MÉMOIRE DE STAGE

VERDIER Quentin

04/ 05/ 2021

EPSI B1

TABLES DES MATIERES

REMERCIEMENTS 4

INTRODUCTION 5

CONTEXT 6

Présentation de l'entreprise 6

Mission attribuée 8

ÉTUDE DE L'ENVIRONEMENT ET DES TECHNOLOGIES 9

Étude de l'environnement 9

Étude des technologies 9

RÉALISATION TECHNIQUE ET LES MESURES 11

Présentation des logiciels, normes et principes 11

Les mesures 13

GESTION DU PROJET 14

DÉROULER DU STAGE 16

CONCLUSION 24

ANNEXES 25

REMERCIEMENTS

Je souhait tout d'abord remercier mon tuteur de stage Michael BRIERE pour m'avoir permis de faire ce stage et de travailler sur ses projets. Je le remercie pour le temps qu'il a pris pour m'accompagner et répondre à mes questions. Je souhaite le remercier pour la confiance qu'il m'a donnée pour ces projets qui m'ont permis d'enrichir mes compétences en développement backend, malgré le fait que je ne connaissais pas le langage JAVA et le télétravail. Je souhaite aussi remercier Émilie SALLÉ qui m'a soutenue pour ma candidature auprès de My Money Bank. Je remercie aussi le reste du service CRM, pour leur bienveillance et leurs accueilles dans leur équipe.

INTRODUCTION

Du 2 mai 2022 au 5 juin 2022, j'ai effectué un stage au sein de l'entreprise My Money Bank (situé au 4 rue du château de l'éraudière, 44300 Nantes). Au cours de ce stage que j'ai effectué en majoritairement en télétravail, j'ai pu m'intéresser au développement back-end dans ce projet de synchronisation de base de données.

Plus largement, ce stage a été une opportunité pour moi de m'immerger en tant que développeur back-end dans une entreprise national. Au-delà d'enrichir mes connaissances en développement, ce stage m'a permis de comprendre dans quelle mesure le développement était l'une des voies dans l'informatique dans laquelle je pourrais me complaire dans mon futur professionnel. Si ce secteur m'attirait, notamment par les enjeux grandissant de l'informatique dans la société, ces cinq semaines à travailler sur ces API m'ont confortée dans mon projet professionnel.

Ainsi, je commencerai par contextualiser le stage dans le cas de ce stage, en vous présentant notamment l'entreprise et le service dans laquelle j'ai opéré, puis les missions qui m'ont été attribué. Par la suite, je vous présenterais l'étude de l'environnement et des technologies, qui fut d'avantage un travail de recherche et de veille afin de trouver les informations utiles à mes différentes missions. Ensuite, je vous exposerai la réalisation technique pour mes missions, composée notamment de la présentation des logiciels, normes et principes utilisés, et des mesures. Enfin, sera évoqué le stage en tant que tel, c'est-à-dire les différentes missions réalisées. Après avoir présenté la gestion du projet sera détaillé son déroulé durant les cinq semaines. Y seront exposés les objectifs à réaliser, les contraintes, les difficultés auxquelles j'ai fait face, les réponses apportées à ces défis, ainsi que la manière dont j'ai réalisé les différentes missions.

CONTEXT

Dans cette rubrique vous sera présenté l'entreprise du stage. Il sera évoqué tout d'abord la présentation de My Money Bank et son organisation. Après cela vous sera exposer les Missions qui m'ont était attribué.

Présentation de l'entreprise

My Money Group est née du rachat de la SOVAC, fondée par Andrée Citroên en 1919, en vue de proposer des solutions de financements pour vendre ses voitures à crédits, par le groupe General Electric. Plus tard, elle fait l'acquisition du Crédit de l'Est pour élargir ses activités en métropole, puis de Réunibail et de la Royal Saint Georges, basés sur les territoire de l'Outre-Mer. En 2017, General Electric cède son activité bancaire à un fond d'investissement, «Ceberus ».

My Money Group est une société qui regroupe plusieurs établissements bancaires, et scindé en plusieurs structures selon ses activités :

- Un pôle métropolitain, appelé en interne « DC » pour plus de simplification :
    - My Money bank, société de regroupements de crédits avec garanties hypothécaires ou non dont les principaux clients sont les clients ou des partenaires bancaires
    - My Partner Bank, racheté fin 2020, qui est une enseigne bancaire spécialisée dans les prêts immobiliers pour les professionnels
- Un pôle Outre Mer, appelé « My Money Outre Mer » ou « DOM » par souci de simplification :
    - Les « SOFI »
        - Sorefi : société Réunionnaise de financement, spécialisée dans le financement de véhicule, propose aussi du prêt à la consommation
        - Somafi- Soguafi : société Réunionnaise de financement, société martiniquaise, guadeloupéenne, guyanaise de financement, spécialisée elle aussi dans le financement de véhicule et dans le prêt à la consommation.

- Banque des Caraïbes, anciennement société générale banque des antilles, racheté le 2 mars 2020, banque de détail traditionnelle.

Dans cette entreprise il y a environ 1000 salariés : 543 salariés en métropole, 230 à Paris, 333 à Nantes, environ 150 à La Réunion, et 300 aux Antilles-Guyane. Malgré les différente marques le groupe travaille main dans la main et mutualise des services et des fonctions supports telles que le services CRM qui s'occupe de la gestion des prospects et des clients pour My Money Bank métropole (« DC ») , pour les « SOFI » et pour la Banque des Caraïbes (« DOMs »).

Lors de la cession de GE Bank au fond d'investissement Cerberus, il était important pour le groupe de refonder son organisation pour développer de nouvelles technologies rapidement et proposer des outils efficaces et compétitif aux collaborateurs, partenaires et clients. C'est donc assez naturellement que l'entreprise s'est tournée vers la *méthode Agile*, et plus particulièrement la *méthode Scrum*, pour développer et suivre ses projets informatiques.

L'équipes CRM est une équipe avec peux d'effectifs. il y a 3 IT, Information Technology, composer d'un lead développeur, Michael BRIERE, et de 2 consultants, Victoire R. et Rodolphe G., qui ne sont présent que à mi-temps. Ils vont effectuer les taches techniques de l'équipe. Ils sont accompagnés d'un PO Emilie LEHOUCQ, Pruduct Owner, ou responsables de produits en français. Le produit est le CRM. Émilie LEHOUCQ accompagne les « proxi PO » ou facilitateurs du CRM DC, géré par José H. et les CRM SOFI et BDC gérés par Emilie S. et Ewen U. pour la partie DOM.

Le rôle des Proxi PO ou facilitateurs est de :

- Récolter les besoins des collaborateurs concernés, en l'occurrence les équipes commerciales qui travaillent au quotidien sur le CRM
- Rédiger les besoins sous forme d' « US » (« user storie ») inscrites dans une backlog commune au CRM,
- De remonter les éventuels beugs sur plateforme CRM,
- De recetter les évolutions dans des environnements dédiés,
- De démontrer les évolutions apportées pendant le sprint lors des revues de sprint

- De former les équipes sur des nouvelles fonctionnalités
- De préparer des tableaux de bords pour le commerce et des paramétrages du quotidien pour permettre au commerce de piloter son activité.

Mission attribuée

Pour commencer mon stage ma première mission a été de me renseigner sur les différents logiciels, méthodes et principes qui sont utilisé dans l'entreprise ainsi que de lire les documentations interne sur les technologies existantes et modes de fonctionnements.

Ensuite lorsque j'ai eu une vision plus claire sur le fonctionnement de l'entreprise et l'organisation j'ai pu faire quelques travaux tel que mettre à jour de la documentations interne, des schémas ou de la documentations sous Ascii Doc. J'en ai aussi profité pour préparer une revue de sprint avec une démonstration pour les utilisateurs.

Après m'être informée sur les technologies et méthodes, j'ai eu pour responsabilité la création d'un service, dans une librairie interne, d'Alerting Teams en cas de présences de leads en erreurs bloqués dans la base de données. Le projet était de préparer l'algorithmie pour ce service après m'être renseigné. En intégrant une fonction de flipping c'est-à-dire de pouvoir désactiver et réactiver les notifications à volonté sans arrêter l'API et pouvoir changer le Webhook qui est le lien vers la conversation d'équipes Teams dans le quelle les notifications doivent arrivées. Ainsi qu'utiliser des variables d'environnement telle que pour la fréquence des notifications pour que chaque API puisse y est indépendante si besoin. J'ai aussi organisé les cas de recettes et effectué les tests en conséquence, puis j'ai mis à jour la documentation suite à mes ajouts. Au travers de cette mission, j'ai dû maitriser le langage JAVA et les librairies utiliser dans l'entreprises.

ÉTUDE DE L'ENVIRONEMENT ET DES TECHNOLOGIES

A présent, penchons-nous sur l'étude de l'environnement, en d'autre-termes, l'étude de l'intégration de la librairie et particulièrement l'Alerting Teams dans les APIs pour analyse et y adapter le service. Ensuite, nous verrons l'étude des technologies possibles dont celles qui existent déjà pour n'en retenir que les plus pertinentes.

Pour rappel, Teams est un outil de conversation collaboratif utilisé au sein du groupe développé par Microsoft. Et les librairies sont des ensembles de fonctions et classe déjà codé et conçu pour être réimplémenter dans d'autre code.

Étude de l'environnement

Avant de commencer le projet d'alerte via teams en cas de leads (prospect) en statut d'erreur, il a fallu que je me renseigne pour connaitre les API qui utiliseront cette nouvelle fonctionnalité. Etant données que les APIs sont exécutées dans un orchestrateur de conteneurs, il est possible qu'une API soit utilisée en multi-instance pour avoir une vitesse de traitement augmentée. C'est-à-dire que la même API serait exécutée de manière identique plusieurs fois pour mais se répartirait les données à traiter pour travailler plus vite. Donc je devais trouver une solution pour ne pas se retrouver dans le cas ou plusieurs API voudrait envoyer le même messages d'alerte en plusieurs exemplaires. J'ai aussi dû me renseigner auprès des utilisateurs pour savoir quelle fréquence serait la plus adéquate pour traiter au plus vite l'erreur sans être inondés d'alertes pour autant. Enfin je me suis intéressé à d'autre projets intégrant la fonctionnalité d'envoi de message dans teams, ce qui a été utile aussi pour la création du projet initiale d'Alerting Teams.

Étude des technologies

Tout d'abord intéressons-nous aux librairies qui sont utilisés dans cette librairie interne. Les librairies principales sont *Spring*, qui intègrent beaucoup de fonctionnalités pour simplifier la création des classes et des configurations pour le démarrage de l'API.

Puis *Mokito* qui permet de créer des « mokes », qui sont des attributs façades que l'on utilise pour faires les tests et renvoyer au code les données qui nous intéressent dans notre test.

Pour la missions, nous avions aussi une base de données fictive pour effectuer nos tests sur des datas non sensibles et prouvant être manipuler à notre guise. Cette base de données contenait aussi des paramètres qui était utilisé par des services divers dans la librairie.

RÉALISATION TECHNIQUE ET LES MESURES

Ici l'IDE, environnement de développement intégré choisie, les extensions, les logiciels, les normes et les principes vont vous être présentés et expliqués afin d'offrir un aperçu précis des conditions dans lesquelles les missions ont été réalisées, avant d'évoquer les mesures du projet.

Présentation des logiciels, normes et principes

Les APIs du CRM de My Money Bank sont développé en Java qui est un langages créés par Sun Microsystems en 1995.

Pour faire ce projet, j'ai utilisé l'environnement de développement *Intellij*, éditée par *JetBrains*. C'est un IDE dédié au développement de logiciel sous Java, qui est un langage dédié à la création d'application embarqué, ou ses déclinaisons. Il est très utile car il dispose d'une assistance intelligente de codage qui vérifie le code pendant la saisie et analyse l'intégralité du projet. Il a aussi une refactorisation rapide et sécurisée et offre une interface visuelle très utile pour gérer le projet avec *Git*. Mais a l'inconvénient d'être très gourment en ressource.

Il aurait donc pu être judicieux de choisir Visual Studio Code. C'est pourquoi ce tableau ci-dessous montre comment ce choix a été orienté, et pourquoi il n'a pas été retenue pour ce projet :

	Intellij	Visual Studio Code
Accessibilité	Gratuit	Gratuit
Spécialité	Développement JAVA	Un peu de tout
Débogage	5/ 5	0/ 5
Examen du code	5/ 5	5/ 5
Développement de logiciel	5/ 5	0/ 5
Facilité d'utilisation de Git	5/ 5	4/ 5
Prise en main	4/ 5	2/ 5

tableau de comparaison entre Intellij et Visual Studio Code

De plus *Intellij* permet d'ajouter beaucoup d'extensions crées par d'autre utilisateurs, telles que *ASCIIDoc*, *Maven Helper*, *SonarLint* et bien d'autre. Les extensions que j'ai citées m'ont été utile lors de mon stage car elles m'ont permis d'être plus efficace et de gagner en productivité. L'extension *ASCIIDoc* permet de visualiser de la documentation écrite sous Ascii Doc qui est un langage de balisage léger ce qui permet que la documentation écrite soit lisible pour l'humain mais aussi rendre un fichier HTML ou bien PDF avec une belle mise en page, ce qui m'a était très utile pour les mises à jours de documentation. *Maven Helper* est une extension qui simplifie énormément l'utilisation de Maven en tant qu'outil de gestion d'un projet car elle permet de gérer les versions des librairies utilisées. Elle permet aussi de pouvoir exclure les versions comportant des failles de sécurité. Cette extension propose aussi beaucoup d'outil intégré, telle que le dependency-check qui consiste à vérifier dépendance des librairies utilisées et de vérifier s'il n'y a pas de failles de sécurité dans ces librairies. Et pour finir *SonarLint* qui est simplement une extension permettant une analyse *Sonarqube* du code sur sa qualité son efficacité, sa sécurité et la quantité de code natif couverte par les classes tests dans l'IDE. Cela optimise le code avant même de le publier sur le git et devoir le corriger ensuite.

Git est un logiciel de gestion de version. Il permet de travailler de façon collaborative sur un seul projet. J'ai donc dû apprendre comment utiliser les branches correctement pour avoir un travail organisé et cohérent. Chez My Money Bank les branches sont organisé par une branche = un développeur. Le nom de la branche doit être explicite par rapport à la tache effectuée, afin de faciliter les recherches dans l'historique des modifications.

Nous avions Docker, un logiciel permettant d'héberger localement des conteneurs logiciels et des bases de données afin de pouvoir faire des tests en local. De plus l'entreprise utilise Docker dans son orchestrateur de conteneur pour les APIs.

PostMan qui nous permet d'interagir avec les bases de données, nous a aidé à faire nos tests en effectuant des requêtes http. Les actions principales sont d'effectuer des demandes de résultat avec des GET, des ajouts à l'aide de POST, des modification avec PATCH, supprimer à l'aide de DEL.

La base de données n'avait pas besoin d'être crée car je travaillais sur une librairie interne qui possédait déjà sa propre base de donnée sous PostgreSQL, cela permet une compatibilité avec les flux JSON et je n'avais que quelque paramètres à stocker ajoutés.

Lors de ma veille j'ai appris les principes qui étaient utilisés dans l'entreprise, telle que le principe de développement du Test-Driven Development. Cela consiste à commencer la réflexion d'abord par les tests qui seront effectuer dans le code. Cela implique de prévoir tous les cas possibles mais a l'avantage de simplifier l'algorithmie et réduire les oublies. Ainsi que la Sémantique des versions avec un référentiel distinct. Par exemple. Les APIs ont des versions 1.0.2, elles sont toutes composer de 3 nombres. Le premier nombre représente les mises à jour majeurs. Le second les mises à jour mineurs. Et le dernier, les corrections de bugs.

Les mesures

Durant ce projet nous avons pu faire plusieurs mesures grâce à des tests local à l'aide de la base de données fictives et de docker pour héberger l'API en local. Un exemple de message envoyé est présentée à l'annexe n°1. Mais la librairie n'a été incorporé dans les APIs seulement après mon départ alors je n'ai pas pu voir le service d'alerte teams fonctionner mais j'ai eu des retours grâce à l'équipe CRM pour me dire que tout se passait bien.

GESTION DU PROJET

Dans cette partie sur la gestion de projet vous sera expliquée l'organisation des missions. En effet, je présenterai ici la réalisation des différentes et leurs planifications.

Etant donné que j'étais majoritairement en télétravail et que je voyais mon maitre de stage généralement 1 jour par semaine sur site je devais donc travailler en autonomie accompagné des conseils de mon maitre de stage. Nous faisions des points réguliers via l'outil de conversation Teams et des réunions sur l'avancement de mes missions.

Pour commencer, l'équipes a pris le temps de m'expliquer l'environnement de travail et quelles est leurs métiers. L'équipe est organisée avec un outil de suivi des tâches dans une backlog appeler Jira. Il est présenté sous forme de tableau (voir annexe n°2). Ses tickets sont ensuite répartis dans des colonne selon leurs avancement.

- La première colonne est « A Faire »,
- ensuite « En Cours »,
- « En Revue » qui correspond à une validation avec les parties prenantes sur l'orientation du ticket,
- « A Valider En Dev » faire valider le code créer par les compères,
- et enfin « A Déployer En Prod » pour mettre à disposition aux collaborateurs via des livraisons de solutions.

L'outil *Jira* est administré par le Scrum Master

Mon maitre de stage m'a d'abord donné des conseils sur les renseignements pour que je maitrise l'environnement afin d'avoir une vision claire des projets. J'ai lu les documentations interne. Il m'a ensuite parler de méthode de travail telle que la méthode Test-Driven Developement, (CF : annexe 3 - schéma d'organisation des APIs, du DataHub et du CRM à l'annexe n°3.).

Suite à cela le sprint touchait à sa fin, j'ai donc aidé à la préparation pour la revue de fin de sprint avec le métier, et j'ai pu participer à cette revue.

Au sprint suivant, j'ai eu comme projet l'Alerting Teams que j'ai commencé par un brainstorming avec Michael pour qu'il m'expliqué précisément le projet et les idées qu'il avait trouvé. J'ai pu y faire mes remarques et proposer des modifications pour l'algorithme. Dans la suite du projet, j'ai organisé des réunions pour récolter des informations auprès des utilisateurs afin de leur livrer un produit répondant à leurs besoins. J'ai pu effectuer des points avec Michael lorsque je me retrouvais bloqué par manque de connaissances mais aussi lorsque je voulais vérifier et valider chaque étape de mon projet. J'en profitais pour réfléchir plus précisément à l'étape suivante.

DÉROULER DU STAGE

A présent, dans cette partie, vous pourrez suivre en détail le déroulement des missions qui ont été accomplies durant ces cinq semaines. De l'objectif à réaliser jusqu'aux moyens utilisés pour livrer mon projet, en passant par les différentes contraintes et les difficultés auxquelles j'ai fait face, et les solutions utilisées pour y remédier. La première partie sera consacré à l'enrichissement des connaissances dans l'entreprise. La seconde partie consternera elle le projet d'Alerting Teams.

Pour comprendre la mission qui m'a été affectée, il était important que je m'imprègne du contexte et de l'organisation de l'entreprise et de l'équipe. J'ai eu beaucoup de réunions avec les personnes de l'équipes CRM pour qu'ils m'expliquent leurs rôles dans l'équipe et leurs organisations. J'ai donc appris comment fonctionne une équipe sous méthode Agile. J'ai aussi pu me renseigner sur les logiciels que je devrais utiliser et leurs finalités. Ensuite, j'ai lu la documentation interne pour comprendre quelles étaient les technologie internes. J'ai approfondie mes connaissances sur l'organisation, notamment le rôle des APIs et les projets en cours. Pour commencer, le principale projet en cours de création était un système permettant de mettre en lien les informations entre le DataHub et le CRM de manière automatique. Avant de trouver une solution, les utilisateurs devaient ressaisir les informations manuellement du CRM au DataHub. Ce qu'ils leurs prenais beaucoup de temps et donc ce n'était pas toujours fait. Un CRM, ou Customer Relation Ship, est un outil dédié au commerce dans lequel se trouve une base prospects et clients, l'ensemble des interactions avec les prospects et clients, et des processus qui permettent aux équipes de finaliser une vente, ou de la reconduire dans le cas des clients. Les prospects arrivent depuis le site web et sont insérer dans le CRM grâce à une API. C'est prospect sont alors dans un espaces appart le temps d'être ajouté au CRM après avoir vérifier les informations. Alors la peuvent être trouvé des erreurs et donc les prospects passe du statut « attente » à « erreur ».

Avant de commencer, Michael m'a demandé d'installer mon environnement de développement, avec les logiciels dont j'avais besoin puis m'a aidé pour les configurer. Malheureusement, durant les configurations, nous avons rencontrés quelques erreurs

bloquantes. Tous d'abord, pour utiliser Git de manière plus simple nous voulions utiliser les fichier .gitconfig et .git-credentials. Mais pour des raisons inconnues le fichier gitconfig ne supportait pas le chemin « ~/ .git-credentials » alors nous avons essayé avec le chemins du fichier en entier mais ça ne corrigea pas le problème. Nous avons décidé de retester avec notre première solution, c'est-à-dire le chemin raccourci, et, cette fois-ci, cela a fonctionné. Nous n'avons pas trouvé d'explications pour cette erreur. Malgré que le fichier git-credentials comportait pourtant tous les bons paramètres. Git demandait quand même l'identifiant et mot de passe. Le problème est que pour respecter les normes de sécurité de l'entreprise les mots de passe sont créés par un logiciel tiers et nous devions nous y connecter pour accéder au Gitlab. Pour nous connecter à Gitlab nous devions utiliser un Token. Après plusieurs essaies et modifications du fichier, les credentials furent reconnue, grâce à un retour à la fin dans le fichiers.

Lorsque j'ai commencé à comprendre l'environnement, j'ai effectué des tâches attribuées par Michael telle que la mise à jour de la documentation interne. Etant donné que de nouvelles API était créés, il fallait donc les ajouter à la documentation. et les intégrer dans le schéma, ce schéma est présent en annexe n°3.

Après cela j'ai effectué des vérification de sécurité. J'utilisais SonarQube qui est utilisé par l'entreprise, pour analyse le code. Je corrigeais ce qui n'allait pas dans le code. Ensuite, je faisais un dependency-check, vérification des dépendance en français, ce qui permettait d'analyser toutes les librairies utiliser dans le code pour nous renvoyer un fichier HTML nous montrant les failles de sécurité existante dans celle-ci. Au début je ne savais pas quoi faire avec, après des recherches en vin j'ai donc demander de l'aide à Michael qui a pu me montrer comment le corrigé. Au début, je ne savais pas quoi faire malgré plusieurs recherches mais en vain. J'ai donc demandé de l'aide. Michael m'a montré comment le corriger :

- Pour commencer, on vérifie dans le fichier pom.xml, qui est utilisé par maven pour gérer les versions des librairies existantes.
- Ensuite, on vérifie que la version utilisée est bien la dernière publiée grâce au site de Maven qui regroupe l'ensemble des librairies et affichent toutes les versions.
- Puis on vérifie s'il reste des vulnérabilités, si la version est bien à jour

- Enfin, on regarde si les librairies comportant des vulnérabilités sont utiles ou non.

Mais cela nécessite une profonde connaissance des projets et des librairies que je n'avais pas encore acquis à ce stade. J'ai donc fait le choix d'exclure les librairies comportant des vulnérabilité puis de relancer l'API afin de vérifier qu'elle exécutait bien tous les tests du code. Si des tests se passais bien, alors je pouvais les exclure. En revanche, si Maven remontait un problème il y a deux possibilités : modifier le code ou laisser la faille à condition qu'elle ne soit pas utilisable ou de la mettre en stand-by le temps de lever la vulnérabilité.

J'ai ensuite assisté Michael sur un ajout pour une API permettant de faire des simulations de prêts. Il y a eu 2 demandes sur cette API. La première demande était l'ajout d'une information qui est le pourcentage de prêts immobiliers par rapport aux autres prêts. Le calcul exact nous avez était donné, c'était « prêts immobiliers / (prêts immobiliers + autre prêts) ». Ensuite nous devions l'afficher en pourcentage. Etant donné qu'il y avait déjà une méthode permettant de mettre en formes les pourcentages pour les afficher. Nous avions juste à faire le calcule et utiliser cette fonction. C'était une actions qui fut rapide et simple. La seconde demande avait pour but d'aider les conseillers. Lorsqu'ils font une simulation et qu'il y a un non-respect des normes de prêts, cela ressort l'erreur. La simulation de prêts était refusé mais les conseillers n'avaient aucune information précise de ce qui avait bloqué la simulation. Même si les conseillers doivent connaître les normes de prêts, il arrive qu'il y ait des erreurs surtout lors de la montée en compétence des équipes. Nous avons donc crée une notification contenant les noms des valeurs qui ne correspondaient pas aux normes définies, afin de faciliter la prise en main des conseillers nouvellement recrutés.

Ensuite nous avions la revue de sprint à préparer. Nous présentions le début du projet de synchronisation entre le CRM et le DataHub donc nous avons fait un schéma expliquant ce qu'il avait était fait et ce qu'il restait à faire. A ce moment du sprint, seulement la coquille de l'API producer avais était faite. Cette API envois un messages dans le DataHub pour prévenir la création d'un prospects. Ensuite une API du Bureau Universelle de My Money Bank li cet évènement et si les informations l'intéresse. Il interroge une autre API pour avoir les informations complémentaire.

Suite à cette revue de sprint nous avons commencé le sprint suivant une réunion pour choisir les tâches pour le prochain sprint. Étant donné que les taches avait déjà été évaluée par ordre de priorité, elles ont ensuite été évalué sur les points d'efforts demandé pour les réaliser. Enfin nous défilions les tâches dans l'ordre de priorité et chaque membre s'attribue une tâche tout en s'assurant qu'il pourra la réaliser dans la temps imparti. Lors de cette planification, j'ai donc eu la tâche de créer un système d'alerte qui envoie un messages dans teams lorsque un leads comportant une erreur arrive dans la base de données. Pour commencer cette réflexion, Michael m'a aidé à analysé le travail à faire. Etant quelque chose qui va être utile pour l'équipe DOM et l'équipe DC je créer donc cette fonctionnalité dans une librairie. Ce qui va permettre de l'intégré plus facilement dans les APIs des 2 coté. Je devais donc réfléchir à comment faire pour que les APIs utilisant cette librairie puissent changer les valeurs telle que la fréquences, le fuseau horaires, le lien WebHook pour le choix du canal Teams. Et je devais aussi intégrer une fonctionnalité en temps réel d'activation/ désactivation du service de notifications.

J'ai commencé par faire un schéma séquentiel du fonctionnement du service de notification teams avec le service schedule / monitoring que vous pouvez retrouver à l'annexe n°4. La fonction scheduled consulte la base de données et vérifie la présence de lead en statut erreur. Si des leads en erreur sont présent alors le service monitoring devra préparer le payload, qui est le formatage du message teams géré grâce à du JSON, en le récupérant et le modifiant avec le nombre de leads en statut d'erreur trouvé. Ensuite ce payload préparé est envoyé au client teams. Le webhook et le paramètre pour la fonction monitoring sont dans une variable d'environnement dans un fichier application.yaml qui permet de surcharger des valeurs au démarrage d'une API, ce qui permettait que chaque API, DOM et DC, puissent choisir leurs valeurs.

J'ai donc mis les valeurs pour mon WebHook et les paramètre du schedule. Étant donné que les personnes qui vérifient la présence de leads en erreur le faisait en début de matinée et en début d'après-midi. J'ai mis une fréquence sous CRON pour que la vérification s'active à 8h et 14h du lundi au vendredi. La CRON était donc « 0 8,14 * * MON-FRI ». CRON vient de chrono table, soit « table de planification ». C'est un utilitaire simple basés sur Unix. Il permet de planifier une action et d'être lisible par l'homme. Le format est simple « *<second>* *<minute>* *<hour>* *<day-of-month>* *<month>*

<day-of-week> ». Les « * » sont utilisé pour comprendre toutes les valeurs, les « - » permettent de faire des intervalles, et les « , » permet de mettre plusieurs valeurs. Peu de temps après, j'ai donc fait un point avec les personnes concernées par ses vérifications pour savoir quelle serait la fréquence la plus adaptée pour ces vérifications. Ces personnes m'ont répondu qu'elles préféreraient avoir une vérification tous les heures entre 8h et 19h compris du lundi au vendredi. J'ai modifié ma CRON par « 0 0 8-19 * * MON-FRI ». J'avais aussi créer une fréquences pour toutes les minutes pour me faciliter les tests en local.

Ensuite, j'ai commencé à préparer la création du Payload. C'est un fichier JSON permettant de mettre en forme un message pour teams. J'ai utilisé le site : https:/ / messagecardplayground.azurewebsites.net/ dédié à la création de Payload JSON et qui permet d'avoir un aperçu en temps réel du message dans le logiciel, Teams dans notre cas.

Après cela j'ai continué la création du service monitoring. J'ai commencé par travailler sur la fonction monitoring, qui ne comportait pas encore beaucoup d'algorithmie. On vérifie seulement la présence de leads en erreurs en utilisant une méthode de récupération de requetés par statut DAO. Data Access Object est une méthode permettant de transcrire les informations récupérée en une entité Objet pour l'utiliser dans une API, qui était déjà créé. Tout en suivant la méthode Test-Driven qui implique de penser les tests avant la solution je me suis souvenue que la requête sur la base de données pourrait me renvoyer des exceptions. Les exceptions étant des messages d'erreurs et si elle ne sont pas anticipé peuvent faire stoppé l'API. La partie la plus compliqué était d'apprendre à gérer les exceptions possible que la requête pouvait créer. Ensuite je me suis penché la récupération du payload et de sa modification. Ici aucune exception n'est possible car le fichier du payload est toujours à la même place et ne bougera pas sur tout le déploiement de l'API.

Ensuite, j'ai fait les classes tests dont les tests pouvant couvrir toutes les possibilités. J'ai donc dû créer la classe test et moké l'argument repository qui est utilisé pour la requête DAO. Grâce à la librairie Mokito je peux définir le contenue renvoyé pour ensuite pouvoir vérifier que le code a bien exécuté ce que je voulais, c'est à dire les méthodes ou tester des variables avec le comportement attendu. Et avec le débuggeur

nous pouvons surveiller e comportement du code ligne par ligne grâce à des points d'arrêts que nous plaçons dans l'IDE. Ceci m'a permis de faire un tests où la requête ne renvoie rien comme si aucun leads était en statut d'erreur tout en vérifiant donc que la préparation du payload n'a pas eu lieu. Ensuite, un test ou la requête renvoie un leads en statut d'erreur trouvé. Où j'ai pu vérifier que le payload a bien été modifié et comportait bien la partie « Présence de 1 lead(s) ». Enfin, j'ai réalisé un test où la requête renvoie une exception pour vérifier que le code l'attrape bien pour et vérifier que l'API ne s'arrête pas car il ne peut pas exécuter la suite.

Suite à cela j'ai fait un point d'organisation afin de faire valider mes tests. Les tests étant validé je me suis donc tourné vers le client teams. Étant donné qu'une API utilise déjà une classe client teams je l'ai donc repris. J'ai fait les tests de l'API et vu si les messages étaient correctement envoyé dans teams. Et j'ai donc réalise que l'image que j'avais choisie n'était pas supporté par teams. J'ai donc essayer de la remplacer, de changer le format, de la compresser mais le meilleur résultat que j'ai obtenu était d'avoir la moitié supérieur de l'image qui s'affichait. J'ai demandé de l'aide à Michael mais nous n'avons pas réussi a trouvé une meilleure solution. Nous avons décidé que cela n'avait pas d'impact sur l'efficacité mais seulement sur son esthétisme. Nous considérons donc cette tâche finis.

Je réalise donc la fonction de flipping pour activer et désactiver les notifications. Cette fonction est très simple car un cas similaire est utilisé dans la librairies pour le système de rétention actif qui consiste à retenir les leads dans la base de données sans les envoyé dans le CRM. Il y a besoin d'ajouter le paramètre dans la base de donnée et de mettre la classe DAO des paramètres à jour pour pouvoir l'utiliser dans l'API. Le paramètre est donc en boolean, il n'a pour valeur que vrai ou faux. Alors dans l'algorithme j'ai juste rajouté une fonction « if » pour si le paramètre est vrai l'on continue le processus sinon l'on s'arrête. Quand la fonction était intégrée j'ai donc mis à jour mes tests avec l'ajout de la fonction de flipping et la documentation de la librairie.

Après une vérification avec Michael nous organisons un petit brainstorming pour réfléchir sur la gestion du problèmes de la multi-instance. Michael m'a laissé un temps de réflexions et j'ai émis la piste d'utiliser la base de données pour mettre les actions en commun entre API. Ensuite Michael m'a expliqué un principe de gestion de multi-

instance utilise par l'entreprise. Il est simple et efficace. Il consiste à enregistrer toutes les actions devant être effectuées qu'une seule fois dans la base de donnée ainsi que l'ID de l'API. Ensuite, un algorigramme a été créé, il est présent dans l'annexe n°5. L'algorithme commence par vérifier si une action est enregistrée. S'il n'y en a pas, on ajoute un registre avec la date et l'heure actuel et l'ID de l'API. S'il y a déjà un registre, on compare l'ID enregistré avec l'ID de l'API. S'ils sont identiques, on met à jour le registre et on poursuit le processus de notifications car la fonction est programmer et ne s'effectuer que toutes les heures entre 8h et 19h du lundi au vendredi. Si c'est son ID, c'est-à-dire quelle a fait la dernière vérification de notifications il y a au moins une heure. Si les ID sont différents, on regarde quand la dernière action a eu lieu. Si elle date de moins d'une heure, on s'arrête et le processus a été fait. Sinon on s'en occupe et on met le registre à jour. Après temps, l'on s'est demander si vérifier l'ID était vraiment utile. Nous avons conclu qu'on pouvait garder l'ID pour l'utiliser en cas informatif si un problème est rencontré.

Pour appliquer cet algorithme dans le code Il a fallu ajouté une table avec PostgreSQL dans la base de données paramètre, comportant 2 colonnes. L'une avec l'ID de l'API et l'autre avec la date et l'heure de la dernière action effectuée. Ensuite, j'ai du créer la classe DAO et les requêtes pour utilise ces paramètres dans le code. Ce qui a était compliqué car pour créer la classe DAO, on utilise une librairie. Et comme je ne connaissais pas j'ai dû regarde comment était créer les autres classes DAO en cherchant dans les fichiers pour s'en inspiré. J'ai donc dut faire une requête pour ajouter les valeurs. Une autre, pour le récupérer sous la classe objet. Et enfin un delete pour supprimer les valeurs déjà présentes pour faire la mise à jours des valeurs. Après avoir fait ces requête DAO j'ai fait les tests pour vérifier qu'elles fonctionnaient bien. Ensuite, j'ai travaillé sur la méthode comportant l'algorithme pour le multi-instance que j'ai met dans une méthode à part pour faire mes tests plus facilement sans devoir recontextualiser ce qui a déjà était tester. Le problème a été que je ne savais pas comme faire pour créer des ID pour chaque API sachant qu'elles auraient le même code. J'ai demandé à Michael d'être accompagné, il m'a appris que les conteneurs d'API avais des ID donc on les utiliserait et cela facilitera la recherche de l'API dans l'orchestrateur de conteneurs. Le problème était qu'on ne savais pas comment les récupérer et qu'on ne pouvait pas le test en local. Nous avons décidé de ne pas traiter cette problématique afin

de pouvoir continuer le projet dans le temps qui m'était imparti. J'ai donc continué ma méthode pour l'intégrer dans le processus de notifications. Ma méthode était faite en boolean comme pour la fonctions de Flipping. Elle renvoie « True » ,soit « Vrai » si l'actions est à faire et « False », soit « Faux » si elle est faite et que l'on doit s'arrêter. J'ai donc rajouté ma fonctions de multi-instances dans mes tests et dans la documentations.

Etant donné que j'arrivais sur la fin de ma période de stage, j'ai donc mis à jour la documentation de l'API. J'ai voulu tester en local encore une fois pour être sûr que tout se passe bien puis pour proposer une merge request sur le Git qui est un proposition du fusion entre la branche cible qui là est le main c'est-à-dire la branche principale que l'on considère comme statut de finalisé avec ma branche qui est une branche de création de nouvelle fonctionnalité, c'est-à-dire en cours jusqu'à ce qu'on la fusionne avec la principale.

CONCLUSION

Lors de ce stage chez My Money Bank sous la tutelle de Michael BRIERE, j'ai pu effectuer beaucoup de mission que les développeurs effectuent habituellement dans leurs quotidiens, telle que ce renseigner, s'organiser à travailler en équipe, et faire des projets comme le projet d'alerte teams en cas de leads en statue d'erreurs dans la bases de données.

Après avoir pris conscience des méthodes de travails, de l'organisations de l'entreprise et de l'équipe. Etant arrivé en cours de sprint, il a fallu un temps d'adaptation pour l'équipe et moi. Ils ont en effet eu du mal à m'attribuer des tâches. Et le travail à distance n'a pas facilité mais m'a rendu plus autonome. De mon côté, je voyais difficilement le projet dans sa globalité. Mais les personnes de l'équipe on fait tous leurs possible pour m'aider. Et m'ont permis de remplir les missions attribues à bien.

Ce stage, en adéquation avec mon projet professionnel m'a permis de découvrir plus amplement JAVA, un langage que je n'avais pas étudié auparavant. Il m'a conforté dans mon projet de travailler dans le développement, puisque j'ai pu réaliser les missions demandées dans le temps qui m'était imparti. J'ai su trouver des solutions aux problèmes auxquels j'étais confronté. J'ai apprécié me confronter à la réalité du développement en entreprise et réaliser une fonction d'API. J'ai hâte de mettre à profit les enseignements des expériences de ce stage de cinq semaines.

ANNEXES

Sources :

JAVA : https:/ / www.lebigdata.fr/ java-guide-complet

IntelliJ / IDEA : https:/ / www.jetbrains.com/ fr-fr/ idea/

Visual Studio Code: https:/ / code.visualstudio.com

GIT : https:/ / www.atlassian.com/ fr/ git/ tutorials/ what-is-git

PostgreSQL : https:/ / www.lebigdata.fr/ postgresql-tout-savoir

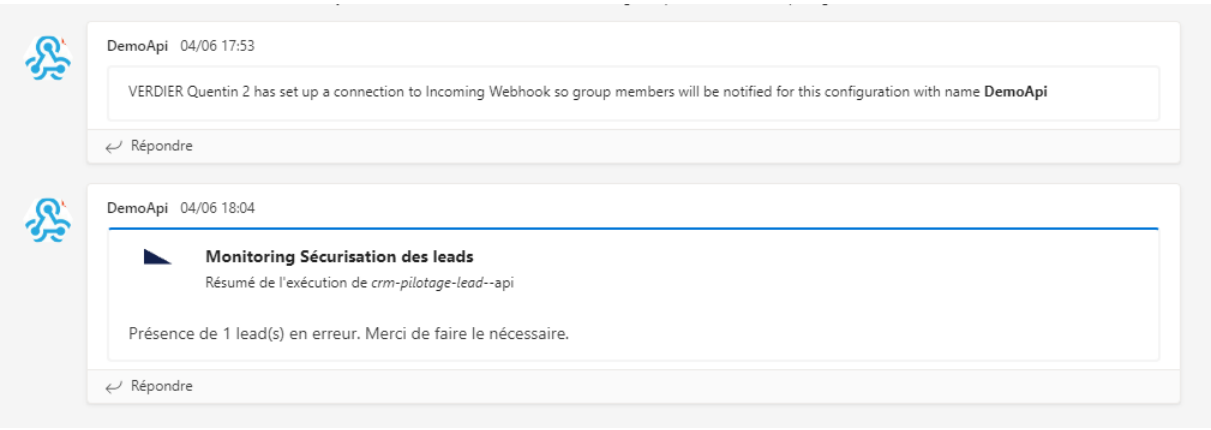
PostMan : https:/ / practicalprogramming.fr/ postman/

Version Sémantique : https:/ / semver.org/ lang/ fr/

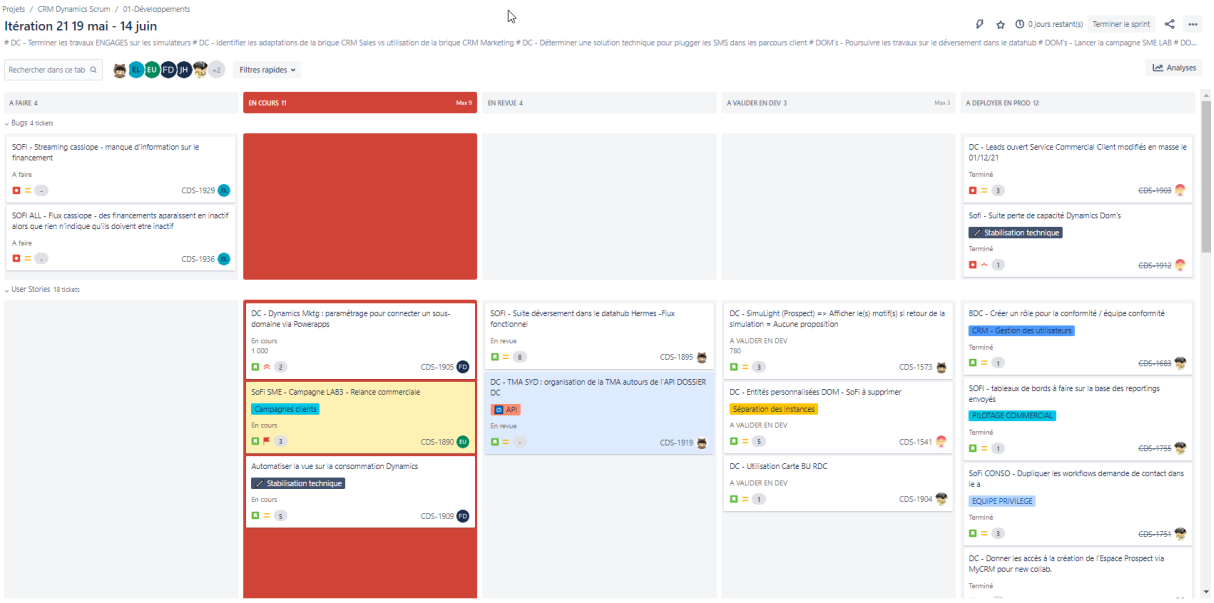
CRON : https:/ / www.baeldung.com/ cron-expressions

DAO : https:/ / cyrille-herby.developpez.com/ tutoriels/ java/ mapper-sa-base-donnees-avec-pattern-dao/

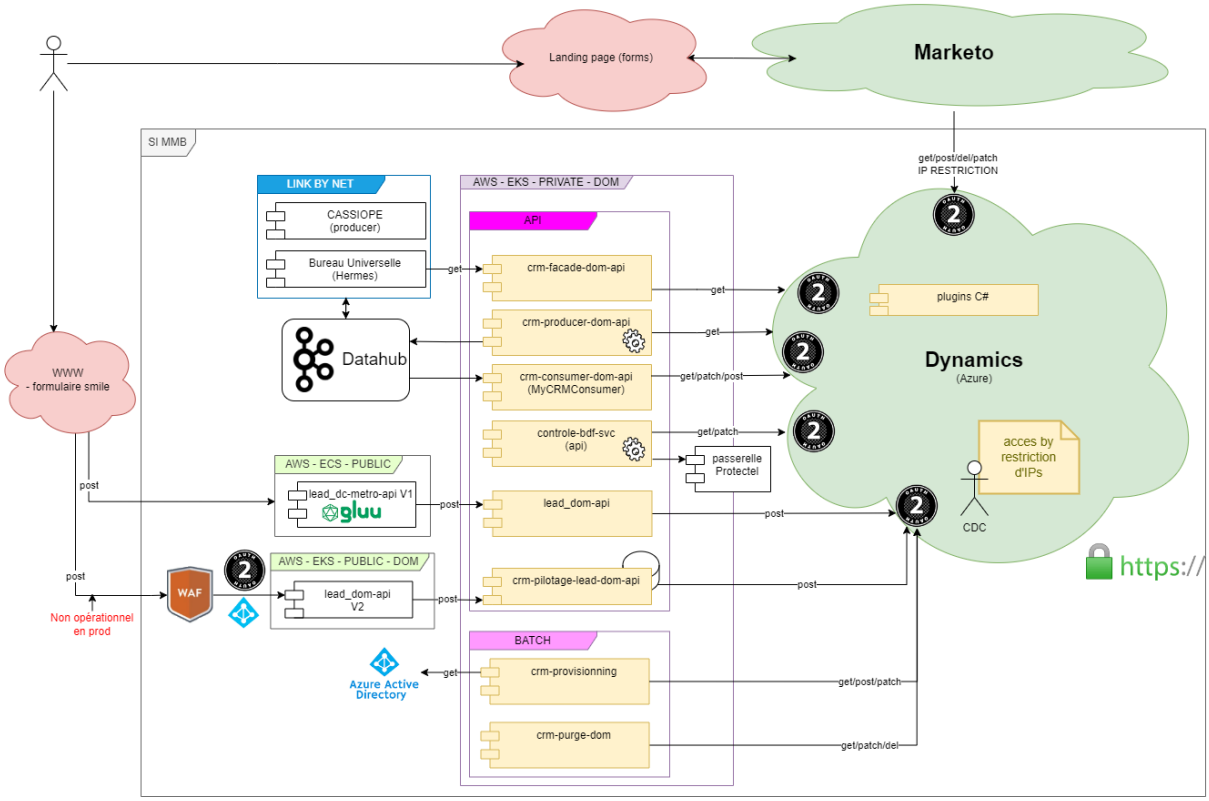
Captures et schémas :



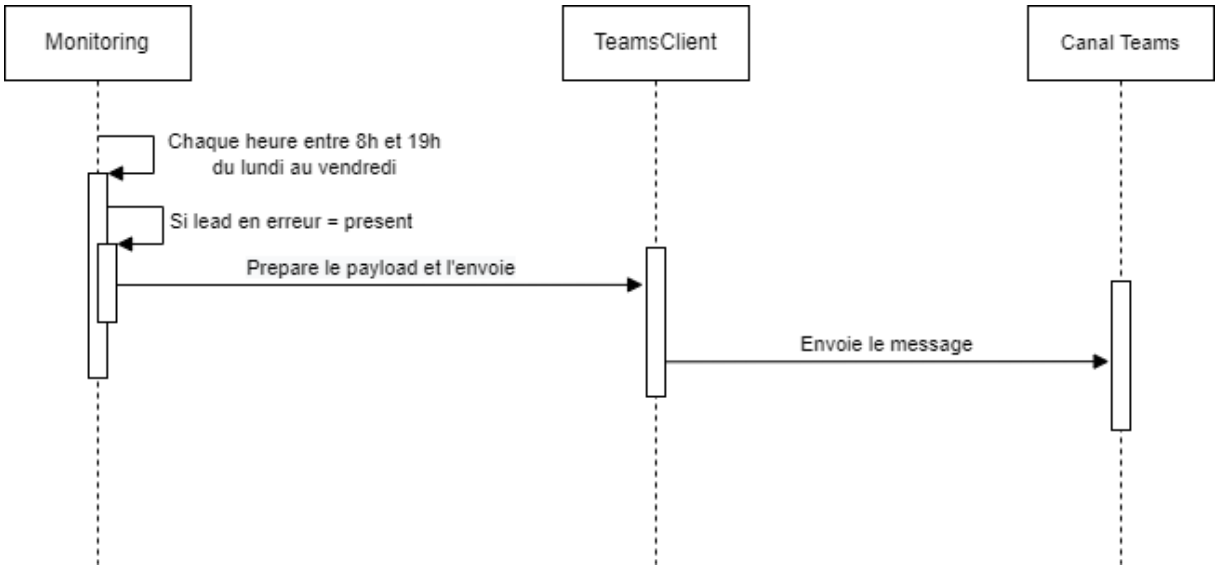
Annexe n°1 : Messages Teams envoyé par l'API



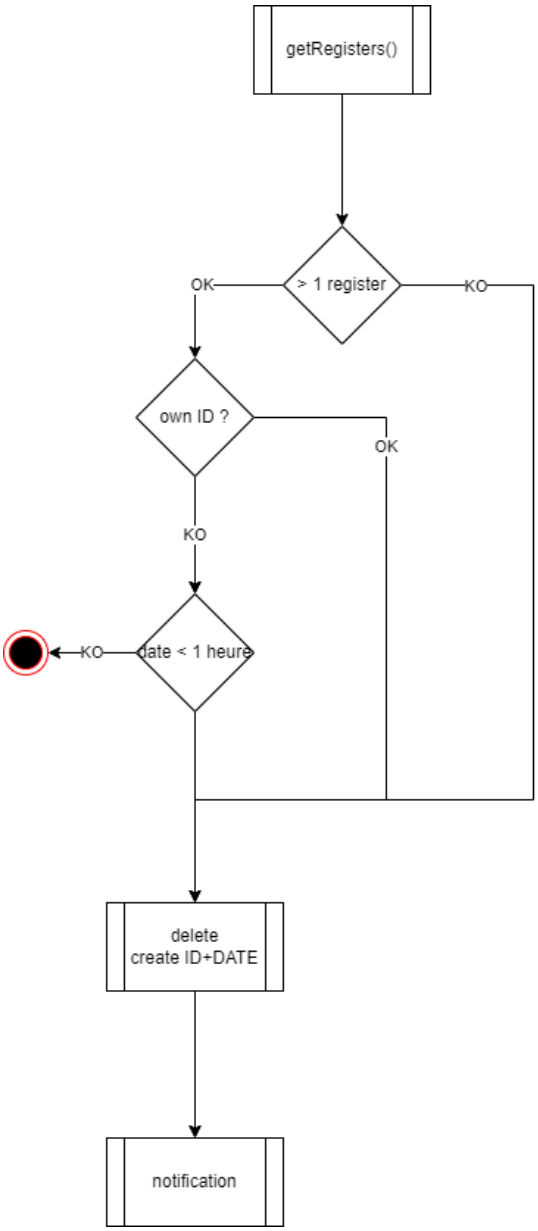
Annexe n°2 : Tableau de gestion des taches Jira



Annexe n°3 : Organisation du réseau My Money Bank



Annexe n°4 : Diagramme Séquentiel de la fonction Monitoring



Annexe n°5 : Algorithme pour le multi-instances