



# Session 2025

## Rapport de projet



---

## Projet Eau Pure

---

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

**Table des matières :**

|  |           |
|--|-----------|
| <b>Partie commune.....</b>   | <b>4</b>  |
| <b>1. Contexte du projet.....</b>  | <b>4</b>  |
| 1.1. Contexte et objectifs du projet de surveillance automatisée des eaux..... | 4         |
| 1.2. Diagramme des exigences.....  | 5         |
| <b>2. Spécifications fonctionnelles.....</b>                                   | <b>6</b>  |
| <b>3. Etude préliminaire.....</b>  | <b>7</b>  |
| 3.1. Analyse des besoins.....  | 7         |
| 3.2. Diagramme de déploiement.....   | 8         |
| 3.3. Présentation de l'architecture matérielle du projet.....                  | 9         |
| 3.4. Justification des choix matériels.....                                    | 9         |
| 3.5. Fonctionnement des capteurs et actionneurs.....                           | 9         |
| 3.6. Mise en œuvre des capteurs.....   | 10        |
| 3.7. Communication et format des données.....                                  | 10        |
| 3.8. Politique de protection des données.....                                  | 10        |
| 3.9. Diagramme de classe.....  | 11        |
| 3.10. Schéma relationnel de la base de données.....                            | 12        |
| <b>4. Planification.....</b>   | <b>13</b> |
| 4.1. Répartition des rôles et responsabilités.....                             | 13        |
| 4.2. Mise en place de l'espace collaboratif.....                               | 14        |
| 4.3. Utilisation et démonstration de l'espace collaboratif.....                | 14        |
| 4.4. Diagramme de Gantt.....   | 15        |
| <b>5. Recette.....</b>   | <b>16</b> |
| <b>6. Conclusion.....</b>  | <b>17</b> |
| <br>   |           |
| <b>Étudiant n°1 - Mathys LE SOURNE.....</b>                                    | <b>20</b> |
| <b>Étudiant n°2 - Alexis SALOU.....</b>  | <b>45</b> |
| <b>Étudiant n°3 - Nolan CONAN.....</b>   | <b>63</b> |
| <br>   |           |
| <b>Annexes.....</b>  | <b>85</b> |
| <b>1. Etudiant n°1.....</b>  | <b>85</b> |
| <b>2. Etudiant n°2.....</b>  | <b>89</b> |
| <b>3. Etudiant n°3.....</b>  | <b>96</b> |

---

## Partie commune

---



M.Le Sourne



A.Salou



N.Conan



## Partie commune

### 1. Contexte du projet

#### 1.1. Contexte et objectifs du projet de surveillance automatisée des eaux

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. Diagramme des exigences

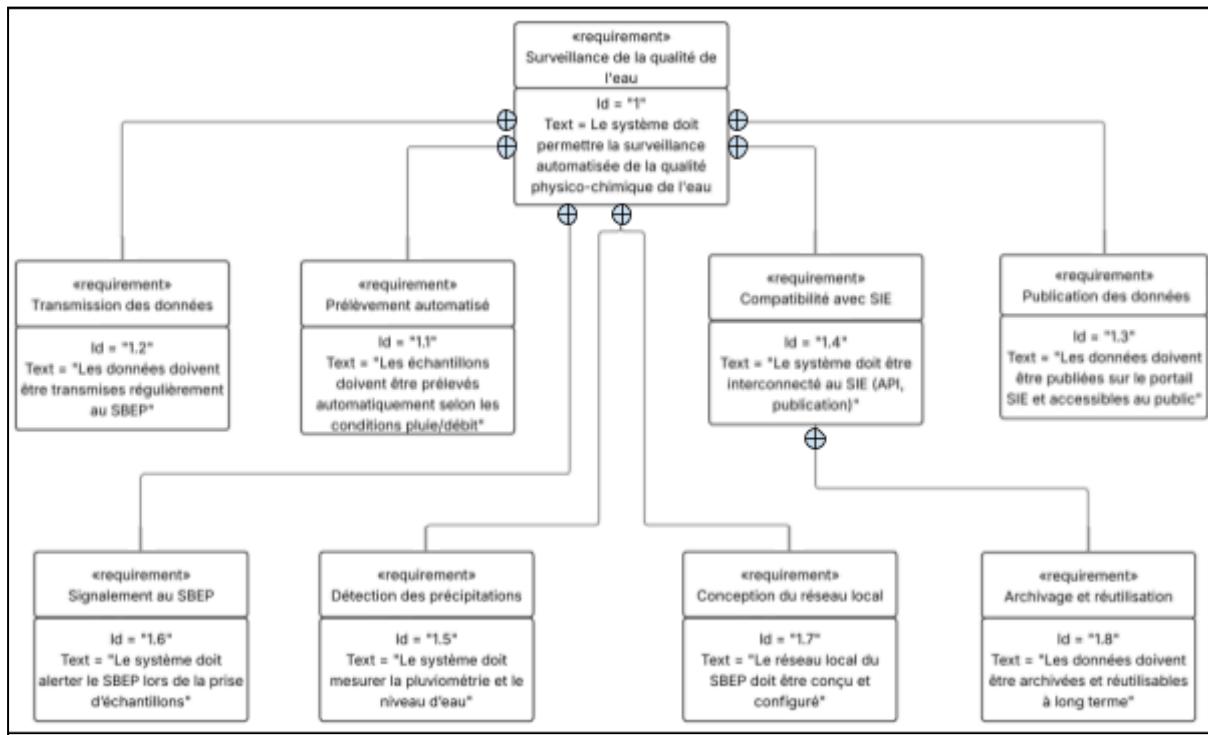


Figure n°1 : Diagramme des exigences fonctionnelles du système de surveillance de la qualité de l'eau

Ce diagramme illustre l'ensemble des **exigences fonctionnelles** associées à un **système de surveillance automatisée de la qualité physico-chimique de l'eau**.

À partir de l'**exigence principale** (Id = "1"), plusieurs **sous-exigences** sont dérivées, définissant les **fonctionnalités clés** attendues du système.

Parmi celles-ci figurent l'**automatisation du prélèvement des échantillons** (1.1), la **transmission régulière des données** (1.2), la **publication des résultats** sur un **portail public interconnecté au Système d'Information Environnemental (SIE)** (1.3 et 1.4), ainsi que l'**archivage et la réutilisation des données** (1.8).

D'autres exigences détaillent les **besoins techniques complémentaires**, tels que la **détection des précipitations** (1.5), le **signalement au SBEP** lors des prélèvements (1.6), et la **conception du réseau local du SBEP** (1.7).

Ce diagramme permet ainsi de **structurer** et de **visualiser les attentes fonctionnelles** du système pour garantir une **gestion efficace, transparente et intégrée de la qualité de l'eau**.

## 2. Spécifications fonctionnelles

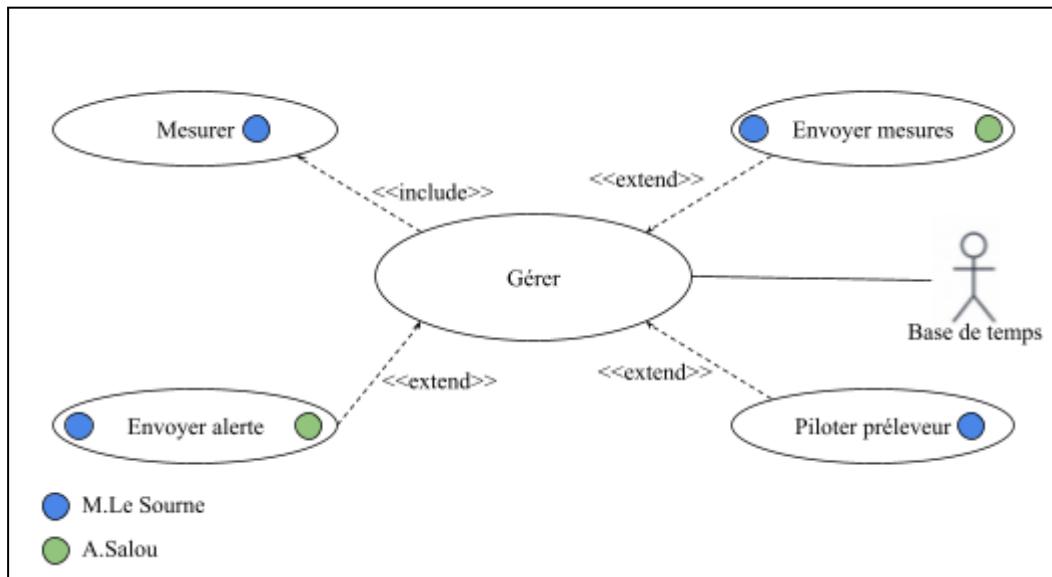


Figure n°2 : Diagramme de cas d'utilisation du sous-système station

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.

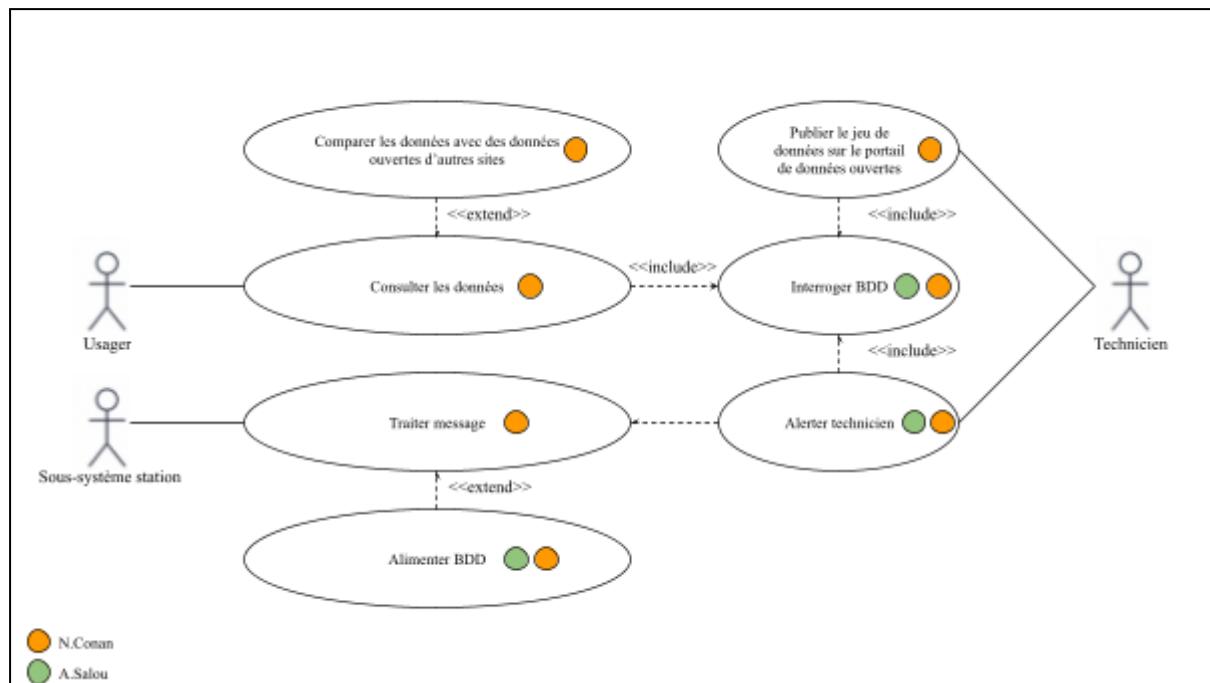


Figure n°3 : Diagramme de cas d'utilisation du sous-système SBEP

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.

**Usager** : consulte depuis l'interface Web les données publiées par le technicien.

**Technicien** : met à jour les données sur le site après avoir collecté un échantillon suite à une alerte et l'avoir fait analyser par un laboratoire.

### 3. Etude préliminaire

#### 3.1. Analyse des besoins

Pour répondre aux exigences de la directive-cadre européenne sur l'eau (2000/60/CE), le SBEP doit respecter des normes strictes de qualité environnementale pour les masses d'eau.

Actuellement, les techniciens réalisent manuellement les prélèvements d'échantillons d'eau, à intervalles réguliers. Ce processus mobilise beaucoup de temps et de ressources humaines.

Il devient donc nécessaire de **moderniser et automatiser** ces tâches. L'objectif est de déclencher les prélèvements de façon plus pertinente, notamment selon les conditions météorologiques (pluie, crues, etc.).

Le SBEP a ainsi besoin d'un **système automatisé** capable de surveiller en continu la qualité physico-chimique des cours d'eau.

En parallèle, il est essentiel d'assurer la **transmission en temps réel** des données mesurées (hauteur d'eau, pluviométrie) vers le SBEP. Ces données devront également être **publiées sur des plateformes ouvertes** comme data.gouv.fr ou le portail du SIE. L'objectif : favoriser la transparence, la réutilisation nationale des données, et contribuer à la **gestion des risques** (ex. Vigicrues).

Enfin, le système devra :

- permettre la **signalisation immédiate des prélèvements** nécessitant une intervention humaine,
- s'intégrer efficacement au **réseau informatique du SBEP**,
- et rester **peu coûteux en maintenance**, grâce à une prise d'échantillons conditionnelle et intelligente.

### 3.2. Diagramme de déploiement

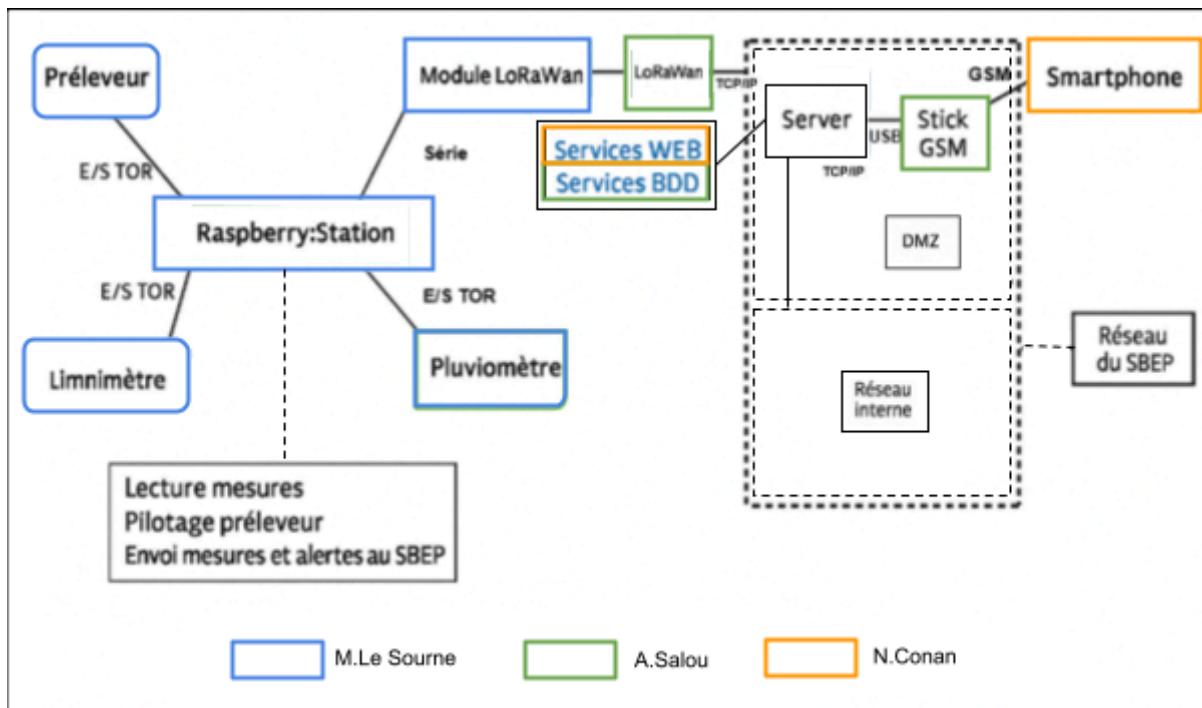


Figure n°4 : Diagramme de déploiement - Schéma global du système

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 :

- Etudiant 1 (M.Le Sourne ; capteurs et réseau) : Mise en place et gestion des capteurs (pluviomètre, limnimètre, prélevageur), configuration du Raspberry Pi et du module LoRaWan.

● Etudiant 2 (A.Salou ; 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.

● Etudiant 3 (N.Conan ; interface web) : Développement de la partie web : carte interactive, affichage des données, formulaire de connexion, intégration avec l'API et services BDD.

Ce diagramme (figure n°4) permet de visualiser clairement la répartition des rôles dans le projet, en lien direct avec l'architecture matérielle et logicielle mise en œuvre.

### **3.3. 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 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.

### **3.4. 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.

Les spécificités techniques de ces capteurs ainsi que leur intégration avec la Raspberry Pi seront détaillées plus en profondeur dans la partie suivante, présentée par M.Le Sourne."

### **3.5. 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.

### **3.6. 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.

### **3.7. 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.

### **3.8. 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.

### 3.9. Diagramme de classe

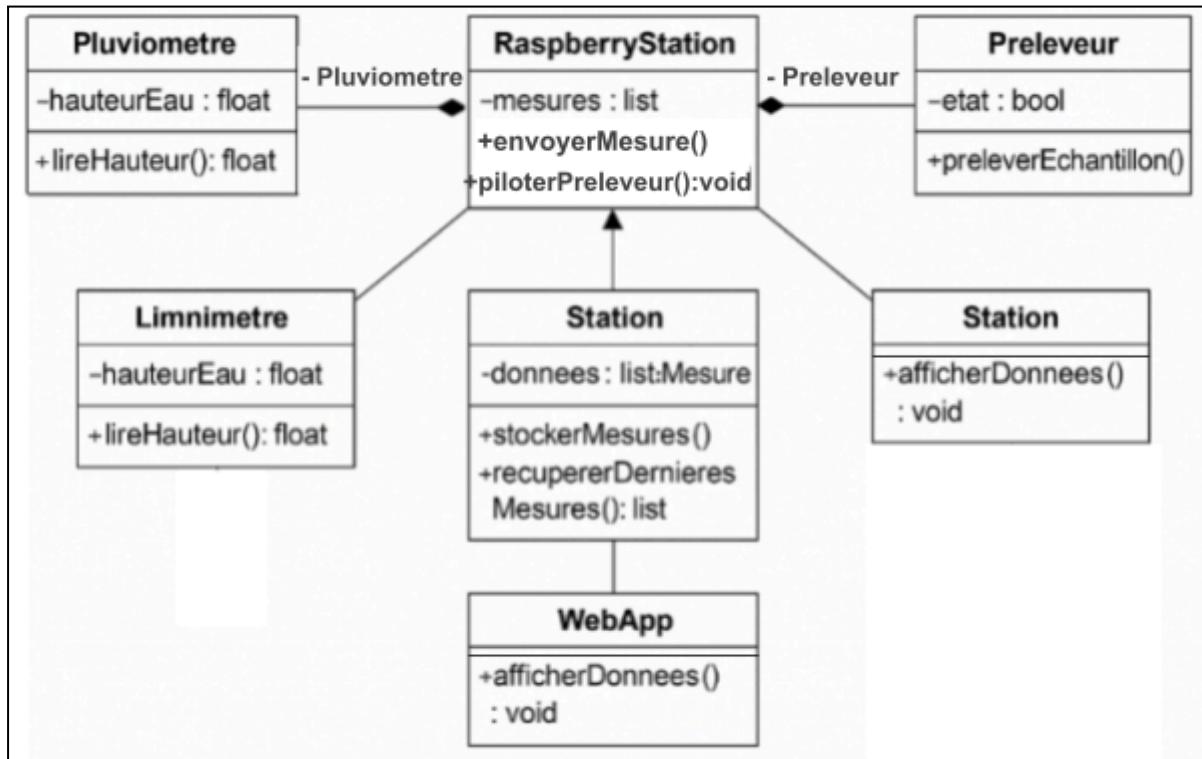


Figure n°5 : Diagramme de classe du système de gestion des mesures environnementales

Le diagramme de classe illustre une architecture modulaire pour un système de collecte, gestion et affichage de données environnementales. La classe centrale, *RaspberryStation*, connecte les modules du système, gère une liste de mesures (*mesures : list*) et propose deux méthodes principales : *envoyerMesure()* pour transmettre les données et *piloterPreleveur()* pour contrôler la classe *Preleveur*. Cette dernière, avec son attribut *etat : bool* (activé/désactivé), effectue les prélèvements via *preleverEchantillon()*.

Les classes *Pluviometre* et *Lnimimetre* mesurent la hauteur d'eau grâce à leur attribut *hauteurEau : float* et la méthode *lireHauteur()*. Ces capteurs alimentent la classe *Station*, qui centralise les données dans une liste (*donnees : list*) et offre des méthodes pour stocker (*stockerMesures()*) et récupérer (*recupererDernieresMesures()*) les informations.

L'affichage des données est assuré par les classes *Station* (local) et *WebApp* (en ligne), chacune dotée de la méthode *afficherDonnees()* pour la visualisation. Cette structure hiérarchisée attribue des responsabilités claires, simplifiant maintenance et évolutivité.

### 3.10. Schéma relationnel de la base de données

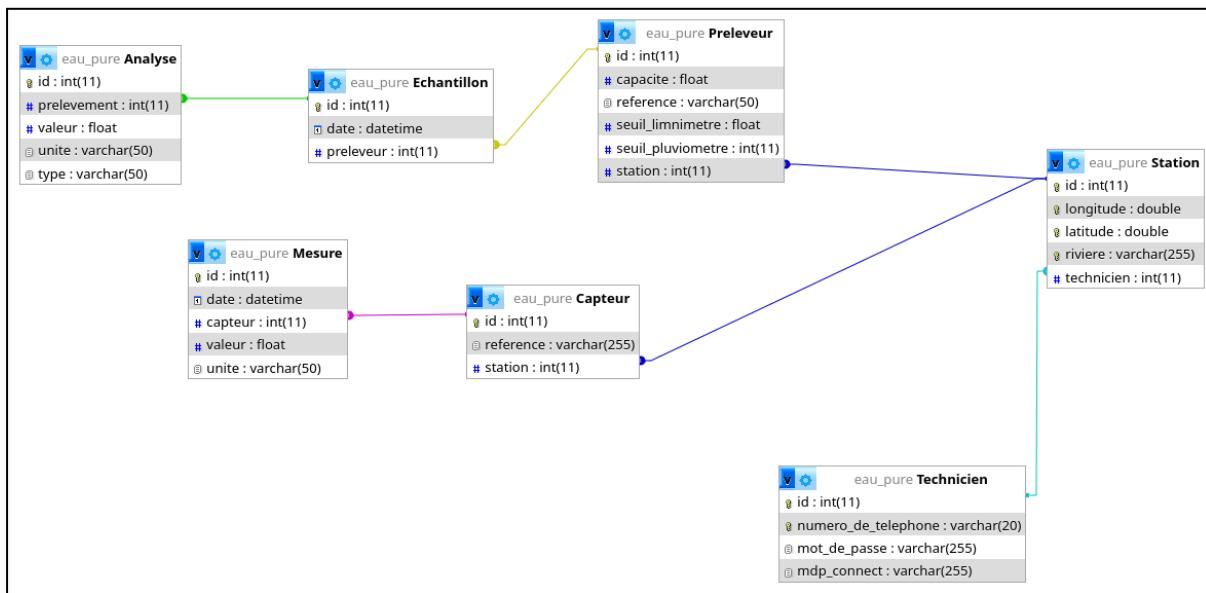


Figure n°6 : Schéma relationnel de la base de données "Eau Pure"

Ce schéma relationnel représente une base de données pour le suivi et l'analyse de la qualité de l'eau. Elle est composée de plusieurs tables interconnectées entre elles.

La table **Station** est centrale dans le système. Chaque station (identifiée par un *id*) est associée à un **Technicien** via la clé étrangère *technicien*. La station est également reliée à deux autres entités : les **Capteurs** et les **Préleveurs**, chacun ayant une clé étrangère *station* pointant vers la table Station. Cela signifie qu'un capteur ou un préleveur est toujours associé à une station spécifique.

Les **Capteurs** sont utilisés pour enregistrer des **Mesures**, chaque mesure étant associée à un capteur via le champ *capteur*. La table **Mesure** enregistre la date, la valeur mesurée et l'unité.

Les **Préleveurs**, quant à eux, sont responsables des **Échantillons**. Chaque échantillon est lié à un préleveur via le champ *prelevageur*. Ensuite, les **Analyses** sont réalisées sur ces échantillons ; chaque analyse fait référence à un *prelevement* (échantillon) et enregistre la valeur, l'unité et le type d'analyse.

Ainsi, les liaisons entre les tables peuvent être résumées comme ci-dessous :

- Une **Station** est gérée par un **Technicien**, et contient plusieurs **Capteurs** et **Préleveurs**.
- Un **Capteur** peut produire plusieurs **Mesures**.
- Un **Préleveur** peut produire plusieurs **Échantillons**.
- Un **Échantillon** peut faire l'objet de plusieurs **Analyses**.

Ce modèle permet un suivi précis de l'origine et de la qualité des données collectées dans le cadre de la surveillance environnementale. Nous verrons plus en détail la structure et les choix techniques de cette base de données dans la partie réalisée par A. Salou, responsable de la BDD.

## **4. Planification**

### **4.1. 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.

#### **4.2. 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.

#### **4.3. 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.

#### 4.4. Diagramme de Gantt

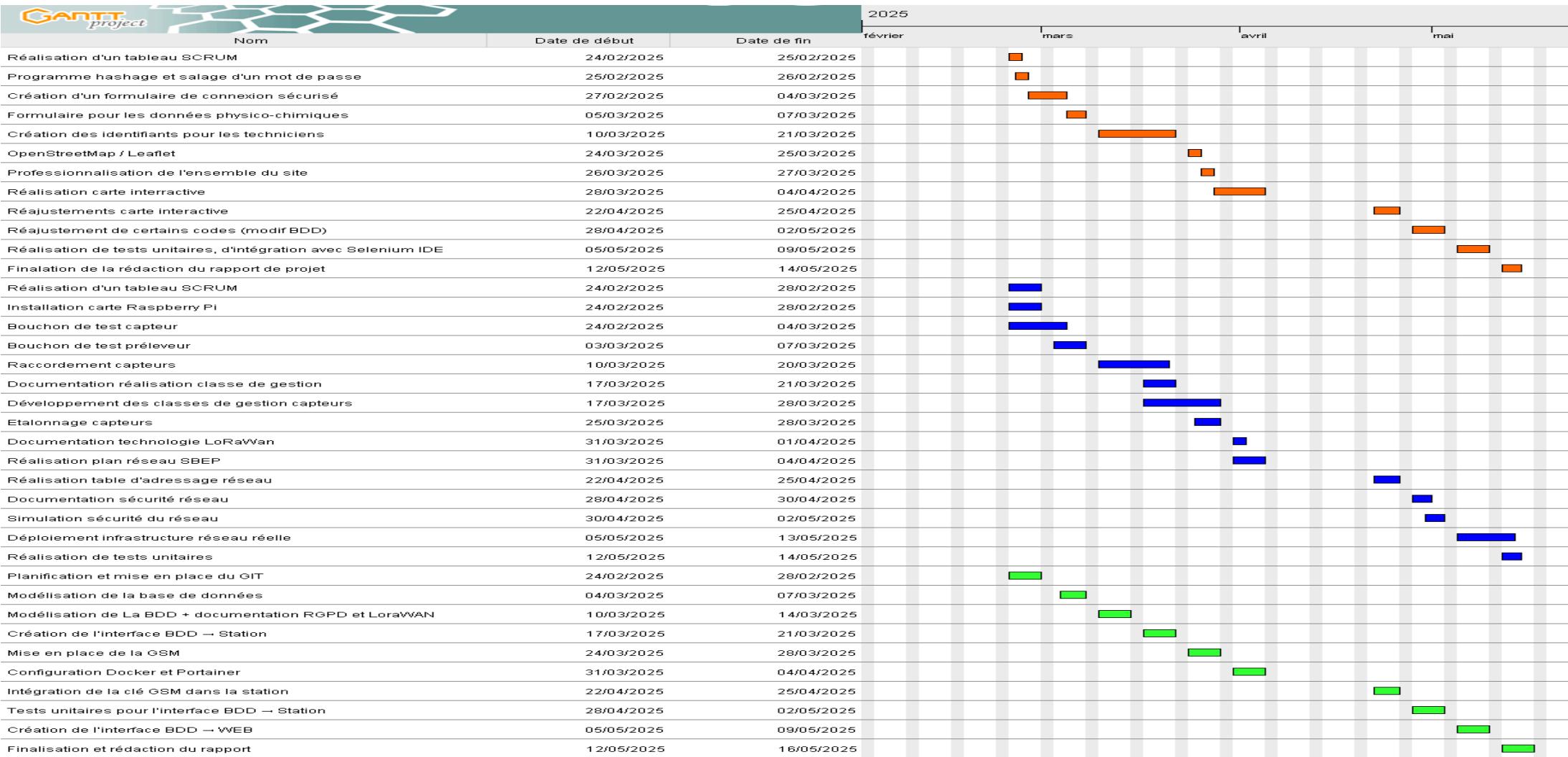


Figure n°7 : Diagramme de Gantt collectif – Suivi des tâches de l'équipe

## 5. Recette

| Section      | Critère                     | Description  | Résultat attendu  | Test de validation   |
|--------------|-----------------------------|--|---|--|
| Prélèvements | Déclenchement conditionnel  | Le prélèvement d'échantillons doit être déclenché automatiquement selon les conditions de pluviométrie et de hauteur d'eau | Échantillons prélevés automatiquement lors de fortes précipitations et pics de niveau | Simulation de pluie et vérification du déclenchement                         |
| Prélèvements | Échantillonnage régulier    | Prélèvement automatique 4 fois par jour en plus des événements particuliers  | 4 échantillons par jour et prélèvements lors d'intempéries                            | Contrôle des prélèvements sur une période de test                            |
| Notification | Notification de prélèvement | Le système doit envoyer une alerte au SBEP lorsqu'un prélèvement est effectué  | Notification envoyée automatiquement ( SMS)   | Vérification de réception de la notification après simulation de prélèvement |
| Mesures      | Mesure en continu           | Les capteurs mesurent le niveau d'eau et la pluviométrie en continu  | Données horodatées disponibles  | Analyse des données collectées sur 48h                                       |
| Mesures      | Transmission des données    | Les mesures doivent être envoyées régulièrement au SBEP  | Flux de données transmis (API)  | Interception et vérification des données envoyées                            |
| Publication  | Affichage Web               | Une interface utilisateur permet de visualiser les données   | Page web avec mesures graphiques et géolocalisation des stations                      | Consultation de la page web et comparaison avec mesures de l'analyse         |
| Réseau local | Réseau local fonctionnel    | Mise en place d'un réseau interne permettant la communication entre capteurs, serveurs et interface Web                    | Communication fluide entre les composants du système                                  | Tests de connectivité réseau (ping, transmission des informations)           |

Figure n°8 : Plan de validation fonctionnelle du système de surveillance environnementale

## 6. Conclusion

Depuis le début du projet, nous avons développé un système de collecte de données environnementales avec des capteurs connectés à une Raspberry Pi. Les données sont collectées, traitées et envoyées vers une base de données. Le raccordement du capteur limnimètre a posé des problèmes techniques, notamment la création d'un circuit et l'utilisation d'un convertisseur ADC. En parallèle, nous avons conçu le réseau et validé son infrastructure avec Cisco Packet Tracer, bien que l'intégration de LoRaWAN ait échoué.

Nous avons également créé une interface Web sécurisée pour saisir les résultats des analyses physico-chimiques, ce qui a demandé des recherches supplémentaires sur la gestion des utilisateurs. La visualisation des données sur une carte interactive a été réalisée avec React, Leaflet et OpenStreetMap, malgré un apprentissage nécessaire sur JavaScript. Enfin, l'automatisation du formatage des données pour la publication en open data a été explorée.

---

## Élève n°1 - Mathys LE SOURNE

---



|   |           |
|---|-----------|
| <b>Étudiant n°1 - Mathys LE SOURNE.....</b>                           | <b>20</b> |
| <b>1. Description du travail demandé.....</b>                         | <b>20</b> |
| <b>2. Tâche à réaliser.....</b>                                       | <b>20</b> |
| <b>3. Réalisation d'un tableau scrum.....</b>                         | <b>21</b> |
| <b>4. Tâches réalisées.....</b>                                       | <b>21</b> |
| <b>4.1. Le raspberry.....</b>   | <b>21</b> |
| <b>4.2. Réalisation bouchon de test des capteurs.....</b>             | <b>22</b> |
| <b>4.3. Réalisation bouchon de test préleveur.....</b>                | <b>22</b> |
| <b>4.4. Raccordement capteurs.....</b>                                | <b>23</b> |
| <b>4.4.1. Capteur limnimètre.....</b>                                 | <b>23</b> |
| <b>4.4.2. Capteur pluviomètre.....</b>                                | <b>25</b> |
| <b>4.5. Lecture des données.....</b>                                  | <b>27</b> |
| <b>5. Partie réseau.....</b>  | <b>29</b> |
| <b>5.1. Plan réseau.....</b>  | <b>29</b> |
| <b>5.2. Déploiement infrastructure.....</b>                           | <b>30</b> |
| <b>6. Sécurité.....</b>   | <b>33</b> |
| <b>8. Travail collaboratif.....</b>                                   | <b>35</b> |
| <b>9. Comparaison planning effectif au planning prévisionnel.....</b> | <b>35</b> |
| <b>10. Conclusion.....</b>  | <b>35</b> |
| <b>11. Procédure de test.....</b>                                     | <b>36</b> |
| <b>12. Planification.....</b>   | <b>39</b> |
| <b>13. Diagramme de gantt.....</b>                                    | <b>40</b> |
| <b>14. Les outils utilisés.....</b>                                   | <b>41</b> |
| <b>Annexes.....</b>   | <b>85</b> |
| <b>1. Etudiant n°1.....</b>   | <b>85</b> |

## Étudiant n°1 - Mathys LE SOURNE

### 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).

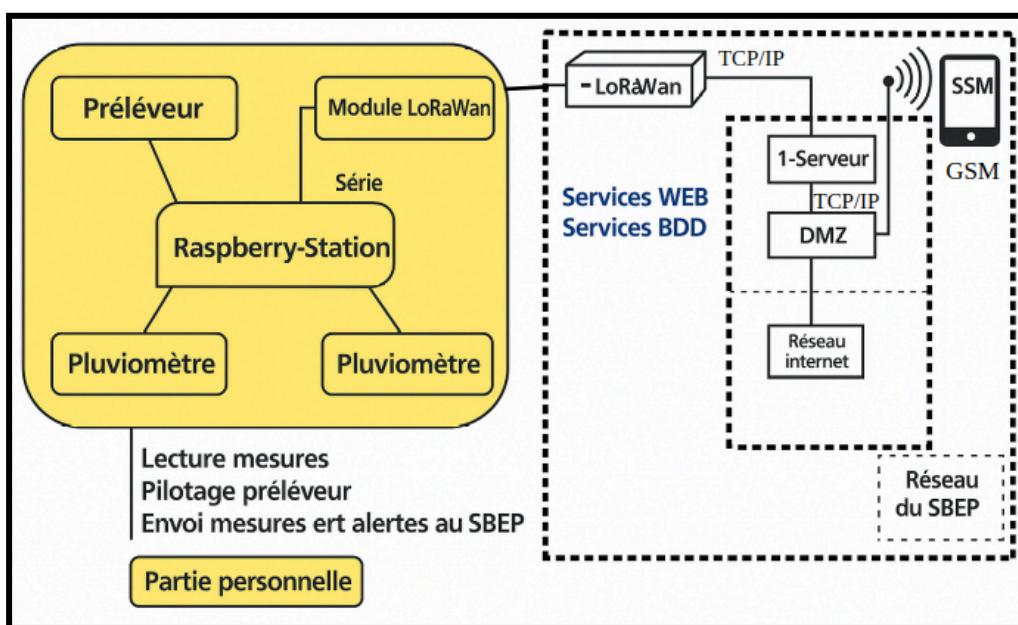


Figure n°1 : Diagramme de déploiement

### 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.

### 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.

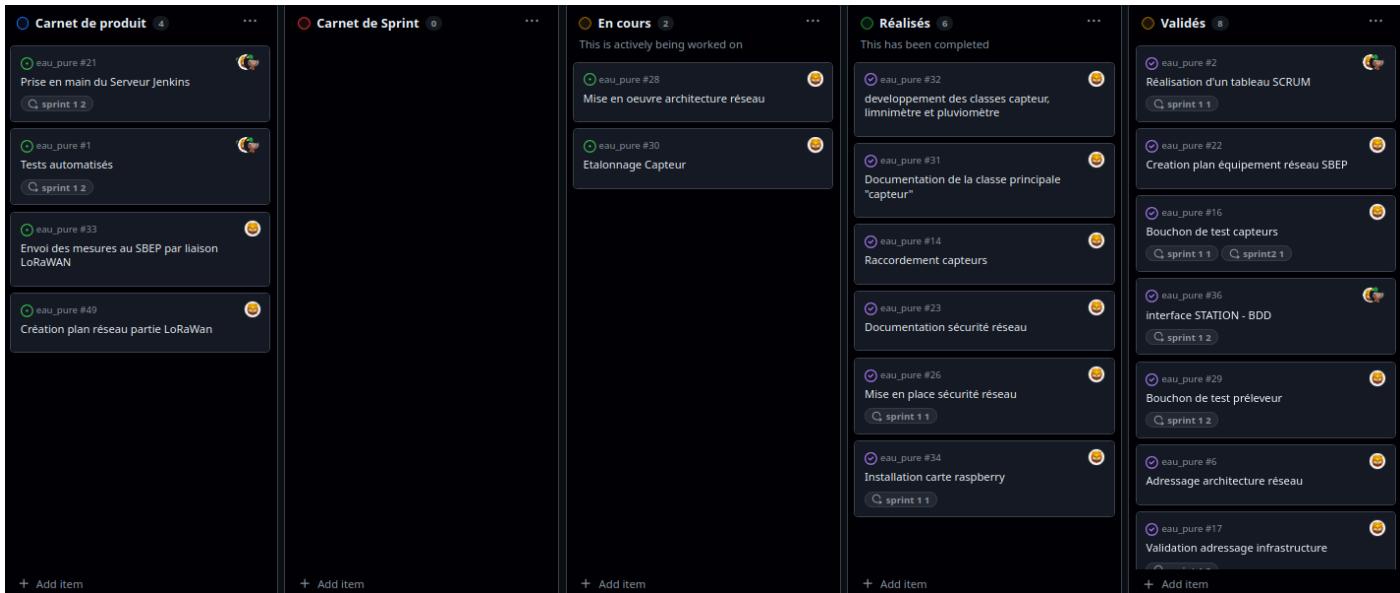


Figure n°2 : Tableau Scrum

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.

### 4. Tâches réalisées

#### 4.1. 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
```

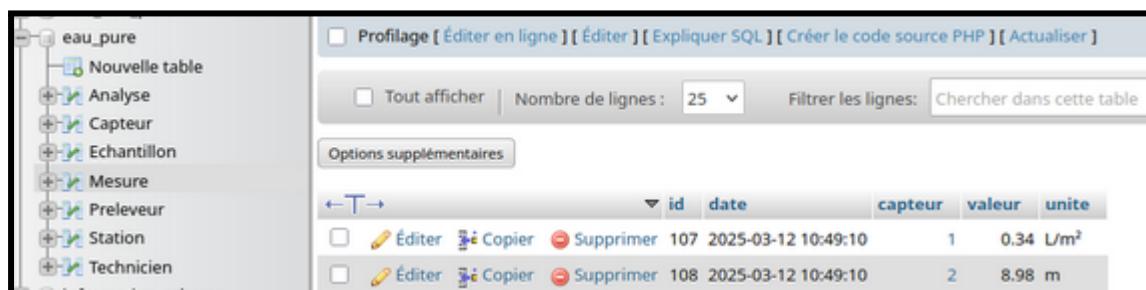
Figure n°3 : Commande création environnement virtuelle

**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).

#### 4.2. 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)



The screenshot shows a MySQL database interface with a sidebar containing tables: eau\_pure, Nouvelle table, Analyse, Capteur, Echantillon, Mesure, Prelevage, Station, and Technicien. The 'Capteur' table is selected. The main area displays the 'mesure' table with the following data:

|  |                                 | <b>id</b>                       | <b>date</b>                        | <b>capteur</b> | <b>valeur</b>       | <b>unite</b>            |
|--|---------------------------------|---------------------------------|------------------------------------|----------------|---------------------|-------------------------|
|  | <input type="checkbox"/> Éditer | <input type="checkbox"/> Copier | <input type="checkbox"/> Supprimer | 107            | 2025-03-12 10:49:10 | 1 0.34 L/m <sup>2</sup> |
|  | <input type="checkbox"/> Éditer | <input type="checkbox"/> Copier | <input type="checkbox"/> Supprimer | 108            | 2025-03-12 10:49:10 | 2 8.98 m                |

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

#### 4.3. 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**).

#### **4.4. Raccordement capteurs**

##### **4.4.1. 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.

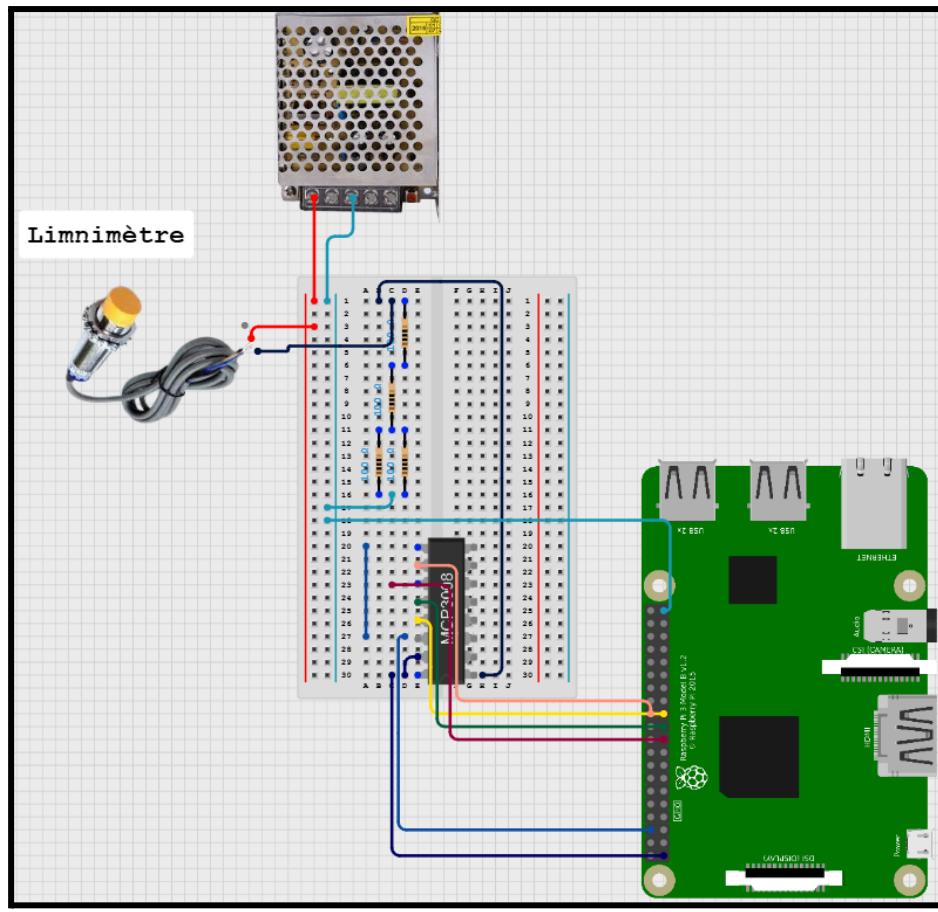


Figure n°5 : Schéma de raccordement capteur limnimètre

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°6 : Plan de branchement du convertisseur analogique-numérique MCP3208

#### 4.4.2. 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  $10.5 \text{ cm} \times 4.5 \text{ cm}$  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 ( $\text{L}/\text{m}^2$ ), 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  $1 \text{ g} = 1 \text{ mL}$ ). Ensuite, j'ai divisé ce volume par la surface de réception du capteur ( $47.25 \text{ cm}^2$ ) 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°7 : Capteur pluviomètre*

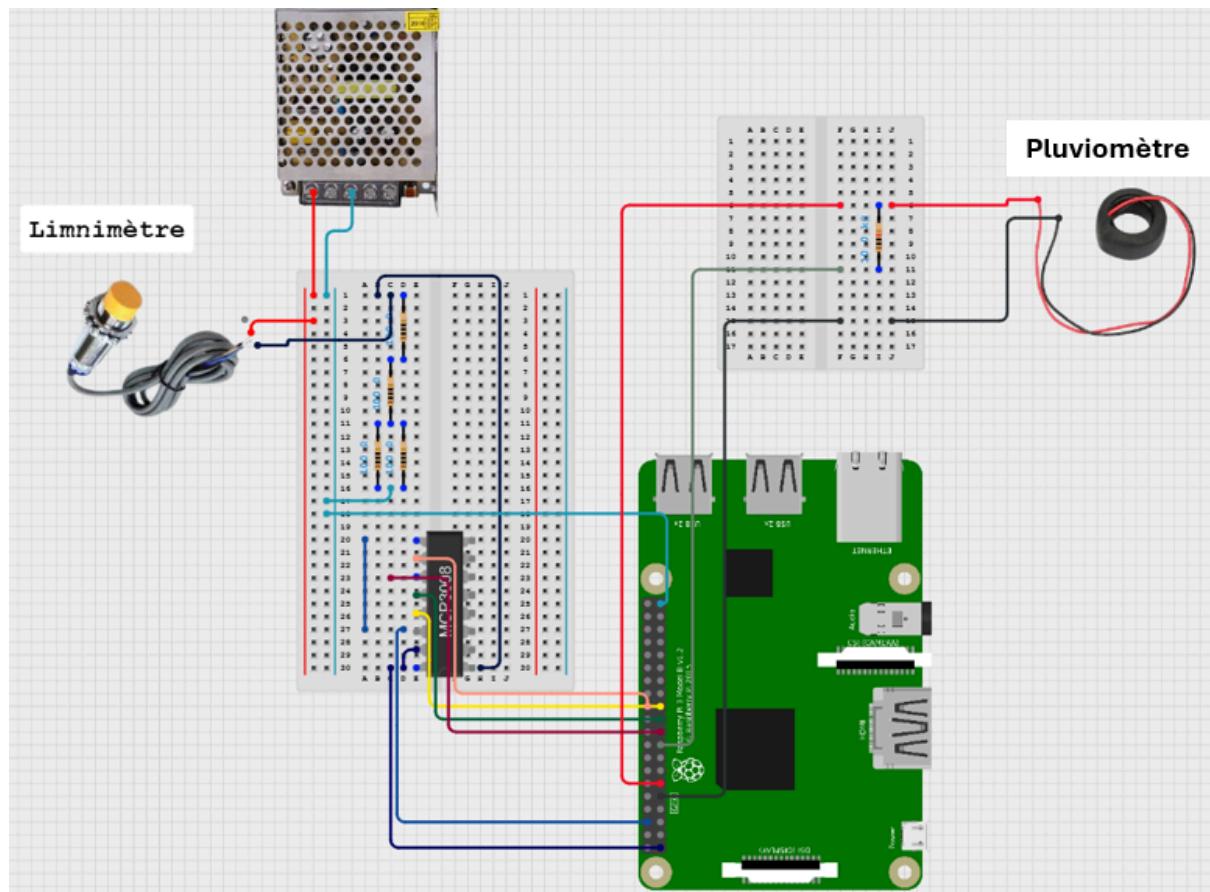


Figure n°8 : Schéma de raccordement capteur pluviometre

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.

#### 4.5. 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 (**annexe 3**). 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 (**annexe 4**).

```
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°9 : Résultat execution programme terminal

| eau_pure |  | Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer 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°10 : 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 (**annexe 5**).

```
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°11 : Résultat execution programme terminal

| Profilage [ Editer en ligne ] [ Editer ] [ Expliquer SQL ] [ Créer le code source PHP ] [ Actualiser ] |  |                         |                          |                           |         |                                |
|--|--|-------------------------|--------------------------|---------------------------|---------|--------------------------------|
|  |  | << < 4 > >>             | Tout afficher            | Nombre de lignes :        | 25      | Filtrer les lignes:            |
|  |  | Options supplémentaires |                          |                           |         |                                |
|  |  | ← T →                   | id                       | date                      | capteur | valeur unite                   |
|  |  |                         | <input type="checkbox"/> | Éditer  Copier  Supprimer | 785     | 2025-04-23 17:21:57 1 2.8 L/m² |

Figure n°12 : 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.

Nous avons 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 (**annexe 6**).

```
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
```

| Profilage [ Editer en ligne ] [ Editer ] [ Expliquer SQL ] [ Créer le code source PHP ] [ Actualiser ] |  |                         |                          |                           |         |                                 |
|--|--|-------------------------|--------------------------|---------------------------|---------|---------------------------------|
|  |  | << < 4 > >>             | Tout afficher            | Nombre de lignes :        | 25      | Filtrer les lignes:             |
|  |  | Options supplémentaires |                          |                           |         |                                 |
|  |  | ← T →                   | id                       | date                      | capteur | valeur unite                    |
|  |  |                         | <input type="checkbox"/> | Éditer  Copier  Supprimer | 802     | 2025-04-24 14:10:43 1 3.15 L/m² |
|  |  |                         | <input type="checkbox"/> | Éditer  Copier  Supprimer | 803     | 2025-04-24 14:10:43 2 0.64 m    |

Figure n°13 : Résultat execution programme main.py

## 5. Partie réseau

### 5.1. 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.

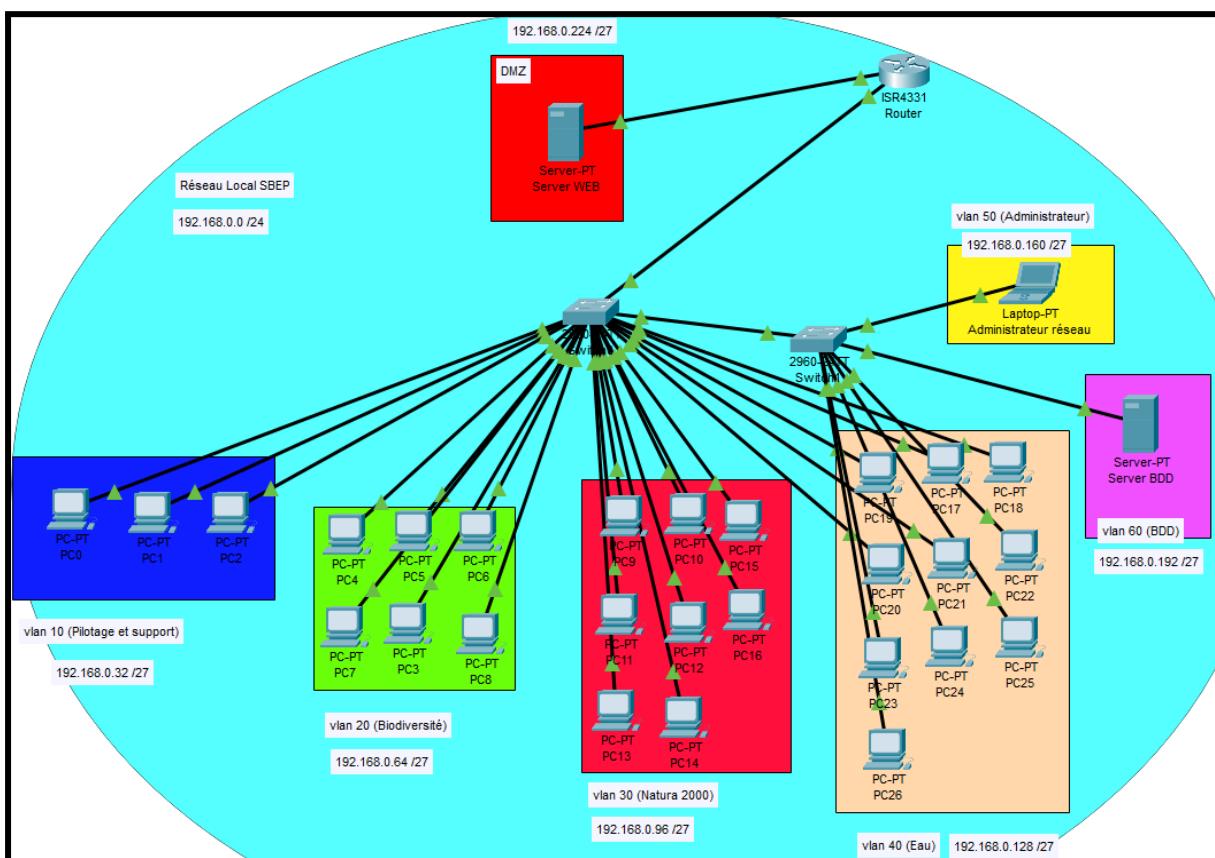


Figure n°14 : Schéma réseau local SBEP

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.

## 5.2. 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°15 : Configuration vlans

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

```
Switch#sh vlan

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 fddinet-default                      act/unsup
1005 trnet-default                        act/unsup
```

Figure n°16 : 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<br>Fa0/7, Fa0/8, Fa0/9, Fa0/10<br>Fa0/11, Fa0/12, Fa0/13, Fa0/14<br>Fa0/15, Fa0/16, Fa0/17, Fa0/18<br>Fa0/19, Gi0/1 |
| 10   | Pilotage_et_support | active    |  |
| 20   | Biodiversite        | active    |  |
| 30   | Natura_2000         | active    |  |
| 40   | Eau                 | active    | Fa0/20, Fa0/21, Fa0/22, Fa0/23<br>Fa0/24   |
| 50   | Administrateur      | active    | Fa0/1  |
| 60   | BDD                 | active    | Fa0/2  |
| 1002 | fdci-default        | act/unsup |  |
| 1003 | token-ring-default  | act/unsup |  |
| 1004 | fdinnet-default     | act/unsup |  |
| 1005 | trnet-default       | act/unsup |  |

Figure n°17 : Association port du switch 2 aux vlan

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

```
configure terminal
interface GigabitEthernet1
switchport mode trunk
switchport trunk encapsulation dot1q
no shutdown
exit
```

Figure n°18 : 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.

Ensuite 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.

L'image montre la configuration du routeur, où l'interface principale GigabitEthernet4 est subdivisée en sous-interfaces, chacune associée à un VLAN spécifique (encapsulation dot1Q) et configurée avec une adresse IP et un masque de sous-réseau. Les VLANs numérotés de 10 à 60 disposent de plages IP distinctes, et une ACL (ACCESS\_BDD) est appliquée sur le VLAN 60 pour sécuriser le trafic entrant. Une autre interface, GigabitEthernet5, est configurée avec une adresse IP (192.168.0.254) pour la DMZ. Cette configuration illustre une segmentation réseau efficace, facilitant une gestion optimisée des ressources et renforçant la sécurité grâce à l'isolation des VLANs et l'application de règles spécifiques.

```

interface GigabitEthernet4
no ip address
duplex auto
speed auto
!
interface GigabitEthernet4.10
encapsulation dot1Q 10
ip address 192.168.0.62 255.255.255.224
!
interface GigabitEthernet4.20
encapsulation dot1Q 20
ip address 192.168.0.94 255.255.255.224
!
interface GigabitEthernet4.30
encapsulation dot1Q 30
ip address 192.168.0.126 255.255.255.224
!
interface GigabitEthernet4.40
encapsulation dot1Q 40
ip address 192.168.0.158 255.255.255.224
!
interface GigabitEthernet4.50
encapsulation dot1Q 50
ip address 192.168.0.190 255.255.255.224
!
interface GigabitEthernet4.60
encapsulation dot1Q 60
ip address 192.168.0.222 255.255.255.224
ip access-group ACCESS_BDD in
!
interface GigabitEthernet5
ip address 192.168.0.254 255.255.255.224
duplex auto
speed auto

```

Figure n°19 : Configuration sous interface sur le routeur

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

```

configure terminal
interface GigabitEthernet4.10
encapsulation dot1Q 10
ip address 192.168.0.62 255.255.255.224
no shutdown
exit

```

Figure n°20 : Crédit de sous interface sur le routeur

## 6. 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°21 : 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 SSH
no ip ssh version 2
no ip ssh
!
# Désactiver Telnet (en supprimant les lignes vty associées)
line vty 0 15
transport input none
exit
!
end
write memory
```

Figure n°22 : 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.

```
ip access-list extended ACCESS_BDD
  permit ip 192.168.0.192 0.0.0.31 192.168.0.128 0.0.0.31
  permit ip 192.168.0.192 0.0.0.31 192.168.0.160 0.0.0.31
  deny   ip 192.168.0.192 0.0.0.31 192.168.0.32 0.0.0.31
  deny   ip 192.168.0.192 0.0.0.31 192.168.0.64 0.0.0.31
  deny   ip 192.168.0.192 0.0.0.31 192.168.0.96 0.0.0.31
  permit ip 192.168.0.192 0.0.0.31 192.168.0.224 0.0.0.31
```

Figure n°23 : Liste de contrôle d'accès

Puis j'ai appliqué l'ACL à l'interface GE 4.60 (sous interface vlan 60) du routeur.

```
configure terminal
interface GigabitEthernet4.60
  ip access-group ACCESS_BDD in
exit
```

Figure n°24 : 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. 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.

## 7. 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.

## **8. 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.

## **9. 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.

## **10. Conclusion**

Pour conclure, j'ai principalement travaillé sur la partie capteurs du projet, qui est désormais fonctionnelle. Les données des capteurs sont correctement collectées, traitées et envoyées dans la base de données. Cependant, le raccordement du capteur limnimètre a présenté des difficultés, notamment la nécessité de concevoir un circuit sur une plaque de test et d'utiliser un convertisseur ADC pour assurer la communication avec la carte Raspberry Pi. Malgré ces problèmes, le développement des programmes Python pour la lecture des données des capteurs a été réalisé rapidement.

Concernant la partie réseau, j'ai élaboré le plan du réseau ainsi que la table d'adressage. J'ai également effectué des tests sur Cisco Packet Tracer pour valider l'infrastructure avant son déploiement sur le matériel physique. Tout au long du projet, des tests ont été réalisés pour garantir le bon fonctionnement du réseau.

Cependant, en raison des difficultés rencontrées avec la technologie LoRaWAN, cette étape n'a pas pu être intégrée dans le projet. Cela représente une piste d'amélioration.

## 11. Procédure de test

Fiche de test n°1

|                      |   |
|----------------------|---|
| <b>Élément testé</b> | Le capteur limnimètre est hors de l'eau   |
| <b>Objectif</b>      | Traitement de la valeur minimale délivré par le limnimètre  |
| <b>Pré-requis</b>    | Un circuit électronique est monté sur une plaque de test comportant plusieurs résistances de 100 ohms, combinées pour obtenir une résistance équivalente de 250 ohms. Les mesures sont réalisées à l'aide d'un voltmètre. Le montage utilise une alimentation de 24V, une colonne d'eau de 1,30 m de hauteur, un capteur limnimètre, ainsi qu'une carte Raspberry Pi sur laquelle un programme en Python est exécuté. |

| Action   | Résultat attendu  | Observations |
|--|---|--------------|
| Brancher le voltmètre aux bornes de la résistance équivalente. | Le voltmètre affiche la valeur 0 volt                                     |              |
| Mettre sous tension le circuit avec l'alimentation (24v)       | Le voltmètre mesure une valeur en volt                                    |              |
| Mettre le capteur limnimètre hors de l'eau                     | le limnimètre ne se trouve plus dans l'eau                                |              |
| Sur le raspberry lancer l'invite de commande                   | un terminal est ouvert avec le prompt affiché                             |              |
| Lancer le programme python main.py                             | Le programme affiche "détexion des mesures en cours"                      |              |
| Attendre 1 minute pour que le programme se termine             | Le programme python affiche "Le capteur limnimètre n'est pas dans l'eau". |              |
| Conclusion   |   |              |

Fiche de test n°2

|                      |   |
|----------------------|---|
| <b>Élément testé</b> | Réalisation d'une mesure dans la colonne d'eau avec le limnimètre   |
| <b>Objectif</b>      | Vérifier que la valeur calculée correspond à la mesure réalisée   |
| <b>Pré-requis</b>    | Un circuit électronique est monté sur une plaque de test comportant plusieurs résistances de 100 ohms, combinées pour obtenir une résistance équivalente de 250 ohms. Les mesures sont réalisées à l'aide d'un voltmètre. Le montage utilise une alimentation de 24V, une colonne d'eau de 1,30 m de hauteur, un capteur limnimètre, ainsi qu'une carte Raspberry Pi sur laquelle un programme en Python est exécuté. |

Fiche de test n°3

|                      |   |
|----------------------|---|
| <b>Élément testé</b> | Traitement du signal de sortie de capteur limnimètre  |
| <b>Objectif</b>      | Une erreur est détecté si le capteur est débranché  |
| <b>Pré-requis</b>    | Un circuit électronique est monté sur une plaque de test comportant plusieurs résistances de 100 ohms, combinées pour obtenir une résistance équivalente de 250 ohms. Les mesures sont réalisées à l'aide d'un voltmètre. Le montage utilise une alimentation de 24V, une colonne d'eau de 1,30 m de hauteur, un capteur limnimètre, ainsi qu'une carte Raspberry Pi sur laquelle un programme en Python est exécuté. |

| Action   | Résultat attendu   | Observations |
|--|--|--------------|
| Brancher le voltmètre aux bornes de la résistance équivalente. | Le voltmètre affiche la valeur 0 volt                                      |              |
| Mettre sous tension le circuit avec l'alimentation (24v)       | Le voltmètre mesure une valeur en volt                                     |              |
| Mettre le capteur limnimètre dans la colonne d'eau             | le limnimètre se trouve dans l'eau   |              |
| Sur le raspberry lancer l'invite de commande                   | un terminal est ouvert avec le prompt affiché                              |              |
| Lancer le programme python main.py                             | Le programme affiche "détection des mesures en cours"                      |              |
| Attendre 1 minute pour que le programme se termine             | Le programme python affiche "Profondeur envoyée : xx m (V=xx V, I=xx mA)". |              |
| Conclusion   |  |              |

| Action   | Résultat attendu  | Observations |
|--|---|--------------|
| Brancher le voltmètre aux bornes de la résistance équivalente. | Le voltmètre affiche la valeur 0 volt   |              |
| Mettre sous tension le circuit avec l'alimentation (24v)       | Le voltmètre mesure une valeur en volt  |              |
| Sur le raspberry lancer l'invite de commande                   | un terminal est ouvert avec le prompt affiché   |              |
| Lancer le programme python main.py                             | Le programme affiche "détection des mesures en cours"   |              |
| Débrancher le fil d'entrée du capteur                          | Le voltmètre affiche la valeur 0 volt et une erreur est affichée dans le terminal "Erreur : Le capteur est débranché ! intensité trop faible" |              |
| Conclusion   |   |              |

Fiche de test n°4

|                      |   |
|----------------------|---|
| <b>Élément testé</b> | Communication entre les différents vlans (10, 20, 30, 40, 50) et serveur Web                              |
| <b>Objectif</b>      | Les ordinateurs des vlans communiquent entre-eux  |
| <b>Pré-requis</b>    | Les switchs et le routeur sont paramétrés, les tests sont des ping effectué depuis un invite de commande. |

Fiche de test n°5

|                      |   |
|----------------------|---|
| <b>Élément testé</b> | Communication entre les différents vlans (10, 20, 30, 40, 50) et la base de donnée                        |
| <b>Objectif</b>      | Uniquement les pc des vlan 40 et 50 peuvent communiquer avec la base de donnée (vlan 60)                  |
| <b>Pré-requis</b>    | Les switchs et le routeur sont paramétrés, les tests sont des ping effectué depuis un invite de commande. |

| Action  | Résultat attendu   | Observations |
|---|--|--------------|
| Depuis l'ordinateur accéder à l'invite de commande        | La commande ping est lancé sur l'invite de commande vers un vlan distant |              |
| Depuis un PC du VLAN 10 :<br>Ping -c 3 -W 1 <pc vlan 20>  | 3 packets transmitted, 3 received, 0% packet loss                        |              |
| Depuis un PC du VLAN 10 :<br>Ping -c 3 -W 1 <pc vlan 30>  | 3 packets transmitted, 3 received, 0% packet loss                        |              |
| Depuis un PC du VLAN 10 :<br>Ping -c 3 -W 1 <pc vlan 40>  | 3 packets transmitted, 3 received, 0% packet loss                        |              |
| Depuis un PC du VLAN 10 :<br>Ping -c 3 -W 1 <pc vlan 50>  | 3 packets transmitted, 3 received, 0% packet loss                        |              |
| Depuis un PC du VLAN 10 :<br>Ping -c 3 -W 1 <Serveur WEB> | 3 packets transmitted, 3 received, 0% packet loss                        |              |
| Conclusion  |  |              |

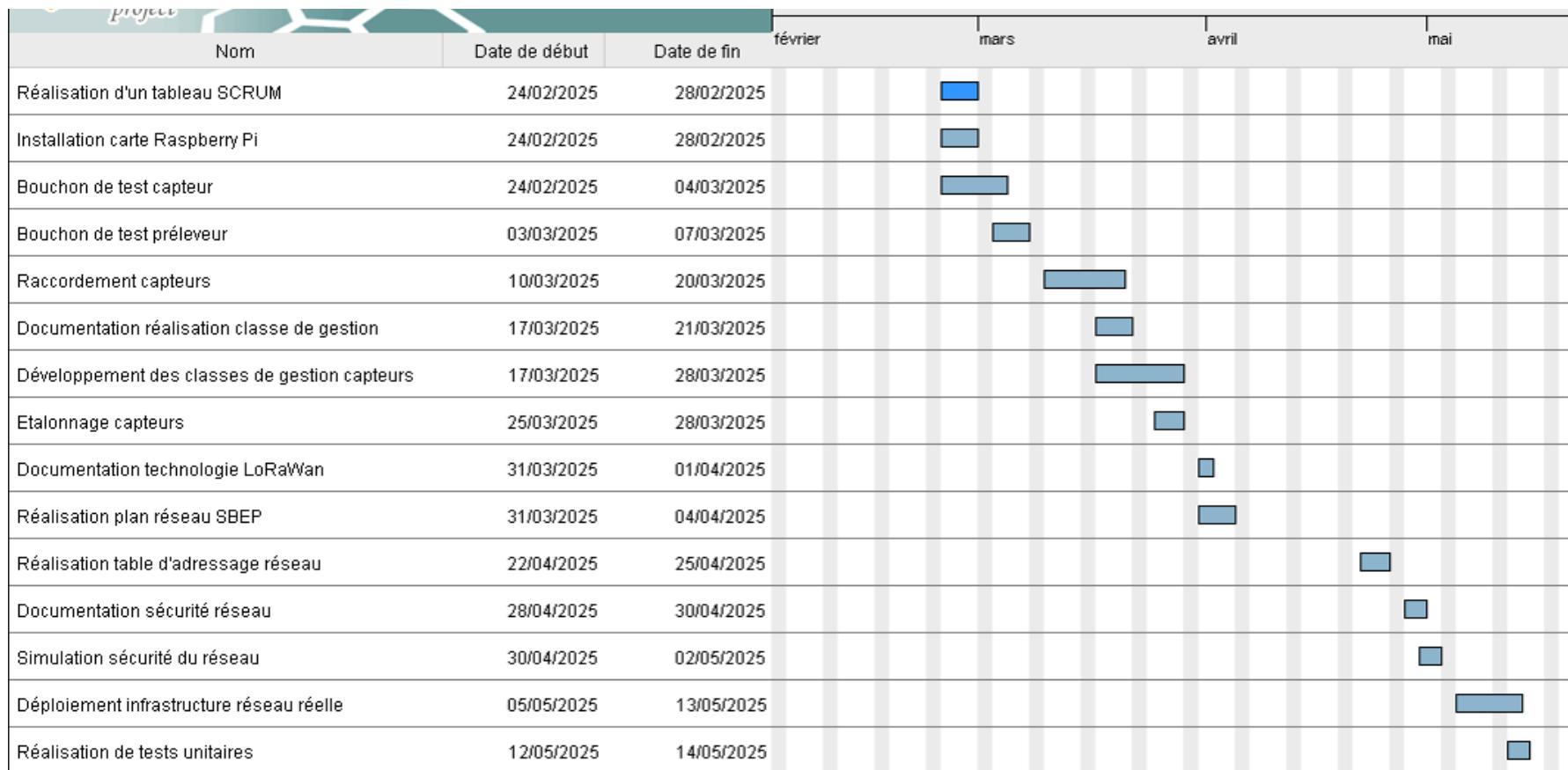
| Action  | Résultat attendu   | Observations |
|---|--|--------------|
| Depuis l'ordinateur accéder à l'invite de commande        | La commande ping est lancé sur l'invite de commande vers un vlan distant |              |
| Depuis un PC du VLAN 10 :<br>Ping -c 3 -W 1 <BDD vlan 60> | 3 packets transmitted, 0 received, 100% packet loss                      |              |
| Depuis un PC du VLAN 20 :<br>Ping -c 3 -W 1 <BDD vlan 60> | 3 packets transmitted, 0 received, 100% packet loss                      |              |
| Depuis un PC du VLAN 30 :<br>Ping -c 3 -W 1 <BDD vlan 60> | 3 packets transmitted, 0 received, 100% packet loss                      |              |
| Depuis un PC du VLAN 40 :<br>Ping -c 3 -W 1 <BDD vlan 60> | 3 packets transmitted, 3 received, 0% packet loss                        |              |
| Depuis un PC du VLAN 50 :<br>Ping -c 3 -W 1 <BDD vlan 60> | 3 packets transmitted, 3 received, 0% packet loss                        |              |
| Conclusion  |  |              |

## 12. Planification

| Semaine | Nombre d'heure | Tâches  | Etat de la tâche   |
|---------|----------------|---|--|
| 1       | 15 heures      | <ul style="list-style-type: none"> <li>Réalisation tableau scrum</li> <li>installation carte raspberry</li> <li>Bouchon test capteur</li> </ul>   | <span style="color: red;">♥</span><br><span style="color: red;">♥</span><br><span style="color: black;">●</span>   |
| 2       | 17 heures      | <ul style="list-style-type: none"> <li>Bouchon test capteur</li> <li>Bouchon test préleveur</li> </ul>  | <span style="color: red;">♥</span><br><span style="color: red;">♥</span>   |
| 3       | 17 heures      | <ul style="list-style-type: none"> <li>Raccordement capteurs</li> </ul>   | <span style="color: black;">■</span>   |
| 4       | 17 heures      | <ul style="list-style-type: none"> <li>Raccordement capteurs</li> <li>Documentation pour la réalisation classes de gestion</li> <li>Développement des classes de gestion des capteurs</li> </ul>  | <span style="color: red;">♥</span><br><span style="color: red;">♥</span><br><span style="color: black;">●</span>   |
| 5       | 16 heures      | <ul style="list-style-type: none"> <li>Développement des classes de gestion des capteurs</li> <li>Etalonnage capteurs</li> </ul>  | <span style="color: red;">♥</span><br><span style="color: red;">♥</span>   |
| 6       | 17 heures      | <ul style="list-style-type: none"> <li>Documentation technologie LoRaWan</li> <li>Intégration de LoRaWan</li> <li>Réalisation du plan du réseau SBEP</li> </ul>                                   | <span style="color: red;">♥</span><br><span style="color: black;">■</span><br><span style="color: black;">●</span> |
| 7       | 9 heures       | <ul style="list-style-type: none"> <li>Intégration de LoRaWan</li> <li>Réalisation du plan du réseau SBEP (Cisco packet tracer)</li> <li>Réalisation de la table d'adressage du réseau</li> </ul> | <span style="color: black;">■</span><br><span style="color: red;">♥</span><br><span style="color: red;">♥</span>   |
| 8       | 10 heures      | <ul style="list-style-type: none"> <li>Intégration de LoRaWan</li> <li>Documentation sécurité réseau</li> <li>Simulation sécurité sur le réseau (Cisco packet tracer)</li> </ul>                  | <span style="color: black;">■</span><br><span style="color: red;">♥</span><br><span style="color: red;">♥</span>   |
| 9       | 10 heures      | <ul style="list-style-type: none"> <li>Intégration de LoRaWan</li> <li>Déploiement de l'infrastructure réseau réelle</li> </ul>   | <span style="color: black;">●</span><br><span style="color: black;">■</span>                                       |
| 10      | 21 heures      | <ul style="list-style-type: none"> <li>Déploiement de l'infrastructure réseau réelle</li> <li>Réalisation de test unitaire, d'intégration</li> </ul>  | <span style="color: red;">♥</span><br><span style="color: red;">♥</span>   |

♥ : Tâche réalisé | ● : Tâche en cours | ■ : Tâche non réalisé

### 13. Diagramme de gantt



## 14. 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.

---

## Élève n°2 - Alexis SALOU

---



|  |    |
|--|----|
| 1. Analyse / Aspect fonctionnel.....               | 43 |
| 1.1 Contexte et Objectifs du Projet.....           | 43 |
| 1.2 Description du Système Global.....             | 43 |
| 1.3 Objectifs techniques.....                      | 44 |
| 2. Travail réalisé.....                            | 44 |
| 2.1 Conception de la Base de Données.....          | 44 |
| 2.2 Développement de l'Interface Python.....       | 46 |
| 2.3 Développement de l'Interface PHP.....          | 47 |
| 2.4 Intégration de la Clé GSM.....                 | 47 |
| 3. Tests Fonctionnels.....                         | 48 |
| 3.1 Tests Unitaires.....                           | 48 |
| 4. Environnement Technique / Contraintes.....      | 49 |
| 4.1 Choix de l'Environnement Dockerisé.....        | 49 |
| 5. Architecture Matérielle / Choix Techniques..... | 50 |
| 5.1 Matériel Utilisé.....                          | 50 |
| 5.2 Architecture du Système.....                   | 51 |
| 6. Conclusion et Perspectives.....                 | 52 |
| 6.1 Bilan du Projet.....                           | 52 |
| 6.2 Limitations et Améliorations.....              | 52 |
| 6.3 Perspectives d'Évolution.....                  | 53 |
| 6.4 Conclusion Finale.....                         | 54 |

## Étudiant n°2 - Alexis SALOU

### 1. Analyse / Aspect fonctionnel

#### 1.1 Contexte et Objectifs du Projet

Le projet "**Eau Pure**" vise à améliorer la gestion des données environnementales, spécifiquement pour la surveillance des niveaux d'eau dans une station de mesure, le suivi de la pluviométrie.

#### Objectif principal :

Créer une architecture logicielle permettant de collecter des données en temps réel, de les stocker de manière sécurisée, et de les visualiser à travers une interface web accessible.

Ce projet répond à un besoin de suivi environnemental crucial dans les secteurs tels que la gestion des inondations ou de la sécheresse. Par exemple, si le niveau de l'eau dépasse un seuil critique, le système envoie automatiquement une alerte par SMS à un technicien pour qu'il puisse intervenir immédiatement. Ce système a donc une dimension à la fois **technique** (collecte et traitement des données) et **opérationnelle** (réactivité face aux événements critiques).

#### 1.2 Description du Système Global

Le système développé pour ce projet est composé de plusieurs composants interconnectés pour fonctionner efficacement :

##### 1. Capteurs :

- Ces dispositifs mesurent en continu les niveaux d'eau, la pluviométrie, et l'état du prélèvement. Les capteurs envoient des données à la Raspberry Pi de manière périodique.

##### 2. Raspberry Pi :

- C'est le cœur du système. Elle récupère les données des capteurs et les envoie à un serveur. En outre, la Raspberry Pi est responsable de la gestion du système d'alerte SMS lorsque des seuils critiques sont atteints, grâce à une clé GSM.

##### 3. Base de données MySQL :

- Toutes les données collectées sont stockées dans une base de données relationnelle MySQL. La base organise les mesures sous forme de tables (stations, capteurs, mesures, technicien, etc.), ce qui permet une gestion sécurisée et évolutive des données.

#### 4. Interface Web (PHP) :

- Un site web a été conçu pour permettre la visualisation des données en temps réel. Ce site permet à un technicien de consulter les données et de recevoir des alertes lorsqu'un seuil critique est franchi.

#### 5. Système d'Alerte SMS :

- Dès que les mesures dépassent un seuil critique, une alerte est envoyée automatiquement à un technicien via SMS grâce à une clé GSM Huawei E3531, connectée à la Raspberry Pi. Cela permet d'assurer une réponse rapide en cas d'urgence.

#### 1.3 Objectifs techniques

Le projet a pour objectif de fournir une solution **fiable, évolutive, et facile à déployer**, avec les spécifications suivantes :

- **Modularité** : Le système doit être suffisamment flexible pour permettre l'ajout de nouveaux capteurs ou fonctionnalités sans complexité.
- **Sécurité** : Assurer la protection des données sensibles et garantir l'intégrité des communications entre les différents composants.

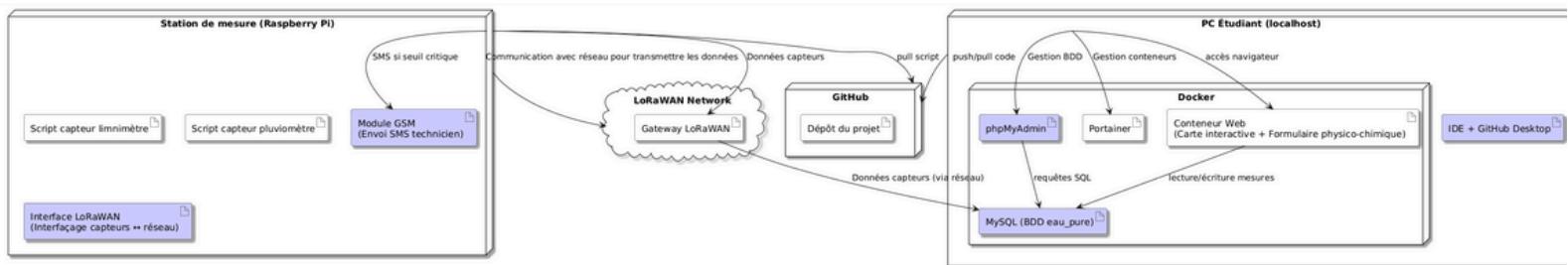


Image n°1 : Schéma du projet global

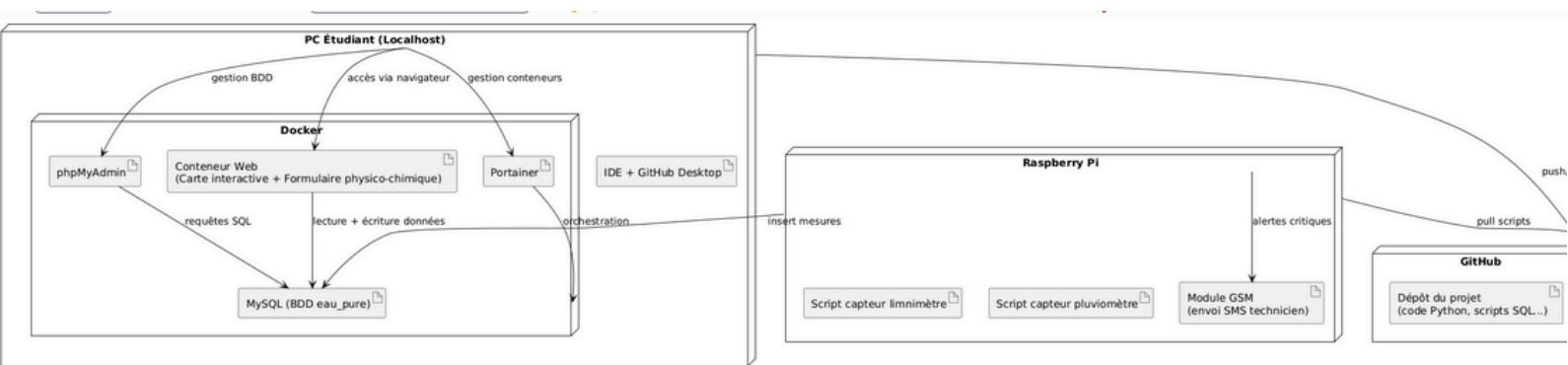


Diagramme n°1 : Diagramme UML

## 2. Travail réalisé

|            |   |
|------------|---|
| Étudiant 2 | <ul style="list-style-type: none"> <li>• Installation et paramétrage d'une passerelle LoRaWAN. Installation et paramétrage d'un serveur The Think Stack (The Think Network)</li> <li>• 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.</li> <li>• 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.</li> <li>• Paramétrage du serveur The Think Stack afin de communiquer au site Web les données.</li> <li>• 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)</li> <li>• 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.</li> </ul> |
|------------|---|

image n°2 : cahier des charges

### 2.1 Conception de la Base de Données

La base de données MySQL a été conçue pour être robuste et évolutive, afin de garantir une gestion fiable des données collectées par les capteurs.

#### Modélisation de la base de données :

Pour concevoir la base de données, j'ai utilisé l'outil PhpMyAdmin afin de créer un schéma adapté aux besoins du système. Le schéma comprend plusieurs tables principales, chacune remplissant un rôle spécifique dans le stockage et la gestion des données.

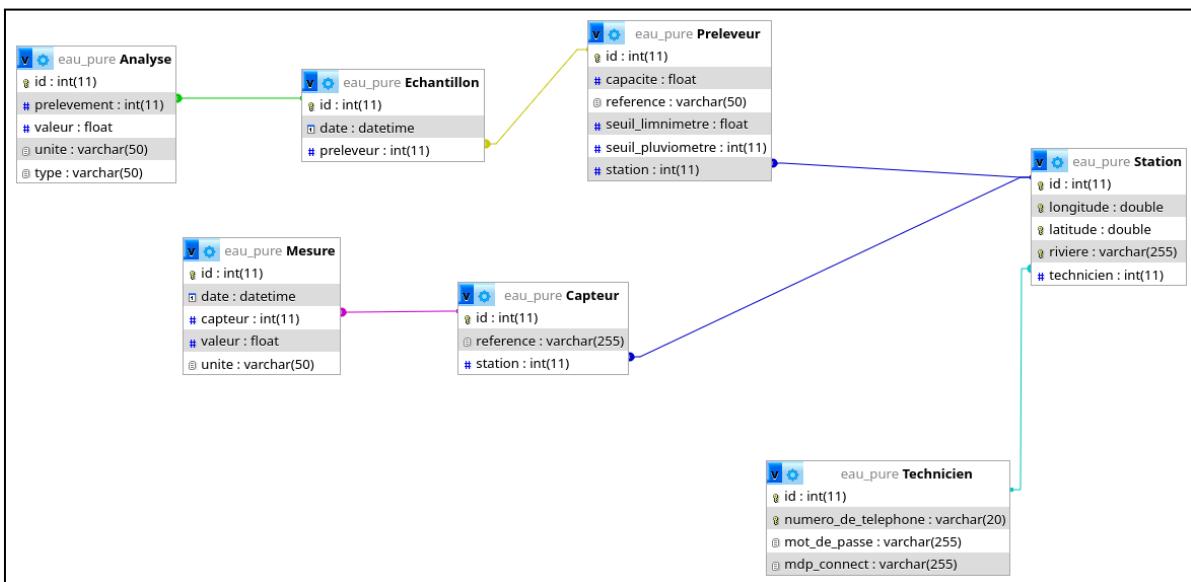
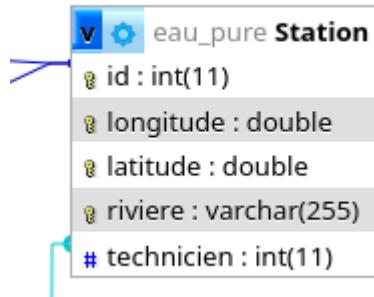


Image n°3 : Modélisation Base de donnée

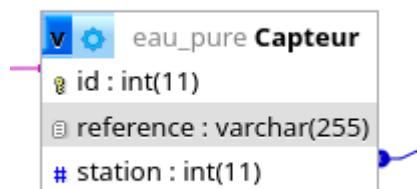
## 1. Table stations :



Cette table contient des informations sur les stations de mesure. Les colonnes principales incluent :

- **ID** : Identifiant unique de la station.
- **rivière** : Le nom de la rivière de la station.
- **Longitude** : La longitude de la station.
- **Latitude** : La latitude de la station
- **Technicien** : Le technicien relié à la station

## 2. Table capteurs :



Cette table enregistre les types de capteurs installés dans chaque station. Les colonnes principales sont :

- **ID** : Identifiant unique du capteur.
- **Type** : Type de capteur (par exemple, 1 pour pluviomètre, 2 pour limmomètre).

- **référence** : la ref du capteur

### 3. Table mesures :

| eau_pure Mesure |                            |
|-----------------|----------------------------|
| !               | <b>id</b> : int(11)        |
| !               | <b>date</b> : datetime     |
| #               | <b>capteur</b> : int(11)   |
| #               | <b>valeur</b> : float      |
| !               | <b>unite</b> : varchar(50) |

Cette table stocke les valeurs mesurées par les capteurs. Elle contient des colonnes comme :

- **ID** : Identifiant unique de la mesure.
- **Unité** : L'unité de la mesure.
- **Date** : La date et l'heure de la mesure.
- **Valeur** : La valeur mesurée (par exemple, niveau d'eau, quantité de pluie).
- **Capteur** : le capteur qui a pris la mesure

### 4. Table Technicien :

| eau_pure Technicien |  |
|---------------------|--|
| !                   | <b>id</b> : int(11)                      |
| !                   | <b>numero_de_telephone</b> : varchar(20) |
| !                   | <b>mot_de_passe</b> : varchar(255)       |
| !                   | <b>mdp_connect</b> : varchar(255)        |

Cette table enregistre les Technicien du SBEP. Elle inclut :

- **ID** : Identifiant unique du technicien.
- **mdp\_connect** : mdp pour se connecter en tant qu'admin

- **Numéro de téléphone** : Le numéro du technicien à prévenir.
- **MDP** : Le MDP haché du Technicien.

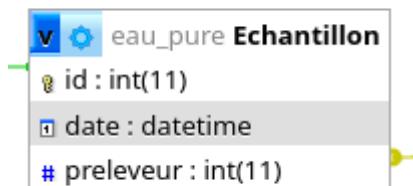
## 5. Table Préleveur :



Cette table contient les différents Préleveur :

- **ID** : Identifiant unique du préleveur.
- **station** : La station où se trouve le préleveur.
- **Capacité** : La capacité du préleveur.
- **seuil limnimètre** : seuil du limnimètre du préleveur
- **seuil pluviomètre** : seuil du pluviomètre du préleveur
- **référence** : référence du préleveur

## 6. Table Échantillon :



Cette table enregistre les Échantillon récupérer par le SBEP. Elle inclut :

- **ID** : Identifiant unique de l'échantillon.

- **préleur**: Le préleur d'où vient l'échantillon.
- **Date** : La date de l'échantillon.
- **Préleur** : le numéro du préleur dont provient l'échantillon.

## 7. Table Analyse :

| eau_pure Analyse |             |
|------------------|-------------|
| id               | int(11)     |
| prélèvement      | int(11)     |
| valeur           | float       |
| unité            | varchar(50) |
| type             | varchar(50) |

Cette table enregistre les Analyse du SBEP Elle inclut :

- **ID** : Identifiant unique de l'analyse.
- **Prélèvement** : L'ID de l'Échantillon.
- **Valeur** : La valeur de l'analyse.
- **Unité** : L'Unité de l'analyse.
- **Type** : Type de l'analyse.

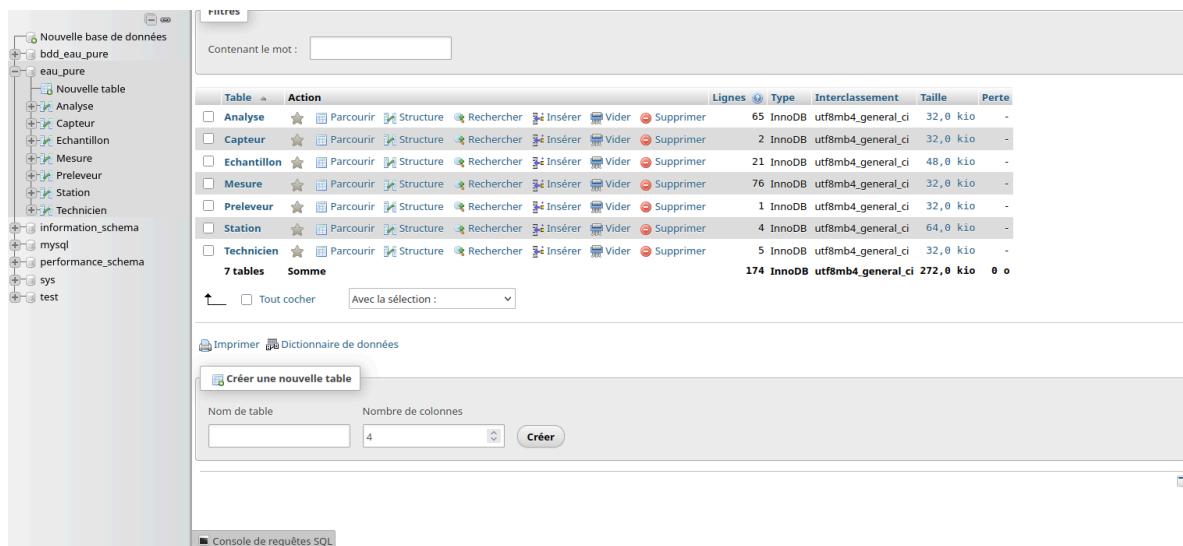
### Sécurité et intégrité des données :

J'ai veillé à ce que la base de données soit sécurisée et intégrée de manière cohérente :

- **Clés primaires et étrangères** : Chaque table utilise des clés primaires pour identifier de manière unique les enregistrements. Les clés étrangères permettent de lier les données entre les tables de manière logique et d'assurer l'intégrité référentielle (par exemple, lier les mesures à leurs stations respectives).

### Évolutivité de la base :

La structure de la base de données a été pensée pour être facilement évolutive. Par exemple, il est simple d'ajouter de nouveaux types de capteurs ou de nouvelles mesures, sans nécessiter de refonte complète du schéma.



The screenshot shows the PhpMyAdmin interface for a MySQL database named 'bdd\_eau\_pure'. The left sidebar lists various databases and tables, including 'Analyse', 'Capteur', 'Echantillon', 'Mesure', 'Prelevage', 'Station', 'Technicien', and 'mesure'. The 'mesure' table is selected, displaying its structure with columns: 'id' (int), 'station' (int), 'date' (date), and 'valeur' (float). The table contains 174 rows. Below the table, there is a 'Créer une nouvelle table' (Create new table) form with 'Nom de la table' set to 'mesure' and 'Nombre de colonnes' set to '4'. A 'Créer' (Create) button is visible.

Image n°4 : Base de Donnée sur PhpMyAdmin

## 2.2 Développement de l'Interface Python

Le script Python joue un rôle clé dans la collecte des données des capteurs et leur envoi vers le serveur. Voici les étapes détaillées du développement de l'interface Python :

### 1. Envoi dans la BDD :

- Une fonction dans ce script permet d'envoyer une donnée dans la BDD, il suffit de faire appel à cette fonction :

`envoi_mesure()`

## 2. Lire les seuils:

- une fonction dans ce script permet aussi de lire les seuils de la station : <

*lire\_seuil()*

## 3. Lire dernière mesure :

- Une fonction permet aussi de lire les dernières mesures envoyées. cette fonction peut permettre de vérifier si le seuil est dépassé ou non :

*lire\_seuil()*

### 2.3 Développement de l'Interface PHP

Les scripts PHP jouent un rôle central dans l'interaction avec la base de données et la partie WEB. L'objectif était de garantir que les données physico chimique et que les mesures soient bien lues par le SBEP afin de pouvoir mettre en place leur carte et leurs graphiques. Cela semblait plus simple car toutes les requêtes SQL seraient accessibles via l'appel de fonction.

### 2.4 Intégration de la Clé GSM

La clé **Huawei E3531** a été intégrée au système pour envoyer des alertes **SMS** en cas de dépassement des seuils critiques.

## 1. Installation et configuration de la clé GSM :

- La clé GSM Huawei E3531 est connectée à la Raspberry Pi via le port USB. Elle utilise un script python pour envoyer des SMS. La communication avec la clé est réalisée par le biais du port série de la Raspberry Pi.

## 2. Utilisation du port série avec Python :

- La communication avec la clé GSM est gérée par un script Python. En utilisant la bibliothèque **pyserial**.

## 3. Note :

La clé GSM est configurée pour envoyer le message à un numéro de téléphone spécifique lorsque les seuils sont dépassés. Le numéro de téléphone du technicien est stocké dans la base de données ou dans un fichier de configuration.

## 4. Automatisation de l'envoi des alertes :

- L'ensemble du processus est automatisé. Dès qu'un seuil critique est atteint, le script Python gère l'envoi de l'alerte via SMS.

### 2.1 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°5 : 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.

### 3. Tests Fonctionnels

#### 3.1 Tests Unitaires

Les tests unitaires sont réalisés pour s'assurer que chaque composant du système fonctionne de manière autonome et dans les conditions prévues. Dans le cadre de ce projet, les tests unitaires ont principalement porté sur le code Python.

##### 1. Tests du script Python :

- Les tests unitaires ont permis de vérifier que les fonctions de collecte des données des capteurs, de gestion des seuils et d'envoi des alertes SMS fonctionnent correctement.
- **pytest** a été utilisé comme framework de test pour Python, permettant d'automatiser l'exécution des tests et de garantir que chaque fonction retourne les résultats attendus.

**Exemple de test unitaire pour vérifier l'envoi d'une alerte SMS voir annexe :**

##### 2. Tests de l'interface BDD→ Station :

- À l'aide de pytest, j'ai pu faire plusieurs tests sur les fonctions de mon interfaces : différents cas d'erreur, cas nominaux ou autres.

Exemple de test python pour valider l'insertion des données : voir annexe

## 4. Environnement Technique / Contraintes

### 4.1 Choix de l'Environnement Dockerisé

L'une des principales décisions techniques a été d'utiliser **Docker** pour le déploiement du système. Docker permet de créer des conteneurs légers et isolés pour exécuter des applications et services. Cette solution garantit la portabilité et la reproductibilité des environnements de développement, en assurant une exécution identique sur toutes les machines, ce qui est particulièrement utile dans un projet qui implique plusieurs technologies et composants.

#### Pourquoi Docker ?

- **Portabilité** : En utilisant Docker, nous garantissons que le système fonctionnera de manière identique sur n'importe quel ordinateur ou serveur, sans être dépendant de l'environnement matériel ou logiciel spécifique.
- **Facilité de gestion** : Docker simplifie l'installation et la mise à jour des services, rendant le projet plus modulaire et plus facile à maintenir.

#### Services Docker utilisés :

1. **MySQL** : Le serveur de base de données qui gère toutes les mesures collectées par les capteurs.
2. **PhpMyAdmin** : Outil graphique pour gérer la base de données MySQL de manière conviviale, permettant de visualiser et manipuler les données sans avoir à utiliser la ligne de commande.
3. **Apache + PHP** : Le serveur web qui héberge l'interface PHP et l'API pour la gestion des données des capteurs.
4. **Portainer** : Interface graphique permettant de gérer les conteneurs Docker. Il fournit une vue d'ensemble et permet de gérer les conteneurs et les volumes de manière intuitive.

#### Commande pour démarrer l'environnement Dockerisé :

```
docker-compose up -d
```

Cette commande lance tous les services définis dans le fichier `docker-compose.yml` en arrière-plan, et ce, de manière isolée et cohérente. Elle assure également que tous les services interagiront correctement entre eux, même si certains services dépendent des autres.

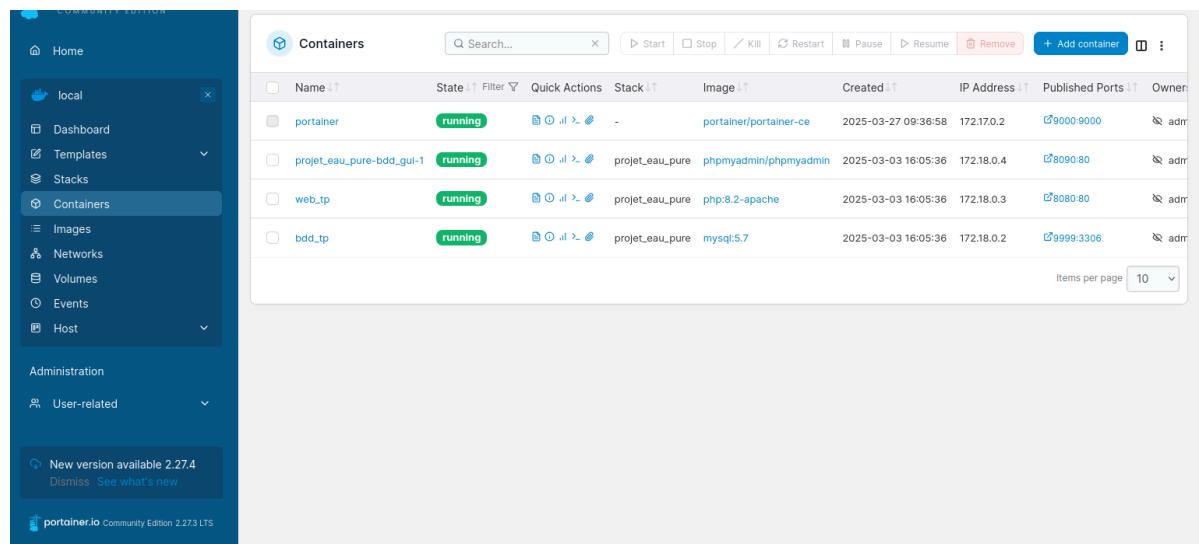


Image n°6 : Portainer

## 5. Architecture Matérielle / Choix Techniques

### 5.1 Matériel Utilisé

Pour la mise en place du projet, plusieurs composants matériels ont été utilisés. Chaque composant joue un rôle spécifique dans la collecte des données, le traitement de celles-ci, et l'envoi des alertes en temps réel.

Voici les principaux équipements matériels utilisés dans ma partie :

## 1. Clé GSM Huawei E3531 :



- **Rôle** : La clé GSM est utilisée pour envoyer des alertes SMS aux techniciens lorsque les seuils critiques sont atteints (par exemple, un niveau d'eau trop élevé).
- **Interface** : Connexion via le port série de la Raspberry Pi. Elle est utilisée avec un script python pour envoyer des SMS.

### Commande d'installation :

Pour installer la clé GSM et permettre à la Raspberry Pi de l'utiliser, il est nécessaire d'installer un paquet permettant de gérer les communications série.

```
sudo apt install minicom  
sudo apt install usb-modeswitch
```

## 2. Serveur Local :

- **Rôle** : Le serveur héberge l'ensemble des services nécessaires au projet, y compris **MySQL**, **PhpMyAdmin**, et l'interface web PHP.
- **Configuration** : Un serveur local équipé de Docker et des services nécessaires pour faire fonctionner le projet en production.

### 5.2 Architecture du Système

L'architecture matérielle du système a été conçue de manière à garantir l'autonomie du système tout en facilitant l'interopérabilité entre les composants. Voici la configuration globale :

### 1. Serveur Local :

- Héberge la **base de données MySQL** et les services nécessaires au fonctionnement de l'interface web.
- Exécute un serveur **Apache** qui gère l'affichage des données sur l'interface web.

### 2. Base de Données (MySQL) :

- Elle stocke les données des capteurs et les historiques des mesures. Chaque donnée est associée à une station de mesure spécifique, et des relations entre les différentes tables permettent de structurer et d'organiser les informations.

### 3. Clé GSM Huawei E3531 :

- Lorsqu'un seuil critique est détecté par la Raspberry Pi, cette clé permet d'envoyer un SMS au technicien pour l'alerter de la situation. La clé est directement connectée à la Raspberry Pi via le port série.

## 6. Conclusion et Perspectives

### 6.1 Bilan du Projet

Le projet a permis de développer un système de collecte de données environnementales fiable et efficace à partir de capteurs, avec une gestion des données centralisée sur un serveur. Ce système offre une solution robuste pour surveiller en temps réel les conditions climatiques, et ce, en utilisant une architecture flexible qui s'adapte à différents types de déploiement. L'intégration de la Raspberry Pi, de capteurs, et de la communication via clé GSM a permis de garantir une continuité de service, même en cas de perte de connexion Internet.

Les objectifs fixés en début de projet ont été largement atteints, à savoir :

- La collecte continue et précise de données.
- La gestion centralisée de ces données dans une base de données.
- L'interface utilisateur permettant de visualiser ces données en temps réel.
- La résilience du système grâce à la redondance de la communication via GSM en cas de perte de réseau.

Les tests ont confirmé que le système fonctionne comme prévu dans des conditions variées et qu'il est suffisamment flexible pour être adapté à différents types de capteurs et de configurations réseau.

### **6.2 Limitations et Améliorations**

Bien que le système ait montré de bons résultats, plusieurs améliorations peuvent être envisagées pour augmenter ses performances et sa portée.

#### **1. Scalabilité :**

- Le système a bien fonctionné avec un nombre limité de capteurs et un seul site de collecte. Toutefois, pour un déploiement à grande échelle, il serait nécessaire d'optimiser la gestion du serveur pour supporter plusieurs sites de collecte simultanés et garantir la rapidité d'accès aux données.

#### **2. Optimisation Énergétique :**

- Si le système devait être déployé dans des environnements sans accès à l'électricité (zones isolées, extérieures), il serait nécessaire d'explorer des solutions énergétiques durables telles que l'intégration de panneaux solaires ou des dispositifs à faible consommation pour les capteurs et la Raspberry Pi.

#### **3. Sécurité des Données :**

- Bien que le système fonctionne de manière fiable, des mesures de sécurité renforcées pourraient être mises en place pour protéger les données envoyées et stockées, en particulier lors de l'utilisation de la communication GSM. Des protocoles de chiffrement et des mécanismes d'authentification pourraient être ajoutés pour garantir la confidentialité des informations collectées.

#### **4. Interface Utilisateur :**

- L'interface utilisateur pourrait être améliorée pour inclure davantage de fonctionnalités telles que la génération de rapports automatiques, des alertes en temps réel sur des seuils spécifiques.

### **6.3 Perspectives d'Évolution**

Le système a ouvert la voie à plusieurs pistes d'évolution pour de futures versions ou des projets similaires. Voici quelques perspectives qui pourraient enrichir le projet à l'avenir :

**1. Intégration d'autres types de capteurs :**

- En plus des capteurs de température et d'humidité, il serait possible d'intégrer d'autres capteurs tels que des capteurs de gaz, de pression, ou encore des capteurs de luminosité pour enrichir la collecte de données et étendre l'utilité du système.

**2. Automatisation et Contrôle à Distance :**

- Le système pourrait évoluer pour inclure des mécanismes d'automatisation, par exemple, en contrôlant des dispositifs à distance en fonction des données collectées.

**3. Amélioration de l'Accessibilité :**

- Une application mobile pourrait être développée pour rendre le suivi des données plus accessible aux utilisateurs. Cette application pourrait inclure des notifications push pour alerter les utilisateurs en temps réel des variations importantes dans les données collectées.

**4. Big Data et Analyse Avancée :**

- Avec un grand volume de données collectées sur de longues périodes, des techniques de Big Data pourraient être utilisées pour analyser les tendances, prédire des événements futurs, ou même optimiser les processus en fonction des conditions climatiques.

#### **6.4 Conclusion Finale**

Ce projet a démontré l'efficacité d'un système de collecte de données environnementales basé sur des technologies accessibles et modulables, telles que la Raspberry Pi, les capteurs et la communication GSM. Il offre une base solide pour des applications variées allant de la surveillance climatique à la gestion de l'environnement.

Bien que le système soit déjà performant, les pistes d'amélioration et d'extension offrent des opportunités pour le rendre encore plus robuste, flexible, et adapté aux besoins d'une large gamme d'utilisateurs et de scénarios.

L'évolution vers un système intelligent, capable de s'adapter en temps réel aux conditions changeantes, représente un défi stimulant mais réalisable, et ouvre la voie à des applications novatrices dans de nombreux domaines, de la gestion de l'agriculture à la gestion des ressources naturelles.

---

## Élève n°3 - Nolan CONAN

---



**Table des matières**

|  |           |
|--|-----------|
| <b>Étudiant n°3 - Nolan CONAN.....</b>   | <b>63</b> |
| <b>1. Analyse / Aspect fonctionnel.....</b>  | <b>63</b> |
| 1.1. Présentation de l'architecture logicielle et matérielle.....                          | 63        |
| 1.2. Analyse du diagramme de cas d'utilisation du sous-système SBEP.....                   | 64        |
| 1.3. Analyse du diagramme des exigences.....   | 65        |
| 1.4. Moyens techniques et outils mis à disposition.....                                    | 65        |
| 1.5. Périmètre des responsabilités.....  | 66        |
| 1.6. Liste des tâches.....   | 66        |
| <b>2. Travail réalisé.....</b>   | <b>67</b> |
| 2.1. Planification et organisation des tâches.....   | 67        |
| 2.2. Mise en place de l'authentification sécurisée.....                                    | 67        |
| 2.3. Ajout de la gestion des utilisateurs et renforcement de la sécurité.....              | 68        |
| 2.4. Développement du formulaire de saisie des données physico-chimiques.....              | 69        |
| <b>3. Création de l'application React.....</b>   | <b>69</b> |
| 3.1. Développement de l'application de visualisation des données physico-chimiques.....    | 69        |
| 3.2. Installation de React.....  | 70        |
| 3.3. Structure du projet.....  | 70        |
| 3.4. Affichage de graphiques avec React et Node.js.....                                    | 71        |
| 3.5. Mise en place du serveur backend (Node.js + Express).....                             | 71        |
| 3.6. Lancement du serveur backend.....   | 72        |
| 3.7. Installation des bibliothèques de graphiques React.....                               | 72        |
| 3.8. Intégration frontend/backend.....   | 73        |
| 3.9. Développement de l'interface de visualisation des données.....                        | 73        |
| 3.10. Installation de l'environnement de travail.....                                      | 73        |
| 3.11. Création du backend et liaison à la base de données.....                             | 73        |
| 3.12. Développement frontend : affichage et interactions.....                              | 74        |
| <b>4. Présentation du diagramme d'activité de l'application Web.....</b>                   | <b>74</b> |
| <b>5. Environnement technique / Contraintes.....</b>                                       | <b>75</b> |
| 5.1. Choix de l'environnement de développement.....  | 75        |
| 5.2. Contraintes imposées par le client.....   | 75        |
| <b>6. Aspects liés à la sécurité / robustesse des éléments mis en œuvre.....</b>           | <b>76</b> |
| 6.1. Inventaire des versions et outils utilisés.....                                       | 76        |
| 6.2. Vérification des mises à jour logicielles et pilotes.....                             | 77        |
| 6.3. Configuration matérielle et logicielle contre les attaques.....                       | 77        |
| 6.4. Gestion des enjeux de cybersécurité et des mises à jour.....                          | 77        |
| <b>7. Tests unitaires.....</b>   | <b>78</b> |
| <b>8. Planification.....</b>   | <b>80</b> |
| <b>9. Analyse de l'avancement et perspectives d'amélioration.....</b>                      | <b>82</b> |
| 9.1. État d'avancement du projet, difficultés rencontrées et justification des écarts..... | 82        |
| 9.2. Pistes d'amélioration.....  | 83        |
| <b>10. Bilan personnel.....</b>  | <b>83</b> |
| <b>Annexes.....</b>  | <b>85</b> |
| <b>3. Etudiant n°3.....</b>  | <b>96</b> |

## Étudiant 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. Analyse du diagramme de cas d'utilisation du sous-système SBEP

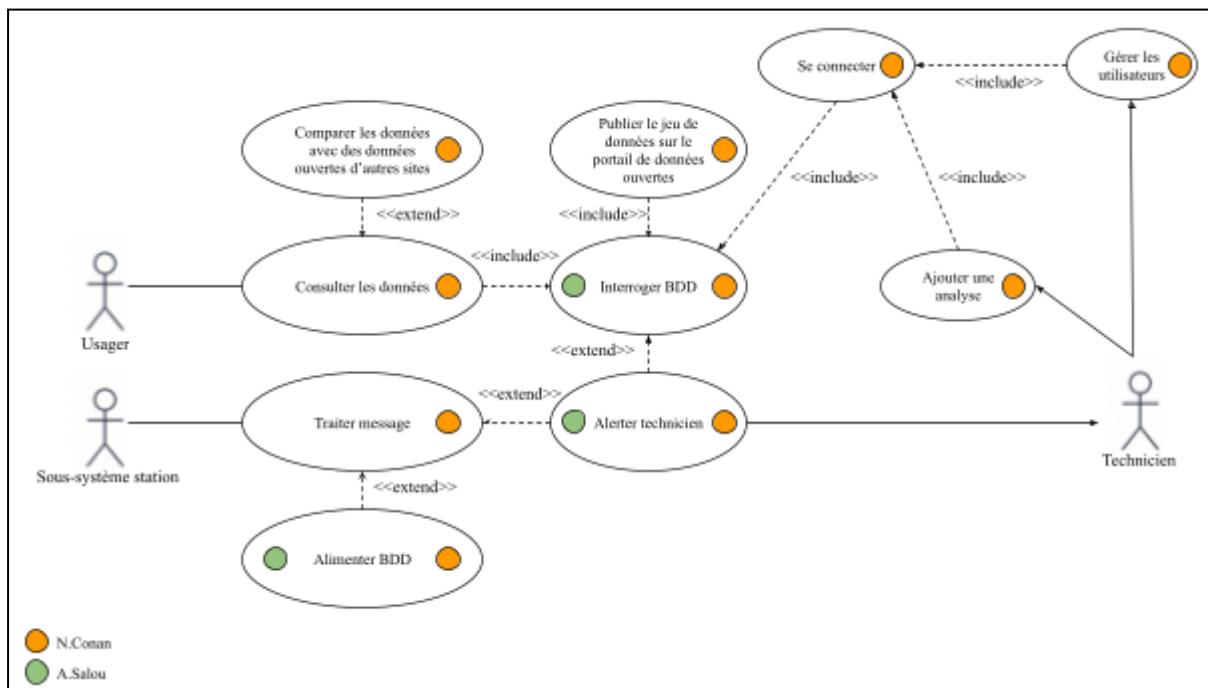


Figure n°1 : Diagramme de cas d'utilisation du sous-système SBEP

Le diagramme de cas d'utilisation du sous-système SBEP illustre les différentes interactions possibles entre les acteurs (technicien, usager, sous-système station) et le système Web dont j'ai assuré le développement.

Ce diagramme a pour objectif de représenter les **fonctionnalités principales offertes par l'interface Web**, en lien avec les données transmises par la station hydrologique.

Lorsqu'un message est reçu par le système, plusieurs traitements sont possibles :

- Les **données mesurées** peuvent être enregistrées dans la base de données via le cas « **Alimenter BDD** ».
- Une **alerte** peut être envoyée au technicien (« **Alerter technicien** »), ce qui entraîne une **recherche de son numéro** dans la base de données (« **Interroger BDD** »).

Une fois les données enregistrées, le technicien peut **ajouter les résultats d'analyse chimique** à travers le cas d'utilisation « **Ajouter une analyse** », disponible uniquement après authentification (« **Se connecter** »). Ce processus assure que seules les personnes autorisées peuvent modifier la base de données.

L'authentification est gérée par le cas « **Se connecter** », qui inclut aussi la possibilité pour un administrateur de **gérer les utilisateurs** (ajouter, modifier ou supprimer un compte), comme représenté dans « **Gérer les utilisateurs** ».

Les données physico-chimiques peuvent ensuite être **publiées sur le portail de données ouvertes** via l'action « **Publier le jeu de données** », et sont également **consultables sur le site Web** grâce au cas « **Consulter les données** ». Les usagers peuvent en plus les **comparer**

avec celles d'autres stations hydrologiques, à travers le cas « **Comparer les données avec les données ouvertes d'autres sites** ».

Ce diagramme montre bien la **complémentarité entre les rôles et la circulation des données** dans le système, depuis leur enregistrement jusqu'à leur publication et leur visualisation.

### 1.3. Analyse du diagramme des exigences

Le diagramme des exigences, présenté en figure n°1 de la partie commune, décrit les principales fonctionnalités attendues du système. Certaines de ces exigences sont directement liées à mon travail sur la partie Web du projet.

L'exigence « **Publication des données** » (id 1.3) stipule que les données doivent être publiées sur le portail du SIE et accessibles au public. J'ai donc développé l'interface Web permettant l'affichage des données de manière claire et structurée, ainsi que leur préparation en vue de leur publication.

L'exigence « **Compatibilité avec SIE** » (id 1.4) demande que le système soit interconnecté avec le portail national. Pour y répondre, j'ai mis en place un traitement automatique des données afin de respecter les formats requis par les API.

```
[{"valeur":8,"type":"ph","unite":"","date":"2025-05-12T10:00:00.000Z","latitude":47.7101,"longitude":-3.3668,"riviere":"Blavet"},  
[...]
```

Enfin, l'exigence « **Archivage et réutilisation** » (id 1.8) concerne la conservation à long terme des données. La gestion du stockage des données a été confiée à A. Salou. Nous avons collaboré ensemble afin que ma partie s'intègre correctement à la structure de données qu'il avait mise en place. Je me suis appuyé sur son travail pour adapter mes développements en fonction du schéma relationnel de la base de données retenu (voir schéma relationnel de la base de données).

[Lien vers le diagramme des exigences](#)

### 1.4. 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.

### 1.5. Périmètre des responsabilités

Dans le cadre de ce projet, je suis responsable de la conception et du développement de la partie web du projet. J'utilise React pour créer l'interface du site local, qui permet de présenter les données de mesures ainsi que les alertes de manière géolocalisée via OpenStreetMap/Leaflet. Je développe également un formulaire sécurisé permettant l'entrée des résultats d'analyse physico-chimique dans la base de données. De plus, j'exploite des API ouvertes afin d'automatiser le formatage des données avant leur publication. En collaboration avec l'étudiant 2, j'interviens également dans la conteneurisation du serveur LAMP et la gestion des interactions avec la base de données.

[Lien vers le diagramme de déploiement](#)

### 1.6. Liste des tâches

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

## 2. Travail réalisé

### 2.1. Planification et organisation des tâches

Pour ce projet, j'ai commencé par insérer les différentes tâches à réaliser, issues du cahier des charges, dans le tableau Scrum que mes camarades et moi avons créé sur GitHub. Le cahier des charges contient des tâches de complexité variable. Il était donc essentiel de les découper autant que possible afin de progresser efficacement sans négliger certaines phases importantes qui, si elles étaient omises, pourraient freiner l'avancement du projet.

Une fois les tâches planifiées, nous avons évalué leur niveau de difficulté à l'aide de la méthode du Planning Poker. Cette approche nous a permis d'estimer l'effort requis pour chaque tâche en leur attribuant une valeur, puis d'ajuster ces estimations collectivement jusqu'à parvenir à un accord.

[Figure n°2 : Réalisation d'un tableau Scrum créé sur GitHub à partir du cahier des charges](#)

[Figure n°3 : 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, il est 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. Ces problèmes étaient causés par le pare-feu Fortinet, utilisé par l'établissement, qui bloquait les échanges entre mon application web et le serveur MySQL distant. Ce filtrage réseau m'empêchait de valider les identifiants utilisateurs et donc d'accéder aux pages suivantes de l'application.

Pour contourner cette contrainte, mon professeur référent m'a mis à disposition un serveur LAMP, ce qui m'a permis de travailler en local, sans les restrictions imposées par le réseau de l'établissement. 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 4 : 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°5 : Réalisation d'un programme permettant de hacher un mot de passe

[Figure n°6 : 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, mon professeur référent m'a demandé d'ajouter une nouvelle fonctionnalité : un bouton permettant à l'administrateur d'ajouter, de modifier ou de 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°7 : Pop-up de connexion administrateur](#)

[Figure n°8 : Pop-up de gestion des utilisateurs](#)

Figure n°9 : Réalisation des fenêtres pop-up ajouter, modifier ou supprimer un utilisateur

#### 2.4. Développement du formulaire de 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°10 : Réalisation du formulaire pour la saisie des données physico-chimiques

### 3. Crédit 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. Affichage de graphiques avec React et 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 “donnees\_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.

#### 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 de graphiques 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. Intégration frontend/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 et liaison à 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 frontend : affichage et 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.).

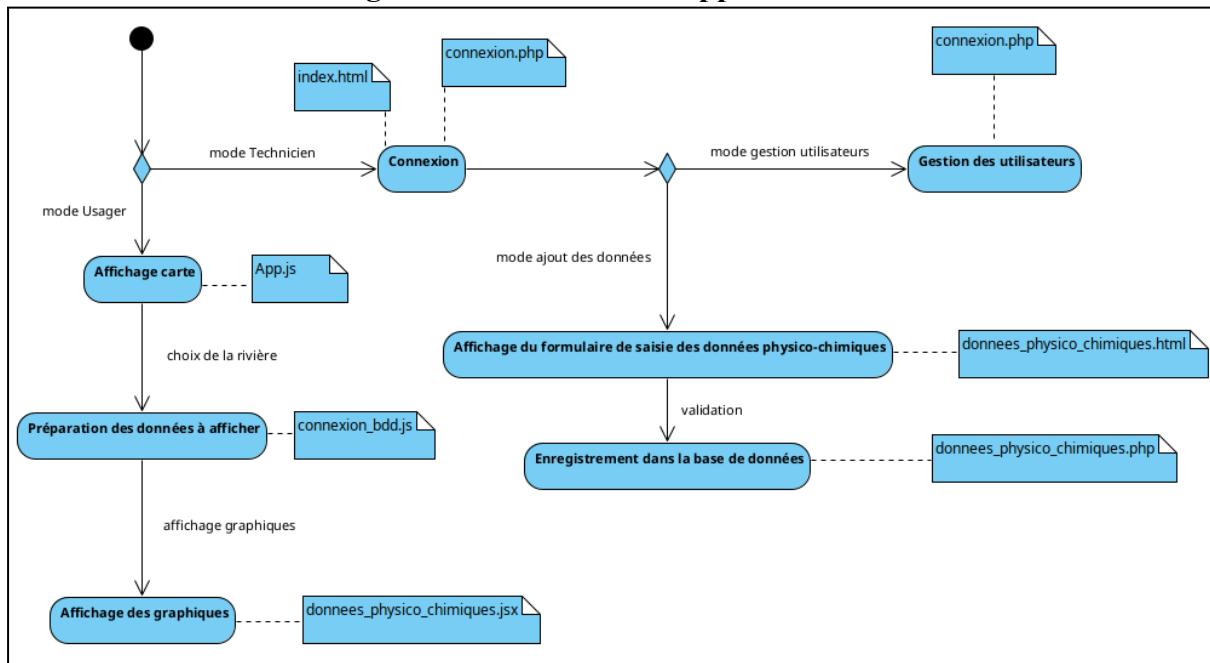
En plus de la carte, j'ai ajouté des graphiques qui s'affichent lorsqu'on choisit une rivière dans une liste déroulante. Ces graphiques permettent de voir l'évolution des différentes données dans le temps. J'ai utilisé une autre bibliothèque appelée Chart.js, qui permet de créer des graphiques facilement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

[Figure n°11 : Présentation de mesures de manière géolocalisée](#)

[Figure n°12 : Présentation des mesures de manière graphique](#)

## 4. Présentation du diagramme d'activité de l'application Web



[Figure n°13 : Diagramme d'activité illustrant la navigation au sein de l'application Web](#)

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

### 6.1. Inventaire des versions et outils utilisés

| Nom de l'outil / bibliothèque     | Fonctionnalité principale   | Version installée                              | Procédure d'installation   | Raison du choix   | Commande de vérification   |
|-----------------------------------|---|--|--|---|--|
| <b>GitHub</b>                     | Plateforme de <b>gestion de version, de collaboration et de suivi de projet.</b>            | Service en ligne (toujours à jour)             | Accès via navigateur web sur <a href="https://github.com">github.com</a> | Outil populaire, riche en fonctionnalités (Issues, Kanban, Pull Requests). Collaboration facilitée. | git --version  |
| <b>PHP</b>                        | Langage de <b>script serveur</b> pour créer des pages web dynamiques.                       | PHP 8.2  | Intégré dans le conteneur via Dockerfile                                 | Langage très répandu, compatible avec MySQL, idéal pour les formulaires sécurisés.                  | php -v   |
| <b>MySQL</b>                      | Système de <b>gestion de base de données relationnelle.</b>                                 | MySQL 5.7                                      | Via Docker avec configuration dans docker-compose.yml                    | Fiable, performant, bien intégré avec PHP.  | mysql -V   |
| <b>Docker</b>                     | Plateforme de <b>conteneurisation</b> pour déployer des environnements isolés et cohérents. | Docker 20.10                                   | Installation selon les instructions officielles sur le site Docker.      | Isolation, portabilité, cohérence des environnements.   | docker --version   |
| <b>Leaflet.js</b>                 | Bibliothèque <b>JavaScript légère</b> pour créer des cartes interactives.                   | v1.7.1   | npm install leaflet  | Simple, performant, idéal pour une intégration rapide de cartes interactives.                       | npm list leaflet   |
| <b>PHPMyAdmin</b>                 | Interface web pour <b>gérer visuellement la base de données MySQL.</b>                      | Version incluse dans l'image Docker officielle | Via Docker avec docker-compose.yml                                       | Simplifie les requêtes SQL et la gestion des tables sans ligne de commande.                         | Accès via navigateur ( <a href="http://localhost:8080">http://localhost:8080</a> ) |
| <b>React</b>                      | <b>Bibliothèque JavaScript</b> pour construire des interfaces web dynamiques en composants. | Dernière version via create-react-app          | npx create-react-app nom_du_projet                                       | Permet une interface moderne, maintenable et modulaire.   | npm list react   |
| <b>Chart.js + react-chartjs-2</b> | Création de <b>graphiques interactifs</b> dans l'application React.                         | Dernières versions via npm                     | npm install chart.js react-chartjs-2                                     | Simple à utiliser, bien intégré avec React, visuellement clair.                                     | npm list chart.js  |
| <b>Axios</b>                      | Envoi de <b>requêtes HTTP</b> entre frontend React et backend Node.js.                      | Dernière version via npm                       | npm install axios  | Pratique pour les appels d'API REST dans React.   | npm list axios   |

## 6.2. Vérification des mises à jour logicielles 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.

## 6.3. Configuration matérielle et logicielle contre les 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.

## 6.4. Gestion des enjeux de cybersécurité 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

## 7. Tests unitaires

|                      |  |
|----------------------|--|
| <b>Élément testé</b> | Formulaire web et interface base de données  |
| <b>Objectif</b>      | Ajouter un technicien SBEP depuis l'interface web  |
| <b>Pré-requis</b>    | Le site Web est accessible à l'URL <a href="http://localhost:8080/">http://localhost:8080/</a> . Le test est réalisé avec Mozilla Firefox (version 137.0 (64 bits)). |

| Action   | Résultat attendu   | Observations |
|--|--|--------------|
| Cliquer sur le bouton « Gestion des utilisateurs »                           | La pop-up « Mot de passe administrateur » est affichée.  | .....        |
| Cliquer sur le bouton de fermeture de la pop-up                              | La pop-up « Mot de passe administrateur » disparaît.   | .....        |
| Cliquer sur le bouton « Gestion des utilisateurs »                           | La pop-up « Mot de passe administrateur » est affichée.  | .....        |
| Cliquer sur le bouton « Annuler » de la pop-up                               | La pop-up « Mot de passe administrateur » disparaît.   | .....        |
| Cliquer sur le bouton « Gestion des utilisateurs »                           | La pop-up « Mot de passe administrateur » est affichée.  | .....        |
| Saisir le mot de passe administrateur puis cliquer sur le bouton « Valider » | La pop-up « Mot de passe administrateur » disparaît.<br><br>La pop-up « Gestion des utilisateurs » est affichée.                               | .....        |
| Cliquer sur le bouton « Ajouter un utilisateur »                             | La pop-up « Ajouter un utilisateur » est affichée.<br><br>La pop-up « Gestion des utilisateurs » disparaît.                                    | .....        |
| Rentrer les informations nécessaires puis cliquer sur le bouton «Ajouter»    | La pop-up avec un message indiquant « L'utilisateur a bien été ajouté. » est affichée.<br><br>La pop-up « Gestion des utilisateurs » apparaît. | .....        |
| <b>Conclusion</b>  | .....  |              |

Figure n°14 : Procédure de test pour l'ajout d'un utilisateur

Pour garantir le bon fonctionnement des différentes fonctionnalités de l'application web, des tests unitaires ont été réalisés à l'aide de Selenium IDE, un outil permettant d'automatiser les interactions avec un navigateur. Grâce à Selenium IDE, il a été possible de simuler des actions telles que la connexion d'un utilisateur, l'ajout, la modification ou la suppression d'un compte, et de vérifier que le comportement de l'interface correspond aux attentes définies dans le cahier des charges. Ces tests ont été conçus pour couvrir à la fois les cas nominaux (utilisation correcte) et les scénarios alternatifs (erreurs de saisie, identifiants incorrects, etc.). Tous les résultats des tests ont été tracés manuellement dans un cahier de test. Cela a permis de valider que chaque fonctionnalité développée répondait bien aux spécifications demandées, tout en facilitant la détection et la correction d'éventuels bugs.

## 8. Planification

| Semaine | Dates          | Nb heures | Nom de la tâche   | Etat d'avancement |
|---------|----------------|-----------|---|-------------------|
| 1       | 24/02 au 28/02 | 13h       | <ul style="list-style-type: none"> <li>• Réalisation d'un tableau SCRUM (3h)</li> <li>• Réalisation d'un programme permettant d'hasher et saler un mot de passe (7h)</li> <li>• Réalisation d'un formulaire de connexion sécurisé (3h)</li> </ul> | ♥<br>♥<br>●       |
| 2       | 03/03 au 07/03 | 17h       | <ul style="list-style-type: none"> <li>• Réalisation d'un formulaire de connexion sécurisé (10h)</li> <li>• Réalisation d'un formulaire pour les données physico-chimiques (7h)</li> </ul>  | ♥<br>●            |
| 3       | 10/03 au 14/03 | 17h       | <ul style="list-style-type: none"> <li>• Création des identifiants de connexion pour les techniciens (17h)</li> </ul>   | ■                 |
| 4       | 17/03 au 21/03 | 17h       | <ul style="list-style-type: none"> <li>• Création des identifiants de connexion pour les techniciens (17h)</li> </ul>   | ♥                 |
| 5       | 24/03 au 28/03 | 16h       | <ul style="list-style-type: none"> <li>• Documentation OpenStreetMap/Leaflet (8h)</li> <li>• Professionnalisation du formulaire de connexion (6h)</li> <li>• Réalisation de la carte interactive + graphiques (2h)</li> </ul>                     | ♥<br>●<br>■       |
| 6       | 31/03 au 04/04 | 17h       | <ul style="list-style-type: none"> <li>• Réalisation de la carte interactive + graphiques (17h)</li> </ul>  | ●                 |
| 7       | 22/04 au 25/04 | 9h        | <ul style="list-style-type: none"> <li>• Réalisation de la carte interactive + graphiques (9h)</li> </ul>   | ♥                 |
| 8       | 28/04 au 02/05 | 10h       | <ul style="list-style-type: none"> <li>• Rédaction du rapport (5h)</li> <li>• Réajustement de certains codes dû à la modification de liens dans la BDD (5h)</li> </ul>  | ●<br>♥            |
| 9       | 05/05 au 09/05 | 10h       | <ul style="list-style-type: none"> <li>• Réalisation de tests unitaires, d'intégration avec Selenium IDE (8h)</li> <li>• Rédaction du rapport (2h)</li> </ul>   | ♥<br>●            |
| 10      | 12/05 au 16/05 | 21h       | <ul style="list-style-type: none"> <li>• Finalisation du rapport (7h)</li> </ul>  | ♥                 |

■ : Tâche non réalisée avec un niveau d'avancement trop faible | ● : Tâche en cours de réalisation | ♥ : Tâche validée, conforme au cahier des charges

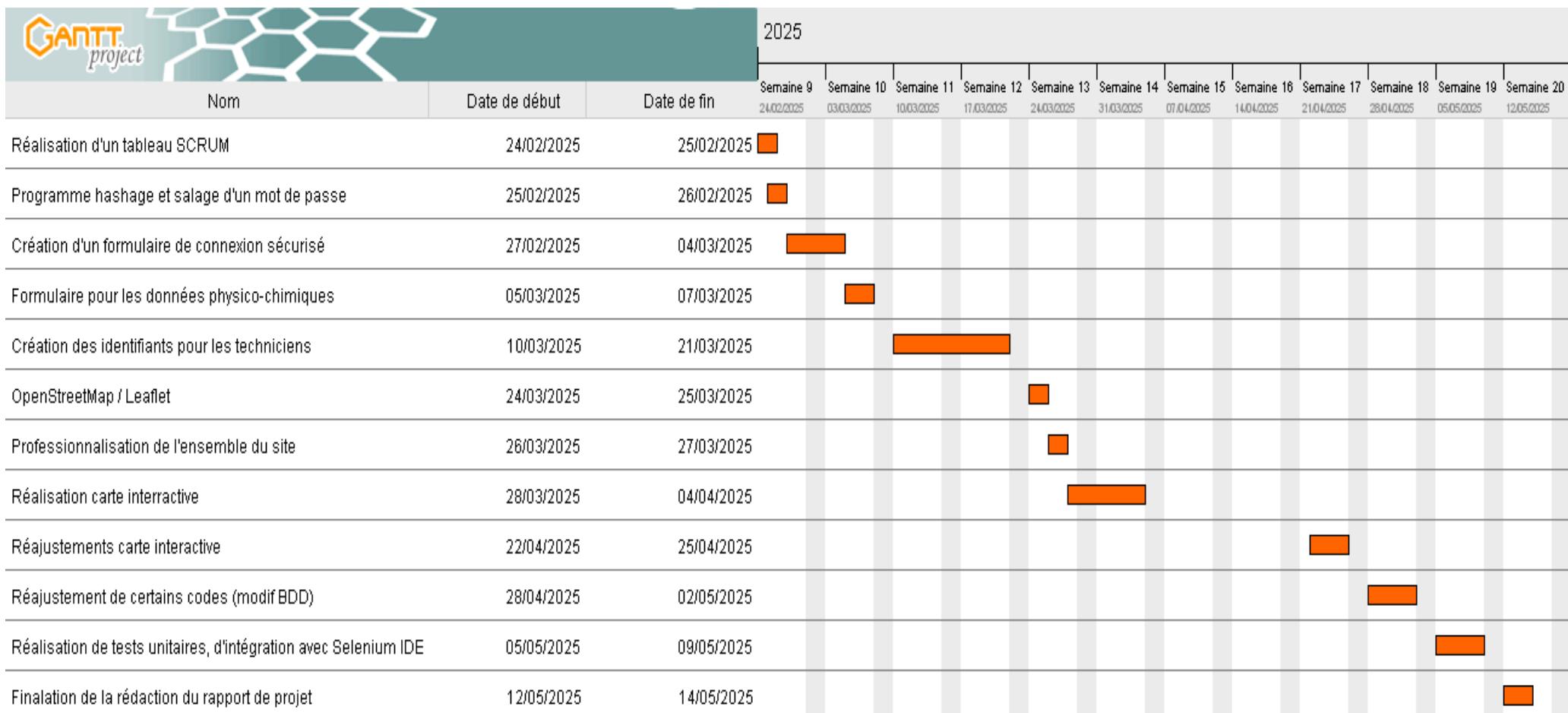


Figure n°15 : Diagramme de Gantt

## 9. Analyse de l'avancement et perspectives d'amélioration

### 9.1. État d'avancement du projet, difficultés rencontrées et justification des écarts

Dans le cadre de ma mission, j'ai été chargé de développer une interface Web sécurisée permettant aux techniciens du SBEP de saisir les résultats des analyses physico-chimiques dans une base de données. La première étape consistait à mettre en place un formulaire sécurisé. Si la création d'un formulaire de connexion basique était prévue et maîtrisée, l'ajout de fonctionnalités avancées comme la gestion des utilisateurs via des fenêtres pop-up (ajout, modification, suppression) s'est révélé plus complexe. Ne disposant pas d'expérience préalable sur ce type de fonctionnalités, j'ai dû approfondir mes connaissances de manière autonome, ce qui a engendré un dépassement du temps prévu initialement.

La seconde grande phase concernait la visualisation des données physico-chimiques sur une carte interactive. Pour cela, j'ai utilisé les bibliothèques Leaflet et OpenStreetMap. Bien que la logique de représentation spatiale ait été relativement abordable, l'implémentation en local via le framework React a nécessité un effort conséquent. Ayant peu pratiqué JavaScript durant ma formation, l'intégration de ce langage avec React m'a demandé beaucoup de temps de documentation et de tests, expliquant également un léger écart avec le planning initial.

Une autre partie, moins prioritaire mais néanmoins importante, concernait l'automatisation du formatage et la validation des données en vue de leur publication sur des portails de données ouvertes. J'ai malgré tout pris le temps de m'y intéresser en explorant les API de data.gouv.fr et api.gouv.fr, afin de comprendre les standards de publication utilisés et d'envisager une intégration future.

À ce jour, la majorité des fonctionnalités prévues sont opérationnelles :

- Saisie sécurisée des résultats d'analyse par les techniciens
- Visualisation géolocalisée des données physico-chimiques sur une carte interactive
- Affichage graphique dynamique des résultats
- Export des données au format JSON, prêt pour la publication en open data

Il me reste à finaliser la conteneurisation du serveur LAMP avec Docker, en collaboration avec un autre membre du groupe, pour assurer un déploiement plus structuré et modulaire.

## 9.2. Pistes d'amélioration

Pour aller plus loin dans la qualité du projet, plusieurs pistes d'amélioration peuvent être envisagées. Tout d'abord, il serait pertinent de moderniser et professionnaliser davantage l'interface web, en travaillant sur l'ergonomie, le design et l'expérience utilisateur afin de rendre la plateforme plus intuitive, agréable à utiliser et adaptée à un usage professionnel.

Par ailleurs, l'ajout de données en temps réel issues des capteurs renforcerait la valeur fonctionnelle de l'application. Plus précisément, il serait intéressant d'afficher la hauteur du cours d'eau mesuré par le capteur limnimètre, ainsi que le niveau de précipitations relevé par le pluviomètre. Ces données viendraient compléter les mesures physico-chimiques déjà intégrées, en apportant une vision plus globale de l'état des cours d'eau surveillés.

Enfin, un point perfectible concerne la navigation entre les pages : pour le moment, l'utilisateur doit utiliser la flèche "Retour" du navigateur pour revenir à l'écran précédent. Il serait donc plus pertinent d'intégrer une fonctionnalité de navigation interne (par exemple, un bouton "Retour" ou un menu de navigation), afin d'améliorer la fluidité et l'autonomie de l'expérience utilisateur.

## 10. Bilan personnel

Ce projet m'a offert une véritable opportunité de développement personnel et professionnel, tant sur le plan technique qu'organisationnel. J'ai renforcé mes compétences en conception et développement web, notamment en apprenant à utiliser de nouveaux outils comme React, Leaflet, Docker ou encore GitHub, que je ne maîtrisais pas auparavant. L'apprentissage en autonomie de ces technologies m'a permis de développer ma capacité à rechercher des solutions, à analyser de la documentation technique et à intégrer rapidement de nouveaux environnements de travail.

Sur le plan méthodologique, j'ai compris l'importance de structurer clairement mes idées en amont pour optimiser le temps de développement et mieux anticiper les difficultés. Cela m'a conduit à améliorer ma rigueur dans la planification des tâches, l'organisation du code, et le versionnement régulier sur Git.

Ce projet m'a également permis de progresser dans le travail en équipe. J'ai appris à écouter et intégrer les retours de mes coéquipiers, à adapter mes choix techniques pour assurer la compatibilité entre nos différentes contributions, et à maintenir une communication fluide au sein du groupe. Ces échanges ont renforcé ma capacité à collaborer efficacement dans un contexte de projet technique.

Avec le recul, je pense qu'il aurait été judicieux de réaliser les diagrammes de conception (cas d'utilisation, architecture, etc.) dès le début du projet. Cela nous aurait permis de mieux structurer notre travail, de clarifier la répartition des responsabilités et de limiter certaines pertes de temps en phase de développement.

Enfin, cette expérience m'a appris à documenter mon travail de manière claire et exploitable, aussi bien pour mes coéquipiers que pour toute personne amenée à reprendre ou faire évoluer le projet par la suite. Elle m'a fait progresser dans des compétences clés telles que la résolution de problèmes, l'adaptabilité, la communication professionnelle et l'autonomie dans l'apprentissage. En somme, ce projet a été formateur à plusieurs niveaux, et constitue une étape significative dans mon parcours de formation.

[Lien vers le GitHub](#)

## Annexes

### 1. Etudiant n°1

#### Annexe n°1

```
# Fonction qui génère des valeurs factices
def generate_fake_data():
    pluviometer_value = round(random.uniform(0, 20), 2) # Valeur aléatoire pour le pluviomètre
    limnimeter_value = round(random.uniform(0, 10), 2) # Valeur aléatoire pour le limnimètre
    return pluviometer_value, limnimeter_value

# Fonction pour insérer les données factices dans la BDD
def insert_fake_data():
    pluviometer_value, limnimeter_value = generate_fake_data()
    current_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    # Insertion des données du pluviomètre
    cursor.execute('''
    INSERT INTO Mesure (capteur, valeur, unite, date)
    VALUES (%s, %s, %s, %s)
    ''', (1,pluviometer_value, 'L/m²', current_time)) #Capteur numéro 1, valeur générée aléatoirement, unité et la date.

    # Insertion des données du limnimètre
    cursor.execute('''
    INSERT INTO Mesure (capteur, valeur, unite, date)
    VALUES (%s, %s, %s, %s)
    ''', (2, limnimeter_value, 'm', current_time)) #Capteur numéro 2, valeur générée aléatoirement, unité et la date.

    conn.commit()
    print(f"Pluviomètre: {pluviometer_value} L/m², Limnimètre: {limnimeter_value} m à {current_time}")

# Envoi des données toutes les secondes
try:
    while True:
        insert_fake_data()
        time.sleep(1)
except KeyboardInterrupt:
    print("Fin")
finally:
    conn.close() # Fermeture connexion
```

Lien github vers le programme :

[https://github.com/Alexisalou/eau\\_pure/blob/main/LE%20SOURNE/Bouchon\\_test\\_capteur.py](https://github.com/Alexisalou/eau_pure/blob/main/LE%20SOURNE/Bouchon_test_capteur.py)

## Annexe n°2

```

13     # Récupérer le seuil du préleur
14     cursor.execute("SELECT seuil_pluviometre FROM Preleur")
15     seuil_pluviometre = cursor.fetchone()
16
17     # Récupérer le seuil du limnimètre
18     cursor.execute("SELECT seuil_limmimetre FROM Preleur")
19     seuil_limmimetre = cursor.fetchone()
20
21     # Récupérer la dernière mesure du pluviomètre
22     cursor.execute("SELECT valeur FROM Mesure WHERE capteur = 1 ORDER BY date DESC LIMIT 1")
23     mesure_pluviometre = cursor.fetchone()
24
25     # Récupérer la dernière mesure du limnimètre
26     cursor.execute("SELECT valeur FROM Mesure WHERE capteur = 2 ORDER BY date DESC LIMIT 1")
27     mesure_limmimetre = cursor.fetchone()
28
29     db.close()
30
31     if mesure_pluviometre and mesure_limmimetre:
32         if mesure_pluviometre[0] >= 20 or mesure_limmimetre[0] > seuil_limmimetre[0]:    # Conditions pour comparer les mesures
33             return 'Préleur ouvert'
34         else:
35             return 'Préleur fermé'
36
37     raise ValueError("Les données nécessaires ne sont pas disponibles.")
38
39     # Information de connexion BDD
40     if __name__ == '__main__':
41         db_config = {
42             'host': '10.0.14.4',
43             'user': 'root',
44             'password': 'ieufdl',
45             'database': 'eau_pure',
46             'port': '9999'
47         }
48         result = check_and_open_prelever(db_config)
49         print(result) # Affichage du résultat (préleur ouvert ou fermé)

```

Lien github vers le programme :

[https://github.com/Alexisalou/eau\\_pure/blob/main/LE%20SOURNE/Bouchon\\_test\\_pr%C3%A9leveur.py](https://github.com/Alexisalou/eau_pure/blob/main/LE%20SOURNE/Bouchon_test_pr%C3%A9leveur.py)

## Annexe n°3

```

1     import RPi.GPIO as GPIO
2
3     class Sensor:
4         def __init__(self, pin, pull_up_down=GPIO.PUD_UP):
5             self.pin = pin
6             self.pull_up_down = pull_up_down
7             self.previous_state = GPIO.LOW
8
9             # Configuration GPIO
10            GPIO.setmode(GPIO.BCM)
11            GPIO.setup(self.pin, GPIO.IN, pull_up_down=self.pull_up_down)
12
13        def read_state(self):
14            return GPIO.input(self.pin)
15
16        def cleanup(self):
17            GPIO.cleanup()

```

Lien github vers le programme :

[https://github.com/Alexisalou/eau\\_pure/blob/main/LE%20SOURNE/sensor.py](https://github.com/Alexisalou/eau_pure/blob/main/LE%20SOURNE/sensor.py)

## Annexe n°4

```

24  v   class WaterLevelSensor(Sensor):
25  v       def __init__(self,
26  v           pin,
27  v           adc_channel,
28  v           v_ref=3.3,
29  v           correction_factor=1.928 / 0.548,
30  v           resistance=250,
31  v           min_current=4,
32  v           max_current=20,
33  v           max_depth=5,
34  v           pull_up_down=GPIO.PUD_UP
35  ):
36      super().__init__(pin, pull_up_down)
37      self.adc = MCP3208()
38      self.adc_channel = adc_channel
39      self.v_ref = v_ref
40      self.correction_factor = correction_factor
41      self.resistance = resistance
42      self.min_current = min_current
43      self.max_current = max_current
44      self.max_depth = max_depth
45
46
47  v   def read_value(self):
48      raw_value = self.adc.read_channel(self.adc_channel)
49      if raw_value is None:
50          return None, None
51      voltage = (raw_value * self.v_ref) / 4096.0
52      corrected_voltage = voltage * self.correction_factor
53      return raw_value, corrected_voltage
54
55  v   def calculate_current(self, voltage):
56      current = (voltage / self.resistance) * 1000 # en mA
57      return current
58
59  v   def calculate_depth(self, current):
60      if current < self.min_current:
61          current = self.min_current
62      elif current > self.max_current:
63          current = self.max_current
64      depth = ((current - self.min_current) / (self.max_current - self.min_current)) * self.max_depth
65      return depth

```

Lien github vers le programme :

[https://github.com/Alexisalou/eau\\_pure/blob/main/LE%20SOURNE/limnim%C3%A8tre.py](https://github.com/Alexisalou/eau_pure/blob/main/LE%20SOURNE/limnim%C3%A8tre.py)

## Annexe n°5

```

1   import RPi.GPIO as GPIO
2   import time
3   from datetime import datetime, timedelta
4   from sensor import Sensor
5
6   class RainSensor(Sensor):
7       def __init__(self, pin, impulse_to_rain_mm):
8           super().__init__(pin)
9           self.impulse_to_rain_mm = impulse_to_rain_mm
10          self.rain_count = 0
11          self.rainfall_mm = 0.0
12          self.last_impulse_time = datetime.now()
13
14      def update(self):
15          # Lire l'état actuel de la broche
16          current_state = self.read_state()
17
18          # Vérifier si l'état est passé de haut à bas (impulsion détectée)
19          if self.previous_state == GPIO.HIGH and current_state == GPIO.LOW:
20              self.rain_count += 1
21              self.rainfall_mm += self.impulse_to_rain_mm
22              self.last_impulse_time = datetime.now()
23              print(f"Impulsion détectée ! Total : {self.rain_count}, Pluie tombée : {self.rainfall_mm:.2f} L/m²")
24
25          # Mettre à jour l'état précédent
26          self.previous_state = current_state
27
28      def reset(self):
29          self.rain_count = 0
30          self.rainfall_mm = 0.0
31
32      def get_rainfall(self):
33          return self.rainfall_mm
34
35      def cleanup(self):
36          GPIO.cleanup()

```

Lien github vers le programme :

[https://github.com/Alexisalou/eau\\_pure/blob/main/LE%20SOURNE/rain\\_sensor.py](https://github.com/Alexisalou/eau_pure/blob/main/LE%20SOURNE/rain_sensor.py)

## Annexe n°6

Lien github vers le programme :

[https://github.com/Alexisalou/eau\\_pure/blob/main/LE%20SOURNE/main.py](https://github.com/Alexisalou/eau_pure/blob/main/LE%20SOURNE/main.py)

## 2. Etudiant n°2

```

code test clé gsm :
import pytest
from unittest.mock import patch, MagicMock
from api import send_sms

# Test : SMS envoyé avec succès
@patch('api.Client')
@patch('api.Connection')
def test_send_sms_success(mock_connection, mock_client_class):
    # Mock de la connexion Huawei
    mock_client = MagicMock()
    mock_client.sms.send_sms.return_value = "OK"
    mock_client_class.return_value = mock_client

    mock_conn_instance = MagicMock()
    mock_connection.return_value.__enter__.return_value = mock_conn_instance

    send_sms('+33612345678', 'Hello')
    mock_client sms.assert_called_once_with('+33612345678', 'Hello')

# Test : Erreur de contenu vide
@patch('api.Client')
@patch('api.Connection')
def test_send_sms_empty_message(mock_connection, mock_client_class):
    mock_client = MagicMock()
    mock_client.sms.send_sms.side_effect = Exception("Message vide")
    mock_client_class.return_value = mock_client

    mock_conn_instance = MagicMock()
    mock_connection.return_value.__enter__.return_value = mock_conn_instance

    with pytest.raises(Exception, match="Message vide"):
        send_sms('+33612345678', '')

# Test : Numéro invalide
@patch('api.Client')
@patch('api.Connection')
def test_send_sms_invalid_number(mock_connection, mock_client_class):
    mock_client = MagicMock()
    mock_client.sms.send_sms.side_effect = Exception("Numéro invalide")
    mock_client_class.return_value = mock_client

    mock_conn_instance = MagicMock()
    mock_connection.return_value.__enter__.return_value = mock_conn_instance

    with pytest.raises(Exception, match="Numéro invalide"):
        send_sms('INVALID', 'Hello')

```

*code test interface python :*

```
import pytest
from unittest.mock import patch, MagicMock
import interf

# =====
# Test de la fonction Envois_mesures
# =====

@patch('interf.mysql.connector.connect')
def test_envois_mesures(mock_connect):
    """
    Teste si Envois_mesures insère correctement une mesure dans la base de données.
    """

    # Création de mocks pour la connexion et le curseur
    mock_conn = MagicMock()
    mock_cursor = MagicMock()

    # Configuration du comportement des mocks
    mock_connect.return_value = mock_conn
    mock_conn.cursor.return_value = mock_cursor
    mock_conn.is_connected.return_value = True

    # Appel de la fonction avec des données factices
    interf.Envois_mesures(
        capteur=1,
        valeur=10.5,
        unite='L/m²',
        date='2025-04-28 10:00:00'
    )
```

```

# Vérification des appels SQL
mock_connect.assert_called_once()
mock_cursor.execute.assert_called_once_with(
    """
    INSERT INTO Mesure (capteur, valeur, unite, date)
    VALUES (%s, %s, %s, %s)
    """
    (1, 10.5, 'L/m²', '2025-04-28 10:00:00')
)
mock_conn.commit.assert_called_once()
mock_cursor.close.assert_called_once()
mock_conn.close.assert_called_once()

# =====
# Test de la fonction lire_seuils
# =====

@patch('interf.mysql.connector.connect')
def test_lire_seuils(mock_connect):
    """
    Teste si lire_seuils récupère correctement les seuils depuis la base.
    """

    # Création de mocks pour la connexion et le curseur
    mock_conn = MagicMock()
    mock_cursor = MagicMock()

    # Configuration du comportement
    mock_connect.return_value = mock_conn
    mock_conn.cursor.return_value = mock_cursor

    # Valeurs de seuils simulées
    mock_cursor.fetchone.side_effect = [(15.5,), (7.2,)]

    # Appel de la fonction
    db_config = {
        'host': 'localhost',
        'user': 'root',
        'password': 'root',
        'database': 'eau_pure',
        'port': 3306
    }
    seuil_pluviometre, seuil_limnimetre = interf.lire_seuils(db_config)

    # Vérification des résultats
    assert seuil_pluviometre == 15.5
    assert seuil_limnimetre == 7.2
    mock_conn.close.assert_called_once()

# =====

```

```
# Test de la fonction lire_mesures
# =====

@patch('interf.mysql.connector.connect')
def test_lire_mesures(mock_connect):
    """
    Teste si lire_mesures récupère correctement les dernières mesures depuis la base.
    """

    # Création de mocks
    mock_conn = MagicMock()
    mock_cursor = MagicMock()

    mock_connect.return_value = mock_conn
    mock_conn.cursor.return_value = mock_cursor

    # Données simulées : dernière valeur du pluviomètre et du limnimètre
    mock_cursor.fetchone.side_effect = [(12.3,), (4.5,)]

    db_config = {
        'host': 'localhost',
        'user': 'root',
        'password': 'root',
        'database': 'eau_pure',
        'port': 3306
    }
    mesure_pluviometre, mesure_limnimetre = interf.lire_mesures(db_config)

    # Vérification des résultats
    assert mesure_pluviometre == 12.3
    assert mesure_limnimetre == 4.5
    mock_conn.close.assert_called_once()

def test_envois_mesures_erreur_logique():
    """
    Exemple de test erroné : appel incorrect de la fonction Envois_mesures
    avec un paramètre de type invalide (valeur = string au lieu de float).
    Ce test échouera car la base de données attend un float pour 'valeur'.
    """

    with pytest.raises(Exception): # On s'attend à une exception
        interf.Envois_mesures(1, 'dix', 'L/m2', '2025-04-28 10:00:00')
```

### Interface BDD STATION

```

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_limmimetre FROM Preleveur")
    seuil_limmimetre = cursor.fetchone()

    db.close()

    if seuil_pluviometre and seuil_limmimetre:
        return seuil_pluviometre[0], seuil_limmimetre[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_limmimetre = cursor.fetchone()

    db.close()

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

fonction send\_sms :

```
from huawei_lte_api.Client import Client
from huawei_lte_api.Connection import Connection

# URL de l'API HiLink
base_url = 'http://192.168.8.1'

# Créer une connexion
with Connection(base_url) as connection:
    client = Client(connection)
    def send_sms(phone, message):

        # Envoyer un SMS
        sms_data = {
            'Index': '-1',
            'Phones': {'Phone': [phone]},
            'Content': message,
            'Length': len(message),
            'Reserved': '1',
            'Date': '2025-04-03 11:45:40'
        }

        response = client.sms.send_sms(phone,message)
        print('Réponse de l'API:', response)
        send_sms('+33643872007','test conan')
```

### 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 tasks like 'Mise en oeuvre architecture réseau', 'Mise en place sécurité réseau', and 'Données ouvertes et API'.
- Carnet de Sprint**: Contains tasks like 'Création des identifiants pour les techniciens', 'OpenStreetMap / Leaflet', and 'tests automatisés'.
- En cours**: Contains tasks like 'Documentation RGPD', 'Mise en place du Git', and 'Bouchon de test capteurs'.
- Réalisés**: Contains tasks like 'modélisation graphique de la BDD', 'Modélisation et mise en place de la BDD', and 'Réalisation d'un tableau SCRUM'.
- Validés**: Contains tasks like 'Documentation de la BDD', 'Réalisation d'un programme permettant d'hasher et saler un mot de passe', and 'Adressage architecture réseau'.

Figure n°2 : 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:**

- Title:** Recherche technologie LoRaWAN #9
- Description:** En tant que technicien, je veux me documenter sur la technologie LoRaWAN afin d'envoyer des mesures.
- Assignee:** MathysLS
- Status:** Carnet de Sprint
- Comments:** Several comments from MathysLS and Alexisalou are visible.

**Planning Poker Session:**

- Participants:** Room 09 68 27 18
- Card Values:** 0, 0.5, 1, 2, 3, 5, 8, 13, 20, 40, 100
- Results Table:**

| Name     | Story Points |
|----------|--------------|
| alexis   | 8            |
| Eau_pure | 8            |
| Mathys   | 8            |
| Nolan    | 8            |

Figure n°3 : 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°6 : Réalisation d'un formulaire de connexion



Figure n°7 : Pop-up de connexion administrateur

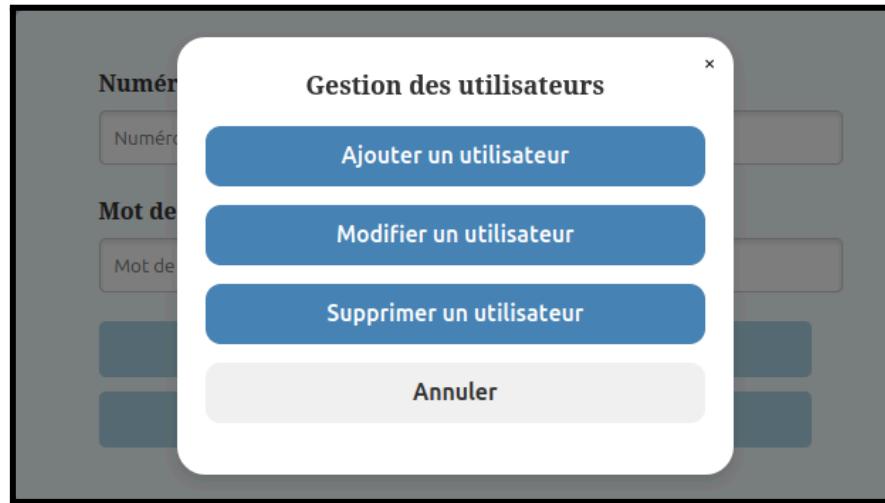


Figure n°8 : Pop-up de gestion des utilisateurs

The image displays three separate modal dialogs:

- Ajouter un utilisateur**: Contains fields for "Numéro de téléphone", "Confirmer le numéro", "Mot de passe", "Confirmer le mot de passe", and two buttons "Ajouter" and "Annuler".
- Modifier le mot de passe**: Contains fields for "Numéro de téléphone", "Mot de passe actuel", "Nouveau mot de passe", "Confirmer le nouveau mot de passe", and two buttons "Modifier" and "Annuler".
- Supprimer un utilisateur**: Contains fields for "Numéro de téléphone", "Confirmer le numéro", "Mot de passe actuel", "Confirmer le mot de passe", and two buttons "Supprimer" and "Annuler".

Figure n°9 : 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  $\text{mS}/\text{cm}$  (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

Figure n°10 : Réalisation du formulaire pour la saisie des données physico-chimiques

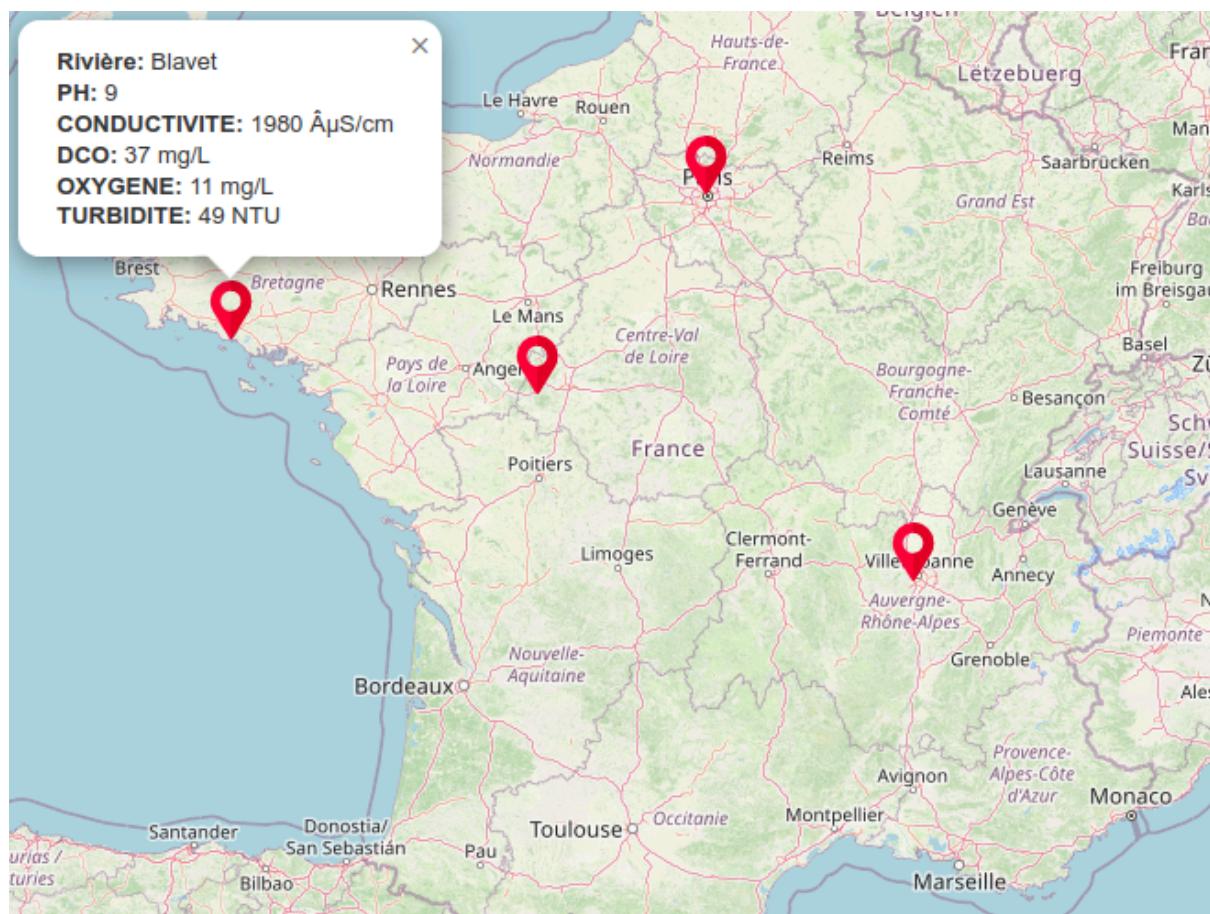
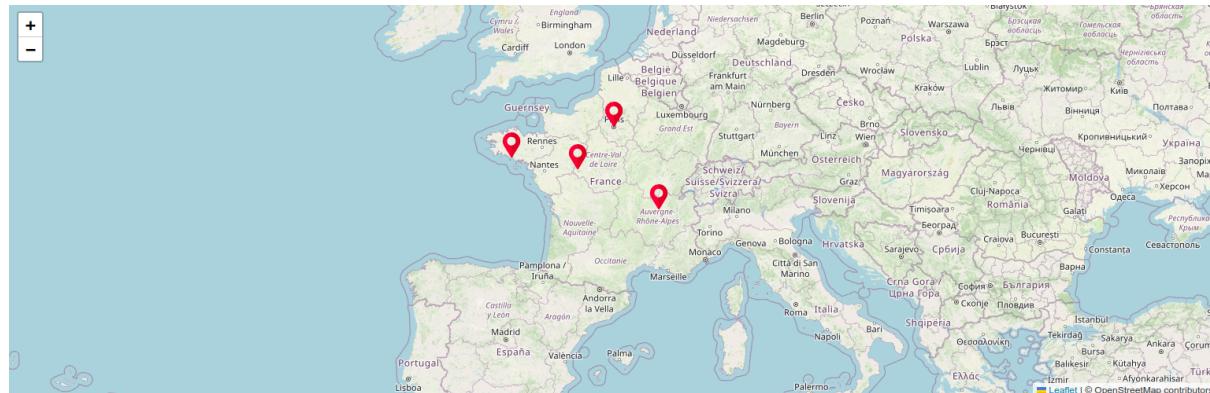


Figure n°11 : Présentation des mesures de manière géolocalisée

### Graphiques des données physico-chimiques - Blavet



Figure n°12 : Présentation des mesures de manière graphique