SENAI-PI – UNIDADE CÂNDIDO ATHAYDE
Aluno: Josué Otoniel Pereira

Síntese sobre BD (Django)

Models: é uma biblioteca Django, com modelos pré-definidos usados para tratar os mais variados tipos de dados em uma aplicação feita por ele. De outro modo, são modelos abstraídos e simplificados, utilizados na criação de tabelas dentro do Django, para tratar os dados que serão inseridos nessas tabelas (nº inteiros, nº decimais, *string* curta, *string* longa, booleanos e etc).

Alguns tipos de *models* mais usados:

CharField: é um *model* usado quando se pretende inserir um dado do tipo *string* curta, como um título, nome ou descrição curta. Possui como **parâmetro obrigatório** *max_length* que serve para definir o tamanho máximo da *string* (até 254 caracteres).

TextField: *model* usado para inserir dados do tipo *string longa*, como descrição, comentários ou textos. Não possui **parâmetro obrigatório.**

IntegerField: usado para armazenar números do tipo *inteiro*, tanto *negativo* quanto *positivo*. Ideal para exibir informações como quantidades, temperatura, por exemplo. Não possui **parâmetro obrigatório**.

FloatField: usado para inserir dados numéricos do tipo *decimal*. Ideal para representar informações monetárias, medidas de comprimentos, constantes matemáticas e etc. Não possui **parâmetro obrigatório**.

BooleanField: *model* usado para dados do tipo *booleano* (verdadeiro/falso) ou que representem dois estados (ligado/desligado) etc. Pode ter um valor padrão definido. Possui um **parâmetro opcional**, o *default* que serve exatamente para definir um valor padrão inicial do estado.

DateFieId: esse *model* é usado para armazenar dados do tipo *data* sem a informação da hora e minutos. Muito útil para representar datas de nascimentos, data de eventos e etc. Não possui **parâmetro obrigatório**.

DateTimeFieId: semelhante ao *model* anterior, o *DateField*, este também serve para representar dados do tipo data com a diferença que também inclui as *horas* e os *minutos*. Ideal para datas de criação/atualização de registros e etc. Não possui **parâmetro obrigatório**.

ForeingFieId: este *model* é usado quando se pretende estabelecer uma relação *muitos-para-um* entre dois modelos (*models*). Por exemplo, é ideal para estabelecer a relação entre um produto e sua categoria. Possui **dois** parâmetros obrigatórios: o *"To"* – indica o modelo ao qual o model estabelece uma relação;
o *"on_delete"* – serve para especificar o que acontece aos objetos que estão relacionados com determinado *model (tabela)*, quando este é excluído.

ManyToManyFieId: esse *model* permite relacionar um modelo com outro, em uma relação de *'muitos-para-muitos'*. Isto é, permite fazer uma relação de um elemento da **tabela1** com outro elemento da **tabela2** e, este último, pode estar relacionado com vários outros elementos da **tabela1** ou outra **tabela qualquer**. Pra ficar mais claro, digamos que eu queira criar um sistema pra uma pizzaria na qual o cliente possa montar sua própria pizza com os ingredientes que desejar. Para isso, crio uma *tabela_pizza* e uma outra contendo os *tabela_ingredientes*. Usando o model **ManyToMany** para a *tabela_ingredientes*, permito com que este esteja relacionado com várias outras *pizzas* e não necessariamente a uma só. Nesse caso o Django se encarrega de criar uma terceira tabela intermediária com todos os relacionamentos entre os tipos de pizzas e ingredientes. Esse *model* possui como parâmetro obrigatório o '*To'* que indica o modelo no qual o campo está associado.

PositiveIntegerFieId: utilizado para armazenar dados numéricos do tipo *inteiro positivo*. Aceita apenas números positivo e rejeita os números negativos e zero. Ideal para ser utilizado, por exemplo, no campo de idades em um formulário. Não possui **parâmetro obrigatório**.

EmailField: *model* utilizado para armazenar endereços de e-mail e validar se o dado inserido é um endereço de e-mail válido. Não possui **parâmetro obrigatório**.

ImageField: utilizado para fazer o *upload* de arquivos de imagens (png, jpeg, svg e etc) para ser salvo em uma pasta dentro do projeto. Possui como **parâmetro opcional** (altamente recomendado como 'boas práticas' e para melhor organização do projeto) o *upload_to* que cria uma pasta para salvar as imagens na qual foram feitas o *upload* neste campo.

FileField: *model* utilizado para fazer o *upload* de qualquer tipo de documento (PDF, excel, word e etc) para ser armazenado em uma pasta dentro do projeto. Esse *model* guarda, na verdade, o caminho para a pasta criada, onde fica armazenado os arquivos. Possui como **parâmetro opcional** (altamente recomendado como 'boas práticas' e organização do projeto) o *upload_to* na qual cria uma pasta para salvar os arquivos adicionados neste campo.

OneToOneField: semelhante ao **ManyToMany**, o **OneToOne** estabelece uma relação 'um-para-um' entre duas tabelas no Django. Ou seja, em uma **tabela1,** cada elemento (objeto) desta está associado a no máximo 1 (um) elemento de uma outra **tabela2.** Possui dois parâmetros obrigatórios: *'To'* na qual indica o modelo (tabela) na qual o campo OneToOne se relaciona. E o '*on_delete'* na qual especifica o que acontece quando o elemento (objeto) relacionado é deletado.