

Django Models - Relacionamentos



Prof. Bruno Gomes



@profbrunogomes



Aula de Hoje

2

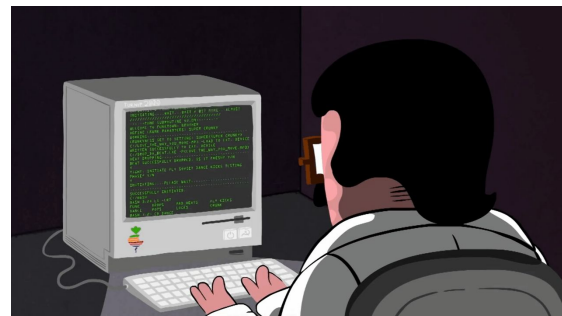
- » **Django**
 - » **Models - Relacionamentos**
 - » **Muitos para um**
 - » **Muitos para muitos**
 - » **CRUD com relacionamentos**
 - » **Tipo de Campo**



Antes de começar

3

- » Baixar **proj_08_rel.zip** do Google Sala de Aula (pasta com materiais de aula).
- » Extrair no mesmo diretório do Virtualenv.



1

Models - Relacionamentos

Exemplo da aula - Diagrama Relacional

5



Ordem de criação dos Models

6

- » Em models.py é necessário criar as classes da menos dependente para a mais dependente:

```
class Publicos(models.Model):  
    nome = models.CharField('Nome', max_length=100)  
  
class Areas(models.Model):  
    nome = models.CharField('Nome', max_length=100)  
  
class Cursos(models.Model):  
    nome = models.CharField('Nome', max_length=100)
```



1.1

Relacionamentos

Muitos para Um

Relacionamento Um para Muitos

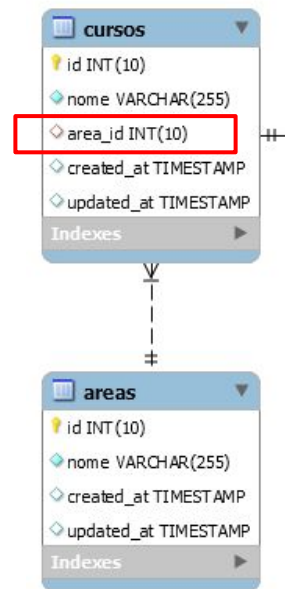
8



Relacionamento um para muitos

9

- » Neste relacionamento, se cria uma **chave estrangeira** no lado do muitos (curso).
- » Ela aponta para a **chave primária** da outra tabela do relacionamento



Chave Primária

10

- » A chave primária de todas as tabelas é gerada automaticamente.
- » Código padrão que é executado (de forma implícita):

```
id = models.AutoField(primary_key=True)
```

Chave Primária

11

- » Se precisar sobrescrever:
 - » Utilizar a opção **primary_key=True**
- » Exemplo:

```
cpf = models.CharField(primary_key=True,max_length=11)
```

Relacionamento muitos para um

12

- » Criado através da função ForeignKey:

```
models.ForeignKey(Model_Correspondente,  
                  on_delete=models.CASCADE)
```

Primeiro argumento que deve ser passado é o do Model correspondente

Relacionamento muitos para um

13

- » Em models.py, acrescentar a chave estrangeira em Cursos para Areas:

```
class Areas(models.Model):  
    nome = models.CharField('Nome', max_length=100)  
  
class Cursos(models.Model):  
    nome = models.CharField('Nome', max_length=100)  
    area = models.ForeignKey(Areas, on_delete=models.CASCADE)
```

core/models.py

1.2

Relacionamentos

Muitos para Muitos

Relacionamento Muitos para Muitos

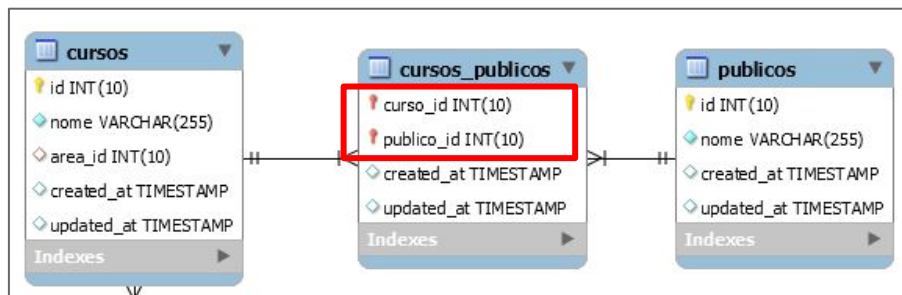
15



Relacionamento muitos para muitos

16

- » Neste relacionamento, uma tabela intermediária deve ser criada, com chaves estrangeiras para as 2 tabelas da relação:



Relacionamento muitos para muitos

17

- » Criado através da função **ManyToManyField**.
 - » Ele gera a tabela intermediária automaticamente.

```
models.ManyToManyField(Classe_Correspondente)
```

Relacionamento muitos-para-muitos

18

- » Em models.py, acrescentar a chave estrangeira em Cursos para Areas:

```
class Publicos(models.Model):  
    nome = models.CharField('Nome', max_length=100)  
  
class Cursos(models.Model):  
    nome = models.CharField('Nome', max_length=100)  
    area = models.ForeignKey(Areas, on_delete=models.CASCADE)  
    publicos = models.ManyToManyField(Publicos)
```

core/models.py

Gerar o Banco de Dados

19

- » Digitar os comandos no Terminal:

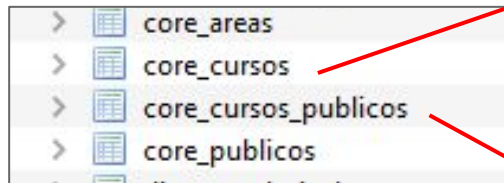
```
python manage.py makemigrations core  
python manage.py migrate
```

Terminal

Visualizar no SQLite Browser

20

» Tabelas geradas:



>	core_areas
>	core_cursos
>	core_cursos_publicos
>	core_publicos

Tabela: core_cursos

id	nome	area_id
Filtro	Filtro	Filtro

Tabela: core_cursos_publicos

id	cursos_id	publicos_id
Filtro	Filtro	Filtro

2

CRUD com Relacionamentos

CRUD do Relacionamento

22

- » 1º passo: adicionar os campos do relacionamento no ModelForm CursosForm, em **forms.py**

```
class CursosForm(ModelForm):  
    class Meta():  
        model = Cursos  
        fields = ['nome', 'area', 'publicos']
```

core/forms.py

CRUD do Relacionamento

23

- » 2º passo: adicionar os campos no template **curso_cadastro.html**:

```
<p>{{form.nome.label}}: {{form.nome}} </p>  
<p>{{form.area.label}}: {{form.area}} </p>  
<p>{{form.publicos.label}}: {{form.publicos}} </p>
```

core/templates/curso_cadastro.html

Testar no Navegador

24

- » Acessar:



Navegador

- » Cadastrar áreas e públicos;
- » Depois acessar o cadastro de Cursos

Cadastro de Curso

Nome:

Area:

Publicos:

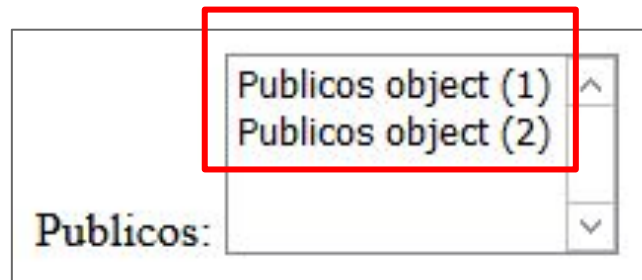
Publicos object (1)

Publicos object (2)

Problema

25

- » O formulário de cadastro exibe o nome do objeto, ao invés do nome do Público e o nome da Área:



Publicos:

Publicos object (1)
Publicos object (2)

Exibindo um valor padrão em Model

26

- » Para retornar um valor padrão ao acessar um objeto, é necessário criar uma função chamada `__str__`, e retornar o atributo que desejar do Model.
- » Exemplo de Areas em **models.py**, retornando o nome:

```
class Areas(models.Model):  
    nome = models.CharField('Nome', max_length=100)  
  
    def __str__(self):  
        return self.nome
```

core/models.py

Exibindo um valor padrão em Model

27

- » Exemplo de Publicos em **models.py**:

```
class Publicos(models.Model):  
    nome = models.CharField('Nome', max_length=100)  
  
    def __str__(self):  
        return self.nome
```

core/models.py

Testar no Navegador

28

- » Acessar:

`http://localhost:8000/`

Navegador

- » Cadastrar áreas e públicos;
- » Depois acessar o cadastro de Cursos

Cadastro de Curso

Nome:

Area: ▼

Discentes	^
Docentes	
	▼

Publicos:

Salvar

3

Tipo de Campo

Documentação

30

» <https://docs.djangoproject.com/en/4.1/ref/forms/widgets/>



Exibindo um valor padrão em Model

31

- » A alteração dos tipos de campos é possível através de **widgets**.
- » Em **forms.py**, alterar area para radio e publicos para checkboxes:

```
class CursosForm(ModelForm):  
    class Meta():  
        model = Cursos  
        fields = ['nome', 'area', 'publicos']  
        widgets = {  
            'area': forms.RadioSelect(),  
            'publicos': forms.CheckboxSelectMultiple(),  
        }
```

core/forms.py

Testar no Navegador

32

» Acessar:

`http://localhost:8000/curso_cadastro`

Navegador

Cadastro de Curso

Nome:

Area:

- ☒ -----
- ☐ Informática
- ☐ Eventos

Publicos:

- ☐ Discentes
- ☐ Docentes

Salvar

Problema

33

- » Por padrão, campos radio vem com o primeiro elemento vazio (-----).

»

Area:

- ☒ -----
- ☐ Informática
- ☐ Eventos

Eliminando o primeiro valor de Radio

34

- » Em forms.py, acrescentar o construtor `__init__`, e alterar a estrutura do campo `area` para que o valor inicial vazio não exista:

```
widgets = {  
    'area': forms.RadioSelect(),  
    'publicos': forms.CheckboxSelectMultiple(),  
}  
  
def __init__(self, *args, **kwargs):  
    super().__init__(*args, **kwargs)  
    self.fields['area'].empty_label = None
```

core/forms.py

Testar no Navegador

35

- » Acessar:

`http://localhost:8000/curso_cadastro`

Navegador

- » Cadastrar e editar pelo menos 2 cursos.

Cadastro de Curso

Nome:

Area:

- ☐ Informática
- ☐ Eventos

Publicos:

- ☐ Discentes
- ☐ Docentes

Salvar

Imprimindo o Radio manualmente

36

» No template **curso_cadastro.html**:

```
<p>Área:<br/>
{% for area in form.area %}
    {{ area }}
    <br />
{% endfor %}
</p>
```

```
<p>Área:<br/>
{% for area in form.area %}
    {{ area.tag }}
    {{ area.choice_label }}
    <br />
{% endfor %}
</p>
```

core/templates/curso_cadastro.html

Imprimindo o Checkbox manualmente

37

» No template **curso_cadastro.html**:

```
<p>Público:<br/>
{% for publico in form.publicos %}
    {{ publico }}
    <br />
{% endfor %}
</p>
```

```
<p>Público:<br/>
{% for publico in form.publicos %}
    {{ publico.tag }}
    {{ publico.choice_label }}
    <br />
{% endfor %}
</p>
```

core/templates/curso_cadastro.html

Dúvidas?

