

# Django Models Avançado



Prof. Bruno Gomes



@profbrunogomes



# Aula de Hoje

2

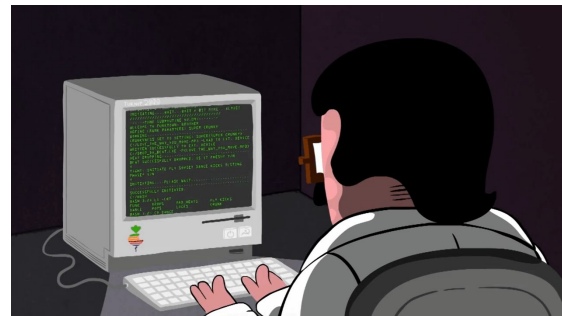
- » **Django**
  - » Customizando Forms
  - » Media
  - » Models e Arquivos
  - » Alterando o banco



# Antes de começar

3

- » Se não tiver o código do projeto até o momento, baixar **aula7.zip** da pasta do Drive;
- » Extrair no mesmo diretório do Virtualenv, e digitar os comandos:
  - » `python manage.py makemigrations core`
  - » `python manage.py migrate`
- » Iniciar servidor e testar o endereço:
  - » <http://localhost:8000/cursos>
- » Realizar as operações de Curso.



# 1.

## Customizando Forms

# Problema

5

- » Como faço para trabalhar e customizar cada um dos campos gerados pelo formulário?

```
<form action="" method="POST">
  {% csrf_token %}
  {{ form }}
<button type="submit">Salvar</button>
```

## Cadastro de Curso

Nome:

- This field is required.

Algoritmos Data de Inicio:

- This field is required.

2019-05-20 Vagas:

- This field is required.

30

# Gerando os campos do Formulário

6

- » Para gerar cada um dos campos do formulário, basta acessar os atributos através da variável form (cursos.html):

```
<form method="post">
    {% csrf_token %}
    {{form.nome}}
    {{form.data_inicio}}
    {{form.vagas}}
    <button type="submit">Salvar</button>
</form>
```

```
forms.py
1 from django.forms import ModelForm
2 from .models import Cursos
3
4 class CursosForm(ModelForm):
5     class Meta:
6         model = Cursos
7         fields = ['nome', 'data_inicio', 'vagas']
```

core/templates/cadastro.html



# Testar no Navegador

7

`http://localhost:8000/cadastro`



Navegador

## Cadastro de Curso

<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="Salvar"/>
----------------------	----------------------	----------------------	---------------------------------------

- » Obs.: A data deve ser no padrão:
  - » **2019-05-20**

# Trabalhando os Campos

8

- » Agora é possível trabalhar com cada campo de forma individual do formulário (adicionar a legenda que desejar, alterar a ordem...):

```
<p>Nome: {{form.nome}}</p>
```

```
<p>Data de Início: {{form.data_inicio}}</p>
```

```
<p>Vagas: {{form.vagas}}</p>
```

[core/templates/cadastro.html](#)



# Testar no Navegador

9



Navegador



## Cadastro de Curso

Nome:

Data de Início:

Vagas:

- » Obs.: A data deve ser no padrão:
  - » **2019-05-20**

# Legenda dos Campos

10

- » É possível também imprimir as legendas definidas no ModelForm:

```
<p>{{form.nome.label}}: {{form.nome}}</p>  
<p>{{form.data_inicio.label}}: {{form.data_inicio}}</p>  
<p>{{form.vagas.label}}: {{form.vagas}}</p>
```

[core/templates/cadastro.html](#)

# Testar no Navegador

11



## Cadastro de Curso

Nome:

Data de Início:

Vagas:

- » Obs.: A data deve ser no padrão:
  - » **2019-05-20**

# Estilizando o formulário com CSS

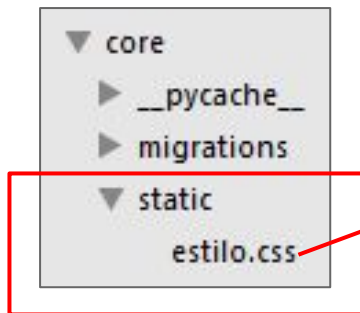
12

- » Existem várias formas:
  - » Uma delas, recomendada pela documentação, é adicionar propriedades e classes diretamente em forms.py:
    - » <https://docs.djangoproject.com/en/4.1/ref/forms/widgets/#customizing-widget-instances>
  - » Outra é utilizar a biblioteca **widget-tweaks**:
    - » <https://pypi.org/project/django-widget-tweaks/>

# Antes de começar

13

- » Criar a pasta **static** em core, dentro dela um arquivo chamado **estilo.css**, e inserir o código:



```
.campo
{
    width: 100%;
    background-color: #F5F5DC;
}
```

# Instalando widget-tweaks

14

- » No terminal, instalar a biblioteca:

```
pip install django-widget-tweaks
```

Terminal



# Adicionar a Aplicação no Projeto

15

- » Em **settings.py**, adicionar a aplicação `widget_tweaks`:

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'widget_tweaks',  
    'core',  
]
```

[site7/settings.py](#)

# Carregamento

16

- » No template **cadastro.html**, carregar static e widget\_tweaks, e adicionar um link para o arquivo estático estilo.css:

```
<!doctype html>
{% load static %}
{% load widget_tweaks %}
<html>
<head>
    <title></title>
    <link href="{% static 'estilo.css' %}" rel="stylesheet"
type="text/css">
```

core/templates/cadastro.html

# Inserindo Classes

17

- » É possível também imprimir as legendas definidas no ModelForm:

```
{{form.nome|add_class:"campo"}}  
{{form.data_inicio|add_class:"campo"}}  
{{form.vagas|add_class:"campo"}}
```

[core/templates/cadastro.html](#)

# Testar no Navegador

18



Cadastro de Curso	
Nome:	<input type="text"/>
Data de Início:	<input type="text"/>
Vagas:	<input type="text"/>
<input type="button" value="Salvar"/>	

- » Obs.: A data deve ser no padrão:
  - » **2019-05-20**

## Tag render\_field

19

- » widget-tweak contém uma tag chamada render\_field que permite escrever diretamente os atributos html na tag que será renderizada:

```
<p>{{form.nome.label}}:
```

```
{% render_field form.nome class="campo"  
placeholder="Digite o seu nome" %}
```

```
</p>
```

core/templates/cadastro.html

# Testar no Navegador

20



Cadastro de Curso	
Nome:	<input type="text"/>
Data de Início:	<input type="text"/>
Vagas:	<input type="text"/>
<input type="button" value="Salvar"/>	

- » Obs.: A data deve ser no padrão:
  - » **2019-05-20**



# 2.

## Arquivos de Media

# Documentação

22

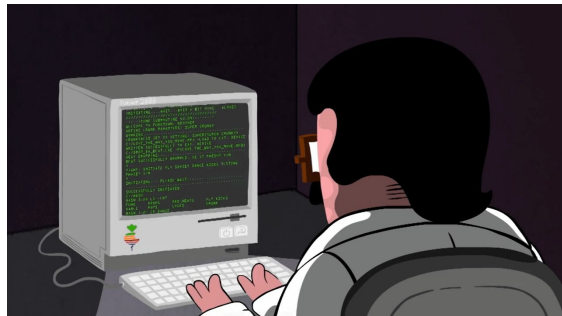
- » <https://docs.djangoproject.com/en/4.1/topics/db/models/>
- » <https://docs.djangoproject.com/en/4.1/howto/static-files/>



# Antes de começar

23

- » Se não tiver o código do projeto até o momento, baixar **aula7\_2.zip** da pasta do Drive;
- » Extrair no mesmo diretório do Virtualenv, e digitar os comandos:
  - » `python manage.py makemigrations core`
  - » `python manage.py migrate`
- » Iniciar servidor e testar o endereço:
  - » <http://localhost:8000/cursos>
- » Realizar as operações de Curso.



## Arquivos de Media

24

- » A nomenclatura Media se refere a imagens, documentos, vídeos, entre outros, que são utilizados no projeto e na maioria das vezes são salvos/criados através de **upload**:
  - » Envio de uma foto de perfil;
  - » Enviar um comprovante de residência;
  - » Enviar nota fiscal para participar de uma promoção.

## Instalar o Pillow

25

- » Para se trabalhar com uploads, é necessário instalar a biblioteca de imagens do Python.

```
pip install Pillow
```

Console

## Definindo as variáveis para media

26

- » MEDIA\_URL: nome da URL para acessar a media (inicia e finaliza com /)
- » MEDIA\_ROOT: Pasta onde ficarão os arquivos de media

```
123 STATICFILES_DIRS = [  
124     'estaticos',  
125 ]  
126
```

```
127 MEDIA_URL = '/media/'  
128  
129 MEDIA_ROOT = 'media'
```



Criar a pasta **media**  
na raiz do projeto



## Acessando Media durante Desenvolvimento

27

- » Durante o desenvolvimento, é necessário adicionar a função abaixo ao fim de urlpatterns (**urls.py**), para que o projeto enxergue os arquivos enviados via upload:

```
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    #urls
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

[aula7\\_2/urls.py](#)

# 3.

## Models e Arquivos

# Documentação

29

» <https://docs.djangoproject.com/en/4.1/topics/db/models/>



# Imagem em Curso

30

- » 1º passo: Em models.py, acrescentar um campo de imagem no modelo Cursos:

```
class Cursos(models.Model):  
    nome = models.CharField('Nome', max_length=100)  
    data_inicio = models.DateField('Data de Início', null=True)  
    vagas = models.IntegerField('Vagas', null=True)  
    foto = models.ImageField('Foto', upload_to='cursos', null=True)
```

core/models.py

# Atualizar Banco

31

- » 2º passo: Executar os comandos:

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

Console



```
upload_to='cursos'
```

- » O atributo `upload_to` é utilizado para informar o nome da pasta (que será criada dentro da pasta media) que a imagem será salva.
  - » Neste exemplo, será criada uma pasta chamada cursos.



## Adicionar o campo no ModelForm

33

- » 3º passo: Em **forms.py**, acrescentar o campo no formulário:

```
class CursosForm(ModelForm):  
    class Meta():  
        model = Cursos  
        fields = ['nome', 'data_inicio', 'vagas', 'foto']
```

core/forms.py

## Adicionar o campo no ModelForm

34

- » 4º passo: No template **cadastro.html**, acrescentar o campo de foto:

```
<p>{{form.vagas.label}}: {{form.vagas|add_class:"campo"}}</p>  
  
<p>{{form.foto.label}}: {{form.foto|add_class:"campo"}}</p>  
  
<button type="submit">Salvar</button>
```

core/forms.py

## Adicionar o campo no ModelForm

35

- » 5º passo: Ainda no template **cadastro.html**, acrescentar o atributo `enctype` para o formulário permitir envio de arquivos:

```
<form method="post" enctype="multipart/form-data">  
{% csrf_token %}
```

`core/forms.py`

## Recebendo Arquivos em Views

36

- » 6º passo: Ao criar CursosForm na função cadastro em **views.py**, acrescentar um novo atributo que verifica se recebeu arquivos do formulário:

```
form = CursosForm(request.POST or None, request.FILES or None)
```

[core/forms.py](#)

## Recebendo Arquivos em Views

37

- » 7º passo: Ao criar CursosForm na função atualizar em **views.py**, acrescentar um novo atributo que verifica se recebeu arquivos do formulário:

```
form = CursosForm(request.POST or None, request.FILES or None,  
instance=curso)
```

core/forms.py

# Testar no Navegador

38



`http://localhost:8000/cadastro`

Navegador



## Cadastro de Curso

Nome:

Data de Início:

Vagas:

Foto:

No file selected.

- » Cadastrar 1 curso com imagem.
- » Obs.: A data deve ser no padrão:
  - » **2019-05-20**

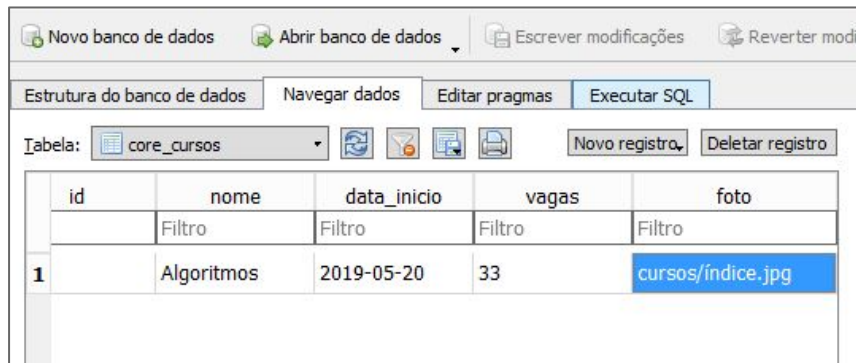
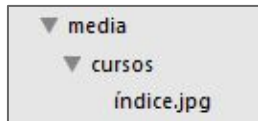


# Verificar

**Atenção**

39

- » Verificar se a pasta cursos foi criada dentro da pasta media, e a imagem está localizada lá. Além disso, verificar o que apareceu no campo foto pelo SQLite Browser:



A screenshot of the SQLite Browser application. The interface shows the 'core\_cursos' table selected. The table has five columns: 'id', 'nome', 'data\_inicio', 'vagas', and 'foto'. The first row of data is highlighted, showing '1' in the 'id' column, 'Algoritmos' in the 'nome' column, '2019-05-20' in the 'data\_inicio' column, '33' in the 'vagas' column, and 'cursos/índice.jpg' in the 'foto' column.

id	nome	data_inicio	vagas	foto
	Filtro	Filtro	Filtro	Filtro
1	Algoritmos	2019-05-20	33	cursos/índice.jpg

## Exibindo a Imagem

40

- » No template cursos.html, exibir a imagem na tag img:

```
<p>  
      
    {{curso.nome}}  
    <a href="{% url 'atualizar' curso.id %}">EDITAR</a>  
    <a href="{% url 'deletar' curso.id %}">DELETAR</a>  
</p>
```

core/templates/cursos.html

# Testar no Navegador

41



`http://localhost:8000/cursos`

Navegador



## Administração de Cursos

[Cadastrar](#)

Lista de Cursos:



Algoritmos [EDITAR](#) [DELETAR](#)

# Cadastrar mais cursos

42

## Administração de Cursos

Cadastrar

Lista de Cursos:



Algoritmos [EDITAR](#) [DELETAR](#)



Banco de Dados [EDITAR](#) [DELETAR](#)

- » A biblioteca **cleanup** faz o gerenciamento dos arquivos que estão sendo salvos no projeto. Se editar, automaticamente o arquivo anterior é excluído. Se remover o curso do banco, a imagem também será excluída.

```
pip install django-cleanup
```

Terminal

# Dica

44

- » Em **settings.py**, adicionar a aplicação cleanup:

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'widget_tweaks',  
    'django_cleanup',  
    'core',  
]
```

site7/settings.py



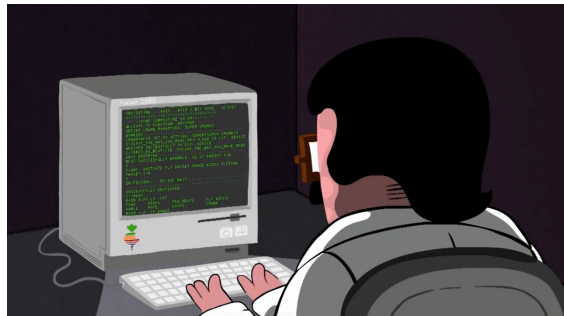
# 4.

## **Alterando o Banco de Dados**

# Antes de começar

46

- » Se não tiver o código do projeto até o momento, baixar **aula7\_3.zip** da pasta do Drive;
- » Extrair no mesmo diretório do Virtualenv, e digitar os comandos:
  - » `pip install django-widget-tweaks`
  - » `pip install Pillow`
  - » `pip install django-cleanup`
  - » `python manage.py makemigrations core`
  - » `python manage.py migrate`
- » Iniciar servidor e testar o endereço:
  - » <http://localhost:8000/cursos>
- » **Cadastrar 1 ou 2 cursos**



- » Você tem um banco de dados criado (muitas vezes já preenchido).
- » Será necessário acrescentar uma coluna em uma tabela que já existe
  - » Esta coluna não permite valores nulos ou brancos.
  - » **Problema:** se não permite nulo ou brancos, o que será inserido neste campo para as instâncias já salvas no banco?
- » Devido a esse problema, o Django não permite criar o banco. Ele oferece 2 opções: Inserir um valor padrão para as instâncias já salvas; ou cancelar operação e retornar ao projeto para corrigir.

# Exemplo

48

- » Adicionar os campos no modelo Cursos:

```
class Cursos(models.Model):  
    nome = models.CharField('Nome', max_length=100)  
    data_inicio = models.DateField('Data de Início', null=True)  
    vagas = models.IntegerField('Vagas', null=True)  
    foto = models.ImageField('Foto', upload_to='cursos', null=True)  
  
    ativado = models.BooleanField('Ativado')  
    criado = models.DateField('Criado', auto_now_add=True)
```

core/models.py

# Reconhecer Alterações

49

- » Executar o primeiro comando no terminal:

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

Terminal

# Atualizar Banco

50

- » A mensagem abaixo será exibida no terminal:

```
(Ambiente) C:\Users\bruno\Desktop\Projetos\aula7_3>python manage.py makemigrations core
You are trying to add a non-nullable field 'ativado' to cursos without a default; we can't do that (the database needs s
omething to populate existing rows).
Please select a fix:
  1) Provide a one-off default now (will be set on all existing rows with a null value for this column)
  2) Quit, and let me add a default in models.py
Select an option:
```

- » Informa que a coluna **ativado** não tem valor padrão para ser inserido nas instâncias que já existem no banco;
  - » **Se pressionar 1**, poderá informar um valor padrão para os valores já armazenados;
  - » **Se pressionar 2**, irá cancelar a operação e poderá resolver diretamente no models.py.



## Opção 1

51

- » Pressione 1 e ENTER:

```
Please select a fix:
  1) Provide a one-off default now (will be set on all existing rows with a null value for this column)
  2) Quit, and let me add a default in models.py
Select an option: 1
Please enter the default value now, as valid Python
The datetime and django.utils.timezone modules are available, so you can do e.g. timezone.now
Type 'exit' to exit this prompt
>>>
```

- » Agora está solicitando que insira o valor padrão para as instâncias que já estão armazenadas.
- » Como o campo **ativado** é do tipo Boolean, digitar:
  - » True
  - » Pressionar ENTER

## Opção 1

52

- » Em seguida, informa que o campo **criado** também não tem valor padrão:

```
>>> True
You are trying to add the field 'criado' with 'auto_now_add=True' to cursos without a default; the database needs something to populate existing rows.

1) Provide a one-off default now (will be set on all existing rows)
2) Quit, and let me add a default in models.py
Select an option:
```

- » Pressione 1, ENTER, depois digite uma data qualquer entre aspas:
  - » '2019-02-01'
  - » Pressione ENTER

```
[default: timezone.now] >>> '2019-02-01'
Migrations for 'core':
  core\migrations\0002_auto_20190525_1531.py
    - Add field ativado to cursos
    - Add field criado to cursos

(Ambiente) C:\Users\bruno\Desktop\Projetos\aula7_3>
```

# Atualizar Banco

53

- » Executar o comando no terminal:

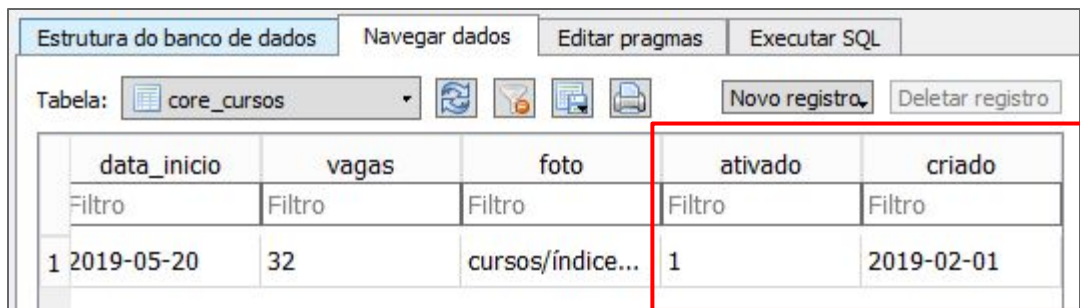
```
python manage.py migrate
```

Terminal

```
(Ambiente) C:\Users\bruno\Desktop\Projetos\aula7_3>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, core, sessions
Running migrations:
  Applying core.0002_auto_20190525_1531... OK
```

# Visualize no SQLite Browser

54



The screenshot shows the SQLite Browser interface with the 'core\_cursos' table selected. The table has five columns: 'data\_inicio', 'vagas', 'foto', 'ativado', and 'criado'. The first row of data is highlighted, showing values for each column. The 'ativado' and 'criado' columns are enclosed in a red rectangular box.

	data_inicio	vagas	foto	ativado	criado
	Filtro	Filtro	Filtro	Filtro	Filtro
1	2019-05-20	32	cursos/índice...	1	2019-02-01

## Opção 2

55

- » Se deseja resolver diretamente em `models.py`, pressionar 2, e no modelo adicionar o atributo `default=value` ou `null=True`:

```
ativado = models.BooleanField('Ativado', default=True)  
ativado = models.BooleanField('Ativado', null=True)
```

`core/models.py`

# Dúvidas?

