Borne de gel connectée

Lycée Polyvalent Jean Rostand
93420 Villepinte

Rapport de Projet Borne de gel connectée

Présentation générale

Equipe

EDDAFAOUI Nawfel

HADJI Elyes

AMIRAT Massil

KANAGARAJAH Thadsayian

Table des matières:

3. Différentes requêtes

- Etudiant 3 (Massil)
    1. Capter le niveau de batterie
    2. Capter le niveau de gel
    3. Envoyer les informations à l’API
    - Etudiant 4 (Thadsayian)
        1. Les bornes sous Android
        2. Interface Responsable Agents/Agent
        3. Fonctionnement de l’application

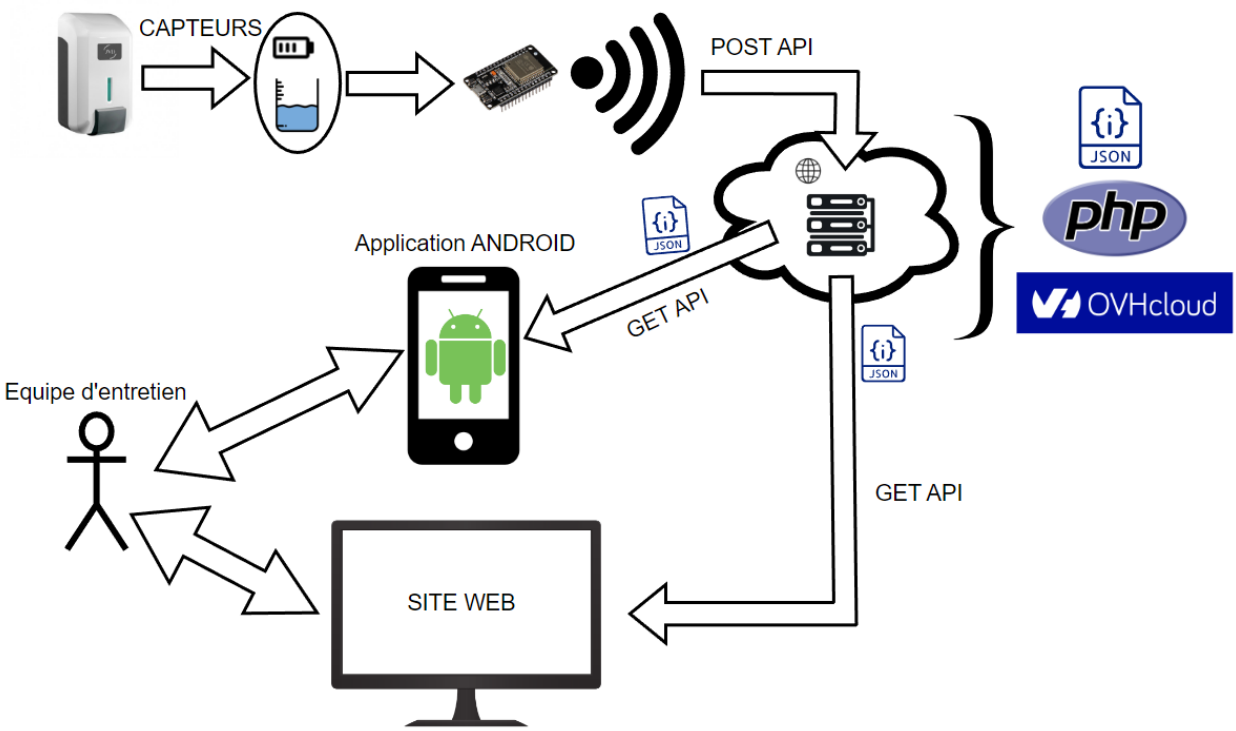
Objectif du projet

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, ce projet vise à permettre une continuité de la disponibilité du gel hydroalcoolique au niveau de chaque borne d’un établissement.

Pour résoudre ce problème, il est nécessaire d'équiper chaque borne de gel d’une partie communicante afin de centraliser le suivi et le contrôle des niveaux de gel hydroalcoolique et de batterie et enfin, alerter en cas de manque de gel et ou de niveau de batterie épuisé.

Synoptique du projet



Cahier des charges

	Fonctions à développer et tâches à effectuer	
Étudiant 1 EC☐IR ■	✓ **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**	Installation : **Installation OS , Workbench , bdd mysql** Mise en œuvre : **mise en œuvre de la bdd et des fichiers Json** Configuration : **Des accès au serveur de bdd** Réalisation : **SGBD avec arborescence REST** Documentation : **Installation, Prise en main et déploiement**
Étudiant 2 EC☐IR ■	✓ **Coder le site web Administrateur :** - Gestion des différents statuts ✓ **Coder le site web côté membre** ✓ **Héberger le site sur le serveur**	Installation : **Installation OS , IDE, client Ftp** Mise en œuvre : : **mise en œuvre du code HTML, CSS, javaScript et PHP** Configuration : **Des serveurs et de l'IDE,** Réalisation : **Site web et hébergement distant** Documentation : **d'Installation, de prise en main et déploiement de la solution**
Étudiant 3 EC☐IR ■	✓ **Coder les bornes électroniques connectés :** - Choix d'une carte ESP de petite taille - 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.	Installation : **Installation OS ou IDE de la carte électronique** Mise en œuvre : **cadenas électronique, capteurs de niveau de gel, de batterie** Configuration : **configuration réseau** Réalisation : **Du prototype de la partie électronique avec tests de validation pour chaque sous-partie** Documentation : **d'Installation, de prise en main et déploiement de la solution**

Étudiant 4 EC☐IR ■	✓ **Coder l'interface de l'application Androïd** ✓ **Mettre en œuvre la communication via JSON vers l'API Rest** ✓ **Récupérer les données et créer des interfaces dynamiques (alertes, visualiser des niveaux des bornes etc..)**	Installation : **Installation OS et IDE Android Studio** Mise en œuvre : **Application Android sur tablette** Configuration : **Android Studio avec tablette virtuelle.** Réalisation : **Application sécurisée** Documentation : **Installation, Prise en main et déploiement**

En résumé :

Nawfel	Elyes	Massil	Thadsayian
Mise en oeuvre d'une API REST et d'une base de données concernants les agents	Création du site web. Administration de son hébergement à distance	Installation des capteurs. Configuration de la carte électronique et de sa communication.	Mise en oeuvre de l'application Android et de son déploiement

Analyse des diagrammes UML

Diagramme de cas d’utilisation

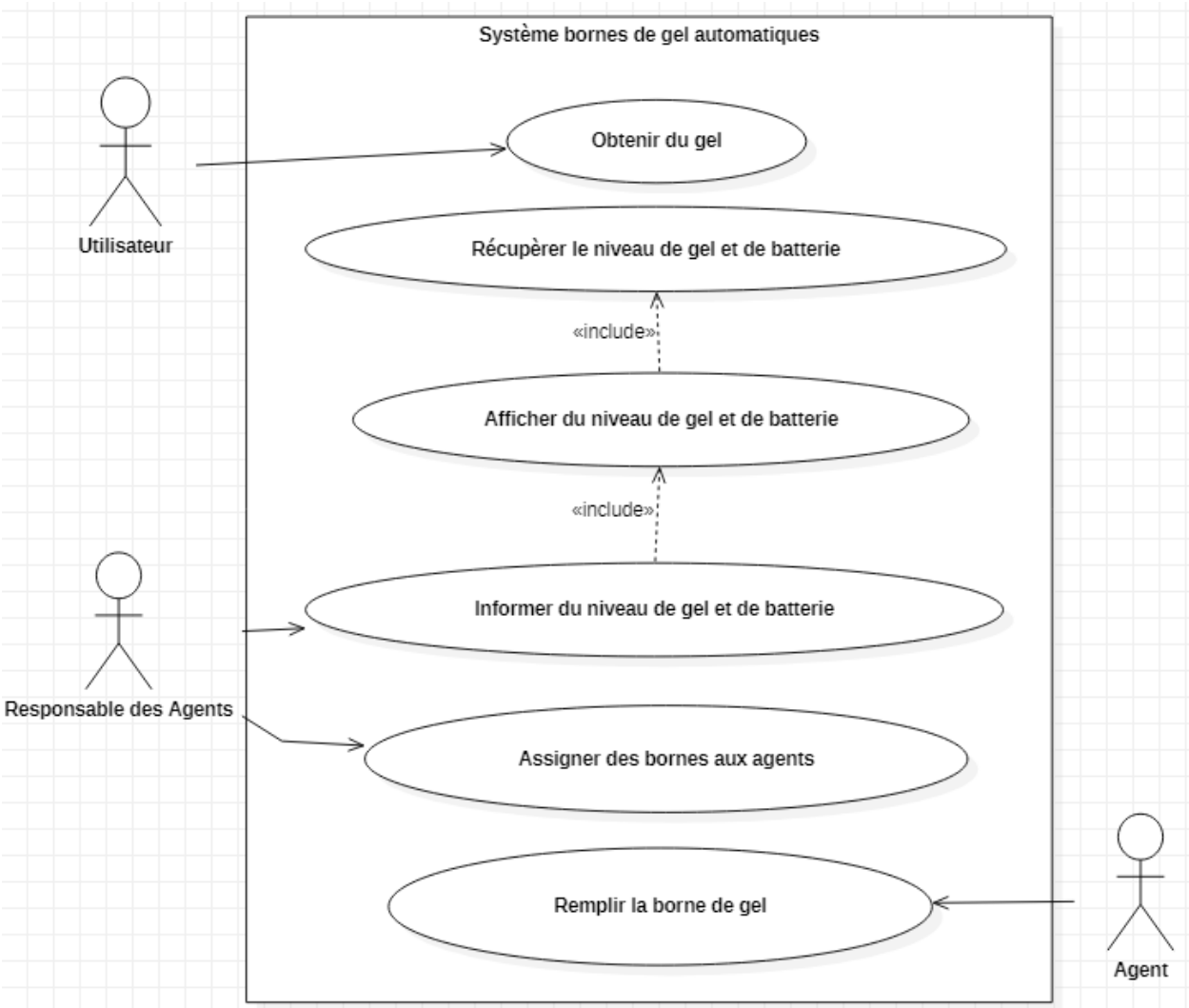


Diagramme de séquence

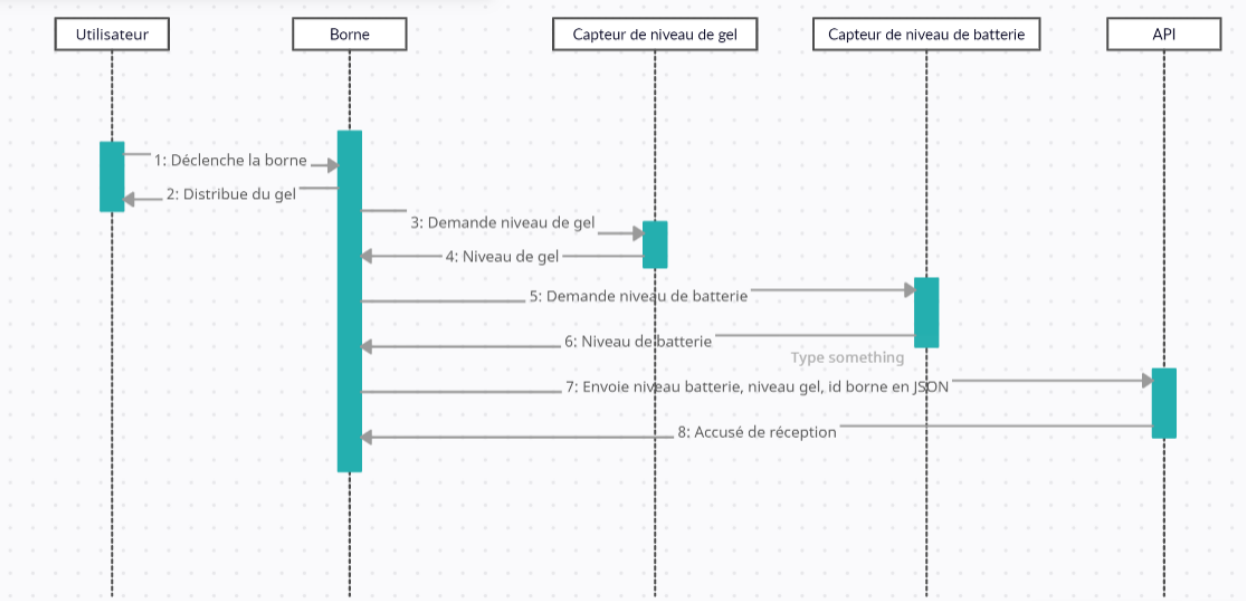
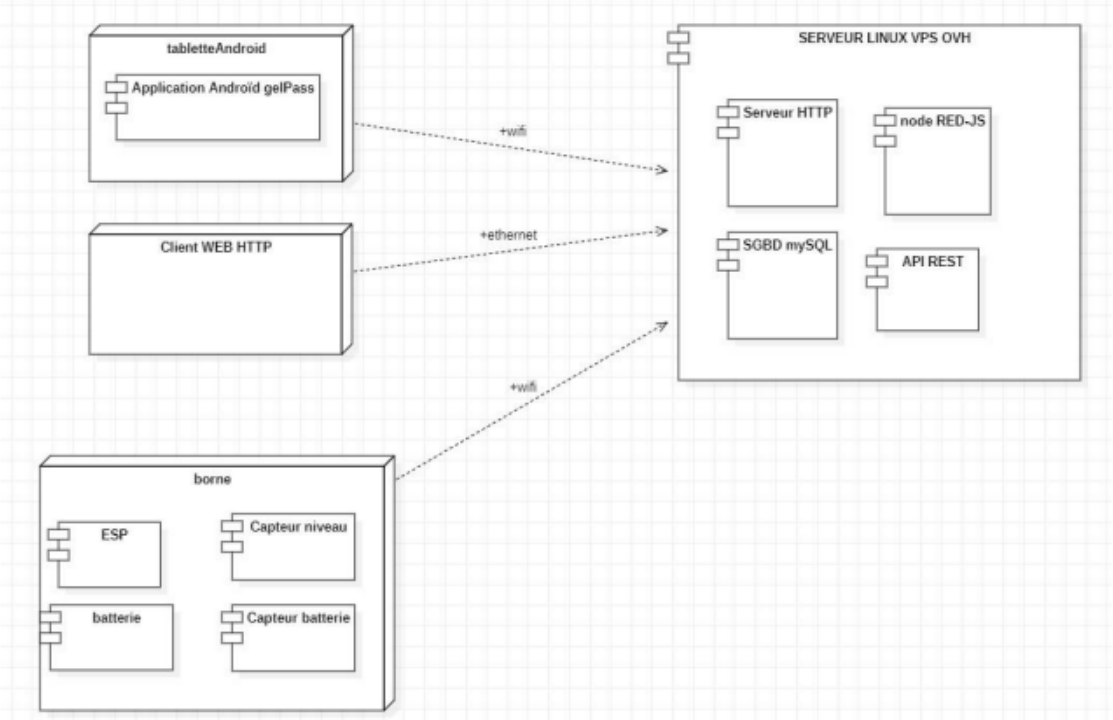


Diagramme de déploiement



Présentation des tâches

Etudiant 1 (Nawfel)

L'étudiant 1 devra réaliser l'API REST et concevoir la base de données :

- **Modélisation de la BDD et mise en œuvre**

- **Mise en œuvre de l'API REST pour stocker et communiquer les données ci-dessous :**

    1. **Le numéro (id) de la carte ESP32**
    2. **Le niveau de gel**
    3. **Le niveau de batterie**
    4. **La salle (lieu) où est installée la borne de gel**

L'api fournira des données au format JSON, l'étudiant fournira également la documentation de celle-ci.

- **Création de l'API REST sur PHPStorm**

```php
<?php
// Connexion à la base de données
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

$conn = mysqli_connect($servername, $username, $password, $dbname);

// Vérification de la connexion
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Récupération des données envoyées par l'ESP32
$esp32_id = $_POST['esp32_id'];
$niveau_gel = $_POST['niveau_gel'];
$niveau_batterie = $_POST['niveau_batterie'];
$salle = $_POST['salle'];

// Stockage des données dans la base de données
$sql = "INSERT INTO borne_gel (esp32_id, niveau_gel, niveau_batterie, salle) VALUES ('$esp32_id', '$niveau_gel', '$niveau_batterie', '$salle')";

if (mysqli_query($conn, $sql)) {
    echo "Enregistrement effectué avec succès.";
} else {
    echo "Erreur : " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
```

- **Création d’un fichier ReadMe pour les instructions de l’API**

API REST pour la borne de gel hydroalcoolique

Cette API permet de stocker et de récupérer les données suivantes de la borne de gel hydroalcoolique :

ID de la carte ESP32, Niveau de gel, Niveau de batterie, Lieu où est installée la borne de gel L'heure et la date de la dernière mise à jour de la table

Utilisation

Pour utiliser cette API, vous devez envoyer une requête HTTP POST avec les paramètres suivants :

esp32_id : l'ID de la carte ESP32

niveau_gel : le niveau de gel de la borne de gel (en %)

niveau_batterie : le niveau de batterie de la borne de gel (en %)

salle : le lieu où est installée la borne de gel

Vous pouvez envoyer la requête en utilisant un outil de requête HTTP tel Postman.

Exemple de requête

Voici un exemple de requête HTTP POST pour envoyer les données de la borne de gel :

POST http://example.com/api/borne-gel

Content-Type: application/content type

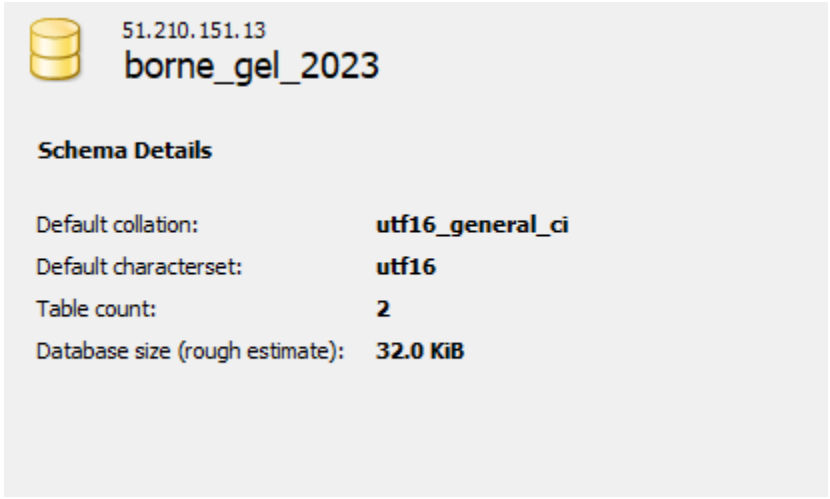
esp32_id=123456&niveau_gel=50&niveau_batterie=80&salle=Salle A

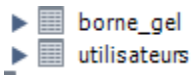
Si la requête est envoyée avec succès, l'API renvoie une réponse HTTP 200 OK avec le message suivant :

Enregistrement effectué avec succès.

Si une erreur se produit lors de l'enregistrement des données dans la base de données, l'API renvoie une réponse HTTP 500 avec un message d'erreur.

- **Création d’une base de données et des tables avec leurs relations**





- **Exemple de données reçus par la borne de gel connectée**

id	esp32_id	niveau_gel	niveau_batterie	salle	timestamp
26	16	34	57	2	2023-05-25 16:38:16
27	46	34	57	3	2023-05-25 16:40:10
29	25	46	52	3	2023-05-25 16:44:01
30	28	40	65	4	2023-05-25 16:44:34
31	73	62	27	5	2023-05-25 16:45:12
32	48	39	98	6	2023-05-25 16:45:41
33	21	19	90	7	2023-05-25 16:46:39
34	81	67	16	8	2023-05-25 16:47:02
35	67	61	59	9	2023-05-25 16:47:25
36	94	60	55	10	2023-05-25 16:47:44
37	37	28	94	11	2023-05-25 16:48:05
38	47	71	32	12	2023-05-25 16:48:23
39	36	30	54	13	2023-05-25 16:48:45
40	41	77	35	14	2023-05-25 16:49:07
NULL	NULL	NULL	NULL	NULL	NULL

- **Arborescence API REST**

Arborescence API REST BORNE DE GEL

- agents
  - /GET *Liste tous les agents*
  - /POST *Crée un nouvel agent*
  - GET /{id} *Récupère un agent spécifique*
  - PUT /{id} *Met à jour un agent spécifique*
  - DELETE /{id} *Supprime un agent spécifique*
- /gel-bornes
  - GET / *Récupère l'état actuel de la borne de gel*
  - POST /action *Effectue une action sur la borne de gel (par exemple, "start", "stop", "reboot")*

Etudiant 2 (Elyes)

Création du Site Web pour la borne de Gel

. Présentation

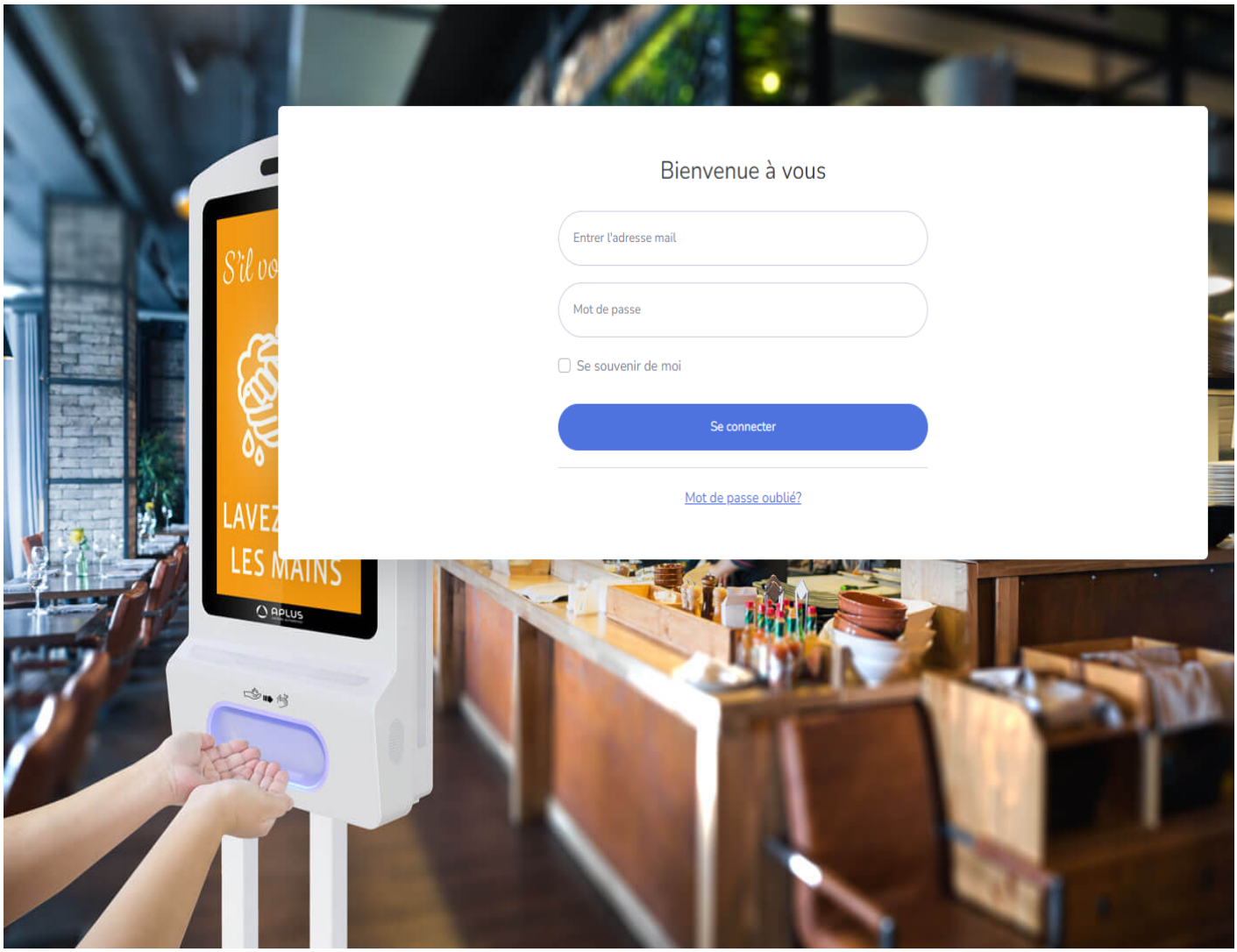
Ma principale tâche est la création d'un site web qui permet de réaliser plusieurs opérations correspondant à chaque statut.
Une page pour le responsable technique qui lui aura les opérations suivantes:

- *Création des comptes RA (Responsable des Agents)*
- *Création comptes Agent*
- *Affectation des missions aux agents*
- *Consultation de l'état des bornes*

Une page pour le responsable des agents qui lui aura les même opérations que le responsable technique sauf que il peut juste créer des comptes pour les agents
Et pour finir, une page pour l'agent qui pourrait alors juste consulter l'état des bornes et voir les notifications qui lui seront envoyées par les responsables

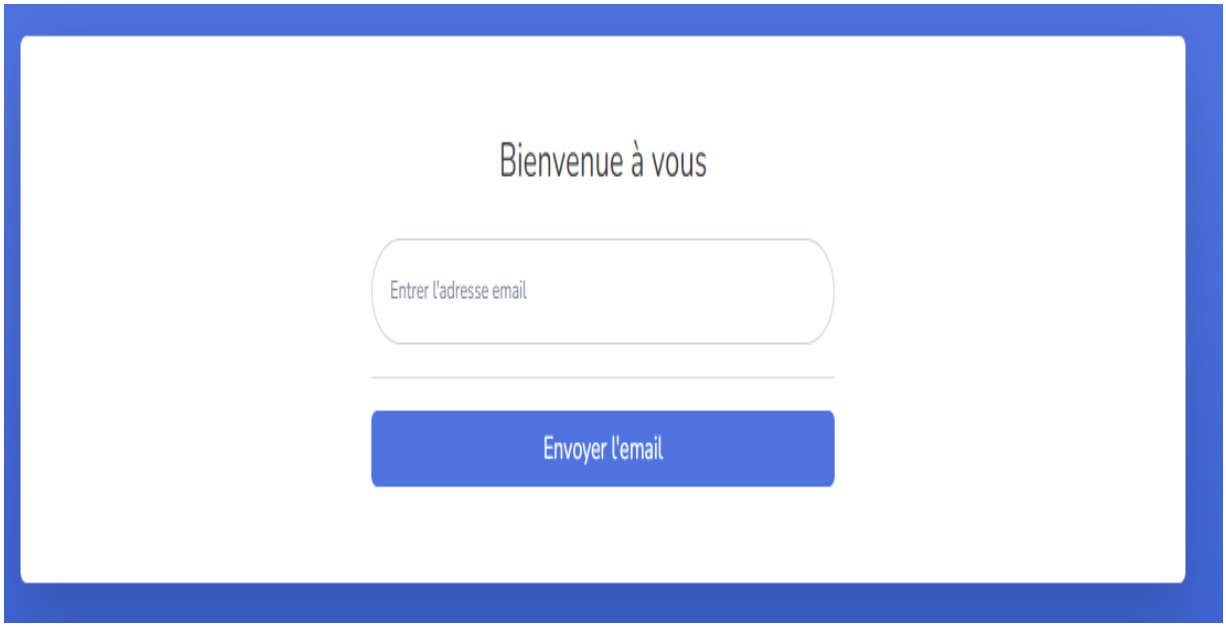
II. Présentation des pages du site web

- **Page de connexion**



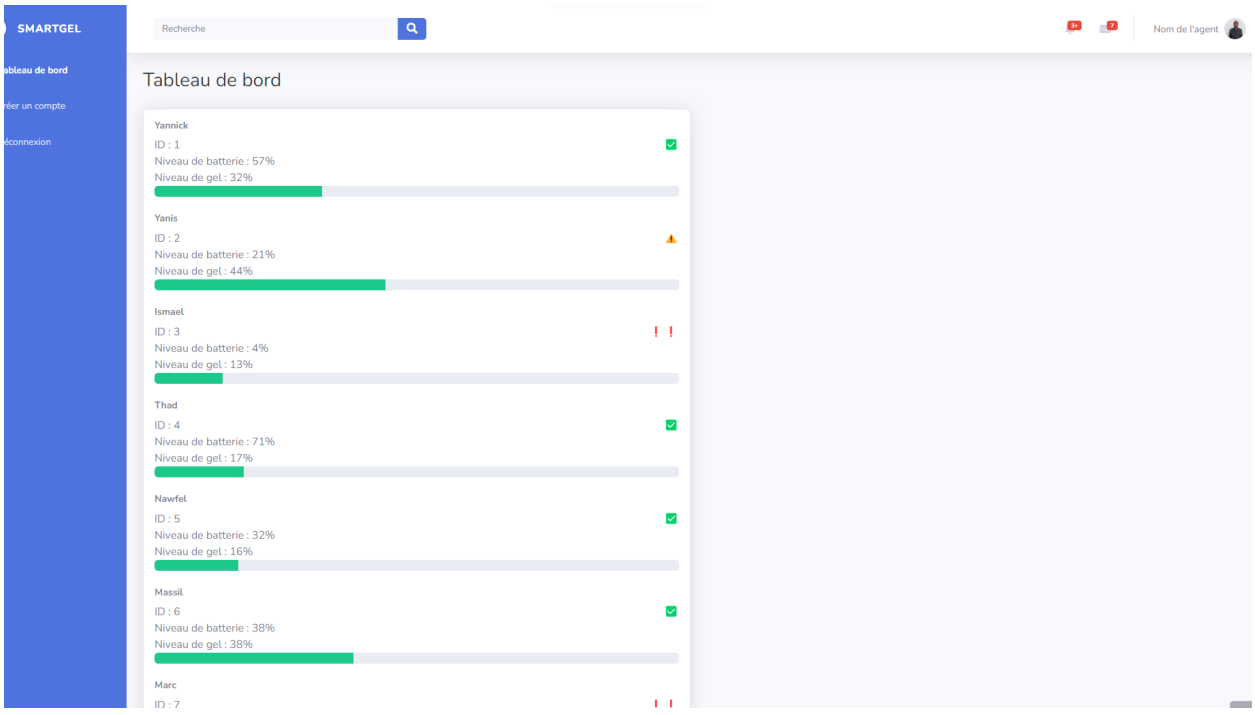
Cette page est la page principale en arrivant sur le site web , elle permet de se connecter .Chaque utilisateur se connectera à sa page qui sera différente selon leur rôles (fournisseur du système Responsable Technique (RT) Responsable des agents (RA) et Agent).

- **Page mot de passe oublié**



Cette page permet d’envoyer un mail à l'utilisateur pour retrouver le mot de passe du compte en cas d'oubli.

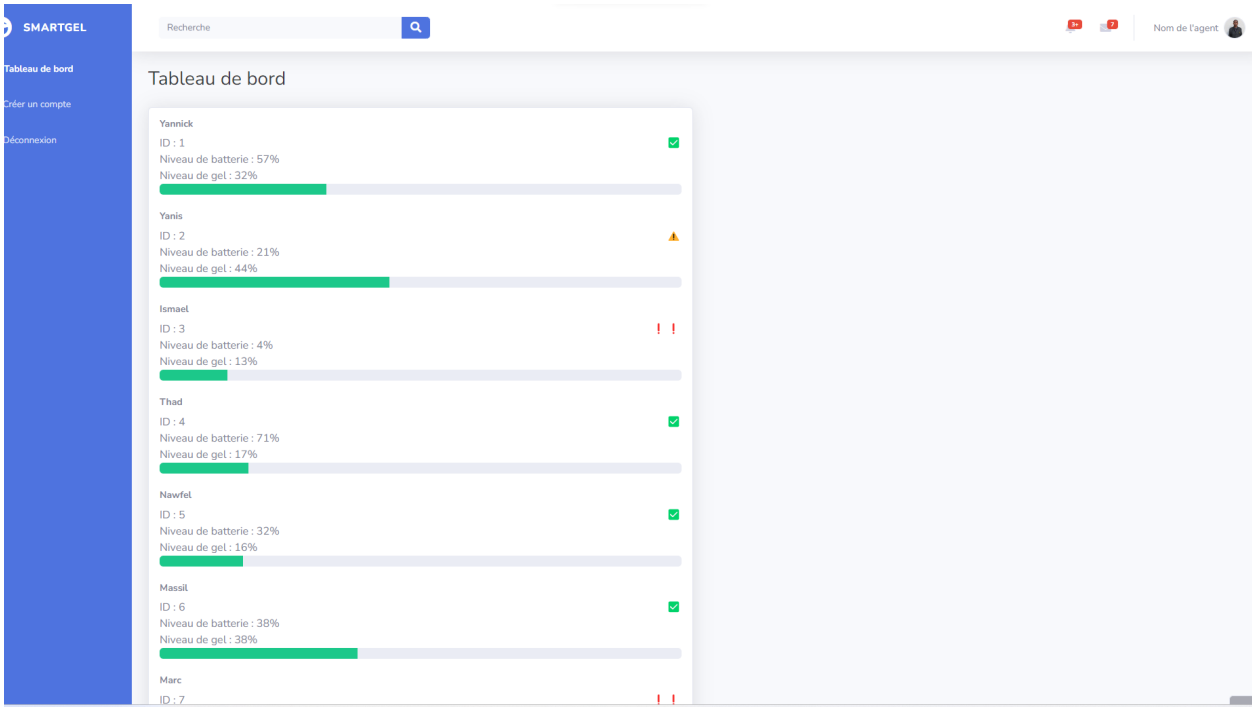
- **Page d'accueil pour les responsables des agents et responsables techniques**



Les utilisateurs qui auront accès à cette page seront les responsables des agents et les responsables techniques.
Le but de cette page est de Consulter les bornes rapidement pour les responsables en regardant si il y a un manque de batterie ou de gel et d'alerter en envoyant une notification aux agents afin qu'ils puissent recharger les bornes et affecter les bornes aux agents

Il manque une page qui servira à assigner des bornes aux agents directement en appuyant sur la borne sélectionnée.

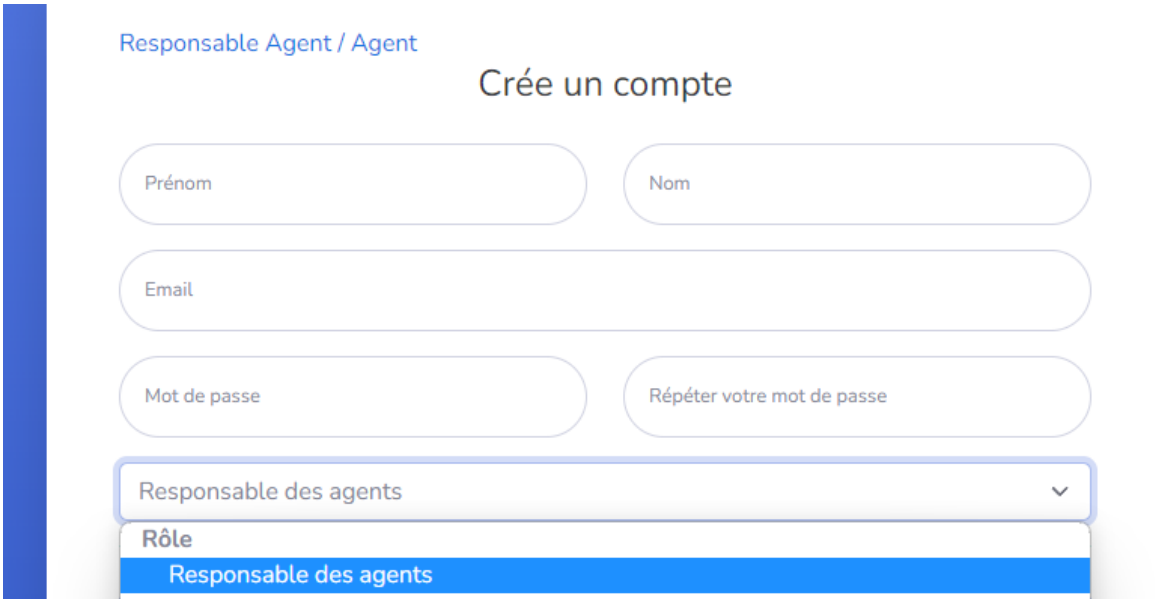
- **Page d'accueil pour les agents**



L'agent lui pourras alors juste consulter l'état des bornes contrairement aux responsables qui eux ont un contrôle totale.elle leur permettra alors de voir quelle borne leur est attribué est ensuite la recharge

- **Page de création de compte pour le responsable des agents**

Cette page est la page création de compte pour les responsable des agents , elle permet à la fois de créer un compte pour un utilisateur qui aura le rôle de responsable des agents ou juste un compte pour un agent



- **Page de création de compte pour le responsable des agents**

Cette page est la page création de compte pour les responsables des agents , elle permet à la fois de créer un compte pour un utilisateur qui aura le rôle de responsable des agents ou juste un compte pour un agent .

III. Différentes requêtes

Connexion:

Lorsque le client souhaite se connecter au site, une requête ajax est effectuée vers l'API pour vérifier les informations saisies dans les champs email et mot de passe.

Inscription:

Si l'utilisateur n'a pas de compte, il doit demander au responsable de créer un compte pour lui en remplissant un formulaire. Dans ce cas, une requête ajax est également effectuée pour récupérer les données saisies par l'utilisateur et les envoyer à l'API.

Conclusion:

Ma partie du projet est presque terminée, mais il reste encore quelques tâches à accomplir avant la prochaine présentation. Au cours de ce projet, j'ai pu acquérir de nouvelles compétences en travaillant en groupe et approfondir mes connaissances pour améliorer mon site web. Ce projet m'a permis de mettre en pratique ce que j'ai appris au cours des deux dernières années.

Etudiant 3 (Massil)

L'objectif ici est de mettre en place une solution physique apportant à une borne de gel hydroalcoolique une manière de communiquer son niveau de gel et de batterie de façon autonome et régulière.

I.Capter le niveau de batterie

La batterie délivrant 6 volts, on utilise un pont diviseur de façon à délivrer aux appareils électroniques 3v - le maximum qu'ils peuvent recevoir. Pour se faire, on utilise une formule qu'est la suivante :

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

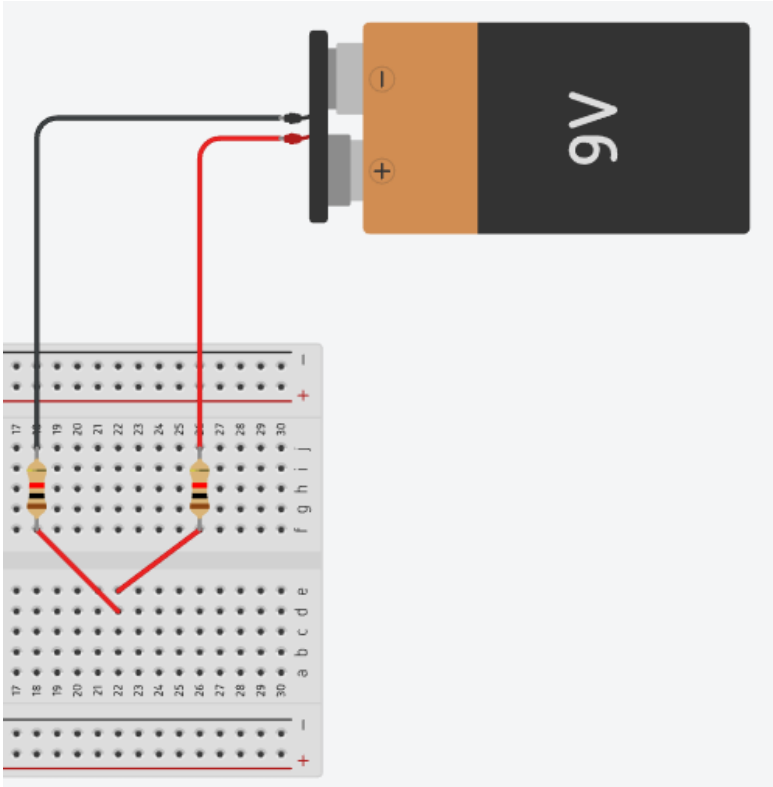
En adaptant à notre situation :

$$\frac{6 * \text{Résistance 1}}{\text{Résistance 1} + \text{Résistance 2}} = 3$$

On se rend compte que les résistances valent 200 chacune, on obtient donc le calcul suivant qui vérifie notre résultat :

$$\frac{6 \quad * 200}{200 \quad + 200} = 3$$

En situation réelle, les branchements ressemblent à ceci :



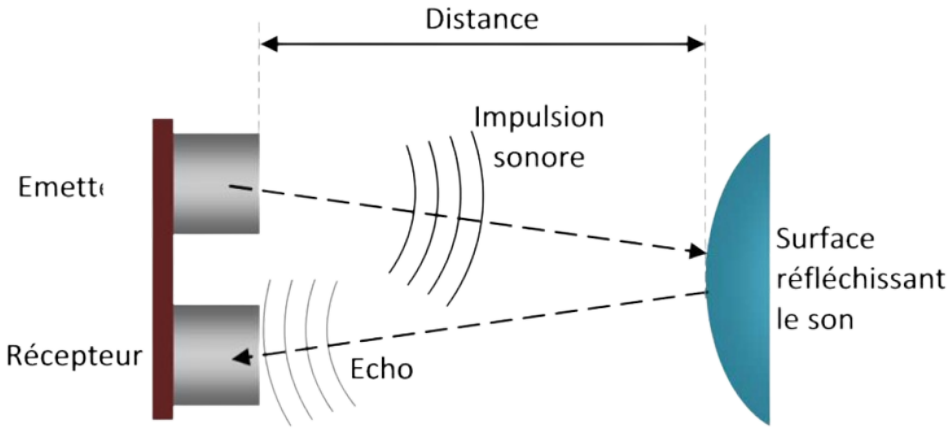
Il nous suffit ensuite de brancher au bout un fil vers l'ESP32 à un pin digital qui reçoit au maximum 3 volts. S'il reçoit 3 volts, il renvoie 4095. En utilisant cette information, il ne nous reste plus qu'à faire un pourcentage pour avoir le taux de batterie.

Le code en lui même ressemblera donc à ça :

```
25  int nivBatterie(){
26    int recu = analogRead(34);
27    recu *= 100;
28    recu /= 4095;
29    return recu;
30  }
```

II.Capter le niveau de gel

Pour calculer le niveau de gel, on utilise un capteur à ultrasons calculant la distance entre lui-même et un objet en face.
Voici un petit schéma explicatif de son fonctionnement :



En le plaçant sur le dessus du récipient de gel, on obtient donc la distance entre le haut et la surface du gel présent dans la borne, il nous restera plus qu'à en faire le pourcentage en prenant en compte la distance maximale atteignable entre le capteur et le fond du récipient donc.

Le code ressemble donc à ceci :

```
long getDist(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  dur = pulseIn(echoPin, HIGH);
  dist = dur * 0.034/2;
  return dist;
}
```

$$d = v \times t$$

On utilise une formule simple permettant de calculer la distance entre le capteur et l'obstacle. On divise par 2 le résultat pour n'avoir qu'un seul trajet de l'ultrason et pas l'aller retour en entier.

III. Envoyer les informations à l'API

L'API ayant été introduite plus tôt, il n'est pas nécessaire d'expliquer son fonctionnement. En revanche, son utilisation avec l'ESP32 reste important à démontrer.

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

//[redacted]

const char* ssid = "[redacted]";
const char* password = "[redacted]";
uint32_t chipID = 0;
HTTPClient http;
Ultrasonic ultrasonic(4);
```

Après avoir appelé des librairies nous facilitant grandement la connexion à un réseau WiFi et l'utilisation d'une API, on définit des constantes utiles pour nous mettre au travail.
Chaque chose en son temps, il faut d'abord connecter l'ESP32 au réseau, cela se présente sous la forme du code suivant :

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(5000);
  Serial.print("."); //tant que l'esp n'est pas connecté, afficher des points
}
```

Tant qu'on est pas connecté, attendre 5 secondes et réessayer.

Une fois connecté, il ne nous reste plus qu'à envoyer les informations reçues par les capteurs par un POST à l'API.

```
41    do{
42      http.begin("https://e59ec5f0-1a3a-45fe-8402-dfe2e722ed61.mock.pstmn.io/UPbornes?");
43      http.addHeader("Content-Type","application/x-www-form-urlencoded");
44      String httpRequestData = "id="+chipId+"&batterie="nivBatterie()"&gel="nivGel();
45      int httpResponseCode = http.POST(httpRequestData);
46    }while(httpResponseCode != 200);
47    http.end();
```

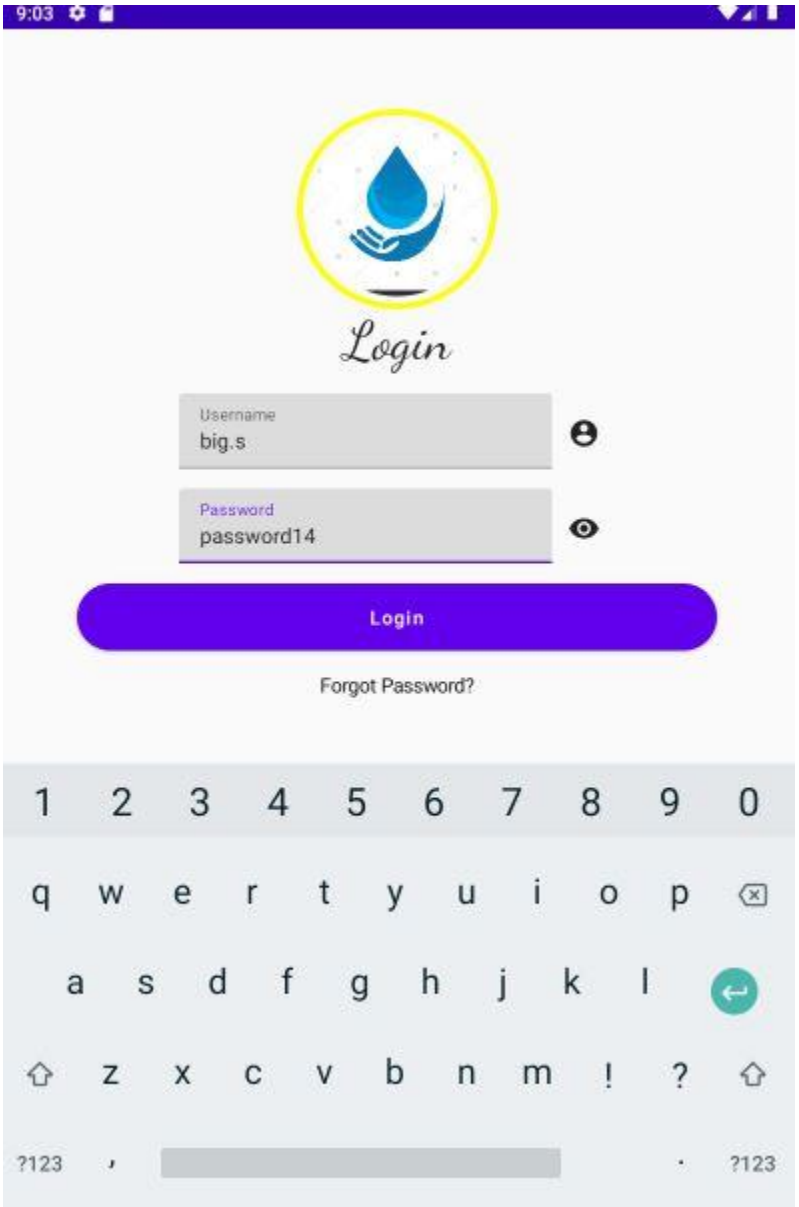
Voilà comment se présente le code. On entre d'abord l'URL de l'API et enfin les informations relatives à la borne.

Etudiant 4 (Thadsayian)

IV. Les bornes sous Android

Les utilisateurs possèdent leur propre compte pour se connecter sur l'application android. En fonction de leur statut dans l'entreprise l'interface est différente . Pour visualiser les informations des bornes à distance,ils doivent se connecter pour avoir l'interface qui leur permet la gestion.

Pour avoir l'interface de responsable des agents (RA) et des agents (A)

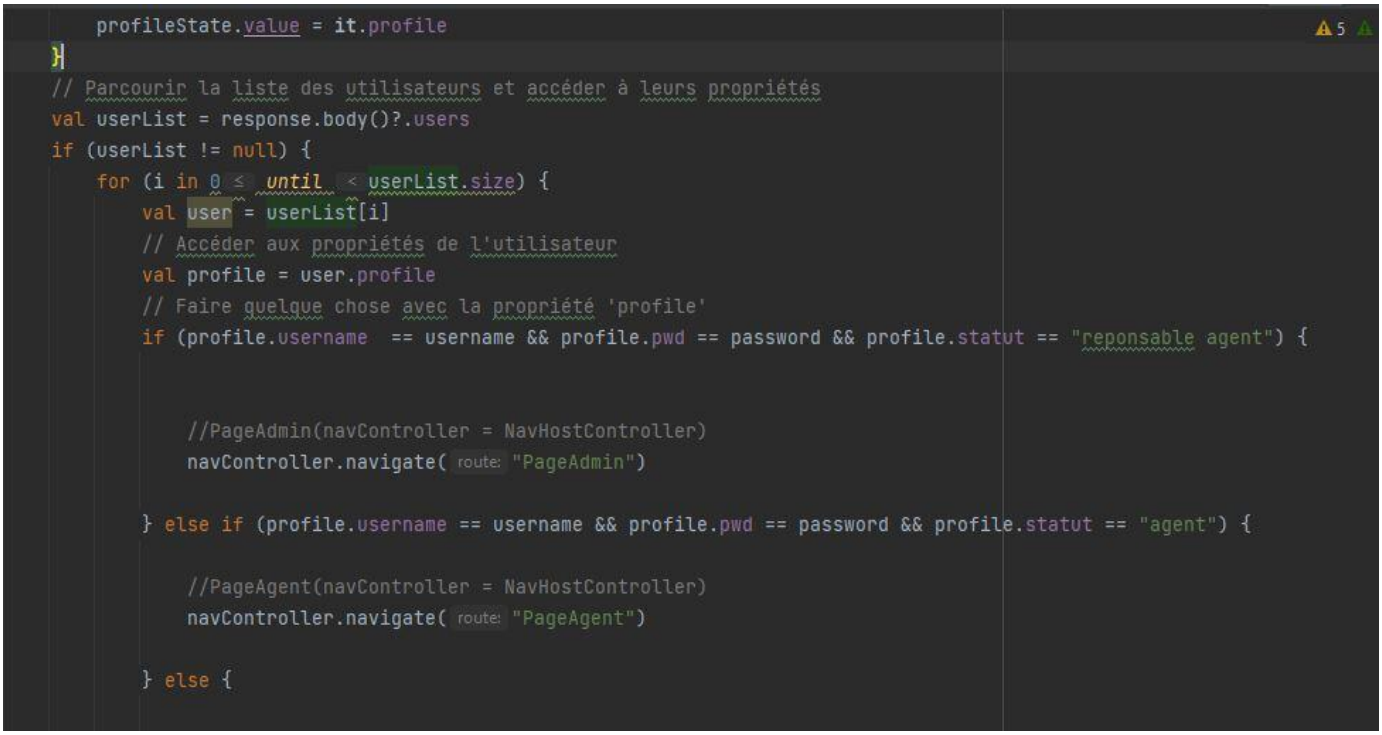


Pour se connecter sur différentes interfaces, Le statut joue un rôle important. Les utilisateurs qui ont le statut agent pouvant avoir l'interface d'agent et les utilisateurs qui ont le statut responsable agent (RA) pouvant avoir l'interface de responsable agent.

```
"statut": "agent",
"pwd": "password13",
"username": "marc.j"
```

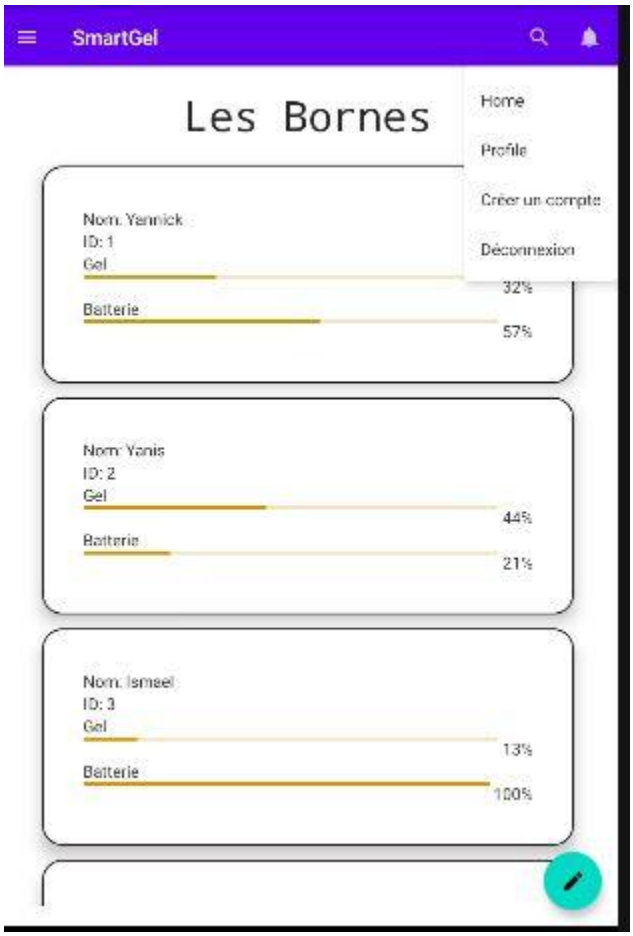
```
"statut": "responsable agent",
"pwd": "password14",
"username": "big.s"
```

Sur cette capture d'écran, on compare les valeur rentrée par utilisateur avec la valeur récupéré par l'API pour avec la bonne interface d'utilisateurs .



```
    profileState.value = it.profile
}
// Parcourir la liste des utilisateurs et accéder à leurs propriétés
val userList = response.body()?.users
if (userList != null) {
    for (i in 0 ≤ until < userList.size) {
        val user = userList[i]
        // Accéder aux propriétés de l'utilisateur
        val profile = user.profile
        // Faire quelque chose avec la propriété 'profile'
        if (profile.username  == username && profile.pwd == password && profile.statut == "reponsable agent") {


            //PageAdmin(navController = NavHostController)
            navController.navigate( route: "PageAdmin")

        } else if (profile.username == username && profile.pwd == password && profile.statut == "agent") {

            //PageAgent(navController = NavHostController)
            navController.navigate( route: "PageAgent")

        } else {
```

Interface Responsable Agents



Interface Agent

Voici les différentes tâches qui m'ont été confiés afin de réaliser l'application Android :

Responsable des Agents

- ➔ Consulter les informations de toutes les bornes,
- ➔ Permettre d'affecter les bornes à recharger à chaque agent
- ➔ Créer des notifications dans l'application (alerte sonore et/ou visuelle) lorsqu'une borne de gel manque de gel ou est déchargée.

Agents :

- ➔ Consulter les bornes qui lui sont affecté avec l'indication de la maintenance (gel ou batterie à changer)

Fournisseur de système :

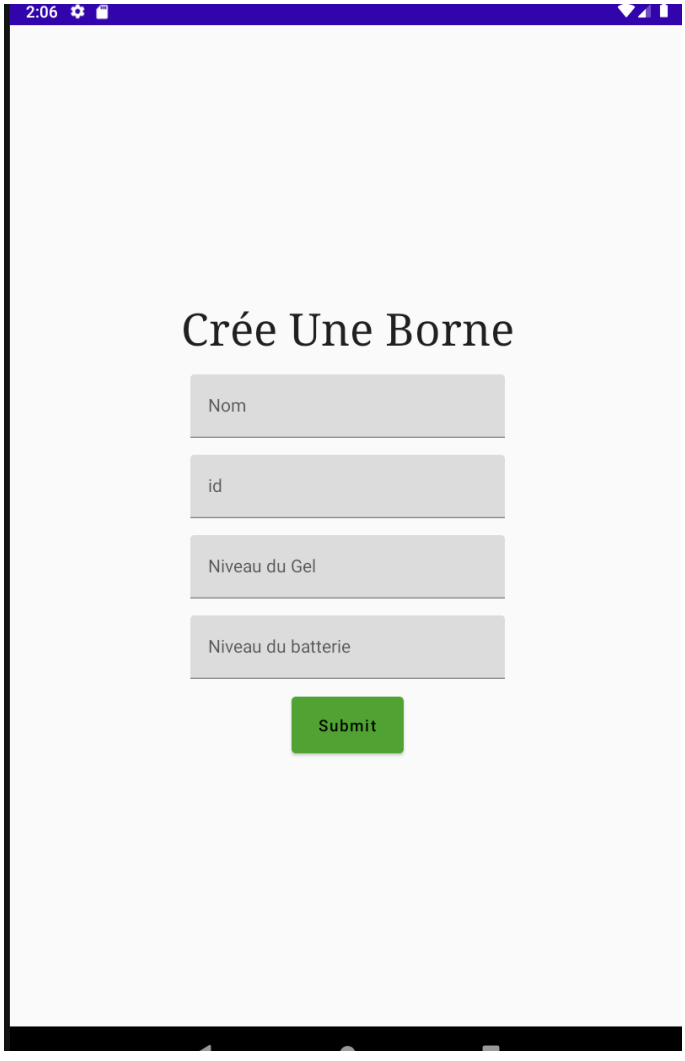
- ➔ Consulter l'ensemble des bornes ainsi que leurs états.

Fonctionnement de l’application

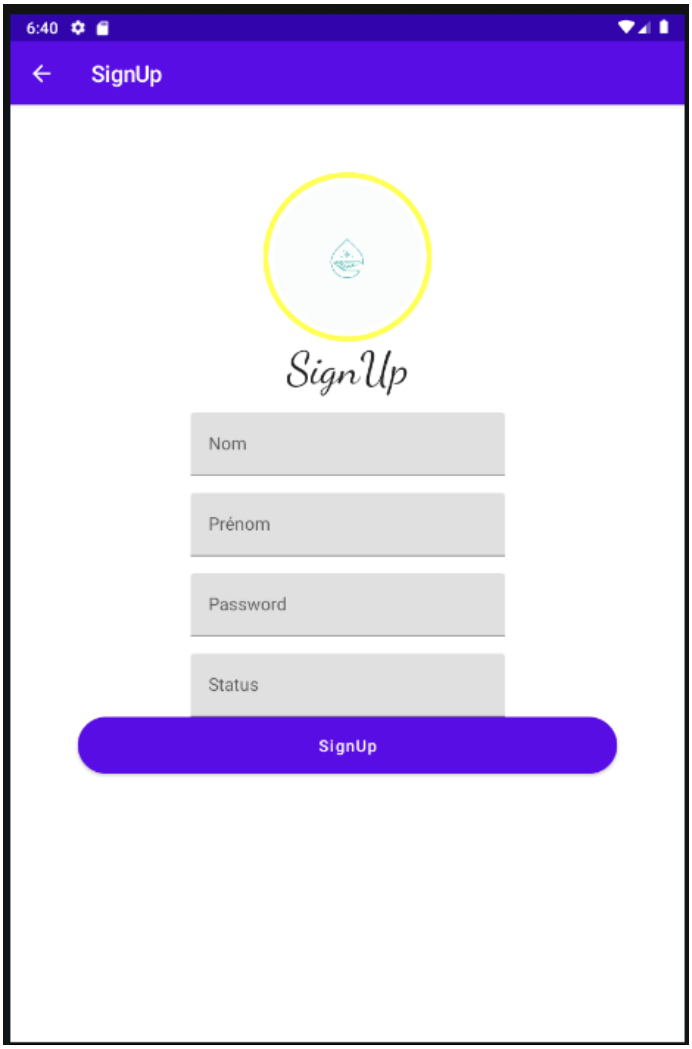
Connexion à une base de données distante

Avant il était impossible de se connecter directement à une base de données distante donc on utilisait un fichier PHP(API) qui faisait le traitement des requêtes.

Formulaire pour créer une Borne de Gel via l’application



Pour créer une borne , il faut saisir les informations demandées sur le formulaire, Puis lorsqu’on click sur le bouton “Submit“ on envoie une requête POST vers le serveur de l’entreprise avec les informations entrées par le Responsable des agents.



Pour créer un utilisateur , il faut saisir les informations demandées sur le formulaire, Puis lorsqu'on clique sur le bouton SignUp, on envoie une requête POST vers le serveur de l'entreprise avec les informations entrées par le Responsable des agents.

ANNEXES

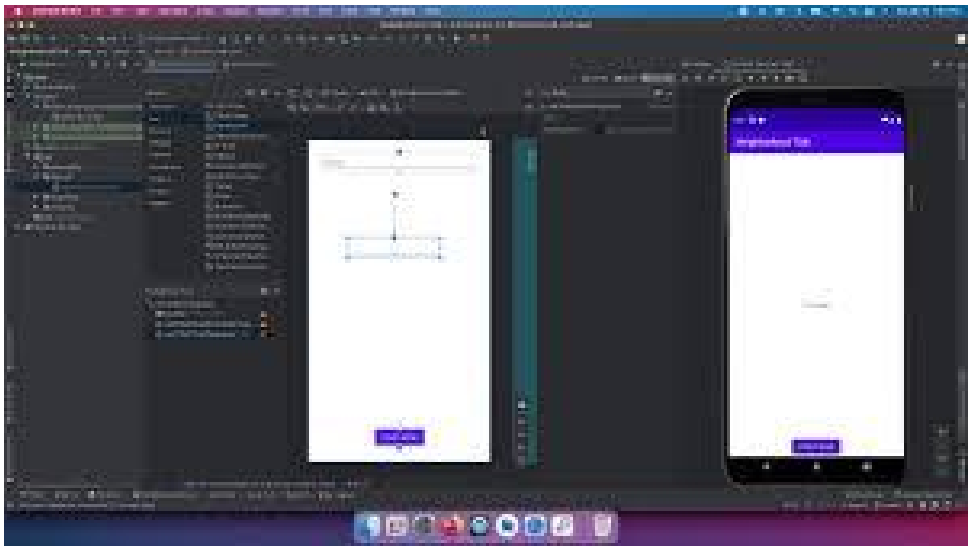
Logiciel utilisés:

PHPSTORM:

WORKBENCH:

POSTMAN:

ANDROID STUDIO:



Arduino

BOOTSTRAP