



Projet Borne de Gel

Lycée Polyvalent Jean Rostand
Villepinte

2024

BTS - Snir

Présenté par :

BOUAKSIM Adam
DIARRA Ladjib Ibrahim
DE ALMEIDA Toni
LOZANO CHAHUA Dally Milagros

SOMMAIRE

REMERCIEMENTS	page 3
INTRODUCTION	page 4
I. PRÉSENTATION DU PROJET	page 5-14
1.1) Contexte	page 5
1.2) Synoptique général du projet	page 6
1.3) Cahier des charges	page 7-8
1.4) Diagramme de cas d'utilisation	page 9-10
1.5) Diagramme de Séquence	page 11-12
1.6) Mock Serveur	page 13-14
II. CONCEPTION DE BASE DE DONNÉES ET API - Diarra Ladj (Étudiant 1)	
2.1) Introduction de la Base de Données et de l'API	page 15
2.2) Présentations des tables.	page 16-17
2.3) Algorithmes d'API (WEB - APPLICATION - IOT)	page 18-20
2.4) Problèmes rencontrés lors de ma partie de projet	page 21-23
III. SITE WEB - De Almeida Toni (Étudiant 2)	
3.1) Synoptique du site web	page 24
3.2) Résumé des tâches réalisées par l'utilisateur	page 25
3.3) Site SmartGel	page 26-37
3.4) Défis rencontrés	page 37
IV. OBJET CONNECTÉ IOT - Bouaksim Adam (Etudiant 3)	
4.1) Présentation du matériels	page 38-39
4.2) Acquérir le niveau de Gel	page 40-43
4.3) Acquérir le niveau de Batterie	page 44-47
4.4) Transmission des informations à l'API	page 47-49
4.5) Mode d'économie d'énergie	page 49
4.6) Résultat final	page 50-51
V. APPLICATION ANDROID - Lozano Dally (Etudiant 4)	
5.1) Synoptique de l'application	page 52
5.2) Résumé des tâches réalisées par utilisateur	page 53
5.3) Application SmartGel	page 54-60
5.4) Contraintes	page 61
VI. CONCLUSION	page 62
VII. ANNEXES	page 63

REMERCIEMENTS

La rédaction de ce rapport marque la conclusion d'une étape cruciale de notre formation en BTS, un parcours que nous avons eu la chance de vivre en équipe grâce au projet "Borne de Gel". Ce dernier a été déterminant pour l'obtention de notre diplôme et nous a permis de développer des compétences clés en informatique et en physique.

Tout d'abord, nous tenons à exprimer notre profonde gratitude à Monsieur TOBJI et Monsieur MADANI, professeurs d'informatique. Leur expertise, leur soutien inébranlable et leur encouragement constant ont été indispensables à la réussite de ce projet. Leur disponibilité et leur volonté de partager leurs connaissances ont grandement enrichi notre apprentissage.

Nous souhaitons également remercier chaleureusement Monsieur LAVARENNE, professeur de physique, pour ses précieuses contributions et ses conseils avisés tout au long de notre parcours. Son approche pédagogique et son engagement envers notre réussite ont été des éléments moteurs dans l'accomplissement de notre projet.

Ce projet, réalisé en équipe, a été une expérience formatrice et stimulante, et c'est avec une immense reconnaissance que nous adressons ces remerciements à nos professeurs. Leur accompagnement et leur soutien ont été des facteurs déterminants dans la réussite de notre formation.

Merci de la part de toute l'équipe.

INTRODUCTION

L'élaboration de ce projet nous a offert l'opportunité de collaborer étroitement, de partager nos compétences et d'apprendre de nouvelles connaissances essentielles à notre domaine d'études tout en reflétant l'importance du travail d'équipe et notre immersion dans les aspects techniques.

Travailler en équipe nous a permis de développer des compétences interpersonnelles et professionnelles, renforçant notre capacité à résoudre des problèmes complexes et à innover ensemble. Cette synergie de groupe a été indispensable pour surmonter les défis techniques et aboutir à des solutions efficaces.

L'immersion dans le projet "Borne de Gel" nous a plongé au cœur des technologies de l'informatique et de la physique, nous confrontant à des situations concrètes et exigeantes. Grâce à cet apprentissage pratique, nous avons acquis une compréhension approfondie des concepts théoriques, tout en développant des compétences pratiques essentielles pour notre future carrière.

En somme, ce projet a été une expérience riche et formatrice, nous préparant efficacement à intégrer le monde professionnel. Ce rapport est le reflet de notre engagement, de nos efforts collectifs et des précieuses leçons apprises tout au long de cette aventure.

I.PRÉSENTATION DU PROJET

1.1) Context

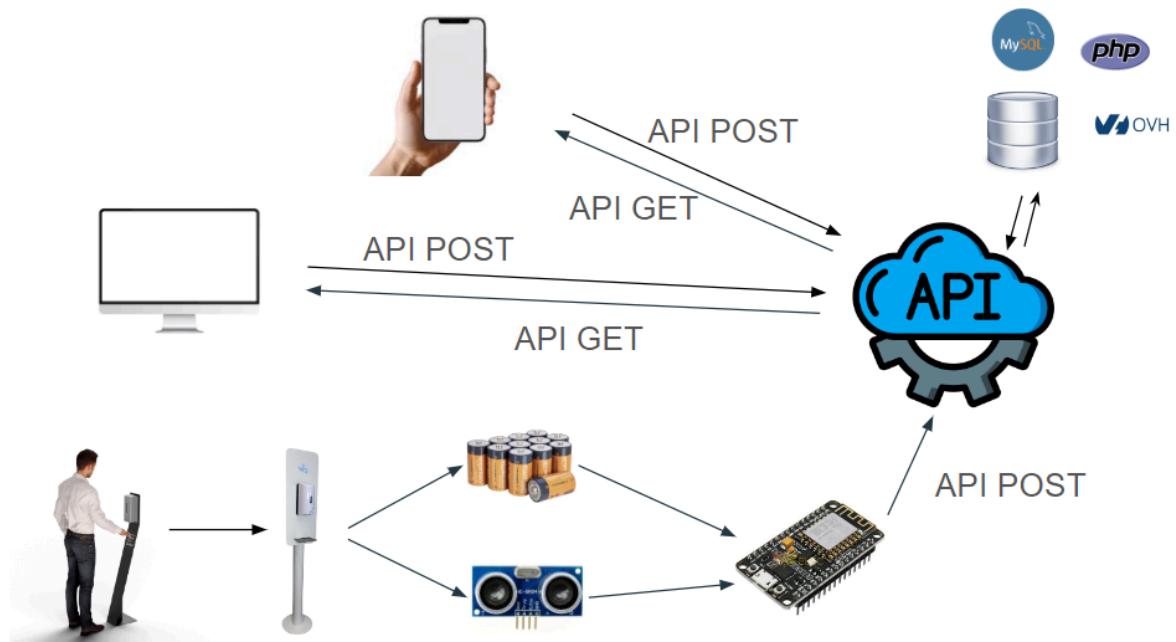
Dans le sillage de la pandémie de COVID-19, l'utilisation de bornes de gel est devenue omniprésente dans de nombreux établissements, devenant un élément essentiel pour garantir une hygiène optimale des mains. Dans ce contexte, nous nous sommes penchés sur le développement d'une solution novatrice. Notre vision est d'optimiser la gestion de ces dispositifs en automatisant la surveillance du niveau de gel et de la batterie, offrant ainsi une approche plus efficace et pratique pour assurer la sécurité sanitaire des utilisateurs.

La borne de gel réalise sa fonction de base : distribution de gel lors de la présence d'une main . Pour améliorer son efficacité, notre équipe cherche à mettre en place une technologie qui nous permettrait de superviser les niveaux de gel et de batterie en temps réel.

L'idée est de centraliser ces informations dans une base de données, permettant ainsi une gestion proactive de l'entretien. Un personnel dédié peut être alerté en temps opportun pour remplir le gel ou changer la batterie, éliminant ainsi le besoin de vérifications constantes de bornes physiquement.

Dans le rapport, nous soulignerons les avantages pratiques de cette automatisation, l'efficacité opérationnelle accrue, et la capacité à réagir rapidement aux besoins spécifiques de chaque borne. Nous insisterons également sur l'importance de cette solution dans un contexte où l'hygiène des mains est primordiale.

1.2) Synoptique général du projet

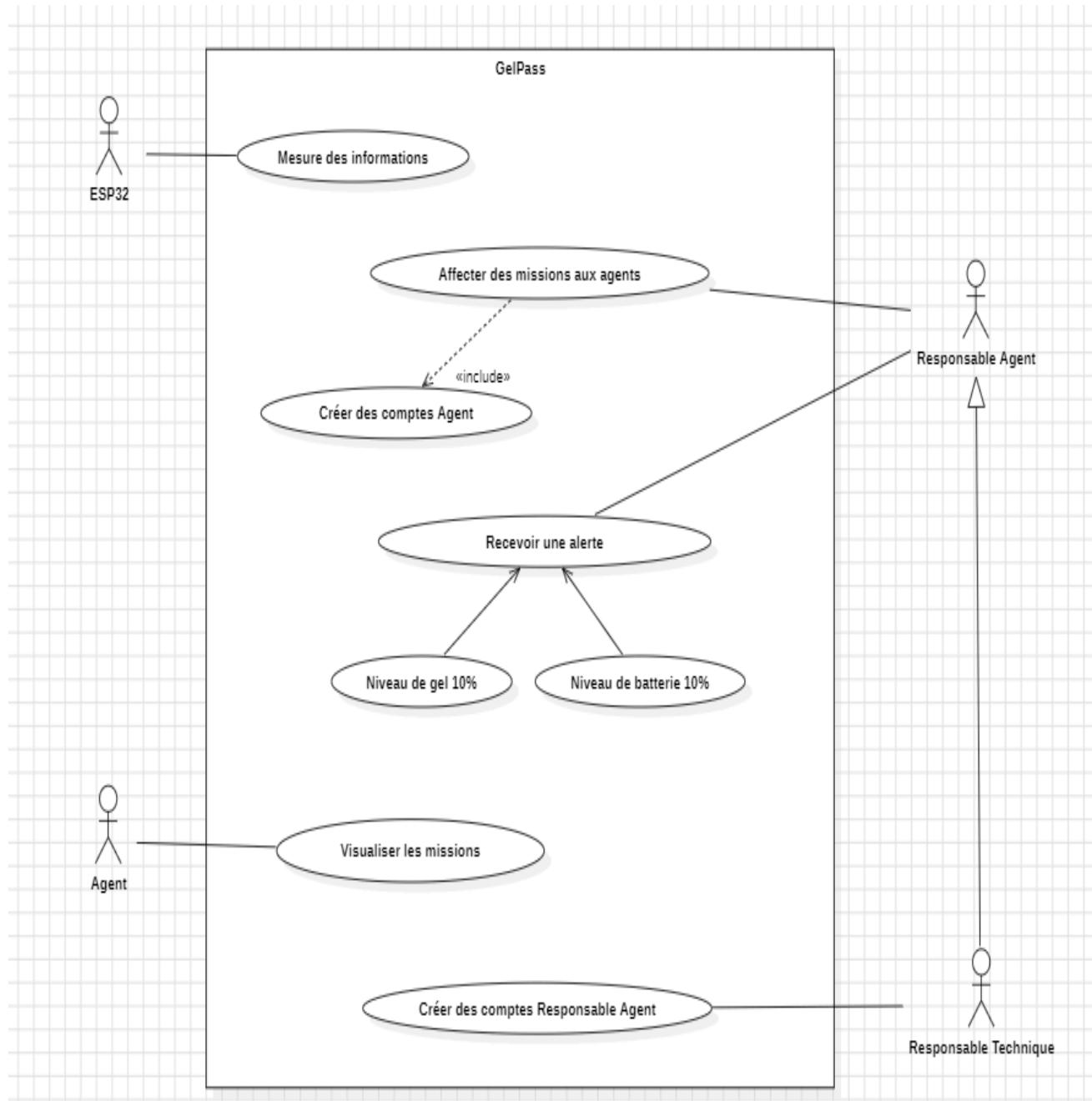


1.3) Cahier des charges

	Fonctions à développer et tâches à effectuer	
Étudiant 1	<ul style="list-style-type: none"> • Modéliser la Base de données • Mettre en œuvre la base de données (création des tables avec les relations) • Mise en œuvre d'une API Rest permettant de fournir un fichier JSON réponse exploitable par l'application ou le site web 	<p>Installation : Installation OS , Workbench , bdd mysql</p> <p>Mise en œuvre : mise en œuvre de la bdd et des fichiers Json</p> <p>Configuration : Des accès au serveur de bdd</p> <p>Réalisation : SGBD avec arborescence REST</p> <p>Documentation : Installation, Prise en main et déploiement</p>
Étudiant 2	<ul style="list-style-type: none"> • Coder le site web Administrateur : <ul style="list-style-type: none"> - Gestion des différents statuts • Coder le site web côté membre • Héberger le site sur le serveur 	<p>Installation : Installation OS , IDE, client Ftp</p> <p>Mise en œuvre :: mise en œuvre du code HTML, CSS, javaScript et PHP</p> <p>Configuration : Des serveurs et de l'IDE,</p> <p>Réalisation : Site web et hébergement distant</p> <p>Documentation : d'Installation, de prise en main et déploiement de la solution</p>
Étudiant 3	<ul style="list-style-type: none"> • Coder les bornes électroniques connectés : - Choix d'une carte ESP de petite 	<p>Installation : Installation OS ou IDE de la carte électronique</p> <p>Mise en œuvre : cadenas électronique,</p>

	<p>taille</p> <ul style="list-style-type: none"> - Configurer l'adressage réseau - Développer le code embarqué - Valider la mesure de niveau de liquide avec un capteur - Choisir une carte permettant l'acquisition de ce signal - Adapter le signal de tension de batterie et le mesurer - Adapter le signal IR de détection de main pour déclencher le réveil la partie électronique - Développer le code embarqué - Tester la communication au format JSON avec l'API REST. 	<p>capteurs de niveau de gel, de batterie</p> <p>Configuration : configuration réseau</p> <p>Réalisation : Du prototype de la partie électronique avec tests de validation pour chaque sous-partie</p> <p>Documentation : d'Installation, de prise en main et déploiement de la solution</p>
Étudiant 4	<ul style="list-style-type: none"> ● Coder l'interface de l'application Andoid ● Mettre en œuvre la communication via JSON vers l'API ● Récupérer les données et créer des interfaces dynamiques (alertes, visualiser des niveaux des bornes etc..) 	<p>Installation : Installation OS et IDE Android Studio</p> <p>Mise en œuvre : Application Android sur tablette</p> <p>Configuration : Android Studio avec tablette virtuelle.</p> <p>Réalisation : Application sécurisée</p> <p>Documentation : Installation</p>

1.4) Diagramme de cas d'utilisation



Dans notre projet, le diagramme de cas d'utilisation illustre les interactions entre quatre types d'utilisateurs et le système de gestion des bornes de gel.

L'utilisateur physique : Il s'agit de la personne qui utilise la borne de gel physiquement pour se désinfecter les mains. Cette interaction est directe et simple : l'utilisateur place sa main sous la borne pour recevoir une dose de gel désinfectant.

Le responsable technique : Cet utilisateur joue un rôle clé dans la supervision globale du système. Il se connecte à la page web ou à l'application pour surveiller les niveaux de gel et de batterie de toutes les bornes. De plus, il a la responsabilité de créer les comptes des responsables agents, leur permettant ainsi d'accéder au système et de gérer les opérations sur le terrain.

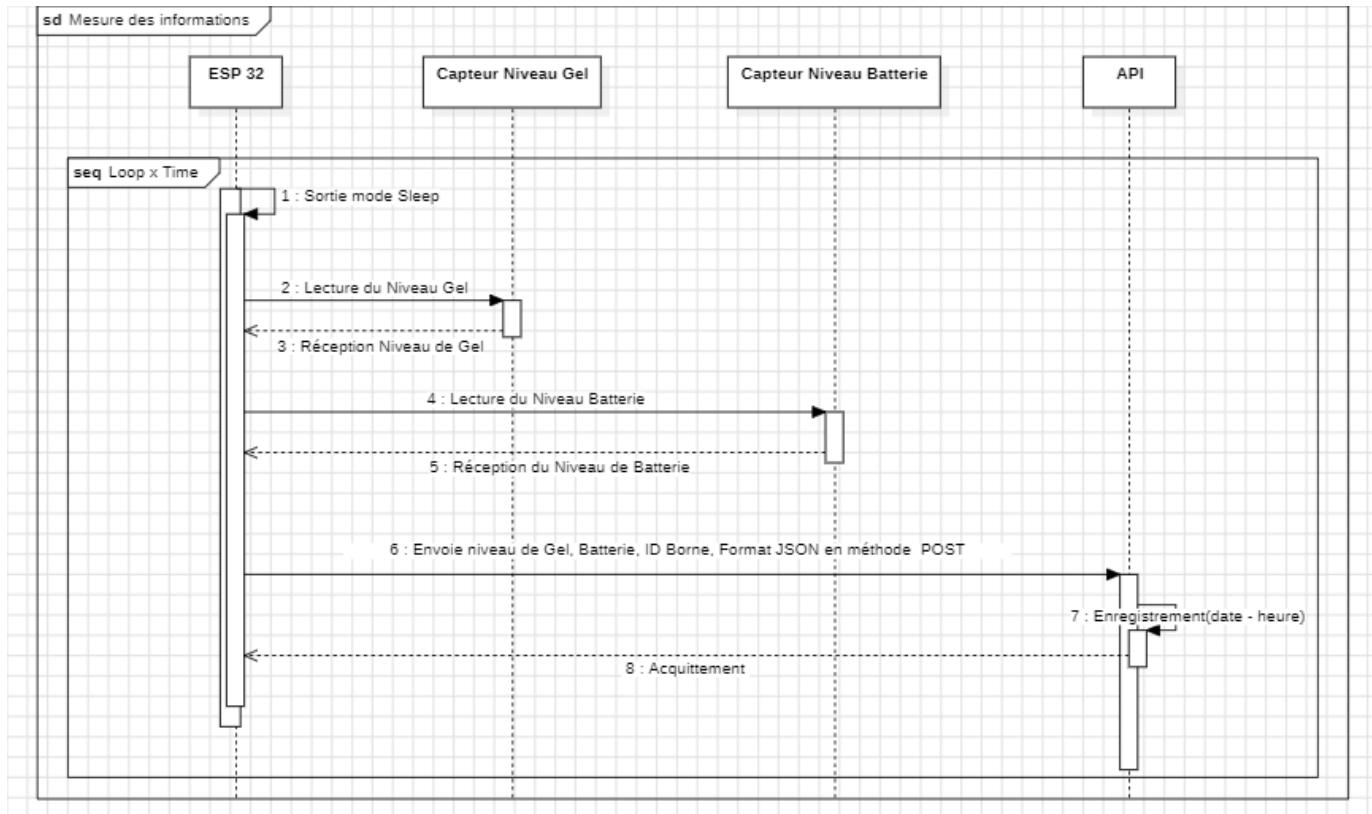
Le responsable agent : Ce rôle est dédié à la gestion des agents sur le terrain. Le responsable agent utilise la page web ou l'application pour consulter les informations relatives aux bornes et pour attribuer des missions aux agents mais aussi il est responsable des comptes des "agents". Par exemple, il peut assigner une tâche à un agent pour remplir une borne de gel ou changer une batterie lorsque le système indique que cela est nécessaire.

L'agent : Les agents sont les utilisateurs qui effectuent les tâches assignées par le responsable agent. En se connectant à la page web ou à l'application, ils peuvent voir les missions qui leur ont été attribuées et agir en conséquence, comme remplir une borne de gel ou remplacer une batterie.

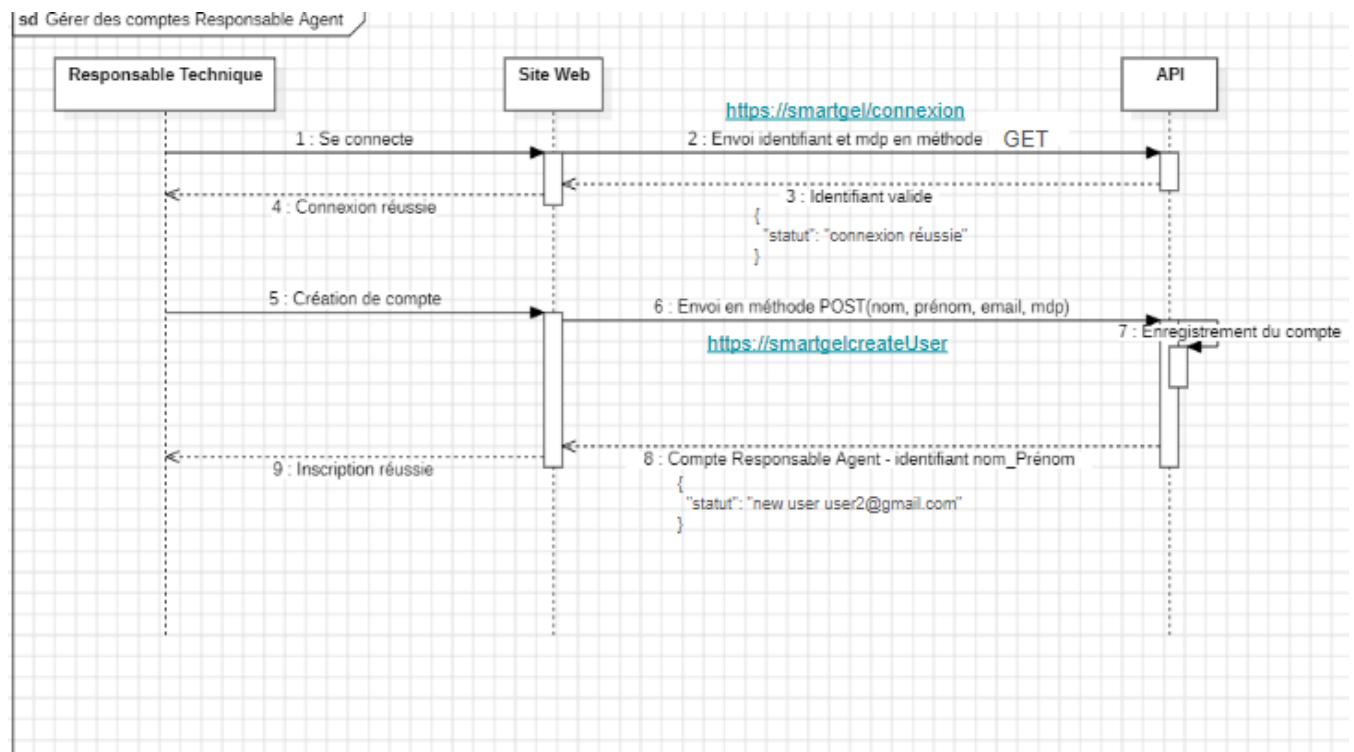
En somme, chaque rôle a des responsabilités spécifiques qui contribuent à la gestion efficace et à l'entretien des bornes, assurant ainsi une hygiène optimale pour tous les utilisateurs.

1.5) Diagramme de Séquence

Mesure des informations



Gérer des comptes de Responsable Agent



1.6) Mock Serveur

Utiliser au début de projet pour interagir avec une API qui nous sert de modèle ou de maquette d'une vraie API.

AYOUT Double Column LANGUAGE Java - OkHttp ⚙

GET Visualiser les données de toutes les bornes d'un établissement

```
https://804b3669-1a04-43a0-8a07-09a076ab6c78.mock.pstmn.io/dashboard?idEtablissement=1
```

Cette requête permet de voir tous les bornes d'un établissement spécifique. Pour cela, il faut indiquer l'id de l'établissement dans l'URL.

PARAMS

idEtablissement	1
-----------------	---

Exemple de réponse attendue.

Example Response

json

```
{  
    "Etablissement": "Jean Rostand",  
    "Bornes": [  
        {  
            "idBorne": 1,  
            "gel": 33,  
            "batterie": 57,  
            "salle": "GR11",  
            "heure": "15:45",  
            "date": "2024-01-31"  
        },  
        {  
            "idBorne": 2,  
            "gel": 44,  
            "batterie": 21,  
            "salle": "HR09",  
            "heure": "11:30",  
            "date": "2024-02-04"  
        },  
        {  
            "idBorne": 3,  
            "gel": 13,  
            "batterie": 10,  
            "salle": "E209",  
            "heure": "17:10",  
            "date": "2024-02-01"  
        }  
    ]  
}
```

II. CONCEPTION DE BASE DE DONNÉES ET API - Diarra Ladji (Etudiant 1)

2.1) Introduction de la Base de Données et de l'API

Après une période de pandémie sans précédent dans l'histoire de l'humanité, les conditions d'hygiène sont devenues un des éléments clés de la protection des personnes. Dans ce contexte, notre projet vise à assurer la disponibilité continue du gel hydroalcoolique à chaque borne d'un établissement. Pour résoudre ce problème, nous avons équipé chaque borne de gel d'une partie communicante, permettant ainsi de centraliser le suivi et le contrôle des niveaux de gel hydroalcoolique et de batterie, et de générer des alertes en cas de besoin.

La solution repose sur une infrastructure technique composée de deux principaux composants : une base de données relationnelle pour stocker les informations essentielles et une API pour permettre la communication entre les bornes, le serveur central, et les interfaces utilisateurs (web et mobiles).

Présentation de la Base de Données

La base de données relationnelle, créée avec MySQL, est le cœur de notre système. Elle contient plusieurs tables interconnectées qui permettent de gérer les informations sur les bornes de gel, les employés, les établissements, les missions, et les rôles des utilisateurs. Voici une description détaillée de chaque table :

2.2) Présentations des tables

Tables Principales :

Table borne

- IdBorne: Identifiant unique de la borne.
- Niveau_Gel: Niveau actuel de gel hydroalcoolique dans la borne.
- Niveau_Batterie: Niveau actuel de la batterie de la borne.
- Salle: Localisation de la borne.
- Heure: Heure de relevé des données.
- Date: Date de relevé des données.
- Id_Etablissement: Clé étrangère référençant l'établissement auquel la borne est associée.

	IdBorne	Niveau_Gel	Niveau_Batterie	Salle	Heure	Date	Id_Etablissement
▶	1	0	0	GR-01	16:53:09	2024-05-21	1
	2	0	0	GR-11	16:53:09	2024-05-21	1
	3	100	45	E12	15:16:10	2024-05-23	1
	4	100	9	C7	09:32:05	2024-05-22	1
	5	8	20	U20	15:16:19	2024-05-23	1
	6	100	10	C10	16:05:53	2024-05-23	1
	7	100	20	E12	09:31:53	2024-05-22	1
	8	60	84	B02	16:46:37	2024-05-13	2
	9	80	84	C12	16:46:37	2024-05-13	1
	15	100	50	B65	09:31:53	2024-05-22	2
	16	65	65	A78	17:08:48	2024-05-13	1
	18	25	84	S45	09:36:10	2024-05-07	2
	19	100	25	A45	09:31:53	2024-05-22	2

Table employes

- IdEmployes: Identifiant unique de l'employé.
- Nom: Nom de l'employé.
- Prenom: Prénom de l'employé.
- Mail: Adresse email de l'employé.
- Mot_De_Passe: Mot de passe de l'employé (hashé).
- Id_Role: Identifiant du rôle de l'employé.
- Id_Etablissement: Clé étrangère référençant l'établissement auquel l'employé est associé.

	IdEmployes	Nom	Prenom	Mail	Mot_De_Passe	Id_Role	Id_Etablissement
▶	2	Peguy	Charle	Peguy@gmail.com	qsdfgh78	1	3
	3	Diarra	Jean	Diarra@hotmail.fr	wxcvbn12	1	2
	4	DE ALMEIDA	Toni	dealmeida-toni@lycee-jeanrostand.fr	azertyuiop93	1	1
	6	IME	Max	Max@gmail.com	Maxim93420	2	2
	7	DIARRA	Ladji	Diarra@gmail.com	diarra123	2	1
	11	Ibrahim	dally	dally_Ibrahim@gmail.com	Dally741852	1	1
	12	Martin	Michael	ladjibrahimdiarra@gmail.com	Martin1234	3	6
	14	TOTO	Titi	tata@gmail.com	tuktuk93420	1	2
	16	Etoile	Patrick	bob@gmail.com	bob1234	2	2
	17	Aigri	Carlos	Diarra-ladjibrahim@lycee-jeanrostand.fr	Azerty123	2	3
	18	DE	Toni	tonidealmeida9@gmail.com	Almeida93	2	3
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table etablissement

- IdEtablissement: Identifiant unique de l'établissement.
- NomEtablissement: Nom de l'établissement.
- Address: Adresse de l'établissement.
- Id_Employes: Clé étrangère référençant l'employé associé à l'établissement

	IdEtablissement	NomEtablissement	Address	Id_Employes
▶	1	Jean Rostand	243 Bd Robert Ballanger, 93420 Villepinte	12
	2	Charle Peguy	Av. Boris Vian, 93420 Villepinte	12
	3	Francoise Dolto	4 Rue Guy Mousset, 93420 Villepinte	12
*	6	plusieurs etabilissement	null	NULL
	NULL	NULL	NULL	NULL

Table mission

- IdMission: Identifiant unique de la mission.
- Type: Type de mission.
- Heure: Heure de la mission.
- Date: Date de la mission.
- Id_Employes: Clé étrangère référençant l'employé concerné par la mission.
- Id_
- Borne: Clé étrangère référençant la borne concernée par la mission.

	IdMission	Type	Heure	Date	Id_Employes	Id_Borne
▶	3	Niveau de batterie et de gel	11:16:50	2024-03-26	4	4
	8	Niveau de gel	09:04:13	2024-03-27	11	7
	9	Niveau de batterie et de gel	09:09:39	2024-03-27	11	8
	11	Niveau de batterie	09:09:59	2024-03-27	3	3
	13	Niveau de gel	10:46:06	2024-04-26	11	6
	15	Niveau de gel	12:20:51	2024-04-26	4	5
	17	Niveau de gel	17:15:01	2024-04-28	11	9

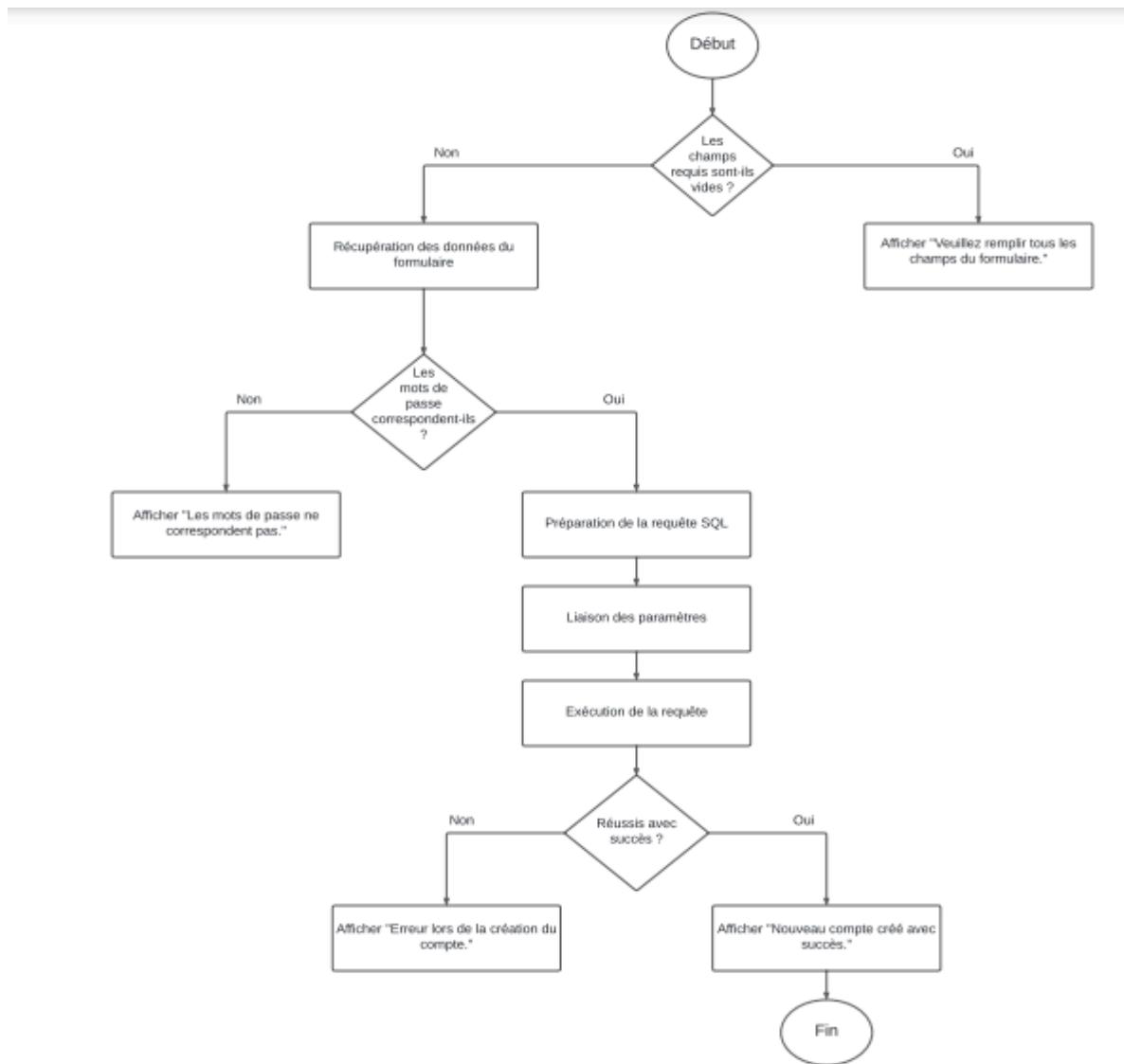
Table rôle

- IdRole: Identifiant unique du rôle.
- Roles: Nom du rôle.

	IdRole	Roles
▶	1	Agent
	2	Responsable Agent
	3	Responsable Technique

2.3) Algorithmes d'API (WEB - APPLICATION - IOT)

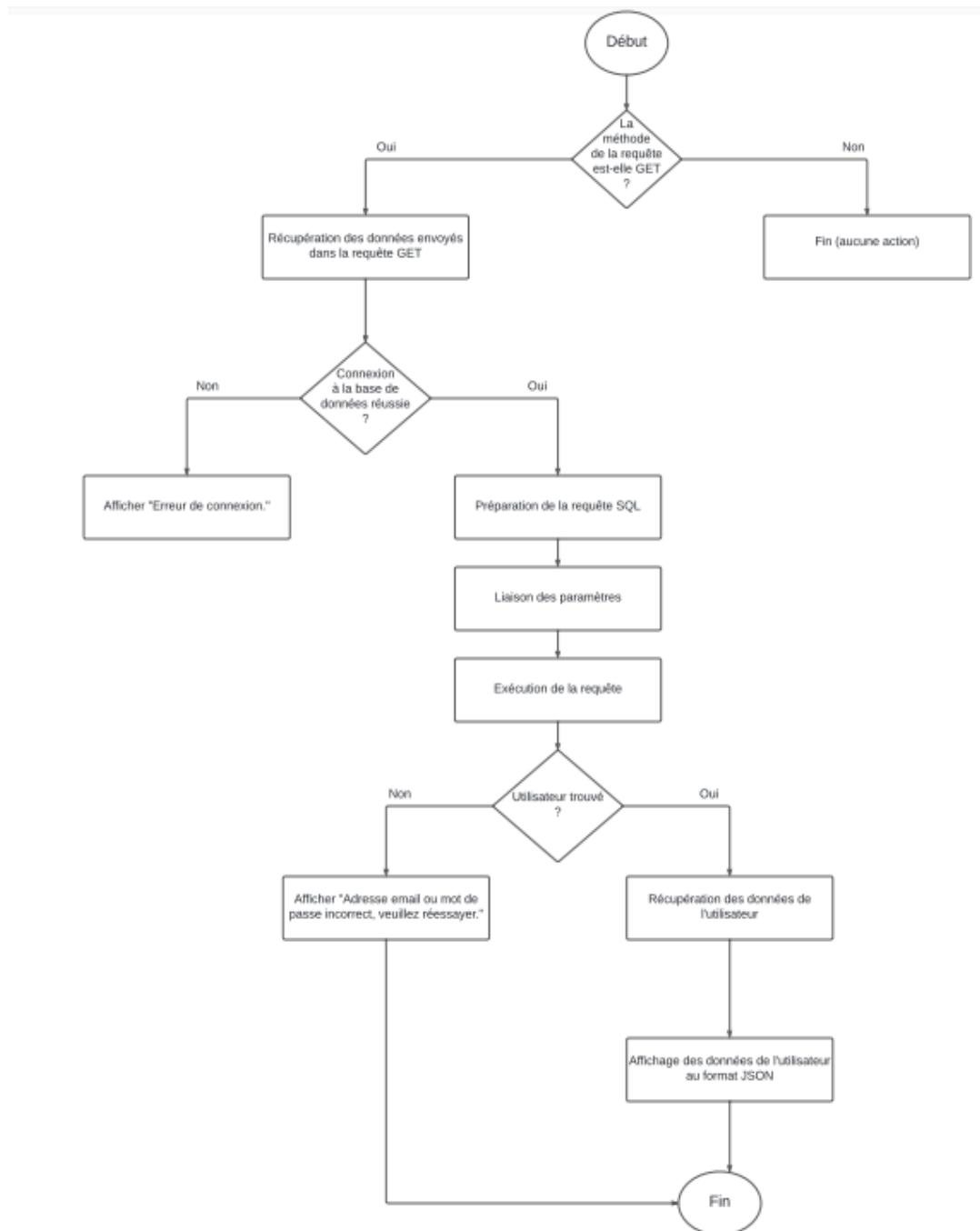
1) Création de compte côté WEB :



L'algorithme de l'API de création de compte commence par vérifier si tous les champs requis du formulaire sont remplis. Si des champs sont vides, un message d'erreur est affiché demandant à l'utilisateur de remplir tous les champs. Ensuite, l'algorithme compare les deux mots de passe fournis par l'utilisateur pour s'assurer qu'ils correspondent. Si les mots de passe ne correspondent pas, un message d'erreur est affiché. Si les mots de passe correspondent, l'algorithme prépare une requête SQL pour insérer les données de l'utilisateur dans la base de données. Les paramètres de la requête sont liés, et la requête est exécutée. Si la création du compte est réussie, un message de succès est affiché.

En cas d'erreur lors de la création du compte, un message d'erreur est affiché.

2) Connexion côté APPLICATION :



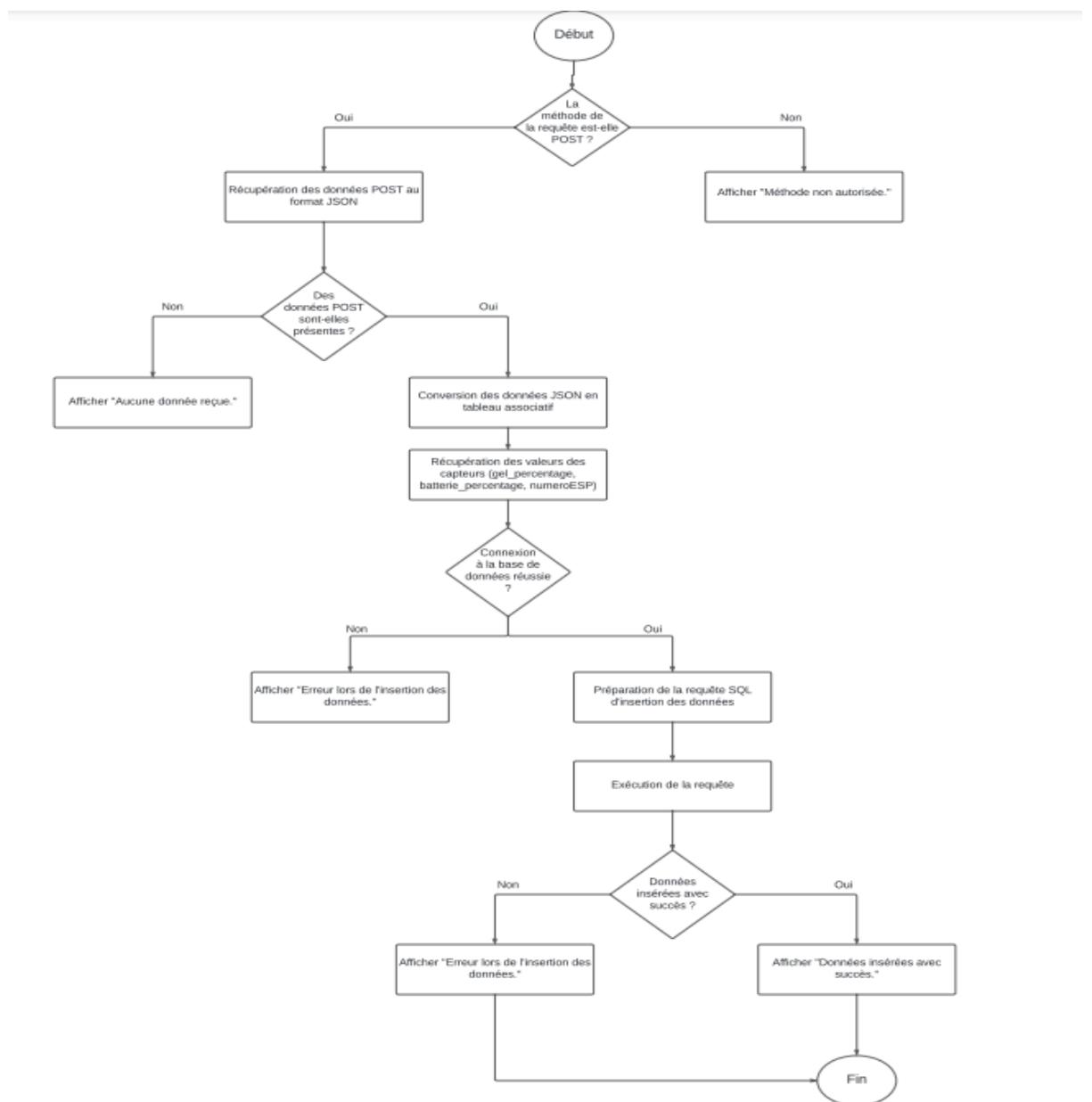
L'API de connexion commence par vérifier si la méthode de la requête est "GET". Si ce n'est pas le cas, le processus se termine immédiatement. Sinon, les données envoyées (email et mot de passe) sont récupérées.

Ensuite, l'API essaie de se connecter à la base de données. Si la connexion échoue, un message d'erreur est renvoyé. Si la connexion réussit, l'API prépare et exécute une requête SQL pour vérifier les informations de l'utilisateur.

Si les informations de l'utilisateur correspondent aux données dans la base de données, les détails de l'utilisateur sont récupérés et renvoyés au format JSON.

Si les informations ne correspondent pas, un message d'erreur indiquant que l'email ou le mot de passe est incorrect est renvoyé.

3) Envoie les informations de la borne côté IOT :



L'API d'insertion de données commence par vérifier si la méthode de la requête est "POST". Si ce n'est pas le cas, un message "Méthode non autorisée" est renvoyé. Si la méthode est correcte, l'API récupère les données POST au format JSON.

Ensuite, l'API vérifie si les données POST sont présentes. Si les données sont absentes, un message "Aucune donnée reçue" est renvoyé. Si les données sont présentes, elles sont converties en tableau associatif, et les valeurs des capteurs (gel_percentage, battery_percentage, numeroESP) sont récupérées.

L'API essaie ensuite de se connecter à la base de données. Si la connexion échoue, un message "Erreur lors de l'insertion des données" est renvoyé. Si la connexion réussit, l'API prépare une requête SQL d'insertion des données, lie les paramètres avec les valeurs récupérées et exécute la requête.

Enfin, si les données sont insérées avec succès, un message "Données insérées avec succès" est renvoyé. Sinon, un message d'erreur est renvoyé pour indiquer l'échec de l'insertion des données.

2.4) Problèmes rencontrés lors de ma partie de projet

1) Problèmes rencontrés lors de la création de la Base de Données

Durant la création de la base de données pour notre projet, plusieurs défis ont émergé :

1. Liens entre les tables :

- L'un des principaux problèmes était de lier correctement les colonnes entre différentes tables. Par exemple, le lien entre la colonne Id_Etablissement de la table employes et la colonne IdEtablissement de la table etablissement a nécessité une attention particulière pour s'assurer que les clés étrangères soient correctement définies et maintenues.
- Des erreurs dans les clés étrangères peuvent entraîner des incohérences dans les données ou des échecs lors des insertions ou des mises à jour. Cela m'a obligé à revoir plusieurs fois la structure des tables et à corriger les erreurs de définition des clés étrangères.

2. Insertion automatique des valeurs de date et d'heure :

- Une autre difficulté était de configurer les colonnes heure et date pour qu'elles se remplissent automatiquement avec l'heure et la date actuelles à chaque nouvelle insertion dans les colonnes Niveau_Gel et Niveau_Batterie.
- Cela nécessitait l'utilisation de triggers ou la configuration correcte des colonnes pour utiliser des fonctions comme NOW() en MySQL, ce qui n'était pas toujours intuitif et a nécessité plusieurs essais et recherches.

2) Problèmes rencontrés lors de la création des APIs

Développer les APIs pour interagir avec la base de données a également présenté plusieurs défis :

1. Erreurs de syntaxe :

- Les erreurs de syntaxe étaient fréquentes, comme l'utilisation incorrecte de majuscules au lieu de minuscules ou l'ajout accidentel de lettres en trop. Par exemple, une erreur dans la déclaration des variables ou des erreurs de typographie dans les noms de colonnes peuvent provoquer des échecs lors de l'exécution des requêtes.
- Une simple faute de frappe dans le nom d'une colonne ou d'une variable a souvent entraîné des erreurs difficiles à diagnostiquer, me faisant perdre du temps à chercher les sources de ces erreurs.

2. Problèmes de connexion à la base de données :

- Parfois, l'exécution des APIs échouait en raison de problèmes de connexion avec la base de données. Ces problèmes étaient souvent liés à des configurations incorrectes ou à des temps de réponse trop longs de la base de données, ce qui provoquait des timeouts.
- La gestion des erreurs de connexion et la mise en place de messages d'erreur clairs pour identifier la cause des problèmes ont nécessité plusieurs ajustements dans le code.

3. Débogage et gestion des erreurs :

- La gestion des erreurs et le débogage des APIs étaient des tâches récurrentes. Des erreurs dans les requêtes SQL ou des réponses inattendues du serveur nécessitaient une analyse minutieuse des logs et des messages d'erreur.
- Parfois, la logique de l'API ne fonctionnait pas comme prévu, par exemple lors de la vérification des données entrantes ou de l'exécution des opérations CRUD (Create, Read, Update, Delete). Cela m'a forcé à revoir le code et à ajuster les contrôles de validation et les manipulations de données.

4. Problèmes de performance :

- L'optimisation des performances des APIs, notamment pour éviter des temps de réponse trop longs, était un autre défi. Les requêtes complexes ou les gros volumes de données pouvaient ralentir les APIs, nécessitant des ajustements pour améliorer l'efficacité des requêtes SQL et des traitements côté serveur.

Utilisation d'intelligence artificielle :

L'utilisation de ChatGPT ou de Gemini a été extrêmement bénéfique pour surmonter ces défis. En tant qu'étudiant, j'ai souvent rencontré des problèmes qui m'ont bloqué pendant un certain temps. Grâce à ces outils, j'ai pu :

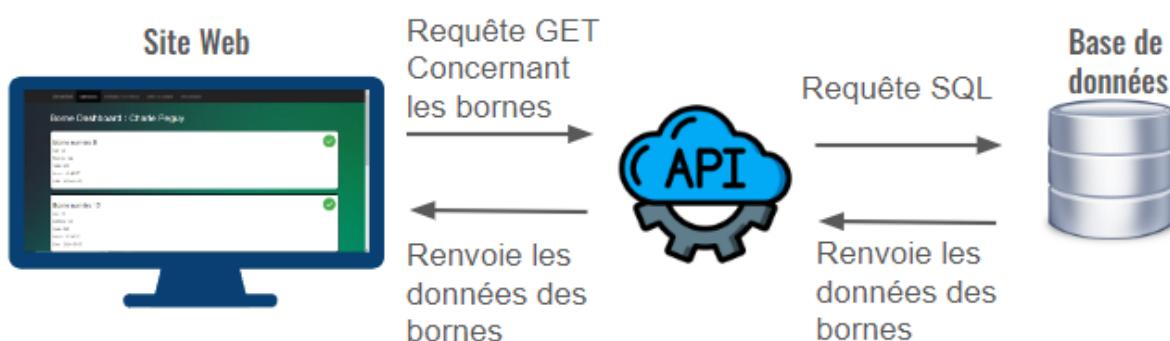
- Diagnostiquer et corriger les erreurs : ChatGPT et Gemini m'ont aidé à identifier les erreurs de syntaxe et de logique dans mon code, en fournissant des solutions claires et des explications détaillées.
- Comprendre les bonnes pratiques : J'ai appris les bonnes pratiques de développement, comme la gestion des erreurs, l'optimisation des requêtes SQL et la validation des données.
- Améliorer mes compétences : En m'aistant à résoudre les problèmes par moi-même, ces intelligences artificielles ont été des outils d'apprentissage continu, me permettant d'acquérir de nouvelles compétences et de renforcer ma confiance en tant que développeur.

Parfois, même avant de tester mon nouveau code, je vérifiais toujours le code par moi-même et réussissais à corriger les erreurs. Grâce à l'aide de ChatGPT et de Gemini, j'ai pu avancer dans mon projet, corriger mes erreurs plus rapidement et apprendre des techniques importantes pour le développement de bases de données et d'APIs. Cela m'a permis de réaliser ma partie du projet avec succès et d'intégrer les meilleures pratiques dans mon travail.

III. SITE WEB - De Almeida Toni (Étudiant 2)

3.1) Synoptique du site web

Grâce à ce schéma, nous comprenons mieux comment un site web interagit avec une base de données via une API. Lorsqu'une action est déclenchée sur le site web, celui-ci envoie une requête à l'API. Voici un exemple d'utilisation de l'API pour le site web, ici le site web fait une requête GET à l'API puis l'API fait une requête SQL pour contacter la base de données qui envoie les données demandées. Une fois ces informations obtenues, l'API les renvoie au site web, permettant ainsi aux utilisateurs de recevoir des réponses instantanément.



3.2) Résumé des tâches réalisé par les utilisateurs

	<i>Responsable Technique (RT)</i>	<i>Responsable des agents (RA)</i>	<i>Agent</i>
<i>Création des comptes RA</i>	x		
<i>Création comptes Agent</i>	x	x	
<i>Affectation des missions aux agents</i>	x	x	
<i>Consultation de l'état des bornes</i>	x	x	x

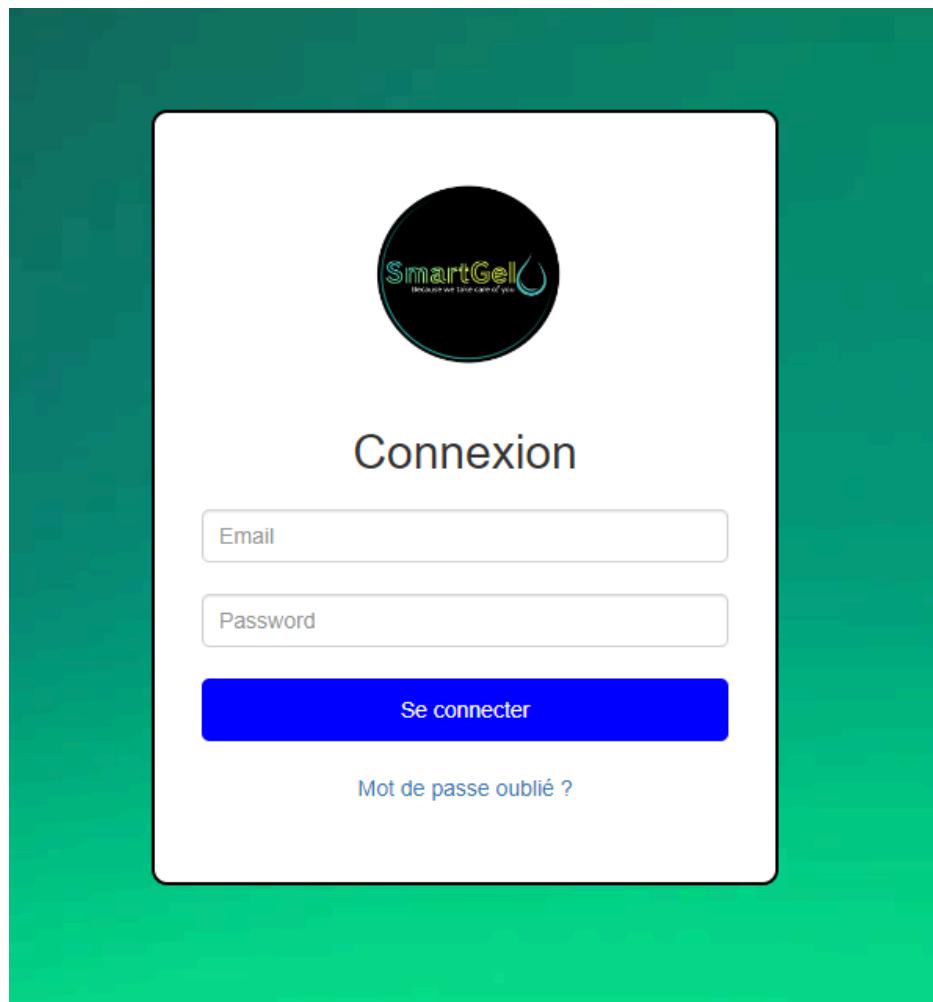
Le **Responsable Technique** dispose de plusieurs fonctionnalités essentielles. Il peut créer des comptes pour les Responsables Agents et les Agents, affecter des missions à ces derniers et consulter l'état des bornes dans tous ses établissements.

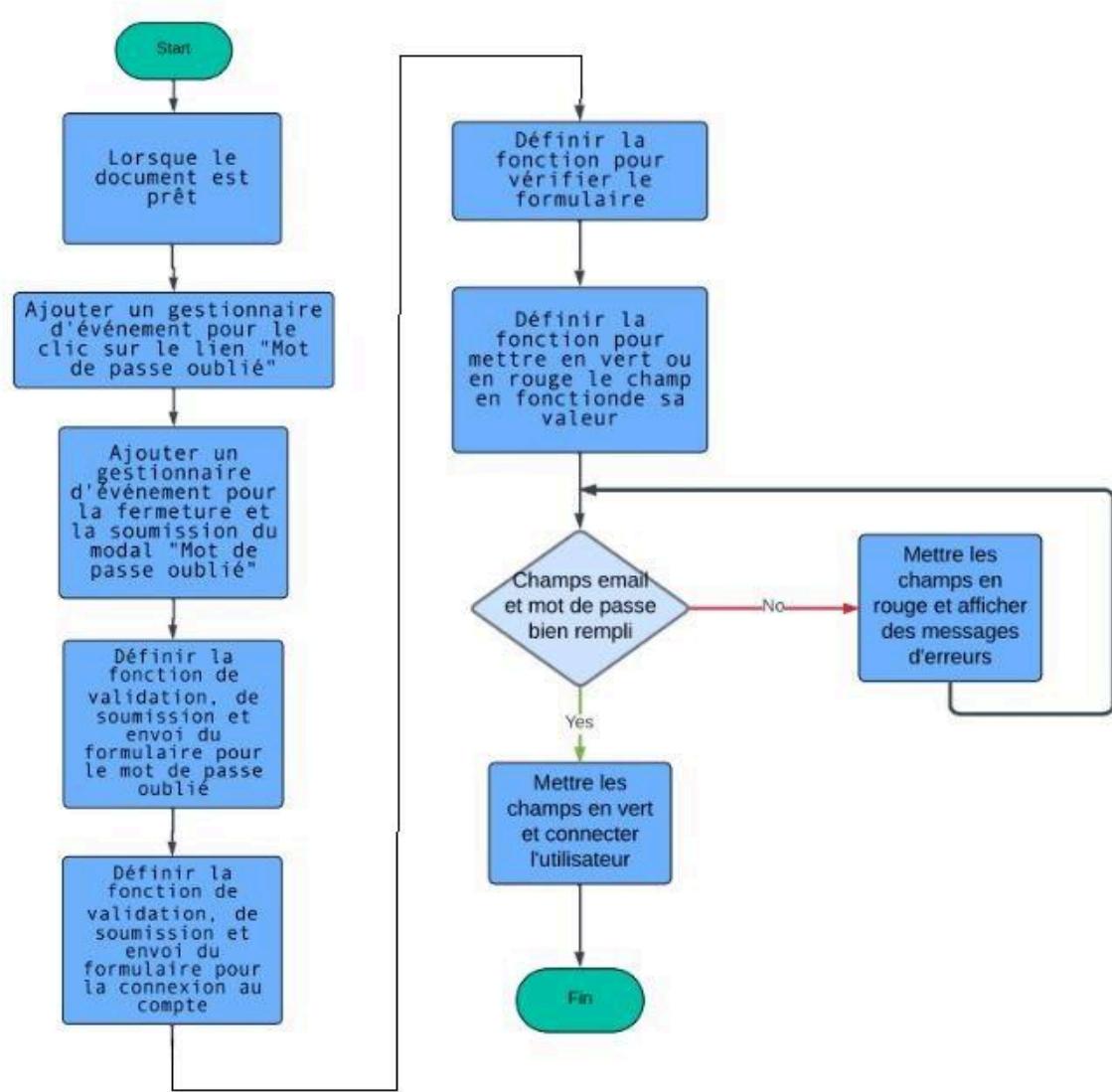
Le **Responsable Agent** a des capacités similaires mais avec un champ d'action plus restreint. Il peut créer des comptes pour les Agents, leur assigner des missions et vérifier l'état des bornes dans son propre établissement.

Quant à **l'Agent**, son rôle est centré sur l'exécution. Il peut visualiser les missions qui lui ont été attribuées.

3.3) Site SmartGel

Page de connexion :





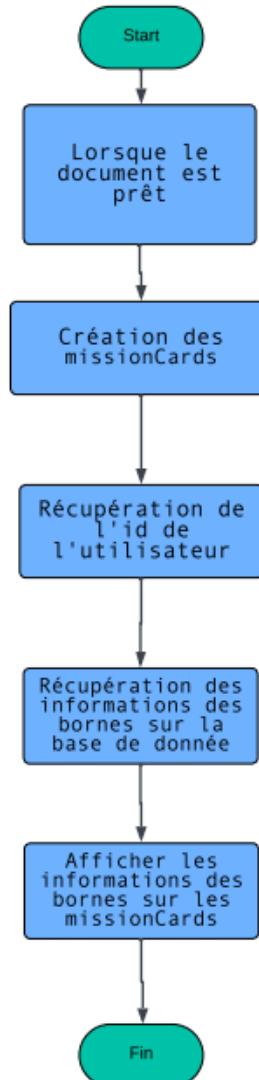
Cette page de connexion utilise une API dédiée pour gérer les profils de rôle des utilisateurs. Les trois profils de rôle concernés sont : Responsable Technique, Responsable Agent, et Agent. Cette gestion centralisée permet une redirection automatique des utilisateurs vers les pages correspondantes en fonction de leur rôle après une authentification réussie.

Page de visualisation de mission pour l'agent :

SmartGel Missions Déconnexion

Missions :

Type de mission: Niveau de batterie ID Borne: 3 Heure: 09:09:59 Date: 2024-03-27	
Type de mission: Niveau de Batterie ID Borne: 4 Heure: 17:05:50 Date: 2024-04-29	
Type de mission: Niveau de gel ID Borne: 5 Heure: 17:06:54 Date: 2024-04-29	



La page de visualisation des missions constitue un outil essentiel pour les utilisateurs connectés, leur permettant d'accéder aux détails des missions qui leur sont assignées. Cette page tire parti d'une API pour récupérer des informations pertinentes sur les bornes liées à ces missions à partir d'une base de données centralisée.

Page de Dashboard :

Version pour Responsable Agent :

The screenshot shows a dashboard interface for a responsible agent. At the top, there is a navigation bar with links: SmartGel, Dashboard, Affectation de mission, Créer un compte, and Déconnexion. The main title is "Borne Dashboard : Charle Peguy". Below the title, there are two mission cards. Each card contains the following information:

Borne numéro 8

- Gel : 60
- Batterie : 84
- Salle: B02
- Heure : 16:46:37
- Date : 2024-05-13

Borne numéro 15

- Gel : 100
- Batterie : 50
- Salle: B65
- Heure : 09:31:53
- Date : 2024-05-22

Each card has a green circular icon with a checkmark in the top right corner.

Version pour Responsable Technique :

The screenshot shows a dashboard interface for a responsible technician. At the top, there is a navigation bar with links: SmartGel, Dashboard, Affectation de mission, Créer un compte, Details des Bornes, and Déconnexion. The main title is "Borne Dashboard :". Below the title, there are two mission cards. Each card contains the following information:

Jean Rostand

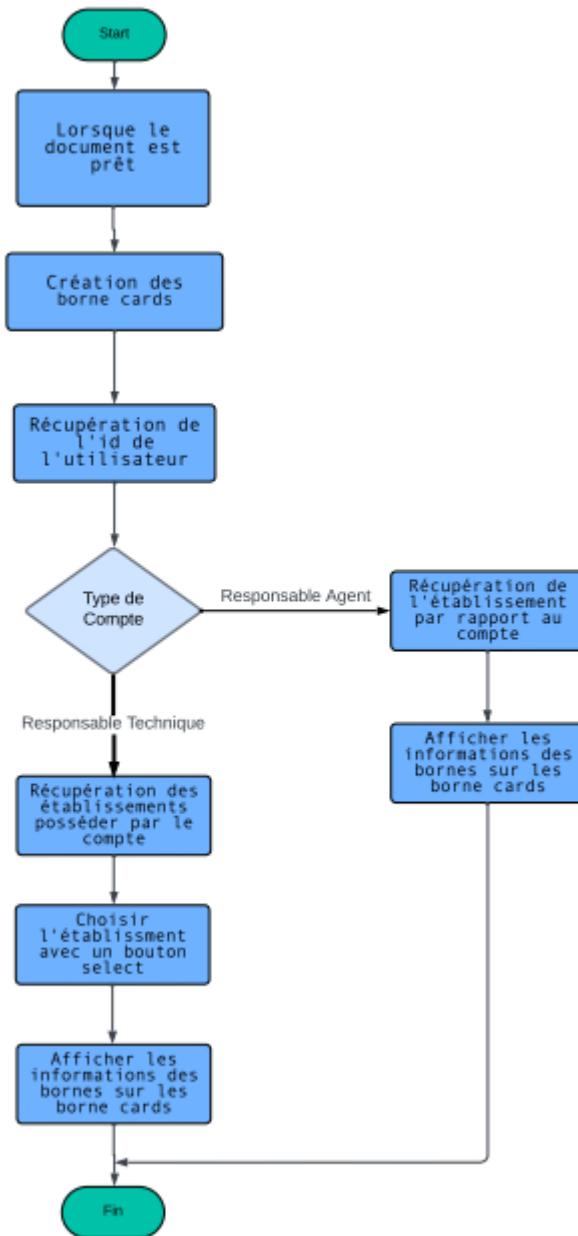
Borne numéro 1

- Gel : 0
- Batterie : 0
- Salle: GR-01
- Heure : 16:53:09
- Date : 2024-05-21

Borne numéro 2

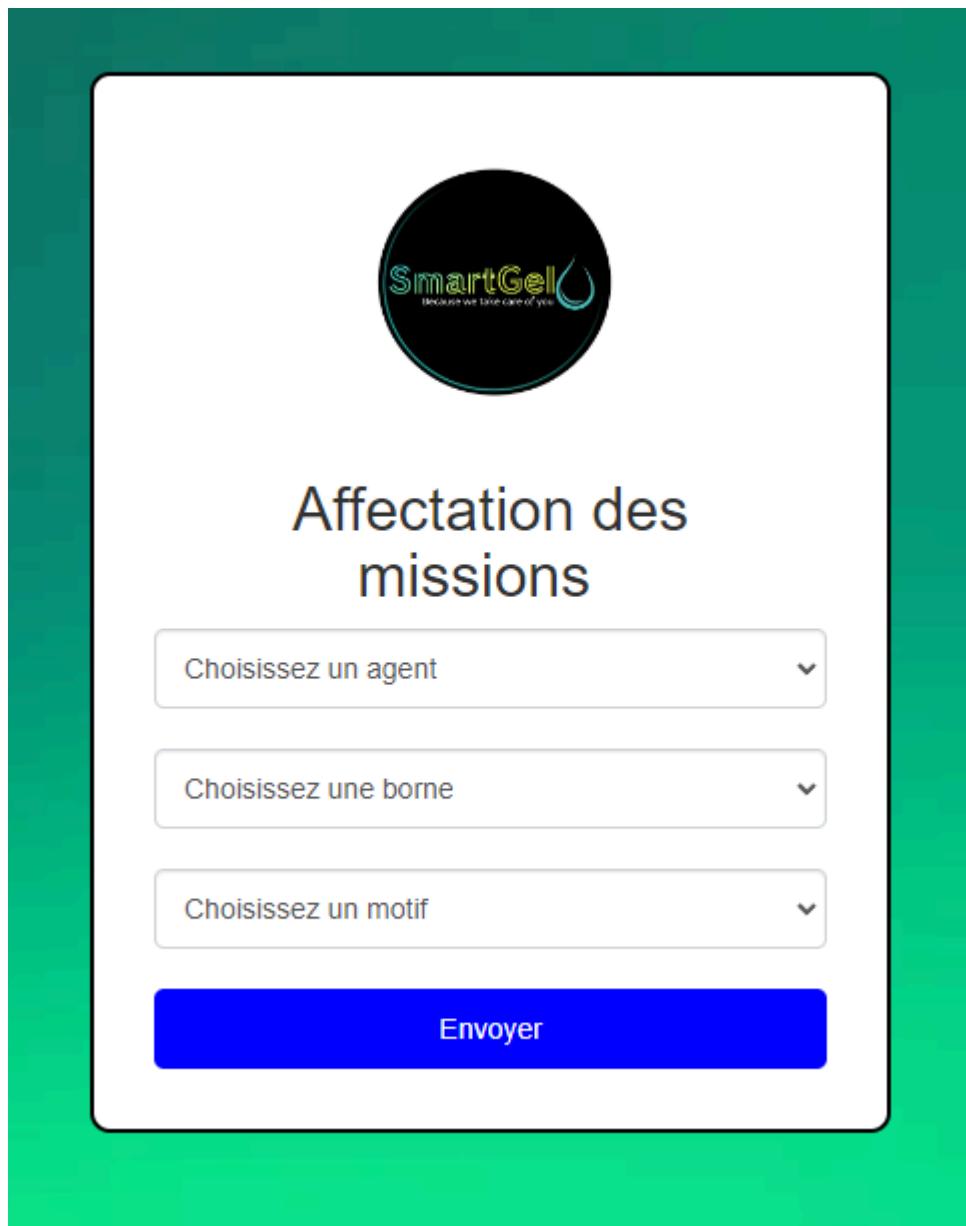
- Gel : 0
- Batterie : 0
- Salle: GR-11
- Heure : 16:53:09
- Date : 2024-05-21

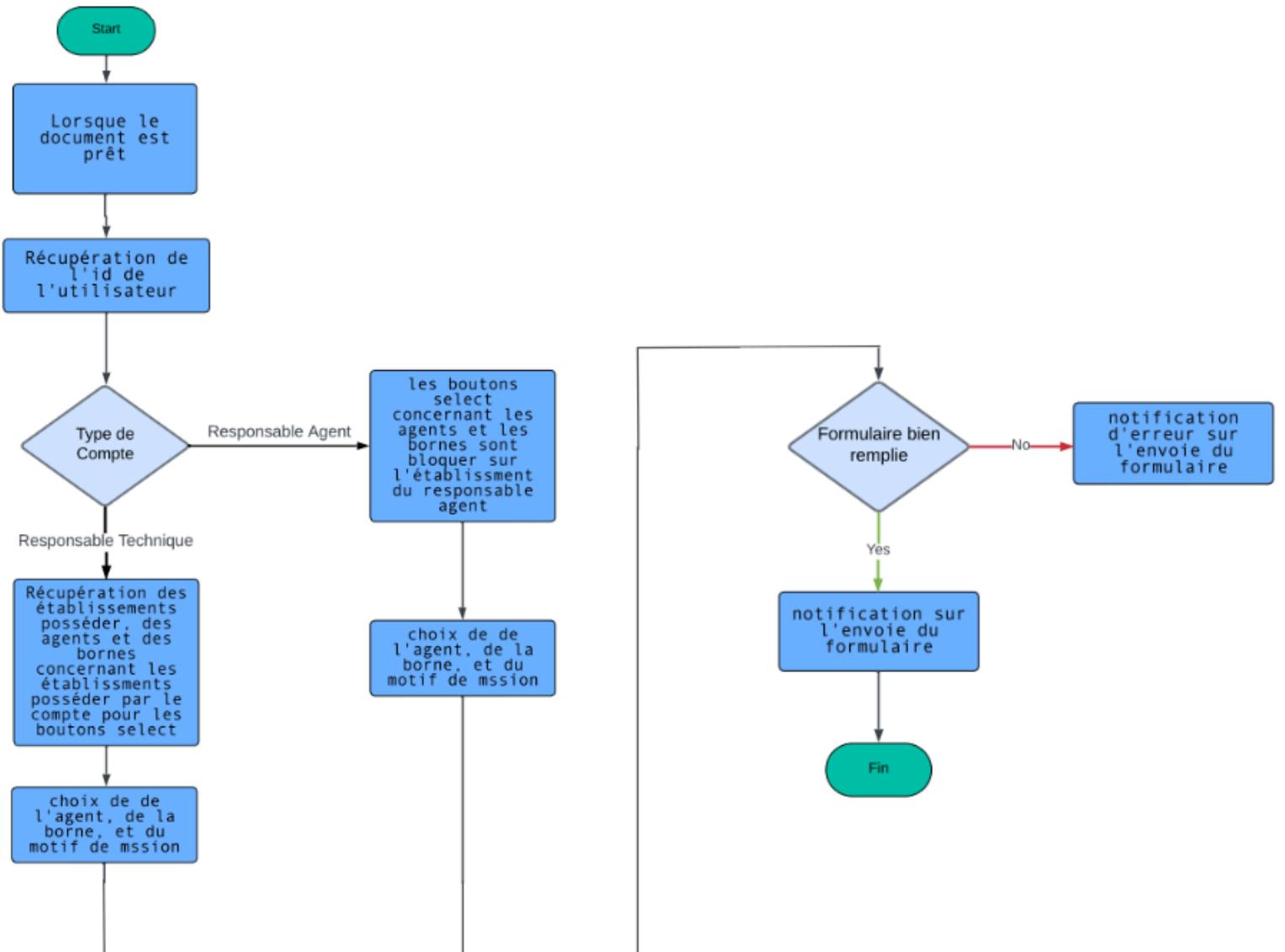
Each card has a red circular icon with a white X in the top right corner.



Le Dashboard des informations sur les bornes constitue une interface centrale pour les utilisateurs connectés, offrant une vue globale et détaillée des données relatives aux bornes. Cette page utilise une API pour récupérer les informations spécifiques aux bornes en fonction de l'utilisateur connecté, tirées d'une base de données centralisée.

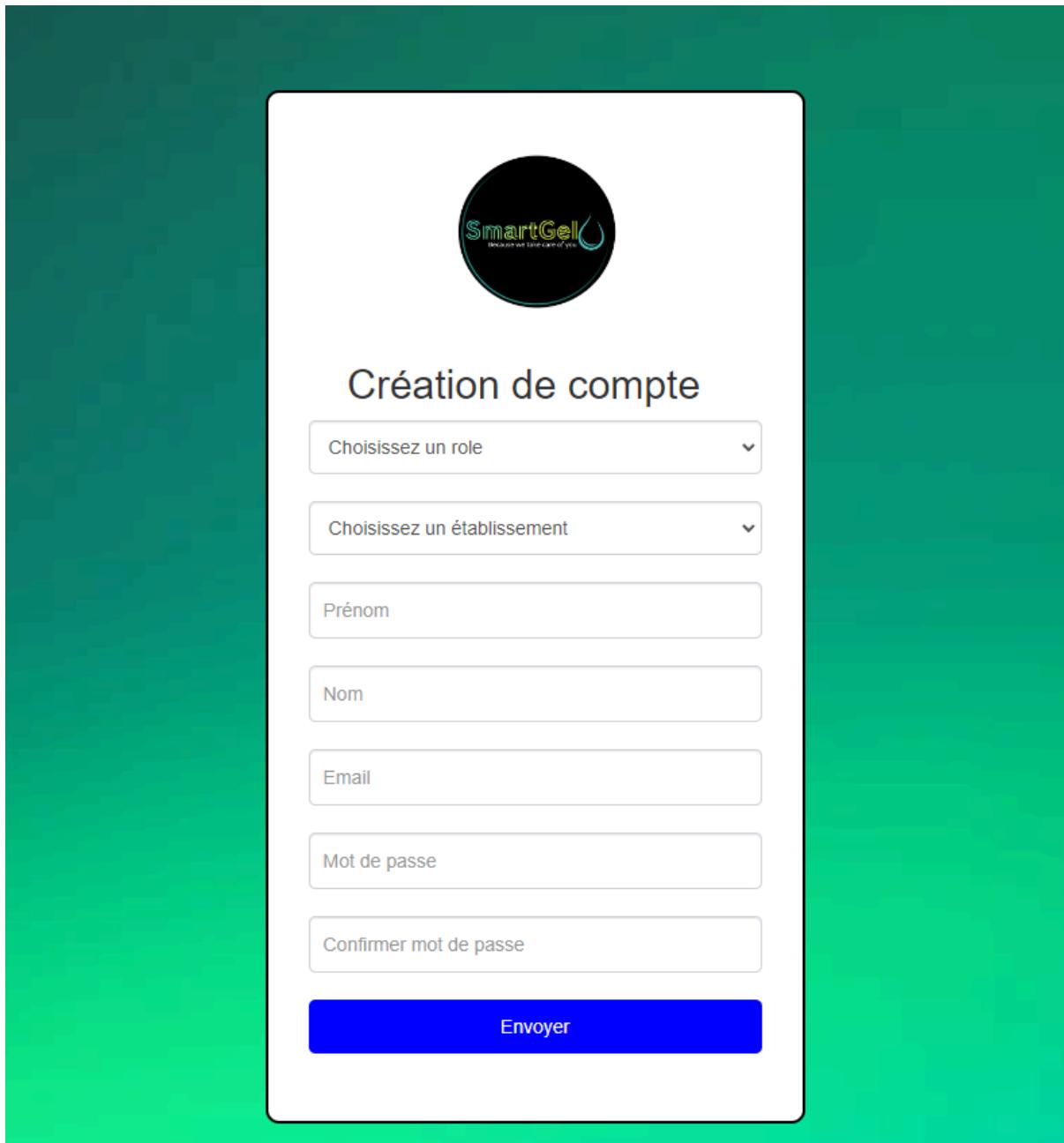
Page d'affectation des missions :

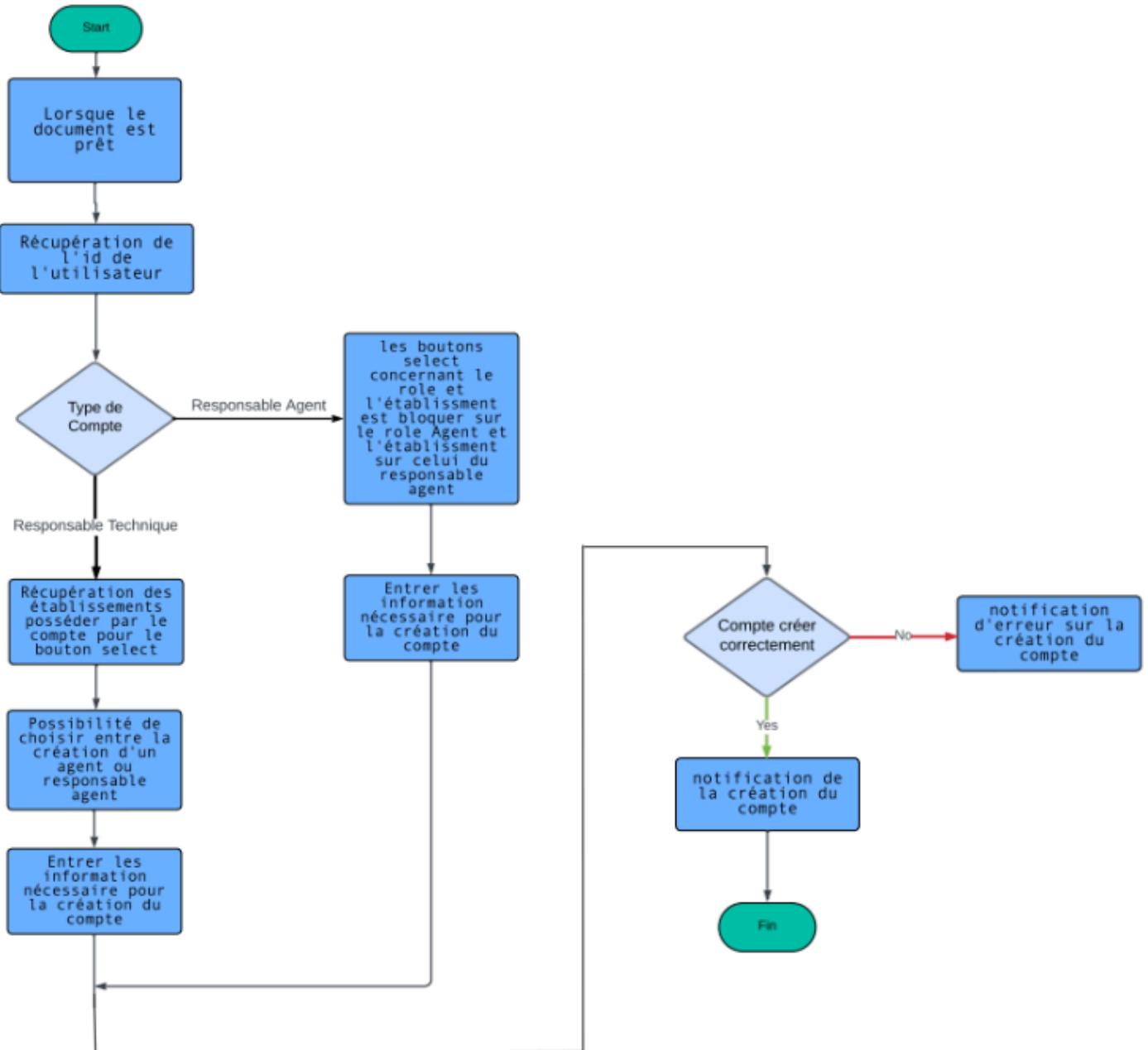




La page d'affectation de mission comporte trois boutons de sélection : agent, établissement et motif. Les boutons de sélection d'agent et de borne fonctionnent en fonction de l'établissement auquel appartient l'utilisateur connecté. Pour un responsable technique, ils permettent de choisir parmi les établissements qu'il supervise. Cette fonctionnalité est rendue possible grâce à une API qui interagit avec la base de données. Le bouton de sélection de motif permet de spécifier la raison de la mission, assurant ainsi une gestion précise et contextualisée des affectations.

Page de création des comptes :





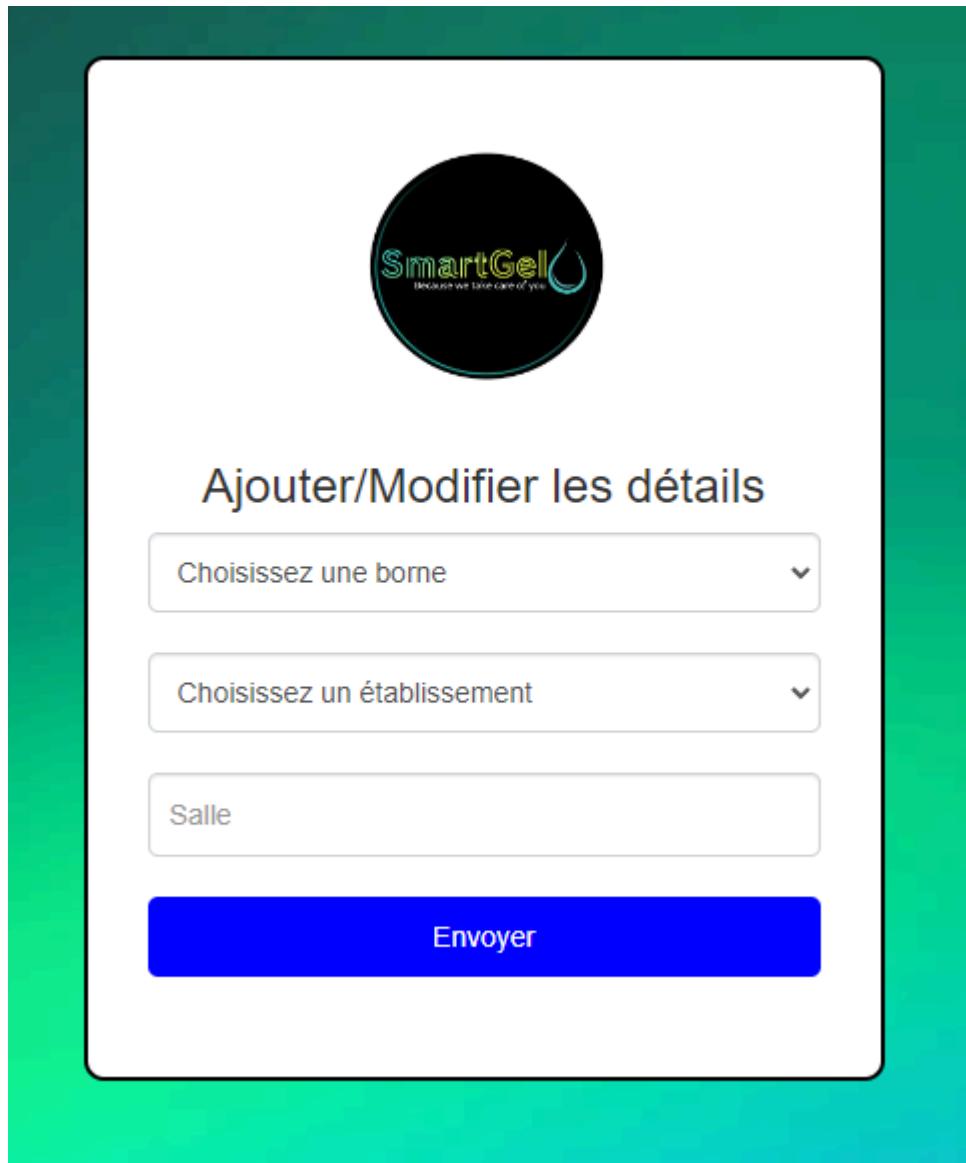
La page de création de compte comporte des champs pour le prénom, le nom, le mail, le mot de passe et la vérification de mot de passe, ainsi que deux boutons de sélection : rôle et établissement. Les boutons de sélection fonctionnent en fonction du rôle de l'utilisateur connecté.

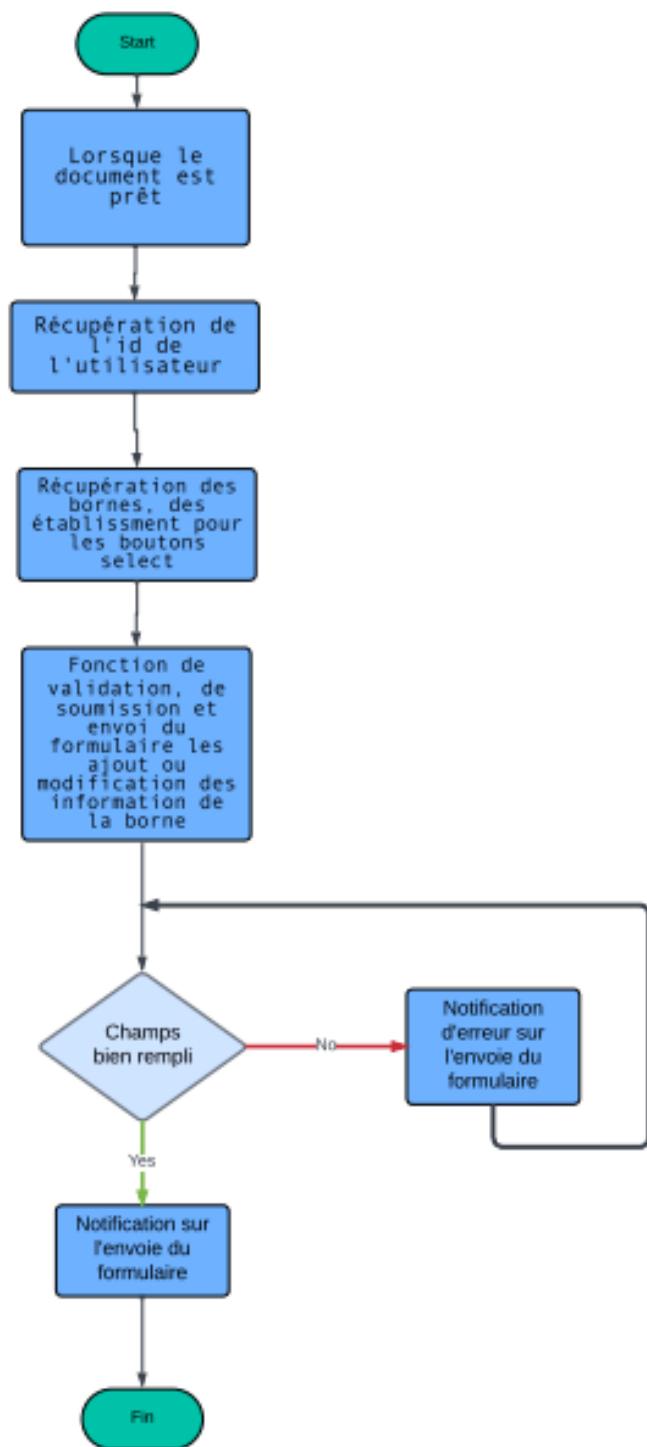
Pour un responsable technique : Les établissements qu'il supervise sont affichés dans le bouton de sélection. Il peut créer des agents et des responsables agents.

Pour un responsable agent : Seuls les agents pour son propre établissement peuvent être créés.

Cette fonctionnalité est rendue possible grâce à une API qui interagit avec la base de données, assurant une gestion contextuelle et sécurisée des nouveaux comptes.

Page d'ajout ou de modification de détails pour les bornes:





La page d'ajout ou de modification de détails pour les bornes permet au responsable technique de gérer les informations des bornes. Elle comprend deux boutons de sélection : un pour les bornes et un pour les établissements, ainsi qu'un champ pour préciser la salle.

Après l'envoi du formulaire, les informations sont transmises à la base de données. L'accès à cette page est réservé exclusivement au responsable technique. Cette fonctionnalité est rendue possible grâce à une API qui interagit avec la base de données, garantissant une gestion sécurisée et efficace des bornes.

3.4) Défis rencontrés

1. Problème de Communication entre AJAX et PHP

Lors du développement du site web, nous avons rencontré des problèmes de communication entre les requêtes AJAX et les scripts PHP. Cela se traduisait par des réponses incorrectes ou des erreurs de connexion, perturbant ainsi le fonctionnement des fonctionnalités dépendant de la base de données.

2. Problème de JavaScript pour la Validation Visuelle des Champs et Boutons Select

Défi : Nous avons eu des difficultés à utiliser JavaScript pour mettre en évidence les champs et les boutons de sélection (select) afin d'indiquer visuellement s'ils étaient valides ou non.

3. Problème de Récupération et de Transmission du Compte Utilisateur

Défi : La transmission des informations de compte de la page de connexion aux autres pages de l'application s'est avérée problématique, ce qui compliquait la gestion de l'état utilisateur.

4. Problème de Filtrage des Données en Fonction de l'Utilisateur Connecté

Défi : Assurer que chaque utilisateur puisse voir uniquement les informations qui le concernent a été un défi majeur, notamment pour les pages réservées aux responsables techniques ou agents.

IV. OBJET CONNECTÉ IOT - Bouaksim Adam (Etudiant 3)

Dans le cadre de ce projet, l'objectif est de moderniser une borne de gel hydroalcoolique en y intégrant un système d'acquisition et de transmission des données. Cette automatisation vise à garantir une gestion optimale des niveaux de gel et de batterie, en envoyant ces informations en temps réel à un serveur d'API sous format JSON.

Dans cette partie il y'a trois principaux buts :

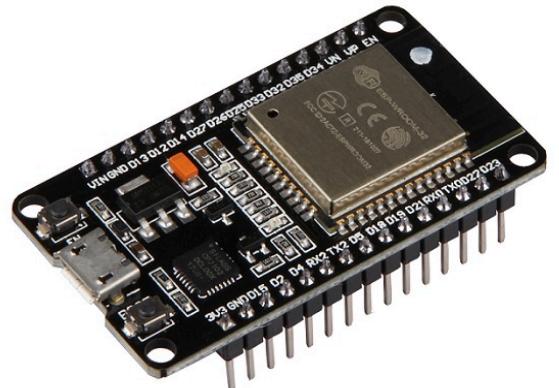
- Acquérir le niveau de gel hydroalcoolique de la borne
- Acquérir le niveau de tension de la batterie de la borne de gel
- Transmettre au serveur d'API ces informations au format JSON.

Commençons d'abord par présenter le matériel que j'ai utilisé :

ESP 32

ESP32 est un microcontrôleur avec des modules Wifi et Bluetooth intégrés.

Très simple d'utilisation, il est petit, léger et possède une capacité de mémoire et de calcul supérieure aux Arduino.



Ultrasonic Distance Sensor - HC-SR04

Le capteur HC-SR04 est un capteur à ultrasons permettant de mesurer une distance par ultrason.

Il émet un ultrason qui se propage dans l'air et s'il y a un objet ou un obstacle sur son chemin, il rebondira vers le module.



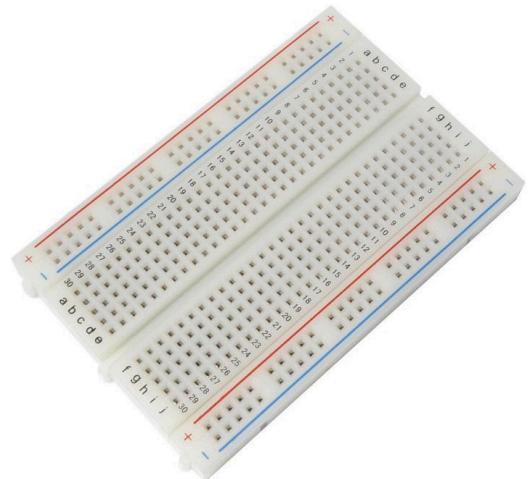
Piles

Les piles nous permettront d'alimenter la borne, ainsi que les différents capteurs.



Breadboard

Une breadboard est une plaque de prototypage utilisée pour construire des circuits électroniques facilement. Elle permet d'insérer et de connecter des composants sans soudure, facilitant les modifications.



Fils de Connexion

Les fils de connexion, ou jumpers, sont utilisés pour relier les composants sur une breadboard. Ils existent en différentes configurations (mâle-mâle, mâle-femelle, femelle-femelle) et facilitent les connexions et l'organisation des circuits.



Résistances

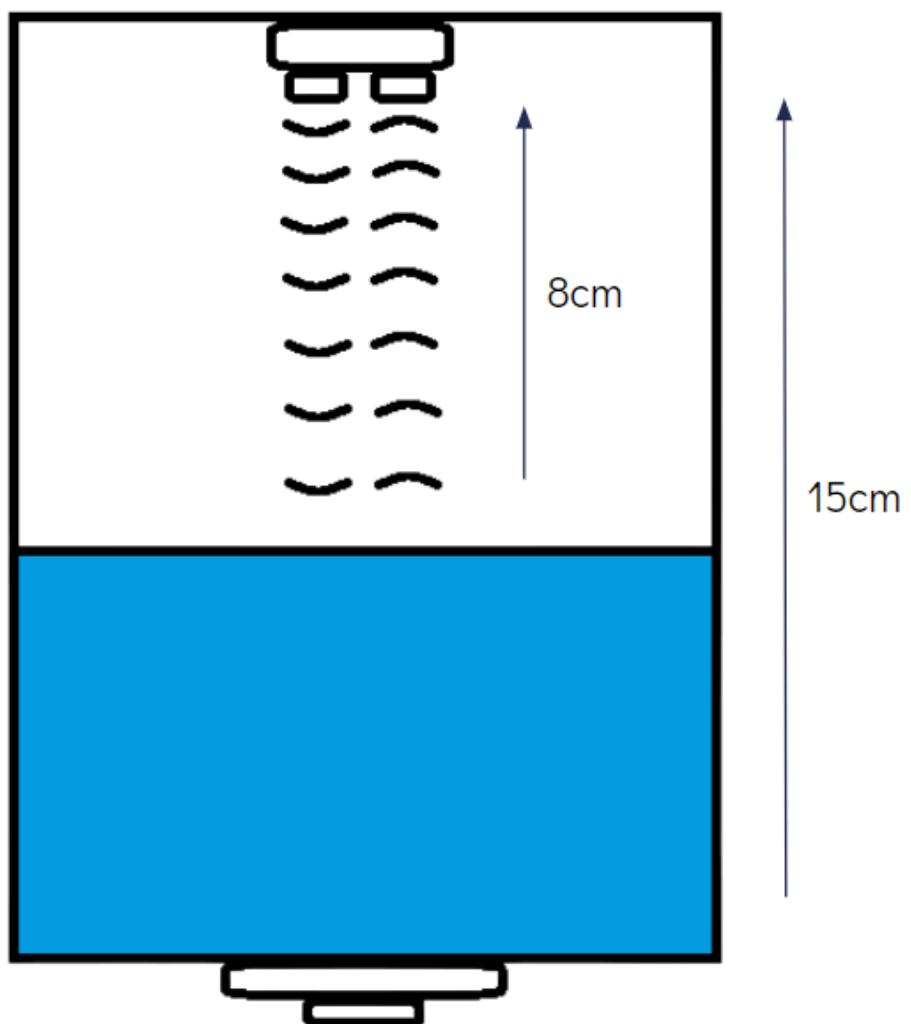
Les résistances sont des composants électroniques qui limitent le courant électrique dans un circuit. Elles sont essentielles pour contrôler la tension et protéger les composants sensibles.



Acquérir le niveau de Gel :

Pour acquérir le niveau de gel de la borne on utilise le capteur ultrason. Nous avons eu l'idée de placer le capteur en haut de la borne. En fonction de la distance entre le gel et le capteur, il est possible d'estimer la quantité de gel dans la borne.

Voici un schéma pour mieux comprendre :



Dans le dessin, le capteur émet un ultrason qui est réfléchi par le gel. Si la distance mesurée aurait été de 15 cm cela aurait signifié qu'il n'y aurait plus de gel.

Ici, le capteur mesure une distance de 8 cm ce qui correspond à environ la moitié de la borne de gel. Donc avec cette distance on peut déduire que la borne est remplie à 46%.

Le calcul effectué pour trouver ce pourcentage est le suivant :

$$\text{Pourcentage de remplissage} = 100 - \left\lfloor \frac{\text{distance mesurée} \times 100}{\text{distance maximale}} \right\rfloor$$

$$\text{Pourcentage de remplissage} = 100 - \left\lfloor \frac{8 \times 100}{15} \right\rfloor$$

$$100 - \frac{8 \times 100}{15} \approx 46\%$$

Le code qui permet à l'ESP32 ce pourcentage est le suivant :

```
int mesurerGel() {
    digitalWrite(trig_pin, LOW);
    delayMicroseconds(2);
    digitalWrite(trig_pin, HIGH);
    delayMicroseconds(TRIG_PULSE_DURATION_US);
    digitalWrite(trig_pin, LOW);
    unsigned long ultrason_duration = pulseIn(echo_pin, HIGH);
    float distance_cm = (ultrason_duration * SOUND_SPEED * 0.0001) / 2;
    float gel_height = MAX_HEIGHT - distance_cm;
    if (gel_height < 0) {
        gel_height = 0;
    }
    if (gel_height > MAX_GEL_HEIGHT) {
        gel_height = MAX_GEL_HEIGHT;
    }
    return round((gel_height / MAX_GEL_HEIGHT) * 100);
}
```

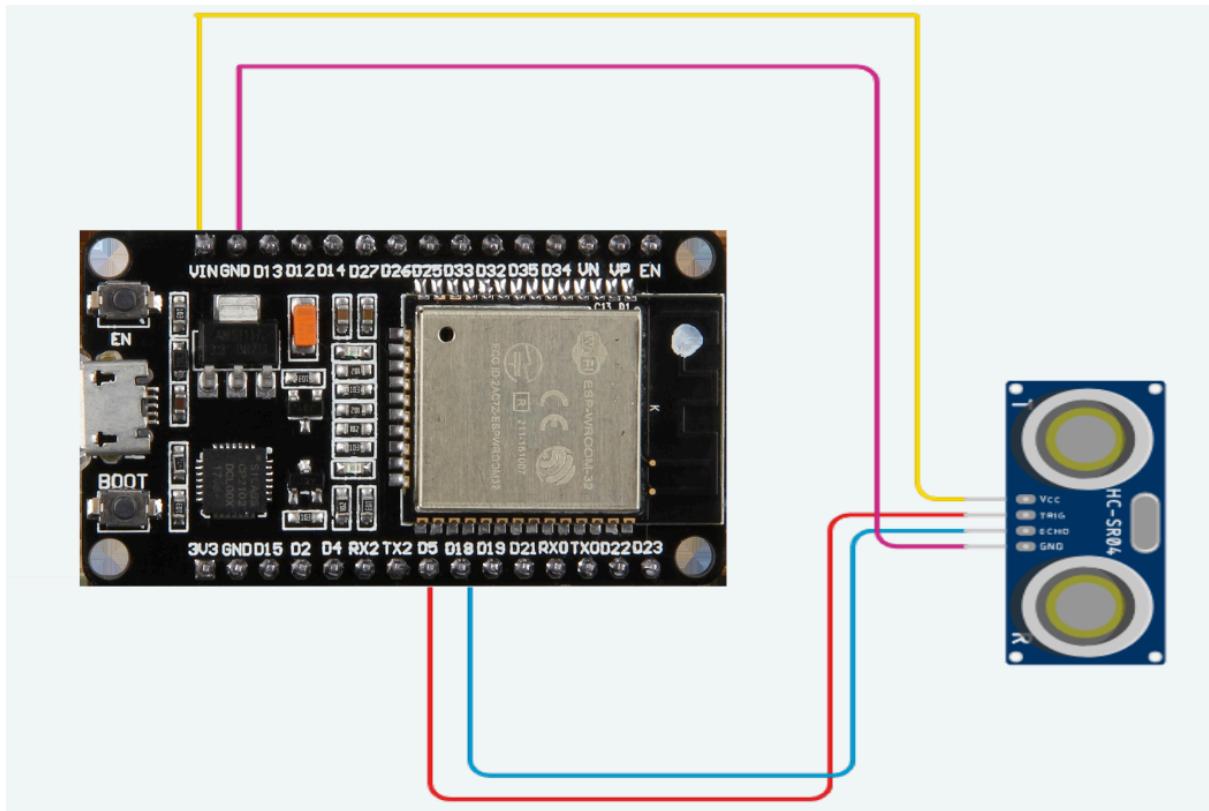
On commence d'abord par configurer le capteur ultrasonique pour envoyer une impulsion.

Le résultat est divisé par 2 car le temps mesuré est celui de l'aller-retour de l'impulsion (aller et retour).

`gel_height` est déterminé en soustrayant la distance mesurée de la hauteur maximale (`MAX_HEIGHT`).

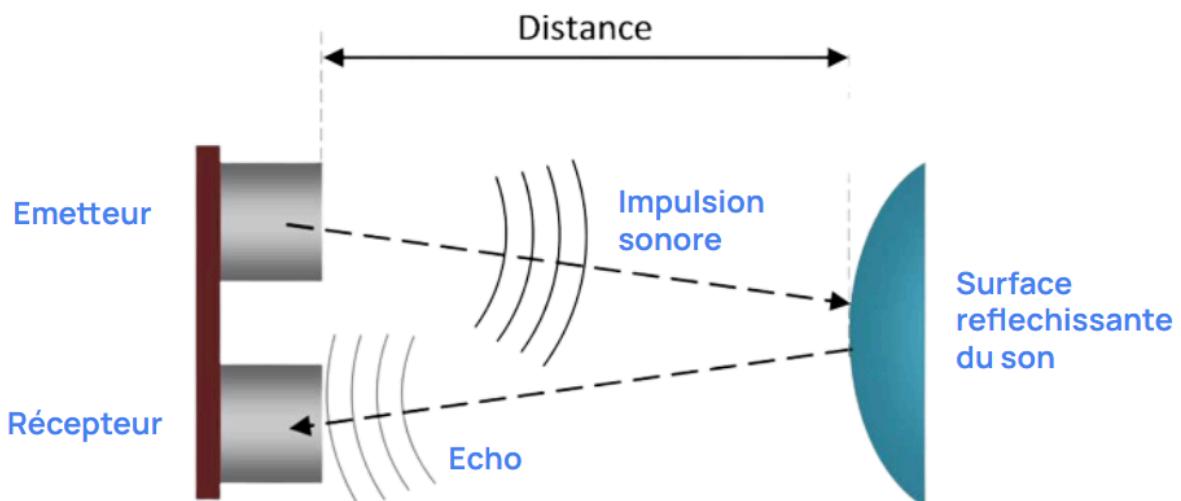
Enfin, le pourcentage de remplissage est calculé en divisant `gel_height` par la hauteur maximale de gel (`MAX_GEL_HEIGHT`) et en multipliant par 100.

Voilà comment se présente les branchements du capteur :



Le capteur ultrason HC-SR04 est un dispositif couramment utilisé pour mesurer des distances en utilisant des ondes sonores. Voici une explication de son fonctionnement et comment l'intégrer dans la partie physique de votre projet pour mesurer le niveau de gel dans un récipient.

Fonctionnement du HC-SR04



Le capteur HC-SR04 dispose de deux transducteurs : un émetteur (trigger) et un récepteur (echo). Lorsqu'un signal électrique est envoyé au pin "trigger" (trig), le capteur émet une onde ultrasonique à 40 kHz.

L'onde ultrasonique voyage à travers l'air et se réfléchit sur un objet (dans ce cas, la surface du gel)

L'onde réfléchie revient vers le capteur et est captée par le récepteur (echo).

La formule utilisée pour calculer la distance est la suivante :

La formule utilisée pour calculer la distance avec un capteur ultrason HC-SR04 est dérivée de la relation fondamentale entre la distance, la vitesse et le temps. La distance totale parcourue par l'onde sonore est calculée, puis divisée par deux pour obtenir la distance unidirectionnelle entre le capteur et l'objet.

$$\text{Distance} = \frac{\text{Temps de vol} \times \text{Vitesse du son}}{2}$$

Acquérir le niveau de Batterie :

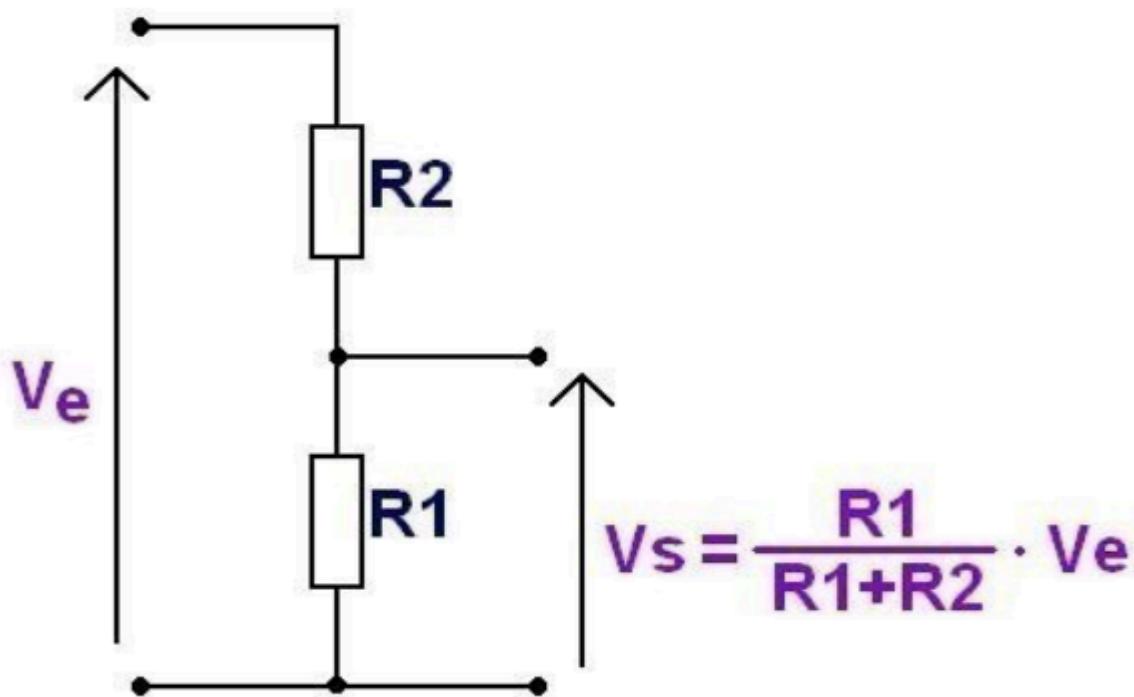
Pour acquérir le niveau de batterie il faut bien comprendre les caractéristique de l'ESP32.

Chaque branches de l'ESP32 ne peut supporter une tension supérieure à 3.3 V hors notre batterie délivre du 6V.

L'idée est de transformer notre tension de 6V en 3.3V

Il faut donc trouver une solution qui nous permettrait d'abaisser cette tension, pour ce faire on utilisera un pont diviseur.

Un pont diviseur de tension est un montage simple en électronique qui permet de diviser une tension en utilisant deux résistances. Voici un schéma pour mieux comprendre.



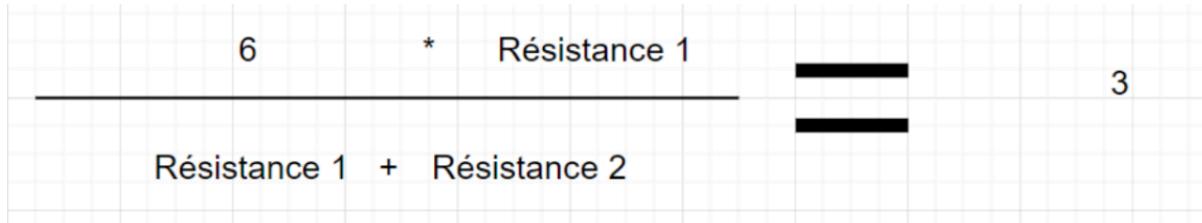
Pour trouver V_s la tension de sortie, il suffit de multiplier la valeur d'entrée V_e par la Résistance 1 puis de diviser par la Résistance 1 additionnée à la Résistance 2

Ce qui donne :

$$\frac{\text{Tension d'entrée} * \text{Résistance 1}}{\text{Résistance 1} + \text{Résistance 2}} = \text{Tension de sortie}$$

Pour ce pont diviseur Résistance 1 vaudra $100 \text{ k}\Omega$ et Résistance 2 vaudra aussi $100 \text{ k}\Omega$.

En adaptant à notre situation :



Cela donne :

$$\frac{6 * 100}{100 + 100} = 3$$

C'est exactement le même principe si on a une valeur d'entrée différente, par exemple 12V.

Résistance 1 vaudra alors $149 \text{ k}\Omega$.
Résistance 2 vaudra alors $55 \text{ k}\Omega$.

Une fois que nous avons une tension acceptable par l'ESP32, on peut passer au code Arduino.

Le code qui permet à l'ESP32 de calculer le pourcentage de batterie :

Tout d'abord on commence par définir nos variables :

```
const int BATTERYPIN = 36;  
const float TensionMin = 1.5;  
const float TensionMax = 3.3;
```

BATTERYPIN est la branche qui reçoit la tension qu'on vient de calculer.

TensionMin correspond à la tension minimale que la batterie devra nous fournir pour le bon fonctionnement de l'ESP.

TensionMax est la tension maximum que L'ESP peut supporter.

```
int getBattery() {  
    float bat = analogRead(BATTERYPIN);  
    int minValue = (4095 * TensionMin) / 3;  
    int maxValue = (4095 * TensionMax) / 3;  
    bat = ((bat - minValue) / (maxValue - minValue)) * 100;  
    return int(bat);  
}
```

Pour faire simple nous prenons notre tension minimal et maximal, nous la multiplions par 4095 (car L'ESP est codé sur 12 bit) puis on divise le résultat par 3.

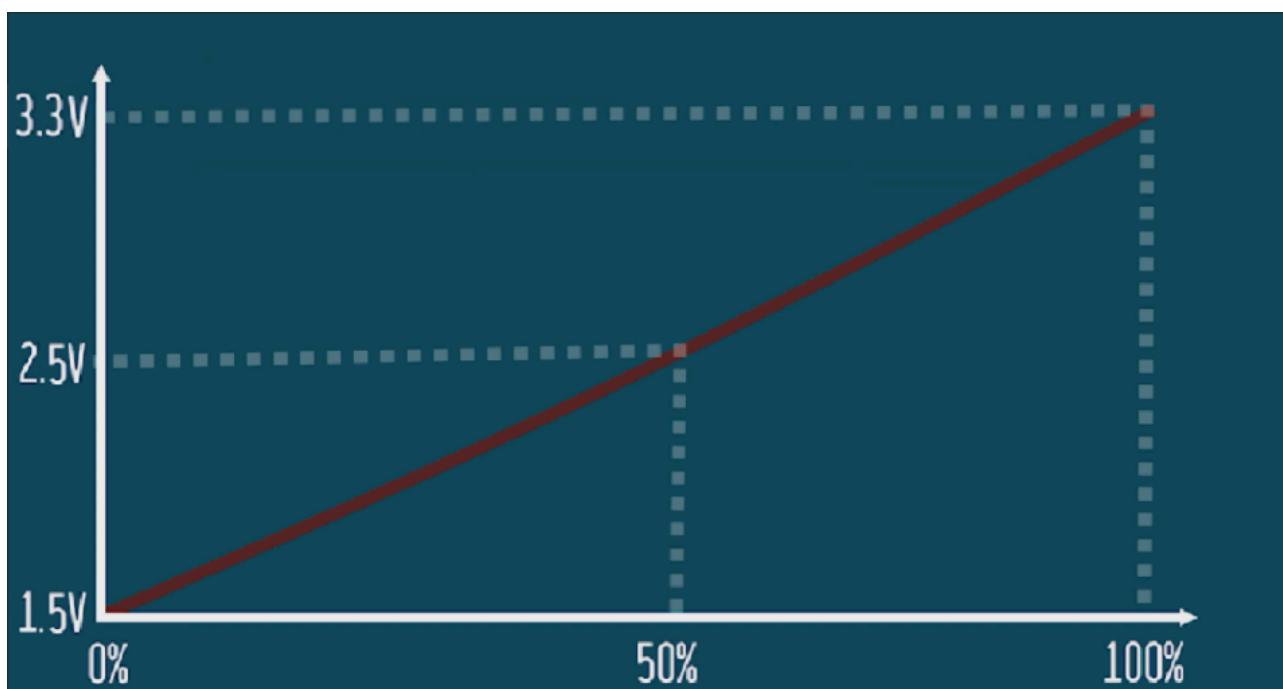
On prend ensuite la tension délivrée par la batterie (celle après le pont diviseur). Pour mieux comprendre, prenons l'exemple d'une tension délivrée de 2.8V.

Nous prenons notre 2.8V que l'on soustrait par notre tension minimal qui sera ensuite divisé par la différence entre la tension maximal et minimal.

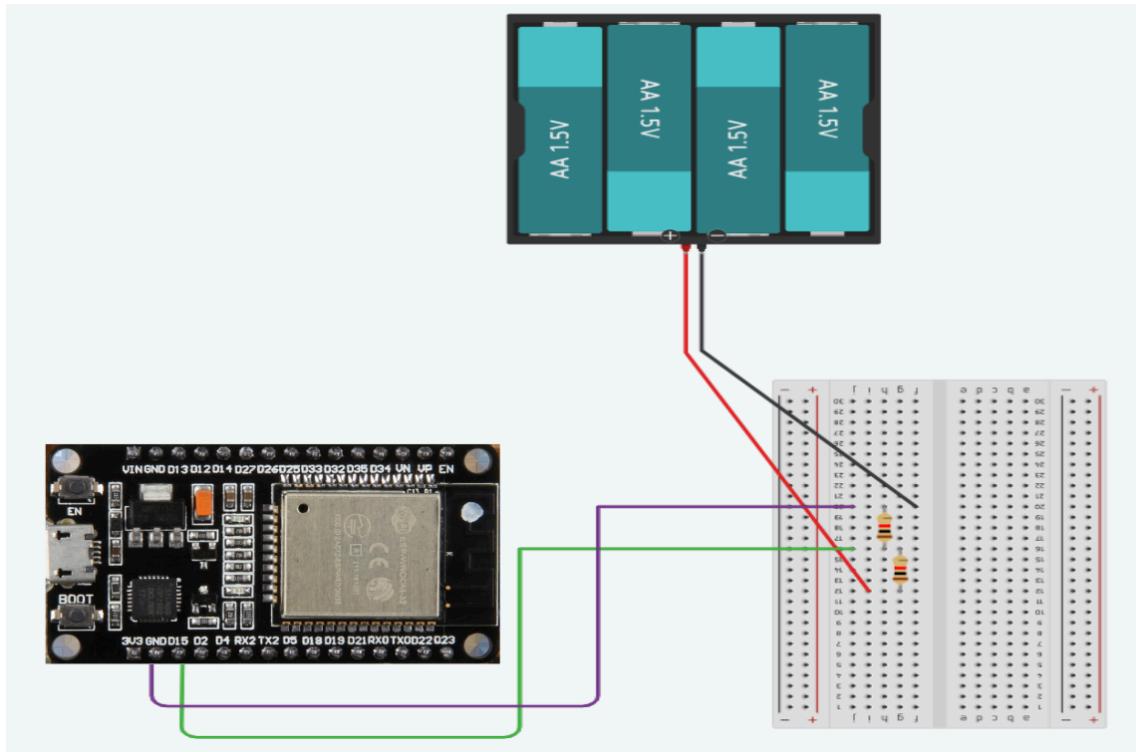
Pour finir on multipliera tout cela par 100 pour avoir le pourcentage.

Nous obtenons alors notre pourcentage de batterie pour une tension délivrée de 2.8V cela correspond à 84% de batterie.

Pour mieux comprendre voici un graphique qui montre comment fonctionne le pourcentage.



Voilà comment se présente les branchements :



Transmettre au serveur d'API ces informations au format JSON :

Maintenant que nous avons réussi à mesurer le niveau de gel et de batterie de la borne, il ne nous reste plus qu'à envoyer ces informations.

Pour ce faire, on doit d'abord connecter l'ESP32 au WIFI.

```
void setup() {
    Serial.begin(115200);
    pinMode(trig_pin, OUTPUT);
    pinMode(echo_pin, INPUT);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.println("Connexion au réseau WiFi...");
    }

    Serial.println("Connecté au réseau WiFi !");
}
```

Il suffit juste de déclarer le nom du réseau WIFI dans la variable `ssid` et le mot de passe dans `password`.

N'oublions pas qu'il faut aussi inclure la bibliothèque WiFi.h de l'ESP32.

```
#include <WiFi.h>
```

Une fois connecté au WIFI on peut enfin envoyer ces données à l'API.

Pour l'envoie des données nous allons utiliser HTTP Client (avec la méthode POST).

Mais nous ne pouvons pas envoyer nos données telles qu'elles le sont actuellement, nous devons d'abord créer un JSON avec nos différentes données.

Créer un JSON en Arduino est très simple, il faut d'abord utiliser la bibliothèque ArduinoJson.h :

```
#include <ArduinoJson.h>
```

Elle permet de créer du JSON simplement.

Voici donc la création de mon JSON :

```
DynamicJsonDocument doc(204);
doc["battery_percentage"] = batteryLevel;
doc["gel_percentage"] = gel_percentage;
doc["numeroESP"] = 1;
String postData;
serializeJson(doc, postData);
Serial.println(postData);
```



```
{
    "battery_percentage" : 84,
    "gel_percentage" : 46,
    "numeroESP" : 1,
}
```

Maintenant que le JSON est prêt , il ne reste plus qu'à l'envoyer.
Pour cela on utilise la bibliothèque `HTTPClient.h` de l'ESP32.

```
#include <HTTPClient.h>
```

Le code qui nous permet d'envoyer les données est le suivant :

```
HTTPClient http;  
http.begin(api_url);  
http.addHeader("Content-Type", "application/json");  
int httpResponseCode = http.POST(postData);  
http.end();
```

On crée un objet `HTTPClient` pour gérer la connexion HTTP.

On initie la connexion HTTP avec l'URL de l'API définit juste avant :

```
const char* api_url = "http://51.210.151.13/btssnir/projets2024/bornege12024/bornege12024/SmartGel/API/API_Borne.php";
```

Ajout de l'en-tête HTTP pour indiquer que les données envoyées sont en JSON.

Et on envoie la requête POST avec les données JSON et réceptionne un code de réponse HTTP.

Enfin, on met fin à la connexion HTTP.

Mise en mode d'économie d'énergie.

Pour conclure il nous a été demandé de réfléchir à un moyen de maximiser l'autonomie de la batterie.

Nous avons donc utilisé le Mode SLEEP de l'ESP32

L'ESP 32 a un mode qui permet l'économie d'énergie qui est le "Sleep Modes".

Pour faire simple l'alimentation de tous les périphériques numériques inutiles est coupée, tandis que la RAM reçoit juste assez d'énergie pour lui permettre de conserver ses données.

Il est très facile d'utilisation :

```
esp_sleep_enable_timer_wakeup(300000000); // 5 minutes en microsecondes  
esp_deep_sleep_start();
```

Nous avons juste à régler le minuteur qui détermine la durée de sommeil puis lancer le mode.

Une fois toutes ces parties rassemblées, lorsque le code est exécuté voici le résultat final qu'on peut visualiser grâce au moniteur série Arduino :

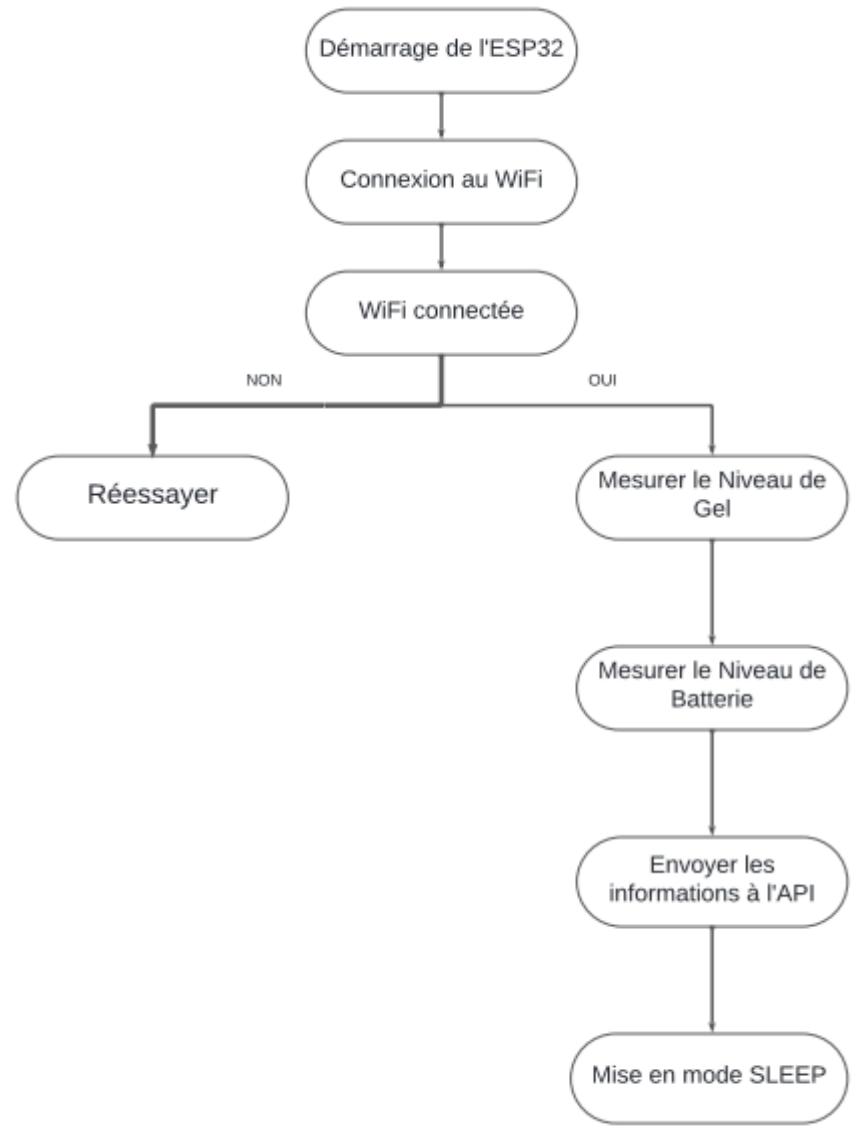
The screenshot shows the Arduino Serial Monitor window titled "COM4". The text output is as follows:

```
rst:0x5 (DEEPSLEEP_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len:13964
load:0x40080400,len:3600
entry 0x400805f0
Connexion au réseau WiFi...
Connexion au réseau WiFi...
Connexion au réseau WiFi...
Connecté au réseau WiFi !
{"battery_percentage":77,"gel_percentage":77,"numeroESP":1}
```

At the bottom of the window, there are two checkboxes: Défilement automatique and Afficher l'horodatage.

On voit le réveil de l'ESP32 ainsi que les mesures de gel et de batterie.

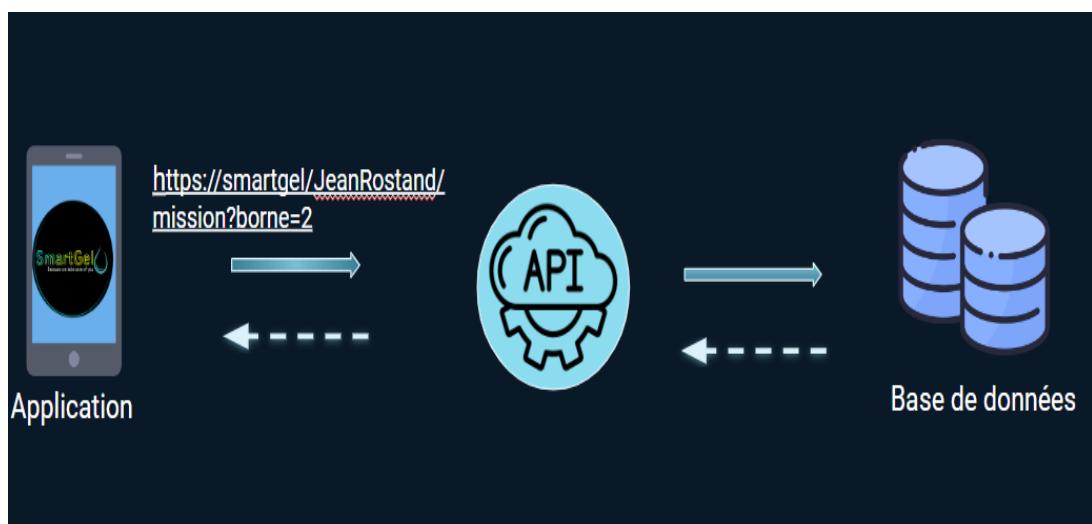
Voici pour finir un algorithme qui résume l'entièreté de la partie IOT :



V. APPLICATION ANDROID - Lozano Dally (Etudiant 4)

5.1) Synoptique de l'Application

Grâce à la synoptique, nous pouvons comprendre de manière générale la façon dont une application communique avec une base de données via une API. Lorsqu'un événement se produit dans l'application, celle-ci interroge l'API. L'API communique ensuite avec la base de données pour obtenir l'information requise. Une fois cette information récupérée, l'API la transmet de nouveau à l'application, permettant ainsi à l'utilisateur de recevoir une réponse en temps réel.



5.2) Résumé des tâches réalisées par utilisateur

Agent :

- ★ Il peut visualiser les bornes de son établissement.
- ★ Il peut visualiser ses missions à réaliser.

Responsable Agent :

- ★ Il est responsable d'un seul établissement.
- ★ Il peut surveiller toutes les bornes d'un établissement.
- ★ Il peut réaliser des missions lorsqu'il y a une borne avec un niveau faible de gel ou de batterie.

Responsable Technique :

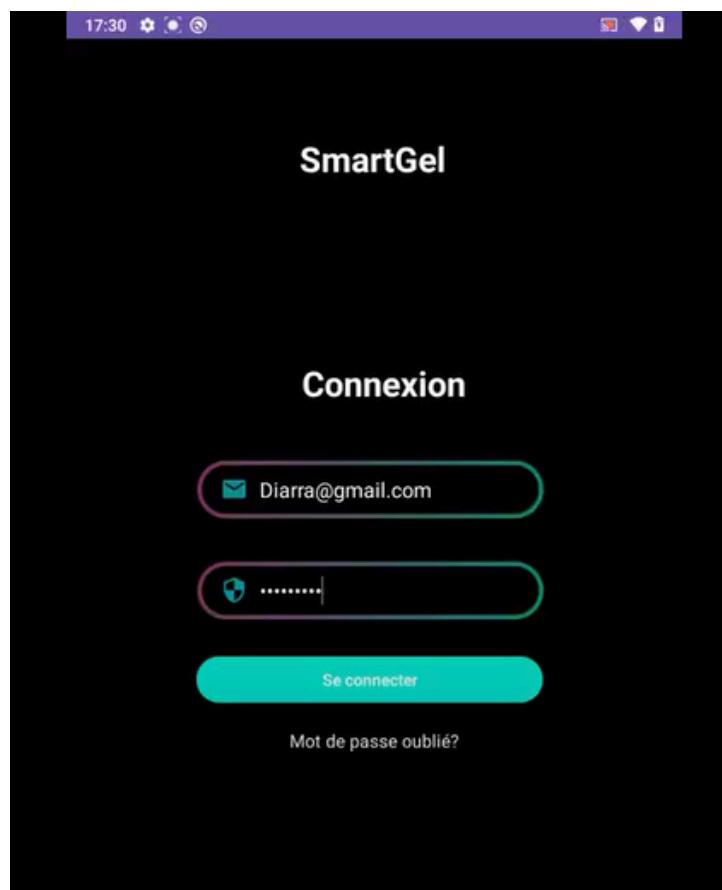
- ★ Il peut superviser les informations de plusieurs établissements dont il a la charge.
- ★ Il peut visualiser les bornes, les alertes, les missions ainsi que l'historique des missions.

5.3) Application SmartGel

Dans cette partie, nous allons vous présenter certaines interfaces utilisateur en fonction du rôle de l'utilisateur, ainsi que certaines interactions que chaque utilisateur peut réaliser.

★ Connexion :

Dans notre application, nous avons trois types d'utilisateurs, chacun pouvant se connecter via l'application pour réaliser des actions spécifiques selon son rôle.



Lorsque nous identifions la personne qui se connecte, l'utilisateur est dirigé vers son interface spécifique. Par exemple, un Responsable agent qui se connecte sera redirigé vers l'interface correspondant à son rôle.

Dans cette ligne de code nous récupérons le rôle de l'utilisateur:

```
int role = response.getInt( name: "Id_Role");
```

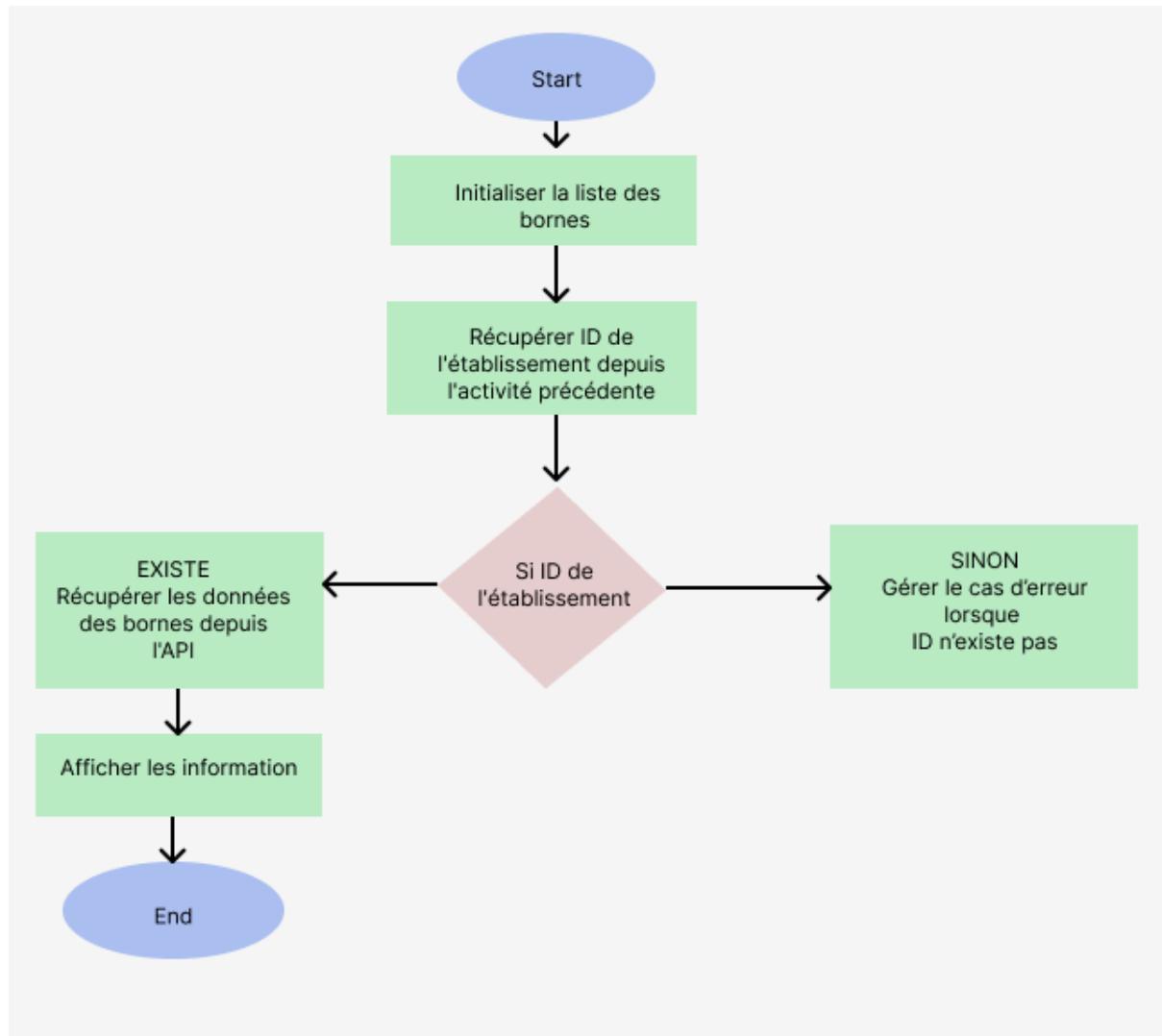
Pour après le diriger vers l'activité qui lui correspond:

```
case 2: // Responsable Agent
Intent intentResponsableAgent = new Intent( packageContext: LoginActivity.this, ResponsableAgentActivity.class);
```

★ Visualiser les Bornes :

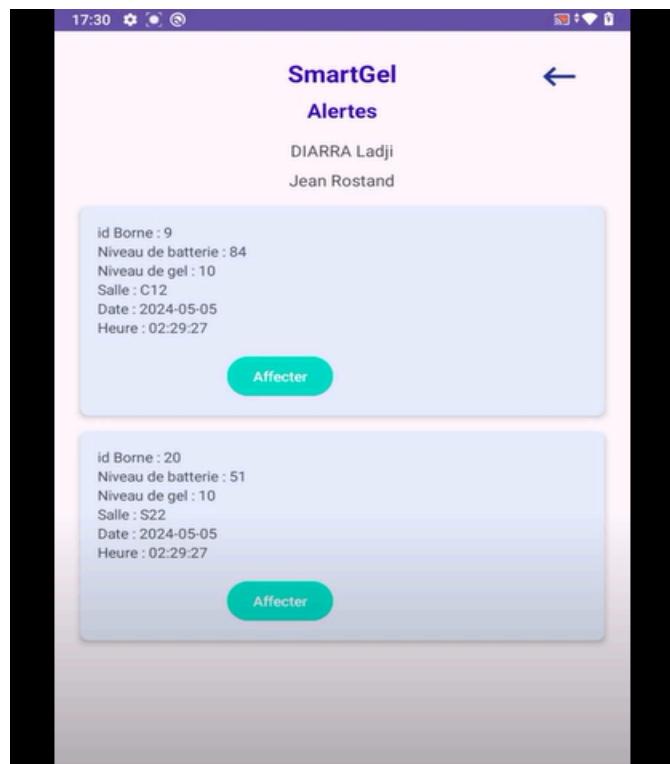
Grâce à l'application, tous les utilisateurs peuvent visualiser les bornes de l'établissement où ils travaillent. De cette façon, ils peuvent consulter ces informations à tout moment, même si les niveaux de borne ou de gel ne sont pas faibles.

Voici l'algorithme pour visualiser les bornes. C'est une façon claire et simplifiée de montrer le fonctionnement interne de cette partie.



★ Alertes et affectations :

Lorsqu'un Responsable agent se connecte via l'application, il peut à tout moment visualiser les bornes ayant un niveau faible de batterie ou de gel. Si nécessaire, il peut alors affecter une mission à un agent du même établissement pour résoudre le problème. Voici un exemple :



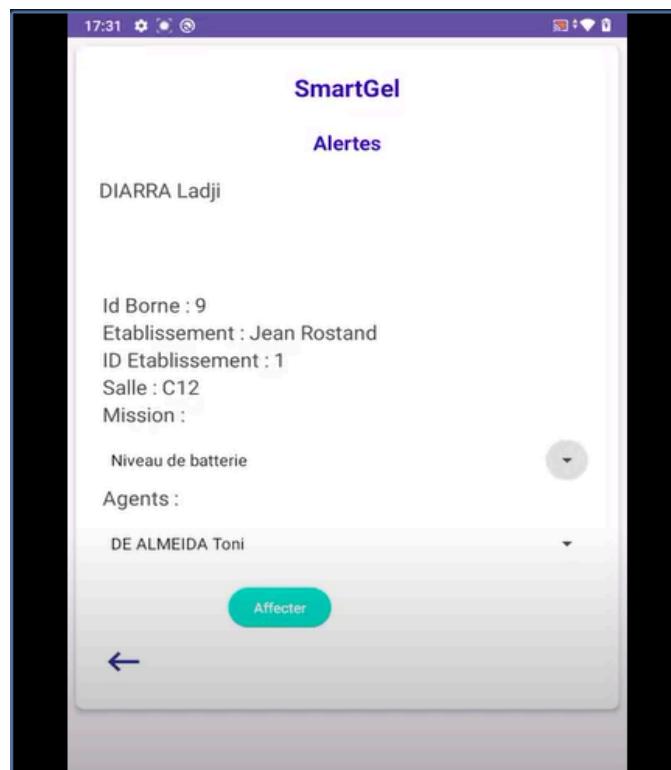
Le responsable agent peut cliquer dans le bouton "Affecter" et commencer une affectation.

Partie de code que génère l'événement du clic de bouton :

```
  DallyLozano
btnAffectation.setOnClickListener(new View.OnClickListener() {
    DallyLozano
    @Override
    public void onClick(View v) {
        // Création de l'intent pour passer à DoMissionActivity
        Intent intent = new Intent(packageContext: AlertesActivity.this, DoMissionActivity.class);
```

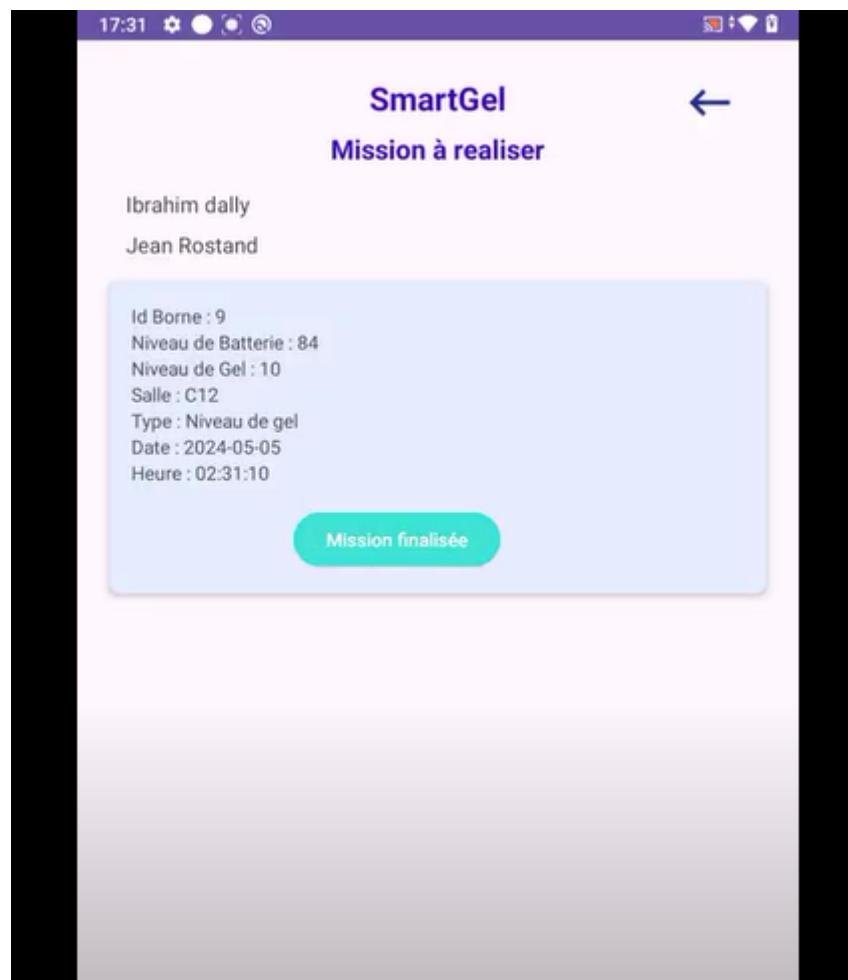
C'est à ce moment que l'utilisateur se dirige vers la Activity Do Mission pour générer une mission pour un Agent.

Interface pour réaliser une affectation :

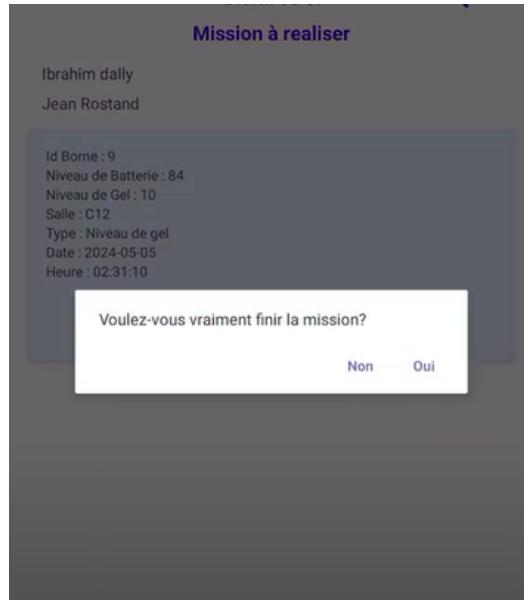


★ Réaliser une mission - Agent :

Un agent peut visualiser les missions qu'il doit effectuer via l'application. Il consulte ses missions, qu'il s'agisse de recharger le gel ou de changer la batterie, et les réalise. Une fois la tâche terminée, il peut la marquer comme complétée dans l'application.



Finalisation de la tâche



Ce bloc de code montre comment gérer la fin d'une mission. La fonction `showConfirmationDialog` affiche une boîte de dialogue avec le message "Voulez-vous vraiment finir la mission ?". Si l'utilisateur clique sur "Oui", l'ID de la mission est récupéré à partir de la liste des missions (`mMissions`). Ensuite, une requête POST est envoyée à l'API pour marquer la mission comme terminée via la fonction `updateMissionStatus`. La mission est alors supprimée de la liste des missions, et l'adaptateur est notifié du changement pour mettre à jour l'affichage.

```
private void showConfirmationDialog(int position) {
    AlertDialog.Builder builder = new AlertDialog.Builder(itemView.getContext());
    builder.setMessage("Voulez-vous vraiment finir la mission?")
        .setPositiveButton("Oui", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                // Récupérer l'ID de la mission à marquer comme terminée
                int idStatus = mMissions.get(position).getIdStatus();

                Log.d("idStatus", "Valeur de idStatus : " + idStatus);

                // Envoyer une requête POST à l'API pour marquer la mission comme terminée
                updateMissionStatus(idStatus);
                // Supprimer la mission de la liste
                mMissions.remove(position);
                // Notifier l'adaptateur du changement
                notifyItemRemoved(position);
            }
        })
        .setNegativeButton("Non", null)
        .create()
        .show();
}
```

5.4) Contraintes

Au cours du processus de création de l'application, plusieurs défis ont été rencontrés et résolus. Tout d'abord, la familiarisation avec le langage Java a représenté un défi capital, nécessitant une compréhension rapide pour pouvoir avancer dans le développement de l'application. Ensuite, la conception de l'application a exigé une vision claire de ce que nous voulions afficher, suivie de la phase de développement proprement dite.

Un des obstacles rencontrés était lié à l'affichage de plusieurs CardView dans le code. Cependant, une solution a été trouvée pour mettre à jour ces CardView avec pour objectif qu'elles reflètent les informations actuelles. De même, l'interrogation de l'API a présenté des difficultés, parfois en raison de l'utilisation incorrecte du format de communication, entraînant des problèmes dans la récupération des données.

Un autre défi majeur était lié à la création des notifications. L'utilisation d'un timer pour interroger l'API à intervalles réguliers n'était pas correctement configurée, ce qui a conduit à des problèmes de fonctionnement. Ces défis ont nécessité une réflexion approfondie et une révision des approches pour parvenir à des solutions efficaces.

En résumé, bien que certains obstacles aient été rencontrés, ces contraintes ont été surmontées grâce à une approche réfléchie et à une allocation de temps supplémentaire pour résoudre les problèmes rencontrés.

VI. CONCLUSION

À l'issue de notre projet final, nous tirons des leçons significatives de notre parcours. La collaboration étroite entre les membres a engendré une dynamique de groupe efficace, mettant en évidence l'importance de travailler ensemble pour résoudre des problèmes complexes. Immersés dans les technologies de l'informatique et de la physique avec notre projet "Borne de Gel", nous avons été confrontés à des défis tangibles, stimulant ainsi notre acquisition de connaissances à la fois pratiques et théoriques.

Cette expérience formatrice nous prépare de manière concluante à notre transition vers le monde professionnel, renforçant nos aptitudes interpersonnelles et techniques. Notre rapport incarne l'engagement collectif que nous avons investi et les leçons inestimables que nous avons intégrées. En somme, ce projet final représente une étape essentielle dans notre formation, nous armant de manière effective pour nos futures carrières avec assurance et compétence.

VII. ANNEXES

Technologies et outils utilisés :



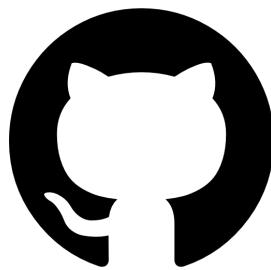
Workbench



Studio Android



Discord



Github