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 №5
дисциплина «Технологии web-программирования»
по теме «REST API»

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

Белгород 2020

Лабораторная работа №5

«REST API»

Цель работы:изучить основы разработки API для web-приложений. Разработать REST API для своего проекта.

Задание к лабораторной работе:

1. Изучить структуру формата представления данных JSON.
2. Изучить типы запросов к API: HEAD, GET, POST, PUT, DELETE.
3. Спроектировать и реализовать собственное REST API (Получение, создание, изменение и удаление каких-либо объектов).
4. В отчёт необходимо предоставить документацию к использованию методов. (Либо словесным описание, либо через Swagger).

Ход работы

В качестве СУБД был использовал PostgreSQL развернутый в Docker контейнере.

Для реализации REST API была использована библиотека Django REST Framework.

Были описаны отображения и сериализаторы для таблиц Thing, Storage, Thing Instance, Thing Category, Section, и настроены URL. (код см. в приложении)

При помощи библиотеки *drf-spectacular* была сгенерирована схема API в формате OpenAPI и размещена статическая страница с документацией на сервисе Github Pages. (https://spam25.github.io/web_labs/docs/)

Приложение

Содержимое файла serializersStorages.py

```
from rest_framework import serializers
from storages.models import Storage,Section
from things.models import ThingInstance
from taggit_serializer.serializers import (TagListSerializerField,
                                           TaggitSerializer)

class StorageSerializer(TaggitSerializer,serializers.ModelSerializer,):
    tags = TagListSerializerField()
    sections = serializers.PrimaryKeyRelatedField(many=True, queryset=Section.objects.all())
    owner = serializers.ReadOnlyField(source='owner.username')
    class Meta:
        model = Storage
        fields = ('id', 'name','owner','sections','tags')

class SectionSerializer(serializers.ModelSerializer):
    things_in_sections =
    ↪  serializers.PrimaryKeyRelatedField(many=True,queryset=ThingInstance.objects.all())
    class Meta:
        model = Section
        fields = ('id', 'name','storage','things_in_sections')
```

Содержимое файла serializersThings.py

```
from rest_framework import serializers
from things.models import Thing,ThingInstance,ThingCategory
from taggit_serializer.serializers import (TagListSerializerField,
                                           TaggitSerializer)

class ThingSerializer(TaggitSerializer, serializers.ModelSerializer):
    tags = TagListSerializerField()
    things_instances =
    ↪  serializers.PrimaryKeyRelatedField(many=True,queryset=ThingInstance.objects.all())
    class Meta:
        model = Thing
        fields = ('id', 'name','description','things_instances','category','tags')

class ThingInstanceSerializer(serializers.ModelSerializer):
    class Meta:
        model = ThingInstance
        fields = ('id', 'section','type','count')

class ThingCategorySerializer(serializers.ModelSerializer):
    things = serializers.PrimaryKeyRelatedField(many=True,queryset=Thing.objects.all())
    class Meta:
        model = ThingCategory
        fields = ('id', 'name','icon_label','things')
```

Содержимое файла urls.py

```
from django.conf.urls import url, include
from django.urls import path
from storages import views as storages_views
from users import views as users_views
from things import views as things_views
from rest_framework.routers import DefaultRouter
```

```
from rest_framework.urlpatterns import format_suffix_patterns
from drf_spectacular.views import SpectacularAPIView, SpectacularRedocView,
↪  SpectacularSwaggerView

router = DefaultRouter()
router.register(r'storages', storages_views.StorageViewset)
router.register(r'sections', storages_views.SectionViewset)
router.register(r'things', things_views.ThingViewset)
router.register(r'instances', things_views.ThingInstanceViewset)
router.register(r'category', things_views.ThingCategoryViewset)
router.register(r'users', users_views.UserViewset)

urlpatterns = [
    url(r'^api/', include(router.urls)),
    url(r'^auth/',include('djoser.urls')),
    url(r'^auth/',include('djoser.urls.jwt')),
    path('api/schema/', SpectacularAPIView.as_view(), name='schema'),
    path('api/schema/redoc/', SpectacularRedocView.as_view(url_name='schema'), name='redoc'),
]
```

Содержимое файла viewsStorages.py

```
from storages.models import Storage,Section
from storages.serializers import StorageSerializer,SectionSerializer
from storages.permissions import IsOwner
from rest_framework.viewsets import ModelViewSet
from rest_framework.permissions import IsAuthenticated

class StorageViewset(ModelViewSet):
    queryset = Storage.objects.all()
    serializer_class = StorageSerializer

    permission_classes = (IsAuthenticated,IsOwner)

    def perform_create(self, serializer):
        serializer.save(owner=self.request.user)

class SectionViewset(ModelViewSet):
    queryset = Section.objects.all()
    serializer_class = SectionSerializer

    permission_classes = (IsAuthenticated,)
```

Содержимое файла viewsThings.py

```
from things.models import Thing,ThingInstance,ThingCategory
from things.serializers import ThingSerializer,ThingInstanceSerializer,ThingCategorySerializer
from rest_framework.viewsets import ModelViewSet
from rest_framework.permissions import IsAuthenticated

class ThingViewset(ModelViewSet):
    queryset = Thing.objects.all()
    serializer_class = ThingSerializer

    permission_classes = (IsAuthenticated,)


class ThingInstanceViewset(ModelViewSet):
    queryset = ThingInstance.objects.all()
    serializer_class = ThingInstanceSerializer
```

```
    permission_classes = (IsAuthenticated,)

class ThingCategoryViewset(ModelViewSet):
    queryset = ThingCategory.objects.all()
    serializer_class = ThingCategorySerializer

    permission_classes = (IsAuthenticated,)
```