

# PYTHON

PROYECTO FINAL

Basilio Montes Castaño



## Basilio Montes Castaño

RESPONSABLE DEL DESARROLLO DE LA APLICACIÓN

**Fecha:** 27, Octubre, 2022



## DESARROLLO

1. Nos piden tener inventariado todos sus productos y cuáles son sus cantidades en el almacén, de tal forma, que cuando el stock esté al 90% nos avise de pedir al proveedor.
2. En la aplicación web sería ideal tener dos tipos de acceso, uno para clientes y otro para proveedores. Además, un usuario administrador que tenga acceso a ambos.
3. Necesitaremos para nuestros clientes unas gráficas de ventas y para nuestros proveedores unas gráficas de compras. Para nosotros, tendremos unas gráficas comparativas, para saber lo que vendemos y los beneficios que sacamos. También se podrá buscar una alternativa para las gráficas, calculando unas estadísticas de ventas y compras y mostrando dichos resultados.
4. Todos los productos deben tener una descripción del producto, así como lo que hay en el almacén, su precio, lugar donde se encuentra, etc. Aquí podéis tomaros licencias sobre la información extra que añadir, como número de referencia, colores...
5. Para los proveedores, debemos tener almacenados todos los datos de contacto (nombre de empresa, teléfono, dirección, cif...), facturación, precios de sus productos, porcentaje de descuento, IVA, etc.
6. Debemos elaborar la aplicación web de la forma más sencilla para el usuario y lo más práctica para nosotros en su manejo y obtención de datos importante para la empresa. Hay que tener en cuenta la Experiencia del Usuario la cual se caracteriza por sencillez, claridad, intuición.

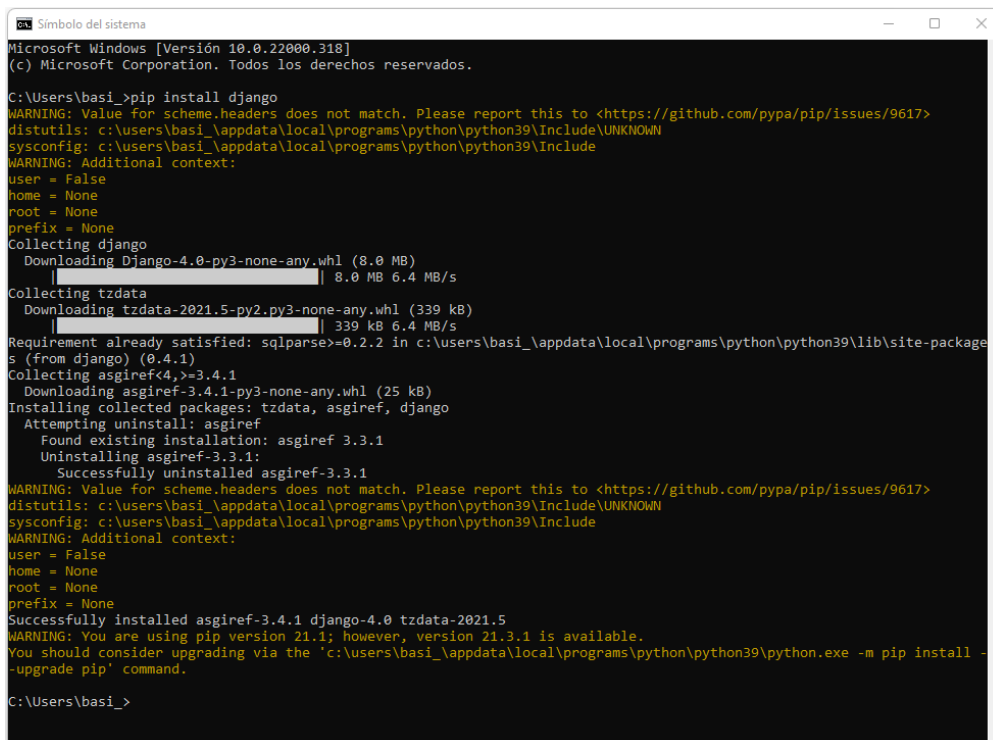
# Proyecto final Django Inventario suministros informáticos

## 1.- Instalación Django

Django es un framework open source de Python para el desarrollo web. Su estructura se basa en el *Model Template View*, las vistas serían el *Template*, para el controlador se usa el *view* y para los datos se utiliza el *model*. Entre sus puntos fuertes destaca por:

- Versatilidad, puede ser usado para construir casi cualquier tipo de sitio web — desde sistemas manejadores de contenidos y wikis, hasta redes sociales y sitios de noticias funciona con cualquier framework en el lado del cliente.
- Seguridad, proporciona una manera segura de administrar cuentas de usuario y contraseñas.
- Escalable, basado en la arquitectura *shared-nothing* (cada parte de la arquitectura es independiente de las otras, y por lo tanto puede ser reemplazado o cambiado si es necesario)
- Portable, está escrito en Python por lo que se puede ejecutar en muchas plataformas, no está sujeto a ninguna en particular y puede ejecutar sus aplicaciones en muchas distribuciones de Linux, Windows y Mac OS X.

Para poder instalar Django en nuestro equipo primero de manera local (También se podría con entornos virtuales) se instala Python. Una vez instalado, abrimos la consola, escribimos *pip install django*.



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.22000.318]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\basi>pip install django
WARNING: Value for scheme.headers does not match. Please report this to <https://github.com/pypa/pip/issues/9617>
distutils: c:\users\basi\appdata\local\programs\python\python39\Include\UNKNOWN
sysconfig: c:\users\basi\appdata\local\programs\python\python39\Include
WARNING: Additional context:
user = False
home = None
root = None
prefix = None
Collecting django
  Downloading Django-4.0-py3-none-any.whl (8.0 MB)
    |#####| 8.0 MB 6.4 MB/s
Collecting tzdata
  Downloading tzdata-2021.5-py2.py3-none-any.whl (339 kB)
    |#####| 339 kB 6.4 MB/s
Requirement already satisfied: sqlparse>=0.2.2 in c:\users\basi\appdata\local\programs\python\python39\lib\site-packages
 (from django) (0.4.1)
Collecting asgiref<4,>=3.4.1
  Downloading asgiref-3.4.1-py3-none-any.whl (25 kB)
Installing collected packages: tzdata, asgiref, django
  Attempting uninstall: asgiref
    Found existing installation: asgiref 3.3.1
    Uninstalling asgiref-3.3.1:
      Successfully uninstalled asgiref-3.3.1
WARNING: Value for scheme.headers does not match. Please report this to <https://github.com/pypa/pip/issues/9617>
distutils: c:\users\basi\appdata\local\programs\python\python39\Include\UNKNOWN
sysconfig: c:\users\basi\appdata\local\programs\python\python39\Include
WARNING: Additional context:
user = False
home = None
root = None
prefix = None
Successfully installed asgiref-3.4.1 django-4.0 tzdata-2021.5
WARNING: You are using pip version 21.1; however, version 21.3.1 is available.
You should consider upgrading via the 'c:\users\basi\appdata\local\programs\python\python39\python.exe -m pip install --
upgrade pip' command.

C:\Users\basi>
```

Para comprobar que se ha instalado y comprobar qué versión se ha instalado, usaremos Python en consola.

```
Selecciónar Símbolo del sistema - python
Microsoft Windows [Versión 10.0.22000.318]
(c) Microsoft Corporation. Todos los derechos reservados.

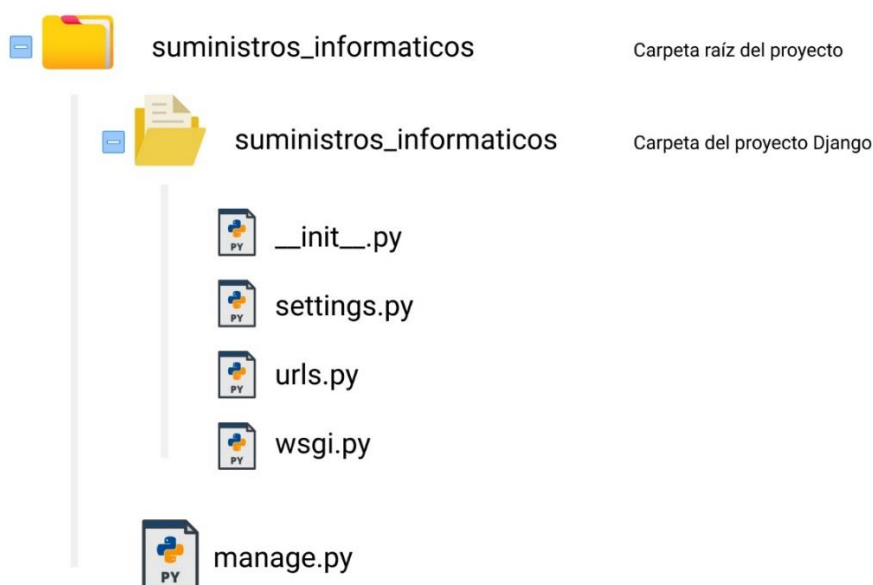
C:\Users\basi_>python
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import django
>>> django.VERSION
(4, 0, 0, 'final', 0)
>>>
```

(4, 0, 0, 'final', 0)

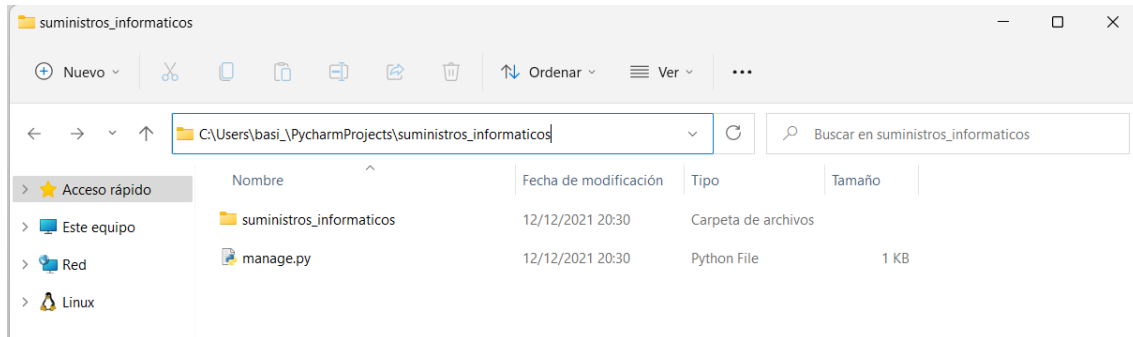
Para crear un nuevo proyecto Django, se ejecuta el siguiente comando: *django-admin startproject myproject*. Previamente, navegamos hacia dónde queremos alojar nuestro proyecto con “cd (Directorio)”

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.22000.318]
(c) Microsoft Corporation. Todos los derechos reservados.

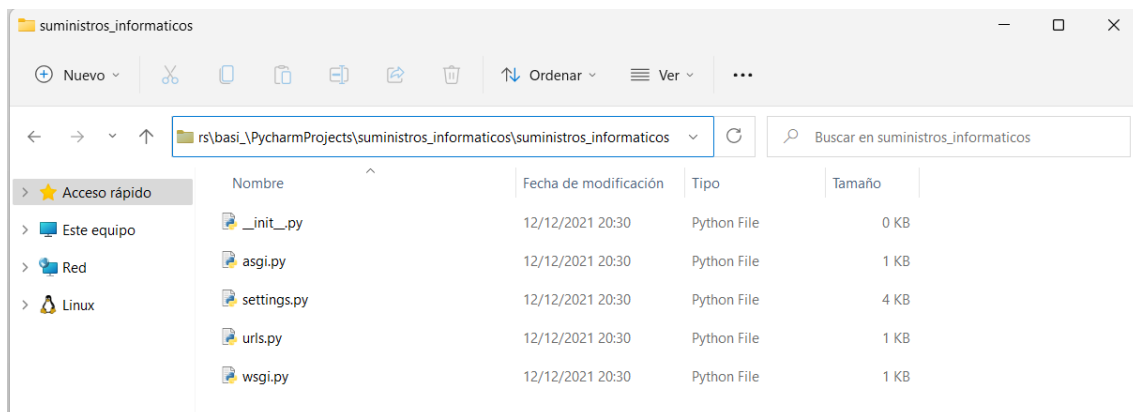
C:\Users\basi_>cd C:\Users\basi_\PycharmProjects
C:\Users\basi_\PycharmProjects>django-admin startproject suministros_informaticos
C:\Users\basi_\PycharmProjects>
```



suministros\_informaticos/



suministros\_informaticos/suministros\_informaticos



La estructura inicial de nuestro proyecto contiene:

- **manage.py** (Directorio raíz): Se usa para ejecutar comandos de administración, como migraciones, crear usuarios, etc...
- **\_\_init\_\_.py**: Este archivo vacío le dice a Python que esta carpeta es un paquete.
- **settings.py**: Contiene la configuración de todo el proyecto.
- **urls.py**: aquí se mapean las rutas y caminos (paths) en nuestro proyecto.
- **wsgi.py**: Este archivo es simplemente una interfaz de puerta de enlace usada para despliegues.
- **asgi.py**: Sucesor de wsgi, es un interfaz del servidor asíncrono.

Django viene con un servidor ya instalado, no tenemos que instalar nada más para ejecutar nuestro proyecto en local. Para comprobarlo ejecutamos *python manage.py runserver*

```
Símbolo del sistema - python manage.py runserver
Microsoft Windows [Versión 10.0.22000.318]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\basi_>cd C:\Users\basi_\PycharmProjects\suministros_informaticos

C:\Users\basi_\PycharmProjects\suministros_informaticos>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
December 18, 2021 - 18:37:57
Django version 4.0, using settings 'suministros_informaticos.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Ahora en un navegador, vamos a la siguiente dirección: <http://127.0.0.1:8000> y ya tendríamos Django instalado y funcionando.

django

View [release notes](#) for Django 4.0

The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.



**Django Documentation**  
Topics, references, & how-to's



**Tutorial: A Polling App**  
Get started with Django



**Django Community**  
Connect, get help, or contribute

Para crear nuestra primera aplicación, vamos al directorio donde está el archivo `manage.py` y lanzamos el siguiente comando: `django-admin startapp suministros_informaticos_app`

En nuestro directorio raíz se ha creado una base de datos por defecto `db.sqlite3`

| C:\Users\basi_\PycharmProjects\suministros_informaticos |  |  |  |  | Buscar en sum         |        |
|---|--|--|--|--|-----------------------|--------|
|   |  |  |  |  |                       |        |
| Nombre  |  |  |  |  | Fecha de modificación | Tamaño |
| suministros_informaticos                                |  |  |  |  | 18/12/2021 18:37      |        |
| suministros_informaticos_app                            |  |  |  |  | 18/12/2021 19:17      |        |
| db.sqlite3  |  |  |  |  | 18/12/2021 18:37      | 0 KB   |
| manage.py   |  |  |  |  | 12/12/2021 20:30      | 1 KB   |

Dentro de la aplicación que acabamos de crear

| C:\Users\basi_\PycharmProjects\suministros_informaticos\suministros_informaticos_app |  |  |  |  | Buscar                |        |
|--|--|--|--|--|-----------------------|--------|
|  |  |  |  |  |                       |        |
| Nombre   |  |  |  |  | Fecha de modificación | Tamaño |
| migrations   |  |  |  |  | 18/12/2021 19:17      |        |
| __init__.py  |  |  |  |  | 18/12/2021 19:17      | 0 KB   |
| admin.py   |  |  |  |  | 18/12/2021 19:17      | 1 KB   |
| apps.py  |  |  |  |  | 18/12/2021 19:17      | 1 KB   |
| models.py  |  |  |  |  | 18/12/2021 19:17      | 1 KB   |
| tests.py   |  |  |  |  | 18/12/2021 19:17      | 1 KB   |
| views.py   |  |  |  |  | 18/12/2021 19:17      | 1 KB   |

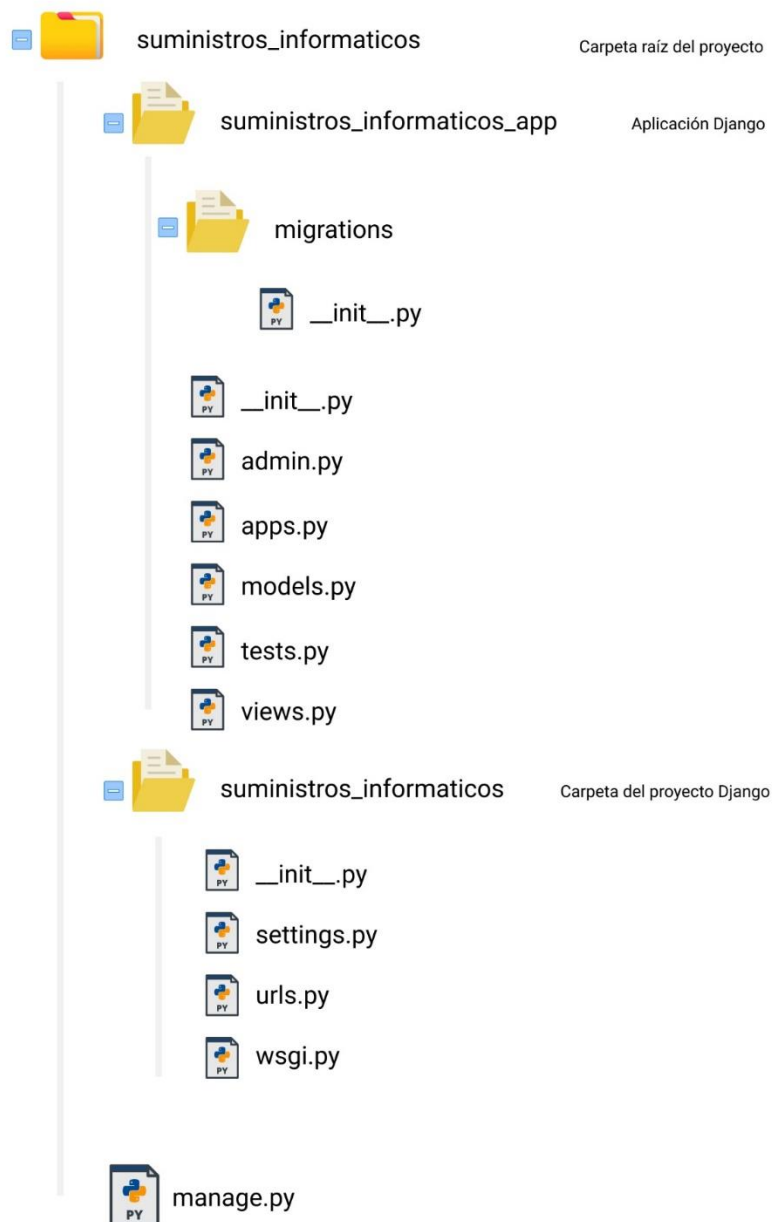
- **migrations:** aquí Django almacena algunos archivos para mantener el registro de los cambios que creas en el archivo `models.py`, para manteniéndolos sincronizados.
- **admin.py:** este es un archivo de configuración para una aplicación compilada de Django llamada Django Admin.
- **apps.py:** este es un archivo de configuración de la aplicación en cuestión.
- **models.py:** aquí es donde definimos las entidades de nuestra aplicación web. Los modelos son traducidos automáticamente por Django a tablas de base de datos.
- **tests.py:** este archivo es utilizado para escribir pruebas unitarias para la aplicación.





- **views.py:** este es un archivo donde manejamos el ciclo de solicitudes/respuestas de nuestra aplicación web. Ahora que hemos creado nuestra primera aplicación, vamos a configurarla para usarla.

De esta manera, nos queda una estructura de fichero así:



## Primera vista con Django

Para crear nuestra primera vista con Django, nos iremos al fichero `views.py` que tenemos en nuestro directorio `suministros_informaticos/suministros_informaticos_app` con lo siguiente:

```
suministros_informaticos_app/views.py

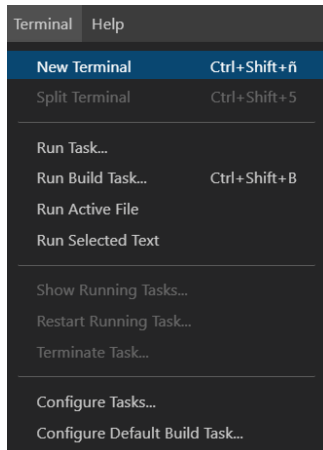
1 # Create your views here.
2
3 from django.http import HttpResponse
4
5 def saludo(request):
6     return HttpResponse('Hola, esta es nuestra primera vista para el proyecto
    final de Tokio School! 🍷')
```

Para el fichero `suministros_informaticos/urls.py` importamos el fichero que acabamos de editar y añadimos un path al que ir en cuanto se ejecute la función escrita en el `suministros_informaticos_app/views.py`

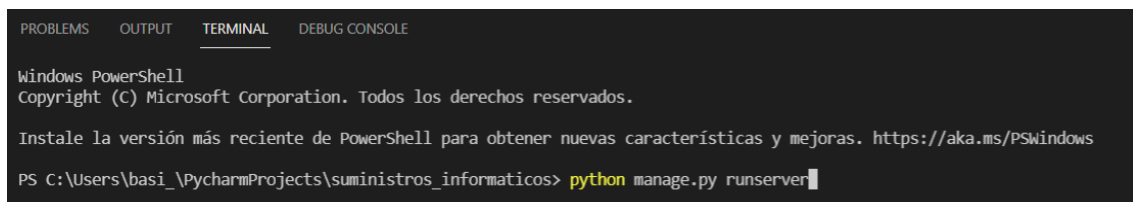
```
suministros_informaticos/urls.py

1 """suministros_informaticos URL Configuration
2
3 The 'urlpatterns' list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/4.0/topics/http/urls/
5 Examples:
6 Function views
7     1. Add an import:  from my_app import views
8     2. Add a URL to urlpatterns:  path('', views.home, name='home')
9 Class-based views
10    1. Add an import:  from other_app.views import Home
11    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
12 Including another URLconf
13    1. Import the include() function: from django.urls import include, path
14    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path
18
19 #Importamos el fichero de views.py
20 from suministros_informaticos_app.views import saludo
21
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24     path('saludo/', saludo),
25 ]
```

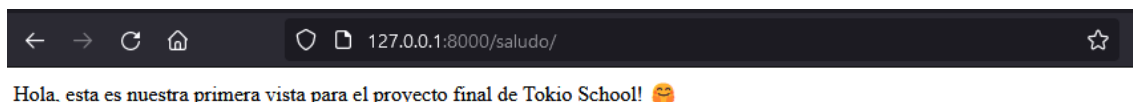
Abrimos una terminal en Visual Studio Code



Ejecutamos *python manage.py runserver*



Este es el resultado en el navegador:



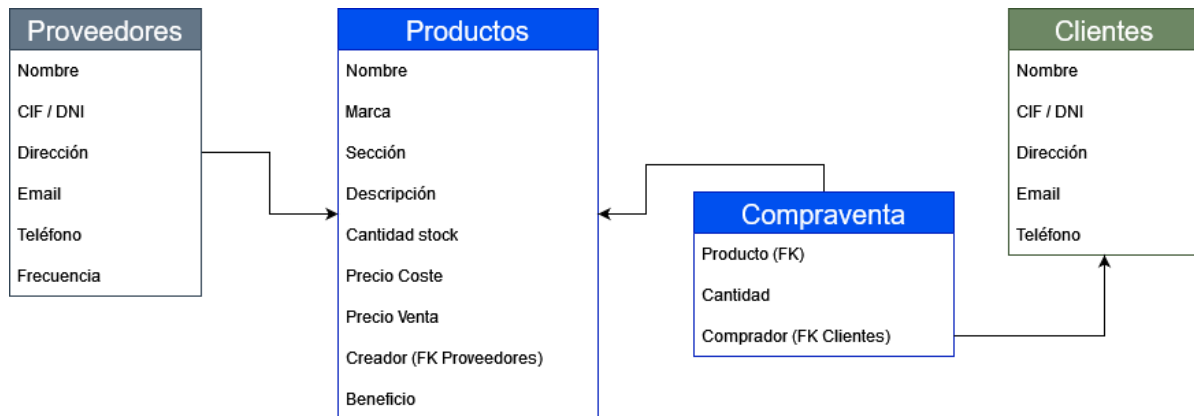
## 2.- Modelo de datos

Para nuestro proyecto, partiremos de tres entidades principales. El producto que se vende es la piedra angular de nuestro modelo, por un lado, están los proveedores que proporcionan ese producto. En el otro lado, están los clientes que adquieren el producto.

Por lo tanto, tendríamos tres tablas centrales, proveedores, productos y clientes. Para conectarlas, vamos a crear una tabla de compraventas donde quedará registrado la venta que hace el proveedor y la compra que almacenaría transacción del producto por parte del cliente.

Cuando hablamos sobre el modelo relacional del proyecto, comprobamos que el producto en sí es el eje entre proveedores y clientes. Un proveedor realiza una transacción de venta del producto o productos, por lo que un proveedor puede realizar muchas ventas y una venta puede tener más de un producto en cuanto a cantidad. Ocurre la misma situación para los clientes que pueden realizar una o varias compras.

Así quedaría el modelo relacional:



Para empezar a construir nuestro modelo de datos, tenemos que editar *models.py*. A continuación, un ejemplo cómo crear una tabla:

```
suministros_informaticos_app/models.py

1 from django.db import models
2 from django.contrib.auth.models import User
3
4 # Create your models here.
5
6 #Tabla de Clientes
7 class Clientes(models.Model):
8     clientes_perfil = models.OneToOneField(User, on_delete=models.CASCADE)
9     nombre = models.CharField(max_length=30)
10    cif_dni = models.CharField(max_length=9)
11    direccion = models.CharField(max_length=50)
12    cp = models.IntegerField(default=0)
13    email = models.EmailField(blank=True, null=True)
14    tfno = models.IntegerField(default=0)
15
16    esCliente = models.BooleanField(default=True)
17    esProveedor = models.BooleanField(default=False)
18
19    def __str__(self):
20        return self.nombre
```

En el fichero *suministros\_informaticos/settings.py* registramos la aplicación de Django, en la lista *Installed\_apps*:

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

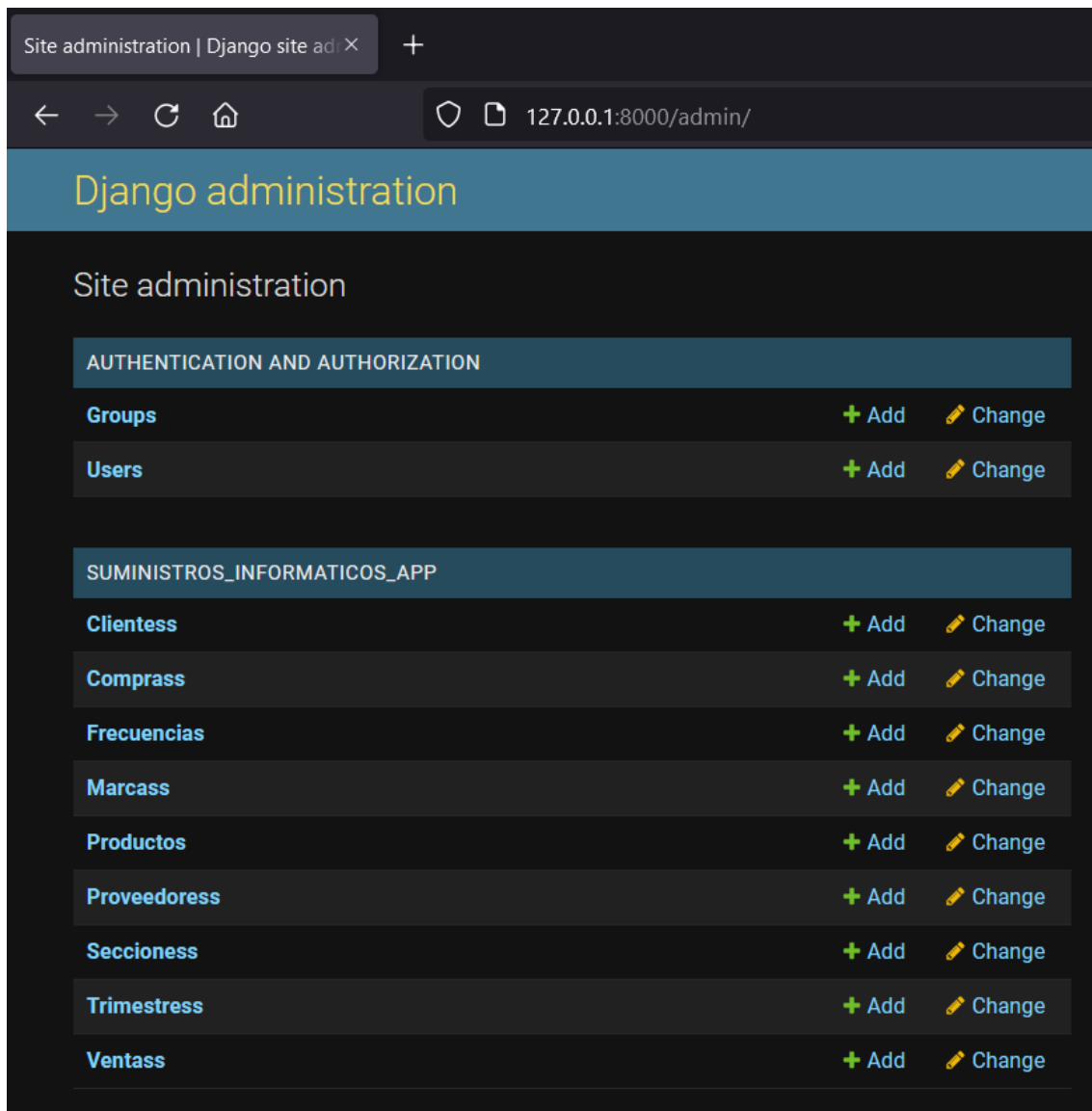
Para registrar el modelo de datos, en el fichero *suministros\_informaticos\_app/admin.py* se escribirá el siguiente código, primero se importa la tabla Clientes, se añade algunas funciones al propio Admin de Django, como qué campos deben aparecer, por cuáles se ordenan o campos de búsqueda de valores y por último se registra.

```
suministros_informaticos_app/admin.py  
  
1 from django.contrib import admin  
2 from suministros_informaticos_app.models import Clientes  
3  
4 #Campos que se quieren ver desde el admin  
5 class ClientesAdmin(admin.ModelAdmin):  
6     list_display = ('nombre', 'cif_dni', 'direccion', 'cp', 'email', 'tfno')  
7     ordering = ['nombre']  
8     search_fields = ['nombre', 'cif_dni', 'tfno']  
9  
10 #Se registra el modelo en el admin  
11 admin.site.register(Clientes, ClientesAdmin)
```

Con la consola de Visual Studio Code se puede lanzar el proyecto. Primeramente, se lanza *python manage.py makemigrations* seguidamente de *python manage.py migrate*. Se necesita crear un superusuario para gestionar la administración de la base de datos, para ello, se ejecuta *python manage.py createsuperuser* donde se nos pedirá un nombre de usuario, email (Opcional) y contraseña.

Ya con todos estos comandos ejecutados podremos acceder al Admin del propio Django en el puerto que se quiera, se especifica así (Si no completamos el puerto, tomará el 8000 por defecto):

```
python manage.py runserver 127.0.0.1:8888
```



### 3.- Funciones CRUD

La base de las aplicaciones dinámicas es el paradigma CRUD (Create, Read, Update y Delete), con el que se puede manipular los registros de una base de datos. Se crearán diferentes ficheros HTML para poder ver los resultados del CRUD, dentro de *suministros\_informaticos\_app/templates*

#### Create

Con esta función se creará un nuevo registro en base de datos, también puede usarse para la creación de usuarios que puedan acceder a la aplicación. Para ello, se editarán *urls.py*, *forms.py* y *views.py* que se encuentran dentro del directorio de la app, *suministros\_informaticos\_app*

En el siguiente ejemplo, se va a escribir el código para crear nuevos usuarios proveedores y además crearlos en base de datos.

```
suministros_informaticos_app/urls.py

1 from django.urls import path
2 from suministros_informaticos_app import views
3
4 urlpatterns = [
5
6     path('crear_proveedores/', views.crear_proveedores, name="Crear Proveedores"),
7
8 ]
```

```
suministros_informaticos_app/views.py

1 from suministros_informaticos_app.models import *
2 from django.shortcuts import render, redirect
3 from .forms import Crear_proveedores
4
5 def crear_proveedores(request):
6     if request.method == "POST":
7         form = Crear_proveedores(request.POST)
8         if form.is_valid():
9             form.save()
10
11             return redirect('/')
12     else:
13         form = Crear_proveedores()
14
15     context = {'form': form}
16     return render(request, 'crear_proveedores.html', context)
```



```
suministros_informaticos_app/forms.py

1 from django import forms
2 from .models import *
3 from django.contrib.auth.forms import UserCreationForm, User
4 from django.db import transaction
5
6 #CRUD PROVEEDORES
7 #Crear proveedores como usuarios
8 class Crear_proveedores(UserCreationForm):
9     #Campos de Proveedor
10     nombre = forms.CharField(required=True)
11     cif_dni = forms.CharField(required=True)
12     direccion = forms.CharField(required=True)
13     email = forms.EmailField(required=True, label="email")
14     tfno = forms.IntegerField(required=True, label="tfno")
15     frecuencia = forms.ChoiceField(choices=((1, ('Diaria')), (2, ('Bisemanal')), (3,
16         ('Semanal')), (4, ('Quincenal')), (5, ('Mensual'))))
17
18     class Meta(UserCreationForm.Meta):
19         model = User
20
21     @transaction.atomic
22     def save(self):
23         user = super(Crear_proveedores, self).save(commit=False)
24         user.save()
25         proveedor = Proveedores.objects.create(proveedor_perfil=user)
26         proveedor.nombre = self.cleaned_data.get('nombre')
27         proveedor.cif_dni = self.cleaned_data.get('cif_dni')
28         proveedor.direccion = self.cleaned_data.get('direccion')
29         proveedor.email = self.cleaned_data.get('email')
30         proveedor.tfno = self.cleaned_data.get('tfno')
31         proveedor.frecuencia = self.cleaned_data.get('frecuencia')
32         proveedor.save()
33         return user
```





```
suministros_informaticos_app/templates/crear_proveedores.t  — □ ×

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Crear proveedores</title>
5 </head>
6 <body>
7     <p>Formulario para crear nuevos proveedores</p>
8     <form method="post" style="color: rgb(5, 12, 70);">
9         {% csrf_token %}
10        {{form.as_p}} <br>
11        <button> Añadir proveedor </button>
12    </form>
13 </body>
14 </html>
15
```

## Read

Con la función de leer se consultan los datos que ya están en base de datos. Sería como mostrar un listado de lo que se tiene en BBDD



## Delete

Para eliminar los registros de bases de datos se usará las siguientes líneas de código como ejemplo

```
Title
1 def eliminar_productos(request, id):
2     productos = get_object_or_404(Producto, id=id)
3     try:
4         productos.delete()
5     except:
6         pass
7
8     return redirect("/acceso")
```

## Update

Con la actualización se haría una consulta a la base de datos, se trae los datos actuales y si se quiere modificar algún campo se guardaría sin problema sin crear una entidad nueva. Por ejemplo:

```
Title
1 def editar_productos(request, id):
2     productos = get_object_or_404(Producto, id=id)
3
4     data = {'form': Editar_Productos(instance=productos)}
5
6     if request.method == "POST":
7         form = Editar_Productos(data=request.POST, instance=productos)
8         if form.is_valid():
9             form.save()
10            return redirect("/acceso")
11            data['form'] = form
12
13    return render(request, 'editar_producto.html', data)
```



Para la parte Frontend del proyecto se instalará Bootstrap4<sup>1</sup> en Django con la consola de comandos:

```
pip install django-bootstrap4
```

Para poder gestionar y renderizar el formulario sin problemas se va a usar la librería *django-widget-tweaks*<sup>2</sup>

#### 4- Login/Logout

Se ha añadido hacia dónde nos llevará el login. En el fichero *suministros\_informaticos/settings.py* se dejan estas dos variables (Líneas 138 y 140) de LOGIN\_REDIRECT\_URL y LOGOUT\_REDIRECT\_URL

```
123 # Static files (CSS, JavaScript, Images)
124 # https://docs.djangoproject.com/en/4.0/howto/static-files/
125
126 STATIC_URL = '/static/'
127
128 STATICFILES_DIRS = [ os.path.join(BASE_DIR, 'static') ]
129
130 STATIC_ROOT = os.path.join(BASE_DIR, 'assets')
131
132 # Default primary key field type
133 # https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field
134
135 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
136
137
138 LOGIN_REDIRECT_URL = '/acceso'
139
140 LOGOUT_REDIRECT_URL = '/'
```

---

<sup>1</sup> <https://pypi.org/project/django-bootstrap4/>

<sup>2</sup> <https://pypi.org/project/django-widget-tweaks/>



Para el fichero de urls.py, se carga la librería auth dentro de Django además de crear los paths correspondientes:

```
suministros_informaticos_app/urls.py

1 from django.contrib.auth import views as auth_views
2
3 urlpatterns = [
4     path('login/', auth_views.LoginView.as_view(template_name='login.html'), name='login'),
5     path('logout/', auth_views.LogoutView.as_view()),
6 ]
```

Para esta demo, se ha creado un logo, una paleta que le diese identidad a esta “marca”. También se han creado dos perfiles, uno de proveedor y otro de cliente. Se pueden crear más si se desea con el propio formulario de la web. La web es compatible con la creación de nuevos productos y/o edición, quedará reflejado en la misma web, además de la creación de nuevos perfiles de usuarios.

### Usuarios:

Usuario proveedor: proveedor1

Contraseña proveedor: sevilla11

Usuario proveedor: cliente1

Contraseña proveedor: sevilla11