Primera parte

Proyecto: Tienda de Antigüedades
Lenguaje: Python
Framework: Django
Editor: VS Code

---

1. Procedimiento para crear carpeta del Proyecto:

`UIII_Antiguedades_0395`

2. Procedimiento para abrir VS Code sobre la carpeta

`UIII_Antiguedades_0395`

3. Procedimiento para abrir terminal en VS Code

4. Procedimiento para crear carpeta de 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_Antiguedades`
(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 aplicación

`app_Antiguedades`

---

12. Aquí el modelo `models.py`

```
from django.db import models
from django.core.validators import MinValueValidator,
MaxValueValidator
import json

# ========================================
# MODELO: PROVEEDOR
# ========================================
class Proveedor(models.Model):
    nombre_empresa = models.CharField(max_length=200,
verbose_name="Nombre de la empresa")
    contacto = models.CharField(max_length=100,
verbose_name="Persona de contacto")
    email = models.EmailField(unique=True, verbose_name="Correo
electrónico")
    telefono = models.CharField(max_length=15,
verbose_name="Teléfono", blank=True, null=True)
    direccion = models.TextField(verbose_name="Dirección")
    especialidad = models.CharField(max_length=100,
verbose_name="Especialidad")
    años_experiencia = models.IntegerField(
        validators=[MinValueValidator(1)],
        verbose_name="Años de experiencia"
    )

    def __str__(self):
        return f"{self.nombre_empresa} - {self.contacto}"

    class Meta:
        verbose_name = "Proveedor"
        verbose_name_plural = "Proveedores"
        ordering = ['nombre_empresa']

# ========================================
# MODELO: CLIENTE
# ========================================
class Cliente(models.Model):
    ESTADOS_CIVILES = [
        ('S', 'Soltero/a'),
        ('C', 'Casado/a'),
        ('V', 'Viudo/a'),
```

```
        ('D', 'Divorciado/a'),
    ]

    nombre = models.CharField(max_length=100, verbose_name="Nombre
completo")
    email = models.EmailField(unique=True, verbose_name="Correo
electrónico")
    telefono = models.CharField(max_length=15,
verbose_name="Teléfono", blank=True, null=True)
    direccion = models.TextField(verbose_name="Dirección")
    fecha_registro = models.DateField(auto_now_add=True,
verbose_name="Fecha de registro")
    estado_civil = models.CharField(max_length=1,
choices=ESTADOS_CIVILES, verbose_name="Estado civil")
    es_coleccionista = models.BooleanField(default=False,
verbose_name="Es coleccionista")

    # RELACIÓN M:M con Proveedor
    proveedores_favoritos = models.ManyToManyField(
        Proveedor,
        through='ClienteProveedor',
        related_name='clientes',
        verbose_name="Proveedores favoritos",
        blank=True
    )

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

    class Meta:
        verbose_name = "Cliente"
        verbose_name_plural = "Clientes"
        ordering = ['nombre']

# ========================================
# MODELO: PIEZA DE ANTIGÜEDAD
# ========================================
class PiezaAntiguedad(models.Model):
    ESTADOS_CONSERVACION = [
        ('E', 'Excelente'),
        ('B', 'Buena'),
```

```
        ('R', 'Regular'),
        ('M', 'Mala'),
    ]

    nombre = models.CharField(max_length=200, verbose_name="Nombre
de la pieza")
    descripcion = models.TextField(verbose_name="Descripción
detallada")
    precio = models.DecimalField(max_digits=12, decimal_places=2,
verbose_name="Precio de venta")
    año_fabricacion = models.IntegerField(
        validators=[MinValueValidator(1000),
MaxValueValidator(1950)],
        verbose_name="Año de fabricación"
    )
    estado_conservacion = models.CharField(
        max_length=1,
        choices=ESTADOS_CONSERVACION,
        verbose_name="Estado de conservación"
    )
    fecha_ingreso = models.DateField(auto_now_add=True,
verbose_name="Fecha de ingreso")

    # RELACIÓN 1:M con Proveedor
    proveedor = models.ForeignKey(
        Proveedor,
        on_delete=models.PROTECT,
        related_name='piezas_proveidas',
        verbose_name="Proveedor"
    )

    # RELACIÓN M:M con Cliente (compradores)
    compradores = models.ManyToManyField(
        Cliente,
        through='CompraPieza',
        related_name='piezas_compradas',
        verbose_name="Clientes compradores",
        blank=True
    )

    def __str__(self):
```

```
        return f"{self.nombre} - {self.año_fabricacion}
(${self.precio})"

    class Meta:
        verbose_name = "Pieza de antigüedad"
        verbose_name_plural = "Piezas de antigüedad"
        ordering = ['-fecha_ingreso']

# ========================================
# MODELOS INTERMEDIOS
# ========================================
class ClienteProveedor(models.Model):
    TIPOS_CLIENTE = [
        ('FRECUENTE', 'Cliente Frecuente'),
        ('OCASIONAL', 'Cliente Ocasional'),
        ('MAYORISTA', 'Mayorista'),
        ('COLECCIONISTA', 'Coleccionista'),
    ]

    cliente = models.ForeignKey(Cliente, on_delete=models.CASCADE)
    proveedor = models.ForeignKey(Proveedor,
on_delete=models.CASCADE)
    fecha_relacion = models.DateTimeField(auto_now_add=True,
verbose_name="Fecha de relación")
    tipo_cliente = models.CharField(max_length=50,
choices=TIPOS_CLIENTE, verbose_name="Tipo de cliente")

    class Meta:
        verbose_name = "Relación Cliente-Proveedor"
        verbose_name_plural = "Relaciones Cliente-Proveedor"
        unique_together = ['cliente', 'proveedor']

class CompraPieza(models.Model):
    cliente = models.ForeignKey(Cliente, on_delete=models.PROTECT)
    pieza = models.ForeignKey(PiezaAntiguedad,
on_delete=models.PROTECT)
    fecha_compra = models.DateTimeField(auto_now_add=True,
verbose_name="Fecha de compra")
    precio_compra = models.DecimalField(max_digits=12,
decimal_places=2, verbose_name="Precio de compra")
```

```
    cantidad = models.IntegerField(default=1,
validators=[MinValueValidator(1)], verbose_name="Cantidad")

    class Meta:
        verbose_name = "Compra de Pieza"
        verbose_name_plural = "Compras de Piezas"
        unique_together = ['cliente', 'pieza']
```

12.5 Procedimiento para realizar las migraciones

Ejecutar en terminal:

```
python manage.py makemigrations
python manage.py migrate
```

---

13. Primero trabajamos con el MODELO: PROVEEDOR

14. En `views.py` de `app_Antiguedades`, crear las funciones con sus códigos correspondientes:

`inicio_antiguedades`, `agregar_proveedor`, `actualizar_proveedor`, `realizar_actualizacion_proveedor`, `borrar_proveedor`.

---

15. Crear la carpeta templates dentro de `app_Antiguedades`.

16. Dentro de `templates`, crear los archivos:

`base.html`, `header.html`, `navbar.html`, `footer.html`, `inicio.html`.

17. En `base.html`, incluir Bootstrap para CSS y JS.

18. En `navbar.html`, incluir las opciones:

- **Sistema de Administración Tienda de Antigüedades**
- “Inicio”

- “Proveedores” → (Agregar, Ver, Actualizar, Borrar)
- “Clientes” → (Agregar, Ver, Actualizar, Borrar)
- “Piezas de Antigüedad” → (Agregar, Ver, Actualizar, Borrar)

(Agregar íconos en las secciones principales, no en los submenús).

---

19. En `footer.html`, incluir derechos de autor, fecha del sistema y texto:

“Creado por Ing. Abdiel Vega, Cbtis 128”,
manteniendo el footer fijo al final de la página.

---

20. En `inicio.html`, colocar información del sistema y una imagen de una tienda o colección de antigüedades (desde internet).

---

21. Crear la subcarpeta `proveedor` dentro de

`app_Antiguedades/templates`.

22. Dentro de esa carpeta crear los archivos:

- `agregar_proveedor.html`
- `ver_proveedores.html` (mostrar tabla con botones **ver**, **editar** y **borrar**)
- `actualizar_proveedor.html`
- `borrar_proveedor.html`

---

23. No utilizar `forms.py`.

24. Crear `urls.py` en `app_Antiguedades` con las rutas para las funciones CRUD de proveedores.

25. Agregar `app_Antiguedades` en `settings.py` del proyecto `backend_Antiguedades`.

26. Configurar el `urls.py` principal de `backend_Antiguedades` para enlazar con `app_Antiguedades`.

27. Registrar los modelos en `admin.py` y volver a realizar las migraciones.

28. Por ahora, trabajar solo con el modelo Proveedor.

Dejar pendiente **Cliente** y **PiezaAntiguedad**.

29. Utilizar colores suaves, atractivos y modernos en el diseño.

30. No validar entradas de datos.

31. Crear toda la estructura de carpetas y archivos antes de comenzar.

32. Proyecto totalmente funcional.

33. Finalmente, ejecutar el servidor en el puerto 8042:

```
python manage.py runserver 8042
```