Факультатив ФКН

РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ НА БАЗЕ ПЛАТФОРМЫ

Преподаватель
Папулин Сергей Юрьевич (***papulin_hse@mail.ru***)

Москва, 2018

Django REST Framework

```
pip install djangorestframework
```

Основные функции

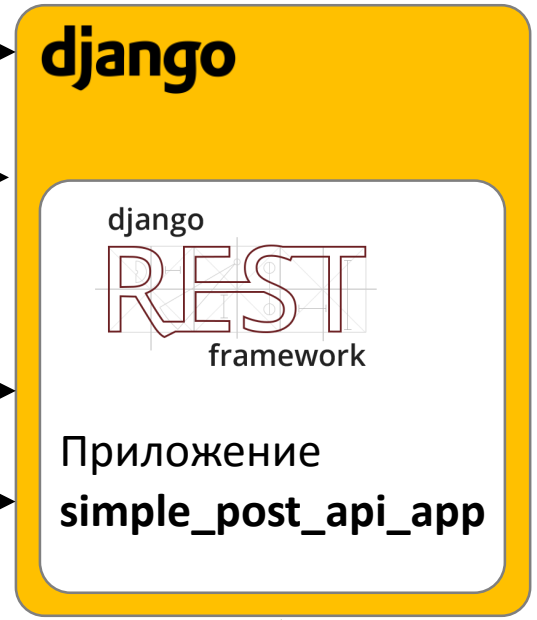
Терминал

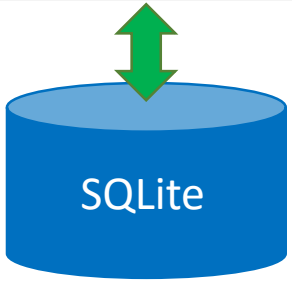
Создать пользователя (регистрация)

Аутентификация и получение токена

Показать список постов

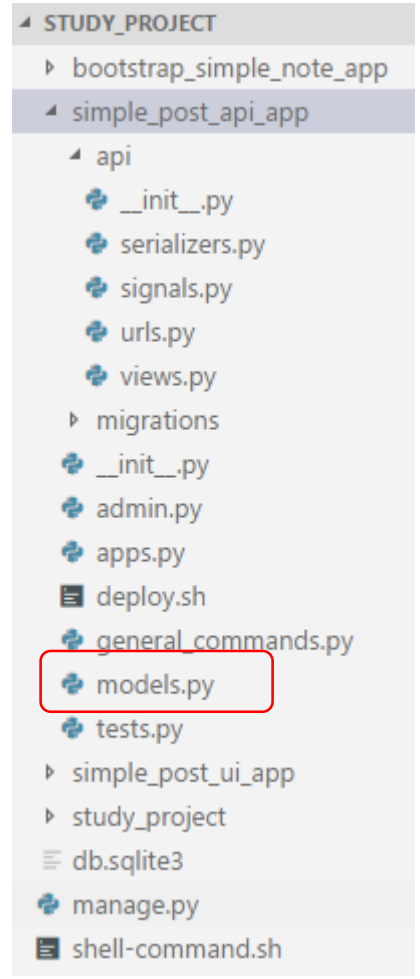
Создать/изменить/удалить пост





Модель данных

```
class SimplePost(models.Model):

    # The id field will be auto-created

    header = models.CharField(max_length=200, blank=False, null=False,
                              verbose_name=_("Header"))
    content = models.TextField(blank=False)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    location = models.CharField(max_length=200, blank=True, null=True)
```



Регистрация пользователя

Регистрация

Терминал

Создать пользователя (регистрация)

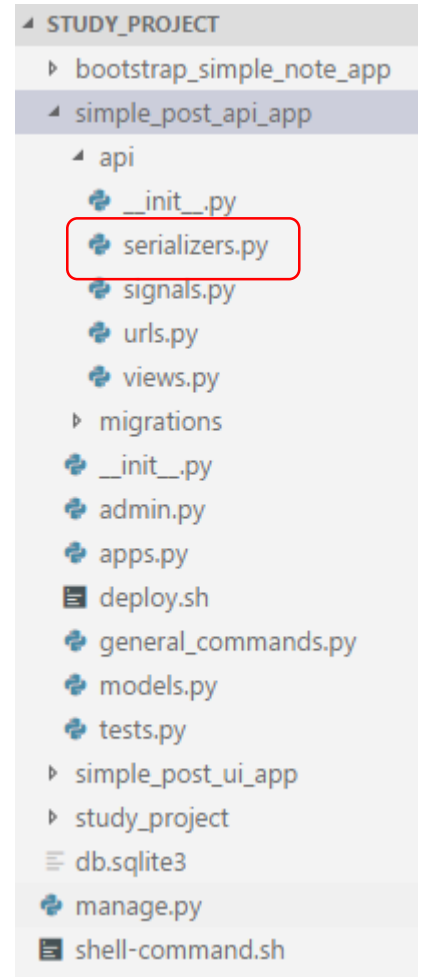
```
POST
127.0.0.1:8000/simple-post-api/api/v1/sign-up/
        {
            "email": "ivan@ivanov.ru",
            "first_name": "ivan",
            "last_name": "ivanov",
            "password": "1",
            "username": "user1"
        }
```

Ответ

```
HTTP/1.1 200 OK
{
        "status": "OK"
}
```

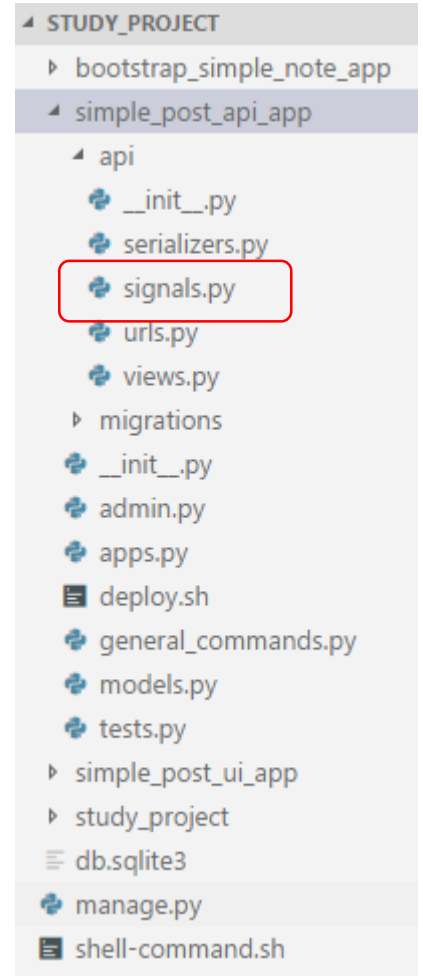
Регистрация. Serializer

```python
class SignUpSerializer(serializers.ModelSerializer):

    class Meta:

        model = User
        fields = ("id", "first_name", "last_name", "username", "email",
        "password")
        write_only_fields = ("password",)
        read_only_fields = ("id",)

    def create(self, validated_data):
        user = User(first_name=validated_data["first_name"],
        last_name=validated_data["last_name"],
        username=validated_data["username"],
        email=validated_data["email"],)

        user.set_password(validated_data["password"])
        user.save()

        return user
```



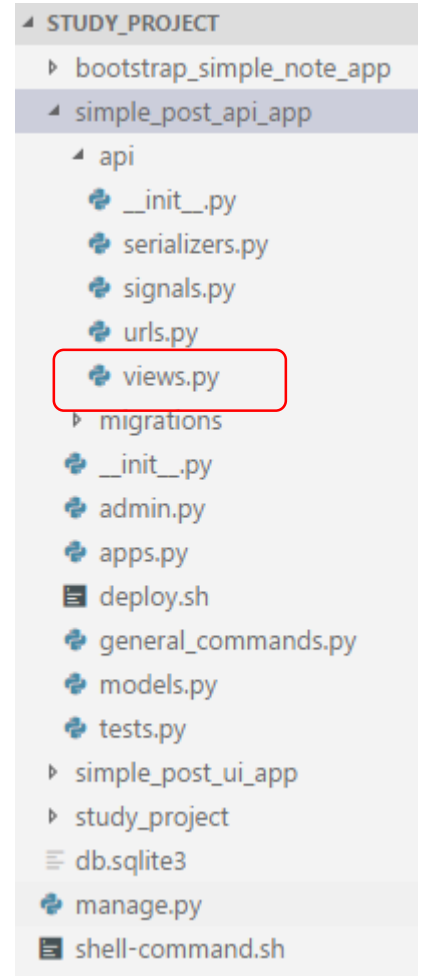
Регистрация. Создание токена. Signal

```python
@receiver(post_save, sender=settings.AUTH_USER_MODEL)
def create_auth_token(sender, instance=None, created=False, **kwargs):
    if created:
        Token.objects.create(user=instance)
```



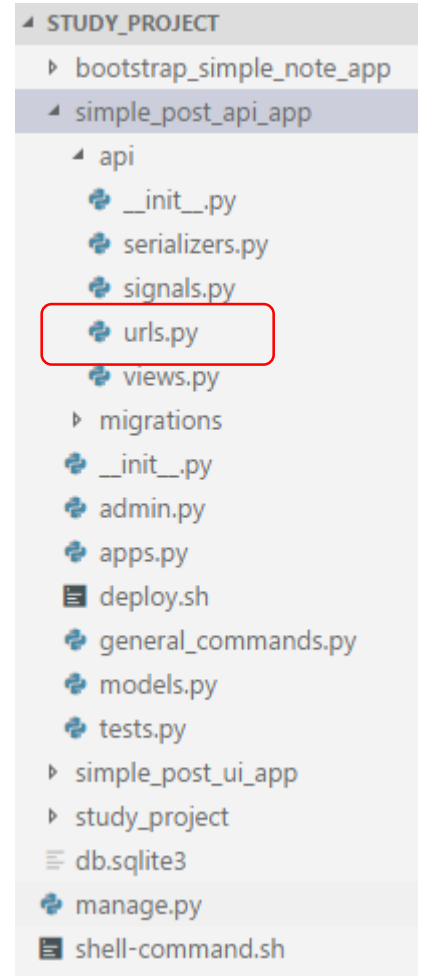
Регистрация. View

```python
@api_view(["POST"])
def sign_up_view(request):

    sign_up_ser = SignUpSerializer(data=request.data)

    if sign_up_ser.is_valid():

        user_obj = sign_up_ser.save()

        response_message = {"status": "OK"}

    else:
        return Response(sign_up_ser.errors)

    return Response(response_message)
```



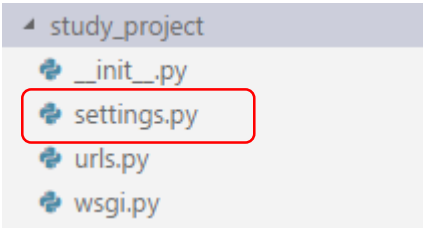
Регистрация. Url

```
url(r'^sign-up/$', sign_up_view),
```



Настройки приложения. Для мобильных веб-приложений

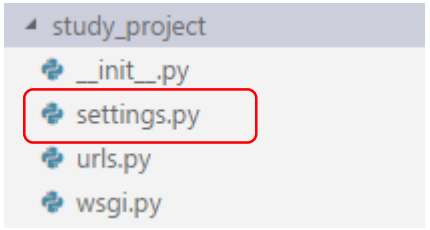
```
pip install django-cors-headers
```




```
INSTALLED_APPS = [
        'corsheaders', # CORS
]

MIDDLEWARE = [
        'corsheaders.middleware.CorsMiddleware', # CORS
]

CORS_ORIGIN_WHITELIST = (
        '127.0.0.1:8000'
)

CORS_ALLOW_METHODS = (
    'DELETE',
    'GET',
    'OPTIONS',
    'PATCH',
    'POST',
    'PUT',
)
```

Настройки приложения

settings.py

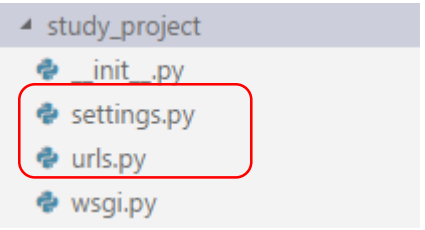



```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'corsheaders', # CORS
    'rest_framework',
    'rest_framework.authtoken', # for a simple token-based HTTP Authentication
    "simple_post_ui_app",
    "simple_post_api_app", # REST API
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.locale.LocaleMiddleware',
    'corsheaders.middleware.CorsMiddleware', # CORS
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

Настройки приложения

settings.py

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
    'rest_framework.authentication.TokenAuthentication',
    #'rest_framework.authentication.SessionAuthentication',
    )
}
```



urls.py

```
url(r'^simple-post-api/api/v1/', include('simple_post_api_app.api.urls')),
```

```
python manage.py makemigrations simple_post_api_app

python manage.py migrate simple_post_api_app

python manage.py runserver
```

Регистрация. Пример

```
sudo apt install httpie
```

```
http --print HBhb POST 127.0.0.1:8000/simple-post-api/api/v1/sign-up/ username=user1 password=1 first_name=ivan
last_name=ivanov email=ivan@ivanov.ru
```

Запрос

```
POST /simple-post-api/api/v1/sign-up/ HTTP/1.1
Accept: application/json
Content-Type: application/json
Host: 127.0.0.1:8000
User-Agent: HTTPie/0.9.2

{
"email": "ivan@ivanov.ru",
"first_name": "ivan",
"last_name": "ivanov",
"password": "1",
"username": "user1"
}
```

Ответ

```
HTTP/1.1 200 OK

{
"status": "OK"
}
```

Аутентификация и получение токена

Аутентификация

Терминал

Аутентификация

```
POST
127.0.0.1:8000/simple-post-api/api/v1/sign-in/
            {
                "password": "1",
                "username": "user1"
            }
```

Ответ

```
HTTP/1.1 200 OK
{
            "token": "c20eece335018fbaae2c91427d60d6e55351c5a7"
}
```

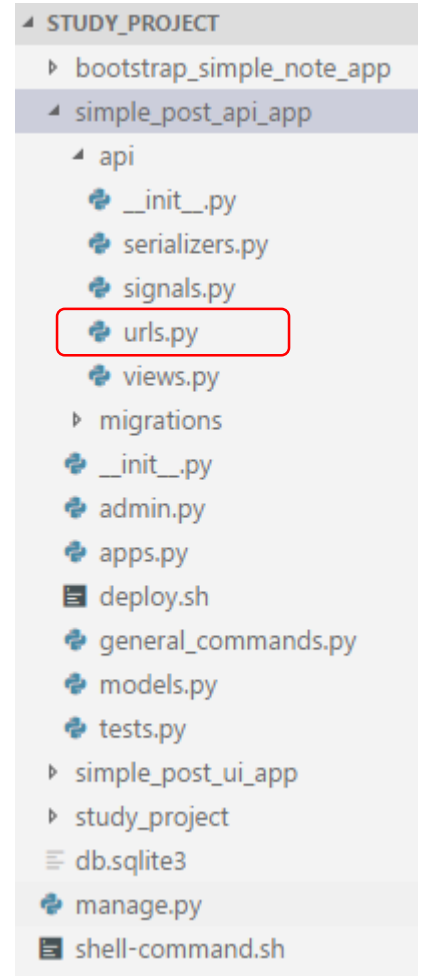
django

django REST framework

Приложение **simple_post_api_app**

Аутентификация. Url

```
from rest_framework.authtoken import views

urlpatterns = [
        url(r'^sign-in/$', views.obtain_auth_token),
```



Аутентификация. Пример

```
http --print HBhb POST 127.0.0.1:8000/simple-post-api/api/v1/sign-in/ username=user1 password=1
```

Запрос

```
POST /simple-post-api/api/v1/sign-in/ HTTP/1.1
Accept: application/json
Content-Type: application/json
Host: 127.0.0.1:8000
User-Agent: HTTPie/0.9.2

{
    "password": "1",
    "username": "user1"
}
```

Ответ

```
HTTP/1.1 200 OK

{
"token": "c20eece335018fbaae2c91427d60d6e55351c5a7"
}
```

Создание поста

Создание поста

Терминал

Создание поста

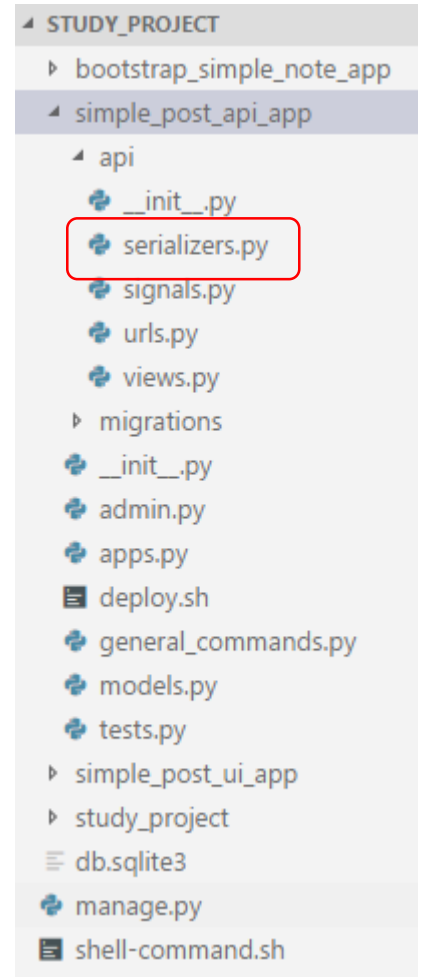
```
POST
127.0.0.1:8000/simple-post-api/api/v1/simple-posts/
      Authorization: Token xxxxxxxxxxx

      {
            "content": "content of my post",
            "header": "my post",
            "location": "Москва"
      }
```

Ответ

```
      HTTP/1.1 201 Created

      {
            "content": "content of my post",
            "created": "2018-05-21T17:31:47.799488Z",
            "header": "my post",
            "id": 1,
            "location": "Москва",
            "updated": "2018-05-21T17:31:47.799557Z"
      }
```

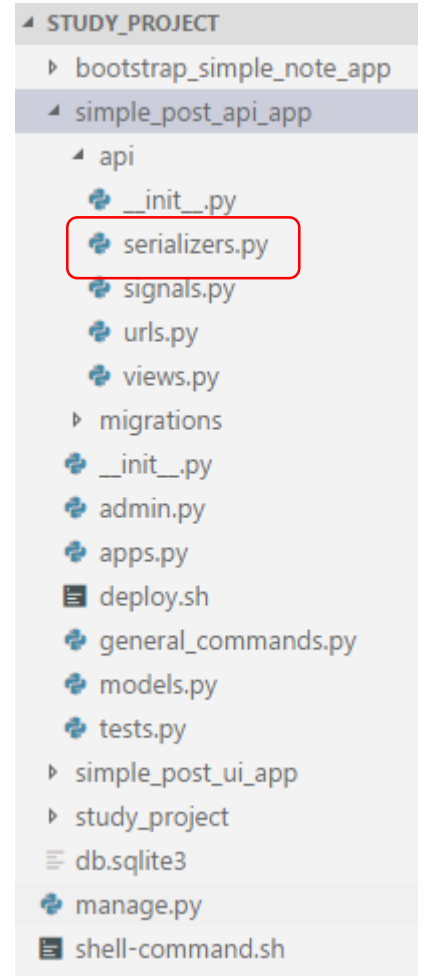
Создание поста. Serializer

```python
# Approach 1 with Serializer
class SimplePostSerializer(serializers.Serializer):

    id = serializers.IntegerField(read_only=True)
    header = serializers.CharField(max_length=200,)
    content = serializers.CharField()
    location = serializers.CharField()
    created = serializers.DateTimeField(read_only=True)
    updated = serializers.DateTimeField(read_only=True)

    def create(self, validated_data):
        return SimplePost.objects.create(**validated_data)

    def update(self, instance, validated_data):

        instance.header = validated_data.get("header", instance.header)
        instance.content = validated_data.get("content", instance.content)
        instance.location = validated_data.get("location", instance.content)

        instance.save()

        return instance
```



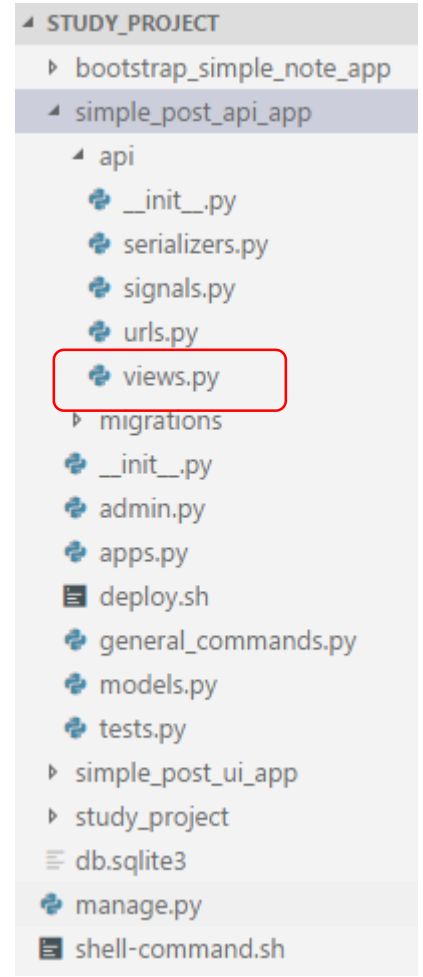
Создание поста. Serializer

```python
# Approach 2 with ModelSerializer
class SimplePostSerializer(serializers.ModelSerializer):

    class Meta:
        model = SimplePost
        fields = (
            "id",
            "header",
            "content",
            "created",
            "update",
        )
        read_only_fields = (
            "id",
            "created"
        )
```



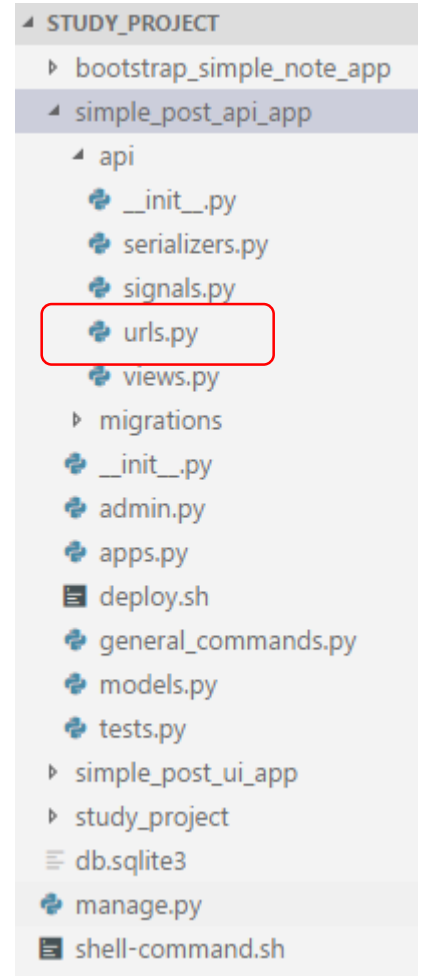
Создание поста. View

```python
@api_view(["GET", "POST"])
@authentication_classes((TokenAuthentication,)) #SessionAuthentication
@permission_classes((IsAuthenticated,))
def simple_post_list_view(request):

    # Retrieve a list
    if request.method == "GET":
        post_objs = SimplePost.objects.all()
        post_ser = SimplePostSerializer(post_objs, many=True)
        return Response(post_ser.data)

    # Create a simple post
    elif request.method == "POST":
        post_ser = SimplePostSerializer(data=request.data)
        if post_ser.is_valid():
            post_ser.save()
            return Response(post_ser.data, status=status.HTTP_201_CREATED)
        return Response(post_ser.errors, status=400)
```



Создание поста. Url

```
url(r'^simple-posts/$', simple_post_list_view),
```



Создание поста. Пример

```
http --print HBhb POST 127.0.0.1:8000/simple-post-api/api/v1/simple-posts/ "Authorization: Token
c20eece335018fbaae2c91427d60d6e55351c5a7" header="my post" content="content of my post" location=Москва
```

Запрос

```
POST /simple-post-api/api/v1/simple-posts/ HTTP/1.1
Accept: application/json
Authorization: Token c20eece335018fbaae2c91427d60d6e55351c5a7
Content-Type: application/json
Host: 127.0.0.1:8000
User-Agent: HTTPie/0.9.2

{
    "content": "content of my post",
    "header": "my post",
    "location": "Москва"
}
```

Ответ

```
HTTP/1.1 201 Created

{
    "content": "content of my post",
    "created": "2018-05-21T17:31:47.799488Z",
    "header": "my post",
    "id": 1,
    "location": "Москва",
    "updated": "2018-05-21T17:31:47.799557Z"
}
```

Получение списка постов

Получение списка постов

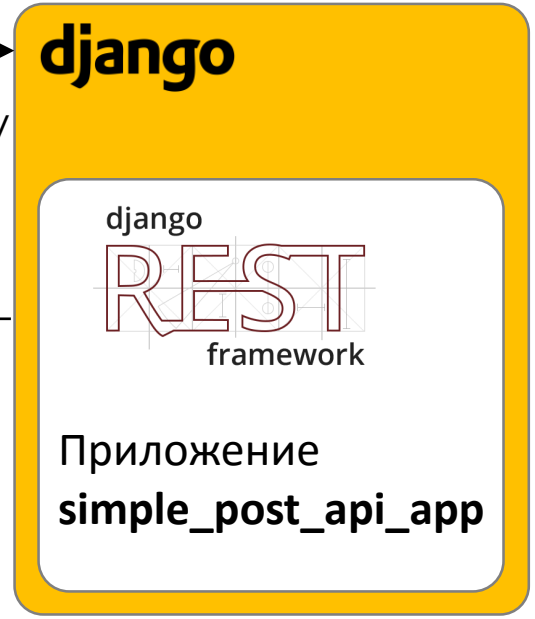
Терминал

Получение списка постов

```
GET
127.0.0.1:8000/simple-post-api/api/v1/simple-posts/
     Authorization: Token xxxxxxxxxxx
```

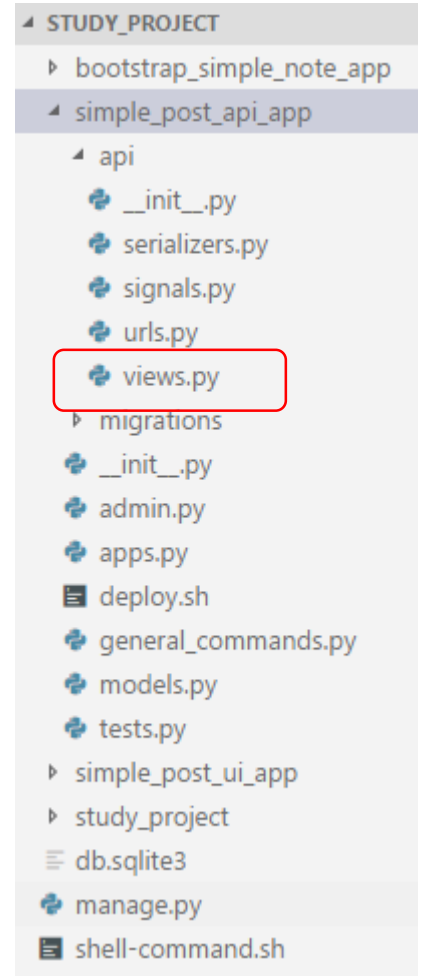
Ответ


```
{
     "content": "content of my post",
     "created": "2018-05-21T17:31:47.799488Z",
     "header": "my post",
     "id": 1,
     "location": "Москва",
     "updated": "2018-05-21T17:31:47.799557Z"
},
{
     "content": "content of my post 2",
     "created": "2018-05-21T17:33:53.806552Z",
     "header": "my post 2",
     "id": 2,
     "location": "Москва",
     "updated": "2018-05-21T17:33:53.806591Z"
}
```

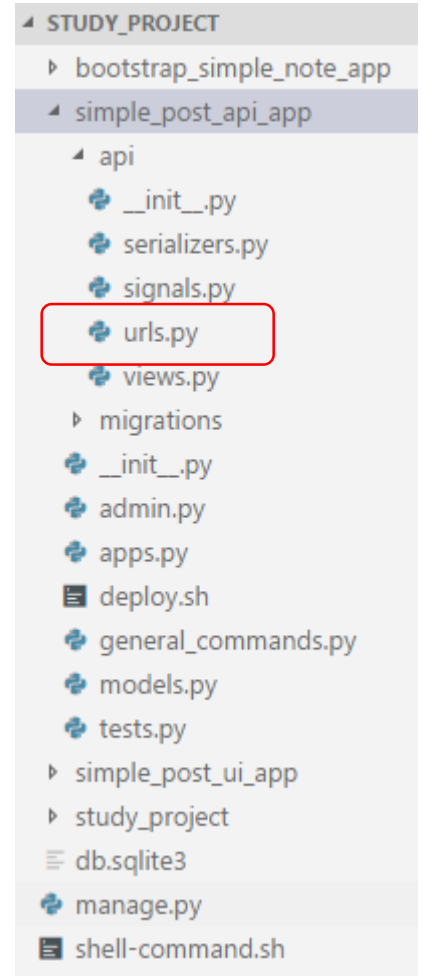
Получение списка постов. View

```python
@api_view(["GET", "POST"])
@authentication_classes((TokenAuthentication,)) #SessionAuthentication
@permission_classes((IsAuthenticated,))
def simple_post_list_view(request):

    # Retrieve a list
    if request.method == "GET":
        post_objs = SimplePost.objects.all()
        post_ser = SimplePostSerializer(post_objs, many=True)
        return Response(post_ser.data)

    # Create a simple post
    elif request.method == "POST":
        post_ser = SimplePostSerializer(data=request.data)
        if post_ser.is_valid():
            post_ser.save()
            return Response(post_ser.data, status=status.HTTP_201_CREATED)
        return Response(post_ser.errors, status=400)
```



Получение списка постов. Url

```
url(r'^simple-posts/$', simple_post_list_view),
```



Получение списка постов. Пример

```
http --print HBhb GET 127.0.0.1:8000/simple-post-api/api/v1/simple-posts/ "Authorization: Token
c20eece335018fbaae2c91427d60d6e55351c5a7"
```

Запрос

```
GET /simple-post-api/api/v1/simple-posts/ HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Authorization: Token c20eece335018fbaae2c91427d60d6e55351c5a7
Connection: keep-alive
Host: 127.0.0.1:8000
User-Agent: HTTPie/0.9.2
```

Ответ


```
[
    {
        "content": "content of my post",
        "created": "2018-05-21T17:31:47.799488Z",
        "header": "my post",
        "id": 1,
        "location": "Москва",
        "updated": "2018-05-21T17:31:47.799557Z"
    },
    {
        "content": "content of my post 2",
        "created": "2018-05-21T17:33:53.806552Z",
        "header": "my post 2",
        "id": 2,
        "location": "Москва",
        "updated": "2018-05-21T17:33:53.806591Z"
    }
]
```

Получение поста

Получение поста по id

Терминал

Получение поста

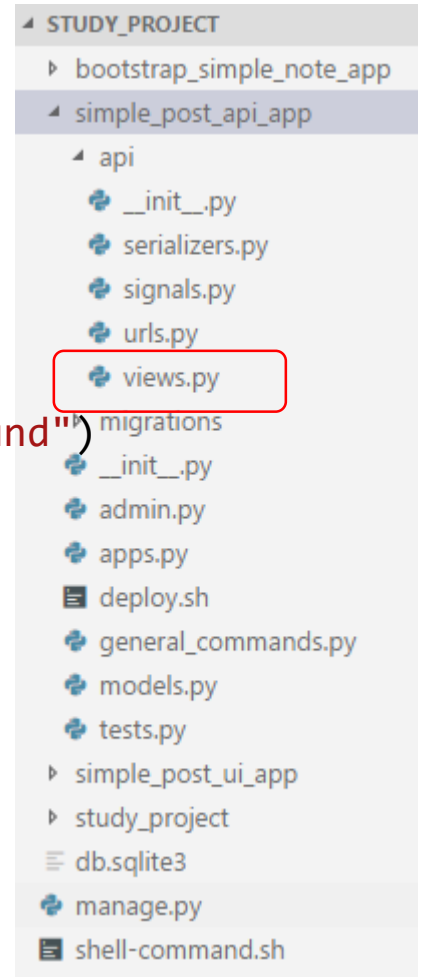
```
GET
127.0.0.1:8000/simple-post-api/api/v1/simple-posts/2/
     Authorization: Token xxxxxxxxxxx
```

Ответ

```
{
    "content": "content of my post 2",
    "created": "2018-05-21T17:33:53.806552Z",
    "header": "my post 2",
    "id": 2,
    "location": "Москва",
    "updated": "2018-05-21T17:33:53.806591Z"
}
```

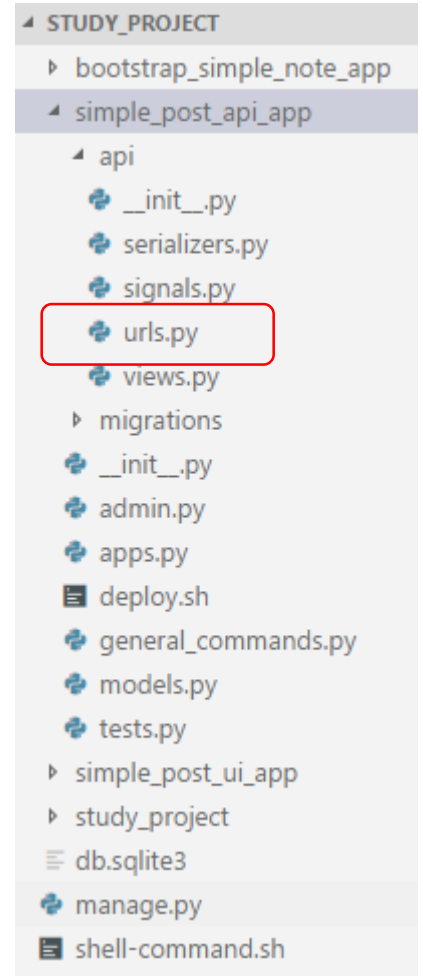
Получение поста по id. View

```python
@api_view(["GET", "PUT", "DELETE"])
@permission_classes((IsAuthenticated,))
def simple_post_detail_view(request, pk=None):

    def get_object(pk=None):
        try:
                obj = SimplePost.objects.get(pk=pk)
        except SimplePost.DoesNotExist:
                raise NotFound(detail="There is no such post", code="not_found")
        return obj

    if pk is None:
        raise ParseError(detail=None, code=None)

    # Get post details
    if request.method == "GET":

        post_obj = get_object(pk)
        post_ser = SimplePostSerializer(post_obj)

        return Response(post_ser.data)

    raise MethodNotAllowed(request.method, detail=None, code=None)
```



Получение поста по id. Url

```
url(r'^simple-posts/(?P<pk>\d+)/$', simple_post_detail_view),
```



Получение поста по id. Пример

```
http --print HBhb GET 127.0.0.1:8000/simple-post-api/api/v1/simple-posts/2/ "Authorization: Token c20eece335018fbaae2c91427d60d6e55351c5a7"
```

Запрос

```
GET /simple-post-api/api/v1/simple-posts/2/ HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Authorization: Token c20eece335018fbaae2c91427d60d6e55351c5a7
Connection: keep-alive
Host: 127.0.0.1:8000
User-Agent: HTTPie/0.9.2
```

Ответ

```
HTTP/1.1 200 OK
Allow: GET, OPTIONS, PUT, DELETE

{
    "content": "content of my post 2",
    "created": "2018-05-21T17:33:53.806552Z",
    "header": "my post 2",
    "id": 2,
    "location": "Москва",
    "updated": "2018-05-21T17:33:53.806591Z"
}
```

Изменение поста

Изменение поста

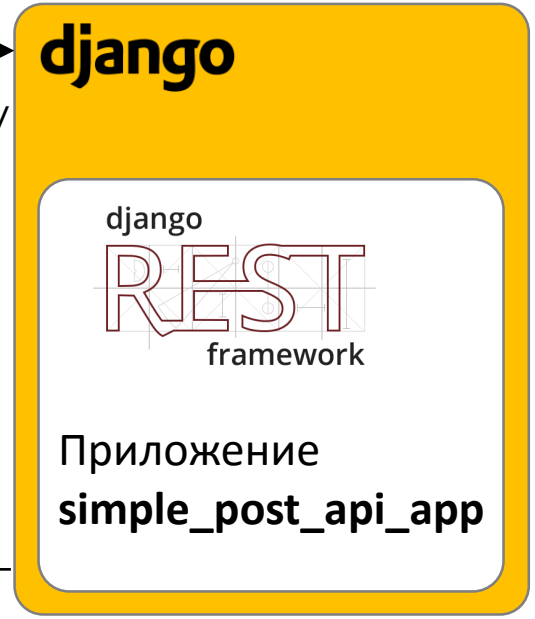
Терминал

Изменение поста

```
PUT
127.0.0.1:8000/simple-post-api/api/v1/simple-posts/2/

      Authorization: Token xxxxxxxxxxx
      {
            "content": "new content",
            "header": "my post new",
            "id": "2",
            "location": "Москва"
      }
```

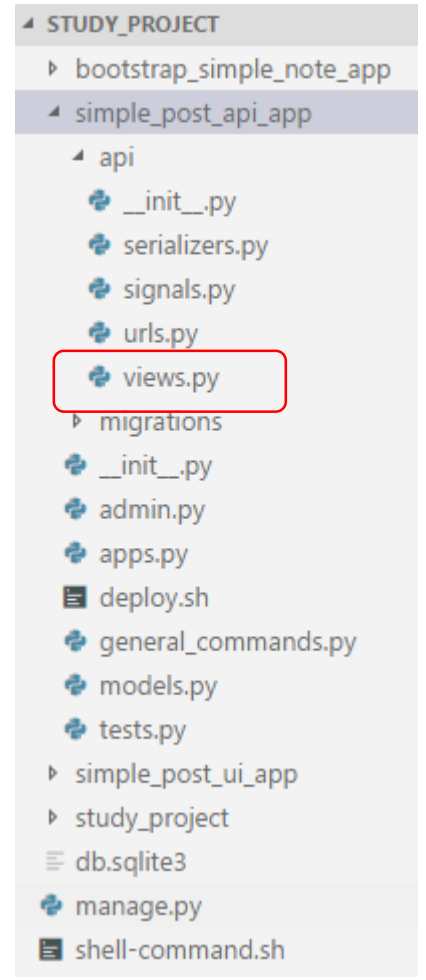
Ответ

```
HTTP/1.1 200 OK
```



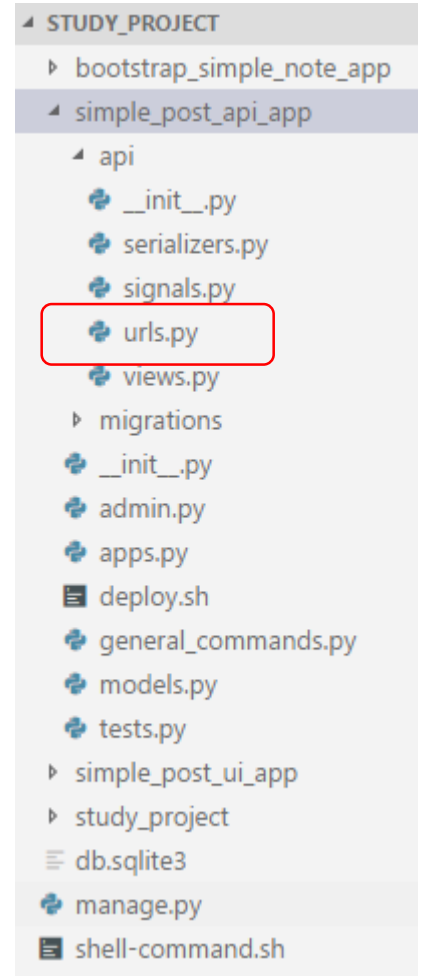
Изменение поста. View

```
@api_view(["GET", "PUT", "DELETE"])
@permission_classes((IsAuthenticated,))
def simple_post_detail_view(request, pk=None):

    def get_object(pk=None):
        try:
                obj = SimplePost.objects.get(pk=pk)
        except SimplePost.DoesNotExist:
                raise NotFound(detail="There is no such post", code="not_found")
        return obj

    if pk is None:
        raise ParseError(detail=None, code=None)

    # Update a post
    elif request.method == "PUT":

        post_obj = get_object(pk)
        post_ser = SimplePostSerializer(data=request.data, instance=post_obj)

        if post_ser.is_valid():
                post_ser.save()
                return Response(status=status.HTTP_200_OK)

        return Response(post_ser.errors, status=status.HTTP_417_EXPECTATION_FAILED)

    raise MethodNotAllowed(request.method, detail=None, code=None)
```



Изменение поста. Url

```
url(r'^simple-posts/(?P<pk>\d+)/$', simple_post_detail_view),
```



Изменение поста. Пример

```
http --print HBhb PUT 127.0.0.1:8000/simple-post-api/api/v1/simple-posts/2/ "Authorization: Token
c20eece335018fbaae2c91427d60d6e55351c5a7" id=2 header="my post new" content="new content" location="Москва"
```

Запрос

```
PUT /simple-post-api/api/v1/simple-posts/2/ HTTP/1.1
Accept: application/json
Authorization: Token
c20eece335018fbaae2c91427d60d6e55351c5a7
Content-Type: application/json
Host: 127.0.0.1:8000
User-Agent: HTTPie/0.9.2

{
    "content": "new content",
    "header": "my post new",
    "id": "2",
    "location": "Москва"
}
```

Ответ

```
HTTP/1.1 200 OK
Allow: GET, OPTIONS, PUT, DELETE
Content-Language: en
Content-Length: 0
Date: Mon, 21 May 2018 17:37:54 GMT
Server: WSGIServer/0.2 CPython/3.5.2
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
```

Удаление поста

Удаление поста

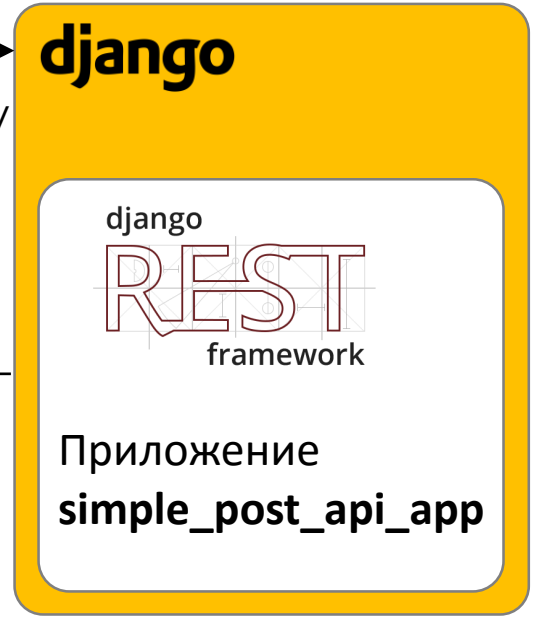
Терминал

Удаление поста

```
DELETE
127.0.0.1:8000/simple-post-api/api/v1/simple-posts/2/

      Authorization: Token xxxxxxxxxxx
```

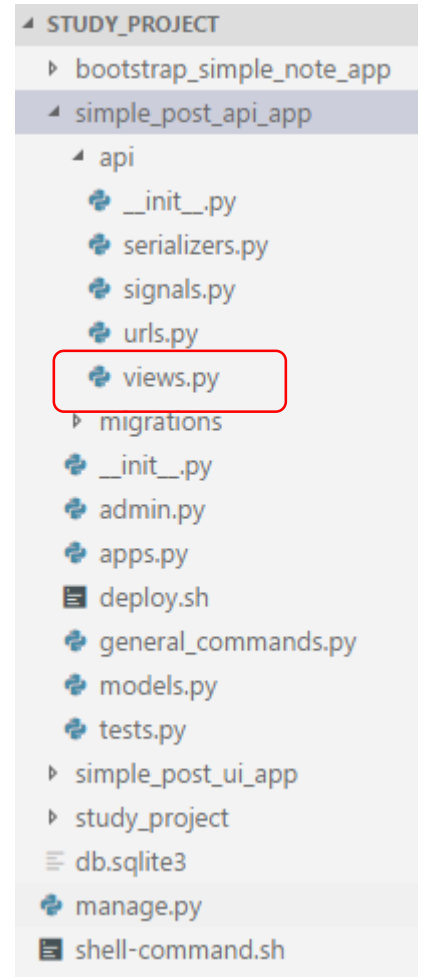
Ответ

```
HTTP/1.1 204 No Content
```



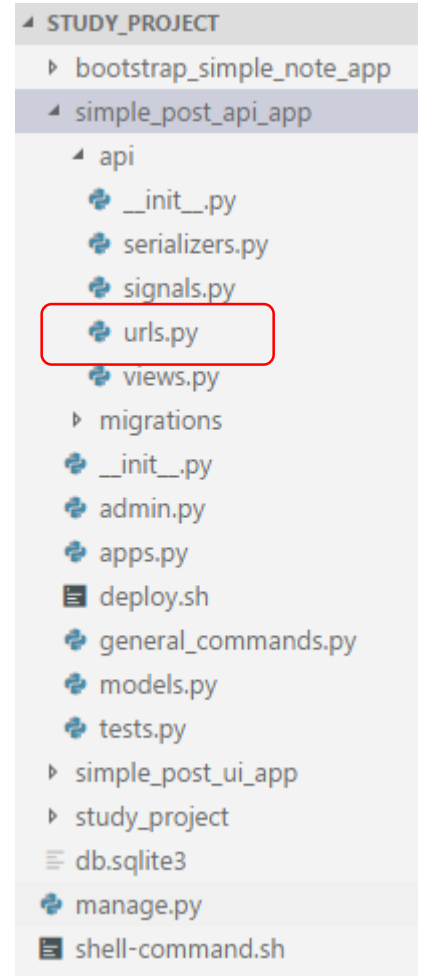
Удаление поста. View

```python
@api_view(["GET", "PUT", "DELETE"])
@permission_classes((IsAuthenticated,))
def simple_post_detail_view(request, pk=None):

    def get_object(pk=None):
        try:
                obj = SimplePost.objects.get(pk=pk)
        except SimplePost.DoesNotExist:
                raise NotFound(detail="There is no such post",
        code="not_found")
        return obj

    if pk is None:
        raise ParseError(detail=None, code=None)

    # Delete a post
    elif request.method == "DELETE":
        get_object(pk).delete()
        return Response(status=status.HTTP_204_NO_CONTENT)

    raise MethodNotAllowed(request.method, detail=None, code=None)
```



Удаление поста. Url

```
url(r'^simple-posts/(?P<pk>\d+)/$', simple_post_detail_view),
```



Удаление поста. Пример

```
http --print HBhb DELETE 127.0.0.1:8000/simple-post-api/api/v1/simple-posts/2/ "Authorization: Token
c20eece335018fbaae2c91427d60d6e55351c5a7"
```

Запрос

```
DELETE /simple-post-api/api/v1/simple-posts/2/ HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Authorization: Token
c20eece335018fbaae2c91427d60d6e55351c5a7
Connection: keep-alive
Content-Length: 0
Host: 127.0.0.1:8000
User-Agent: HTTPie/0.9.2
```

Ответ

```
HTTP/1.1 204 No Content
Allow: GET, OPTIONS, PUT, DELETE
Content-Language: en
Content-Length: 0
Date: Mon, 21 May 2018 17:40:58 GMT
Server: WSGIServer/0.2 CPython/3.5.2
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
```

Источники

https://docs.djangoproject.com/en/2.0/

http://www.django-rest-framework.org/