

# Creación dun proxecto con Django

Xurxo Freitas

Data	Motivo da modificación
13/11/25	Versión inicial

## Índice

1. Introducción.....	4
2. Preparar o entorno.....	4
2.1. Instalar Python.....	4
2.2. Instalar Django.....	4
2.2.A) Preparar entorno virtual.....	4
2.2.B) Instalación de Django.....	6
3. Crear un proxecto.....	7
4. Modelos.....	9
Preparar Vscode para Django.....	10
4.1. Usuarios.....	11
4.2. Crear perfiles para os usuarios.....	12
4.3. Post.....	16
4.4. Categorías.....	19
4.5. Comentarios.....	20
5. Panel de administración.....	24
5.1. Super usuario.....	24
5.2. Administración – Usuarios.....	26
5.3. Administración – Post.....	29
5.4. Administración – Categorías.....	30
5.5. Administración – Comentarios.....	31
6. Deseño.....	33
6.1. Index.....	38
6.2. Post » base.html, login.html y register.html.....	47
7. Login.....	51
7.1. Formulario de rexistro.....	53
8. Vistas (Views).....	63
Fontes.....	76

## Índice de imaxes

Figura 1: Comprobar versión Python.....	4
Figura 2: Arrancar servidor con python manage.py runserver.....	7
Figura 3: Obter ruta arquivo interprete de Python.....	10
Figura 4: Instalación de pillow e execución do comando python manage.py makemigrations.....	22
Figura 5: Execución do comando python manage.py migrate.....	23
Figura 6: Instalación de Bootstrap 4 en Django.....	34
Figura 7: base.html.....	47
Figura 8: login.html.....	47
Figura 9: register.html.....	48

## 1. Introducción

Este manual está enfocado para instalar e crear un pequeno proxecto en **Windows 11** con **Django 5.2.8**. Tamén ter en conta que fará falta ter unha base en **Python**

As rutas dos arquivos do proxecto (da nosa web) partirán do directorio **entorno\_django**

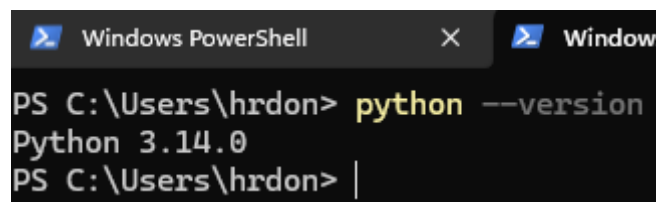
## 2. Preparar o entorno

### 2.1. Instalar Python

Descarga **Python** dende a [web oficial](#) ou co comando

**winget install Python.Python.[versión]**

Utilizaremos a **versión 3.14.0** xa que é a ultima versión **compatible con Django**



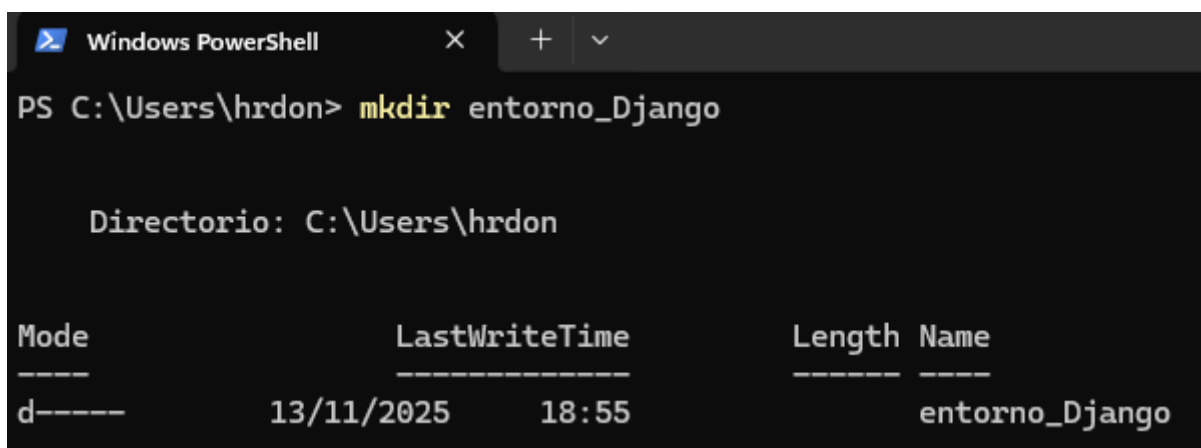
```
PS C:\Users\hrdon> python --version
Python 3.14.0
PS C:\Users\hrdon> |
```

Figura 1: Comprobar versión Python

### 2.2. Instalar Django

#### 2.2.A) Preparar entorno virtual

Antes de comenzar vamos a preparar o noso un **entorno virtual** para manter as dependencias de Python illadas para cada proxecto

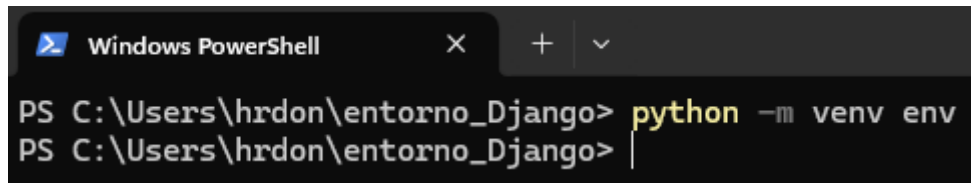


```
PS C:\Users\hrdon> mkdir entorno_Django

Directorio: C:\Users\hrdon

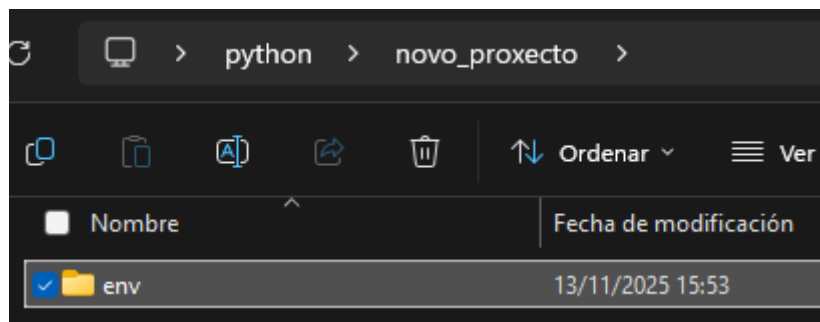
Mode                LastWriteTime         Length Name
----                -
d-----         13/11/2025   18:55                entorno_Django
```

A continuación creamos o entorno virtual **dentro do directorio** que acabamos de crear co comando **python -m venv env**

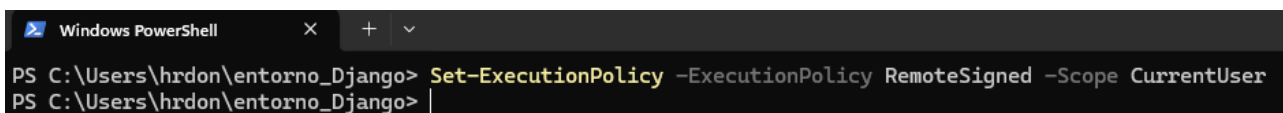


```
Windows PowerShell
PS C:\Users\hrdon\entorno_Django> python -m venv env
PS C:\Users\hrdon\entorno_Django> |
```

Agora xa temos unha nova carpeta chamada **env** con todo o necesario

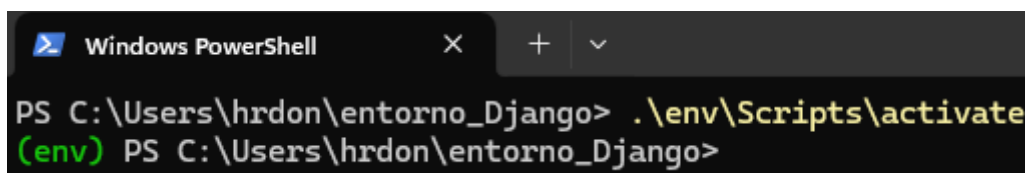


Antes de activar o entorno **habilitamos a execución de scripts** como **administrador**  
**Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser**



```
Windows PowerShell
PS C:\Users\hrdon\entorno_Django> Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
PS C:\Users\hrdon\entorno_Django> |
```

Por último vamos activalo co comando **env\Scripts\activate**



```
Windows PowerShell
PS C:\Users\hrdon\entorno_Django> .\env\Scripts\activate
(env) PS C:\Users\hrdon\entorno_Django>
```

## 2.2.B) Instalación de Django

Tendo o **entorno virtual activo** executamos o seguinte comando

```
Windows PowerShell
(env) PS C:\Users\hrdon\entorno_Django> pip install django
Collecting django
  Using cached django-5.2.8-py3-none-any.whl.metadata (4.1 kB)
Collecting asgiref<=3.8.1 (from django)
  Using cached asgiref-3.10.0-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse<=0.3.1 (from django)
  Using cached sqlparse-0.5.3-py3-none-any.whl.metadata (3.9 kB)
Collecting tzdata (from django)
  Using cached tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
Using cached django-5.2.8-py3-none-any.whl (8.3 MB)
Using cached asgiref-3.10.0-py3-none-any.whl (24 kB)
Using cached sqlparse-0.5.3-py3-none-any.whl (44 kB)
Using cached tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.10.0 django-5.2.8 sqlparse-0.5.3 tzdata-2025.2

[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
(env) PS C:\Users\hrdon\entorno_Django> |
```

para asegurarnos comprobamos a instalación con **django-admin --versión**

```
Windows PowerShell
(env) PS C:\Users\hrdon\entorno_Django> django-admin --version
5.2.8
(env) PS C:\Users\hrdon\entorno_Django> |
```

### 3. Crear un proxecto

Imos crear un novo proxecto, falémolo co comando **django-admin startproject [nome]** .

**Olo: o punto ao final refírese ao directorio actual**

```
(env) PS C:\Users\hrdon\entorno_Django> django-admin.exe startproject web_xurxo
(env) PS C:\Users\hrdon\entorno_Django>
```

Os ficheiros que temos agora dentro de **entorno\_django** son os seguintes:

env	13/11/2025 18:57	Carpeta de archivos	
web_xurxo	20/11/2025 16:13	Carpeta de archivos	
db.sqlite3	20/11/2025 16:13	Archivo SQLITE3	0 KB
manage.py	20/11/2025 16:08	Archivo de origen ...	1 KB

Agora comprobamos dende **entorno\_django** (donde temos gardado o ficheiro **manage.py**) executamos o comando **python manage.py runserver** para arrancar o servidor.

```
Windows PowerShell
(env) PS C:\Users\hrdon\entorno_Django> python.exe .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

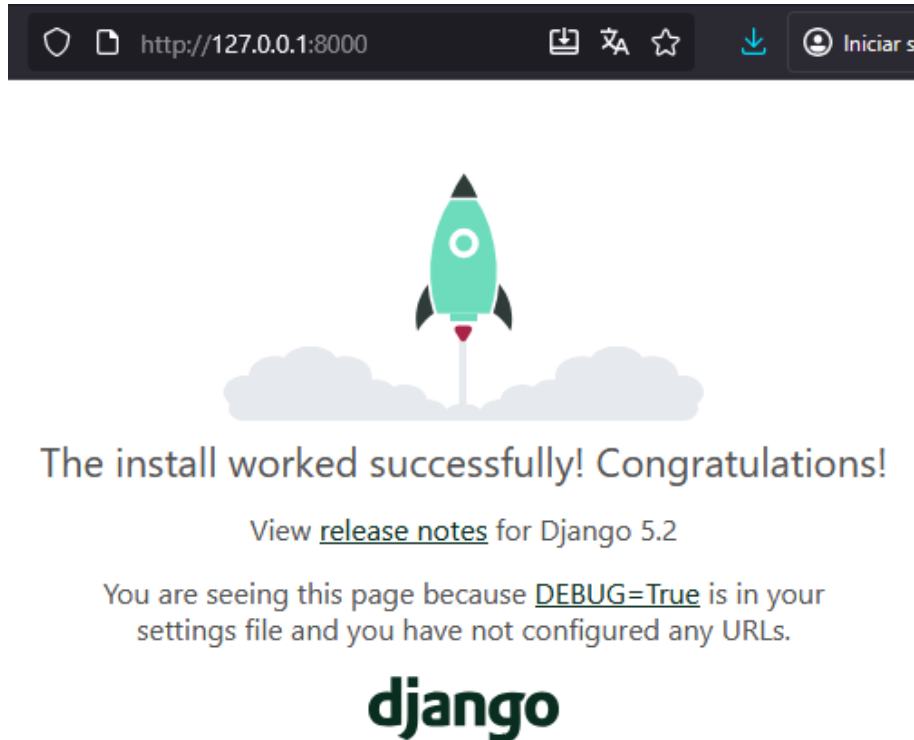
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
November 20, 2025 - 16:27:08
Django version 5.2.8, using settings 'web_xurxo.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/
```

Figura 2: Arrancar servidor con **python manage.py runserver**

**comprobamos** se arrancou correctamente buscando **https://127.0.0.1:8000** no navegador



Logo para **deter o servidor** podemos facer un **Ctrl+C** ou simplemente **pechando a terminal**



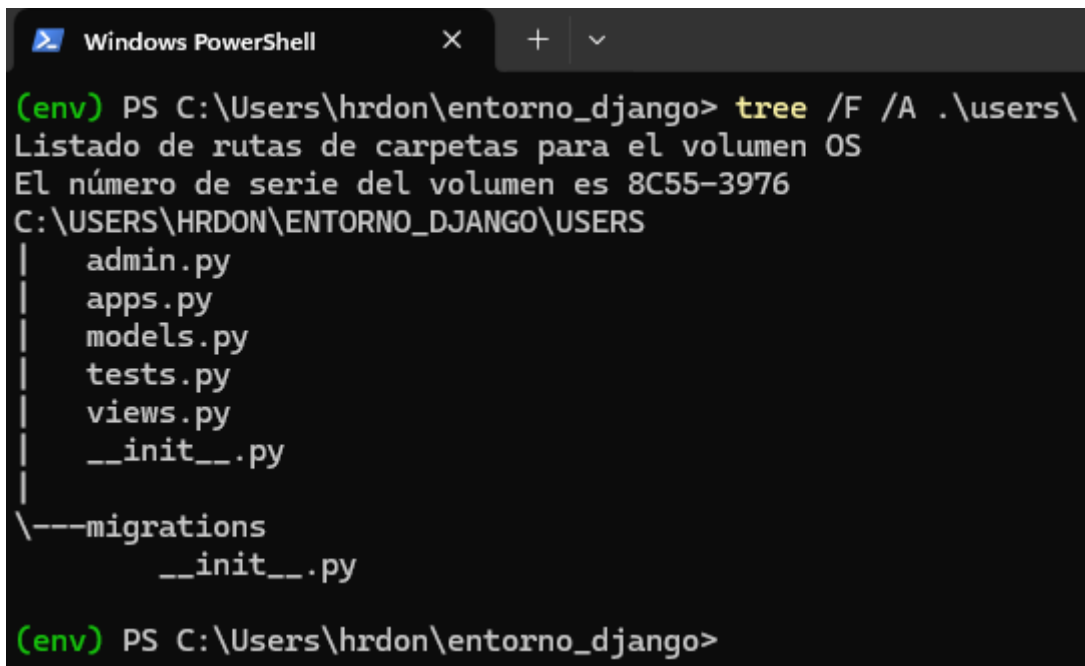
## 4. Modelos

Agora que xa temos a web funcionando vamos crear un **modelo** para a nosa web.

Django ten o seu propio sistema de **autenticación** e modelo para usuarios. Utilizarémolo para crear a nosa propia clase

Para crear o modelo de usuarios executaremos o seguinte comando:

**python manage.py startup users**



```
(env) PS C:\Users\hrdon\entorno_django> tree /F /A .\users\  
Listado de rutas de carpetas para el volumen OS  
El número de serie del volumen es 8C55-3976  
C:\USERS\HRDON\ENTORNO_DJANGO\USERS  
|   admin.py  
|   apps.py  
|   models.py  
|   tests.py  
|   views.py  
|   __init__.py  
|  
|--migrations  
|   __init__.py  
  
(env) PS C:\Users\hrdon\entorno_django>
```

Aparte deses ficheiros en **entorno\_django** tamén nos creou outro ficheiro chamado **db.sqlite3**

## Preparar Vscode para Django

Si vscode non utiliza o **interprete correcto** de **Python** teremos que **indicarlo** nos.

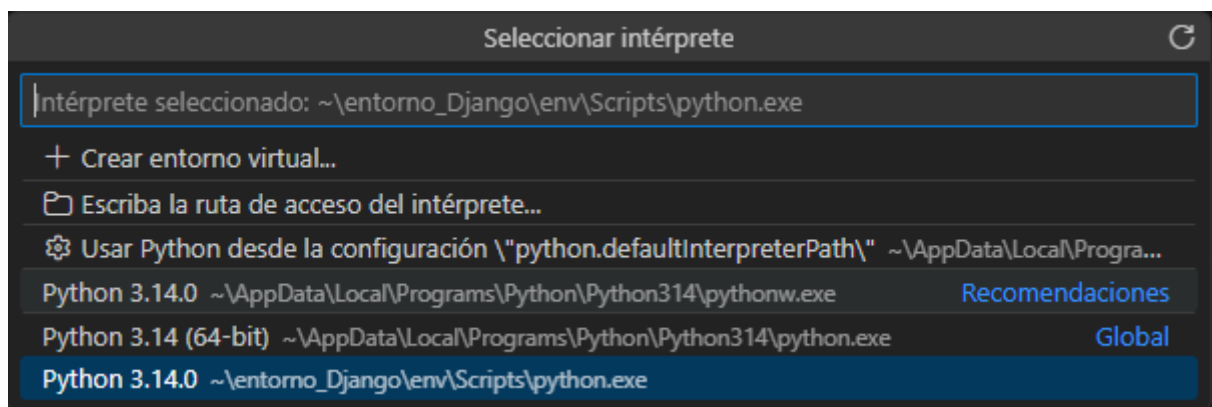
Antes de nada comproba [aquí](#) que as versións de Python e Django sexan **compatibles**, recorda que neste manual estamos utilizando **Python 3.14** e **Django 5.2.8**

Agora vamos buscar a ruta do interprete de Python para indicarlo a vscode, para iso en **powershell** utiliza o comando **python -c "import sys; print(sys.executable)"**

```
(env) PS C:\Users\hrdon\entorno_Django\web_xurxo> python -c "import sys; print(sys.executable)"
C:\Users\hrdon\entorno_Django\env\Scripts\python.exe
(env) PS C:\Users\hrdon\entorno_Django\web_xurxo> |
```

Figura 3: Obter ruta arquivo interprete de Python

A continuación dende **vscode** preme en **Ctrl+shift+P** e escribe **select interpreter** no panel de comandos e saírache un desplegable coma o seguinte



Preme en **Escriba la ruta de acceso del intérprete...** e pega a ruta do paso anterior

Por último **reinicia vscode** e debería funcionar correctamente

## 4.1. Usuarios

Unha vez estes pasos completados vamos co interesante, vamos modificar o ficheiro **users/models.py**

O ficheiro estará practicamente baleiro, engádelle o seguinte código:

```
models.py X
users > models.py > Profile > __str__
1  """Users models."""
2
3  # Django
4  from django.contrib.auth.models import User
5  from django.db import models
6
7
8  class Profile(models.Model):
9      """Profile model.
10
11      Proxy model that extends the base data with other
12      information.
13      """
14      user = models.OneToOneField(User, on_delete=models.PROTECT)
15
16      website = models.URLField(max_length=200, blank=True)
17
18      photo = models.ImageField(
19          upload_to='users/pictures',
20          blank=True,
21          null=True
22      )
23
24      date_modified = models.DateTimeField(auto_now=True)
25
26
27  def __str__(self):
28      """Return username."""
29      return self.user.username
```

O que fai este anaco de código é **importar a clase User** de Django, que xa nos prove de varios campos para a nosa táboa.

O seguinte que facemos na nosa clase **Profile** é relacionar a clase User coa nosa clase un a un e engadiremos a opción **on\_delete=models.PROTECT** que impide que se eliminen usuarios, isto porque temos a opción de desactivar os usuarios, e xa que os usuarios están relacionados con varias táboas o eliminar usuarios poderíamos eliminar información importante.

A continuación personalizamos a clase e engadimos os **nosos propios campos**, para este proxecto vamos engadir a opción de que o usuario poida **gardar a súa website**, engadir unha **foto** e a **data de modificación** do usuario.

Por último engadimos o método **\_\_str\_\_** e retornamos o nome de usuario, isto utilizámolo para cando fagamos un print dun obxecto user nos amose o nome de usuario, o que nos dará máis información do obxecto a simple vista.

## 4.2. Crear perfiles para os usuarios

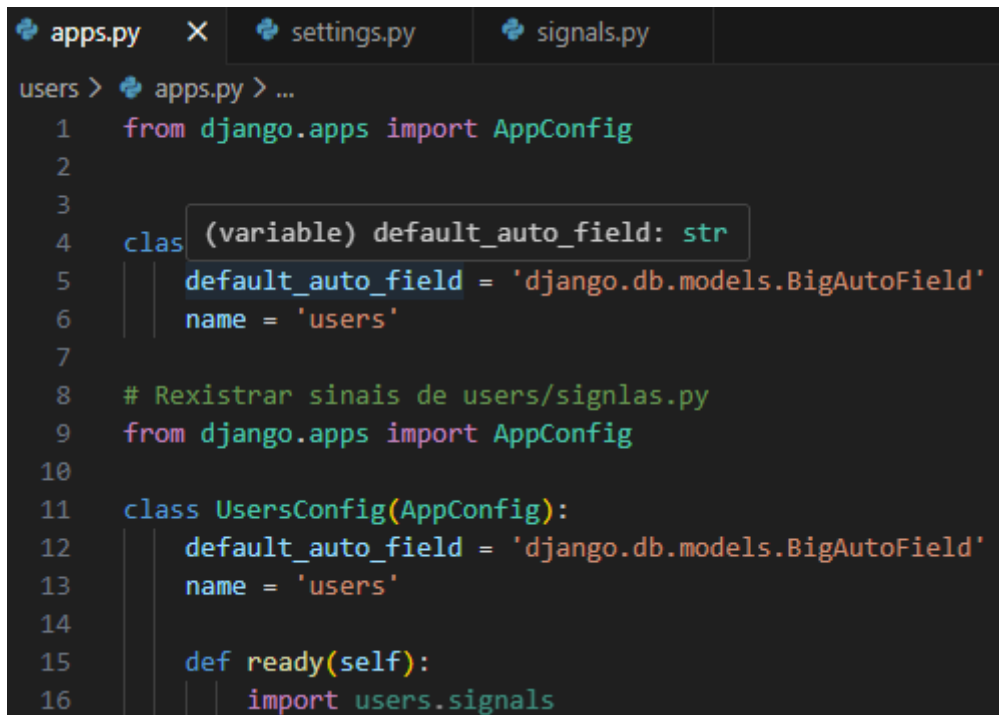
Agora vamos engadir o código para crear os perfiles de usuario cando creemos un usuario novo. Creamos o ficheiro **users/signals.py** e engade o seguinte código:

```

signals.py X  apps.py  settings.py
users > signals.py > ...
1  from django.db.models.signals import post_save
2  from django.dispatch import receiver
3  from django.contrib.auth.models import User
4  from .models import Profile
5
6  @receiver(post_save, sender=User)
7  def create_user_profile(sender, instance, created, **kwargs):
8      if created:
9          Profile.objects.create(user=instance)
10
11  @receiver(post_save, sender=User)
12  def save_user_profile(sender, instance, **kwargs):
13      instance.profile.save()

```

A continuación dirixímonos a **users/apps.py** e comprobamos que temos o seguinte código:

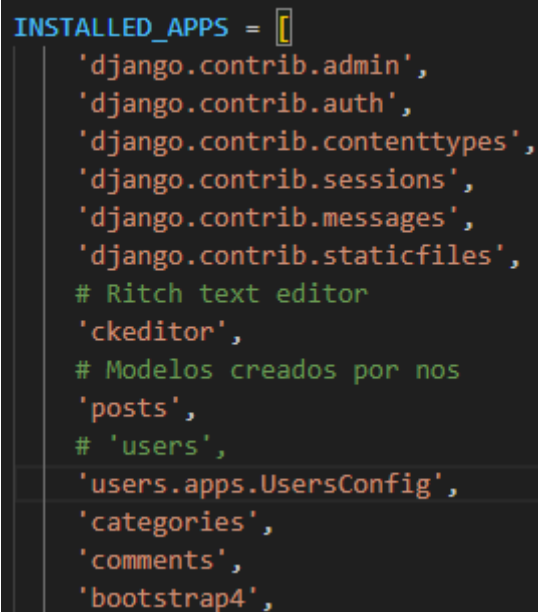


```

apps.py X settings.py signals.py
users > apps.py > ...
1  from django.apps import AppConfig
2
3
4  class (variable) default_auto_field: str
5      default_auto_field = 'django.db.models.BigAutoField'
6      name = 'users'
7
8  # Registrar sinais de users/signlas.py
9  from django.apps import AppConfig
10
11  class UsersConfig(AppConfig):
12      default_auto_field = 'django.db.models.BigAutoField'
13      name = 'users'
14
15      def ready(self):
16          import users.signals

```

Por último vámonos a `web_xurxo/settings.py` e no array `INSTALLED_APPS` sustituimos `'users'` por `'users.apps.UserConfig'`



```

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Ritch text editor
    'ckeditor',
    # Modelos creados por nos
    'posts',
    # 'users',
    'users.apps.UsersConfig',
    'categories',
    'comments',
    'bootstrap4',

```

Co código que acabamos de engadir xa se crean os perfíles de usuario automaticamente cando creamos un usuario novo.

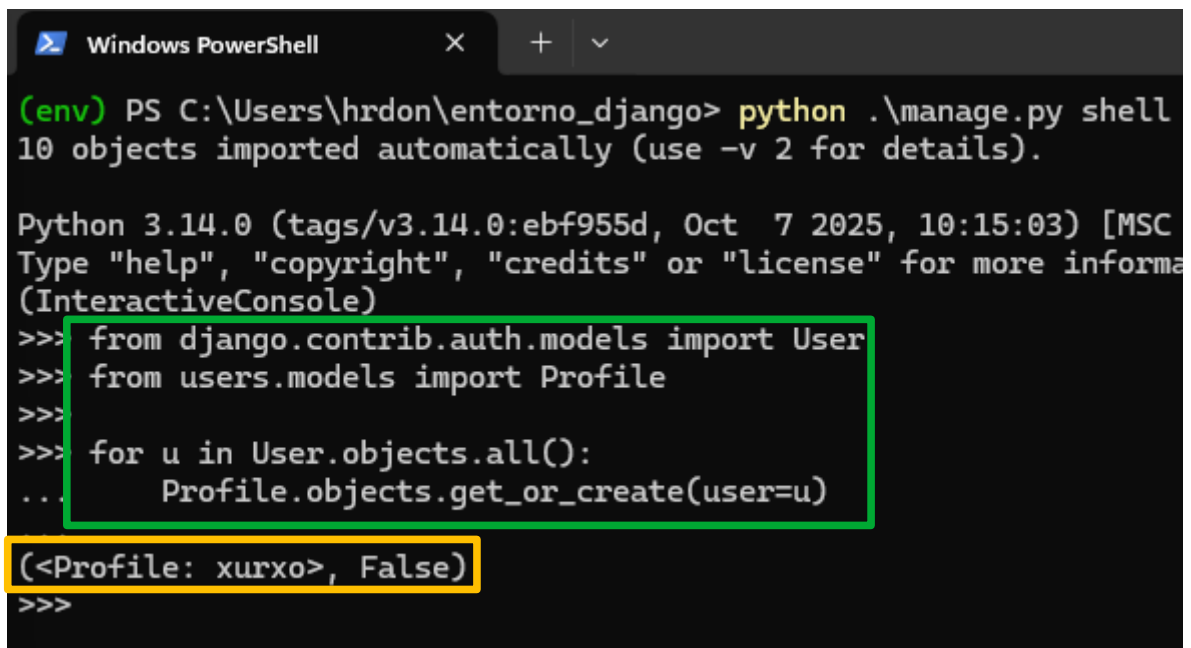
Para os usuarios existentes primeiro abrimos a **shell** de Django » **python manage.py shell**

E unha vez aberto a **shell** do noso entorno virtual engadimos o seguinte código:

```
from django.contrib.auth.models import User
from users.models import Profile

for u in User.objects.all():
    Profile.objects.get_or_create(user=u)
```

E prememos en **Enter** para executar o código e que cree os perfíles.



```
Windows PowerShell
(env) PS C:\Users\hdon\entorno_django> python .\manage.py shell
10 objects imported automatically (use -v 2 for details).

Python 3.14.0 (tags/v3.14.0:ebf955d, Oct  7 2025, 10:15:03) [MSC
Type "help", "copyright", "credits" or "license" for more informa
(InteractiveConsole)
>>> from django.contrib.auth.models import User
>>> from users.models import Profile
>>>
>>> for u in User.objects.all():
...     Profile.objects.get_or_create(user=u)
(<Profile: xurxo>, False)
>>>
```

No **cadro amarelo** indícasenos que **non engadiu o perfil** a ese usuario, en principio porque **xa o tiña**, así que vámolos comprobar:

Sen pechar o shell engade o seguinte código:

```
for u in User.objects.all():  
    print(u.username, hasattr(u, 'profile'))
```

Ao executalo amosaranos a mensaxe do usuario **xurxo True**

```
>>> for u in User.objects.all():  
...     print(u.username, hasattr(u, 'profile'))  
...  
xurxo True  
>>>
```

E con isto xa temos solucionado o tema dos perfíles de usuario.

## 4.3. Post

Agora vamos co **modelo Post** e ao igual que con Users creamos un modelo para os **posts**, para os **comentarios** e outro para **categorías**

```
Windows PowerShell
(env) PS C:\Users\hrdon\entorno_django> python .\manage.py startapp posts
(env) PS C:\Users\hrdon\entorno_django> python .\manage.py startapp categories
(env) PS C:\Users\hrdon\entorno_django> python .\manage.py startapp comments
```

Logo instalamos o paquete **django-ckeditor** que necesitaremos máis adiante

```
Windows PowerShell
(env) PS C:\Users\hrdon\entorno_Django\web_xurxo> pip install django-ckeditor
Collecting django-ckeditor
  Downloading django_ckeditor-6.7.3-py3-none-any.whl.metadata (33 kB)
Requirement already satisfied: Django>=3.2 in c:\users\hrdon\entorno_django\env\lib\site-packages (from django-ckeditor) (5.2.8)
Collecting django-js-asset>=2.0 (from django-ckeditor)
  Downloading django_js_asset-3.1.2-py3-none-any.whl.metadata (6.4 kB)
Requirement already satisfied: asgiref>=3.8.1 in c:\users\hrdon\entorno_django\env\lib\site-packages (from Django>=3.2->django-ckeditor) (3.10.0)
Requirement already satisfied: sqlparse>=0.3.1 in c:\users\hrdon\entorno_django\env\lib\site-packages (from Django>=3.2->django-ckeditor) (0.5.3)
Requirement already satisfied: tzdata in c:\users\hrdon\entorno_django\env\lib\site-packages (from Django>=3.2->django-ckeditor) (2025.2)
Downloading django_ckeditor-6.7.3-py3-none-any.whl (2.5 MB)
2.5/2.5 MB 27.4 MB/s 0:00:00
Downloading django_js_asset-3.1.2-py3-none-any.whl (5.9 kB)
Installing collected packages: django-js-asset, django-ckeditor
Successfully installed django-ckeditor-6.7.3 django-js-asset-3.1.2

[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
(env) PS C:\Users\hrdon\entorno_Django\web_xurxo> |
```



Con todo instalado engadimos o seguinte código en **post/models.py**

```
models.py X
posts > models.py > Post
1  """Posts models."""
2
3  # Django
4  from django.db import models
5
6  from django.utils.text import slugify
7  from django.contrib.auth.models import User
8  from ckeditor.fields import RichTextField
9  from categories.models import Category
10
11 class Post(models.Model):
12     """Post model."""
13
14     user = models.ForeignKey(User, on_delete=models.PROTECT)
15     profile = models.ForeignKey('users.Profile', on_delete=models.PROTECT)
16
17     title = models.CharField(max_length=255)
18     image_header = models.ImageField(upload_to='posts/photos')
19     post = RichTextField()
20
21     created = models.DateTimeField(auto_now_add=True)
22     modified = models.DateTimeField(auto_now=True)
23     is_draft = models.BooleanField(default=True)
24     url = models.SlugField(max_length=255, unique=True)
25     views = models.PositiveIntegerField(default=0)
26     categories = models.ManyToManyField(Category)
27
28     class Meta:
29         ordering = ('title',)
30
31     def __str__(self):
32         """Return title and username."""
33         return '{} by {}'.format(self.title, self.user.username)
34
35     def save(self, *args, **kwargs):
36         self.url = slugify(self.title)
37         super(Post, self).save(*args, **kwargs)
```

Para este modelo importamos:

- **slugify** que nos servirá para crear URL amigables en Django
- Clase **User** xa que un post será creado por un usuario
- Tipo de dato **RichTextField**, que é un campo que ven da librería que instalamos anteriormente

O seguinte que vamos facer será crear a clase **Post** e relacionar os usuarios cos posts coa función **ForeignKey**, será unha **relación 1 a N** xa que un usuario pode crear varios posts e un post pertence a un único usuario. Como no modelo dos usuarios utilizaremos **PROTECT** xa que non queremos perder os posts.

Agora describiremos los demais campos:

- **title**: De tipo CharField e un tamaño máximo de 255 caracteres.
- **image\_header**: Será a cabeceira do noso post gardáremolo en media/posts/photos.
- **post**: Será de tipo RichTextField que nos permitirá crear textos con distintos tamaños, tipos de letra, cores, etc.
- **date\_created**: Data de creación. Gardarase a data actual por defecto polo atributo `auto_now_add` na creación.
- **date\_modified**: Data de modificación. Gardarase a data actual por defecto cada vez que se actualice o post polo atributo `auto_now`.
- **is\_draft**: Un campo de tipo booleano, guardaremos si es un borrador o lo queremos mostrar en la web.
- **url**: De tipo slugField, aquí guardaremos la url, se creará a partir del título.
- **views**: Número de visitas.
- **categories**: El listado de categorías asignadas a un post, es una relación de tipo N a N ya que un post puede tener varias categorías y una categoría puede pertenecer a varios posts.

Despois declaramos a clase **Meta** e engadimos o orde por título, isto fará que cando retornemos unha lista de obxectos tipo post, por defecto ordenará por post.

O método `__str__` que xa o explicamos anteriormente

O método **save**. Aquí o que facemos é que cando se vaia gardar o **Post** pasamos a función **slugify** sobre o título e gardámolo no campo url.

## 4.4. Categorías

Este non ten ciencia ningunha, simplemente ten un campo **name**

```
models.py X
categories > models.py > Category > __str__
1  """Categories"""
2
3  from django.db import models
4
5  # Models
6
7  # Create your models here.
8  class Category(models.Model):
9      """Category model."""
10
11     name = models.CharField(max_length=100,unique=True)
12
13     class Meta:
14         ordering = ('name',)
15
16     def __str__(self):
17         return self.name
```

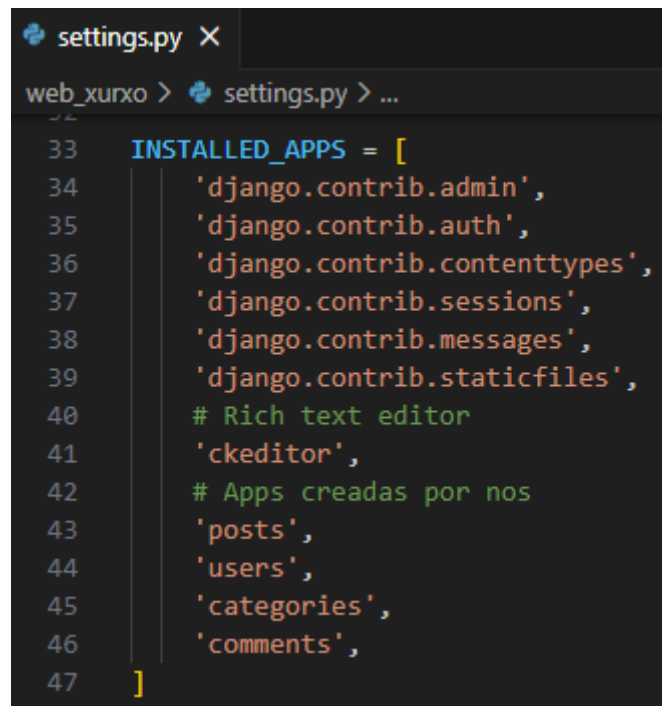
## 4.5. Comentarios

Por último creamos o modelo para os **comentarios**:

```
models.py X
comments > models.py > Comment > __str__
1  """Comments"""
2
3  from django.db import models
4  from django.contrib.auth.models import User
5  from posts.models import Post
6
7  # Model
8
9  # Create your models here.
10 class Comment(models.Model):
11     """Comment model."""
12
13     user = models.ForeignKey(User, on_delete=models.PROTECT)
14     profile = models.ForeignKey('users.Profile', on_delete=models.PROTECT)
15     post = models.ForeignKey(Post, on_delete=models.PROTECT)
16     comment = models.CharField(max_length=5000)
17
18     def __str__(self):
19         return self.comment
```

Con este código establecemos relacións 1-a-N entre comentarios, usuarios e publicacións, e evitamos cargar automaticamente tódolos comentarios dende o modelo de Post para mellorar o rendemento e obtelos só cando sexan necesarios.

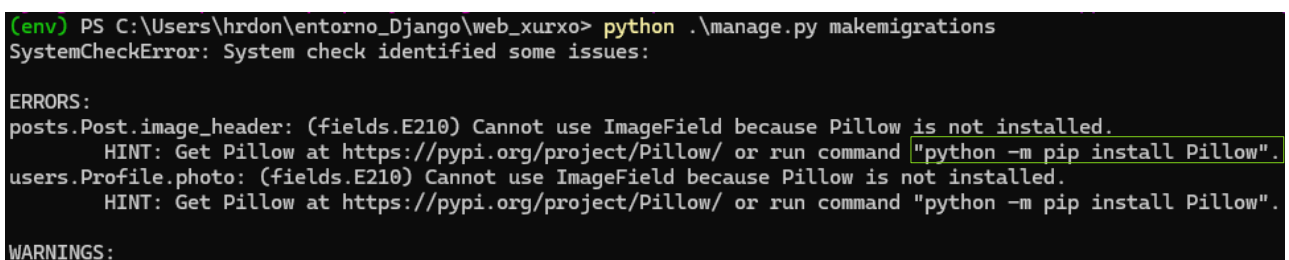
A continuación modificamos o arquivo **web\_xurxo/settings.py** e engadimos os modelos que acabamos de crear:



```
settings.py X
web_xurxo > settings.py > ...
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     # Rich text editor
41     'ckeditor',
42     # Apps creadas por nos
43     'posts',
44     'users',
45     'categories',
46     'comments',
47 ]
```

Unha vez creados os modelos executamos os comandos

- **python manage.py makemigrations**
- **python manage.py migrate**



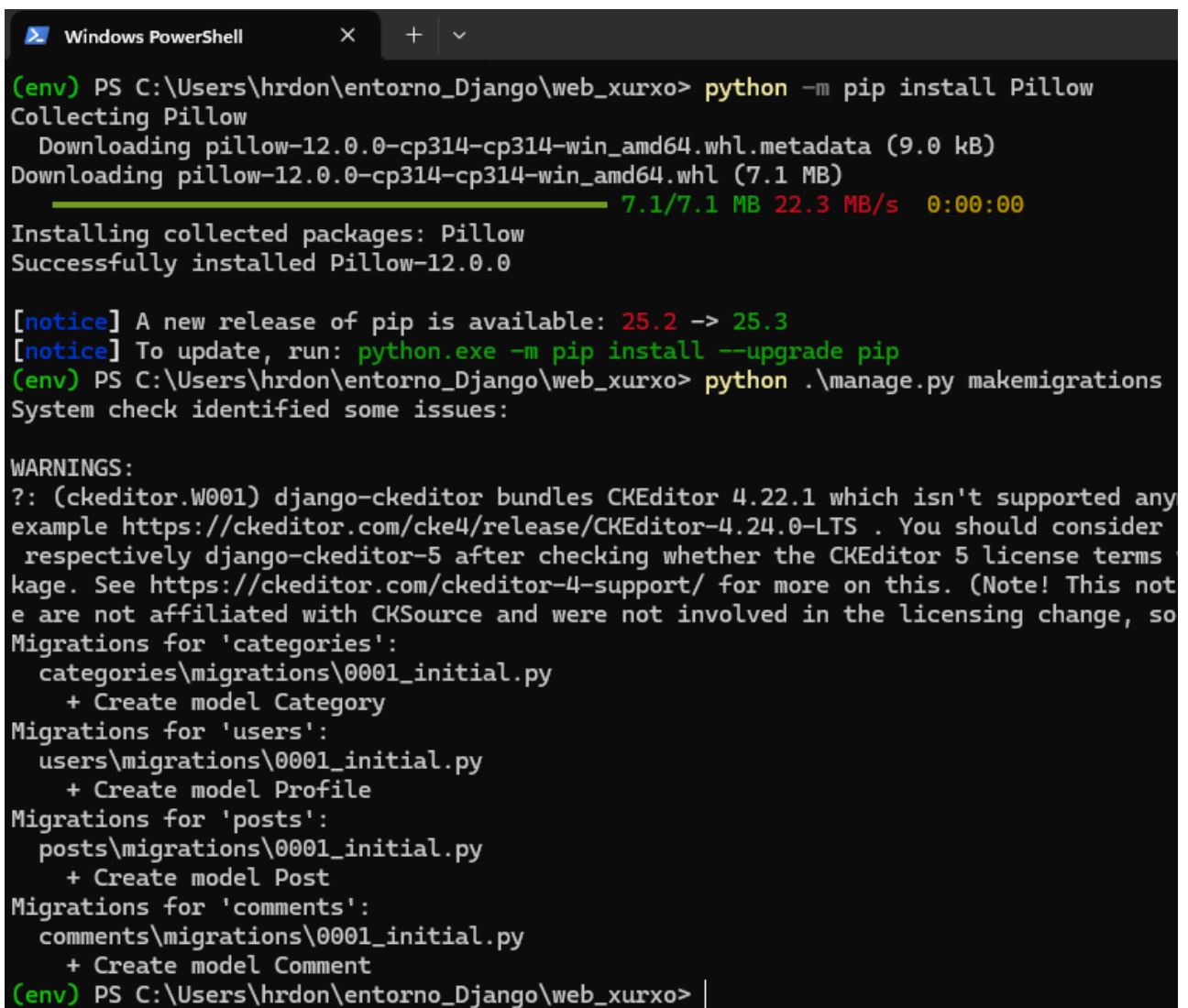
```
(env) PS C:\Users\hrdon\entorno_Django\web_xurxo> python .\manage.py makemigrations
SystemCheckError: System check identified some issues:

ERRORS:
posts.Post.image_header: (fields.E210) Cannot use ImageField because Pillow is not installed.
  HINT: Get Pillow at https://pypi.org/project/Pillow/ or run command "python -m pip install Pillow".
users.Profile.photo: (fields.E210) Cannot use ImageField because Pillow is not installed.
  HINT: Get Pillow at https://pypi.org/project/Pillow/ or run command "python -m pip install Pillow".

WARNINGS:
```

Sáltanos **erro** porque **non** temos **instalado Pillow**

Simplemente executando o comando que nos indica (**python -m pip install Pillow**) solucionamos o problema

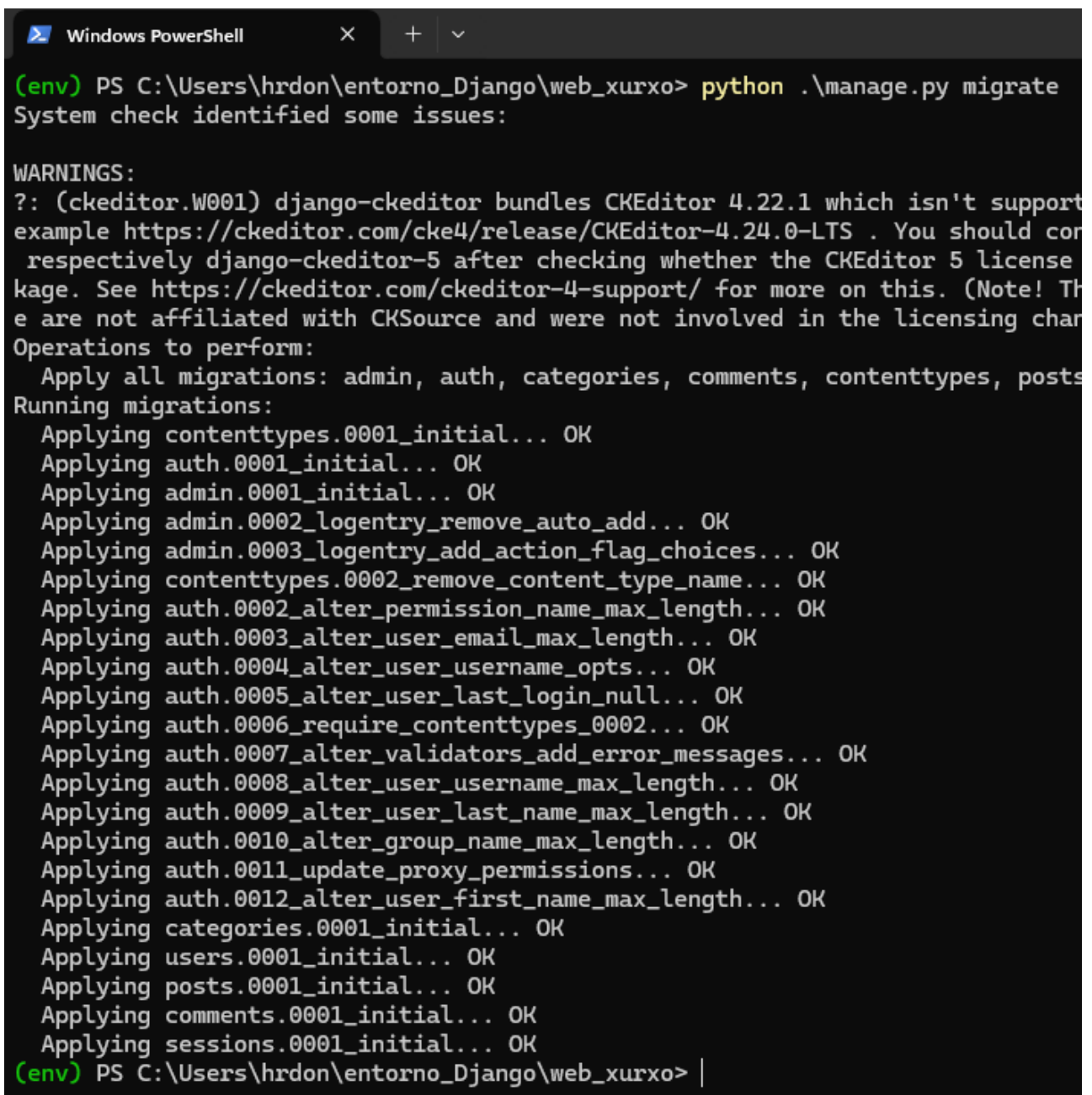


```
(env) PS C:\Users\hrdon\entorno_Django\web_xurxo> python -m pip install Pillow
Collecting Pillow
  Downloading pillow-12.0.0-cp314-cp314-win_amd64.whl.metadata (9.0 kB)
  Downloading pillow-12.0.0-cp314-cp314-win_amd64.whl (7.1 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.1/7.1 MB 22.3 MB/s 0:00:00
Installing collected packages: Pillow
Successfully installed Pillow-12.0.0

[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
(env) PS C:\Users\hrdon\entorno_Django\web_xurxo> python .\manage.py makemigrations
System check identified some issues:

WARNINGS:
?: (ckeditor.W001) django-ckeditor bundles CKEditor 4.22.1 which isn't supported any
example https://ckeditor.com/cke4/release/CKEditor-4.24.0-LTS . You should consider
respectively django-ckeditor-5 after checking whether the CKEditor 5 license terms
kage. See https://ckeditor.com/ckeditor-4-support/ for more on this. (Note! This not
e are not affiliated with CKSource and were not involved in the licensing change, so
Migrations for 'categories':
  categories\migrations\0001_initial.py
    + Create model Category
Migrations for 'users':
  users\migrations\0001_initial.py
    + Create model Profile
Migrations for 'posts':
  posts\migrations\0001_initial.py
    + Create model Post
Migrations for 'comments':
  comments\migrations\0001_initial.py
    + Create model Comment
(env) PS C:\Users\hrdon\entorno_Django\web_xurxo> |
```

Figura 4: Instalación de pillow e ejecución do comando `python manage.py makemigrations`



```
(env) PS C:\Users\hrrdon\entorno_Django\web_xurxo> python .\manage.py migrate
System check identified some issues:

WARNINGS:
?: (ckeditor.W001) django-ckeditor bundles CKEditor 4.22.1 which isn't support
example https://ckeditor.com/cke4/release/CKEditor-4.24.0-LTS . You should con
respectively django-ckeditor-5 after checking whether the CKEditor 5 license
kage. See https://ckeditor.com/ckeditor-4-support/ for more on this. (Note! Th
e are not affiliated with CKSource and were not involved in the licensing char
Operations to perform:
  Apply all migrations: admin, auth, categories, comments, contenttypes, posts
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 categories.0001_initial... OK
  Applying users.0001_initial... OK
  Applying posts.0001_initial... OK
  Applying comments.0001_initial... OK
  Applying sessions.0001_initial... OK
(env) PS C:\Users\hrrdon\entorno_Django\web_xurxo> |
```

Figura 5: Ejecución do comando *python manage.py migrate*

## 5. Panel de administración

A interface de administración permítenos ler os metadatos dos modelos creados no noso proxecto e administrar o contido noso sitio web.

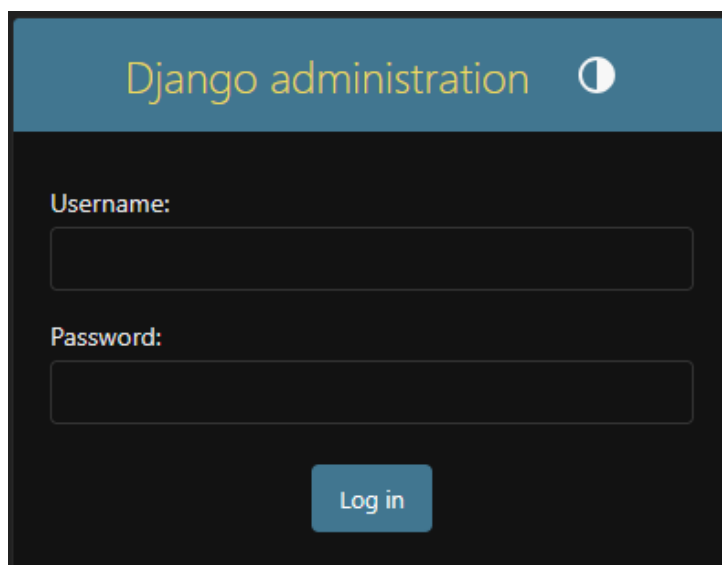
### 5.1. Super usuario

Para poder acceder ao panel primeiro teremos que crear un **súper usuario**

```
Windows PowerShell
(env) PS C:\Users\hrdon\entorno_django> python .\manage.py createsuperuser
System check identified some issues:

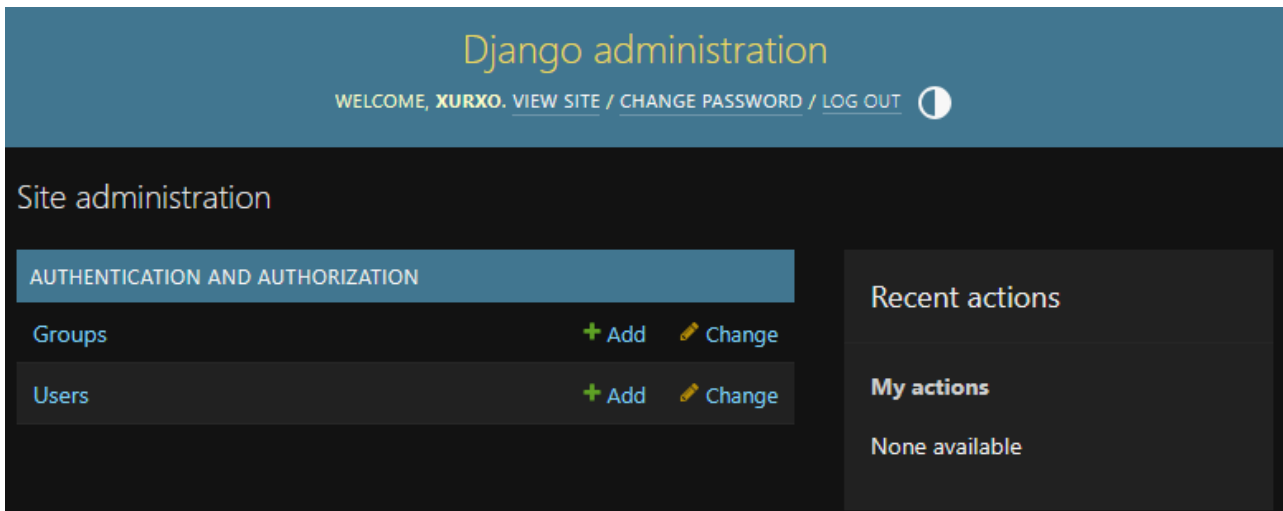
WARNINGS:
?: (ckeditor.W001) django-ckeditor bundles CKEditor 4.22.1 which isn't supported by the license terms work for you) or switch to the non-free CKEditor 4 LTS package.
Username (leave blank to use 'hrdon'): xurxo
Email address: example@email.com
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
(env) PS C:\Users\hrdon\entorno_django> |
```

Unha vez creado o usuario accedemos a url <http://127.0.0.1:8000/admin> (asegúrate de ter o servidor correndo)





Ingresamos o **usuario** e **contrasinal** que acabamos de crear e listo



Dende aquí podemos engadir grupos e usuarios. Só se ven estes dous porque son os que veñen por defecto en Django

## 5.2. Administración – Usuarios

Abrimos o ficheiro **users/admin.py** e pegamos o seguinte código:

```
admin.py X
users > admin.py > ProfileAdmin
1  """User admin classes."""
2
3  # Django
4  from django.contrib.auth.admin import UserAdmin as BaseUserAdmin
5  from django.contrib import admin
6
7  # Models
8  from django.contrib.auth.models import User
9  from users.models import Profile
10
11
12  @admin.register(Profile)
13  class ProfileAdmin(admin.ModelAdmin):
14      """Profile admin."""
15
16      list_display = ('pk', 'user', 'photo')
17      list_display_links = ('pk', 'user',)
18      list_editable = ('photo',)
19
20      search_fields = (
21          'user__email',
22          'user__username',
23          'user__first_name',
24          'user__last_name',
25      )
26
27      list_filter = (
28          'user__is_active',
29          'user__is_staff',
30          'date_modified',
31      )
32
33      fieldsets = (
34          ('Profile', {
35              'fields': (('user', 'photo', 'website'),),
36          }),
37          ('Extra info', {
38              'fields': (('date_modified'),),
39          })
40      )
41
42      readonly_fields = ('date_modified',)
```

Con este código importamos a librería admin e os nosos modelos de **User** e **Profile**.

Con **admin.register(Profile)** rexistramos o noso modelo, o que fará que nos apareza no panel.

Despois creamos a nosa clase **ProfileAdmin**, por convección engádese ao final do nome a palabra **Admin** e aquí é onde vamos a meter configuracións personalizas:

- **list\_display**: Amosaranos a lista de usuarios creados cando esteamos no apartado de profile, a información que se amosará é a que engadimos na lista, neste caso a primary key, o usuario e a foto.
- **list\_display\_links**: Engadirá un link para a edición nos campos que especifiquemos, neste caso na primary key e no usuario.
- **list\_editable**: Esta funcionalidade permitiranos editar a foto dun usuario dende o panel de listados.
- **search\_fields**: Permite a procura polos campos que engadas.
- **list\_filter**: Engade un panel de filtros cos campos que engadimos.
- **field\_sets**: Permite personalizar como se visualizarán os datos no panel, no noso caso, profile e metadata serían as cabeceiras dos campos e dentro de fields se os datos están dentro dunha tupla aparecerán horizontalmente (noso caso), senon verticalmente.
- **readonly\_fields**: Os datos que engadas aquí amosaranse só como lectura.

Por último vamos crear a opción de crear usuarios dende o mesmo panel de control. Para iso utilizaremos a clase **StackedInline**. Engadimos o seguinte código ao final do arquivo **users/admin.py**

```
admin.py X
users > admin.py > UserAdmin
42     readonly_fields = ('date_modified',)
43
44     class ProfileInline(admin.StackedInline):
45         """Profile in-line admin for users."""
46
47         model = Profile
48         can_delete = False
49         verbose_name_plural = 'profiles'
50
51
52     class UserAdmin(BaseUserAdmin):
53         """Add profile admin to base user admin."""
54
55         inlines = (ProfileInline,)
56         list_display = (
57             'username',
58             'email',
59             'first_name',
60             'last_name',
61             'is_active',
62             'is_staff'
63         )
64
65
66     admin.site.unregister(User)
67     admin.site.register(User, UserAdmin)
```

Neste código, primeiro creamos a clase **ProfileInline**, nesta gardamos o modelo que queremos engadir ao modelo pai.

Logo o que temos que facer é **quitar** a **clase User** que é a que está por defecto e rexistrar a nosa clase nova. Logo de declarala engadimos a clase de tipo **StackedInline** e no **list\_display** metemos os datos que queremos que se amosen ao listar os usuarios.

Por último utilizamos o método **unregister** para quitar a nosa clase **User** e rexistramos a nova clase pasando o modelo base e a nova clase.

## 5.3. Administración – Post

Para crear os paneis de administración abrimos o ficheiro **web\_xurxo/posts/admin.py** e pegamos o seguinte código

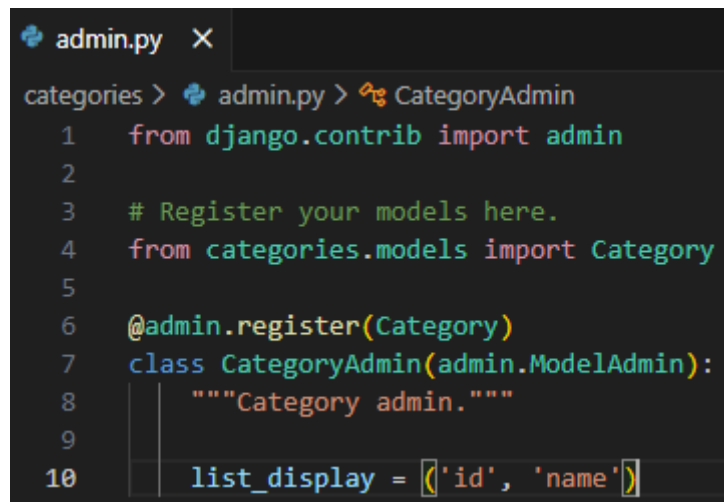
```
admin.py X
posts > admin.py > ...
1  from django.contrib import admin
2
3  # Register your models here.
4  from posts.models import Post
5
6  @admin.register(Post)
7  class PostAdmin(admin.ModelAdmin):
8      """Post admin."""
9
10     list_display = ('id', 'user', 'title', 'image_header')
11     search_fields = ('title', 'user__username', 'user__email')
12     list_filter = ('created', 'modified')
13
14
15     def get_form(self, request, obj=None, **kwargs):
16         self.exclude = ('url', )
17         form = super(PostAdmin, self).get_form(request, obj, **kwargs)
18         form.base_fields['user'].initial = request.user
19         form.base_fields['profile'].initial = request.user.profile
20         return form
```

Neste caso engadimos o método **get\_form** que nos permite realizar cambios no formulario antes de amosalo.

Os cambios que fixemos son: **ocultar a URL** co método **self.exclude** e que por defecto o creador do post sexa o usuario co que estás logueado.

## 5.4. Administración – Categorías

No modelo `categories` do noso proxecto abrimos o ficheiro de administración `web_xurxo/categories/admin.py`

A screenshot of a code editor window titled 'admin.py'. The editor shows the following Python code:

```
categories > admin.py > CategoryAdmin
1  from django.contrib import admin
2
3  # Register your models here.
4  from categories.models import Category
5
6  @admin.register(Category)
7  class CategoryAdmin(admin.ModelAdmin):
8      """Category admin."""
9
10     list_display = ('id', 'name')
```

Usamos a clase **CategoryAdmin** para personalizar a administración e con **list\_display** amosamos as columnas **id** e **name**.

Con isto podemos ver e modificar as categorías dende a interface de administración de Django.

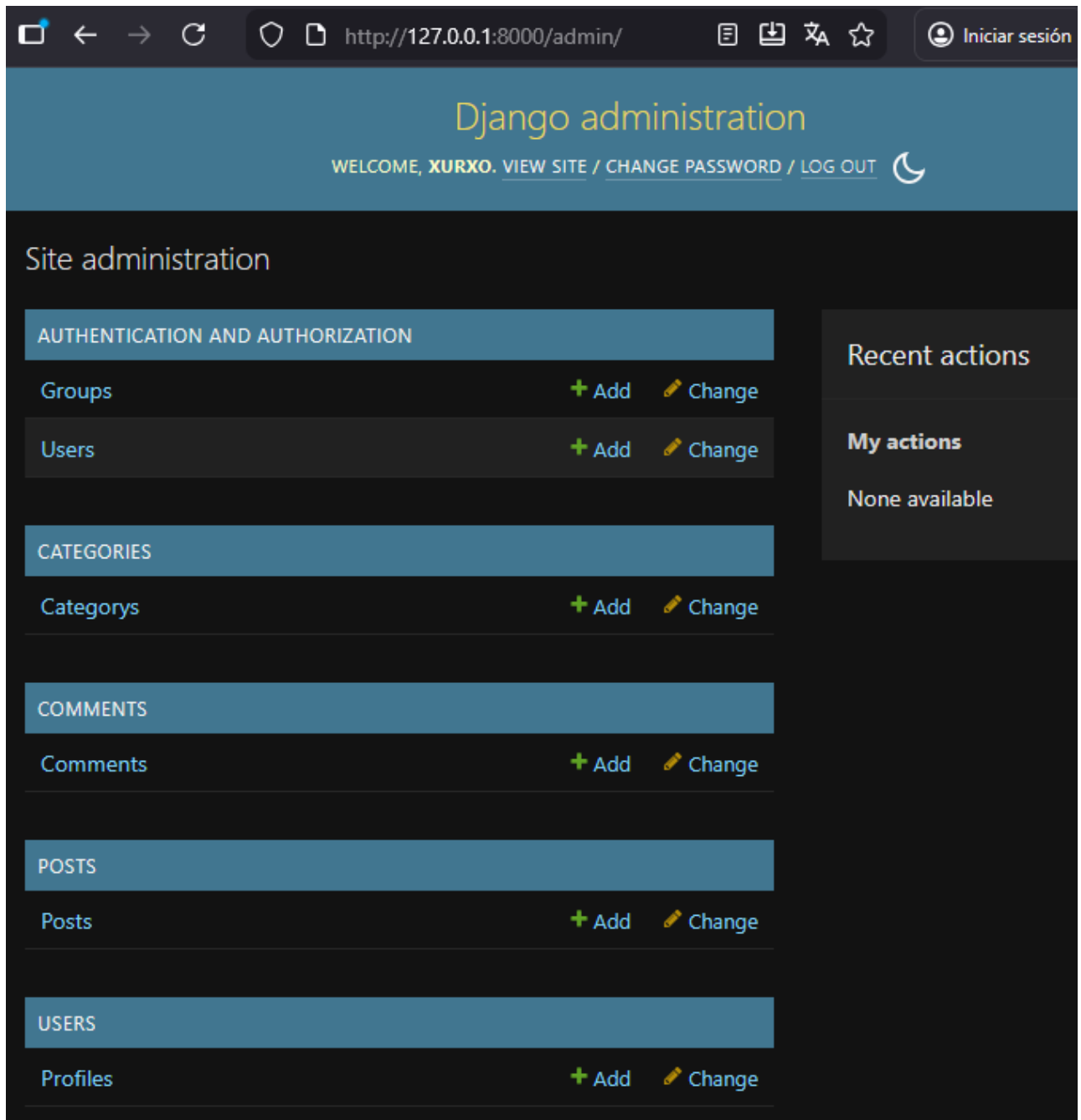
## 5.5. Administración – Comentarios

Neste caso utilizaremos o modelo **comments** para listar os comentarios de forma que podamos buscar comentarios que non cumpran a nosa política e así poder eliminalos.

Abrimos o ficheiro **web\_xurxo/comments/admin.py** e pegamos o seguinte código:

```
admin.py ×
comments > admin.py > CommentAdmin
1  from django.contrib import admin
2
3  # Register your models here.
4  from comments.models import Comment
5
6  @admin.register(Comment)
7  class CommentAdmin(admin.ModelAdmin):
8      """Comment admin."""
9
10     list_display = ('id', 'user', 'post', 'comment')
```

Se agora (co servidor arrancado) nos diriximos ao [panel de administración](#) do noso Django deberíamos ver algo como o da páxina seguinte:





## 6. Deseño

Utilizaremos **Bootstrap** xa que faremos énfase no framework de **Django** e non na parte frontend. Para traballar con templates vamos utilizar o **motor de plantillas de Django**.

O primeiro que temos que facer é **engadir a navegación** ao noso blog.

A nosa web terá unha **páxina principal** onde se amosarán todos os posts, unha páxina de **detalle** por **cada post**, unha de **login**, outra para o **registro** e a páxina “**sobre min**”.

Para poder ver as páxinas que vamos crear necesitamos crealas e apuntar onde están para que cando se acceda mediante a url aparezan, para iso abrimos o arquivo **web\_xurxo/urls.py** e pegamos o seguinte código:

```
urls.py  X
web_xurxo > urls.py > ...
1 > """ ...
17 from django.contrib import admin
18 from django.urls import path
19 from django.views.generic import TemplateView
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path(
24         route='',
25         view=TemplateView.as_view(template_name='posts/index.html'),
26         name='index'
27     ),
28     path(
29         route='post/my-post.html',
30         view=TemplateView.as_view(template_name='posts/detail.html'),
31         name='detail'
32     ),
33     path(
34         route='login',
35         view=TemplateView.as_view(template_name='users/login.html'),
36         name='login'
37     ),
```

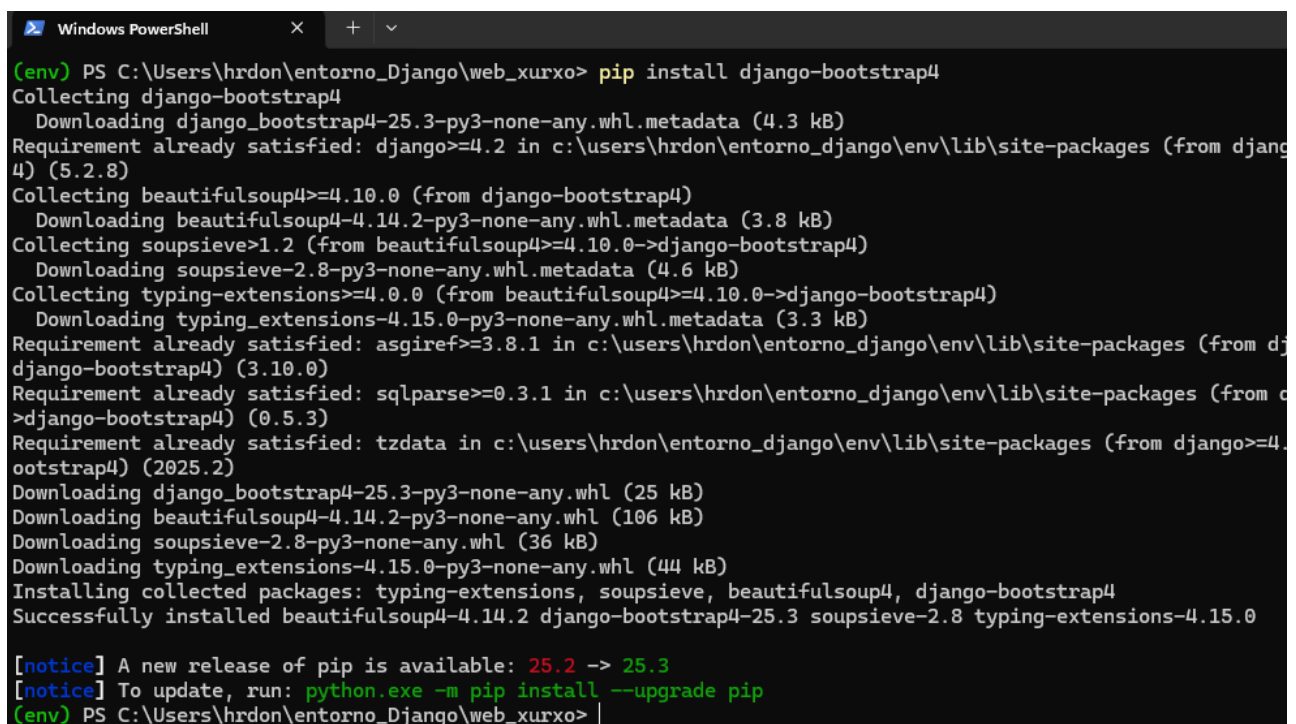
```

38     path(
39         route='registro',
40         view=TemplateView.as_view(template_name='users/register.html'),
41         name='register'
42     ),
43     path(
44         route='sobre-mi',
45         view=TemplateView.as_view(template_name='about.html'),
46         name='about'
47     )
48 ]

```

Engadimos unha **función** de tipo **path** na que gardamos a ruta que se debe escribir para acceder a nosa vista, o nome da plantilla e un nome que se utilizará posteriormente cando teñamos que engadir unha url nas nosas plantillas.

A continuación vamos descargar **bootstrap**, para iso executamos o seguinte comando:



```

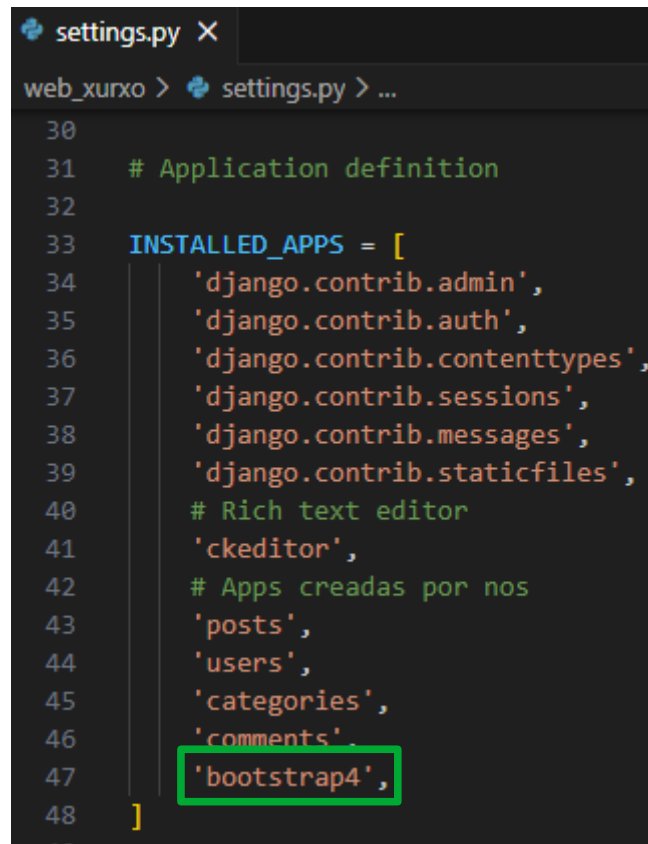
Windows PowerShell
(env) PS C:\Users\hrdon\entorno_Django\web_xurxo> pip install django-bootstrap4
Collecting django-bootstrap4
  Downloading django_bootstrap4-25.3-py3-none-any.whl.metadata (4.3 kB)
Requirement already satisfied: django>=4.2 in c:\users\hrdon\entorno_django\env\lib\site-packages (from django-bootstrap4) (5.2.8)
Collecting beautifulsoup4>=4.10.0 (from django-bootstrap4)
  Downloading beautifulsoup4-4.14.2-py3-none-any.whl.metadata (3.8 kB)
Collecting soupsieve>1.2 (from beautifulsoup4>=4.10.0->django-bootstrap4)
  Downloading soupsieve-2.8-py3-none-any.whl.metadata (4.6 kB)
Collecting typing-extensions>=4.0.0 (from beautifulsoup4>=4.10.0->django-bootstrap4)
  Downloading typing_extensions-4.15.0-py3-none-any.whl.metadata (3.3 kB)
Requirement already satisfied: asgiref>=3.8.1 in c:\users\hrdon\entorno_django\env\lib\site-packages (from django-bootstrap4) (3.10.0)
Requirement already satisfied: sqlparse>=0.3.1 in c:\users\hrdon\entorno_django\env\lib\site-packages (from django-bootstrap4) (0.5.3)
Requirement already satisfied: tzdata in c:\users\hrdon\entorno_django\env\lib\site-packages (from django-bootstrap4) (2025.2)
Downloading django_bootstrap4-25.3-py3-none-any.whl (25 kB)
Downloading beautifulsoup4-4.14.2-py3-none-any.whl (106 kB)
Downloading soupsieve-2.8-py3-none-any.whl (36 kB)
Downloading typing_extensions-4.15.0-py3-none-any.whl (44 kB)
Installing collected packages: typing-extensions, soupsieve, beautifulsoup4, django-bootstrap4
Successfully installed beautifulsoup4-4.14.2 django-bootstrap4-25.3 soupsieve-2.8 typing-extensions-4.15.0

[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
(env) PS C:\Users\hrdon\entorno_Django\web_xurxo>

```

Figura 6: Instalación de Bootstrap 4 en Django

E engadimos a app ao noso arquivo **web\_xurxo/settings.py**



```
settings.py X
web_xurxo > settings.py > ...
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     # Rich text editor
41     'ckeditor',
42     # Apps creadas por nos
43     'posts',
44     'users',
45     'categories',
46     'comments',
47     'bootstrap4',
48 ]
```

No mesmo arquivo engadimos a ruta onde se gardarán as plantillas, para isto temos que editar a constante **TEMPLATES**:

- importamos **os**: módulo estándar de Python que permite interactuar co sistema operativo
- Definir **BASE\_DIR**: variable que apunta ó cartafol raíz do noso proxecto Django

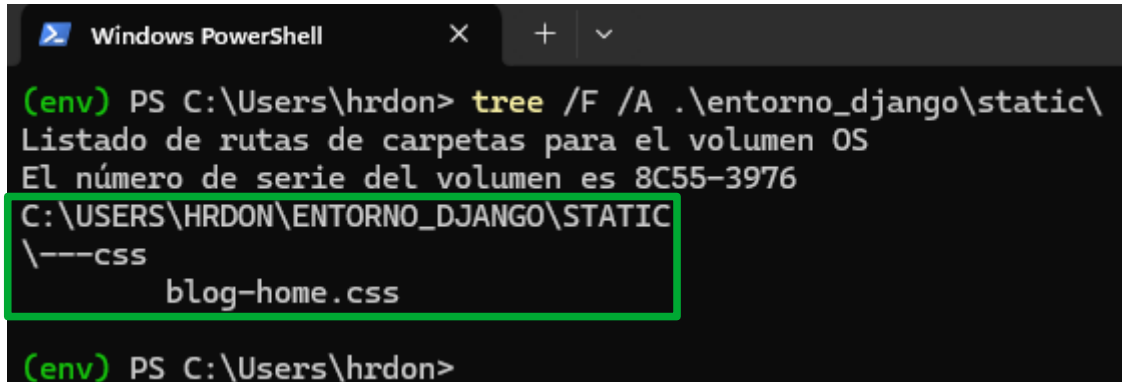
```
settings.py X
web_xurxo > settings.py > ...
62 import os
63
64 TEMPLATES = [
65     {
66         'BACKEND': 'django.template.backends.django.DjangoTemplates',
67         'DIRS': [os.path.join(BASE_DIR, 'templates')],
68         'APP_DIRS': True,
69         'OPTIONS': {
70             'context_processors': [
71                 'django.template.context_processors.debug',
72                 'django.template.context_processors.request',
73                 'django.contrib.auth.context_processors.auth',
74                 'django.contrib.messages.context_processors.messages',
75             ],
76         },
77     ],
78 ]
```

Seguimos modificando o mesmo arquivo. Agora engadimos o seguinte código:

```
settings.py X
web_xurxo > settings.py > ...
125 # Static files (CSS, JavaScript, Images)
126 # https://docs.djangoproject.com/en/5.2/howto/static-files/
127
128 STATIC_URL = '/static/'
129 STATICFILES_DIRS = (
130     os.path.join(BASE_DIR, 'static'),
131 )
132 STATICFILES_FINDERS = [
133     'django.contrib.staticfiles.finders.FileSystemFinder',
134     'django.contrib.staticfiles.finders.AppDirectoriesFinder'
135 ]
```

Para indicarlle a Django onde gardaremos os arquivos estáticos (css, js, imaxes, etc.)

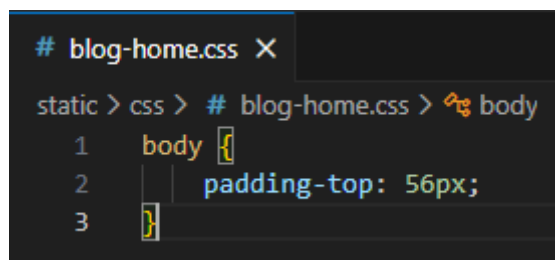
Unha vez gardados os cambios na raíz do noso proxecto creamos o cartafol **static** e dentro desta o cartafol **css** que conterá todas as follas de estilo que creemos, nesta carpeta creamos un arquivo **blog-home.css**. A estrutura debería verse así:



```
(env) PS C:\Users\hrdon> tree /F /A .\entorno_django\static\  
Listado de rutas de carpetas para el volumen OS  
El número de serie del volumen es 8C55-3976  
C:\USERS\HRDON\ENTORNO_DJANGO\STATIC  
|--css  
    blog-home.css  
(env) PS C:\Users\hrdon>
```

Para este proxecto utilizaremos [esta plantilla](#) creada por [startbootstrap](#), utilizaremolo no seguinte paso ([Login](#))

Por último engadimos o seguinte código no ficheiro css que acabamos de crear



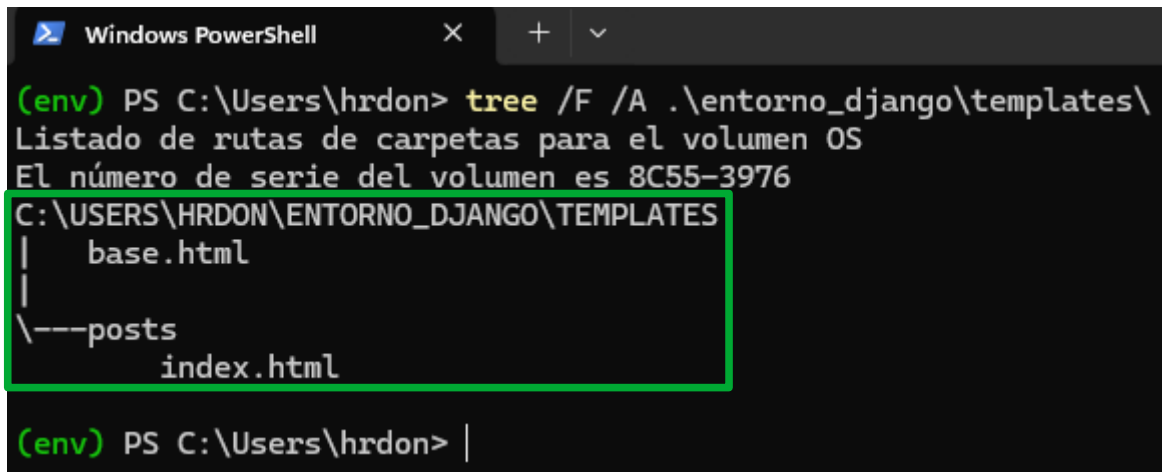
```
# blog-home.css X  
static > css > # blog-home.css > body  
1  body {  
2    padding-top: 56px;  
3  }
```

Despois, no directorio raíz, creamos a carpeta **templates** que será na que gardemos todas as plantillas que utilizemos.

## 6.1. Index

Dentro da carpeta **templates** que acabamos de crear, creamos un arquivo chamado **base.html** que será o que conteña a estrutura básica da web.

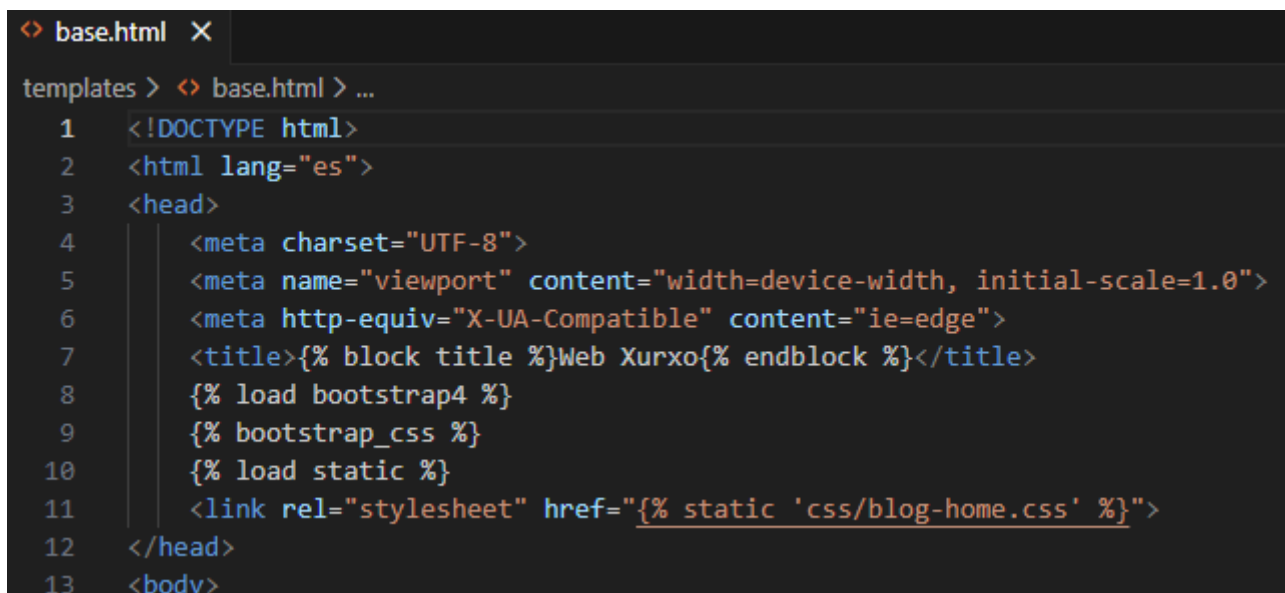
Crearemos tamén unha carpeta chamada **posts** e dentro dela un arquivo chamado **index.html** que será a páxina principal do blog. A estrutura será a seguinte:



```
(env) PS C:\Users\hrdon> tree /F /A .\entorno_django\templates\
Listado de rutas de carpetas para el volumen OS
El número de serie del volumen es 8C55-3976
C:\USERS\HRDON\ENTORNO_DJANGO\TEMPLATES
|   base.html
|
\---posts
      index.html

(env) PS C:\Users\hrdon> |
```

Unha vez creada a estrutura de carpetas vamos crear as plantillas. Abrimos o arquivo **base.html** e engadimos o seguinte código:



```
<> base.html X
templates > <> base.html > ...
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <meta http-equiv="X-UA-Compatible" content="ie=edge">
7      <title>{% block title %}Web Xurxo{% endblock %}</title>
8      {% load bootstrap4 %}
9      {% bootstrap_css %}
10     {% load static %}
11     <link rel="stylesheet" href="{% static 'css/blog-home.css' %}">
12 </head>
13 <body>
```

```

14
15     <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
16         <div class="container">
17             <a class="navbar-brand" href="#">Web xurxo</a>
18             <button class="navbar-toggler" type="button" data-toggle="collapse"
19                 data-target="#navbarResponsive" aria-controls="navbarResponsive"
20                 aria-expanded="false" aria-label="Toggle navigation">
21                 <span class="navbar-toggler-icon"></span>
22             </button>
23             <div class="collapse navbar-collapse" id="navbarResponsive">
24                 <ul class="navbar-nav ml-auto">
25                     <li class="nav-item active">
26                         <a class="nav-link" href="#">Blog
27                         <span class="sr-only">(current)</span>
28                         </a>
29                     </li>
30                     <li class="nav-item">
31                         <a class="nav-link" href="#">Sobre min</a>
32                     </li>
33                     <li class="nav-item">
34                         <a class="nav-link" href="#">Login</a>
35                     </li>
36                     <li class="nav-item">
37                         <a class="nav-link" href="#">Registro</a>
38                     </li>
39                 </ul>
40             </div>
41         </div>
42     </nav>
43
44     <div class="container">
45         {% block content %}{% endblock %}
46
47         <!-- Sidebar Widgets Column -->
48         <div class="col-md-4">
49             <!-- Categories Widget -->
50             <div class="card my-4">
51                 <h5 class="card-header">Categorias</h5>
52                 <div class="card-body">
53                     <div class="row">
54                         <div class="col-lg-6">
55                             <ul class="list-unstyled mb-0">
56                                 <li>

```

```

57         <a href="#">Diseño web</a>
58     </li>
59     <li>
60         <a href="#">HTML</a>
61     </li>
62     <li>
63         <a href="#">Cortesía</a>
64     </li>
65 </ul>
66 </div>
67 <div class="col-lg-6">
68     <ul class="list-unstyled mb-0">
69         <li>
70             <a href="#">JavaScript</a>
71         </li>
72         <li>
73             <a href="#">CSS</a>
74         </li>
75         <li>
76             <a href="#">Tutoriales</a>
77         </li>
78     </ul>
79 </div>
80 </div>
81 </div>
82 </div>
83
84
85 <div class="card my-4">
86     <h5 class="card-header">Widget lateral</h5>
87     <div class="card-body">
88         Puedes colocar lo que quieras dentro de estos widgets
89         laterales. Son fáciles de usar e incluyen los nuevos
90         contenedores de tarjetas de Bootstrap 4!
91     </div>
92 </div>
93
94 </div>
95 <footer class="py-5 bg-dark">
96     <div class="container">
97         <p class="m-0 text-center text-white">Copyright &copy; Simple Blog
98         {% now "Y" %}</p>
99     </div>
100 </footer>
101 {% bootstrap_javascript jquery='full' %}
102 </body>
103 </html>

```



Agora que temos a nosa plantilla base vamos ver para que serve cada **elemento** de **Django template** que usamos.

```
<title>{% block title %}Web Xurxo{% endblock %}</title>
```

Na anterior imaxe definimos un bloque de plantilla para o **<title>** que as páxinas fillas poden sobrescribir ao estender **base.html**

```
{% load bootstrap4 %}
{% bootstrap_css %}
```

O código de aquí arriba **carga** a librería **bootstrap4** que instalamos ([figura 6](#)) anteriormente en Django e engade o CSS de Bootstrap a nosa plantilla

- `{% load bootstrap4 %}` » activa as etiquetas da librería django-bootstrap4
- `{% bootstrap_css %}` » inserta automaticamente o enlace do arquivo CSS de Bootstrap

```
{% load static %}
<link rel="stylesheet" href="{% static 'css/blog-home.css' %}">
```

Este código **carga** o sistema de arquivos estáticos de Django e logo inserta no **<link>** a url correcta do arquivo **css/blog-home.css** dentro da nosa carpeta **static**

```
{% bootstrap_javascript jquery='full' %}
```

Este último bloque **inserta** automaticamente o **JavaScript de Bootstrap** na nosa plantilla e ademais carga **jQuery completo** como dependencia.

Agora que xa temos a nosa plantilla base configurada engadiremos o seguinte código a nosa plantilla **index.html**

```

<> index.html X <> base.html
templates > posts > <> index.html > div.row > div.col-md-8
1  {% extends "base.html" %}
2  {% block content %}
3      <div class="row">
4          <div class="col-md-8">
5
6              <h1 class="my-4">Simple Blog
7              |  {% comment %} <small>Other text</small> {% endcomment %}
8              </h1>
9
10             <div class="card mb-4">
11                 
13                 <div class="card-body">
14                     <h2 class="card-title">Post Title</h2>
15                     <p class="card-text">Lorem ipsum dolor sit amet, consectetur
16                     adipisicing elit. Reiciendis aliquid atque, nulla? Quos cum ex
17                     quis soluta, a laboriosam. Dicta expedita corporis animi vero
18                     voluptate voluptatibus possimus, veniam magni quis!</p>
19                     <a href="#" class="btn btn-primary">Read More &rarr;</a>
20                 </div>
21                 <div class="card-footer text-muted">
22                     Posted on January 1, 2017 by
23                     <a href="#">Start Bootstrap</a>
24                 </div>
25             </div>
26
27             <div class="card mb-4">
28                 
30                 <div class="card-body">
31                     <h2 class="card-title">Post Title</h2>
32                     <p class="card-text">Lorem ipsum dolor sit amet, consectetur
33                     adipisicing elit. Reiciendis aliquid atque, nulla? Quos cum ex
34                     quis soluta, a laboriosam. Dicta expedita corporis animi vero
35                     voluptate voluptatibus possimus, veniam magni quis!</p>
36                     <a href="#" class="btn btn-primary">Read More &rarr;</a>
37                 </div>

```

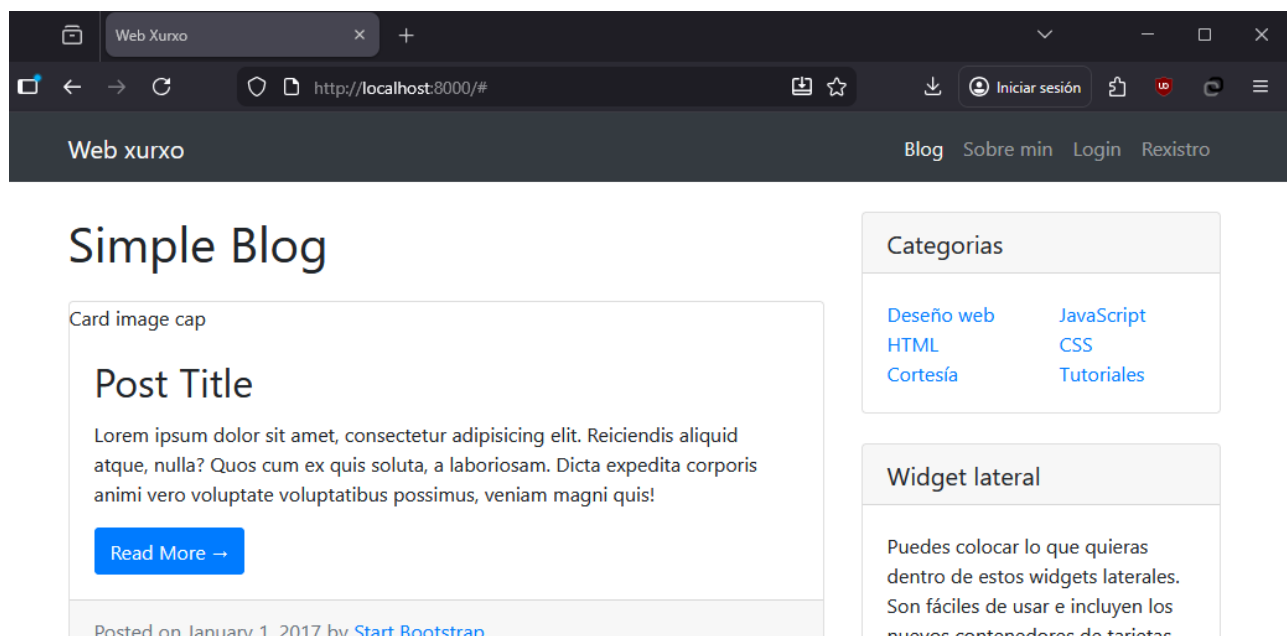
```

30     <div class="card-footer text-muted">
31         Posted on January 1, 2017 by
32         <a href="#">Start Bootstrap</a>
33     </div>
34 </div>
35
36     <ul class="pagination justify-content-center mb-4">
37         <li class="page-item">
38             <a class="page-link" href="#">&larr; Older</a>
39         </li>
40         <li class="page-item disabled">
41             <a class="page-link" href="#">Newer &rarr;</a>
42         </li>
43     </ul>
44
45 </div>
46 {% endblock %}

```

Aquí o que facemos é **estender** da plantilla **base.html** e no bloque **content** engadimos o corpo da nosa páxina.

Agora se arrancamos o servidor e buscamos **localhost:8000** no navegador verase a nosa web



Agora que temos a páxina visible, vamos crear o detalle dun post, para iso dentro de **templates/post** creamos o arquivo **detail.html** e engadimos o seguinte código

```

<> detail.html X
templates > posts > <> detail.html > div.container > div.row
1  {% extends "base.html" %}
2  {% block content %}
3      <div class="container">
4          <div class="row">
5              <div class="col-lg-8">
6                  <h1 class="mt-4">Post Title</h1>
7                  <p class="lead">
8                      by
9                      <a href="#">Start Bootstrap</a>
10                 </p>
11                 <hr>
12                 <p>Posted on January 1, 2019 at 12:00 PM</p>
13                 <hr>
14                 
16                 <hr>
17                 <p class="lead">Lorem ipsum dolor sit amet, consectetur adipisicing
18                     elit. Ducimus, vero, obcaecati, aut, error quam sapiente nemo saepe
19                     quibusdam sit excepturi nam quia corporis eligendi eos magni
20                     recusandae laborum minus inventore?</p>
21                 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ut,
22                     tenetur natus doloremque laborum quos iste ipsum rerum obcaecati
23                     impedit odit illo dolorum ab tempora nihil dicta earum fugiat.
24                     Temporibus, voluptatibus.</p>
25                 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eos,
26                     doloribus, dolorem iusto blanditiis unde eius illum consequuntur
27                     neque dicta incidunt ullam ea hic porro optio ratione repellat
28                     perspiciatis. Enim, iure!</p>
29                 <blockquote class="blockquote">
30                     <p class="mb-0">Lorem ipsum dolor sit amet, consectetur
31                         adipiscing elit. Integer posuere erat a ante.</p>
32                     <footer class="blockquote-footer">Someone famous in
33                         <cite title="Source Title">Source Title</cite>
34                     </footer>
35                 </blockquote>
36                 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Error,
37                     nostrum, aliquid, animi, ut quas placeat totam sunt tempora commodi
38                     nihil ullam alias modi dicta saepe minima ab quo voluptatem
39                     obcaecati?</p>

```

```

26      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Harum,
      dolor quis. Sunt, ut, explicabo, aliquam tenetur ratione tempore
      quidem voluptates cupiditate voluptas illo saepe quaerat numquam
      recusandae? Qui, necessitatibus, est!</p>
27      <hr>
28      <div class="card my-4">
29          <h5 class="card-header">Leave a Comment:</h5>
30          <div class="card-body">
31              <form>
32                  <div class="form-group">
33                      <textarea class="form-control" rows="3"></textarea>
34                  </div>
35                  <button type="submit" class="btn btn-primary">Submit</button>
36              </form>
37          </div>
38      </div>
39      <div class="media mb-4">
40          
42          <div class="media-body">
43              <h5 class="mt-0">Commenter Name</h5>
44              Cras sit amet nibh libero, in gravida nulla. Nulla vel metus
45              scelerisque ante sollicitudin. Cras purus odio, vestibulum in
46              vulputate at, tempus viverra turpis. Fusce condimentum nunc ac
47              nisi vulputate fringilla. Donec lacinia congue felis in
48              faucibus.
          </div>
      </div>
  </div>
{% endblock %}

```

Logo de pegar o código vamos encargarnos da parte de login. O primeiro que vamos facer é: no noso directorio **templates** engadir o cartafol **users**, dentro desta crearemos 3 arquivos que se chamarán **base.html**, **login.html** e **register.html**.

A estrutura de cartafoles debe quedar algo asi:

```
Windows PowerShell
(env) PS C:\Users\hrdon> tree /F /A .\entorno_django\templates\
Listado de rutas de carpetas para el volumen OS
El número de serie del volumen es 8C55-3976
C:\USERS\HRDON\ENTORNO_DJANGO\TEMPLATES
|   base.html
|
+---posts
|   detail.html
|   index.html
|
\---users
    base.html
    login.html
    register.html

(env) PS C:\Users\hrdon> |
```

## 6.2. Post » base.html, login.html y register.html

```

<> base.html X
templates > users > <> base.html > html
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <meta http-equiv="X-UA-Compatible" content="ie=edge">
7      <title>{% block title %}Simple blog - Login{% endblock %}</title>
8      {% load bootstrap4 %}
9      {% bootstrap_css %}
10     {% load static %}
11     <link rel="stylesheet" href="{% static 'css/signin.css' %}">
12 </head>
13 <body class="text-center">
14     {% block content %}{% endblock %}
15     {% bootstrap_javascript jquery='full' %}
16 </body>
17 </html>

```

Figura 7: base.html

```

<> login.html X
templates > users > <> login.html > ...
1  {% extends "users/base.html" %}
2  {% block content %}
3  <form class="form-signin">
4      <h1><a href="#" class="title-link">Simple Blog</a></h1>
5      <h1 class="h3 mb-3 font-weight-normal">Login</h1>
6      <label for="inputEmail" class="sr-only">Email</label>
7      <input type="email" id="inputEmail" class="form-control"
8      placeholder="Email" required autofocus>
9      <label for="inputPassword" class="sr-only">Contraseña</label>
10     <input type="password" id="inputPassword" class="form-control"
11     placeholder="Contraseña" required>
12     <div class="checkbox mb-3"></div>
13     <button class="btn btn-lg btn-primary btn-block" type="submit">Login</
14     button>
15     <p class="mt-5 mb-3 text-muted">&copy; Simple Blog {% now "Y" %}</p>
16 </form>
17 {% endblock %}

```

Figura 8: login.html

```
<> register.html X
templates > users > <> register.html > ...
1  {% extends "users/base.html" %}
2  {% block title %}Simple blog - Registro{% endblock %}
3  {% block content %}
4  <form class="form-signin">
5      <h1><a href="#" class="title-link">Simple Blog</a></h1>
6      <h1 class="h3 mb-3 font-weight-normal">Registro</h1>
7      <label for="inputEmail" class="sr-only">Email</label>
8      <input type="email" id="inputEmail" class="form-control"
9          placeholder="Email address" required autofocus>
10     <label for="inputUsername" class="sr-only">Nombre de usuario</label>
11     <input type="text" id="inputUsername" class="form-control"
12         placeholder="Nombre de usuario" required>
13     <label for="inputPassword" class="sr-only">Contraseña</label>
14     <input type="password" id="inputPassword" class="form-control"
15         placeholder="Contraseña" required>
16     <label for="inputPassword" class="sr-only">Repetir contraseña</label>
17     <input type="password" id="inputPasswordRepeat" class="form-control"
18         placeholder="Repetir contraseña" required>
19     <div class="checkbox mb-3"></div>
20     <button class="btn btn-lg btn-primary btn-block"
21         type="submit">Registrarse</button>
22     <p class="mt-5 mb-3 text-muted">&copy; Simple Blog {% now "Y" %}</p>
23 </form>
24 {% endblock %}
```

Figura 9: register.html



Dentro de **static/css** creamos o arquivo **signin.css** e engadimos o seguinte código:

```
# signin.css X
static > css > # signin.css > ...
1  html,
2  ▼ body {
3      height: 100%;
4  }
5
6  ▼ body {
7      display: -ms-flexbox;
8      display: flex;
9      -ms-flex-align: center;
10     align-items: center;
11     padding-top: 40px;
12     padding-bottom: 40px;
13     background-color: #f5f5f5;
14 }
15
16 ▼ .form-signin {
17     width: 100%;
18     max-width: 330px;
19     padding: 15px;
20     margin: auto;
21 }
22
23 ▼ .form-signin .checkbox {
24     font-weight: 400;
25 }
26
27 ▼ .form-signin .form-control {
28     position: relative;
29     box-sizing: border-box;
30     height: auto;
31     padding: 10px;
32     font-size: 16px;
33 }
34
35 ▼ .form-signin .form-control:focus {
36     z-index: 2;
37 }
38
39 ▼ .form-signin input[type="email"] {
40     border-bottom-right-radius: 0;
41     border-bottom-left-radius: 0;
42 }
```

```
43
44 .form-signin input[type="text"] {
45     border-radius: 0;
46 }
47
48 .form-signin input[type="password"] {
49     border-radius: 0;
50 }
51
52 #inputPasswordRepeat {
53     border-top-left-radius: 0;
54     border-top-right-radius: 0;
55 }
56
57 .title-link {
58     color: #000;
59     cursor: pointer;
60 }
61
62 .title-link:hover {
63     color: #000;
64     text-decoration: none;
65 }
```

Agora só nos falta a paxina **sobre min**, para iso dentro da carpeta **templates** creamos o arquivo **about.html** e engadimos o seguinte código:

```

about.html X
templates > about.html > div.container > div.row
1  {% extends "base.html" %}
2  {% block title %}Simple blog - Sobre mi{% endblock %}
3  {% block content %}
4      <div class="container">
5          <div class="row">
6              <div class="col-lg-8">
7                  <h1 class="mt-4">Sobre mi</h1>
8                  <hr>
9                  
10                 <hr>
11                 <p class="lead">Lorem ipsum dolor sit amet, consectetur adipisicing elit.
12                 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ut, tenetur
13                 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eos, dolorib
14                 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eos, dolorib
15                 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eos, dolorib
16                 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Error, nostr
17                 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Harum, dolor
18                 <hr>
19             </div>
20  {% endblock %}

```

Agora que xa temos as páxinas listas vamonos ao ficheiro **templates/base.html** e engadimos as urls das páxinas na cabeceira co bloque `{% url 'home' %}`

```

<div class="collapse navbar-collapse" id="navbarResponsive">
  <ul class="navbar-nav ml-auto">
    <li class="nav-item active">
      <a class="nav-link" href="{% url 'index' %}">Blog
      <span class="sr-only">(current)</span>
    </a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="{% url 'about' %}">Sobre min</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="{% url 'users:login' %}">Login</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="{% url 'users:register' %}">Registro</a>
    </li>
  </ul>
</div>

```

A sintaxe dos bloques seria a seguinte:

**url** -> refírese ao ficheiro **urls.py** do directorio **web\_xurxo**

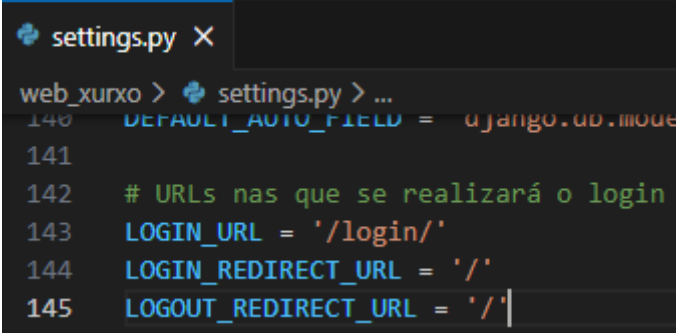
**'index'** -> nome que asignamos no **path** do array **urlpatterns**

**users:register** -> chamamos aos enlaces de **users/urls.py** que incluiremos no noso **web\_xurxo/urls.py** máis adiante

## 7. Login

Para esta parte utilizaremos o sistema de autenticación e autorización que nos prove Django.

Primeiro teremos que indicar as urls nas que se realizará o login, a redirección ao facelo e a redirección ao facer logout, para iso modificaremos o ficheiro **web\_xurxo/settings.py** e ao final deste engadimos o seguinte código:



```
settings.py X
web_xurxo > settings.py > ...
140 DEFAULT_AUTO_FIELD = django.db.models
141
142 # URLs nas que se realizará o login
143 LOGIN_URL = '/login/'
144 LOGIN_REDIRECT_URL = '/'
145 LOGOUT_REDIRECT_URL = '/'
```

O seguinte paso que temos que facer será crear o ficheiro **users/url.py** e mover as urls de login e de rexistro que estaban no ficheiro **web\_xurxo/urls.py**

Aproveitando que temos que modificar as rutas vamos mellorar a lexibilidade do código e vamos cambiar os **path** pola súa forma compacta.

Así quedaría o noso ficheiro **web\_xurxo/urls.py**

```

urls.py  X
web_xurxo > urls.py > ...
1 > """ ...
17 from django.contrib import admin
18 from django.urls import path, include
19 from django.views.generic import TemplateView
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('', TemplateView.as_view(template_name='posts/index.
24         html'), name='index'),
25     path('post/my-post/', TemplateView.as_view(template_name='posts/
26         detail.html'), name='detail'),
27     path('sobre-mi', TemplateView.as_view(template_name='about.
28         html'), name='about'),
29     path('', include(('users.urls', 'users'), namespace='users'))
30 ]

```

E así quedaría **users/urls.py**

```

urls.py  users  X  urls.py  web_xurxo  views.py  1
users > urls.py > ...
1  """Users URLs."""
2
3  # Django
4  from django.urls import path
5  from django.views.generic import TemplateView
6  # View
7  from users import views
8
9  urlpatterns = [
10     path('login', views.LoginView.as_view(), name='login'),
11     path('registro', views.SignupView.as_view(), name='register'),
12     path('logout/', views.LogoutView.as_view(), name='logout'),
13     path('registro_completado/', TemplateView.as_view
14         (template_name='users/registerok.html'), name='registerok'),
15 ]

```

O que acabamos de facer foi modificar as urls de **login** e **rexistro** para que apunten aos ficheiros **user/views.py** e ás clases que lles darán a funcionalidade.

Tamén creamos a ruta para o rexistro completo que apunta a unha plantilla que crearemos no seguinte paso e o logout.

## 7.1. Formulario de rexistro

Creamos o ficheiro **users/forms.py** e pegamos o seguinte código:

```
forms.py X
users > forms.py > SignupForm
1  """User forms."""
2
3  # Django
4  from django import forms
5
6  # Models
7  from django.contrib.auth.models import User
8  from users.models import Profile
9
10
11 class SignupForm(forms.Form):
12     """Sign up form."""
13
14     email = forms.CharField(
15         min_length=6,
16         max_length=70,
17         widget=forms.EmailInput()
18     )
19     username = forms.CharField(
20         min_length=6,
21         max_length=70,
22         widget=forms.TextInput()
23     )
24     password = forms.CharField(
25         max_length=70,
26         widget=forms.PasswordInput()
27     )
28     password_confirmation = forms.CharField(
29         max_length=70,
30         widget=forms.PasswordInput()
31     )
```

```

32
33
34     def clean(self):
35         """Verify password confirmation match."""
36         data = super().clean()
37
38         password = data['password']
39         password_confirmation = data['password_confirmation']
40
41         if password != password_confirmation:
42             raise forms.ValidationError('Las contraseñas no coinciden.')
43
44         return data
45
46     def save(self):
47         """Create user and profile."""
48         data = self.cleaned_data
49         data.pop('password_confirmation')
50
51         user = User.objects.create_user(**data)
52         profile = Profile(user=user)
53         profile.save()

```

Neste código o que facemos é declarar a clase **SignupForm** que conterá todos os campos do noso formulario. Co campo **widget** declaramos o **tipo de input**, neste caso todos os campos son de tipo texto, excepto os campos email e password.

A continuación declaramos a función **clean** que se executará despois de enviar o formulario e validará se o contrasinal é igual ao contrasinal repetido.

Por último cramos a función **save** que se encarga de eliminar o campo **password\_confirmation** e gardar a información enviada na táboa **User** e **Profile**.

Agora vamos encargarnos da parte visual, abrimos o ficheiro **users/views.py** e pegamos o seguinte código:

```
views.py X signals.py forms.py
users > views.py > LogoutView
1  from django.shortcuts import redirect
2  from django.views.generic import FormView
3  from django.urls import reverse_lazy
4  from django.contrib.auth.mixins import LoginRequiredMixin
5  from django.contrib.auth import views as auth_views
6
7  # Forms
8  from users.forms import SignupForm
9
10 class SignupView(FormView):
11     """Users sign up view."""
12
13     template_name = 'users/register.html'
14     form_class = SignupForm
15     success_url = reverse_lazy('users:registerok')
16
17     def form_valid(self, form):
18         """Save form data."""
19         form.save()
20         return redirect(self.success_url)
21
22 class LoginView(auth_views.LoginView):
23     """Login view."""
24     template_name = 'users/login.html'
25
26
27 class LogoutView(LoginRequiredMixin, auth_views.LogoutView):
28     """Logout view."""
29     template_name = 'users/logged_out.html'
```

No **template\_name** gardamos a plantilla que vamos utilizar.

No **form\_class** gardamos o formulario que creamos anteriormente.

**success\_url** será a url á que redireccionaremos ó usuario se todo foi ben.

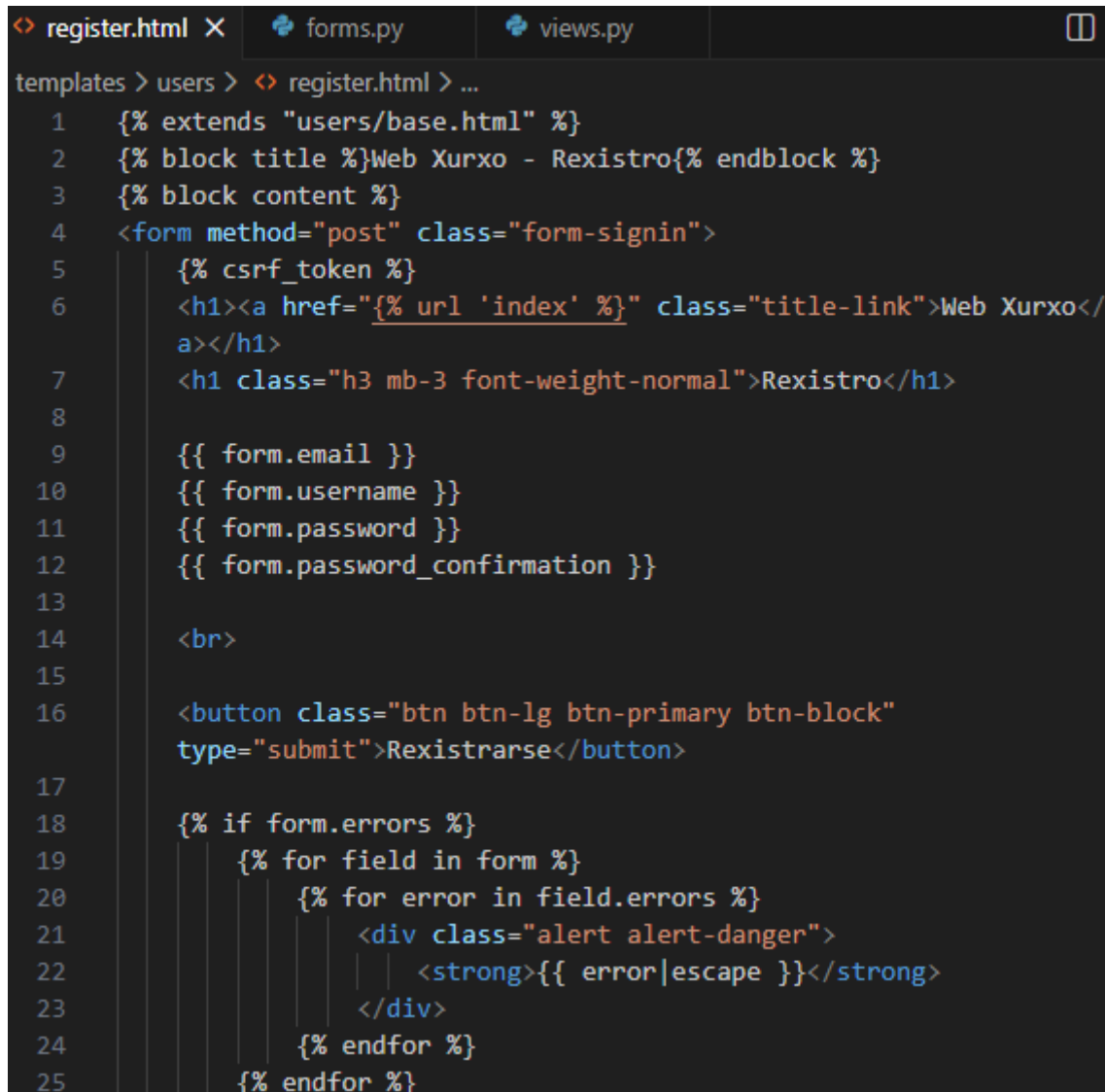
Logo usamos **form\_valid()** para gardar o usuario se todo vai ben.

Á clase **LoginView** asignámoslle unha plantilla e toda a lóxica encárgase de facela Django.

No caso do **logout** é exactamente igual aínda que como neste caso non necesita plantilla engadimos unha (aínda que non exista).

Agora toca modificar a plantilla que creamos para o rexistro no punto [6. Deseño](#).

Abre o ficheiro **templates/users/register.html** e pega o seguinte código:



```
1  {% extends "users/base.html" %}
2  {% block title %}Web Xurxo - Rexistro{% endblock %}
3  {% block content %}
4  <form method="post" class="form-signin">
5      {% csrf_token %}
6      <h1><a href="{% url 'index' %}" class="title-link">Web Xurxo</a></h1>
7      <h1 class="h3 mb-3 font-weight-normal">Rexistro</h1>
8
9      {{ form.email }}
10     {{ form.username }}
11     {{ form.password }}
12     {{ form.password_confirmation }}
13
14     <br>
15
16     <button class="btn btn-lg btn-primary btn-block"
17         type="submit">Rexistrarse</button>
18
19     {% if form.errors %}
20         {% for field in form %}
21             {% for error in field.errors %}
22                 <div class="alert alert-danger">
23                     <strong>{{ error|escape }}</strong>
24                 </div>
25             {% endfor %}
26         {% endfor %}
```



```

26         {% for error in form.non_field_errors %}
27             <div class="alert alert-danger">
28                 <strong>{{ error|escape }}</strong>
29             </div>
30         {% endfor %}
31     {% endif %}
32
33     <p class="mt-5 mb-3 text-muted">&copy; Web Xurxo {% now "Y" %}</p>
34 </form>
35 {% endblock %}

```

Engadimos o **csrf\_token** que necesitaremos para validar o formulario e se hay erros ao crear usuarios amosará unha alerta.

Agora os campos do formularios creamolos co obxecto **form.campo** que é unha clase que creamos no **views.py** donde instanciamos o obxecto da clase **SignupForm** e **LoginForm**

Se todo sae ben á hora de crear un usuario enviáremolo a unha plantilla avisando de que o usuario foi creado correctamente, pera iso creamos o ficheiro **templates/users/registerok.html** e engadimos o seguinte código:

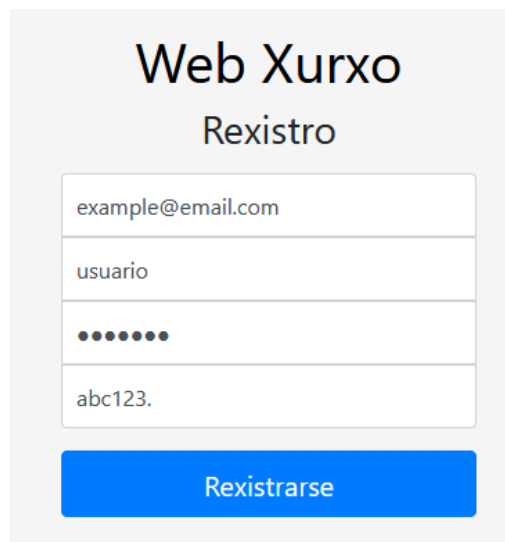
```

<> registerok.html X <> register.html forms.py views.py
templates > users > <> registerok.html > ...
1  {% extends "users/base.html" %}
2  {% block title %}Web Xurxo - Rexistro completado{% endblock %}
3  {% block content %}
4  <main role="main" class="container">
5      <div class="starter-template">
6          <h1>O teu usuario foi creado con éxito</h1>
7          <p class="lead">Se a páxina non é redirixida en 5 segundos
8              pulsa <a href="{% url 'users:login' %}">aquí</a></p>
9      </div>
10 </main>
11 <script>
12     setTimeout(function(){
13         window.location.replace("{% url 'users:login' %}");
14     }, 5000);
15 </script>
16 {% endblock %}

```

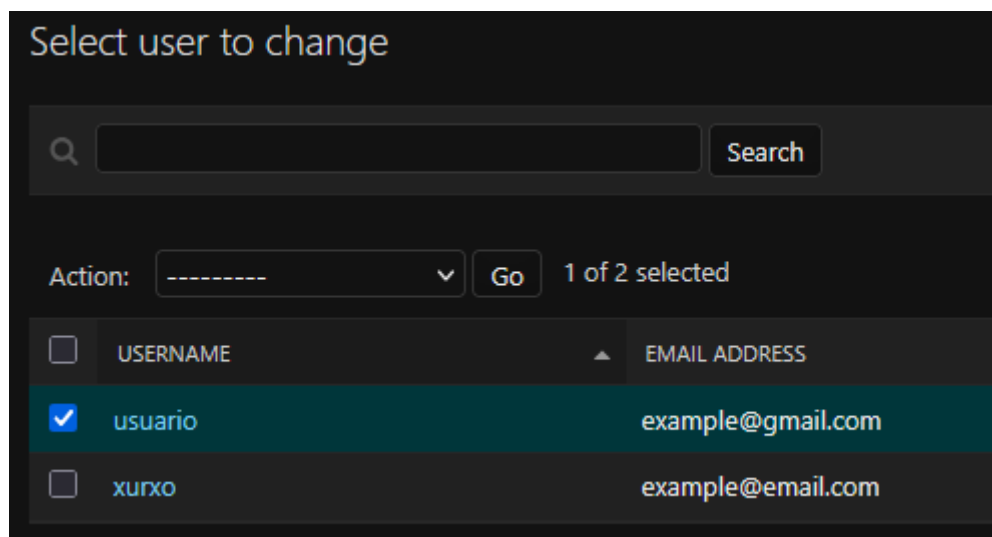
Amosamos unha mensaxe e redireccionamos á paxina de login utilizando `{% url 'user:login' %}`, isto mira nos nosos arquivos `urls.py` e xera unha url en base ao `namespace` e `name` que engadimos as nosas urls.

Para acceder ao rexistro entramos <http://127.0.0.1:8000/register> e probamos a crear un usuario para crear probas posteriormente



The image shows a registration form titled "Web Xurxo" and "Registro". It contains four input fields: an email field with "example@email.com", a username field with "usuario", a password field with masked characters "••••••", and a confirmation field with "abc123.". Below the fields is a blue button labeled "Rexistrarse".

Comprobamos que estea creado na parte de administración e listo:



The image shows the Django admin interface for selecting a user to change. It features a search bar, an "Action:" dropdown menu, and a "Go" button. Below the search bar, there is a table with two columns: "USERNAME" and "EMAIL ADDRESS". The table lists two users: "usuario" with email "example@gmail.com" and "xurxo" with email "example@email.com". The "usuario" row is highlighted in green, and its checkbox is checked.

	USERNAME	EMAIL ADDRESS
<input checked="" type="checkbox"/>	usuario	example@gmail.com
<input type="checkbox"/>	xurxo	example@email.com

Para rematar abrimos o ficheiro `templates/users/login.html` e modificamos o código polo seguinte:

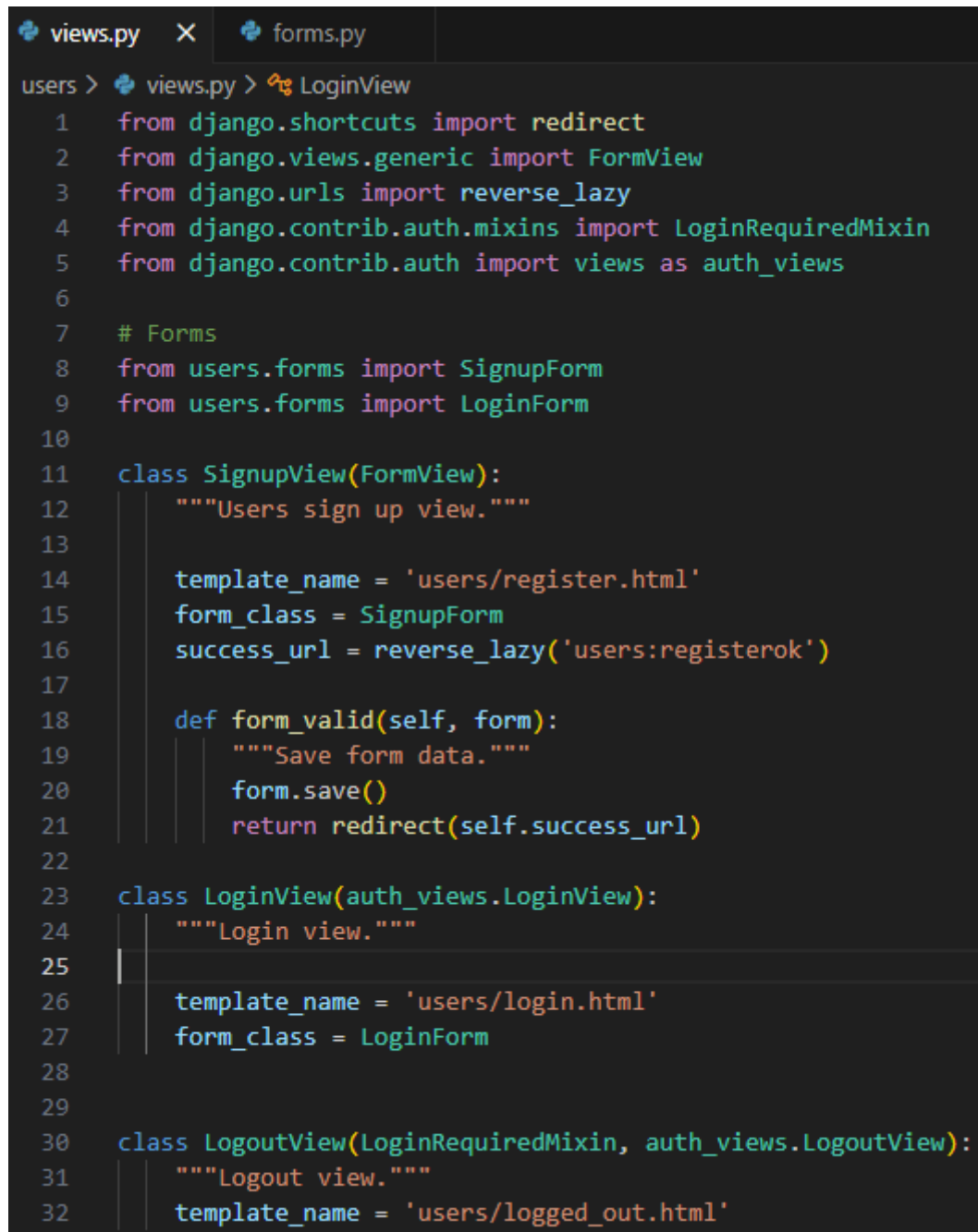
```

login.html X views.py forms.py
templates > users > login.html > form.form-signin > br
1  {% extends "users/base.html" %}
2  {% block content %}
3  <form method="POST" action="{% url 'users:login' %}"
    class="form-signin">
4      {% csrf_token %}
5      <h1><a href="{% url 'index' %}" class="title-link">Web Xurxo</a></
    h1>
6      <h1 class="h3 mb-3 font-weight-normal">Login</h1>
7
8      {{ form.username }}
9      {{ form.password }}
10
11     <br>
12
13     <button class="btn btn-lg btn-primary btn-block"
        type="submit">Login</button>
14     {% if form.errors %}
15         {% for field in form %}
16             {% for error in field.errors %}
17                 <div class="alert alert-danger">
18                     <strong>{{ error|escape }}</strong>
19                 </div>
20             {% endfor %}
21         {% endfor %}
22         {% for error in form.non_field_errors %}
23             <div class="alert alert-danger">
24                 <strong>{{ error|escape }}</strong>
25             </div>
26         {% endfor %}
27     {% endif %}
28     <p class="mt-5 mb-3 text-muted">&copy; Web Xurxo {% now "Y" %}</p>
29 </form>
30 {% endblock %}

```

Ao igual que na plantilla de rexistro engadimos o **csrf\_token** e se hay erros capturámoslos.

Ademais tamén chamamos os campos do formulario dende o **views.py**, que falando del debería quedar así o noso **users/views.py**



```
views.py X forms.py
users > views.py > LoginView
1  from django.shortcuts import redirect
2  from django.views.generic import FormView
3  from django.urls import reverse_lazy
4  from django.contrib.auth.mixins import LoginRequiredMixin
5  from django.contrib.auth import views as auth_views
6
7  # Forms
8  from users.forms import SignupForm
9  from users.forms import LoginForm
10
11 class SignupView(FormView):
12     """Users sign up view."""
13
14     template_name = 'users/register.html'
15     form_class = SignupForm
16     success_url = reverse_lazy('users:registerok')
17
18     def form_valid(self, form):
19         """Save form data."""
20         form.save()
21         return redirect(self.success_url)
22
23 class LoginView(auth_views.LoginView):
24     """Login view."""
25
26     template_name = 'users/login.html'
27     form_class = LoginForm
28
29
30 class LogoutView(LoginRequiredMixin, auth_views.LogoutView):
31     """Logout view."""
32     template_name = 'users/logged_out.html'
```

E o noso **users/forms.py** terá que quedar así:

```
forms.py X
users > forms.py > LoginForm
1  """User forms."""
2
3  # Django
4  from django import forms
5  from django.contrib.auth.forms import AuthenticationForm
6
7  # Models
8  from django.contrib.auth.models import User
9  # from users.models import Profile
10
11 # Clase cos campos do formulario de rexistro
12 class SignupForm(forms.Form):
13     """Sign up form."""
14
15     email = forms.CharField(
16         min_length=6,
17         max_length=70,
18         widget=forms.EmailInput(attrs={
19             'id': 'inputEmail',
20             'class': 'form-control',
21             'placeholder': 'Correo electrónico'
22         })
23     )
24     username = forms.CharField(
25         min_length=6,
26         max_length=70,
27         widget=forms.TextInput(attrs={
28             'id': 'inputUsername',
29             'class': 'form-control',
30             'placeholder': 'Nome de usuario',
31         }),
32     )
33     password = forms.CharField(
34         max_length=70,
35         widget=forms.PasswordInput(attrs={
36             'id': 'inputPassword',
37             'class': 'form-control',
38             'placeholder': 'Contrasinal',
39             'auto-complete': 'new-password'
40         })
41     )
```

```

41         )
42         password_confirmation = forms.CharField(
43             max_length=70,
44             widget=forms.PasswordInput(attrs={
45                 'id': 'inputPasswordRepeat',
46                 'class': 'form-control',
47                 'placeholder': 'Repetir contrasinal'
48             })
49         )
50
51
52     def clean(self):
53         """Verify password confirmation match."""
54         data = super().clean()
55
56         password = data['password']
57         password_confirmation = data['password_confirmation']
58
59         if password != password_confirmation:
60             raise forms.ValidationError('Os contrasinais non coinciden.')
61
62         return data
63
64     def save(self):
65         """Create user and profile."""
66         data = self.cleaned_data
67         data.pop('password_confirmation')
68         user = User.objects.create_user(**data)
69         return user
70
71 # Clase cos campos do formulario de Login
72 class LoginForm(AuthenticationForm):
73     """Login Form"""
74
75     username = forms.CharField(
76         min_length=6,
77         max_length=70,
78         widget=forms.TextInput(attrs={
79             'id': 'id_username',
80             'class': 'form-control',
81             'placeholder': 'Usuario',
82         }),
83     )

```

```

84     password = forms.CharField(
85         max_length=70,
86         widget=forms.PasswordInput(attrs={
87             'id': 'id_password',
88             'class': 'form-control',
89             'placeholder': 'Contraseña',
90         })
91     )

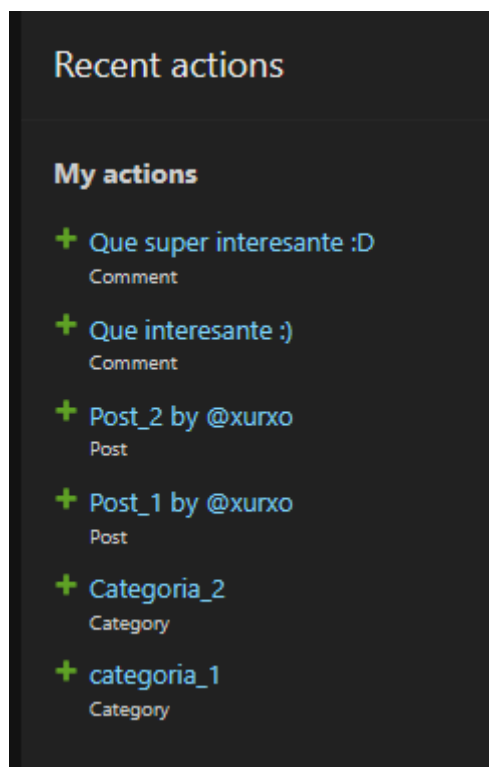
```

## 8. Vistas (Views)

Neste último paso vamos conectar todas las partes de nuestra aplicación. O primeiro que faremos será encher con datos a nosa **base de datos**.

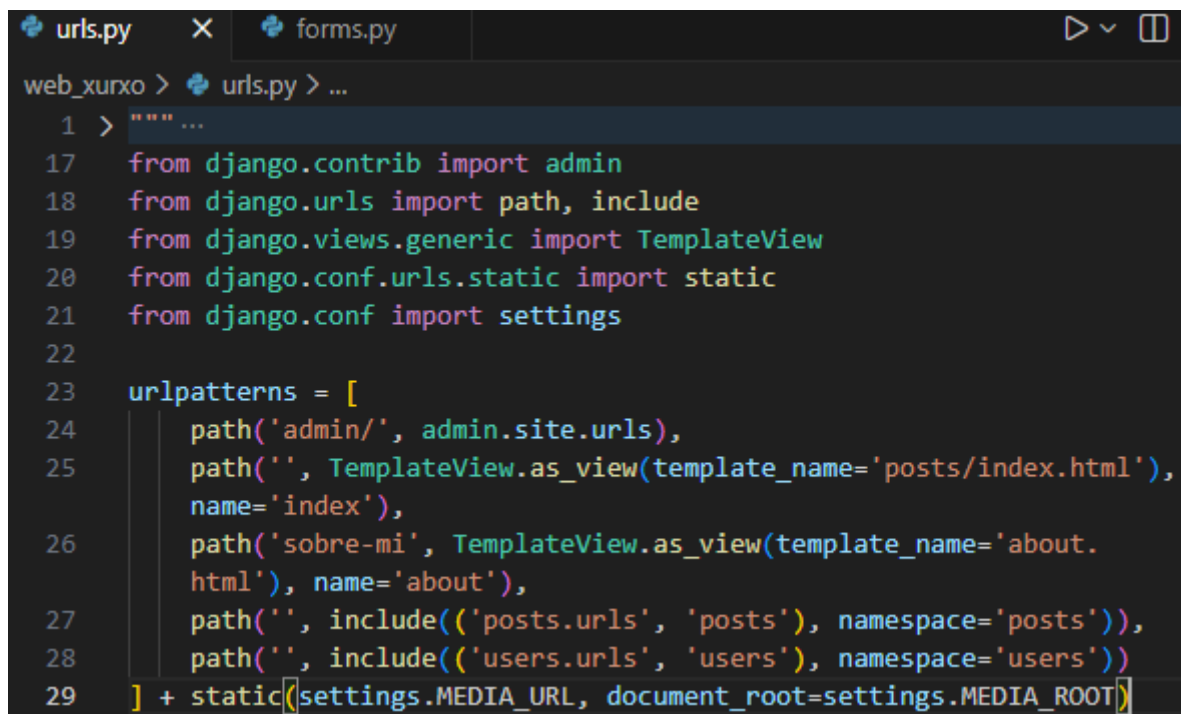
Antes de crear os **posts** teremos que crear o directorio **media** dentro do noso entorno e logo dentro engadimos **posts/photos** que será o cartafol no que gardaremos as fotos para amosalas na nosa web.

Para iso vámonos a **localhost:8000/admin** e agora engadimos algúns post, categorías, comentarios...



Recorda **desmarcar o cuadro is\_draft** para que os **posts** se amosen no blog, senón Django entende que son borradores e non se amosarán.

Agora vamos cambiar as urls para que apunten á vista que crearemos no seguinte paso. Abrimos o ficheiro **web\_xurxo/urls.py** e modificamos o código que temos polo seguinte:



```

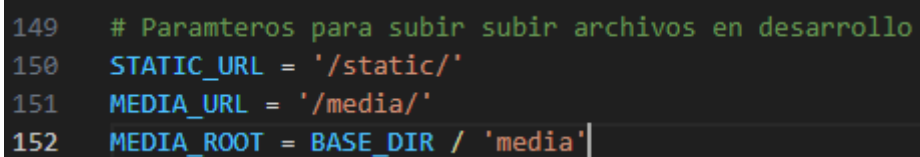
1 > """ ...
17 from django.contrib import admin
18 from django.urls import path, include
19 from django.views.generic import TemplateView
20 from django.conf.urls.static import static
21 from django.conf import settings
22
23 urlpatterns = [
24     path('admin/', admin.site.urls),
25     path('', TemplateView.as_view(template_name='posts/index.html'),
26         name='index'),
27     path('sobre-mi', TemplateView.as_view(template_name='about.
28         html'), name='about'),
29     path('', include(('posts.urls', 'posts'), namespace='posts')),
30     path('', include(('users.urls', 'users'), namespace='users')),
31 ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

Como se pode ver fixemos para os enlaces de **posts** o mesmo que cos enlaces de **users**. Agora vamos gardar tódalas urls de posts dentro de **posts/urls.py**.

A liña **static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)** que xa estaba en settings e se utiliza para **acceder ós arquivos** aloxados na carpeta **media**.

Logo dirixímonos ao ficheiro **web\_xurxo/settings.py** e no fondo do arquivo engadimos este código:



```

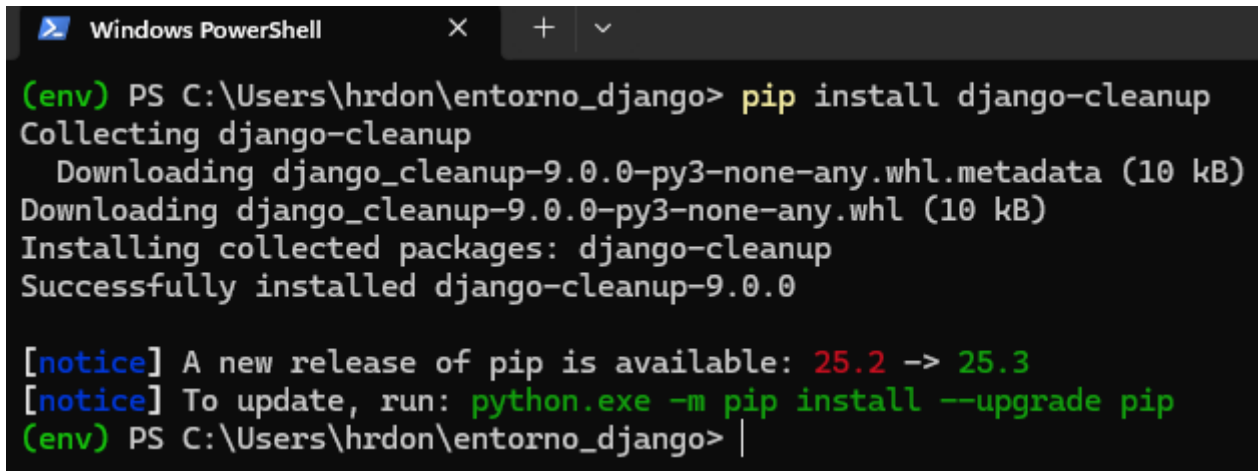
149 # Paramteros para subir subir archivos en desarrollo
150 STATIC_URL = '/static/'
151 MEDIA_URL = '/media/'
152 MEDIA_ROOT = BASE_DIR / 'media'

```



Por último para non deixar lixo no servidor instalaremos **django-cleanup**.

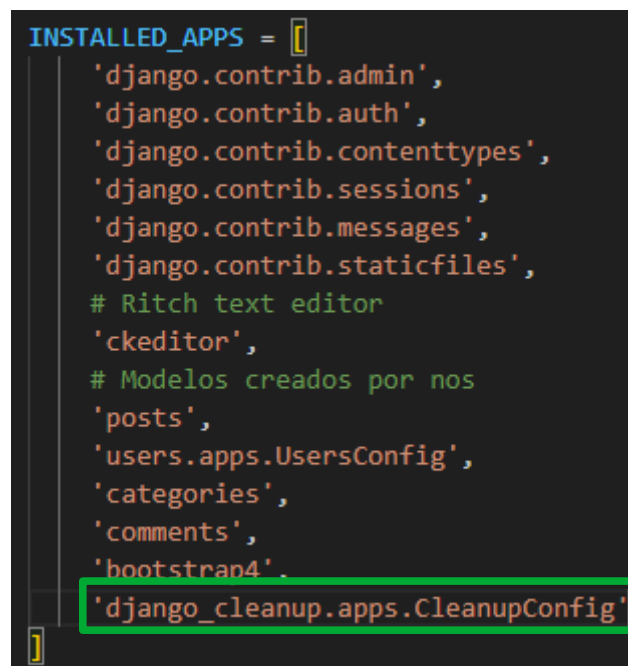
Abrimos a terminal co noso contorno activo e executa o comando **pip install django-cleanup**



```
(env) PS C:\Users\hrdon\entorno_django> pip install django-cleanup
Collecting django-cleanup
  Downloading django_cleanup-9.0.0-py3-none-any.whl.metadata (10 kB)
  Downloading django_cleanup-9.0.0-py3-none-any.whl (10 kB)
Installing collected packages: django-cleanup
Successfully installed django-cleanup-9.0.0

[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
(env) PS C:\Users\hrdon\entorno_django> |
```

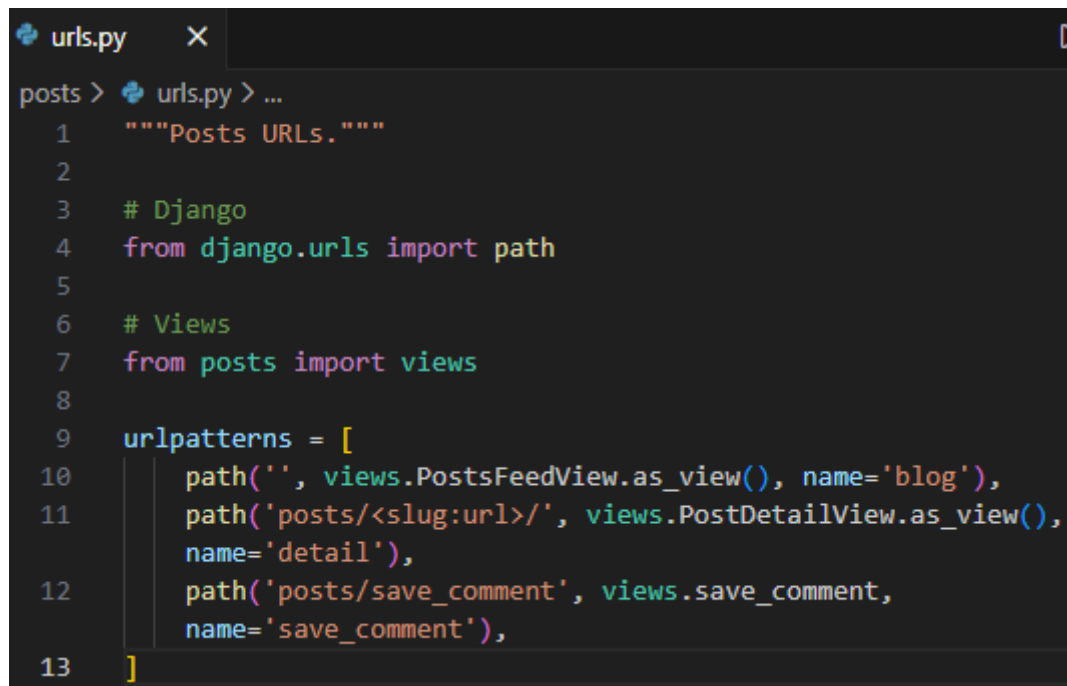
Unha vez instalado engadímoloo ao **INSTALLED\_APPS**



```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Ritch text editor
    'ckeditor',
    # Modelos creados por nos
    'posts',
    'users.apps.UsersConfig',
    'categories',
    'comments',
    'bootstrap4',
    'django_cleanup.apps.CleanupConfig'
```

E listo, agora se **eliminamos** un post tamén se eliminan as imaxes do cartafol **media**

Despois creamos o ficheiro **posts/urls.py** e engadimos o seguinte código para que as nosas urls apunten ás funcións que vamos crear nas nosas vistas:



```
posts > urls.py > ...
1  """Posts URLs."""
2
3  # Django
4  from django.urls import path
5
6  # Views
7  from posts import views
8
9  urlpatterns = [
10     path('', views.PostsFeedView.as_view(), name='blog'),
11     path('posts/<slug:url>/', views.PostDetailView.as_view(),
12         name='detail'),
13     path('posts/save_comment', views.save_comment,
14         name='save_comment'),
15 ]
```

Acabamos de crear 3 urls:

1. Para a páxina principal.
2. Será o detalle do post e no que usaremos o método **slug** para recuperar a última parte do path da url.
3. Encargarase de gardar os comentarios.

Antes de poñernos coa vista necesitamos crear un formulario para posteriormen gardar os comentarios que os usuarios escriban nos nosos posts, así que crearemos o arquivo **comments/forms.py** e insertamos o código que aparece na seguinte páxina:

```

forms.py X
comments > forms.py > CreateCommentForm > Meta
1  """User forms."""
2
3  # Django
4  from django import forms
5
6  # Models
7  from comments.models import Comment
8  from django.contrib.auth.models import User
9
10 class CreateCommentForm(forms.ModelForm):
11     """Post model form."""
12
13     comment = forms.CharField(widget=forms.Textarea)
14
15     class Meta:
16         """Form settings."""
17
18         model = Comment
19         fields = ('user', 'profile', 'post', 'comment')

```

Con este código estamos creando un campo de tipo **textarea** que será o que garde o comentario do usuario. Na clase **Meta** indicamos o modelo de referencia e os campos que ten que gardar, neste caso o usuario que o escribiu, o post ao que vai referenciado e o comentario.

Agora que temos algo de información que amosar vamos recuperala na vista para que se amoso lo na nosa web, para iso editamos **post/views.py** e engadimos este código:

```

views.py X
posts > views.py > save_comment
1  from django.shortcuts import render, redirect
2  from django.views.generic import DetailView, ListView
3  from django.contrib.auth.decorators import login_required
4  from django.http import HttpResponseRedirect
5
6  # Models
7  from posts.models import Post
8  from categories.models import Category
9  from comments.models import Comment
10

```

```
11 # Forms
12 from comments.forms import CreateCommentForm
13
14
15 class PostsFeedView(ListView):
16     """Index."""
17     template_name = 'posts/index.html'
18     model = Post
19     ordering = ('-created',)
20     paginate_by = 10
21     context_object_name = 'posts'
22     queryset = Post.objects.filter(is_draft=False)
23
24
25     def get_context_data(self, **kwargs):
26         context = super().get_context_data(**kwargs)
27         context['categories'] = Category.objects.all()
28         return context
29
30
31 class PostDetailView(DetailView):
32     """Detail post."""
33     template_name = 'posts/detail.html'
34     model = Post
35     context_object_name = 'post'
36     slug_field = 'url'
37     slug_url_kwarg = 'url'
38
39
40     def get_queryset(self):
41         return Post.objects.filter(is_draft=False)
42
43
44     def get_context_data(self, **kwargs):
45         context = super().get_context_data(**kwargs)
46         context['categories'] = Category.objects.all()
47         context['comments'] = Comment.objects.filter(post=self.get_object()).all()
48         context['form_comments'] = CreateCommentForm()
49         return context
50
51
```

```

52  @login_required
53  def save_comment(request):
54      if request.method == 'POST':
55          url = request.POST['url']
56          post = {
57              'user': request.user.id,
58              'profile': request.user.id,
59              'comment': request.POST['comment'],
60              'post': request.POST['post']
61          }
62          form = CreateCommentForm(post)
63          if form.is_valid():
64              form.save()
65              return redirect('posts:detail', url=url)
66          else:
67              return HttpResponse(status=405)
68          return HttpResponse(status=500)

```

Neste arquivo temos as tres vistas ás que se chaman dende as urls, vamos explicar que fan:

O que estamos a facer na clase **PostsFeedView** é utilizar unha das vistas predeterminadas que ten Django, neste caso **ListView** que permite retornar unha listas de obxectos:

- **template\_name**: A plantilla que será a que reciba toda a información.
- **model**: O modelo que vamos utilizar, neste caso **Post**.
- **Ordering**: A orde polo que queremos que aparezan os posts, neste caso queremos que aparezan por orde de creación de forma descendente.
- **paginate\_by**: O número de posts que se amosarán por páxina.
- **queryset**: Para engadir condicións adicionais, non queremos que se amosen os borradores así que descartámoslos.

A parte utilizamos a función **get\_context\_data** para engadir información extra a nosa vista, neste caso as categorías para amosalas no menú lateral.

A clase chamada **PostDetailView** é de tipo **DetailView** e encargase de amosar un post concreto, para iso recupera o título na url e utilízao a modo de consulta. Para recuperalo utiliza **slug\_field = 'url'** e **slug\_url\_kwarg = 'url'**

Neste código tamén utilizamos a función **get\_context\_data** para recoller as categorías e os comentarios que poida ter o post ademais do formulario que creamos anteriormente para gardar comentarios e que só se amosará en caso de que o usuario esté logeado.

Por último temos a función **saveComment** co decorado **@login\_required** que nos obriga a estar logeados para poder gardar un comentario. Tamén revisamos que a chamada a faga mediante o método **POST** e senon devolvemos un erro. Se todo foi ben gardamos e recargamos a páxina na que estamos.

Agora que temos a vista completada vamos darlle un pouco de funcionalidade a todos os enlaces da web e amosar os posts e categorías, para iso abrimos o noso arquivo **templates/base.html** e substituímos o código polo seguinte:

```

<> base.html X
templates > <> base.html > html > body > footer.py-5.bg-dark > div.container > p.m-0
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width,
6          initial-scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      <link rel="icon" type="image/jpg" href="https://png.pngtree.com/
9          png-clipart/20190630/original/
10         pngtree-vector-web-icon-png-image_4141376.jpg"/>
11      <title>{% block title %} Web Xurxo {% endblock %}</title>
12      {% load bootstrap4 %}
13      {% load static %}
14      <link rel="stylesheet" href="{% static 'css/blog-home.css' %}">
15  </head>
16  <body>
17      <!-- Navigation -->
18      <nav class="navbar navbar-expand-lg navbar-dark bg-dark
19          fixed-top">
20          <div class="container">
21              <a class="navbar-brand" href="{% url 'posts:blog' %}">Web
22              Xurxo</a>

```

```

19     <button class="navbar-toggler" type="button"
20         data-toggle="collapse" data-target="#navbarResponsive"
21         aria-controls="navbarResponsive" aria-expanded="false"
22         aria-label="Toggle navigation">
23         <span class="navbar-toggler-icon"></span>
24     </button>
25     <div class="collapse navbar-collapse"
26         id="navbarResponsive">
27         <ul class="navbar-nav ml-auto">
28             <li class="nav-item active">
29                 <a class="nav-link" href="{% url 'posts:blog' %}"
30                 >Blog
31                 <span class="sr-only">(current)</span>
32             </a>
33             </li>
34             <li class="nav-item">
35                 <a class="nav-link" href="{% url 'about' %}"
36                 >Sobre min</a>
37             </li>
38             {% if user.is_authenticated %}
39             <li class="nav-item dropdown">
40                 <a class="nav-link dropdown-toggle"
41                 data-toggle="dropdown" href="#" role="button"
42                 aria-haspopup="true" aria-expanded="false">
43                     {{request.user.username}}</a>
44                 <div class="dropdown-menu">
45                     <a class="dropdown-item" href="{% url
46                     'users:logout' %}">Logout</a>
47                 </div>
48             </li>
49             {% else %}
50             <li class="nav-item">
51                 <a class="nav-link" href="{% url 'users:login' %}"
52                 >Login</a>
53             </li>
54             <li class="nav-item">
55                 <a class="nav-link" href="{% url
56                 'users:register' %}">Rexistro</a>
57             </li>
58             {% endif %}
59         </ul>
60     </div>
61 </div>
62 </nav>

```

```

51     <!-- Page Content -->
52     <div class="container">
53         {% block content %}{% endblock %}
54         <!-- Sidebar Widgets Column -->
55         <div class="col-md-4">
56             <!-- Categories Widget -->
57             <div class="card my-4">
58                 <h5 class="card-header">Categorias</h5>
59                 <div class="card-body">
60                     <div class="row">
61                         <div class="col-lg-12">
62                             <ul class="list-unstyled mb-0">
63                                 {% for category in categories %}
64                                 <li>
65                                     <a href="#">{{category.name}}</a>
66                                 </li>
67                                 {% endfor %}
68                             </ul>
69                         </div>
70                     </div>
71                 </div>
72             </div>
73             <!-- Side Widget -->
74             <div class="card my-4">
75                 <h5 class="card-header">Widget lateral</h5>
76                 <div class="card-body">
77                     Podes colocar o que quieras dentro destes widgets
78                     laterais. Son fáciles de usar e incluen os novos
79                     contedores de tarxetas de Bootstrap 4!
80                 </div>
81             </div>
82         </div>
83         <!-- Footer -->
84         <footer class="py-5 bg-dark">
85             <div class="container">
86                 <p class="m-0 text-center text-white">Copyright &copy;
87                 Web Xurxo {% now "Y" %}</p>
88             </div>
89             <!-- /.container -->
90         </footer>
91         {% bootstrap_javascript jquery='full' %}
92     </div>
93 </body>
94 </html>

```



Os **cambios** máis **importantes** que fixemos son as **urls**, unha condición para **comprobar se o usuario está logeado** con `{% if user.is_authenticated %}` e se está así pintamos o botón coa opción de **logout**, senon a opción de ir á pantalla de rexistro e de login. Tamén listamos as categorías.

Agora que xa temos a base completa abrimos o arquivo **templates/posts/index.html** e modificamos o contido polo seguinte:

```

<> index.html X
templates > posts > <> index.html > div.row
2  {% block content %}
3      <div class="row">
4          <div class="col-md-8">
5              <h1 class="my-4">Web Xurxo
6              </h1>
7              {% for post in posts %}
8              <div class="card mb-4">
9                  
11                  <div class="card-body">
12                      <h2 class="card-title">{{post.title}}</h2>
13                      <p class="card-text">{{post.post|safe|truncatechars:300}}
14                      </p>
15                      <a href="{% url 'posts:detail' post.url%}" class="btn
16                      btn-primary">Leer máis &rarr;</a>
17                  </div>
18                  <div class="card-footer text-muted">
19                      Escrito o {{post.created}}
20                      <a href="#">{{post.user.username}}</a>
21                  </div>
22              </div>
23              {% endfor %}
24              <ul class="pagination justify-content-center mb-4">
25                  <li class="page-item">
26                      <a class="page-link" href="#">&larr; Anterior</a>
27                  </li>
28                  <li class="page-item disabled">
29                      <a class="page-link" href="#">Seguinte &rarr;</a>
30                  </li>
31              </ul>
32          </div>
33      </div>
34  {% endblock %}

```

Acabamos de engadir os datos dos posts e substituílos polos que tiñamos de proba. O máis destacable sería o enlace ao detalle que é este `{% url 'posts:detail' post.url %}`. O que estamos facendo é coller a url do detalle e concatenarlle a url que xeramos con slug para crear o enlace ao detalle do post.

Por último editamos o arquivo `templates/posts/details.html` e substituímos o código existente por este:

```

<> detail.html X
templates > posts > <> detail.html > ...
1  {% extends "base.html" %}
2  {% block content %}
3      <div class="container">
4          <div class="row">
5              <div class="col-lg-8">
6                  <h1 class="mt-4">{{post.title}}</h1>
7                  <p class="lead">
8                      por
9                      <a href="#">{{post.user.username}}</a>
10                 </p>
11                 <hr>
12                 <p>Escrito o {{post.created}}</p>
13                 <hr>
14                 
16                 <hr>
17                 <p class="lead">{{post.post|safe}}</p>
18                 <hr>
19                 {% if user.is_authenticated %}
20                 <div class="card my-4">
21                     <h5 class="card-header">Deixa un comentario:</h5>
22                     <div class="card-body">
23                         <form method="POST" action="{% url 'posts:save_comment' %}">
24                             <input type="hidden" name="url" value="{{post.url}}" />
25                             <input type="hidden" name="post" value="{{post.id}}" />
26                             <div class="form-group">
27                                 <textarea class="form-control" name="comment"
28                                     rows="3"></textarea>
29                             </div>
30                             <button type="submit" class="btn btn-primary">Enviar</
31                             button>
32                         </form>
33                     </div>
34                 </div>
35                 {% endif %}
36             </div>
37         </div>
38     </div>

```

```

33     {% endif %}
34     {% for comment in comments %}
35         <div class="media mb-4">
36             {% if comment.user.profile.photo %}
37                 
38             {% else %}
39                 
40             {% endif %}
41             <div class="media-body">
42                 <h5 class="mt-0">{{ comment.user.username }}</h5>
43                 {{ comment.comment }}
44             </div>
45         </div>
46     {% endfor %}
47 </div>
48 {% endblock %}

```

Ó igual que no index engadimos os datos do post e modificamos o formulario para poder gardar os comentarios que escriba o usuario. Tamén se amosa a lista de tódolos comentarios se os ten o post.

Con isto xa teríamos todo o básico, aínda faltarían algunhas cousas como a paxinación, o login co email e a validación do email cando un novo usuario se rexistra.

## Fontes

- Aprende Django desde 0 -> [cosasdedevs.com](https://cosasdedevs.com)
- Primera app con Django -> [djangoproject.com](https://djangoproject.com)
- Solución problema interprete Python en vscode -> [you tube – Andrés Cruz](#)
- Como crearl URLs amigables en Django -> [cosasdedevs.com](https://cosasdedevs.com)