Primera parte
Proyecto: **Pizzeria**.
lenguaje: **Python**.
Framework: **Django**.
Editor: **VS code**.
1 Procedimiento para crear carpeta del Proyecto: **UIII_Pizzeria_0019**
2 procedimiento para abrir vs code sobre la carpeta
UIII_Pizzeria_0019
3 procedimiento para **abrir terminal en vs code**
4 Procedimiento para crear carpeta entorno virtual "**.venv**" desde
terminal de vs code
5 Procedimiento para **activar el entorno virtual**.
6 procedimiento para **activar intérprete de python**.
7 Procedimiento para instalar **Django**
8 procedimiento para crear proyecto **backend_Pizzeria** sin duplicar
carpeta.
9 procedimiento para ejecutar servidor en el **puerto 8019**
10 procedimiento para copiar y pegar el link en el navegador.
11 procedimiento para crear aplicacion **app_Pizzeria**
12 Aqui el modelo models.py
=-=-=-=-=-

```
from django.db import models

# ========================================
# MODELO: Proveedores (Actualizado)
# ========================================
class Proveedores(models.Model):
    # id_proveedor es automático (AutoField)
    nombre_proveedor = models.CharField(max_length=100, unique=True)
    telefono_contacto = models.CharField(max_length=15, blank=True, null=True)
    email_contacto = models.EmailField(max_length=100, blank=True, null=True)
    direccion = models.CharField(max_length=255, blank=True, null=True)
    tipo_producto = models.CharField(max_length=100, blank=True, null=True)
    rfc = models.CharField(max_length=20, blank=True, null=True, unique=True) # RFC
debería ser único si existe
    fecha_registro = models.DateField(auto_now_add=True) # auto_now_add es útil para la
fecha de registro
    activo = models.BooleanField(default=True) # Conservado del modelo original

    def __str__(self):
        return self.nombre_proveedor # Actualizado para que coincida con el nuevo nombre de
campo

# ========================================
# MODELO: Inventario (Nuevo)
# ========================================
class Inventario(models.Model):
    # id_articulo es automático (AutoField)
```

```
    nombre_articulo = models.CharField(max_length=100)
    stock = models.DecimalField(max_digits=10, decimal_places=2, default=0.0)
    unidad = models.CharField(max_length=20) # Ej: 'kg', 'litro', 'pieza'
    fecha_ultima_compra = models.DateField(null=True, blank=True)
    stock_minimo = models.DecimalField(max_digits=10, decimal_places=2, default=0.0)
    costo_unitario = models.DecimalField(max_digits=10, decimal_places=2, default=0.0)

    # Relación: Un artículo de inventario pertenece a UN proveedor
    proveedor = models.ForeignKey(
        Proveedores,
        on_delete=models.SET_NULL, # Si se borra el proveedor, el artículo no se borra, solo
se quita la relación
        null=True,
        blank=True,
        related_name="articulos_inventario",
        db_column="fk_id_proveedor" # Coincide con tu diagrama
    )

    def __str__(self):
        return f"{self.nombre_articulo} ({self.stock} {self.unidad})"

# ========================================
# MODELO: Menu (Nuevo)
# ========================================
class Menu(models.Model):
    # id_producto es automático (AutoField)
    nombre = models.CharField(max_length=100)
    descripcion = models.TextField(blank=True, null=True)
    precio = models.DecimalField(max_digits=10, decimal_places=2)
    categoria = models.CharField(max_length=50) # Ej: 'Bebida', 'Postre', 'Plato Fuerte'
    tamaño = models.CharField(max_length=50, blank=True, null=True) # Ej: 'Chico', 'Grande'
    disponible = models.BooleanField(default=True)

    # Relación (Como solicitaste):
    # Un producto del menú (ej: Hamburguesa) usa VARIOS artículos del inventario (ej: Pan,
Carne, Queso)
    # Y un artículo del inventario (ej: Queso) puede ser usado en VARIOS productos del
menú (ej: Hamburguesa, Nachos)
    articulos = models.ManyToManyField(
        Inventario,
        related_name="productos_menu",
        blank=True # Un producto puede existir sin artículos de inventario definidos
    )

    def __str__(self):
        return f"{self.nombre} - ${self.precio}"
========================================
```

12.5 Procedimiento para realizar las migraciones(makemigrations y

migrate.
13 primero trabajamos con el MODELO: PROVEEDORES
14 En view de **app_Pizzeria** crear las funciones con sus códigos correspondientes (inicio_pizzeria, agregar_proveedor,

actualizar_proveedor, realizar_actualizacion_proveedor, borrar_proveedor)
15 Crear la carpeta "**templates**" dentro de "**app_Pizzeria**".
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 Pizzeria", "Inicio", "proveedores",en submenu de proveedores(Aregar proveedores,ver proveedores, actualizar proveedores, borrar proveedores), "Inventario", en submenu de Inventario(Aregar articulo,ver articulo, actualizar articulo, borrar articulo)
"Menu" en submenu de Menu(Aregar producto,ver producto, actualizar producto, borrar producto), 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 Estudiante Carlos Lozano, Cbtis 128" 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 pizzeria.
21 Crear la subcarpeta carpeta proveedores dentro de app_Pizzeria\templates.
22 crear los archivos html con su codigo correspondientes de (agregar_ proveedores.html, ver_ proveedores.html mostrar en tabla con los botones ver, editar y borrar, actualizar_ proveedores.html, borrar_ proveedores.html)
dentro de **app_Pizzeria\templates\proveedores**.
23 No utilizar forms.py.
24 procedimiento para crear el archivo urls.py en **app_Pizzeria** con el código correspondiente para acceder a las funciones de views.py para operaciones de crud en proveedores.
25 procedimiento para agregar **app_Pizzeria** en settings.py de **backend_Pizzeria**
26 realizar las configuraciones correspondiente a urls.py de **backend_Pizzeria** para enlazar con **app_Pizzeria**

27 procedimiento para registrar los modelos en admin.py **y volver a realizar las migraciones.**
27 por lo pronto solo trabajar con "proveedores" dejar pendiente # MODELO: INVENTARIO y # MODELO: MENÚ
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 **puerto 8019**.