



# FILTROS, BUSCA E ORDENAÇÃO NO DJANGO REST FRAMEWORK

Esse ebook mostra como estender as funcionalidades de sua API, adicionando Filtros, Busca Textual e Ordenação às APIs desenvolvidas com o poderoso Django REST Framework

# Crie Ebooks técnicos incríveis em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código!**



Chega de formatar código no Google Docs



Deixe que nossa IA faça o trabalho pesado

 Syntax Highlight

 Adicione Banners Promocionais

 Edite em Markdown em Tempo Real

 Infográficos feitos por IA

**TESTE AGORA** 

Faaaaala **DEV**!

Estamos juntos para mais um passeio no parque do nosso querido **Django REST Framework**, ou **DRF**!

Vamos mostrar nesse post como podemos estender nossas APIs para adicionar as funcionalidades de Filtro, Busca Textual e Ordenação.

Essas funcionalidades são muito importantes e veremos aqui como são simples de implementar no DRF.

**Ahhh, caso não tenha visto nosso post sobre DRF:** 

*Esse post faz parte da construção de APIs de Fundos Imobiliários iniciada no nosso post sobre [Django REST Framework](#)! Para entender as referências e configurações do projeto, sugiro começar por ele!*

Agora que está craque em DRF, **#VamosNessa!** 

## Introdução

O DRF nos possibilita adicionar as funcionalidades de ordenação de resultados, busca textual e filtragem de dados de maneira **extremamente simples!**

Para nos auxiliar, vamos começar instalando o pacote `django-filter`.

Na raiz do seu projeto, com o ambiente virtual instalado, execute:

```
pip install django-filter
```

Em seguida, adicione `django_filters` à variável `INSTALLED_APPS` do arquivo `settings.py` do projeto.

Feito isso, podemos começar!

## Filtro

Suponha que seja necessário adicionar filtros à nossa API para que os usuários possam fazer, por exemplo:

```
http://localhost:8000/api/v1/fundos?setor=hibrido
```

E nossa API retorne apenas os Fundos em que `setor=hibrido`.

Para utilizar filtros em nossa API, adicionamos os atributos `filter_backends` e `filterset_fields`:

- Em `filter_backends` colocamos o *Backend* que irá processar os filtros;
- Em `filterset_fields` adicionamos quais campos queremos expôr para filtragem.

Com isso, nossa `ViewSet` fica assim:

```

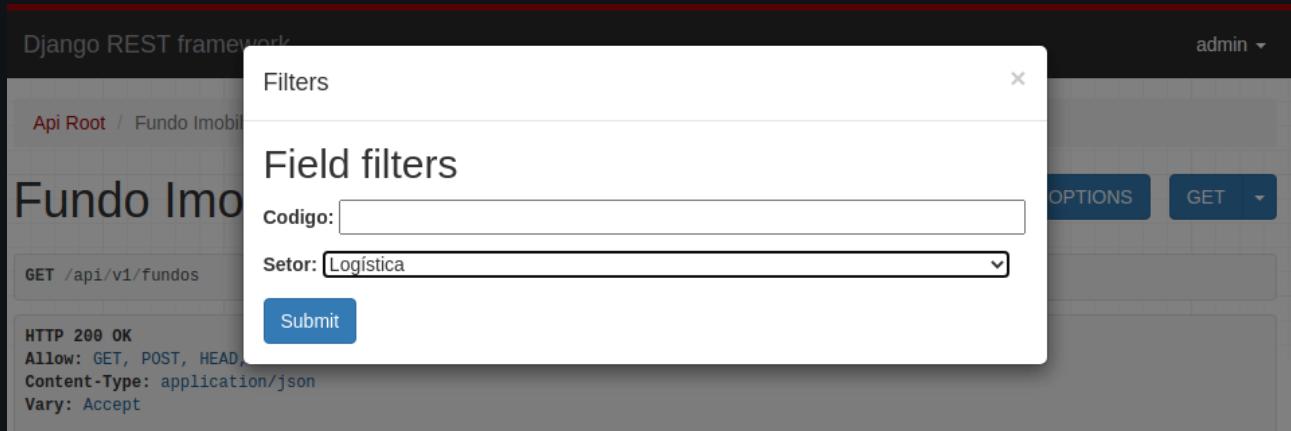
from django_filters.rest_framework import DjangoFilterBackend
from api.serializers import FundoImobiliarioSerializer
from rest_framework import viewsets, permissions
from api.models import FundoImobiliario

class FundoImobiliarioViewSet(viewsets.ModelViewSet):
    queryset = FundoImobiliario.objects.all()
    serializer_class = FundoImobiliarioSerializer
    permission_classes = [permissions.IsAuthenticated]
    filter_backends = [DjangoFilterBackend]
    filterset_fields = ['codigo', 'setor']

```

Dessa forma podemos filtrar Fundos Imobiliários pelo `código` e `setor`!

Agora, abra a interface navegável e veja que o DRF adicionou um botão “*Filters*”: nele podemos testar os filtros!



Teste a URL `http://localhost:8000/api/v1/fundos?setor=hibrido` com o Postman e veja que **show!**

## Busca Textual

A busca textual serve para adicionar a funcionalidade de realizar buscas dentro de determinados valores de texto armazenados na base de dados.

Contudo a busca só funciona para campos de texto, como `CharField` e `TextField`.

Para utilizar a busca textual, devemos promover duas alterações em nossa `ViewSet`:

- Novamente alterar o atributo `filter_backends`, adicionando o *Backend SearchFilter* que irá processar a busca; e
- Adicionar o atributo `search_fields`, contendo os campos que permitirão a busca.

Assim:

```
from django_filters.rest_framework import DjangoFilterBackend
from rest_framework.filters import SearchFilter
from api.serializers import FundoImobiliarioSerializer
from rest_framework import viewsets, permissions
from api.models import FundoImobiliario

class FundoImobiliarioViewSet(viewsets.ModelViewSet):
    queryset = FundoImobiliario.objects.all()
    serializer_class = FundoImobiliarioSerializer
    permission_classes = [permissions.IsAuthenticated]
    filter_backends = [DjangoFilterBackend, SearchFilter]
    filterset_fields = ['codigo', 'setor']
    search_fields = ['codigo', 'setor']
```

Na interface navegável, clique novamente em *Filters* e veja que foi adicionada o campo de busca **Search!**

Django REST framework

admin ▾

Api Root / Fundo Imóvel

## Fundo Imóvel

GET /api/v1/fundos

HTTP 200 OK

Allow: GET, POST, HEAD, PUT, PATCH, DELETE

Content-Type: application/json

Vary: Accept

```
[{"id": "c41c2a9...", "codigo": "MAX...", "setor": "hibrido", ...}]
```

OPTIONS GET

Nele você pode realizar busca por valores contidos em `codigo` (ex: ALZR11, HGLG11, etc) e `setor` (ex: hibrido, hospital, residencial, etc).

**DEMAIS!**

## Ordenação

E por último, e não menos importante: a **Ordenação**.

Essa funcionalidade serve para que possamos definir a ordem que os resultados devem ser apresentados.

A funcionalidade de Ordenação é configurada de forma semelhante aos Filtros e à Busca Textual. Assim:

- Adicionar o *Backend* `OrderingFilter` ao atributo `filter_backends` ;
- Adicionar o atributo `ordering_fields` , que são quais campos poderão ser ordenados
- Adicionar opcionalmente o atributo `ordering` que configura a ordenação que será aplicada por padrão ao chamar endpoints que retornem mais de um resultado.

Portanto, nossa versão final da nossa `ViewSet` de Fundos Imobiliários deverá ficar assim:

```
from django_filters.rest_framework import DjangoFilterBackend
from rest_framework.filters import SearchFilter, OrderingFilter
from api.serializers import FundoImobiliarioSerializer
from rest_framework import viewsets, permissions
from api.models import FundoImobiliario

class FundoImobiliarioViewSet(viewsets.ModelViewSet):
    queryset = FundoImobiliario.objects.all()
    serializer_class = FundoImobiliarioSerializer
    permission_classes = [permissions.IsAuthenticated]
    filter_backends = [DjangoFilterBackend, SearchFilter, OrderingFilter]
    filterset_fields = ['codigo', 'setor']
    search_fields = ['codigo', 'setor']
    ordering = ['-dividend_yield_medio_12m']
    ordering_fields = [
        'dividend_yield_medio_12m',
        'setor',
        'vacancia_financeira',
        'vacancia_fisica',
        'quantidade_ativos']
```

Veja que a Interface Navegável já foi atualizada e adicionou a seção *Ordering*, com os campos configurados:

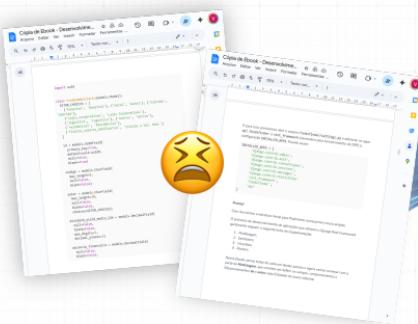
The screenshot shows the Django REST framework's browsable API interface for a 'Fundo Imobiliário' endpoint. At the top, there are 'OPTIONS' and 'GET' buttons. Below them, a 'Filters' section contains fields for 'Codigo:' and 'Setor:', both with dropdown menus. A 'Submit' button is also present. Underneath, a 'Search' bar with a magnifying glass icon and the word 'Search' is shown. The main area displays a JSON response with three items, each containing fields like 'id', 'codigo', 'setor', 'dividend\_yield', 'vacancia\_financeira', 'vacancia\_fisica', and 'quantidade\_ativos'. Below the JSON, an 'Ordering' section lists various sorting options, with 'dividend\_yield\_medio\_12m - descending' selected and highlighted with a blue background and a checkmark.

✨ **WOW!!!** ✨

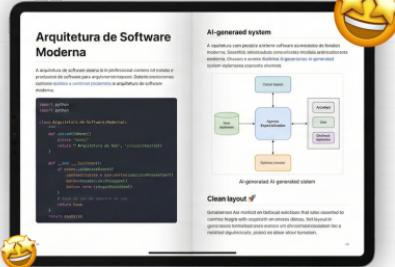
💡 *Estou desenvolvendo o **DevBook**, uma plataforma que usa IA para gerar ebooks técnicos profissionais. Te convido a conhecer clicando no botão abaixo!*

## Crie Ebooks técnicos incríveis em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código!**



Chega de formatar código no Google Docs



Deixe que nossa IA faça o trabalho pesado

 Syntax Highlight

 Adicione Banners Promocionais

 Edite em Markdown em Tempo Real

 Infográficos feitos por IA

**TESTE AGORA! PRIMEIRO CAPÍTULO 100% GRÁTIS** 

## Conclusão

Como sempre, o Django sendo o **MELHOR framework** para Desenvolvimento Web!

Vimos nesse post como é simples estender e adicionar as funcionalidades de Filtros, Busca Textual e Ordenação à nossa API construída com o DRF!

*E você, o que achou? Curte o DRF? Tem sugestão de conteúdo?*

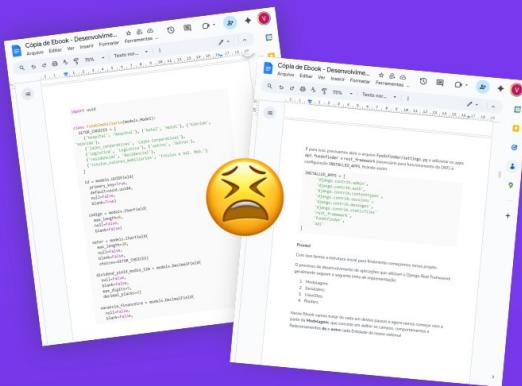
**Conta aqui embaixo pra gente!** 😊

Até mais!!!



# Crie Ebooks técnicos em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código!**



Chega de formatar código no Google Docs



**Arquitetura de Software Moderna**

A arquitetura de software alvo é profissional contendo o e-mail e produções de software para arquiteturas modernas. Oferece recursos como interface gráfica com interface de usuário.

```
import python
import python

class Arquitetura_de_Software_Moderna:
    ...
    def share(self):
        pass
    ...
    return "Arquitetura de NeXt", "arquitetura_moderna"
}

def __init__(self):
    if user_authenticated():
        self.user_authenticated = user_authenticated()
        self.user_email = self.user_authenticated['email']
        self.user_name = self.user_authenticated['name']
    ...
    # Envie AI para gerar um código
    return type
}
resource_available
```

**AI-generated system**

A arquitetura com propósito alvo é software amigável de usuários modernos. Seus recursos incluem conceitos de interface gráfica de usuário moderna. Oferece recursos para implementar.

**Infográficos feitos por IA**



Infográficos feitos por IA

</> Syntax Highlight



Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado



Edite em Markdown em Tempo Real

**TESTE AGORA**

PRIMEIRO CAPÍTULO 100% GRÁTIS