

TRABALHANDO COM DJANGO



TRABALHANDO COM DJANGO

Vamos dar sequência em implementação de Aplicativos em DJANGO.
Nesse exemplo vamos implementar um aplicativo que efetua cadastro de Funcionário gerando relatório em PDF e gera gráfico dos funcionários.

Vamos criar o novo aplicativo.

django-admin startapp funcionarios

Vamos copiar os arquivos disponibilizados na aula para dentro da estrutura do projeto

Vamos agora adicionar o nome do aplicativo ao **settings.py** do nosso projeto em **installed_apps**

```
'funcionarios',
```

TRABALHANDO COM DJANGO

Criando a estrutura de tabelas do banco de dados do novo aplicativo.

```
python manage.py makemigrations funcionarios
```

Validando a estrutura de tabelas que será criada no banco de dados

```
python manage.py sqlmigrate funcionarios 0001
```

Compilar as alterações com o comando migrate

```
python manage.py migrate
```

TRABALHANDO COM DJANGO

Adicionar o acesso das rotas do aplicativo ao projeto existente no arquivo urls.py.

```
path(' ', include('funcionarios.urls')),  
path('funcionarios/', include('funcionario.urls')),
```

TRABALHANDO COM DJANGO

Acessando o sistema de Cadastro de Funcionários e acessando suas funcionalidades.

[LISTAGEM](#) [NOVO](#) [GRAFICOS](#)



ID	Nome	Idade	Sexo	Salário

[Imprimir Relatório](#)

TRABALHANDO COM DJANGO

Acessando Administração da Aplicação.

Selecione pessoa para modificar

ADICIONAR PESSOA +



Pesquisar

Ação:



Ir

0 de 1 selecionados



NOME

IDADE

SEXO

SALARIO



joao

36

M

444,00

1 pessoa

TRABALHANDO COM DJANGO

Utilizando DJANGO como API.

Podemos utilizar o DJANGO também como uma API REST. Como já visto com o FLASK.

O que é API REST.

API REST, também chamada de API RESTful, é uma interface de programação de aplicações (API ou API web) que está em conformidade com as restrições do estilo de arquitetura REST, permitindo a interação com serviços web RESTful.

REST é a sigla em inglês para transferência representacional de estado.

Quando um cliente faz uma solicitação usando uma API RESTful, essa API transfere uma representação do estado do recurso ao solicitante ou endpoint. Essa informação (ou representação) é entregue via HTTP utilizando um dos vários formatos possíveis: Javascript Object Notation (JSON), HTML, XLT, Python, PHP ou texto sem formatação. O formato JSON é a linguagem de programação mais usada porque, apesar de seu nome, é independente de qualquer outra linguagem e pode ser lido por máquinas e humanos.

TRABALHANDO COM DJANGO

Para isso vamos instalar no nosso ambiente virtual a biblioteca responsável por criação de APIS com o DJANGO a DJANGORESTFRAMEWORK.

```
pip install djangorestframework
```

Vamos criar nosso aplicativo no projeto do Django.

```
django-admin startapp djangoapi
```

Vamos agora adicionar o nome do aplicativo ao `settings.py` do nosso projeto em `installed_apps`

```
'rest_framework',  
'djangoapi',
```


TRABALHANDO COM DJANGO

Vamos criar os models do aplicativo da api dentro do models.py onde iremos criar os campos.

```
class Livros(models.Model):
```

```
    class Meta:
```

```
        db_table = 'livros'
```

```
    titulo = models.CharField(max_length=200)
```

```
    nome = models.CharField(max_length=200)
```

```
    editora = models.CharField(max_length=200)
```

```
    def __str__(self):
```

```
        return self.titulo
```

TRABALHANDO COM DJANGO

Criando a estrutura de tabelas do banco de dados do novo aplicativo.

```
python manage.py makemigrations djangoapi
```

Validando a estrutura de tabelas que será criada no banco de dados

```
python manage.py sqlmigrate djangoapi 0001
```

Compilar as alterações com o comando migrate

```
python manage.py migrate
```

TRABALHANDO COM DJANGO

Criando o arquivo **serializers.py**. O módulo serializers irá transformar os campos dos modelos em formato JSON. Padrão de uso de uma API.

```
from rest_framework import serializers  
from .models import Livros  
class LivrosSerializer(serializers.ModelSerializer):
```

```
    class Meta:
```

```
        model = Livros  
        fields = '__all__'
```

TRABALHANDO COM DJANGO

Criando as views utilizando as classes em formato da classe da biblioteca `django rest framework`. Listando todos os objetos da classe do modelo.

```
from rest_framework import generics
from .models import Livros
from .serializers import LivrosSerializer

class LivrosLista(generics.ListCreateAPIView):

    queryset = Livros.objects.all()
    serializer_class = LivrosSerializer
```

TRABALHANDO COM DJANGO

Criando as rotas de urls no arquivo urls.py. Utilizando a função urls do Django.

```
from django.urls import re_path as url  
from . import views
```

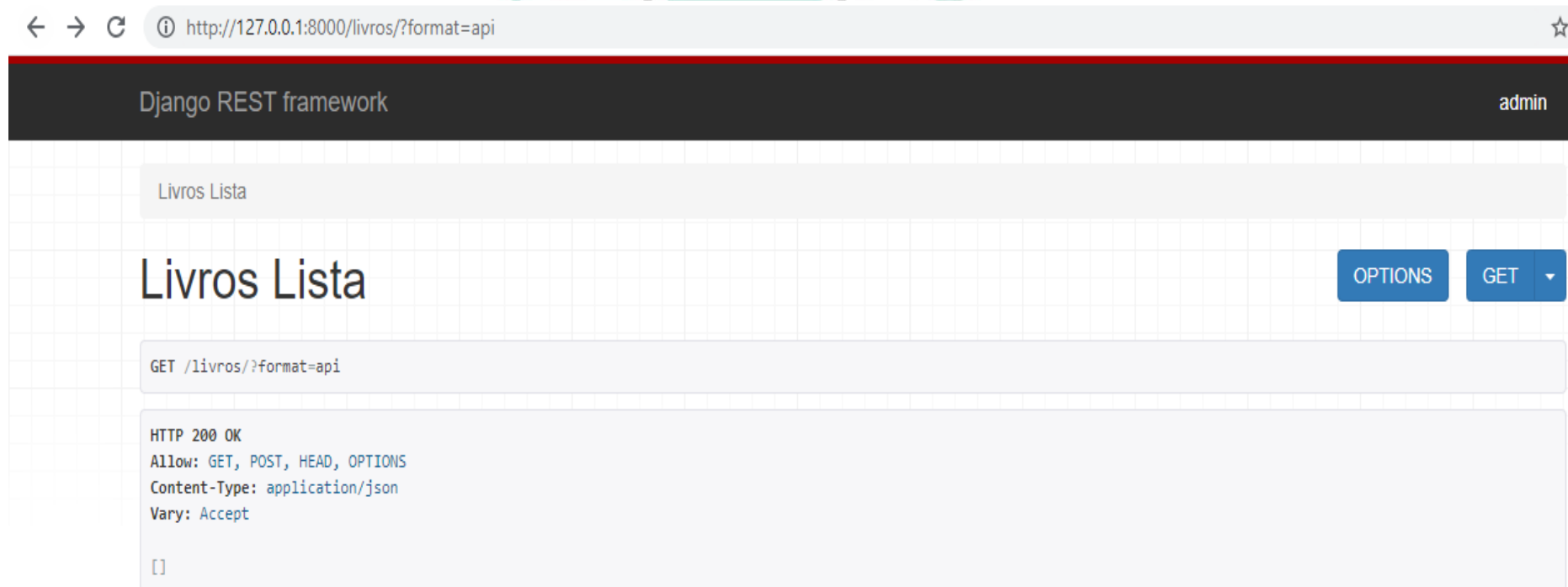
```
urlpatterns = [  
    url(r'^livros/$', views.LivrosLista.as_view(), name='lista-livros'),  
]
```

Adicionando a rota no arquivo urls.py do projeto.

```
url(r'^$', include('djangoapi.urls')),
```

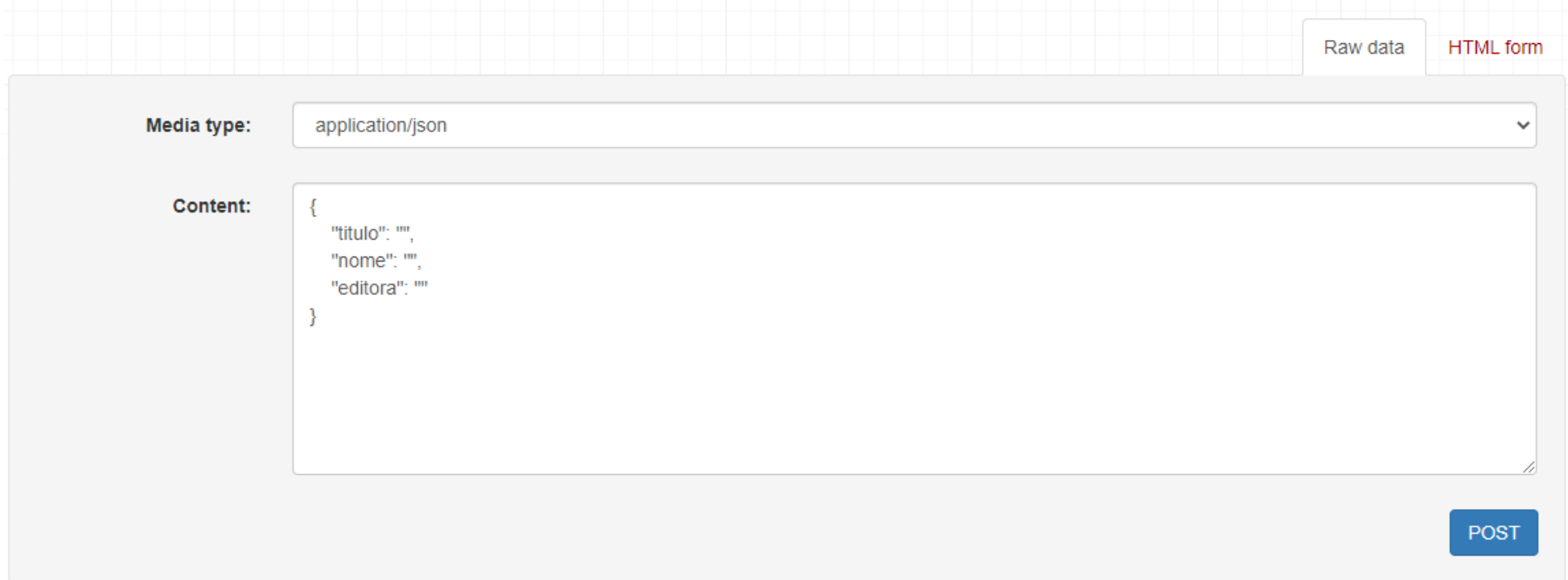
TRABALHANDO COM DJANGO

Testando a api usando métodos GET e POST.



TRABALHANDO COM DJANGO

Testando a api usando métodos GET e POST.



The image shows a REST client interface with a grid background. At the top right, there are two tabs: "Raw data" and "HTML form". The "Raw data" tab is active. Below the tabs, there is a "Media type:" dropdown menu set to "application/json". Below that, there is a "Content:" text area containing a JSON object:

```
{  "titulo": "",  "nome": "",  "editora": ""}
```

. At the bottom right, there is a blue button labeled "POST".