



# Stage de première année

## I. Présentation globale

- A. L'objectif du stage
- B. Organisations et directives du projet
- C. Mon environnement

## II. Les technologies utilisées

- A. Vue.js
- B. Kuzzle

## III. Le projet

- A. L'affichage des données
- B. Partie administration
- C. Authentification
- D. Création des capteurs, automatisation
- E. SMTP
- F. La sécurité

## IV. Conclusion

## I. Présentation globale

### A. L'objectif du stage

Loca Service utilise des capteurs pour leurs meubles réfrigérées, mais aussi pour son smart building, c'est à dire mieux gérer les ressources utilisées par l'entreprise afin d'assurer une consommation d'énergie durables. Pour suivre en temps réel les données de ces capteurs, ils utilisent une plateforme en ligne, IceConnect. Mais les prix d'abonnements peuvent être assez élevés à la longue, ainsi leur ai venu l'idée d'expérimenter une solution interne, développé en interne, permettant de consulter les données des capteurs, surtout pour ceux dans le cadre du smart building, ou des capteurs en test. Ainsi, ils m'ont donné la mission de développer cette solution sous forme d'un site web.

### B. Organisations et directives du projet

Pour développer le site, l'entreprise m'a imposé le choix du framework Vue.js pour la solution web, de Kuzzle pour la gestion des trames HTTP reçues en backend. Mais aussi, le projet devait être installé sur une VM Linux Debian 12 sur les serveurs de l'entreprise. De ce fait, le projet pouvait être sauvegardé et hébergé en interne, pour recevoir les données des capteurs connectés au réseau.

Avec ces directives, j'ai été laissé en totale autonomie sur la gestion du projet et l'apparence que pouvait prendre la solution, seules quelques directives spécifiques m'ont été données quant au fonctionnalité à faire apparaître sur le site, que voici:

### **Frontend:**

- Afficher la date / heure / minute / seconde de la réception du message
- Afficher le facteur de qualité réseau
- Ajouter un affichage pour le capteur d'ouverture et fermeture des portes
- Créer un graphique (axe x : temps / axe y : donnée)
- Améliorer le graphique pour le rendre paramétrable en temps

### **Backend:**

- Gestion des requêtes http :
  - Créer un filtre qui reconnaît les trames http en provenance d'Actility.
  - Créer un gestionnaire d'erreurs http.
- Ajout d'objets connectés :
  - Ajouter le décodeur du capteur d'ouverture et fermeture de portes.
- Améliorations documents :
  - Enregistrer la date / heure / minute / seconde de la réception du message
  - Enregistrer le facteur de qualité réseau

Avec ce cahier des charges, j'ai décidé de me créer un planning pour mieux m'organiser dans le développement de la solution:

Semaine	Période	Tâches Planifiées	Tâches Réalisées
1	13/05 - 17/05	Prise en main des outils : Vue.js, Kuzzle, Chart.js, NodeMailer. Lecture des documentations.	Tests initiaux, mise en place de l'environnement de développement.
2	20/05 - 24/05	Compréhension des trames HTTP et récupération des données des capteurs.	Création des premières requêtes Kuzzle pour récupérer température et humidité.
3	27/05 - 31/05	Affichage des premières données et mise en place de l'abonnement temps réel.	Ajout des abonnements sur Vue.js pour actualisation en direct. Tests et corrections.
4	03/06 - 07/06	Développement du routage et navigation. Mise en place d'un menu latéral (SideMenu).	Implémentation de Vue Router et création des différentes pages de capteurs.
5	10/06 - 14/06	Amélioration de l'affichage des données : couleurs, icônes et ergonomie.	Ajout de couleurs conditionnelles, réorganisation des éléments UI.
6	17/06 - 21/06	Développement des graphiques interactifs avec Chart.js.	Création des courbes pour température et humidité, ajout du zoom et filtrage temporel.
7	24/06 - 28/06	Finalisation du projet, optimisation du code, rédaction de la documentation.	Nettoyage du code, amélioration de la responsivité, corrections de bugs et tests finaux.

Tous les lundi, nous faisions un point avec toute l'équipe informatique de l'entreprise pour parler des projets en cours, de la vie de l'entreprise. Tous les vendredi soirs, nous faisions un point avec les personnes en charge des capteurs et du responsable informatique sur l'avancement du site, des axes d'améliorations possibles, de ce fait je pouvais avoir des retours et ainsi mieux orienter mon développement, mais surtout avoir un retour direct de l'utilisateur final, ce qui m'a grandement aidé à bien diriger le développement de l'application.

## C. Mon environnement

J'étais intégré dans l'équipe d'informatique de l'entreprise, mais plus précisément dans la division génies électriques, qui se composent de deux personnes spécialisées dans la conception des capteurs de l'entreprise. En étant à leur contact j'ai pu leur poser directement les questions nécessaires sur les capteurs et leurs fonctionnement. Notamment sur la manière d'extraire les données des trames HTTP de ces derniers. Ils m'ont été d'une aide très précieuse lors de la conception du backend Kuzzle pour recevoir et décoder les trames HTTP.

# II. Les technologies utilisées

## A. Vue.js

Le projet utilise Vue.js qui est un framework JavaScript permettant de construire des interfaces web dynamiques de manière rapide et organisée. Il repose principalement sur une approche **composentielle** : chaque partie de l'interface est divisée en **composants** autonomes, chacun ayant son propre code HTML, CSS et JavaScript.

Le fonctionnement de Vue.js repose sur trois concepts techniques :

- **Data Binding (liaison de données)** : Vue connecte automatiquement les données JavaScript à l'affichage HTML. Si la donnée change, la vue est mise à jour instantanément sans recharger la page.
- **Réactivité** : Vue observe les données déclarées dans les composants. Lorsqu'une donnée est modifiée, seul ce qui est concerné dans le DOM est mis à jour, grâce à un **DOM virtuel** optimisé.
- **Directives** : Vue utilise des attributs spéciaux dans le HTML (ex : `v-if`, `v-for`, `v-bind`, `v-model`) pour rendre dynamiques les comportements classiques (affichage conditionnel, boucles, liaisons d'attributs, formulaires, etc.).

Vue.js permet également :

- La **communication entre composants** (via des props et événements).
- La **navigation entre pages** avec Vue Router.
- La **gestion d'état centralisée** avec Vuex pour les applications complexes.

Ne connaissant pas le framework, j'ai dû apprendre à le prendre en main et pour se faire j'ai réalisé un mini projet qui permet de créer des capteurs et de les ajouter dans un tableau en dessous et les supprimer également:

**Enregistrer un capteur**

Nom Capteur	DevEUI	Type Capteur	
Capteur Température	FAEB45C1F999A033	temperature	
Test	DEVEUITEST	temperature	
Ouverture	Test	ouvertFerm	

Réaliser ce petit projet m'a permis de comprendre la gestion des états, la manière de fonctionner du framework et la structure des documents. J'ai réalisé ce petit projet à l'aide de la documentation officielle de Vue.js et du forum Stack Overflow.

## B. Kuzzle

Kuzzle est une **plateforme backend** conçue pour simplifier le développement d'applications web, mobiles et IoT en fournissant des services prêts à l'emploi comme :

- **Base de données temps réel** (généralement Elasticsearch).
- **API multi-protocoles** (HTTP, WebSocket, MQTT) pour interagir avec les données en temps réel.
- **Gestion des utilisateurs et authentification** (authentification locale, OAuth, SSO, etc.).
- **Système de permissions** finement configurable pour sécuriser les accès.

Techniquement, Kuzzle agit comme un serveur intermédiaire entre l'application cliente et la base de données :

- Il traite les **requêtes CRUD** (Create, Read, Update, Delete) et **notifie les clients** en temps réel lors des changements de données.
- Il supporte les **filtres d'abonnement** (ex: recevoir uniquement les événements qui correspondent à certaines conditions).
- Il est **extensible** grâce à des **plugins** permettant d'ajouter des fonctionnalités côté serveur.
- Il propose un **système d'abonnement temps réel** très pratique : il permet à une application de s'abonner à une base de données et d'être informée en direct dès qu'un document est ajouté, modifié ou supprimé. Dans le cadre de mon stage, j'ai utilisé ce système pour **suivre en temps réel les données issues de capteurs**, et les afficher dynamiquement dans l'application cliente dès qu'une nouvelle valeur était reçue.

Je ne connaissais pas cette plateforme avant mon stage, c'est pourquoi j'ai dû me pencher sur l'utilisation de cette dernière mais aussi d'Elasticsearch n'étant pas familier avec cette méthode. Elasticsearch est un **moteur de recherche et d'indexation** de données. Il permet de **stocker**, **chercher** et **analyser** de grandes quantités de données très rapidement. Les données sont organisées sous forme de **documents JSON** dans des **index** (équivalent à des bases de données), et peuvent être recherchées avec des requêtes très flexibles.

J'ai de ce fait réalisé un petit document me servant de référence lorsque je devais faire des requêtes, ce dernier contient les requêtes les plus basiques et utilise pour réaliser ce projet.

Voici par exemple deux fichiers JSON contenant des données de capteurs extraites avec le backend de l'appli:

```

▼ □ M3ikXZAB42UrKSq91beL updated
  ▼ Object
    type: "capteurPorte"
    deviceId: "P370F7"
    devEUI: "FAEB45C1F999A045"
    ▼ state: Object
      state: "Ouverte"
    ▼ dateFormat: Object
      dateFormat: "28-06-2024 09:08:21"
    ▼ reseau: Object
      reseau: -71
    ▼ ouverture: Object
      ouverture: 4
    ▼ moyenne: Object
      last: false
    ▼ _kuzzle_info: Object
      author: null
      createdAt: "28/06/2024, 09:00:02"
      updatedAt: "28/06/2024, 09:08:20"
      updater: null

  ▼ □ 7niPVJAB42UrKSq9LrZU
  ▼ Object
    type: "capteurTemp"
    deviceId: "Proto2.1_TH_4"
    devEUI: "FAEB45C1F999A033"
    ▼ humidity: Object
      humidity: 66.9
    ▼ temperature: Object
      degree: 22.3
    ▼ dateFormat: Object
      dateFormat: "26-06-2024 14:39:48"
    ▼ reseau: Object
      reseau: -69
    ▼ moyenne: Object
      last: true
    ▼ alert: Object
      tempMax: 26
      tempMin: 16
      compteur: 0
    ▼ minMax: Object
      minTemp: 19.9
      maxTemp: 23
      minHumi: 66.9
      maxHumi: 76.3
    ▼ _kuzzle_info: Object
      author: null
      createdAt: "26/06/2024, 14:39:48"
      updatedAt: "27/06/2024, 09:29:29"
      updater: "Maxime"

```

On stocke ces documents dans des collections, il y a trois grande collections, celle qui contient les utilisateurs, celle des capteurs enregistrés et celle des mesures reçues.

The screenshot shows the Kuzzle interface for the 'iot-data' index. On the left, a sidebar lists 'All indexes' and the 'iot-data' index, which contains three collections: 'capteurs', 'groupes', and 'measures'. The main area displays these collections with columns for 'Name', a checkbox, and edit/delete icons.

Name			
capteurs	<input type="checkbox"/>		
groupes	<input type="checkbox"/>		
measures	<input type="checkbox"/>		

Voici par exemple comment s'organise la collection des utilisateurs:

The screenshot shows the Kuzzle interface for the 'Users' collection. The sidebar has links for 'Users', 'Profiles', and 'Roles'. The main area shows a list of users with their roles. Each user entry includes a checkbox, a 'consulter' role indicator, and edit/delete icons.

User	Role		
UserOrléans	consulter		
UserRennes	consulter		
UserParisOuest	consulter		
UserParisEst	consulter		
UserNancy	consulter		
UserLyon	consulter		
UserLille	consulter		
UserCarcassonne	consulter		
UserBordeaux	consulter		

Ces derniers contiennent les informations les concernants mais les mots de passe sont bien entendu stocké après encryption.

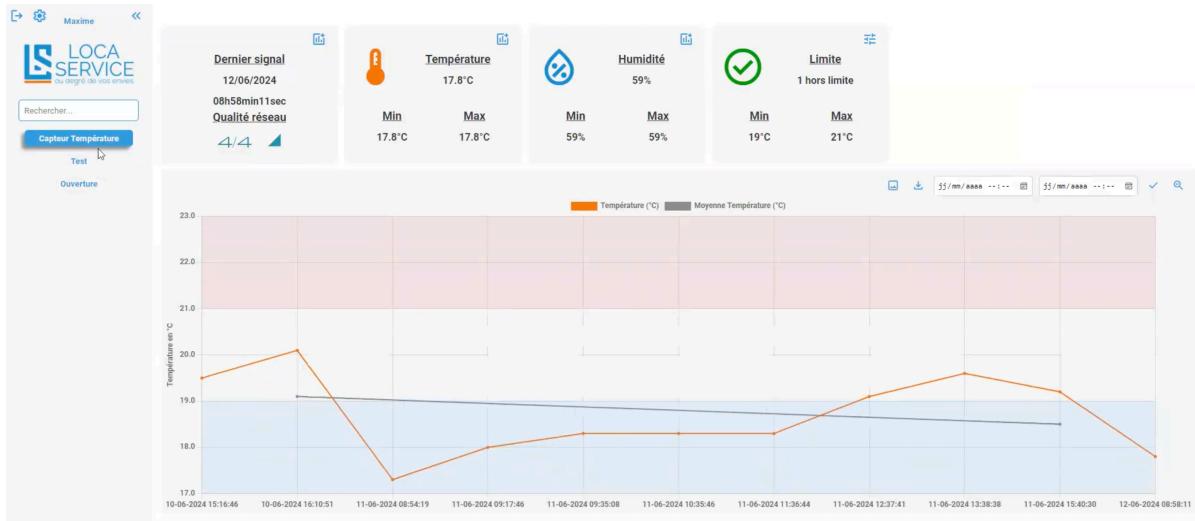
## III. Le projet

Le projet contient de nombreuses fonctionnalités (voir le document récapitulatif), mais les plus importantes sont l'affichage des données enregistrées et l'administration des capteurs et des utilisateurs.

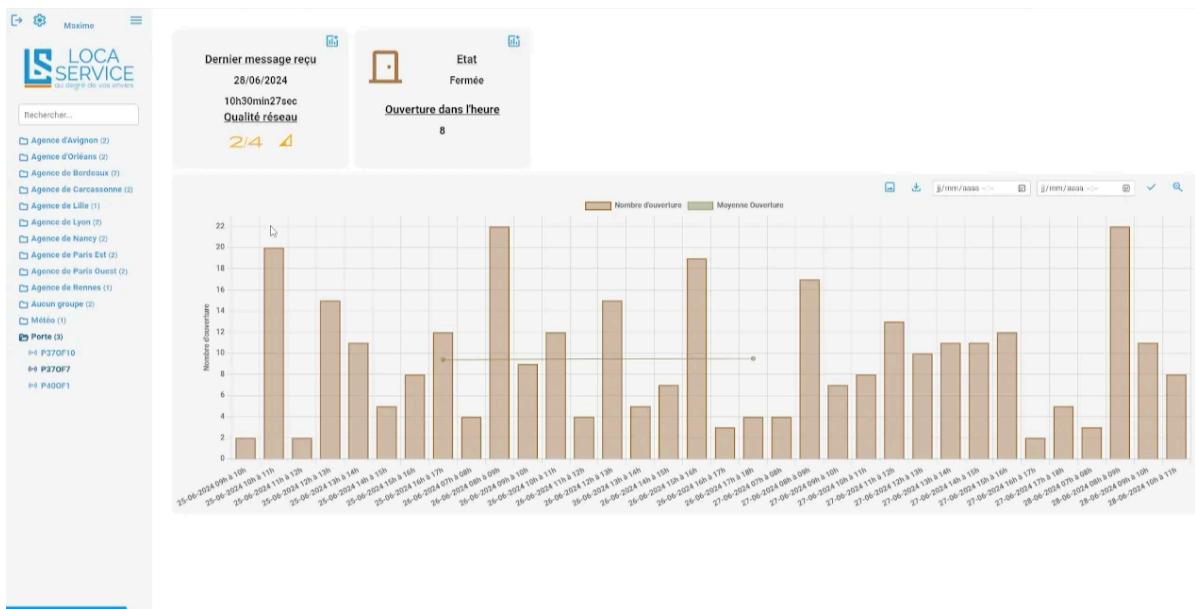
# A. L'affichage des données

Pour afficher les données, j'ai opté pour cette affichage:

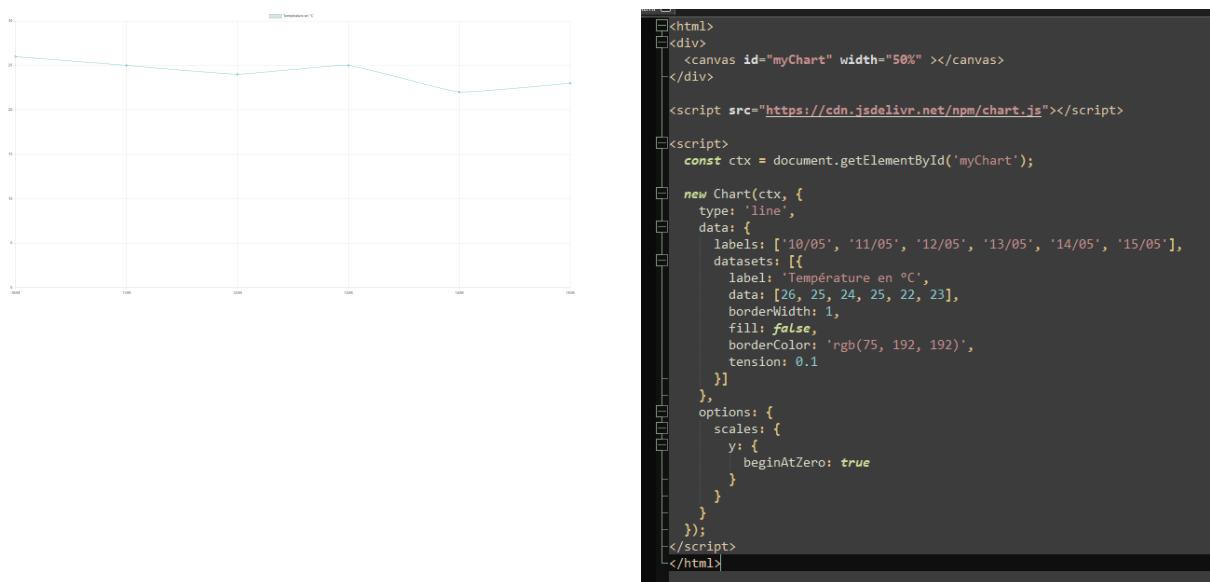
Pour le capteur de température:



Pour le capteur ouverture / fermeture:



Ici j'utilise la bibliothèque Chart.js pour réaliser les graphiques. J'ai suivi la documentation officielle et des forums Stack Overflow pour me former sur cette dernière. Pour me former là dessus j'ai réalisé un graphique statique qui utilisait des tableaux définit sans base de données que voici:



On peut zoomer sur le graphiques, choisir la période que l'on veut visualiser. Les graphiques peuvent être extraits au format png ou même au format CSV. Pour exporter au format CSV, il faut choisir une période et le CSV indiquera toutes les données relevées durant la période choisie. Voilà par exemple à quoi peut ressembler le CSV:

Date	Température (°C)	Humidité (%)	Qualité du réseau
21/06/2024 10:05 19.6	73	-60	
20/06/2024 16:04 19.9	69.6	-81	
20/06/2024 15:03 19.2	71.6	86	
20/06/2024 13:02 19.5	68.7	-65	
20/06/2024 12:01 19.1	69.6	-71	
20/06/2024 11:00 18.1	71.2	68	
20/06/2024 09:59 18	71.3	-76	
20/06/2024 08:58 18	71.4	-74	
19/06/2024 16:04 20	66.9	72	
19/06/2024 15:03 19.3	71.5	-76	
19/06/2024 13:02 19.7	73	-66	
19/06/2024 12:01 19	75	60	
19/06/2024 11:00 18	81.1	-72	

Ici les abonnements fournis par Kuzzle permettent de rafraîchir les données à chaque fois qu'un nouveau document est créé dans le backend Kuzzle lors de la réception d'un signal du capteur.

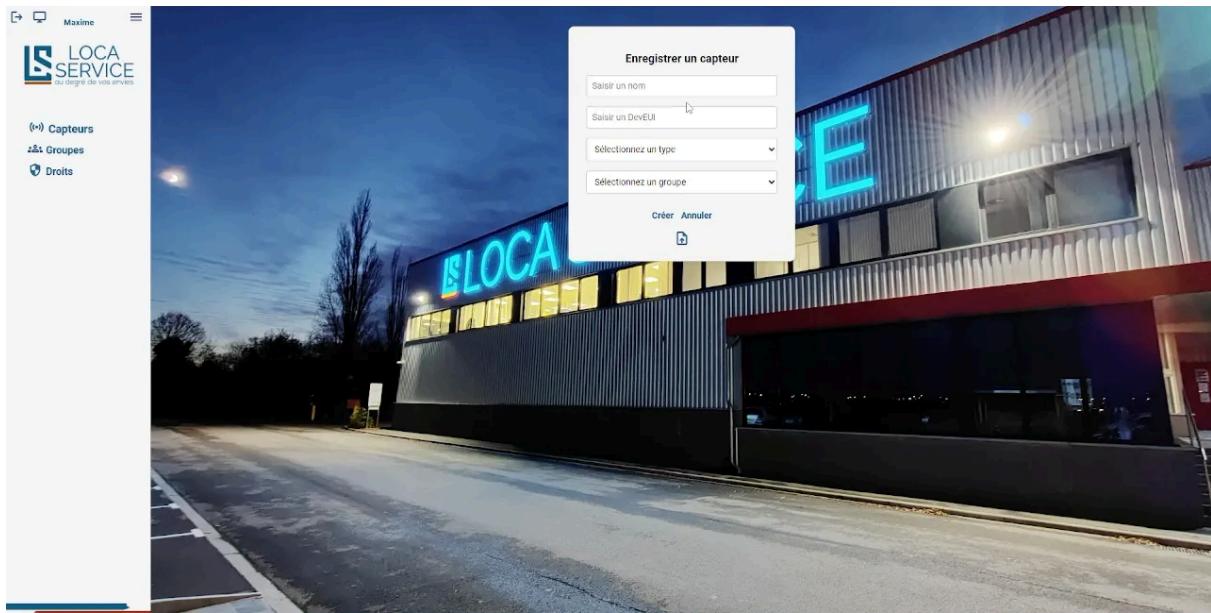
## B. Partie administration

Le projet permet de créer des affichages automatiquement pour les capteurs que l'on va enregistrer dans l'application. Via le site web, on peut ajouter un capteur et ce dernier sera enregistré comme "existant" et ses données seront traitées par la plateforme. Un affichage sera automatiquement créé à son ajout. Un capteur appartient à des groupes restreints ce qui limite l'accès à ces derniers uniquement aux utilisateurs autorisés. Voici la partie administration des capteurs, on peut gérer la création, la modification et la suppression de capteur. On peut aussi ajouter un fichier CVS contenant plusieurs capteurs pour en ajouter plusieurs à la fois.

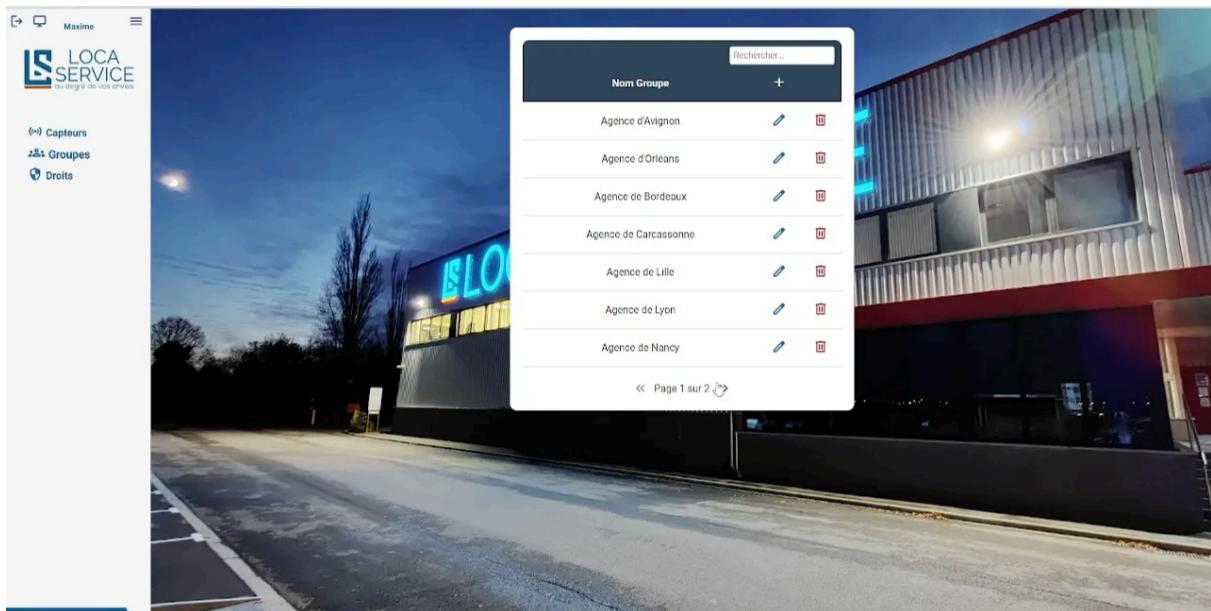
The screenshot shows a web application interface for sensor management. On the left, there's a sidebar with navigation links: 'Capteurs', 'Groupes', and 'Droits'. The main area displays a table of sensor data. The columns are: Nom Capteur, DevEUI, Type Capteur, Groupes, and actions (edit and delete). The data rows are:

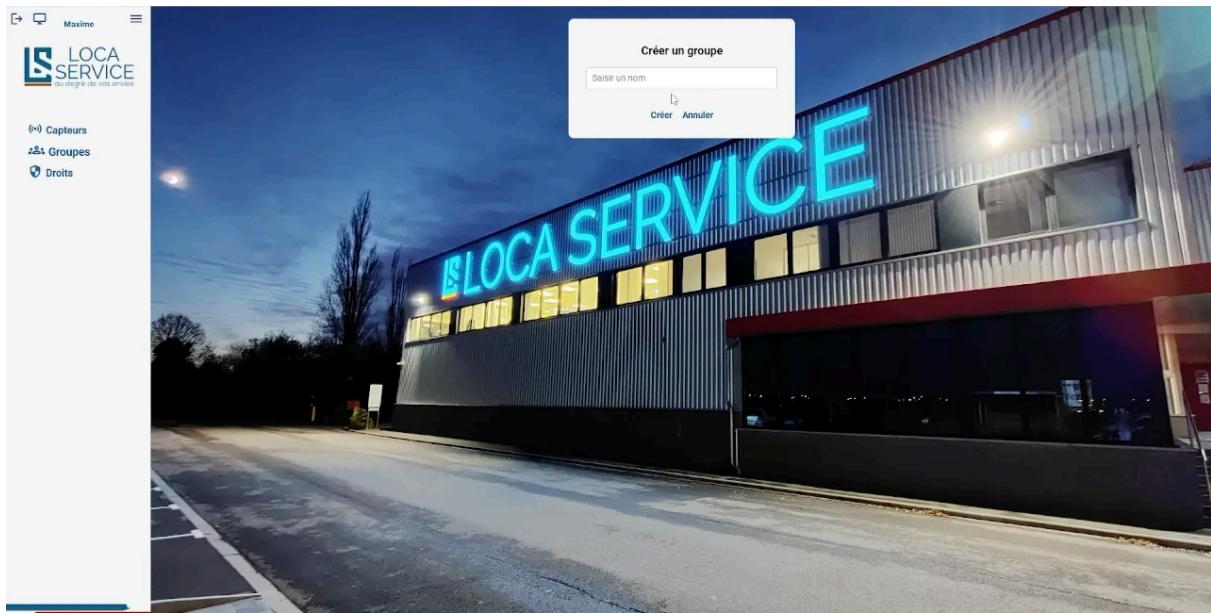
Nom Capteur	DevEUI	Type Capteur	Groupes	
Météo Avignon	MétéoAvignon	temperature	Agence d'Avignon	[Edit] [Delete]
Météo Beauvais	MétéoBeauvais	temperature		[Edit] [Delete]
Météo Bordeaux	MétéoBordeaux	temperature	Agence de Bordeaux	[Edit] [Delete]
Météo Carcassonne	MétéoCarcassonne	temperature	Agence de Carcassonne	[Edit] [Delete]
Météo Lille	MétéoLille	temperature	Agence de Lille	[Edit] [Delete]
Météo Lyon	MétéoLyon	temperature	Agence de Lyon	[Edit] [Delete]
Météo Nancy	MétéoNancy	temperature	Agence de Nancy	[Edit] [Delete]

At the bottom of the table, it says '<< Page 1 sur 4 >>'. The background of the page features a photograph of a solar panel array under a blue sky.

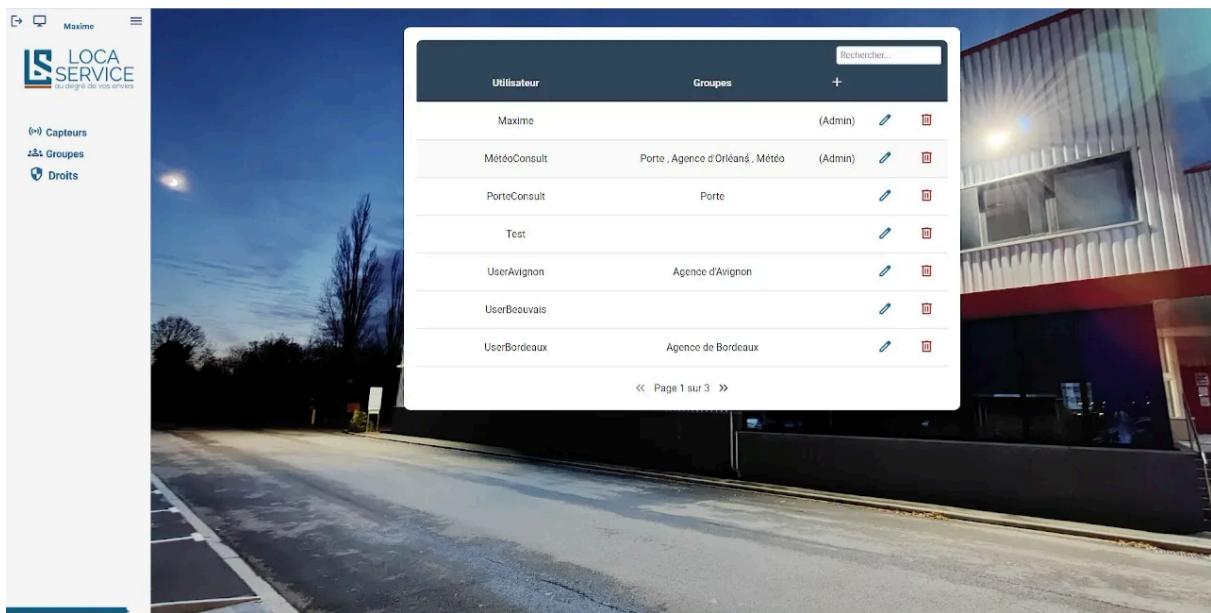


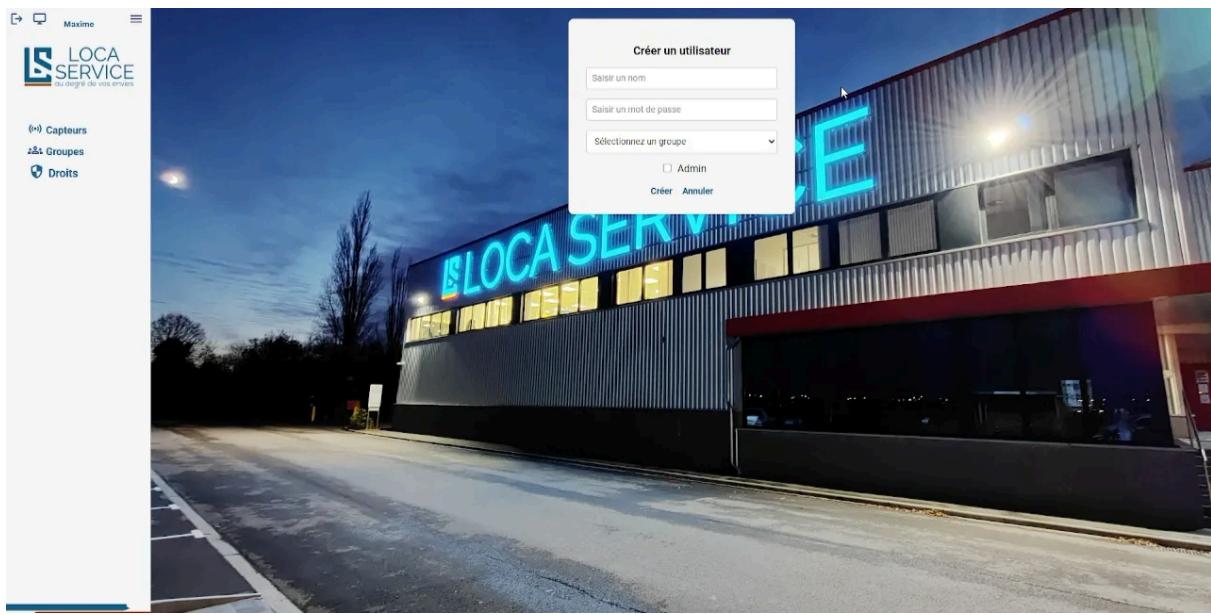
La gestion des groupes est aussi incluse dans le site web, on peut créer, modifier et supprimer des groupes.





Via la partie admin, on peut aussi gérer les utilisateurs et leurs accès. On peut leur donner les groupes pour donner l'accès aux capteurs du même groupe. On peut créer des utilisateurs, les modifier, les supprimer. Voici les interfaces concernées:





Via l'interface des données on peut aussi venir modifier les limites que les données ne doivent pas dépasser, uniquement par les administrateurs, dans le cas d'un dépassement récurrent alors un mail sera envoyé au gestionnaire des capteurs.



## C. Authentification

L'authentification est gérée directement par Kuzzle, la solution donne des outils pour créer des JWT et les vérifier en temps réel. Donc pour chaque action sur le site, le JWT stocké à la connexion sous forme de cookie, est vérifiée par Kuzzle pour être sur que la connexion est valide et sécurisée. Ensuite c'est les rôles de l'utilisateur qui sont récupérés et analysés, encore une fois avec Kuzzle qui permet de récupérer les informations de l'utilisateur connecté.



## D. Création des capteurs, automatisation

Les affichages des données des capteurs sont créés automatiquement au renseignement des capteurs dans la partie administrateur. Lorsqu'un capteur est créé, il est considéré comme existant aux yeux de Kuzzle et ce dernier va collecter et enregistrer les données qu'il envoi. Lorsque l'utilisateur se connecte, les capteurs sont récupérés en fonction des groupes qu'ils possèdent. Par exemple un utilisateur du groupe Lille se connecte, les capteurs avec le groupe Lille seront récupérés avec leurs données et le type du capteur. Ils seront tous mis au sein du groupe Lille qui s'affichera dans le menu de gauche.



Lorsque le capteur est sélectionné, le site va vérifier s'il s'agit d'un capteur de température ou d'ouverture et fermeture. Ensuite il sélectionnera le composant nécessaire entre les deux disponibles, DoorSensor ou TempSensor. Dans ces composants, le DevEUI du capteur (son adresse MAC) est sélectionné et stocké en cookie, permettant de le garder en mémoire le temps des manipulations. Avec ce DevEUI, on effectue les requêtes avec Kuzzle pour récupérer les données du capteur sélectionné. Ainsi, l'affichage se met à jour automatiquement à chaque changement de capteurs, seul deux composants sont disponibles, qui vont se remplir avec les données du capteurs contenu dans le cookie correspondant. Voici comment l'URL se comporte lors de la sélection d'un capteur:

/TempSensor/FAEB45C1F999A033

/DoorSensor/FAEB45C1F999A050

Un capteur de température

Un capteur ouverture / fermeture

## E. SMTP

Comme mentionné plus haut, le projet peut envoyer des mails lorsqu'un capteurs communiquent des données qui indiquent un danger, un dysfonctionnement. Pour se faire j'ai intégrer Node Mailer au projet, directement dans la partie backend avec Kuzzle. J'ai dû apprendre à me servir de cette bibliothèque en utilisant la documentation officielle. J'ai réalisé en guise de test le code suivant en utilisant un SMTP gratuit et jetable, l'entreprise n'ayant pas été disposé à fournir un SMTP pour le test:

```
const nodemailer = require("nodemailer");
let mailConfig: any;

mailConfig ={
  host: 'smtp.ethereal.email',
  port: '587',
  auth:{
    user:'marietta.mccullough34@ethereal.email',
    pass:'aq6dcfAaU6s7x6wHf'
  }
}
let transporter = nodemailer.createTransport(mailConfig);

// Fonction pour envoyer des messages
async function messageLimitTemp(device, compteur, date, limiteMax, limiteMin,temp){
  try {
    const info = await transporter.sendMail({
      from: '"Test Kuzzle" <marietta.mccullough34@ethereal.email>',
      to: "marietta.mccullough34@ethereal.email",
      subject: "Dépassement des limites du capteur "+device+" le "+date,
      text: "Le capteur de température "+device+" a atteint "+temp+"°C et a dépassé les limites données ( Max: "+limiteMax+"°C Min: "+limiteMin+"°C )",
      html: "<b>Le capteur de température "+device+" a atteint "+temp+"°C et a dépassé les limites données ( Max: "+limiteMax+"°C Min: "+limiteMin+"°C )"
    });
    console.log('Email envoyé :', info.response);
  } catch (error) {
    console.error("Erreur lors de l'envoi de l'email :", error);
  }
}
```

Il conviendrait de mieux le sécuriser, surtout au niveau de l'authentification, mais ce code n'était que pour tester si le backend parvenait à comprendre si les données indiquaient un problème et de ce fait envoyer un mail. Dans le backend, lors de la réception des données, si la données récupérées étaient supérieure à la limite fixé dans la configuration du capteur enregistrée lors de sa création, alors le code regardait les 5 dernières mesures, et si les 5 dernières mesures dépassaient la limite, alors un mail était envoyé.

## F. La sécurité

Comme expliqué plus tôt, j'ai mis en place une authentication, ce qui permet de valider qui est connecté et par la vérification de ses rôles de lui accorder les

droits associés. De ce fait un utilisateur qui peut seulement consulter les données des capteurs de Lille ne pourra voir que les capteurs de Lille. Alors qu'un utilisateur admin pourra accéder à tous les groupes, mais aussi à la page d'administration. A noter qu'un utilisateur peut avoir plusieurs groupes. La vérification des rôles est faites à chaque action sur le site, via l'utilisation de cookie et de JWT sécurisés fournis par Kuzzle. Kuzzle se charge de fournir un JWT valide et de le vérifier. Kuzzle gère aussi l'authentification via des fonctions prédéfinies. Voici par exemple comment j'ai mis en place une vérification des cookies à chaque action sur le site:

```
watch: {
    async '$route'(to:any,from:any){
        console.log('URL changée de', from fullPath, 'à', to fullPath);
        (this as any).verifParametre();
        if(from fullPath != '/' && from fullPath != '/LoginPage'){
            (this as any).isLog = await (this as any).checkLogin();
            if((this as any).isLog === true){
                (this as any).menuVisible = true;
            }
            else{
                (this as any).menuVisible = false;
                document.cookie = 'jwt=; Max-Age=0; path=/; samesite=strict';
                document.cookie = 'telemetry=; Max-Age=0; path=/; samesite=strict';
                document.cookie = 'devEUI=; Max-Age=0; path=/; samesite=strict';
                alert("Session expirée ou non valide");
            }
        }
    },
}
```

Ici si l'utilisateur n'est pas connecté, alors il est redirigé, seul un utilisateur est autorisé à accéder au site.

## IV. Conclusion

Ce stage m'a permis de découvrir beaucoup de technologies intéressantes, comme Kuzzle, c'est ma première confrontation à une base de données autre que du SQL et j'ai trouvé celà très enrichissant. De plus découvrir le framework Vue.js m'a permis de mieux comprendre le fonctionnement de Node.js et de toutes les choses permises par ce dernier, mais aussi d'utiliser des bibliothèques comme NodeMailer ou Chart.js. Nul doute que ces compétences acquises me serviront à l'avenir dans le développement d'une solution web moderne. De plus j'ai grandement apprécié l'équipe dans laquelle je me suis

retrouvé, ils m'ont bien accueilli et ont tout fait pour rendre mon expérience et mon apprentissage agréable. Je suis assez fier d'avoir su délivrer le site dans un état convenable et des progrès que j'ai pu constater dans mes compétences de développeur. Bien que je pense que le manque de contact avec une équipe de développeurs puisse m'avoir manqué, je n'ai de ce fait pas eu, ou alors très peu, de retour sur mes capacités de code et mes analyses, etc.