Rapport du Projet Pluridisciplinaire
d'Informatique Intégrative

Pierre AUGUSTE, Aurélie DEMURE, Nicolas FERNANDEZ, Manon LECUBIN

2022-2023

Responsables du module : Olivier FESTOR, Anne-Claire HEURTEL et Gérald OSTER

Les jardins partagés

Table des matières

1 Introduction

Ce rapport vise à retracer l'ensemble de notre travail dans le cadre du Projet Pluridisciplinaire d'Informatique Intégrative. Le thème de ce projet est "Les jardins partagés". Son objectif est d'appliquer nos connaissances en Gestion de projet, Base de données, Web et Algorithmie pour réaliser une application répondant à un besoin concernant cette thématique en optimisant nos ressources.

Nous avons commencé par définir ce qu'était un jardin partagé avant de faire un état de l'art pour mieux cerner l'existant et nous orienter dans notre projet.

Ici, nous allons garder la version synthétique de notre état de l'art.

Nom	Idée en plus	Historique	Critique
SEEED	Réseau social accessible via Facebook	Budget initial : 5000€ Début 2021 : lancement	Peu de recul sur la fiabilité de la plateforme
LePotiron	Formulaire de mise en contact Blog	2010 : lancement 2017 : renouveau	Manque d'anticipation
PlantCatching	Ajout des plantes et des graines	2022 : fermeture	2022 : fermeture
Fruiteefy	Ajout confitures et sauces	Juin 2019 : création	Valorisation des circuits courts
Leaf	Entre particuliers sur tout le terroir	Juin 2022 : création	Consommation locale non promue
Aux Arbres Citoyens	Cueillettes collectives et solidaires	Fondée en 2020	Aucune application seulement une page Web

2 Conception et implémentation

2.1 Base de données

Cinq tables ont été réalisées dans notre base de données afin de mettre en place l'application.

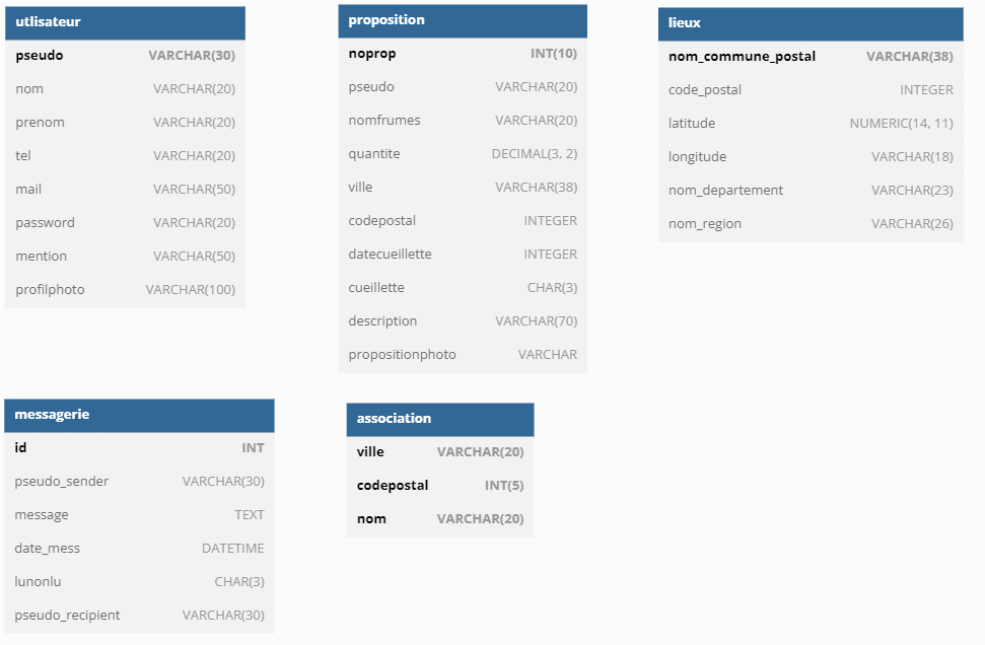

Figure 1: Schéma de la base de données.

La table *utilisateur* comporte les informations relatives aux utilisateurs :

- Pseudo : il s'agit de la clé primaire et est donc unique. Il permet aux utilisateurs de s'identifier sur le site.
- Nom : contient le nom de l'utilisateur.
- Prénom : contient le prénom de l'utilisateur.
- Tel : permet l'enregistrement du numéro de téléphone de l'utilisateur.
- Mail : permet de même l'enregistrement de l'adresse électronique de l'utilisateur sous la bonne forme
- Password : enregistre le mot de passe de l'utilisateur sous forme cryptée avec l'algorithme associé.
- Mention : contient les informations que l'utilisateur souhaite partager sur son profil.

- Phofilphoto : contient l'adresse de la photo de profil de l'utilisateur.

La table *proposition* enregistre les données des propositions créées par les utilisateurs :

- Noprop : Il s'agit du numéro de la proposition. Cet identifiant est unique et constitue la clé primaire de la table.
- Pseudo : contient le pseudo du propriétaire de la proposition.
- Nomfrumes : contient l'information sur le fruit ou légume proposé.
- Quantite : permet à l'utilisateur d'indiquer la quantité de fruits/légumes proposée.
- Ville : indique la ville dans laquelle se trouve la proposition.
- Code postal : contient le code postal associé à la ville.
- Cueillette : contient l'information sur la cueillette (à réaliser ou non)
- Datecueillette : la date à laquelle les fruits ont été récoltés si la cueillette est déjà faite.
- Dateexpiration : la date à laquelle la proposition sera automatiquement supprimée.
- Description : une courte description de la proposition.
- Propositionphoto : contient l'adresse de la photo de la proposition.

La table *lieux* comporte des informations relatives aux différents lieux que l'application peut supporter :

- Nom_commune_postal : Il s'agit du nom de la ville. Il est élément de la clé primaire.
- Code_postal : il s'agit du code postal associé à la ville. Il est également élément de la clé primaire.
- Latitude : contient la coordonnée de latitude correspondante à la ville.
- Longitude : contient la coordonnée de longitude correspondante à la ville.
- Nom_departement : permet d'associer la ville avec son département.
- Nom_region : permet d'associer la ville avec sa région.

La table *association* contient les informations concernant les associations que l'application propose :

- Ville : permet de connaitre la ville dans laquelle se situe l'association.

- Codepostal : contient le code postal de la ville.
- Nom : correspond à la dénomination de l'association.

Enfin, la table *messagerie* qui permet la mise en place du système de communication entre les utilisateurs :

- Id : il s'agit de l'identifiant du message. Il est unique et s'auto-incrémente, constituant ainsi la clé primaire de la table.
- Pseudo_sender : contient le pseudonyme de l'utilisateur qui a envoyé le message.
- Pseudo_recipient : contient le pseudonyme de l'utilisateur qui a reçu le message.
- Message : variable de texte qui enregistre le message de l'utilisateur.
- Date_mess : La date et l'heure à laquelle le message a été envoyé.
- Lunonlu : contient l'information sur le visionnage du message

2.2 Serveurs WEB

2.2.1 Gestion de la session

L'utilisateur a la possibilité de se connecter sur notre application web et donc d'avoir une session. Celle-ci est gérée grâce à la fonction **Session** du module **flask_session**. On initialise la session comme suit :

```
app=Flask(__name__, static_folder="./static", template_folder="./templates")
app.config["SESSION_PERMANENT"]=False
app.config["SESSION_TYPE"]='filesystem'
Session(app)
```

Figure 2: Initialisation de la session.

Ensuite, on active la session avec **session["name"]=mail**. Le mail de l'utilisateur est utilisé ici pour le nom de la session, car il définit uniquement l'utilisateur dans la base de données, et nous pouvons donc facilement récupérer ces informations pour les afficher sur l'application. La session peut se désactiver avec **session["name"]=None**.

L'état de la session (connectée ou non) va influencer l'affichage et l'accès aux différentes pages de l'application. On peut avoir accès à cet état avec **session.get("name")**.

2.2.2 La barre de navigation

Il existe deux barres de navigation différentes : une quand l'utilisateur n'est pas connecté, et une autre lorsqu'il est connecté.

Les deux barres se devait d'être facilement utilisables, ainsi, elles sont contenues dans une balise **nav** à laquelle nous avons appliqué le CSS ci-dessous. De plus, chaque icône des barres renvoie vers une page de l'application, grâce à des balises **a**.

```
nav {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    align-items: center;
    justify-content: space-around;
    background-color: var(--light-pink);
    height: 5%;
    box-shadow: 0 1px 8px 0 rgba(0,0,0,.2),0 3px 4px 0 rgba(0,0,0,.14),0 3px 3px -2px rgba(0,0,0,.12);
}
```

Figure 3: Le CSS des barres de navigation.

Sur les deux barres, nous retrouvons le magnifique logo de l'application Opti'Fruit, qui renvoie vers la page d'accueil /. Il y a aussi un logo de groupe et un de loupe, renvoyant respectivement vers la page des associations **/associations**, et vers la page de recherche **/recherche**.

En plus de cela, la barre déconnectée possède un bouton "Inscription" et un bouton "Connexion", renvoyant respectivement vers la page d'inscription

/inscription et celle de connexion **/connexion**. Cela nous donne la première barre de navigation :

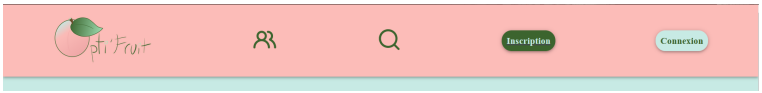

Figure 4: Barre de navigation déconnectée.

Pour la barre connectée, nous retrouvons une icône pour la messagerie **/messagerie/None**, ainsi que la photo de profil et le pseudo de l'utilisateur, revoyant vers la page de profil **/profil/mail_utilisateur**. À noter qu'à chaque apparition d'une image de profil sur l'application, on teste son existence pour l'utilisateur donné dans la base de données, avec **{%if profil[0]["profilphoto"]%}**. Si la photo n'existe pas, l'application renvoie une image de profil par défaut. Cela donne donc la deuxième barre de navigation :

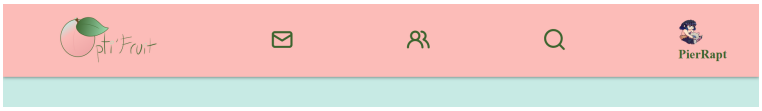

Figure 5: Barre de navigation connectée.

2.2.3 Utilisation des layouts

Les barres de navigation étant communes à toutes les pages de l'application, il semble pertinent d'utiliser des layouts plutôt que de recopier les barres de navigation sur chaque page. De plus, cela offre un meilleur contrôle sur quelle barre choisir.

Ainsi, dans les fichiers html des barres de navigation, se trouvent à la fin :

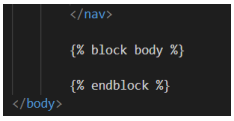

Figure 6: Lignes de code indiquant que du contenu peut y être ajouté.

Les fichiers contenants le contenu additionnel sont sous cette forme :

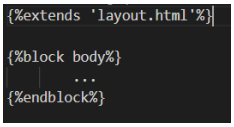

Figure 7: Syntaxe indiquant que le contenu s'imbrique dans le fichier "layout.html".

Ici, nous voulons que la barre de navigation s'adapte à l'état de la session. Pour cela, lors de la requête d'une page, on teste si la session est active ou non, comme vu précédemment. Si la session est active, c'est la barre de navigation connectée que l'on veut, sinon c'est l'autre. Le nom de la barre ainsi voulue est stocké dans une variable **navbar**, puis envoyé avec la requête du fichier html voulu. Ce dernier aura donc une en-tête de la forme :

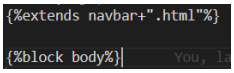

Figure 8: En-tête des fichiers html de l'application (sauf des deux layouts).

2.2.4 Page d'accueil

Sur la page d'accueil se trouvent les choses les plus importantes de l'application :

Une liste synthétique des propositions existantes, avec l'image de profil et le pseudo de l'utilisateur qui a posté la proposition, la description et l'image associées. On peut cliquer sur chacune des propositions, renvoyant sur la page de la proposition en question **/proposition/numéro_proposition**. Si aucune proposition n'est disponible, l'application informe l'utilisateur qu'il n'y en a pas.

À côté de cette liste, il y a une image de carte renvoyant sur la page de rechercher **/recherche**, ainsi qu'un bouton "Je propose", qui renvoie vers la page d'ajout de proposition **/propose**.

Figure 9: La page d'accueil d'Opti'Fruit.

2.2.5 Page d'ajout de proposition

Si l'utilisateur n'est pas connecté, il est redirigé vers la page de connexion **/connexion**.

La page est composée d'un form avec plusieurs champs à remplir : Les fruits ou légumes proposés, la quantité en kg, le code postal, la ville, la description sous forme de **textarea**, une checkbox à cocher si les produits sont à cueillir, si cette dernière n'est pas cochée, une date de cueillette, une date d'expiration de la demande et un bouton pour importer une image. Enfin, un bouton "VALIDER", qui envoie les informations recueillies avec une requête POST vers **/propose**.

Lorsque la page reçoit une requête POST, elle vérifie que toutes les informations importantes sont présentes et que le code postal et la ville sont en accord avec la base de données "lieux". S'il y a un problème, l'application renvoie le form avec le message d'erreur associé au problème. Sinon, il enregistre l'image sur le serveur, et les infos dans la base de données, puis redirige vers la page de la proposition ainsi créée **/proposition/nouvelle_proposition**.

Figure 10: La page d'ajout de proposition d'Opti'Fruit.

2.2.6 Page d'info d'une proposition

La page prend en paramètre un identifiant qui renvoie sur la page d'info de la proposition associée. Il y a donc une page par proposition. Sur celle-ci apparaissent toutes les info de la proposition, ainsi qu'un bouton "Home" en bas renvoyant sur la page d'accueil /.

De plus, si l'utilisateur est connecté, et que c'est lui qui a posté la proposition, un bouton "Supprimer la proposition" renvoie sur la page **/supprPropose/id**. Cette dernière vérifie que l'utilisateur est bien connecté (dans le cas contraire elle le renvoie sur la page d'accueil /), supprime la proposition avec l'identifiant **id** dans la base de données, et renvoie sur une page validant la suppression, avec un bouton "OK" validant la suppression.

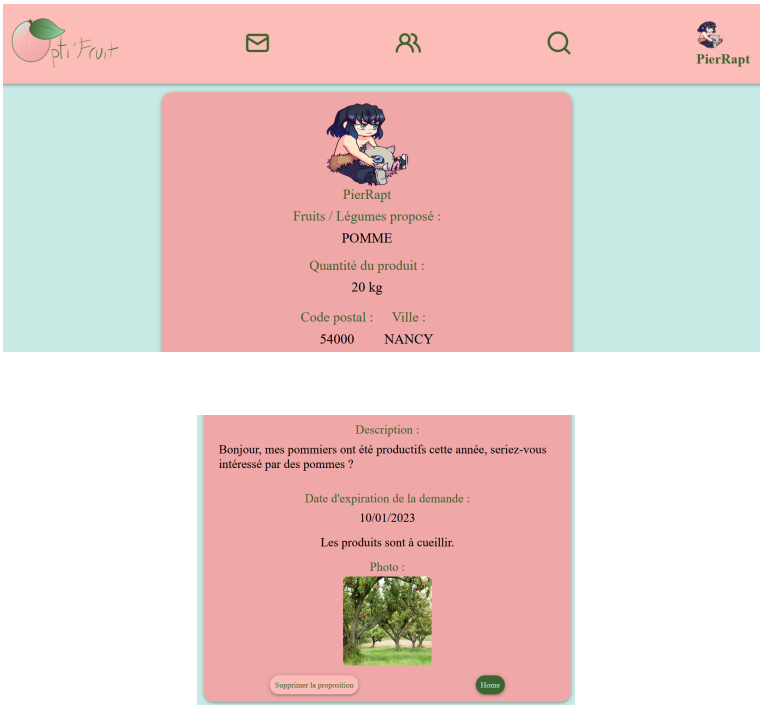

Figure 11: La page d'information d'une proposition d'Opti'Fruit.

2.2.7 Page de Messagerie

Une fonctionnalité importante de l'application est de pouvoir échanger avec les autres utilisateurs. Si l'utilisateur n'est pas connecté, il est redirigé vers la page d'accueil /. Sinon, l'application récupère dans la base de données l'ensemble des utilisateurs à qui l'utilisateur connecté a envoyé un message, ou de qui il a reçu un message, triés dans l'ordre décroissant de la date du dernier message échangé. Si cette liste est vide, l'application informe l'utilisateur qu'il n'a aucune discussion.

De plus, la page prend en paramètre un pseudo d'un utilisateur existant. Si celui-ci est **None**, alors l'application ouvre la dernère discussion en cours. Si celui-ci est dans la liste des utilisateurs avec qui l'utilisateur connecté a déja échangé, la liste des messages échangés est affichée, par ordre d'envoi, accompagnée de la photo de profil et du pseudo de celui qui l'a envoyée. Le message est affiché à droite si l'auteur est l'utilisateur connecté, à gauche sinon. Tout en bas de cette liste, un champ est disponible pour écrire un message, avec un bouton "Envoyer" envoyant le message entré avec une requête POST vers **/messagerie/pseudo_destinataire**. Si le pseudo n'est pas dans la liste des utilisateurs avec qui l'utilisateur connecté a déjà échangé, la liste des messages est remplacée par un texte informant l'utilisateur connecté qu'il n'y a aucun message avec cet utilisateur, avec toujours le champ pour envoyer un message.

Si la page reçoit une requête POST, il insert dans la base de données le message envoyé à l'utilisateur avec qui la discussion est ouverte, et recharge la page de messagerie, affichant le nouveau message dans le flux.

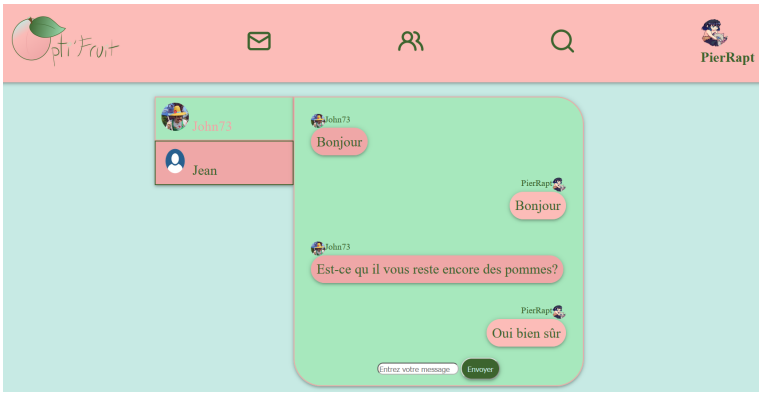

Figure 12: La page de messagerie d'Opti'Fruit.

2.2.8 Page associations

La page d'affichage des associations permet d'afficher toutes les associations auxquelles il est possible de donner ses fruits et légumes par ordre alphabétique des villes. Cette page permet d'afficher le nom des associations, leur code postal

et leur ville après avoir récupéré ces informations dans la table **association** en utilisant une requête SQL sélectionnant la **ville**, le **codepostal** et le**nom**, avec la commande **ORDER BY** sur la **ville**. La page contient également un formulaire qui permet de rechercher les associations dans une certaine ville en utilisant dans la requête SQL la commande **LIKE** sur le nom de la ville rentrée par l'utilisateur. L'affichage se fait ensuite grâce à une grille.

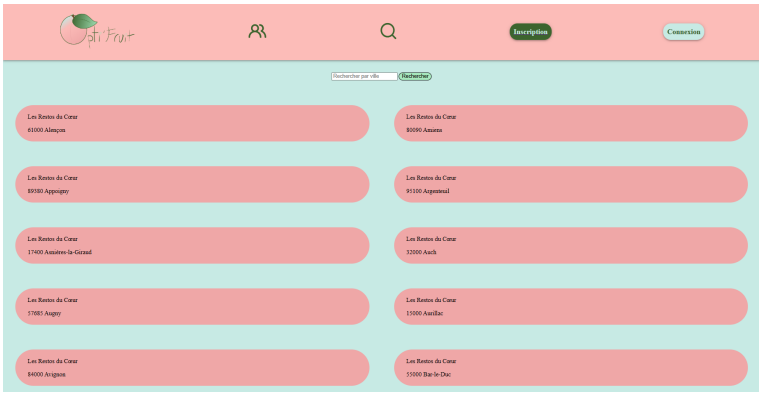

Figure 13: La page des associations

2.2.9 Page inscription

Le principe du site étant d'avoir des utilisateurs, les nouveaux arrivants pourront s'inscrire en remplissant une série de champs. Leurs **nom**, **prénom**, **pseudo**, **numéro de téléphone**, **adressse mail** leur seront demandés ainsi qu'un **mot de passe** qu'ils devront confirmer. Ce mot de passe devra également contenir entre 8 et 20 caractères pour des raisons de complexité et sa validité sera vérifiée avant son cryptage. Le cryptage permet de ne pas enregistrer le mot de passse brut dans la base de données. Une **mention** et une **photo de profil** optionnelles seront également demandées. Sinon, aucune mention n'apparaîtra dans le profil et une photo par défaut sera ajoutée. Une fois tous les champs remplis, un clic sur le bouton **Valider** activera la méthode **POST** de la page. Les données seront récupérées grâce à des request.

```
mention = request.form.get("mention")
profilphoto=request.files['profilphoto']
```

Figure 14: Différents champs de récupération des données

Si des informations obligatoires ne sont pas remplies, des **render_templates** permettent d'afficher à l'utilisateur le message d'erreur adapté. Il faut ensuite tester les contraintes de la base de données notamment l'unicité des pseudos qui nous servent de clé primaire pour la table utilisateur. Seuls certains

types de fichiers sont acceptés pour les photos grâce à un champ dans le html d'inscription.

```
<input class="input" name="profilphoto" type="file" value={{profilphoto}} accept="image/x-png,image/jpg,image/jpeg">
```

Figure 15: Rechercher les extensions jpg, jpeg et png

Il est alors possible d'insérer les données de l'utilisateur dans la base de données et de le rediriger directement vers sa page de profil.

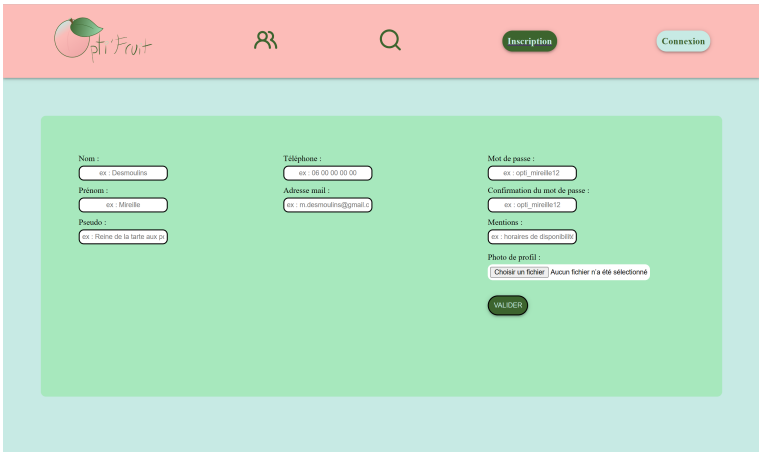

Figure 16: Page inscription à remplir

Les champs étant étalés sur trois colonnes pour un écran d'ordinateur, il a fallu adapter le css pour des écrans de plus petites tailles tels qu'un téléphone.

```
@media screen and (max-width: 450px) {
    .page {
        column-count: 1;
        margin-left: 5%;
        margin-top: 5%;
        margin-right: 5%;
        margin-bottom: 5%;
        background-color: var(--light-green);
        padding-left: 5%;
        padding-right: 5%;
        padding-top: 5%;
        padding-bottom: 5%;
        border-radius: 10px;
    }
```

Figure 17: CSS pour un téléphone portable : affichage sur une seule colonne

2.2.10 Page connexion

Quand les utilisateurs se sont déjà inscrits une première fois, il est nécessaire qu'ils puissent accéder de nouveau à leur profil. En se rendant dans Connexion sur le bandeau déconnecté, ils arriveront sur la page Connexion.

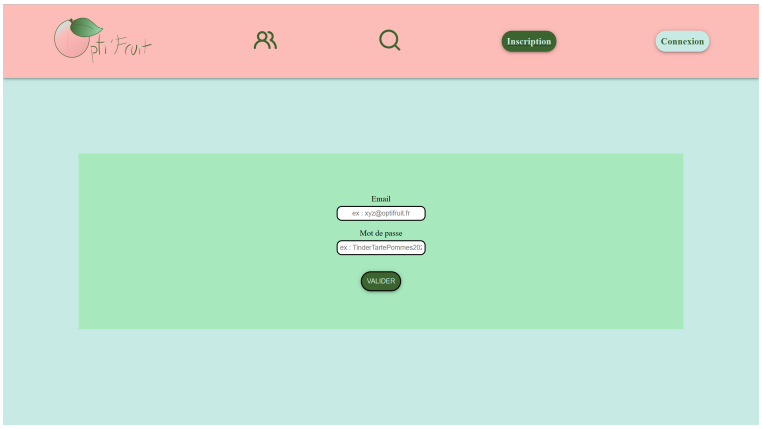

Figure 18: Page de connexion

Son **adresse mail** et son **mot de passe** sont récupérés à l'aide de request form. Si au moins un des deux champs n'est pas rempli, un render_template permet de le signaler à l'utilisateur en affichant un message d'erreur explicite. Il s'agit ensuite de vérifier la véracité des informations fournies. La première étape est de crypter le mot de passe avec le même algorithme que celui utilisé à l'inscription. Il faut ensuite récupérer les mots de passe sous le même format pour les comparer. Si les données sont correctes, l'utilisateur est redirigé vers son profil avec sa session activée. Sinon, l'utilisateur est informé d'un problème de cohérence entre les données.

```
if password2 == password and password!=[]:
    pseudo=db.execute("SELECT pseudo FROM utilisateur WHERE mail=?",mail)
    pseudo=pseudo[0]['pseudo']
    session["name"]=pseudo
    return redirect('/profil/'+pseudo)
else:
    return render_template("connexion.html", message="Adresse mail ou mot de passe incorrect")
```

Figure 19: Vérification que le mot de passe correspond à l'adresse renseignée

2.2.11 Page de profil

La page de profil regroupe les différentes informations d'un utilisateur auxquelles tout le monde peut y accéder. Les informations sont récupérées à partir du pseudo de l'utilisateur grâce à une requête SQL, les informations affichées sont alors son nom, son prénom, sa photo de profil, ses mentions et ses différentes propositions, qui seront présentées sous la même forme que sur la page d'accueil, en-dessous des informations sur la personne. Si l'utilisateur n'avait pas renseigné de photo de profil, une photo par défaut apparaît. Ensuite, différents boutons apparaissent selon l'utilisateur connecté : si l'utilisateur qui est connecté est le même que celui dont la page de profil s'affiche, il apparaît un bouton **Se déconnecter**, un bouton **Supprimer mon compte** et un bouton **Modifier mes informations**.

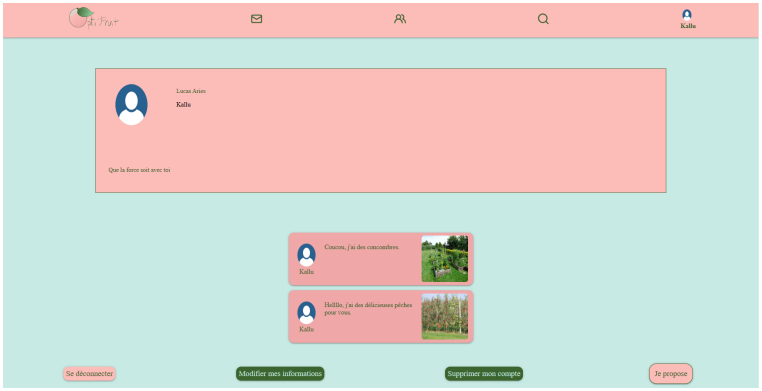

Figure 20: Page de profil lorsque l'utilisateur connecté est celui du profil affiché

Le bouton **Se déconnecter** redirige vers la page **/logout**, c'est-à-dire vers la page d'accueil tout en déconnectant l'utilisateur. Le bouton **Supprimer mon compte** supprime l'utlisateur, ses différentes informations, ses propositions et ses messages de la base de données grâce à une requête SQL. Le bouton **Modifier mes informations** redirige vers la page **/modifierprofil/pseudo**. Si l'utilisateur connecté n'est pas le même que celui dont la page de profil est affichée en revanche, un seul bouton apparaît : **Envoyer un message à pseudo_destinataire** qui permet d'envoyer un message à la personne dont le profil est affiché en redirigeant vers la page **/messagerie/pseudo_destinataire**.

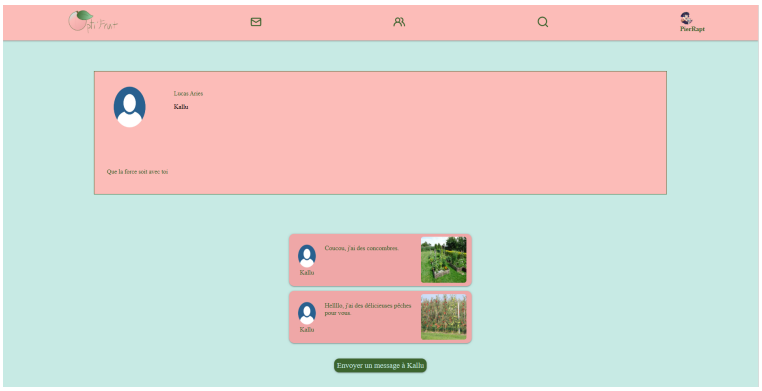

Figure 21: Page de profil lorsque l'utilisateur connecté est différent de celui du profil affiché

2.2.12 Page de modification du profil

Les utilisateurs ont la possibilité de modifier leurs informations grâce à cette page. Ils peuvent modifier les informations qu'ils souhaitent en remplissant uniquement les champs qu'ils souhaitent modifier : leurs **nom**, **prénom**, **pseudo**, **numéro de téléphone**, **adressse mail**, une **mention**, une **photo de profil** ou leur **mot de passe** qu'ils devront confirmer. Ce mot de passe devra également contenir entre 8 et 20 caractères pour des raisons de complexité et sa validité sera vérifiée avant son cryptage. Le cryptage permet de ne pas enregistrer le mot de passse brut dans la base de données. Si le mot de passe n'est pas conforme à ces attentes, un message d'erreur adapté sera renvoyé, comme pour la page d'inscription. Une fois que l'utilisateur a rempli les champs qu'il souhaitait modifier, un clic sur le bouton **Modifier** activera la méthode **POST** de la page.

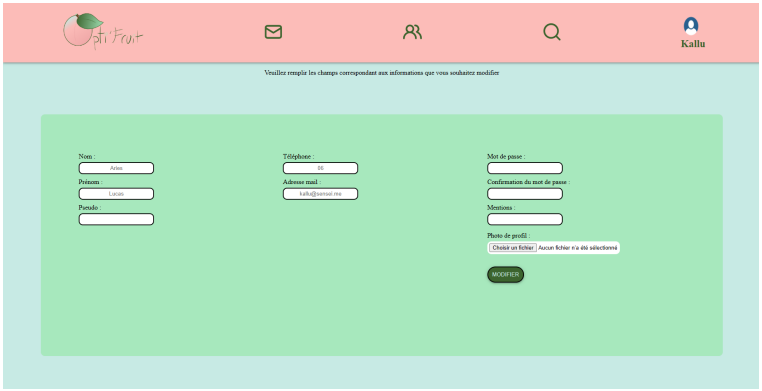

Figure 22: Page de modification du profil

Les données seront récupérées grâce à des request. Pour chaque information que l'utilisateur a changé, la commande **UPDATE** pour les requêtes SQL est utilisée. Enfin, si le pseudo fait partie des informations à changer, étant donné qu'il est la clé primaire de la table **utilisateur** et une clé étrangère des tables **proposition** et **messagerie** plusieurs requêtes SQL doivent être faites. Il faut insérer le nouveau tuple correspondant dans la table **utilisateur** en récupérant au préalable toutes les autres informations dans la table concernant cet utilisateur, faire les modifications nécessaires dans les tables **proposition** et **messagerie** et enfin supprimer le tuple avec l'ancien pseudo dans la table **utilisateur**.

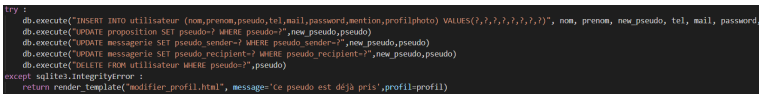

Figure 23: Différentes requêtes SQL

L'utilisateur est ensuite redirigé vers la page **profil**.

2.2.13 Page recherche

Cette page permet à chaque utilisateur d'effectuer des recherches de fruits et légumes selon plusieurs critères: le code postal, le département et la région. Les deux premiers critères de recherches permettent de trouver des résultats à proximité et sont ainsi représentatifs de circuits courts. Le critère de recherche par région est moins pertinent, mais permet à chacun d'avoir une vision d'ensemble de ce qui est proposé sur l'application, et semble ainsi pertinente pour les nouveaux utilisateurs ou ceux qui hésiteraient encore à se servir de l'application et qui souhaiteraient simplement voir ce qui peut se trouver dessus. Les résultats sont **ORDER BY** par code postal, quelque soit le critère de recherche.

2.3 Algorithmes de traitement

2.3.1 Suppression d'une proposition quand la date est dépassée

Afin de ne pas surcharger l'application avec des propositions dépassées, un algorithme supprime automatiquement ces propositions à l'ouverture de l'application.

Pour cela, nous avons créé une fonction **pastPropositions**, ne prenant pas d'argument d'entrée, et ne renvoyant rien. Cette fonction récupère la date actuelle, et, sous forme de liste, les identifiants et les dates d'expiration de toutes les propositions existantes. Ensuite, la fonction compare l'année de fin de la proposition avec l'année actuelle. Si elle est plus petite, la fonction supprime dans la base de donnée la proposition portant le même identifiant. Sinon, si elles sont égales, elle fait les mêmes tests pour le mois, puis le jour.

La complexité de cet algorithme est linéaire.

2.3.2 Vérification de l'extension des fichiers

Lors de l'inscription ou lors de l'ajout d'une proposition, on a la possibilité d'ajouter une photo. Dans les deux cas, il faut bien s'assurer que les fichiers récupérés soient bien des images.

Pour cela, la fonction **allowed_file** récupère le nom du fichier et renvoie un booléen caractérisant si l'extention du fichier est dans la liste **allowed_extention**, définie au préalable. Cette dernière est la liste des extentions voulues. La fonction va séparer le nom donné en paramètre en plusieurs chaînes de caractères, à chaque apparition du caractère **.** , avec la fonction **split**. On aura donc une liste de chaque partie du nom ainsi séparé. Si le nom est un nom de fichier valide, la liste doit être de longueur 2, composée du nom donné par l'utilisateur, et de son extension. Si ce n'est pas le cas, la fonction renvoie **False**. Sinon, elle teste si l'extension est dans la liste **allowed_extention**, et renvoie le résultat de ce test.

La complexité de cet algorithme est linéaire.

2.3.3 Cryptage du mot de passe

Lors de l'inscription ou de la modification du mot de passe d'un utilisateur, ce mot de passe doit être crypté pour des raisons de sécurité. L'algorithme prend en argument notre mot de passe sous forme de chaînes de caractères, le transforme ensuite en liste de caractères et crée une liste de la même taille que notre mot de passe contenant uniquement des 0 qui représentera par la suite le mot de passe crypté.

À l'aide d'une fonction auxiliaire récursive, on va ensuite crypter notre mot de passe, cette fonction va prendre en argument le mot de passe à crypter sous forme de liste et l'indice de la liste donnant la position du caractère à crypter dans notre mot de passe. La fonction prend chaque élément de la liste (donc chaque caractère du mot de passe) et le cherche dans une liste contenant les caractères ASCII : la recherche démarre au premier élément de la liste ASCII et s'arrête lorsque le caractère est trouvé. Tant que le caractère n'est pas trouvé, chaque élément de la liste représentant notre mot de passe crypté, à partir de l'indice pris en argument (qui correspond au caractère actuellement étudié du mot de passe) est incrémenté de 1. Ensuite la fonction auxiliare est réappliquée à la liste représentant le mot de passe privée de l'élément qui vient d'être étudié et à l'indice de la liste donnant la position du caractère à crypter, c'est-à-dire le précédent indice incrémenté de 1.

Lorsque la liste représentant le mot de passe à crypter donnée en argument à la fonction récursive est vide (qu'il n'y a plus de caractère à crypter), la liste de chiffres représentant le mot de passe crypté est alors changée en une liste de caractères en remplaçant chaque nombre par l'élément correspondant dans la liste ASCII (si le nombre est plus grand que la longueur de la liste ASCII, il est décrémenté de cette même longueur jusqu'à arriver à un nombre compris entre 0 et la longueur de la liste ASCII). Cette liste de caractères est alors changée en une chaîne de caractères pour obtenir le mot de passe crypté.

La complexité de cet algorithme est quadratique.

3 Tests et performances

3.1 Suppression d'une proposition quand la date est dépassée

Un ensemble de tests a été réalisé, dans un fichier **test_pastPropose.py**, invocable par pytest. Tout d'abord, nous testons différentes valeurs pour le jour, le mois et l'année de la date (plus petite, égale ou plus grande que la date actuelle). Ensuite, on teste si l'un des trois n'est pas un entier, ou s'il est manquant, et si la date est bien du bon format. Puis, on teste si la date est vide, s'il n'y a pas de date ou s'il n'y a pas de proposition. Enfin, on teste pour deux propositions, lorsque les deux sont dépassées, et lorsque seulement l'une des deux est dépassée.

3.2 Vérification de l'extension des fichiers

Un ensemble de tests a été réalisé, dans un fichier **test_allowedFile.py**, invocable par pytest. Les tests considérés sont les suivants : avec un nom de fichier avec chacune des bonnes extensions, avec un nom de fichier avec une extension non valide, avec seulement un nom de fichier sans extension, avec un nom de fichier et un point, avec une extension valide mais un point dans le nom du fichier, avec une chaîne de caractères aléatoires, avec un nom de fichier vide, avec la chaîne vide, avec un entier et avec une chaîne de caractères très longue.

3.3 Cryptage du mot de passe

L'algorithme de cryptage du mot de passe est l'algorithme le plus complexe de ce projet. Une estimation manuelle de sa complexité nous a conduit à un résultat en $O(n^2)$, de par la présence de la boucle for imbriquée dans la boucle while. Nous avons par la suite souhaité vérifier ce postulat. Cette vérification se situe dans le fichier **test_crypte_mdp.py**. Nous avons pour cela créé une fonction *timer* qui permet de mesurer le temps d'exécution de l'algorithme pour des valeurs croissantes de longueurs de mots de passe, puis d'en tracer la courbe. Les longueurs choisies varient de 1 à 1000 caractères avec un pas de 50. La figure suivante illustre ce comportement quadratique

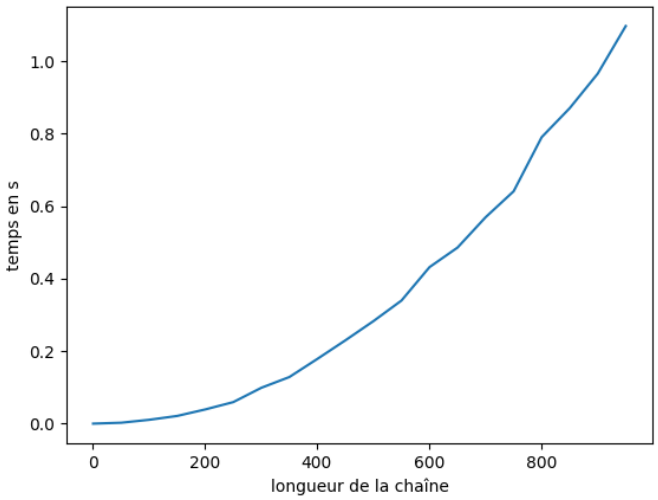

Figure 24: Temps d'exécution en fonction de la longueur de la chaîne

Le cryptage du mot de passe se réalise à l'aide de deux algorithmes. Le premier *mdpcorrect* permet de vérifier le format du mot de passe. Les tests réalisés sur cet algorithme sont présents dans le fichier **test_mdpcorrect.py**. Ces tests se portent sur l'examen de différentes chaînes de caractères, de tous formats et de toutes natures. Les chaînes de caractères vides mais également les éléments qui ne sont pas des chaînes de caractères sont testés.

Le second algorithme est celui qui crypte le mot de passe avant de l'insérer dans la base de données. Ce second algorithme est bien plus complexe et les tests plus difficiles à déterminer. Deux tests principaux ont ainsi été effectués. Le premier s'attarde sur la longueur des mots de passes cryptés et cherche à vérifier que cette longueur est bien égale à la longueur du mot de passe initial. Pour cela, il génère des mots de passe aléatoires parmi la table ascii de longueurs différentes (de 1 à 1000 caractères) et réalise le test de longueur. Le second test s'intéresse à l'unicité du mot de passe crypté, vérifiant ainsi que deux mots de passe différents ne puissent être cryptés de la même manière. Le test génère ainsi un nombre important de mots de passe **différents** et vérifie que la liste contenant les mots de passe cryptés ne contient pas deux fois la même itération d'un mot de passe.

4 Gestion de projet

4.1 Cahier des charges

4.1.1 Contexte et description du projet

Présentation de l'application pour les clients

Les pommes, c'est bon. C'est encore mieux quand elles viennent de notre jardin. On sait comment elles ont été faites, elles sont bios, locales, et c'est ce qui nous les fait aimer encore plus. Mais quand notre pommier est trop gros, on a trop de pommes, et bien souvent la majorité tombent par terre et ne sont jamais récupérées. De même quand on part en voyage et qu'on ne peut pas les ramasser, elles pourrissent aussi sans qu'on puisse en profiter.

Mais pourquoi ne pas en faire profiter les autres ? Comment trouver un moyen d'avertir les personnes intéressées de notre surplus de fruits ou légumes ? C'est de là que vient l'idée de notre application web : une interface de rencontre et de partage, pour échanger, donner et recevoir des fruits et légumes locaux, et venir en aide à nos voisins. Facilitant l'accès à des produits bios et locaux, cette application permet aux personnes de vendre ou donner ce qu'ils ont cultivé, ou de rechercher des produits près de chez eux. Pour cela rien de plus simple ; il suffit de s'inscrire, renseigner sa localisation, et spécifier nos besoins. Grâce à une carte interactive, on peut facilement avoir accès à ce que proposent nos voisins.

Envie de vendre, échanger ou donner ses produits ? Besoin d'aide pour les récoltes que l'on ne peut pas effectuer ? Une petite annonce et le tour est joué ! Et en y ajoutant des photos, les autres utilisateurs se feront une meilleure idée. Mais si personne ne veut de mes produits ? Pas de panique, si au bout d'un certain temps personne ne se propose, l'application vous donnera la possibilité de faire don à une association. Une envie de fruits ou légumes en particulier ? Vous pouvez simplement faire une recherche, et l'application vous retourne les résultats près de chez vous. Quelque soit votre besoin, les propositions tiendront compte de la distance géographique. Enfin, lorsque vous trouvez des personnes intéressées, un système de messagerie permet alors d'échanger diverses informations, de discuter, et d'avoir une trace des engagements de chacun.

Ainsi, cette application web permet aux jardins privés de limiter le gaspillage, mais aussi d'échanger avec d'autres personnes qui partagent vos envies.

Présentation et description technique de l'application

Notre projet consiste en une application web permettant de vendre et d'échanger ses fruits et légumes, tout en proposant des cueillettes directement chez le particulier. Pour cela, il y aura possibilité de déposer des annonces de vente de

fruits et légumes, de troc ou encore de cueillette.

Chaque particulier pourra alors rechercher le produit qu'il désire parmi ces annonces proches de chez lui. En effet, une carte interactive permettra de voir la localisation des différents produits, cette localisation ne sera pas précise mais donnée par zone géographique. Les produits seront proposés dans un périmètre donné autour de la localisation du client, périmètre qui augmentera au fil du temps, dans un objectif de limiter la consommation de carburant provoquée par de longs trajets et donc la pollution atmosphérique.

Un système de messagerie permettra de mettre en contact la personne intéressée et la personne proposant le produit, afin qu'elles puissent convenir d'une date et d'un lieu de rendez-vous. La problématique des cambriolages s'est posée lors de la réflexion sur le système de cueillette, cependant chaque particulier ne donnera sa localisation précise uniquement s'il le souhaite via la messagerie et décidera seul s'il accepte qu'une autre personne vienne chez lui sans sa présence. De plus, les cueillettes peuvent également se faire lorsque la personne est présente, par exemple lors d'une journée de télé-travail.

Dans les derniers jours de consommation du produit, s'il n'a pas pu être vendu ou échangé, il sera proposé à l'utilisateur d'en faire don à une association.

4.1.2 Objectif et périmètre du projet

Nous souhaitons ainsi offrir à nos utilisateurs un service plus complet que ceux existant actuellement. Une application qui leur permet de gérer leurs surplus, mais également d'aider à la récolte même, un élément souvent oublié lorsque l'on parle de gaspillage. Pour cela, nous nous concentrerons sur tout le territoire français.

4.1.3 Description fonctionnelle des besoins

Fonction	Objectif	Degré de liberté	Niveau de priorité
Permettre à l'utilisateur de proposer ses fruits	Saisir facilement ses informations	Faible	Élevé
Accéder à l'ensemble des informations via une carte interactive	Permettre à l'utilisateur de s'y retrouver facilement	Moyen	Moyen
Permettre aux utilisateurs de se contacter via l'application	Faciliter les échanges entre utilisateurs	Élevé	Moyen
Renvoyer vers une association	Ne pas gaspiller alors que des solutions existent	Faible	Élevé

4.1.4 Enveloppe budgétaire et ressources disponibles

Ce projet n'a pas de budget alloué. L'équipe projet est composée de 4 membres qui travailleront directement avec leurs matériels personnels ou celui de l'établissement.

4.1.5 Délais

Le projet se décompose en différents livrables et différentes échéances:

- Le premier livrable se compose d'un rapport à déposer avant le 20/10/2022.
- Le second livrable correspond au rendu du projet. Ce dernier a lieu le 11/01/2023.

4.2 Charte Projet

4.2.1 Contexte

Aujourd'hui, les circuits courts sont les alternatives les plus intéressantes face au défi qu'est devenu la gestion de l'alimentation et de l'eau. De nombreuses solutions sont apparues pour faire face à ce défi. On peut retrouver les jardins partagés, les micro-fermes mais également le partage de ressources produites au sein de jardins privés.

Ces applications sont principalement utilisées par les propriétaires de jardin privés. Chacune d'entre elles permet de répondre à sa manière à la problématique du surplus de production dans ces jardins ainsi que leur gestion.

Le cahier des charges permet d'identifier de manière immédiate les principaux besoins et fonctions auxquels l'application doit répondre. Il s'agit notamment de permettre de mettre en relation des propriétaires de jardins privés, de permettre de gérer le surplus de production mais également de permettre une meilleure gestion de la récolte afin d'éviter les pertes initiales.

4.2.2 Finalités et importance du projet: Business Case

Ce projet est aujourd'hui essentiel afin de répondre aux problématiques actuelles concernant la gestion de l'alimentation en circuits courts. De plus, la nature même du projet et ses coûts extrêmements limités réduisent la néceesité d'un retour sur investissement, permettant ainsi un départ peu risqué. L'application visant à proposer un service sans recherche de bénéfices, l'estimation de ces derniers est inutile.

De manière prévisionnelle, on peut estimer que l'application permettra de mettre en relation de nombreux propriétaires de jardins privés au sein de communes principalement en zone rurale. Sa gestion des récoltes participera à une

nouvelle manière de partager des surplus de fruits non consommés, une innovation qui permettra une ouverture plus grande que la plupart des projets existants.

Les parties prenantes relatives au projets sont: l'équipe projet, composée de quatre membres, les enseignants encadrants, ainsi que les potentiels utilisateurs de l'application.

Le périmètre du projet s'étend sur tout le territoire français, mais la nature même de son fonctionnement favorise une utilisation dans des zones rurales où le nombre de jardins privés est bien plus important. A cette limite géographique s'ajoutent de nombreux prérequis à obtenir: obtention des compétences nécessaires au développement d'une application web et de la création d'algorithme, être dans un cadre légal quant à la possibilité de venir récolter les fruits chez une autre personne.

Cette analyse du Business Case permet de proposer une matrice SWOT identifiant les forces, faiblesses, opportunités ainsi que menaces du projet.

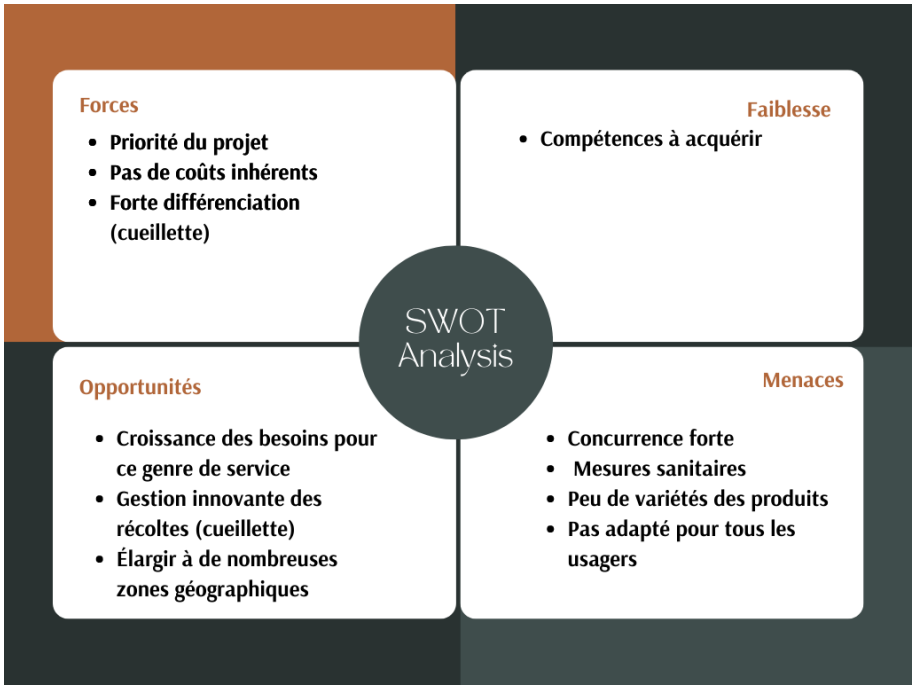

Figure 25: SWOT relatif à la réalisation du projet

En plus de ce SWOT relatif aux contraintes de réalisation, il est possible d'analyser les forces, faiblesses, opportunités et menaces pour l'équipe et le cadre du projet.

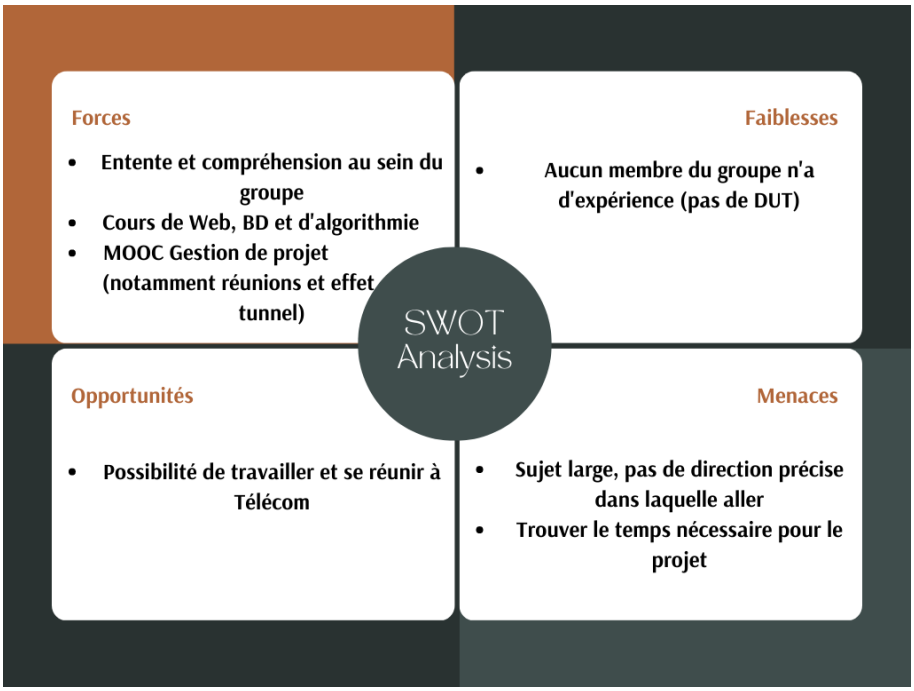

Figure 26: SWOT relatif à l'équipe et aux consignes

Le manque d'expérience peut être limité par l'exploitation de toutes les ressouces du cours et par un approfondissement sur les documentations officielles des langages utilisés, internet ou en demandant conseil. L'état de l'art permet également de recentrer les sujets et de nous montrer les principales fonctions déjà existantes. De plus, la possibilité de travailler à Télécom pour faire des réunions ou travailler à plusieurs sur un sujet délicat est un gain de temps précieux.

4.2.3 Objectifs et résultats opérationnels

Ces applications connaissent un franc succès aujourd'hui, de par la nécessité de faire face au défi du gaspillage alimentaire, ainsi que par le développement des circuits courts. L'innovation ici proposée, concernant la participation à la récolte même des fruits, permet de s'attaquer à l'une des principales causes de perte de ces derniers. Ces différents éléments sont des indicateurs de la possibilité de succès de ce projet.

4.2.4 Ressources

Les moyens à mobiliser sont : PC personnels et/ou de l'école, postgress4SQL, python3

4.2.5 Jalons

Jalon	Description	Date
Etape 1 : Exigences opérationnelles	Validation de l'idée lors de la soutenance	22/10/22
Etape 2 : Création de la base de données	Création de la base de données relative aux fruits et légumes disponibles	20/11/22
Etape 3 : Réalisation de l'interface	Réalisation de l'interface utilisateur	20/12/22
Etape 4 : Ajout d'une carte	Ajout d'une carte permettant de repérer où se situent les fruits et légumes	20/12/22
Etape 6 : Réalisation messagerie	Réalisation d'un système de contact entre les utilsateurs (messagerie) pour obtenir plus d'informations sur un produit en vente ou les conditions d'accueil pour une cueillette	02/01/23

4.3 Matrice RACI

La matrice RACI permet de nous répartir globalement le travail dans le projet. En effet pour la plupart des sous-lots, on peut identifier un responsable (l'autorité A), ceux qui réalisent (R), des conseillers (C) et des personnes à informer (I). Nous n'avons pas fait figurer les I dans la matrice pour ne pas l'encombrer, car nous avons décidé au début du projet d'un commun accord que nous nous informerions le plus possible les uns les autres sur l'avancement de toutes les parties.

Matrice RACI

Projet PP2I

Livrable	Équipe projet			
	Pierre	Aurélie	Nicolas	Manon
Base de données				
Schéma relationnel base de données	R	R	R	R
Création base de données			AR	R
Conception des requêtes	R	R	R	R
Serveurs Web				
Page associations				AR
Page d'accueil	AR			
Barre de navigation	AR			
Page de connexion		AR		RC
Page d'inscription		AR		
Messagerie	AR		C	
Page de profil		RC		AR
Page d'ajout d'une proposition	AR			
Page de recherche d'une proposition			AR	
Implémentation des cartes			AR	
Page de présentation d'une proposition	AR			
Page de modification du profil				AR
Algorithmes de traitement				
Cryptage de mot de passe		C	RC	AR
Suppression d'une proposition à sa date de fin	AR			
Vérification du format d'un fichier	AR			
Vérification que le mot de passe saisi est du bon format		C	C	AR
Création d'un nouvel identifiant lorsqu'un utilisateur s'inscrit				
Rapport				
Introduction		AR		
Base de données			AR	
Serveurs Web	R	R	R	R
Algorithmes de traitement	R		R	R
Tests et performances	R		AR	R
Gestion de projet		AR	R	R
Analyse post mortem		AR		

Figure 27: Matrice RACI du projet

5 Analyse Post Mortem

5.1 Résultats

Nous avions la volonté de réaliser une application complète avec de nombreuses pages et fonctionnalités. Une bonne partie a été réalisée même si au cours du projet nous avons laissé de côté certaines idées pour approfondir ce que nous avions commencé. Des connaissances d'autres langages tels que le JavaScript auraient pu être utiles mais nous avons su contourner ces difficultés ou ajuster nos idées pour obtenir une application fonctionnelle. Cette dernière permet de s'inscrire et se connecter par la suite, de proposer des fruits et des légumes issus de son jardin à d'autres utilisateurs ou de venir récupérer ou cueillir ceux des autres. Les propositions se retrouvent sur la page d'accueil, dans la zone de recherche ou encore sous le profil de celui qui les propose. De plus, vers la date de fin d'une proposition, la page association permet de donner les fruits et légumes avant leur date de péremption. Enfin, les propositions dépassées en date sont automatiquement supprimées.

5.2 Notions apprises

5.2.1 Gestion de Projet

La réalisation d'un état de l'art, la rédaction d'une charte projet, l'analyse de matrices SWOT ou encore la réalisation d'une matrice RACI sont autant d'étapes qui nous ont permis de comprendre le projet, ses enjeux et dangers, et de veiller à répartir et suivre l'avancement des tâches au sein du groupe. De plus, les réunions régulières ont évité à des membres du groupe de chercher des solutions seuls trop longtemps ainsi que la création d'un effet tunnel.

5.2.2 Base de données, WEB et Algorithmie

- Nous avons pu créer notre propre base de données après une réflexion collective pour trouver la base optimale. Cette dernière a évolué selon les besoins du projet.
- De nombreuses connaissances ont été acquises lors du développement des différentes pages du site aussi bien au niveau du HTML que du CSS selon les fonctionnalités des pages telles que les formulaires ou l'insertion et la récupération de données spécifiques de la base de données.
- L'algorithme de cryptage, de suppression des propositions et les autres plus petits algorithmes réalisés nous ont également aidés à nous familiariser avec les méthodes de tests.

5.3 Points à améliorer

Pour notre prochain projet, nous consacrerons moins de temps à la maquette que nous avions réalisé pour investir ce temps dans la réalisation du projet

même. De plus, il n'est pas nécessaire d'être trop ambitieux, notre objectif est d'avoir un travail réalisant les fonctionnalités qui nous paraissent essentielles sans chercher à multiplier les possibilités du site.