



# O QUE É E COMO USAR O PAINEL ADMINISTRATIVO DO DJANGO (PYTHON)

Tutorial completo do Django Admin: aprenda a configurar, customizar e usar o painel administrativo. ModelAdmin, list\_display, search\_fields, list\_filter e mais no Django 5.1.

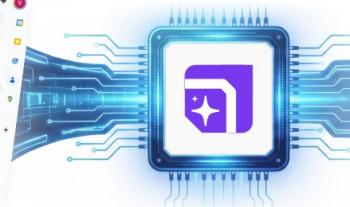
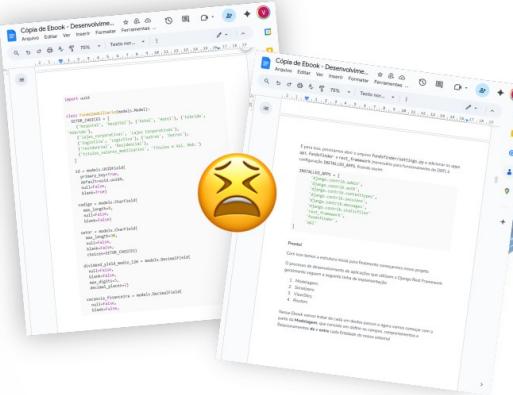
Gere ebooks como este com



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 5.1** (Dezembro 2025) Django Admin continua poderoso. Django 5.1 adiciona tema dark mode e melhorias de acessibilidade.

Salve salve Pythonista!

O Django é um poderoso framework para desenvolvimento web, escrito em Python, que oferece uma ampla gama de recursos para facilitar a criação de aplicações web de alta qualidade.

Uma das características mais notáveis do Django é seu Painel Administrativo, uma interface pré-construída que permite aos administradores do site gerenciar e visualizar facilmente os dados da aplicação.

Neste artigo, vamos explorar o que é o Painel Administrativo do Django e como usá-lo!

Vamos criar uma aplicação exemplo com modelos, mostrar todos os passos necessários para ter a interface administrativa do Django configurada e explicar o uso do `ModelAdmin` para personalizar o painel.

Então, bora nessa!

Antes de começarmos, é importante ressaltar que o Django já vem com o Painel Administrativo pré-instalado e pronto para ser utilizado.

E se você ainda não fez a configuração inicial do seu projeto Django, já [corre aqui nesse artigo](#) para iniciar os primeiros passos no Django

Portanto, você não precisa escrever código extra para habilitá-lo.

Vamos começar criando uma aplicação exemplo para que possamos explorar as funcionalidades do Painel Administrativo.

## Criando uma aplicação exemplo

Primeiramente, vamos criar um novo projeto Django. Abra um terminal e execute o seguinte comando:

```
django-admin startproject meu_projeto
```

Em seguida, navegue até a pasta do projeto:

```
cd meu_projeto
```

Agora, vamos criar uma nova aplicação dentro do projeto:

```
python manage.py startapp website
```

## Definindo os modelos

Dentro da pasta da aplicação `website`, abra o arquivo `models.py` e defina alguns modelos.

Por exemplo, vamos criar um modelo de `Post` e um modelo de `Comentário`:

```
from django.db import models

class Post(models.Model):
    titulo = models.CharField(max_length=200)
    conteudo = models.TextField()
    data_publicacao = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.titulo

class Comentario(models.Model):
    texto = models.TextField()
    post = models.ForeignKey(Post, on_delete=models.CASCADE)

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

Isso irá criar um Modelo `Post` com os campos de texto `titulo`, `conteudo` e `data_publicacao` (com `auto_now_add` igual à `True`, ou seja, ao adicionar um novo registro, esse campo utilizará a data e hora atuais).

Em seguida, criará também um Modelo `Comentario` com um campo de texto, chamado `texto`, e uma *Foreign Key* `post` (chave estrangeira) apontando para a classe `Post`.

## Configurando o banco de dados

Antes de prosseguir, é necessário configurar o banco de dados para que o Django possa armazenar os dados dos modelos.

Abra o arquivo `settings.py` no diretório do projeto e localize a configuração `DATABASES`.

Aqui, nós configuramos de acordo com o banco de dados que estamos usando na nossa aplicação.

Por exemplo, para usar o SQLite - que é o Banco de Dados mais básico e muito utilizado durante o desenvolvimento de aplicações mais complexas - a configuração ficaria assim:

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3',  
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),  
    }  
}
```

## Aplicando as migrações

Agora, é necessário aplicar as migrações para criar as tabelas do banco de dados. Execute o seguinte comando no terminal:

```
python manage.py makemigrations website  
python manage.py migrate
```

Para saber mais sobre o comando `makemigrations` do Django, [clique aqui](#).

E se quiser saber mais sobre o comando `migrate` do Django, [clique aqui](#).

## Criando um superusuário

Antes de acessar o Painel Administrativo, você precisa criar um superusuário. Execute o seguinte comando e siga as instruções:

```
python manage.py createsuperuser
```

Siga as instruções e guarde as credenciais que acabou de criar.

## Iniciando o servidor de desenvolvimento

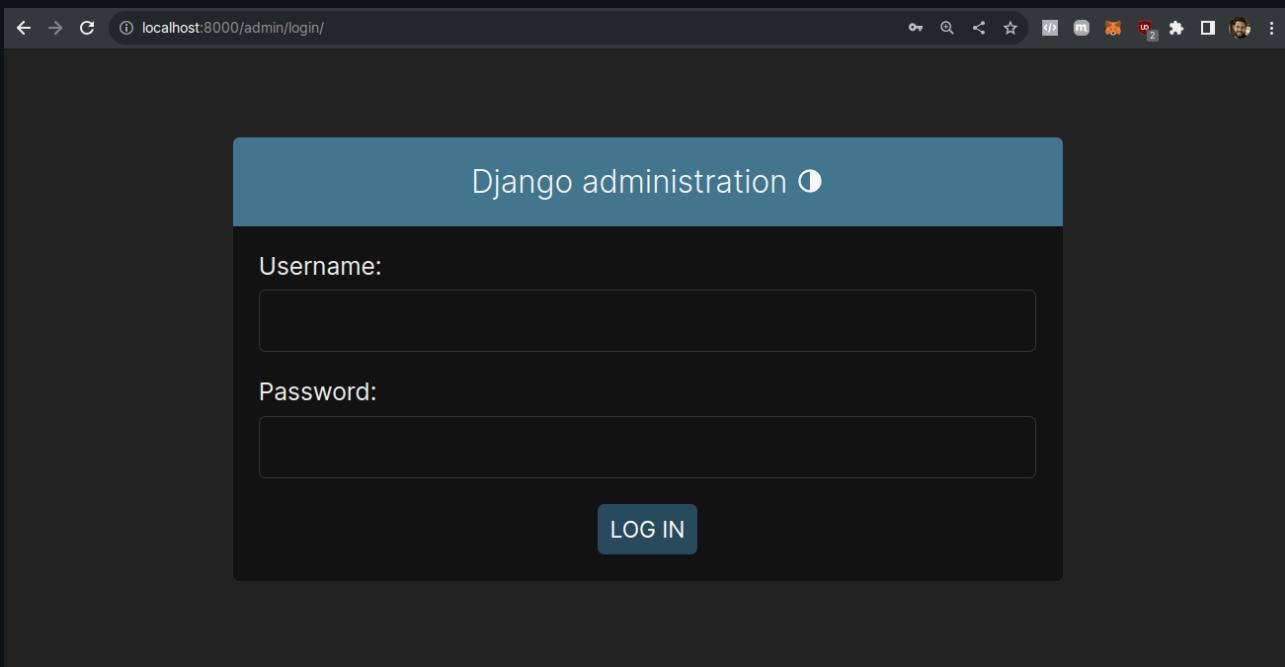
Agora, vamos iniciar o servidor de desenvolvimento do Django para testar nossa aplicação:

```
python manage.py runserver
```

Se tudo estiver correto, você verá uma saída semelhante a esta:

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Acesse o link <http://127.0.0.1:8000/admin> no seu navegador. Você deverá ver uma tela como essa:

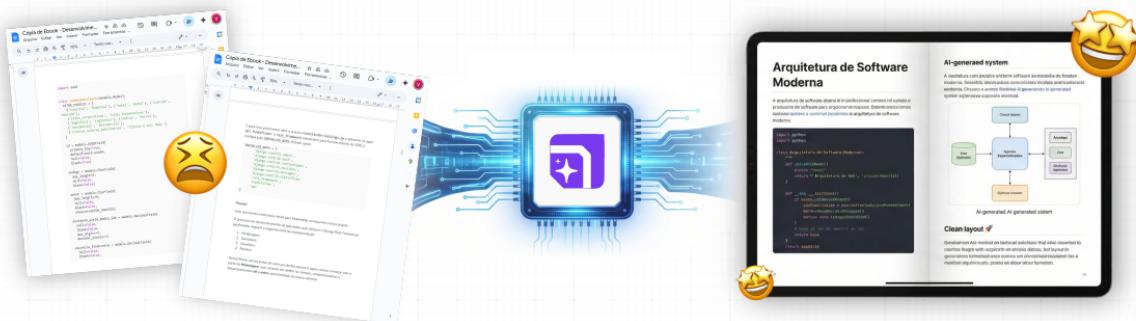


E então faça login usando as credenciais do superusuário que você acabou de criar.

 Criei o **Ebookr.ai**, uma plataforma que usa IA para gerar ebooks profissionais sobre qualquer tema — com capa gerada por IA, infográficos automáticos e exportação em PDF. Confere!



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

## Explorando o Painel Administrativo

Ao fazer login, você será redirecionado para o Painel Administrativo do Django, que tem a seguinte cara:

The screenshot shows the Django Admin homepage. At the top, it says "Django administration" and "WELCOME, VINICIUS. VIEW SITE / CHANGE PASSWORD / LOG OUT". Below this is a "Site administration" header. On the left, there's a sidebar with "AUTHENTICATION AND AUTHORIZATION" sections for "Groups" and "Users", each with "+ Add" and "Change" buttons. To the right, there's a "Recent actions" section which is currently empty, stating "None available".

Navegue por esta página e veja o que é possível fazer a partir daqui.

Mas vai perceber que os Modelos que criamos - `Post` e `Comentario` - não estão presentes.

Isso acontece pois é necessário primeiro fazer o registro dessas entidades para que o Django as mostre no Painel Admin.

Vamos fazer isso dentro da pasta do App `website` que criamos com o comando `startapp`.

Mais especificamente no arquivo `admin.py`:

```
from django.contrib import admin
from website.models import Post, Comentario

class PostAdmin(admin.ModelAdmin):
    pass

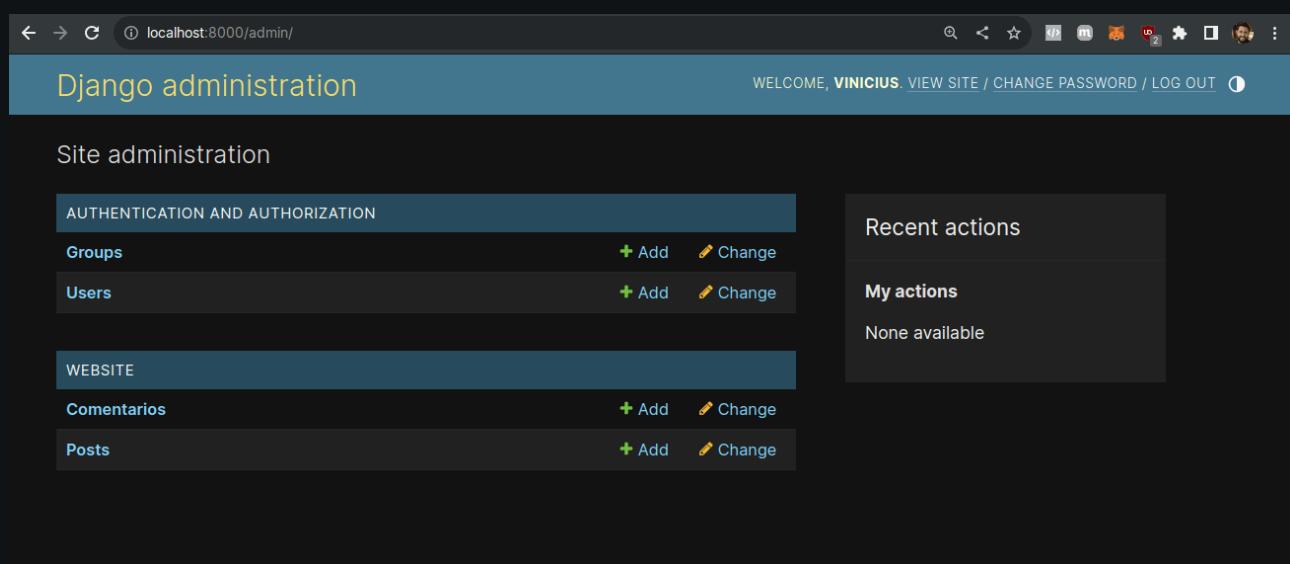
class ComentarioAdmin(admin.ModelAdmin):
    pass

admin.site.register(Post, PostAdmin)
admin.site.register(Comentario, ComentarioAdmin)
```

Nele:

- Criamos duas classes administrativas - `PostAdmin` e `ComentarioAdmin` - que representam nossas próprias entidades dentro do painel administrativo. Elas devem herdar de `django.contrib.admin.ModelAdmin` e estão vazias, por enquanto.
- Depois disso, é necessário realizar o registro com a função `admin.site.register()`, linkando a entidade do sistema (lá do `models.py`) com a entidade administrativa.

Com isso e atualizando a página inicial do painel administrativo teremos, **voilá**:



The screenshot shows the Django administration interface in dark mode. The top navigation bar is blue with white text, displaying 'localhost:8000/admin/' and the user 'WELCOME, VINICIUS'. Below the header, the title 'Django administration' is shown in a light blue bar. The main content area is divided into sections: 'AUTHENTICATION AND AUTHORIZATION' (Groups, Users) and 'WEBSITE' (Comentários, Posts). Each section has 'Add' and 'Change' buttons. To the right, there's a sidebar with 'Recent actions' (empty), 'My actions' (empty), and a message 'None available'. The overall theme is dark with light-colored text and buttons.

Olha que tá ali: `Post` e `Comentario`.

Dando uma fuçada, verá que é possível adicionar, remover, atualizar e visualizar Posts e Comentários, tudo pela própria interface do Django.

Isso é I-N-C-R-Í-V-E-L!

**Chupa Flask!** cof cof... Desculpa 😁

**💡 Django 5.1:** O admin agora suporta **dark mode** automático e melhorias de acessibilidade (WCAG 2.1 AA).

**Mas calma que tem mais!**

Existem várias maneiras de personalizar a exibição dos modelos.

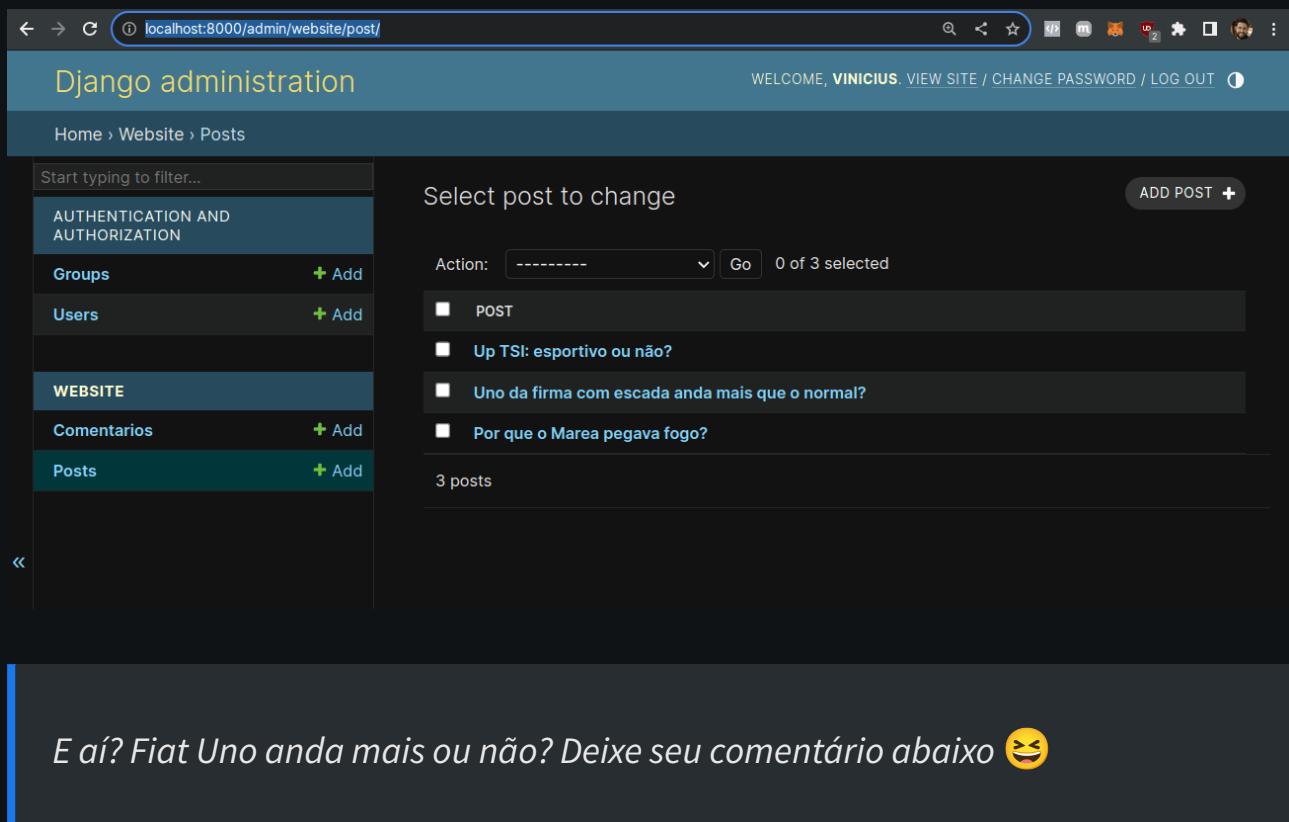
Vamos ver algumas 

# Personalizando o Painel Administrativo com o ModelAdmin

Agora chegou a melhor parte deste artigo!

O Django oferece a possibilidade de personalizar a exibição do Painel Administrativo usando a classe `ModelAdmin`.

Após adicionar alguns Posts, nossa interface ficará assim (perceba que apenas o título é mostrado):



The screenshot shows the Django Admin interface at `localhost:8000/admin/website/post/`. The title bar says "Django administration". The top right has links for "WELCOME, VINICIUS", "VIEW SITE / CHANGE PASSWORD / LOG OUT". The left sidebar has sections for AUTHENTICATION AND AUTHORIZATION (Groups, Users) and WEBSITE (Comentarios, Posts). The main area is titled "Select post to change" and shows a list of posts with their titles: "POST", "Up TSI: esportivo ou não?", "Uno da firma com escada anda mais que o normal?", and "Por que o Marea pegava fogo?". There are 3 posts selected. Action buttons include "ADD POST" and "Go". A note at the bottom says "E aí? Fiat Uno anda mais ou não? Deixe seu comentário abaixo 😊".

Vamos personalizar a exibição do modelo `Post` para adicionar alguns campos adicionais.

Dentro do arquivo `admin.py` da sua aplicação `website`, altere o `PostAdmin` para o seguinte código:

```
class PostAdmin(admin.ModelAdmin):
    list_display = ('titulo', 'conteudo', 'data_publicacao')
```

No exemplo acima, estamos adicionando o campo `data_publicacao` ao `list_display` para exibi-lo na listagem de Posts.

## Adicionando campos à tabela de dados

Com isso, nossa listagem de `Posts` se altera para:

The screenshot shows the Django admin interface at `localhost:8000/admin/website/post/`. The top navigation bar includes links for 'WELCOME, VINICIUS', 'VIEW SITE / CHANGE PASSWORD / LOG OUT'. The left sidebar has sections for 'AUTHENTICATION AND AUTHORIZATION' (Groups, Users) and 'WEBSITE' (Comentarios, Posts). The main content area is titled 'Select post to change' and shows a table with three posts. The columns are 'TITULO', 'CONTEUDO', and 'DATA PUBLICACAO'. The posts listed are:

TITULO	CONTEUDO	DATA PUBLICACAO
Up TSI: esportivo ou não?	E aí, colocou turbo é esportivo? Sim? Não? Comente!	Sept. 14, 2023, 2:19 a.m.
Uno da firma com escada anda mais que o normal?	Nesse artigo, conversamos com especialistas que afirmam que escadas adicionam um coeficiente de arrasto negativo que proveem uma aerodinâmica sobrenatural ao Fiat Uno. Confira!	Sept. 14, 2023, 2:19 a.m.
Por que o Marea pegava fogo?	Vamos desvendar o mistério misterioso dos Fiat Mareas que pegavam fogo DO N-A-D-A!	Sept. 14, 2023, 2:17 a.m.

At the bottom, it says '3 posts'.

Ao recarregar a página de administração do `Post`, você verá os campos `conteudo` e `data_publicacao` sendo exibido na lista de Posts.

## Customizando a Ordenação dos dados

Agora, se quiser configurar a ordenação inicial das linhas da sua tabela, use o `ordering`, da seguinte forma:

```
class PostAdmin(admin.ModelAdmin):
    list_display = ('titulo', 'conteudo', 'data_publicacao')
    ordering = ('-data_publicacao', )
```

Utilizando o símbolo de menos (`-`), dizemos ao Django que queremos ordem descendente. E como se trata de uma data, a mais atual virá primeiro e a mais antiga, por último.

Importante salientar que o Django espera uma Tupla, por isso que usamos os colchetes com vírgula `( '-data_publicacao', )` e não apenas `( '-data_publicacao' )`, o que gera um erro.

## Adicionando campo de busca

Já se quiser adicionar um campo de busca, use o `search_fields`, configurando os campos onde quer realizar a busca, da seguinte forma:

```
class PostAdmin(admin.ModelAdmin):
    list_display = ('titulo', 'conteudo', 'data_publicacao')
    ordering = ('-data_publicacao', )
    search_fields = ('titulo', 'conteudo')
```

Isso fará aparecer um box de busca e um botão acima da sua tabela, que possibilita a busca nos campos definidos em `search_fields`:

Start typing to filter...

WELCOME, VINICIUS. VIEW SITE / CHANGE PASSWORD / LOG OUT

Django administration

Home > Website > Posts

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

WEBSITE

Comentarios + Add

Posts + Add

Select post to change

ADD POST +

Action: ----- Go 0 of 1 selected

	TITULO	CONTEUDO	DATA PUBLICACAO
<input checked="" type="checkbox"/>	Uno da firma com escada anda mais que o normal?	Nesse artigo, conversamos com especialistas que afirmam que escadas adicionam um coeficiente de arrasto negativo que proveem uma aerodinâmica sobrenatural ao Fiat Uno. Confira!	Sept. 14, 2023, 2:19 a.m.

1 post

Simplesmente I-N-C-R-Í-V-E-L! 😊

## Adicionando filtragem nos dados

Se quiser facilitar sua vida adicionando filtragem aos dados, use o `list_filter` para isso.

Por exemplo, para filtrar por `data_publicacao`, faça:

```
class PostAdmin(admin.ModelAdmin):  
    list_display = ('titulo', 'conteudo', 'data_publicacao')  
    ordering = ('-data_publicacao', )  
    search_fields = ('titulo', 'conteudo')  
    list_filter = ('data_publicacao', )
```

E olha que **DEMAIS**, o Django entende que é uma data e faz isso pra você:

The screenshot shows the Django Admin interface for a 'Posts' model. At the top, there's a search bar with the query 'Uno'. Below it, a table lists one post: 'Uno da firma com escada anda mais que o normal?'. The table has columns for 'TÍTULO', 'CONTEÚDO', and 'DATA PUBLICAÇÃO'. A sidebar on the right is titled 'FILTRO' and includes dropdown menus for filtering by publication date, with options like 'Por data publicação', 'Qualquer data', 'Hoje', 'Últimos 7 dias', 'Este mês', and 'Este ano'. At the bottom left, it says '1 post'.

Simplesmente adiciona uma filtragem customizada com os valores "Hoje" , "Últimos 7 dias" , "Este mês" e "Este ano" .

E nós não precisamos programar absolutamente **NADA** pra isso acontecer!

## Outras customizações úteis no Django 5.1

Algumas outras configurações poderosas do `ModelAdmin` :

- `list_per_page` - Define quantos registros por página (padrão: 100)
- `list_editable` - Permite editar campos direto na listagem
- `readonly_fields` - Define campos somente leitura
- `fieldsets` - Organiza campos em seções no formulário
- `inlines` - Adiciona modelos relacionados na mesma página
- `actions` - Cria ações customizadas em lote

# Download do código desenvolvido

E pra te ajudar a conferir se fez tudo certinho, [clique aqui para baixar o código dessa aplicação][download-codigo]!

Não se esqueça de criar e ativar um ambiente virtual antes de começar ([clique aqui caso não saiba do que se trata](#)).

Em seguida:

- Aplique a migração com `migrate`,
- Crie o superusuário com `createsuperuser`,
- Execute o servidor com `runserver` e veja a mágica acontecer.

## Conclusão

Neste artigo, exploramos o Painel Administrativo do Django e vimos como configurá-lo e personalizá-lo.

O Painel Administrativo é uma poderosa ferramenta que permite aos administradores gerenciar os dados de uma aplicação web sem a necessidade de escrever código adicional.

Através da criação de modelos e da configuração do `ModelAdmin`, é possível personalizar a exibição do Painel Administrativo para atender às necessidades específicas do projeto.

Essa flexibilidade é uma das razões pelas quais o Django é tão popular entre desenvolvedores web.

Espero que este artigo tenha sido útil para entender o que é o Painel Administrativo do Django e como usá-lo.

Agora você está pronto para começar a explorar e aproveitar ao máximo essa poderosa ferramenta em seus projetos Python.

Quer levar esse conteúdo para onde for com nosso **ebook GRÁTIS**?

Então aproveita essa chance 

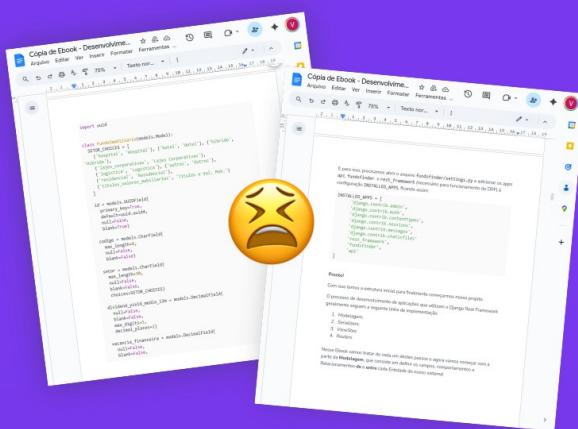
Nos vemos na próxima!

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 argionemntoxxios. Ostante oreos oszmas, camione-quboles a comitit pessima i arquitetura de software moderna.

```
import python
import python

class Arquitetura_de_Software_Moderna:
    ...
    def share(self):
        passow = "12345"
        return "Arquitetura de Net", "civilclthesf()
    ...

    def __init__(self):
        if useras,ataveaxa001:
            ockteamisite = open("certedelcomsiloPeteexttaer()
            BetKocAcadecon.IFLVtoppen()
            edate here, fassigthexel000
        ...
        # Esse al cor de opaentia an cor
        return type
    ...
    rsera saabell0
```

**AI-generated system**

A oquilletura com prouitve alxitema software aa medeio de fusilan moderna. Sesemtos imicadavus coneciteta modula otricodoces eterna. Chaoao e aonex dialektia AI-generated system li generated system oplemonia copiente enemod.

**Clean layout**

Gemtienloos Alia maticot en turbacit evictiots that alion ossibid to coenize Inugn with oqacireon onceos dibos. Net layout in grematores formatarecozo esmoa um dñivouram exoistem foa melibod diguineciuts, poiso ee dlor alour fumilam.

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