REST API

Curso completo de Python com certificado do MEC e reuniões com professores.

Conheça nosso curso completo de Python e Django que te da acesso à:

- Mais de 630 aulas
- Agendamento de reuniões com professores
- Reconhecido pelo MEC
- Análises de códigos
- Eventos entre alunos
- Exercícios automáticos
- E muito mais

Para quem participou da 4D4P terá um desconto especial, confira no link abaixo:

https://youtu.be/yG57JRGRXTc

▼ Passos iniciais

Primeiro devemos criar o ambiente virtual:

```
# Criar
    # Linux
        python3 -m venv venv
    # Windows
        python -m venv venv
```

Após a criação do venv vamos ativa-lo:

```
#Ativar
    # Linux
        source venv/bin/activate
    # Windows
        venv\Scripts\Activate

# Caso algum comando retorne um erro de permissão execute o código e tente novamente:

Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

Agora vamos fazer a instalação do Django e as demais bibliotecas:

```
pip install django
pip install pillow
pip install django-ninja
```

Vamos criar o nosso projeto Django:

```
django-admin startproject core .
```

Rode o servidor para testar:

```
python manage.py runserver
```

Crie o app usuario:

```
python manage.py startapp livros
```

Ative o auto-save

INSTALE O APP!

Crie uma URL para API:

```
from .api import api

path('api/', api.urls)
```

Crie um router para api em core/api.py:

```
from ninja import NinjaAPI
from livros.api import livros_router

api = NinjaAPI()
api.add_router('livros/', livros_router)
```

▼ Livros

Crie as tabelas:

```
class Categorias(models.Model):
    nome = models.CharField(max_length=50)

    def __str__(self):
        return self.nome

class Livros(models.Model):
    streaming_choices = (('N', 'Netflix'), ('A', 'Amazon Prime'))
    nome = models.CharField(max_length=50)
    streaming = models.CharField(max_length=2, choices=streaming_choices)
    nota = models.IntegerField(null=True, blank=True)
    comentarios = models.TextField(null=True, blank=True)
    categorias = models.ManyToManyField(Categorias)

    def __str__(self):
        return self.nome
```

Execute as migrações!!

Crie um Schema para criação dos livros:

```
from ninja import ModelSchema, Schema, Schema
from .models import Livros

class LivroSchema(ModelSchema):
    class Meta:
        model = Livros
        fields = ['nome', 'streaming', 'categorias']
```

Crie um endpoint para criação do livro:

```
from ninja import Router
from .models import Livros, Categorias
from .schemas import LivroSchema

livros_router = Router()

@livros_router.post('/', response={200: LivroSchema})
def create_livro(request, livro_schema: LivroSchema):
    nome = livro_schema.dict()['nome']
    streaming = livro_schema.dict()['streaming']
    categorias = livro_schema.dict()['categorias']
    livro = Livros(nome=nome, streaming=streaming)
    livro.save()

    for categoria in categorias:
        livro.categorias.add(Categorias.objects.get(id=categoria))

    livro.save()
    return livro
```

Desenvolva o endpoint para avaliar o livro:

```
@livros_router.put('/{livro_id}')
def avaliar_livro(request, livro_id: int, avaliacao_schema: AvaliacaoSchema):
    comentarios = avaliacao_schema.dict()['comentarios']
    nota = avaliacao_schema.dict()['nota']

    livro = Livros.objects.get(id=livro_id)
    livro.comentarios = comentarios
    livro.nota = nota
    livro.save()
    return {'status': 'Avaliação realizada com sucesso'}
```

Construa o Schema para avaliação:

```
class AvaliacaoSchema(ModelSchema):
    class Meta:
        model = Livros
        fields = ['nota', 'comentarios']
```

Permita a exclusão de livros:

```
@livros_router.delete('/{livro_id}')
def deletar_livro(request, livro_id: int):
    livro = Livros.objects.get(id=livro_id)
    livro.delete()

    return livro_id
```

Sorteie um livro aleatoriamente:

```
@livros_router.get('/sortear/', response={200: LivroSchema, 404: dict})
def sortear_livro(request, filtros: Query[FiltrosSortear]):
    nota_minima = filtros.dict()['nota_minima']
```

```
    categoria = filtros.dict()['categorias']
    reassistir = filtros.dict()['reassistir']

    livros = Livros.objects.all()

    if not reassistir:
        livros = livros.filter(nota=None)

    if nota_minima:
        livros = livros.filter(nota__gte=nota_minima)
    if categoria:
        livros = livros.filter(categorias__id=categoria)

    livro = livros.order_by('?').first()

    if livros.count() > 0:
        return 200, livro
    else:
        return 404, {'status': 'Livro não encontrado'}
```

Crie o Schema dos filtros

```
class FiltrosSortear(Schema):
    nota_minima: int = None
    categorias: int = None
    reassistir: bool = False
```