

📍 Avenue V. Maistriau 8a
B-7000 Mons

☎ +32 (0)65 33 81 54

📧 scitech-mons@heh.be

WWW.HEH.BE

Rapport Technique

- Ce document est le rapport technique de SmartCity – Énergie.

Bachelier en Informatique – ITR2 – GR10

Projet Interdisciplinaire

SMARTCITY (Énergie) – Décembre 2024

Tom Deneyer
Anthony Vergeylen
Quentin Hemeryck
Kelian Noullet

Tables des matières

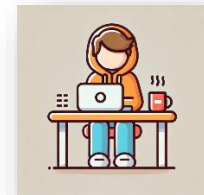
Contributeurs.....	4
Logs.....	4
Infrastructure et configuration réseau	5
Topologie Logique Réseau	5
Table d'adressage vlan	5
Table d'adressage réseau	6
Configuration Switch.....	7
Matériel	7
Backup	7
Configuration apportée.....	7
Windows serveur	8
Global.....	8
Local	8
Linux serveur	9
Global.....	9
Local	10
Plan de sauvegarde	11
Modélisation Bases de données.....	12
Base de données globale	12
Base de données locale	13
Replication Maste-Slave.....	14
Analyses UML.....	14
Use Case:.....	14
Class:.....	15
Séquence:	15
Maquettes application web.....	16
Site Locale.....	16
Wireframe	16
Prototype Fonctionnelle	20
.....	23
Site Globale	25
Codes Python.....	26
Générer des données de capteurs aléatoires	26
1. Connexion à la Base de Données.....	26
2. Récupération des Capteurs Actifs	26

3. Insertion des Données dans la Table capteurs_energie.....	27
4. Insertion des Données dans la Table consommation_energie	27
5. Tests Unitaires	28
Détection des surcharges automatiquement et envoyer une alerte	29
1. Récupération de la Production des 7 Derniers Jours	29
3. Calcul de la Moyenne de Production Anterieure	29
4. Vérification des Alertes Récentes	30
5. Insertion d'une Nouvelle Alerte.....	30
6. Vérification des Anomalies de Production	31
Tests Unitaires.....	31
Codes PHP.....	32
Système d'Authentification via LDAP	32
2. Authentification Initiale	32
3. Recherche de l'Utilisateur dans le Répertoire.....	32
4. Validation de l'Utilisateur	33
5. Gestion des Rôles.....	33
Tests Unitaires.....	34
Gérer les capteurs de SmartCity Energie	34
2. Calcul de la Production des Dernières Heures	34
3. Récupération des Capteurs par Type	35
4. Récupération de Tous les Capteurs	36
5. Comptage des Données d'un Capteur	36
6. Mise à Jour de l'État d'un Capteur	37
Tests Unitaires.....	37
Accueil du site web	38
1. Récupération du Message de Bienvenue	38
2. Récupération de la Production d'Énergie des Dernières 24 Heures.....	38
3. Récupération des Informations Globales	39
4. Récupération de la Consommation des Derniers 30 Jours	40
5. Récupération de la Production des Derniers 30 Jours	40
6. Récupération des Alertes et Incidents	41
7. Tests Unitaires	42
Branches Github	42
Conclusion	42
Test effectués.....	43

Contributeurs

Répartition des Rôles

- **Chef d'équipe : Vergeylen Anthony .**
 - ❖ Référent principal pour le client.
 - ❖ Coordination des tâches et respect des délais.
 - ❖ Participation active au développement ou à la gestion réseau.
- **Développeurs : Vergeylen Anthony, Kelian Noullet .**
 - ❖ Développement des modules fonctionnels.
 - ❖ Résolution des problèmes techniques.
 - ❖ Documentation des travaux réalisés.
- **Gestionnaires réseau : Deneyer Tom, Hemeryck Quentin**
 - ❖ Test des modules pour garantir leur robustesse.
 - ❖ Configuration, sécurisation et maintenance de l'infrastructure réseau.
 - ❖ Gestion, installation et configuration .



Logs

<i>Date et heure</i>	<i>Modificateur</i>	<i>Description modification</i>
16/12/2024 14 :50	Deneyer Tom	Modification de la topologie et adressage.
16/12/2024 15 :00	Hemeryck Quentin	Ajout Maquette Application WEB .
16/12/2024 15 :32	Hemeryck Quentin	Modification Table d'adressage réseau + Vlans.
16/12/2024 17 :29	Deneyer Tom	Mise à jour IP et masques réseaux pour s'adapter aux besoins client (nombre plus important d'adresses).
17/12 /24 12h15	Hemeryck Quentin	Ajout de la MCD et MLD Globale suite à la modification effectuée .
17h15	Hemeryck Quentin	Ajoute des taches effectuées durant la journée + maquettes .
18/12/24 17h29	Deneyer Tom	Ajout des manipulations Windows et serveur Windows backup déporté à la table d'adressage.
11h30	Noullet Kelian	Ajout des différents diagrammes.
18/12/24 8h20	Hemeryck Quentin	Ajout des manipulations Linux serveur + Backup Globale et Locale.
18/12/24 13 :10	Deneyer Tom	Modification script configuration switch
20/12/2024 16 :00	Deneyer Tom	Ajouts codes et descriptions

Infrastructure et configuration réseau

Topologie Logique Réseau

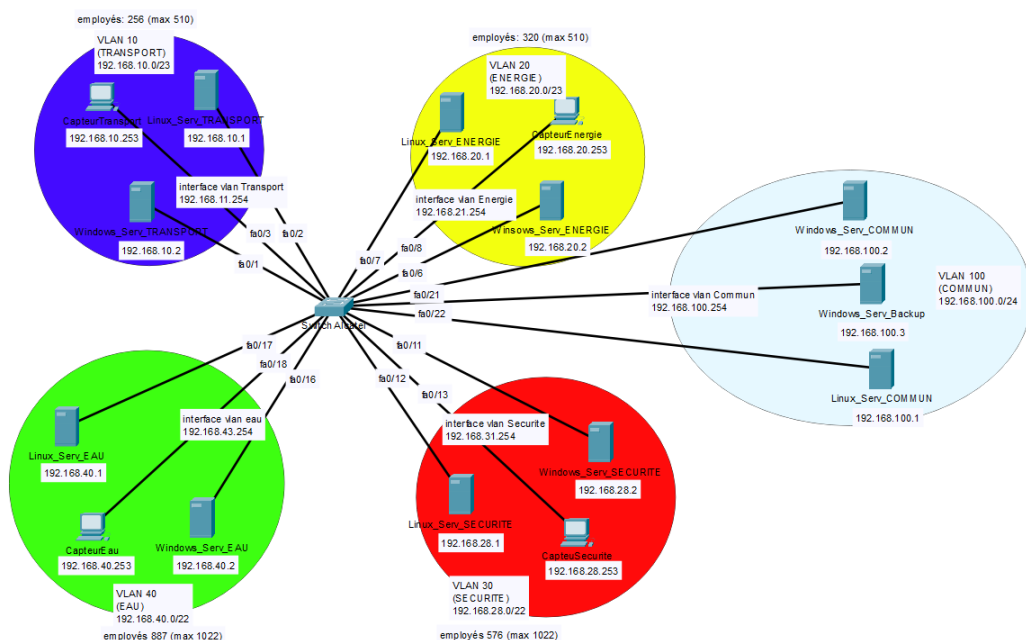


Table d'adressage vlan

Vlan	Adresse	Adresse Interface Vlan	Nom	Groupe	Description
10	192.168.10.0/23	192.168.11.254	Transport	9	Gestion des capteurs de parking, des feux connectés, ...
20	192.168.20.0/23	192.168.21.254	Energie	10	Suivi de la production solaire/éolienne, alertes de surcharge, ...
30	192.168.28.0/22	192.168.31.254	Securite	11	: Gestion des caméras connectées et des capteurs anti-intrusion, ...
40	192.168.40.0/22	192.168.43.254	Eau	12	Détection de fuites et gestion de la qualité de l'eau, ...
100	192.168.100.0/24	192.168.100.254	Commun	9,10,11,12	Administration base de données centrale, DCRoot, DHCP principal...

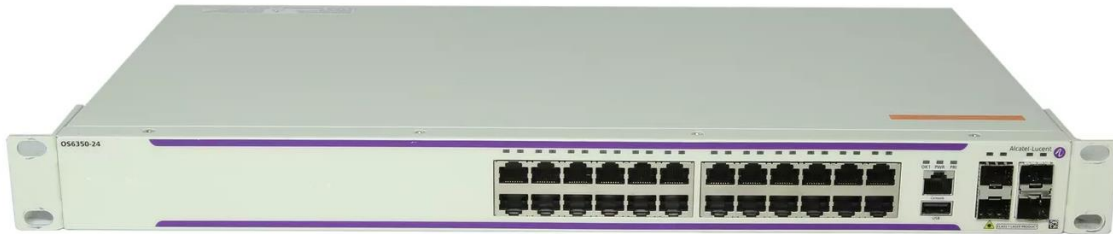
Table d'adressage réseau

VLAN	RéseauVlan	Nom Machine	IP	Port
VLAN10 (TRANSPORT)	192.168.10.0/23	Linux_Serv_TRANSPORT	192.168.10.1	fa0/2
VLAN10 (TRANSPORT)	192.168.10.0/23	Windows_Serv_TRANSPORT	192.168.10.2	fa0/1
VLAN20 (ENERGIE)	192.168.20.0/23	Linux_Serv_ENERGIE	192.168.20.1	fa0/7
VLAN20 (ENERGIE)	192.168.20.0/23	Windows_Serv_ENERGIE	192.168.20.2	fa0/6
VLAN40 (EAU)	192.168.40.0/22	Linux_Serv_EAU	192.168.40.1	fa0/17
VLAN40 (EAU)	192.168.40.0/22	Windows_Serv_EAU	192.168.40.2	fa0/16
VLAN30 (SECURITE)	192.168.30.0/22	Linux_Serv_SECURITE	192.168.28.1	fa0/12
VLAN30 (SECURITE)	192.168.30.0/22	Windows_Serv_SECURITE	192.168.28.2	fa0/11
VLAN100 (COMMUN)	192.168.100.0/24	Linux_Serv_COMMUN	192.168.100.1	fa0/21
VLAN100 (COMMUN)	192.168.100.0/24	Windows_Serv_COMMUN	192.168.100.2	fa0/22
VLAN100 (COMMUN)	192.168.100.0/24	Windows_Serv_Backup	192.168.100.3	Fa0/23
VLAN10 (TRANSPORT)	192.168.10.0/23	CapteurTransport	192.168.10.253	fa0/3
VLAN20 (ENERGIE)	192.168.20.0/23	CapteurEnergie	192.168.20.253	fa0/8
VLAN30 (SECURITE)	192.168.30.0/22	CapteurSecurite	192.168.28.253	fa0/13
VLAN40 (EAU)	192.168.40.0/22	CapteurEeau	192.168.40.253	fa0/18

Configuration Switch

Matériel

Alcatel-Lucent OS6350-24



Backup

- ❖ Récupération du fichier de configuration du switch par connexion FTP via Filezilla
- ❖ Fichier contenant la configuration du switch :
https://github.com/AnthonyCodeDev/SmartCity_CDCGR10/tree/Ressources/ConfigSwitch

Configuration apportée

- ❖ Définition du nom system, contact, localisation et time...
- ❖ Vlan 10 20 30 40 avec noms, attributions des ports et une interface par vlan.
- ❖ Configuration accès ssh
- ❖ Configuration Polices Qos condition, action et règles pour ACL
- ❖ Accès sécurité avec 5 essais maximums pour connexion et 60secondes
- ❖ Port Security sur les ports clients avec 2 adresses mac maximum (du à la connexion client par vm HyperV) avec status admin activé.
- ❖ DHCP Snooping activé, configuration relay DHCP par vlan avec deux serveurs (le global et le local du à la configuration failover)
- ❖ Ports trust configurés pour les périphérique configurés avec une ip statiques. Et en ip-source-filter port pour les clients DHCP.

Site distant : /flash/working					
<div>flash</div> <div>working</div>					
Nom de fichier	Taille de fi...	Type de fic...	Dernière modif...	Droits d'ac...	Propriétaire...
..					
boot.cfg	3 384	Fichier sou...	02/01/2000 07:...		
cloudagent.cfg	139	Fichier sou...	02/12/2024 15:...		
KF3base.img	16 633 755	Fichier d'i...	01/12/2019 15:...		
KF3eni.img	6 650 981	Fichier d'i...	01/12/2019 15:...		
KF3os.img	3 698 674	Fichier d'i...	01/12/2019 15:...		
KF3secu.img	621 836	Fichier d'i...	01/12/2019 15:...		
software.lsm	759	Fichier LSM	01/12/2019 15:...		
7 fichiers. Taille totale : 27 609 528 octets					

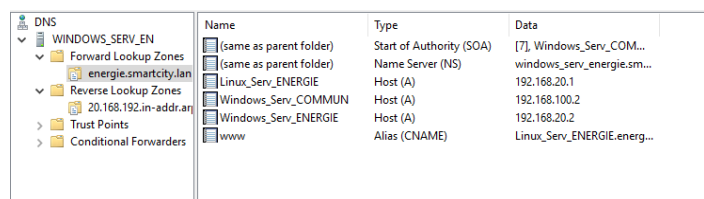
Windows serveur

Global

- ❖ Définition du nom et l'adresse IP du serveur Windows destiné à la configuration globale. (voir adressage)
- ❖ Installer les rôles principaux, notamment :
 - Les services de domaine Active Directory (AD DS).
 - Le rôle DHCP.
 - La fonctionnalité Windows Backup.
- ❖ Élever le serveur au rang de contrôleur de domaine et créer une forêt nommée **smartcity.lan**.
- ❖ Paramétrer le DNS en :
 - Créant une zone de recherche directe.
 - Enregistrant les hôtes avec leurs pointeurs et alias correspondants (une zone inverse pour chaque zone déléguée).
- ❖ Configurer les délégations DNS pour les différents départements.
- ❖ Définir l'étendue des DHCP par vlan avec des adresses exclues pour capteurs, serveurs, et imprimantes potentielles.
- ❖ Configuration des DHCP failover avec load-balancing sur les DHCP des serveurs locaux.
- ❖ Ajouter un nouveau site dans Active Directory et finaliser son paramétrage.

Local

- ❖ Joindre le domaine smartcity.lan et configurer l'adresse IP pour établir la connexion au switch et autres serveurs.

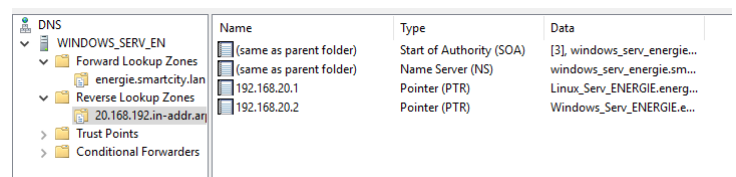


Name	Type	Data
(same as parent folder)	Start of Authority (SOA)	[7] Windows_Serv_COM...
(same as parent folder)	Name Server (NS)	windows_serv_energie.sm...
Linux_Serv_ENERGIE	Host (A)	192.168.20.1
Windows_Serv_COMMUN	Host (A)	192.168.100.2
Windows_Serv_ENERGIE	Host (A)	192.168.20.2
www	Alias (CNAME)	Linux_Serv_ENERGIE.energ...

- ❖ Installer les rôles nécessaires :

- DNS
- DHCP

- ❖ Configurer le DNS en créant une zone primaire (energie.smartcity.lan), puis en accédant à la délégation.



Name	Type	Data
(same as parent folder)	Start of Authority (SOA)	[3] windows_serv_energie...
(same as parent folder)	Name Server (NS)	windows_serv_energie.sm...
192.168.20.1	Pointer (PTR)	Linux_Serv_ENERGIE.energ...
192.168.20.2	Pointer (PTR)	Windows_Serv_ENERGIE.e...

Linux serveur

Global

1. Mise à jour du système.
2. Installation de MariaDB.
 - Installation des paquets MariaDB.
 - Démarrage et activation de MariaDB.
 - Configuration sécurisée : Le script configure MariaDB en sécurisant l'accès avec un mot de passe root et en désactivant les options non sécurisées.
3. Configuration de Django avec MySQL.
 - Installation des dépendances:
 - Python 3 et Pip.
 - Développement MariaDB et Python.
 - Connecteurs MySQL pour Python.
 - Création d'un projet Django:
 - Le projet est initialisé avec `django-admin startproject`.
 - Les paramètres `settings.py` sont configurés pour utiliser une base de données MySQL avec les détails fournis.
 - Création de la base de données:
 - MariaDB est configurée avec une base de données et un utilisateur dédié.
 - Exécution des migrations Django.
 - Création d'un superutilisateur Django.
4. Configuration d'Apache:
 - Installation et démarrage du serveur web Apache.
 - Configuration pour écouter sur toutes les interfaces réseau (0.0.0.0) sur les ports HTTP et HTTPS.
 - Apache est prêt pour héberger les sites et interagir avec Django si nécessaire.
5. Configuration de Samba:
 - Installation et activation de Samba.
 - Configuration des permissions pour le répertoire partagé:
 - Le chemin `/home/admin/projet-inter/linux/` est partagé avec les permissions maximales.
 - Un fichier de configuration Samba est créé pour définir un partage non authentifié.
 - Ce fichier est inclus dans la configuration principale de Samba.
 - Redémarrage des services Samba.
6. Configuration des pare-feu:
 - Ouverture des ports pour Samba et Django.
7. Lancement des services:
 - Le serveur Django est démarré en arrière-plan et accessible via l'adresse réseau.
 - Apache et Samba sont également opérationnels pour gérer les requêtes web et les partages réseau.

Script de configuration : <https://github.com/trifoil/projet-inter/tree/linux/linux>

Local

1. Mise à jour du système:
 - Assurer que le système d'exploitation et les paquets sont à jour.
2. Installation et configuration d'Apache :
 - Installation des paquets nécessaires :
 - Apache est installé pour fournir un serveur web.
 - Démarrage et activation d'Apache:
 - Apache est démarré et configuré pour se lancer automatiquement au démarrage du système.
 - Configuration du réseau:
 - Apache est configuré pour écouter sur toutes les interfaces réseau pour HTTP et HTTPS.
 - Fichier PHP de test:
 - Un fichier PHP simple est ajouté pour tester la configuration.
3. Installation et configuration de MariaDB:
 - Installation de MariaDB:
 - MariaDB (base de données) est installée et prête à être configurée.
 - Démarrage et activation de MariaDB.
 - Configuration sécurisée:
 - Commande interactive pour sécuriser MariaDB.
 - Création d'un utilisateur admin avec des privileges:
 - Un utilisateur admin avec un mot de passe sécurisé est créé pour gérer les bases de données.
4. Installation et configuration de PhpMyAdmin:
 - Installation de PhpMyAdmin.
 - Interface web pour la gestion des bases de données.
 - Configuration des accès:
 - Permissions ajustées pour autoriser l'accès depuis n'importe où.
 - Redémarrage d'Apache.
5. Configuration du pare-feu:
 - Ouverture des ports nécessaires:
 - Les ports HTTP, HTTPS, et MySQL sont ouverts pour autoriser les connexions externes.
6. Sauvegarde des données:
 - Création d'un répertoire de sauvegarde:
 - Si nécessaire, un répertoire dédié aux sauvegardes est créé.
 - Sauvegarde du site web:
 - Les fichiers web sont archivés dans une archive compressée.
 - Sauvegarde de la base de données:
 - Un export complet de la base de données est réalisé via mysqldump.

Script de configuration :

https://github.com/AnthonyCodeDev/SmartCity_CDCGR10/blob/Ressources/ConfigLinuxServLocal/ScriptConfigLinuxLocale.sh

Plan de sauvegarde

Un plan de sauvegarde sous Linux a été mis en place pour garantir la sécurité et la disponibilité des données. Cela représente une stratégie essentielle, visant à définir les fichiers ou répertoires à sauvegarder, la fréquence des sauvegardes, les outils à utiliser et le lieu de stockage. Cette approche permet de minimiser les risques liés aux pannes, aux attaques ou aux erreurs humaines adapté la réalisation de projet.

Dans ce contexte, un script a été développé pour automatiser les sauvegardes des éléments essentiels du projet, à savoir :

- ❖ Le site web, dont les fichiers sont situés dans le répertoire `/var/www/energie`.
- ❖ La base de données MySQL associée, nommée `db_smartcity_energie`.

1. Fonctionnement du script

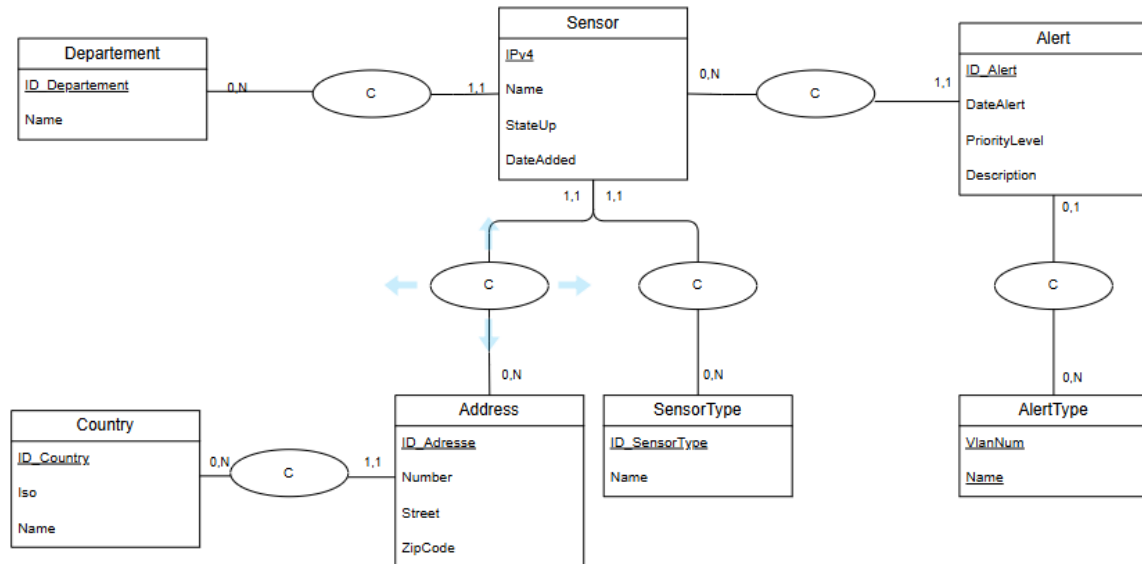
Ce script effectue des sauvegardes complètes en deux étapes :

1. Backup des fichiers du site web :
 - I. Le répertoire contenant les fichiers du site est compressé dans une archive `.tar.gz` pour réduire l'espace de stockage tout en conservant tous les fichiers nécessaires.
 - II. Chaque fichier de sauvegarde est horodaté pour un suivi clair et une organisation chronologique.
 - III. Exemple de fichier généré : `web_backup_2024-12-20_12-30-45.tar.gz`.
2. Backup de la base de données :
 - I. La base de données est exportée au format `.sql` via `mysqldump`, garantissant une copie complète et fidèle des données.
 - II. Les informations d'accès à la base (hôte, utilisateur, mot de passe) sont gérées pour sécuriser l'export.
 - III. Exemple de fichier généré : `db_backup_2024-12-20_12-30-45.sql`.

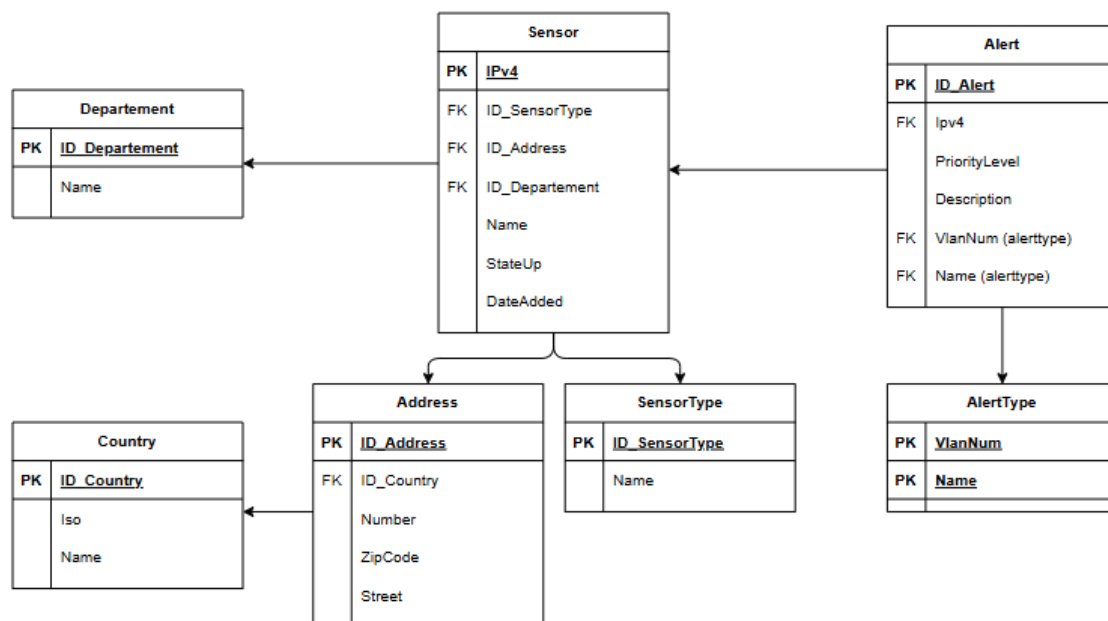
Modélisation Bases de données

Base de données globale

MCD Globale :

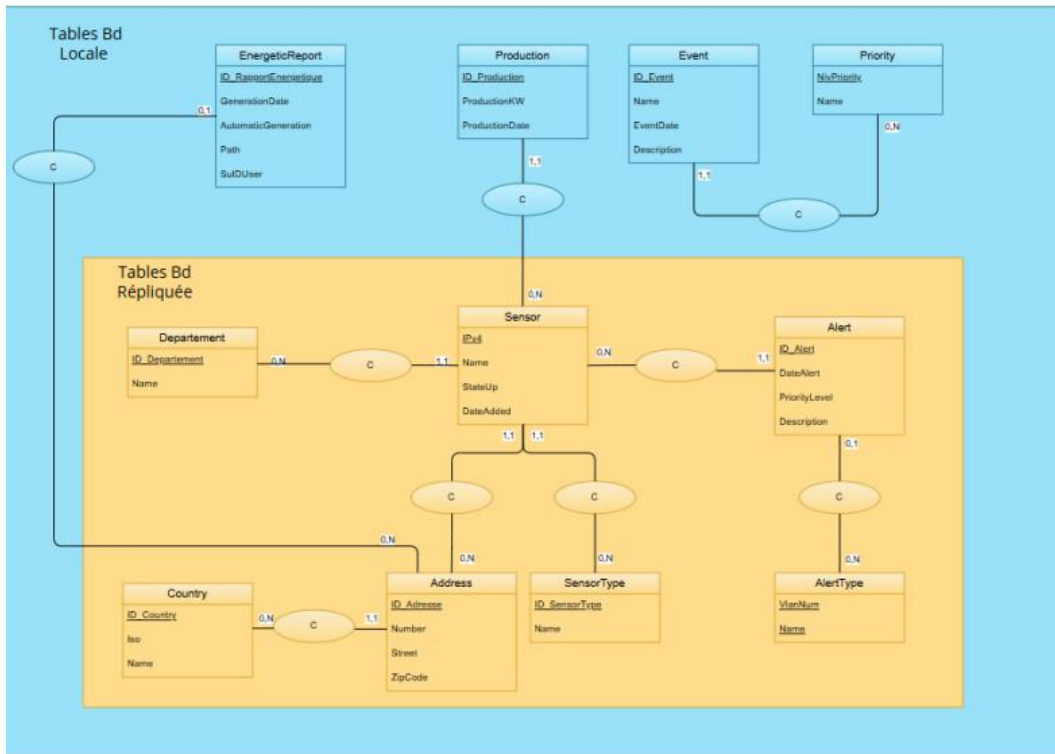


MLD Globale

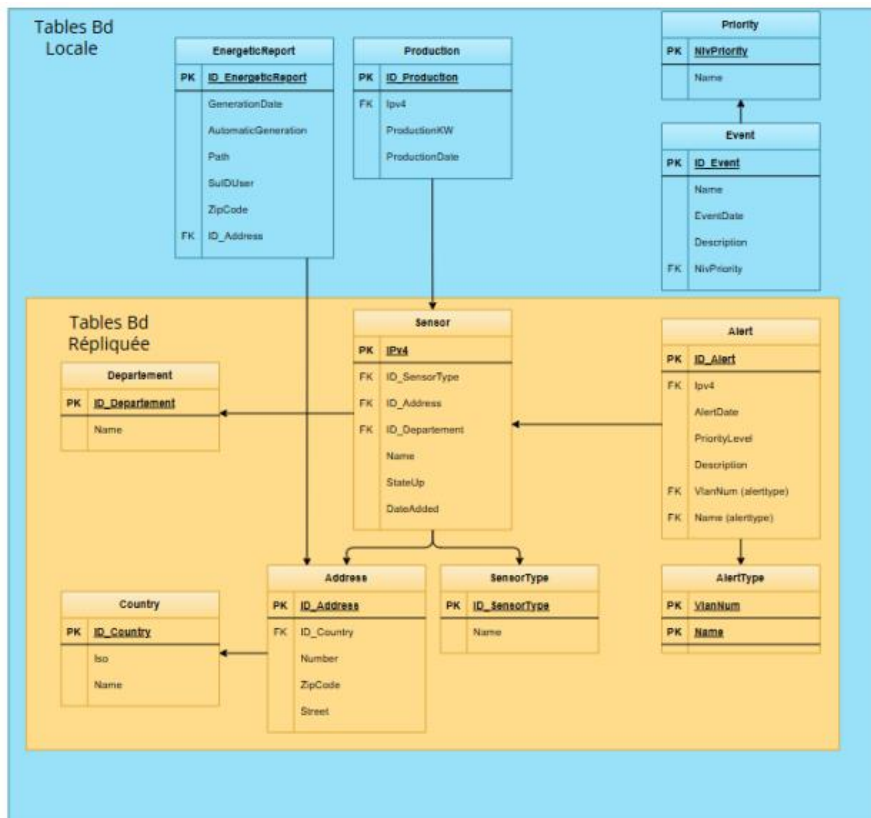


Base de données locale

MCD Locale :



MLD Locale :



A savoir: Les Events sont gérés localement et ne sont pas des alertes visibles sur l'application globale. Elles sont créées par des utilisateurs de l'application locale.

Replication Maste-Slave

Dans ce projet, une réplication Master-Slave a été mise en place pour garantir la redondance et la disponibilité des données.

Configuration Master-Slave :

- Le serveur Linux global a été configuré en tant que Master, hébergeant la base de données principale.
- Les serveurs Linux locaux ont été configurés en tant que Slaves, recevant automatiquement les mises à jour effectuées sur le Master.

Fonctionnement :

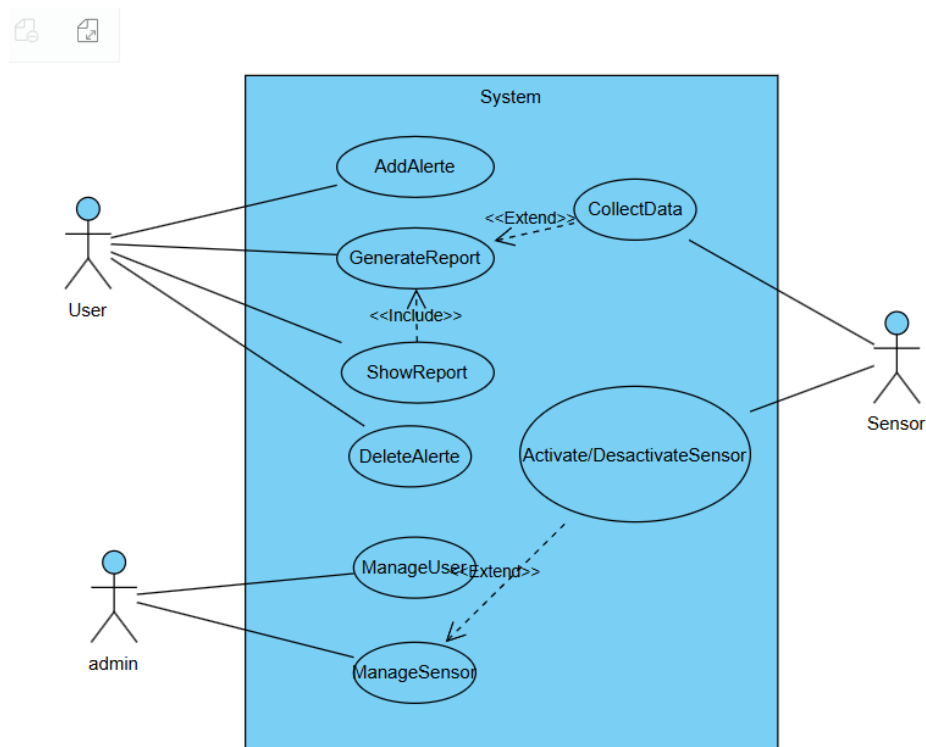
- Toute modification apportée à la base de données sur le Master est répliquée en temps réel sur les Slaves locaux.
- Cette configuration permet de sécuriser les données et de garantir leur disponibilité en cas de panne du serveur principal.

Validation :

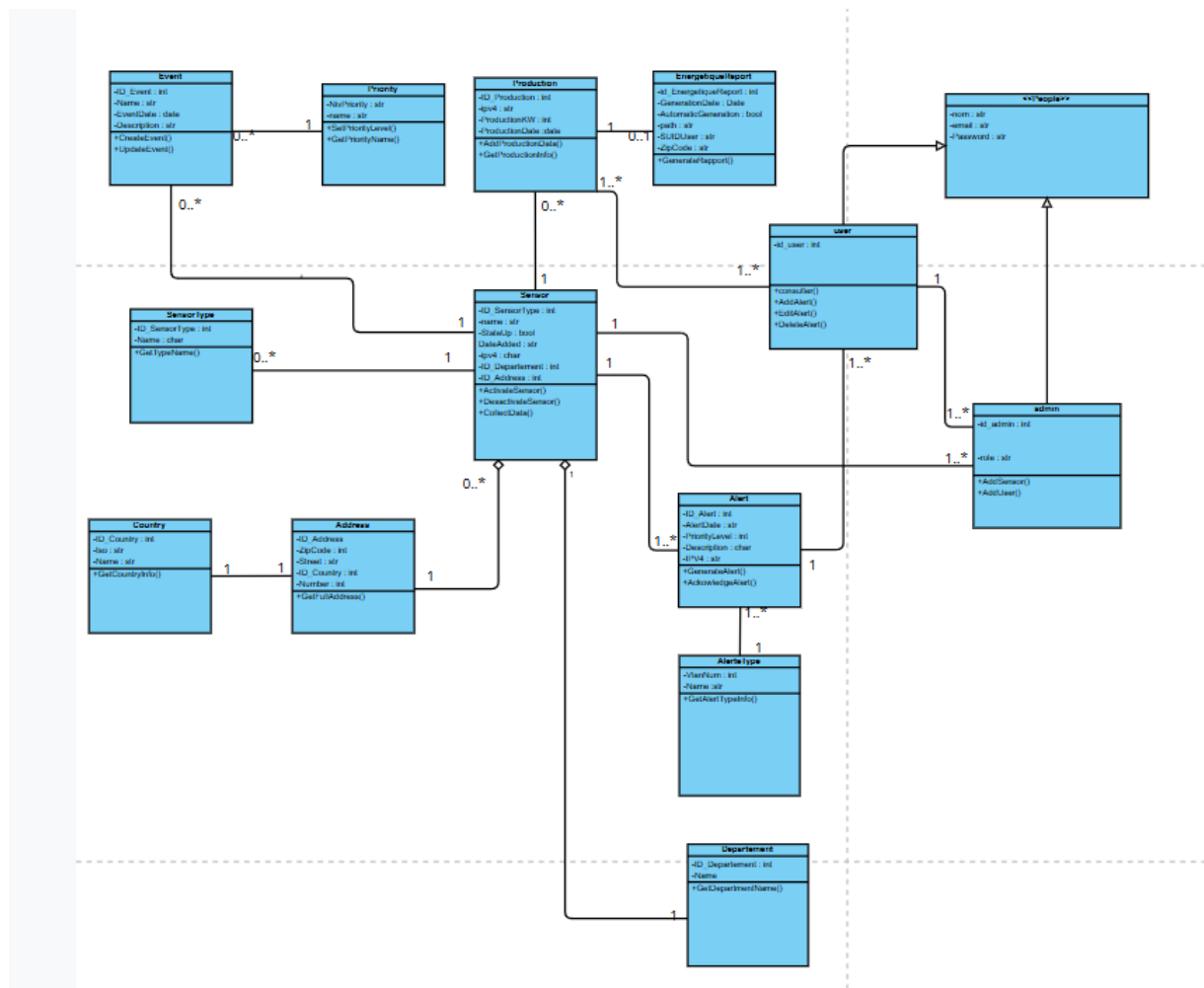
- La réplication Master-Slave a été testée avec succès sur les Slaves locaux, assurant une synchronisation fiable des données entre les serveurs.

Analyses UML

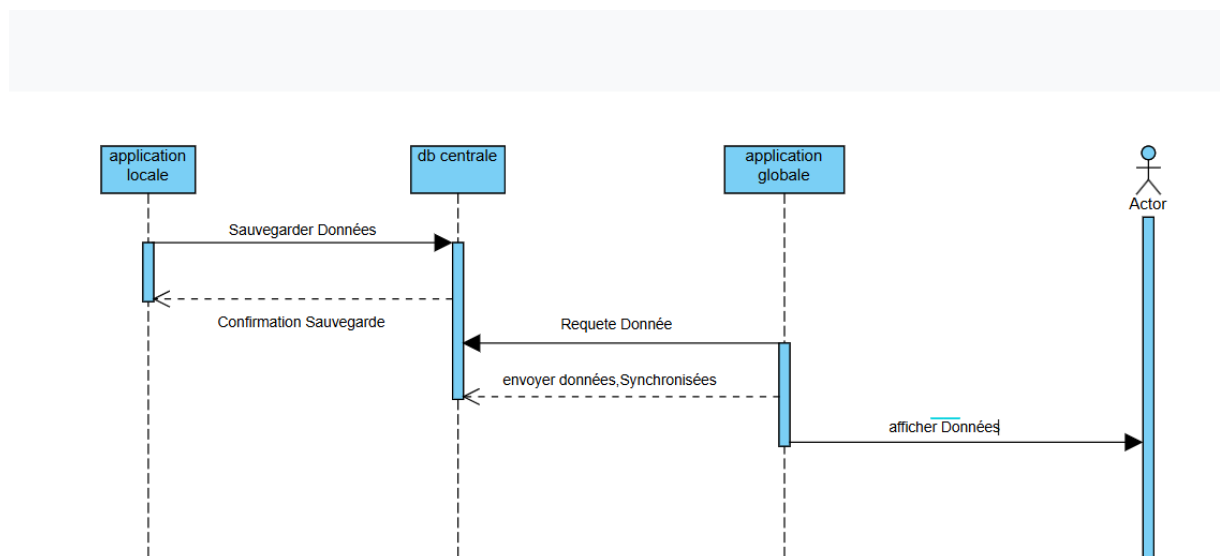
Use Case:



Class:



Séquence:

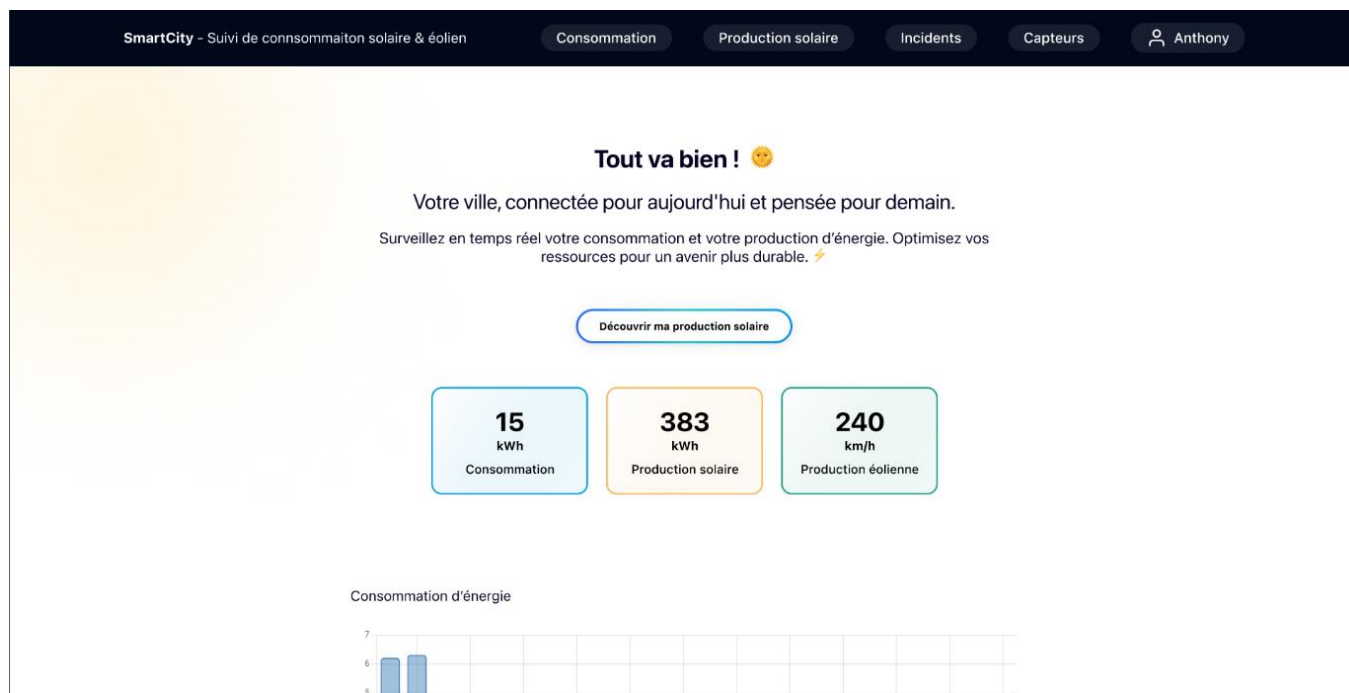


Maquettes application web

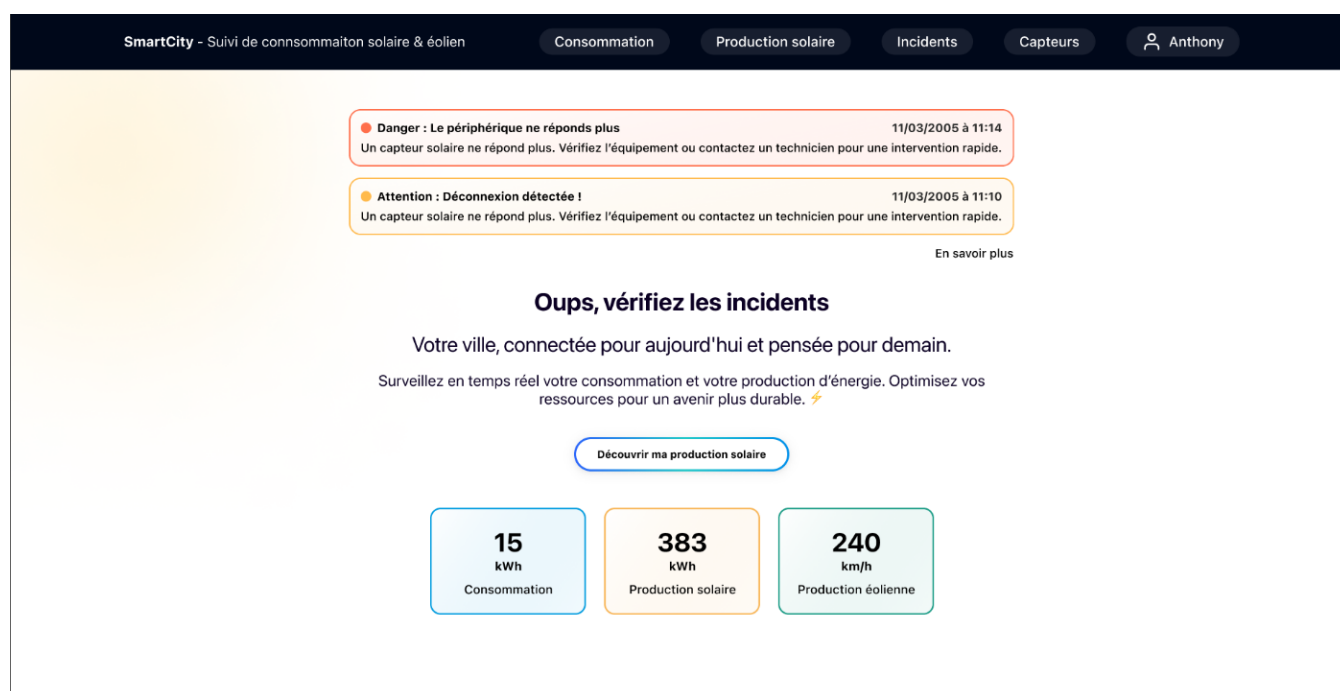
Site Locale

Wireframe

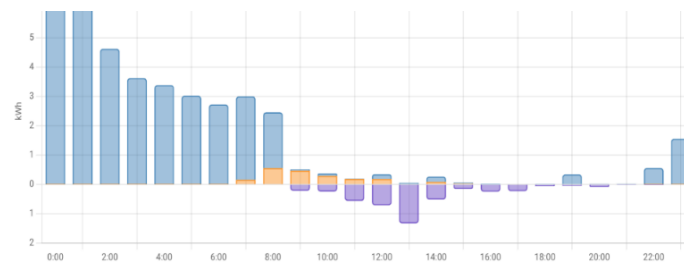
Dans le cas ou tout va bien :



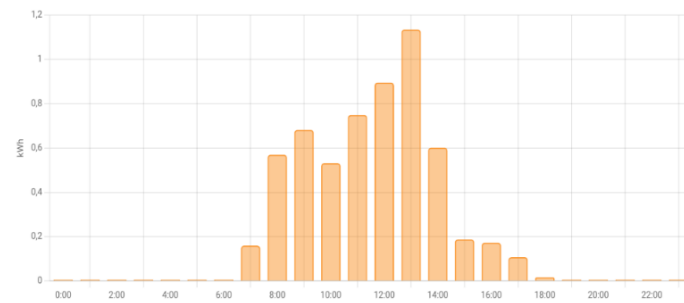
Si le système rencontre des problèmes :



En dessous de la page d'accueil :



Production solaire



Si le système rencontre des problèmes :

SmartCity - Suivi de consommation solaire & éolien

Consommation

Production solaire

Incidents

Capteurs

Anthony

Danger : Le périphérique ne réponds plus

11/03/2005 à 11:14

Un capteur solaire ne répond plus. Vérifiez l'équipement ou contactez un technicien pour une intervention rapide.

Attention : Déconnexion détectée !

11/03/2005 à 11:10

Un capteur solaire ne répond plus. Vérifiez l'équipement ou contactez un technicien pour une intervention rapide.

En savoir plus

Oups, vérifiez les incidents

Votre ville, connectée pour aujourd'hui et pensée pour demain.

Surveillez en temps réel votre consommation et votre production d'énergie. Optimisez vos ressources pour un avenir plus durable. ⚡

Découvrir ma production solaire

15

kWh

Consommation

383

kWh

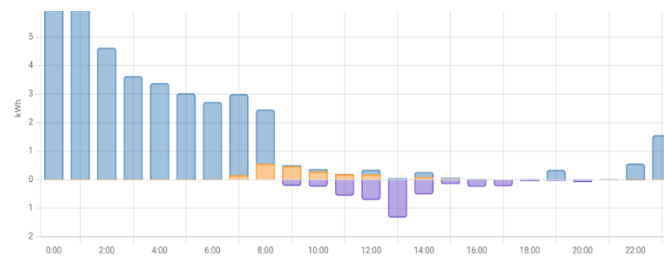
Production solaire

240

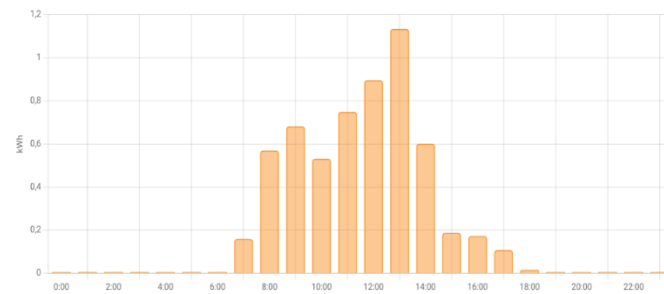
km/h

Production éolienne

En dessous de la page d'accueil :



Production solaire



La page contenant la liste des incidents :

SmartCity - Suivi de consommation solaire & éolien

Consommation

Production solaire

Incidents

Capteurs

Anthony

Créer un incident

Derniers incidents

Vérifiez les derniers incidents et intervenez si besoin.

11/03/2005 • 2 incidents

Danger : Le périphérique ne répond plus

11/03/2005 à 11:14

Un capteur solaire ne répond plus. Vérifiez l'équipement ou contactez un technicien pour une intervention rapide.

Attention : Déconnexion détectée !

11/03/2005 à 11:10

Un capteur solaire ne répond plus. Vérifiez l'équipement ou contactez un technicien pour une intervention rapide.

08/03/2005 • 1 incidents

Attention : Déconnexion détectée !

08/03/2005 à 15:12

Un capteur solaire ne répond plus. Vérifiez l'équipement ou contactez un technicien pour une intervention rapide.

01/03/2005 • 4 incidents

Attention : Déconnexion détectée !

01/03/2005 à 12:12

Un capteur solaire ne répond plus. Vérifiez l'équipement ou contactez un technicien pour une intervention rapide.

Attention : Déconnexion détectée !

01/03/2005 à 12:05

18

La listes des différents capteurs :

SmartCity - Suivi de consommation solaire & éolien

Consommation

Production solaire

Incidents

Capteurs

Anthony

Liste des capteurs

Vérifiez les derniers incidents et intervenez si besoin.

Capteurs solaire

Capteur Solaire #1

192.168.1.1 est actif. Production actuelle de 5 kWh.

Capteur Solaire #2

192.168.1.13 est actif. Production actuelle de 2 kWh.

Capteur Solaire #3

Un capteur solaire 192.168.1.1 ne répond plus.

Capteurs éolien

Capteur Éolien #1

192.168.1.1 est actif. Production actuelle de 21 kWh.

Capteur Éolien #2

Un capteur solaire 192.168.1.1 ne répond plus.

Page de connexion (dès l'arrivée sur le site, pour sécuriser les données du site etc)

SmartCity - Suivi de consommation solaire & éolien

👋 Je suis heureux de vous voir

Adresse email

jean.dupont@consommation.lan

Mot de passe

Se connecter

19

Prototype Fonctionnelle

Page de Création d'un incident (Nom, description, danger) :

SmartCity - Suivi de consommation solaire & éolien

Consommation

Production solaire

Incidents

Capteurs

Déconnexion

Derniers incidents

Vérifiez les derniers incidents et intervenez si besoin.

Créer un incident

Nom :

Entrez le nom de l'incident

Description :

Décrivez l'incident

Niveau :

Résolu

Créer

SmartCity - Suivi de consommation solaire & éolien

Consommation

Production solaire

Incidents

Capteurs

Déconnexion

Derniers incidents

Vérifiez les derniers incidents et intervenez si besoin.

Créer un incident

Je suis deux

Je suis deux Je suis deuxJe suis deuxJe suis deuxJe suis deuxJe suis deux

2024-12-17 10:34:39

Test

Je suis un Je suis un Je suis un Je suis un Je suis un

2024-12-17 10:34:26

Message de prévention en cas de suppression d'un incident :

andi

om

SC

anthony indique

Voulez-vous vraiment supprimer cet incident ?

OK

Annuler

/!\ Crud mis en place pour les incidents /!\

Créer un incident

Nom :

Je suis deux

Description :

Je suis deux Je suis deuxJe suis deuxJe suis deuxJe suis deuxJe suis deux

Niveau :

Danger

Mettre à jour

Page de génération de rapport complet :

SmartCity - Suivi de consommation solaire & éolien

Consommation

Production solaire

Incidents

Capteurs

Déconnexion

Générer un rapport de complet

Les rapports sont générés tous les jours à 00:00 UTC+1.

5
kWh
Consommation

5
kWh
Consommation

5
kWh
Consommation

Créer un rapport instantané

La génération du rapport peut prendre jusqu'à 1 minute.

Rapports de SmartCity

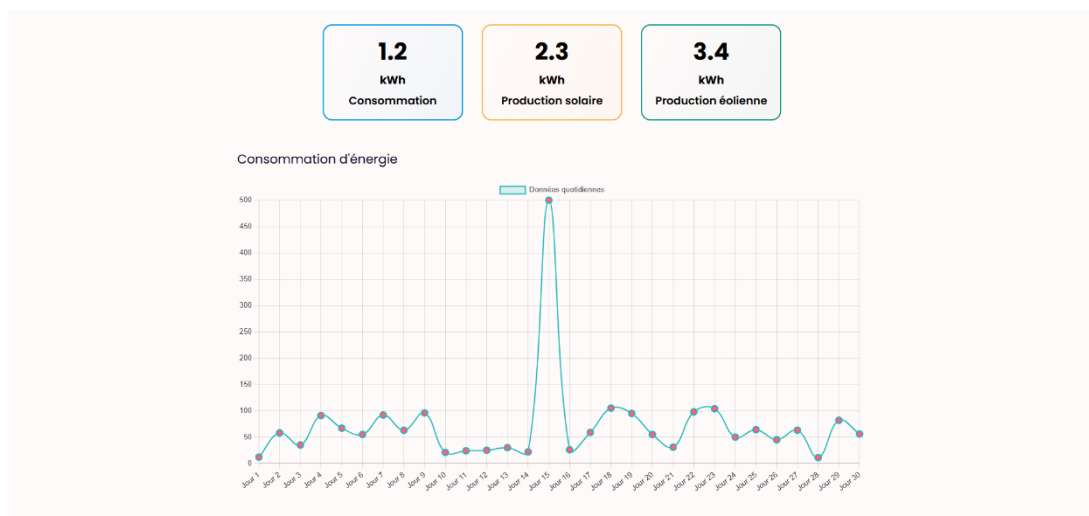
Rapport automatique

Accéder au rapport automatique du 10/10/2021

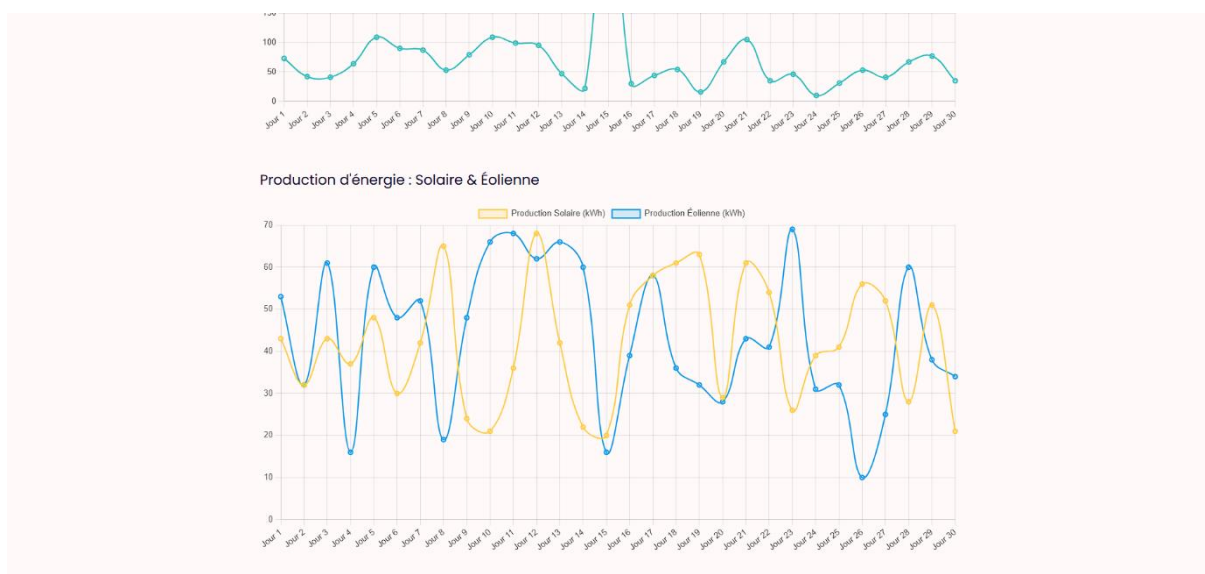
Rapport manuel

Accéder au rapport manuel du 10/10/2021

Affichage graphique de la consommation énergétique:



Affichage de la production solaire et éolienne .



Page de connexion:

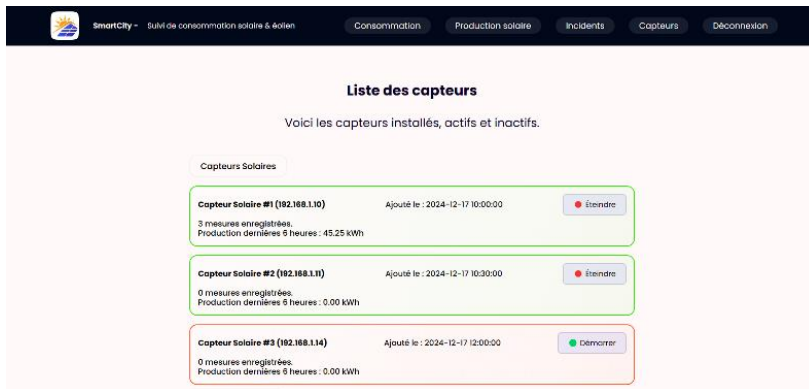
SmartCity - Suivi de consommation solaire & éolien

🔥 Je suis heureux de vous voir

Adresse email
jean.dupont@consommation.lan

Mot de passe
Mot de passe

Se connecter



Remarques :

Dynamique avec la base de données.

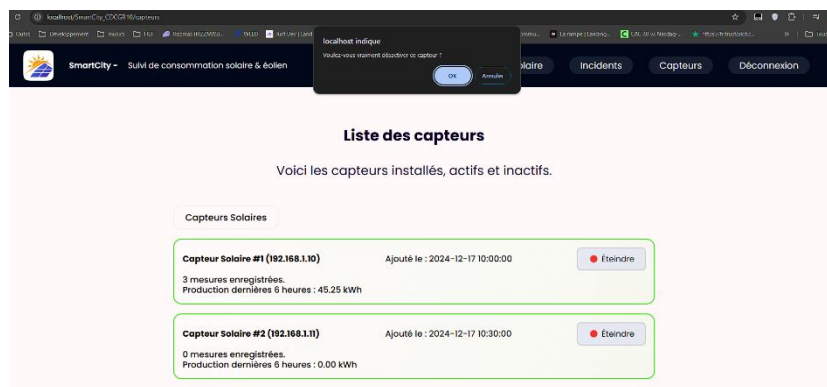
Fonctionnalité Éteindre/Démarrer le capteur.

Trié par en premier les ont, puis les on, et par on ou off, trier par le nombre de mesures enregistrées, et ensuite une somme de la production des 6 dernières heures.

Remarque et confirmation de désactivation/activation des capteurs.

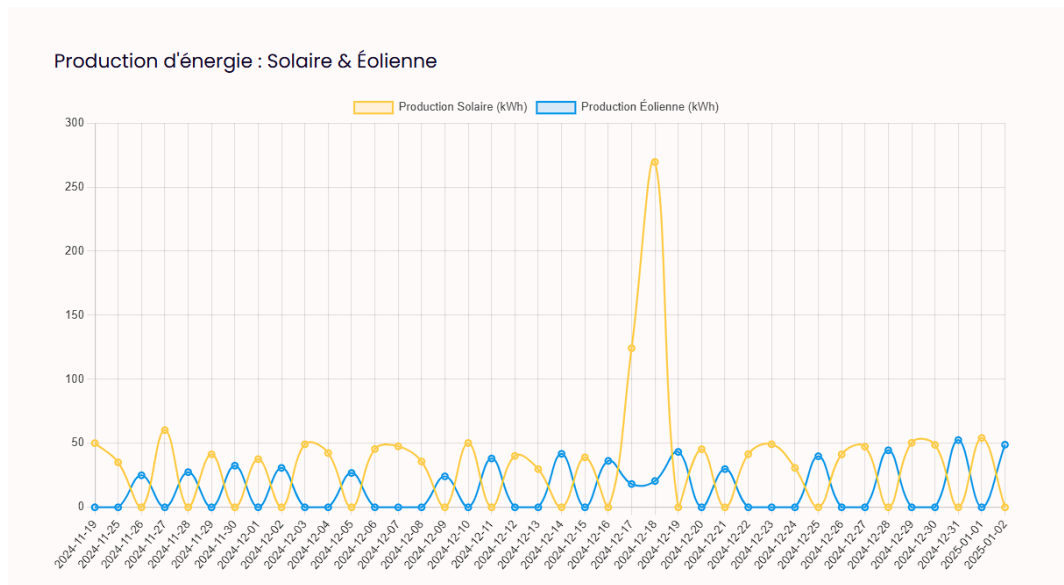
Le site web est quasiment terminé, une fois que l'AD sera terminée, les capteurs installés, tout fonctionnera quand ça sera relié.

A l'accueil, les infos des dernières 24h sont désormais affichées :

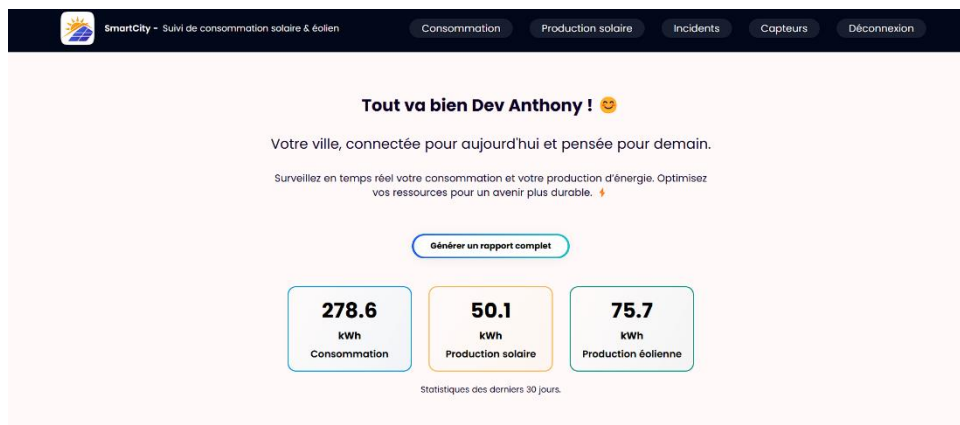


A l'accueil les productions sont affichées dynamiquement selon la base de données.

Affichage Dynamique :



Page lors de la connexion d'un utilisateurs:



L'affichage dynamique énoncé ci-dessus se réalise correctement !

Site Globale



Codes Python

Générer des données de capteurs aléatoires

Code disponible :

https://github.com/AnthonyCodeDev/SmartCity_CDCGR10/blob/GestionAutomatiqueDonnee/donneeCapteur.py

1. Connexion à la Base de Données

La première étape du processus consiste à établir une connexion à la base de données **MySQL**. Nous utilisons « **mysql.connector** » pour gérer le connexion avec **MySQL**.

```
def connect_to_db():
    """Connecte à la base de données et retourne la connexion."""
    try:
        conn = mysql.connector.connect(**db_config)
        return conn
    except mysql.connector.Error as err:
        print(f"Erreur de connexion: {err}")
        exit(1)
```

- ❖ **db_config** contient les informations de connexion nécessaires à l'accès à la base de données. Ce dictionnaire comprend l'hôte, l'utilisateur, le mot de passe et le nom de la base de données.
- ❖ La fonction « **connect_to_db** » tente de se connecter en utilisant ces informations. En cas d'échec (par exemple, si la base de données est inaccessible), une erreur est affichée et le programme se termine (exit(1)).

2. Récupération des Capteurs Actifs

Le code récupère les capteurs actifs dans la base de données. Les capteurs sont identifiés par leur état StateUp = 1, ce qui signifie qu'ils sont en état de fonctionnement.

```
def fetch_sensors(conn):
    """Récupère les capteurs actifs (StateUp = 1) et leurs informations depuis la table sensor."""
    cursor = conn.cursor(dictionary=True)
    query = "SELECT IPv4, ID_SensorType FROM sensor WHERE StateUp = 1"
    try:
        cursor.execute(query)
        sensors = cursor.fetchall()
        return sensors
    except mysql.connector.Error as err:
        print(f"Erreur lors de la récupération des capteurs: {err}")
        return []
    finally:
        cursor.close()
```

- ❖ « **fetch_sensors** » exécute une requête SQL pour obtenir les capteurs actifs, c'est-à-dire ceux dont StateUp = 1.

- ❖ La méthode « **cursor.fetchall()** » permet de récupérer toutes les lignes renvoyées par la requête sous forme de liste de dictionnaires, ce qui facilite l'accès aux données (par exemple, l'adresse IP et le type de capteur).

3. Insertion des Données dans la Table capteurs_energie

Une fois les capteurs récupérés, des données aléatoires sont insérées dans la table « **capteurs_energie** ». Cela simule la collecte d'informations sur la production d'énergie des capteurs.

```
def insert_into_capteurs_energie(conn, sensors):
    """Insère une entrée aléatoire dans la table capteurs_energie."""
    if not sensors:
        print("Aucun capteur disponible pour l'insertion.")
        return

    cursor = conn.cursor()
    sensor = random.choice(sensors)
    ip = sensor['IPv4']
    type_energie = 'solaire' if sensor['ID_SensorType'] == 1 else 'éolienne'
    valeur = round(random.uniform(1, 50), 2)
    date_mesure = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    query = (
        "INSERT INTO capteurs_energie (id_capteur, type_energie, valeur, date_mesure) "
        "VALUES (%s, %s, %s, %s)"
    )
    data = (ip, type_energie, valeur, date_mesure)

    try:
        cursor.execute(query, data)
        conn.commit()
        print(f"Donnée insérée dans capteurs_energie: {data}")
    except mysql.connector.Error as err:
        print(f"Erreur lors de l'insertion dans capteurs_energie: {err}")
    finally:
        cursor.close()
```

- ❖ « **insert_into_capteurs_energie** » choisit un capteur actif aléatoire à partir de la liste des capteurs.
- ❖ En fonction du type de capteur, il génère une valeur d'énergie (aléatoire entre 1 et 50) et l'insère dans la table « **capteurs_energie** » avec la date actuelle.
- ❖ La requête SQL utilise des paramètres (%s) pour éviter les injections SQL.

4. Insertion des Données dans la Table consommation_energie

Similaire à l'insertion dans « **capteurs_energie** », une entrée aléatoire est insérée dans la table « **consommation_energie** », simulant une consommation d'énergie dans des adresses spécifiques.

```

def insert_into_consommation_energie(conn):
    """Insère une entrée aléatoire dans la table consommation_energie."""
    cursor = conn.cursor()
    id_adresse = random.randint(1, 100) # Générer un id_adresse aléatoire
    consommation = round(random.uniform(1, 5), 2)
    date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    query = (
        "INSERT INTO consommation_energie (id_adresse, consommation, date) "
        "VALUES (%s, %s, %s)"
    )
    data = (id_adresse, consommation, date)

    try:
        cursor.execute(query, data)
        conn.commit()
        print(f"Donnée insérée dans consommation_energie: {data}")
    except mysql.connector.Error as err:
        print(f"Erreur lors de l'insertion dans consommation_energie: {err}")
    finally:
        cursor.close()

```

- ❖ « **insert_into_consommation_energie** » génère une entrée de la db aléatoire pour l'adresse et la consommation énergétique. Les valeurs sont insérées dans la table correspondante avec la date du moment.

5. Tests Unitaires

Des tests unitaires ont été réalisés pour vérifier le bon fonctionnement des fonctions. Pour simuler la connexion à la base de données, l'exécution des requêtes SQL et les insertions de données.

Les tests couvrent les scénarios suivants :

- ❖ **Connexion réussie à la base de données.**
- ❖ **Échec de la connexion à la base de données.**
- ❖ **Récupération correcte des capteurs.**
- ❖ **Insertion correcte des données dans les tables « capteurs_energie » et « consommation_energie ».**

Les tests permettent de s'assurer que chaque partie du code fonctionne comme prévu (sans nécessiter de connexion réelle à la base de données).

Détecter des surcharges automatiquement et envoyer une alerte

Code disponible :

https://github.com/AnthonyCodeDev/SmartCity_CDCGR10/blob/GestionAutomatiqueDonnee/detect_sensor_overload.py

1. Récupération de la Production des 7 Derniers Jours

Le code récupère les données de production pour chaque capteur actif sur les 7 derniers jours. Ces données sont regroupées par capteur et totalisées.

```
def fetch_last_week_production(conn):
    """Récupère la production des capteurs pour les 7 derniers jours."""
    cursor = conn.cursor(dictionary=True)
    end_date = datetime.now()
    start_date = end_date - timedelta(days=7)

    query = (
        "SELECT id_capteur, SUM(valeur) AS total_production "
        "FROM capteurs_energie "
        "WHERE date_mesure BETWEEN %s AND %s "
        "GROUP BY id_capteur"
    )
    cursor.execute(query, (start_date, end_date))
    results = cursor.fetchall()
    cursor.close()
    return results
```

- ❖ La méthode « **fetch_last_week_production** » exécute une requête SQL pour calculer la somme de la production (**SUM(valeur)**) de chaque capteur (**id_capteur**) sur une période de 7 jours.
Les résultats sont retournés sous forme de liste de dictionnaires.

3. Calcul de la Moyenne de Production Anterieure

Cette étape calcule la moyenne de production pour un capteur spécifique sur les périodes précédant les 7 derniers jours.

```
def fetch_previous_average_production(conn, id_capteur, start_date):
    """Récupère la production moyenne avant une période donnée."""
    cursor = conn.cursor(dictionary=True)
    query = (
        "SELECT AVG(valeur) AS avg_production "
        "FROM capteurs_energie "
        "WHERE id_capteur = %s AND date_mesure < %s"
    )
    cursor.execute(query, (id_capteur, start_date))
    result = cursor.fetchone()
    cursor.close()
    return result['avg_production'] if result['avg_production'] else 0
```

- ❖ La fonction « **fetch_previous_average_production** » exécute une requête SQL pour calculer la moyenne (**AVG(valeur)**) des valeurs de production pour un capteur donné (id_capteur) **avant une date spécifique**.

4. Vérification des Alertes Récentes

Ce bout de code vérifie si une alerte a été récemment enregistré pour éviter les doublons dans la détection d'anomalies.

```
def has_recent_alert(conn, id_capteur):
    """Vérifie si une alerte a été envoyée pour un capteur dans les dernières 24 heures."""
    cursor = conn.cursor(dictionary=True)
    check_date = datetime.now() - timedelta(hours=24)

    query = (
        "SELECT COUNT(*) AS recent_alerts "
        "FROM alertes_surcharge "
        "WHERE id_capteur = %s AND date_signalement > %s"
    )
    cursor.execute(query, (id_capteur, check_date))
    result = cursor.fetchone()
    cursor.close()
    return result['recent_alerts'] > 0
```

- ❖ La fonction « **has_recent_alert** » vérifie dans la table « **alertes_surcharge** » si une alerte a été enregistrée pour le capteur dans les dernières 24 heures.
Cela évite de générer plusieurs alertes pour la même anomalie.

5. Insertion d'une Nouvelle Alerte

Lorsqu'une anomalie est détectée et qu'il n'existe pas d'alerte récente, une nouvelle alerte est ajoutée à la table « **alertes_surcharge** ».

```
def insert_alert(conn, id_capteur, description, niveau):
    """Insère une alerte dans la table alertes_surcharge."""
    cursor = conn.cursor()
    query = (
        "INSERT INTO alertes_surcharge (id_capteur, description, niveau, date_signalement) "
        "VALUES (%s, %s, %s, %s)"
    )
    data = (id_capteur, description, niveau, datetime.now())
    try:
        cursor.execute(query, data)
        conn.commit()
        print(f"Alerte insérée: {data}")
    except mysql.connector.Error as err:
        print(f"Erreur lors de l'insertion de l'alerte: {err}")
    finally:
        cursor.close()
```

La fonction « **insert_alert** » insère une alerte avec un niveau (critique/moyen/...), une description et la date actuelle dans la table « **alertes_surcharge** ».

6. Vérification des Anomalies de Production

La fonction principale coordonne **toutes les étapes pour détecter les anomalies et insérer des alertes si nécessaire.**

```
def check_production_anomaly():
    """Vérifie les anomalies de production et insère des alertes si nécessaire."""
    conn = connect_to_db()
    try:
        last_week_data = fetch_last_week_production(conn)

        for entry in last_week_data:
            id_capteur = entry['id_capteur']
            total_production = entry['total_production']
            start_date = datetime.now() - timedelta(days=7)

            previous_avg = fetch_previous_average_production(conn, id_capteur, start_date)

            if previous_avg > 0 and total_production > 2 * previous_avg:
                if not has_recent_alert(conn, id_capteur):
                    niveau = 'critique' if total_production > 3 * previous_avg else 'moyen'
                    description = (
                        f"Surcharge détectée: production {total_production} kWh, "
                        f"soit +{round((total_production - previous_avg) / previous_avg * 100)}% par rapport à la "
                        f"moyenne."
                    )
                    insert_alert(conn, id_capteur, description, niveau)
                else:
                    print(f"Alerte récente déjà présente pour le capteur {id_capteur}, aucune nouvelle alerte "
                        f"envoyée.")
            finally:
                conn.close()
```

Cette fonction combine toutes les étapes précédentes :

- ❖ Récupération des données de production des 7 derniers jours.
- ❖ Comparaison avec les moyennes passées.
- ❖ Vérification des alertes récentes.
- ❖ Envoi d'une nouvelle alerte en cas d'anomalie détectée.

Tests Unitaires

Des tests unitaires ont été réalisés pour valider les fonctions principales, incluant :

- La connexion à la base de données.
- La récupération des données de production.
- Le calcul des moyennes passées.
- La vérification des alertes récentes.
- L'insertion d'alertes dans la base.

Ces tests simulent les interactions avec la base de données.

Codes PHP

Système d'Authentification via LDAP

Code disponible :

https://github.com/AnthonyCodeDev/SmartCity_CDCGR10/blob/SiteWeb/modele/AuthModele.php

1. Connexion au Serveur LDAP

Le code fait une connexion sécurisée **au serveur LDAP** spécifié via son URL et son port.

```
$ldapConnection = ldap_connect($this->ldapHost, $this->ldapPort);

if (!$ldapConnection) {
    return false; // Échec de connexion au serveur LDAP
}

ldap_set_option($ldapConnection, LDAP_OPT_PROTOCOL_VERSION, 3);
ldap_set_option($ldapConnection, LDAP_OPT_REFERRALS, 0);
```

- ❖ La fonction **ldap_connect** établit une connexion avec l'hôte et le port LDAP définis.
- ❖ Les options **LDAP_OPT_PROTOCOL_VERSION** et **LDAP_OPT_REFERRALS** sont configurées pour s'assurer que la communication respecte le protocole LDAP version 3.

2. Authentification Initiale

Une première authentification est effectuée avec le format **username@domain** pour vérifier les informations d'identification.

```
$ldapBind = @ldap_bind($ldapConnection, $username . "@smartcity.lan", $motDePasse);

if (!$ldapBind) {
    $_SESSION['error'] = "Échec de l'authentification : mot de passe incorrect.";
    ldap_close($ldapConnection);
    return false;
}
```

- ❖ **ldap_bind** tente de lier l'utilisateur avec **son nom d'utilisateur** et **son mot de passe**.
- ❖ En cas d'échec, un message d'erreur est enregistré dans la session et **la connexion est fermée**.

3. Recherche de l'Utilisateur dans le Répertoire

Une requête **LDAP** est exécutée pour récupérer les informations de l'utilisateur à l'aide d'un filtre basé sur son **sAMAccountName**.


```

$searchFilter = "(sAMAccountName=$username)";
$searchResult = @ldap_search($ldapConnection, $this->ldapBaseDn, $searchFilter);

if (!$searchResult) {
    $_SESSION['error'] = "Erreur LDAP lors de la recherche : " . ldap_error($ldapConnection);
    ldap_close($ldapConnection);
    return false;
}

```

- ❖ **ldap_search** effectue une recherche dans le répertoire **LDAP** à partir de la base **DN** spécifiée (**\$this->ldapBaseDn**).

Si la recherche échoue, **une erreur descriptive est ajoutée à la session**.

4. Validation de l'Utilisateur

Les résultats de la recherche sont analysés pour vérifier si l'utilisateur existe. Si c'est le cas, une seconde liaison est effectuée avec le **DN (Distinguished Name)** complet de l'utilisateur pour **valider son mot de passe**.

```

$entries = ldap_get_entries($ldapConnection, $searchResult);

if ($entries['count'] === 0) {
    $_SESSION['error'] = "Utilisateur non trouvé dans le répertoire LDAP.";
    ldap_close($ldapConnection);
    return false;
}

$userDn = $entries[0]['dn'];

if (@ldap_bind($ldapConnection, $userDn, $motDePasse)) {
    $memberOf = $entries[0]['memberof'] ?? [];
    $role = 'user'; // Rôle par défaut

    if (is_array($memberOf) && in_array('CN=GG_AdminEnergie,OU=GG,OU=Groups,DC=smartcity,DC=lan', $memberOf)) {
        $role = 'admin';
    }

    ldap_unbind($ldapConnection);

    return [
        'nom' => $entries[0]['sn'][0] ?? '',
        'prenom' => $entries[0]['givenname'][0] ?? '',
        'email' => $entries[0]['mail'][0] ?? '',
        'role' => $role
    ];
} else {
    $_SESSION['error'] = "Échec de l'authentification : mot de passe incorrect.";
}

```

- ❖ La méthode **ldap_get_entries** permet de récupérer les informations utilisateur sous forme de tableau.
- ❖ La vérification du champ **memberof** détermine si l'utilisateur appartient au groupe **GG_AdminEnergie**, ce qui lui **confère le rôle admin**.

5. Gestion des Rôles

Le module **attribue un rôle par défaut** (user) à chaque utilisateur, sauf s'il appartient au groupe spécifique des **administrateurs**.

```

if (is_array($memberOf) && in_array('CN=GG_AdminEnergie,OU=GG,OU=Groups,DC=smartcity,DC=lan', $memberOf)) {
    $role = 'admin';
}

```

- ❖ La liste des groupes auxquels l'utilisateur appartient est parcourue pour vérifier la présence du groupe **GG_AdminEnergie**.
- ❖ **En cas de correspondance**, le rôle est modifié en **admin**.

Tests Unitaires

Des tests unitaires ont été réalisés pour vérifier les cas suivants :

- **Connexion au serveur LDAP avec des identifiants valides.**
- **Échec de connexion avec des identifiants incorrects.**
- **Recherche d'un utilisateur inexistant.**
- **Gestion des rôles (user et admin).**
- **Échec de connexion au serveur LDAP simulant un serveur inaccessible.**

Ces tests garantissent le bon fonctionnement des principales fonctionnalités et la gestion appropriée des erreurs.

Gérer les capteurs de SmartCity Energie

Code disponible :

https://github.com/AnthonyCodeDev/SmartCity_CDCGR10/blob/SiteWeb/modele/CapteurModule.php

2. Calcul de la Production des Dernières Heures

Ce code calcule **la production totale d'un capteur donné** sur une période définie en heures.

```

public function calculerProductionDernieresHeures($ipCapteur, $heures = 6) {
    /*
    QUI: Vergeylen Anthony
    QUAND: 18-12-2024
    QUOI: Calculer la production des dernières heures

    Arguments: ipCapteur (string), heures (int)
    Return: int
    */
    $stmt = $this->pdo->prepare("
        SELECT SUM(valeur) as total
        FROM capteurs_energie
        WHERE id_capteur = :ipCapteur
        AND date_mesure >= NOW() - INTERVAL :heures HOUR
    ");
    $stmt->bindParam(':ipCapteur', $ipCapteur);
    $stmt->bindParam(':heures', $heures, PDO::PARAM_INT);
    $stmt->execute();
    $result = $stmt->fetch();
    return $result['total'] ?? 0; // Retourne 0 si aucun résultat
}

```

- ❖ La méthode exécute une requête SQL qui additionne (**SUM(valeur)**) les valeurs enregistrées pour un capteur donné (id_capteur) sur une période récente (**NOW()** - **INTERVAL :heures HOUR**).
- ❖ Si aucune donnée n'est trouvée, la méthode retourne 0.

3. Récupération des Capteurs par Type

Cette méthode **retourne une liste de capteurs filtrés par leur type**, incluant des informations comme leur **état** et le **nombre de données associées**.

```
public function recupererCapteursParType($type) {
    /*
    QUI: Vergeylen Anthony
    QUAND: 18-12-2024
    QUOI: Récupérer les capteurs par type

    Arguments: type (int)
    Return: array
    */
    $stmt = $this->pdo->prepare("
        SELECT
            s.IPv4,
            s.Name,
            s.StateUp,
            s.DateAdded,
            COUNT(c.id) as nombre_donnees
        FROM sensor s
        LEFT JOIN capteurs_energie c ON s.IPv4 = c.id_capteur
        WHERE s.ID_SensorType = :type
        GROUP BY s.IPv4, s.Name, s.StateUp, s.DateAdded
        ORDER BY s.StateUp DESC, nombre_donnees DESC, s.DateAdded DESC
    ");
    $stmt->bindParam(':type', $type, PDO::PARAM_INT);
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
```

- ❖ La méthode exécute une jointure entre les tables « **sensor** » et « **capteurs_energie** » pour associer les capteurs à leurs données.
- ❖ Elle regroupe les résultats par capteur (**GROUP BY**) et les trie selon plusieurs critères : **état, nombre de données, et date d'ajout.**

4. Récupération de Tous les Capteurs

Cette méthode retourne une liste complète de **tous les capteurs enregistrés**, avec des informations de base comme leur état et leur date d'ajout.

```
public function recupererCapteurs() {  
    /*  
    QUI: Vergeylen Anthony  
    QUAND: 18-12-2024  
    QUOI: Récupérer tous les capteurs avec leur statut  
  
    Arguments: aucun  
    Return: array  
    */  
    $stmt = $this->pdo->query("  
        SELECT IPv4, Name, StateUp, DateAdded  
        FROM sensor  
        ORDER BY DateAdded DESC  
    ");  
    return $stmt->fetchAll();  
}
```

La méthode exécute une requête SQL pour récupérer toutes les informations des capteurs (sensor) et les trie par date d'ajout décroissante (**ORDER BY DateAdded DESC**).

5. Comptage des Données d'un Capteur

Cette méthode **compte le nombre de données enregistrées** pour un capteur spécifique.

```
public function compterDonneesCapteur($ipCapteur) {  
    /*  
    QUI: Vergeylen Anthony  
    QUAND: 18-12-2024  
    QUOI: Compter le nombre de données de consommation pour chaque capteur  
  
    Arguments: ipCapteur (string)  
    Return: int  
    */  
    $stmt = $this->pdo->prepare("  
        SELECT COUNT(*) as total  
        FROM capteurs_energie  
        WHERE id_capteur = :ipCapteur  
    ");  
    $stmt->bindParam(':ipCapteur', $ipCapteur);  
    $stmt->execute();  
    $result = $stmt->fetch();  
    return $result['total'];  
}
```

La méthode utilise **COUNT(*)** pour compter toutes les entrées associées à un capteur spécifique dans la table `capteurs_energie`.

6. Mise à Jour de l'État d'un Capteur

Cette méthode met à jour l'état (**StateUp**) d'un capteur pour indiquer s'il est actif (**1**) ou non (**0**).

```
public function mettreAJourEtatCapteur($ipCapteur, $etat) {
    /*
     * QUI: Vergeylen Anthony
     * QUAND: 18-12-2024
     * QUOI: Mettre à jour l'état d'un capteur (Start/Stop)

     Arguments: ipCapteur (string), etat (bool)
     Return: aucun
     */
    $stmt = $this->pdo->prepare("
        UPDATE Sensor
        SET StateUp = :etat
        WHERE IPv4 = :ipCapteur
    ");
    $stmt->bindParam(':etat', $etat, PDO::PARAM_BOOL);
    $stmt->bindParam(':ipCapteur', $ipCapteur);
    $stmt->execute();
}
```

La méthode exécute une requête **SQL UPDATE** pour modifier l'état du capteur identifié par son adresse IP (IPv4).

Tests Unitaires

Des tests unitaires ont été réalisés pour valider les fonctionnalités suivantes :

- Calcul de la production totale sur une période.
- Récupération des capteurs par type et vérification de leurs informations.
- Mise à jour de l'état d'un capteur.
- Comptage des données associées à un capteur.
- Récupération de la liste complète des capteurs.

Ces tests simulent les interactions avec la base de données garantissant que **chaque méthode produit les résultats attendus.**

Accueil du site web

Code disponible :

https://github.com/AnthonyCodeDev/SmartCity_CDCGR10/blob/SiteWeb/modele/HomeModel.e.php

1. Récupération du Message de Bienvenue

Cette méthode retourne un message de bienvenue **basé sur les informations utilisateur** disponibles dans la session.

```
public function getWelcomeMessage() {  
    /*  
    QUI: Vergeylen Anthony  
    QUAND: 18-12-2024  
    QUOI: Retourne un message de bienvenue  
  
    Arguments: aucun  
    Return: string  
    */  
    $utilisateur = $_SESSION['utilisateur'] ?? null;  
  
    if (!$utilisateur) {  
        return "Tout va mal ! 🤖";  
    }  
  
    $nomUtilisateur = ucwords("$utilisateur[nom] $utilisateur[prenom]");  
  
    return "Tout va bien $nomUtilisateur ! 😊";  
}
```

- ❖ La méthode vérifie si les **informations utilisateur** sont présentes dans **\$_SESSION**.
- ❖ Si elles sont disponibles, elle retourne un **message personnalisé** contenant le nom et prénom formatés.

2. Récupération de la Production d'Énergie des Dernières 24 Heures

Cette méthode **calcule la production totale d'énergie sur les dernières 24 heures** pour un type d'énergie donné (**1 pour solaire, 2 pour éolien**).

```

public function recupererProductiondER($typeEnergie) {
    /*
    QUI: Vergeylen Anthony
    QUAND: 18-12-2024
    QUOI: Retourne la production d'énergie des 24 dernières heures

    Arguments: typeEnergie (int)
    Return: int
    */
    $stmt = $this->pdo->prepare("
        SELECT SUM(valeur) as total
        FROM capteurs_energie
        WHERE type_energie = :typeEnergie
        AND date_mesure >= NOW() - INTERVAL 24 HOUR
    ");
    $stmt->bindParam(':typeEnergie', $typeEnergie, PDO::PARAM_INT);
    $stmt->execute();
    $result = $stmt->fetch();
    return $result['total'] ?? 0; // Retourne 0 si aucune donnée
}

```

- ❖ La méthode exécute une requête SQL pour additionner (**SUM(valeur)**) les valeurs d'énergie mesurées **au cours des dernières 24 heures**.
- ❖ Le type d'énergie est passé en paramètre, permettant de **différencier solaire et éolien**.

3. Récupération des Informations Globales

Ce code **récupère et retourne les données globales** de consommation et de production pour après les afficher sur la vue de la page !

```

public function recupererInformationsGlobales() {
    /*
    QUI: Vergeylen Anthony
    QUAND: 18-12-2024
    QUOI: Retourne les informations globales de consommation et production

    Arguments: aucun
    Return: array
    */
    $consommation = $this->recupererConsommationdER(); // Consommation sur 24h
    $productionSolaire = $this->recupererProductiondER(1); // Type 1 pour solaire
    $productionEolienne = $this->recupererProductiondER(2); // Type 2 pour éolien

    return [
        'consommation' => $consommation,
        'productionSolaire' => $productionSolaire,
        'productionEolienne' => $productionEolienne,
    ];
}

```

La méthode appelle d'autres fonctions pour récupérer :

- **La consommation énergétique** des 24 dernières heures.
- **La production d'énergie solaire et éolienne**.

4. Récupération de la Consommation des Derniers 30 Jours

Cette méthode retourne **une liste des consommations énergétiques** par jour **sur les 30 derniers jours**.

```
public function recupererConsommation30Jours() {
    /*
    QUI: Vergeylen Anthony
    QUAND: 18-12-2024
    QUOI: Retourne la consommation d'énergie des 30 derniers jours

    Arguments: aucun
    Return: array
    */
    $stmt = $this->pdo->query("
        SELECT DATE(date) as jour, SUM(consommation) as total
        FROM consommation_energie
        WHERE date >= NOW() - INTERVAL 30 DAY
        GROUP BY DATE(date)
        ORDER BY DATE(date) ASC
    ");
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
```

- La requête regroupe les données par date (**GROUP BY DATE(date)**) et calcule la somme de la **consommation pour chaque jour**.

5. Récupération de la Production des Derniers 30 Jours

Cette méthode **retourne la production quotidienne d'énergie solaire et éolienne** pour les **30 derniers jours**.

```
public function recupererProduction30Jours() {
    /*
    QUI: Vergeylen Anthony
    QUAND: 18-12-2024
    QUOI: Retourne la production d'énergie des 30 derniers jours

    Arguments: aucun
    Return: array
    */
    $stmt = $this->pdo->query("
        SELECT DATE(date_mesure) as jour,
            SUM(CASE WHEN type_energie = 1 THEN valeur ELSE 0 END) as solaire,
            SUM(CASE WHEN type_energie = 2 THEN valeur ELSE 0 END) as eolienne
        FROM capteurs_energie
        WHERE date_mesure >= NOW() - INTERVAL 30 DAY
        GROUP BY DATE(date_mesure)
        ORDER BY DATE(date_mesure) ASC
    ");
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
```


La méthode **utilise des clauses CASE** pour différencier les productions solaire et éolienne dans une seule requête.

6. Récupération des Alertes et Incidents

Cette méthode **récupère les alertes de surcharge et les incidents des 3 derniers jours**, limités aux quatre entrées les plus récentes.

```
public function recupererAlertesProduction() {
    /*
    QUI: Vergeylen Anthony
    QUAND: 18-12-2024
    QUOI: Retourne les alertes de surcharge des 3 derniers jours et les incidents

    Arguments: aucun
    Return: array
    */
    $stmtAlertes = $this->pdo->query("
        SELECT *
        FROM alertes_surcharge
        WHERE date_signalement >= NOW() - INTERVAL 3 DAY
        ORDER BY date_signalement DESC
        LIMIT 4
    ");
    $alertes = $stmtAlertes->fetchAll(PDO::FETCH_ASSOC);

    $stmtIncidents = $this->pdo->query("
        SELECT *
        FROM incidents
        WHERE date_creation >= NOW() - INTERVAL 3 DAY
        ORDER BY date_creation DESC
        LIMIT 4
    ");
    $incidents = $stmtIncidents->fetchAll(PDO::FETCH_ASSOC);

    return [
        'alertes' => $alertes,
        'incidents' => $incidents
    ];
}
```

La méthode exécute **deux requêtes SQL** :

- Une pour récupérer les alertes.
- Une autre pour récupérer les incidents.

7. Tests Unitaires

Des tests unitaires ont été réalisés pour vérifier les points suivants :

- **Le message de bienvenue s'adapte correctement aux informations utilisateur.**
- **Les données de production et de consommation sont calculées et agrégées correctement.**
- **Les alertes et incidents sont récupérés dans les délais et limites spécifiés.**
- **Les données des 30 derniers jours sont regroupées par jour.**

Ces tests ont été réalisés pour **voir le résultat attendu, voir si c'est correcte** et **aider au debug** et à la **correction** et la **protections des méthodes de l'app**.

Branches Github

Voici un visuel de nos branches Github.

https://github.com/AnthonyCodeDev/SmartCity_CDCGR10

1. **main** : Il s'agit de la branche principale du projet.
2. **GestionAutomatiqueDonnee** : Une branche dédiée à la gestion automatisée des données.
Cela inclus des scripts pour automatiser les tâches liées aux données, comme **detect_sensor_overload.py** ou **donneeCapteur.py**.
3. **Ressources** : Cette branche contient des fichiers de ressources, des documents, comme des configurations windows, linux, switch, base de données, etc.
4. **SiteWeb** : Une branche dédiée au développement du site web de SmartCity Énergie.

Conclusion

Cette documentation technique offre une vue d'ensemble claire et détaillée des concepts, processus, et fonctionnalités clés développés dans le cadre du projet. Elle permet de comprendre les mécanismes, tout en assurant la complétude et la finalité du code !

En adoptant une approche structurée et explicative, ce document vise à faciliter la compréhension, l'utilisation, et l'évolution du projet.

Enfin, ce travail de documentation souligne l'importance d'une méthodologie rigoureuse et d'une attention particulière aux détails, contribuant ainsi à la robustesse et à la pérennité du projet !

Test effectués

Tests unitaires Authentification :

https://github.com/AnthonyCodeDev/SmartCity_CDCGR10/blob/SiteWeb/controleur/AuthControleur.php

Tests unitaires Capteur :

https://github.com/AnthonyCodeDev/SmartCity_CDCGR10/blob/SiteWeb/controleur/CapteurControleur.php

https://github.com/AnthonyCodeDev/SmartCity_CDCGR10/blob/SiteWeb/controleur/GenererRapportControleur.php

Tests unitaires Générer rapport :

https://github.com/AnthonyCodeDev/SmartCity_CDCGR10/blob/SiteWeb/controleur/GenererRapportControleur.php

Tests unitaires Message de Bienvenu et récupération données statistiques :

https://github.com/AnthonyCodeDev/SmartCity_CDCGR10/blob/SiteWeb/controleur/HomeControleur.php

Tests unitaires Incidents :

https://github.com/AnthonyCodeDev/SmartCity_CDCGR10/blob/SiteWeb/controleur/IncidentsControleur.php