

P2022 : SunShare
AHAMADA Ayaad

Dossier technique du projet - partie individuelle

Table des matières

1 - SITUATION DANS LE PROJET	3
1.1 - RAPPEL DES TÂCHES PROFESSIONNELLES À RÉALISER	3
1.2 - PRÉSENTATION DE LA PARTIE PERSONNELLE	4
1.2.1 - Introduction	4
1.2.2 - Synoptique de la réalisation	5
1.2.3 - Diagramme de déploiement	6
1.2.4 - Aperçu de l'application web	7
1.2.5 - Constitution d'une page web HTML	9
1.2.6 - Méthodes HTTP	10
1.2.7 - Technologies logicielles utilisées	11
1.2.8 - Diagramme de cas d'utilisation	12
1.2.9 - Schéma de la base de donnée	12
2 - RÉALISATION DE LA TÂCHE « AVOIR DES DONNÉES INSTANTANÉES »	14
2.1 - CAS D'UTILISATION	14
2.2 - PROCÉDURE DE TEST	15
2.2.1 - Rapport d'exécution	15
2.3 - UN RÉSUMÉ ÉNERGÉTIQUE	16
2.4 - CONCEPTION DÉTAILLÉE	16
2.4.1 - Requêtes pour la journée d'hier	16
2.4.2 - Requêtes pour le mois passé	16
2.4.3 - Requête pour l'année passé	16
2.5 - PROCÉDURE DE TEST	17
2.5.1 - Rapport d'exécution	18
3 - RÉALISATION DE LA TÂCHE « AFFICHER DES GRAPHIQUES »	19
3.1 - CAS D'UTILISATION	19
3.2 - CONCEPTION DÉTAILLÉE	19
3.2.1 - Récupération de données sur la base de données	20
3.3 - TESTS UNITAIRES	23
3.3.1 - Procédure de test	23
3.3.2 - Rapport d'exécution	26
4 - RÉALISATION DE LA TÂCHE « AFFICHER UN HISTORIQUE »	26
4.1 - CAS D'UTILISATION	26
4.2 - PROCÉDURE DE TEST	26
4.2.1 - Affichage historique journalière 2022-04-01 au 2022-04-07	28
4.2.2 - Affichage historique hebdomadaire 2022-05-01 au 2022-06-04	28

4.2.3 - Affichage historique mensuelle 2022-04-07 au 2022-11-07.....	29
4.2.4 - Affichage historique annuelle 2019-2022.....	29
4.2.5 - Rapport d'exécution.....	29

5 - BILAN DE LA RÉALISATION PERSONNELLE.....	30
5.1 - STATUT DES FONCTIONS À MA CHARGE.....	30
5.2 - AMÉLIORATION ENVISAGEABLE.....	30
5.3 - CONCLUSION.....	30
5.3.1 - Point positifs.....	30
5.3.2 - Point négatifs.....	30

1 - Situation dans le projet

1.1 - Rappel des tâches professionnelles à réaliser

Tâches	Description
FP1 :	Installer et utiliser la base de données Mise en place d'un SGBD (Système de gestion de base de données) pour le stockage de nos données provenant des compteurs. Concevoir les tables de la base de données
FP2 :	Afficher des données instantanées Suivre en temps réel sur une page web : la production locale, la consommation et l'injection dans le réseau Enedis
FP3 :	Afficher un historique de données Voir sur une période définie au préalable par un utilisateur, des données qui ont été enregistrées dans la base de données. <ul style="list-style-type: none">• Journalière, hebdomadaire• Mensuelle• Annuelle
FP4 :	Afficher des graphiques Sur une période choisie, on doit voir une représentation graphique de notre donnée. <ul style="list-style-type: none">• Journalière, hebdomadaire• Mensuelle• Annuelle
FS1 :	Télécharger les graphiques en PDF Cette fonction permettra à l'utilisateur s'il souhaite d'avoir ces graphiques en format PDF (une fonction qui n'est pas mentionnée dans le cahier des charges).
FS2 :	Mettre les données dans un fichier format CSV Cette fonction permettra à l'utilisateur s'il souhaite d'avoir son historique en format CSV (une fonction qui n'est pas mentionnée dans le cahier des charges). Un fichier CSV (comma separated values) est le fichier de base des données recueillies - sans formatage particulier. Chaque champ est séparé par une virgule.

1.2 - Présentation de la partie personnelle

1.2.1 - Introduction

L'objectif de ma partie personnelle est de réaliser une Interface Homme Machine simple et ergonomique, qui permettra à un utilisateur qui a peu de connaissances en informatique de consulter chez lui, sa consommation, sa production et injection dans le réseau électrique (chiffre, historique, graphique ...), que le compteur Linky soit dans la maison ou distant sur la voie publique.

Cette interface de visualisation des indicateurs va permettre d'afficher des données instantanées, sous forme historique ou de graphiques.

La réalisation du projet s'est déroulée en plusieurs phases jusqu'à présent :



I. Découverte du projet

- Lecture du cahier des charges
- Prise en main des outils de gestion de projet (Trello et Gitlab)
- Première prise en main du matériel (recherche)

II. Sprint 1

- Choix du SGBD (système gestionnaire de base de données)
- **Prise en main du matériel (Raspberry pi)**
 - Mise en place du système d'exploitation sur ma Rpi
 - installation et configuration (SGBD, PhpMyAdmin et serveur Apache)
- Analyse SysML (Diagramme séquence)
- Première proposition de schéma base de donnée

III. Sprint 2

- Schéma base de données définitif
- Création des tables sur la BdD Raspberry Pi
- Analyse SysML (Diagramme de navigation entre page)
- Mise en place de l'API (application programming interface) avec Nicolas LHOMMEAU
- Prise en main du framework Bootstrap

IV. Sprint 3

- Afficher des données instantanées sur l'application web.
- Affichage des données sous forme d'historique et graphique.

1.2.2 - Synoptique de la réalisation

Aperçue générale du projet, avec comme acteurs un panneaux solaire/éolienne, les compteurs (SDM120D et Linky), une Raspberry, un ordinateur et un smartphone).

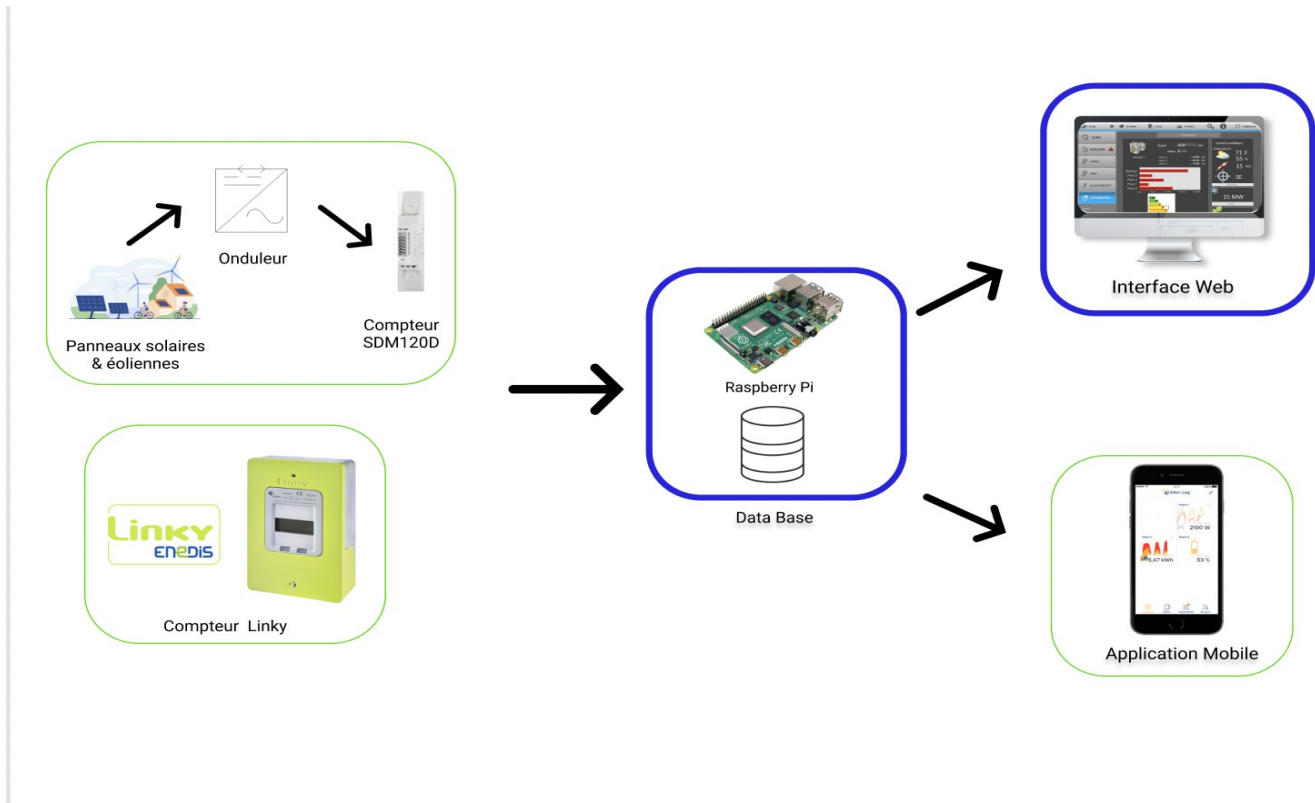


Figure 1: Synoptique du système

Comme indiqué sur la figure ci-dessus, on a dans un premier temps une collecte des données provenant des compteurs (SDM120D et Linky).

Ensuite, grâce à la Raspberry Pi, qui héberge une base de données, les indicateurs mesurés (production, consommation et injection) pourront être stockés pendant un certain temps.

Et pour finir, on affiche les mesures sur un smartphone ou sur un navigateur web.

Dans ce projet j'interviens sur la partie colorée en violet, préparation de la base de données pour que l'étudiant 1 **Melvin COURJAUD** puisse stocker les données mais également, pour l'affichage de ses données sur ma page web ou dans l'application smartphone de **Nicolas LHOMMEAU**.

1.2.3 - Diagramme de déploiement

On retrouve ici, un diagrammes structuraux, qui décrit le déploiement physique et la manière dont les composants du système sont répartis ainsi que leurs relations.

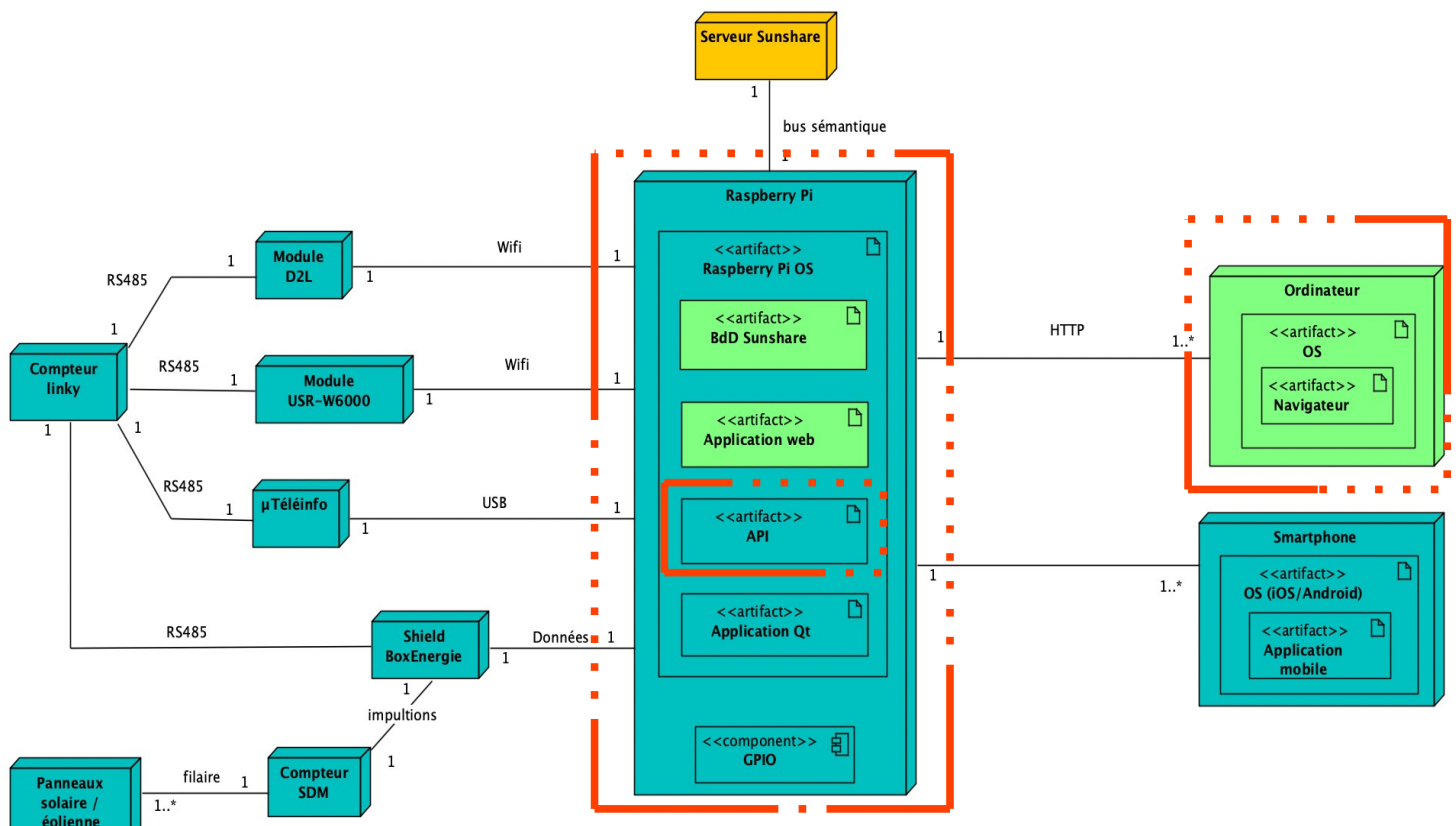


Figure 2: Diagramme déploiement ma partie

Comme indiqué ci-dessus, j'interviens que sur les parties colorées en **vert**, c'est-à-dire la gestion de la base de données locale (Conception des tables de la base de données) via **PhpMyAdmin***, mais également l'affichage des données sur un navigateur web.

PhpMyAdmin* : est un outil logiciel libre écrit en PHP, destiné à gérer l'administration de MySQL/MariaDB sur le Web. Une interface pratique qui permet d'exécuter, très facilement et sans grandes connaissances en bases de données, des requêtes comme les créations de table de données, insertions, mises à jour, suppressions et modifications de structure de la base de données.

J'interviens également sur la création de l'API (Application Programming Interface). Ce dernier permet d'exposer les contenus en provenance de la base de données.

C'est-à-dire, permettre à des systèmes tiers d'accéder à des fonctionnalités et ces contenus. Une interface de programmation d'application a donc pour but de faire dialoguer plusieurs appareils (par exemple ici : smartphone et la base de données).

Dans ma partie, on retrouve

- une **carte Raspberry PI** qui héberge :
 - ✓ Une **base de données locale** «SunshareBdD»,

✓ Une **application web**

Pour le smartphone de l'étudiant 3 (**Nicolas LHOMMEAU**), une **API** (Application Programming Interface) a été implémenter pour la communication entre la base de données et le smartphone.

L'étudiant 1 a également implémenter son **application Qt** dans la RPi, pour l'écriture et la rétention de données qui seront dans la base de données.

- Un **ordinateur** ou tout appareil chose pouvant utilisant un navigateur web (connecté sur le même réseau que la Raspberry), pour la visualisation de la production, consommation et injection locale.

1.2.4 - Aperçu de l'application web

L'application est composée de trois pages :

- La **page d'accueil (figure 1)** : On verra sur cette page le résumé de notre énergie (temps réel, journée d'hier, mois passé et année passée).

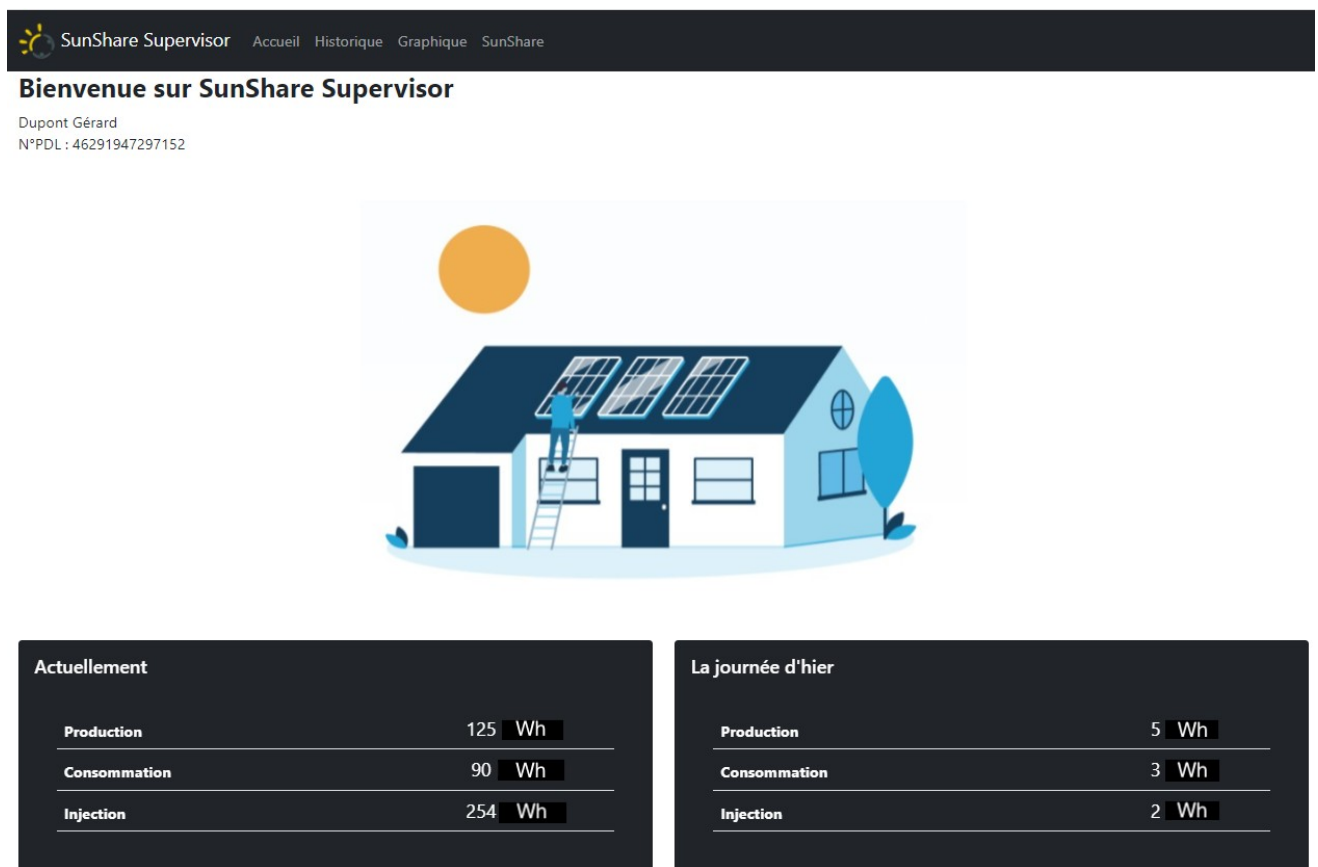


Figure 3: La page d'accueil

- La **page historique (figure 4)**. Cette page permettra à l'utilisateur de voir sur une longue période les énergies enregistrées dans la base de données (énergie produite, consommée et injectée) dans le passé. En saisissant une date de début (ex : 2016) et une date de fin (ex : 2018) mais également une période (journalière, hebdomadaire, mensuelle et annuelle.).

Figure 4: La page historique

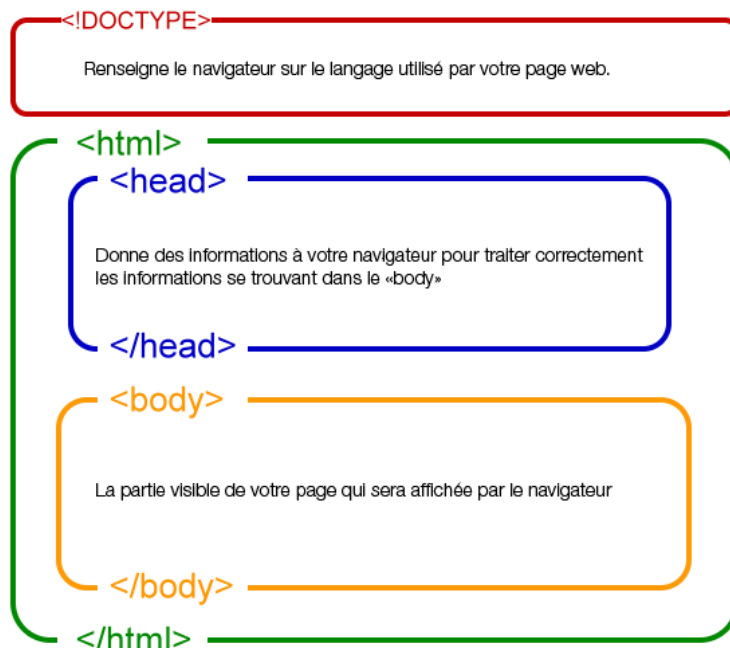
Correction unité : les données affichés sur la page sont en **Wh** et non en **kwh**.

- La **page graphique**. Sur cette page on aura une représentation en graphique de l'énergie journalière, hebdomadaire, mensuelle ou annuelle. J'affiche les graphiques deux manières différentes en forme de barre et de ligne.

Figure 5: La page graphique

1.2.5 - Constitution d'une page web HTML

L'application web est découpée selon le schéma suivant :



Pour ma page web, à l'intérieur de la balise Head, on va retrouver par exemple principalement l'inclusion des différentes bibliothèques (Bootstrap, ChartJS etc...) que j'utilise dans ce projet.

Et à l'intérieur du bloc body, on va retrouver par exemple :

- Une balise nav (navigation) (**<nav>...</nav>**), représente une section d'une page ayant des liens vers d'autres pages ou des fragments de cette page. Autrement dit, c'est une section destinée à la navigation dans un document (avec des menus etc.).
- Des balises div (division) (**<div>...</div>**), je l'utilise pour regrouper le contenu de la page afin qu'il puisse être facilement stylé à l'aide des attributs **class*** ou **id*** (*identifiant unique*). Les divs n'ont aucun effet sur le contenu ou la mise en page tant qu'il n'est pas mis en forme d'une manière quelconque à l'aide de **CSS** (Cascading Style Sheets, utilisé pour **définir l'aspect d'un site**).

Attribut class et id* : Ces deux attributs vont être principalement utilisés dans un but de mise en forme : ils vont servir à appliquer des styles CSS aux éléments qui vont les contenir.

- On retrouve 2 types de div (container et fluide) :
 - Un élément avec la classe **.container-fluid**, va avoir une largeur de 100%.
 - Un élément avec la classe **.container**, va être basé sur la taille de largeur de l'écran, cela veut dire que l'élément va s'adapter en fonction de l'écran de l'utilisateur, Bootstrap les classe en 5 classes :

Class Prefix	Description
.col-	Pour les très petits appareils (Extra Small), les téléphones.
.col-sm-	Pour les petits équipements (Small), les tablettes.
.col-md-	Pour les moyens équipements (Medium), comme les petits ordinateurs portables dont la largeur égale ou supérieure 768px .
.col-lg-	Pour les grands équipements (Large) dont la largeur est supérieur ou égale à 992px .
.col-xl-	Pour les gigantesques équipements (Extra Large), comme les ordinateurs portables et ordinateurs de bureau dont la largeur est supérieur ou égale 1200px .

- Balise script `<script>...</script>`, utilisé pour intégrer ou faire référence à un script exécutable.

1.2.6 - Méthodes HTTP

Grâce au protocole **HTTP** (Hypertext Transfer Protocol), qui assure la communication entre les clients et le serveur, j'affiche les indicateurs (énergie produite, soutirée et injectée) sur ma page. Cela fonctionne comme une demande demande et une réponse.

Dans le protocole HTTP, une méthode est une commande spécifiant un type de requête, c'est-à-dire qu'elle demande au serveur d'effectuer une action.

Les deux méthodes utilisées couramment en langage web, pour la communication avec le serveur :

METHODE	Rôle
GET	<p>La méthode GET demande une représentation de la ressource spécifiée. Les requêtes GET doivent uniquement être utilisées afin de récupérer des données.</p> <p>Dans un formulaire HTML, elle est spécifiée ainsi:</p> <pre><form method="get" action="page.html"> </form></pre> <p>Avec cette méthode, les données du formulaire seront encodées dans une URL. Celle-ci est composée du nom de la page ou du script à charger avec les données de formulaire empaquetée dans une chaîne.</p> <p>Les données sont séparées de l'adresse de la page par le code <code>?</code> et entre elles par le code <code>&</code>.</p> <p>Comme par exemple : <code>https://127.0.0.1/page.html?Donnee&AutreDonnee</code></p>
POST	<p>La méthode POST est utilisée pour envoyer une entité vers la ressource indiquée. Cela entraîne généralement un changement d'état ou des effets de bord sur le serveur.</p> <p>Dans un formulaire HTML, elle est spécifiée ainsi:</p> <pre><form method="post" action="page.php"> </form></pre> <p>Les données du formulaire n'apparaissent pas dans l'URL</p>







Dans ma partie, j'utilise la méthode **POST**, pour la demande de données au près de la base de données.

Tout simplement parce que la méthode (**POST**) est à favoriser lorsqu'un utilisateur doit soumettre des données ou des fichiers au serveur, par exemple pour remplir des formulaires ou télécharger des photos.

Contrairement à la méthode **GET**, qui elle est bien adapté pour les paramétrages d'un site Web (filtres, tri, saisies de recherche, etc.)

1.2.7 - Technologies logicielles utilisées

Dans ma partie, on ne pas réellement imposait de contrainte de réalisation, c'est-à-dire que je suis libre de choisir le framework ou la bibliothèque que je souhaitais utiliser dans la tâche qui m'a été confier

Technologies utilisé :	Description :
	Figma , est une plateforme collaborative qui permet de faire du prototypage. Elle permet de concevoir des design systems pour faciliter la création de sites web et d'applications mobiles.
	Bootstrap , est une collection d'outils utiles à la création du design de sites et d'applications web (graphisme, animation et interactions avec la page dans le navigateur, etc.). C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option. Et il a été conçu par deux développeurs, Mark Otto et Jacob Thornton .
	Chart.js , est une bibliothèque JavaScript open source conçu pour représenter des données sous forme de graphes statistiques. Elle utilise les fonctionnalités HTML5 comme les Canvas (composant du langage Web HTML qui permet d'effectuer des rendus dynamiques.) et gère l'aspect responsive (graphique conçu et développé de façon à pouvoir s'adapter à toutes les résolutions d'écran).
	Gitlab , Plateforme de développement collaborative open-source. J'utilise gitlab, pour la sauvegarde et la partage de mon travail a toutes l'équipe.
	Trello , un outil de gestion de projet gratuite, qui nous permet d'organiser notre projet sous forme de tableaux, eux-mêmes composés de listes en colonnes, qui répertorient des tâches sous formes de cartes.
	Visual Paradigm , logiciel de modélisation UML (Unified Modeling Language)/SysML (Systems Modeling Language).

1.2.8 - Diagramme de cas d'utilisation

On retrouve ici, un diagramme qui représente les différentes façons dont l'utilisateur interagira avec le système. Par interface web, l'utilisateur verra l'affichage des indicateurs en instantanée, sous format historique ou de graphique.

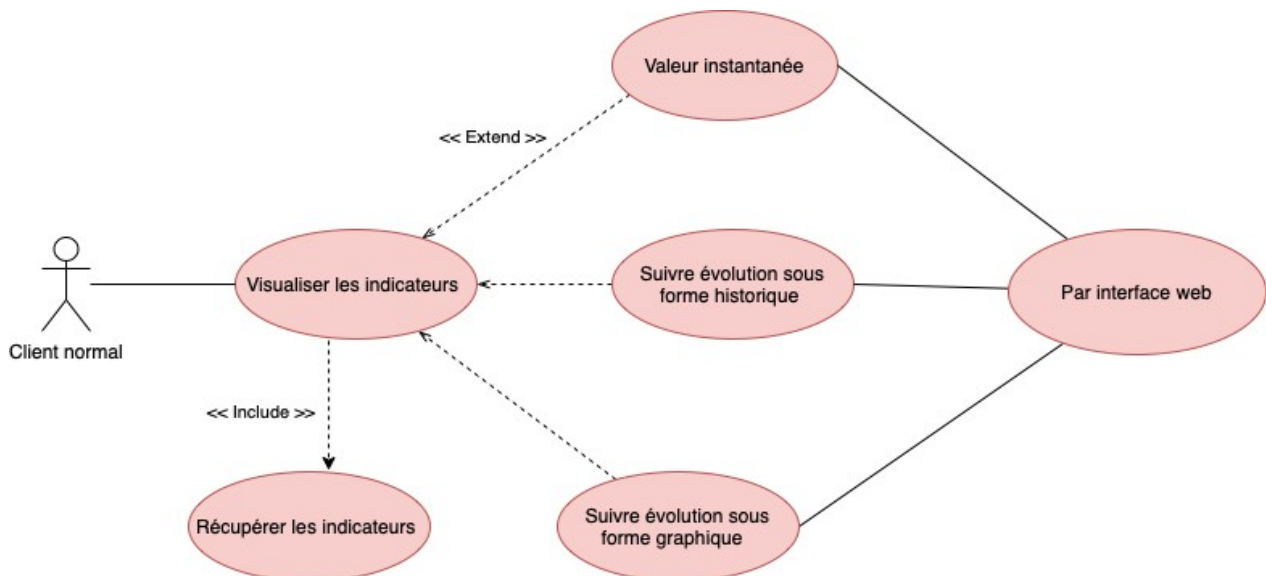


Figure 6: Diagramme cas d'utilisation de ma partie

1.2.9 - Schéma de la base de donnée

Afin d'afficher les indicateurs sur ma page web, les données sont stockées sur une base. Pour l'unanimité de tous les membres du groupe, 2 schémas ont été proposés pour le stockage des indicateurs dans la base de données.

En premier, j'avais décidé de créer seulement 2 tables (**Energies** et **Utilisateur**), mais cette solution ne pouvait durer dans le temps. La table **Energies** qui allait contenir l'énergie capturée en temps réel provenant des compteurs et la table **utilisateur** qui allait contenir les informations d'identification (Nom, Prénom et numéro PDL) de l'utilisateur.

En effet, arrivera un certain temps, le processeur de la Raspberry ne sera plus apte à traiter correctement toutes ses données, car il y aura beaucoup de données à traiter et donc le temps de recherche sera assez long.

Le premier schéma de la base de données se présentait ainsi :

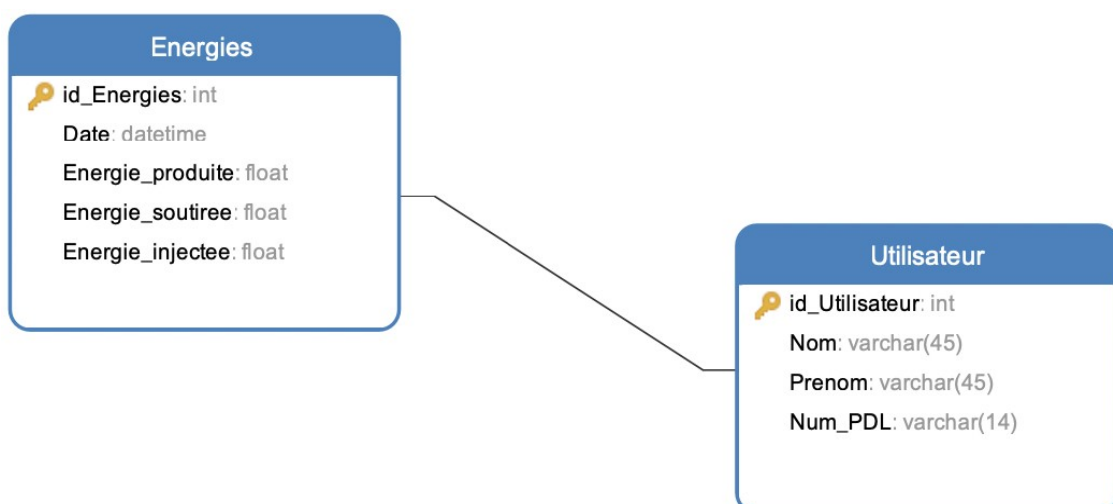


Figure 7: Le premier schéma BdD

En ce qui concerne les tables :

- La table **Utilisateur** allait contenir tout ce qui va permettre d'identifier tout personne qui utilisera notre système, comme le **Nom** et le **Prénom** de type Varchar et d'une taille maximal de 45.
Elle contient également le numéro **PDL** (*Point De Livraison*), un numéro qui permettra au fournisseur d'électricité d'identifier son compteur électrique associé à son logement, d'établir un contrat à son nom, de procéder à la mise en service de l'électricité à son domicile et enfin le faire facturer les consommations enregistrées par le compteur. Et donc, c'est un numéro unique composé de 14 chiffres et qui sera du type Varchar.
- La table **Energies** allait contenir toutes les mesures que les compteurs vont envoyer à la base de données (Compteur Linky et Compteur SD120D) ainsi que la date (**Date**) à laquelle les mesurer on était faite. Ces mesures incluent donc l'énergie produite (**energie_produite**), l'énergie consommée (**energie_soutiree**), l'énergie injectée (**energie_injectee**).

Le deuxième schéma de la base de données se présente ainsi :

Et donc, ensemble on a décidé de partager la table en plusieurs parties.

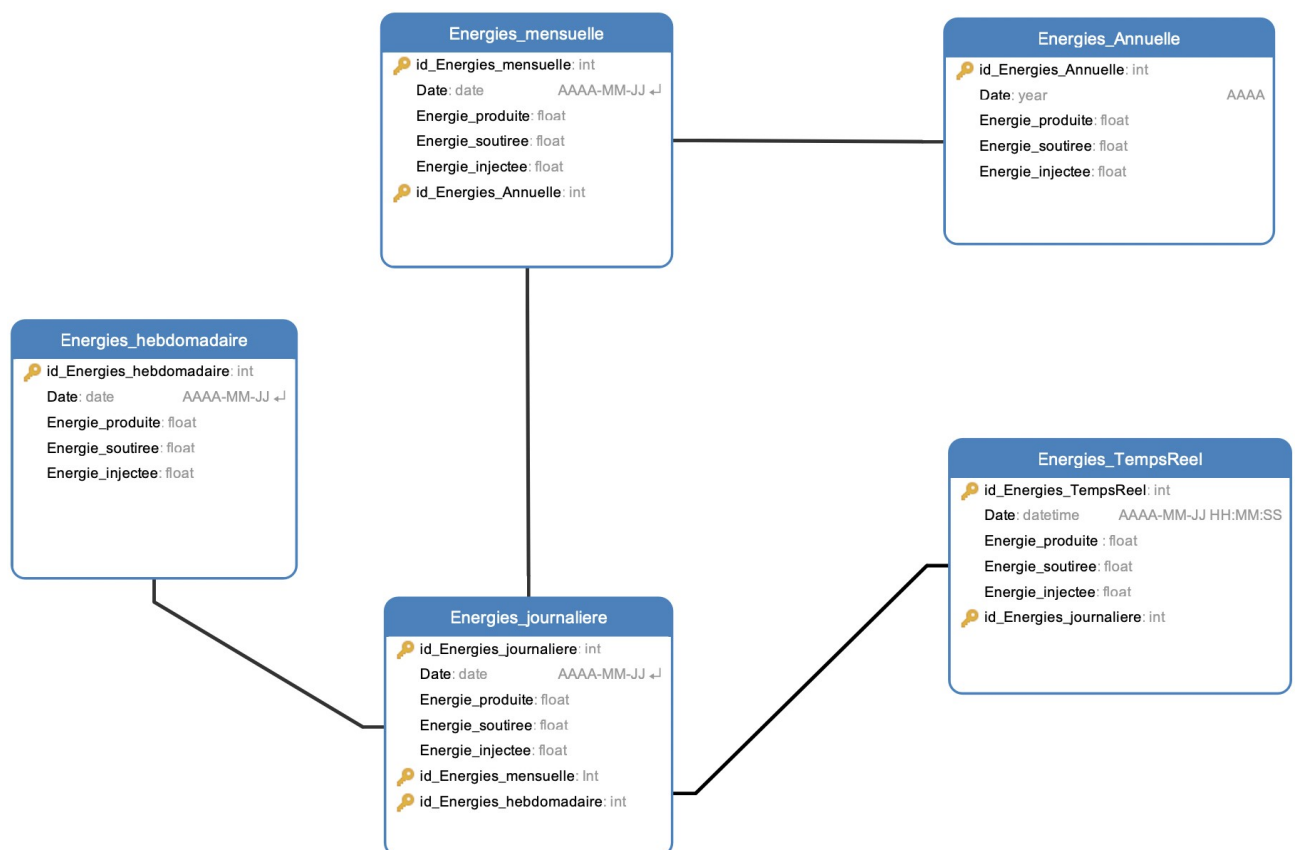


Figure 8: Le deuxième schéma de la base de données

La base de données actuelle est composée de 5 tables :

- La table **Energies_Instantanee**, qui contient les énergies que les compteurs (Compteur Linky et Compteur SD120D) vont envoyer à la base de données, mesurées en temps réel ainsi que la date (**Date**) à laquelle les mesures ont été faites. Ces mesures comprennent donc l'énergie produite (**energie_produite**), l'énergie consommée (**energie_soutiree**), l'énergie injectée (**energie_injectee**).
- La table **Energies_journaliere** qui contiendra les énergies (**Energie_produite**, **Energie_soutiree** et **Energie_injectee**) moyennes d'une journée. Ces valeurs seront calculées chaque jour et permettent de gagner du temps pour la consultation journalière.

- La table **Energies_hebdomadaire** va contenir les énergies (**Energie_produite**, **Energie_soutiree** et **Energie_injectee**) moyennes d'une semaine. Les données de chaque jour durant 7 jours, calculées en valeur moyenne.
- La table **Energies_mensuelle** va contenir les énergies (**Energie_produite**, **Energie_soutiree** et **Energie_injectee**) moyennes de chaque mois

La table **Energies_annuelle** contiendra les énergies (**Energie_produite**, **Energie_soutiree** et **Energie_injectee**) moyennes d'une année

Sur ce schéma, la table **utilisateur** n'a pas été retenu pour la simple et bonne raison que ce système sera utilisé par seulement un utilisateur, donc cela n'était pas nécessaire de le faire figurer sur la base de données. Les informations contenues dans la table utilisateur seront mises dans un fichier texte php.

Et donc, en fonction de ce que je veux afficher sur ma page web, je change le nom de la table dans ma requête SQL (Structured Query Language), ainsi on accède plus facilement et plus rapidement aux données.

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> Energies	★ Parcourir Structure Rechercher Insérer Vider Supprimer	5	InnoDB	utf8_general_ci	32,0 kio	-
<input type="checkbox"/> Energies_Annuelle	★ Parcourir Structure Rechercher Insérer Vider Supprimer	14	InnoDB	utf8_general_ci	32,0 kio	-
<input type="checkbox"/> Energies_Hebdomadaire	★ Parcourir Structure Rechercher Insérer Vider Supprimer	13	InnoDB	utf8_general_ci	16,0 kio	-
<input type="checkbox"/> Energies_Journaliere	★ Parcourir Structure Rechercher Insérer Vider Supprimer	7	InnoDB	utf8_general_ci	32,0 kio	-
<input type="checkbox"/> Energies_Mensuelle	★ Parcourir Structure Rechercher Insérer Vider Supprimer	9	InnoDB	utf8_general_ci	16,0 kio	-
<input type="checkbox"/> Energies_TempsReel	★ Parcourir Structure Rechercher Insérer Vider Supprimer	57	InnoDB	utf8_general_ci	16,0 kio	-
<input type="checkbox"/> utilisateurs	★ Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	utf8_general_ci	32,0 kio	-
7 tables	Somme	108	InnoDB	utf8_general_ci	176,0 kio	0 o

Figure 9: Notre BdD dans PhpMyAdmin

Sur la **figure 9**, a tables **Energies** et **Utilisateurs** ne sont plus présents dans la base donnée.

2 - Réalisation de la tâche « avoir des données instantanées »

2.1 - Cas d'utilisation

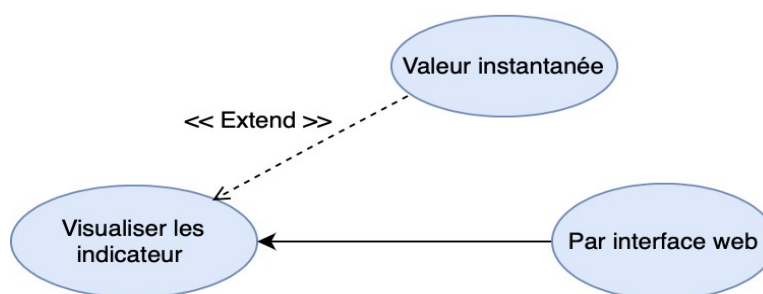


Figure 10: Visualiser les indicateurs via interface web

2.2 - Procédure de test

L'objectif de ce test, est de tester la réception en temps réel des énergies présentes dans la base de données, sur la page web de l'utilisateur.

Id.	Méthode testée	Procédure de test
	Description Sommaire	Résultats attendus
U1.0	Ouvrir la page d'accueil	sur la barre de recherche, tapez @ip de la Rpi: http://10.0.122.166
		Accéder à la page d'accueil
U1.1	Ouvrir PhpMyAdmin	Ouvrir un autre onglet
		Sur la barre de recherche, tapez : http://10.0.122.166/phpmyadmin
U1.2	Se connecter PhpMyAdmin	Afficher la page de connexion de PhpMyAdmin
		Utilisateur : admin
U1.3	Ajouter des valeurs dans la BdD	mot de passe : admin
		Accéder à PhpMyAdmin
U1.4	Afficher l'énergie instantanée (production, consommation et injection) sur la page web.	Aller dans l'onglet « SQL » et Taper une requête qui permet d'ajouter des valeurs dans la base de données
		<p>INSERT INTO Energies_TempsReel (idEnergies_TempsReel date, energie_produite, energie_soutiree, energie_injectee)</p> <p>VALUES (NULL, CURRENT_TIME, 1, 0,0) ;</p> <ul style="list-style-type: none">Requête fonctionnelle (test sur phpmyadmin)Les valeurs ont été ajouter
U1.4	Afficher l'énergie instantanée (production, consommation et injection) sur la page web.	La requête demande à avoir les données qui ont été ajouter.
		Affichage du résultat.
<div><div>Actuellement</div><div><div><div>Production</div><div>0 kwh</div></div><div><div>Consommation</div><div>583 kwh</div></div><div><div>Injection</div><div>0 kwh</div></div></div></div>		

2.2.1 - Rapport d'exécution

Id.	OK	!OK	Observations
U1.0	*		Résultat OK
U1.1	*		Résultat OK
U1.2	*		Résultat OK
U1.3	*		Résultat OK
U1.4	*		Résultat OK

2.3 - Un résumer énergétique

En plus, de l'énergie instantanées, j'ai ajouté un résumer énergétique (journalière, hebdomadaire et annuelle). La procédure de teste est la même que l'affichage instantanée.

2.4 - Conception détaillée

2.4.1 - Requêtes pour la journée d'hier

Pour avoir l'énergie de la journée précédente, par ordre décroissant je prends à chaque fois l'énergie qui est inférieure à la date actuelle de la table **Energies_Journaliere** en faisant une comparaison de date dans la base de données.

```
$Requete = "SELECT date, energie_soutiree, energie_injectee, energie_produite  
FROM Energies_Journaliere  
WHERE date < CURRENT_DATE  
ORDER BY date DESC LIMIT 1;" ;
```

2.4.2 - Requêtes pour le mois passé

Pour avoir l'énergie moyenne du mois dernier, par ordre décroissant je prends à chaque fois l'énergie qui est inférieure à la date actuelle de la table **Energies_Mensuelle** en faisant une comparaison de date dans la base de données.

```
$Requete = "SELECT date, energie_soutiree, energie_injectee, energie_produite  
FROM Energies_Mensuelle  
WHERE date < CURRENT_DATE  
ORDER BY date DESC LIMIT 1;" ;
```

2.4.3 - Requête pour l'année passé

Pour avoir l'énergie moyenne de l'an dernier, par ordre décroissant je prends à chaque fois l'énergie qui est inférieure à la date actuelle de la table **Energies_Annuelle** en faisant une comparaison de date dans la base de données.

```
$Requete = "SELECT date, energie_soutiree, energie_injectee, energie_produite  
FROM Energies_Annuelle  
WHERE date < CURRENT_DATE  
ORDER BY date DESC LIMIT 1;" ;
```


2.5 - Procédure de test

Le but de ce test est de tester si nous recevons bien les énergies moyennes (Energie_produite, Energie_soutiree et Energie_injectee) pour la journée d'hier, le mois dernier, ainsi que de l'an dernier.

Id.	Méthode testée Description Sommaire	Procédure de test				
		Résultats attendus				
U2.0	Ouvrir la page d'accueil	sur la barre de recherche, tapez @ip de la Rpi: http://10.0.122.166				
		Accéder à la page d'accueil				
U2.1	Ouvrir PhpMyAdmin	Ouvrir un autre onglet				
		Sur la barre de recherche, tapez : http://10.0.122.166/phpmyadmin				
		Afficher la page de connexion				
U2.2	Se connecter PhpMyAdmin	Utilisateur : admin mot de passe : admin				
		Accéder à phpmyadmin				
U2.3	Sur PhMyAdmin ajouter des valeurs (avec comme date inférieure et supérieure à la date de maintenant) dans la BdD.	Aller dans l'onglet « SQL » et Taper une requête qui permet d'ajouter des valeurs dans la base de données.				
		INSERT INTO Energies_TempsReel (idEnergies_TempsReel date, energie_produite, energie_soutiree, energie_injectee) VALUES (NULL, '2022-05-12', 1, 0,0) ;				
		<ul style="list-style-type: none">Requête fonctionnelle (test sur phpmyadmin)Ajout de valeurs				
U2.4	Afficher sur la page web l'énergie journalière d'hier, du mois passé, et de l'année passée.	Demande à avoir les données qui sont inférieur à la date actuelle.				
		Afficher un résultat. <div><div>La journée d'hier</div><table><tr><td>Production</td><td>10 kwh</td></tr><tr><td>Consommation</td><td>5 kwh</td></tr><tr><td>Injection</td><td>5 kwh</td></tr></table></div>	Production	10 kwh	Consommation	5 kwh
Production	10 kwh					
Consommation	5 kwh					
Injection	5 kwh					

Le résultat final après la récupération des données voulues :



Figure 11: résumer énergétique

2.5.1 - Rapport d'exécution

Id.	OK	!OK	Observations
U2.0	*		Résultat OK
U2.1	*		Résultat OK
U2.2	*		Résultat OK
U2.3	*		Résultat OK
U2.4	*		Résultat OK

3 - Réalisation de la tâche « afficher des graphiques »

3.1 - Cas d'utilisation

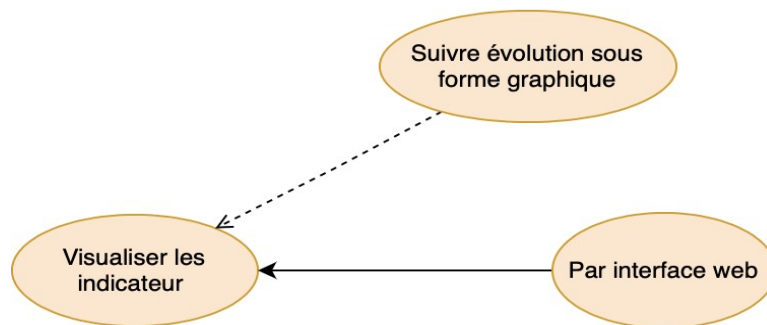


Figure 12: Visualiser les indicateurs via interface web

3.2 - Conception détaillée

Sur cette tâche, je dois afficher un graphique en utilisant des données provenant de la base de données, pour que l'utilisateur du site web puisse suivre visuellement son évolution énergétique.

Afin de mettre en pratique cette tâche, j'ai choisi la librairie « chartjs » qui est non seulement OpenSource, mais également en raison du large choix options de personnalisation qu'il propose. En effet, tous les graphiques qu'offre cette librairie, peuvent être animés et sont entièrement personnalisables.. Pour un utilisateur qui souhaite avoir un site ergonomique, c'est un excellent choix.

De plus Chart.js n'a besoin d'aucune autre bibliothèque pour fonctionner, est assez léger (environ 281 ko zippé) et offre de nombreuses options de personnalisation.

3.2.1 - Récupération de données sur la base de données

Pour afficher mes graphiques, j'ai utilisé un **switch case** pour choisir une requête SQL. J' affiche les graphiques en fonction du choix de la date et de la période (journalière, hebdomadaire etc...).

```
//Selon le choix de l'utilisateur, une requête va être choisie
if(isset($_POST['Select']))
{
    switch ($Select)
    {
        //choix 1 : journalier (pour consulter l'évolution journalière)
        Case 'journalier' :
            $Requete = "SELECT date, energie_soutiree, energie_injectee, energie_produite FROM Energies_Journaliere
                        WHERE date BETWEEN \"".$DateD."\" AND \"".$DateF."\" ORDER BY date DESC LIMIT 7 ";
            break ;
        //choix 2 : hebdomadaire (pour consulter l'évolution hebdomadaire)
        Case 'hebdomadaire' :
            $Requete = "SELECT date, energie_soutiree, energie_injectee, energie_produite FROM Energies_Hebdomadaire
                        WHERE date BETWEEN \"".$DateD."\" AND \"".$DateF."\" ORDER BY date DESC LIMIT 7 ";
            break ;
        //choix 3 : mensuel (pour consulter l'évolution mensuelle)
        Case 'mensuelle' :
            $Requete = "SELECT energie_soutiree, energie_injectee, energie_produite, date FROM Energies_Mensuelle
                        WHERE date BETWEEN \"".$DateD."\" AND \"".$DateF."\" ORDER BY date DESC LIMIT 7 ";
            break ;
        //choix 3 : annuel (pour consulter l'évolution annuelle)
        Case 'annuel' :
            $Requete = "SELECT energie_soutiree, energie_injectee, energie_produite, date FROM Energies_Annuelle
                        WHERE date BETWEEN \"".$DateD."\" AND \"".$DateF."\" ORDER BY date DESC LIMIT 7 ";
            break ;
        Default :
            break ;
    }
}
```

Figure 13: choix de requête

Figure 14, avec une boucle **while**, et un **mysqli_fetch_array()** je récupère le résultat obtenu : *energie_produite*, *energie_produite* et *energie_produite*.

Et on fait cela jusqu'à ce que la condition soit fausse, c'est-à-dire jusqu'à ce qu'il n'y ait plus de lignes dans le résultat obtenue via la base de données.

```
while($row=mysqli_fetch_array($Resultat))
{
    //récupérer la colonne energie_produite
    $energie_produite[] = $row["energie_produite"];

    //récupérer la colonne energie_soutiree
    $energie_soutiree[] = $row["energie_soutiree"];

    //récupérer la colonne energie_injectee
    $energie_injectee[] = $row["energie_injectee"];

    //récupérer la colonne energie_injectee
    $date [] = $row["date"];
}
```

Figure 14: Récupération de données sur la base de données

```

<!-- Script pour la création de graphique -->
<script>

    //1er bloc setup

    //Conversion des données (energie_produite, energie_soutiree, energie_produite et date)
    //en format Json pour que les données soient lisible pour javascript
    const energie_produite = <?php echo json_encode($energie_produite); ?>;
    const energie_soutiree = <?php echo json_encode($energie_soutiree); ?>;
    const energie_injectee = <?php echo json_encode($energie_injectee); ?>;
    const date = <?php echo json_encode($date); ?>;

    const data = {
        Code...
    };

    //2ème Bloc Config graph histogramme
    const config = {
        Code...
    };

    // 3ème Bloc de rendu graph histogramme
    const MyChartBar = new Chart( document.getElementById('MyChartBar'),
        Config
    );

    //Autre bloc config graph ligne
    const ConfigLine = {
        Code...
    };

    // Autre bloc de rendu graph ligne
    const MyChartLine = new Chart(document.getElementById('MyChartLine'),
        ConfigLine
    );

```

Figure 15: Script graphique

Sur la **figure 15**, pour pouvoir afficher un graphique j'ai utilisé trois blocs différent :

- Un **bloc setup**, Ce bloc de configuration ou de données comprend toutes les données par défaut que nous allons utiliser dans notre graphique. Il peut s'agir de points de données (data), d'étiquettes (label), de couleurs d'arrière-plan, de couleurs de survol, etc.

Ce bloc de données est la première partie à être lancée. Parce que, les autres blocs dépendent de ce bloc de données.

- Un **bloc config** (configuration), il regroupe toutes les configurations du graphique. ce bloc de configuration comprend donc la référence du bloc de données, les options et le type de graphique (histogramme, donuts...).

Il y a des propriétés pour contrôler le style, les polices, la légende, etc. Il doit être placé avant le bloc render.

À l'intérieur de ce bloc de config, on trouve trois parties principales aussi : le type de graphique, la variable data qui est liée au bloc de configuration et les options.

```

<!-- Script pour la création de graphique -->
<Script>
//2ème Bloc Config graph histogramme
const config={

    type: 'bar',
    data,
    options:{

        responsive : true,
        scales:
        {

            yAxes:
            {

                beginAtZero : true,
                title:
                {

                    display : true,
                    text : 'Energie (Kwh)'

                }

            }

        },

        plugins : {

            legend : {

                position: 'bottom',

            },

        }

    }

};
</Script>

```

Figure 16: Intérieure du bloc config bar

- Un **bloc render** (rendu ou d'initialisation), est toujours le dernier. Sinon, cela donnerait une erreur et pourrait exclure certains blocs de code.

En fonction des autres blocs, il lance ou active le processus de dessin du graphique dans la balise **canvas** (balise HTML utilisé pour dessiner des graphiques sur une page Web) à laquelle il se réfère.

```

<!-- Script pour la création de graphique -->
<script>

// 3ème Bloc de rendu graph histogramme
const MyChartBar = new Chart( document.getElementById('MyChartBar'),
    Config
);

// Bloc de rendu graph ligne
const MyChartLine = new Chart(document.getElementById('MyChartLine'),
    ConfigLine
);
</script>

```

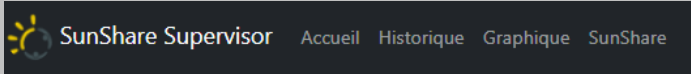

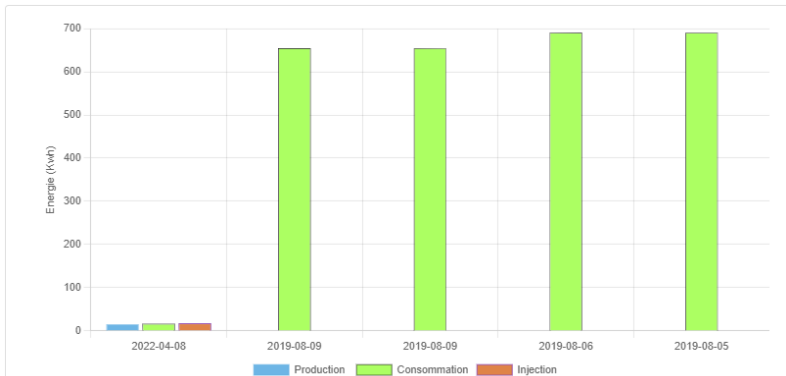
Figure 17: Intérieure du bloc rendu (histogramme et ligne)

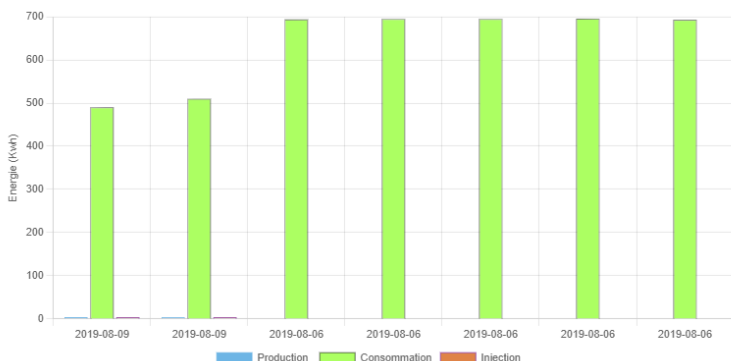
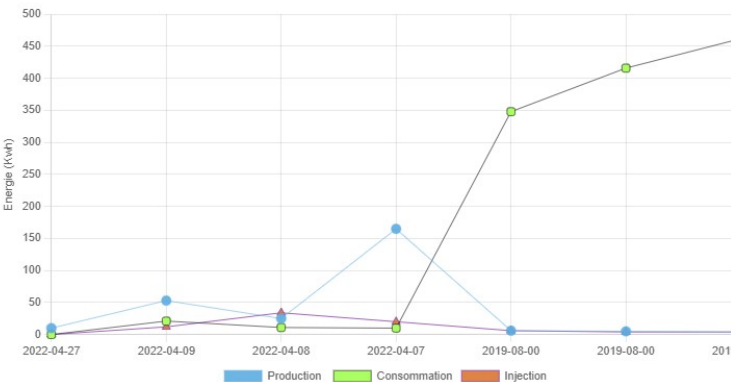
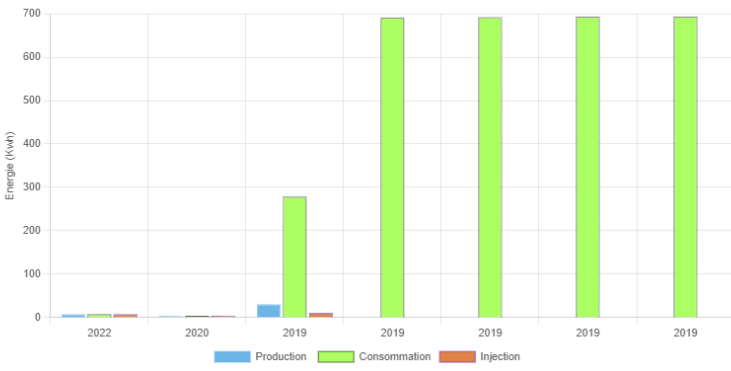
- Sur la **figure 17**, le **getElementById** fait référence à l'identifiant du canevas qui est nommé dans le document HTML. Dans cet exemple, il est nommé "myCharBar", mais on peut toujours le changer par n'importe quel nom que l'on souhaite. On doit simplement veiller à faire correspondre correctement le nom de l'identifiant du canevas.

3.3 - Tests unitaires

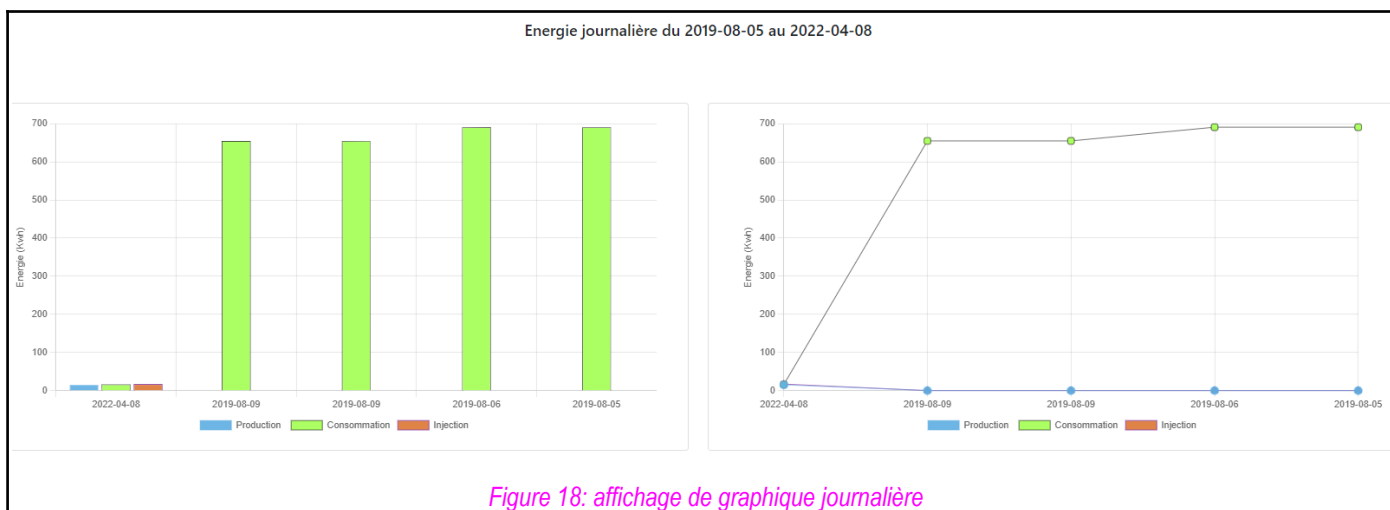
Dans ce test, en entrant une plage de dates (début et fin), je fais apparaître un graphique. À partir de cette plage de date, on va chercher des données et afficher des graphiques.

3.3.1 - Procédure de test

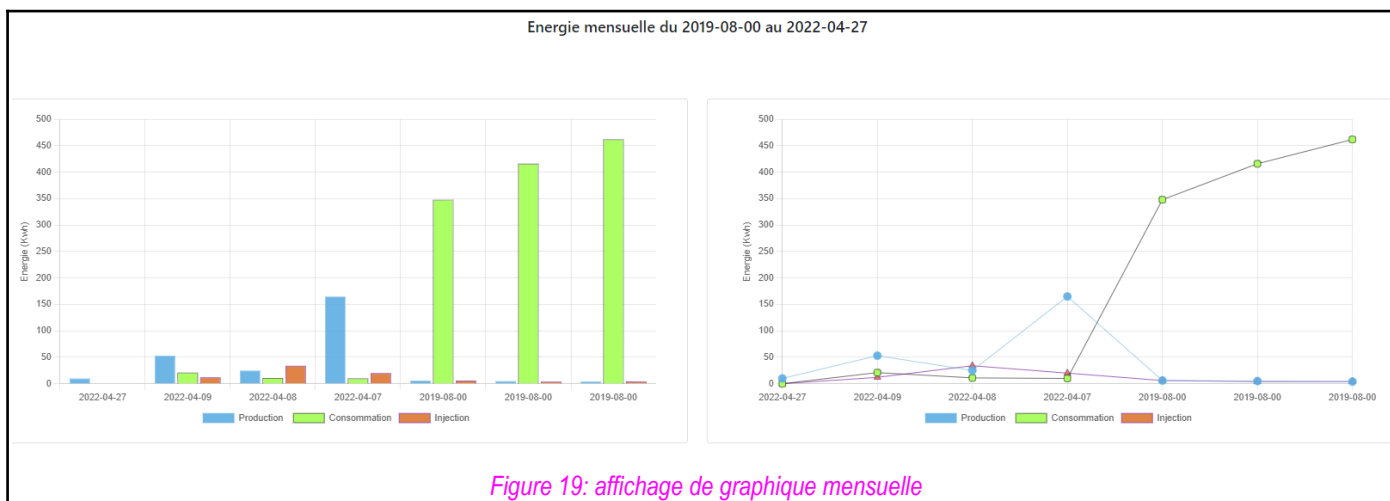
Id.	Méthode testée Description Sommaire	Procédure de test
		Résultats attendus
U3.0	<p>Tout d'abord, l'utilisateur se trouve sur la page d'accueil soit sur la page historique.</p> <p>Se rendre sur la page graphique, en choisissant graphique sur le barre de menus</p>	<p>Sur la page d'accueil :</p>  <p>Sur la page historique :</p>  <p>Accéder à la page graphique</p>
		<p>Saisir la date et choisir une période :</p> <p>ex :</p> <ul style="list-style-type: none"> Date de début 2019 -08-07 et date de fin 2019 -08-09 Période : journalière valider le choix
		<p>Afficher le graphiques</p> 
U3.2	Voir évolution énergétique hebdomadaire	<p>Saisir la date et choisir une période :</p> <p>ex :</p> <ul style="list-style-type: none"> Date de début 2019 -08-07 et date de fin 2019 -08-09 Période : hebdomadaire valider le choix

		
U3.3	Voir évolution énergétique mensuelle	<p>Saisir la date et choisir une période :</p> <p>ex :</p> <ul style="list-style-type: none"> • Date de début 2019 -08-00 et date de fin 2022 -04-27 • Période : Mensuelle • valider le choix 
U3.4	Voir évolution énergétique annuelle	<p>Saisir la date et choisir une période :</p> <p>ex :</p> <ul style="list-style-type: none"> • Date de début 2019 et date de fin 2022 • Période : Annuelle • valider le choix 

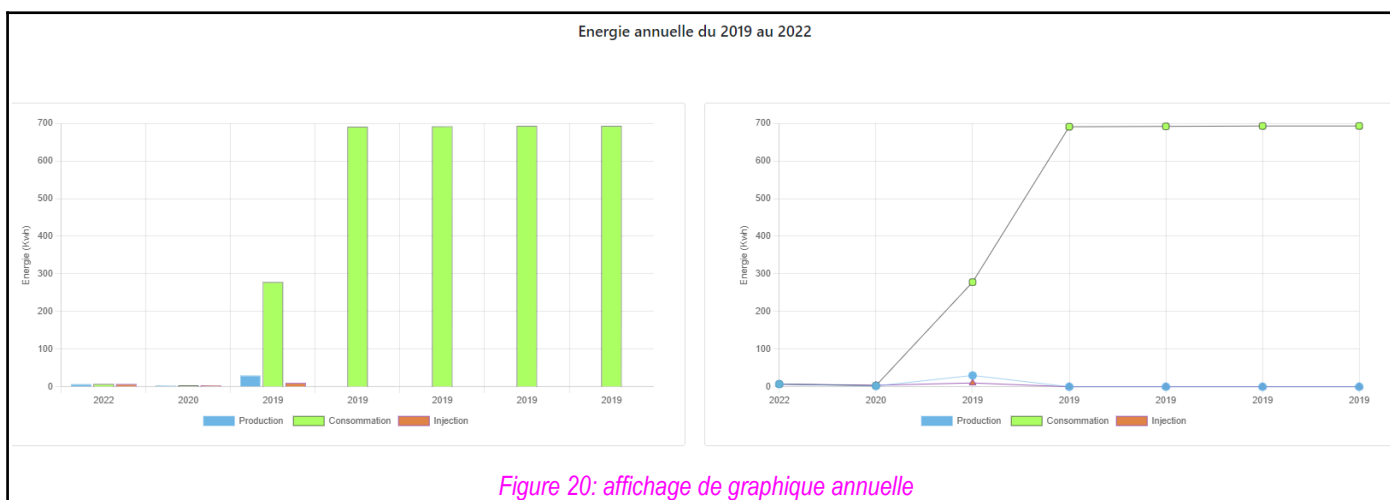
Résultat pour la période 2019-08-05 au 2022-04-08 ::



Résultat pour la période 2019-08-00 au 2022-04-27 :



Résultat pour la période 2019-2022 :



3.3.2 - Rapport d'exécution

Id.	OK	!OK	Observations
U3.0	*		Résultat OK
U3.1	*		Résultat OK
U3.2	*		Résultat OK
U3.3	*		Résultat OK
U3.4	*		Résultat OK

4 - Réalisation de la tâche « afficher un historique »

Dans ce test, en entrant une plage de dates (début et fin), je fais apparaître un historique de données dans un tableau, selon la date que je saisi. La procédure sera toujours la même, si on veut voir l'historique journalier, mensuel ou annuel.

4.1 - Cas d'utilisation

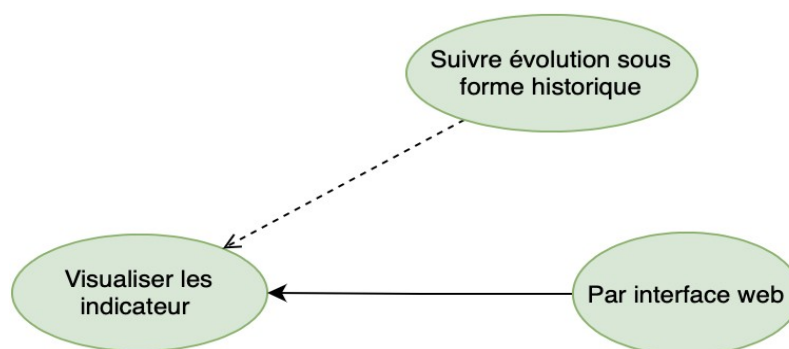
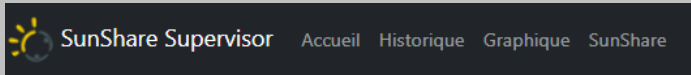
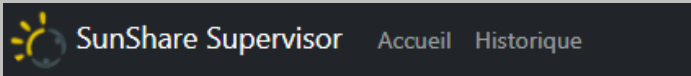


Figure 21: Visualiser les indicateurs via interface web

4.2 - Procédure de test

Id.	Méthode testée Description Sommaire	Procédure de test
		Résultats attendus
U4.0	Se rendre sur la page historique , en choisissant historique sur le barre de menus.	Menu de la page d'accueil :
		
		Menu de la page graphique :
		
		Accéder à la page historique

U4.1	Voir l'historique énergétique journalière	<p>Saisir une date (début et fin) et choisir une période.</p> <ul style="list-style-type: none"> Date de début et de fin : 2022-04-01 au 2022-04-07 historique : journalière <div> <p>Définir une période, pour consulter votre historique (*)</p> <div> <input type="text" value="aaaa-mm-jj"/> <input type="text" value="aaaa-mm-jj"/> <input type="button" value="Historique..."/> <input type="button" value="Valider"/> </div> </div> <ul style="list-style-type: none"> valider le choix La communication entre la page web et le serveur est opérationnelle. Affichage de l'historique journalière sur le site web
U4.2	Voir l'historique énergétique hebdomadaire	<p>Saisir une date (début et fin) et choisir une période.</p> <ul style="list-style-type: none"> Date de début et de fin : 2022-05-01 au 2022-06-04 historique : Hebdomadaire valider le choix <ul style="list-style-type: none"> La communication entre la page web et le serveur est opérationnelle. Affichage de l'historique hebdomadaire sur le site web
U4.3	Voir l'historique énergétique mensuelle	<p>Saisir une date (début et fin) et choisir une période.</p> <ul style="list-style-type: none"> Date de début et de fin : 2022-04-07 au 2019-11-07 Historique : mensuelle <ul style="list-style-type: none"> La communication entre la page web et le serveur est opérationnelle. Affichage de l'historique mensuelle sur le site web
U4.4	Voir l'historique énergétique annuelle	<p>Saisir une date (début et fin) et choisir une période.</p> <ul style="list-style-type: none"> Date de début et de fin : 2019-2022 Historique : annuelle <ul style="list-style-type: none"> La communication entre la page web et le serveur est opérationnelle. Affichage de l'historique annuelle sur le site web

4.2.1 - Affichage historique journalière 2022-04-01 au 2022-04-07

Pour réaliser ce teste j'ai dû ajoutés des données fictives via Phpmyadmin.

Date	Énergie produite (Kwh)	Énergie consommée (Kwh)	Énergie injectée (Kwh)
2022-04-07	3	2	1
2022-04-06	4	2	2
2022-04-05	2	2	0
2022-04-04	5	2	3
2022-04-03	2	2	0
2022-04-02	2	2	0
2022-04-01	1	0	0

Figure 22: Affichage de l'historique journalière

4.2.2 - Affichage historique hebdomadaire 2022-05-01 au 2022-06-04

Date	Énergie produite (Kwh)	Énergie consommée (Kwh)	Énergie injectée (Kwh)
2022-06-04	9	4	3
2022-05-28	14	7	7
2022-05-21	20	10	10
2022-05-14	12	5	7
2022-05-07	21	10	11
2022-05-01	10	9	1

Figure 23: Affichage de l'historique hebdomadaire

4.2.3 - Affichage historique mensuelle 2022-04-07 au 2022-11-07

Date	Énergie produite (Kwh)	Énergie consommée (Kwh)	Énergie injectée (Kwh)
2022-11-07	10	5	5
2022-10-07	9	4	5
2022-09-07	12	5	7
2022-08-07	3	2	1
2022-07-07	10	9	1
2022-06-07	8	4	4
2022-05-07	6	4	3
2022-04-07	165	10	20

Figure 24: Affichage de l'historique mensuelle

4.2.4 - Affichage historique annuelle 2019-2022

Date	Énergie produite (Kwh)	Énergie consommée (Kwh)	Énergie injectée (Kwh)
2019	0	685	0
2019	0	689	0
2019	0	691	0
2019	0	692	0
2019	0	693	0
2019	0	693	0
2019	63	69	18
2020	2	3	4
2022	7	7	7

Figure 25: Affichage de l'historique annuelle

4.2.5 - Rapport d'exécution

Id.	OK	!OK	Observations
U4.0	*		Résultat OK
U4.1	*		Résultat OK
U4.2	*		Résultat OK
U4.3	*		Résultat OK
U4.4	*		Résultat OK

5 - Bilan de la réalisation personnelle

5.1 - Statut des fonctions à ma charge

Tâches	Description	Statut
FP1 :	Installer et utiliser la base de données	Terminé
FP2 :	Afficher des données instantanées	Terminé
FP3 :	Afficher un historique de données	Terminé
FP4 :	Afficher des graphiques	Terminé
FS1 :	Télécharger sous format PDF un graphique	Terminé
FS2 :	Mettre des données dans un fichier format CSV	En cours

5.2 - Amélioration envisageable

- Sur la page historique, écrire un message du type «pas de données disponibles», si il n'y a pas de données disponibles avec la date choisie. Idem pour la page graphique.
- Afficher correctement le nom et numéro PDL de l'utilisateur sur la page d'accueil (réel information) .

5.3 - conclusion

5.3.1 - Point positifs

Ce projet m'a permis de développer encore plus mes connaissances dans le développement web, à découvrir des outils que je n'avais jamais utilisés auparavant, comme :

- La découverte du framework bootstrap
 - Prise en main assez simple et rapide pour un débutant.
- La découverte de la bibliothèque Chart.js
- La gestion du projet dans un groupe

5.3.2 - Point négatifs

- Problème avec la librairie qui permet de télécharger des graphiques fait en JavaScript en format PDF.
 - Difficile de trouver une bibliothèque qui est simple à mettre en place.