Primera parte
Proyecto: **aeropuerto.**
lenguaje: **Python.**
Framework: **Django.**
Editor: VS code.
1 Procedimiento para crear carpeta del Proyecto: **UIII_aeropuerto_0409**
2 procedimiento para **abrir vs code sobre la carpeta UIII_aeropuerto_0409**
3 procedimiento para abrir terminal en vs code0
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_aeropuerto** 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_aeropuerto**
12 Aqui el modelo models.py
=-=-=-=-=-

```
from django.db import models



# ========================================
# MODELO: AVION
# ========================================
class Avion(models.Model):
    matricula = models.CharField(max_length=20, unique=True)
    modelo = models.CharField(max_length=50)
    fabricante = models.CharField(max_length=50)
    capacidad = models.PositiveIntegerField()
    anio_fabricacion = models.PositiveIntegerField()
    tipo = models.CharField(max_length=30)
    estado = models.CharField(max_length=30)

    def __str__(self):
        return f"{self.matricula} - {self.modelo}"
```

```
# ========================================
# MODELO: VUELO
# ========================================
class Vuelo(models.Model):
    codigo_vuelo = models.CharField(max_length=20, unique=True)
    origen = models.CharField(max_length=50)
    destino = models.CharField(max_length=50)
    fecha_salida = models.DateTimeField()
    fecha_llegada = models.DateTimeField()
    duracion_horas = models.DecimalField(max_digits=5, decimal_places=2)
    estatus = models.CharField(max_length=30)

    # Relación 1 a muchos → un avión puede tener varios vuelos
    avion = models.ForeignKey(Avion, on_delete=models.CASCADE,
related_name="vuelos")

    def __str__(self):
        return f"{self.codigo_vuelo} ({self.origen} → {self.destino})"




# ========================================
# MODELO: EMPLEADO
# ========================================
class Empleado(models.Model):
    nombre = models.CharField(max_length=100)
    apellido = models.CharField(max_length=100)
    cargo = models.CharField(max_length=50)
    salario = models.DecimalField(max_digits=10, decimal_places=2)
    fecha_contratacion = models.DateField()
    licencia = models.CharField(max_length=30)
    turno = models.CharField(max_length=20)

    # Relación muchos a muchos → un empleado puede trabajar en varios aviones y
viceversa
    aviones = models.ManyToManyField(Avion, related_name="empleados")

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

12.5 Procedimiento para realizar las migraciones(makemigrations y migrate.
13 primero trabajamos con el MODELO: AVION
14 En view de **app_Aeropuerto** crear las funciones con sus códigos correspondientes (inicio_aeropuerto, agregar_avion,

actualizar_avion, realizar_actualizacion_avion, borrar_avion)
15 Crear la carpeta **“templates”** dentro de **“app_aeropuerto”**.
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 Aeropuerto”, “Inicio”, “avion”,en submenu de avion(Agregar avion, ver avion, actualizar avion, borrar avion), “Vuelos” en submenu de Vuelos(Agregar vuelos, ver vuelos, actualizar vuelos, borrar vuelos)
“Empleados” en submenu de Empleados(Agregar empleados, ver empleados, actualizar empleados, borrar empleados), 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 Ing. Bryan Villalobos, 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 aeropuerto.
21 Crear la subcarpeta carpeta **avion** dentro de app_Aeropuerto\templates.
22 crear los archivos html con su codigo correspondientes de (agregar_avion.html, ver_aviones.html mostrar en tabla con los botones ver, editar y borrar, actualizar_avion.html, borrar_avion.html)
dentro de **app_Aeropuerto\templates\avion.**
23 No utilizar forms.py.
24 procedimiento para crear el archivo urls.py en **app_Aeropuerto** con el código correspondiente para acceder a las funciones de views.py para operaciones de crud en aviones.
25 procedimiento para agregar **app_Aeropuerto** en settings.py de **backend_Aeropuerto**
26 realizar las configuraciones correspondiente a urls.py de **backend_Aeropuerto** para enlazar con **app_Aeropuerto**

27 procedimiento para registrar los modelos en admin.py y volver a realizar las migraciones.
27 por lo pronto solo trabajar con “avion” dejar pendiente # MODELO: VUELO y # MODELO: EMPLEADO
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.**