

---

# Rapport Final PPII

**Clément GEHIN**  
**Romain NATANELIC**  
**Nicolas RENARD**

**Année 2022-2023**



# Déclaration sur l'honneur de non-plagiat

Je soussigné(e),

Nom, prénom : GEHIN Clement

Élève-ingénieur(e) régulièrement inscrit(e) en 3<sup>e</sup> année à TELECOM Nancy

Numéro de carte de l'étudiant(e) : 32010615

Année universitaire : 2022-2023

Auteur(e) du document, mémoire, rapport ou code informatique intitulé :

## Rapport Final du Projet Pluridisciplinaire d'Informatique Intégrative

Par la présente, je déclare m'être informé(e) sur les différentes formes de plagiat existantes et sur les techniques et normes de citation et référence.

Je déclare en outre que le travail rendu est un travail original, issu de ma réflexion personnelle, et qu'il a été rédigé entièrement par mes soins. J'affirme n'avoir ni contrefait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier texte ou code informatique, dans le but de me l'accaparer.

Je certifie donc que toutes formulations, idées, recherches, raisonnements, analyses, programmes, schémas ou autre créations, figurant dans le document et empruntés à un tiers, sont clairement signalés comme tels, selon les usages en vigueur.

Je suis conscient(e) que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université, et qu'en cas de manquement aux règles en la matière, j'encourrais des poursuites non seulement devant la commission de discipline de l'établissement mais également devant les tribunaux de la République Française.

Fait à Villers les Nancy, le 10 janvier 2023

Signature :

A handwritten signature in black ink, consisting of a stylized 'G' followed by 'CG'.



# Déclaration sur l'honneur de non-plagiat

Je soussigné(e),

Nom, prénom : NATANELIC Romain

Élève-ingénieur(e) régulièrement inscrit(e) en 3<sup>e</sup> année à TELECOM Nancy

Numéro de carte de l'étudiant(e) : 31914357

Année universitaire : 2022-2023

Auteur(e) du document, mémoire, rapport ou code informatique intitulé :

## Rapport Final du Projet Pluridisciplinaire d'Informatique Intégrative

Par la présente, je déclare m'être informé(e) sur les différentes formes de plagiat existantes et sur les techniques et normes de citation et référence.

Je déclare en outre que le travail rendu est un travail original, issu de ma réflexion personnelle, et qu'il a été rédigé entièrement par mes soins. J'affirme n'avoir ni contrefait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier texte ou code informatique, dans le but de me l'accaparer.

Je certifie donc que toutes formulations, idées, recherches, raisonnements, analyses, programmes, schémas ou autre créations, figurant dans le document et empruntés à un tiers, sont clairement signalés comme tels, selon les usages en vigueur.

Je suis conscient(e) que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université, et qu'en cas de manquement aux règles en la matière, j'encourrais des poursuites non seulement devant la commission de discipline de l'établissement mais également devant les tribunaux de la République Française.

Fait à Villers les Nancy, le 10 janvier 2023

Signature :





# Déclaration sur l'honneur de non-plagiat

Je soussigné(e),

Nom, prénom : RENARD Nicolas

Élève-ingénieur(e) régulièrement inscrit(e) en 3<sup>e</sup> année à TELECOM Nancy

Numéro de carte de l'étudiant(e) : 31912191

Année universitaire : 2022-2023

Auteur(e) du document, mémoire, rapport ou code informatique intitulé :

## Rapport Final du Projet Pluridisciplinaire d'Informatique Intégrative

Par la présente, je déclare m'être informé(e) sur les différentes formes de plagiat existantes et sur les techniques et normes de citation et référence.

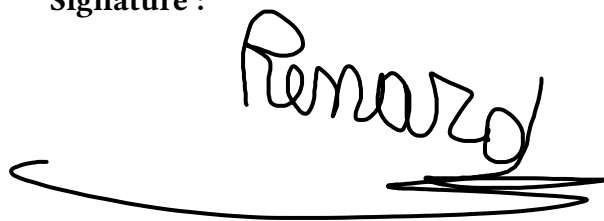
Je déclare en outre que le travail rendu est un travail original, issu de ma réflexion personnelle, et qu'il a été rédigé entièrement par mes soins. J'affirme n'avoir ni contrefait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier texte ou code informatique, dans le but de me l'accaparer.

Je certifie donc que toutes formulations, idées, recherches, raisonnements, analyses, programmes, schémas ou autre créations, figurant dans le document et empruntés à un tiers, sont clairement signalés comme tels, selon les usages en vigueur.

Je suis conscient(e) que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université, et qu'en cas de manquement aux règles en la matière, j'encourrais des poursuites non seulement devant la commission de discipline de l'établissement mais également devant les tribunaux de la République Française.

Fait à Villers les Nancy, le 10 janvier 2023

Signature :

A handwritten signature in black ink, appearing to read 'Renard', with a long horizontal flourish underneath.





---

# Rapport Final PPII

**Clément GEHIN**  
**Romain NATANELIC**  
**Nicolas RENARD**

**Année 2022-2023**

Clément GEHIN  
Romain NATANELIC  
Nicolas RENARD

[clement.gehin@telecomnancy.eu](mailto:clement.gehin@telecomnancy.eu)  
[romain.natanelic@telecomnancy.eu](mailto:romain.natanelic@telecomnancy.eu)  
[nicolas.renard@telecomnancy.eu](mailto:nicolas.renard@telecomnancy.eu)

TELECOM Nancy  
193 avenue Paul Muller,  
CS 90172, VILLERS-LÈS-NANCY  
+33 (0)3 83 68 26 00  
[contact@telecomnancy.eu](mailto:contact@telecomnancy.eu)



# Remerciements

*Ce rapport a pour finalité de retracer nos travaux durant le Projet Pluridisciplinaire d'Informatique Intégrative.*

*A ce titre, nous tenons à remercier tous les professeurs encadrant, pour leur accompagnement et le partage de leurs connaissances tout au long de ce projet.*

–Clément, Nicolas et Romain



# Table des matières

<b>Remerciements</b>	<b>ix</b>
<b>Table des matières</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Conception de l'application</b>	<b>3</b>
2.1 Base de données . . . . .	3
2.2 Serveur web . . . . .	5
2.2.1 Architecture MVC . . . . .	6
2.2.2 Routing . . . . .	7
2.3 Algorithmes de traitement . . . . .	8
2.3.1 Algorithme de recherche optimisé . . . . .	8
2.3.2 Barre de recherche . . . . .	9
<b>3 Tests et performances</b>	<b>10</b>
3.1 Tests . . . . .	10
3.2 Performances . . . . .	11
<b>4 Gestion de projet</b>	<b>12</b>
<b>5 Conclusion</b>	<b>15</b>
<b>Bibliographie / Webographie</b>	<b>17</b>
<b>Liste des illustrations</b>	<b>19</b>
<b>Glossaire</b>	<b>21</b>
<b>Résumé</b>	<b>23</b>
<b>Abstract</b>	<b>23</b>



# 1 Introduction

Dans le cadre du premier Projet Pluridisciplinaire d'Informatique Intégrative (PPII) proposé par l'établissement, nous allons parcourir en détails le travail que nous avons tous les trois effectué. Cette année, Télécom Nancy, nous a proposé de traiter des sujets de jardins partagés, des micro-fermes, ainsi que tout ce qui se rapproche des Associations pour le Maintien d'une Agriculture Paysanne.

Etant donné l'augmentation de la population et la prolifération actuelle du gaspillage, ce à quoi nous pouvons ajouter le fait que les denrées deviennent de plus en plus rares; nous avons opté pour le développement d'une application favorisant les jardins partagés.

En effet, avec Parta'Jardin, le but de notre initiative est de réunir deux générations, étudiants et retraités, afin qu'ils puissent partager de véritables moments de convivialité, tout en évitant le gaspillage alimentaire. Grâce à cette application, apparentée à un réseau social inter-générationnel, Les plus anciens peuvent mettre à dispositions leurs cultures pour la nouvelle génération, en perte continuelle de pouvoir d'achat.

A travers ce rapport, nous allons pouvoir revenir sur la conception et le développement de notre idée, dans tous ses aspects : De la réalisation dans tous les domaines informatiques (Gestion de bases de données, algorithmie, ...), jusqu'à la partie gestion de projet, en passant par la phase performances et tests.





## 2 Conception de l'application

Tout d'abord, abordons le chapitre technique consacré au développement de l'application. Nous avons dû effectuer notre projet dans un cadre technique imposé, et donc comportant des contraintes. En effet, nous avons dû utiliser la structure **Flask**[1], qui nous oblige de coder avec le langage **Python**[4]. Nous allons voir dès à présent que cela avait des répercussions sur l'ensemble de notre réalisation, à commencer par les bases de données.

### 2.1 Base de données

En ce qui concerne la présence des informations au sein de notre application, nous avons le choix sur ce que nous voulions utiliser.

Nous avons 3 solutions possibles : Une base de données MySQL[2] MariaDB, une base de données PostgreSQL[3] ou une base de données SQLite. Nous avons vu en cours des bases de données PostgreSQL et SQLite.

Cependant, nous avons très vite éliminé la base de données PostGre car c'était celle avec laquelle nous avions le moins d'expérience. Ensuite, la base de données SQLite proposait une solution beaucoup plus simple et rapide à mettre en place bien qu'il y avait plusieurs contraintes : Tout d'abord les données étaient plus difficiles à visualiser car nous ne pouvions les afficher que dans la console. Ensuite la manière d'ajouter des données de manière rapide pour avoir un jeu de données complet pour utiliser notre application était relativement complexe et lent. Enfin nous avons estimé que le temps que nous passerions à implémenter une base de données MySQL MariaDB sur notre projet serait utile car son utilisation par la suite nous ferait gagner du temps.

C'est donc pour cela que nous avons choisi d'utiliser une base de données MySQL MariaDB sur notre projet. En voici un schéma en figure 2.1, représentant les différentes relations entre les tables :

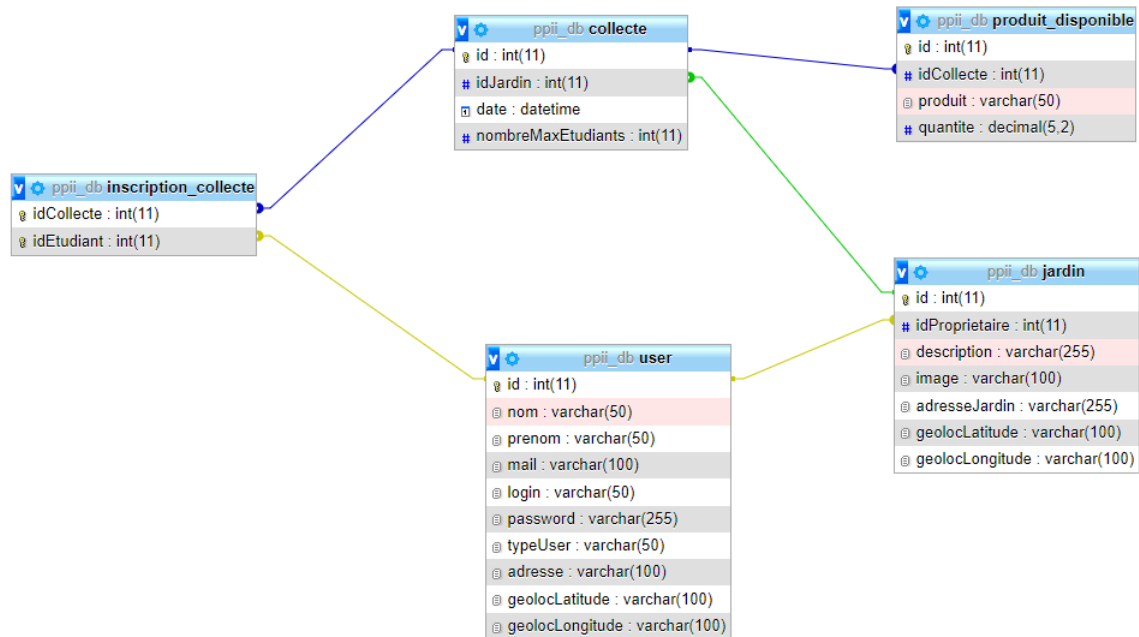


FIGURE 2.1 – Schéma de notre base de données

Chaque utilisateur possède un identifiant unique afin d'autoriser les homonymes. Les utilisateurs verront également login et mot de passe stocké au sein de la base. Leur mot de passe sera hashé pour des questions de sécurité. La localisation de l'utilisateur est conservé en base à partir de son adresse.

Chaque jardin possède un identifiant unique et est relié à un seul utilisateur. Il possède également un lien vers une image et une adresse qui servira pour la localisation tout comme l'utilisateur. Un jardin peut posséder des collectes.

Chaque collecte a un identifiant unique et est reliée à un jardin avec une date et un nombre d'étudiant maximum. Une collecte peut avoir plusieurs produit disponible avec des produits et une quantité disponible.

Chaque étudiant peut s'inscrire à une collecte de son choix.

Chacun des champs de ces tables ont des tailles définies afin de pouvoir limiter la taille de la base dans le cas où nous aurions une grande quantité de données. Les champs comme idEtudiant présent dans la table inscription\_collecte sont des clés étrangères.

## 2.2 Serveur web

Concernant la partie Web, comme nous l'avons évoqué dans l'introduction auparavant, la consigne nous imposait d'utiliser le micro-framework Flask, basé sur Python. En effet, on utilise ici le terme de micro-framework, car le but de Flask est de garder une architecture simple. Parmi les fonctionnalités qui nous ont été utiles avec Flask, on retrouve la simplification des tests unitaires, la fonction de debugger, mais surtout celle de serveur Web de développement. En effet, grâce à Flask, nous pouvons visualiser directement notre site, avec les ajustements apportés en temps réel.

Le web s'appuie sur la partie visuelle, le frontend, mais surtout sur la partie immergée de l'iceberg : le backend. En effet, une application web comme la nôtre inclue de nombreuses fonctionnalités, qui sont permises par ce fameux backend, à commencer par la partie inscription/connexion.

Grâce au serveur web géré par Flask, nous avons pu assembler et "donner vie" à notre application. En effet, nous avons créé une page d'inscription associée à une page de connexion. Une fois le formulaire rempli, nous arrivons sur la page d'accueil, qui résume les informations de notre utilisateur.

Par la suite, nous avons accès, grâce à la barre de navigation, aux différentes pages web suivantes :

- Profil (pour consulter ses informations)
- Recherche Optimisée (permettant de voir le chemin le plus court afin de récupérer toutes mes collectes) basé sur l'algorithme du voyageur de commerce (nous y reviendrons par la suite)
- Visualisation Collecte (pour voir les collectes créées par l'utilisateur ou auxquelles ce dernier est inscrit)
- Visualisation Jardin (pour voir dans quels jardins je suis inscrit)

Il faut savoir que tous ces visuels sont donc permis grâce au serveur Web, qui permet l'affichage des fichiers écrits dans le langage HTML, et "bonifiés" par l'apport des fichiers CSS, responsable des couleurs, images, etc... des pages. Pour les utilisateurs, c'est cette partie la plus importante, afin qu'ils aient accès à notre contenu.

### 2.2.1 Architecture MVC

Pour cette partie visuelle, nous avons fait le choix d'utiliser l'architecture applicative MVC (Modèle, Vue, Contrôleur), la raison principale étant le besoin de "???" et la lisibilité du projet. Concrètement nous avons implémenté des contrôleurs qui sont la logique de base des pages avec différents sous-manager pour ces modèles simplifiant l'écriture globale du modèle (écriture des fonctions lourdes dans le manager, la fonction principale est dans le contrôleur).

Concernant le modèle, nous nous reposons sur les objets entités proposés par SQLAchemy qui est un ORM (Object Relational Mapping) proposant diverses fonctions d'un modèle (add, delete, etc). Grâce à cet ORM nous pouvons récupérer directement nos données sous forme d'objets ce qui est extrêmement pratique (en ayant évidemment auparavant indiqué le schéma voulu à l'ORM) :

```
1 class Collecte(Base):
2     __tablename__ = "collecte"
3
4     id = Column(Integer, primary_key=True)
5     idJardin = Column(Integer, ForeignKey('jardin.id'), nullable=False)
6     date = Column(DateTime, nullable=False)
7     nombreMaxEtudiants = Column(Integer, nullable=True)
8
9     def __init__(self, id: int, idJardin: int, date: datetime,
10 nombreMaxEtudiants: int):
11         self.id = id
12         self.idJardin = idJardin
13         self.date = date
14         self.nombreMaxEtudiants = nombreMaxEtudiants
15
16     def __repr__(self):
17         return f'<Collecte {self.id!r}>'
```

Enfin pour l'implémentation de la vue, nous utilisons la bibliothèque Jinja2 qui est un moteur de template. Grâce à celui-ci, nous avons simplement à créer nos pages HTML, les ajouter dans le dossier que nous avons référencé comme dossier source pour nos templates et réaliser ce qu'on appelle un rendu du template. Jinja2 est particulièrement pratique car il nous permet d'injecter des données directement dans notre template.

Chaque fichier de notre application a maintenant une fonction plus précise.

Prenons un exemple : Un utilisateur est sur une page où il a une saisie possible qui demande une modification dans une base de données, l'utilisateur décide de saisir son information et de la sélectionner, le schéma d'appel de notre système va se comporter comme tel :

- Action de l'utilisateur sur la vue -> Demande une action au contrôleur de modifications de données
- Le contrôleur reçoit cette demande, il doit demander une modification de données, il s'adresse donc au modèle -> Demande de modification du modèle
- Le modèle reçoit la demande du contrôleur, effectue la demande de modification de données -> notifie le contrôleur de la fin de sa modification afin qu'il puisse continuer son traitement
- Le contrôleur reçoit la notification, poursuit ses éventuels traitements et enfin, rend l'information à la vue.

### 2.2.2 Routing

L'utilisation du site est rendue plus simple grâce à une navigation simplifiée dite "Navigation par route" utilisant la bibliothèque Workzeug. Via cette bibliothèque nous pouvons simplifier l'écriture de nos routes. Cela permet une embellie du site en masquant la localisation précise du template dans l'URL, par exemple une route qui aurait été nommée "Localhost/static/pages/ma-page.html" peut être directement accédée via l'url "Localhost/ma-page" si le développeur le souhaite. Il suffit d'indiquer à l'application que le template nommé ci-dessus est pointé lors de l'appel à l'URL "Localhost/ma-page".

Cela est particulièrement pratique dans des applications qui sont destinées à des utilisateurs novices. Une URL "Propre" est source de confiance et répond à une UI/UX plus poussée.

## 2.3 Algorithmes de traitement

En ce qui concerne les algorithmes, nous avons choisi d'étudier l'algorithme de recherche de cycle optimisé et le fonctionnement d'une barre de recherche.

### 2.3.1 Algorithme de recherche optimisé

Cet algorithme correspond à un problème bien connu nommé "Le voyageur de commerce". Ce problème consiste à trouver le chemin le plus court pour parcourir une liste de sommets avec une distance qui sépare chaque sommet.

Il y a de nombreuses manières de traiter ce problème, par exemple la manière "brute force" qui consiste à tester toutes les solutions possibles. La solution que nous avons choisi d'implémenter est un algorithme d'insertion.

En quoi consiste-t-il ?

Cet algorithme va prendre 2 points pour commencer. Par la suite il va prendre un 3ème point et tester la position à laquelle on obtient le meilleur résultat (distance à parcourir la plus faible). Par exemple, comme on peut le voir sur la figure 4.2, on a le chemin de point 0-1 et l'on doit placer le point 2. On va donc essayer les différentes possibilités de placement. Une fois que le chemin le plus court pour ces 3 points aura été trouvé, on reproduira l'algorithme sur ce chemin et un 4ème point, etc ...

01 → 2 ? : 012 ou 021 ou 201

FIGURE 2.2 – Exemple algorithme d'insertion

Dans le cas de cet algorithme, il y a un point à prendre en compte si l'on veut optimiser au maximum la solution :

On sait qu'à la fin du parcours, on revient à la position initiale, il n'est donc pas nécessaire de tester la dernière position pour le point en cours si l'on a testé la position initiale.

Si on se penche du côté de la complexité algorithmique, cet algorithme possède une complexité de  $\Theta(n^2)$ . La boucle la plus externe va tester pour chaque point que l'on veut intégrer dans le parcours, donc  $n$  points. Ensuite pour chaque point que l'on veut insérer, on va tester pour tous les points appartenant au chemin déjà construit.

Nous sommes dans le cas d'un algorithme qui peut se représenter sous la forme suivante :

```
1 for i in range(2,n):  
2     for j in range(0,i):  
3         testerPositionActuelle()  
4
```

Il existe des algorithmes "complémentaires". C'est-à-dire qu'ils vont pouvoir potentiellement améliorer le résultat trouvé au dépit de nouvelles opérations à effectuer. Nous aborderons plus en détail ce point dans la partie "Performance".

### 2.3.2 Barre de recherche

La page Adhésion Collecte dispose d'une fonctionnalité de recherche portant sur deux critères, afin d'y arriver, nous séparons les deux critères en utilisant la fonction "Split(',')". Une fois les deux critères obtenus, nous les injectons dans deux requêtes SQL distinctes.

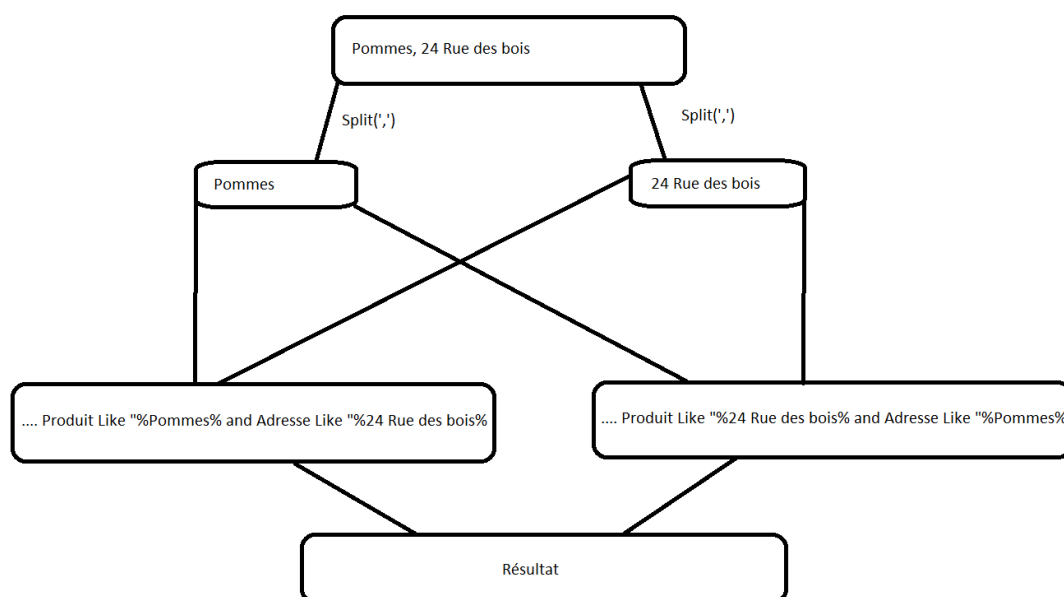


FIGURE 2.3 – Exemple de fonctionnement de la recherche

Dans le cas où nous n'avons qu'un seul ou aucun critère, les requêtes réalisent des clauses `<WHERE *** LIKE "%'vide'%">`

## 3 Tests et performances

### 3.1 Tests

Afin de mieux gérer l'intégration et la fiabilité du projet, nous avons mis en place différents tests portant sur la vérification de l'ajout des données. Concrètement, nous avons utilisé la bibliothèque "unittest" de python qui permet de réaliser des tests unitaires. Les commandes utilisées sont principalement :

- `assertIsInstance` (paramètre à vérifier, donnée qu'on doit obtenir). Vérifie que le paramètre est du même type que la donnée qu'on doit obtenir. (vérifié quand : `int = int`, `Objet = Objet` etc.)
- `assertEqual` (paramètre à vérifier, donnée qu'on doit obtenir). Vérifie que la valeur affectée du paramètre est la même que celle voulue. (vérifié quand : `1=1` , `"abcd"="abcd"`)
- `assertNotEqual` (paramètre à vérifier, donnée qu'on doit obtenir). Vérifie que la valeur affectée du paramètre est différente que celle voulue. (vérifié quand : `4!=1` , `"abcd"!="efgh"`)

Voici par exemple la rédaction d'un test :

```
1 class CollecteTest(unittest.TestCase):
2     collecteToTest = Collecte(4, 1, datetime(21, 4, 2001), 2)
3
4     def testTypeUser(self):
5         self.assertIsInstance(self.collecteToTest, Collecte)
```

Ces tests nous permettent de nous assurer de la cohérence des données conformément au modèle.



## 3.2 Performances

Sur la partie Performance, nous allons nous concentrer sur l'algorithme de recherche optimisé.

Tout d'abord nous avons choisi de tester la solution dite "brute force". C'était avant tout la solution la plus basique et la plus simple à mettre en place. On peut voir l'évolution du temps en fonction du nombre de collectes sur la figure 3.1 présente ci-dessous.

Nombre de collectes	10	11	12	13	14	15
Temps nécessaire	0,226	2,694	34,04	487,632	>7200	ERREUR

FIGURE 3.1 – Evolution du temps en fonction du nombre de collecte

On peut constater que le temps de calcul dépasse très vite les 30 secondes, puis 8 minutes, puis plus de 2 heures. Cet algorithme est viable pour des listes très courtes, par exemple 10 points, avec un temps d'opération inférieur à 1 seconde.

Même si dans notre cas, nous aurions souvent eu des listes inférieures à 10 points, nous avons dû choisir une solution plus optimisée avec la recherche par insertion.

Sur la figure 3.2 ci-dessous on peut voir que les différents temps d'opérations sont bien plus faible que l'algorithme brute force et que le nombre de points traités est bien plus élevé.

Nombre de collectes	10	20	50	100	200	400	500	777
Temps nécessaire	0	0	0	0,01	0,025	0,164	0,3	1,079

FIGURE 3.2 – Evolution du temps en fonction du nombre de collecte (optimisé)

On remarque très vite une différence entre les 2 algorithmes :

Le premier entre en erreur à partir de 15 collectes car il y a trop de possibilités à tester alors que le second réalise les opérations pour plus de 700 collectes en un temps légèrement supérieur à 1 seconde.

En terme de performance, on peut noter un gain non négligeable dès 10 collectes. Avant ce nombre de collectes, les 2 algorithmes ont les mêmes résultats à la seconde près, qu'on peut négliger du fait du temps de rendement de la page web, du temps de réponse de notre serveur, etc ...

## 4 Gestion de projet

Enfin, nous allons conclure ce rapport par le chapitre traitant de la gestion de projet.

Dans un projet informatique comme le nôtre, la gestion de projet va consister en l'ensemble des activités qui permettent de planifier, de coordonner et de contrôler les ressources (humaines, matérielles, immatérielles (et financières)) nécessaires à la réalisation de notre projet. De plus, elle consiste à définir les objectifs du projet, à établir un plan d'action, c'est-à-dire ici de réaliser un planning, en assignant les tâches aux membres de l'équipe et à surveiller l'avancement du projet pour s'assurer que nous respectons bien les délais que nous nous sommes imposés. Nous avons dû également apprendre à gérer les problèmes potentiels qui peuvent intervenir au cours de la réalisation du projet.

Nous avons débuté notre projet par la réalisation d'un état de l'art. Cela signifie que nous avons fait de nombreuses recherches afin de comparer les différentes solutions qui ont d'ores et déjà vu le jour. A partir de ces recherches, nous avons pu analyser ce qui n'existait pas encore sur le marché, et dans le même temps les "critères essentiels" afin de réaliser une bonne application sur le sujet des jardins partagés.

Avec ce travail, nous avons pu réaliser nos premières réunions, et ainsi retenir les points-clés sur lesquels nous allions nous appuyer, préparer le planning des rendus, ainsi que la répartition des tâches. Pour ce faire, nous avons réalisé un **diagramme de Gantt**, répertoriant toutes les missions que nous avons à réaliser, et les échéances que nous avons à respecter.

Concernant la répartition, il y avait 3 grandes parties de développement, comme nous l'avons vu auparavant. Venant tous les trois de DUT Informatique, nous étions relativement à l'aise dans chaque domaine, même si nous avons nos forces et nos faiblesses. C'est donc dans cette optique que nous avons décidé de mettre l'un de nous comme "personne à consulter" dans chacun des domaines :

- Base de données : Romain
- Frontend (Partie visuelle du web) : Clement
- Backend (Partie non-visible de l'application) : Nicolas

Nous avons pu mettre en place un diagramme de Gantt, nous permettant de planifier efficacement chacune de nos tâches, leurs dates de début et de fin, ainsi que l'enchaînement de ces dernières.

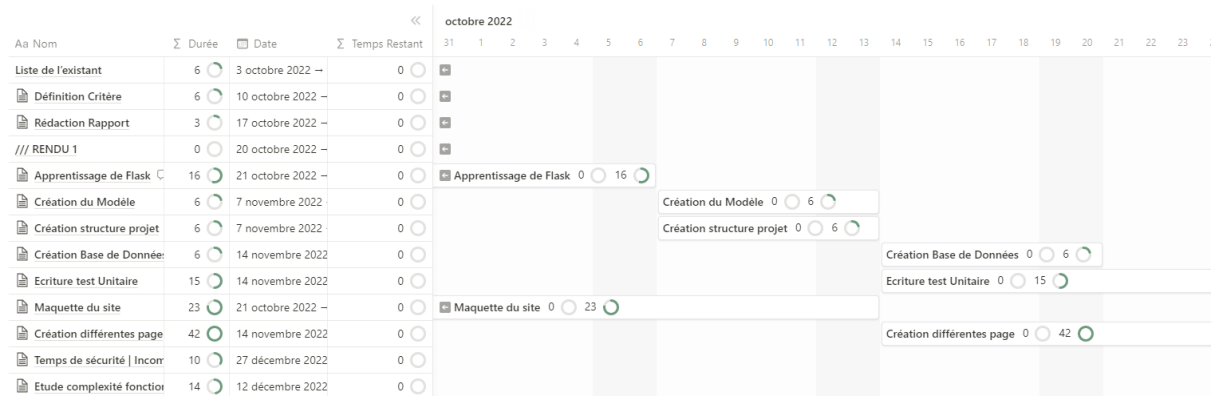


FIGURE 4.1 – Diagramme de Gantt

Pour réaliser cette planification, il a également été nécessaire de réaliser l'analyse de nos forces et faiblesses, en tant que groupe et en tant qu'individus, notamment grâce à **l'analyse SWOT**. Cette méthode permet de voir les points faibles/forts du projet que nous sommes en train de réaliser.



FIGURE 4.2 – Exemple de matrice SWOT



## 5 Conclusion

Pour conclure, nous avons réalisé une application sur les jardins partagés, à l'aide de Flask. Nous avons tout d'abord étudié le sujet, et nous nous sommes inspirés de l'existant. A partir de cela, nous avons pu mettre en oeuvre le projet, en reprenant les points essentiels de chaque site web analysé ; Nous nous sommes répartis les différentes missions en fonction de nos faiblesses et de nos forces respectives.

Nous avons alors conçu une application réunissant étudiants et personnes âgées, sur laquelle il faut s'inscrire et se connecter. Par la suite, il est possible de créer ou participer à des événements, comme la récolte de pommes dans un verger, ou simplement une dégustation de raisins dans le jardin d'un particulier. Le but de notre application était essentiellement social, nous voulions réunir les différentes générations, dont les relations n'ont pas toujours au beau fixe.

Nous avons pour ce faire, utilisé des bases de données, un serveur Web grâce à Flask, ainsi que le langage de programmation Python. De plus, afin de mener à bien notre projet, nous avons pu nous appuyer sur les outils de gestion de projet informatique, tels que la matrice SWOT, l'état de l'art, le diagramme de Gantt, ...

Sur ce projet, nous avons également certains points que nous aurions aimé réaliser. Par exemple, nous aurions aimé réaliser une partie blog afin que les différents clients puissent communiquer sur des sujets. Nous aurions également voulu mettre en place une partie administration afin de pouvoir gérer tous les clients.

Cette expérience nous a beaucoup appris, notamment à travailler en groupe, mais également en autonomie, tout en mettant en oeuvre les méthodes de gestion de projet. D'un point de vue technique, nous avons également pu apprendre à utiliser et à maîtriser davantage les technologies énumérées tout au long de ce rapport.



## Bibliographie / Webographie

- [1] Flask. Documentation flask. <https://flask.palletsprojects.com/en/2.2.x/>, 2010. 3
- [2] Oracle. Documentation mysql. <https://dev.mysql.com/doc>, 2023. 3
- [3] PostGre. Documentation postgresql. <https://www.postgresql.org/docs/>, 1996. 3
- [4] Python. Documentation python. <https://docs.python.org/fr/3/>, 2001. 3





## Liste des illustrations

2.1	Schéma de notre base de données . . . . .	4
2.2	Exemple algorithme d'insertion . . . . .	8
2.3	Exemple de fonctionnement de la recherche . . . . .	9
3.1	Evolution du temps en fonction du nombre de collecte . . . . .	11
3.2	Evolution du temps en fonction du nombre de collecte (optimisé) . . . . .	11
4.1	Diagramme de Gantt . . . . .	13
4.2	Exemple de matrice SWOT . . . . .	13



# Glossaire

**backend** Terme désignant un étage de sortie d'un logiciel devant produire un résultat. On l'oppose au front-end qui lui est la partie visible de l'iceberg. 5

**Flask** Micro framework open-source de développement web en Python. Il est classé comme microframework car il est très léger. Flask a pour objectif de garder un noyau simple mais extensible. 5

**frontend** Correspond aux productions HTML, CSS et JavaScript d'une page internet ou d'une application qu'un utilisateur peut voir et avec lesquelles il peut interagir directement. 5

**ORM** type de programme informatique qui se place en interface entre un programme applicatif et une base de données relationnelle pour simuler une base de données orientée objet. 6



# Résumé

Nous avons un peu plus de 3 mois afin de réaliser une application à l'aide du framework Python Flask, sur le sujet des jardins partagés. Afin de mener ce projet pluridisciplinaire à bien, nous étions 3 et avons dû nous répartir le travail, afin de nous organiser face aux échéances, malgré les périodes de cours et le temps demandé par les autres matières. Dans cette optique, nous avons mis à profit nos connaissances dans le domaine de la gestion de projet, afin d'évaluer les besoins de notre application, nos forces, nos faiblesses, les échéances importantes, ... Toutes les données nécessaires afin de mener à bien un projet informatique.

Du point de vue de la réalisation cette fois, nous avons pu mettre à profit os connaissances acquises lors de nos formations de DUT informatique (et de licence professionnelle pour Nicolas). Nous avons travaillé avec une base de données MySQL, le serveur Web de Flask nous a permis de montrer notre maîtrise des langages frontal web HTML et CSS. Enfin le langage Python nous a permis de réaliser toute la partie "fonctionnelle" de l'application, la "face cachée" de l'iceberg.

Ce projet nous a permis à tous les trois d'engranger beaucoup d'expérience, que ce soit d'un point de vue technique, mais surtout sur l'apport de la gestion de projet informatique en groupe.

**Mots-clés : Equipe, Autonomie, Projet informatique, expérience**

# Abstract

We had a little more than 3 months to create an application using the Python framework Flask, on the subject of shared gardens. In order to carry out this multidisciplinary project, we were 3 and had to divide the work between us, in order to organize ourselves in front of the deadlines, in spite of the periods of courses and the time required by the other subjects. In this perspective, we used our knowledge in the field of project management to evaluate the needs of our application, our strengths, our weaknesses, the important deadlines, ... All the necessary data to carry out an IT project.

From the point of view of the realization this time, we were able to take advantage of our knowledge acquired during our technical degree in computer science (and a professional license for Nicolas). We worked with a MySQL database, the Flask Web server allowed us to show our mastery of the HTML and CSS front-end languages. Finally, the Python language allowed us to realize the "functional" part of the application, the "hidden face" of the iceberg.

This project allowed to all of us to gain a lot of experience, from a technical point of view, but especially on the contribution of the management of computer project by group.

**Keywords :**