Projet IOT & Réseaux

Piscine / Poulailler Connecté

SELVA Vincent – REJNERI Clément

URL : http://51.210.15.67:8080/
Identifiant : admin
Mot de passe : admin

Table des matières

I. DESCRIPTION DU PROJET

Pour le projet de fin d'année, dans le cours Iot & Réseaux, nous avons eu l'idée de réaliser un jardin connecté pour des particuliers. Le but était de rassembler, sur une interface web, des commandes facilitant la vie des utilisateurs.

Pour aller plus loin et approfondir notre idée initiale, nous devions rajouter une fonctionnalité. Pour cela, nous avons pensé rajouter un système de connexion sur l'interface web pour sécuriser le tout. L'utilisateur en possession du projet devra avoir un compte et se connecter pour pouvoir utiliser les fonctionnalités mise à sa disposition.

L'utilisateur pourra contrôler depuis une interface web différentes choses. La personne en possession de l'interface web, pourra se connecter pour avoir accès à différentes informations.

Sur la page des "graphiques", l'utilisateur aura accès à la température de sa piscine en temps réel ainsi que d'une pièce de sa maison.

Il y aura une page pour la piscine, et une page pour le poulailler. Sur chacune d'elle il pourra ouvrir ou fermer manuellement sa piscine ou son poulailler. Ce système de fermeture et d'ouverture sera aussi automatique avec le levé et le coucher du soleil.

De plus, il pourra allumer et éteindre depuis l'interface web des lumières présente dans son jardin, grâce à un bouton.

La particularité de notre projet et de mettre en place de la sécurité web avec une interface de connexion. L'utilisateur aura besoin d'un identifiant et d'un mot de passe créé préalablement pour utiliser les fonctionnalités

II. CONCEPTION

A. TECHNOLOGIES & LANGAGES UTILISÉES

Dans le développement de notre projet, nous avons utilisés plusieurs fonctionnalités. Des fonctionnalités dans le langage utilisé par l'interface Arduino et en HTML / CSS / Vuejs et JavaScript.
Au début de la création de notre projet, nous avons commencé le développement en local avec un serveur local et une base de données local.
Par la suite nous avons décidé de prendre un hébergement en ligne pour travailler en simultané et avoir une seule base de données.

1. HÉBERGEMENT

Concernant l'hébergement, nous avons utilisés et testés plusieurs hébergements (voir section - *Problèmes rencontrées*)

Notre solution finale a été de partir sur une solution cloud avec OVH.
La condition la plus importante était l'accès SSH au serveur. Etant donné que nous devons exécuter le fichier Node js et le run serve pour avoir une application web fonctionnelle.
Nous avons choisi un VPS (Serveurs privés virtuels) pour les performances, la simplicité et bien sur l'accès SSH.
Après avoir commandé le VPS, une adresse IP nous a été attribué, nous avons donc effectué toutes les modifications nécessaires dans le code de l'application en créant une url pour le Node http://51.210.15.67/, en spécifiant le port 8080 étant donné que c'est une application vuejs.

2. BASE DE DONNÉES

Nous avons utilisé la base de données en ligne gratuite MongoDB Atlas pour pouvoir stocker toutes les données de l'ESP. Après avoir mis en place la base de données (création d'un projet et d'un cluster), le premier point a été la partie "Network Access". La partie "Network Access" permet d'ajouter des adresses IP "Whitelist", c'est-à-dire d'autoriser certaines adresses IP à publier sur notre base de données. Nous avons donc autorisé nos 2 adresses IP pour pouvoir ainsi publier tous les deux sur la base de données en ligne du projet.

3. VUEJS

Nous avons choisi d'utiliser Framework vuejs pour plusieurs points.
Premièrement, vue js est un Framework frontend, cela nous permet de gagner du temps sur la réalisation et l'intégration de contenus (textes, pages, etc…).
Vuejs est un des Framework les plus légers existant, cela se ressent principalement au chargement et changement des pages ce qui est important pour l'utilisateur.
Puis, ce qui nous a fait choisir vue js est son architecture via des composants. Cela permet un meilleur découpage d'une application. De plus, le plus gros avantage est que les composants peuvent ensuite être réutilisés, ailleurs dans l'application ou même dans un autre projet. Un composant est décomposé en 3 parties : le template, le JS, le style (css).

4. ARDUINO

Sur Arduino, nous avons aussi dû installer des bibliothèques tel que "WiFi.h", "OneWire.h" ou encore "DallasTemperature.h". Ces bibliothèques énoncées nous permettent de faire fonctionner le programme.
La librairie « DallasTemperature.h » a été spécialement conçue pour le capteur que nous utilisons et nous propose donc des fonctions simplifiées. Ce qui permet de simplifier au maximum le code.
La librairie « OneWire.h » est une librairie dédiée pour le capteur de température.

Notre projet consiste à la réalisation d'un programme contrôlable depuis une interface web. C'est pour cela que nous avons dû connecter la plaquette au Wifi.
Pour la partie connexion, nous avons utilisé la fonction "connect_wifi()".

Elle permet de gérer la connexion de la plaquette ESP 32. Nous avons dû renseigner le nom de la box internet ainsi que le mot de passe.

B. APPORT TECHNIQUES

Le principal atout de notre projet est la réutilisabilité.
Sur l'interface web, nous avons décidé de créer différentes pages.
Une pour le poulailler qui permettra de fermer ou d'ouvrir le poulailler en fonction de la journée, le matin ou le soir.

Une pour la piscine, qui affichera la température de l'eau et qui permettra aussi d'ouvrir ou de fermer le volet.
Il y aura une page avec les graphiques de la température et de la luminosité en temps réel.

L'interface web est disponible grâce à l'hébergement sur l'URL : http://51.210.15.67:8080/

Pour la connexion, nous utilisons PuTTY. PuTTY est un émulateur de terminal doublé d'un client pour les protocoles SSH.
Pour avoir accès au site internet, nous devons lancer deux terminales. Un ou nous faisons la commande "npm run serve" permettant la connexion au server. Et un ou nous faisons "node node_lucioles_v2.js" permettant le chargement des données température et luminosité.

Sur les pages Piscine et Poulailler nous avons rajouté un système d'invité. L'utilisateur a la possibilité d'ajouter une nouvelle adresse mac ce qui lui permet de visualiser ses données.

1. PISCINE

Etant donné que la piscine est du confort pour l'utilisateur, nous avons décidé de mettre en place l'ouverture et la fermeture de la piscine de façon manuelle. L'utilisateur à la main sur sa piscine. Il n'y a pas d'ouverture ou de fermeture en fonction de la luminosité environnante ou de la température.
Sur la page piscine, nous avons installé différents boutons. En arrivant sur la page web, la piscine est forcément fermée.

Nous avons créé une fonction lastValue() permettant de récupérer la dernière valeur de la température et de la luminosité. La fonction lastValue() permet donc de récupérer la dernière valeur dans la liste des données. Nous avons défini une adresse mac pour la page Piscine permettant d'avoir que ces données.

2. POULAILLER

Pour le poulailler, l'ouverture et la fermeture se fait de manière automatique en fonction de la luminosité environnante. S'il fait jour le matin, le poulailler va s'ouvrir de manière à faire sortir les poules. Et le soir lorsqu'il fait nuit le poulailler va se fermer. De cette façon les poules sont en sécurité la nuit. L'utilisateur n'a plus à se soucier s'il a bien pensé à fermer le poulailler.
Pour cette page, nous avons créé aussi la fonction lastValue() mais nous avons récupéré les valeurs d'une adresse mac définit dans la page Poulailler.vue.

3. GRAPHIQUES

Sur la page des graphiques, nous avons accès aux données luminosités et température en temps réel. Chaque donnée est rentrée et affichée sous forme d'un graphique.
Pour ce qui est des fonctionnalités, nous avons utilisés la fonctionnalité highchart.
Sur la page graphique, nous avons voulu afficher sur un grand lapse de temps deux courbes pour la température et deux courbes pour la luminosité. Une correspond à la piscine et l'autre au poulailler.
L'utilisateur a la possibilité de zoomer sur la partie qui l'intéresse pour plus de précision.

4. CONNEXION

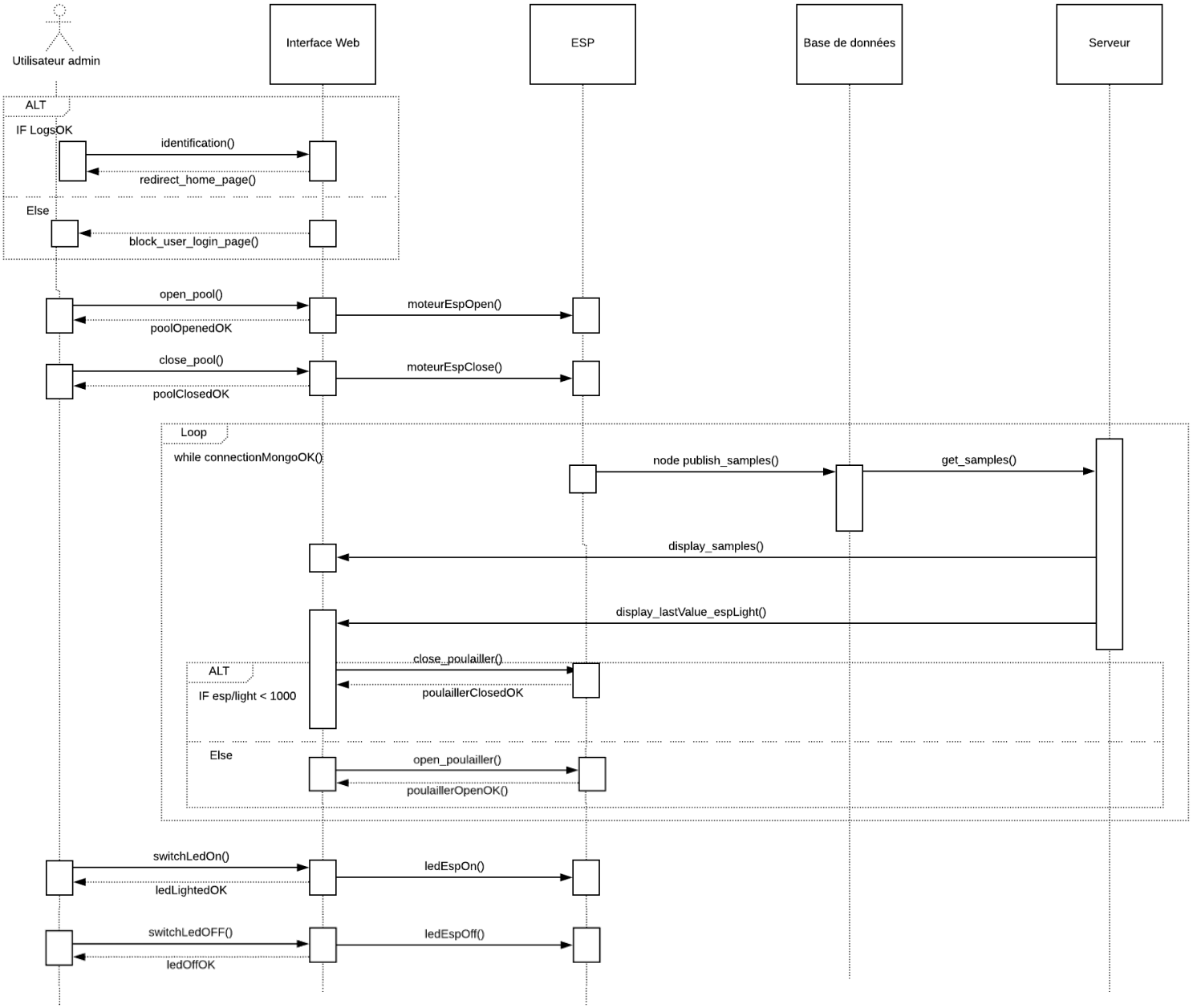
Après avoir développé les principales lignes de notre projet, nous avons réfléchi à l'aspect "Sécurité" car tout le monde pouvait accéder à notre URL et donc déclencher toutes les actions qu'il souhaitait (ouvrir/fermer le volet ou la porte, …).
Nous avons donc réalisé une interface de connexion. Sur celle-ci l'utilisateur peut se connecter en tant qu'administrateur pour pouvoir avoir le contrôle de toutes les fonctionnalités. Il peut aussi se connecter en tant que client pour avoir accès uniquement à la page graphique et juste voir les données.

5. AJOUT INVITÉ

Sur la page graphique, l'utilisateur a la possibilité d'ajouter une nouvelle personne en renseignant son adresse mac. De ce fait, une troisième courbe apparaîtra et l'utilisateur aura accès à la température et à la luminosité d'une nouvelle pièce de sa maison.
Il pourra aussi s'allumer et éteindre une led grâce au bouton ping situé à côté de son adresse mac sur la page graphique. Idem pour les pages Piscine et Poulailler.

1.

III. DIAGRAMME DE SÉQUENCES



Sur notre diagramme de séquence nous avons représenté 5 objets avec un acteur principal, l'utilisateur (admin). Lorsque l'utilisateur arrive sur l'interface web, il a l'obligation de se connecter, si les identifiants sont bons alors il peut accéder à l'interface sinon il reste bloqué sur la page de connexion.
L'utilisateur a la possibilité de déclencher plusieurs actions comme l'ouverture et la fermeture du volet de la piscine ou allumer les leds de la piscine.
Concernant les données, sur Arduino nous lançons l'ESP32 qui fait une boucle et publie sur le broker MQTT. Apres, la fonction get_samples() fait le lien entre l'interface web et la base du node js. Elle opère une requête get sur le node js pour récupérer les samples (lumière ou température) de l'ESP. Par la suite, les données sont publiées sur la base de données mongo et pour finir elles sont affichées sur l'interface web pour être visible par l'utilisateur connecté.

IV. PROBLEME RENCONTRES

A. HÉBERGEMENT

Nous avons testé différents types d'hébergement avant d'arriver à quelques choses de fonctionnels pour notre projet.

- Amazon
- Heroku
- OVH

Au début, nous avons utilisé la solution Heroku mais nous avons rencontrés différents problèmes lors de chaque modification dans le code de notre projet. Nous devons redéployer l'application sur Heroku à chaque modification. Ce n'était pas optimisé étant donné que nous étions deux à travailler en même temps sur le projet.
Après discussion avec monsieur Menez, nous avons regardé la solution Amazon, mais celle-ci semblait difficile car elle n'était pas adaptée.
Pour finir, nous nous sommes penchés vers une solution payante, avec l'hébergeur OVH. Nous avons choisi cette solution car nous la connaissons et nous l'avons déjà utilisé dans nos alternances respectives.

Sur OVH, nous avons choisi un VPS (serveur virtuel privé). Cette version regroupe une mémoire de 2 Go, un stockage de 20 Go SSD SATA et une bande passante publique (100 Mbit/s) ce qui est largement suffisant pour notre application.
L'atout principal avec cette version, et que nous avons un accès SSH pour pouvoir exécuter le "node" et le "run serve" directement sur le serveur.

B. CHARGEMENT DES DERNIÈRES DONNÉES

Après avoir créé nos différentes pages, nous voulions récupérer la dernière valeur sur le graphique pour l'afficher sur une autre page. Il fallait créer une nouvelle fonction dans le fichier Piscine et Poulailler permettant d'appeler les valeurs venant de la page Graphiques. Par exemple, récupérer la dernière valeur température pour donner à l'utilisateur la température de sa piscine, sur la page piscine et non la température de sa piscine il y a 2 heures.

Pour cela, nous avons réalisé une fonction lastValue() qui récupère les derniers éléments du tableau de données sur le "path" "esp/temp" ou "esp/light". Cette fonction a donc besoin en paramètre de l'adresse mac de l'utilisateur, de l'url du node et du path.
Ce qui nous donne les requêtes suivantes :

- http://51.210.15.67:3000/esp/light?who=30:AE:A4:86:CA:7C
- http://51.210.15.67:3000/esp/temp?who=30:AE:A4:86:CA:7C

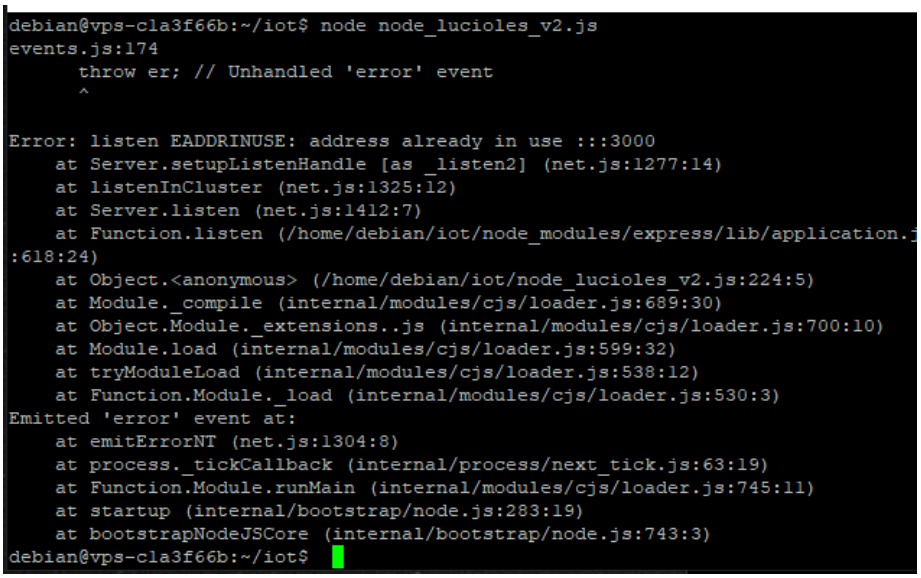
Pour plus d'informations sur ces requêtes, nous nous sommes servis de la console web avec l'onglet "Network". L'onglet "Network" offre la possibilité de voir toutes les informations sur nos requêtes (status, type, adresse mac, protocole, etc…)

Sur notre application, nous utilisons un stockage côté client qui est un moyen rapide et efficace. Les données sont donc stockées sur le navigateur au lieu d'aller chercher ces données sur le serveur. Nous avons donc utilisé le « Web Storage » grâce à la fonction localStorage().

C. CHARGEMENT DU NODE

La dernière grosse difficulté que nous avons rencontrée est la suivante.
Certaines fois, une fois connecté en SSH au serveur, au moment de lancer la commande node, une erreur apparaissait pour nous dire que le port 3000 était déjà utilisé (voir Screenshot ci-dessous).

On ne pouvait donc pas exécuter notre fichier node et donc notre interface web n'affiche aucune donnée.



Pour cela nous avons exécuté la commande : lsof -i:3000 qui permet de faire un ListOf des processus en cours sur le port 3000 et nous afficher le PID pour ensuite effectuer un kill -9 sur le PID en question.

- sudo apt-get install lsof
- lsof -i:3000

La commande nous affiché donc le résultat suivant :

$ lsof -i:3000
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
lsof 13402 zero 4u IPv4 2847851 0t0 TCP *:3000 (LISTEN)

Ce qui nous intéresse ici est donc le PID 13402, grâce à nous cours de l'année dernière nous avons exécuté la commande suivante :

$ kill -9 13402

V. MISE EN ROUTE

Pour ce qui est de la mise en route du projet, l'utilisateur n'a pas besoin de télécharger le projet sur le serveur ou sur GitHub.
Uniquement à se connecter à l'URL : http://51.210.15.67:8080/. En arrivant sur l'interface web, il vous sera demandé de rentrer un identifiant et un mot de passe pour accéder aux fonctionnalités.
Les identifiant "admin" sont :

- Login : admin
- Password : admin

Les identifiant "client" sont :

- Login : user
- Password : user

Vous aurez donc accès à notre projet sans rien télécharger et exécuter de votre côté grâce à notre déploiement en ligne.

Si par ailleurs, vous souhaitez télécharger notre projet pour regarder le code, vous devez le télécharger depuis notre dépôt GitHub :
https://github.com/VinceSelva/projet-iot-rejneri-selva.

D. Mise en ligne

Pour la mise en ligne, nous avons utilisé la commande pm2 permettant de lancer le serveur pour une durée indéterminée. Pour cela nous avons du build le projet pour garder la dernière version.
$ sudo npm install pm2 -g
$ npm run build
$ pm2 start projet-vue

Autres :

Attention, si vous télécharger le code, le projet ne sera pas fonctionnel en local sur votre ordinateur. Il faudra remplacer certaines parties du code, notamment toutes les urls node http://51.210.15.67:3000 et mettre à la place http://localhost:3000.

Puis deux commandes seront nécessaires :

1. Dans le dossier projet-iot-rejneri-selva/projet-vue/
    - Exécuter la commande "npm run serve" pour que l'application tourne sur votre serveur local : http://localhost:8080/

2. Dans le dossier racine projet-iot-rejneri-selva/

- Exécuter la commande “node node_lucioles_v2.js”