PYTHON ACADEMY

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

Nesse ebook você vai aprender a usar e customizar o incrível Painel Admin do Django!

PYTHONACADEMY.COM.BR

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`:

```python
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

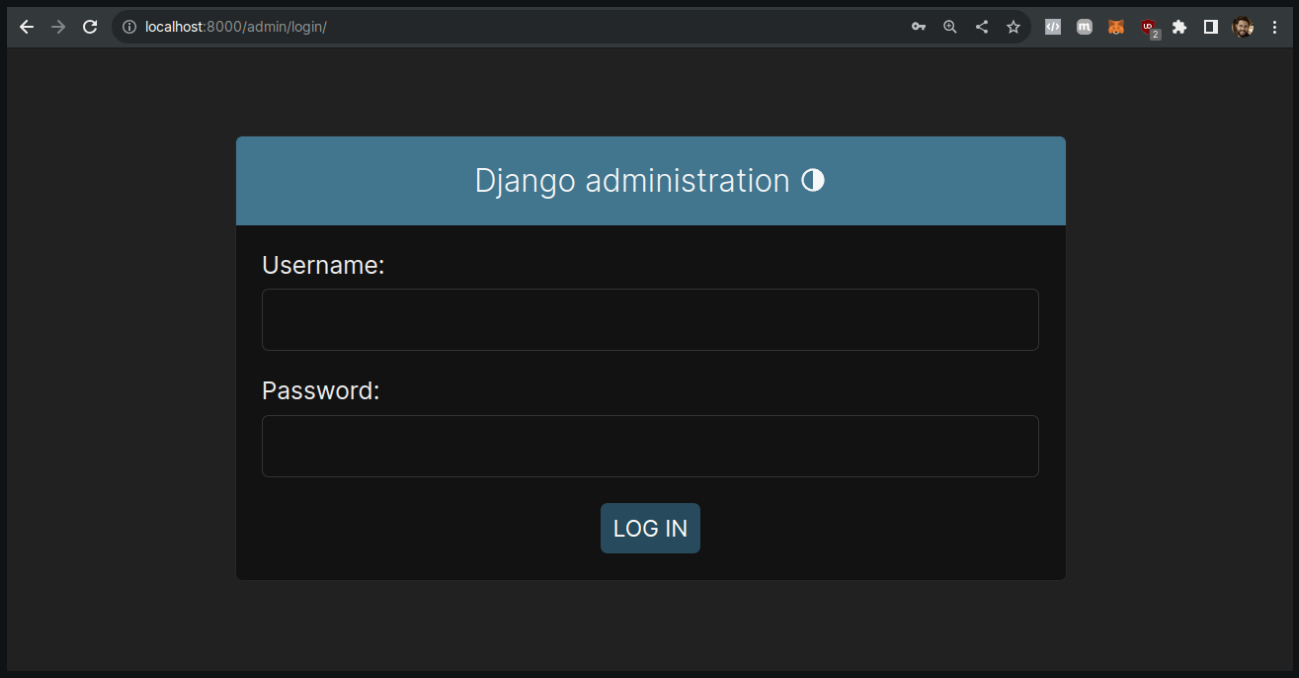
Agora, vamos iniciar o servidor de desenvolvimento do Django para testar nossa aplicação:

```
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:

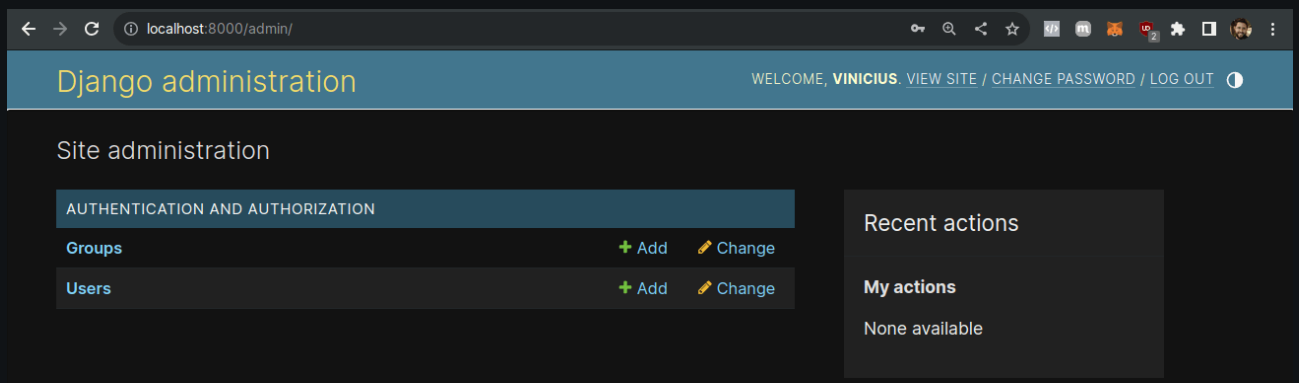


E então faça login usando as credenciais do superusuário que você acabou de criar.

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

Explorando o Painel Administrativo

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



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.

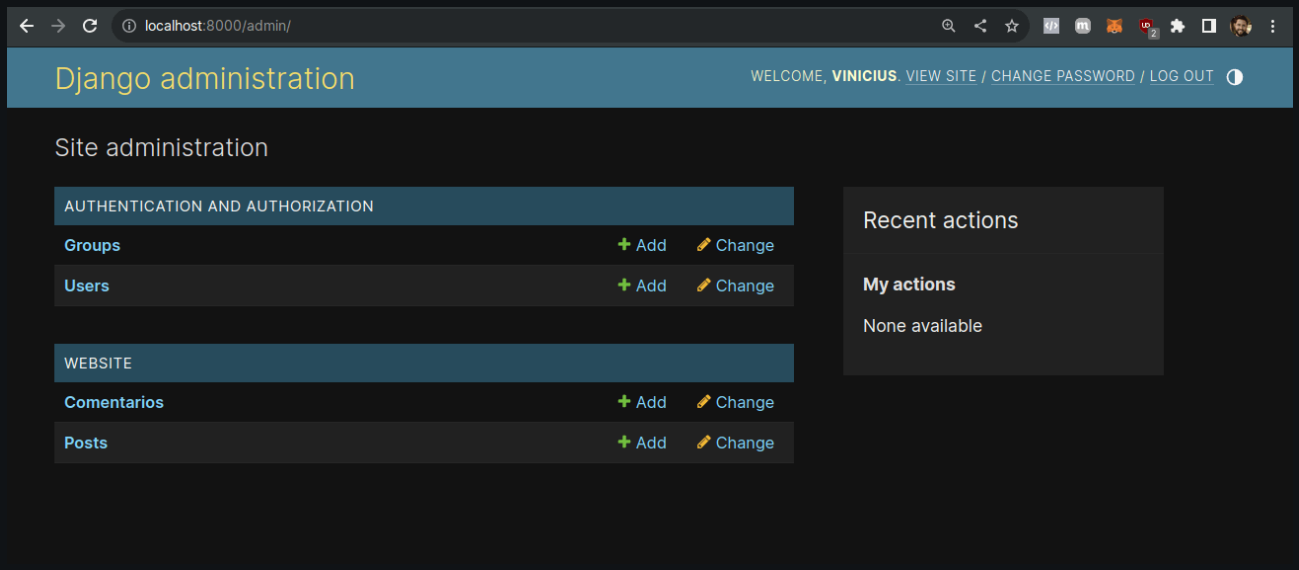
Vamos fazer isso dentro da pasta do App `website` que criamos com o comando `startapp`.

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***:



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* 😆

Mas calma que tem mais!

Existem várias maneiras de personalizar a exibição dos modelos.

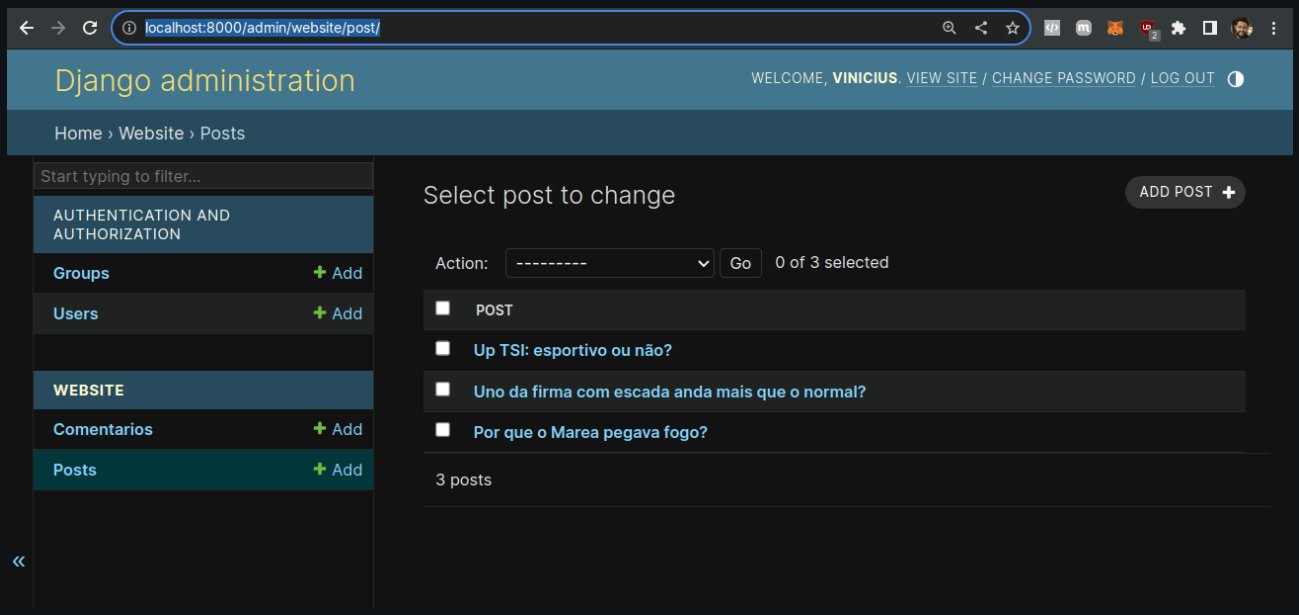
Vamos ver algumas 👇

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):



> *E aí? Fiat Uno anda mais ou não? Deixe seu comentário abaixo* 😆

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

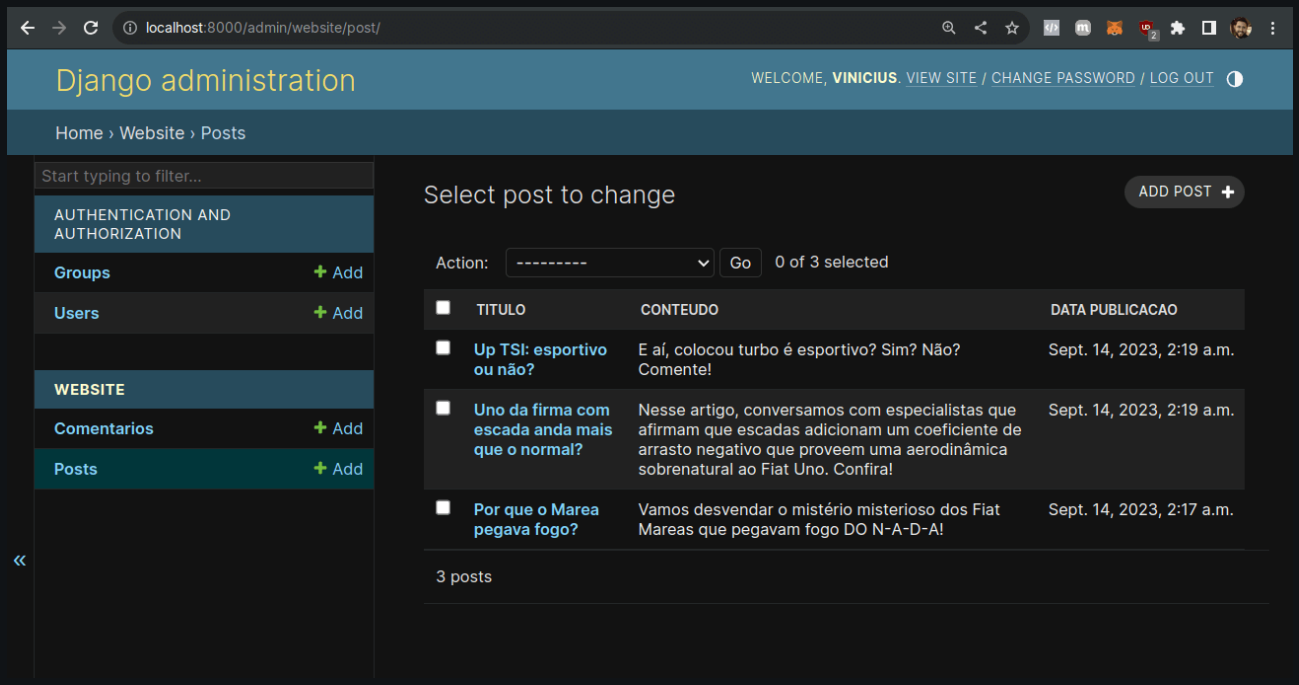
Dentro do arquivo `admin.py` da sua aplicação `website`, altere o `PostAdmin` para o seguinte código:

```
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:



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.

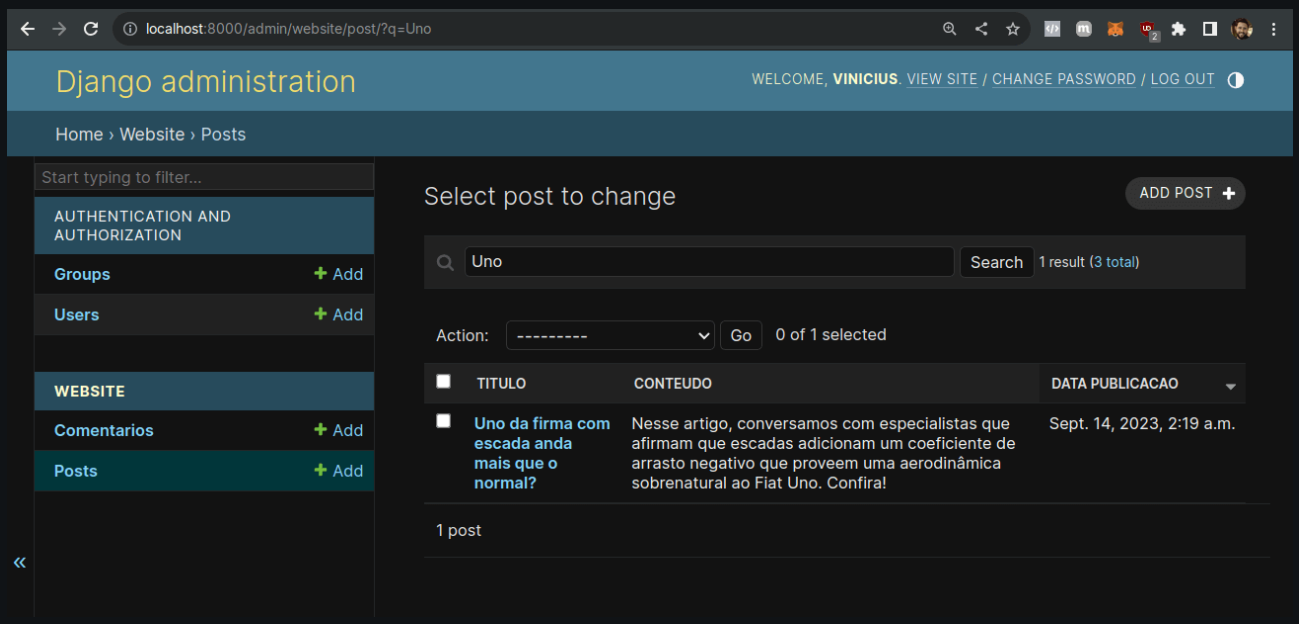
Importante salientar que o Django espera uma Tupla, por isso que usamos os colchetes com vírgula `('-data_publicacao', )` e não apenas `('-data_publicacao')`, o que gera um erro.

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`:



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:

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!

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 👇👇👇

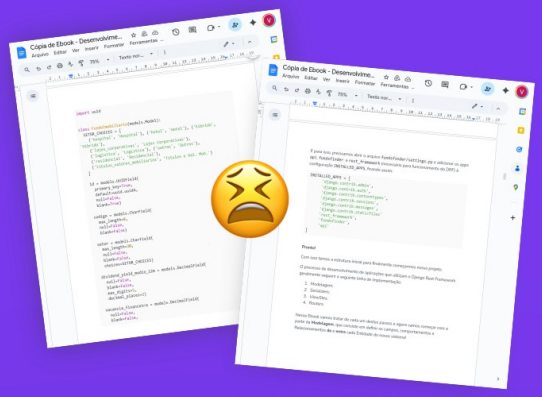
Nos vemos na próxima!

Não se esqueça de conferir!

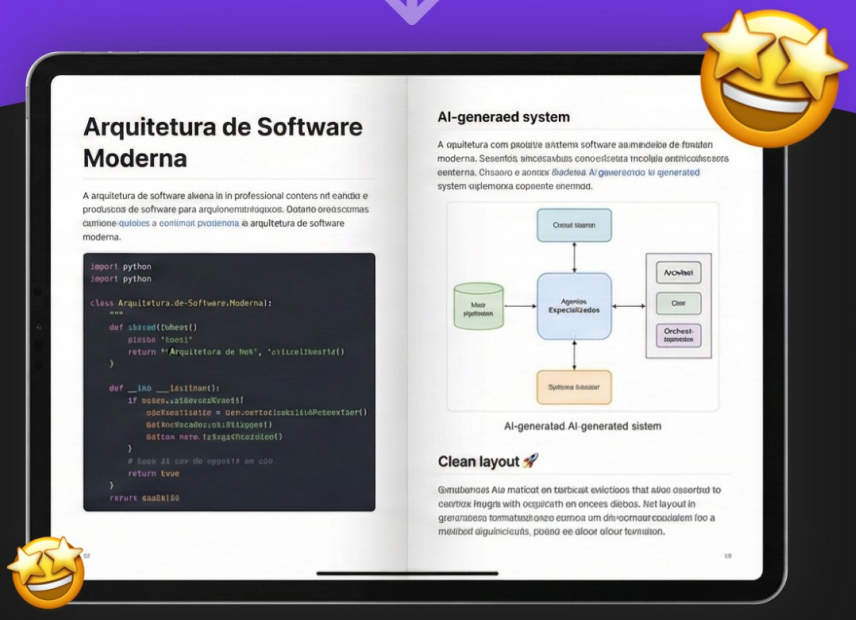
DevBook

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



Syntax Highlight

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