Banco de Dados

Tipos de Models (Django)

# O que são models?

Models é a parte Django (uma “biblioteca de predefinições”) responsável pela organização e comportamentos de dados; são utilizados em Python para desenvolvimento web.

Existe alguns meios de comparação para a melhor assimilação, por exemplo, é possível comparar Django com receitas de cozinha. Cada receita tem instruções específicas para preparação de um prato.

Por exemplo, se você está seguindo uma receita para fazer um bolo, você mistura os ingredientes na ordem e assa no forno, mas se a receita for uma pizza, você prepara a massa, adiciona molho e recheio e coloca para assar.

Ou seja, os models no Django são como receitas, cada model tem suas próprias instruções específicas para lidar com vários tipos de dados. No caso, cada tipo de dado vai ser melhor tratado com um model correspondente.

# Model CharField

Este campo lida com o tipo de dado “texto” (**string**); é usado para textos curtos, como nomes, títulos e endereços, por exemplo.

Possui parâmetro obrigatório: ***max\_length*** - define o comprimento máximo da string (limitado ao valor máximo de 254 caracteres).

Código:

nome = models.CharField(max\_length=100)

# Model TextField

Este campo também lida com o tipo de dado “texto”, mas nesse caso sendo textos longos, como descrições, comentários, biografias e etc.

Não possui parâmetro obrigatório.

Código:

sobre\_voce = models.TextField()

# Model EmailField

É usado para armazenar endereços de e-mail. Ele verifica automaticamente se o valor inserido é um e-mail válido.

Não possui parâmetro obrigatório.

Código:

email = models.EmailField()

# Model IntegerField

Lida com números inteiros, pode ser utilizado para representar Ids, quantidades, código postal e entre outros.

Não possui parâmetro obrigatório.

Código:

idade = models.IntegerField()

# Model PositiveIntegerField

Lida com números inteiros, mas apenas os positivos, não aceitando números negativos e zero. Útil para representar dados que devem ser apenas positivos como Idade ou quantidade, por exemplo.

Não possui parâmetro obrigatório.

Código:

idade = models.PositiveIntegerField()

# Model FloatField

Lida com números “flutuantes”, números não inteiros/decimais. Usado para representar valores monetários, medidas como altura e etc.

Não possui parâmetro obrigatório.

Código:

altura = models.FloatField()

# Model BooleanField

Armazena dados booleanos valores que representam apenas duas opções: verdadeiro/falso, 0/1, “sim”/”não”, etc.

Possui parâmetro opcional: **default -** defini o valor padrão.

Código:

possui\_filhos = models.BooleanField(default=False)

# Model DateField

Usado para armazenar somente datas, representa datas de nascimento, datas de evento, entre outros.

Não possui parâmetro obrigatório.

Código:

data\_nascimento = models.DateField()

# Model DateField

Armazena datas e horas. Útil para representar datas de criação/alteração de registros, etc.

Não possui parâmetro obrigatório.

Código:

data\_de\_criacao = models.DateTimeField()

# Model ForeignKey

Campo usado para estabelecer uma relação entre dois modelos (de muitos-para-um), sendo um deles dependente do outro. Exemplos: relacionamentos de um Autor de um livro, categoria de um produto, etc.

Possui parâmetros obrigatórios:

1: ***To*** - aponta com qual modelo a chave está relacionada.

2: ***on\_delete*** - especifica o que acontece com os objetos relacionados quando o objeto referenciado é excluído.

Código:

autor = models.ForeignKey(Pessoa, on\_delete=models.CASCADE)

# Model ManyToManyField

Estabelece uma relação entre dois modelos (de muitos-para-muitos), sendo ambos modelos independentes apesar de relacionados. Útil para representar quando um objeto pode estar associado a vários outros objetos e vice-versa. Por exemplo, **Músicas e Playlists:** Uma música pode estar em várias playlists e uma playlist pode conter várias músicas.

Possui parâmetro obrigatório: ***To*** - indica o modelo ao qual o campo many-to-many está associado.

Código:

livros\_produzidos = models.ManyToManyField(Livro)

# Model OneToOneField

Estabelece uma relação entre dois modelos (de um-para-um). Cada objeto em um modelo está associado a no máximo um objeto no outro modelo. Exemplo: “Pessoa Física” e CPF.

Possui parâmetros obrigatórios:

1: ***To* -** aponta para qual modelo o campo OneToOneField se refere.

2: ***on*** - delete, especifica o que acontece com os objetos relacionados quando o objeto referenciado é excluído.

Código:

aluno = models.OneToOneField(Aluno, on\_delete=models.CASCADE)

# Model ImageField

Lida somente com dados do tipo “imagem”, armazena a imagem no sistema de arquivos do servidor e armazena o caminho para essa imagem no banco de dados. Serve para fazer upload e guardar essas imagens em um banco de dados.

Possui parâmetro opcional: ***upload\_to*** - cria uma pasta para salvar somente os arquivos deste campo.

Código:

foto\_perfil = models.ImageField(upload\_to="perfil/")

# Model FileField

Semelhante ao ImageField, mas consegue lidar com qualquer tipo de arquivo, como documentos, PDFs, etc. Ele armazena o caminho para o arquivo no sistema de arquivos do servidor. Serve para fazer upload e armazenar documentos em um banco de dados.

Possui parâmetro opcional: ***upload\_to*** - cria uma pasta para salvar somente os arquivos deste campo.

Código:

documentos = models.FileField(upload\_to="documentos/")