CURSO: Técnico de informática para internet

ALUNO(A): Maria Vitória Medeiros da Silva.

**BANCO DE DADOS**

**TIPOS DE MODELS (DJANGO)**

**O que é o Módulo models no Django?**

No Django, o módulo **models** é uma parte fundamental do desenvolvimento de aplicativos web. Que você pode estruturar e o comportamentos dos seus dados de uma maneira orientada a objetos.

O módulo **models** contém as classes de modelo, que são classes Python que representam as tabelas do banco de dados.

**Por exemplo:**

class Postagem(models.Model):

titulo = models.CharField (max\_length=200)

conteudo = models.TextField()

data\_publicacao = models.DateTimeField (auto\_now\_add=True)

Neste exemplo, A **Postagem** ela define três campos: **titulo**, **conteudo** e **data\_publicacao**, que são representados no banco de dados como colunas na tabela **postagem**.

* **tipos de models Django:**

**CharField**: Armazena strings curtas,como nomes,títulos ou descrição curtas.

**Parâmetro obrigatório:** max\_length (valor maxímo de 254).

class Exemplo(models.Model):

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

**TextField**: Armazena texto de comprimento variável.

**Não possui parâmetro obrigatório.**

class Exemplo(models.Model):

descricao = models.TextField()

**IntegerField**: Armazena números inteiros.

**Não possui parâmetro obrigatório.**

class Exemplo(models.Model):

idade = models.IntegerField()

**FloatField**: Armazena números de ponto flutuante,ou seja,números decimais.

**Não possui parâmetro obrigatório.**

class Exemplo(models.Model):

preco = models.FloatField()

**BooleanField**: Armazena valores booleanos (True ou False).

**Parâmetro opcional:** Definir o valor padrão usando o argumento default.

class Exemplo(models.Model):

ativo = models.BooleanField(default=False)

**DateField**: Armazena datas.

**Não possui parâmetro obrigatório.**

class Exemplo(models.Model):

data\_criacao = models.DateField()

**DateTimeField**: Armazena data e hora.

**Não possui parâmentro obrigatório.**

class Exemplo(models.Model):

data\_hora\_criacao = models.DateTimeField()

**ForeignKey**: Define uma relação muitod para um entre dois modelos.

**Parâmetro obrigatório 1:”**to”, indica o modelo a qual a chave estrangeira se refere.

**Parâmetro obrigatório 2:** “on-delete”, especifica que os objetos referenciado é excluído.

class Exemplo(models.Model):

outro\_modelo = models.ForeignKey(OutroModelo, on\_delete=models.CASCADE)

**ManyToManyField**:

**Parâmetro obrigatório:**”to”, indica o modelo ao qual o campo many-to-many está associado.

class Livro(models.Model):

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

autores = models.ManyToManyField(Autor)

def \_str\_ (self):

return self.titulo

**BANCO DE DADOS**

**(parte 2)**

**PositiveIntegerField:** Aceita apenas valores inteiros positivos, rejeitando números negativos e 0.

Não possui parâmetro obrigatório.

class Produto(models.Model):

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

quantidade = models.PositiveIntegerField()

def \_str\_ (self):

return self.nome

**EmailField:** É usado para armazenar endereços de email em um modelo Django.

Não possui parâmetro obrigatório.

class Usuario (models.Model):

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

email = models.EmailField()

def \_str\_(self):

return self.nome

**ImageField:** É usado para armazenar imagens em um modelo Django. Ele armazena o caminho para o arquivi da imagem no sistema de arquivi do servidor.

**Parâmetro opcional:** “upload\_to”, Cria uma pasta para salva os arquivo deste campo.

class Produto(models.Model):

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

imagem = models.ImageField (upload\_to='imagens\_produto/')

def \_str\_ (self):

return self.nome

**FileField:**ppode ser usado para armazenar qualquer tipo de arquivo,como documentos,PDF s,etc.

**Parâmetro opcional: “uploud\_to”,** cria uma pasta para salvar somente os arquivos desse campo.

class Documento(models.Model):

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

arquivo = models.FileField (upload\_to='documentos/')

def \_str\_ (self):

return self.nome

**OneToOneField**: é usado para definir uma relação de um-para-um entre dois modelos.

**Parâmetro obrigatório 1**: “to”, indica o modelo ao qual o campo OneToOneField se refere.

**Parâmetro obrigatório 2**: “on=delete”, especifica que o objetos referenciado é excluído.

class Perfil(models.Model):

usuario = models.OneToOneField (User, on\_delete=models.CASCADE)

data\_nascimento = models.DateField()

bio = models.TextField()

def \_str\_ (self):

return self.usuario.username