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

Лабораторная работа по теме

«REST API»  
дисциплина:

Технологии web-программирования

Выполнил: ст. группы ВТ-41   
Фаракшин Н. Р.  
Проверил: ст. пр. 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)

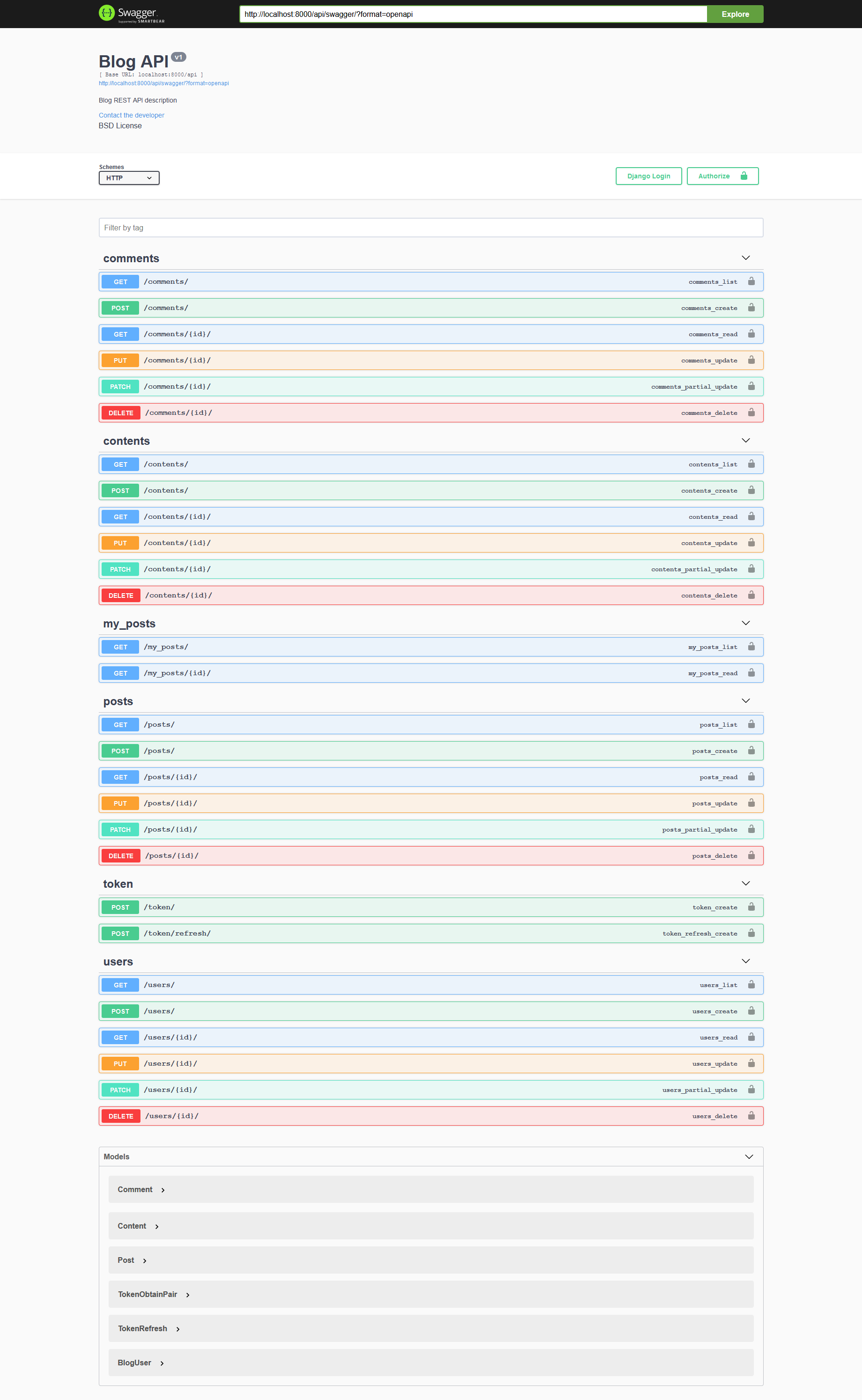
**Выполнение работы**

Для реализации REST API была использована библиотека Django REST Framework. В качестве вспомогательных библиотек использовались библиотеки Django-filters для обеспечения фильтрации запросов согласно параметрам и drf-simple-jwt для предоставления возможности аутентификации с помощью JWT токенов.

В ходе выполнения описаны классы сериализаторов моделей, использующие в качестве основы стандартный класс ModelSerializer, а также отображения (views), основанные на стандартных классах ModelViewSet. Поля моделей, использующиеся в API, ограничены с помощью полей fields и excluded данных классов.

Документация к API сгенерирована с помощью библиотеки drf-yasg. Результатом работы является описание API в формате Swagger. Получившееся описание представлено с помощью веб-клиента Swagger UI.

**Описание API**

****

**Описание моделей, использующихся в API**

****

**Приложение**

Исходный код файлов serializers.py и views.py, реализующих сериализаторы и отображения, соответственно. Исходный код файла urls.py, управляющего структурой URL-адресов в проекте.

**serializers.py**

from rest\_framework import serializers

from logic.models import Post, BlogUser, Comment, PostPiece

class PostSerializer(serializers.ModelSerializer):

    creation\_time = serializers.DateTimeField(format='%D.%M.%Y')

    class Meta:

        model = Post

        fields = "\_\_all\_\_"

class ContentSerializer(serializers.ModelSerializer):

    class Meta:

        model = PostPiece

        exclude = ['order', 'parent']

class BlogUserSerializer(serializers.ModelSerializer):

    class Meta:

        model = BlogUser

        fields = ['id', 'username', 'email', 'profile\_pic']

class CommentSerializer(serializers.ModelSerializer):

    creation\_time = serializers.DateTimeField(format='%D.%M.%Y')

    class Meta:

        model = Comment

        fields = "\_\_all\_\_"

**views.py**

import json

from django.http import HttpResponse

from django\_filters.rest\_framework import DjangoFilterBackend

from rest\_framework import viewsets

from rest\_framework.permissions import IsAuthenticated

from api.serializers import BlogUserSerializer, CommentSerializer, PostSerializer, ContentSerializer

from logic.models import Post, BlogUser, Comment, PostPiece

class MyPostsROViewSet(viewsets.ReadOnlyModelViewSet):

    serializer\_class = PostSerializer

    permission\_classes = [IsAuthenticated]

    def get\_queryset(self):

        return Post.objects.filter(user=self.request.user)

class ContentViewSet(viewsets.ModelViewSet):

    serializer\_class = ContentSerializer

    queryset = PostPiece.objects.order\_by('order')

    filterset\_fields = ['parent']

class PostViewSet(viewsets.ModelViewSet):

    queryset = Post.objects.order\_by('creation\_time')

    serializer\_class = PostSerializer

    filterset\_fields = ['user']

class BlogUserViewSet(viewsets.ModelViewSet):

    queryset = BlogUser.objects.all()

    serializer\_class = BlogUserSerializer

class CommentViewSet(viewsets.ModelViewSet):

    queryset = Comment.objects.all()

    serializer\_class = CommentSerializer

    filterset\_fields = ['post', 'user']

**urls.py**

from django.conf.urls import url

from django.urls import path

from rest\_framework import routers, permissions

from drf\_yasg.views import get\_schema\_view

from drf\_yasg import openapi

from rest\_framework\_simplejwt.views import TokenObtainPairView, TokenRefreshView

from api.views import BlogUserViewSet, CommentViewSet, PostViewSet, ContentViewSet, get\_user\_id, MyPostsROViewSet

schema\_view = get\_schema\_view(

   openapi.Info(

      title="Blog API",

      default\_version='v1',

      description="Blog REST API description",

      contact=openapi.Contact(email="contact@snippets.local"),

      license=openapi.License(name="BSD License"),

   ),

   public=True,

   permission\_classes=[permissions.AllowAny],

)

router = routers.SimpleRouter()

router.register(r'contents', ContentViewSet, basename='contents')

router.register(r'posts', PostViewSet, basename='posts')

router.register(r'users', BlogUserViewSet, basename='users')

router.register(r'comments', CommentViewSet, basename='comments')

router.register(r'my\_posts', MyPostsROViewSet, basename="my\_posts")

urlpatterns = [

   path('token/',

         TokenObtainPairView.as\_view(),

         name='token\_obtain\_pair'),

   path('token/refresh/',

         TokenRefreshView.as\_view(),

         name='token\_refresh'),

   url(r'^swagger(?P<format>\.json|\.yaml)$', schema\_view.without\_ui(cache\_timeout=0), name='schema-json'),

   url(r'^swagger/$', schema\_view.with\_ui('swagger', cache\_timeout=0), name='schema-swagger-ui'),

   url(r'^redoc/$', schema\_view.with\_ui('redoc', cache\_timeout=0), name='schema-redoc'),

]

urlpatterns += router.urls