Primera parte
Proyecto: GYM.
lenguaje: Python.
Framework: Django.
Editor: VS code.

1. Procedimiento para crear carpeta del Proyecto: **UIII_GYM_0433**
2. Procedimiento para abrir vs code sobre la carpeta **UIII_GYM_0433**
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_Gym** sin duplicar carpeta.
9. Procedimiento para ejecutar servidor en el **puerto 8033**
10. Procedimiento para copiar y pegar el link en el navegador.
11. Procedimiento para crear aplicacion **app_gym**
12. Aqui el modelo models.py

=-=-=-=-=-

```
from django.db import models

# ========================================
# MODELO: MIEMBRO
# ========================================
class Miembro(models.Model):
    nombre = models.CharField(max_length=100)
    apellido = models.CharField(max_length=100)
    fecha_nacimiento = models.DateField()
    email = models.EmailField(unique=True)
    telefono = models.CharField(max_length=15, blank=True, null=True)
    fecha_inscripcion = models.DateField(auto_now_add=True)
    membresia_activa = models.BooleanField(default=True)

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

# ========================================

# MODELO: CLASE
# ========================================
class Clase(models.Model):
    nombre_clase = models.CharField(max_length=100)
    descripcion = models.TextField(blank=True, null=True)
    horario = models.CharField(max_length=50) # Ejemplo: "Lunes 10:00-11:00"
    duracion_minutos = models.IntegerField()
    cupo_maximo = models.IntegerField()
    nivel_dificultad = models.CharField(
```

```
        max_length=20,
        choices=[
            ('Principiante', 'Principiante'),
            ('Intermedio', 'Intermedio'),
            ('Avanzado', 'Avanzado'),
        ],
        default='Principiante'
    )
    # Relación de Muchos a Muchos con Miembro
    miembros_inscritos = models.ManyToManyField(Miembro,
related_name='clases_inscritas', blank=True)

    def __str__(self):
        return self.nombre_clase

# ========================================
# MODELO: EMPLEADO
# ========================================
class Empleado(models.Model):
    nombre = models.CharField(max_length=100)
    apellido = models.CharField(max_length=100)
    fecha_contratacion = models.DateField(auto_now_add=True)
    puesto = models.CharField(max_length=100) # Ej: "Instructor",
"Recepcionista", "Gerente"
    salario = models.DecimalField(max_digits=10, decimal_places=2, blank=True,
null=True)
    email = models.EmailField(unique=True)
    telefono = models.CharField(max_length=15, blank=True, null=True)
    # Relación de Uno a Muchos con Clase (un empleado puede impartir muchas
clases)
    clases_impartidas = models.ForeignKey(
        Clase,
        on_delete=models.SET_NULL, # Si la clase se elimina, el empleado sigue
existiendo pero no tendrá clase asignada
        related_name='instructores',
        blank=True,
        null=True
    )

    def __str__(self):
        return f"{self.nombre} {self.apellido} ({self.puesto})"
=-=-=-=-=-
```

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

13. Primero trabajamos con el **MODELO: MIEMBRO**

14. En view de **app_gym** crear las funciones con sus códigos

correspondientes (inicio_gym, agregar_miembro, actualizar_miembro, realizar_actualizacion_miembro, borrar_miembro)

15. Crear la carpeta **“templates”** dentro de **“app_gym”.**
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 Gym”, “Inicio”:
    “Miembro”,en submenu de miembro (Aregar Miembro,ver miembro, actualizar miembro, borrar miembro);
    “Clase” en submenu de Clase (Aregar clase,ver clase, actualizar clase, borrar clase)
    “Empleado” en submenu de empleado (Aregar empleado,ver empleado, actualizar empleado, borrar empleado), 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 Lucia Nava, 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 gym.
21. Crear la subcarpeta carpeta **miembro** dentro de **app_gym\templates.**
22. Crear los archivos html con su codigo correspondientes de (agregar_miembros.html, ver_miembros.html mostrar en tabla con los botones ver, editar y borrar, actualizar_miembros.html, borrar_miembro.html) dentro de **app_gym\templates\miembros.**
23. No utilizar forms.py.
24. Procedimiento para crear el archivo urls.py en **app_gym** con el código correspondiente para acceder a las funciones de views.py para operaciones de crud en miembros.
25. Procedimiento para agregar **app_gym** en settings.py de **backend_Gym**
26. Realizar las configuraciones correspondiente a urls.py de **backend_Gym** para enlazar con **app_gym**
27. **Procedimiento para registrar los modelos en admin.py y volver a realizar las migraciones.**
28. Por lo pronto solo trabajar con “miembro” dejar pendiente #MODELO: CLASE y # MODELO: EMPLEADO
29. Utilizar colores suaves, atractivos y modernos, el código de las páginas web sencillas.
30. No validar entrada de datos.
31. Al inicio **crear la estructura completa** de carpetas y archivos.
32. Proyecto totalmente funcional.
33. Finalmente ejecutar servidor en el puerto puerto **8033**.