Framework de partage de données en utilisant IoT, Edge Computing et IA

Réalisé par :
Kaoutar RHFOUDA (AI)
Killian BOISSEAU (AI)
Gauthier PERRAULT (IOT)
Sana DEIRI (IOT)
Piotr KRUPINSKI (AI)

Encadré par :
NDIAYE Fatou Diop

Présentation et contexte

L'objectif du projet est de mettre en place un capteur qui mesure la température et l'humidité ambiante. Les données seront envoyées pour être traités grâce à l'Edge Computing et d'autres méthodes de traitement. Ensuite, les données seront envoyées afin qu'ils soient stockés dans une base de données dans le cloud. L'objectif ensuite est de restituer ces données dans une interface utilisateur afin de les visualiser graphiquement.

Le but est de détecter la production d'un incendie en se basant sur les données de température et d'humidité relevées. Si la température est supérieure à un certain seuil et l'humidité est inférieure à un certain seuil, ceci doit indiquer le déclanchement d'un incendie.

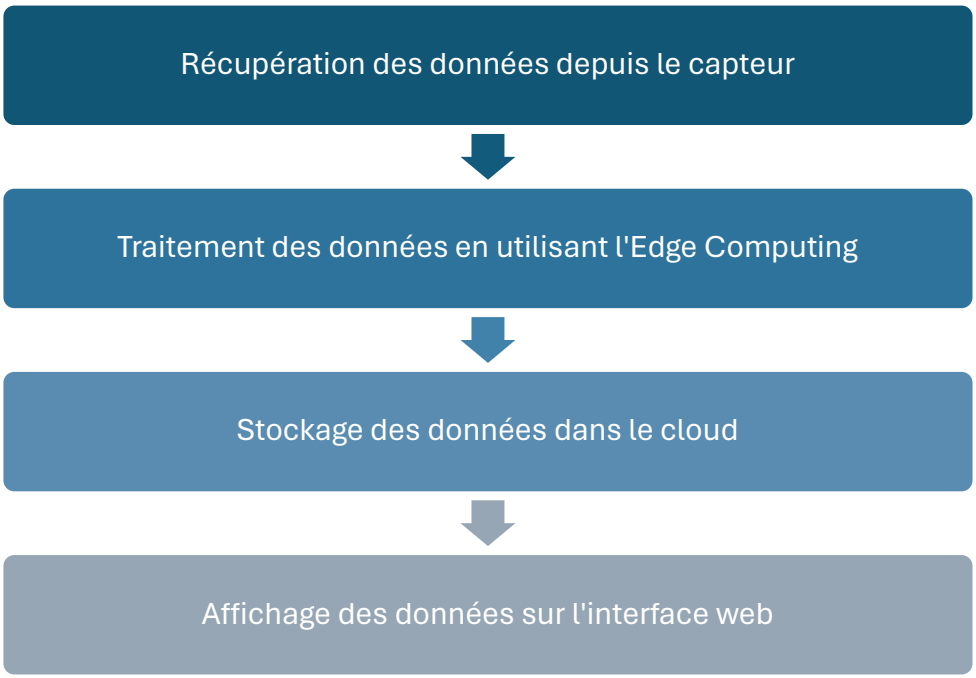


Schéma présentant l'architecture de la Framework

1. Couche Capteurs et Appareils IoT

La première étape consiste à configurer le matériel : branchement du capteur de température et d'humidité à la Raspberry.

Suivant le type de capteur, les pins auxquels les câbles sont branchés peuvent différer.

Voici les pins à respecter pour un capteur du type DHT22 (le capteur que nous avons utilisé):

Module DHT11	Raspberry Pi Pico
1	GPIO17
2	3V3
3	GND

Tableau présentant les pins à utiliser pour un capteur DHT11

Ensuite, en regardant le schéma qui montre le nom de chaque pin, on peut faire le branchement.

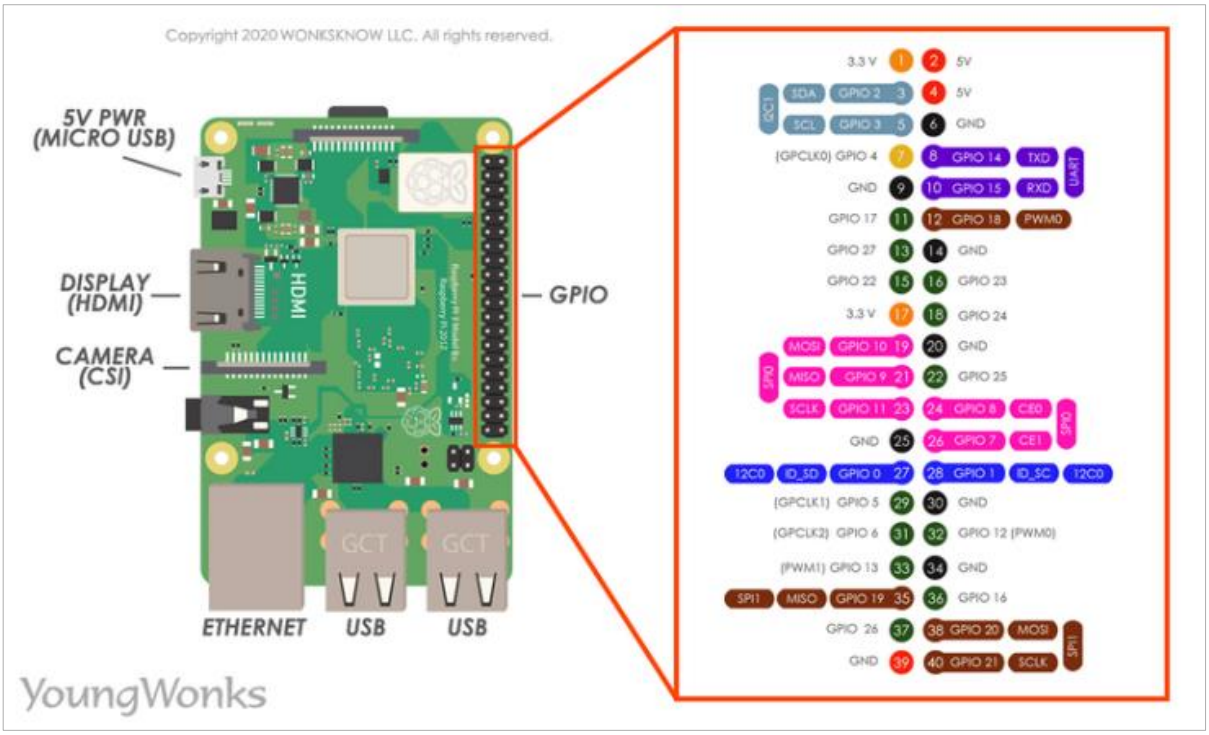
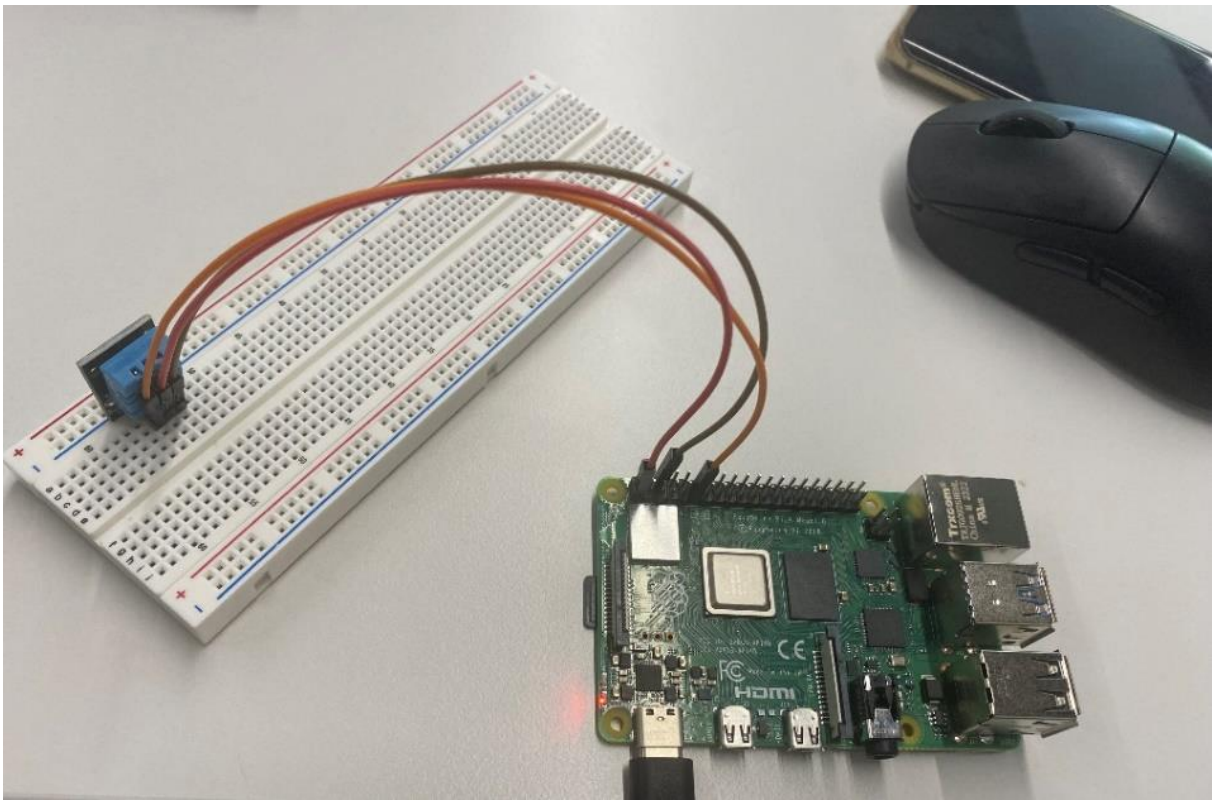


Schéma montrant les pins d'un Raspberry Pi 4 (upesy.fr)

Voici le branchement réalisé :



Ensuite, il faut renvoyer les données récupérées par le capteur vers une API en utilisant une méthode POST.

La route de l'api qu'on va utiliser est 192.168.213.35:5000/sensor-data

Les données sont sous forme json :

```
{
    'id': 1,
    'temperature': temp,
    'humidity' : humidity
}
```

Voici un exemple de donnée renvoyée. A chaque 10 lignes de données récupérées, on les envoie à l'API et le code de succès s'affiche :

```
(venv) GSKKP@GSKKP:~/fire_projet $ python3 fire.py
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
```

```
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Status Code: 200
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Temp=24.0°C, Humidity=43.0%
Status Code: 200
```

2. Couche Passerelle IoT

Le but de cette couche était de traiter les données récupérées par le capteur et de les filtrer avant de les envoyer à la couche suivante. Dans notre cas, les données recueillies ne sont pas complexes et n'ont pas forcément besoin de traitement dans cette couche, car ce sont des données de température et d'humidité faciles à lire. Nous avons donc décidé de faire le traitement des données, notamment le contrôle de la température et de l'humidité, dans la couche Edge Computing.

3. Couche Edge Computing

3.1. Traitement des données

Le traitement des données a été fait suivant 2 aspects distincts.

Contrôle sur la température et l'humidité

L'objectif de ce traitement est de vérifier la température et l'humidité mesurées à chaque donnée reçue, pour décider s'il y a un incendie ou non.

Une température au-dessus de 50° C combinée avec une humidité en dessous de 30% signifie la présence d'un incendie.

Préparation des données à envoyer au cloud

Le deuxième traitement consistait à décider quelles données envoyer à la couche suivante (la couche Cloud).

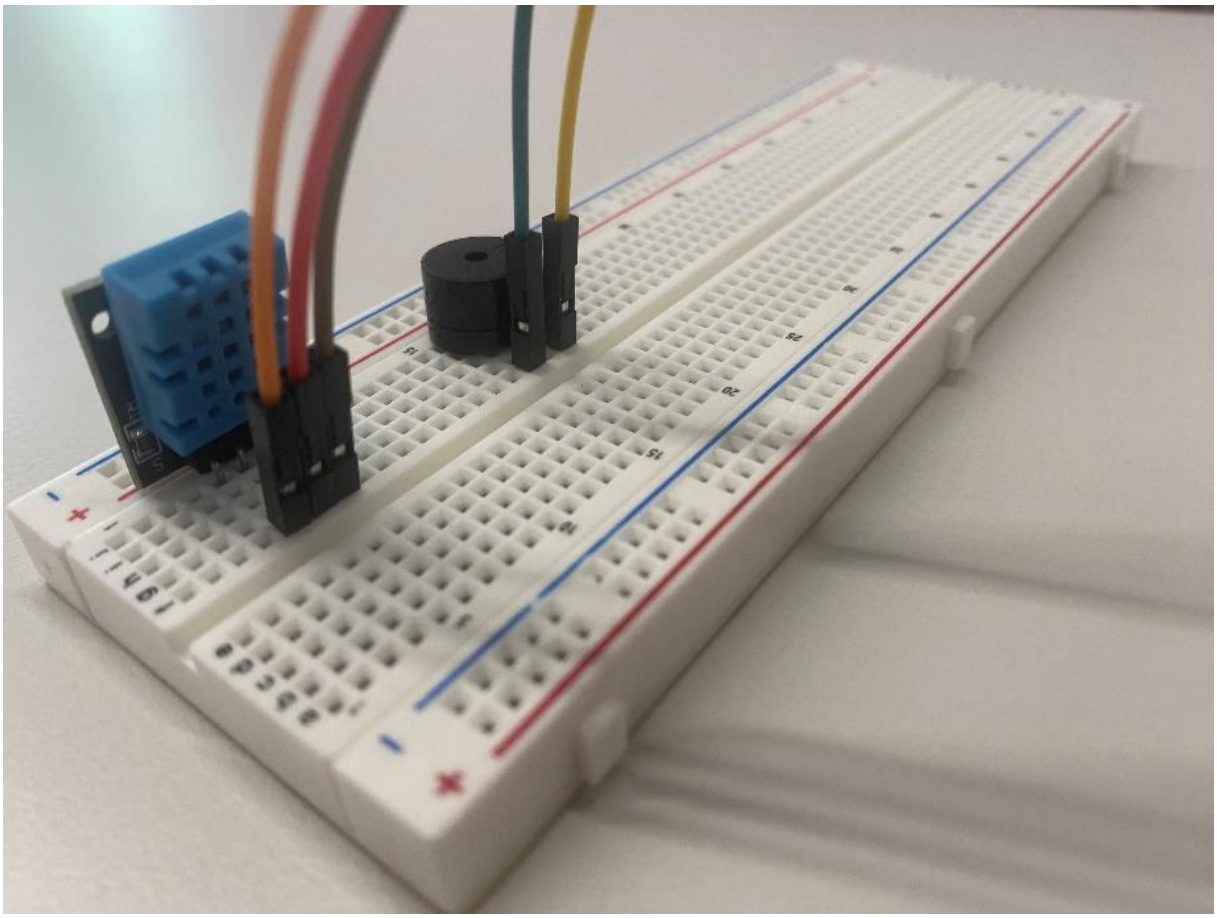
Actuellement, la couche Edge reçoit des lots de 10 données. Pour éviter d'envoyer trop de données, nous avons décidé d'envoyer, pour chaque lot de 10 mesures, la moyenne de ces 10 mesures.

De plus, nous avons ajouté un horodatage à chaque valeur que nous renvoyons afin de pouvoir avoir une cohérence dans la visualisation des données dans le temps.

3.2. Ajout de la fonctionnalité d'alarme sonore

Nous avons ajouté une fonctionnalité qui permet de déclencher une alarme sonore (un buzzer) en cas de détection d'incendie.

Le branchement a été réalisé en respectant les spécifications des pins d'un buzzer actif : le pin (–) est relié au GND (ground) tandis que le pin (+) est relié à une des broches GPIO.



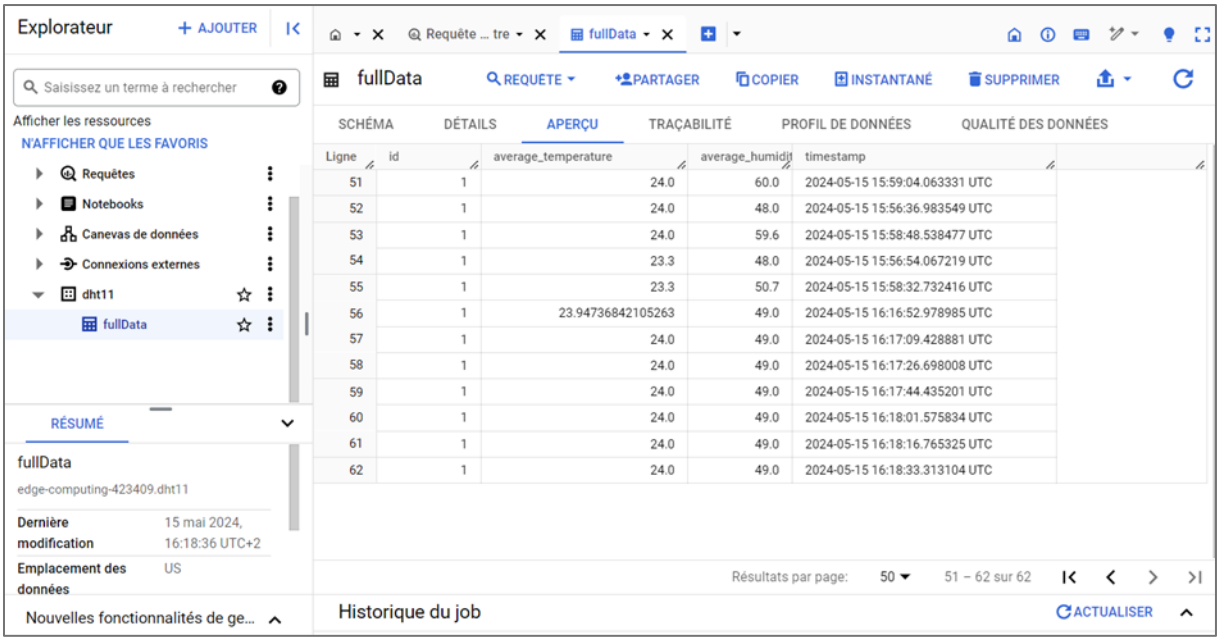
Ainsi, si le programme estime un incendie, le buzzer sonne.

4. Couche Cloud

Le but de la couche Cloud est de récupérer les données envoyées par la couche Edge Computing, de les stocker et de les organiser dans une base de données claire et bien structurée.

Cette base de données dans le cloud permet de conserver les informations de manière sécurisée et accessible. Ces données seront ensuite utilisées par l'interface utilisateur pour permettre aux utilisateurs de visualiser et d'analyser les informations facilement.

Voici un extrait de la base de données créée avec des données capturées par le capteur :

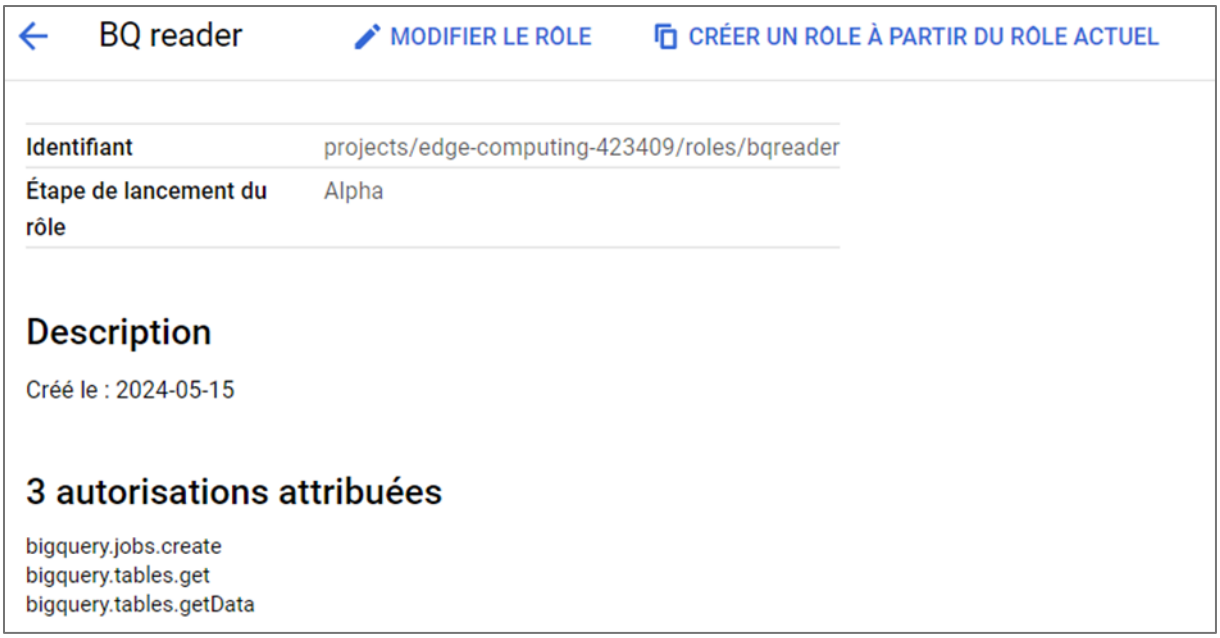


Le programme qui envoie les données vers le cloud possède un fichier JSON utilisé comme identifiants pour se connecter à Google Cloud Platform et télécharger les données sur BigQuery.

Nous avons mis en place une API qui récupère les données de la table BigQuery et cette API est appelée par notre site web pour afficher les données. Encore une fois, pour récupérer les données, notre API a besoin d'identifiants pour lire la table BigQuery.

Ces deux identifiants ont été créés pour n'utiliser que les permissions nécessaires afin qu'ils ne puissent pas accéder à d'autres ressources de Google Cloud Platform, réduisant ainsi le risque de compromettre l'ensemble du projet si la clé est volée par exemple.

Voici les permissions accordées pour chaque fichier JSON:



Les identifiants ’bqreader’ sont utilisés pour seulement lire les données de la table BigQuery.

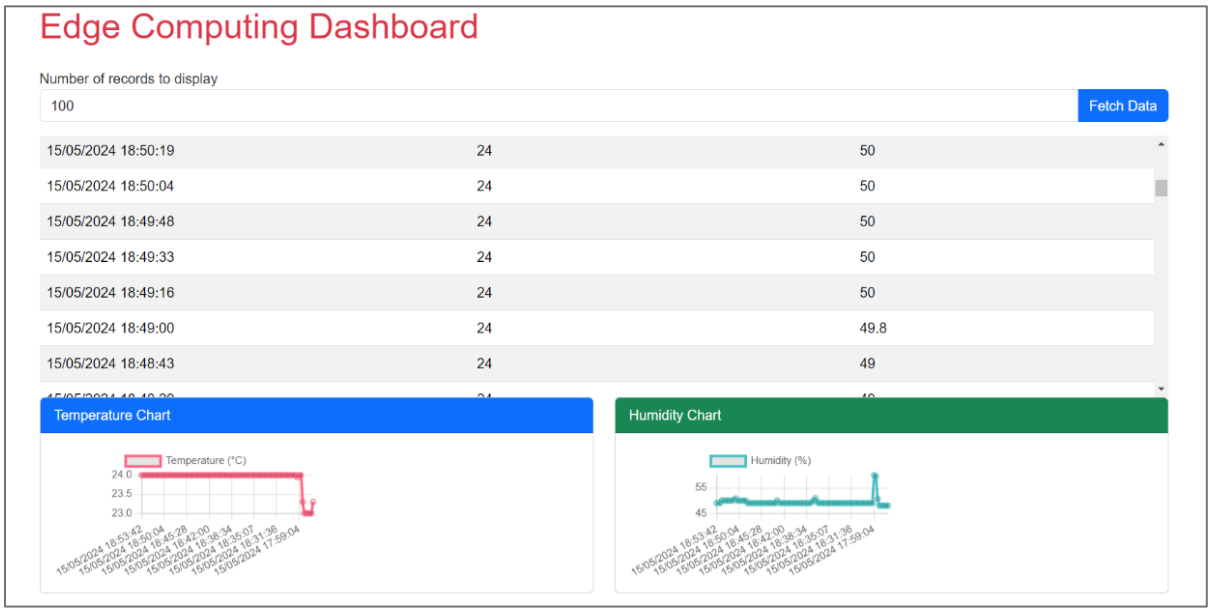
Les identifiants ’BQuploader’ sont utilisés pour ajouter les données de la table BigQuery.

5. Couche Analyse de données IA

La Couche d'Analyse de données était destinée à utiliser l'IA pour analyser les données collectées et en extraire des informations précieuses pour l'entreprise. Cependant, en raison de la limitation de la quantité de données disponibles, le traitement IA était quelque peu restreint. Étant donné que les données se limitaient principalement à des mesures de température et d'humidité, il était difficile d'appliquer des techniques d'apprentissage automatique telles que l'apprentissage supervisé, non supervisées ou par renforcement pour identifier des modèles significatifs ou effectuer des prédictions pertinentes.

6. Couche Interface utilisateur

La Couche Interface utilisateur reçoit les données du cloud et les présente à travers des graphiques, des tableaux, etc. Les utilisateurs peuvent interagir avec ces données en choisissant le nombre de lignes à afficher à l'écran et en explorant les informations à travers une interface conviviale.



Capture montrant les données reçues et affichées dans l'interface