

# Fuctura

## Desenvolvimento WEB com Django

Módulo 2 - Aula 5

---



Prof. Jéssica Andrade

# Bora Revisar?

**1º) Oque é uma View?**

**2º) Oque é CSRF?**

**3º) Oque é método HTTP?**

**4º) Oque é StatusCode HTTP?**

**5º) Oque são os statusCode da familia 400?**



**6º) O que são os statuscode da familia 500?**

**7º) O que são os statuscode da familia 200?**

**8º) Cite 2 statusCode da familia 500 e para que servem?**

**9º) Cite 2 statusCode da familia 200 e para que servem?**

**10º) Cite 2 statusCode da familia 400 e para que servem?**

**11º) Para que serve a anotação @login\_required?**

Sobre os exercícios propostos nas ultimas aulas, sigo no aguardo do link do github de vocês, lembrando que a partir de agora, todos os exercícios serão aceitos apenas via link do github.

Qual o link do linkedin de vocês? Vamos criar conexões?

Oque vamos fazer hoje?

Na aula passada desenvolvemos toda a parte de criação de views e regras de negocio do nosso projeto.

Nesta aula vamos continuar com nosso projeto, criando nossos template, que serão como htmls visuais, para que assim possamos os usuarios consigam acessam nosso sistema.

**Passo 1:** Abra o projeto dentro da sua virtualenv como aprendido anteriormente em aula. Deve ser executado no cmd(windows), no terminal linux ou no bash do git

## Windows

- O comando abaixo vai depender do seu windows
- `cd venv/Scripts`
- 1. `source activate` ou `activate` ou `./activate`**
- Este pode ser executado em qualquer versão
- 2. `cd ../..`**
- 3. `cd barbearia_pdv`**

## Linux

1. `source`  
`venv/bin/activate`
3. `cd barbearia_pdv`

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <title>{{ titulo }}</title>
</head>
<body>
    <h1>{{ titulo }}</h1>
    <p>Bem-vindo, {{ usuario.nome }}!</p>
    <ul>
        {% for item in lista %}
            <li>{{ item }}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

Exemplo de template django.

No django temos tags específicas. Por exemplo, usamos {{variavel}} para carregar informações dentro de um template.



As tags de template são delimitadas por **{%variavel%}** e outras por **{{variavel}}**. Dentro dessas chaves, você pode usar várias instruções que o Django interpreta e executa ao renderizar o template.

Diferente do python que usa indentação para pular pedaços do código, no django, usamos tags específicas para abrir e fechar blocos.

```
{% block title %}Título Padrão{% endblock %}
```

A tag **with** permite que você defina uma variável temporária. Neste exemplo, a variável **total** é definida como **usuario.conta.total** e pode ser usada dentro do bloco **with**.

```
{% with total=usuario.conta.total %}  
    <p>Total: {{ total }}</p>  
{% endwith %}
```

Assim como no python, podemos herdar ou estender de outros templates. Muito cuidado, estender é diferente de importar templates filhos! Veremos melhor mais a frente

```
{% extends 'base.html' %}
```

Assim como no python, possuímos algumas tags de controle, ou seja, é possível fazer ifs e else no django, de forma que podemos criar validações. Um exemplo disso usaremos no nosso projeto em seguida.

```
{% if usuario %}
    <p>Olá, {{ usuario.nome }}!</p>
{% else %}
    <p>Olá, visitante!</p>
{% endif %}
```

Temos tags de repetição, para controle de valores mutiplos

```
<ul>  
    {% for item in lista %}  
        <li>{{ item }}</li>  
    {% endfor %}  
</ul>
```

E tambem temos filtros, para que possamos limitar a saida de algumas variaveis

```
<p>Data:  {{ data|date:"d/m/Y" }}</p>
```

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}Título Padrão{% endblock %}</title>
</head>
<body>
    <header>
        <h1>Meu Site</h1>
    </header>
    <main>
        {% block content %}{% endblock %}
    </main>
    <footer>
        <p>Rodapé do site</p>
    </footer>
</body>
</html>
```

É possível herdar valores de um template para outro, de forma que podemos chamar e compartilhar templates filhos, em partes de templates pais. Veremos na prática no nosso projeto.

Se você quiser adicionar comentários que não apareçam no HTML renderizado, você pode usar a tag de comentário.

```
{% comment %} <a href="{% url 'criar_cliente' %}" class="btn">Criar Cliente</a> {% endcomment %}
```

```
{# Este é um comentário e não aparecerá no HTML renderizado #}
```



**Variáveis:** Use `{{ variavel }}` para exibir dados dinâmicos.

**Tags:** Use `{% tag %}` para lógica de controle, como loops e condicionais.

**Condicionais:** Use `{% if %}` e `{% else %}` para exibir conteúdo com base em condições.

**Loops:** Use `{% for item in lista %}` para iterar sobre listas e dicionários.

**Blocos:** Use `{% block nome %}` para definir seções que podem ser substituídas em templates filhos.

**Extensão:** Use `{% extends 'base.html' %}` para herdar de um template base.

**Substituição:** Use `{% block %}` para definir áreas que podem ser substituídas em templates filhos.

**Inclusão:** Use `{% include 'template.html' %}` para incluir o conteúdo de outro template.

**Manipulação de Dados:** Use filtros como `{{ variavel|filtro }}` para modificar a saída, como `{{ data|date:"d/m/Y" }}`.

**Renderização de Formulários:** O Django facilita a renderização de formulários. Use `{{ form.as_p }}`, `{{ form.as_table }}` ou `{{ form.as_ul }}` para renderizar formulários de maneira rápida e fácil.

**Validação:** Certifique-se de exibir mensagens de erro de validação de forma clara para o usuário.

`{% load static %}`: Esta linha carrega o módulo de arquivos estáticos, permitindo que você use a tag `{% static %}`.

`{% static 'css/style.css' %}`: Esta tag gera a URL correta para o arquivo CSS **style.css** que está localizado na pasta **static/css/**.

`{% static 'images/logo.png' %}`: Similarmente, isso gera a URL para a imagem **logo.png**.

`{% static 'js/script.js' %}`: Isso gera a URL para o arquivo JavaScript **script.js**.

**{% url %}**: permite que você separe a lógica de URL da lógica de apresentação. Se você mudar a URL em **urls.py**, não precisará alterar todos os links em seus templates, apenas a definição da URL

```
<a href="{% url 'criar_cliente' %}">Criar Novo Cliente</a>
```

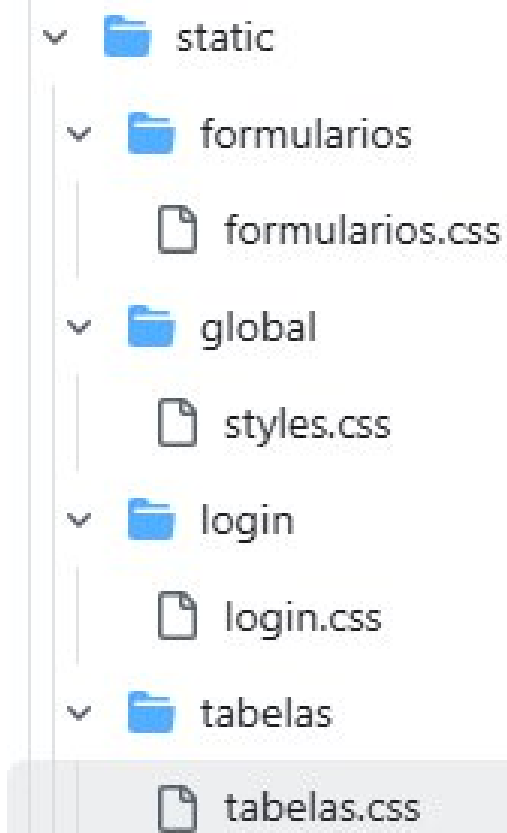
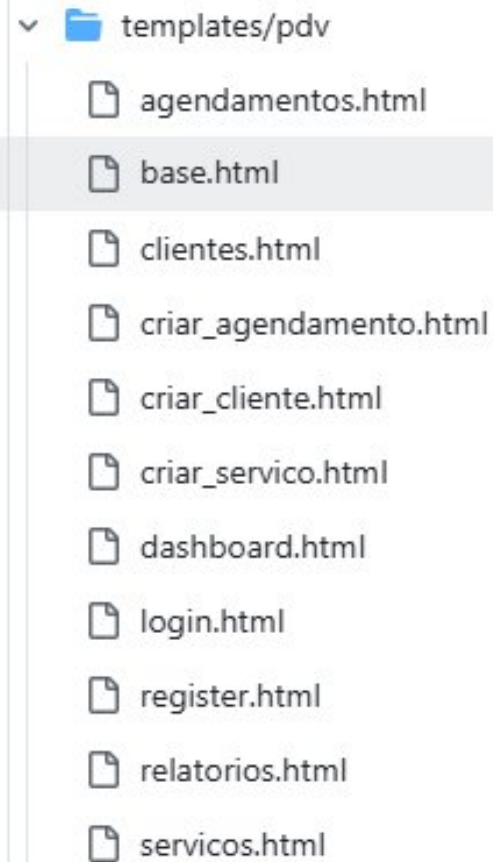
**{% block extra\_styles %}{% endblock %}** é usada em templates Django para definir um bloco de conteúdo que pode ser substituído em templates que estendem um template base. Essa abordagem é parte do sistema de herança de templates do Django, que permite que você crie uma estrutura de layout reutilizável e modular.

**{% block %}** Um bloco é uma seção de um template que pode ser preenchida ou substituída em um template filho.

**{% block title %}**: Título da pagina template

**Vamos a praticar  
no nosso  
projeto**

**Crie os arquivos e pastas de dentro da pasta static e template/pdv para que fique exatamente abaixo.**



**MUITO CUIDADO PARA SALVAR TODOS OS ARQUIVOS E COM CADA LETRA, VIRGULA E PONTO MAISCULO OU MINUSCULO.**

**Link para os htmls**

<https://github.com/jessicaprofessora/projeto-barbearia/tree/main/pdv/templates/pdv>

**Link para os css**

<https://github.com/jessicaprofessora/projeto-barbearia/tree/main/pdv/static>



# DJANGO

---

- Para facilitar a visualização de vocês, estou disponibilizando o link do github do projeto no seu modelo final, para não repetir as imagens, é melhor vocês terem nesse ultimo momento o html para comparar.
- Por favor, não copiem código, usem de comparação para escrever seus códigos, e analisar oque foi criado ate agora em todas as aulas. Comparem todos os arquivos, não apenas os htmls, mas as views, models, forms, urls e demais que foram alterados nas aulas 2, 3, 4 e 5.
- Este é o projeto final da barbearia, usem este projeto como base para realizar o projeto de vocês do hotel.

Vocês conseguem!

<https://github.com/jessicaprofessora/projeto-barbearia/tree/main>



**É hora de  
descoisar as  
coisas  
que estão  
coisadas**



[www.artpoin.com](http://www.artpoin.com)