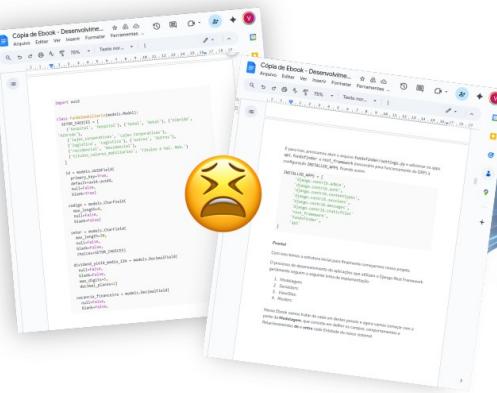


COMANDO MAKEMIGRATIONS DO DJANGO (PYTHON)

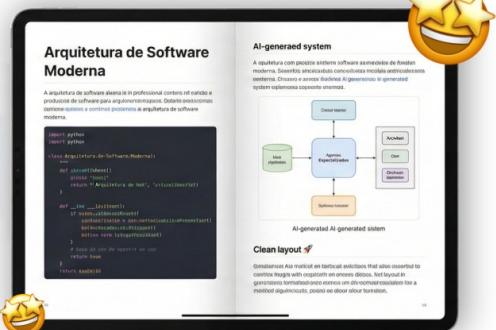
Nesse ebook, vamos ver o funcionamento do comando makemigrations do Django, suas opções e quando devemos usá-lo.

Crie Ebooks técnicos incríveis em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código!**



Chega de formatar código no Google Docs



Deixe que nossa IA faça o trabalho pesado

 Syntax Highlight

 Adicione Banners Promocionais

 Edite em Markdown em Tempo Real

 Infográficos feitos por IA

TESTE AGORA 

Salve salve Pythonista! 

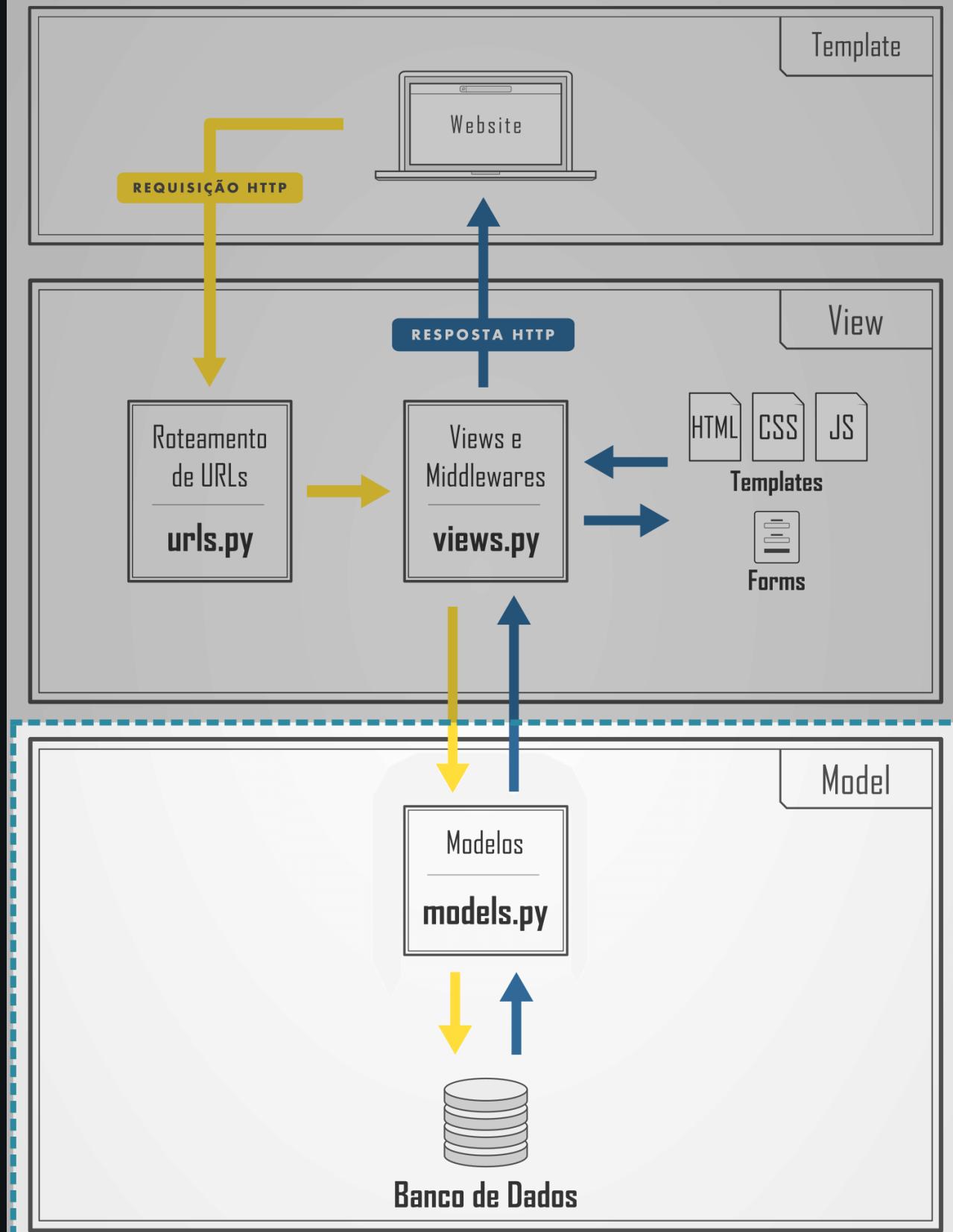
Nesse post, vamos ver o que o comando `makemigrations` faz, as opções do comando e o porquê de ele ser tão importante na vida do desenvolvedor Django!

Ajusta sua cadeira e vamos nessa!

Onde estamos...

Primeiramente, vamos nos situar...

ARQUITETURA DO django



No [primeiro post da nossa Série de Django](#), tratamos de conceitos introdutórios do *framework*, uma visão geral da sua arquitetura, sua instalação 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 `makemigrations`.

O comando `makemigrations`

O comando `makemigrations` analisa se foram feitas mudanças nos modelos e, em caso positivo, cria novas migrações (`Migrations`) para alterar a estrutura do seu banco de dados, refletindo as alterações feitas.

Vamos entender o que eu acabei de dizer: toda vez que você faz uma **alteração** em seu modelo, é necessário que ela seja **aplicada** à estrutura presente no banco de dados.

A esse processo é dado o nome de **Migração**!

De acordo com a documentação do Django:

*Migração é a forma do Django de **propagar as alterações** feitas em seu modelo (adição de um novo campo, deleção de um modelo, etc...) ao seu esquema do banco de dados. Elas foram desenvolvidas para serem (a maioria das vezes) **automáticas**, mas cabe a você saber a hora de fazê-las, de executá-las e de resolver os problemas comuns que você possa vir a ser submetidos.*

Portanto, toda vez que você alterar o seu modelo (`models.py`), não se esqueça: **execute** `python manage.py makemigrations` !

Ao executar esse comando no projeto que iniciamos no [primeiro post sobre Django](#), devemos ter a seguinte saída:

```
$ python manage.py makemigrations

Migrations for 'helloworld':
  helloworld\migrations\0001_initial.py
    - Create model Funcionario
```

Observação: Talvez seja necessário executar o comando referenciando o app onde estão definidos os modelos: `python manage.py makemigrations helloworld`. Daí pra frente, apenas `python manage.py makemigrations` vai bastar!

Agora, podemos ver que foi criada uma pasta chamada `migrations` dentro de `helloworld`.

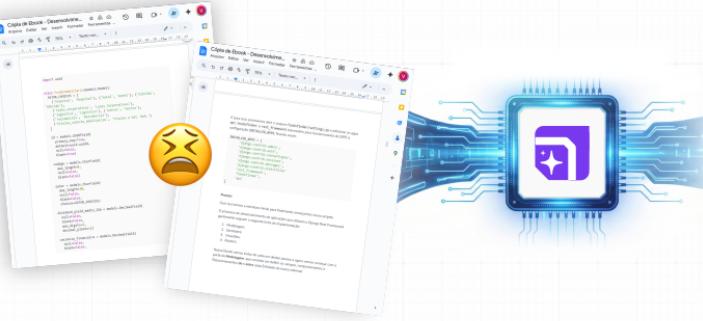
Dentro dela, você pode ver um arquivo chamado `0001_initial.py`: ele contém a `Migration` que cria o `model Funcionario` no banco de dados (*preste atenção na saída do comando `makemigrations`: `Create model Funcionario` 😊*)

💡 Estou construindo o **DevBook**, uma plataforma que usa IA para gerar ebooks técnicos profissionais. Te convido a conhecer!

 DevBook

Crie Ebooks técnicos incríveis em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código!**



Chega de formatar código no Google Docs



Deixe que nossa IA faça o trabalho pesado

 Syntax Highlight  Adicione Banners Promocionais  Edite em Markdown em Tempo Real  Infográficos feitos por IA

TESTE AGORA! PRIMEIRO CAPÍTULO 100% GRÁTIS 

Opções do comando makemigrations

Abaixo vamos ver as opções do comando.

- `--noinput` ou `--no-input` : Evita prompts ao usuário. Muito útil quando esse comando for usado em scripts bash.

- `--empty` : Produz uma migração vazia para os *apps* especificados, para edição manual. Isso é para usuários avançados e não deve ser usado, a menos que você esteja familiarizado com o formato da migração, as operações de migração e as dependências entre suas migrações.
- `--dry-run` : Mostra quais migrações seriam feitas sem realmente gravar nenhum arquivo de migração no disco, ou seja, sem aplicar no banco de dados. O uso dessa opção junto com `-verbosity 3` também mostrará os arquivos de migração completos que seriam gravados.
- `--merge` : Permite corrigir conflitos de migrações.
- `--name NAME, -n NAME` : Permite nomear as migrações geradas em vez de usar um nome gerado pelo Django. O nome deve ser um identificador Python válido.
- `--no-header` : Gera arquivos de migração sem a versão do Django e o cabeçalho de data e hora (exige Django ≥ 2.2).
- `--check` : Faz com que o comando `makemigrations` termine com um status diferente de zero quando alterações de modelo sem migrações são detectadas.

Conclusão

Vimos nesse post como utilizar o comando `makemigrations`, suas opções e como ele facilita nossa vida (já imaginou fazer as migrações e resolvê-las na unha? 😱 😱 😱).

É o Django facilitando nossa vida cada vez mais!

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

Então aproveita essa chance 

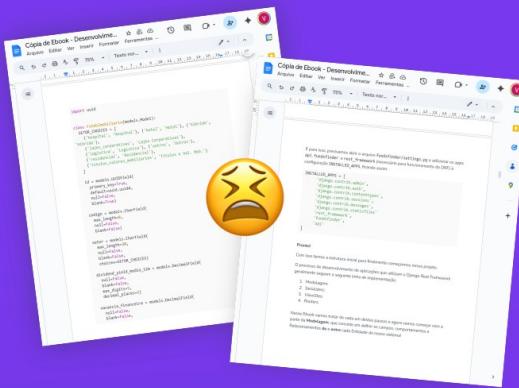
Nos vemos por aí!

Bom desenvolvimento! 



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

Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado

Infográficos feitos por IA

Edite em Markdown em Tempo Real

TESTE AGORA

PRIMEIRO CAPÍTULO 100% GRÁTIS