Primera parte
Proyecto: Uber.
lenguaje: Python.
Framework: Django.
Editor: VS code.
1 Procedimiento para crear carpeta del Proyecto: UIII_Uber_0569
2 procedimiento para abrir vs code sobre la carpeta
UIII_Uber_0569
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_Uber sin duplicar
carpeta.
9 procedimiento para ejecutar servidor en el puerto 8069
10 procedimiento para copiar y pegar el link en el navegador.
11 procedimiento para crear aplicacion app_uber
12 Aqui el modelo models.py
=-=-=-=-=-

```
from django.db import models

# ========================================
# MODELO: Chofer
# ========================================
class Chofer(models.Model):
    id_chofer = models.AutoField(primary_key=True)
    nombre = models.CharField(max_length=100)
    licencia = models.CharField(max_length=50)
    telefono = models.CharField(max_length=15)
    direccion = models.CharField(max_length=150)
    email = models.EmailField(unique=True)
    edad = models.IntegerField()
    fecha_ingreso = models.DateField()

    def __str__(self):
        return self.nombre


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

# MODELO: Usuario_pasajero
# ========================================
class UsuarioPasajero(models.Model):
    id_usuario = models.AutoField(primary_key=True)
    nombre = models.CharField(max_length=100)
    email = models.EmailField(unique=True)
```

```
    telefono = models.CharField(max_length=15)
    direccion = models.CharField(max_length=150)
    fecha_registro = models.DateField()
    genero = models.CharField(max_length=10)
    ciudad = models.CharField(max_length=100)

    def __str__(self):
        return self.nombre

# ========================================
# MODELO: Viaje
# ========================================
class Viaje(models.Model):
    id_viaje = models.AutoField(primary_key=True)
    destino = models.CharField(max_length=100)
    fecha = models.DateField()
    hora_salida = models.TimeField()
    duracion = models.CharField(max_length=50)
    costo = models.DecimalField(max_digits=8, decimal_places=2)
    estatus = models.CharField(max_length=20)

    # 🔹 Relación 1:N con Chofer
    chofer = models.ForeignKey(Chofer, on_delete=models.CASCADE,
related_name='viajes')

    # 🔹 Relación N:M con UsuarioPasajero
    pasajeros = models.ManyToManyField(UsuarioPasajero, related_name='viajes')

    def __str__(self):
        return f"Viaje a {self.destino} con {self.chofer.nombre}"
```

=-=-=-=-=-

12.5 Procedimiento para realizar las migraciones(makemigrations y migrate.

13 primero trabajamos con el MODELO: CATEGORÍA

14 En view de app_Uber crear las funciones con sus códigos correspondientes (inicio_uber, agregar usuario_pasajero,

actualizar_usuario_pasajero, realizar_actualizacion_usuario_pasajero, borrar_usuario_pasajero)

15 Crear la carpeta “templates” dentro de “app_Uber”.

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 Uber”, “Inicio”, “usuario_pasajero”,en submenu de categorias(Aregar usuario pasajero,ver usuario_pasajero, actualizar usuario_pasajero, borrar usuario_pasajero), “chofer” en submenu de chofer(Aregar chofer,ver

chofer, actualizar chofer, borrar chofer)
“Viajes” en submenu de Viajes(Aregar viaje,ver viaje,
actualizar viaje, borrar viaje), 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. Aarón Dominguez, 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 uber.
21 Crear la subcarpeta carpeta usuario_pasajero dentro de
app_Uber\templates.
22 crear los archivos html con su codigo correspondientes de
(agregar_usuario_pasajero.html, ver_usuario_pasajero.html mostrar en tabla con
los botones ver, editar y borrar, actualizar_usuario_pasajero.html,
borrar_usuario_pasajero.html)
dentro de app_Uber\templates\usuario_pasajero.
23 No utilizar forms.py.
24 procedimiento para crear el archivo urls.py en app_Cinepolis con
el código correspondiente para acceder a las funciones de views.py
para operaciones de crud en usuario_pasajero.
25 procedimiento para agregar app_Uber en settings.py de
backend_Uber
26 realizar las configuraciones correspondiente a urls.py de
backend_Uber para enlazar con app_Uber

27 procedimiento para registrar los modelos en admin.py y volver a
realizar las migraciones.
27 por lo pronto solo trabajar con “categoría” dejar pendiente #
MODELO: Chofer y # MODELO: Viaje
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 8069.