**Primera parte**
Proyecto: **Biblias.**
lenguaje: **Python**.
Framework: **Django.**
Editor: **VS code.**
1 Procedimiento para crear carpeta del Proyecto: **UIII_Biblias_0486**
2 procedimiento para abrir vs code sobre la carpeta
**UIII_Biblias_0486**
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_Biblia** sin duplicar
carpeta.
9 procedimiento para ejecutar servidor **en el puerto 0486**
10 procedimiento para copiar y pegar el link en el navegador.
11 procedimiento para crear aplicacion **app_Biblia**
12 Aqui el modelo models.py
=-=-=-=-=-

```
from django.db import models

from django.db import models

# ====================
# MODELO: CLIENTE_GLOBAL (Corresponde a la tabla CLIENTES_GLOBAL)
# ====================
class ClienteGlobal(models.Model):
    # cliente_id (PK) es automático por defecto en Django
    nombre = models.CharField(max_length=100)
    apellido = models.CharField(max_length=100)
    email = models.CharField(max_length=255, unique=True) # Se usa CharField para email
con unique=True
    fecha_nacimiento = models.DateField(null=True, blank=True)
    pais_residencia = models.CharField(max_length=100)
    codigo_postal = models.CharField(max_length=15, null=True, blank=True)
    telefono = models.CharField(max_length=20, null=True, blank=True)
    # Los campos 'int' del diagrama se mapean a CharField o IntegerField
    # según si requieren operaciones matemáticas o son solo códigos/números largos.

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

# ====================
# MODELO: PEDIDO (Corresponde a la tabla PEDIDOS)
# ====================
class Pedido(models.Model):
```

```
    # pedido_id (PK) es automático por defecto en Django
    fecha_hora = models.DateTimeField(auto_now_add=True) # Se usa auto_now_add para
registrar la fecha y hora de creación
    total_neto = models.DecimalField(max_digits=10, decimal_places=2, default=0)
    metodo_pago = models.CharField(max_length=50)
    estado_pedido = models.CharField(max_length=50, default='Pendiente')
    impuesto_total = models.DecimalField(max_digits=10, decimal_places=2, default=0)
    costo_envio = models.DecimalField(max_digits=10, decimal_places=2, default=0)
    direccion_envio = models.CharField(max_length=255)

    # RELACIÓN 1 a N: Un cliente puede tener varios pedidos.
    cliente = models.ForeignKey(
        ClienteGlobal,
        on_delete=models.CASCADE,
        related_name="pedidos" # 'pedidos' será el nombre para la relación inversa
    )

    def __str__(self):
        return f"Pedido #{self.pk} - Cliente: {self.cliente.apellido}"

# ====================
# MODELO: DETALLE_PEDIDO (Corresponde a la tabla DETALLE_PEDIDO)
# ====================
class DetallePedido(models.Model):
    # detalle_id (PK) es automático por defecto en Django
    cantidad = models.PositiveIntegerField(default=1)
    precio_venta_unitario = models.DecimalField(max_digits=10, decimal_places=2)
    subtotal_detalle = models.DecimalField(max_digits=10, decimal_places=2)
    porcentaje_descuento = models.DecimalField(max_digits=5, decimal_places=2,
default=0)
    impuesto_detalle = models.DecimalField(max_digits=10, decimal_places=2, default=0)
    notas_item = models.CharField(max_length=255, blank=True, null=True)
    numero_linea = models.PositiveIntegerField()
    producto_id = models.IntegerField() # Este campo es un FK a una tabla 'Producto' que no
está en el diagrama, se deja como IntegerField

    # RELACIÓN 1 a N: Un pedido puede tener varios detalles.
    pedido = models.ForeignKey(
        Pedido,
        on_delete=models.CASCADE,
        related_name="detalles" # 'detalles' será el nombre para la relación inversa
    )

    # Puedes definir una clave primaria compuesta si fuera necesario,
    # pero Django usa 'id' (detalle_id) como PK por defecto.

    class Meta:
        verbose_name_plural = "Detalles de Pedido"
```

```
        ordering = ['pedido', 'numero_linea'] # Ordenar por pedido y línea para lógica

    def __str__(self):
        return f"Detalle de Pedido #{self.pedido.pk}, Línea {self.numero_linea}"
```

13 Primero trabajamos con el **MODELO: CLIENTE_GLOBAL**

14 En `views`de `app_Biblias` crear las funciones con sus códigos correspondientes (**inicio_biblias**, **agregar_cliente**, **actualizar_cliente**, **realizar_actualizacion_cliente**, **borrar_cliente**),

15 Crear la carpeta "templates" dentro de "**app_Biblias**".

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 de Biblias**", "Inicio", "**Clientes**", en submenú de Clientes (**Agregar Cliente**, **Ver Clientes**, **Actualizar Cliente**, **Borrar Cliente**), "**Pedidos**" en submenú de Pedidos (**Agregar Pedido**, **Ver Pedidos**, **Actualizar Pedido**, **Borrar Pedido**), "**Detalles**" en submenú de Detalles (**Agregar Detalle**, **Ver Detalles**, **Actualizar Detalle**, **Borrar Detalle**), incluir iconos a las opciones principales, no en los submenú.

19 En el archivo footer.html incluir derechos de autor,fecha del sistema y “Creado por Ing. Jorge Dominguez, 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 Biblias.

21 Crear la subcarpeta carpeta categoria dentro de app_Biblias\templates.

22 Crear los archivos html con su código correspondientes de (**agregar_cliente.html**, **ver_clientes.html** mostrar en tabla con los botones ver, editar y borrar, **actualizar_cliente.html**, **borrar_cliente.html**) dentro de **app_Biblias**`\templates\cliente`.

23 No utilizar forms.py

24 Procedimiento para crear el archivo `urls.py` en **app_Biblias** con el código correspondiente para acceder a las funciones de `views.py` para operaciones de CRUD en **clientes**.

25 Procedimiento para agregar **app_Biblias** en `settings.py` de **backend_Biblias**

26 Realizar las configuraciones correspondiente a `urls.py` de **backend_Biblias** para enlazar con **app_Biblias**

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

27Por lo pronto solo trabajar con "**ClienteGlobal**" dejar pendiente # MODELO: **PEDIDO** y # MODELO: **DETALLE_PEDIDO**

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 0486.