Développement d'une application de bureau pour la Météo

[3]:
```
import tkinter as tk
from tkinter import *
import customtkinter as ctk
from customtkinter import *
import os
from PIL import Image, ImageTk
import requests
import json
import matplotlib.pyplot as plt
from datetime import datetime
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

# Fonction pour obtenir les données météo
def obtenir_meteo(ville):
    api = f"http://api.openweathermap.org/data/2.5/forecast?
 ↪q={ville}&appid=27e6d531b18fbd4ce46509bf2987f027"
    try:
        # Envoyer la requête à l'API
        response = requests.get(api)
        # Vérifier si la réponse est OK (code 200)
        if response.status_code == 404:
            print("Erreur 404 : Ville non trouvée.")
            return None
        # Vérifier si l'API a renvoyé des données valides
        json_data = response.json()
        if json_data.get("cod") != "200":
            print(f"Erreur API: {json_data.get('message', 'Erreur inconnue')}")
            return None
        # Extraction des températures horaires
        temeperature_horraires = []
        for forecast in json_data['list']:
            temperature = int(forecast['main']['temp'] - 273.15)
            temeperature_horraires.append(temperature)
        # Informations actuelles
        meteo_actuel = json_data['list'][0]
        condition = meteo_actuel['weather'][0]['main']
        temperature = int(meteo_actuel['main']['temp'] - 273.15)
        humidite = meteo_actuel['main']['humidity']
        vits_vent=int(meteo_actuel['wind']['speed'] * 3.6)
        return temperature, humidite, condition, vits_vent,␣
 ↪temeperature_horraires
    except Exception as e:
        print(f"Erreur : {e}")
        return None
```

```
# Sauvegarder les données dans un fichier JSON
def sauv_donnees_meteo(donnees, file="donnees_meteo.json"):
    with open(file, 'w') as f:
        json.dump(donnees, f)
# Charger les données depuis un fichier JSON
def charge_donnees_meteo(file="donnees_meteo.json"):
    try:
        with open(file, 'r') as f:
            donnees = json.load(f)
        return donnees
    except FileNotFoundError:
        print("Le fichier n'a pas été trouvé.")
        return None
# Fonction pour générer un graphique de l'historique des températures
def generer_graph_temp(temperatures):
    plt.clf()
    plt.plot(temperatures, color='blue',label='Température (°C)
    plt.title('Historique des températures
    plt.ylabel('Température (°C)
    plt.xlabel('heurs')
    plt.legend
    plt.tight_layout()  # Ajuste la mise en page
    return plt.gcf()  # Retourne la figure courante
# Fonction pour afficher les données dans l'interface
def affiche_info_meteo(ville, temperature, humidite, description, vits_vent,␣
↪temeperature_horraires):
    temp_label.configure(text=f"Température actuelle : {temperature}°C")
    humidite_label.configure(text=f"Humidité : {humidite}%")
    vent_label.configure(text=f"Vitesse de vent :{vits_vent} km/h ")
    description_label.configure(text=f"Conditions : {description}")
    # Générer le graphique de l'historique des températures
    fig = generer_graph_temp(temeperature_horraires)
    canvas = FigureCanvasTkAgg(fig, master=frame_graph)
    canvas.draw()
    canvas.get_tk_widget().pack()
# Fonction appelée lorsque le bouton "Obtenir Météo" est cliqué
def on_obtenir_meteo():
    ville = ville_Entry.get()
    donnees = obtenir_meteo(ville)
    if donnees:
        temperature, humidite, description, vits_vent, temeperature_horraires =␣
↪donnees
        sauv_donnees_meteo(donnees)
        affiche_info_meteo(ville, temperature, humidite, description, vits_vent,␣
↪temeperature_horraires)
# Fonction appelée pour charger les données depuis un fichier JSON
def on_charge_donnees():
```

```
    donnees = charge_donnees_meteo()
    if donnees:
        temperature, humidite, description, vits_vent, temeperature_horraires =␣
↪donnees
        affiche_info_meteo("Ville chargée depuis JSON", temperature, humidite,␣
↪description, vits_vent, temeperature_horraires)
# Fonction intermédiaire pour sauvegarder les données
def on_sauv_donnees():
    donnees = charge_donnees_meteo()  # Charger les données pour vérifier leur␣
↪existence
    if donnees:
        sauv_donnees_meteo(donnees)
        print("Données sauvegardées avec succès.")
    else:
        print("Aucune donnée à sauvegarder.")
# Configuration de l'apparence
ctk.set_appearance_mode("light")
ctk.set_default_color_theme("blue")
# Créer la fenêtre principale avec Tkinter
Application = ctk.CTk()
# Configurer la taille de la fenêtre
Application.title("Météo Maroc")
Application.iconbitmap("C:\\Users\\khadi\\cloudy.ico")
Application.geometry("800x600")
# Définir l'image de fond
bg = tk.PhotoImage(file="arriere_plan.png")
ma_label = tk.Label(Application, image=bg)
ma_label.place(x=-2, y=-2)
# Charger les images pour la température, l'humidité, et les conditions
temp_img = Image.open("celsius.png")
humidite_img = Image.open("humidity (1).png")
vent_image= Image.open("windyy.png")
condition_img = Image.open("sun-cloud.png")
# Redimensionner les images
temp_img = temp_img.resize((30, 30))
humidite_img = humidite_img.resize((30, 30))
vent_image = vent_image.resize((30, 30))
condition_img = condition_img.resize((30, 30))
# Convertir les images en CTkImage
temp_icon = ctk.CTkImage(temp_img)
humidite_icon = ctk.CTkImage(humidite_img)
vent_icon=ctk.CTkImage(vent_image)
condition_icon = ctk.CTkImage(condition_img)
# Frame pour les champs de saisie
ville_label = ctk.CTkLabel(Application, text="Ville:", font=("Arial", 17,␣
↪"bold"))
ville_label.pack(pady=10)
```

```
ville_Entry = ctk.CTkEntry(Application)
ville_Entry.pack(pady=5)
# Bouton pour obtenir les données mété
bouton_obten_meteo= ctk.CTkButton(Application, text="Obtenir la Méteo",␣
 ↪command=on_obtenir_meteo)
bouton_obten_meteo.pack(pady=10)
# Labels pour afficher les informations météo avec les images
temp_label = ctk.CTkLabel(Application, text="Température actuelle : N/D",␣
 ↪image=temp_icon, compound="left",padx=10)
temp_label.pack(pady=5)
humidite_label = ctk.CTkLabel(Application, text="Humidité : N/D",␣
 ↪image=humidite_icon, compound="left",padx=10)
humidite_label.pack(pady=5)
vent_label = ctk.CTkLabel(Application, text="Vitesse de vent : N/D",␣
 ↪image=vent_icon, compound="left",padx=10)
vent_label.pack(pady=5)
description_label = ctk.CTkLabel(Application, text="Conditions : N/D",␣
 ↪image=condition_icon, compound="left",padx=10)
description_label.pack(pady=5)
# Bouton pour charger les données depuis le fichier JSON
load_data_button = ctk.CTkButton(Application, text="Charger les données depuis␣
 ↪JSON", command=on_charge_donnees)
load_data_button.pack(pady=5)
# Bouton pour sauvegarder les données
save_data_button = ctk.CTkButton(Application, text="Sauvegarder les données",␣
 ↪command=on_sauv_donnees)
save_data_button.pack(pady=5)
# Frame pour afficher le graphique
frame_graph = ctk.CTkFrame(Application)
frame_graph.pack(fill="both", expand=False)
# Lancer l'application Tkinter
Application.mainloop()
```


Réalisée par :
LAMHOUR Khadija