Rapport Final projet PPII

Cappellina Arthur, Mentec Jeremi, Pernin Thomas

Janvier 2023

Table des matières

Chapitre 1

Introduction

Pour privilégier les circuits courts dans le domaine alimentaire, nous avons développé "Potagix", un site permettant aux utilisateurs de mettre à disposition les produits qu'ils ont pu cultiver. La conception de ce site a nécessité l'utilisation de différentes bases de données, la création d'un serveur web, ainsi que la conception de différents algorithmes de traitement, en fonction des besoins de l'application. Notre objectif a alors été de faciliter l'accès aux micro-fermes, jardins partagés et vendeurs indépendants pour tous les utilisateurs de notre application. Dans ce rapport, nous allons vous présenter les différentes fonctionnalités que nous avons trouvées pertinentes, ainsi que la manière dont nous nous sommes organisés pour les réaliser.

Chapitre 2

Fonctionnalités et description du projet

L'application que nous avons développée s'appelle "Potagix". Cette application possède plusieurs fonctionnalités proposées aux utilisateurs.
Tout d'abord, la page "accueil" propose à l'utilisateur non connecté les jardins partagés et les AMAP récemment ajoutés. Cependant, si l'utilisateur est connecté, on lui propose à la fois les AMAP récemment ajoutées et des produits qui pourraient lui plaire grâce à un algorithme qui prend en considération ses dernsières commandes.
Puis, grâce à une carte interactive l'utilisateur peut voir les produits, les micro-fermes et les AMAP présents sur Nancy. Il peut choisir les produits qu'il veut voir grâce à des filtres mis à sa disposition. Ces filtres permettent de choisir s'il veut avoir accès à des produits payants, gratuits ou encore, au type de produit (pomme de terre, carotte ...).
Ensuite, grâce à l'onglet amap, l'utilisateur peut avoir accès aux différentes amap présentent proches de chez lui (la distance est affichée sous le nom de chacune des amaps). En cliquant sur une des amaps, l'utilisateur peut avoir des informations suppémentaires, telles qu'une description de l'amap sélectionnée, ainsi qu'un lien pour avoir accès à cette dernière (si elle possède un site internet).
Une autre des fonctionnalités de l'application est l'épicerie virtuelle. Cette épicerie permet aux différents utilisateurs de réserver des produits proposés par les différentes structures de l'application (particulier, micro-ferme, jardin partagé) en fonction du périmètre de recherche sélectionné (en kilomètres). Le jour de la collecte ainsi que l'horaire de début et de fin est indiqué quand la personne réserve le produit.

L'utilisateur peut aussi proposer un produit sur l'application. Il peut décider de mettre une image du produit, le type (fruit ou légume), une description, la quantité, le prix, la date de la colecte, ainsi que l'entité qui va vendre le produit (lui même en tant que particulier, sa micro-ferme ou son jardin partagé si il en possède)
Pour finir, l'utilisateur a accès à un onglet "Graine", lui permettant de savoir quelles graines planter pour un mois choisi ou pour le mois en cours.
Un utilisateur peut aussi ajouter une micro-ferme ou un jardin partagé s'il en est le propriétaire. Il peut accéder à cette fonctionnalité via la page "/mon-compte". C'est de cette manière que l'on peut ajouter une micro-ferme ou un jardin partagé dans Potagix. Enfin, afin de pouvoir acheter les produits, un utilisateur peut acheter des Ducky, la monnaie utilisée sur Potagix. Pour passer à la commande, il peut ensuite utiliser la fonction "panier" qui stocke les produits que l'utilisateur souhaite commander.

Chapitre 3

Partie Technique

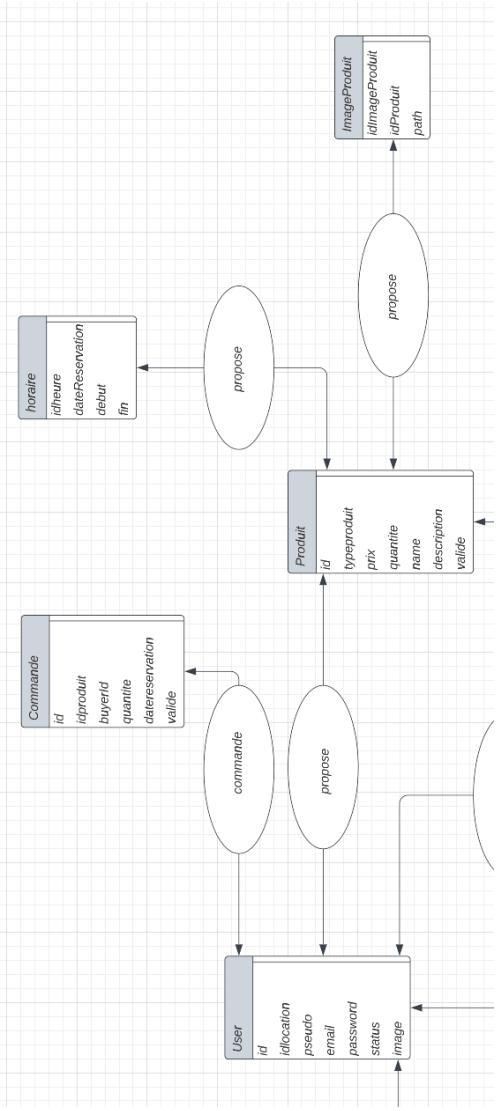
3.1 Conception et implémentation des bases de données

Ici, on s'intéresse à la partie base de données de l'application. L'objectif est de décrire un système de base de données fonctionnelles. Tout d'abord, on va effectuer le schéma des relations de la base, puis, nous établirons le modèle entité/association associé aux différentes tables.
Schéma de la base :

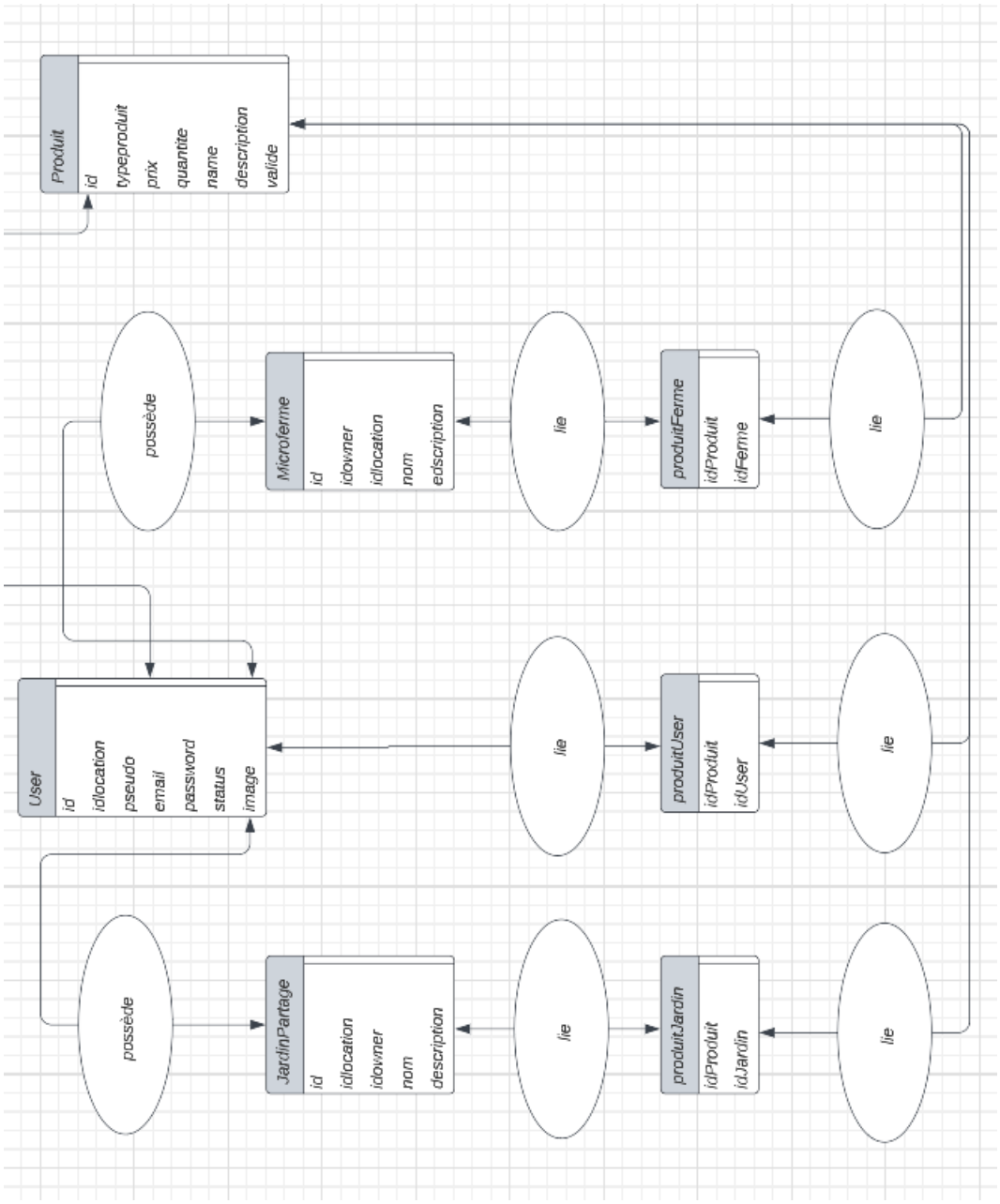
- Utilisateurs(id, email, pseudo, password, statut, adresse, imageProfil)
- Produit(id, typeProduit, prix, quantite, #vendeurId, nom, description, #idHeure, #idImage)
- Commande(#acheteurId, #vendeurId, #produitId, quantite, dateReservation)
- JardinPartage(id, #ownerId, #idLocation, nom, description)
- MicroFerme(idMicroFerme, #ownerId, #idLocation, nom, description)
- AMAP(idAmap,#ownerId, #idLocation nom, link, description, image)
- ImageJardin(idImageJardin, #idJardin, path)
- ImageAMAP(idImageAmap, #idAmap, path)
- ImageProduit(idImageProduit, #idProduit, path)
- ImageFerme(idImageFerme, #idFerme, path)
- Horaires(idHeure, dateHeures, debut, fin)
- Graine(id, nom, debutPlantation, finPlantation, image)
- ProduitJardin(#JardinId, #produitId)
- ProduitFerme(#FermeId, #produitId)

— ProduitUser(#UserId, #produitId)
— locations(id, adresse, latitude, longitude)
— codeConfirmation(id, code)
— transactions(id,#userId, cagnotteTemp, valide, date)

Modèle entités/associations :



Modèle "entités/associations" partie 1



Modèle "entités/associations" partie 2

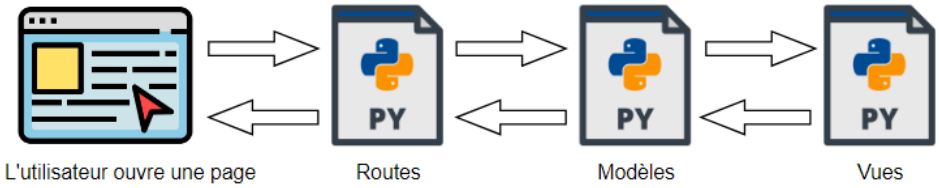
Ci-dessus, vous pouvez voir le modèle "entités/associations". Le principal intérêt du modèle est de pouvoir voir les relations entre les différentes tables du schéma précédent. Par exemple, nous pouvons voir qu'un utilisateur peut soit commander des produits, soit en vendre et que pour cela, nous avons besoin de la table "Commande". Il en va de même pour les autres tables du schéma, comme les tables "Users" permettant d'enregistrer un utilisateur, la table "produit" permettant d'enregistrer un produit ou encore la table "JardinPartage" permettant d'enregistrer un jardin partagé.

On notera que les tables "ProduitJardinsPartages", "ProduitUser" et "ProduitMicroFerme" permettent de faire le lien entre les produits et la structure qui vend les produits. On peut alors retrouver les produits vendus par une ferme mais également retrouver les fermes proposant un certain type de produit.

Enfin, il était nécessaire de créer des tables "Images" afin de pouvoir stocker plusieurs images d'un produit ou d'une ferme.

3.2 Conception et implémentation du serveur web

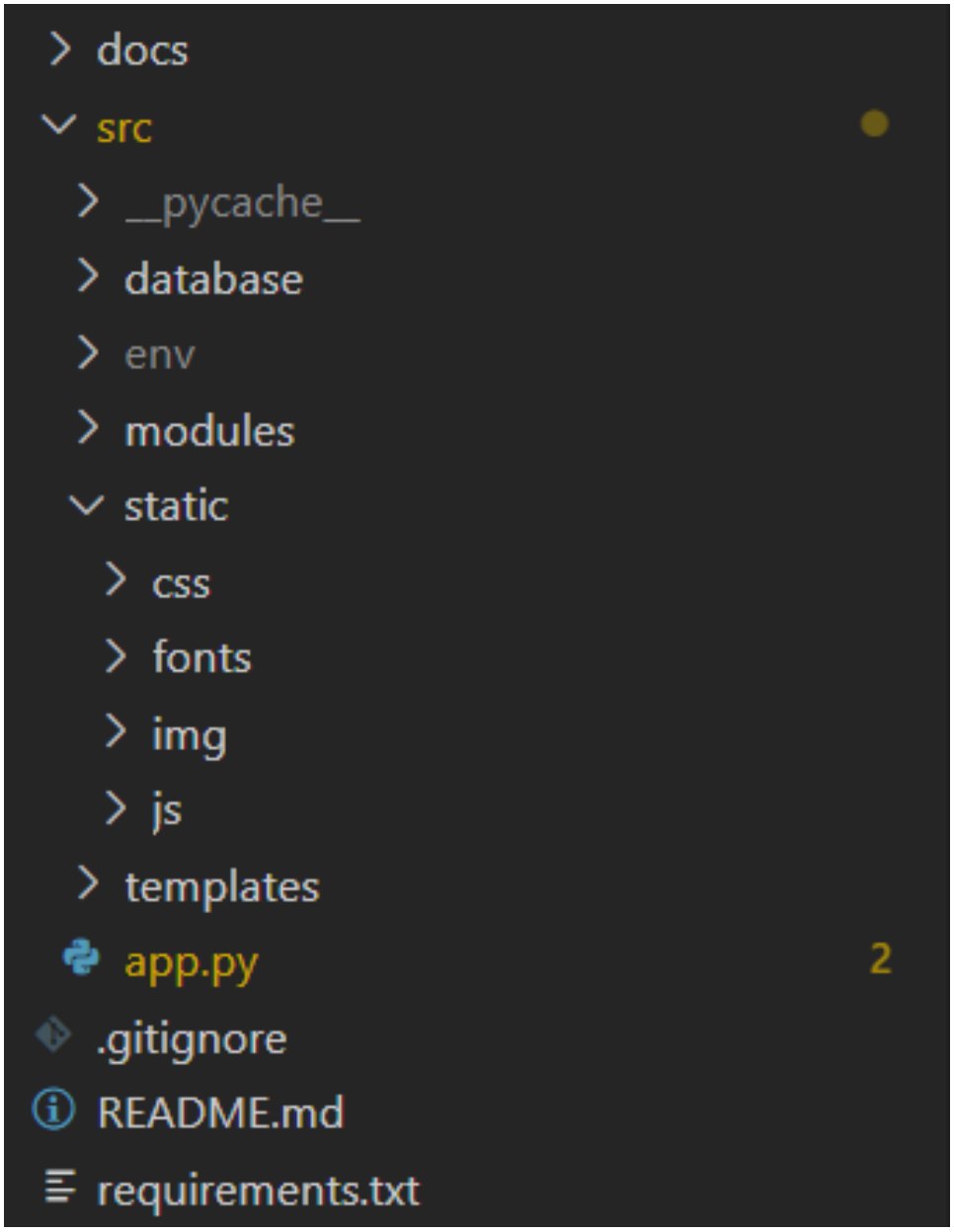
Nous sommes ici dans le coeur du projet. C'est la partie la plus importante car si elle ne fonctionne pas ou qu'elle est mal implémantée, c'est l'intégralité du projet qui est remis en cause. Par conséquent, il a fallu organiser le code et la conception du serveur de sorte à ce que l'on puisse facilement corriger des bugs et ajouter des fonctionnalités. Pour cela, nous avons décidé de séparer chaque partie du serveur dans différents fichiers (cf. ci-dessous).



On peut alors voir que la structure du serveur se décompose en trois parties distinctes. La première est le fichier où les routes sont stockées. Ce fichier est appelé au moment où l'utilisateur décide de générer une page sur le site internet. A noter que dans une approche MVC, cela correnspondrait au controleur. Lorsque le fichier contenant les routes sait quelle page l'utilisateur veut générer, il appelle un fichier "modèle" qui a pour objectif de générer les données à afficher. Une fois que ces données sont générées, elles sont envoyées à un fichier "vue" afin qu'elles soient affichées. Prenons un

exemple pour être plus clair. L'utilisateur veut afficher la page d'un produit, il entre alors l'URL correspondant ("/produit/8" par exemple). Le fichier "route" détecte que l'utilisateur veut afficher un produit, il demande alors au fichier modèle consacré au produit de récupérer les données du produit à l'identifiant 8. Ces données sont alors envoyée au fichier "vue" qui affiche le produit sur la page de l'utilisateur.

Une fois que la communication entre les différents fichiers est possibles, il faut créer une arborescence de fichiers (manière dont les fichiers sont ordonnés) cohérente et facile à utiliser. Vous pouvez voir l'arborescence choisie ci-dessous.



Nous pouvons voir les fichiers et dossiers à la racine du projet. Parmi les fichiers, nous remarquons la présence des fichiers nécessaires à un projet informatique tels que le "README.md" qui contient une description du projet ou le "requirements.txt" qui contient les librairies nécessaires au projet.

Le premier dossier est le dossier "docs" dans lequel les comptes rendus de réunion et rapports sont stockés. Le second dossier stocke tous les fichiers de code du projet. Dans ce dossier, on remarque le fichier "app.py" qui correspond au fichier qui contient les routes du site internet. De plus, nous pouvons voir d'autres dossiers. Tout d'abord, le dossier "modules", contient tous les modèles du projet (comme sur le schéma précédent), tout comme le dossier "templates" qui contient les vues du projet. Ensuite, nous avons le dossier "database" qui contient la base de données. Pour finir, le dossier "static" permet de stocker d'autres fichiers tels que les images, le css (style du site internet), les polices d'écritures ou les fichiers javascripts (fichiers permettant de gérer l'intéraction entre l'utilisateur et le site internet).

3.3 Conception et implémentation des algorithmes de traitement

3.3.1 algorithme de distance

Pour de nombreuses fonctionnalités de l'application, nous avons besoin d'un algorithme permettant de calculer la distance entre les différents utilisateurs. Le but de cet algorithme est de calculer la distance minimale entre deux points se trouvant sur une sphère.
Pour réaliser cet algorithme, nous avons décidé de travailler avec la latitude et la longitude décimale des différents utilisateurs. Ces données sont récupérées lors de l'inscription d'une personne et stockées dans une session lorsque l'utilisateur se connecte à son compte.
Une fois ces données récupérées, nous les passons en radians afin d'effectuer les calculs suivants. Nous avons décidé d'utiliser la méthode de calcul appelée "la méthode des sinus" (une des méthodes trouvées sur http://villemin.gerard-.free.fr/aGeograp/Distance.htm
Voici la fonction implémentée :

```python
def distance2(lat1,long1,lat2,long2):
    r = 6378137
    lat1rad = lat1/180*math.pi
    lat2rad = lat2/180*math.pi
    long1rad = long1/180*math.pi
    long2rad = long2/180*math.pi
    inte = math.sin(lat1rad)*math.sin(lat2rad)+math.cos(lat1rad)*math.cos(lat2rad)*math.cos(abs(long2rad-long1rad))
    print(inte)
    dist_point = r*math.acos(inte)
    return int(dist_point/1000)
```

Algorithme de calcul des distances

Une fois le calcul réalisé avec la méthode des sinus, nous multiplions le résultat obtenu par 6378137, correspondant au demi-grand axe de la sphère sur laquelle le calcul est effectué. Une fois ce calcul effectué, nous obtenons la distance entre deux utilisateurs (en kilomètres).

Nous avons également eu besoin d'un autre algorithme, permettant de passer d'une adresse postale à la latitude/longitude décimale. Nous avons donc eu recours à l'API nommée 'Nominatim'. Cette API utilise les données de OpenStreetMap pour convertir l'adresse postale donnée en paramètre en latitude et longitude.
Voici la fonction implémentée :

```
def conv_adresse_to_GPS(adresse):
    location = geocoder.geocode(adresse)
    if location != None:
        return (location.latitude,location.longitude)
    else :
        return "Adresse non trouvé"
```

Algorithme permettant d'obtenir la latitude et longitude (décimale)

Nous avons rajouté le test "location != None" afin de s'assurer que l'adresse donnée en paramètre renvoie des coordonnées géographiques correctes.

3.3.2 algorithme de préférence

Afin de créer une page d'accueil spécialisée pour chaque utilisateur connecté, nous avons décidé de mettre en place un algorithme permettant de trouver une micro-ferme, une AMAP ou un jardin partagé qui pourrait plaire à l'utilisateur. L'objectif était donc de créer un algorithme de préférence cohérent et efficace. Pour cela, il a fallu énumérer les paramètres à prendre en considération. Le premier est la distance, plus l'utilisateur est proche d'une entité, plus il y a de chances que cette entité lui soit proposée. Ainsi, le facteur de distance se calcule comme suit : 1 / distance en km, ce qui permet d'avoir une courbe qui décroît de moins en moins en vite. Par exemple, entre une ferme à onze kilomètres et une autre à douze kilomètres, l'impact de la distance n'est pas important. Le second paramètre est un facteur de similitude

entre ce qu'a acheté l'utilisateur et ce que l'entité propose. Par exemple, si l'utilisateur achète souvent des fruits et qu'une ferme propose des fruits, cette ferme lui sera probablement proposée sur la page d'accueil.

3.3.3 algorithme de fusion

Le dernier algorithme dont nous avons eu besoin nous permet de fusionner les résultats des requêtes effectuées (pour avoir un unique jeu de données à envoyer à nos templates).
Voici l'algorithme implémenté :

```
def fuse(l1,l2):
    res = []
    temp=[]
    first = l1
    second = l2
    i = 0
    while first != []:
        for item in first :
            for word in item :
                temp.append(word)
            for word2 in second[i]:
                temp.append(word2)
            res.append(temp)
            first.pop(0)
            i+=1
            temp = []
    return res
```

Algorithme de fusion

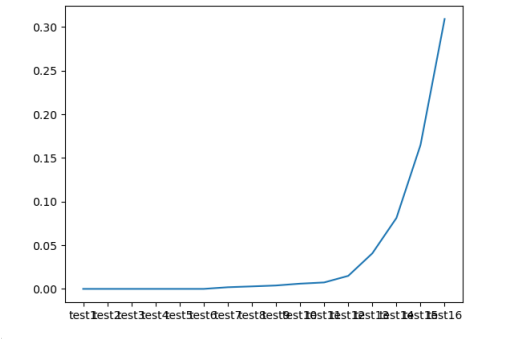
Cette fonction prend en argument deux listes (les résultats de deux requêtes différentes) pour renvoyer une unique liste de liste. Dans les listes intérieures, cela nous permettait d'avoir toutes les informations relatives à un produit (qui le vend, la date de récupération, le lieu de collecte ...)

Chapitre 4

Tests et Performances

4.1 Fonction fuse

La fonction implémentée est de compléxité temporelle exponentielle. Sa compléxité spatiale est de compléxité linéaire, en fonction des listes passées en argument (la longueur d'une des deux listes, car elle possède la même longueur). Pour tester cette fonction, nous avons passé des paramètres de taille croissante (passant de listes vide , à des listes possédant 20000 tuples, possédant eux mêmes 100 éléments). Les tests de cette fonction sont disponibles dans le fichier "test-fuse".
Voici le graphique des mesures temporelles de la fonction "fuse" :

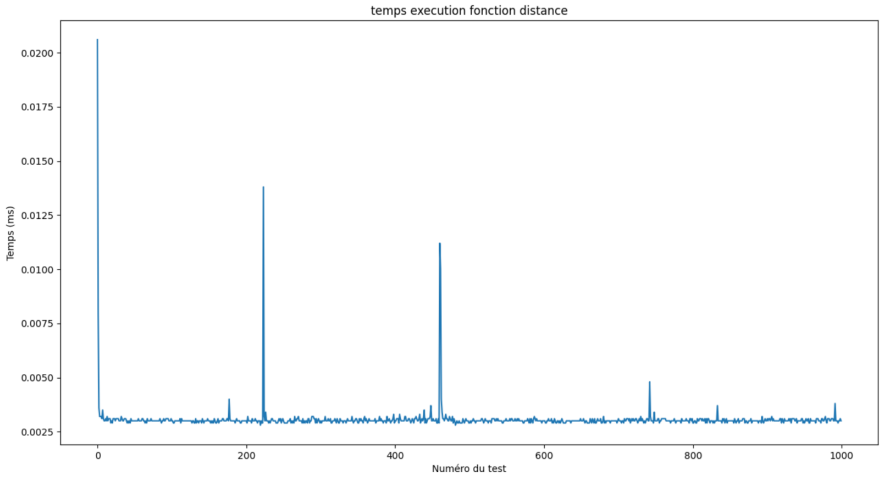


Graphique représentant les temps d'exécution de la fonction fuse

Comme nous pouvons le voir sur le graphique, à partir du huitième test, le temps d'exécution semble augmenter de façon exponentielle, ce qui était attendu passant de 0s pour le premier test à 0,51 s pour le dernier test

4.2 Fonction distance

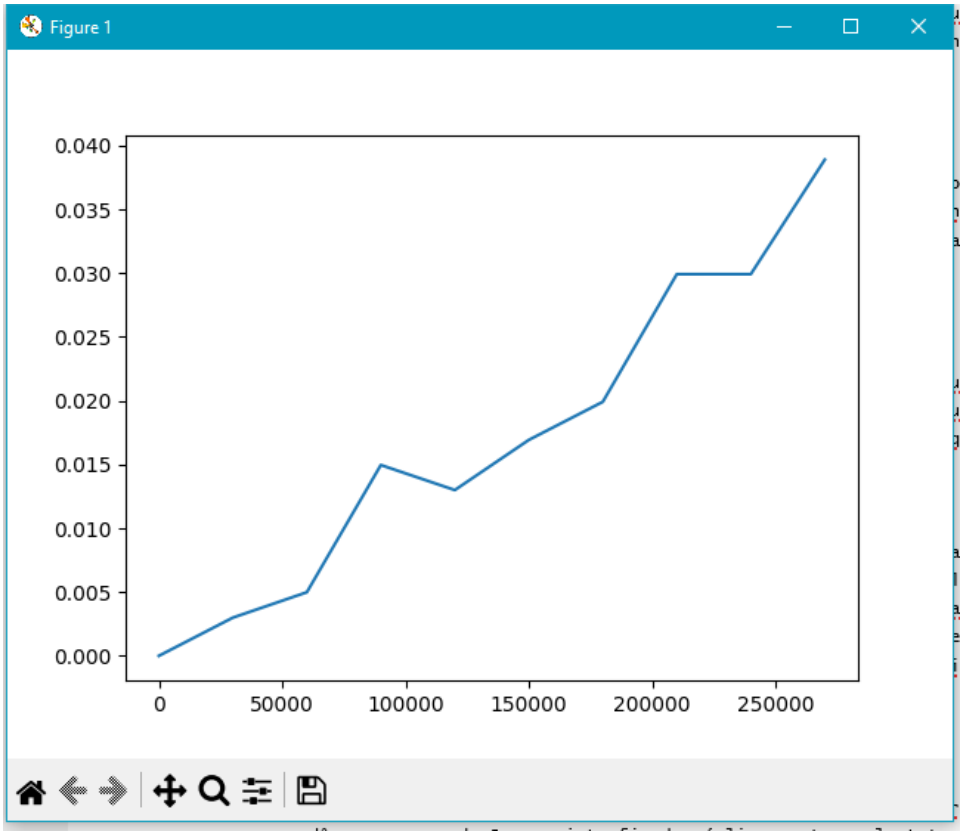
Il s'agit là d'une fonction très basique mais nécessaire qui calcule une distance entre deux positions GPS à l'aide d'une formule mathématique. La complexité est constante car les valeurs des coordonnées GPS n'ont pas d'influence sur le calcul de la distance. Le graphe ci-dessous montre le temps d'exécution de la fonction pour 1000 couples de positions différentes choisies aléatoirement. On constate à l'exception de quelques irrégularités que le temps d'exécution de la fonction est constant (3 microsecondes).



4.3 Fonction des préférences

Pour les préférences, nous avons décidé de tester la fonction qui permet d'identifier à quel point un utilisateur commande le type de produit que propose un jardin partagé ou une micro-ferme. Pour cela, la fonction renvoie un réel entre 0 et 1 en fonction du taux de similitude entre ces derniers.

Nous avons donc testé plusieurs cas, parmi lesquelles : pour des utilisateurs n'ayant achetés aucun produit, des utilisateurs ayant uniquement des produits en commun avec une micro-ferme ou jardin partagé... L'objectif était alors de vérifier que chaque cas renvoyait un cas cohérent. De plus, afin de vérifier le comportement de la méthode sur de grandes données, nous avons créé un graphique qui montre comment se porte la fonction lorsque nous passons des utilisateurs qui ont énormément de produits commandés. Nous obtenons la courbe suivante.



Nous remarquons que nous obtenons bien une fonction linéaire, ce qui est esperé pour cette fonction.

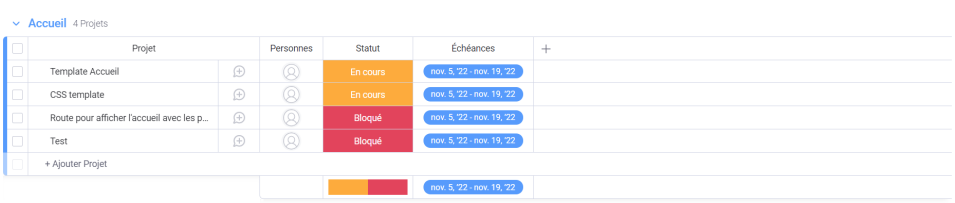
4.4 Fonction de vérification de l'extension d'un fichier

La fonction implémentée est de compléxité linéaire, en fonction de la taille de la chaîne de caractères passée en argument. Pour effectuer les tests de validation de la fonction, nous avons traité les trois cas acceptés par la fonction (le format jpg, jpeg et png), ainsi que des extensions qui ne correspondent pas aux trois cités precédemment. Les tests de cette fonction sont disponibles dans le fichier "test-validation.py"

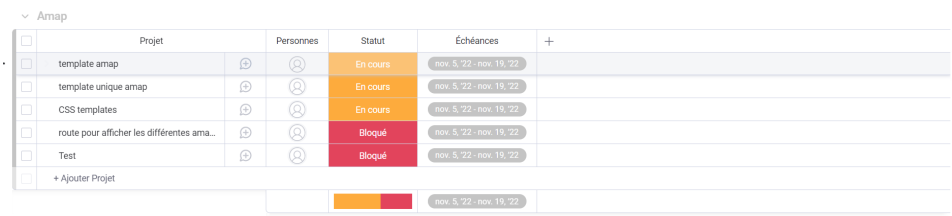
Chapitre 5

Gestion de Projet

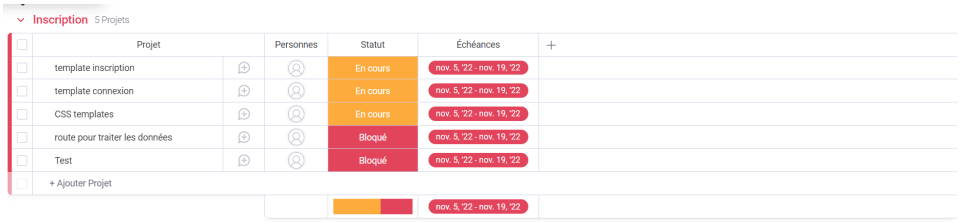
Afin de mener à bien la conception de notre projet, nous avons établi différents types de documents : matrice SWOT, matrice RACI, WBS .... Ces différents documents étaient présents dans le premier rapport du projet. Pour la phase de développement du projet, nous avons établi un diagramme de Gannt pour échelonner le travail à faire. Voici différentes parties du tableau pour le diagramme de Gannt :



Première partie du tableau pour le diagramme de Gant



Deuxième partie du tableau pour le diagramme de Gant



Troisième partie du tableau pour le diagramme de Gant

Pour concevoir ce tableau, nous avons utilisé le WBS réalisé précédemment. Les différentes couleurs représentent la personne qui doit implémenter la fonctionnalité (bleu pour Arthur, Rouge pour Jéremi et Gris pour Thomas). Grâce à ce tableau, nous avons pu construire le diagramme de Gannt suivant :

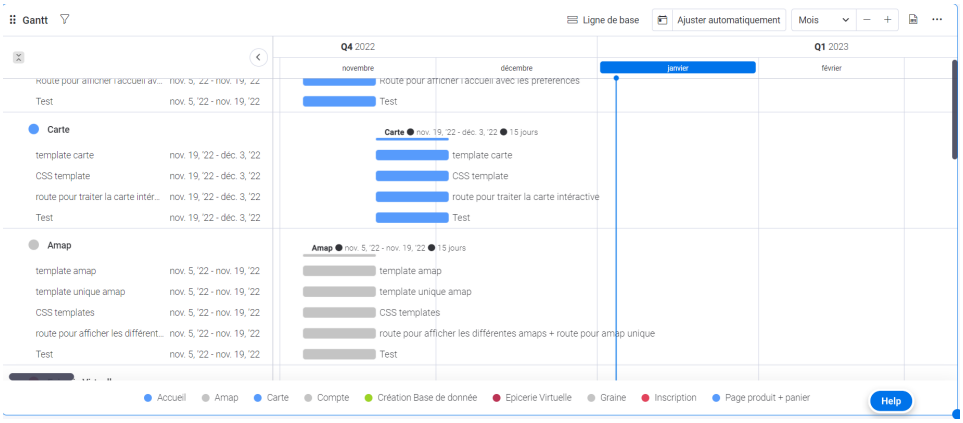
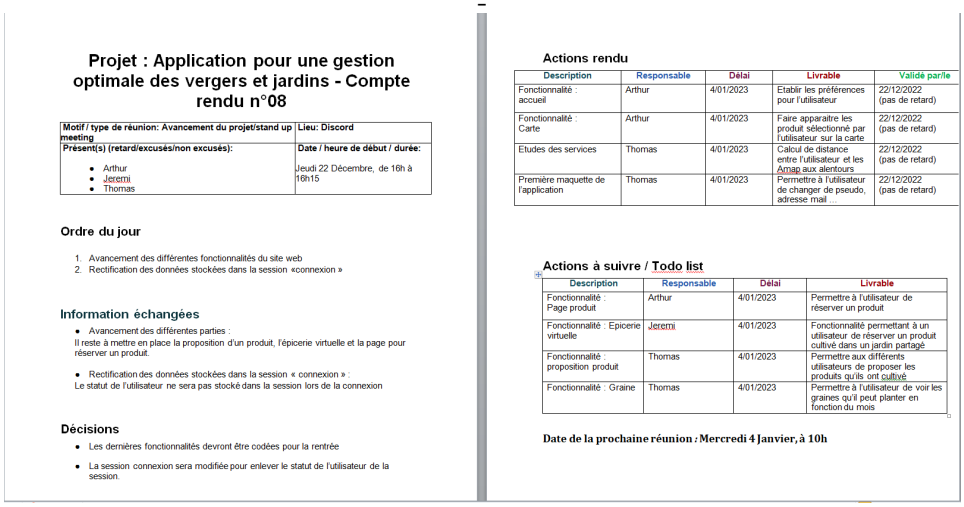


Diagramme de Gannt du projet

Lorsque nous commencions une nouvelle tâche, nous débloquions la partie du tableau qui correspondait à notre fonctionnalité (dans le tableau, on passe de "bloqué" à "en cours"). Une fois les foncionnalités validées, nous passions le statut de la fonctionnalité de "en cours" à "fait". Nous pouvions ainsi voir les différents progrès que nous faisions tout au long du projet.

Nous avons décidé d'allouer une période de 2 semaines pour implémenter les fonctionnalités dont nous avions la charge (sauf pendant la péride des partiels). A la fin des deux semaines, nous effectuions une réunion d'avan-

cement. Cette réunion avait pour but de contrôler si nous avions bien codé la fonctionnalité qui nous avait été attribuée, afin de passer à la suivante. Un compte rendu de la réunion était ensuite rédigé pour dire les fonctionnalités qui ont été rendues, ainsi que les nouvelles fonctionnalités que nous commencions. Voici un exemple de compte rendu de réunion : A la fin de



Projet : Application pour une gestion optimale des vergers et jardins - Compte rendu n°08

Motif / type de réunion: Avancement du projet/stand up meeting	**Lieu: Discord**
Présent(s) (retard/excusés/non excusés): • Arthur • Jeremi • Thomas	**Date / heure de début / durée:** Jeudi 22 Décembre, de 16h à 16h15

Ordre du jour

1. Avancement des différentes fonctionnalités du site web
2. Rectification des données stockées dans la session «connexion »

Information échangées

- Avancement des différentes parties :
Il reste à mettre en place la proposition d'un produit, l'épicerie virtuelle et la page pour réserver un produit.

- Rectification des données stockées dans la session « connexion » :
Le statut de l'utilisateur ne sera pas stocké dans la session lors de la connexion

Décisions

- Les dernières fonctionnalités devront être codées pour la rentrée
- La session connexion sera modifiée pour enlever le statut de l'utilisateur de la session.

Actions rendu

Description	Responsable	Délai	Livrable	Validé par/le
Fonctionnalité : accueil	Arthur	4/01/2023	Etablir les préférences pour l'utilisateur	22/12/2022 (pas de retard)
Fonctionnalité : Carte	Arthur	4/01/2023	Faire apparaitre les produit sélectionné par l'utilisateur sur la carte	22/12/2022 (pas de retard)
Etudes des services	Thomas	4/01/2023	Calcul de distance entre l'utilisateur et les Amap aux alentours	22/12/2022 (pas de retard)
Première maquette de l'application	Thomas	4/01/2023	Permettre à l'utilisateur de changer de pseudo, adresse mail ...	22/12/2022 (pas de retard)

Actions à suivre / Todo list

Description	Responsable	Délai	Livrable
Fonctionnalité : Page produit	Arthur	4/01/2023	Permettre à l'utilisateur de réserver un produit
Fonctionnalité : Epicerie virtuelle	Jeremi	4/01/2023	Fonctionnalité permettant à un utilisateur de réserver un produit cultivé dans un jardin partagé
Fonctionnalité : proposition produit	Thomas	4/01/2023	Permettre aux différents utilisateurs de proposer les produits qu'ils ont cultivé
Fonctionnalité : Graine	Thomas	4/01/2023	Permettre à l'utilisateur de voir les graines qu'il peut planter en fonction du mois

Date de la prochaine réunion : Mercredi 4 Janvier, à 10h

Compte rendu de la 9e réunion

la réunion, nous décidions d'une date pour la prochaine réunion. Celle-ci se situait dans la semaine où nous devions avoir fini d'implémenter la fonctionnalité dont nous avions la responsabilité (cette réunion avait généralement lieu 2 semaines après la dernière réunion).

Chapitre 6

Post Mortem

Lors de la réalisation de notre projet, nous avons eu plusieurs difficultés lors de l'implémentation des différentes fonctionnalités. De ces difficultés, nous en avons tiré des leçons à la fois technique et organisationnelle

— **Envoie de mail avec Flask :**
Lors de l'inscription ou lorsqu'un utilisateur effectuait une proposition incorrecte d'un produit, nous devions envoyer un mail à cet utilisateur. Cependant, la documentation Flask ainsi que les différents tutoriels n'étaient pas forcément clairs ou n'étaient pas totalement complets. Nous avons donc pris plus de temps que prévu afin de bien comprendre et de bien implémenter l'envoi des mails.

— **Problème avec la base de données :**
Nous avons rencontré quelques problèmes avec la base de données, au début de notre projet. Dans certaines des tables, des colonnes avaient des noms qui étaient trop ambigus. Nous avons donc décidé de refaire les différentes tables qui possédaient ce défaut, afin de continuer le projet sans problèmes au niveau de la base de données

— **Algorithme de distance :**
Nous avons rencontré deux soucis au niveau de l'algorithme de distance. Tout d'abord, nous devions choisir une méthode parmi plusieurs afin de calculer la distance, sans perdre en précision. Le deuxième problème était que le site proposant la méthode des sinus n'indiquait pas que les données traitées étaient converties en radians. Nous nous en sommes rendus compte lorsque nous avons testé la fonction implémentée et nous avons rectifié l'erreur.

— **Communication avec l'API de google map :**
Pour la fonctionnalité de la carte interactive, nous avions besoin d'une API de google maps. Cependant, l'appel à cette API n'était pas simple à effectuer. En effet, nous avons dû passer par du Javascript afin de réaliser cet appel et traiter les données relatives à la création de la carte. Cela nous a permis de comprendre comment fonctionnait un appel API et surtout comment fonctionne les applications ou sites internet qui proposent d'utiliser une carte virtuelle.-

— **Nomenclature de la base de données :**
Un des problèmes que nous avons aussi recontré est que nous n'avons pas créé de nomenclature claire pour la base de données. Par exemple, dans la table Microferme l'atttribut correspondant à "identifiant" se nomme "idMicroFerme" alors que dans JardinPartage, il se nomme "id" et non pas "idJardinPartage". Cela provoque un manque de clarté et une perte de temps lors du développement

— **Manque de communication et d'organisation sur la structure du projet :**
Pour prendre un peu plus de hauteur, la qualité de notre travail aurait pu être meilleure. En effet, dans différents modèles nous écrivons des méthodes qui réalisent les mêmes fonctionnalités. Avec une meilleure communication et une meilleure organisation, nous aurions été plus rapides et le projet aurait été plus clair.

Nous avons aussi pensé à certaines améliorations à apporter à notre projet :

— **Conseil pour les graines :**
L'onglet des graines n'est actuellement pas très développé : il permet à l'utilisateur de connaître les graines qu'il peut planter. Pour rendre cet onglet plus utile, nous pensions ajouter des conseils donnés par les utilisateurs. Cela permettrait aux utilisateurs d'avoir des conseils sur la manière de planter, des conseils en fonction du climat, de leur localisation ....

Chapitre 7

Conclusion

7.1 Conclusion générale

Nous avons apprécié de réaliser ce projet. Nous sommes globalement satisfaits de ce que nous avons pu réaliser, tant au niveau du site que des fonctionnalités que nous proposons. Cependant, nous n'avons pas réussi à totalement aller au bout de ce que nous voulions faire avec les fonctionnalités (exemple : la fonctionnalité graine du site).

7.2 Point de vue personnel : d'Arthur

Venant de DUT, j'avais auparavant eu l'occasion de créer des sites web. Seulement, c'est la première fois que j'utilisais Flask et Python pour faire tourner le site interne. Ce projet m'a alors permis d'apprendre un nouveau langage pour développer un site internet. D'un autre côté, j'ai pu réutiliser mes connaissances en développement web pour aider mes camarades à créer un site propre et bien organisé. J'ai donc pu accroître mes capacités à transmettre des connaissances.

7.3 Point de vue personnel : Jeremi

Travailler sur le projet était intéressant, j'ai pu comprendre et maîtriser les outils pour développer un site web, gérer des bases de données. Par ailleurs, ce projet était une opportunité pour apprendre à travailler en équipe (délai pour créer une fonctionnalité, nécessité d'adapter son code avec celui de ses camarades).

7.4 Point de vue personnel : Thomas

Ce projet m'a permis de découvrir et d'acquérir de l'expérience en tant que chef de projet (gérer les délais, les documents pour initialiser un projet et pour suivre un projet ...). Il m'a aussi permis de dévélopper mes compétences en web (découvrir l'envoi de mail via, approfondir l'utilisation du CSS...), en base de données (requête sql, création d'une base de données) et en algorithmie. J'ai pu également développer ma capacité à travailler en équipe grâce à ce projet (délai à respceter, prendre en considération l'avis de chacun ...).