

### Django

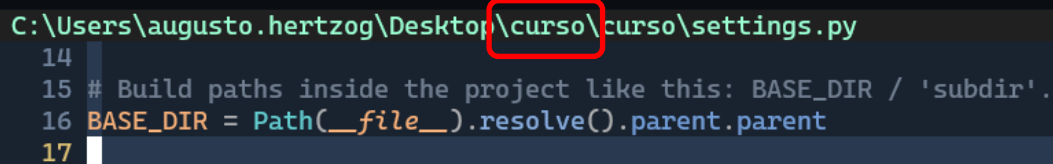
#### Colisão de Nomes

Vamos olhar melhor como está sendo a importação do arquivo /linguagem\_cpp/templates/inicio.html.

Agora, o Django está buscando o arquivo inicio.html dentro da nossa pasta /linguagem\_cpp/templates. Esse nome de templates é o padrão para o Django. Esse nome pode ser alterado no arquivo /curso/settings.py, mas isso pode deixar o fluxo do nosso código muito confuso já que estaríamos saindo do padrão do Django.

Contudo, podemos adicionar uma nova pasta de templates personalizada em /curso/settings.py. Ali dentro, temos a constante **TEMPLATES**. Dentro dela, temos um dicionário com uma chave chamada DIRS, que possui como valor uma lista.

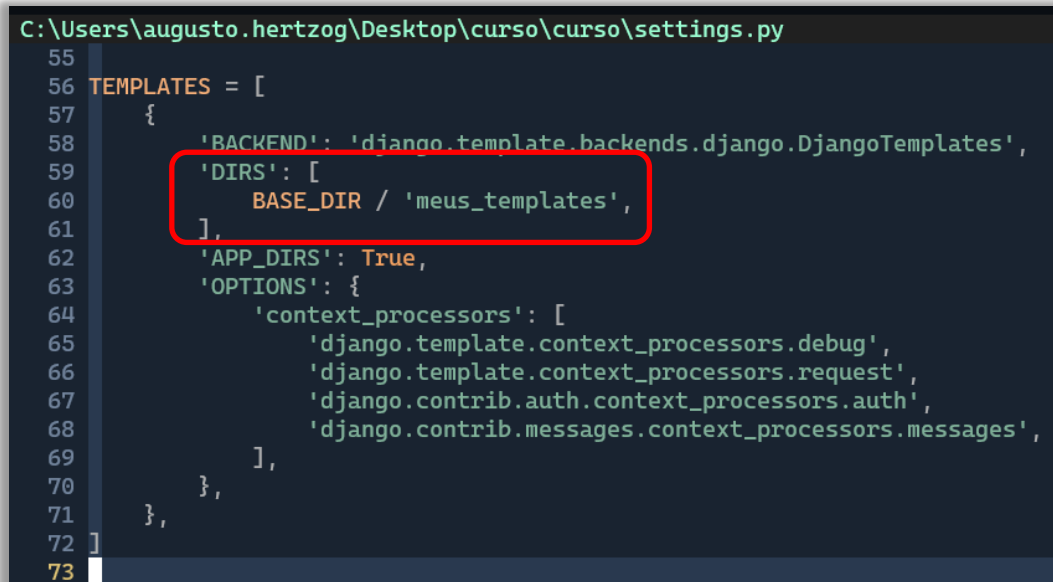
Esse mesmo arquivo também tem uma constante chamada BASE\_DIR. Ela serve para que o Django saiba exatamente qual é a pasta raiz de nosso projeto (destaque na imagem abaixo).



```
C:\Users\augusto.hertzog\Desktop\curso\curso\settings.py
14
15 # Build paths inside the project like this: BASE_DIR / 'subdir'.
16 BASE_DIR = Path(__file__).resolve().parent.parent
17
```

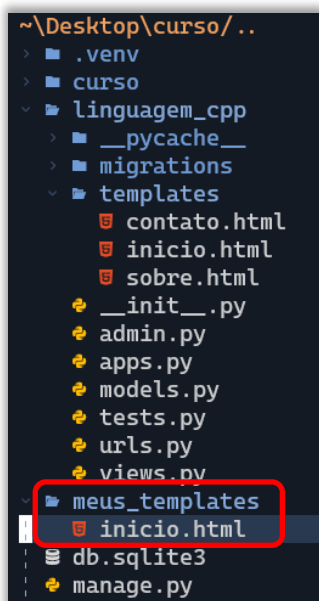
Usando-a, podemos especificar uma nova pasta na raiz do nosso projeto e adicionar lá arquivos HTML que terão um conteúdo em comum para nosso site (como um cabeçalho ou um rodapé, por exemplo).

Veja abaixo como vai ficar com essa alteração:



```
C:\Users\augusto.hertzog\Desktop\curso\curso\settings.py
55
56 TEMPLATES = [
57     {
58         'BACKEND': 'django.template.backends.django.DjangoTemplates',
59         'DIRS': [
60             BASE_DIR / 'meus_templates',
61         ],
62         'APP_DIRS': True,
63         'OPTIONS': {
64             'context_processors': [
65                 'django.template.context_processors.debug',
66                 'django.template.context_processors.request',
67                 'django.contrib.auth.context_processors.auth',
68                 'django.contrib.messages.context_processors.messages',
69             ],
70         },
71     },
72 ]
73
```

Agora vamos criar a pasta que citamos acima (cria-la na raiz do nosso projeto, junto com as pastas `curso` e `linguagem_cpp`). Dentro dela, vamos criar um arquivo HTML chamado `inicio.html` e adicionar um conteúdo dentro.



Uma vez realizada a configuração acima, nossa página <http://127.0.0.1:8000/cpp/> vai mudar de:



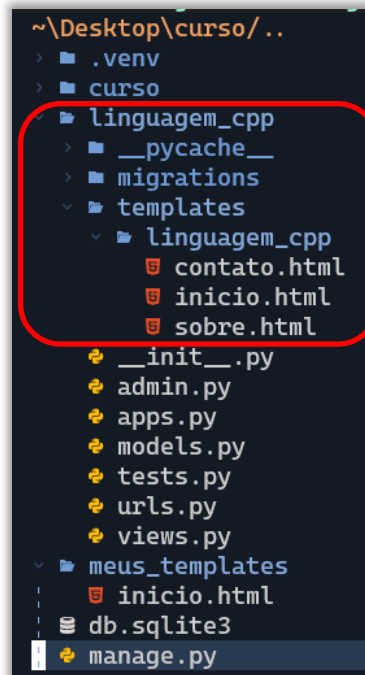
para:



Por que isso aconteceu? O Django carrega o arquivo `inicio.html` que ele encontrar primeiro. Como a pasta `/meus_templates/` está em uma hierarquia acima da pasta `/linguagem_cpp/templates/`, ele é carregado primeiro.

Isso é chamado de Colisão de Nomes. Para evitar isso, usamos um **namespace** para nossos arquivos dentro das pastas de conterão os arquivos HTMLs, CSSs e Js. Fazemos isso criando uma outra pasta dentro de `/linguagem_cpp/templates/` do nome do nosso aplicativo, que terá o mesmo nome do aplicativo. Depois movemos todos nossos arquivos HTML lá para dentro.

Veja como vai ficar a nova estrutura:



Agora que nossos HTMLs estão em uma nova pasta, temos que atualizar a chamada deles nas funções da /linguagem\_cpp/views.py.

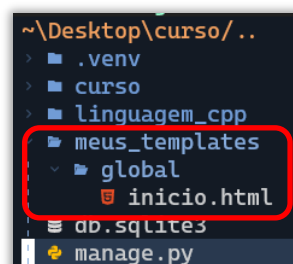
Veja como vai ficar:

```
C:\Users\augusto.hertzog\Desktop\curso\linguagem_cpp\views.py
13 from django.shortcuts import render
12
11
10 def view_home(request):
9     dicionario = {'nome': 'Tom', 'sobrenome': 'Cruise'}
8     return render(request, 'linguagem_cpp/inicio.html', context=dicionario)
7
6
5 def view_sobre(request):
4     return render(request, 'linguagem_cpp/sobre.html')
3
2
1 def view_contato(request):
14     return render(request, 'linguagem_cpp/contato.html')
```

Agora as páginas do nosso aplicativo estão sendo carregadas corretamente da pasta /linguagem\_cpp/templates/linguagem\_cpp/<arquivo>.html.

Para evitar problemas de usarmos algum arquivo da pasta /meus\_templates/ por acidente (afinal, tudo lá dentro está sendo buscado pelo Django agora) criaremos uma pasta chamada de **global** dentro dela. Depois que colocarmos nossos arquivos HTML lá dentro, sabemos que, se quisermos chamar algum deles, teremos que usar a chamada '**global/<arquivo>.html**'.

Veja como vai ficar a estrutura de pastas:



Veja uma função usando a chamada do HTML global:

```
C:\Users\augusto.hertzog\Desktop\curso\linguagem_cpp\views.py
13 from django.shortcuts import render
12
11
10 def view_home(request):
9     dicionario = {'nome': 'Tom', 'sobrenome': 'Cruise'}
8     return render(request, 'linguagem_cpp/inicio.html', context=dicionario)
7
6
5 def view_sobre(request):
4     return render(request, 'linguagem_cpp/sobre.html')
3
2
1 def view_contato(request):
14     return render(request, 'global/inicio.html')
```

E veja como ficará a página dentro do aplicativo:



## Dividir e Conquistar

Agora, vamos começar a organizar nossas páginas web.

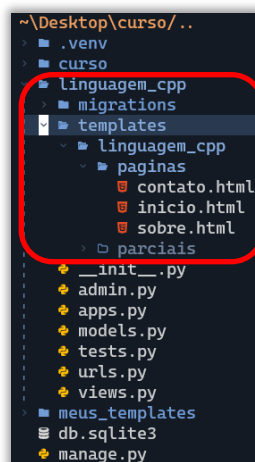
Lembra que quando temos muita repetição de código, é porque podemos simplificar? Pois bem, o Django compartilha da mesma filosofia.

Dentro da pasta /linguagem\_cpp/templates/linguagem\_cpp/, vamos criar outras duas pastas chamadas de **paginas** (sem acento mesmo) e **parciais**. Essas pastas servirão para organizarmos melhor nossos arquivos HTML.

A pasta **parciais** vai guardar os arquivos HTML que conterão apenas trechos de código HTML.

A pasta **paginas** vai guardar os arquivos HTML que representarão as nossas páginas web do aplicativo.

Veja como vai ficar a nova estrutura:



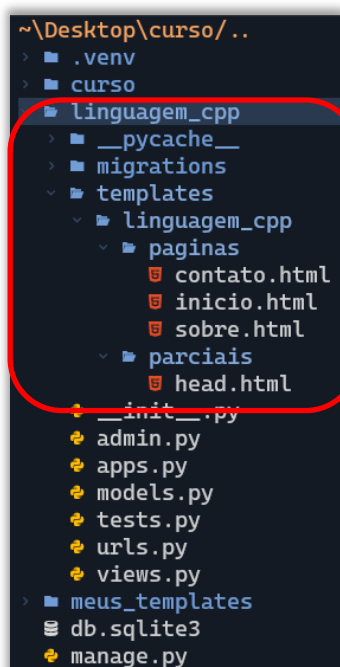
Como ainda não temos nenhum arquivo HTML com apenas trechos de código, nossa pasta **parciais** está vazia, enquanto que a pasta **pagina** está com os arquivos que criamos anteriormente (lembre-se de atualizar a chamada desses arquivos em /linguagem\_cpp/views.py).

```
C:\Users\augusto.hertzog\Desktop\curso\linguagem_cpp\views.py
13 from django.shortcuts import render
12
11
10 def view_home(request):
9     dicionario = {'nome': 'Tom', 'sobrenome': 'Cruise'}
8     return render(request, 'linguagem_cpp/paginas/inicio.html', context=dicionario)
7
6
5 def view_sobre(request):
4     return render(request, 'linguagem_cpp/paginas/sobre.html')
3
2
1 def view_contato(request):
14     return render(request, 'global/inicio.html')
```

Vamos criar nosso primeiro arquivo parcial. Ele vai guardar todo o conteúdo da tag HEAD do nosso aplicativo. Vamos chamá-lo de **head.html** que vai ficar na pasta /linguagem\_cpp/templates/linguagem\_cpp/parciais/head.html.

Depois de criado e copiado, removeremos a tag e todo seu conteúdo dos arquivos HTML que estão na pasta /linguagem\_cpp/templates/linguagem\_cpp/paginas/\*. Isso porque, agora iremos importar o arquivo **head.html** para dentro de todos eles.

Veja abaixo como vai ficar a estrutura de arquivos:



Um dos arquivos da pasta **paginas** (todos os arquivos nessa pasta tem que ficar sem a tag head agora):

```
C:\U\A\D\curso\linguagem_cpp\templates\linguagem_cpp\paginas\inicio.html
2 <!DOCTYPE html>
1 <html lang="en">
3
1 <body>
2     <h1>Sou o HOME do site</h1>
3     <p>exibindo o nome completo {{nome}} {{sobrenome}}</p>
4 </body>
5 </html>
```

O arquivo head.html:

```
C:\U\A\D\curso\linguagem_cpp\templates\linguagem_cpp\parciais\head.html
3 <head>
2   <meta charset="UTF-8">
1   <meta name="viewport">
4   <title>Páginas C++</title>
1 </head>
```

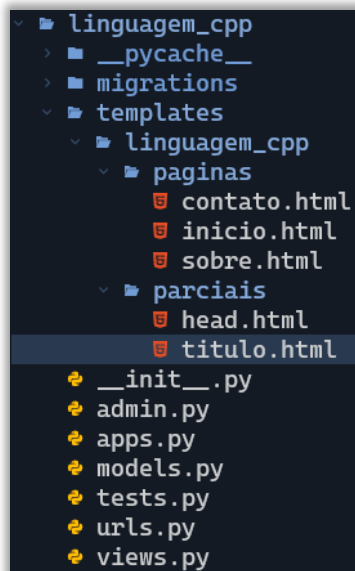
Uma vez realizada as etapas acima, temos que importar o arquivo head.html para as nossas páginas. Para fazer isso, temos que usar a tag do Django {% <conteúdo> %}. Anteriormente nós usamos {{ <variável> }} para exibir uma variável, mas se quisermos executar um trecho de código Python do Django em nosso HTML, temos que usar a tag do Django acima.

Veja como vai ficar um dos arquivos HTML na pasta **paginas**:

```
C:\U\A\D\curso\linguagem_cpp\templates\linguagem_cpp\paginas\inicio.html
1 <!DOCTYPE html>
2 <html lang="en">
3   {% include 'linguagem_cpp/parciais/head.html' %}
4 <body>
5   <h1>sou o HOME do site</h1>
6   <p>exibindo o nome completo {{nome}} {{sobrenome}}</p>
7 </body>
8 </html>
```

Digamos que queremos criar um texto padrão na tag h1 para todas as páginas do nosso aplicativos, criamos um arquivo parcial chamado **titulo.html** na pasta **parciais** e importamos ele para todos nossos arquivos na pasta **paginas**.

Veja como ficará a estrutura de arquivos:



O código no novo arquivo:

```
C:\U\A\D\c\linguagem_cpp\templates\linguagem_cpp\parciais\titulo.html
1 <h1>Páginas C++</h1>
2
```

Um dos arquivos com a chamada do arquivo:

```
C:\U\A\D\curso\linguagem_cpp\templates\linguagem_cpp\paginas\inicio.html
1 <!DOCTYPE html>
2 <html lang="en">
3     {% include 'linguagem_cpp/parciais/head.html' %}
4 <body>
5     {% include 'linguagem_cpp/parciais/titulo.html' %}
6     <p>exibindo o nome completo {{nome}} {{sobrenome}}</p>
7 </body>
8 </html>
```

Por que tudo isso? Imagina que você precise alterar/adicionar um arquivo CSS, JS, uma foto padrão no cabeçalho de todas as páginas do aplicativo etc. É muito mais fácil realizar essa tarefa alterando em apenas UM arquivo e todos os trocentos que o importam serão automaticamente atualizados.

## Arquivos Estáticos

Arquivos estáticos são arquivos que são entregues exatamente como estão salvos. Esses arquivos quase não sofrem alterações, por isso, o navegador pode salvá-los em cache para que o conteúdo da página carregue mais rapidamente para o usuário final.

Alguns tipos de arquivos que podem ser considerados estáticos:

- Imagens
- Vídeos
- HTML (como o Django já busca por eles, podemos desconsiderar ele em especial)
- CSS
- JavaScript
- Arquivos para download etc.

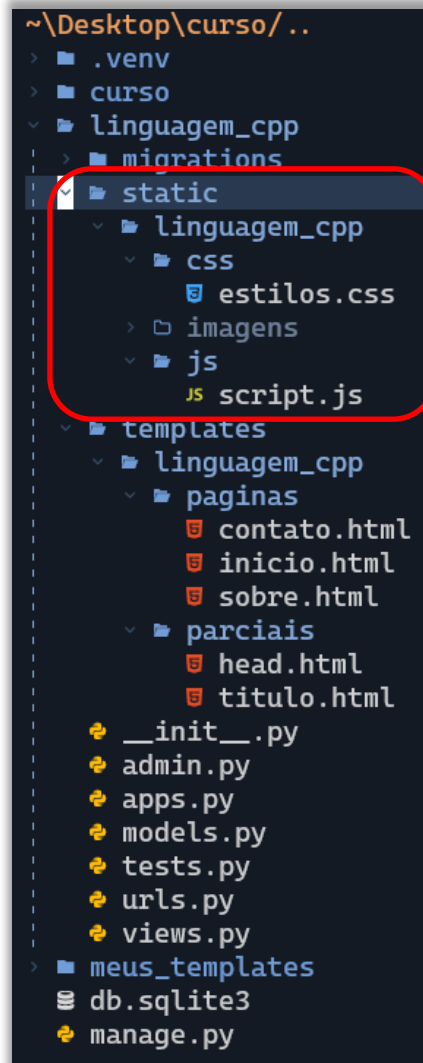
Agora, vamos criar um arquivo CSS para guardar todo o nosso código de estilo em um arquivo separado.

O Django já vem configurado para ler arquivos estáticos de dentro do aplicativo (na nossa pasta /linguagem\_cpp/).

Para isso, temos que criar uma pasta em /linguagem\_cpp/ chamada de static. Assim como a pasta /linguagem\_cpp/templates/, temos que adicionar um namespace dentro dela. Então, vamos criar uma pasta linguagem\_cpp dentro da pasta static.

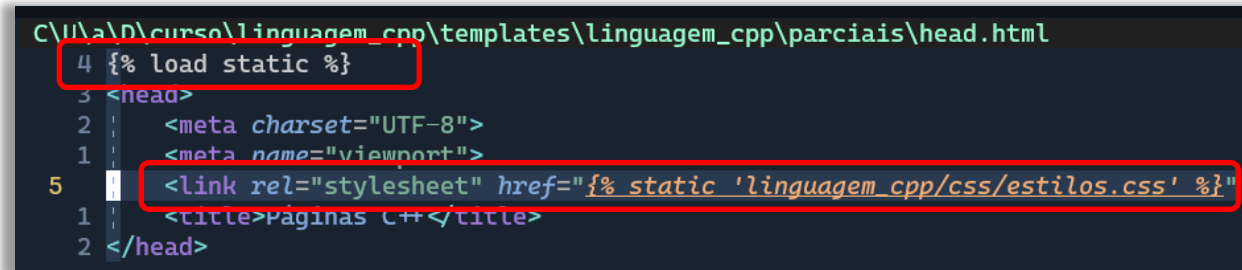
Dentro da pasta /linguagem\_cpp/static/linguagem\_cpp/ vamos criar as pastas que seriam usadas para uma aplicação web tradicional, como css, js, imagens etc.

Veja como vai ficar a estrutura abaixo:



Agora, temos que realizar a chamada desses arquivos nos nossos arquivos HTML.

Veja como vai ficar:



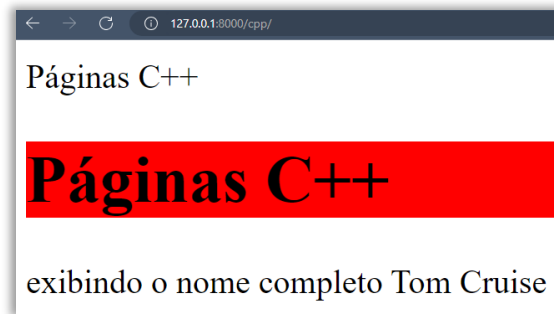
No primeiro destaque da imagem, temos que usar a tag Django para explicitar que esse arquivo HTML vai ter que carregar algum arquivo estático. Dentro dela colocamos **load static**.

No segundo destaque, usamos novamente a tag Django para importar o arquivo. Agora, iremos novamente usar o **static**, mas seguido do caminho onde está o arquivo que queremos importar. Nesse caso é o arquivo CSS.

PS.: se acontecer do CSS não ser carregado, reinicie o servidor.



Veja como vai ficar nossa página web com o CSS:



## Exercícios para Praticar

1. Se ainda estiver inseguro quanto aos passos feitos até agora, execute novamente o passo a passo da aula 01, 02, 03 e 04 de uma só vez. Provavelmente a partir da próxima aula já haverá etapas demais para fazer do começo. Então essa será a última vez que será pedido.
2. Se já estiver seguro quando às aulas 01, 02 e 03, repita os passos apenas dessa aula em um novo aplicativo.
3. Assim como foi feito com o CSS, experimente carregar para suas páginas um arquivo JS e algumas imagens.
4. Com tudo o que foi feito até agora, crie um currículo formatado usando HTML e CSS.