


Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

Prof. Israel Gomes da Silva

**PROGRAMAÇÃO WEB BACK END
UTILIZANDO FRAME WORK DJANGO PYTHON
(TEORIA E PRÁTICA)**

^{2a}. Edição
Março/2024

A solid gray horizontal bar spanning the width of the page.

Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

Nesta Unidade Curricular temos como objetivo:

“Proporcionar a aquisição de capacidades técnicas relativas ao desenvolvimento de sistemas web promovendo a interação de aplicação entre cliente e servidor e outros sistemas computacionais, realizando persistência de dados, bem como o desenvolvimento de capacidades sociais, organizativas e metodológicas adequadas a diferentes situações profissionais.”

(Fonte: Objetivos do Plano de Curso)

E neste segundo Sprint, para desenvolvimento desse objetivo utilizaremos a linguagem de programação **Python** com o framework **Django** onde desenvolveremos como referência para aquisição do conhecimento um **Sistema de Apoio a Professores**.

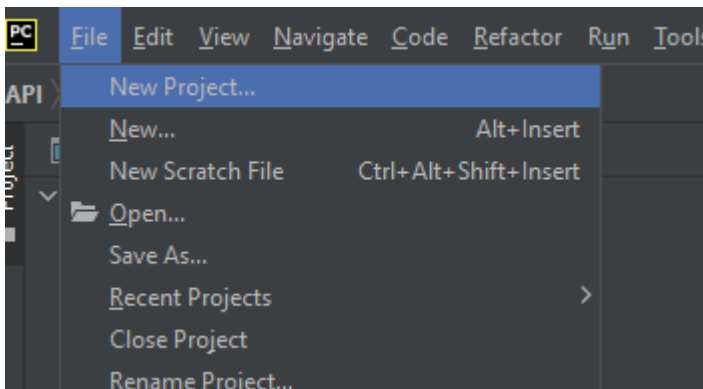
Criaremos o **Projeto_Escola** a partir do zero como exemplo , ou seja, desde a configuração dos ambientes.

Criando o **Projeto_Escola** a partir do zero como exemplo.

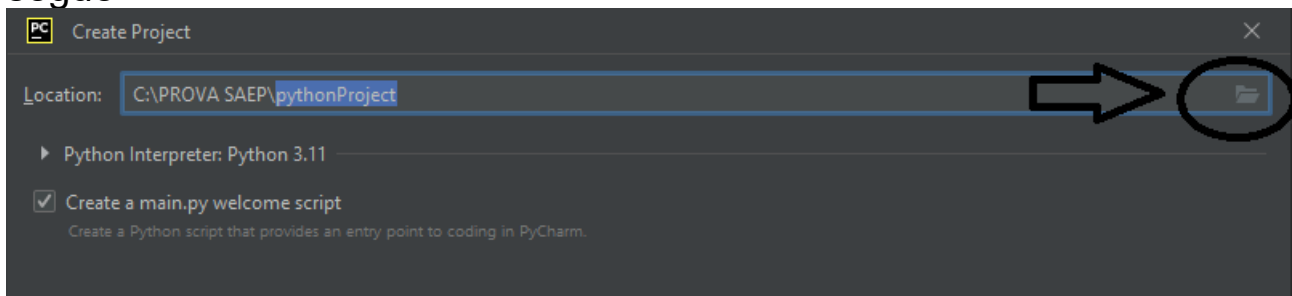
Capítulo 1 – Criando o Projeto_Escola e seus App's

1.1 – Com o PyCharm aberto, clique em

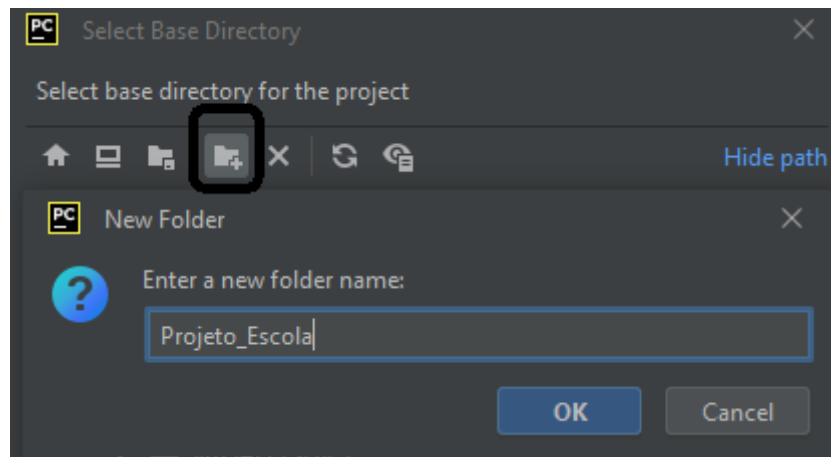
“File > New Project”



1.2 – No campo **Location selecione ou crie a pasta onde você armazenará o seu projeto selecionando o ícone conforme a figura que segue**



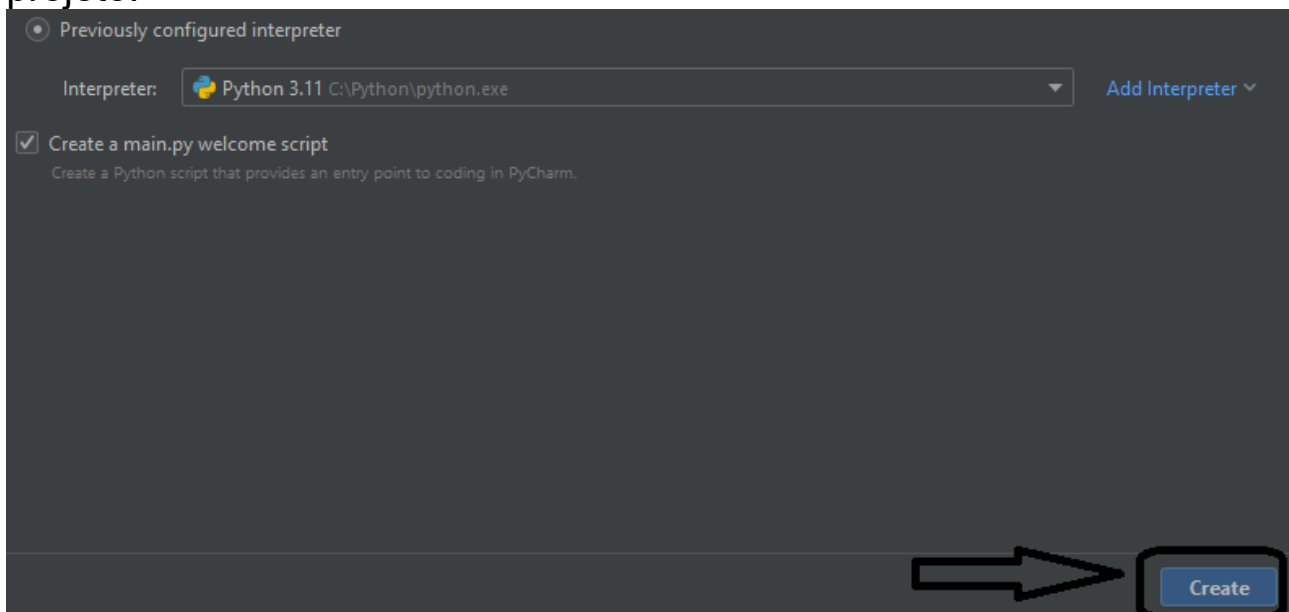
1.3 – No meu caso criei uma nova pasta (New Directory) no caminho
C:\PWBE\Sprint_2\Projeto_Escola



1.4 - Após criar a pasta do projeto e clicar em **OK**, deve ficar como indicado na figura abaixo. 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).

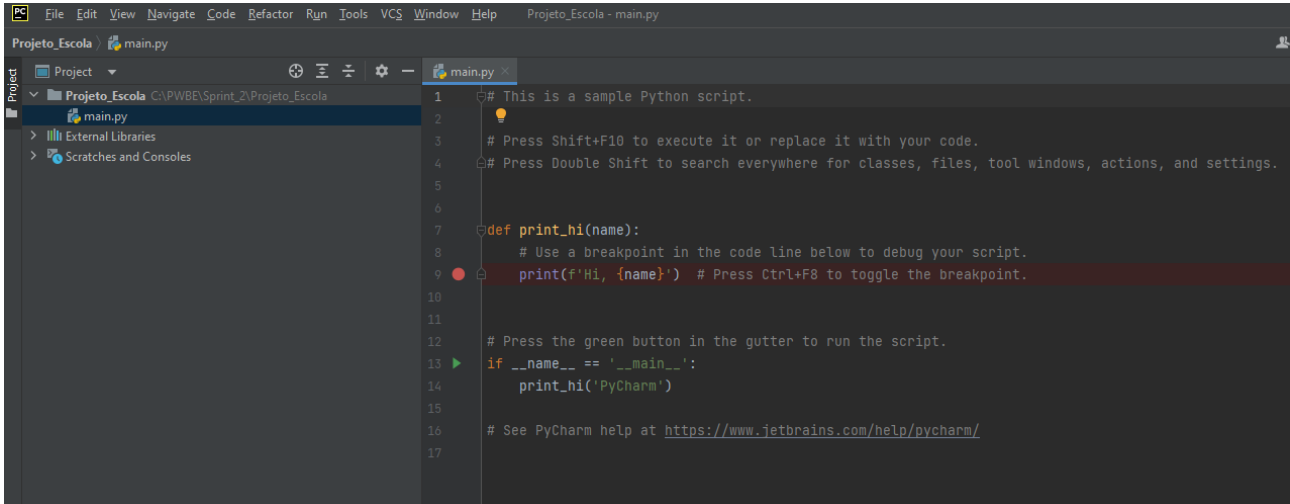
Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

Feito isso clique no botão **Create** na canto inferior da tela e abra o projeto.



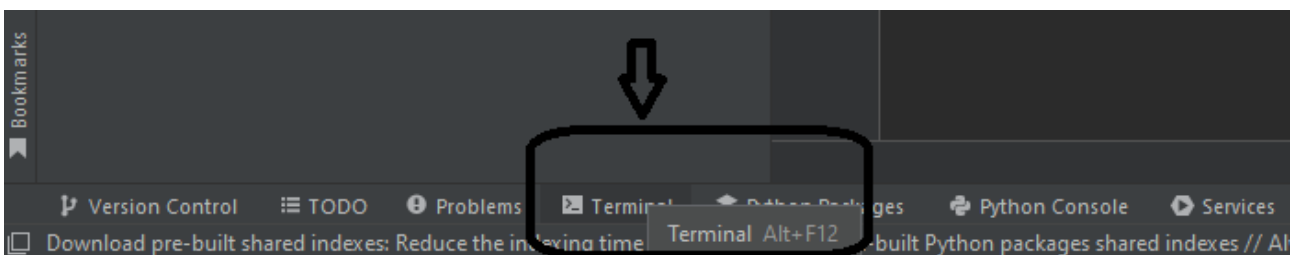
Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

A tela deve ficar como mostrado abaixo

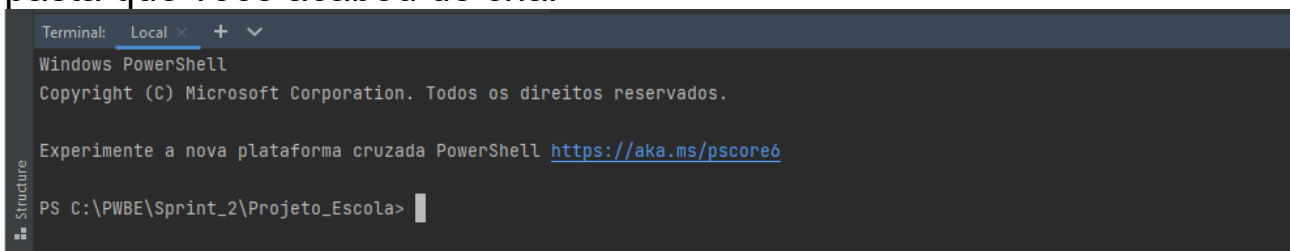


1.5 – Vamos criar um novo **ambiente virtual** onde acomodaremos todas as instalações de bibliotecas que utilizaremos.

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



Quando o ambiente do terminal aparecer, certifique-se que esteja na pasta que você acabou de criar



Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

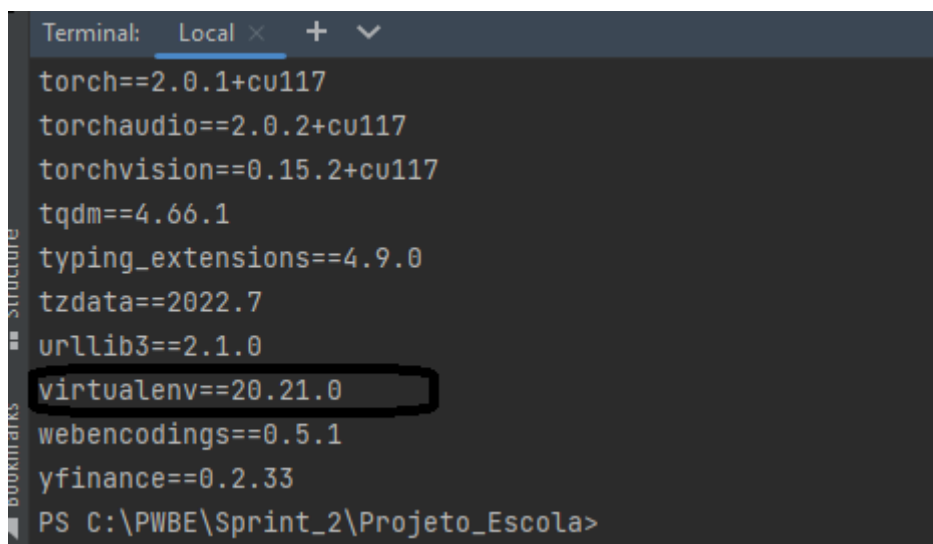
Para criar o ambiente virtual é necessário que a biblioteca **virtualenv** esteja instalada no seu computador. Para isso, execute o comando como segue:

```
PS C:\PWBE\Sprint_2\Projeto_Escola> pip install virtualenv
```

Após exibir algumas linhas referente a instalação e retornar ao prompt de comando “PS C:\PWBE\Sprint_2\Projeto_Escola>”, vamos verificar se a biblioteca foi corretamente instalada executando o comando **pip freeze**. Este comando exibe todas as bibliotecas **Python** que estão instaladas no seu computador. Execute então o comando como segue:

```
PS C:\PWBE\Sprint_2\Projeto_Escola> pip freeze
```

Dentre as várias bibliotecas que já possam estar instaladas em seu computador e que ele poderá exibir, você deve localizar a biblioteca em questão que é a **virtualenv** conforme a figura que segue:

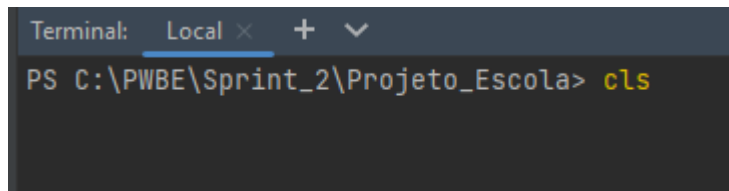


```
Terminal: Local x + v
torch==2.0.1+cu117
torchaudio==2.0.2+cu117
torchvision==0.15.2+cu117
tqdm==4.66.1
typing_extensions==4.9.0
tzdata==2022.7
urllib3==2.1.0
virtualenv==20.21.0
webencodings==0.5.1
yfinance==0.2.33
PS C:\PWBE\Sprint_2\Projeto_Escola>
```

Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

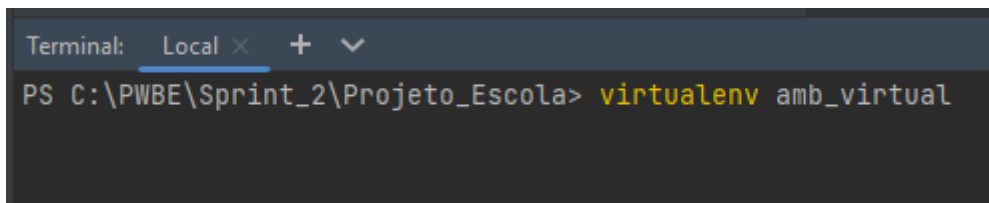
Para limpar a tela do terminal dessas várias linhas, execute o comando **cls** como segue:

```
PS C:\PWBE\Sprint_2\Projeto_Escola> cls
```

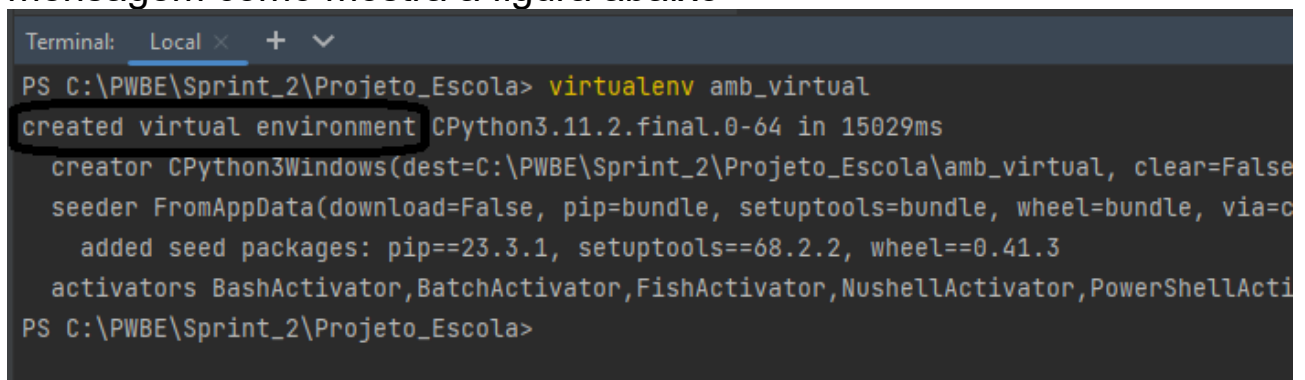
A screenshot of a Windows terminal window. The title bar says 'Terminal: Local'. The command prompt shows 'PS C:\PWBE\Sprint_2\Projeto_Escola> cls'. The screen is currently blank, indicating the command was successful in clearing the terminal.

Agora então vamos criar o ambiente virtual para o nosso Projeto_Escola. Digite o comando para isso como segue:

```
PS C:\PWBE\Sprint_2\Projeto_Escola> virtualenv amb_virtual
```

A screenshot of a Windows terminal window. The title bar says 'Terminal: Local'. The command prompt shows 'PS C:\PWBE\Sprint_2\Projeto_Escola> virtualenv amb_virtual'. The terminal is currently blank, waiting for the command to complete.

Se o ambiente virtual foi criado com sucesso, deve exibir uma mensagem como mostra a figura abaixo

A screenshot of a Windows terminal window showing the successful execution of the 'virtualenv' command. The title bar says 'Terminal: Local'. The command prompt shows 'PS C:\PWBE\Sprint_2\Projeto_Escola> virtualenv amb_virtual'. The output is as follows:
created virtual environment CPython3.11.2.final.0-64 in 15029ms
creator CPython3Windows(dest=C:\PWBE\Sprint_2\Projeto_Escola\amb_virtual, clear=False
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=c
added seed packages: pip==23.3.1, setuptools==68.2.2, wheel==0.41.3
activators BashActivator, BatchActivator, FishActivator, NushellActivator, PowerShellActi
PS C:\PWBE\Sprint_2\Projeto_Escola>

Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

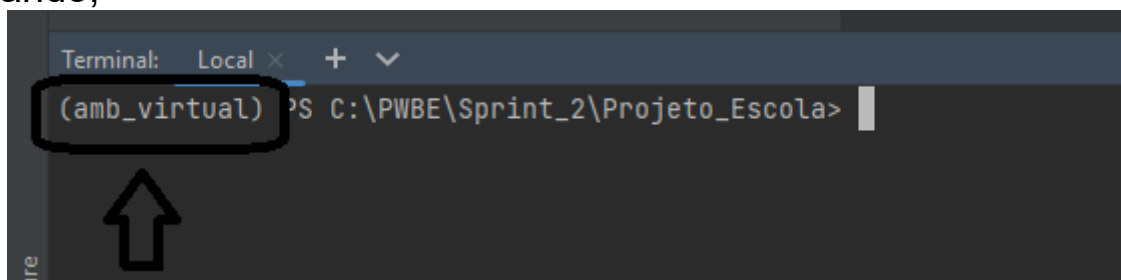
Limpe novamente a tela digitando `cls`

```
PS C:\PWBE\Sprint_2\Projeto_Escola>cls
```

1.6 – Agora iremos ativar o nosso ambiente virtual. Digite então o comando como segue

```
PS C:\PWBE\Sprint_2\Projeto_Escola>amb_virtual\Scripts\activate
```

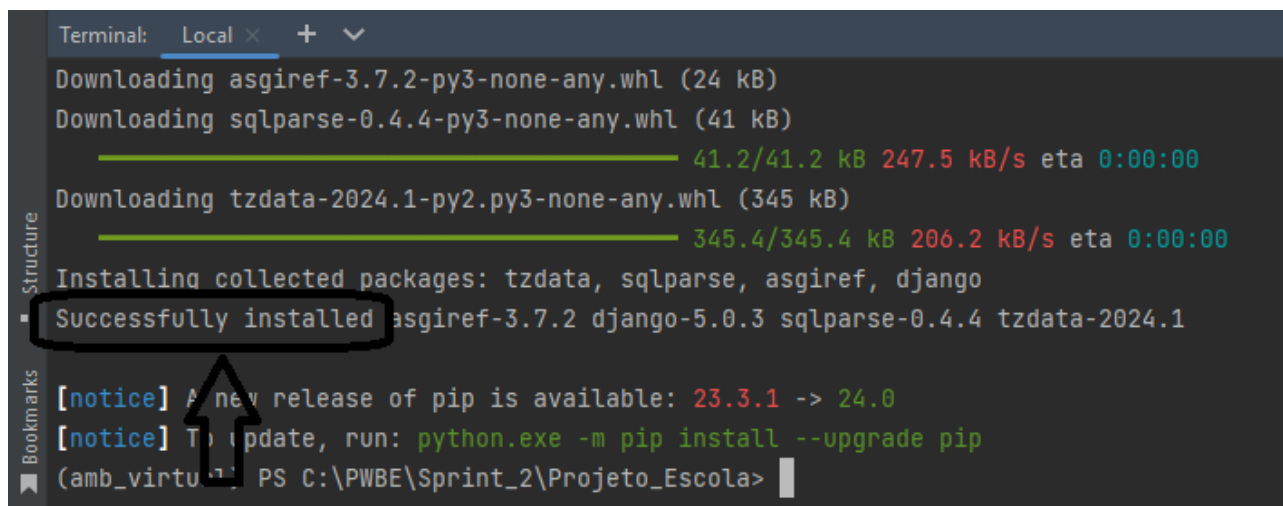
A resposta esperada é a que segue na figura abaixo onde o nome do ambiente virtual aparece entre parênteses antes do prompt de comando,



1.7 – Neste ambiente devemos fazer as instalações que precisaremos. A primeira é o Framework **DJANGO**. Digite o comando como segue:

```
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola> pip install django
```

A instalação deve transcorrer sem erros exibindo algumas linhas dentre elas a informação de que a instalação foi feita com sucesso como a imagem que segue.

Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

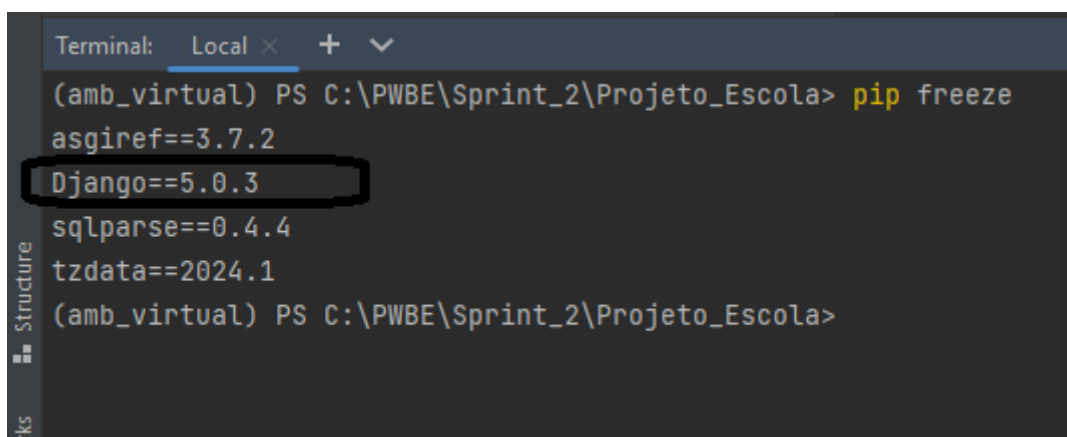
```
Terminal: Local x + v
Downloading asgiref-3.7.2-py3-none-any.whl (24 kB)
Downloading sqlparse-0.4.4-py3-none-any.whl (41 kB)
41.2/41.2 kB 247.5 kB/s eta 0:00:00
Downloading tzdata-2024.1-py2.py3-none-any.whl (345 kB)
345.4/345.4 kB 206.2 kB/s eta 0:00:00
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.7.2 django-5.0.3 sqlparse-0.4.4 tzdata-2024.1
[notice] A new release of pip is available: 23.3.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola>
```

Para certificar-se que o **Django** foi instalado no ambiente virtual vamos executar novamente o comando **pip freeze**. Mas antes disso, execute novamente o comando **cls** para limpar a tela.

```
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola> cls
```

Agora sim, com a tela mais limpa, vamos executar o comando **pip freeze** para verificar se o **Django** foi instalado. Digite o comando como segue:

```
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola> pip freeze
```



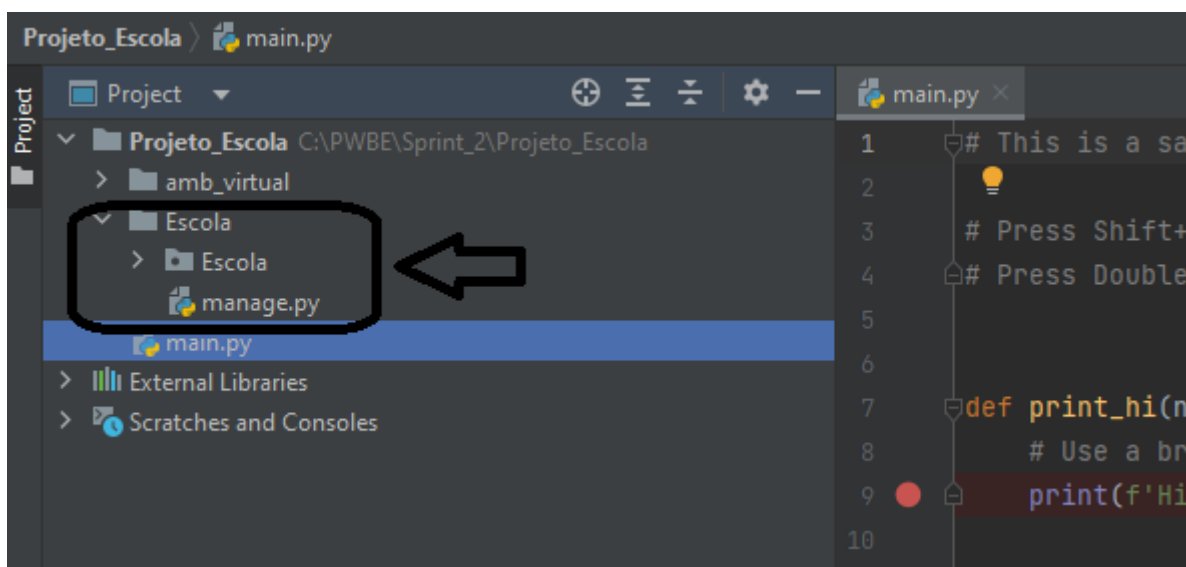
```
Terminal: Local x + v
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola> pip freeze
asgiref==3.7.2
Django==5.0.3
sqlparse==0.4.4
tzdata==2024.1
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola>
```

1.8 – 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\Sprint_2\Projeto_Escola> django-admin startproject 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:

```
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola> cd Escola
```

O resultado é o que segue abaixo:

Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

```
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola\Escola>
```

1.9 – Um projeto Django é composto por um ou mais App's. Agora dentro do projeto **Escola**, criaremos o **App_Escola**. Antes, limpe novamente a tela do terminal digitando o comando **cls**.

Com a tela do terminal limpa vamos digitar o comando **startapp** para criar o nosso **App_Escola**. Digite então o comando como segue:

```
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola\Escola> python manage.py  
startapp App_Escola
```

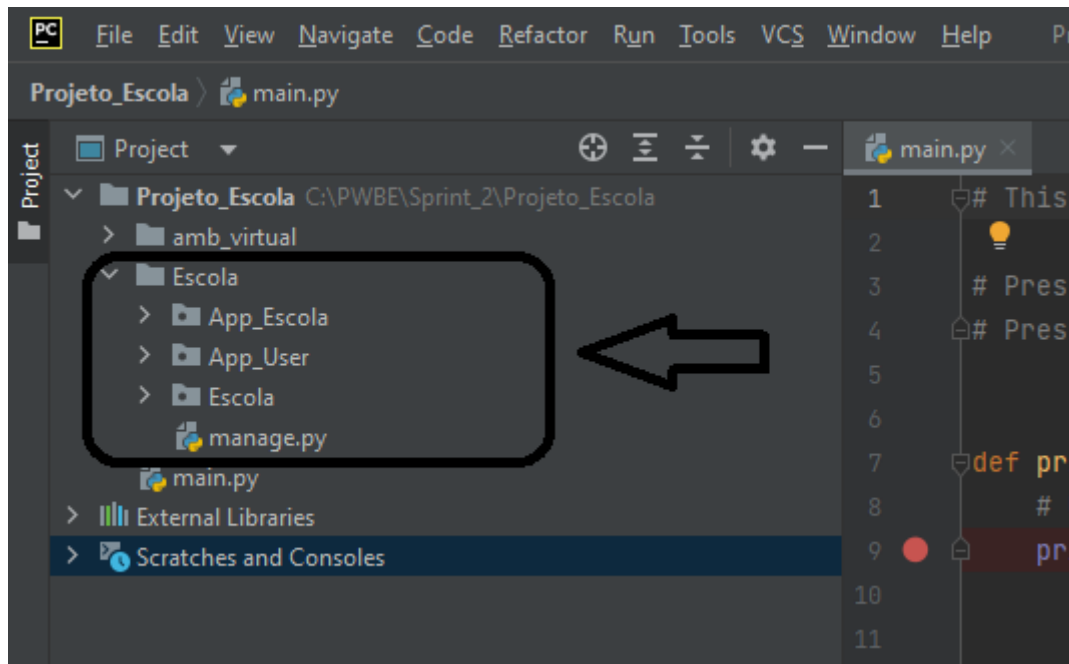
Vamos deixar criado um segundo app para o nosso projeto **Escola** que será o app responsável por gerenciar os acessos de usuários ao sistema.

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:



Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

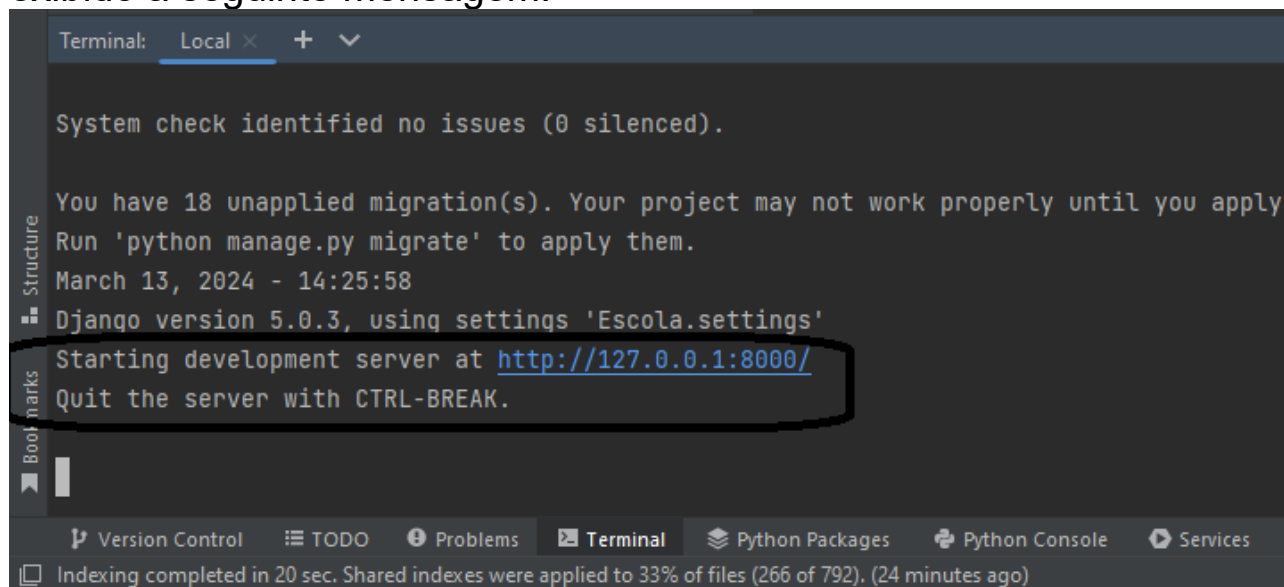
1.10 - Vamos testar se o Django já está funcionando. Para isso executamos o servidor local onde será executado nosso projeto;

Digite o comando que segue:

```
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola\Escola> python manage.py  
runserver
```

Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

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



```
Terminal: Local x + v

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply
Run 'python manage.py migrate' to apply them.
March 13, 2024 - 14:25:58
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.
```

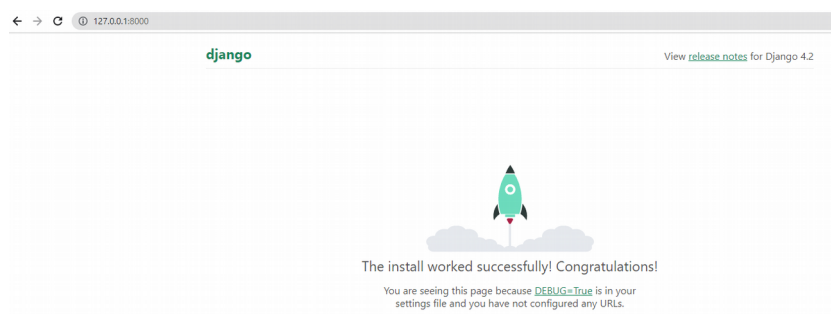
Structure

Bookmarks

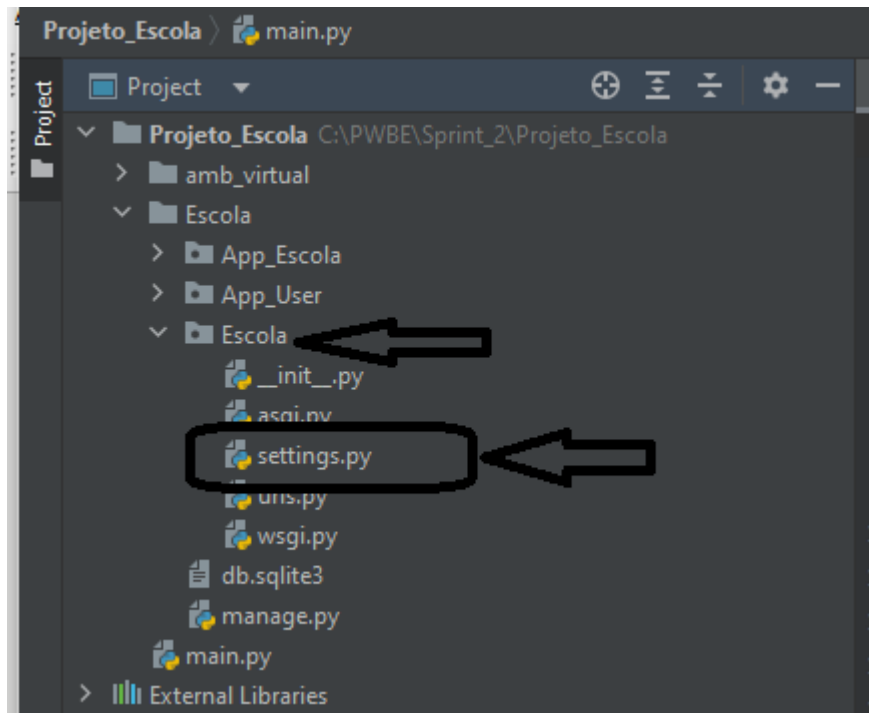
Version Control | TODO | Problems | Terminal | Python Packages | Python Console | Services

Indexing completed in 20 sec. Shared indexes were applied to 33% of files (266 of 792). (24 minutes ago)

Agora abriremos o Browser, neste exemplo usaremos o Google Chrome, no endereço indicado <http://127.0.0.1:8000/> . Deve ser mostrado a imagem que segue:



1.11 – Vamos fazer uma primeira alteração que é configurar para que o Django seja exibido em Português. Acesse o arquivo **settings.py** que está dentro da pasta do Escola como mostrado na figura que segue:

Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

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:

Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

A instalação foi com sucesso! Parabéns!

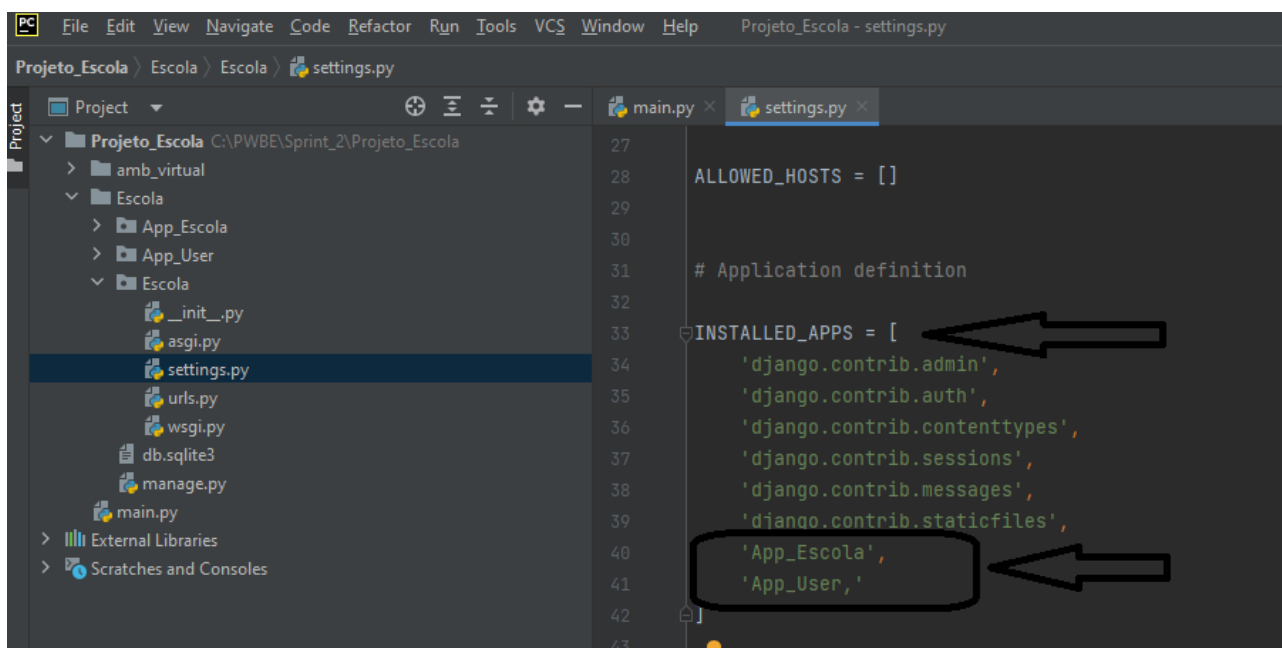
Você está vendo esta página pois possui `DEBUG=True` no seu arquivo de configurações e não configurou nenhuma URL.

1.12 – Associando o App ao Projeto.

Como dito anteriormente, 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.

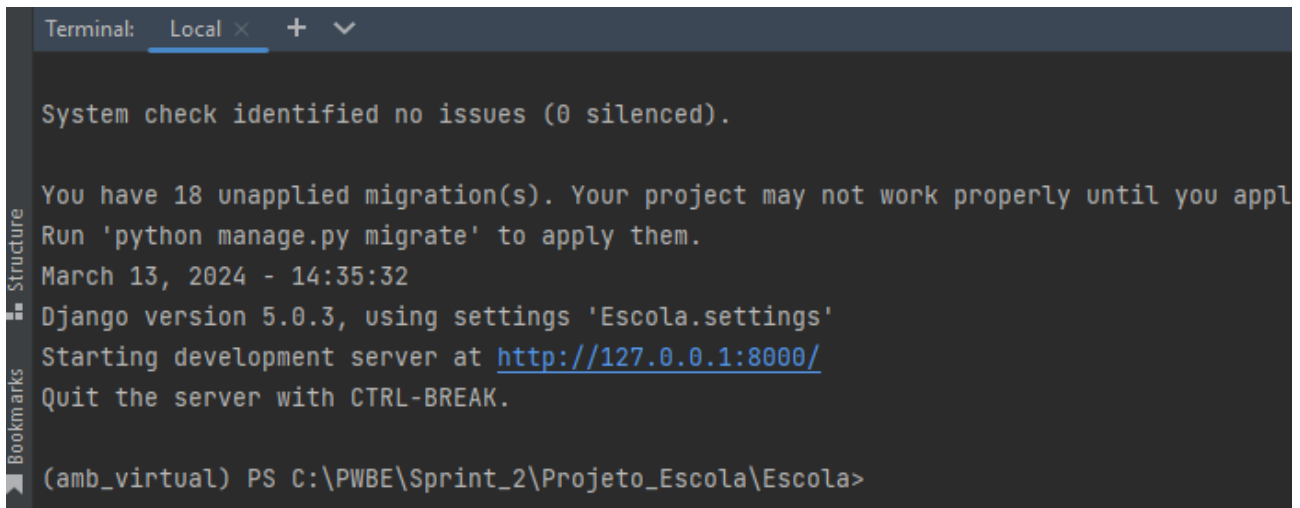
Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

1.13 – Migrate.

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:

Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

```
Terminal: Local x + v

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply them.
Run 'python manage.py migrate' to apply them.
March 13, 2024 - 14:35:32
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.

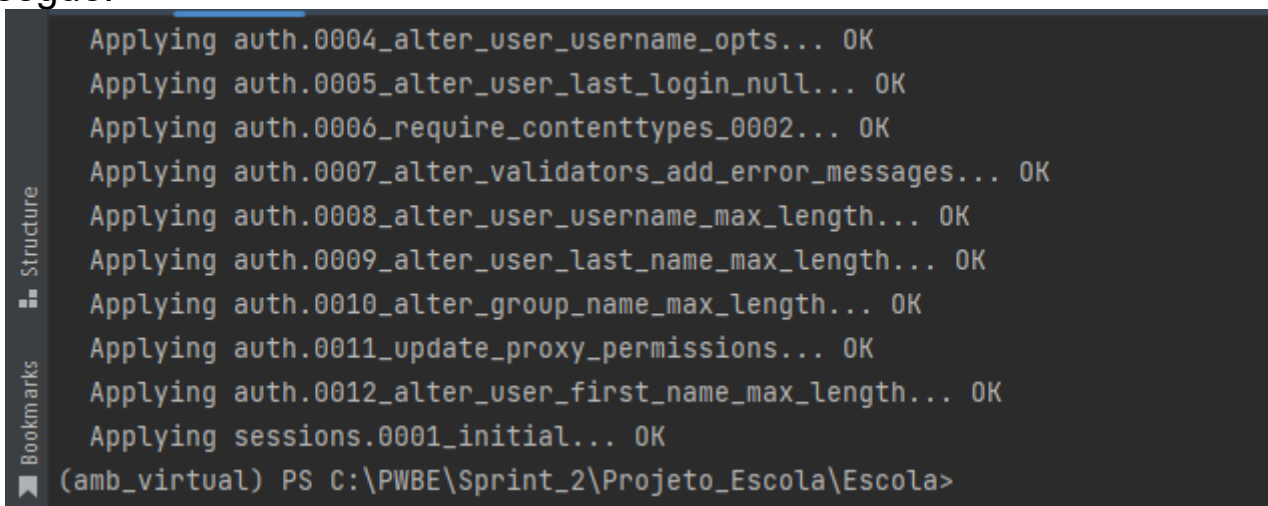
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola\Escola>
```

Execute novamente o comando **cls** para limpar a tela.

Com a tela limpa, execute o comando **migrate** como segue:

```
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola\Escola> python manage.py migrate
```

Uma série de linhas serão executadas conforme mostra a figura que segue:



```
Applying auth.0004_alter_user_username_opts... OK
Applying auth.0005_alter_user_last_login_null... OK
Applying auth.0006_require_contenttypes_0002... OK
Applying auth.0007_alter_validators_add_error_messages... OK
Applying auth.0008_alter_user_username_max_length... OK
Applying auth.0009_alter_user_last_name_max_length... OK
Applying auth.0010_alter_group_name_max_length... OK
Applying auth.0011_update_proxy_permissions... OK
Applying auth.0012_alter_user_first_name_max_length... OK
Applying sessions.0001_initial... OK
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola\Escola>
```

1.14 – Criando Super Usuário para o Projeto.

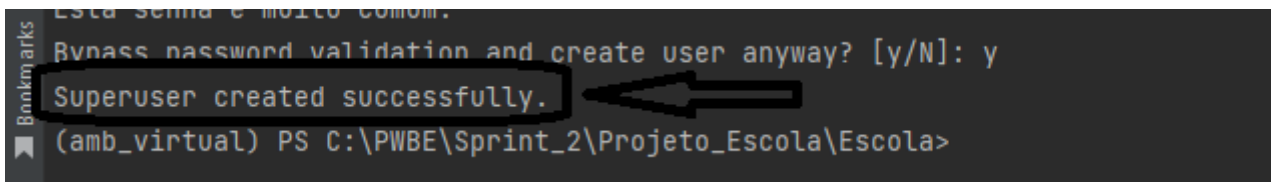
Executar o comando **migrate** no passo anterior fez com que o Banco de Dados usado pelo Administrador do projeto Django fosse criado. Porém para que possamos acessá-lo é necessário a criação de um **super usuário** ainda pela linha de comando do Terminal.

Mais uma vez execute o comando **cls** para limpar a tela 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.

A screenshot of a Windows PowerShell terminal window. The prompt is "(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola\Escola>". The user has entered "python manage.py createsuperuser". The terminal shows the prompt "Bypass password validation and create user anyway? [y/N]: y". Below that, the message "Superuser created successfully." is displayed, which is highlighted with a black rectangular box. A white arrow points from the right towards this box. The terminal prompt is then "(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola\Escola>".

```
PS C:\PWBE\Sprint_2\Projeto_Escola\Escola> python manage.py  
createsuperuser  
Bypass password validation and create user anyway? [y/N]: y  
Superuser created successfully.  
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola\Escola>
```

1.15 – 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.

Limpe a tela do Terminal com o comando **cls**.

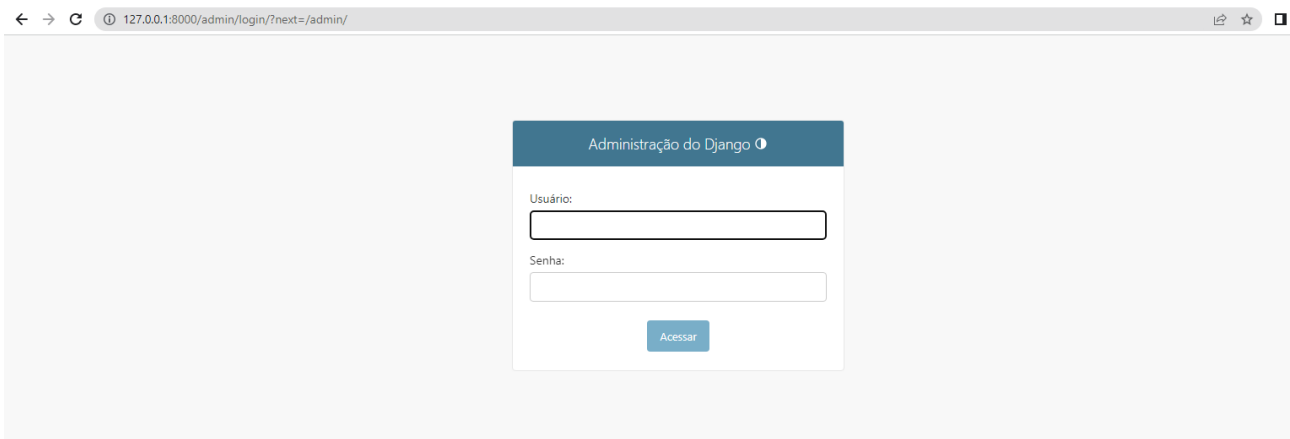
Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

Execute novamente o servidor local como segue:

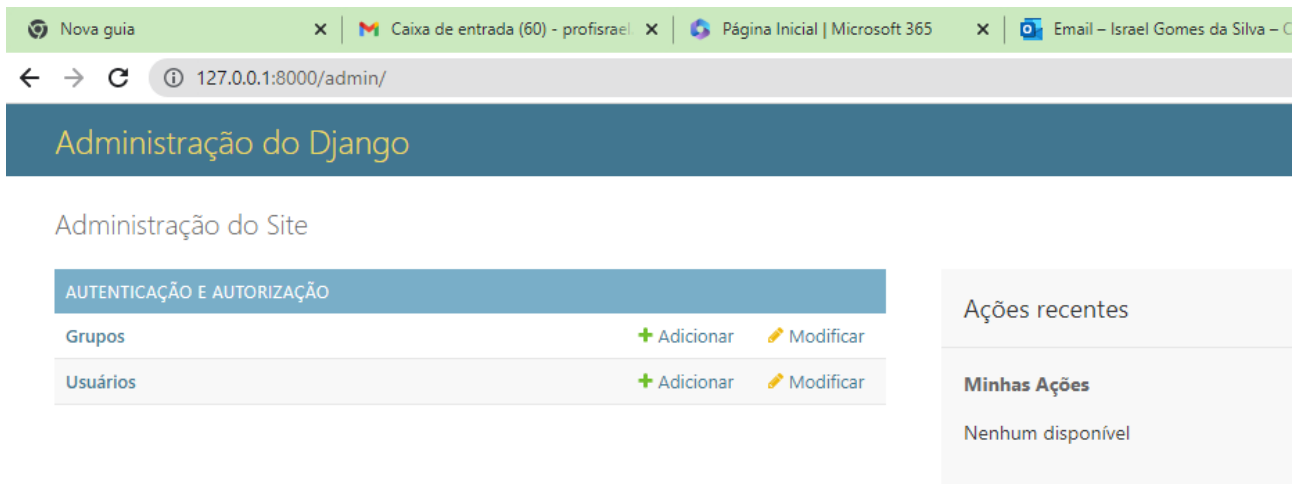
```
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola\Escola>python manage.py  
runserver
```

Agora abriremos o Browser, aqui estou usando o Google Chrome, no endereço indicado <http://127.0.0.1:8000/admin> .

Deve ser mostrado a imagem que segue:



Informe nos campos **usuário** e **senha** os dados do super usuário que você criou no passo anterior. Estando corretos a tela que segue será exibida.

Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

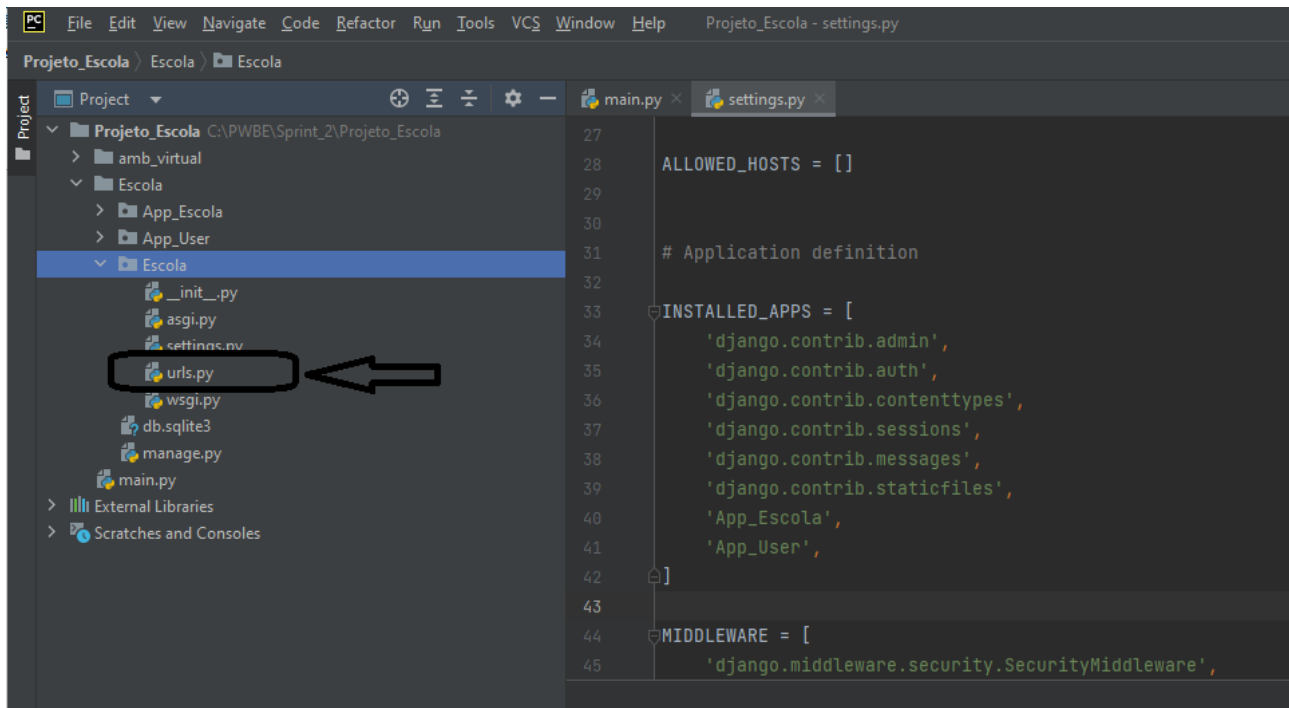
Interrompa a execução do servidor digitando no Terminal **<ctrl> + 'c'** e em seguida digite **cls** para limpar a tela.

1.16 – Criando URL's para o projeto.

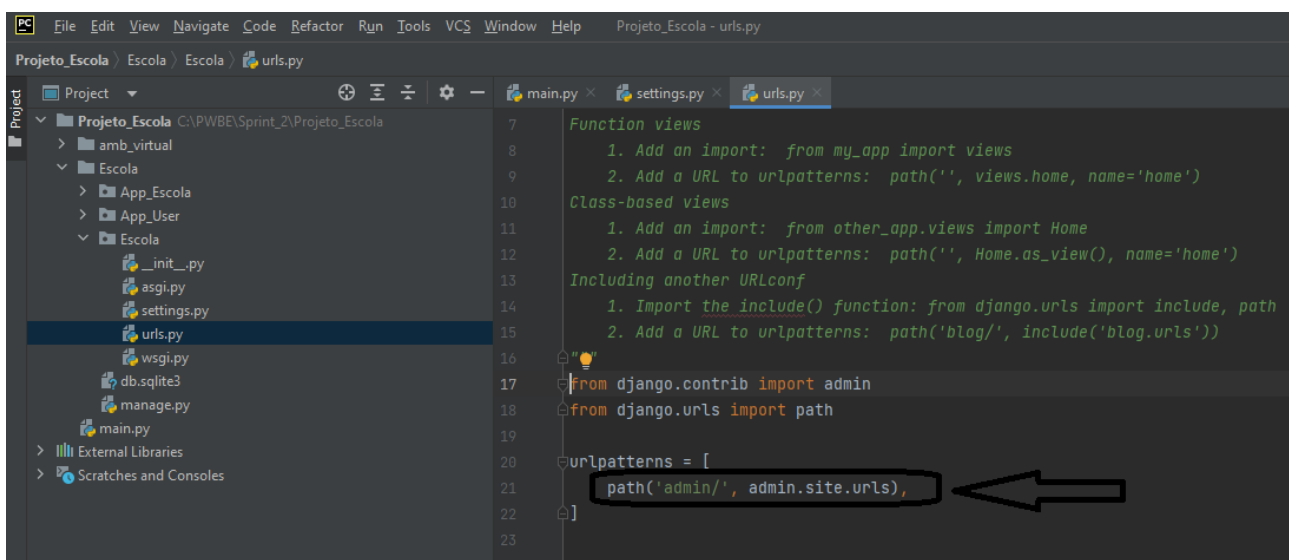
Ao executar o servidor nosso projeto responde na raiz do endereço, ou seja, <http://127.0.0.1:8000/>. que faz aparecer a tela do Django instalado com sucesso.

Observamos também que o administrador do nosso projeto Django responde no endereço <http://127.0.0.1:8000/admin>.

Este endereço está configurado no arquivo **urls.py** do projeto e foi criado pelo Django quando criamos o projeto **Escola**.

Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

Vamos abrir este arquivo para edição dando um duplo clique sobre ele:



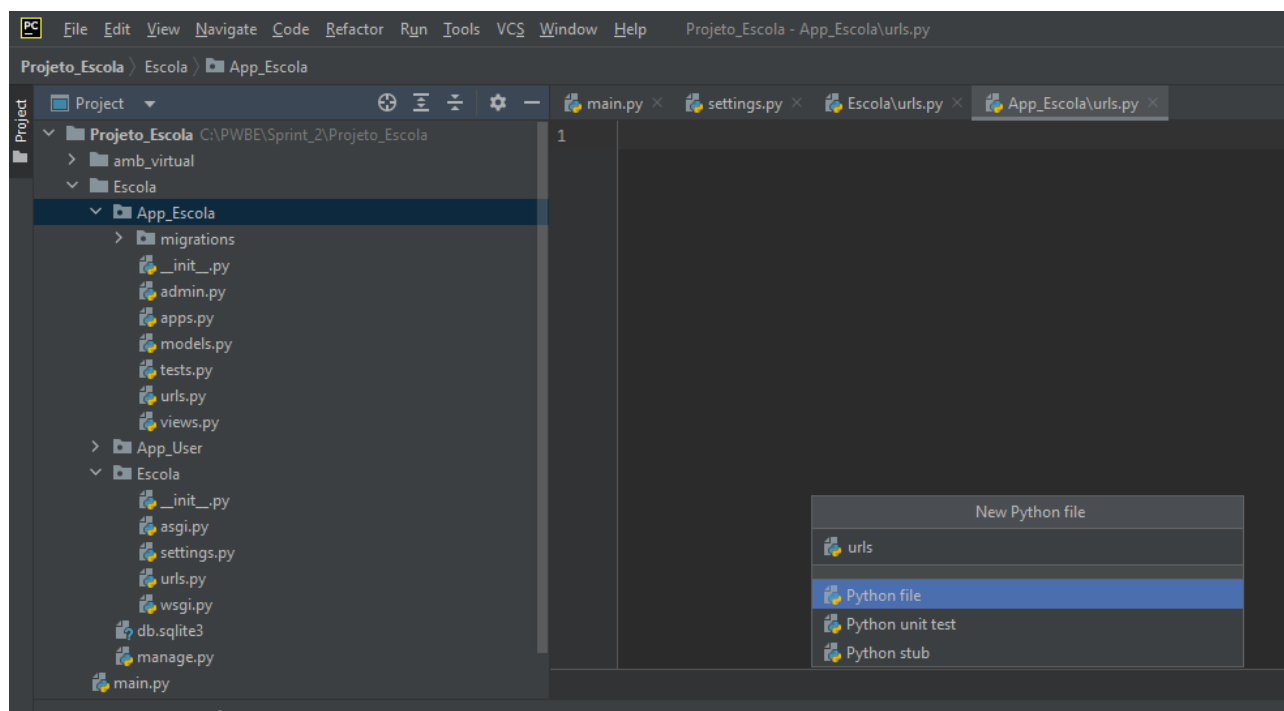
Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

Observe na figura acima que o caminho **admin** onde acessamos o administrador do projeto no Django está configurado na sessão **urlpatterns**. 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**



Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

Acesse novamente o arquivo **urls.py** do projeto e dê um duplo clique para edição.

Neste arquivo acrescentaremos a biblioteca **include** e criaremos um caminho(rota) que fará a inclusão do arquivo que **urls.py** do App_Escola que acabamos de criar.

Como dissemos no primeiro parágrafo deste tópico, quando iniciamos o servidor com o comando **runserver** o endereço que ele responde é o que chamamos de **endereço raiz** do projeto que é o <http://127.0.0.1:8000/>

Assim, nós utilizaremos esse **endereço raiz** para redirecionarmos para o arquivo **urls.py** do nosso App_Escola.

Para adicionar a biblioteca **include** altere a linha como segue:

```
from django.urls import path, include
```

E para incluirmos o arquivo **urls.py** do App_Escola acrescente a linha na sessão `urlpatterns` como segue:

```
path('', include('App_Escola.urls')),
```

O comando **path** utilizado nos arquivos de urls tem a seguinte estrutura:

path(url, view, name)

Quando no lugar da **url** utilizamos `''` estamos dizendo para responder na raiz a url, no nosso caso <http://127.0.0.1:8000/> ; no lugar da **view** que trataremos mais detalhes a frente, estamos dizendo para incluir o

Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

arquivo **urls.py** do **App_Escola** e o parâmetro **name** não utilizaremos por enquanto mas serve para nos referenciarmos à **url** por esse **name** dentro do projeto.

O arquivo **urls.py** do projeto deve ficar assim:



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows the project structure: App_User, Escola (selected), and External Libraries. The Escola folder contains: __init__.py, asgi.py, settings.py, urls.py (selected), wsgi.py, db.sqlite3, manage.py, and main.py. The code editor shows the contents of urls.py:

```
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
```

Agora trabalharemos o arquivo **urls.py** do **App_Escola**

Copie a estrutura do arquivo **urls.py** do **projeto** (Ctrl + C) e cole no arquivo **urls.py** do **App_Escola** (Ctrl + V) pois a estrutura do arquivo é a mesma.

Agora faremos as alterações necessárias no arquivo **urls.py** do **App_Escola**.

Apague as linhas

```
from django.contrib import admin e path('admin/', admin.site.urls),
```

Acrescente a linha

```
from . import views
```

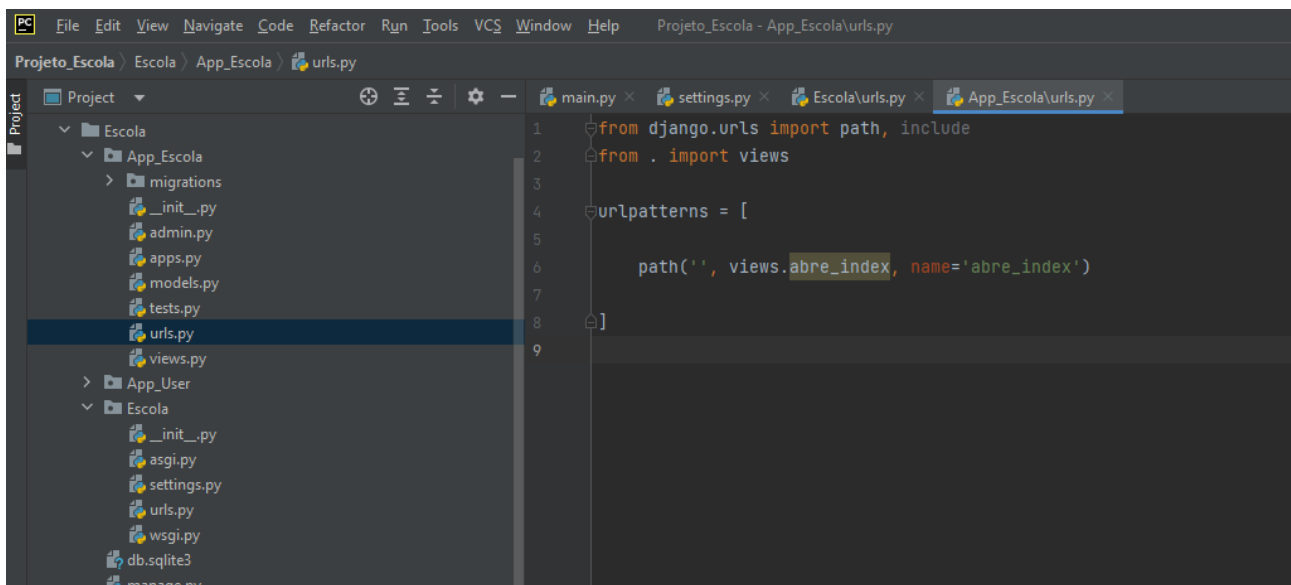
Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

Na sessão **urlpatterns** altere a linha do **path** para:

```
path('', views.abre_index, name='abre_index')
```

Conforme dito acima, quando utilizamos `''` estamos dizendo para que o sistema responda no caminho(rota) raiz. O parâmetro **views.abre_index** ainda não criamos suas funções e serão criadas sem seguida. E o parâmetro **name** chamaremos também de **abre_index**.

O arquivo **urls.py** do **App_Escola** deve ficar como segue:



1.18 – Acessando a primeira página do projeto Escola.

Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

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.

Neste arquivo acrescente as seguintes linhas de instrução:

```
def abre_index(request):  
    return render(request, 'Index.html')
```

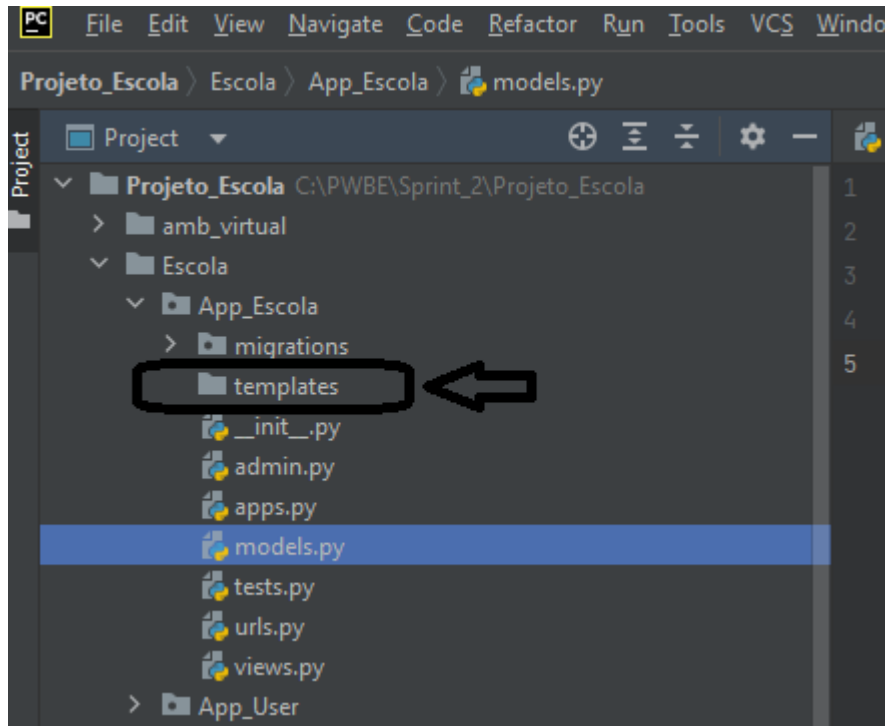
Nas linhas acima nós criamos a função **abre_index** e mandamos que seja respondido com a página **Index.html**.

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**.

Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

A estrutura de pastas deve ficar como segue:



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/'
```

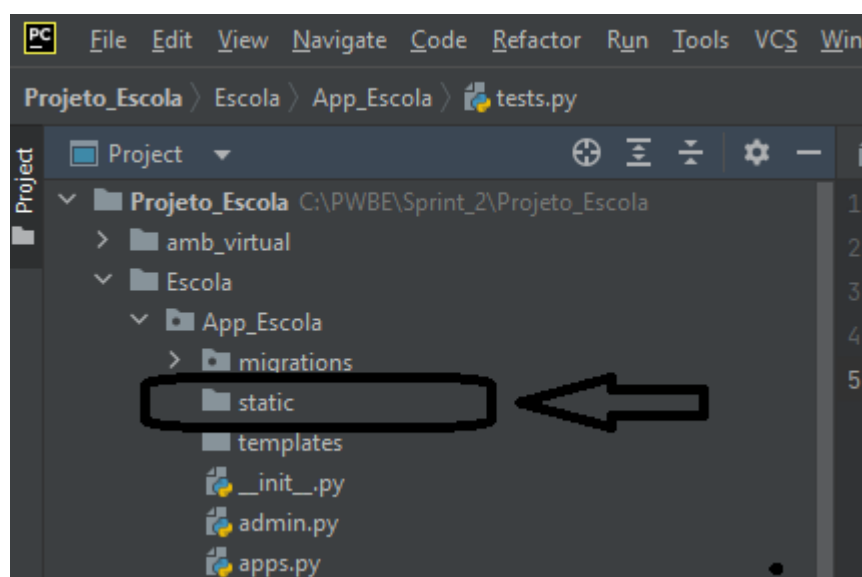
(Atenção para colocar entre duas barras e apóstrofes.)

Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

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** .

A estrutura de pastas deve ficar como segue:



Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

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**.

Agora com todos os arquivos de CSS, JS, Imagens e Html devidamente acomodados em suas pastas, vamos começar o desenvolvimento “**Back End**” do nosso projeto Escola.

Abra o arquivo **Index.html** dando um duplo clique para edição.

Como primeira linha do arquivo acrescente a instrução:

```
{% load static %}
```

Dentro da tag <head> altere a linha que faz o link com o arquivo **css** conforme abaixo:

```
<link rel="stylesheet" type="text/css" href="{% static  
'css/main.css' %}">
```

Altere também a tag que faz o link com o logo do Senai conforme segue:

```
<a href="index.html"></a>
```

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** .

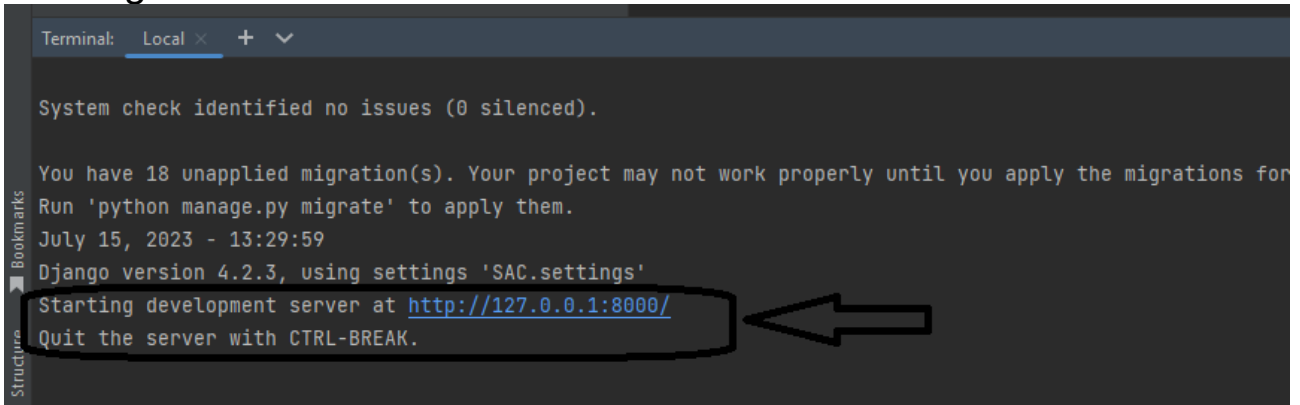
Inicie novamente o servidor digitando:



Curso Análise e Desenvolvimento de Sistemas
Programação Web Back End

```
(amb_virtual) PS C:\PWBE\Sprint_2\Projeto_Escola\Escola> python manage.py  
runserver
```

Se tudo certo até aqui, no Terminal deve ter exibido a seguinte mensagem:



```
Terminal: Local x + v  
  
System check identified no issues (0 silenced).  
  
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for  
Run 'python manage.py migrate' to apply them.  
July 15, 2023 - 13:29:59  
Django version 4.2.3, using settings 'SAC.settings'  
Starting development server at http://127.0.0.1:8000/  
Quit the server with CTRL-BREAK.
```

Abra o Browser, no endereço indicado <http://127.0.0.1:8000/> que será o endereço **raiz** do nosso projeto. Deve ser mostrado a imagem que segue:

