TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE OAXACA

Ingeniería en Sistemas Computacionales

Desarrollo de Software para la Toma de Decisiones

Integrantes:

Cruz Sanchez Jhoan Marvin (21160617)

Juárez Monjaraz Griselda Itzel (22161115)

Ortega Patiño Nimsi Joana (A22700026)

Espinoza De La Rosa Uriel (22161681)

José Sebastián Jafet (22161112)

Grupo: 8SB

Horario: 12:00-13:00

Tema: Programa Funcional: Implementación de un Modelo de Decisión

Catedrático: Martínez Nieto Adelina

23 de febrero de 2026

MANUAL DE USO

THE LP PRODUCT-MIX MODEL FORMULATION

DSS de Mezcla Óptima de Producción (LP Product-Mix)

Versión con Interfaz Gráfica (GUI - Tkinter)

- ## Descripción del Modelo

El sistema implementa el modelo de Mezcla de Productos (Product-Mix Model) descrito en la página 168 del libro de Efraim Turban. El objetivo es maximizar la utilidad total Z produciendo dos productos (CC-7 y CC-8), sujetos a restricciones de mano de obra, presupuesto y requerimientos mínimos de mercado.

Función Objetivo:

Max Z = 8000X1 + 12000X2

Restricciones:

300X1 + 500X2 ≤ 200000 (Mano de obra)

10000X1 + 15000X2 ≤ 8000000 (Presupuesto)

X1 ≥ 100 (Demanda mínima CC-7)

X2 ≥ 200 (Demanda mínima CC-8)

- ## Problema que Resuelve

Determina cuántas unidades producir de cada producto para maximizar la utilidad total, considerando: utilidad por unidad, límite de mano de obra, presupuesto de materiales y mínimos obligatorios de mercado.

- ## Requisitos Previos

- Requiere Python 3.9 o superior y la librería PuLP instalada.
- Instalación: pip install pulp

- ## Estructura del Proyecto

- gui_main.py: Implementa la Interfaz Gráfica de Usuario (Tkinter).
- Logic.py: Contiene el modelo matemático de Programación Lineal.
- PuLP: Librería utilizada para resolver el modelo con el método Simplex (CBC)

- ## Cómo Ejecutar el Programa

python gui_main.py

- ## Datos que Solicita el Sistema

- Utilidad CC-7 ($): Ganancia por cada unidad producida del modelo CC-7.
- Utilidad CC-8 ($): Ganancia por cada unidad producida del modelo CC-8.
- Días de labor disponibles: Total de días de mano de obra disponibles.
- Presupuesto de materiales ($): Límite máximo de inversión en materiales.
- Mínimo CC-7 a producir: Demanda mínima obligatoria del mercado.
- Mínimo CC-8 a producir: Demanda mínima obligatoria del mercado.

En total, el sistema solicita seis parámetros dinámicos. Todos deben ingresarse como valores numéricos (enteros o decimales).

- ## Ejemplo de Prueba

Datos utilizados para pruebas:

Utilidad CC-7: 8000

Utilidad CC-8: 12000

Límite labor: 200000

Presupuesto: 8000000

Mínimo CC-7: 100

Mínimo CC-8: 200

- ## Interpretación de Resultados

El sistema muestra el estado del modelo (Óptimo o Infactible), la estrategia sugerida de producción y la utilidad total proyectada. Además, incluye un análisis de recursos, indicando si los recursos fueron completamente utilizados o si existe capacidad disponible.

- ## Solución de Problemas

- Error 'No module named pulp': ejecutar pip install pulp

- Modelo infactible: revisar mínimos y recursos disponibles

- Error de entrada: ingresar solo valores numéricos

Código Fuente

→ *Captura 1:* Estructura general de la interfaz gráfica

inputs.py | gui_main.py

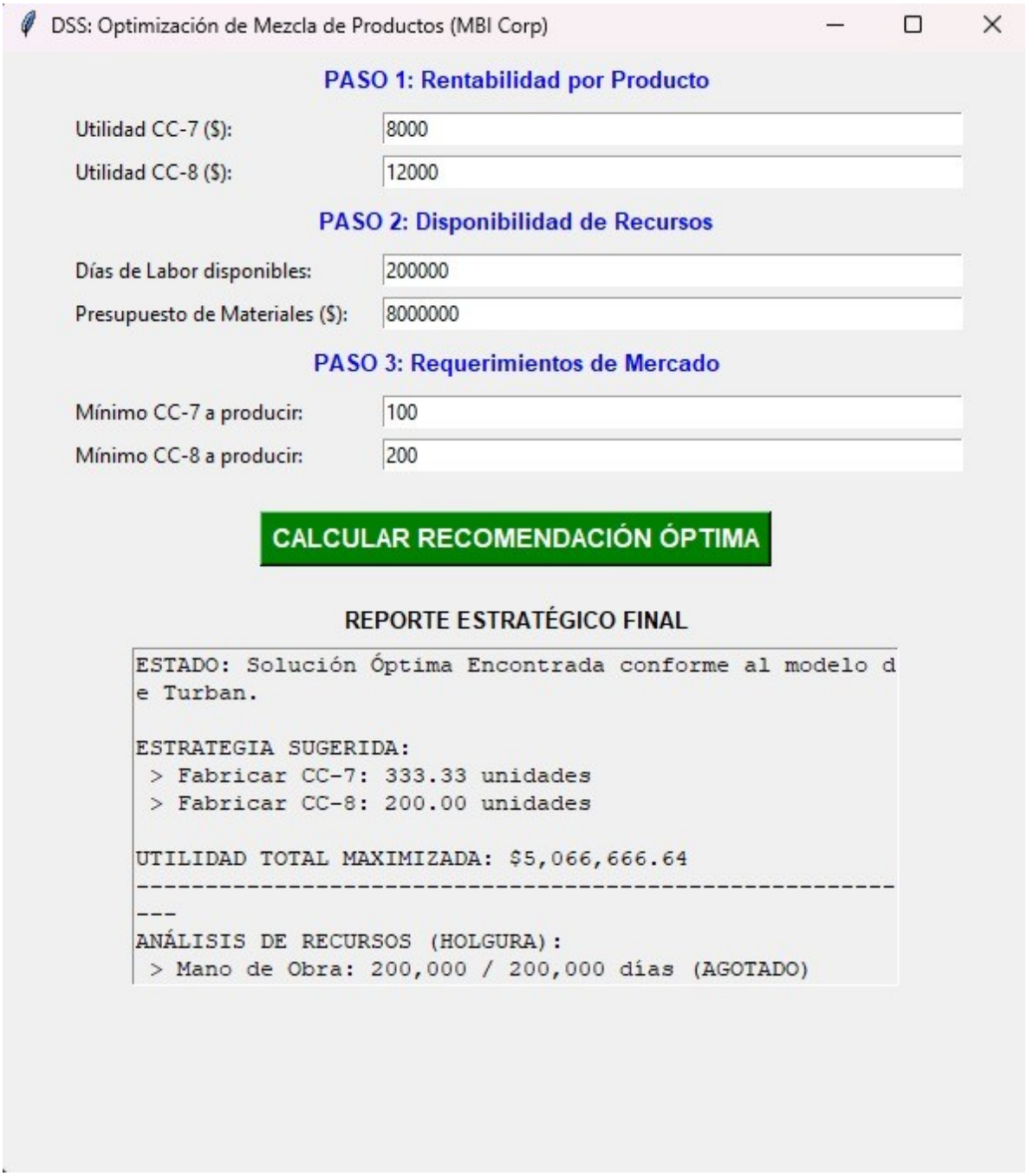
ProgramaFuncional_ModelodeDecision-main > gui_main.py > ...

```python
import tkinter as tk
from tkinter import messagebox, ttk
import logic  # Importamos tu lógica matemática existente

# Responsabilidad: Interfaz Gráfica de Usuario (GUI)
# Este módulo convierte el modelo matemático en una herramienta visual intuitiva.

class DSSInterface:
    def __init__(self, root):
        self.root = root
        self.root.title("DSS: Optimización de Mezcla de Productos (MBI Corp)")
        self.root.geometry("500x650")

        # Estilo de etiquetas según la terminología de Turban
        style_header = ("Arial", 10, "bold")

        # --- SECCIÓN 1: RENTABILIDAD (Variables de Resultado) ---
        tk.Label(root, text="PASO 1: Rentabilidad por Producto", font=style_header, fg="blue").pack(pady=5)

        self.u_cc7 = self.crear_entrada("Utilidad CC-7 ($):", "8000") # Datos de MBI
        self.u_cc8 = self.crear_entrada("Utilidad CC-8 ($):", "12000")

        # --- SECCIÓN 2: RECURSOS (Variables Incontrolables / Restricciones) ---
        tk.Label(root, text="PASO 2: Disponibilidad de Recursos", font=style_header, fg="blue").pack(pady=5)

        self.limit_labor = self.crear_entrada("Días de Labor disponibles:", "200000")
        self.limit_budget = self.crear_entrada("Presupuesto de Materiales ($):", "8000000")

        # --- SECCIÓN 3: MERCADO (Límites Inferiores) ---
        tk.Label(root, text="PASO 3: Requerimientos de Mercado", font=style_header, fg="blue").pack(pady=5)

        self.min_cc7 = self.crear_entrada("Mínimo CC-7 a producir:", "100")
        self.min_cc8 = self.crear_entrada("Mínimo CC-8 a producir:", "200")

        # --- BOTÓN DE ACCIÓN: Ejecutar Modelo ---
        self.btn_resolver = tk.Button(root, text="CALCULAR RECOMENDACIÓN ÓPTIMA",
                                      command=self.ejecutar_modelo, bg="green", fg="white", font=("Arial", 11, "bold"))
        self.btn_resolver.pack(pady=20)

        # --- SECCIÓN DE RESULTADOS ---
        tk.Label(root, text="REPORTE ESTRATÉGICO FINAL", font=style_header).pack()
        self.txt_resultados = tk.Text(root, height=12, width=55, state="disabled", bg="#f0f0f0")
        self.txt_resultados.pack(pady=5)

    def crear_entrada(self, texto, default_val):
        """Crea una etiqueta y un cuadro de texto con un valor por defecto."""
        frame = tk.Frame(self.root)
        frame.pack(fill="x", padx=40, pady=2)
        tk.Label(frame, text=texto, width=25, anchor="w").pack(side="left")
        entry = tk.Entry(frame)
        entry.insert(0, default_val)
        entry.pack(side="right", expand=True, fill="x")
        return entry

    def ejecutar_modelo(self):
        """Extrae los datos, llama al cerebro lógico y muestra el reporte."""
        try:
            # Captura dinámica de datos
            datos = {
```

➔ *Captura 2:* Ejecución del modelo y llamada al motor lógico.

```
                'utilidad_cc7': float(self.u_cc7.get()),
                'utilidad_cc8': float(self.u_cc8.get()),
                'limite_labor': float(self.limit_labor.get()),
                'limite_presupuesto': float(self.limit_budget.get()),
                'min_cc7': float(self.min_cc7.get()),
                'min_cc8': float(self.min_cc8.get())
            }

            # Ejecución del motor lógico (Llamada al Integrante 1)
            res = logic.calcular_mezcla_optima(datos)

            # Presentación de resultados
            self.mostrar_reporte(res, datos)

        except ValueError:
            messagebox.showerror("Error de Datos", "Por favor, ingrese solo números válidos.")

    def mostrar_reporte(self, res, datos):
        """Muestra el análisis de la solución óptima y la holgura (Slack)."""
        self.txt_resultados.config(state="normal")
        self.txt_resultados.delete("1.0", tk.END)

        if res['status'] == 1:
            reporte = (
                f"ESTADO: Solución Óptima Encontrada conforme al modelo de Turban.\n\n"
                f"ESTRATEGIA SUGERIDA:\n"
                f" > Fabricar CC-7: {res['x1']:,.2f} unidades\n"
                f" > Fabricar CC-8: {res['x2']:,.2f} unidades\n\n"
                f"UTILIDAD TOTAL MAXIMIZADA: ${res['ganancia_total']:,.2f}\n"
                f"-------------------------------------------------------\n"
                f"ANÁLISIS DE RECURSOS (HOLGURA):\n"
            )

            # Cálculo de recursos usados para análisis de Slack
            labor_usada = 300 * res['x1'] + 500 * res['x2']
            sobrante_pres = datos['limite_presupuesto'] - (10000 * res['x1'] + 15000 * res['x2'])

            reporte += f" > Mano de Obra: {labor_usada:,.0f} / {datos['limite_labor']:,.0f} días (AGOTADO)\n"
            reporte += f" > Presupuesto Sobrante: ${sobrante_pres:,.2f} (DISPONIBLE)\n"

            self.txt_resultados.insert(tk.END, reporte)
        else:
            self.txt_resultados.insert(tk.END, "ERROR: El escenario es INFACTIBLE.\nNo hay recursos suficientes para cumplir mínimos.")

        self.txt_resultados.config(state="disabled")

# Inicio de la aplicación
if __name__ == "__main__":
    root = tk.Tk()
    app = DSSInterface(root)
    root.mainloop()
```

➔ *Captura 3:* Reporte estratégico final y análisis de holgura.



Este DSS demuestra cómo un modelo matemático de optimización puede convertirse en una herramienta funcional para apoyar la toma de decisiones estratégicas, integrando teoría y programación en una solución práctica.