API é um meio de comunicação entre o meu sistema com outros sistemas, ele não faz parte do django e necessita ser instalado à parte

REST é acrônimo de Representational State Transfer, e tem como objetivo primário a definição de características fundamentais para a construção de aplicações Web seguindo boas práticas.

Criar projeto escola
Criar app Curso
NO MODELS

```
from django.db import models

class Base(models.Model):
    criacao = models.DateTimeField(auto_now_add=True)
    atualizacao = models.DateTimeField(auto_now=True)
    ativo = models.BooleanField(default=True)

    class Meta:
        abstract = True


class Curso(Base):
    titulo = models.CharField(max_length=255)
    url = models.URLField(unique=True)

    class Meta:
        verbose_name = 'Curso'
        verbose_name_plural = 'Cursos'

    def __str__(self):
        return self.titulo

class Avaliacao(Base):
    curso = models.ForeignKey(Curso, related_name='avaliacaoes', on_delete=models.CASCAD
E)
    nome = models.CharField(max_length=255)
    email = models.EmailField()
    comentario = models.TextField(blank = True, default = '')
    avaliacao = models.DecimalField(max_digits=2, decimal_places=1)

    class Meta:
        verbose_name = 'Avaliação'
        verbose_name_plural = 'Avaliações'
        # Aquele email não poderá avaliar o curso mais de uma vez
        unique_together = ['email', 'curso']

    def __str__(self):
```

```
    return f'{self.nome} avaliou o curso {self.curso} com a nota {self.avaliacao}'
```

NO SETTINGS

```
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
MEDIA_URL = ' media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'MEDIA')

#DRF
REST_FRAMEWORK = {
  # Autentificação por sessão
  'DEFAULT_AUTHENTICATION_CLASSES': (
    'rest_framework.authentication.SessionAuthentication',
  ),
  # Autorização
  'DEFAULT_PERMISSION_CLASSES': (
    'rest_framework.permissions.IsAuthenticatedOrReadOnly',
  ),
}
```

NO ADMIN

```
from django.contrib import admin
from .models import Curso, Avaliacao

@admin.register(Curso)
class CursoAdmin(admin.ModelAdmin):
  list_display = ('titulo', 'url', 'criacao', 'atualizacao', 'ativo')


@admin.register(Avaliacao)
class AvaliacaoAdmin(admin.ModelAdmin):
  list_display = ('curso', 'nome', 'email', 'avaliacao', 'criacao', 'atualizacao', 'at
ivo')
```

NO TERMINAL

```
pip install djangorestframework markdown django-filter
pip freeze > requirements.txt
```

NA URL DO PROJETO

```
  path('auth/', include('rest_framework.urls'))
```

Criar serializers.py em curso

```
from rest_framework import serializers
```

```python
from .models import Curso, Avaliacao


class AvaliacaoSerializer(serializers.ModelSerializer):

    class Meta:
        # Não mostra o email quando alguém buscar as informações
        extra_kwargs = {
            'email': {'write_only': True}
        }
        # Qual model eu quero usar
        model = Avaliacao
        # Quais campos eu que mostrar
        fields = (
            'id',
            'curso',
            'nome',
            'email',
            'comentario',
            'avaliacao',
            'criacao',
            'ativo',
        )


class CursoSerializer(serializers.ModelSerializer):

    class Meta:
        model = Curso
        fields = (
            'id',
            'titulo',
            'url',
            'criacao',
            'ativo',
        )
```

na views

```python
from rest_framework.views import APIView
from rest_framework.response import Response

from .models import Curso, Avaliacao
from .serializers import CursoSerializer, AvaliacaoSerializer

class CursoAPIView(APIView):
    """
    API de cursos
```

```
    """
    def get(self, request):
        cursos = Curso.objects.all()
        serializer = CursoSerializer(cursos, many=True)
        return Response(serializer.data)


class AvaliacaoAPIView(APIView):
    """
    API de avaliações
    """
    def get(self, request):
        avaliacoes = Avaliacao.objects.all()
        serializer = AvaliacaoSerializer(avaliacoes, many=True)
        return Response(serializer.data)
```

***o que está entre ‘’ Sera o que estará escrito no menu do site
colocar as urls
na escola

```
    path('api/v1/', include('cursos.urls')),
```

no curso

```
    path('cursos/', CursoAPIView.as_view(), name='cursos'),
    path('avaliacoes/', AvaliacaoAPIView.as_view(), name='avaliacoes'),
```

---------- Metodo POST ----------

```
from rest_framework import status

class CursoAPIView(APIView):
. . .
    def post(self, request):
        serializer = CursoSerializer(data=request.data)
        # Verifica se os dados são válido, colocando um excessão
        serializer.is_valid(raise_exception=True)
        serializer.save()
        # Retorna
        return Response(serializer.data, status=status.HTTP_201_CREATED)
```

***Dá pra enviar mensagens assim

```
        return Response({"msg": "Criado com sucesso"}, status=status.HTTP_201_CREATED)
```

*** Ou

```
        return Response({"id": serializer.data['id'], "curso": serializer.data['titulo']},
status=status.HTTP_201_CREATED)
```

Irá ter um formulário onde eu posso enviar arquivo formato JSON

```
 {
      "titulo": "Teste",
      "url": "https://www.youtube.com",
   }
```

---------- Views Genéricas ----------

no views

```python
from rest_framework import generics

from .models import Curso, Avaliacao
from .serializers import CursoSerializer, AvaliacaoSerializer

class CursosAPIView(generics.ListCreateAPIView):
    queryset = Curso.objects.all()
    serializer_class = CursoSerializer

class CursoAPIView(generics.RetrieveUpdateDestroyAPIView):
    queryset = Curso.objects.all()
    serializer_class = CursoSerializer

class AvaliacoesAPIView(generics.ListCreateAPIView):
    queryset = Avaliacao.objects.all()
    serializer_class = AvaliacaoSerializer

class AvaliacaoAPIView(generics.RetrieveUpdateDestroyAPIView):
    queryset = Avaliacao.objects.all()
    serializer_class = AvaliacaoSerializer
```

na url

```python
from django.urls import path

from .views import CursosAPIView, CursoAPIView, AvaliacoesAPIView,  AvaliacaoAPIView

urlpatterns = [
    path('cursos/', CursosAPIView.as_view(), name='cursos'),
    path('cursos/<int:pk>', CursoAPIView.as_view(), name='curso'),
    path('avaliacoes/', AvaliacoesAPIView.as_view(), name='avaliacoes'),
    path('avaliacoes/<int:pk>', AvaliacaoAPIView.as_view(), name='avaliacao'),

]
```

---------- **Sobrescrevendo views genéricas** ----------

na url

```
path('cursos/<int:curso_pk>/avaliacoes/', AvaliacoesAPIView.as_view(), nam
e="curso_avaliacoes"),
  path('cursos/<int:curso_pk>/avaliacoes/<int:avaliacao_pk>', AvaliacaoAPIView
.as_view(), name="curso_avaliacao"),
  path('avaliacoes/', AvaliacoesAPIView.as_view(), name='avaliacoes'),
  path('avaliacoes/<int:avaliacao_pk>', AvaliacaoAPIView.as_view(), name='avaliacao'),
```

*** Ao sobrescrever enviando argumento nomear os argumentos igualmente `<int:avaliacao_pk>`

Na views

```
from rest_framework.generics import get_object_or_404


class AvaliacoesAPIView(generics.ListCreateAPIView):
  queryset = Avaliacao.objects.all()
  serializer_class = AvaliacaoSerializer

  def get_queryset(self):
    # Se tiver enviando o curso_pk eu só devolvo as avaliações dele
    if self.kwargs.get('curso_pk'):
      return self.queryset.filter(curso_id=self.kwargs.get('curso_pk'))
    return self.queryset.all()

class AvaliacaoAPIView(generics.RetrieveUpdateDestroyAPIView):
  queryset = Avaliacao.objects.all()
  serializer_class = AvaliacaoSerializer

  def get_object(self):
    if self.kwargs.get('curso_pk'):
      return get_object_or_404(self.get_queryset(),
        curso_id=self.kwargs.get('curso_id'),
        pk=self.kwargs.get('avaliacao_pk')
        )
    return get_object_or_404(self.get_queryset(), pk=self.kwargs.get('avaliacao_pk'))
```

---------- **Fazendo uma nova versão da Api** ----------

*****Além daquilo que já temos adicionar mais isso, para que assim possamos rodas 2 versões ao mesmo tempo**

Na views

```
from rest_framework import generics, viewsets
from rest_framework.decorators import action
from rest_framework.response import Response

'''
API Versão 2.0
'''

class CursoViewSet(viewsets.ModelViewSet):
```

```
    queryset = Curso.objects.all()
    serializer_class = CursoSerializer

# detail cria a nova rota que será criada, onde somente pode acessar o  method get
    # Agora é possivel acessar http://127.0.0.1:8000/api/v2/cursos/8/avaliacoes
    @action(detail=True, methods=['get'])
    def avaliacoes(self, request, pk=None):
        curso = self.get_object()
        # Pega as avaliacoes(reslated_name) do curso
        serializer = AvaliacaoSerializer(curso.avaliacoes.all(), many=True)
        return Response(serializer.data)


class AvaliacaoViewSet(viewsets.ModelViewSet):
    queryset = Avaliacao.objects.all()
    serializer_class = AvaliacaoSerialize
```

Na url do app

```
from django.urls import path

from rest_framework.routers import SimpleRouter

from .views import (
    CursoViewSet,
    AvaliacaoViewSet
)

router = SimpleRouter()
router.register('cursos', CursoViewSet)
router.register('avaliacoes', AvaliacaoViewSet)
```

Na url do projeto

```
from cursos.urls import router

    path('api/v2/', include(router.urls)),
```

**********Outras dicas

Caso eu não queira libear algumas coisas, assim só excluo o que eu não quiser

```
from rest_framework import mixins

class AvaliacaoViewSet(
    mixins.ListModelMixin,
    mixins.CreateModelMixin,
    mixins.RetrieveModelMixin,
    mixins.UpdateModelMixin,
```

```
    mixins.DestroyModelMixin,
    viewsets.GenericViewSet):
  queryset = Avaliacao.objects.all()
  serializer_class = AvaliacaoSerializer
```

----- Mostrar as avaliacoes na página de cursos
Usar em casos onde tenham poucas avaliacoes
serializers.py

```
class CursoSerializer(serializers.ModelSerializer):
  # Nested Relationship
  avaliacoes =AvaliacaoSerializer(many=True, read_only=True)

  class Meta:
    model = Curso
    fields = (
. . .
      'avaliacoes'
    )
```

Mostrar hyperlink

```
  #Hyperlinked Relaeted Field
avaliacoes = serializers.HyperlinkedRelatedField(
    many=True, read_only=True, view_name="avaliacao-detail")
```

Mostrar apenar Primary Key

```
  # Primary Key Related Field
  avaliacoes = serializers.PrimaryKeyRelatedField(many=True, read_only=True)
```

---------- **Paginação** ----------

No settings

```
REST_FRAMEWORK = {
. . .
  # Paginação
  'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
  'PAGE_SIZE': 2
}
```

*Lembrar de colocar um ordering no models, senão dá warning

Porém, nas views não padrão não recebem essa paginação e é necessário adicionar
no views

```
  @action(detail=True, methods=['get'])
  def avaliacoes(self, request, pk=None):
    self.pagination_class.page_size = 2
    avaliacoes = Avaliacao.objects.finter(curso_id=pk)
    page = self.paginate_queryset(avaliacoes)
```

```
    if page is not None:
        serializer = AvaliacaoSerializer(page, many=True)
        return self.get_paginated_response(serializer.data)

    # Pega as avaliacoes(reslated_name) do curso
    serializer = AvaliacaoSerializer(avaliacoes.all(), many=True)
    return Response(serializer.data)
```

---------- **Autentificação via token** ----------

no settings

```
  'rest_framework.authtoken',

  'DEFAULT_AUTHENTICATION_CLASSES': (
    # 'rest_framework.authentication.SessionAuthentication',
    'rest_framework.authentication.TokenAuthentication',
  ),
```

python manage.py migrate

*O processo abaixo pode ser feito no admin
python manage.py shell
from rest_framework.authtoken.models import Token
from django.contrib.auth.models import User
 user = User.objects.get(id=1)
token = Token.objects.create(user=user)
token.key
copiar token

4257eb9bcba2a1825a25724f4fa938fded691e2d

Caso eu queira enviar pelo insomnia é necessário ir no header e colocar
Authorization Token 4257eb9bcba2a1825a25724f4fa938fded691e2d

PERMISSÕES QUE O USUÁRIO TEM
Criar um usuario com a permissão de adicionar curso
na views

```
from rest_framework import permissions

class CursoViewSet(viewsets.ModelViewSet):
  permission_classes = (permissions.DjangoModelPermissions, )
```

***Neste momento se você tentar acessar a listagem de cursos anonimamente não irá conseguir
Se eu tentar adicionar um curso com o usuario criado dará certo, podem deletar não dará

Dentro de cursos criar permissions.py

```
from rest_framework import permissions

class EhSuperUser(permissions.BasePermission):

    def has_permission(self, request, view):
        if request.method == 'DELETE':
            if request.user.is_superuser:
                return True
            else:
                return
        return True
```

na views.py

```
from .permissions import EhSuperUser

class CursoViewSet(viewsets.ModelViewSet):
    permission_classes = (
        EhSuperUser,
        permissions.DjangoModelPermissions,
    )
```

***Colocar em ordem pois caso a primeira atenda a requisição não é necessário ir adiante
Agora se eu tentar deletar um curso sem ser super usuário eu sou impedido

---------- Delimitar quantidade de requisições ----------
no settings

```
REST_FRAMEWORK = {
. . .
    #  Delimitar requisições
    'DEFAULT_THROTTLE_CLASSES': [
        'rest_framework.throttling.AnonRateThrottle',
        'rest_framework.throttling.UserRateThrottle'
    ],
    #  Definindo quantidade de requisições
    'DEFAULT_THROTTLE_RATES': {
        'anon': '5/minute',
        'user': '10/minute'
    }
}
```

*Dá pra usar second, minute, day, year

---------- Validação dos campos ----------
no serializers.py
em `AvaliacaoSerializer`

```
  def validate_avaliacao(self, valor):
      if valor in range(1, 6):
          return valor
      raise serializers.ValidationError('A avaliação precisa ser um número entre 1 e 5')
```

**É necessário que o nome seja `validate_`

```
from django.db.models import Avg
```

em `CursoSerializer`

```
  media_avaliacoes = serializers.SerializerMethodField()

fields = (
        . . .
        'media_avaliacoes'
    )

  def get_media_avaliacoes(self, obj):
      # pega as avaliacoes faz a media com cada uma das avaliaco e pega eles pelo av
              aliacao__avg
      media = obj.avaliacoes.aggregate(Avg('avaliacao')).get('avaliacao__avg')

      if media is None:
          return 0
      # Apenas arredonda a media
      return round(media * 2 ) / 2
```

---------- Testando API ----------
* Os testes serão feitos no pycharm pois são mais fáceis de serem executados
pip install requests
*importante: estar com servidor rodando
no arquivo raiz criar teste_requests.py e nele

```
import requests

# GET AVALIAÇÕES

avaliacoes = requests.get('http://localhost:8000/api/v2/avaliacoes/')
# ACESSANDO O CÓDIGO DE STATUS HTTP
print(avaliacoes.status_code)
```

botão direito no arquivo run teste_requests

→ Mais alguns testes

```
import requests
```

```python
# GET AVALIAÇÕES

# avaliacoes = requests.get('http://localhost:8000/api/v2/avaliacoes/')
# ACESSANDO O CÓDIGO DE STATUS HTTP
# print(avaliacoes.status_code)

# Acessando os dados da resposta
# Retorna os dados em um dicionário
# print(avaliacoes.json())

# Acessando a quantidade de registro
# print(avaliacoes.json()['count'])

# Acessando a próxima página de resultados
# print(avaliacoes.json()['next'])

# Acessoando os resultados dessa página
# Retorna os dados em uma lista
# print(avaliacoes.json()['results'])

# Acessando o primeiro elemento da lista de resultados
# print(avaliacoes.json()['results'][0])

# Acessando o ultimo elemento da lista de resultados
# print(avaliacoes.json()['results'][-1])

# Acessando o nome da pessoa que fez a ultima avaliação
# print(avaliacoes.json()['results'][-1]['nome'])

# GET AVALIAÇÃO
# avaliacao = requests.get('http://localhost:8000/api/v2/avaliacoes/5/')
# print(avaliacao.json())

#GET CURSOS
# Para acessar os cursos é necessário ser um usuário autenticado
headers = {'Authorization': 'Token 7db06996f551b69f08dad5b1942f63bb93ddf11f'}

cursos = requests.get(url='http://localhost:8000/api/v2/cursos/', headers=headers)
print(cursos.status_code)
print(cursos.json())
```

---------- Testes com jsonpath ----------
pip install jsonpath
*importante: estar com servidor rodando
na base do projeto criar teste_jsonpath.py

```
import requests
import jsonpath

avaliacoes = requests.get('http://localhost:8000/api/v2/avaliacoes/')

# resultados = jsonpath.jsonpath(avaliacoes.json(), 'results')

# print(resultados)

# primeira = jsonpath.jsonpath(avaliacoes.json(), 'results[0]')
# print(primeira)

# nome = jsonpath.jsonpath(avaliacoes.json(), 'results[0].nome')
# print(nome)

# nota_dada = jsonpath.jsonpath(avaliacoes.json(), 'results[0].avaliacao')
# print(nota_dada)

# Todos os nomes das pessoas que avaliaram o curso
# nomes = jsonpath.jsonpath(avaliacoes.json(), 'results[*].nome')
# print(nomes)

# Todos as avaliações do curso
notas = jsonpath.jsonpath(avaliacoes.json(), 'results[*].avaliacao')
print(notas)
```

---------- Testes GET ----------
criar na base do projeto teste_get.py

```
import requests

headers = {'Authorization': 'Token 7db06996f551b69f08dad5b1942f63bb93ddf11f'}

url_base_cursos = 'http://localhost:8000/api/v2/cursos/'
url_base_avaliacoes = 'http://localhost:8000/api/v2/avaliacoes/'

resultado = requests.get(url=url_base_cursos, headers=headers)

# print(resultado.json())

# Testando se o endpoint está correto
assert resultado.status_code == 200

# Testando a quantidade de registros
# Útil quando espero resultados fixos
```

```
# assert resultado.json()['count'] == 3

# Testando se o titulo do primeiro curso está correto
# Teste de valores iguais
assert resultado.json()['results'][0]['titulo'] == 'Teste v2'
```

---------- Testes POST ----------

```
import requests

headers = {'Authorization': 'Token 7db06996f551b69f08dad5b1942f63bb93ddf11f'}

url_base_cursos = 'http://localhost:8000/api/v2/cursos/'
url_base_avaliacoes = 'http://localhost:8000/api/v2/avaliacoes/'

novo_curso = {
  "titulo": "Curso criado através de teste5",
  "url": "http://www.cursoteste5.com"
}

resultado = requests.post(url=url_base_cursos, headers=headers, data=novo_curso)

# Testando o código de estatus HTTP 201
assert resultado.status_code == 201

# Testando se o título retornado é o mesmo que o informado
assert resultado.json()['titulo'] == novo_curso['titulo']

# Lembrando que é necessário mudar o titulo e url em cada envio
```

---------- Testes PUT ----------

```
import requests

headers = {'Authorization': 'Token 7db06996f551b69f08dad5b1942f63bb93ddf11f'}

url_base_cursos = 'http://localhost:8000/api/v2/cursos/'
url_base_avaliacoes = 'http://localhost:8000/api/v2/avaliacoes/'

curso_atualizado = {
  "titulo": "Curso atualizado",
  "url": "http://cursoatualizado.com"
}

# Buscanco curso com id 9
# curso = requests.get(url=f'{url_base_cursos}9/', headers=headers)
```

```
# print(curso.json())

resultado = requests.put(url=f'{url_base_cursos}9/', headers=headers, data=curso_atualizado)

# Testando código de status HTTP
assert resultado.status_code == 200

# Testando se título é igual
assert resultado.json()['titulo'] == curso_atualizado['titulo']
```

---------- Testes DELETE ----------

```
import requests


headers = {'Authorization': 'Token 7db06996f551b69f08dad5b1942f63bb93ddf11f'}

url_base_cursos = 'http://localhost:8000/api/v2/cursos/'
url_base_avaliacoes = 'http://localhost:8000/api/v2/avaliacoes/'

resultado = requests.delete(url=f'{url_base_cursos}9/', headers=headers)

# Testando o código HTTP
assert resultado.status_code == 204

# Testando se o tamanho do conteúdo retornado é 0
assert len(resultado.text) == 0

# Lembrando que só pode ser executado uma vez, depois mudar id
```

---------- Testando com Pytest ----------

pip install pytest
na base testes_pytest.py

```
import requests


class TestCursos:
    headers = {'Authorization': 'Token 7db06996f551b69f08dad5b1942f63bb93ddf11f'}

    url_base_cursos = 'http://localhost:8000/api/v2/cursos/'
    url_base_avaliacoes = 'http://localhost:8000/api/v2/avaliacoes/'
```

```python
  # É necessário ser test_
  def test_get_cursos(self):
    resposta = requests.get(url=self.url_base_cursos, headers=self.headers)

    assert resposta.status_code == 200

  def test_get_curso(self):
    resposta = requests.get(url=f'{self.url_base_cursos}21/', headers=self.headers)

    assert resposta.status_code == 200

  def test_post_curso(self):
    novo_curso = {
      "titulo": "Curso criado através de Pytest",
      "url": "http://www.cursopytest.com"
    }
    resposta = requests.post(url=self.url_base_cursos, headers=self.headers, data=novo
_curso)

    assert resposta.status_code == 201
    assert resposta.json()['titulo'] == novo_curso['titulo']

  def test_put_curso(self):
    curso_atualizado = {
      "titulo": "Curso atualizado Pytest",
      "url": "http://cursoatualizado.com"
    }
    resposta = requests.put(url=f'{self.url_base_cursos}24/', headers=self.headers, d
ata=curso_atualizado)

    assert resposta.status_code == 200

  def test_put_titulo_curso(self):
    curso_atualizado = {
      "titulo": "Curso atualizado Pytest",
      "url": "http://cursoatualizado.com"
    }
    resposta = requests.put(url=f'{self.url_base_cursos}24/', headers=self.headers, d
ata=curso_atualizado)

    assert resposta.json()['titulo'] == curso_atualizado['titulo']

  def test_delete_curso(self):
    resposta = requests.delete(url=f'{self.url_base_cursos}24/', headers=self.headers)

    assert resposta.status_code == 204 and len(resposta.text) == 0
```

```
# LEMBRAR DE MUDAR ID's E TOKEN DE USUÁRIO
```

pytest teste_pytest.py