Proyecto: Florería

Lenguaje

- Python

Framework

- Django

Editor

- VS Code

Procedimientos para el Proyecto

1. Procedimiento para crear la carpeta del Proyecto: UIII_Floreria_0283

1. Abre tu terminal en la ubicación donde deseas crear el proyecto.
2. Ejecuta el siguiente comando para crear la carpeta:

```
mkdir UIII_Floreria_0283
cd UIII_Floreria_0283
```

2. Procedimiento para abrir VS Code sobre la carpeta UIII_Floreria_0283

1. Abre VS Code.
2. En la terminal de VS Code, navega hasta la carpeta del proyecto:

```
cd ruta/del/proyecto/UIII_Floreria_0283
```

3. Luego ejecuta:

```
code .
```

3. Procedimiento para abrir terminal en VS Code

1. Abre VS Code.
2. Ve a "Ver" > "Terminal" o usa el atajo `Ctrl + ``.

4. Procedimiento para crear carpeta entorno virtual .venv desde la terminal de VS Code

1. En la terminal, dentro de la carpeta del proyecto, ejecuta:

```
python -m venv .venv
```

5. Procedimiento para activar el entorno virtual

- **Windows**:

```
.venv\Scripts\activate
```

- **Mac/Linux**:

source .venv/bin/activate

6. Procedimiento para activar el intérprete de Python

1. En VS Code, ve a "Command Palette" (Ctrl + Shift + P).
2. Busca "Python: Select Interpreter" y selecciona el intérprete de tu entorno virtual.

7. Procedimiento para instalar Django

En la terminal de VS Code, ejecuta el siguiente comando:

pip install django

8. Procedimiento para crear el proyecto backend_floreria sin duplicar la carpeta

1. En la terminal, estando en la carpeta raíz del proyecto, ejecuta:

django-admin startproject backend_floreria .

9. Procedimiento para ejecutar el servidor en el puerto 8083

1. Ejecuta el siguiente comando en la terminal:

python manage.py runserver 8083

10. Procedimiento para copiar y pegar el link en el navegador

1. Abre el navegador y pega la siguiente URL:

http://127.0.0.1:8083/

11. Procedimiento para crear la aplicación app_floreria

1. En la terminal, ejecuta el siguiente comando para crear la app:

python manage.py startapp app_floreria

Modelos models.py (modificados para Florería)

```
from django.db import models


# ========================================
# MODELO: PEDIDO
# ========================================
class Pedido(models.Model):
    id_pedido = models.AutoField(primary_key=True)
    fecha_pedido = models.DateField()
    fecha_entrega = models.DateField()
```

```
    total = models.DecimalField(max_digits=10, decimal_places=2)
    estado = models.CharField(max_length=50)
    metodo_pago = models.CharField(max_length=50)
    direccion_envio = models.CharField(max_length=200)
    cliente = models.ForeignKey('Cliente', on_delete=models.CASCADE)  # Relación con Cliente
    producto = models.ForeignKey('Producto', on_delete=models.CASCADE)  # Relación con Producto

    def __str__(self):
        return f"Pedido {self.id_pedido} - {self.estado}"

# ========================================
# MODELO: PRODUCTO
# ========================================
class Producto(models.Model):
    id_producto = models.AutoField(primary_key=True)
    nombre = models.CharField(max_length=100)
    descripcion = models.TextField()
    precio = models.DecimalField(max_digits=10, decimal_places=2)
    stock = models.IntegerField()
    categoria = models.CharField(max_length=50)
    proveedor = models.CharField(max_length=100)
    fecha_agregado = models.DateField()

    def __str__(self):
        return self.nombre

# ========================================
# MODELO: CLIENTE
```

```
# ========================================

class Cliente(models.Model):
    id_cliente = models.AutoField(primary_key=True)
    nombre = models.CharField(max_length=100)
    apellido = models.CharField(max_length=100)
    telefono = models.CharField(max_length=15)
    email = models.EmailField()
    direccion = models.CharField(max_length=200)
    ciudad = models.CharField(max_length=100)
    codigo_postal = models.CharField(max_length=10)


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

Procedimiento para las Migraciones

12.1 Realizar las migraciones:

Ejecuta el siguiente comando para crear las migraciones:

python manage.py makemigrations

Luego, ejecuta:

python manage.py migrate

para aplicar los cambios en la base de datos.

Creación de Vistas

Trabajo con el modelo Pedido:

En el archivo views.py de la aplicación app_floreria, crea las funciones para realizar las operaciones CRUD en Pedido.

Estructura de Carpetas y Archivos

1. Crear la carpeta templates dentro de app_floreria.
2. En la carpeta templates, crea los siguientes archivos HTML:
    - base.html
    - header.html

    - navbar.html
    - footer.html
    - inicio.html
3. En el archivo base.html, agrega Bootstrap para CSS y JS.
4. En el archivo navbar.html, incluye las opciones principales como:
    - "Sistema de Administración Florería"
    - "Inicio"
    - "Pedidos" en submenu (Agregar Pedido, Ver Pedidos, Actualizar Pedido, Borrar Pedido)
    - "Productos" en submenu (Agregar Producto, Ver Productos, Actualizar Producto, Borrar Producto)
    - "Clientes" en submenu (Ver Clientes, Actualizar Cliente, Borrar Cliente)
5. En el archivo footer.html, incluye derechos de autor, fecha del sistema y "Creado por Ing. Ambar Mercado".
6. En el archivo inicio.html, agrega información general sobre la florería y una imagen representativa.
7. Crear la subcarpeta pedido dentro de app_floreria/templates.
8. Crear los archivos HTML dentro de app_floreria/templates/pedido:
    - agregar_pedido.html
    - ver_pedido.html (mostrar en tabla con los botones Ver, Editar y Borrar)
    - actualizar_pedido.html
    - borrar_pedido.html

Nota: No utilices forms.py.

Configuración de URLs

1. Crea el archivo urls.py dentro de la aplicación app_floreria con el código correspondiente para acceder a las funciones de views.py para operaciones CRUD en Pedido.
2. Agrega app_floreria en settings.py de backend_floreria.
3. Configura correctamente el archivo urls.py de backend_floreria para enlazar con app_floreria.
4. Registra los modelos en admin.py y realiza las migraciones nuevamente.

Diseño de la Página

1. Utiliza colores suaves, atractivos y modernos en las páginas web.
2. Valida las entradas de datos solo cuando sea necesario.
3. Al inicio, crea la estructura completa de carpetas y archivos.
4. Finalmente, ejecuta el servidor en el puerto 8083.