- Código

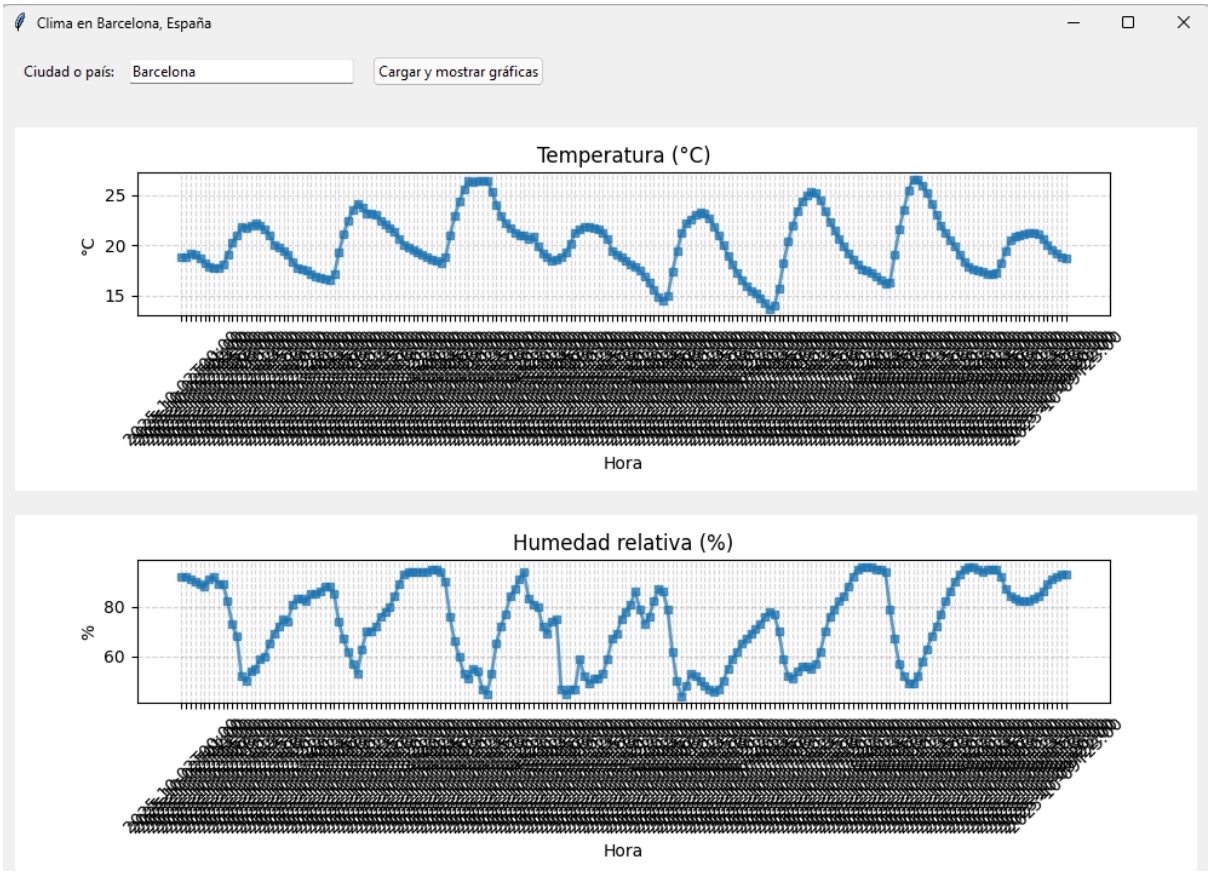
```
import tkinter as tk
from tkinter import ttk, messagebox
import requests
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg


def geocode_city(city_name):
    """Convierte el nombre de ciudad o país en lat/lon usando
Open-Meteo."""
    try:
        url =
f"https://geocoding-api.open-meteo.com/v1/search?name={city_name}&count
=1&language=es&format=json"
        response = requests.get(url, timeout=10)
        response.raise_for_status()
        data = response.json()

        if "results" not in data or not data["results"]:
            raise ValueError("No se encontró la ciudad/país")

        result = data["results"][0]
        return result["latitude"], result["longitude"], result["name"],
result.get("country", "")
    except Exception as e:
        messagebox.showerror("Error", f"No se pudo geocodificar la
ciudad:\n{e}")
        return None, None, None, None


def fetch_data(lat, lon):
    """Obtiene temperatura, humedad relativa y velocidad del viento
(km/h) desde Open-Meteo."""
    try:
        url = (
            "https://api.open-meteo.com/v1/forecast"
            f"?latitude={lat}&longitude={lon}"
            "&hourly=temperature_2m,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"]
        temperaturas = data["hourly"]["temperature_2m"]
        humedades = data["hourly"]["relativehumidity_2m"]
        vientos = data["hourly"]["windspeed_10m"]  # en km/h

        # Debug en consola
        print("Datos obtenidos:")
        print("Temperaturas:", len(temperaturas))
        print("Humedades:", len(humedades))
        print("Vientos:", len(vientos))

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


def create_line_chart(horas, valores, titulo, ylabel):
    """Crea una gráfica de línea con estilo mejorado."""
    fig, ax = plt.subplots(figsize=(6, 3))

    if valores:  # Si hay datos
        ax.plot(horas, valores, linestyle="-", marker="s",
markersize=4,
                linewidth=2, alpha=0.7)
    else:  # Si no hay datos
        ax.text(0.5, 0.5, "Sin datos disponibles", ha="center",
va="center", fontsize=12, color="red")

    ax.set_title(titulo)
    ax.set_xlabel("Hora")
    ax.set_ylabel(ylabel)
    ax.tick_params(axis="x", rotation=45)
    ax.grid(True, linestyle="--", alpha=0.5)
    fig.tight_layout()
    return fig
```

```python
def mostrar_graficas(frm, horas, temps, hums, vientos):
    """Inserta SIEMPRE las tres gráficas en el frame de tkinter."""
    for widget in frm.winfo_children():
        widget.destroy()  # limpiar gráficas anteriores

    # Temperatura
    fig1 = create_line_chart(horas, temps, "Temperatura (°C)", "°C")
    canvas1 = FigureCanvasTkAgg(fig1, master=frm)
    canvas1.draw()
    canvas1.get_tk_widget().pack(pady=10, fill="x")

    # Humedad relativa
    fig2 = create_line_chart(horas, hums, "Humedad relativa (%)", "%")
    canvas2 = FigureCanvasTkAgg(fig2, master=frm)
    canvas2.draw()
    canvas2.get_tk_widget().pack(pady=10, fill="x")

    # Viento
    fig3 = create_line_chart(horas, vientos, "Velocidad del viento
(km/h)", "km/h")
    canvas3 = FigureCanvasTkAgg(fig3, master=frm)
    canvas3.draw()
    canvas3.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("1000x1300")

    top_frame = ttk.Frame(win, padding=12)
    top_frame.pack(fill="x")

    ttk.Label(top_frame, text="Ciudad o país:").pack(side="left",
padx=5)

    city_var = tk.StringVar(value="León")
    city_entry = ttk.Entry(top_frame, textvariable=city_var, width=30)
    city_entry.pack(side="left", padx=5)

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

```
    def cargar():
        ciudad = city_var.get().strip()
        if not ciudad:
            messagebox.showwarning("Atención", "Escribe una ciudad o
país")
            return

        lat, lon, nombre, pais = geocode_city(ciudad)
        if lat is None:
            return

        horas, temps, hums, vientos = fetch_data(lat, lon)
        if horas:
            win.title(f"Clima en {nombre}, {pais}")
            mostrar_graficas(frm, horas, temps, hums, vientos)

    ttk.Button(top_frame, text="Cargar y mostrar gráficas",
command=cargar).pack(side="left", padx=10)


# Para pruebas independientes
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()
```

- Gráficas



- Cambios del programa

En el código original, los cambios eran limitados: solo se pedían datos de temperatura y humedad para León (`past_days=1`), no se podía elegir otra ciudad o país, y la gráfica de velocidad del viento no siempre se generaba. Además, las gráficas no tenían marcadores personalizados, líneas gruesas, transparencia ni rejilla.

En el código mejorado, se añadieron estos cambios: se permite ingresar cualquier ciudad o país usando la API de geocodificación para obtener latitud y longitud automáticamente; se amplió el rango de días (`past_days=2` y `forecast_days=1`) para garantizar datos de viento; se aseguraron las tres gráficas (temperatura, humedad y viento) incluso si algún dato faltara; y se mejoró la visualización con marcadores cuadrados, líneas más gruesas, transparencia y rejilla.