

Criação de página web com Django

Django é um framework (ambiente) de desenvolvimento de software que simplifica significativamente o processo de criação de aplicações web, usando Python.

O Django dá relevância à reutilização de componentes, também conhecido como o princípio DRY (Don't Repeat Yourself), e vem com recursos prontos para utilização, como sistema de login, ligação com base de dados e operações CRUD (Create Read Update Delete).

O Django segue o padrão de design MVT (Model View Template) que separa o processo de desenvolvimento em 3 camadas:

Model – Definição do modelo conceptual de base de dados no modelo (lógico) relacional. Um modelo em Django é uma classe Python que representa uma tabela de uma base de dados relacional. O processo de codificação SQL para a criação e manipulação da BD é automatizado. É, portanto, uma camada de acesso a dados e que lida com eles.

Os modelos, geralmente, estão localizados num ficheiro chamado `models.py`.

View - Um manipulador de requisições que retorna o modelo e conteúdo relevantes - com base na solicitação do utilizador. Contém o código-fonte referente à lógica da aplicação (business logic). É a interface do utilizador — o que se vê no navegador ao renderizar um site (representado por HTML/CSS/Javascript).

As view, geralmente, estão localizados num ficheiro chamado `views.py`.

Template – Define a interface do sistema através de um conjunto de páginas HTML que, por sua vez, definem a interação web com o utilizador. São ficheiros de HTML contendo o layout da página web e que descreve como o sistema será representado. Um template é geralmente um ficheiro .html, com código HTML.

Instalação do django dentro de um ambiente virtual

Ativa o teu ambiente virtual:

- No PowerShell do Windows: `.\nome_amb_virtual\Scripts\Activate.ps1` ;
- No cmd: `.\nome_amb_virtual\Scripts\activate.bat` ;

Se necessário muda para a localização da pasta onde está instalado o ambiente virtual e executa: `nome_amb_virtual\Scripts\activate.bat`

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\DD\ESCOLA 22-23\PI6\PSD 11 - PI6\virtual_Python\venv> cd..  
PS C:\DD\ESCOLA 22-23\PI6\PSD 11 - PI6\virtual_Python> venv\Scripts\activate  
(venv) PS C:\DD\ESCOLA 22-23\PI6\PSD 11 - PI6\virtual_Python> pip install django
```

Após instalação do Django

```
Collecting asgiref<4,>=3.5.2  
  Downloading asgiref-3.6.0-py3-none-any.whl (23 kB)  
Collecting tzdata  
  Downloading tzdata-2022.7-py2.py3-none-any.whl (340 kB)  
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 340.1/340.1 kB 4.2 MB/s eta 0:00:00  
Collecting sqlparse<=0.2.2  
  Downloading sqlparse-0.4.3-py3-none-any.whl (42 kB)  
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.8/42.8 kB 2.2 MB/s eta 0:00:00  
Installing collected packages: tzdata, sqlparse, asgiref, django  
Successfully installed asgiref-3.6.0 django-4.1.4 sqlparse-0.4.3 tzdata-2022.7  
(venv) PS C:\DD\ESCOLA 22-23\PI6\PSD 11 - PI6\virtual_Python>
```

Para instalar uma versão exata do django fazer:

```
>> pip install django==4.1.4
```

Para verificar qual versão do django instalada:

```
(venv) PS C:\DD\ESCOLA 22-23\PI6\PSD 11 - PI6\virtual_Python> django-admin --version  
4.1.4  
(venv) PS C:\DD\ESCOLA 22-23\PI6\PSD 11 - PI6\virtual_Python>
```

Criar um projeto em django:

No terminal fazer:

```
>>django-admin startproject <nome_do_projeto> .
```

```
(venv) PS C:\DD\ESCOLA 22-23\PI6\PSD 11 - PI6\virtual_Python> django-admin startproject djangoweb .  
(venv) PS C:\DD\ESCOLA 22-23\PI6\PSD 11 - PI6\virtual_Python>
```

<nome_do_projeto> será o nome do projeto django e o ponto (.) para evitar criar o projeto numa nova pasta com o mesmo nome e onde ficariam os ficheiros do projeto.

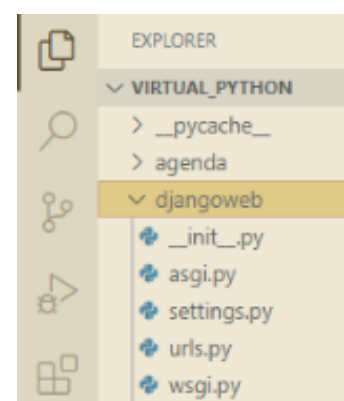
django-admin - é o utilitário de linha de comando do Django para tarefas administrativas.

Neste caso, é criada a pasta do projeto de nome *djangoweb*, cujo conteúdo é constituído por 5 ficheiros:

__init__.py Inicialmente vazio, identifica a pasta como um pacote Python para a integração e manipulação de recursos do projeto Django;

settings.py , tem todas as configurações do projeto;

urls.py , fornece uma maneira de navegar pelas diferentes páginas de um site;



wsgi.py e **asgi.py** contém as referências de funcionamento das requisições entre a aplicação web e um servidor (definem os servidores compatíveis para o desenvolvimento do projeto Django).

O ficheiro **wsgi.py**, possui as referências de requisições do tipo WSGI (*Web Server Gateway Interface*) que representa a interface de requisição do tipo síncrona, onde é realizada uma requisição de cada vez para um servidor e que implica ficar em espera até receber uma resposta para que seja realizada uma nova requisição.

Já no ficheiro **asgi.py**, teremos as referências de requisições do tipo ASGI (*Asynchronous Server Gateway Interface*), nessa interface a requisição é do tipo assíncrona, onde após realizarmos uma requisição para o servidor, não é necessário esperarmos pela resposta da primeira requisição antes de enviarmos outras requisições, permitindo a realização de requisições em paralelo.

O ficheiro **manage.py**

É criado automaticamente em cada projeto Django. Faz a mesma coisa que o **django-admin**, mas também define a variável de ambiente **DJANGO_SETTINGS_MODULE** para que aponte para o arquivo settings.py de um projeto. Esta ferramenta permite interagir e configurar o projeto Django em desenvolvimento.

Geralmente, ao trabalhar num único projeto Django, é mais fácil usar manage.py do que django-admin. Se for necessário alternar entre vários ficheiros de configurações do Django, usar django-admin com DJANGO_SETTINGS_MODULE ou a opção de linha de comando --settings.

Cada secção de uma página web será entendida no python como uma app.

Uma app em Python é um aplicativo da web que tem um significado específico num projeto, como uma página inicial, um formulário de contacto ou uma base de dados de utilizadores registados.

Antes de iniciar a construção da página iniciaremos um servidor local (do django). No terminal inicia o servidor local:

```
>>python manage.py runserver
```

Caso não tenhas os DLL necessários à utilização do servidor instalada (servidor local do django) obterás o erro seguinte:

```
File "C:\ProgramData\Anaconda3\lib\sqlite3\dbapi2.py", line 27, in <module>
    from _sqlite3 import *
ImportError: DLL load failed while importing _sqlite3: Impossível localizar o módulo especificado.
```

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.13 (ver

O SQLite - é uma base de dados relacional de código aberto e que dispensa o uso de um servidor na sua atuação. Armazena os ficheiros dentro de sua própria estrutura, e é capaz de funcionar muito bem em aplicações diversas, principalmente, websites de tráfego médio e sistemas mobile.

Faz o download das bibliotecas adequadas ao teu sistema operativo em:

<https://www.sqlite.org/download.html>

Precompiled Binaries for Windows

sqlite-dll-win32-x86-3400000.zip (559.71 KiB)	32-bit DLL (x86) for SQLite version 3.40.0. (sha3: 261df51a967a86adb937a205520edc55f5a1d5a3ed32e20f21177d748da12e)
sqlite-dll-win64-x64-3400000.zip (895.97 KiB)	64-bit DLL (x64) for SQLite version 3.40.0. (sha3: f931db5ba9aa65a98493d2ca820d2a8f3640fbda1696d014ef72724663c95485)

Verificar a localização da pasta DLL

```
C:\Windows\system32\cmd.exe

(base) C:\Users\Silvia Martins>pip -V
pip 22.2.2 from C:\ProgramData\Anaconda3\lib\site-packages\pip (python 3.9)

(base) C:\Users\Silvia Martins>
```

Copiar os ficheiros `sqlite3.dll` e `sqlite3.def` para a pasta DLLs:
C:\ProgramData\Anaconda3\DLLs

Volta a executar o comando:

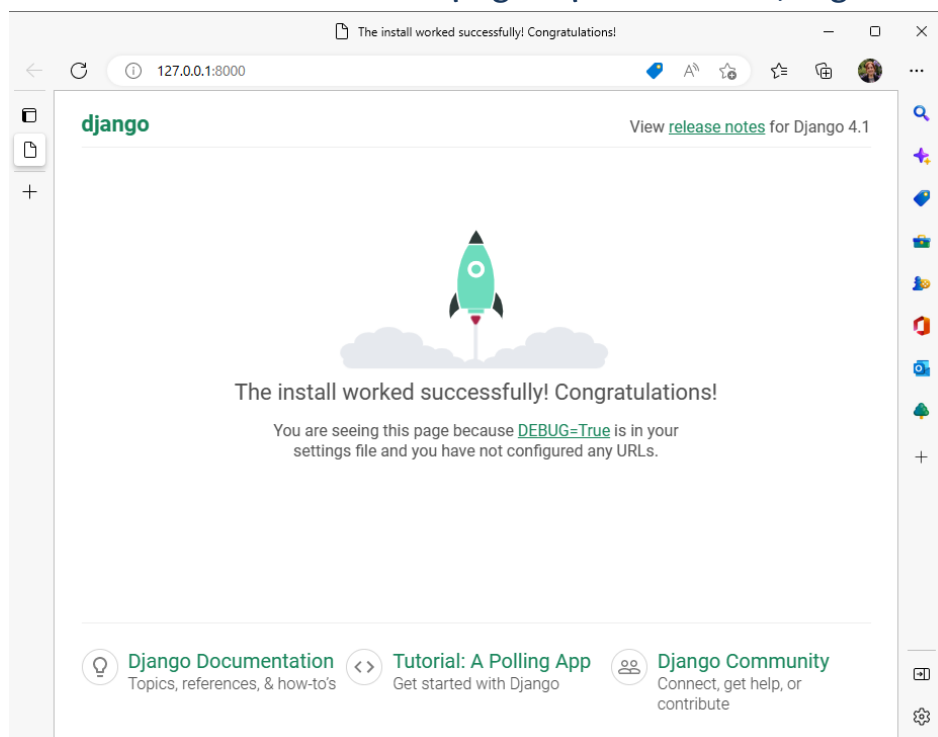
```
>>python manage.py runserver
```

Deverás ter um resultado semelhante ao que se segue:

```
all You have 18 unapplied migration(s). Your project may not work properly until you ap
    contenttypes, sessions.
★ 5 Run 'python manage.py migrate' to apply them.
    December 21, 2022 - 21:43:28
all Django version 4.1.4, using settings 'djangoweb.settings'
    Starting development server at http://127.0.0.1:8000/
★ 5 Quit the server with CTRL-BREAK.
```

Para fechar o servidor fazer CTRL+c

Ao clicares no link abrirá a página padrão do Django:



Fecha o servidor.

O que é uma app (aplicação) num projeto Django?

Para cada funcionalidade do projeto web, uma aplicação (app) pode ser criada como um módulo completamente independente.

Por exemplo, na criação de um Blog, devem ser criados módulos separados para Comentários, publicações, Login/Logout, etc. No Django, esses módulos são conhecidos como app. Existindo uma app diferente para cada tarefa.

Vantagens na utilização de **app** Django

- ✓ As **app** são reutilizáveis, podendo ser usadas em vários projetos;
- ✓ É uma componente quase independente;
- ✓ Várias pessoas podem trabalhar em diferentes componentes;
- ✓ A deputação e organização do código é facilitada;
- ✓ Possui recursos integrados, como páginas de administração, etc. reduzindo o esforço de criar componentes a partir do zero.

Apps pré-instaladas

O Django fornece algumas aplicações pré-instaladas para os utilizadores.

Para ver as **app** pré-instaladas, navegue até `projectName -> settings.py` no ficheiro `settings.py`, encontra-se **INSTALLED_APPS**. As **app** aí listadas (**INSTALLED_APPS**) são fornecidos pelo Django para facilitar o trabalho do desenvolvedor.

Como criar uma **app** (sessões dentro do site) e um **template** para cada **app**

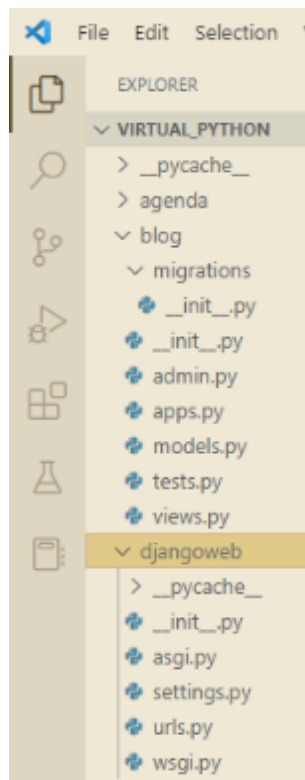
Criar uma **app** (de nome blog)

Executa o comando:

```
>> python manage.py startapp blog
```

```
December 21, 2022 - 21:54:18
Django version 4.1.4, using settings 'djangoweb.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
(venv) PS C:\DD\ESCOLA 22-23\PI6\PSD 11 - PI6\virtual_Python> python manage.py startapp blog
(venv) PS C:\DD\ESCOLA 22-23\PI6\PSD 11 - PI6\virtual_Python>
```

A pasta **blog** ficará criada no mesmo nível da pasta do projeto – **djangoweb**



Sempre que se cria uma **app**, este deve ser registada em dois locais – `settings.py` e `url.py`.

1. Em `settings.py`, adicionar em **INSTALLED_APP** a linha '**blog.apps.BlogConfig**', sendo **BlogConfig** uma classe do ficheiro `apps.py`:

```
djangoweb > settings.py > ...
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'blog.apps.BlogConfig'
41 ]
```

2. Cria um novo ficheiro de nome `urls.py` na pasta **blog**

3. Na pasta do projeto – `django web` , no ficheiro `urls.py` :

a. Importar o método `include`, da classe `django.urls`

b. adicionar na variável `urlpatterns` a linha:

`path ('blog/', 'includeblog.urls'))`



```
14 | 2. Add a URL to urlpatterns: path('blog/', inclu
15 | """
16 | from django.contrib import admin
17 | from django.urls import path, include
18 |
19 | urlpatterns = [
20 |     path('admin/', admin.site.urls),
21 |     path('blog/', include('blog.urls')),
22 |
23 | ]
```

3. Abre o ficheiro `views.py`, da app `blog`:



```
blog > views.py
1 | from django.shortcuts import render
2 |
3 | # Create your views here.
```

No ficheiro `views.py`, da app `blog`, cria o método `index` (1.º ficheiro a ser chamado, por convenção- podia ter outro nome):


```
settings.py  urls.py blog 1  urls.py.djangoweb

blog > views.py > ...
1  from django.shortcuts import render
2  from django.http import HttpResponse
3
4  # Create your views here.
5
6  def index(request):
7      return HttpResponse('Olá Mundo!')
8
```

No ficheiro urls.py, criado na app blog escrever o código seguinte (sendo parte dele padrão):

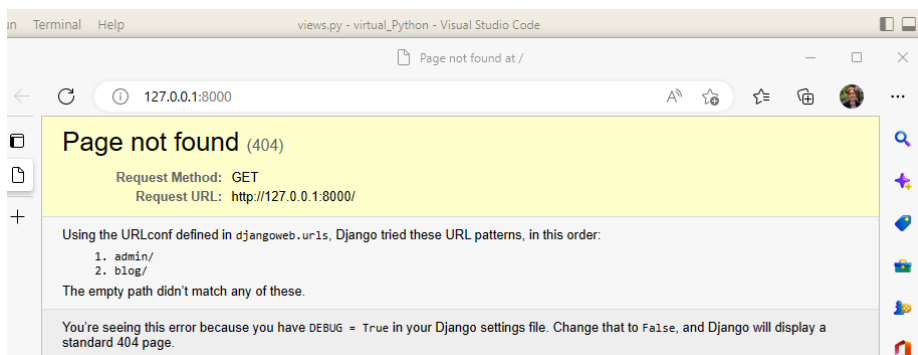
```
settings.py  urls.py blog 1 x  urls.py.djangoweb  views.py 2

blog > urls.py > ...
1  from django.urls import path      # referencia o ficheiro urls
2  from . import views                # o ponto referencia o próprio ficheiro
3
4  urlpatterns = [
5      path('', views.index)
6  ]
```

Volta a executar o comando, para iniciar o servidor:

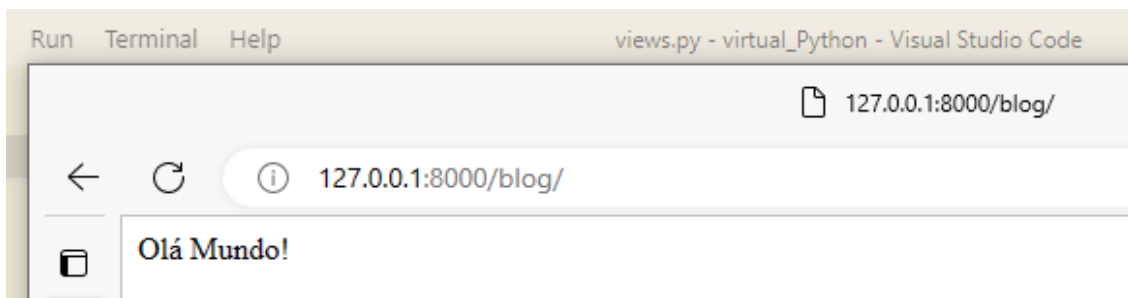
```
>>python manage.py runserver
```

Abre o URL



Este erro acontece porque a nossa página está definida dentro da app blog...
... logo, acrescentamos o nome da app, no

caminho do endereço da página index



Fecha o servidor (ctrl +c)

Criar uma nova app (de nome 'produto')

Executa o comando:

```
>> python manage.py startapp produto
```

Vais, agora, repetir os passos que foram executados aquando da criação da aplicação blog:

1.º registar a app em – settings.py e url.py, na pasta do projeto –.djangoweb

settings.py

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'blog.apps.BlogConfig',  
    'produto.apps.ProdutoConfig',  
]
```

urls.py

```
19 urlpatterns = [  
20     path('admin/', admin.site.urls),  
21     path('blog/', include('blog.urls')),  
22     path('produto/', include('produto.urls')),  
23 ]  
24
```

2.º Criar o ficheiro `urls.py`, na aplicação produto (correspondente a)

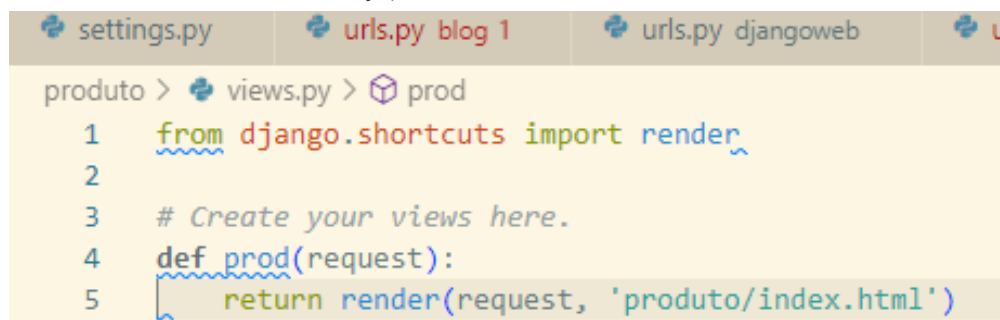
3.º No ficheiro `urls.py` criado inserir o mesmo código que foi escrito para o ficheiro `urls.py` da app `blog`, mudando apenas o nome do método para **prod** (em vez de `index`)

4.º Criar uma nova pasta de nome **templates**, pasta onde o django irá procurar os ficheiros `html`.

Dentro dessa pasta criar uma outra pasta com o mesmo nome da app, onde serão colocados os ficheiros `html` da app **produto**.

Cria, ainda, dentro da pasta `templates/produto` um ficheiro `html`, com o nome **index.html**

5.º No ficheiro `views.py` escrever o método **prod**:



```
settings.py | urls.py blog 1 | urls.py.djangoweb | ui
produto > views.py > prod
1  from django.shortcuts import render
2
3  # Create your views here.
4  def prod(request):
5      return render(request, 'produto/index.html')
```

6.º Escrever no ficheiro `index.html` criado a frase ‘Olá mundo!’

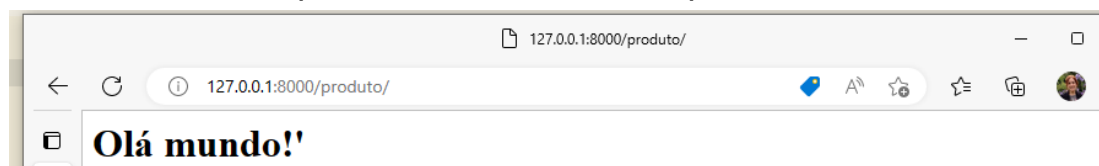


```
settings.py | urls.py blog 1 | urls.py d
produto > templates > index.html > h1
1  <h1>Olá mundo! </h1>
```

Volta a executar o comando, para iniciar o servidor:

```
>>python manage.py runserver
```

Abre o URL `http://127.0.0.1:8000/produto`



Ao invés de “Olá Mundo”, digita em `index.html` ‘`html`’ e seleciona ‘`html 5`’, para criar um template para `html 5`.

Será gerado o código seguinte:



```
index.html x | settings.py
produto > templates > produto > index.html > html > head > meta
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9  <body>
10
11 </body>
12 </html>
```

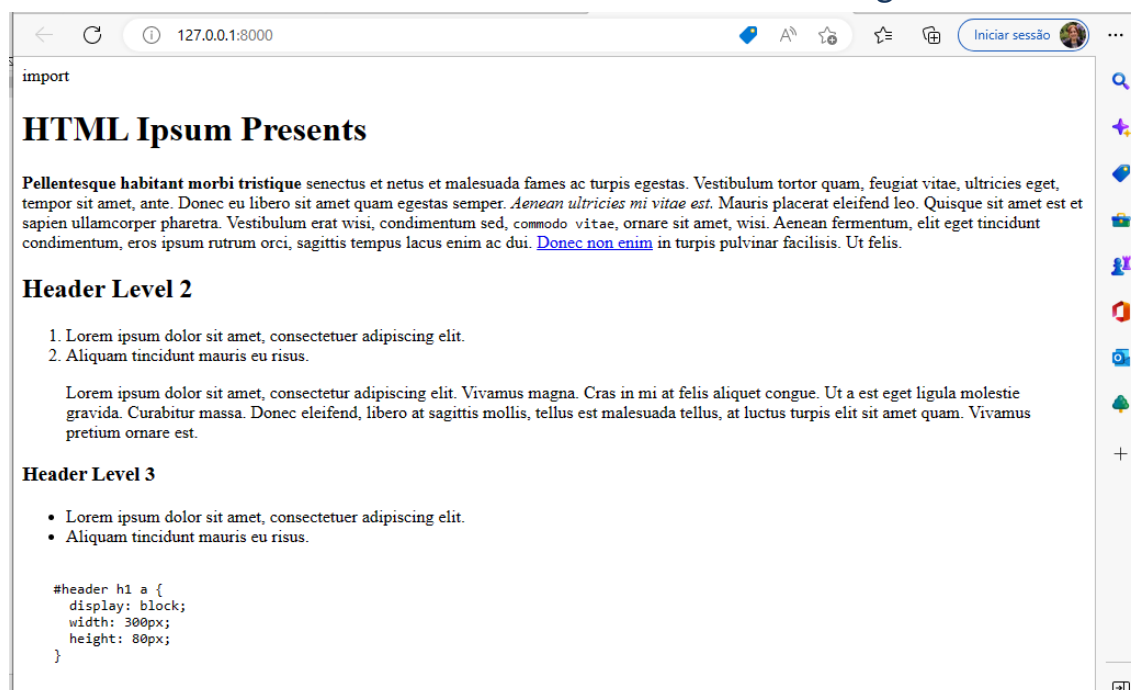
Pesquisa na net um html de base “lorem ipsum”, por exemplo em:

<https://www.webfx.com/web-design/html-ipsum/>

copia, por exemplo, o modelo ‘Kitchen Sink’ e cola-o para a tag body do ficheiro.

Antes de executares o servidor muda, no ficheiro settings.py do projeto, o valor da variável: ROOT_URLCONF = 'produto.urls' (em vez do nome do projeto)

Ao executares o servidor visualizarás no browser o seguinte:



Bibliografia

Vasconcelos, J. (2015). Python - Algoritmia e Programação Web. Lisboa:FCA

<https://cursos.alura.com.br/forum/topico-arquivo-asgi-py-236340>

<https://www.treinaweb.com.br/blog/principais-comandos-do-django-cli>

<https://docs.djangoproject.com/en/4.1/ref/django-admin/>

<https://www.javatpoint.com/how-to-connect-mysql-to-django>

https://www.w3schools.com/django/django_create_virtual_environment.php

<https://www.sqlite.org/download.html>

<https://www.javatpoint.com/django-app>

https://www.dev2qa.com/how-to-fix-importerror-dll-load-failed-while-importing_sqlite3-the-specified-module-could-not-be-found/

Para Bd MySQL: <https://www.youtube.com/watch?v=fG1-OurK5CI>

Curso de Python 3 do básico ou Avançado – Udemy

<https://www.geeksforgeeks.org/django-project-mvt-structure/>

[Como criar um aplicativo em Django? - GeeksforGeeks](#)

<https://www.treinaweb.com.br/blog/criando-um-app-com-django>