Partie VI

Interface utilisateur

La bibliothèque flask

Flask est un micro-framework web en Python, simple et léger. Il permet de créer rapidement un petit site web ou une API sans dépendances complexes.

Dans notre cas, Flask permet de :

- transformer votre Raspberry Pi en serveur web local ;
- créer une interface web pour afficher les données météo en temps réel ;
- séparer le traitement (dans capteur.py) de l'affichage (dans app.py).

Installation de flask

```
pip3 install flask
```

Importation de flask

Importons flask au début de notre code :

```
from flask import Flask, render_template_string
```

- Flask : la classe qui permet de **créer l'application web**.
- render_template_string : permet de **générer une page HTML directement dans le script** (sans créer de fichier .html pour le moment).

Importation des données de l'application métier

Dans notre fichier app.py, nous avons besoin d'accéder aux données produites par nos capteurs (température, humidité, etc.) et aux fonctions de calcul (point de rosée, humidex...).

Ces fonctions ne sont **pas écrites directement dans app.py**, mais dans un autre fichier appelé capteur.py. Ce fichier est ce qu'on appelle notre **application métier** : il contient toute la **logique de calcul et de lecture**.

```
from capteur import (
    lire_donnees_capteur,
    calculer_point_de_rosee,
    calculer_humidex,
    recuperer_date_heure
)
```

- from capteur : on indique qu'on veut importer **depuis le fichier capteur.py**.
- import (...) : on précise **quelles fonctions on veut utiliser** dans ce fichier.

C'est un peu comme si on disait :

« Va chercher dans la boîte capteur.py ces outils bien précis, et rends-les disponibles ici. »

Création de l'application Flask

```
# Création de l'application Flask
app = Flask(__name__)
```

Cette ligne est essentielle. Elle signifie :

- On crée une instance de notre application Flask.
- name est une variable spéciale en Python qui contient le nom du fichier. Flask s'en sert pour retrouver le chemin vers les fichiers associés (comme les templates HTML ou les fichiers CSS).

Autrement dit :

« Je démarre une application web avec ce fichier comme point d'entrée. »

Définir une route dans Flask

```
@app.route('/')
def index():
```

@app.route('/') : Indique que la fonction juste en dessous va être exécutée quand un utilisateur accède à l'adresse « / », c'est-à-dire la racine du site, la page d'accueil.

def index(): C'est une fonction Python classique, appelée ici index. Elle contiendra le code qui définit ce que l'on affiche ou retourne quand on visite cette page.

Récupération des données

```
humidity, temperature = lire_donnees_capteur()
if humidity is not None and temperature is not None:
    point_de_rosee = calculer_point_de_rosee(temperature, humidity)
    humidex = calculer_humidex(temperature, point_de_rosee)
    date_heure = recuperer_date_heure()
```

nous mettons en œuvre **la logique métier définie dans notre fichier capteur.py** :

- **lire_donnees_capteur()** : interroge le capteur DHT22 et renvoie les valeurs d'humidité et de température.
- **calculer_point_de_rosee()** : calcule le point à partir duquel l'humidité de l'air commence à se condenser.
- **calculer_humidex()** : estime la température ressentie selon l'humidité ambiante.
- **recuperer_date_heure()** : renvoie l'heure exacte de la mesure.

Génération simple d'une page HTML avec une f-string

```
html = f"""
<h1>Données météo locales</h1>
<ul>
  <li>Date et heure : {date_heure}</li>
  <li>Température : {temperature} °C</li>
  <li>Humidité : {humidity} %</li>
  <li>Point de rosée : {point_de_rosee} °C</li>
  <li>Humidex : {humidex}</li>
  </ul>
"""
```

Nous créons **une chaîne de caractères contenant du HTML**, dans laquelle nous insérons directement les **valeurs des variables Python** (température, humidité, etc.).

- La **balise h1** sert à afficher titre de niveau 1.
- La **balise ul** sert à afficher une liste à puce, les **balises li** correspondent aux éléments de la liste.
- La **balise p** sert à afficher un paragraphe.

Pourquoi le f devant la chaîne ?

Le f signifie que c'est une **f-string** (ou formatted string literal en anglais).

Cela permet d'écrire **des variables à l'intérieur de la chaîne** entre des accolades {}.

C’est une **méthode claire et moderne pour mélanger texte et variables** dans une même ligne.

Et si la lecture échoue ?

```
else:
    html = "<p>Erreur de lecture du capteur.</p>"
```

Ce code gère le cas où le capteur ne renvoie pas de valeurs valides. Dans ce cas, on prépare un message simple à afficher sur la page web.

Affichage de la page web et démarrage de l'application

```
return render_template_string(html)
```

Ce code indique à Flask d'**afficher le HTML généré comme contenu de la page web**.

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Ce code permet de lancer l'application :

- **host='0.0.0.0'** : rend l'application accessible sur le réseau.
- **port=5000** : port utilisé pour acceder au site.

Code

```
#importations
from flask import Flask, render_template_string
from capteur import (
    lire_donnees_capteur,
    calculer_point_de_rosee,
    calculer_humidex,
    recuperer_date_heure
)

# Définition de l'application Flask
app = Flask(__name__)

@app.route('/')
def index():
    humidity, temperature = lire_donnees_capteur()
```

```
    if humidity is not None and temperature is not None:
        point_de_rosee = calculer_point_de_rosee(temperature, humidity)
        humidex = calculer_humidex(temperature, point_de_rosee)
        date_heure = recuperer_date_heure()

        html = f"""
        <h1>Données météo locales</h1>
        <ul>
            <li>Date et heure : {date_heure}</li>
            <li>Température : {temperature} °C</li>
            <li>Humidité : {humidity} %</li>
            <li>Point de rosée : {point_de_rosee} °C</li>
            <li>Humidex : {humidex}</li>
        </ul>
        """

    else:
        html = "<p>Erreur de lecture du capteur.</p>"

    return render_template_string(html)
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

capture-station.png

Révision #9
Créé 2025-08-11 07:17:49 UTC
Mis à jour 2025-08-11 08:40:16 UTC