Entrega de Documento Reto Final POO

Felipe Marin Zuluaga, John Alejandro Restrepo, Harol Felipe Sanchez

Ingeniería de Sistemas, Facultad de Ingeniería, Corporación Universitaria Minuto de Dios

NRC 74169: Programación Orientada a Objetos

Jasson Diaz Zamundio

Mayo 28, 2025

Justificación del Proyecto

La justificación del código radica en la necesidad de mejorar la gestión de los servicios de alimentación en Ciudad Bolívar. Actualmente, los procesos de registro de beneficiarios, asignación de cupos y entrega de alimentos se manejan de forma manual, lo que dificulta el control eficiente y puede llevar a errores o falta de transparencia en la distribución.

Desarrollar un sistema en Python permitirá:

Automatizar procesos para reducir errores y agilizar el registro de datos.

Facilitar la asignación de cupos diarios evitando sobrecargas o desperdicio de recursos.

Generar reportes de entregas de alimentos para mejorar la toma de decisiones y garantizar que la ayuda llegue a quienes más lo necesitan.

Este sistema proporcionará un control más preciso y permitirá una mejor organización de los comedores comunitarios, beneficiando directamente a las familias que dependen de este servicio.

```python
from modelo import Microbus, Camioneta, Mototaxi

class Controlador:
    def __init__(self, vista):
        self.vista = vista
        self.rutas = []

    def iniciar(self):
        while True:
            opcion = self.vista.mostrar_menu()
            if opcion == "1":
                self.registrar_ruta()
            elif opcion == "2":
                self.registrar_estudiante()
            elif opcion == "3":
                self.listar_estudiantes_por_ruta()
            elif opcion == "4":
                self.marcar_asistencia()
            elif opcion == "5":
                break
            else:
                self.vista.mostrar_mensaje("Opción no válida.")

    def registrar_ruta(self):
        trayecto, capacidad, conductor, tipo = self.vista.solicitar_datos_ruta()
        if tipo == "1":
            ruta = Microbus(trayecto, capacidad, conductor)
        elif tipo == "2":
            ruta = Camioneta(trayecto, capacidad, conductor)
        elif tipo == "3":
            ruta = Mototaxi(trayecto, capacidad, conductor)
        else:
            self.vista.mostrar_mensaje("Tipo no válido.")
            return
        self.rutas.append(ruta)
        self.vista.mostrar_mensaje("Ruta registrada.")

    def registrar_estudiante(self):
        if not self.rutas:
            self.vista.mostrar_mensaje("Primero registre una ruta.")
            return
        nombre = self.vista.solicitar_nombre_estudiante()
        seleccion = self.vista.mostrar_rutas(self.rutas)
        if 0 <= seleccion < len(self.rutas):
            mensaje = self.rutas[seleccion].asignar_estudiante(nombre)
            self.vista.mostrar_mensaje(mensaje)
        else:
            self.vista.mostrar_mensaje("Ruta no válida.")

    def listar_estudiantes_por_ruta(self):
        for ruta in self.rutas:
            self.vista.mostrar_estudiantes_ruta(ruta)

    def marcar_asistencia(self):
        nombre = self.vista.solicitar_nombre_estudiante()
        for ruta in self.rutas:
            mensaje = ruta.marcar_asistencia(nombre)
            if "marcada" in mensaje or "no encontrado" not in mensaje:
                self.vista.mostrar_mensaje(mensaje)
                return
        self.vista.mostrar_mensaje("Estudiante no encontrado en ninguna ruta.")
```

```python
from abc import ABC, abstractmethod

class Transporte(ABC):
    def __init__(self, trayecto, capacidad, conductor):
        self.trayecto = trayecto
        self.capacidad = capacidad
        self.conductor = conductor
        self.estudiantes = []

    @abstractmethod
    def tipo(self):
        pass

    def asignar_estudiante(self, estudiante):
        if len(self.estudiantes) < self.capacidad:
            self.estudiantes.append({"nombre": estudiante, "asistencia": False})
            return f"Estudiante {estudiante} asignado a la ruta {self.trayecto}."
        else:
            return "No hay cupo disponible en esta ruta."

    def listar_estudiantes(self):
        return self.estudiantes

    def marcar_asistencia(self, nombre):
        for est in self.estudiantes:
            if est["nombre"] == nombre:
                est["asistencia"] = True
                return f"Asistencia marcada para {nombre}."
        return "Estudiante no encontrado en esta ruta."

class Microbus(Transporte):
    def tipo(self):
        return "Microbús"

class Camioneta(Transporte):
    def tipo(self):
        return "Camioneta"

class Mototaxi(Transporte):
    def tipo(self):
        return "Mototaxi"
```

```python
class Vista:
    def mostrar_menu(self):
        print("\n1. Registrar ruta")
        print("2. Registrar estudiante")
        print("3. Listar estudiantes por ruta")
        print("4. Marcar asistencia")
        print("5. Salir")
        return input("Seleccione una opción: ")

    def solicitar_datos_ruta(self):
        trayecto = input("Trayecto: ")
        capacidad = int(input("Capacidad: "))
        conductor = input("Conductor: ")
        print("Tipo de transporte: 1. Microbús 2. Camioneta 3. Mototaxi")
        tipo = input("Seleccione tipo: ")
        return trayecto, capacidad, conductor, tipo

    def solicitar_nombre_estudiante(self):
        return input("Nombre del estudiante: ")

    def mostrar_rutas(self, rutas):
        for idx, ruta in enumerate(rutas):
            print(f"{idx+1}. {ruta.trayecto} ({ruta.tipo()}) - Cupo: {len(ruta.estudiantes)}/{ruta.capacidad}")
        return int(input("Seleccione la ruta: ")) - 1

    def mostrar_estudiantes_ruta(self, ruta):
        print(f"\nRuta: {ruta.trayecto} ({ruta.tipo()})")
        for est in ruta.listar_estudiantes():
            print(f"- {est['nombre']} (Asistencia: {'Sí' if est['asistencia'] else 'No'})")

    def mostrar_mensaje(self, mensaje):
        print(mensaje)
```

```python
from controlador import Controlador
from vista import Vista

if __name__ == "__main__":
    vista = Vista()
    controlador = Controlador(vista)
    controlador.iniciar()
```