1 September 2019
Alexandre ALLANI

Encadrant : Jean PRUVOST, consultant en datascience
Conseiller d'étude : Romain BILLOT
Responsable filière : Cécile BOTHOREL

Rapport de stage de fin d'étude

Consultant en datascience chez Sia Partners

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

1 Remerciements

Premièrement, je tiens à remercier **M. Pierre Leplatois** et **M. Pierre – Antoine Merle**, les directeurs de l'Unité de Compétence (UC) data science de Sia Partners, de m'avoir accueilli lors de ce stage. Je remercie Pierre de m'avoir m'avoir permis de travailler sur des problématiques orientés plus data engineering et pas seulement data scientist.

Je remercie **M. Matthieu Thiboust** et **M. Nicolas Guinard** pour leurs conseils techniques leurs intéressements aux divers sujets traités par l'équipe data.

De plus je remercie tout particulièrement **Paul Saffers** et **M. Gabriel Creti** pour m'avoir accompagné et fourni l'aide nécessaire pendant mes missions.

Je suis particulièrement reconnaissant envers l'ensemble des membres de l'UC de Data Science qui ont su me consacrer de leur temps pour me former. De plus, je les remercie pour leur accueil chaleureux, leur gentillesse et leur bienveillance tout au long de ce stage.

Contents

2 Introduction

Dans le cadre de ma dernière année à IMT Atlantique, je devais réaliser un stage de 6 mois en entreprise afin de perfectionner mes compétences d'ingénieur et de faciliter mon insertion professionnelle.

Ayant suivi le parcours datascience de la fillière 3, au cours de ma dernière année, je désirais trouver un stage spécialisé dans ce domaine afin de mettre en pratique dans un cadre professionnel les formations que j'ai suivies au cours de mon premier semestre de 3ème année.

J'ai choisi d'effectuer mon stage chez Sia Partners pour les raisons suivantes :

- Sia Partners m'est apparu comme une alternative aux grandes entreprises de conseil. J'étais intéressé par le fait de profiter des avantages du conseil, à savoir pouvoir expérimenter plusieurs domaines et secteurs d'activités au cours des missions, tout en évoluant dans une entreprise à taille humaine.
- Les sujets traités par l'équipe data correspondaient exactement à mes attentes. De plus, les consultants data scientists travaillent sur les technologies avec lesquelles je souhaitais évoluer lors de mon stage.
- Sia Partners possède un incubateur de startups avec lesquels nous sommes amenés à travailler. Je trouvais très intéressant d'avoir la possibilité de travailler avec des startups tout en étant dans une entreprise de conseil en management. J'ai eu l'occasion de travailler 3 mois avec l'une d'entre elles.
- Une grande partie du travail de consultant data scientist est technique, ce qui est le point qui m'a le plus intéressé lors des entretiens techniques.

Je chercherai dans ce rapport à présenter d'abord le contexte général dans lequel s'inscrivent mes différentes missions avant de développer le déroulement de ces dernières, en insistant sur ce que j'ai pu apporter dans ces différentes missions. Pour chacune de ces missions, j'essaierai d'approfondir certains point mettant en évidences mes capacités d'ingénieurs.

Dans un premier temps, ce rapport présentera le contexte dans lequel mon stage s'est déroulé avec une présentation de l'entreprise ainsi que la présentation de l'Unité de Compétence Data Science à laquelle j'étais rattachée et ses missions. Nous rentrerons ensuite dans la présentation de mes missions. Dans une troisième partie j'étudierai la Politique de Responsabilité Sociétale de l'Entreprise Sia Partners. Enfin, je conclurai sur un bilan personnel de ce que ce stage a pu m'apporter et j'exposerai mes perspectives professionnelles à l'issue de cette expérience professionnelle.

3 Présentation de l'entreprise

3.1 Historique

J'ai effectué mon stage de fin d'études chez Sia Partners. Ce cabinet se situe dans le 8ème arrondissement de Paris, près des Champs-Élysées. Sia Partners est un cabinet de conseil en management et stratégie opérationnelle. Il a été fondé en 1999 à Paris par M. Matthieu Courtecuisse. L'entreprise s'est d'abord développée au sein de ses bureaux parisiens avant d'ouvrir des succursales à l'international. Le développement du jeune cabinet s'est ensuite effectué par acquisitions, ses fonds et sa réputation le lui permettant. Afin de changer son image de cabinet « franco-français », le nom même de l'entreprise a été modifié en 2013 : Sia Conseil est devenu Sia Partners.

Avec un effectif de plus de 1400 collaborateurs répartis sur 25 bureaux dans 16 pays (Europe, Amérique du Nord, Asie, Moyen-Orient), Sia Partners est le leader du conseil en management en France. De plus, Sia Partners est une société de partenariat détenue à 100% par ses dirigeants.

La société s'est construite autour du secteur des services financiers avant de développer son expertise dans les secteurs de l'énergie, les télécoms et médias, le transport et logistique, dans le manufacturing ou encore dans le secteur public. Les clients sont de tailles très variées et près de la moitié des entreprises du CAC 40 ont fait ou font encore appel à Sia Partners.

Sia Partners affiche pour son chiffre d'affaires une croissance annuelle à deux chiffres depuis sa création ainsi qu'une croissance en termes d'effectif très importante, doublant ses effectifs en 4 ans.

3.2 Unité de Compétence Data Science

J'ai effectué mon stage de fin d'études au sein de l'unité de compétence (UC) Data Science de Sia Partners. Il s'agit d'une unité transverse qui est donc amenée à travailler sur des missions issues de tous secteurs. Née en janvier 2015 d'un fort besoin des entreprises d'être accompagnées dans leur transformation digitale et la valorisation de leurs données, l'équipe se compose à l'heure actuellement de 68 consultants data scientists.

Le rôle de ces consultants data scientists est de répondre aux problématiques des clients de Sia Partners par la mise en place d'une solution à forte composante en machine learning. Cela passe donc par :

- La compréhension des enjeux métiers : le data scientist est souvent accompagné d'un consultant du secteur en question
- La récolte des données nécessaires à l'étude
- La mise en place du modèle statistique
- La mise en place du modèle statistique
- La restitution des résultats clés au client.

Il s'agit donc, en d'autres termes, de guider les entreprises dans le contexte de la quatrième révolution industrielle que nous vivons actuellement, et dont la donnée, sous toutes ses formes, est le catalyseur.

L'équipe Data Science a connu une croissance exponentielle : créée il y a 4 ans, elle comptait il y a 2 ans 20 consultants, elle en compte aujourd'hui plus de 60 et l'objectif est à la hausse pour 2020. La Data Science à une importance stratégique pour le Matthieu Courtecuisse, PDG de Sia Partners.

Déjà de nombreuses missions ont été effectuées notamment dans le domaine de l'énergie (ENEDIS, EDF, ENEXIS), de la banque (Société Générale), de l'immobilier (Nexity, Vinci), du transport (RATP, Vinci), de l'assurance (AXA, Pacifica) et même dans le secteur public (DDTM, DGCCRF). L'UC Data Science présente son savoir-faire au travers d'une plateforme web, le DataScience Lab, accessible à l'adresse suivante : *https://datascience-lab.sia- partners.com/*.

Cette plateforme présente une version de démonstration des projets réalisés en interne. Ces projets tentent de répondre à des problématiques métiers en utilisant un maximum de données en open source, comme les données de la SNCF ou de Météo France, par exemple. Il s'agit pour la plupart de bots informatiques, c'est-à-dire d'interfaces logiciels qui interagissent avec des serveurs informatiques de manière quotidienne afin de récupérer et mettre à jour des données ou utiliser des moteurs de calcul pour modéliser une grandeur par exemple.

Les bots réalisés sont mis en avant au travers une campagne d'information sur LinkedIn. Le but est de susciter le besoin chez de potentiels clients qui trouveraient dans nos bots un cas d'usage au sein de leur entreprise, mais également de montrer comment nous nous emparons des technologies pour répondre à des problèmes concrets.

3.3 Organisation de l'entreprise

Une des particularités de Sia Partners en tant que cabinet de conseil est son organisation matricielle, visant à regrouper les consultants par unité de compétences (UC). Ainsi, en plus d'avoir des équipes dédiées à chaque secteur (énergie, transport, secteur public, banque, etc.), Sia Partners a fait le choix d'une division transversale à ces secteurs en créant des équipes fonctionnelles (transformation digitale, conseil aux DSI, aux directions financières, etc.), ce qui permet de valoriser à la fois expertises techniques et connaissances métier.

Les principes de l'organisation interne de Sia Partners s'articulent autour de trois grands axes :

- Une grande flexibilité
- Une organisation sur un design plat basé sur une proximité étroite entre l'équipe de direction et les consultants
- Un compte de résultat global unique basé sur une approche homogène entre les entités géographiques et les pratiques du groupe.

Comme expliqué précédemment, le cabinet est organisé selon une approche à la fois sectorielle et fonctionnelle. Les 6 UC sectorielles sont les suivantes :

- Assurance
- Banque
- Télécommunication, Médias et Digital
- Energie, Utilité et Environnement
- Transport Industrie et Distribution
- Secteur Publique

Les 7 UC transverses sont les suivantes :

- Actuariat
- Performance Financière
- Stratégie IT
- Conformité
- Ressources Humaines et Transformation
- Performance de la fonction achat
- Data Science

L'entreprise réalise l'essentiel de son chiffre d'affaires grâce aux UC Énergie et Finance. Ceci s'explique du fait qu'elles sont les unités de compétences les plus anciennes et disposent du plus grand nombre de consultant ainsi que de plus gros comptes.

Avec un CA de 5 millions d'euros en 2017, l'UC Data Science vise à être l'un des piliers dans la stratégie de développement du cabinet. Il est amené à devenir un pôle transverse aux différentes unités de compétences, pour être en appui sur des problématiques innovantes autour de la donnée au contact d'autres UC plus à l'aise sur leur expertise métier. Il est fréquent de voir des consultants d'une UC travailler sur des missions d'une autre UC.

Des fonctions supports et d'administration permettent aux consultants de mener à bien leurs missions. Sia Partners dispose également de son propre institut de formation, le Sia Institute. Le cabinet multiplie les formations en interne pour faire monter en compétences ses consultants sur des sujets divers et variés. Ces formations sont dispensées par des consultants plus expérimentés, sur des thématiques orientées métier ou sur des outils utilisés dans le quotidien du consultant.

Le cabinet dispose également de son propre incubateur de startup (Studio) créé en 2017. Il a été doté d'un fond de cinq millions d'euros. Le but pour le cabinet est entre autres de s'intégrer plus facilement à l'écosystème startup, de soutenir des projets grâce à l'expertise interne et de renforcer son image de cabinet innovant.

4 Mission : webscrapping pour Cleep (1 mois)

Cette partie est consacrée à la première mission que j'ai effectué sur le sujet du webscrapping pour Cleep. Le webscrapping est la collecte de données/informations sur des sites internet via un ou plusieurs scripts. Le but sera alors principalement de "piocher" dans le code source du site internet à webscrapper, ou via les requêtes effetuées par le serveur, les données que l'on souhaite récupérer.

Cette mission mêle à la fois des problématiques de récupérations de données, mais aussi d'automatisation de webscrapping via certains algorithmes que j'ai développé.

4.1 Présentation de Cleep

Cleep est une start-up créée en 2017, développant une application éponyme sur le sujet de l'achat en ligne. L'application permet de sauvegarder un produit repéré sur un site marchand quelconque (que ce soit un site généraliste comme Amazon ou un site spécialisé comme Asos). Cette application mobile est disponible sur mobile (iOS & Android) mais aussi via un site Internet [1]. Le principe de l'application est de pouvoir enregistrer n'importe quel produit pour ensuite le retrouver dans l'application. Il est alors possible de consulter via cette dernière le prix, la description, le nom, le site d'origine ainsi que les images reliés à ce produit. Le but de l'application est donc de faciliter l'achat de produits en ligne en ne passant que par une seule et unique plateforme. Par ailleurs, il est possible de créer une liste de "Cleep" (ie de produit) et de partager ces listes. Les listes publiques peuvent être trouvées via le moteur de recherche intégré.

Figure 1: Logo de Cleep

Sia Partners a investi 400000€ dans Cleep fin 2018, et travaille en collaboration sur différents sujets, notamment en matière de datascience. En plus du côté simplification du shopping en ligne, Cleep avec ses listes et son moteur de recherche permet de suivre des listes plus ou moins influentes. Ainsi l'application permet de se tenir au courant des modes actuelles et de celles à venir. Cela explique l'investissement de Sia Partners.
Vocabulaire spécifique à Cleep :

- cleep : Produit en ligne, enregistré dans la base de donnée. Un cleep est composé d'un prix, des images associés au produit, d'une description, du nom du produit ainsi de du site internet d'origine.
- liste de Cleep : Liste de Cleep, pouvant être partagé entre plusieurs utilisateurs. Chaque cleep est enregistré dans une liste.
- cleeper : Action d'enregistrer un produit dans l'application

J'ai travaillé pendant près de 3 mois pour Cleep. Je communiquais principalement avec Damien Meurisse côté Cleep et Paul Saffer côté Sia Partners.J'ai travaillé sur deux sujets :

- Webscrapping : Acquisition et traitement des données. Cela est fondamental pour le fonctionnement de Cleep
- Moteur de recommandation. Avant mon arrivé, aucun moteur de recommandation n'existait. Cette fonctionnalité était de plus en plus nécessaire afin que l'utilisateur puisse chercher de nouveau cleeps autrement que par le moteur de recherche présent dans l'application.

4.2 Objectifs et problématique

Cleep est une application reposant sur les articles et produits qu'elle permet d'enregistrer, mais aussi sur les méta-données qui leurs sont reliés. En effet, en allant sur Cleep, l'utilisateur doit pouvoir retrouver l'ensemble des informations qu'il recherche, sans pour autant retourner sur le lien de l'article. Il existe deux moyens d'enregistrer un produit :

- Par capture d'écran du site : l'image est enregistré dans l'application. L'utilisateur peut alors renseigner lui même les informations dont il a besoin (ie le prix, le nom de l'article, etc.). Le cleep créé est composé de la capture d'écran et des informations renseignés.
- Avec le lien du site : Grâce à un procédé, détaillé plus tard, le produit avec ses images et ses méta-données sont cleepés.

En général l'enregistrement via capture d'écran n'est pas optimal pour l'application. En effet, les utilisateurs prennent rarement le temps d'enregistrer toutes les informations, ce qui les intéresse en priorité est de garder l'image et le nom du produit dans l'application afin d'avoir un souvenir de ce dernier. Le cleep ainsi enregistré n'est pas utile pour Cleep car manquant de beaucoup d'informations.
En effet, Cleep possède une base de donnée dans laquelle les produits sont enregistrés avec leur méta-données. Un produit est composé des éléments suivant :

- URL du site internet
- Nom
- Prix d'origine
- Prix avec réduction (si disponible)
- Description (si disponible)
- URL des différentes images

Un cleep et un produit sont deux entités différentes. Un produit est une entité composée des éléments énoncés précédemment, tandis qu'un cleep est une entité pouvant faire référence à un produit créé par un utilisateur et avec lequel l'utilisateur interagit. Par ailleurs le produit est définit de manière unique par son URL. Par exemple, les mêmes chaussures disponibles sur un site A et sur un site B sont considérés comme deux produits différents. Le schéma ci-dessous explique la structure.

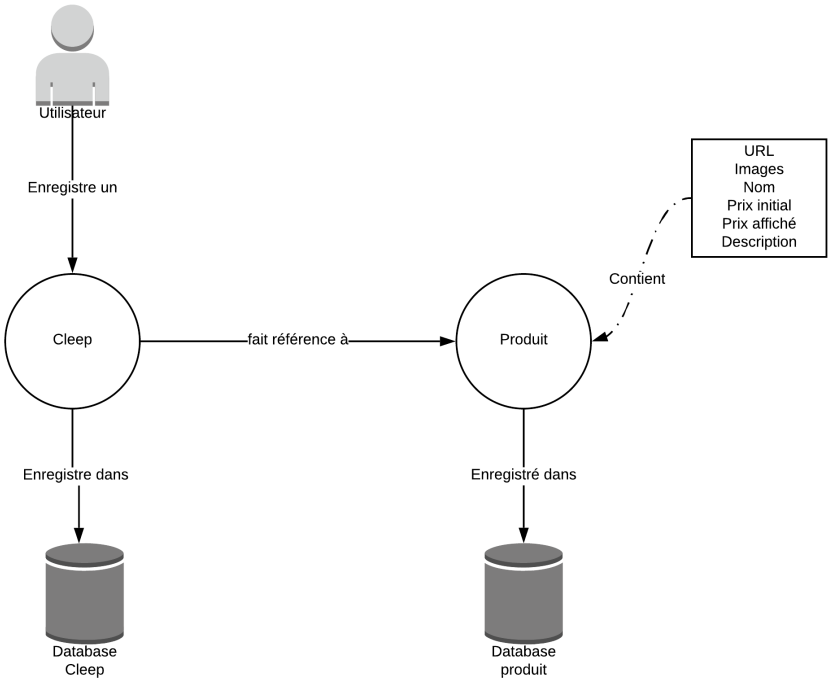

Figure 2: Organisation des données de cleep

Relier un cleep à un produit est un défi pour l'application, car les utilisateurs sont plus enclins à rester dans l'environnement Cleep lorsqu'ils ont les informations à leur disposition dans l'application. En effet les cleeps contenant ces information, l'utilisateur est moins tenté de suivre le lien du produit et a plus de chance de rester sur l'application et parcourir les listes de clepp d'autres utilisateurs.
Cependant pour pouvoir faire cela, il faut récupérer les informations des produits au préalables. C'est ici qu'intervient le webscrapping. Les méta-données qui nous intéresse sont récupérées par ce moyen et entré dans la base de donnée. L'objectif de cette mission était donc d'enrichir la base de donnée des produits de manière automatique.

4.3 Travail effectué

Cette sous-section relate le travail effectué sur cette mission.

4.3.1 Contexte

Avant mon arrivé, plusieurs methode de webscrapping étaient déjà utilisées:

- Méthode automatique
- Méthode spécifique

4.3.1.1 Méthode automatique:

Cette méthode était la plus utilisée pour récupérer les informations des produits. L'utilisateur rentre un lien dans l'application, par la suite ce lien est envoyé au serveur sur lequel la méthode est implémentée. Le serveur génère un navigateur internet sans interface graphique (headless browser) et requête le lien internet rentré par l'utilisateur. Ensuite, en analysant les balises du code HTML chargé, l'algorithme par certaines règles essaie de récupérer les méta-données. Ces informations sont alors transférés à l'utilisateur. Le schéma ci-après décrit le processus. Bien que cette méthode de scrapping soit indépendant du site internet, elle reste néan-

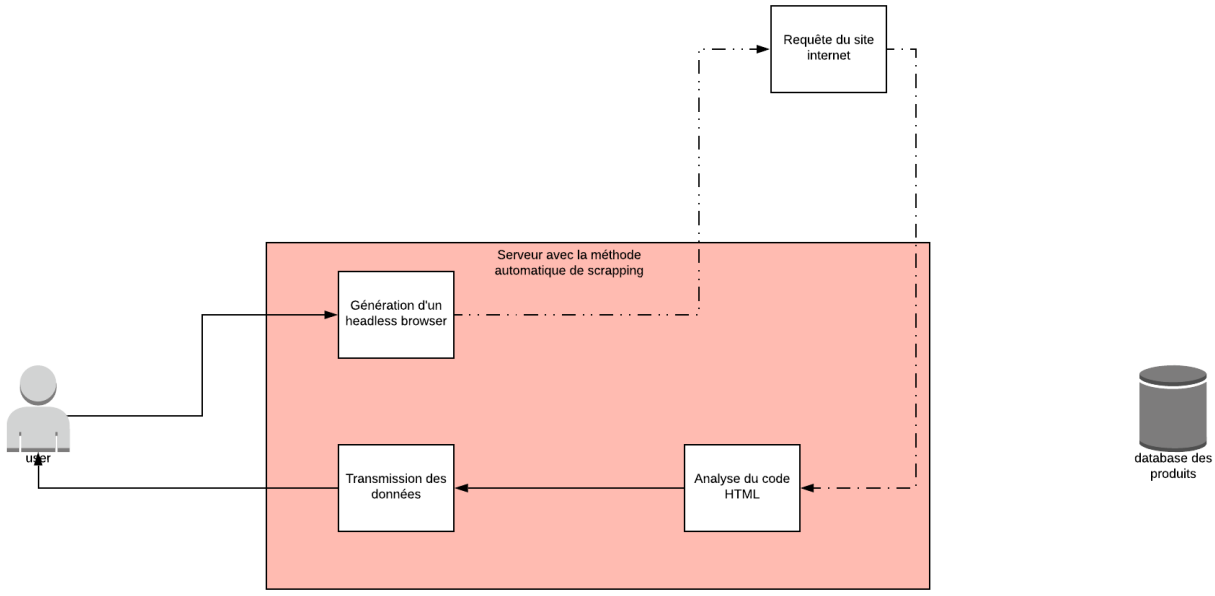

Figure 3: Méthode automatique

moins assez "précaire", car il est rare d'avoir toutes les informations nécessaires à disposition. Il est fréquent d'avoir certains attributs manquant, car les règles créées pour l'analyse du code HTML n'était pas assez vaste. Prenons les deux sites internet suivant :

Site N°1:

```
<!DOCTYPE html>
<html>
<head>
    <meta name="price" content="99EUR">
</head>
<body>
    <h1>My shopping website</h1>
    ...
</body>
</html>
```

Site N°2:

```
<!DOCTYPE html>
<html>
<body>
    <h1>My shopping website</h1>
    <p>Prix: 99 EUR</p>
</body>
</html>
```

Dans le cas N°1, il est facile de mettre une règle pour récupérer le prix dans le meta tag. Dans le cas N°2 cependant, le prix est "juste" inclut dans une balise <p>, rendant la règle précédente inutile. La méthode automatique implique donc beaucoup d'erreurs et d'approximations. Pour des soucis de mémoire, ces données ne sont pas enregistrées dans la base de donnée des produits, afin de ne pas garder les "fausse" données récupérées. Cette méthode pose un autre soucis, c'est son temps de réponse. En effet, afin de scrapper le site, le serveur est obligé de générer un headless browser, de faire la requête, analyser le code HTML pour finalement retourner la réponse. Le temps de réponse doit être relativement bas, car l'utilisateur au moment du cleep doit pouvoir continuer ses activités sur Internet sans être interrompu trop longtemps par le temps que met Cleep à cleeper son produit, et donc eviter toute frustration.

4.3.1.2 Méthode spcéifique:

C'est sur ce sujet que s'est concentré mon travail. Le principe de cette méthode est de prendre les sites internet un à un et d'appliquer un script de webscrapping qui leur est propre. Cela permet d'éviter les problèmes rencontrés précédemment. La structure du code HTML derrière les pages produits est dans la quasi-totalité des cas la même pour un nom de domaine particulier. Il est possible que selon les sous-domaines d'un site cette page varie, par exemple pour la fnac on a:

- https://livre.fnac.com/a13195869/Agnes-Martin-Lugand-A-la-lumiere-du-petit-matin
- https://jeux-video.fnac.com/a13608859/LEGEND-OF-ZELDA-LINKS-AWAKENING-FR-SWITCH-Jeu-Nintendo-Switch
- ...

Dans ce cas, on distingue les différents cas et ensuite on analyse le code HTML de chacune de ces pages. Cette méthode est bien plus fiable et donne de meilleurs résultats sur les méta-données requise.
Par ailleurs, afin de réduire le temps de réponse, une difficulté est rajouté : la génération d'un headless browser est supprimée m'obligeant à travailler avec le code source de la page (ie le résultat d'une requête cURL à la page), et les requêtes API.
En effet la génération d'un navigateur permettait de faciliter la tâche, car le code HTML étant totalement généré, il suffisait de récupérer le contenu des balises afin d'obtenir les informations nécessaires. Cependant via une requête cURL (ie le code source de la page), les script JavaScript ne sont pas déclenchés, il faut donc chercher au travers des differents script et requêtes API les éléments que l'on souhaite récupérer.

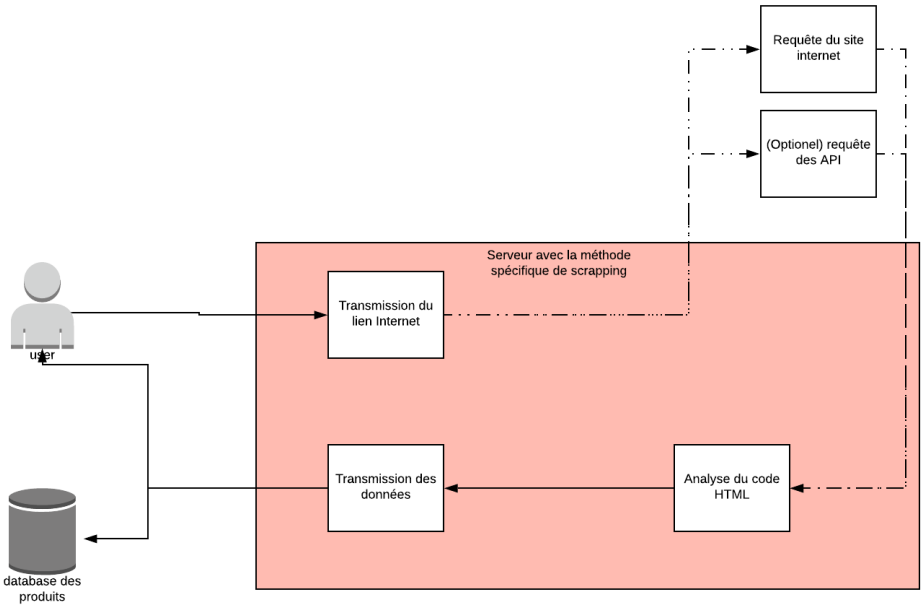

Figure 4: Méthode spécifique

4.3.1.3 Démarche

Le processus **figure 4** montre les différentes étapes de la méthode spécifique. Mon travail s'est principalement concentré sur l'algorithme analysant un site web en particulier et la gestion des requêtes.
Prenons l'exemple suivant [2] : Les encadrés rouges correspondent aux éléments que l'on doit récupérer. Vous

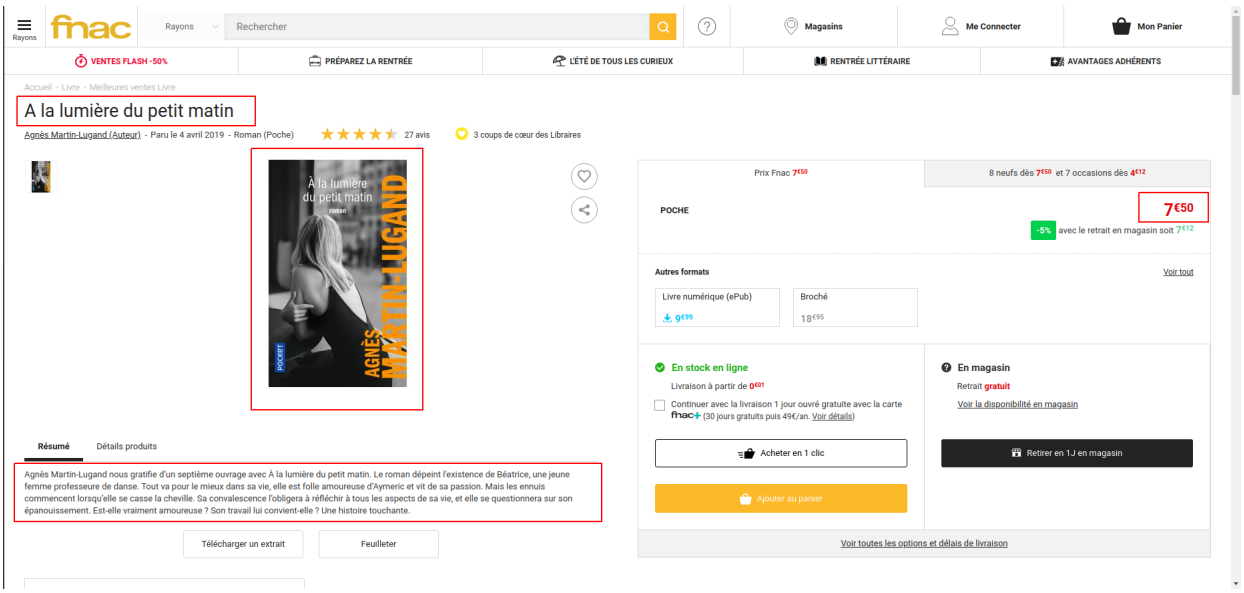

Figure 5: Exemple fnac

pouvez retrouver en annexe une version allégé du code HTML (Annexe 9.1). Il est alors facile de comprendre que récupérer l'attribut $data-src-zoom$ de la balise <li> nous permet de récupérer l'image et le JSON quant à lui permet de récupérer les autres éléments requis.
La démarche adopter pour scrapper les différents sites à toujours été la même. Tout d'abord on analyse le code source afin de savoir quels éléments sont à récupérer. Ensuite on teste avec plusieurs liens différents afin de tester la robustesse de l'algorithme. Une validation finale était faite côté client, et ensuite l'algorithme était mis en production.

Le dernier point nécessitant du travail : où trouver les liens permettant de scrapper les produits et peupler

la base de donnée? L'ensemble des liens déjà cleepés sont enregistrés, et donc peuvent être utilisé pour répondre à cette question. Le client possédait déjà un script permettant de trier les liens par nom de domaines. Ainsi pour chaque nom de domaine que j'ai traité, j'ai du trié les liens pertinents (ie de produits) des liens inutiles (page principal, page de catégories etc). Par exemple, pour la fnac certains utilisateurs ont cleepé des pages présentant plusieurs produit [3] ce qui ne convient pas au processus détaillé auparavant.

4.3.1.4 Résultats

A la fin de cette mission, j'ai développé un script de webscrapping pour plus de 56 sites différents. Cela a permis à Cleep de peupler sa base d'environ 220 000 produits différents.
Sur une vision plus large, les utilisateurs passent en moyenne plus de temps sur l'application (effet recherché), mais aussi ils ont tendance à cleeper plus de produit sur les sites dont le webscrapping a été fait.

4.4 Vision critique et apport personnel

La mission de webscrapping était plutôt simple, dans le sens qu'une fois les notions de webscrapping acquises, cette mission consistait à analyser un code HTML et les requêtes faits en javascript, pour trouver les informations nécessaires. Mon apport personnel s'est plus concentrés sur les discussions avec le client au sujet de l'amélioration du scrapper générique et le choix des technologies ainsi que la compréhension de l'infrastructure de Cleep.

4.4.1 Scrapper générique

Le scrapper générique déjà présent n'était pas assez efficaces pour pouvoir enregistrer les données des produits et il était trop lent pour être utilisé en production indéfiniment (pour l'instant, si un lien n'appartient pas à la liste des 56 domaines scrappés, il passe par le scrapper générique). Comme il n'est pas non plus possible de scrapper de manière spécifique l'ensemble des sites commerciaux existants (la base de donées de Cleep compte plus de 8000 nom de domaines différents), j'ai discuté avec le client sur la nécessité d'un scrapper générique plus performant.
Ainsi en parallèle, j'ai essayé de developper un scrapper générique plus performant n'utilisant pas un "headless browser". La valeur ajouté de Cleep reposant sur les images du produit enregistré dans l'application, j'ai travaillé sur un scrapper générique permettant de récupérer les images du produit.
Ce scrapper suit les 2 étapes suivantes :

- Récupérations des liens "dynamique" de la page HTML
- Clustering des liens

4.4.1.1 Récupérations des liens "dynamique" de la page HTML:

Cette étape consiste à récupérer l'ensemble des liens images de la page n'étant des liens statique. Un lien image est considéré comme statique s'il est présent plusieurs fois sur différentes pages. En générale ces images sont les logos et autre widgets communément présents. L'exemple **figure 6** ci-après montre les images statiques sur le site de la fnac. Une fois connus, ces liens ne sont pas pris en compte par le scrapper générique.
Ainsi, à la fin de l'étape 1 (cf **figure 7** pour voir le procédé), l'ensemble des liens dynamique (ie liens d'images changeantes) est connu. Cependant, parmis ces images nous avons les images du produits, mais aussi celle fournies par les recommandations. Il faut donc trouver un moyen de séparer ces images .

4.4.1.2 Clustering des liens

Le but ici est de regrouper les liens de recommandations entre eux et les liens concernant le produits entre eux aussi. Pour cela j'ai utilisé un algorithme de clustering : dbscan. Le clustering se fait sur la distance en nombre de caractère dans le code HTML par rapport au début du code.
Cela permet donc d'avoir en général 2 cluster distincts. En me basant sur l'ensemble des sites que j'ai scrappé auparavant, le 1er cluster est choisi comme celui contenant les images que l'on souhaite. Ainsi à la fin de cette étape on est sensé avoir l'ensemble des images du produit. Afin de vérifier si les liens sont bien ceux recherchés, on compare les liens entre eux en utilisant la distance de Levenshtein [4]. La distance de Levenshtein

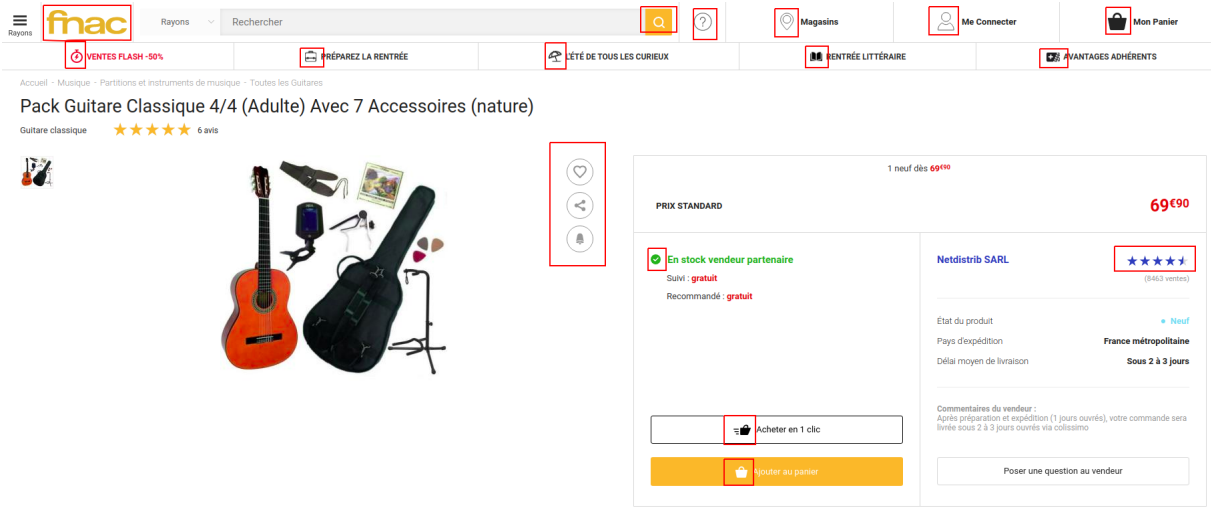

Figure 6: Exemple d'images statiques

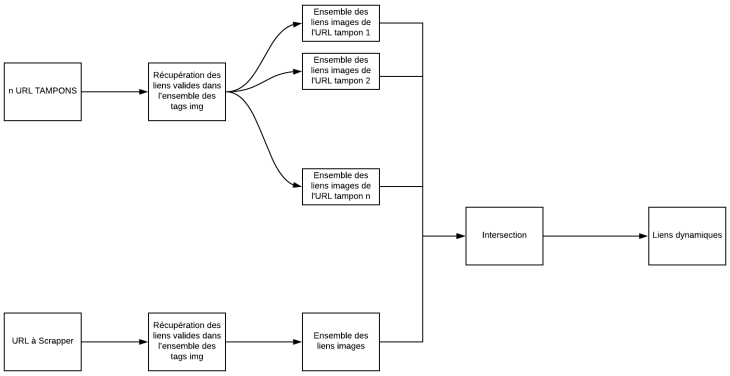

Figure 7: Etape 1

entre deux chaînes de caractère A et B est le nombre minimum de transformation (ajout/suppression/changement d'un caractère) pour passer de A à B. Ainsi, si dans le cluster, un lien paraît plus éloigné des autres, il est retiré de la liste.
Cette règle se base sur le fait que les images d'un même produits ont souvent des liens très similaires et sont différencié par un identifiant dans l'URL.

4.4.1.3 Tests et résultats

Les tests ont été réalisé sur une vingtaine de site que j'avais au préalable scrapper afin de pouvoir évaluer la différence. Il apparaît que ce scrapper générique arrive à récupérer toutes les images d'un produit spécifique. Cependant, il récupère trop d'image dans le sens où il peut récupérer 4 fois la même image mais ayant des tailles différentes. Encore une fois, pour des raisons de capacités mémoire, cette solution n'a pas été retenue par le client en l'état, mais est toujours en cours de développement.

4.4.2 Technologie utilisée

L'ensemble du travail a été réalisé en NodeJS (interpreteur Javascript). Cette technologie a été utilisé car l'ensemble de l'architecture client était en NodeJS. Il était possible de travailler avec les librairies Python permettant de scrapper et faire des requetes HTML (scrappy, beautifulsoup) car cette partie est plutôt indépendante. Cependant pour éviter l'overhead introduit par un changement de langage, j'ai préféré suivre le langage du client.
Par ailleurs, plusieurs sites internet ont des protections vis à vis du webscrapping. En effet les requêtes fait

aux différents serveurs affecte les capacités de ces derniers. Ainsi deux problèmes se sont posés:

- Ne pas lancer "involontairement" une attaque contre les serveur du site en question: pour cela on a trouvé une certaine limite temporelle avec le client
- Les adresses IP des serveurs de Cleep qui étaient déjà sur liste noire de plusieurs sites.

Le deuxième problème s'explique par une technologie de protection employé par les site internet contre le webscrapping : Datadome. Datadome detecte si une requête est faite par un utilisateur ou un robot en regardant si le javascript de la page est chargé et executé. Si ce dernier n'est pas executé, il s'agit probablement d'un bot ayant requêté uniquement le code source de la page, ce qui est exactement notre cas. Une fois detecté par Datadome, l'adresse IP qui a servi pour la requête est mise sur liste noire et il n'est plus possible de faire de requête.

La solution trouvé avec le client est de passer par un service de Proxy payant : Luminati, qui permet de changer d'adresse IP à chaque requête et donc de pouvoir utiliser les scripts de scrapping. Il est a noté que ce service facture à la requête, ainsi seul les sites ayant une protections datadome ou similaire passent par ce proxy.

Cette mission m'a permis d'apprendre ce qu'était le webscrapping et m'a donné les clés pour passer à une mission plus intéressante, la création d'un moteur de recommandation.

5 Mission : Moteur de recommandation pour Cleep (3 mois)

Cette partie est consacrée au développement d'un moteur de recommandation pour la start-up Cleep. Cette mission fait directement suite à celle de webscrapping et reste dans l'esprit de développer des fonctions nécessaires à la croissance de l'application. Lors de cette mission, j'ai profité d'une grande autonomie car aucune solutions antérieures n'existaient auparavant. J'étais cependant en contact avec Damien Meurisse, co-fondateur de Cleep, et Paul Saffers, consultant datascientist à Sia Partners, afin de critiquer et valider mon avancement

5.1 Contexte et problématique

Le but de cette mission est de développer un moteur de recommandation : à partir d'un élément et des données qui lui sont liées, le moteur doit retourner un ou plusieurs éléments similaires. Cette pratique est vastement répandue parmi les sites marchands/ sites de services (Amazon, Netflix, Pinterest ...).

Avant le début de cette mission, Cleep n'avait pas de moteur de recommandation à proprement dit, certaines ébauches de moteur avait été faites, mais rien de concret. Pour une application comme Cleep dont le principe est de rendre l'utilisateur indépendant des sites marchands et de lui offrir une alternative au shopping basique sur Internet, la création d'un moteur de recommandation devenait nécessaire.
En effet, il est possible de partager au travers de l'application les produits cleepés ainsi que les listes de cleeps. Un utilisateur peut donc voir les cleeps d'autres utilisateurs et en particulier ceux correspondants à ses centres d'intérêts. Les interactions avec les cleeps déjà disponibles sur la plateforme sont donc importantes, plus un utilisateur navigue sur Cleep, plus il passe de temps sur l'application ce qui le fidélise. Cependant sans moteur de recommandation, la navigation et la découverte de nouveaux cleeps est assez difficile pour l'utilisateur. Celle-ci passe soit par la bar de recherche pour la recherche de cleep(s) spécifique(s), soit par les pages catégories (triant les cleeps des utilisateurs par catégorie).
C'est donc dans l'optique de créer un moteur de recommandation fonctionnel et adapté à l'architecture déjà présente de Cleep que la mission m'a été confiée.

5.1.0.1 Objectif(s)

L'objectif premier de cette mission qui a durée 2.5 mois était de créer un moteur de recommandation à partir des données de Cleep. Cette mission s'est divisée en deux grandes étapes:

- Etudes des différents moteur de recommandations existant et études des bases de données en graphe
- Implémentation sur la technologie choisie d'un moteur de recommandation pour Cleep

La deuxième étape inclue l'ensemble des discussions avec le client au sujet des caractéristiques du moteur de recommandation, comment il est sensé se comporter, les améliorations possibles par rapport au cas de Cleep, ainsi que les manières de tester et valider la solution apportée.
Dans le cas précis de Cleep, un moteur de recommandation consiste à fournir à partir d'un cleep un certains nombre de cleep pouvant être utile à l'utilisateur (cf schéma **figure 8**).

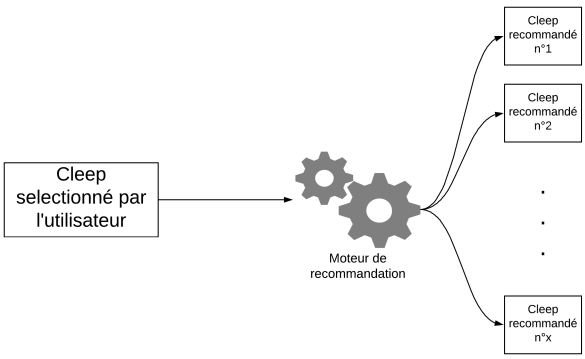

Figure 8: Etape 1

La boite noire qu'est pour l'instant le moteur de recommandation reposera sur l'ensemble des éléments présents dans la base de donnée de Cleep afin d'avoir des recommandations précises et pertinentes. En

effet, plus l'analyse des données et de leurs relations est poussée, plus le moteur retournera des cleeps pertinents. Tout du moins pertinent par rapport aux tendances évoquées par les données.

5.1.0.2 État initial

Avant le début de ma mission, il y avait déjà eu quelques idée pour la création et le développement d'un moteur de recommandation.
Cleep est une application qui repose beaucoup sur le visuel, et donc sur les images des produits. Cette image permet à l'utilisateur de reconnaître la nature du produit et donc de savoir ce qu'il a cleepé. Ainsi une idée envisagée était de baser le moteur de recommandation uniquement sur l'analyse des images. Pour cela il existe une API developpé par Google qui permet de fournir des "tags images" ainsi qu'une pourcentage de confiance associé. Dans la **figure 9**, on peut voir l'application de l'algorithme derrière cette API sur un pull.

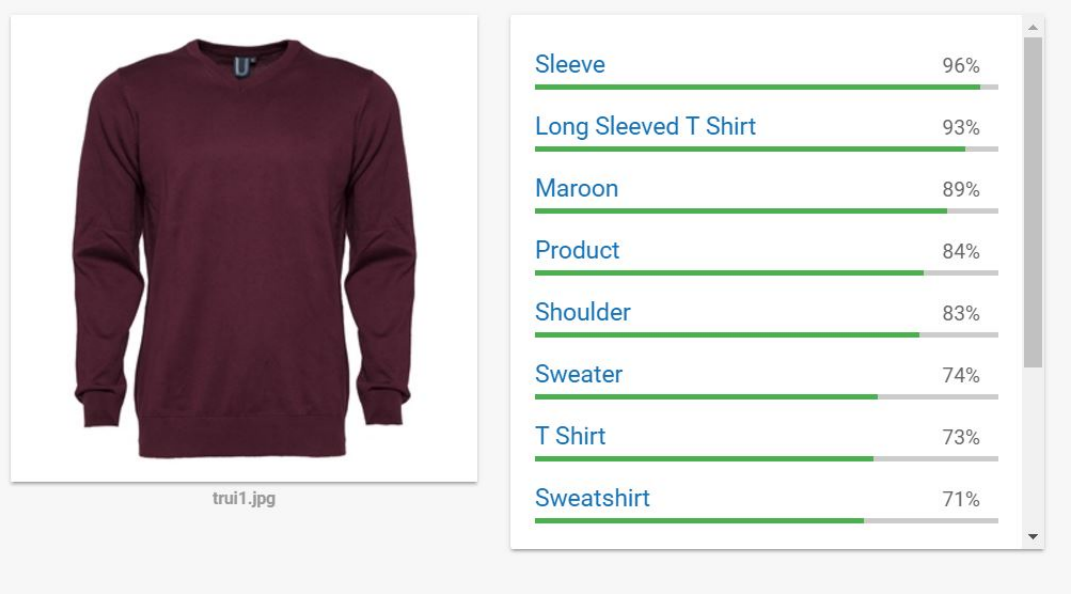

Figure 9: Etape 1

Dans cette ébauche de moteur de recommandation, l'idée était donc de tagger toutes les images de la base de donnée en premier lieu. Ensuite, pour un cleep dont on veut les recommandations, on fait passer ses images par le même algorithme, et on récupère ses tags. La dernière étape du processus envisagé est de retourner les images ayant le plus de tags en commun avec ceux du cleep original, tout en prenant en compte le pourcentage de confiance.
Cette idée, bien que fonctionnelle, ne prend en compte qu'une infime partie de la base de donnée de Cleep (ie les images). Ainsi, il est très peu personnalisé et retourne des recommandations génériques. Le but premier de cette mission est donc d'utiliser au mieux l'ensemble des informations afin de fournir des recommandations personnalisées.

5.1.0.3 Écosystème de Cleep

Il est nécessaire pour comprendre le travail que j'ai effectué de connaître l'ensemble des éléments à disposition dans la base de donnée Cleep. Je reviendrais plus en profondeur par la suite, en détaillant précisément le type de base ainsi que la structure choisie pour faire fonctionner ce moteur de recommandation.

La base de donnée est composé de:

- Informations de l'utilisateur anonymisées : son âge, son sexe, ses interêts ses interactions avec les sites internet marchands etc.
- Informations sur les cleeps : la liste à laquelle il est associé, ses tags provenant de google vision, sa date de créations, l'utilisateur qui l'a créé, et son statut (s'il est toujours en ligne ou s'il a été supprimé par son propriétaire)
- Information sur les listes de cleeps: son statut, son propriétaire et l'ensemble des personnes ayant accédé/partagé cette liste

Par ailleurs, les activités des utilisateurs sont enregistrées, ce qui donne des informations en plus, notamment sur les interactions entre les trois entités : cleeps, utilisateurs, listes. Sont a disposition les activités sur les relations suivantes:

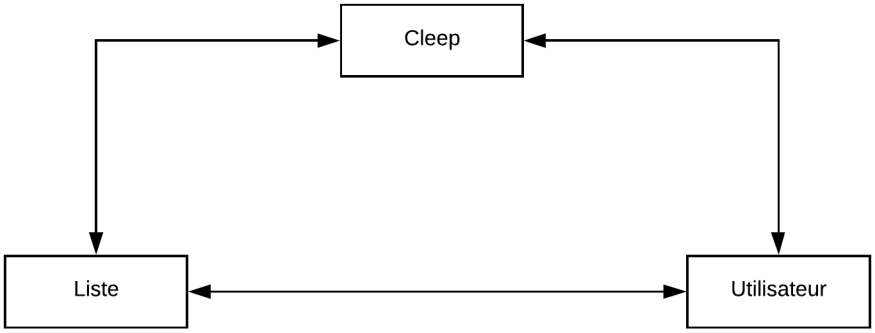

Figure 10: Etape 1

5.2 Travail effectué

Dans cette partie je vais décrire le travail effecuté pour la création du moteur de recommandation et revenir sur les principales étapes de ce processus. Comme dit précédemment, cette mission s'est d'abord axée sur l'étude des système de recommandations déjà existant ainsi que les technologies qui leurs sont associées. Ensuite, il a fallu designer et implémenter le moteur de recommandation en se basant sur les conclusions de la première partie.

5.2.1 Étude des moteur de recommandations similaires

Le cas de Cleep n'est pas du tout unique, il y a plusieurs exemples d'application similaires sur le marchés utilisant des moteurs de recommandations. On peut penser notamment à l'ensemble des sites marchands que j'ai scrappés en première partie. Chacun d'entre eux possède un système leur permettant de recommander des produits à partir d'un article.
Il y a donc autant de manière de faire un système de recommandation qu'il y a de site en proposant. Cependant il y a des objectifs clairs vis à vis de ce dernier :

- A partir d'un articles, plusieurs articles doivent en ressortir
- La recommandations doit être personnalisée
- La recommandation doit être rapide
- La mise en place du système ne doit pas être chère

C'est donc avec ces objectifs que j'ai commencé les recherches.

5.2.2 Article de recherche : moteur de recommandation de Pinterest

Après quelques recherches, ma culture personnelle ainsi que sur les conseils de Damien et Paul j'ai orienté mes recherches sur les moteurs de recommandations ayant une solution basé sur le parcours de graphe. Damien m'a alors recommandé la lecture de l'article de recherche sur le moteur de recommandation de Pinterest : Pixie [5].
Pinterest [6], est un réseau social, dont le principe est de partager de l'information via des photos, vidéos, GIFs. Chaque élément partagé peut avoir une description un titre et des tags au choix de l'utilisateur. Ce réseau est en général utilisé pour partager autour de certains centres d'intérêts comme par exemple la photographie, le sport, la cuisine etc. Le moteur de recommandation de Pinterest est donc une bonne source d'inspiration pour Cleep. Le parallèle se faisant sur le côté très graphique de l'application. Ce sont les images qui attirent l'œil avant tout, mais aussi sur le but du moteur qui est d'identifier les pins (entité regroupant l'ensemble des informations liés aux images sur Pinterest)/cleeps répondant au mieux à l'envie de l'utilisateur. Ainsi, pour les deux application ce n'est pas nécessairement de pousser à l'achat, mais de faire rester sur l'application qui est important.

Pixie est un moteur de recommandation basé sur le parcours d'un graphe en marche aléatoire. Le graphe créé pour Pixie se base sur l'ensemble des entités présentes dans l'écosystème Pinterest. Pour résumer, les noeuds du graphe sont les pins et les utilisateurs. Les arrêtes sont les relations : pins $\leftrightarrow$ pins, pins $\leftrightarrow$ utilisateurs, utilisateurs $\leftrightarrow$ pins. Chacune des entités (arrêtes et nœuds) contiennent de l'information.
Le graphe établit, il est alors possible d'établir une distance entre les pins. Cette distance est très importante car c'est elle qui va déterminer la qualité du moteur de recommandation. Le calcul de cette distance est spécifique à Pinterest, je ne vais donc pas la décrire. Cependant il est à noter que Pinterest utilise la quasi-totalité de l'information qu'il possède afin de calculer cette distance.
Il est possible de penser que le travail est fait et qu'il suffit uniquement de récupérer pour un pin les pins les plus proche de ce dernier sur le graphe. Cependant, comme le titre du papier de recherche l'indique, il y a plus de 3 milliards de pins et plus de 200 millions d'utilisateurs. Ainsi, pour chaque nœud il faudrait calculer la distance avec l'ensemble des autres nœuds, ce qui est très long voir impossible. Mais aussi il faudrait stocker cette donnée ce qui est aussi compliqué vu la taille des données. En effet, en supposant que le résultat soit un entier codé sur un 4 octets mais aussi que la distance pour aller d'un pin A à un pin B soit indépendant du chemin, il faudrait pour un nœud 12 gigaoctets. Soit pour l'ensemble de la base de donnée 36 exaoctets ($36.10^9\ Go$). De plus, à chaque ajout de nœuds, il faudrait tout recalculer, car les relations peuvent changer.
Les problèmes de stockage, de calculs mais aussi de mise à jour en temps réel rendent la solution évoquée précédemment irréalisable. De plus vu que les arrêtes du graphe portent de l'information, la distance dépend bien du chemin parcouru ainsi les calculs ont été (volontairement) vu à la baisse.

Le papier répond à ces problèmes avec l'heuristique suivante (exemple **11**):

- Étape 1 : Récolte des informations sur le premier Pin
- Étape 2 : N marches aléatoires sur le graphe ($N \in \mathbb{N}$)
- Étape 3 : Calcul du score (distance) pour chaque marche aléatoire avec le dernier pin disponible sur la marche.
- Étape 4 : Parmis les N scores, seul les pins recommandés ayant un socre supérieur à un seuil déterminé au préalable sont gardés
- Étape 5 : S'il n'y a pas assez de pins pour la recommandation, le processus est relancé
- Étape 6 : Les recommandations sont poussées sur l'application et montrées à l'utilisateur

A partir de maintenant, le terme score désigne la distance calculé avec les informations et le terme distance la distance nœuds à nœuds.
Cette heuristique permet donc d'outrepasser l'ensemble des problèmes liés à la mémoire car les scores ne sont jamais sauvegardées. De plus le calcul du score dépend bien du chemin parcouru : il est possible d'avoir la situation où le même pin est retourné deux fois, sans pour autant avoir le même score car la traversée du graphe était différente. Comme c'est une heuristique, le résultat fournit n'est bien évidement pas la solution optimale, mais elle s'en rapproche pour un temps réalisable.
La critique qui peut être émise vis à vis cette heuristique est le côté marche aléatoire. Il va être fréquent de tomber sur les mêmes nœuds / les nœuds les plus proches (en terme de distance nœuds à nœuds). Au vu du nombre de noeuds existant dans la base de Pinterest (et existant aussi dans la base de Cleep), il est assez

rare de tomber plusieurs fois sur le même chemin. Ensuite pour la deuxième critique, c'est le but même de l'heuristique, rester assez proche du nœud original. Plus on va loin dans la propagation dans le graphe, plus le nœud original est le voisin d'un voisin d'un voisin ... du nœud actuel. Il y a donc de base peu de relations entre les deux nœuds ce qui va en général retourner un mauvais score.

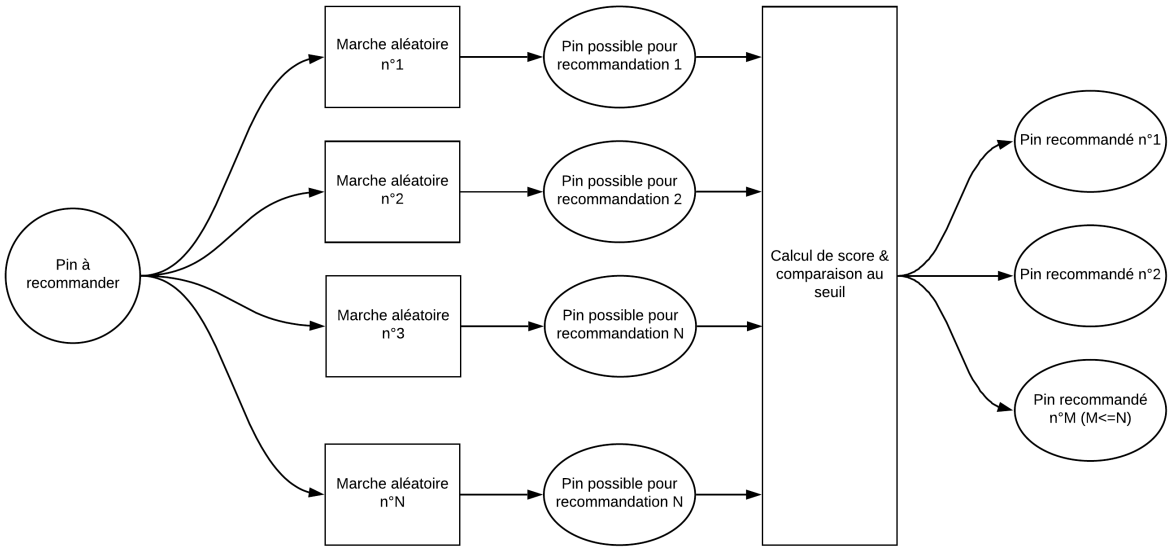

Figure 11: Heuristique choisie

C'est à partir de cette heuristique que j'ai alors implémenté le moteur de recommandation pour Cleep. Cependant, je ne pouvais pas suivre l'ensemble de papier de Pinterest car il y avait quelques limitations en matière de ressources exigées.
Pinterest crée son graphe avec une partie de sa base de donnée. Ce sont les pins étant les plus tendances, et les plus vues pendant une période donnée qui y sont inclus, mais aussi les utilisateurs ayant le plus d'interactions avec les autres/ les pins. Le graphe est créé en C++ de manière dynamique. Ainsi le graphe est stocké en mémoire RAM des serveurs, l'heuristique lui est ensuite appliquée. Le graphe est régénérée fréquemment afin d'avoir des recommandations à jour. Ce mode de fonctionnement est un problème car les serveurs de Pinterest coûtent assez cher, ce qui leur permet de faire tenir en RAM des graphes ayant une taille avoisinant le Téraoctet. Cleep ne peut raisonner de la même manière, créer un graphe compilé est donc compliqué.

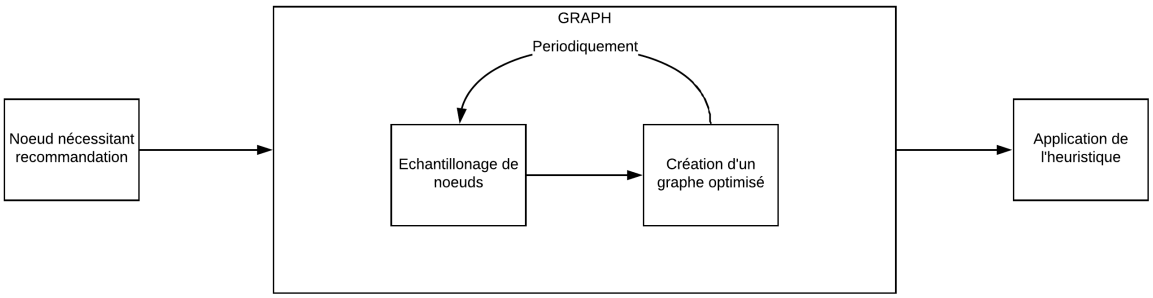

Figure 12: Fonctionnement (simplifié) de Pixie

Par ailleurs la conception d'un graphe avec requêtes optimisés prend beaucoup de temps. Cleep avait besoin d'un moteur de recommandation assez rapidement. Pour ces raisons, il a été choisi de ne pas utiliser un graphe précompilé. C'est ainsi que je me suis tourné vers une solution plus viable : les bases de données en graphe.

5.2.2.1 Interêt des bases de données en graphe

Les bases de données en graphes, sont une type de base de donnée noSQL dont l'organisation de la donnée repose sur la structure en graphe. Pour rappel, une base de donnée noSQL se caractérise par le fait qu'elle ne se conforme pas aux requêtes SQL, plus particulièrement elle n'est pas conforme au modèle RDBMS (Relational Database Management System).
Les définitions suivantes sont spécifiques aux base de données en graphe et peuvent avoir un sens légèrement différent par rapport à l'usage classique en théorie des graphes:

- Nœud : Entité du graphe contenant de la donnée
- Arrête : Entité du graphe reliant deux nœuds pouvant contenir de la donnée
- Collection: Ensemble de nœuds ou arrêtes du même type
- Document (plus rare): Informations contenu dans un nœuds (équivalent de document en base MongoDB)

La donnée dans une base noSQL ne suit pas un schéma particulier et peut évoluer rapidement. Ce premier point est très intéressant pour notre cas, car Cleep étant une start-up avec un produit naissant, beaucoup de données sont encore à acquérir pour cerner au mieux les interactions sur la plateforme. Dans le cas d'une base de donnée en graphe, la donnée est souvent stockée au format JSON.
En terme de performances, les bases de données graphe repose sur l'index-free adjacency. Dans les bases de données traditionnelles, la recherche d'éléments se fait par recherche d'index. Afin de trouver un élément précis dans une table il faut parcourir l'ensemble des indexes jusqu'à trouver le bon. Cela fait une complexité en $O(n)$ où n est le nombre de donnée dans la table.
Dans une base de donnée en graphe, les nœuds n'ont en général pas d'index prédéfini par la bases. Cependant ils ont tout de même une ID qui leur est propre et l'ID de leurs nœuds voisins. Ainsi pour rechercher 1 nœud en particulier dans la base, une base en graphe va agir comme une RDBMS, et faire une comparaison d'ID. Mais ensuite à partir de ce nœud, trouver l'ensemble des nœuds voisins, ou parcourir le graphe selon un certain modèle est en complexité $0(1)$ car l'information est déjà dans le nœud. La **Figure 13** permet de comprendre ce que cela implique comme changement. Cette caractéristique est l'un des grands avantages de ces bases de données, car cela accélère grandement les requêtes. Dans le cas de Cleep, c'est d'autant plus intéressant que nous cherchons à faire une marche aléatoire sur le graphe. Ce type de base est donc bien adapté au problème.
Par ailleurs, les bases de données en graphes sont efficaces en matière de rapidité de requête car les noeuds et arrêtes concernés par la requête sont tous chargés en RAM.
Finalement, l'un des derniers problèmes que résolvent les bases de données en graphes, c'est la mise à jour et l'ajout de donné en temps réelles. Dans le modèle relationnel, l'ajout de donnée se fait souvent en batch (plusieurs données ajoutées en même temps), car bien plus rapide. Dans le modèle graphe, seul les noeuds voisins à celui ajouté vont être affecté. D'où la rapidité et la modularité de l'insertion de donnée.

Figure 13: Principe du index-free adjacency

5.2.2.2 Les désavantages

Les bases de données graphe ont plusieurs défauts qui ont la même origine : la technologie n'est pas encore assez mature:

- La plupart des bases de données ont un souci de management de la mémoire. Les très grosses requêtes ont tendance remplir la RAM, et peu de base de données disposent de "spill sur disk" (utiliser la mémoire SDD ou HDD qui possède des cycles bien plus long). Cependant la marche aléatoire chargera très peu de donnée en mémoire RAM par rapport à au seuil critique de "requête lourde".

- Le garbage collector (script dont le but est de libérer l'espace RAM qui n'est plus utilisé) de ces bases de données pose parfois soucis ce qui nous ramène souvent au premier problème
- Chaque base de donnée possède son propre langage contrairement aux RDBMS. Contrairement aux bases RDBMS qui utilisent toutes du SQL, le langage de requête n'est pas uniforme. Cela est un grand désavantage car cela force à réinvestir du temps si l'ont veut changer de base de donnée graphe.
- Les bibliothèques dans les principaux langages de programmations (python, nodejs ...) sont peu souvent mis à jour et offrent peu de modularité.

Les bases de données en graphe ont qualités et défauts. J'ai jugé que les défauts était négligeable pour le moment par rapport au gain de temps et la "facilité" de developpement qu'offre ce modèle. Par la suite, il fallait donc trouver et étudier le fonctionnement des différentes base de données graphe existantes sur le marché. Selon les recommandations du client, j'ai commencé par étudier la faisabilité du projet sur la base de donnée graphe ArangoDB. En approfondissant mes recherches, et en expérimentant les limites d'ArangoDB, je me suis tourné vers la base graphe la plus utilisée sur le marché : Neo4j.

5.2.2.3 Comparatif ArangoDB / Neo4j

J'ai réalisé plusieurs tests comparatifs entre ArangoDB et Neo4j. Je présenterai ici que les résultats. J'approfondirai cette partie dans la prochaine sous-section. ArangoDB et Neo4j sont assez différentes dans leur manière d'envisager les bases de données en graphe.
ArangoDB est une base de donnée qui reprend beaucoup d'élément de mongoDB, en organisant ses données selon des collections et des documents. Ainsi ArangoDB propose une indexation qui permet de récupérer les éléments de manière plus rapide (identifié par leur collection. Le modèle graphe n'est donc pas le seul que propose ArangoDB (le modèle key-value et document sont aussi présents). A l'inverse, Neo4j ne propose qu'un seul type de modèle, celui en graphe, mais le moteur derrière se retrouve plus optimisé que celui d'ArangoDB.

L'un des grands avantages de Neo4j est sa documentation (bien qu'encore assez sommaire) vis à vis des algorithmes développés pour la traversée de graphe [7]. Cette base de donnée possède certains algorithmes de traversée qu'il est possible d'appeler directement via son langage de requête (Cypher). En particulier, il y a une version optimisé pour neo4j de la marche aléatoire. A l'inverse, ArangoDB ne possède pas d'algorithmes de traversé inclus originellement, ce qui nous oblige à les créer par nous même. Les tests comparatifs que j'ai effectué montre justement que les performances de la marche aléatoire d'ArangoDB développé par moi même sont inférieurs par rapport à celle de Neo4j en terme de temps d'execution et de mémoire RAM consommé.
Après discussion avec le client sur la marche à suivre et l'ensemble du travail réalisé jusque là, nous nous sommes mis d'accord sur l'utilisation de l'heuristique de Pinterest avec Neo4j comme support pour le graphe.

5.2.3 Implémentation

L'implémentation de l'heuristique de Pinterest adapté au problème de Cleep a pris du temps, notamment pour des raisons techniques (manque de documentation sur l'utilisation de Neo4j, les API/wrappers dans les autres langages ne fonctionnant pas ...), mais aussi pour des raisons de conceptions. L'implémentation s'est divisée en trois étapes :

- Adaptation sur Neo4J, instanciation des serveurs, création des plu-gins adaptés et première version simple d'un moteur de recommandation
- Études des améliorations en particulier au niveau de la fonction de scoring (qui permet d'évaluer la qualité d'une recommandation)
- Tests de la première version du moteur

Avant de commencer cette partie, vous pouvez retrouver en **Annexe 9.3** de l'heuristique que j'ai implémenté par la suite.

5.2.3.1 Première version du moteur de recommandation

Tout d'abord, l'ensemble des données que possède Cleep est stocké dans une base mongoDB. Le problème est qu'il n'est pas possible de réaliser un simple dump de base. En effet, seule les informations concernant les cleeps, les utilisateurs, les listes de cleeps et les relations nous intéresse. Donc la première étape a été d'établir les données utiles qui allaient composer la base de donnée de Neo4j.

Par ailleurs, pour créer et tester un moteur de recommandation fonctionnel, je ne nécessitais la base en entière. En effet je pouvais commencer à développer en local avec une base de donnée factice afin d'expérimenter. Ainsi, avec le client nous nous sommes mis d'accord sur le format des données à utiliser (ie comment ils seront enregistrés dans la base) et j'ai commencer l'implémentation.
En premier lieu j'ai créé un script permettant de générer les données nécessaires et de commencer l'implémentation du moteur de recommandation. Le paragraphe suivant permet de comprendre ce que contient la base de donnée ce qui va être utile pour l'étude de la fonction de scoring. Le format des données est le suivant:
Noeud cleep :

- Nom de domaine du Cleep
- Devise du produit
- Prix du produit
- Tags fournis par google vision
- Date de création
- Statut du cleep : actif ou archivé par l'utilisateur

Noeud utilisateur:

- Genre de la personne
- Âge de la personne
- Nombre d'activités vis à vis d'un domaine et date de la dernière activité. Les activités peuvent être : la création d'un cleep, la vue d'un cleep, l'ajout d'un like à un cleep, la copie d'un cleep où l'action d'aller sur le site original du cleep.

Noeud Listing:

- Statut de la liste : active ou archivée par l'utilisateur

Relation utilisateur $\leftrightarrow$ cleep:

- Nombre d'activité d'un utilisateur sur le cleep relié et la date de dernière activité. Les activités peuvent être : la création d'un cleep, la vue d'un cleep, l'ajout d'un like à un cleep, la copie d'un cleep où l'action d'aller sur le site original du cleep.

Relation utilisateur $\leftrightarrow$ listing:

- Nombre d'activité d'un utilisateur sur la liste reliée et la date de dernière activité. Les activités peuvent être : la vue d'une liste ou l'action de suivre la liste.

Relation cleep $\leftrightarrow$ listing: il n'y pas de donnée particulière sur cette relation.

Pendant la première partie de l'implémentation, j'ai donc travaillé avec une base factice composé de ces éléments. Comme énoncé précédemment, Neo4j possède un algorithme de marche aléatoire qui peut être appelé directement par son langage de requête : Cypher [8]. Cette algorithme permet donc de retourner à partir d'un cleep, un chemin aléatoire. Il prend plusieurs paramètres en entré dont deux primordiaux : le nombre de marches aléatoire à faire et la profondeur de la marche aléatoire.
Le problème est que le dernier nœud d'une marche aléatoire n'est pas toujours un cleep. De plus, la requête retourne uniquement les ID des nœuds et pas le chemin en entier avec les données associés. Une autre requête est donc nécessaire pour avoir les chemins en entiers.
Cependant Neo4j offre la possibilité de développer des plu-gins Java qui peuvent être aussi appeler pendant une requête. Ainsi deux solutions étaient possibles :

- Effectuer deux requêtes pour récupérer les chemins en entiers et ensuite traiter les chemins pour avoir une score.
- Effectuer une seule requête retournant les cleeps recommandé via le développement d'un plu-gin.

J'ai choisi la deuxième solution, car cela permettait d'éliminer une requête dans le processus et ainsi réduire le temps total d'exécution.

5.2.3.2 Description du processus

Le but maintenant de savoir comment créer le plu-gin afin de pouvoir appliquer un score sur le chemin. Il existe différents type de plu-gin Java pouvant être créé sur Neo4j. Ce qui nous intéresse sont les plu-gins d'agrégation [9]. La marche aléatoire sous Neo4j fonctionne de manière séquentielle, elle ne ressort pas directement un chemin mais les nœuds un par un. La fonction de score doit donc être capable de garder en mémoire un certain nombre de paramètre et d'appliquer un score à chaque étape de la marche aléatoire. C'est pourquoi les plu-gins d'agrégation sont adaptés à ce problème.
La figure suivante (**14**) décrit le processus global pour appliquer un score et avoir une recommandation.

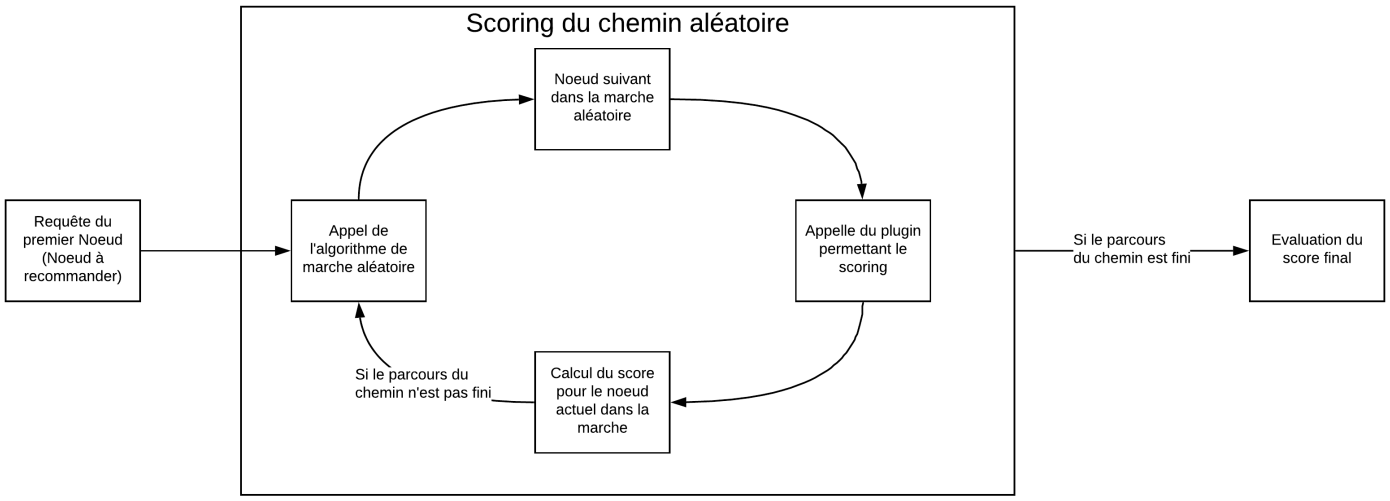

Figure 14: Processus pour appliquer un score

Le processus étant défini, il reste la fonction de scoring à décrire.

5.2.3.3 Fonction de scoring

5.2.3.3.1 1ère version

La première version de la fonction de scoring était assez sommaire, dans le sens que mon but était de montrer que j'avais mis l'architecture en place et qu'il est possible de faire un scoring complexe utilisant toutes les données.
Tout d'abord le score est normalisé entre 0 et 1. Plus le score est proche de 1, plus le cleep est à recommander, et inversement quand le score est proche de 0. Ce qui suit présente les différents calculs effectué à chaque étape de la fonction de scoring. Cette fonction prend donc en entrée, le cleep original, le nœud sur lequel vient de s'arrêter la marche aléatoire et le nœud le précédent ainsi que les relations qui en découlent. Pour rappel, la fonction de scoring est une fonction d'agrégation qui calcul un score pour chaque nœud de la marche

Pour les relations contenant des activités (ie cleep $\leftrightarrow$ user), et les activités utilisateur : un coefficient est associé. Ce dernier est plus ou moins grand en fonction de l'activité. Par ailleurs une deuxième donnée est disponibles pour les activités : le nombre d'interaction avec l'activité. Ce nombre d'interaction est normalisé avec la fonction (1) et multiplié au coefficient :

$$f(x) = \frac{atan(1/factor \times x)}{\pi/2} \qquad \text{où } x \text{ est le nombre d'interacitons} \\ factor \text{ est le facteur de normalisation} \tag{1}$$

Ainsi pour chaque activité la formule suivante est utilisée:

$$s_{act} = coefficient \times \frac{atan(1/factor \times x)}{\pi/2}$$

x est le nombre d'interacitons,
$factor$ est le facteur de normalisation
$coefficient$ le coefficient relié à l'activité

(2)

C'est cette formule que j'ai décidé de garder car elle permet de prendre en compte les différences entre les activités et d'adapter le poids de ces dernières dans le score. J'ai utilisé cette fonction de normalisation car j'avais besoin d'une injection $\mathbb{N} \to [0; 1]$ (**15**).

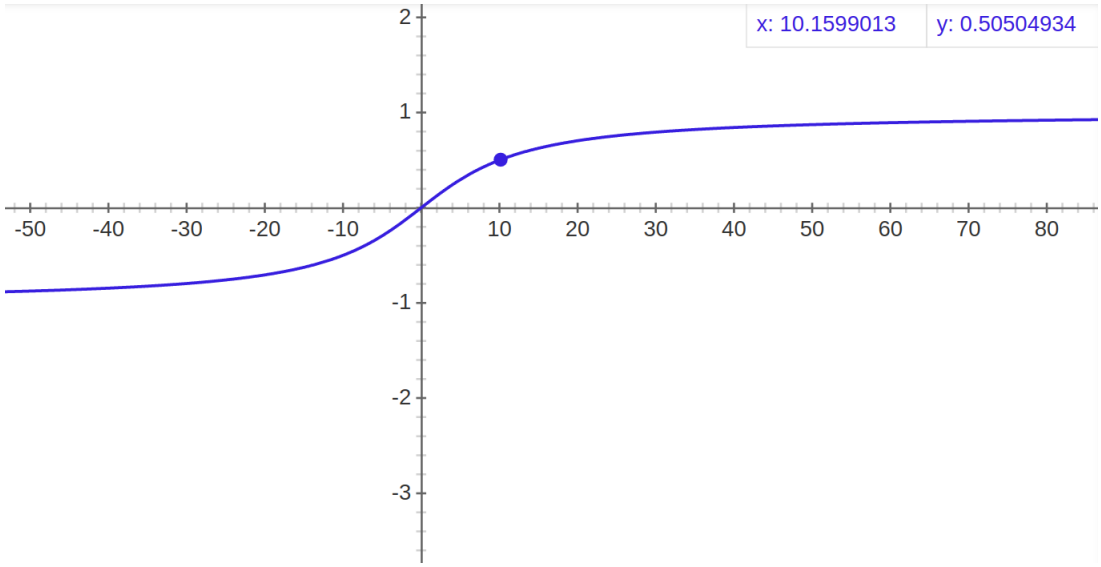

Figure 15: Fonction de normalisation

De plus il est possible de contrôler la mi-valeur de cette fonction avec $factor$. Les deux inconnus à déterminer pour chaque activités sont donc $coefficient$ et $factor$. $factor$ est une variable assez facile à déterminer, le but est de savoir à partir de quelle valeur, on considère nous sommes dans la moyenne d'interaction. Ainsi j'ai calculé via des requêtes Cypher la moyenne pour chaque activité ce qui m'a donné les valeurs de $factor$. La manière de trouver les valeurs de $coefficient$ est empirique et sera décrit dans la partie test. Ainsi pour chaque activité est retournée une valeur s_act.

Concernant les données utilisateurs, si l'utilisateur a eu des interactions avec le cleep original, alors un bonus de 10% est donné au score.

Concernant les données de cleep, c'est la différence de prix entre le cleep original et le cleep actuel qui va jouer avec cette formule (3):

$$s_{price} = coefficient \times \frac{atan(1/factor \times |price_{cur} - price_{original}|)}{\pi/2}$$

$price_{cur}$ est prix du cleep actuel,
$price_{original}$ le prix du cleep original
$factor$ est le facteur de normalisation
$coefficient$ le coefficient relié à l'activité

(3)

De plus si les deux cleeps proviennent du même domaine, alors un bonus de 10% est donné au score. Finalement les tags google vision sont pris en compte selon la formule suivante:

$$s_{vision} = coefficient \times \frac{\sum_{t \in tags} c_t}{n_t}$$

$coefficient$ le coefficient relié au google tag vision
$tags$ est l'ensemble des tags communs au cleep orignal et actuel
c_t est le taux de confiance par rapport au tag
n_t est le cardinal de $tags$ (ie le nombre de tags en commun)

(4)

Une fois ces scores intermédiaire calculés, une moyenne pondéré par les coefficient est calculée que l'on nomme s_{mean}, ce qui nous donne un résultat entre 0 et 1. Le score final pour cette étape du chemin aléatoire est alors donnée par la formule suivante :

$$score = \frac{s_{mean} + score}{2} \qquad score \text{ est le score calculé à l'étape précédente} \tag{5}$$

Cette première version de la fonction de scoring a permis de fournir une première version fonctionnelle du moteur de recommandation. Plusieurs améliorations ont par la suite été apporté, cependant la base restera la même, notamment en matière de calcul des scores intermédiaires.

5.2.3.3.2 2ème version

La deuxième version moteur de recommandation inclut un troisième type de noeud : les listes de cleep. Jusqu'à présent je travaillais uniquement avec les nœuds cleep et utilisateurs afin de produire rapidement un moteur de recommandation. Pareillement, les activité liés aux listes seront calculé de la même manière. De plus si le cleep orignal appartient à la liste étudiée et si la liste étudiée n'appartient pas à l'utilisateur qui demande la recommandation, alors un bonus de 10% est donné au score. De même si le cleep original et un cleep étudié lors de la marche aléatoire sont dans la même liste, un bonus de 10% est donné au score. Ces bonus sont attribué, car les listes créées par les utilisateurs sont sensées regrouper des cleeps de même catégorie, donc des cleeps qui ont de forte chance d'être recommandable.

J'ai intentionnellement homis plusieurs détails dans le développement du moteur de recommandation afin de ne pas alourdir la lecture du rapport. Notamment les erreurs de conceptions vis à vis du scoring, mais aussi vis à vis du fonctionnement de la base de donnée.

5.2.3.4 Tests du moteur

Le moteur de recommandation est passé par plusieurs phase de tests. Tout d'abord la phase préliminaire de tests entre le client et moi même. Pour chaque modification effectué sur le moteur de recommandation, nous testions le moteur de recommandation avec des tests unitaire développé pour. Ces derniers testait le fonctionnement attendu selon les différents cas possible. Par exemple, un chemin $cleep \leftrightarrow cleep$ n'est pas possible et donc doit retourner une erreur. Un chemin $cleep \leftrightarrow utilisateur \leftrightarrow cleep$ doit ou les deux cleeps sont le cleep original doit retourner une "NaN" value, vu que le chemin ne donne aucune recommandation possible. L'ensemble de ces tests permettent de vérifier si le moteur de recommandation fonctionne comme il a été pensé.
Au sujet des coefficients, leur valeurs ont d'abord été évalué par moi en essayant de ne pas être aberrant. Par exemple le coefficient lié à la vue d'un cleep est bien moins grand que le coefficient lié au partage d'un cleep vu que l'action de partager est plus rare et engage plus l'utilisateur. Par la suite afin de raffiner la valeur de ces coefficients, une réunion interne à Sia Partners et Cleep a été organisée.
Finalement une phase d'A&B testing a été lancé pour voir si le moteur de recommandation est efficace. Le principe est de proposer à une base d'utilisateur l'application avec le moteur de recommandation et une autre base d'utilisateur l'application sans moteur de recommandation (ie avec des recommandations proposé au hasard). Si les utilisateurs avec le moteur de recommandation suivent plus les recommandations, alors le moteur est efficace sinon il ne l'est pas. Durant le développement du moteur, j'ai aussi intégré une interface de log, permettant de comprendre pourquoi certaines recommandations ne marchent pas et pourquoi d'autres fonctionnent. Je ne suis pas resté assez longtemps pour avoir les résultats de l'A&B testing, cependant à la suite de ce dernier, les coefficients auraient une nouvelle fois changé pour être plus adapté aux actions de la population de Cleep.

5.3 Vision critique et apport personnel

Dans cette partie, je reviens sur quelques éléments apparus au cours du développement du moteur de recommandation qui demandent un peu plus d'approfondissement.

5.3.0.1 Tests sur ArangoDB et Neo4j

Afin de comparer les deux bases de donnée, j'ai effectué plusieurs tests comparatifs sur les bases de données notamment en terme de temps d'exécution d'une marche aléatoire. L'étude a été réalisé en local, et donc avec une RAM limité.

Ainsi les tests ont été réalisé sur une base de donnée contenant 500 utilisateurs, 10.000 cleeps et 1.4 millions de connexions. J'ai délibérément ignorer les listes pour rendre les tests plus léger et moins compliquer à mettre en place. En effet pour avoir une idée du temps d'exécution et du temps de requête assez proche de ce qu'on allait avoir en production, des bases avec des données factices ont été générée. Le but étant d'avoir des nœuds et des arrêtes contenant des données comme c'est le cas pour Cleep.
L'étude étant réalisée sur la marche aléatoire, il y a trois paramètres sur chaque graphe:

- Number walker : Nombre de marche aléatoire faites
- Depth : La profondeur de la marche aléatoire (ie le nombre d'étape faite pendant la marche)
- Time : Le temps médian d'execution. Pour chaque couple (number walker, depth) 100 réalisations ont été effectuées et la médiane est prise comme référence

Les figures suivantes montrent la comparaison entre les deux solutions. Des graphes supplémentaires sont disponibles en **Annexes 9.2**.

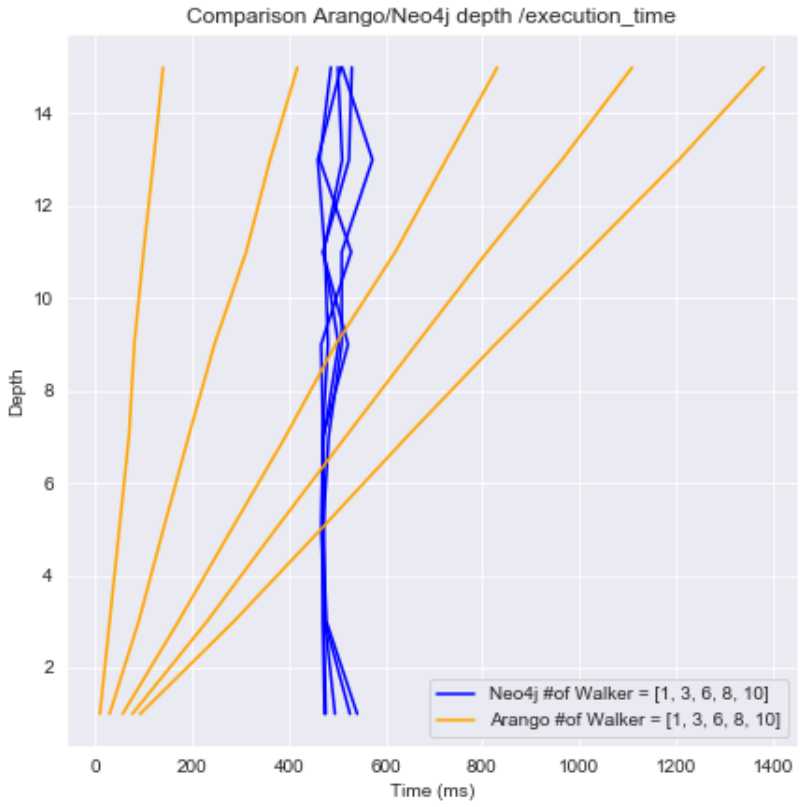

Figure 16: Comparaison des bases en fonction de la profondeur et du temps médian

Les figures suivantes montrent la comparaison entre les deux solutions. Des graphes supplémentaires sont disponibles en **Annexes 9.2**. Ces graphes représentent les même résultat mais de manière séparé (et un peu plus lisible) pour chaque base de donnée. De ces deux graphes (**figure ??fig:depthtime]** & 17), deux conclusions peuvent être tirées:

- Le temps d'exécution pour arangoDB de la marche aléatoire est plus lente dès qu'on augmente le nombre de walkers ou la profondeur.
- Le temps d'exécution pour Neo4j de la marche aléatoire semble être constant. La moyenne de 500 ms qu'on peut voir sur tout les graphes (ceux en Annexes inclus) est peut être biaisé par la puissance de la machine sur laquelle j'ai effectué le test. Il est possible que sur un serveur le temps de traitement soit moins long. Cela indiquerait donc que Neo4j est plus gourmand en ressources.

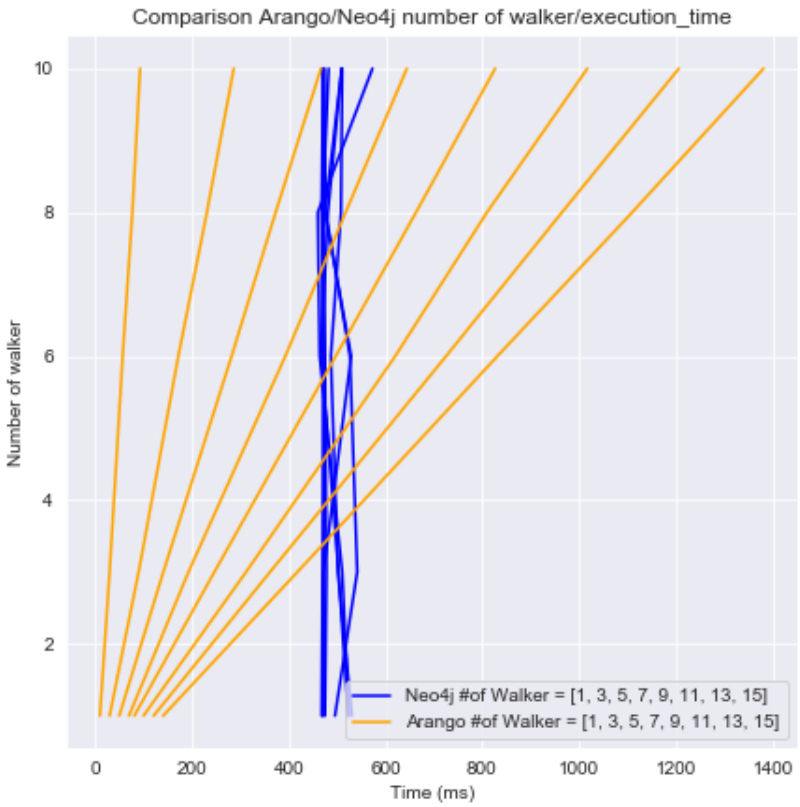

Figure 17: Comparaison des bases en fonction du nombres de chemin aléatoire et du temps médian

Par ailleurs ces tests ne prennent pas en compte la latence dû à la connexion au serveur. Ici le travail a été fait en local, cette latence est nulle. Cependant, dans le cas de Neo4j, l'ensemble de la marche aléatoire est effectuée en une requête. Dans le cas d'ArangoDB, le nombre de requête est de l'ordre de $O(depth \times numberofwalker)$, ce qui peut être problématique en terme de latence.
Cette différence s'explique par le fait que j'ai du créer moi même l'algorithme de marche aléatoire pour ArangoDB. Avec ce dernier, le passage au noeud suivant dans la marche aléatoire implique à chaque fois une nouvelle requête. J'ai été en contact avec les développeurs travaillant sur ArangoDB. Ces derniers m'ont confirmé que l'algorithme que j'avais développé était la seule manière de faire pour le moment car ArangoDB gère très mal la sélection aléatoire de nœuds.

Sachant que dans le cas de Cleep, le nombre de marche faite pour recommander un nœud sera important (vu que nous faisons des marches aléatoires jusqu'à l'obtention d'un assez grand nombre de cleep recommandés). De même la profondeur est un paramètre qui peut aussi devenir grand en fonction de la qualité des cleeps que ressort le moteur de recommandation. Si les cleep proches sont de "bonnes qualité", alors il n'est pas nécessaire d'aller en profondeur, cependant si cela n'est pas le cas, le paramètre peut augmenter. Pour toute ces raisons Neo4j a été choisi par rapport à ArangoDB.

5.3.0.2 Amélioration de la marche aléatoire de Neo4j

Contrairement à ArangoDB, la marche de Neo4j est efficace en terme de temps d'exécution. Ce dernier provient d'une procédure développé par l'équipe de Neo4j, dont le code est disponible en open-source.
En réalisant les tests sur les dernières version du moteur de recommandation, je me suis rendu compte que la marche aléatoire suivait bien un mouvement Brownien (ce qui en soit est le principe de la marche aléatoire) car le moteur de recommandation retournait souvent les mêmes cleeps. Et en regardant les chemins dans l'interface de debug, j'ai remarqué que les cleeps recommandés étaient les cleeps voisins du cleep original. Cela pose problème car nous n'avons plus la diversité dans nos recommandations car nous sommes limités à une échantillon des possibles.
Afin de résoudre ce problème, j'ai modifier la procédure qu'utilisait Neo4j pour faire la marche aléatoire. J'ai fait en sorte que cette marche pseudo-aléatoire ait plus de chance de s'éloigner du cleep d'origine, que de s'en approcher. Les tests après implémentations de cette nouvelle caractéristique montrent bien l'éloignement des marches aléatoires, et il est plus rare de retomber plusieurs fois sur le même cleep.

5.3.0.3 Optimisation du calcul de score

Le calcul de score dépend grandement du chemin parcouru. En effet, selon le chemin, les cleeps, listes et utilisateurs qui le composent peuvent être différents et donc fournir des résultats différents. Cependant avec la marche aléatoire il arrive souvent de faire une boucle (**firgure 18**). Le problème dans cet exemple est que la boucle va impacter le score (que ce soit négativement ou positivement). Le score du chemin 1 $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ 4 impacte le score du chemin 5 $\rightarrow$ 6. Ce qui nous intéresse c'est le chemin du cleep #1 au cleep #3 sans revenir sur son chemin car ca n'a pas grand interêt pour la recommandation.
En effet, avec une boucle il est possible de recommander n'importe quel cleep, en faisant une boucle qui ait un impact très positif. Par ailleurs, on s'éloigne aussi de la notion de proximité et de distance dans le graphe : dans l'exemple, le cleep #2 a autant d'influence sur le score que le cleep #3.
La solution adaptée est la suivante:

- L'état des variables nécessaire au fonctionnement de l'algorithme est enregistré à chaque étape
- Au début de la boucle un état E_1 est enregistré
- A la fin de le boucle cet l'état E_1 devient l'état actuel de l'algorithme

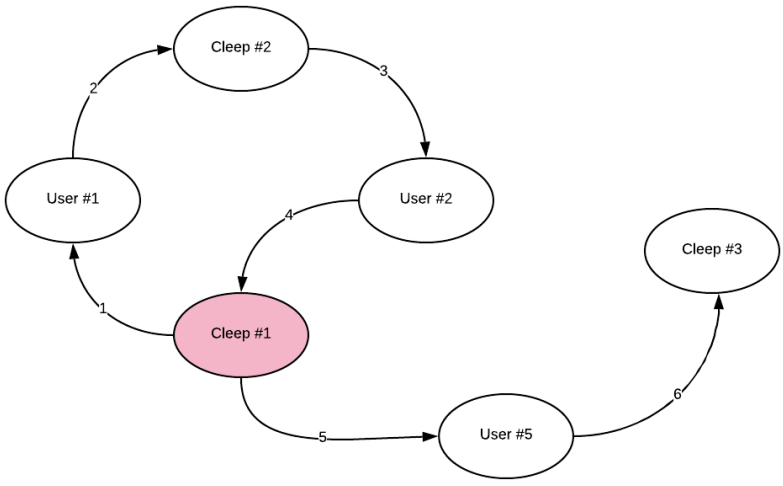

Figure 18: Boucle chemin aléatoire

Ainsi les boucles n'ont pas d'impact sur la procédure adapté. Plusieurs cas particuliers de relations se sont aussi imposés comme des boucles (la **figure 19** explicite le terme de boucle conditionné par une relation):

- cleep → user : où l'utilisateur est le propriétaire du cleep. A l'inverse la relation user → cleep où l'utilisateur est le propriétaire du cleep n'est pas considéré comme une boucle. Dans le premier cas on considère que le chemin aléatoire n'avance pas car il retourne vers le propriétaire d'un cleep et le but du moteur est de recommander des cleeps et non des utilisateurs. C'est pourquoi le second cas n'est pas considéré comme une boucle.
- cleep → liste : où la liste possède le cleep. (même justification que pour cleep → user)
- User → liste et liste → User

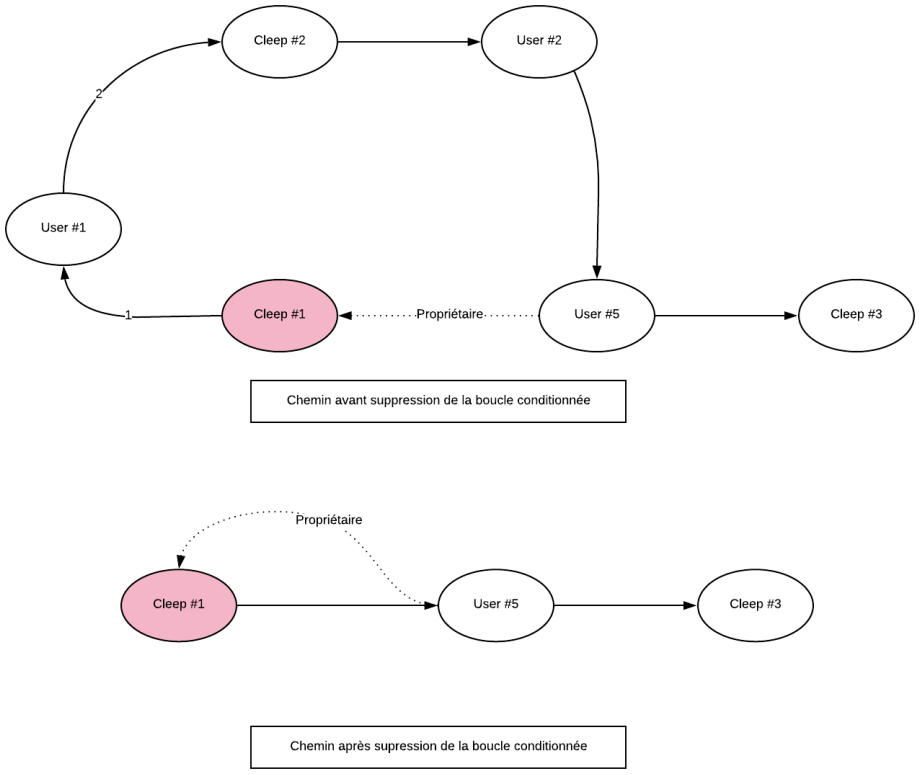

Figure 19: Boucle conditionné par la relation cleep ↔ user

La suppression des boucles est une petite optimisation par rapport au reste du travail qui a été fait. Cependant, cela reste dans le but d'avoir des chemins qui avancent en profondeur dans le graphe s'éloignant un maximum du point d'origine. Cela permet aussi d'avoir une certaine uniformité, évitant plusieurs chemins donnant des scores très différents.
Ce dernier point est l'une des forces, mais aussi faiblesse de ce moteur de recommandation. C'est la force principale dans le sens où l'utilisation des données intermédiaires permet de comprendre la connexion entre les cleeps. C'est une faiblesse pour les chemins très longs revenant proche du cleep original, l'information sur ce chemin n'est plus assez pertinente par rapport à la proximité.

Cette mission m'a permis de (re)découvrir les différents types de base de donnée et savoir quelles sont leurs avantages et leurs inconvénients. J'ai pu aussi me documenter sur les différents types de moteur de recommandation et voir lesquels pouvaient le mieux s'adapter au cas de Cleep. Grâce à ce sujet, j'ai pu animer une formation avec Paul Saffers devant les consultants de Sia Partners sur ce sujet ce qui m'a permis d'approfondir le sujet d'autant plus.

6 Mission : Aide à l'amélioration d'une plateforme de déploiement pour Sia Partners (2 mois)

Cette mission est orientée data engineer, développeur backend / frontend et donc est moins orienté dans le traitement et l'analyse de la donnée, mais plus sur l'infrastructure permettant d'utiliser et de présenter au mieux la donnée. Pendant cette mission, la personne référente était Gabriel Creti et Mathieu Thiboust qui ont grandement participé au développement de cette plateforme.

6.1 Présentation du sialab

La plateforme sialab est composée de deux parties distinctes :

- Les bots : l'ensemble des bots developpé par l'équipe datascience en fait partie. Un bot est en général une solution technique developpé par l'équipe datascience de Sia Partners répondant à un problème spécifique. Le bot est utilisable via une interface graphique, le but étant d'avoir un produit fini et pas seulement un algorithme de machine learning. La **Figure 20** est un exemple de bot disponible via la plateforme [10].
- Les plateformes sialab : Plateformes développées pour les clients. Contrairement aux bots qui sont plus pour un usage démo, les plateformes sialabs sont pour un usage opérationnel. Elles permettent de lancer et programmer des **tâches**. Les tâches sont des scripts qui ont vocations à être lancer de manière régulières : par exemple, une tâche qui va rechercher des données de manière régulières au travers d'API publiques comme la météo ou le trafic routier. Elles permettent aussi la création de Restful API et d'interfaces personnalisées.

Figure 20: Exemple de bot

La première version des plateformes sialabs est née d'un besoin client il y a un peu plus de 2 ans. Le client voulait au départ une plateforme mettant à disposition des API à travers lesquels ils pouvaient récupérer ses données de prédiction stockées en base. Ensuite un autre ajout a été très rapidement demandé, celui de pouvoir lancer et programmer des scripts permettant d'avoir ces prédictions. Par la suite, cette plateforme est devenu un argument de vente et plusieurs autres plateformes de ce type ont été créé pour les clients dont un pour Sia Partners permettant de lancer des process internes.

Ainsi l'ensemble bots et plateformes est ce que l'on appelle le sialab.

6.2 Objectifs et problématique

Avant mon arrivé, le sialab dans sa première version posait plusieurs problèmes du fait du développement anarchiques des bots et des plateformes.

6.2.1 Bots

Les développeurs de bots à l'époque étaient obligés de travailler en local, ce qui les obligeaient à installer toutes les librairies qu'ils utilisaient pour ensuite déployer l'application sur un serveur appartenant à Sia Partners ou sur machine virtuel OVH/Amazon etc. Plusieurs problèmes découle alors de ce mode de fonctionnement:

6.2.1.1 Backups Il n'y pas de backups. Les serveurs sur lesquels étaient déployés les bots n'avaient pas de structure permettant de répliquer les applications et avoir un service qui tourne en permanence. Les serveurs étaient donc des points individuels de défaillances (single point of failures). Ainsi les application étaient copiées via une connexion SSH sur les serveurs et accessible via l'adresse : "IP du serveur (ou nom de domaine Sia Partners) " + "nom de l'application". Par exemple :

- Bot KYC Adverse News disponible sur *http://192.95.33.110/shiny/adb1w1b/*.
- Bot Reg Review disponible sur *http://regwatch.sia-partners.com/demo*

6.2.1.2 Contrôle de Version Les consultants développant en local, et déployant par leurs propres moyens les bots, utilisaient rarement le logiciel de version de contrôle décentralisé autrement connu sous le nom de git à leur disposition. Le manque d'utilisation de cet outil entraîne plusieurs autres problèmes qui pour certains projets peuvent faire perdre beaucoup de temps :

- Manque de versions fonctionnels du bot (v1, v2 etc)
- Pas d'information sur les patchs fait au bots. Donc impossible de restaurer le bot dans un état précédent si la modification ne marche pas sur le serveur.
- Pas de backup de code. Donc si le code est perdu sur le serveur et en local, le bot est perdu

Cette pratique est très importante, mais du fait des conditions de développements chaotiques, l'habitude n'est pas prise.

6.2.1.3 La stabilité Comme les bots sont éparpillés sur plusieurs serveurs, environ une trentaines différents, chaque serveur avait sa propre configuration. Ainsi pour chaque bot qui venait s'ajouter à un serveur, ajoutait ses dépendances (en terme de librairies mais aussi en termes de services). Il était donc possible de se retrouver avec 2 bots différents sur un serveur, utilisant des librairies qui entrent en conflits.
La stabilité n'est donc pas assurée, et les serveurs étaient souvent polluées par plusieurs librairies inutilisées.

6.2.1.4 L'administration Avec autant de serveurs à administrer, il y a autant problèmes possibles, que ce soit en terme de stabilité, d'installation ou tout simplement de problèmes internes aux serveurs (image à mettre à jour etc.).
Par ailleurs, il peut y avoir un long moment entre l'apparition d"un bug et la découverte de celui-ci, car aucun serveur n'avait ses logs centralisés. Ainsi, le seul moyen de "trouver" un bug était d'expérimenter le bug ou d'en avoir les retours par les utilisateurs. L'administration devient d'autant plus problématique à chaque ajout de bots. Cette tâche là était donc pénible et faisait perdre beaucoup de temps.

6.2.2 Plateforme

6.2.2.1 Contrôle de Version et Backups Comme plusieurs consultants peuvent développer sur la plateforme, le contrôle de version était utilisé. Ainsi le code en lui même de la plateforme était versionné. Ainsi contrairement aux bots, les plateformes possèdaient des backups. De plus la base de donnée reliée avait aussi des backups.
Cependant les scripts de déploiement des plateformes n'étaient présents que sur les serveurs et n'étaient pas

inclus dans le contrôle de version. Ainsi si un serveur avec une plateforme perdait ses données, le déploiement de cette dernière s'en retrouvait affecté.

6.2.2.2 Stabilité et administration Par ailleurs, comme pour les bots, les plateformes étaient aussi éparpillés sur plusieurs serveurs. Ainsi, les même problèmes étaient présents, et la même pénibilité d'administration qui en découle. Cela était d'autant plus problématique que ces plateformes étaient destiné aux clients, donc avec une exigence plus grande concernant la résolution des bugs.

6.2.3 Objectifs

Une deuxième version du sialab était en cours de création lorsque j'ai rejoint l'équipe pour participer à son developpement. L'objectif principale de cette mission est de répondre à l'ensemble des problèmes cités précédemment, en particulier avoir une interface d'administration globale permettant d'avoir connaissance rapidement des bugs sur la plateforme.

Cela implique aussi de centraliser l'ensemble des projets existants et d'avoir une méthode automatique de déploiement des bots/plateformes. Le but étant de se rapprocher au maximum du déploiement en un clique.

Pendant cette mission j'ai travaillé en méthode Agile avec l'équipe sur le projet (Gabriel Creti et Matthieu Thiboust). Chaque semaine il y avait une réunion hebdomadaire avec une liste de tâche à faire et la présentation des avancements qui ont été fait. Cela a permis d'avancer rapidement sur le projet et d'avoir une version stable en Septembre.

6.3 Travail effectué

Dans cette section je vais décrire l'ensemble des fonctionnalités nouvelles le travail qui a été fait sur la version 2 du sialab. J'ai participé à toutes les étapes du développement, que ce soit au niveau de l'infrastructure, au niveau du déploiement ou au niveau du développement (frontend et backend) de la plateforme.

6.3.1 Processus de déploiement

L'ensemble des problèmes listés auparavant reposaient sur deux éléments:

- Le code n'est pas contrôlé : les consultants n'étaient pas "forcés" à utiliser le contrôle de version (git)
- Le déploiement n'est pas normalisé : les consultants déployaient leurs bots sur les serveurs qu'ils connaissaient

C'est pourquoi l'ensemble de la chaîne de déploiement a été repensé. Cela implique des changement dans la manière de développer mais aussi dans la manière d'utiliser les outils à disposition. La **Figure 21** décrit le nouveau processus de déploiement qui se décrit en 3 parties distinctes :

- Développement local avec Docker
- Contrôle de version et build des images avec Gitlab
- Déploiement sur la plateforme Google Cloud Platform (GCP)

Le nouveau processus permet d'avoir des rôles répartis de manière équitable entre les consultants et d'alléger la charge de travail au sujet du déploiement des projets sur la plateforme. Chacune des trois parties requiert des compétences différentes, cependant en tant que consultant développant un nouveau projet, il n'est pas nécessaire d'acquérir les compétences technique niveau Git et niveau Google Cloud Plateform. En effet avec ce nouveau processus, un consultant doit savoir uniquement se servir des outils Git et GCP, mais il ne doit pas en général développer dessus. C'est à dire qu'il n'aura jamais à configurer le Gitlab et les pipeline de déploiement, juste à les utiliser.

Les prochains paragraphes vont détailler cette structure et son fonctionnement.

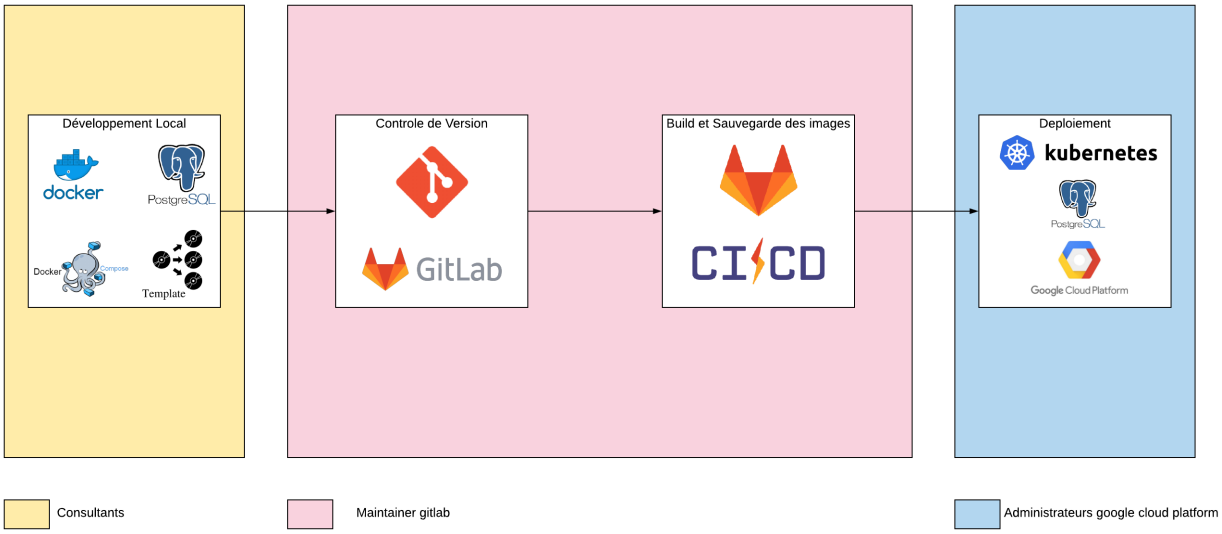

Figure 21: Processus de déploiement de la plateforme

6.3.1.1 Développement local

Afin d'uniformiser les projets et les conditions de développements, des template sont utilisés. Un template est la réplique d'un dépôt Git contenant au préalable plusieurs fichiers de configurations ainsi qu'une base de code suivant la charte graphique imposé. Le template est donc différent en fonction du projets qui est créé. Pour l'instant il y a 7 template différents :

- Web App : Application en HTML static et javascript.
- Shiny App : Application R-shiny
- Dash App : Application en python Dash
- Bokeh App : Application en python Bokeh
- Flask App : Aplication en python Flask
- ReactJS App : Application en ReactJS
- Sialab App : Template pour la création d'une plateforme sialab. Les plateformes sialab seront détaillées plus en détail par la suite.

La **figure 22** illustre les fichiers déjà présents dans le template flask. L'image de gauche est ce qu'il y a à la base du template, et l'image de droite est ce qu'il ya dans le dossier /app. Les fichiers à la racine qui ne sont pas des extensions ".md" ou ".py" sont des fichiers de configurations pour le déploiement du projet mais aussi pour le développement en local. J'y reviendrai plus en détail dans les prochaines parties. Le dossier /app contient les bases nécessaires pour développer sur le template. Dans l'exemple **figure 22** le dossier /app, contient les fichiers/dossiers important à la création d'une application Flask avec un exemple d'application simple respectant la charte graphique de Sia Partners.

La création d'un projet commence donc par la création d'un dépôt git avec le template requis sur GitLab. Un script permet d'automatiser le processus, ainsi il n'y a pas de manipulation à faire à chaque lancement de projet.

Le git créé, les consultants travaillant sur le projet doivent alors cloner le dépôts sur leur machine en local. Le nouveau processus mis en place avec les templates permet aux consultants de développer sur leur machines dans un environnement similaire à celui de la production (donc déployé sur un serveur) sans devoir procéder à l'installation de l'ensemble des packages/librairies/software nécessaires. Cela est possible grâce à Docker et Docker-compose.

Docker permet la création d'image système spécifique, et donne la possibilité de monter ces images dans des containers indépendamment du système d"exploitation sur lequel il se trouve. Un container est l'équivalent

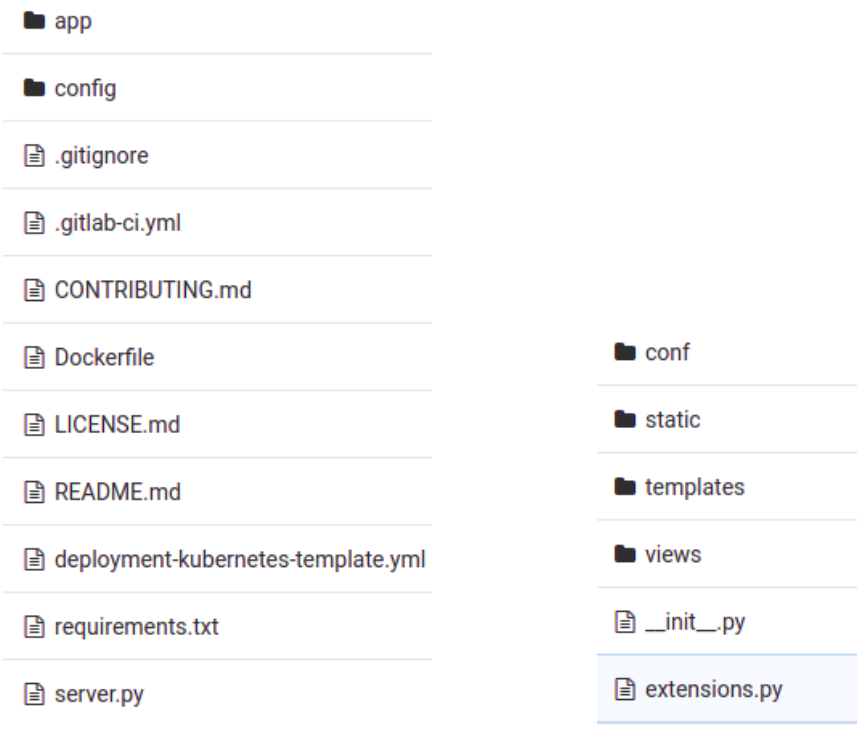

Figure 22: Fichier présents dans le template Flask

d'une machine virtuelle, et l'image est le système d'exploitation avec l'ensemble des fichiers, services ... qui sont dessus. Cette comparaison se limite à une comparaison et les spécificités techniques d'un containers Docker et d'une machine virtuelle sont bien différentes [11].

Le Dockerfile présents à la racine de chaque template est un fichier qui configure l'image docker que l'on veut créer. En général, il permet de configurer les services à lancer au démarrage, l'ensemble des packages à installer, de copier le code de l'application à lancer et finalement de lancer cette dernière. Une fois que l'image est créée, celle-ci peut être build, ce qui permet de créer un container avec l'application en train de tourner à l'intérieure et accessible en localhost selon le chemin défini au préalable (par exemple : "http://localhost/monbot). Dans le cas où l'application développée est plus complexe et nécessite plusieurs images qui communiquent entre elles (comme pour les plateformes sialab qui nécessitent une image pour le serveur frontend, une pour le serveur backend et une pour un nginx), Docker-compose est utilisé. Ce dernier build les images, créé les containers et créé les réseaux permettant de aux containers de communiquer.

Pour des raisons de sécurités, les donnée comme les identifiants permettant l'authentifications à la base de donnée ainsi que d'autres éléments sensibles sont sauvegardés dans les variables d'environnement des serveurs sur lesquels sont déployés les projets (cf **Figure 38 en Annexe 9.4**). Ainsi aucune de ces valeurs ne doivent être codés en durs, donc aucune de ces valeurs ne doivent apparaître sur les différents commit du git.
Ainsi tout consultant développant un bot ou une plateforme sialab travail uniquement avec docker sans installations externes quelconque. Cela permet d'avoir un environnement de travail propre et surtout qui ne varie pas par rapport à l'environnement de production. J'explique pourquoi dans la partie suivante.

Finalement, une fois que l'application/une fonctionnalité de l'application est développée. Le consultant pousse ses modifications sur le dépôt, et c'est ici que le travaille du consultant s'arrête. C'est cette action qui va enclencher tout le processus de déploiement qui est donc continue. La figure suivante (**figure 23**) résume les différentes étapes du développement en local.

6.3.1.2 Déploiement (gitlab)

Une fois que le code est poussé sur le dépôt du projet dans Gitlab, c'est l'intégration et le déploiement en continue qui suit. Gitlab dispose d'un module CI/CD (Continuous Integration / Continuous Deployment) qui permet de faire de l'intégration/déploiement en continue. Les fichiers de la **figure 22** qui nous intéresse

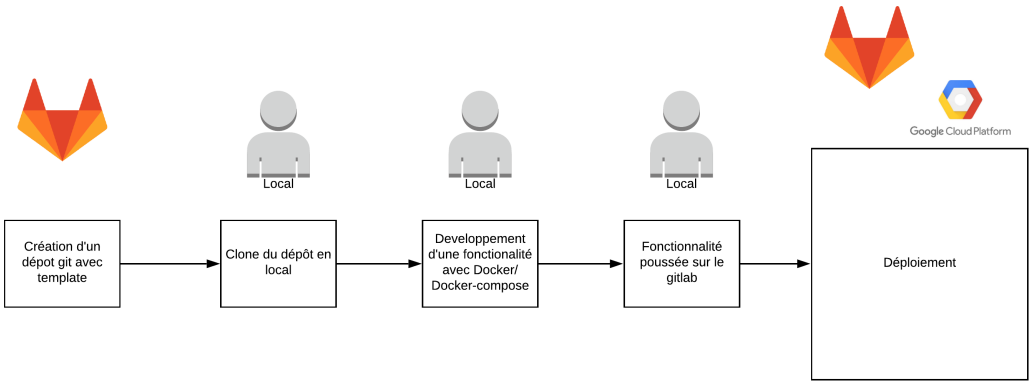

Figure 23: Résumé de la procédure

maintenant est le fichier .gitlab-ci.yml. C'est un fichier qui permet de configurer les pipelines.
Une pipeline est une suite d'actions effectuées les unes à la suite des autres (exemple **figure 25**). Dans notre cas les actions effectuées pour chaque projets sont :

- [Obligatoire] Build des images dockers : Cette étapes créé les images dockers sur le serveur Gitlab en se basant sur les Dockerfiles.
- [Optionnel] Test des images venant d'être build : Cette étape test les images créées. Ces tests peuvent concerner le linting (le formatage du code, l'import de librairie inutilisé, la création de variable inutilisé etc.), mais aussi des tests unitaires. Ainsi, si les tests unitaires échouent ou si le code n'est pas linté, la pipeline s'arrête à ce moment là. Ces tests permettent d'avoir un service qui marche en continue, mais aussi d'avoir un code "propre".
- [Obligatoire] Push des images dockers : A cette étape les images créées précédemment sont enregistré dans le registre gitlab. Ainsi ces images peuvent être téléchargées (pull) du serveur avec la bonne adresse.
- [Obligatoire] Déploiement des images sur Google Cloud Plateforme

Le fichier de configuration .gitlab-ci.yml, permet de préciser pour chaque étape ce qui doit être fait durant cette étape. Par exemple pendant l'étape du build, le fichier précise quel script doit être exécuté, en l'occurrence ici les commandes de build de l'images vont être exécuter. La **Figure 24** est un exemple de configuration pour une étape de build.

```
build_frontend:
  stage: build
  image: docker:18-git
  variables:
    DOCKER_HOST: tcp://localhost:2375/
    DOCKER_DRIVER: overlay2
    PIP_CACHE_DIR: "$CI_PROJECT_DIR/.cache/pip"
  services:
    - docker:18-dind
  cache:
    paths:
      - .cache
  before_script:
    - docker info
    - echo "$CI_REGISTRY_PASSWORD" | docker login -u "$CI_REGISTRY_USER" --password-stdin  $CI_REGISTRY
  script:
    - docker pull "$CI_REGISTRY_IMAGE"/frontend:"$CI_COMMIT_REF_NAME"-staging || true
    - docker build  --cache-from "$CI_REGISTRY_IMAGE"/frontend:"$CI_COMMIT_REF_NAME"-staging -t "$CI_REGISTRY_IMAGE"/frontend:"$CI_COMMIT_REF_NAME"-staging
    - docker push "$CI_REGISTRY_IMAGE"/frontend:"$CI_COMMIT_REF_NAME"-staging
  only:
    changes:
      - frontend/**/*
  tags:
    - ci-build
```

Figure 24: Configuration d'une étape build d'une pipeline.

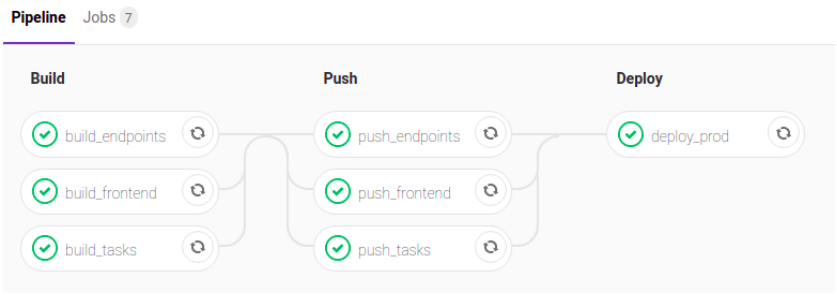

Figure 25: Exemple de pipeline

GitLab permet aussi l'enregistrement de variables d'environnements grâce à son module CI/CD. Il les intègre directement dans la pipeline, c'est comme cela qu'on retrouve les variables d'environnements dans les containers Google Cloud Plateform.

A la fin de cette étape, les images docker ont été testé et enregistré sur les serveurs Gitlab. Cependant cette étape permet uniquement cela et ne permet pas de déployer l'application sur les serveurs GCP. C'est la dernière étape de la pipeline qui permet cela.

6.3.1.3 Déploiement (Google Cloud Plateform)

Google Cloud Plateforme (GCP) est le service cloud de google qui nous permet d'héberger les applications du sialab (que ce soit bot ou plateforme). GCP propose plusieurs services, mais deux vous nous intéresser pour le déploiement :

- CloudSQL est un service GCP permettant d'héberger des bases de données SQL. C'est donc ce service qui héberge la base de donnée PostGreSQL qui alimente l'ensemble des bots et des plateforme sialab.
- Kubernetes qui agit comme orchestrateur des ressources des serveurs GCP. Kubernetes permet le déploiement automatique des containers Dockers que l'on a créé précédemment. Ces containers sont déployés dans des pods qui sont des instances de serveurs qui peuvent être configuré par l'utilisateurs. Kubernetes offre bien entendu plusieurs autres possibilité que je détaillerai juste après.

Le dernier fichier important est deployment-kubernetes-template.yml. C'est le fichier permettant la configuration des pods lors du déploiement de l'application par Kubernetes. Lors de la dernière étape de la pipeline, un script s'exécute prenant en compte cette configuration pour déployer l'application qui vient juste d'être push dans le registre Gitlab. Ce fichier détail pour chaque pods les éléments suivants:

- Le nombre de réplique de pods à avoir. Si ce nombre est à 1, il y aura une interruption de service pour ce pods lors du redéploiement de l'application.
- Les caractéristiques "réseau" de l'application. Par exemple l'adresse internet (port inclu) sur laquelle exposé l'application
- L'image docker qui va être monté sur le pod
- Certaines caractéristique interne au pod, comme son timeout après un certain temps d'activité, les certificats SSL pour assurer la connexion etc.

Le déploiement se fait donc par Kubernetes sur les serveurs GCP.

GCP permet aussi de contrôler l'activité de la plateforme en permanence. Les logs de l'ensemble des projets sont centralisés sur cette plateforme. Cela permet d'atténuer la pénibilité d'administration des serveurs, car nous sommes au courant plus facilement et surtout plus rapidement s'il y a un bug. Par ailleurs il est aussi possible de contrôler via l'interface Kubernetes de GCP l'activité des pods et la charge de travail. Ainsi, si un pods vient à manquer de mémoire, ou manque de puissance de calcul. Cette interface permet de détecter cela et de réagir en conséquence.
Ce qui est d'autant plus intéressant est que la partie orchestration des ressources est totalement laissée

à Kubernetes et GCP, ce qui annule en grande partie la charge associée auparavant à l'administration des serveurs. Par exemple, Kubernetes va par lui même associer les ressources nécessaires à un pod, il va gérer ce qui est répliqué et les backups de serveurs automatiquement. Surtout, il va relancer les pods qui ont crash automatiquement sans intervention d'un administrateur. Kubernetes et GCP offrent plusieurs autres services interessants, je ne vais pas tous les lister ici par soucis de lisibilité.

Mon travail sur cette partie déploiement a surtout été de faire, en collaboration avec Gabriel Creti, toutes les configurations nécessaire pour l'établissement de Kubernetes. Par la suite, j'ai adapté ces configuration aux projets déjà existants (bots et plateformes) mais aussi aux projets futurs. M'étant formé sur le sujet, j'ai aidé les autres consultants à prendre en main la plateforme, mais aussi à résoudre les bugs reliés à cette dernière quand ils étaient de l'ordre du déploiement (gitlab ou GCP).

6.3.2 Plateforme Sialab

Le travail que j'ai effectué pour le sialab ne se résume pas uniquement à l'établissement du processus de développement, j'ai aussi participé au développement de la plateforme sialab pour Sia Partners ainsi que l'infrastructure qu'il y a derrière l'ensemble des ces plateformes (template sialab).

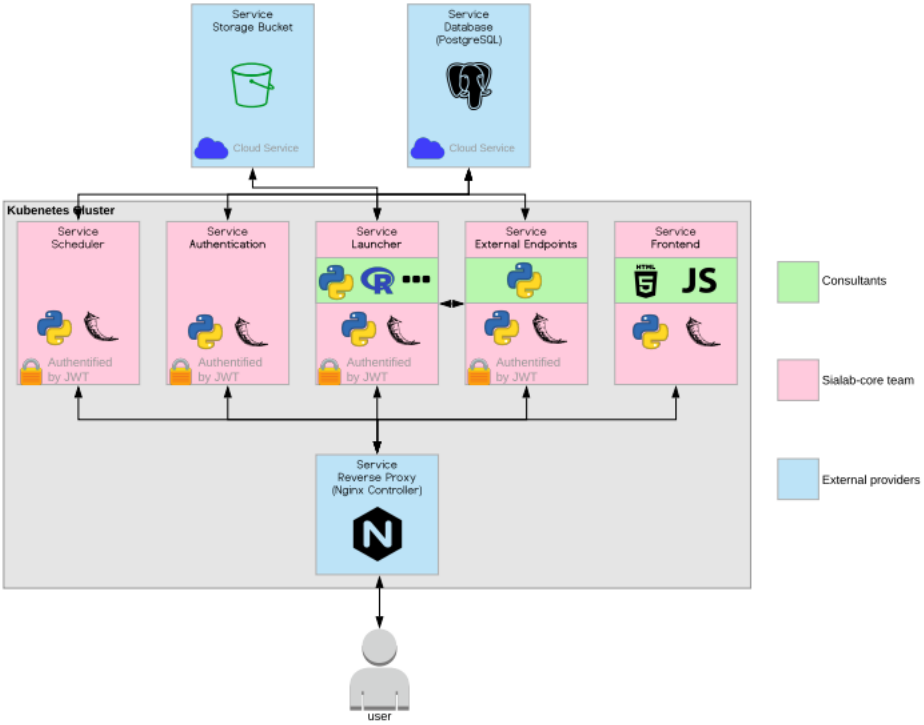

Figure 26: Structure des plateformes sialab

Les plateformes sialab, suivent le même processus de déploiement expliqué précédemment. Elles sont formées de 5 différents micro-services (**figure 26**):

- External Endpoints : API créées par les développeurs du projet. Ces API sont reliées au besoin métier qui nécessitaient la création de la plateforme.
- Authentication : Module qui gère l'authentification des utilisateurs sur la plateforme, mais aussi les droits d'accès sur cette dernière (Admin, lecture, écriture).
- Launcher : Service qui permet de lancer des scripts (Python, NodeJs, Java ...). Ces scripts sont créés par les développeurs sur le projet. De même que pour custom endpoints, l'exécution de ces scripts relèvent d'un besoin métier
- Scheduler : Ce service permet de programmer à une fréquence précise l'exécution d'un script.

- Frontend : L'interface graphique de la plateforme. Elle sert en général à avoir une interface utilisateur afin d'utiliser les API ou montrer les résultats des scripts lancés.

Les plateformes sialab sont toutes basées sur un même template, et chacun des micro-services est codé en python Flask. Le frontend possède évidement une composante en javascript, mais les serveurs en eux même sont des serveurs Flask orchestrés par Gunicorn. Chacun des micro-services est protégé par une authentification basé sur des tokens. Ces tokens sont émis par le service authentification pour chaque utilisateur et sont utilisé pour toutes les communications, que ce soit entre service interne aux plateformes sialab, ou entre l'utilisateur et la plateforme. Le choix a été fait d'avoir des communications protégés entre les différents services afin de s'assurer une sécurité maximale. Ainsi, si un des services est compromis, il ne peut pas compromettre les autres.

Comme montré sur la **figure 26**, les services sont derrières un nginx [12] qui est le seul serveur à communiquer avec Internet. Ce dernier va transmettre les requêtes qu'il reçoit aux services appropriés, ces requêtes pouvant provenir d'Internet (l'utilisateur) ou d'autres micro-services. Nginx offre plusieurs options permettant de réguler l'ensemble du trafic sur la plateforme. Finalement, on utilise deux service de cloud pour stocker les données, PostGreSQL pour l'ensemble des données stockés en RDBMS, et les buckets google pour les données lourdes comme les modèles de réseau de neurones etc.

J'ai travaillé de manière assez transverse sur ce sujet. J'ai réalisé la partie front en entière avec tout les enjeux de connexions et d'interface utilisateurs. J'ai aussi développer en partie le launcher de tâche (scripts). Gabriel Creti a développé le reste des micro-services.

6.4 Vision critique et apport personnel

Dans cette partie je vais revenir sur l'ensemble des changements majeurs qu'ont connu les plateformes sialab. En effet l'infrastructure derrière ces plateformes a bien changé en trois mois, pour arriver à une infrastructure composé de 5 micro-services.

6.4.1 Images des plateformes sialab

Dans l'idée de faciliter le développement des consultant sur les plateformes sialab, l'image docker des plateformes est divisée en deux parties:

- le sialab-core : c'est une version générique d'une plateforme sialab.
- le sialab utilisateur : la version du sialab sur laquelle développe les consultants

La **figure 27** décrit la création de l'image d'une plateforme sialab. L'image finale d'une plateforme se construit sur l'image du sialab-core. Ensuite, vient s'ajouter comme une brique l'image du sialab utilisateur.

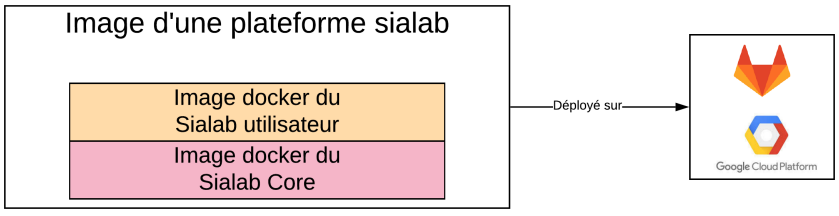

Figure 27: Construction de l'image d'une plateforme sialab

Les principales fonctionnalités des plateformes sialab sont développées dans le core, car partie commune à toute. Ainsi le scheduler, et l'authentification sont uniquement présent dans le core. Les customs endpoints quant à eux sont uniquement dans la partie utilisateurs car seul les consultants vont développer dedans. Le problème réside dans le Launcher et le frontend. En effet la partie permettant dans lancer des scripts est codée dans le sialab core, cependant les consultants doivent pouvoir coder les scripts qui vont être lancés dans cette même partie. De même, plusieurs fonctionnalités du frontend sont codés dans le core comme par exemple la gestion de l'authentification des utilisateurs, les interfaces graphiques permettant de lancer un

script etc.

La solution pour pallier à ce problème a été de créer nous même des dockerfile assez restrictifs afin de permettre cette division. Cependant le désavantage est que cela force des configurations que les consultants peuvent ne pas comprendre sans connaître l'architecture du core. Par exemple, les pages internets sont rendu en python Flask via des templates HTML-Jinja2. La fonction de copier en Docker est prioritaire sur ce qui existe déjà dans le container, ainsi un dossier est supprimé si un dossier du même nom existe. Les templates HTML-Jinja2 dans les sialab utilisateur sont à cause de cela stockés dans un dossier /custom_templates alors qu'ils sont stockés dans le dossier /templates dans le sialab-core.
Un autre exemple est l'importation des script Javascript dans les templates HTML. Comme on utilise un système de templating, on a un template master qui englobe tout les autres. Ainsi ces derniers ne font qu'étendre le premier. Le problème est que le template master et les templates du sialab-core sont les seules à charger ses scripts Javascript dans les balises <header> ou <footer>, car ils sont dans l'image du core. Ainsi les templates du silab utilisateur n'y ont pas accès, ce qui rend le chargement des pages Internet moins optimisés.

A ce jour ces problèmes existent toujours rendant les consultant développant sur ces plateformes un peu plus dépendant de la documentation fournie et de nous même, cependant la divison en deux parties reste tout de même un gain de temps et la possibilité de s'inscrire dans un projet sans avoir à connaître la globalité de son fonctionnement.

6.4.2 Changement d'infrastructure

Au départ, les plateformes sialab n'étaient pas divisés en 5 micro-services différents (**figure 26**), mais en 2 comme le montre la **figure 28**, un backend composé des éléments actuels suivant : launcher, scheduler, Authentication et External endpoints, et un frontend.

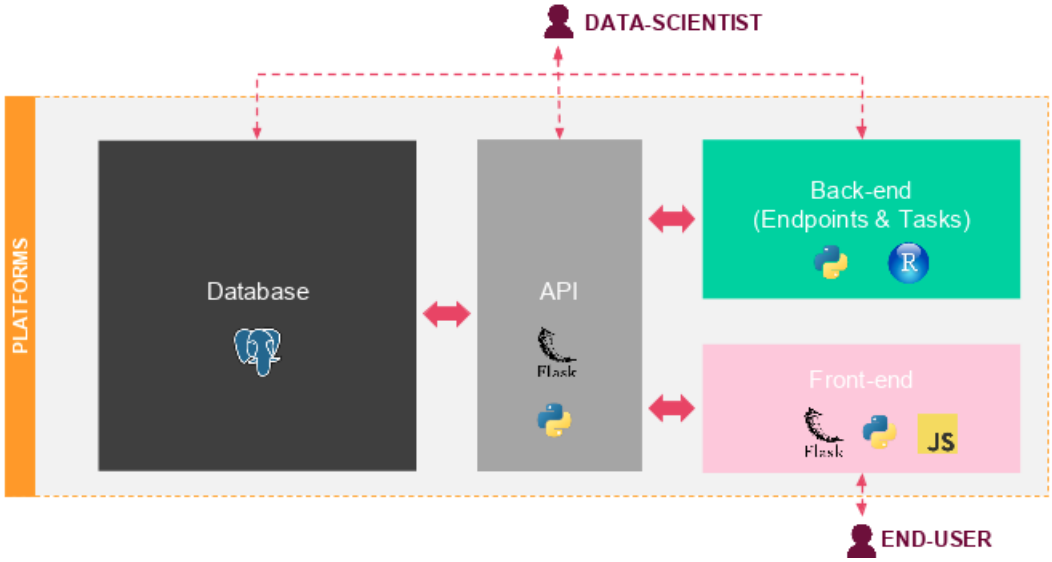

Figure 28: Ancienne structure du sialab

Cette ancienne structure ne respectait pas le paradigme micro-service ce qui a posé au cours du développement plusieurs problèmes notamment avec le scheduler. Le scheduler est donc le micro-service qui lance des tâches de manière répétée. Ce scheduler comme tout les services produit des logs qui sont enregistrés dans la base de donnée PostGreSQL, mais il va aussi récupérer la programmation des tâches dans cette même base. Par ailleurs, au niveau du déploiement, dans les configuration du fichier deployment-kubernetes-template.yml, le pods concernant le backend est configuré pour pouvoir disposé de deux répliques au même moment. Ainsi lors d'un redéploiement, il n'y a pas d'arrêt de service. Les services peuvent être dédoublés, en particulier le scheduler. Plusieurs de problèmes de concurrences survenaient lors du redéploiement à cause de cela, et les tâches étaient lancées en double.

Une solution à ce problème aurait pu être de ne pas lancer le scheduler tant qu'il y a deux répliques au niveau GCP. Cependant cela restait très sommaire comme résolution et revenait à contourner le problème

au lieu de trouver une solution réelle. C'est pourquoi courant Août nous avons changé d'infrastructure pour celle que j'ai présenté dans la section précédente. La structure en micro-service permet de configurer le pod contenant le scheduler, et lui imposer qu'une seule réplique possible

Cette structure, en plus de résoudre ce type de bug, offre plusieurs avantages, notamment un contrôle plus clair sur la provenance des bugs, vu que l'on sait à quel pod les bugs sont reliés. De plus, si un des micro-services est interrompu et rentre dans une boucle de crash (i.e. le service ne peut pas se relancer), les autres services n'en seront pas affecté et donc continueront à marcher. Ainsi, si le frontend par exemple "tombe en panne", les API resteront toujours disponibles car indépendantes du frontend.

Cette mission m'a permis de découvrir toute la partie data engineer (avec Kubernetes) qui m'était inconnue. J'ai pu aussi participer à la conception et l'implémentation d'un processus, ce qui m'a permis de mieux appréhender le travail en groupe sur un tel projet.

7 Politique de Responsabilité Sociétale de l'Entreprise Sia Partners

Cette section correspond à une analyse personnelle de Sia Partners vis à vis des enjeux RSO

La culture de Sia Partners est fondée autour de 6 valeurs partagées par l'ensemble des employés :

- L'excellence
- L'entreprenariat
- L'innovation
- La culture du partage
- La culture de la bienveillance
- Équilibre Travail – Vie privée

Les stratégies de l'entreprise sont très régulièrement partagées aux consultants par l'intermédiaires de communiqués de presse interne. Concernant la stratégie de l'UC data science, tous les 6 mois à lieu une plénière au cours de laquelle les directeurs nous font part de la stratégie mise en place sur les mois à venir.

A chaque nouvelle acquisition, un mail est envoyé aux employés pour expliquer ce choix stratégique. Sia Partners organise également des soirées annuelles pour ses clients au cours desquelles sont communiquées les différentes actions et décisions mises en place par l'entreprise sur cette année.

Promouvoir la diversité c'est appliquer une politique d'égalité dans l'intégration aux opportunités d'emploi offertes. A cette fin, Sia Partners s'engage à reconnaitre les diversités culturelles, religieuses et individuelles de ses clients et employés. Le groupe s'engage également à proposer les aménagements de reconnus RQTH. Particulièrement attentive à la prévention des discriminations, l'équipe RH de Sia Partners s'engage à respecter les principes de non-discrimination dans la mise en place de sa politique RH et à appliquer la charte du recrutement responsable.

Sia Partners dispose d'une charte d'achat responsable qu'elle fait signer à l'ensemble de ses fournisseurs.Cette charte fait partie intégrante de l'approche de Sia Partners en matière de développement durable. Au travers de cette charte, Sia Partners invite les fournisseurs à agir contre la corruption sous toutes ses formes, y compris l'extorsion de fonds et les pots-de-vin. Le non-respect de cette charte peut engendrer la résiliation du contrat.

Le code de conduite mis en place par Sia Partners et signé par les employés stipule que les employés de Sia Partners s'engagent à ne tirer aucun gain personnel au détriment des intérêts du groupe et à n'exercer aucune fonction externe qui pourrait compromettre son rôle chez Sia Partners.

8 Conclusion

8.1 Bilan des missions et bilan personnel

Durant ce stage j'ai pu donc participer à 3 missions :

- Webscrapping pour Cleep : Plus de 220 000 sites scrappés et un scrapper générique d'image presque au point.
- Moteur de recommandation pour Cleep : Moteur créé de A à Z, fonctionnel et en production. Il continue à être améliorer par les développeurs de Cleep, mais le squelette reste le même.
- Travaux sur le sialab : J'ai aidé à l'amélioration du processus de déploiement mais aussi j'ai travaillé sur l'infrastructure des plateformes sialab ainsi que leur adaptation au processus de déploiement.

Je trouve que ces 6 mois passés à Sia Partners m'ont permis d'apprendre beaucoup vis à vis de la data science, mais aussi du métier d'ingénieur. Concernant la data science, j'ai pu apprendre plus sur le sujet du webscrapping. J'ai pu utiliser plusieurs technologies différentes notamment en base de données (Base de donnée noSQL en graphe, Base de donnée SQL). J'ai pu aussi produire un moteur de recommandation de manière presque autonome et donnant un résultat concret en production, en se basant uniquement sur des recherches faites au préalabes. Finalement, la partie data engineering et développement fullstack m'a permis d'étudier toute l'infrastructure derrière ce qu'on appelle le "one-click deploy" et d'expérimenter les difficultés de monter son propre cluster, même avec les services comme Google Cloud qui facilitent le travail.

Concernant le métier d'ingénieur en général, j'ai appris au cours de cette expérience qu'être ingénieur ne résidait pas en maîtriser un grand nombre de sujets, mais en savoir rechercher efficacement des réponses à une problématique et trouver les bons outils qui vont nous permettre d'y répondre le plus efficacement possible. J'ai également appris qu'un ingénieur se doit d'être en constant apprentissage que ce soit sur le plan métier, gestion de projet ou technique.

8.2 Perspectives professionnelles

Concernant la data science, ce stage n'a fait que conforter le fait que je désirais travailler en tant que data scientist. Travailler sur des applications très différentes de la data science, au cours de ce stage, n'a fait qu'augmenter mon intérêt pour ce domaine et le désire de continuer d'apprendre dans ce secteur.

J'ai par ailleurs aimé travailler dans le milieu du conseil chez Sia Partners. Contrairement à d'autres entreprises, on ne fait pas que présenter les solutions possibles avec une Proof of concept, on accompagne les entreprises jusqu'au bout du projets. Mon travail sur le sialab est l'un des meilleurs exemples.

Cependant, bien que mon stage soit une réussite, je choisi de continuer mes études pour l'instant à Imperial College of London, pour un master en machine learning. Les cours proposés ainsi que la vision plus recherche de la datascience m'intéresse afin d'avoir toutes clés en mains pour mon avenir professionnel.

9 Annexes

9.1 Code source allégé du site de la fnac

9.2 Graphe tests ArangoDB / Neo4j

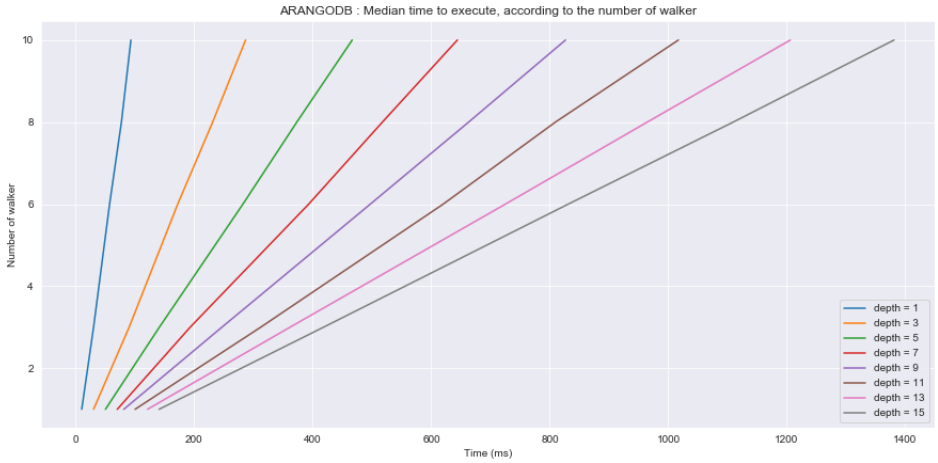

Figure 29: Performances d'ArangoDB sur marche aléatoire en fonction du nombre de chemin aléatoire

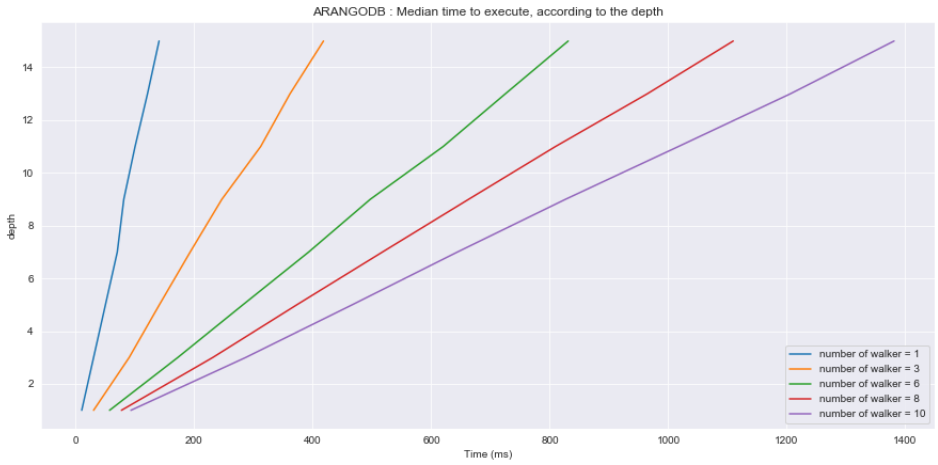

Figure 30: Performances d'ArangoDB sur marche aléatoire en fonction de la profondeur

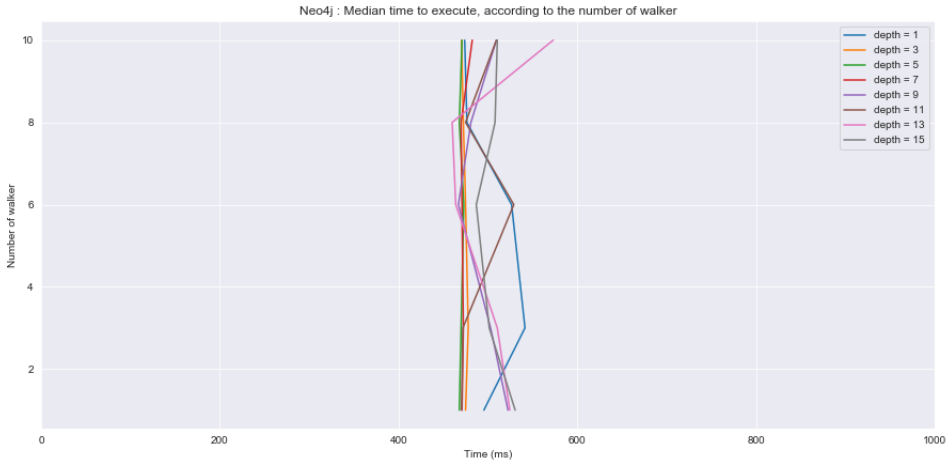

Figure 31: Performances d'ArangoDB sur marche aléatoire en fonction du nombre de chemin aléatoire

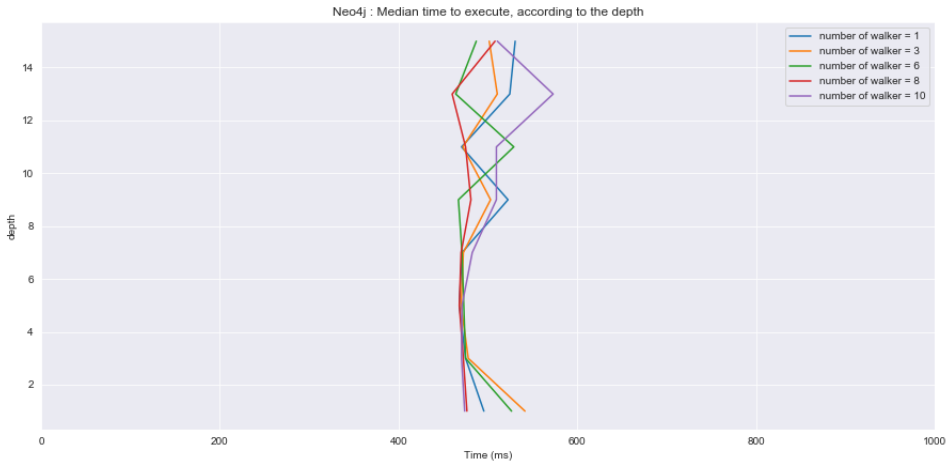

Figure 32: Performances d'ArangoDB sur marche aléatoire en fonction de la profondeur

9.3 Exemple : Heuristique de Pinterest appliqué au cas de Cleep

Le graphe **figure 33** sert d'exemple pour cette partie. Le cleep en rose est le cleep à recommander. Les cleeps violets sont les autres cleeps du graphe et ceux en gris sont les utilisateurs. Pour des raisons de simplicités, je n'ai pas inclus les listes.

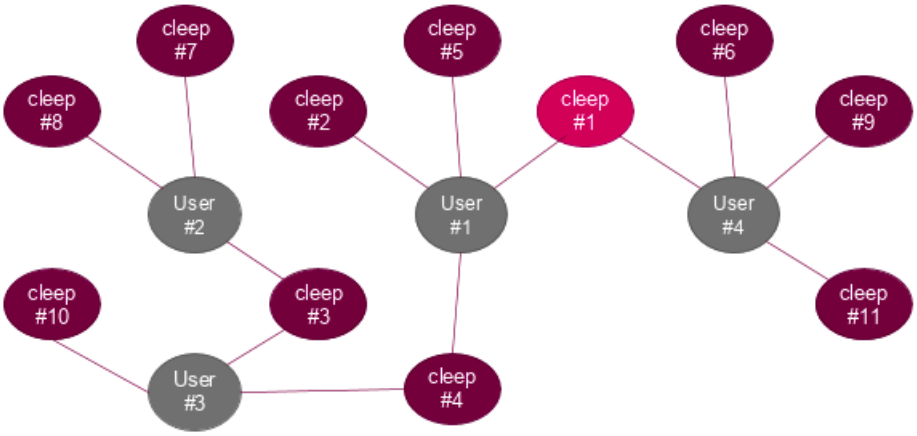

Figure 33: Graphe de départ

Les **figures 34, 35, 36** décrivent le processus de chemin aléatoire. Dans cet exemple, il y a 3 marches aléatoire effectuées (désignées par les chemins vert) donc 3 cleeps possiblement recommandés : les cleeps #8, #10 et #11.

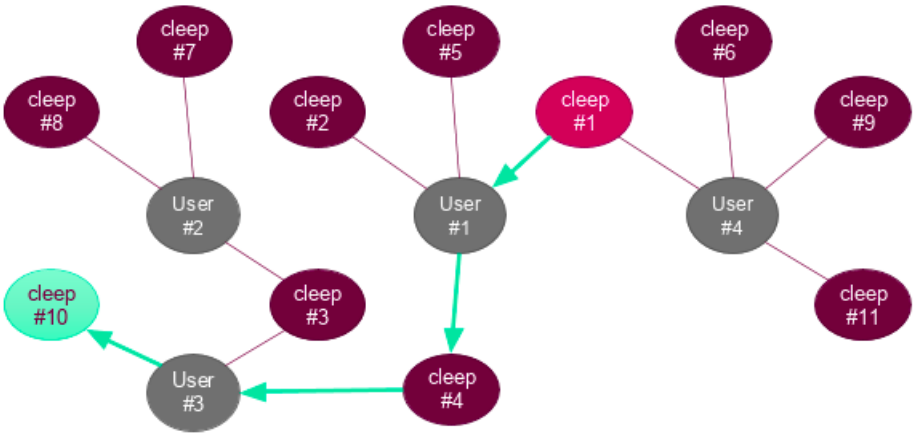

Figure 34: Chemin aléatoire n°1

La **figure 37** décrit le processus de selection des meilleurs cleeps. Les cleeps se voient d'abord appliquer un score. Si ce score dépasse un certain seuil comme c'est le cas pour les cleeps #10 et #11, alors ces cleeps peuvent être poussés comme recommandations.

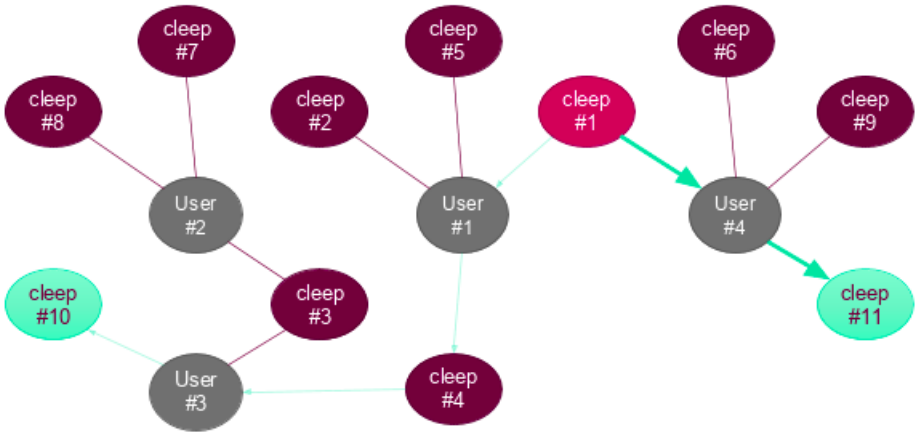

Figure 35: Chemin aléatoire n°2

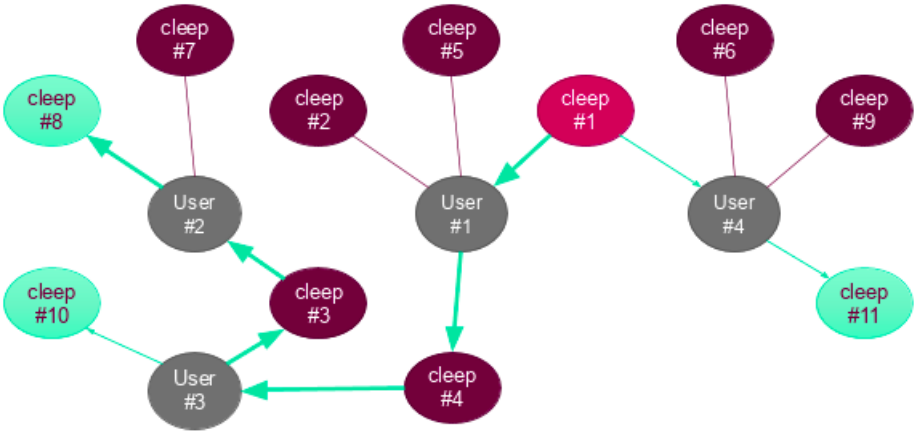

Figure 36: Chemin aléatoire n°3

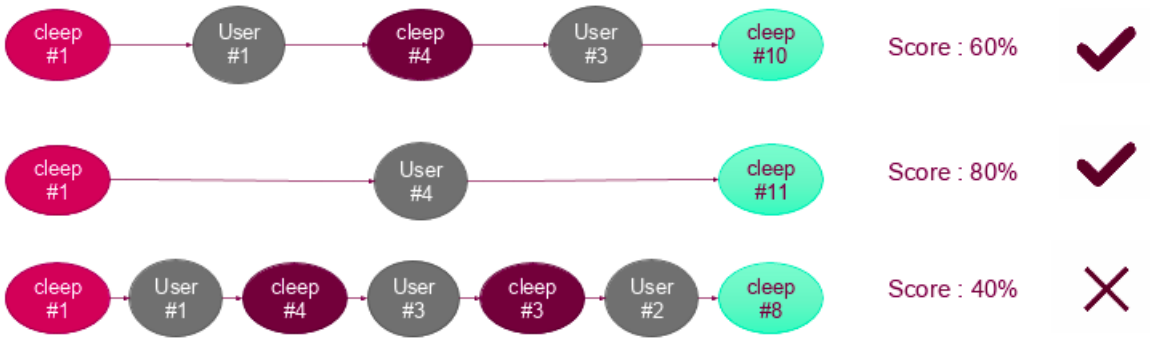

Figure 37: Scoring des Cleeps

9.4 Exemple de variable d'environnent

```yaml
---
authentication:
project:
  hostname: lab.sia-partners.ai
  log-level: INFO
  name: showroom/bot
  protocol: https
project-database:
  hostname: "0.0.0.0"
  name: db
  password: pwd
  port: "5432"
  type: postgres
  user: user
  database: db
```

```yaml
---
authentication:
  inhouse:
    jwt-secret-key: secretKey
    password-recovery-secret-key: pwd
    type: bearer
project:
  hostname: lab.sia-partners.ai
  log-level: INFO
  name: sialab-project-name
  protocol: https
project-database:
  hostname: "0.0.0.0"
  name: sialab-project-name
  password: pwd
  port: '5432'
  schema: sialab_backend
  type: postgres
  user: user
```

Figure 38: Variables d'environnements

10 Glossaire

References

[1] Site de Cleep : *https://app.cleep.io/*

[2] Exemple de la Fnac : *https://livre.fnac.com/a13195869/Agnes-Martin-Lugand-A-la-lumiere-du-petit-matin*

[3] Exemple site internet avec plusieurs produits : *https://livre.fnac.com*

[4] Approfondissement sur la
de Levenshtein : *https://en.wikipedia.org/wiki/Levenshtein_distance*

[5] Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time : *https://labs.pinterest.com/user/themes/pin_labs/assets/paper/paper-pixie.pdf*
Auteurs : Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, Jure Leskovec

[6] Site Internet de Pinterest: *https://www.pinterest.com/*

[7] Documentation sur les algorithmes de neo4j:
https://neo4j.com/docs/graph-algorithms/current/experimental-algorithms/

[8] Documentation sur Cypher :
https://neo4j.com/docs/cypher-manual/current/introduction/#cypher-introduction

[9] Fonction d'aggregation sur Cypher:
https://neo4j.com/docs/java-reference/current/extending-neo4j/procedures-and-functions/aggregation-functions/

[10] Bot détection de fraude:
https://lab.sia-partners.ai/showroom/fraud-detection/

[11] Comparaison machine virtuelle/ container docker:
https://www.sdxcentral.com/containers/definitions/what-is-docker-container/

[12] Site Internet de nginx:
https://www.nginx.com/

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

IMT Atlantique Bretagne - Pays de la Loire - www.imt-atlantique.fr

Campus de Brest
Technopôle Brest-Iroise
CS 83818
29238 Brest Cedex 03
T +33 (0)2 29 00 11 11
F +33 (0)2 29 00 10 00

Campus de Nantes
4, rue Alfred Kastler - La Chantrerie
CS 20722
44307 Nantes Cedex 03
T +33 (0)2 51 85 81 00
F +33 (0)2 51 85 81 99

Campus de Rennes
2, rue de la Châtaigneraie
CS 17607
35576 Cesson Sévigné Cedex
T +33 (0)2 99 12 70 00
F +33 (0)2 99 12 70 08