

# BackEnd

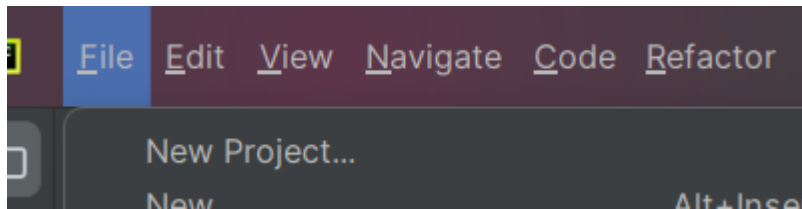
## Django

Django é um framework web Python de alto nível que permite o rápido desenvolvimento de sites seguros e de fácil manutenção. Construído por desenvolvedores experientes, o Django cuida de grande parte do trabalho de desenvolvimento web, para que você possa se concentrar em escrever seu aplicativo sem precisar reinventar a roda.

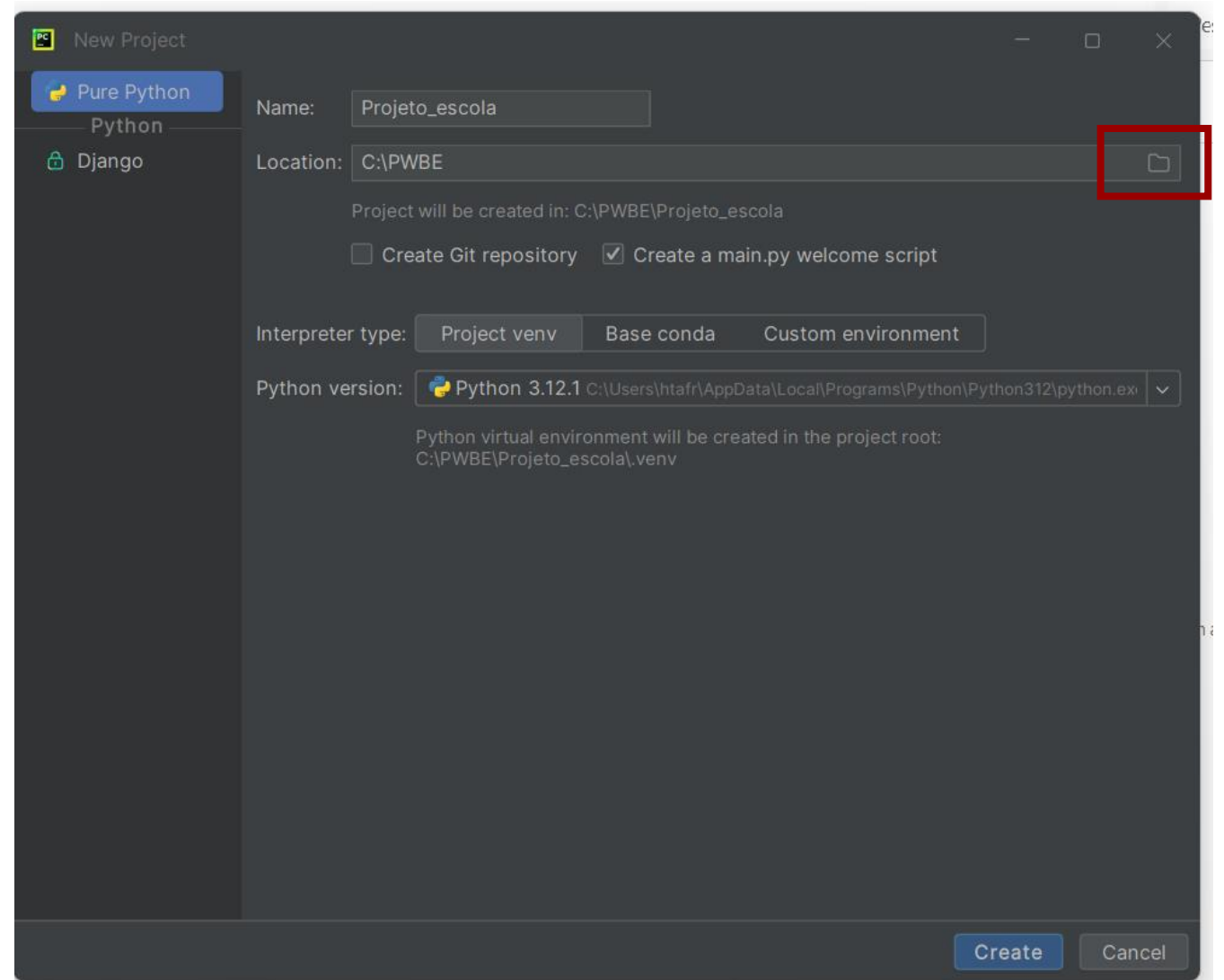
Django ajuda os desenvolvedores a evitar os erros de segurança mais comuns, fornecendo um framework que foi desenhado para "fazer as coisas certas", de modo a proteger o website automaticamente. Por exemplo, Django fornece uma maneira segura de gerenciar as contas dos usuários e suas senhas, evitando erros comuns, tais como colocar informações da sessão em cookies, onde ficam vulneráveis (ao invés disso os cookies contêm apenas uma chave e os dados são armazenados no banco de dados), ou armazenar as senhas de forma direta, ao invés de gravar um hash para essas senhas.

# BackEnd

1 Passo será criar um novo projeto (File -> New Project)

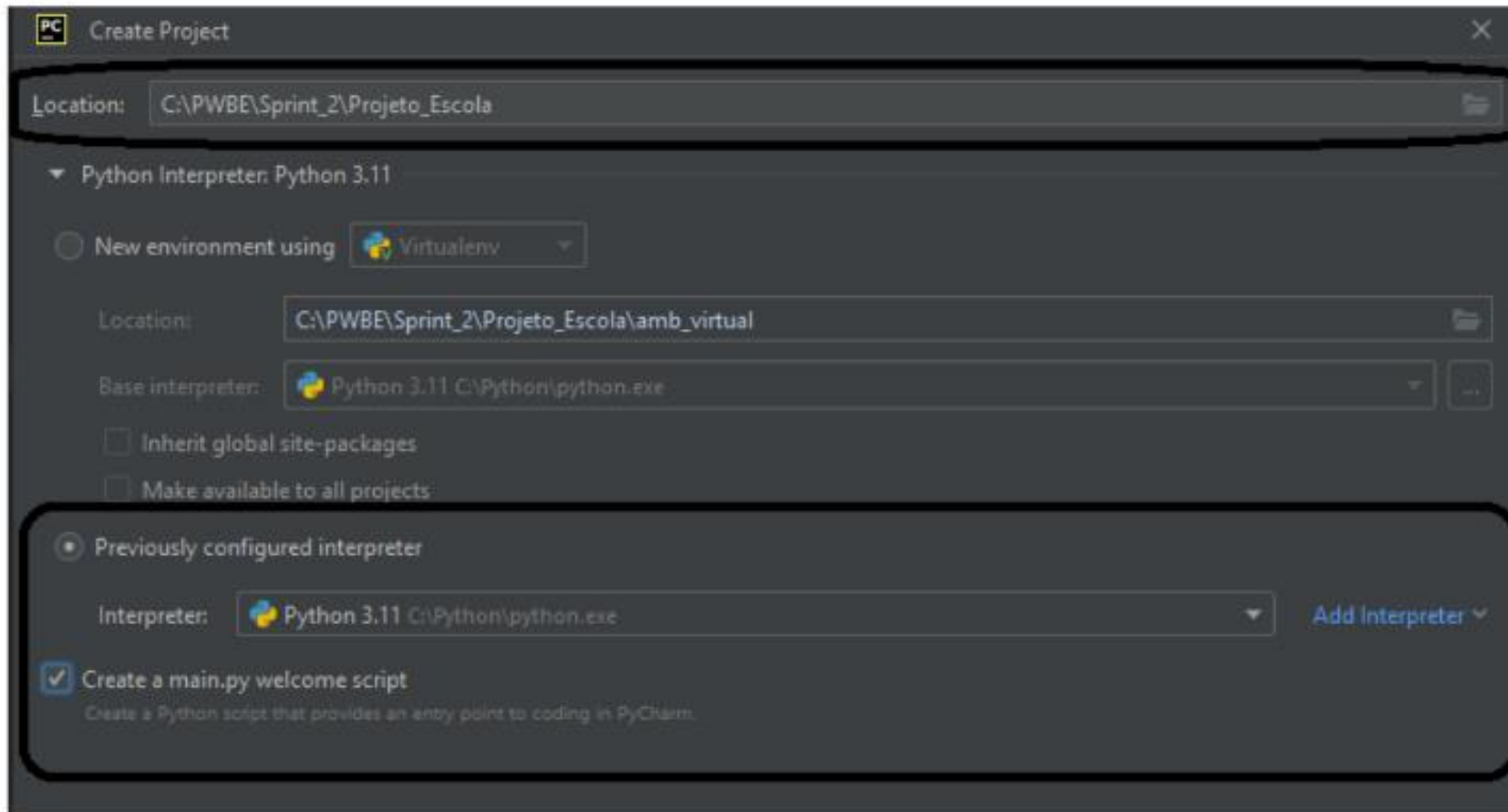


Em location, é possível selecionar a pasta onde queremos armazenar nosso projeto



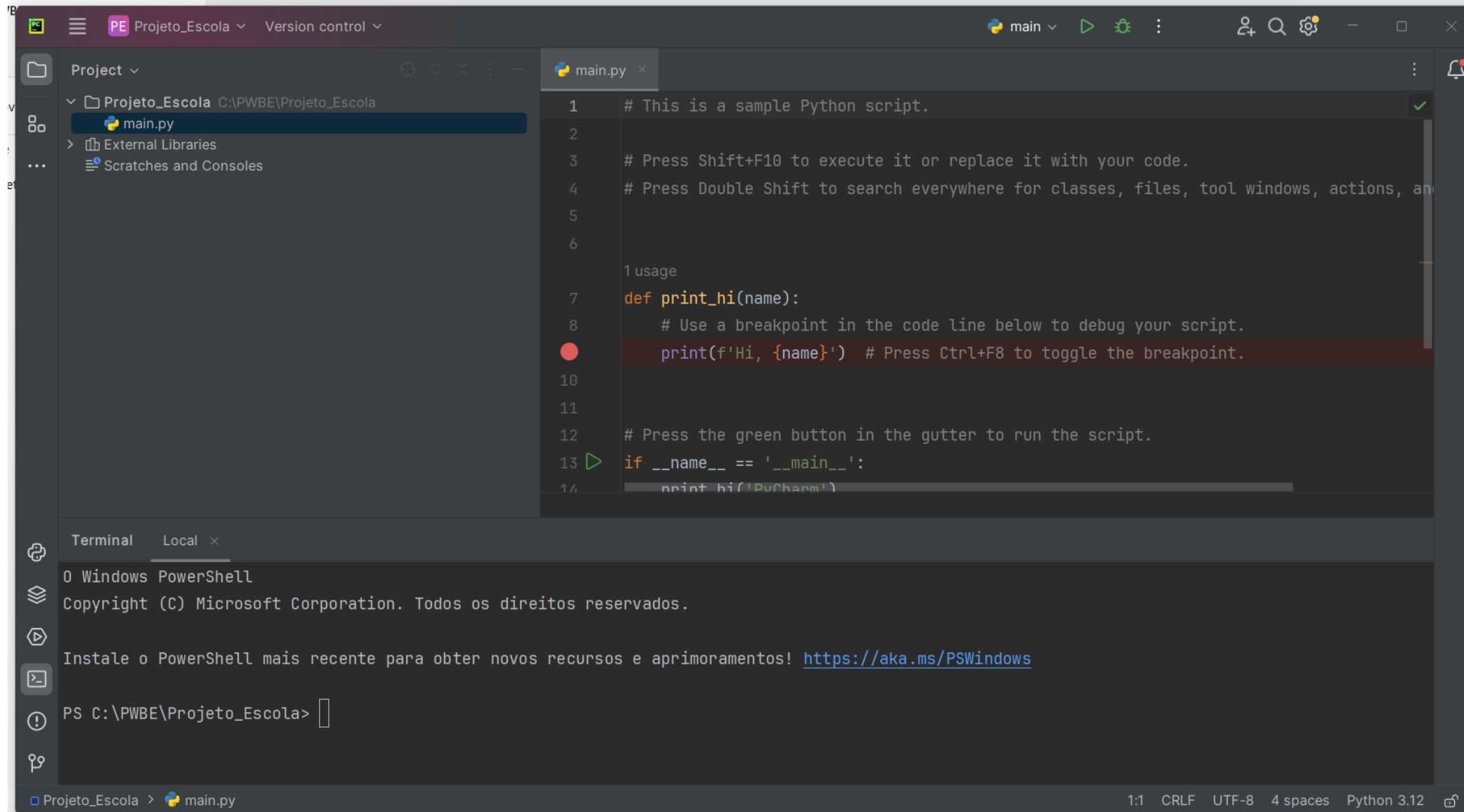
# BackEnd

Atente para que a opção Previously configured interpreter esteja selecionado. Isto permitirá que façamos a criação de um novo ambiente virtual e de toda as instalações necessárias das bibliotecas para criação do Projeto\_Escola a partir do início( a partir do zero).



# BackEnd

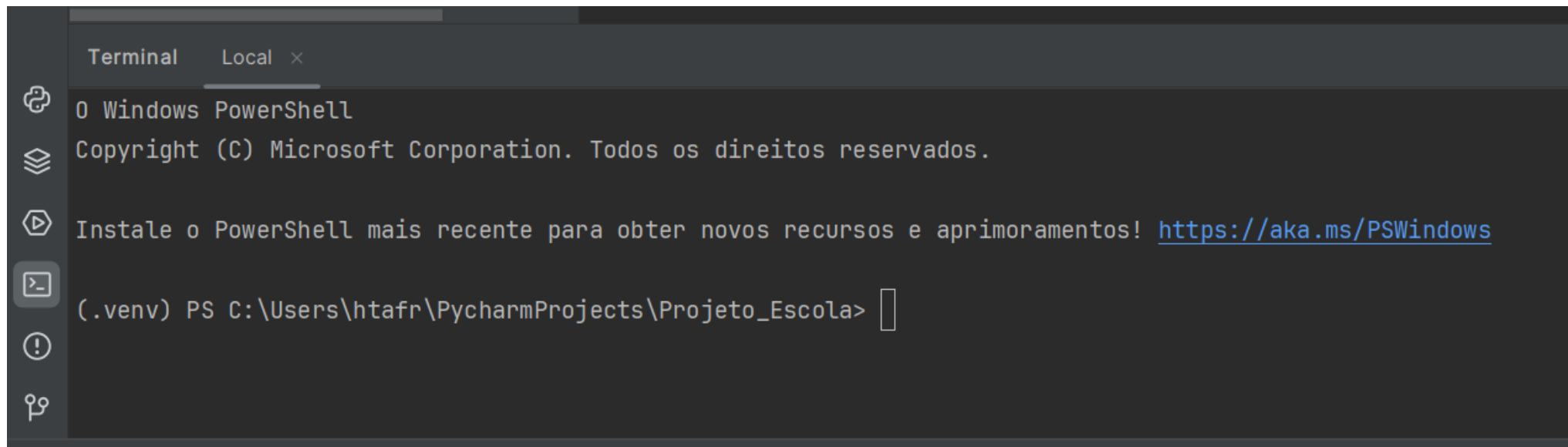
Selecione o “Create”, vai aparecer uma tela:



# BackEnd

Vamos criar um novo ambiente virtual onde acomodaremos todas as instalações de bibliotecas que utilizaremos.

Para isso vá para o ambiente do Terminal clicando em Terminal na parte inferior da tela conforme a figura que segue:



```
Terminal Local x
0 Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Instale o PowerShell mais recente para obter novos recursos e aprimoramentos! https://aka.ms/PSWindows

(.venv) PS C:\Users\htafr\PycharmProjects\Projeto_Escola>
```

# BackEnd

Para criar o ambiente virtual é necessário que a biblioteca virtualenv esteja instalada no seu computador. Para isso, execute o comando como segue:

```
PS C:\PWBE\Projeto_Escola> pip install virtualenv
```

Depois da execução é exibida a mensagem de sucesso

```
Downloading filelock-3.13.3-py3-none-any.whl (11 kB)  
Using cached platformdirs-4.2.0-py3-none-any.whl (17 kB)  
Installing collected packages: distlib, platformdirs, filelock, virtualenv  
Successfully installed distlib-0.3.8 filelock-3.13.3 platformdirs-4.2.0 virtualenv-20.25.1
```

# BackEnd

Para verificar se tudo foi instalado corretamente, podemos executar o comando **pip freeze**

```
[notice] A new release of pip is available: 23.2.1 -> 24.0
Python Packages date, run: python.exe -m pip install --upgrade pip
(.venv) PS C:\Users\htafr\PycharmProjects\Projeto_Escola> pip freeze
distlib==0.3.8
filelock==3.13.3
platformdirs==4.2.0
virtualenv==20.25.1
(.venv) PS C:\Users\htafr\PycharmProjects\Projeto_Escola>
```

E para limpar todas as linhas de comando, podemos executar **cls**

E então vamos criar nosso ambiente virtual, fazendo:

```
PS C:\PWBE\Projeto_Escola> virtualenv amb_virtual
```



# BackEnd

Se o ambiente foi criado com sucesso deve ser exibida uma mensagem no terminal parecida com:

```
Werkzeug==3.0.1
PS C:\PWBE\Projeto_Escola> virtualenv amb_virtual
created virtual environment CPython3.12.1.final.0-64 in 7326ms
  creator CPython3Windows(dest=C:\PWBE\Projeto_Escola\amb_virtual, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, via=copy, app_data_dir=C:\Users\htafre\AppData\Local\pypa\virtualenv)
    added seed packages: pip==24.0
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
PS C:\PWBE\Projeto_Escola>
```

E em seguida vamos ativar esse ambiente virtual

```
PS C:\PWBE\Projeto_Escola> amb_virtual\Scripts\activate
(amb_virtual) PS C:\PWBE\Projeto_Escola>
```

Podendo agora fazer as demais instalações, como a do próprio Django

```
(amb_virtual) PS C:\PWBE\Projeto_Escola> pip install django
```

# BackEnd

Podendo agora fazer as demais instalações, como a do próprio Django

```
Downloading tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)
Downloading Django-5.0.3-py3-none-any.whl (8.2 MB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.2/8.2 MB 11.6 MB/s eta 0:00:00
Downloading asgiref-3.8.1-py3-none-any.whl (23 kB)
Downloading sqlparse-0.4.4-py3-none-any.whl (41 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41.2/41.2 kB ? eta 0:00:00
Downloading tzdata-2024.1-py2.py3-none-any.whl (345 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 345.4/345.4 kB 22.3 MB/s eta 0:00:00
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.8.1 django-5.0.3 sqlparse-0.4.4 tzdata-2024.1
(amb_virtual) PS C:\Users\htafr\PycharmProjects\Projeto_Escola>
```

```
(amb_virtual) PS C:\Users\htafr\PycharmProjects\Projeto_Escola> pip freeze
asgiref==3.8.1
Django==5.0.3
sqlparse==0.4.4
tzdata==2024.1
(amb_virtual) PS C:\Users\htafr\PycharmProjects\Projeto_Escola>
```

# BackEnd

Com o Django devidamente instalado no nosso ambiente virtual, vamos criar o nosso projeto no Django que vai se chamar Escola.

Mas antes, novamente, execute o comando cls para limpar a tela. Depois digite o comando para criar o nosso projeto no Django como segue:

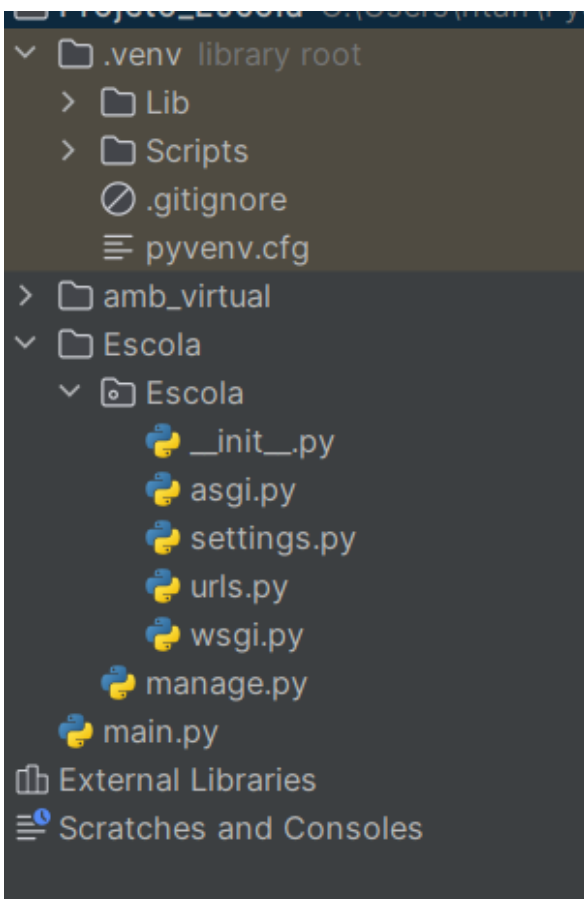
```
(amb_virtual) PS C:\PWBE\Projeto_Escola> django-admin startproject Escola  
(amb_virtual) PS C:\PWBE\Projeto_Escola> 
```

Quando criamos o projeto Escola, o Django criou uma nova pasta de mesmo nome ficando assim a estrutura.

Criando o projeto, o Django também criou o arquivo manage.py e será muitas vezes invocado daqui em diante. Portanto sempre que formos invocar o arquivo manage.py em algum comando, deveremos estar na pasta Escola. Para isso vamos mover uma pasta a frente digitando o comando como segue:

cd Escola

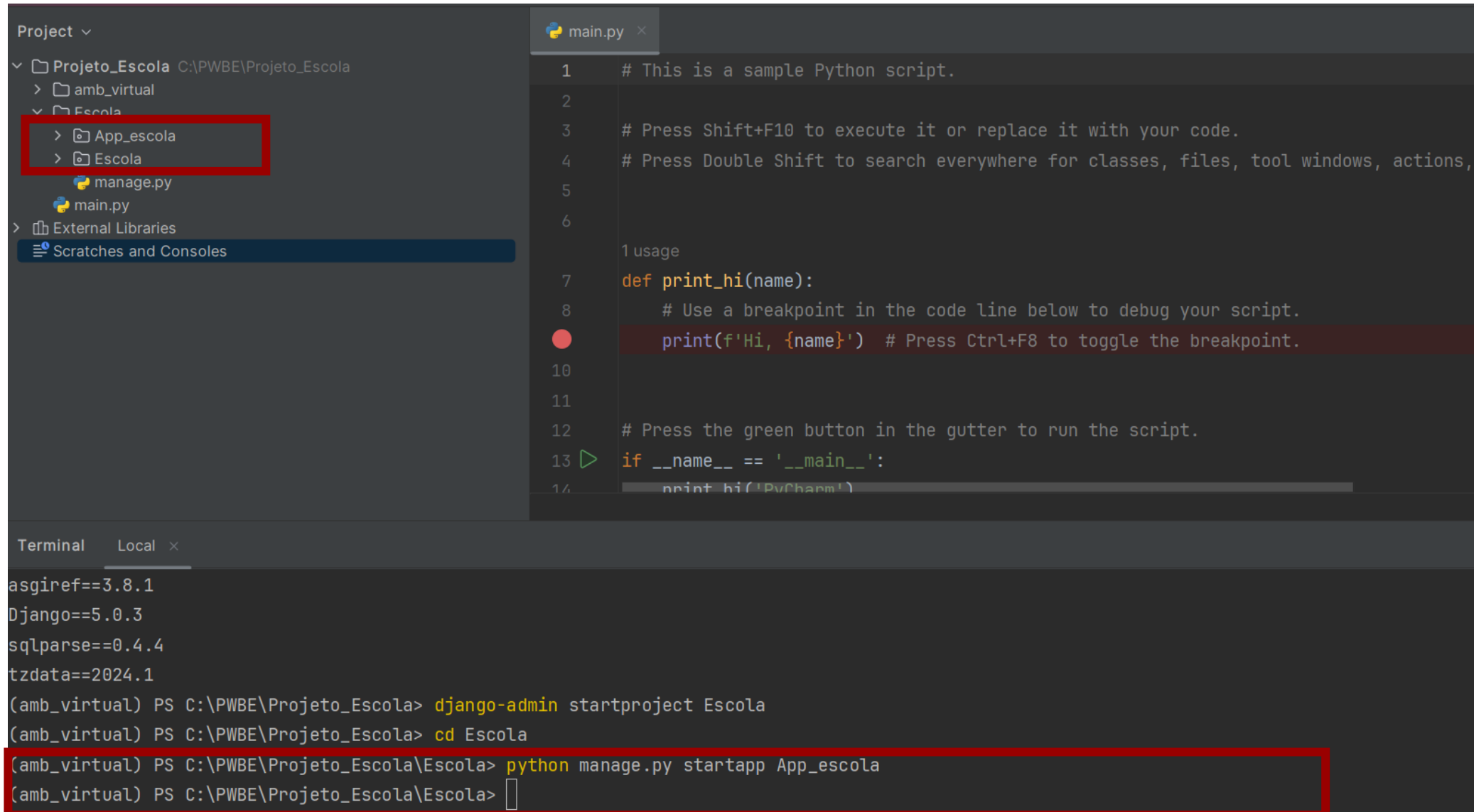
```
(amb_virtual) PS C:\PWBE\Projeto_Escola> cd Escola  
(amb_virtual) PS C:\PWBE\Projeto_Escola\Escola> 
```



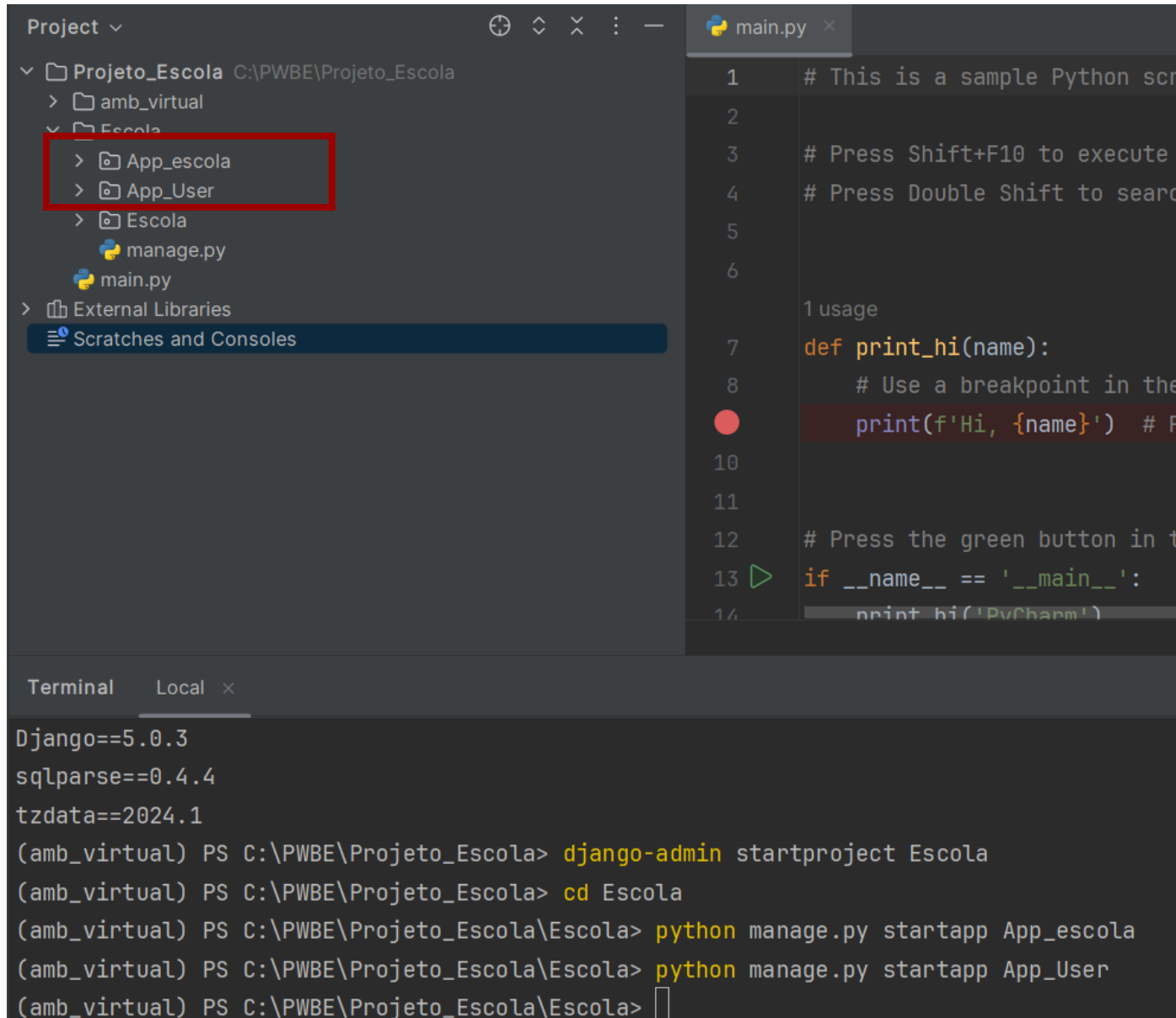
# BackEnd

Um projeto Django é composto por um ou mais App's.

```
python manage.py startapp App_Escola
```



# BackEnd



The screenshot shows an IDE with a project named 'Projeto\_Escola' at the path 'C:\PWBE\Projeto\_Escola'. The project structure in the left sidebar is as follows:

- Projeto\_Escola
  - amb\_virtual
  - Escola
    - App\_escola
    - App\_User
    - Escola
  - manage.py
  - main.py
- External Libraries
- Scratches and Consoles

The 'App\_escola' and 'App\_User' folders are highlighted with a red box. The main editor shows a Python file 'main.py' with the following code:

```
1 # This is a sample Python script
2
3 # Press Shift+F10 to execute the script
4 # Press Double Shift to search everywhere for packages, classes,
5 # modules, etc.
6
7 1 usage
8
9 def print_hi(name):
10     # Use a breakpoint in the following line of code
11     # Press F9 to toggle a breakpoint
12     print(f'Hi, {name}') # P
13
14 # Press the green button in the gutter to run the script
15
16 if __name__ == '__main__':
17     print_hi('PyCharm')
```

The terminal at the bottom shows the following commands and output:

```
Django==5.0.3
sqlparse==0.4.4
tzdata==2024.1
(amb_virtual) PS C:\PWBE\Projeto_Escola> django-admin startproject Escola
(amb_virtual) PS C:\PWBE\Projeto_Escola> cd Escola
(amb_virtual) PS C:\PWBE\Projeto_Escola\Escola> python manage.py startapp App_escola
(amb_virtual) PS C:\PWBE\Projeto_Escola\Escola> python manage.py startapp App_User
(amb_virtual) PS C:\PWBE\Projeto_Escola\Escola>
```

Vamos deixar criado um segundo app para o nosso projeto Escola que será o app responsável por gerenciar os acessos de usuários ao sistema.

Digite então o comando como segue:  
(amb\_virtual) PS  
C:\PWBE\Sprint\_2\Projeto\_Escola\Escola>  
**python manage.py startapp App\_User**

Após a criação dos dois app's do projeto, a estrutura de pastas deve ficar como segue:

# BackEnd

Para ver se o Django está funcionando corretamente, executamos o servidor local, através do comando runserver:

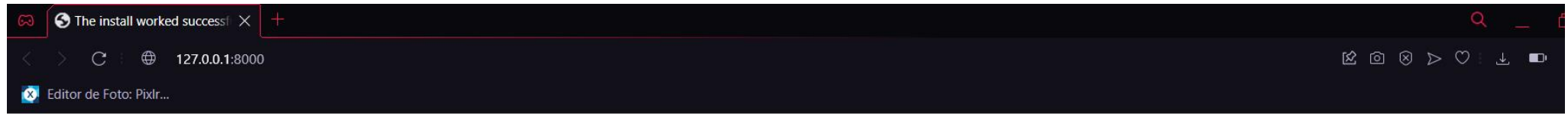
```
(amb_virtual) PS C:\PWBE\Projeto_Escola\Escola> python manage.py runserver
```

Se o servidor local conseguiu ser executado, no Terminal deve ter exibido a seguinte mensagem:

```
ssions.  
Run 'python manage.py migrate' to apply them.  
March 31, 2024 - 06:30:15  
Django version 5.0.3, using settings 'Escola.settings'  
Starting development server at http://127.0.0.1:8000/  
Quit the server with CTRL-BREAK.
```



# BackEnd



django

View [release notes](#) for Django 5.0



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.



**Django Documentation**

Topics, references, & how-to's



**Tutorial: A Polling App**

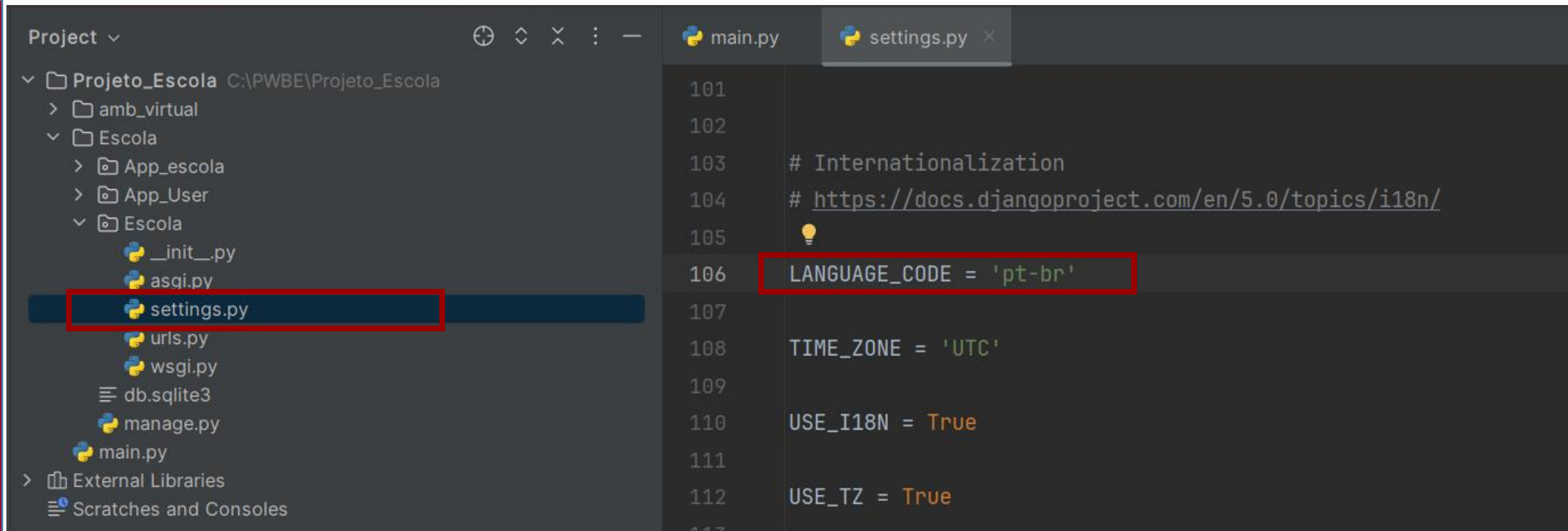
Get started with Django



**Django Community**

Connect, get help, or contribute

# BackEnd



Dê um duplo clique sobre o nome do arquivo settings.py e ele será aberto para edição.

Este arquivo comporta uma série de configurações relacionadas ao seu projeto DJANGO.

Vá até a sessão onde aparece a variável **LANGUAGE\_CODE = 'en-us'** Altere para **LANGUAGE\_CODE = 'pt-br'**

Acesse novamente o Browser no endereço <http://127.0.0.1:8000/>, aperte a tecla F5 para atualizar e a imagem que segue deve ser mostrada:



# BackEnd



django

Ver as [notas de lançamento](#) do Django 5.0



A instalação foi com sucesso! Parabéns!

Você está vendo esta página pois possui `DEBUG=True` no seu arquivo de configurações e não configurou nenhuma URL.



**Documentação do Django**

Tópicos, referências, & how-to's



**Tutorial: Um aplicativo de votação**

Comece a usar Django



**Comunidade Django**

Conecte-se, obtenha ajuda ou contribua

# BackEnd

Este arquivo settings.py comporta uma série de configurações sobre o seu projeto DJANGO. Neste mesmo arquivo devemos associar os app's criados ao projeto Escola. Neste caso, associar o App\_Escola e App\_User ao projeto Escola.

Para isso, no mesmo arquivo settings.py localize a sessão INSTALLED\_APPS e acrescente o 'App\_Escola' e 'App\_User', como segue.

Atente para colocação entre apóstrofe e vírgula ao final na linha.

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

# BackEnd

O comando migrate é responsável no Django por criar e alterar as estruturas da Banco de Dados (BD) da aplicação. Quando instalamos o Django neste ambiente e criamos o projeto e os app's, ficaram pendentes de serem criadas as tabelas de BD utilizadas pelo Django.

Para isso é necessário executar este comando pela primeira vez após a instalação do Django e assim termos acesso ao ambiente administrativo do Django.

Assim, voltamos ao ambiente do Terminal para executar o comando. Se o servidor local ainda estiver em execução, clique na área do terminal para que o cursor fique selecionado e digite <ctrl> + 'C', ou seja, tecla control + a tecla C simultaneamente para que seja interrompido a execução do servidor e retorne a linha de comando do Terminal devendo ficar como mostrado na figura que segue:

```
(amb_virtual) PS C:\PWBE\Projeto_Escola\Escola> 
```

```
(amb_virtual) PS C:\PWBE\Projeto_Escola\Escola> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
```

# BackEnd

## Criando Super Usuário para o Projeto.

Executar o comando migrate no passo anterior fez com que o Banco de Dados usado pelo Administrador do projeto Django fosse criado. Porém para que possamos acessá-lo é necessário a criação de um super usuário ainda pela linha de comando do Terminal.

Vamos então criar este super usuário digitando o comando createsuperuser como segue:

(amb\_virtual) PS C:\PWBE\Sprint\_2\Projeto\_Escola\Escola> **python manage.py createsuperuser**

Preencha os campos conforme solicitado até que a mensagem “Superuser created successfully” apareça conforme mostrado na figura que segue.

```
Usuário (leave blank to use 'htafr'): nandafretes
Endereço de email: nandafretes@gmail.com
Password:
Password (again):
Superuser created successfully.
(amb_virtual) PS C:\PWBE\Projeto_Escola\Escola> 
```

# BackEnd

## Acessando o ambiente Admin do seu projeto Django.

Agora com o super usuário criado, já podemos acessar o ambiente Admin (ambiente de administração) do seu Projeto Django.

Execute novamente o servidor local como segue:

(amb\_virtual) PS C:\PWBE\Sprint\_2\Projeto\_Escola\Escola>**python manage.py runserver**

Agora abriremos o Browser, aqui estou usando o Google Chrome, no endereço indicado

**<http://127.0.0.1:8000/admin>** .

```
(amb_virtual) PS C:\PWBE\Projeto_Escola\Escola> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
March 31, 2024 - 07:21:58
Django version 5.0.3, using settings 'Escola.settings'
Starting development server at http://127.0.0.1:8000/
```

# BackEnd

127.0.0.1:8000/admin/login/

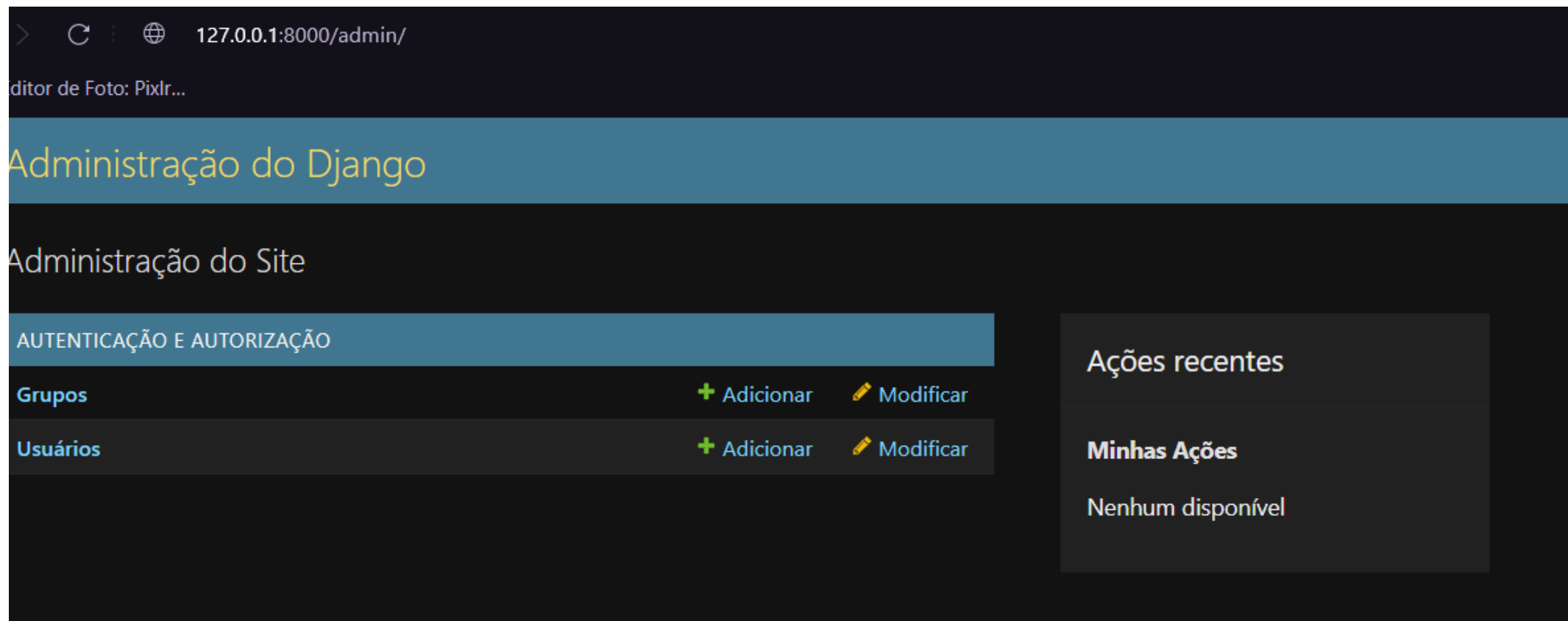
Administração do Django

Usuário:

Senha:

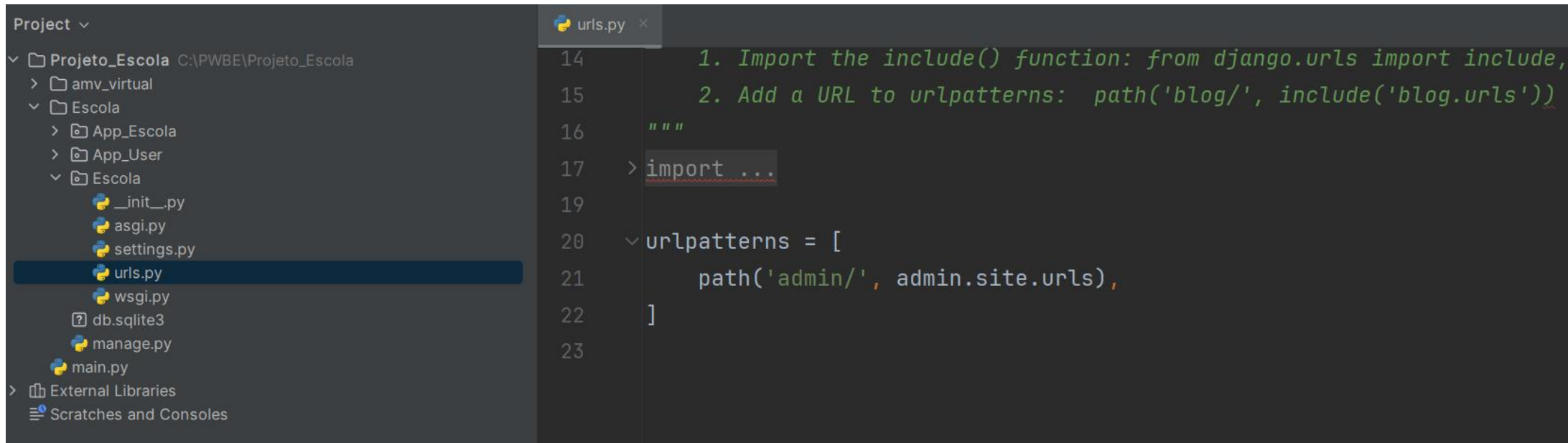
Acessar

# BackEnd



# BackEnd

O caminho admin onde acessamos o administrador do projeto no Django está configurado na sessão urlpatterns.



The screenshot shows the PyCharm IDE interface. On the left, the 'Project' view displays the file structure of 'Projeto\_Escola'. The 'Escola' app directory is expanded, showing files like \_\_init\_\_.py, asgi.py, settings.py, urls.py (highlighted), wsgi.py, db.sqlite3, manage.py, and main.py. On the right, the 'urls.py' file is open, showing the following code:

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

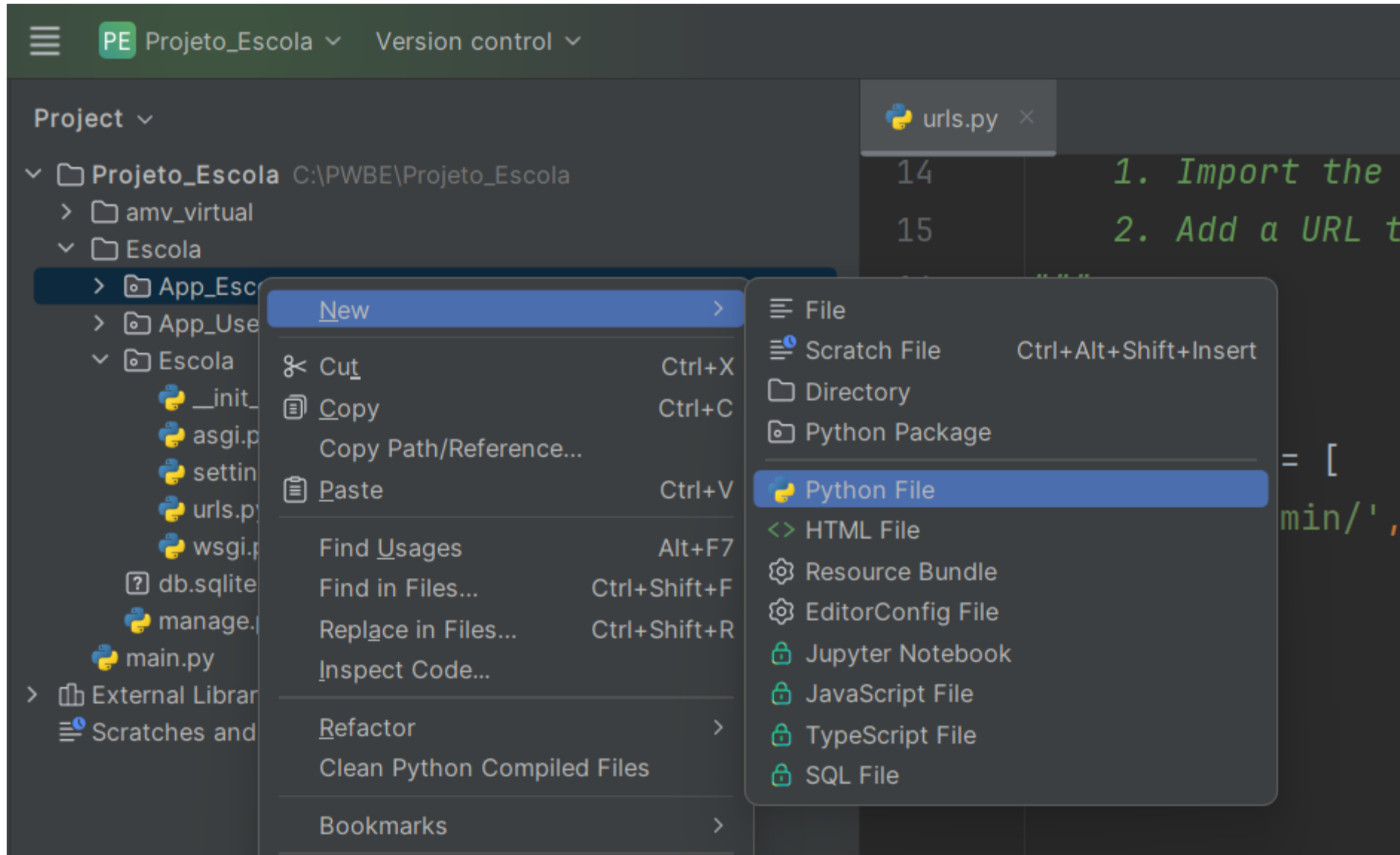
Nesta mesma sessão podemos criar outros 'caminhos' (rotas) para acesso do projeto.

Uma boa prática em projetos Django é termos um arquivo urls.py vinculado aos app's que o projeto contém. Assim sendo criaremos um arquivo urls.py para o App Escola onde criaremos os caminhos (rotas) para funcionamento do App\_Escola.

A partir do PyCharm, clique sobre a pasta **App\_Escola**, clique botão direito do mouse > **New** > **Python File** e na caixa que aparece digite urls



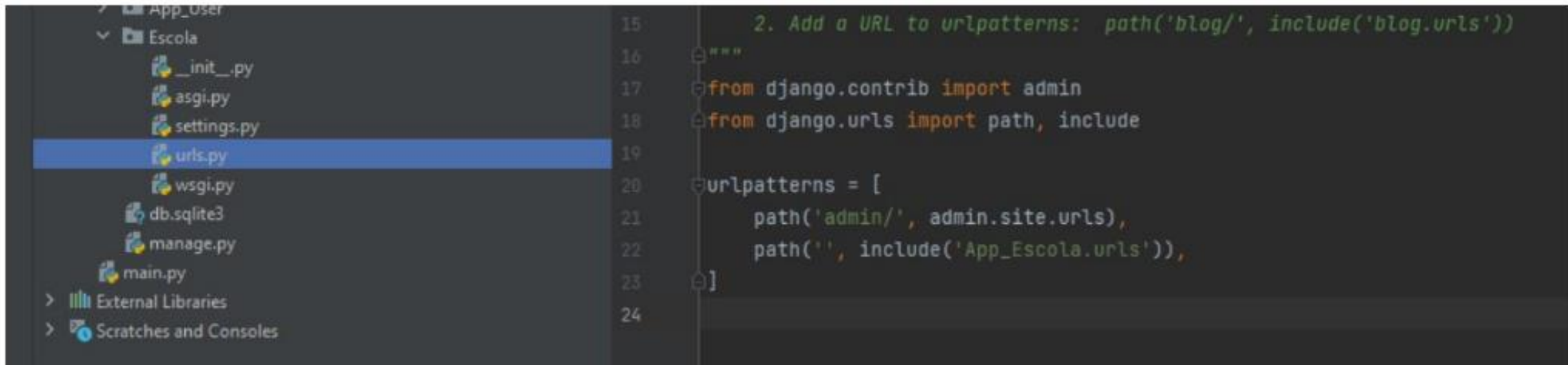
# BackEnd



# BackEnd

Para adicionar a biblioteca include altere a linha que tenha Django.urls, colocando “, include” no final

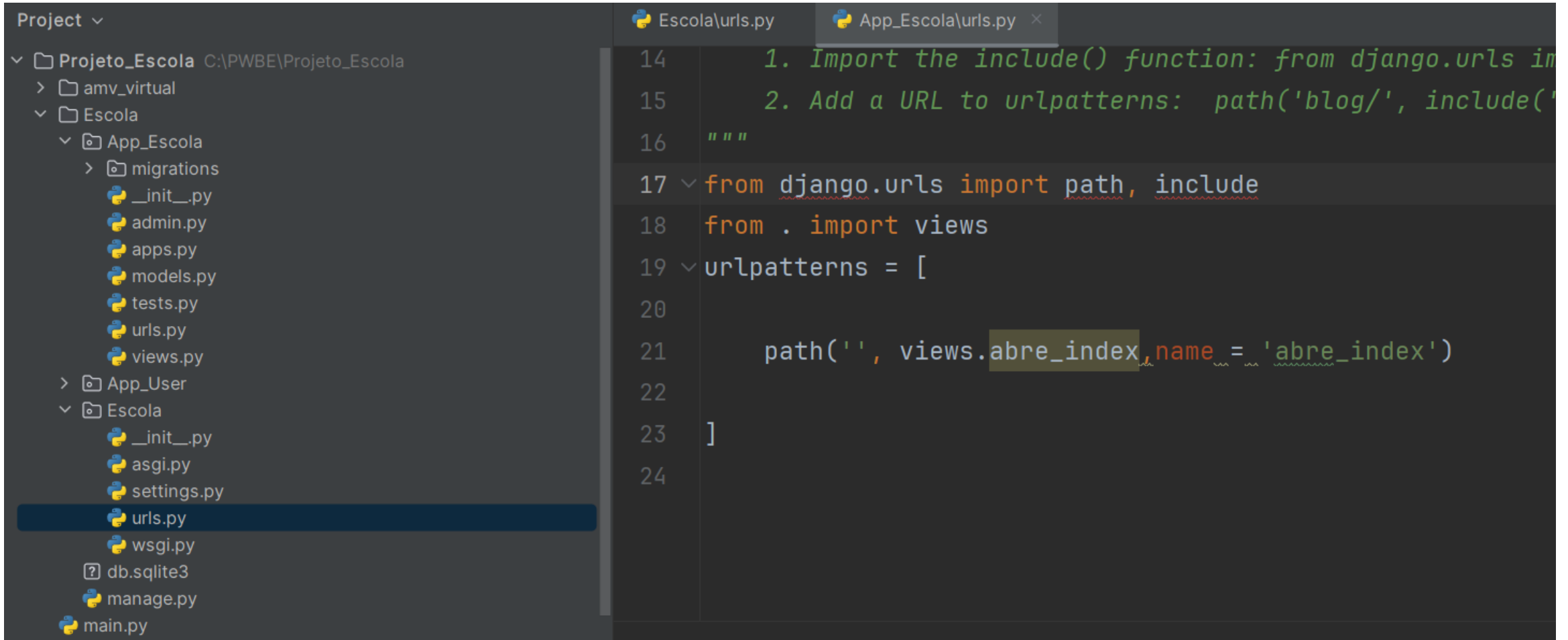
E para incluirmos o arquivo urls.py do App\_Escola acrescente a linha na sessão urlpatterns conforme a imagem



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a folder named 'Escola' containing files like \_\_init\_\_.py, asgi.py, settings.py, urls.py (highlighted), wsgi.py, db.sqlite3, manage.py, and main.py. The code editor shows the contents of settings.py, specifically the urlpatterns list. The code is as follows:

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

# BackEnd



The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure for 'Projeto\_Escola' located at 'C:\PWBE\Projeto\_Escola'. The 'Escola' app is selected, and its 'urls.py' file is highlighted. The code editor shows the 'App\_Escola\urls.py' file with the following code:

```
14 1. Import the include() function: from django.urls import include
15 2. Add a URL to urlpatterns: path('blog/', include('
16 """
17 from django.urls import path, include
18 from . import views
19 urlpatterns = [
20
21     path('', views.abre_index, name = 'abre_index')
22
23 ]
24
```

# BackEnd

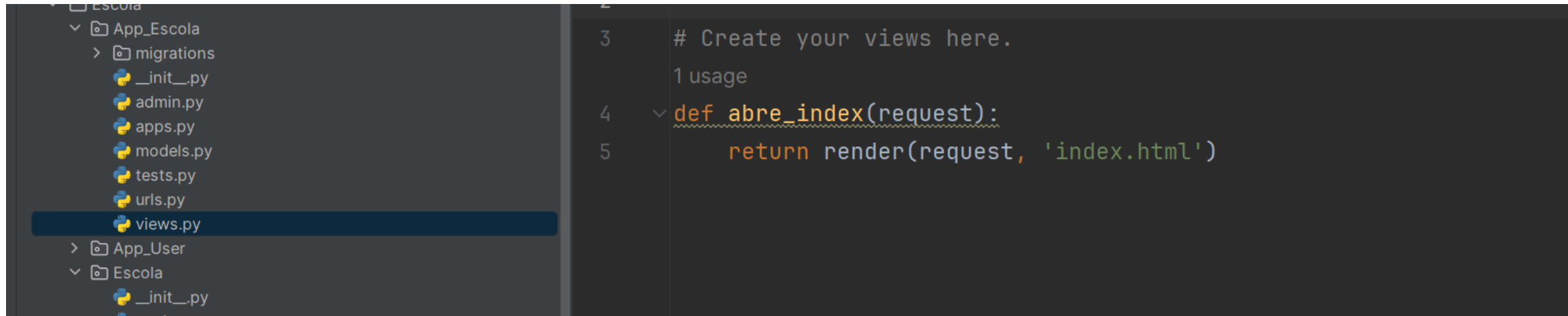
Observe que a função `abre_index` aparece em vermelho pois ainda não criamos essa função dentro do arquivo `views.py`.

O arquivo `views.py` foi criado automaticamente pelo Django dentro do `App_Escola` quando criamos esse app.

Neste arquivo deveremos descrever todas as funções do nosso app. A primeira função que descreveremos nesse arquivo é justamente a função `abre_index` que será responsável por chamar a primeira página de nossa aplicação

Para isso, abra o arquivo `views.py` do `App_Escola` dando um duplo clique sobre o arquivo para edição.

# BackEnd



The image shows a screenshot of a Django project's file explorer and code editor. The file explorer on the left shows the project structure, with the 'views.py' file in the 'App\_Escola' app selected. The code editor on the right shows the content of 'views.py', which includes a comment '# Create your views here.', a docstring '1 usage', and a function definition 'def abre\_index(request):' that returns 'render(request, 'index.html')'.

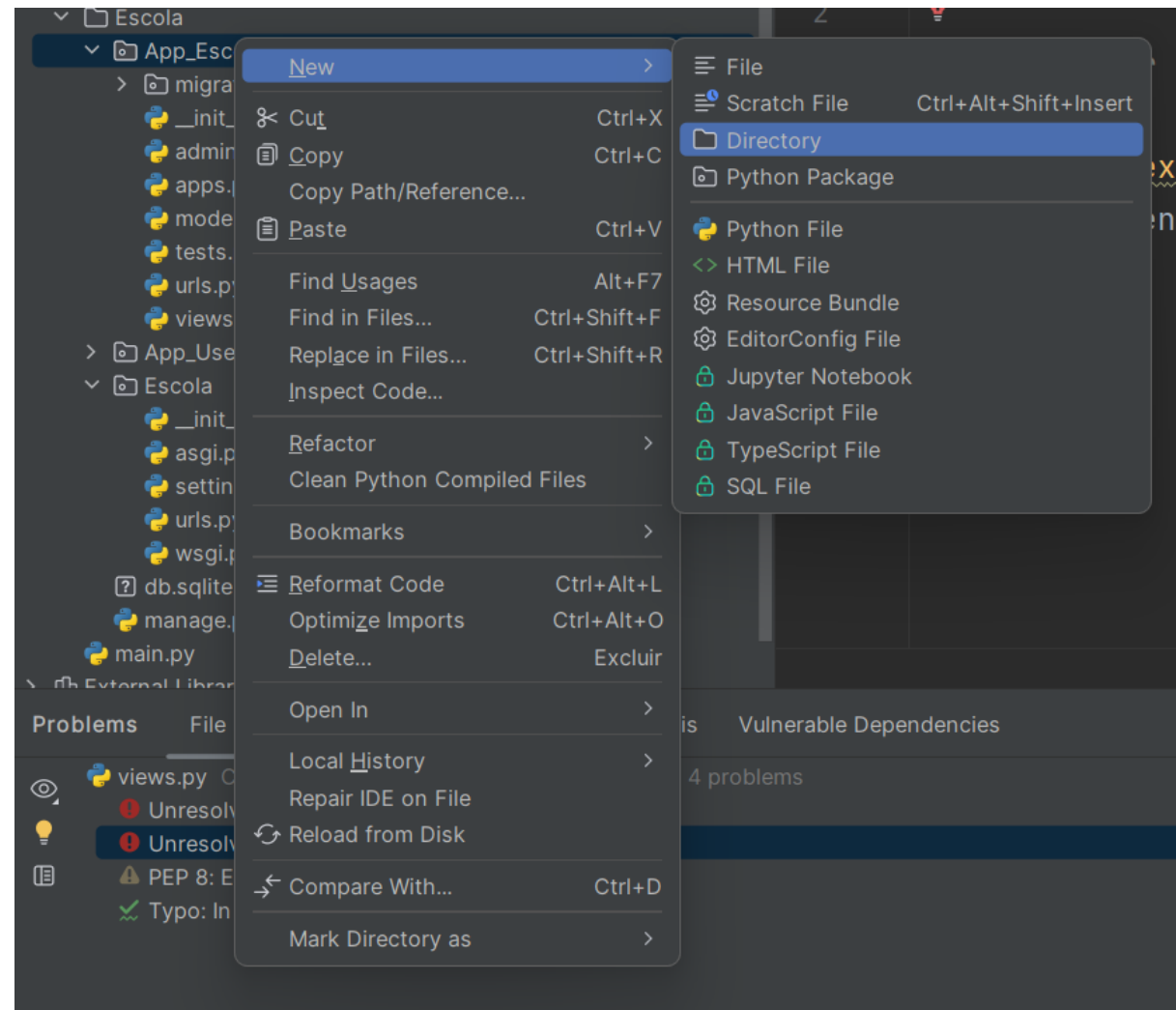
```
2  
3 # Create your views here.  
1 usage  
4 def abre_index(request):  
5     return render(request, 'index.html')
```

# BackEnd

Nas linhas acima nós criamos a função abre\_index e mandamos que seja respondido com a página Index.html.

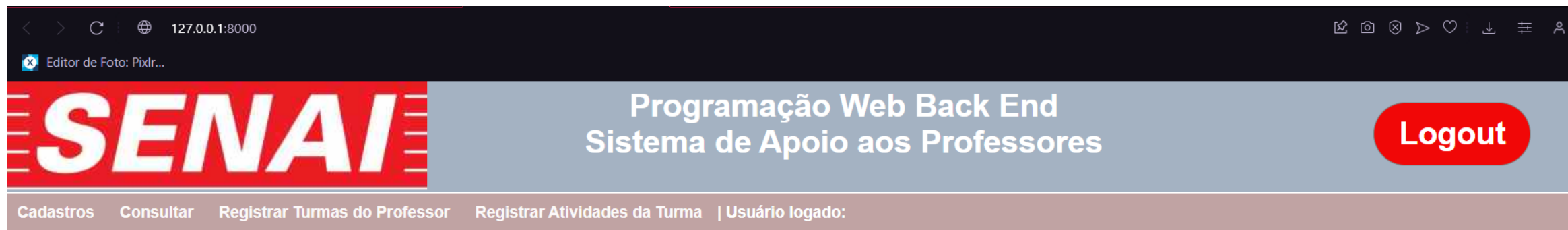
O Django tem como padrão receber todas as páginas html dentro da pasta **templates** do App. Porém essa pasta ele não cria automaticamente e nós vamos criá-la agora.

Clique sobre a pasta **App\_Escola** > clique com o botão direito do mouse ➤ **New > Directory** e na caixa que aparece digite **templates**. E em seguida vamos acomodar um arquivo HTML inicial.



# BackEnd

Fernanda Fretes



# BackEnd

Dentro dessa pasta templates devemos colocar todos os arquivos html do App\_Escola desenvolvidos como “Front End” do sistema.  
Faça isso para que possamos continuar o desenvolvimento do App\_Escola.

Os arquivos CSS, JS e Imagens também têm local definido no Django.  
No arquivo **settings.py** do projeto (já usamos esse arquivo lá no começo para configurar a linguagem do Admin do Django), temos uma sessão chamada `STATIC_URL` onde devemos indicar onde colocaremos os arquivos estáticos do app.

Configure para:

**`STATIC_URL = '/static/'`**

(Atenção para colocar entre duas barras e apóstrofes.)

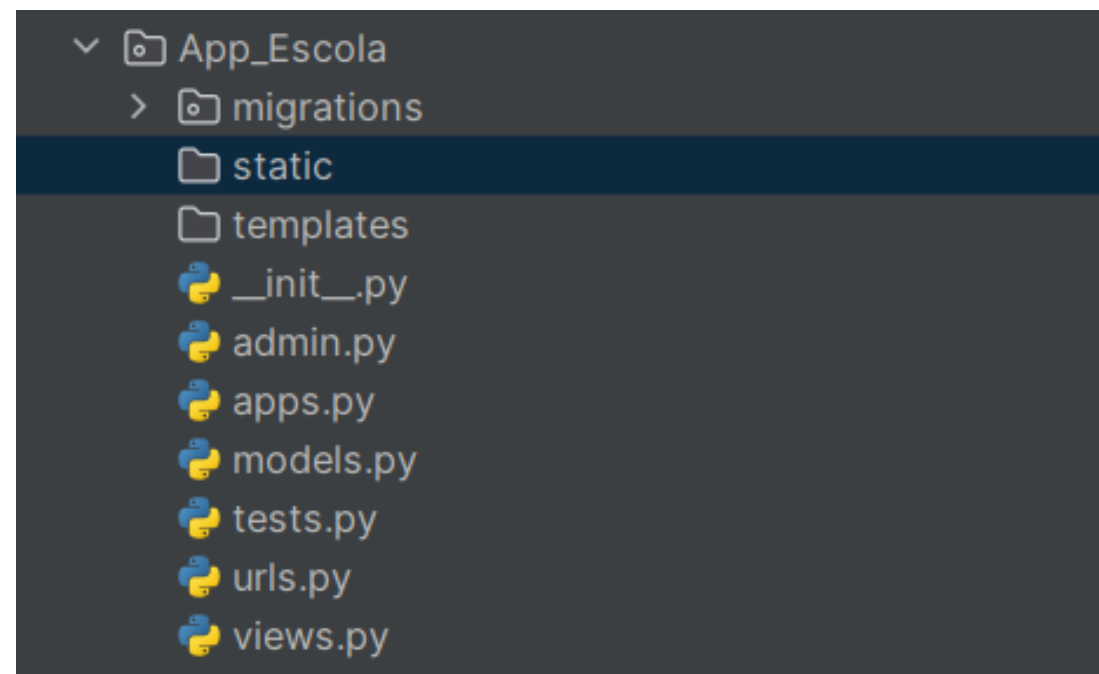
```
STATIC_URL = '/static/'
```



# BackEnd

Agora crie uma nova pasta na mesma estrutura onde você criou a pasta templates, ou seja, dentro do App\_Escola com o nome static.

Para isso, a partir do PyCharm, Clique sobre a pasta **App\_Escola** > clique com o botão direito do mouse **New** > **Directory** e na caixa que aparece digite **static** .

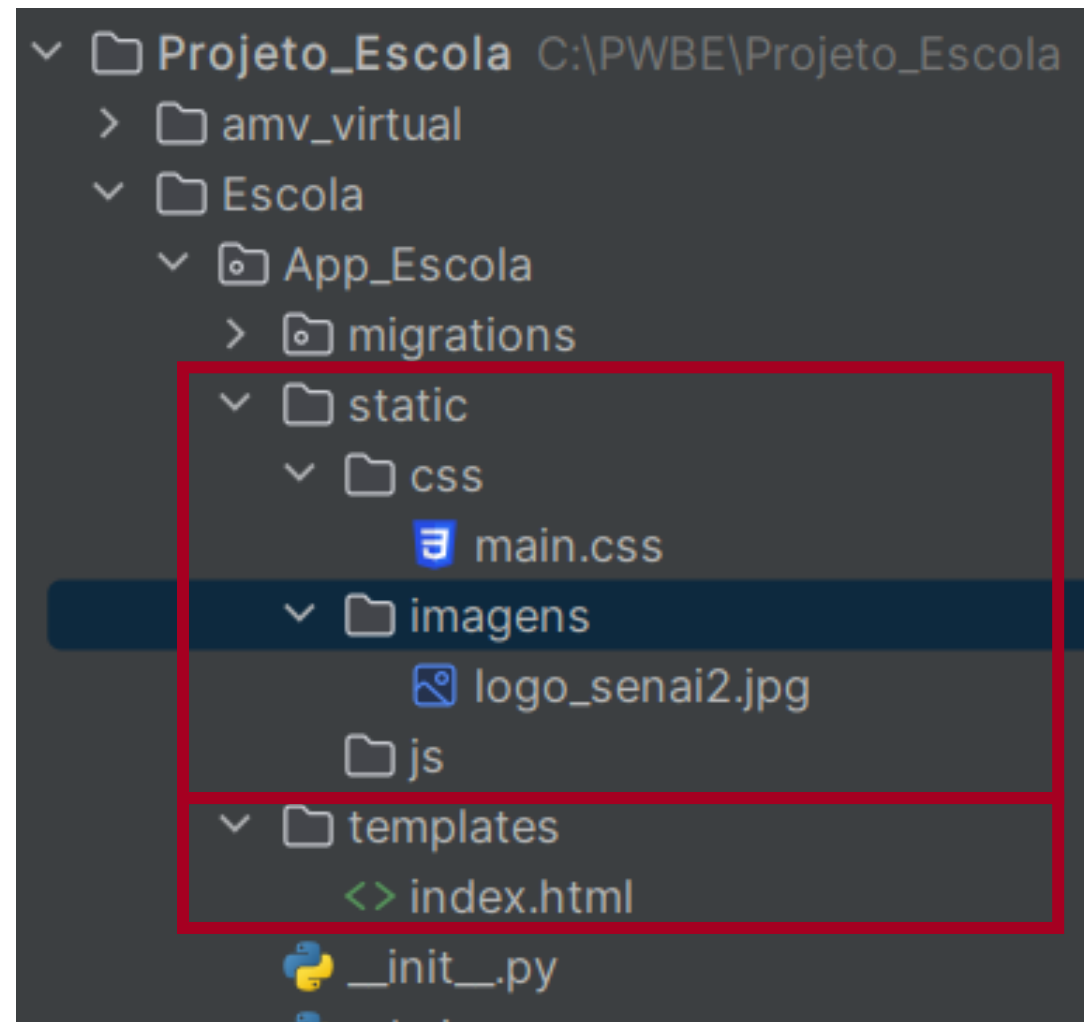


# BackEnd

Agora crie uma nova pasta na mesma estrutura onde você criou a pasta templates, ou seja, dentro do App\_Escola com o nome static.

Para isso, a partir do PyCharm, Clique sobre a pasta **App\_Escola** > clique com o botão direito do mouse **New** > **Directory** e na caixa que aparece digite **static**.

Dentro dessa pasta static devemos criar as pastas CSS, JS e Imagens desenvolvidas como “Front End” do sistema. Faça isso para que possamos continuar o desenvolvimento do App\_Escola.



# BackEnd

Agora abra o arquivo Index.html e coloque a 1 linha de código

```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" type="text/css" href="{% static 'css/main.css' %}">

    <title>Programação Web Back End</title>
</head>

<header class="login">
    <div class="fundo_menu">
        <div class="logo_senai">
            <a href="index.html"></a>
```

Dentro da tag altere a linha que faz o link com o arquivo css conforme a imagem  
Altere também a tag que faz o link com o logo do Senai conforme mostrado:

# BackEnd

Feito esses passos, vamos acessar o servidor e verificar se já está respondendo a primeira página do nosso projeto Escola.

Caso o servidor esteja em execução, interrompa para que possamos reiniciá-lo. A partir do Terminal digite Ctrl + C. Limpe a tela do terminal executando o comando `cls` . Inicie novamente o servidor digitando: **`python manage.py runserver`**

