Rapport final de Projet Pluridisciplinaire d'Informatique Intégrative

Les jardins partagés

Alexis MARCEL
Lucas LAURENT
Noé STEINER
Mathias AURAND-AUGIER

Responsable du module :
Olivier FESTOR
Anne-Claire HEURTEL
Gerald OSTER

6 Janvier 2023

Contents

1 Base de donnée **2**
- 1.1 Conception . . . 2
  - 1.1.1 Les besoins de notre application . . . 2
  - 1.1.2 Schéma entité-association . . . 2
  - 1.1.3 Passage du modèle au entité-association au relationnel . . . 3
- 1.2 Implémentation de la base de donnée dans le backend de l'application . . . 4
  - 1.2.1 Création de la base de données . . . 4
  - 1.2.2 Utilisation de SQLAlchemy . . . 4

2 Serveur et client Web **5**
- 2.1 structure de l'application . . . 5
- 2.2 Fonctionnalités de l'application . . . 5
  - 2.2.1 Authentification . . . 5
  - 2.2.2 Gestion de compte . . . 5
  - 2.2.3 Gestion des jardins . . . 5
  - 2.2.4 Carte interractive . . . 5
  - 2.2.5 Interractions avec le jardin . . . 5

3 Algorithme **7**
- 3.1 Principe de l'algorithme . . . 7
- 3.2 Implémentation . . . 7
  - 3.2.1 Modélisation du problème . . . 7
  - 3.2.2 Modélisation du problème en un système d'équations linéaires . . . 8
  - 3.2.3 Optimisation du système d'équations linéaires . . . 9
- 3.3 Analyse en complexité . . . 10
- 3.4 Analyse de performance . . . 10

4 Gestion de projet **11**
- 4.1 Équipe de projet . . . 11
- 4.2 Organisation au sein de l'équipe projet . . . 11
- 4.3 Objectifs SMART . . . 12
- 4.4 Matrice des objectifs . . . 12
- 4.5 Triangle qualité-cout-délai . . . 12
- 4.6 Matrice SWOT . . . 13
- 4.7 Profil de projet . . . 14
- 4.8 WBS : comment concrétiser l'application . . . 14
- 4.9 Diagramme de Gantt : planification . . . 15
- 4.10 Matrice RACI . . . 15
- 4.11 Gestion des risques . . . 16

5 Conclusion **16**

6 Annexes **17**

1 Base de donnée

1.1 Conception

1.1.1 Les besoins de notre application

Pour commencer, notre application permet à des utilisateurs de s'enregistrer sur la plateforme. Une compte utilisateur est composé d'un email, un pseudo, son prénom, son nom, ainsi que la date de la création de son compte. Ensuite un utilisateur peut créer un jardin et ce jardin peut être rejoint par d'autres utilisateurs. Un jardin est constitué d'un nom, un propriétaire, un type (public ou privé) et enfin une adresse. Le propriétaire du jardin peut ensuite créer des parcelles dans son jardin. Une parcelle est composée d'un nom et d'une plante. Une plante à un nom et un besoin en eau. Les parcelles sont elle même consituées d'unités qui réprésentent un espace réel dans le jardin. Une unité est caractérisé par un index déterminant sa position dans le jardin. De plus, chaque parcelle peuvent être associé à une liste de tâche à effectuer. Une tâche est caractérisé par une date limite, un nom, une description, un état (à faire, en cours, terminé), une date limite et enfin une personne qui s'en occupe.

1.1.2 Schéma entité-association

Ces besoins donnent lieu à la création de plusieurs entités consituant le schéma entité-association suivant :

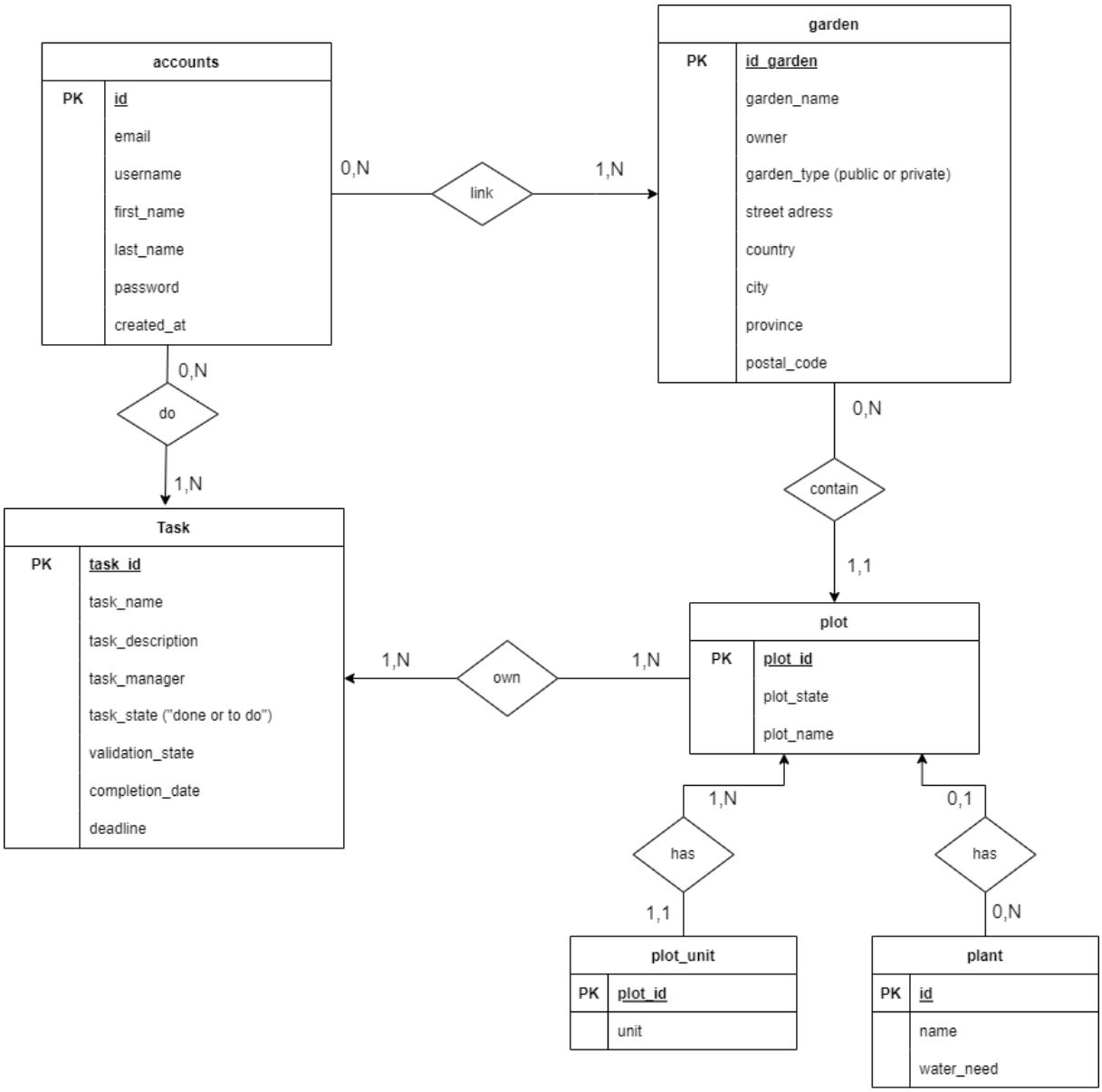

Figure 1: Schéma entité-association

Ce schéma respecte les contraintes logiques de cardinalités suivantes :

- Une personnes peut posséder/rejoindre un ou plusieurs jardins.
- Un jardin peut avoir plusieurs parcelles mais une parcelle est associée à un seul jardin.
- Une parcelle peut avoir plusieurs unités mais une unité est associée à une seule parcelle.
- Une parcelle peut avoir une plante et une plante peut être attribué à plusieurs parcelles.
- Une personne peut effectuer plusieurs tâches et une tâche peut être effectuée par plusieurs personnes.

1.1.3 Passage du modèle au entité-association au relationnel

A présent, nous allons transformer notre schéma entité-association en modèle relationnel en respectant les règles de la troisième forme normale.

- account(id, email, username, first name, last name, password, created at)
- garden(id_garden, garden_name, owner, manager, garden_type, street_adress, country, city, province, postal_code)

- plot(plot_id, garden_id, plot_state, plot_name, plant)
- task(task_id, plot_id, task_name, task_description, task_manager, task_state, completion_state, validation_state, deadline)
- plot_unit(plot_id, unit)
- plant(id, plant_name, water_need)
- do(account_id, task_id)
- link(account_id, garden_id)

1.2 Implémentation de la base de donnée dans le backend de l'application

1.2.1 Création de la base de données

En utilisant le systeme de gestion de base de données sqlite, nous avons créé notre base dans un fichier data.db avec le script SQL sauvegardé dans un fichier nommé "creation table.sql". Ces deux fichiers sont dans le dossier backend/data du projet.

1.2.2 Utilisation de SQLAlchemy

SQLAlchemy est un ORM (Object-Relational Mapping) permettant de manipuler la base de données via des objets python. Les requêtes en python sont ainsi "traduites" en SQL et la réponse reçu se présentera sous la forme d'un objet python avec lequel on peut interagir. SQLAlchemy constitue donc un pont entre la base de données et notre application. Pour que l'ORM puisse fonctionner, il faut définir des classes python qui correspondent aux tables de la base de données. Ces classes ont des attributs qui correspondent aux attributs des tables. On appelle ça des modèles. Les modèles sont ensuite utilisés pour créer des requêtes SQL.

Les sessions de SQLAlchemy permettent de gérer les transactions SQL, autrement dit un ensemble de requêtes. Si l'une d'elles échoue, l'ensemble de la transaction est annulée et aucune requête n'est communiquée à la base. L'avantage de ce système est la sécurité.

2 Serveur et client Web

2.1 structure de l'application

L'application est divisée en deux parties afin de pouvoir maitriser complètement le coté client, important pour la modélisation de jardins :

- Le backend, qui est le coeur de l'application. C'est un serveur web flask. Celui-ci est décomposé en une multitudes de routes, retournant toutes du JSON. Il s'agit d'une API REST.
- Le frontend, qui est la partie visible de l'application. Il est entièrement réalisé avec Javascript, accompagné de la librairie React.js. Le frontend communique avec le backend via des requêtes HTTP, réalisée à l'aide de la librairie Axios.

2.2 Fonctionnalités de l'application

2.2.1 Authentification

L'application dispose d'un système d'authentification complet, afin de sécuriser l'ensemble, ainsi que pour personnaliser les fonctionnalités des utilisateur. Cette authentification se déroule via des tokens JWT, générées par le serveur au moment de la connection, puis stockés dans les cookies du navigateur de l'utilisateur. Ces tokens sont ensuite utilisés pour vérifier l'identité de l'utilisateur, et ainsi lui permettre d'accéder aux routes protégées. Cette authentification permet entre autres à l'utilisateur de créer, rejoindre des jardins, de les gérer, et de les partager en leur nom.

2.2.2 Gestion de compte

Un utilisateur authentifié dispose de plusieurs fonctionnalités afin de personnaliser et modifier son compte. Il peut ainsi modifier ses informations personnelles, telles que sa photo de profil et son nom.

2.2.3 Gestion des jardins

De nombreuses routes sont dédiées au management des jardins sur l'application. On peut en effet retrouver ces différentes opérations :

- Récupérer un jardin, en fonction de son identifiant.
- Créer un jardin, en lui donnant un nom, une description, une adresse et en choisissant le type de jardin (privé ou public).
- Supprimer un jardin qu'un utilisateur a créé.
- Modifier les informations et les membres d'un jardin.
- Rejoindre un jardin.

2.2.4 Carte interractive

Afin de rendre plus accessible la fonctionnalité de rejoindre un jardin, il est également possible de le faire depuis une carte interractive, sur laquelle est listée l'intégralité des jardins publiques, ainsi que ceux dont l'utilisateur est membre. Cette carte est réalisée à l'aide de la librairie Leaflet.

2.2.5 Interractions avec le jardin

Les utilisateurs membres de jardins peuvent interagir avec celui-ci, en fonction de leurs permissions. On peut ainsi retrouver les fonctionnalités suivantes :

- Créateur d'un jardin :
  - Modeliser le jardin
  - Gérer les utilisateurs du jardin
  - Supprimer le jardin
- Les membres et le créateur :

- Ajouter / supprimer des tâches à une parcelle
- Modifier les plantes des parcelles

Ces fonctionnalités personnalisées leur permettent donc de personnaliser au maximum leurs jardins.

3 Algorithme

3.1 Principe de l'algorithme

Le problème à résoudre ici est de pouvoir alimenter en eau le potager de manière optimale. Pour cela, nous sommes parti sur l'idée de placer des bouteilles d'eau goutte à goutte sur les parcelles du potager. Cela permet de pouvoir arroser les plantes de manière optimale, en fonction de leurs besoins. Mais aussi, de pouvoir automatiser l'alimentation en eau du potager. Ainsi le but de cet algorithme est donc de placer de manière optimale les bouteilles d'eau sur les parcelles pour en placer le moins possible.

3.2 Implémentation

3.2.1 Modélisation du problème

Tout d'abord, nous avons d'abord modéliser le jardin sous la forme d'une matrice avec le module numpy pour avoir une matrice optimisée en python, c'était la façon la plus naturel pour nous pour la représenter aussi bien en frontend qu'en algorithmique. Ensuite, nous avons implémenté l'algorithme de placement des bouteilles d'eau. Pour cela, on a posé des contraintes :

- Une bouteille d'eau alimente en eau la parcelle sur laquelle elle est placée et les parcelles adjacentes.
- Une parcelle porte un poids qui correspond à la quantité d'eau dont elle a besoin.
- Une bouteille d'eau ne peut être placée que sur une parcelle.
- Une bouteille d'eau n'arrose qu'une fois une parcelle, ainsi une plante ayant un besoin de deux devra être arrosée par deux points d'eau différents et ainsi de suite.

Nous avons tout d'abord penser à résoudre ce problème en développant un algorithme basé sur le backtracking sur le même principe que knapsack (problème des sacs à dos). Cependant, le fait de devoir tester toutes les possibilités de placement des bouteilles d'eau rendait l'algorithme trop long à exécuter. Il n'était donc pas possible de l'utiliser pour un jardin de plus de 6x6 parcelles ce qui était plutot limitant dans notre cas. Nous avons ensuite tenté de placer les points d'eau sur les plantes qui avaient le plus forts besoin mais encore une fois, le placement n'était clairement pas optimisé et il fallait placer beaucoup de points d'eau pour arroser toutes les plantes surtout dans le cas où la matrice commencait à avoir une taille représentative d'un jardin. Finalement, en prenant un autre point de vue sur le problème, il a été possible de le résoudre de manière très rapide. Il suffit alors de se rendre compte que ce problème peut etre traiter sous la forme d'un problème mathématique, à travers un système d'équations contenant la position des points d'eau et les besoins des plantes. Ainsi, la difficulté réside ici en la modélisation de ce problème en un système d'équiations linéaires pour ensuite le résoudre sous la forme d'un modèle de programmation linéaire en nombres entiers. Comment avons nous tout d'abord modéliser le potager en une matrice numpy ? :

- Chaque parcelle est représentée par un nombre entier positif et est correspond à des coordonnées (x,y) dans la matrice.
- Les parcelles vides sont représentées par un 0.
- Les parcelles contenant une plante sont représentées par un nombre entier positif correspondant au besoin en eau de la plante.

Prenons alors le cas d'un potager de taille 4x4 avec les besoins suivants :

$$M = \begin{bmatrix} 2 & 2 & 1 & 1 \\ 2 & 2 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Nous ferons allusion à la matrice M pour représenter le potager dans la suite pour plus de clarté et avoir un exemple concret.
Nous avons donc ici un potager de taille 4x4 avec un ensemble de parcelles 2x2 contenant des plantes ayant un besoin en eau de 2. Puis un ensemble de parcelles 2x4 contenant des plantes ayant un besoin en eau de 1. Le reste de la matrice contient des zéros car les parcelles sont creusées mais ne contiennent pas de plantes, elles ont donc un besoin de 0.
Nous allons maintenant modéliser ce problème en un système d'équations linéaires.

3.2.2 Modélisation du problème en un système d'équations linéaires

Nous allons donc modéliser ce problème en un système d'équations linéaires.
Pour cela, on part du constat que pour chaque parcelle de légumes, si on veut que cette parcelle voit son besoin en eau accompli, il faut que sa propre parcelle ou les parcelles adjacentes contiennent le nombre de points d'eau nécessaires pour assouvir son besoin. Ainsi, le poids de cha parcelle correspond au nombre de points d'eau qu'il faut autour d'elle ou sur elle même. Ainsi, cela constitue ce qu'on appelle une contrainte(*), on crée donc une équation pour chaque parcelle contenant une plante. Pour cela, on va écrire ce système d'équations linéaires sous la forme d'un produit matriciel.
On pose alors X la matrice contenant l'ensemble des points d'eau que l'on peut placer, cela correspond au nombre d'éléments dans la matrice M :

$$X = \begin{bmatrix} X_1 \\ X_2 \\ . \\ . \\ X_n \end{bmatrix}$$

X_i est égal à 1 si la case i est une source, et 0 sinon. On prend donc ici des valeurs binaires, soit on place un point d'eau, soit on en place pas.
On pose ensuite la matrice C, qui est la matrice donnant la position pour chaque parcelle des sources d'eau :

$$C = \begin{bmatrix} c_{1,1} & c_{1,2} & . & . & c_{1,n} \\ c_{2,1} & c_{2,2} & . & . & c_{2,n} \\ . & . & . & . & . \\ . & . & . & . & . \\ c_{n,1} & c_{n,2} & . & . & c_{n,n} \end{bmatrix}$$

$c_{i,j}$ est égal à 1 si la case j est adjacente à la case i, et 0 sinon.
On pose ensuite la matrice B, qui est la matrice donnant le besoin en eau pour chaque parcelle des sources d'eau :

$$B = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_n \end{bmatrix}$$

b_i est égal au besoin en eau de la parcelle i.
Ainsi, à l'aide de la contrainte que l'on se fixe ici (*), on peut poser alors notre système d'équations linéaires sous la forme :

$$C \cdot X = B \tag{1}$$

On a alors un système d'équations détaillé de cette forme :

$$\begin{aligned} &c_{1,1}x_1 + c_{1,2}x_2 + ... + c_{1,n^2}x_{n^2} \geq b_1 \ , x_{n^2} \in \{0,1\} \\ &c_{2,1}x_1 + c_{2,2}x_2 + ... + c_{2,n^2}x_{n^2} \geq b_2 \ , x_{n^2} \in \{0,1\} \\ &... \\ &c_{n^2,1}x_1 + c_{n^2,2}x_2 + ... + c_{n^2,n^2}x_{n^2} \geq b_{n^2} \ , x_{n^2} \in \{0,1\} \end{aligned} \tag{2}$$

où ci, j indique si le point d'eau peut être placé pour augmenter son besoin ou pas.
On peut alors résoudre ce système d'équations linéaires pour trouver le nombre de points d'eau à placer pour chaque parcelle.
Montrons maintenant concrète ce qu'il se passe au travers de la matrice M constituant un exemple de

potager.
Tout d'abord, voici la matrice C de l'explication précédente :

$$C = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Ensuite on a la matrice B correspondante au besoin des plantes dans la potager :

$$B = \begin{bmatrix} 2 & 2 & 1 & 1 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

On obtient alors le système d'équations linéaires suivante :

$$\begin{aligned}
&x_1 + x_2 + x_5 + x_6 \geq 2 \ , x_{n^2} \in \{0,1\} \\
&x_1 + x_2 + x_3 + x_5 + x_6 + x_7 \geq 2 \ , x_{n^2} \in \{0,1\} \\
&x_2 + x_3 + x_4 + x_6 + x_7 + x_8 \geq 1 \ , x_{n^2} \in \{0,1\} \\
&x_3 + x_4 + x_7 + x_8 \geq 1 \ , x_{n^2} \in \{0,1\} \\
&x_1 + x_2 + x_5 + x_6 + x_1 + x_2 + x_9 + x_{10} \geq 2 \ , x_{n^2} \in \{0,1\} \\
&x_1 + x_2 + x_3 + x_5 + x_6 + x_7 + x_9 + x_{10} + x_{11} \geq 2 \ , x_{n^2} \in \{0,1\} \\
&x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_{10} + x_{11} + x_{12} \geq 1 \ , x_{n^2} \in \{0,1\} \\
&x_3 + x_4 + x_7 + x_8 + x_{11} + x_{12} \geq 1 \ , x_{n^2} \in \{0,1\} \\
&x_6 + x_7 + x_8 + x_{10} + x_{11} + x_{12} + x_{14} + x_{15} + x_{16} \geq 1 \ , x_{n^2} \in \{0,1\} \\
&x_7 + x_8 + x_{11} + x_{12} + x_{15} + x_{16} \geq 1 \ , x_{n^2} \in \{0,1\} \\
&x_{10} + x_{11} + x_{12} + x_{14} + x_{15} + x_{16} \geq 1 \ , x_{n^2} \in \{0,1\} \\
&x_{11} + x_{12} + x_{15} + x_{16} \geq 1 \ , x_{n^2} \in \{0,1\}
\end{aligned} \tag{3}$$

Ainsi trouver les x_i qui vérifient ce système d'équations est une solution pour placer les points d'eau. Mais il existe de multiples solutions à ce problème, alors comment faire pour avoir la plus optimale ?

3.2.3 Optimisation du système d'équations linéaires

Pour cela, on va utiliser la méthode de la programmation linéaire. On va donc chercher à minimiser le nombre de points d'eau à placer pour que chaque parcelle ait son besoin en eau accompli. L'astuce ici réside dans le fait de chercher à minimiser la somme des points d'eau à placer, car xi, j correspond à des valaurs binaires, soit 0 soit 1.
On pose alors la fonction objectif qui est la somme des points d'eau à placer :

$$f(X) = \sum_{i=1}^{n^2} x_i \tag{4}$$

On peut donc écrire notre problème sous la forme :

$$\min(f(X)) \tag{5}$$

Il faut savoir qu'en programmation linéaire, il faut fixer un objectif à atteindre car c'est le but de l'optimisation linéaire, ici, cela sera de minimiser le plus possible la somme des points d'eau résultante. Ainsi en fournissant ces matrices, la contrainte étant le système d'équations linéaires qui découle du produit matriciel entre la matrice des variables (points d'eau), la matrice C qui est leurs proximités avec chaque parcelle et la matrice B le besoin en eau de chaque parcelle, on peut alors résoudre ce problème en programmation linéaire à l'aide ici du module Gurobipy de Python en fixant l'objectif décrit ci-dessus. On obtient finalement la position des points d'eau à placer dans le jardin, et grâce au modèle mathématique qui repose sur la programmation linéaire, on peut être sûr que la solution fournie est optimale.
Ainsi, dans notre exemple, on obtient :

$$B = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Et on observera alors que, effectivement, dans notre cas, le nombre de point d'eau ne peut être inférieur à 4, et il nous propose bien le positionnement de 4 points d'eau. On a donc bien trouver la solution optimale à notre problème ainsi qu'un placement possible (car il existe parfois d'autres placements de ses points qui vérifient aussi la solution mais cela n'a pas d'intêret ici)

3.3 Analyse en complexité

Si on compare par rapport à notre première version :

Le script utilise la technique de backtracking pour trouver la solution optimale, c'est-à-dire le minimum de points d'eau nécessaires pour arroser toutes les plantes dans le potager. Pour chaque case (i, j) du potager, le script explore deux possibilités: mettre un point d'eau ou ne pas en mettre. Cela signifie que pour chaque case du potager, le script fait un appel récursif, ce qui multiplie par deux le nombre d'appels récursifs à chaque itération. De plus, la fonction "backtrack" parcourt chaque case du potager (nm cases au total), ce qui multiplie encore une fois le nombre total d'appels récursifs par nm.
Ainsi, la complexité de ce script est en

$$\mathcal{O}(2^(m * n) * m * n)$$

=

$$\mathcal{O}(3^m * n)$$

.
Dans notre seconde algorithme :
L'algorithme utilise l'optimiseur Gurobi pour résoudre un modèle de programmation linéaire en nombres entiers. La complexité de cet algorithme dépend de la complexité de l'optimiseur Gurobi. La première partie de l'algorithme, qui construit les matrices C et B à partir de la matrice M, est en temps

$$\mathcal{O}(n^4)$$

, où n est la taille de la matrice M (n est égale à longueur * longueur). Cependant, cette partie est effectuée une seule fois. L'optimisation elle-même est généralement considérée comme étant en

$$\mathcal{O}(p * n^3)$$

, où p est le nombre de contraintes et n est le nombre de variables. Il est possible d'optimiser les performances en fonction des caractéristiques spécifiques du modèle. Dans l'ensemble, on peut considérer que la complexité globale de cet algorithme est de l'ordre de

$$\mathcal{O}(n^4)$$

.

3.4 Analyse de performance

4 Gestion de projet

4.1 Équipe de projet

Ce projet est un projet local réalisé en groupe de 4 personnes :

- Alexis MARCEL
- Lucas LAURENT
- Noé STEINER
- Mathias AURAND-AUGIER

Le comité de pilotage est constitué de :

- Anne-Claire HEURTEL
- Olivier FESTOR
- Gérald OSTER

Ces personnes constituent les parties prenantes de notre projet ainsi que les acteurs influents sur le livrables.

Ceux qui...	*Sont...*
... Demandent, financent le projet, propriétaires du livrable	Télécom Nancy.
... Pilotent le projet et assurent la coordination avec le client	Les professeurs gérants les différents groupes.
... Réalisent le projet	Les élèves de l'équipe projet.
... Sont également concernés	Les communautés derrière les jardins partagés/circuits-courts.

Figure 2: Parties prenantes

4.2 Organisation au sein de l'équipe projet

Nous avons réalisé plusieurs réunions, en présentiel dans les locaux de Télécom Nancy mais la plupart de notre collaboration a eu lieu sur Discord. Ces réunions nous ont permis de mettre en commun nos avancés régulièrement, de partager nos connaissances sur des problématiques et de nous organiser de manière optimale. En plus des réunions d'avancement règulières, nous avons également réalisé des réunions techniques afin de résoudre un problème ou bien de réflechir à la conception. Les comptes rendus des réunions réalisés sont présents dans l'Annexe 1.

De plus, dès le début de notre projet nous avons mis en place un projet Trello. Trello est une application permettant d'organiser facilement un projet en reposant sur une organisation en planches listant des cartes, chacune représentant des tâches. Ces tâches peuvent ensuite être déplacées permettant de découper notre projet en plusieurs jalons dynamiquement.

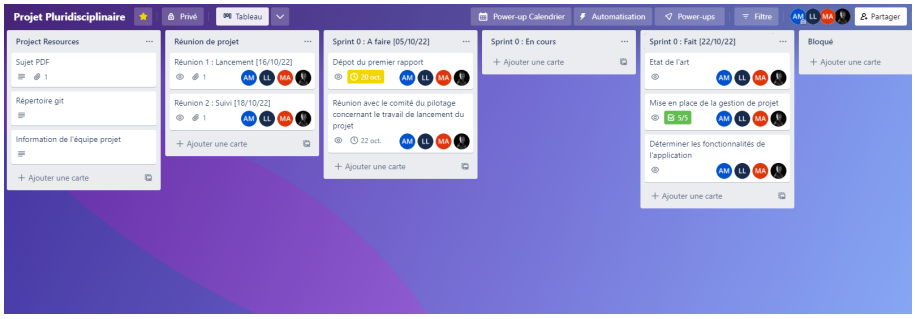

Figure 3: Organisation Trello

Ensuite, nous avons utilisé GitLab pour gérer les différentes versions du développement de notre application, ainsi que les différentes branches nous permettant de travailler simultanément sans conflit.

Enfin, la rédaction des differents comptes rendu de réunion et des rapports ont été rédigé en LaTeX.

4.3 Objectifs SMART

La méthode SMART que l'on rappelle ci-dessous nous a permis de définir nos différents objectifs :

	Critère	Indicateur
S	Spécifique	L'objectif est clairement défini.
M	Mesurable	On peut suivre et quantifier la progression de l'objectif.
A	Atteignable	L'objectif prend en compte la capacité des membres du projet à l'atteindre et des moyens mis à disposition.
R	Réaliste	L'objectif doit être réaliste, réalisable et pertinent par rapport à la situation.
T	Temporellement défini	Le projet doit être limité dans le temps, avec une date de fin.

Figure 4: Objectif SMART

4.4 Matrice des objectifs

Nous avons conçu, à l'aide de la méthode SMART, la matrice des objectifs suivante :

	LIVRABLE	DOCUMENTATION	VALORISATION DU PROJET	ACQUISITION DE COMPETENCES / FORMATIONS
INSUFFISANT	Créer un nouveau compte et utiliser l'application	Pas de recherche effectuée	Projet approuvé par tous les membres	Acquis superficiels
RÉUSSITE ACCEPTABLE	Reproduire son jardin sous forme numérique	Recherche sur les jardins partagés et les circuits-courts ainsi que ses possibilités	Conception du projet validé	
BON TRAVAIL	Créer un nouveau jardin et l'ouvrir aux autres	Étude des applications de jardins partagés		Être capable de réaliser une autre application similaire
EXCELLENT	Participer et gérer un jardin privé ou partagé	Tableau comparatif des applications étudiées		Avoir acquis intégralement toutes les notions utilisées durant le projet pour l'ensemble du groupe

Figure 5: Matrice des objectifs

4.5 Triangle qualité-cout-délai

Afin d'établir des objectifs cohérents, et réalisables dans les délais, nous avons réalisé le triangle qualité-coût-délai. On remarque ainsi, les délais étant courts, que nous avons tout intérêt à ne pas se fixer des objectifs trop ambitieux sous peine de devoir renoncer à certaines fonctionnalités et de ne pas rendre le livrable annoncé initialement.

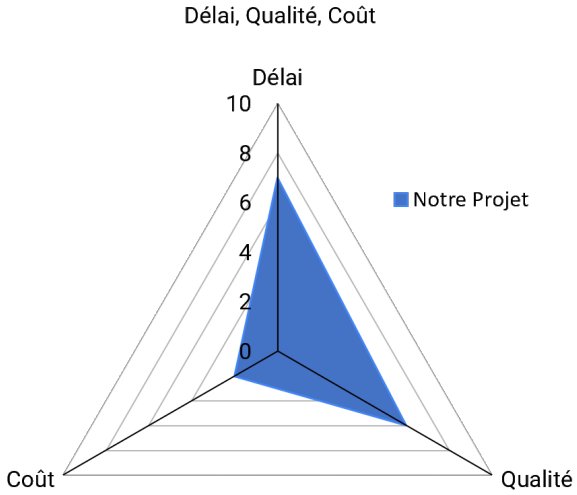

Figure 6: Triangle DQC

4.6 Matrice SWOT

Afin d'avoir une vision plus globale de nos ressources et des facteurs interne et externe agissant sur le projet, nous avons ensuite réalisé la matrice SWOT (Strengths, Weaknesses, Opportunities, Threats) de notre projet.

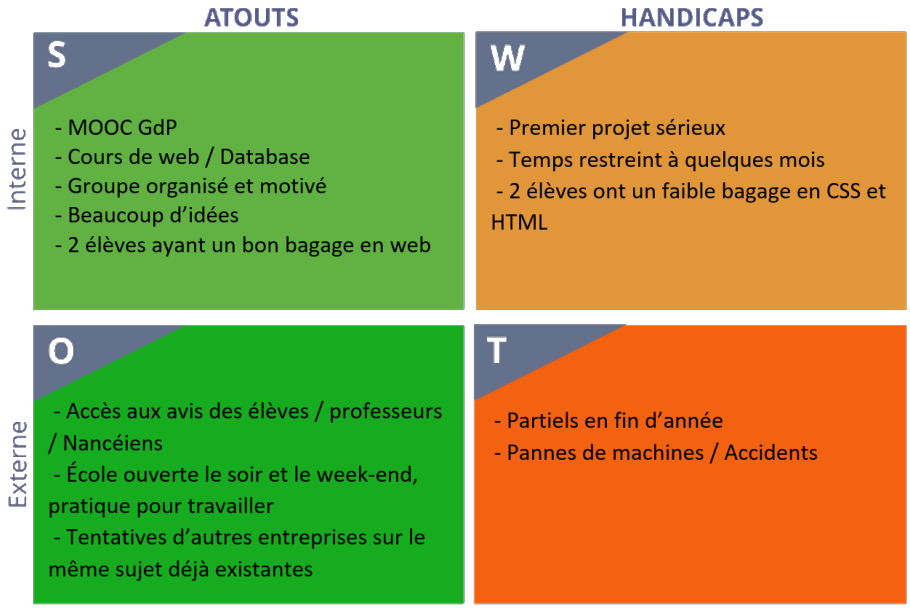

Figure 7: Matrice SWOT

On peut ainsi remarquer que notre projet présente de nombreux points fort notamment grâce aux connaissances acquises lors des cours de Télécom Nancy mais également de part l'expérience forte de deux des membres de l'équipe projet qui ont déjà réalisé des applications similaires. Cependant, plusieurs facteurs internes constituent nos faiblesses notamment les courts délais qui nous oblige à être concis et efficaces dans notre travail, ou encore le faible bagage informatique de deux des membres de l'équipe. Néanmoins, ces lacunes constituent pour eux l'opportunité d'apprendre, et de progresser avec l'aide des membres expérimentés de l'équipe.

De plus, nous devons anticiper les charges de travail dans le cadre de notre formation à Télécom Nancy qui s'avèrent être plus élevée en décembre lors des partiels de fin d'année. Nous allons donc devoir prendre cela en compte dans notre gestion des tâches.

4.7 Profil de projet

Afin d'avoir une vision plus globale sur notre projet, nous avons également réalisé le profil du projet (le budget étant égal à 0, nous avons choisi de ne pas le représenter dans notre profil). On remarque que, du fait des nombreuses fonctionnalités que nous avons l'intention d'implémenter dans notre application, que notre projet est de taille moyenne mais de complexité élevée.

Cependant, les enjeux du projet ne sont pas très importants (en dehors de la note finale qui compte dans notre moyenne) car l'échec du projet n'engendra pas la chute d'une organisation et le budget est négligeable.

De plus, au vu de l'état de l'art établi, l'innovation du projet est importante puisque nous avons choisi de combiner différentes fonctionnalités existantes de plusieurs applications et d'en rajouter de nouvelles.

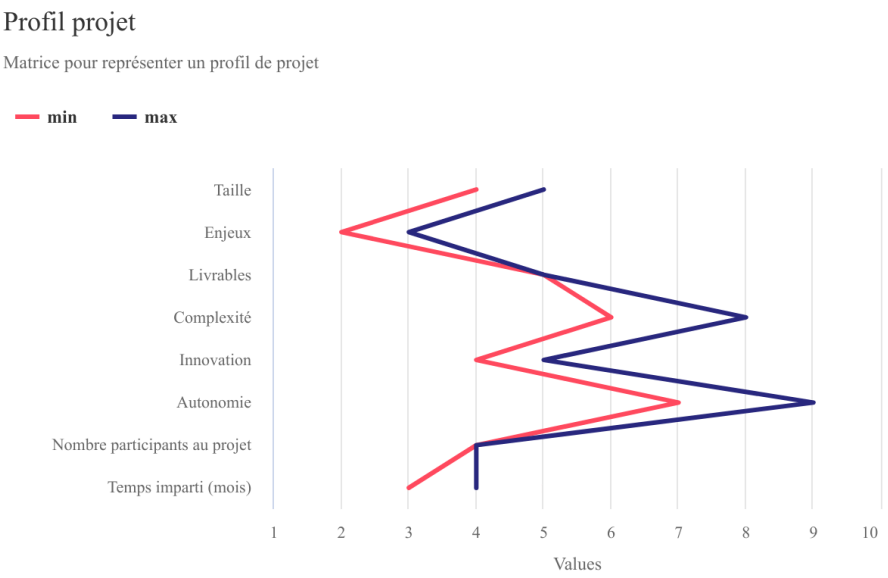

Figure 8: Profil du projet

4.8 WBS : comment concrétiser l'application

Ceci étant fait, nous avons maintenant choisi de détailler les lots de travail à effectuer pour fabriquer notre application. Nous avons ainsi réalisé le WBS (Work Breakdown Structure) de notre application : il apparait ainsi les grandes étapes de notre projet que sont : definition du cadre de l'application, développement des fonctionnalités de l'application et écriture du rapport.

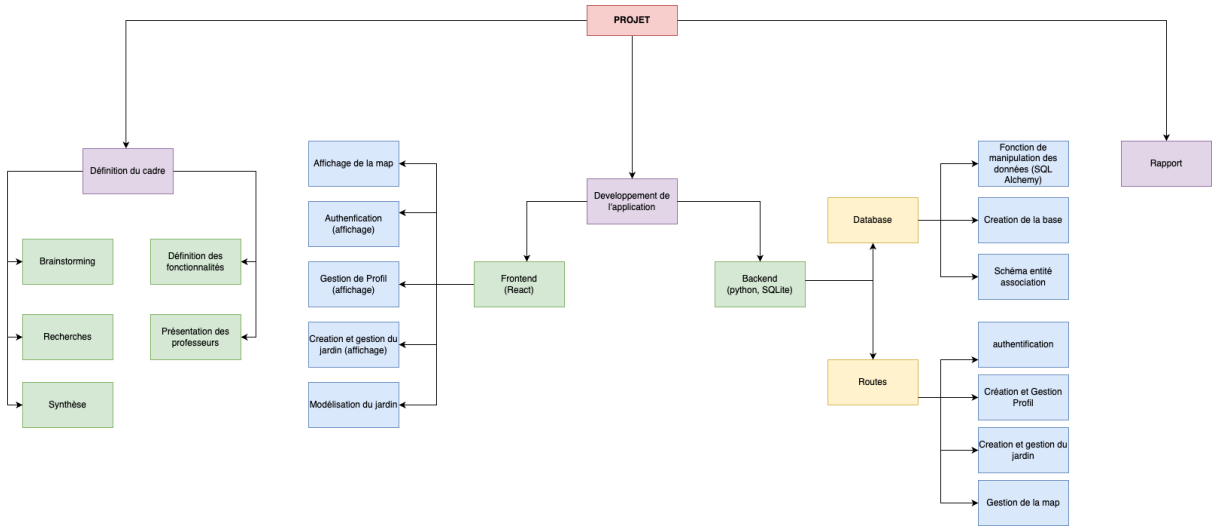

Figure 9: WBS

4.9 Diagramme de Gantt : planification

Maintenant que nous avons un détail des lots de travail qui constitue notre application, il faut maintenant les mettre en relation pour créer un planning efficace où chaque tâche est effectuée dans l'ordre.

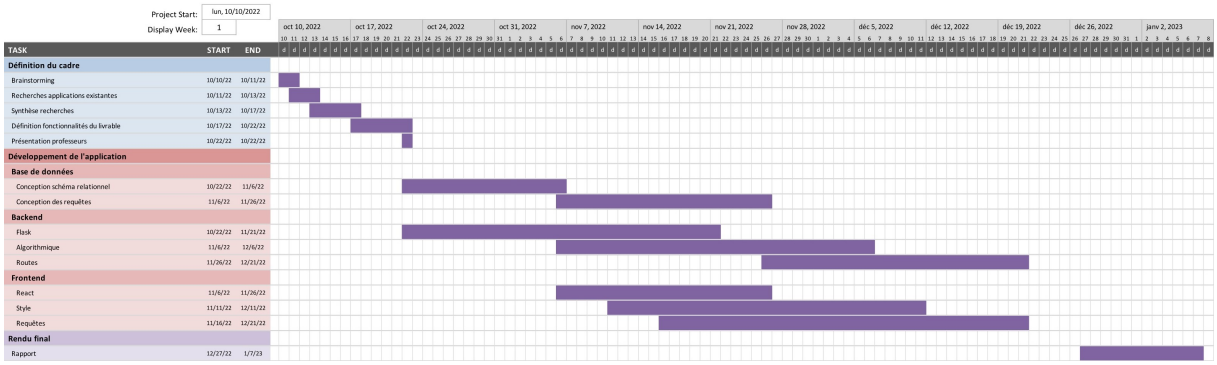

Figure 10: Diagramme de GANTT

Ce diagramme est une première version générale des tâches à effectuer, il sera modifié et détaillé davantage une fois la conception et les maquettes du projet réalisé.

4.10 Matrice RACI

Maintenant que toutes les étapes sont planifiées, nous devons répartir le travail entre les membres de l'équipe. On utilise ainsi une matrice RACI synthétisant les rôles de chacun.

Matrice RACI (R = Réalise ; A = Autorité ; C = Consulté ; I = Informé)	**Acteurs**				
Tâches	Lucas	Noé	Alexis	Matthias	Autres élèves
Base de données					
Schéma relationnel base de données	R	R	R	AR	
Création de la base de données	R	R	R	AR	
Ensemble des requêtes	R	R	R	AR	
Frontend					
Style	R	C	AR	I	
Requêtes HTTP	AR	R	R	C	
React	C	AR	C	R	
Backend					
Algorithmique	AR	R	R	R	
Python/Flask	AR	R	R	C	
Routes	I	AR	R	I	
Rédaction du rapport	R	R	R	R	C

Figure 11: Matrice RACI

4.11 Gestion des risques

Nous avons également penser à prévoir une partie des risques pouvant se dresser sur notre route, les risques les plus classiques étant la gestion du temps et le manque de comprehension de certaines personnes de l'équipe

Description	Gravité 1-4	Fréquence 1-4	Criticité	Resp	Prévention	Réparation / Plan B
Un des membres de l'équipe se démotive ou se désinteresse du projet	3,5	2,5	**8,75**	Noé Steiner	*A chaque réunion, faire le bilan de ce qui a été fait. La mise en commun de toutes les avancées du groupe est indispensable pour ne perdre personne.*	
Mésententes dans l'équipe	3	2,5	**7,5**	Mathias Aurand-Augier	*Ecouter les autres et se remettre en question. Expliciter les problèmes et trouver une solution commune*	Proposition de changement de tâches vers une partie jugée plus intéressante
Délai non respectés/Mauvaise gestion du temps	3	2	**6**	Alexis Marcel	*Respecter au mieux le diagramme de GANTT établi lors de l'étude préalable du projet.*	Réunion d'urgence pour combler les vides et répartir la charge de travail.
Le résultat ne correspond pas aux attentes du client	3,5	1,5	**5,25**	Mathias Aurand-Augier	*Tenter de respecter au mieux le cahier des charges établi*	
Un des membres de l'equipe est incompétent(e)	2	2	**4**	Noé Steiner	*Réactualiser les connaissances nécessaires, et planifier des formations en fonction des besoins. Mettre à disposition les ressources nécéssaires.*	Alléger les tâches et mettre à disposition les éléments nécessaires pour poursuivre le travail.
L'ordinateur ne se lance pas le jour de la soutenance	3,5	1	**3,5**	Alexis Marcel	*Chacun apportera son ordinateur pour que chaque personne soit en mesure de lancer le projet en cas de problème*	
Le projet est inutilisable par le prof (problème de dépendances, ...)	3	1	**3**	Lucas Laurent	*Expliquer au mieux dans un fichier README tous les modules à installer pour faire fonctionner l'application*	Consolider la documentation utilisateur

Figure 12: Plan de gestion des risques

5 Conclusion

Ce projet nous a permis de réaliser un projet de A à Z, de la conception à la réalisation. Nous avons pu mettre en pratique les connaissances acquises en cours et nous avons pu nous familiariser avec les outils de gestion de projet. Nous avons également pu nous rendre compte de la difficulté de la gestion de projet et de la nécessité de bien planifier son travail. Nous avons réalisé la totalité des fonctionnalités et des éléments prévus lors de la phase de conception, ce qui nous permet de qualifier notre travail d'excellent selon notre matrice des objectifs. Ce travail a également été l'occasion de découvrir de nouvelles technologies, comme les librairies React, Gurobipy et SQLAlchemy, mais également de découvrir de nouvelles méthodes de résolution de problème, via notre algorithme d'optimisation de l'arrosage. Nous sommes fier du rendu final de notre travail et espérons retravailler ensemble sur d'autres projets, y compris externes à ceux donné dans le cadre des cours.

6 Annexes

Compte rendu de réunion

Noé Steiner - Alexis Marcel - Lucas Laurent - Mathias Aurand-Augier

16 Octobre 2022

Projet PPII - Compte rendu n°01 - réunion de lancement

Motif / type de réunion:	Lieu:
• Alexis : Présent • Noé : Présent • Lucas : Présent • Mathias : Présent	• Le 16 Octobre 2022 • De 20h à 21h • Visioconférence sur Discord

Ordre du jour:

- Tâches à effectuer
- Répartition des tâches
- Vision globale de l'application

Information échangées

- Point de vue des membres sur les fonctionnalités de l'application

Remarques / Questions

Aucune question ou remarque spécifique relevée.

Décisions

- Fonctionnalités de l'application

Actions à suivre / Todo list

- Etudier les application similaire présentes sur le marché (Tout le monde)
- Etablir une listes des documents à produire relatifs à la gestion de projet (Tout le monde)
- Etablir une liste des fonctionnalités possibles de l'application (Tout le monde)

Date de la prochaine réunion

La prochaine réunion aura lieue le Mardi 18 Octobre 2022, de 20h à 21h.

Compte rendu de réunion

Noé Steiner - Alexis Marcel - Lucas Laurent - Mathias Aurand-Augier

18 Octobre 2022

Projet PPII - Compte rendu n°02 - réunion de suivi

Motif / type de réunion:	Lieu:
• Alexis : Présent • Noé : Présent • Lucas : Présent • Mathias : Présent	• Le 18 Octobre 2022 • De 20h à 21h • Visioconférence sur Discord

Ordre du jour:

- Point sur l'avancement des tâches
- Demande d'aide en cas de difficultés

Information échangées

- Principaux éléments de gestion de projet terminés
- Intégration du livrable en LaTeX en cours
- Liste des fonctionnalités détaillées de l'application

Remarques / Questions

- Comment créer le diagramme de GANTT ? Quelle application ?

Décisions

Actions à suivre / Todo list

- Intégration des documents vers un document LaTeX (Noé)
- Réalisation graphique des figures (Lucas)
- Rédaction de la partie gestion de projet et fonctionnalités de l'application (Alexis et Mathias)

Date de la prochaine réunion

La prochaine réunion aura lieue le Samedi 22 Octobre 2022, de 20h à 21h.

Compte rendu de réunion

Noé Steiner - Alexis Marcel - Lucas Laurent - Mathias Aurand-Augier

18 Novembre 2022

Projet PPII - Compte rendu n°03 - réunion de lancement

Motif / type de réunion:	Lieu:
• Alexis : Présent • Noé : Présent • Lucas : Présent • Mathias : Présent	• Le 18 Novembre 2022 • De 21h à 22h • Visioconférence sur Discord

Ordre du jour:

- Répartition des tâches pour le premier jalon de développement du projet
- Hébergement du projet
- Spécification de l'architecture de l'application

Informations échangées

- Répartition des tâches pour le premier jalon de développement du projet :
  - Alexis : Développement du JSX à partir des maquettes
  - Lucas : Développement du JSX à partir des maquettes
  - Noé : Authentification et gestion des utilisateurs
  - Mathias : Base de données et gestion des données
- Hébergement du projet :
  - Hébergement sur un serveur personnel de Lucas
- Spécification de l'architecture de l'application :
  - Utilisation de ViteJs pour la compilation du Front-End
  - Utilisation de Axios pour les requêtes HTTP
  - Utilisation de Flask SQLAlchemy pour la gestion de la base de données

Remarques / Questions

Comment va-t-on stocker la modélisation du jardin ?

Décisions

-

Actions à suivre / Todo list

- Progression du développement du JSX
- Vérification de la base de données avec SQLAlchemy
- Bon fonctionnement de l'authentification

Compte rendu de réunion

Noé Steiner - Alexis Marcel - Lucas Laurent - Mathias Aurand-Augier

30 Novembre 2022

Projet PPII - Compte rendu n°04 - réunion d'avancement

Motif / type de réunion:	Lieu:
• Alexis : Présent • Noé : Présent • Lucas : Présent • Mathias : Présent	• Le 30 Novembre 2022 • De 21h à 22h • Visioconférence sur Discord

Ordre du jour:

- - Progression du développement du JSX
- Vérification de la base de données avec SQLAlchemy
- Bon fonctionnement de l'authentification

Informations échangées

- Répartition des tâches pour la prochaine réunion :
  - Alexis : aide Mathias à faire page de profil
  - Lucas : Faire rejoindre un jardin depuis la carte
  - Noé : Rejoindre un jardin tout court
  - Mathias : Page de profil d'un utilisateur
- Bilan travail fait :
  - Base de donnée avec SQLAlchemy, authentification

Remarques / Questions

Notre code est il optimisé ? Aurions nous pu faire autrement ?

Décisions

- Coacher Mathias pour l'apprentissage de Javascript et Réact

Actions à suivre / Todo list

- Test pour rejoindre un jardin
- Finalisation carte emplacement jardin
- Bon fonctionnement de la page de profil d'utilisateur

Date de la prochaine réunion

La prochaine réunion aura lieue le Vendredi 16 decembre 2022, de 20h à 21h.

Compte rendu de réunion

Noé Steiner - Alexis Marcel - Lucas Laurent - Mathias Aurand-Augier

18 Novembre 2022

Projet PPII - Compte rendu n°05 - réunion d'avancement

Motif / type de réunion:	Lieu:
• Alexis : Présent • Noé : Présent • Lucas : Présent • Mathias : Présent	• Le 16 decembre 2022 • De 20h à 21h • Visioconférence sur Discord

Ordre du jour:

- Répartition des tâches pour le premier jalon de développement du projet
- Hébergement du projet
- Spécification de l'architecture de l'application

Informations échangées

- Répartition des tâches pour le premier jalon de développement du projet :
  - Alexis : Page de profil, creation de jardin
  - Lucas : Recherche jardin sur la map
  - Noé : Joindre un jardin
  - Mathias : Modification des informations du profils

Actions à suivre / Todo list

- Progression du développement du JSX
- Bon fonctionnement de la carte des jardins

Date de la prochaine réunion

La prochaine réunion aura lieue le mardi 27 decembre 2022, de 14h à 15h.

Compte rendu de réunion

Noé Steiner - Alexis Marcel - Lucas Laurent - Mathias Aurand-Augier

18 Novembre 2022

Projet PPII - Compte rendu n°08 - réunion d'avancement

Motif / type de réunion:	Lieu:
• Alexis : Présent • Noé : Présent • Lucas : Présent • Mathias : Présent	• Le 27 décembre 2022 • De 14h à 15h • Visioconférence sur Discord

Ordre du jour:

- Répartition des tâches la suite du développement du projet
- Progression developpement JSX

Informations échangées

- Répartition des tâches pour le premier jalon de développement du projet :
  - Alexis : modélisation jardin et gestion parcelles
  - Lucas : Optimisation du code écrit
  - Noé : modélisation jardin et gestion parcelles
  - Mathias : Finaliser gestion de projet et commencer rapport

Remarques / Questions

Que fera notre algorithme ?

Actions à suivre / Todo list

- Vérification de la gestion du jardin
- Test bon fonctionnement de l'application

Date de la prochaine réunion

La prochaine réunion aura lieue le samedi 7 janvier 2023, de 20h à 21h.