



PPII - Projet Pluridisciplinaire d'Informatique Intégrative

Circuit-court et jardin partagé

Responsables du module :

Oliver FESTOR
Anne-Claire HEURTEL
Gérald OSTER

Équipe projet

Lucas ARIES
Paul DEVEAUX,
Théo HORNBERGER
Terry TEMPESTINI

Table des matières

1	Introduction	3
2	Descriptif de l'application	4
3	Gestion du répertoire git	6
4	Serveur web	7
4.1	Structure du serveur	7
4.2	Flask-Login	8
4.3	SQLAlchemy	8
4.4	Pytest	8
4.5	Wezkzeug	8
4.6	Visuel des pages Web	8
5	Base de données	10
5.1	Conception de la base	10
5.2	Utilisation de la base dans le projet	12
6	Algorithmes de traitement	13
6.1	Algorithme suggestion de panier	13
6.1.1	Division en lots	13
6.1.2	Tri des produits en fonction du prix	14
6.1.3	Création d'une liste avec répartition équitable	14
6.1.4	Remplissage du panier	15
6.2	Différents algorithmes utilisés	15
7	Tests et performance	16
7.1	Les tests	16
7.2	Performance	16
8	La gestion de projet	19
8.1	Les différents documents	19
8.2	La répartition des tâches	19
8.3	Rythme du travail	20
8.4	Outils utilisés	20
8.5	Difficultés rencontrées	20
9	Conclusion	21
9.1	Conclusion	21
9.2	Si c'était à refaire	21

10 Annexes	23
10.1 Compte rendu de réunion	24
10.2 Diagramme de Gantt	38

Chapitre 1

Introduction

Ce projet a démarré par une phase de brainstorming construit autour de plusieurs réunions. Notre groupe a décidé de réaliser le PPII en accordant un temps particulièrement conséquent en amont du développement afin de réaliser au mieux le projet et de nous permettre de rencontrer le moins d'accrocs et d'éviter le plus de problèmes en cours de route.

Pour ce faire nous avons essayé d'anticiper le mieux possible les besoins de notre programme et de sa conception, de sélectionner les différents éléments techniques les plus pertinents pour la faisabilité du programme. Suite à cela, nous avons divisé les éléments importants afin de pouvoir travailler le plus indépendamment des autres tâches en cours de développements. En plus de répartir le travail, cela a permis à chacun de pouvoir travailler à son rythme.

Finalement, nous avons réalisé des sessions de vérification de notre travail afin d'assurer la cohérence de l'ensemble du projet et de son fonctionnement.

Chapitre 2

Descriptif de l'application

Jardi'Quest sera une application qui a pour objectifs de rendre les jardins partagés ludique, comme un jeu et d'insister sur le mot "partagé" car actuellement les jardins partagés représentent plus souvent un terrain découpé en parcelles.

Notre application a pour but de faire des jardins un travail d'équipe, qu'il soit entretenu par tous avec tout le monde qui apporte sa contribution à celui-ci à l'aide de quêtes.



Il est important à la compréhension de ce document de recontextualiser notre programme et ses fonctionnalités.

Tout d'abord, notre application est composée de deux types d'utilisateurs différents : les utilisateurs et les propriétaires.

En fonction de leur statut, ils ont accès à différentes actions sur le site. Un utilisateur peut accéder à son jardin, au blog associé et aux différentes quêtes à réaliser dans celui-ci. En revanche, s'il n'a pas rejoint de jardin, il aura la possibilité d'en rejoindre un au choix ou bien de créer son propre jardin et en devenir le propriétaire.

Un propriétaire, quant à lui, a accès à des fonctionnalités supplémentaires. Il peut accéder à la gestion de son jardin dans laquelle il pourra effectuer plusieurs actions comme déposer des quêtes, cycliques ou non, à réaliser par les participants, ajouter des récoltes dans le marché du jardin ou encore modifier les informations de son jardin.

Tout l'enjeu de notre application tourne autour du système de quêtes et de l'économie qui en découle. Chaque jardin possède sa propre monnaie que l'utilisateur peut gagner en effectuant les quêtes proposées par le propriétaire. Ainsi chaque jardin possède sa propre économie : la monnaie d'un jardin ne peut pas être échangée avec celle d'un autre jardin. Cette monnaie est uniquement échangeable avec les récoltes de son jardin.

Le but de ce système est donc de proposer un moyen ludique permettant de rendre plus attractif le jardinage tout en récompensant les utilisateurs les plus actifs. Celui-ci offre donc un jardin partagé plus convivial tout en y apportant une certaine équité.


Chapitre 3

Gestion du répertoire git

Pour pouvoir travailler efficacement, nous avons décidé d'utiliser les différentes fonctionnalités git. Nous avons surtout utilisé le système de branches git afin de pouvoir programmer le projet en utilisant une branche distincte par bloc de fonctionnalités. Pour rendre plus facile d'utilisation le programme, un fichier de documentation (README.md) a été réalisé et ajouté au dépôt git avec les différentes commandes nécessaires au fonctionnement du projet.

Voici l'aperçu de celui-ci.

Description du projet



JardiQuest

Application Web qui permet de gérer un **jardin** de façon **innovante** à l'aide de **quêtes**. Plus besoin de séparer son jardin en parcelles, ici tout le monde travaille main dans la main pour être récompensé et recevoir les fruits de son labeur.

Installation du projet

Au root du projet veuillez effectuer ces commandes pour installer et lancer l'application

```
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
flask run
```

Lancement du projet

Après avoir installé le projet vous pouvez le lancer en étant au root du projet avec

```
source venv/bin/activate
flask run
```

Lancement des tests

Dans le root du projet :

```
source venv/bin/activate
pytest
```

Chapitre 4

Serveur web

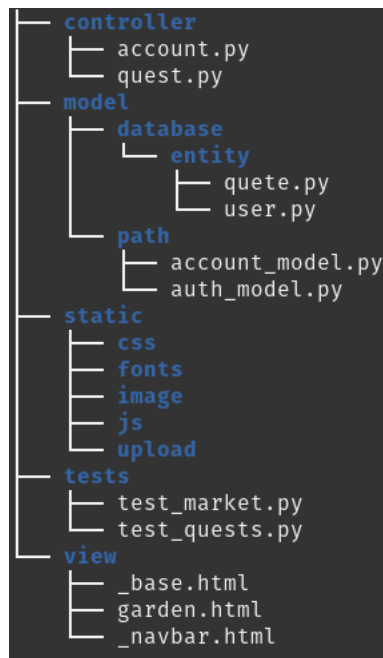
4.1 Structure du serveur

Le serveur web est le point central du projet. Ainsi, il est important que celui-ci soit correctement construit afin de faciliter à la fois son implémentation et les éventuelles mise à jour qu'il pourrait subir.

C'est ainsi que nous avons décidé d'utiliser une architecture MVC. Une architecture Model-View-Controller est un motif d'architecture logicielle qui consiste à distinguer trois entités distinctes étant le modèle, la vue et le contrôleur. Dans notre projet, les controllers sont les fonctions Python qui créent les URL du serveur et appelle les modèles à utiliser pour celles-ci, les modèles contiennent les différentes fonctionnalités et algorithmes utiles au serveur et appellent les différentes vues voulues. Les vues sont les différentes templates html qui correspondent donc aux différents affichages.

En plus de cette architecture particulière, nous avons décidé de séparer complètement le projet informatique et les autres documents qui ne sont pas utiles à son bon fonctionnement (exemple : documents de gestion de projet) en créant un répertoire au-dessus du projet. Cependant dans un souci de facilité d'accès, nous avons décidé de garder un fichier python à la racine afin de pouvoir lancer le programme avec la commande basique de flask ("flask run").

Voici un aperçu de l'arborescence du projet (version simplifiée)



Le serveur utilise plusieurs dépendances pour gérer différentes fonctionnalités.

4.2 Flask-Login

Pour le bon fonctionnement du serveur vis-à-vis des connexions nous avons décidé d'ajouter Flask-login à celui-ci. Flask-login est un supplément à Flask qui permet de gérer la connexion des utilisateurs de manière simplifiée. Cela permet de s'assurer de l'authentification de l'utilisateur pour certains chemins et de récupérer certaines informations.

4.3 SQLAlchemy

Avec un serveur Web, il y a toujours une base de données. Il est donc important de choisir la meilleure solution permettant la flexibilité et la sécurité nécessaire au bon fonctionnement du serveur. Ici, nous avons implémenté SQL à l'aide de Python. De nombreuses solutions sont possibles pour le faire. Il est possible de joindre directement une base de données à Python, cependant cela avait de nombreux inconvénients comme l'obligation d'écrire les requêtes SQL en entiereté.

De ce fait, nous nous sommes portés sur l'utilisation d'un ORM (Object Relational Mapper) qui permet de faciliter grandement la manipulation des données de la base. Notre choix s'est donc porté sur l'ORM SQLAlchemy qui semblait être le plus simple d'utilisation tout en nous permettant de réaliser toutes les fonctionnalités que nous avions prévu. De plus, SQLAlchemy possède un atout particulièrement utile : il va générer automatiquement la base de données au lancement du programme si celle-ci n'existe pas. Cela permet de lancer le projet informatique même sans avoir le schéma de la base de données car SQLAlchemy va directement le créer d'après sa configuration.

4.4 Pytest

Pour gérer les bugs et pouvoir tester notre serveur, il nous fallait un système de tests. Pytest permet d'écrire des tests en python pour vérifier l'absence de bugs. Les tests peuvent être lancés facilement à l'aide de la commande "pytest" et ainsi permettent d'assurer un bon niveau de qualité et de fonctionnement global du programme. Ainsi, son implémentation était logique et immédiate.

4.5 Wezkezeug

A propos de la sécurité du serveur, Wezkezeug permet de hasher les mots de passe lors de la sauvegarde de ceux-ci dans la base de données. Le hashage du mot de passe avant le stockage est une étape primordiale pour la sécurité des données et pour s'assurer de la sûreté des comptes utilisateurs et de leur mot de passe.

4.6 Visuel des pages Web

Pour la réalisation de l'aspect visuel de l'application, nous avons utilisé plusieurs ressources différentes. Tout d'abord, notre site web est en entiereté codé en CSS. Nous avons fait le choix de ne pas utiliser de moyen tiers comme Bootstrap. Ce choix nous limite grandement dans l'implémentation d'un beau visuel mais nous avons l'envie de concevoir notre site à 100%. Le CSS a été une part entière du

développement de notre application.

De plus, à l'aide de Flask et des templates html, nous avons pu rendre notre site plus agréable. Un des meilleurs exemples est la barre de navigation. Elle est séparée du reste des pages html et est implémentée dans chacune de celles-ci en y précisant une variable. Ainsi, elle reconnaît sur quels endroits du site elle se situe et modifie la couleur de l'endroit courant.

Enfin, nous avons rajouté du dynamisme au site en y implémentant du javascript. Celui-ci est utilisé en petite dose tout autour du projet, mais il est très important au niveau des barres de navigation et de recherche. En effet, l'ajout de javascript à celles-ci rajoutent du dynamisme et de la modernité aux pages Web.

Chapitre 5

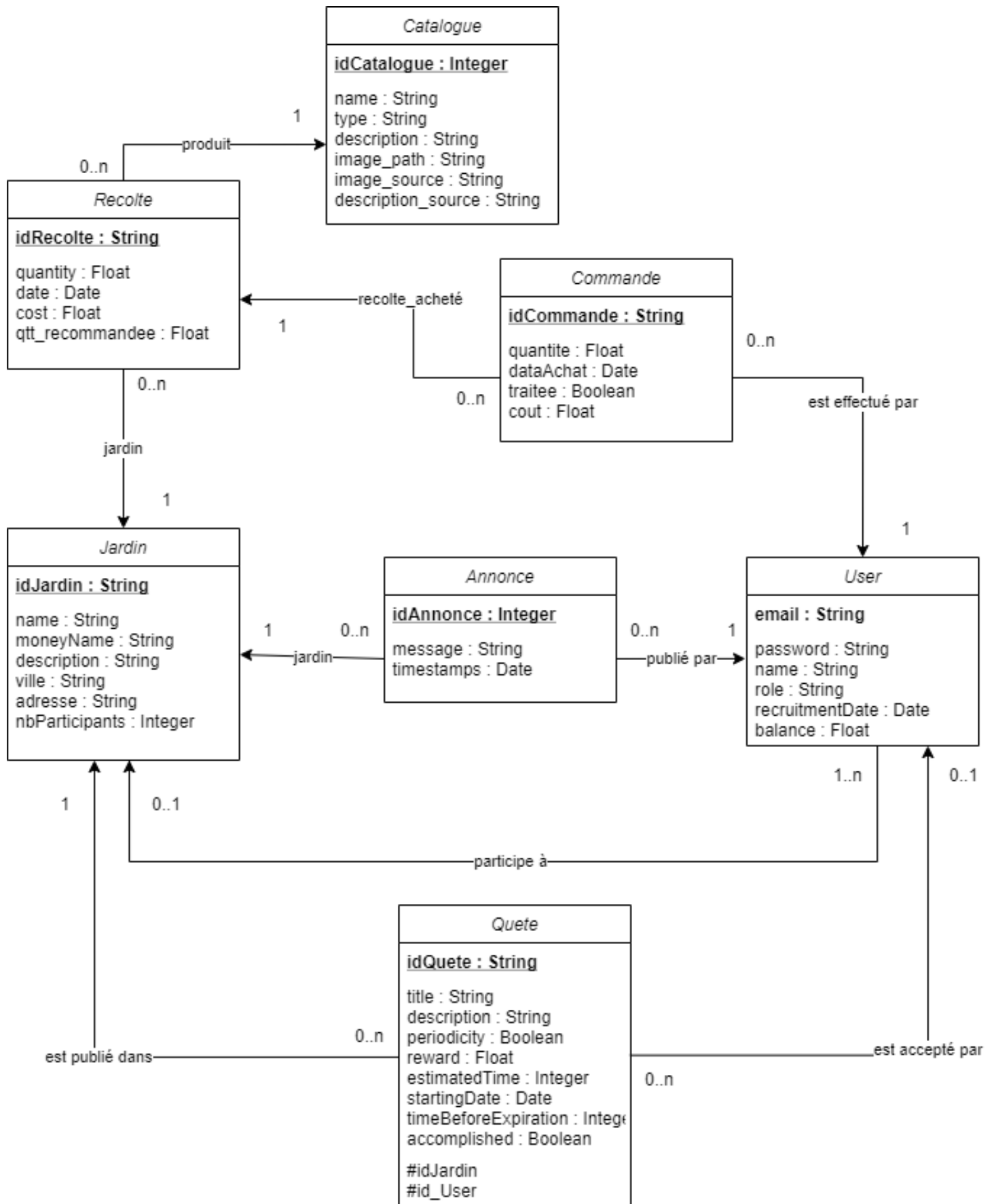
Base de données

5.1 Conception de la base

La conception initiale de la base de données est très proche de la base finale réalisée dans l'application. Cependant, des détails techniques, des optimisations ou encore des ajouts de fonctionnalités nous ont poussé à apporter quelques modifications dans la base. On peut citer par exemple, l'ajout de l'adresse et de la ville d'appartenance d'un jardin ou encore pour l'utilisateur, le passage d'un identifiant unique à son email. Voici donc la conception actuelle :

- Un utilisateur est représenté par son adresse email. Il possède un nom, un mot de passe, un rôle (gérant, participant, ou rien), une date de recrutement et un solde. Il peut s'associer à un jardin.
- Un jardin est représenté par un identifiant. Il possède un nom, un nom de monnaie, une description, une ville d'appartenance, une adresse et un nombre de participants.
- Une quête est représentée par son identifiant. Elle possède un titre, une description, un booléen qui indique si la quête est périodique et doit donc se répéter après avoir été effectué, une récompense, une date de début, un entier correspondant au temps en jour avant que la quête expire à partir de sa date de début, et un booléen qui indique si elle a été accomplie. Elle appartient forcément à un jardin, et elle peut ou non avoir été réalisée par un utilisateur.
- Un catalogue est représenté par son identifiant. Il possède un nom, un type (légume, fruit, herbe aromatique), une description, un chemin vers son image, la source de l'image et la source de sa description.
- Une récolte est représentée par un identifiant. Elle est vendue à un coût/kg, a une certaine quantité et a été récoltée à une certaine date. Elle correspond à un produit du catalogue, et a été récolté dans un jardin. Elle dispose également d'une quantité recommandée par produit.
- Une commande est représentée par un identifiant. Elle possède une certaine quantité et un certain coût, une date d'achat et un booléen pour savoir si elle a été traitée ou non. Elle est effectuée par un utilisateur sur une récolte.
- Une annonce est représentée par son identifiant. Elle possède un message et une date d'envoi. Cette annonce est effectuée par un utilisateur dans un jardin.

Voici le schéma de celle-ci



5.2 Utilisation de la base dans le projet

Comme expliqué précédemment, nous avons utilisé SQLAlchemy pour construire notre base de données.

Les fichiers permettant cela sont stockés dans un dossier propre : `jardquest/model/database/entity`. Chaque fichier correspond à une table de la base de données avec ses relations par rapport aux autres tables.

Pour implémenter le catalogue de produits, au lancement de l'application celui-ci va être automatiquement créé s'il n'existe pas déjà dans la base de données. Cela est effectué en important les données d'un fichier `.csv` qui a été rempli au début du projet. Ces données sont plutôt exhaustives et permettent donc aux utilisateurs lambdas d'avoir un catalogue à disposition directement. De plus, la liste pourrait être facilement étoffée au besoin.

Afin de gérer l'expiration des quêtes dans le temps, nous avons utilisé `flask_apscheduler`. Cette extension de flask va effectuer un test tous les jours sur l'entièreté des quêtes afin de savoir lesquelles sont accomplies. Si la quête est périodique, alors une nouvelle quête ayant les mêmes caractéristiques est créée. Ainsi, si le jardin doit être arrosé tous les jours, une quête pour réaliser cette tâche n'a pas besoin d'être recrée à chaque fois. Si elle n'est pas périodique, la quête est tout simplement supprimée.

Un autre point pour lequel nous nous servons de la base de données est le système d'upload d'images. Ce système permet aux utilisateurs propriétaires de montrer à quoi ressemble leur jardin. Cela a été implémenté avec toutes les sécurités classiques d'upload de fichiers, une taille limite à la requête, un nombre spécifique d'extensions de fichier accepté et un renommage du nom des fichiers.

Lorsque l'image est uploadée, celle-ci est renommée par l'identifiant du jardin. Cela sert à éviter les potentiels noms dangereux des fichiers. Mais également à assurer l'unicité de chaque nom d'image pour chaque jardin et donc ne pas consommer de l'espace de stockage inutile en ayant plusieurs images par jardin.

Les images sont stockées dans le `static/upload`. Elles se situent dans le module static car celles-ci doivent être accessibles à tous les utilisateurs et cela permet de pouvoir vérifier si le fichier existe rapidement et afficher une image par défaut le cas échéant.

Chapitre 6

Algorithmes de traitement

6.1 Algorithme suggestion de panier

En ce qui concerne les algorithmes de traitement avancés, nous avons choisi de développer un système de suggestion de panier. Le but étant de proposer un assortiment de produits que le client pourra acheter. Cet algorithme cherche à optimiser la diversité des produits tout en minimisant le coût. Pour le réaliser nous avons étudié deux approches différentes :

- la première avec un algorithme type sac à dos (en listant les solutions de manière exhaustive et en sélectionnant la meilleure), cette approche a été abandonnée du fait de sa complexité qui ne permettait de donner une liste dans un temps raisonnable pour seulement une vingtaine de lots au maximum. De plus, la solution optimale n'apportait pas assez d'avantages par rapport à une solution approchée.
- la seconde avec un algorithme glouton (en cherchant une solution acceptable la plus proche possible de la solution optimale), cette approche a été choisie du fait de sa complexité plus acceptable et de son résultat très proche de la solution. Celui-ci contient trois étapes que nous allons détailler :
 - Division des récoltes en lots
 - Tri des lots selon leur prix
 - Tri pour répartition équitable
 - Remplissage du panier

6.1.1 Division en lots

Dans un premier temps, nous avons divisé les récoltes des différents produits en lots (en fonction de leur quantité recommandée) afin d'obtenir une liste de lots de légumes correspondant à une récolte et un produit en particulier.

A partir de cette liste et de l'argent que dispose l'utilisateur, le but est de trouver le meilleur panier, c'est-à-dire celui qui maximise le nombre de lots en minimisant le prix des produits.

Pour se représenter le problème voici un exemple :

N°lot	1	2	3	4	5	6	...
Produit	Carrote id=6	Carotte id=6	Haricot id=1	Tomate id=5	Brocolis id=7	Haricot id=1	...
Prix	0.5€	0.5€	1.2€	0.7€	3€	0.9€	...
Quantité	400g	400g	200g	150g	500g	200g	...
Récolte	rec ₁	rec ₁	rec ₂	rec ₃	rec ₅	rec ₄	..

Nous pouvons voir ici, qu'à choisir, il est préférable de prendre le lot 6 que le lot 3 (prise en compte du produit le moins cher).

L'algorithme trouvant ce résultat a une complexité en $O(n)$ car on parcourt la liste des récoltes et pour chaque élément on crée un nombre fini de lots (cela ne dépendant pas de n) : complexité = $a \cdot O(n) + b \cdot O(n) + c \cdot O(n) + \dots$ (avec a, b, c, \dots des constantes)

6.1.2 Tri des produits en fonction du prix

Nous avons utilisé un tri à bulles pour arranger les produits en fonction de leur prix. De cette manière, les lots les moins coûteux se retrouvent en début de liste.

Pour l'exemple nous obtenons la liste de lots suivante (les chiffres sont les id des légumes) :
6,6,6,6,6,5,5,5,5,1,2,2,2,3,3,1,1,4,4,7

La complexité du tri à bulles est en $O(n^2)$ dans le pire des cas (le plus petit élément est à la fin du tableau).

6.1.3 Création d'une liste avec répartition équitable

Nous avons créé une fonction récursive qui transforme la liste triée en liste répartissant équitablement les légumes. Le premier légume le moins cher est pris une seule fois par cycle :

6,6,6,6,6,5,5,5,5,1,2,2,2,3,3,1*,1,4,4,7 → **6,5,1,2,3,4,7** → **appel n°1**
6,6,6,6, 5,5,5, 2,2, 3,1,1, 4, → **6,5,2,3,1,4** → **appel n°2**
6,6,6, 5,5, 2, 1, → **6,5,2,1** → **appel n°3**
6,6, 5, → **6,5** → **appel n°4**
6, → **6** → **appel n°5**
 *le légume 1 à déjà été pris (on ne le reprend pas pour ce cycle)

Liste finale obtenue : 6,5,1,2,3,4,7,6,5,2,3,1,4,6,5,2,1,6,5,6

Cette liste obtenue répartit équitablement les différents produits et avec un prix croissant. De cette manière, un panier intéressant est un panier comportant un maximum de ces produits.

La complexité est en $O(n^2)$ car dans le pire des cas : Liste de départ qui ne contient que des mêmes éléments on aura n appels de complexité n (car parcours de la liste entière).

6.1.4 Remplissage du panier

L'étape finale est le remplissage du panier. En fonction de l'argent que dispose le client, on prend un maximum des ces légumes pris dans l'ordre donné pour notre fonction.

Un panier possible serait alors : **6,5,1,2,3,4,7,6,5,2,3** ou **6,5,1,2**

Complexité en $O(n)$ car on parcourt la liste finale jusqu'à que le solde de l'utilisateur soit atteint (pire des cas parcourt de la liste entière : $O(n)$)

Pour conclure cette fonction est en complexité $O(n^2)$ car les quatre étapes qui se succèdent sont de complexités respectives : $O(n) + O(n^2) + O(n^2) + O(n)$

6.2 Différents algorithmes utilisés

Au cours de ce projet, nous avons mis en place des fonctionnalités de filtrage et de tri de listes. Nous avons utilisé des expressions régulières pour permettre à l'utilisateur de filtrer les éléments de la liste en fonction de différents critères.

Nous avons également ajouté une fonction de tri qui permet à l'utilisateur de choisir l'argument de tri (par exemple, ordre alphabétique ou ordre de prix croissant).

En outre, nous avons implémenté des algorithmes de vérification des données entrées par l'utilisateur lors de la connexion (login et mot de passe).

Ces algorithmes visent à s'assurer que les données sont correctes et sécurisées, ce qui est essentiel pour protéger les informations de l'utilisateur et garantir la confidentialité de ses données.

Grâce à ces fonctionnalités de filtrage, de tri et de vérification des données, nous avons permis à l'utilisateur de manipuler et de travailler de manière efficace avec les données présentes dans la liste, tout en garantissant la sécurité des informations qu'il a entrées.

Chapitre 7

Tests et performance

7.1 Les tests

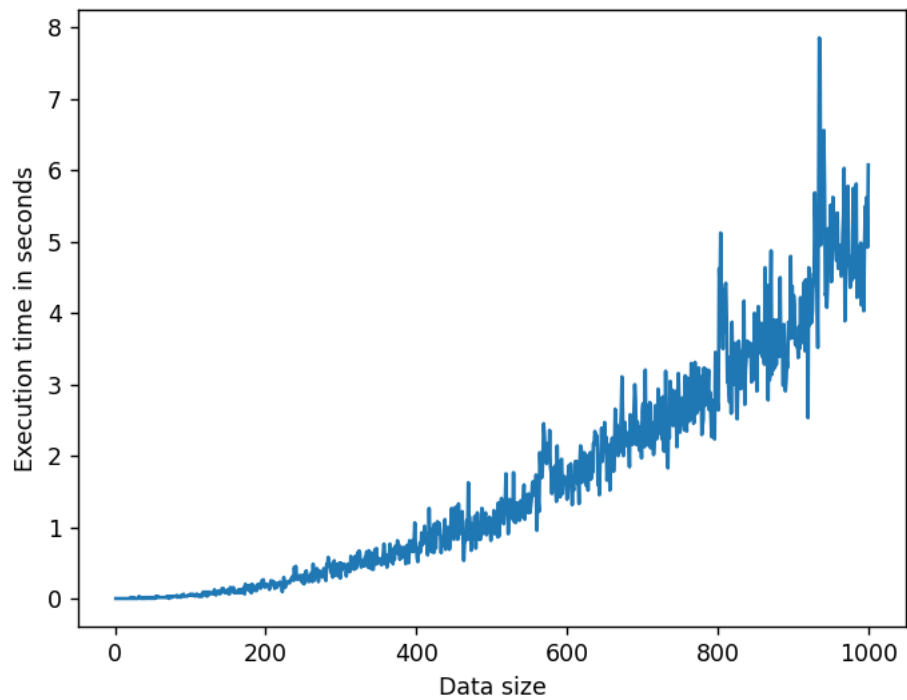
Les tests unitaires sont importants afin de pouvoir s'assurer du bon fonctionnement de l'application tout au long du projet.

Ils ont ici été réalisés grâce à `pytest-flask`, un plugin du framework de test `pytest`. Une autre base de données a été créée afin de pouvoir simuler l'application sans modifier la base de données originale. Un dataset de test y est également importé. Pour chaque route, nous testons si le code de réponse http est le bon. S'il doit rediriger quelque part et si la redirection est la bonne. Les routes sont souvent testées avec plusieurs cas distinct, comme par exemple lorsque l'utilisateur n'est pas connecté, lorsque l'utilisateur essaye d'accéder à une ressource qui ne fait pas parti de son jardin, ...

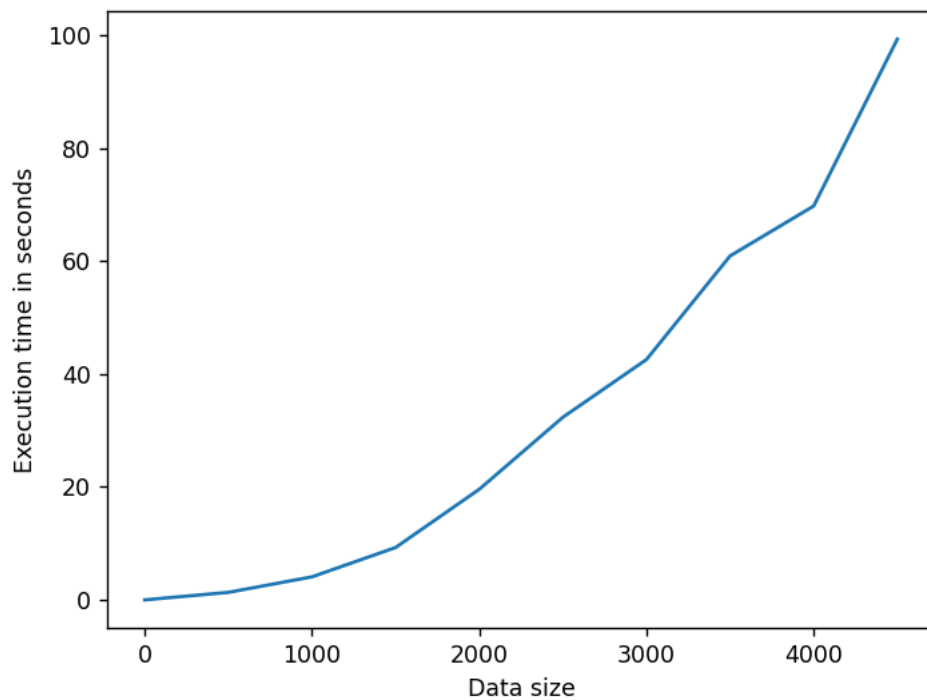
7.2 Performance

Afin de créer des graphiques de performance, nous avons utilisé la bibliothèque `matplotlib`.

Voici le graphique correspondant au temps d'exécution de l'algorithme de suggestion, pour des données de taille allant de 1 à 1000 :



La plage de données étant trop faible, nous avons refait un test avec un échantillon plus petit mais une plus grande plage :



Il est difficile de tester avec des valeurs plus grandes à cause du temps d'exécution.

Ainsi on peut voir que les données du graphes sont cohérentes avec la complexité théorique mentionnée précédemment dans la section dédiée à l'algorithme, soit $O(n^2)$.

Il est donc difficilement utilisable sur un grand nombre de données, mais le nombre de récoltes d'un même jardin ne devrait pas être énorme, sachant que les produits peuvent périmer et donc ne resteront pas en stock indéfiniment. L'algorithme reste donc utilisable et efficace pour la tâche demandée.

Chapitre 8

La gestion de projet

8.1 Les différents documents

Nous avons élaboré de nombreux documents tout au long du projet, principalement dans la phase de conception, pour nous guider dans la création de ce projet. Les documents présents dans le répertoire du projet sont les suivants :

- Un diagramme de Gantt qui a permis de gérer l’avancement du projet et d’ainsi nous permettre de nous rendre compte de l’avancée du projet et d’éviter de prendre du retard. (cf Annexe)
- Les compte rendus des différentes réunions pour planifier le projet et le concevoir. (cf Annexe)

On peut aussi noter les conversations plus informelles que nous avons pu avoir que ce soit sur un serveur discord ou dans les couloirs de l’école, qui nous ont permis de discuter d’un ensemble de points d’interrogation ou de problèmes rencontrés sur le projet.

8.2 La répartition des tâches

Pour la répartition des tâches, nous avons décidé de séparer le travail par rapport aux grandes fonctionnalités du site web. Une fois la conception du projet réalisée avec tous les membres du groupe, les différentes parties ont été attribuées selon les envies et préférences de chacun.

ARIES Lucas : création du serveur, environnement des dépendances, architecture, authentification, page de connexion et de profil, création de tests, écriture du rapport, préparation de la soutenance.

DEVAUX Paul : dataset de fruits et légumes, création du marché et de la gestion des quêtes, création de tests, écriture du rapport, préparation de la soutenance.

HORNBERGER Théo : implémentation de la base de données, page de gestion des jardins et du blog, création de tests, écriture du rapport, préparation de la soutenance.

TEMPESTINI Terry : page du jardin, maquettage du site, algorithme de suggestion de panier, création de tests, écriture du rapport, préparation de la soutenance.

A noter aussi, l’implication plus conséquente de Lucas. En s’étant proposé en tant que chef de projet, il a eu une part plus importante dans la gestion du projet tant dans la gestion de l’avancement du projet que dans la rédaction des comptes rendus de réunion.

8.3 Rythme du travail

Bien que nous avons chacun un rythme différent de travail, des échéances étaient programmées pour lesquelles chaque fonctionnalité requise devait être fusionnée sur la branche principale du projet afin d’être testée, vérifiée, et possiblement améliorée à travers des remarques ou idées de chaque membre.

Une même fonctionnalité était donc souvent fusionnée sur la branche principale, pour être vérifiée et améliorée plusieurs fois avant d’atteindre sa version finale. Ce rythme de travail peut être associé à la méthode PDCA bien que les dates limites généralement fixes ne l’étaient pas ici, étant donné les contraintes de disponibilités différentes pour chacun des membres du groupe pour le projet, comme le projet est sur le temps libre de chacun.

8.4 Outils utilisés

Nous avons utilisé plusieurs outils pour la réalisation de la gestion de projet.

Tout d’abord, pour la partie graphique de l’application, nous avons utilisé Figma pour faire un maquettage du site web et en tirer une charte graphique. Cela nous a permis de créer une identité propre au site en amont du développement. De la même façon, nous avons utilisé Canva pour créer le logo de l’application.

Ensuite, pour le diagramme de Gantt, nous avons utilisé Gantt Project. Pour la communication du groupe, nous avons utilisé principalement Discord qui nous a permis de communiquer sur l’avancement du projet et de débattre en équipe.

Enfin pour la rédaction des différents documents nous avons utilisé les outils Google : Google Document et Google Présentations. Ces documents nous permettant de travailler en simultané sont la solution la plus simple. De la même manière, nous avons utilisé Overleaf pour la rédaction en LaTeX

8.5 Difficultés rencontrées

Dans la globalité du projet, nous avons rencontré peu de difficultés. Notre conception et notre expérience nous ont permis d’éviter de nombreux pièges ou problèmes.

Cependant, on peut noter la difficulté principale étant le temps. En effet, bien que la période consacrée au projet soit conséquente, notre longue conception nous a fait perdre un temps précieux. De plus, la partie de développement se situait en grande partie pendant la période d’examens et de vacances de fin d’année.

Ainsi, le temps consacré au projet se trouvait parfois grandement réduit par les temps de révisions ou les fêtes de fin d’année.

Nous avons aussi rencontré d’autres problèmes mineurs. On peut citer notamment la gestion des tests et de leur impact sur la base de données. Nous avons dû réfléchir et implémenter plusieurs solutions, modifier la base de données du serveur en ‘production’, ou alors utiliser (une base de données simulée qui reproduit le comportement de notre base originelle.

Chapitre 9

Conclusion

9.1 Conclusion

Ce projet permet de réaliser du développement web et de réellement voir de quoi il s'agit, ce n'est pas un simple html/css mais bien tout un serveur avec un back-end, un front-end, une base de données et les différentes sécurités à mettre en place.

Le PPII est un projet multidisciplinaire qui nous permet réellement de nous rendre compte de la difficulté du développement web et de la gestion de projet. Le développement web ne se limite pas à des pages simples sans dynamismes mais nous pousse à devoir créer un véritable environnement avec un serveur, sa base de données, les différentes pages et les différentes sécurités à mettre en place. De plus il nous permet de nous montrer l'importance de la gestion de projet lors d'un développement informatique en groupe. Cela peut devenir vite très ardu de ne pas se perdre et de s'assurer que l'ensemble du groupe avance dans la même direction.

Étant donné que tous les membres de ce groupe proviennent de DUT, nous avons déjà réalisés des projets informatiques (notamment en web). Nous avons déjà pu constater ces différentes difficultés, cependant nous avons l'impression générale que de la part de la gestion de projet dans le PPII est plus importante et nous permet de mieux intégrer notre futur rôle d'ingénieur. C'est grâce à ce contexte et cet environnement que nous avons eu une réelle volonté de faire les choses bien et de façon professionnelle.

De plus, ce projet nous a plongé dans un environnement web différent à travers Python et flask. Aucun de nous n'avait utilisé cet environnement, ce qui nous permet d'engranger une nouvelle expérience informatique.

Enfin, nous avons eu un réel intérêt pour le projet grâce à son ouverture et la liberté que nous proposait le sujet initial. Il nous a permis de laisser place à notre créativité, ou à la recherche d'innovation. Cependant, l'absence d'encadrement surtout pour la partie développement aurait pu nous rendre la tâche bien plus difficile et nous aurions pu acquérir de mauvaises méthodes de programmation si nous n'avions pas eu une certaine expérience.

9.2 Si c'était à refaire

Si le projet était à refaire, nous passerions cependant moins de temps sur la partie de conception. Bien qu'elle nous ait été très utile, nous aurions pu, à l'aide de notre expérience, éviter tout aussi bien les problèmes. Les examens finaux sont chronophages et cela a été un point difficile à gérer lors de la

création du projet. Nous aurions dû nous faire un peu plus confiance et utiliser à bon escient notre expérience passée.

Un dernier point à souligner serait notre non-utilisation d'un framework comme Bootstrap pour le visuel de l'application. Nous aurions pu avancer bien plus vite et ainsi accentuer nos efforts sur la partie algorithmique ou sur le nombre de fonctionnalités de l'application. Cependant, nous avons très vite remarqué qu'aucun d'entre nous possédait un attrait sur le développement frontend. Dans un esprit d'apprentissage, nous avons jugé qu'il était alors préférable de développer ce point faible et donc de programmer l'entièreté du visuel par nous-même. Nous préférons avoir de vraies bases solides dans ce domaine que de développer plus fonctionnalités dans un domaine qui nous est bien plus familier.

Chapitre 10

Annexes

10.1 Compte rendu de réunion

Projet PPII - Compte rendu réunion N°1

Motif / Type de réunion : Préparation du projet PPII Découverte des consignes	Lieu : Telecom Nancy
Présent(s)retard/excusés : TEMPESTINI Terry DEVEAUX Paul HORNBERGER Théo ARIES Lucas	Date / Heure de début / Durée : 13/10/22 début 13h20 fin 14h40

Ordre du jour :

- Repérer les différentes tâches à réaliser
- Réfléchir à une idée d'application
- Organiser la répartition du travail

Todo List :

Description	Responsable	Délai	Livrable	Validé par
Faire l'analyse de l'existant	x	1 jour	x	x
Réfléchir à une idée d'application web	x	1 jour	x	x

Projet PPII - Compte rendu réunion N°2

Motif / Type de réunion : Finaliser l'idée, répartir tâches gestion de projet	Lieu : Telecom Nancy
Présent(s)retard/excusés : TEMPESTINI Terry DEVEAUX Paul HORNBERGER Théo ARIES Lucas	Date / Heure de début / Durée : 14/10/22 début 13h fin 14h

Ordre du jour :

- Définir clairement l'idée d'application
- Répartir les différentes tâches

Todo List :

Description	Responsable	Délai	Livrable	Validé par
État de l'art	Terry, Théo	3 jours	Document (Analyse de l'existant)	x
Gestion de projet	Paul, Lucas	3 jours	SWOT, charte de projet, todo-list	x

Projet PPII - Compte rendu réunion N°3

Motif / Type de réunion : Visualiser l'avancement du projet et re-diriger le travail si nécessaire.	Lieu : Telecom Nancy
Présent(s)retard/excusés : TEMPESTINI Terry DEVEAUX Paul HORNBERGER Théo ARIES Lucas	Date / Heure de début / Durée : 17/10/22 début 18h fin 18h25

Ordre du jour :

- Notifier l'avancement des tâches
- Organiser la prochaine réunion

Todo List :

Description	Responsable	Délai	Livrable	Validé par
Terminer les différents documents	Équipe projet	2 jours	Charte projet et état de l'art	Équipe projet
Mettre en forme le rapport final	Équipe projet	1 jours	Rapport final dû au jeudi	Équipe projet

Projet PPII - Compte rendu réunion N°4

Motif / Type de réunion : Préparation orale + Création Diaporama	Lieu : Telecom Nancy
Présent(s)retard/excusés : TEMPESTINI Terry DEVEAUX Paul HORNBERGER Théo ARIES Lucas	Date / Heure de début / Durée : 20/10/22 début 13h40 fin 14h10h

Ordre du jour :

- Vérifier le rapport final
- Réaliser le diaporama
- Préparer la soutenance

Todo List :

Description	Responsable	Délai	Livrable	Validé par
Terminer le diaporama	x	1 jour	x	x
Préparer la soutenance	x	1 jour	x	x

Projet PPII - Compte rendu réunion N°5

Motif / Type de réunion : Préparation du développement	Lieu : Telecom Nancy
Présent(s)retard/excusés : TEMPESTINI Terry DEVEAUX Paul HORNBERGER Théo ARIES Lucas	Date / Heure de début / Durée : 10/11/22 début 13h40 fin 15h

Ordre du jour :

- Préparer la base de données (Schéma entité relation)
- Répartir les principales tâches du développement

Todo List :

Description	Responsable	Délai	Livrable	Validé
Réaliser base de données (SQL)	HORNBERGER Théo	10jours	Code SQL	Groupe
Faire Token oauth	ARIES Lucas	10jours	Code Python	Groupe
Maquettage	TEMPESTINI Terry, DEVEAUX Paul	10jours	Maquette FIGMA	Groupe
Récupérer dataset légumes	DEVEAUX Paul	10jours	Données formaté	Groupe

Projet PPII - Compte rendu réunion N°6

Motif / Type de réunion : Répartition du développement	Lieu : à distance
Présent(s)retard/excusés : TEMPESTINI Terry DEVEAUX Paul HORNBERGER Théo ARIES Lucas	Date / Heure de début / Durée : 24/11/22 début 21h05 fin 21h54

Ordre du jour :

- Vérification de la maquette figma et des différents travaux réalisés
- Répartitions des tâches

Todo List :

Description	Responsable	Délai	Livrable	Validé par
authentification barre de navigation, affichage du profil	Lucas	22 jours	code	Lucas
développement onglets quêtes et marché	Paul	22 jours	code	Lucas
développement onglets jardin (+ fonctionnalités affichage recherche)	Terry	22 jours	code	Lucas
développement onglets blog et gérer son jardin	Théo	22 jours	code	Lucas

Projet PPII - Compte rendu réunion N°7

Motif / Type de réunion : Stand-Up meeting sur discord	Lieu : à distance (discord)
Présent(s)retard/excusés : TEMPESTINI Terry DEVEAUX Paul HORNBERGER Théo ARIES Lucas	Date / Heure de début / Durée : 17/12/22 début 14h43 fin 14h56

Ordre du jour :

- Etat des lieux sur l'avancement de chacun

Projet PPII - Compte rendu réunion N°8

Motif / Type de réunion : Vérification du travail effectué	Lieu : À distance (discord)
Présent(s)retard/excusés : TEMPESTINI Terry DEVEAUX Paul HORNBERGER Théo excusé (à l'étranger) ARIES Lucas	Date / Heure de début / Durée : 20/12/22 début 21h55 fin 22h22

Ordre du jour :

- Vérification des pages jardins et discussion d'amélioration
- Avancement des autres pages

Todo List :

Description	Responsable	Délai	Livrable	Validé par
diverses améliorations à réaliser + bug à corriger	Terry	xxx	Code informatique	Lucas

Projet PPII - Compte rendu réunion N°9

Motif / Type de réunion : Discussion d'avancement et d'aide	Lieu : À distance (discord)
Présent(s)retard/excusés : TEMPESTINI Terry DEVEAUX Paul HORNBERGER Théo excusé (à l'étranger) ARIES Lucas	Date / Heure de début / Durée : 22/12/22 début 16h30 fin 18h

Ordre du jour :

- Discussion procédure appelée tous les jours
- Développement de l'algorithme
- Rapport

Todo List :

Description	Responsable	Délai	Livrable	Validé par
Fonction CRON	Paul	xxx	Code informatique	Lucas
Développement de l'algorithme	Terry	xxx	Code informatique	Lucas
Préparation du rapport	Paul, Lucas	xxx	Google doc	Lucas

Projet PPII - Compte rendu réunion N°10

Motif / Type de réunion : Stand up meeting	Lieu : À distance (discord)
Présent(s)retard/excusés : TEMPESTINI Terry DEVEAUX Paul HORNBERGER Théo excusé (à l'étranger) ARIES Lucas	Date / Heure de début / Durée : 27/12/22 début 17h07 fin 17h14

Ordre du jour :

- Avancement du développement

Projet PPII - Compte rendu réunion N°11

Motif / Type de réunion : Discussion d'avancement et d'aide	Lieu : À distance (discord)
Présent(s)retard/excusés : TEMPESTINI Terry DEVEAUX Paul HORNBERGER Théo excusé (à l'étranger) ARIES Lucas	Date / Heure de début / Durée : 28/12/22 début 14h50 fin 15h14

Ordre du jour :

- Rapport
- Algorithme

Todo List :

Description	Responsable	Délai	Livrable	Validé par
Répartition de la rédaction du rapport	Projet	xxx	Code informatique	Projet

Projet PPII - Compte rendu réunion N°12

Motif / Type de réunion : Vérification et amélioration du projet	Lieu : À distance (discord)
Présent(s)retard/excusés : TEMPESTINI Terry DEVEAUX Paul HORNBERGER Théo ARIES Lucas	Date / Heure de début / Durée : 01/01/23 début 13h08 fin 15h14

Ordre du jour :

- Vérification de l'avancement du projet
- Amélioration du projet

Todo List :

Description	Responsable	Délai	Livrable	Validé par
Design de la page des quêtes	Paul	xxx	Code informatique	Lucas
Design de la page jardin	Terry	xxx	Code informatique	Lucas
Design de la page de gestion de jardin + bug	Théo	xxx	Code informatique	Lucas
Harmonisation du code	Lucas	xxx	Code informatique	Lucas

Projet PPII - Compte rendu réunion N°13

Motif / Type de réunion : Résolution de problème	Lieu : À distance (discord)
Présent(s)retard/excusés : TEMPESTINI Terry DEVEAUX Paul HORNBERGER Théo ARIES Lucas	Date / Heure de début / Durée : 03/01/23 début 22h12 fin 22h23

Ordre du jour :

- Résoudre le problème de remplissage du catalogue

Projet PPII - Compte rendu réunion N°14

Motif / Type de réunion : Correction bugs et amélioration	Lieu : À distance (discord)
Présent(s)retard/excusés : TEMPESTINI Terry DEVEAUX Paul HORNBERGER Théo ARIES Lucas	Date / Heure de début / Durée : 05/01/23 début 17h54 fin 18h23

Ordre du jour :

- Vérification du code fait et manquant
- Idée d'amélioration

10.2 Diagramme de Gantt

