



Cambios realizados

- Latitud y longitud
- Se agregó la función para crear la gráfica de dispersión
- Se agregó la rejilla en todas las graficas
- Se creó la funcion para mostrar la grafica de dispersion
- Se modifiko el tamaño de la ventana

```
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 la ubicación especificada (últimas 24 horas).
    Devuelve dos listas: horas y temperaturas.
    """
    try:
        url = (
            "https://api.open-meteo.com/v1/forecast"
            "?latitude=36.16028230248094&longitude=138.06289161790033"
            "&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, c="green", linestyle="--", marker="o", markersize=3)
    ax.set_title("Temperatura en Tokio, Japón (línea)")
    ax.set_xlabel("Hora")
    ax.set_ylabel("°C")
    ax.tick_params(axis="x", rotation=45)
    ax.grid(True, linestyle="--", alpha=0.5)
    fig.tight_layout()
    return fig
```

```
def create_bar_chart(horas, temps):
    """Gráfica de barras."""
    fig, ax = plt.subplots(figsize=(6, 3))
```

```
def create_bar_chart(horas, temps):  
    fig, ax = plt.subplots(figsize=(6, 3))  
    ax.bar(horas, temps, alpha=0.7)  
    ax.set_title("Temperatura en Tokio, Japón (barras)")  
    ax.set_xlabel("Hora")  
    ax.set_ylabel("°C")  
    ax.tick_params(axis="x", rotation=45)  
    ax.grid(True, linestyle="--", alpha=0.5)  
    fig.tight_layout()  
    return fig
```

```
def create_scatter_chart(horas, temps):  
    """Gráfica de dispersión."""  
    fig, ax = plt.subplots(figsize=(6, 3))  
    ax.scatter(horas, temps, c="red", s=20, alpha=0.7)  
    ax.set_title("Temperatura en Tokio, Japón (dispersión)")  
    ax.set_xlabel("Hora")  
    ax.set_ylabel("°C")  
    ax.tick_params(axis="x", rotation=45)  
    ax.grid(True, linestyle="--", alpha=0.5)  
    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")

    # Dispersión
    fig3 = create_scatter_chart(horas, temps)
    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("960x1400")

    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()

```