Criar Projeto em Django

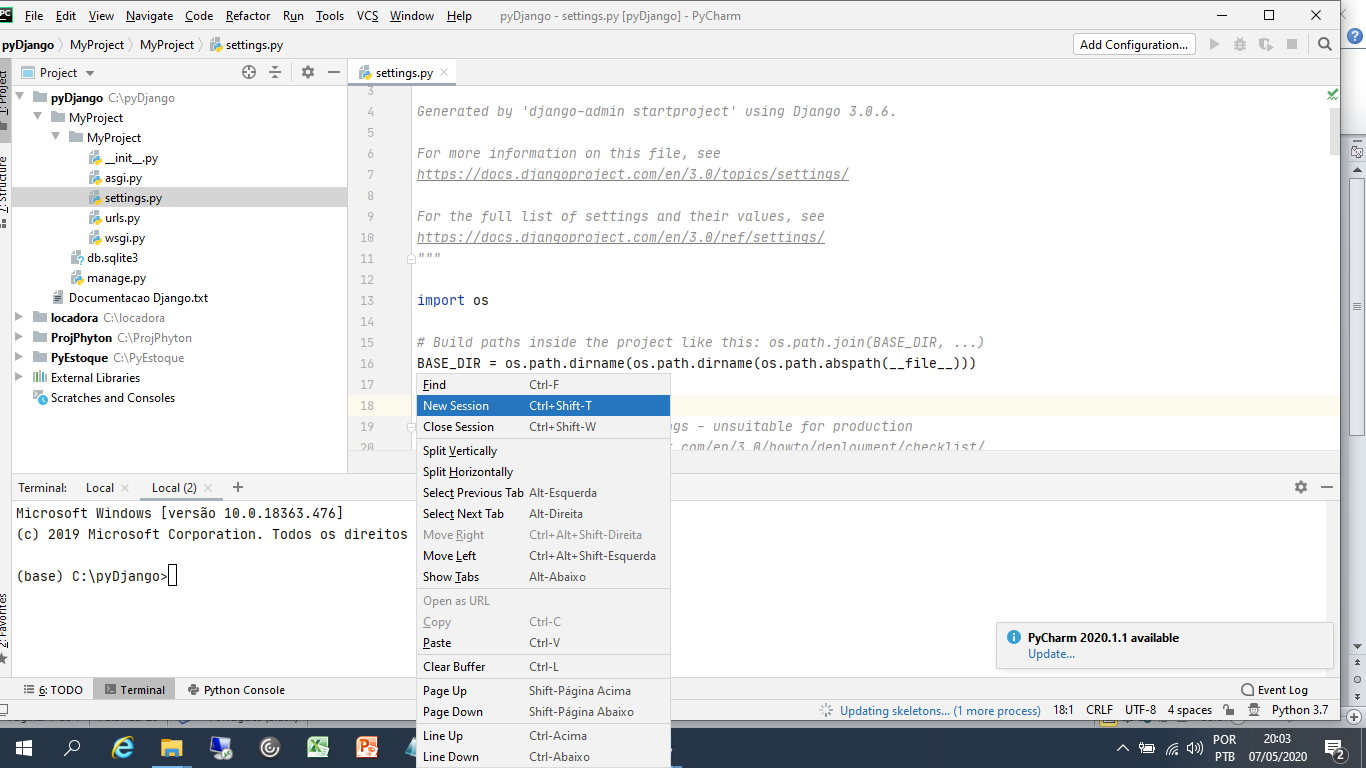
django-admin startproject MyProject

1. Ativar o servidor

python manage.py runserver

1. criar um segundo ambiente para a manipulacao dos arquivos

python manage.py migrate



1. Criar superusuario

python manage.py createsuperuser

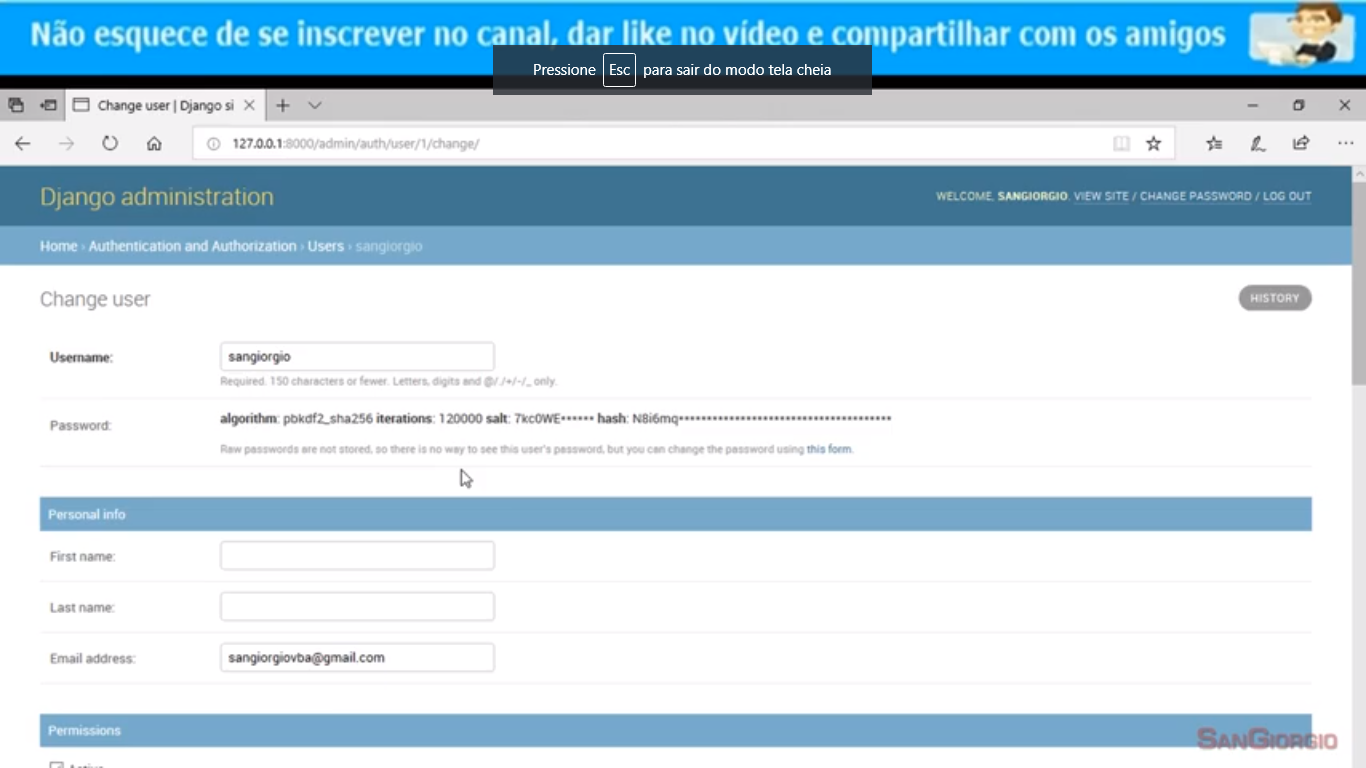
dalton (SqlLite3) e dalton1 (MySql)

[dalton\_tavares@bol.com.br](mailto:dalton_tavares@bol.com.br)

pydlt123

1. Administracao do Django

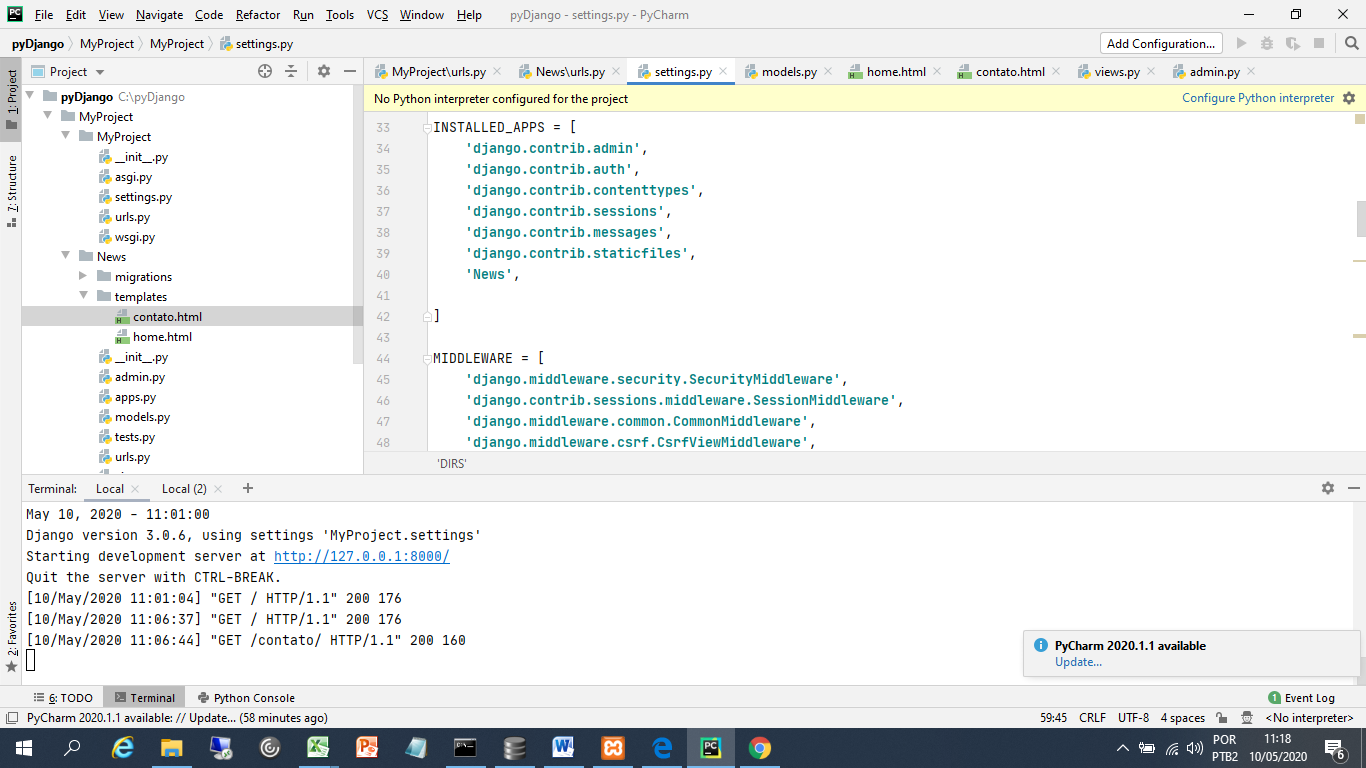
<http://127.0.0.1:8000/admin/>



1. Criar a aplicação

Python manage.py startapp News

Adiciona no settings.py do projeto

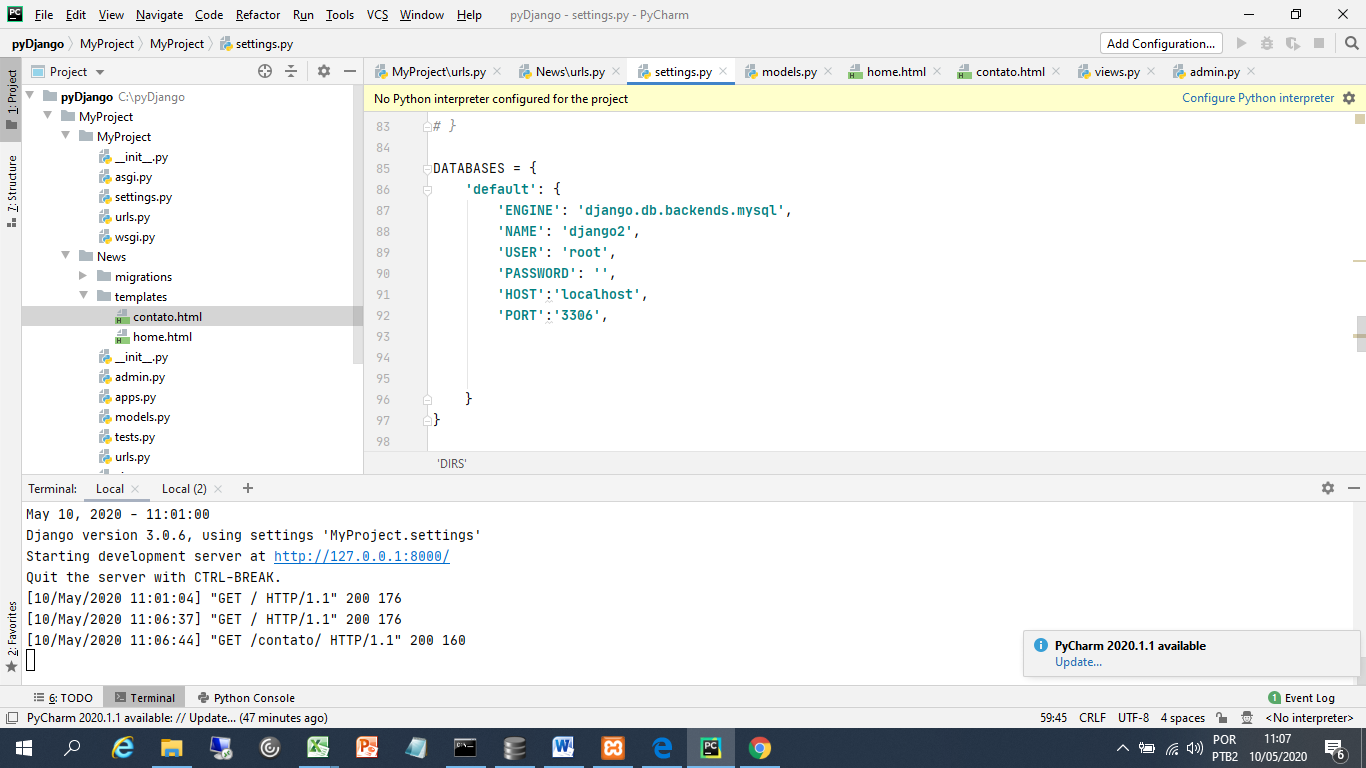


1. Instalar MySql

pip install pymysql

pip install mysqlclient

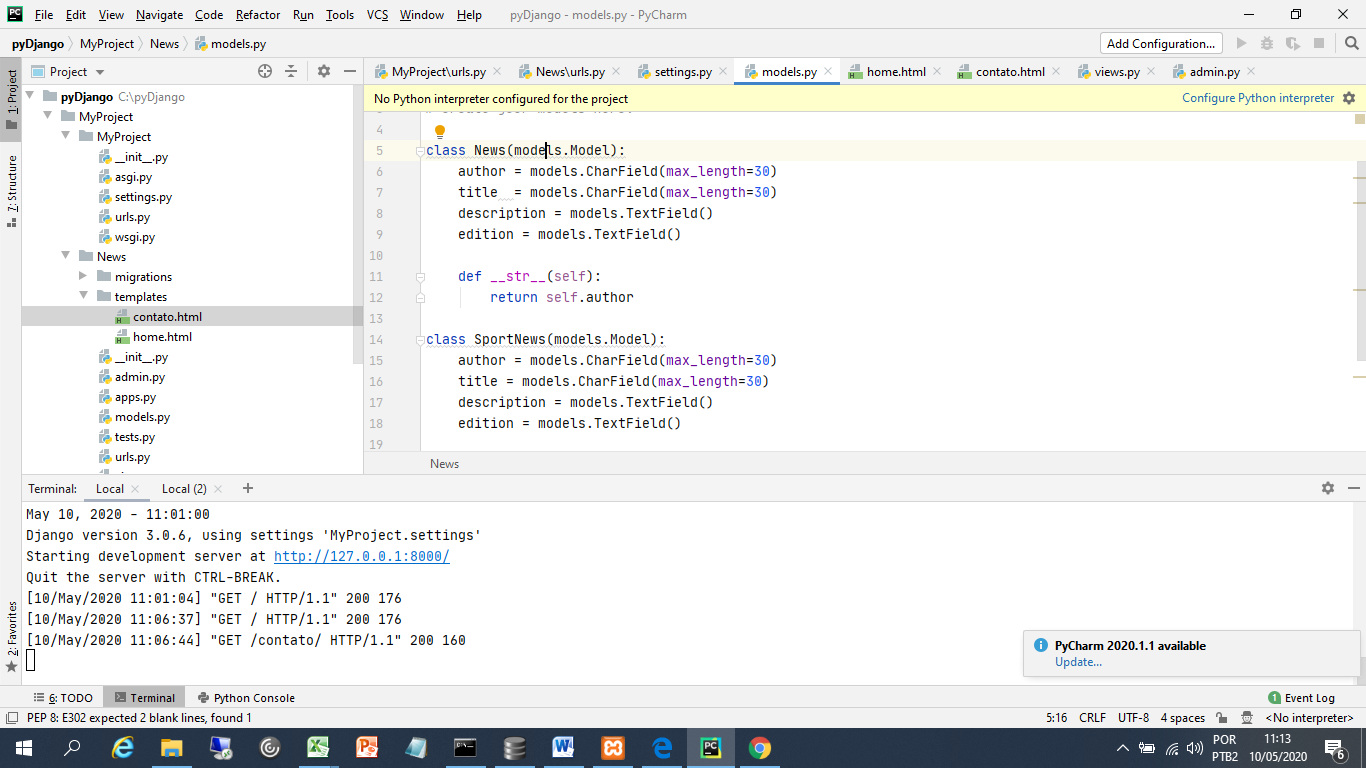
1. Configura o banco de dados no settings.py do projeto.



1. Toda alteração feita no arquivo models.py deve-se executar os seguintes comandos:

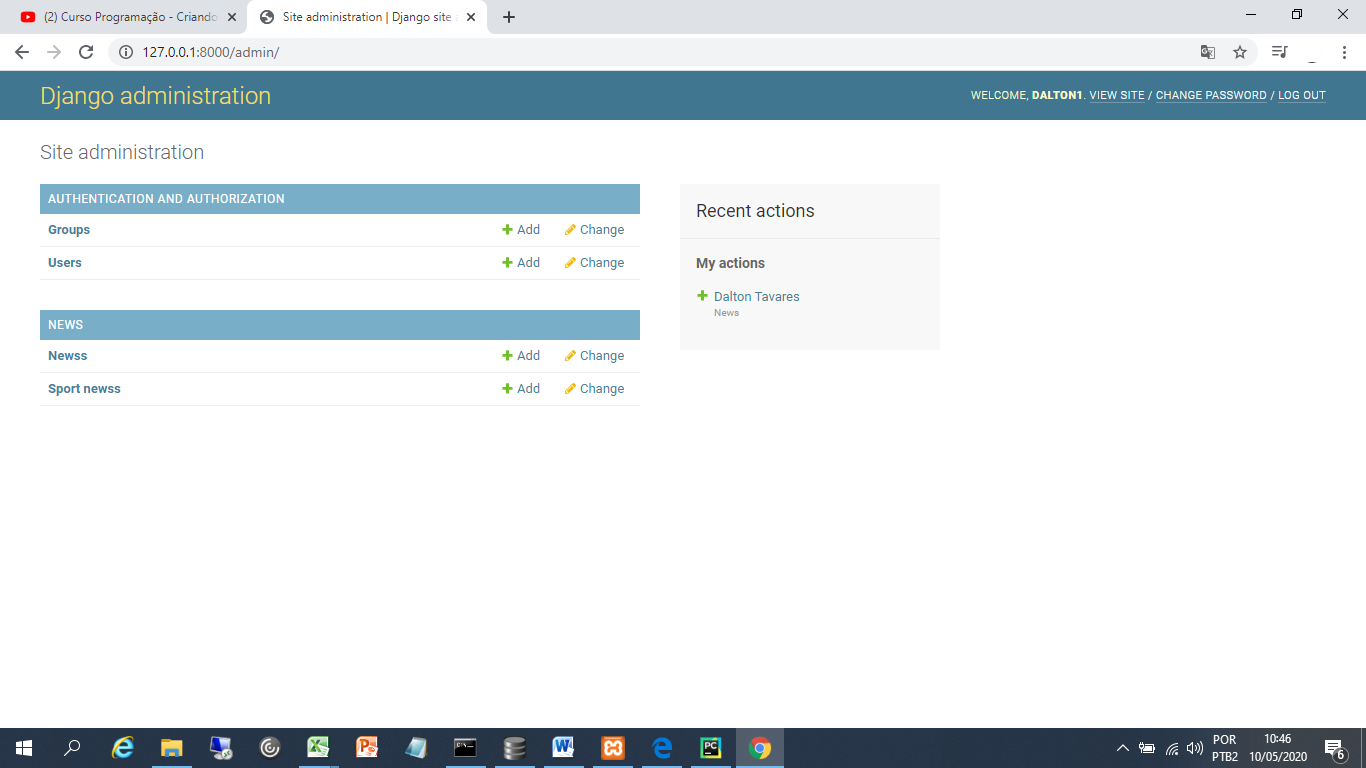
python manage.py makemigrations (Gera um log de alterações no Django)

python manage.py migrate (Leva alteração para o banco de dados)



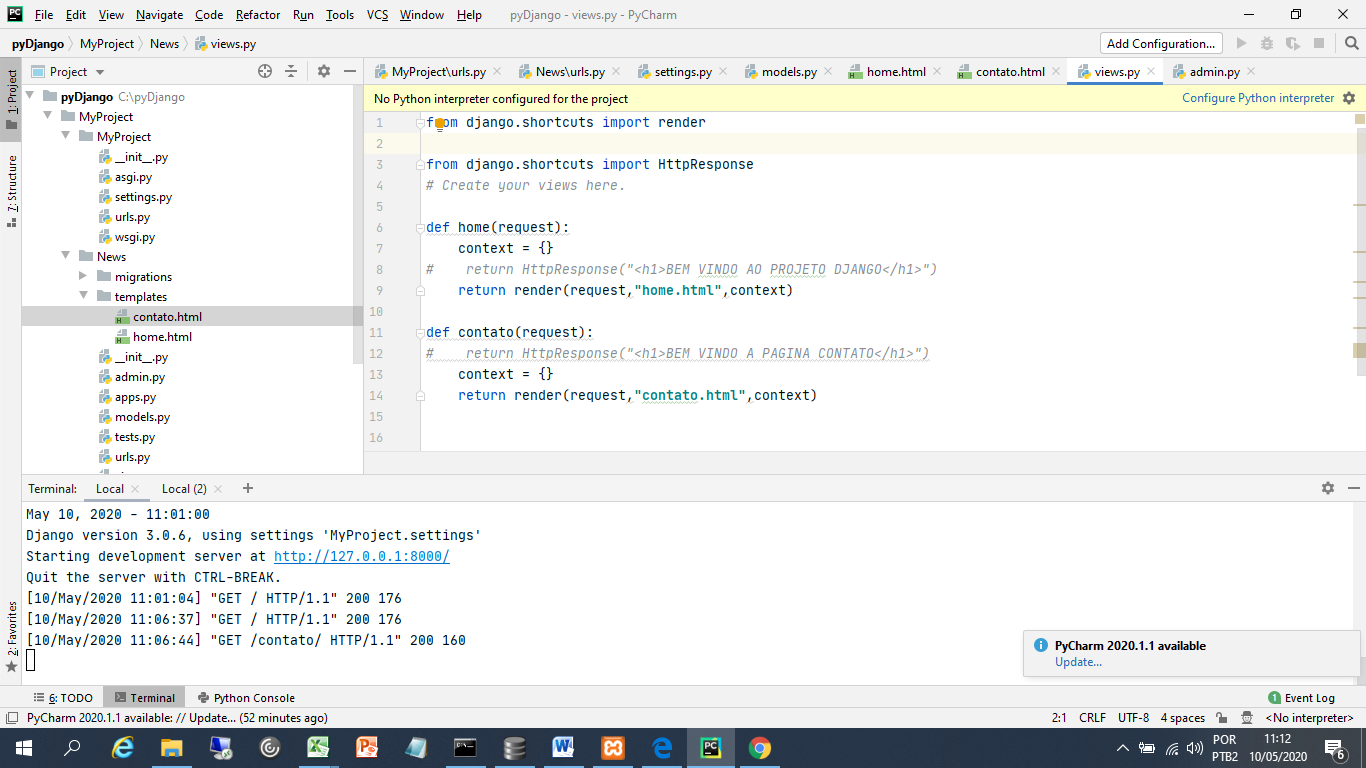
1. Após criar as classe no arquivo model.py, deve-se registra-las no arquivo admin.py do projeto. Desta forma elas ficarão disponíveis no ambiente admin para manutenção

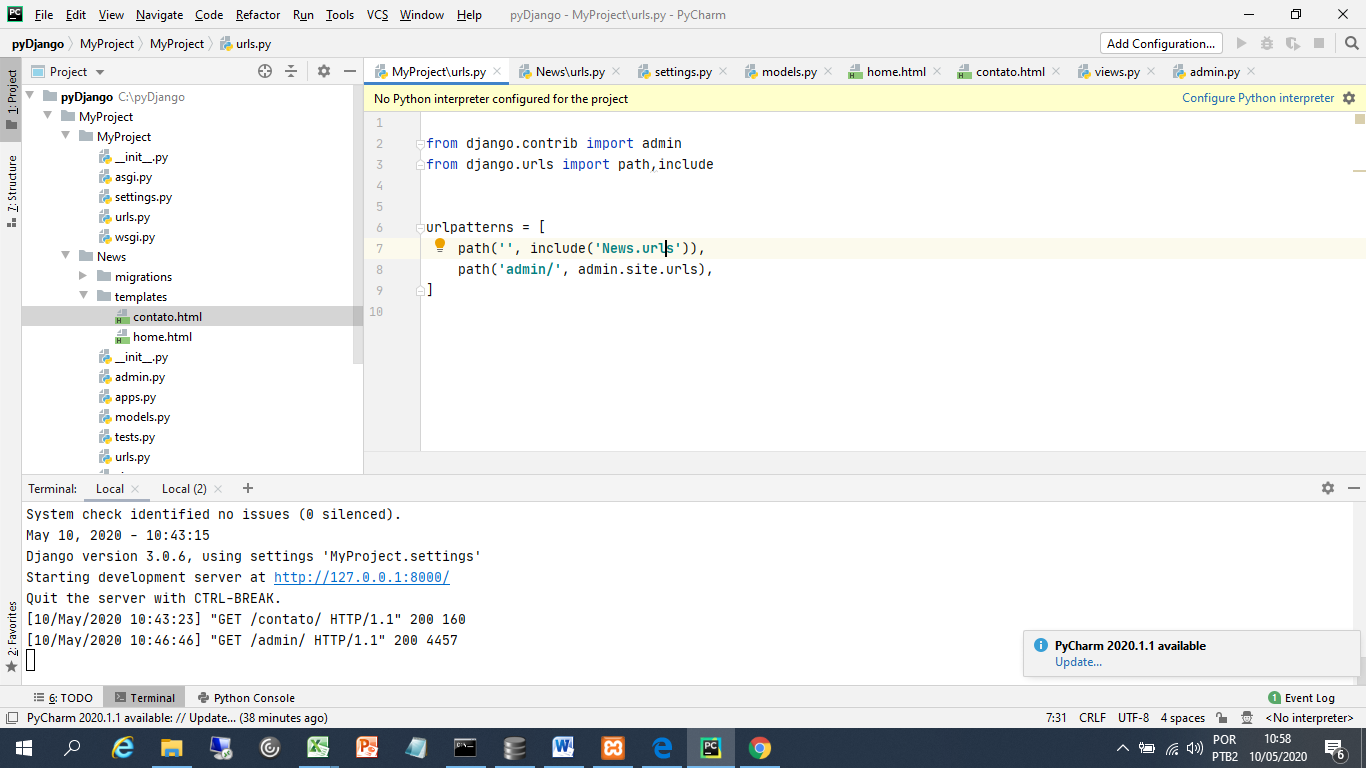
admin.site.register(News)  
admin.site.register(SportNews)

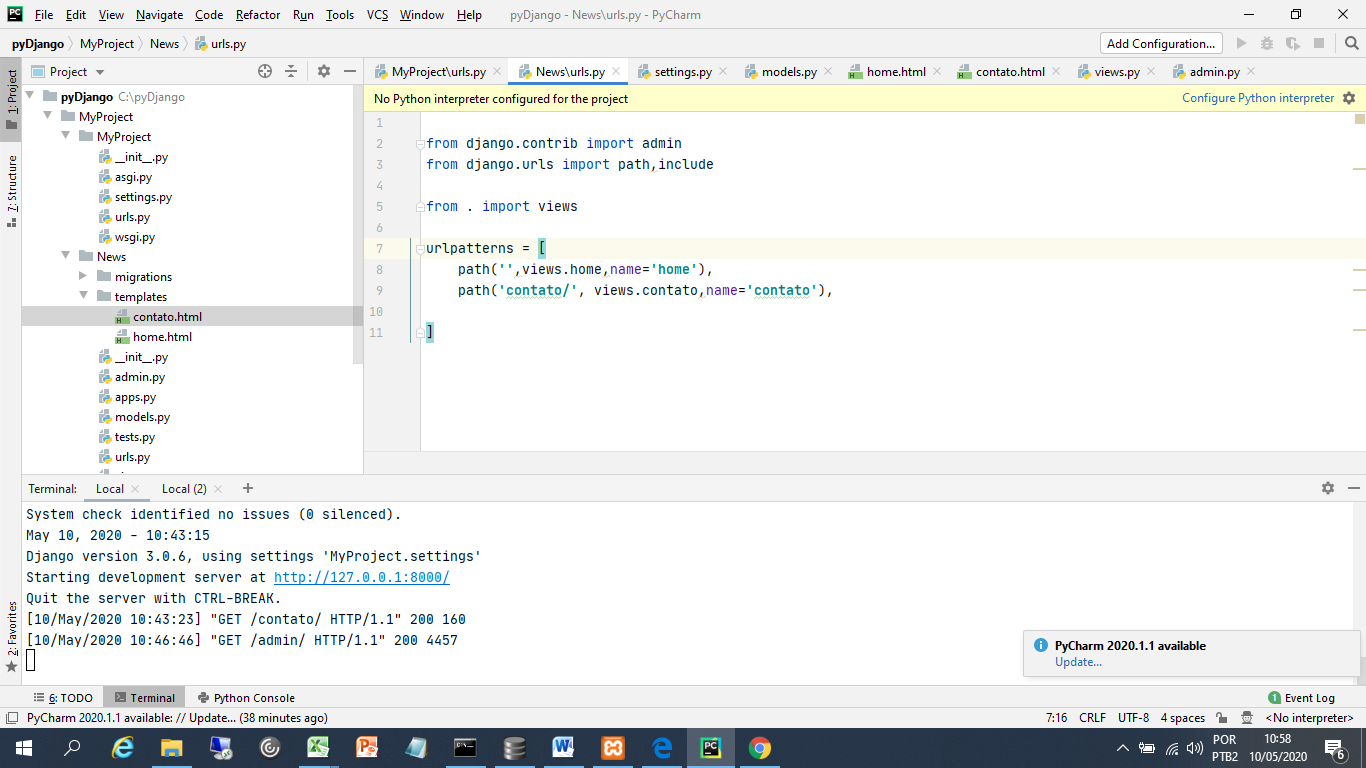


1. Quando criadas as classes no arquivo view.py (web pages), estas devem ser configuradas no arquivo urls.py do projeto, ou também no arquivo urls.py da aplicação (desde que seja transferido a ela esta função) para que ele reconheça seu endereço quando chamado pelo Browser.

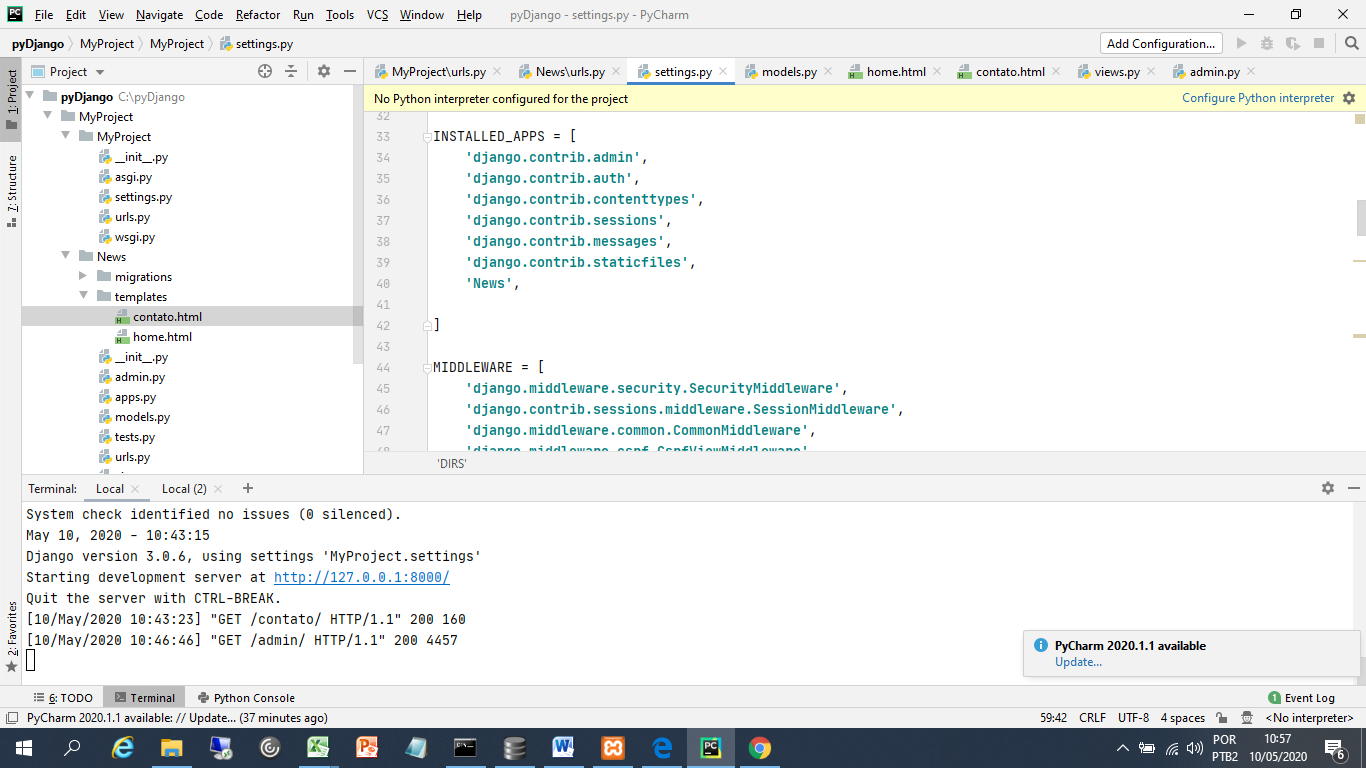
Quando a URL é chamada no Browser passa primeiro pelo urls.py do projeto e em seguida através deste é direcionada para o urls.py da aplicação. Ver caminho completo abaixo.







1. É necessário incluir a aplicação no Installed\_apps do settings.py



1. As paginas HTML devem ficar guardadas todas na pasta configurada no DIRS do arquivo settings.py, conforme figura abaixo. Quando ela for chamada pela classe HttpResponse, ela ira procurar o arquivo HTML nesta pasta.

Exemplo: return render(request,"home.html",context), o Django ira buscar o arquivo home.html nesta pasta.

1. Para o sistema reconhecer as homepages HTML, foi criada uma pasta chamada ‘template” dentro da aplicação e configurada sua localização no arquivo settings.py do projeto na cláusila ‘DIRS’:

Após chamar a url no browser o sistema faz o seguinte percurso:

Urls.py do projeto->

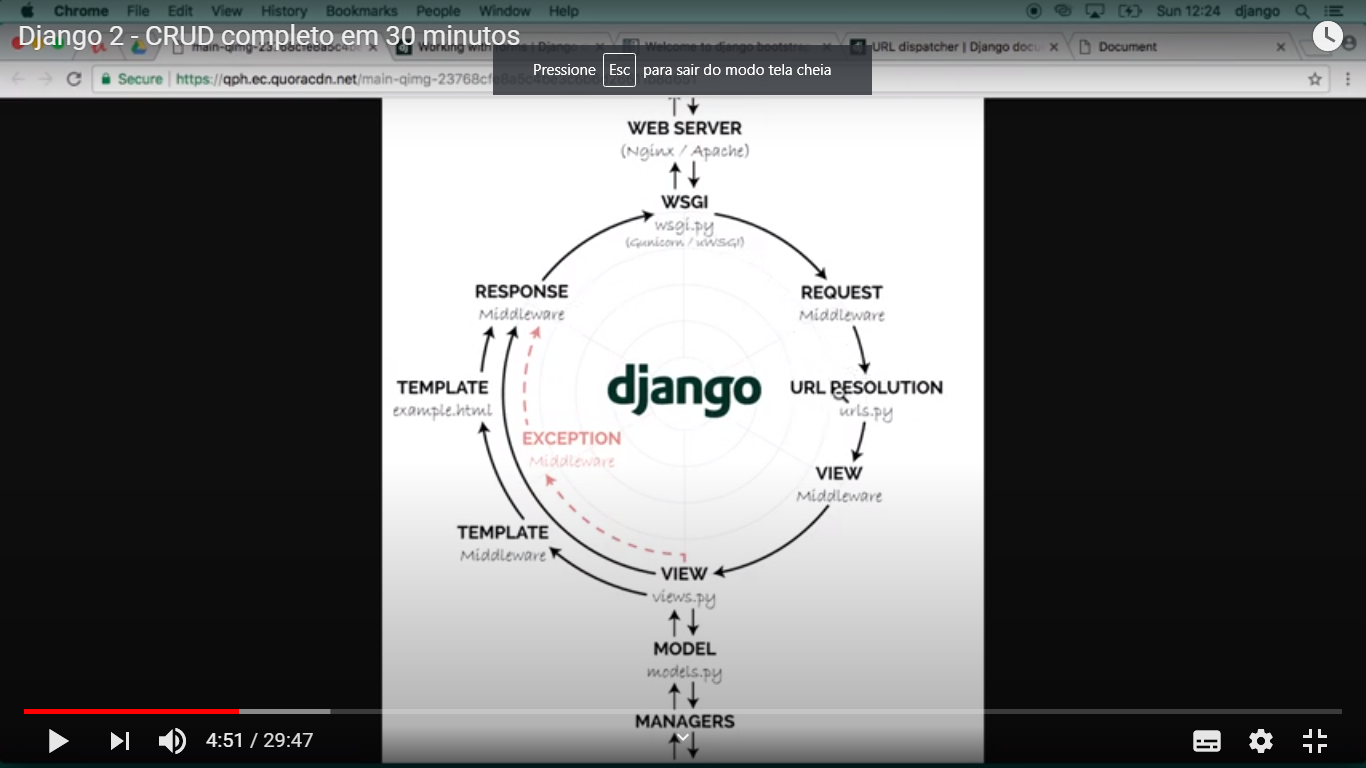
urls.py da aplicação->

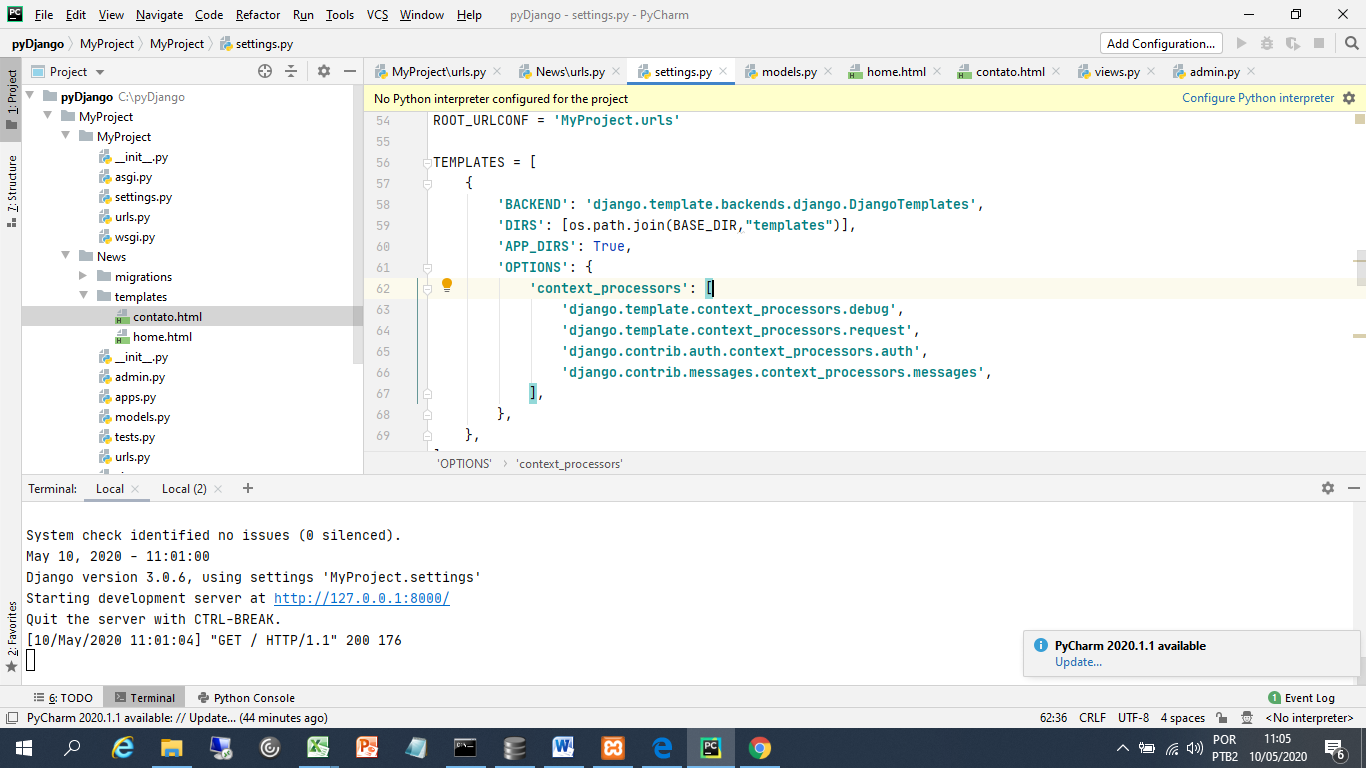
arquivo view.py da aplicação->

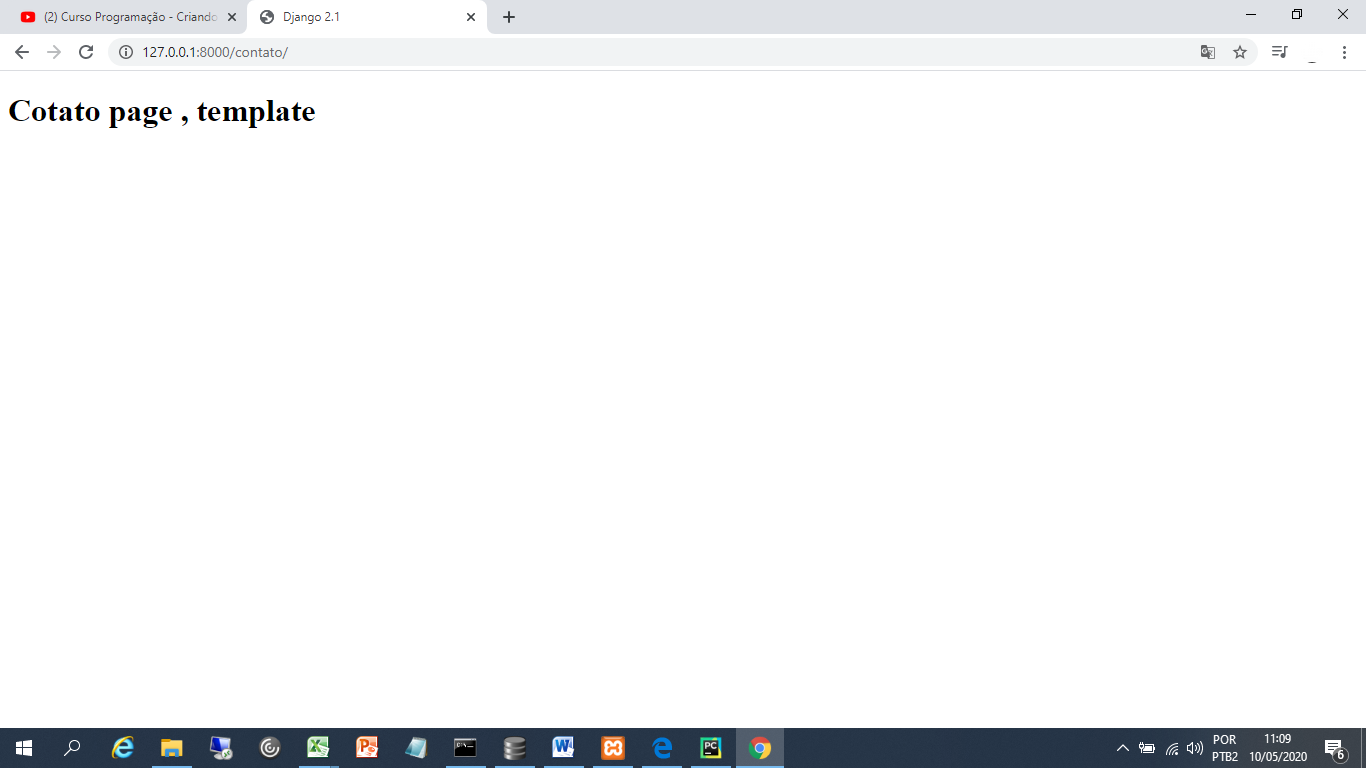
Vai ao settings.py do projeto (Templates->DIRS e busca o local do html)->

Chama o HTML.

O HTML deve estar dentro da pasta Templates.





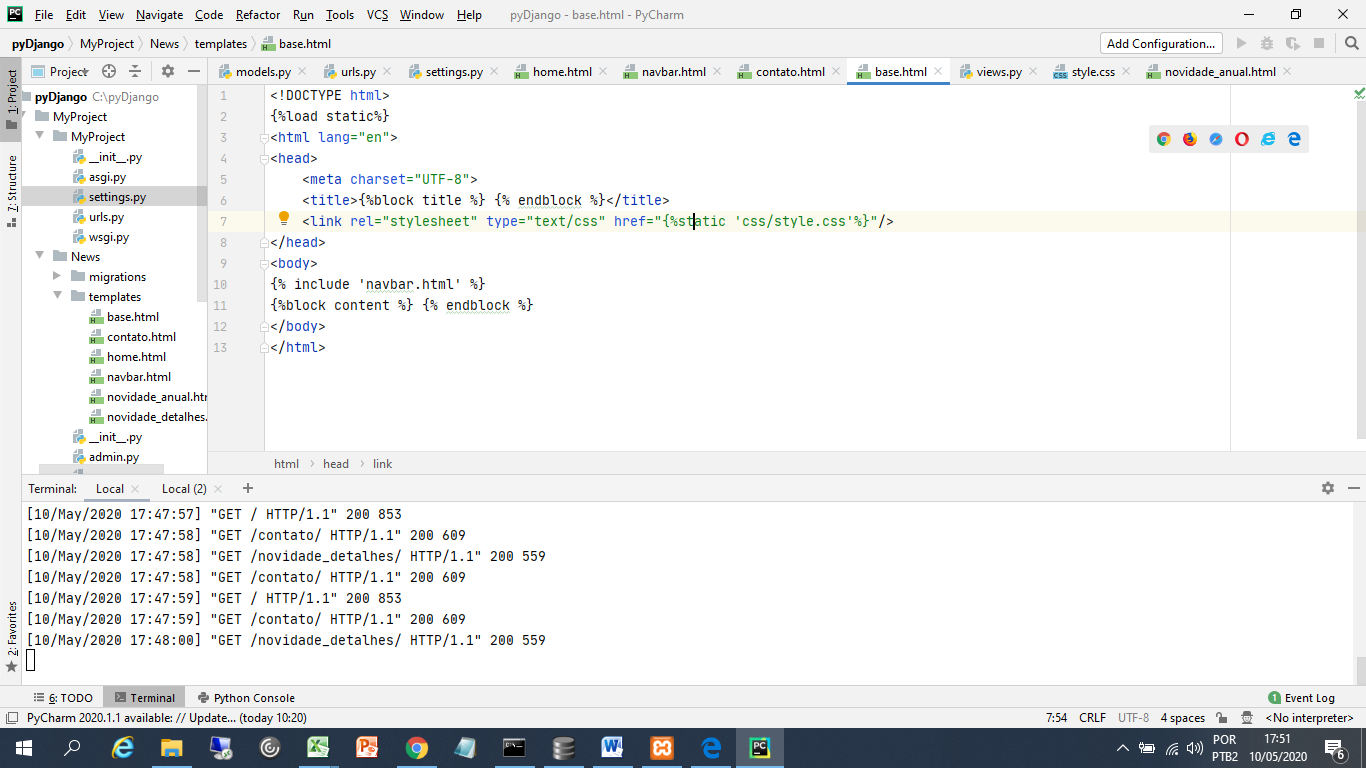


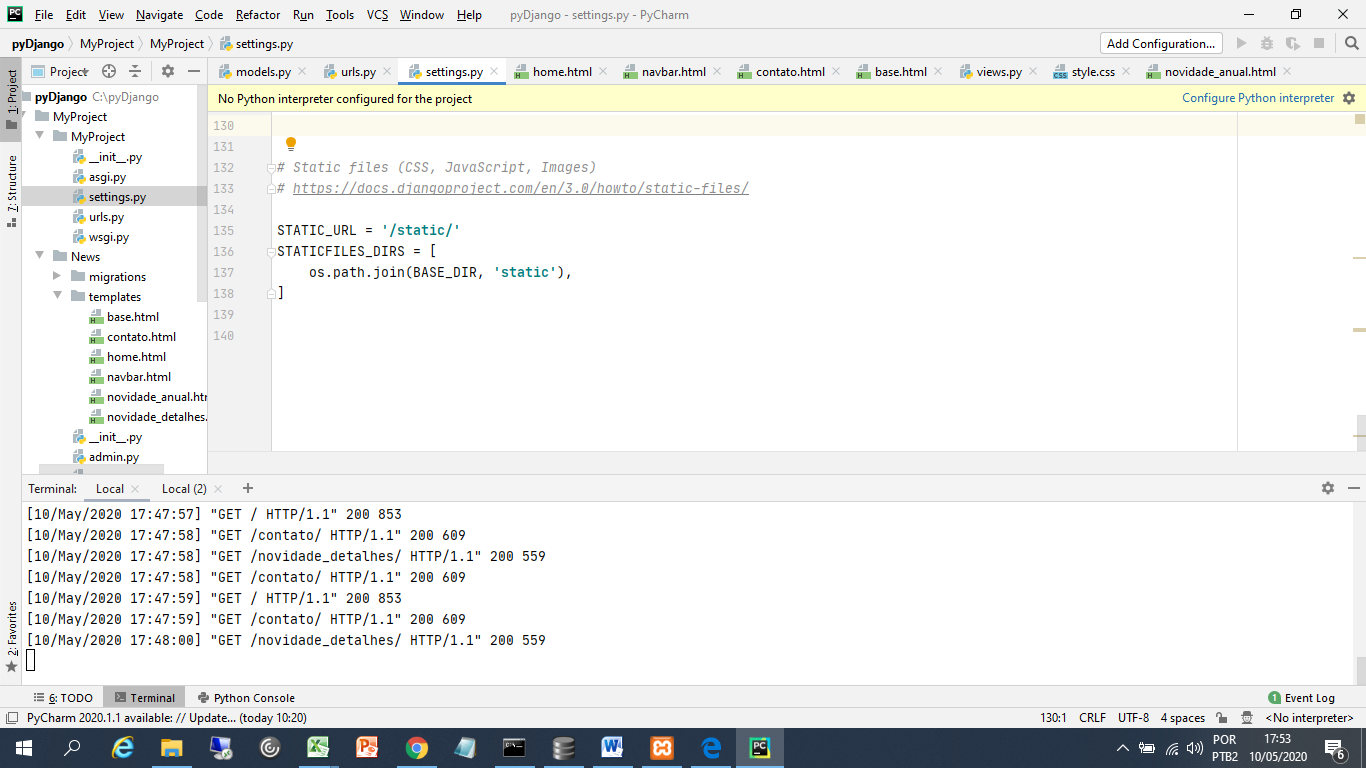
1. A pasta static onde ficam os CSS e Bootstrap é criada no mesmo nível da pasta do projeto e não no nível da aplicação; A pasta do projeto é a pasta base para o Django localizar os arquivos.

Arquivo settings.py:

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))

1. Quando adicionado o CSS, foi adicionado o diretório style/css e o mesmo foi configurado no settings.py (seção STATICFILES\_DIRS)





Para utilizar o bootstrap nos formulários utilizar a biblioteca django bootstrap

<https://django-bootstrap-form.readthedocs.io/en/latest/>

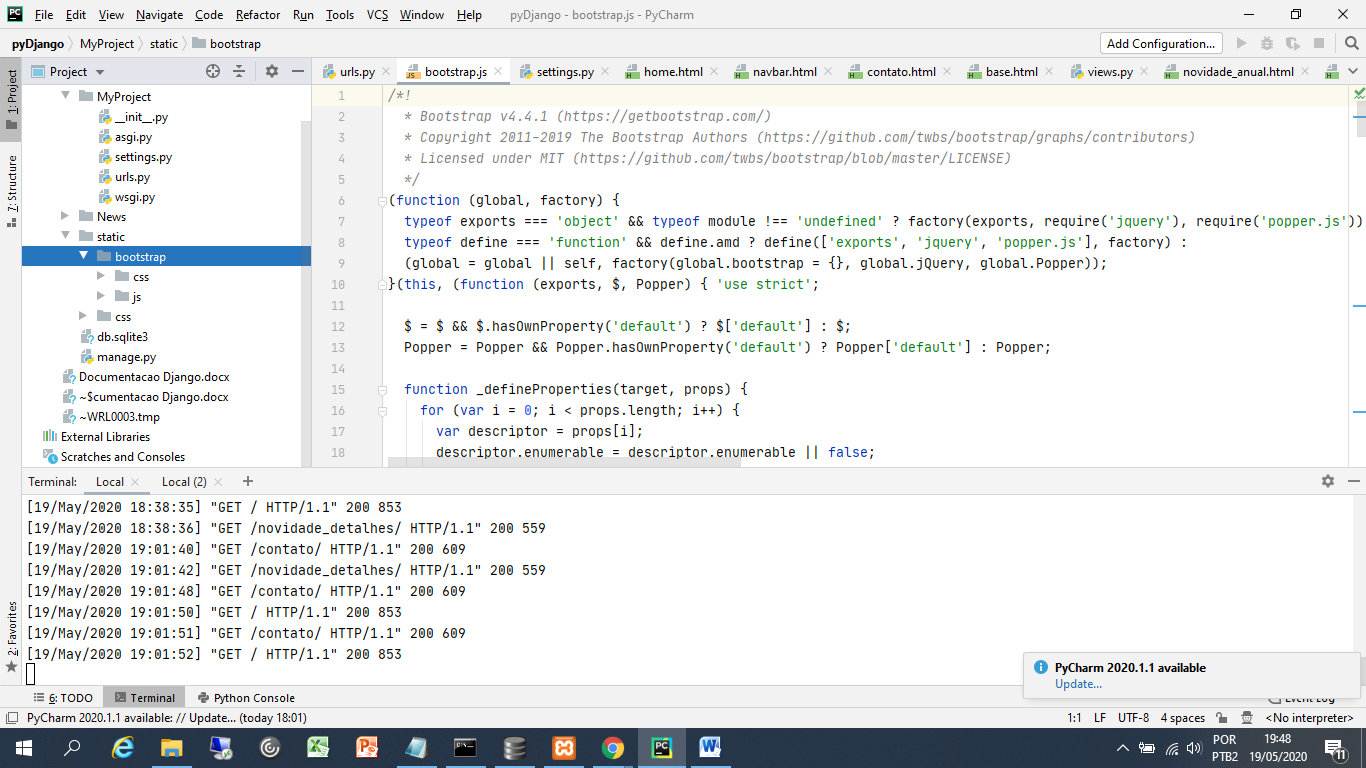
1. A API que permite o manuseio da base de dados sem que se escreva SQL no arquivo view.py é a

# QuerySet API reference[¶](https://docs.djangoproject.com/en/3.0/ref/models/querysets/" \l "queryset-api-reference" \o "Permalink to this headline)

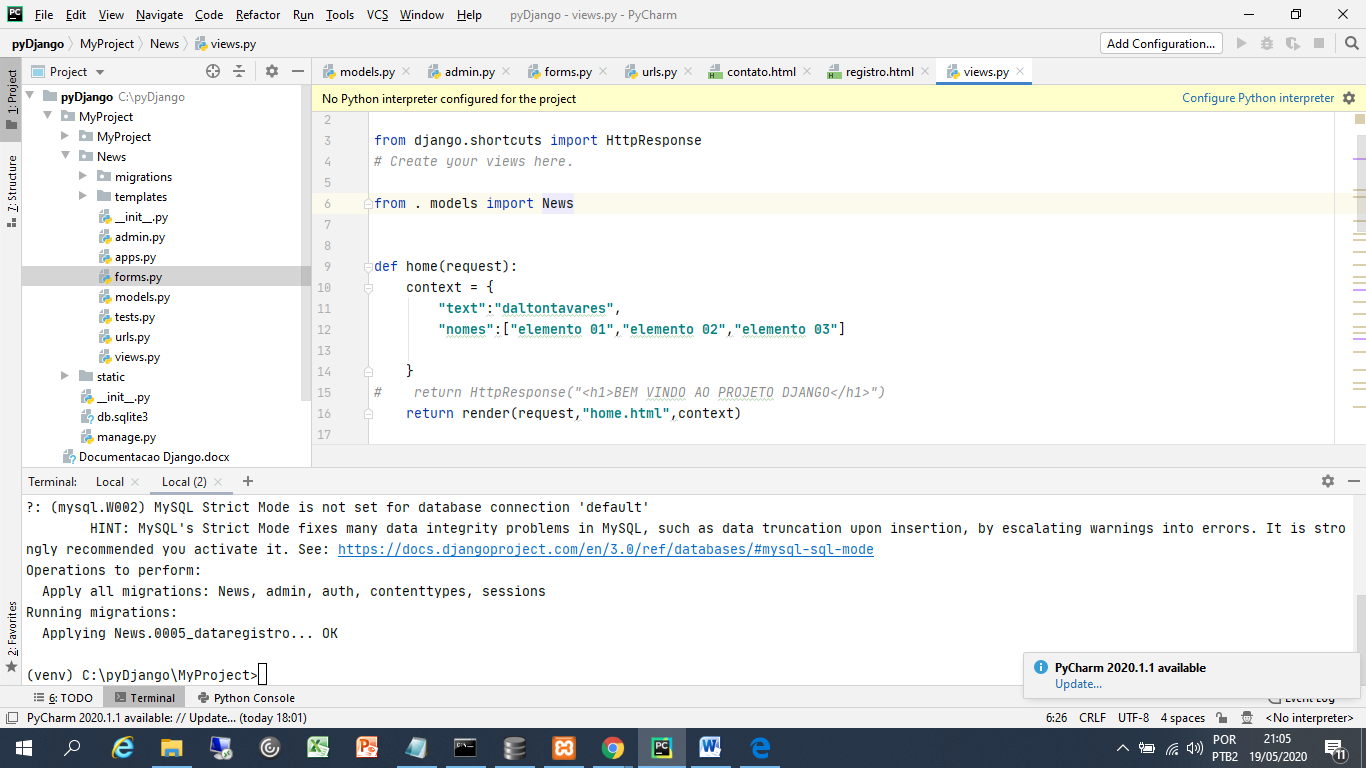
1. Para linkar uma pagina Web que deseja abrir, utilizar a sintaxe "{%url 'home'%}" onde ’home’, ‘contato’,’novidades\_detalhes’ são os nomes das paginas definidas no arquivo urls.py;

<nav>  
  
 <ul>  
  
 <li> <a href="{%url 'home'%}">Home</a> </li>  
 <li> <a href="{%url 'contato'%}">Contato</a> </li>  
 <li> <a href="{%url 'novidade\_detalhes'%}">Novidade detalhes</a> </li>  
  
  
  
 </ul>  
  
</nav>

1. Fazer o download do Bootstrap. Renomear a pasta para bootstrap e adiciona-lo na pasta static;



1. Para criação de formulários foi criada o arquivo forms.py dentro da aplicacao.



Cria uma classe de formulário no arquivo forms.py

Configura o arquivo no arquivo view.py (importa o formulário do forms.py)

Define no contexto da classe do view.py e chama o formulário

1. Validacao para update e insert

form = RegistroForm(request.POST or None)   
## Verifica se receberu algum submit. Caso nao tenha recebido, form.is\_valid() não é valido

Com isso podemos direcionar para a pagina que desejarmos, ou de inclusão oua a que lista todos os usuários

if form.is\_valid():

1. Comandos do Git

Git init

Cria o arquivo .gitignore para selecionar os arquivos que não deverão subir para o Git. Criar no mesmo nível do projeto e não da aplicação.

Levar os arquivos para o Git:

Git add .

git commit -m "initial commit" leva os arquivos para o Git

git status – mostra tudo que é preciso fazer commit;

Extrai em um arquivo todas as bibliotecas adicionadas com o pip

pip freeze >requirements-dev.txt

Acessar prompt do DOS como administrador

Procurar o diretório onde tem o arquivo netrc

Setar a variável HOMEDRIVE diretamente no Prompt do DOS (Como Administrador)

set HOMEDRIVE= C:\Program Files\heroku\client\node\_modules\netrc-parser\lib

set HOMEDRIVE=C:\Program Files\heroku\client\node\_modules\netrc-parser\lib

https://myprojectspy.herokuapp.com/ | <https://git.heroku.com/myprojectspy.git>

<https://myprojectspy.herokuapp.com/News/templates/login.html>

git push heroku master –force

IMPORTANTE: Ao criar o arquivo Procfile exigido pelo Heroku, este não tem extensão. Quando cirado no PyCharm as vezes cria a extensão .txt e não é pra te-la.

