Universidad Tecnológica de Panamá
Facultad de Ingeniería en Sistemas Computacionales
Asignatura: Desarrollo Lógico Y Algoritmo
Ejercicio Práctico2

Profesor: Napoleón Ibarra Valor: 100 puntos

Estudiantes: Rick Jimenez-Isaac Moreno Cédula: 4-904-2017 4-836-1992

Fecha Inicio: 13/10/2025 --> 4:10 PM
Fecha Entrega: 14/10/2025 -->2:25 PM
Fecha Cierre (Tope): 20/10/2025 -->4:10 PM

Procedimiento:

De manera INDIVIDUAL o en grupos de 2 PERSONAS, realizar la asignación.
Investigue, descargar e instalar en una máquina virtual (Usted elige el IDE) una distribución de Windows Server (2016/2019).
Se debe entregar al profesor: Documento digital: entrega en la plataforma (TEAM) el y/o los códigos desarrollando los problemas. Sustente su trabajo en el aula de clases.

Criterios de Evaluación:

Criterios	*Puntos (Mínimo=1, Máximo=5)*	*Porcentaje*
Desarrollo	1-5	70 %
Sustentación	1-5	15 %
Puntualidad	1-5	15 %

I Parte. Laboratorio. *Valor 20 puntos*

1. Elegir un software para virtualización (Virtual Box / VMware), descargar, instalar y configurarlo.
2. Descargar, instalar la distribución de Windows Server (2016/2019) elegida dentro de la máquina virtual.
3. Realizar el proceso de configuración correspondiente al realizar el proceso de la instalación del SO.
4. Configurar una carpeta para hacer respaldos de códigos dentro del SO, verifique su instalación, configuración.
5. Realizar la configuración del Servidor de Datos (que se accesible por Remote Desktop Connection, ANYDESK, FTP).
6. Hacer pruebas de funcionamiento.

II Parte. Caso de Estudio. *Valor 20 Puntos*

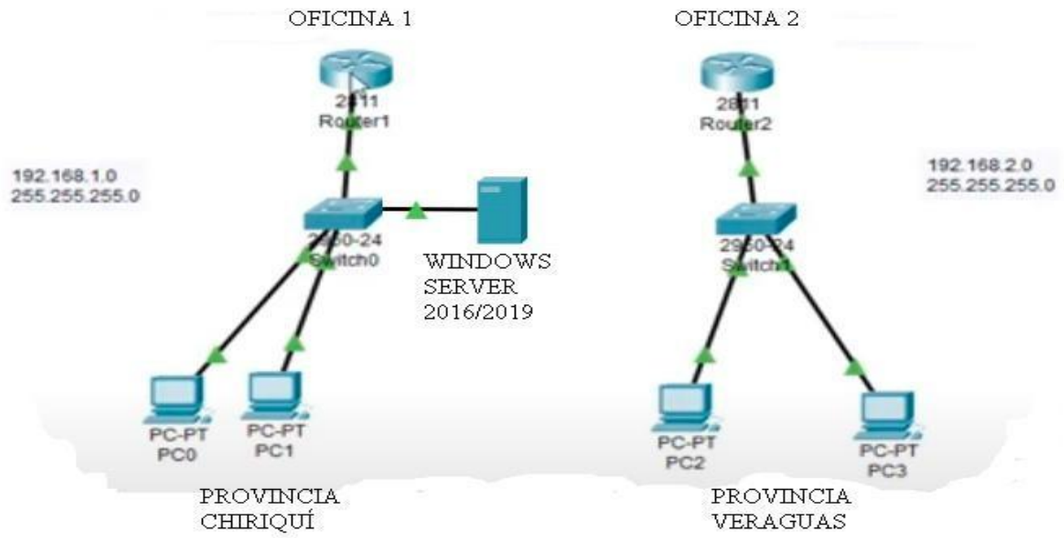


Figura 1. Prototipo de diagrama

Procedimiento:

1. Teniendo en cuenta el siguiente plano de oficina (Figura 1), confeccione el esquema de RED LAN, puesto que en ambas oficinas 1 y 2 se va a utilizar el servidor de datos.

III Parte. Pseudocódigos, diagrama de flujo. *Valor 10 Puntos*

1. Desarrolle los problemas en pseudocódigo para la parte IV de la asignación.

IV Parte. Desarrollo de problemas en Python. *Valor 20 Puntos*

1. Escriba un programa en Python que lea la hora en notación de 24 horas y que imprima en notación de 12; por ejemplo, si la entrada es 13:45, la salida será 1:45 pm. El programa debe solicitar al usuario que introduzca exactamente cinco caracteres para especificar una hora. *Valor 10 puntos*
2. Una compañía de refrescos comercializa tres productos: de cola, de naranja y de limón. Se dese realizar una aplicación que calcule las ventas realizadas de cada producto. Para ello, se leerá la cantidad vendida (máximo 5000000) y el precio en Balboas de cada producto y se mostrará un informe de ventas como el que sigue:

```
Producto      Ventas      Precio          Total
-------------------------------------------------
Cola         1000000        0.17      170000.00
Naranja       350000        0.20       70000.00
Limon         530000        0.19      100700.00
                           TOTAL      340700.00
```

NOTA: Para el desarrollo de la parte IV en Python, utilizar la interfaz Tkinter, tome en cuenta después del punto (decimal) 2 cifras significativas.

BUENA SUERTE

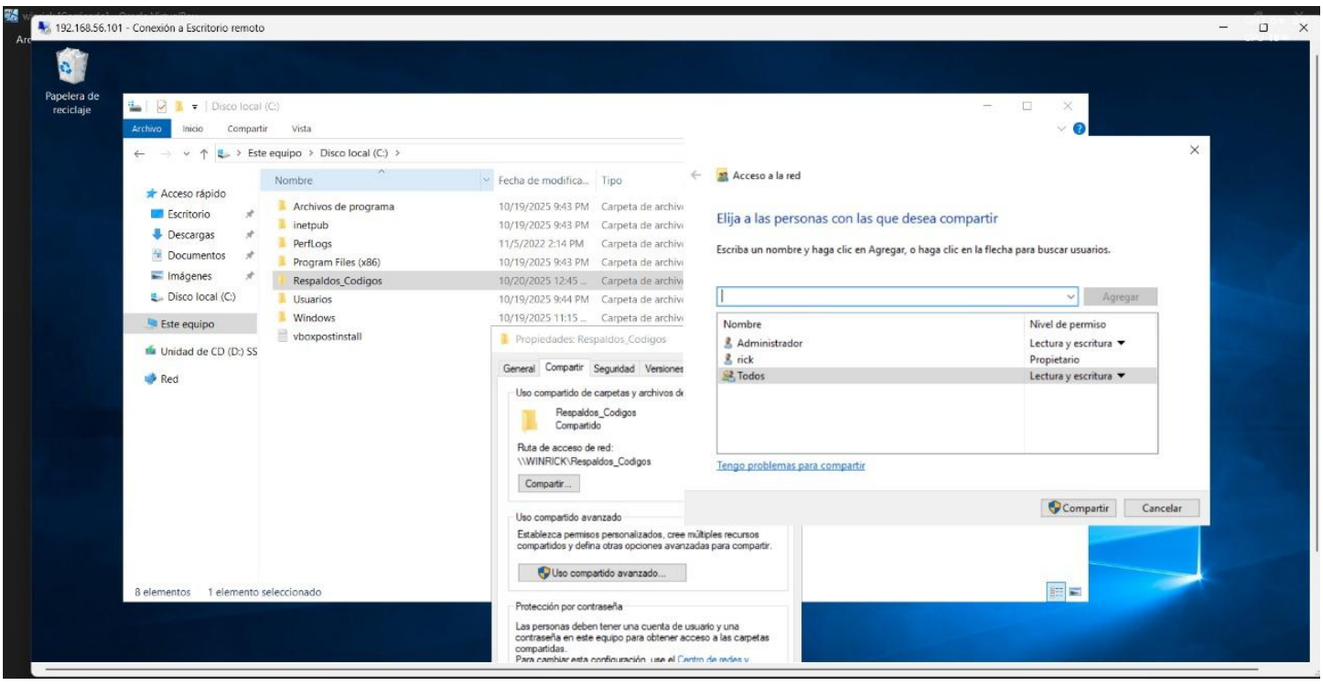
I parte.

III Parte

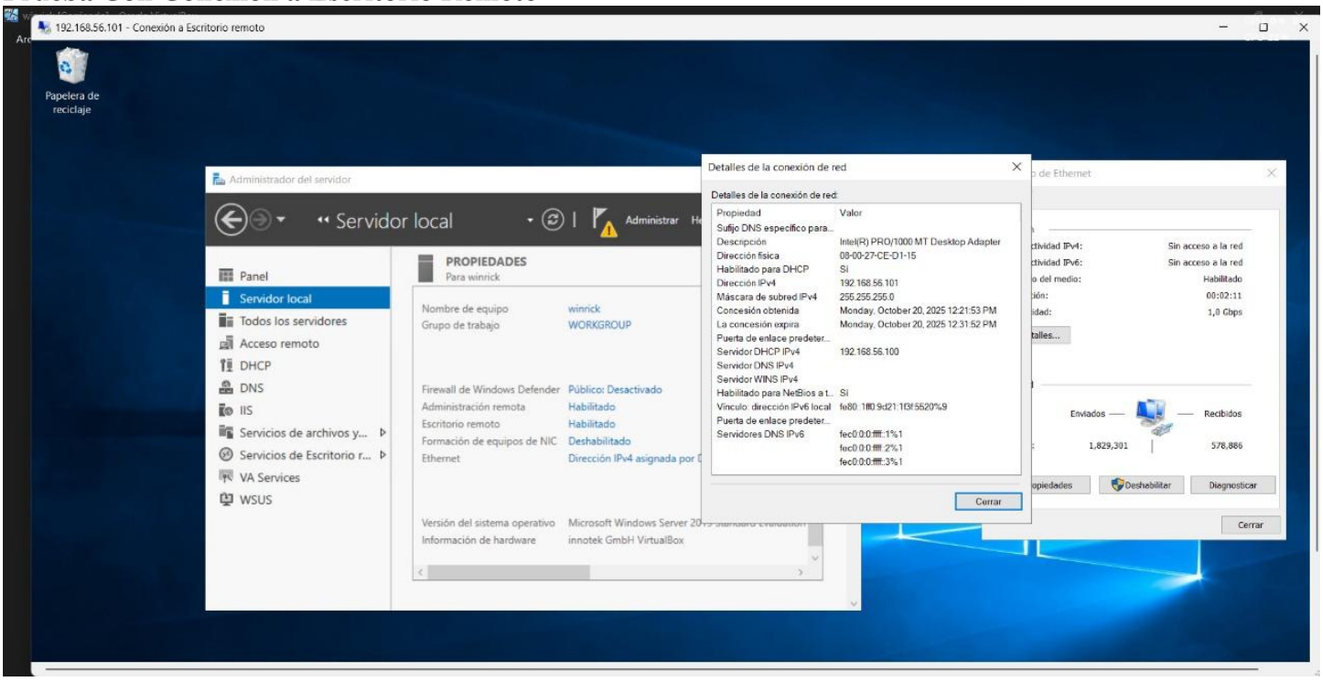
1. Problema: Conversor de Hora (24h a 12h)

1.1. Pseudocódigo

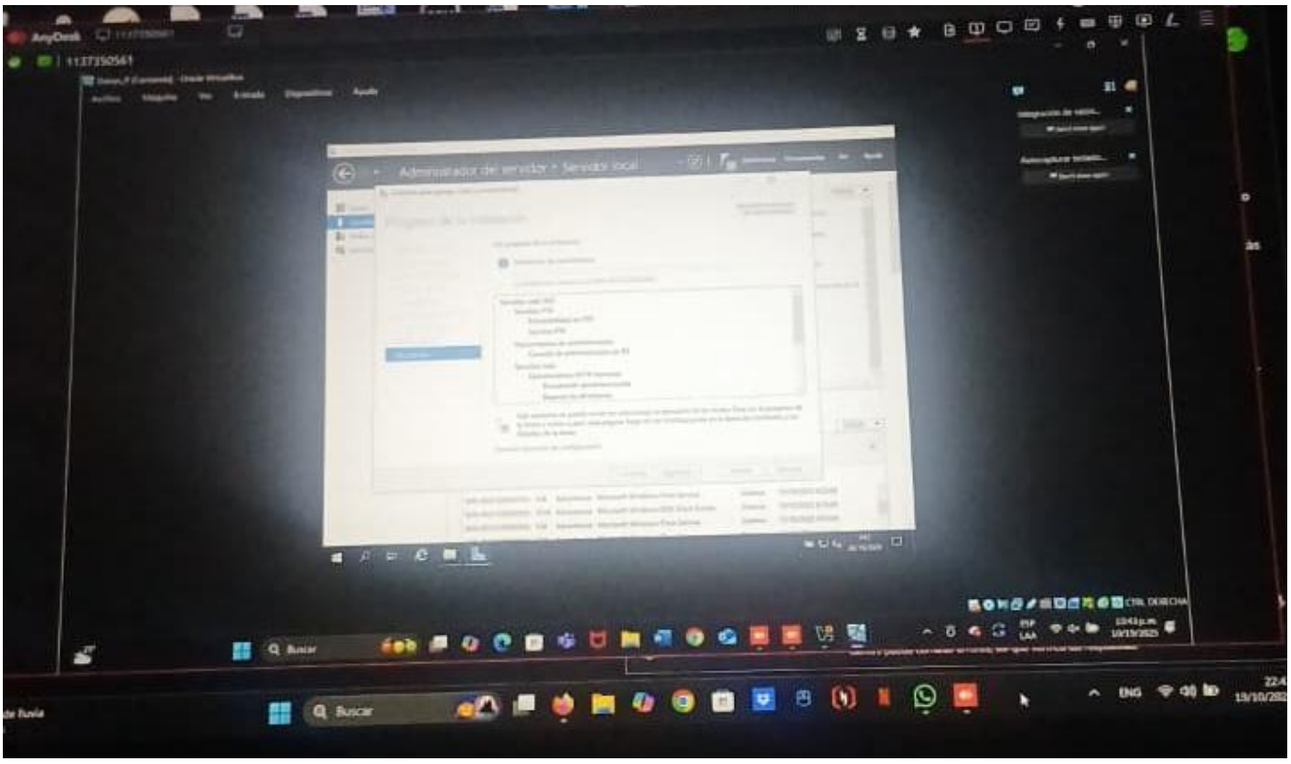
Creacion de la carpeta y permitir compartirla



Prueba Con Conexión a Escritorio Remoto

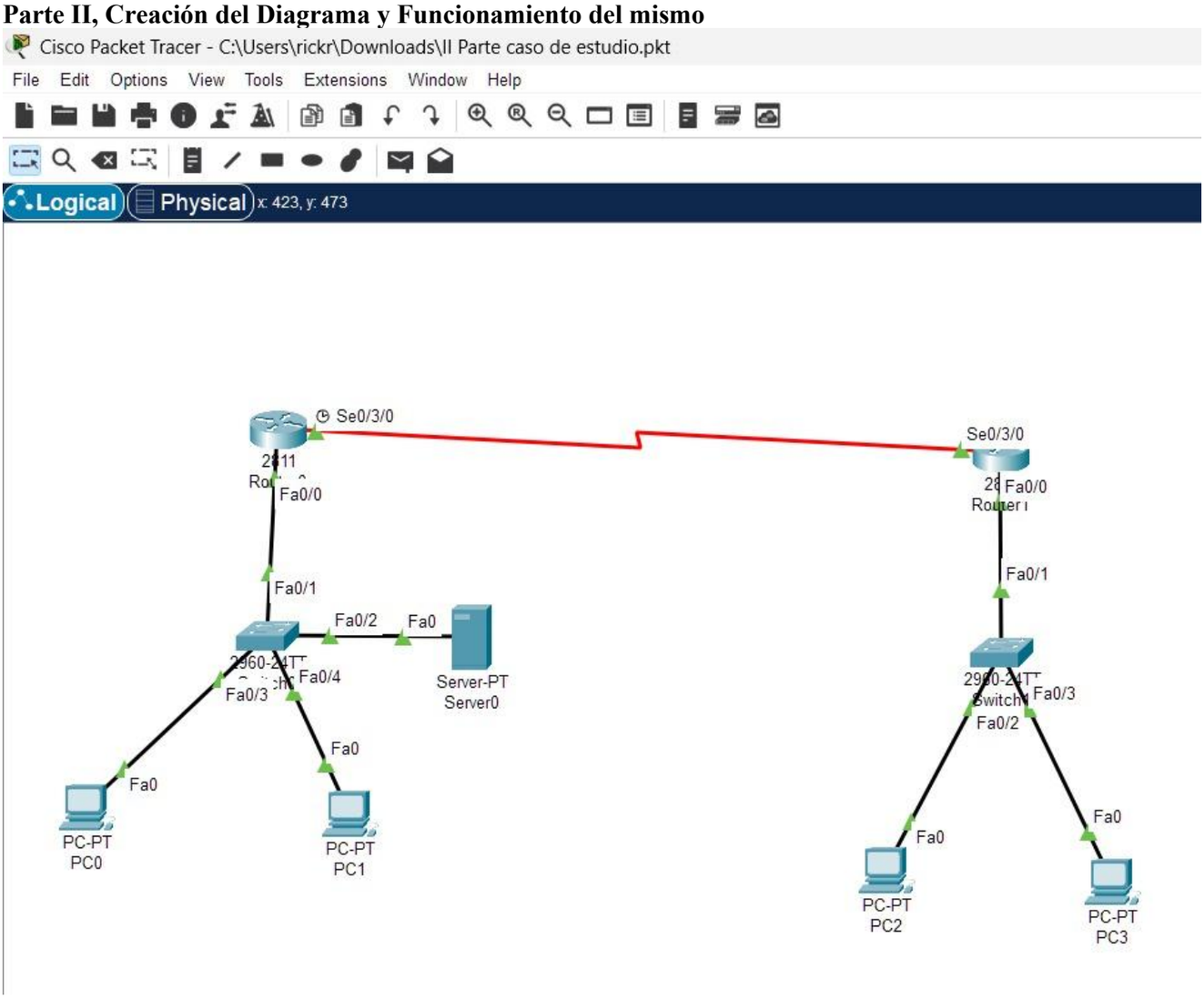


Conexion Con AnyDesk



Conexión Con FTP

Parte II, Creación del Diagrama y Funcionamiento del mismo

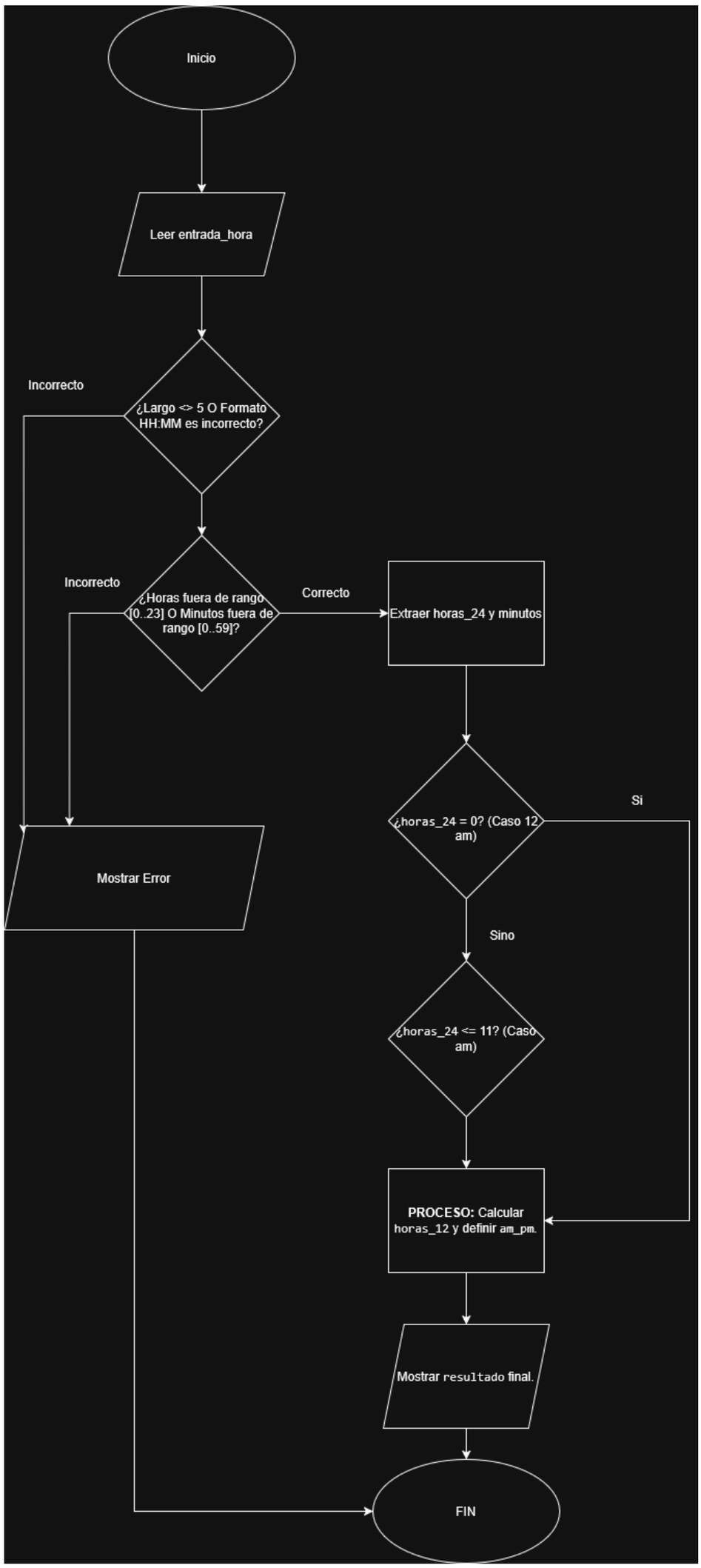


III Parte, Pseudocodigos y Diagramas de Flujo

```
Inicio Algoritmo Conversor_Hora
    DEFINIR entrada_hora como CADENA (Lectura de la GUI)
    DEFINIR horas_24, minutos, horas_12 como ENTERO
    DEFINIR am_pm, resultado como CADENA
    // PASO 1: Validaciones de Formato (HH:MM y 5 caracteres)
    SI LARGO(entrada_hora) <> 5 O SUBSTRING(entrada_hora, 3, 1) <> ":" ENTONCES
        MOSTRAR_ERROR("Formato inválido. Debe ser exactamente HH:MM.")
        RETORNAR
    FIN SI
    // PASO 2: Extracción y Validación de Rango
    horas_24 $\leftarrow$ VALOR_ENTERO(SUBSTRING(entrada_hora, 1, 2))
    minutos $\leftarrow$ VALOR_ENTERO(SUBSTRING(entrada_hora, 4, 2))
    SI horas_24 NO ESTÁ EN [0..23] O minutos NO ESTÁ EN [0..59] ENTONCES
        MOSTRAR_ERROR("Hora o minutos fuera de rango válido.")
        RETORNAR
    FIN SI
    // PASO 3: Lógica de Conversión 24h a 12h
    SI horas_24 = 0 ENTONCES
        horas_12 $\leftarrow$ 12; am_pm $\leftarrow$ "am"
    SINO SI horas_24 ESTÁ EN [1..11] ENTONCES
        horas_12 $\leftarrow$ horas_24; am_pm $\leftarrow$ "am"
    SINO SI horas_24 = 12 ENTONCES
        horas_12 $\leftarrow$ 12; am_pm $\leftarrow$ "pm"
    SINO
        horas_12 $\leftarrow$ horas_24 - 12; am_pm $\leftarrow$ "pm"
    FIN SI
    // PASO 4: Formato y Salida
```

27 minutos_str $\leftarrow$ FORMATO_DOS_DIGITOS(minutos)
28 resultado $\leftarrow$ CONCATENAR(horas_12, ":", minutos_str, " ", am_pm)
29 MOSTRAR_RESULTADO(resultado)
Fin Fin Algoritmo

1.2 Diagrama de Flujo

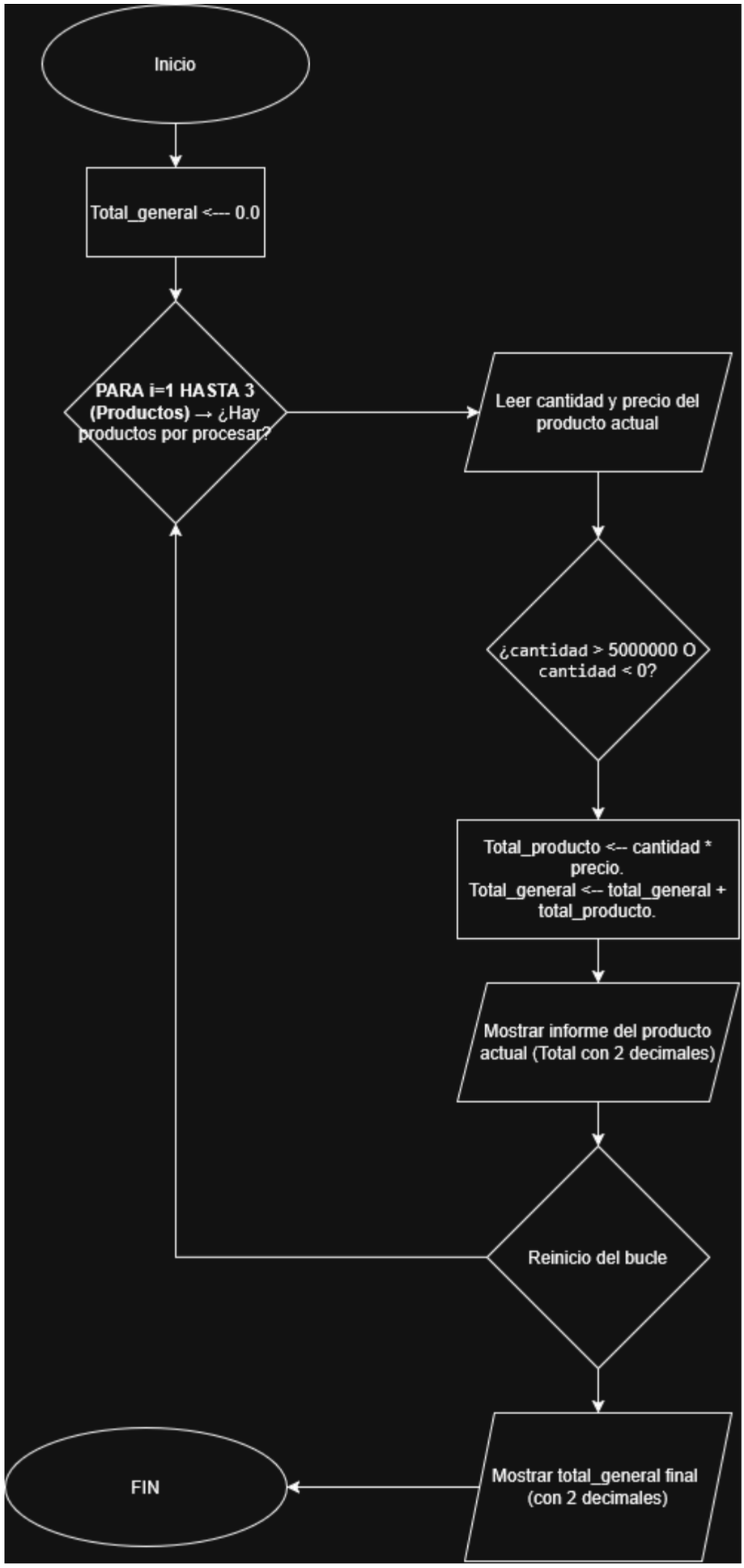


2. Problema: Aplicación de Informe de Ventas

2.1. Pseudocódigo

```
Inicio Algoritmo Informe_Ventas
1       DEFINIR PRODUCTOS como ARREGLO $\leftarrow$ ["Cola", "Naranja", "Limón"]
2       DEFINIR cantidad como ENTERO
3       DEFINIR precio, total_producto, total_general como REAL $\leftarrow 0.0$
4       DEFINIR CANTIDADES_LEIDAS, PRECIOS_LEIDOS como ARREGLOS (Lectura de la GUI)
5       DEFINIR resultados_validos como BOOLEANO $\leftarrow$ VERDADERO
6       // PASO 1: Iterar sobre productos para Calcular y Validar
7       PARA i DESDE 1 HASTA 3 HACER
8           producto_actual $\leftarrow$ PRODUCTOS[i]
9           cantidad $\leftarrow$ VALOR_ENTERO(CANTIDADES_LEIDAS[i])
10          precio $\leftarrow$ VALOR_REAL(PRECIOS_LEIDOS[i])
11          SI cantidad < 0 O cantidad > 5000000 ENTONCES
12              MOSTRAR_ERROR("Cantidad fuera de rango para " + producto_actual + ". Max: 5000000.")
13              resultados_validos $\leftarrow$ FALSO
14              ROMPER CICLO
15          FIN SI
16          total_producto $\leftarrow$ cantidad $\times$ precio
17          total_general $\leftarrow$ total_general + total_producto
18          // Actualizar informe individual (Formato de 2 decimales)
19          ACTUALIZAR_LABEL(total_producto, FORMATO_MONEDA(total_producto))
20      FIN PARA
21      // PASO 2: Mostrar Total General
22      SI resultados_validos ES VERDADERO ENTONCES
23          total_final_formato $\leftarrow$ FORMATO_MONEDA(total_general)
24          ACTUALIZAR_LABEL_TOTAL(total_final_formato)
25      FIN SI
Fin     Fin Algoritmo
```

2.2. Diagrama de Flujo



IV Parte, Códigos en Python

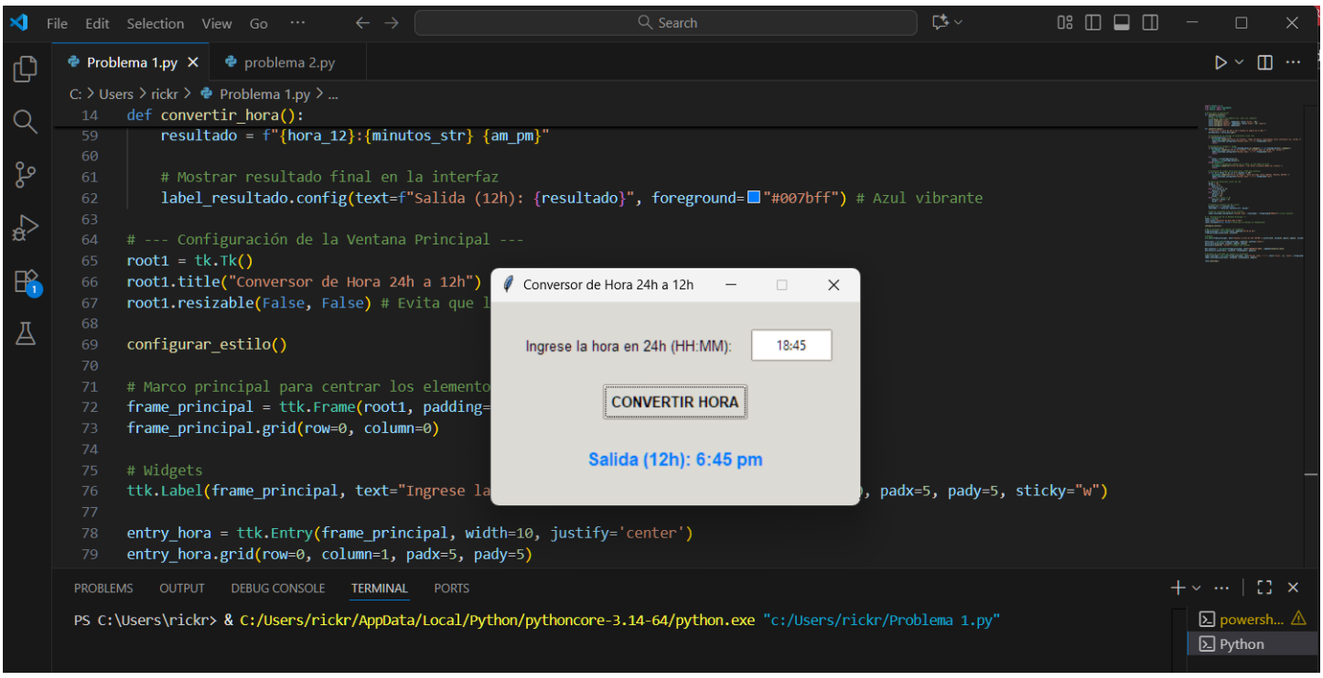
1. Problema: Conversor de Hora (24h a 12h)

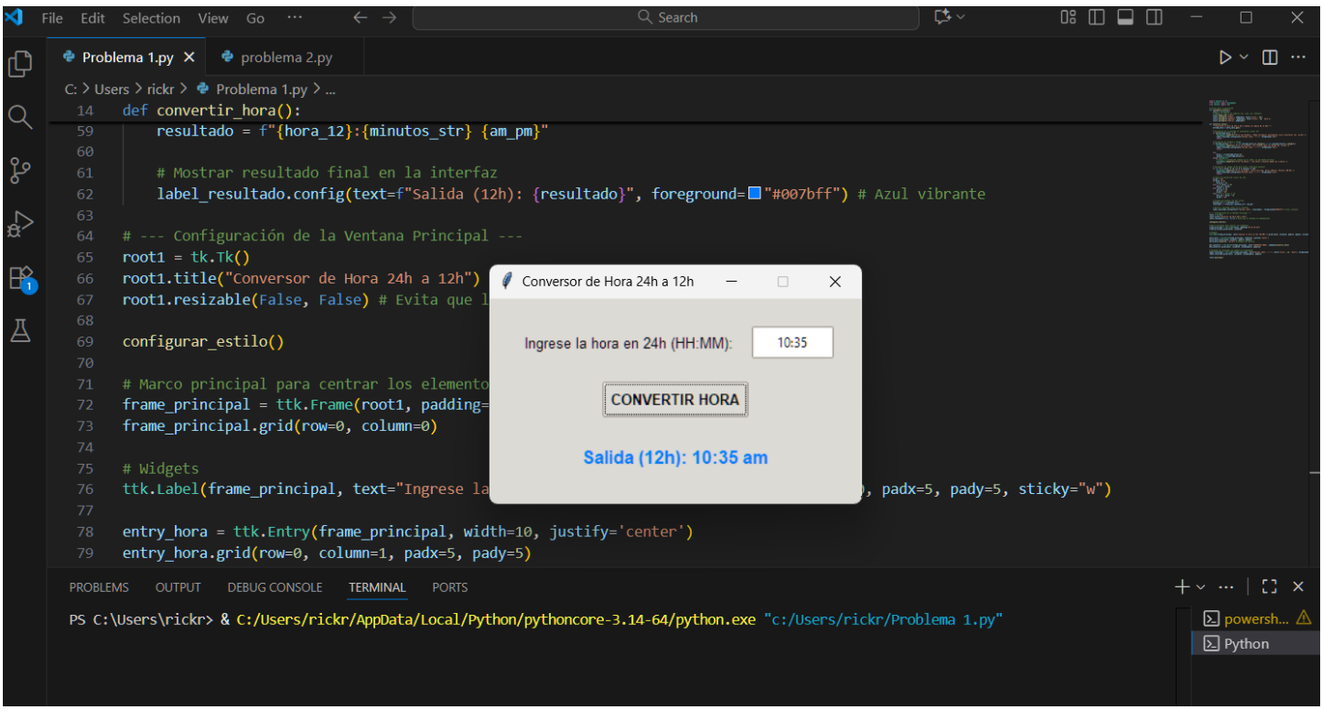
Python

```
import tkinter as tk
from tkinter import messagebox
from tkinter import ttk

# --- Configuración de Estilo ---
def configurar_estilo():
    style = ttk.Style()
    style.theme_use("clam")
    style.configure("TLabel", padding=5, font=('Arial', 10))
    style.configure("TButton", padding=5, font=('Arial', 10, 'bold'))

def convertir_hora():
    """Convierte la hora de 24h a 12h, valida la entrada y actualiza la GUI."""
    entrada_hora = entry_hora.get()

    # 1. Validación de longitud (5 caracteres)
    if len(entrada_hora) != 5:
        messagebox.showerror("Error de Formato", "Debe introducir exactamente cinco caracteres (ej. 13:45).")
        label_resultado.config(text="Salida (12h): --:--", foreground="red")
        return

    # 2. Validación de formato (HH:MM)
    if entrada_hora[2] != ':' or not entrada_hora[:2].isdigit() or not entrada_hora[3:].isdigit():
        messagebox.showerror("Error de Formato", "El formato debe ser HH:MM (ej. 13:45).")
        label_resultado.config(text="Salida (12h): --:--", foreground="red")
        return

    try:
        horas = int(entrada_hora[:2])
        minutos = int(entrada_hora[3:])
    except ValueError:
        messagebox.showerror("Error de Datos", "Las horas y minutos deben ser números.")
        return

    # 3. Validación de rangos (0-23 para horas, 0-59 para minutos)
    if not (0 <= horas <= 23 and 0 <= minutos <= 59):
        messagebox.showerror("Error de Rango", "Hora inválida. Horas (00-23), Minutos (00-59).")
        label_resultado.config(text="Salida (12h): --:--", foreground="red")
        return

    # 4. Lógica de Conversión
    am_pm = "am"
    if horas == 0:
        hora_12 = 12
    elif 1 <= horas <= 11:
        hora_12 = horas
    elif horas == 12:
        hora_12 = 12
        am_pm = "pm"
    else: # 13 <= horas <= 23
        hora_12 = horas - 12
        am_pm = "pm"

    # Formato de minutos con dos cifras
    minutos_str = f"{minutos:02d}"
    resultado = f"{hora_12}:{minutos_str} {am_pm}"

    # Mostrar resultado final en la interfaz
    label_resultado.config(text=f"Salida (12h): {resultado}", foreground="#007bff")

# --- Configuración de la Ventana Principal ---
root1 = tk.Tk()
root1.title("Conversor de Hora 24h a 12h")
root1.resizable(False, False)

configurar_estilo()

# Marco principal para centrar los elementos
frame_principal = ttk.Frame(root1, padding="20 20 20 20")
frame_principal.grid(row=0, column=0)

# Widgets
ttk.Label(frame_principal, text="Ingrese la hora en 24h (HH:MM):").grid(row=0, column=0, padx=5, pady=5, sticky="w")

entry_hora = ttk.Entry(frame_principal, width=10, justify='center')
entry_hora.grid(row=0, column=1, padx=5, pady=5)
```

```
entry_hora.insert(0, "13:45") # Ejemplo por defecto

btn_convertir = ttk.Button(frame_principal, text="CONVERTIR HORA", command=convertir_hora)
btn_convertir.grid(row=1, column=0, columnspan=2, pady=15)

label_resultado = ttk.Label(frame_principal, text="Salida (12h): --:--", font=('Arial', 12, 'bold'), foreground="gray")
label_resultado.grid(row=2, column=0, columnspan=2, pady=5)

root1.mainloop()
```

Prueba de escritorio





2. Problema: Aplicación de Informe de Ventas

Python

```
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox

# Definición de productos y estructuras de datos
PRODUCTOS = ["Cola", "Naranja", "Limón"]
entradas_cantidad = {}
entradas_precio = {}
labels_resultado = {}

# --- Configuración de Estilo ---
def configurar_estilo_ventas():
    style = ttk.Style()
    style.theme_use("clam")
    style.configure("Header.TLabel", font=('Arial', 9, 'bold'), background='#f0f0f0')
    style.configure("Total.TLabel", font=('Arial', 11, 'bold'), foreground='#004d99')

def calcular_ventas():
    """Calcula las ventas, valida las entradas (máx 5000000) y actualiza el informe (2 decimales)."""
    total_general = 0.0
    resultados_validos = True

    # Limpiar o resetear resultados al inicio
    for producto in PRODUCTOS:
        labels_resultado[producto]['cantidad'].config(text="---")
        labels_resultado[producto]['precio'].config(text="---")
        labels_resultado[producto]['total'].config(text="---")
    label_total_final.config(text="0.00")

    for i, producto in enumerate(PRODUCTOS):
        try:
            # 1. Validación de cantidad (máximo 5000000)
            cantidad = int(entradas_cantidad[producto].get())
            if not (0 <= cantidad <= 5000000):
                messagebox.showerror("Error de Cantidad", f"Cantidad de {producto} debe ser entre 0 y 5,000,000.")
                resultados_validos = False
                break

            # 2. Captura y limpieza de precio (permite coma o punto decimal)
            precio_str = entradas_precio[producto].get()
            precio = float(precio_str.replace(',', '.'))

            # 3. Cálculo
            total_producto = cantidad * precio
            total_general += total_producto

            # 4. Formateo de los resultados
            # Formato de cantidad con separador de miles
            cantidad_formato = f"{cantidad:,.0f}".replace(",", "X").replace(".", ",").replace("X", ".")

            # Formato de Precio y Total (2 cifras decimales - Requisito de Balboas)
            precio_formato = f"{precio:.2f}"
            total_formato = f"{total_producto:.2f}"

            # 5. Actualizar la tabla de resultados
            labels_resultado[producto]['cantidad'].config(text=cantidad_formato)
            labels_resultado[producto]['precio'].config(text=precio_formato)
            labels_resultado[producto]['total'].config(text=total_formato)

        except ValueError:
            messagebox.showerror("Error de Entrada", f"Ingrese valores numéricos válidos para Cantidad y Precio de {producto}.")
            resultados_validos = False
            break

    # 6. Mostrar el total general con 2 decimales
    if resultados_validos:
        total_general_formato = f"{total_general:.2f}"
        label_total_final.config(text=total_general_formato)

# --- Configuración de la Ventana Principal ---
root2 = tk.Tk()
root2.title("Aplicación de Informe de Ventas")
root2.resizable(False, False)
configurar_estilo_ventas()

# --- Marco de Entrada (Inputs) ---
frame_entrada = ttk.LabelFrame(root2, text="Ventas Diarias (Cantidad y Precio en B/.)", padding="15 10 15 10")
```

```
frame_entrada.grid(row=0, column=0, padx=10, pady=10, sticky="ew")

# Encabezados de entrada
ttk.Label(frame_entrada, text="Producto", style="Header.TLabel").grid(row=0, column=0, padx=10, pady=5, sticky="w")
ttk.Label(frame_entrada, text="Cantidad Vendida", style="Header.TLabel").grid(row=0, column=1, padx=10, pady=5)
ttk.Label(frame_entrada, text="Precio Unitario", style="Header.TLabel").grid(row=0, column=2, padx=10, pady=5)

# Campos de entrada
for i, producto in enumerate(PRODUCTOS):
    ttk.Label(frame_entrada, text=f"{producto}:").grid(row=i+1, column=0, padx=10, pady=5, sticky="w")

    cantidad_entry = ttk.Entry(frame_entrada, width=15, justify='right')
    cantidad_entry.grid(row=i+1, column=1, padx=10, pady=5)
    entradas_cantidad[producto] = cantidad_entry

    precio_entry = ttk.Entry(frame_entrada, width=10, justify='right')
    precio_entry.grid(row=i+1, column=2, padx=10, pady=5)
    entradas_precio[producto] = precio_entry

# Botón de cálculo
btn_calcular = ttk.Button(root2, text="CALCULAR INFORME", command=calcular_ventas)
btn_calcular.grid(row=1, column=0, pady=10)

# --- Marco de Salida (Output / Reporte) ---
frame_salida = ttk.LabelFrame(root2, text="Informe de Resultados", padding="15 10 15 10")
frame_salida.grid(row=2, column=0, padx=10, pady=10, sticky="ew")

# Encabezados del informe
ttk.Label(frame_salida, text="Producto", style="Header.TLabel").grid(row=0, column=0, padx=5, pady=5, sticky="w")
ttk.Label(frame_salida, text="Ventas", style="Header.TLabel").grid(row=0, column=1, padx=5, pady=5)
ttk.Label(frame_salida, text="Precio (B/.)", style="Header.TLabel").grid(row=0, column=2, padx=5, pady=5)
ttk.Label(frame_salida, text="Total (B/.)", style="Header.TLabel").grid(row=0, column=3, padx=5, pady=5)

# Filas de resultados
for i, producto in enumerate(PRODUCTOS):
    ttk.Label(frame_salida, text=producto).grid(row=i+1, column=0, padx=5, pady=2, sticky="w")

    # Etiquetas de resultados (alineadas a la derecha)
    labels_resultado[producto] = {
        'cantidad': ttk.Label(frame_salida, text="---", anchor="e", width=12, background='#e8f0fe'),
        'precio': ttk.Label(frame_salida, text="---", anchor="e", width=8, background='#e8f0fe'),
        'total': ttk.Label(frame_salida, text="---", anchor="e", width=12, background='#e8f0fe')
    }
    labels_resultado[producto]['cantidad'].grid(row=i+1, column=1, padx=5, pady=2, sticky="e")
    labels_resultado[producto]['precio'].grid(row=i+1, column=2, padx=5, pady=2, sticky="e")
    labels_resultado[producto]['total'].grid(row=i+1, column=3, padx=5, pady=2, sticky="e")

# Fila del Total General
ttk.Label(frame_salida, text="TOTAL", style="Total.TLabel").grid(row=len(PRODUCTOS)+1, column=2, padx=5, pady=10,
sticky="e")
label_total_final = ttk.Label(frame_salida, text="0.00", style="Total.TLabel", anchor="e", width=12)
label_total_final.grid(row=len(PRODUCTOS)+1, column=3, padx=5, pady=10, sticky="e")

root2.mainloop()
```

Prueba de escritorio