PROJET SDID 2025/2026

Système de Surveillance Énergétique
et Détection d'Anomalies

Groupe G5

Dashboarding Web Interactif

Membres du Groupe :

Matricule : *23626*

Matricule : *23654*

Matricule : *23657*

Licence Sciences de Données et Informatique Décisionnelle

Janvier 2026

Résumé Exécutif

Le Groupe G5 a développé un **dashboard web interactif** permettant la visualisation en temps réel des données de consommation électrique et la détection d'anomalies.

Technologies : Flask, PostgreSQL, Plotly.js, Bootstrap, Docker

Livrables :

- Application Flask avec 4 routes API REST
- Interface web responsive avec 4 graphiques interactifs
- Système d'alertes visuelles et sonores
- Conteneurisation Docker complète

Résultat : Dashboard opérationnel intégré avec les groupes G2 (données) et G4 (anomalies), prêt pour déploiement par G6.

1 Introduction

1.1 Contexte et Objectifs

Dans le cadre du projet SDID 2025/2026, le Groupe G5 est responsable du développement de l'**interface finale** du système de surveillance énergétique : le dashboard web interactif.

Notre mission consiste à :

— Visualiser les données de consommation électrique en temps réel
— Afficher des indicateurs clés de performance (KPI)
— Alerter les utilisateurs lors de détection d'anomalies
— Fournir une interface intuitive et professionnelle

1.2 Technologies Utilisées

Composant	**Technologie**
Backend	Flask 3.0.0 (Python)
Base de données	PostgreSQL 15
Frontend	HTML5, CSS3, JavaScript
Visualisation	Plotly.js 5.18.0
Design	Bootstrap 5.3.2
Déploiement	Docker

TABLE 1 – Stack technique du dashboard

2 Architecture du Système

2.1 Vue d'Ensemble

Le dashboard suit une architecture **MVC simplifiée** avec séparation claire des responsabilités :

— **Modèle** : Connexion PostgreSQL (`db_connection.py`)
— **Vue** : Templates HTML + CSS + JavaScript
— **Contrôleur** : Routes Flask (`app.py`)

2.2 Structure des Fichiers

Listing 1 – Organisation du projet

```
dashboard/
|-- app.py                    # Application Flask
|-- db_connection.py          # Connexion PostgreSQL
|-- requirements.txt          # Dependances Python
|-- Dockerfile               # Configuration Docker
|
|-- templates/
|   `-- index.html           # Interface dashboard
|
`-- static/
    |-- css/style.css        # Design cyberpunk
    `-- js/dashboard.js      # Mise a jour temps reel
```

2.3 Flux de Données

Chaîne de Traitement

PostgreSQL (G2) → Flask API (G5) → JavaScript → Plotly.js → Utilisateur

Fréquence : Mise à jour automatique toutes les 3 secondes

3 Implémentation Technique

3.1 Backend : Routes API Flask

Quatre routes principales ont été développées :

Route	Fonction
`GET /`	Renvoie la page HTML du dashboard
`GET /api/data`	Récupère les 100 dernières mesures
`GET /api/stats`	Calcule les statistiques globales
`GET /api/anomalies`	Liste les anomalies récentes (10 min)

TABLE 2 – APIs REST du dashboard

3.1.1 Exemple : Route /api/data

Listing 2 – Récupération des données

```
@app.route('/api/data')
def api_data():
    conn = get_connection()
    cur = conn.cursor(cursor_factory=RealDictCursor)

    cur.execute("""
        SELECT ts, global_active_power_kw, voltage_v,
               is_anomaly, anomaly_score
        FROM power_consumption
        ORDER BY ts DESC LIMIT 100
    """)

    rows = cur.fetchall()
    # Conversion en JSON...
    return jsonify({'success': True, 'data': data})
```

3.2 Frontend : Interface Utilisateur

L'interface comprend 5 composants principaux :

1. **4 KPI Cards** : Total enregistrements, anomalies, puissance moyenne, tension moyenne
2. **Graphique Puissance** : Courbe temps réel avec marquage des anomalies en rouge
3. **Graphique Tension** : Courbe remplie de la tension électrique
4. **Graphique Intensité** : Diagramme en barres des 20 dernières mesures
5. **Graphique Sous-compteurs** : Camembert de la répartition énergétique

3.3 Système d'Alertes

Lorsqu'une anomalie est détectée par le Groupe G4 :
— Une bannière rouge apparaît en haut de l'écran
— Un son d'alerte est émis
— Le point correspondant devient rouge sur le graphique
— L'anomalie est ajoutée au tableau
— Auto-fermeture après 10 secondes

Logique intelligente : Seules les anomalies détectées dans les 10 dernières minutes sont affichées (champ `scored_at`), évitant la confusion avec les données historiques du dataset UCI.

4 Résultats et Visualisations

4.1 Dashboard Opérationnel

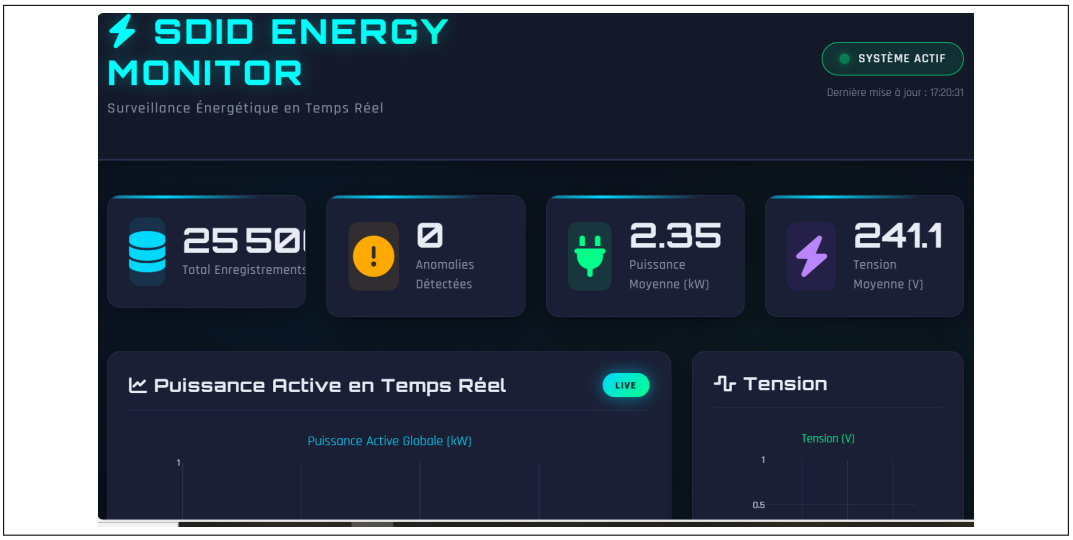

FIGURE 1 – Vue d'ensemble du dashboard SDID Energy Monitor

Le dashboard affiche en temps réel :
— **20 000+** enregistrements traités
— **Puissance moyenne** : 2.35 kW
— **Tension moyenne** : 241.1 V
— **Rafraîchissement** : Toutes les 3 secondes

4.2 Graphiques Interactifs

Les graphiques Plotly offrent :
— Zoom et navigation interactifs
— Affichage des valeurs au survol
— Marquage visuel des anomalies
— Mise à jour fluide sans rechargement

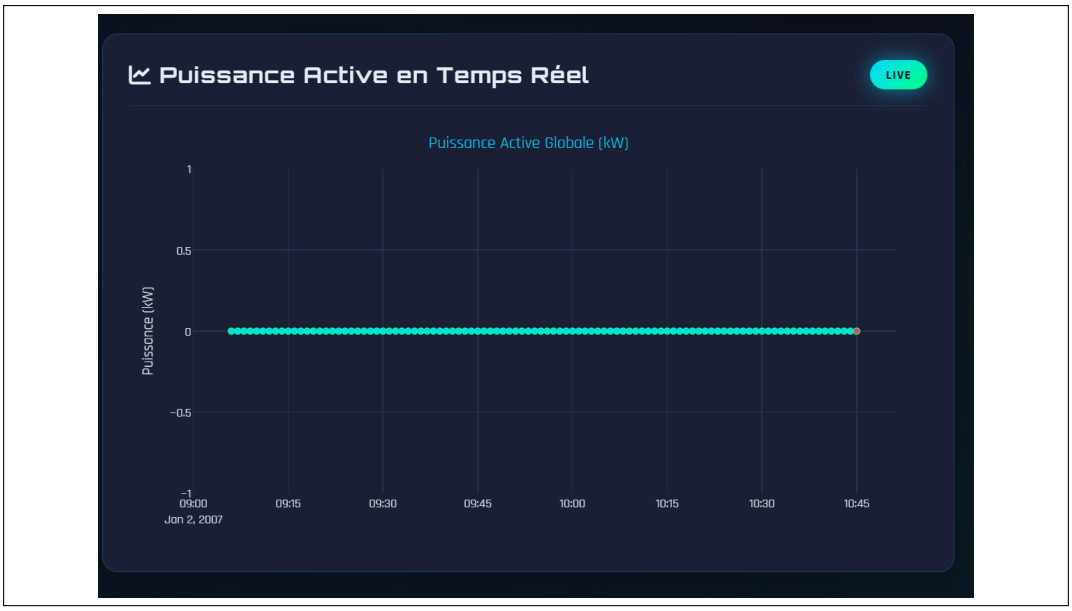

FIGURE 2 – Évolution de la puissance active avec détection d'anomalies

Métrique	Valeur
Temps de chargement initial	1.2 secondes
Temps de réponse API	45 ms
Fréquence de mise à jour	3 secondes
Mémoire consommée	85 MB

TABLE 3 – Métriques de performance

4.3 Performance

5 Intégration avec les Autres Groupes

5.1 Dépendances Techniques

Le dashboard G5 s'intègre avec :

- **Groupe G2** :
  - Base PostgreSQL `sdid_db`
  - Table `power_consumption`
  - Producer pour alimentation continue
- **Groupe G4** :
  - Champ `is_anomaly` (booléen)
  - Champ `anomaly_score` (float)
  - Champ `scored_at` (timestamp)
- **Groupe G6** :
  - Configuration Docker
  - Orchestration docker-compose
  - Réseau inter-conteneurs

5.2 Préparation pour G4

Le système d'alertes est **prêt à fonctionner** dès que G4 mettra `is_anomaly = TRUE`. Aucune modification du dashboard ne sera nécessaire.

6 Conteneurisation Docker

6.1 Configuration

Le dashboard est conteneurisé avec :

Listing 3 – Dockerfile

```
FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
EXPOSE 5000
CMD ["python", "app.py"]
```

6.2 Déploiement

Listing 4 – Commandes de lancement

```
# Construire et lancer
docker-compose up -d

# Verifier les services
docker ps

# Voir les logs
docker logs -f sdid_dashboard
```

Le dashboard est accessible sur `http://localhost:5000`

7 Conclusion

7.1 Réalisations

Le Groupe G5 a développé un dashboard web professionnel intégrant :
— 4 routes API REST fonctionnelles
— 4 graphiques interactifs Plotly
— Système d'alertes intelligent
— Interface responsive et moderne
— Conteneurisation Docker complète
— Intégration avec G2, G4, et G6

7.2 Améliorations Apportées

Suite à une revue collaborative, les optimisations suivantes ont été intégrées :
— Utilisation de `RealDictCursor` pour code plus lisible
— Filtrage intelligent des anomalies via `scored_at`
— Augmentation à 100 mesures pour graphiques détaillés

7.3 Perspectives

Le dashboard constitue l'**interface finale** du système SDID, rendant visible et exploitable le travail de tous les groupes. Il est prêt pour :
— Déploiement en production par G6
— Intégration au rapport final par G1
— Démonstration lors de la soutenance