
PYTHON&DJANGO

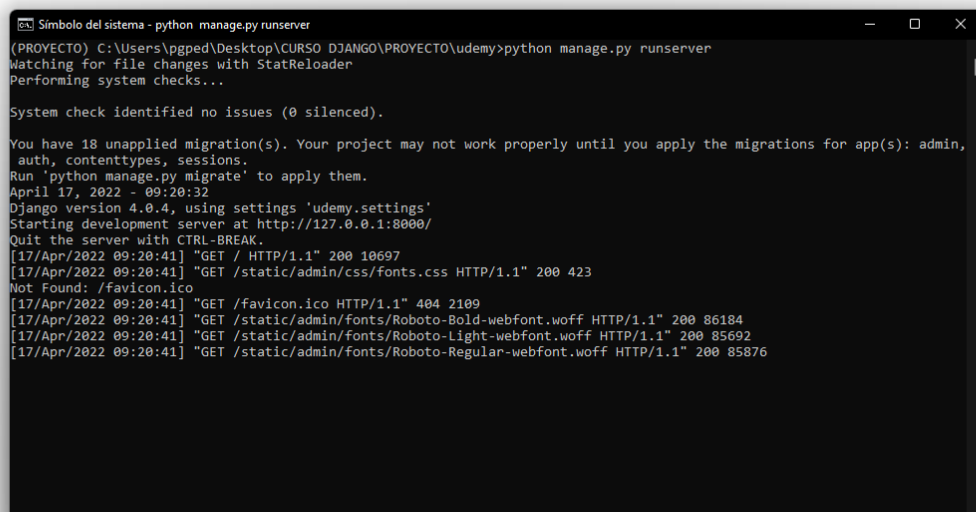


Para empezar, he creado un entorno virtual y he accedido a él, dentro de este he descargado la última versión de Django:

```
C:\Users\pgped\Desktop\CURSO DJANGO\PROYECTO>.\Scripts\activate

(PROYECTO) C:\Users\pgped\Desktop\CURSO DJANGO\PROYECTO>pip3 install django
Collecting django
  Downloading Django-4.0.4-py3-none-any.whl (8.0 MB)
----- 8.0/8.0 MB 30.3 MB/s eta 0:00:00
Collecting tzdata
  Using cached tzdata-2022.1-py2.py3-none-any.whl (339 kB)
Collecting asgiref<4,>=3.4.1
  Downloading asgiref-3.5.0-py3-none-any.whl (22 kB)
Collecting sqlparse>=0.2.2
  Downloading sqlparse-0.4.2-py3-none-any.whl (42 kB)
----- 42.3/42.3 KB ? eta 0:00:00
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.5.0 django-4.0.4 sqlparse-0.4.2 tzdata-2022.1
```

En este paso ya he creado mi proyecto haciendo uso de la siguiente instrucción:
<python manage.py runserver> [Si usaba python3 en vez de python daba un error “Error01”]:



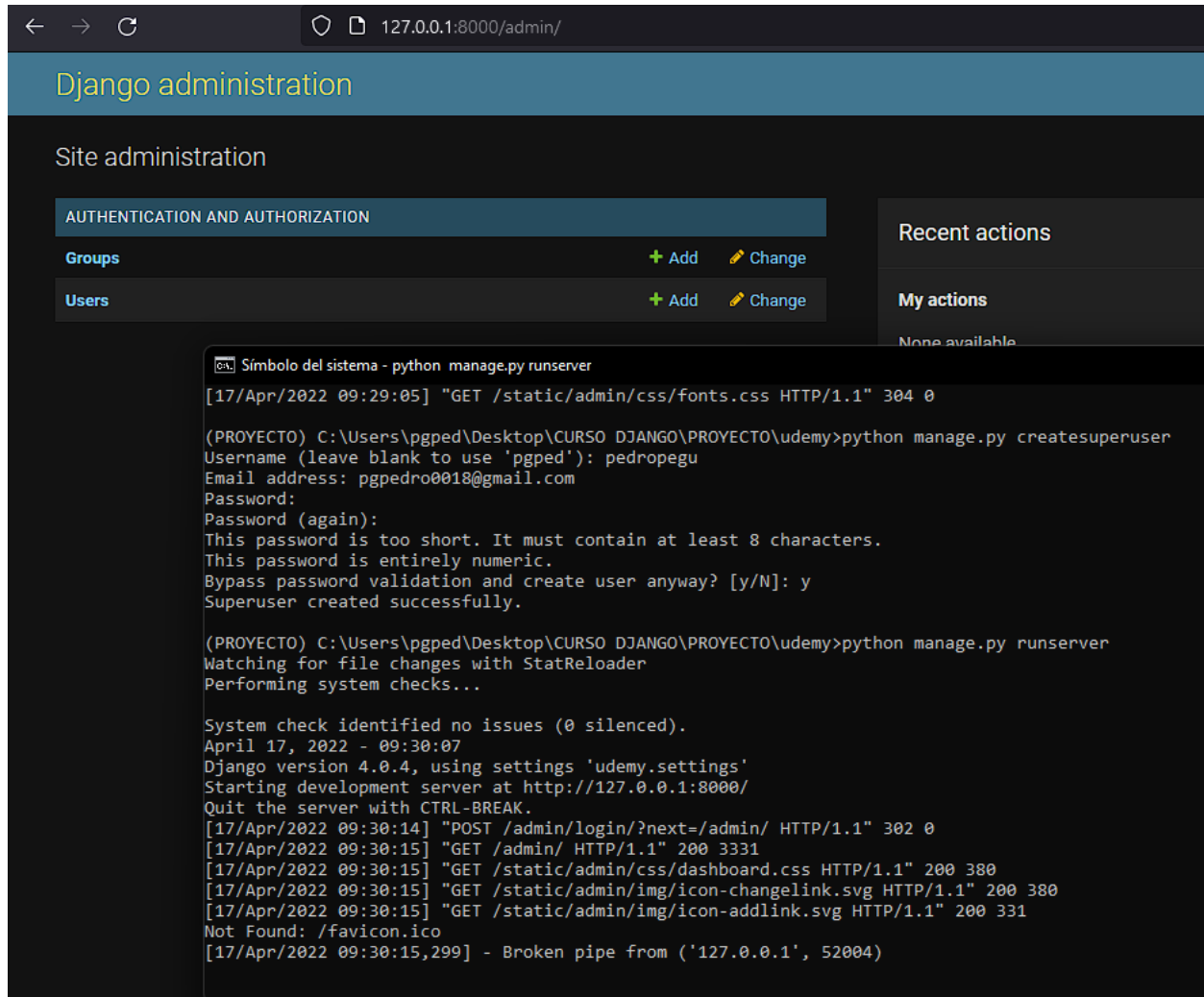
gratulations!

ue is in your
ny URLs.

Ahora cierro el servidor para hacer las migraciones <python manage.py migrate>:

```
(PROYECTO) C:\Users\pgped\Desktop\CURSO DJANGO\PROYECTO\udemy>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

Ahora vamos a acceder a la interfaz de administración de Django, para ello debemos de crear primero un usuario y su contraseña, para ello haremos uso de la siguiente instrucción `<python manage.py createsuperuser>` y cuando tengamos el usuario creado levantaremos el servidor y accederemos a “ <http://127.0.0.1:8000/admin/> “:



The image shows a screenshot of a web browser displaying the Django administration interface at `127.0.0.1:8000/admin/`. The page title is "Django administration". Under the "Site administration" section, there is a tab for "AUTHENTICATION AND AUTHORIZATION". Below this tab, there are two sections: "Groups" and "Users". Each section has a "+ Add" button and a "Change" button. To the right of these sections, there are two boxes: "Recent actions" and "My actions", both showing "None available".

Below the browser window, a terminal window is open, showing the command prompt and the output of the `python manage.py createsuperuser` command. The output shows that the user "pedropegu" was created successfully. The terminal also shows the command `python manage.py runserver` being executed, which starts the Django development server at `http://127.0.0.1:8000/`. The terminal output includes system checks, the Django version (4.0.4), and the server status.

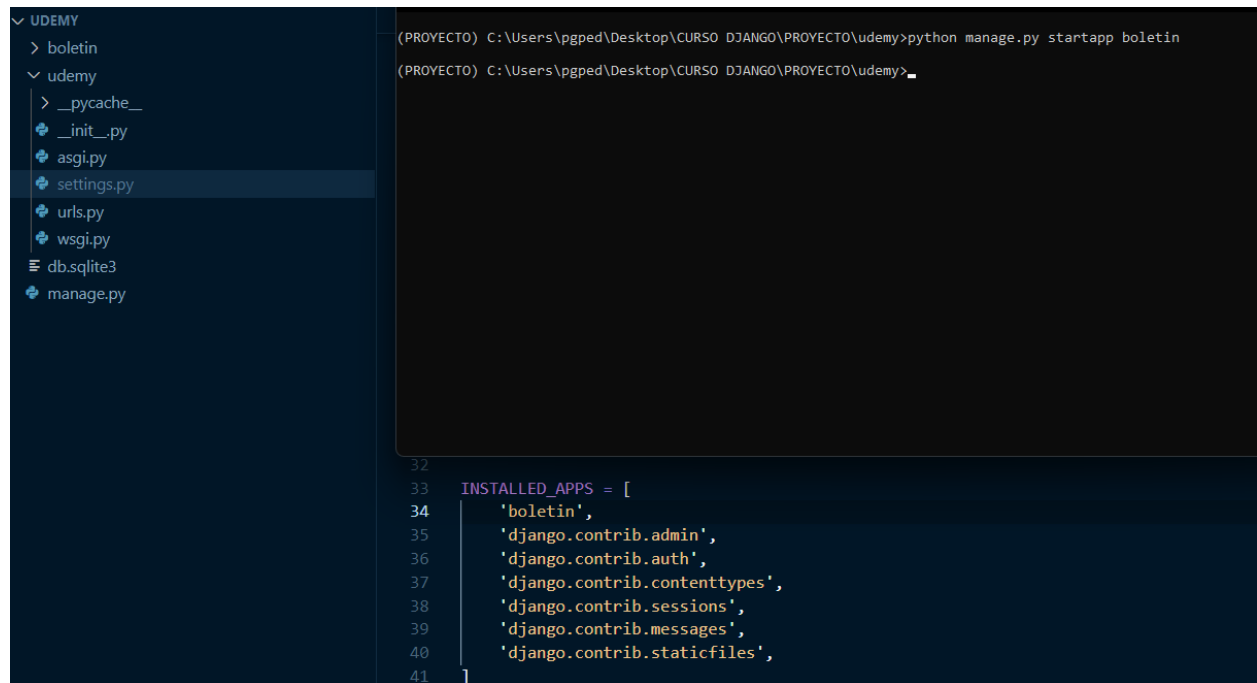
```
Símbolo del sistema - python manage.py runserver
[17/Apr/2022 09:29:05] "GET /static/admin/css/fonts.css HTTP/1.1" 304 0

(PROYECTO) C:\Users\pgped\Desktop\CURSO DJANGO\PROYECTO\udemy>python manage.py createsuperuser
Username (leave blank to use 'pgped'): pedropegu
Email address: pgpedro0018@gmail.com
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

(PROYECTO) C:\Users\pgped\Desktop\CURSO DJANGO\PROYECTO\udemy>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 17, 2022 - 09:30:07
Django version 4.0.4, using settings 'udemy.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[17/Apr/2022 09:30:14] "POST /admin/login/?next=/admin/ HTTP/1.1" 302 0
[17/Apr/2022 09:30:15] "GET /admin/ HTTP/1.1" 200 3331
[17/Apr/2022 09:30:15] "GET /static/admin/css/dashboard.css HTTP/1.1" 200 380
[17/Apr/2022 09:30:15] "GET /static/admin/img/icon-changelink.svg HTTP/1.1" 200 380
[17/Apr/2022 09:30:15] "GET /static/admin/img/icon-addlink.svg HTTP/1.1" 200 331
Not Found: /favicon.ico
[17/Apr/2022 09:30:15,299] - Broken pipe from ('127.0.0.1', 52004)
```

Creamos nuestra aplicación y la registramos, para registrarla tenemos que acceder a `settings.py` y añadir el nombre de nuestra aplicación a `INSTALLED_APPS`:



The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'UDEMY' with a subdirectory 'boletin' and a file 'settings.py' selected. The code editor shows the contents of 'settings.py' with the following code:

```
32
33 INSTALLED_APPS = [
34     'boletin',
35     'django.contrib.admin',
36     'django.contrib.auth',
37     'django.contrib.contenttypes',
38     'django.contrib.sessions',
39     'django.contrib.messages',
40     'django.contrib.staticfiles',
41 ]
```

A terminal window is open above the code editor, showing the following commands and output:

```
(PROYECTO) C:\Users\pgped\Desktop\CURSO DJANGO\PROYECTO\udemy>python manage.py startapp boletin
(PROYECTO) C:\Users\pgped\Desktop\CURSO DJANGO\PROYECTO\udemy>_
```

En el apartado models.py de nuestra aplicación vamos a crear nuestro primer modelo, también haremos las migraciones, para estas haremos uso de dos comandos <python manage.py makemigrations> y <python manage.py migrate>:

```
boletin > models.py > Registrado > __str__
3  from django.db import models
4
5  # Create your models here.
6  class Registrado(models.Model):
7      Nombre = models.CharField(max_length=100, blank=True, null=True)
8      email = models.EmailField()
9      timestamp = models.DateTimeField(auto_now_add=True, auto_now=False)
10
11
12      def __str__(self):
13          return self.email
```

Símbolo del sistema

```
(PROYECTO) C:\Users\pgped\Desktop\CURSO DJANGO\PROYECTO\udemy>python manage.py startapp boletin
(PROYECTO) C:\Users\pgped\Desktop\CURSO DJANGO\PROYECTO\udemy>python manage.py makemigrations
Migrations for 'boletin':
  boletin\migrations\0001_initial.py
    - Create model Registrado
(PROYECTO) C:\Users\pgped\Desktop\CURSO DJANGO\PROYECTO\udemy>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, boletin, contenttypes, sessions
Running migrations:
  Applying boletin.0001_initial... OK
```

Creamos objetos en Python Shell y registramos Model en Admin.

The screenshot shows the Django Admin interface. On the left, the sidebar has a search bar and a menu with 'AUTHENTICATION AND AUTHORIZATION' (Groups, Users) and 'BOLETIN' (Registrados). The main area is titled 'Select registrado to change' and shows a list of two objects: 'paco@gmail.com' and 'pepe@gmail.com'. Below the list, a terminal window is open, showing the command prompt where the 'Registrado' model is created and the server is run. The terminal output shows the Django version, settings, and the server starting at http://127.0.0.1:8000/. The server logs show several GET requests to the admin site, including the one that created the 'paco@gmail.com' object.

Para poder ver Registrados en la página de administración hemos tenido que agregar a admins.py (De nuestra aplicación) nuestro modelo y registrarlo.

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows the 'boletin' directory with files like __pycache__, migrations, __init__.py, admin.py, apps.py, models.py, tests.py, and views.py. The code editor shows the content of 'admin.py', which includes the following code:

```
1 from django.contrib import admin
2 from .models import Registrado
3 # Register your models here.
4
5 admin.site.register(Registrado)
```

Ahora vamos a personalizar este modelo en el Admin, para ello tenemos lo siguiente:

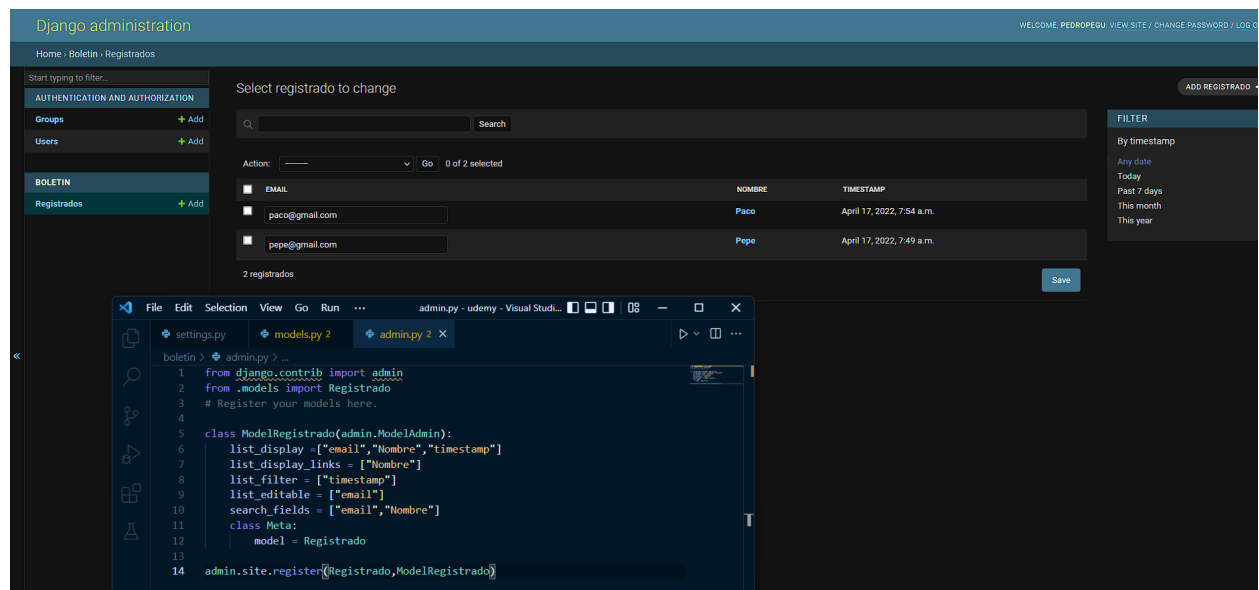
ModelAdmin.list_editable: Establezca list_editable en una lista de nombres de campo en el modelo que permitirá la edición en la página de la lista de cambios. Es decir, los campos enumerados en list_editable se mostrarán como widgets de formulario en la página de lista de cambios, lo que permitirá a los usuarios editar y guardar varias filas a la vez.

ModelAdmin.list_display: Configure list_display para controlar qué campos se muestran en la página de la lista de cambios del administrador.

ModelAdmin.list_filter: para activar los filtros en la barra lateral derecha de la página de la lista de cambios del administrador

ModelAdmin.search_fields:habilita un cuadro de búsqueda en la página de la lista de cambios del administrador. Esto debe establecerse en una lista de nombres de campo que se buscarán cada vez que alguien envíe una consulta de búsqueda en ese cuadro de texto.

ModelAdmin.list_display_links: para controlar si y qué campos en list_display deben estar vinculados a la página de "cambio" para un objeto.



The screenshot displays the Django Admin interface for a model named 'Registrados'. The interface includes a sidebar with navigation links, a search bar, and a table of records. A code editor window is overlaid on the bottom left, showing the ModelAdmin configuration for 'Registrado'.

ModelAdmin Configuration (from code editor):

```
1 from django.contrib import admin
2 from .models import Registrado
3 # Register your models here.
4
5 class ModelRegistrado(admin.ModelAdmin):
6     list_display = ("email", "Nombre", "timestamp")
7     list_display_links = ("Nombre",)
8     list_filter = ("timestamp",)
9     list_editable = ("email",)
10    search_fields = ("email", "Nombre")
11    class Meta:
12        model = Registrado
13
14 admin.site.register(Registrado, ModelRegistrado)
```

Registrados Table:

| EMAIL | NOMBRE | TIMESTAMP |
|----------------|--------|---------------------------|
| paco@gmail.com | Paco | April 17, 2022, 7:54 a.m. |
| pepe@gmail.com | Pepe | April 17, 2022, 7:49 a.m. |

En el siguiente paso vamos a configurar las plantillas para ello nos dirigimos al fichero views.py de nuestra aplicación:

```
boletin > views.py > inicio
1  from django.shortcuts import render
2
3  # Create your views here.
4  def inicio(request):
5      return render(request, "boletin/index.html", {})
```

He añadido una simple página de inicio, ahora nos vamos a dirigir al fichero urls.py del proyecto y vamos darle una url a esta vista:

```
from django.contrib import admin
from django.urls import path
from boletin import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.inicio, name='inicio'),
]
```

He dejado la ruta en blanco ya que así, mi página de inicio será a la que se accede principalmente.

Para hacer que me muestre este index.html he creado (En la carpeta de la aplicación) una carpeta templates que dentro contiene otra que tiene el nombre de la aplicación y dentro de esta última el index.html

```
templates\boletin
index.html
```

Vamos a crear un formulario y guardar los datos de este en un modelo, para ello en la carpeta de nuestra aplicación creamos un fichero forms.py el cual va a tener los campos de nuestro formulario.

```
boletin > forms.py > RegForm
1  from django import forms
2
3  class RegForm(forms.Form):
4      nombre = forms.CharField(max_length=100)
5      email = forms.EmailField()
```

Para seguir en el index.html vamos a agregar este formulario:

```
1  <h1>HOLA MUNDO</h1>
2
3  <form method="post" action="">
4      {% csrf_token %}
5      {{ form }}
6      <input type="submit" value="Registro"/>
7  </form>
```

Ahora en el fichero views.py vamos a validar este formulario y enviarlo a nuestro modelo Registrado que creamos varios pasos atrás. [Varias puntos explicados dentro del fichero]

```
3  from django.shortcuts import render
4  from .forms import RegForm #Agregamos nuestro formulario
5  from .models import Registrado #Agregamos nuestro modelo
6
7  # Create your views here.
8  #ver todo lo que podemos hacer con un formulario print(dir(form))
9  def inicio(request):
10
11      form = RegForm(request.POST or None)
12
13      if request.method == 'POST' and form.is_valid(): #Verificamos si es metodo POST para que no tengamos errores y validamos el formulario
14          correo=request.POST["email"] #Extraemos el correo del formulario enviado
15          usuario=request.POST["nombre"] #Extraemos el nombre del formulario enviado
16
17          obj = Registrado.objects.create(email=correo,Nombre=usuario) #Añadimos nuestro objeto
18      context = {
19          "form":form,
20      }
21      return render(request,"boletin/index.html", context)
```

[ERROR: form.cleaned_data no me funciona en ningun lado por eso el request.POST["campo"]]

Ahora vamos a crear un formulario para Contacto, este va a tener un campo que va a ser Mensaje el cual será un text area:

```
4  class ContactForm(forms.Form):
5      nombre = forms.CharField()
6      email = forms.EmailField()
7      mensaje = forms.CharField(widget=forms.Textarea)
```

En views.py debemos crear la vista correspondiente a contacto:

```
def contact(request):
    form = ContactForm(request.POST or None)
    if request.method == 'POST' and form.is_valid:
        email = request.POST["email"]
        mensaje = request.POST["mensaje"]
        nombre = request.POST["nombre"]
        print(email,nombre,mensaje)
    context = {
        "form":form
    }
    return render(request, "boletin/contact.html", context)
```

También lo debemos agregar una url a la vista, para ello en urls.py:

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.inicio, name='inicio'),
    path('contacto/', views.contact, name='contacto'),
]
```

En el fichero html tendremos lo siguiente:

```
1 <form method="post" action="">
2     {% csrf_token %}
3     {{ form.as_p }}
4     <input type="submit" value="Registro"/>
5 </form>
```

Voy a seguir configurando Email, en mi caso voy a usar gmail por tanto en settings.py realizo la siguiente configuración:

```
ALLOWED_HOSTS = []

EMAIL_HOST = 'smtp.gmail.com'
EMAIL_HOST_USER = ''
EMAIL_HOST_PASSWORD =
EMAIL_PORT = 587
EMAIL_USE_TLS = True
```

Ahora en el archivo views.py importamos los siguientes modulos:

```
from django.core.mail import send_mail
from django.conf import settings
```

Una vez hechos estos dos pasos y estando en views.py modificamos la vista de contacto de la siguiente manera (Con esto podemos enviar un correo con los datos):

```
def contact(request):
    form = ContactForm(request.POST or None)
    if request.method == 'POST' and form.is_valid():
        email = request.POST["email"]
        mensaje = request.POST["mensaje"]
        nombre = request.POST["nombre"]
        asunto = 'Form Contacto'
        email_mensaje = 'Hola que tal'
        email_from = settings.EMAIL_HOST_USER
        email_to = [email_from, "otro_email@gmail.com"]

        send_mail(asunto, email_mensaje, email_from, email_to, fail_silently=True) #El ultimo campo es para que no de error ya que los correos no existen

    context = {
        "form": form
    }
    return render(request, "boletin/contact.html", context)
```

[ERROR:Poner una ñ o una tilde hace que no funcione nada]

Vamos a seguir configurando los archivos estáticos:

En settings.py añadimos las siguientes líneas:

```
STATIC_URL = '/static/'
STATIC_URL = '/media/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static_pro", "static"),
]

STATIC_ROOT = os.path.join(os.path.dirname(BASE_DIR), "static_env", "static_root")
MEDIA_ROOT = os.path.join(os.path.dirname(BASE_DIR), "static_env", "media_root")
```

Y en urls.py (Si esta el modo desarrollo activado), añadimos:

```
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
    urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

Ahora crearemos esas carpetas y la static_pro dentro de udemy:

| | | | |
|------------|------------------|-----------------------|------|
| Include | 17/04/2022 9:12 | Carpeta de archivos | |
| Lib | 17/04/2022 21:58 | Carpeta de archivos | |
| Scripts | 17/04/2022 22:02 | Carpeta de archivos | |
| static_env | 18/04/2022 10:22 | Carpeta de archivos | |
| udemy | 18/04/2022 10:22 | Carpeta de archivos | |
| .gitignore | 17/04/2022 9:12 | Archivo de origen ... | 1 KB |
| pyenv | 17/04/2022 9:12 | Archivo de origen ... | 1 KB |

Por último ejecutamos el siguiente comando:

```
C:\Users\pedropegu\Desktop\CURSO-DJANGO\PROYECTO\udemy>python manage.py collectstatic
128 static files copied to 'C:\Users\pedropegu\Desktop\CURSO-DJANGO\PROYECTO\static_env\static_root'.
```

Ahora vamos a realizar el siguiente diseño usando Bootstrap y herencias:

PROBAR DJANGO

Home Link Disabled

Search



DJANGO

Bienvenido pedropegu

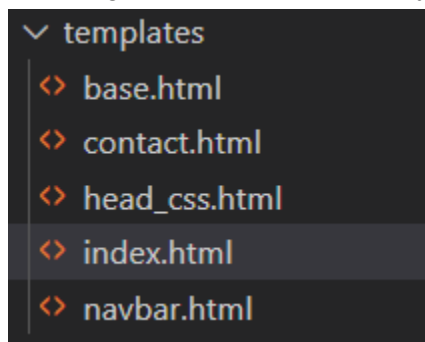
Nombre:

Email:

En static_pro añadiremos el css bajado de la pagina de Bootstrap:

| | | | |
|---|-----------------|---------------------|--------|
|  bootstrap.min | 18/04/2022 9:42 | Documento de hoj... | 161 KB |
|  navbar-top | 18/04/2022 9:42 | Documento de hoj... | 1 KB |

Para seguir vamos a templates y agregamos los archivos html correspondientes:



En el fichero base.html cargamos los ficheros css y ahora creamos bloques para especificar dónde se va a sustituir y añadimos las etiquetas include para el estilo de la cabecera:

```
{% load static %}
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors">
    <meta name="generator" content="Hugo 0.88.1">
    <title>{% block head_title %}{% endblock %}</title>

    {% include "head_css.html" %}
    <link rel="canonical" href="https://getbootstrap.com/docs/5.1/examples/navbar-static/">

  </head>
  <body>

    {% include "navbar.html" %}

    {% block jumbotron %}
      <div class="jumbotron">
        {% block jumbotron_content %}

      </div>
    {% endblock %}

    {% block content %}
    {% endblock %}

    <script src="/docs/5.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYs0g+OMhuP+IlR9"
  </body>
</html>
```


El fichero index quedará algo tal que así:

```
{% extends "base.html" %}
{% block head_title %}Inicio{% block.super %}{% endblock %}
{% block jumbotron_content %}
    <h1>DJANGO</h1>
{% endblock %}

{% block content %}
    {{ titulo }}
    <br><br>
    <form method="POST" action=""> {% csrf_token %}
    {{ form.as_p }}
    <input type="submit" value="Regístrate" />
    </form>

{% endblock %}
```

Lo que hacemos es extender el diseño base para poder usar su HTML y su css.

Vamos a cambiar el aspecto del formulario para ello instalamos crispy forms:

```
<pip install django-crispy-forms>
```

También lo añadimos a nuestras aplicaciones y añadimos también

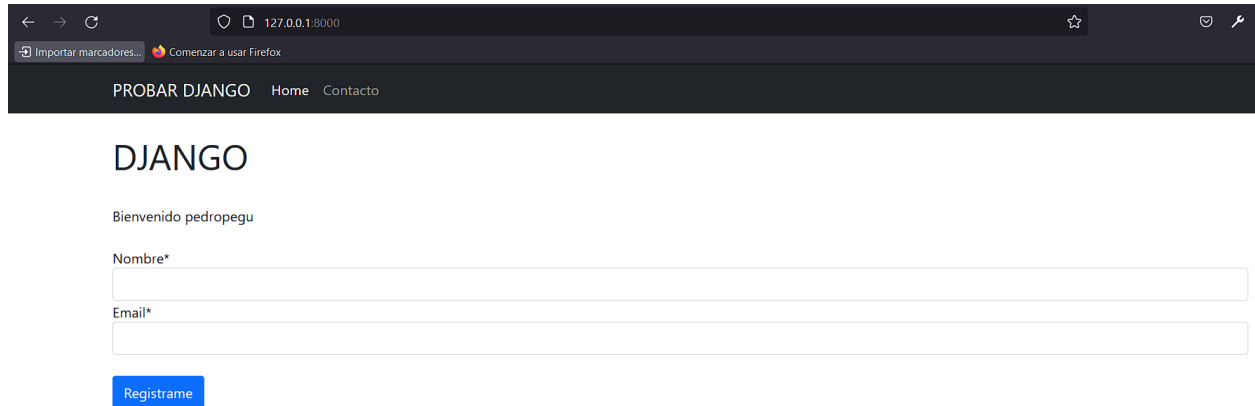
```
[CRISPY_TEMPLATE_PACK = bootstrap3]
```

```
INSTALLED_APPS = [  
    'crispy_forms',  
    'boletin',  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
]  
  
MIDDLEWARE = [  
    'django.middleware.security.SecurityMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'django.middleware.common.CommonMiddleware',  
    'django.middleware.csrf.CsrfViewMiddleware',  
    'django.contrib.auth.middleware.AuthenticationMiddleware',  
    'django.contrib.messages.middleware.MessageMiddleware',  
    'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]  
  
ROOT_URLCONF = 'udemy.urls'  
CRISPY_TEMPLATE_PACK = 'bootstrap3'
```

Voy a configurar las urls de los botones de la página, para no escribir la url a fuego simplemente usamos el nombre que le dimos en urls.py:

```
<a class="nav-link active" aria-current="page" href="{% url 'inicio' %}">Home</a>
</li>
<li class="nav-item">
  <a class="nav-link" href="{% url 'contacto' %}">Contacto</a>
</li>
```

Actualmente mi página tendría el siguiente aspecto:



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000'. The page has a dark navigation bar with the text 'PROBAR DJANGO' and two links: 'Home' and 'Contacto'. Below the navigation bar, the word 'DJANGO' is displayed in a large, bold, black font. Underneath, the text 'Bienvenido pedropegu' is shown. There are two input fields: 'Nombre*' and 'Email*', both with asterisks indicating they are required. Below these fields is a blue button labeled 'Regístrate'.

Vamos a pasar a Django Registration Redux, por tanto para instalarlo:

```
(PROYECTO) C:\Users\pedropegu\Desktop\CURSO-DJANGO\PROYECTO\udemy>pip install django-registration-redux
Collecting django-registration-redux
  Downloading django_registration_redux-2.10-py2.py3-none-any.whl (213 kB)
    | 213 kB 2.2 MB/s
Installing collected packages: django-registration-redux
Successfully installed django-registration-redux-2.10
WARNING: You are using pip version 20.1.1; however, version 22.0.4 is available.
You should consider upgrading via the 'C:\Users\pedropegu\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.P
ython.3.7_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip' command.
```

Añadimos la aplicación a nuestra lista de aplicaciones:

```
INSTALLED_APPS = [
    'django.contrib.sites',
    'registration',
    'crispy_forms',
    'boletin',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

Configuramos estas dos variables:

```
ACCOUNT_ACTIVATION_DAYS = 7
REGISTRATION_AUTO_LOGIN = True
```

Ahora realizamos las migraciones:

<python manage.py makemigrations> & <python manage.py migrate> & <python manage.py
runserver>

Ahora ya podremos acceder a localhost:8000/accounts/register

PROBAR DJANGO [Home](#) [Contacto](#)

Registrarte Gratis!

Nombre de usuario*

Requerido. 150 caracteres como máximo. Únicamente letras, dígitos y @/./+/-/_

Correo Electrónico*

Contraseña*

- Su contraseña no puede asemejarse tanto a su otra información personal.
- Su contraseña debe contener al menos 8 caracteres.
- Su contraseña no puede ser una clave utilizada comúnmente.
- Su contraseña no puede ser completamente numérica.

Contraseña (confirmación)*

Para verificar, introduzca la misma contraseña anterior.

[Registrarme](#)

Ya tienes cuenta? [Iniciar Sesión.](#)

También he añadido al menú botones para el login, register y logout. [El botón logout solo sale si estas logueado]:

```
<ul class="nav navbar-nav navbar-right" style="color: ■ white; text-decoration: None;">  
    {% if request.user.is_authenticated %}  
        <li><a href="{% url 'auth_logout' %}" style="text-decoration: None; color: ■ white;">Salir</li></a>  
    {% else %}  
        <li><a href="{% url 'registration_register' %}" style="text-decoration: None; color: ■ white;">Registrarse</li></a>  
        &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~  
        <li><a href="{% url 'auth_login' %}" style="text-decoration: None; color: ■ white;">Login</li></a>  
    {% endif %}
```

PROBAR DJANGO

[Home](#) [Contacto](#)

[Registrarse](#) [Login](#)

Bienvenido

Nombre*

Email*

Registrarme

Ahora vamos a hacer que puedas loguear desde la barra de navegación, para ello creamos un formulario en la barra el cual tendrá lo siguiente:

```
{% if not request.user.is_authenticated and not '/'|accounts/login/' in request.get_full_path %}
<form class="navbar-form navbar" method="POST" action="{% url 'auth_login' %}">% csrf_token %}
    <br>
    <div class="form-group">
        |   <input type="text" class="form-control" name="username" placeholder="Usuario"/>
    </div>
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
    <div class="form-group">
        |   <input type="text" class="form-control" name="password" placeholder="Contraseña"/>
    </div>
    ~~~~~~
    &nbsp;&nbsp;&nbsp;&nbsp;&~
    |   <button type="submit">Entrar</button>
    </form>
{% endif %}
```

El if que tenemos sirve para que si el usuario está logueado o está la ruta localhost:8000/accounts/login no le aparezca este formulario.

PROBAR DJANGO

[Home](#) [Contacto](#)

Regístrate

Entrar

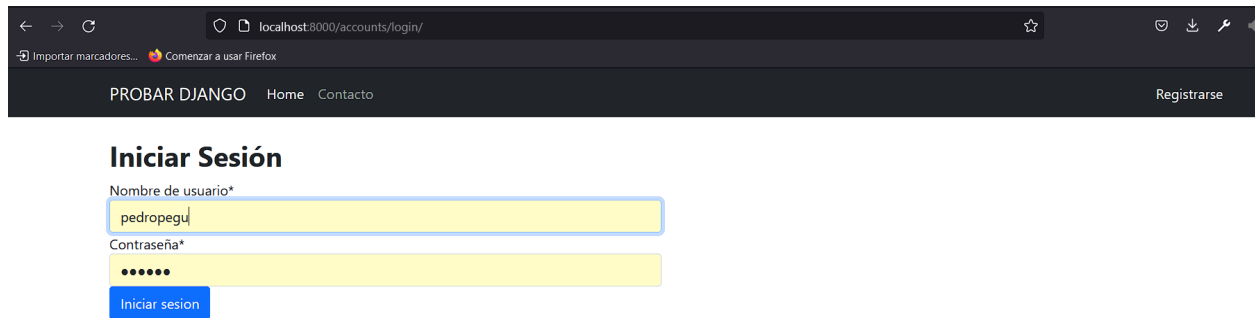
Bienvenido

Nombre*

Email*

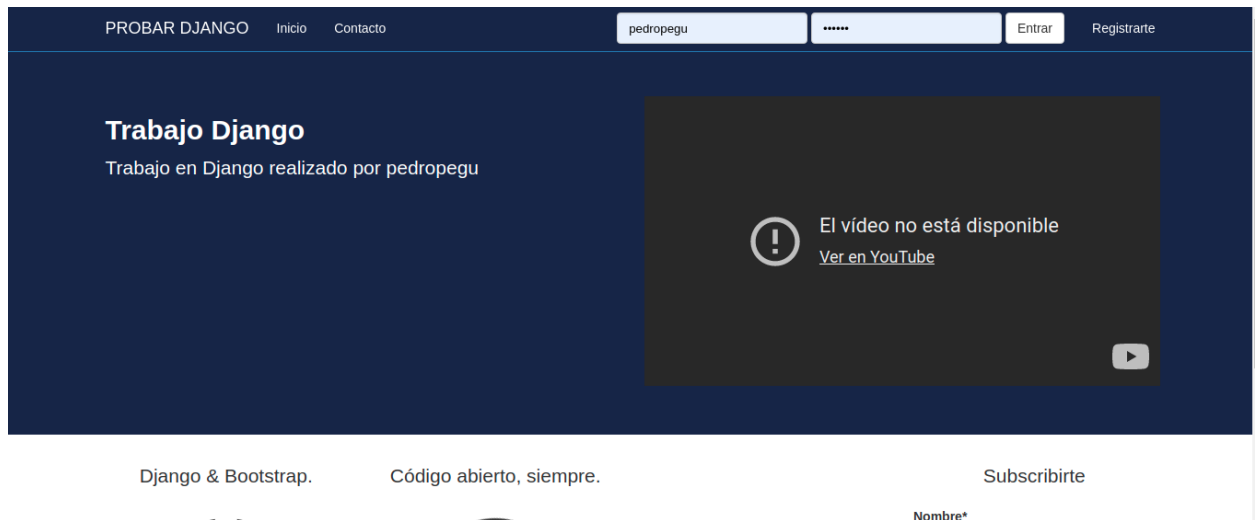
Regístrate

Si estuviésemos en la página de accounts/login:



A screenshot of a web browser showing the Django login page. The address bar indicates the URL is localhost:8000/accounts/login/. The page has a dark header with links for 'PROBAR DJANGO', 'Home', 'Contacto', and a 'Registrarse' button. The main content area is titled 'Iniciar Sesión' and contains two input fields: 'Nombre de usuario*' with the text 'pedropegu' and 'Contraseña*' with masked characters. A blue 'Iniciar sesion' button is at the bottom.

Para poder seguir bien con el tutorial he seguido el diseño del tutorial y queda de la siguiente manera:



A screenshot of the Django homepage. The header is dark blue with links for 'PROBAR DJANGO', 'Inicio', 'Contacto', and buttons for 'Entrar' and 'Registrarte'. The main content area has a dark blue background. On the left, it says 'Trabajo Django' and 'Trabajo en Django realizado por pedropegu'. On the right, there is a video player with a message 'El vídeo no está disponible' and a link 'Ver en YouTube'. The footer is white and contains the text 'Django & Bootstrap.', 'Código abierto, siempre.', 'Suscribirse', and a 'Nombre*' label.

Si el usuario está registrado, no le saldrá el apartado de “Subscribirte” y le saldrá un mensaje de bienvenida.

Bienvenido pedropegu



Ahora vamos a trabajar con querysets [Un QuerySet te permite leer los datos de la base de datos, filtrarlos y ordenarlos.] por tanto en el archivo views.py modificamos para pasar Queryset al context:

```
if request.user.is_authenticated and request.user.is_staff:
    queryset = Registrado.objects.all()
    context = {
        "queryset": queryset,
    }
```

En mi caso solo si el usuario está autenticado y es parte del staff podra verlo, ahora en el html para mostrar cada objeto lo hacemos de la siguiente manera:

```
{% else %}
<br>
<table class="table">
{% for instance in queryset %}
<tr><td>{{ instance }}</td><td>{{ instance.email }}</td><td>{{ instance.timestamp }}</td>
</tr>

{% endfor %}
</table>
```

El else del principio corresponde a este if:

```
{% if not request.user.is_staff %}
```

Esto está hecho para lo dicho anteriormente, que solo pueda verlo los Staff.

Ahora si un staff de loguea podrá ver lo siguiente:

| PROBAR DJANGO | | | Inicio | Contacto | Salir |
|---------------|------------------------|---------------------------------|--------|----------|-------|
| Pepe | pepe@gmail.com | 17 de Abril de 2022 a las 07:49 | | | |
| Paco | paco@gmail.com | 17 de Abril de 2022 a las 07:54 | | | |
| Pedro | pedroejemplo@gmail.com | 17 de Abril de 2022 a las 10:26 | | | |
| perez | perez@gmail.com | 17 de Abril de 2022 a las 19:58 | | | |
| pedorpera | ataf@gmail.com | 18 de Abril de 2022 a las 09:46 | | | |

ERROR01

```
(PROYECTO) C:\Users\pgped\Desktop\CURSO DJANGO\PROYECTO\udemy>python3 manage.py runserver
Traceback (most recent call last):
  File "C:\Users\pgped\Desktop\CURSO DJANGO\PROYECTO\udemy\manage.py", line 11, in main
    from django.core.management import execute_from_command_line
ModuleNotFoundError: No module named 'django'

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "C:\Users\pgped\Desktop\CURSO DJANGO\PROYECTO\udemy\manage.py", line 22, in <module>
    main()
  File "C:\Users\pgped\Desktop\CURSO DJANGO\PROYECTO\udemy\manage.py", line 13, in main
    raise ImportError(
ImportError: Couldn't import Django. Are you sure it's installed and available on your PYTHONPATH environment variable?
Did you forget to activate a virtual environment?
```

Este error me lo causó en windows debido a que usé `<python3 manage.py runserver>` en vez de `<python manage.py runserver>`