PYTHON ACADEMY

COMO CONECTAR O DJANGO (PYTHON) AO BANCO DE DADOS POSTGRESQL

Chega de usar o SQLite nas suas aplicações. Aprenda a conectar o poderoso SGBD PostgreSQL à sua aplicação Web Django!

PYTHONACADEMY.COM.BR

Salve salve Pythonista!

O Django é um framework de desenvolvimento web em Python que permite a criação de aplicações robustas e escaláveis.

Uma das principais tarefas ao construir uma aplicação Django é conectar o framework a um banco de dados para armazenar as informações da sua aplicação.

Neste artigo, vamos explorar como conectar o Django a um Banco de Dados Postgres, um sistema de gerenciamento de banco de dados relacional extremamente poderoso.

Mas primeiro... O que é o PostgreSQL?

PostgreSQL é um popular Sistema de Gerenciamento de Banco de Dados (SGBD) de código aberto, conhecido por sua confiabilidade, escalabilidade e recursos avançados.

Ele suporta o modelo de banco de dados relacional, que é a base de muitas aplicações modernas.

O Postgres é amplamente utilizado em projetos de todos os tamanhos, desde pequenas startups até grandes empresas.

Executando o Postgres localmente com Docker

Antes de conectar o Django ao Postgres, precisamos ter uma instância local do Postgres em execução.

Uma maneira fácil de fazer isso é usando o Docker.

O Docker permite que você execute aplicativos em contêineres isolados, facilitando a configuração e o gerenciamento de bancos de dados e outras dependências.

Para executar o Postgres localmente com Docker, você precisará ter o Docker instalado em sua máquina.

Uma vez instalado, você pode usar o seguinte comando para iniciar uma instância do Postgres:

```
docker run -d -p 5432:5432 --name postgres-db -e POST-
        GRES_PASSWORD=minha_senha -e POSTGRES_USER=meu_usuario post-
        gres
```

Altere `minha_senha` para uma senha de sua preferência e `meu_usuario` para um nome de usuário de sua preferência.

Isso iniciará um contêiner do Postgres na porta 5432 e definirá a senha e o usuário especificados nas variáveis de ambiente.

Configurando o Django para se conectar ao Postgres

Agora que temos uma instância do Postgres em execução, podemos configurar nosso projeto Django para se conectar a ela.

Abra o arquivo `settings.py` do seu projeto Django e localize o dicionário de configurações `DATABASES`.

Substitua o valor `default` pela seguinte configuração:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'nome_do_banco',
        'USER': 'meu_usuario',
        'PASSWORD': 'minha_senha',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

Certifique-se de substituir `nome_do_banco`, `meu_usuario` e `minha_senha` pelos valores corretos para o seu banco de dados.

> 💡 *Estou construindo o **DevBook**, uma plataforma que usa IA para criar ebooks técnicos — com código formatado e exportação em PDF. Não deixe de conferir!*

Além disso, é necessário instalar uma outra dependência para que o Django saiba conversar com o Banco de Dados.

Essa dependência é a `psycopg2`.

Contudo, instalar essa dependência não é algo trivial pois exige a compilação de diversas bibliotecas internas.

Pode ter certeza que o simples `pip install psycopg2` não irá funcionar.

Para isso, existe uma alternativa muito útil que é a biblioteca `psycopg2-binary` que já vem com essas diversas bibliotecas internas já compiladas, o que facilita muito nossa vida.

Dessa forma, e com seu ambiente virtual ativado (saiba mais sobre ambientes virtuais e Virtualenv clicando aqui), instale-a com `pip install psycopg2-binary`.

Contudo, atente-se pois seu uso **não é aconselhado em ambiente de Produção**.

Portanto, use-o apenas em ambiente local e de desenvolvimento. Quando for instalar sua aplicação em Produção, compile as bibliotecas necessárias e utilize o `psycopg2`.

Definindo os Modelos da nossa aplicação

Agora que o Django está configurado para se conectar ao Postgres, podemos criar os modelos de dados que serão usados em nossa aplicação.

Vamos criar um exemplo simples de uma aplicação de Blog.

Crie um novo arquivo chamado `models.py` dentro do diretório de sua aplicação e adicione o seguinte código:

```
from django.db import models

class Post(models.Model):
    title = models.CharField(max_length=250)
    content = models.TextField()
    pub_date = models.DateTimeField(auto_now_add=True)
```

Neste exemplo, criamos um modelo chamado `Post` com três campos: `title`, `content` e `pub_date`.

O campo `pub_date` é configurado para ser preenchido automaticamente com a data e hora atuais sempre que um novo `Post` for criado.

Executando as Migrações no Banco de Dados

Antes de podermos usar nosso modelo `Post`, precisamos aplicar as migrações ao banco de dados.

As migrações são um recurso do Django que permite controlar e gerenciar as alterações no esquema do banco de dados ao longo do tempo.

Para executar as migrações, use o seguinte comando:

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

Isso criará as tabelas necessárias no banco de dados e deixará tudo pronto para começarmos a usar os modelos em nossa aplicação.

Conclusão

Conectar o Django a um Banco de Dados Postgres é uma tarefa essencial ao desenvolver aplicações web com o framework.

Neste artigo, exploramos como executar o Postgres localmente com Docker e como configurar o Django para se conectar a ele.

Também foi mostrado como criar e usar modelos de dados com o Django.

Agora, você está pronto para construir aplicações robustas e escaláveis utilizando o Django e o poderoso banco de dados Postgres.

Quer levar esse conteúdo para onde for com nosso **ebook GRÁTIS de Desenvolvimento Web com Python e Django**?

Então aproveita essa chance 👇 👇 👇

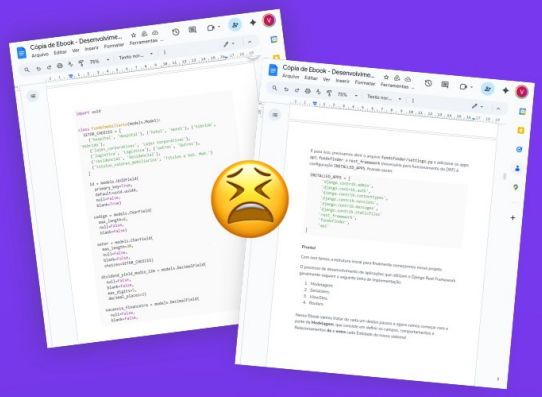
Nos vemos na próxima! 👋

Não se esqueça de conferir!

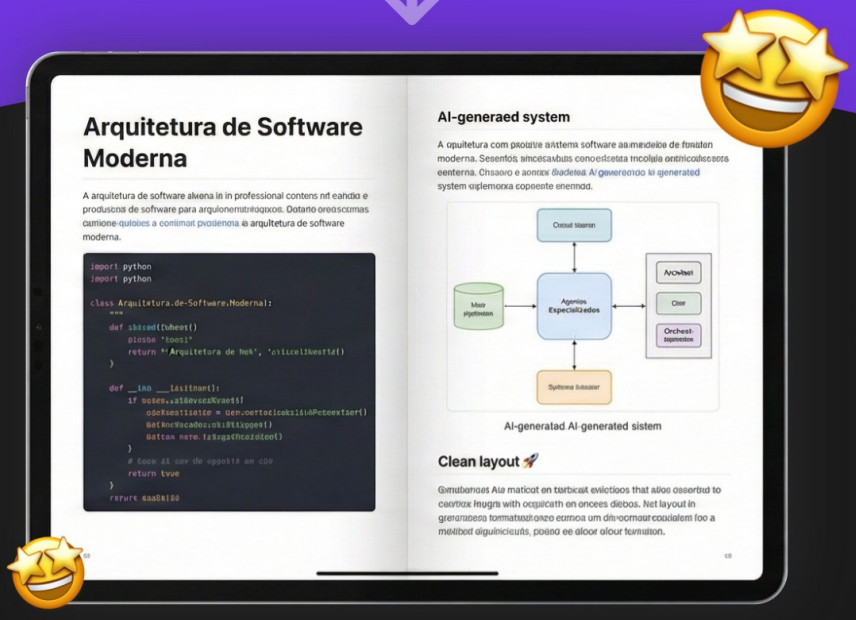
DevBook

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