

Back-End Frameworks

Aula 8 - 12/05/2023



https://bit.ly/23_1_backend_n



O que temos para **hoje**?

- Situação problema;
- API;
- Criação de API;



Situação problema



Situação problema

- Você desenvolveu um sistema de e-commerce;
- Você deve comunicar o seu sistema com o sistema dos correios para calcular o valor do frete e o prazo de entrega;
- Você deve comunicar o seu sistema com o sistema de um gateway de pagamento para processar as compras;



Situação problema

● Como fazer?



API

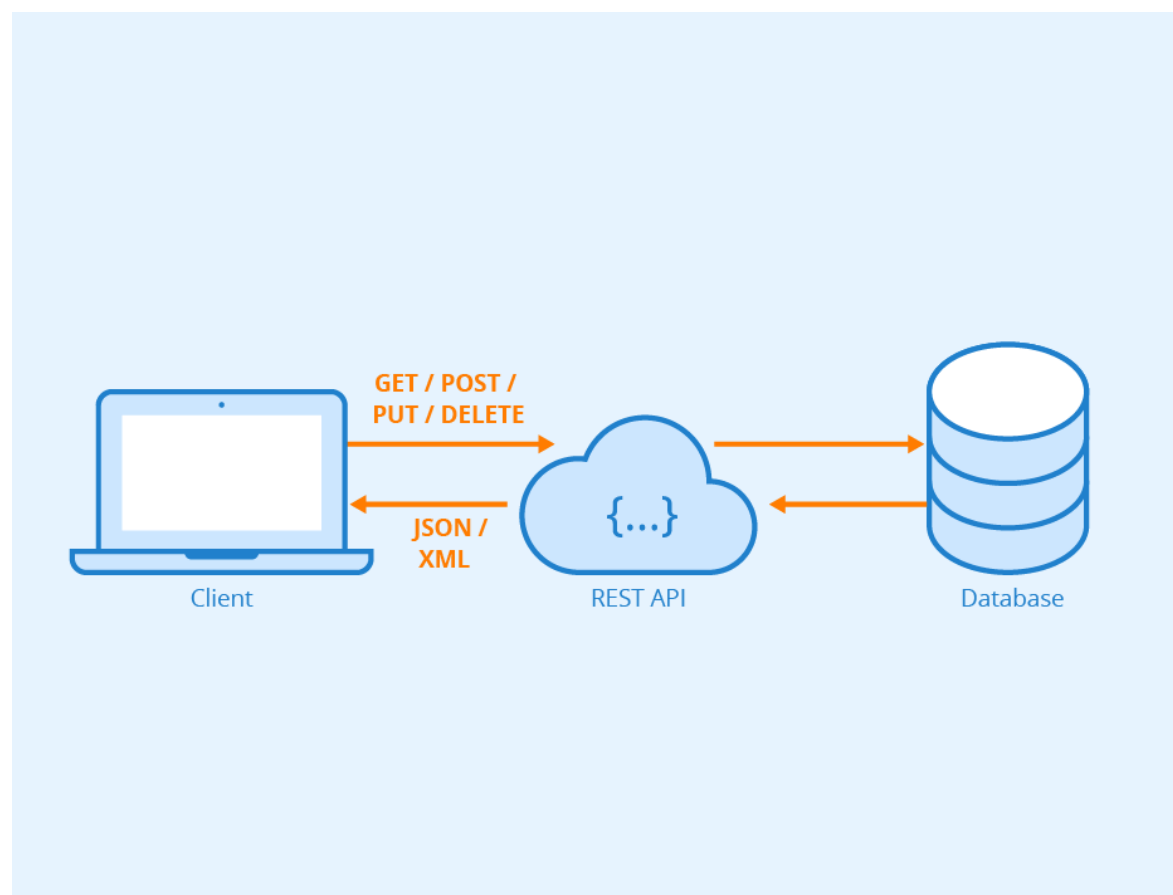


API

- API: Application Programming Interface (Interface de Programação de Aplicação);
- Uma forma de comunicar dois ou mais sistemas por meio de uma linguagem comum;



API





API

- ◎ REST-API: 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



API

- ◎ REST: Representational State Transfer (Transferência de Estado Representacional)
- ◎ Não é um protocolo ou padrão, mas sim um conjunto de restrições de arquitetura;



API

- Quando um cliente faz uma solicitação usando uma API RESTful, essa API transfere uma representação do estado do recurso ao solicitante ou endpoint;

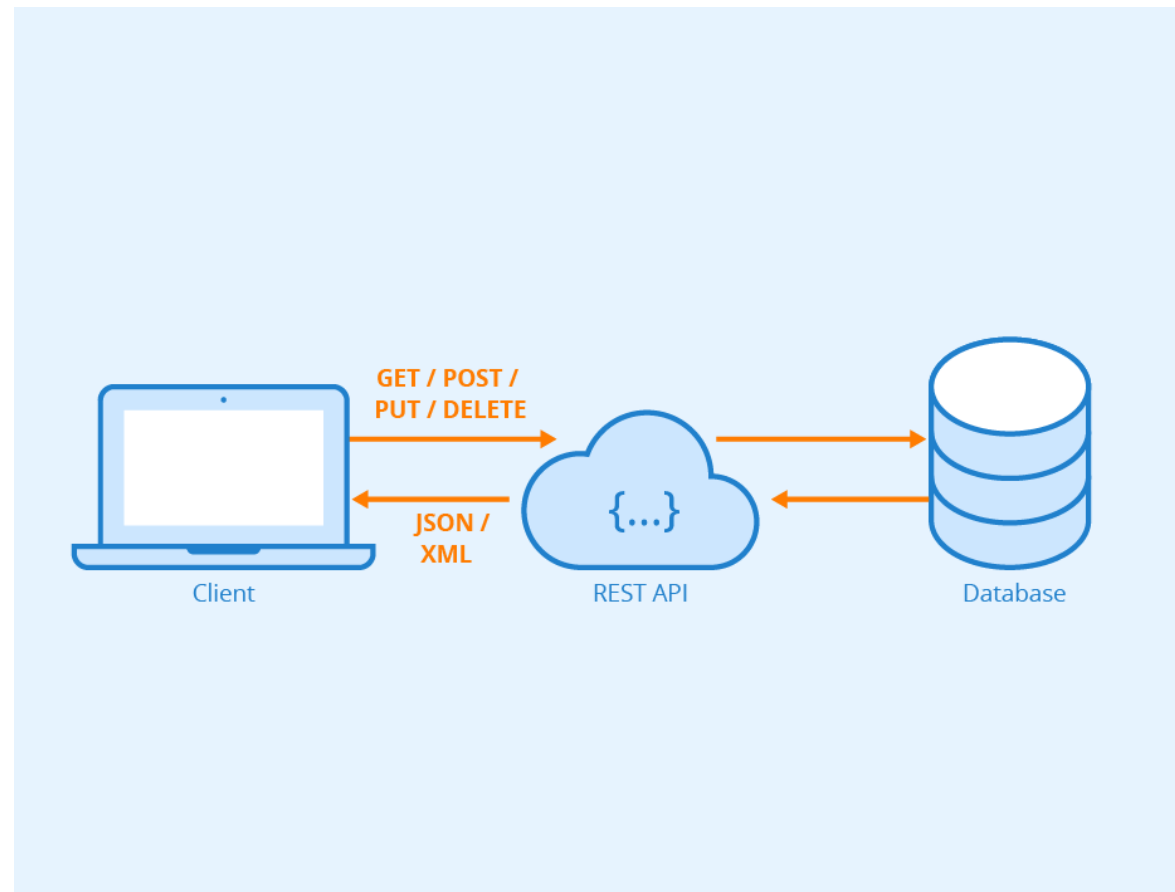


API

- 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.



API



**API**

- Acessar:
<https://randomuser.me/api/>



API

○ Resultado?



API

- Acessar:
<https://jsonformatter.curiousconcept.com/>
- Colar o valor do site anterior nesse site;
- Clicar em “Process”;



API

```
{
  "results": [
    {
      "gender": "male",
      "name": {
        "title": "Mr",
        "first": "Johan",
        "last": "Henry"
      },
      "location": {
        "street": {
          "number": 2646,
          "name": "Place de L'Abbé-Jean-Lebeuf"
        },
        "city": "Nanterre",
        "state": "Haute-Savoie",
        "country": "France",
        "postcode": 18596,
        "coordinates": {
          "latitude": "-18.3974",
          "longitude": "-71.9367"
        },
        "timezone": {
          "offset": "+3:00",
          "description": "Baghdad, Riyadh, Moscow, St. Petersburg"
        }
      }
    }
  ]
}
```



API

- Formato mais usado em APIs atualmente: JSON;
- JSON: JavaScript Object Notation – Notação de Objetos JavaScript;
- É uma formatação leve de troca de dados



API

```
1  {
2    "string": "Hi",
3    "number": 2.5,
4    "boolean": true,
5    "null": null,
6    "object": { "name": "Kyle", "age": 24 },
7    "array": ["Hello", 5, false, null, { "key": "value", "number": 6 }],
8    "arrayOfObjects": [
9      { "name": "Jerry", "age": 28 },
10     { "name": "Sally", "age": 26 }
11   ]
12 }
13
```



Criação de API



Criação de API

- A criação de uma API depende da programação backend;
- Várias linguagens podem ser usadas;
- Vamos usar Python com Django!

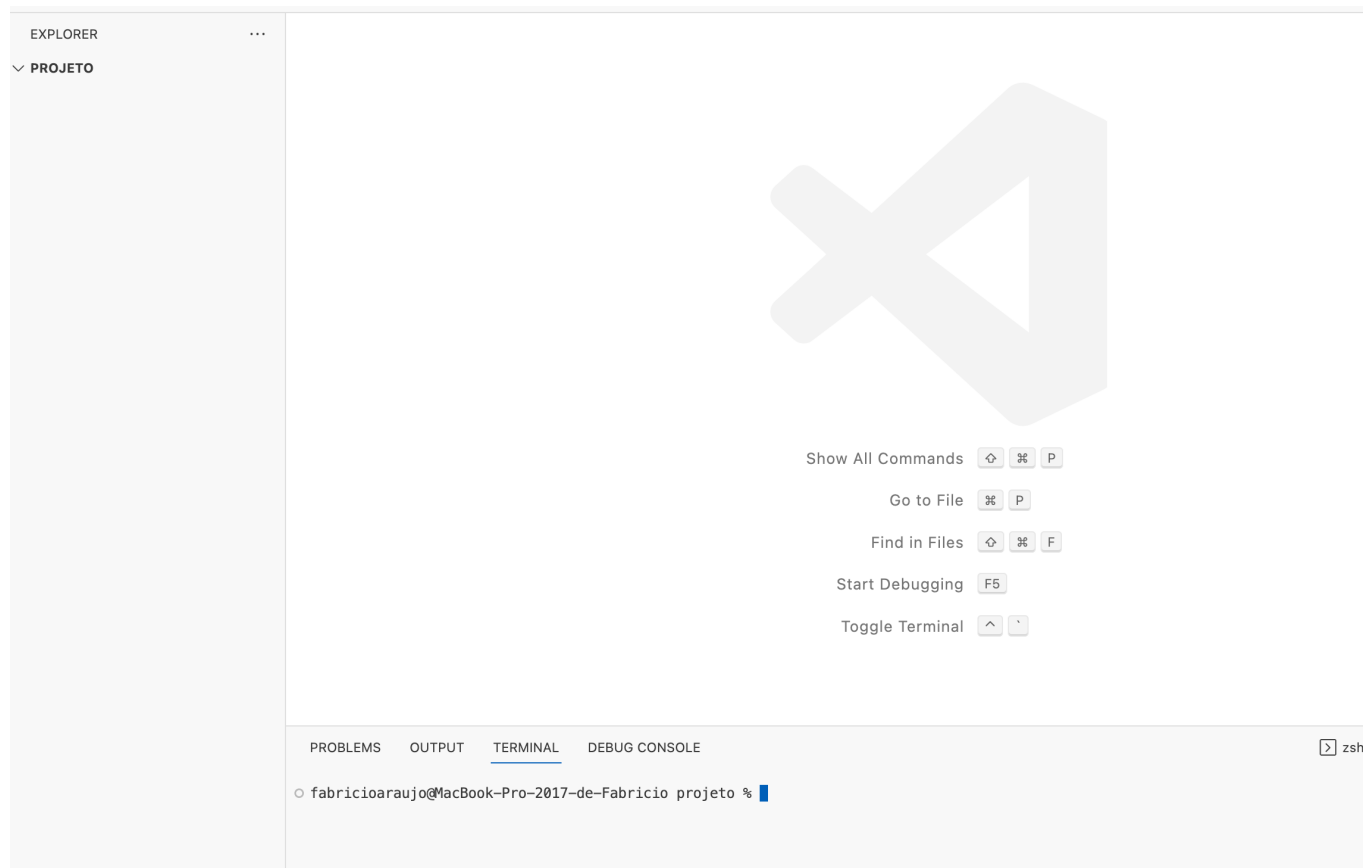


Criação de API

- Criar uma pasta “projeto” e abrir esta pasta no Visual Studio;



Criação de API





Criação de API

- Digitar no terminal:
`py -m pip install django`
- Depois:
`py -m django --version`



Criação de API

```
● fabricioaraujo@MacBook-Pro-2017-de-Fabricio projeto % python3 -m pip install django
Requirement already satisfied: django in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (4.1.3)
Requirement already satisfied: asgiref<4,>=3.5.2 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from django) (3.5.2)
Requirement already satisfied: sqlparse>=0.2.2 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from django) (0.4.2)
WARNING: You are using pip version 21.2.4; however, version 23.1.2 is available.
You should consider upgrading via the '/Library/Frameworks/Python.framework/Versions/3.10/bin/python3 -m pip install --upgrade pip' command.
● fabricioaraujo@MacBook-Pro-2017-de-Fabricio projeto % python3 -m django version
4.1.3
○ fabricioaraujo@MacBook-Pro-2017-de-Fabricio projeto %
```

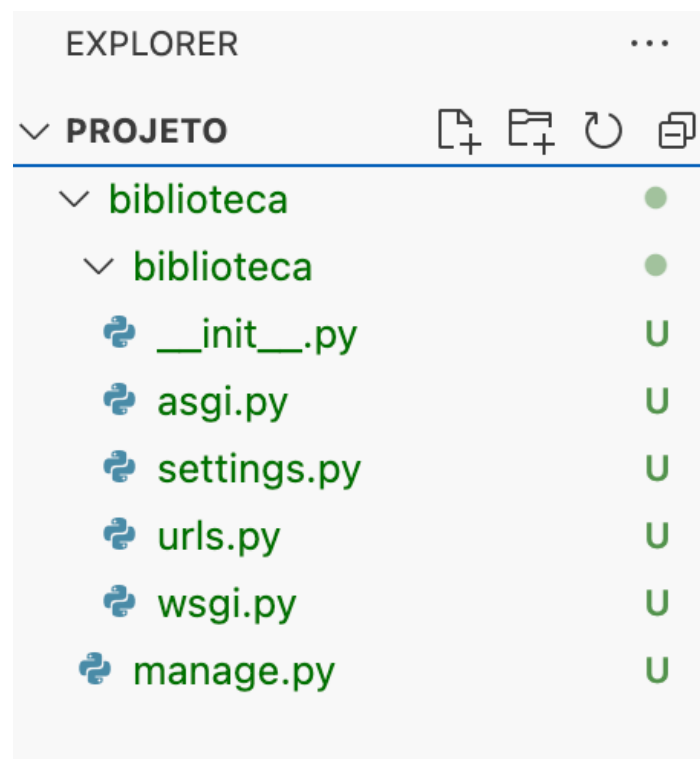


Criação de API

- Digitar no terminal:
`py -m django startproject biblioteca`



Criação de API





Criação de API

- Digitar no terminal:
cd biblioteca



Criação de API

- Digitar no terminal:
`py -m django startapp livros`



Criação de API

▼ biblioteca	●
▼ biblioteca	●
> __pycache__	●
+ __init__.py	U
+ asgi.py	U
+ settings.py	U
+ urls.py	U
+ wsgi.py	U
▼ livros	●
> __pycache__	●
> migrations	●
+ __init__.py	U
+ admin.py	U
+ apps.py	U
+ models.py	U
+ tests.py	U
+ views.py	U
≡ db.sqlite3	U
+ manage.py	U



Criação de API

- Digitar no terminal:
`py manage.py runserver`



Criação de API

```
Watching for file changes with StatReloader
Performing system checks...
```

```
System check identified no issues (0 silenced).
```

```
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
```

```
Run 'python manage.py migrate' to apply them.
```

```
May 10, 2023 - 22:55:29
```

```
Django version 4.1.3, using settings 'biblioteca.settings'
```

```
Starting development server at http://127.0.0.1:8000/
```




Criação de API

django

View [release notes](#) for Django 4.1



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.



Criação de API

- Digitar no terminal:
`py manage.py migrate`



Criação de API

Operations to perform:

Apply all migrations: admin, auth, contenttypes, sessions

Running migrations:

```
Applying contenttypes.0001_initial... OK
Applying auth.0001_initial... OK
Applying admin.0001_initial... OK
Applying admin.0002_logentry_remove_auto_add... OK
Applying admin.0003_logentry_add_action_flag_choices... OK
Applying contenttypes.0002_remove_content_type_name... OK
Applying auth.0002_alter_permission_name_max_length... OK
Applying auth.0003_alter_user_email_max_length... OK
Applying auth.0004_alter_user_username_opts... OK
Applying auth.0005_alter_user_last_login_null... OK
Applying auth.0006_require_contenttypes_0002... OK
Applying auth.0007_alter_validators_add_error_messages... OK
Applying auth.0008_alter_user_username_max_length... OK
Applying auth.0009_alter_user_last_name_max_length... OK
Applying auth.0010_alter_group_name_max_length... OK
Applying auth.0011_update_proxy_permissions... OK
Applying auth.0012_alter_user_first_name_max_length... OK
Applying sessions.0001_initial... OK
```



Criação de API

- Digitar no terminal:
`py -m pip install djangorestframework`



Criação de API

```
Collecting djangoestframework
  Downloading djangoestframework-3.14.0-py3-none-any.whl (1.1 MB)
    |████████████████████| 1.1 MB 959 kB/s
Collecting pytz
  Downloading pytz-2023.3-py2.py3-none-any.whl (502 kB)
    |████████████████████| 502 kB 4.3 MB/s
Requirement already satisfied: django>=3.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from djangoestframework) (4.1.3)
Requirement already satisfied: sqlparse>=0.2.2 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from django>=3.0->djangoestframework) (0.4.2)
Requirement already satisfied: asgiref<4,>=3.5.2 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (from django>=3.0->djangoestframework) (3.5.2)
Installing collected packages: pytz, djangoestframework
Successfully installed djangoestframework-3.14.0 pytz-2023.3
WARNING: You are using pip version 21.2.4; however, version 23.1.2 is available.
You should consider upgrading via the '/Library/Frameworks/Python.framework/Versions/3.10/bin/python3 -m pip install --upgrade pip' command.
```



Criação de API

- Na pasta biblioteca, abrir o arquivo “settings.py”;
- Buscar a linha 39 e deixar assim:



Criação de API

```
33  ✓ INSTALLED_APPS = [  
34      'django.contrib.admin',  
35      'django.contrib.auth',  
36      'django.contrib.contenttypes',  
37      'django.contrib.sessions',  
38      'django.contrib.messages',  
39      'django.contrib.staticfiles',  
40      'rest_framework',  
41      'livros'  
42  ]
```



Criação de API

- Na pasta livros, abrir o arquivo “models.py”;
- Editar o arquivo para que ele fique assim:



Criação de API

biblioteca > livros >  models.py >  Livros

```
1  from django.db import models
2  from uuid import uuid4
3
4  # Create your models here.
5  class Livros(models.Model):
6      id_livro = models.UUIDField(primary_key=True, default=uuid4, editable=False)
7      titulo = models.CharField(max_length=255)
8      autor = models.CharField(max_length=255)
9      ano_lancamento = models.IntegerField()
10     estado = models.CharField(max_length=50)
11     paginas = models.IntegerField()
12     editora = models.CharField(max_length=255)
13     dt_criacao = models.DateTimeField(auto_now_add=True)
```



Criação de API

- Digitar no terminal:
`py manage.py makemigrations`
- Depois:
`py manage.py migrate`



Criação de API

- fabricioaraujo@MacBook-Pro-2017-de-Fabricio biblioteca % python3 manage.py makemigrations
Migrations for 'livros':
 livros/migrations/0001_initial.py
 – Create model Livros
- fabricioaraujo@MacBook-Pro-2017-de-Fabricio biblioteca % python3 manage.py migrate
Operations to perform:
 Apply all migrations: admin, auth, contenttypes, livros, sessions
Running migrations:
 Applying livros.0001_initial... OK
- fabricioaraujo@MacBook-Pro-2017-de-Fabricio biblioteca %





Criação de API

- Dentro da pasta “livros”, criar uma pasta chamada “api”;
- Dentro da pasta “api”, criar dois arquivos:
 - serializers.py
 - viewsets.py



Criação de API

▼	livros	●
>	__pycache__	●
▼	api	●
	 serializers.py	U
	 viewsets.py	U



Criação de API

- Os serializers do DRF são componentes essenciais do framework.
- Eles servem para traduzir entidades complexas, como querysets e instâncias de classes em representações simples que podem ser usadas no tráfego da web, como JSON e XML.
- Esse processo é chamado de Serialização.



Criação de API

- Serializers também servem para fazer o caminho contrário: a Desserialização.
- Isto é, transformar representações simples (como JSON e XML) em representações complexas, instanciando objetos, por exemplo.



Criação de API

- As ViewSets definem quais operações REST estarão disponíveis e como seu sistema vai responder às chamadas à sua API.
- Em outros frameworks, são chamados de Resources ou Controllers.
- ViewSets herdam e adicionam lógica às Views padrão do Django.



Criação de API

- ◎ Suas responsabilidades são:
 - Receber os dados da Requisição (formato JSON ou XML)
 - Validar os dados de acordo com as regras definidas na modelagem e no Serializer
 - Desserializar a Requisição e instanciar objetos
 - Processar regras de negócio (aqui é onde implementamos a lógica dos nossos sistemas)
 - Formular uma resposta e responder a quem chamou sua API



Criação de API

- Abrir o arquivo `serializers.py` e deixar assim:



Criação de API

biblioteca > livros > api >  serializers.py >  LivrosSerializer >  Meta

```
1  ∨ from rest_framework import serializers
2  ∨ from livros import models
3
4  ∨ class LivrosSerializer(serializers.ModelSerializer):
5  ∨     class Meta:
6  ∨         model = models.Livros
7  ∨         fields = '__all__'
```



Criação de API

- Abrir o arquivo `viewsets.py` e deixar assim:



Criação de API

biblioteca > livros > api >  viewsets.py >  LivrosViewSet

```
1  from rest_framework import viewsets
2  from livros.api import serializers
3  from livros import models
4
5  class LivrosViewSet(viewsets.ModelViewSet):
6      serializer_class = serializers.LivrosSerializer
7      queryset = models.Livros.objects.all()
```



Criação de API

- Na pasta biblioteca, abrir o arquivo “urls.py” e deixar assim:



Criação de API

```
16  ✓ from django.contrib import admin
17    from django.urls import path, include
18
19    from rest_framework import routers
20
21    from livros.api import viewsets as livrosviewsets
22
23    route = routers.DefaultRouter()
24    route.register(r'livros', livrosviewsets.LivrosViewSet, basename="Livros")
25
26  ✓ urlpatterns = [
27      path('admin/', admin.site.urls),
28      path('', include(route.urls))
29  ]
30
```



Criação de API

- Os Routers nos auxiliam na geração das URLs da nossa aplicação.
- Como o REST possui padrões bem definidos de estrutura de URLs, o DRF as gera automaticamente para nós, já no padrão correto.



Criação de API

- No terminal, digitar o comando:
`py manage.py runserver`



Criação de API

Django REST framework

Api Root

Api Root

The default basic root view for DefaultRouter

OPTIONS GET

GET /

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{  
  "livros": "http://127.0.0.1:8000/livros/"  
}
```



Criação de API

- Clicar no link;



Criação de API

Livros List

OPTIONS

GET

GET /livros/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[]

Raw data

HTML form

Título

Autor

Ano lançamento

Estado

Páginas

Editora

POST



Criação de API

- Cadastrar um livro;



Criação de API

Api Root / Livros List

Livros List

OPTIONS GET ▾

GET /livros/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
[
  {
    "id_livro": "d26136a0-09bf-41e9-b138-dd3cc0651f73",
    "titulo": "Título do livro",
    "autor": "Autor do livro",
    "ano_lancamento": 2022,
    "estado": "Novo",
    "paginas": 250,
    "editora": "Editora do livro",
    "dt_criacao": "2023-05-10T23:41:35.955621Z"
  }
]
```



Criação de API

- Pegar o valor de “id_livro” e adicionar no final da url;
- Por exemplo:
`http://127.0.0.1:8000/livros/d26136a0-09bf-41e9-b138-dd3cc0651f73/`



Criação de API

Livros Instance

DELETE OPTIONS GET ▾

GET /livros/d26136a0-09bf-41e9-b138-dd3cc0651f73/

HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "id_livro": "d26136a0-09bf-41e9-b138-dd3cc0651f73",
  "titulo": "Titulo do livro",
  "autor": "Autor do livro",
  "ano_lancamento": 2022,
  "estado": "Novo",
  "paginas": 250,
  "editora": "Editora do livro",
  "dt_criacao": "2023-05-10T23:41:35.955621Z"
}
```

Raw data HTML form



Criação de API

- No terminal, executar o comando:
`py -m pip install pillow`



Criação de API

- Na pasta “biblioteca”, abrir o arquivo “settings.py” e deixar assim:



Criação de API

```
15 # Build paths inside the project like this: BASE_DIR / 'subdir'.  
16 BASE_DIR = Path(__file__).resolve().parent.parent  
17  
18 MEDIA_URL = "/uploads/"  
19 MEDIA_ROOT = BASE_DIR
```



Criação de API

- Na pasta “biblioteca”, abrir o arquivo “urls.py” e deixar assim:



Criação de API

```
16 ✓ from django.contrib import admin
17   from django.urls import path, include
18
19   from rest_framework import routers
20
21   from livros.api import viewsets as livrosviewsets
22
23   from django.conf.urls.static import static
24   from django.conf import settings
25
26   route = routers.DefaultRouter()
27   route.register(r'livros', livrosviewsets.LivrosViewSet, basename="Livros")
28
29 ✓ urlpatterns = [
30     path('admin/', admin.site.urls),
31     path('', include(route.urls))
32 ]
33 urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```



Criação de API

- Na pasta “livros”, abrir o arquivo “models.py” e deixar assim:



Criação de API

```
1  ✓ from django.db import models
2    from uuid import uuid4
3
4    # Create your models here.
5  ✓ class Livros(models.Model):
6      id_livro = models.UUIDField(primary_key=True, default=uuid4, editable=False)
7      titulo = models.CharField(max_length=255)
8      autor = models.CharField(max_length=255)
9      ano_lancamento = models.IntegerField()
10     estado = models.CharField(max_length=50)
11     paginas = models.IntegerField()
12     editora = models.CharField(max_length=255)
13     dt_criacao = models.DateTimeField(auto_now_add=True)
14     imagem = models.ImageField(upload_to='uploads/', default='')
```



Criação de API

- No terminal, rodar o comando:
`py manage.py makemigrations`
- Depois:
`py manage.py migrate`
- Depois:
`py manage.py runserver`



Criação de API

- Fazer o cadastro de 5 livros;



Criação de API

- Dever de casa:
- Fazer o deploy dessa aplicação;



Obrigado!

Alguma pergunta?

Contato:

☎ 040601692@prof.unama.br