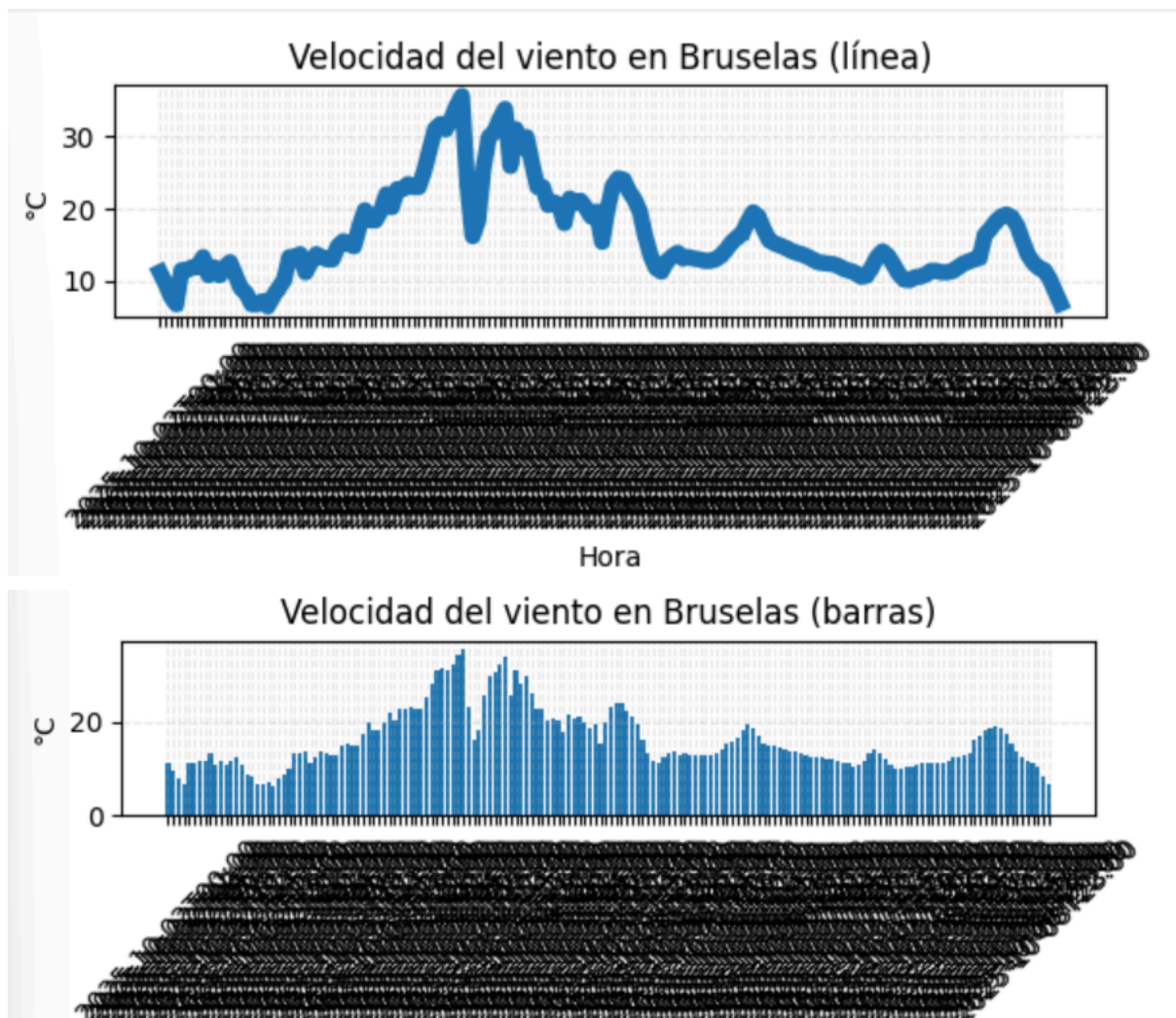


Sesión 7

Cambios

los cambios que hubieron fueron el tomar la ciudad de Bruselas, el aumentar el ancho de la primera gráfica a 6, poner marcadores s, agregar una rejilla con una transparencia de 0.25 a las dos gráficas, tomar la velocidad del tiempo en lugar de la temperatura, cambiar el nombre de que tenían las dos gráficas para que digan correctamente que estoy midiendo la velocidad del aire

Imagen de las graficas



Codigos:

```
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=50.84837444914778&longitude=4.3534409366962885"
            "&hourly=windspeed_10m"
            "&timezone=auto"
        )
        response = requests.get(url, timeout=15)
        response.raise_for_status()
        data = response.json()

        horas = data["hourly"]["time"]
        temperaturas = data["hourly"]["windspeed_10m"]

        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, linewidth= 6, linestyle="--", marker="s",
markersize=3)
    ax.set_title("Velocidad del viento en Bruselas (línea)")
    ax.set_xlabel("Hora")
    ax.set_ylabel("°C")
    ax.tick_params(axis="x", rotation=45)
    ax.grid(True, linestyle="--", alpha=.25)
    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("Velocidad del viento en Bruselas (barras)")
    ax.set_xlabel("Hora")
    ax.set_ylabel("°C")
    ax.tick_params(axis="x", rotation=45)
    ax.grid(True, linestyle="--", alpha=.25)
    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()
```