

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

Table des matières

1	Contexte du projet	2
2	Introduction	2
3	Base de données	2
3.1	Conception	2
3.1.1	Les besoins de notre application	2
3.1.2	Schéma entité-association	2
3.1.3	Passage du modèle entité-association au relationnel	3
3.2	Implémentation de la base de données dans le backend de l'application	4
3.2.1	Création de la base de données	4
3.2.2	Utilisation de SQLAlchemy	4
4	Serveur et client Web	5
4.1	Structure de l'application	5
4.2	Fonctionnalités de l'application	5
4.2.1	Authentification	5
4.2.2	Gestion de compte	5
4.2.3	Gestion des jardins	5
4.2.4	Carte interactive	5
4.2.5	Interactions avec le jardin	5
5	Algorithme	7
5.1	Principe de l'algorithme	7
5.2	Implémentation	7
5.2.1	Modélisation du problème	7
5.2.2	Modélisation du problème en un système d'équations linéaires	8
5.2.3	Optimisation du système d'équations linéaires	10
5.3	Analyse en complexité	10
5.4	Analyse de performance	11
6	Gestion de projet	13
6.1	Équipe de projet	13
6.2	Organisation au sein de l'équipe projet	13
6.3	Objectifs SMART	14
6.4	Matrice des objectifs	14
6.5	Triangle qualité-cout-délai	14
6.6	Matrice SWOT	15
6.7	Profil de projet	16
6.8	WBS : comment concrétiser l'application	16
6.9	Diagramme de Gantt : planification	17
6.10	Matrice RACI	17
6.11	Gestion des risques	18
7	Conclusion	18
8	Annexes	19

1 Contexte du projet

Ce rapport rend compte du Projet Pluridisciplinaire d'Informatique Intégrative dans le cadre de la première année du cycle ingénieur à TELECOM Nancy. L'objectif de ce projet est de concevoir, en groupe, une application dédiée à l'optimisation des ressources dans les vergers et potagers partagés du territoire.

2 Introduction

Les enjeux climatiques n'ont jamais été aussi élevés. Notre planète arrive à la fin de ses énergies fossiles. Les ressources deviennent rares, les changements climatiques provoquent des pénuries, des sécheresses et des catastrophes naturelles. Le mode de vie que nous connaissons aujourd'hui doit changer et devenir plus écoresponsable. L'objectif est de changer les habitudes de consommation de produits venants du monde entier en se restreignant à une zone géographique bien plus petite. La politique actuelle serait de privilégier les produits de notre pays mais essayons d'aller plus loin... En effet, un mode de vie plus judicieux écologiquement serait de revenir à des productions locales en passant directement par les producteurs. Certaines personnes ont une surface de production (jardin, verger) mais n'ont pas le temps ou les moyens de s'en occuper. Tandis que d'autres personnes, n'ont pas de terrain mais sont prêtes à donner de leur temps pour pouvoir consommer localement. Notre solution : utiliser les technologies modernes du web pour proposer une application permettant à n'importe qui de créer/rejoindre un jardin mais surtout de pouvoir gérer facilement un jardin sur lequel plusieurs personnes vont agir.

3 Base de données

3.1 Conception

3.1.1 Les besoins de notre application

Pour commencer, notre application permet à des utilisateurs de s'enregistrer sur la plateforme. Un 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 a un nom et un besoin en eau. Les parcelles sont elles mêmes constituées d'unités qui représentent un espace réel dans le jardin. Une unité est caractérisée par ses coordonnées déterminant sa position dans le jardin. De plus, chaque parcelle peut être associée à une liste de tâches à effectuer. Une tâche est caractérisée par une date limite, un nom, une description, un état (à faire, en cours, terminé) et enfin une personne qui s'en occupe.

3.1.2 Schéma entité-association

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

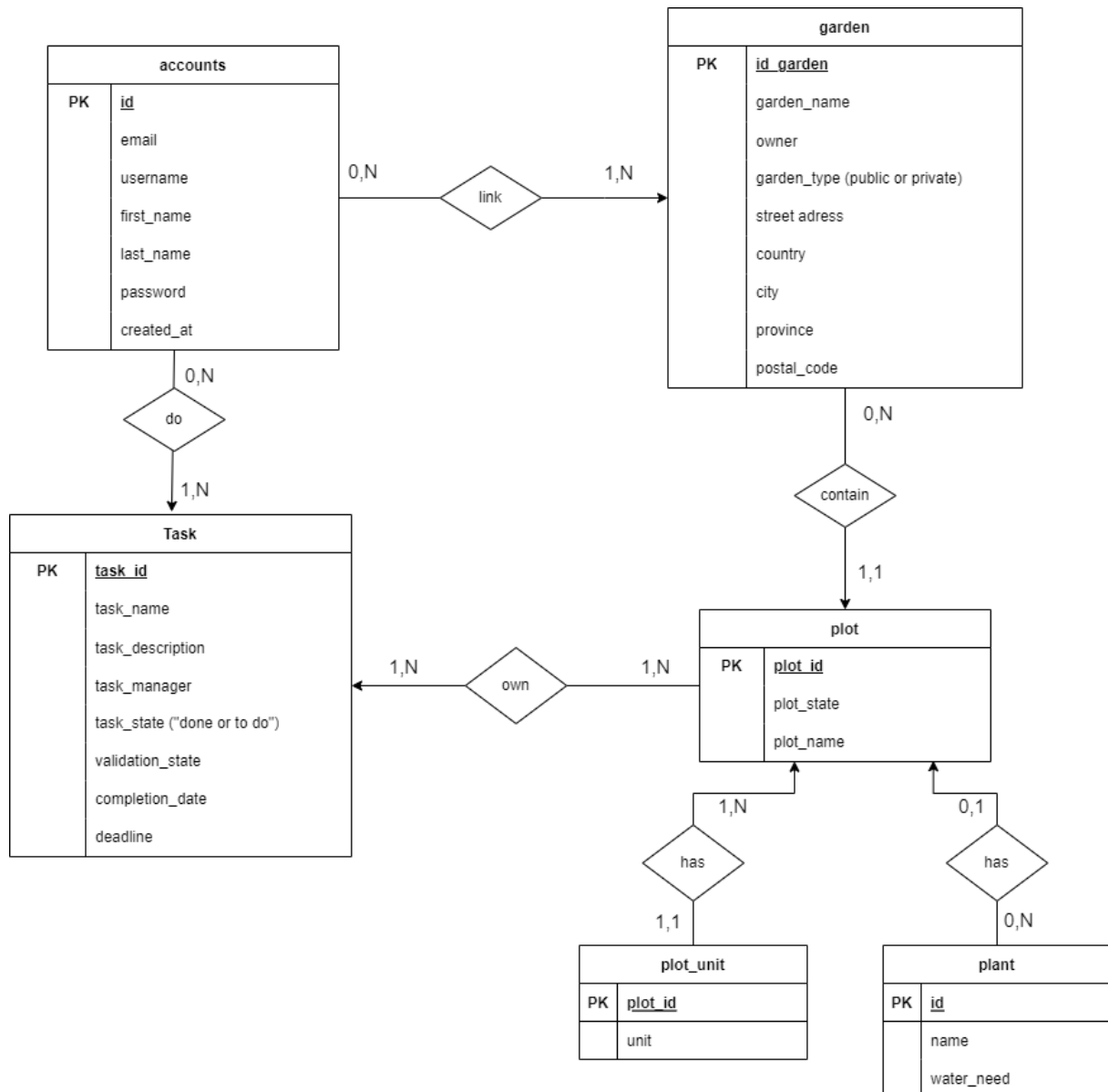


Figure 1: Schéma entité-association

Ce schéma respecte les contraintes logiques de cardinalités suivantes :

- Une personne 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 attribuée à plusieurs parcelles.
- Une personne peut effectuer plusieurs tâches et une tâche peut être effectuée par plusieurs personnes.

3.1.3 Passage du modèle 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, x, y)`
- `plant(id, plant_name, water_need)`
- `do(account_id, task_id)`
- `link(account_id, garden_id)`

3.2 Implémentation de la base de données dans le backend de l'application

3.2.1 Création de la base de données

En utilisant le système 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.

3.2.2 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 reçue 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.

4 Serveur et client Web

4.1 Structure de l'application

L'application est divisée en deux parties afin de pouvoir maîtriser complètement le côté 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 multitude 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, réalisées à l'aide de la librairie Axios.

4.2 Fonctionnalités de l'application

4.2.1 Authentification

L'application dispose d'un système d'authentification complet, afin de sécuriser l'ensemble, ainsi que pour personnaliser les fonctionnalités des utilisateurs. Cette authentification se déroule via des tokens JWT, générés par le serveur au moment de la connexion, 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.

4.2.2 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.

4.2.3 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.

4.2.4 Carte interactive

Afin de rendre plus accessible la fonctionnalité de rejoindre un jardin, il est également possible de le faire depuis une carte interactive, sur laquelle est listée l'intégralité des jardins publics, ainsi que ceux dont l'utilisateur est membre. Cette carte est réalisée à l'aide de la librairie Folium. On a modifié l'html de la carte retournée pour pouvoir afficher des boutons dans les popup des jardins non rejoints pour pouvoir les rejoindre par ce biais. On change aussi pour chaque utilisateur la couleur du marker de chaque jardin pour indiquer si il a été rejoint ou non. La carte étant une variable locale, le script a été optimisé pour que l'on ait pas besoin de recréer entièrement la carte à chaque modification de celle-ci.

4.2.5 Interactions avec le jardin

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

- Créateur d'un jardin :
 - Modéliser 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 leur permettent donc de personnaliser au maximum leurs jardins.

5 Algorithme

5.1 Principe de l'algorithme

Le problème à résoudre ici est de pouvoir alimenter en eau le potager de manière optimale. Pour cela, nous sommes partis sur l'idée de placer des arroseurs 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 arroseurs sur les parcelles pour en placer le moins possible.



Figure 2: Arroseur sur pic

5.2 Implémentation

5.2.1 Modélisation du problème

Tout d'abord, nous avons modélisé 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 naturelle pour nous pour la représenter aussi bien en frontend qu'en algorithmique. Ensuite, nous avons implémenté l'algorithme de placement des arroseurs. Pour cela, on a posé des contraintes :

- Un arroseur alimente en eau la parcelle sur laquelle il est placé et les parcelles adjacentes.
- Une parcelle porte un poids qui correspond à la quantité d'eau dont elle a besoin.
- Un arroseur ne peut être placé que sur une parcelle.
- Un arroseur 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 pensé 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 arroseurs 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 plutôt limitant dans notre cas. Nous avons ensuite tenté de placer les points d'eau sur les plantes qui avaient les plus forts besoins 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 commençait 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 être traité 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'équations linéaires pour ensuite le résoudre sous la forme d'un modèle de programmation linéaire.

Comment avons nous tout d'abord modélisé le potager en une matrice numpy ? :

- Chaque parcelle est représentée par un nombre entier positif et 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.

5.2.2 Modélisation du problème en un système d'équations linéaires

Pour cela, on part du constat que pour chaque parcelle, il suffit que la somme des sources d'eau aussi bien sur elle qu'autour d'elle soit supérieur ou égal à son besoin. Donc, le poids de chaque 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 :

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

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 :

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

$c_{i,j}$ 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 :

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

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 :

$$C \cdot X = B \quad (1)$$

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

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

où $c_{i,j}$ indique si le point d'eau peut être placé pour augmenter son besoin ou pas.

On peut alors résoudre ce système d'inéquations linéaires pour trouver le nombre de points d'eau à placer pour chaque parcelle.

Montrons maintenant concrètement 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 le potager :

$$B = [2 \quad 2 \quad 1 \quad 1 \quad 2 \quad 2 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]$$

On obtient alors le système d'équations linéaires suivante :

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

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 ?

5.2.3 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 $x_{i,j}$ correspond à des valeurs binaires, soit 0 soit 1.

On pose alors la fonction objectif qui est la somme des points d'eau à placer :

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

On peut donc écrire notre problème sous la forme :

$$\min(f(X)) \quad (5)$$

Il faut savoir qu'en programmation linéaire, le but est de maximiser ou minimiser la fonction objectif, 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 leur proximité avec chaque parcelle et la matrice B le besoin en eau de chaque parcelle. On peut alors résoudre ce problème de programmation linéaire à l'aide 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 points d'eau ne peut être inférieur à 4, et il nous propose bien le positionnement de 4 points d'eau. On a donc bien trouvé la solution optimale à notre problème ainsi qu'un placement possible (car il existe parfois d'autres placements de ces points qui vérifient aussi la solution mais cela n'a pas d'intérêt ici).

5.3 Analyse en complexité

Comparons les deux versions développées pour répondre au problème, la première version étant celle qui utilise la technique du backtracking et la seconde celle qui utilise la programmation linéaire :

Par rapport à notre première version de résolution du problème :

L'algorithme 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 :

$$\mathcal{O}(2^{(m*n)} * m * n) = \mathcal{O}(3^{(m*n)})$$

Par rapport à notre deuxième version de résolution du problème :

L'algorithme utilise l'optimiseur Gurobi pour résoudre un modèle de programmation linéaire en nombres

binaires. 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

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

Où n est le nombre d'élément dans la matrice du potager, car pour créer la matrice C, on crée pour chaque élément parcouru dans la matrice, on crée une liste de taille n, on a donc une complexité en :

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

Contrairement à la construction de la matrice B qui est en :

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

L'optimisation elle-même est généralement considérée comme étant en :

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

Où n est le nombre de variables. Dans l'ensemble, on peut considérer que la complexité globale de cet algorithme est de l'ordre de :

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

Où n est le nombre d'éléments de la matrice du potager.

5.4 Analyse de performance

Ici, nous allons en profiter pour comparer nos deux algorithmes avec des matrices carrées de taille n, dont la taille va augmenter de 1 à chaque fois. Voici les graphes résultants de ses tests :

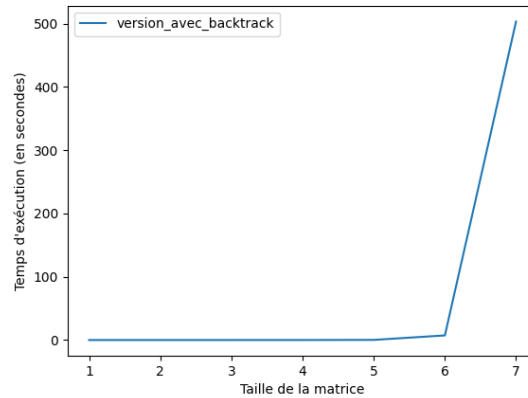


Figure 3: Courbe de performance de l'algorithme de backtracking

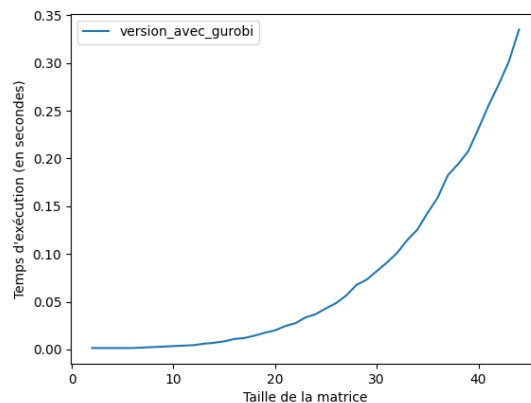


Figure 4: Courbe de performance de l'algorithme de programmation linéaire

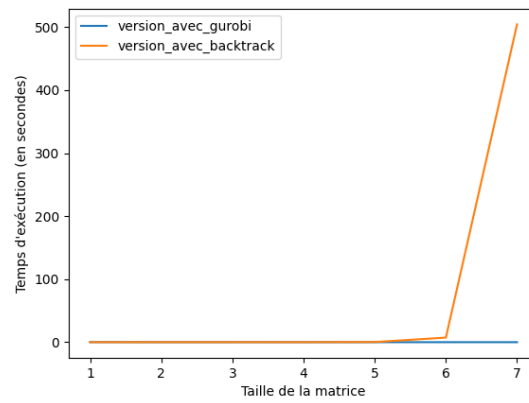


Figure 5: Comparaison des deux algorithmes

On voit que très clairement, la version avec le module Gurobipy est beaucoup plus rapide que la version avec backtracking, et cela est logique en vu de la complexité de ces deux algorithmes explicités précédemment. On remarque surtout que, dans le cas du backtracking, la complexité croit très vite et que pour des matrices de taille 7, le temps d'exécution est déjà très long à tel point qu'il est déjà beaucoup trop long pour une matrice 8x8. Alors que pour la version avec gurobi, la complexité est beaucoup plus stable, et on peut même voir que pour des matrices de taille 45x45, le temps d'exécution est toujours inférieur à 0.4 secondes. Ce qui en fait notre algorithme préféré pour résoudre ce problème, surtout qu'il est le seul à pouvoir traiter notre potager de taille 12x12 en un temps plus que raisonnable.

6 Gestion de projet

6.1 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
- Gérard 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 6: Parties prenantes

6.2 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 avancées régulièrement, de partager nos connaissances sur des problématiques et de nous organiser de manière optimale. En plus des réunions d'avancement régulières, nous avons également réalisé des réunions techniques afin de résoudre un problème ou bien de réfléchir à la conception. Les comptes rendus des réunions réalisées 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.

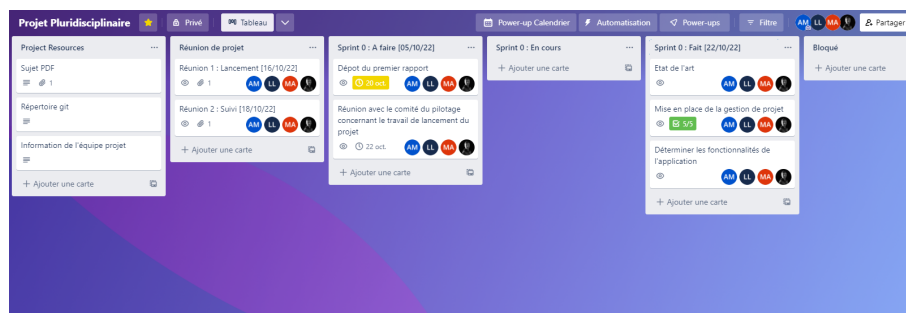


Figure 7: Organisation Trello

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 différents comptes rendu de réunion et des rapports ont été rédigé en L^AT_EX.

6.3 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 8: Objectifs SMART

6.4 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 9: Matrice des objectifs

6.5 Triangle qualité-cout-délai

Afin d'établir des objectifs cohérents, et réalisables dans les délais, nous avons réalisé le triangle qualité-cout-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.

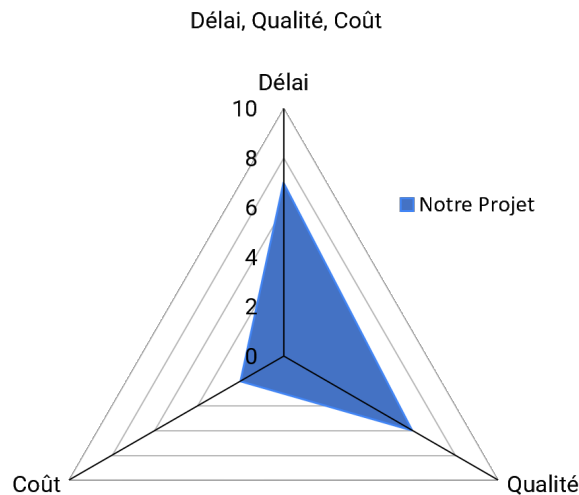


Figure 10: Triangle DQC

6.6 Matrice SWOT

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



Figure 11: Matrice SWOT

On peut ainsi remarquer que notre projet présente de nombreux points forts 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 obligent 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 élevées en décembre lors des partiels de fin d'année. Nous allons donc devoir prendre cela en compte dans notre gestion des tâches.

6.7 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'engendrera 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.

Profil projet

Matrice pour représenter un profil de projet

— min — max

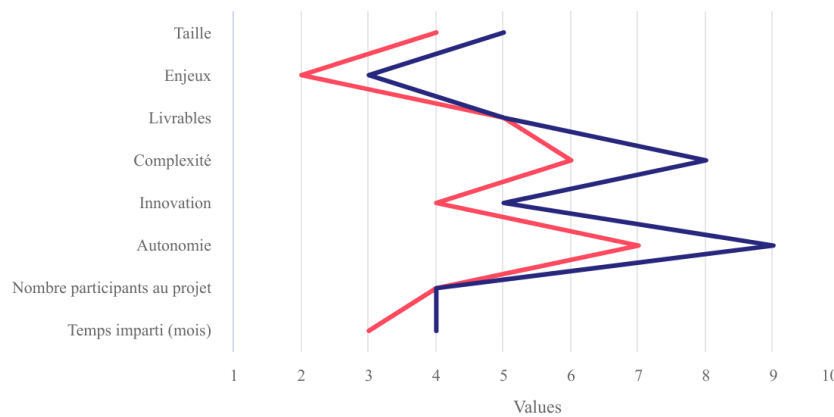


Figure 12: Profil du projet

6.8 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 apparaît ainsi les grandes étapes de notre projet que sont : définition du cadre de l'application, développement des fonctionnalités de l'application et écriture du rapport.

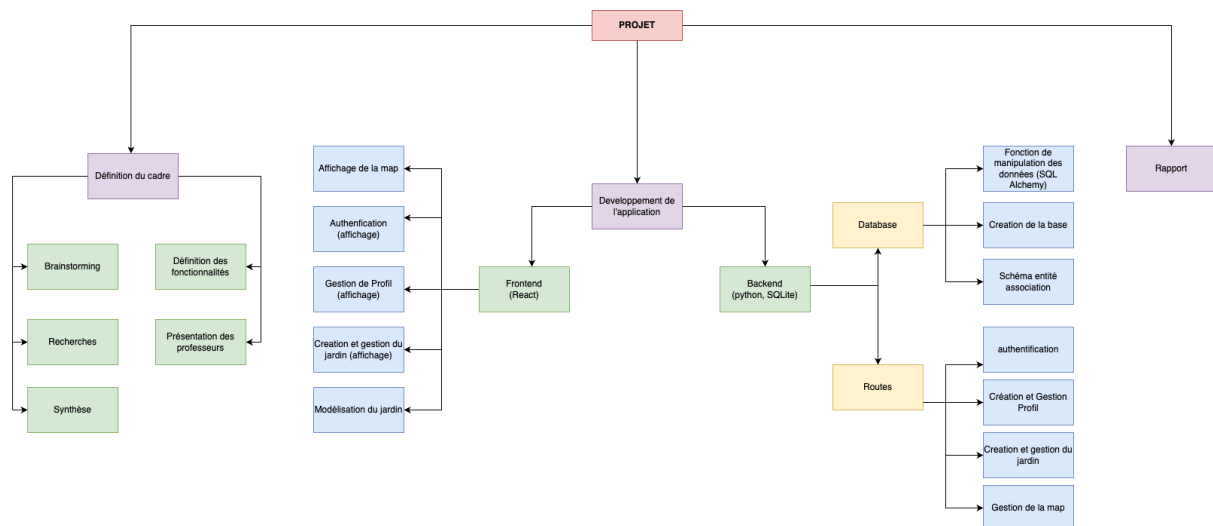


Figure 13: WBS

6.9 Diagramme de Gantt : planification

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

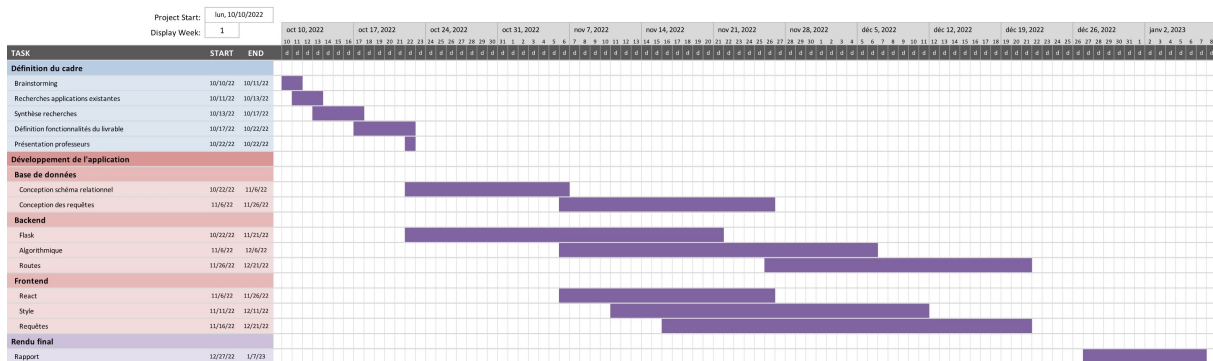


Figure 14: Diagramme de GANTT

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 réalisées.

6.10 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 15: Matrice RACI

6.11 Gestion des risques

Nous avons également pensé 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 compréhension 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 désintéresse 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'équipe est incompetent(e)	2	2	4	Noé Steiner	Réactualiser les connaissances nécessaires, et planifier des formations en fonction des besoins. Mettre à disposition les ressources nécessaires.	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 16: Plan de gestion des risques

7 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 bibliothèques 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 fiers du rendu final de notre travail et espérons retravailler ensemble sur d'autres projets, y compris externes à ceux donnés dans le cadre des cours.

8 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:
<ul style="list-style-type: none">• Alexis : Présent• Noé : Présent• Lucas : Présent• Mathias : Présent	<ul style="list-style-type: none">• 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 applications similaires présentes sur le marché (Tout le monde)
- Etablir une liste 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 lieu 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:
<ul style="list-style-type: none">• Alexis : Présent• Noé : Présent• Lucas : Présent• Mathias : Présent	<ul style="list-style-type: none">• 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 L^AT_EX 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 L^AT_EX (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 lieu 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:
<ul style="list-style-type: none">• Alexis : Présent• Noé : Présent• Lucas : Présent• Mathias : Présent	<ul style="list-style-type: none">• 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:
<ul style="list-style-type: none">• Alexis : Présent• Noé : Présent• Lucas : Présent• Mathias : Présent	<ul style="list-style-type: none">• 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 données 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 lieu le Vendredi 16 decembre 2022, de 20h à 21h.

Compte rendu de réunion

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

16 Décembre 2022

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

Motif / type de réunion:	Lieu:
<ul style="list-style-type: none">• Alexis : Présent• Noé : Présent• Lucas : Présent• Mathias : Présent	<ul style="list-style-type: none">• 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 profil

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 lieu le mardi 27 decembre 2022, de 14h à 15h.

Compte rendu de réunion

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

20 Décembre 2022

Projet PPII - Compte rendu n°06 - réunion technique

Motif / type de réunion:	Lieu:
<ul style="list-style-type: none">• Alexis : Absent• Noé : Présent• Lucas : Présent• Mathias : Absent	<ul style="list-style-type: none">• Le 20 decembre 2022• De 21h à 23h• Visioconférence sur Discord

Ordre du jour:

- Régler les problèmes d’affichages de la carte pour rejoindre un jardin

Compte rendu de réunion

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

23 Décembre 2022

Projet PPII - Compte rendu n°07 - réunion technique

Motif / type de réunion:	Lieu:
<ul style="list-style-type: none">• Alexis : Présent• Noé : Présent• Lucas : Absent• Mathias : Absent	<ul style="list-style-type: none">• Le 23 decembre 2022• De 21h à 23h• Visioconférence sur Discord

Ordre du jour:

- Mettre en place la solution de modélisation du jardin

Compte rendu de réunion

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

27 Décembre 2022

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

Motif / type de réunion:	Lieu:
<ul style="list-style-type: none">• Alexis : Présent• Noé : Présent• Lucas : Présent• Mathias : Présent	<ul style="list-style-type: none">• Le 27 décembre 2022• De 14h à 15h• Visioconférence sur Discord

Ordre du jour:

- Répartition des tâches à la suite du développement du projet
- Progression développement 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 du bon fonctionnement de l'application

Date de la prochaine réunion

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