

Cegep Serre Tech

Département Informatique du Cégep de Jonquière (DICJ)

Document d’analyse

Table des matières

[**Introduction** 5](#_Toc90470470)

[Rôles 5](#_Toc90470471)

[Organisation 5](#_Toc90470472)

[Rencontre avec l’expert 6](#_Toc90470473)

[Rencontre Boréalis 6](#_Toc90470474)

[Rencontre La Rouquine 6](#_Toc90470475)

[**Application similaire** 7](#_Toc90470476)

[Apply Yourself App: 7](#_Toc90470477)

[Farmhand : 7](#_Toc90470478)

[**Étude de technologies** 8](#_Toc90470479)

[Frontend : 8](#_Toc90470480)

[Solution #1 Angular 8](#_Toc90470481)

[Solution # 2 React 8](#_Toc90470482)

[Solution #3 Vue 9](#_Toc90470483)

[Backend : 9](#_Toc90470484)

[Solution #1 Laravel 9](#_Toc90470485)

[Mode offline : 9](#_Toc90470486)

[**Analyse globale** 10](#_Toc90470487)

[Fonctionnalités générales 10](#_Toc90470488)

[Évaluation de l’envergure du projet 10](#_Toc90470489)

[Liste des acteurs 11](#_Toc90470490)

[Liste des événements 11](#_Toc90470491)

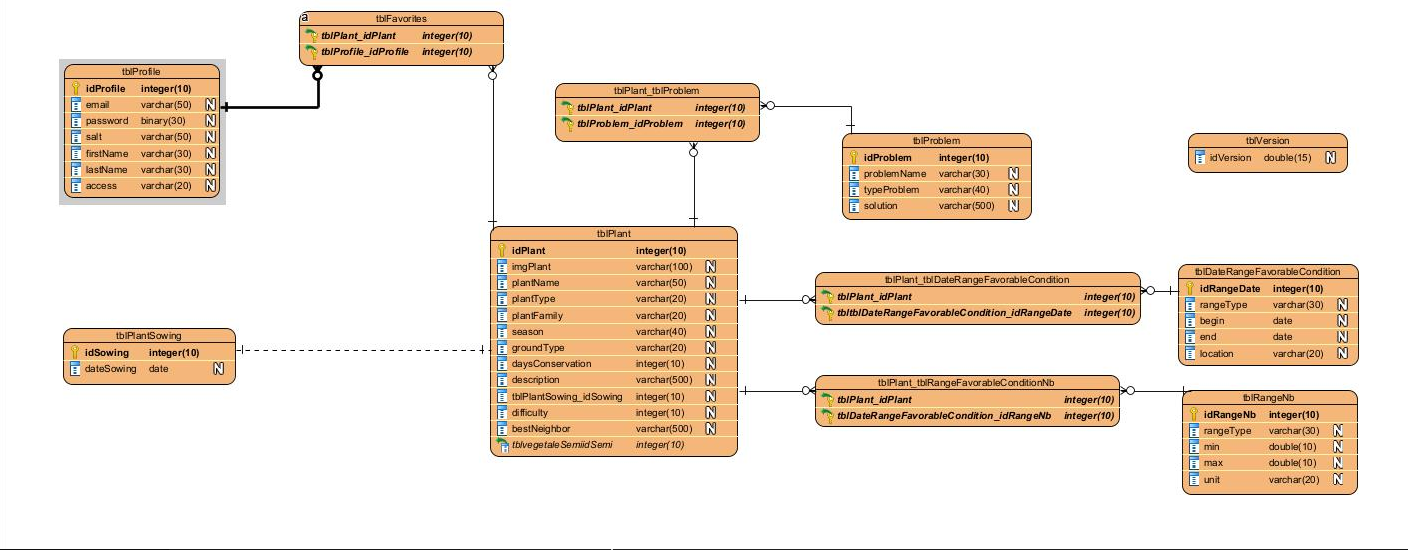
[**Modèle de données (Sprint 1)** 12](#_Toc90470492)

[Modèle conceptuel (V1) 12](#_Toc90470493)

[Modèle relationnel (V1) 12](#_Toc90470494)

[**Modèles de données (Sprint 2)** 13](#_Toc90470495)

[Modèle relationnel (V2) 13](#_Toc90470496)

[ 13](#_Toc90470497)

[Modification de la BD(V2) 13](#_Toc90470498)

[Modèle de données (Sprint 3) 14](#_Toc90470499)

[Modèle relationnel (V3) 14](#_Toc90470500)

[Modification de la BD(V3) 14](#_Toc90470501)

[**Maquettes visuelles** 15](#_Toc90470502)

[Maquettes mobiles 15](#_Toc90470503)

[Page d’accueil(V1) 15](#_Toc90470504)

[Maquettes web 16](#_Toc90470505)

[Page d’accueil(V1) 16](#_Toc90470506)

[Page d’info(V1) 16](#_Toc90470507)

[Page de connexion(V1) 16](#_Toc90470508)

[**Annexes** 18](#_Toc90470509)

[Test fonctionnels 18](#_Toc90470510)

[Documentation API 21](#_Toc90470511)

[**Sprint** 22](#_Toc90470512)

[Objectif du Sprint 1 22](#_Toc90470513)

[Conclusion du sprint 1 22](#_Toc90470514)

[Objectif du sprint 2 23](#_Toc90470515)

[Conclusion du sprint 2 23](#_Toc90470516)

[Objectif du Sprint 3 23](#_Toc90470517)

[Conclusion du sprint 3 24](#_Toc90470518)

[**Formation** 25](#_Toc90470519)

[API 25](#_Toc90470520)

[Ajout 25](#_Toc90470521)

[Modification 30](#_Toc90470522)

[Supprimer 33](#_Toc90470523)

[*Supprimer une Condition Favorable Date* 33](#_Toc90470524)

[*Supprimer une Condition Favorable Date* 33](#_Toc90470525)

[*Supprimer une famille* 33](#_Toc90470526)

[*Enlever l’assignement d’un problème à une plante* 33](#_Toc90470527)

[*Enlever l’assignement d’une condition favorable date à une plante* 33](#_Toc90470528)

[*Enlever l’assignement d’une condition favorable nombre à une plante* 34](#_Toc90470529)

[*Enlever l’assignement à un admin* 34](#_Toc90470530)

[Chercher 34](#_Toc90470531)

[*Chercher une plante* 34](#_Toc90470532)

[*Get un problème* 34](#_Toc90470533)

[*Get un profil* 34](#_Toc90470534)

[*Get une condition favorable date* 34](#_Toc90470535)

[*Get une condition favorable nombre* 34](#_Toc90470536)

[*Get all Plante* 34](#_Toc90470537)

[*Get all Problème* 34](#_Toc90470538)

[*Get all Profile* 34](#_Toc90470539)

[*Get all Condition favorable Date* 34](#_Toc90470540)

[*Get all Condition favorable Nb* 35](#_Toc90470541)

[*Get un package* 35](#_Toc90470542)

[*Get la dernière version de la BD* 35](#_Toc90470543)

[*Get les favoris d'un profile* 35](#_Toc90470544)

[Gestion de langue 36](#_Toc90470545)

[GitHub 38](#_Toc90470546)

[Pour commencer 39](#_Toc90470547)

[Pour changer le remote 39](#_Toc90470548)

[Quand une tâche est finie 39](#_Toc90470549)

[----On peut commencer à travailler----- 39](#_Toc90470550)

[Hostinger 40](#_Toc90470551)

[Outil Importation 42](#_Toc90470552)

[**Glossaire** 44](#_Toc90470553)

[**Médiagraphie** 45](#_Toc90470554)

[**Tableau des versions** 46](#_Toc90470555)

# **Introduction**

Le département informatique du cégep de Jonquière souhaite créer une application permettant à une serre de faire pousser ses plantes selon des conditions idéales, soit la température, l’éclairage, l’humidité de l’air et du sol. La consultation des données sera accessible par application web ou mobile. Également, l’application doit pouvoir fonctionner sans réseau Wi-Fi. Cette application ne sert qu’à des fins informatives et ne sera pas relié à une serre en temps réel, ce qui permet à tous de s’en servir sans posséder nécessairement une serre.

Le projet est pour le DICJ. Les objectifs principaux sont de :

* Développer l’interface web et mobile de l’application
* Développer la base de données de l’application
* Développer un système de profil et de favoris
* Développer un système de cache accessible hors ligne

## Rôles

Nous sommes une équipe composée dans le cadre du cours de projet qui a été contactée pour remplir ce mandat. Notre équipe est composé de :

* Francis Lavoie (Scrum master, Git Master et analyse qualité)
* Alec Lespérance (Responsable de la documentation, Recherche de technologies, du Client et Analyste)
* Maxime Lepage (Git Master, responsable du Backend, mise en production)
* Guillaume Dufour (Responsable de la communication entre les équipes et du Frontend)
* Alex Verreault (Responsable de la communication entre les équipes, Recherche de technologies, et du Frontend)
* Matis Larouche (Responsable de la communication avec le client et Analyste)

## Organisation

Afin de garder une certaine organisation dans l’équipe, nous avons intégrés des méthodes agiles :

* Scrum meeting à chaque jour avec tableau de temps et de tâches (10-20 minutes)
* Les scrums meeting du lundi sont orientés vers la planification de la semaine
* Planification sur Teams des tâches (Backlog)
* Un repos Git commun qui est divisé en branches propre à chaque tâche
* Nous avons tous choisi VSCode pour programmer et OneNote pour noter nos avancements journaliers
* Assurance qualité à chaque merge (Vérification des branches par le Git Master)
* Des normes de programmations ont été établies ainsi qu’une documentation sur celles-ci
* Environnement de test et de production
* Migrations permettant de garder des sauvegardes du projet

## Rencontre avec l’expert

Puisque notre projet est basé sur le partage d’informations, il nous fallait une source d’informations fiable. Pour ce faire, notre équipe a été référé a 2 expert, soit Mathieu Ouellette (Pépinière Boréalis) et Sarah Tremblay (La Rouquine), par notre professeur (Vincent Morin). Plusieurs questions leurs ont étés posées afin de discerner comment construire notre base de données et pour une meilleure compréhension du sujet.

### Rencontre Boréalis

Notre première rencontre fut avec Mathieu Ouellette, le propriétaire de la Pépinière Boréalis, jeudi le 16 septembre en après-midi. Alec, Matis et Maxime se sont rendus sur les lieux afin de lui poser leurs questions. Malheureusement, Mathieu a dû s’absenter, et nous a laissés avec son équipe. Malgré ceci, son équipe a très bien su nous répondre et nous a fait le tour du propriétaire. Vers la fin de notre séjour, nous avons pu échanger quelques questions avec Mathieu, notamment sur s’il connaissait des sources en ligne fiable que l’on pourrait utiliser. Celui-ci nous a référé à un bon livre nommé « Le jardinier maraîcher », où toutes les informations des plantes y sont notées.

### Rencontre La Rouquine

Notre deuxième rencontre fut avec Sarah Tremblay, propriétaire de La Rouquine, lundi le 20 septembre. Cette rencontre s’est effectué par téléphone, où seulement Alec et Mathis était présent. Puisque la plupart de nos questions avait déjà été répondu avec notre première rencontre, nous lui avons plus demandé comment elle opérait sa serre, afin d’avoir un point de vue différent.

# **Application similaire**

Dans l’optique de commercialiser ce projet, une recherche sur des applications similaires existantes a été faites.

## Apply Yourself App:

Apply Yourself est une application complètement polyvalente. Elle fonctionne sans problème sur téléphone portable ou tablette et est disponible pour les appareils Apple ou Android. Cette application est développée par Agri-Inject. Ses caractéristiques principales sont :

* Fonctionne avec toutes les unités métriques et anglaises.
* Cinq questions simples simplifient le choix et l'étalonnage de la pompe.
* Le système de classement intelligent s'ajuste en fonction de multiples facteurs pour déterminer la pompe qui est le mieux adaptée à votre application.
* Détails et photos fournis pour chaque modèle de pompe.
* L'utilisateur peut conserver une liste personnalisée des pompes pertinentes pour leur fonctionnement.
* Données d'étalonnage fournies dans trois réglages de l'heure communs.
* La minuterie est dotée d'alertes visuelles et sonores.
* Les événements de calibrage multiples sont simplifiés grâce aux fonctions simples de remise à zéro et de saisie des temps.

## Farmhand :

À l'aide de données en temps réel provenant de capteurs et de caméras à la ferme, Farmhand® Connect vous permet de suivre les conditions climatiques de votre ferme, d'établir des paramètres pour des conditions de croissance idéales et de recevoir des avis si des changements surviennent dans l'environnement.

Vous pouvez surveiller et contrôler la température, l'humidité, le CO2, les nutriments et le pH à l'intérieur de leur ferme, directement à partir de votre appareil iOS ou de votre navigateur Web. L'état en temps réel de chaque composant à l'intérieur de la ferme est facilement accessible, de sorte que vous savez toujours comment les choses fonctionnent. FH Connect offre un stockage en nuage sécurisé de toutes les données de la ferme, y compris l'historique du climat et des alarmes, les durées d'utilisation de l'équipement, les journaux d'accès et les informations de configuration du système.

# **Étude de technologies**

## Frontend :

Pour le Frontend, nous avons considéré Angular, React et Vue. Voici les avantages et désavantages de chacun :

### Solution #1 Angular

Premièrement, Angular est un Framework très populaire grâce à sa capacité de construire des applications complexes. Ses avantages sont :

* Model-View-Controller architecture
* Data binding double-sens pour réduire le temps de développement
* Les versions récentes d’Angular offre des façons plus robustes de gérer les erreurs

Ses désavantages sont:

* Le framework offre des ressources limitées pour le routage
* Le debugging est plus complexe en Angular
* Il est généralement difficile de rajouter du contenu externe

### Solution # 2 React

React est essentiellement une librairie Frontend JavaScript développé par Facebook en 2013 pour son interface utilisateur. React est surtout utilisé pour créer des pages web simple ou des applications mobiles. Ses avantages sont :

* React est basé sur JavaScript, ce qui lui permet de construire des interfaces dynamiques plus facilement puisqu’on peut développer des éléments HTML réutilisable.
* React utilise “Redux”, un container pour JavaScript permettant aux programmeurs de coder des applications pouvant se lancer dans plusieurs environnement, serveur, etc.
* React possède un data binding à sens unique, permettant un code plus stable

Ses désavantages sont :

* Plutôt complexe à apprendre
* Des nouveaux outils sont constamment en train de sortir, ce qui rend difficile le fait de se mettre à jour
* L’architecture de React change constamment

### **Solution #3 Vue**

Vue est un framework open-source JavaScript qui est primairement utilise pour le développement de page simple et d’interface. Ses avantages sont :

* Permet de créer des DOM virtuels afin d’en faire des copies
* Modèles de création interactive proposée
* Features data binding that helps users manipulate the elements of a webpage using a web browser
* Vue offre des contenant de HTML réutilisable

Ses désavantages sont :

* Puisque Vue est relativement nouveau, il est difficile de trouver de la documentation.

## Backend :

Pour le Backend, nous avons considéré Laravel.

### Solution #1 Laravel

Pour le développement de L'API nous avons décidé de choisir le framework Laravel pour PHP. Notre choix est basé sur le fait que nous connaissons déjà ce langage ce qui nous permettrait de sauver du temps sur l'apprentissage d'un nouveau langage ou d'un nouveau framework. Une autre raison majeure de ce choix est que notre API ne contiendra pas énormément de données et nous n'avons pas besoin d'une grosse structure pour soutenir le tout, donc Laravel subvient convenablement à nos besoins.

## Mode offline :

Après une recherche concernant le fonctionnement et le déploiement du mode hors-ligne de l’application, la solution qui serait la plus envisageable serait la méthode Cache First. Pour faire court, cette méthode consiste à télécharger les données et les sauvegarder dans le cache du navigateur de l’appareil mobile du client. Ainsi, le ou les clients auront donc la possibilité d’avoir un mode hors-ligne pour consulter la liste des végétaux disponible sur l’application. Une documentation plus approfondit se trouve à la fin de ce document.

# **Analyse globale**

## Fonctionnalités générales

Voici les fonctionnalités générales du projet

* Affichage de légumes et de fruits
* Système de recherche avec filtres
* Système de favoris
* Interface utilisateur et admin différentes pour permettre différentes fonctionnalités
* Système de redirection vers une plateforme d’achat

## Évaluation de l’envergure du projet

L’application contient les données sur les conditions optimales pour permettre la pousse d’une plante. L’application permettra de rechercher avec les filtres ou avec les favoris de l’utilisateur. L’administrateur pourra ajouter, modifier et supprimer des plantes ainsi que leurs informations. Ces données seront stockées dans une base de données.

## Liste des acteurs

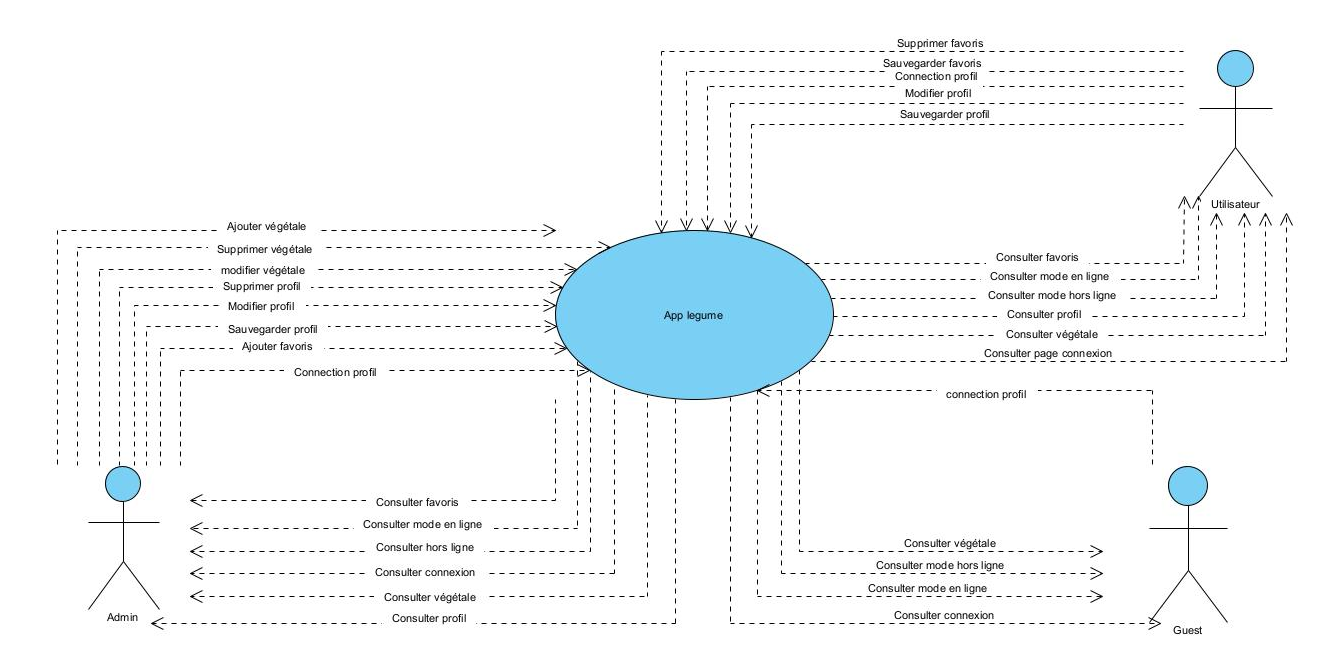
|  |  |
| --- | --- |
| **Acteurs** | **Descriptions** |
| Invité | L’invité peut seulement consulter les informations des plantes. |
| Utilisateur | L’utilisateur peut seulement consulter les informations des plantes et ajouter des favoris. |
| Administrateur | L’administrateur peut consulter/ajouter/modifier/supprimer des informations sur les plantes et les profils. |

## Liste des événements

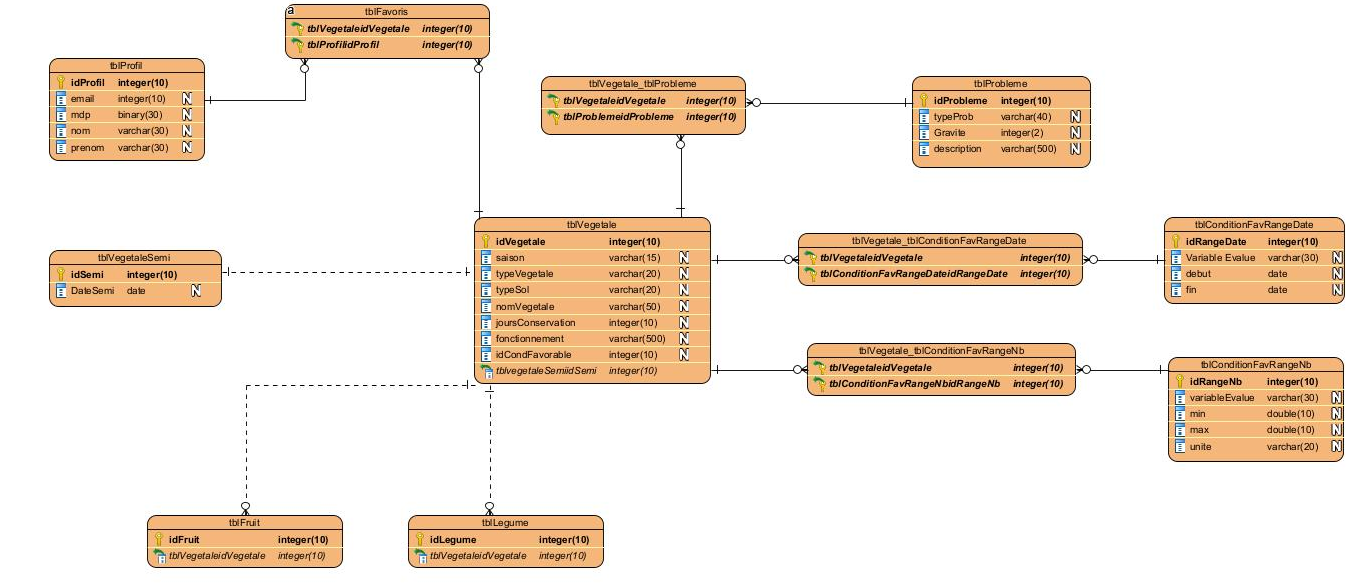
|  |  |
| --- | --- |
| **Évènements** | **Descriptions** |
| Ajouter Plantes | Affiche une page qui permet d’ajouter une plante à l’aide d’un bouton prévu à cet effet. |  |
| Consulter/Modifier/Supprimer Plantes | Affiche une interface qui permet d’accéder aux informations sur les plantes et permet d’en ajouter ou de les modifier/supprimer à l’aide d’un bouton prévu à cet effet. |  |
| Ajouter Profil | Affiche une page d’inscription et permet d’ajouter un profil à l’aide d’un bouton prévu à cet effet. |
| Consulter /Modifier/Supprimer Profil | Affiche les informations d’un profil et permet de les modifier/supprimer à l’aide d’un bouton prévu à cet effet. |
| Ajouter/Supprimer Favoris | Affiche la page des favoris, permettant ainsi d’en ajouter/supprimer à l’aide d’un bouton prévu à cet effet. |
| Consulter/Ajouter/Modifier/Supprimer Problèmes | Affiche une page de problèmes récurrent face à une plante en question, permettant ainsi d’en ajouter/supprimer/modifier à l’aide d’un bouton prévu à cet effet. |
| Ajouter/Supprimer/Modifier Conditions Favorables | Affiche une page permettant d’ajouter, de supprimer ou de modifier les conditions favorables d’une plante. |

# **Modèle de données (Sprint 1)**

## Modèle conceptuel (V1)



Modèle relationnel (V1)



# **Modèles de données (Sprint 2)**

## Modèle relationnel (V2)

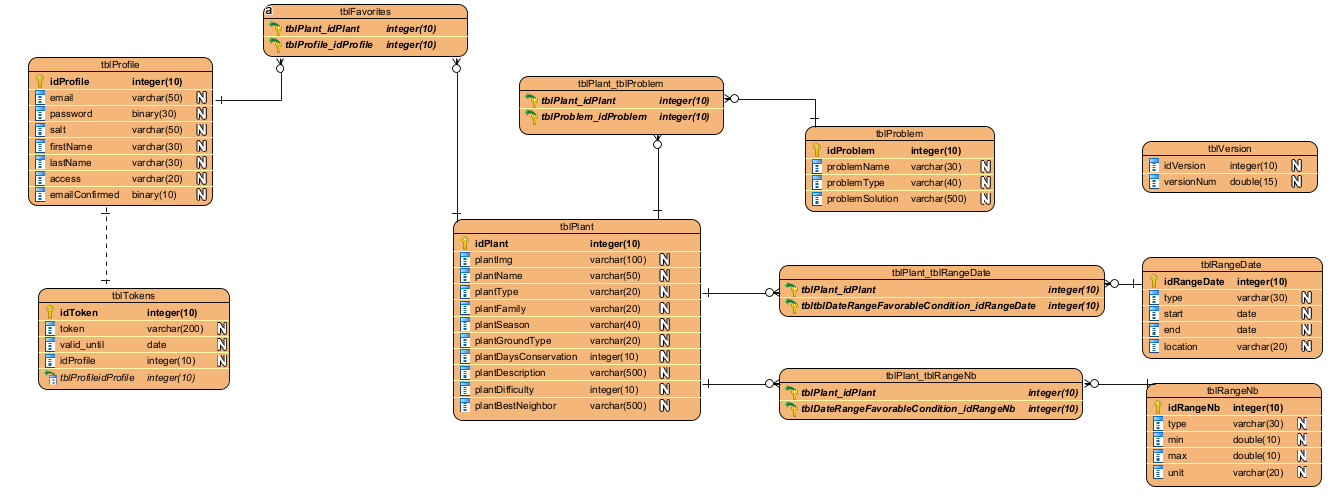
## 

## Modification de la BD(V2)

Lors de notre rencontre avec les experts de chez Boréalis et La Rouquine nous avons eu des informations supplémentaires qu’on a jugées important d’ajouter au projet. Nous avons attendu pendant Sprint 1 pour ne pas tout refaire et se concentrer sur faire quelque chose de fonctionnelle avant de modifier la BD plusieurs fois. Alors, pour le Sprint 2, nous avons fait les quelques modifications suivantes. Dans la table « tblProblem », nous avons changé la table « importanceLvl » pour « ProblemName », changer le champ description pour un champ « problemSolution », on a modifié le champ « typeProblem » pour « problemType »et nous avions eu pour but d’ajouter des images de problèmes, mais il n’y avait pas d’image d’espèces d’insectes spécifiques libres de droits alors on a laissé tomber l’idée. Dans la table « tblPlant », nous avons changé la valeur du varchar de « season » pour 40, nous avons ajouté les colonne « plantImg », « plantDifficulty », « plantFamily » et « plantBestNeighbor ». On a aussi changé le format d’écriture pour plant « variable » pour la table « tblPlant ». Nous avons aussi changé la colonne « functioning » pour description. On a ajouté Ajout la colonne « access » dans la table « tblProfile ». Ajout d’une table version, suppression des tables « tblFruit » et « tblVegetable », ajout d’un champ location et modifier le champs begin pour start dans la table « tblDateRangeFavorableCondition » où on a changé le nom de la table pour « tblRangeDate ». Pour finir on a ajouté un numéro de version en double dans la table « tblVersion » en mettant l’id en Int.

# Modèle de données (Sprint 3)

## Modèle relationnel (V3)



## Modification de la BD(V3)

Nous avons changé la variable de champ « plantDifficulty » pour un integer puisqu’on veut gérer le niveau de difficulté avec des nombres au lieu d’utiliser des chaines de caractères puisqu’il y a moins de place à l’erreur. Sinon, nous avons enlevé la table « tblPlantSowing » puisqu’elle était inutile a notre application. Pour finir nous avons ajouté une table « tblToken » qui sert à la création de token pour l’authentification.

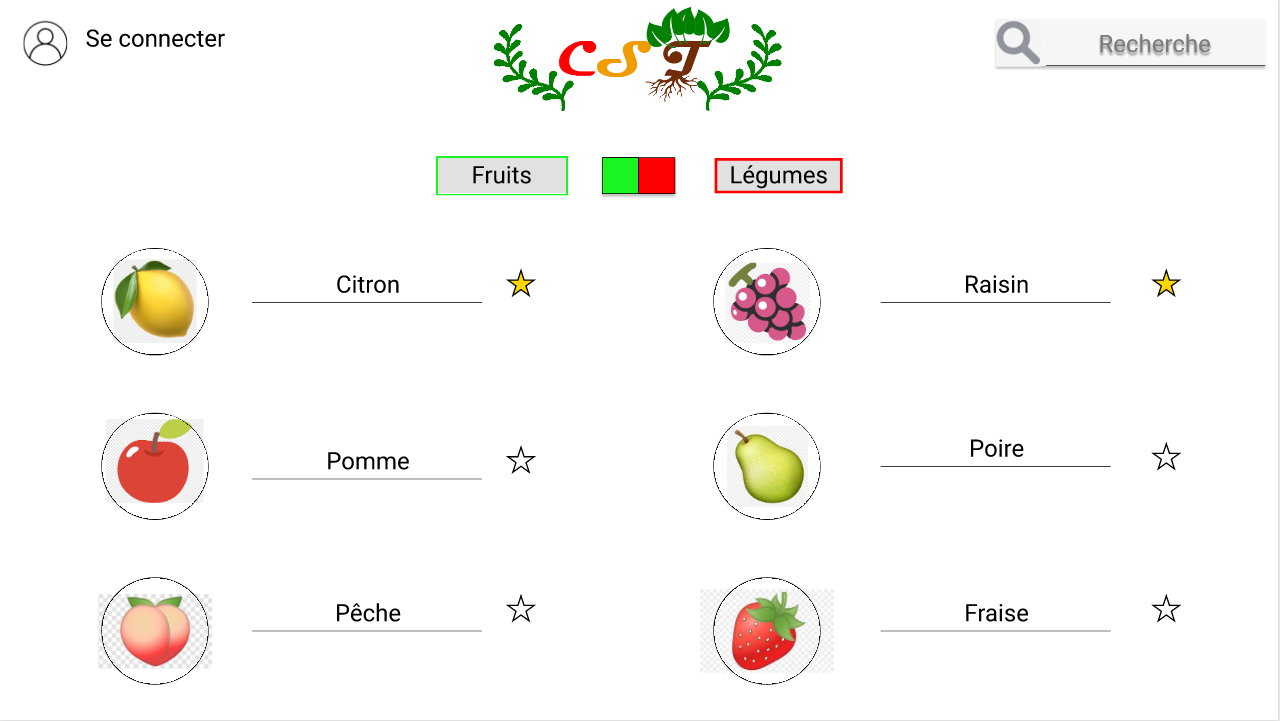
# **Maquettes visuelles**

## Maquettes mobiles

Page d’accueil(V1) Page d’info(V1)

## Maquettes web

### Page d’accueil(V1)



### Page d’info(V1)

### Page de connexion(V1)



# **Annexes**

## Test fonctionnels

#001 **Ajout d’une plante dans la base de données**

Acteur Administrateur

Prérequis Une base de données existante

Opérations

1. L’administrateur remplit les champs d’une plante.
2. L’administrateur crée une plante.
3. Le système accepte d’ajouter la plante dans la base de données.

Résultats attendus

* Un message d’ajout est affiché, expliquant que la plante a été ajouté avec succès.
* Le système ajoute la plante.

#002 **Ajout d’une plante invalide dans la base de données**

Acteur Administrateur

Prérequis Une base de données existante

Opérations

1. L’administrateur remplit les champs d’une plante incorrectement.
2. L’administrateur crée une plante.
3. Le système refuse d’ajouter la plante dans la base de données.

Résultats attendus

* Un message d’erreur est affiché, expliquant de corriger l’erreur.
* Le système supprime la plante récemment ajoutée.

#003 **Modification d’une plante dans la base de données**

Acteur Administrateur

Prérequis Une base de données ainsi qu’une plante existante

Opérations

1. L’administrateur ouvre la page de détail de la plante qu’il souhaite modifier.
2. L’administrateur appuie sur le bouton « Modifier ».
3. L’administrateur modifie les champs selon les nouvelles données.
4. L’administrateur sauvegarde ses changements à l’aide du bouton « Sauvegarder ».

Résultats attendus

* Un message de sauvegarde est affiché, expliquant que les changements ont été sauvegardés.
* Le système modifie la plante selon les changements apportés.

#004 **Suppression d’une plante dans la base de données**

Acteur Administrateur

Prérequis Une base de données ainsi qu’une plante existante

Opérations

1. L’administrateur ouvre la page de détail de la plante qu’il souhaite supprimer.
2. L’administrateur appuie sur le bouton « Supprimer ».

Résultats attendus

* Un message est affiché, expliquant que la plante a été supprimé avec succès.
* Le système supprime la plante.

#005 **Création d’un profil**

Acteur Invité

Prérequis Une base de données existante

Opérations

1. L’invité clique sur le bouton « Inscription ».
2. L’invité remplit les champs selon ses informations.
3. L’invité appuie sur le bouton « Créer mon profil ».

Résultats attendus

* Un message de succès est affiché, expliquant que le profil a été créer, et l’invité est connecté sur son profil d’utilisateur.
* Le système sauvegarde le profil.

#006 **Création incorrecte d’un profil**

Acteur Invité

Prérequis Une base de données existante

Opérations

1. L’invité clique sur le bouton « Inscription ».
2. L’invité remplit incorrectement les champs.
3. L’invité appuie sur le bouton « Créer mon profil ».

Résultats attendus

* Un message d’erreur est affiché, expliquant que le profil n’a pas été créer, et que l’invité doit corriger les champs invalides.
* Le système ne sauvegarde pas le profil.

#007 **Modification d’un profil**

Acteur Utilisateur

Prérequis Une base de données ainsi qu’un profil existant

Opérations

1. L’utilisateur clique sur le bouton « Mon profil ».
2. L’utilisateur modifie ses champs avec ses nouvelles informations.
3. L’utilisateur appuie sur le bouton « Sauvegarder mon profil ».

Résultats attendus

* Un message de succès est affiché, expliquant que le profil a été modifié et l’utilisateur est ramené à la page d’accueil.
* Le système modifie le profil.

#008 **Suppression d’un profil dans la base de données**

Acteur Administrateur

Prérequis Une base de données ainsi qu’un profil existant

Opérations

1. L’administrateur ouvre la page de détail du profil qu’il souhaite supprimer.
2. L’administrateur appuie sur le bouton « Supprimer ».

Résultats attendus

* Un message est affiché, expliquant que le profil a été supprimé avec succès.
* Le système supprime le profil.

#009 **Ajout d’un problème dans la base de données**

Acteur Administrateur

Prérequis Une base de données existante

Opérations

1. L’administrateur remplit les champs d’un problème.
2. L’administrateur crée un problème.
3. Le système accepte d’ajouter le problème dans la base de données.

Résultats attendus

* Un message d’ajout est affiché, expliquant que le problème a été ajouté avec succès.
* Le système ajoute le problème.

#010 **Ajout d’un problème invalide dans la base de données**

Acteur Administrateur

Prérequis Une base de données existante

Opérations

1. L’administrateur remplit les champs d’un problème incorrectement.
2. L’administrateur crée un problème.
3. Le système refuse d’ajouter le problème dans la base de données.

Résultats attendus

* Un message d’erreur est affiché, expliquant de corriger l’erreur.
* Le système supprime le problème récemment ajoutée.

#011 **Modification d’un problème**

Acteur Utilisateur

Prérequis Une base de données ainsi qu’un problème existant

Opérations

1. L’utilisateur clique sur le bouton « Modifier ».
2. L’utilisateur modifie ses champs avec ses nouvelles informations.
3. L’utilisateur appuie sur le bouton « Sauvegarder mon problème ».

Résultats attendus

* Un message de succès est affiché, expliquant que le problème a été modifié et l’utilisateur est ramené à la page d’accueil.
* Le système modifie le problème.

#012 **Suppression d’un problème dans la base de données**

Acteur Administrateur

Prérequis Une base de données ainsi qu’un problème existant

Opérations

1. L’administrateur ouvre la page de détail du problème qu’il souhaite supprimer.
2. L’administrateur appuie sur le bouton « Supprimer ».

Résultats attendus

* Un message est affiché, expliquant que le problème a été supprimé avec succès.
* Le système supprime le problème.

# **Sprint**

## Objectif du Sprint 1

Pour le Sprint 1, nous avons pour objectif de faire un API fonctionnel avec un affichage minimal. Nous devons faire une base de donnée où nous allons mettre des données de test pour tester que notre API fonctionne. Un autre objectif est de créer un API le plus rapidement possible pour que le Back-End puisse l’utiliser pour vérifier leurs données avec les nôtres. Nous allons créer un document Excel qui va contenir un certain nombre de plante qui va être utilisé pour entrer des données dans la BD. Afin de vérifier que nos champs et nos données sont adéquates, nous allons rencontrer des experts dans le domaine des serres pour leur poser nos questions et améliorer notre projet. Sinon, nous allons commencer à regarder de la documentation sur le mode hors ligne. Pour finir, nous allons faire une assurance qualité du code ainsi que des tests unitaires pour vérifier que tout fonctionne bien.

## Conclusion du sprint 1

Au niveau du Sprint 1, cela a généralement bien été malgré qu’on ait eu quelques problématiques durant les quatre semaines de travail. En effet, nous avons eu plusieurs problèmes qu’on a surmonté tandis que d’autres ont été remis au Sprint 2. Pour commencer, nous avons eu de la difficulté à monter les environnements puisque les VM nous mettaient des bâtons dans les roues et parce qu’on n’a pas monté toutes les environnements en même temps. Sinon, il y a eu quelques problèmes au niveau de l’ajout de Cors puisque la version 6 de Laravel ne prenait pas Cors comme un login ce qui a amené à ajouter un fichier module dans le HT Access pour l’intégrer. Nous avons aussi créé des branches pour chaque membre de l’équipe au lieu de créer une branche par tâche ce qui rendait les mises en productions plus compliquées ce qu’on va changer pour le Sprint 2. Ensuite, nous avons eu de la difficulté à intégrer l’offline avec Chrome parce que cela ne voulait pas télécharger puisqu’il fallait avoir accès à internet, mais après un peu de trouble shooting nous avons réussis à l’intégrer. Sinon, nous avons eu quelques difficultés avec le token Git puisqu’il changeait souvent donc nous devions toujours changer les URL et comprendre pourquoi nos connexions à notre environnement étaient refusées. Par la suite, un des gros points majeurs qui a touché plusieurs membres de l’équipe étaient de se remettre dans le bain puisque cela faisait longtemps qu’on n’avait pas programmé, donc nous avions de la difficulté à produire du code avancé. Il y a eu des problèmes de composants qui ont été simple à régler. Il y a seulement eu à refaire le processus du début et tout était réglé. Sinon, le HasMAny était difficile à apprendre, et pour finir nous avons eu des problèmes avec l’ajout de favoris par URL puisqu’on pouvait pas chercher les Id, alors nous avons créé une page avec une fonction de redirection pour changer l’URL pour la route.

Par ailleurs, nous avons réussis tous nos objectifs qui étaient de créer un API fonctionnel qui accède à une BD fonctionnelle avec un affichage minimal. Nous avons même été au-delà de nos espérances en intégrant plusieurs tests unitaires, en testant les pages, en intégrant le filtre de fruit et légume et en réussissant à intégrer l’Offline à notre projet. Au niveau des objectifs, non atteints nous en avons aucun, nous avons tous réussis à faire ce qu’on espérait accomplir durant le sprint.

## Objectif du sprint 2

Au niveau des objectifs du Sprint 2, nous avons pour but d’intégrer les profils et leurs accès, intégrer des formulaires et un affichage pour ces mêmes formulaires, ajouter de vrais problèmes dans la BD, améliorer le visuel des pages interactives , intégrer le filtre alphabétique, le filtre de famille des plantes, le filtre de difficultés et le filtre de durée de temps, ajuster la BD avec les informations de l’expert, intégrer les versions au Offline tout en rendant les profils accessibles en offline, faire des tests sur les post , les « Get », les « Delete » et pour les modifications et pour finir une assurance qualité sur tous les aspects du Sprint 2.

## Conclusion du sprint 2

En ce qui concerne les objectifs fixés pour le Sprint 2 nous avons réussis à tout faire ce que nous avions pour objectif de faire à l’exception des filtres et l’intégration de vrais problèmes au projet. Nous avons réussi à intégrer des profils et leurs accès ainsi qu’un formulaire d’inscription fonctionnel. Nous avons ajusté la BD avec les informations que Boréalis nous a fournis. Nous avons fait les tests de Get, Post, Update, et Delete pour tout le projet. Par contre, nous avons eu quelques problèmes avec les tests « UpdateProfile » et « GetProfile ». Nous avons pris une semaine du sprint pour faire l’assurance qualité et merge toutes les parties des membres de l’équipe. Nous avons intégré un système de versions au mode hors-ligne et nous avons laissé tomber l’idée de rendre les profils accessibles en hors-ligne puisqu’on ne voyait pas l’intérêt de les avoir et ça aurait été trop compliqué. Pour finir, nous avons fait un affichage acceptable pour la majorité des pages du site web qui seront modifiées au Sprint 3 pour qu’elles soient encore plus belle.

Au niveau des problèmes, nous en avons eu quelques-uns qui ont ralentis notre progression générale. Tout d’abord, Guillaume a eu un problème avec la machine virtuelle qui a durée une semaine puisqu’elle bloquait des paramètres pour la création du HMI. Après avoir recherché des solutions il a tout réinstallé ce qu’il a installé au départ et ça à fonctionner. Par la suite, nous voulions intégrer Passport au projet pour la gestion des profils qui était trop compliquer pour pas grand-chose. Nous avons passé une semaine à essayer de l’intégrer pour au final abandonner l’idée et fonctionner avec des middlewares qui ont pris 2 journées à configurer. Pour finir, Alex à mal compris ce qu’il avait à faire et il a créé des conflits de routes avec les routes de l’API originales en créant des routes différentes qui mène au même fonction.

## Objectif du Sprint 3

Notre objectif de Sprint 3 est majoritairement de la finalisation puisque c’est notre dernier Sprint et que nous avons assez de fonctionnalités pour livrer une version utilisable. Nous allons corriger les bugs de l’API et terminer les dernières fonctionnalités, terminer les tests unitaires significatifs, terminer les tableaux Excel de plantes et de problèmes, terminer la documentation, améliorer l’interface et ses filtres et pour finir préparer une présentation finale du projet pour les 1ères et 2èmes années de la technique.

## Conclusion du sprint 3

Lors du Sprint 3, nous avons vider notre back log pour finir tous les objectifs que nous nous sommes fixé. Premièrement, nous avons essayé d’ajouter une confirmation de création de profil par mail mais c’était trop compliqué à intégrer pour le nombre de semaine qui nous restait à travailler. Nous avons aussi intégré un système de token pour l’authentification des profils et pour rester connecté quand on ferme le site web et nous l’avons intégré au HMI. Nous avons intégré l’interface que Alex a fait au niveau des routes d’API. Nous avons ajouté des images de fruits et légumes au HMI. Sinon, on a refait toute l’interface web entièrement au niveau desktop et nous avons fait l’affichage du mobile entièrement. Nous avons ajouté les problèmes que nous avons regroupé dans un tableau Excel ainsi que leurs liaisons aux plantes dans la base de donnée. Sinon, on a fait une documentation de toute notre environnement et sur comment l’utiliser pour les autres équipes qui seront attribué à notre projet. Par la suite, nous avons intégré les affichages des problèmes et des conditions favorables. Nous avons aussi intégré une gestion de langue qui gère les traductions anglaises et françaises. Pour finir, nous avons préparé notre présentation de notre projet pour les 1ères années.

Nous avons eu quelques problèmes qui nous ont causés quelques pertes de temps. Premièrement, nous avons eu de la difficulté à intégrer le système de token puisque les middlewares ne fonctionnaient pas avec les routes existantes. Sinon, il y a eu beaucoup de conflit lorsqu’on a intégré l’affichage de Alex puisqu’il a changé des routes. Par la suite, nous avons eu des problèmes avec l’outil d’importation ainsi que des bugs avec les données Excel parce que la taille des varchar étaient pas assez élevées ou la variable n’était pas la bonne. Sinon, toute l’équipe a eu des problèmes de Cors alors nous avons intégré un composant Cors Everywhere à nos navigateurs qui a corrigé notre bug. Par la suite, Guillaume a eu des difficultés à merge, il a eu des problèmes de machine virtuelle et il a eu des difficultés avec le CSS des components. Marco de son côté avait de la difficulté à avoir les données quand il lançait l’application, des fois il en avait des fois il en avait pas. Nous avons eu un problème avec Safari qui ne voulait pas ouvrir notre site web. Alors, nous avons joué dans les paramètres de Safari dans les réglages et ça a fonctionné. Pour finir, nous avons eu des problèmes dans la gestion de langue au niveau des ports et des versions des composants ajoutés au projet.

# **Formation**

Dans cette section, nos diverses technologies vous seront expliquées afin de pouvoir transmettre notre projet à une nouvelle équipe de travail au cas où il y aurait une suite. Cette partie a pour but de montrer ce que nous avons implémenté et comment l’utiliser en tant que développeur.

## API

L’API de l’aide à la décision du projet Cegep Serre Tech permet de créer des profils, des plantes, des problèmes reliés aux plantes, des conditions associées aux plantes ainsi que des favoris pour un compte.

Liens vers l’interface de L’API : http://apipcst.xyz/fr

Voici toutes les actions possibles d’effectuer sur notre API :

### Ajout

#### Ajouter une plante

* L’id de la plante
* Une image pour la plante
* Le nom de la plante
* Le type de plante
* La famille de la plante
* Les saisons favorables pour la plante
* Le type de sol favorable pour la plante
* Le nombre de jour que la plante se conserve
* Une description de la plante
* Le niveau de difficulté à la faire pousser
* Les plantes avoisinantes favorable pour la plante



Via un URL :

<http://apipcst.xyz/api/new/plant/addPlant> (POST)

{

        "plantImg": "https://static.wikia.nocookie.net/minecraft\_gamepedia/images/5/54/Golden\_Apple\_JE2\_BE2.png/revision/latest/scale-to-width-down/160?cb=20200521041809",

        "plantName": "Golden Apple",

        "plantType": "Fruit",

        "plantFamily": "Courgeticus",

        "plantSeason": "Spring",

        "plantGroundType": "Dirt",

        "plantDaysConservation": 1,

        "plantDescription": "Plant it",

        "plantDifficulty": 1,

        "plantBestNeighbor": "Cucumber "

    }

#### Un problème

* L’id du problème
* Le type de problème
* La solution au problème
* Le nom du problème

Une image contenant texte

Description générée automatiquement

Via un URL :

<http://apipcst.xyz/api/new/problem/addProblem>  (POST)

{

        "problemName": "Marmotte",

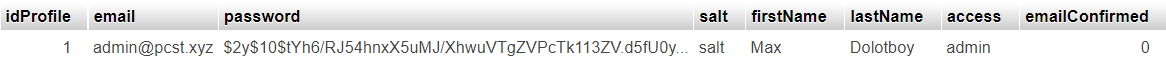
        "problemType": "Vermine",

        "problemSolution": "Insecticide"

 }

#### Un profil

* L’id du profil
* Le courriel du profil
* Le mot de passe du profil
* Le sel pour encrypter le profil (Ne concerne pas les utilisateurs)
* Prénom
* Nom
* Les accès du profil



Via un URL :

<http://apipcst.xyz/api/new/profile/addProfile>  (POST)

{

        "email": "homer.simpsons@yellowworld.com",

        "password": "donut69",

        "firstName": "Homer",

        "lastName": "Simpsons",

        "access": "admin"

     }

#### Un favori

* L’id de la plante
* L’id du profil



Via un URL :

<http://testenv.apipcst.xyz/api/new/favorite/1/1>  (POST, IdPlant, IdProfile)

#### Une condition favorable de date

* L’id de la condition
* Le type de condition
* La date de début
* La date de fin
* L’endroit concerné par ces dates



Via un URL :

<http://apipcst.xyz/api/new/condition/addFavCondDate> (POST)

{

        "type": "Limite de plantaison",

        "start": "2021-12-01",

        "end": "2021-12-31",

        "location": "Québec"

     }

#### Une condition favorable de nombre

* L’id de la condition
* Le type de condition
* Le minimum
* Le maximum
* L’unité utilisée



<http://apipcst.xyz/api/new/condition/addFavCondNb> (POST)

{

        "type": "Limite de plantaison",

        "min": "69",

        "max": "420",

        "unit": "Temperature"

     }

#### Une famille

* L’id de la famille
* Le nom de la famille

Une image contenant texte

Description générée automatiquement

Via un URL :

<http://apipcst.xyz/api/new/family/addFamily>  (POST)

{

        "familyName": "Rosacée"

}

#### Assigner une condition favorable de date à une plante

* L’id de la plante
* L’id de la condition



Via un URL :

<http://apipcst.xyz/api/assign/condition/date/1/1>  (POST, IdPlant, IdCondition)

#### Assigner une condition favorable de nombre à une plante

* L’id de la plante
* L’id de la condition

Une image contenant texte

Description générée automatiquement

Via un URL :

<http://apipcst.xyz/api/assign/condition/nb/1/8>  (POST, IdPlant, IdCondition)

#### Assigner un problème à une plante

* L’id de la plante
* L’id du problème



Via un URL :

<http://testenv.apipcst.xyz/api/assign/problem/1/4>  (POST, IdPlant, IdProblem)

#### Ajouter un utilisateur en tant qu’admin

* L’id du profil

Via un URL :

<http://apipcst.xyz/api/new/admin/addAdmin/2>  (PUT, IdProfile)

### Modification

#### Modifier une plante

* Une image pour la plante
* Le nom de la plante
* Le type de plante
* La famille de la plante
* Les saisons favorables pour la plante
* Le type de sol favorable pour la plante
* Le nombre de jour que la plante se conserve
* Une description de la plante
* Le niveau de difficulté à la faire pousser
* Les plantes avoisinantes favorable pour la plante



Via un URL :

<http://apipcst.xyz/api/edit/plant/editPlant/1>  (PUT, IdPlant)

{

        "plantImg": "https://static.wikia.nocookie.net/minecraft\_gamepedia/images/5/54/Golden\_Apple\_JE2\_BE2.png/revision/latest/scale-to-width-down/160?cb=20200521041809",

        "plantName": "Notch Golden Apple",

       "plantType": "Fruit",

        "plantFamily": "Courgeticus",

        "plantSeason": "Spring",

        "plantGroundType": "Dirt",

        "plantDaysConservation": 1,

        "plantDescription": "Just fucking plant it",

        "plantDifficulty": 1,

        "plantBestNeighbor": "otjfhj"

}

#### Modifier un problème

* Type de problème
* Solution au problème
* Nom du problème

Une image contenant texte

Description générée automatiquement

Via un URL :

<http://apipcst.xyz/api/edit/problem/editProblem/1>  (PUT, IdProblem)

{

        "problemType": "Vermine",

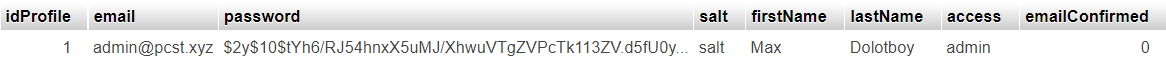
        "problemSolution": "Tu la neye",

        "problemName": "Une criss de marmotte bouffe la salade"

}

#### Modifier un profil

* Le courriel du profil
* Prénom
* Nom
* Accès du compte



Via un URL :

<http://apipcst.xyz/api/edit/profile/editProfile/1>  (PUT, IdProfile)

{

        "email": "homer.simpsons@yellowworld.com",

        "firstName": "Homerrrrrr",

        "lastName": "Simpsons",

        "access": "user"

 }

#### Modifier une condition favorable date

* Type de condition
* Date de début de la condition
* Date de fin de la condition
* Endroit concerné par la condition



Via un URL :

<http://apipcst.xyz/api/edit/condition/editFavCondDate/1> (PUT, IdCondition)

{

        "type": "Limite de plantaison",

        "start": "2021-12-01",

        "end": "2021-12-31",

        "location": "Cancune"

 }

#### Modifier une condition favorable nombre

* Type de condition
* Minimum
* Maximum
* Unité utilisée



Via un URL :

[http://apipcst.xyz[/api/edit/condition/editFavCondNb/1](http://testenv.apipcst.xyz/api/edit/condition/editFavCondNb/1)](http://apipcst.xyz/api/edit/condition/editFavCondDate/1) (PUT, IdCondition)

{

        "type": "Limite de plantaison",

        "min": "59",

        "max": "789",

        "unit": "Québec"

 }

#### Modifier une famille

* Nom de la famille

Une image contenant texte

Description générée automatiquement

Via un URL :

<http://apipcst.xyz/api/edit/family/editFamily/1> (PUT, IdFamily)

{

        "familyName": "Blabla"

}

### Supprimer

#### Supprimer une plante

<http://apipcst.xyz/api/delete/plant/1> (DELETE, IdPlant)

#### Supprimer un problème

<http://apipcst.xyz/api/delete/problem/1> (DELETE, IdProblem)

#### Supprimer un favori

<http://apipcst.xyz/api/delete/favorite/1/1> (DELETE, IdPlant, IdProfile)

#### Supprimer une Condition Favorable Date

<http://apipcst.xyz/api/delete/condition/date/1> (DELETE, IdPlant, IdCondition)

#### Supprimer une Condition Favorable Date

<http://apipcst.xyz/api/delete/condition/number/1> (DELETE, IdPlant, IdCondition)

#### Supprimer une famille

<http://apipcst.xyz/api/delete/family/1> (DELETE, IdFamily)

#### Enlever l’assignement d’un problème à une plante

<http://apipcst.xyz/api/unassign/problem/1/4> (DELETE, IdPlant, IdProblem)

#### Enlever l’assignement d’une condition favorable date à une plante

<http://apipcst.xyz/api/unassign/condition/date/1/1> (DELETE, IdPlant, IdCondition)

#### Enlever l’assignement d’une condition favorable nombre à une plante

<http://apipcst.xyz/api/unassign/condition/nb/1/8> (DELETE, IdPlant, IdCondition)

#### Enlever l’assignement à un admin

<http://apipcst.xyz/api/delete/admin/removeAdmin/2>  (PUT, IdProfile)

### Chercher

#### Chercher une plante

[http://apipcst.xyz/api/search/plant/1](http://testenv.apipcst.xyz/api/search/plant/1) (GET) (Condition id)

#### Get un problème

<http://apipcst.xyz/api/search/problem/1> (GET) (Condition id

#### Get un profil

<http://apipcst.xyz/api/search/profile/1> (GET) (Condition id)

#### Get une condition favorable date

<http://apipcst.xyz/api/search/condition/date/1> (GET) (Condition id)

#### Get une condition favorable nombre

[http://apipcst.xyz/api/search/condition/nb/2](http://apipcst.xyz/api/search/condition/nb/2 )(GET) (Condition id)

#### Get all Plante

<http://apipcst.xyz/api/searchAll/plant> (GET)

#### Get all Problème

<http://apipcst.xyz/api/searchAll/problem> (GET)

#### Get all Profile

<http://apipcst.xyz/api/searchAll/profile> (GET)

#### Get all Condition favorable Date

<http://apipcst.xyz/api/searchAll/condition/date> (GET)

#### Get all Condition favorable Nb

<http://apipcst.xyz/api/searchAll/condition/nb> (GET)

#### Get un package

<http://apipcst.xyz/api/search/package/2> (GET, IdPlant)

#### Get la dernière version de la BD

<http://apipcst.xyz/api/search/last/version> (GET)

#### Get les favoris d'un profile

<https://apipcst.xyz/api/searchAll/favorite/1>  (GET, IdProfile)

Vérifier des credentials:

<https://apipcst.xyz/api/login/checkLogin/> (POST)

{

"email" : "homer.simpsons@yellowworld.com",

"password" : "donut69"

}

## Gestion de langue

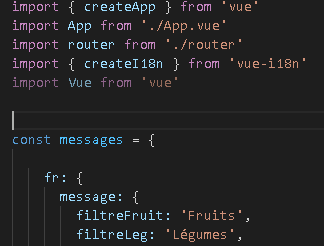
Afin de rendre notre projet disponible dans plusieurs langues , nous avons ajouté un package de gestion de langue qui se nomme vue-i18n. Nous avons utilisé une version du package qui est compatible avec Vue3 puisque nous utilisons la version 3 de vue dans notre projet.

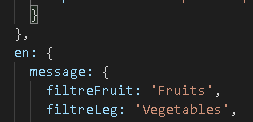
Tout d’abord, il faudra cloner notre projet pour ensuite lancer la commande suivante dans un invite de commande au chemin .\PCST\GreenHouse-Tech\HMI\node\_modules : **npm install vue-i18n@next** . Après avoir entré la commande, vous devriez être capable de voir la gestion de langue lorsque vous lancez le serveur web.

Pour faire une traduction dans une vue ou un fichier html, il faut ajouter des lignes de ce format dans le html : {{ $t("message.traductionMessage") }}. Voici un exemple :

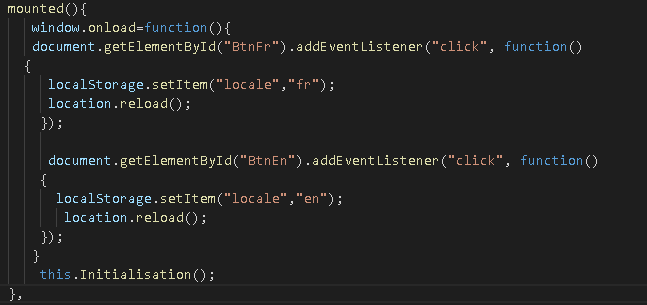


Par la suite, dans le main.js nous avons toutes les traductions. Nous allons devoir créer une instance traductionMessage dans chaque langue de traduction dans la valeur message. Voici un exemple :





Pour finir, la gestion de langue se gère avec des clés dans le localStorage et cela est géré dans le App.vue de la ligne 75 à 87. Lorsque le bouton est activé nous devons recharger la page pour que les modifications soient faites d’où le pourquoi il y a un reload dans les deux boutons. Voici le code de la gestion des boutons :



## GitHub

Durant le projet, nous avons dû utiliser l’outil de gestion de versions Git ainsi que le service web d’hébergement de projet GitHub. Nous nous en sommes servi pour pouvoir implémenter des parties au projet après la réalisation de chaque tâche. Au tout début du projet, nous avons adopté une méthode d’utilisation de GitHub qui consistait à avoir chacun une branche à notre nom et pousser nos modifications à l’intérieur de cette même branche, puis nous faisions un « merge » de tous les branches dans le « main » à la fin du sprint. À la fin du premier sprint, nous avons remarqué que ce n’était pas une bonne méthode de travail et ce n’est pas une bonne façon d’utiliser GitHub. Nous avons donc mis au point une nouvelle façon de travailler avec GitHub et les tâches que nous avions à réaliser. À chaque fois qu’un membre de l’équipe avait une nouvelle tâche d’assigner, il allait écrire le numéro de la tâche, son nom et la date dans un tableau dans OneNote, donc on pouvait garder une trace de qui s’est assignée quelle tâche. Ensuite, sur GitHub, il allait se créer une branche avec le nom « request#??? » les points d’interrogation étant bien sur le numéro de la tâche. Une fois que la requête avait été effectuée, la personne effectuait un « push » sur sa branche pour ensuite faire un « pull » de la branche « dev » sur sa branche pour pouvoir régler les conflits et s’assurer que tout fonctionnerait bien une fois dans le dev. Si tout fonctionnait et que l’assurance qualité avait été faite, il allait sur la branche « dev » et effectuait un « merge » avec sa branche. Donc, nous avons utilisé GitHub pour garder des traces des versions antérieures, modifications précédentes, des personnes qui ont effectué des modifications quelconques ainsi que pour pouvoir se partager les parties de notre projet entre les membres de notre équipe.

Tout au long du projet, nous avons utilisé ces commandes dans Git pour nous permettre de bien l’utiliser

* Git init – Pour initier un repository Git local
* Git clone – Pour cloner un repository Git sur le poste local
* Git branch – Pour créer une branche
* Git checkout – Pour changer de branche
* Git pull origin – Pour aller chercher la dernière version d’une branche
* Git push origin – Pour pousser les modifications dans une branche
* Git merge – Pour synchroniser deux branches ensemble
* Git remote -v – Pour savoir à quelle repository le projet actuel est connecté
* Git remote add origin – Pour ajouter un lien vers un repository
* Git remote rm origin – Pour supprimer un lien vers un repository
* Git add – Pour ajouter les modifications effectuées
* Git commit – Pour commit les modifications effectuées
* Git status – Pour voir les fichiers modifiés ou non depuis la dernière version

Voici la petite documentation que nous nous étions faits au cours de notre projet pour nous aider à utiliser Git.

Lien du repos : <https://github.com/Dolotboy/GreenHouse-Tech>

### Pour commencer

Git clone <https://github.com/Dolotboy/GreenHouse-Tech>

Git branch VOTRENOM

Git checkout VOTRENOM

Git remote set-url origin git remote add origin <https://github.com/Dolotboy/GreenHouse-Tech.git>

### Pour changer le remote

git remote rm origin

git remote add origin <https://github.com/Dolotboy/GreenHouse-Tech.git>

### Quand une tâche est finie

git add …

git commit –m "Push request#…"

git push origin request#...

git checkout dev

git pull origin dev

git merge request#…

git push origin dev

git branch (nouvelle requete)

git checkout (nouvelle branche)

### ----On peut commencer à travailler-----

Git status

Git add \*.extension

Git status

Git commit -m "Message significatif"

Git pull origin VOTRENOM

Git push origin VOTRENOM

**Important de faire une branche à votre nom !!**

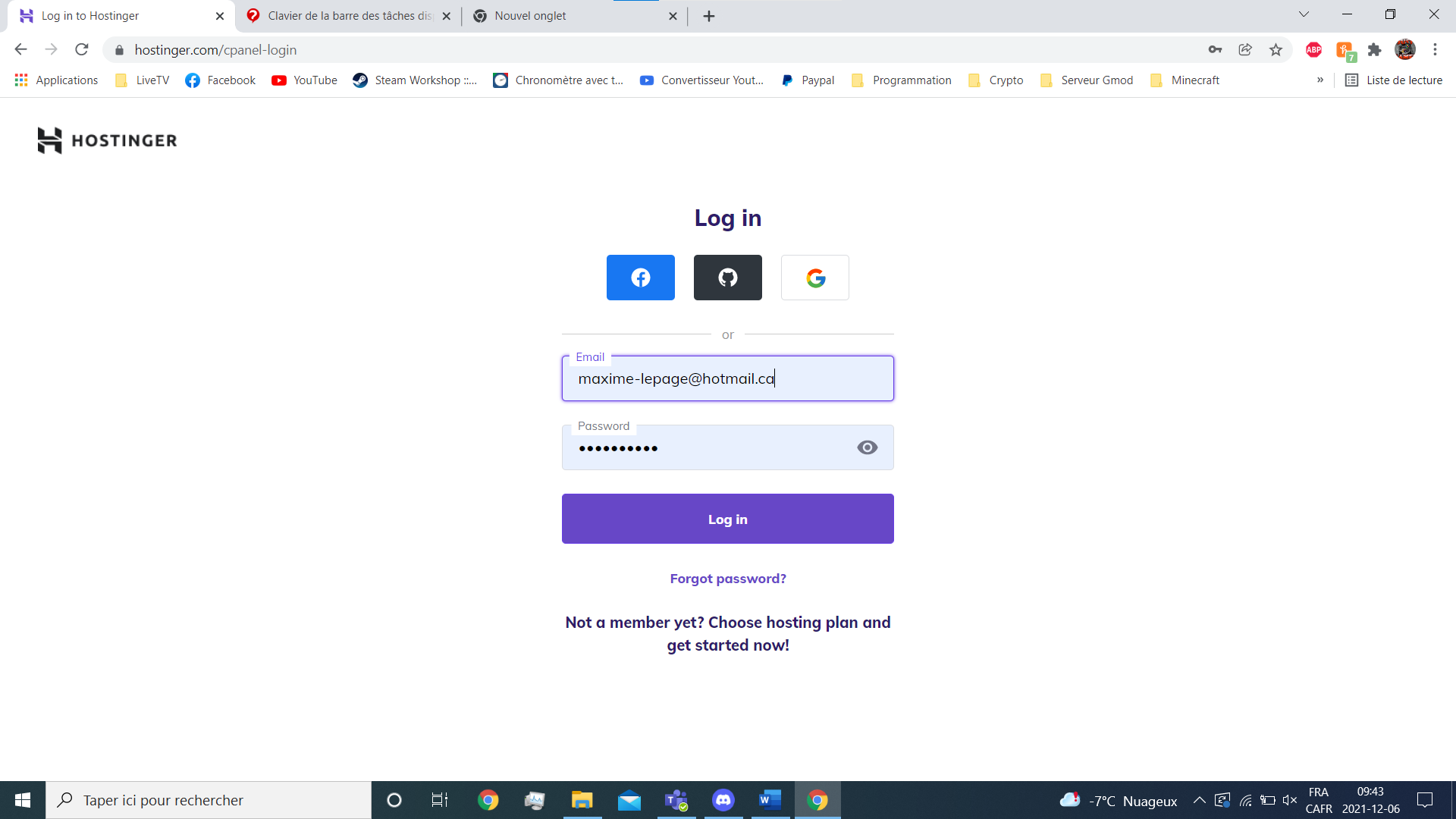
**NE PAS PUSH DANS LE MASTER**

### 

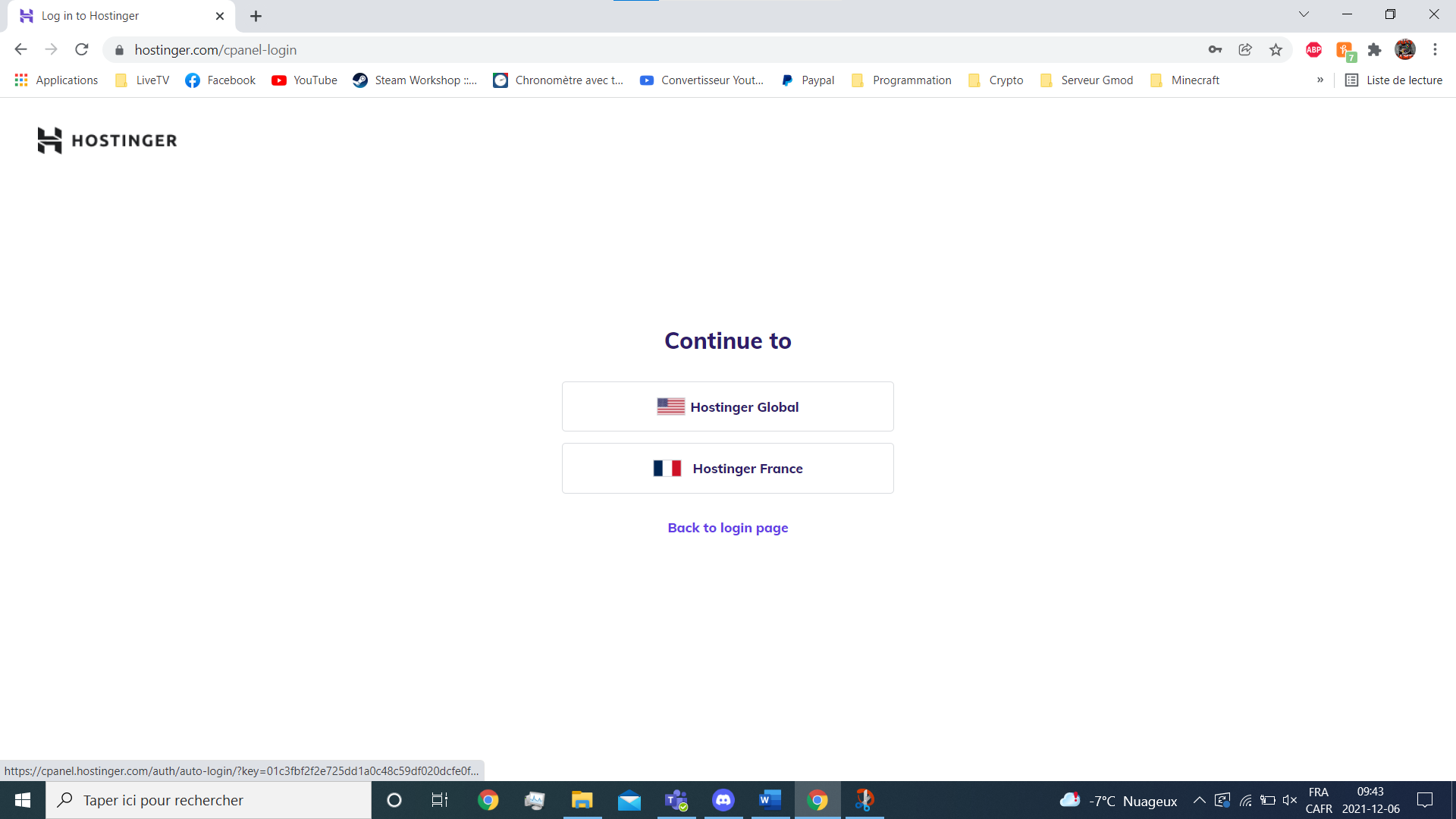
## Hostinger

Pour notre « hosting », nous avons choisi Hostinger, pas pour sa performance ou son accessibilité, mais bien parce que Maxime avait déjà un « hosting » prêt et qu’il voulait bien prendre la responsabilité de le gérer. On a utilisé notre « hosting » pour pouvoir héberger tous les API et applications web concernant le projet pour les relier à un nom de domaine et les rendre accessibles facilement à tout le monde.

Pour utiliser Hostinger, vous devez commencer par vous authentifier.



Une fois que s’est fait, choisissez Hostinger Global, il y a un meilleur support et plus d’option.



Maintenant que vous êtes connecté, vous devez aller dans la section Hosting en haut et vous serez amené à l’endroit montré ci-dessous. Dans la section Hosting, vous pouvez choisir de faire la gestion de tous vos sites web. Ce que nous voulons c’est gérer apipcst.xyz, donc on a cliqué sur Manage pour se rendre au Dashboard de notre projet.

Une image contenant texte, capture d’écran, moniteur, ordinateur

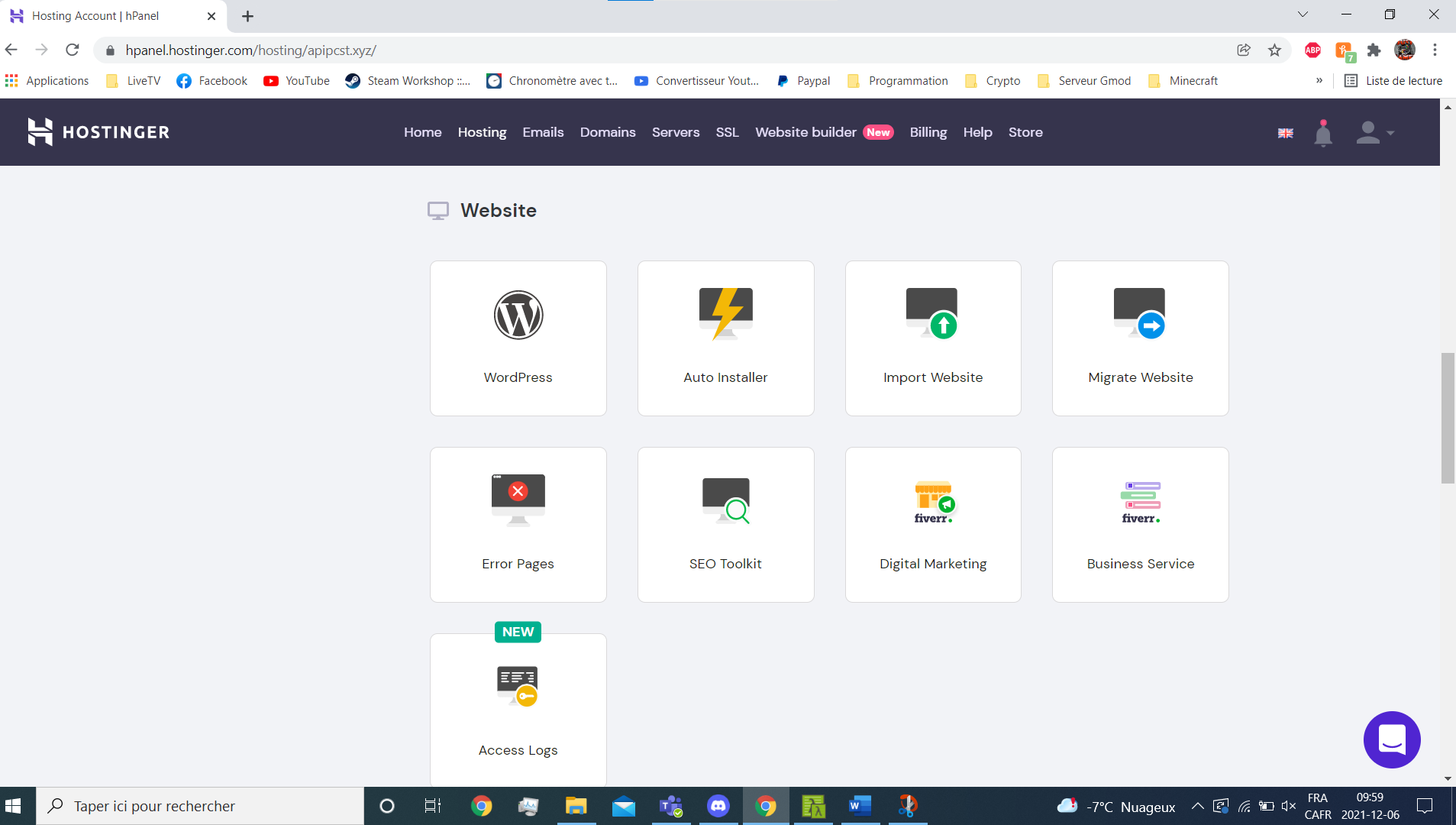
Description générée automatiquement

Voici le Dashboard de notre site web. Ici, on peut y voir toutes les options de notre projet.

Une image contenant texte, capture d’écran, ordinateur, portable

Description générée automatiquement

Il y a par exemple le « Auto installer » qui nous a permis d’installer Laravel sur tous nos sites web où Laravel était nécessaire.



Dans ces sections on peut créer des comptes FTP pour accéder aux fichiers du serveur ou simplement aller dans le file manager pour pouvoir les voir et les gérer.

Une image contenant texte, capture d’écran, ordinateur

Description générée automatiquement

Pour terminer, l’utilisation que nous avons faite de Hostinger, nous avons créé des sous-domaines dans la section sub-domain.

## Human Machine Interface

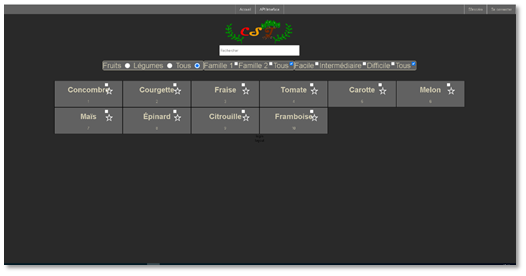
Introduction

Le Human Machine Interface (HMI) du projet Serre-Tech, plus précisément dans la branche d’aide à la décision, constitue de la partie « frontend ». Cela englobe toute l’affichage et tout ce qui est de l’accessibilité du site. Pour notre mandat, l’affichage devait inclure toutes les caractéristiques des plantes, les pages de connexion/d’inscription et les favoris de l’utilisateur, le tout en ayant un prototype de bilinguisme. Dans cette section, vous serez informé du déroulement de l’affichage, vous saurez comment utiliser celle-ci sur une base régulière et les recommandations.

L’affichage au fil du temps

Pour commencer, l’interface utilisateur créée avait été pensée pour être de base, c’est-à-dire que toutes ses fonctions puissent être fonctionnelles. Durant les deux premiers sprints, l’affichage n’était pas digne d’une œuvre de Da Vinci, il manquait de professionnalisme.

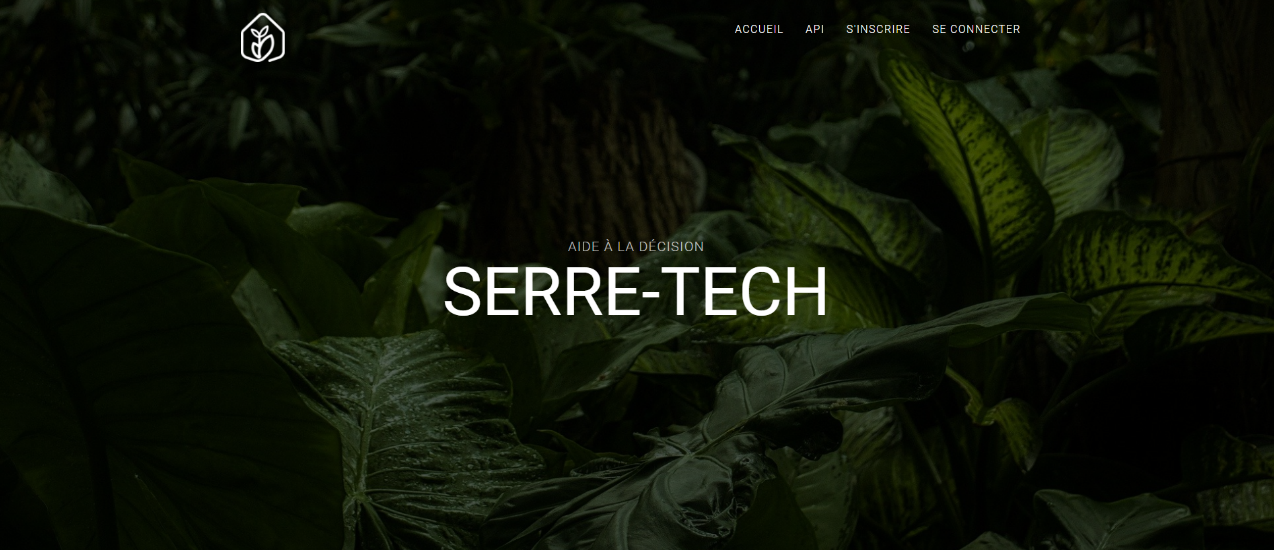
Voici une image de l’interface de la version finale de l’affichage lors du sprint 1 et du 2 :



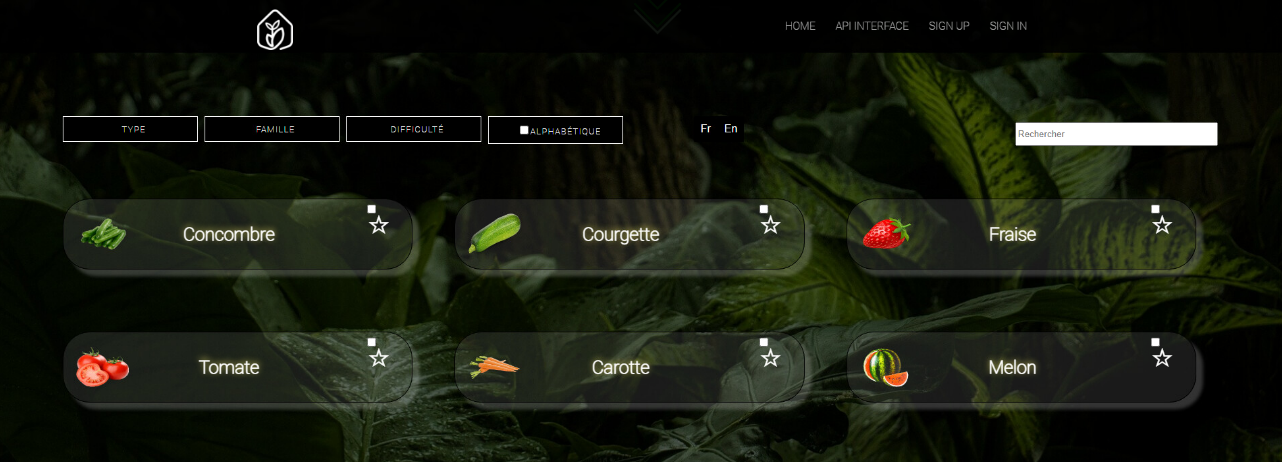
Effectivement, l’affichage n’était pas propre en tant que tel, mais bel et bien fonctionnel. Les filtres, les favoris et les pages de connexion et d’inscription étaient d’une qualité de base, mais comme dit précédemment, tout était fonctionnel.

Plus tard lors du sprint 2, un renouveau était bien évidemment nécessaire. Un membre d’une autre équipe nous a aidé afin de refaire l’interface au grand complet. Tout le code a été retravaillé durant près de 7 semaines afin de retrouver un certain cachet au projet.

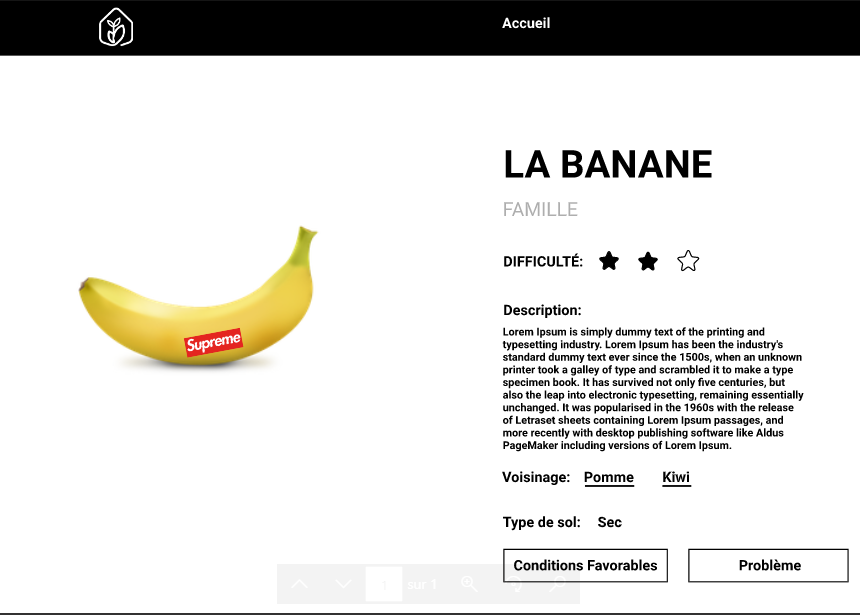
Voici la page d’accueil depuis le renouvellement :



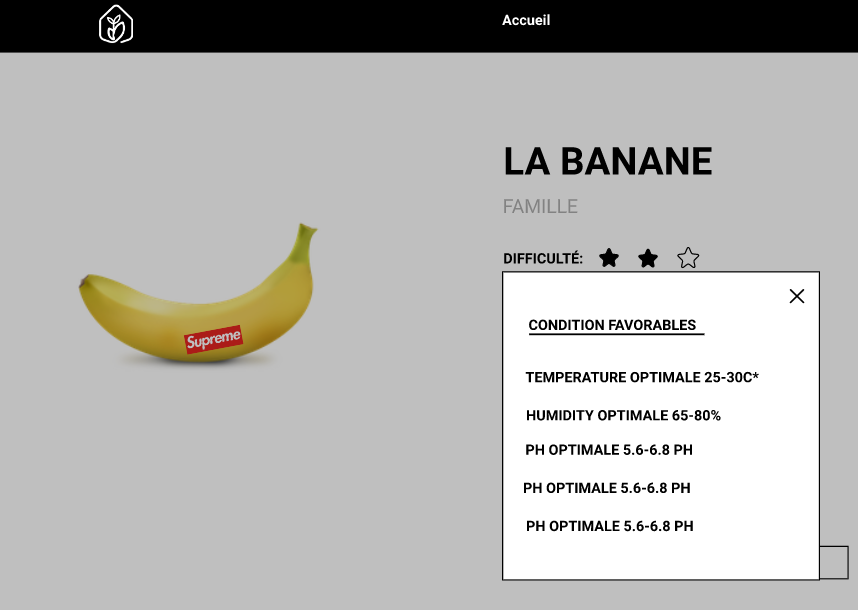
Voici la page d’affichage des plantes :



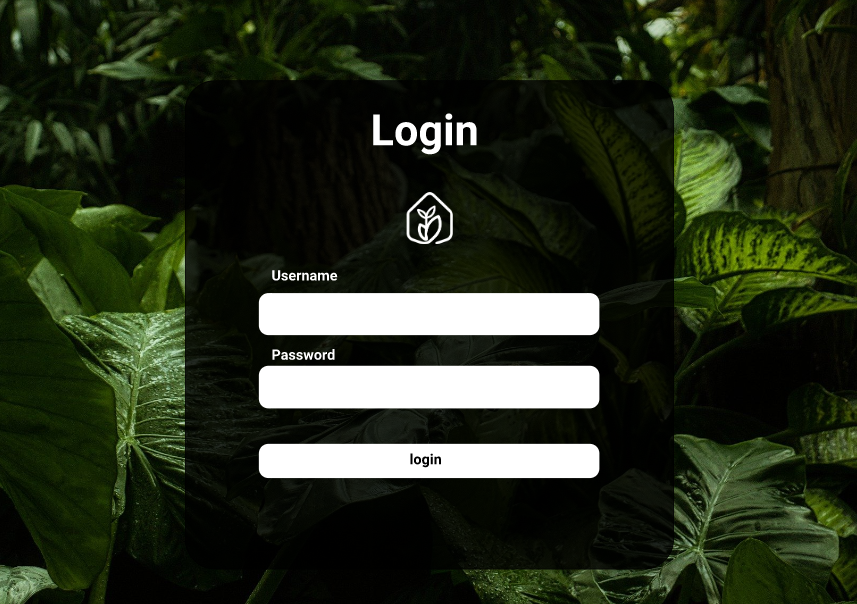
 Voici la page des détails lorsque l’on clique sur une plante :



Ensuite, voici l’affichage du component regroupant les conditions favorables lorsque le bouton est cliqué :



  Voici la page de connexion :



Comment ça marche ?

L’interface est très « user friendly », donc elle est très simple à utiliser pour n’importe quel type d’utilisateur. Certains aspects du site peuvent être ambiguë tels que les filtres, la recherche de plants et les problèmes et les conditions favorables de chaque plante. Cette section vous décrira comment vous familiariser avec l’interface.

Premièrement, les filtres sont simplement une liste déroulante affichant chacun d’eux les différents paramètres. Il y a des filtres pour les types de plante (fruit ou légume), les familles de plantes et les difficultés. Tous d’eux sont des listes déroulantes interconnectées afin que les filtres soient dynamiques. Ensuite, un autre filtre ayant pour but de classer en ordre alphabétique a été conçu pour faciliter la recherche. Pour voir les filtres, regarder les captures d’écrans ci-haut.

Deuxièmement, la recherche de plantes avec la barre de navigation est très pratique. En effet, une fonctionnalité permettant d’avoir toutes les plantes selon le début de votre recherche est intégrée. Celle-ci aide à savoir d’avance si votre recherche est valide ou correspond à ce qui se trouve dans la base de données. En gros, c’est une complétion automatique pour guider l’utilisateur dans ses recherches.

Troisièmement, pour trouver les conditions favorables et les problèmes de chaque plante, vous devez cliquer sur la plante en question. Cela affichera les détails de celle-ci et deux boutons y seront afin d’ouvrir les pages en mode « popup ». Pour avoir une idée d’affichage de cela, voir les captures d’écrans ci-hautes.

Recommandations

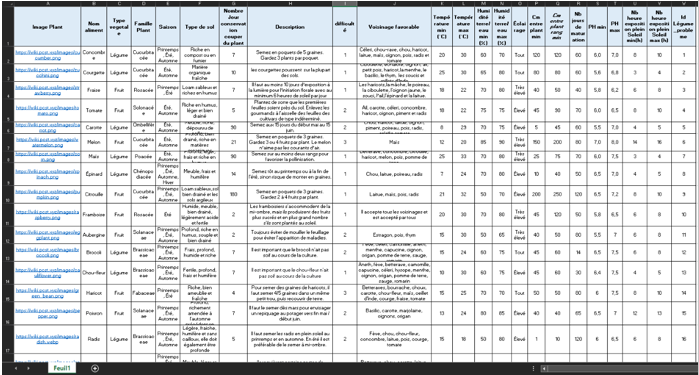
Durant tout le long du projet, plusieurs points bonus sont venus à l’idée des personnes ayant travaillées sur l’interfaçage. Ces points bonus n’étaient pas spécialement utiles au déroulement de base du HMI. Voici une liste de points à améliorer ou à ajouter pour le futur :

* Avoir un responsive plus travaillé;
* Faire une validation de courriel pour la création de compte;
* Avoir des pop-ups;
* Connexion automatique après la création d’un compte
* Changement de langue sans et téléchargement des données sans rechargement de la page;
* Correction de petits bugs d’affichage;
* Problème de liste déroulante des filtres;

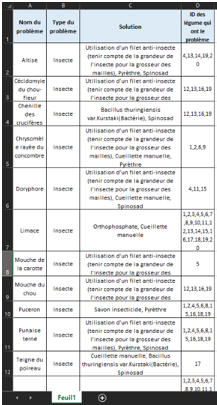
## Outil Importation

Cet outil est utilisé afin d’ajouter facilement des données à la base de données de notre API tout en utilisant l’API lui-même.  Le fonctionnement est assez simple : l’outil d’importation lis un fichier .xlsx et s’occupe de faire des requêtes d’API afin d’ajouter les différentes plantes, problèmes et conditions favorables.  À noter que les problèmes sont ajoutés dans un fichier .xlsx à part, afin d’éviter la répétition dans le fichier des plantes.  Bien évidemment, il est important que ce fichier .xlsx respecte les normes établis par rapport aux colonnes et par rapport au type de données qui lui sont attribuées.  Par exemple, il est impossible d’ajouter une chaine de caractères pour la difficulté d’une plante.

Voici le format utilisé pour les plantes : (veggies\_3.xlsx)

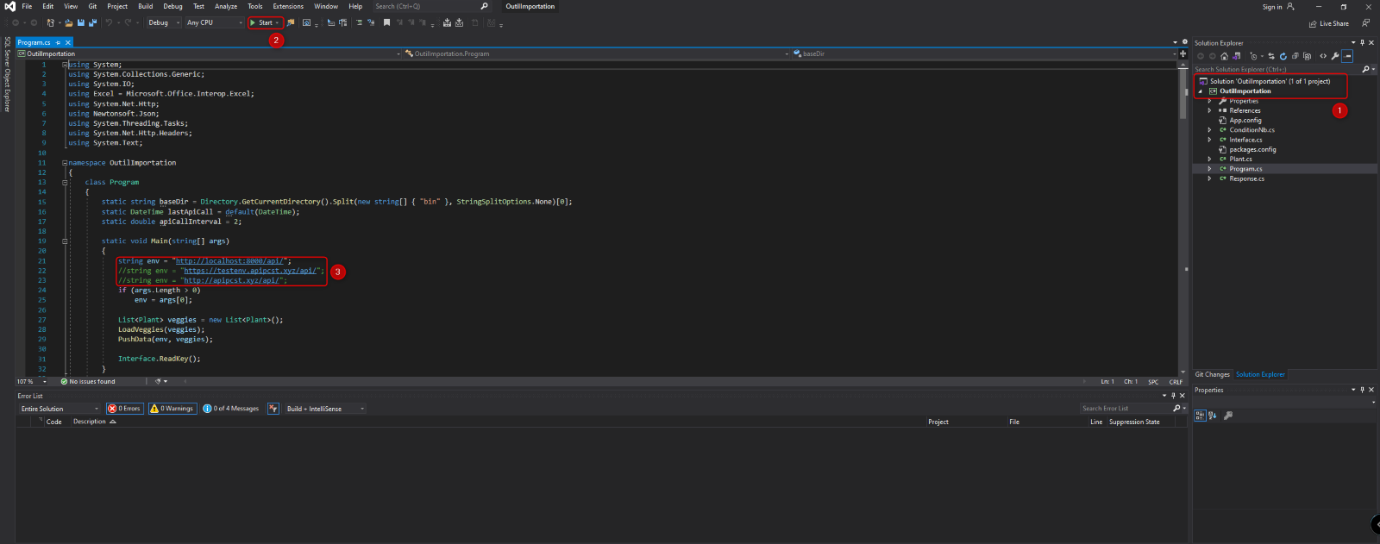


Voici le format du fichier à utiliser pour l’assignation de problèmes (problems\_1.xlsx)

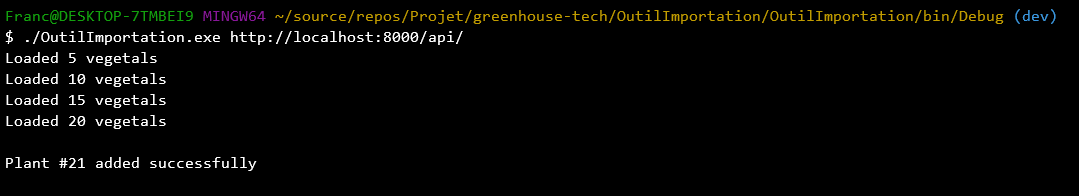
Saut de page

Il est également important de savoir que cet outil est une application .NET Core développée en C#.  Pour l’utiliser, deux choix s’offrent à vous :

Premièrement, vous pouvez ouvrir la solution dans Visual Studio, choisir l’environnement que vous voulez (le lien vers l’API) et lancer l’application.



Deuxièmement, vous pouvez vous rendre dans le dossier de l’application et lancer la commande suivante en spécifiant l’environnement que vous voulez (le lien vers l’API).  À noter que si vous voulez utiliser cette solution, vous devez vous assurer que les fichiers de données se trouve dans le dossier « Data » placé à coté de l’exécutable.



# **Glossaire**

|  |  |
| --- | --- |
| **Terme Technique** | **Description** |
| Frontend | Partie visible et interactive d’une application. |
| Backend | Partie cachée de l’utilisateur d’une application. |
| Backlog | Liste des tâches à effectuer. |
| Git Master | Personne responsable de la gestion du repos GitHub, vérifiant la qualité du code avant de merge. |
| Merge | Fusion du nouveau code au code existant |
| Scrum Meeting | Rencontre régulière où l’on explique les tâches effectuées la veille, les tâches auquel des difficultés ont été éprouvées et les tâches à effectuer le jour même. |
| Framework | Cadre de développement contenant les outils pour les développeurs. |

# **Médiagraphie**

* Technologies Frontend : <https://cubettech.com/resources/blog/laravel-with-react-js-vs-angular-js-vs-vue-js/>
* Apply Yourself App : <https://www.agriexpo.online/fr/prod/agri-inject/product-170720-123351.html>
* Farmhand : <https://www.agriexpo.online/fr/prod/freight-farms/product-181170-48208.html>
* OneNote Projet : [https://cegepjonquiere-my.sharepoint.com/personal/1932614\_etu\_cegepjonquiere\_ca/\_layouts/15/Doc.aspx?sourcedoc={604c4bd3-25c2-4969-b6cf-68d479a6e02f}&action=edit&wd=target%28Organisation.one%7C8804E685-E47F-4AE3-ADBB-D10063F5442E%2FProc%C3%A9dure%20Git%7C110dbd65-36cc-49eb-9ed9-477555e9cfb3%2F%29](https://cegepjonquiere-my.sharepoint.com/personal/1932614_etu_cegepjonquiere_ca/_layouts/15/Doc.aspx?sourcedoc=%7b604c4bd3-25c2-4969-b6cf-68d479a6e02f%7d&action=edit&wd=target%28Organisation.one%7C8804E685-E47F-4AE3-ADBB-D10063F5442E%2FProc%C3%A9dure%20Git%7C110dbd65-36cc-49eb-9ed9-477555e9cfb3%2F%29)
* GitHub Projet : <https://github.com/Dolotboy/GreenHouse-Tech>
* Documentation hors ligne : <https://cegepjonquiere.sharepoint.com/:w:/s/Projetlgumebig/Ec7b2ilQtbtKhcLOJMoCsvMBybV_YvtySjhxAMkY3yT4wQ?e=bHBcl7>
* Documentation normes de programmation :<https://teams.microsoft.com/l/file/84213801-7755-403E-8D94-6A6CFEA5377C?tenantId=880510e3-b42a-4483-a907-75cd19e3ceba&fileType=docx&objectUrl=https%3A%2F%2Fcegepjonquiere.sharepoint.com%2Fsites%2FProjetlgumebig%2FDocuments%20partages%2FGeneral%2FNorme_Prog.docx&baseUrl=https%3A%2F%2Fcegepjonquiere.sharepoint.com%2Fsites%2FProjetlgumebig&serviceName=teams&threadId=19:-QH0JS0VaLtHhWIIAZalutJFMoU9CCn2CRbcOC30Py01@thread.tacv2&groupId=d41a113c-012c-414f-aec0-a6b5bf8921f2>
* Documentation pour API : [https://cegepjonquiere-my.sharepoint.com/personal/1932614\_etu\_cegepjonquiere\_ca/\_layouts/15/Doc.aspx?sourcedoc={604c4bd3-25c2-4969-b6cf-68d479a6e02f}&action=edit&wd=target%28Alec.one%7Ceac9a0e1-20f2-444f-887f-a360faf90448%2FDocument%20Analyse%7Cfcc30a22-d85e-4411-bd09-356c1641db06%2F%29](https://cegepjonquiere-my.sharepoint.com/personal/1932614_etu_cegepjonquiere_ca/_layouts/15/Doc.aspx?sourcedoc=%7b604c4bd3-25c2-4969-b6cf-68d479a6e02f%7d&action=edit&wd=target%28Alec.one%7Ceac9a0e1-20f2-444f-887f-a360faf90448%2FDocument%20Analyse%7Cfcc30a22-d85e-4411-bd09-356c1641db06%2F%29)

# **Tableau des versions**

|  |  |
| --- | --- |
| **Date** | 2021-09-02 |
| **Numéro de Version** | 1.0 |
| **Responsable de Livraison** | Alec Lespérance |
| **Date** | 2021-12-14 |
| **Numéro de Version** | 2.0 |
| **Responsable de Livraison** | Matis Larouche |