

# RAPPORT DE PROJET TECHNIQUE



## Mise en place et exploitation d'une base de données à partir du backend SIGFOX

*PROPOSÉ L'ASSOCIATION LES ABEILLES DE L'ÉTANG DE DESVRES*

Réalisé par:  
Massine MOUNTASSER  
Abdellah KHOUYA

Encadré par:  
Philippe GHEERAERT

Année universitaire:  
2021 / 2022

---

## Introduction

Dans le cadre du projet technique de seconde année du cycle d'ingénieurs de l'Ecole d'Ingénieurs du Littoral Côte d'Opale - filière informatique, nous sommes amenés à travailler en petit groupe sur une problématique spécifique dans un contexte réel. Dans notre cas, nous sommes un binôme ayant eu l'occasion d'employer nos acquis en méthodes de conception et principalement de développement web dans le but de contribuer au projet Ruche Connectée.

Il s'agit là de concevoir une plateforme commercialisable, autonome, et qui peut certainement être d'assistance précieuse pour les gérants qui cherchent à mieux surveiller leurs fermes. Le service devrait permettre d'obtenir à distance et en temps réel les différentes informations sur les états de leurs ruchers. Ceci permettrait ainsi de non seulement avoir la main sur un modèle assez organisé, mais aussi de suivre l'évolution des colonies dans ses ruches.

## Remerciements

Nous nous excusons par avance si on passe à côté d'une ou deux de toutes les personnes qui nous ont été d'un grand support pour nos réalisations d'une part, puis qui ont pris la peine de nous diriger, conseiller et fournir de précieuses ressources de l'autre.

De tous ceux qui nous ont épaulé tout au long de nos travaux, on cite sans hésitation notre professeur et directeur des études Mr. Georges STIENNE, qui nous a fourni l'orientation nécessaire pour le bon déroulement du projet. Bien évidemment, nous remercions également notre encadrant Mr. Philippe GHEERAERT qui nous a dirigé pendant notre travail et approvisionné de toutes les données indispensables pour une compréhension claire et nette des attentes. N'oublions surtout pas les autres équipes - qu'elles soient de filière Info de même ou Indus (Longueness) - qui ont également déjà accompli plusieurs sous-tâches nécessaires à d'autres fonctionnalités, ce qui permettra de faire des tests, et qui en plus ont été autant disponibles que nécessaire pour toute demande d'échanges d'infos ou de ressources.

---

## Table des matières

|  |          |
|--|----------|
| <b>Introduction</b>                          | <b>1</b> |
| <b>Remerciements</b>                         | <b>1</b> |
| <b>Table des matières</b>                    | <b>2</b> |
| <b>Présentation du projet</b>                | <b>4</b> |
| <b>Contexte</b>                              | <b>4</b> |
| L'apiculture en quelques mots                | 4        |
| Qu'est-ce qu'une ferme apicole?              | 4        |
| Comment gèrent les apiculteurs leurs fermes? | 5        |
| Contexte historique                          | 6        |
| Méthodes anciennes                           | 6        |
| Méthodes modernes                            | 7        |
| <b>Besoin</b>                                | <b>7</b> |
| Les difficultés de surveillance              | 7        |
| Les risques pour les ruches                  | 7        |
| Les challenges des apiculteurs               | 7        |
| <b>Attentes</b>                              | <b>8</b> |
| La solution potentielle proposée             | 8        |
| L'idée générale                              | 8        |
| <b>Cahier des charges</b>                    | <b>9</b> |
| <b>Approche fonctionnelle</b>                | <b>9</b> |
| Fonctionnalités nécessaires                  | 9        |
| Avancements des deux dernières années        | 9        |
| Contraintes actuelles                        | 9        |
| Diagramme de cas d'utilisation               | 9        |
| Schéma de procédure du service               | 11       |
| Diagramme de séquences                       | 11       |
| Approche technique                           | 12       |
| Modèle adapté à la plateforme                | 12       |



---

|   |           |
|---|-----------|
| Diagramme de classes                            | 12        |
| <b>Gestion du projet</b>                        | <b>15</b> |
| Organisation du déroulement du projet           | 15        |
| Diagramme de GANTT                              | 15        |
| <b>Solutions apportées</b>                      | <b>17</b> |
| <b>Méthodes employées</b>                       | <b>17</b> |
| Frameworks et logiciels                         | 17        |
| Organisation du code source                     | 21        |
| Back-end  | 21        |
| Front-end                                       | 22        |
| Résultats atteints                              | 23        |
| Démonstration                                   | 23        |
| Installation                                    | 23        |
| Tests   | 23        |
| <b>Conclusion</b>                               | <b>30</b> |
| <b>Récapitulatif</b>                            | <b>30</b> |
| Résumé & avancement                             | 30        |
| <b>Soucis rencontrés</b>                        | <b>30</b> |
| Obstacles auxquels on ne s'était pas préparé    | 30        |
| Perspectives d'avenir                           | 31        |
| Quelques points qui mériteraient de l'attention | 31        |

---

## Présentation du projet

### Contexte

→ *L'apiculture en quelques mots*

↳ Qu'est-ce qu'une ferme apicole?



Une ferme apicole se définit par un ensemble de ruches disposées dans le but d'en produire du miel. Pour ce qui va suivre, nous considérons ce terme pour désigner une base de données d'un apiculteur, ou plutôt un modèle de ses propriétés en cascade de ruchers (enclos), ruches, cadres et reines.

---

↳ Comment gèrent les apiculteurs leurs fermes?



L'objectif de nos apiculteurs étant d'optimiser leurs productions, ils sont dans la nécessité de garder constamment un œil sur leurs biens: états des ruches, évolutions des colonies...

Ceci implique qu'à chaque besoin de vérification - tous les 9 jours plus ou moins, des contraintes à ne pas négliger s'en suivent: une perte de temps en plus de l'impact sur la colonie en question. Les résultats de ces cas de figure peuvent être graves selon l'accessibilité et la fréquence des visites.



→ *Contexte historique*

↳ Méthodes anciennes



Le miel et la cire font partie du quotidien des hommes depuis plus de 10 000 ans. En effet, avec la découverte du feu, et de l'effet calmant de la fumée sur les abeilles, la récupération du miel est devenue plus facile et moins dangereuse. On fabriquait notamment des ruches avec l'écorce des arbres. L'apiculture a ensuite progressivement trouvé sa place dans l'agriculture.

## ↳ Méthodes modernes



L'inspection visuelle de l'intérieur des ruches modernes avec tout le matériel nécessaire est toujours le moyen privilégié pour assurer un suivi assez régulier de sa ferme. Cependant, cette méthode présente tout de même quelques soucis de gestion.

## Besoin

### → *Les difficultés de surveillance*

#### ↳ Les risques pour les ruches

L'inspection en question présente des risques de destruction des alvéoles de stockage, de refroidissement du couvain, d'arrêt de production causé par l'étourdissement des abeilles qui peuvent en plus générer un risque pour les personnes ou les animaux qui se trouvent dans l'environnement.

#### ↳ Les challenges des apiculteurs

Tout apiculteur qui se voit rencontrer des problèmes similaires devrait songer à une solution lui permettant de vérifier les états de ses biens sans avoir à faire une visite d'inspection classique. Pour cela, l'idée est venue de concevoir ce système de gestion



---

qui propose une solution qui prenne en compte les soucis cités.

## Attentes

→ *La solution potentielle proposée*

↳ L'idée générale

Mettre à disposition des systèmes d'information et de gestion de ses ruches, sans avoir à le faire manuellement et périodiquement pour chaque bien de ses propriétés.

---

## Cahier des charges

### Approche fonctionnelle

#### → *Fonctionnalités nécessaires*

##### ↳ Avancements des deux dernières années

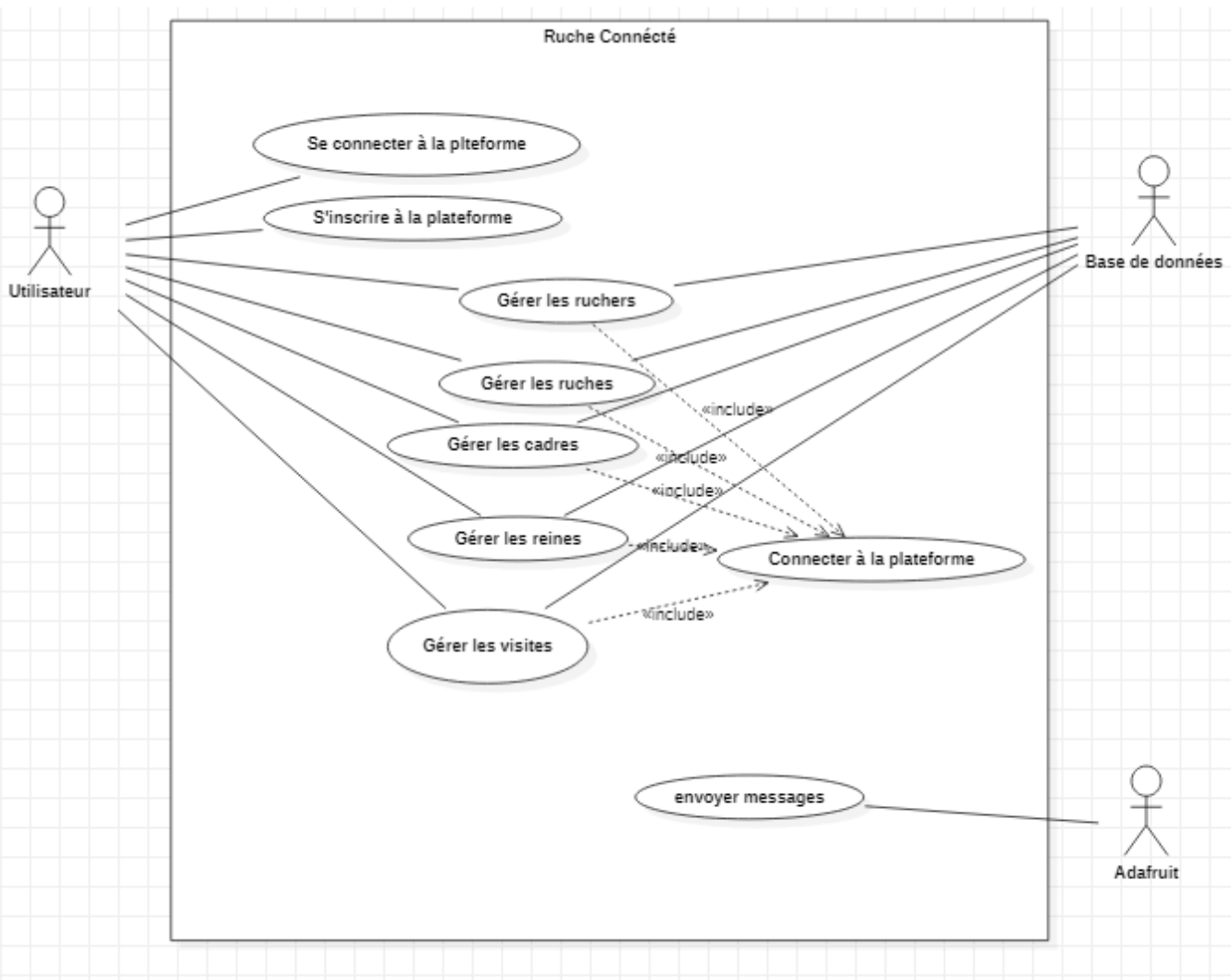
Après avoir consulté le code source des deux dernières années, nous avons constaté que plusieurs fonctionnalités sont absentes. Et nous avons remarqué l'existence de plusieurs anomalies qui nous ont obstrué le processus de développement des tâches demandés.

##### ↳ Contraintes actuelles

Vu qu'on est pas des experts dans le domaine de l'informatique industrielle, le temps qu'on a pris pour s'en renseigner, ainsi que la reformulation du code source des deux années précédentes afin de tester la liaison entre la plateforme et la carte réseau SigFox, nous ont fait perdre énormément du temps.

## ↳ Diagramme de cas d'utilisation

On peut représenter le comportement fonctionnel de cette application dans le diagramme suivant:



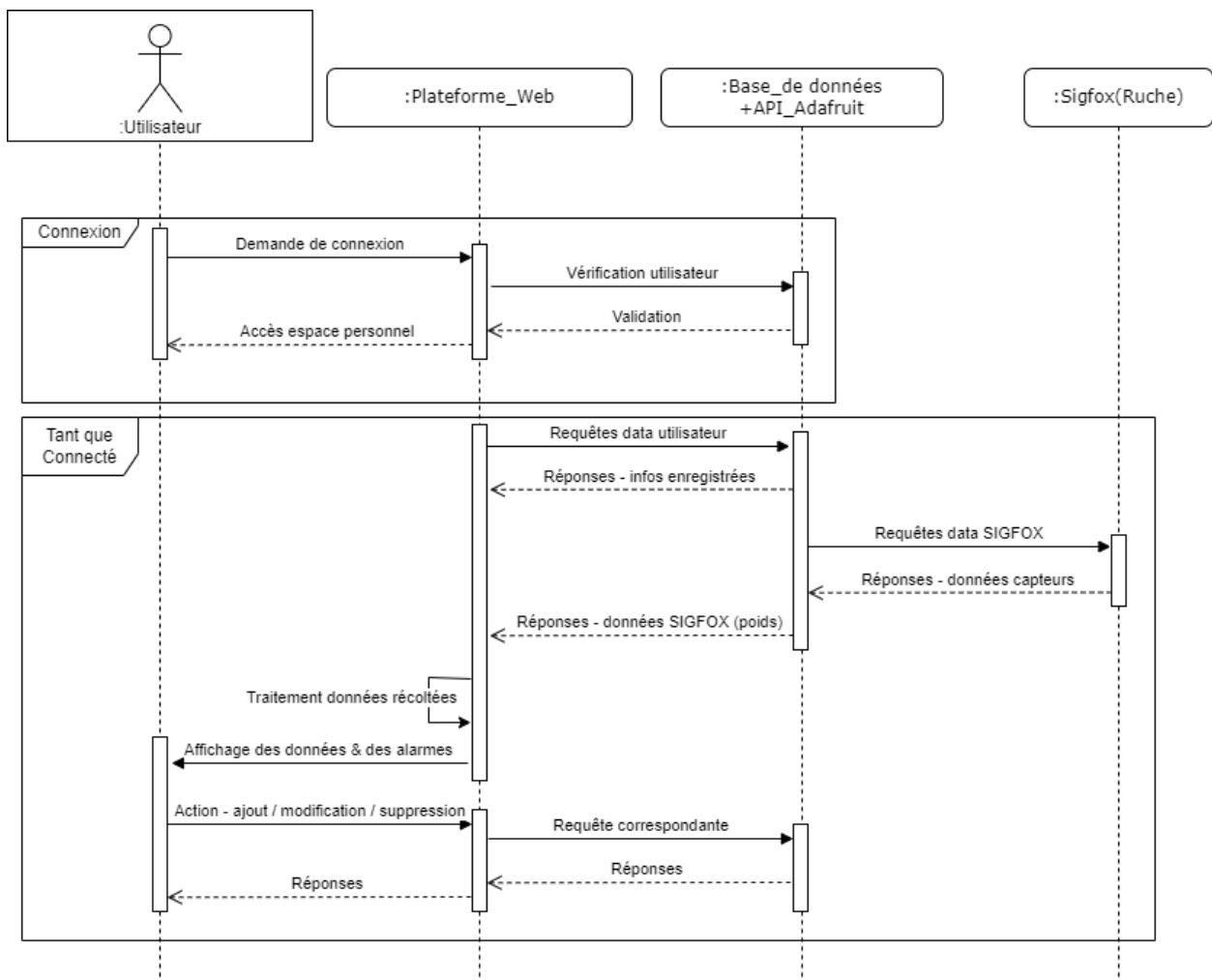
L'utilisateur doit être authentifié pour pouvoir accéder aux fonctionnalités de notre plateforme. Un utilisateur peut gérer les ruchers, ruches, cadres, reines et visites. Adafruit est responsable d'envoyer un message contenant le poids de la ruche, ainsi le

système peut savoir si on doit envoyer une alerte ou non. Et bien évidemment, toutes les interactions avec la plateforme sont stockées dans la base de données.

## → Schéma de procédure du service

### ↳ Diagramme de séquences

Pour mieux illustrer le comportement des différents mécanismes et interactions de l'application, on peut le modéliser avec le diagramme séquentiel suivant:



Tout d'abord, quand l'utilisateur accède à la plateforme, ces identifiants sont requis en premier lieu soit pour se connecter ou s'inscrire. La plateforme envoie ensuite ces informations pour les sauvegarder ou vérifier le mot de passe s'il est en mode connexion.

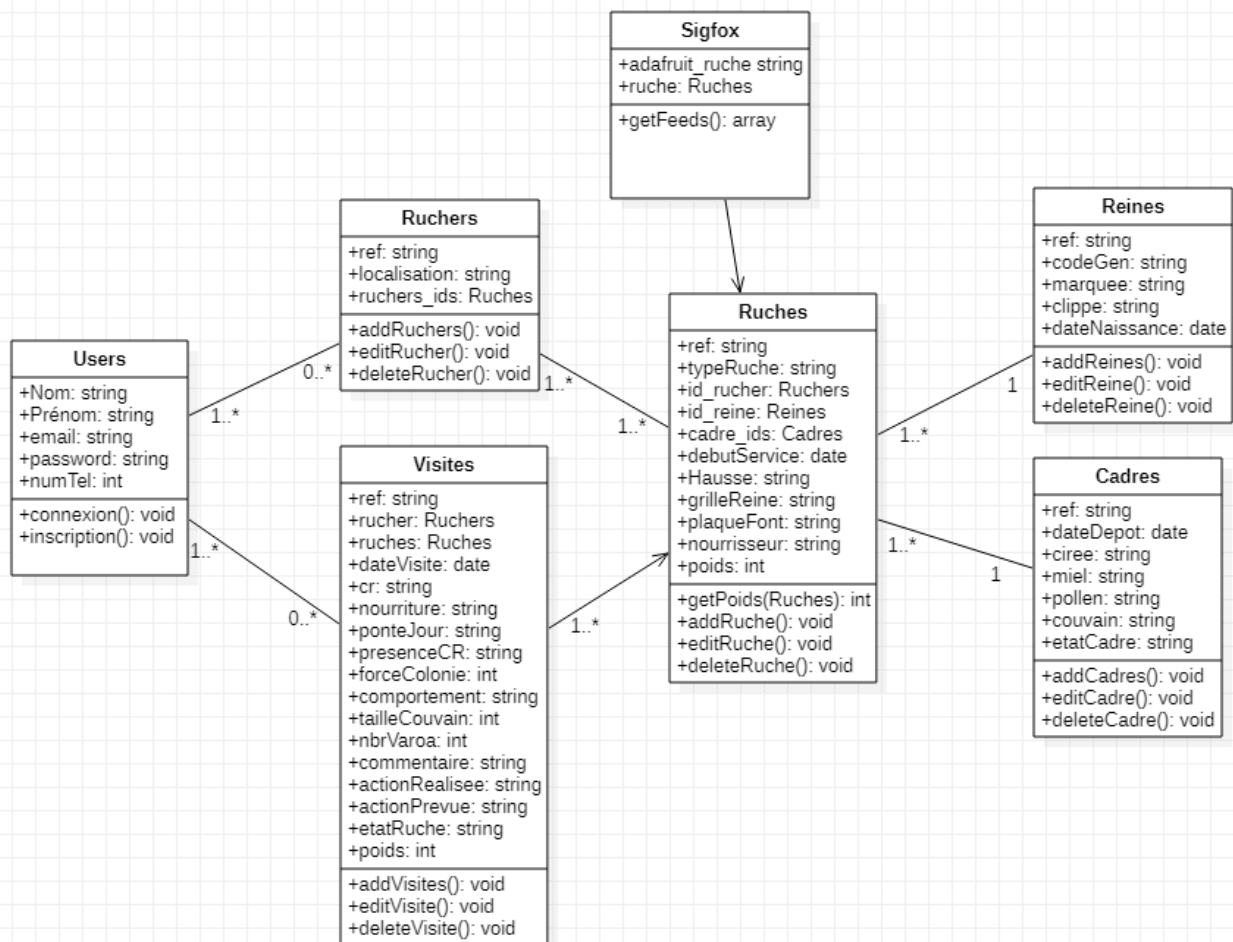
Une fois que l'utilisateur accède à son espace personnel (en cas de respect de la procédure), il a alors accès à son panneau de contrôle général, au gestionnaire des ruchers, ruches, cadres et reines puis celui des visites. De plus, il peut visualiser ses notifications ou alertes récentes ou stockées. L'application visuelle interagit à chaque demande de l'utilisateur avec l'API (Back-end) qui communique en même temps avec sa base de données et la carte réseau Sigfox.

## Approche technique

### → Modèle adapté à la plateforme

#### ↳ Diagramme de classes

Pour avoir une meilleur connexion entre l'interface de l'application et la base de données on a décidé de suivre le diagramme de classe au dessus qui contient 7 tables:



---

**Users:** qui va collecter les informations des utilisateurs, tout ce qui est nom, prénom, mail, mot de passe et numéro de téléphone elle contient aussi les deux méthodes connexion et inscription pour gérer la connexion de l'utilisateur

- **connexion()** : Vérification de l'inscription de l'utilisateur ensuite le connecter à l'application si il est déjà inscrit
- **inscription()** : L'enregistrement des utilisateurs

**Ruchers:** elle inclut les informations les différentes ruchers de l'utilisateur et des méthodes pour piloter l'ajout, suppression et la modifications des ruchers

- **addRuchers()**: L'ajout des ruchers
- **editRucher(string id)**: la modification du rucher
- **deleteRucher()**: Suppression du rucher

**Ruche:** contient les différentes informations en relation avec la ruche, son type, ses cadres, sa reine, etc. et des méthodes pour diriger l'ajout, suppression et la modifications des ruches et aussi une méthode pour récupérer le poids envoyé par la carte Sigfox

- **addRuches()**: L'ajout des ruches
- **editRuche(string id)**: la modification de la ruche
- **deleteRuche()**: Suppression de la ruche
- **getPoids()**: Récupération du poids envoyé par la carte réseau Sigfox

**Cadres:** qui contient les différentes informations des cadres des ruches enregistrées, (la date du dépôt, miel, pollen, etc ) et aussi des méthodes pour manager les cadres

- **addCadres()**: L'ajout des cadres
- **editCadre(string id)**: la modification de la cadre
- **deleteCadre()**: Suppression de la cadre



---

**Reine:** contient les informations des reines qui demeurent dans les ruches, ainsi que des méthodes pour manager les reines (changement des informations de la reine, suppression de la reine, ajout)

- **addReines():** L'ajout des reines
- **editReine(string id):** la modification de la reine
- **deleteReine():** Suppression de la reine

**Visites:** enregistre les différentes visites effectuées par l'utilisateur avec des méthodes qui peuvent ajouter, supprimer ainsi que modifier les informations d'une précédente Visites.

- **addVisites():** L'ajout des visites
- **editVisite(string id):** la modification de la visite
- **deleteVisite():** Suppression de la visite

**Sigfox:** celle qui relie les enregistrer par la carte réseau Sigfox avec les ruches créer dans la plateforme pour mieux communiquer avec cette dernière

- **getFeeds():** récupérations des différents ruches enregistré dans la plateforme de Sigfox

## Gestion du projet

### → Organisation du déroulement du projet

#### ↳ Diagramme de GANTT

| Task Name                                | Séptembre | Octobre | Novembre | Décembre | Janvier | Février | Mars | Avril |
|--|-----------|---------|----------|----------|---------|---------|------|-------|
| Choix de sujet du projet Tech            |           |         |          |          |         |         |      |       |
| Etude du sujet et dispersion des tâches  |           |         |          |          |         |         |      |       |
| Etude du code source de l'année dernière |           |         |          |          |         |         |      |       |
| Documentation - Sigfox/NodeJS            |           |         |          |          |         |         |      |       |
| Conception                               |           |         |          |          |         |         |      |       |
| Réalisation                              |           |         |          |          |         |         |      |       |
| Tests du projet                          |           |         |          |          |         |         |      |       |

En octobre, nous avons rempli le formulaire de choix du projet TECH, ensuite l'école nous a affecté ce projet qui nous a paru comme un challenge à relever puisqu'on ne connaissait rien sur SigFox. Ensuite, nous avons passé les deux premières semaines du mois Novembre dans l'étude du projet pour mieux comprendre le besoin et ce qui est demandé. Puis, nous avons passé beaucoup de temps en essayant de comprendre le code source des années antérieures. Ensuite, nous nous sommes documentés auprès de SigFox, et plus précisément de la récupération de données à l'aide de NodeJS Là nous avons décidé de refaire la globalité du projet dans le but d'avoir l'environnement idéal pour assurer la transmission des messages vers la plateforme afin d'afficher les alarmes et les notifications. Après, nous sommes passés à l'étape de la conception où nous avons réalisé tous les diagrammes figurant ci-dessus. Cette étape est incontestablement importante au début de la réalisation de chaque projet. Elle permet de modéliser le besoin en diagramme pour éviter la perte de temps dans la phase de développement. Cette dernière est venue juste après la phase de conception où on a implémenté les méthodes pour remplir les tâches demandées. Enfin, nous avons effectué des tests dans le but de s'assurer l'absence d'anomalies.

---

## Solutions apportées

### Méthodes employées

→ *Frameworks et logiciels*

a- HTML5



Le programme de base en matière de structuration, de mise en réseau et de contenu sur le World Wide Web.

b- CSS3



CSS3 est un langage de programmation déclaratif qui permet d'affecter un style au code HTML.

### c- SASS



SASS est le langage d'extension CSS de qualité professionnelle le plus mature, stable et puissant au monde. L'écriture avec SASS est mieux organisée, plus lisible, moins répétitive. Ce qui booste les performances du site web et rend le travail sur le fichier plus agréable.

### d- Javascript



Javascript est un langage de programmation impératif qui s'exécute dans les navigateurs web. Il permet de rajouter la dynamique à une page web pour créer de l'interactivité dans une page web. C'est aussi un langage de programmation orienté objet.

---

## e- Axios

# A X I O S

Axios est une bibliothèque JavaScript fonctionnant comme un client http. Elle permet de communiquer avec des apis en utilisant des requêtes.

## f- VueJS



Vue.js (aussi appelé plus simplement Vue), est un Framework JavaScript open-source utilisé pour construire des interfaces utilisateur et des applications web monopages. Ce Framework se compose d'une bibliothèque de base, de Framework et d'autres outils qui permettent un développement frontend facile et rapide.

---

## g- NuxtJS



Nuxt.js est un Framework d'applications WEB gratuit et open source basé sur Vue.js, node.js. Ce Framework est présenté comme un méta-cadre pour les applications universelles. Une application universelle se différencie des applications web classiques par sa capacité à pouvoir s'exécuter aussi bien côté client que côté serveur. Cela ouvre la possibilité de pouvoir construire une page côté serveur puis de l'envoyer déjà constituée au client.

## h- NodeJS



Node.js est une plateforme logicielle libre en JavaScript, orientée vers les applications réseau évènementielles hautement concurrentes qui doivent pouvoir monter en charge.



---

## i- ExpressJS

# express

Express.js est un Framework pour construire des applications web basées sur Node.js. C'est de fait le Framework standard pour le développement de serveur en Node.js.

## j- MongoDB



MongoDB de l'anglais humongous qui peut être traduit par « énorme », est un système de gestion de base de données orienté documents, répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données.

---

## → Organisation du code source

### ↳ Back-end

Tous les dossiers et les fichiers qui sont en relation avec la partie back-end, sont dans le dossier api, ce dossier contient les composants suivants :

- **app.js**: C'est le script qui connecte notre api à la base de données MongoDB
- **Models**: Il contient les descriptifs des tables utilisés dans la base de données: users, ruchers, ruches, reines, cadres, visites, sigfox
- **Routes**: Il contient les gestionnaires des requêtes envoyées à l'api et les traitées pour retourner à la fin un résultat.
- **.env**: Un fichier qui contient des variables d'environnement

### ↳ Front-end

Tous les dossiers et les fichiers qui sont en relation avec la partie front-end, sont dans le dossier app, ce dossier contient les dossiers et les fichiers suivants:

- **Assets**: On utilise ce dossier généralement pour assurer la mise en page de notre site et de l'adapter aux différents appareils qu'on peut utiliser pour y accéder. Il contient les images et la conception utilisée pour la mise en page du site.
- **Layouts**: Il contient les composants répétitifs de notre site : Default.vue, error.vue.
- **Middleware**: Ce dossier gère la sécurité de l'application, permission d'accès aux pages
- **package.json**: Il contient les dépendances utilisées dans la partie front-end de l'application
- **nuxt.config.js**: Il contient la configuration de l'application: port, l'emplacement des dossiers utilisés
- **Components**: Il contient les différents composants des pages de l'application

- 
- **Plugins:** Il contient le script des requêtes envoyer au back-end.
  - **Pages:** Ce dossier contient toutes les pages web de l'application. Notre application contient 4 pages web :
    - **index.vue** : page de connexion
    - **hives.vue** : page du gestionnaire des ruchers, ruches, cadres, reines
    - **dashboard.vue** : page du panneau de contrôle
    - **visits.vue** : page du gestionnaire des visites
  - **Store:** Ce dossier gère la session de l'utilisateur dans le navigateur.

---

## Résultats atteints

### → Démonstration

#### ↳ Installation

Après avoir télécharger le dossier du projet technique, vous pouvez le retrouver dans le lien suivant: "<https://gitlab.com/p6719/projet-tech/-/tree/develop>". Ensuite, vous devriez ouvrir le dossier dans **Visual Studio Code**, dans un terminal **Powershell**, en partant de la racine du projet, il faut accéder au répertoire **app**, exécuter la commande suivante: **npm install**, pour installer les dépendances qu'on aura besoin dans cette partie, qui existent dans le fichier package.json, ensuite on exécute la commande: **npm run dev**, l'**app** va, donc, se lancer sur localhost au port, pour y accéder avec le lien suivant "**localhost:8080**". Pour le moment, on a lancé la partie **Front-end** de l'application.

Pour le lancement de l'**api**, c'est comme l'**app**, il faut commencer par ouvrir un terminal en partant de la racine du projet technique et taper la commande suivante: **npm install** pour installer toutes les dépendances utiliser dans l'**api** qui existent dans le fichier **package.json**, ensuite on exécute la commande: **npm run dev** pour lancer l'**api** qui va se déployer sur le serveur local "**localhost**" au port **5000** et vous pouvez y accéder avec le lien suivant: "**localhost:5000**".

#### ↳ Tests

Maintenant que les deux parties front-end et back-end sont lancées, on n'a qu'à aller à l'adresse de l'**app**, la page **index.vue**. Si l'utilisateur est déjà enregistré, alors il lui suffit juste de rentrer l'email et le mot de passe, après il clic sur se connecter, l'application va donc vérifier si l'email, que l'utilisateur a saisi, est enregistré dans la base de données, si oui, elle va donc vérifier si son mot de passe est correct, sinon un message d'erreur va être affiché en informant l'utilisateur que, soit son email ou son mot de passe n'est pas valide.

Bienvenue sur Ruche Connectée

**Ruche Connectée - plateforme de gestion de fermes apicoles**

Notre objectif: faciliter l'automatisation de la surveillance de l'état de vos ruches.

Connectez ou inscrivez vous pour accéder à votre espace personnel.

CONNEXION

INSCRIPTION

Saisissez votre adresse email et votre mot de passe pour continuer.

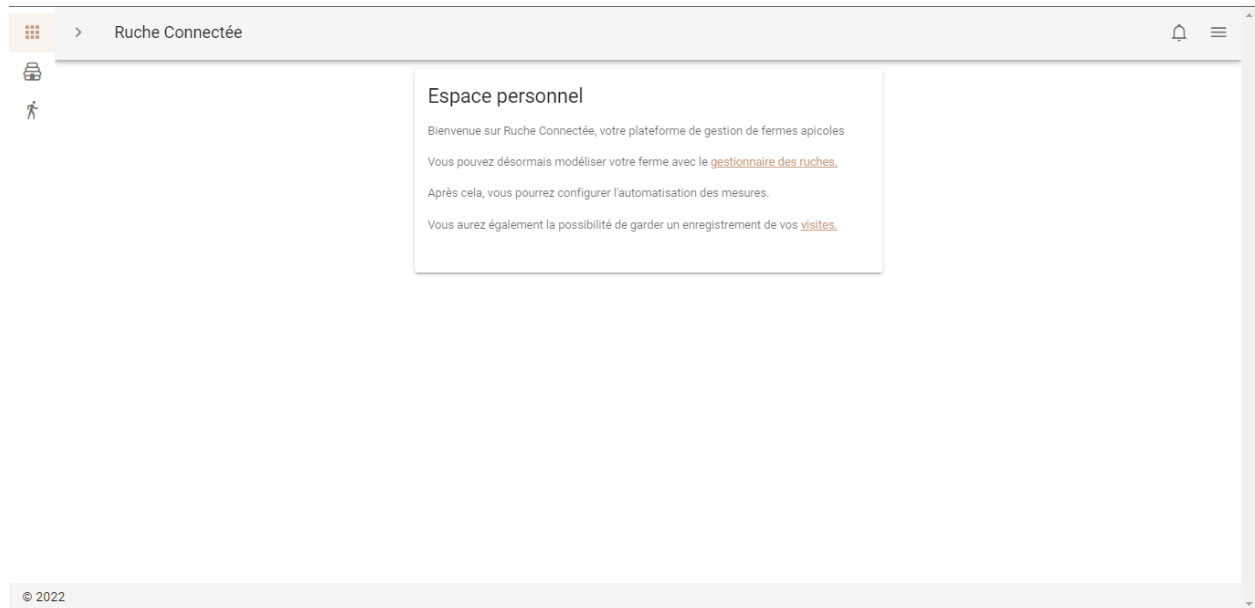
Adresse email

Mot de passe

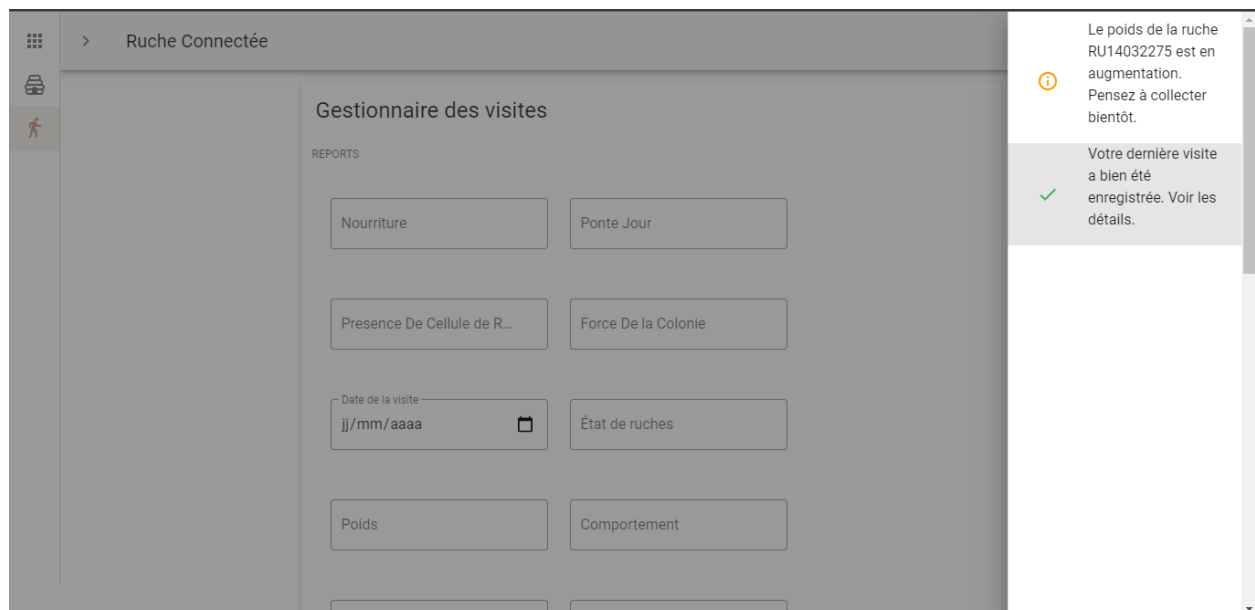
SE CONNECTER

© 2022

Si c'est la première fois que l'utilisateur accède à notre plateforme. L'utilisateur est obligé de créer un compte pour pouvoir bénéficier de ses fonctionnalités. Il doit remplir les champs : **Nom, Prénom, Adresse Mail, Numéro de téléphone, mot de passe** et la **confirmation du mot de passe**. Le backend vérifie que l'adresse mail délivrée respecte le format des email (**des caractères + @ + un nom de domaine**), il vérifie aussi la conformité des champs de mots de passe. Si toutes les informations fournies sont valides, on instancie un utilisateur et ensuite on le stock au niveau de la base de données après avoir effectué un hachage au mot de passe. Sinon, la fenêtre affiche un message d'erreur indiquant l'anomalie à rectifier.

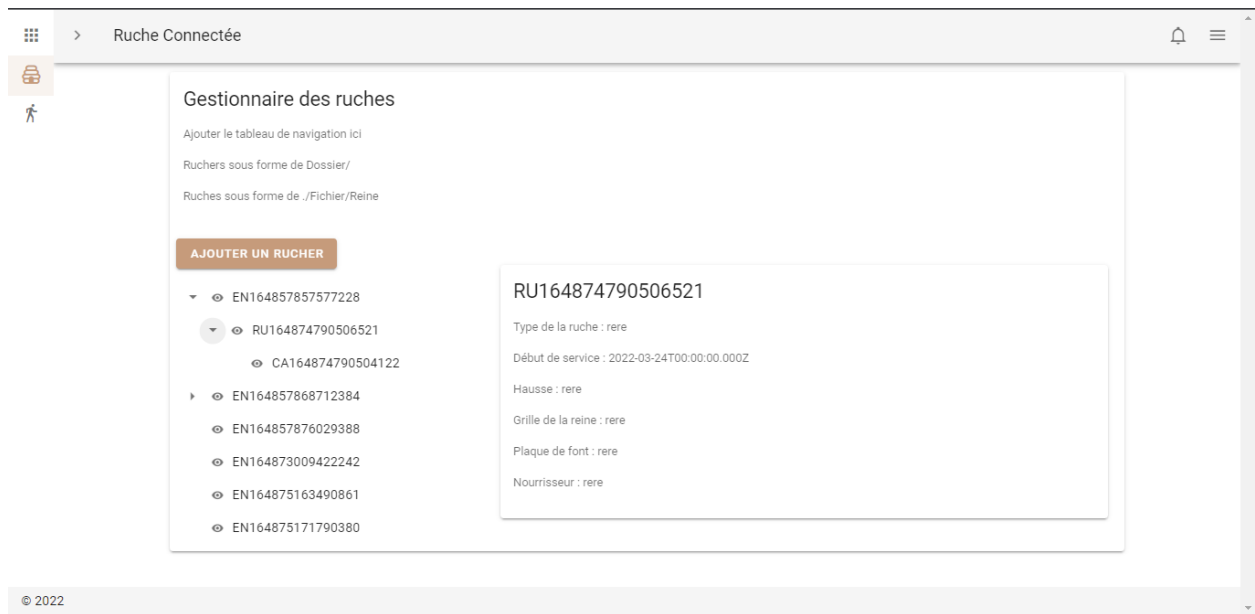


Lorsque l'utilisateur est connecté, il est automatiquement redirigé vers la page dashboard. Cette page est le portail vers toutes les autres. Elle est dynamique, elle contient des raccourcis qui dirigent vers les gestionnaires. Aussi, en haut à droite de la page elle offre des icônes. celle des notifications, qui affiche des alarmes dans une barre latérale. Et une autre pour se déconnecter.





En cliquant sur la deuxième icône figurant en haut à gauche de la page ci-dessous, elle redirige l'utilisateur vers la page de gestion des ruchers. Là, il aura la possibilité de visualiser les ruchers, ruches disponibles, leurs cadres et reines. Ainsi que d'ajouter un rucher ou une ruche, en remplissant les champs demandés.



**Gestionnaire des ruches**

Ajouter le tableau de navigation ici

Ruchers sous forme de Dossier /

Ruches sous forme de /Fichier/Reine

**AJOUTER UN RUCHER**

- EN164857857577228
- RU164874790506521**
  - CA164874790504122
- EN164857868712384
- EN164857876029388
- EN164873009422242
- EN164875163490861
- EN164875171790380

**RU164874790506521**

Type de la ruche : rere

Début de service : 2022-03-24T00:00:00.000Z

Hausse : rere

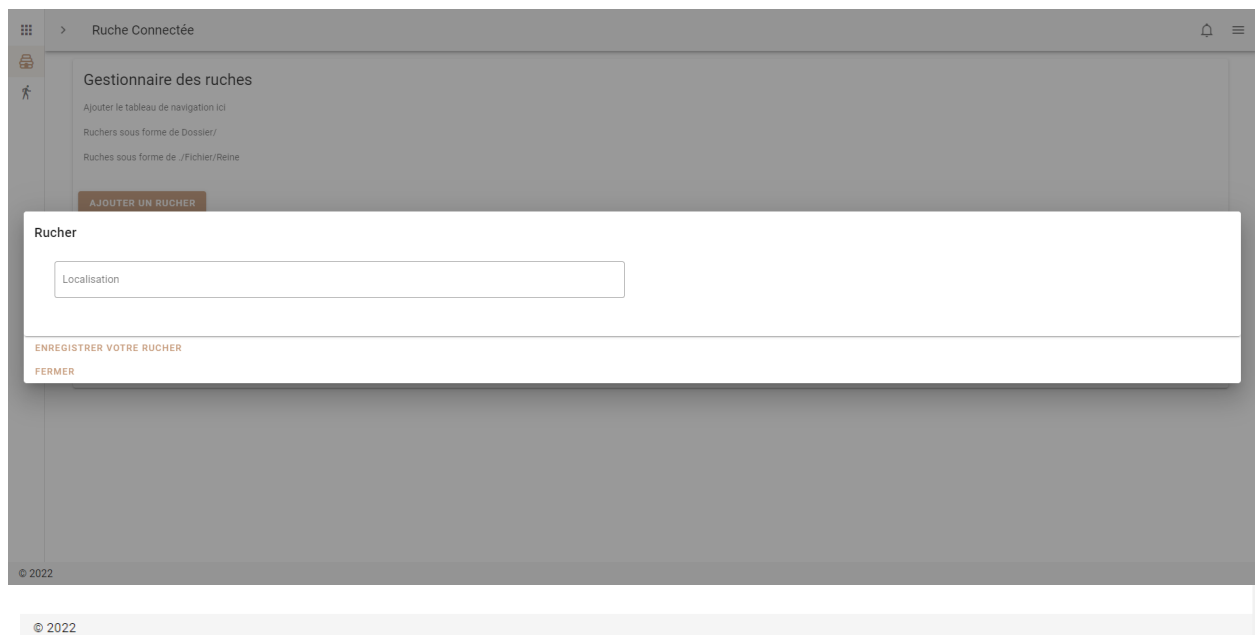
Grille de la reine : rere

Plaque de font : rere

Nourrisseur : rere

© 2022

La figure ci-dessous représente le formulaire d'ajout d'un rucher.



**Rucher**

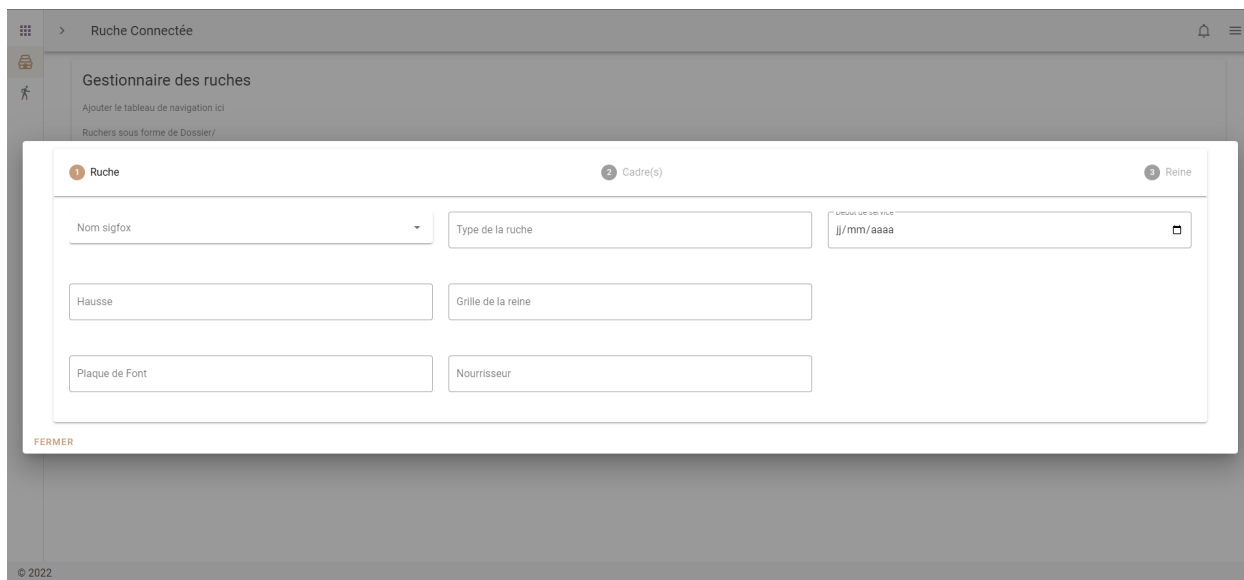
Localisation

**ENREGISTRER VOTRE RUCHER**

**FERMER**

© 2022

Le formulaire d'insertion d'une ruche dans la base de données, se compose de trois étapes. La première réservée aux informations qui concernent la ruche. Le premier champ permet de sélectionner les ruches enregistrées par la carte réseau SigFox.

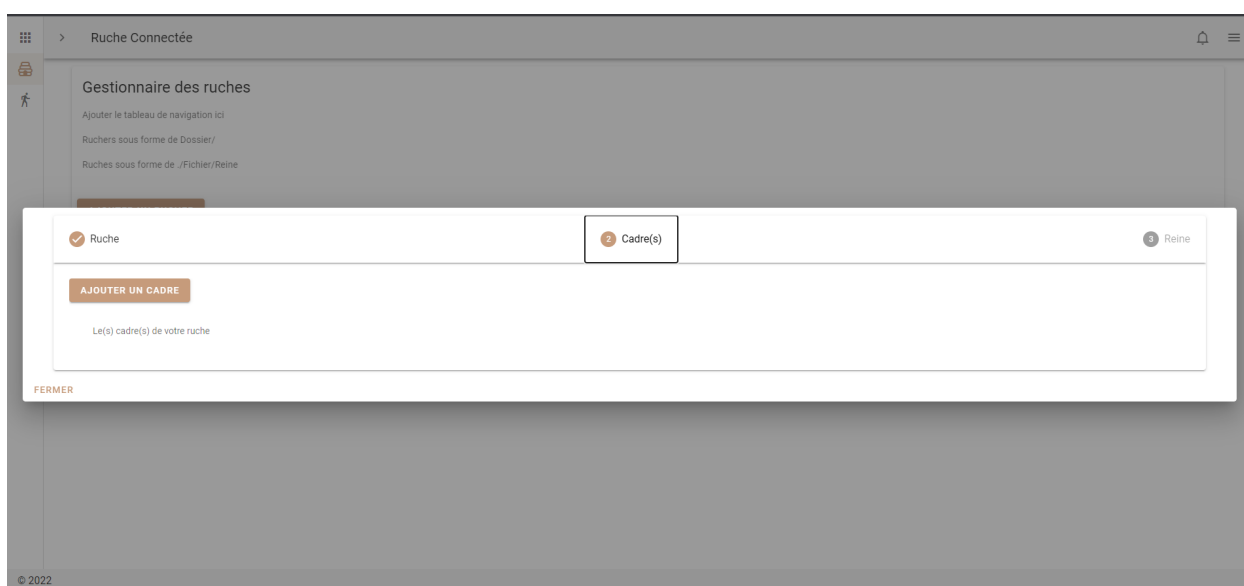


The screenshot shows the 'Gestionnaire des ruches' interface. The first step, '1 Ruche', is active. The form includes the following fields:

- Nom sigfox (dropdown menu)
- Type de la ruche (text input)
- Date de la ruche (calendar icon, format jj/mm/aaaa)
- Hausse (text input)
- Grille de la reine (text input)
- Plaque de Font (text input)
- Nourrisseur (text input)

A 'FERMER' button is located at the bottom left of the form.

La deuxième étape consiste à ajouter un ou plusieurs cadres de la ruche en question

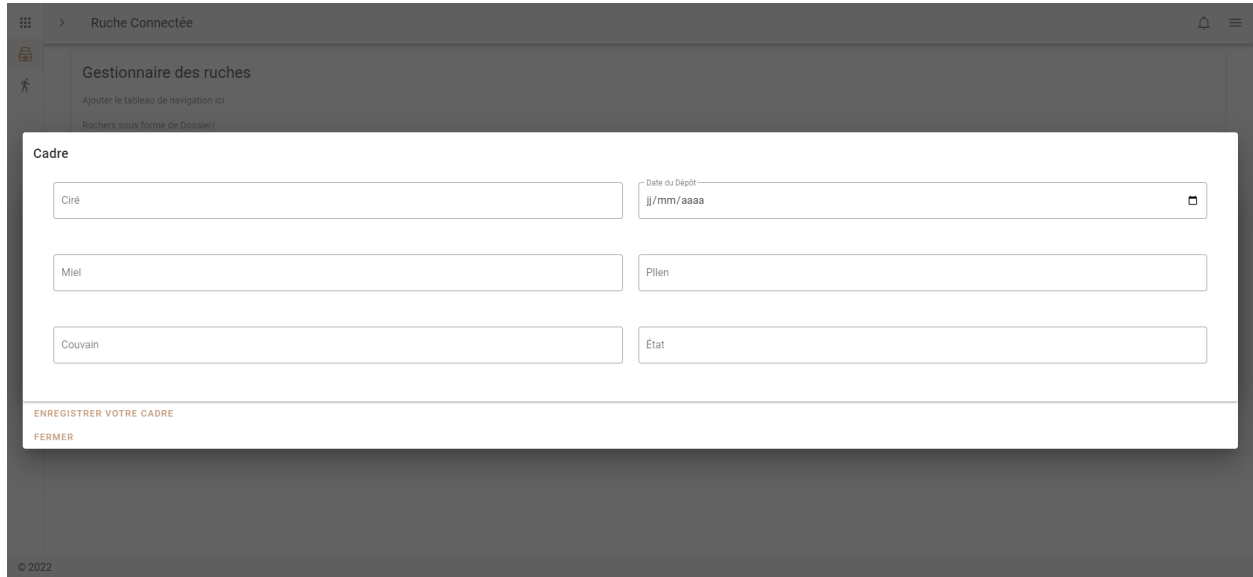


The screenshot shows the 'Gestionnaire des ruches' interface. The second step, '2 Cadre(s)', is active. The 'Ruche' tab is selected and marked with a checkmark. The form includes the following elements:

- AJOUTER UN CADRE (button)
- Le(s) cadre(s) de votre ruche (text input)

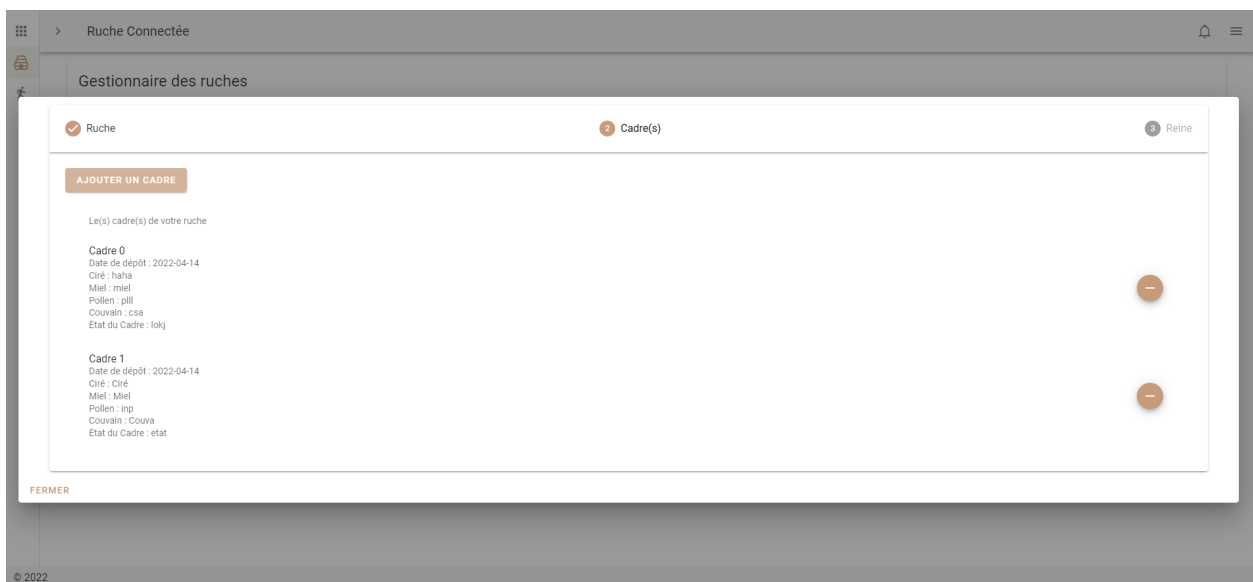
A 'FERMER' button is located at the bottom left of the form.

En appuyant sur le bouton **“ajouter un cadre”**, le formulaire suivant s’affiche:



The screenshot shows a web application interface for 'Ruche Connectée'. The main section is titled 'Gestionnaire des ruches' and contains a sub-section 'Cadre'. The 'Cadre' section has four input fields: 'Ciré', 'Date du Dépôt' (with a date picker icon), 'Miel', 'Pllen', 'Couvain', and 'État'. Below the input fields are two buttons: 'ENREGISTRER VOTRE CADRE' and 'FERMER'. The footer of the application shows '© 2022'.

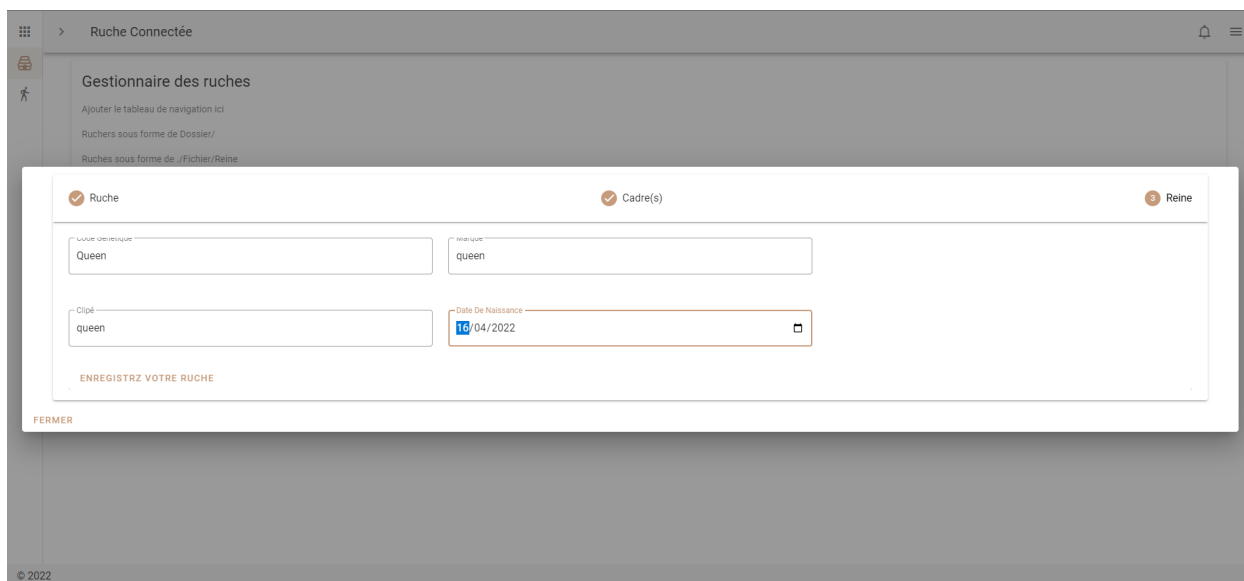
Après l’ajout de plusieurs cadres, l'utilisateur a la possibilité de retirer quelques-uns voire tous.



The screenshot shows the same web application interface, but now the 'Cadre(s)' section is active. It displays a list of two frames, 'Cadre 0' and 'Cadre 1', each with its own set of details (Date de dépôt, Ciré, Miel, Pollen, Couvain, État du Cadre). To the right of each frame's details is a minus sign icon, indicating that frames can be removed. The 'AJOUTER UN CADRE' button is still visible at the top left of the list. The footer shows '© 2022'.

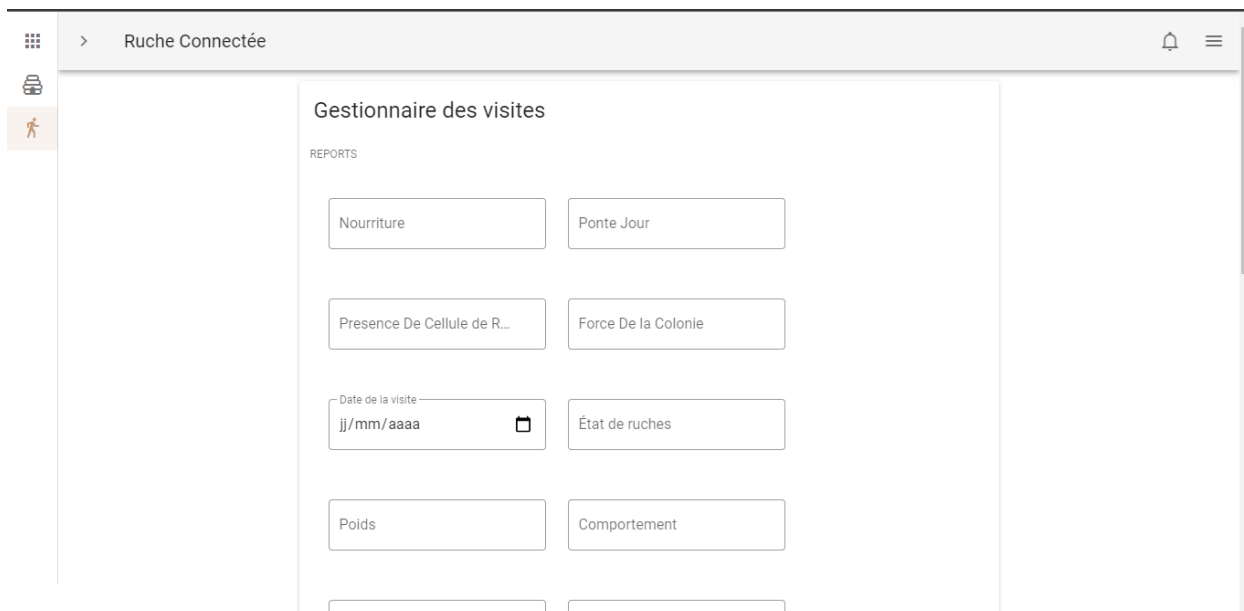
Ensuite vient l’étape finale de l’insertion d’une ruche, c’est de remplir les informations

de la reine. Finalement, l'utilisateur peut donc enregistrer la ruche dans la base de données en cliquant sur le bouton **enregistrer**.



Ruche créer avec succès

Enfin, le dernier icône en haut à gauche de la page est un raccourci vers le gestionnaire des visites. Il fournit une page contenant un formulaire d'ajout d'une visite pour la stocker dans la base de données.



---

## Conclusion

### Récapitulatif

#### → *Résumé & avancement*

Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici  
Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici.

### Soucis rencontrés

#### → *Obstacles auxquels on ne s'était pas préparé*

- Nous avons rencontré des problèmes lors de la phase de récupération du code déjà existant. C'était développé en React JS, mais nous n'avons eu que le build. Nous n'avons pas pu récupérer le code source au début, ce qui nous a poussé à refaire ce qui existait déjà avec NuxtJS, VueJS. Ce qui nous a pris beaucoup de temps.
- Nous n'avons jamais travaillé avec AdaFruit avant le commencement du développement de notre application. Ce qui a ralenti la phase de développement. Tester ses fonctionnalités au niveau de notre interface graphique était problématique au point qu'on a eu pas mal d'erreurs à gérer.
- Nous nous sommes retrouvés dans un dilemme. Soit nous continuons sur le code développé l'année dernière, ou on refait tout à partir de 0.
- Manque de communication efficace sur les conventions à prendre entre les groupes travaillant sur le même projet.
- Difficulté de modéliser quelques objets avec un nombre considérable d'attributs.

---

## Perspectives d'avenir

### → *Quelques points qui mériteraient de l'attention*

- Utiliser le protocole du Web Socket pour acquérir les données en temps réel. Surtout quand il s'agit de notifications et alarmes.
- L'implémentation d'un modèle d'intelligence artificielle qui permet de prévoir les alertes à l'avance - en fonction des données récoltées.
- L'optimisation de la solution reste toujours une étape indispensable de la maintenance. Même si nous avons essayé de développer d'une manière compréhensible et optimale. En respectant les bonnes pratiques du développement d'un projet web, il reste toujours une façon d'optimiser encore plus.
- Au lieu d'avoir deux gestionnaires: ruches et visites, on pourrait implémenter des gestionnaires plus complexes concernant les cadres et les reines par exemple ou autres.
- On peut mettre en place une version mobile de l'application. Les notifications y seront plus réactives.
- Il serait intéressant de notifier ses utilisateurs par mail et SMS.