



Basado en Curso Django de Píldoras Informáticas:

<https://www.youtube.com/watch?v=7XO1AzwkPPE&list=PLU8oAIHdN5BmfwxFO7HdPciOCmmYneAB&index=2>



[www.sena.edu.co](http://www.sena.edu.co)



**Píldoras  
Informáticas**

Cursos de informática

# Panel de Administración



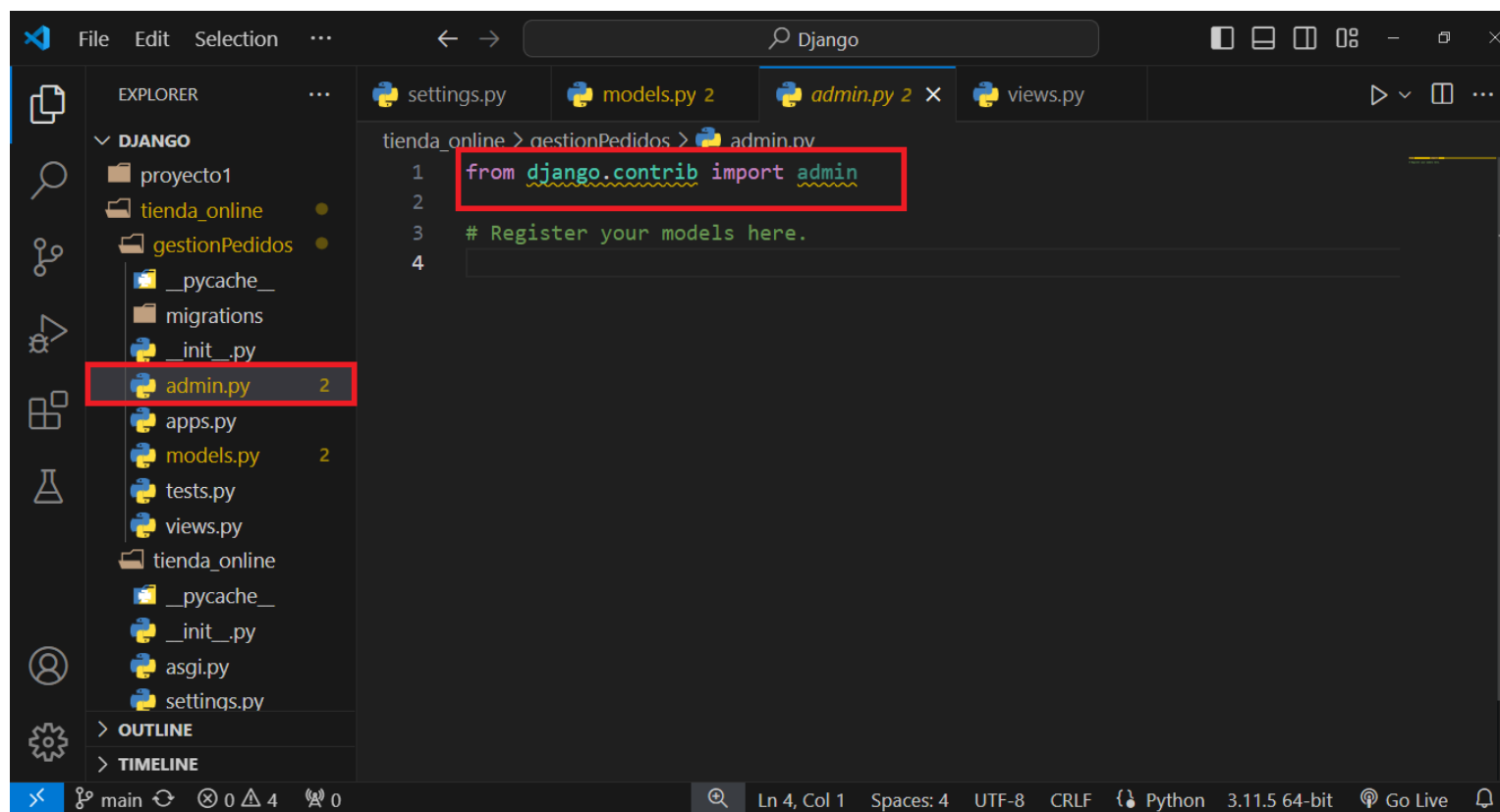
# Panel de Administración

El panel de administración en Django es una interfaz web preconstruida que permite a los desarrolladores gestionar y administrar los datos de su aplicación de manera sencilla. Esta herramienta es muy poderosa y flexible, y ofrece una serie de características que facilitan la administración del contenido del sitio web. Características:

- ✓ Configuración Rápida: El panel de administración se puede configurar rápidamente mediante la definición de modelos en el código de la aplicación. Una vez definidos los modelos, el panel de administración se genera automáticamente.
- ✓ Gestión de Modelos: Permite a los administradores agregar, editar y eliminar instancias de los modelos definidos en la aplicación. Esto incluye la gestión de relaciones entre modelos.
- ✓ Personalización: Es altamente personalizable. Puedes definir qué campos se muestran, aplicar filtros, ordenar datos, y crear formularios personalizados para la entrada de datos.
- ✓ Seguridad: Incluye autenticación y permisos de usuarios, lo que permite definir quién tiene acceso al panel y qué acciones pueden realizar.

# Panel de Administración en Nuestro Proyecto

En nuestro proyecto tenemos un archivo que se llama admin, el cual sirve para manejar modificar configurar lo relativo al panel de administración



```

File Edit Selection ...
Django

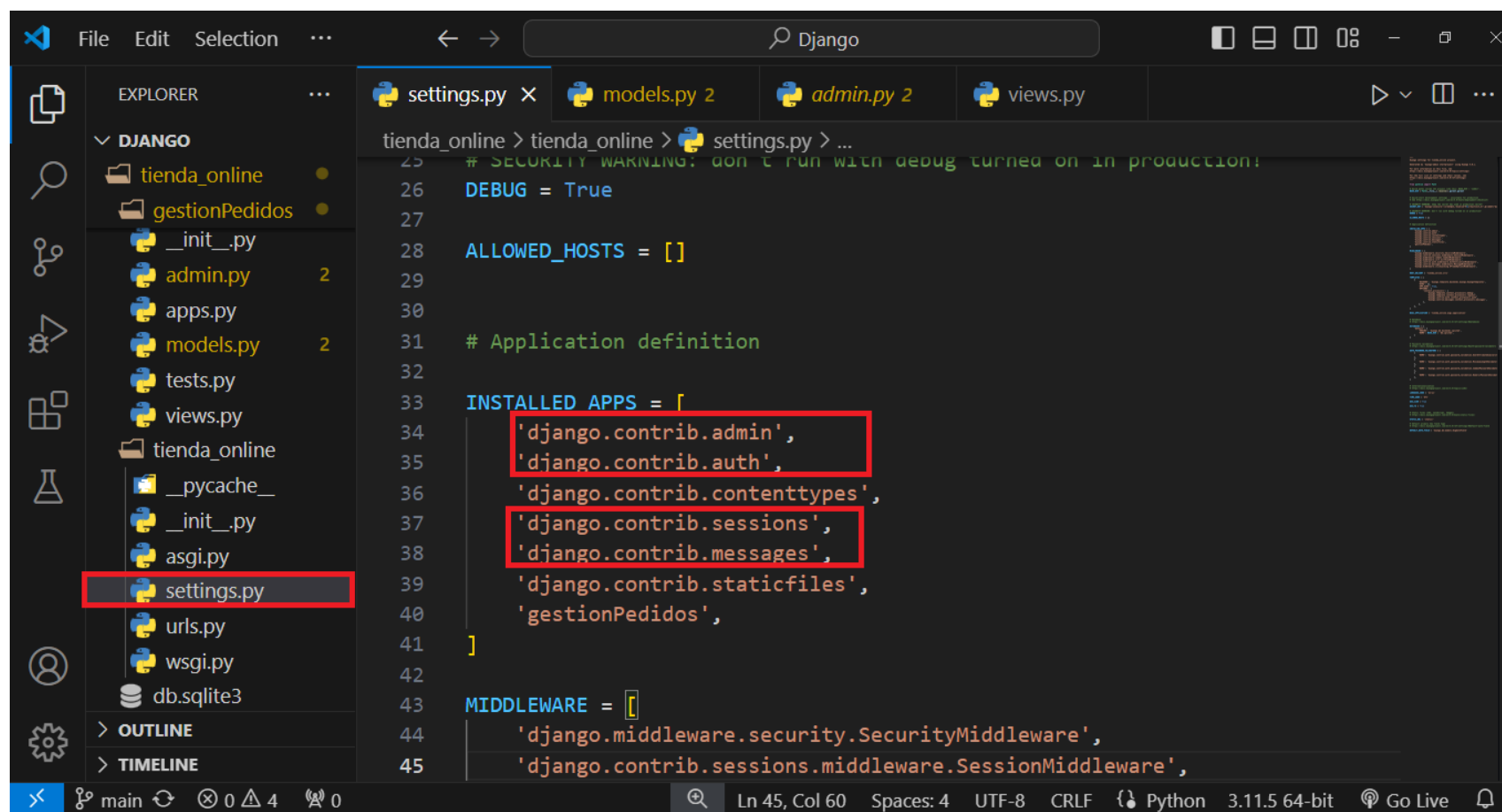
EXPLORER
DJANGO
  proyecto1
  tienda_online
  gestionPedidos
    __pycache__
    migrations
    init_.py
    admin.py 2
    apps.py
    models.py 2
    tests.py
    views.py
  tienda_online
    __pycache__
    init_.py
    asgi.py
    settings.py
  > OUTLINE
  > TIMELINE

tienda_online > gestionPedidos > admin.py
1 from django.contrib import admin
2
3 # Register your models here.
4

Ln 4, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.5 64-bit Go Live
  
```

# Panel de Administración en Nuestro Proyecto

En el archivo settings por defecto donde se instalan las apps tenemos librerías que permiten trabajar con todo lo relativo al panel de administración para poder identificar usuarios para poder enviar mensajes para poder manejar sesiones

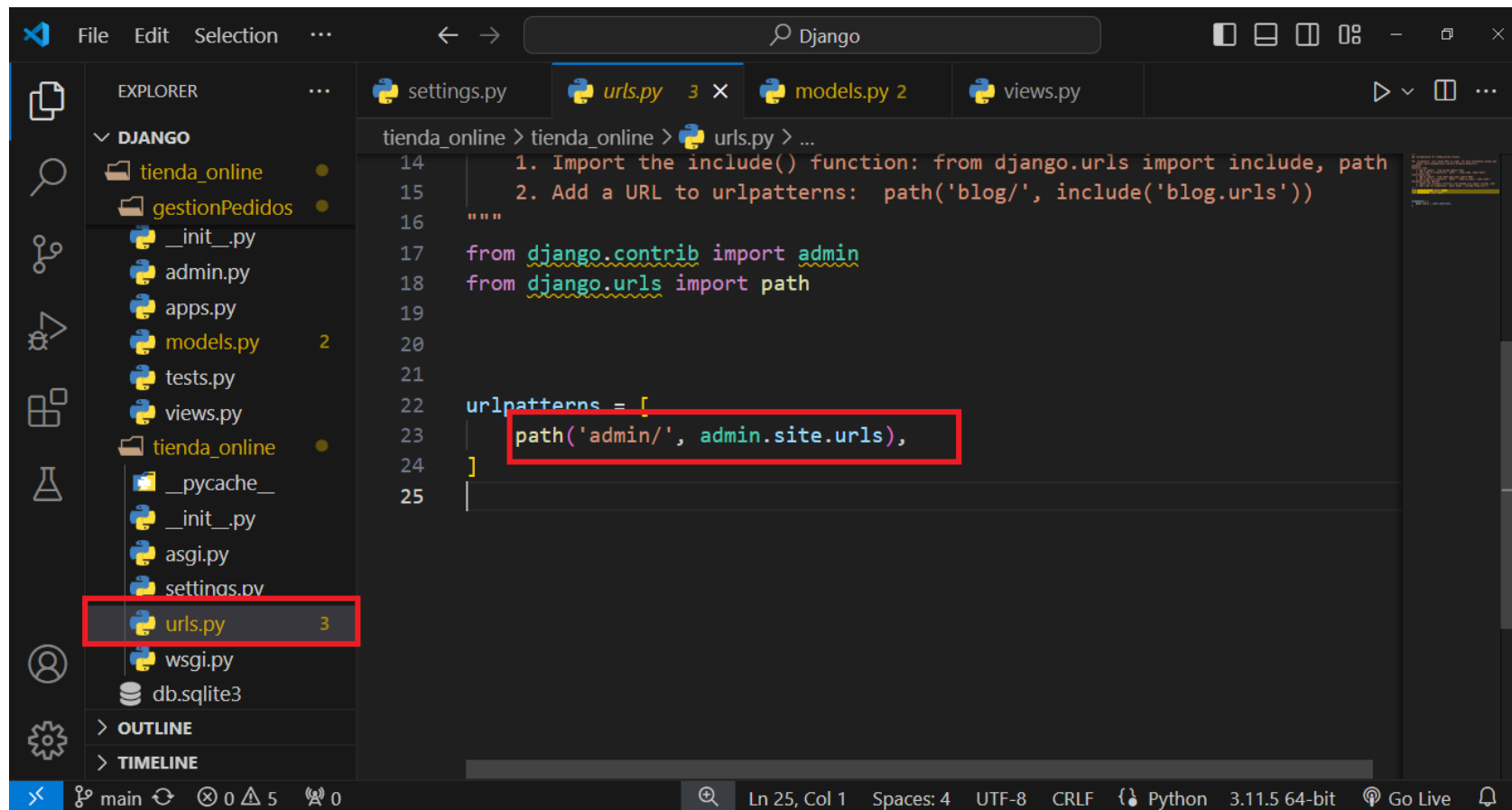


```

tienda_online > tienda_online > settings.py > ...
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30 # Application definition
31
32 INSTALLED_APPS = [
33     'django.contrib.admin',
34     'django.contrib.auth',
35     'django.contrib.contenttypes',
36     'django.contrib.sessions',
37     'django.contrib.messages',
38     'django.contrib.staticfiles',
39     'gestionPedidos',
40 ]
41
42 MIDDLEWARE = [
43     'django.middleware.security.SecurityMiddleware',
44     'django.contrib.sessions.middleware.SessionMiddleware',
45 ]
  
```

# Panel de Administración en Nuestro Proyecto

Si nos vamos al archivo urls en el que agregamos direcciones url dentro de url patterns ya hay una por defecto que hace referencia al panel de administración y se llama 'admin'



The screenshot shows the Visual Studio Code interface with a Django project. The Explorer panel on the left shows the project structure, with the `urls.py` file highlighted. The main editor window displays the content of `urls.py`, which includes imports for `include`, `path`, `admin`, and `path`. The `urlpatterns` list contains a default entry for the admin panel, which is highlighted with a red box.

```

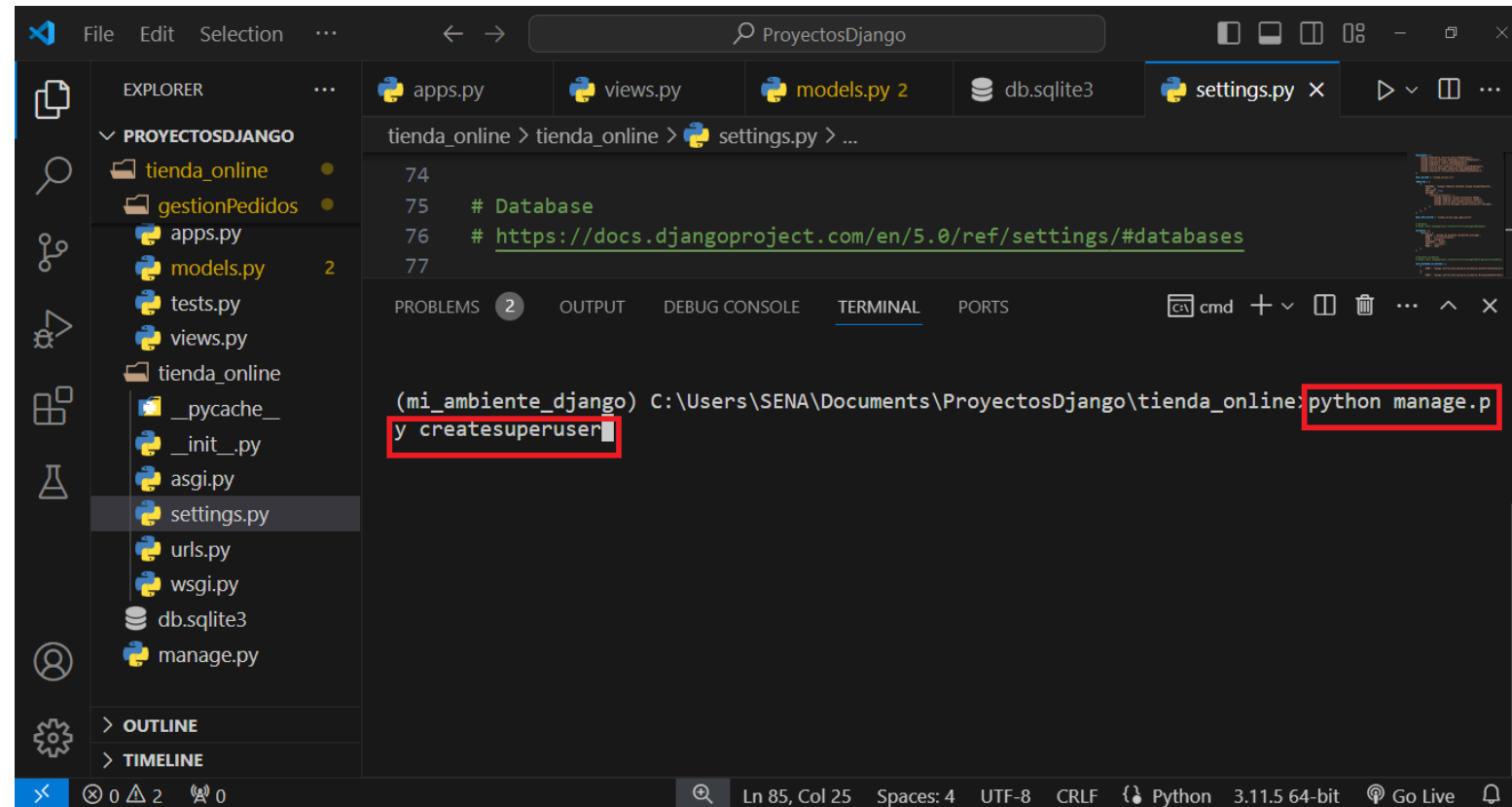
14 1. Import the include() function: from django.urls import include, path
15 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 from django.contrib import admin
18 from django.urls import path
19
20
21
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24 ]
25

```

# Acceso al Panel de Administración

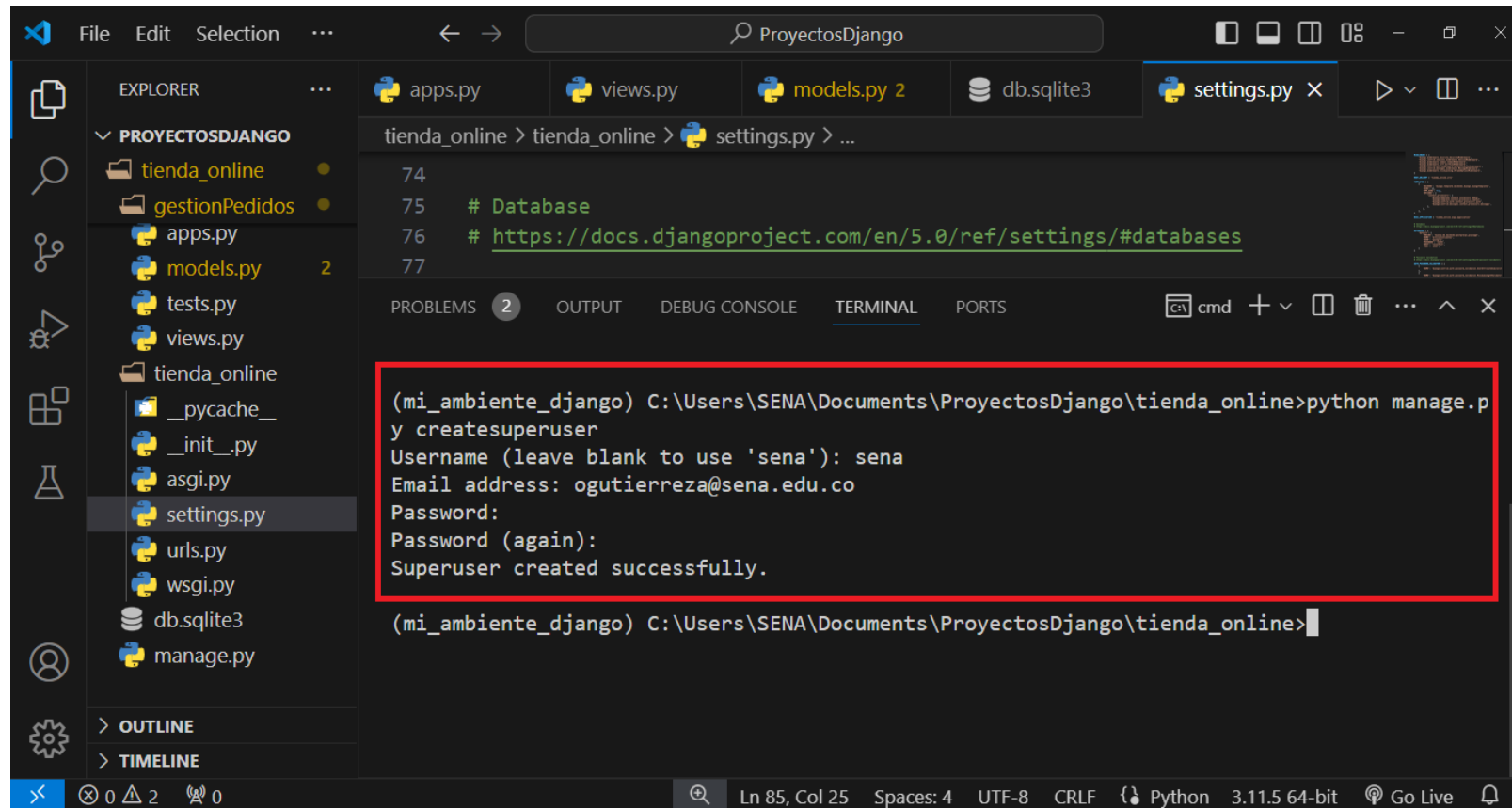
Para poder acceder al panel de administración es necesario crear un superusuario en nuestro proyecto para ello:

1. Ejecutar el comando: `python manage.py createsuperuser`



The screenshot shows the Visual Studio Code interface with a Django project named 'ProyectosDjango'. The Explorer panel on the left shows the project structure, including the 'tienda\_online' directory. The main editor area displays the 'settings.py' file, which contains the database configuration. The terminal panel at the bottom shows the command `(mi_ambiente_django) C:\Users\SENA\Documents\ProyectosDjango\tienda_online:python manage.py createsuperuser` being entered, with the command text highlighted by a red box.

Completar la información solicitada. Nota usaremos como  
usuario: sena  
password: sena12345

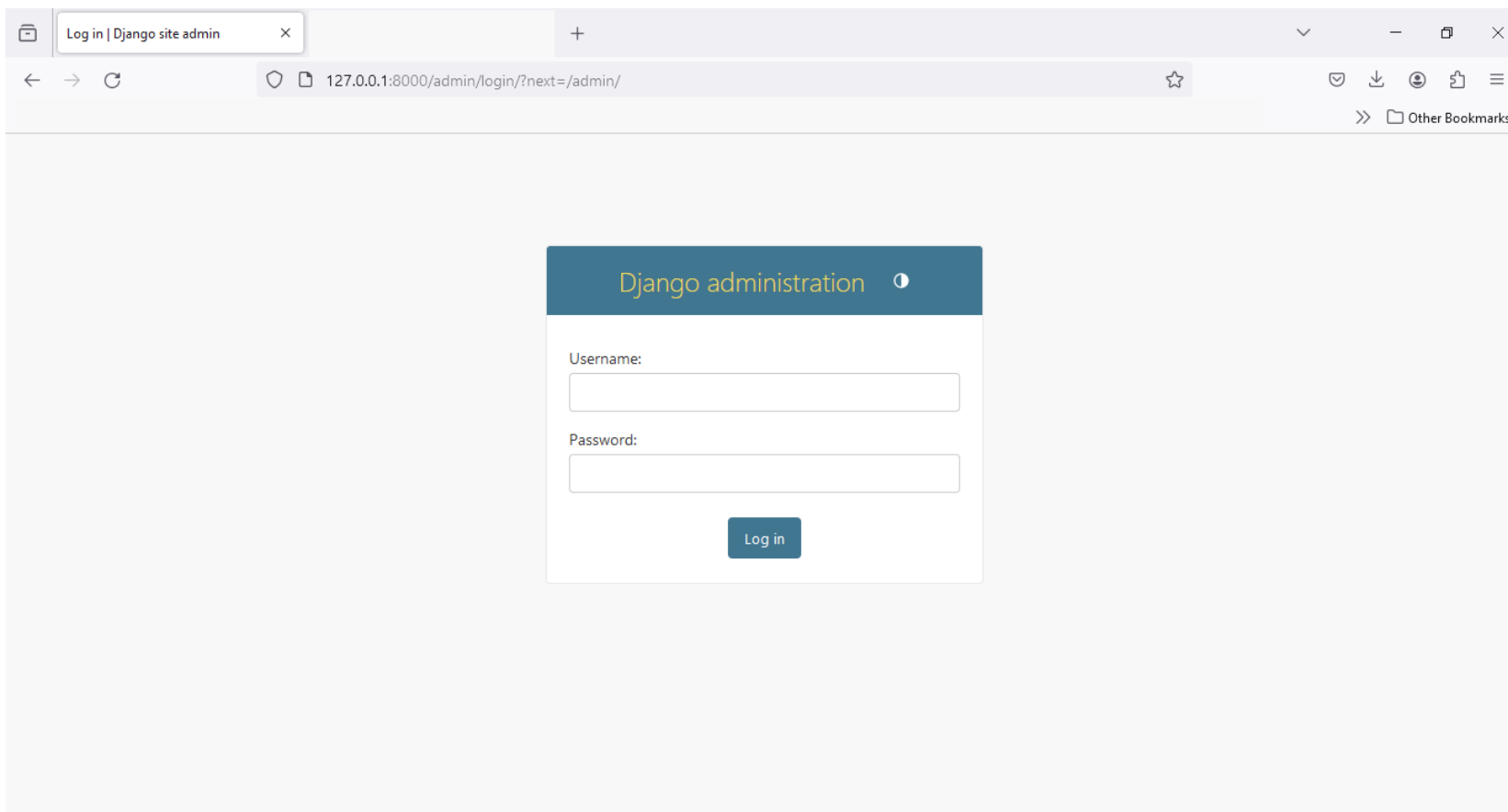




# Acceso al Panel de Administración

Ejecutar el servidor con: **python manage.py runserver**

Ingresa al panel con: **http://127.0.0.1:8000/admin/**

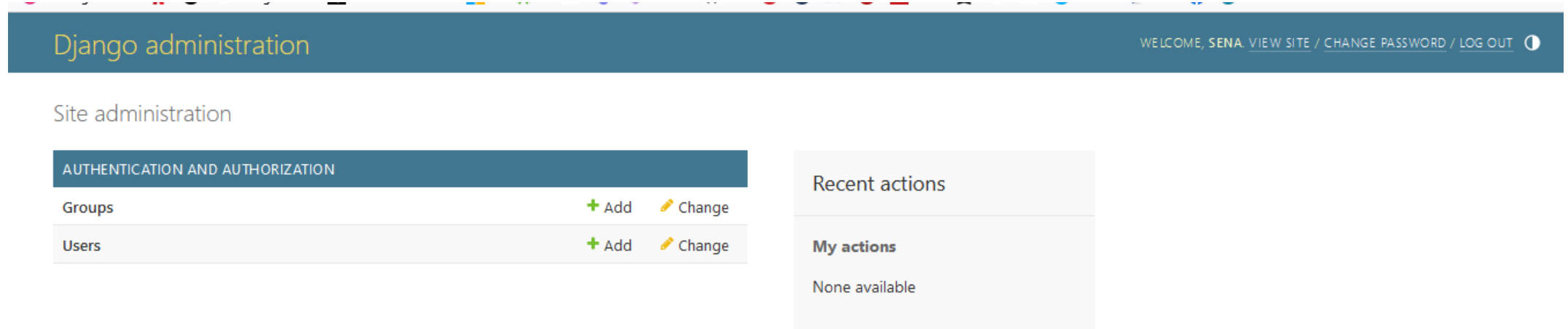


The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/admin/login/?next=/admin/`. The page content features a central login form titled "Django administration". The form includes two input fields: "Username:" and "Password:", each followed by a text input box. Below these fields is a blue "Log in" button. The browser's tab is labeled "Log in | Django site admin".



# Acceso al Panel de Administración

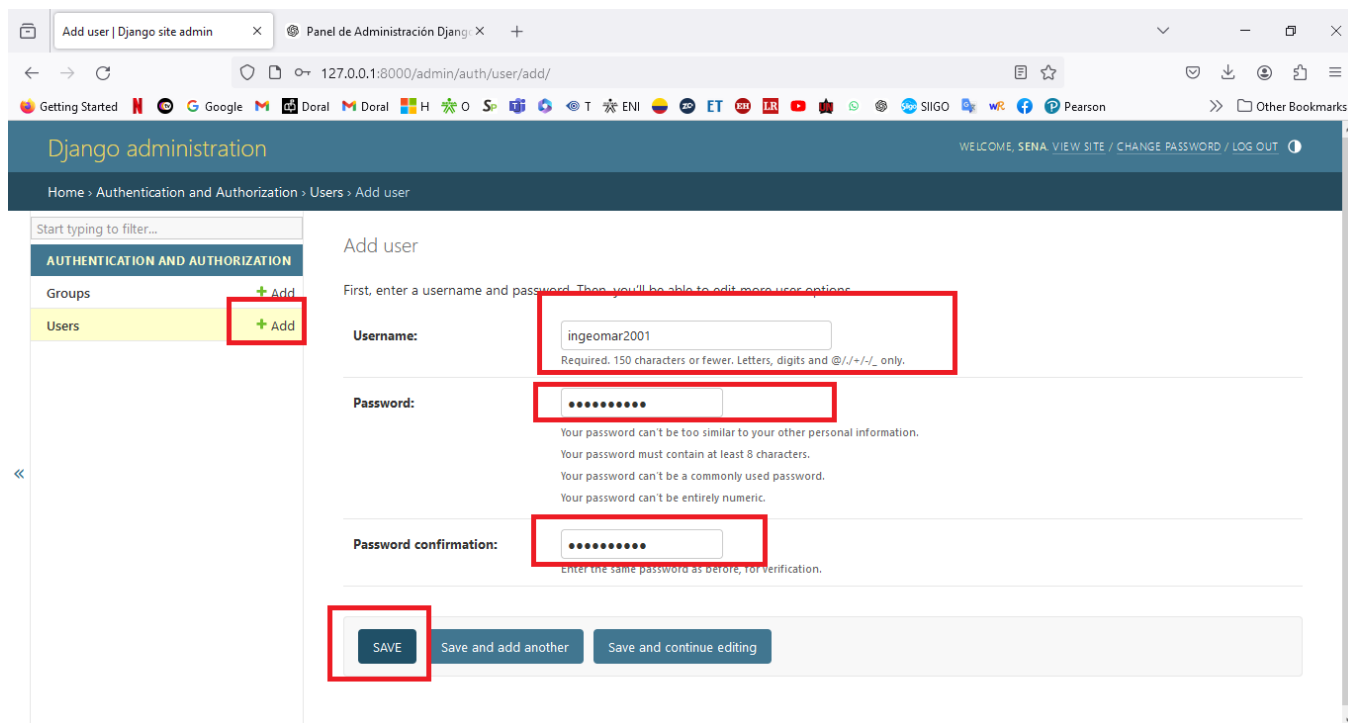
Ingresar al panel con el usuario y password creados





# Acceso al Panel de Administración

Podríamos añadir más usuarios



Django administration

Home > Authentication and Authorization > Users > Add user

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

Add user

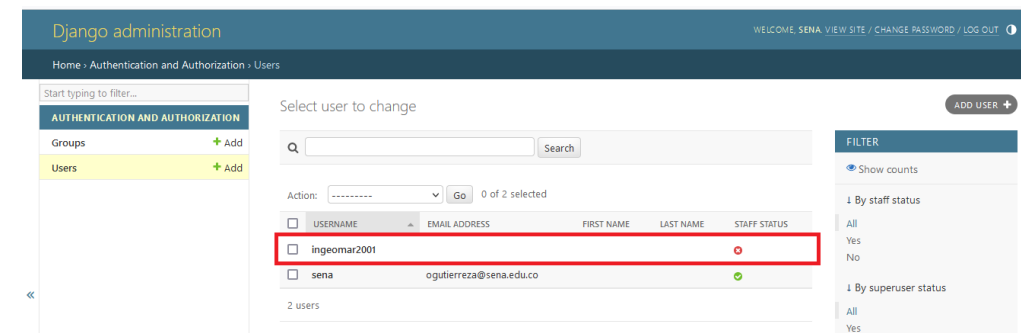
First, enter a username and password. Then, you'll be able to edit more user options.

Username: ingeomar2001  
Required: 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

Password: .....  
Your password can't be too similar to your other personal information.  
Your password must contain at least 8 characters.  
Your password can't be a commonly used password.  
Your password can't be entirely numeric.

Password confirmation: .....  
Enter the same password as before, for verification.

SAVE Save and add another Save and continue editing



Django administration

Home > Authentication and Authorization > Users

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

Select user to change

Q: Search

Action: 0 of 2 selected

	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
<input type="checkbox"/>	ingeomar2001				
<input type="checkbox"/>	seña	ogutierrez@seña.edu.co			

2 users

FILTER

Show counts

By staff status

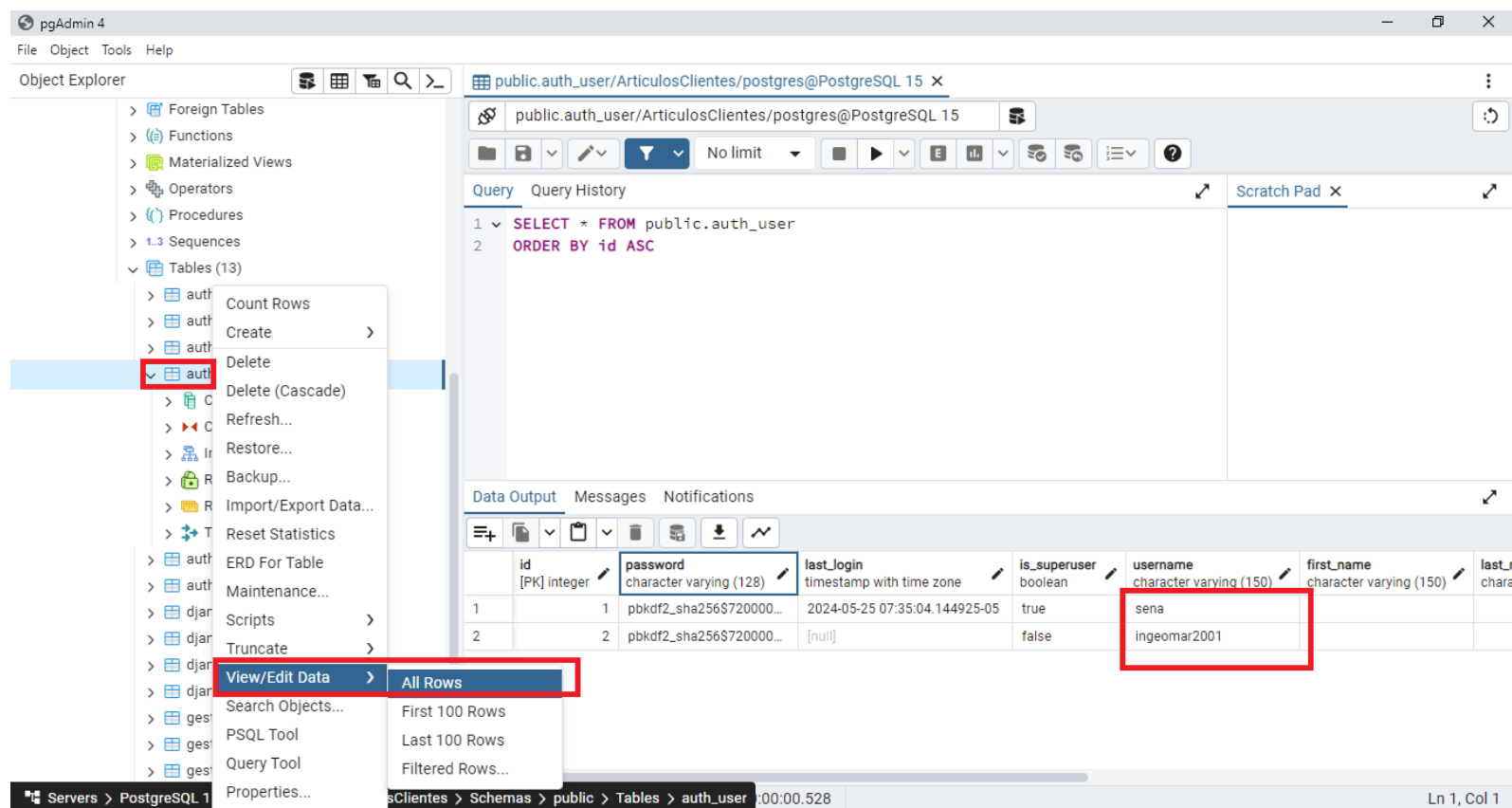
All  
Yes  
No

By superuser status

All  
Yes

# Panel de Administración en Nuestro Proyecto

Podemos ver los usuarios creados con pgadmin4 en : Base de Datos ArticulosClientes->Schema->public->Tables->auth\_user



The screenshot shows the pgAdmin 4 interface. In the Object Explorer on the left, the 'auth\_user' table is selected under 'Tables (13)'. A context menu is open over the table, with 'View/Edit Data' selected, and the 'All Rows' sub-option is highlighted. The main pane displays the table's structure and data. The table has columns: id (PK integer), password (character varying (128)), last\_login (timestamp with time zone), is\_superuser (boolean), username (character varying (150)), first\_name (character varying (150)), and last\_name (character varying (150)). The data shows two rows. The first row has id 1, password 'pbkdf2\_sha256\$720000...', last\_login '2024-05-25 07:35:04.144925-05', is\_superuser true, username 'sena', first\_name null, and last\_name null. The second row has id 2, password 'pbkdf2\_sha256\$720000...', last\_login null, is\_superuser false, username 'ingeomar2001', first\_name null, and last\_name null. The 'username' column for the first row is highlighted with a red box.

id	password	last_login	is_superuser	username	first_name	last_name
1	pbkdf2_sha256\$720000...	2024-05-25 07:35:04.144925-05	true	sena	[null]	[null]
2	pbkdf2_sha256\$720000...	[null]	false	ingeomar2001	[null]	[null]

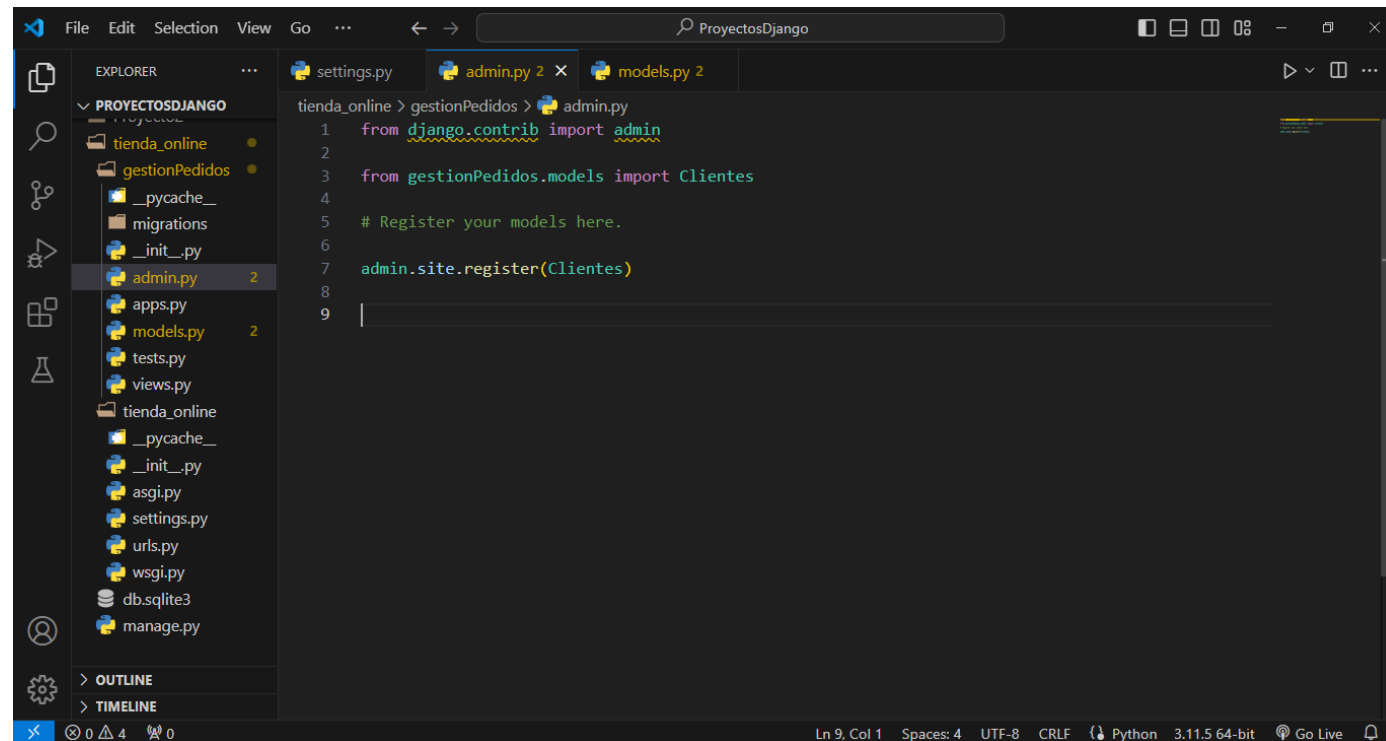
# Panel de Administración Registrar modelos

Se va a registrar el modelo de clientes. Primero, se importa el modelo de clientes en admin.py. Se añade la línea:

**from gestionPedidos.models import Clientes**

Después, se registra el modelo en el panel de administración con:

**admin.site.register(Clientes)**



```

tienda_online > gestionPedidos > admin.py
1  from django.contrib import admin
2
3  from gestionPedidos.models import Clientes
4
5  # Register your models here.
6
7  admin.site.register(Clientes)
8
9

```

# Panel de Administración Registrar modelos

Se guardan los cambios y se arranca el servidor con **python manage.py runserver**. Luego, se accede a la URL del panel de administración:

**http://localhost:8000/admin**

Se inicia sesión y se verifica que la tabla de clientes está disponible. Desde aquí se puede agregar y modificar registros, además de eliminar clientes y verificar los cambios en la base de datos. Nota: Django le agrega una 's' al final a todas las tablas

## Django administration

### Site administration

#### AUTHENTICATION AND AUTHORIZATION

Groups

[+ Add](#) [Change](#)

Users

[+ Add](#) [Change](#)

#### GESTIONPEDIDOS

Clientess

[+ Add](#) [Change](#)

# Panel de Administración Añadir Registros a Modelos

Al momento de agregar un usuario aparecen los campos de la tabla en negrita, eso quiere decir que son requeridos. Por defecto, Django hace que desde el panel de administración todos los campos sean requeridos. No se puede dejar ningún campo en blanco. Esto se puede modificar posteriormente. Cuando un campo sea opcional, aparecerá en un color normal, no en negrita,

Django administration

Home > Gestionpedidos > Clientess > Add clientes

Start typing to filter...

**AUTHENTICATION AND AUTHORIZATION**

Groups + Add

Users + Add

**GESTIONPEDIDOS**

Clientess + Add

Add clientes

**Nombre:**

**Direccion:**

**Email:**

**Tfno:**

**SAVE** Save and add another Save and continue editing

Django administration

Home > Gestionpedidos > Clientess

Start typing to filter...

**AUTHENTICATION AND AUTHORIZATION**

Groups + Add

Users + Add

**GESTIONPEDIDOS**

Clientess + Add

Select clientes to change

Action: ----- Go 0 of 2 selected

☐ CLIENTES

☐ Clientes object (2)

☐ Clientes object (1)

2 clientess



# Panel de Administración Modificar Registros a Modelos

Nos devolvemos al **Home** del panel de administración y seleccionamos la opción **Change**

Django administration

Home > Gestionpedidos > Clientess

Start typing to filter...

**AUTHENTICATION AND AUTHORIZATION**

Groups + Add

Users + Add

**GESTIONPEDIDOS**

Clientess + Add

Select clientes to change

Action: [-----] Go 0 of 2 selected

☐ CLIENTES

☐ Clientes object (2)

☐ Clientes object (1)

2 clientess

Django administration

Site administration

**AUTHENTICATION AND AUTHORIZATION**

Groups + Add Change

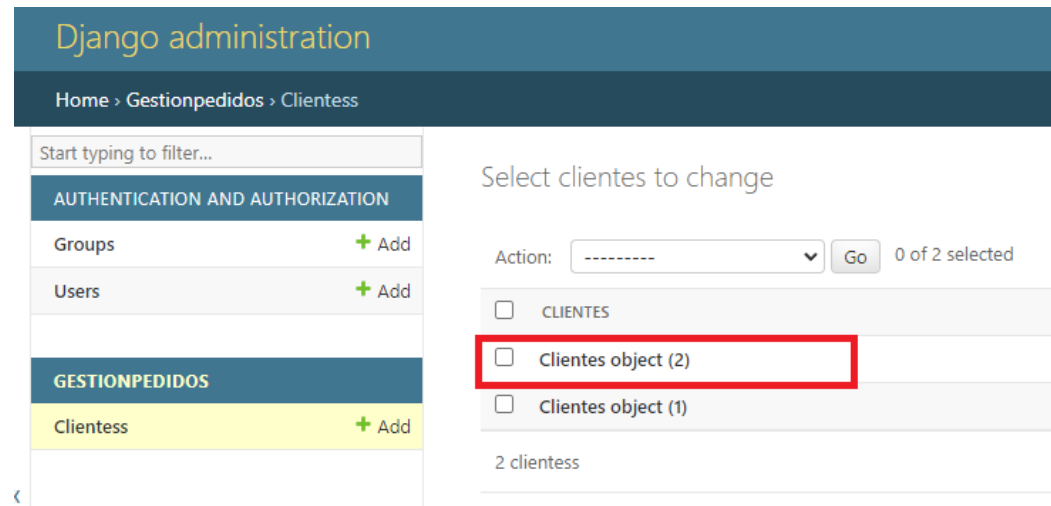
Users + Add Change

**GESTIONPEDIDOS**

Clientess + Add Change

# Panel de Administración Modificar Registros a Modelos

Se nos muestra un listado de objetos que representan los registros de clientes que tenemos. Seleccionamos el (2) para poder modificarlo



Django administration

Home > Gestionpedidos > Clientess

Start typing to filter...

**AUTHENTICATION AND AUTHORIZATION**

- Groups + Add
- Users + Add

**GESTIONPEDIDOS**

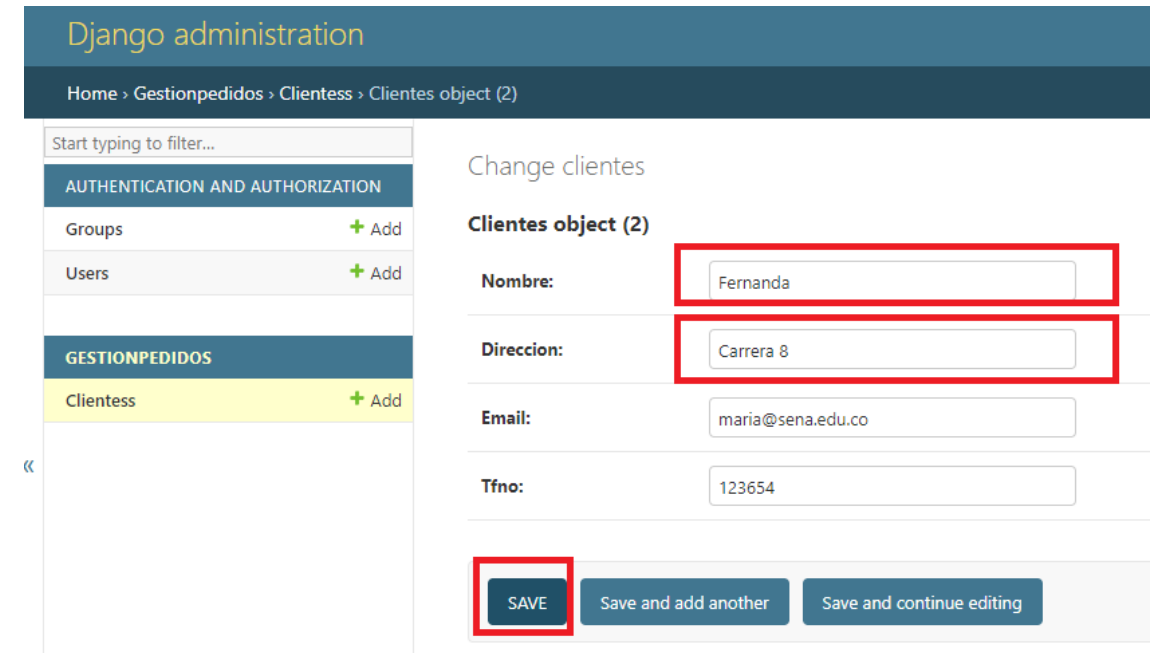
- Clientess + Add

Select clientes to change

Action: [dropdown] Go 0 of 2 selected

- ☐ CLIENTES
- ☒ Clientes object (2)
- ☐ Clientes object (1)

2 clientess



Django administration

Home > Gestionpedidos > Clientess > Clientes object (2)

Start typing to filter...

**AUTHENTICATION AND AUTHORIZATION**

- Groups + Add
- Users + Add

**GESTIONPEDIDOS**

- Clientess + Add

«

Change clientes

**Clientes object (2)**

Nombre:

Direccion:

Email:

Tfno:

# Panel de Administración Eliminar Registros a Modelos

En el listado de objetos que representan los registros de clientes que tenemos. Seleccionamos el (2) para poder borrarlo. Nos pregunta que si queremos borrarlo y le decimos que Si.

Django administration

Home > Gestionpedidos > Clientess

Start typing to filter...

**AUTHENTICATION AND AUTHORIZATION**

Groups + Add

Users + Add

**GESTIONPEDIDOS**

Clientess + Add

Select clientes to change

**2** Action: Delete selected clientess **3** Go 1 of 2 selected

**1** ☒ Clientes object (2)

☐ CLIENTES

☐ Clientes object (1)

2 clientess

Django administration

Home > Gestionpedidos > Clientess > Delete multiple objects

Start typing to filter...

**AUTHENTICATION AND AUTHORIZATION**

Groups + Add

Users + Add

**GESTIONPEDIDOS**

Clientess + Add

Are you sure?

Are you sure you want to delete the selected clientes? All of the following objects and their related items will be deleted:

**Summary**

- Clientess: 1

**Objects**

- Clientes: Clientes object (2)

**Yes, I'm sure** **No, take me back**

# Panel de Administración Agregar más Modelos

Agregamos las tablas en la importación y registro del Admin y guardamos. No es necesario realizar migraciones ya que los modelos no se han modificado. En la página del panel hacemos F5 para actualizar.

## Django administration

### Site administration

#### AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#) [Change](#)

Users [+ Add](#) [Change](#)

#### GESTIONPEDIDOS

Articuloss [+ Add](#) [Change](#)

Clientess [+ Add](#) [Change](#)

Pedidoss [+ Add](#) [Change](#)

### Recent actions

#### My actions

✖ Clientes object (2)  
Clientes

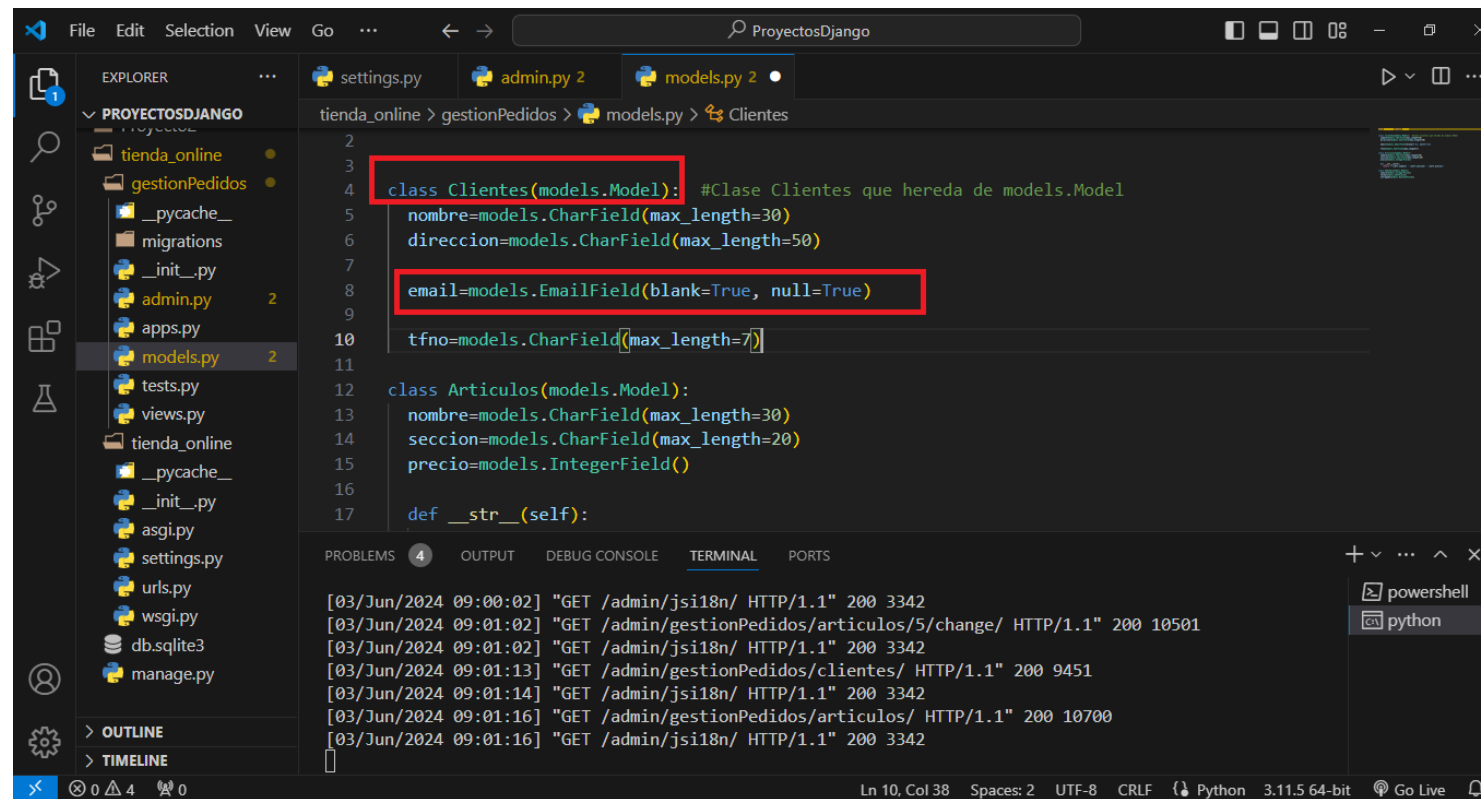
[Change](#) Clientes object (2)  
Clientes

+ Clientes object (2)  
Clientes

+ ingeomar2001  
User

# Panel de Administración Campos No Obligatorios

Si queremos hacer opcional el campo de email en la tabla de clientes, editamos el modelo de clientes y añadimos `blank=True, null=True` al campo email:



```

2
3
4 class Clientes(models.Model): #Clase Clientes que hereda de models.Model
5     nombre=models.CharField(max_length=30)
6     direccion=models.CharField(max_length=50)
7
8     email=models.EmailField(blank=True, null=True)
9
10    tfno=models.CharField(max_length=7)
11
12 class Articulos(models.Model):
13     nombre=models.CharField(max_length=30)
14     seccion=models.CharField(max_length=20)
15     precio=models.IntegerField()
16
17     def __str__(self):

```

# Panel de Administración Campos No Obligatorios

Guardamos los cambios y realizamos una migración:

```
C:\Users\SENA\Documents\ProyectosDjango\tienda_online>workon mi_ambiente_django
(mi_ambiente_django) C:\Users\SENA\Documents\ProyectosDjango\tienda_online>python manage.py makemigrations
Migrations for 'gestionPedidos':
  gestionPedidos\migrations\0002_alter_clientes_email.py
    - Alter field email on clientes

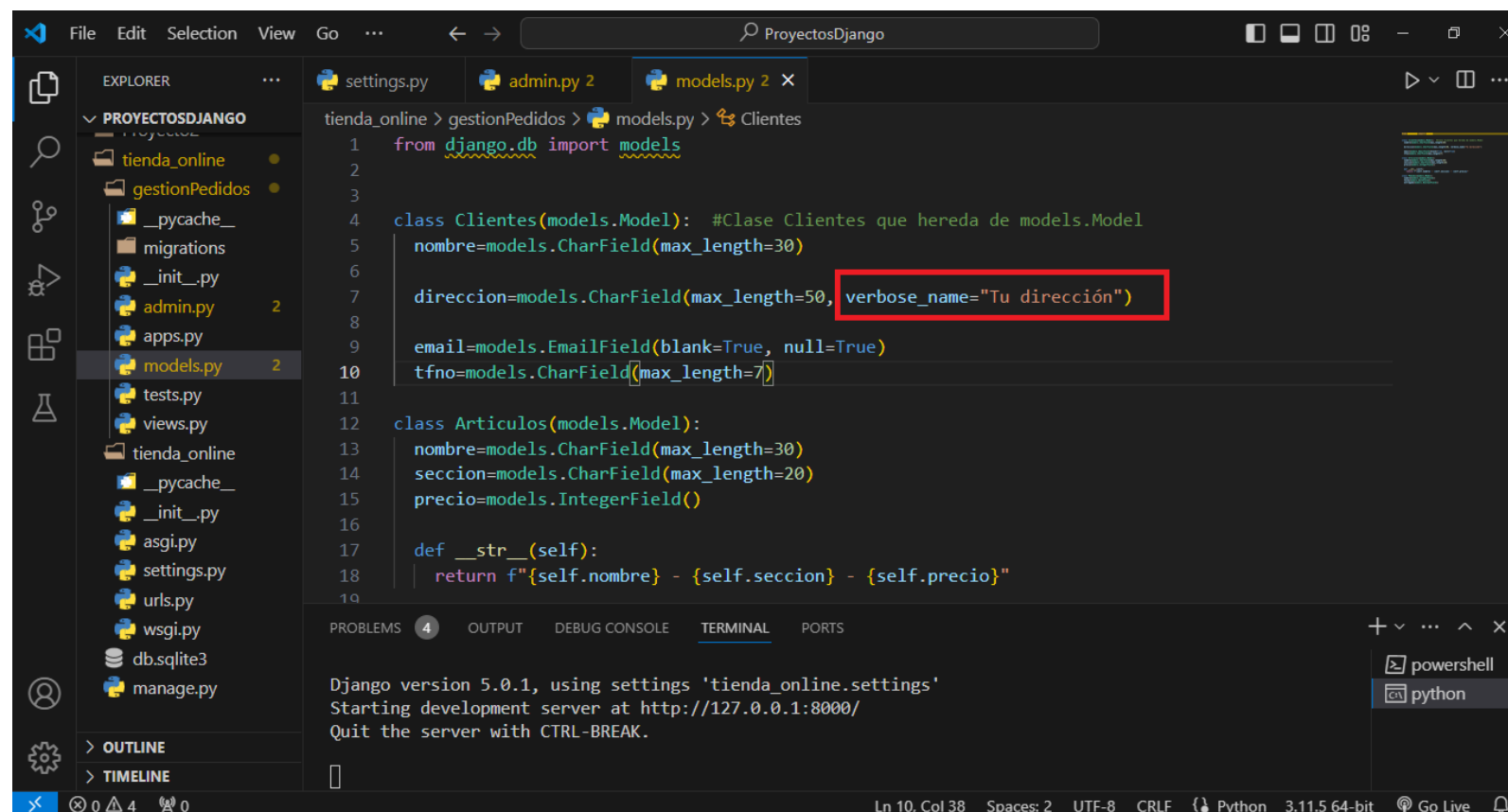
(mi_ambiente_django) C:\Users\SENA\Documents\ProyectosDjango\tienda_online>
```

```
(mi ambiente django) C:\Users\SENA\Documents\ProyectosDjango\tienda_online>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, gestionPedidos, sessions
Running migrations:
  Applying gestionPedidos.0002_alter_clientes_email... OK
```

Refrescamos el panel de administración y verificamos que el campo email ya no es obligatorio

# Panel de Administración Cambiar nombre del campo a mostrar en el panel

Para cambiar los nombres de los campos de las tablas en el panel de administración de Django, se configura el atributo `verbose_name` en el modelo:

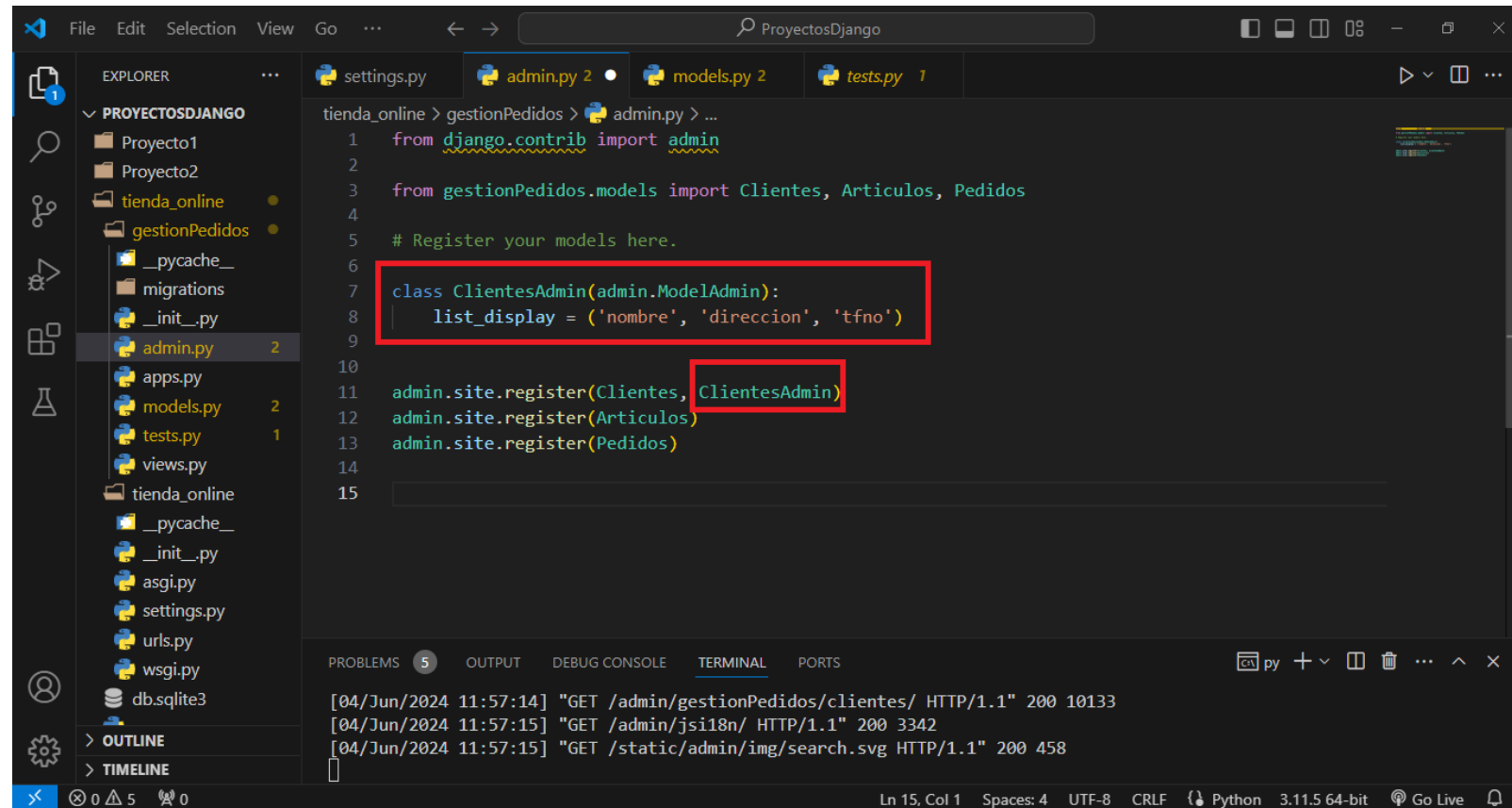


```
1 from django.db import models
2
3
4 class Clientes(models.Model): #Clase Clientes que hereda de models.Model
5     nombre=models.CharField(max_length=30)
6
7     direccion=models.CharField(max_length=50, verbose_name="Tu dirección")
8
9     email=models.EmailField(blank=True, null=True)
10    tfno=models.CharField(max_length=7)
11
12 class Articulos(models.Model):
13     nombre=models.CharField(max_length=30)
14     seccion=models.CharField(max_length=20)
15     precio=models.IntegerField()
16
17     def __str__(self):
18         return f"{self.nombre} - {self.seccion} - {self.precio}"
19
```

Django version 5.0.1, using settings 'tienda\_online.settings'  
Starting development server at http://127.0.0.1:8000/  
Quit the server with CTRL-BREAK.

# Panel de Administración Cambiar nombre del campo a mostrar en el administrador

En el archivo admin.py, se crea una clase que hereda de admin.ModelAdmin y se registra el modelo con esta clase:



```

tienda_online > gestionPedidos > admin.py > ...
1  from django.contrib import admin
2
3  from gestionPedidos.models import Clientes, Articulos, Pedidos
4
5  # Register your models here.
6
7  class ClientesAdmin(admin.ModelAdmin):
8      list_display = ('nombre', 'direccion', 'tfno')
9
10
11  admin.site.register(Clientes, ClientesAdmin)
12  admin.site.register(Articulos)
13  admin.site.register(Pedidos)
14
15

```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

[04/Jun/2024 11:57:14] "GET /admin/gestionPedidos/clientes/ HTTP/1.1" 200 10133
[04/Jun/2024 11:57:15] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[04/Jun/2024 11:57:15] "GET /static/admin/img/search.svg HTTP/1.1" 200 458

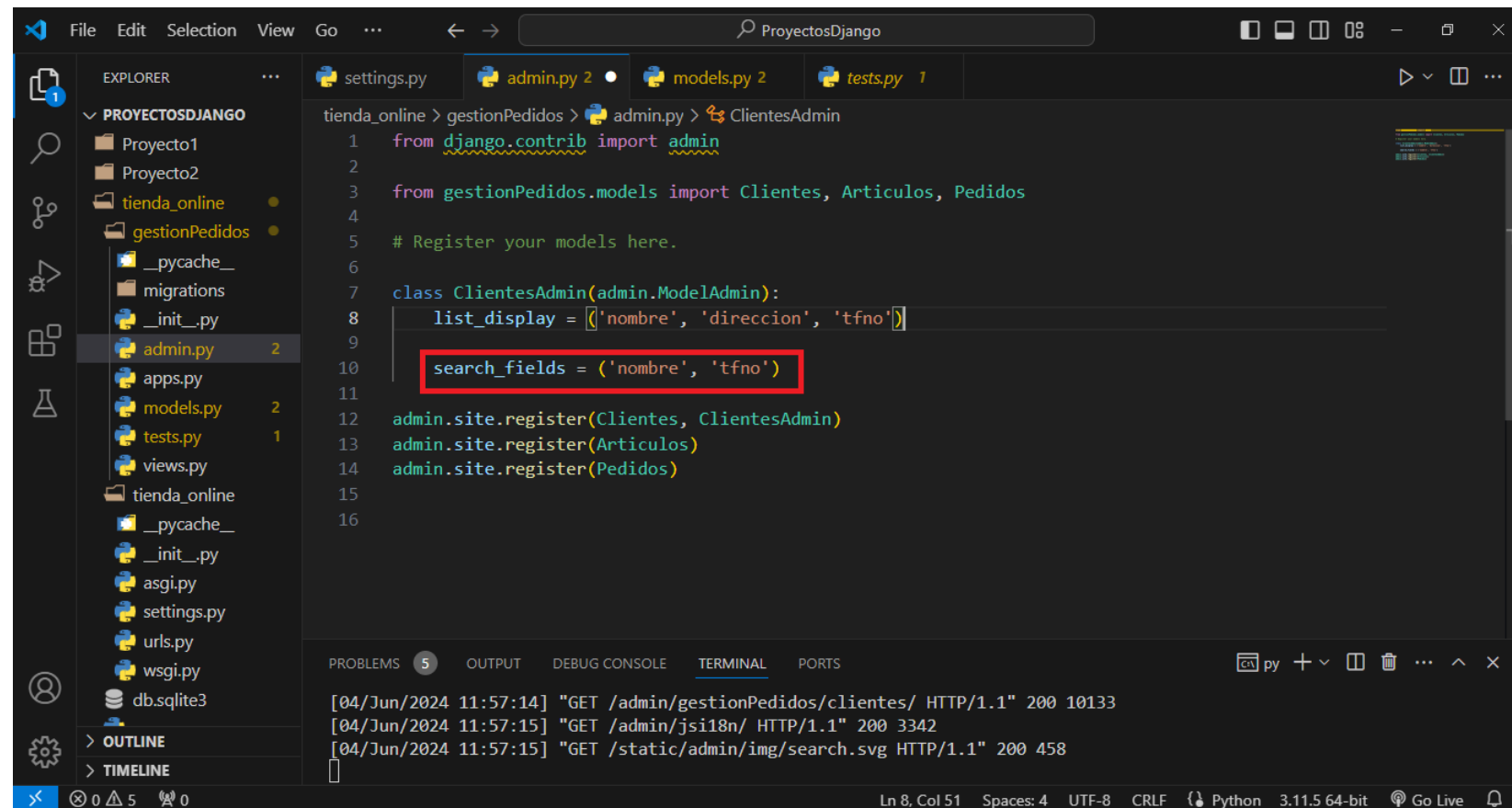
```

Ln 15, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.5 64-bit Go Live



# Panel de Administración Configurar campos de búsqueda

En el archivo admin.py, dentro de la clase que hereda de admin.ModelAdmin, se define el atributo search\_fields.



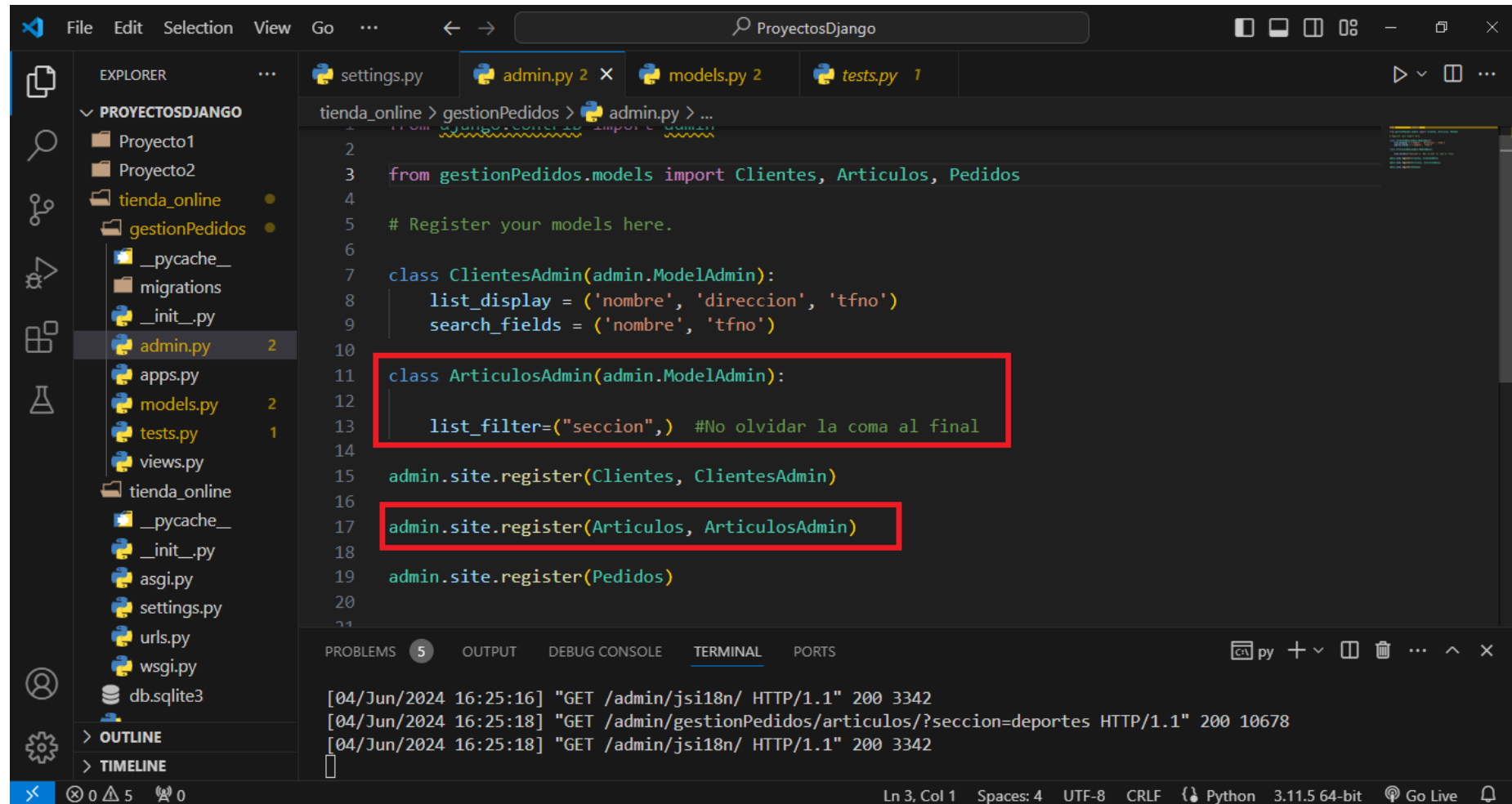
```
1 from django.contrib import admin
2
3 from gestionPedidos.models import Clientes, Articulos, Pedidos
4
5 # Register your models here.
6
7 class ClientesAdmin(admin.ModelAdmin):
8     list_display = ('nombre', 'direccion', 'tfno')
9
10     search_fields = ('nombre', 'tfno')
11
12 admin.site.register(Clientes, ClientesAdmin)
13 admin.site.register(Articulos)
14 admin.site.register(Pedidos)
15
16
```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS

[04/Jun/2024 11:57:14] "GET /admin/gestionPedidos/clientes/ HTTP/1.1" 200 10133  
[04/Jun/2024 11:57:15] "GET /admin/jsi18n/ HTTP/1.1" 200 3342  
[04/Jun/2024 11:57:15] "GET /static/admin/img/search.svg HTTP/1.1" 200 458

Ln 8, Col 51 Spaces: 4 UTF-8 CRLF Python 3.11.5 64-bit Go Live

# Panel de Administración Agregar Filtro



The screenshot shows a VS Code editor window with the following structure:

- EXPLORER:**
  - PROYECTOSDJANGO
    - Proyecto1
    - Proyecto2
    - tienda\_online
      - gestionPedidos
        - \_\_pycache\_\_
        - migrations
        - \_\_init\_\_.py
        - admin.py (2)
          - apps.py
          - models.py (2)
          - tests.py (1)
          - views.py
        - tienda\_online
          - \_\_pycache\_\_
          - \_\_init\_\_.py
          - asgi.py
          - settings.py
          - urls.py
          - wsgi.py
          - db.sqlite3

- EDITOR:**
- admin.py
 

```

1 from django.contrib import admin
2
3 from gestionPedidos.models import Clientes, Articulos, Pedidos
4
5 # Register your models here.
6
7 class ClientesAdmin(admin.ModelAdmin):
8     list_display = ('nombre', 'direccion', 'tfno')
9     search_fields = ('nombre', 'tfno')
10
11 class ArticulosAdmin(admin.ModelAdmin):
12
13     list_filter=("seccion",) #No olvidar la coma al final
14
15 admin.site.register(Clientes, ClientesAdmin)
16
17 admin.site.register(Articulos, ArticulosAdmin)
18
19 admin.site.register(Pedidos)
20
21
          
```
- TERMINAL:**

```

[04/Jun/2024 16:25:16] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[04/Jun/2024 16:25:18] "GET /admin/gestionPedidos/articulos/?seccion=deportes HTTP/1.1" 200 10678
[04/Jun/2024 16:25:18] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
          
```



# GRACIAS

Línea de atención al ciudadano: 01 8000 910270  
Línea de atención al empresario: 01 8000 910682



[www.sena.edu.co](http://www.sena.edu.co)