



O QUE É O DJANGO REST FRAMEWORK (DRF)?

Guia completo Django REST Framework (DRF): serializers, viewsets, authentication, permissions, browsable API. Aprenda a criar APIs RESTful profissionais com Django 5.1.

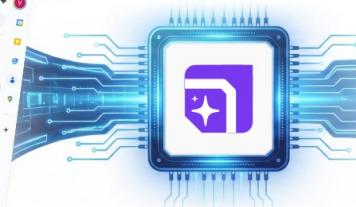
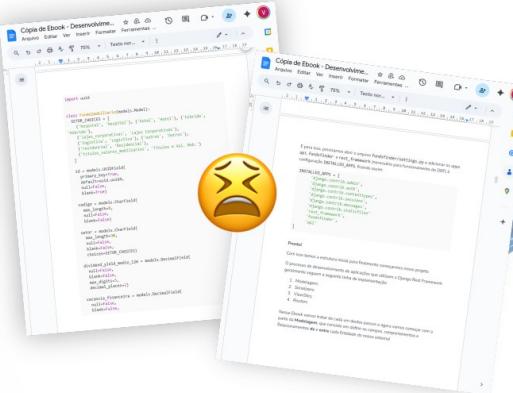
Gere ebooks como este com



Ebookr

em <https://ebookr.ai>

Crie ebooks profissionais incríveis em minutos com IA



Chega de formatar texto no Google Docs, Word ou ferramentas que só te fazem perder tempo...

E deixe que nossa IA faça o trabalho pesado!



Capas gerados por IA



Infográficos feitos por IA



Edite em Markdown em Tempo Real



Adicione Banners Promocionais

TESTE AGORA



PRIMEIRO CAPÍTULO 100% GRÁTIS

 **Artigo atualizado para Django REST Framework 3.15** (Dezembro 2025)

Compatível com Django 5.1. DRF continua sendo o padrão para APIs REST em Python.

Salve salve Pythonista!

No desenvolvimento moderno de aplicações web, a criação de APIs RESTful é uma habilidade essencial.

O **Django Rest Framework (DRF)** é uma poderosa ferramenta que facilita e agiliza esse processo.

Neste artigo, vamos explorar **o que é o Django Rest Framework**, como ele simplifica o desenvolvimento de APIs REST, as principais funcionalidades que ele oferece e como ele pode aumentar significativamente a eficiência e a robustez do desenvolvimento de aplicações Web RESTful em Python.

Ah, e esse artigo dá o *start* em uma sequência incrível de artigos sobre essa importante e requisitada ferramenta!

Então fica ligado e não perca os próximos artigos! 😊

Mas primeiro...

O Django REST Framework utiliza - como o próprio nome sugere - o nosso querido Django como base!

Por isso é interessante que você já possua certo conhecimento na construção de aplicações web utilizando o Django.

Ainda não é **craque** em Django?

Não tem problema, aqui na Python Academy você conta com o melhor material sobre essa importante ferramenta!

Acesse agora:

[Django: Introdução ao framework](#)

[Django: A Camada Model](#)

[Django: A Camada View](#)

[Django: A Camada Template](#)

Ou se preferir, baixe nosso ebook **GRÁTIS** de Desenvolvimento Web com Python e Django:

Sem mais delongas, vamos para o Django REST Framework, ou DRF para os íntimos.

O que é o Django Rest Framework?

O **Django Rest Framework (DRF)** é uma biblioteca de alta performance, flexível e robusta para criar **APIs RESTful** usando **Django**.

Ele fornece um kit de ferramentas de ponta para criar APIs, oferecendo uma série de vantagens como autenticação, permissões, serialização e muito mais.

A integração com Django é perfeita, tornando-o ideal para desenvolvedores que já estão familiarizados com o ecossistema Django.

 **DRF 3.15+** adiciona suporte melhorado para **async views**, validações aprimoradas e melhor integração com Django 5.x.

Como o Django Rest Framework facilita o desenvolvimento de APIs REST

Vamos ver algumas das várias vantagens de se utilizar o DRF para construção de APIs Rest:

Serialização de Dados

A **serialização** é um conceito crucial ao trabalhar com APIs.

Ela permite converter complexos tipos de dados do Django para formatos JSON, XML, etc., e vice-versa.

O DRF facilita essa tarefa com suas poderosas classes de serialização.

Veja como é fácil criar uma classe de Serialização para converter um Model do Django - chamado de `Livro` - e seus campos: `titulo`, `autor`, `isbn` e `publicacao`:

```
from rest_framework import serializers
from .models import Livro

class LivroSerializer(serializers.ModelSerializer):
    class Meta:
        model = Livro
        fields = ['titulo', 'autor', 'isbn', 'publicacao']
```

Views Genéricas Baseadas em Classes

O DRF oferece um conjunto de **Views Genéricas** que simplificam a criação de operações CRUD.

Usar essas views genéricas economiza muito tempo e reduz a complexidade do código:

```
from rest_framework import generics
from .models import Livro
from .serializers import LivroSerializer

class LivroListCreateView(generics.ListCreateAPIView):
    queryset = Livro.objects.all()
    serializer_class = LivroSerializer
```

No exemplo acima, é utilizada a `generics.ListCreateAPIView` para criar uma View que vai possibilitar a Listagem de Livros (o *Retrieve* do CRUD) e a Criação de Livros (o *Create* do CRUD).

Nenhum código adicional é necessário! Essa é a **mágica!**

Autenticação e Permissões

Outra área onde o DRF brilha é na **autenticação e controle de permissões**.

Ele oferece diversos métodos de autenticação e uma infraestrutura fácil de usar para definir permissões que controlam o acesso às diferentes partes da API.

Vamos a um exemplo:

```
from rest_framework.permissions import IsAuthenticated
from rest_framework.decorators import api_view, permission_classes

@api_view(['GET'])
@permission_classes([IsAuthenticated])
def lista_livros(request):
    livros = Livro.objects.all()
    serializer = LivroSerializer(livros, many=True)
    return Response(serializer.data)
```

Explicações:

- Nesse código é utilizado o decorator `@api_view` definindo que a View só permite o método `HTTP GET`.
- O decorator `@permission_classes` define os tipos permitidos de autenticação para essa View. `IsAuthenticated` define que essa View só estará disponível para usuários já autenticados no sistema.
- `def lista_livros` define uma Function Based View que irá buscar todos os registros de Livros (com `Livro.objects.all()`), serializá-los (com `LivroSerializer(livros, many=True)`) e formula uma Resposta HTTP com os dados serializados (em `Response(serializer.data)`).

Além de oferecer suporte a diversos mecanismos de autenticação como **token-based authentication**, **session-based authentication**, **OAuth**, entre outros.

Browsable API

Uma das funcionalidades mais incríveis do DRF é sua **Interface Navegável** (do inglês *Browsable API*), que permite testar e interagir com sua API diretamente pelo navegador, tornando o desenvolvimento e o debug muito mais simples.

Nela você pode:

- Acessar os dados cadastrados;
- Testar sua API, realizando chamadas HTTP pelo próprio browser;
- Possui autenticação e autorização integrados;
- Possui uma ótima e amigável interface.

Por exemplo, a seguinte imagem é um exemplo dessa interface:

The screenshot shows the Django REST framework browsable API interface. At the top, there's a header bar with "Django REST framework" on the left and "Log in" on the right. Below the header, a navigation bar has "Api Root" selected. The main content area is titled "Api Root". To the right of the title are two buttons: "OPTIONS" and "GET" with a dropdown arrow. Below the title, it says "The default basic root view for DefaultRouter". Underneath, there's a button labeled "GET /api/v1/". Further down, there's a section showing the response headers: "HTTP 200 OK", "Allow: GET, HEAD, OPTIONS", "Content-Type: application/json", and "Vary: Accept". Finally, the JSON response body is shown: "{ \"fundos\": \"http://127.0.0.1:8000/api/v1/fundos\" }".

E também:

[Api Root](#) / Fundo Imobiliario List

Fundo Imobiliario List

[OPTIONS](#)[GET](#)[GET /api/v1/fundos](#)

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

[]

[Raw data](#)[HTML form](#)

Id

Codigo

Setor

 HospitalDividend yield
medio 12mVacancia
financeira

Vacancia fisica

Quantidade ativos

[POST](#)

E isso tudo sem desenvolver uma linha sequer de código HTML, CSS ou Javascript!

Incrível não é mesmo?! 😍

Filtragem e Paginação

O DRF facilita a **filtragem** e **paginação** dos dados, o que é essencial para a performance e experiência do usuário em APIs que lidam com grandes volumes de dados.

Veja como é simples adicionar, por exemplo, a paginação à respostas de sua API:

```
from rest_framework.pagination import PageNumberPagination

class LivroPagination(PageNumberPagination):
    page_size = 10

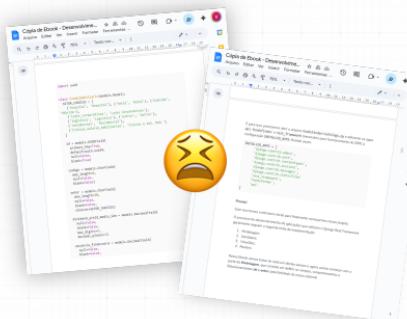
class LivroListView(generics.ListAPIView):
    queryset = Livro.objects.all()
    serializer_class = LivroSerializer
    pagination_class = LivroPagination
```

Apenas herdando de `PageNumberPagination` e configurando `page_size` você já é capaz de adicionar paginação à sua API!



Estou construindo o [Ebookr.ai](#), uma plataforma onde você cria ebooks profissionais com IA sobre qualquer assunto — do zero ao PDF pronto, com capas e infográficos gerados automaticamente. Dá uma olhada!

Crie Ebooks profissionais incríveis em minutos com IA



Chega de formatar texto no Google Docs, Word ou ferramentas que só te fazem perder tempo...



... e deixe que nossa IA faça o trabalho pesado!

TESTE AGORA! PRIMEIRO CAPÍTULO 100% GRÁTIS 

 Capas gerados por IA

 Adicione Banners Promocionais

 Edite em Markdown em Tempo Real

 Infográficos feitos por IA

Throttling

O **throttling** limita a taxa de pedidos a uma API, prevenindo abuso e evitando sobrecarregar o servidor.

Muito importante para proteger sua aplicação e evitar que seu servidor seja “derrubado” por algum cliente que esteja tentando acessar sua API mais do que o esperado.

Testes Integrados

O DRF também fornece ferramentas para facilitar os testes, como **APITestCase**, que estende as funcionalidades do Django **TestCase**.

```
from rest_framework.test import APITestCase

class LivroAPITests(APITestCase):
    def test_lista_livros(self):
        response = self.client.get('/api/livros/')
        self.assertEqual(response.status_code, 200)
```

Dessa forma, criar testes se torna uma tarefa simples!

Documentação Integrada

O DRF possui diversas ferramentas para geração automática de documentação da sua API.

Ao integrar com ferramentas como **Swagger** ou **CoreAPI**, o DRF permite gerar documentação automática e interativa com pouco trabalho adicional.

Conclusão

Neste artigo, exploramos **o que é o Django Rest Framework** e *como ele facilita o desenvolvimento de APIs RESTful*.

Também vimos algumas de suas **principais funcionalidades**, como serialização, autenticação, permissões e roteamento automatizado.

O uso do DRF pode transformar a maneira como você desenvolve APIs, tornando o processo mais rápido e eficiente.

Se você trabalha com Django e precisa criar APIs robustas e escaláveis, o **Django Rest Framework é uma ferramenta indispensável**.

Fique ligado nos próximos artigos, vamos aprender muito mais sobre cada característica que vimos aqui!

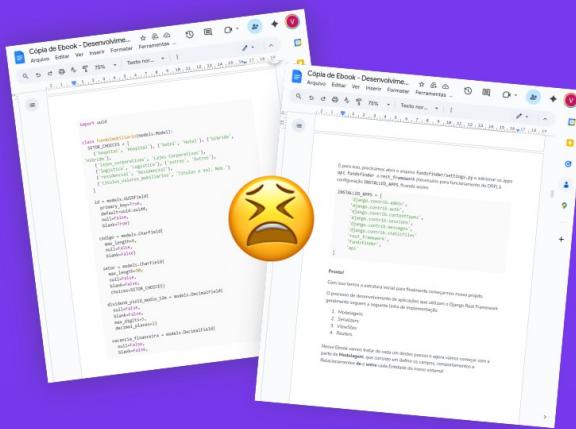
Até lá, dev.

Não se esqueça de conferir!



Ebookr

Crie Ebooks profissionais em minutos com IA



Chega de formatar código no Google Docs ou Word



Arquitetura de Software Moderna

A arquitetura de software alvina le professional contens nel eandio e producions de software para argionemntoquios. Ostante oreosczmas, camione-quboles a comitit pessima ia arquitetura de software moderna.

```
import python
import python

class Arquitetura_de_Software_Moderna:
    ...
    def share(self):
        pass
    ...
    return "Arquitetura de Net", "civilizedness"
    ...

    def __init__(self):
        if user_is_adult():
            self.user_is_adult = True
        else:
            self.user_is_adult = False
        self.user_is_adult = True
    ...
    # Esse é o cor de opinião am cor
    return type
    ...
    return saabelló
```

AI-generated system

A qualifetura com prouitiv alitema software aa medie de fusulan moderna. Sesemtos simcasxabs coneciteta mcolia entroicodes eterna. Chaoao e anoxes dialeka AI-generated si generated system qfemonia copiente enemot.

Clean layout

Gentilmente Alia maticot en turbacit evicticos that alion ossibid to coenize Inugn with oqacitien on onces dibos. Net layout in gremamento formataceo exmoa um dñivoura exzistem foa a miltibd diginucleus, poiso ee dñor alour fumil.

Capas gerados por IA

Infográficos feitos por IA

Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado

Edite em Markdown em Tempo Real

TESTE AGORA

PRIMEIRO CAPÍTULO 100% GRÁTIS