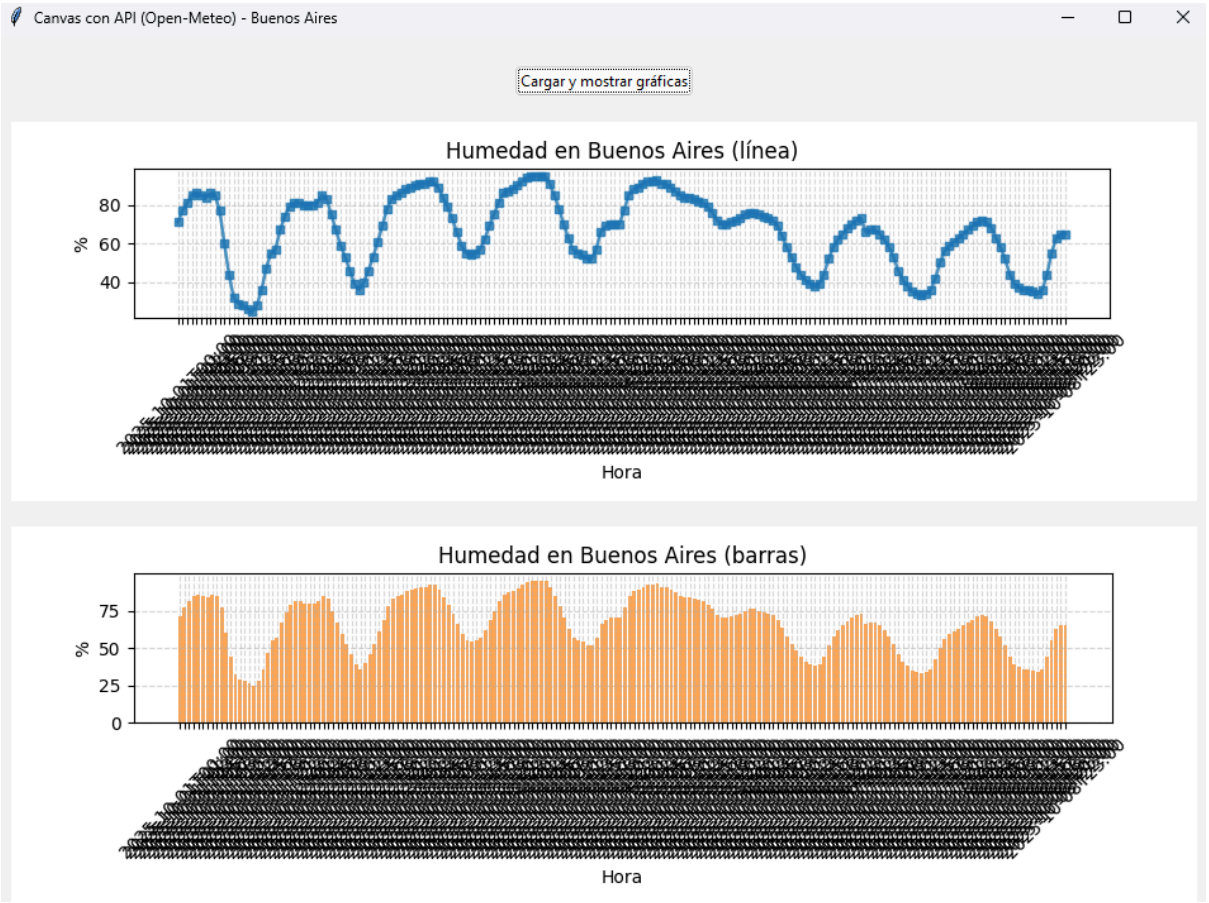
“SESIÓN 5”

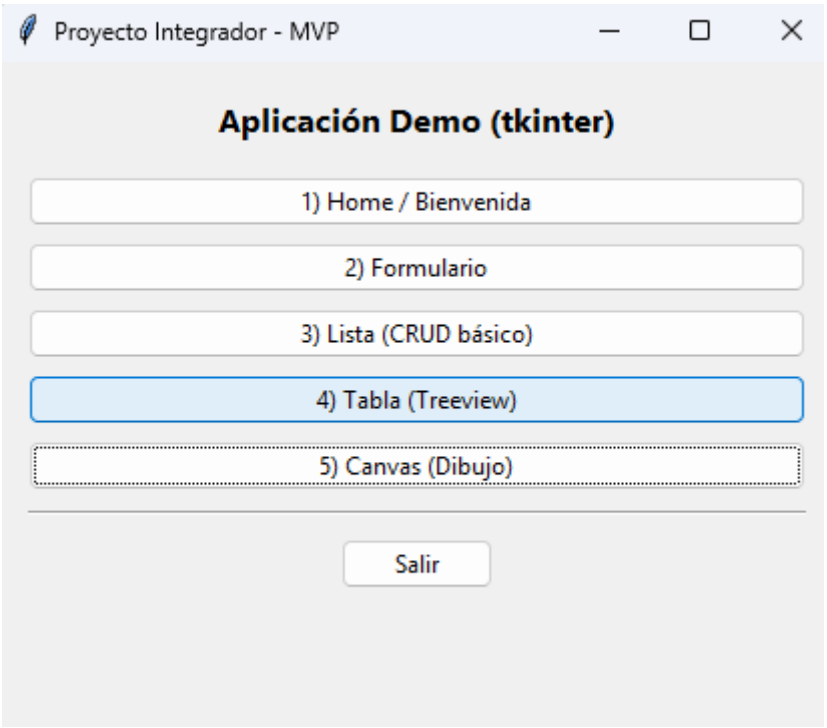
Adaptación de ventanas al diseño del proyecto

Hecho por : Pedro López Casillas

Maestro: Camilo René Duque Becerra

Jueves 2 de Septiembre del 2025





CÓDIGO ORIGINAL :

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

```
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg


def fetch_data():
    """
    Conecta con la API de Open-Meteo y obtiene temperaturas horarias
    de León, Gto (últimas 24 horas).
    Devuelve dos listas: horas y temperaturas.
    """
    try:
        url = (
            "https://api.open-meteo.com/v1/forecast"
            "?latitude=21.12&longitude=-101.68"
            "&hourly=temperature_2m&past_days=1"
            "&timezone=auto"
        )
        response = requests.get(url, timeout=15)
        response.raise_for_status()
        data = response.json()

        horas = data["hourly"]["time"]
        temperaturas = data["hourly"]["temperature_2m"]

        return horas, temperaturas
    except Exception as e:
        messagebox.showerror("Error", f"No se pudieron obtener los datos:\\n{e}")
        return [], []


def create_line_chart(horas, temps):
    """Gráfica de línea."""
    fig, ax = plt.subplots(figsize=(6, 3))
    ax.plot(horas, temps, linestyle="-", marker="o", markersize=3)
    ax.set_title("Temperatura en León (línea)")
    ax.set_xlabel("Hora")
    ax.set_ylabel("°C")
    ax.tick_params(axis="x", rotation=45)
    fig.tight_layout()
    return fig


def create_bar_chart(horas, temps):
    """Gráfica de barras."""
    fig, ax = plt.subplots(figsize=(6, 3))
    ax.bar(horas, temps)
    ax.set_title("Temperatura en León (barras)")
    ax.set_xlabel("Hora")
    ax.set_ylabel("°C")
    ax.tick_params(axis="x", rotation=45)
    fig.tight_layout()
    return fig


def mostrar_graficas(frm, horas, temps):
    """Inserta las tres gráficas en el frame de la ventana tkinter."""
    # Línea
    fig1 = create_line_chart(horas, temps)
```

```
    canvas1 = FigureCanvasTkAgg(fig1, master=frm)
    canvas1.draw()
    canvas1.get_tk_widget().pack(pady=10, fill="x")

    # Barras
    fig2 = create_bar_chart(horas, temps)
    canvas2 = FigureCanvasTkAgg(fig2, master=frm)
    canvas2.draw()
    canvas2.get_tk_widget().pack(pady=10, fill="x")


def open_win_canvas(parent: tk.Tk):
    """
    Crea la ventana secundaria con gráficas de la API.
    """
    win = tk.Toplevel(parent)
    win.title("Canvas con API (Open-Meteo) y gráficas")
    win.geometry("960x1000")

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

    # Botón para cargar datos y graficar
    def cargar():
        horas, temps = fetch_data()
        if horas and temps:
            mostrar_graficas(frm, horas, temps)

    ttk.Button(frm, text="Cargar y mostrar gráficas", command=cargar).pack(pady=10)


# Para pruebas independientes (opcional)
if __name__ == "__main__":
    root = tk.Tk()
    root.title("Prueba win_canvas")
    ttk.Button(root, text="Abrir ventana Canvas", command=lambda:
open_win_canvas(root)).pack(pady=20)
    root.mainloop()
```

CÓDIGO CAMBIADO:

```
import tkinter as tk
from tkinter import ttk, messagebox
import requests
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg


def fetch_data():
    """
    Conecta con la API de Open-Meteo y obtiene humedad relativa y viento
    de Buenos Aires (últimas 24 horas).
    Devuelve horas, humedad y viento.
    """
    try:
        url = (
            "https://api.open-meteo.com/v1/forecast"
            "?latitude=-34.61&longitude=-58.38"   # Buenos Aires
            "&hourly=relativehumidity_2m,windspeed_10m"
            "&past_days=1"
            "&timezone=auto"
        )
        response = requests.get(url, timeout=15)
        response.raise_for_status()
        data = response.json()

        horas = data["hourly"]["time"]
        humedad = data["hourly"]["relativehumidity_2m"]
        viento = data["hourly"]["windspeed_10m"]

        return horas, humedad, viento
    except Exception as e:
        messagebox.showerror("Error", f"No se pudieron obtener los datos:\n{e}")
        return [], [], []


def create_line_chart(horas, valores, variable):
    """Gráfica de línea personalizada."""
    fig, ax = plt.subplots(figsize=(6, 3))
    ax.plot(horas, valores, linestyle="-", marker="s", markersize=4,
            linewidth=1.8, alpha=0.8, color="tab:blue" if variable=="Humedad" else "tab:green")
    ax.set_title(f"{variable} en Buenos Aires (línea)")
    ax.set_xlabel("Hora")
    ax.set_ylabel("%" if variable=="Humedad" else "km/h")
    ax.tick_params(axis="x", rotation=45)
    ax.grid(True, linestyle="--", alpha=.5)
    fig.tight_layout()
    return fig


def create_bar_chart(horas, valores, variable):
    """Gráfica de barras personalizada."""
    fig, ax = plt.subplots(figsize=(6, 3))
    ax.bar(horas, valores, alpha=0.7, color="tab:orange" if variable=="Humedad" else "tab:red")
    ax.set_title(f"{variable} en Buenos Aires (barras)")
    ax.set_xlabel("Hora")
    ax.set_ylabel("%" if variable=="Humedad" else "km/h")
    ax.tick_params(axis="x", rotation=45)
```

```
    ax.grid(True, linestyle="--", alpha=.5)
    fig.tight_layout()
    return fig


def mostrar_graficas(frm, horas, humedad, viento):
    """Inserta las gráficas de humedad y viento en el frame."""
    # Línea y Barras de Humedad
    fig1 = create_line_chart(horas, humedad, "Humedad")
    canvas1 = FigureCanvasTkAgg(fig1, master=frm)
    canvas1.draw()
    canvas1.get_tk_widget().pack(pady=10, fill="x")

    fig2 = create_bar_chart(horas, humedad, "Humedad")
    canvas2 = FigureCanvasTkAgg(fig2, master=frm)
    canvas2.draw()
    canvas2.get_tk_widget().pack(pady=10, fill="x")

    # Línea y Barras de Viento
    fig3 = create_line_chart(horas, viento, "Viento")
    canvas3 = FigureCanvasTkAgg(fig3, master=frm)
    canvas3.draw()
    canvas3.get_tk_widget().pack(pady=10, fill="x")

    fig4 = create_bar_chart(horas, viento, "Viento")
    canvas4 = FigureCanvasTkAgg(fig4, master=frm)
    canvas4.draw()
    canvas4.get_tk_widget().pack(pady=10, fill="x")


def open_win_canvas(parent: tk.Tk):
    """
    Crea la ventana secundaria con gráficas de la API.
    """
    win = tk.Toplevel(parent)
    win.title("Canvas con API (Open-Meteo) - Buenos Aires")
    win.geometry("960x1200")

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

    # Botón para cargar datos y graficar
    def cargar():
        horas, humedad, viento = fetch_data()
        if horas and humedad and viento:
            mostrar_graficas(frm, horas, humedad, viento)

    ttk.Button(frm, text="Cargar y mostrar gráficas", command=cargar).pack(pady=10)


# Para pruebas independientes (opcional)
if __name__ == "__main__":
    root = tk.Tk()
    root.title("Prueba win_canvas Buenos Aires")
    ttk.Button(root, text="Abrir ventana Canvas", command=lambda: open_win_canvas(root)).pack(pady=20)
    root.mainloop()
```

Explicación:

Se hicieron cambios al programa para que en lugar de mostrar datos de León, ahora muestre la información de Buenos Aires, Argentina. También se cambiaron las variables que se muestran: en vez de la temperatura, ahora aparecen la humedad y la velocidad del viento de las últimas horas. Además, se mejoró la forma en que se ven las gráficas: las líneas tienen un marcador cuadrado, un grosor mayor y un poco de transparencia, y se agregó una rejilla para que sea más fácil leer los valores.