TELECOM Nancy

AMPLET
Démocratie Participative

Auteurs :
Malo DAMIEN
Louis-Vincent CAPELLI
Jules BRUNET
Tristan SMAGGHE

Responsable de module :
Olivier FESTOR

6 janvier 2022

Table des matières

Chapitre 1

Présentation du projet

1.1 Introduction au sujet

Récemment on a vu apparaître un nouveau type de service informatique destiné à renforcer l'engagement citoyen : La CIVIC TECH qui regroupe par exemple : la consultation citoyenne, pétitions, espaces de discussion et de création. Cette technologie a pour but de renforcer l'implication quotidienne des citoyens dans la vie démocratique à plusieurs échelles (locale, départementale, régionale et nationale). En groupe de projet nous avons donc décidé de développer une application qui permet à chacun de participer à la vie démocratique. Dans le cadre du module CS54 (Computer Science 54) de la première année du cycle ingénieur sous statut étudiant de TELECOM Nancy.

L'objectif est d'implémenter une application de démocratie participative pour une ville. Le travail s'est décomposé en trois parties : Le développement web, l'utilisation et la création d'une base de données et enfin l'implémentation d'algorithmes de traitement avancés.

1.2 Problématiques soulevées

Ainsi le sujet pose la problématique suivante : Comment, au travers d'une application web, renforcer l'engagement des citoyens dans la vie démocrative de la localité ?

Défini ainsi, le sujet est très vaste et nous a dans un premier lieu semblé difficile à aborder. En effet, il s'agissait d'abord pour nous de comprendre qu'est ce que la démocratie participative et plus particulièrement quel rôle jouent les Civic Tech dans le développement de cette dernière.

1.3 Hypothèses de solution

C'est pourquoi il a été, dans un premier temps, laborieux de trouver une idée satisfaisante de projet. Dès le début nous avions eu l'idée d'une plateforme de proposition de projets pour la commune et de discussions autour de ces derniers. Cependant, l'idée était très basique (et surtout manquait d'originalité), et reflétait notre manque de compréhension de ce qu'étaient les Civic Tech. Après plus de recherches, nous avons commencé à entrevoir de nouvelles pistes pour notre projet, plus axées sur la consultation de l'avis des citoyens et de leurs besoins, sur l'entraide entre citoyens. C'est ainsi que nous sont venues 3 idées principales :

— Un réseau social où les citoyens peuvent parler entre eux à propos des élections, projets démocratiques à venir et gagner des points en participant ;
— Un système de lignes de bus participatives, où les citoyens entrent les lieux qu'ils visitent le plus souvent et à quels horaires ils ont besoin d'y être transportés, pour ensuite proposer des lignes de bus adaptées ;
— Une application semblable à du covoiturage, mais pour les courses chez des commerçants locaux ;

Assez rapidement, nous nous sommes accordés à dire que la troisième idée était à la fois la plus accessible, mais aussi la plus en accord avec le sujet selon nous. C'est ainsi qu'est né Amplet !

1.4 Présentation d'Amplet

Nous avons donc décidé de développer un site web dont le nom est : Amplet. Il s'agit d'une plateforme d'entraide pour l'achat de produits frais locaux, mais également tout autre type de produits disponibles dans des commerces locaux. Il permet aux communes d'avoir une interface web pour fournir un système de navettes qui rempliraient cette fonction, en choisissant les commerces visités grâce à un système de votes. Cela permettrait de créer un écosystème démocratique pour que chacun puisse à la fois aider et profiter des services que propose

l'application. L'application permettrait également à n'importe qui de réaliser des courses pour les personnes qui ne le peuvent pas et/ou qui n'en auraient pas le temps.

L'objectif d'Amplet est de s'assurer que chaque commerçant finira par être désigné, de sorte à accompagner même les plus petits des commerces, et ce même s'ils ne sont pas les plus demandés. Ainsi, Amplet permet de redynamiser l'économie locale tout en renforçant la cohésion citoyenne.

Chapitre 2

État de l'art

2.1 L'apparition de la Civic Tech

Les premières Civic Tech font leur apparition à la fin des années 90, sous formes de forums autour de la démocratie participative et de la vie de quartiers, de sites de crowdfunding, d'applications de votes ou de consultation de résultats d'élections, ... Elles n'étaient alors que le fruit de petites initiatives isolées, mais vers la fin des années 2000 / début des années 2010, un véritable mouvement de promotion des Civic Tech a commencé. (cf. ref. [2])

Par exemple, en 2011, une organisation mondiale de la Civic Tech (Open Government Partnership) fut créée par plusieurs gouvernements : Brésil, Indonésie, Mexique, Norvège, Philippines, Afrique du Sud, Royaume-Unis et les États-Unis. Leur objectif est de fournir des feuilles de routes pour les gouvernements, que ceux-ci seront chargés de tenir, afin de respecter la vision de l'organisation. Ils souhaitent notamment rendre les gouvernements plus transparents, responsables et plus sensibles à leurs citoyens. Ils ont comme but final d'améliorer les services et les stratégies gouvernementales (cf. ref. [6]).

2.2 La Civic Tech aujourd'hui

De nos jours, les Civic Tech se sont démocratisées et nous accompagnent au quotidien (application liée au réseau de bus, par exemple). Elles sont souvent classifiées selon deux types d'objectif :

— Améliorer la transparence du gouvernement ;
— Améliorer la cohésion et les intéractions entre les citoyens.

On peut noter qu'il existe aussi une classification sous trois types de Tech :

— La Civic Tech, visant à plus de participation citoyenne ;
— La Pol Tech, qui concerne la politique en général et l'amélioration des fonctions démocratiques du gouvernement ;
— La Gov Tech, qui concerne les outils qui aident les gouvernements dans leur quête de transparence.

Mais il existe aussi de nombreux autres critères qui permettent de différencier les Civic Tech les unes des autres :

— Si l'application est à but lucratif ou non ;
— Le degré de changement auquel pousse l'utilisation de l'application (son impact) ;
— Le degré d'intéractions sociales, entre citoyens, qu'elle engendre ;
— Le niveau de technologie sur lequel elle repose.

Particulièrement, le but lucratif ou non d'une application peut avoir une grande importance, puisque cela peut venir remettre en question le but première de la Tech, en particulier si elle reste ou non dans le cadre d'une Civic Tech. (cf. ref. [3])

À l'aide de ces critères, on va pouvoir comparer quelques Civic Tech :

— CovidTracker, une application permettant de suivre l'épidémie de Coronavirus ;
— CitizenLab, une plateforme de consultation citoyenne destinées aux collectivités locales ;
— Stan, une application permettant de suivre en temps réel les horaires de bus ;
— BlaBlaCar, une application de covoiturage entre particuliers ;
— Shopopop, une application qui propose à des particuliers de se faire livrer par d'autres particuliers ou les commerçants eux-mêmes ;

Ainsi, cette étude de différentes Civic Tech nous permet de confirmer l'idée que nous ne faisons d'Amplet :

— Objectifs : Cohésion & services proposés par la Mairie ;
— Impact : Modéré, local ;

— But : Non lucratif ;
— Niveau technologique : Accessible.

Application	Objectif	But lucratif	Impact	Intéractions sociales	Niveau technologique
CovidTracker	Informer et transparence	Non	Élevé (Sécurité)	Aucunes	Élevé
CitizenLab	Consultation citoyenne	Oui	Modéré (Communication)	Élevées	Modéré
Stan	Information Citoyenne	Non	Faible (Communication)	Faibles	Modéré
BlaBlaCar	Améliorer la cohésion	Oui	Modéré (Environnement)	Élevées	Modéré
Shopopop	Améliorer la transparence	Oui	Modéré (Environnement)	Élevées	Modéré

FIGURE 2.1 – Tableau comparatif de différentes Civic Tech

Chapitre 3

Développement

3.1 Intégration en continue

Afin de pouvoir vérifier facilement les différentes versions du projet, et pour pouvoir vérifier depuis n'importe quelle plateforme, nous avons mis en place une intégration en continue. Elle utilise Docker, logiciel permettant de créer des conteneurs qui ne dépendent pas de la machine hôte (évite le "mais ça fonctionne sur ma machine"). Son fonctionnement est assez simple :

- Lorsqu'un membre du groupe décide de publier une nouvelle version il crée un "Tag" sur Gitlab (plateforme qui héberge le projet) ;
- L'intégration en continue se lance alors.
- Dans un premier temps, les fichiers sont envoyés à un "runner" (machine qui se charge de la compilation).
- Ensuite cette machine crée l'image du projet (permet de lancer plus tard un conteneur).
- Puis celui-ci publie cette image sur Gitlab (qui reste privée).
- Ensuite, le "runner" envoie un message à notre serveur.
- Ce-dernier télécharge alors l'image, et relance le conteneur du projet.
- Le projet devient alors accessible sur `amplet.fr`.

3.2 Partie Web

3.2.1 Routes

3.2.1.1 Page d'accueil

Lorsque l'on arrive sur la page d'accueil nous avons une courte présentation du site faite avec bootstrap qui explique rapidement le principe de l'application.

3.2.1.2 Se connecter

Lorsque l'utilisateur veut se connecter il rentre son identifiant ainsi que son mot de passe afin d'être connecté sur l'ensemble du site web. On utilise alors la fonction de sécurité implémentée dans la partie "Algorithmie".

3.2.1.3 S'inscrire

Lorsque l'utilisateur ne possède pas de compte, il peut en créer un en renseignant un nom d'utilisateur, une adresse mail, un mot de passe et son adresse. En effet il est indispensable d'avoir l'adresse de chaque utilisateur sinon ils ne peuvent pas s'incrire à une Amplet.

3.2.1.4 Se déconnecter

L'utilisateur peut se déconnecter à tout moment, à condition d'être connecté.

3.2.1.5 Profil

L'utilisateur peut accéder à son profil mais également à celui des autres. Son profil est élargi, il peut voir toutes ses informations personnelles. Cette route est également utilisable pour trouver un utilisateur et avoir des informations sur cette personne, son adresse est alors masquée et remplacée par son code postal uniquement.

3.2.1.6 Modification de profil

L'utilisateur peut modifier à sa guise ses informations dans la page prévue à cet effet. Il peut changer son mail, son adresse e-mail mais également son adresse. Bien sûr chaque demande de modification est précédée par une vérification de la base de données afin de ne pas pouvoir avoir le même mail ou le même nom d'utilisateur qu'une autre personne sur le site. Un utilisateur peut aussi choisir de changer son avatar, l'ancien sera alors supprimé et remplacé par le nouveau.

3.2.1.7 Amplets en cours

L'utilisateur peut rechercher les différentes Amplets en choisissant parmi plusieurs filtres (dates, type) ainsi que les classer pour ensuite pouvoir s'inscrire à celles-ci (il sera alors placé en attente).

3.2.1.8 Mes Amplets

L'utilisateur peut en accédant à cette page voir les Amplets auxquelles il est inscrit, ses votes pour la navette et leur statut, mais il peut également accepter ou refuser les autres utilisateurs s'étant inscrits sur les Amplets qu'il a mises en ligne.

3.2.1.9 Création d'Amplets

L'utilisateur peut sur cette page créer une Amplet, d'autres utilisateurs pourront s'y inscrire et choisir leurs produits parmi la liste proposée par l'utilisateur. Il pourra aussi définir le nombre de places, ainsi que la date.

3.2.1.10 Navette

L'utilisateur peut émettre un vote, un souhait de produits depuis la page navette. Ce choix a été arbitrairement fixé à 5 produits maximum (il pourrait ultérieurement être modifié à la guise de l'utilisateur gérant la navette). Suite à ce vote il pourra suivre l'évolution de l'élection des marchands qui est fait par notre algorithme sur la page Mes Amplets. Les votes sont émis par produits et pas par commande, ainsi chaque produit a une chance d'être sélectionné.

Afin de s'assurer qu'il ne vote qu'une fois par navette, lorsqu'un utilisateur vote, sont enregistrés dans la base de données la navette sur laquelle il a voté ainsi que les produits demandés. Nous gardons donc une trace de tous les votes effectués sur une navette ce qui rend impossible le fait de pouvoir voter plusieurs fois.

3.2.1.11 Tableau de bord

Cette page affiche le contenu de la base de donnés en le formatant pour qu'il soit plus lisible lors de nos sessions de debug. Elle est accessible à tout le monde pour la facilité de développement, mais dans un cas réel seul l'admin y aurait accès.

3.2.1.12 Création de Marchands

Accessible uniquement par certaines personnes (administrateurs et "maires"), elle permet d'ajouter un marchand à la base de données.

3.2.1.13 Ajout de Produits

Accessible uniquement par certaines personnes (administrateurs et "maires") elle permet d'ajouter un produit à un marchand en lui donnant un prix.

3.2.1.14 Création de Navettes

Accessible uniquement par un utilisateur faisant partie du groupe "maire". Il peut décider de créer une Navette en choisissant toutes ses caractéristiques.

3.2.1.15 Gestion de Navettes

Accessible uniquement par un utilisateur faisant partie du groupe "maire". Il peut lancer (ou non) les votes pour certaines Navettes (il peut donc les fermer, les rendre inaccessibles et donc figer les votes)

3.2.1.16 Trajet

Page non finie par manque de temps mais qui présente tout de même les informations nécessaires.

Il s'agit d'une route qui utilise la fonction des plus courts chemins pour calculer le trajet de navette optimal sous forme d'une boucle. Elle sera consulté par le conducteur de la navette et proposera donc l'ordre des marchands par lesquels passer.

3.2.1.17 Gestion des erreurs

Lorsqu'une erreur 404 est détectée, une page personnalisée apparaît afin de ne pas rompre avec la direction artistique du site.

3.2.2 SocketIO

3.2.2.1 Fonctionnement

Afin de permettre aux clients de pouvoir communiquer entre eux (pour se mettre d'accord, etc), nous avons décidé d'implémenter un système de chat qui fonctionne comme fonctionneraient des SMS.

Pour réaliser cela, nous avons utilisé SocketIO (Flask-SocketIO). C'est une librairie qui permet de créer facilement un WebSocket (qui permet une communcation bi-directionnelle entre client et serveur). On pourra ainsi définir des routes à la manière de Flask, qui correspondront à des noms d'événements (par exemple : "message_recu")

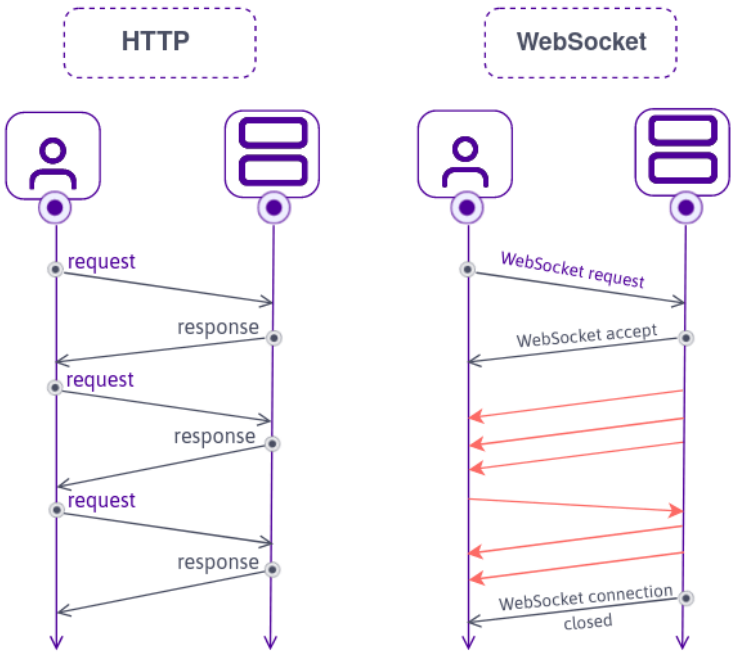

FIGURE 3.1 – Schéma explicitant la différence entre WebSocket et HTTP (cf. ref. [11])

3.2.2.2 Chat en temps réel

Lorsqu'un utilisateur est connecté grâce à Flask-Login, celui-ci a accès aux routes du WebSocket. Il peut alors s'authentifier et être assigné à une "Room" (qui est adressable). Ainsi, lorsqu'un utilisateur veut envoyer un message à une personne, on peut lui envoyer certaines informations sur le contact (photo de profil et identifiant notamment), puis grâce à du JavaScript, mettre à jour sa page pour qu'il puisse le voir dans son chat. Il pourra ensuite parler à son contact : en envoyant un message au websocket, le serveur s'occupe d'enregistrer le message dans la base de données et, si le contact est en ligne, envoyer le message dans sa "Room" pour que lui aussi puisse l'afficher. Cela permet une communication en temps réel entre les clients, mais aussi en différé (si le contact n'est pas en ligne, lorsqu'il se connectera, il pourra voir le message dans le chat).

3.2.2.3 Recherche d'utilisateurs

Afin de simplifier les démarches d'un utilisateur, celui-ci peut rechercher d'autres utilisateurs grâce à la barre de recherche. Celle-ci utilise SocketIO puisque lorsque l'utilisateur rentre le nom d'une personne, on voudrait mettre à jour la page dynamiquement. Ainsi, à chaque entrée, celui-ci envoie la requête à SocketIO, qui va répondre une liste d'utilisateurs trouvés dans la base de données.

3.3 Partie Base de Données

3.3.1 SQLAlchemy

Dès le début de notre projet nous avons décidé d'utiliser une librairie python permettant un accès efficace aux bases de données et une parfaite adaptation avec l'environnement python. Ci-joint voici l'exemple de la déclaration d'une table dans un fichier python utilisant SQLAlchemy

```
class User(db.Model, UserMixin):
    __tablename__ = 'users'
    id = db.Column(db.String(40), primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password_hash = db.Column(db.String(128), nullable=False)
```

Listing 3.1 – Fichier python users.py (raccourci)

Comme on peut le remarquer, cela nous permet d'utiliser les classes python pour faire une table et de lui associer des méthodes, ici on utilisera la méthode pour générer le hash SHA-1 du mot de passe. Cela permet une gestion plus claire des profils utilisateurs et un code plus propre ce qui est très appréciable pour un gros projet.

Voici l'exemple d'une requête SQL réalisée à l'aide de SQLAlchemy.

```
utilisateur = User.query.get(id='351464456645645')
utilisateur.username -> Contient le nom d'utilisateur
```

Notre projet nécessite de nombreuses tables afin d'être chacunes en troisième forme normale.

La liste de nos tables, des attributs ainsi que le schéma entités-associations sont dans l'Annexe 1.

3.3.2 Particularités de certains enregistrements

3.3.2.1 Dates

Afin d'enregistrer les dates, nous avons choisi d'enregistrer des horodatages en millisecondes écoulées depuis l'epoch UNIX (1er Janvier 1970). Ceci permet à la fois d'enregistrer un entier, mais permet aussi (si besoin) de pouvoir échanger ces horodatages facilement.

3.3.2.2 ID

Concernant les clé primaires des tables qui en ont besoin, nous utilisons des "Snowflakes" pour générer des clés uniques. Un "Snowflake" est un mot binaire de 64 bits dont :

- 41 bits correspondant à l'horodatage ;
- 10 bits correspondant à l'identifiant du générateur (différent pour chaque table) ;
- 12 bits correspondant à un numéro de séquence (qui s'incrémente de 1 à chaque passage dans le générateur) ;

Ce système permet d'être sûr que deux IDs générés sur la même machine à la même milliseconde seront bien différents (et uniques), il est plus largement utilisé dans de gros systèmes distribués tel que Discord ou Twitter (cf. ref. [7]).

3.4 Partie Algorithmie

3.4.1 SHA-1

Nous avons décidé d'implémenter un système de login. Afin de sécuriser celui-ci nous avions deux choix :

- Utiliser une librarie comme Werkzeug Security ;
- Créer notre propre implémentation.

Nous avons donc opté pour créer notre propre fonction afin d'en apprendre un petit peu sur la cryptographie. Cependant, afin de pouvoir respecter les contraintes de temps, nous avons choisi l'algorithme SHA-1.

Avantages	Inconvénients
Pas trop compliqué à mettre en place. Fonction connue et facilement testable.	Obsolète.

FIGURE 3.2 – Tableau des avantages et inconvénients du SHA-1

3.4.1.1 Fonctionnement de SHA-1

Afin de correspondre au standard SHA-1, nous avons fait appel à la publication officielle de la FIPS (cf. ref. [10]) concernant celui-ci.

3.4.1.1.1 Opérateurs sur les bits utilisés

- $\wedge$: ET logique ;
- $\vee$: OU "inclusif" logique ;
- $\oplus$: OU "exclusif" logique.
- $\neg$: NON logique ;
- $\ll$: Décalage de bit à gauche ;
- $\gg$: Décalage de bit à droite ;
- $ROTG^n(x)$: Rotation de n bit vers la gauche du mot x de taille k, définie par :

$$ROTG^n(x) = (x \ll n) \vee (x \gg k - n)$$

3.4.1.1.2 Pré-traitement

- Rembourrage : On veut que notre entrée soit d'une taille multiple de 512 (en bits). Afin de réaliser ceci, on rajoute un "1" à la fin du message, puis on ajoute n "0". Enfin, on ajoute la taille du message original encodé sur 64 bits. On choisit n de telle sorte à ce que le message final soit d'une taille multiple de 512.
- Analyse : En découpant le message en blocs de 512, on aura donc plusieurs blocs (M blocs), qui seront eux-mêmes découpés plus loin en morceaux de 64 bits (16 morceaux).
- On met en place les valeurs initiales données par le standard appelées $H_i^{(n)}$ (n étant alors le numéro du bloc en cours).

3.4.1.1.3 Traitement d'un bloc n

- On calcule 80 valeurs :
  - les 16 premières sont les 16 morceaux du bloc en cours ;
  - les 64 suivantes sont calculées par récurrence :

$$v[i] = ROTG^1(v[i-3] \oplus v[i-8] \oplus v[i-14] \oplus v[i-16])$$

- On définit $H_i^{(n)}$ (pour $1 \leq i \leq 5$) (soit à partir des valeurs $H_i^{(n-1)}$, soit à partir des valeurs initiales définies par le standard)
- On fait 80 itérations qui vont modifier ces valeurs (grâce à des fonctions définies par le standard).
- On obtient alors $H_i^{(n+1)}$, puis on continue sur le bloc suivant.

3.4.2 Sécurisation et implémentation de SHA-1

Afin de ne pas seulement stocker le premier hash du mot de passe dans la base de données (ce qui est peu sécurisé puisqu'il y a beaucoup de chances que ce hash soit déjà connu), nous allons appliquer une méthode différente.

3.4.2.1 Termes utilisés

- Sel : Mot aléatoire qui change pour chaque mot de passe.
- Poivre : Mot aléatoire commun à chaque mot de passe.
- Iterations : Nombre aléatoire très grand.

3.4.2.2 Procédé

Ce procédé est très connu et est même très souvent utilisé dans des cas d'applications concrets (ormis le fait que SHA-256 ou SHA-512 seront utilisés dans ce genre de cas).

Si nous notons X l'entrée utilisateur (mot de passe en clair), nous aurons :

$$X'_0 = sel + X + poivre$$

Ensuite on fait une récurrence jusqu'à $n = iter$:

$$X'_n = SHA(X'_{n-1})$$

Il suffit d'enregistrer ceci dans la base de données pour avoir toute les informations nécessaires (avec un séparateur pour pouvoir les retrouver plus tard).

```
1 data = f'{poivre}${sel}${iterations}${hash}'
```

Listing 3.2 – String à enregistrer

3.4.2.3 Vérification

Afin de vérifier un mot de passe :

— On décode depuis la base de données le sel, poivre, itérations et le hash ;
— On applique le procédé avec ces valeurs ;
— On vérifie que le hash trouvé est égal au hash importé.

3.4.3 Tri Fusion

À plusieurs reprises nous avons eu besoin de trier des éléments, qui n'étaient pas forcément des listes d'entiers (par exemple, des listes de dictionnaires ou des listes de tuples). Il n'y a alors pas vraiment de fonctions prévues par python pour gérer ce genre de cas, et nous avons dû réimplémenter des algorithmes de tri adéquats. Pour cela, nous avons décidés de nous baser sur l'algorithme de tri fusion (merge sort) et de l'adapter aux données d'entrée. Ainsi, le principe reste le même, et seule la manière de comparer les éléments de la liste change :

— Si la liste est vide ou ne contient qu'un seul élément, on la retourne ;
— Sinon, on la divise en deux sous-listes que l'on trie de la même manière ;
— Une fois les deux listes triées, on compare un à un les éléments en tête des listes et on ajoute le plus petit des deux dans la liste finale.

Pour la comparaison, tout dépend des objets que l'on trie. Sur le schéma ci-dessous, on se contente de comparer l'attribut 'nombre' des objets pour les trier.

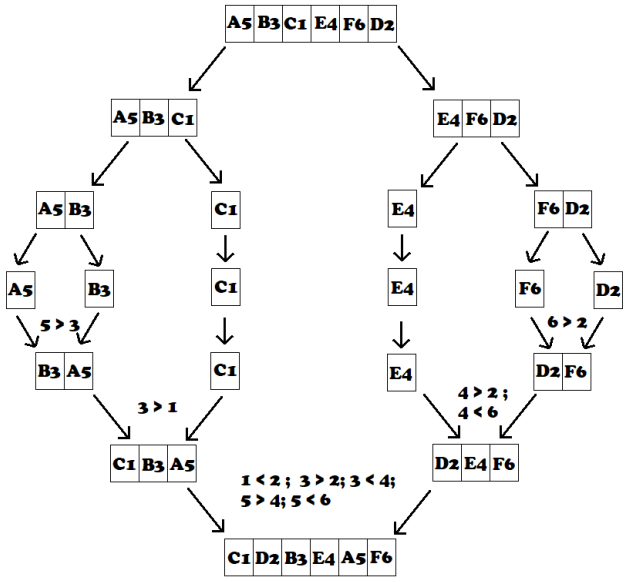

FIGURE 3.3 – Schéma explicitant le tri fusion sur des objets à 2 attributs

3.4.3.1 complexité

Le tri fusion a une complexité de $\Theta(nlog(n))$ en moyenne, peu importe la configuration de départ. Démontrons le.

On commence par supposer que la liste à trier est de taille $N = 2^p$ où $p \in \mathbb{N}$.
Si $p = 1$, alors la liste est de taille 1 et on retourne directement la liste. Sinon, on divise la liste en deux et on trie chacune de ces listes de taille $N/2$, avant de refusionner les N éléments. Ce qui donne :

$$C(N) = \begin{cases} 1 & \text{si } N = 1 \\ 2 * C(\frac{N}{2}) + N & \text{sinon.} \end{cases}$$

En sachant que $N = 2^p$, et en divisant par 2^p on obtient :

$$\frac{C(2^p)}{2^p} = \begin{cases} 1 & \text{si } p = 0 \\ \frac{C(2^{p-1})}{2^{p-1}} + 1 & \text{sinon.} \end{cases}$$

Si, ensuite, on pose $up = \frac{C(2^p)}{2^p}$, on a :

$$u_p = \begin{cases} 1 & \text{si } p = 0 \\ u_{p-1} + 1 & \text{sinon.} \end{cases}$$

Alors dans ce cas, u_p est une suite arithmétique telle que $u_p = p + 1$, d'où :

$$C(N) = 2^p * u_p = 2^p * p + 2^p$$

Or, $p = log_2(N)$, donc finalement :

$$C(N) = N * log_2(N) + N = \Theta(Nlog(N))$$

Donc dans le cas où $N = 2^p$, la complexité est en $\Theta(Nlog(N))$.

Dans le cas général, $\exists p \in \mathbb{N}$ tel que :

$$2^p \leq N \leq 2^{p+1}$$

Comme la complexité est une fonction croissante de la taille, on a :

$$C(2^p) \leq C(N) \leq C(2^{p+1})$$

Soit :

$$p * 2^p \leq C(N) \leq (p + 1) * 2^{p+1}$$

Or, comme $2^p \leq N \leq 2^{p+1}$, $\exists k_1 \in [1, +\infty[$ et $\exists k_2 \in]0, 1]$ tel que :

$$p2^p = k_1 * Nlog_2(N) \text{ et } (p + 1)2^{p+1} = k_2 * Nlog_2(N)$$

D'où :

$$k_1 * Nlog_2(N) \leq C(N) \leq k_2 * Nlog_2(N)$$

Et par théorème d'encadrement, on a :

$$C(N) = \Theta(Nlog_2(N)) \text{ (cf. ref. [1])}$$

3.4.3.2 Justification du choix et limites

La question suivante se pose alors : pourquoi ce type de tri et pas un autre ? La première raison est sa complexité moyenne optimale : aucun tri basé sur des comparaisons ne peut avoir une complexité meilleure que du $\Theta(Nlog(N))$. Bien qu'en moyenne deux fois plus lent que le tri rapide, sa complexité dans le pire des cas reste en $\Theta(Nlog(N))$, alors que celle du tri rapide, elle, est en $\Theta(N^2)$. Enfin, c'est un type de tri que nous connaissions au préalable, et qu'il était donc plus facile pour nous de modifier à notre guise.

Cependant, il pose un inconvénient majeur : c'est un tri récursif. Or le risque pour ce type de tri est d'atteindre la profondeur maximale de récursion (qui est de 1000 dans notre cas). Y a-t-il alors une chance qu'elle soit atteinte dans notre projet ? Commençons par étudier le nombre de récursions effectuées pour une liste de taille n :

On divise le tableau $log_2(n)$ fois au total, et à chaque étape on a 2 récursions qui se font, ce qui fait au total $2^{log_2(n)} = n$ récursions.

Ainsi, il s'agit de se demander si les listes triées ont une chance d'atteindre une taille avoisinant 1000 :

— Dans le premier cas (tri_fusion dans utils.vote_marchand.py), on se contente de trier une liste contenant le nombre de magasins ayant reçus des votes pour une navette donnée. Au vu de la taille et de l'ampleur actuelle du projet, il n'y a aucun risque d'atteindre 1000 éléments triés (Même si on élargissait le projet à tout Nancy, on atteindrait au maximum environ 800 commerçants, en supposant qu'ils soient tous partenaires et tous sélectionnés par la navette).

— Dans le second cas (recherche_par, dans utils.recherche_par), le tri est utilisé pour trier la liste des amplets particuliers à afficher à l'utilisateur soit par date de début, soit par proximité. De la même manière, à l'heure actuelle le site est loin d'avoir 1000 Amplets disponibles (et n'est pas encore conçu pour en afficher autant). Cependant, il n'est pas exclu que si le projet venait à être étendu sur une grande agglomération, on puisse ici dépasser la limite de récursions.

Ainsi, même si à l'heure actuelle, atteindre cette limite n'est pas quelque chose de préoccupant, il faudrait sûrement repenser l'algorithme de tri (tout comme beaucoup d'autres parties du site), soit en le remplaçant par un autre, non récursif, soit en le dérecursivant en utilisant la méthode des piles d'appels.

3.4.3.3 Cas du tri "recherche_par.py"

Cette section est simplement dédiée à l'éclaircissement du fonctionnement du tri fusion 'recherche_par'. Comme dit plus haut, il est utilisé pour trier selon un critère au choix une liste d'Amplets. Enfait, on aurait presque pu faire deux sous-fonctions, chacune effectuant un tri fusion adapté à un critère de recherche précis. La fonction 'recherche_par' a donc été écrite telle que ces deux sous-fonctions se retrouvent sous un seul et même algorithme, pour éviter d'avoir à définir trop de fonctions annexes.

3.4.4 Problème du Voyageur de Commerce et algorithme de résolution

Afin d'avoir une attitude responsable et écologique notre navette se doit de choisir le trajet qui est le plus court. Nous avons décidé d'implémenter une fonction qui, pour une liste de points donnés, calculerait la boucle la plus courte en distance passant par tous les points une seule fois. Bien que cela soit une approximation plutôt grossière car elle se fait uniquement sur des déplacement à vol d'oiseau. Nous nous refusions l'utilisation d'un logiciel préfait pour résoudre ce problème (type API Google Map). Ce problème que nous voulons résoudre est une instance du Problème du Voyageur de commerce encore appelé circuit hamiltonien.

Voici un exemple sur un graphe composé de six sommets on peut facilement voir qu'il y a plusieurs solutions possibles : certaines sont bonnes et d'autres sont "moins optimisées"

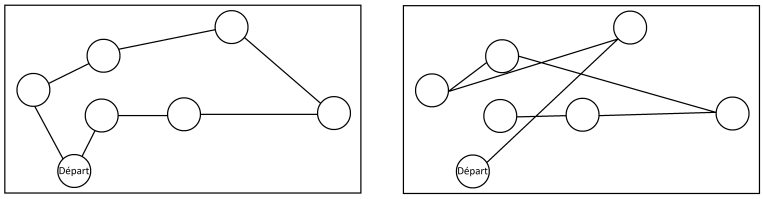

FIGURE 3.4 – À gauche un trajet optimisé raisonablement, à droite un trajet aléatoire non optimisé

3.4.4.1 Première Approche

Le problème du voyageur de commerce est connu pour être un problème NP-Complet. Cela signifie qu'il n'est pas possible de trouver la solution avec un coût polynomial.

On va donc dans un premier temps essayer de trouver une solution satisfaisante qui n'est pas optimale. Mais qui nous permet d'avoir une approximation raisonable du plus court chemin.

3.4.4.1.1 Création d'une matrice d'arêtes

Pour faciliter le développement d'algorithmes de résolution. Il faut d'abord créer une matrice d'arêtes avec pour chaque sommet la distance à tous les autres. En effet on va considérer pour notre instance du problème que chaque marchand possède des arêtes vers tous les autres marchands.

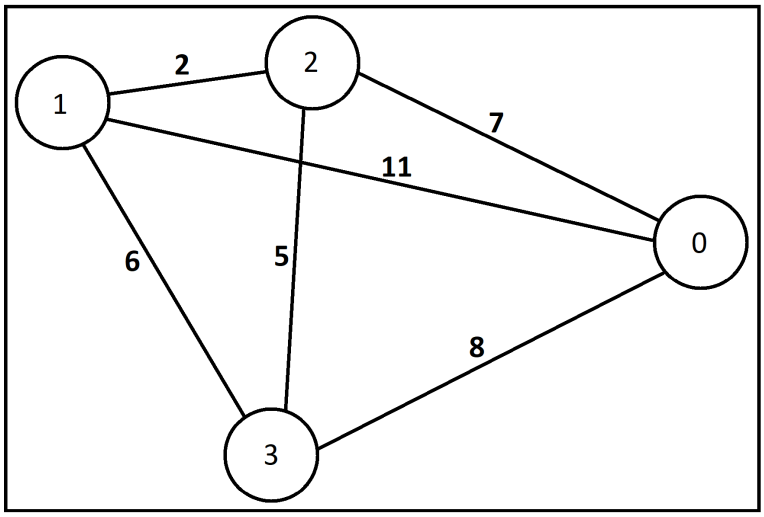

FIGURE 3.5 – Graphe complet non orienté à quatres sommets

Par exemple, pour ce graphe, voici sa matrice d'arêtes.

$$A = \begin{bmatrix} 0 & 11 & 7 & 8 \\ 11 & 0 & 2 & 6 \\ 7 & 2 & 0 & 3 \\ 8 & 6 & 5 & 0 \end{bmatrix}$$

On peut constater que cette matrice est symétrique. Notre premier algorithme parcourt le graphe en partant du départ et va à chaque fois choisir le sommet le plus proche qui n'a pas déjà été visité. Cet algorithme est grandement facilité par notre matrice d'arêtes.

Au niveau de l'algorithme la fonction de création de matrice prend en entrée une liste de points qui possèdent chacun des coordonnées x et y et grâce à la formule du calcul de la distance dans un repère orthonormal on obtient une distance que l'on va choisir comme arrête entre ces deux points.

$$d = \sqrt{(xb - xa)^2 - (yb - ya)^2}$$

On remplit ainsi notre matrice au fur et à mesure, il possède une complexité en $\Theta(n^2)$.

3.4.4.1.2 Premier Algorithme

Cet algorithme prend en entrée une liste de sommets désignés par leurs coordonnées x et y. On crée sa matrice d'arêtes et on prend le premier point comme point de départ. Puis en se référant à la matrice et à une liste de sommets déjà visités on construit une liste qui part du départ, passe par tous les points, puis revient au départ.

Sur une librairie connue d'instances du problèmes du voyageur de commerce (TSPLIB), en moyenne cet algorithme est 1.26 fois plus grand que le trajet optimal. Il semble donc que les données proposées par cet algorithme soit acceptables dans le cas où il y a beaucoup de sommets.

De plus cet algorithme possède une complexité en $\Theta(n^2)$ car on parcourt la matrice d'arêtes de taille n^2, pour choisir à chaque fois le sommet le plus proche d'un point tout en vérifiant qu'il n'est pas dans la liste des points déjà visités.

3.4.4.2 Approche Exhaustive

Une approche exhaustive qui consisterait à lister toutes les permutations possibles et tester leur longueur totale permettrait d'obtenir la solution optimale, mais à quel prix ?

Dans le cas général du problème du voyageur de commerce il y a $\frac{1}{2} * (n-1)!$ chemins possibles avec n étant le nombre de sommets à visiter. En effet on peut dénombrer cela en considérant qu'il s'agit d'une liste d'éléments tous différents dont le premier élément est connu, il y a donc $(n-1)!$ possibilités. De plus comme on forme une boucle, la moitié des chemins sont inutiles, ce qui nous donne bien $\frac{1}{2} * (n-1)!$ chemins possibles. On génère alors l'ensemble des permutations possibles et avec un algorithme de complexité $\Theta(n!)$.

Cependant notre instance du problème ne peut avoir plus de six éléments. En effet la fonction de vote a en sortie une liste de 1 à 5 éléments qui sont les marchands à visiter. Donc en ajoutant le point correspondant au départ on a 6 sommets maximums ce qui nous donne 120 permutations possibles soit 60 chemins possibles. Ce qui est tout à fait acceptable compte tenu de l'exécution machine d'une permutation (voir chapitre performance)

3.4.4.2.1 Algorithme de Heap

Pour réaliser notre algorithme du plus court trajet on utilise l'algorithme de Heap. Le principe est que chaque permutation est générée à partir de la précédente. Lors de la k-ième étape il génère les $k!$ permutations des k premiers éléments en faisant des appels récursifs sur l'ensemble qu'ils constituent. Tout d'abord un appel récursif est exécuté sans faire aucune modification pour calculer les permutations ayant le k-ième élément fixé. Ensuite, on échange deux éléments et on exécute un appel récursif supplémentaire $k-1$ fois.

Ci-joint un arbre de récursions pour les permutations des lettres ABC Ici les lettres sont fixées de gauche à droite.

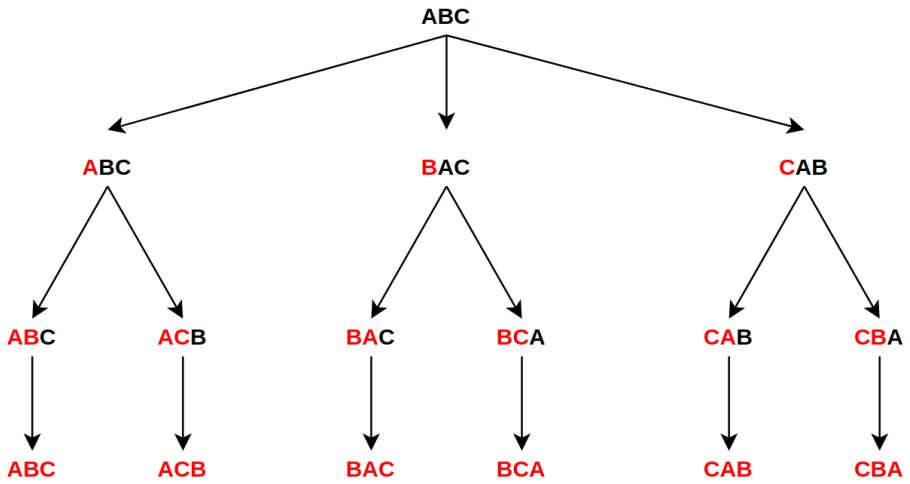

FIGURE 3.6 – Arbre de récursions de l'algorithme de Heap (cf. ref. [5])

On peut montrer que le nombre d'échanges que fait l'algorithme est exactement de $n!$ et que c'est le minimum d'échanges possible. La complexité est en $\Theta(n!)$

3.4.4.3 Optimisation possible

Le manque d'optimisation de nos algorithmes vient du fait que l'on essaye de résoudre le problème sans informations préalables.

Bien évidemment il est possible d'optimiser l'algorithme, pour le problème de voyageur de commerce. Par exemple en réalisant plusieurs itérations du trajet et en décroisant les arêtes qui se croisent sur le graphe ou encore en essayant de trouver l'arc de bouclage le plus court.

3.4.5 Mise à jour des multpilicateurs marchands

3.4.5.1 Contexte et explication de l'algorithme

Enfin, nous avions besoin d'un moyen d'assurer que chaque commerçant avait une chance d'être représenté par une navette. En effet, comme les commerçants par lesquels passent la navette sont décidés par les votes des citoyens, et qu'il y a un nombre de places limité, certains magasins peuvent ne pas être sélectionnés. Afin de leur donner une chance supplémentaire d'être choisis au vote suivant, nous avons décidé d'attribuer à chaque

commerçant un multiplicateur dynamique, qui pondère les votes attribués aux commerçants lors des choix de la navette. Ainsi, si un magasin est peu sélectionné, son multiplicateur augmentera pour lui donner plus de chances d'être visité par la prochaine navette.

Le fonctionnement de l'algorithme est assez simple, puisque lorsqu'une navette est votée :

— On remet le multiplicateur des magasins choisis à 1 (valeur par défaut) ;

— Pour l'ensemble des magasins ayant reçu au moins un vote et n'ayant pas été sélectionnés, on multiplie leur multiplicateur actuel par leur poids dans les votes totaux des magasins non choisis (cf schéma ci-dessous).

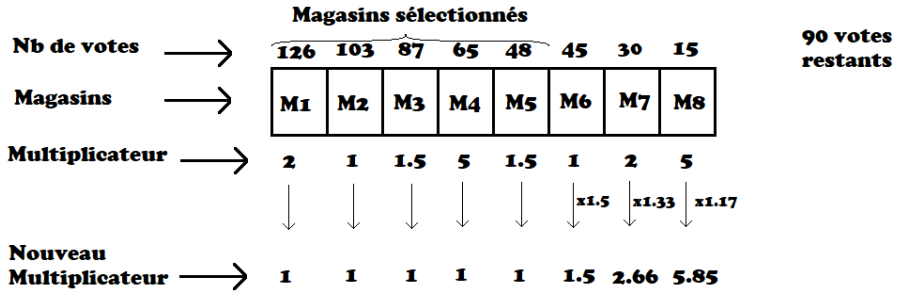

FIGURE 3.7 – Schéma explicitant la mise à jour des multiplicateurs suite à un vote

3.4.5.2 Avantages et inconvénients d'un tel système

L'avantage d'un tel système, c'est que ce ne sont pas toujours les mêmes magasins qui apparaissent dans la navette, et permettent ainsi de satisfaire les besoins de tous les utilisateurs. De plus, ce système empêche qu'un commerçant détienne le monopole d'un marché sur le site : en effet, la navette ne sélectionne qu'un magasin par type par voyage (exemple : un seul primeur). Ainsi si un primeur propose bien plus de produits que les autres, il ne sera quand même pas choisi à tous les coups, permettant de faire vivre l'économie de tous les commerces locaux.

Cependant, on peut critiquer l'algorithme qui ne tient pas compte de l'historique des valeurs des multiplicateurs, mais simplement de leur valeur instantanée. Ainsi la mise à jour de ces valeurs est moins précise et surtout plus susceptible aux fortes variations des votes. Par exemple, un magasin qui reste trop peu choisi par rapport à sa demande réelle se verra immédiatement remis à 1 lorsqu'il apparaîtra enfin, alors qu'en analysant l'évolution des votes pour ce commerce, on aurait pu déterminer une valeur par défaut "plus optimale".

Ceci dit, par manque de réelles compétences dans l'analyse de données, et surtout de temps (il aurait fallu réimplémenter une nouvelle base de données pour l'historique des valeurs, et surtout revoir entièrement l'algorithme en lui-même) nous avons opté pour ce système plus simple.

Toutefois, on peut remarquer que l'algorithme a une complexité linéaire en le nombre de magasins enregistrés dans la base de données (2 boucles for non imbriquées).

Chapitre 4

Tests et performances

4.1 Introduction

Afin de tester nos algorithmes, nous avons mis en places des tests unitaires grâce au module Pytest. Cependant, lors de la réalisation de tests qui nécessites la base de données, nous ne pouvions pas utiliser Pytest (l'import de la base de données faisait planter les tests). Cependant, pour pallier à ce problème nous avons créé une base de données "test" qui comporte déjà des informations que nous connaissions, ainsi que le page tableau de bord. Nous avons ainsi donc pu tester l'ensemble du site en se comportant comme un utilisateur lambda qui essaierais de "casser" le site.

4.2 Tests

4.2.1 SHA-1 et hashage

Afin de tester au mieux notre implémentation, nous utilisons l'implémentation de la librairie hashlib. Ainsi, nous pouvons comparer les résultats de nos hashs (en sachant que le hashage est une fonction déterministe). Pour vérifier l'implémentation de sécurité, nous créeons la valeur de plusieurs mot de passe, puis nous comparons avec la fonction de comparaison.

```
def test_hash():
    assert hash(b'Celine') == sha1(b'Celine').hexdigest()
    assert hash(b'SLQJKHDKJSHdklqD') == sha1(b'SLQJKHDKJSHdklqD').hexdigest()
    assert hash(b'QslkJQLS') == sha1(b'QslkJQLS').hexdigest()
    assert hash(b'mOtDePaSSeEtReSEcURISE') == sha1(b'mOtDePaSSeEtReSEcURISE').hexdigest()
    assert hash(b'aaa[...]aaaaaa') == sha1(b'aaa[...]aaaaaa').hexdigest()

def test_verif():
    hash = generate_password_hash('pASwOrd')
    assert check_password_hash(hash, 'pASwOrd')
    hash = generate_password_hash('SLQJKHDKJSHdklqD')
    assert check_password_hash(hash, 'SLQJKHDKJSHdklqD')
    hash = generate_password_hash('QslkJQLS')
    assert check_password_hash(hash, 'QslkJQLS')
    hash = generate_password_hash('mOtDePaSSeEtReSEcURISE')
    assert check_password_hash(hash, 'mOtDePaSSeEtReSEcURISE')
```

Listing 4.1 – Tests réalisé sur le SHA-1

4.2.2 Timestamps et formattage

Pour garder une certaine cohérence dans les textes renvoyés à l'utilisateurs, nous avons décider d'un formattage unique des dates : JJ/MM/AAAA, ainsi il faut vérifier que le formattage est valide, mais aussi qu'un timestamp soit bien lié à la bonne date.

```
def test_timestamp():
    # Test de la validite du timestamp
    date = timestamp_to_date(1640794133000)
    assert date.hour == 16
    assert date.minute == 8
    assert date.second == 53
    # Test du formattage
    date = timestamp_to_date(1640794133000, format=True)
    assert date == "29/12/2021 - 16h08"
```

```
    date = timestamp_to_date(1340794133000, format=True)
    assert date == "27/06/2012 - 10h48"
```

Listing 4.2 – Tests réalisés sur le SHA-1

4.2.3 Plus court chemin

```
def test_pluscourtchemin():
    assert pluscourtchemin([(5,6),(3,2),(8,7),(6,9),(0,0)]) == [(6, 9), (8, 7), (3, 2), (0, 0)
    , (5, 6)]
    assert pluscourtchemin([(20,13),(21,4),(45,9),(20,28),(8,7)]) == [(21, 4), (8, 7), (20,
    28), (45, 9), (20, 13)]
    assert pluscourtcheminexhaustif([(20,13,"maison"),(21,4,"point 1"),(45,9,"point 2")
    ,(20,28,"point 3"),(8,7,"point 4")]) == ([(8, 7, 'point 4'), (21, 4, 'point 1'), (45, 9, '
    point 2'), (20, 28, 'point 3'), (20, 13, 'maison')], 97.67401020960276)
```

Listing 4.3 – Tests réalisés sur "Plus court Chemin"

Afin de réaliser un trajet assez fiable, il faut vérifier que notre algorithme de plus court trajet nous renvoit des trajets acceptables qui sont correctement optimisés. Pour ce qui concerne les tests, on peut réaliser des test à la mains en calculant le plus court trajet effectif et vérifier que notre algorithme nous renvoie la solution attendue en fonction de son fonctionnement.

4.3 Performances

On utilisera les librairies line_profiler et memory_profiler afin d'analyser les performances temporelles et spatiales.

4.3.1 SHA-1 et hashage

Ligne	Occurences	Temps	Temps par occurences	Ligne
1	10001	6719.0 μs	0.7 μs	for i in range(10000) :
2	10000	80172.0 μs	8.0 μs	size = random.randint(6,20)
3	10000	402370.0 μs	40.2 μs	rdm = gen_salt(size).encode()
4	10000	10758593.0 μs	1075.9 μs	hash(rdm)
5	10000	48199.0 μs	4.8 μs	sha1(rdm).hexdigest()
6	101	240.0 μs	2.2 μs	for i in range(100) :
7	100	1450.0 μs	14.5 μs	size = random.randint(6,20)
8	100	4900.0 μs	49.0 μs	rdm = gen_salt(size)
9	100	676011890.0 μs	6760118.9 μs	generate_password_hash(rdm)

FIGURE 4.1 – Tableau récapitulatif des temps pour chaque lignes de code

Grâce à ce tableau, nous pouvons en conclure que notre implémentation de SHA-1 n'est pas très performante (250 fois plus lente). Cela s'explique par le fait que nous la comparons à une implémentation faites en C (grâce à CPython), or ce language compilé est beaucoup plus performant que Python. Nous pouvons aussi voir que la génération de hash sécurisé est très lente (donc il en est de même pour la vérification), d'une part à cause de Python, d'autres part puisque nous voulons que la générations de hash sécurisés soit lente pour éviter qu'un agent malveillant trouve très vite le mot de passe à partir d'un hash.

4.3.2 Plus court chemin

On peut voir ici le résultat des deux implémentations de l'algorithme des plus court chemin sachant que notre navette doit effectuer une boucle ne passant par tous les points. On retrouve une instance du problème du voyageur de commerce. L'algorithme de recherche du plus court chemin simple est plutôt efficace car la compléxité est quadratique mais le résultat est une approximation de la solution optimale.

On peut voir que les test de performance pour l'algorithme de recherche de plus court exaustif est peu performant. En effet la complexité est en O(n!) puisque que l'on calcule toutes les substitutions possibles. Cet

algorithme pourrait être simplifié car on réalise une boucle donc un chemin dans un sens ou dans l'autre est le même. Cependant cela ne changerait pas la complexité.

Ci joint le tableau de performance du programme pour des tailles de données cohérentes avec notre projet.

Ligne	Occurences	Temps	Temps par occurences	Ligne
1	1001	531.0 μs	0.5 μs	for i in range(1000) :
2	1000	563.0 μs	0.6 μs	L = []
3	6009	6706.0 μs	1.1 μs	for j in range(randint(4, 6)) :
4	5009	36134.0 μs	7.2 μs	L.append((randint(0,1000),randint(0,1000)))
5	1000	71673.0 μs	71.7 μs	pluscourtchemin(L)
7	1001	1213.0 μs	1.2 μs	for i in range(1000) :
8	1000	672.0 μs	0.7 μs	L = []
9	6033	10258.0 μs	1.7 μs	for j in range(randint(4, 6)) :
10	5033	40203.0 μs	8.0 μs	L.append((randint(0,1000),randint(0,1000)))
11	1000	805357.0 μs	805.4 μs	pluscourtcheminexhaustif(L)

FIGURE 4.2 – Tableau récapitulatif des temps pour chaque lignes de code

Chapitre 5

Gestion de Projet

5.1 Équipe de projet

L'équipe se compose de 4 étudiants en première année :
— Tristan SMAGGHE
— Jules BRUNET
— Louis-Vincent CAPELLI
— Malo DAMIEN
Tristan SMAGGHE est désigné chef de projet il a pour rôle d'animer les réunions, suivre l'avancement du projet et donner les axes directeurs à suivre afin de mener à bien le projet.

L'équipe est la même que celle de nos travaux dirigés en CS54. Ce qui fait qu'on avait déjà travaillé ensemble sur des petits projets.

5.2 Analyse de Projet

5.2.1 Définition des objectif

Les objectifs ont été déterminés à l'aide de la méthode SMART : ils sont spécifiques, mesurables, atteignables, réalistes et temporellement définis.

5.2.2 Analyse des risques : Matrices SWOT

Nous avons évalué les forces et les faiblesses de l'équipe mais également les risques et les opportunités qui pourraient impacter la réalisation du projet. Nous avons résumé cela dans une matrice SWOT.

FORCE	FAIBLESSE
- Tristan a beaucoup de connaissances en développement de site web avec bases de données - Équipe motivée et investie - Connaissance de Flask/Python grâce aux cours de CS54 - Tristan sait réaliser une intégration continue pour effectuer des livrables intermédiaires	- Premier gros projet pour la plupart des membres - Manque d'expérience dans la programmation pour LV, Jules et Malo
OPPORTUNITÉ	**MENACES**
- Malo peut avoir de l'aide d'ancien amis de prépa pour rédiger en latex - Tristan peut avoir l'aide d'amis ayant déjà réalisé un SHA0 pour le stockage des mots de passe	- Partiels - Vacances de noël (fêtes)

FIGURE 5.1 – Matrice SWOT

5.3 Organisation du projet

Le projet a commencé durant le mois de novembre pour finir début janvier. Afin que chaque membre de l'équipe puisse travailler sur une partie base de données, une partie web et une partie algorithmique nous avons réalisé un découpage en lots de travail réalisables et individualisés (cf. WBS annexe 3) que nous avons ensuite distribués aux membres de l'équipe en fonction de leurs compétences, leurs affinités et pour chacun ait une quantité de travail équivalente grâce à une matrice RACI (cf. annexe 4). Cet répartition a ensuite été révisée pour s'adapter aux imprévus et à l'avancement plus ou moins rapide des lots de chacun des membres.

5.4 Outils de travail

5.4.1 IDE

L'ensemble de l'équipe a utilisé Visual Studio Code et notamment son extension SQLLite Reader.

5.4.2 Logiciel de gestion de version

Durant la totalité du projet nous avons utilisé le répertoire GitLab de l'école et notre dossier prévu à cet effet. Les clients git utilisés sont GitKraken, et le logiciel git. L'entretien de la branche a été géré par Tristan SMAGGHE : il a notamment organisé le projet et vérifié quotidiennement le bon fonctionnement du git. Une erreur notamment des emails ayant effectué les commits a été remarquée.

5.4.3 Rédaction du rapport

Pour la rédaction du rapport nous avons utilisé le rédacteur et compilateur en ligne Overleaf (l'instance de l'école).

5.4.4 Communication

Pour la communication et l'échange entre les membres du groupe nous avons mis en place un serveur Discord. Sur ce dernier nous avons déployé un bot qui affichait sous forme de notifications chaque commit, ce qui nous permettait d'être au courant des dernières modifications. De plus lorsque nous n'étions pas disponibles pour une réunion en présentiel nous utilisions ce serveur pour faire nos réunions en distanciel.
L'utilisation de cet outil présente tout de même un inconvénient : beaucoup d'informations étaient échangées hors-réunions ce qui peut compliquer le compréhension des compte-rendus successifs.

Chapitre 6

Bilan

6.1 Bilan du travail réalisé sur Amplet

Fonctionnalités implémentées
- Système de Login
- Une page profil pour consulter et modifier ses données
- Consulter et trier les amplets particuliers
- S'inscrire sur l'amplet d'un particulier
- Voter pour une navette
- Créer une Amplet
- Points gagnés à chaque amplet effectuée
- Consulter les Amplets sur lesquelles on est inscrit
- Consulter les produits commandés sur les Navettes
- Gérer ses Amplets, gérer les participants
- Discuter avec d'autres utilisateurs
- (Maire) Définir un nouveau marchand
- (Maire) Définir un nouveau produit
- (Maire) Créer une navette
- (Maire) Fermer une navette et déclencher le vote
- Mise à jour des multiplicateurs suite au vote
- Calcul de trajet (page à l'état d'ébauche)

Fonctionnalités non implémentées
- Page Trajet & affichage des produits à récupérer pour le conducteur de la navette
- Priorisation des participants à la navette selon le nombre de points
- Fermeture automatique des Amplets & Navettes
- Notification Mail à la fin d'une Navette / Amplet
- Gestion du prix selon l'unité choisie

6.2 Retour sur le projet

Erreurs commises
- Projet trop ambitieux -Se justifie par : Premier projet de ce type -Mais aussi une mauvaise gestion de projet -On a sous estimé les tâches et le temps de travail disponible -La définition du projet et l'idée qu'on s'en fesait était trop floue et fluctuante dans le temps, ce qui a provoqué des incohérences dans le projet, et a coûté beaucoup de temps a réparé.

Points positifs à retenir
- Travail en équipe (personne n'était seul dans son coin) - Une bonne ambiance dans l'équipe, pas de dispute même dans les moments difficiles - Bonne redistribution des tâches aux moments cruciaux, bonne - adaptation aux différentes nouvelles tâches de la part de tous les membres -Entraide et partages des connaissances entre les membres, les plus expérimentés ont pris le temps d'expliquer les concepts nouveaux aux autres Tout le monde est resté motivé et investi jusqu'au bout

6.3 Ressentis personnels

6.3.1 Jules BRUNET

Points positifs	Première expérience en équipe sur un projet de ce type agréable Rush de fin tendu mais au final bien vécu Expérience gagné pour ce type de projet
Difficultées rencontrées	Nouveaux outils (SQL Alchemy et git à plusieurs) Manque de motivation par moments (surtout au début du projet) Problème de VM (reboot 2 fois au cours du projet)
Expérience personnelle	Agréablement surpris du travail en équipe Gain d'expérience en base de donnée, html,css et Latex
Axes d'améliorations	Gestion du temps de travail Communication sur la définition même du projet

6.3.2 Louis-Vincent CAPELLI

Points positifs	Première expérience sur un projet d'une telle envergure, en groupe qui plus est. M'a permis de mettre en application l'ensemble des domaines de l'informatique étudiés depuis le début de l'année. M'a permis de me rendre compte du comportement que j'adoptais face à ce genre de travail. Apprentissage de la rédaction d'un rapport.
Difficultées rencontrées	Se mettre dans le rythme au début du projet et travailler régulièrement dessus plutôt qu'en grosses sessions éparses. Utilisation d'un module python pour gérer les bases de données plutôt que SQL.
Expérience personnelle	J'ai particulièrement aimé travailler sur ce projet qui a réellement donné une dimension concrète et a lié ce que j'ai étudié au premier semestre. L'ambiance au sein du groupe était très agréable et chacun aidait les autres. J'ai aussi trouvé stimulant les derniers jours où nous nous sommes beaucoup vus et avons travaillé ensemble simultanément sur le projet.
Axes d'améliorations	Se mettre dans l'ambiance plus tôt pour être un peu moins pressé à la fin. Adopter une manière de rédiger les compte-rendus fixe dès le début du projet. Ne pas avoir les yeux plus gros que le ventre lors de la définition du projet.

6.3.3 Malo DAMIEN

Points positifs	Travail en équipe et particulièrement rush de fin de projet Expérience sur le développement web et l'utilisation de bases de données Équipe projet toujours disponible pour de l'aide
Difficultées rencontrées	Certaines difficultés de communications de mes problèmes Manque de temps à cause de la taille du projet trop ambitieux Corrections de mes algorithmes souvent laborieuses
Expérience personnelle	Apprentissage de l'utilisation de sqlAlchemy Apprentissage de nouveaux algorithme Satisfaction de voir les parties de chacun s'imbriquer ensemble (malgré un degug nécessaire)
Axes d'améliorations	Meilleures gestions des jalons avec des dates fixés Travail plus régulièrement réparti pour éviter l'effet crunch de fin de projet Être plus raisonable sur la taille du projet

6.3.4 Tristan SMAGGHE

Points positifs	Expérience de travail en équipe. Expérience sur l'intégration en continu. Premier projet donc possibiliter d'apprendre beaucoup.
Difficultées rencontrées	Manque de temps / préparation. Gestion de Projet.
Expérience personnelle	Progression en algorithmie et informatique. Progression en LaTeX.
Axes d'améliorations	Redaction du rapport à faire de manière progressive. Gestion de Projet.

6.4 Taux horaires

Personne	Temps sur Gestion de Projet	Temps de Travail	Total
Tristan SMAGGHE	16h	69h	85h
Malo DAMIEN	19h	64h	83h
Louis-Vincent Capelli	24h	47h	71h
Jules BRUNET	16h	53h	69h

FIGURE 6.1 – Tableau des taux horaires des membres du groupe

Annexes

Annexe 1

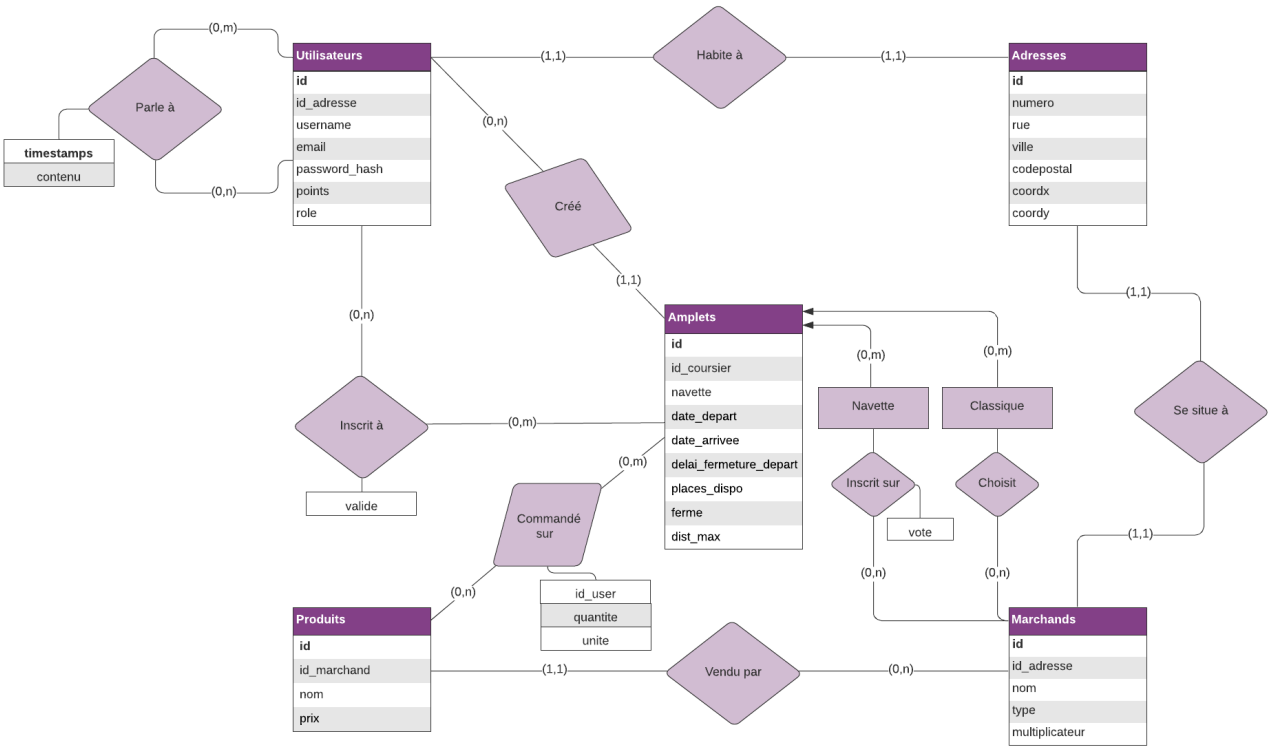

FIGURE 2 – Schéma entité association de notre base de données

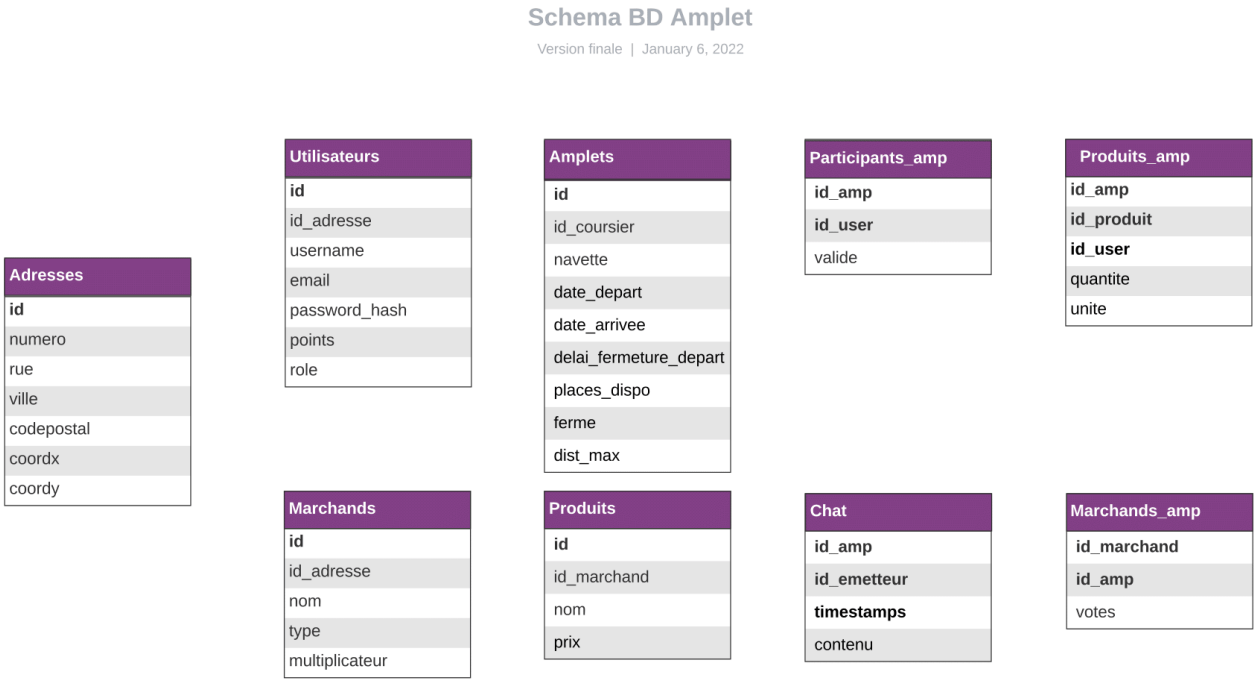

FIGURE 3 – Tables et attributs utilisés par la base de données

Nom de la table	Nom de l'attribut	Description
Utilisateurs	id	Clé primaire définis par un snowflake.
Utilisateurs	id_adresse	Clé étrangère qui permet de localiser l'utilisateur.
Utilisateurs	username	Nom d'utilisateur.
Utilisateurs	email	Adresse mail de l'utilisateur.
Utilisateurs	password_hash	Hash du mot de passe de l'utilisateur.
Utilisateurs	points	Points de l'utilisateur.
Utilisateurs	role	Rôle de l'utilisateur (Maire, Admin, Utilisateur).
Marchands	id	Clé primaire définis par un snowflake.
Marchands	id_adresse	Clé étrangère qui permet de localiser l'utilisateur.
Marchands	nom	Nom du marchand.
Marchands	type	Type du marchand (primeur etc..).
Marchands	multiplicateur	Multiplicateur pour avantager le marchand dans la navette s'il n'a jamais été choisi.
Adresses	id	Clé primaire définis par un snowflake.
Adresses	numero	Numéro de rue de l'adresse.
Adresses	rue	Rue de l'adresse.
Adresses	ville	Ville de l'adresse.
Adresses	codepostal	Code Postal de l'adresse.
Adresses	coordx	Coordonnées X en projection légale de l'adresse (utile pour les plus courts chemins).
Adresses	coordy	Coordonnées Y en projection légale de l'adresse (utile pour les plus courts chemins).
Produits	id	Clé primaire définis par un snowflake.
Produits	id_marchand	Clé étrangère du marchand qui vend ce produit.
Produits	nom	Nom du produit.
Produits	prix	Prix du produit.
Amplets	id	Clé primaire définis par un snowflake.
Amplets	id_coursier	Clé étrangère qui permet de savoir qui à créer l'Amplet.
Amplets	navette	Permet de savoir si cette Amplet est une navette ou non.
Amplets	date_depart	Horodatage de la date de départ.
Amplets	date_arrive	Horodatage de la date de fin prévu de l'Amplet.
Amplets	delai_fermeture_depart	Période de fermeture de l'Amplet avant date_arrive .
Amplets	places_dispo	Places disponibles dans l'Amplet.
Amplets	ferme	Savoir si l'Amplet est fermée ou non.
Amplets	dist_max	Distance maximum pour filtrer les Amplets.
Chat	timestamps	Clé primaire qui permettra de retracer la date et l'heure d'envoi d'un message.
Chat	contenu	Contient le contenu du message.

FIGURE 4 – Tableau dictionnaire qui définis les attributs

Annexe 2

Compte rendu de la réunion du 25/11

Présents

— Tristan
— Jules
— Malo
— Louis-Vincent

Ordre du jour

— Mise en commun des idées de chacun pour notre application
— Choix parmi les idées si l'une d'elles semble prometteuse

Avancement

Mise en commun des idées

LV propose un réseau social où l'on gagne des points de "hype" en accomplissant des actions civiques comme se rendre dans un bureau de vote pour y voter ou participer à la vie de sa commune en assistant à des réunions, des conseils communaux.
Jules et Malo ont réfléchi à une application qui permettrait de poster ses besoins de transports, ses déplacements prévus pour mettre en place un réseau de bus dont l'itinéraire est optimisé pour passer à des endroits qui conviennent au plus grand nombre.
LV propose "BlaBlaCourse" une application qui permet à chacun de profiter des trajets de course des autres, ce qui favorise l'entraide entre citoyens.

Choix du sujet du projet

L'idée de BlaBlaCourse -dont le nom sera modifié car il ressemble trop à des services déjà existant- a plu aux autres membres du groupe et c'est donc l'idée qui a été sélectionnée.

Pour la prochaine réunion le 16/12

Personne affféctée	Tâche
Malo	Commencer le site (page d'accueil, ...) via Botostrap
Jules et LV	Choisir les outils de gestion de projet à mettre en oeuvre
Tristan	Mettre en place l'intégration continue du site
LV	Réfléchir à un nouveau nom et éclaircir la définition du projet pour envoyer un mail à M. Festor

Compte rendu de la réunion du 16/12

Présents

— Tristan
— Jules
— Malo
— Louis-Vincent

Ordre du jour

— Uniformisation des visions du projet
— Choix des outils de gestion de projet

Avancement

LV propose le nom Amplet pour le site : il fait l'unanimité.

Uniformisation des visions du projet

Tristan pensait/souhaitait faire d'Amplet une plateforme sur laquelle les services rendus sont rémunérés mais les autres membres préfèrent que cela soit participatif et bénévole. C'est cette dernière vision qui est adoptée.

Choix des outils de gestion de projet

Jules a proposé une matrice SWOT pour confirmer le choix du sujet, ainsi qu'un WBS pour découper le travail en lot qui pourront être répartis dans une matrice RACI, tout cela une fois que toutes les fonctions envisagées du site auront été définies.

Pour la prochaine réunion le 17/12

Personne affectée	Tâche
Tout le monde	Se réunir pour détailler tous les lots du WBS, et se faire une idée de qui souhaite faire quoi et remplir la matrice RACI
Tout le monde	Commencer à réfléchir à l'architecture de la DB

Compte rendu de la réunion du 17/12

Présents

— Tristan
— Jules
— Malo
— Louis-Vincent

Ordre du jour

— Explication de l'intégration continue par Tristan aux autres
— Mettre en commun les idées pour la DB

Avancement

Explication de l'intégration continue

Tristan s'est occupé de mettre en place une intégration en continue afin de pouvoir facilité le développement. Le site est automatiquement mis à jour sur : (https ://amplet.fr)

Afin de lancer cette mise à jour, il nous suffit d'aller sur la page du projet sur Gitlab, puis d'aller dans "Tags", puis de créer un tag avec pour nom le numéro de la version.

Mettre en commun les idées pour la DB

On s'est rendu compte après avoir esquissé un schéma que l'architecture allait être amenée à beaucoup évoluer au gré des divers fonctionnalité que chacun implémentera. LV mettra donc à jour sur git le schéma de la DB actuelle de temps en temps, après chaque changement.

Pour la prochaine réunion le 23/12

Personne afféctée	Tâche
Malo	Implémenter le register et le login
LV et Jules	Continuer leurs pages respectives (cf. backlog)
Tristan	Gestion du mot de passe

Compte rendu de la réunion du 23/12

Présents

— Tristan
— Jules
— Malo
— Louis-Vincent

Ordre du jour

— Refonte du schéma de base de donnée et passage en 3NF

Avancement

Refonte du schéma de base de donnée et passage en 3NF

Chacun fait part de ses besoins vis-à-vis de la base de données et nous réflechissons à la manière de l'implémenter, et de la passer en 3NF.

Pour la prochaine réunion le 28/12

Personne affféctée	Tâche
Tout le monde	Avancer chacun sur les lots qui nous ont été attribués dans la matrice RACI dans la mesure du possible
Tout le monde	Essayer de travailler un peu même si c'est Noël
Malo	Travailler sur l'algo du plus court chemin
Jules	Link la base de données à la page des Amplets en cours
LV	Link la base de données à la page Mes Amplets
Tristan	Implémenter le chat

Compte rendu de la réunion du 28/12

Présents

— Tristan
— Jules
— Malo
— Louis-Vincent

Ordre du jour

— Point sur les tâches de chacun et sur les problèmes rencontrés

Avancement

Point sur les lots de chacun

— Réflexions sur la possibilité ou non d'optimiser les clients sélectionnés par un particulier en fonction de leur liste du courses et de la capacité de la voiture : difficile à mettre en place mais le coursier peut refuser lui-même un client s'il constate que la liste de courses demandée est trop chargée.
— Distance entre deux points : utilisation d'une api pour convertir les adresses des clients en coordonnées et pouvoir mettre en œuvre un algorithme de parcours de graphe pour calculer le plus court chemin. (DB à moifier du coup)
— Mise à jour du schéma de la BD en fonction pour qu'elle prenne en compte les modifications qui ont été faites depuis la dernière réunion.
— On va utiliser une api pour faire en sorte de pouvoir fermer une Amplet X heures après sa création.

Pour la prochaine réunion le 31/12

Personne affféctée	Tâche
Tristan	Implémenter SHA pour les mots de passe
Tristan et Malo	Commencer à rédiger la partie technique du rapport
Jules	Partie État de l'art du rapport
LV	Partie GdP du rapport

Compte rendu de la réunion du 31/12

Présents

— Tristan
— Jules
— Malo
— Louis-Vincent

Ordre du jour

— Points sur les jalons
— Répartition des lots qui restent
— Points sur l'avancement du rapport

Avancement

Points sur les jalons

— Navette : Corrections successives de la navette, le form est enfin fonctionnel (sans erreur) et permet de créer une participation dans la BD. Il ne reste plus que le vote.
— Consultation / Inscription sur les Amplet des particuliers : Mise en forme fonctionnelle du système de consultation et de recherche d'amplets, inscriptions sur les amplets en forme (à relier à la base de données)
— Création d'amplets particuliers : il reste à lier à la BD le post d'une Amplet pour les stocker, et la possibilité d'accepter ou non un participant.

Répartition des lots qui restent

— Il reste 1.4 que Jules récupère car c'est très lié à 1.3 qu'il a réalisé
— Les adresses précises des utilisateurs ne sont pas stockées, seulement les codes postaux, on retire donc les fonctionnalités Distance maximale (1.5.2) et le tri des Amplet par proximité (qui reste à l'état d'ébauche)
— Pour les plus courts trajets l'équipe est d'accord pour faire une recherche exhaustive tant que le nombre de magasins ne dépasse pas 6. → 60 chemins

Points sur l'avancement du rapport

— Tristan et Malo ont avancé sur la partie technique du rapport notamment la section performance et test mais également la partie développement. LV va s'occuper de la partie gestion de projet et de corriger les fautes.
— La matrice RACI a été actualisée et le schéma E/A aussi. Il faudra régler le problème entre adresse et coordonnées pour qu'elle soit à nouveau en 2NF.

TODO

Personne affectée	Tâche
Jules	A un instant T, faire une fonction qui calcule à partir d'un id.amp : -la liste des magasins choisis -la liste des participants (et leurs produits) choisis -Actualise le score des magasins
Malo	Link l'envoi de la liste à la base de données (user mis 'en attente' dans participant.amp)
LV	Afficher à l'utilisateur quels produits vont lui être servis ou non sur la page commande
Tristan	Faire, sur le serveur, un système qui permet de fermer les votes d'une navette donnée à une date donnée et de lancer le processus de 'fin de vote'.

Annexe 3

WBS

1. Fonctionnalités utilisateur

- 1.1 Gestion du compte
  - 1.1.1 Register / login
  - 1.1.2 Modification des informations (mot de passe, adresses)
  - 1.1.3 Personnalisation de compte (photo, bannière, pseudo, description)
- 1.2 Chat
  - 1.2.1 DM
  - 1.2.2 Fil de discussion propre à chaque Amplet
- 1.3 Consultation des Amplet à proximité
  - 1.3.1 Liste des Amplet classées par proximité
  - 1.3.2 Recherche par type de produit
- 1.4 Inscription sur l'Amplet d'un particulier
  - 1.4.1 Possibilité de postuler sur l'Amplet d'un particulier
  - 1.4.2 Possibilité de spécifier sa liste de course à l'inscription
- 1.5 Post d'une proposition d'Amplet en tant que particulier
  - 1.5.1 Choix d'un ou plusieurs types de magasins
  - 1.5.2 Possibilité de mettre une 'Distance maximale' des potentiels bénéficiaires
  - 1.5.3 Possibilité d'accepter ou non une demande, et de fermer l'Amplet lorsque plein
- 1.6 Post d'une demande d'Amplet sur la navette
  - 1.6.1 Spécifier les produits demandés
  - 1.6.2 Accepter la redirection vers un particulier

2. Gestion des courses

- 2.1 Navette
  - 2.1.1 Calcul du trajet optimal
  - 2.1.2 Choix des commerçants (vote)
    - 2.1.2.1 Attribution de points à chaque commerçant qui propose le produit demandé
    - 2.1.2.2 Modificateur propre aux commerçants qui diminue avec la fréquence de participation à la navette
  - 2.1.3 Valeur du vote des particuliers augmente avec leur score
- 2.2 Date livraison, date arrivée
  - 2.2.1 Fermeture de l'Amplet 6 heures avant le départ
  - 2.2.2 Envoi d'un mail lorsque les produits sont disponibles
- 2.3 Système de score des utilisateurs
  - 2.3.1 Calcul du score en fonction des actions
  - 2.3.2 Mise à jour du score de l'utilisateur
  - 2.3.3 Système de vérification du statut "handicapé"

FIGURE 5 – WBS

Annexe 4

Matrice RACI

Lot	Tristan	Malo	Jules	LV
1.1	C	RA		
1.2	RA			
1.3	CI		RA	I
1.4	C		RA	
1.5	C			RA
1.6	C	RA		
2.1.1	C	RA		
2.1.2	C		RA	
2.1.3	C			
2.2	C		RA	
2.3.1	C			RA
2.3.2	C			RA
2.3.3	RA			

FIGURE 6 – Matrice RACI

Bibliographie

[1] Calcul de complexité du tri fusion - laure gonnord. http://laure.gonnord.org/pro/teaching/AlgoProg1213_IMA/complexite_fusion.pdf. [En ligne ; lu le 5 Janvier 2022].

[2] The civic tech field guide. https://civictech.guide/timeline/. [En ligne ; lu le 30 Décembre 2021].

[3] Civic technology - wikipedia. https://en.wikipedia.org/wiki/Civic_technology. [En ligne ; lu le 30 Décembre 2021].

[4] Cours de sébastien le digabel sur le problème de voyageur de commerce. https://www.gerad.ca/Sebastien.Le.Digabel/MTH6311/8_applications_2.pdf. [En ligne ; lu le 27 Décembre 2021].

[5] Generate all permutations of an array. https://www.baeldung.com/cs/array-generate-all-permutations. [En ligne ; lu le 31 Décembre 2021].

[6] Mission et stratégie. https://www.opengovpartnership.org/fr/mission-and-strategy/. [En ligne ; lu le 6 Janvier 2022].

[7] Page wikipedia concernant les snowflakes. https://en.wikipedia.org/wiki/Snowflake_ID. [En ligne ; lu le 8 Décembre 2021].

[8] Page wikipedia de l'algorithme de heap pour générer des permutations. https://en.wikipedia.org/wiki/Heap/. [En ligne ; lu le 1er Janvier 2022].

[9] Qu'est ce qu'une civic tech et que peut-elle apporter ? https://www.consultvox.co/blog/civic-tech-definition-apports/. [En ligne ; lu le 6 Janvier 2022].

[10] FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION. Secure hash standard. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf/, 2015. [En ligne ; lu le 27 Décembre 2021].

[11] Grégoire de Turckheim Rowena Jones. Iot hub : What use case for websockets ? https://blog.scaleway.com/iot-hub-what-use-case-for-websockets/, 2021. [En ligne ; lu le 23 Décembre 2021].