



TELECOM Nancy

Rapport de Projet Pluridisciplinaire d'Informatique Intégrative

# PP2I - Etat de l'art : le numérique dans les circuits courts

BARBILLON Hélène, FORNOFF Léo, MANGOLD Mathis, TIGER Maelan,

Octobre 2022-Janvier 2023

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Contexte, objectifs, et buts . . . . .	4
1.2	Présentation de l'application réalisée . . . . .	4
<b>2</b>	<b>Etat de l'art : le numérique dans les circuits courts</b>	<b>5</b>
2.1	Exemples de circuits courts . . . . .	5
2.2	Plan de relance . . . . .	6
2.3	Description de plusieurs types de projets existants . . . . .	6
<b>3</b>	<b>Gestion de projet</b>	<b>7</b>
3.1	Communication et réunions . . . . .	7
3.2	Cahier des charges . . . . .	7
3.3	Matrice SWOT . . . . .	7
3.4	Profil du projet . . . . .	8
3.5	Organigramme des tâches . . . . .	9
3.6	Diagramme de Gantt . . . . .	10
<b>4</b>	<b>Gestion des données</b>	<b>12</b>
4.1	Schéma prévisionnel de la BD . . . . .	12
4.1.1	Réflexion sur la base de données . . . . .	12
4.1.2	Choix du type de base de données . . . . .	12
4.1.3	Modifications futures . . . . .	12
4.2	Création de la BD plantes et affinités . . . . .	12
4.2.1	Rappel des besoins . . . . .	12
4.2.2	Construction de la base de données . . . . .	13
4.3	Schéma final et détail de certaines données stockées . . . . .	13
4.3.1	Schéma de la base de données . . . . .	14
4.3.2	Tables SQL . . . . .	14
<b>5</b>	<b>Conception des algorithmes nécessaires</b>	<b>17</b>
5.1	Présentation générale . . . . .	17
5.2	Affichage des parcelles . . . . .	17
5.2.1	Contexte . . . . .	17
5.2.2	Traitement des données . . . . .	17
5.3	Affichage à l'utilisateur . . . . .	18
5.3.1	Utilisation brute de la matrice . . . . .	18
5.3.2	Problématique de lenteur . . . . .	18
5.3.3	Solution finale . . . . .	19
5.4	Algorithme de suggestion de plantes . . . . .	19
5.4.1	Construction de fonctions annexes et simples . . . . .	19
5.4.2	Algorithme de placement . . . . .	20
5.4.3	Inclusion sur le site . . . . .	21

---

<b>6</b>	<b>Application Web</b>	<b>22</b>
6.1	Préambule . . . . .	22
6.2	Organisation des fichiers . . . . .	22
6.3	Aspect visuel du site . . . . .	22
6.4	Gestion des demandes utilisateur par le serveur (Backend) . . . . .	24
6.4.1	Répartition des routes . . . . .	24
6.4.2	Gestion des sessions . . . . .	25
6.4.3	Page de connexion/inscription . . . . .	25
6.4.4	Liaison avec les autres fonctions . . . . .	26
6.4.5	Exemples de fonctions contenues dans Web/Fonctions . . . . .	27
<b>7</b>	<b>Tests et performance</b>	<b>28</b>
7.1	Test de la fonction <code>algo_placement</code> . . . . .	28
7.1.1	Premiers tests . . . . .	28
7.1.2	Deuxième série de tests . . . . .	29
7.2	Test de la fonction <code>string_to_lists</code> . . . . .	29
<b>8</b>	<b>Bilan</b>	<b>31</b>
8.1	Comparaison entre nos attentes de début de projet et ce qui a été réalisé . . . . .	31
8.2	Pistes d'améliorations . . . . .	31
8.3	Bilans individuels . . . . .	32
8.3.1	Liens pour le Python : . . . . .	35
8.3.2	Liens pour le CSS et HTML . . . . .	35
8.3.3	Liens pour l'état de l'art . . . . .	35
	<b>Annexes</b>	<b>36</b>

# Chapitre 1

## Introduction

### 1.1 Contexte, objectifs, et buts

"Le Bonheur Est Dans Le Jardin" est le projet que sur lequel nous avons travaillé dans le cadre du Projet Pluridisciplinaire d'Informatique Intégrative (PP2I) en première année d'école d'ingénieur à TELECOM Nancy. L'objectif est la mise en application des connaissances accumulées en cours, mais également l'apprentissage du travail en groupe. En effet, le projet s'inscrit dans la continuité du module Gestion De Projet et devient donc pertinent à analyser de ce point de vue là. La mise en application des connaissances représente quant à elle la partie technique du projet à réaliser et se décompose en trois volets principaux :

- Base de données
- Web
- Algo

Le sujet imposé pour le PP2I est : "Potager, Vergers, Circuits-Courts". C'est ainsi qu'a été imaginé "Le Bonheur Est Dans Le Jardin", une application constituant une solution innovante et pratique pour la gestion de jardins partagés.

### 1.2 Présentation de l'application réalisée

Notre application web se nomme "Le bonheur est dans le potager".

Elle permet à des jardiniers de gérer des parcelles qui leur sont attribuées dans un jardin partagé. Ils peuvent indiquer les plantes qu'ils cultivent, et peuvent recevoir des propositions pour agrandir leur potager en suivant un modèle de permaculture, c'est-à-dire en cultivant plusieurs types de plantes qui poussent bien ensemble. La partie suggestion de plante a deux variantes : une qui permet à l'utilisateur de choisir un emplacement et qui lui suggère une variété de végétal, et une autre variante qui propose à l'utilisateur de placer une plante, parmi les variétés qui seraient adaptées à ce qu'il a déjà dans le jardin. Les utilisateurs peuvent également consulter la page 'Dico des plantes' du site pour prendre connaissance des affinités entre les plantes.

Le fonctionnement général repose sur les jardins partagés, et les parcelles qui se situent dans ces jardins, dans lesquelles les utilisateurs peuvent ajouter des plantes. Les utilisateurs du site peuvent avoir plusieurs statuts : référent, administrateur, jardinier. Dans un jardin, il y a un utilisateur référent, c'est le propriétaire du jardin. Un jardin a des administrateurs (souvent le référent est aussi administrateur) qui peuvent créer des parcelles en indiquant leur taille (ou en supprimer), et les attribuer à des jardiniers. Ces jardiniers qui obtiennent donc des parcelles peuvent y cultiver des plantes.

Chaque utilisateur peut consulter les informations de son compte et choisir une photo de profil. Chaque utilisateur peut aussi voir les autres membres des jardins dans lesquels il jardine, afin de pouvoir contacter quelqu'un en cas de problème.

La suite de ce rapport vous présentera plus en détail ces fonctionnalités et leur implémentation.

## Chapitre 2

# Etat de l’art : le numérique dans les circuits courts

Les circuits courts permettent d’être plus respectueux de l’environnement en réduisant les temps de transport et en consommant des produits biologiques et de saison, tout en favorisant le producteur qui se voit rémunéré à un prix plus élevé.

Cet état de l’art a pour but de présenter différentes mises en application de ces circuits courts dans le cadre de la production et de la distribution alimentaire, en s’intéressant ensuite aux solutions numériques existantes.

### 2.1 Exemples de circuits courts

**Les AMAP** (Associations pour le Maintien de l’Agriculture Paysanne) sont des associations réparties sur toute la France qui créent un lien direct entre les producteurs et les consommateurs. Elles permettent au consommateur d’acheter des produits biologiques de saison directement auprès du producteur, sans passer par des intermédiaires polluants, avec un minimum d’emballages plastique, et en rémunérant le producteur à un prix plus juste. Contrairement à la grande distribution, tout ce qui est produit est vendu, il n’y a pas de sélection à la récolte, ce qui réduit le gaspillage alimentaire et conduit ici encore le producteur à faire plus de bénéfices. On peut notamment mentionner le club Marché de TELECOM Nancy qui propose aux étudiants des paniers de fruits et légumes biologiques de saison en partenariat avec l’AMAP de la MJC Lorraine.

**Les jardins partagés**, aussi appelés jardins communautaires sont des jardins conçus et cultivés dans le cadre d’une association gérée par les habitants d’un village ou d’un quartier. Ces potagers se situent le plus souvent sur des terrains mis à disposition par la commune. Ils permettent aux habitants du quartier ne possédant pas de jardin d’aller y cultiver des fruits, des légumes ou d’autres plantes pour se nourrir en respectant l’environnement. Mais ces jardins sont avant tout des lieux culturels de partage et d’apprentissage qui animent la vie d’un quartier. Ils jouent donc un rôle social, ainsi qu’un rôle de sensibilisation à l’environnement et au respect du vivant.

Certains de ces jardins partagés peuvent aussi créer des emplois et permettre une réinsertion sociale et professionnelle de personnes en difficulté. C’est notamment le cas des Jardins de Cocagne à Thaon-les-Vosges.

**“Les Jardins de Cocagne”**, situés dans la commune de Thaon-les-Vosges (ville de plus de 8000 habitants à 45min en voiture de Nancy), sont une production maraîchère biologique en circuit court, mais également un chantier d’insertion par l’activité économique. Cette association à but non lucratif créée en 1994 permet aux adhérents d’acheter des paniers de fruits et légumes biologiques cultivés sur la commune en respectant le rythme des saisons. La production s’étend sur 12 hectares, et compte une soixantaine de salariés dont une équipe permanente de 16 personnes et 44 salariés en contrat d’insertion. Les Jardins de Cocagne participent activement à la vie de la ville et jouent un rôle essentiel de sensibilisation à l’environnement et au respect de la nature. De nombreuses sorties scolaires y sont organisées par les écoles primaires et maternelles de la ville, et des animations y sont proposées tout au long de l’année. Ils accompagnent également des événements d’entreprises et de particuliers en proposant la buffets et de plateaux repas locaux.

## 2.2 Plan de relance

Lors de la crise sanitaire, les français se sont tournés vers les circuits courts pour s'alimenter. Le gouvernement a voulu entretenir cette dynamique en lançant des appels à projets en 2021 dans le cadre d'un plan de relance sous les noms "quartiers fertiles" et "jardins partagés". Ces appels à projets ont pour but de développer l'agriculture urbaine en encourageant la création de jardins partagés, de jardins d'insertions, de micro-fermes, et d'autres initiatives dans les zones urbaines et péri-urbaines. L'État y a investi 30 millions d'euros : 13 millions d'euros pour le projet "quartiers fertiles" et 17 millions pour les jardins partagés. Des aides, accessibles par des appels à projets au niveau départemental, seront accordées pour financer les investissements matériels et immatériels nécessaires au fonctionnement de jardins partagés, ainsi que des prestations de formation et d'accompagnement.

## 2.3 Description de plusieurs types de projets existants

Projet/Attributs	Principe	Plateformes	Statut	Echelle	Limites
Jardins Partagés	Prêt / location de terrains communaux pour faire des parcelles de potagers Subventionner l'installation de grandes jardinières par les syndicats d'immeuble	Visibilité en ligne sur les sites communaux	Projet communal	Moyenne échelle (entre plusieurs individus) Local	Chaque commune fait à sa manière Communication orale (présence obligatoire, de personne à personne) Peu de conseils Chacun fait son propre jardin
Application de jardinage	Reconnaissance de plantes et partage d'informations Conseils de jardinage	Application	Produit d'une entreprise de développement (but de rentabilité)	Réduite aux particuliers Potentiellement internationale (accessibilité langue)	Aucun échange, partage, contact
Fruit and Food	Mettre en relation l'utilisateur avec des personnes qui ne consomment pas tout ce que produit leur jardin	Application	Produit d'une entreprise (but de rentabilité)	Réduite aux particuliers Se distingue suivant les régions et l'implication des gens Nationale (potentiellement internationale)	Seulement de particuliers à particuliers (pas d'associations, de communes, etc. à pas de notion de partage de jardin)

## Chapitre 3

# Gestion de projet

Cette partie a pour but de présenter les outils de gestion de projet que nous avons utilisé et mis en place.

### 3.1 Communication et réunions

Afin de faciliter la communication dans le groupe nous avons mis en place dès le début un serveur discord organisé en plusieurs catégories. Nous utilisons également un drive partagé pour se partager des documents plus facilement, ainsi que overleaf pour la rédaction de documents.

Afin d'avoir une bonne vision d'ensemble du projet et pour pouvoir avancer de manière efficace et contrôlée, nous avons réalisé a minima une réunion par semaine avec tous les membres du groupe. Un certain nombre d'entre elles se sont avérées être des Stand-Up Meeting pour faire un bilan des avancées. Vous trouverez les comptes-rendus des autres réunions en annexe.

### 3.2 Cahier des charges

Un cahier des charges a été réalisé au début du projet, indiquant les fonctionnalités prévues pour l'application, les contraintes et les objectifs. Vous pouvez le consulter dans son entièreté en *annexe*.

### 3.3 Matrice SWOT

Au tout début du projet, lors de sa phase de définition, nous avons réalisé une matrice SWOT afin de faire le point sur les différents facteurs internes et externes qui peuvent affecter notre avancement, et ce afin de prévoir comment résoudre ces différents problèmes si ils se mesuraient à nous.

	Positif	Négatif
Origines internes	Forces : bonne marge de manoeuvre au début du projet, équipe dynamique et motivée	Faiblesses : manque d'expérience de chacun, risque de l'effet tunnel
Origines externes	Opportunités : retours d'expériences, équipe enseignante disponible, école qui ferme tard pour y travailler en groupe le soir	Menaces : accidents, pannes, imprévus, facteur temps

**Forces :** Notre équipe-projet est composée de membres dynamiques qui ont beaucoup d'idées et qui sont prêts à s'investir pour la réussite de ce projet. La séance de brainstorming que nous avons réalisée au début du projet

---

nous a permis de rapidement trouver une idée d'application originale pour pouvoir avancer. Nous profitons d'avoir une bonne marge de manoeuvre au début du projet et de ne pas avoir trop d'examens pour avancer le plus possible sur le projet.

**Faiblesses** : Aucun des membres de l'équipe n'a déjà participé à un tel projet. De plus nous n'avons pas ou très peu d'expérience dans le développement d'application WEB. Nous allons donc mettre à profit les cours de gestion de projet suivis au début d'année en mettant en place un maximum d'outils qui nous aideront tout au long du projet. Il est également important de rappeler la présence de l'équipe pédagogique si nous rencontrons des problèmes importants. Nous allons également être attentifs au moindre signe de création d'un effet tunnel, ce manque de visibilité et de communication qui peut mettre en péril tout un projet.

**Opportunités** : De nombreux étudiants de Télécom Nancy ont été à notre place et ont réalisé un projet PP2I. Les retours d'expérience, des étudiants de deuxième année tout particulièrement, peuvent nous être très bénéfiques. Ils peuvent nous conseiller sur les outils à utiliser et les erreurs à éviter. Par exemple, il nous a été conseillé d'utiliser Overleaf pour rédiger notre rapport. L'équipe enseignante est également disponible et à notre écoute en cas de problème. Nous avons par exemple sollicité M. Oster concernant le choix de la base de données. Nous avons la chance d'être dans une école qui ferme tard le soir, ce qui nous permet de travailler en groupe sur le projet après les cours.

**Menaces** : Nous ne sommes pas à l'abri d'une panne informatique à un moment clé du projet ou que l'un de nous se retrouve incapable de travailler suite à un imprévu ou un accident. C'est pourquoi nous travaillons au maximum sur des documents partagés accessibles par chaque membre de l'équipe-projet afin d'éviter de perdre des données, et nous essayons d'être tous polyvalents pour pouvoir pallier l'absence d'un membre dans le cas où il y aurait un problème.

Le temps est la ressource la plus importante du projet, nous savons qu'il va jouer contre nous, en particulier en décembre lors des partiels. Nous y portons donc une attention particulière notamment dans le diagramme de Gantt, et avons prévu d'avancer le plus possible sur le projet avant cette période chargée.

### 3.4 Profil du projet

Ce projet est un projet étudiant qui laisse une place assez grande à l'innovation bien qu'il existe déjà des applications sur les circuits courts et les jardins partagés, et que nous sommes limités par nos connaissances. Ce projet nous permet d'être très créatifs, nous devons créer une application innovante. Cependant nous ne sommes pas indépendants sur ce projet. Nous avons des contraintes à respecter comme par exemple les 3 volets imposés : base de données, algorithmie et web. Nous avons également des contraintes de temps. C'est un projet étudiant, et dans ce sens il n'y a donc pas de problématique de sécurité. L'échec de ce projet n'aurait une influence que sur nos notes. Pour finir, le budget accordé pour le projet est nul.



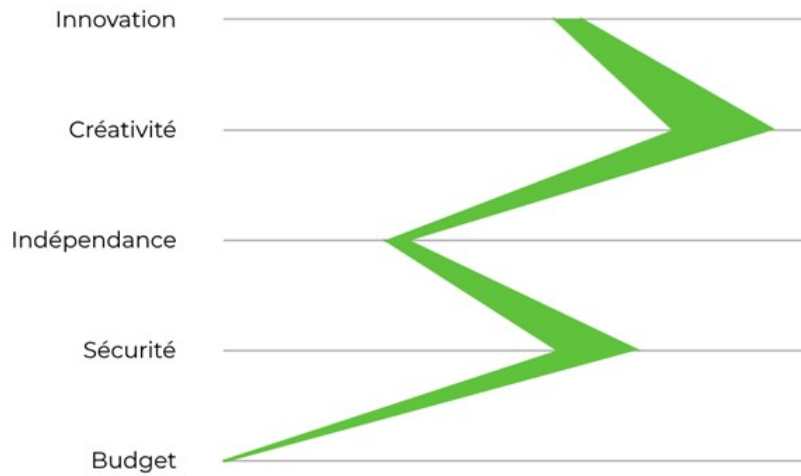


FIGURE 3.1 – Profil du projet

### 3.5 Organigramme des tâches

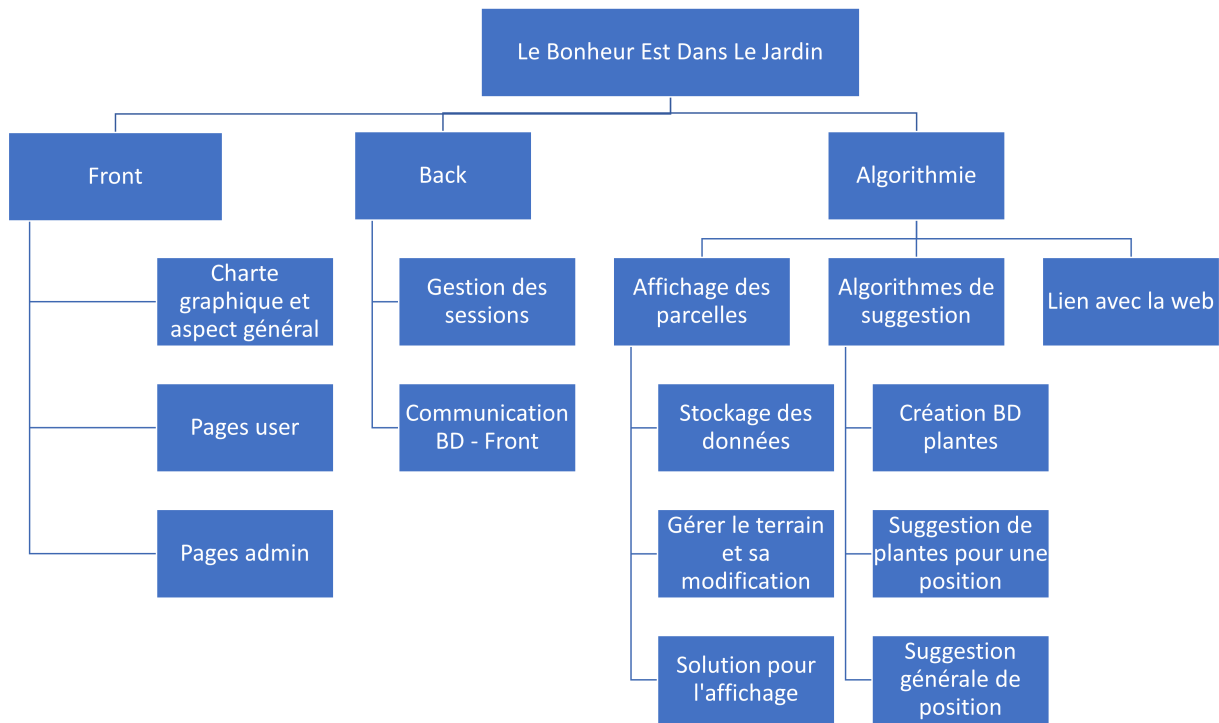


FIGURE 3.2 – Organigramme des tâches

### 3.6 Diagramme de Gantt

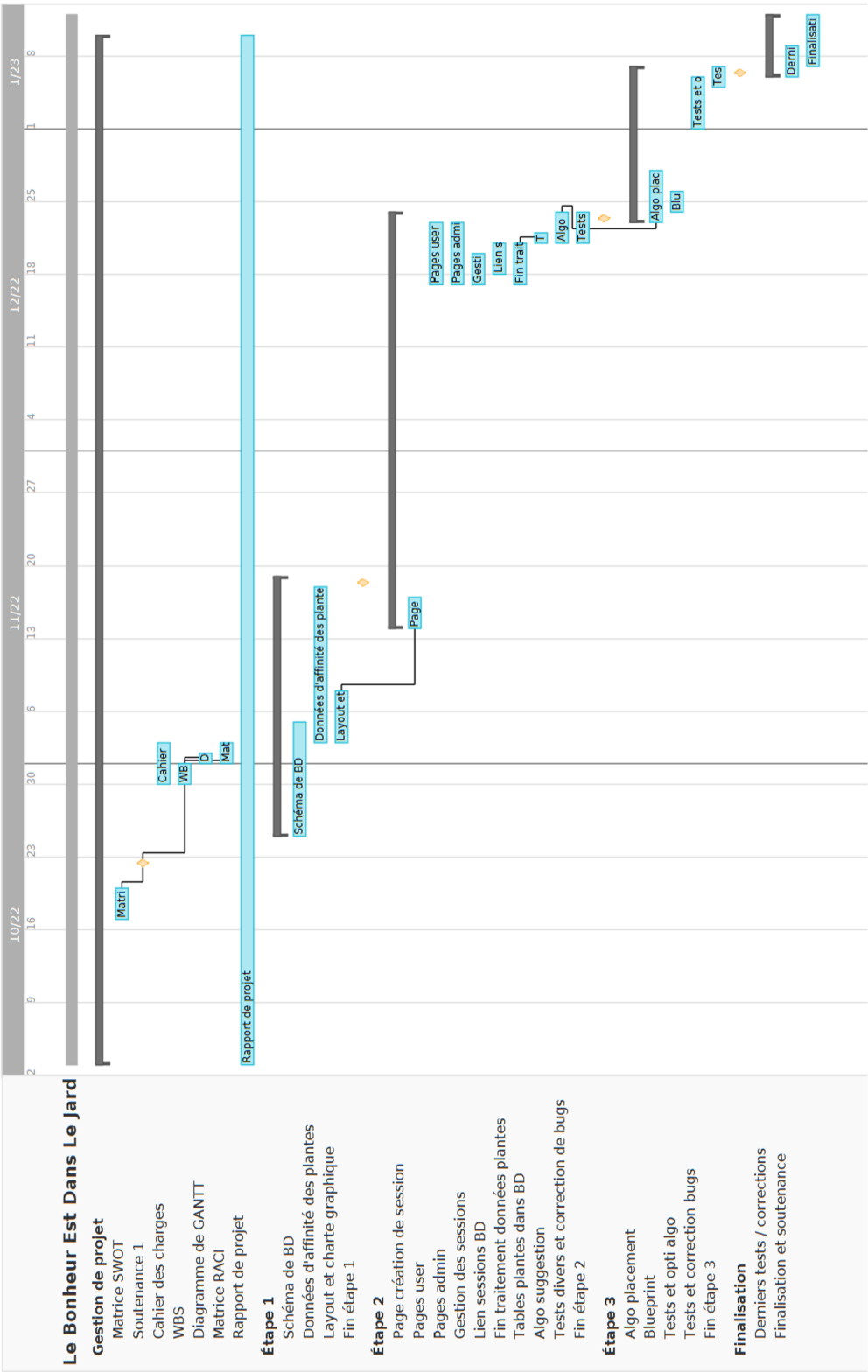


FIGURE 3.3 – Diagramme de Gantt

---

Le diagramme de Gantt a été réalisé à partir de l'organigramme des tâches avec le site TeamGantt. Les responsables des tâches n'apparaissent pas sur cette image mais étaient indiqués dans les notes.

# Chapitre 4

## Gestion des données

### 4.1 Schéma prévisionnel de la BD

#### 4.1.1 Réflexion sur la base de données

La base de données, véritable colonne vertébrale de l'application de par son omniprésence et son importance, devait être la source d'une réflexion précise et adéquate. En effet, c'est elle qui contient toutes les informations des utilisateurs, mais aussi celles des jardins et de leurs parcelles. Elle ne doit pas former de redondance d'informations et doit surtout respecter des contraintes d'accès et de modification simples.

#### 4.1.2 Choix du type de base de données

Pour gérer la base de données, nous hésitions entre deux outils vus en cours : SQLite et PostgreSQL. Le premier est une librairie plus simple d'utilisation, mais ne permet pas de modifier une base de données simultanément par plusieurs utilisateurs. Cependant, cet aspect n'est pas un problème pour notre projet qui n'a pas vocation à être implémenté sur le marché. Le second outil est plus complet, mais surtout plus complexe, et est intéressant à utiliser pour les projets de plus grande ampleur. Après avoir demandé conseil à un enseignant, notre choix s'est porté sur SQLite.

#### 4.1.3 Modifications futures

Un projet est un processus continu pendant une période de temps définie. C'est en ce sens que les choses sont vouées à être améliorées au fur et à mesure de son avancement. La base de données a donc été modifiée à plusieurs reprises pour répondre à des besoins mieux définis et définis différemment que ceux prévisionnels.

### 4.2 Création de la BD plantes et affinités

#### 4.2.1 Rappel des besoins

Afin de pouvoir afficher des plantes sur les parcelles de l'utilisateur, reconnaître informatiquement ces plantes et surtout en suggérer de manière pertinente à ce même utilisateur, il est nécessaire de constituer une base de données. En effet, il faut pouvoir identifier ces plantes, leur donner des caractéristiques et les manipuler.

Une première solution envisagée a été la recherche d'une base de données existante regroupant à la fois suffisamment de plantes et avec des caractéristiques intéressantes pour notre projet. De telles bases de données existent de manière plus ou moins libre sur internet, mais il est très compliqué d'en trouver. Les données dont nous avons besoin étaient des relations d'affinité entre les plantes. Ce type d'information est très spécifique et n'était donc pas référencé dans les bases de données trouvées. De plus, ces bases en libre accès offraient un panel très large de propriétés et d'attributs pour chaque plante. On se serait donc retrouvés noyés dans une mer d'informations et de noms latins sans réussir à en tirer quoi que ce soit. Finalement, la méthode de recherche de BD existantes a été écartée car trop chronophage pour des résultats inadéquats.

La problématique est donc devenue : comment créer une base de données un minimum représentative avec les informations dont nous avons besoin ?

---

## 4.2.2 Construction de la base de données

### Nouvelle session de recherches

Après avoir convenu ce nouvel objectif, il a donc fallu rechercher des données d'affinités, de compagnonnage de plantes (associations bénéfiques pour l'entraide mutuelle des plantes). La difficulté a résidé dans le fait de regrouper un grand nombre de plantes et de relations entre elles dans un ou des format(s) facilement analysable(s). Cela a également représenté une dépense de temps non négligeable. En fin de compte, nous nous sommes basés sur un document PDF à 3 colonnes, le seul fichier trouvé qui paraisse minimum exploitable : nom de la plante, plantes amies, plantes ennemies.

### Analyse et récupération des données

L'analyse des données du PDF a été faite en Python. Il existe de nombreuses bibliothèques dans ce langage pour l'extraction des données d'un document. Pourtant aucune n'était parfaite et chacune avait ses défauts. Étant donné que le fichier était un tableau, on pouvait parfois se retrouver avec une extraction de données faite ligne par ligne dans tout le document. On obtenait alors des mots coupés et on perdait définitivement tout moyen de regrouper les plantes avec la liste de leurs ennemies et de leurs amies. En effet, l'objectif était d'arriver à une liste totale contenant elle-même des listes intermédiaires. Ces listes intermédiaires relatives à une plante contiennent chacune le nom de la plante en question, la liste de ses amies et la liste de ses ennemies. L'extraction du PDF est donc passée par un grand nombre de tests et de corrections d'erreurs uniques (fautes dans des noms de plante, accents mal gérés, etc.). Avec du recul, il aurait peut-être été plus simple d'importer les données traitées dans un fichier CSV et ensuite de travailler dessus en sauvegardant. En effet, si l'on exécute actuellement le fichier `into_db.py`, le programme fera la lecture entière du fichier, son traitement, et la gestion des erreurs uniques. Ainsi, il n'y a pas moyen de modifier à la main les données et de garder ensuite le résultat en mémoire, celui-ci étant généré à chaque exécution. L'utilisation d'un CSV aurait facilité l'analyse des données car les erreurs de frappe auraient pu être traitées manuellement plutôt qu'en lignes python.

Une fois le traitement fait, il a fallu importer les informations dans la base de données. Nous avons fait le choix de la trier ainsi : une table "plante" qui associe un nom de plante, un id unique, une taille et une couleur ainsi que 2 tables "ennemis" et "amis" similaires qui mettent en relation 2 id. La couleur est une valeur RGB en hexadécimal générée aléatoirement à partir de l'identifiant de la plante. Cette couleur sera celle de la plante dans les images des parcelles. Quant à la taille de la plante, elle est par défaut de 20 (carré de 20x20cm). Il a par ailleurs été choisi de mettre une contrainte  $id2 > id1$  dans les tables "amies" et "ennemies". Cette contrainte permet d'éviter les redondances d'information.

## 4.3 Schéma final et détail de certaines données stockées

Vous pouvez trouver en annexe le *code de la base de donnée*.

### 4.3.1 Schéma de la base de données

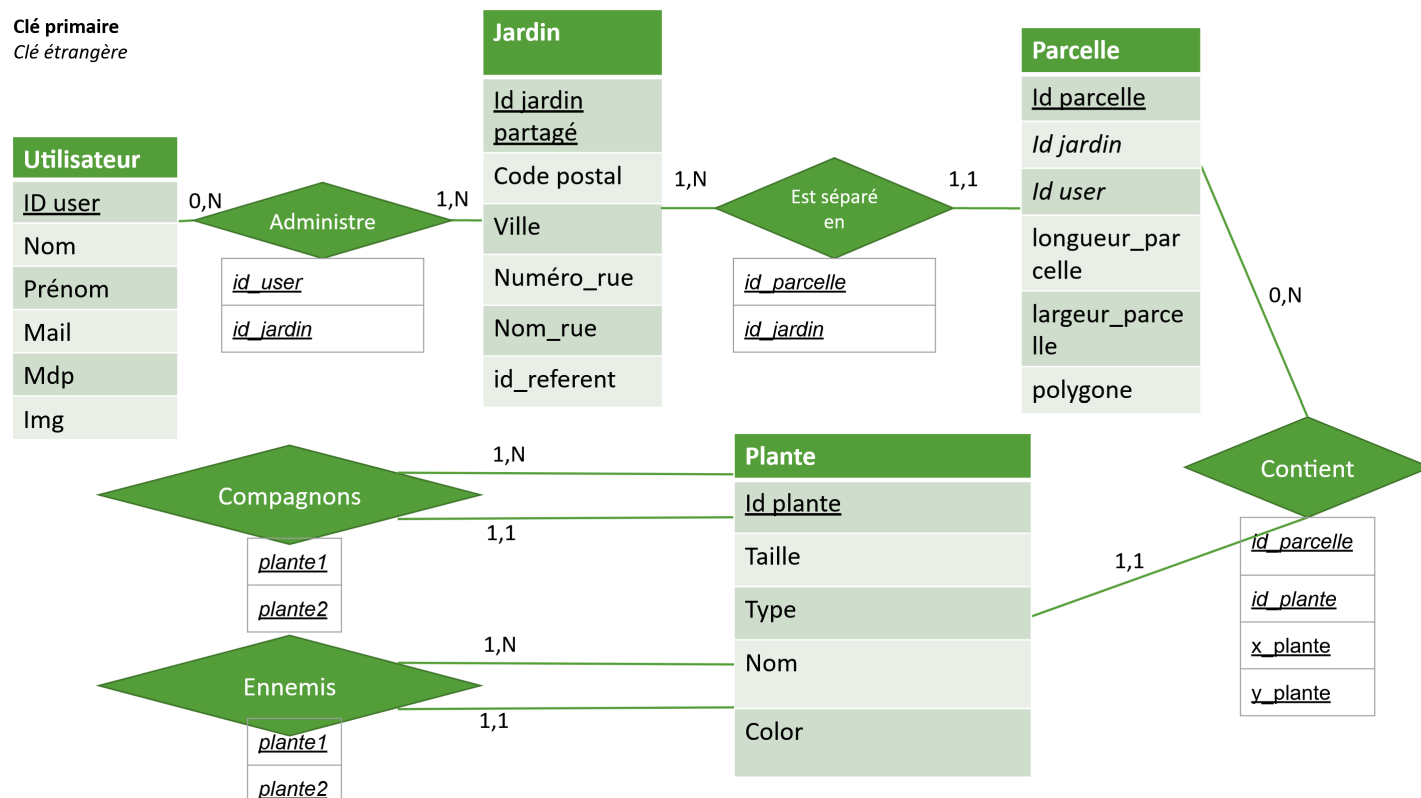


FIGURE 4.1 – Schéma de la base de données

### 4.3.2 Tables SQL

#### Table administre

id_user	id_jardin
---------	-----------

TABLE 4.1 – Table administre

Clé Primaire : id\_user, id\_plante

- id\_user : clé étrangère de la table utilisateur
- id\_plante : clé étrangère de la table plante

#### Table compagnons

plante1	plante2
---------	---------

TABLE 4.2 – Table compagnons

Clé Primaire : plante1, plante2

- plante1 : clé étrangère de la table plante (#id\_plante)
- plante2 : clé étrangère de la table plante (#id\_plante)

---

## Table contient

id_parcelle	id_plante	x_plante	y_plante
-------------	-----------	----------	----------

TABLE 4.3 – Table contient

**Clé Primaire : id\_parcelle, id\_plante, x\_plante, y\_plante**

- id\_parcelle : clé étrangère de la table parcelle (#id\_parcelle)
- id\_plante : clé étrangère de la table plante (#id\_plante)
- x\_plante : donne la position en x de la plante dans le jardin
- y\_plante : donne la position en y de la plante dans le jardin

## Table ennemis

plante1	plante2
---------	---------

TABLE 4.4 – Table ennemis

**Clé Primaire : plante1, plante2**

- plante1 : clé étrangère de la table plante (#id\_plante)
- plante2 : clé étrangère de la table plante (#id\_plante)

## Table jardin

id_jardin	code_postal	ville	numero_rue	nom_rue	id_referent
-----------	-------------	-------	------------	---------	-------------

TABLE 4.5 – Table jardin

**Clé Primaire : id\_jardin**

- id\_jardin : clé primaire auto-incrémenté mono-atomique
- code\_postal : code postal de l'adresse du jardin (integer)
- ville : ville du jardin
- numero\_rue : numéro dans la rue
- nom\_rue : nom de la rue
- id\_referent : clé étrangère de la table utilisateur (#id\_user)

## Table parcelle

id_parcelle	id_jardin	id_user	longueur_parcelle	largeur_parcelle
-------------	-----------	---------	-------------------	------------------

TABLE 4.6 – Table parcelle

**Clé Primaire : id\_parcelle**

- id\_parcelle : clé primaire auto-incrémentée mono-atomique
- id\_jardin : clé étrangère de la table jardin (id\_jardin)
- id\_user : clé étrangère de la table utilisateur (id\_user)
- longueur\_parcelle : longueur de la parcelle (dimension en x)

- 
- largeur\_parcelle : largeur de la parcelle (dimension en y)

## Table plante

id_plante	taille	nom	color
-----------	--------	-----	-------

TABLE 4.7 – Table plante

### Clé Primaire : id\_plante

- id\_plante : clé primaire auto-incrémentée mono-atomique
- taille : taille de la plante (carré de côté taille)
- nom : nom de la plante
- color : couleur pour l’affichage

## Table utilisateur

id_user	nom	prenom	mail	mdp	img
---------	-----	--------	------	-----	-----

TABLE 4.8 – Table utilisateur

### Clé Primaire : id\_user

- id\_user : clé primaire auto-incrémentée mono-atomique
- nom : nom de l’utilisateur
- prenom : prénom de l’utilisateur
- mail : mail de l’utilisateur
- mdp : mot de passe hashé de l’utilisateur
- img : image de profil de l’utilisateur



# Chapitre 5

## Conception des algorithmes nécessaires

### 5.1 Présentation générale

Le projet réalisé a dû répondre à des consignes bien précises parmi lesquelles l'utilisation de python pour de l'algorithmie. C'est ainsi qu'il a été décidé au début du projet de réaliser un algorithme qui suggère à l'utilisateur des plantes à mettre dans sa parcelle et à des positions elles aussi suggérées. Il a fallu également réfléchir à un moyen efficace pour gérer l'affichage des parcelles. La solution adoptée pour cela représente le deuxième volet de cette partie algorithmique.

### 5.2 Affichage des parcelles

#### 5.2.1 Contexte

Pour permettre la gestion de chaque parcelle par l'utilisateur et afin de le conseiller, il est primordial de trouver un moyen de stockage et de traitement efficace et réfléchi des informations d'une parcelle. L'aspect stockage a été traité dans le cadre de la gestion de la base de données puis modifié ensuite en fonction des besoins. De plus, il est nécessaire de pouvoir afficher de manière rapide et compréhensible le contenu de chaque parcelle.

**Détail des étapes suivies :**

- Traitement des données
  - Choix du type de données pour le traitement (différent des données)
  - Création d'outils pour faciliter le traitement
- Affichage à l'utilisateur
  - Utilisation "brute" depuis la matrice
  - Solution plus optimisée

#### 5.2.2 Traitement des données

##### Choix du type de données à manipuler

Nous avons décidé d'utiliser des matrices numpy pour représenter la parcelle d'un utilisateur. Pour se faire, on discrétise le terrain avec une échelle définie (ici : choix de l'échelle 1 case = 1cm). On peut ainsi mettre dans chacune des cases de la matrice, la valeur correspondant à l'identifiant `id_plante` de la plante présente.

La matrice est donc une représentation assez directe du terrain qui lui correspond. Il est possible de modifier cette matrice lorsqu'une modification de la parcelle est faite. Et il est possible de consulter la matrice pour connaître la disposition des plantes dans le terrain.

De plus, il est facile de la construire à partir des informations contenues dans la base de données. En ayant les plantes et leurs positions respectives dans un terrain, on peut facilement générer la matrice correspondante.

Pourtant ce choix de données n'est pas optimal dans toutes les situations, notamment lorsqu'il s'agit d'afficher les informations d'un terrain. Une matrice de 1000x1000 est difficilement représentable sur un écran sans traitement graphique.

---

De même, il n'est pas toujours facile de tester ses algorithmes sans avoir recours à des outils graphiques afin de visualiser le rendu et les possibles erreurs.

## Utilisation d'une classe pour gérer une parcelle

Afin de simplifier la création mais surtout la manipulation et l'affichage d'une matrice représentant une parcelle, nous avons créé un objet Terrain et des fonctions qui lui sont propres.

Cet objet permet de créer une matrice correspondant à un terrain d'une certaine taille et avec une certaine adaptation d'échelle (toujours 1cm par case pour l'échelle standard).

Il est ensuite possible d'ajouter ou de supprimer des plantes dans la matrice. Plante définies par leur coin en haut à gauche, par leur id et par leur taille. L'objet se charge de vérifier que l'ajout ou la suppression est possible. Si jamais ce n'est pas le cas, la modification n'est pas réalisée et un message d'erreur est retourné par la fonction de l'objet.

Pour gérer tout cela, la classe repose sur une fonction clé : `modification_terrain`. Cet algorithme n'est jamais appelé directement mais est au coeur de beaucoup d'autres fonctions. Il permet de modifier de manière directe les données de la matrice. Son mode de fonctionnement repose sur l'utilisation d'une copie de la matrice avant de retourner cette copie suivie d'un booléen de vérification et d'un message. Si le booléen est à True, le processus s'est déroulé sans problème et on peut remplacer la matrice `self.mon_terrain` de la classe par celle retournée par la fonction. Dans le cas contraire, la variable de classe `self.mon_terrain` n'est pas modifiée et le message d'erreur est retourné, comme décrit dans le paragraphe précédent.

Mais alors comment définir le booléen de réussite si la fonction modifie la matrice de manière brute et directe ?

Rappelons tout d'abord que ajouter une plante, c'est mettre dans la matrice la valeur `id_plante` entre `(x_plante, y_plante)` et `(x_plante+taille, y_plante+taille)`. Pour cela, on crée une matrice ligne de longueur `taille` composée uniquement de la valeur `id_plante`, puis on remplace ligne par ligne en incrémentant de 1 à chaque fois. La vérification intervient quant à elle tout au long de ce processus. On récupère la ligne que l'on va modifier avant de le faire pour la comparer avec la matrice nulle de même taille. Si ça concorde, la ligne était libre. Sinon, cela veut dire qu'il y avait une plante : le booléen passe à False et le code d'erreur est généré.

## Exemple d'application

Nous avons utilisé dans plusieurs cas cet objet :

- Pour créer une matrice de parcelle à partir des informations sur les plantes dans la base de données.
- Dans l'algorithme de suggestion des plantes afin de vérifier que la suggestion est bien valide.

## 5.3 Affichage à l'utilisateur

### 5.3.1 Utilisation brute de la matrice

Il a rapidement été nécessaire de pouvoir afficher la matrice de parcelle puisque la visibilité sur un terminal avec des nombres s'avère très réduite.

Ainsi un programme provisoire d'affichage a été créé. Pour chaque élément de la matrice, il crée un bloc dans une grid HTML puis lui associe une couleur qui lui est propre.

### 5.3.2 Problématique de lenteur

Cette méthode d'affichage a de nombreux avantages comme la facilité de mise en place et sa simplicité. Mais elle possède aussi un gros défaut, la taille en du code HTML. En effet, il n'est pas du tout efficace d'afficher des milliers de cases si ce n'est des centaines de milliers. Cela entraînait de la latence chez l'utilisateur, notamment lors du redimensionnement de la page.

Une solution simple à ce problème est de créer préalablement une image puis de l'afficher à l'utilisateur. Mais cela ne permet pas d'avoir de l'interaction avec ce dernier.

---

### 5.3.3 Solution finale

L'objectif est d'avoir un affichage rapide et dynamique, mais également un affichage qui permet une meilleure accessibilité par l'utilisateur.

#### Map et area en HTML

Suite à des recherches sur internet, nous nous sommes inspirés d'une méthode qui est utilisée pour rendre des cartes cliquables. Cette méthode se base sur un ensemble de points qui représentent donc les sommets du polygone. En reliant l'ensemble de ces sommets, on obtient bien le contour de la figure.

Il devient ainsi réaliste de rendre cliquable l'entièreté de la map en créant la bonne quantité de zones à partir de leurs sommets. Pour cela, on se base sur un algorithme de calcul des contours.

#### Algorithme de contour

Afin de mettre en application cette fonction d'affichage de la map, il nous a donc fallu créer un programme qui extrait, pour chaque polygone de la matrice, ses sommets. Une zone est donc définie comme la plus grande surface continue possible contenant la même plante.

#### Principe de fonctionnement

Le principe est le suivant : on démarre arbitrairement en haut à gauche puis on parcourt le contour en se déplaçant toujours au maximum vers le bas et la droite. Les tournants sont gérés de manière très simple suivant les différents cas. Ainsi, l'algorithme sait précisément lorsqu'il doit tourner et dans quel direction afin de continuer à suivre le contour.

On répète ensuite la méthode sur tout le terrain de gauche à droite et de haut en bas. L'algorithme teste d'abord si le point en haut à gauche appartient déjà à un contour avant de lancer l'analyse d'un nouveau contour.

Et c'est ainsi que sont créés tous les polynômes d'une parcelle. A chaque fois, on traite les contours qui n'ont pas encore été traités et on avance dans la matrice jusqu'à avoir parcouru l'entièreté de la matrice.

Une fois que l'on possède tous les points de tous les contours, on y applique un nouveau programme qui va permettre de simplement retrouver les sommets de tous ces contours.

Le résultat de ces deux étapes est ensuite utilisé pour dessiner l'image du terrain ainsi que pour faire les zones de area (HTML).

Pour afficher la map à partir d'un polygone, on utilise une fonction qui sert à créer l'image à partir de quelques informations.

#### Problème de temps de calcul

Contrairement au premier processus d'affichage, celui-ci est beaucoup plus léger pour le client mais bien moins pour le serveur. Même si le processus de calcul pour des matrices de (1000x1000) peut prendre un temps de l'ordre de la seconde. Il est ainsi préférable de stocker l'information des polygones permettant la génération de l'image et de ne pas les recréer tant que la parcelle n'a pas été modifiée.

## 5.4 Algorithme de suggestion de plantes

### 5.4.1 Construction de fonctions annexes et simples

#### Point de démarrage

Le but de cette partie d'algorithmie était donc de réaliser une fonction de placement et de suggestion de plantes à l'utilisateur. La base de données contient pour une parcelle donnée la liste des végétaux qui y sont plantés et leurs

---

coordonnées. Et bien sûr il va falloir s'appuyer sur les tables "amis" et "ennemis" pour trouver les plantes que l'on va conseiller.

### Prévoir les besoins

Le choix a ensuite été fait de travailler sur les ID des plantes pour éviter toute complication de lecture dans la base de données. Or il est préférable en termes de compréhension que les fonctions de suggestion renvoient des noms de plantes à l'utilisateur plutôt que des ID. Ainsi il a fallu prévoir des fonctions très simples mais nécessaires de conversion (`id --> nom`) et (`liste_id --> liste_noms`) et inversement.

La suite de la construction a été dédiée à des fonctions renvoyant les plantes amies (respectivement ennemies) d'une plante, puis la liste des amies (resp. ennemies) d'une liste de plantes. Pour enfin créer une dernière fonction qui va, à partir d'une liste de plantes en entrée, retourner les plantes amies communes à cette liste d'entrée.

Il a également été utile de créer une fonction de génération de valeurs pour les tests qui a pour but d'ajouter à la base de données un certain nombre de plantes aléatoires à des positions aléatoires dans une parcelle.

### Algorithme de suggestion pour une coordonnée

Après cette préparation faite, il devient très simple de faire un algorithme qui suggère des plantes pour une certaine position donnée. A noter que l'algorithme ne vérifie pas si il est possible de planter quelque chose à ces coordonnées ou si elles sont déjà prises. Il se contente de retourner une liste de suggestions.

Le fonctionnement est très simple et se décompose en 3 grandes étapes :

- Récupération des informations sur la parcelle (plantes et coordonnées)
- Utilise la fonction `coords_voisins()` créée pour l'occasion afin de récupérer les plantes dans un rayon de 3 mètres autour de la position analysée
- Retourne la liste des amies des plantes présentes dans ce rayon de 3m

#### 5.4.2 Algorithme de placement

##### Premier jet non optimisé, approche brute

La première version de l'algorithme de placement avait pour but d'organiser la structure du programme. C'est en ce sens que la boucle principale du programme n'a pas été optimisée (double `for` sur la longueur et la largeur de la matrice puis encore des tests sur la liste complète des résultats). La fonction se contentait de créer une liste contenant toutes les positions sans plantes et en tirait toutes les positions libres d'une certaine taille. Ensuite le programme cherchait toutes les plantes amies et ennemies de pour ces positions et un pick aléatoire était fait pour suggérer une plante et une position.

### Solutions envisagées et appliquées

Plusieurs solutions se sont offertes pour l'optimisation de l'algorithme. Le but étant de faire tendre la complexité vers une courbe de type  $n^2$ . Tout d'abord, une idée était de chercher d'abord les positions libres et hors d'influence d'autres plantes (rayon de 3m). En effet, si une telle position existe, alors on peut suggérer à l'utilisateur d'y mettre n'importe quelle plante de la base de données. Ensuite, le principe serait le même : récupérer les positions libres de la bonne taille. Mais au lieu de toutes les récupérer, trier celles qui sont concernées par les mêmes suggestion de plantes. En travaillant ainsi sur les zones d'influence des plantes voisines, on peut éliminer des zones entières de la matrice à tester. Et on se retrouve donc avec un nombre limité de tests à effectuer. Cette solution n'a pas été adoptée car trop compliquée à mettre en place pour un résultat possiblement moins optimisé que l'original. De plus il aurait fallu travailler énormément sur la matrice et les requêtes Python dans les listes sont plutôt longues à être traitées.

L'autre solution envisagée était de ne pas travailler sur les position vides mais sur les plantes. En effet, on possède directement de la base de données la liste de toutes les plantes avec leurs coordonnées respectives. Ainsi, en parcourant simplement les plantes de la parcelle, on peut obtenir une boucle principale plus courte et plus optimisée. Une amélioration serait encore de travailler sur la liste la plus courte entre la liste des positions vides et la liste des plantes. Cette solution est celle pour laquelle nous avons opté, mais en travaillant tout le temps sur les plantes

---

et jamais sur les positions vides. En effet, cela simplifie la construction de la fonction et n'enlève qu'un cas très improbable. En effet il faudrait plus de 1250 plantes sur un jardin de 1000x1000 pour avoir moins de positions libres que de positions occupées. De plus il est très simple de calculer les positions des sommets de chaque plante pour ensuite analyser les voisins les plus proches et en déduire des positions adéquates pour ajouter une plante. Par ailleurs, une autre optimisation qui a été faite est la création d'un dictionnaire au lieu d'une liste. En effet, rechercher de l'information dans un dictionnaire est plus optimisé que de parcourir une liste. Ce dictionnaire possède comme clés des id de plantes, associées à des couples de coordonnées auxquelles cette plante peut être ajoutée. Cela permet de récupérer plus rapidement les suggestions de positions que si on devait parcourir une liste.

## Limites

L'algorithme n'est cependant pas parfait et a des limites évidentes qui n'ont pas pu être corrigées ou modifiées. En effet, la fonction récupère des coordonnées entre chaque plante et ses voisins respectifs. Coordonnées auxquelles pourraient être mise une nouvelle plante. Ainsi, les positions proposées ne sont pas particulièrement variées. De plus, si les plantes du jardin sont disposées dans un coin ou en carré, l'algorithme ne détectera aucune position de libre. En effet, il aurait fallu, pour corriger cela, travailler sur les plantes tout en effectuant en amont un scan des positions libres du potager.

Une autre limite majeure est la vérification des positions. L'algorithme ne vérifie en aucun cas si la plante peut être posée aux coordonnées suggérées. C'est lorsque la page HTML reçoit une requête sur les positions suggérées pour une plante que le test est réalisé. Ce choix a été fait afin de gagner du temps et d'éviter une redondance. En effet la classe terrain possède déjà une fonction de modification qui vérifie l'accessibilité d'une position avant d'y ajouter une plante. Ainsi il suffit de faire la requête d'ajout de plante à la position donnée et analyser le résultat de la fonction du terrain. Si ce n'est pas valide, on teste avec les autres positions fournies.

### 5.4.3 Inclusion sur le site

Une fois les algorithmes réalisés, il a fallu les inclure sur l'application WEB et gérer leur accessibilité par l'utilisateur.

## Problématiques générales

L'inclusion à la page WEB passe par l'importation des fonctions nécessaires. Mais cela ne suffit pas forcément au bon fonctionnement des algorithmes. En effet, il s'est avéré que les appels à la base de donnée ne fonctionnaient pas correctement. L'algorithme Python étant appelé par un autre fichier, l'ouverture de la base de données dans le fichier source n'est pas possible. Il a alors fallu ajouter un paramètre d'entrée "database" à chaque fonction et donc ajouter également ce paramètre à chaque appel de chaque fonction afin de permettre la circulation de la base de données. C'est une erreur apparemment classique qui n'a pas été prévue en amont. Ce genre de problématique permet d'accroître son expérience et donc d'éviter à l'avenir que la situation ne se reproduise.

L'autre problématique générale a été, comme mentionné précédemment, le test des positions disponibles pour accueillir une nouvelle plante. Il a fallu tester dans la fonction de la route si l'ajout de plante aux positions suggérées était réalisable. Il aurait certainement été plus judicieux dans un soucis de clarté de ces fonctions de routes de faire un algorithme complet à appeler pour ajouter une plante. De même pour la suite d'instructions à réaliser pour générer une image, etc.

## Adaptation au fichier

Lors de l'importation du volet algorithmie, une autre problématique importante a été l'organisation. Il a fallu chercher comment greffer au mieux ces nouvelles fonctionnalités contraignantes à l'arbre déjà complexe que forme l'ensemble des routes. Ainsi, il a été décidé de rester sur une seule et même route et d'y inclure une variable "type" qui indique si il faut afficher les suggestions de placement, les suggestions de plantes, etc. Durant cette étape il a fallu également gérer d'autres problèmes annexes comme par exemple la communication du dictionnaire complet à travers la page HTML puis à travers la redirection. Cela a nécessité la manipulation d'un dictionnaire dict vers str puis de nouveau vers dict.

## Chapitre 6

# Application Web

### 6.1 Préambule

L'application web est gérée par le module Flask de Python qui utilise des pages en django HTML. Elle constitue le coeur du projet PP2I puisqu'elle permet d'articuler l'interface utilisateur, la base de donnée et l'algorithme.

### 6.2 Organisation des fichiers

Représentation simplifiée de l'arbre des dossiers du projet :

- **Web** (Les fichiers de l'application et le document pour lancer l'application)
  - **App** Les fichiers de l'application
    - **Database** (Les fichiers relatifs à la base de donnée)
    - **Fonctions** (Toutes les fonctions utilisées par le site web)
    - **Main** (Les fichiers de routes)
    - **Static** (Les fichiers et documents static comme le css ou les logos)
    - **Template** (Les templates HTML)
  - **Flask\_session** (Les fichiers de session)

**Lancement de l'application :** l'application est lancée grâce au fichier **run.py** présent dans le répertoire Web. Ce fichier fait appel au fichier **\_\_init\_\_.py** qui utilise le sous-module Flask **Blueprint**. Ce dernier permet de répartir les routes du site web entre plusieurs fichiers qui sont présents dans le dossier **main**.

**Le module Blueprint** permet de répartir le contenu d'un fichier Python entre plusieurs autres fichiers, ce qui les rend plus lisibles. Cela nous fait gagner du temps lorsqu'on cherche une fonction spécifique dans un fichier, ou une erreur.

Nous avons cependant rencontré quelques difficultés à ajouter ce module à notre application, notamment à cause de problèmes de portabilité de la base de données SQLite, ainsi que de nos faibles connaissances sur les modules Python.

### 6.3 Aspect visuel du site

Cette partie présente les principales pages du site, dont le contenu s'adapte en fonction de la taille de l'écran. Toutes ces pages utilisent des données de session de l'utilisateur et font des requêtes à la base de donnée. Vous pouvez voir les pages mentionnées en *annexe*.

## Routes du site (hors requête POST des formulaires)

Tableau :

Fichier	Route	Commentaire
Admin	/admin/attribution_parcelles	Zone d'administration des parcelles
Authentification	/signin	Page d'inscription et de connexion
	/register	Page d'inscription
	/login	Page de connexion
	/logout	Page de déconnexion
Main	/status	Page "Up and Running"
	/	Page d'accueil
Plante	/id_plante/<numero>	Renvoie à la page du dictionnaire associée au numero de la plante
User	/users	Montre les informations des jardiniers de l'application
	/users/<numero>	Affiche les informations des jardiniers d'un certain jardin
	/mesparcelles	Affiche les parcelles de l'utilisateur
	/mesparcelles/<numero>	Affiche les informations d'une certaine parcelle
	/user/vos_informations	Affiche les informations de l'utilisateur
	/changePhoto	Affiche les différentes photos de profil disponibles à l'utilisateur
	/dico/<num>	Affiche la page des amis/ ennemis d'une certaine plante

TABLE 6.1 – Tableau des routes du site

## La barre de navigation

Pour l'aspect général du site, nous avons choisi des tons verts qui rappellent le jardinage. Une barre de navigation (que l'on appellera navBar) est présente sur chaque page du site pour pouvoir naviguer facilement.

On peut y voir le nom du site ainsi qu'un message invitant l'utilisateur à se connecter si ce n'est pas déjà le cas :

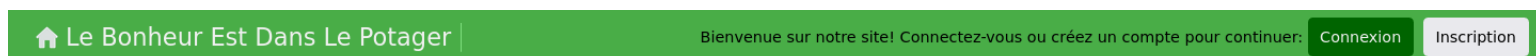


FIGURE 6.1 – Barre de Navigation au démarrage

Lorsque l'utilisateur est connecté, il a accès aux différents onglets :

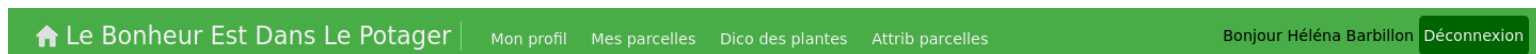


FIGURE 6.2 – Barre de Navigation une fois connecté

Nous avons également prévu un réajustement des icônes lorsque la taille de la page diminue, avec un menu déroulant :



FIGURE 6.3 – Barre de Navigation réduite avec menu déroulant

## La page "Mon profil"

Elle permet aux utilisateurs de voir les **informations de leur compte** : nom, prénom, email, identifiant de jardinier. Les utilisateurs peuvent aussi **changer leur photo** de profil (en choisissant entre plusieurs images). L'utilisateur peut visualiser les informations des parcelles dans lesquelles il jardine en voyant leur identifiant et leur taille. Il peut également choisir de **visualiser tous les jardiniers** du jardin dans lequel se trouve sa parcelle.

## La page "Mes parcelles"

Cette page permet de lister les parcelles de l'utilisateur et d'afficher les plantes qui s'y trouvent. L'utilisateur peut ensuite gérer chaque parcelle individuellement sur une autre page dédiée. C'est sur cette dernière que se trouvent les outils de suggestion de plantes.

## La page "Dico des plantes"

Cette page permet de trouver les affinités d'une plante avec d'autres, pour aider les jardiniers. Elle permet d'afficher directement les informations de la database.

## La page "Attrib parcelles"

Cette page n'est accessible que pour les comptes ayant le rôle d'administrateur, c'est-à-dire ceux gérant un jardin. Sur cette page, ils peuvent ajouter et supprimer des parcelles dans les jardins qu'ils administrent et les attribuer à des jardiniers.

## La page d'affichage des jardiniers

Cette page permet d'afficher la liste des jardiniers, administrateurs et référents au sein d'un jardin, et donne les informations de contact de ces personnes.

Cette page utilise un court script de JS pour afficher les profils.

## 6.4 Gestion des demandes utilisateur par le serveur (Backend)

### 6.4.1 Répartition des routes

Les routes ont été réparties en 5 Blueprints fonctionnant de manière analogue à l'utilisation de la route par défaut "app". Ainsi on débloque des routes différentes comme par exemple "@nomDuBlueprint.route('/maRoute')" et l'acheminement des requêtes au bon endroit est géré par la librairie.

**Nom des différentes routes (.py pour le nom du fichier correspondant) :**

- admin
- authentication

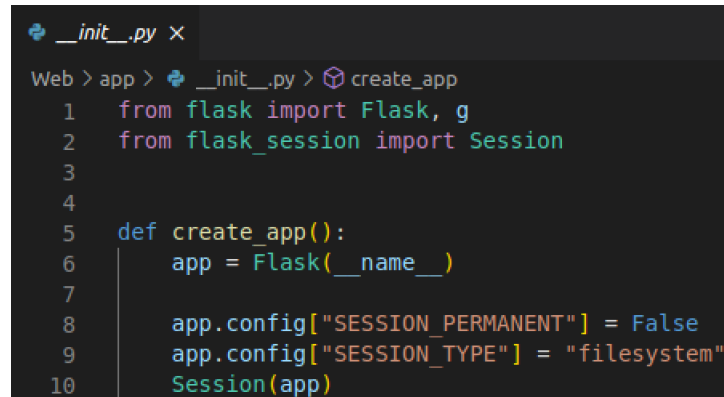


- 
- main
  - plantes
  - user

La répartition des routes est intuitive puisqu'elle suit le nom du fichier. Une requête relative à un administrateur sera donc traitée dans le fichier `admin.py`.

### 6.4.2 Gestion des sessions

Les sessions sont gérées avec l'extension `flask_session`. Elles sont initialisées dans le fichier `__init__.py` :



```
__init__.py x
Web > app > __init__.py > create_app
1  from flask import Flask, g
2  from flask_session import Session
3
4
5  def create_app():
6      app = Flask(__name__)
7
8      app.config["SESSION_PERMANENT"] = False
9      app.config["SESSION_TYPE"] = "filesystem"
10     Session(app)
```

FIGURE 6.4 – Initialisation des sessions

Elles sont gérées avec un dictionnaire "session" qui est rempli à chaque connexion avec les données de la base de données, puis vidé à chaque déconnexion. Ce dictionnaire session comprend :

- `session["email"]` : adresse mail de l'utilisateur
- `session["name"]` : prénom & nom de l'utilisateur
- `session["id_user"]` : identifiant de l'utilisateur
- `session["admin"]` : "oui" si l'utilisateur est administrateur (gère un jardin) , sinon vide
- `session["num_jardin_a"]` : [id\_1, id\_2, ...] : renvoie les identifiants des jardins dans lesquels l'utilisateur est admin
- `session["parcelles"]` : [id\_1, id\_2, ...] : renvoie les id des parcelles que l'utilisateur gère

Ces sessions permettent de réguler l'accès à certaines pages du site, et de les adapter à l'utilisateur (en affichant uniquement les parcelles de l'utilisateur et pas celles des autres par exemple).

### 6.4.3 Page de connexion/inscription

Trois routes permettent d'accéder aux formulaires de connexion et d'inscription `/signin`, `/register` et `/login`. Ces trois routes affichent respectivement : le formulaire de connexion, le formulaire d'inscription et les deux. Elles utilisent le même template html. Avoir ces 3 affichages différents nous permet de rendre la page plus responsive.

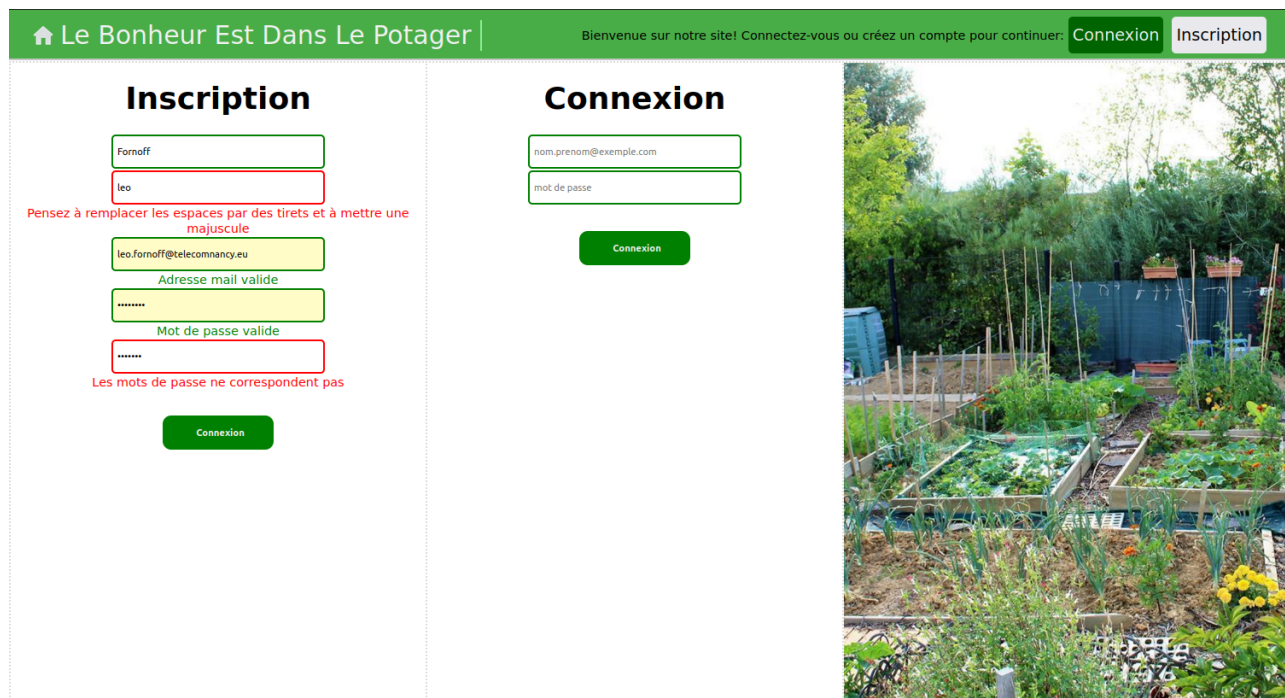


FIGURE 6.5 – Page de connexion /login

Cette image ci-dessus montre les deux formulaires. Chacun des formulaires utilise un script java-script afin de signaler à l'utilisateur que sa saisie est valide ou invalide. Pour des raisons de sécurité, des tests similaires sont réalisés dans le backend de l'application par des fonctions python.

Chaque saisie est vérifiée par des expressions régulières. Pour qu'une saisie soit valide, elle doit remplir plusieurs conditions :

- les mot de passes doivent contenir X caractères, au moins une majuscule, un chiffre, un caractère spécial.
- Les adresses mails doivent respecter un format d'adresse mail
- les noms et prénoms doivent commencer par des majuscules et ne pas comporter de ....

La fonction **register()** contenue dans le fichier `register.py` dans `Web/Fonction/login` permet de vérifier que les données envoyées par un utilisateur souhaitant s'enregistrer sont valides. La fonction vérifie donc chacun des champs rentrés par l'utilisateur et s'assure que les entrées correspondent aux caractéristiques demandées.

Son fonctionnement : La fonction demande en argument les valeurs des champs récupérés par une route via une méthode POST. Ensuite pour chacun des champs elle vérifie que le format de la donnée est bon en faisant un ou des test d'expressions régulières. Si l'un des tests ne passe pas la fonction retourne un code d'erreur et le message correspondant à l'erreur. Et si tout les tests sont passés, elle valide la sélection (renvoie le booléen True).

#### 6.4.4 Liaison avec les autres fonctions

Lorsque l’affichage de la page demande un trop grand nombre d’opérations à effectuer, une fonction écrite dans un autre fichier est importée afin de limiter la quantité de code dans les fichiers de routes.

En suivant la même logique que précédemment, les fonctions sont ajoutées dans Web/Fonctions/... Dossier dans lequel se trouve différents sous-dossiers dans lesquels sont classées les fonctions en fonctions de leur rôle. Globalement, chaque fonction est contenue dans un fichier qui porte le même nom que la fonction mis à part pour quelques rares fichiers.

---

### 6.4.5 Exemples de fonctions contenues dans Web/Fonctions

Voici quelques exemples variés de "petites" fonctions présentes dans le répertoire fonctions. Les fonctions présentées dans la section sur l'algorithmique y sont aussi présentes.

La fonction **string\_to\_list()** contenue dans le fichier `to_text.py` présent dans `Web/Fonctions/potager` permet de transformer une liste ainsi que ses sous listes et ses tuples qui sont sous forme de string en liste et sous liste.

Son fonctionnement :

Analyse caractère par caractère de manière récursive sur la string donnée. Lorsqu'un caractère d'ouverture apparaît une nouvelle liste est créée puis ajoutée. La sous-liste ajoutée se termine lorsqu'un caractère de fermeture est rencontré.

De cette manière, on descend récursivement jusqu'à la plus petite sous-liste que l'on ajoute puis on remonte en la construisant avant de passer à la sous-liste suivante.

Sa complexité est facilement calculable puisqu'un seul parcours de la liste est fait de manière linéaire. On en déduit une complexité en  $O(n)$ .

# Chapitre 7

## Tests et performance

### 7.1 Test de la fonction algo\_placement

#### 7.1.1 Premiers tests

Un algorithme pertinent sur lequel travailler en terme de complexité est l'algorithme de placement. La première version mettait plusieurs secondes à être réalisée. La nouvelle a une complexité en  $n^2$  sur la liste des plantes. Ainsi, même sur une liste de 50 plantes, ce qui est déjà conséquent pour un jardin quelques  $m^2$ , l'algorithme tourne en 3 centièmes de seconde.

La première série de tests effectués jouait donc sur le nombre de plantes dans la parcelle : entre 0 et 100. On remarque rapidement que le temps d'exécution du programme devient de plus en plus grand au fur et à mesure des tests. On reste néanmoins dans des ordres de grandeur acceptables au vu de la première version de l'algorithme qui prenait une dizaine de secondes d'exécution pour un cas très simple. Chaque test était réalisé 100 fois en générant à chaque fois des nouvelles valeurs dans la BD. Cet aspect est nécessaire pour gagner en précision mais a considérablement ralenti les tests. En effet, ajouter 100 valeurs dans la base de données puis les supprimer sont des actions plutôt lourdes à effectuer un grand nombre de fois. C'est pourquoi une autre série de tests sera effectuée avec un autre mode de fonctionnement.

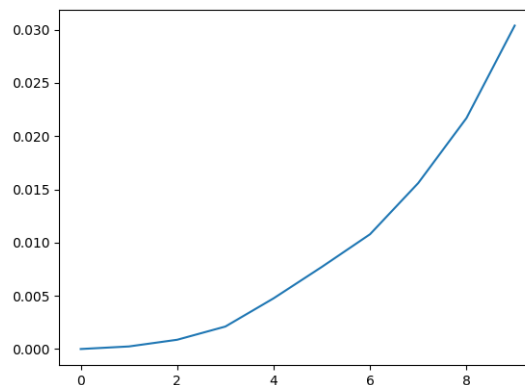


FIGURE 7.1 – Test sur une version non optimisée de l'algorithme de placement (Légende x : nombre de plantes dans le potager | Légende y : temps d'exécution en secondes)

On remarque clairement que le temps d'exécution suit une courbe en  $n^2$  ce qui peut rapidement poser problème sur un grand nombre de valeurs. C'est pour cela que des optimisations de boucles ont été faites. On arrive alors à une version beaucoup plus optimisée qui approche la complexité en  $n$ . Ces deux graphes ont été générés à partir de 100 tests sur un nombre de plantes placées aléatoirement entre 0 et 10.

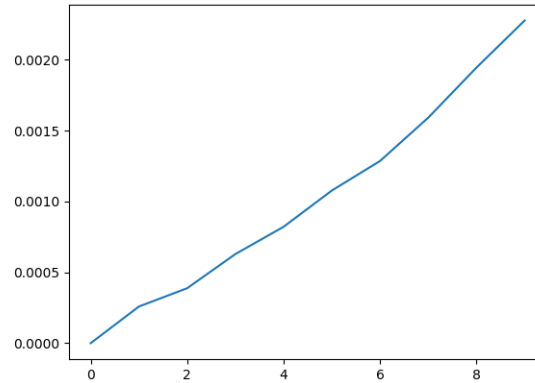


FIGURE 7.2 – Test sur la version optimisée de l’algorithme de placement (Légende x : nombre de plantes dans le potager | Légende y : temps d’exécution en secondes)

Ce test montre que l’algorithme est satisfaisant pour un nombre de valeurs entre 0 et 10.

### 7.1.2 Deuxième série de tests

Comme précisé ci-dessus, cette deuxième série de tests portera sur un plus large éventail de valeurs. Afin de ne pas avoir à faire tourner le test pendant trop longtemps, la manière de faire sera optimisée : on ajoutera 5 par 5 les valeurs dans la BD, en faisant tourner l’algorithme à chaque fois. Ainsi il n’y a pas de génération d’un grand nombre de valeurs simultanément. En faisant tourner cette boucle 100 fois et en calculant la moyenne, on obtiendrait alors un test pertinent et analysable. Le test a été assez long à se réaliser. Mais on obtient une courbe satisfaisante.

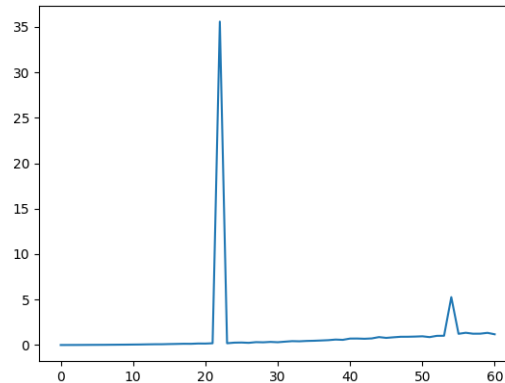


FIGURE 7.3 – Test de l’algorithme de placement sur 0 à 300 plantes (Légende x : nombre de plantes dans le potager (1 = 5 plantes) | Légende y : temps d’exécution en secondes)

## 7.2 Test de la fonction `string_to_lists`

Comme présentée dans la partie précédente, la fonction retransforme une liste qui a été mise sous forme de string en liste (type).

**Complexité :** Comme dit précédemment, la complexité de cette fonction est de  $n$  où  $n$  est la longueur de la liste.

---

Nous avons réalisé des tests grâce au module `time` et avons affiché les résultats grâce au module `matplotlib`. Afin d'éviter les perturbations au maximum, nous avons testé la fonction pour des tailles de listes allant de 1 à 320. Pour chaque valeur nous avons fait 5000 tests de vitesse d'exécution puis enregistré la médiane.

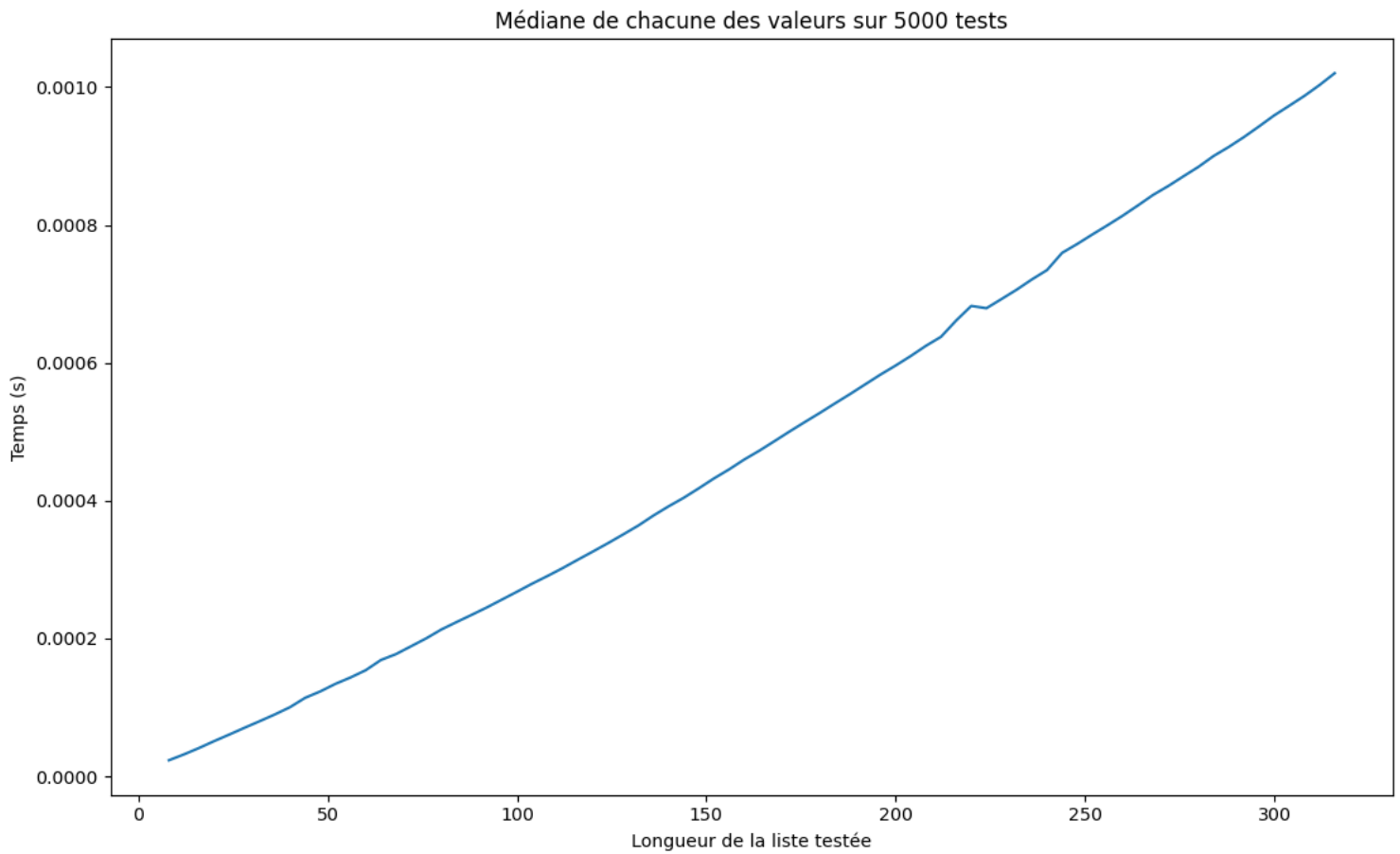


FIGURE 7.4 – Graphique de test de `string_to_list`

# Chapitre 8

## Bilan

### 8.1 Comparaison entre nos attentes de début de projet et ce qui a été réalisé

L'objectif initial était de réaliser une application mettant en relation des jardiniers et leur proposant des suggestions de plantes. Nous avons atteint cet objectif principal. Nous avons beaucoup d'ambitions au début de ce projet, et beaucoup d'idées de fonctionnalités qui n'étaient pas toutes réalistes par rapport à nos connaissances et au temps dont nous disposions.

La partie suggestion de plante, qui était centrale au projet, a bien pu être réalisée, et est fonctionnelle.

L'ajout des plantes dans les parcelles pourrait être plus intuitif. Initialement, nous avions prévu la possibilité de cliquer sur une parcelle pour y placer une plante, mais nous avons dû abandonner cette idée en cours de projet.

Notre idée de départ qui était d'avoir des administrateurs de jardins, des référents et des jardiniers s'est avérée être assez compliqué au niveau de la base de données et compliquait l'utilisation de l'application, mais nous sommes quand même allés au bout de cette idée et avons réussi à obtenir un modèle qui fonctionne, avec la possibilité pour les administrateurs de gérer leurs jardins et d'attribuer des parcelles.

Nous avons envisagé d'autres fonctionnalités pour cette application que nous n'avons pas implémentées faute de temps, comme la possibilité de voir la météo dans un jardin, envoyer un mail de confirmation au moment de l'inscription, et la possibilité de réinitialiser son mot de passe s'il était oublié. Mais nous avons conscience dès le départ que nous n'aurions probablement pas le temps de les implémenter.

En conclusion, le fonctionnement global de l'application est satisfaisant, malgré nos ambitions irréalistes au début du projet.

### 8.2 Pistes d'améliorations

Ce type de projet était une première pour tous les membres du groupes. Nous avons essayé d'appliquer au mieux les outils vus en cours, qu'ils concernent la gestion de projet ou le développement d'application. Nous sommes restés dynamiques et soudés, et nous avons donné le meilleur de nous-même pour mener à bien ce projet. Malgré tous nos efforts, nous avons fait des erreurs, et avons eu des difficultés. Nous allons en parler dans cette section, et voir comment nous pouvons nous améliorer en vue des prochains projets.

#### — Concernant la gestion de projet :

Nous avons mis en place divers outils de gestion de projet dès le départ. Cependant, certains n'étaient pas judicieusement employés par rapport à la nature du projet et à notre manque d'expérience. Le diagramme de Gantt par exemple ne nous a pas beaucoup aidé à être plus efficaces. Nous n'avons pas su utiliser le plein potentiel de cet outil, puisque nous ne connaissions pas les étapes précises du développement de notre application à l'avance. Une méthode agile de gestion de projet serait plus adaptée, puisqu'elle nous offrirait plus de flexibilité en travaillant par cycles, et en adaptant les livrables au cours du projet.

Nous utilisons une Todo liste que nous mettions à jour à chaque réunion, mais qui n'apparaissait que sur les comptes-rendus de réunion, ce qui ne nous permettait pas de voir l'avancement de chaque tâche entre ces réunions. Elle n'était aussi pas toujours complète, et les tâches étaient parfois très vagues. La tenir à

---

jour sur un document partagé pourrait être plus bénéfique : cela permettrait à chaque membre d'indiquer ses avancées de façon plus régulière et précise pour avoir une meilleure vision d'ensemble sur le projet.

— **D'un point de vue interne au groupe :**

Malgré la bonne entente entre les membres du groupe, nous avons eu plusieurs problèmes de communication et de coordination. Il est arrivé que des fonctions et des routes aient été créées alors qu'elles existaient déjà. Ce problème peut s'apparenter à un effet tunnel, il est dû à un manque de visibilité et de transparence des membres du groupe, même si c'était involontaire. Une Todo liste mise à jour par chaque membre plus régulièrement comme évoqué au point précédent pourrait aider à remédier à ce problème.

Nous avons également dû nous réorganiser rapidement en constatant les absences de Maelan à plusieurs réunions, et face à son manque d'implication dans le projet. Les vacances de décembre ont été décisives de ce côté, puisque nous n'arrivions pas à le contacter. Cette réorganisation soudaine a eu un impact sur le projet, nous n'avons pas pu coder certaines fonctionnalités prévues initialement. Cette difficulté n'était pas prévisible, mais nous nous sommes adaptés au mieux.

## 8.3 Bilans individuels

### Hélène Barbillon

Ce projet était l'occasion de mettre en application les principes et méthodes vus en cours. Il m'a permis de revoir les concepts que je n'avais pas bien compris et d'approfondir mes connaissances. Le projet PP2I m'a fait vraiment progresser sur ce plan et m'a permis de développer des compétences qui me seront utiles pour la suite.

M'entendant très bien avec les membres du groupe, je m'attendais à ce que la communication soit très fluide et qu'on avance assez vite. C'était en général le cas, même si quelques problèmes d'organisation nous ont ralenti. Je regrette notamment le manque d'investissement d'un membre du groupe, qui nous a tous pénalisés et énervés, mais nous avons quand même réussi à nous adapter.

La mise en place des outils de gestion de projet vus lors du MOOC Central Lille a été plus compliquée que ce que je pensais initialement, surtout au début quand nous n'avions pas vraiment de vision d'ensemble. Je pense sincèrement qu'une méthode agile (la méthode SCRUM par exemple) est plus adaptée pour des projets informatiques, et j'espère avoir l'occasion de la mettre en place lors d'un futur projet. Mais c'était une expérience très intéressante dont je tire des enseignements pour l'avenir.



---

## Léo Forno

Bilan Léo FORNOFF	
Mes attentes	Personnellement j'avais beaucoup d'attentes vis à vis de ce projet. Avec la prépa, je n'avais plus le temps de faire des projets et le pp2i représentait une bonne occasion pour remettre un pied dans les projets. De plus, il s'agit de mon premier vrai projet de groupe. En plus, le projet représentait l'opportunité de mettre en application les notions vues en cours et même de les approfondir.
Points positifs	J'ai pu approfondir ma connaissance de python. Revoir le HTML et le css et apprendre les rudiments du java-script. Me faire une première expérience de projet à plusieurs et de prendre conscience des points d'amélioration pour les prochains projets. Réaliser un site web.
Difficultés rencontrées	Difficultés au niveau de l'algorithme générant les contours. Parfois des difficultés à appliquer la bonne méthode de gestion de projet. Cela a entraîné un retard dans le projet. Éviter l'effet tunnel Parfois quelques difficultés à gérer le code fait par les autres membres du groupe. Il n'étais pas toujours évident de savoir ce que les autres membres du groupe ont fait ainsi que le fonctionnement de leurs fonctions. L'évolution rapide des documents à aussi eu tendance à rendre difficile la lisibilité de ces derniers.
Axes d'amélioration	Utiliser une méthode de gestion de projet agile. Avoir un chef de projet qui assume complètement son rôle. Rentrer plus rapidement dans le projet pour ne pas être pas prendre de retard. Augmenter l'efficacité des réunions.

TABLE 8.1 – Tableau Bilan Léo

## Mathis Mangold

Ce projet PP2I, bien qu'il représente une approche très directe et très rude du travail en groupe autonome, a été pour moi une grande source d'apprentissages. La gestion de projet est quelque chose de complexe et méthodique qui ne laisse pas de place à l'oubli. Je pense qu'une erreur du groupe a été ce manque de rigueur autant dans la mise en place de la gestion de projet que dans l'organisation générale et temporelle.

Ce projet a donc été globalement source d'apprentissages autant d'un point de vue scolaire que humain et organisationnel. Beaucoup d'imprévus se sont présentés, beaucoup de défis ont dûs être relevés. C'est une expérience que j'ai trouvé particulièrement intéressante et enrichissante. Je pense qu'à l'avenir, dans de futurs projets, j'aborderais le travail en groupe d'une autre manière.

Je tenais à remercier les membres de mon groupe qui m'ont beaucoup apporté de par leur expertise et de par les discussions que l'on a pu avoir. Et malgré l'énorme imprévu qu'a représenté l'absence continue de Maelan, c'est aussi un apprentissage que de se rendre compte de l'interdépendance très forte entre les membres d'une équipe-projet.

## Maelan Tiger

# Table des figures

3.1	Profil du projet . . . . .	9
3.2	Organigramme des tâches . . . . .	9
3.3	Diagramme de Gantt . . . . .	10
4.1	Schéma de la base de données . . . . .	14
6.1	Barre de Navigation au démarrage . . . . .	23
6.2	Barre de Navigation une fois connecté . . . . .	23
6.3	Barre de Navigation réduite avec menu déroulant . . . . .	24
6.4	Initialisation des sessions . . . . .	25
6.5	Page de connexion / <b>login</b> . . . . .	26
7.1	Test sur une version non optimisée de l'algorithme de placement (Légende x : nombre de plantes dans le potager   Légende y : temps d'exécution en secondes) . . . . .	28
7.2	Test sur la version optimisée de l'algorithme de placement (Légende x : nombre de plantes dans le potager   Légende y : temps d'exécution en secondes) . . . . .	29
7.3	Test de l'algorithme de placement sur 0 à 300 plantes (Légende x : nombre de plantes dans le potager (1 = 5 plantes)   Légende y : temps d'exécution en secondes) . . . . .	29
7.4	Graphique de test de string_to_list . . . . .	30
8.1	Page de profil . . . . .	36
8.2	Page "Mes parcelles" . . . . .	37
8.3	Page "Dico des plantes" . . . . .	37
8.4	Page d'attribution des parcelles . . . . .	38
8.5	Page des jardiniers . . . . .	38
8.6	Création de la base de donnée . . . . .	66
8.7	Création de la base de donnée(suite) . . . . .	67

# Bibliographie

## Bibliographie

### 8.3.1 Liens pour le Python :

- <https://www.youtube.com/watch?v=NB5LGzmSiCs>
- <https://www.youtube.com/watch?v=rdbul3p7Bug&t=2s>
- <https://www.youtube.com/watch?v=eSUC6pRM2qg&t=89s>
- <https://www.youtube.com/watch?v=Hrv5KjLPYpc>
- <https://www.youtube.com/watch?v=6Ltk49YhrWY&t=73s>
- <https://www.youtube.com/watch?v=MYyd4sImfmI&pp=ugMICgJmchABGAE%3D>
- <https://www.youtube.com/watch?v=o7WfO6mFKyY&t=202s>
- <https://flask.palletsprojects.com/en/2.2.x/tutorial/views/>
- <https://www.youtube.com/watch?v=-amkXnxdUNE>
- <https://stackoverflow.com/questions/39682397/sharing-code-between-blueprint>
- <https://mail.python.org/pipermail/flask/2019-October/001316.html>
- <https://stackoverflow.com/questions/39965970/flask-blueprint-cannot-read-sqlite3-databases-from-config-file>

### 8.3.2 Liens pour le CSS et HTML

- <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- <https://www.youtube.com/watch?v=qNgO9rXb3-8>
- <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/area>
- <https://stackoverflow.com/questions/72236983/pop-up-on-hover-area-tag-html>
- <https://forum.alsacreations.com/topic-4-71216-1-Roll-over-sur-une-carte-map-area.html>
- <https://www.youtube.com/watch?v=CNosVpZFwfc>
- <https://openclassrooms.com/forum/sujet/hover-sur-une-map-area>
- <https://developer.mozilla.org/fr/docs/Web/CSS/margin>
- <https://developer.mozilla.org/fr/docs/Web/CSS/padding>

### 8.3.3 Liens pour l'état de l'art


- <https://www.economie.gouv.fr/plan-de-relance/mesures/jardins-partages-et-agriculture-urbaine>
- <https://thaonlesvosges.fr/ville-au-quotidien/emploi/les-structures-de-linsertion-par-lactivite-economique/les-jardins-de-cocagne/>
- <https://monjardinmamaison.maison-travaux.fr/mon-jardin-ma-maison/conseils-jardinage/jardin-partage-choses-a-savoir-concept-cartonne-246200.html#item=1>
- <https://agriculture.gouv.fr/francerelance-lancement-de-lappel-projets-jardins-partages>

# Annexes

## Annexe 1 : Pages du site

Les pages suivantes sont évoquées dans la partie *Aspect visuel du site*.

### Page Profil


 Le Bonheur Est Dans Le Potager | [Mon profil](#) [Mes parcelles](#) [Dico des plantes](#) [Attrib parcelles](#) Bonjour Cy Tn [Déconnexion](#)

## Votre profil

**Vos informations**

Nom : Tn  
Prénom : Cy  
Email : tncy@telecom.eu  
Numéro identifiant: 8

Choisir une autre photo de profil



### Vos parcelles

*Vous pouvez gérer vos parcelles dans l'onglet '[Mes parcelles](#)'.*

**Parcelle numéro 96**  
Présente dans le jardin : 2 ([Voir les jardiniers du jardin 2](#))  
De taille : 200.0 x 100.0

**Parcelle numéro 97**  
Présente dans le jardin : 2 ([Voir les jardiniers du jardin 2](#))  
De taille : 100.0 x 100.0

### Les jardins que vous administrez

*Vous administrez ces jardins, cela veut dire que vous pouvez y créer de nouvelles parcelles et les attribuer à des jardiniers.  
Vous pouvez gérer ces jardins dans l'onglet '[Attrib parcelles](#)'.*

**Jardin numéro 4** ([Voir les jardiniers du jardin 4](#))  
Situé : 193 rue des Paul Muller dans la ville de Villers 54600  
Référent.e de ce jardin : Cy Tn (identifiant: 8)

FIGURE 8.1 – Page de profil

## Page de gestion des parcelles

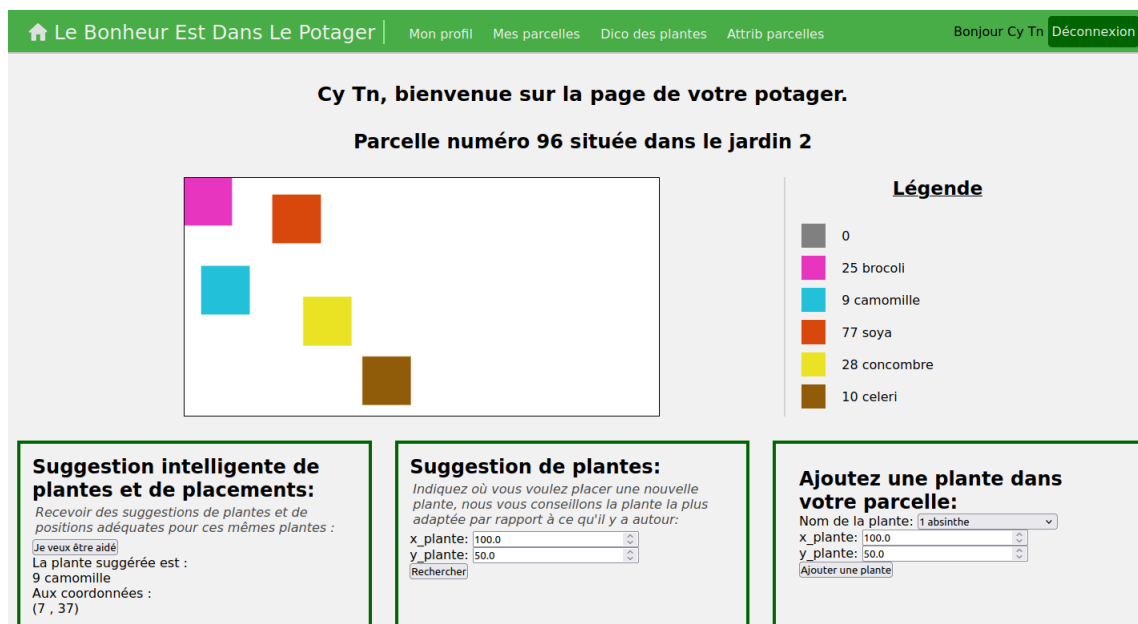


FIGURE 8.2 – Page "Mes parcelles"

## Page "Dico des plantes"



FIGURE 8.3 – Page "Dico des plantes"

Page "Attrib parcelles"

Le Bonheur Est Dans Le Potager

Mon profil

Mes parcelles

Dico des plantes

Attrib parcelles

Bonjour Cy Tn

Déconnexion

Attribution des parcelles

Vous avez accès à cette section car vous administrez un ou plusieurs jardin(s). Dans cette section vous pouvez créer des parcelles dans ces jardins et les attribuer, ou en supprimer.

Créer une nouvelle parcelle:

Numéro du jardin: 4

Longueur de la parcelle: 100

Largeur de la parcelle: 100

Numéro d'identifiant du jardinier:

1 (Léo Fornof)

CRÉER UNE NOUVELLE PARCELLE

Jardin n°4

Situé au 193 rue Paul Muller à Villers (54600)

Référent.e: Cy Tn (n°8)

Numéro parcelle	Longueur	Largeur	Jardinier	
98	100.0	100.0	Léo Fornoff (n°1)	SUPPRIMER PARCELLE
99	100.0	100.0	Cy Tn (n°8)	SUPPRIMER PARCELLE
100	300.0	200.0	Hélène Barbillon (n°4)	SUPPRIMER PARCELLE

Voir les jardiniers du jardin 4

FIGURE 8.4 – Page d’attribution des parcelles

Page des jardiniers

Le Bonheur Est Dans Le Potager

Mon profil

Mes parcelles

Dico des plantes

Attrib parcelles

Bonjour Cy Tn

Déconnexion

Les jardiniers du jardin 4

Cliquez pour afficher un profil!

Noms	Statut (dans ce jardin)
Léo Fornoff	- jardinier -
Hélène Barbillon	- jardinier -
Cy Tn	- administrateur.trice - jardinier- référent.e -

NOM: Tn

PRÉNOM: Cy

MAIL: tncy@telecom.eu

IDENTIFIANT: 8

STATUT: - administrateur.trice - jardinier- référent.e -

ADMINISTRE LE JARDIN: 4

JARDINE DANS LES PARCELLES: 96, 97, 99

RÉFÉRENT.E DU JARDIN: 4

FIGURE 8.5 – Page des jardiniers

38

---

## Annexe 2 : Comptes Rendus de réunion

CR du 16 Octobre 2022

---

### Ordre du jour (Hélène) :

- Définition du projet (reprise du pdf des attendus)
- Ses limites
- Rappel des objectifs et des échéances
- Etat de l'art :
  - ce que l'on a trouvé
  - mise au point des rôles
  - ce qu'il reste à faire
- point sur les outils de gestion de projet que l'on devrait mettre en oeuvre
- idées
- conclusion
- Mise en place de la prochaine réunion

### Début de la réunion : 12h04

Création du drive du projet  
Direction de la réunion par hélène

### Définition du projet

#### Objectifs

Faire une app innovante dédiée à l'utilisation des ressources dans les vergers et potager du territoire

#### Cadre existant du domaine

Déjà sur le drive

#### Etude des app num déjà existantes

déjà commencé sur le web

#### Définitions des critères

- web (Hélène)
- base de données (Hélène)
- fond algorithmique (Léo)
- attributs des offres particulières sur le web avec tableau comparatif
  - site web (Maelan)
  - application (Maelan)
  - entreprise / associatif (Léo)
  - paiement (Maelan)
  - local/ national/ international (Léo)
  - Cas particulier ou plus général (Mathis)

---

## Ses limites

### Choses à garder en tête

- web (Hélène)
- base de données (Hélène)
- fond algorithmique (Léo)
- attributs des offres particulière sur le web avec
  - tableau comparatifs
  - site web (Maelan)
  - application (Maelan)
  - entreprise/ associatif (Léo)
  - paiement (Maelan)
  - local/ national/ International (Léo)
  - Cas particuliers ou plus général (Mathis)

### Rappel des objectifs

- Etat de l'art jeudi à envoyé avant jeudi 18h sur le GIT (Hélène), Maelan s'en occupera (Maelan)
- Modification du fichier README.mk sur le git (Léo)
- Présentation du projet samedi
  - Avoir un support pwp (Mathis et Maelan)
  - Explicité les démarches du projet
  - Le PWP
    - Fabrication du pwp partagé puis mise en forme par Maelan
    - mot sur le contenu du pwp (Mathis)
    - Mise en forme précise lors de la prochaine réunion (Léo)
    - Mais essentiel fait pour jeudi (Mathis)
    - Répétition vendredi à 13h

## Etat de l'art

### Attendus

- Réponse appel d'offre
- Déjà existant
- Application numérique
- tableau de comparaison avec nos propres critères (défini précédemment)
- LaTeX

### Ce qui à déjà été fait

- Recherche sur toutes choses que l'on a trouvé sur le web à propos des circuits courts et jardin partagé (Mathis)
- AMAP : association de mise en lien entre producteur et consommateur (ex : club marché), réduction des coûts car il n'y a pas d'intermédiaire. Tout ce qui est produit est vendu (Hélène)
- Jardin partagé : définition(s) (Hélène)
- Association sur le jardinage, conseils etc. (Mathis)
- Jardin partagé géré par des associations (Hélène)
- Jardin partagé : production reste tout de même limité
- Le jardin de Cocagne à Thaon-Voges, production maréchère bio en circuit courts qui permet une réinsertion pro (Hélène)
- Bâle propose subvention pour faire des cultures dans les espaces verts des immeubles, petite échelle mais proximité, mais différences d'implication entre les personnes (Mathis)
- Application :
  - Fruit and Food (Léo)
  - "Jardin partagé" mise en relation des personnes (Hélène)



---

"Jardin partagé" une deuxième application qui permet d'avoir des informations sur les plantes dans des jardins partagés (Hélène)

Potager City, potager locaux, interfaces ergonomique, grande variété, fonctionnement avec des paniers (Maelan)

— Ajout de la part de Maelan et de Hélène (Léo)

— Partir sur un nombre réduit de projets qui représente bien ce que l'on a trouvé sur internet (Mathis)

## Point sur les outils de gestion de projet que l'on devrait mettre en oeuvre

### Charte Projet

- Cadrage
- déroulement du projet
- Fait par le chef de groupe (Léo) -> Maelan va faire la charte projet
- Poster sur le Drive et sur le Git (Maelan)
- Validation avant la prochaine réunion (Léo)
- Etat de l'art + analyse (Mathis), "j'ai déjà des idées" -> fait pour la prochaine réunion
- Attendre avant de faire le WBS mais reste sur la liste des choses à faire (Léo et Hélène)
- Todo list jusqu'à la prochaine réunion (Léo)

### Idées

- Site web, et si on le temps application mobile (Léo)
- Redemander si forcément application (sur smartphone)
- idée de l'application applicable par des groupes locaux (Léo), semble compliqué (Hélène)
- Faire un contrat entre un organisme entre des personnes et un besoin de l'organisation (Mathis)
- Base de donnée qui recense les jardins qu'il y a autour (Hélène)

### Conclusion

- Noté toutes les idées
- Prochaine réunion lundi 17 octobre 18h médialab
- répétition vendredi entre 13h et 14h
- Compte rendu de la réunion LaTeX (Léo)
- Charte projet LaTeX (Maelan)
- Modification README.mk sur le git (Léo)
- Etat de l'art + analyse LaTeX (Mathis)
- Diapo (Hélène) à faire après la prochaine réunion

### Todo list

- Noté toutes les idées
- Prochaine réunion lundi 17 octobre 18h médialab
- répétition vendredi entre 13h et 14h
- Compte rendu de la réunion LaTeX (Léo)
- Charte projet LaTeX (Maelan)
- Modification README.mk sur le git (Léo)
- Etat de l'art + analyse LaTeX (Mathis)
- Diapo (Hélène) à faire après la prochaine réunion

### Prochaine réunion

- Partage des idées et choix/ définition du projet (Hélène)

---

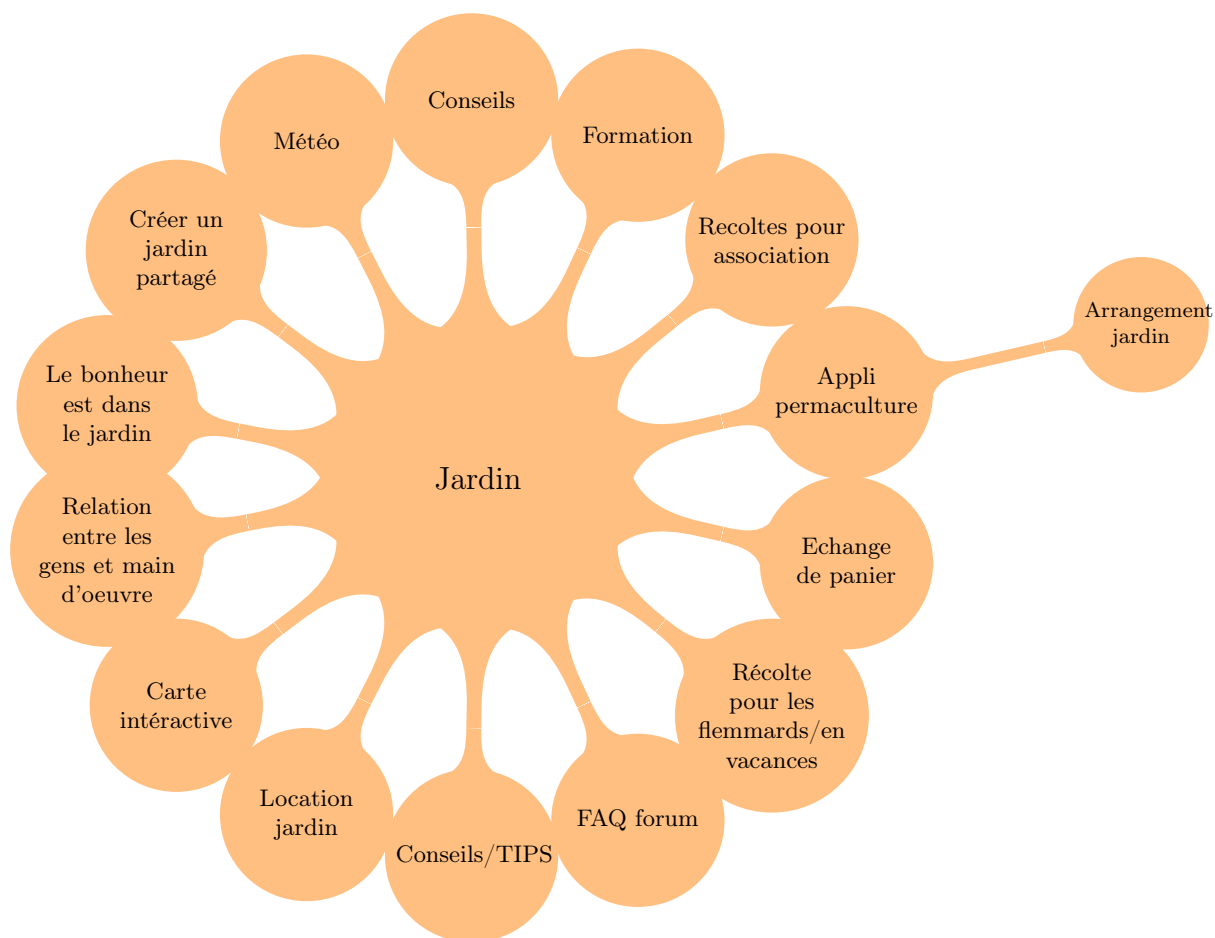
**Fin de la Réunion 13h30**

## Ordre du jour

- Brainstorm
- Définition des attendus et des fonctionnalités innovantes

## Début de la réunion 18h06

Tableau récapitulatif des idées :



Sélection des composantes intéressantes et innovantes, première définition exhaustive des fonctionnalités

---

## LE BONHEUR EST DANS LE POTAGER (titre provisoire)

Gestionnaire de jardin partagé

- gestion des partielles
  - Admin
  - User → Chacun leur parcelle
- Pour chaque parcelle :
  - Choix
    - Légumes
    - Fruits
    - Herbes aromatiques
  - Permaculture
- Idées supplémentaire si le temps le permet
  - Météo (API Météo France)
  - Wiki
  - Forum de discussion
  - Carte interactive

**Fin de réunion 18h45**

## Compte rendu de la réunion du 24 octobre 2022

Début de la réunion 13h19. Secrétaire de réunion : Léo

### Ordre du jour :

Backlog

Retour sur la présentation de projet du 22-10-2022

Partie web

Partie base de données

Partie algorithmique

To do list mise à jour

Outils de gestion de projet

### Backlog

Ce qui a été fait depuis la dernière fois :

- Définition du projet
- État de l'art
- Premier compte-rendu projet en Latex
- Modification du Readme.md
- Diapo de présentation
- Rédaction des comptes-rendus de réunion en LaTeX
- Gestion de projet (cf compte rendu projet)

## Retour sur la présentation de projet du 22 10 2022

Principales remarques :

- Rester sur l'idée de base, ne pas se disperser
- Analyser la matrice SWOT (forces et menaces)
- Penser à mettre une introduction et une conclusion
- Ajouter des délais pour les objectifs
- Commencer par un backlog
- Avancer rapidement dans le projet (! temps)
- Quelques fautes de français

## Web

### Front

Les choses à faire pour le front :

- Page home
- Login (lié au back)
- Page profil
- Page gestion de sa parcelle

- 
- Page admin
  - Style de la page (UI)

## Back

Les choses à faire pour le back :

- Login
- Communication avec la base de données
- Gestion de session

## BDD

Les choses à faire :

- Choisir le type de base de données
- Schéma relationnel de la base de données
- Vérifier que le schéma est en 3NF
- Créer les tables de la base de données
- Récupérer les données à mettre dans la base

## Algorithmique

Les choses à faire :

- Algorithme de création de terrain
- Algorithme de suggestion de plante
- Algorithme de placement
- Combinaison des deux algorithmes
- Implémentation avec la base de données

## Mise à jour todo liste

- Prochaine Réunion Lundi 31-10-22 à 14h
- WBS Mathis pour la prochaine réunion (Mathis)
- Design du front global (couleur et structure)
- Choix de la base de donnée à faire aujourd'hui (Léo)
- Schéma relationnel base de données en 3NF pour la prochaine réunion (31/10) (Maelan)
- Diagramme de Gantt pour la prochaine réunion
- Formation accélérée sur la gestion utilisateur pour lundi (Hélène)

Tâche	Responsable	A faire pour le
Prochaine Réunion Lundi 31/10/22		
WBS	Mathis	31/10/22
Design du front	Maelan	31/10/22
Schéma relationnel base de données	Maelan	31/10/22
Diagramme de Gantt		
Formation accélérée sur la gestion utilisateur (web)	Hélène	31/10/22

## Fin de la réunion 13h57

## Réunion du 31/10/2022

Secrétaire(s) de réunion : Hélène

Absent : Maelan

## Backlog

Tâche	responsable	Avancement
WBS	Mathis	fait
Design du front	Maelan	?
Schéma relationnel BD	Maelan	?
Choix BD + esquisse du schéma relationnel BD	Léo	fait
Formation gestion utilisateur	Hélène	en cours
4 Compte-rendu réu laTeX	Léo	fait
Création d'un objet Terrain	Léo	fait

## Ordre du jour

- Gestion de projet : WBS/Gantt
- Base de données : schéma relationnel
- WEB : design, gestion utilisateur
- Objet Terrain
- Todo list

## Gestion de projet

Retours sur le WBS fait par Mathis. Pour la partie algorithmique, il faudra être en méthode agile. Il faut prioriser les tâches à faire et réaliser un Gantt. Il faut aussi mettre à l'écrit le cahier des charges

## BD

Présentation du schéma relationnel fait par Léo.

Modifs à faire :

- Enlever la clé primaire "ID plante" dans la table "relation plante jardin"
- Enlever la table "utilisateur parcelle"
- Ajouter "taille effective de la plante" dans la relation "plante-jardin"
- Ajouter "taille de la parcelle" dans "parcelle"

## WEB

Front : absence de Maelan à la réunion donc sujet reporté à la prochaine réunion.

Gestion des users : utilisation des sessions avec Flask à poursuivre. Sessions gérées au niveau du client ou du serveur.

---

## Objet Terrain

Présentation par Léo de l'objet Terrain qu'il a codé : création d'un terrain, mise à l'échelle, ajout de plantes, affichage avec colorisation.

## Todo liste

Prochaine réu : Mercredi 02/11 14h30

Tâche	Responsable	A faire pour le
Gantt	Mathis	2/11
Cahier des charges	Léo	2/11
Design	Maelan	2/11
Schéma relationnel BD corrigé avec contraintes au propre	Hélène	2/11
CR de réunion en LaTeX	Hélène	2/11



## Réunion du 02/11/2022

Secrétaire(s) de réunion : Mathis

### Backlog

Tâche	responsable	Avancement
GANTT	Mathis	fait
Design du front	Maelan	fait
Schéma BD	Hélène	fait
Cahier des charges	Léo	fait

### Ordre du jour

- GANTT
- Cahier des charges
- Base de données : décision à propos du schéma relationnel
- WEB : design et charte graphique
- Todo list

### GANTT

Retours sur le GANTT fait par Mathis.

- nécessité de lire chacun le GANTT pour se répartir les tâches à la prochaine réunion
- nécessité de refaire le WBS qui a été fait dans la mauvaise optique
- nécessité de faire des tirets plus courts sur le GANTT (problème de lisibilité)

### Cahier des charges

Présentation du travail fait par Léo : Retour sur la contrainte de l'innovation et le problème de trouver des objectifs.

Quelques modifications apportées suite à la discussion : changement de la problématique, des objectifs et fonctionnalités

### BD

Présentation du modèle relationnel fait au propre par Hélène. La validation était nécessaire pour pouvoir ensuite créer la base de données. Quelques précisions/modifications ont été apportées sur certains attributs et relations.

---

## WEB

Front : présentation par Maelan de sa proposition de design.

Couleurs et aspects validés par tout le monde : début de la réalisation des autres pages sur un modèle similaire

## Todo liste

Prochaine réu : Mardi 08/11 16h en présentiel

Tâche	Responsable	A faire pour le
Lire le GANTT et réfléchir à ce qu'on veut faire	Tout le monde	8/11
Phrases plus courtes sur le GANTT	Mathis	8/11
Ébauches des pages HTML	Léo	8/11
Création BD et tables	Maelan	8/11
Analyse de la matrice SWOT	Hélène	8/11
Se renseigner sur les bases de données de permaculture	Mathis	8/11

## Réunion du 08/11/2022

Secrétaire(s) de réunion : Hélène

## Backlog

Tâche	Responsable	Avancement
Lire le GANTT et réfléchir à ce qu'on veut faire	Tout le monde	fait
Phrases plus courtes sur le GANTT	Mathis	fait
Ébauches des pages HTML	Léo	fait
Création BD et tables	Maelan	en cours
Analyse de la matrice SWOT	Hélène	fait
Se renseigner sur les bases de données de permaculture	Mathis	en cours

## Ordre du jour

- Diagramme de Gantt
- WEB
- Partie sql

## Diagramme de Gantt

Répartition des tâches sur le diagramme de Gantt. (voir diagramme Gantt)

## WEB

Léo nous montre les pages html qu'il a faites. Maelan va faire les designs des pages d'accueil. Discussion sur la page login/register, décision de faire une page de connexion différente de la page d'inscription.

## Partie sql

Il faut créer la bd pour avoir une récupération des données : à faire pour la prochaine réunion. Mise au point sur la structure de l'application.

## Todo liste

Prochaine réunion : mardi 15/11 à 16h

---

Tâche	responsable	A faire pour le
Création BD, ajout d'utilisateur	Maelan	15/11
Back de vérification	Léo	15/11
Fenêtre d'affichage des parcelles	Léo	25/11
Se renseigner sur les bases de données de permaculture	Mathis	15/11
Page de gestion des participants	Hélène	15/11
CR, mise en forme du latex	Hélène	15/11

---

CR du 15 Novembre 2022

Réunion du 15/11/22

Secrétaire(s) de réunion : Fornoff Léo

Début de la réunion 16h20, fin de la réunion 17h10

Backlog

Tâche	responsable	Avancement
Création BD, ajout d'utilisateur	Maelan	Reporté
Back de vérification	Léo	Fait
Fenêtre d'affichage des parcelles	Léo	En cours
Se renseigner sur les bases de données de permaculture	Mathis	Fait
Page de gestion des participants	Hélène	Reporté
CR, mise en forme du latex	Hélène	Fait

## Ordre du jour

- Compte rendu
- Problèmes rencontrés depuis la dernière réunion
- Explication sur le JS
- Organisation travail en groupe

## Compte rendu

Réorganisation du fichier main par Hélène. Il faudra penser à mettre tous les outils de gestion de projet que l'on a utilisé. Rajouter une photo du Gantt dans le git.

## Résolution de problèmes

Problème de gestion du temps, notamment à cause de projet de la vie associative.  
Mathis : difficulté d'extraction et de traitement de certaines informations depuis les sources internet.  
Hélène : difficulté sur le Latex mais a finalement réussi à s'en sortir.  
Des difficultés à comprendre certaines parties du code des autres.  
Un peu de manque de visibilité globale du projet  
Plus d'explication pour le code à la fin des réunions en réel

## Explication JS

Mise au point et explication de fonctionnement du code

## Organisation travail en groupe

Création de la base de données le 15/11/2022

---

## Todo liste

Tâche	responsable	A faire pour le
Visite du jardin botanique	Mathis	22/11
Fin mise en place de la base de plantes	Mathis	22/11
Ajout des utilisateurs dans la base de données	Maelan	22/11
Fenêtre d'affichage des parcelles	Léo	22/11
Création de la base de données	Hélène	16/11
Page gestion des participants	Hélène	22/11

**Prochaine réunion : mardi 22 novembre 2022 à TélécomNancy à 16h (a été annulée)**

---

CR du 29 Novembre 2022

Réunion du 29/11/22

Secrétaire(s) de réunion : Héléna

## Backlog

Tâche	responsable	A faire pour le	avancement
Visite du jardin botanique	Mathis	22/11	fermé, abandon
Fin mise en place de la base de plantes	Mathis	22/11	en cours
Ajout des utilisateurs dans la base de données	Maelan	22/11	fait
Fenêtre d’affichage des parcelles	Léo	22/11	fait
Création de la base de données	Héléna	16/11	fait
Page gestion des participants	Héléna	22/11	fait
Sessions	Héléna		en cours

## Ordre du jour

- Page utilisateur
- Sessions
- Page mon potager
- Todo liste

## Page utilisateur

Page des Participants : n’afficher que les utilisateurs du jardin dans lequel on est, et mettre en premier les admins du jardin/ mettre une petite étoile, si on est pas connecté, redirection sur le register.

Se mettre d’accord sur tout mettre en français ou en anglais.

## Sessions

reste à faire : fermeture de sessions+vérifications

## Page mon potager

Couleurs pour chaque id de plante. Explications zones cliquables pour renvoyer des infos sur les plantes.  
Choix de données pour afficher jardin en 2d : matrices.n

---

## Todo liste

Tâche	responsable	A faire pour le
Fin mise en place de la base de plantes	Mathis	prochaine reu
Sessions	Hélène	prochaine reu
Améliorations page des participants	Hélène	prochaine reu
Avancements page mon potager	Léo	prochaine reu



---

CR du 18 décembre 2022

Réunion du 18/12/22

Secrétaire(s) de réunion : Mangold Mathis

Absent : Maelan Tiger

## Backlog

Tâche	responsable	Avancement
Visite du jardin botanique	Mathis	Avorté
Fin mise en place de la base de plantes	Mathis	Reporté maladie et exams
Ajout des utilisateurs dans la base de données	Maelan et Léo	Fait
Fenêtre d'affichage des parcelles	Léo	Fait et en cours d'amélioration
Création de la base de données	Hélène	Fait
Page gestion des participants	Hélène	Fait mais à tester

## Ordre du jour

- Sessions et améliorations faites par Hélène
- Avancements du côté de Léo
- Répartition des tâches pour les prochains jours (moins en moins de temps restant)

## Sessions et améliorations faites par Hélène

Gestion de la déconnexion en fonction de la taille de la page (bouton différent si la page est plus petite)

Reconnaissance de l'utilisateur et onglets accessibles par l'utilisateur

A faire :

bloquer l'accès aux pages par l'URL si non connecté

## Avancements du côté de Léo

Page de création d'une nouvelle parcelle :

Donner un ID de jardin

Donner une largeur et une longueur

Affichage d'une parcelle :

Optimisation des fonctions pour l'affichage et la reconnaissance des parcelles == si aucune modification de la parcelle alors utilisation de ce qu'on a en mémoire (ne pas recalculer les contours)

Base de donnée == ajout de "polygone" pour chaque parcelle (coordonnées coins) pour faciliter la création de l'image

Page profil d'utilisateur

Page d'erreur == faire un render template avec son message d'erreur

A faire :

Accessibilité par admins à la page nouvelle parcelle == gérer un type de session "admin" pour simplifier

session["admin"]= true (sinon y en a pas), et session["num jardin"]=[1],[2] pour savoir on est admin sur quel jardin, session["parcelle"]=[22],[2] pour connaître les parcelles à qui on a accès

Menu déroulant des ID et des noms de jardin

Gérer ID user à qui appartient la parcelle

---

Erreur affichage image  
Tester la page et les algos (possibles erreurs non détectées)  
Possible problème de stockage de toutes les images == supprimer de manière régulière et automatique  
Vérifier accès au jardin par la bonne personne  
Page attribution des parcelles  
Récupérer info de session pour profil et sécuriser accès pour l'utilisateur connecté

## Répartition des tâches pour les prochains jours (moins en moins de temps restant)

Temps restant diminue == augmenter cadence des réunions et accorder un maximum de temps au projet

### Todo liste

Tâche	Responsable	A faire pour le
Finir affichage parcelle	Léo	Prochaine Réunion (à gérer avec Maelan)
Page d'attribution parcelles admin	Léo	Prochaine Réunion
Mise à jour des pages avec les sessions	Léo	Prochaine Réunion
Page ajout de plante	Mathis	Prochaine Réunion
Plus d'informations dans le dico session	Hélène	Cette aprem
Finir génération BD affinités plantes	Mathis	Prochaine Réunion
Commencer algo de suggestion	Mathis	Prochaine Réunion
Checker les "A faire"	Tout le monde	Dès qu'on a un peu de temps
Page de profil pour jardin	Maelan	prochaine réu

---

CR du 21 décembre 2022

Réunion du 21/12/22

Secrétaire(s) de réunion : Mangold Mathis

Absent : Maelan Tiger

## Backlog

Voir plus bas les avancements de chacun

## Ordre du jour

- Avancement Léo
  - Organisation et division des routes
  - Pages ajoutées
- Avancement Hélène
  - Sessions toujours et encore
  - Pages "Mes parcelles"
  - Changement des données affichées dans "Jardiniers"
- Avancement Mathis
  - Relations entre des plantes ennemies/amies à partir d'un pdf
  - Algorithme de suggestion de plantes
  - Ajout de plantes

## Avancements Léo

### Utilisation de Blueprint et organisation des routes

Organisation en sous-dossier et sous-routes plus précises (admin, etc.)

Permet une vision globale == lancer le fichier run.py et il utilise un fichier d'initialisation pour créer l'app. On peut ensuite importer d'autres fichiers et enregistrer ces fichiers comme des blueprint. C'est à dire créer d'autres routes vers lesquelles seront redirigées les requêtes "GET" correspondantes.

Dans le fichier en question == importer les autres fichiers et créer des routes @auth.route("/signin") ou @admin.route

Choix de ne pas changer le chemin du dossier templates pour chaque fichier de routes == soucis de simplification et de facilité d'accès

Problèmes à résoudre :

- Erreur pour ouvrir un fichier tiers == probablement que le fichier est simplement en dehors du dossier "de base" et que la librairie n'accepte pas de revenir autant en arrière
- Erreur globale à résoudre lors du lancement du run.py

## Pages ajoutées

- Page administrateur d'affichage des jardins sur lesquels il a les droits
- A faire encore :

Possibilité de supprimer une parcelle

Possibilité de changer le propriétaire d'une parcelle

---

## Avancements Hélène

Improvements sur les sessions et les accès admin etc.

Improvements sur les pages Parcelles et Jardiniers

A faire :

Page qui affiche les jardiniers de chaque jardin en passant par les parcelles (but : afficher les jardiniers de notre jardin) + si ça avance vite, finir page attribution parcelles + ne pas avoir accès aux parcelles si on n'est pas dedans.

## Avancements Mathis

explication algo mathis : à partir d'un doc pdf récupérer relations entre plantes et mettre dans bd sql

A faire : mettre relations entre les plantes dans "vraie" base de donnée + ajout de plante avec l'objet classe terrain (explications de Léo sur comment l'utiliser)+mettre CR sur git (on en a loupés qquns, vérifier où on s'était arrêté.

Réu Léo/Hélène demain 11h pour faire le point sur les fonctions du site, faire le tri

En attendant que le Blueprints soit fini : si modifs dans le app.py : mettre un warning que la fonction a été modifiée

## Todo liste

Tâche	responsable	A faire pour le
Problèmes énoncés plus haut à résoudre	Léo	Prochaine réu
Mettre relations plantes dans la db	Mathis	Prochaine réu
Ajout de plante avec l'objet classe terrain	Mathis	Prochaine réu
Page jardinier + attribution parcelles	Hélène	Prochaine réu

---

CR du 06 janvier 2023

Réunion du 01/06/23

Secrétaire(s) de réunion : Mangold Mathis

## Backlog

Tout le monde présent

Tâche	responsable	Avancement
Problèmes énoncés plus haut à résoudre(cf CR précédent)	Léo	fait
Mettre relations plantes dans la db	Mathis	fait
Ajout de plante avec l'objet classe terrain	Mathis	fait
Page jardinier + attribution parcelles	Hélène	fait

## Ordre du jour

- Mise au point
- Todo liste

## Mise au point

La réunion a commencé par une mise au point globale : chacun a dit ce qu'il a fait. La plupart des pages de l'application sont faites, mais il reste des bugs à corriger et l'algorithme de suggestion de Mathis à intégrer à l'application.

Maelan est allé parler avec M.Oster car il n'a pas de commits sur le projet (pb persos), il doit ajouter une fonctionnalité avant la fin. Il compte ajouter la météo qui s'affiche pour un jardin

Il faut aussi rédiger le rapport. Pour la prochaine réunion nous allons tous réfléchir à ce qu'il faut mettre dedans pour nous répartir le travail plus facilement.

Sujet des tests de complexités évoqués.

Prochaine réunion : dimanche 14h sur discord.

## Todo liste

Tâche	A faire pour	responsable
Faire une liste des trucs à mettre ds le rapport	dimanche	tt le monde
implémenter l'algo sur la page	dim	mathis
météo	dim	maelan
page home	dim	maelan
pb affichage parcelle	dim	léo
redirection qd on clique sur une page + dico	dim	Hélène
légende de l'affichage d'une parcelle -> nom des plantes	dim	Hélène
faire des tests sur le site	dim	ceux qui peuvent
test de complexité	dim	ceux qui peuvent

---

CR du 08 janvier 2023

Réunion du 08/01/2023

Secrétaire(s) de réunion : Héléna

Présents : tt le monde

## Backlog

Tâche	responsable	avancement
Faire une liste des trucs à mettre ds le rapport	tt le monde	fait
implémenter l'algo sur la page	mathis	quasi fini(détails)
météo	maelan	presque fini
page home	maelan	fait
pb affichage parcelle	léo	fait
redirection qd on clique sur une page + dico	Héléna	fait
légende de l'affichage d'une parcelle(nom des plantes)	Héléna	fait
faire des tests sur le site	ceux qui peuvent	en cours
test de complexité	ceux qui peuvent	pas fait

## Ordre du jour

- Rapport
- Todo liste

## Ce qu'il reste à faire

A part quelques détails que Maelan et Mathis doivent apporter, on considère que l'application est finie.

## Rapport

Plan du rapport :

- Intro  
Parler du sujet, des objectifs, présentation TRES GENERALE
- Etat de l'art  
X. sous partie Présentation de l'application et sa pré-construction prévisionnelle :  
détail des fonctionnalités, des buts et de l'innovation par rapport aux solutions existantes
- Gestion projet  
à modifier un peu pour analyser SWOT et changer Gantt prévisionnel (+ analyse)
  1. choix de la méthode agile, outils de communication mis en place
  2. Matrice SWOT
  3. profil de projet
  4. WBS
  5. RACI
  6. GANTT

---

## 7. CR -> annexes

- Gestion des données
  1. création du schéma : compliqué de tout prévoir dès le début, modifications en cours de route
  2. base de données plantes (construction, choix et concessions, doc pdf)
  3. détail des données de la base de données (données complémentaires à celles prévues initialement : polygone pour l'algo)  
faire un/des schéma visuel en 3NF  
schéma final
- Etapes de la conception de l'algorithmie
  - classe de terrain pour l'affichage des données
    1. Détail de toutes les choses importantes, besoins importants (voire séparer en 2 pour détailler plus certains algos)
    2. Premier jet simple et non optimisé pour les test -> vite pas assez performant
    3. Solution finale : création d'images
  - Algo suggestion
    1. beaucoup de petites fonctions à faire pour commencer / choix de travailler sur les id mais nécessité par exemple donc fonctions id to noms etc. (optimiser le nombre de requêtes sql)
    2. algorithme de suggestion pour une position = plutôt rapide à faire / 2 grandes étapes
    3. algorithme de suggestion ET de placement
      1. premier jet non optimisé pour se rendre compte des problématiques / travailler sur toutes les positions vides du jardin pour extraire seulement les positions acceptables ? : pas satisfaisant
      2. 4. problématique d'inclusion sur le site : quelques changements nécessaires (j'ai pas preshot certains trucs)
  - Optimisations faites / possibles encore, détail complexités  
lister plusieurs fonctions intéressantes et plusieurs tests possiblement sur chaque fonction
- Application Web  
!!!séparer en fonction des fonctionnalités
  - Gestion générale
    1. Utilisation de Blueprint pour l'organisation et la clarté
    2. gestion des sessions  
détailler plusieurs points / problématiques
    3. météo
    4. responsive / charte graphique / pages de créations de session etc.
  - Utilisateur normal  
détail pour chaque page importante / groupe de pages
    1. Affichage des informations pour l'utilisateur
      1. Son profil, jardiniers
      2. Ses parcelles
    2. Modification des infos importantes :
  - Admin
- Bilan et post mortem
  - rappel brefs des buts / cahier des charges et analyse par rapport au résultat
    1. fonctionnement global du site satisfaisant ?
    2. améliorations possibles, pistes de réflexions

- 
- problématique du manque d'expérience
    1. des erreurs auraient pu être évitées (exemples)
    2. problèmes indépendants rencontrés et possiblement mal gérés (SWOT?)
    3. rappeler but = se familiariser gdp etc.
  - Conclusion générale et mot de la fin
    1. bilans persos de chacun
    2. remerciements
  - Bibliographie
  - Table des figures

## Todo liste

Rédaction du rapport : répartition sur doc partagé google drive.



---

## Annexe 3 : Cahier des charges

---

### Cahier des Charges

Léo Fornoff

#### Fonctionnalités nécessaires de la solution

- Application web
  - Interface utilisateur
  - Interface administrateur
  - Facilité d'utilisation
- Algorithmique
  - Aide à l'agencement spatial de son jardin
  - Conseils et suggestions de jardinage dans le cadre de la permaculture
- Base de donnée
  - Permet de sauvegarder l'état des jardins de l'utilisateur
  - Gestion sécurisée de la connexion à son compte (interface utilisateur/ administrateur)

#### Les contraintes

- Temps, projet à rendre en Janvier 2023
- Devra utiliser :
  - de l'algorithmique
  - du web
  - de la base de données
  - Le tout devra être géré par la mise en place d'outils de gestion de projet
- Membres fixes de l'équipe projet
  - Maelan Tiger
  - Hélène Barbillon
  - Mathis Mangold
  - Léo Fornoff
- Fait dans un cadre éducatif
  - Temps en dehors des heures de cours
  - Premier projet d'école d'ingénieur
  - Chacun des membres devra travailler sur chaque aspect du projet
- Forte concurrence de la part des autres équipes projet

#### Réponse à un besoin

Ce projet doit répondre à une problématique environnementale visant à éviter le gaspillage des fruits et légumes dans nos jardins. Le projet devra s'inscrire dans une dimension locale et devra pouvoir être mis en oeuvre dans un cadre plus global.

Ce projet répond à un besoin organisationnel des jardins partagés.

#### Nos objectifs

- Fournir une application fonctionnelle qui facilite la gestion d'un jardin partagé
- Découvrir et mettre en application les notions de gestion de projet vu en cours
- Approfondir nos connaissances en BD, algorithmique et web

## Annexe 4 : Création de la base de données

```
CREATE TABLE utilisateur (
    id_user INTEGER PRIMARY KEY AUTOINCREMENT ,
    nom VARCHAR,
    prenom VARCHAR,
    mail VARCHAR,
    mdp VARCHAR
, img VARCHAR CONSTRAINT val_default DEFAULT "/static/images/photos_profil/gardener.png");
CREATE TABLE sqlite_sequence(name,seq);
CREATE TABLE jardin (
    id_jardin INTEGER PRIMARY KEY AUTOINCREMENT,
    code_postal varchar,
    ville varchar,
    numero_rue int,
    nom_rue varchar,
    id_referent,
    CONSTRAINT fk_referent
        foreign key (id_referent)
        references utilisateur(id_user)
        ON DELETE CASCADE
);
CREATE TABLE parcelle (
    id_parcelle INTEGER primary key AUTOINCREMENT,
    id_jardin int,
    id_user int,
    longueur_parcelle real,
    largeur_parcelle real, polygone varchar,

    CONSTRAINT fk_id_jardin
        foreign key (id_jardin)
        references jardin(id_jardin)
        ON DELETE CASCADE,
    CONSTRAINT fk_id_user
        foreign key (id_user)
        references utilisateur(id_user)
);
CREATE TABLE administre (
    id_user int,
    id_jardin int,
    CONSTRAINT pk
        PRIMARY KEY (id_user, id_jardin),
    CONSTRAINT fk
        FOREIGN KEY (id_user)
        references utilisateur(id_user)
        ON DELETE CASCADE ,
        FOREIGN KEY (id_jardin)
        references jardin(id_jardin)
        ON DELETE CASCADE
);
```

FIGURE 8.6 – Création de la base de donnée

```

);
CREATE TABLE est_separer_en (
    id_parcelle int,
    id_jardin int,
    CONSTRAINT pk
        PRIMARY KEY (id_parcelle, id_jardin),
    CONSTRAINT fk
        FOREIGN KEY (id_parcelle)
        references parcelle(id_parcelle)
        ON DELETE CASCADE ,
        FOREIGN KEY (id_jardin)
        references jardin(id_jardin)
        ON DELETE CASCADE
);
CREATE TABLE contient (
    id_parcelle int,
    id_plante int,
    x_plante int,
    y_plante int,
    CONSTRAINT pk
        PRIMARY KEY (id_parcelle, id_plante, x_plante, y_plante),
    CONSTRAINT fk
        FOREIGN KEY (id_parcelle)
        references parcelle(id_parcelle)
        ON DELETE CASCADE ,
        FOREIGN KEY (id_plante)
        references plante(id_plante)
        ON DELETE CASCADE
);
CREATE TABLE plante (
    id_plante int PRIMARY KEY,
    taille int,
    nom text unique,
    color varhar);
CREATE TABLE compagnons (
    plante1 int,
    plante2 int,
    constraint croissant check(plante1<plante2)
    CONSTRAINT pk
        primary key (plante1, plante2),
    CONSTRAINT fk
        FOREIGN KEY (plante1)
        references plante(id_plante)
        ON DELETE CASCADE ,
        FOREIGN KEY (plante2)
        references plante(id_plante)
        ON DELETE CASCADE
);
CREATE TABLE ennemis (
    plante1 int,
    plante2 int,
    constraint croissant check(plante1<plante2)
    CONSTRAINT pk
        primary key (plante1, plante2),
    CONSTRAINT fk
        FOREIGN KEY (plante1)
        references plante(id_plante)
        ON DELETE CASCADE ,
        FOREIGN KEY (plante2)
        references plante(id_plante)
        ON DELETE CASCADE
);

```

FIGURE 8.7 – Création de la base de donnée(suite)