Getting started with Django restframework

Installation

```
virtualenv ./venv --python=python3.8
. ./venv/bin/activate
pip install django djangorestframework
django-admin startproject projectname
cd projectname
django-admin startapp appname
python manage.py migrate
python manage.py runserver
```

Configure project

Create super user

```
python manage.py runserver createsuperuser
```

FILE `projectname/projectname/settings.py`

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'rest_framework.authtoken',
    'db'
]
```

Before `AUTH_PASSWORD_VALIDATORS`

```
REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ]
}
```

Cors headers

Install the cors-headers package with

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

Adds to your installed apps

```
INSTALLED_APPS = [
    ...
    'corsheaders',
    ...
]
```

Add on your MIDDLEWARE remember to add as being the first in the list

```
MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',
    ...
]
```

Before installed apps put this configuration for anyone to access

```
CORS_ORIGIN_ALLOW_ALL=True
```

Or a list of known locations, this is more secure

```
CORS_ORIGIN_ALLOW_ALL=False

CORS_ORIGIN_WHITELIST = [
    'http://localhost:3000'
]
```

FILE `projectname/projectname/urls.py`

```
from django.contrib import admin
from django.urls import path
from django.conf.urls import include
from rest_framework.authtoken.views import obtain_auth_token
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('appname/', include('appname.urls')),
    path('auth/', obtain_auth_token)
]
```

Configure app

FILE `projectname/appname/admin.py`

```
from django.contrib import admin
from .models import MyModel

admin.site.register(MyModel)
```

FILE `projectname/appname/models.py`

```
from django.db import models

class MyModel(models.Model):

    title = models.CharField(max_length=32)
    description = models.TextField(max_length=360)

    # Calculcated field - not exposed as an endpoint but rather
    # as another field in the database which is calculated upon
    # request
    def title_length(self):
        return len(self.title)
```

FILE `projectname/appname/serializers.py`

```
from rest_framework import serializers

from .models import MyModel


class MyModelSerializer(serializers.ModelSerializer):

    class Meta:
        model = MyModel
        fields = ('titel', 'description', 'title_length')
```

FILE `projectname/appname/views.py`

```
from rest_framework import viewsets
from rest_framework.authentication import TokenAuthentication
from rest_framework.permissions import IsAuthenticated # already default
in settings

from .models import MyModel
from .serializers import MyModelSerializer


class MyModelViewSet(viewsets.ModelViewSet):
    queryset = MyModel.objects.all()
    serializer_class = MyModelSerializer
    authentication_classes = (TokenAuthentication, )
    permission_classes = (IsAuthenticated, )
```

FILE `projectname/appname/urls.py`

```
from django.urls import path
from rest_framework import routers
from django.conf.urls import include
from .views import MyModelViewset

router = routers.DefaultRouter()
router.register('model', MyModelViewset)

urlpatterns = [
    path('', include(router.urls))
]
```

Database migrations

```
python manage.py makemigrations
python manage.py migrate
```

Examples

Models: validators and User class

```
from django.db import models
from django.contrib.auth.models import User
from django.core.validators import MaxValueValidator, MinValueValidator
```

```
class Movie(models.Model):

    title = models.CharField(max_length=32)
    description = models.TextField(max_length=360)

    def no_of_ratings(self):
        ratings = Rating.objects.filter(movie=self)
        return len(ratings)

    def avg_rating(self):
        ratings = Rating.objects.filter(movie=self)
        stars = [rating.stars for rating in ratings]
        if len(ratings) > 0:
            return sum(stars) / len(stars)
        else:
            return None


class Rating(models.Model):

    movie = models.ForeignKey(Movie, on_delete=models.CASCADE)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    stars = models.IntegerField(validators=(MinValueValidator(1),
MaxValueValidator(5)))

    class Meta:
        unique_together = (('user', 'movie'),)
        index_together = (('user', 'movie'),)
```

Serializers: User Serializer, Token

```
from rest_framework import serializers
from django.contrib.auth.models import User
from rest_framework.authtoken.models import Token

from .models import Movie, Rating, User


class UserSerializer(serializers.ModelSerializer):

    class Meta:
        model = User
        fields = ('id', 'username', 'password')
        extra_kwargs = {
            'password': {
                'write_only': True,
                'required': True
                }
            }
    def create(self, validated_data):
        user = User.objects.create_user(**validated_data)
        token = Token.objects.create(user=user)
        return user


class RatingSerializer(serializers.ModelSerializer):

    class Meta:
        model = Rating
        fields = ('id', 'stars', 'user', 'movie')


class MovieSerializer(serializers.ModelSerializer):

    class Meta:
        model = Movie

        fields = ('id', 'title', 'description', 'no_of_ratings',
'avg_rating')
```

Views: custom update method, authentication

```
from rest_framework import viewsets, status
from rest_framework.response import Response
from rest_framework.decorators import action
from rest_framework.authentication import TokenAuthentication
from rest_framework.permissions import IsAdminUser #, IsAuthenticated #
Already default in settings
from django.contrib.auth.models import User

from .models import Movie, Rating
from .serializers import MovieSerializer, RatingSerializer, UserSerializer


class UserViewSet(viewsets.ModelViewSet):
    queryset = User.objects.all()
    serializer_class = UserSerializer
    authentication_classes = (TokenAuthentication, )
    permission_classes = (IsAdminUser, )


class MovieViewSet(viewsets.ModelViewSet):
    queryset = Movie.objects.all()
    serializer_class = MovieSerializer
    authentication_classes = (TokenAuthentication, )
    # permission_classes = (IsAuthenticated, ) # Already set to default in
settings

    @action(detail=True, methods=['POST'])
    def rate_movie(self, request, pk=None):
        if 'stars' in request.data:

            movie = Movie.objects.get(id=pk)
            stars = request.data['stars']
            user = request.user # Token auth needed

            try:
                rating = Rating.objects.get(user=user.id, movie=movie.id)
                rating.stars = stars
                rating.save()
            except:
                Rating.objects.create(user=user, movie=movie, stars=stars)

            response = {'message': '{} set {}  movie rating to
{}'.format(user, movie.title, stars)}
            return Response(response, status=status.HTTP_200_OK)
        else:
            response = {'data': 'no rating passed in request'}
            return Response(response, status=status.HTTP_400_BAD_REQUEST)


class RatingViewSet(viewsets.ModelViewSet):
    queryset = Rating.objects.all()
    serializer_class = RatingSerializer
    authentication_classes = (TokenAuthentication, )

    def update(self, request, *args, **kwargs):
        response = {'message': 'cannot update like this (use custom
method)'}
        return Response(response, status=status.HTTP_400_BAD_REQUEST)

    def create(self, request, *args, **kwargs):
        response = {'message': 'cannot create like this (use custom
method)'}
        return Response(response, status=status.HTTP_400_BAD_REQUEST)
```

Urls: router

```
from django.urls import path
from rest_framework import routers
from django.conf.urls import include
from .views import MovieViewSet, RatingViewSet, UserViewSet

router = routers.DefaultRouter()
router.register('users', UserViewSet)
router.register('movies', MovieViewSet)
router.register('ratings', RatingViewSet)

urlpatterns = [
    path('', include(router.urls))
]
```