PYTHON ACADEMY

COMANDO MIGRATE DO DJANGO (PYTHON)

Nesse ebook, vamos ver o funcionamento do comando migrate do Django, suas opções e como ele ajuda os desenvolvedores na hora de criar o banco de dados da nossa aplicação Django.

PYTHONACADEMY.COM.BR

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) O comando* `migrate` *permanece essencial para aplicar migrations ao banco de dados.*

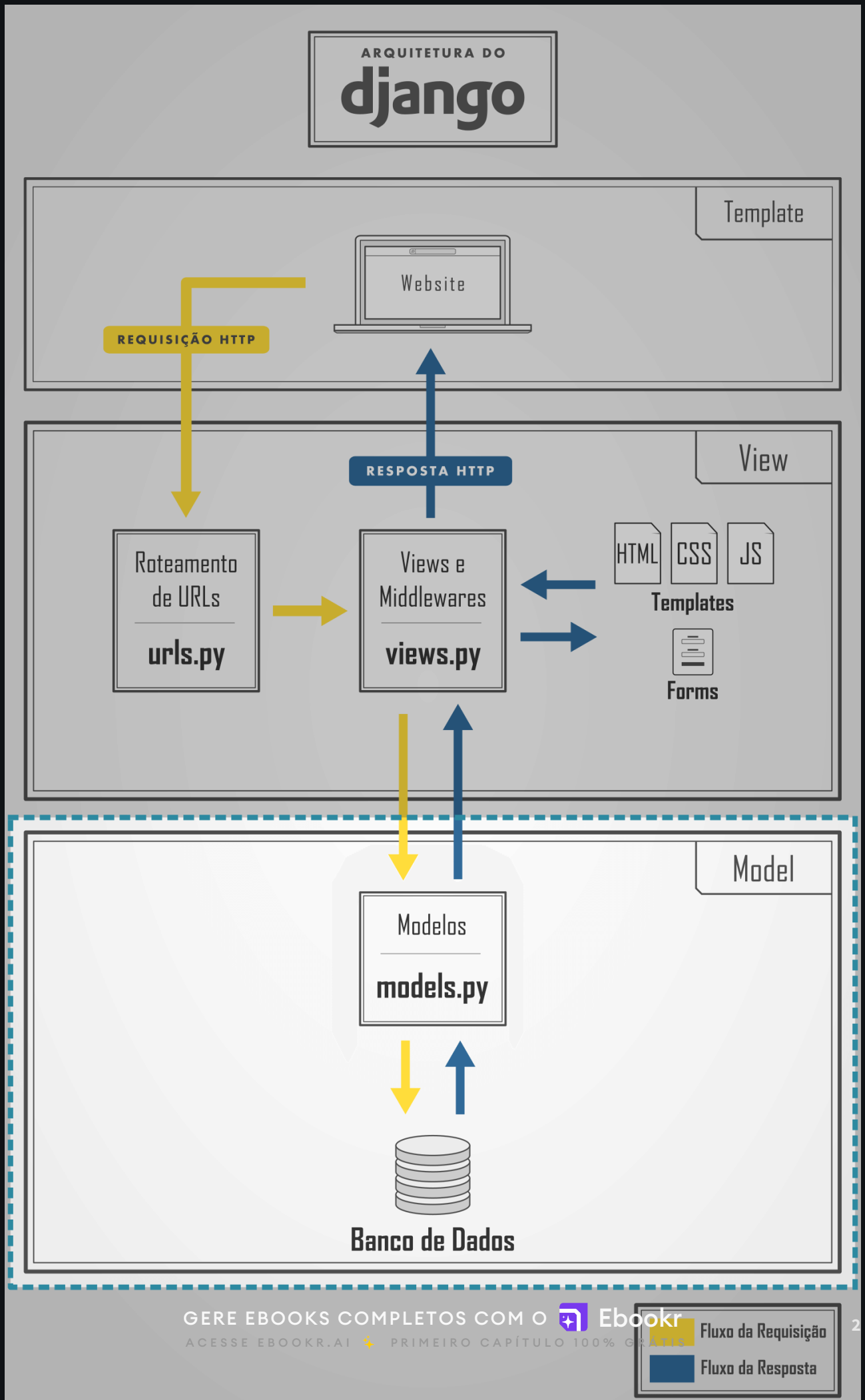
Salve salve Pythonista! 👋

Nesse post, vamos ver o que o comando `migrate` faz, as opções do comando e como ele ajuda a vida do desenvolvedor Django na hora de criar a estrutura do banco de dados da nossa aplicação!

Bora nessa!

Onde estamos...

Como usual, vamos nos situar nesse mundão Django!



No primeiro post da nossa Série de Django, tratamos de conceitos introdutórios do *framework*, uma visão geral da sua arquitetura, sua instalacão e a criação do ***Hello World, Django***.

Já no segundo post da Série, tratamos da camada *Model*, onde definimos as entidades do nosso sistema, a interface com o banco de dados e aprendemos como utilizar a API de acesso a dados provida pelo Django - que facilita muito nossa vida!

Agora, vamos ver o funcionamento de um dos comandos mais importantes da camada *Model* do Django: o `migrate`.

O comando `migrate`

Quando executamos o comando `makemigrations`, o Django cria o banco de dados e as *migrations*, mas não as executa, isto é: não **realmente** aplica as alterações no banco de dados.

Para que o Django as aplique, são necessárias três coisas, basicamente:

- **1.** Que a configuração da interface com o banco de dados esteja descrita no arquivo de configurações `settings.py`.
- **2.** Que os modelos e *migrations* estejam definidos para esse projeto.
- **3.** Execução do comando `migrate`.

Se você acompanhou os últimos posts, você criou o projeto *Hello World, Django* através do comando `django-admin.py createproject helloworld` e, portanto, a configuração padrão foi aplicada.

Procure pela configuração `DATABASES` no `settings.py`. Ela deve ser a seguinte:

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

Por padrão, o Django utiliza um banco de dados leve e compacto chamado [SQLite][sqlite]. Já já vamos falar mais sobre ele.

Sobre os modelos e *migrations*, eles já foram feitos com a definição dos modelos no arquivo `models.py` e com a execução do comando `makemigrations`.

Agora só falta **executar o comando `migrate`**, propriamente dito!

Para isso, vamos para a raíz do projeto e executamos: `python manage.py migrate`. A saída deve ser:

```
$ python manage.py migrate

Operations to perform:
  Apply all migrations: admin, auth, contenttypes, helloworld, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying helloworld.0001_initial... OK
  Applying sessions.0001_initial... OK
```

*Calma lá... O comando `makemigrations` que executamos havia definido apenas **uma** Migration e foram aplicadas 15!!! Por quê???*

Se lembra que a configuração `INSTALLED_APPS` contém vários `apps` (*e não apenas os nossos*)?

Pois então, cada `app` desses contém seus próprios modelos e *migrations*. *Sacou?!*

Com a execução do `migrate`, o Django irá criar as diversas tabelas desse outros `apps` no banco de dados **E** as tabelas definidas nos nosso arquivos de modelo. Uma delas é a nossa, similar à:

```
CREATE TABLE helloworld_funcionario (
    "id" serial NOT NULL PRIMARY KEY,
    "nome" varchar(255) NOT NULL,
    "sobrenome" varchar(255) NOT NULL,
    ...
);
```

Isso está muito abstrato! Como eu posso ver o banco, as tabelas e os dados na prática?

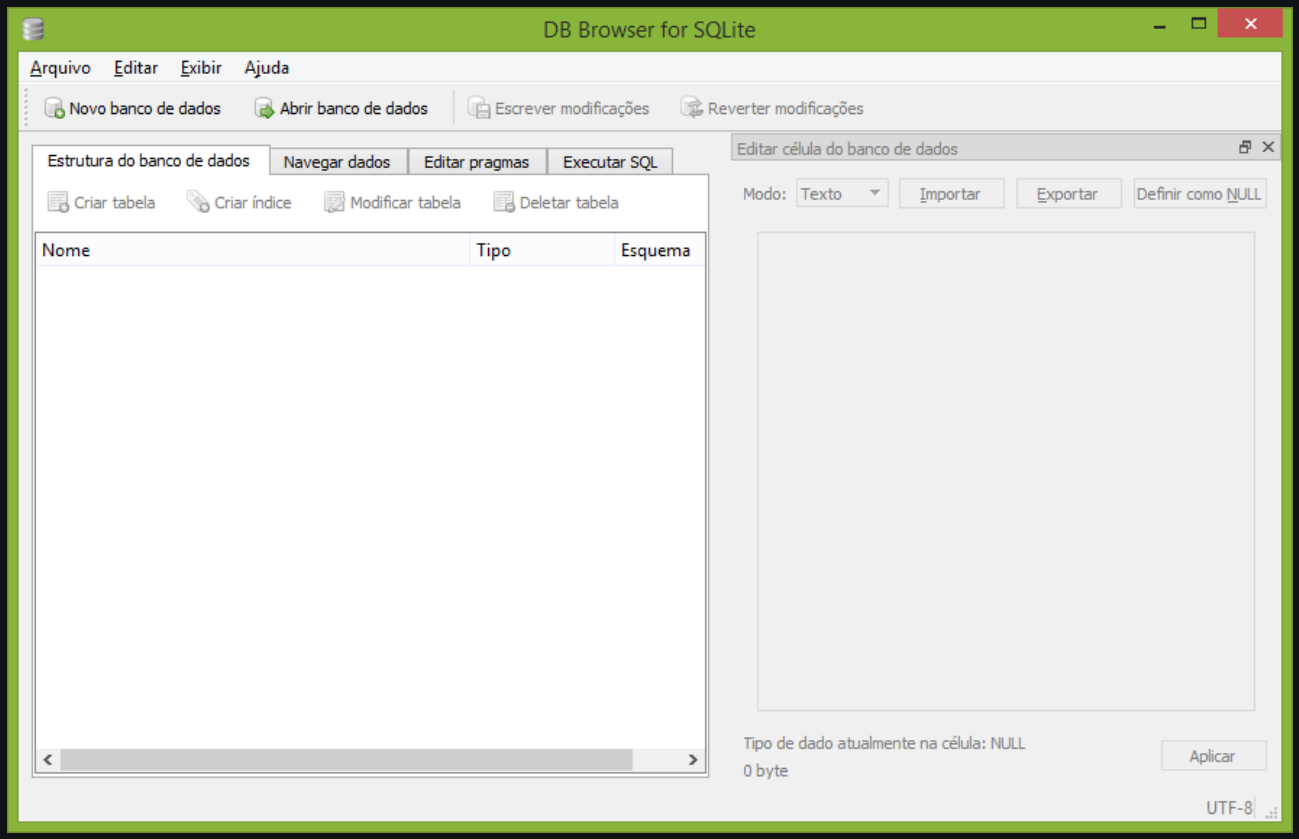
DB Browser for SQLite

Apresento-lhes uma ferramenta **MUITO** prática: o *DB Browser for SQLite*!

Com ela, podemos ver a estrutura do banco de dados, alterar dados em tempo real, fazer *queries*, e muito mais!

[Clique aqui para fazer download do DB Browser for SQLite][db-browser-sqlite] e instalá-lo. Ao terminar a instalação, abra o *DB Browser for SQLite*.

Temos:



Nele, podemos clicar em "**Abrir banco de dados**" e procurar pelo banco de dados do nosso projeto `db.sqlite3`.

Ao importá-lo, teremos uma visão geral, mostrando Tabelas, Índices, *Views* e *Triggers*.

Para ver os dados de cada tabela, vá para a aba “**Navegar dados**”, escolha nossa tabela `helloworld_funcionario` e...

Voilá! O que temos? **NADA** 😞

Calma jovem... Ainda não adicionamos nada 😒

*Mas, se você está ansioso para continuar nossa jornada Django, você pode ir direto para o próximo post, onde tratamos da Camada **View** do Django!* 😉

> 💡 *Estou construindo o **Ebookr.ai**, uma plataforma onde você cria ebooks profissionais com IA sobre qualquer assunto — do zero ao PDF pronto, com capas e infográficos gerados automaticamente. Dá uma olhada!*

Opções do comando `migrate`

O formato padrão do comando `migrate` é o seguinte:

```
django-admin migrate [label_app] [nome_migration] [opções]
```

Abaixo vamos ver as opções do comando.

- Sem argumentos: aplica todas as migrações disponíveis de todos os *apps* definidos em `INSTALLED_APPS`.
- `<label_app>`: Aplica-se todas as migrações do app especificado até a mais recente. Pode envolver a aplicação de outras migrações, pois pode haver dependência entre migrações.
- `<label_app> <nome_migrations>`: Leva a estrutuda do banco de dados à um estado em que a migração é aplicada, sem aplicar as migrações seguintes.
- `--database DATABASE`: Especifica o banco de dados para executar as migrações. As opções disponíves são àquelas que foram definidas pela configuração `DATABASES` no `settings.py`.
- `--fake`: Opção avançada que permite marcar migrações como **aplicadas**, sem realmente aplicá-las (ou seja, sem modificar a estrutuda do seu banco de dados). Opção geralmente utilizada para recuperação da base de dados, ou quando se quer alterar manualmente a tabela interna de estados das migrações do Django.
- `--fake-initial`: Permite ao Django pular a migração inicial, caso todas as tabelas já tenham sido previamente criadas. Essa opção é usada quando a estrutura inicial de tabelas já foi criada previamente no banco de dados (por exemplo, sistema legado ou se um DBA já executou um script de criação das tabelas).

- `--plan` : Apenas mostra as migrações que serão aplicadas com a execução do comando `migrate` .
- `--run-syncdb` : Permite a criação da estrutura de tabelas sem as migrações. Essa opção não verifica a estrutura das tabelas nem verifica a questão das dependências entre elas. Essa opção é usada com sistemas muito grandes, com centenas de modelos, onde o framework de migrações do Django pode ser lento.
- `--noinput` ou `--no-input` : Evita confirmações de usuário. Opção útil quando essa opção for utilizada em scripts bash.

Conclusão

Vimos nesse post como utilizar o comando `migrate` , suas opções e como ele facilita nossa vida.

Já imaginou ter que montar toda a estrutura de banco de dados na unha - com SQL e tudo mais? 😱😱😱

Tá curtindo o conteúdo? Que tal levar esse conteúdo para onde for com nosso **ebook GRÁTIS sobre Desenvolvimento Web com Python e Django**?

Então aproveita essa chance 👇👇👇

Atá a próxima!

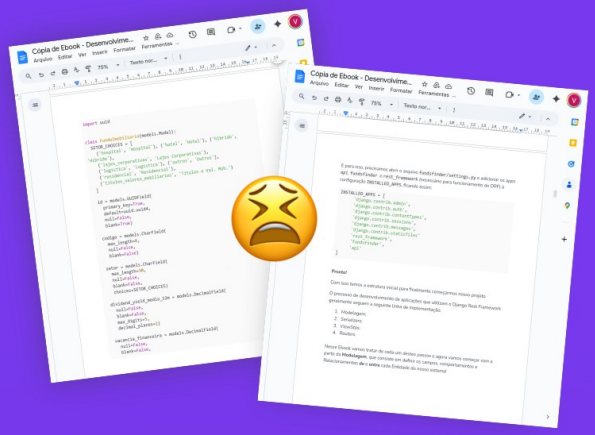
Bom desenvolvimento! 😉

[django-model-reference]//docs.djangoproject.com/en/5.1/ref/models/fields/ [sqlite]: https://sqlite.org/index.html [view-post]: https://pythonacademy.com.br/blog/desenvolvimento-web-com-python-e-django-view [db-browser-sqlite]: http://sqlitebrowser.org/

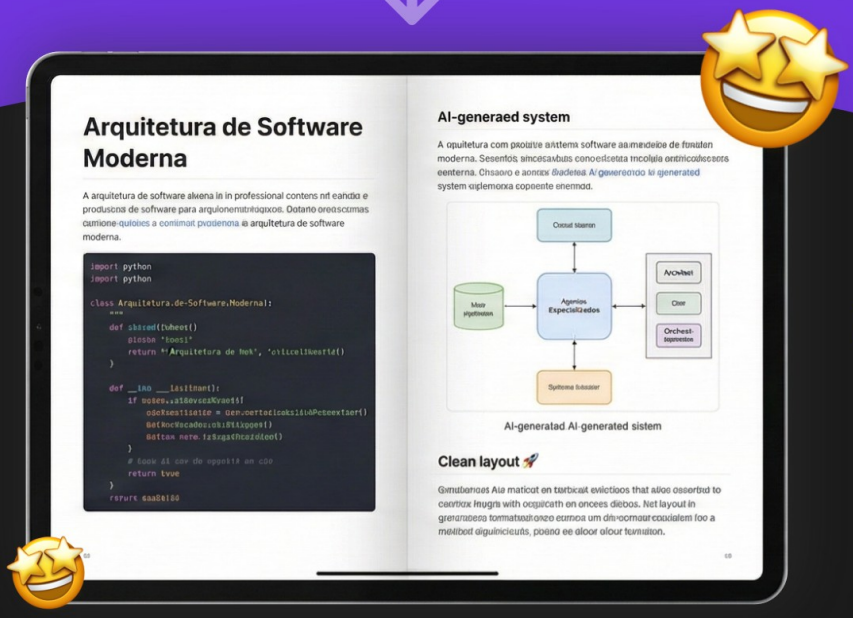
Não se esqueça de conferir!

Ebookr

Crie Ebooks profissionais em minutos com IA



Chega de formatar código no Google Docs ou Word



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