

Django

Projeto Time – Banco de Dados



UNINASSAU

Profª. Esp. Sônia Gomes de Oliveira

PAULISTA - 2024



Criando a Virtualenv

Criando a virtualenv

Crie uma pasta no seu computador



Entre na pasta, através do prompt

```
C:\>cd django
```

Instale a virtualenv

```
C:\django>pip install virtualenv
```

Crie uma virtualenv

```
C:\django>virtualenv venv
```



Outra alternativa caso a anterior não funcione:

```
python -m venv venv
```



Ativando a Virtualenv

Ativando a VENV

Entre na virtualenv

```
C:\django>cd venv
```

Entre na pasta de Scripts

```
C:\django\venv>cd Scripts
```

Ative a virtualenv

```
C:\django\venv\Scripts>activate
```

Ativada!

```
(venv) C:\django\venv\Scripts>
```

ATTENTION

Após a ativação, utilize o comando **cd..** para sair da pasta de Scripts



Instalando o pacote DJANGO

Instalando o Django

Instale o Django

```
(venv) C:\django\venv>pip install Django
```

Saia da pasta venv

```
(venv) C:\django\venv>cd ..
```

Crie as configurações
padrões do Django

```
(venv) C:\django>django-admin startproject sistema .
```

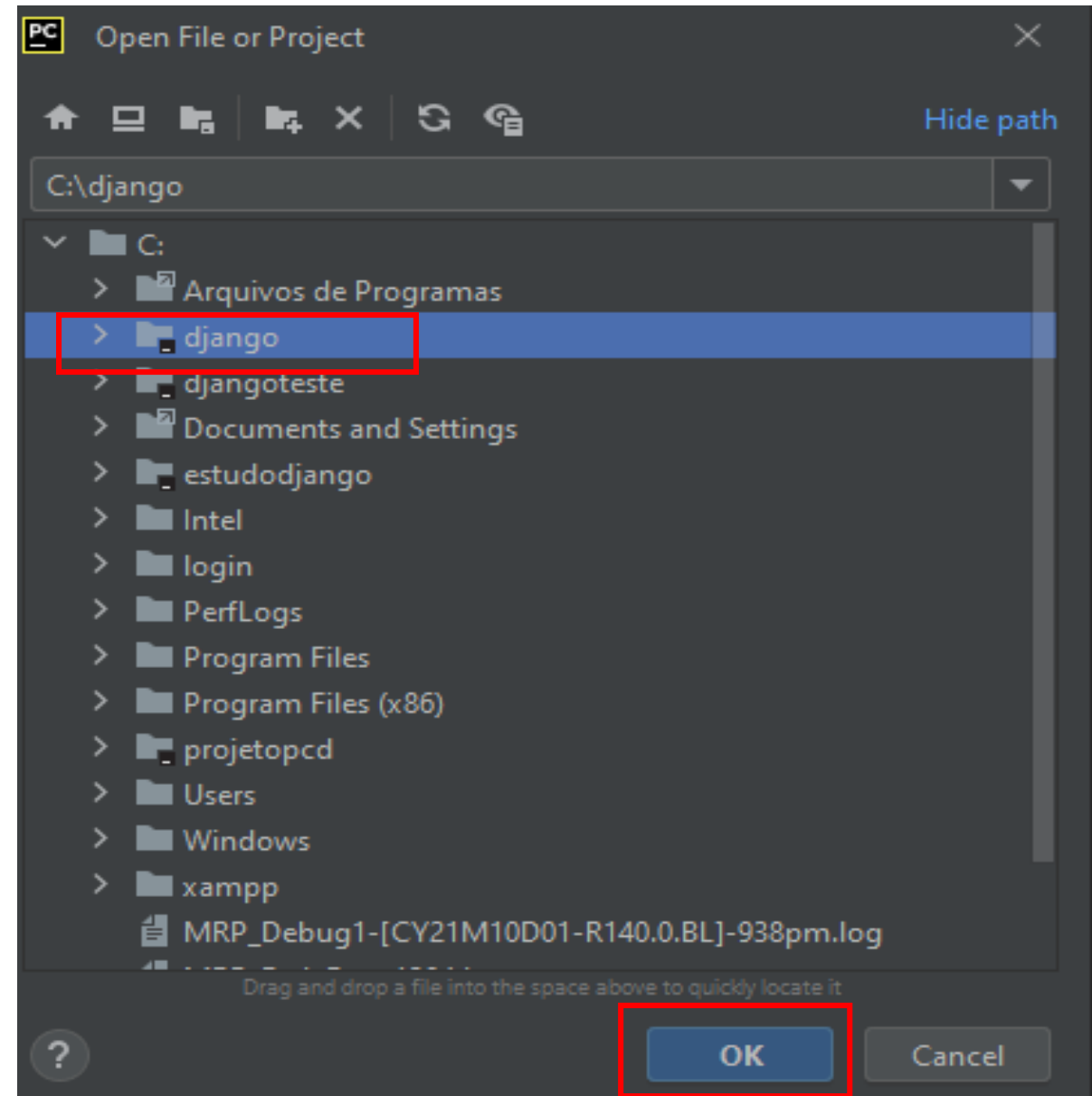
Preste atenção!



Deve instalar o django dentro do ambiente virtual (não instale diretamente no sistema operacional).

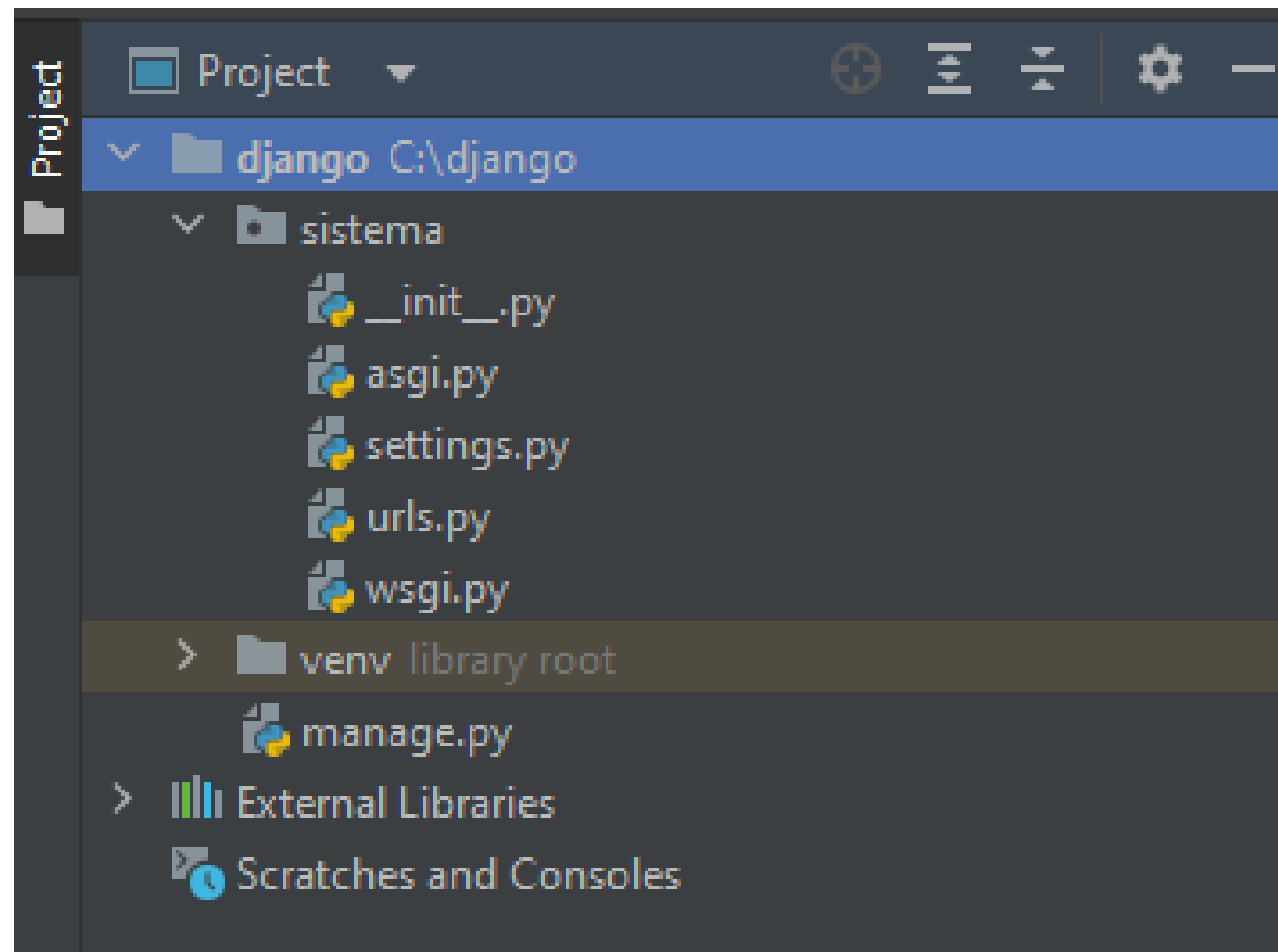
Entre no Ambiente de Desenvolvimento

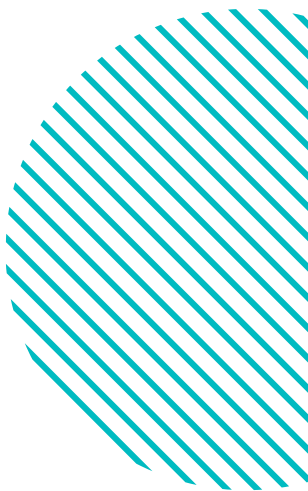

Entre no **Pycharm** e selecione a pasta que foi criada



Resultado esperado

Resultado esperado





Visualizando página no
Servidor

Verificando página web

Execute o servidor →

```
(venv) C:\django>python manage.py runserver
```

C:\ Prompt de Comando - python manage.py runserver

```
(venv) C:\django>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

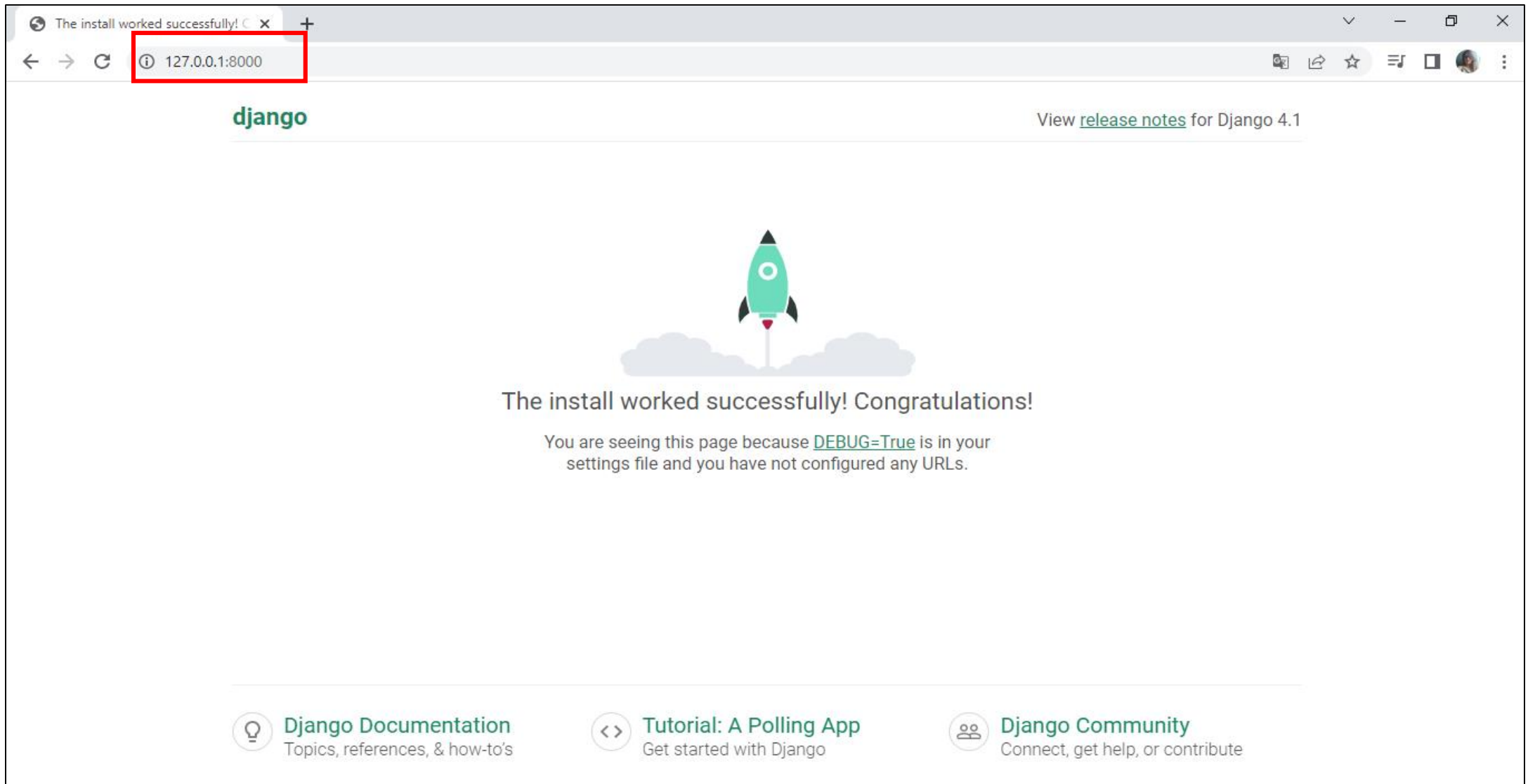
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work prop
e migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
November 27, 2022 - 02:30:28
Django version 4.1.3, using settings 'sistema.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

URL web

Pegue essa URL e digite na barra do navegador da sua preferência.

Resultado esperado!



Saindo do Servidor

Para sair do servidor aperte: CTRL + C

0% Prompt de Comando - python manage.py runserver

```
(venv) C:\django>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly
without these migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
November 27, 2022 - 02:30:28
Django version 4.1.3, using settings 'sistema.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Estamos dentro do comando **runserver**



Criando a primeira aplicação **MTV**

Models, Templates e Views

Criando a primeira aplicação

Digite no PROMPT dentro da pasta django: `python manage.py startapp futebol`



Prompt de Comando

```
(venv) C:\django>django-admin startapp futebol
```

```
(venv) C:\django>_
```

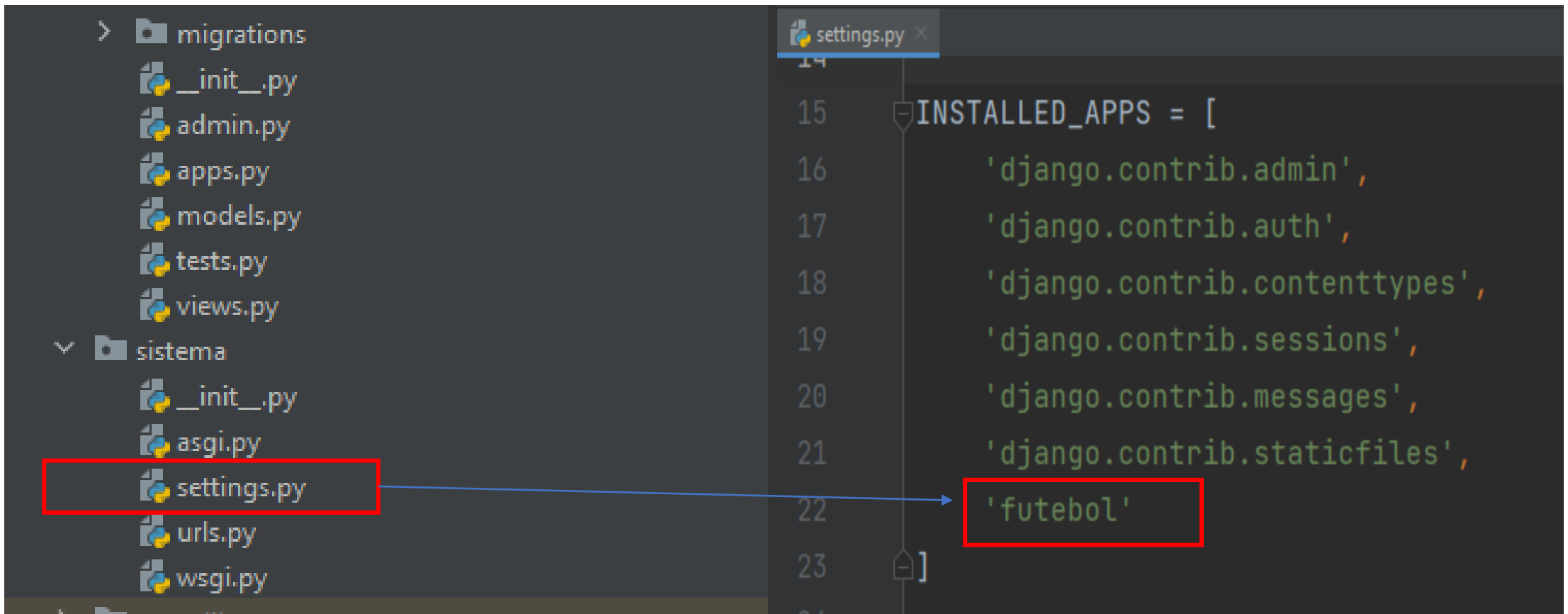


Registrando o App

Models, Templates e Views

Registrando o App cadastro

No Pycharm entre no arquivo de configurações do Django: **settings.py**





Criando Tabelas no Django

Criando tabelas no Django

Agora vamos criar as tabelas padrões do Django, comando:

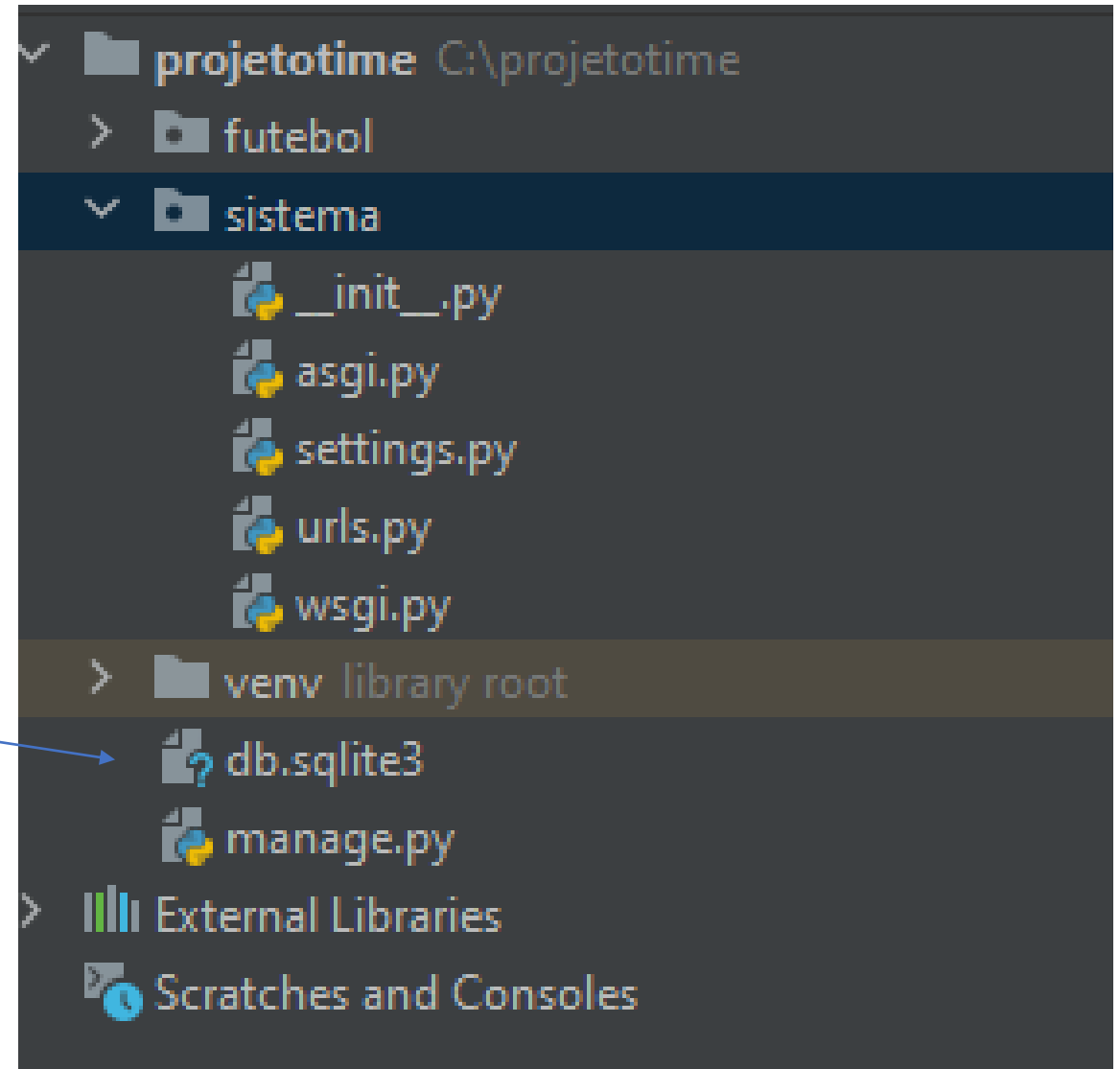
`python manage.py migrate`

```
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
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  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

(venv) C:\django>
```

Verificando se foi criado

Banco de dados
conectado

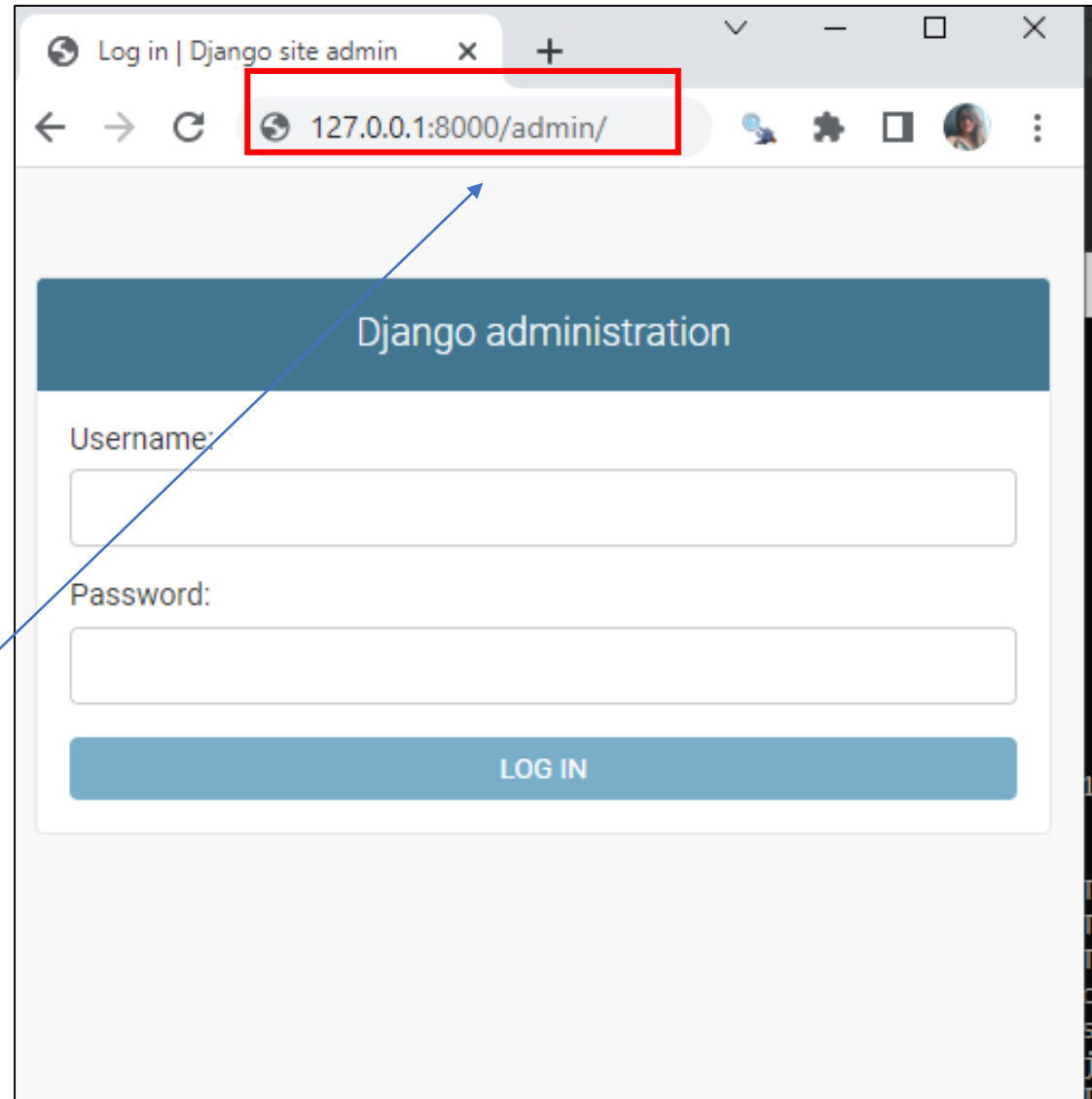


Verificando se foi criado

Vamos verificar se as tabelas foram criadas.

```
python manage.py  
runserver
```

Coloque o `/admin` para ele encontrar o administrador



Tudo ok!



Criando o SUPER USUÁRIO

Criando o Super-Usuário

Vamos criar o administrador, digite o comando:

`python manage.py createsuperuser`

username: **admin**

Email:

Password: **12345**

Password: **12345**

Validate: **y**

```

Prompt de Comando

(venv)C:\django>python manage.py createsuperuser
Username (leave blank to use 'compaq'): admin
Email address:
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

(venv) C:\django>
```

Verificando acesso

Username é = admin

Senha é = 1234

Django administration

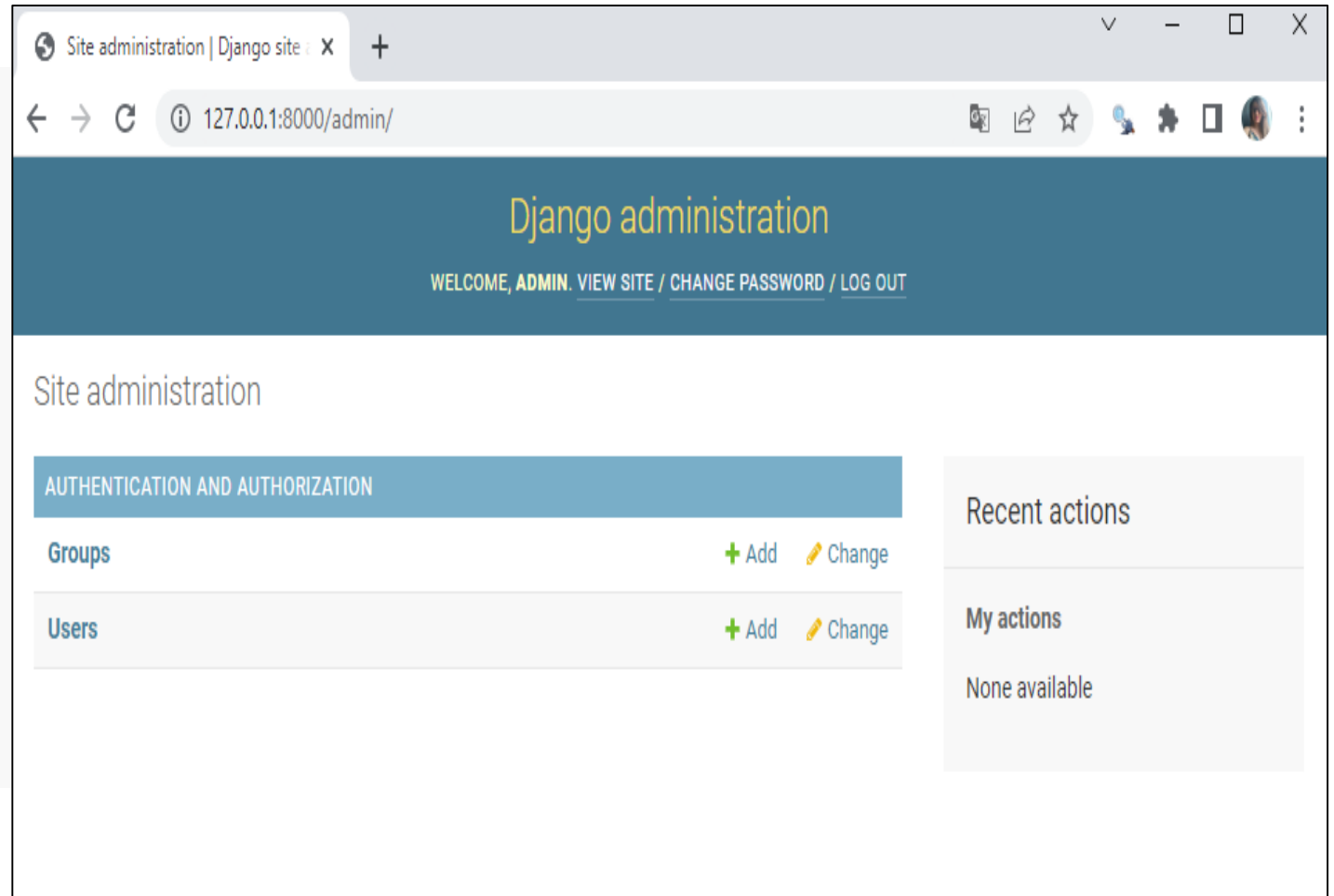
Username:

admin

Password:

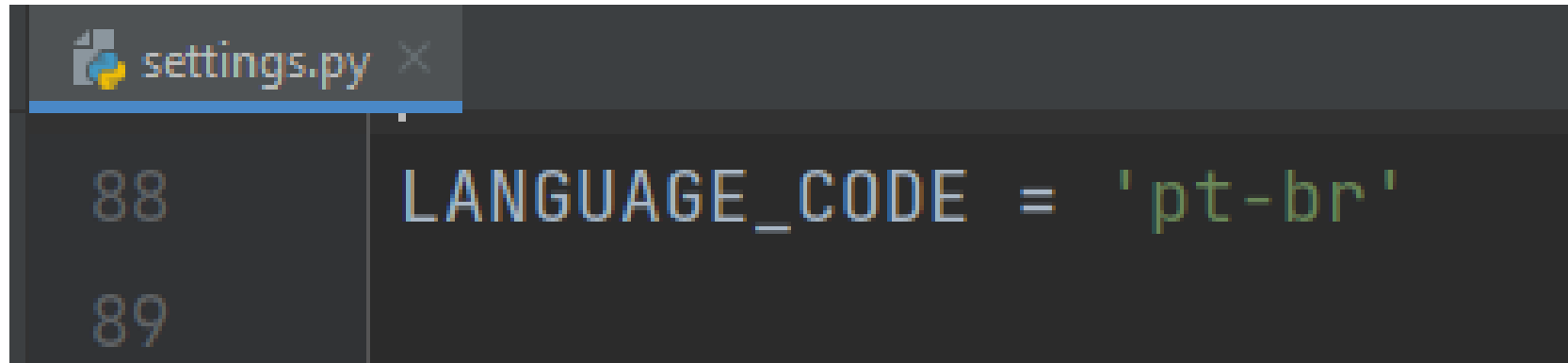
....

Log in



Mudando o idioma do Projeto

Entre na pasta **sistema** e clique no arquivo **settings**, modifique o idioma no comando **LANGUAGE_CODE**



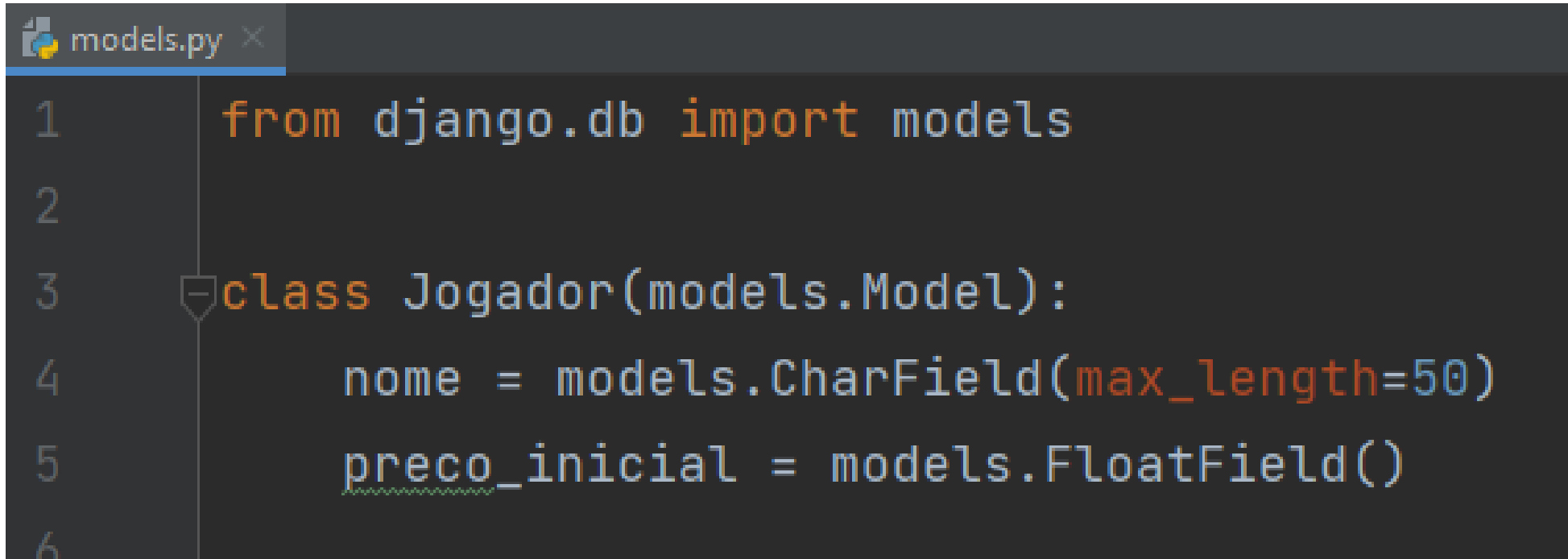
```
settings.py x
88 LANGUAGE_CODE = 'pt-br'
89
```



Criando Modelo de Tabela

Criando a tabela

Entre no app **futebol** e clique no arquivo **models** e faça a configuração da tabela




```
models.py ×  
1  from django.db import models  
2  
3  class Jogador(models.Model):  
4      nome = models.CharField(max_length=50)  
5      preco_inicial = models.FloatField()  
6
```

Criando a tabela

Para criar a tabela feita manualmente, digite:
python manage.py makemigrations

Tabela criada

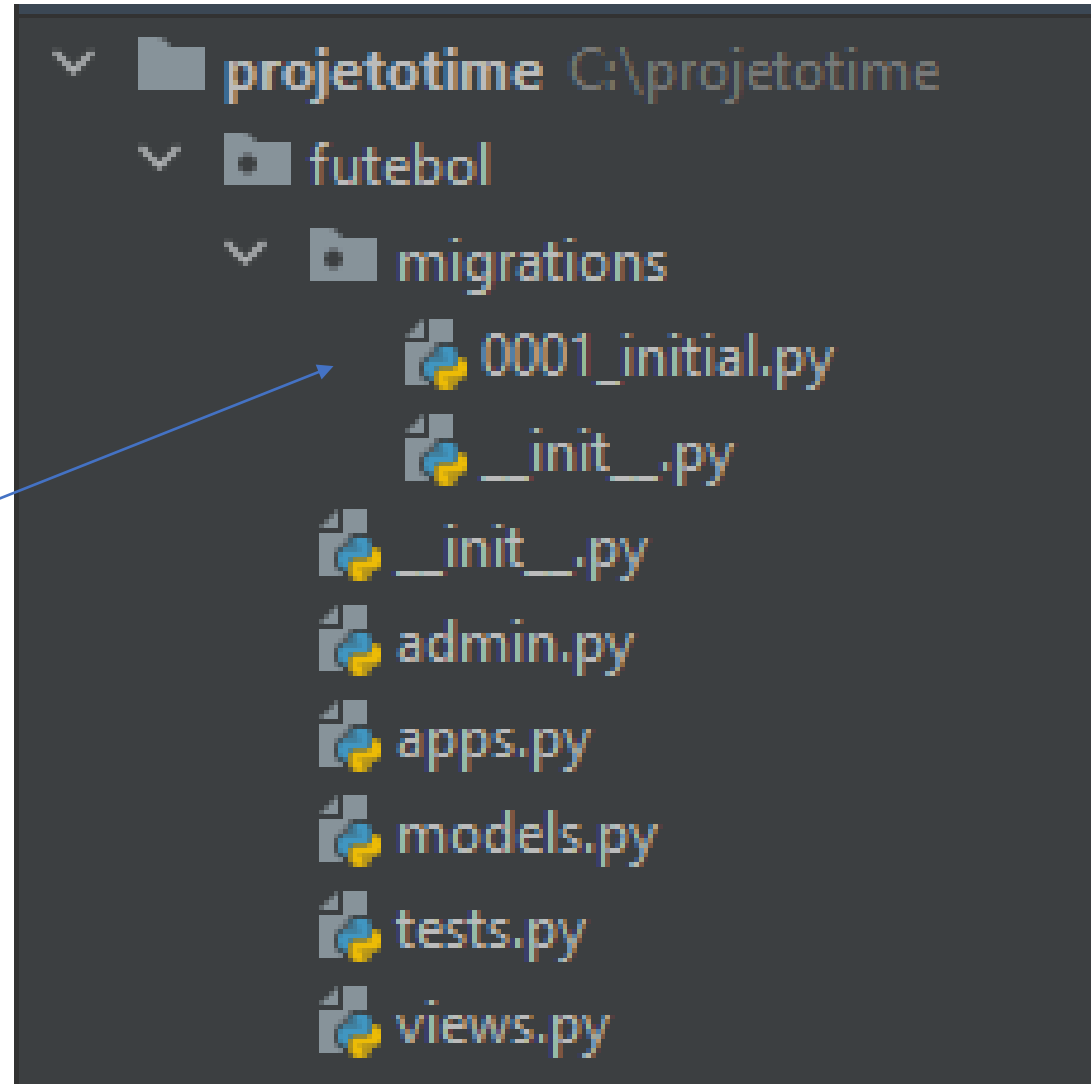
 Prompt de Comando

```
(venv) C:\django>python manage.py makemigrations
Migrations for 'futebol':
  futebol\migrations\0001_initial.py
  - Create model jogador

(venv) C:\django>
```

Verificando Migrações

Migrações realizadas
com sucesso!



Enviando as migrações para o banco admin

Digite: **python manage.py migrate**

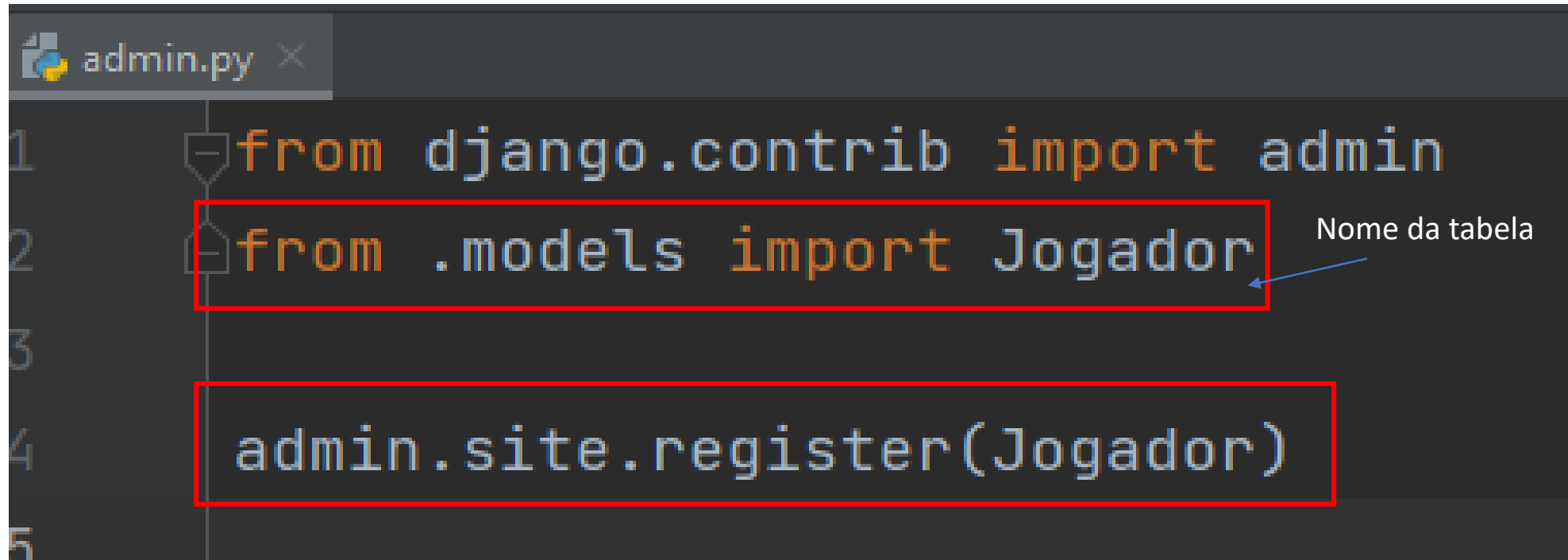
 Prompt de Comando

```
(venv) C:\django>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, futebol, sessions
Running migrations:
  Applying futebol.0001_initial... OK

(venv) C:\django>
```

Registrando e deixando visível no banco

Vá na pasta **sistema**, clique em **admin.py** e registre os models



The screenshot shows a code editor window titled 'admin.py'. The code is as follows:

```
1 from django.contrib import admin
2 from .models import Jogador
3
4 admin.site.register(Jogador)
5
```

Two red rectangular boxes highlight the import statement on line 2 and the registration call on line 4. A blue arrow points from the text 'Nome da tabela' to the word 'Jogador' in the import statement.

Verificando na web

Digite: `python manage.py runserver`

Administração do Django

Administração do Site

AUTENTICAÇÃO E AUTORIZAÇÃO

Grupos

[+ Adicionar](#) [✎ Modificar](#)

Usuários

[+ Adicionar](#) [✎ Modificar](#)

FUTEBOL

Jogadores

[+ Adicionar](#) [✎ Modificar](#)

Verificando na web

Digite: `python manage.py runserver`

Administração do Django

BEM-VINDO(A), **ADMIN**. [VER O SITE](#) / [ALTERAR SENHA](#) / [ENCERRAR SESSÃO](#)

[Início](#) > [Futebol](#) > [Jogadores](#) > Adicionar jogador

Comece a digitar para filtrar...

AUTENTICAÇÃO E AUTORIZAÇÃO

Grupos [+ Adicionar](#)

Usuários [+ Adicionar](#)

FUTEBOL

Jogadores [+ Adicionar](#)

Adicionar jogador

Nome:

Messi

Preço inicial:

20.000

Salvar e adicionar outro(a)

Salvar e continuar editando

SALVAR

Verificando na web

Selecione jogador para modificar

Ação: 0 de 1 selecionados

<input type="checkbox"/>	JOGADOR
<input type="checkbox"/>	jogador object (1)

1 jogador

Vamos consertar esse nome

Retornando nomes

Vá na pasta **sistema**, clique em **models.py** e crie um método de retorno de string.

```
models.py x
1  from django.db import models
2
3  class jogador(models.Model):
4      nome = models.CharField(max_length=50)
5      preco_inicial = models.FloatField()
6
7      def __str__(self):
8          return self.nome
```

Selecione jogador para modificar

Ação:

<input type="checkbox"/>	JOGADOR
<input type="checkbox"/>	Messi

1 jogador

Retornando todos os valores da tabela Jogador

Vá na pasta **sistema**, clique em **admin.py** e crie uma classe administradora para mostrar todos os dados da tabela dentro do banco de dados do django.

```
admin.py x
1 from django.contrib import admin
2 from .models import Jogador
3
4 class JogadorAdmin(admin.ModelAdmin):
5     list_display = ('nome', 'preco_inicial')
6     search_fields = ('nome',) #criando uma busca no banco
7     list_per_page = int = 2
8     model = Jogador
9
10 admin.site.register(Jogador, JogadorAdmin)
11
```

Q Pesquisar

Ação: Ir 0 de 2 selecionados

<input type="checkbox"/>	NOME	PREÇO INICIAL
<input type="checkbox"/>	Carlos	23455,0
<input type="checkbox"/>	José	22500,0

1 2 3 5 jogadores Mostrar tudo

Registre a classe

Criando relacionamentos entre tabelas

TABELA Jogador

João – 2500
José – 3000
Carlos – 4000
Mario - 3400

RELACIONAMENTO

TABELA MeuTime

[João, José]
[João, José, Carlos]
[Carlos, Mario]

TABELA Time

Brasil
Alemanha

RELACIONAMENTO

TABELA Jogos

dataJogo
TimeA (foreignkey)
TimeB(foreignkey)
gols

Criando relacionamentos entre tabelas

Vá na pasta **sistema**, clique em **models.py** e crie as tabelas.

```
class Time(models.Model):
    nome = models.CharField(max_length=50)

    def __str__(self):
        return self.nome

class MeuTime(models.Model):
    players = models.ManyToManyField(Jogador)

    def __str__(self):
        return [jogador_atual.nome for jogador_atual in self.players.all().__str__()]
```

Criando relacionamentos entre tabelas

Vá na pasta **sistema**, clique em **models.py** e crie a tabela Jogo.

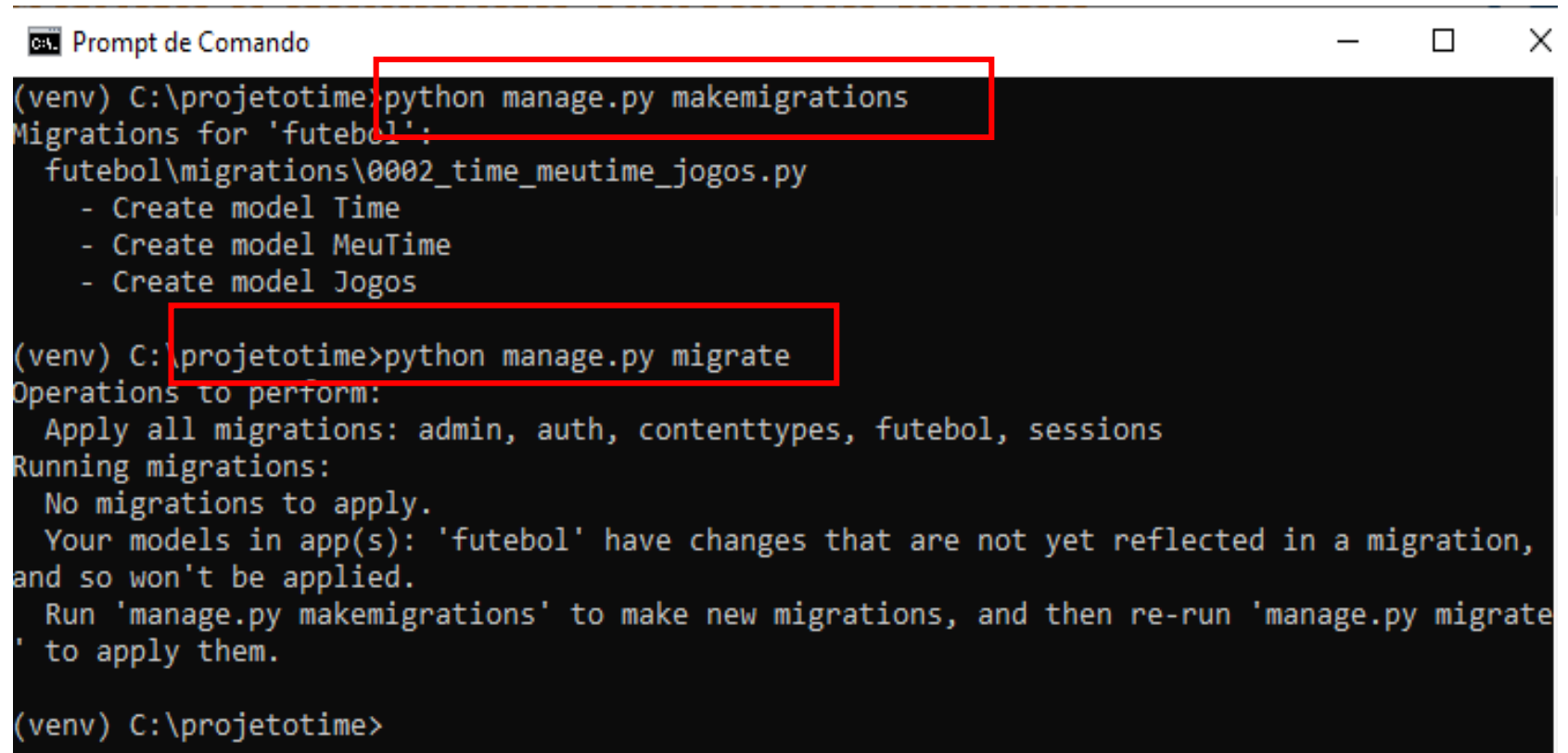
```
class Jogo(models.Model):
    data = models.DateTimeField()
    timeA = models.ForeignKey(Time, on_delete=models.PROTECT, related_name='timeA')
    timeA_gol = models.IntegerField(default=0)
    timeB = models.ForeignKey(Time, on_delete=models.PROTECT, related_name='timeB')
    timeB_gol = models.IntegerField(default=0)

    def __str__(self):
        return f'{self.timeA} x {self.timeB}'
```

Criando a tabela

Agora precisamos criar a tabela e enviar para o banco, digite: **python manage.py makemigrations** e depois **python manage.py migrate**

Tabela criada
e
Enviada para
o banco



```
C:\> Prompt de Comando

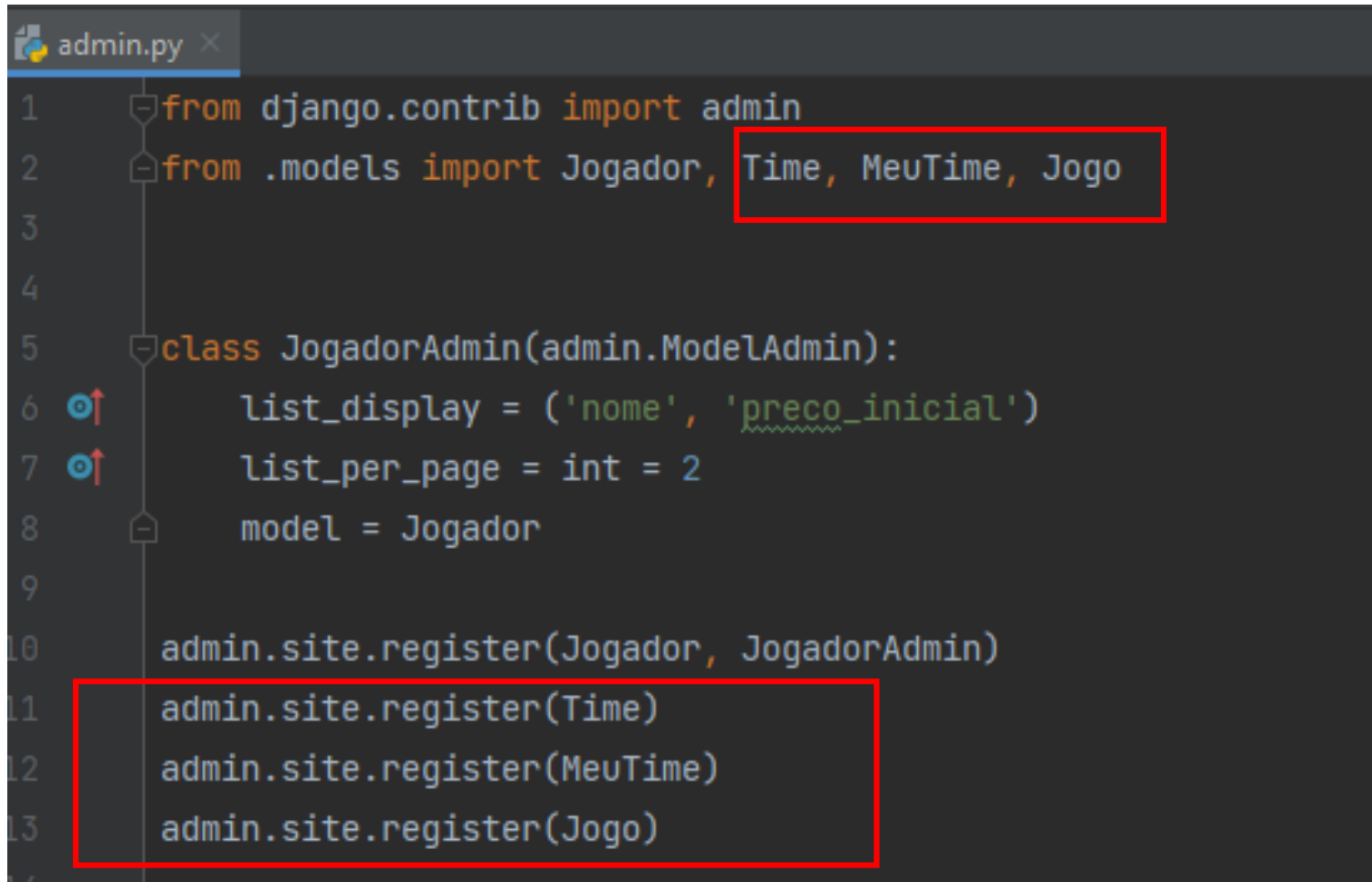
(venv) C:\projetotime>python manage.py makemigrations
Migrations for 'futebol':
  futebol\migrations\0002_time_meutime_jogos.py
    - Create model Time
    - Create model MeuTime
    - Create model Jogos

(venv) C:\projetotime>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, futebol, sessions
Running migrations:
  No migrations to apply.
  Your models in app(s): 'futebol' have changes that are not yet reflected in a migration,
  and so won't be applied.
  Run 'manage.py makemigrations' to make new migrations, and then re-run 'manage.py migrate'
  to apply them.

(venv) C:\projetotime>
```


Registrando e deixando visível no banco

Vá na pasta **sistema**, clique em **admin.py** e registre os models



```
admin.py x
1  from django.contrib import admin
2  from .models import Jogador, Time, MeuTime, Jogo
3
4
5  class JogadorAdmin(admin.ModelAdmin):
6      list_display = ('nome', 'preco_inicial')
7      list_per_page = int = 2
8      model = Jogador
9
10     admin.site.register(Jogador, JogadorAdmin)
11     admin.site.register(Time)
12     admin.site.register(MeuTime)
13     admin.site.register(Jogo)
```

The image shows a code editor window titled 'admin.py'. The code is written in Python and registers Django models. Two red boxes highlight specific parts: the first box encloses the import statement for 'Time, MeuTime, Jogo' from '.models' on line 2; the second box encloses the registration calls 'admin.site.register(Time)', 'admin.site.register(MeuTime)', and 'admin.site.register(Jogo)' on lines 11, 12, and 13 respectively. The code also includes a 'JogadorAdmin' class on lines 5-8 and a registration for 'Jogador' on line 10.

Registrando e deixando visível no banco

```
python manage.py runserver
```

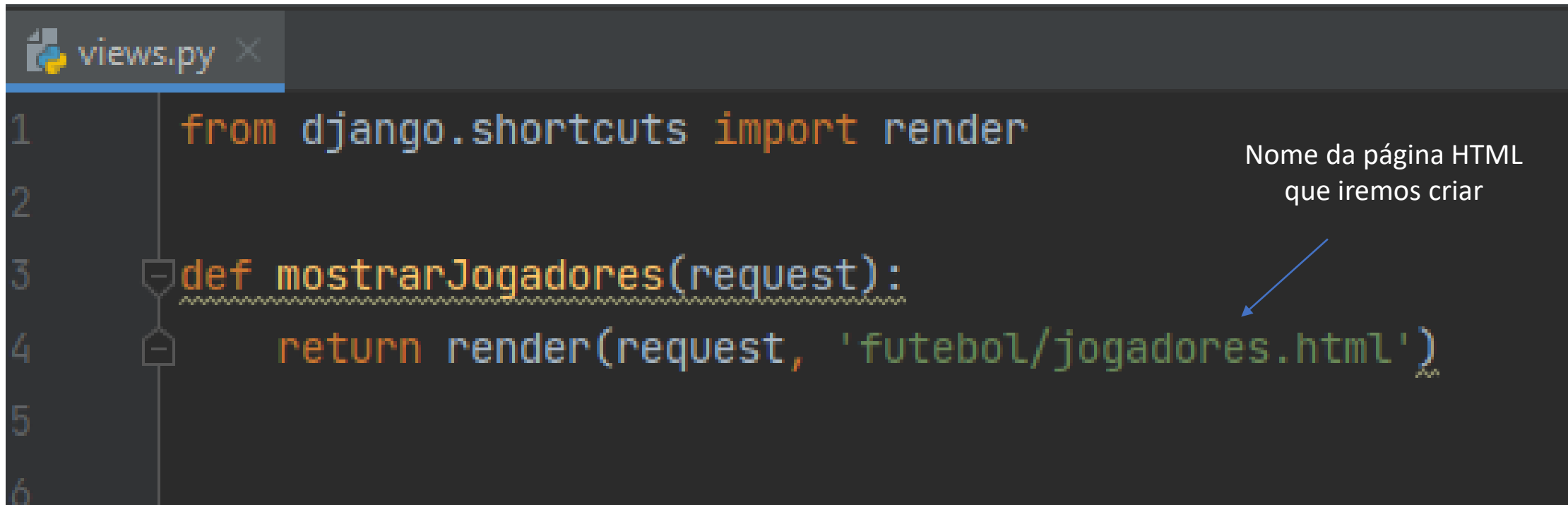




Criando URLs Externas

Criando páginas webs

Para criar uma página web, devemos ir no app **futebol** e clicar em **views**. Após isso, iremos criar uma função para receber a página HTML.



```
views.py x
1  from django.shortcuts import render
2
3  def mostrarJogadores(request):
4      return render(request, 'futebol/jogadores.html')
5
6
```

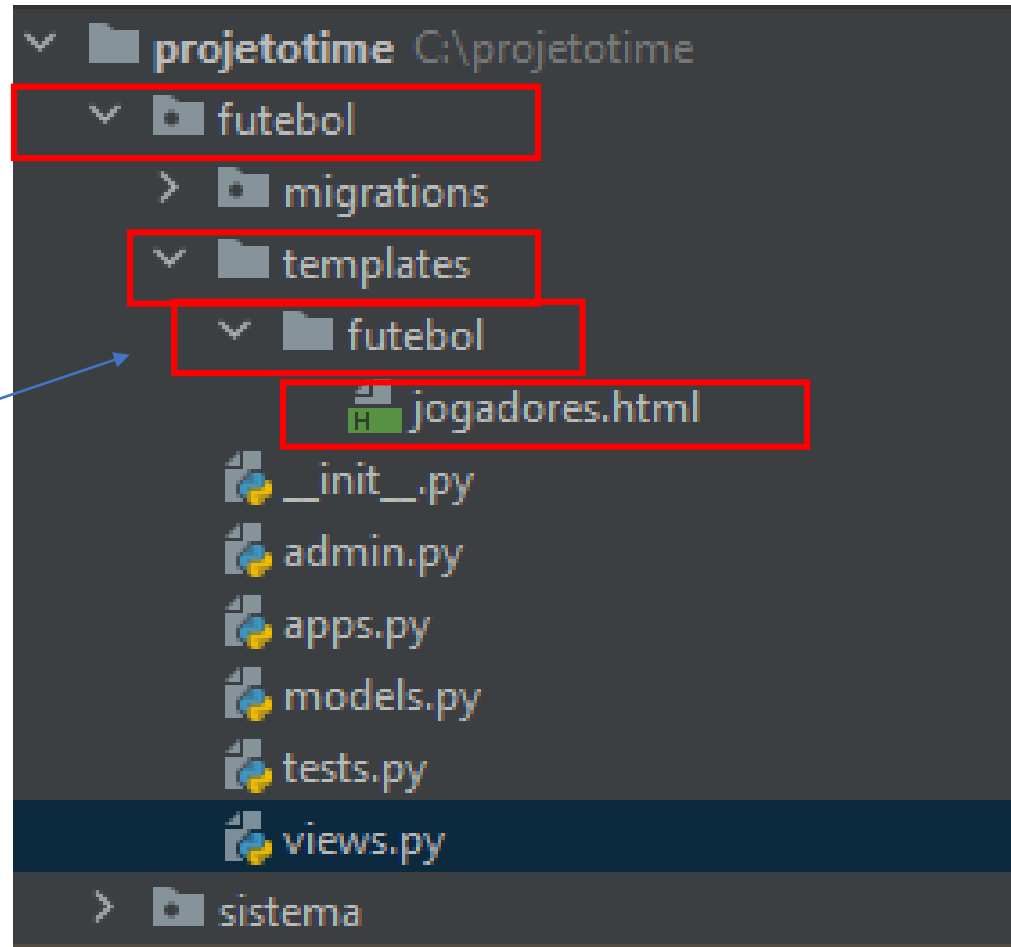
Nome da página HTML
que iremos criar

The image shows a code editor window titled 'views.py'. It contains a Python function named 'mostrarJogadores' that takes a 'request' object and returns the result of 'render(request, 'futebol/jogadores.html')'. A blue arrow points from the text 'Nome da página HTML que iremos criar' to the string 'futebol/jogadores.html' in the code.

Criando páginas webs

Agora vamos criar o diretório chamado de **templates**, e uma subpasta direcionando para a raiz, chamada de **futebol** para receber as páginas HTML.

Guarda todas as páginas
HTML



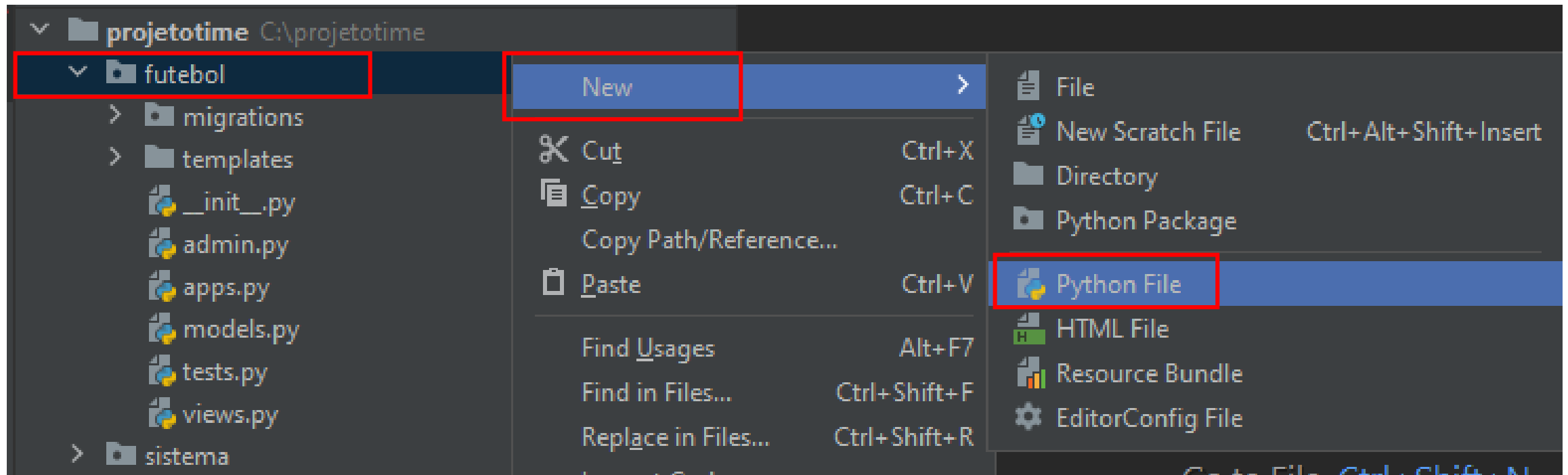
Pegue o código no GitHub: Página_django. HTML

https://github.com/Sonia-95/Biblioteca/blob/main/pagina_django.html.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <title>Página Django</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
  <div class="container-fluid p-5 bg-primary text-white text-center">
    <h1>Minha primeira página Django</h1>
    <p>Jogadores versus Times</p>
  </div>
</body>
</html>
```


Registrando as URLs

Para registrar uma URL, devemos ir no nosso app e criar um arquivo chamado de **urls.py**



Registrando as URLs

Para registrar uma URL, devemos ir no nosso app e criar um arquivo chamado de **urls.py**. Após isso, registramos nas `urlpatterns` o nome e a função criada.

A screenshot of a code editor window titled 'urls.py'. The code is written in Python and defines the URL patterns for a Django application. It imports the 'path' function from 'django.urls' and the 'mostrarJogadores' function from the local 'views' module. Then, it creates a list named 'urlpatterns' containing a single path entry: 'mostrar', which points to the 'mostrarJogadores' function.

```
1 from django.urls import path
2 from .views import mostrarJogadores
3
4 urlpatterns = [
5     path('mostrar', mostrarJogadores),
6 ]
7
```


Enviando URL para o arquivo principal

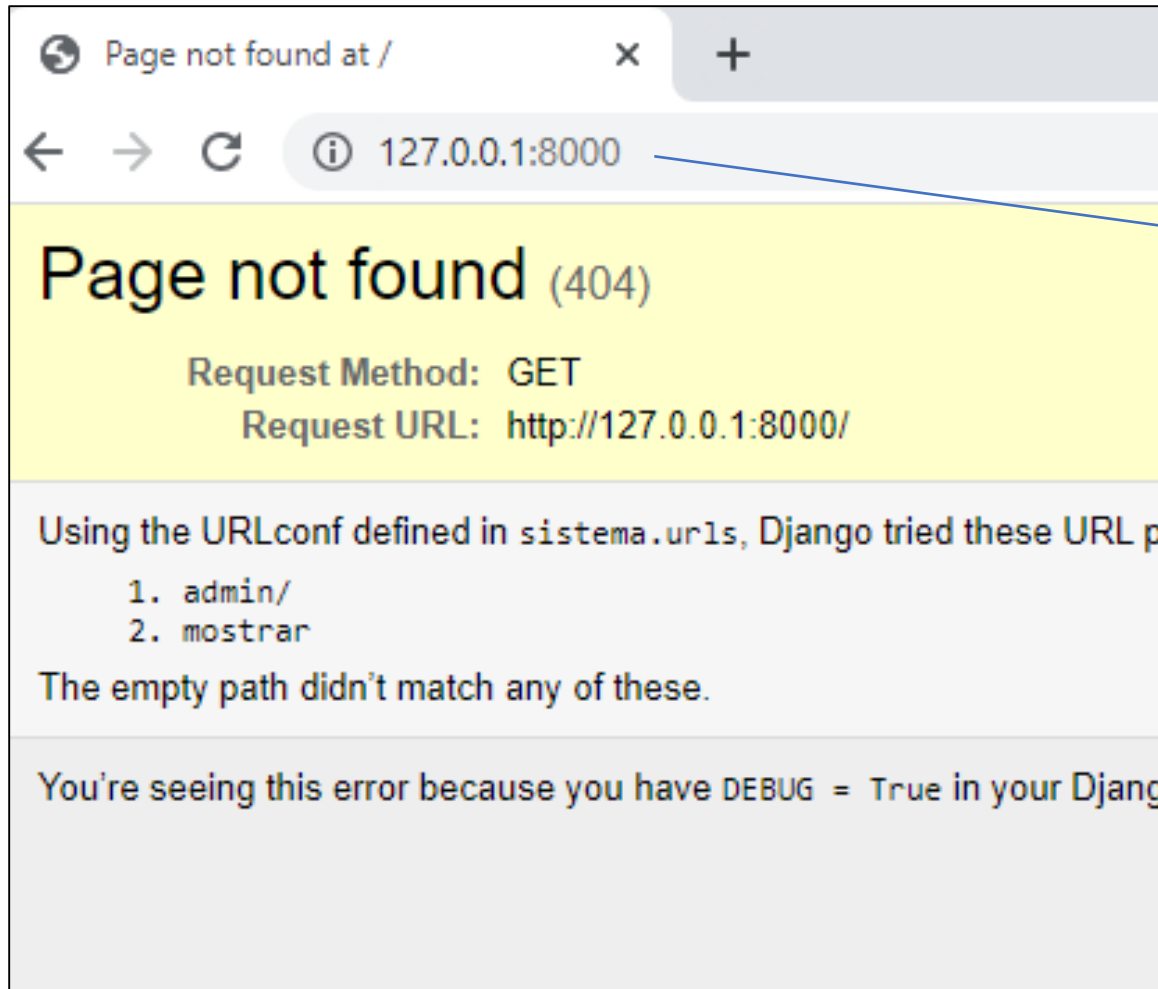
Agora entre na pasta de **sistema** e clique no arquivo `urls.py` para enviar as informações criadas para o servidor.

A screenshot of a code editor window titled 'urls.py'. The code is written in Python and defines the URL patterns for a Django application. It includes imports for 'admin' from 'django.contrib' and 'path', 'include' from 'django.urls'. It then defines a list 'urlpatterns' containing two entries: a path for 'admin/' pointing to 'admin.site.urls' and a path for the root '' pointing to 'futebol.urls'.

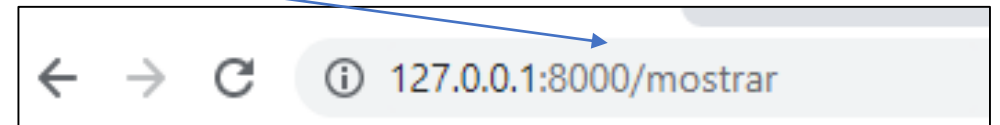
```
1
2 from django.contrib import admin
3 from django.urls import path, include
4
5 urlpatterns = [
6     path('admin/', admin.site.urls),
7     path('', include('futebol.urls')),
8 ]
```

Verificando na web

Agora devemos adicionar no endereço o nome da página

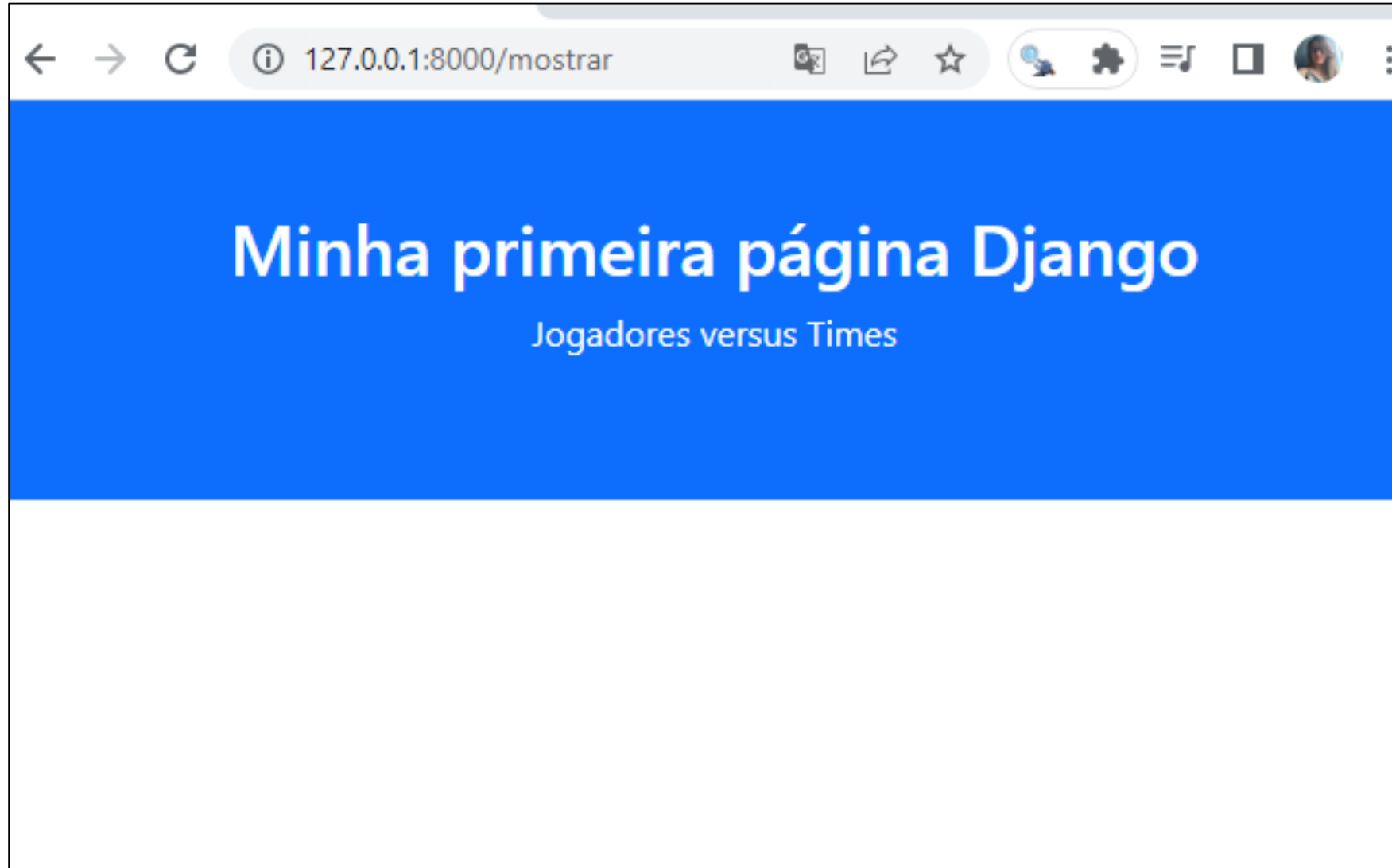


python manage.py runserver



Visualizando na web

`python manage.py runserver`



Queremos
retornar na
página todos
os nomes dos
jogadores

Voltando para views

Aqui na views devemos colocar dentro da função um comando que faça com que retorne na página todos os jogadores

```
dores.html x views.py x
from django.shortcuts import render
from .models import Jogador

def mostrarJogadores(request):
    mostrar = Jogador.objects.all()
    return render(request, 'futebol/jogadores.html', {'mostrar': mostrar})
```

Página HTML

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <title>Página Django</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
<div class="container-fluid p-5 bg-primary text-white text-center">
  <h1>Minha primeira página Django</h1>
  <p>Jogadores versus Times</p>
</div>
```

```
<div class="container mt-5">
  <div class="row">
    <div class="col-sm-4">
      <h3>Jogadores</h3>
      {% for jogador in mostrar %}
        <li> Jogador = {{ jogador.nome }} </li>
      {% endfor %}
    </div>
  </div>
</body>
</html>
```

Visualizando

`python manage.py runserver`

Minha primeira página Django

Jogadores versus Times

Jogadores

- Jogador = Messi
- Jogador = Ronaldo
- Jogador = João
- Jogador = José
- Jogador = Carlos

Retorne a página web com as seguintes informações:

Minha primeira página Django

Jogadores versus Times

Jogadores

- Jogador = Messi ---> Salário = 50000,0
- Jogador = Ronaldo ---> Salário = 10000,0
- Jogador = João ---> Salário = 15034,0
- Jogador = José ---> Salário = 22500,0
- Jogador = Carlos ---> Salário = 23455,0
- Jogador = Paulo ---> Salário = 30000,0

Times

Brasil
Argentina
Japão
Alemanha
Portugal
Camarões
Jamaica

Resolução - Entre na views e configure a tabela

Importe a tabela

Crie uma variável para receber os atributos da tabela Time

Coloque no dicionário, o nome da variável que recebeu a tabela.

```
views.py
1  from django.shortcuts import render
2  from .models import Jogador, Time
3
4  def mostrarJogadores(request):
5      mostrar = Jogador.objects.all()
6      mostrarTime = Time.objects.all()
7
8      return render(request, 'futebol/jogadores.html',
9                    {'mostrar': mostrar,
10                   'mostrarTime': mostrarTime,
11                   })
12
```

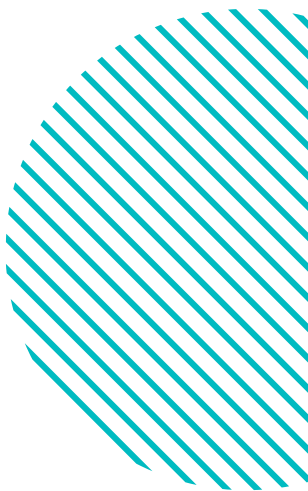


Resolução



Retorne a
página web
com as
seguintes
informações:

```
<div class="container mt-5">
  <div class="row">
    <div class="col-sm-4">
      <h3>Jogadores</h3>
      {% for jogador in mostrar %}
        <li>Jogador = {{ jogador.nome }} ---> Salário = {{ jogador.preco_inicial }}</li>
      {% endfor %}
    </div>
  </div>
</div>
```

```
<div class="container mt-5">
  <div class="row">
    <div class="col-sm-4">
      <h3>Times</h3>
      <tbody>
        {% for time in mostrarTime %}
          <tr>
            <td> {{ time.nome }} <br/> </td>
          </tr>
        {% endfor %}
      </tbody>
    </div>
  </div>
</div>
```



Agora vamos configurar a tabela que tem
um relacionamento **manyto many**
(muitos-para-muitos) e retornar na
página web.

Voltando para views do app futebol

Fazer os
mesmos
procedimentos
do exemplo
anterior

```
views.py x
1  from django.shortcuts import render
2  from .models import Jogador, Time, MeuTime
3
4  def mostrarJogadores(request):
5      mostrar = Jogador.objects.all()
6      mostrarTime = Time.objects.all()
7      meusTimes = MeuTime.objects.all()
8
9      return render(request, 'futebol/jogadores.html',
10                    {'mostrar': mostrar,
11                     'mostrarTime': mostrarTime,
12                     'meusTimes': meusTimes,
13                    })
```

Visualizando uma tabela manytomany na página web

Como os times é um conjunto de nomes, devemos percorrer duas vezes para encontrar os nomes.

```
<div class="container mt-5">
  <div class="row">
    <div class="col-sm-4">
      <h3>Meus Times</h3>
      {% for time in meusTimes %}
        <ul>
          <li> Time com os Jogadores:<br/>
            {% for t in time.players.all %}
              {{ t }} <br/>
            {% endfor %}
          </li>
        </ul>
      {% endfor %}
    </div>
  </div>
</div>
</body>
</html>
```

Resultado

Resultado esperado:

Meus Times

- Time com os Jogadores:
Carlos
Paulo
- Time com os Jogadores:
Messi
Ronaldo
João
José
- Time com os Jogadores:
João
José
- Time com os Jogadores:
Ronaldo
João
José
Carlos

Retorne na página os resultados dos jogos desta forma:

Jogos

- 3 de Dezembro de 2022 às 19:24
- Brasil 3 x 0 Argentina
- 3 de Dezembro de 2022 às 23:30
- Argentina 1 x 2 Japão
- 3 de Dezembro de 2022 às 23:35
- Japão 0 x 0 Brasil
- 12 de Dezembro de 2022 às 23:35
- Portugal 0 x 3 Camarões

Voltando para views do app futebol

Fazer os
mesmos
procedimentos
do exemplo
anterior

```
views.py
1  from django.shortcuts import render
2  from .models import Jogador, Time, MeuTime, Jogo
3
4  def mostrarJogadores(request):
5      mostrar = Jogador.objects.all()
6      mostrarTime = Time.objects.all()
7      meusTimes = MeuTime.objects.all()
8      meusJogos = Jogo.objects.all()
9
10     return render(request, 'futebol/jogadores.html',
11                   {'mostrar': mostrar,
12                   'mostrarTime': mostrarTime,
13                   'meusTimes': meusTimes,
14                   'meusJogos': meusJogos,
15                   })
16
```

Visualizando uma tabela foreingkey na página web

Podemos customizar a página HTML

```
<div class="container mt-5">
  <div class="row">
    <div class="col-sm-4">
      <h3>Jogos</h3>
      {% for time in meusJogos %}
        <ul>
          <li> {{ time.data }}</li>
          <li> {{ time.timeA }} {{ time.timeA_gol }} x {{ time.timeB_gol }} {{ time.timeB }} </li>
        </ul>
      {% endfor %}
    </tbody>
  </div>
</div>
</div>
```