

tercera parte

Proyecto: **Costco**

Lenguaje: **Python.**

Framework: **Django.**

Editor: **VS code.**

Carpeta del Proyecto: **UIII_Costco_0599**

Proyecto: **backend_Costco**

aplicación: **app_Costco**

1 Aquí el modelo models.py que ya está creado

```
=====
from django.db import models

class Usuario(models.Model):
    # Campos de Usuario
    nombre_usuario = models.CharField(max_length=100, unique=True)
    email = models.EmailField(unique=True)
    nombre = models.CharField(max_length=100)
    apellido = models.CharField(max_length=100)
    direccion = models.TextField()
    telefono = models.CharField(max_length=20, blank=True, null=True)
    fecha_registro = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.nombre_usuario

class Producto(models.Model):
    # Campos de Producto
    nombre = models.CharField(max_length=200)
    descripcion = models.TextField()
    precio = models.DecimalField(max_digits=10, decimal_places=2)
    stock = models.PositiveIntegerField(default=0)
    categoria = models.CharField(max_length=100, blank=True, null=True)
    codigo_barras = models.CharField(max_length=50, unique=True,
                                     blank=True, null=True)
    imagen_url = models.URLField(blank=True, null=True) # Nuevo campo
    fecha_creacion = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.nombre

class Pedido(models.Model):
    # Relación uno a muchos: Un Usuario puede tener muchos Pedidos
    usuario = models.ForeignKey(Usuario, on_delete=models.CASCADE,
```

```

related_name='pedidos')

# Campos de Pedido
fecha_pedido = models.DateTimeField(auto_now_add=True)
estado_pedido = models.CharField(max_length=50,
default='Pendiente')
direccion_envio = models.TextField()
total_pedido = models.DecimalField(max_digits=10, decimal_places=2,
default=0.00)
metodo_pago = models.CharField(max_length=50)
fecha_entrega_estimada = models.DateField(blank=True, null=True)
numero_seguimiento = models.CharField(max_length=100, blank=True,
null=True)

# Relación muchos a muchos: Un Pedido puede tener muchos Productos
y un Producto puede estar en muchos Pedidos
productos = models.ManyToManyField(Producto,
related_name='pedidos')

def __str__(self):
    return f"Pedido #{self.id} de {self.usuario.nombre_usuario}"
```

=====

2 Procedimiento para realizar las migraciones(makemigrations y migrate.

3 Ahora trabajamos con el MODELO: **Pedido**

4 En view de app_Costco crear las funciones con sus códigos correspondientes (agregar_pedido, actualizar_pedido, realizar_actualizacion_pedido, borrar_pedido)

5 En la opcion de agregar_pedido, la tabla película está relacionada con Usuario y Producto, primero se captura los datos no relacionados del pedido y luego en el campo de usuario debe aparecer un combobox con los usuarios por su nombre y poder seleccionar uno de ellos, en el campo productos debe aparecer un combobox con los productos disponibles por su nombre y poder seleccionarlos

6 Modificar el archivo navbar.html que está dentro de templates,para actualizar la opcion “pedido” en submenu de Pedido(Aregar pedido,ver pedido en forma de tabla, actualizar pedido, borrar pedido)

7 Crear la subcarpeta **pedido** dentro de **app_Costco\templates**.

8 crear los archivos html con su codigo correspondientes de (agregar_pedido.html, ver_pedido.html mostrar en tabla con los botones ver, editar y borrar, actualizar_pedido.html,

borrar_pedido.html) dentro de app_Costco\templates\pedido

- 9 No utilizar forms.py.
- 10 procedimiento para agregar en urls.py en app_Costco con el código correspondiente para acceder a las funciones de views.py para operaciones de crud en pedido.
- 10 procedimiento para registrar los modelos en admin.py y volver a realizar las migraciones.
- 11 trabajando con todos los modelos
- 12 Utilizar colores suaves, atractivos y modernos, el código de las páginas web sencillas para pedido.
- 13 No validar entrada de datos.
- 14 Al inicio crear la estructura completa de carpetas y archivos actualizados, donde se muestra la carpeta pedido dentro de templates con los archivos html correspondientes a las operaciones crud pedido.
- 15 proyecto totalmente funcional.
- 16 finalmente ejecutar servidor en el puerto puerto 8018.