# Django

Prof. José Matheus

# Outros bancos de dados

# Outros bancos de dados

O Django suporta vários bancos de dados, incluindo:

**SQLite**: Um banco de dados leve embutido no Django e adequado para pequenas aplicações ou para desenvolvimento e testes.

**PostgreSQL**: Um banco de dados relacional de código aberto e altamente escalável.

**MySQL**: Um banco de dados relacional amplamente utilizado, conhecido por sua facilidade de uso e desempenho.

**Oracle**: Um banco de dados comercial popular com recursos avançados.

**Microsoft SQL Server**: Um banco de dados relacional desenvolvido pela Microsoft, comumente usado em ambientes corporativos.

# Como configurar o Django para utilizar o PostgreSQL como banco de dados

**Passo 1: Instalar as dependências**

Antes de mais nada deve-se ter o PostgreSQL instalado em seu sistema. Você também precisará instalar o pacote psycopg2, que é o adaptador de banco de dados PostgreSQL para o Django. Você pode instalá-lo usando o pip:

```
pip install psycopg2
```

# Configurando o banco de dados Postgres

**Passo 2: Configurar as informações do banco de dados**

Abra o arquivo settings.py do seu projeto Django e localize a seção DATABASES. Substitua as configurações existentes pelo seguinte código ao lado.

**Passo 3: Executar as migrações**

Após configurar o banco de dados, execute as migrações para criar as tabelas necessárias no PostgreSQL.

python manage.py migrate

PYTHON

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'nome_do_banco_de_dados',
        'USER': 'seu_usuario',
        'PASSWORD': 'sua_senha',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

# Arquivos de mídia

# Arquivos de mídia

**Passo 1: Configuração**

No arquivo **settings.py** do seu projeto Django, você precisa configurar as definições relacionadas aos arquivos de mídia. Defina as configurações **MEDIA_ROOT** e **MEDIA_URL**.

**MEDIA_ROOT**: Especifica o caminho absoluto para o diretório onde os arquivos de mídia serão armazenados [lembre-se de criar uma pasta na raiz do projeto com o mesmo nome]. Por exemplo:

```
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

**MEDIA_URL**: Define o URL base para os arquivos de mídia. Por exemplo:

```
MEDIA_URL = '/media/'
```

# Arquivos de mídia

**Passo 1: Configuração**

Também será necessário instalar a seguinte dependência:

```
pip install Pillow
```

A Python Imaging Library (PIL) adiciona recursos de processamento de imagem ao seu interpretador Python. Esta biblioteca oferece amplo suporte a formatos de arquivo, uma representação interna eficiente e recursos de processamento de imagem bastante poderosos.

# Arquivos de mídia

**Passo 2: Configuração de URL**

Adicionar a seguinte configuração ao arquivo meu_projeto/urls.py

PYTHON

```python
from django.conf import settings
from django.conf.urls.static import static


urlpatterns = [
    # ... suas outras URLs ...
] + static(settings.MEDIA_URL , document_root=settings.MEDIA_ROOT)
```

# Arquivos de mídia

**Passo 3: Upload de arquivos de mídia**

Em seu modelo Django, você precisa definir um campo de arquivo para lidar com o upload de arquivos de mídia. Use a classe FileField ou ImageField do módulo django.db.models.fields

PYTHON

```python
class Produto(models.Model):

    # ... seus outros campos ...

    imagem = models.ImageField( upload_to='produtos/', null=True, blank=True)
```

# Arquivos de mídia

**Passo 4: Manipulação de arquivos de mídia**

Ao receber um arquivo de mídia no seu aplicativo, o Django cuidará automaticamente do processo de armazenamento. O arquivo será salvo no diretório definido em **MEDIA_ROOT** com um nome único.

# Dúvidas?

# Referências

- **Documentação do Django:** https://docs.djangoproject.com/
- **Pillow:** https://pypi.org/project/Pillow/