
Rapport Travail 1 : Intégration des APIs Ecowatt et Météo France avec OAuth2

◆ Introduction

L'objectif de ce travail était d'implémenter une solution permettant d'accéder aux données des services Ecowatt (RTE) et des Données d'observations de Météo France tout en sécurisant les échanges via le protocole **OAuth2**. Ce rapport détaille l'implémentation technique réalisée.

◆ 1. Authentification OAuth2

📌 Objectif

L'API Ecowatt nécessite une authentification OAuth2 pour accéder aux données. Nous avons mis en place une fonction permettant de récupérer un **jeton d'accès (access token)** à partir de l'endpoint d'authentification fourni par RTE.

🖥️ Implémentation

Python

```
def get_oauth_token():
    url = "https://digital.iservices.rte-france.com/token/oauth"
    headers = {
        'Authorization': 'Basic
OTRlMDkyZjctMjYyYS00NTIwLWFmYTctNDcwNGJlYjAwNjEyOjVmNjYyMTY1LWQ2MDctNGI3Ny1h
NjYzLTc0Y2U0NzRlMDc1ZA=',
        'Content-Type': 'application/x-www-form-urlencoded',
    }

    response = requests.post(url, headers=headers)
    if response.status_code == 200:
        token_data = response.json()
        print("✅ Token récupéré :", token_data['access_token'])
```

```
    return token_data['access_token']
else:
    print(f"❌ Erreur {response.status_code}: {response.text}")
    return None
```

Explication

- Une requête **POST** est envoyée avec les **identifiants OAuth2 encodés en Base64**.
- Si la réponse est **200 OK**, le **token d'accès** est extrait.
- En cas d'erreur, un message explicatif est affiché.

◆ 2. Récupération des données Ecowatt

Objectif

Une fois authentifié, nous devons récupérer les **signaux Ecowatt** indiquant les pics de consommation d'électricité.

Implémentation

Python

```
def fetch_ecowatt_data(token, file_path):
    url = "https://digital.iservices.rte-france.com/open_api/ecowatt/v5/signals"
    headers = {"Authorization": f"Bearer {token}", "Accept": "application/json"}

    max_retries = 5
    retry_delay = 10 # secondes

    for attempt in range(max_retries):
        response = requests.get(url, headers=headers)
        if response.status_code == 200:
            try:
                data = response.json()
                with open(file_path, "w", encoding="utf-8") as f:
                    json.dump(data, f, indent=4)
                print(f"✅ Données Ecowatt sauvegardées dans {file_path}")
```

```

        return data
    except json.JSONDecodeError:
        print("❌ Erreur: Réponse API non valide")
        return None

    elif response.status_code == 429:
        print(f"⚠️ Trop de requêtes, attente {retry_delay} secondes...")
        time.sleep(retry_delay)
    else:
        print(f"❌ Erreur {response.status_code}: {response.text}")
        return None

print("❌ Échec après plusieurs tentatives")
return None

```

Explication

- Une requête **GET** est envoyée avec le **token d'accès**.
- Les données JSON sont **sauvegardées localement** dans `ecowatt.json`.
- Un **système de gestion des erreurs** avec 5 tentatives est intégré pour éviter les **erreurs 429 (Trop de requêtes)**.

◆ 3. Analyse des données Ecowatt

Objectif

Les signaux récupérés doivent être traités afin d'identifier les périodes de **tension électrique**.

Implémentation avec une liste chaînée

Python

```

class Node:
    def __init__(self, jour, pas):
        self.jour = jour
        self.pas = pas
        self.next = None

def add_to_linked_list(head, jour, pas):

```

```

new_node = Node(jour, pas)
if not head:
    return new_node
last = head
while last.next:
    last = last.next
last.next = new_node
return head

def analyze_ecowatt_data(file_path):
    if not os.path.exists(file_path):
        print(f"❌ Fichier {file_path} introuvable")
        return
    with open(file_path, "r", encoding="utf-8") as f:
        data = json.load(f)
    head = None
    for signal in data.get('signals', []):
        jour = signal['jour']
        for entry in signal.get('values', []):
            if entry.get('hvalue') == 0:
                head = add_to_linked_list(head, jour, entry['pas'])

    current = head
    while current:
        print(f"📅 Jour: {current.jour}, ⌚ Heure: {current.pas}h")
        current = current.next

```

Explication

- Une **liste chaînée** est utilisée pour stocker les périodes critiques.
- Les créneaux horaires sont affichés dans la console.

◆ Conclusion

Ce travail nous a permis de mettre en place une **authentification sécurisée** avec OAuth2, d'effectuer des requêtes vers l'API **Ecowatt de RTE**, et d'analyser les signaux récupérés.


✅ Résultats obtenus :

- Récupération du **token OAuth2**.

- Téléchargement des **signaux.Ecowatt** en JSON.
- Analyse et extraction des périodes critiques.

Améliorations possibles :

- Intégrer l'API de **Météo France** pour croiser les données.
- Stocker les résultats dans une **base de données** pour une analyse plus approfondie.

 **Prochaine étape :** Intégration du délestage énergétique avec Phidget !