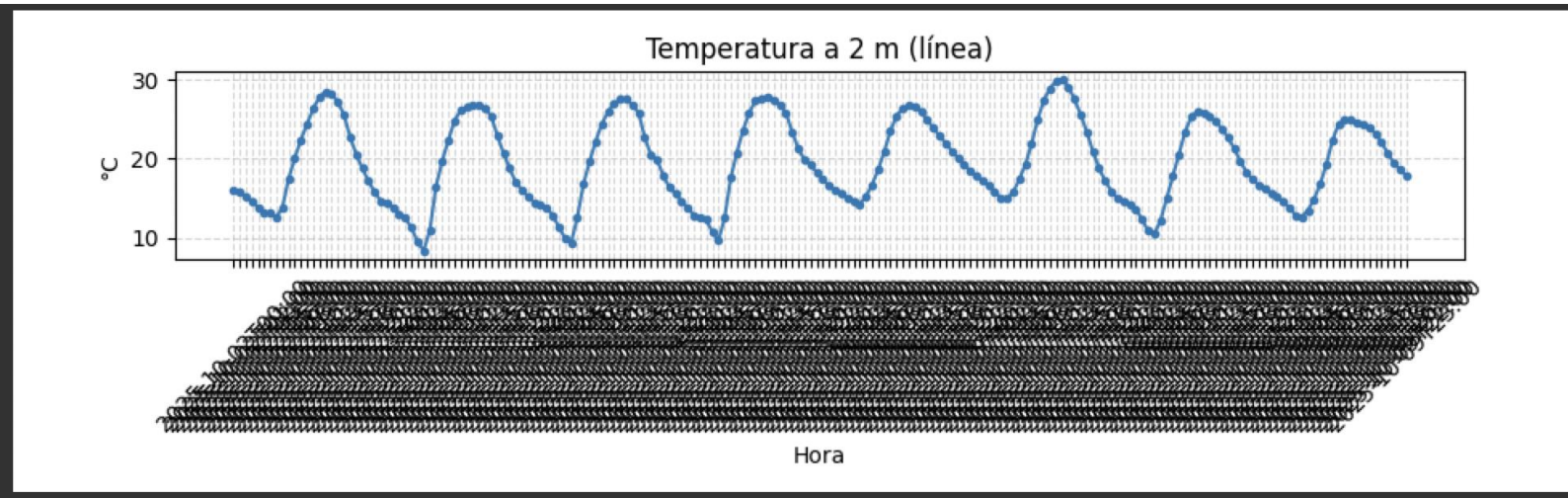

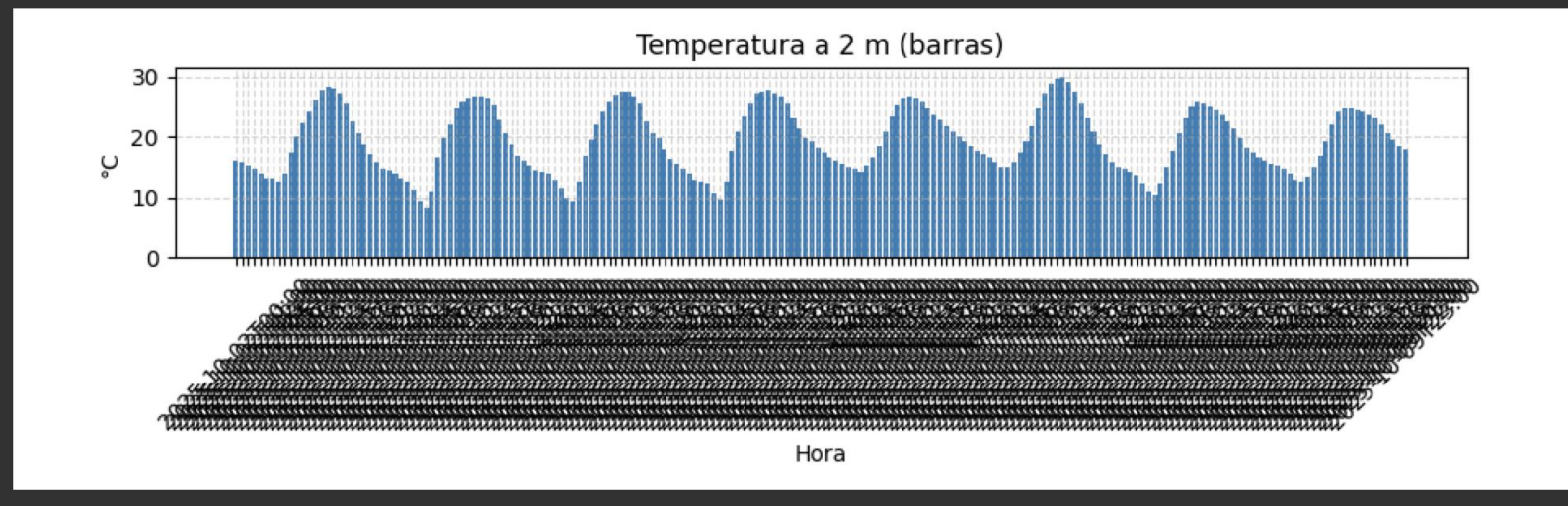

Le añadí controles para elegir la ciudad, escoger qué medir (temperatura, humedad o viento) y ajustar el estilo de la línea, activé la rejilla para leer mejor, puse títulos y unidades según el parámetro. Me apoye de chat gpt

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 fetch_data(latitude: float, longitude: float, hourly_param: str): """ Conecta con la API de Open-Meteo y obtiene el dato horario seleccionado de las últimas 24 h (past_days=1), para las coordenadas dadas. Devuelve dos listas: horas y valores. """ try: url = ("https://api.open-meteo.com/v1/forecast" f"?latitude={latitude}&longitude={longitude}" f"&hourly={hourly_param}&past_days=1" "&timezone=auto") response = requests.get(url, timeout=15) response.raise_for_status() data = response.json()

```
    horas = data["hourly"]["time"]
    valores = data["hourly"][hourly_param]
    return horas, valores
except Exception as e:
    messagebox.showerror("Error", f"No se pudieron obtener los datos:\n{e}")
    return [], []


def y_label_and_title(hourly_param: str): """ Devuelve etiqueta de eje Y y texto del parámetro para el título, según el parámetro elegido. """ mapping_units = { "temperature_2m": "°C", "relativehumidity_2m": "%", "windspeed_10m": "m/s", } mapping_text = { "temperature_2m": "Temperatura a 2 m", "relativehumidity_2m": "Humedad relativa a 2 m", "windspeed_10m": "Velocidad del viento a 10 m", } return mapping_units.get(hourly_param, ""), mapping_text.get(hourly_param, hourly_param)

def create_line_chart(horas, vals, ylabel, title_text, marker_style="o", linewidth=1.5, alpha=1.0): """Gráfica de línea con opciones de marcador, grosor y transparencia + rejilla.""" fig, ax = plt.subplots(figsize=(6, 3)) # Si el usuario elige "Sin marcador" marker = None if marker_style == "Sin marcador" else marker_style ax.plot(horas, vals, linestyle="-", marker=marker, markersize=3, linewidth=linewidth, alpha=alpha) ax.set_title(f"{title_text} (línea)") ax.set_xlabel("Hora") ax.set_ylabel(ylabel) ax.tick_params(axis="x", rotation=45) ax.grid(True, linestyle="--", alpha=.5) fig.tight_layout() return fig

def create_bar_chart(horas, vals, ylabel, title_text, alpha=1.0): """Gráfica de barras con transparencia + rejilla.""" fig, ax = plt.subplots(figsize=(6, 3)) ax.bar(horas, vals, alpha=alpha) ax.set_title(f"{title_text} (barras)") ax.set_xlabel("Hora") ax.set_ylabel(ylabel) ax.tick_params(axis="x", rotation=45) ax.grid(True, linestyle="--", alpha=.5) fig.tight_layout() return fig

def mostrar_graficas(frm, horas, vals, ylabel, title_text, marker_style, linewidth, alpha): """Limpia el frame y coloca las dos gráficas actualizadas.""" # Limpia contenido previo for w in frm.winfo_children(): if isinstance(w, (tk.Canvas, ttk.Frame)): pass for child in list(frm.children.values()): # Solo destruye widgets de gráficos previos (FigureCanvasTkAgg genera tk.Widget) try: child.destroy() except Exception: pass

# Línea
fig1 = create_line_chart(horas, vals, ylabel, title_text, marker_style, linewidth, alpha)
canvas1 = FigureCanvasTkAgg(fig1, master=frm)
canvas1.draw()
canvas1.get_tk_widget().pack(pady=10, fill="x")
```

```
# Barras
fig2 = create_bar_chart(horas, vals, ylabel, title_text, alpha)
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 controles y gráficas de la API. """ win = tk.Toplevel(parent) win.title("Canvas con API (Open-Meteo) y gráficas") win.geometry("1000x1050")

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

# -- Controles superiores --
controls = ttk.LabelFrame(outer, text="Controles")
controls.pack(fill="x", pady=6)

# Presets de ciudades
ttk.Label(controls, text="Ciudad preset:").grid(row=0, column=0, padx=6, pady=6,
sticky="w")
ciudades = {
   "León, MX": (21.12, -101.68),
   "CDMX, MX": (19.4326, -99.1332),
   "Guadalajara, MX": (20.6736, -103.344),
   "Monterrey, MX": (25.6866, -100.3161),
   "Austin, US": (30.2672, -97.7431),
   "Madrid, ES": (40.4168, -3.7038),
   "Personalizado": (None, None),
}
ciudad_var = tk.StringVar(value="León, MX")
cmb_ciudad = ttk.Combobox(controls, textvariable=ciudad_var,
values=list(ciudades.keys()), state="readonly", width=18)
cmb_ciudad.grid(row=0, column=1, padx=6, pady=6, sticky="w")

# Lat/Long editables
ttk.Label(controls, text="Lat:").grid(row=0, column=2, padx=(18,6), pady=6, sticky="e")
lat_var = tk.StringVar(value=str(ciudades["León, MX"][0]))
ent_lat = ttk.Entry(controls, textvariable=lat_var, width=12)
ent_lat.grid(row=0, column=3, padx=6, pady=6, sticky="w")

ttk.Label(controls, text="Lon:").grid(row=0, column=4, padx=(18,6), pady=6, sticky="e")
```

```
lon_var = tk.StringVar(value=str(ciudades["León, MX"][1]))
ent_lon = ttk.Entry(controls, textvariable=lon_var, width=12)
ent_lon.grid(row=0, column=5, padx=6, pady=6, sticky="w")

def on_city_change(event=None):
    sel = ciudad_var.get()
    lat, lon = ciudades[sel]
    if lat is not None and lon is not None:
        lat_var.set(str(lat))
        lon_var.set(str(lon))
        ent_lat.configure(state="disabled")
        ent_lon.configure(state="disabled")
    else:
        # Personalizado
        ent_lat.configure(state="normal")
        ent_lon.configure(state="normal")

cmb_ciudad.bind("<<ComboboxSelected>>", on_city_change)
on_city_change()

# Parámetro horario
ttk.Label(controls, text="Parámetro:").grid(row=1, column=0, padx=6, pady=6,
sticky="w")
param_var = tk.StringVar(value="temperature_2m")
cmb_param = ttk.Combobox(
    controls,
    textvariable=param_var,
    values=["temperature_2m", "relativehumidity_2m", "windspeed_10m"],
    state="readonly",
    width=22
)
cmb_param.grid(row=1, column=1, padx=6, pady=6, sticky="w")

# Marcador
ttk.Label(controls, text="Marcador línea:").grid(row=1, column=2, padx=(18,6), pady=6,
sticky="e")
marker_var = tk.StringVar(value="o")
cmb_marker = ttk.Combobox(
    controls,
    textvariable=marker_var,
    values=["o", "s", "^", "D", "Sin marcador"],
```

```
    state="readonly",
    width=12
)
cmb_marker.grid(row=1, column=3, padx=6, pady=6, sticky="w")

# Linewidth
ttk.Label(controls, text="Linewidth:").grid(row=1, column=4, padx=(18,6), pady=6,
sticky="e")
lw_var = tk.DoubleVar(value=1.5)
spn_lw = ttk.Spinbox(controls, textvariable=lw_var, from_=0.5, to=5.0, increment=0.5,
width=6)
spn_lw.grid(row=1, column=5, padx=6, pady=6, sticky="w")

# Alpha
ttk.Label(controls, text="Alpha:").grid(row=2, column=0, padx=6, pady=6, sticky="w")
alpha_var = tk.DoubleVar(value=0.95)
sld_alpha = ttk.Scale(controls, from_=0.2, to=1.0, orient="horizontal",
variable=alpha_var)
sld_alpha.grid(row=2, column=1, columnspan=2, padx=6, pady=6, sticky="we")

# Botones
def cargar():
    try:
        lat = float(lat_var.get())
        lon = float(lon_var.get())
    except ValueError:
        messagebox.showwarning("Coordenadas inválidas", "Latitud/longitud deben ser
números.")
        return

    hourly_param = param_var.get()
    horas, vals = fetch_data(lat, lon, hourly_param)
    if horas and vals:
        ylabel, text = y_label_and_title(hourly_param)
        mostrar_graficas(plot_area, horas, vals, ylabel, text, marker_var.get(), lw_var.get(),
alpha_var.get())

ttk.Button(controls, text="Cargar y mostrar gráficas", command=cargar).grid(row=2,
column=4, columnspan=2, padx=6, pady=6, sticky="we")

for i in range(6):
```

```
    controls.grid_columnconfigure(i, weight=1)

# -- Área de gráficos --
plot_area = ttk.Frame(outer)
plot_area.pack(fill="both", expand=True, pady=(8,0))

# Hint inicial
ttk.Label(plot_area, text="Elige ciudad/parámetro y pulsa «Cargar y mostrar
gráficas».").pack(pady=16)
```