Manual

Nombre: Tipanluisa Walter

Creamos la clase Tipo con sus respectivos atributos

```
class Tipo(models.Model):
    id=models.AutoField(primary_key=True)
    nombre=models.CharField(max_length=150)
    descripcion=models.TextField(max_length=200)
    def __str__(self):
        fila="{0} \Rightarrow {1} \Rightarrow {2}"
        return fila.format(self.id,self.descripcion, self.nombre)
You, hace 4 minutos | 1 author (You)
```

En el modelo Cliente actualizamos y agregamos el tipo

// Importante designar como null y blank en modo true para poder realizar las migraciones

```
You, hace 10 minutos | 1 author (You)

class Cliente(models.Model):

id=models.AutoField(primary_key=True)
cedula=models.CharField(max_length=100)
apellido=models.CharField(max_length=100)
nombre=models.CharField(max_length=100)

direccion=models.TextField(max_length=100)

fechas_nacimiento=models.DateField()
correo=models.EmailField()
tipo=models.ForeignKey(Tipo, null=True, blank=True, on_delete=models.PROTECT)

def __str__(self):
    fila="{0} \ightarrow {1} \ightarrow {2} \ightarrow {3} \ightarrow {4} \ightarrow {7} \ighta
```

Realizamos las migraciones con makemigrations

```
migrations for 'Facturacion':
Aplicaciones/Facturacion/migrations/0002_tipocliente_cliente_tipo_cliente.py
- Create model TipoCliente
- Add field tipo_cliente to cliente
- Add field tipo_cliente to cliente

python manage.py migrate

Operations to perform:
Apply all migrations: Facturacion, admin, auth, contenttypes, sessions

Running migrations:
Applying Facturacion.0002_tipocliente_cliente_tipo_cliente... OK

O > ~/Documentos/proyectos/septimo/tienda  on  P main !3 ?2

python manage.py runserver
```

Registramos en el archivo admin.py el modelo Tipo que creamos en models

```
You, hace 2 minutos | 1 author (You)

from django.contrib import admin

from .models import Cliente, Tipo

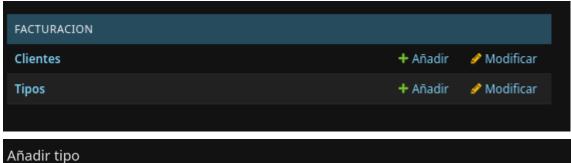
# Register your models here.

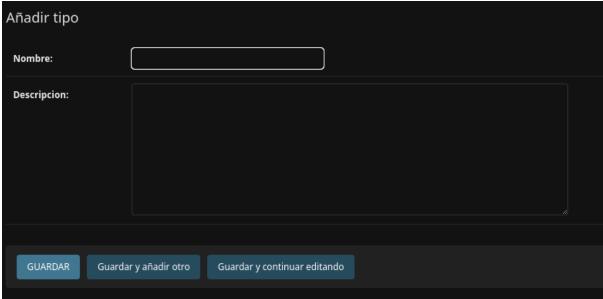
admin.site.register(Cliente)

admin.site.register(Tipo)

6
```

Corremos el servidor y evidenciamos que ya está nuestra nueva tabla creada como modelo y podemos agregar la informacion a nuestra disposicion







Definimos en nuestra lista de clientes para la extracción de datos en este caso se llamada tipoBdd

```
You, nace / minutos | rautnor(you)

from django.shortcuts import render, redirect

from .models import Cliente, Tipo

from django.contrib import messages

# Create your views here.

def home(request):

clientesBdd=Cliente.objects.all()

tipoBdd=Tipo.objects.all()

return render (request, 'listadoClientes.html', {'clientes': clientesBdd , 'tipos': tipoBdd})
```

Actualizamos nuestra funcion para guardar clientes, agregamos el tipo y el tipoSeleccionado que capturara en la base de datos

Realizamos lo mismo para el proceso de actualizacion de cliente y editar cliente

```
def editarCliente (request, id):
    clienteEditar=Cliente.objects.get(id=id)
    tipoBdd=Tipo.objects.all()
    return render (request, 'editarCliente.html', {'cliente': clienteEditar, 'tipos': tipoBdd})
```

```
def procesarActualizacionCliente (request):
   id=request.POST['id']
    id_tipo=request.POST['id_tipo']
    tipoSeleccionado=Tipo.objects.get(id=id_tipo)
    cedula=request.POST['cedula']
    nombre=request.POST['nombre']
    apellido=request.POST['apellido']
    fechas_nacimiento=request.POST['fechas_nacimiento']
    correo=request.POST['correo']
   direccion=request.POST['direccion']
    # Actualizar el cliente
   cliente=Cliente.objects.get(id=id)
   cliente.cedula=cedula
   cliente.tipo=tipoSeleccionado
    cliente.nombre=nombre
    cliente.apellido=apellido
    cliente.fechas_nacimiento=fechas_nacimiento
    cliente.correo=correo
    cliente.direccion=direccion
    cliente.save()
    messages.success(request, 'Cliente actualizado correctamente')
   return redirect ('/')
```

En nuestro formulario aumentamos un select con jinja2 y llamamos a nuestros datos que insertamos en el admin de django

Realizamos lo mismo en el formulario de editar debemos ingresar obligatoriamente el id y el name en este caso se llama id_tipo

```
ou, hace 1 segundo * Uncommitted changes

<label for="" class="form-label">Tipo de cliente</label>
  <select name="id_tipo" required class="form-control m-1" id="id_tipo">
        <option value="" disabled>— Seleccione una opcion --</option>
        {% for tipo in tipos %}
        <option value="{{tipo.id}}">{{tipo.nombre}}</option>
        {% endfor %}
        <select>
        <script>
        document.getElementById('id_tipo').value = '{{cliente.tipo.id}}'
        </script>
```

Ahora podemos observar que ya nos sale la información tanto en el formulario como en la tabla.

Ademas de ello podemos agregar eliminar editar la informacion que nosotros requerimos



