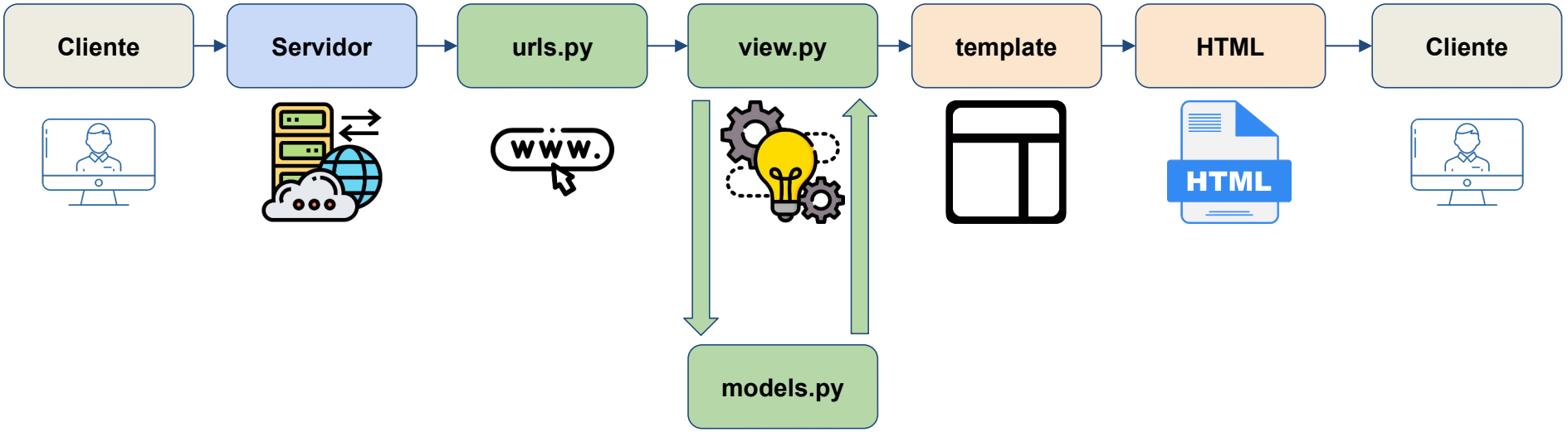
Framework Django

Projeto 1

Django

- Fluxo básico de funcionamento do Django:



Django

- **Fluxo de Requisição no Django**
- **Cliente faz a requisição:**
  - O navegador ou app envia uma requisição HTTP (ex: `GET /comentarios`).
- **Servidor Django recebe a requisição:**
  - O servidor web repassa a requisição para o Django.
- **`urls.py` identifica a rota:**
  - O Django verifica o arquivo para encontrar qual view (função) está associada àquela URL.

Django

- **Fluxo de Requisição no Django**
- **View é executada**
  - A função ou classe da view é chamada.
  - Ela pode acessar o banco de dados, processar dados, validar formulários, etc.
- **View retorna uma resposta**
  - A view pode chamar `render()` para gerar um HTML usando um template.
  - Ou retornar um json, no caso de uma API.

Django

- **Template é renderizado**
  - O Django monta o HTML final com base no template e nos dados enviados pela view.
- **Resposta é enviada ao cliente**
  - O HTML gerado é enviado de volta ao navegador, que exibe a página para o usuário.

Django - Projeto1

Mãos à obra!

Vamos criar um projeto básico com python e django.

Django - Projeto1

- Vamos criar um projeto com uma página simples de comentários.
- Instalação do Django:
  - Caso o python e o `pip` estejam configurados no PATH do sistema, basta instalar.

```
pip  install django
```

  - Caso contrário é necessário acessar a pasta do executável do `pip`. Depois fazer a instalação. (windows)

```
cd C:\Users\Fulano\AppData\Local\Programs\Python\Python313\Scripts\
 .\pip3.exe install django
```

Neste caso todos os comandos devem utilizar o caminho completo da pasta onde o python está instalado.

Django - Projeto1

- Vamos seguir com ambiente Windows. Caso use Linux alguns comandos devem ser adaptados.
- Para melhor organizar as pastas dos diferentes projetos que poderemos criar, vamos criar um ambiente isolado para cada projeto.
- Uma pasta “raiz” irá guardar as pastas dos projetos.

```
meus_projetos/
|── projeto1/
|   |── venv/
|   |── manage.py
|   └── projeto1/
|── projeto2/
|   |── venv/
|   |── manage.py
|   └── projeto2/
```

Django - Projeto1

- O ambiente virtual em Python é um espaço isolado onde você pode instalar e gerenciar pacotes (bibliotecas) sem afetar o restante do sistema ou outros projetos.
- Em outras palavras: Um ambiente virtual cria uma “mini instalação do Python” separada, com suas próprias dependências e configurações.
- Imagine que você tem dois projetos: Projeto A precisa do Django 3 e Projeto B precisa do Django 5.0. Se você instalar o Django “globalmente”, um dos dois projetos vai quebrar. Com um ambiente virtual, cada projeto tem suas próprias versões de pacotes, sem interferência entre si.

Django - Projeto1

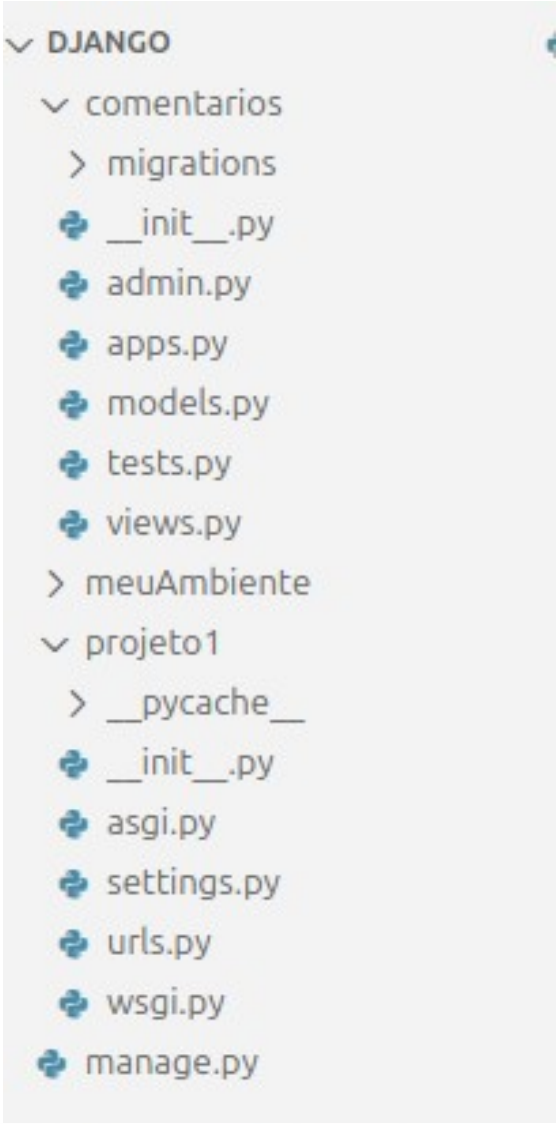
- Vamos criar o ambiente virtual para seguir com as instalações.
- Dentro de `meus_projetos > projeto1`

```
python -m venv meuAmbiente
meuAmbiente\Scripts\activate
pip install django
```

Django - Projeto1

- 1 - Criar o projeto e o app

```
django-admin startproject projeto1
cd .\projeto1\
python manage.py startapp comentarios
```



- Na lista de arquivos do vscode é possível ver os arquivos criados.

Django - Projeto1

- 2 - Criando o modelo do banco
- Em `comentarios > models.py`.

```
# Create your models here.
from django.db import models

# Modelo que representa um comentário simples
class Comentario(models.Model):
    autor = models.CharField(max_length=100)      # nome do autor do comentário (campo de texto curto)
    texto = models.TextField()                    # conteúdo do comentário (texto livre)
    data_criacao = models.DateTimeField(auto_now_add=True)  # data/hora de criação preenchida automaticamente

    def __str__(self):
        # Representação legível do objeto: "Autor: texto..." (mostra os primeiros 30 caracteres)
        return f"{self.autor}: {self.texto[:30]}"  # [:30] limita a exibição para não mostrar o comentário
completo
```

Django - Projeto1

- 3 - Criando a lógica na view
- Em `comentarios > views.py`.

```
from django.shortcuts import render, redirect  # atalhos para renderizar templates e redirecionar
from .models import Comentario                 # modelo Comentario usado para persistência

# View para salvar um comentário e mostrar o formulário/lista de comentários
def salvar_comentario(request):
   # Se o método for POST, processa o envio do formulário
   if request.method == 'POST':
       autor = request.POST.get('autor')      # obtém o valor do campo 'autor' (pode ser None)
       texto = request.POST.get('texto')      # obtém o valor do campo 'texto'
       comentario = Comentario(autor=autor, texto=texto)  # cria instância do modelo
       comentario.save()                      # salva no banco de dados
       # Redireciona para a mesma view (padrão PRG - Post/Redirect/Get) para evitar reenvio do formulário
       return redirect('salvar_comentario')

   # Se não for POST (normalmente GET), recupera todos os comentários ordenados do mais recente ao mais
antigo
   comentarios = Comentario.objects.all().order_by('-data_criacao')
   # Renderiza o template passando os comentários no contexto
   return render(request, 'comentarios/formulario.html', {'comentarios': comentarios})
```

Django - Projeto1

- 4 - Criando as urls do app comentarios
- Em comentarios, crie um arquivo `urls.py`.

```
# Rota raiz do app 'comentarios':
# - GET  -> exibe o formulário e a lista de comentários
# - POST -> envia o formulário e salva um novo comentário (a view faz PRG: redireciona após POST)
from django.urls import path
from .views import salvar_comentario

urlpatterns = [
   path('', salvar_comentario, name='salvar_comentario'),
]
```

Django - Projeto1

- 5 - Criando as urls do principal
- Em `projeto1 > urls.py`.

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
   path('admin/', admin.site.urls),
   path('', include('comentarios.urls')),  # Inclui as URLs do app 'comentarios' na raiz do projeto
]
```

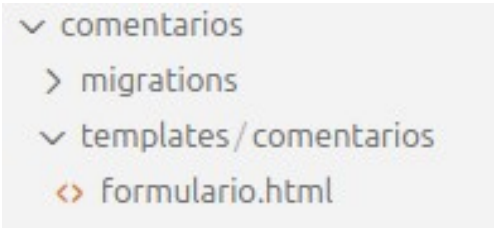
Django - Projeto1

- 6 - Template com form e uma lista.
- Neste ponto, a pasta templates deve ser criada, sendo possível duas abordagens:
  - Local: Criar a pasta templates na pasta do app
    - ex.: `comentarios > templates > comentarios > formulario.html`
  - Global: Criar a pasta templates na pasta do projeto principal
    - ex.: `projeto1 > templates > formulario.html`
    - Editar `projeto1 > settings.py`
      - incluir: `'DIRS': [BASE_DIR / 'templates'],`

Django - Projeto1

- 6 - Template com form e uma lista.
- Vamos seguir com a abordagem da pasta local
  - em comentarios criar uma pasta `templates`
  - em `comentarios > templates` - criar outra pasta chamada comentarios
  - em `comentarios > templates > comentarios` - criar o arquivo html

```
comentarios/
├── templates/
│   └── comentarios/
│       └── formulario.html
```



Django - Projeto1

- 6 - Template com form e uma lista.

```
<!-- Template que exibe o formulário para enviar um comentário e lista os comentários existentes -->

<h2>Enviar Comentário</h2>
<form method="POST">
 <!-- Proteção contra CSRF necessária para formulários POST no Django -->
 {% csrf_token %}

 <!-- Campo para o nome do autor do comentário -->
 <label>Autor:</label><br>
 <input type="text" name="autor"><br><br>

 <!-- Campo para o texto do comentário -->
 <label>Texto:</label><br>
 <textarea name="texto"></textarea><br><br>

 <!-- Botão que submete o formulário; a view faz (redireciona após POST) -->
 <button type="submit">Enviar</button>
</form>

<h2>Comentários</h2>
<ul>
 <!-- Itera sobre a lista de comentários passada pela view -->
 {% for comentario in comentarios %}
   <!-- Exibe autor e texto do comentário -->
   <li>{{ comentario.autor }}: {{ comentario.texto }}</li>
 {% empty %}
   <!-- Caso não haja comentários, mostra mensagem alternativa -->
   <li>Nenhum comentário disponível.</li>
 {% endfor %}
</ul>
```

Django - Projeto1

- 7 - Configurar o app.
- Em `projeto1 > settings.py`
- Adicionar o app comentarios, na sessão de apps instalados

```
# Application definition

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

Django - Projeto1

- 8 - Migrações
- Realizar as migrações para criar a estrutura do banco.
  - Neste caso não estamos especificando um sgbd específico, como: mysql ou postgres, então o Django usa por padrão o sqlite3.
  - Caso deseje alterar para outro sgbd, basta editar o arquivo `settings.py`

```
# Database
# https://docs.djangoproject.com/en/5.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Django - Projeto1

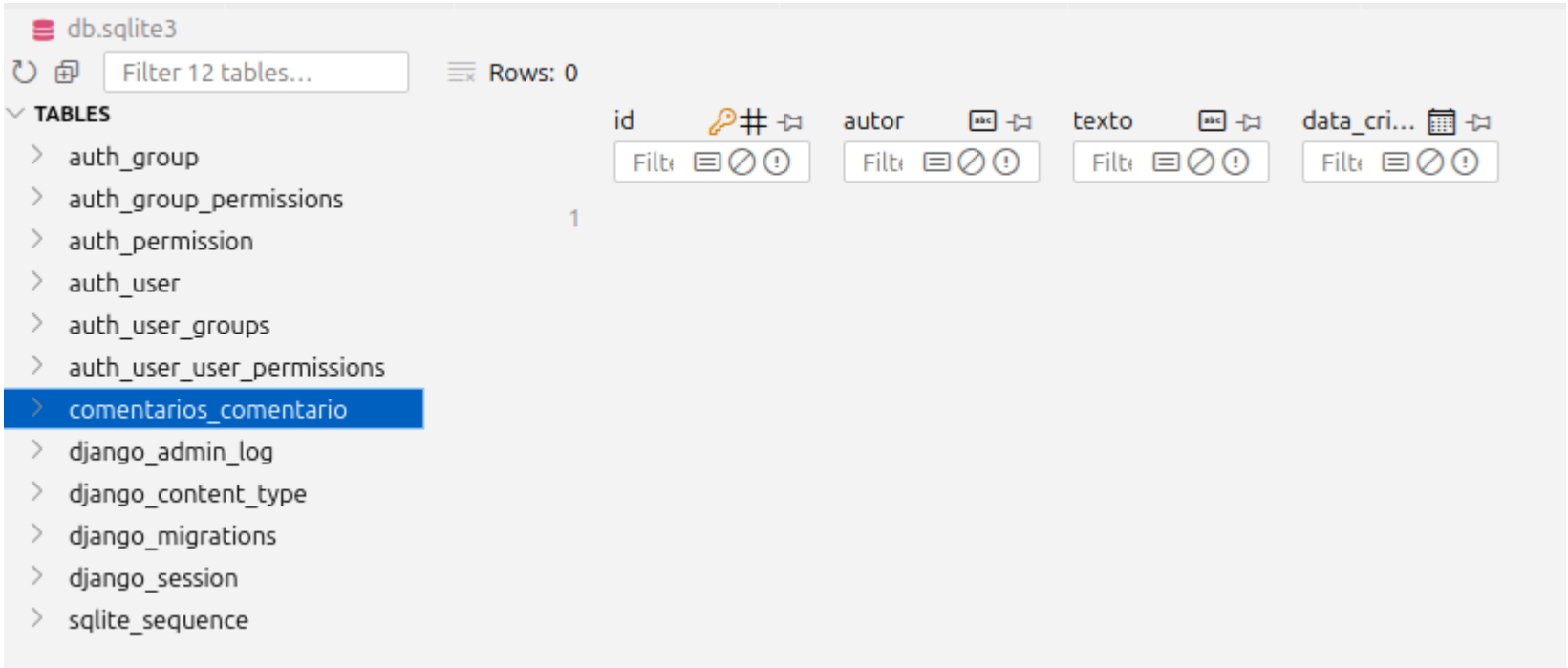
- 8 - Migrações
- Realizando as migrações

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

- Após realizar as migrações o django irá criar um arquivo chamado db.sqlite3, com as tabelas do banco.
- No vscode, com a extensão SQLite Viewer é possível visualizar as tabelas e seu conteúdo.

Django - Projeto1

- 8 - Migrações
- Além das tabelas no nosso projeto, ele também cria as tabelas que ele precisa para funcionar.



Django - Projeto1

- 9 - Subir o servidor
- Agora vamos executar o servidor e testar.

```
python manage.py runserver
```

```
(meuAmbiente) aluno@osboxes:~/Django$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 30, 2025 - 12:57:46
Django version 5.2.7, using settings 'projeto1.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/
```

- Na saída é possível ver o link onde o server está ouvindo
  - `127.0.0.1:8000` ou
  - `localhost:8000`

Django - Projeto1

- 9 - Subir o servidor
- Teste no navegador

Django - Projeto1

- 9 - Subir o servidor
- De volta ao banco no vscode, vemos os comentários salvos.

Django - Projeto1

- Primeiro projeto funcional de Django Web.
- Use esse projeto como base para expandir seu conhecimento.

FIM