Partie V

Logique métier

Nous sommes prêt à structurer **plus proprement notre application** en séparant :

- la **partie capteurs** (lecture, calculs...) -> **Logique métier**
- et l'**application web** (routes, affichage)

Création d'un fichier capteur.py

Dans une démarche de développement propre, modulaire et réutilisable, on **sépare la logique métier** (la lecture des données et les calculs) de **l'interface utilisateur** (site web, affichage).

Le fichier capteur.py **ne contient plus de boucle** while True ni de traitement direct. Il est composé **exclusivement de fonctions**, que l'on pourra appeler **depuis une autre application**, ici app.py (notre serveur Flask).

Ce script agit comme **une boîte à outils**, il regroupe :

- la lecture du capteur DHT22,
- le calcul du point de rosée,
- le calcul de l'indice humidex,
- la récupération de la date et heure actuelles.

Code

```
#importations
import adafruit_dht
import board
import math
from datetime import datetime



def lire_donnees_capteur():
```

```
    try:
        dhtDevice = adafruit_dht.DHT22(board.D4)
        humidity = dhtDevice.humidity
        temperature = dhtDevice.temperature
        dhtDevice.exit()
        if humidity is not None and temperature is not None:
            return round(humidity, 1), round(temperature, 1)
        else:
            return None, None

    except RuntimeError as error:
        print("Erreur de lecture :", error)
        return None, None

    except Exception as error:
        dhtDevice.exit()
        raise error

def calculer_point_de_rosee(temperature, humidity):
    #Formule pour calculer le point de rosée
    alpha = 17.27
    beta = 237.7
    gamma = (alpha * temperature) / (beta + temperature) + math.log(humidity / 100)
    point_de_rosee = (beta * gamma) / (alpha - gamma)
    return round(point_de_rosee, 1)

def calculer_humidex(temperature, point_de_rosee):
    #Formule pour calculer l'indice humidex
    humidex = temperature + (5/9) * (6.11 * math.exp(5417.7530 * ((1/273.16) - (1/273.15 +
point_de_rosee))) - 10)
    return round(humidex, 1)

def recuperer_date_heure():
    return datetime.now().strftime("%d-%m-%Y %H:%M:%S")
```

Commentaires sur la fonction lire_donnees_capteur() :

- Cette fonction permet de lire la température et l’humidité à partir du capteur **DHT22** connecté au **GPIO4** du Raspberry Pi.
- Elle renvoie ces deux valeurs sous forme de nombres arrondis à une décimale.
- Si la lecture échoue, elle retourne None, None.

```
try:
```

- On démarre un bloc qui va tenter de réaliser la lecture
- Si une erreur survient, Python basculera dans le bloc except.

```
dhtDevice = adafruit_dht.DHT22(board.D4)
```

- Cette ligne crée un objet capteur, ici un DHT22, branché sur le GPIO4 (représenté par board.D4).
- □□ □□ C’est une étape indispensable pour communiquer avec le capteur. L'objet est créé directement dans la fonction pour éviter tous risques de blocage du capteur.

```
humidity = dhtDevice.humidity temperature = dhtDevice.temperature
```

Ces lignes interrogent le capteur pour récupérer :

- humidity → le taux d’humidité.
- temperature → la température.

```
dhtDevice.exit()
```

- Cette commande est **très importante** : elle **libère les ressources GPIO** utilisées par le capteur.
- Cela évite les erreurs fréquentes sur Raspberry Pi telles que : “Lost access to message queue”.

```
if humidity is not None and temperature is not None:
```

- On vérifie que le capteur a bien répondu avec des données valides.
- Les capteurs DHT peuvent parfois échouer à donner une valeur.

```
return round(humidity, 1), round(temperature, 1)
```

Si les valeurs sont valides :

- On les arrondit à une décimale pour un affichage plus lisible.
- Puis on les renvoie sous la forme de deux nombres.

```
else: return None, None
```

Si le capteur n’a pas répondu correctement, la fonction renvoie None pour les deux mesures.

```
except RuntimeError as error:
```

- Gestion des erreurs courantes dues aux capteurs DHT (perte temporaire de lecture).
- On affiche l’erreur mais le programme continue de fonctionner.

```
except Exception as error:
```

- Gestion des erreurs plus graves (problème matériel, GPIO bloqué, etc.).
- On ferme proprement le capteur avec dhtDevice.exit() avant de relancer l’erreur (raise error) pour éventuellement arrêter le programme.

Révision #10
Créé 2025-08-11 06:54:56 UTC
Mis à jour 2025-08-11 07:19:16 UTC