🛠️ Instrucciones Adaptadas para el Proyecto "Joyería"

Proyecto: Joyería **Autor:** Diego Alberto Cruz Acosta **Lenguaje:** Python **Framework:** Django **Editor:** VS Code **Puerto:** 8475

Configuración Inicial del Entorno

1.- **Crear Carpeta del proyecto:** Crea la carpeta principal con la nomenclatura `UIII_Joyería_8475`.

2.- **Procedimiento para abrir vs Code:** Abre VS Code sobre la carpeta `UIII_Joyería_8475`.

3.- **Procedimiento para abrir la terminal:** Abre la terminal dentro de VS Code (Terminal > Nueva Terminal).

4.- **Procedimiento para crear carpeta entorno virtual ".venv":** Crea la carpeta del entorno virtual llamada `.venv` desde la terminal: `bash python -m venv .venv`

5.- **Procedimiento para activar el entorno virtual:** Activa el entorno virtual. * **Windows (Command Prompt):** `.\.venv\Scripts\activate` * **Windows (PowerShell):** `.\.venv\Scripts\Activate.ps1` * **Linux/macOS:** `source .venv/bin/activate`

6.- **Procedimiento para activar intérprete de python:** Asegúrate de que el intérprete de Python seleccionado en VS Code sea el del entorno virtual (`.venv`).

7.- **Procedimiento para instalar Django:** Instala el framework Django dentro del entorno virtual: `bash pip install django`

Creación del Proyecto y la Aplicación

8.- **Procedimiento para crear proyecto `backend_Joyería` sin duplicar carpeta:** Crea el proyecto Django sin duplicar la carpeta, nombrando el backend como `backend_Joyería`: `bash django-admin startproject backend_Joyería .`

9.- **Procedimiento para ejecutar servidor en el puerto 8475:** Ejecuta el servidor en el puerto **8475** para verificar la instalación: `bash python manage.py runserver 8475`

10.- **Procedimiento para copiar y pegar el link en el navegador:** Copia y pega el link `http://127.0.0.1:8475/` en tu navegador.

11.- **Procedimiento para crear aplicación `app_Joyería`:** Crea la aplicación principal con el nombre `app_Joyería`: `bash python manage.py startapp app_Joyería`

Modelos y Migraciones

12.- **Aquí el modelo `models.py`:** Copia y pega los siguientes modelos en el archivo `app_Joyería/models.py`.

```python
from django.db import models

# =-=-=-=-=-
# ========================================
# MODELO: PROVEEDOR
# ========================================
class Proveedor(models.Model):
    id_proveedor = models.AutoField(primary_key=True)
    nombre = models.CharField(max_length=100)
    apellido = models.CharField(max_length=100)
    direccion = models.CharField(max_length=200)
    telefono = models.CharField(max_length=15)
    correo = models.EmailField()
    tipo_suministro = models.CharField(max_length=100)

    def __str__(self):
        return f"{self.nombre} {self.apellido}"

# ========================================
# MODELO: PRODUCTO
# ========================================
class Producto(models.Model):
    id_producto = models.AutoField(primary_key=True)
    nombre = models.CharField(max_length=100)
    material = models.CharField(max_length=100)
    precio = models.DecimalField(max_digits=10, decimal_places=2)
    tipo = models.CharField(max_length=100)
    stock = models.IntegerField()
    id_proveedor = models.ForeignKey(Proveedor, on_delete=models.CASCADE,
related_name='productos')

    def __str__(self):
        return self.nombre

# ========================================
# MODELO: VENTA
# ========================================
class Venta(models.Model):
    id_venta = models.AutoField(primary_key=True)
    id_cliente = models.IntegerField()
```

```
    id_empleado = models.IntegerField()
    fecha_venta = models.DateField()
    total = models.DecimalField(max_digits=10, decimal_places=2)
    metodo_pago = models.CharField(max_length=50)
    productos = models.ManyToManyField(Producto, related_name='ventas')

    def __str__(self):
        return f"Venta #{self.id_venta} - Total: ${self.total}"
# =-=-=-=-=-
```

12.5.- **Procedimiento para realizar las migraciones:** Genera y aplica las migraciones: `bash python manage.py makemigrations python manage.py migrate`

Desarrollo del CRUD (Proveedor)

13.- **Primero trabajamos con el MODELO: PROVEEDOR.**

14.- **En view de `app_Joyería` crear las funciones con sus códigos correspondientes:** (e.g., `inicio_joyeria`, `agregar_proveedor`, `actualizar_proveedor`, `realizar_actualizacion_proveedor`, `borrar_proveedor`).

15.- **Crear la carpeta "templates" dentro de `app_Joyería`.**

16.- **En la carpeta templates crear los archivos html:** (`base.html`, `header.html`, `navbar.html`, `footer.html`, `inicio.html`).

17.- **En el archivo `base.html` agregar bootstrap para css y js.**

18.- **En el archivo `navbar.html` incluir las opciones:** ("Sistema de Administración Joyería", "Inicio", "Proveedores" (en submenu: Agregar Proveedor, Ver Proveedores, Actualizar Proveedores, Borrar Proveedores), "Productos" (en submenu: Agregar Productos, Ver Productos, Actualizar Productos, Borrar Productos), "Ventas" (en submenu: Agregar Ventas, Ver Ventas, Actualizar Ventas, Borrar Ventas), incluir iconos a las opciones principales, no en los submenu.

19.- **En el archivo `footer.html` incluir derechos de autor, fecha del sistema y "Creado por Diego Alberto Cruz Acosta"** y mantenerla fija al final de la página.

20.- **En el archivo `inicio.html` se usa para colocar información del sistema más una imagen tomada desde la red sobre joyería.**

21.- **Crear la subcarpeta `proveedor` dentro de `app_Joyería/templates`.**

22.- **Crear los archivos html con su codigo correspondientes:** (`agregar_proveedor.html`, `ver_proveedores.html` (mostrar en tabla con los botones ver, editar y borrar), `actualizar_proveedor.html`, `borrar_proveedor.html`) dentro de `app_Joyería/templates/proveedor`.

23.- **No utilizar `forms.py`.**

24.- **Procedimiento para crear el archivo `urls.py` en `app_Joyería`** con el código correspondiente para acceder a las funciones de `views.py` para operaciones de crud en proveedores.

25.- **Procedimiento para agregar `app_Joyería` en `settings.py` de `backend_Joyería`.**

26.- **Realizar las configuraciones correspondiente a `urls.py` de `backend_Joyería` para enlazar con `app_Joyería`.**

27.- **Procedimiento para registrar los modelos en `admin.py` y volver a realizar las migraciones.**

27.- **Por lo pronto solo trabajar con "Proveedor"** dejar pendiente # MODELO: PRODUCTO y # MODELO: VENTA.

28.- **Utilizar colores suaves, atractivos y modernos, el código de las páginas web sencillas.**

28.- **No validar entrada de datos.**

29.- **Al inicio crear la estructura completa de carpetas y archivos.**

30.- **Proyecto totalmente funcional.**

31.- **Finalmente ejecutar servidor en el puerto 8475.**