Tecnológico de Monterrey

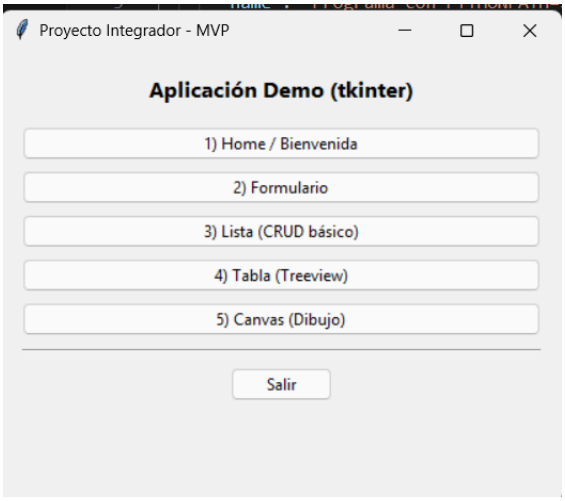
“Construcción del esqueleto del proyecto y primer “Happy Path””

Carlos David Martínez Rocha (A01352717)
Adrian Navarro Romo (A00575101)
Diego Adiel Flores Navarro (A00573953)
Andrés Alejandro Sánchez Rábago
Pedro López Casillas (A00575320)

23 de septiembre del 2025

Tecnológico de Monterrey
Departamento de Ciencias
Pensamiento Computacional para Ingeniería

Main:




```python
import tkinter as tk
from tkinter import ttk
from app.win_home import open_win_home
from app.win_form import open_win_form
from app.win_list import open_win_list
from app.win_table import open_win_table
from app.win_canvas import open_win_canvas

def main():
    root = tk.Tk()
    root.title("Proyecto Integrador - MVP")
    root.geometry("420x340")

    frame = ttk.Frame(root, padding=16)
    frame.pack(fill="both", expand=True)

    ttk.Label(frame, text="Aplicación Demo (tkinter)", font=("Segoe UI", 12, "bold")).pack(pady=(0, 12))
    ttk.Button(frame, text="1) Home / Bienvenida", command=lambda: open_win_home(root)).pack(pady=4, fill="x")
    ttk.Button(frame, text="2) Formulario", command=lambda: open_win_form(root)).pack(pady=4, fill="x")
    ttk.Button(frame, text="3) Lista (CRUD básico)", command=lambda: open_win_list(root)).pack(pady=4, fill="x")
    ttk.Button(frame, text="4) Tabla (Treeview)", command=lambda: open_win_table(root)).pack(pady=4, fill="x")
    ttk.Button(frame, text="5) Canvas (Dibujo)", command=lambda: open_win_canvas(root)).pack(pady=4, fill="x")
    ttk.Separator(frame).pack(pady=6, fill="x")
    ttk.Button(frame, text="Salir", command=root.destroy).pack(pady=6)

    root.mainloop()

if __name__ == "__main__":
    main()
```

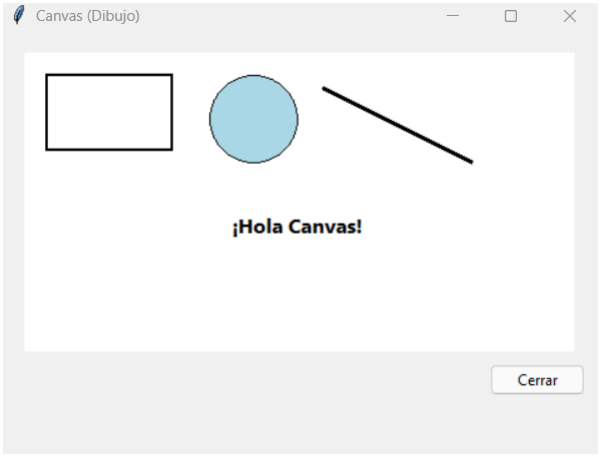
La documentación de este código se trata de la siguiente:

ejemplo de lo que da chat: #
==
Proyecto Integrador - MVP
Aplicación Demo con Tkinter
--
Este archivo define la ventana principal de la aplicación.
Desde aquí se navega hacia diferentes ventanas secundarias:
- Home / Bienvenida
- Formulario
- Lista (CRUD básico)
- Tabla (Treeview)
- Canvas (Dibujo)
==

```
import tkinter as tk            # Librería estándar para GUIs en Python
from tkinter import ttk         # Widgets modernos (estilizados) para Tkinter

# Importación de funciones que abren las ventanas secundarias
from app.win_home import open_win_home
from app.win_form import open_win_form
from app.win_list import open_win_list
from app.win_table import open_win_table
from app.win_canvas import open_win_canvas


def main():
    """
    Función principal que inicializa la ventana raíz (root)
    y construye el menú principal de la aplicación.
    """
    # Crear la ventana principal
    root = tk.Tk()
    root.title("Proyecto Integrador - MVP")   # Título de la ventana
    root.geometry("420x340")                  # Dimensiones iniciales (ancho x alto)

    # Crear un Frame contenedor con padding
    frame = ttk.Frame(root, padding=16)
    frame.pack(fill="both", expand=True)      # Se ajusta al espacio disponible

    # Etiqueta principal (título dentro de la ventana)
    ttk.Label(
        frame,
        text="Aplicación Demo (tkinter)",
        font=("Segoe UI", 12, "bold")
    ).pack(pady=(0, 12))

    # ----------------------------
    # Botones de navegación
    # Cada botón abre una ventana distinta
    # ----------------------------
    ttk.Button(
        frame,
        text="1) Home / Bienvenida",
        command=lambda: open_win_home(root)
    ).pack(pady=4, fill="x")

    ttk.Button(
        frame,
        text="2) Formulario",
        command=lambda: open_win_form(root)
    ).pack(pady=4, fill="x")
```

```
    ttk.Button(
        frame,
        text="3) Lista (CRUD básico)",
        command=lambda: open_win_list(root)
    ).pack(pady=4, fill="x")

    ttk.Button(
        frame,
        text="4) Tabla (Treeview)",
        command=lambda: open_win_table(root)
    ).pack(pady=4, fill="x")

    ttk.Button(
        frame,
        text="5) Canvas (Dibujo)",
        command=lambda: open_win_canvas(root)
    ).pack(pady=4, fill="x")

    # Separador visual entre los botones de navegación y el de salida
    ttk.Separator(frame).pack(pady=6, fill="x")

    # Botón de salida: cierra la aplicación al hacer clic
    ttk.Button(
        frame,
        text="Salir",
        command=root.destroy
    ).pack(pady=6)

    # Bucle principal de la aplicación (mantiene la ventana abierta)
    root.mainloop()


# ------------------------------------------------------------
# Punto de entrada del programa
# ------------------------------------------------------------
if __name__ == "__main__":
    # Llamada a la función principal
    main()
```

Canvas:




```python
import tkinter as tk
from tkinter import ttk

def open_win_canvas(parent: tk.Tk):
    win = tk.Toplevel(parent)
    win.title("Canvas (Dibujo)")
    win.geometry("480x340")

    frm = ttk.Frame(win, padding=12)
    frm.pack(fill="both", expand=True)

    canvas = tk.Canvas(frm, width=440, height=240, bg="white")
    canvas.pack()

    # Dibujos de ejemplo
    canvas.create_rectangle(20, 20, 120, 80, outline="black", width=2)
    canvas.create_oval(150, 20, 220, 90, fill="lightblue")
    canvas.create_line(240, 30, 360, 90, width=3)
    canvas.create_text(220, 140, text="¡Hola Canvas!", font=("Segoe UI", 12, "bold"))

    ttk.Button(frm, text="Cerrar", command=win.destroy).pack(pady=8, anchor="e")
```

La documentación de este código se trata de la siguiente:

```
# ============================================================
# Módulo: win_canvas.py
# ------------------------------------------------------------
# Este archivo define la ventana secundaria "Canvas (Dibujo)".
# La ventana contiene un área de dibujo con ejemplos de figuras
# geométricas y un botón para cerrarla.
# ============================================================

import tkinter as tk            # Librería base para la interfaz gráfica
from tkinter import ttk         # Versión moderna de los widgets

def open_win_canvas(parent: tk.Tk):
    """
    Crea y abre una nueva ventana secundaria (Toplevel)
    que incluye un canvas para realizar dibujos simples.

    Parámetros:
```

```
    parent (tk.Tk): La ventana principal desde la cual se abre.
"""
# Crear ventana secundaria (hija de la ventana principal)
win = tk.Toplevel(parent)
win.title("Canvas (Dibujo)")       # Título de la ventana
win.geometry("480x340")            # Tamaño inicial de la ventana

# Frame contenedor con padding
frm = ttk.Frame(win, padding=12)
frm.pack(fill="both", expand=True)

# Crear el widget Canvas (área de dibujo)
canvas = tk.Canvas(
    frm,
    width=440,      # Ancho en píxeles
    height=240,     # Alto en píxeles
    bg="white"      # Color de fondo
)
canvas.pack()

# ============================================================
# Dibujos de ejemplo dentro del Canvas
# ------------------------------------------------------------
# 1. Rectángulo
canvas.create_rectangle(
    20, 20, 120, 80,     # Coordenadas: x1, y1, x2, y2
    outline="black",     # Color del borde
    width=2              # Grosor de línea
)

# 2. Óvalo
canvas.create_oval(
    150, 20, 220, 90,    # Coordenadas del cuadro envolvente
    fill="lightblue"     # Color de relleno
)

# 3. Línea
canvas.create_line(
    240, 30, 360, 90,    # Puntos de inicio y fin (x1, y1, x2, y2)
    width=3              # Grosor de línea
)

# 4. Texto
canvas.create_text(
    220, 140,                    # Coordenadas (x, y)
    text="¡Hola Canvas!",            # Texto a mostrar
    font=("Segoe UI", 12, "bold")    # Fuente y estilo
)
```

```
# ============================================================

# Botón para cerrar la ventana secundaria
ttk.Button(
    frm,
    text="Cerrar",
    command=win.destroy
).pack(pady=8, anchor="e")          #
```

Formulario:


```python
import tkinter as tk
from tkinter import ttk, filedialog, messagebox

def open_win_form(parent: tk.Tk):
    win = tk.Toplevel(parent)
    win.title("Formulario")
    win.geometry("420x260")
    frm = ttk.Frame(win, padding=16)
    frm.pack(fill="both", expand=True)

    ttk.Label(frm, text="Nombre:").grid(row=0, column=0, sticky="w")
    ent_nombre = ttk.Entry(frm, width=28)
    ent_nombre.grid(row=0, column=1, pady=4)

    ttk.Label(frm, text="Edad:").grid(row=1, column=0, sticky="w")
    ent_edad = ttk.Entry(frm, width=10)
    ent_edad.grid(row=1, column=1, sticky="w", pady=4)

    def validar_y_guardar():
        nombre = ent_nombre.get().strip()
        edad_txt = ent_edad.get().strip()
        if not nombre:
            messagebox.showerror("Error", "El nombre es requerido.")
            return
        if not edad_txt.isdigit():
            messagebox.showerror("Error", "La edad debe ser un número entero.")
            return
        ruta = filedialog.asksaveasfilename(defaultextension=".txt",
                                            filetypes=[("Texto", "*.txt")])
        if ruta:
            with open(ruta, "w", encoding="utf-8") as f:
                f.write(f"Nombre: {nombre}\nEdad: {edad_txt}\n")
            messagebox.showinfo("OK", "Datos guardados.")

    ttk.Button(frm, text="Guardar", command=validar_y_guardar)\
        .grid(row=3, column=0, pady=12)
    ttk.Button(frm, text="Cerrar", command=win.destroy)\
        .grid(row=3, column=1, sticky="e", pady=12)
```

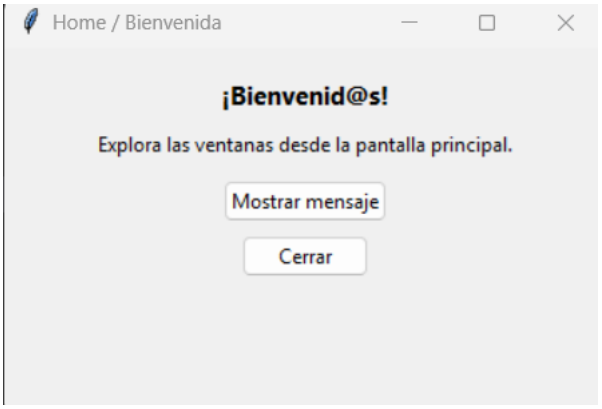
La documentación de este código se trata de la siguiente:

==
Módulo: win_form.py
--
Este archivo define la ventana secundaria "Formulario".
Permite al usuario ingresar un nombre y una edad,
validar los datos, y guardarlos en un archivo de texto.
==

```
import tkinter as tk                        # Librería base para GUIs
from tkinter import ttk, filedialog, messagebox  # Widgets modernos y diálogos estándar


def open_win_form(parent: tk.Tk):
    """
    Crea y abre una nueva ventana secundaria (Toplevel)
    que contiene un formulario simple para capturar
    nombre y edad, con validación y guardado en archivo.

    Parámetros:
        parent (tk.Tk): La ventana principal desde la cual se abre.
    """
    # Crear ventana secundaria
    win = tk.Toplevel(parent)
    win.title("Formulario")          # Título de la ventana
    win.geometry("420x260")          # Tamaño inicial

    # Frame principal con padding
    frm = ttk.Frame(win, padding=16)
    frm.pack(fill="both", expand=True)

    # ============================================================
    # Campo: Nombre
    # ------------------------------------------------------------
    ttk.Label(frm, text="Nombre:").grid(row=0, column=0, sticky="w")
    ent_nombre = ttk.Entry(frm, width=28)      # Caja de texto para nombre
    ent_nombre.grid(row=0, column=1, pady=4)

    # Campo: Edad
    ttk.Label(frm, text="Edad:").grid(row=1, column=0, sticky="w")
    ent_edad = ttk.Entry(frm, width=10)        # Caja de texto para edad
    ent_edad.grid(row=1, column=1, sticky="w", pady=4)
    # ============================================================

    def validar_y_guardar():
        """
        Función interna que valida los datos ingresados
        y los guarda en un archivo de texto.
        """
        # Obtener valores ingresados
        nombre = ent_nombre.get().strip()
        edad_txt = ent_edad.get().strip()

        # Validaciones básicas
        if not nombre:
            messagebox.showerror("Error", "El nombre es requerido.")
```

```
    return
if not edad_txt.isdigit():
    messagebox.showerror("Error", "La edad debe ser un número entero.")
    return

# Diálogo para seleccionar ubicación de guardado
ruta = filedialog.asksaveasfilename(
    defaultextension=".txt",           # Extensión por defecto
    filetypes=[("Texto", "*.txt")]     # Filtro de tipos de archivo
)

#
```

Home:




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

def open_win_home(parent: tk.Tk):
    win = tk.Toplevel(parent)
    win.title("Home / Bienvenida")
    win.geometry("360x220")
    frm = ttk.Frame(win, padding=16)
    frm.pack(fill="both", expand=True)

    ttk.Label(frm, text="¡Bienvenid@s!", font=("Segoe UI", 11, "bold")).pack(pady=(0, 8))
    ttk.Label(frm, text="Explora las ventanas desde la pantalla principal.").pack(pady=(0, 12))
    ttk.Button(frm, text="Mostrar mensaje",
               command=lambda: messagebox.showinfo("Info", "¡Equipo listo!")).pack()
    ttk.Button(frm, text="Cerrar", command=win.destroy).pack(pady=8)
```

La documentación de este código se trata de la siguiente:

==
Módulo: win_home.py
--
Este archivo define la ventana secundaria "Home / Bienvenida".
Es la pantalla de bienvenida de la aplicación y permite
mostrar un mensaje informativo al usuario.
==

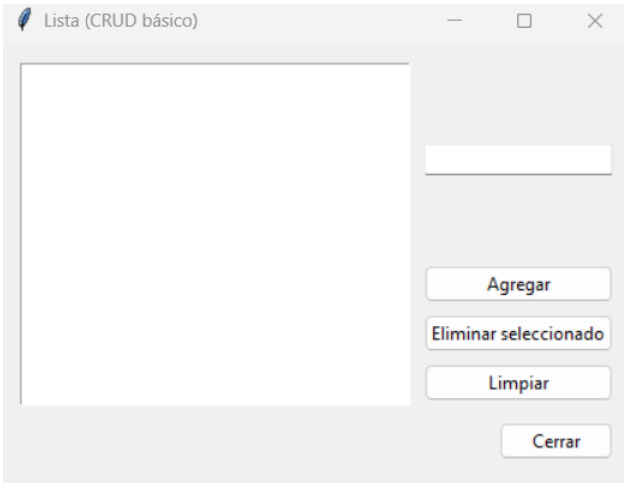
import tkinter as tk # Librería base para GUIs
from tkinter import ttk, messagebox # Widgets modernos y diálogos estándar

def open_win_home(parent: tk.Tk):
    """
    Crea y abre una nueva ventana secundaria (Toplevel)
    que sirve como pantalla de bienvenida.

    Parámetros:
        parent (tk.Tk): La ventana principal desde la cual se abre.
    """
    # Crear ventana secundaria

```
win = tk.Toplevel(parent)
win.title("Home / Bienvenida")    # Título de la ventana
win.geometry("360x220")           # Tamaño inicial

# Frame contenedor con padding
frm = ttk.Frame(win, padding=16)
frm.pack(fill="both", expand=True)

# ============================================================
# Etiquetas de bienvenida
# ------------------------------------------------------------
ttk.Label(
    frm,
    text="¡Bienvenid@s!",
    font=("Segoe UI", 11, "bold")   # Fuente y estilo
).pack(pady=(0, 8))

ttk.Label(
    frm,
    text="Explora las ventanas desde la pantalla principal."
).pack(pady=(0, 12))
# ============================================================

# Botón para mostrar un mensaje informativo
ttk.Button(
    frm,
    text="Mostrar mensaje",
    command=lambda: messagebox.showinfo("Info", "¡Equipo listo!")
).pack()

# Botón para cerrar la ventana
ttk.Button(
    frm,
    text="Cerrar",
    command=win.destroy
).pack(pady=8)
```

Lista:




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

def open_win_list(parent: tk.Tk):
    win = tk.Toplevel(parent)
    win.title("Lista (CRUD básico)")
    win.geometry("420x300")

    frm = ttk.Frame(win, padding=12)
    frm.pack(fill="both", expand=True)

    lb = tk.Listbox(frm, height=10)
    lb.grid(row=0, column=0, rowspan=4, sticky="nsew", padx=(0, 8))
    frm.columnconfigure(0, weight=1)
    frm.rowconfigure(0, weight=1)

    ent_item = ttk.Entry(frm)
    ent_item.grid(row=0, column=1, sticky="ew")

    def agregar():
        v = ent_item.get().strip()
        if v:
            lb.insert("end", v)
            ent_item.delete(0, "end")
        else:
            messagebox.showwarning("Aviso", "Escribe un texto para agregar.")

    def eliminar():
        sel = lb.curselection()
        if sel:
            lb.delete(sel[0])

    def limpiar():
        lb.delete(0, "end")

    ttk.Button(frm, text="Agregar", command=agregar).grid(row=1, column=1, sticky="ew", pady=4)
    ttk.Button(frm, text="Eliminar seleccionado", command=eliminar).grid(row=2, column=1, sticky="ew", pady=4)
    ttk.Button(frm, text="Limpiar", command=limpiar).grid(row=3, column=1, sticky="ew", pady=4)

    ttk.Button(frm, text="Cerrar", command=win.destroy).grid(row=4, column=0, columnspan=2, pady=10, sticky="e")
```

La documentación de este código se trata de la siguiente:

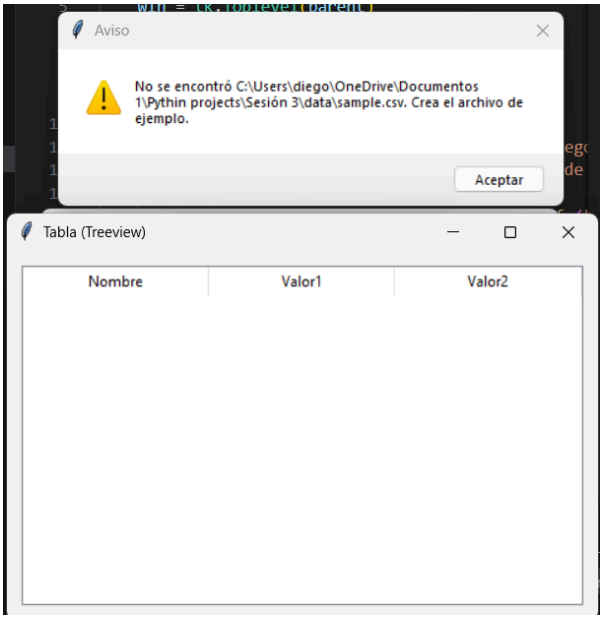
==
Módulo: win_list.py
--
Este archivo define la ventana secundaria "Lista (CRUD básico)".
Permite al usuario agregar, eliminar y limpiar elementos
de una lista usando un Listbox, simulando operaciones CRUD simples.
==

import tkinter as tk # Librería base para GUIs
from tkinter import ttk, messagebox # Widgets modernos y diálogos estándar

```
def open_win_list(parent: tk.Tk):
    """
    Crea y abre una nueva ventana secundaria (Toplevel)
    con un Listbox y operaciones básicas de CRUD.

    Parámetros:
        parent (tk.Tk): La ventana principal desde la cual se abre.
    """
    # Crear ventana secundaria
    win = tk.Toplevel(parent)
    win.title("Lista (CRUD básico)")   # Título de la ventana
    win.geometry("420x300")            # Tamaño inicial

    # Frame contenedor con padding
    frm = ttk.Frame(win, padding=12)
    frm.pack(fill="both", expand=True)

    # ============================================================
    # Listbox (lista de elementos)
    # ------------------------------------------------------------
    lb = tk.Listbox(frm, height=10)
    lb.grid(row=0, column=0, rowspan=4, sticky="nsew", padx=(0, 8))

    # Configurar pesos para que la columna/filas se expandan
    frm.columnconfigure(0, weight=1)
    frm.rowconfigure(0, weight=1)

    # Entry para ingresar nuevos elementos
    ent_item = ttk.Entry(frm)
    ent_item.grid(row=0, column=1, sticky="ew")
    # ============================================================

    # ============================================================
    # Funciones CRUD básicas
    # ------------------------------------------------------------
    def agregar():
        """Agrega el texto del Entry a la lista."""
        v = ent_item.get().strip()
        if v:
            lb.insert("end", v)        # Insertar al final
            ent_item.delete(0, "end")   # Limpiar Entry
        else:
            messagebox.showwarning("Aviso", "Escribe un texto para agregar.")

    def eliminar():
        """Elimina el elemento seleccionado de la lista."""
        sel = lb.curselection()
```

```
    if sel:
        lb.delete(sel[0])

def limpiar():
    """Limpia todos los elementos de la lista."""
    lb.delete(0, "end")
# ============================================================

# ============================================================
# Botones de acción
# ------------------------------------------------------------
ttk.Button(frm, text="Agregar", command=agregar)\
    .grid(row=1, column=1, sticky="ew", pady=4)

ttk.Button(frm, text="Eliminar seleccionado", command=eliminar)\
    .grid(row=2, column=1, sticky="ew", pady=4)

ttk.Button(frm, text="Limpiar", command=limpiar)\
    .grid(row=3, column=1, sticky="ew", pady=4)

# Botón para cerrar la ventana
ttk.Button(frm, text="Cerrar", command=win.destroy)\
    .grid(row=4, column=0, columnspan=2, pady=10, sticky="e")
# ============================================================
```

Tabla:




```python
import tkinter as tk
from tkinter import ttk, messagebox
import csv
from pathlib import Path

def open_win_table(parent: tk.Tk):
    win = tk.Toplevel(parent)
    win.title("Tabla (Treeview)")
    win.geometry("480x300")
    frm = ttk.Frame(win, padding=12)
    frm.pack(fill="both", expand=True)

    cols = ("nombre", "valor1", "valor2")
    tv = ttk.Treeview(frm, columns=cols, show="headings", height=10)
    for c in cols:
        tv.heading(c, text=c.capitalize())
        tv.column(c, width=120, anchor="center")
    tv.pack(fill="both", expand=True)

    ruta = Path(__file__).resolve().parents[2] / "data" / "sample.csv"  # <repo>/data/sample.csv
    if not ruta.exists():
        messagebox.showwarning("Aviso", f"No se encontró {ruta}. Crea el archivo de ejemplo.")
        return

    with open(ruta, "r", encoding="utf-8") as f:
        reader = csv.DictReader(f)
        for row in reader:
            tv.insert("", "end", values=(row["nombre"], row["valor1"], row["valor2"]))

    ttk.Button(frm, text="Cerrar", command=win.destroy).pack(pady=8, anchor="e")
```

La documentación de este código se trata de la siguiente:

==
Módulo: win_table.py
--
Este archivo define la ventana secundaria "Tabla (Treeview)".
Permite mostrar datos tabulares cargados desde un archivo CSV
usando el widget Treeview de Tkinter.
==

import tkinter as tk # Librería base para GUIs

```
from tkinter import ttk, messagebox   # Widgets modernos y diálogos estándar
import csv                            # Lectura de archivos CSV
from pathlib import Path              # Gestión de rutas de archivos


def open_win_table(parent: tk.Tk):
    """
    Crea y abre una nueva ventana secundaria (Toplevel)
    que muestra datos tabulares en un Treeview a partir
    de un archivo CSV.

    Parámetros:
        parent (tk.Tk): La ventana principal desde la cual se abre.
    """
    # Crear ventana secundaria
    win = tk.Toplevel(parent)
    win.title("Tabla (Treeview)")      # Título de la ventana
    win.geometry("480x300")            # Tamaño inicial

    # Frame contenedor con padding
    frm = ttk.Frame(win, padding=12)
    frm.pack(fill="both", expand=True)

    # ============================================================
    # Configuración del Treeview
    # ------------------------------------------------------------
    cols = ("nombre", "valor1", "valor2")    # Nombres de columnas
    tv = ttk.Treeview(frm, columns=cols, show="headings", height=10)

    # Configurar encabezados y columnas
    for c in cols:
        tv.heading(c, text=c.capitalize())   # Encabezado visible
        tv.column(c, width=120, anchor="center")  # Ancho y alineación

    tv.pack(fill="both", expand=True)
    # ============================================================

    # ============================================================
    # Cargar datos desde CSV
    # ------------------------------------------------------------
    ruta = Path(__file__).resolve().parents[2] / "data" / "sample.csv"
    # Se espera que el archivo esté en: <repo>/data/sample.csv

    if not ruta.exists():
        # Si no se encuentra el archivo, mostrar advertencia
        messagebox.showwarning(
            "Aviso",
            f"No se encontró {ruta}. Crea el archivo de ejemplo."
```

```
    )
    return

# Abrir y leer el CSV
with open(ruta, "r", encoding="utf-8") as f:
    reader = csv.DictReader(f)
    for row in reader:
        tv.insert(
            "",
            "end",
            values=(row["nombre"], row["valor1"], row["valor2"])
        )
# ============================================================

# Botón para cerrar la ventana
ttk.Button(frm, text="Cerrar", command=win.destroy)\
    .pack(pady=8, anchor="e")
```

data/sample: (este es un archivo **sample.csv** que se usa en win_table.py.)

src > data > sample.csv

```
1   nombre,valor1,valor2
2   A,10,20
3   B,15,25
4   C,12,30
```

Explicacion:

nombre: Identificador de la fila (puede ser texto o código).
valor1 y valor2: Valores numéricos de ejemplo que se muestran en la tabla (Treeview).

Uso en la aplicación:

win_table.py lee este archivo usando csv.DictReader.
Cada fila se inserta como una entrada en el Treeview.
Si el archivo no existe, se muestra un mensaje de advertencia al usuario.
.vscode/ launch.json:

.vscode/Launch.json:

Configura la **depuración y ejecución** de Python en VS Code para el proyecto.
Permite ejecutar la aplicación principal (main.py) con las rutas correctas y variables de entorno necesarias.

.vscode > {} launch.json > ...

```
1   {
2     "version": "0.2.0",
3     "configurations": [
4       {
5         "name": "Programa con PYTHONPATH=src",
6         "type": "python",
7         "request": "launch",
8         "program": "${workspaceFolder}/src/app/main.py",
9         "cwd": "${workspaceFolder}",
10        "env": { "PYTHONPATH": "${workspaceFolder}/src" },
11        "console": "integratedTerminal",
12        "justMyCode": true
13      }
14    ]
15  }
```

"""
==========
Archivo: launch.json (VS Code)

Este archivo configura la depuración y ejecución del proyecto
Python en VS Code. Permite ejecutar el programa principal
main.py asegurando que todos los módulos en src/app/ sean
importables mediante PYTHONPATH.

Versión del esquema: 0.2.0
==========

Ejemplo de configuración:

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Programa con PYTHONPATH=src",
      "type": "python",
      "request": "launch",
      "program": "${workspaceFolder}/src/app/main.py",
      "cwd": "${workspaceFolder}",
      "env": { "PYTHONPATH": "${workspaceFolder}/src" },
      "console": "integratedTerminal",
      "justMyCode": true
    }
  ]
}
```

Explicación de campos:

- version: versión de la configuración de VS Code.
- configurations: lista de configuraciones de ejecución o depuración.
- name: nombre descriptivo para elegir la configuración.
- type: tipo de depurador; "python" para proyectos Python.
- request: tipo de solicitud; "launch" inicia el programa.
- program: ruta al script principal (main.py).
- cwd: carpeta de trabajo actual.
- env: variables de entorno; PYTHONPATH asegura que se puedan importar módulos desde src/.
- console: tipo de terminal donde se ejecuta el programa (integratedTerminal abre la terminal integrada de VS Code).
- justMyCode: si es True, el depurador se centra solo en el código del usuario, ignorando librerías externas.

Recomendación:
Mantener el archivo en .vscode/ para que VS Code lo reconozca automáticamente.
"""

Roles de los 5 integrantes

Persona	Matrícula	Aportación
Diego Adiel Flores Navarro	A00573953	**Preparación inicial**: Creó el repositorio en GitHub y subió la estructura base del proyecto.
Carlos David Martínez Rocha	A01352717	**Organización de carpetas y archivos**: armó y organizó las carpetas y los archivos donde se pegaría el código proporcionado.
Adrián Navarro Romo	A00575101	**Acomodación del código**: Pegó y ajustó el código de las ventanas 1, 2 y 5 dentro de sus archivos y revisó que cada ventana se llamará correctamente.
Pedro López Casillas	A00575320	**Configuración de ejecución en VS Code**: creó y probó el código, verificando que se pueda depurar con F5 sin errores de import. También organizó el contenido dentro del repositorio de GitHub.

Andrés Alejandro Sánchez Rábago	A00574862	**Evidencia y documentación**: redactó este documento en, organizó las capturas y escribió las instrucciones de ejecución y la explicación de la estructura.