



Rapport commun



Projet Eau Pure

**Rédacteurs : M. CONAN Nolan, M. LE SOURNE
Mathys, M. SALOU Alexis**

Table des matières :

1. Partie commune.....	4
1.1. Contexte du projet.....	4
1.2. Analyse des besoins.....	5
51	

1. Partie commune

1.1. Contexte du projet

Le projet vise à automatiser la surveillance des eaux de surface et à diffuser en ligne des bulletins interactifs portant sur la qualité et la quantité de ces eaux. L'objectif principal est de contribuer à l'amélioration de la biodiversité et de la qualité de vie en fournissant un suivi précis et en temps réel des rivières de la région.

Ce suivi est assuré par le **Service Biodiversité Eau et Paysage (SBEP)**, un service rattaché à la **Direction Régionale de l'Environnement, de l'Aménagement et du Logement (DREAL)**. Le SBEP a pour mission de mettre en œuvre, sous l'autorité du préfet de région, les politiques publiques liées à la transition écologique.

Pour répondre à ces enjeux, le SBEP a déployé un réseau de **stations hydrologiques connectées** le long des cours d'eau. Chaque station est équipée des éléments suivants :

- Un système embarqué **Raspberry Pi** pour le pilotage et le traitement local.
- Deux capteurs : un **limnimètre** (niveau d'eau) et un **pluviomètre** (mesure des précipitations).
- Un **préleveur automatique** d'eau (actionneur).
- Un module de communication **LoRaWan** pour la transmission des données.
- Une source d'énergie **autonome** (ex : panneau solaire, batterie).

Ces stations, reliées à une **plateforme IoT**, forment un réseau de capteurs communicants via **LoRaWan**, un protocole sans fil à faible consommation d'énergie. Les données collectées sont transmises aux serveurs du SBEP via des passerelles LoRaWan connectées à Internet.

Les stations automatisent également le **prélèvement d'échantillons d'eau** en fonction des conditions hydrologiques et météorologiques (par exemple, lors d'une crue ou de fortes pluies). À chaque prélèvement, une **alerte par SMS** est envoyée à un technicien du SBEP, chargé de récupérer l'échantillon pour analyse en laboratoire.

Les résultats des analyses chimiques sont ensuite stockés dans une **base de données centralisée** et intégrés aux bulletins interactifs mis en ligne, permettant ainsi un suivi transparent et accessible de la qualité des eaux de surface.

Le SBEP occupe un bâtiment, dans ce bâtiment il y a 4 unités et chacune à son propre bureau. Il y a l'unité **pilotage et support** qui contient 3 personnes, l'unité **biodiversité** qui contient 6 personnes, l'unité **Natura 2000** qui compte 8 personnes et l'unité **eau** qui a un effectif de 10 personnes.

1.2. Rappel des attendus fonctionnels

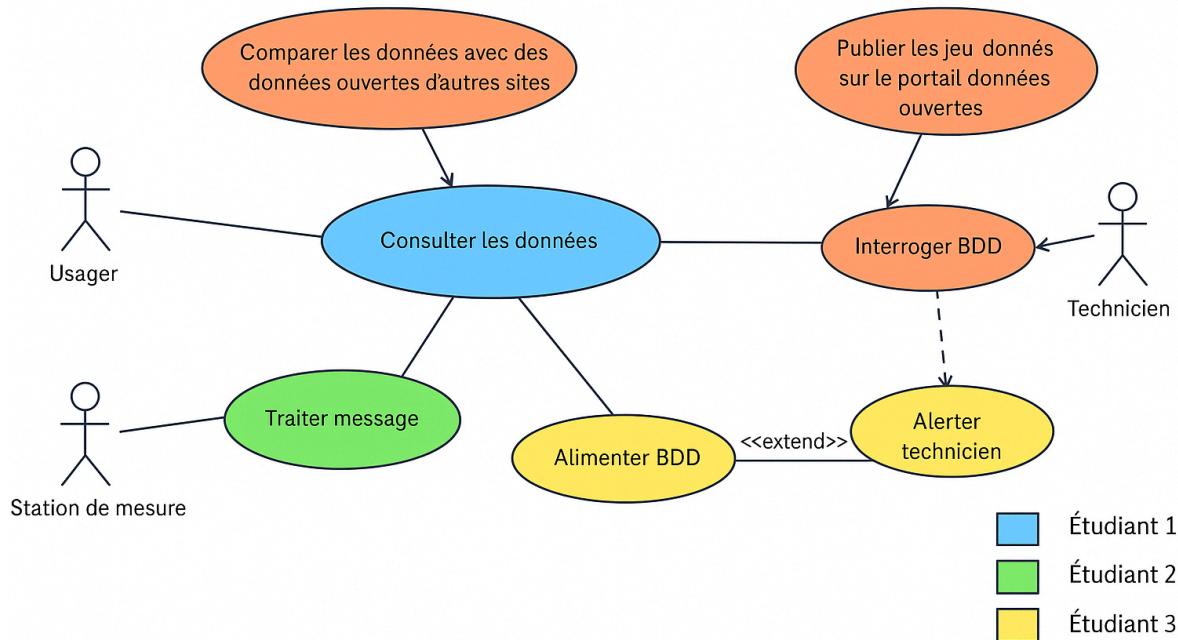


Figure 4: cas d'utilisation du sous-système SBEP

Le cas d'utilisation "**Traiter message**" est déclenché à la réception de tout message provenant du sous-système station. En fonction des données extraites, deux cas d'utilisation peuvent être activés :

- "**Alimenter BDD**", lorsqu'il s'agit d'enregistrer des mesures dans la base de données.
- "**Alerter technicien**", lorsqu'il s'agit d'envoyer une alerte.
Ce dernier cas d'utilisation "**Alerter technicien**" déclenche à son tour "**Interroger BDD**" afin de récupérer le numéro de téléphone du technicien responsable du prélèvement d'échantillons d'eau.

Une fois les mesures enregistrées dans la base de données, le technicien peut publier les résultats sur le portail de données ouvertes du SIE, via le cas d'utilisation "**Publier le jeu de données sur le portail de données ouvertes**", permettant ainsi une réexploitation nationale des informations.

Le cas d'utilisation "**Consulter les données**" permet à un utilisateur d'accéder, via un navigateur Internet, aux dernières mesures et prélèvements effectués. La consultation se fait de manière géolocalisée : en sélectionnant une station sur une carte, les informations correspondantes sont affichées. Ce cas déclenche également "**Interroger BDD**" pour récupérer les données nécessaires.

Enfin, le cas d'utilisation "**Comparer les données avec les données ouvertes d'autres sites**" utilise les informations disponibles sur le portail national pour permettre une comparaison entre les mesures locales et celles d'autres stations hydrologiques en France.

1.3. Analyse des besoins

Pour répondre aux exigences de la directive-cadre européenne sur l'eau (**2000/60/CE**), qui impose le respect de normes de qualité environnementale pour les masses d'eau, le SBEP a besoin d'un **système automatisé** capable de surveiller en continu la **qualité physico-chimique** des cours d'eau. Actuellement, les prélèvements d'échantillons d'eau sont réalisés manuellement et à intervalles réguliers par les techniciens, ce qui mobilise du temps et des ressources humaines. Le besoin est donc de **moderniser et automatiser ces tâches**, notamment en fonction des conditions météorologiques (pluie, crues) pour garantir des prélèvements pertinents. En parallèle, il est nécessaire d'assurer la **transmission en temps réel** des données mesurées (hauteur d'eau, pluviométrie) vers le SBEP, et de **diffuser ces données sur des plateformes publiques** (comme data.gouv.fr et le portail du SIE) pour assurer la transparence, la réutilisation nationale, et la gestion des risques (ex. Vigicrues). Enfin, le système doit aussi permettre la **signalisation immédiate des prélèvements** pour intervention humaine, et s'intégrer efficacement au **réseau informatique du SBEP**, tout en étant peu coûteux en maintenance grâce à une prise d'échantillon conditionnelle et intelligente.

2. Étudiant n°1 - Mathys LE SOURNE.....	10
2.1. Description du travail demandé.....	10
2.2 Tâche à réaliser.....	11
2.3. Réalisation d'un tableau scrum.....	12
3. Tâches réalisées.....	13
3.1. Le raspberry.....	13
3.2 Réalisation bouchon de test des capteurs.....	13
3.3. Réalisation bouchon de test préleur.....	14
3.4. Raccordement capteurs.....	14
3.4.1. Capteur limnimètre.....	14
3.4.2. Capteur pluviomètre.....	16
3.5. Lecture des données.....	18
4. Partie réseau.....	20
4.1. Plan réseau.....	20
4.2. Déploiement infrastructure.....	21
5. Sécurité.....	24
6. Travail collaboratif.....	26
7. Comparaison planning effectif au planning prévisionnel.....	26
8. Les outils utilisés.....	27
4. Annexes.....	66
4.1. Etudiant n°1.....	66

2. Étudiant n°1 - Mathys LE SOURNE

2.1. Description du travail demandé

Dans ce projet, mon rôle est de raccorder sur une carte raspberry deux capteurs (un capteur limnimètre et un capteur pluviomètre) afin de réaliser une station hydrologique. Dans un second temps je dois pouvoir lire les données des capteurs, les transformer si besoin et les transmettre dans une base de données via une passerelle LoRaWan.

Une autre partie est également à faire, je dois mettre en place une infrastructure réseau afin de faire communiquer l'intégralité des équipements et des services (station hydrologique, service SBEP et les serveurs WEB et BDD).

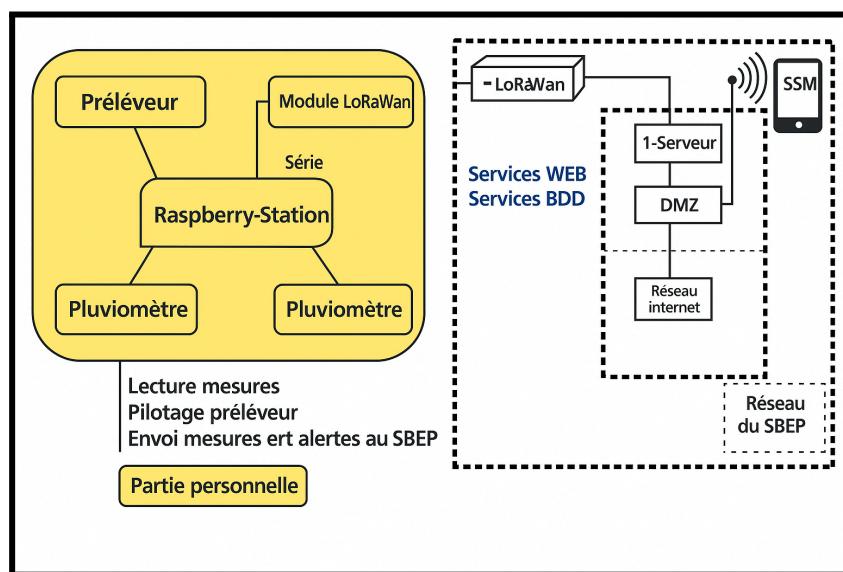


Figure n°1 : Diagramme de déploiement

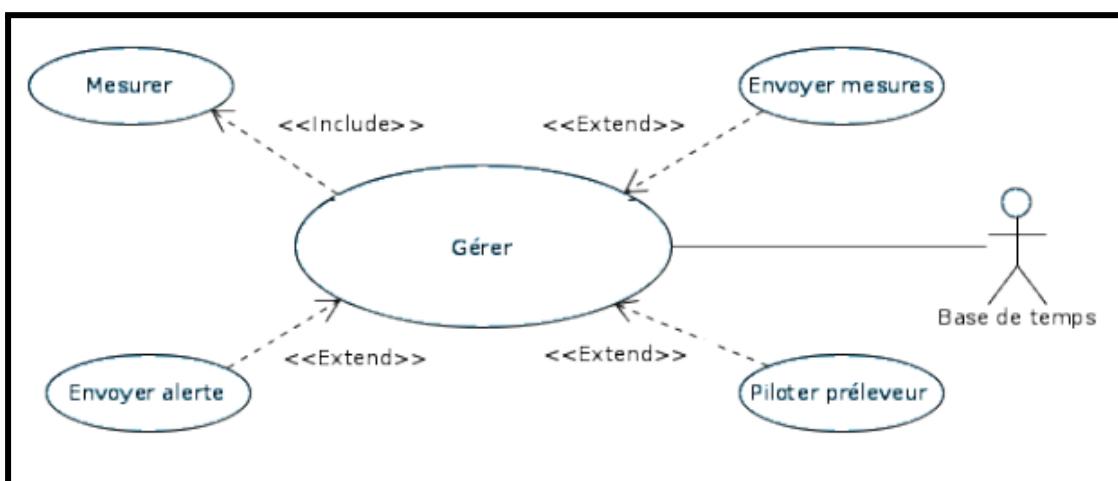


Figure n°2 : Diagrammes SYSML (cas d'utilisation du sous-système station)

Le cas d'utilisation Gérer est déclenché par une base de temps interne ; la période envisagée est la minute. Il inclut le cas d'utilisation Mesurer qui lit les capteurs, puis, une fois les mesures réalisées, il en effectue l'analyse. Il déclenche périodiquement le cas d'utilisation Envoyer mesures pour acheminer un récapitulatif des dernières mesures au sous-système SBEP ; la fréquence de déclenchement envisagée est de quatre fois par jour. Il déclenche également de manière conditionnelle les cas Envoyer alerte et Piloter préleveur en fonction de l'évolution des mesures effectuées afin d'envoyer une alerte et d'effectuer un prélèvement d'eau.

2.2 Tâche à réaliser

Dans le cadre de ce projet, plusieurs tâches m'ont été confiées à partir du cahier des charges. Tout d'abord, il s'agit d'installer une carte Raspberry ainsi que l'environnement logiciel nécessaire. Je dois ensuite raccorder les capteurs, à savoir le limnimètre et le pluviomètre à godets, à la carte. L'ensemble devra être alimenté par une source d'énergie autonome. Je dois également concevoir des bouchons de test permettant de simuler le fonctionnement des capteurs ainsi que l'ouverture du préleveur. Enfin, les mesures collectées devront être transmises à une base de données via une liaison LoRaWan.

Pour la deuxième partie de mes tâches, il s'agit de mettre en place une architecture réseau. Cela comprend la réalisation d'un plan du réseau, ainsi que d'un plan d'adressage permettant de configurer chaque équipement. Je suis également chargé d'étudier les différentes stratégies de sécurité à adopter afin de garantir la protection du réseau. Enfin, je devrai déployer cette architecture sur les équipements réels.

2.3. Réalisation d'un tableau scrum

Avant de se lancer concrètement dans la réalisation des tâches j'ai réalisé un tableau scrum qui contient un carnet de produit, c'est un document contenant l'objectif de produit et la liste des récits utilisateur qui peuvent être réalisés au cours d'une phase d'un projet et qui sont classés en ordre de priorité.

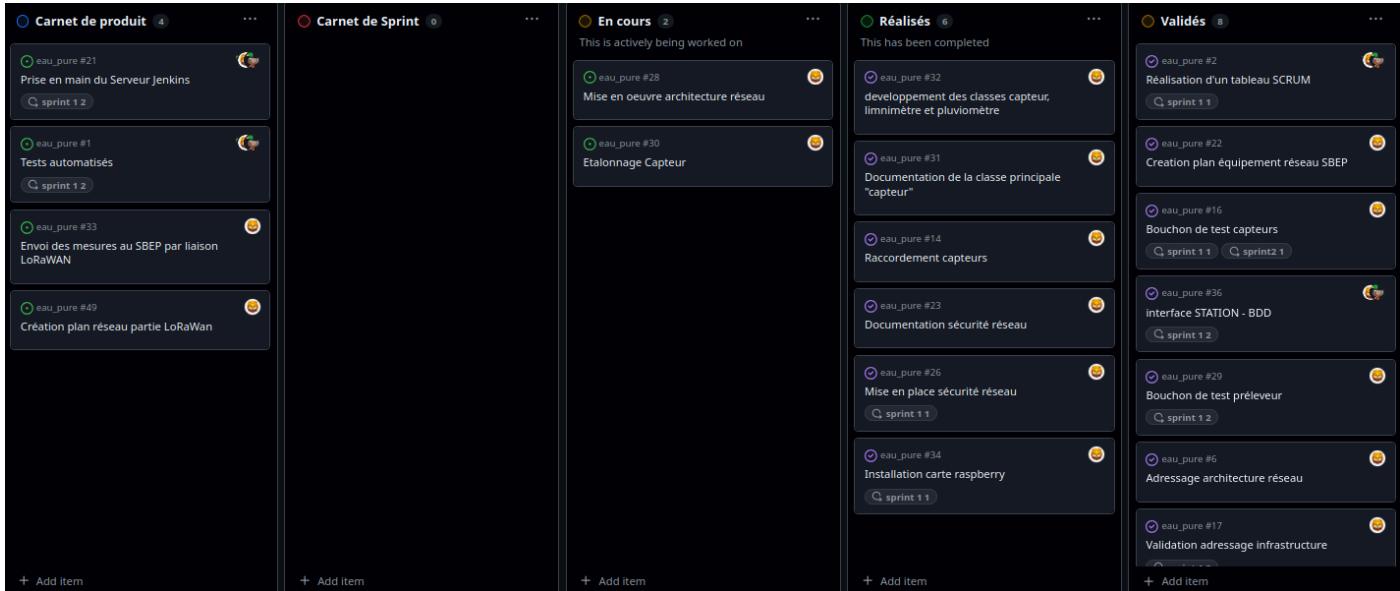


Figure n°2 : Tableau Scrum

Sur la colonne de gauche, nous pouvons voir le carnet de produit. J'ai sélectionné une tâche du cahier des charges que je décompose en plusieurs étapes. Cette méthode permet d'exécuter la tâche dans le bon ordre et de suivre une direction claire. Les récits utilisateurs peuvent être déplacés entre les différentes colonnes en fonction de leur progression. Cela est pratique pour suivre l'état d'avancement du projet, que ce soit le mien ou celui de mes collègues.

3. Tâches réalisées

3.1. Le raspberry

J'ai commencé par réinstaller le Raspberry Pi afin de disposer d'un environnement de travail complètement neuf, exempt de toute configuration antérieure pouvant interférer avec le projet. Ensuite, j'ai installé Python 3, en raison de sa richesse en bibliothèques adaptées aux projets sur Raspberry Pi. Parmi celles-ci, j'ai ajouté les bibliothèques essentielles pour le développement, notamment pip3, sqlite3, RPi.GPIO et mysql.connector.

Cependant, une erreur est survenue lors de l'installation de mysql.connector, indiquant qu'il était impossible d'installer un paquet Python dans un environnement géré de manière externe.

Pour contourner ce problème, j'ai créé un environnement virtuel à l'aide des commandes appropriées, ce qui m'a permis de continuer l'installation sans encombre.

```
python3 -m venv mon_env
source mon_env/bin/activate
```

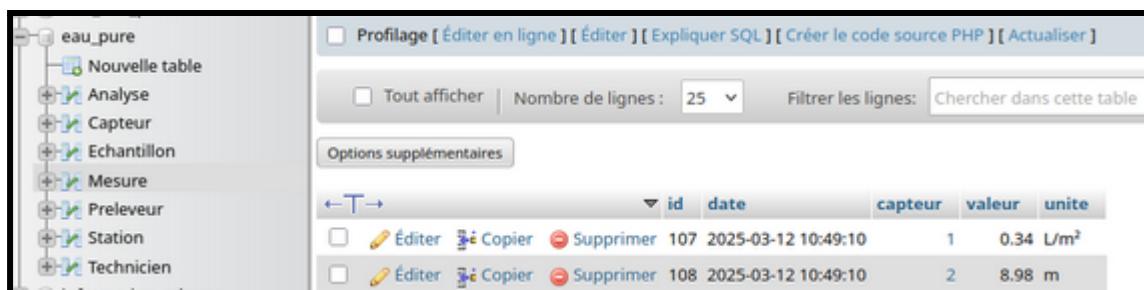
Figure n°3 : Commande création environnement virtuelle

Un environnement virtuel Python est un environnement isolé qui permet d'installer des packages Python pour un projet particulier sans interférer avec les packages installés dans l'environnement global (système).

3.2 Réalisation bouchon de test des capteurs

La conception du bouchon de test des capteurs a été réalisée en python. Le programme se connecte à la base de données avec le module mysql.connector et les informations de connexion que j'ai renseignées (login, mot de passe). Ensuite une fonction génère des valeurs factices pour les deux capteurs avec le module random, de 0 à 20 mm pour le pluviomètre et de 0 à 10 mètres pour le limnimètre. Une deuxième fonction qui insère les valeurs factices dans une requête sql. Pour finir la requête SQL s'exécute et envoie les informations vers la base de données (**annexe 1**).

Suite à l'exécution du programme, des données sont enregistrées dans la base de données.
(Le capteur numéro 1 correspond au pluviomètre, le numéro 2 au limnimètre)



		id	date	capteur	valeur	unite
		107	2025-03-12 10:49:10	1	0.34	L/m ²
		108	2025-03-12 10:49:10	2	8.98	m

Figure n°4 : Table mesure dans la base de donnée

3.3. Réalisation bouchon de test préleveur

Pour réaliser le bouchon de test du préleveur, j'ai également utilisé Python. Le programme établit une connexion à la base de données, de manière similaire au programme précédent. Il récupère ensuite les informations contenues dans la base et analyse la mesure la plus récente. En fonction de cette analyse, il vérifie le seuil du cours d'eau ainsi que les millimètres de

pluie enregistrés par le pluviomètre. Si le limnimètre indique une hauteur supérieure à 5 mètres ou si le pluviomètre enregistre 20 millimètres de pluie ou plus, le préleveur s'ouvre. Dans le cas contraire, il reste fermé. (**annexe 2**).

3.4. Raccordement capteurs

3.4.1. Capteur limnimètre

Le capteur limnimètre ALS-MPM-2F fonctionne sur le principe de la mesure de pression hydrostatique. Lorsqu'il est immergé dans un liquide, la pression exercée par la colonne d'eau est proportionnelle à la hauteur du liquide au-dessus du capteur. Cette pression est ensuite convertie en un signal électrique analogique de 4-20 mA, proportionnel au niveau de liquide mesuré. Le capteur peut mesurer des hauteurs de liquide allant de 0 à 5 mètres, ce qui le rend idéal pour surveiller des réservoirs, puits ou cours d'eau.

Pour raccorder le capteur limnimètre à la carte Raspberry Pi, il a fallu réaliser un circuit suiveur sur une plaque de test. En effet, les broches de la carte ne supportent que 16 mA, alors que le capteur peut délivrer jusqu'à 20 mA. De plus, un convertisseur analogique-numérique (ADC) est nécessaire pour pouvoir lire les données du capteur, la Raspberry Pi ne possédant pas d'entrée analogique native

Afin de réaliser le circuit, une résistance de 250 ohms est nécessaire. Ne disposant pas de cette valeur, j'ai utilisé quatre résistances de 100 ohms : deux branchées en série, puis deux autres en parallèle avec ce premier ensemble, ce qui permet d'obtenir une résistance équivalente proche de 250 ohms.

On utilise une résistance de 250 ohms car la carte Raspberry Pi reçoit un signal de courant de 4-20 mA. La tension maximale admissible sur son entrée est de 5 volts. En appliquant la loi d'Ohm ($R = U / I$), on obtient : $R = 5 / 0,02 = 250$ ohms. Cette résistance permet donc de convertir le signal de courant (4-20 mA) en un signal de tension compris entre 1 V et 5 V, lisible par l'entrée analogique.

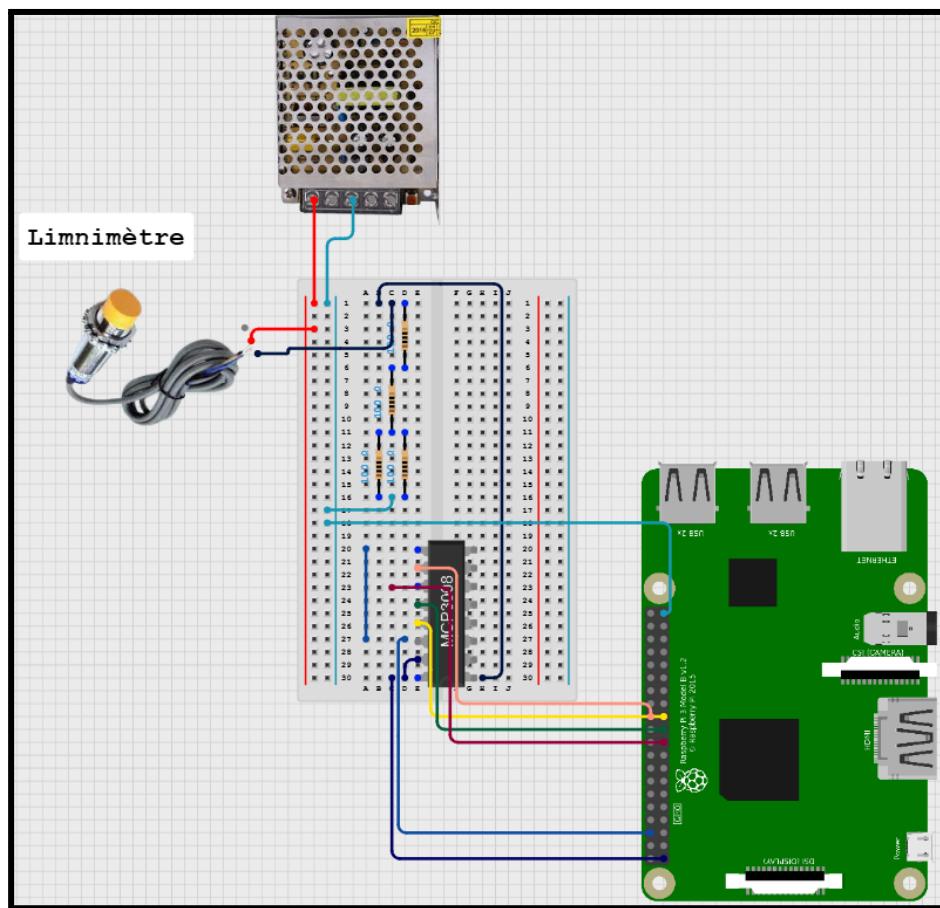


Figure n°5 : Schéma de raccordement capteur limnimètre

Le schéma ci-dessus montre une alimentation de 24 V qui alimente le capteur. La sortie de ce dernier est connectée à une résistance équivalente, dont l'entrée est également reliée à la broche CH0 du convertisseur. La sortie de la résistance est, quant à elle, reliée à la masse.

4. Connexion du MCP3208 à la Raspberry Pi :

- **VDD (broche 16)** du MCP3208 à **3.3V (broche 1)** de la Raspberry Pi.
- **VREF (broche 15)** du MCP3208 à **3.3V (broche 1)** de la Raspberry Pi.
- **AGND (broche 14)** du MCP3208 à **GND (broche 6)** de la Raspberry Pi.
- **DGND (broche 9)** du MCP3208 à **GND (broche 6)** de la Raspberry Pi.
- **CLK (broche 13)** du MCP3208 à **GPIO11 (SCLK, broche 23)** de la Raspberry Pi.
- **DOUT (broche 12)** du MCP3208 à **GPIO9 (MISO, broche 21)** de la Raspberry Pi.
- **DIN (broche 11)** du MCP3208 à **GPIO10 (MOSI, broche 19)** de la Raspberry Pi.
- **CS/SHDN (broche 10)** du MCP3208 à **D8 (CE0, broche 24)** de la Raspberry Pi.

Figure n°.. : Plan de branchement du convertisseur analogique-numérique MCP3208

3.4.2. Capteur pluviomètre

Ce pluviomètre est un capteur météorologique permettant de mesurer avec précision les précipitations atmosphériques. L'eau de pluie est collectée sur une surface de réception de 10.5 cm × 4.5 cm avant d'être dirigée vers un système de godet basculeur à l'intérieur du capteur. Ce système fonctionne sur le principe du balancier : lorsque l'un des godets atteint un volume prédéfini (correspondant à un certain nombre de millimètres de pluie), il bascule sous le poids de l'eau, se vide, et enclenche un contact électrique générant une impulsion.

Chaque impulsion représente une quantité précise d'eau, ce qui permet de calculer la hauteur de précipitation en litres par mètre carré (L/m²), équivalente à des millimètres (mm).

Pour déterminer la quantité d'eau correspondant à une impulsion du godet basculeur, j'ai utilisé une balance de précision. En versant de l'eau progressivement dans le capteur, j'ai observé qu'un basculement (une impulsion) se produisait tous les 1.64 grammes d'eau, soit 1.64 millilitres (puisque 1 g = 1 mL). Ensuite, j'ai divisé ce volume par la surface de réception du capteur (47.25 cm²) afin d'obtenir la hauteur de pluie correspondant à une impulsion, soit environ 0.35 mm.

$$\text{Hauteur de pluie (mm)} = \frac{1.64 \text{ mL}}{0.004725 \text{ m}^2} = \frac{0.00164 \text{ L}}{0.004725 \text{ m}^2} \approx 0.347 \text{ mm}$$

Ce type de capteur est souvent utilisé pour les stations météorologiques automatiques grâce à sa fiabilité et sa précision.



Figure n°6 : Capteur pluviomètre

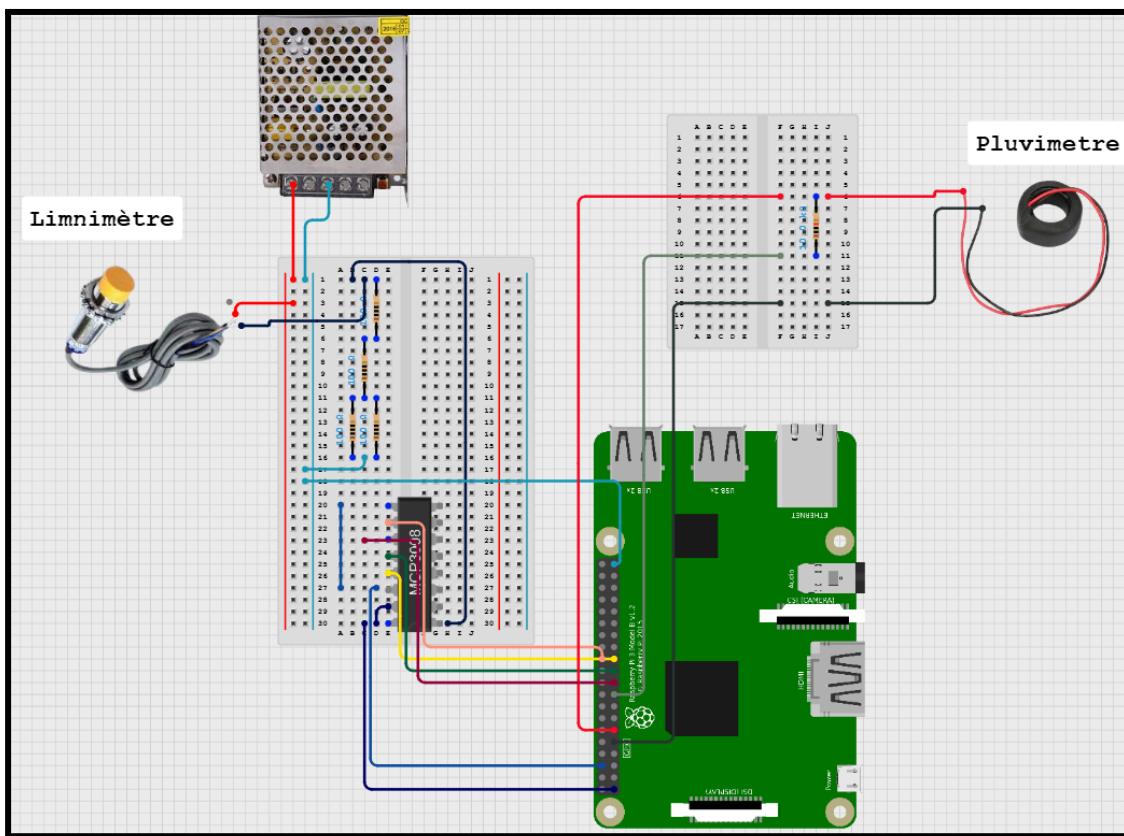


Figure n°7 : Schéma de raccordement capteur pluviometre

Le circuit pour connecter le pluviomètre à la carte Raspberry Pi est plus simple que celui du limnimètre. Une résistance de $10\text{ k}\Omega$ est utilisée car elle est couramment employée dans des circuits pour connecter des capteurs à une Raspberry Pi, assurant ainsi la protection du microcontrôleur tout en garantissant une lecture fiable des données du capteur.

Le terminal négatif du capteur est relié à la masse de la carte, tandis que le terminal positif est connecté à l'entrée de la résistance et à la broche GPIO17. La sortie de la résistance est, quant à elle, branchée sur le 3,3 V de la carte.

3.5. Lecture des données

Afin de lire les données des capteurs, j'ai commencé par développer un programme principal en Python intégrant les classes nécessaires à leur gestion. Par la suite, j'ai conçu un programme spécifique dédié à la lecture des données transmises par le limnimètre.

Ce programme Python permet l'acquisition, le traitement et la sauvegarde de mesures provenant d'un capteur de niveau d'eau 4–20 mA, via un convertisseur ADC MCP3208 connecté en SPI à une Raspberry Pi. Le capteur est instancié via la classe `WaterLevelSensor`, qui hérite d'une classe `Sensor` générique. Le module lit les données brutes (12 bits) depuis un canal du MCP3208, convertit cette valeur en tension (en tenant compte d'un facteur de correction pour compenser les pertes ou imprécisions), puis calcule le courant en mA en fonction d'une résistance de mesure.

Ce courant est ensuite traduit en profondeur (en mètres), selon une échelle linéaire basée sur la plage 4–20 mA et une profondeur maximale définie. Les résultats sont arrondis, horodatés, et insérés dans la base de données MySQL distante. Le système effectue une mesure toutes les 30 secondes, avec une gestion des erreurs SPI et de la connexion SQL.

```
projet-ep@raspberrypi:~/Desktop/capteur limnimetre $ python3 testliminemtre.py
Lecture des données du capteur de niveau d'eau...
Enregistrement inséré dans la table Mesure : ('2025-04-23 14:43:45', 0.8786241319444441, 'm', 2)
Tension : 1.703 V | Courant : 6.81 mA | Profondeur : 0.88 m
```

Figure n°8 : Résultat execution programme terminal

eau_pure		Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Crée le code source PHP] [Actualiser]					
		Tout afficher		Nombre de lignes :	25	Filtrer les lignes:	Chercher dans cette ta
		Options supplémentaires					
		← T →		id	date	capteur	valeur unité
				785	2025-04-23 14:43:45	2	0.88 m

Figure n°9 : Résultat execution programme phpmyadmin

J'ai également développé un programme pour lire les données du pluviomètre.

Ce programme définit une classe `RainSensor` permettant de mesurer les précipitations à l'aide d'un pluviomètre à godets basculants connecté à une broche GPIO de la Raspberry Pi. À chaque basculement (impulsion électrique), le capteur génère une transition de niveau logique de haut à bas, détectée par la méthode `update()`. Chaque impulsion correspond à une quantité fixe de pluie (définie par le paramètre `impulse_to_rain_mm`) qui est cumulée pour obtenir une mesure en millimètres de précipitation. Le programme garde également un historique de la dernière impulsion détectée, et fournit des méthodes pour réinitialiser ou récupérer la valeur totale de pluie tombée (**ajouter annexe**).

```
projet-ep@raspberrypi:~/Desktop/fonctionnel $ python3 main.py
Détection de pluie en cours...
Impulsion détectée ! Total : 1, Pluie tombée : 0.35 L/m²
Impulsion détectée ! Total : 2, Pluie tombée : 0.70 L/m²
Impulsion détectée ! Total : 3, Pluie tombée : 1.05 L/m²
Impulsion détectée ! Total : 4, Pluie tombée : 1.40 L/m²
Impulsion détectée ! Total : 5, Pluie tombée : 1.75 L/m²
Impulsion détectée ! Total : 6, Pluie tombée : 2.10 L/m²
Impulsion détectée ! Total : 7, Pluie tombée : 2.45 L/m²
Impulsion détectée ! Total : 8, Pluie tombée : 2.80 L/m²
Mesure insérée: 1, 2.8000000000000003, L/m², 2025-04-23 17:21:57
Envoi des données: 2.80 L/m² à 2025-04-23 17:21:57
```

Figure n°8 : Résultat execution programme terminal

The screenshot shows the phpMyAdmin interface for the 'eau_pure' database. On the left, there's a tree view of tables: Nouvelle table, Analyse, Capteur, Echantillon, Mesure, Prelevage, and Station. On the right, a table view for the 'Mesure' table is displayed. The table has columns: id, date, capteur, valeur, and unite. One row is visible: id=785, date=2025-04-23 17:21:57, capteur=1, valeur=2.8, unite=L/m².

Figure n°9 : Résultat execution programme phpmyadmin

Par la suite, je me suis coordonné avec l'étudiant 2, afin de définir l'interface de transmission des données entre les capteurs et la base de données. Nous avons convenu d'utiliser le langage Python pour assurer cette communication.

L'étudiant 2 a développé un programme principal, **main.py**, qui centralise la réception des données provenant des deux capteurs et automatise leur envoi vers la base de données, sans que je n'aie à écrire directement de requêtes SQL. Pour la gestion des requêtes SQL, il a conçu un second programme, **interf.py**, qui les encapsule, qui contient les fonctions. Les deux scripts sont interconnectés pour assurer un traitement fluide et structuré des données.

```
Détection des capteurs en cours...
Impulsion détectée ! Total : 1, Pluie tombée : 0.35 L/m²
Impulsion détectée ! Total : 2, Pluie tombée : 0.70 L/m²
Impulsion détectée ! Total : 3, Pluie tombée : 1.05 L/m²
Impulsion détectée ! Total : 4, Pluie tombée : 1.40 L/m²
Impulsion détectée ! Total : 5, Pluie tombée : 1.75 L/m²
Impulsion détectée ! Total : 6, Pluie tombée : 2.10 L/m²
Impulsion détectée ! Total : 7, Pluie tombée : 2.45 L/m²
Impulsion détectée ! Total : 8, Pluie tombée : 2.80 L/m²
Impulsion détectée ! Total : 9, Pluie tombée : 3.15 L/m²
Mesure insérée: 1, 3.1500000000000004, L/m², 2025-04-24 14:10:43
Pluie envoyée : 3.15 L/m² à 2025-04-24 14:10:43
Mesure insérée: 2, 0.64, m, 2025-04-24 14:10:43
Profondeur envoyée : 0.64 m (V=1.52V, I=6.06mA)
Réponse de l'API: OK
```

The screenshot shows the phpMyAdmin interface for the 'eau_pure' database. On the left, there's a tree view of tables: Echantillon, Mesure, Prelevage, Station, and Technicien. On the right, a table view for the 'Mesure' table is displayed. The table has columns: id, date, capteur, valeur, and unite. Two rows are visible: id=802, date=2025-04-24 14:10:43, capteur=1, valeur=3.15, unite=L/m²; and id=803, date=2025-04-24 14:10:43, capteur=2, valeur=0.64, unite=m.

Figure n°80 : Résultat execution programme main.py

4. Partie réseau

4.1. Plan réseau

Pour mettre en place l'infrastructure réseau, il est essentiel de commencer par établir un schéma représentant l'ensemble des équipements disponibles. Celui-ci inclut deux switchs, un routeur, une passerelle LoRaWAN, une station hydrologique, ainsi que les serveurs Web et de base de données. Le réseau du SBEP comprend également 27 postes de travail répartis entre quatre services : Pilotage et Support, Biodiversité, Natura 2000 et Eau.

J'ai donc utilisé l'outil Cisco Packet Tracer pour réaliser un schéma d'ensemble du réseau et effectuer quelques tests préliminaires avant de déployer la configuration sur le matériel réel.

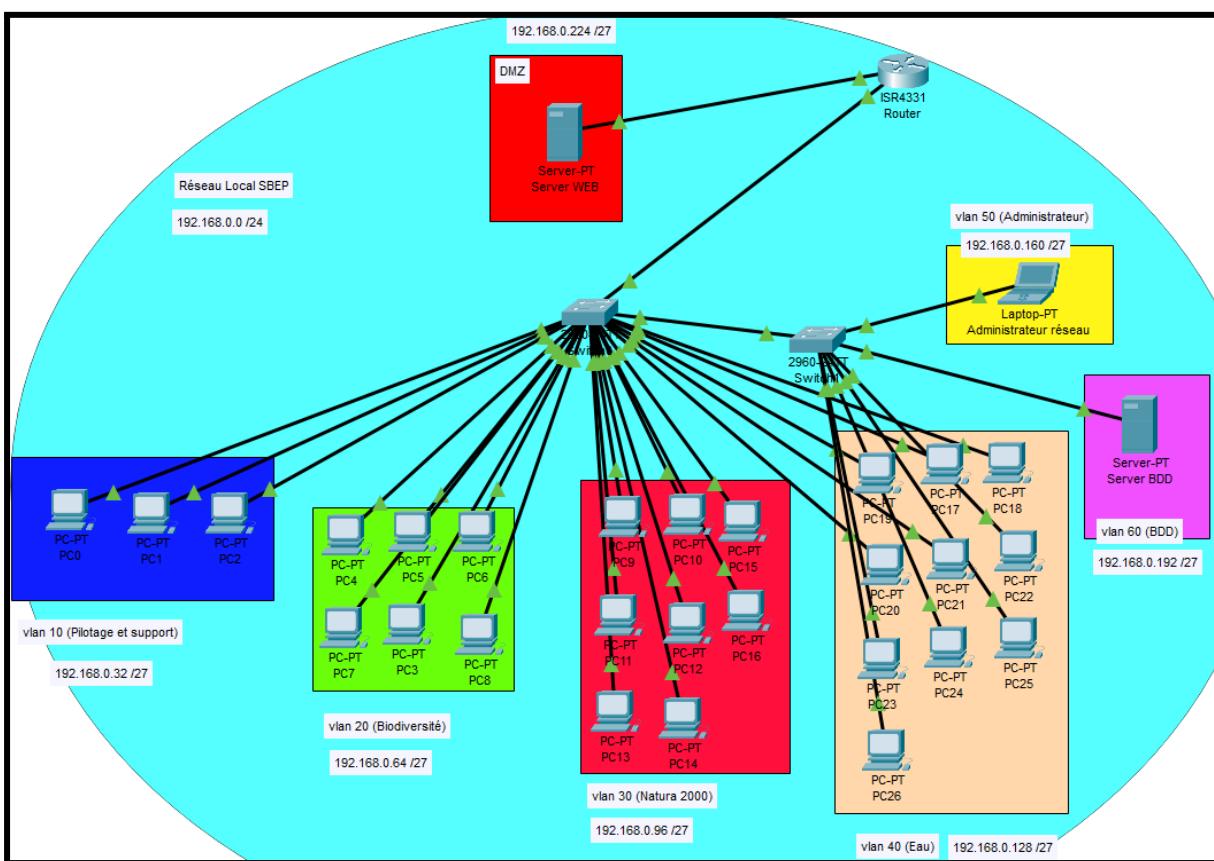


Figure n°9 : Schéma réseau local SBEP

J'ai mis en place des VLANs pour chacun des services afin de structurer le réseau de manière logique et sécurisée. Ensuite, j'ai élaboré un plan d'adressage IP détaillé pour chaque machine du réseau (**annexe 3**). J'ai opté pour une adresse de classe C, adaptée à la taille modeste du réseau, puis je l'ai subdivisée en sous-réseaux distincts, chacun correspondant à un VLAN. Chaque VLAN dispose ainsi de sa propre adresse réseau, d'une adresse de diffusion (broadcast) et d'une passerelle dédiée.

Sur les switchs, j'ai attribué les VLANs aux ports correspondants en fonction des services, puis j'ai configuré les liaisons entre les deux switchs et le routeur en mode trunk afin de

permettre le transport des VLANs. Côté routeur, j'ai créé des sous-interfaces, une par VLAN, pour assurer le routage inter-VLAN et permettre la communication entre les différents segments du réseau.

4.2. Déploiement infrastructure

Tout d'abord, je crée les VLANs sur les switchs (cisco) en utilisant l'interface ligne de commande. Ci-dessous, vous pouvez voir la création du VLAN 20 (Biodiversité) et son affectation aux ports 4 à 9 du switch.

```
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)#vlan 20
Switch(config-vlan)#name Biodiversite
Switch(config-vlan)#exit
Switch(config)#int range fa0/4-9
Switch(config-if-range)#switchport mode access
Switch(config-if-range)#switchport access vlan 20
Switch(config-if-range)#exit
Switch(config)#exit
```

Figure n°10 : Configuration vlans

J'ai répété cette tâche pour chaque VLAN sur le switch 1. Les VLANs 50 et 60 ont été configurés sur le deuxième switch.

VLAN	Name	Status	Ports
1	default	active	Gi0/1
10	Pilotage_et_support	active	Fa0/1, Fa0/2, Fa0/3
20	Biodiversite	active	Fa0/4, Fa0/5, Fa0/6, Fa0/7 Fa0/8, Fa0/9
30	Natura_2000	active	Fa0/10, Fa0/11, Fa0/12, Fa0/13 Fa0/14, Fa0/15, Fa0/16, Fa0/17
40	Eau	active	Fa0/18, Fa0/19, Fa0/20, Fa0/21 Fa0/22, Fa0/23, Fa0/24
50	Administrateur	active	
60	BDD	active	
1002	fddi-default	act/unsup	
1003	token-ring-default	act/unsup	
1004	fdtnet-default	act/unsup	
1005	trnet-default	act/unsup	

Figure n°11 : Association port du switch 1 aux vlans

Voici la configuration des vlans 50 et 60 sur le deuxième switch.

VLAN Name	Status	Ports
1 default	active	Fa0/3, Fa0/4, Fa0/5, Fa0/6 Fa0/7, Fa0/8, Fa0/9, Fa0/10 Fa0/11, Fa0/12, Fa0/13, Fa0/14 Fa0/15, Fa0/16, Fa0/17, Fa0/18 Fa0/19, Gig0/2
40 Eau	active	Fa0/20, Fa0/21, Fa0/22, Fa0/23 Fa0/24
50 Administrateur	active	Fa0/1
60 BDD	active	Fa0/2
1002 fddi-default	active	
1003 token-ring-default	active	
1004 fddinet-default	active	
1005 trnet-default	active	

Figure n°12 : Association port du switch 2 aux vlan

Ensuite, j'ai activé le mode trunk entre les switchs avec les lignes de commande suivantes.

```
enable
configure terminal
interface GigabitEthernet0/1
switchport mode trunk
```

Figure n°13 : Configuration mode trunk

J'ai activé le mode trunk sur les ports **Gi0/1** et **Gi0/2** du switch 1 : le port **Gi0/1** est relié au routeur, et le port **Gi0/2** est connecté au switch 2. Sur le switch 2, j'ai configuré le port **Gi0/1** en mode trunk pour établir la liaison avec le switch 1.

Ensute j'ai configuré le routeur, j'ai créé une sous interface par vlan car il n'y a pas assez de port sur le routeur. Cela permet aussi le routage inter-vlan.

Voici la configuration :

```
interface GigabitEthernet0/0/0.10
  encapsulation dot1Q 10
  ip address 192.168.0.62 255.255.255.224
!
interface GigabitEthernet0/0/0.20
  encapsulation dot1Q 20
  ip address 192.168.0.94 255.255.255.224
!
interface GigabitEthernet0/0/0.30
  encapsulation dot1Q 30
  ip address 192.168.0.126 255.255.255.224
!
interface GigabitEthernet0/0/0.40
  encapsulation dot1Q 40
  ip address 192.168.0.158 255.255.255.224
!
interface GigabitEthernet0/0/0.50
  encapsulation dot1Q 50
  ip address 192.168.0.190 255.255.255.224
!
interface GigabitEthernet0/0/0.60
  encapsulation dot1Q 60
  ip address 192.168.0.222 255.255.255.224
  ip access-group BDD_ACCESS in
!
interface GigabitEthernet0/0/1
  ip address 192.168.0.254 255.255.255.224
  duplex auto
  speed auto
```

Figure n°14 : Configuration sous interface sur le routeur

Ci-dessous, nous pouvons voir les commandes pour créer la sous interface du vlan 10.

```
enable
configure terminal

interface GigabitEthernet0/0/0.10
  encapsulation dot1Q 10
  ip address 192.168.0.62 255.255.255.224

exit
```

Figure n°15 : Création sous interface sur le routeur

5. Sécurité

Aujourd'hui, il est crucial de sécuriser son réseau en raison des nombreuses attaques qui se produisent chaque année et des données sensibles que peut contenir une entreprise. C'est pourquoi j'ai mis en place plusieurs mesures de sécurité pour protéger le réseau.

La mise en place de mot de passe sur les équipements switch et routeur pour éviter que quelqu'un de mal intentionné ait accès aux appareils.

```
enable
configure terminal
enable secret monmotdepasse
line con 0
password monmotdepasseconsole
login
write memory
```

User Access Verification

Password: |

Figure n°16 : Mise en place mot de passe équipement cisco

Activer le Spanning Tree Protocol (STP) qui est un protocole de réseau qui permet de prévenir les boucles de commutation dans un réseau local (LAN). Dans un réseau avec plusieurs switches, des boucles peuvent se former si des chemins multiples existent entre les switches. Ces boucles peuvent provoquer des tempêtes de broadcast et rendre le réseau instable.

Par défaut il est activer mais il faut vérifier avec la commande : **show spanning-tree**

J'ai également désactiver telnet et ssh car j'en ai pas besoin dans le projet et cela limite les possibilité de connexion aux équipements.

```
enable
configure terminal

# Désactiver Telnet
line vty 0 4
transport input none

# Désactiver SSH
no ip ssh

write memory
```

Figure n°17 : Désactivation protocole Telnet et SSH

Pour sécuriser la base de données, j'ai restreint son accès en configurant une liste de contrôle d'accès (ACL) sur le routeur. Celle-ci permet uniquement au service Eau (VLAN 40) ainsi qu'à l'administrateur d'établir une communication avec le VLAN 60, où se trouve la base de données.

```
Router(config)# ip access-list extended BDD_ACCESS
Router(config-ext-nacl)# permit ip 192.168.0.192 0.0.0.31 192.168.0.128 0.0.0.31
Router(config-ext-nacl)# permit ip 192.168.0.192 0.0.0.31 192.168.0.160 0.0.0.31
Router(config-ext-nacl)# deny ip 192.168.0.192 0.0.0.31 192.168.0.32 0.0.0.31
Router(config-ext-nacl)# deny ip 192.168.0.192 0.0.0.31 192.168.0.64 0.0.0.31
Router(config-ext-nacl)# deny ip 192.168.0.192 0.0.0.31 192.168.0.96 0.0.0.31
Router(config-ext-nacl)# permit ip 192.168.0.192 0.0.0.31 any
Router(config-ext-nacl)# exit
```

Figure n°18 : Liste de contrôle d'accès

Puis j'ai appliqué l'ACL à l'interface GE 0/0/0.60 (sous interface vlan 60) du routeur.

```
Router(config)# interface GigabitEthernet0/0.192
Router(config-if)# ip access-group BDD_ACCESS in
Router(config-if)# exit
```

Figure n°19 : Mise en application de la liste de contrôle d'accès

Zone DMZ

Le serveur web a été placé dans une DMZ afin de mieux contrôler le trafic entrant et sortant de cette zone. En cas d'intrusion, la DMZ permet de limiter les risques en réduisant la surface d'attaque, car elle constitue un segment isolé du reste du réseau interne.

DMZ : Une DMZ (Demilitarized Zone) est une zone neutre du réseau placée entre le réseau interne d'une organisation et l'extérieur (généralement Internet). Elle sert à héberger des services accessibles depuis l'extérieur, comme un serveur web ou mail, tout en protégeant le réseau interne. Les équipements dans la DMZ sont isolés par des pare-feux, ce qui permet de contrôler précisément les flux entrants et sortants. En cas de compromission, l'attaquant reste confiné dans cette zone, limitant ainsi les risques pour le reste du système d'information.

Politique de sécurité relative aux mots de passe

J'ai également mis en place une politique de sécurité concernant les mots de passe : ceux-ci doivent contenir au minimum une majuscule, un caractère spécial, et comporter au moins 7 caractères.

6. Travail collaboratif

Nous utilisons GitHub comme outil collaboratif, ce qui permet à tous les membres de l'équipe de contribuer efficacement. Grâce à cet espace partagé, chacun peut accéder aux documents, effectuer des modifications, et suivre les différentes étapes du projet en temps réel. Cet outil facilite également la gestion des versions, permettant à chaque membre de revenir sur des changements ou d'analyser l'historique des modifications. Ainsi, GitHub renforce la transparence et la coordination, tout en assurant que tous les membres disposent des mêmes informations pour avancer ensemble.

7. Comparaison planning effectif au planning prévisionnel

Dans le cadre du projet, le planning effectif a été comparé au planning prévisionnel afin d'établir un constat sur l'avancement des tâches. Cette analyse a permis d'identifier certains écarts, notamment le raccordement du capteur limnimètre, qui a pris plus de temps que prévu en raison d'un problème détecté dans le circuit. Ce constat met en évidence l'importance d'une gestion rigoureuse et d'une anticipation des imprévus techniques pour minimiser les retards dans le projet.

8. Les outils utilisés

Github :

Description : GitHub est une plateforme de gestion de versions et de collaboration basée sur Git. Elle est largement utilisée dans le développement logiciel pour gérer les projets, suivre les modifications et faciliter le travail en équipe.

Choix de cet outil : Accessibilité, intégrations multiples, sécurité et sauvegarde.

Phpmyadmin :

Description : PhpMyAdmin est une application web open source qui permet de gérer facilement des bases de données MySQL via une interface graphique. Accessible depuis un navigateur web, il simplifie l'administration des bases de données en offrant une interface intuitive et des fonctionnalités puissantes.

Choix de cet outil : accessibilité, facilité d'utilisation, open source et gratuit.

Cisco packet tracer :

Description : Cisco Packet Tracer est un outil de simulation de réseaux développé par Cisco. Il permet de concevoir, configurer et simuler des infrastructures réseau complexes dans un environnement virtuel. Cet outil est largement utilisé pour l'apprentissage et la conception de réseaux, tout en offrant une interface graphique intuitive.

Choix de cet outil : formation et apprentissage, économie de ressources, facilité d'utilisation.

Python :

Description : Python est un langage de programmation interprété, open source et polyvalent, reconnu pour sa simplicité et sa lisibilité. Il est largement utilisé dans de nombreux domaines tels que le développement web, la science des données, l'intelligence artificielle, l'automatisation et bien plus encore.

Choix de cet outil : polyvalence, rapidité de développement et écosystème riche.

MySQL :

Description : MySQL est un système de gestion de bases de données relationnelles (SGBDR) open source largement utilisé pour stocker, organiser et interroger des données. Connue pour sa performance, sa fiabilité et sa simplicité, il est particulièrement adapté aux applications web et aux systèmes nécessitant une gestion efficace des données.

Choix de cet outil : fiabilité éprouvée, adapté aux applications web, efficacité.

Étudiant n°2 - Alexis SALOU

Étudiant n°2 - Alexis SALOU	11
1. Synoptique/contexte	14
2. Diagramme de cas d'utilisation	14
3. Architecture des composants	15
4. Tâches à réaliser (cahier des charges)	16
5. Gestion du temps et du travail	16
6. Listes des tâches effectuées	17
7. Description du Matériel/Logiciel	21
8. Politique de protection et de sécurité des données	49
4. Annexe	51
4.2. Etudiant n°2	52
	56

1. Synoptique/contexte

Dans le projet, je suis l'étudiant numéro 2. Mon rôle est de faire le lien entre la partie station et la partie WEB. Pour cela, j'ai utilisé plusieurs outils : Base de données, interfaçage etc ...

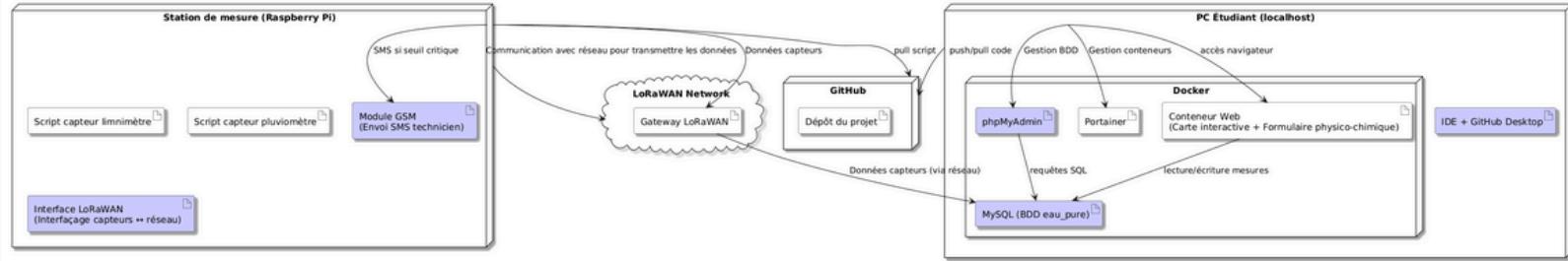


image n°1 : Diagramme

2. Diagramme de cas d'utilisation

En tant qu'étudiant n°2, mon périmètre fonctionnel inclut principalement la création, la gestion et l'interfaçage de la base de données. Le cas d'utilisation principal me concernant est « Gérer les données capteurs », qui se décline en « Créer les tables de la BDD », « Recevoir des mesures » et « Rendre les données accessibles au site web ». Un autre cas d'usage est « Gérer les alertes technicien », activé par la réception de données critiques. Ces scénarios sont modélisés dans un diagramme de cas d'utilisation :

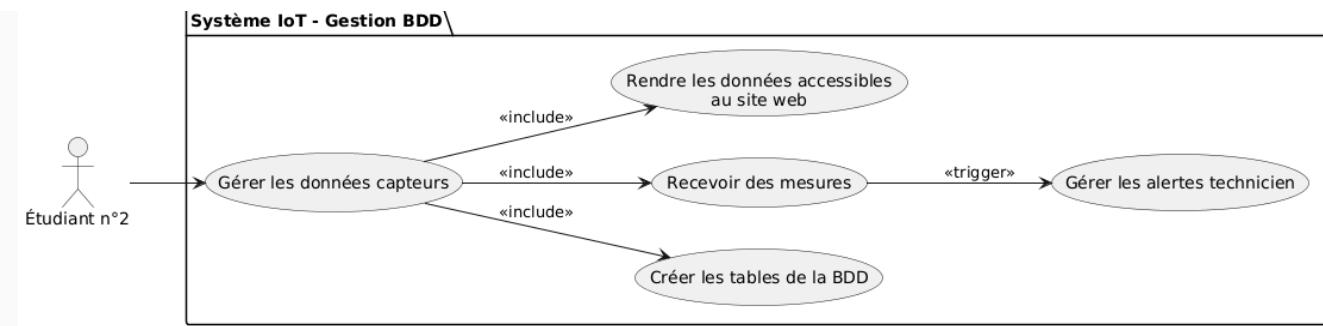


Image n°2 : Diagramme d'utilisation

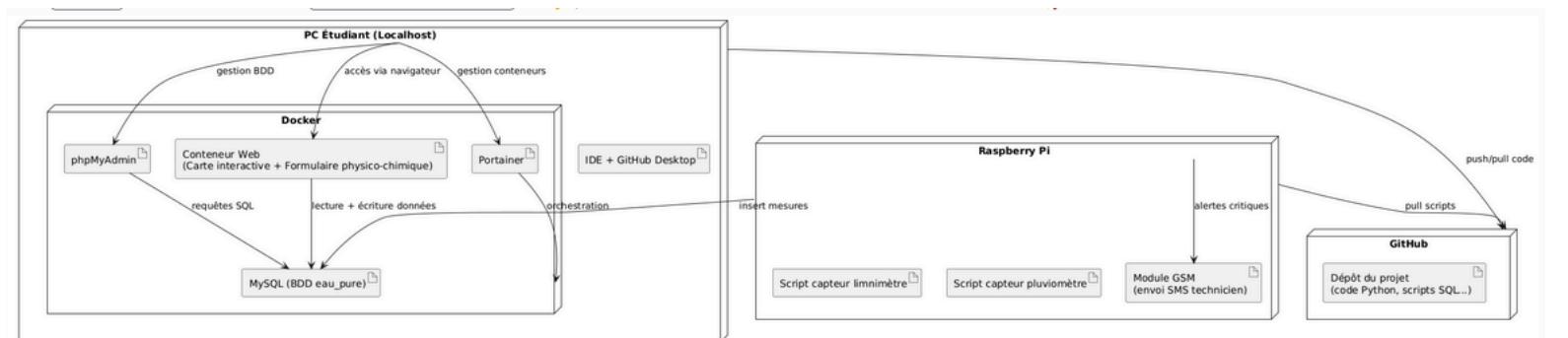


Image n°3 : Diagramme UML

3. Architecture des composants

L'architecture du système repose sur une segmentation claire entre les stations hydrologiques, les serveurs d'analyse et l'interface utilisateur web. Chaque station, équipée d'un Raspberry Pi, transmet ses données via LoRaWAN à une passerelle, elle-même connectée à un serveur central. Ce serveur héberge un environnement LAMP (Linux, Apache, MySQL, PHP) conteneurié avec Docker, permettant de traiter les mesures reçues et de les stocker dans une base de données accessible via PhpMyAdmin. L'étudiant 3 accède à cette base pour alimenter la carte interactive web. Un diagramme UML de déploiement est disponible en annexe pour illustrer cette répartition logicielle et matérielle.

PhpMyAdmin est une interface web puissante permettant la gestion et l'administration des bases de données MySQL. Elle permet d'exécuter des requêtes SQL, de visualiser les données sous forme de tables, et de créer ou modifier des structures de manière intuitive. C'est un outil indispensable pour suivre les flux de données provenant des stations.

Docker permet d'exécuter chaque service (base de données, serveur web, interface Portainer...) dans un conteneur isolé. Cela simplifie la maintenance et rend l'environnement facilement duplicable d'une machine à une autre. C'est une bonne pratique industrielle que nous avons choisi d'intégrer dès le début du projet.

Grâce à Portainer, nous pouvons accéder à une vue d'ensemble de l'état de chaque conteneur, ses logs, sa configuration réseau, et même redémarrer les services à distance. C'est un excellent outil de monitoring intégré au système.

4. Tâches à réaliser (cahier des charges)

Dès le début du projet, nous avons reçu un cahier des charges avec dedans différentes tâches à réaliser (*voir image n°3*)

Étudiant 2	<ul style="list-style-type: none"> • <i>Installation et paramétrage d'une passerelle LoRaWAN. Installation et paramétrage d'un serveur The Think Stack (The Think Network)</i> • Conception du format des messages à destination du SBEP. Conception et réalisation d'une classe de pilotage d'envoi de message via LoRaWAN en collaboration avec l'étudiant 1. • Conception et réalisation d'une classe de pilotage de l'automate à état fini de gestion de la station, de la BDD du SBEP, du pilotage du préleveur ; ce pilotage étant conditionné à une analyse des informations fournies par les capteurs. Fourniture des requêtes SQL d'interrogation et d'alimentation de la BDD utilisées dans le SBEP. • Paramétrage du serveur The Think Stack afin de communiquer au site Web les données. • Installation d'un serveur LAMP local avec création et installation de la BDD. Réception des mesures en coopération avec l'étudiant 1 et alimentation de la BDD. Conteneurisation du serveur LAMP en plusieurs conteneurs Docker (en collaboration avec l'étudiant 4-3) et mise en œuvre d'un orchestrateur (Docker Swarm/Portainer) • Paramétrage d'une clé GSM sur l'ordinateur cible. Configuration de l'environnement logiciel. Conception et réalisation d'une classe de gestion de la clé USB pour l'envoi des SMS d'alerte au technicien.
------------	--

Image n°3 : liste des tâches

5. Gestion du temps et du travail

Afin de nous organiser, nous avons pris du temps au début du projet afin de mettre en place plusieurs outils :

- **Github**

Ce github commun nous permet de faire un maximum de choses. Avec ce dernier, nous pouvons y déposer différents fichiers, nous avons de même créer un “planning” (*voir image n°4*). Ce planning nous permet de créer des cas utilisateurs (*voir image n°5*). Ces derniers nous permettent de diviser nos différentes tâches. De plus Github permet de créer un “calendrier” qui permet de placer nos différentes tâches dans le temps. (*voir image n°6*)

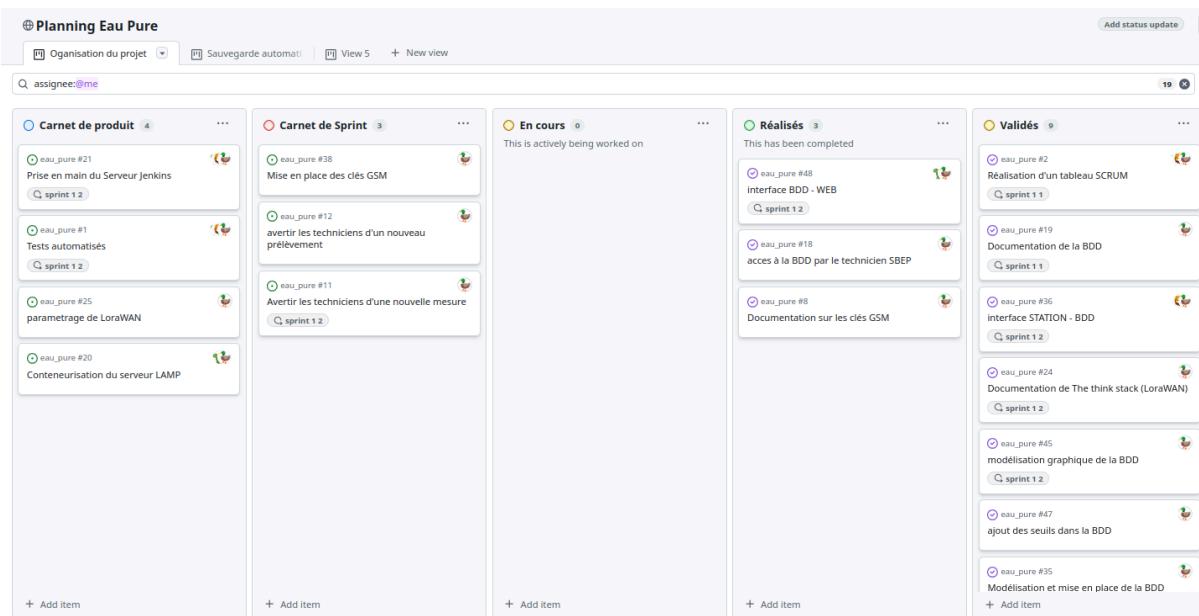


Image n°4 : Planning

• Sprint

Toujours dans le but d'optimiser notre temps, nous avons divisé notre calendrier en “sprint” : un sprint est une période de 2 semaines durant laquelle nous nous donnons plusieurs tâches à réaliser. Cela permet de voir notre avancée dans le temps.

• DailyScrum et PlanningPoker

Un DailyScrum est un petit oral. C'est une méthode qui à chaque nouvelle semaine nous permet d'énumérer les différentes tâches réalisées depuis le dernier DailyScrum, les difficultés rencontrées et les tâches à réaliser. Cela nous permet aussi de nous entraîner à l'oral.

Le planningPoker quant à lui consiste à mettre des points de difficultés à chaque utilisateur, cela nous permet de nous rendre compte de la difficulté du sprint.

6. Listes des tâches effectuées

• Création du dépôt GitHub :

Nous avons utilisé GitHub pour le suivi des versions, la gestion collaborative du code, et l'organisation du projet à l'aide des issues et d'un kanban intégré. Chaque branche représente une fonctionnalité distincte, facilitant le travail parallèle et la revue de code.

Dès le début du projet, J'ai créé un dépôt GitHub commun afin de centraliser et partager tous les codes sources, documentations et autres fichiers nécessaires au projet. Cela permet à toute l'équipe d'accéder facilement aux ressources et de

collaborer efficacement.

Nous avons utilisé GitHub pour le suivi des versions, la gestion collaborative du code, et l'organisation du projet à l'aide des issues et d'un kanban intégré. Chaque branche représente une fonctionnalité distincte, facilitant le travail parallèle et la revue de code.

- **Répartition des tâches avec le tableau Scrum :**

Avec mon équipe, nous avons mis en place un tableau Scrum pour organiser et répartir les tâches. Nous avons défini des cas utilisateur et les avons planifiés dans des sprints de deux semaines afin de nous répartir les tâches.

- **Modélisation de la base de données :**

J'ai conçu le modèle de la base de données en collaboration avec Nolan et notre professeur référent, M. Auffret. Nous avons défini les tables, les relations et les contraintes nécessaires pour stocker les données collectées par les stations de mesure et les données physico-chimiques.

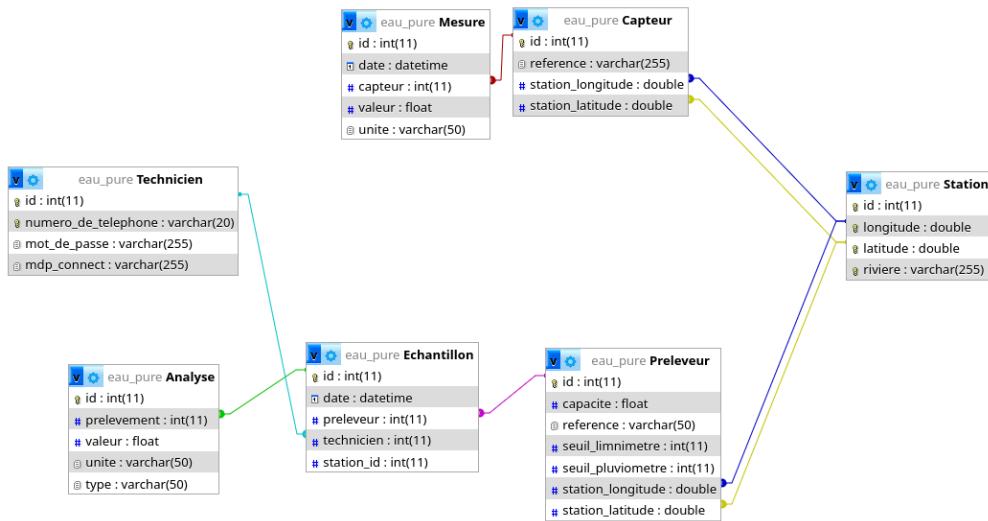


Image n°6 : Modélisation de la BDD

- **Mise en place de la base de données sur PhpMyAdmin :**

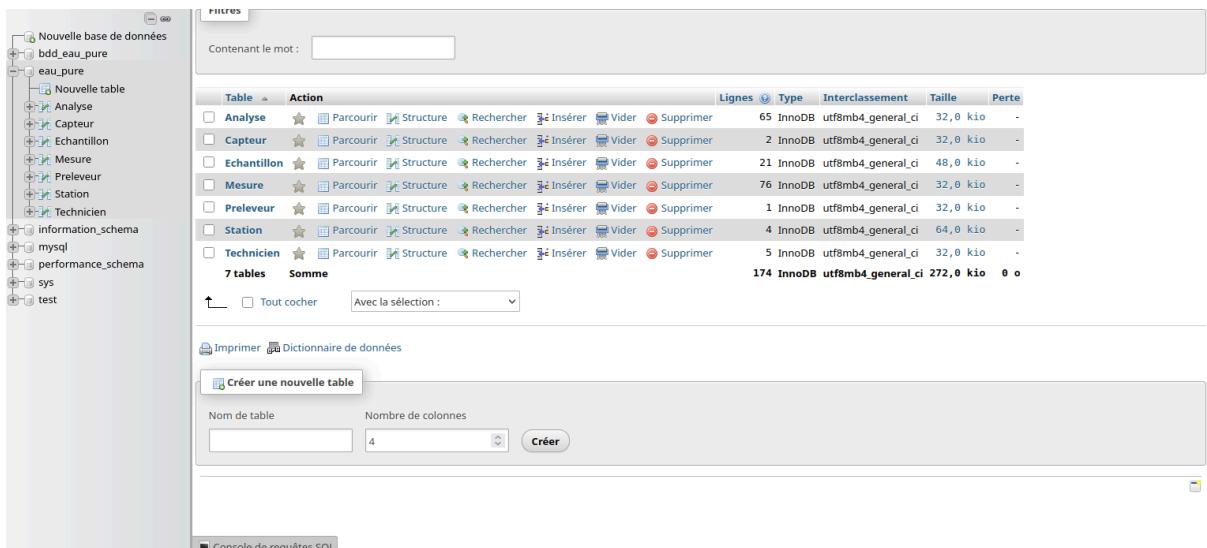
PhpMyAdmin est une interface web puissante permettant la gestion et l'administration des bases de données MySQL. Elle permet d'exécuter des requêtes SQL, de visualiser les données sous forme de tables, et de créer ou modifier des structures de manière intuitive. C'est un outil indispensable pour suivre les flux de

données provenant des stations. J'ai installé et configuré PhpMyAdmin pour gérer la base de données. J'ai également utilisé Docker pour déployer l'application, ce qui facilite la gestion et la maintenance de l'environnement de développement.

PhpMyAdmin est une interface web puissante permettant la gestion et l'administration des bases de données MySQL. Elle permet d'exécuter des requêtes SQL, de visualiser les données sous forme de tables, et de créer ou modifier des structures de manière intuitive. C'est un outil indispensable pour suivre les flux de données provenant des stations.

Docker permet d'exécuter chaque service (base de données, serveur web, interface Portainer...) dans un conteneur isolé. Cela simplifie la maintenance et rend l'environnement facilement duplicable d'une machine à une autre. C'est une bonne pratique industrielle que nous avons choisi d'intégrer dès le début du projet.

Grâce à Portainer, nous pouvons accéder à une vue d'ensemble de l'état de chaque conteneur, ses logs, sa configuration réseau, et même redémarrer les services à distance. C'est un excellent outil de monitoring intégré au système.



The screenshot shows the PhpMyAdmin interface. On the left, there's a tree view of the database structure under 'Nouvelle base de données'. The 'eau_pure' database is expanded, showing tables like 'Analyse', 'Capteur', 'Echantillon', 'Mesure', 'Preleveur', 'Station', and 'Technicien'. Below this, other databases like 'information_schema', 'mysql', 'performance_schema', 'sys', and 'test' are listed. The main area shows a table list titled 'Action'. It contains 7 tables: Analyse, Capteur, Echantillon, Mesure, Preleveur, Station, and Technicien. Each row has columns for 'Table', 'Action' (with icons for Parcourir, Structure, Rechercher, Insérer, Vider, Supprimer), 'Lignes', 'Type', 'Interclassement', 'Taille', and 'Perte'. A summary at the bottom right shows 174 rows, InnoDB type, utf8mb4_general_ci encoding, and 272,0 kio size. At the bottom, there are buttons for 'Tout cocher' and 'Avec la sélection'. Below the table list, there are buttons for 'Imprimer', 'Dictionnaire de données', and 'Créer une nouvelle table'. The 'Nom de table' field is empty, and the 'Nombre de colonnes' field is set to 4. A 'Créer' button is visible. At the very bottom, there's a 'Console de requêtes SQL' section.

Image n° 7 : Base de donnée sur PhpMyAdmin

- **Lancement du serveur Docker :**

Docker permet d'exécuter chaque service (base de données, serveur web, interface Portainer...) dans un conteneur isolé. Cela simplifie la maintenance et rend l'environnement facilement duplicable d'une machine à une autre. C'est une bonne pratique industrielle que nous avons choisi d'intégrer dès le début du projet. Grâce à Portainer, nous pouvons accéder à une vue d'ensemble de l'état de chaque conteneur, ses logs, sa configuration réseau, et même redémarrer les services à distance. C'est un excellent outil de monitoring intégré au système.

J'ai configuré et lancé un serveur Docker pour héberger la base de données et permettre à mon équipe d'envoyer leurs données et de faire des requêtes. Cela garantit un environnement de développement stable et reproductible.

Docker permet d'exécuter chaque service (base de données, serveur web, interface Portainer...) dans un conteneur isolé. Cela simplifie la maintenance et rend

l'environnement facilement duplicable d'une machine à une autre. C'est une bonne pratique industrielle que nous avons choisi d'intégrer dès le début du projet.

- **Interfaçage Base de donnée - WEB :**

Afin de faciliter la tâches à Nolan et de faciliter ses différents Code, j'ai mis en place une interface en PHP. Cette interface contient plusieurs fonctions qui permet en autre de faire des requêtes SQL. Cela évite en cas de modification de la base de données une modification de son code. Nolan à juste à appeler les fonctions pour les utiliser. J'ai ici coder en php car Nolan utilise des balises PHP.(*voir annexe n°3*).

- **Interfaçage Base de donnée - Station :**

Afin de faciliter la tâche à Mathys, j'ai mis aussi en place une interface cette fois-ci en python. Cette interface contient aussi des fonctions afin de faire des requêtes SQL. Comme pour l'interface WEB, cette interface est utile en cas de changement de la BDD. Ici je l'ai codé en Python car mathys utilise du python pour injecter les données. J'ai du alors modifier son programme "main" en y ajoutant les appels de fonctions. (*voir annexe n°4*).

- **Mise en place de la clé GSM :**

Dans le cahier des charges, je devais mettre en place une clé GSM. Cette clé nous permet d'envoyer des messages aux techniciens afin de les prévenir lors d'ajout d'une nouvelle mesure ou d'un nouveau prélèvement. La clé est une Huawei E3531 (*voir image n°8*). Pour cela, après m'être renseigné j'ai pu ajouter la clé sur notre raspberry. À l'aide d'un programme python, (*voir annexe n°5*) j'ai réussi à envoyer des messages à notre technicien. Il fallait juste faire un lien entre le main programmé par Mathys et mon code pour que lorsqu'il envoie les données des capteurs alors cela envois aussi un message à notre technicien.



Image n° 8 : Clé GSM

- **Ajout d'un conteneur Portainer :**

Grâce à Portainer, nous pouvons accéder à une vue d'ensemble de l'état de chaque conteneur, ses logs, sa configuration réseau, et même redémarrer les services à distance. C'est un excellent outil de monitoring intégré au système. Ce conteneur Portainer nous sert afin de visualiser graphiquement les autres conteneurs. Il offre une visualisation graphique et une meilleure compréhension des conteneurs (*voir image n° 8*). Grâce à Portainer, nous pouvons accéder à une vue d'ensemble de l'état de chaque conteneur, ses logs, sa configuration réseau, et même redémarrer les services à distance. C'est un excellent outil de monitoring intégré au système.

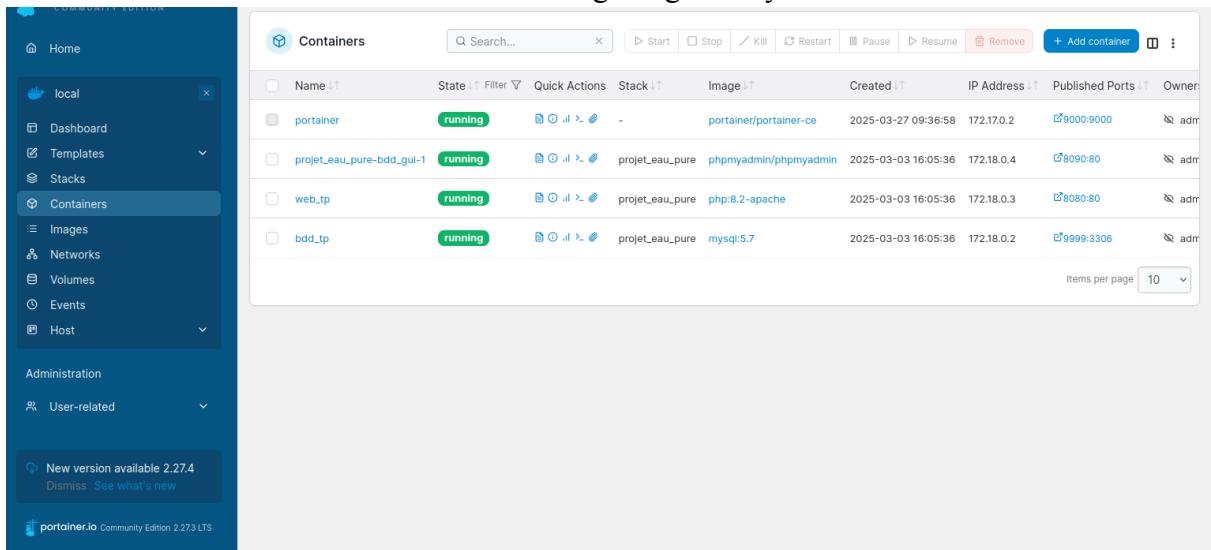


Image n° 8: Portainer

Grâce à Portainer, nous pouvons accéder à une vue d'ensemble de l'état de chaque conteneur, ses logs, sa configuration réseau, et même redémarrer les services à distance. C'est un excellent outil de monitoring intégré au système.

- **LoraWAN :**

Malheureusement, nous n'avons pas pu mettre en place le système de LoraWAN. Cependant, j'ai pu tout de même utiliser un petit peu Lora et notamment la gateway the things GATEWAY (The things Network)



Image n°9 : The Things GATEWAY

Avec ceci, j'ai pu seulement accéder à une interface graphique mais à cause de soucis avec la connexion entre la raspberry et un autre outil utilisé pour LoraWAN. Nous n'avons pas pu l'ajouter à notre projet. Pour résoudre ce problème, nous travaillons toujours en local.

7. Description du Matériel/Logiciel

1. GitHub :

Nous avons utilisé GitHub pour le suivi des versions, la gestion collaborative du code, et l'organisation du projet à l'aide des issues et d'un kanban intégré. Chaque branche représente une fonctionnalité distincte, facilitant le travail parallèle et la revue de code.

GitHub : Partage de code et documentation, version installée : service en ligne toujours à jour.

Nous avons utilisé GitHub pour le suivi des versions, la gestion collaborative du code, et l'organisation du projet à l'aide des issues et d'un kanban intégré. Chaque branche représente une fonctionnalité distincte, facilitant le travail parallèle et la revue de code.

Procédure d'installation : Service en ligne

Raison du choix : Outil utilisé en cours.

2. **PhpMyAdmin :**

PhpMyAdmin est une interface web puissante permettant la gestion et l'administration des bases de données MySQL. Elle permet d'exécuter des requêtes SQL, de visualiser les données sous forme de tables, et de créer ou modifier des structures de manière intuitive. C'est un outil indispensable pour suivre les flux de données provenant des stations.

PhpMyAdmin : Gestion de la base de données, version installée : 5.2.2.

PhpMyAdmin est une interface web puissante permettant la gestion et l'administration des bases de données MySQL. Elle permet d'exécuter des requêtes SQL, de visualiser les données sous forme de tables, et de créer ou modifier des structures de manière intuitive. C'est un outil indispensable pour suivre les flux de données provenant des stations.

Procédure d'installation : Installer via Docker.

Docker permet d'exécuter chaque service (base de données, serveur web, interface Portainer...) dans un conteneur isolé. Cela simplifie la maintenance et rend l'environnement facilement duplicable d'une machine à une autre. C'est une bonne pratique industrielle que nous avons choisi d'intégrer dès le début du projet.

Grâce à Portainer, nous pouvons accéder à une vue d'ensemble de l'état de chaque conteneur, ses logs, sa configuration réseau, et même redémarrer les services à distance. C'est un excellent outil de monitoring intégré au système.

Raison du choix : Interface web pour gérer les bases de données utilisées en cours.

Ligne de commande pour vérifier l'installation : docker ps

3. **Docker :**

Docker permet d'exécuter chaque service (base de données, serveur web, interface Portainer...) dans un conteneur isolé. Cela simplifie la maintenance et rend l'environnement facilement duplicable d'une machine à une autre. C'est une bonne pratique industrielle que nous avons choisi d'intégrer dès le début du projet. Grâce à Portainer, nous pouvons accéder à une vue d'ensemble de l'état de chaque conteneur, ses logs, sa configuration réseau, et même redémarrer les services à distance. C'est un excellent outil de monitoring intégré au système.

Docker : Conteneurisation des applications, version installée : 26.1.4.

Docker permet d'exécuter chaque service (base de données, serveur web, interface Portainer...) dans un conteneur isolé. Cela simplifie la maintenance et rend l'environnement facilement duplicable d'une machine à une autre. C'est une bonne pratique industrielle que nous avons choisi d'intégrer dès le début du projet.

Grâce à Portainer, nous pouvons accéder à une vue d'ensemble de l'état de chaque conteneur, ses logs, sa configuration réseau, et même redémarrer les services à distance. C'est un excellent outil de monitoring intégré au système.

Procédure d'installation : Outil déjà présent sur l'ordinateur.

Raison du choix : Facilite le déploiement et la gestion des applications déjà utilisées en cours.

Ligne de commande pour vérifier l'installation : docker --version

4. Portainer :

Grâce à Portainer, nous pouvons accéder à une vue d'ensemble de l'état de chaque conteneur, ses logs, sa configuration réseau, et même redémarrer les services à distance. C'est un excellent outil de monitoring intégré au système.

Portainer : Visualisation graphique des différents conteneurs

Grâce à Portainer, nous pouvons accéder à une vue d'ensemble de l'état de chaque conteneur, ses logs, sa configuration réseau, et même redémarrer les services à distance. C'est un excellent outil de monitoring intégré au système.

Procédure d'installation : Service en ligne

Raison du choix : Portainer offre une meilleure visualisation graphique et surtout une meilleure compréhension des conteneurs.. Grâce à Portainer, nous pouvons accéder à une vue d'ensemble de l'état de chaque conteneur, ses logs, sa configuration réseau, et même redémarrer les services à distance. C'est un excellent outil de monitoring intégré au système.

8. Politique de protection et de sécurité des données

Afin de respecter les contraintes de cybersécurité, plusieurs mesures ont été mises en place : utilisation de requêtes préparées dans les scripts PHP et Python via l'interface pour éviter les injections SQL, restriction des accès réseau aux conteneurs critiques, et gestion des utilisateurs via des mots de passe sécurisés. Un hashage des mots de passe. De plus, l'environnement étant localisé dans un réseau fermé (DMZ interne au SBEP), le risque d'attaque distante est fortement réduit

Etudiant n°3 - Nolan CONAN.....	23
1. Analyse / Aspect fonctionnel.....	23
1.1. Présentation de l'architecture logicielle et matérielle.....	23
1.2. Diagramme de présentation de l'architecture logicielle et matérielle.....	24
1.3. Diagramme des exigences.....	25
1.4. Liste des tâches.....	26
1.5. Moyens techniques et outils mis à disposition.....	27
1.6. Répartition des rôles et responsabilités.....	27
1.7. Représentation fonctionnelle des tâches par étudiant.....	28
2. Travail réalisé.....	29
2.1. Planification et organisation des tâches.....	29
2.2. Mise en place de l'authentification sécurisée.....	29
2.3. Ajout de la gestion des utilisateurs et renforcement de la sécurité.....	30
2.4. Développement du formulaire de saisie des données physico-chimiques.....	31
3. Création de l'application React.....	31
3.1. Développement de l'application de visualisation des données physico-chimiques.....	31
3.2. Installation de React.....	31
3.3. Structure du projet.....	32
3.4. Mise en place du système d'affichage des graphiques avec React et un serveur Node.js.....	32
3.5. Mise en place du serveur backend (Node.js + Express).....	33
3.6. Lancement du serveur backend.....	34
3.7. Installation des bibliothèques nécessaires à l'affichage des graphiques dans React.....	34
3.8. Lien entre le frontend et le backend.....	34
3.9. Développement de l'interface de visualisation des données.....	35
3.10. Installation de l'environnement de travail.....	35
3.11. Création du backend : faire le lien avec la base de données.....	35
3.12. Développement du frontend : l'affichage et les interactions.....	35
4. Diagramme de Gantt.....	36
5. Environnement technique / Contraintes.....	36
5.1. Choix de l'environnement de développement.....	36
5.2. Librairies et outils tiers utilisés.....	37
5.3. Contraintes imposées par le client.....	37
6. Architecture matérielle / Caractéristiques et choix.....	38
6.1. Présentation de l'architecture matérielle du projet.....	38
6.2. Architecture logicielle du projet.....	38
6.3. Justification des choix matériels.....	39
7. Spécifications techniques.....	39
7.1. Fonctionnement des capteurs et actionneurs.....	39
7.2. Mise en œuvre des capteurs.....	39
7.3. Communication et format des données.....	40
7.4. Interfaces entre les composants.....	40
7.5. Politique de protection des données.....	41
8. Travail collaboratif.....	41
8.1. Mise en place de l'espace collaboratif.....	41
8.2. Utilisation et démonstration de l'espace collaboratif.....	41
8.3. Comparaison entre le planning prévisionnel et le planning effectif.....	42
8.4. Justification des écarts entre le temps prévu et le temps réalisé.....	42
9. Gestion des versions.....	42
9.1. Liste et présentation des différents reportings et versionings.....	42
9.2. Présentation de l'espace collaboratif pour les versions de livraison.....	42
9.3. Mise en place de l'outil de versionnement et documentation des composants.....	42
9.4. Intégration et packaging de la dernière livraison du projet.....	43
10. Aspects liés à la sécurité / robustesse des éléments mis en œuvre.....	43
10.1. Liste des versions des matériels et logiciels(revoir).....	43
10.2. Les outils utilisés.....	43
10.3. Vérification de la mise à jour des logiciels et pilotes.....	45
10.4. Configuration des éléments matériels et logiciels pour une résistance aux attaques.....	45
10.5. Prise en compte des enjeux de cybersécurité (failles de sécurité, signature, certificats, etc.) et des mises à jour.....	46

Etudiant n°3 - Nolan CONAN

1. Analyse / Aspect fonctionnel

1.1. Présentation de l'architecture logicielle et matérielle

L'objectif principal de ce projet est d'automatiser la surveillance des eaux de surface, et de publier en ligne des bulletins interactifs portant sur les aspects quantitatifs et qualitatifs de ces eaux. Cette initiative est portée par le Service Biodiversité Eau et Paysage (SBEP), une entité rattachée à la Direction Régionale de l'Environnement, de l'Aménagement et du Logement (DREAL). Le SBEP œuvre à l'amélioration de la biodiversité et de la qualité de vie par une évaluation régulière de la qualité des cours d'eau.

Dans le cadre du projet, des stations hydrologiques ont été déployées le long de certains cours d'eau. Chaque station est équipée d'un Raspberry Pi, de capteurs (un limnimètre pour mesurer la hauteur d'eau, un pluviomètre pour les précipitations), d'un module de communication LoRaWan, d'une source d'alimentation autonome, et d'un actionneur simulé (remplacé par un bouchon de test pour des raisons de coût et de simplification).

Ces stations communiquent via un réseau LoRaWan basse consommation avec une passerelle, qui transmet ensuite les données vers le serveur Web du SBEP à travers Internet. Les données mesurées sont centralisées, historisées et exploitées sur cette plateforme.

Le prélèvement automatisé est déclenché en fonction des conditions hydrologiques et météorologiques. Lorsqu'un échantillon est prélevé, une alerte SMS est automatiquement envoyée à un technicien pour qu'il assure le transport vers un laboratoire d'analyse chimique. Les résultats de ces analyses sont ensuite conservés dans la base de données du serveur, en vue de leur publication.

L'ensemble de l'infrastructure repose sur un réseau local interne au bâtiment du SBEP, où tous les équipements informatiques sont interconnectés. Le serveur Web, la base de données ainsi que les équipements réseaux sont installés dans la salle informatique du service, qui regroupe quatre unités.

1.2. Diagramme de présentation de l'architecture logicielle et matérielle

Ce diagramme de déploiement UML représente l'architecture matérielle et logicielle du système automatisé de surveillance hydrologique développé pour le SBEP. Il met en évidence les différents noeuds du système, à commencer par la station hydrologique composée de capteurs (pluviomètre et limnimètre), d'un Raspberry Pi et d'un module LoRaWAN, chargé de transmettre les données vers le serveur **The Things Stack** via une passerelle LoRaWAN.

Les données sont ensuite redirigées vers un **serveur LAMP local**, hébergeant une **base de données MySQL** ainsi qu'un site web permettant aux utilisateurs de se connecter, consulter les mesures via une **carte interactive**, ou encore saisir des résultats d'analyse dans un **formulaire physico-chimique**.

Enfin, le diagramme montre que le SBEP peut accéder aux données collectées pour les analyser, tandis que les utilisateurs accèdent aux services via un navigateur.

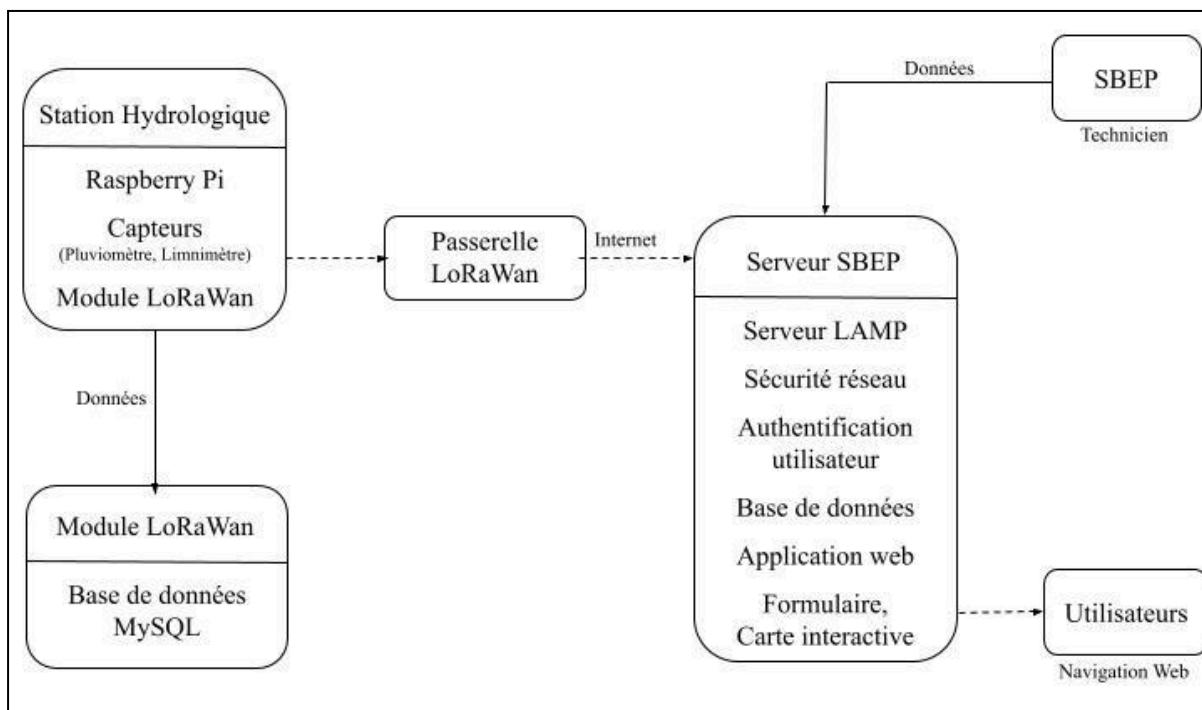


Figure 1 : Diagramme de déploiement UML de l'architecture du système de surveillance hydrologique

1.3. Diagramme des exigences

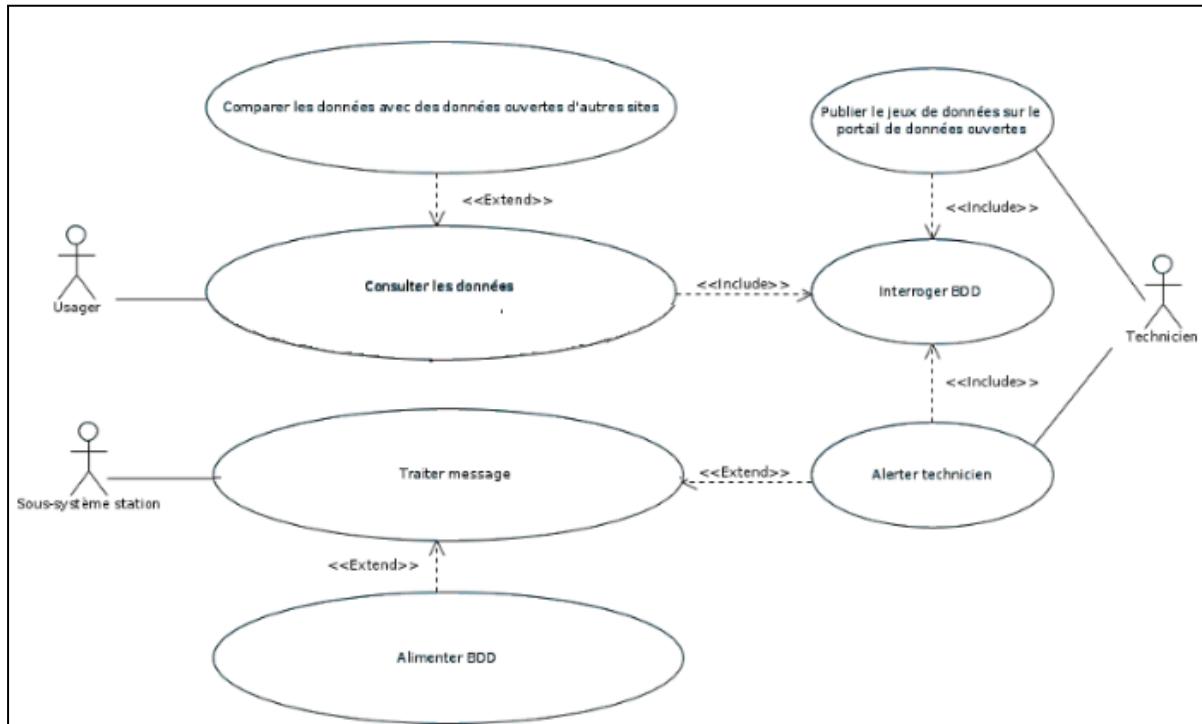


Figure 2 : Diagramme de cas d'utilisation du sous-système SBEP

Le cas d'utilisation “**Traiter message**” est déclenché par tout message reçu du sous-système station. En fonction des données récupérées le cas d'utilisation “**Alimenter BDD**” est déclenché s'il s'agit de mesures à enregistrer dans la BDD ou bien c'est le cas d'utilisation “**Alerter technicien**” s'il s'agit d'une alerte. Celui-ci déclenche à son tour le cas d'utilisation “**Interroger la BDD**” pour connaître le numéro de téléphone du technicien c'est-à-dire l'opérateur de prélèvement de l'échantillon d'eau. Une fois la BDD alimentée, le technicien peut partager les données relatives à la mesure sur le portail de données ouvertes du “SIE” afin d'être ré-exploitées au niveau national, c'est le cas d'utilisation “**Publier le jeu de données sur le portail de données ouvertes**”. Le cas d'utilisation “**Consulter les données**” permet de consulter à partir d'un navigateur Internet, les dernières données et les derniers prélèvements effectués. La visualisation s'effectue de manière géolocalisée en pointant sur une carte la station choisie, ce qui provoque l'affichage des informations souhaitées. Ce cas déclenche le cas d'utilisation “**Interroger la BDD**” pour obtenir les informations nécessaires. Le cas d'utilisation “**Comparer les données avec les données ouvertes d'autres sites**” exploite des données publiées sur le portail de données ouvertes et permet de comparer les mesures de la station de prélèvement avec celles d'autres stations en France.

1.4. Liste des tâches

Étudiant n°3	<ul style="list-style-type: none"> • <i>Conception et réalisation du site WEB local en utilisant le framework JavaScript React.</i> • <i>Présentation des données de mesures et d'alerte de manière graphique et géolocalisée en utilisant le mécanisme OpenStreetMap / Leaflet.</i> • <i>Conception d'un formulaire sécurisé permettant d'alimenter la base de données avec les résultats de l'analyse physico-chimique</i> • <i>Exploitation des données et API ouvertes data.gouv.fr et api.gouv.fr.</i> • <i>Automatisation du formatage des données et de leurs validation contre un modèle de données (schéma) en vue de leur publication par le technicien sur le portail de données ouvertes.</i> • <i>En collaboration avec l'étudiant 2 : conteneurisation du serveur LAMP en plusieurs conteneurs Docker</i>
--------------	---

1.5. Moyens techniques et outils mis à disposition

Pour la réalisation de la partie web, j'ai utilisé GitHub comme espace collaboratif pour organiser le travail et suivre l'avancement du projet. Les tâches ont été découpées et estimées à l'aide de la méthode du Planning Poker. Pour le développement, j'ai utilisé Geany comme éditeur de texte et PHP, HTML, CSS pour créer le formulaire de connexion sécurisé. La partie carte interactive a été développée en React et JavaScript, avec une forte dépendance à des ressources en ligne pour apprendre ces technologies. Le projet utilise également un serveur LAMP, mis en place par mon camarade responsable de la base de données, auquel j'accède via son adresse IP locale.

1.6. Répartition des rôles et responsabilités

L'étudiant 1 est chargé de l'installation et de l'intégration des capteurs (pluviomètre à godets, limnimètre) et de leur raccordement à la carte Raspberry Pi. Il est également responsable de l'alimentation autonome du système via un panneau solaire et une batterie. Son périmètre inclut l'étalonnage et l'interface des capteurs, ainsi que la gestion des signaux (filtrage, conversion, amplification si nécessaire). Il conçoit les classes de gestion des capteurs et assure la transmission des mesures via LoRaWAN en coopération avec l'étudiant 2. Il prend également en charge la conception de l'architecture réseau, la simulation et la validation de celle-ci, ainsi que la mise en œuvre de la configuration des équipements réseaux.

L'étudiant 2 est responsable de l'installation et du paramétrage de la passerelle LoRaWAN et du serveur The Think Stack. Il conçoit le format des messages à envoyer au SBEP et développe des classes pour le pilotage de l'envoi des messages via LoRaWAN. Son périmètre inclut également la gestion du serveur de base de données (LAMP) et la réception des mesures pour alimenter la base de données du SBEP. En collaboration avec l'étudiant 3, il assure la conteneurisation du serveur LAMP en plusieurs conteneurs Docker et participe à l'orchestration via Docker Swarm/Portainer. Il est aussi responsable de la configuration de la clé GSM pour l'envoi des alertes SMS au technicien.

L'étudiant 3, qui s'occupe de la partie web, est chargé de la conception et du développement du site local en utilisant React. Il met en place une interface pour la présentation des données de mesures et des alertes de manière géolocalisée à l'aide d'OpenStreetMap/Leaflet. Il développe également un formulaire sécurisé pour l'entrée des résultats d'analyse physico-chimique dans la base de données. Son travail inclut l'exploitation des API ouvertes pour automatiser le formatage des données avant leur publication. En collaboration avec l'étudiant 2, il participe à la conteneurisation du serveur LAMP et à la gestion des interactions avec la base de données.

1.7. Représentation fonctionnelle des tâches par étudiant

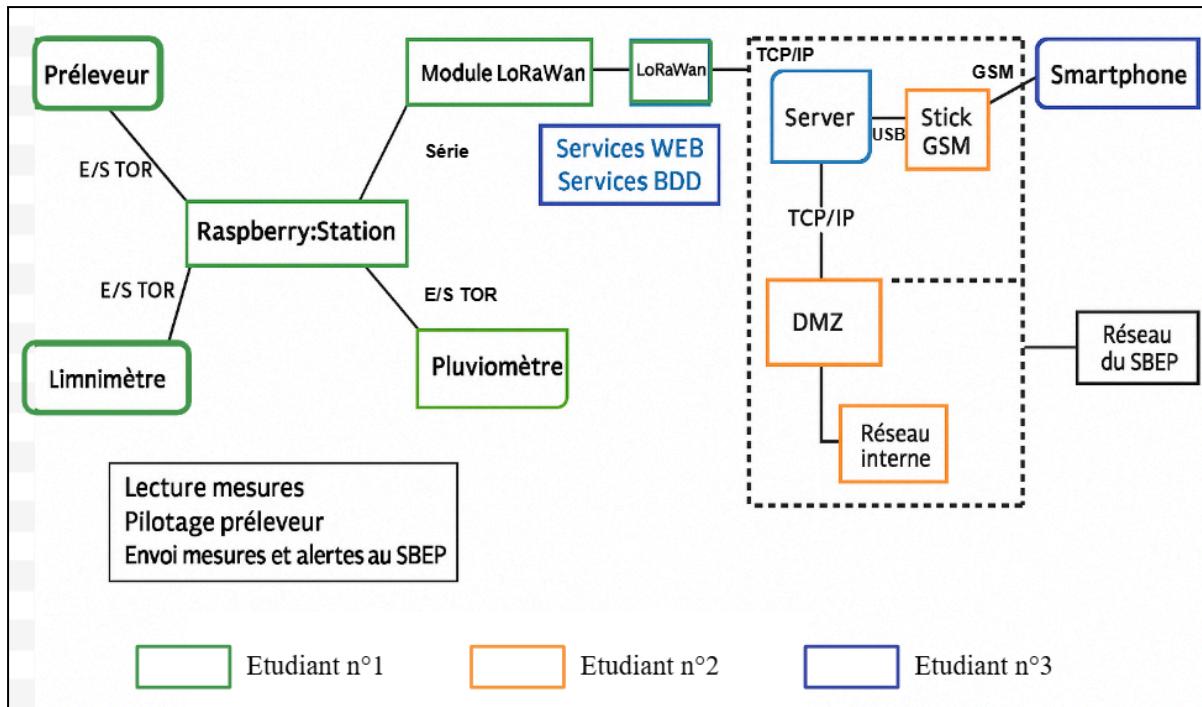


Figure 3 : Répartition fonctionnelle des membres de l'équipe – Schéma global du système

Le schéma ci-dessous illustre l'architecture générale du système de surveillance automatisée des eaux de surface, ainsi que le périmètre d'intervention de chacun des membres de l'équipe, identifié par un code couleur.

Chaque membre s'est vu attribuer des responsabilités spécifiques dans la mise en œuvre du projet :

- **Candidat 1 (capteurs et réseau) :** Mise en place et gestion des capteurs (pluviomètre, limnimètre, préleur), configuration du Raspberry Pi et du module LoRaWan.
- **Candidat 2 (base de données et infrastructure) :** Configuration de la passerelle LoRaWan, du serveur, de la base de données, du réseau (DMZ, stick GSM), et de l'infrastructure via Docker.
- **Candidat 3 (interface web) :** Développement de la partie web : carte interactive, affichage des données, formulaire de connexion, intégration avec l'API et services BDD.

Ce diagramme permet de visualiser clairement la répartition des rôles dans le projet, en lien direct avec l'architecture matérielle et logicielle mise en œuvre.

2. Travail réalisé

2.1. Planification et organisation des tâches

Pour ce projet, j'ai commencé par insérer les différentes tâches à réaliser à partir du cahier des charges dans le tableau Scrum créé sur GitHub avec mes camarades. Le cahier des charges comprend diverses tâches plus ou moins imposantes à réaliser. Il était donc essentiel de les diviser au maximum afin de progresser judicieusement sans oublier certaines étapes qui pourraient nous retarder.

Une fois les tâches planifiées, nous avons évalué leurs difficultés à l'aide du Planning Poker. Cette méthode nous a permis d'estimer l'effort nécessaire pour chaque tâche en attribuant une valeur, puis en ajustant ces estimations grâce aux discussions jusqu'à atteindre un consensus.

Figure n°4 : Réalisation d'un tableau Scrum créé sur GitHub à partir du cahier des charges

Figure n°5 : Évaluation de la difficulté des tâches à l'aide du Planning Poker

2.2. Mise en place de l'authentification sécurisée

J'ai décidé de commencer par la conception d'un formulaire sécurisé permettant d'alimenter la base de données avec les résultats de l'analyse physico-chimique. Cela me semblait plus cohérent pour l'ensemble du projet. Cette phase consiste à réaliser un formulaire de connexion sécurisé afin que le technicien du SBEP (Service Biodiversité Eau et Paysage), qui a prélevé l'échantillon d'eau, puisse, une fois l'analyse effectuée, entrer les données physico-chimiques dans ce formulaire.

J'ai tout d'abord implémenté un système d'authentification sécurisé. Pour cela, j'ai mis en place un mécanisme de hachage et de salage des mots de passe afin de protéger au mieux les informations des utilisateurs. Le hachage transforme le mot de passe en une suite de caractères illisible et irréversible grâce à une fonction cryptographique "password_hash()". Cependant, une simple empreinte hachée peut être vulnérable aux attaques par rainbow table, qui permettent aux attaquants d'utiliser des bases de données précalculées pour retrouver un mot de passe à partir de son empreinte. Pour éviter cela, j'ai intégré un salage, c'est-à-dire l'ajout d'une chaîne aléatoire unique à chaque mot de passe avant de le hacher. Ainsi, même si deux utilisateurs ont le même mot de passe, leurs empreintes hachées seront différentes, rendant les attaques par rainbow table inefficaces.

Ensuite, j'ai conçu un formulaire de connexion (index.html) dans lequel l'utilisateur doit renseigner son numéro de téléphone et son mot de passe. Le traitement de l'authentification est géré dans "connexion.php", qui vérifie si les identifiants correspondent à ceux stockés dans la base de données. Pour renforcer la sécurité, j'ai utilisé la fonction "password_verify()" afin de comparer le mot de passe saisi avec celui haché.

Une fois connecté, l'utilisateur est automatiquement redirigé vers la page "données_physico_chimiques.html", où il peut renseigner les résultats des analyses effectuées sur l'échantillon d'eau.

Lors de la mise en place de l'authentification, j'ai rencontré des problèmes de connexion entre mon formulaire et la base de données, causés par Fortinet, qui bloquait les échanges entre mon application et le serveur MySQL. Cette contrainte m'empêchait de valider les identifiants et donc d'accéder aux pages suivantes. Pour contourner cette difficulté, mon professeur référent m'a fourni un serveur LAMP, ce qui m'a permis de travailler en local. Une fois MySQL installé sur mon poste, j'ai pu interagir correctement avec la base de données. J'ai également utilisé la commande suivante dans le terminal pour accéder à mon serveur web via Docker et exécuter les scripts MySQL :

```
sudo docker exec -it web_tp sh
#docker-php-ext-install mysqli
#apachectl restart
```

Figure 6 : Lignes de commande pour accéder à mon serveur web via Docker et exécuter les scripts MySQL

Mot de passe haché : \$2y\$10\$1360lzCjiBcMcueF1MNxQ.bRwmh5yx6pAgSz3BtAEUSdvkBe9mEXe

Figure n°7 : Réalisation d'un programme permettant de hacher un mot de passe

[Figure n°8 : Réalisation d'un formulaire de connexion](#)

2.3. Ajout de la gestion des utilisateurs et renforcement de la sécurité

Une fois cette première partie réalisée, il m'a été demandé d'ajouter une nouvelle fonctionnalité : un bouton permettant à l'administrateur d'ajouter, modifier ou supprimer un utilisateur. Pour cela, j'ai intégré des fenêtres pop-up directement dans mon formulaire de connexion afin de gérer ces actions. Ces pop-up demandent un numéro de téléphone et un mot de passe en fonction de l'action choisie (ajout, modification ou suppression d'un utilisateur).

À cette occasion, j'ai également mis en place une politique de mot de passe pour renforcer la sécurité. J'ai imposé un minimum de 8 caractères, dont une majuscule et un caractère spécial, afin de limiter les risques d'attaques par force brute. De plus, j'ai ajouté des messages d'erreur adaptés pour guider l'utilisateur en cas d'échec d'authentification, tout en restant concis pour ne pas donner d'indices aux potentiels attaquants. Cela permet d'améliorer à la fois la sécurité et l'expérience utilisateur en rendant l'interface plus professionnelle.

[Figure n°9 : Pop-up de connexion administrateur](#)

[Figure n°10 : Pop-up de gestion des utilisateurs](#)

Figure n°11 : Réalisation des fenêtres pop-up pour ajouter, modifier ou supprimer un utilisateur

2.4. Développement du formulaire de saisie des données physico-chimiques

Mise en place d'un formulaire sécurisé pour la saisie des données physico-chimiques. Après avoir sécurisé l'authentification des utilisateurs, j'ai développé un formulaire de saisie des données physico-chimiques permettant aux techniciens du SBEP d'entrer les résultats des analyses d'eau effectuées en laboratoire. Ce formulaire intègre plusieurs paramètres essentiels tels que le pH, la conductivité électrique, la turbidité, l'oxygène dissous et la demande chimique en oxygène (DCO). Pour faciliter la saisie, chaque champ est accompagné d'un texte explicatif détaillant les valeurs attendues et leurs interprétations.

Afin d'améliorer la précision des enregistrements, j'ai ajouté deux nouveaux champs : l'un pour la localisation de la rivière et l'autre pour l'horodatage du prélèvement. Cette modification permet d'associer chaque mesure à un point géographique précis et à une date exacte, garantissant ainsi une meilleure traçabilité des données. J'ai également veillé à ce que chaque saisie soit liée à l'ID unique de la rivière sélectionnée afin d'assurer une parfaite correspondance entre les données et le lieu de prélèvement.

Pour assurer la fiabilité et la sécurité de ce système, j'ai utilisé des requêtes SQL préparées afin de prévenir les attaques par injection SQL. Ce type d'attaque consiste à insérer du code malveillant dans un champ de saisie pour manipuler la base de données. En utilisant cette méthode sécurisée, les valeurs saisies sont traitées comme de simples données et non comme des commandes SQL exécutables, ce qui empêche toute tentative de piratage. Grâce à cette approche, les informations collectées restent intactes et exploitables en toute sécurité.

Figure n°12 : Réalisation du formulaire pour la saisie des données physico-chimiques

3. Création de l'application React

3.1. Développement de l'application de visualisation des données physico-chimiques

Dans le cadre de notre projet de surveillance des eaux de surface, j'ai été chargé de la création d'une interface web permettant d'afficher les données physico-chimiques sous forme de graphiques. Pour cela, j'ai utilisé React, une bibliothèque JavaScript permettant de construire des interfaces utilisateurs dynamiques. Le choix de React m'a été imposé dans le cahier des charges, mais j'ai appris à l'utiliser en autonomie grâce à des recherches personnelles, car cette technologie n'avait pas été abordée en cours.

3.2. Installation de React

Avant de commencer le développement, j'ai installé React en ligne de commande à l'aide de l'outil "create-react-app", qui permet de créer rapidement une base de projet prête à l'emploi.

Voici les commandes utilisées :

```
npx create-react-app eau_pure
cd eau_pure
```

Cette commande a généré un dossier nommé "eau_pure", contenant la structure de base d'un projet React. À l'intérieur de ce dossier, on trouve notamment :

- un dossier "public" : il contient les fichiers statiques accessibles directement par le navigateur (comme "index.html"),
- un dossier "src" : c'est ici que l'on développe les composants React. J'y ai ajouté mon fichier "donnees_physco_chimiques.jsx",
- un fichier "package.json" : il gère les bibliothèques nécessaires au fonctionnement du projet,
- et d'autres fichiers de configuration utiles pour démarrer rapidement le projet.

Une fois dans ce dossier, j'ai lancé l'application avec :

```
npm start
```

Cette commande démarre un serveur de développement et ouvre automatiquement l'application dans un navigateur à l'adresse "http://localhost:3000".

3.3. Structure du projet

J'ai organisé les fichiers de cette manière pour bien séparer le frontend (React) du backend (Node.js) :

```
/eau_pure
  /src
    App.js
    donnees_physco_chimiques.jsx
  package.json
  ...
  /backend
    server.js
    package.json
  ...
```

→ “donnees_physco_chimiques.jsx” est le fichier que j’ai ajouté pour gérer l’affichage des données physico-chimiques sous forme de graphiques.

3.4. Mise en place du système d’affichage des graphiques avec React et un serveur Node.js

Après l’installation de React, j’ai développé une fonctionnalité permettant d’afficher les données physico-chimiques sous forme de graphiques. Pour cela, j’ai créé un fichier nommé “donnes_physco_chimiques.jsx”, placé dans le dossier “src” du projet React. Ce fichier contient un composant React, c’est-à-dire une partie de l’interface qui se charge de créer et d’afficher les graphiques.

Mais pour alimenter ces graphiques avec des données en temps réel, j’ai mis en place un serveur backend, c’est-à-dire un programme qui va chercher les données dans la base de données et les envoyer au frontend (la partie visible de l’application). Ce backend a été développé avec Node.js et Express, deux outils JavaScript utilisés pour créer facilement des serveurs web.

3.5. Mise en place du serveur backend (Node.js + Express)

Pour que l’interface puisse récupérer les données de la base MySQL, j’ai créé un serveur backend en Node.js.

Voici les différentes étapes de configuration :

1% Installation de Node.js

Avant de pouvoir créer le serveur backend avec Node.js, j’ai dû installer manuellement l’environnement Node.js sur ma machine. Pour cela, je me suis rendu sur le site officiel “nodejs.org” et j’ai téléchargé la version adaptée à mon système : “node-v22.14.0-linux-x64.tar.xz”. Une fois le fichier téléchargé, je l’ai extrait et installé sans utiliser le terminal.

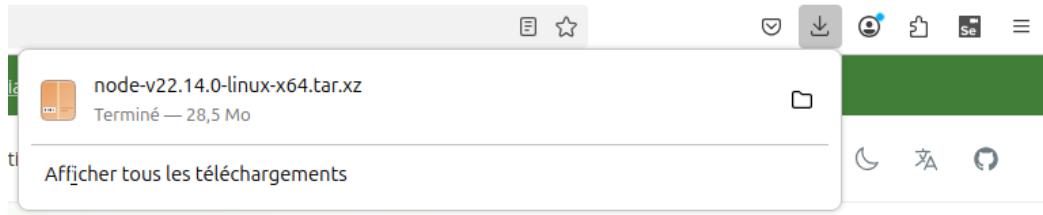


Figure n°13 : Téléchargement du paquet node.js

2% Création du dossier backend

```
mkdir backend
cd backend
```

3°/ Initialisation du projet Node.js

```
npm init -y
```

Cette commande crée un fichier “package.json” qui permettra de gérer les dépendances nécessaires au bon fonctionnement du serveur.

4°/ Installation des dépendances nécessaires

```
npm install express mysql cors
```

- express : permet de créer un serveur HTTP,
- mysql : permet de se connecter à la base de données MySQL,
- cors : pour autoriser les échanges entre le serveur backend et l’application frontend.

5°/ Création du fichier “server.js”

Dans le dossier “server”, j’ai créé un fichier nommé server.js. Ce fichier contient le code qui :

- lance un serveur sur le port 3001,
- se connecte à la base de données,
- exécute des requêtes SQL,
- renvoie les résultats au frontend sous forme de réponse JSON.

3.6. Lancement du serveur backend

Une fois le fichier “server.js” configuré, j’ai lancé le serveur avec la commande suivante :

```
node server.js
```

Si tout est bien en place, un message s’affiche dans le terminal pour indiquer que le serveur est en écoute sur le port 3001.

3.7. Installation des bibliothèques nécessaires à l’affichage des graphiques dans React

Dans le répertoire du projet React (eau_pure), j’ai installé les bibliothèques qui permettent :

- de créer les graphiques,
- et de communiquer avec le backend pour récupérer les données.

```
npm install axios chart.js react-chartjs-2
```

- axios : permet d’envoyer des requêtes HTTP pour récupérer les données du serveur,

- chart.js : bibliothèque utilisée pour générer les graphiques,
- react-chartjs-2 : permet d'utiliser "chart.js" facilement avec React.

3.8. Lien entre le frontend et le backend

Dans le fichier "donnees_physco_chimiques.jsx", j'ai codé un composant React qui envoie automatiquement une requête vers l'API backend à l'adresse suivante :

http://localhost:3001/api/data

Le serveur répond avec les données récupérées depuis la base, et celles-ci sont ensuite affichées dans des graphiques.

L'application React affiche en temps réel les données physico-chimiques collectées. Grâce à cette architecture, il est possible de visualiser facilement les paramètres des cours d'eau surveillés, avec une interface fluide et accessible via un simple navigateur.

3.9. Développement de l'interface de visualisation des données

Dans ce projet de surveillance des eaux de surface, ma mission principale a été de créer une interface web, c'est-à-dire une page que l'on peut consulter dans un navigateur internet, qui permet de visualiser les données recueillies par les capteurs placés dans les rivières. Cette interface permet de consulter les données physico-chimiques (comme le pH ou la turbidité de l'eau), en temps réel et de manière simple. Pour cela, j'ai travaillé à la fois sur ce que l'utilisateur voit (ce qu'on appelle le frontend) et sur ce qui se passe derrière (appelé backend).

3.10. Installation de l'environnement de travail

Avant de pouvoir commencer à coder, j'ai dû installer les outils nécessaires. J'ai d'abord téléchargé Node.js depuis son site officiel. Node.js est un programme qui permet d'exécuter du JavaScript sur un ordinateur, même en dehors d'un navigateur. C'est ce qui m'a permis de créer ce qu'on appelle un petit serveur, qui va jouer le rôle d'intermédiaire entre les données stockées et l'affichage de celles-ci sur la page web.

Une fois Node.js installé, j'ai pu utiliser un outil qui s'appelle Create React App, qui prépare automatiquement un projet de base pour utiliser React. React est une bibliothèque JavaScript (c'est-à-dire un ensemble d'outils prêts à l'emploi) qui permet de construire des interfaces modernes, interactives et modulaires. C'est cette technologie que j'ai utilisée pour créer l'affichage de la carte et des graphiques. L'usage de React était imposé par le cahier des charges du projet.

3.11. Crédit du backend : faire le lien avec la base de données

J'ai ensuite mis en place ce qu'on appelle le backend, c'est-à-dire la partie du programme qui n'est pas visible par l'utilisateur mais qui permet de faire fonctionner l'application. Le rôle principal du backend est de récupérer les données dans la base de

données et de les envoyer au frontend. Pour cela, j'ai utilisé un outil appelé Express, qui permet de créer facilement des routes, c'est-à-dire des adresses spécifiques auxquelles le frontend peut envoyer des demandes pour récupérer des données.

Le backend se connecte à la base de données MySQL, qui contient toutes les informations relevées par les stations (valeurs, coordonnées géographiques, dates, etc.). J'ai configuré des chemins pour récupérer, d'une part, les coordonnées des stations de mesure, et d'autre part, les résultats des analyses physico-chimiques. Ces données sont ensuite envoyées sous forme de fichiers JSON (un format de données lisible par les navigateurs) au frontend.

3.12. Développement du frontend : l'affichage et les interactions

Une fois les données disponibles, j'ai développé le frontend, c'est-à-dire tout ce que l'utilisateur voit et utilise sur la page web. J'ai intégré une carte interactive avec la bibliothèque Leaflet. Cette carte affiche des petits points (appelés marqueurs) aux emplacements des stations de mesure. Lorsque l'on clique sur un marqueur, une fenêtre apparaît avec les dernières données enregistrées à cet endroit (par exemple le taux d'oxygène, le pH, etc.).

En plus de la carte, j'ai ajouté des graphiques qui s'affichent lorsqu'on choisit une rivière dans une liste déroulante. Ces graphiques permettent de voir l'évolution des différentes données dans le temps. J'ai utilisé une autre bibliothèque appelée Chart.js, qui permet de créer des graphiques facilement à partir des données.

L'interface permet aussi de visualiser directement les données brutes (au format JSON) dans une section dédiée, pour avoir une vue complète et précise des informations.

4. Diagramme de Gantt

[Figure n°14 : Diagramme de Gantt](#)

5. Environnement technique / Contraintes

5.1. Choix de l'environnement de développement

Le développement du site web a été réalisé principalement en HTML, PHP et CSS, pour la création des pages statiques et dynamiques, notamment le formulaire de connexion. Ce choix permet une compatibilité directe avec le serveur LAMP mis en place en local. Pour le développement de la carte interactive, j'ai utilisé React, un framework JavaScript moderne permettant de concevoir des interfaces utilisateur réactives et modulaires. N'ayant jamais étudié cette technologie en cours, j'ai appris à m'en servir à travers de nombreux tutoriels et documentations en ligne.

L'environnement de développement local est basé sur Geany, un éditeur de texte léger et rapide adapté aux langages utilisés dans ce projet. Docker a également été utilisé en début de

projet pour permettre un travail indépendant et isolé, sans dépendre de l'avancée des autres membres de l'équipe.

5.2. Librairies et outils tiers utilisés

Plusieurs bibliothèques et outils ont été employés :

- Leaflet.js a été utilisé pour afficher une carte interactive. Cette bibliothèque est légère et facile à prendre en main. Elle permet de représenter les stations sur une carte avec des popups affichant les données.
- React a permis de structurer l'affichage de la carte et de ses composants, en rendant l'interface plus dynamique et maintenable.
- PHPMyAdmin a été utilisé pour consulter la base de données de manière visuelle.
- GitHub a servi d'outil de gestion de version, de suivi des tâches, et de coordination entre les membres de l'équipe. Nous avons également utilisé la méthode du Planning Poker pour estimer collectivement la difficulté des tâches à réaliser.

5.3. Contraintes imposées par le client

Le projet doit répondre à un ensemble de contraintes fonctionnelles et techniques définies par le client. L'une des principales exigences concerne la sécurité des données. En effet, les informations collectées (mesures environnementales, données saisies manuellement par le technicien) doivent être protégées contre toute tentative d'intrusion ou de falsification. Cela se traduit par la mise en place de mécanismes de hachage des mots de passe, de sécurisation des requêtes (pour éviter les injections SQL), et plus généralement par une attention particulière portée à la protection des communications entre le site web et la base de données.

Le client attend également une interface utilisateur intuitive, claire et facile à prendre en main. Celle-ci doit permettre à un technicien de consulter les données de manière lisible, de les saisir manuellement via un formulaire, ou encore de les visualiser géographiquement sur une carte. Cela implique une ergonomie réfléchie ainsi qu'une bonne réactivité de l'interface, notamment grâce à l'utilisation de bibliothèques comme Leaflet pour la carte et React pour l'interactivité.

D'un point de vue technique, le client impose aussi des contraintes matérielles et réseau : les capteurs doivent communiquer via LoRaWAN, les données doivent être stockées en local sur un serveur LAMP, et le site doit rester accessible depuis l'infrastructure réseau mise en place

par l'équipe. Enfin, une autre contrainte importante concerne la valorisation des données : les résultats d'analyse doivent pouvoir être structurés et validés pour une future publication sur des plateformes ouvertes comme data.gouv.fr, selon un format conforme à des standards publics.

6. Architecture matérielle / Caractéristiques et choix

6.1. Présentation de l'architecture matérielle du projet

L'architecture matérielle repose sur un ensemble de composants interconnectés permettant la collecte, la transmission, le traitement et l'affichage des données environnementales. Elle comprend principalement une carte Raspberry Pi comme unité centrale, raccordée à plusieurs capteurs physiques tels qu'un pluviomètre à godets et un limnimètre, eux-mêmes alimentés par un système autonome composé d'un panneau solaire et d'une batterie. Les données collectées par les capteurs sont transmises via un réseau LoRaWAN vers une passerelle, qui les redirige ensuite vers un serveur local de type LAMP (Linux, Apache, MySQL, PHP), sur lequel est hébergé le site web développé. Bien que je ne sois pas directement responsable de cette partie, cette infrastructure constitue le socle sur lequel s'appuie la partie web que je développe.

6.2. Architecture logicielle du projet

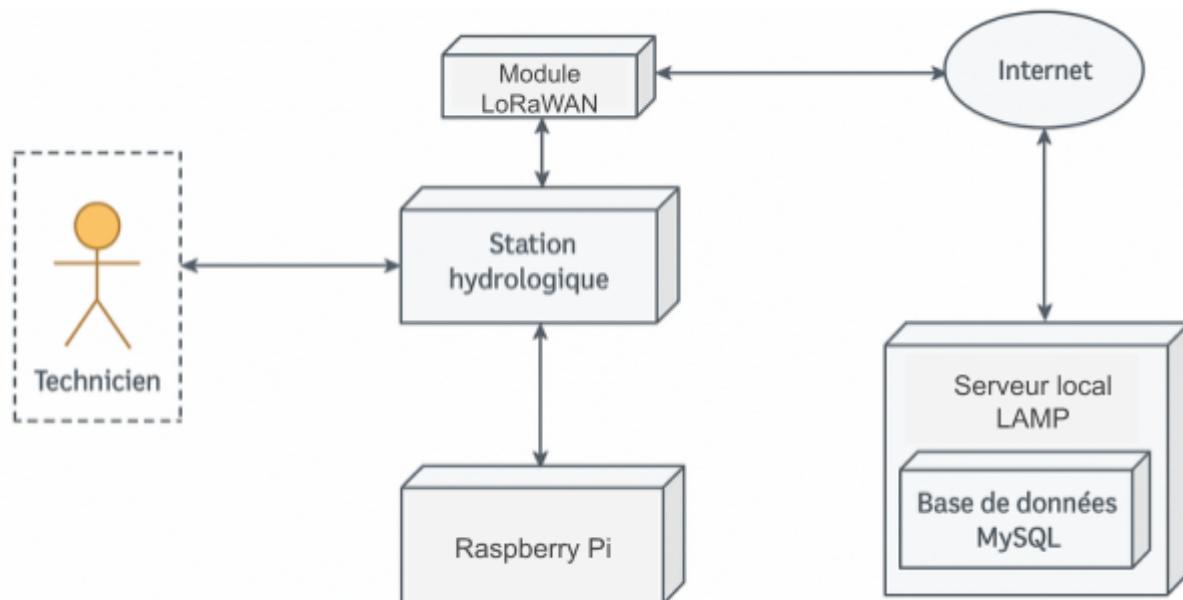


Figure 15 : Diagramme de déploiement de l'architecture logicielle du projet

Le diagramme ci-dessus illustre l'architecture de déploiement des différents composants logiciels et matériels impliqués dans le projet de surveillance automatisée des eaux de surface. Il permet de visualiser où et comment les différents éléments du système sont installés et interconnectés.

Au cœur de l'architecture, on retrouve la **station Raspberry Pi**, qui agit comme un nœud central recueillant les données des capteurs connectés : le **prélevEUR**, le **limnimètre**, et le **pluviomètre**. Elle est également connectée à un **module LoRaWAN** qui permet la transmission des mesures vers une **passerelle LoRaWAN**, laquelle achemine les données vers un **serveur distant** via le protocole TCP/IP.

Sur ce serveur sont hébergés les **services web** et les **interfaces de base de données**, essentiels pour l'exploitation des données à distance. Ce serveur est également relié à un **réseau GSM**, permettant l'envoi d'alertes vers un **smartphone de technicien** pour une intervention rapide en cas d'anomalie.

Enfin, la **connexion au réseau du SBEP** permet la centralisation des informations et leur analyse par l'organisme concerné. Ce diagramme met en avant la distribution logique des rôles de chaque composant, tout en soulignant leur interdépendance dans le bon fonctionnement global du système.

6.3. Justification des choix matériels

Le choix des capteurs s'est porté sur un pluviomètre à godets et un limnimètre, deux instruments reconnus pour leur fiabilité dans les mesures hydrologiques. Leur installation sur la Raspberry Pi permet une collecte de données locale, en temps réel, tout en restant économique en énergie grâce à l'alimentation solaire. Ces capteurs ont été sélectionnés en fonction de leur compatibilité avec les entrées de la carte Raspberry et de leur facilité d'interfaçage. Bien que je n'aie pas directement manipulé ces matériels, leur fonctionnement a un impact direct sur la fiabilité des données que je traite et affiche via l'interface web.

7. Spécifications techniques

7.1. Fonctionnement des capteurs et actionneurs

Le système s'appuie sur plusieurs capteurs pour collecter les données environnementales nécessaires à la surveillance des eaux de surface. Le pluviomètre à godets permet de mesurer les précipitations en comptant les basculements d'un godet. Chaque basculement équivaut à un volume précis d'eau, ce qui permet une mesure fiable des pluies. Le capteur limnimètre est utilisé pour mesurer le niveau d'eau d'un cours d'eau ou d'un réservoir. Ces capteurs envoient des signaux analogiques ou numériques, en fonction du modèle, qui sont ensuite traités avant d'être transmis.

Le système comprend également un actionneur, le prélevEUR, qui peut être activé automatiquement en fonction des données analysées (par exemple, en cas de crue ou de pollution détectée). Cet actionneur est piloté via une classe dédiée qui suit un automate à états finis, garantissant un comportement cohérent en fonction des différents scénarios.

7.2. Mise en œuvre des capteurs

Les capteurs sont installés sur une carte Raspberry Pi, configurée pour recevoir les données issues des dispositifs de mesure. Avant leur mise en service, les capteurs sont calibrés afin d'assurer la fiabilité des mesures. Le raccordement électrique est sécurisé, avec une alimentation autonome fournie par un panneau solaire connecté à une batterie. Un conditionnement du signal est réalisé si nécessaire (amplification, filtrage ou conversion), notamment dans le cas de signaux analogiques. Des bouchons de test ont également été réalisés pour simuler les capteurs en cas de maintenance ou de tests sans matériel actif.

7.3. Communication et format des données

Les données sont transmises à distance via une passerelle LoRaWAN vers un serveur local (SBEP). Le format des messages envoyés respecte une structure standardisée comprenant l'identifiant de la station, le type de capteur, la date, l'heure et la valeur mesurée. Ces données sont ensuite intégrées dans une base de données MySQL via une API ou directement par des scripts côté serveur. La communication suit un protocole léger, adapté aux faibles débits du réseau LoRaWAN, et garantit l'intégrité des messages.

7.4. Interfaces entre les composants

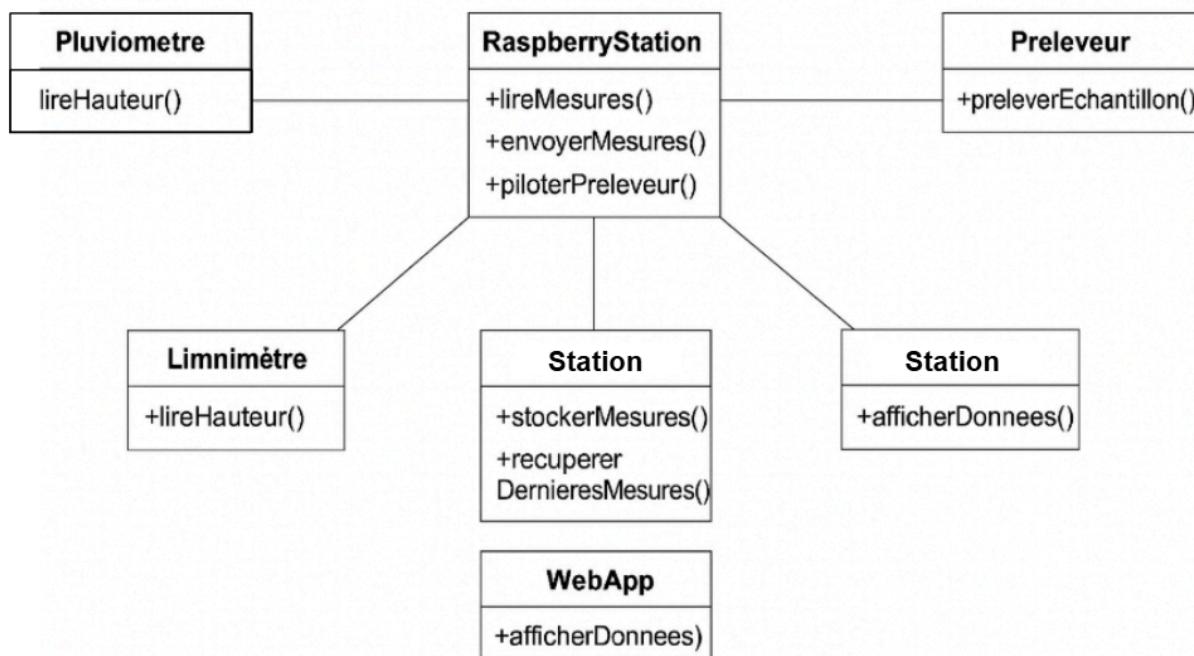


Figure n°16 : Diagramme des interfaces entre les composants du système

Ce diagramme représente les **interfaces logicielles et matérielles** entre les différents composants du **système de surveillance des eaux de surface**. Il permet de visualiser comment les **modules interagissent**, notamment la manière dont les **capteurs (pluviomètre, limnimètre, prélevageur)** sont reliés à la **station Raspberry Pi** via des **E/S TOR** (entrées / sorties tout ou rien), et comment les **données** sont transmises au **serveur** via la **passerelle LoRaWAN**. On y voit également les **services web** et la **base de données MySQL** interfacés avec le **serveur local LAMP**, ainsi que l'**accès distant** possible par **smartphone** via le **réseau GSM**. Ce schéma met en évidence les **rôles de chaque composant** et leur **mode de communication**, qu'il soit physique ou en réseau, ce qui est essentiel pour la bonne compréhension de l'**architecture du système**.

7.5. Politique de protection des données

La sécurité des données collectées et transmises est une priorité. Toutes les communications sont protégées par des mécanismes de chiffrement natifs du réseau LoRaWAN. Les mots de passe des utilisateurs du site web sont hachés avant d'être stockés dans la base de données. Des protections contre les injections SQL sont mises en place dans les scripts PHP. De plus, seules les données validées via un schéma sont autorisées à être publiées sur le portail des données ouvertes, garantissant leur conformité. Enfin, les mises à jour des composants logiciels (frameworks, bibliothèques, serveurs) sont régulièrement effectuées pour corriger d'éventuelles failles de sécurité connues.

8. Travail collaboratif

8.1. Mise en place de l'espace collaboratif

Pour faciliter la coordination entre les membres de l'équipe, nous avons utilisé GitHub comme espace collaboratif principal. Cet outil a permis de centraliser le code, de partager des fichiers importants (diagrammes, schémas, documentation), et de suivre l'évolution du projet. Chaque membre pouvait ainsi accéder aux dernières versions des éléments produits et proposer des modifications via le système de "pull requests".

Une "pull request" est une fonctionnalité qui permet à un développeur de soumettre les changements effectués dans une branche afin qu'ils soient relus et validés avant d'être intégrés dans la branche principale du projet. Ce mécanisme favorise la revue de code, garantit une meilleure qualité du développement, et assure que tous les membres soient au courant des modifications importantes. Ce choix a grandement facilité le travail en équipe, notamment lors des phases où plusieurs éléments devaient être développés en parallèle.

8.2. Utilisation et démonstration de l'espace collaboratif

GitHub a été utilisé à la fois pour le versionnement du code et pour le suivi de projet. Chaque tâche était représentée sous forme d'issues, classées par thème ou composant, ce qui permettait de répartir les responsabilités de manière claire. Des branches ont été créées pour chaque fonctionnalité majeure, notamment pour le développement de la carte interactive ou du formulaire de saisie, afin de ne pas perturber la version stable du projet. La revue de code et l'intégration continue ont permis de détecter rapidement d'éventuelles erreurs ou conflits.

8.3. Comparaison entre le planning prévisionnel et le planning effectif

Le planning prévisionnel a été défini en équipe grâce à la méthode du Planning Poker, qui permettait d'estimer la complexité des tâches. Si certaines fonctionnalités ont été réalisées dans les délais, d'autres ont pris plus de temps que prévu. C'est le cas notamment du développement de la carte interactive en React et Leaflet, ainsi que de la gestion des utilisateurs via des fenêtres pop-up (ajout, modification, suppression), qui ont demandé plus d'efforts techniques que prévu.

8.4. Justification des écarts entre le temps prévu et le temps réalisé

Les écarts constatés sont principalement dus à l'apprentissage nécessaire de technologies que je n'avais jamais vues en cours. La mise en place de la carte interactive en React a exigé de nombreuses recherches en ligne. De plus, la mise en œuvre des fonctionnalités liées à la gestion des utilisateurs, avec des fenêtres modales interactives, a demandé une réflexion particulière sur l'ergonomie et le traitement côté serveur avec PHP. Ces imprévus ont allongé les délais, mais m'ont permis de monter en compétence et d'assurer une meilleure qualité pour ces fonctionnalités.

9. Gestion des versions

9.1. Liste et présentation des différents reportings et versionings

Tout au long du projet, l'équipe a utilisé GitHub pour assurer le suivi des modifications et des versions. Chaque évolution significative du code a été documentée à travers des commits clairs, souvent associés à une issue ou une pull request. Ces reportings ont permis de tracer les ajouts, corrections de bugs, ajustements fonctionnels ou techniques, assurant une bonne lisibilité de l'historique du projet.

9.2. Présentation de l'espace collaboratif pour les versions de livraison

L'espace GitHub contient l'ensemble des versions du projet, organisées par branches thématiques (ex. : formulaire-utilisateur, map-interactive, connexion-bdd). Les livrables intermédiaires ont été regroupés sous forme de versions balisées (tags) correspondant aux jalons définis dans notre planning. Cela permet à chaque membre de revenir à un état stable du projet en cas de besoin.

9.3. Mise en place de l'outil de versionnement et documentation des composants

Git a été utilisé comme outil de versionnement. Chaque fonctionnalité majeure (comme la connexion utilisateur, la carte interactive, ou l'intégration des données physico-chimiques) a été développée dans une branche dédiée, testée, puis fusionnée avec la branche principale (main). Chaque composant est accompagné de commentaires dans le code expliquant son rôle, facilitant la compréhension et la reprise du projet.

9.4. Intégration et packaging de la dernière livraison du projet

La dernière version du projet a été intégrée sur le serveur LAMP local, accessible via l'adresse IP locale du réseau. Grâce au travail de mon camarade chargé de la base de données et du Docker, cette version est conteneurisée pour assurer un déploiement cohérent. Tous les composants développés (site web, carte, formulaires, interfaces API) sont intégrés dans cette version finale et ont été testés ensemble.

10. Aspects liés à la sécurité / robustesse des éléments mis en œuvre

10.1. Liste des versions des matériels et logiciels(revoir)

L'ensemble des outils utilisés dans ce projet sont à jour, ce qui garantit une compatibilité optimale et réduit les risques liés à d'éventuelles failles de sécurité connues. Les principales versions sont :

10.2. Les outils utilisés

Github :

- **Nom de l'outil :** GitHub
- **Fonctionnalité :** GitHub est une plateforme de développement collaboratif qui permet de gérer des projets, de suivre les versions du code source et de collaborer avec d'autres développeurs.

- **Version installée :** Service en ligne, toujours à jour
- **Procédure d'installation :** GitHub est une plateforme en ligne accessible via un navigateur web. Il n'y a pas d'installation nécessaire pour utiliser GitHub. Il suffit de créer un compte sur « <https://github.com> » et de créer ou rejoindre des dépôts pour commencer à collaborer.
- **Raison du choix de cet outil :** J'ai choisi GitHub pour sa popularité et ses fonctionnalités robustes de gestion de version, de collaboration et d'intégration continue. GitHub facilite également la gestion de projets en utilisant des outils comme les Issues, les Projets (tableaux Kanban), et les Pull Requests.
- **Ligne de commande pour vérifier l'installation :** git --version

PHP :

- **Nom de l'outil :** PHP
- **Fonctionnalité :** PHP est un langage de script côté serveur utilisé pour le développement web. Il permet de créer des pages web dynamiques et interactives.
- **Version installée :** PHP 8.2
- **Procédure d'installation :** Pour installer PHP, j'ai utilisé Docker pour créer un conteneur avec l'image officielle de PHP. La procédure d'installation est donc intégrée dans le fichier « Dockerfile » utilisé par Docker.
- **Raison du choix de cet outil :** J'ai choisi PHP car c'est un langage de script très répandu et largement utilisé pour le développement web. De plus, il est compatible avec de nombreux systèmes de gestion de bases de données, dont MySQL, ce qui facilite l'intégration et le développement.
- **Ligne de commande pour vérifier l'installation :** php -v

MySQL :

- **Nom de l'outil :** MySQL
- **Fonctionnalité :** MySQL est un système de gestion de bases de données relationnelles. Il permet de stocker et de gérer les données de manière structurée et efficace.
- **Version installée :** MySQL 5.7
- **Procédure d'installation :** L'installation de MySQL a également été réalisée via Docker. J'ai utilisé l'image officielle de MySQL et configuré le service dans le fichier « docker-compose.yml ».
- **Raison du choix de cet outil :** MySQL est choisi pour sa fiabilité, ses performances et sa facilité d'utilisation. Il est également bien intégré avec PHP, ce qui simplifie le développement et la gestion des données.
- **Ligne de commande pour vérifier l'installation :** mysql -V

Docker :

- **Nom de l'outil :** Docker
- **Fonctionnalité :** Docker est une plateforme de conteneurisation qui permet de créer, déployer et exécuter des applications dans des conteneurs. Cela garantit une exécution cohérente et isolée des applications.
- **Version installée :** Docker 20.10
- **Procédure d'installation :** Docker peut être installé en suivant les instructions officielles sur le site de Docker. Pour ce projet, j'ai utilisé Docker pour créer des conteneurs pour PHP, MySQL et PHPMyAdmin.
- **Raison du choix de cet outil :** Docker est choisi pour sa capacité à isoler les environnements de développement, ce qui permet de travailler de manière cohérente et reproductible sur différentes machines. Il simplifie également le déploiement et la gestion des dépendances.
- **Ligne de commande pour vérifier l'installation :** docker –version

Leaflet :

- **Nom de l'outil :** Leaflet
- **Fonctionnalité :** Leaflet est une bibliothèque JavaScript open-source utilisée pour créer des cartes interactives. Elle est légère, performante et facile à utiliser, ce qui la rend idéale pour des projets nécessitant des cartes interactives sur le web.
- **Version installée :** La version installée est Leaflet v1.7.1.
- **Procédure d'installation :** npm install leaflet
- **Raison du choix de cet outil :** Leaflet a été choisi pour sa simplicité, sa légèreté et ses performances élevées. Il offre une large gamme de fonctionnalités pour créer des cartes interactives et est largement adopté par la communauté des développeurs. De plus, son intégration facile avec d'autres bibliothèques JavaScript et ses nombreuses extensions en font un choix idéal pour des projets nécessitant des cartes dynamiques.
- **Ligne de commande pour vérifier l'installation :** nmp list leaflet

Tous les outils ont été installés à l'aide de Docker, ce qui facilite la gestion des versions et permet une mise à jour centralisée et maîtrisée.

10.3. Vérification de la mise à jour des logiciels et pilotes

Le recours à Docker permet de déployer des conteneurs contenant les dernières versions stables des outils utilisés. Cela garantit que les composants logiciels sont à jour, limitant les vulnérabilités et assurant une compatibilité avec les bibliothèques récentes. Les images Docker officielles ont été privilégiées pour fiabiliser l'ensemble du système.

10.4. Configuration des éléments matériels et logiciels pour une résistance aux attaques

Pour limiter les risques d'attaques :

- Les mots de passe sont hachés avant d'être stockés dans la base de données.
- Le formulaire de connexion a été conçu en évitant les failles classiques comme l'injection SQL, grâce à l'usage de requêtes préparées.
- L'accès aux données est restreint aux utilisateurs authentifiés.
- Le serveur étant local, seules les personnes connectées au réseau peuvent y accéder, ajoutant une couche de sécurité physique.

De plus, l'utilisation de conteneurs Docker permet d'isoler les services (base de données, serveur web, interface d'administration...), ce qui renforce la sécurité globale du système en cloisonnant les composants.

10.5. Prise en compte des enjeux de cybersécurité (failles de sécurité, signature, certificats, etc.) et des mises à jour

Le projet prend en compte plusieurs enjeux de cybersécurité :

- Mises à jour régulières des images Docker utilisées.
- Séparation des services dans des conteneurs indépendants, permettant une gestion fine des accès et des permissions.
- Anticipation des failles grâce à l'utilisation de versions stables et éprouvées des outils.
- L'approche collaborative avec GitHub permet aussi une traçabilité des changements et un suivi rigoureux du code, limitant les risques d'introduction de failles non contrôlées.

4. Annexes

4.1. Etudiant n°1

4.2. Etudiant n°2

interface BDD - WEB #48

Alexisalou opened on Mar 10

En tant qu'étudiant, je souhaite mettre en place des fonctions utilisable par les technicien SBEPM afin de faciliter les envois et réception de données

Create sub-issue

Assignees

- Alexisalou
- nolanconan

Labels

No labels

Projects

Planning Eau Pure

Status: Réalisés

sprint 1	sprint 1 2 + Mar 10 - Mar 23
doc raspberry	No date
sprint2	Choose an iteration

Milestone

No milestone

Relationships

None yet

Development

Create a branch for this issue or link a pull request.

Notifications

Unsubscribe

You're receiving notifications because you're

Image n°4 : Cas utilisateur

Image n°5 : Calendrier

Annexe n°3 :

```

def Envois_mesures(capteur, valeur, unite, date):
    try:
        # Connexion à la BDD
        conn = mysql.connector.connect(
            host=DATABASE_HOST,
            database=DATABASE_NAME,
            user=DATABASE_USER,
            password=DATABASE_PASSWORD,
            port=DATABASE_PORT,
        )
        cursor = conn.cursor()

        # Insertion des données
        cursor.execute('''
            INSERT INTO Mesure (capteur, valeur, unite, date)
            VALUES (%s, %s, %s, %s)
        ''', (capteur, valeur, unite, date))

        # Valider la transaction
        conn.commit()

        print(f"Mesure insérée: {capteur}, {valeur}, {unite}, {date}")

    except mysql.connector.Error as err:
        print(f"Erreur: {err}")

    finally:
        # Fermer la connexion
        if conn.is_connected():
            cursor.close()
            conn.close()

def lire_seuils(db_config):
    db = mysql.connector.connect(
        host=db_config['host'],
        user=db_config['user'],
        password=db_config['password'],
        database=db_config['database'],
        port=db_config['port']
    )
    cursor = db.cursor()

    # Récupérer le seuil du pluviomètre
    cursor.execute("SELECT seuil_pluviometre FROM Preleveur")
    seuil_pluviometre = cursor.fetchone()

    # Récupérer le seuil du limnimètre
    cursor.execute("SELECT seuil_limnimetre FROM Preleveur")
    seuil_limnimetre = cursor.fetchone()

    db.close()

    if seuil_pluviometre and seuil_limnimetre:
        return seuil_pluviometre[0], seuil_limnimetre[0]
    else:
        raise ValueError("Les seuils nécessaires ne sont pas disponibles.")

```

```
def lire_mesures(db_config):
    db = mysql.connector.connect(
        host=db_config['host'],
        user=db_config['user'],
        password=db_config['password'],
        database=db_config['database'],
        port=db_config['port']
    )
    cursor = db.cursor()

    # Récupérer la dernière mesure du pluviomètre
    cursor.execute("SELECT valeur FROM Mesure WHERE capteur = %s ORDER BY date DESC LIMIT 1", (PLUVIOMETER_SENSOR_ID,))
    mesure_pluviometre = cursor.fetchone()

    # Récupérer la dernière mesure du limnimètre
    cursor.execute("SELECT valeur FROM Mesure WHERE capteur = %s ORDER BY date DESC LIMIT 1", (LIMNIMETER_SENSOR_ID,))
    mesure_liminimetre = cursor.fetchone()

    db.close()

    if mesure_pluviometre and mesure_liminimetre:
        return mesure_pluviometre[0], mesure_liminimetre[0]
    else:
        raise ValueError("Les mesures nécessaires ne sont pas disponibles.")
```

4.3. Etudiant n°3

The screenshot shows a GitHub project board for 'Planning Eau Pure'. The board is organized into five columns representing the Scrum process:

- Carnet de produit**: Contains 15 tasks related to product architecture, network security, and documentation.
- Carnet de Sprint**: Contains 10 tasks related to creating developer identifiers, OpenStreetMap/Leaflet, and automated tests.
- En cours**: Contains 10 tasks related to documentation (RGPD, Git), test fixtures, and test probes.
- Réalisés**: Contains 10 tasks related to documentation (BDD, BDD-Web, STATION-BDD), validation, and research into LoRaWAN technology.
- Validés**: Contains 10 tasks related to creating a tableaux SCRUM, documentation, and a password hashing program.

Figure n°4 : Réalisation d'un tableau Scrum créé sur GitHub à partir du cahier des charges

The screenshot shows a GitHub issue page for 'Recherche technologie LoRaWAN #9' and a concurrent Planning Poker session on ScrumPoker-online.org.

GitHub Issue Details:

- Assignees:** MathysLS
- Labels:** No labels
- Projects:** Planning Eau Pure
- Status:** Carnet de Sprint
- Iteration:** sprint 1 (Choose an iteration)
- Milestone:** No milestone
- Development:** Create a branch for this issue or link a pull request.
- Notifications:** Customize, Subscribe

Planning Poker Session:

- ScrumPoker-online.org Interface:** Shows a grid of cards with values 8, 13, 20, 40, and 100.
- Results Table:**

Name	Story Points
alexis	8
Eau_pure	8
Mathys	8
Nolan	8

Figure n°5 : Évaluation de la difficulté des tâches à l'aide du Planning Poker



The screenshot shows a light blue-themed login interface. At the top, there is a header section with the text "Numéro de téléphone :" followed by a text input field labeled "Numéro de téléphone". Below this is another header section with the text "Mot de passe :" followed by a text input field labeled "Mot de passe". Underneath these fields are two large, light blue rectangular buttons with white text: "Se connecter" and "Gestion des utilisateurs".

Figure n°8 : Réalisation d'un formulaire de connexion



Figure n°9 : Pop-up de connexion administrateur

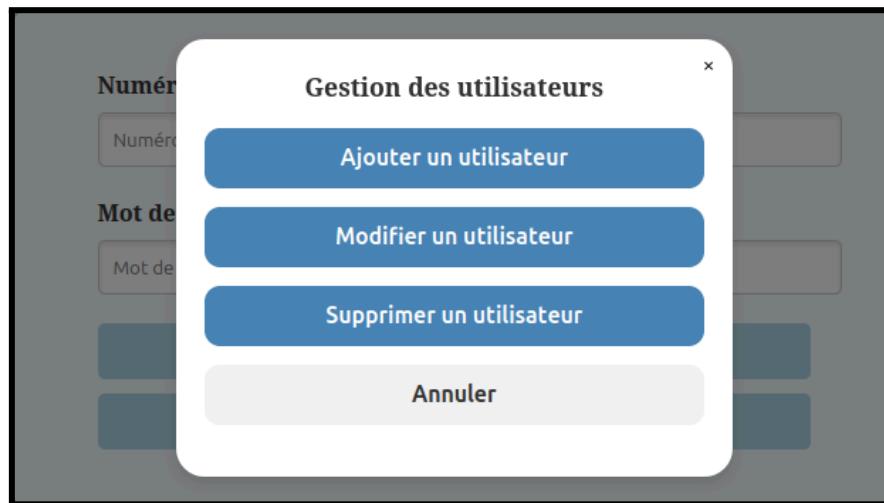
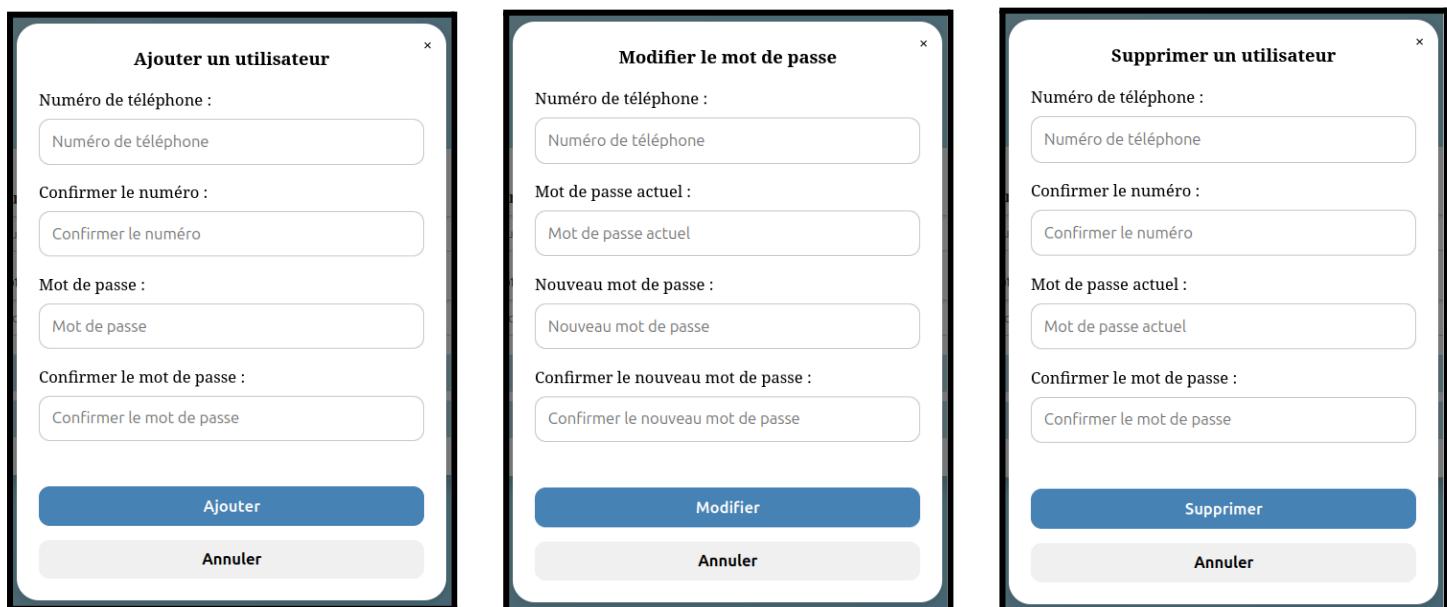


Figure n°10 : Pop-up de gestion des utilisateurs



Ajouter un utilisateur

Numéro de téléphone :

Confirmer le numéro :

Mot de passe :

Confirmer le mot de passe :

Modifier le mot de passe

Numéro de téléphone :

Mot de passe actuel :

Nouveau mot de passe :

Confirmer le nouveau mot de passe :

Supprimer un utilisateur

Numéro de téléphone :

Confirmer le numéro :

Mot de passe actuel :

Confirmer le mot de passe :

Figure n°11 : Réalisation des fenêtres pop-up pour ajouter, modifier ou supprimer un utilisateur

Formulaire de Données Physico-Chimiques pour le Traitement des Eaux

pH (0-14):

Le pH est une mesure de l'acidité ou de la basicité d'une solution.
Plages de Mesure Typiques :

- Acides forts : 0 à 3
- Acides faibles : 3 à 6
- Neutre (eau pure) : 7
- Bases faibles : 8 à 11
- Bases fortes : 12 à 14

Conductivité Électrique ($\mu\text{S}/\text{cm}$) (0.05-10000):

La conductivité électrique est une mesure de la capacité de l'eau à conduire un courant électrique.
Plages de Mesure Typiques :

- Eau ultra-pure : 0.05 à 1 $\mu\text{S}/\text{cm}$
- Eau de pluie : 2 à 100 $\mu\text{S}/\text{cm}$
- Eau potable : 50 à 1500 $\mu\text{S}/\text{cm}$
- Eau de rivière propre : 100 à 2000 $\mu\text{S}/\text{cm}$
- Eau de mer : 30 à 50 mS/cm (30,000 à 50,000 $\mu\text{S}/\text{cm}$)
- Eaux usées : 1000 à 10000 $\mu\text{S}/\text{cm}$

Turbidité (NTU) (0-1000):

La turbidité est une mesure de la clarté de l'eau.
Plages de Mesure Typiques :

- Eau très claire : 0 à 1 NTU
- Eau potable : 0 à 5 NTU
- Eau de rivière propre : 1 à 50 NTU
- Eau de rivière polluée : 50 à 200 NTU
- Eau très brouillée : 200 à 1000 NTU
- Eaux usées non traitées : 1000 NTU et plus

Oxygène Dissous (mg/L) (0-14):

L'oxygène dissous est une mesure de la quantité d'oxygène présente dans l'eau.
Plages de Mesure Typiques :

- Eau très propre : 8 à 14 mg/L
- Eau potable : 6 à 12 mg/L
- Eau de rivière propre : 6 à 12 mg/L
- Eau de rivière polluée : 2 à 6 mg/L
- Eaux usées : 0 à 2 mg/L

Demande Chimique en Oxygène (DCO) (mg/L) (0-1000):

La Demande Chimique en Oxygène (DCO) est une mesure de la quantité d'oxygène nécessaire pour oxyder chimiquement les matières organiques et inorganiques présentes dans l'eau.
Plages de Mesure Typiques :

- Eau très propre : 0 à 20 mg/L
- Eau légèrement polluée : 20 à 50 mg/L
- Eau de rivière polluée : 50 à 200 mg/L
- Eaux usées domestiques traitées : 20 à 100 mg/L
- Eaux usées domestiques non traitées : 200 à 600 mg/L
- Eaux usées industrielles : 200 à 1000 mg/L ou plus

Envoyer

Figure n°12 : Réalisation du formulaire pour la saisie des données physico-chimiques

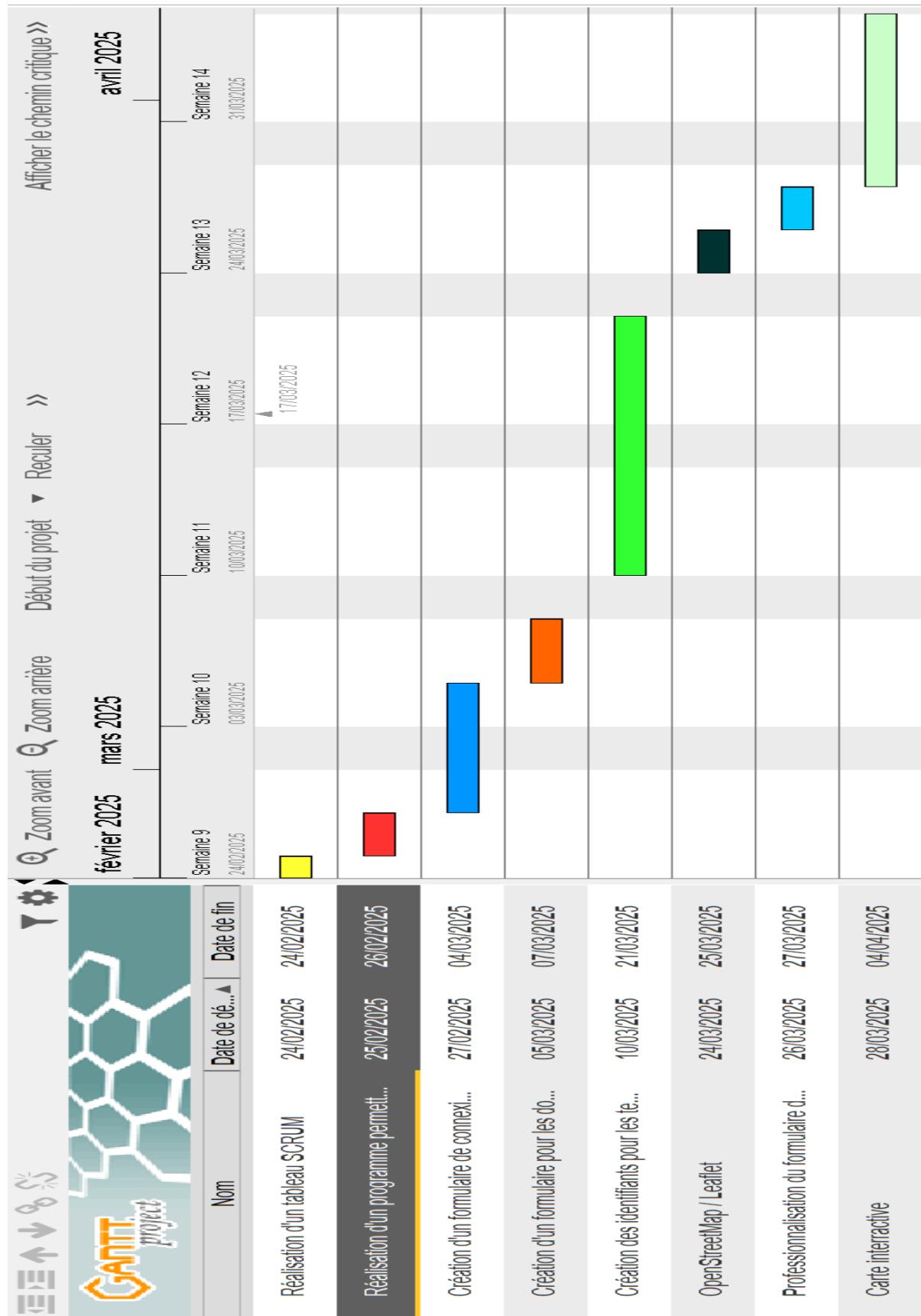


Figure n°14 : Diagramme de Gantt

				id	prelevement	valeur	unite	type
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	16		1	3	ph
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	17		1	1000	ÂµS/cm conductivité
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	18		1	500	NTU turbidité
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	19		1	14	mg/L oxygène
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	20		1	500	mg/L dco

Implémentation des valeurs physico-chimiques dans la base de données