

Roteiro como usar o GitHub

No diretório RAIZ do projeto:

- 1) Git init
- 2) Git add .
- 3) Git config --global user.email "[email@dominio.com](#)"
- 4) Git config --global user.name "Nome usuário Github"
- 5) Git commit -m "first commit" (-am = força todos os arquivos)
- 6) Git remote add origin "path do repositório no github"
- 7) Git push --set-upstream origin main ou master

Para atualizar os projetos:

- 1) Git add .
- 2) Git commit -m "nome da alteração realizada"
- 3) Git push --set

Roteiro como inicializar o servidor AWS

Criando ambiente do projeto

- 1) Criar a pasta do projeto
- 2) Criar o ambiente virtual (python -m venv venv)
- 3) Ativar o ambiente (.\\venv\\Scripts\\activate)**
- 4) Instalar o Django (pip install Django)

Criando / Executando projeto Django

- 1) Criando o projeto (Django-admin startproject nome_principal_do_projeto .)
- 2) Executando o projeto (python **manage.py** runserver)

Estrutura/Arquivos do Projeto

- 1) Manage.py -> gerencia todas as ações do projeto (testes, migrates, execução)
- 2) Settings.py -> todas as configurações do projeto (app, bd, timezone etc)
- 3) __init__.py -> informa para o compilador que o projeto é um sistema python
- 4) Urls.py -> contém todas as rotas do projeto (www.livros.com, admin etc)

Divisões das APPs dentro do projeto Django

- 1) Cada APP gerencia uma parte do projeto (gerenciar livros, leitores, autores etc, CRUD)

Criando App no Django

- 1) Python manage.py startapp livros
- 2) Inserir a app criada no arquivo settings.py na seção **INSTALLED_APPS**

Criando o banco de dados (Migrations)

- 1) Criando o script com a estrutura da base de dados (python manage.py makemigrations) – Busca em todas as app além da estrutura padrão do Admin do Django
- 2) Executar o script para realmente gerar a estrutura do BD (python manage.py migrate)

Criando Super Usuário

- 1) Python manage.py createsuperuser
Username: pelicano
Email adress: spelicano@email.com
Password: ABC@1234d
Password(again):

Criando o models.py (Filmes)

Criar a classe dos campos da tabela que será criada no banco de dados

```

3  class Filme(models.Model):
4      id = models.AutoField(primary_key=True)
5      titulo = models.CharField(max_length=100)
6      sinopse = models.TextField()
7      duracao = models.IntegerField()
8      ano = models.IntegerField()
9      genero = models.CharField(max_length=100)
10     diretor = models.CharField(max_length=100)
11     atores = models.CharField(max_length=100)
12     # poster = models.ImageField(upload_to='posters/')
13
14     def __str__(self):
15         return self.titulo

```

- 1) python manage.py makemigrations (cria o arquivo 0001 initial.py na pasta migrations)
- 2) python manage.py migrate
- 3) Para que a tabela apareça no ADMIN temos que registrar o modelo no **admin.py**, criando a classe FilmeAdmin, mostrando quais os campos serão mostrado na tela. (importar o modelo filme)

```

1  from django.contrib import admin
2  from filmes.models import Filme
3
4  class FilmeAdmin(admin.ModelAdmin):
5      list_display = ('titulo', 'ano', 'genero', 'diretor', 'atores')
6      search_fields = ('titulo', 'ano', 'genero', 'diretor', 'atores')
7
8  admin.site.register(Filme, FilmeAdmin)

```

OBS: Configurações do arquivo **settings.py**:

- 1) Alterar LANGUAGE_CODE = 'pt-br'
- 2) Alterar TIME_ZONE = 'America/São_Paulo'

Criando chave estrangeira FK (models.py)

Criar a classe de gênero acima da classe Filme

```
3 class Genero(models.Model):
4     id = models.AutoField(primary_key=True)
5     nome = models.CharField(max_length=100)
6
7     def __str__(self):
8         return self.nome
```

Alterar o campo gênero na classe Filme

```
genero = models.ForeignKey(Genero, on_delete=models.PROTECT, related_name='filmes')
```

Criar a migrations:

Python manage.py makemigrations

Python manage.py migrate

Alterar o arquivo Admin.py, incluindo a classe de gênero.

```
class GeneroAdmin(admin.ModelAdmin):
    list_display = ('nome')
    search_fields = ('nome')

admin.site.register(Genero, GeneroAdmin)
```


Inserir um registro de foto no melder Filme (models.py)

Incluir o campo poster

```
poster = models.ImageField(upload_to='posters/', blank=True, null=True)
```

ATENÇÃO: Instalar a biblioteca pillow (pip install pillow)

Executar os comandos para criar a nova migration

Configura o arquivo **settings.py** o caminho para armazenar imagens.

```
13 import os
14 from pathlib import Path
```

```
127 MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
128 MEDIA_URL = '/media/'
```

Alterar o arquivo **urls.py** para localizar a caminho para salvar as imagens

```
17 from django.contrib import admin
18 from django.urls import path
19 from django.conf import settings
20 from django.conf.urls.static import static
21
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24 ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

VIEW / URLS

Exemplo sem o uso de templates (*somente para fins de esclarecimento das url/views*)

- 1) Na pasta principal do projeto acessar o arquivo **urls.py**
 - a. Incluir uma url
 - b. Configura a view que será chamada para esse url

```
8  urlpatterns = [  
9      path('admin/', admin.site.urls),  
10     path('filmes', filmes_view),  
11 ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)  
12
```

- 2) Criar a view no arquivo **views.py** da app filmes

```
1  from django.shortcuts import render  
2  from django.http import HttpResponse  
3  
4  def filmes_view(request):  
5      html = ''  
6      <html>  
7      <head>  
8          <title>Filmes</title>  
9      </head>  
10     <body>  
11         <h1>Filmes</h1>  
12         <ul>  
13             <li>Matrix</li>  
14             <li>Star Wars</li>  
15             <li>De volta para o futuro</li>  
16             <li>Harry Potter</li>  
17         </ul>  
18     </body>  
19     </html>  
20     ''  
21     return HttpResponse(html)  
22
```

OBS: Visão do administrador(admin) e do usuário comum do site

TEMPLATES

- 1) Dentro da app filmes criar uma pasta com nome **'templates'**, dentro dessa pasta crie um arquivo com nome **filmes.html**.
- 2) Alterar o arquivo **view.py** para acessar o template correspondente

```
1  from django.shortcuts import render
2
3  def filmes_view(request):
4      return render(request, 'filmes.html')
5
```

Django template language

- 1) No arquivo **filmes.html** incluir a tag de bloco de conteúdo ({% block content %} e {% endblock %})
- 2) Importar o model filmes no arquivo **view.py** (from filmes.models import Filme)
- 3) Alterar o arquivo **filmes.html** para incluir os dados do banco de dados

```
4  </head>
5  {% block content %}
6
7  <body>
8      <h1>Filmes</h1>
9      {% for filme in filmes %}
10         <H3> {{ filme.titulo }} | {{ filme.genero }} | {{ filme.diretor }} </H3>
11         {% endfor %}
12 </body>
13 {% endblock %}
```

ORM (Object Relation Mapping)

- 1) Incluir um filtro para pesquisas no arquivo **views.py**, alterar o objects de **all** para **filter**.

```
def filmes_view(request):  
    filmes = Filme.objects.filter(titulo='Star Wars' )
```

```
4 def filmes_view(request):  
5     # filmes = Filme.objects.filter(titulo='Star Wars' ) # busca por titulo  
6     filmes = Filme.objects.filter(genero__nome='ficção' ) # busca por genero  
7  
8     return render(request, 'filmes.html' , {'filmes': filmes})
```

Início projeto Aula 2 e 4

- 2) Model__contains (faz a pesquisa contendo somente a string)

```
4 def filmes_view(request):  
5     filmes = Filme.objects.filter(titulo__contains='Wars' )  
6  
7     return render(request, 'filmes.html' , {'filmes': filmes})
```

Objeto request (views)

- 1) Por meio do **request** o usuário pode passar **parâmetros** para explicar o que realmente ele quer ver na pesquisa

Ex. na url digitar:

127.0.0.1:8000/filmes?search=Teste

No arquivo **views.py** incluir o comando **print(request)** para a ver o resultado da requisição no terminal de comando

- 2) Com o request podemos capturar o valor que o usuário quer pesquisar.

```
4  def filmes_view(request):
5      filmes = Filme.objects.all()
6      search = request.GET.get('search')
7      if search:
8          filmes = Filme.objects.filter(titulo__contains=search)
9      return render(request, 'filmes.html' , {'filmes': filmes})
10
```

- 3) Para ordenar a lista, após o filtro pode usar o **order_by**:
 - a. `Filme.objects.all().order_by('titulo')`

Melhorando os templates e views

Obs.: Lembrando Não sou especialista em Front-end (CSS, bootstrap etc)

- 1) Alterar o arquivo **urls.py** para pesquisa encontrar a rota/url para pesquisar o filme por título

```
8  urlpatterns = [  
9      path('admin/', admin.site.urls),  
10     path('filmes/', filmes_view, name='filmes_list'),  
11 ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Incluir o CSS do arquivo filmes.html (github aulas 3 e 4)

Criando o templates base

- 1) Na pasta principal do projeto (**core**) criar a pasta **templates** copiar o arquivo **base.html** do Github das aulas 3 e 4.
- 2) No arquivo **settings.py** alterar o bloco de templates incluindo a nova pasta de templates em DIRS

```
45  TEMPLATES = [  
46      {  
47          'BACKEND': 'django.template.backends.django.DjangoTemplates',  
48          'DIRS': ['core/templates'], # adicionei essa linha  
49          'APP_DIRS': True,  
50          'OPTIONS': {  
51              'context_processors': [  
52                  'django.template.context_processors.debug',  
53                  'django.template.context_processors.request',  
54                  'django.contrib.auth.context_processors.auth',  
55                  'django.contrib.messages.context_processors.messages',  
56              ],  
57          },  
58      ],  
59  ]
```

- 3) No arquivo **filmes.html** incluir o **extends** na primeira linha para poder usar o templates base.

```
1  {% extends "base.html" %}  
2  
3  {% block content %}  
4  <style>
```

Forms no Django

- 1) Vamos começar incluindo uma nova rota no arquivo **urls.py** para direcionar para o formulário de cadastro de livros.

```

10  urlpatterns = [
11      path('admin/', admin.site.urls),
12      path('filmes/', filmes_view, name='filmes_list'),
13      path('novo_livro/', novo_filmes_view, name='novo_filmes'),
14  ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
15

```

- 2) No arquivo **views.py** de filmes, criar a função que será a view que terá o novo formulário de cadastro de filmes. (Incluir o import no **url.py**)

```

11
12  def novo_filmes_view(request):
13      return 'novo filme'
14

```

```

7  from filmes.views import novo_filmes_view
8

```

- 3) Criar o template referente ao form de cadastro do novo filme. Na pasta templates de filmes criar o arquivo **novo_filme.html** (conforme exemplo do github)

```

1  {% extends "base.html" %}
2
3  {% block content %}
4      <form method="POST" enctype="multipart/form-data">
5          {% csrf_token %}
6
7      </form>
8  {% endblock %}

```


- 4) Na pasta de filmes criar o arquivo **forms.py** (conforme exemplo do github)

```
1  from django import forms
2  from filmes.models import Genero
3
4  class FilmeForm(forms.Form):
5      (variable) sinopse: CharField (length=100)
6      sinopse = forms.CharField(widget=forms.Textarea)
7      duracao = forms.IntegerField()
8      ano = forms.IntegerField()
9      genero = forms.ModelChoiceField(Genero.objects.all())
10     diretor = forms.CharField(max_length=100)
11     atores = forms.CharField(max_length=100)
12     poster = forms.ImageField()
```

- 5) Alterar o arquivo **views.py** para inserir os dados do formulário (importar o FilmeForm do arquivo **forms.py**)

```
3  from filmes.forms import FilmeForm
```

```
13  def novo_filmes_view(request):
14      novo_filme_form = FilmeForm()
15      return render(request, 'novo_filme.html', {'novo_filme_form': novo_filme_form})
```