

ESCOLA E FACULDADE DE TECNOLOGIA SENAI ROBERTO MANGE

DESENVOLVIMENTO DE SISTEMAS

JULHO DE 2022 — SENAI-SP



ESCOLA SENAI "ROBERTO MANGE"
UM. REAGRIZADO DA INDÚSTRIA

SENAI
SÃO PAULO

CRIANDO OS MODELS (TABELAS) DO BACKEND COM DJANGO

1. Os models em django são nada mais nada menos que as tabelas de dados que serão criados em seu banco, e devem estar localizadas em seu APP, portando acesse o ***models.py***
2. Crie suas tabelas usando classes e a biblioteca models do django.db, a classe é a tabela e os atributos abaixo dela são as colunas do banco, onde o tipo de dado a ser criado no database é definido em cada atributo, exemplo:

```
from django.db import models
from django.utils import timezone

# Create your models here.

class Planet(models.Model):
    name = models.CharField(max_length=150, null=False) #varchar
    climate = models.CharField(max_length=50, null=True, blank=False)
    created = models.DateTimeField(default=timezone.now()) #decimal do mysql
    diameter = models.DecimalField(max_digits=20, decimal_places=2)
    gravity = models.IntegerField() #int do mysql
    population = models.BigIntegerField() #bigint do mysql

    def __str__(self):
        return self.name
```

- Caso coloque **models.IntegerField**, por exemplo, o django irá criar uma coluna no banco do tipo **int**, e assim por diante.

- Não é necessário criar coluna id, isso é feito automaticamente pelo django.

- A função def `__str__` vai considerar o print da classe como sendo a coluna 'nome' presente na mesma



CRIANDO OS MODELS (TABELAS) DO BACKEND COM DJANGO

```
class Starships(models.Model):  
    name = models.CharField(max_length=150) # varchar do mysql  
    model = models.CharField(max_length=20, blank=False, null=True)  
    passengers = models.IntegerField(null=False) #int do mysql  
    created = models.DateTimeField(auto_now_add=True) # datetime default now() do mysql  
    length = models.DecimalField(max_digits=10, decimal_places=2)  
  
    def __str__(self):  
        return self.name
```

- Habilite se um campo não é obrigatório com `null=True`, caso não coloque nada ele considerará `null=False`, ou seja, o campo passará a ser obrigatório para inserir um registro nesta tabela desejada.

- Caso queira permitir que um campo não seja obrigatório mas que não permita a colocação de dados em branco (espaços por exemplo) você pode utilizar `null=True`, `blank=False`
- Para fazer um campo de data pegar a data/hora atual de cadastro pode-se usar o `auto_now_add=True` ou o `default=timezone.now()` (mas é necessário importar a biblioteca `timezone`)

CRIANDO OS MODELS (TABELAS) DO BACKEND COM DJANGO

3. Caso deseje fazer um relacionamento entre colunas usando chaves estrangeiras, basta utilizar a seguinte abordagem:

```
class People(models.Model):
    name = models.CharField(max_length=150) # varchar do mysql
    birthYear = models.CharField(max_length=20)
    eyeColor = models.CharField(max_length=50)
    gender = models.CharField(max_length=50, blank=False, null=True)
    hairColor = models.CharField(max_length=50)
    height = models.DecimalField(max_digits=4, decimal_places=2)
    mass = models.DecimalField(max_digits=10, decimal_places=2)
    created = models.DateTimeField(auto_now_add=True) # datetime default now() do mysql
    planet = models.ForeignKey(Planet, related_name="planet", on_delete=models.CASCADE)
    starship = models.ForeignKey(Starships, related_name="starship", on_delete=models.CASCADE, blank=True, null=True)

    def __str__(self):
        return self.name
```

- Utilize **models.ForeignKey(NOME_DA_TABELA_A_RELACIONAR, related_name="NOME_DESEJADO", on_delete="COMPORTAMENTO_AO_DELETAR_O_RELACIONAMENTO")**

CRIANDO OS MODELS (TABELAS) DO BACKEND COM DJANGO

4. Registre as suas tabelas/models em **admin.py** de modo a fazer com que elas possam ser gerenciadas pela tela de administração do django:

```
from django.contrib import admin
from .models import * #importando o People e o Planet!!!

# Register your models here.

class detPeople(admin.ModelAdmin):
    list_display = ('id', 'name', 'hairColor')
    list_display_links = ('id', 'name',)
    search_fields = ('name',)
    list_per_page = 10

#registra as configurações realizadas do model na página de admin
admin.site.register(People, detPeople)
```

- list_display: irá definir quais campos serão mostrados na página de administração em formato de tabela

-list_display_links: irá definir quais campos serão links que quando clicados na tela de admin irá abrir todos os dados da tabela

-search_fields: quais campos serão usados como chave de busca na tela de admin

-list_per_page: lista quantos dados serão exibidos por página

PASSOS PARA CRIAR O PROJETO EM DJANGO NO VSCODE

5. **py .\manage.py makemigrations** → Prepare as migrações das novas tabelas criadas (sempre que criar um novo model ou alterá-lo precisará dar este comando)
6. **py .\manage.py migrate** → Executa as migrações que foram preparadas anteriormente para o banco de dados.
7. **py .\manage.py runserver** → Executa o servidor web
8. Abra a tela de admin do django e suas tabelas já devem estar lá, podendo criar registros, consultar, excluir ou alterar.

