

Livrables PLants Humidity Sensors

Sommaire

1) Reformulations des besoins	3
2) Spécifications fonctionnelles	3
3) Spécifications techniques	5
4) Maquettage	6
5) Documentation	7
6) Test	11
7) Déploiement	13
8) Conclusion	14

1) Reformulations des besoins

Réalisation d'un outil applicatif permettant d'observer et de calculer en pourcentage le taux d'humidité d'une plante grâce à un capteur ESP 8266.

Le client demande :

- une application desktop multi plateforme
- pouvoir gérer plusieurs capteurs
- afficher les statistiques sur l'évolution de l'humidité

2) Spécifications fonctionnelles

Cette section décrit en détail les fonctionnalités attendues de l'application.

Capteur d'Humidité ESP8266:

Le capteur doit être connecté via Wifi à Internet pour transmettre les données d'humidité.

Chaque capteur possède un identifiant unique de 32 caractères hexadécimaux.

Le capteur sera basé sur un circuit intégré ESP8266 et sera programmable en C++ via l'IDE Arduino et son circuit TPL5111 .

Cloud & Base de Données:

L'API sur le cloud recevra l'identifiant unique du capteur ainsi que la valeur d'humidité.

Les données seront stockées dans une base de données "supabase".

Un mécanisme d'alerte est mis en place pour détecter quand une plante a besoin d'eau.

Application Desktop

L'application permettra de saisir un ou plusieurs identifiants uniques de capteurs.

Elle affichera des graphiques pour visualiser l'évolution du taux d'humidité au fil du temps pour chaque capteur.

L'application affichera une alerte si une plante a besoin d'eau.

3) Spécifications techniques

Front End :

Dans le domaine du Front End, le choix s'est porté sur Angular en raison de sa communauté dynamique et de sa documentation complète. Angular se distingue par sa capacité à créer des "Single Page Applications", améliorant ainsi l'expérience utilisateur.

Le framework CSS Tailwind a été utilisé pour le design de l'application. Grâce à sa simplicité d'utilisation et à sa flexibilité, il permet une personnalisation des composants affichés.

Back End :

L'API repose sur Node.js. Pour garantir une évolution fluide, l'API est hébergée sur un serveur personnel OVH cloud.

OVH propose des services d'hébergement et d'infrastructure dans le cloud. Cela accélère significativement le développement et le déploiement d'applications, tout en offrant une flexibilité accrue pour ajuster les ressources en fonction des besoins changeants.

Base de données :

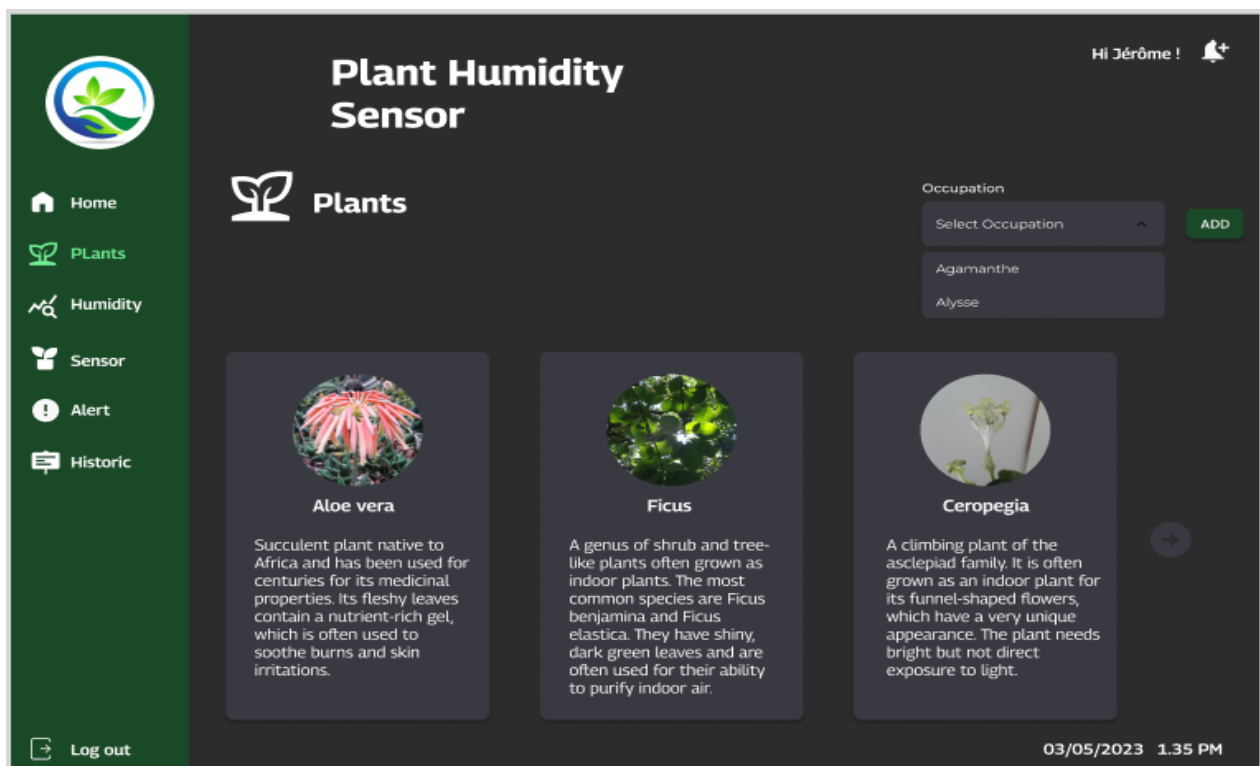
La base de données sélectionnée est Supabase, une solution open source qui facilite l'accès à une base de données Postgres. Cette plateforme offre aux développeurs une manière simple et rapide d'interagir avec la base de données, tout en fournissant des outils complets pour la gestion, la mise à jour et l'analyse des données.

Application desktop:

La demande du client étant une application desktop, nous utilisons le framework **Electron** afin de répondre aux besoins.

Electron est un framework open source qui permet de développer des applications de bureau multiplateformes. Il utilise un modèle basé sur le rendu de pages web dans des fenêtres natives. Electron combine le moteur de rendu Chromium pour la partie client avec Node.js pour la gestion côté serveur, ce qui offre un large éventail de possibilités pour la création d'applications robustes et personnalisées.

4) Maquettage



Voici une partie de la maquette de l'application faite sur figma.

Cliquez sur [Lien figma](#) pour voir les différentes pages maquetter.

5) Documentation

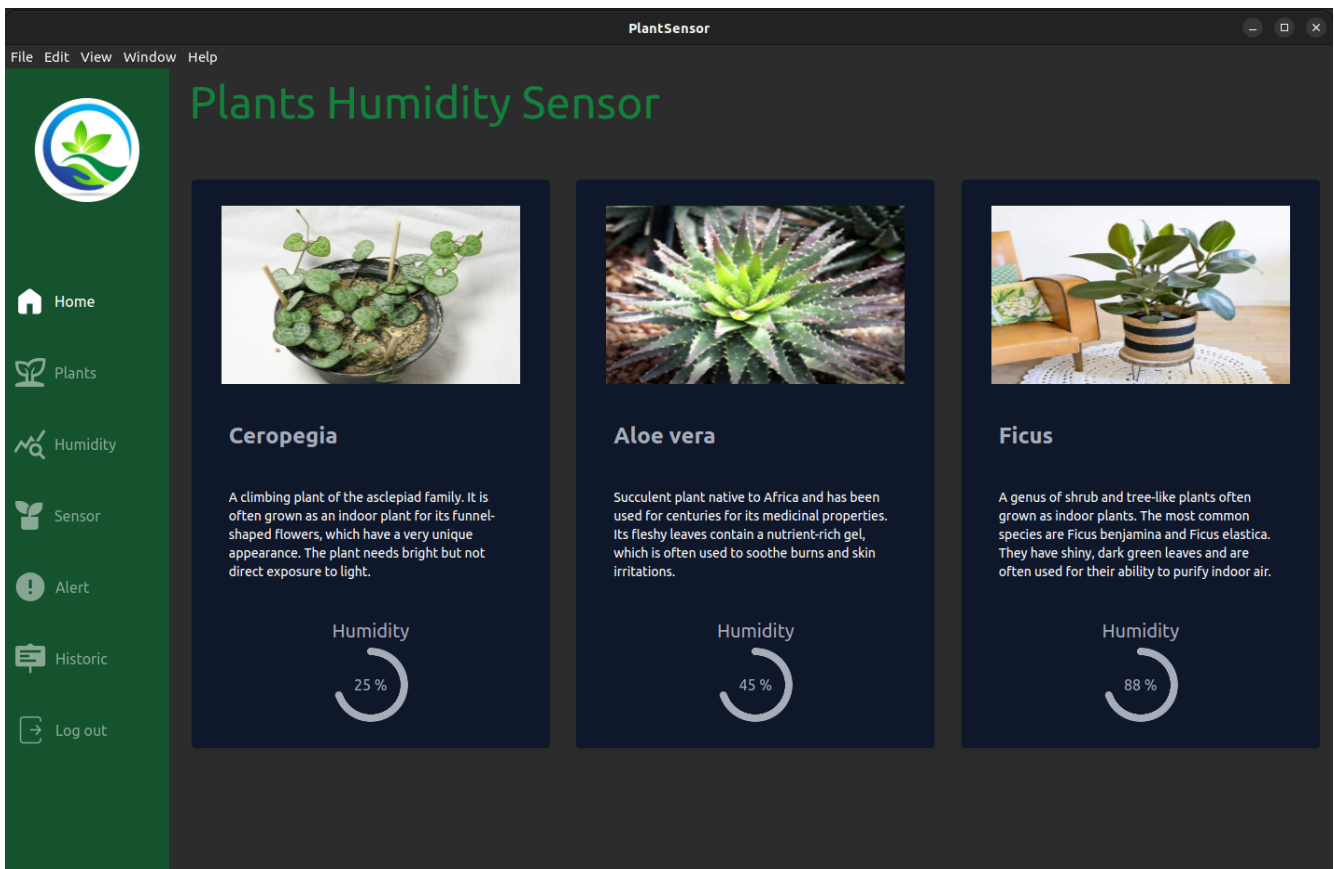
Mise en route de l'application:

Récupérer la partie Front-end (Plants-Sensor) et Back-end (Plants-Sensor-back) dans github.

Faire un git clone des deux dépôts puis un npm i dans la console de VScode. Le back-end est hébergé dans le serveur OVH et fonctionne via “docker”.

Lancer ensuite le Front-end de l'application depuis le terminal de VScode en tapant `ng serve -o` et pour l'avoir en application desktop nous utilisons la commande “`npm run electron`” .

Accueil:



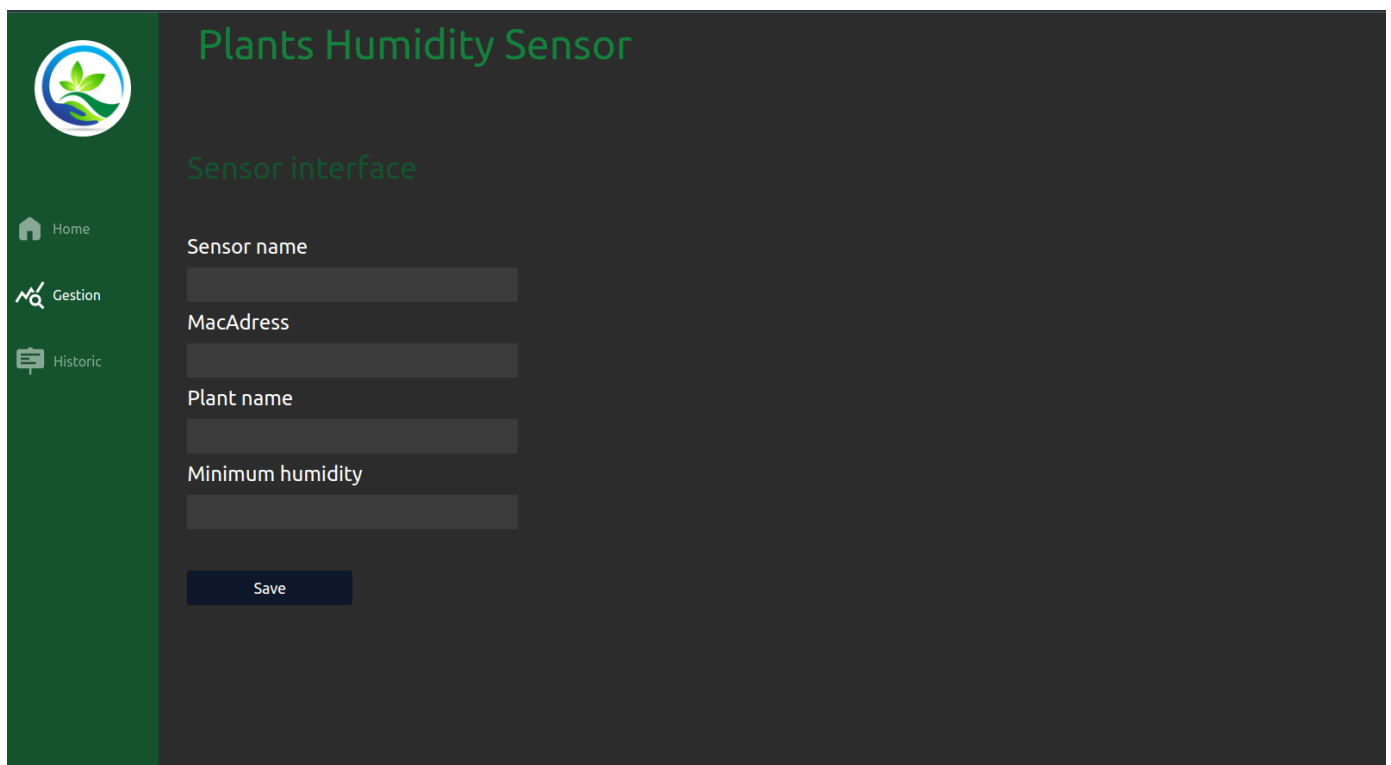
On arrive sur la page d'accueil ou il est possible de voir les différentes plantes, le taux d'humidité de celles-ci et les descriptions.

Au stade actuel de développement, il n'y a qu'un seul capteur en marche.

Gestion:

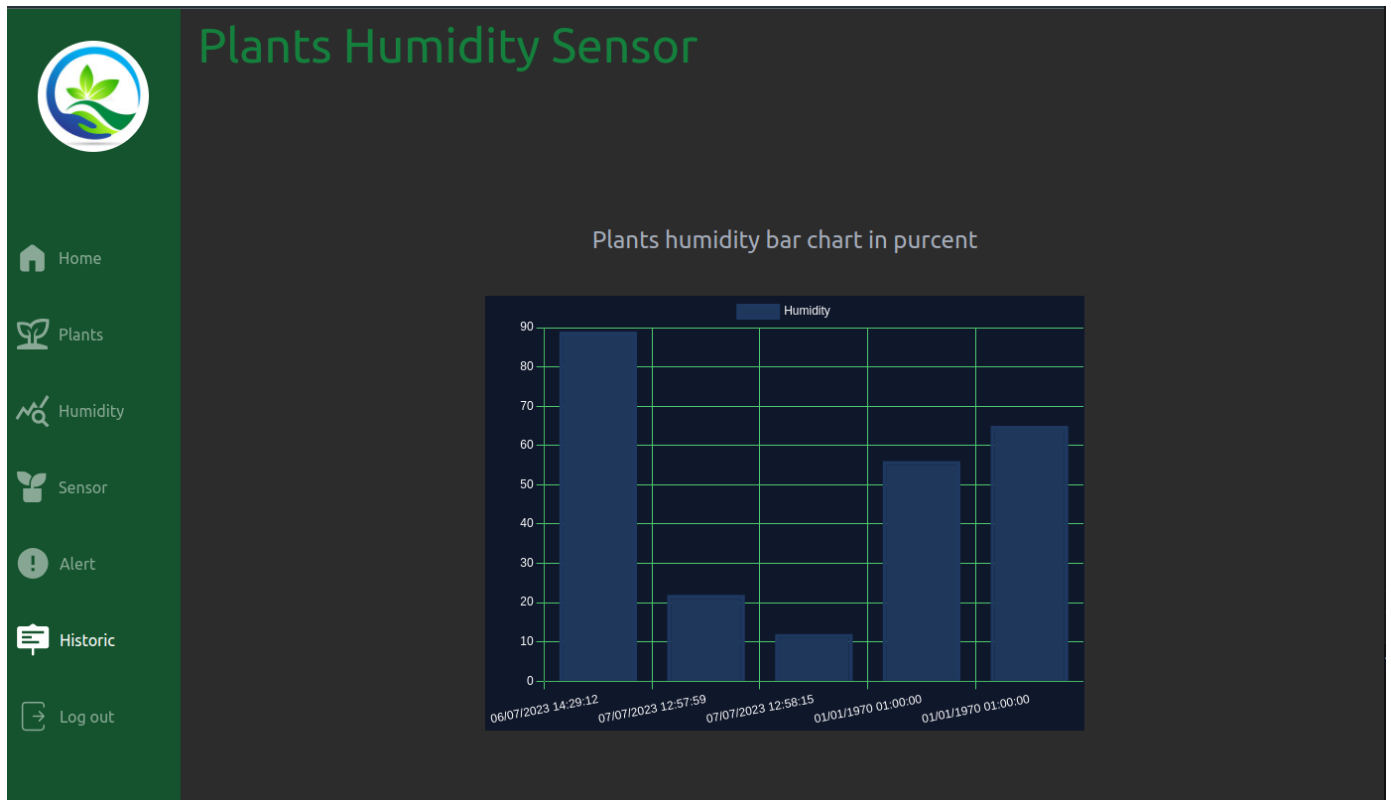
Sur l'interface gestion il est possible de rajouter via un formulaire un nouveau capteur en y renseignant son nom, son adresse mac, le nom de la plante et le taux d'humidité minimum.

Cet enregistrement se fait sur la base de données Supabase.



The screenshot displays the 'Plants Humidity Sensor' web application. On the left is a dark green sidebar with a circular logo at the top containing a stylized green plant. Below the logo are three menu items: 'Home' with a house icon, 'Gestion' with a leaf icon, and 'Historic' with a document icon. The main content area has a dark gray background. At the top, the title 'Plants Humidity Sensor' is written in green. Below it, the section 'Sensor interface' is also in green. The form contains four input fields with light gray labels: 'Sensor name', 'MacAdress', 'Plant name', and 'Minimum humidity'. Each label is positioned above a dark gray rectangular input box. At the bottom of the form is a dark blue button with the white text 'Save'.

Les informations renvoyées sont la date et l'heure, le taux d'humidité actuel (visible sous forme graphique)



Ces données proviennent de la table "historic" , en utilisant un trigger chaque nouveau relevé enregistré sur la table capteur et copié sur cette table . Le capteur est branché directement afin de recevoir les données.

Table editor: Xavier64's Org / Plants-Sensor

schema: public

Tables (2): historic, sensor

	id int8	created_at timestamp	humidity int8	macAdd... t...	+
<input type="checkbox"/>	2	2023-07-07 12:57:59.354638	22	AC:08:FB:D9:9E:01	
<input type="checkbox"/>	1	2023-07-06 14:29:12.541227	89	AC:08:FB:D9:9E:01	
<input type="checkbox"/>	4	NULL	56	1111AC:08:FB:D9:9E:01	
<input type="checkbox"/>	3	2023-07-07 12:58:15.059905	12	AC:08:FB:D9:9E:01	
<input type="checkbox"/>	5	NULL	65	AC:08:FB:D9:9E:01	

Illustration de la base de données supabase avec les tables “historic” et “sensor”.

Definition

The language below should be written in `plpgsql`.

Change the language in the Advanced Settings below.

```

1  create function public.update_historic()
2  returns trigger
3  language plpgsql
4  as $$
5  begin
6      insert into public.historic("humidity","macAdress")
7      values (new."humidity", new."macAdress");
8      return new;
9  end;
10 $$;
11 create trigger on_update_data
12 after update on public.sensor
13 for each row execute procedure public.update_historic();

```

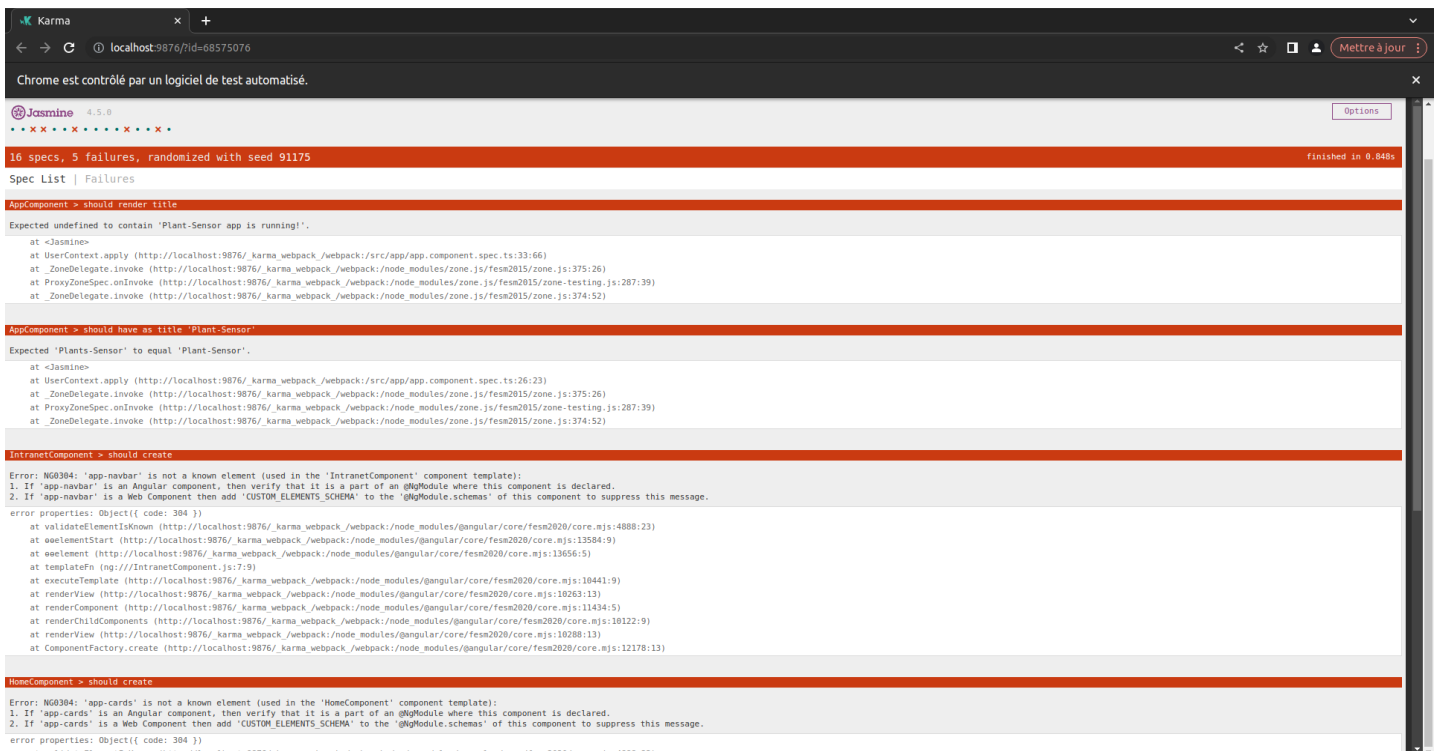
Le trigger permettant à chaque nouveau relevé enregistré sur la table “sensor” d’être copié sur la table “historic”.

6) Test

Les tests sont effectués via **Karma** et l'utilisation de **Sonar Cloud**.

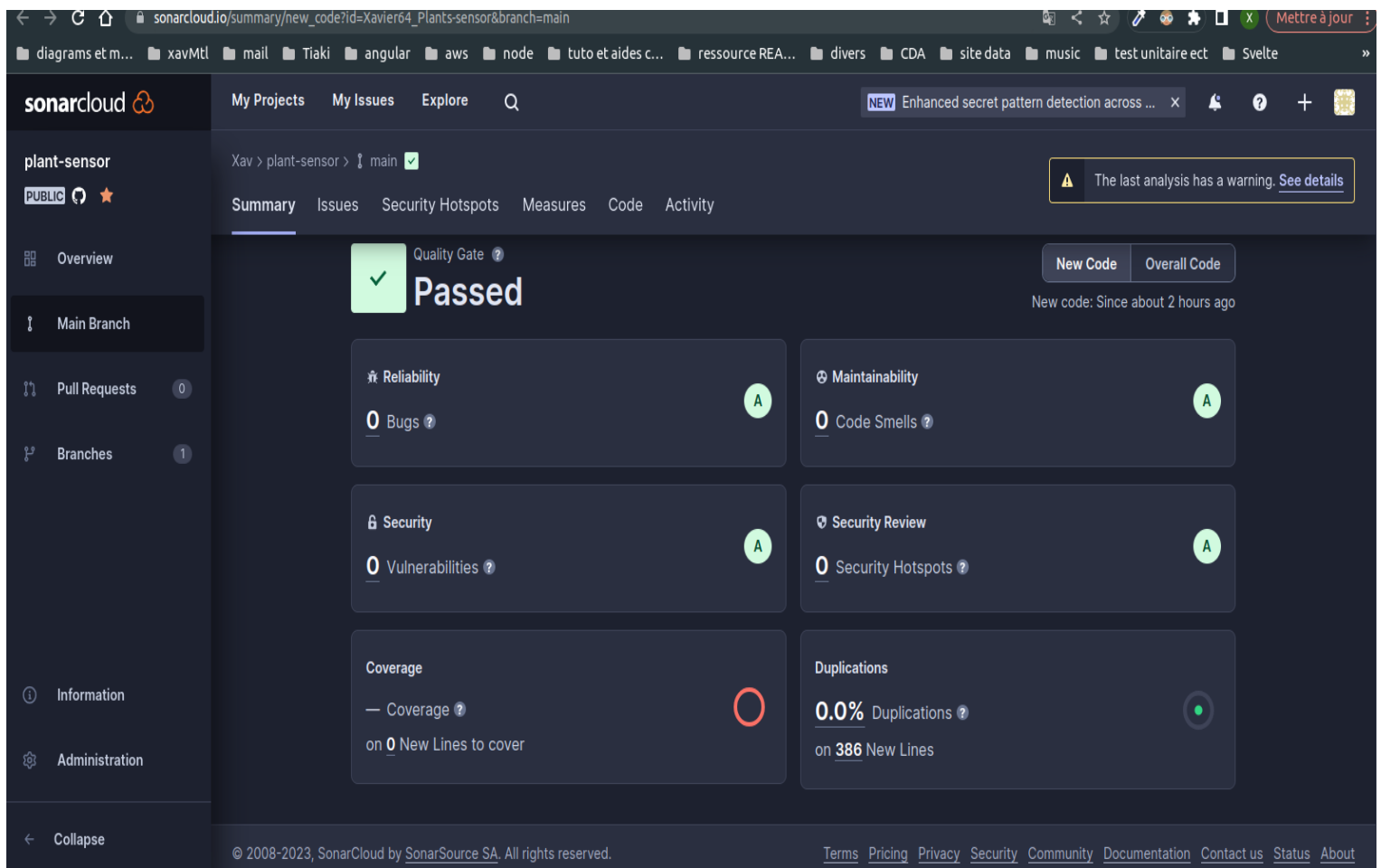
Karma est un outil de test unitaire dans Angular, permettant l'automatisation des tests en exécutant le code de test dans divers navigateurs réels ou virtuels. Intégré souvent avec Angular CLI, il offre la flexibilité de tester dans plusieurs navigateurs, facilite le développement piloté par les tests grâce à l'exécution continue, génère des rapports détaillés et s'intègre aisément aux pipelines d'intégration continue, contribuant ainsi à garantir la qualité et la compatibilité des applications Angular.

Pour illustrer voici un exemple de test fait avec Karma.



Sonar Cloud est une plateforme d'analyse continue du code qui identifie les problèmes de qualité et de sécurité dans les projets logiciels. Grâce à une analyse statique du code source, elle détecte les erreurs, les vulnérabilités et les problèmes de conception, attribuant des scores de qualité aux projets. Intégrable aux flux de travail de développement, Sonar Cloud fournit des rapports détaillés et des tableaux de bord pour améliorer la qualité du code tout au long du processus de développement.

Voici une image pour illustrer l'interface Sonar Cloud permettant ces analyses de codes.



Pour finir, les tests et les analyses se font automatiquement grâce à un workflow.

```
github / workflows /  unittest.yml
1  name: 🚀 Run Unit Tests and SonarCloud Analysis
2
3  on:
4    push:
5      branches:
6        - main
7    pull_request:
8      branches:
9        - main
10
11  jobs:
12    build:
13      runs-on: ubuntu-latest
14
15      strategy:
16        matrix:
17          node-version: [16.x]
18
19      steps:
20        - uses: actions/checkout@v2
21        - name: Use Node.js ${{ matrix.node-version }}
22          uses: actions/setup-node@v2
23          with:
24            node-version: ${{ matrix.node-version }}
25        - run: npm ci
26        - run: npm run test
27
28        - name: SonarCloud Analysis
29          run: |
30            npm install -g sonarqube-scanner
31            sonar-scanner -Dsonar.projectKey=Xavier64_Plants-sensor -Dsonar.organization=xavier64 -Dsonar.host.url=https://sonarcloud.io -Dsonar.sources=. -Dsonar.javascript.lcov.
32          env:
33            SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
34
```

7) Déploiement

Le déploiement se fait à l'aide d'un fichier **docker** pour le **back-end** sur le serveur OVH Cloud.

Docker est une technologie de virtualisation légère qui emploie des conteneurs pour isoler et exécuter des applications avec toutes leurs dépendances. En encapsulant le code et les ressources nécessaires, Docker assure la cohérence entre les environnements de développement et de production, simplifie le déploiement et favorise la portabilité des applications, quel que soit le système d'exploitation.

8) Conclusion

Le projet a été réalisé via l'environnement de développement VScode.

PLant-humidity-sensor est une application programmée en Angular v15.2 pour le front-end, nodeJS v16.18 pour le back-end hébergé sur un serveur OVH Cloud et sa base de données se trouve sur supabase.

Elle est disponible en version logiciel grâce au framework Electron.

Les tests se font grâce à l'outil Angular Karma ainsi que Sonar Cloud et se font automatiquement grâce au workflow inclus dans le projet.

Le déploiement se fait grâce à docker sur le serveur OVH Cloud.

Sources utiles:

Cahier des charges:

[IOT Capteur d'humidité](#)

Trigger supabase:

<https://supabase.com/docs/guides/database/postgres/triggers>

<https://supabase.com/>

SonarCloud documentation:

<https://docs.sonarcloud.io/>

OVH Cloud documentation:

https://help.ovhcloud.com/csm/fr-documentation?id=kb_home