

segunda parte
Proyecto: **mascotas**.
lenguaje: **Python**.
Framework: **Django**.
Editor: **VS code**.
carpeta del Proyecto: **UIII_mascotas_0237**
proyecto: **backend_mascotas**
aplicacion: **app_tienda**

Aqui el modelo [models.py](#) ya esta crrado

==--==--==

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

```
# =====
```

```
# MODELO: ANIMALES
```

```
# La tabla central que es referenciada por Clientes y Empleados
```

```
# =====
```

```
class Animal(models.Model):
```

```
    id_animales = models.AutoField(primary_key=True)
    nombre = models.CharField(max_length=100)
    edad = models.CharField(max_length=20, blank=True, null=True)
    peso = models.DecimalField(max_digits=6, decimal_places=2)
    cuidados = models.TextField(blank=True, null=True)
    enfermedades = models.TextField(blank=True, null=True)
    especie = models.CharField(max_length=50)
    alimentacion = models.TextField(blank=True, null=True)
```

```
    def __str__(self):
```

```
        return f"{self.nombre} ({self.especie}) - ID: {self.id_animales}"
```

```
# =====
```

```
# MODELO: CLIENTES
```

```
# Relacion 1 a muchos: Un Cliente puede tener muchos Animales.
```

```
# =====
```

```
class Cliente(models.Model):
```

```
    # Clave primaria
```

```
    id_cliente = models.AutoField(primary_key=True)
```

```
    apepaterno = models.CharField(max_length=100)
```

```
    apematerno = models.CharField(max_length=100)
```

```
    nombre = models.CharField(max_length=100)
```

```
    domicilio = models.CharField(max_length=255, blank=True, null=True)
```

```
    correo = models.CharField(max_length=100, unique=True)
```

```
    telefono = models.CharField(max_length=20)
```

```
    alergias = models.TextField(blank=True, null=True)
```

```
    def __str__(self):
```

```
        return f"{self.nombre} {self.apepaterno}"
```

```
Animal.add_to_class('dueno', models.ForeignKey(
```

```

    Cliente,
    on_delete=models.CASCADE,
    related_name="animales_poseidos",
    blank=True,
    null=True # Permite animales sin dueño registrado si es necesario
))

```

```

# =====
# MODELO: EMPLEADOS #
# =====
Relación Muchos a Muchos: Un Empleado atiende muchos Animales, # y un Animal puede
ser atendido por muchos Empleados. #
===== class Empleado(models.Model): #
Clave primaria id_empleado = models.AutoField(primary_key=True) # Atributos personales
apepaterno = models.CharField(max_length=100) apematerno =
models.CharField(max_length=100) nombre = models.CharField(max_length=100) telefono
= models.CharField(max_length=20) domicilio = models.TextField(blank=True, null=True)
ocupacion = models.CharField(max_length=100) alergias = models.TextField(blank=True,
null=True) # RELACIÓN MUCHOS A MUCHOS (M:M) # Crea una tabla intermedia
automáticamente (ej: Empleado_animales_a_cargo). animales_a_cargo =
models.ManyToManyField('Animal', related_name="empleados_responsables", blank=True
) def __str__(self): return f"Empleado: {self.nombre} {self.apepaterno} ({self.ocupacion})"

```

1. Procedimiento para realizar las migraciones(makemigrations y migrate.

2 ahora trabajamos con el MODELO: clientes

3 En view de **app_mascotas** crear las funciones con sus códigos correspondientes (inicio_clientes, agregar_clientes, actualizar_clientes, realizar_actualizacion_clientes, borrar_clientes)

4 Crear la carpeta “**templates**” dentro de “**app_mascotas**”.

5 modificar el archivo navbar.html que esta dentro de templates para actualizar la opcion clientes con la carpeta “clientes ” en submenu de clientes (inicio_clientes, agregar_clientes, actualizar_clientes, realizar_actualizacion_clientes, borrar_clientes)

6 Crear la subcarpeta carpeta **clientes** dentro de **app_mascotas\templates**.

7 crear los archivos html con su codigo correspondientes de (agregar_**clientes**.html, ver_**clientes**.html mostrar en tabla con los botones ver, editar y borrar, actualizar_**clientes**.html, borrar_**clientes**.html) dentro de **app_mascotas\templates\clientes**.

8 No utilizar [forms.py](#).

9 procedimiento para agregar en urls.py en **app_mascotas** con el código correspondiente para acceder a las funciones de views.py para operaciones de crud en **clientes**.

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

11 por lo pronto solo trabajar con “**clientes**” dejar pendiente # MODELO: empleados

12 Utilizar colores suaves, atractivos y modernos, el código de las páginas web sencillas para **clientes**.

13 No validar entrada de datos.

14 Al inicio **crear la estructura completa** de carpetas y archivos actiañizados donde se muestra la carpeta **clientes** dentro de templates con los archivos correspondientes a la operación crud **clientes**.

15 proyecto totalmente funcional.

16 finalmente ejecutar servidor en el puerto puerto **8037**.