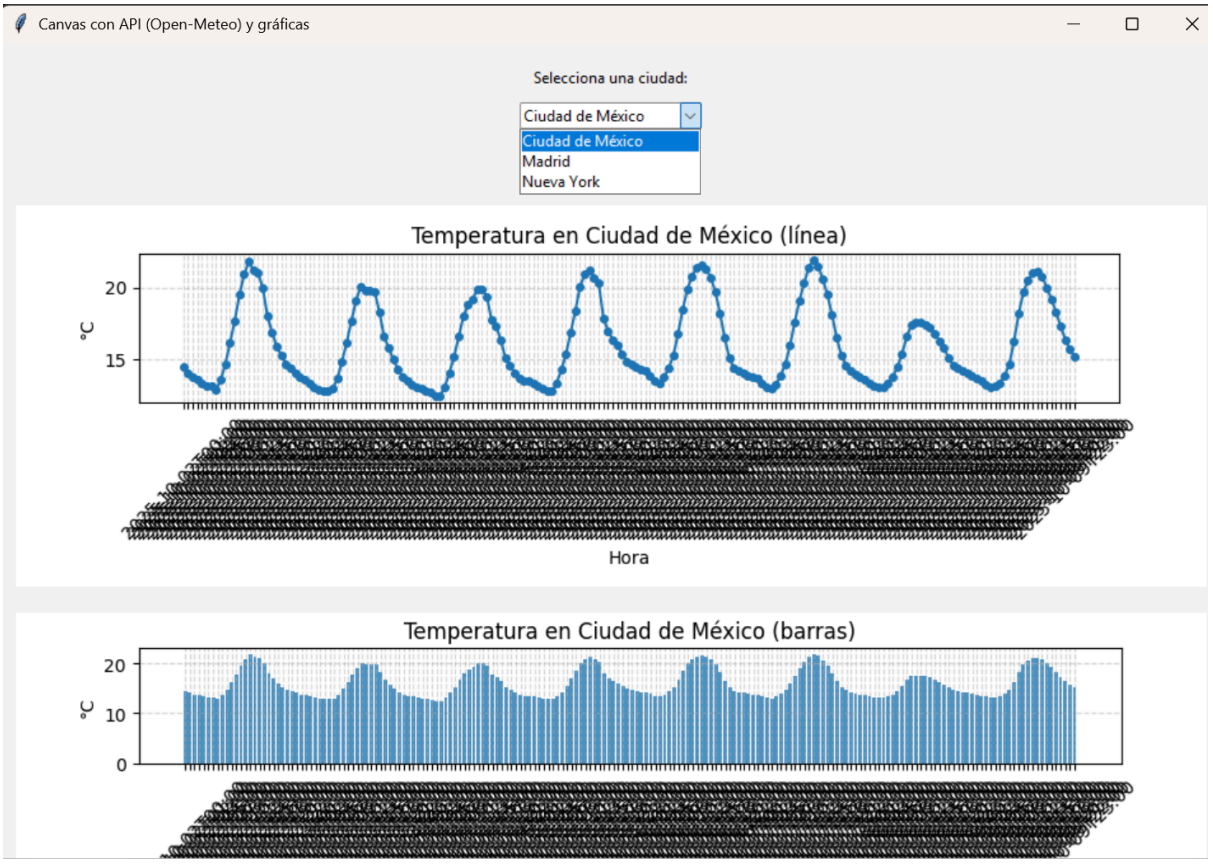
```
import tkinter as tk
from tkinter import ttk, messagebox
import requests
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

# Coordenadas predefinidas
CIUDADES = {
    "Ciudad de México": (19.43, -99.13),
    "Madrid": (40.42, -3.70),
    "Nueva York": (40.71, -74.01)
}


def fetch_data(lat, lon):
    """
    Conecta con la API de Open-Meteo y obtiene temperaturas horarias
    según lat/lon seleccionadas.
    """
    try:
        url = (
            "https://api.open-meteo.com/v1/forecast"
            f"?latitude={lat}&longitude={lon}"
            "&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 [], []


# --------- FUNCIONES DE GRÁFICAS ---------
def create_line_chart(horas, temps, ciudad):
    fig, ax = plt.subplots(figsize=(6, 3))
```

```
    ax.plot(horas, temps, linestyle="-", marker="o", markersize=4,
linewidth=1.5)
    ax.set_title(f"Temperatura en {ciudad} (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, ciudad):
    fig, ax = plt.subplots(figsize=(6, 3))
    ax.bar(horas, temps, alpha=0.8)
    ax.set_title(f"Temperatura en {ciudad} (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 mostrar_graficas(frm, horas, temps, ciudad):
    """Inserta las dos gráficas en el frame de la ventana tkinter."""
    # Línea
    fig1 = create_line_chart(horas, temps, ciudad)
    canvas1 = FigureCanvasTkAgg(fig1, master=frm)
    canvas1.draw()
    canvas1.get_tk_widget().pack(pady=10, fill="x")

    # Barras
    fig2 = create_bar_chart(horas, temps, ciudad)
    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 selección de ciudad y gráficas de la
API.
    """
```

```
    win = tk.Toplevel(parent)
    win.title("Canvas con API (Open-Meteo) y gráficas")
    win.geometry("960x900")

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

    # Selector de ciudad
    ttk.Label(frm, text="Selecciona una ciudad:").pack(pady=5)
    combo_ciudades = ttk.Combobox(frm, values=list(CIUDADES.keys()),
state="readonly")
    combo_ciudades.current(0)  # por defecto CDMX
    combo_ciudades.pack(pady=5)

    # Botón para cargar datos y graficar
    def cargar():
        ciudad = combo_ciudades.get()
        lat, lon = CIUDADES[ciudad]
        horas, temps = fetch_data(lat, lon)
        if horas and temps:
            mostrar_graficas(frm, horas, temps, ciudad)

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



El código original solo mostraba la gráficas con datos fijos de León, mientras que la versión modificada incluye un selector de ciudades las cuales son :CDMX, Madrid y Nueva York , también agrega rejilla para mejorar la lectura, y cambia los títulos de las gráficas de forma dinámica según la ciudad elegida, manteniendo únicamente las dos gráficas pero de manera más clara.