Primera parte
Proyecto: VictorySports.
lenguaje: Python.
Framework: Django.
Editor: VS code.
1 Procedimiento para crear carpeta del Proyecto: UIII_Victorysports_0362
2 procedimiento para abrir vs code sobre la carpeta
UIII_Victorysports_0362
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_Victorysports sin duplicar
carpeta.
9 procedimiento para ejecutar servidor en el puerto 8036
10 procedimiento para copiar y pegar el link en el navegador.
11 procedimiento para crear aplicacion app_victorysports
12 Aqui el modelo models.py

```
from django.db import models

# --- Modelo para Proveedor ---
class Proveedor(models.Model):
    # id_proveedor int pk - Django crea un 'id' autoincremental por defecto,
    # pero si quieres llamarlo explícitamente 'id_proveedor':
    id_proveedor = models.AutoField(primary_key=True)
    nombre_empresa = models.CharField(max_length=150, unique=True)
    telefono_empresa = models.CharField(max_length=15)
    email_empresa = models.EmailField(max_length=100) # Se usa EmailField para validación
    pais_origen = models.CharField(max_length=50)
    contacto_principal = models.CharField(max_length=100)
    fecha_registro = models.DateField(auto_now_add=True) # auto_now_add para la fecha
inicial de registro
    direccion = models.TextField()

    class Meta:
        verbose_name = "Proveedor"
        verbose_name_plural = "Proveedores"

    def __str__(self):
        return self.nombre_empresa

# --- Modelo para Producto ---
```

```
class Producto(models.Model):
    # id_producto int pk - Igual que antes, se define explícitamente si se desea ese nombre
    id_producto = models.AutoField(primary_key=True)
    nombre = models.CharField(max_length=255)
    precio_unitario = models.DecimalField(max_digits=10, decimal_places=2)
    stock = models.IntegerField(default=0)
    marca = models.CharField(max_length=100)
    img_url = models.URLField(max_length=255, blank=True, null=True) # URLField para URLs,
permite vacío
    categoria = models.CharField(max_length=50)
    color = models.CharField(max_length=50)

    class Meta:
        verbose_name = "Producto"
        verbose_name_plural = "Productos"

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

# --- Modelo para ProductoProveedor (Tabla Intermedia/Many-to-Many con Datos Adicionales)
---
class ProductoProveedor(models.Model):
    # id_producto int pk > producto.id_producto
    # ForeignKey al modelo Producto, relacionado con 'id_producto'
    producto = models.ForeignKey(Producto, on_delete=models.CASCADE,
related_name='relaciones_proveedor')

    # id_proveedor int pk > proveedor.id_proveedor
    # ForeignKey al modelo Proveedor, relacionado con 'id_proveedor'
    proveedor = models.ForeignKey(Proveedor, on_delete=models.CASCADE,
related_name='relaciones_producto')

    # Las claves primarias compuestas se gestionan con la restricción unique_together

    precio_compra = models.DecimalField(max_digits=10, decimal_places=2)
    fecha_ultima_compra = models.DateField(null=True, blank=True) # Permite ser NULL si
nunca ha habido compra
    cantidad_comprada = models.IntegerField(default=1)
    referencia_pedido = models.CharField(max_length=50, blank=True, null=True)
    es_proveedor_activo = models.BooleanField(default=True)

    class Meta:
        # Define la clave primaria compuesta (id_producto, id_proveedor)
        # Esto asegura que cada par (producto, proveedor) sea único.
```

```
        unique_together = (('producto', 'proveedor'),)
        verbose_name = "Producto por Proveedor"
        verbose_name_plural = "Productos por Proveedores"

    def __str__(self):
        return f"Relación: {self.producto.nombre} - {self.proveedor.nombre_empresa}"
```

12.5 Procedimiento para realizar las migraciones(makemigrations y migrate.
13 primero trabajamos con el MODELO: CATEGORÍA
14 En view de app_victorysports crear las funciones con sus códigos correspondientes (inicio_victorysports, agregar_proveedor,

actualizar_proveedor, realizar_actualizacion_proveedor, borrar_proveedor)
15 Crear la carpeta “templates” dentro de “app_victorysports”.
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 Cinepolis”, “Inicio”, “categoria”,en submenu de categorias(Aregar Categoria,ver categoria, actualizar categoria, borrar categoria), “Salas” en submenu de Salas(Aregar salas,ver salas, actualizar salas, borrar salas)
“Peliculas” en submenu de Peliculas(Aregar peliculas,ver peliculas, actualizar peliculas, borrar peliculas), 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 Tec. Ricardo Santiago, 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 victorysports.
21 Crear la subcarpeta carpeta categoria dentro de app_victorysports\templates.
22 crear los archivos html con su codigo correspondientes de (agregar_proveedor.html, ver_proveedor.html mostrar en tabla con los botones ver, editar y borrar, actualizar_proveedor.html, borrar_proveedor.html)
dentro de app_victorysports\templates\proveedor.
23 No utilizar forms.py.
24 procedimiento para crear el archivo urls.py en app_victorysports con el código correspondiente para acceder a las funciones de views.py para operaciones de crud en proveedor.
25 procedimiento para agregar app_victorysports en settings.py de

backend_victorysports
26 realizar las configuraciones correspondiente a urls.py de backend_victorysports para enlazar con app_victorysports

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

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 8036.