



Django do Zero

Como Criar um Aplicativo Web

Canal youtube.com/@VamosCodar - Wanderson Reis

03 - Primeira Aplicação - Olá Django

IMPORTANTE: As aulas são sequenciais, por isso praticamente todas as aulas dependem da implementação do conteúdo da aula anterior. Por este motivo, verifique abaixo se você já concluiu os requisitos antes de começar. Para uma boa fluidez também é importante você seguir o mesmo padrão de nomes de variáveis, classes e funções utilizadas desde o início.

Requisitos

- 1) Aula 01 - Introdução a Frameworks Web e Django: <https://youtu.be/LpFNMn6Uw5s>
- 2) Aula 02 - Configurações do Ambiente: https://youtu.be/7zxJi3_HRuW

Nesta aula vamos criar nossa primeira Aplicação “Olá Django”, que será a base para todo o projeto. O objetivo é termos o primeiro contato com o Django e observar como ele funciona. Por este motivo vamos revisar / refazer rapidamente a preparação do ambiente virtual Python realizado na aula anterior, será o **Passo 0**. Caso já tenha realizado, pode seguir direto para o **Passo 1**. Este curso usará como tema um sistema de enquetes (polls) (perguntas e respostas). É o mesmo assunto utilizado na documentação oficial do Django para introdução aos principais recursos do Framework. A diferença é que vamos fazer uma nova abordagem focando em iniciantes e adicionando novos recursos.

Referência: <https://docs.djangoproject.com/pt-br/4.2/intro/tutorial01/>

Nesta aula:

- 1) Criar projeto e App Django
- 2) Habilitar o módulo polls no projeto webapp
- 3) Criar uma view para o app polls
- 4) Adicionar as respectivas rotas para as views
- 5) Inicializar o servidor de desenvolvimento para testes
- 6) Criando uma view que retorna um template HTML

Passo 0: Preparação do ambiente virtual Python

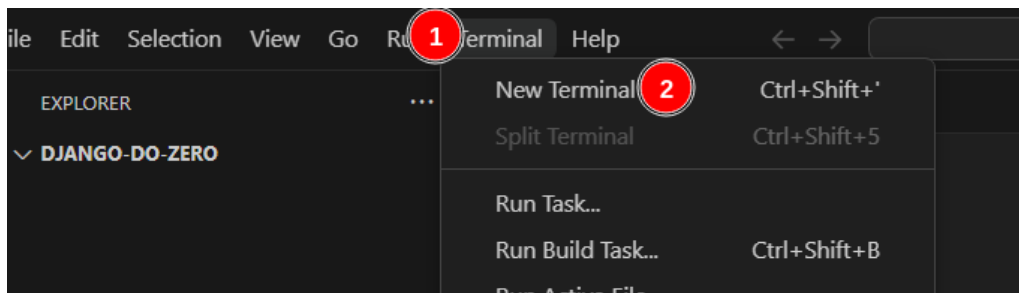
Esta configuração inicial já foi realizada na Aula 02. A ideia de colocar novamente a preparação do ambiente é apenas para referência e revisão rápida do que foi realizado.

Vamos criar um venv (virtual environment), ativar o venv, e instalar o Django no venv.

Abra o gerenciador de arquivos (Windows Explorer) e crie uma pasta chamada **django-do-zero** dentro de Documentos. A função desta pasta é abrigar todos os arquivos do projeto Django, além de deixarmos tudo padronizado para as aulas.

Abra a pasta **django-do-zero** no VS Code.

Agora abra um novo terminal pelo menu “Terminal” -> “New Terminal” (Command Prompt), e execute os comandos:



```
python -m venv venv (ENTER)
```

Se estiver no Windows:

```
.\venv\Scripts\activate.bat (ENTER)
```

Se estiver no Linux:

```
source venv/bin/activate (ENTER)
```

Uma pasta chamada **venv** será criada e gerenciada automaticamente. É nesta pasta que todos os pacotes e configurações do Python ficarão salvos. Observe que o venv precisa estar ativado o tempo todo e visualmente é fácil saber se ele está ativo ou não pela indicação **(venv)** no início da linha do prompt.

Com o venv ativado instale o django com o comando:

```
pip install django (ENTER)
```

Vamos complementar este passo iniciando a gestão das dependências do projeto salvando os requisitos atuais no arquivo requirements.txt (usado pelo pip / Python). No terminal execute:

```
pip freeze > requirements.txt (ENTER)
```

O arquivo requirements.txt serve para fazermos a gestão das dependências e deverá ser utilizado para instalar todos os pacotes que são dependências do projeto de uma única vez com o comando (“quando necessário”):

```
pip install -r requirements.txt (ENTER)
```

Passo 1: Criar projeto e App Django

Retorne ao terminal aberto no VS Code e execute (**existe um espaço e um ponto depois de webapp**) :

django-admin startproject webapp . (ENTER)

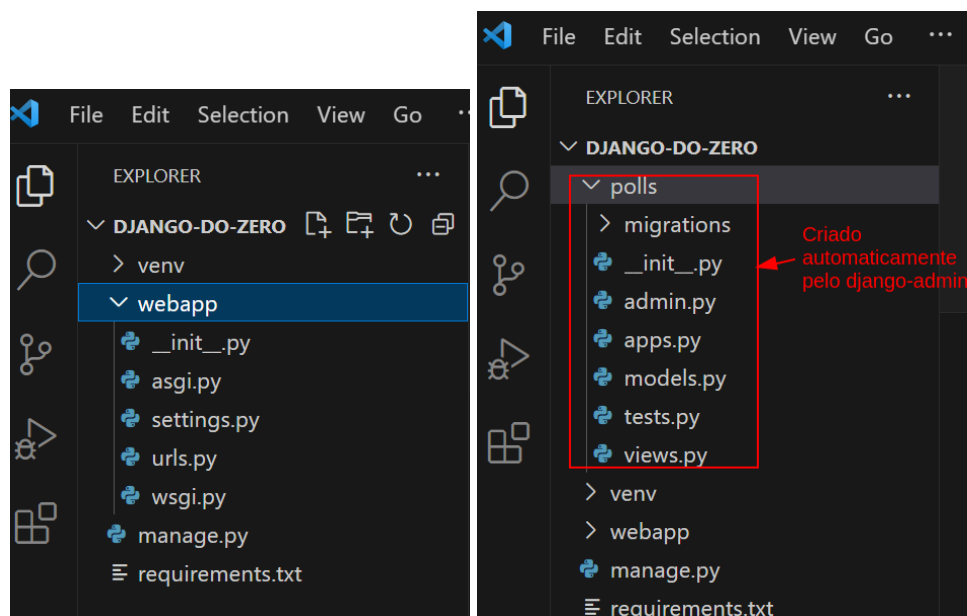
Se o comando acima for executado sem erros, uma estrutura básica de arquivos do projeto será criada dentro de uma pasta **webapp** e **gerará os arquivos básicos**.

Após a criação do projeto vamos criar e configurar a nossa primeira aplicação, que na prática será um módulo Python. Para criar o módulo execute o seguinte comando no terminal:

django-admin startapp polls (ENTER)

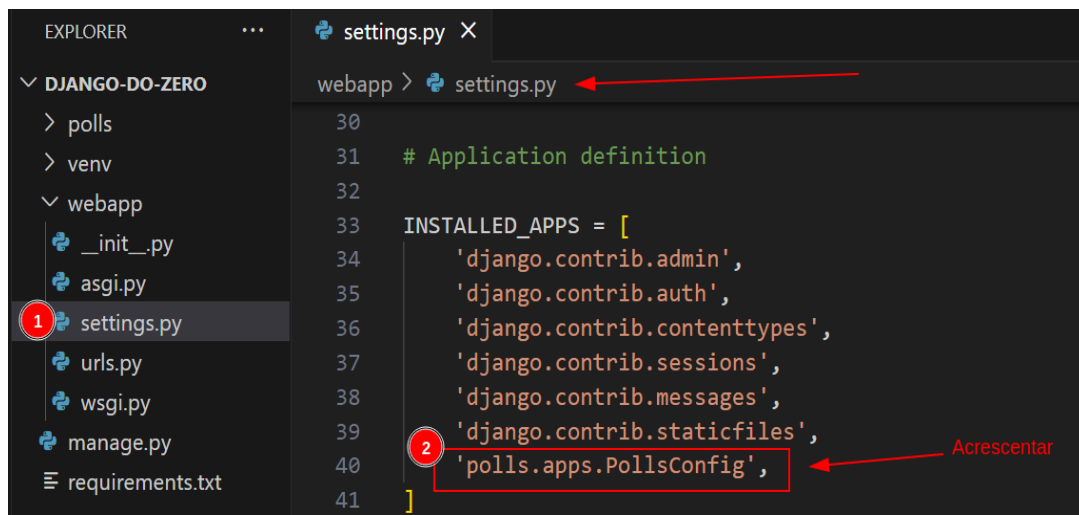
O comando acima criará uma nova pasta **polls**, contendo a estrutura básica de um app Django que também é um módulo Python.

Veja abaixo como ficará as estruturas das pastas **webapp** e **polls** (**ambas criadas automaticamente pelo django-admin**):



Passo 2: Habilitar o módulo polls no projeto webapp

Uma vez criado ou incluído um novo app (módulo) é necessário habilitar o mesmo nas configurações do projeto Django. Este procedimento é realizado editando o arquivo **webapp/settings.py**. Abra o arquivo em modo de edição, localize as definições de apps instalados identificado por **INSTALLED_APPS** e acrescente o app como indicado na imagem:



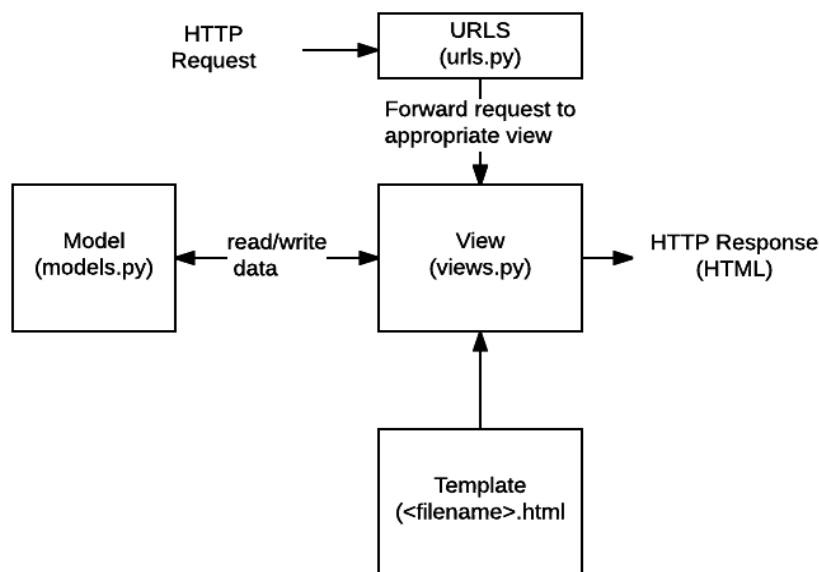
```
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'polls.apps.PollsConfig',
41 ]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

(salve o arquivo)

Esta configuração informa ao projeto que o app / módulo polls será utilizado e portanto será carregado ao inicializar o sistema.

Passo 3: Criar uma view para o app polls

Revisão do Fluxo de Requisição e Resposta



As views no Django são basicamente funções ou classes responsáveis por tratar / processar uma requisição e retornar uma resposta. As views são independentes em cada módulo e ficam dentro do arquivo **<pasta_módulo>/views.py**. No nosso caso em **polls/views.py** que é um arquivo básico gerado na criação do app (startapp). Vamos criar uma view simples baseada em função e sem template. Para isso, abra o arquivo **polls/views.py** em modo de edição complete o código da seguinte forma:

The screenshot shows the VS Code interface. On the left, the Explorer sidebar shows the project structure for 'DJANGO-DO-ZERO'. The 'polls' directory is expanded, and 'views.py' is selected, indicated by a red circle with the number '2'. On the right, the editor shows the content of 'views.py'. It starts with a red circle and the number '1' next to the 'polls' directory in the Explorer. The code in 'views.py' includes imports for 'render' from 'django.shortcuts' and 'HttpResponse' from 'django.http'. It defines two functions: 'index(request)' which returns 'Olá Django - index', and 'ola(request)' which returns 'Olá Django'.

(salve o arquivo)

Com as funções **index()** e **ola()** criadas agora precisamos configurar as respectivas rotas para elas, desta forma o projeto quando ativo saberá quando executar estas views.

Passo 4: Adicionar as respectivas rotas para as views

O mecanismo de rotas é o que ajuda os sistemas web a decidir o que fazer quando chega uma determinada requisição (acesso a determinada URL). Nos projetos Django existe um arquivo padrão em <pasta_projeto>/urls.py que configura as **rotas principais**. E as rotas do App devem ficar em seu próprio arquivo **urls.py** dentro da respectiva pasta. No nosso caso será o arquivo **polls/urls.py**. Este arquivo ainda não existe, mas será o responsável pelas configurações de todas as rotas do módulo polls. Esta estrutura é a mais indicada pois mantém as rotas do app polls independentes, facilitando a manutenção e portabilidade do módulo para outro projeto se necessário.

Vamos **criar** o novo arquivo **polls/urls.py** contendo o conteúdo conforme indicado na imagem:

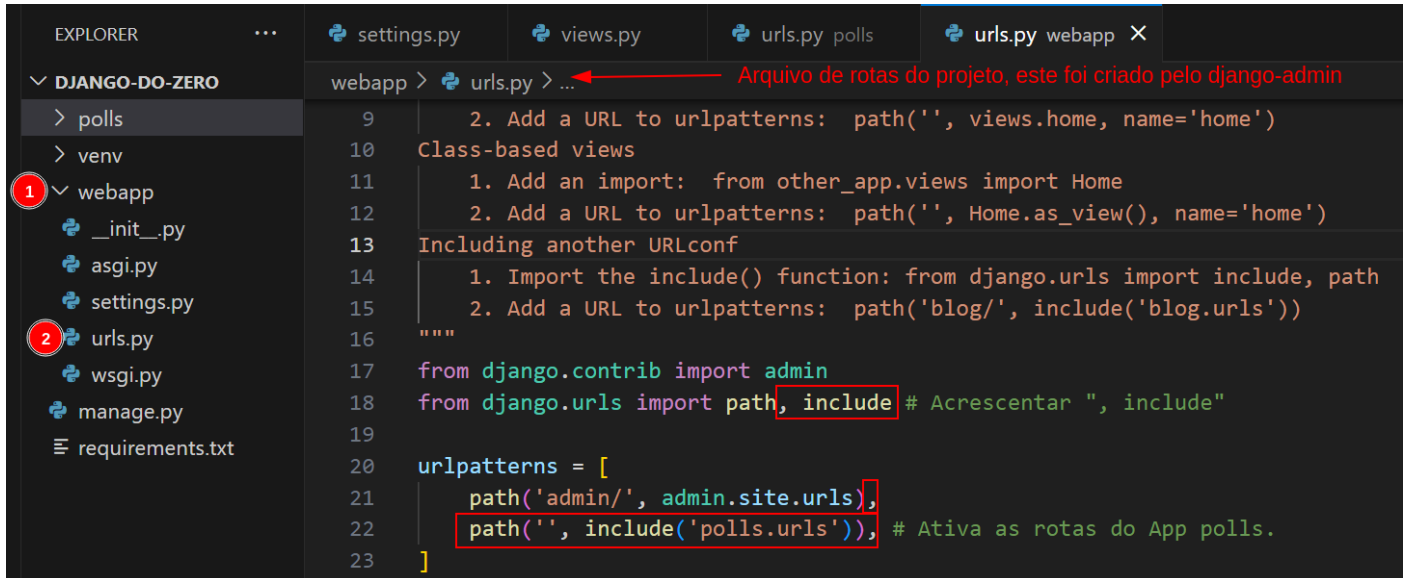
The screenshot shows the VS Code interface. On the left, the Explorer sidebar shows the project structure for 'DJANGO-DO-ZERO'. The 'polls' directory is expanded, and 'urls.py' is selected, indicated by a red circle with the number '2'. On the right, the editor shows the content of 'urls.py'. It starts with a red circle and the number '1' next to the 'polls' directory in the Explorer. The code in 'urls.py' includes imports for 'path' from 'django.urls' and 'index, ola' from 'polls.views'. It defines a list 'urlpatterns' containing two path entries: 'index' and 'ola'.

(salve o arquivo)

Na configuração acima importamos nossas views diretamente do arquivo **polls/views.py** e criamos duas rotas indicando o caminho, função responsável por produzir a resposta e o identificador interno da rota (**name**).

O arquivo de rotas principais do projeto localizado em **webapp/urls.py** contém a configuração básica para todas as rotas do sistema.

Para concluir a configuração das rotas abra o arquivo **webapp/urls.py** no modo edição e altere para que fique conforme indicado na imagem. Nesta alteração vamos incluir no arquivo de rotas principal a informação de onde serão carregadas as rotas para o módulo polls.



```
9      2. Add a URL to urlpatterns: path('', views.home, name='home')
10      Class-based views
11      1. Add an import: from other_app.views import Home
12      2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13      Including another URLconf
14      1. Import the include() function: from django.urls import include, path
15      2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16      """
17      from django.contrib import admin
18      from django.urls import path, include # Acrescentar ", include"
19
20      urlpatterns = [
21          path('admin/', admin.site.urls),
22          path('', include('polls.urls')), # Ativa as rotas do App polls.
23      ]
```

(salve o arquivo)

Feito isso, o próximo passo é executar servidor em ambiente de desenvolvimento para testar o funcionamento desta rotas que devem produzir as respostas programadas pelas **views index() e ola()**.

Passo 5: Inicializar o servidor de desenvolvimento para testes

Abre o terminal, caso não esteja aberto e execute o seguinte comando:

python manage.py runserver (ENTER)

O terminal fica ocupado (preso) e pode ser liberado pressionando CTRL + C

Veja mais informações sobre a execução do comando na próxima imagem.

```
(venv) C:\Users\Vamos Codar\Documents\django-do-zero>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
```

NAO SE PREOCUPE COM ESTA MENSAGEM

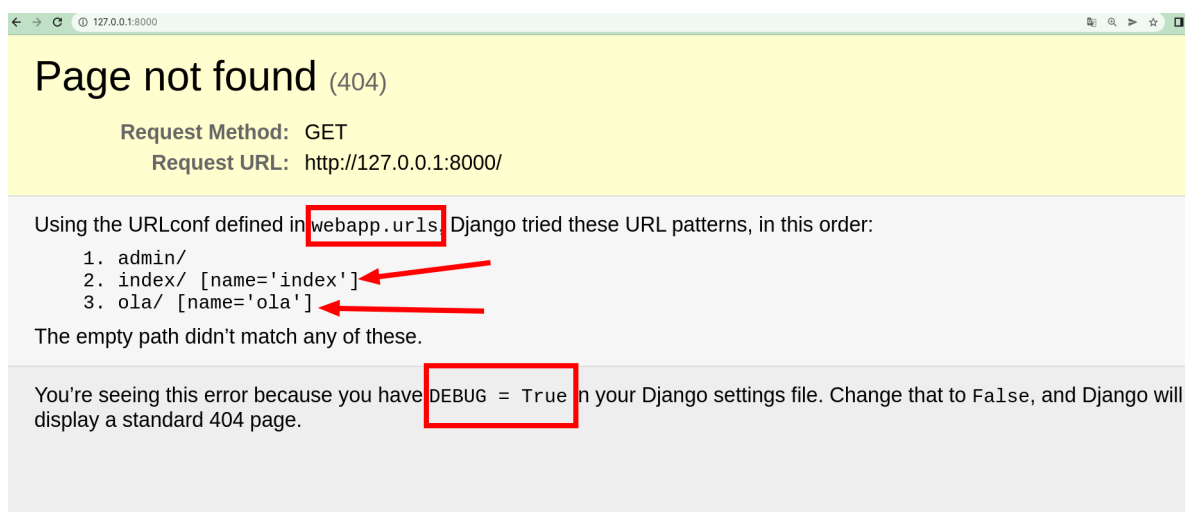
```
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s):
  admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
September 25, 2023 - 17:21:25
Django version 4.2.5, using settings 'webapp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

O servidor de desenvolvimento fica ativo neste IP : PORTA basta acessar no navegador web

Pressionar CTRL + C para o servidor

Se não ocorrer nenhum erro fatal o servidor de desenvolvimento estará ativo e rodando localmente no endereço localhost (127.0.0.1) e na porta 8000.

Abra um navegador e digite na barra de endereços 127.0.0.1:8000 (ENTER)



Deverá ser carregada a “página de erro” acima, pois o projeto inicial fica com o modo **DEBUG** ativado. O erro indica que nenhuma rota padrão (“/” ou “ ”) foi encontrada e faz com que liste todas as rotas disponíveis.

Experimente acessar as rotas configuradas:

127.0.0.1:8000/index

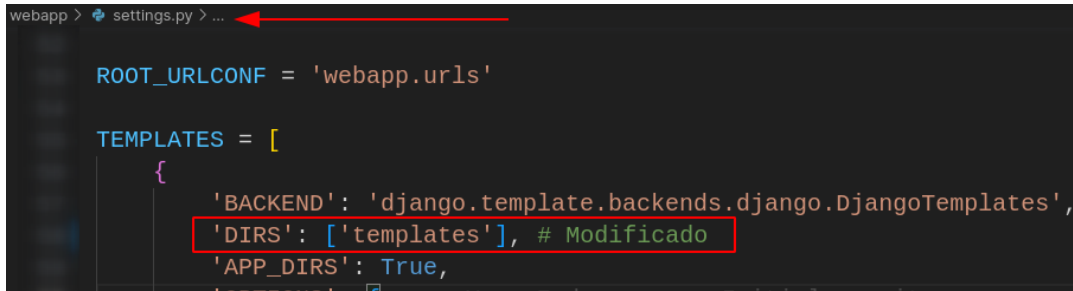
127.0.0.1:8000/ola

Estas rotas devem carregar a resposta configurada nas views **index()** e **ola()** respectivamente.

Passo 6: Criando uma view que retorna um template HTML

Para criar uma view que processa um template precisamos configurar o projeto para trabalhar com templates. Crie uma pasta com nome **templates** (na raiz da pasta do projeto **django-do-zero/**). Depois edite o arquivo de configurações do projeto localizado em **webapp/settings.py** conforme indicado na próxima imagem.

webapp/settings.py:



```
webapp > settings.py > ...  
  
ROOT_URLCONF = 'webapp.urls'  
  
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
        'DIRS': ['templates'], # Modificado  
        'APP_DIRS': True,  
    },  
]
```

(salve o arquivo)

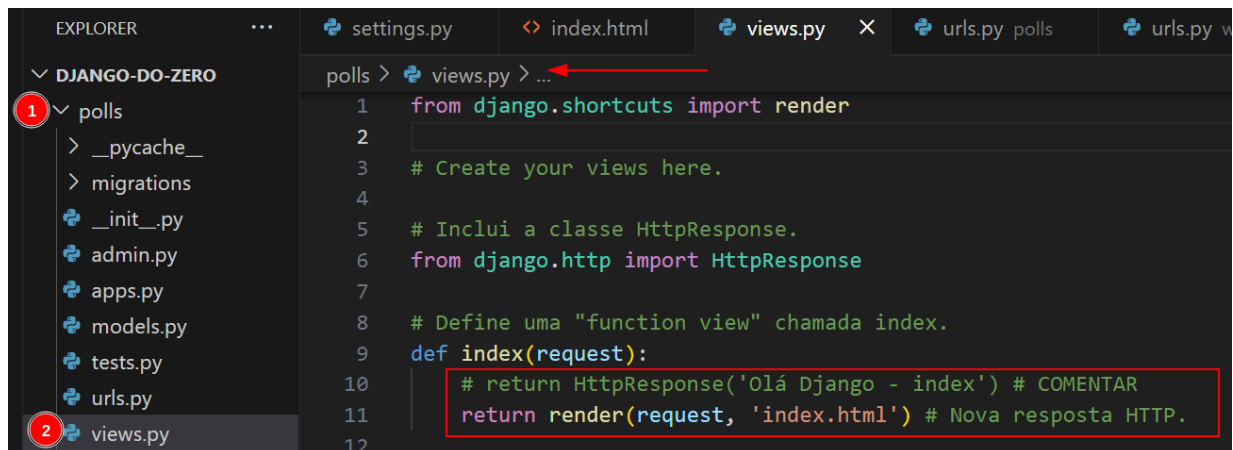
Agora vamos criar um arquivo HTML mínimo para ser o template de **resposta** para a nossa **view index()**. Dentro da pasta **templates** (criada anteriormente) crie um novo arquivo chamado **index.html** (use o modelo que desejar ou a referência abaixo).

django-do-zero/templates/index.html:

```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <title>Minhas Enquetes</title>  
  </head>  
  <body>  
    <main>  
      <h1>Bem vindo ao Sistema de Enquetes</h1>  
    </main>  
  </body>  
</html>
```

Para concluir a configuração vamos modificar a **view index()** para processar o template e devolvê-lo como resposta. Siga conforme a próxima imagem:

polls/views.py:



```
polls > views.py > ...  
1  from django.shortcuts import render  
2  
3  # Create your views here.  
4  
5  # Inclui a classe HttpResponse.  
6  from django.http import HttpResponse  
7  
8  # Define uma "function view" chamada index.  
9  def index(request):  
10     # return HttpResponse('Olá Django - index') # COMENTAR  
11     return render(request, 'index.html') # Nova resposta HTTP.  
12
```

(salve o arquivo)

Para testar carregue no navegador a respectiva rota para a view **index()**:

127.0.0.1:8000/index

Observação: O servidor do ambiente de desenvolvimento deve estar em execução.

python manage.py runserver (ENTER)

Retorne ao **Passo 5** e revise as informações, se tiver dificuldade para executar o servidor.

Conclusão

Após a execução dos passos deste material podemos compreender a arquitetura básica de um sistema web criado com o Framework Django. Nas próximas aulas vamos aprofundar o uso e configuração dos sistemas de template para incluir variáveis dinâmicas e outros objetos. E na sequência continuaremos estudando outros recursos deste poderoso Framework que habilita o programador a desenvolver rapidamente diversos tipos de sistemas.