

Proyecto Neumatic

Introducción

1. Instalación de Python

<https://www.python.org/>

2. Instalación de Visual Studio Code

<https://code.visualstudio.com/>

3. Acerca de PEP8

PEP 8 es la **Guía de Estilo para el Código Python**. Define las mejores prácticas para escribir código Python más legible y consistente. A continuación, un resumen de sus principales recomendaciones:

3.1. Indentación

- Usa 4 espacios por nivel de indentación. No mezcles tabulaciones con espacios.

3.2. Longitud de líneas

- Limita las líneas a 79 caracteres para facilitar la lectura y evitar el scroll horizontal.

3.3. Espacios en blanco

- Usa espacios para mejorar la legibilidad, pero sin excesos:
 - Coloca un espacio después de comas y alrededor de operadores aritméticos (`a = b + 1`).
 - No coloques espacios en exceso dentro de paréntesis o listas (`func(1)` , no `func(1)`).

3.4. Nombres de variables y funciones

- Usa **snake_case** para nombres de funciones y variables (`mi_variable` , `calcular_total`).
- Usa **CamelCase** para nombres de clases (`MiClase`).

- Las constantes se nombran en **mayúsculas** con guiones bajos (`PI` , `MAX_VALOR`).

3.5. Comentarios

- Los comentarios deben ser **claros y concisos**:
 - Usa comentarios de una línea con `#` .
 - Usa **docstrings** para documentar módulos, clases y funciones, entre comillas triples (`"""`).

3.6. Importaciones

- Coloca todas las **importaciones** al inicio del archivo.
- Importa módulos completos en lugar de funciones específicas, a menos que sea necesario.
- El orden de las importaciones debe ser:
 1. Módulos estándar de Python.
 2. Módulos de terceros.
 3. Módulos locales.

3.7. Funciones y métodos

- Deja dos líneas en blanco entre definiciones de clases o funciones a nivel superior.
- Deja una línea en blanco entre métodos de clases.

3.8. Convenciones de encadenado

- Si una expresión es muy larga, se puede dividir en varias líneas, usando **paréntesis** o **líneas continuadas** con una barra invertida (`\`).

3.9. Control de excepciones

- Maneja las excepciones con bloques `try-except` , usando excepciones específicas y no genéricas.

3.10. Comparaciones

- Para comparaciones con valores de un tipo vacío como `None` , `True` o `False` , utiliza las palabras clave (`if x is None` , NO `if x == None`).

Siguiendo estas reglas, el código es más legible, mantenible y consistente con el estilo de la comunidad Python.

El enlace web de PEP 8 es:

[PEP 8 – Style Guide for Python Code | peps.python.org](https://peps.python.org/pep-0008/)

4. Notaciones CamelCase y snake_case

Cuando se crea un identificador de archivo, variable, objeto, clase, etc, que involucra varias palabras y no es posible usar espacios en blanco, debemos unir las palabras. Para tener una mejor visibilidad del identificador, existen varias formas de hacerlo, en nuestro caso vamos a usar las notaciones:

UpperCamelCase (NotacionUpperCamelCase)

Snake Case (notacion_snake_case)

4.1. Notación UpperCamelCase

La notación **UpperCamelCase** (más conocida como PascalCase) establece que en un identificador, la primera letra de cada palabra debe ser en mayúscula y el resto de las letras de la palabra (no de todo el identificador)

| Identificación Genérica | Identificador (UpperCamelCase) |
|--------------------------|--------------------------------|
| personas | Personas |
| clientes | Clientes |
| productos | Productos |
| zonas horarias | ZonasHorarias |
| tipo documento identidad | TipoDocumentoidentidad |

4.2. Notación Snake Case

La notación **Snake Case** combina las palabras usando un guión bajo `_` como nexos. Existen dos versiones de esta notación, una en la que todas las letras están en minúscula y otra en la que todas las letras están en mayúscula.

| Identificación Genérica | Identificador (Snake Case) |
|--------------------------|----------------------------|
| personas | personas |
| clientes | clientes |
| productos | productos |
| zonas horarias | ZONAS_HORARIAS |
| tipo documento identidad | tipo_documento_identidad |

5. Proyecto Neumatic

5.1. Carpeta contenedora del Proyecto

La carpeta contenedora del proyecto es la base para la creación del Proyecto. Procedemos a crear la carpeta contenedora del Proyecto, en la ubicación que definamos. Sin embargo, en nuestro caso trabajaremos desde el prompt del MDOS o Power Shell, con la siguiente ubicación inicial:

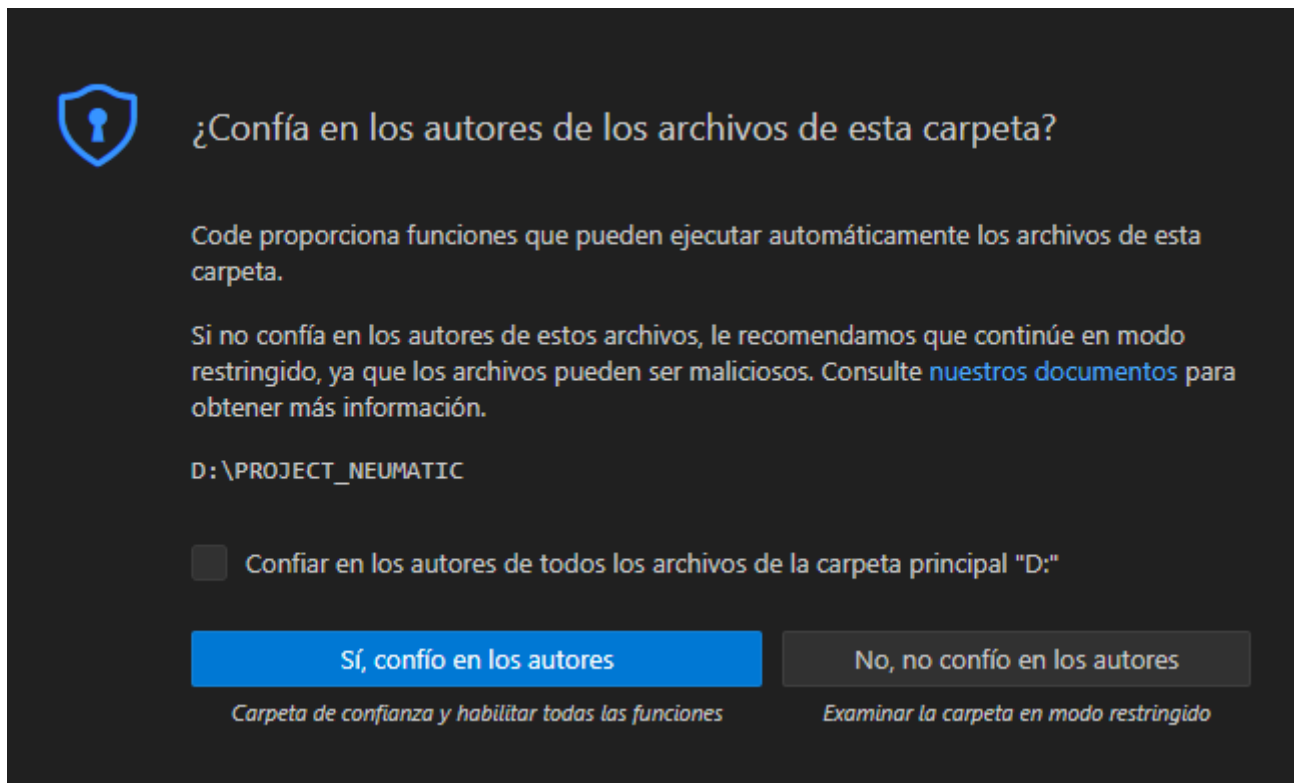
```
D:\> MD PROJECT_NEUMATIC
D:\> CD PROJECT_NEUMATIC
D:\PROJECT_NEUMATIC>
```

5.2. Ejecutar Visual Studio Code

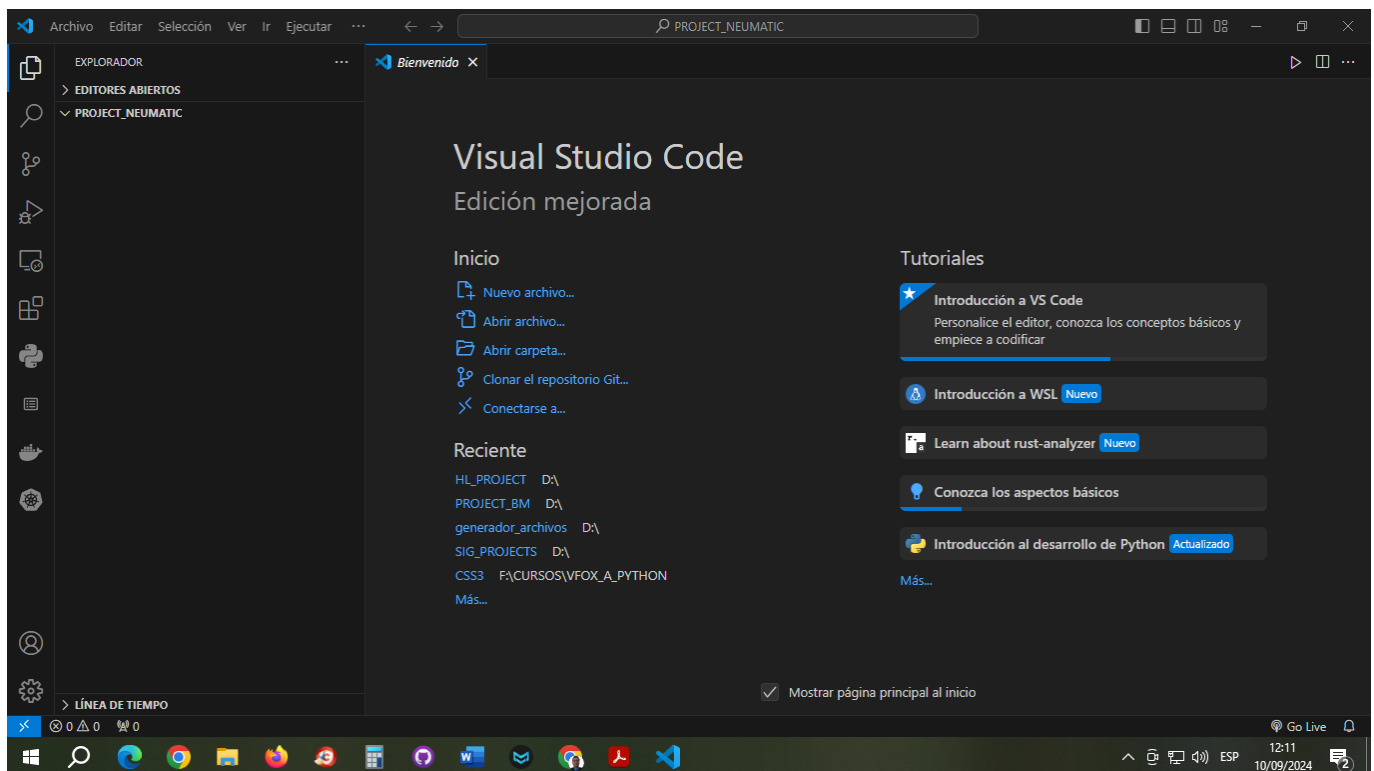
Al estar ubicados en la carpeta contenedora del proyecto. Escribimos la instrucción (code espacio punto), invocamos a **Visual Studio Code**, que en adelante llamaremos **VSC**.

```
D:\PROJECT_NEUMATIC>code .
```

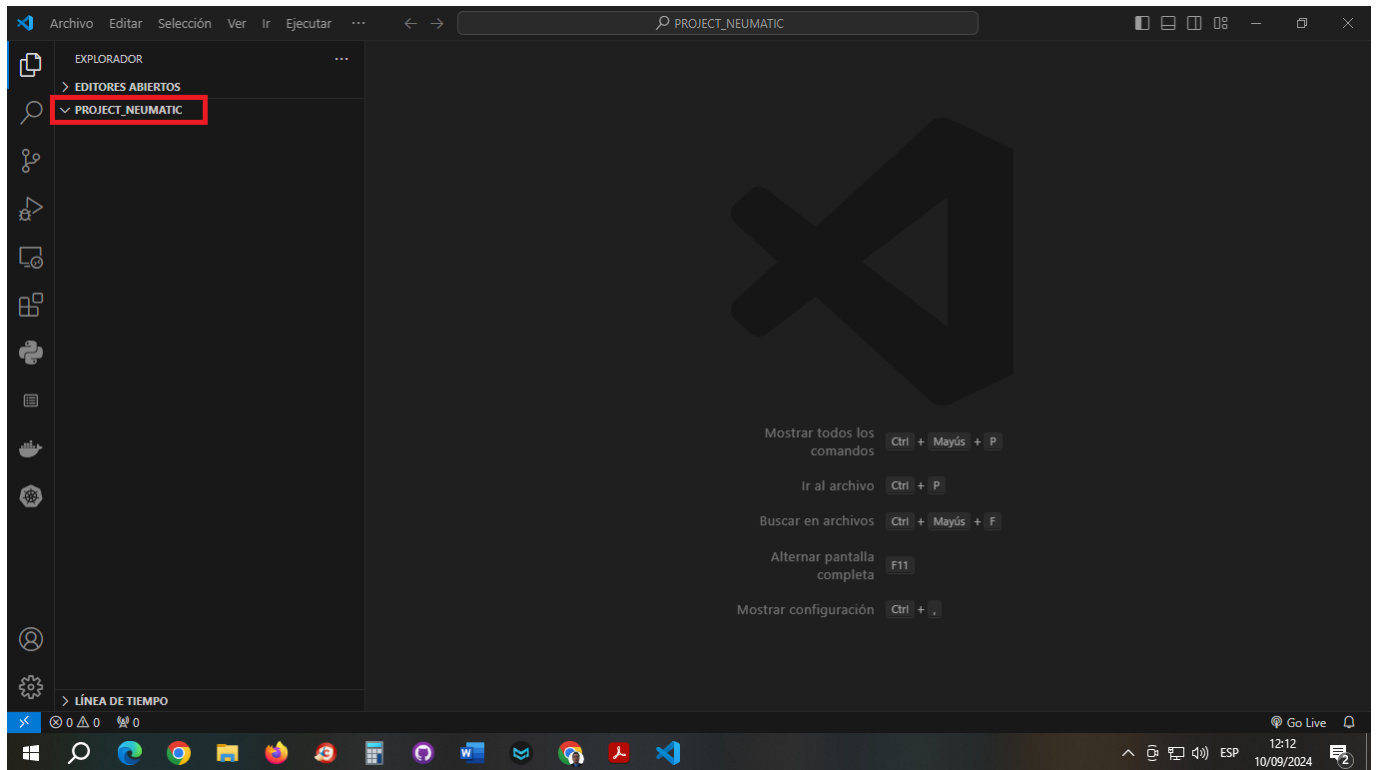
Es muy probable que aparezca la siguiente advertencia, a la cual debe reponder responder afirmativamente (Sí, confío en los autores).



Al confirmar, observaremos la pantalla de trabajo de VSC.

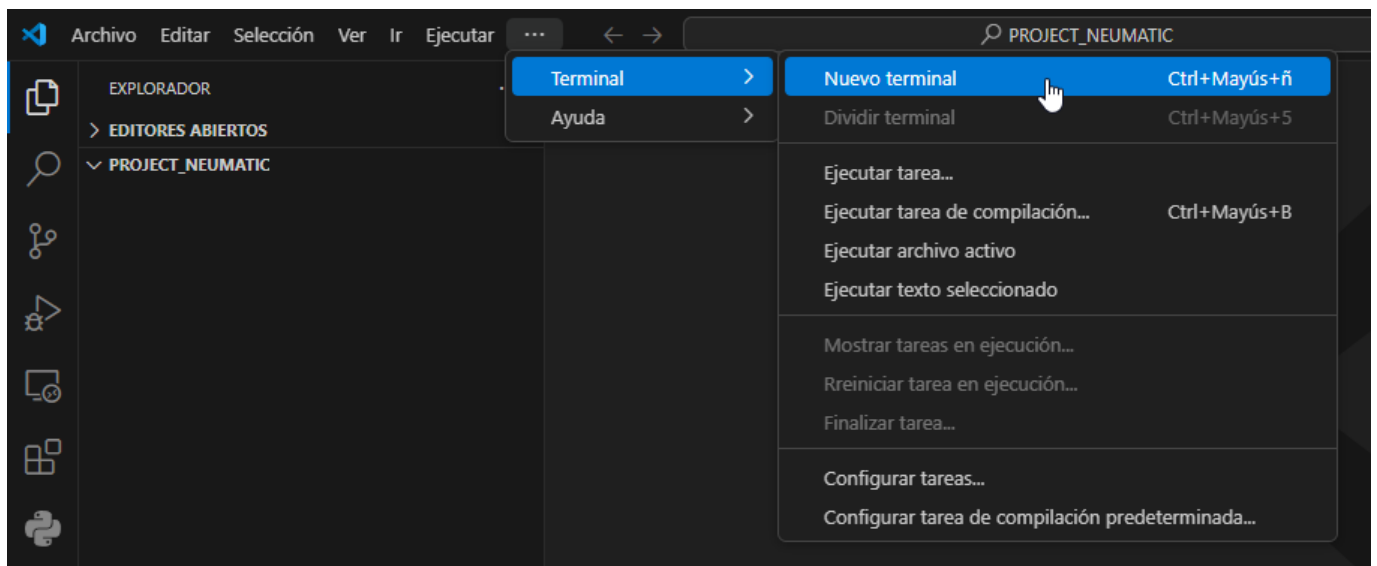


Luego de cerrar la pantalla de bienvenida, observaremos en la parte superior izquierda, el nombre de la carpeta contenedora del Proyecto.

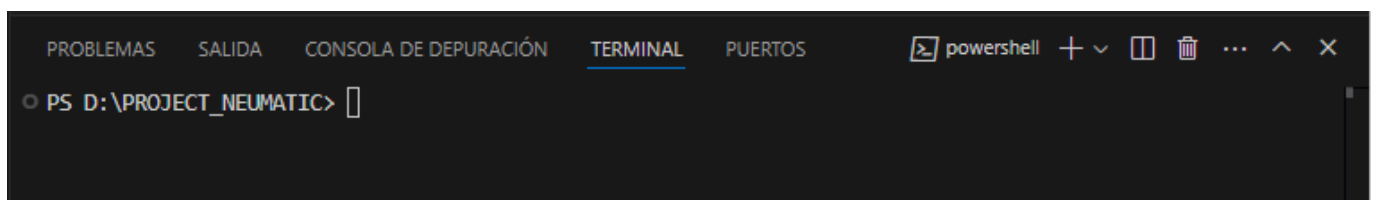


5.3. Abrir una nueva Terminal en VSC

Desde el menú de VSC seleccione la opción **Terminal**, luego la opción **Nuevo Terminal** o **New Terminal**.



Al final debe aparecer en la parte inferior derecha del editor de código de VSC lo siguiente:



5.4. El entorno virtual

Un **entorno virtual en Python** es una herramienta que permite crear un espacio aislado para instalar y gestionar dependencias de proyectos sin interferir con las de otros proyectos o con las del sistema global.

5.4.1. ¿Por qué usar un entorno virtual?

Aislamiento de dependencias:

- Cada proyecto puede tener su propio conjunto de bibliotecas y versiones, sin conflicto con otros proyectos o con las bibliotecas instaladas globalmente en el sistema.
- Si un proyecto requiere una versión de una biblioteca diferente a la de otro, no habrá problemas de compatibilidad.

Fácil portabilidad:

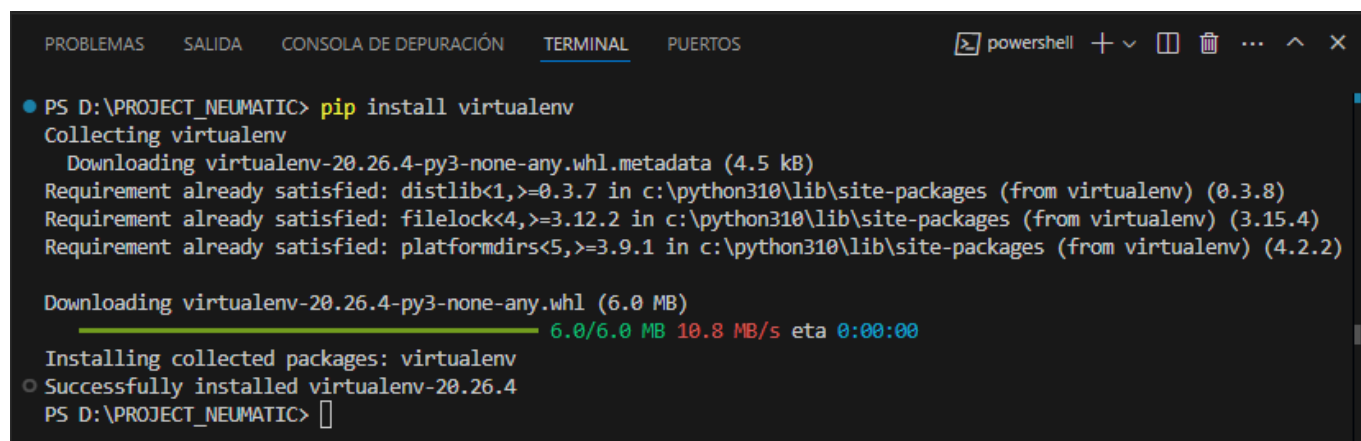
- Los entornos virtuales facilitan la compartición del proyecto con otras personas. Con un archivo como `requirements.txt`, que lista las dependencias, cualquiera puede recrear el entorno del proyecto en su propio sistema.

Previene errores en el sistema:

- Las instalaciones globales pueden afectar otros programas del sistema o proyectos, lo cual se evita con el aislamiento que ofrece un entorno virtual.

5.4.2. Instalación del paquete virtualenv

Para crear el entorno virtual, debemos instalar módulo de Python **virtualenv**, con la instrucción:
pip install virtualenv



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  powershell + v [ ] [ ] ... ^ X

● PS D:\PROJECT_NEUMATIC> pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.26.4-py3-none-any.whl.metadata (4.5 kB)
Requirement already satisfied: distlib<1,>=0.3.7 in c:\python310\lib\site-packages (from virtualenv) (0.3.8)
Requirement already satisfied: filelock<4,>=3.12.2 in c:\python310\lib\site-packages (from virtualenv) (3.15.4)
Requirement already satisfied: platformdirs<5,>=3.9.1 in c:\python310\lib\site-packages (from virtualenv) (4.2.2)

Downloading virtualenv-20.26.4-py3-none-any.whl (6.0 MB)
 6.0/6.0 MB 10.8 MB/s eta 0:00:00
Installing collected packages: virtualenv
○ Successfully installed virtualenv-20.26.4
PS D:\PROJECT_NEUMATIC> [ ]
```

5.4.3. Creación del entorno virtual

Para crear el entorno virtual, debemos ejecutar la instrucción: **python -m virtualenv venv**, donde **venv** es la carpeta donde se almacenarán las dependencias.

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
PS D:\PROJECT_NEUMATIC> python -m virtualenv venv
created virtual environment CPython3.10.1.final.0-64 in 2875ms
  creator CPython3Windows(dest=D:\PROJECT_NEUMATIC\venv, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\Ricardo
Ramos\AppData\Local\pypa\virtualenv)
  added seed packages: pip==24.2, setuptools==72.2.0, wheel==0.44.0
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
PS D:\PROJECT_NEUMATIC> 
```

5.4.4. Activación del entorno virtual

Para activar el entorno virtual, debemos ejecutar la instrucción: **venv\Scripts\activate**

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
PS D:\PROJECT_NEUMATIC> venv\Scripts\activate
○ (venv) PS D:\PROJECT_NEUMATIC> 
```

La indicación del entorno virtual activado es el nombre de la carpeta **venv** entre paréntesis, como prefijo del prompt del MS-DOS o Shell: **(venv)**

5.5. Creación del Proyecto Neumatic

5.5.1. Instalación del framework Django

Con el entorno virtual activado, instalamos el framework Django con la siguiente instrucción: **pip install django**

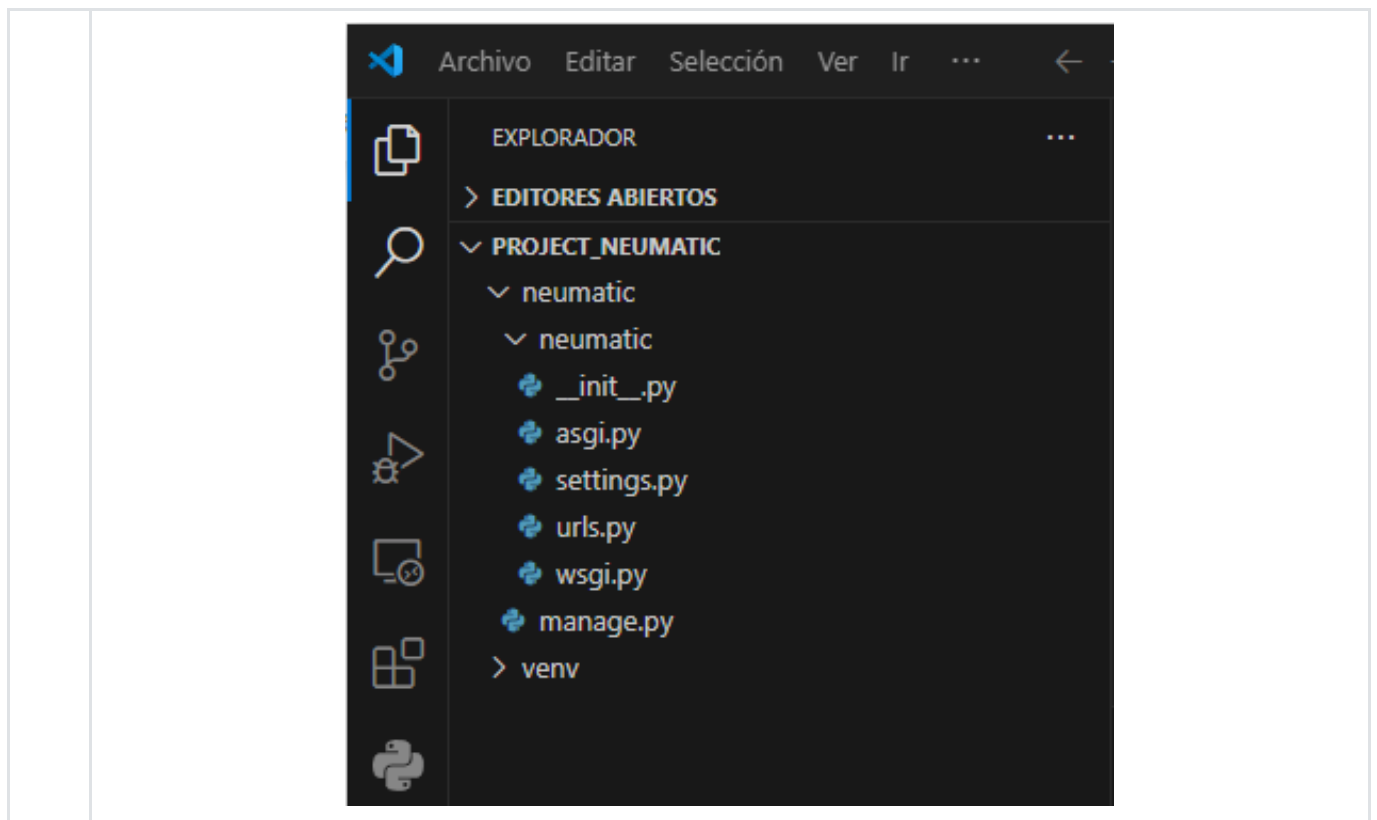
```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
● (venv) PS D:\PROJECT_NEUMATIC> pip install django
Collecting django
  Using cached Django-5.1.1-py3-none-any.whl.metadata (4.2 kB)
Collecting asgiref<4,>=3.8.1 (from django)
  Using cached asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from django)
  Using cached sqlparse-0.5.1-py3-none-any.whl.metadata (3.9 kB)
Collecting tzdata (from django)
  Using cached tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting typing-extensions>=4 (from asgiref<4,>=3.8.1->django)
  Using cached typing_extensions-4.12.2-py3-none-any.whl.metadata (3.0 kB)
Using cached Django-5.1.1-py3-none-any.whl (8.2 MB)
Using cached asgiref-3.8.1-py3-none-any.whl (23 kB)
Using cached sqlparse-0.5.1-py3-none-any.whl (44 kB)
Using cached tzdata-2024.1-py2.py3-none-any.whl (345 kB)
Using cached typing_extensions-4.12.2-py3-none-any.whl (37 kB)
Installing collected packages: tzdata, typing-extensions, sqlparse, asgiref, django
Successfully installed asgiref-3.8.1 django-5.1.1 sqlparse-0.5.1 typing-extensions-4.12.2 tzdata-2024.1
○ (venv) PS D:\PROJECT_NEUMATIC> 
```

5.5.2. Creación del Proyecto Neumatic

Con el entorno virtual activado, creamos el proyecto **neumatic** con la siguiente instrucción:
django-admin startproject neumatic

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  powershell + - □ □ ... ^ X
(venv) PS D:\PROJECT_NEUMATIC> django-admin startproject neumatic
(venv) PS D:\PROJECT_NEUMATIC> 
```

En el explorador de VSC observará la estructura inicial del Proyecto neumatic:



Al crearse un proyecto, se crea una carpeta principal con ese mismo nombre, y dentro de él se crea otra carpeta con el mismo nombre del proyecto. Sin embargo, la segunda carpeta tiene la utilidad de ser la carpeta de configuración del proyecto, donde el archivo de configuración es `settings.py`

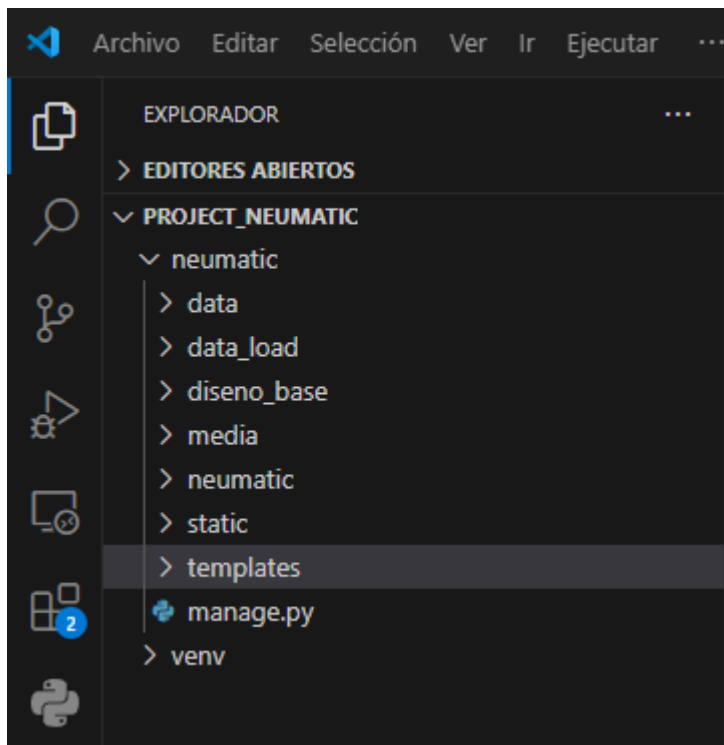
| Descripción de carpeta o archivo | Carpeta o archivo |
|--|--|
| Carpeta del proyecto | D:\PROJECT_NEUMATIC\neumatic |
| Carpeta de configuración del proyecto | D:\PROJECT_NEUMATIC\neumatic\neumatic |
| Archivo de configuración del proyecto | D:\PROJECT_NEUMATIC\neumatic\settings.py |
| archivo de rutas del proyecto | D:\PROJECT_NEUMATIC\neumatic\urls.py |
| Script para la creación y administración de recursos y levantar el servidor para desarrollo del proyecto | D:\PROJECT_NEUMATIC\neumatic\manage.py |

5.5.2. Cambios en la estructura del proyecto

En el proyecto se van a crear un conjunto de carpetas, en las cuales será posible apoyar y controlar la funcionalidad de las aplicaciones. En las tablas siguientes mostraremos la funcionalidad y las rutas de los proyectos:

| Funcionalidad | Carpeta |
|--|--|
| Carpeta para las bases de datos en SQLite 3 | D:\PROJECT_NEUMATIC\neumatic\data |
| Carpeta donde están los algoritmos y estructuras de datos portables que permiten migrar datos a los modelos | D:\PROJECT_NEUMATIC\neumatic\data_load |
| Carpeta donde se ubican los archivos que definen los diseños Bootstrap o Tailwind | D:\PROJECT_NEUMATIC\neumatic\diseno_base |
| Carpeta para almacenar y organizar los archivos tipo media que puede subir el usuario. Es recomendable en fase de desarrollo. En producción, se recomienda usar un repositorio, como Amazon S3, Google Drive | D:\PROJECT_NEUMATIC\neumatic\media |
| Carpeta donde se almacenan los archivos estáticos del proyecto | D:\PROJECT_NEUMATIC\neumatic\static |
| | D:\PROJECT_NEUMATIC\neumatic\static\assets |
| Carpeta donde se almacenan los archivos estáticos assets del proyecto | D:\PROJECT_NEUMATIC\neumatic\static\css |
| Carpeta donde se almacenan los archivos estáticos tipo imágenes del proyecto | D:\PROJECT_NEUMATIC\neumatic\static\img |
| Carpeta donde se almacenan los archivos JavaScript del proyecto | D:\PROJECT_NEUMATIC\neumatic\static\js |
| Carpeta donde se almacenan las plantillas de uso general del proyecto | D:\PROJECT_NEUMATIC\neumatic\templates |

Al aplicar los cambios, visualizaremos la siguiente estructura de carpetas:



6. Aplicaciones del Proyecto Neumatic

En Django, el **proyecto** es la estructura general que contiene la configuración y los recursos necesarios para que funcione un sitio web o aplicación web completa. Dentro del proyecto, puedes tener una o más **aplicaciones**.

Conceptos clave:

- **Proyecto:**
 - Es el contenedor principal que define la configuración global del sitio web, como la base de datos, las URLs principales, y las configuraciones de seguridad.
 - Tiene archivos como `settings.py` (configuraciones), `urls.py` (enrutamiento global), y `wsgi.py` o `asgi.py` (para desplegar el proyecto).
- **Aplicación:**
 - Es un módulo independiente dentro del proyecto, que gestiona una funcionalidad específica. Por ejemplo, podrías tener una aplicación para los catálogos, otra para ventas, otra para compras, otra para usuarios, etc.
 - Cada aplicación tiene sus propios modelos, vistas, formularios, y rutas (URLs) que manejan una parte específica del sistema.

Relación:

- Un **proyecto** puede tener **múltiples aplicaciones**, y cada aplicación puede ser reutilizada en otros proyectos.
- Las aplicaciones colaboran bajo el mismo proyecto para construir un sistema completo, compartiendo configuraciones globales y, a veces, datos.

En resumen, el **proyecto** es el contenedor global que agrupa una o más **aplicaciones**, y cada aplicación maneja una funcionalidad o característica específica dentro de ese proyecto.

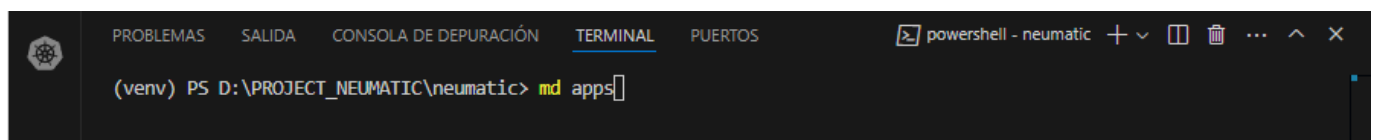
6.1. Definición de las aplicaciones del Proyecto

| Aplicación | Descripción |
|------------|--|
| maestros | Aplicación que permite gestionar las tablas base y catálogos, comunes a todo el proyecto |
| ventas | Aplicación que permite gestionar el módulo ventas del proyecto |
| compras | Aplicación que permite gestionar el módulo compras del proyecto |
| caja | Aplicación que permite gestionar el módulo caja del proyecto |
| inventario | Aplicación que permite gestionar el módulo inventario del proyecto |
| opciones | Aplicación que permite gestionar el opciones generales del proyecto |
| informes | Aplicación que permite generar los reportes y estadísticas del proyecto |
| usuarios | Aplicación que permite gestionar los usuarios del proyecto |

6.2. Aplicaciones base del proyecto

Luego de crear el proyecto y haber realizado los cambios a su estructura inicial, debemos crear nuestras aplicaciones básicas. Para un mayor orden creamos una carpeta contenedora de aplicaciones, en la siguiente ruta:

D:\PROJECT_NEUMATIC\neumatic\apps



Dentro de la carpeta contenedora se crearán las siguientes aplicaciones básicas

| Aplicación | Descripción de la aplicación | Ubicación |
|-----------------|--|--|
| maestros | Aplicación que permite gestionar las tablas base y catálogos, comunes a todo el proyecto | D:\PROJECT_NEUMATIC\neumatic\apps\maestros |
| <i>usuarios</i> | Aplicación que permite gestionar los usuarios del proyecto | D:\PROJECT_NEUMATIC\neumatic\apps\usuarios |

6.3. Creación de la Aplicación maestros

Para crear la primera aplicación, llamada maestros, nos trasladamos a la carpeta apps.

```

(venv) PS D:\PROJECT_NEUMATIC\neumatic> cd apps
(venv) PS D:\PROJECT_NEUMATIC\neumatic\apps>

```

Para crear la aplicación maestros, en el prompt de MS-DOS debe ejecutar la siguiente instrucción: **django-admin startapp maestros**

```

(venv) PS D:\PROJECT_NEUMATIC\neumatic> cd apps
(venv) PS D:\PROJECT_NEUMATIC\neumatic\apps> django-admin startapp maestros
(venv) PS D:\PROJECT_NEUMATIC\neumatic\apps>

```

6.4. Creación de la Aplicación usuarios

Para crear la aplicación usuarios, en el prompt de MS-DOS debe ejecutar la siguiente instrucción: **django-admin startapp usuarios**

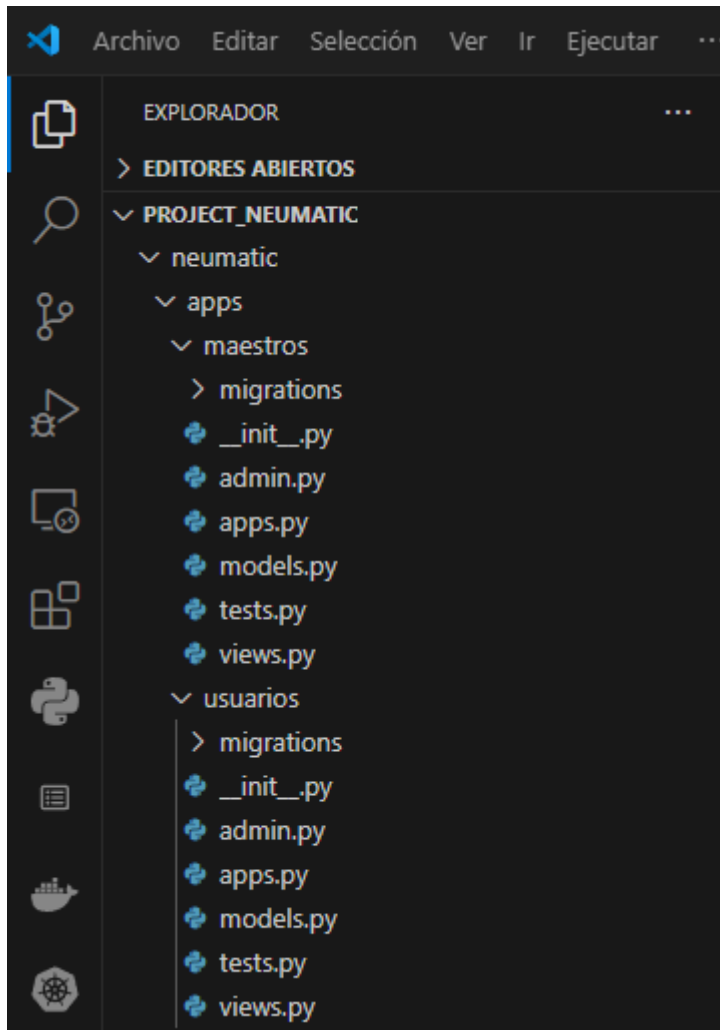
```

(venv) PS D:\PROJECT_NEUMATIC\neumatic\apps> django-admin startapp usuarios
(venv) PS D:\PROJECT_NEUMATIC\neumatic\apps>

```

6.5. Estructuras de la aplicación mestros y usuarios

Las estructuras iniciales de las aplicaciones maestros y usuarios son las siguientes:



Estas estructuras, nos permiten iniciar el desarrollo de las aplicaciones. Sin embargo, vamos a hacer cambios en ellas.

6.6. Cambios en la estructura de la aplicación maestros

Los cambios en la estructura de la aplicación maestros, serán aplicables de forma similar en las nuevas aplicaciones que se van creando, salvo en la aplicación usuarios que tiene una variante. Los cambios que se aplicarán en la estructura de la aplicación maestros los podemos observar en las siguientes tablas.

| Eliminar los archivos |
|--|
| D:\PROJECT_NEUMATIC\neumatic\apps\maestros\models.py |
| D:\PROJECT_NEUMATIC\neumatic\apps\maestros\views.py |

| Crear las carpetas |
|---|
| D:\PROJECT_NEUMATIC\neumatic\apps\maestros\forms |
| D:\PROJECT_NEUMATIC\neumatic\apps\maestros\models |
| D:\PROJECT_NEUMATIC\neumatic\apps\maestros\templates\maestros |
| D:\PROJECT_NEUMATIC\neumatic\apps\maestros\templatetags |
| D:\PROJECT_NEUMATIC\neumatic\apps\maestros\views |
| Crear el archivo |
| D:\PROJECT_NEUMATIC\neumatic\apps\maestros\urls.py |

6.7. Cambios en la estructura de la aplicación usuarios

| Eliminar los archivos |
|---|
| D:\PROJECT_NEUMATIC\neumatic\apps\usuarios\views.py |
| Crear las carpetas |
| D:\PROJECT_NEUMATIC\neumatic\apps\usuarios\forms |
| D:\PROJECT_NEUMATIC\neumatic\apps\usuarios\templates\usuarios |
| D:\PROJECT_NEUMATIC\neumatic\apps\usuarios\templatetags |
| D:\PROJECT_NEUMATIC\neumatic\apps\usuarios\views |
| Crear el archivo |
| D:\PROJECT_NEUMATIC\neumatic\apps\usuarios\urls.py |

6.8. Configuración de las aplicaciones

Luego de la crear la aplicaciones base: **maestros** y **usuarios**, es necesario configurarlas. La primera configuración debe realizarse en el **archivo settings.py** ubicado en la carpeta de configuración del Proyecto, que en nuestro caso es:

D:\PROJECT_NEUMATIC\neumatic\neumatic\settings.py

Dentro del archivo **settings.py** hay una sección donde se definen las aplicaciones del proyecto, donde debe agregar las instrucciones:

'apps.maestros', 'apps.usuarios',

Que finalmente mostrará los siguientes resultados:

```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'apps.maestros',
    'apps.usuarios',
]
```

Por otro lado, debemos revisar la configuración a nivel de las aplicaciones en sus respectivos archivos:

D:\PROJECT_NEUMATIC\neumatic\apps\maestros\apps.py

```
from django.apps import AppConfig

class UsuariosConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'apps.maestros'
```

D:\PROJECT_NEUMATIC\neumatic\apps\usuarios\apps.py

```
from django.apps import AppConfig

class UsuariosConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'apps.usuarios'
```

IMPORTANTE: cuando se crea una aplicación, Django configura por defecto, en los archivos respectivos apps.py, lo siguiente:

```
name = 'maestros'  
name = 'usuarios'
```

Sin colocar el prefijo apps. Pero es necesario corregir esa configuración inicial por:

```
name = 'apps.maestros'  
name = 'apps.usuarios'
```

7. Patrones Arquitectónicos

Un **patrón arquitectónico** es una solución general y reutilizable para un problema común en la arquitectura de software. Se trata de una **estructura de alto nivel** que define cómo se organiza y se relacionan los principales componentes de un sistema de software. Los patrones arquitectónicos proporcionan una **guía abstracta** que ayuda a diseñar sistemas escalables, mantenibles y robustos, abordando aspectos como la separación de responsabilidades, la interacción entre componentes y la distribución de las funcionalidades.

7.1. Características clave de un patrón arquitectónico

- **Abstracción de alto nivel:** Define la organización global del sistema, estableciendo cómo se deben estructurar los módulos, subsistemas o componentes.
- **Solución probada:** Es un enfoque basado en prácticas que han demostrado ser efectivas en varios proyectos y contextos.
- **Modularidad:** Fomenta la separación clara de responsabilidades entre componentes, lo que facilita la escalabilidad y mantenibilidad del sistema.
- **Flexibilidad:** Permite la adaptación a distintos tipos de aplicaciones y requisitos, sin imponer una implementación rígida.

Ejemplos de patrones arquitectónicos:

- **MVC (Model-View-Controller):** Divide una aplicación en tres componentes principales: el modelo (gestión de datos), la vista (interfaz de usuario) y el controlador (lógica de control). Es común en aplicaciones web.
- **MTV (Model-Template-View):** Divide una aplicación en tres componentes principales: el modelo (gestión de datos), el template (interfaz de usuario) y la vista (lógica de control). **Es utilizado por el framework Django.**

- **Microservicios:** Organiza un sistema en una colección de servicios pequeños y autónomos que se comunican a través de interfaces bien definidas. Cada servicio tiene una responsabilidad específica.
- **Arquitectura en capas (n-tier):** Divide el sistema en capas funcionales (como presentación, lógica de negocio y acceso a datos), donde cada capa tiene una responsabilidad clara y se comunica con las capas adyacentes.
- **Event-Driven:** Los componentes del sistema se comunican a través de eventos, lo que permite que los componentes se desacoplen y respondan de manera más flexible y reactiva.
- **Arquitectura hexagonal:** Busca desacoplar las reglas de negocio de los detalles externos, permitiendo que el sistema se adapte fácilmente a nuevos tipos de interfaces de usuario, bases de datos u otras dependencias.

7.2. El patrón arquitectónico Model-Template-View

El patrón arquitectónico que utiliza Django se llama **Model-Template-View (MTV)**, que es muy similar al patrón **Model-View-Controller (MVC)** pero con una pequeña diferencia en la organización de los componentes.

7.2.1. Model-Template-View (MTV) en Django

Es el patrón arquitectónico que Django implementa para organizar el código de las aplicaciones web, donde se divide la aplicación en tres componentes principales:

- **Modelo (Model):** Gestiona la lógica de acceso a datos y define la estructura de los datos (normalmente almacenados en bases de datos). Cada modelo en Django representa una tabla en la base de datos y contiene la lógica de negocio y las interacciones con los datos.
- **Vista (View):** Es la capa que controla la lógica de la aplicación, recibe las solicitudes del usuario, interactúa con los modelos y selecciona la plantilla adecuada para mostrar los datos. En Django, las vistas son funciones o clases que procesan las solicitudes HTTP y devuelven una respuesta.
- **Plantilla (Template):** Define la estructura de presentación de los datos. Las plantillas en Django son archivos HTML con lógica embebida (usualmente usando el lenguaje de plantillas de Django) para mostrar los datos proporcionados por la vista de manera dinámica.

7.2.2. Diferencia entre MTV y MVC:

- En **MVC**, el **Controlador (Controller)** es un componente explícito que recibe la solicitud del usuario y determina cómo manejarla. En **MVT**, este papel lo realiza la **Vista (View)** de

Django, mientras que el **Template** maneja la parte de la presentación, separando claramente la lógica del negocio y la presentación de la interfaz de usuario.

7.2.3. Ventajas del patrón MTV en Django:

- **Separación de responsabilidades:** Cada componente tiene una responsabilidad clara, lo que facilita el mantenimiento y la escalabilidad.
- **Reutilización de plantillas:** Permite usar las mismas plantillas para diferentes vistas, lo que hace más eficiente el manejo de la interfaz de usuario.
- **Modularidad:** El patrón MVT permite dividir el sistema en módulos que pueden desarrollarse y probarse de manera independiente.

Este patrón es fundamental en el desarrollo con Django, facilitando la organización del código y promoviendo buenas prácticas en el diseño de aplicaciones web escalables.

8. Organización de los componentes en Django

Django, basado en el patrón arquitectónico **Model-Template-View (MTV)**, organiza los diferentes componentes de un proyecto web de manera estructurada y modular, lo que facilita el desarrollo y mantenimiento. A continuación, te explico cómo se organizan los modelos, formularios, vistas, rutas, plantillas y administración en un proyecto Django bajo el patrón MTV.

8.1. Modelos (Model)

Los modelos en Django representan las estructuras de datos y la lógica de negocio. Están mapeados directamente a tablas en la base de datos, y cada modelo define los campos y comportamientos de los datos. Los modelos son la base para interactuar con la base de datos (crear, leer, actualizar y eliminar registros).

- **Archivo:** `models.py` dentro de cada aplicación.
- **Responsabilidad:** Definir la estructura de los datos, relaciones entre tablas y reglas de validación.
- **Ejemplo:**

```
class Producto(models.Model):  
    nombre = models.CharField(max_length=100)  
    precio = models.DecimalField(max_digits=10, decimal_places=2)
```

8.2. Formularios

Los formularios en Django gestionan la interacción con los datos, tanto para capturar como para validar la entrada de datos de los usuarios. Django ofrece clases de formularios que pueden estar vinculadas directamente a los modelos o ser personalizados para diferentes necesidades.

- **Archivo:** `forms.py` dentro de cada aplicación.
- **Responsabilidad:** Capturar y validar los datos ingresados por los usuarios antes de almacenarlos en la base de datos o utilizarlos en otras partes de la lógica de la aplicación.
- **Ejemplo:**

```
from django import forms
from .models import Producto

class ProductoForm(forms.ModelForm):
    class Meta:
        model = Producto
        fields = ['nombre', 'precio']
```

8.3. Vistas (View)

Las vistas en Django son el núcleo de la lógica de la aplicación. Reciben las solicitudes HTTP de los usuarios, interactúan con los modelos para obtener o manipular los datos y deciden qué plantilla debe ser renderizada para devolver una respuesta al usuario.

- **Archivo:** `views.py` dentro de cada aplicación.
- **Responsabilidad:** Contienen la lógica de negocio, obtienen datos de los modelos, los envían a las plantillas, y gestionan las solicitudes y respuestas HTTP.
- **Tipos de vistas:** Funciones o vistas basadas en clases.
- **Ejemplo:**

```
from django.shortcuts import render
from .models import Producto

def listar_productos(request):
    productos = Producto.objects.all()
    return render(request, 'productos/lista.html', {'productos': productos})
```

8.4. Rutas (URL Routing)

Las rutas en Django definen cómo las solicitudes HTTP se dirigen a las vistas adecuadas. Utilizan expresiones regulares o patrones de URL para enlazar una URL con una vista específica.

- **Archivo:** `urls.py` dentro de cada aplicación y un archivo principal en el proyecto.
- **Responsabilidad:** Asociar URLs con las vistas correspondientes.
- **Ejemplo:**

```
from django.urls import path
from . import views

urlpatterns = [
    path('productos/', views.listar_productos, name='listar_productos'),
]
```

8.5. Plantillas (Template)

Las plantillas en Django definen la estructura de presentación de la aplicación. Utilizan el motor de plantillas de Django, que permite incluir lógica básica (como bucles y condicionales) para generar HTML dinámico a partir de los datos proporcionados por las vistas.

- **Archivo:** Archivos HTML ubicados en un directorio llamado `templates` dentro de la aplicación o en un directorio central.
- **Responsabilidad:** Mostrar los datos al usuario de una manera organizada y estéticamente agradable.
- **Ejemplo** (`templates/productos/lista.html`):

```
<h1>Lista de Productos</h1>
<ul>
    {% for producto in productos %}
        <li>{{ producto.nombre }} - {{ producto.precio }}</li>
    {% endfor %}
</ul>
```

8.6. Administración

Django proporciona un sistema de administración **out-of-the-box**, lo que permite gestionar fácilmente los modelos sin necesidad de crear interfaces personalizadas. Este sistema de administración permite a los administradores del sitio crear, leer, actualizar y eliminar registros de las tablas definidas por los modelos.

- **Archivo:** `admin.py` dentro de cada aplicación.
- **Responsabilidad:** Registrar los modelos para que sean gestionados desde la interfaz de administración de Django.
- **Ejemplo:**

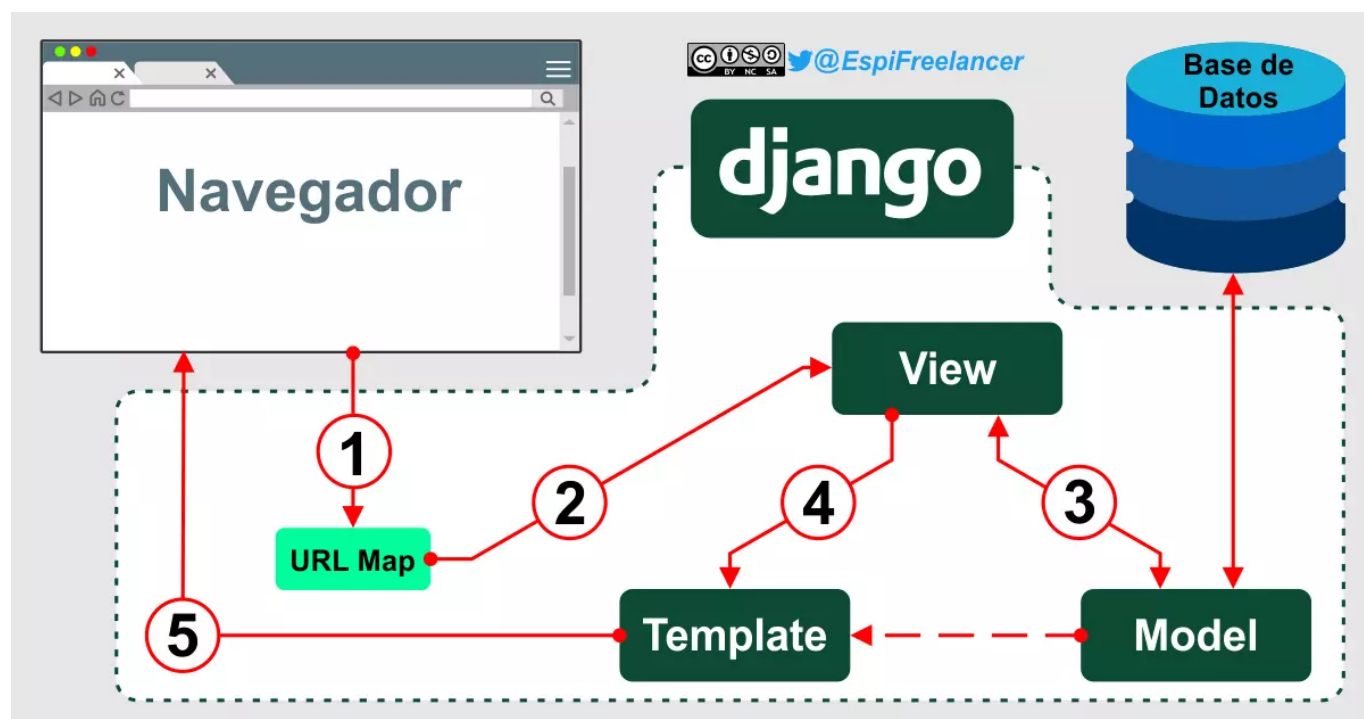
```
from django.contrib import admin
from .models import Producto

@admin.register(Producto)
class ProductoAdmin(admin.ModelAdmin):
    list_display = ('nombre', 'precio')
```

8.7. Organización del proyecto Django

1. **Modelos:** Definen la estructura y la lógica de negocio (ubicados en `models.py`).
2. **Formularios:** Capturan y validan los datos del usuario (ubicados en `forms.py`).
3. **Vistas:** Contienen la lógica de negocio que interactúa con los modelos y selecciona la plantilla adecuada (ubicados en `views.py`).
4. **Rutas:** Enlazan las URLs del proyecto con las vistas correspondientes (ubicados en `urls.py`).
5. **Plantillas:** Definen la interfaz de usuario y cómo se muestran los datos (archivos HTML en la carpeta `templates`).
6. **Administración:** Un panel de control autogenerado para gestionar modelos desde una interfaz gráfica.

8.8. Flujo de trabajo en Django bajo el patrón MTV



1. Un usuario solicita una URL específica.
2. **Rutas** dirigen la solicitud a una **vista**.

3. La **vista** interactúa con los **modelos** para obtener los datos necesarios.
4. Los datos son pasados a una **plantilla**, que renderiza el contenido y lo devuelve al usuario como una respuesta HTML.

En resumen, este enfoque modular y basado en MVT de Django facilita la separación de responsabilidades y la escalabilidad del sistema. Además, con la interfaz de administración incluida, se puede gestionar el contenido y los datos de manera eficiente sin necesidad de desarrollar un sistema de administración personalizado.

9. Modelos iniciales en la aplicación maestros

Una de las etapas fundamentales en el desarrollo del Proyecto en Django, es la creación de los modelos. Las estructuras de los modelos, nacen en la etapa de análisis. En ese sentido, empezaremos con la creación de los modelos de la aplicación maestros.

9.1. Creación del modelo ModeloBaseGenerico

El modelo ModeloBaseGenerico es una clase abstracta, que nos servirá de base para crear los modelos de las tablas y catálogos de la aplicación maestros. Es decir, los modelos que luego crearemos, heredarán atributos (campos) y métodos (procedimientos) del modelo ModeloBaseGenerico. El modelo ModeloBaseGenerico, se creará dentro del archivo:

D:\PROJECT_NEUMATIC\neumatic\apps\maestros\models\base_gen_models.py

El contenido será el siguiente:


```

# D:\PROJECT_NEUMATIC\neumatic\apps\maestros\models\base_gen_models.py
from django.db import models

import socket
from datetime import datetime

# from django.http import request
# from django.contrib.auth.models import User

# Importa el modelo User Personalizado
from apps.usuarios.models import User

class ModeloBaseGenerico(models.Model):
    usuario = models.CharField(max_length=20, null=True, blank=True)
    estacion = models.CharField(max_length=20, null=True, blank=True)
    fcontrol = models.CharField(max_length=22, null=True, blank=True)

    class Meta:
        abstract = True

    def save(self, *args, **kwargs):
        # Obtiene el usuario actual
        if not self.usuario:
            # Reemplaza con el valor correcto
            try:
                # Intenta obtener el usuario actual si ha iniciado sesión
                self.usuario = User.objects.get(
                    username="nombre_de_usuario_actual")

            except User.DoesNotExist:
                # Si el usuario no existe o no ha iniciado sesión,
                # puedes establecer un valor predeterminado
                self.usuario = "UserPrueba"

        # Obtiene el nombre del equipo (estación) en Windows
        if not self.estacion:
            self.estacion = socket.gethostname()

        # Obtiene la fecha y hora actual en el formato deseado
        if not self.fcontrol:
            # Reemplaza con el formato deseado
            self.fcontrol = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

        super(ModeloBaseGenerico, self).save(*args, **kwargs)

```

10. Modelo de la aplicación usuarios

Un punto fundamental en el desarrollo del proyecto, es la seguridad. En ese sentido, uno de los puntos base para iniciar el trabajo de la seguridad del proyecto, es la gestión de usuarios y el control de sesiones. Por otro lado, si queremos ampliar los datos del registro de usuarios, es muy recomendable crear un modelo que hereda de la clase `AbstractUser`, lo que permite extender el modelo de usuario predeterminado de Django sin reescribirlo completamente. El modelo base ya incluye campos como `username`, `password`, etc. Para ello vamos a modificar el archivo:

```
# D:\PROJECT_NEUMATIC\neumatic\apps\usuarios\models.py
from django.db import models
from django.contrib.auth.models import AbstractUser
from django.db.models.signals import post_save
from django.dispatch import receiver

class User(AbstractUser):
    email = models.EmailField("Correo electrónico")
    email_alt = models.EmailField("Correo alternativo", max_length=50,
                                   null=True, blank=True)
    telefono = models.CharField("Teléfono", max_length=9,
                                 null=True, blank=True)

    iniciales = models.CharField("Iniciales", max_length=3,
                                   null=True, blank=True)
    jerarquia = models.CharField("Jerarquía", max_length=1,
                                   null=True, blank=True)
    vendedor = models.BooleanField(default=False, null=True, blank=True)

    # [id_vendedor] [int] NOT NULL,
    # [id_sucursal] [int] NOT NULL,
    # [punto_venta] [int] NOT NULL,

#-- Al crear un nuevo usuario este quede activo por defecto.
@receiver(post_save, sender=User)
def set_user_active(sender, instance, created, **kwargs):
    if created and not instance.is_active:
        instance.is_active = True
        instance.save()
```

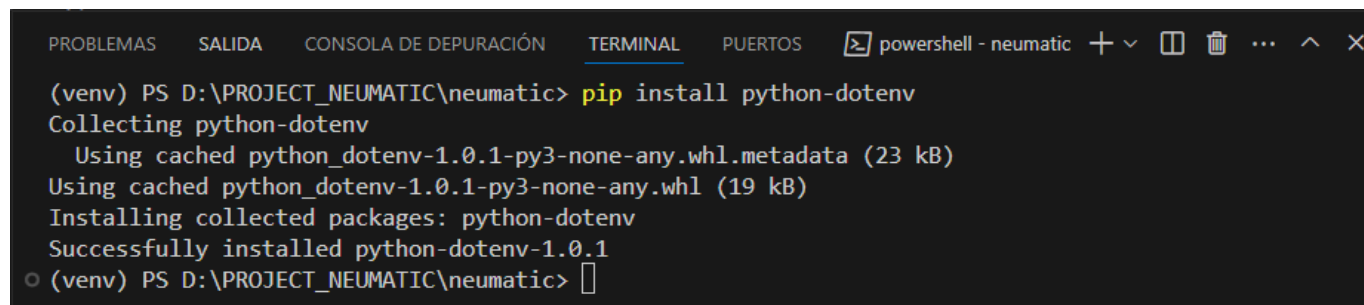
11. Configuración del Proyecto

11.1. Instalación de python-dotenv

El paquete `python-dotenv` es una herramienta útil para cargar variables de entorno desde un archivo `.env` en tu entorno de ejecución. Esto es especialmente útil para almacenar

configuraciones sensibles, como claves de API, configuraciones de base de datos, y otros valores que no quieres incluir directamente en tu código fuente por razones de seguridad o portabilidad.

Para instalar `python-dotenv`, utiliza el siguiente comando: `pip install python-dotenv`

A screenshot of a PowerShell terminal window. The title bar shows 'powershell - neumatic'. The terminal content shows the command 'pip install python-dotenv' being executed in a virtual environment. The output indicates that the package is collected from a cache and installed successfully.

```
(venv) PS D:\PROJECT_NEUMATIC\neumatic> pip install python-dotenv
Collecting python-dotenv
  Using cached python_dotenv-1.0.1-py3-none-any.whl.metadata (23 kB)
  Using cached python_dotenv-1.0.1-py3-none-any.whl (19 kB)
Installing collected packages: python-dotenv
Successfully installed python-dotenv-1.0.1
○ (venv) PS D:\PROJECT_NEUMATIC\neumatic> 
```

¿Qué es un archivo `.env` ?

Un archivo `.env` es un archivo de texto plano que contiene pares clave-valor que definen variables de entorno. Por ejemplo:

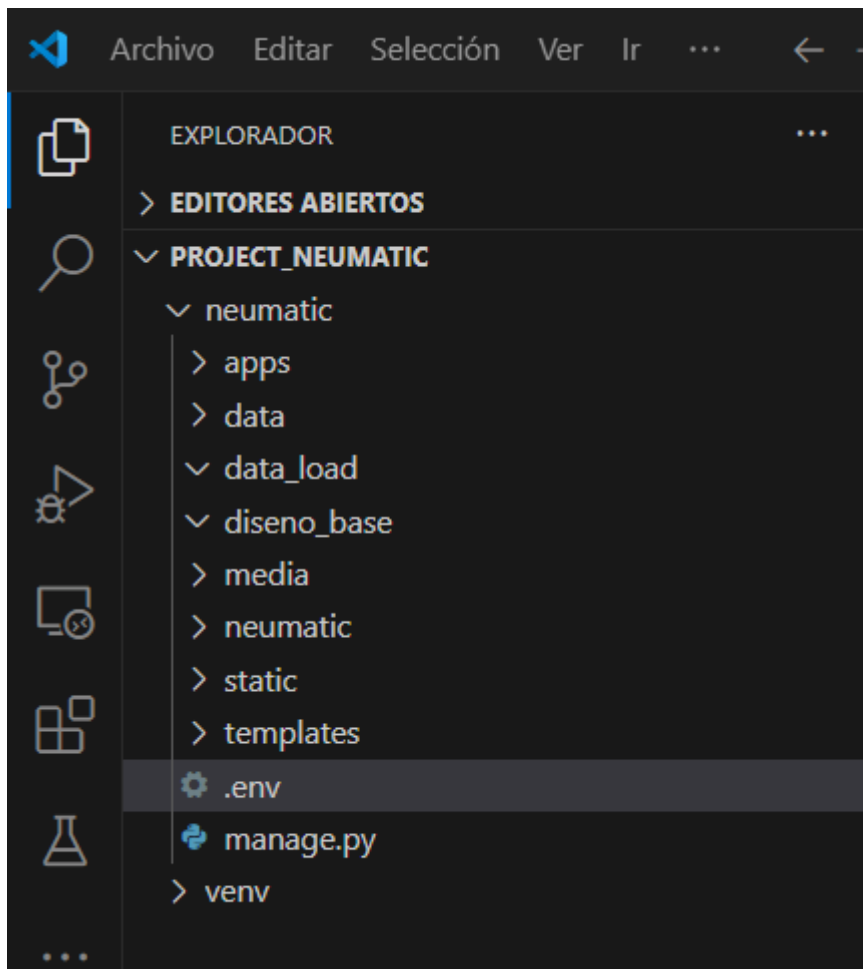
```
SECRET_KEY=your_secret_key
DEBUG=True
DATABASE_URL=postgres://user:password@localhost:5432/mydb
```

11.2. Crear en archivo `.env`

En la carpeta del proyecto, creamos el archivo `.env`:

`D:\PROJECT_NEUMATIC\neumatic\.env`

La estructura general del Proyecto Neumatic, quedará del siguiente modo:



En el archivo .env, escriba el siguiente contenido y luego guarde los cambios:

```
SECRET_KEY = 'django-insecure-==#&zbor^0xfg(!q6^7&-73syfn%0vitw587*+g*t@3(^@x$ut#'
```

11.3. Configurar el archivo settings.py

Sobre escriba el siguiente contenido sobre el archivo settings.py del Proyecto:

```

"""
Django settings for neumatic project.

Generated by 'django-admin startproject' using Django 5.1.1.

For more information on this file, see
https://docs.djangoproject.com/en/5.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.1/ref/settings/
"""

from pathlib import Path
from dotenv import load_dotenv
from os import path, getenv

#-- Cargar las variables de entorno del archivo .env
load_dotenv()

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = getenv('SECRET_KEY')

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'apps.maestros',
    'apps.usuarios',
]

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 = 'neumatic.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [
            # Carpeta templates a nivel del proyecto
            path.join(BASE_DIR, 'templates')
        ],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    ],
]

WSGI_APPLICATION = 'neumatic.wsgi.application'

# Database
# https://docs.djangoproject.com/en/5.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': path.join(BASE_DIR, 'data', 'db_neumatic.db'),
    }
}

# Password validation
# https://docs.djangoproject.com/en/5.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },

```

```
{
    'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
},
{
    'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
},
{
    'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
},
]
```

```
# Internationalization
# https://docs.djangoproject.com/en/5.1/topics/i18n/
```

```
LANGUAGE_CODE = 'es-ar'
```

```
TIME_ZONE = 'America/Argentina/Buenos_Aires'
```

```
USE_I18N = True
```

```
USE_TZ = True
```

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/
```

```
STATIC_URL = '/static/'
STATICFILES_DIRS = (path.join(BASE_DIR, 'static'),)
```

```
# Archivos media
MEDIA_URL = '/media/'
MEDIA_ROOT = path.join(BASE_DIR, 'media')
```

```
# Default primary key field type
# https://docs.djangoproject.com/en/5.1/ref/settings/#default-auto-field
```

```
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

```
# URL de redireccionamiento de la vista para iniciar sesión.
```

```
LOGIN_URL = '/usuarios/sesion/iniciar/'
```

```
# URL de redireccionamiento una vez logueado.
```

```
LOGIN_REDIRECT_URL = '/'
```

```
# URL de redireccionamiento al cerrar sesión.
```

```
LOGOUT_REDIRECT_URL = '/usuarios/sesion/iniciar/'
```

```
# Para evitar el "secuestro" de la sesión de usuario por JavaScript desde el front.
```

```
SESSION_COOKIE_HTTPONLY = True
```

```
# La sesión del usuario se cierra al cerrar el navegador.
```

```
SESSION_EXPIRE_AT_BROWSER_CLOSE = True
```

```
# Modelo Usuario personalizado.  
AUTH_USER_MODEL = 'usuarios.User'
```

El archivo `settings.py` modificado de tu proyecto Django contiene varias configuraciones importantes para la correcta operación del proyecto. A continuación, te explico cada sección relevante y los ajustes que se han realizado.

Configuración básica del entorno

```
from dotenv import load_dotenv  
from os import path, getenv
```

Aquí se utiliza el paquete `python-dotenv` para cargar variables de entorno almacenadas en un archivo `.env`. Esto es útil para gestionar configuraciones sensibles como la clave secreta sin exponerla en el código fuente.

Directorio base

```
BASE_DIR = Path(__file__).resolve().parent.parent
```

Define la ruta base del proyecto, la cual es usada para resolver rutas relativas en el proyecto.

Clave secreta y depuración

```
SECRET_KEY = getenv('SECRET_KEY')  
DEBUG = True
```

La clave secreta se obtiene del archivo `.env` para mayor seguridad.

`DEBUG = True` debe estar en `False` en producción para evitar la exposición de errores detallados.

Aplicaciones instaladas


```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'apps.maestros',  
    'apps.usuarios',  
]
```

Aquí se incluyen tanto aplicaciones del framework Django como dos aplicaciones personalizadas: `maestros` y `usuarios`.

Middleware

```
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',  
]
```

Define una lista de "middleware", que son capas de procesamiento entre las solicitudes del usuario y las respuestas de Django. Incluye medidas de seguridad como `SecurityMiddleware`, protección CSRF y protección contra ataques de "clickjacking".

Configuración de plantillas

```
'DIRS': [  
    path.join(BASE_DIR, 'templates')  
],
```

Configura la carpeta de plantillas a nivel del proyecto, permitiendo que las plantillas HTML se ubiquen en `BASE_DIR/templates`.

Base de datos

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': path.join(BASE_DIR, 'data', 'db_neumatic.db'),
    }
}
```

Usa una base de datos SQLite. La base de datos se almacenará en una carpeta `data` dentro del directorio base del proyecto.

Validación de contraseñas

```
AUTH_PASSWORD_VALIDATORS = [
    # Validadores de contraseñas
]
```

Define validadores que aseguran que las contraseñas de los usuarios cumplan con ciertos criterios de seguridad, como longitud mínima o evitar contraseñas comunes.

Internacionalización

```
LANGUAGE_CODE = 'es-ar'
TIME_ZONE = 'America/Argentina/Buenos_Aires'
```

El idioma se ha configurado como "es-ar" (español de Argentina), y la zona horaria está ajustada a Buenos Aires.

Archivos estáticos y media

```
STATICFILES_DIRS = (path.join(BASE_DIR, 'static'),)
MEDIA_URL = '/media/'
MEDIA_ROOT = path.join(BASE_DIR, 'media')
```

Los archivos estáticos se almacenan en la carpeta `static`, y los archivos subidos por el usuario se almacenan en `media`.

Configuración de inicio y cierre de sesión

```
LOGIN_URL = '/usuarios/sesion/iniciar/'
LOGIN_REDIRECT_URL = '/'
LOGOUT_REDIRECT_URL = '/usuarios/sesion/iniciar/'
```

Define las URLs para las vistas de inicio y cierre de sesión, y el redireccionamiento después de una sesión iniciada o cerrada.

Configuración de seguridad de sesión

```
SESSION_COOKIE_HTTPONLY = True
SESSION_EXPIRE_AT_BROWSER_CLOSE = True
```

Se asegura que la cookie de sesión no sea accesible desde JavaScript (`HTTPOnly`), lo que mitiga ciertos ataques XSS.

La sesión se configura para expirar cuando el navegador se cierra, mejorando la seguridad en dispositivos compartidos.

Modelo de usuario personalizado

```
AUTH_USER_MODEL = 'usuarios.User'
```

Define un modelo de usuario personalizado, `User` , ubicado en la aplicación `usuarios` . Esto permite la personalización de los atributos del modelo de usuario, como agregar campos adicionales o modificar su comportamiento predeterminado.

11.4. Configurar los modelos en la aplicación maestros

En Django, los modelos de una aplicación normalmente se definen dentro del archivo `models.py` . Django los detecta automáticamente cuando ejecutas comandos como `makemigrations` y `migrate` . Sin embargo, si decides organizar los modelos en varios archivos dentro de una carpeta llamada `models` , Django no detectará esos archivos adicionales automáticamente. Esto se debe a que, por defecto, Django solo busca un archivo llamado `models.py` en la raíz de cada aplicación.

Para solucionar este problema, es necesario indicarle a Django dónde encontrar esos modelos adicionales. Una forma de hacerlo es a través del método `ready()` en el archivo `apps.py` de la aplicación, que se ejecuta cuando la aplicación se carga por primera vez. En este método, se pueden importar explícitamente los archivos donde están definidos los modelos. Al hacerlo, Django incluirá esos modelos en el sistema de migraciones.

En nuestro caso, hemos organizado varios archivos, que contienen definiciones de modelos:

```
D:\PROJECT_NEUMATIC\neumatic\apps\maestros\models\  
├─ base_gen_models.py  
├─ base_models.py  
├─ cliente_models.py  
├─ empresa_models.py  
├─ numero_models.py  
├─ parametro_models.py  
├─ producto_models.py  
├─ proveedor_models.py  
├─ sucursal_models.py  
└─ vendedor_models.py
```

Para que Django detecte los archivos de modelos, utilizamos el método `ready()`:

```
# D:\PROJECT_NEUMATIC\neumatic\apps\maestros\apps.py  
from django.apps import AppConfig  
  
class MaestrosConfig(AppConfig):  
    default_auto_field = 'django.db.models.BigAutoField'  
    name = 'apps.maestros'  
  
    def ready(self):  
        # Importamos explícitamente cada archivo de modelos  
        import apps.maestros.models.base_gen_models  
        import apps.maestros.models.base_models  
        import apps.maestros.models.cliente_models  
        import apps.maestros.models.empresa_models  
        import apps.maestros.models.numero_models  
        import apps.maestros.models.parametro_models  
        import apps.maestros.models.producto_models  
        import apps.maestros.models.proveedor_models  
        import apps.maestros.models.sucursal_models  
        import apps.maestros.models.vendedor_models
```

Explicación de la solución

- **Importación en el método `ready()`** : Al sobrescribir el método `ready()` en el archivo `apps.py` de la aplicación `maestros`, importas explícitamente cada archivo donde has definido modelos. Cuando Django inicia y carga la aplicación `maestros`, ejecuta el método `ready()`, lo que permite que Django registre esos modelos.
- **Registro automático de modelos**: Al importar los archivos dentro del método `ready()`, todos los modelos definidos en esos archivos son registrados por Django, lo que asegura que los comandos como `makemigrations` y `migrate` los detecten.

Flujo de trabajo

- **Crear modelos en archivos separados:** Al tener los modelos distribuidos en varios archivos, tienes una organización más limpia y modular de tu código, lo que facilita el mantenimiento y escalabilidad del proyecto.
- **Importar modelos en `apps.py`:** Al importar los archivos de modelos en `apps.py`, te aseguras de que Django los registre.
- **Ejecutar las migraciones:** Después de hacer los cambios en los modelos o al definir nuevos modelos, puedes ejecutar los comandos de migración.

11.5. Aplicar las migraciones

En Django, las **migraciones** son el mecanismo utilizado para aplicar cambios en la estructura de la base de datos a lo largo del ciclo de desarrollo del proyecto. Esto es importante porque, a medida que creas o modificas los modelos en tu código, Django necesita reflejar esos cambios en la base de datos.

Hay dos comandos clave para gestionar las migraciones en Django:

makemigrations

El comando `makemigrations` crea un conjunto de archivos de migración que describen los cambios que se deben realizar en la base de datos en función de los cambios realizados en los modelos.

Funcionalidad:

- Detecta cambios en los modelos (agregar, eliminar o modificar campos/tablas) y genera un archivo de migración para aplicarlos.
- Los archivos de migración se almacenan en la carpeta `migrations` de cada aplicación.
- **No realiza cambios en la base de datos directamente.** Solo prepara las instrucciones para los cambios.

migrate

El comando `migrate` aplica las migraciones pendientes a la base de datos. Es decir, ejecuta las instrucciones almacenadas en los archivos de migración y realiza los cambios necesarios en la estructura de la base de datos.

Funcionalidad:

- Aplica las migraciones pendientes en la base de datos.
- Si se han creado tablas nuevas, se generan en la base de datos. Si se han modificado, las tablas existentes se actualizan.
- El estado de las migraciones aplicadas se guarda en una tabla especial llamada `django_migrations`, que Django usa para saber qué migraciones ya se han ejecutado y cuáles no.

Aplicando makemigrations al Proyecto

```

PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS

• (venv) PS D:\PROJECT_NEUMATIC\neumatic> python manage.py makemigrations
Migrations for 'maestros':
  apps\maestros\migrations\0001_initial.py
    + Create model Actividad
    + Create model ComprobanteCompra
    + Create model ComprobanteVenta
    + Create model Localidad
    + Create model Moneda
    + Create model Operario
    + Create model ProductoDeposito
    + Create model ProductoEstado
    + Create model ProductoFamilia
    + Create model ProductoModelo
    + Create model Provincia
    + Create model TipoDocumentoIdentidad
    + Create model TipoIva
    + Create model TipoPercepcionIb
    + Create model TipoRetencionIb
    + Create model Empresa
    + Create model Parametro
    + Create model ProductoMarca
    + Create model ProductoMinimo
    + Create model Producto
    + Create model ProductoStock
    + Add field id_provincia to localidad
    + Create model Sucursal
    + Add field id_sucursal to productodeposito
    + Create model Numero
    + Create model Proveedor
    + Create model Vendedor
    + Create model Cliente
Migrations for 'usuarios':
  apps\usuarios\migrations\0001_initial.py

```

Aplicando migrate al Proyecto

```
● (venv) PS D:\PROJECT_NEUMATIC\neumatic> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, maestros, sessions, usuarios
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0001_initial... 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 usuarios.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 maestros.0001_initial... OK
  Applying sessions.0001_initial... OK
○ (venv) PS D:\PROJECT_NEUMATIC\neumatic> █
```

12. Registro de los modelos en el administrador

12.1. Registro de los modelos de la aplicación maestros

12.1.1. Registro Básico

```

from django.contrib import admin

from .models.cliente_models import Cliente
from .models.empresa_models import Empresa
from .models.numero_models import Numero
from .models.parametro_models import Parametro
from .models.producto_models import Producto
from .models.proveedor_models import Proveedor
from .models.sucursal_models import Sucursal
from .models.vendedor_models import Vendedor
from .models.base_models import *

# Registramos los modelos independientes
admin.site.register(Cliente)
admin.site.register(Empresa)
admin.site.register(Numero)
admin.site.register(Parametro)
admin.site.register(Producto)
admin.site.register(Proveedor)
admin.site.register(Sucursal)
admin.site.register(Vendedor)

# Registramos los modelos base
admin.site.register(Actividad)
admin.site.register(ComprobanteCompra)
admin.site.register(ComprobanteVenta)
admin.site.register(Localidad)
admin.site.register(Moneda)
admin.site.register(Operario)
admin.site.register(ProductoDeposito)
admin.site.register(ProductoEstado)
admin.site.register(ProductoFamilia)
admin.site.register(ProductoMarca)
admin.site.register(ProductoModelo)
admin.site.register(ProductoStock)
admin.site.register(Provincia)
admin.site.register(TipoDocumentoIdentidad)
admin.site.register(TipoIva)
admin.site.register(TipoPercepcionIb)
admin.site.register(TipoRetencionIb)

```

12.1.1. Registro Personalizado

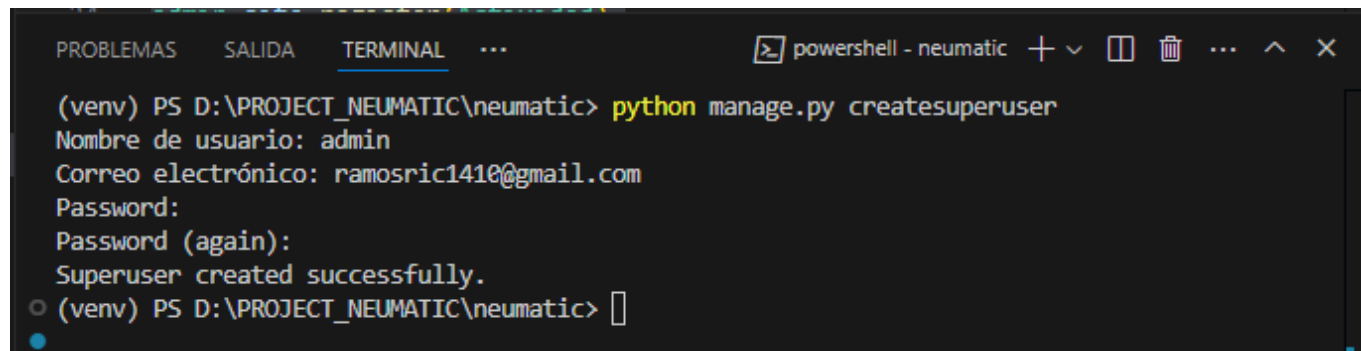
13. Creación del superusuario del Proyecto

El **superusuario** en Django es una cuenta de usuario con permisos administrativos completos sobre la aplicación web, lo que le permite gestionar todos los modelos, usuarios y otras

configuraciones a través del panel de administración. El superusuario tiene permisos totales, como crear, leer, actualizar y eliminar registros (CRUD) de cualquier modelo registrado en la administración de Django.

Creación del superusuario

Para crear un superusuario en Django, sigue estos pasos:



```
PROBLEMAS  SALIDA  TERMINAL  ...  powershell - neumatic  + v  [ ]  [ ]  ...  ^  X

(venv) PS D:\PROJECT_NEUMATIC\neumatic> python manage.py createsuperuser
Nombre de usuario: admin
Correo electrónico: ramosric1410@gmail.com
Password:
Password (again):
Superuser created successfully.
(venv) PS D:\PROJECT_NEUMATIC\neumatic> [ ]
```

La información solicitada es:

- **Nombre de usuario:** Este es el identificador único que usarás para iniciar sesión.
- **Correo electrónico** (dependiendo de la configuración del modelo de usuario personalizado).
- **Contraseña:** Necesitarás ingresar una contraseña segura dos veces para confirmarla.

El sistema genera un usuario con todos los permisos administrativos.

Funcionalidades del superusuario:

- **Acceso completo al panel de administración:** El superusuario puede acceder a todas las secciones del panel administrativo en `/admin/`, incluidas las secciones de los modelos que hayas registrado en el archivo `admin.py`.
- **Gestión de usuarios y permisos:** El superusuario puede crear, modificar y eliminar usuarios normales, además de asignarles permisos específicos.
- **Acceso y modificación de todos los modelos:** El superusuario puede ver y modificar todas las entradas de los modelos registrados, incluso aquellos modelos que no son visibles para usuarios con permisos más limitados.
- **Funcionalidades CRUD completas:** El superusuario puede crear, leer, actualizar y eliminar cualquier registro de cualquier modelo en la base de datos a través de la interfaz de administración.

Configuración del superusuario:

- Django proporciona un modelo de usuario por defecto (`User`) que incluye campos como `username` , `email` , `password` , `is_staff` , `is_superuser` , y `is_active` .
- Cuando un superusuario es creado, los campos `is_staff` y `is_superuser` se configuran como `True` , lo que le permite gestionar todo desde el panel de administración.

Diferencia entre `is_staff` e `is_superuser` :

- `is_staff` : Define si el usuario tiene acceso al panel de administración. Los usuarios con `is_staff=True` pueden acceder al panel administrativo, pero pueden tener permisos limitados dependiendo de cómo se configuren.
- `is_superuser` : Define si el usuario tiene permisos administrativos completos. Los usuarios con `is_superuser=True` tienen todos los permisos y pueden hacer cualquier cosa en el panel de administración sin restricciones.

Recomendaciones de seguridad:

- **Contraseña segura:** Asegúrate de que la contraseña del superusuario sea segura y compleja.
- **MFA (Autenticación multifactor):** Es recomendable habilitar autenticación multifactor para el superusuario en aplicaciones de producción.
- **Uso limitado del superusuario:** Es una buena práctica crear otros usuarios administradores con permisos limitados, para evitar que el superusuario sea utilizado innecesariamente, reduciendo el riesgo de errores o abusos.

Resumen:

- El superusuario tiene acceso completo a la administración de Django.
- Puedes crear un superusuario ejecutando `python manage.py createsuperuser` .
- El superusuario puede gestionar todos los modelos, usuarios y permisos en el panel administrativo de Django.
- Usa el superusuario con cuidado y sigue las mejores prácticas de seguridad.

14. El servidor de desarrollo de Django

El **servidor de desarrollo de Django** es una herramienta integrada que se usa para probar y desarrollar aplicaciones de manera rápida sin la necesidad de configurar servidores web complejos, como Apache o Nginx. Este servidor es útil para detectar errores durante el

desarrollo, pero no está diseñado para entornos de producción debido a su baja eficiencia y falta de seguridad para manejar tráfico real.

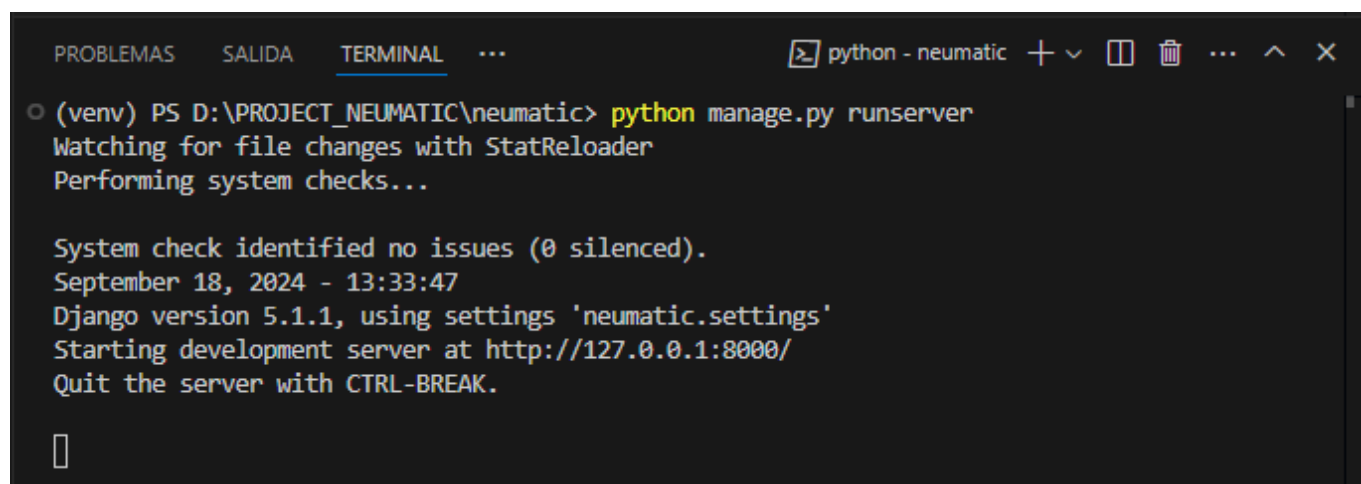
14.1. Funcionalidad del servidor de desarrollo

El servidor de desarrollo te permite:

1. **Visualizar y probar tu aplicación localmente** sin necesidad de configuraciones adicionales.
2. **Detección automática de cambios:** Cada vez que realizas cambios en tu código, el servidor se reinicia automáticamente, lo que facilita las pruebas rápidas.
3. **Debugging:** Si tienes el modo `DEBUG = True` en el archivo `settings.py`, el servidor muestra errores detallados en el navegador, lo que ayuda a identificar problemas durante el desarrollo.

14.2. Comando para ejecutar el servidor de desarrollo

El comando más básico para ejecutar el servidor de desarrollo es:



```
PROBLEMAS  SALIDA  TERMINAL  ...  python - neumatic  +  -  [ ]  [ ]  ...  ^  X

○ (venv) PS D:\PROJECT_NEUMATIC\neumatic> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

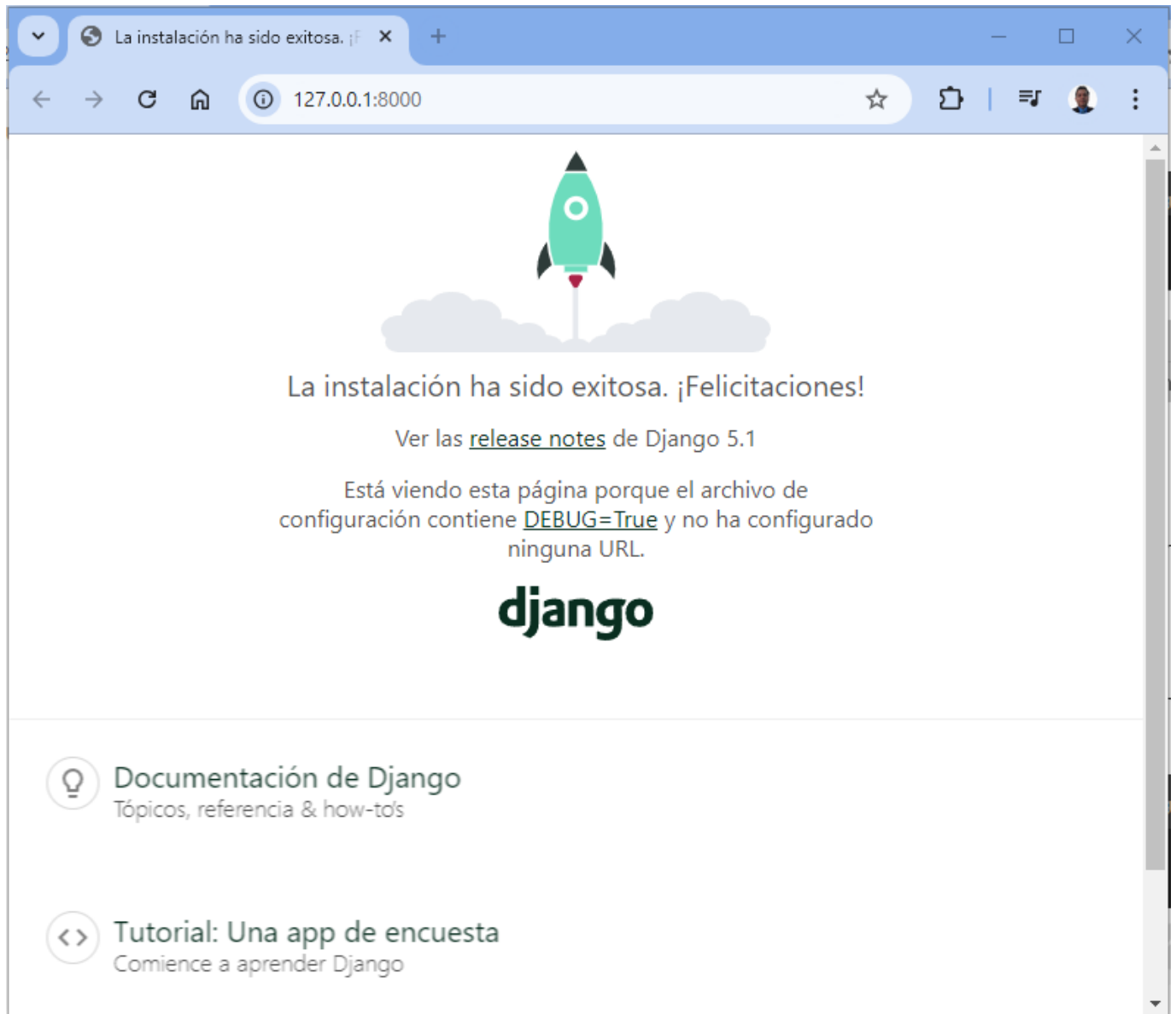
System check identified no issues (0 silenced).
September 18, 2024 - 13:33:47
Django version 5.1.1, using settings 'neumatic.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[ ]
```

De manera predeterminada, esto levantará el servidor en la dirección `127.0.0.1` (localhost) en el **puerto 8000**, lo que significa que puedes acceder a la aplicación en tu navegador con la dirección `http://127.0.0.1:8000/` o `http://localhost:8000/`.

Una forma práctica y directa, para acceder a la aplicación desde el Terminal de VSC, es presionar la tecla CTRL y luego dar click a <http://127.0.0.1:8000/>

Se mostrará la siguiente página en el navegador:



Cambiar la dirección IP y el puerto del servidor de desarrollo

Puedes modificar tanto la IP como el puerto del servidor usando argumentos opcionales al comando `runserver`.

Cambiar el puerto

Si quieres cambiar el puerto (por ejemplo, al puerto 8080), puedes hacerlo pasando el puerto como argumento:

```
PROBLEMAS  SALIDA  TERMINAL  ...  python - neumatic  + v  [ ]  [ ]  ...  ^  X

(venv) PS D:\PROJECT_NEUMATIC\neumatic> python manage.py runserver 8085
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
September 18, 2024 - 13:48:17
Django version 5.1.1, using settings 'neumatic.settings'
Starting development server at http://127.0.0.1:8085/
Quit the server with CTRL-BREAK.

[ ]
```

Esto ejecutará el servidor en `127.0.0.1:8085` (o `http://localhost:8085/`).

Cambiar la dirección IP

Por defecto, Django solo permite conexiones desde `127.0.0.1` (localhost). Si deseas acceder a la aplicación desde otras máquinas en la misma red local o desde tu propia máquina usando la IP, necesitas especificar una dirección IP diferente (por ejemplo, `0.0.0.0` para que escuche en todas las interfaces de red).

```
(venv) PS D:\PROJECT_NEUMATIC\neumatic> python manage.py runserver 0.0.0.0:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
September 18, 2024 - 14:13:14
Django version 5.1.1, using settings 'neumatic.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CTRL-BREAK.

[ ]
```

Esto hace que el servidor escuche en **todas las interfaces de red** en el puerto 8000. Ahora, puedes acceder a la aplicación desde cualquier dispositivo en la red local usando la dirección IP de tu máquina.

Cambiar tanto la dirección IP como el puerto

Si deseas cambiar tanto la dirección IP como el puerto al mismo tiempo, puedes hacerlo así:

```
⊗ (venv) PS D:\PROJECT_NEUMATIC\neumatic> python manage.py runserver 192.168.18.6:8080
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
Error: [WinError 10013] Intento de acceso a un socket no permitido por sus permisos de acces
o
(venv) PS D:\PROJECT_NEUMATIC\neumatic> █
```

El error `WinError 10013: Intento de acceso a un socket no permitido por sus permisos de acceso` generalmente indica que hay una restricción de permisos en el uso del puerto o la dirección IP en tu sistema.

Consideraciones de seguridad

- **No usar en producción:** El servidor de desarrollo está optimizado para la facilidad de desarrollo, no para rendimiento o seguridad. No está recomendado utilizar este servidor para entornos de producción.
- **Exponer a redes externas:** Si configuras el servidor para escuchar en todas las interfaces de red (`0.0.0.0`), asegúrate de que solo se use para desarrollo local y en un entorno controlado. Cualquier persona en la misma red podría acceder a tu aplicación, lo que puede ser riesgoso si no estás detrás de un firewall adecuado.

Resumen

- El servidor de desarrollo de Django te permite probar y depurar tu aplicación localmente.
- Usa `python manage.py runserver` para levantar el servidor en la IP `127.0.0.1` y el puerto `8000` de manera predeterminada.
- Puedes cambiar la dirección IP y el puerto pasando argumentos adicionales como `python manage.py runserver 0.0.0.0:8080`.
- Este servidor no es apto para producción; solo debe usarse en entornos de desarrollo.

15. Personalizar la página de inicio del Proyecto

15.1. La página de inicio por defecto de Django

Hasta el momento hemos realizado una serie de cambios en la estructura del Proyecto. Ahora vamos a levantar el servidor de desarrollo, para luego ejecutar la aplicación en el navegador de nuestra estación de trabajo.

Cuando ejecutamos en el terminal de VSC, la instrucción:

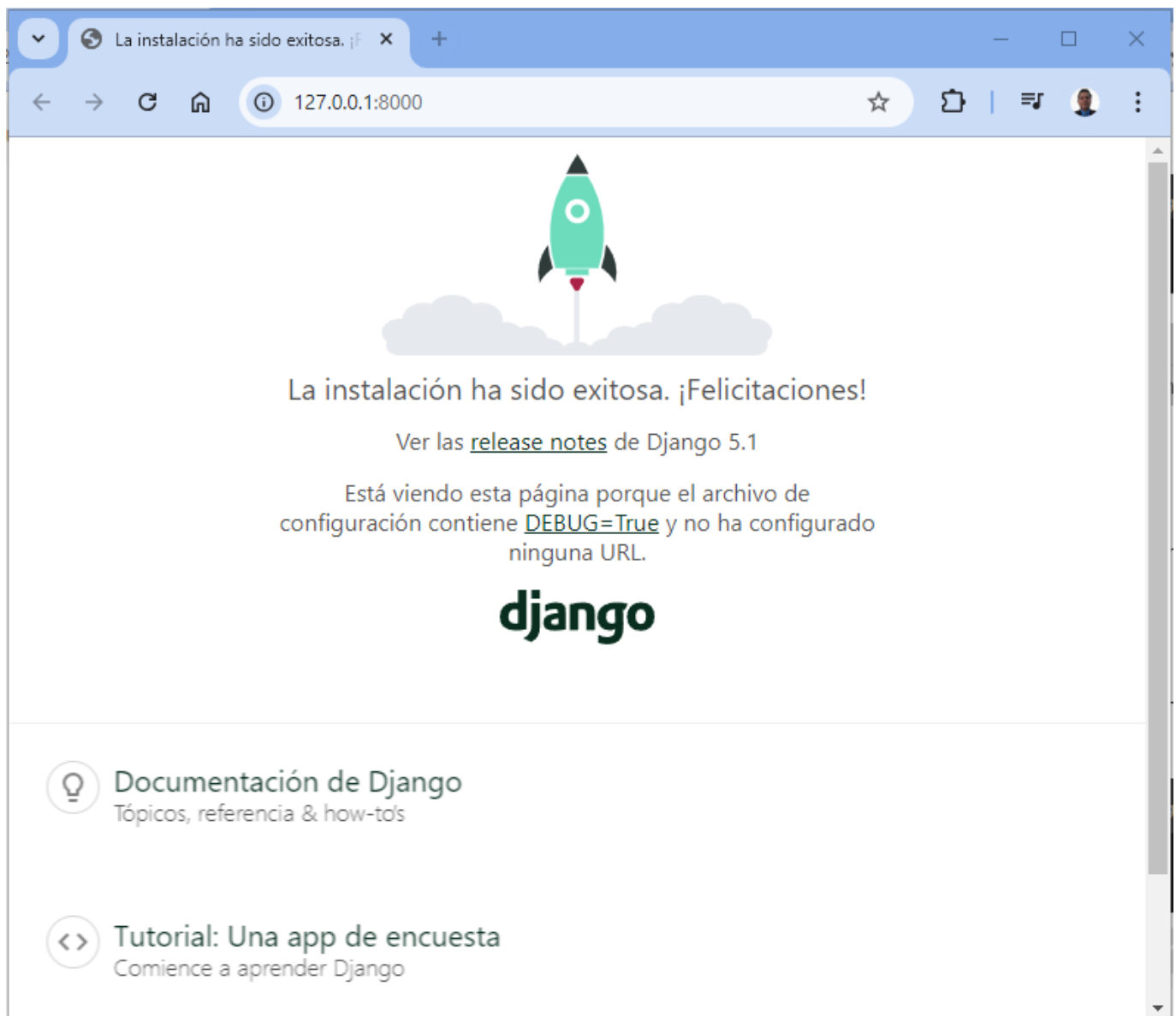
```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  python - neumatic  + v  [ ]  [X]  ..

o (venv) PS D:\PROJECT_NEUMATIC\neumatic> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
September 18, 2024 - 14:49:10
Django version 5.1.1, using settings 'neumatic.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[ ]
```

Y luego accedemos a la aplicación, presionando la tecla CTRL y damos click a <http://127.0.0.1:8000/> observaremos en el navegador, la página de inicio, por defecto de Django:



Sin embargo, podemos configurar el acceso al Proyecto, a través de un Login y personalizar la página de inicio con un menú y un dashboard.

15.2. Implementación de la página de inicio con el menú general y dashboard

15.2.1. Creación de la vista, la ruta y las plantillas para la página de inicio del Proyecto

En la carpeta de configuración del Proyecto vamos a crear la vista y la ruta asociada.

Creación de la vista

En la carpeta de configuración del Proyecto: D:\PROJECT_NEUMATIC\neumatic\neumatic, creamos la vista dentro del archivo D:\PROJECT_NEUMATIC\neumatic\neumatic\views.py y le colocamos el siguiente contenido:

```
from django.shortcuts import render, redirect
from django.utils import timezone

def home_view(request):
    if request.user.is_authenticated:
        fecha_actual = timezone.now()
        return render(request, 'home.html', {'fecha': fecha_actual})
    else:
        return redirect('iniciar_sesion')
```

Creación de la ruta

En la carpeta de configuración del Proyecto: D:\PROJECT_NEUMATIC\neumatic\neumatic, modificamos el archivo

D:\PROJECT_NEUMATIC\neumatic\neumatic\urls.py, insertado la ruta:


```

"""
URL configuration for neumatic project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/5.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from .views import home_view

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', home_view, name='home'),
]

```

Creación de la plantilla asociada a la vista y ruta (home.html)

La creación de la plantilla home.html y sus plantillas asociadas, se alojarán en la carpeta templates de todo el Proyecto, cuya ruta es la siguiente:

D:\SIG_PROJECTS\SIGCOERP\templates

El contenido de la plantilla home.html es:

```

{% extends 'base.html' %}
{% load static %}

<!-- Block Title -->
{% block title %}
    Inicio
{% endblock title %}

{% block header %}
    {% include 'top_nav.html' %}
{% endblock header %}

<!-- Block Main -->
{% block main %}
    {% block sidebar %}
        {% if user.is_authenticated %}
            {% include 'sidebar.html' %}
        {% endif %}
    {% endblock sidebar %}

    {% block maincomponent %}

        <!-- Main principalcomponent Start -->
        {% block principalcomponent %}
            {% include 'dashboard.html' %}
        {% endblock principalcomponent %}
        <!-- Main principalcomponent End -->
    {% endblock maincomponent %}
{% endblock main %}

<!-- Block Fotter -->
{% block footer %}
{% endblock footer %}

```

En adelante, cree las plantillas que se indican, en la misma carpeta donde creó la plantilla home.html. Ahora cree la plantilla base.html, cuyo contenido es:

```

{% load static %}
<!doctype html>
<html lang="es">
    <head>
        <meta charset="utf-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no" />
        <meta name="description" content="" />
        <meta name="author" content="" />

        <title>
            {% block title %}
            {% endblock title %}
        </title>

        <link rel="icon" href="{% static 'img/favicon.png' %}" type="image/x-icon">

        <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhY6hW+ALEwIH"
crossorigin="anonymous">
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.11.3/font/bootstrap-icons.min.css">
        <link rel="stylesheet" href="https://unpkg.com/bs-
brain@2.0.4/components/logins/login-9/assets/css/login-9.css">

        <style>
            {% block style %}
            {% endblock style %}
        </style>

        <link rel="stylesheet" href="{% static 'css/styles.css' %}">
        <link rel="stylesheet" href="{% static 'css/botones.css' %}">
        <script src="{% static 'js/control_list.js' %}"></script>
    </head>

    <body class="sb-nav-fixed">

        {% block header %}
        {% endblock header %}

        <div id="layoutSidenav">
            {% block main %}

                {% block sidebar %}
                {% endblock sidebar %}

                {% block maincomponent %}
                {% block principalcomponent %}

```

```

        {% endblock principalcomponent %}
    {% endblock maincomponent %}

    {% endblock main %}
</div>

{% block modals %}
{% endblock modals %}

{% block footer %}
{% endblock footer %}

{% block script %}
{% endblock script %}

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.8.0/Chart.min.js"
crossorigin="anonymous"></script>
        <script src="https://cdn.jsdelivr.net/npm/simple-
datatables@7.1.2/dist/umd/simple-datatables.min.js" crossorigin="anonymous"></script>
        <script src="https://use.fontawesome.com/releases/v6.3.0/js/all.js"
crossorigin="anonymous"></script>
        <script src="{% static 'assets/demo/chart-area-demo.js' %}"></script>
        <script src="{% static 'assets/demo/chart-bar-demo.js' %}"></script>
        <script src="{% static 'js/datatables-simple-demo.js' %}"></script>
        <script src="{% static 'js/scripts.js' %}" defer></script>
    </body>
</html>

```

Cree la plantilla top_nav.html, cuyo contenido es:

```

<nav class="sb-topnav navbar navbar-expand navbar-dark bg-primary">
  <!-- Navbar Brand-->
  <a class="navbar-brand ps-3" href="{% url 'home' %}">SigcoERP</a>

  <!-- Sidebar Toggle-->
  <button class="btn btn-link btn-sm order-1 order-lg-0 me-4 me-lg-0"
id="sidebarToggle" href="#!"><i class="fas fa-bars"></i></button>

  <!-- Se agrega la clase flex-fill para que "empuje" los siguientes elementos al
final del div -->
  <!-- <div class="col-auto align-self-center text-white"> -->
  <div class="col-auto align-self-center text-white flex-fill">
    {{ fecha.date|date:"d/m/Y" }}
  </div>

  <!-- Navbar Search-->
  <!-- <form class="d-none d-md-inline-block form-inline ms-auto me-0 me-md-3 my-2
my-md-0">
    <div class="input-group">
      <input class="form-control" type="text" placeholder="Buscar..." aria-
label="Search for..." aria-describedby="btnNavbarSearch" />
      <button class="btn btn-primary" id="btnNavbarSearch" type="button"><i
class="fas fa-search"></i></button>
    </div>
  </form> -->

  <div class="col-auto align-self-center text-white">
    {% if user.is_authenticated %}
      {% if user.first_name%}
        {{user.first_name}} {{user.last_name}}
      {% else %}
        {{ user }}
      {% endif %}
    {% endif %}
  </div>

  <!-- Navbar-->
  <ul class="navbar-nav ms-auto ms-md-0 me-3 me-lg-4">
    <li class="nav-item dropdown">
      <a class="nav-link dropdown-toggle" id="navbarDropdown" href="#"
role="button" data-bs-toggle="dropdown" aria-expanded="false"><i class="fas fa-user
fa-fw"></i></a>
      <ul class="dropdown-menu dropdown-menu-end" aria-
labelledby="navbarDropdown">
        <li><a class="dropdown-item" href="#!">Configuración</a></li>
        <li><a class="dropdown-item" href="#!">Registro de Actividad</a></li>
        {% if user.is_staff or user.is_superuser %}
          <li><a class="dropdown-item" href="{% url 'grupo_listar'
%}">Grupos</a></li>
          <li><a class="dropdown-item" href="{% url 'usuario_listar'

```

```

    %}">Usuarios</a></li>
        {% endif %}
    </li>
        <hr class="dropdown-divider" />
    </li>
    <li><a class="dropdown-item" href="{% url 'cerrar_sesion' %}">Cerrar
Sesión</a></li>
    </ul>
</li>
</ul>

</nav>

```

Cree la plantilla sidebar.html, cuyo contenido es:

```

<div id="layoutSidenav_nav">
    {% comment %} <nav class="sb-sidenav accordion sb-sidenav-light"
id="sidenavAccordion"> {% endcomment %}
    <nav class="sb-sidenav accordion sb-sidenav-blue" id="sidenavAccordion">
        <div class="sb-sidenav-menu">
            <div class="nav">
                <div class="sb-sidenav-menu-heading">Core</div>

                <a class="nav-link" href="{% url 'home' %}">
                    <div class="sb-nav-link-icon"><i class="fas fa-tachometer-alt">
</i></div>
                    Panel
                </a>

                <div class="sb-sidenav-menu-heading">Ventas</div>

                <!-- Opciones del Módulo Ventas Start -->

                <!-- Archivos Start -->
                <a class="nav-link collapsed" href="#" data-bs-toggle="collapse" data-
bs-target="#collapseFiles" aria-expanded="false" aria-controls="collapseFiles">
                    <div class="sb-nav-link-icon"><i class="fas fa-book-open"></i>
</div>
                    Archivos
                    <div class="sb-sidenav-collapse-arrow"><i class="fas fa-angle-
down"></i></div>
                </a>

                <div class="collapse" id="collapseFiles" aria-labelledby="headingTwo"
data-bs-parent="#sidenavAccordion">
                    <nav class="sb-sidenav-menu-nested nav accordion"
id="sidenavAccordionPages">

                        <a class="nav-link collapsed" href="#" data-bs-
toggle="collapse" data-bs-target="#pagesCollapseAuth" aria-expanded="false" aria-
controls="pagesCollapseAuth">
                            Catálogos
                            <div class="sb-sidenav-collapse-arrow"><i class="fas fa-
angle-down"></i></div>
                        </a>
                        <div class="collapse" id="pagesCollapseAuth" aria-
labelledby="headingOne" data-bs-parent="#sidenavAccordionPages">
                            <nav class="sb-sidenav-menu-nested nav">
                                <a class="nav-link" href="#">Clientes</a>
                                <a class="nav-link" href="#">Proveedores</a>
                                <a class="nav-link" href="#">Productos</a>
                                <a class="nav-link" href="#">Vendedores</a>
                                <a class="nav-link" href="#">Empresa</a>
                                <a class="nav-link" href="#">Sucursales</a>
                                <a class="nav-link" href="#">Parámetros</a>

```

```

        <a class="nav-link" href="#">Números</a>

    </nav>
</div>

    <a class="nav-link collapsed" href="#" data-bs-
toggle="collapse" data-bs-target="#pagesCollapseError" aria-expanded="false" aria-
controls="pagesCollapseError">
        Tablas
        <div class="sb-sidenav-collapse-arrow"><i class="fas fa-
angle-down"></i></div>
    </a>
    <div class="collapse" id="pagesCollapseError" aria-
labelledby="headingOne" data-bs-parent="#sidenavAccordionPages">
        <nav class="sb-sidenav-menu-nested nav">
            <a class="nav-link" href="#">Actividad</a>
            <a class="nav-link" href="#">Prod. Depósito</a>
            <a class="nav-link" href="#">Prod. Familia</a>
            <a class="nav-link" href="#">Prod. Marca</a>
            <a class="nav-link" href="#">Prod. Modelo</a>
            <a class="nav-link" href="#">Prod. Mínimo</a>
            <a class="nav-link" href="#">Prod. Stock</a>
            <a class="nav-link" href="#">Prod. Estado</a>
            <a class="nav-link" href="#">Comp. Venta</a>
            <a class="nav-link" href="#">Comp. Compra</a>
            <a class="nav-link" href="#">Moneda</a>
            <a class="nav-link" href="#">Provincia</a>
            <a class="nav-link" href="#">Localidad</a>
            <a class="nav-link" href="#">Tipo Documento</a>
            <a class="nav-link" href="#">Tipo Iva</a>
            <a class="nav-link" href="#">Tipo Percepción</a>
            <a class="nav-link" href="#">Tipo Retención</a>
            <a class="nav-link" href="#">Operario</a>
        </nav>
    </div>
</nav>
</div>
<!-- Archivos End -->

<!-- Procesos Start -->
    <a class="nav-link collapsed" href="#" data-bs-toggle="collapse" data-
bs-target="#collapseProcesses" aria-expanded="false" aria-
controls="collapseProcesses">
        <div class="sb-nav-link-icon"><i class="fas fa-book-open"></i>
    </div>
        Procesos
        <div class="sb-sidenav-collapse-arrow"><i class="fas fa-angle-
down"></i></div>
    </a>

    <div class="collapse" id="collapseProcesses" aria-

```



```

labelledby="headingTwo" data-bs-parent="#sidenavAccordion">
    <nav class="sb-sidenav-menu-nested nav accordion"
id="sidenavAccordionPages">

        <a class="nav-link collapsed" href="#" data-bs-
toggle="collapse" data-bs-target="#pagesCollapseAuth" aria-expanded="false" aria-
controls="pagesCollapseAuth">
            Ventas
            <div class="sb-sidenav-collapse-arrow"><i class="fas fa-
angle-down"></i></div>
        </a>
        <div class="collapse" id="pagesCollapseAuth" aria-
labelledby="headingOne" data-bs-parent="#sidenavAccordionPages">
            <nav class="sb-sidenav-menu-nested nav">
                <a class="nav-link" href="#">Doc. Venta</a>
                <a class="nav-link" href="#">Anular DV</a>
            </nav>
        </div>

        <a class="nav-link collapsed" href="#" data-bs-
toggle="collapse" data-bs-target="#pagesCollapseAuth" aria-expanded="false" aria-
controls="pagesCollapseAuth">
            Compras
            <div class="sb-sidenav-collapse-arrow"><i class="fas fa-
angle-down"></i></div>
        </a>
        <div class="collapse" id="pagesCollapseAuth" aria-
labelledby="headingOne" data-bs-parent="#sidenavAccordionPages">
            <nav class="sb-sidenav-menu-nested nav">
                <a class="nav-link" href="#">Doc. Compra</a>
                <a class="nav-link" href="#">Anular DC</a>
            </nav>
        </div>

    </nav>
</div>
<!-- Procesos End -->

<!-- Informes Start -->
<a class="nav-link collapsed" href="#" data-bs-toggle="collapse" data-
bs-target="#collapseReports" aria-expanded="false" aria-controls="collapseReports">
    <div class="sb-nav-link-icon"><i class="fas fa-book-open"></i>
</div>

    Informes
    <div class="sb-sidenav-collapse-arrow"><i class="fas fa-angle-
down"></i></div>
</a>
<!-- Informes End -->

<!-- Opciones Start -->

```

```

        <a class="nav-link collapsed" href="#" data-bs-toggle="collapse" data-
bs-target="#collapseOptions" aria-expanded="false" aria-controls="collapseOptions">
            <div class="sb-nav-link-icon"><i class="fas fa-book-open"></i>
</div>
            Opciones
            <div class="sb-sidenav-collapse-arrow"><i class="fas fa-angle-
down"></i></div>
        </a>
        <!-- Opciones End -->

        <!-- Opciones del Módulo Ventas Start -->

        <!-- Opciones de Estadísticas Start -->
        <div class="sb-sidenav-menu-heading">Estadísticas</div>

        <a class="nav-link" href="charts.html">
            <div class="sb-nav-link-icon"><i class="fas fa-chart-area"></i>
</div>
            Gráficos
        </a>
        <a class="nav-link" href="tables.html">
            <div class="sb-nav-link-icon"><i class="fas fa-table"></i></div>
            Tablas
        </a>
        <!-- Opciones de Estadísticas End -->
    </div>
</div>
<div class="sb-sidenav-footer">
    <div class="small">Logged in as:</div>
    Start Bootstrap
</div>
</nav>
</div>

```

Cree la plantilla dashboard.html, cuyo contenido es:

```

<div id="layoutSidenav_content">
  <main>
    <div class="container-fluid px-4">
      <h4 class="mt-4">Panel: Módulos e Indicadores de Gestión</h4>
      {% comment %} <ol class="breadcrumb mb-4">
        <li class="breadcrumb-item active">Módulos</li>
      </ol> {% endcomment %}

      {% comment %} Módulos Start{% endcomment %}
      <div class="row">
        <div class="col-xl-2 col-md-3">
          <div class="card bg-primary bg-opacity-25 text-dark mb-4">
            <div class="card-body">Archivo</div>
            <div class="card-footer d-flex align-items-center justify-
content-between">
              <a class="small text-blue stretched-link" href="#">Ver
Detalles</a>
              <div class="small text-dark"><i class="fas fa-angle-
right"></i></div>
            </div>
          </div>
        </div>

        <div class="col-xl-2 col-md-3">
          <div class="card bg-success bg-opacity-25 text-dark mb-4">
            <div class="card-body">Ventas</div>
            <div class="card-footer d-flex align-items-center justify-
content-between">
              <a class="small text-blue stretched-link" href="#">Ver
Detalles</a>
              <div class="small text-dark"><i class="fas fa-angle-
right"></i></div>
            </div>
          </div>
        </div>

        <div class="col-xl-2 col-md-3">
          <div class="card bg-primary bg-opacity-25 text-dark mb-4">
            <div class="card-body">Compras</div>
            <div class="card-footer d-flex align-items-center justify-
content-between">
              <a class="small text-blue stretched-link" href="#">Ver
Detalles</a>
              <div class="small text-dark"><i class="fas fa-angle-
right"></i></div>
            </div>
          </div>
        </div>

        <div class="col-xl-2 col-md-3">

```

```

        <div class="card bg-success bg-opacity-25 text-dark mb-4">
            <div class="card-body">Caja</div>
            <div class="card-footer d-flex align-items-center justify-
content-between">
                <a class="small text-blue stretched-link" href="#">Ver
Detalles</a>
                <div class="small text-dark"><i class="fas fa-angle-
right"></i></div>
            </div>
        </div>
    </div>
    <div class="col-xl-2 col-md-3">
        <div class="card bg-primary bg-opacity-25 text-dark mb-4">
            <div class="card-body">Inventario</div>
            <div class="card-footer d-flex align-items-center justify-
content-between">
                <a class="small text-blue stretched-link" href="#">Ver
Detalles</a>
                <div class="small text-dark"><i class="fas fa-angle-
right"></i></div>
            </div>
        </div>
    </div>
    <div class="col-xl-2 col-md-3">
        <div class="card bg-success bg-opacity-25 text-dark mb-4">
            <div class="card-body">Informes</div>
            <div class="card-footer d-flex align-items-center justify-
content-between">
                <a class="small text-blue stretched-link" href="#">Ver
Detalles</a>
                <div class="small text-dark"><i class="fas fa-angle-
right"></i></div>
            </div>
        </div>
    </div>
</div>

{% comment %} Módulos End{% endcomment %}

{% comment %} Gráficos Start{% endcomment %}
<div class="row">
    <div class="col-xl-6">
        <div class="card mb-4">
            <div class="card-header">
                <i class="fas fa-chart-area me-1"></i>
                Ventas del Mes
            </div>
            <div class="card-body"><canvas id="myAreaChart" width="100%"
height="40"></canvas></div>
        </div>
    </div>
</div>

```

```

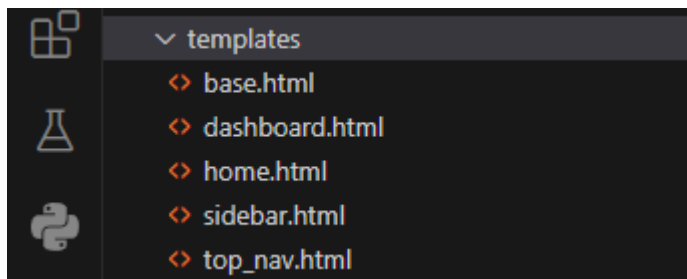
        <div class="col-xl-6">
            <div class="card mb-4">
                <div class="card-header">
                    <i class="fas fa-chart-bar me-1"></i>
                    Gastos Mensuales
                </div>
                <div class="card-body"><canvas id="myBarChart" width="100%"
height="40"></canvas></div>
            </div>
        </div>
    </div>
    {% comment %} Gráficos End{% endcomment %}

</div>
</main>
<footer class="py-4 bg-light mt-auto">
    <div class="container-fluid px-4">
        <div class="d-flex align-items-center justify-content-between small">
            <div class="text-muted">Copyright © Your Website 2023</div>
            <div>
                <a href="#">Privacy Policy</a>
                .
                <a href="#">Terms & Conditions</a>
            </div>
        </div>
    </div>
</footer>
</div>

```

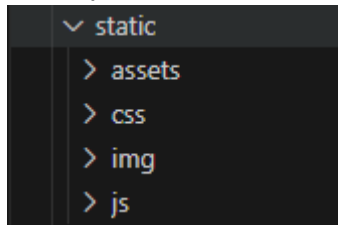
En la carpeta: D:\PROJECT_NEUMATIC\neumatic\templates

Finalmente debes tener la siguiente estructura de archivos:



15.2.2. Archivos de estilo, imágenes y scripts de JavaScript asociados a las plantillas, en la carpeta static

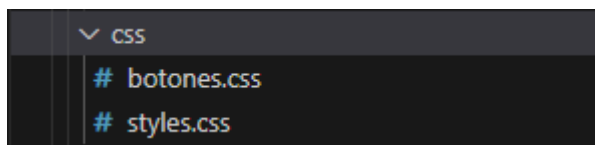
La carpeta `D:\PROJECT_NEUMATIC\neumatic\static` debe tener la siguiente estructura:



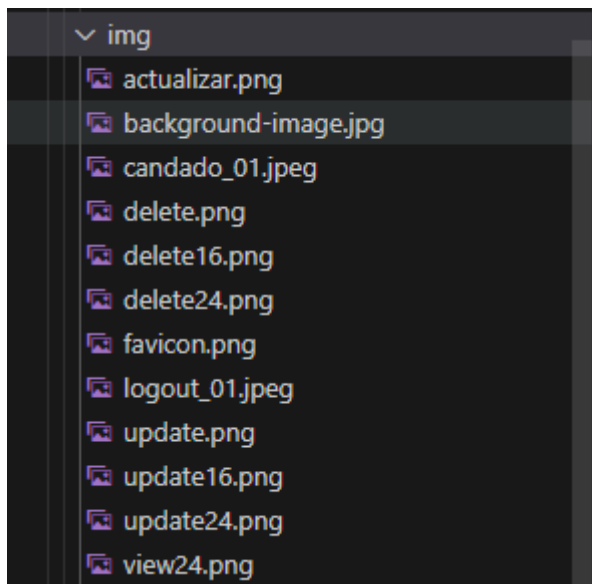
La estructura y contenido de la carpeta `assets`, debe obtenerse del repositorio de **GitHub**:



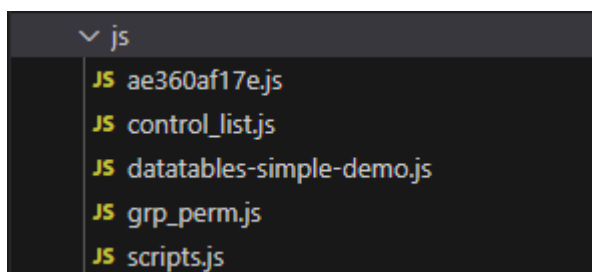
La estructura y contenido de la carpeta `css`, debe obtenerse del repositorio de **GitHub**:



La estructura y contenido de la carpeta `img`, debe obtenerse del repositorio de **GitHub**:



La estructura y contenido de la carpeta `js`, debe obtenerse del repositorio de **GitHub**:



16. CRUDs

Un **CRUD** es un acrónimo que se refiere a las cuatro operaciones básicas que se pueden realizar en una base de datos o en un sistema que maneja información:

- **C - Create** (Crear): Crear un nuevo registro o entrada en la base de datos.
- **R - Read** (Leer): Leer o consultar datos existentes en la base de datos.
- **U - Update** (Actualizar): Modificar los datos de un registro existente en la base de datos.
- **D - Delete** (Eliminar): Borrar un registro existente de la base de datos.

16.1 CRUD en el contexto de Django

Django es un framework web en Python que facilita enormemente la creación de aplicaciones CRUD. Cada operación del CRUD está asociada con una vista y un formulario que permiten interactuar con los datos almacenados en la base de datos a través de los **modelos**.

Aquí mostramos un ejemplo básico de cómo se relaciona el **CRUD** con Django:

Create (Crear):

- Se puede crear un nuevo objeto (registro) usando una vista de tipo `CreateView` o escribiendo una vista personalizada.
- Usamos un formulario para ingresar los datos y luego guardarlos en la base de datos.

Read (Leer):

- Se usa una vista de tipo `ListView` para mostrar una lista de todos los registros, o una vista de tipo `DetailView` para ver los detalles de un solo registro.

Update (Actualizar):

- Se usa una vista de tipo `UpdateView` o una vista personalizada para permitir la edición de un registro existente. Un formulario cargado con los datos actuales permite modificar el registro y luego guardarlo.

Delete (Eliminar):

- Se usa una vista de tipo `DeleteView` para eliminar un registro de la base de datos. Django generalmente requiere una confirmación antes de proceder con la eliminación.

16.2. Secuencia de CRUD en Django

Para implementar un CRUD básico, necesitas seguir los siguientes pasos:

- **Modelo:** Definir un modelo en `models.py` que representa la estructura de los datos que se almacenarán en la base de datos.
- **Formulario:** Crear un formulario que permita interactuar con los datos.
- **Vistas:** Configurar vistas para las operaciones de creación, lectura, actualización y eliminación.
- **URLs:** Definir las rutas en `urls.py` para que el sistema pueda acceder a las vistas correspondientes.
- **Plantillas:** Crear plantillas HTML que permitan al usuario interactuar con las vistas de manera visual.

16.3 Ventajas del uso de CRUD en Django

- **Automatización:** Django proporciona clases genéricas como `CreateView`, `ListView`, `UpdateView` y `DeleteView`, que facilitan la implementación rápida de CRUDs sin mucho código repetitivo.
- **Seguridad:** Django se encarga de muchos aspectos de seguridad automáticamente, como la protección contra inyecciones SQL y la validación de formularios.
- **Modularidad:** Los CRUDs en Django son fácilmente modificables y se pueden adaptar a diferentes necesidades.

Con esta base, podemos comenzar a construir CRUDs para cualquier modelo en Django, ajustando las vistas y plantillas a las necesidades específicas del proyecto.

17. CRUDs de la aplicación usuarios

Hemos aplicado las migraciones iniciales (`makemigrations` y `migrate`), pero antes de levantar el servidor de desarrollo para acceder a las aplicaciones de nuestro Proyecto, debemos desarrollar la funcionalidad básica de la aplicación usuarios.

17.1. El Modelo User


```
# D:\PROJECT_NEUMATIC\numatic\apps\usuarios\models.py
from django.db import models
from django.contrib.auth.models import AbstractUser
from django.db.models.signals import post_save
from django.dispatch import receiver

class User(AbstractUser):
    email = models.EmailField("Correo electrónico")
    email_alt = models.EmailField("Correo alternativo", max_length=50,
                                   null=True, blank=True)
    telefono = models.CharField("Teléfono", max_length=9,
                                 null=True, blank=True)

    iniciales = models.CharField("Iniciales", max_length=3,
                                   null=True, blank=True)
    jerarquia = models.CharField("Jerarquía", max_length=1,
                                   null=True, blank=True)
    vendedor = models.BooleanField(default=False, null=True, blank=True)

    #-- Al crear un nuevo usuario este quede activo por defecto.
    @receiver(post_save, sender=User)
    def set_user_active(sender, instance, created, **kwargs):
        if created and not instance.is_active:
            instance.is_active = True
            instance.save()
```

Este modelo de Django representa una clase personalizada de usuario que hereda de `AbstractUser`, el cual es un modelo base que ofrece Django para implementar usuarios con campos predefinidos como `username`, `password`, y `email`. El modelo se llama `User` y extiende el `AbstractUser` agregando nuevos campos y funcionalidad, además de un "signal" para asegurar que los nuevos usuarios se creen como activos por defecto. A continuación, se explica cada parte del código.

Extensión del modelo `User`

```
class User(AbstractUser):
    email = models.EmailField("Correo electrónico")
    email_alt = models.EmailField("Correo alternativo", max_length=50,
                                   null=True, blank=True)
    telefono = models.CharField("Teléfono", max_length=9,
                                 null=True, blank=True)
    iniciales = models.CharField("Iniciales", max_length=3,
                                  null=True, blank=True)
    jerarquia = models.CharField("Jerarquía", max_length=1,
                                  null=True, blank=True)
    vendedor = models.BooleanField(default=False, null=True, blank=True)
```

Explicación de los campos personalizados

- `email` : Un campo de tipo `EmailField` , que sobrescribe el campo `email` del modelo `AbstractUser` . Es un campo obligatorio para almacenar la dirección de correo electrónico principal del usuario.
- `email_alt` : Un campo de tipo `EmailField` opcional (permitiendo valores nulos y en blanco) para almacenar un correo electrónico alternativo del usuario. Tiene un límite de 50 caracteres.
- `telefono` : Un campo de tipo `CharField` para almacenar el número de teléfono del usuario. Es opcional y puede contener hasta 9 caracteres (probablemente el formato de teléfono sin el código de país).
- `iniciales` : Un campo de tipo `CharField` para almacenar las iniciales del usuario. Es opcional y puede contener hasta 3 caracteres, lo que podría ser útil para identificar rápidamente a un usuario mediante abreviaciones.
- `jerarquia` : Un campo de tipo `CharField` que parece almacenar un valor de jerarquía o nivel en la empresa, usando una sola letra. Es opcional y podría usarse para clasificar el rango o nivel del usuario en una estructura jerárquica.
- `vendedor` : Un campo de tipo `BooleanField` que indica si el usuario tiene el rol de vendedor. El valor por defecto es `False` , pero se puede cambiar a `True` . Es opcional, permitiendo `null` y `blank` para manejar situaciones en las que este campo no esté disponible inicialmente.

Signal para activar automáticamente nuevos usuarios

```
@receiver(post_save, sender=User)
def set_user_active(sender, instance, created, **kwargs):
    if created and not instance.is_active:
        instance.is_active = True
        instance.save()
```

Explicación del Signal

- **@receiver(post_save, sender=User)** : Este decorador indica que se ejecutará una acción (la función `set_user_active`) justo después de que se guarde un objeto del modelo `User` en la base de datos, lo que se conoce como un "signal". Aquí se usa el signal `post_save`, lo que significa que se ejecutará después de guardar un nuevo usuario.
- **created** : Este argumento indica si la instancia del modelo fue creada o simplemente actualizada. Si `created` es `True`, significa que un nuevo objeto de usuario fue creado.
- **instance.is_active** : `is_active` es un campo booleano del modelo `AbstractUser`, que determina si el usuario está activo y puede autenticarse. Por defecto, un nuevo usuario puede no estar activo.
- **Funcionalidad**: Si un nuevo usuario es creado (`created == True`) y no está marcado como activo (`not instance.is_active`), esta función cambia el campo `is_active` a `True` y guarda nuevamente el objeto de usuario. Esto asegura que los nuevos usuarios sean activados automáticamente.

Comentarios adicionales sobre los campos

En el comentario se observan referencias a tres campos (`id_vendedor`, `id_sucursal`, y `punto_venta`), que parecen corresponder a relaciones o atributos relacionados con vendedores, sucursales y puntos de venta. Estos campos no están actualmente implementados en el modelo, pero su inclusión podría sugerir futuras expansiones del modelo para soportar características adicionales.

- `id_vendedor` : Podría ser un `ForeignKey` a un modelo de `Vendedor`.
- `id_sucursal` : Podría ser un `ForeignKey` a un modelo de `Sucursal`.
- `punto_venta` : Un entero que podría representar un código de punto de venta asignado al usuario.

Resumen de la funcionalidad del modelo `User`

- Este modelo extiende el modelo de usuario estándar de Django, añadiendo campos personalizados para manejar información adicional, como correos alternativos, jerarquía, teléfono y si el usuario es un vendedor.

- Utiliza un signal `post_save` para asegurarse de que los nuevos usuarios se marquen como activos automáticamente al ser creados.
- Permite el manejo de datos opcionales con los atributos `null=True` y `blank=True`, lo que le da flexibilidad para almacenar o no cierta información adicional de los usuarios.

17.2. Formularios de la aplicación Usuarios

```
# D:\PROJECT_NEUMATIC\numatic\apps\usuarios\forms\user_form.py
from django import forms

from django.contrib.auth.forms import UserCreationForm, UserChangeForm
from django.contrib.auth.models import Group

#from apps.usuarios.models.user_models import User
from apps.usuarios.models import User

# -- Registrar Usuario
class RegistroUsuarioForm(UserCreationForm):

    email = forms.EmailField(required=True)

    class Meta:
        model = User
        fields = [
            'username',
            'first_name',
            'last_name',
            'email',
            'email_alt',
            'telefono',
            'is_active',
            'is_staff',
            'password1',
            'password2'
        ]

class EditarUsuarioForm(UserChangeForm):

    email = forms.EmailField(required=True)

    class Meta:
        model = User
        fields = [
            'username',
            'first_name',
            'last_name',
            'email',
            'email_alt',
            'telefono',
            'is_active',
            'is_staff',
        ]

class GroupForm(forms.ModelForm):
    class Meta:
```

```
model = Group
fields = ["name"]
```

Los formularios que se han mostrado, están diseñados para manejar la creación, edición y asignación de grupos a usuarios en Django. Utilizan las clases base proporcionadas por `django.contrib.auth.forms` para gestionar la autenticación de usuarios, como `UserCreationForm` y `UserChangeForm`, además de formularios personalizados para manejar el modelo de usuario extendido y el modelo `Group`. A continuación se explica cada formulario.

17.2.1. Formulario `RegistroUsuarioForm`

Este formulario hereda de `UserCreationForm`, que es un formulario prediseñado por Django para crear nuevos usuarios. Se ha personalizado para incluir campos adicionales del modelo de usuario personalizado `User`.

```
class RegistroUsuarioForm(UserCreationForm):

    email = forms.EmailField(required=True)

    class Meta:
        model = User
        fields = [
            'username',
            'first_name',
            'last_name',
            'email',
            'email_alt',
            'telefono',
            'is_active',
            'is_staff',
            'password1',
            'password2'
        ]
```

17.2.2. Formulario `EditarUsuarioForm`

Este formulario está diseñado para permitir la edición de usuarios existentes y hereda de `UserChangeForm`, que es el formulario de Django destinado a actualizar información de usuarios.

```
class EditarUsuarioForm(UserChangeForm):

    email = forms.EmailField(required=True)

    class Meta:
        model = User
        fields = [
            'username',
            'first_name',
            'last_name',
            'email',
            'email_alt',
            'telefono',
            'is_active',
            'is_staff',
        ]
```

Campos personalizados y explicaciones:

- **email** : Al igual que en `RegistroUsuarioForm`, se define un campo `EmailField` obligatorio para asegurarse de que se proporcione un correo electrónico válido cuando se edite un usuario.
- **Meta** : Al igual que en el formulario de registro, el bloque `Meta` define que el modelo que se está utilizando es `User` y especifica los campos que se pueden editar.
 - **Campos en `fields`** :
 - `username` : Permite modificar el nombre de usuario.
 - `first_name` : Permite modificar el nombre.
 - `last_name` : Permite modificar el apellido.
 - `email` : Permite modificar el correo electrónico principal.
 - `email_alt` : Permite modificar el correo electrónico alternativo.
 - `telefono` : Permite modificar el número de teléfono.
 - `is_active` : Permite cambiar el estado del usuario a activo o inactivo.
 - `is_staff` : Permite cambiar si el usuario tiene permisos de administrador.

Funcionalidad:

Este formulario es utilizado para editar la información de un usuario ya existente en el sistema. El `UserChangeForm` maneja la lógica de edición de los usuarios, pero este formulario lo personaliza para incluir campos adicionales como el correo alternativo y el teléfono, que son específicos del modelo `User`.

17.2.3. Formulario `GroupForm`

Este formulario está destinado a manejar la creación o modificación de grupos. Hereda de `forms.ModelForm`, lo que le permite trabajar directamente con el modelo `Group` de Django.

```
class GroupForm(forms.ModelForm):
    class Meta:
        model = Group
        fields = ["name"]
```

Campos personalizados y explicaciones:

- **Meta** : Define que el modelo que se utiliza es `Group` (un modelo incorporado en Django para manejar permisos y roles de usuarios) y que el único campo que se mostrará en el formulario es `name`.
 - **Campos en `fields`** :
 - `name` : El nombre del grupo, utilizado para organizar a los usuarios en roles o permisos específicos.

Funcionalidad:

Este formulario se utiliza para crear o editar grupos en Django. Los grupos en Django permiten asignar conjuntos de permisos a varios usuarios al mismo tiempo, y este formulario se enfoca en la edición o creación del nombre del grupo.

17.2.4. Resumen

- **RegistroUsuarioForm** : Este formulario se utiliza para registrar nuevos usuarios. Hereda de `UserCreationForm` y añade campos adicionales del modelo `User` personalizado.
- **EditarUsuarioForm** : Se usa para editar usuarios existentes, personalizando el formulario `UserChangeForm` para permitir la modificación de campos específicos del usuario.
- **GroupForm** : Permite crear y editar grupos de usuarios utilizando el modelo `Group` de Django, enfocándose en la gestión del campo `name`.

Estos formularios son esenciales en cualquier sistema que maneje autenticación de usuarios y permisos, permitiendo la personalización tanto en la creación como en la edición de usuarios y grupos.

17.3. Vistas de la aplicación Usuarios


```

# D:\PROJECT_NEUMATIC\neumatic\apps\usuarios\views\user_views_generics.py
from django.contrib.auth.views import LoginView, LogoutView
from django.views.generic import ListView, CreateView, UpdateView, DeleteView
from django.db.models import Q

#-- Vistas Genéricas Basada en Clases

class GenericLoginView(LoginView):
    pass

class GenericLogoutView(LogoutView):
    pass

class GenericListView(ListView):
    cadena_filtro = ""

    def get_queryset(self):
        queryset = super().get_queryset()
        text = self.request.GET.get('buscar', '')
        if text and self.cadena_filtro:
            queryset = queryset.filter( eval(self.cadena_filtro) )
        return queryset

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context["buscar"] = self.request.GET.get('buscar', '')
        return context

class GenericCreateView(CreateView):
    pass

class GenericUpdateView(UpdateView):
    #-- Nombre del argumento de clave primaria pasado en el url. Por defecto es pk.
    #pk_url_kwarg = "id"
    pass

class GenericDeleteView(DeleteView):
    #-- Nombre del argumento de clave primaria pasado en el url. Por defecto es pk.
    #pk_url_kwarg = "id"
    pass

```

```

# D:\PROJECT_NEUMATIC\neumatic\apps\usuarios\views\user_views.py
from django.urls import reverse_lazy

#from django.contrib.auth import authenticate, login, logout
#from django.contrib.auth.forms import AuthenticationForm

from django.contrib.auth.models import Group, Permission
from django.contrib.contenttypes.models import ContentType
#from django.contrib.auth.decorators import login_required

from .user_views_generics import *
from apps.usuarios.forms.user_form import *

#from apps.usuarios.models.user_models import User
from apps.usuarios.models import User

#-- Indicar las aplicaciones del proyecto para poder filtrar los modelos de las
mismas.
project_app_labels = ['usuarios', 'maestros', 'facturacion']

#-- Vista Login.
class CustomLoginView(GenericLoginView):
    template_name = 'usuarios/sesion_iniciar.html'

#-- Vista Logout.
class CustomLogoutView(GenericLogoutView):
    template_name = 'usuarios/sesion_cerrar.html'
    http_method_names = ["get", "post", "options"] # He tenido que incluir el método
GET para que funcione. NO DEBERÍA SER!!!

#-- Vistas de Grupos de usuarios.
#@method_decorator(login_required, name='dispatch')
class GrupoListView(GenericListView):
    model = Group
    context_object_name = 'grupos'
    template_name = "usuarios/grupo_list.html"
    cadena_filtro = "Q(name__icontains=text)"
    extra_context = {
        "home_view_name": "home",
    }

#@method_decorator(login_required, name='dispatch')
class GrupoCreateView(GenericCreateView):
    model = Group
    form_class = GroupForm

```

```

template_name = "usuarios/grupo_form.html"
success_url = reverse_lazy("grupo_listar") # Nombre de la url.
extra_context = {
    "accion": "Nuevo Grupo",
    "list_view_name": "grupo_listar"
}

}

#@method_decorator(login_required, name='dispatch')
class GrupoUpdateView(GenericUpdateView):
    model = Group
    form_class = GroupForm
    template_name = "usuarios/grupo_form.html"
    success_url = reverse_lazy("grupo_listar")
    extra_context = {
        "accion": "Editar Grupo",
        "list_view_name": "grupo_listar"
    }

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

        #-- Instancia del grupo que se edita.
        grupo = self.get_object()
        #-- Obtener los permisos asignados al grupo.
        permisos_asignados = grupo.permissions.all()

        #-- Obtener permisos disponibles.
        #-- Filtrar ContentType para solo incluir los de las apps del proyecto.
        project_content_types =
ContentType.objects.filter(app_label__in=project_app_labels)
        #-- Obtener permisos basados en los ContentType filtrados.
        permisos_disponibles =
Permission.objects.filter(content_type__in=project_content_types)

        context["permisos_asignados"] = permisos_asignados
        context["permisos_disponibles"] = permisos_disponibles

        return context

    def form_valid(self, form):
        # Guarda el formulario y realiza otras operaciones necesarias
        response = super().form_valid(form)

        # Procesa los permisos asignados y guarda en la base de datos
        permisos_asignados = self.request.POST.getlist('permisos_asignados')

        grupo = self.get_object()
        grupo.permissions.set(permisos_asignados)
        return response

```

```

#@method_decorator(login_required, name='dispatch')
class GrupoDeleteView(GenericDeleteView):
    model = Group
    template_name = "usuarios/grupo_confirm_delete.html"
    success_url = reverse_lazy("grupo_listar") # Nombre de la url.
    extra_context = {
        "accion": "Eliminar Grupo",
        "list_view_name": "grupo_listar"
    }

#-- Vistas de Usuarios.
#@method_decorator(login_required, name='dispatch')
class UsuarioListView(GenericListView):
    model = User
    context_object_name = 'usuarios'
    template_name = "usuarios/usuario_list.html"
    cadena_filtro = "Q(username__icontains=text) | Q(first_name__icontains=text) | Q(last_name__icontains=text) | Q(email__icontains=text)"
    extra_context = {
        "home_view_name": "home",
    }

#@method_decorator(login_required, name='dispatch')
class UsuarioCreateView(GenericCreateView):
    model = User
    form_class = RegistroUsuarioForm
    template_name = "usuarios/usuario_crear_form.html"
    success_url = reverse_lazy("usuario_listar") # Nombre de la url.
    extra_context = {
        "accion": "Registro de Usuario",
        "list_view_name": "usuario_listar"
    }

#@method_decorator(login_required, name='dispatch')
class UsuarioUpdateView(GenericUpdateView):
    model = User
    form_class = EditarUsuarioForm
    template_name = "usuarios/usuario_editar_form.html"
    success_url = reverse_lazy("usuario_listar") # Nombre de la url.
    extra_context = {
        "accion": "Editar Usuario",
        "list_view_name": "usuario_listar"
    }

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

```

```

    #-- Instancia del grupo que se edita.
    usuario = self.get_object()
    #-- Obtener los permisos asignados al grupo.
    permisos_asignados = usuario.user_permissions.all()

    #-- Obtener permisos disponibles.
    #-- Filtrar ContentType para solo incluir los de las apps del proyecto.
    project_content_types =
ContentType.objects.filter(app_label__in=project_app_labels)
    #-- Obtener permisos basados en los ContentType filtrados.
    permisos_disponibles =
Permission.objects.filter(content_type__in=project_content_types)

    #-- Obtener grupos asignados.
    grupos_asignados = usuario.groups.all()
    #-- Obtener grupos disponibles.
    grupos_disponibles = Group.objects.all()

    context["grupos_asignados"] = grupos_asignados
    context["grupos_disponibles"] = grupos_disponibles
    context["permisos_asignados"] = permisos_asignados
    context["permisos_disponibles"] = permisos_disponibles

    return context

def form_valid(self, form):
    # Guarda el formulario y realiza otras operaciones necesarias
    response = super().form_valid(form)

    # Procesa los grupos y/o permisos asignados y guarda en la base de datos
    grupos_asignados = self.request.POST.getlist('grupos_asignados')
    permisos_asignados = self.request.POST.getlist('permisos_asignados')

    usuario = self.get_object()
    usuario.groups.set(grupos_asignados)
    usuario.user_permissions.set(permisos_asignados)

    return response

#@method_decorator(login_required, name='dispatch')
class UsuarioDeleteView(GenericDeleteView):
    model = User
    template_name = "usuarios/usuario_confirm_delete.html"
    success_url = reverse_lazy("usuario_listar") # Nombre de la url.
    extra_context = {
        "accion": "Eliminar Usuario",
        "list_view_name": "usuario_listar"
    }

```

Explicación de las vistas

17.3.1. CustomLoginView

- Hereda de `GenericLoginView`.
- Renderiza una plantilla de inicio de sesión `usuarios/sesion_iniciar.html`.
- Esta vista permite a los usuarios iniciar sesión.

17.3.2. CustomLogoutView

- Hereda de `GenericLogoutView`.
- Renderiza la plantilla `usuarios/sesion_cerrar.html`.
- Maneja las solicitudes de cierre de sesión, permitiendo métodos GET, POST y OPTIONS.
- **Nota:** El uso del método GET no es ideal para cerrar sesiones, pero se ha incluido para hacer que funcione correctamente.

17.3.3. GrupoListView

- Hereda de `GenericListView` y muestra una lista de grupos de usuarios.
- Renderiza la plantilla `usuarios/grupo_list.html`.
- El filtro de búsqueda permite buscar grupos cuyo nombre contenga una determinada cadena.
- Incluye un contexto adicional para especificar la vista de inicio (`home_view_name`).

17.3.4. GrupoCreateView

- Hereda de `GenericCreateView`.
- Utiliza el formulario `GroupForm` para crear nuevos grupos de usuarios.
- Renderiza la plantilla `usuarios/grupo_form.html`.
- Redirige a `grupo_listar` después de crear el grupo con éxito.
- Añade contexto adicional para indicar la acción "Nuevo Grupo".

17.3.5. GrupoUpdateView

- Hereda de `GenericUpdateView`.
- Permite actualizar la información de un grupo existente utilizando `GroupForm`.
- Renderiza la plantilla `usuarios/grupo_form.html`.
- Añade contexto adicional para indicar la acción "Editar Grupo".
- En el método `get_context_data`, agrega al contexto los permisos asignados al grupo y los permisos disponibles filtrados por las aplicaciones del proyecto.

- En el método `form_valid`, procesa los permisos asignados al grupo después de guardar los cambios.

17.3.6. GrupoDeleteView

- Hereda de `GenericDeleteView`.
- Permite eliminar un grupo de usuarios.
- Renderiza la plantilla de confirmación `usuarios/grupo_confirm_delete.html`.
- Redirige a `grupo_listar` después de eliminar el grupo con éxito.
- Incluye un contexto adicional para indicar la acción "Eliminar Grupo".

17.3.7. UsuarioListView

- Hereda de `GenericListView`.
- Muestra una lista de usuarios.
- Renderiza la plantilla `usuarios/usuario_list.html`.
- Permite buscar usuarios por nombre de usuario, nombre, apellido o correo electrónico.
- Añade un contexto adicional para especificar la vista de inicio.

17.3.8. UsuarioCreateView

- Hereda de `GenericCreateView`.
- Utiliza el formulario `RegistroUsuarioForm` para registrar un nuevo usuario.
- Renderiza la plantilla `usuarios/usuario_crear_form.html`.
- Redirige a `usuario_listar` después de registrar el usuario con éxito.
- Añade un contexto adicional para indicar la acción "Registro de Usuario".

17.3.9. UsuarioUpdateView

- Hereda de `GenericUpdateView`.
- Utiliza el formulario `EditarUsuarioForm` para actualizar la información de un usuario existente.
- Renderiza la plantilla `usuarios/usuario_editar_form.html`.
- Añade contexto adicional sobre los permisos y grupos asignados al usuario, además de los disponibles para asignar.
- En el método `form_valid`, guarda los permisos y grupos asignados al usuario después de actualizar.

17.3.10. UsuarioDeleteView

- Hereda de `GenericDeleteView`.
- Permite eliminar un usuario.
- Renderiza la plantilla de confirmación `usuarios/usuario_confirm_delete.html`.
- Redirige a `usuario_listar` después de eliminar al usuario con éxito.
- Añade un contexto adicional para indicar la acción "Eliminar Usuario".

17.4. Rutas de la aplicación Usuarios

```
# D:\PROJECT_NEUMATIC\neumatic\apps\usuarios\urls.py
from django.urls import path
from apps.usuarios.views.user_views import *

urlpatterns = [
    #-- Login/Logout
    path('sesion/iniciar/', CustomLoginView.as_view(), name='iniciar_sesion'),
    path('sesion/cerrar/', CustomLogoutView.as_view(), name='cerrar_sesion'),

    #-- Grupos
    path('grupo/listar/', GrupoListView.as_view(), name='grupo_listar'),
    path('grupo/crear/', GrupoCreateView.as_view(), name='grupo_crear'),
    path('grupo/editar/<int:pk>/', GrupoUpdateView.as_view(), name='grupo_editar'),
    path('grupo/eliminar/<int:pk>/', GrupoDeleteView.as_view(),
name='grupo_eliminar'),

    #-- Usuarios
    path('usuario/listar/', UsuarioListView.as_view(), name='usuario_listar'),
    path('usuario/crear/', UsuarioCreateView.as_view(), name='usuario_crear'),
    path('usuario/editar/<int:pk>/', UsuarioUpdateView.as_view(),
name='usuario_editar'),
    path('usuario/eliminar/<int:pk>/', UsuarioDeleteView.as_view(),
name='usuario_eliminar')
]
```

Las rutas definidas en este archivo `urls.py` en el proyecto Django configuran las URL para manejar operaciones relacionadas con la autenticación, grupos de usuarios y usuarios en la aplicación `usuarios`. Aquí está la explicación de cada ruta:

Login/Logout

- `path('sesion/iniciar/', CustomLoginView.as_view(), name='iniciar_sesion')`:
 - **Ruta:** `/sesion/iniciar/`
 - **Vista:** `CustomLoginView`
 - **Descripción:** Gestiona la vista de inicio de sesión (login). Redirige al usuario a la página para iniciar sesión.

- **Nombre de URL:** `iniciar_sesion`
- `path('sesion/cerrar/', CustomLogoutView.as_view(), name='cerrar_sesion') :`
 - **Ruta:** `/sesion/cerrar/`
 - **Vista:** `CustomLogoutView`
 - **Descripción:** Gestiona la vista de cierre de sesión (logout). Permite que el usuario cierre su sesión.
 - **Nombre de URL:** `cerrar_sesion`

Grupos

- `path('grupo/listar/', GrupoListView.as_view(), name='grupo_listar') :`
 - **Ruta:** `/grupo/listar/`
 - **Vista:** `GrupoListView`
 - **Descripción:** Muestra una lista de grupos de usuarios. Permite ver todos los grupos en el sistema.
 - **Nombre de URL:** `grupo_listar`
- `path('grupo/crear/', GrupoCreateView.as_view(), name='grupo_crear') :`
 - **Ruta:** `/grupo/crear/`
 - **Vista:** `GrupoCreateView`
 - **Descripción:** Permite crear un nuevo grupo de usuarios.
 - **Nombre de URL:** `grupo_crear`
- `path('grupo/editar/<int:pk>', GrupoUpdateView.as_view(), name='grupo_editar') :`
 - **Ruta:** `/grupo/editar/<int:pk>/`
 - **Vista:** `GrupoUpdateView`
 - **Descripción:** Permite editar un grupo existente. El identificador `<int:pk>` indica el grupo específico a editar.
 - **Nombre de URL:** `grupo_editar`
- `path('grupo/eliminar/<int:pk>', GrupoDeleteView.as_view(), name='grupo_eliminar') :`
 - **Ruta:** `/grupo/eliminar/<int:pk>/`
 - **Vista:** `GrupoDeleteView`
 - **Descripción:** Permite eliminar un grupo existente. El identificador `<int:pk>` indica el grupo específico a eliminar.
 - **Nombre de URL:** `grupo_eliminar`

Usuarios

- `path('usuario/listar/', UsuarioListView.as_view(), name='usuario_listar') :`
 - **Ruta:** `/usuario/listar/`

- **Vista:** `UsuarioListView`
- **Descripción:** Muestra una lista de usuarios registrados en el sistema.
- **Nombre de URL:** `usuario_listar`
- `path('usuario/crear/', UsuarioCreateView.as_view(), name='usuario_crear') :`
 - **Ruta:** `/usuario/crear/`
 - **Vista:** `UsuarioCreateView`
 - **Descripción:** Permite crear un nuevo usuario.
 - **Nombre de URL:** `usuario_crear`
- `path('usuario/editar/<int:pk>', UsuarioUpdateView.as_view(), name='usuario_editar') :`
 - **Ruta:** `/usuario/editar/<int:pk>/`
 - **Vista:** `UsuarioUpdateView`
 - **Descripción:** Permite editar un usuario existente. El identificador `<int:pk>` indica el usuario específico a editar.
 - **Nombre de URL:** `usuario_editar`
- `path('usuario/eliminar/<int:pk>', UsuarioDeleteView.as_view(), name='usuario_eliminar') :`
 - **Ruta:** `/usuario/eliminar/<int:pk>/`
 - **Vista:** `UsuarioDeleteView`
 - **Descripción:** Permite eliminar un usuario existente. El identificador `<int:pk>` indica el usuario específico a eliminar.
 - **Nombre de URL:** `usuario_eliminar`

Conclusión

Estas rutas configuran las vistas para gestionar el inicio y cierre de sesión, así como la administración de usuarios y grupos. Cada ruta está asociada con vistas genéricas personalizadas (`GenericListView`, `GenericCreateView`, `GenericUpdateView`, `GenericDeleteView`), que facilitan operaciones CRUD sobre los modelos `User` y `Group`.

18. CRUDs de la aplicación maestros

18.1. El modelo Actividad

El modelo ya fue creado previamente, pero vamos a mostrar su estructura

```
# D:\PROJECT_NEUMATIC\numatic\apps\maestros\models\base_models.py
from django.db import models
from .base_gen_models import ModeloBaseGenerico
from .sucursal_models import Sucursal

# -- Datos estándares aplicables a los modelos base
ESTATUS_GEN = [
    (True, 'Activo'),
    (False, 'Inactivo'),
]

class Actividad(ModeloBaseGenerico):
    id_actividad = models.AutoField(primary_key=True)
    estatus_actividad = models.BooleanField("Estatus", default=True,
                                           choices=ESTATUS_GEN)
    descripcion_actividad = models.CharField(max_length=30)
    fecha_registro_actividad = models.DateField(blank=True, null=True)

    class Meta:
        db_table = 'actividad'
        verbose_name = ('Actividad')
        verbose_name_plural = ('Actividades')
        ordering = ['descripcion_actividad']
```

18.2. El formulario ActividadForm

¿Cómo crear un archivo de formulario?

Para definir el nombre del archivo que contiene el formulario asociado a un modelo, debemos seguir las siguientes reglas, según la notación Snake Case en minúsculas:

- Si el nombre del modelo es de una sola palabra:

Empresa ---> **empresa_forms.py**

- Si el nombre del modelo es de dos palabras:

NivelEducativo ---> **nivel_educativo_forms.py**

- Si el nombre del modelo es de tres palabras:

TipoDocumentIdentidad ---> **tipo_documento_identidad_forms.py**

La ubicación del archivo depende de la estructura del proyecto, en nuestro caso pertenece a la aplicación maestros, por lo tanto, se ubica en la carpeta maestros/forms

En nuestro caso: **Actividad** ---> **actividad_forms.py**

El contenido del archivo **actividad_forms.py** es:

```
# neumatic\apps\maestros\forms\actividad_forms.py
from django import forms
from ..models.base_models import Actividad
from diseno_base.disenio_bootstrap import (
    formclasstext, formclassselect, formclassdate)

class ActividadForm(forms.ModelForm):

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

        #-- Agregar clases CSS a los campos con errores.
        for field in self.fields:
            if self[field].errors:
                self.fields[field].widget.attrs['class'] += ' border-danger is-
invalid'

    class Meta:
        model = Actividad
        fields = '__all__'

        widgets = {
            'estatus_actividad':
                forms.Select(attrs={**formclassselect}),
            'descripcion_actividad':
                forms.TextInput(attrs={**formclasstext,
                                        'placeholder': 'Descripción Actividad'}),
            'fecha_registro_actividad':
                forms.TextInput(attrs={**formclassdate,
                                        'placeholder': 'Fecha de Registro' }),
        }
```

18.3. Las vistas del CRUD del modelo Actividad

```

# neumatic\apps\maestros\views\cruds_views_generics.py
from django.views.generic import ListView, CreateView, UpdateView, DeleteView,
DetailView
from django.db.models import Q
from django.http import JsonResponse

#-- Recursos necesarios para proteger las rutas.
from django.utils.decorators import method_decorator
from django.contrib.auth.decorators import login_required

#-- Recursos necesarios para los permisos de usuarios sobre modelos.
from django.contrib.auth.mixins import PermissionRequiredMixin
from django.contrib import messages
from django.shortcuts import redirect

from django.utils import timezone

# -- Vistas Genéricas Basada en Clases -----
@method_decorator(login_required, name='dispatch')
class MaestroListView(ListView):
    cadena_filtro = ""
    paginate_by = 8

    search_fields = []
    ordering = []

    table_headers = {}
    table_data = []
    pagination_options = [8, 20, 30, 40, 50]

    def get_queryset(self):
        #-- Acá ya determina el Modelo con el que se trabaja.
        #-- Obtiene todos los registros sin filtro.
        #-- Con lo cual no es necesario un filter.all().
        #-- luego cambiar a que por defecto no haya registros.
        queryset = super().get_queryset()

        #-- Obtener el valor de paginate_by de la URL, si está presente.
        paginate_by_param = self.request.GET.get('paginate_by')
        if paginate_by_param is not None:
            try:
                #-- Intentar convertir a entero, usar valor predeterminado si falla.
                paginate_by_value = int(paginate_by_param)
                self.paginate_by = paginate_by_value
            except ValueError:
                pass

        #-- Obtener la cadena de filtro.
        query = self.request.GET.get('busqueda', None)

```

```

    #-- Crear la cadena de filtro en base a la lista search_fields-
    cadena_filtro = ""
    for field in self.search_fields:
        expression = f"Q({field}__icontains='{query}')"
        cadena_filtro += expression + " | "

    #-- Eliminar el último " | " en la cadena de filtro.
    cadena_filtro = "(" + cadena_filtro[:-3] + ")"

    #-- Ejecutar la consulta.
    if query and cadena_filtro:
        queryset = queryset.filter(eval(cadena_filtro))

    # Ordenar el queryset según la lista ordering
    queryset = queryset.order_by(*self.ordering)

    return queryset

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context["busqueda"] = self.request.GET.get('busqueda', '')

    #-- Agregar valores de paginación y valor seleccionado.
    context['pagination_options'] = self.pagination_options
    context['selected_pagination'] = int(self.paginate_by)
    # Para pasar la fecha a la lista del maestro
    context['fecha'] = timezone.now()

    return context

def get(self, request, *args, **kwargs):
    #-- Obtener el valor de paginate_by de la URL, si está presente.
    paginate_by_param = self.request.GET.get('paginate_by')
    if paginate_by_param is not None:
        try:
            #-- Intentar convertir a entero, usar valor predeterminado si falla.
            paginate_by_value = int(paginate_by_param)
            self.paginate_by = paginate_by_value
        except ValueError:
            pass

    #-- Mantener el valor de paginate_by en el formulario de paginación.
    self.request.GET = self.request.GET.copy()
    self.request.GET['paginate_by'] = str(self.paginate_by)

    return super().get(request, *args, **kwargs)

def get_paginate_by(self, queryset):
    #-- Utilizar el valor actualizado de paginate_by.
    return self.paginate_by

```

```

@method_decorator(login_required, name='dispatch')
class MaestroCreateView(PermissionRequiredMixin, CreateView):
    list_view_name = None

    def form_invalid(self, form):
        """
        Si el formulario no es válido, renderiza el formulario con los errores.
        """
        return self.render_to_response(self.get_context_data(form=form))

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        #-- Asegurarse de que el formulario en el contexto sea el mismo que se validó
        if 'form' not in context:
            context['form'] = self.get_form()
        context['requerimientos'] = obtener_requerimientos_modelo(self.model)
        return context

    #-- Método que agrega mensaje cuando no tiene permiso de crear.
    def handle_no_permission(self):
        messages.error(self.request, 'No tienes permiso para realizar esta acción.')
        return redirect(self.list_view_name)

@method_decorator(login_required, name='dispatch')
class MaestroUpdateView(PermissionRequiredMixin, UpdateView):
    list_view_name = None

    def form_invalid(self, form):
        """
        Si el formulario no es válido, renderiza el formulario con los errores.
        """
        return self.render_to_response(self.get_context_data(form=form))

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        #-- Asegurarse de que el formulario en el contexto sea el mismo que se validó
        if 'form' not in context:
            context['form'] = self.get_form()
        context['requerimientos'] = obtener_requerimientos_modelo(self.model)
        return context

    #-- Método que agrega mensaje cuando no tiene permiso de modificar.
    def handle_no_permission(self):
        messages.error(self.request, 'No tienes permiso para realizar esta acción.')
        return redirect(self.list_view_name)

@method_decorator(login_required, name='dispatch')

```

```

class MaestroDeleteView(PermissionRequiredMixin, DeleteView):
    list_view_name = None

    #-- Método que agrega mensaje cuando no tiene permiso de eliminar.
    def handle_no_permission(self):
        messages.error(self.request, 'No tienes permiso para realizar esta acción.')
        return redirect(self.list_view_name)

@method_decorator(login_required, name='dispatch')
class GenericDetailView(DetailView):
    def get_data(self, obj):
        """
        Este método debe ser sobrescrito en la clase hija
        para proporcionar los datos específicos.
        """
        return {}

    def render_to_response(self, context, **response_kwargs):
        obj = self.get_object()
        data = self.get_data(obj)
        return JsonResponse(data)

def obtener_requerimientos_modelo(modelo):
    requerimientos = {}

    for field in modelo._meta.fields:
        exclud_fields = ['usuario', 'estacion', 'fcontrol']
        field_info = []

        if not field.primary_key and field.name not in exclud_fields:
            if not field.blank:
                field_info.append("Este campo es obligatorio")

            if hasattr(field, "max_length") and field.max_length is not None:
                field_info.append(f"Debe tener un máximo de {field.max_length} caracteres")

            if field.unique:
                field_info.append("Debe ser único")

            requerimientos[field.verbose_name] = field_info

    return requerimientos

```



```

# neumatic\apps\maestros\views\actividad_views.py
from django.urls import reverse_lazy
from ..views.cruds_views_generics import *
from ..models.base_models import Actividad
from ..forms.actividad_forms import ActividadForm

class ConfigViews():
    # Modelo
    model = Actividad

    # Formulario asociado al modelo
    form_class = ActividadForm

    # Aplicación asociada al modelo
    app_label = model._meta.app_label

    #-- Deshabilitado por redundancia:
    # # Título del listado del modelo
    # master_title = model._meta.verbose_name_plural

    #-- Usar esta forma cuando el modelo esté compuesto de una sola palabra: Ej.
    # Color.
    model_string = model.__name__.lower() #-- Usar esta forma cuando el modelo esté
    # compuesto de una sola palabra: Ej. Color.

    #-- Usar esta forma cuando el modelo esté compuesto por más de una palabra: Ej.
    # TipoCambio colocar "tipo_cambio".
    #model_string = "tipo_cambio"

    # Permisos
    permission_add = f"{app_label}.add_{model_string}"
    permission_change = f"{app_label}.change_{model_string}"
    permission_delete = f"{app_label}.delete_{model_string}"

    # Vistas del CRUD del modelo
    list_view_name = f"{model_string}_list"
    create_view_name = f"{model_string}_create"
    update_view_name = f"{model_string}_update"
    delete_view_name = f"{model_string}_delete"

    # Plantilla para crear o actualizar el modelo
    template_form = f"{app_label}/{model_string}_form.html"

    # Plantilla para confirmar eliminación de un registro
    template_delete = "base_confirm_delete.html"

    # Plantilla de la lista del CRUD
    template_list = f'{app_label}/maestro_list.html'

```

```

# Contexto de los datos de la lista
context_object_name = 'objetos'

# Vista del home del proyecto
home_view_name = "home"

# Nombre de la url
success_url = reverse_lazy(list_view_name)

class DataViewList():
    search_fields = ['descripcion_actividad']

    ordering = ['descripcion_actividad']

    paginate_by = 8

    table_headers = {
        'descripcion_actividad': (2, 'Descripción'),
        'fecha_registro_actividad': (2, 'Fecha Registro'),
        'acciones': (2, 'Acciones'),
    }

    table_data = [
        {'field_name': 'descripcion_actividad', 'date_format': None},
        {'field_name': 'fecha_registro_actividad', 'date_format': 'd/m/Y'},
    ]

# ActividadListView - Inicio
class ActividadListView(MaestroListView):
    model = ConfigViews.model
    template_name = ConfigViews.template_list
    context_object_name = ConfigViews.context_object_name

    search_fields = DataViewList.search_fields
    ordering = DataViewList.ordering

    extra_context = {
        "master_title": ConfigViews.model._meta.verbose_name_plural,
        "home_view_name": ConfigViews.home_view_name,
        "list_view_name": ConfigViews.list_view_name,
        "create_view_name": ConfigViews.create_view_name,
        "update_view_name": ConfigViews.update_view_name,
        "delete_view_name": ConfigViews.delete_view_name,
        "table_headers": DataViewList.table_headers,
        "table_data": DataViewList.table_data,
    }

# ActividadCreateView - Inicio

```

```

class ActividadCreateView(MaestroCreateView):
    model = ConfigViews.model
    list_view_name = ConfigViews.list_view_name
    form_class = ConfigViews.form_class
    template_name = ConfigViews.template_form
    success_url = ConfigViews.success_url

    #-- Indicar el permiso que requiere para ejecutar la acción.
    # (revisar de donde lo copiaste que tienes asignado permission_change en vez de
permission_add)
    permission_required = ConfigViews.permission_add

    extra_context = {
        "accion": f"Editar {ConfigViews.model._meta.verbose_name}",
        "list_view_name" : ConfigViews.list_view_name
    }

# ActividadUpdateView
class ActividadUpdateView(MaestroUpdateView):
    model = ConfigViews.model
    list_view_name = ConfigViews.list_view_name
    form_class = ConfigViews.form_class
    template_name = ConfigViews.template_form
    success_url = ConfigViews.success_url

    #-- Indicar el permiso que requiere para ejecutar la acción.
    permission_required = ConfigViews.permission_change

    extra_context = {
        "accion": f"Editar {ConfigViews.model._meta.verbose_name}",
        "list_view_name" : ConfigViews.list_view_name
    }

# ActividadDeleteView
class ActividadDeleteView (MaestroDeleteView):
    model = ConfigViews.model
    list_view_name = ConfigViews.list_view_name
    template_name = ConfigViews.template_delete
    success_url = ConfigViews.success_url

    #-- Indicar el permiso que requiere para ejecutar la acción.
    permission_required = ConfigViews.permission_delete

    extra_context = {
        "accion": f"Eliminar {ConfigViews.model._meta.verbose_name}",
        "list_view_name" : ConfigViews.list_view_name,
        "mensaje": "Estás seguro de eliminar el Registro"
    }

```

18.4. Las rutas del CRUD del modelo Actividad

18.4.1. Las rutas a nivel de la aplicación

```
# \apps\maestros\urls.py
from django.urls import path

from .views.actividad_views import *

urlpatterns = [
    path('actividad/', ActividadListView.as_view(), name='actividad_list'),
    path('actividad/nueva/', ActividadCreateView.as_view(), name='actividad_create'),
    path('actividad/<int:pk>/editar/', ActividadUpdateView.as_view(),
name='actividad_update'),
    path('actividad/<int:pk>/eliminar/', ActividadDeleteView.as_view(),
name='actividad_delete'),
]
```

18.4.2. Las rutas a nivel del Proyecto

```
# neumatic\neumatic\urls.py
from django.contrib import admin
from django.urls import path, include
from .views import home_view

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', home_view, name='home'),
    path('usuarios/', include('apps.usuarios.urls')),
    path('maestros/', include('apps.maestros.urls')),
]
```

18.5. Las plantillas del CRUD del modelo Actividad

Para el listado del CRUD de todos los modelos en la aplicación maestros se utiliza una plantilla única: `neumatic\apps\maestros\templates\maestros\maestro_list.html`

```

{% extends 'base.html' %}
{% load static %}

{% load custom_tags %}

<!-- Block Title -->
{% block title %}
    {{ master_title }}
{% endblock title %}

{% block style %}
    body {
        background-color: rgb(0, 204, 255);
    }

    .tbl-container {

    }

    .tbl-fixed {
        overflow-x: scroll;
        overflow-y: scroll;
        height: fit-content;
        max-height: 60vh;
        margin-top: 0px;
        font-size: 80%;
    }
    table {
        min-width: max-content;
    }
    table th {
        position: sticky;
        top: 0px;
    }
{% endblock style %}

{% debug %}

<!-- Block Header -->
{% block header %}
    {% include 'top_nav.html' %}
{% endblock header %}

<!-- Block Main -->
{% block main %}
    {% block sidebar %}
        {% if user.is_authenticated %}
            {% include 'sidebar.html' %}
        {% endif %}
    {% endblock sidebar %}

```

```

{% block maincomponent %}
<!-- Main Component Start -->

    <!-- Main principalcomponent Start -->
    {% block principalcomponent %}
    <div id="layoutSidenav_content">
        <main>
            <div class="container-fluid tbl-container">
                <div class="card border-secondary mb-3 mt-2">
                    <div class="card-header bg-primary bg-opacity-25 text-dark d-
flex
                    justify-content-between p-1">
                        <h4>{{ master_title }}</h4>
                        <!-- Botón cerrar lista -->
                        <button type="button" class="btn-close btn-close-blue"
                            aria-label="Close"
                            onclick="window.location.href='{% url home_view_name
%}'">

                    </button>
                </div>
                <div class="card-body">
                    <div class="row">
                        <div class="col col-10">
                            {% include 'buscador.html' %}
                        </div>

                        <div class="col col-2">
                            <a href="{% url create_view_name %}"
                                class="btn btn-primary opacity-75">
                                Nuevo
                            </a>
                        </div>
                    </div>
                    <div class="row tbl-fixed my-2 mx-1">
                        <!-- Inicio Tabla de Maestro -->
                        <table class="table table-bordered table-striped
table-hover">

                            <thead class="table-primary">
                                {% for key, value in table_headers.items %}
                                    <th class="col-{{ value.0 }}">
                                        <a href="{% url list_view_name %}"

                                            {{ value.1 }}
                                            {% if key == order_by %}
                                                ▲
                                            {% endif %}
                                        </a>
                                    </th>
                                {% endfor %}
                            </thead>
                        </table>
                    </div>
                </div>
            </div>
        </main>
    </div>
    {% endblock principalcomponent %}
{% endblock maincomponent %}

```

```

        </thead>

        <tbody>
            {% for objeto in objetos %}
                <tr>
                    {% for data_info in table_data %}
                        {% if data_info.date_format %}
                            <td>{{
objeto|get_attribute:data_info.field_name|default:"" |date:data_info.date_format }}
                        </td>
                            {% else %}
                                <td>{{
objeto|get_attribute:data_info.field_name|default:"" }}</td>
                            {% endif %}
                        {% endfor %}

                        <td>
                            <!-- Botón Actualizar -->
                            <button type="button" class="btn
btn-outline-primary btn-sm btn-edit boton-oculto" data-form-index="{ { forloop.counter0
}}"

onclick="window.location.href='{% url update_view_name objeto.pk %}'"
                                data-bs-toggle="tooltip"
                                data-bs-placement="top" title="Actualizar Registro"
                                style="font-size: 80%;
padding: 0.3rem 0.6rem;">
                                <i class="fa fa-edit"></i>
                            </button>

                            <!-- Botón Eliminar -->
                            <button type="button" class="btn
btn-outline-danger btn-sm btn-delete boton-oculto"
                                data-bs-toggle="modal"
                                data-bs-target="#deleteModal"
                                data-bs-whatever-url="{%
url delete_view_name objeto.pk %}" data-bs-whatever-name="{ { objeto }}"
                                style="font-size: 80%;
padding: 0.3rem 0.6rem;">
                                <i class="fa fa-trash-alt">
                            </i>
                            </button>

                            <!-- Botón Ver -->
                            <button type="button" class="btn
btn-outline-success btn-sm btn-view boton-oculto"
                                data-bs-toggle="tooltip"
                                data-bs-placement="top" title="Ver Registro"
                                style="font-size: 80%;
padding: 0.3rem 0.6rem;">
                                <i class="fa fa-eye"></i>

```

```

</button>

</td>
</tr>
{% endfor %}
</tbody>
</table>
<!-- Final Tabla de Maestro -->
</div>

{% block paginador %}
    {% include 'paginador.html' %}
{% endblock paginador %}
</div>
</div>
</div>
</main>
</div>
{% endblock principalcomponent %}
<!-- Main principalcomponent End -->

{% endblock maincomponent %}

{% endblock main %}

{% block modals %}
    <!-- Modal para confirmar eliminación -->
    <div class="modal fade" id="deleteModal" tabindex="-1" aria-
labelledby="deleteModallabel" aria-hidden="true">
        <div class="modal-dialog d-flex align-items-center">
            <div class="modal-content border border-white">
                <div class="modal-header bg-primary bg-opacity-25">
                    <h1 class="modal-title fs-5" id="deleteModallabel">Confirmar
Eliminación</h1>
                    <button type="button" class="btn-close" data-bs-dismiss="modal"
aria-label="Close"></button>
                </div>
                <div class="modal-body">
                    <p>Se eliminará: <strong id="deleteItemName"></strong></p>
                    ¿Estás seguro que eliminar el registro?
                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Cancelar</button>
                    <form id="deleteForm" method="post" class="d-inline">
                        {% csrf_token %}
                        <button type="submit" class="btn btn-danger">Sí,
eliminar</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>

```



```

        </div>
    </div>

    <!-- Modal para el manejo de los mensajes de error sobre los permisos de usuario.
-->
{% endblock modals %}

{% block script %}
    <script>
        var exampleModal = document.getElementById('deleteModal')
        exampleModal.addEventListener('show.bs.modal', function (event) {
            var button = event.relatedTarget
            var recipient = button.getAttribute('data-bs-whatever-name')
            var url = button.getAttribute('data-bs-whatever-url')

            var modalTitle = exampleModal.querySelector('.modal-title')
            var modalBodyInput = exampleModal.querySelector('.modal-body strong')
            var form = exampleModal.querySelector('#deleteForm')

            modalTitle.textContent = 'Eliminar ' + recipient
            modalBodyInput.textContent = recipient
            form.action = url
        })
    </script>

    <script>
        document.addEventListener('DOMContentLoaded', function () {
            const messageList = document.getElementById('message-list');
            if (messageList.children.length > 0) {
                const errorModal = new
bootstrap.Modal(document.getElementById('errorModal'));
                errorModal.show();
            }
        });
    </script>
{% endblock script %}

```

Para la creación y actualización de registros del modelo Actividad, se utiliza la plantilla personalizada: `neumatic\apps\maestros\templates\maestros\actividad_form.html`

[illegible]

```

form.estatus_actividad.errors }}

    </div>
  </div>
  <div class="col-md-4">
    <label class="form-label text-
primary mb-0" for="{ form.descripcion_actividad.id_for_label }">
      {{
form.descripcion_actividad.label_tag }}

    </label>
    {{ form.descripcion_actividad}}
    <div class="invalid-feedback">
      {{
form.descripcion_actividad.errors }}

    </div>
  </div>
</div>

  <div class="container mt-3">
    <button class="btn btn-primary" type="submit"
id="guardarBtn">

      Guardar
    </button>
    <a class="btn btn-secondary" href="{% url
list_view_name %}">

      Cancelar
    </a>
  </div>
</form>
</div>
</div>
</div>
</main>
</div>
{% endblock principalcomponent %}
{% endblock maincomponent %}

```

Observamos que la plantilla actividad_form.html, hereda de la plantilla:

neumatic\templates\maestro_form.html cuyo contenido es:

```

{% extends 'base.html' %}
{% load static %}

<!-- Block Header ----->
{% block header %}
    {% include 'top_nav.html' %}
{% endblock header %}

{% comment %} Block Main {% endcomment %}
{% block main %}
    {% block sidebar %}
        {% include 'sidebar.html' %}
    {% endblock sidebar %}

    {% block maincomponent %}
        {% block principalcomponent %}
            {% endblock principalcomponent %}
        {% endblock maincomponent %}
    {% endblock main %}

<!-- Block Fotter ----->
{% block footer %}
{% endblock footer %}

{% block script %}
{% endblock script %}

```

La plantilla maestro_form.htm hereda de la plantilla base.html. Pero adicionalmente incluye dos plantillas: neumatic\templates\top_nav.html, cuyo contenido es:

```

<nav class="sb-topnav navbar navbar-expand navbar-dark bg-primary">
  <!-- Navbar Brand-->
  <a class="navbar-brand ps-3" href="{% url 'home' %}">MaaSoft</a>

  <!-- Sidebar Toggle-->
  <button class="btn btn-link btn-sm order-1 order-lg-0 me-4 me-lg-0"
id="sidebarToggle" href="#!"><i class="fas fa-bars"></i></button>

  <!-- Se agrega la clase flex-fill para que "empuje" los siguientes elementos al
final del div -->
  <!-- <div class="col-auto align-self-center text-white"> -->
  <div class="col-auto align-self-center text-white flex-fill">
    {{ fecha.date|date:"d/m/Y" }}
  </div>

  <!-- Navbar Search-->
  <!-- <form class="d-none d-md-inline-block form-inline ms-auto me-0 me-md-3 my-2
my-md-0">
    <div class="input-group">
      <input class="form-control" type="text" placeholder="Buscar..." aria-
label="Search for..." aria-describedby="btnNavbarSearch" />
      <button class="btn btn-primary" id="btnNavbarSearch" type="button"><i
class="fas fa-search"></i></button>
    </div>
  </form> -->

  <div class="col-auto align-self-center text-white">
    {% if user.is_authenticated %}
      {% if user.first_name%}
        {{user.first_name}} {{user.last_name}}
      {% else %}
        {{ user }}
      {% endif %}
    {% endif %}
  </div>

  <!-- Navbar-->
  <ul class="navbar-nav ms-auto ms-md-0 me-3 me-lg-4">
    <li class="nav-item dropdown">
      <a class="nav-link dropdown-toggle" id="navbarDropdown" href="#"
role="button" data-bs-toggle="dropdown" aria-expanded="false"><i class="fas fa-user
fa-fw"></i></a>
      <ul class="dropdown-menu dropdown-menu-end" aria-
labelledby="navbarDropdown">
        <li><a class="dropdown-item" href="#!">Configuración</a></li>
        <li><a class="dropdown-item" href="#!">Registro de Actividad</a></li>
        {% if user.is_staff or user.is_superuser %}
          <li><a class="dropdown-item" href="{% url 'grupo_listar'
%}">Grupos</a></li>
          <li><a class="dropdown-item" href="{% url 'usuario_listar'

```

```
%}">Usuarios</a></li>
    {% endif %}
    <li>
        <hr class="dropdown-divider" />
    </li>
    <li><a class="dropdown-item" href="{% url 'cerrar_sesion' %}">Cerrar
Sesión</a></li>
    </ul>
</li>
</ul>

</nav>
```

neumatic\templates\sidebar.html, cuyo contenido es:

```

<div id="layoutSidenav_nav">
    {% comment %} <nav class="sb-sidenav accordion sb-sidenav-light"
id="sidenavAccordion"> {% endcomment %}
    <nav class="sb-sidenav accordion sb-sidenav-blue" id="sidenavAccordion">
        <div class="sb-sidenav-menu">
            <div class="nav">
                <div class="sb-sidenav-menu-heading">Core</div>

                <a class="nav-link" href="{% url 'home' %}">
                    <div class="sb-nav-link-icon"><i class="fas fa-tachometer-alt">
</i></div>

                Panel
            </a>

            <div class="sb-sidenav-menu-heading">Ventas</div>

            <!-- Opciones del Módulo Ventas Start -->

            <!-- Archivos Start -->
            <a class="nav-link collapsed" href="#" data-bs-toggle="collapse" data-
bs-target="#collapseFiles" aria-expanded="false" aria-controls="collapseFiles">
                <div class="sb-nav-link-icon"><i class="fas fa-book-open"></i>
</div>

                Archivos
                <div class="sb-sidenav-collapse-arrow"><i class="fas fa-angle-
down"></i></div>
            </a>

            <div class="collapse" id="collapseFiles" aria-labelledby="headingTwo"
data-bs-parent="#sidenavAccordion">
                <nav class="sb-sidenav-menu-nested nav accordion"
id="sidenavAccordionPages">

                    <a class="nav-link collapsed" href="#" data-bs-
toggle="collapse" data-bs-target="#pagesCollapseAuth" aria-expanded="false" aria-
controls="pagesCollapseAuth">

                        Catálogos
                        <div class="sb-sidenav-collapse-arrow"><i class="fas fa-
angle-down"></i></div>
                    </a>
                    <div class="collapse" id="pagesCollapseAuth" aria-
labelledby="headingOne" data-bs-parent="#sidenavAccordionPages">
                        <nav class="sb-sidenav-menu-nested nav">
                            <a class="nav-link" href="#">Clientes</a>
                            <a class="nav-link" href="#">Proveedores</a>
                            <a class="nav-link" href="#">Productos</a>
                            <a class="nav-link" href="#">Vendedores</a>
                            <a class="nav-link" href="#">Empresa</a>
                            <a class="nav-link" href="#">Sucursales</a>
                            <a class="nav-link" href="#">Parámetros</a>

```

```

        <a class="nav-link" href="#">Números</a>

    </nav>
</div>

    <a class="nav-link collapsed" href="#" data-bs-
toggle="collapse" data-bs-target="#pagesCollapseError" aria-expanded="false" aria-
controls="pagesCollapseError">
        Tablas
        <div class="sb-sidenav-collapse-arrow"><i class="fas fa-
angle-down"></i></div>
    </a>
    <div class="collapse" id="pagesCollapseError" aria-
labelledby="headingOne" data-bs-parent="#sidenavAccordionPages">
        <nav class="sb-sidenav-menu-nested nav">
            <a class="nav-link" href="{% url 'actividad_list'
%}">Actividad</a>

            <a class="nav-link" href="#">Prod. Depósito</a>
            <a class="nav-link" href="#">Prod. Familia</a>
            <a class="nav-link" href="#">Prod. Marca</a>
            <a class="nav-link" href="#">Prod. Modelo</a>
            <a class="nav-link" href="#">Prod. Mínimo</a>
            <a class="nav-link" href="#">Prod. Stock</a>
            <a class="nav-link" href="#">Prod. Estado</a>
            <a class="nav-link" href="#">Comp. Venta</a>
            <a class="nav-link" href="#">Comp. Compra</a>
            <a class="nav-link" href="#">Moneda</a>
            <a class="nav-link" href="#">Provincia</a>
            <a class="nav-link" href="#">Localidad</a>
            <a class="nav-link" href="#">Tipo Documento</a>
            <a class="nav-link" href="#">Tipo Iva</a>
            <a class="nav-link" href="#">Tipo Percepción</a>
            <a class="nav-link" href="#">Tipo Retención</a>
            <a class="nav-link" href="#">Operario</a>
        </nav>
    </div>
</nav>
</div>
<!-- Archivos End -->

<!-- Procesos Start -->
    <a class="nav-link collapsed" href="#" data-bs-toggle="collapse" data-
bs-target="#collapseProcesses" aria-expanded="false" aria-
controls="collapseProcesses">
        <div class="sb-nav-link-icon"><i class="fas fa-book-open"></i>
    </div>

        Procesos
        <div class="sb-sidenav-collapse-arrow"><i class="fas fa-angle-
down"></i></div>
    </a>

```



```

        <div class="collapse" id="collapseProcesses" aria-
labelledby="headingTwo" data-bs-parent="#sidenavAccordion">
            <nav class="sb-sidenav-menu-nested nav accordion"
id="sidenavAccordionPages">

                <a class="nav-link collapsed" href="#" data-bs-
toggle="collapse" data-bs-target="#pagesCollapseAuth" aria-expanded="false" aria-
controls="pagesCollapseAuth">
                    Ventas
                    <div class="sb-sidenav-collapse-arrow"><i class="fas fa-
angle-down"></i></div>
                </a>
                <div class="collapse" id="pagesCollapseAuth" aria-
labelledby="headingOne" data-bs-parent="#sidenavAccordionPages">
                    <nav class="sb-sidenav-menu-nested nav">
                        <a class="nav-link" href="#">Doc. Venta</a>
                        <a class="nav-link" href="#">Anular DV</a>
                    </nav>
                </div>

                <a class="nav-link collapsed" href="#" data-bs-
toggle="collapse" data-bs-target="#pagesCollapseAuth" aria-expanded="false" aria-
controls="pagesCollapseAuth">
                    Compras
                    <div class="sb-sidenav-collapse-arrow"><i class="fas fa-
angle-down"></i></div>
                </a>
                <div class="collapse" id="pagesCollapseAuth" aria-
labelledby="headingOne" data-bs-parent="#sidenavAccordionPages">
                    <nav class="sb-sidenav-menu-nested nav">
                        <a class="nav-link" href="#">Doc. Compra</a>
                        <a class="nav-link" href="#">Anular DC</a>
                    </nav>
                </div>

            </nav>
        </div>
        <!-- Procesos End -->

        <!-- Informes Start -->
        <a class="nav-link collapsed" href="#" data-bs-toggle="collapse" data-
bs-target="#collapseReports" aria-expanded="false" aria-controls="collapseReports">
            <div class="sb-nav-link-icon"><i class="fas fa-book-open"></i>
        </div>

        Informes
        <div class="sb-sidenav-collapse-arrow"><i class="fas fa-angle-
down"></i></div>
    </a>
    <!-- Informes End -->

```

```

        <!-- Opciones Start -->
        <a class="nav-link collapsed" href="#" data-bs-toggle="collapse" data-
bs-target="#collapseOptions" aria-expanded="false" aria-controls="collapseOptions">
            <div class="sb-nav-link-icon"><i class="fas fa-book-open"></i>
</div>

            Opciones
            <div class="sb-sidenav-collapse-arrow"><i class="fas fa-angle-
down"></i></div>
        </a>
        <!-- Opciones End -->

        <!-- Opciones del Módulo Ventas Start -->

        <!-- Opciones de Estadísticas Start -->
        <div class="sb-sidenav-menu-heading">Estadísticas</div>

        <a class="nav-link" href="charts.html">
            <div class="sb-nav-link-icon"><i class="fas fa-chart-area"></i>
</div>

            Gráficos
        </a>
        <a class="nav-link" href="tables.html">
            <div class="sb-nav-link-icon"><i class="fas fa-table"></i></div>
            Tablas
        </a>
        <!-- Opciones de Estadísticas End -->
    </div>
</div>
<div class="sb-sidenav-footer">
    <div class="small">Logged in as:</div>
    Start Bootstrap
</div>
</nav>
</div>

```