IUT de Villetaneuse
Sorbonne Paris Nord
99 Avenue Jean Baptiste
93430 Villetaneuse

Sage
10 place Belgique
92250 La Garenne-Colombes

Rapport de stage

Intitulé du poste : Développeuse logiciel

© Locaux Sage de Paris

Selma MATAICHE
Du 22 Janvier au 24 Mars 2025

Maître de stage : M. Ouissame GHMIMAT, développeur chez Sage
Tuteur de stage: M. Martinez

BUT Informatique – 2ème année

Remerciements

Je tiens à exprimer ma gratitude et mes sincères remerciements aux personnes qui m'ont accueilli et ont contribué à la réussite de mon stage au sein de Sage. Leur accueil chaleureux, leur pédagogie et la confiance qu'ils ont placé en mes capacités ont sublimé cette expérience, la rendant à la fois très enrichissante sur les plans professionnel et personnel.

Leurs encouragements m'ont également permis de renforcer ma confiance en moi et de confirmer mon envie de poursuivre mon parcours dans le domaine.

Je tiens tout d'abord à remercier mon maître de stage, Ouissame Ghmimat, développeur chez Sage, pour sa patience, son accompagnement et les précieux conseils qu'il m'a donnés tout au long de mon stage. Je remercie également Emmanuel Bodard, team leader, ainsi que Guillaume Boisson, de m'avoir permis d'effectuer mon stage au sein de son équipe et de m'y avoir intégré chaleureusement.

Je suis également très reconnaissante envers le reste de l'équipe, Nadine, Marie-Ange, Sylvie, Thierry, Xavier, Karim, François et William, qui m'ont fait une place dans leur équipe, et qui ont toujours été disponibles pour m'aider et répondre à mes questions. Leur accueil a fortement contribué à mon épanouissement pour cette première expérience en entreprise.

Je remercie par ailleurs, toutes les personnes qui ont pris de leurs temps pour me présenter avec passion leur rôle et leur travail quotidien : Mushiya, Florent, Yannick, Murielle, Catherine, Christophe et Joanna.

Je remercie enfin les alternants de l'entreprise qui ont participé à m'offrir un cadre de travail agréable. Leurs expériences et leurs conseils d'étudiant ont été très précieux.

Sommaire

Introduction

Actuellement en deuxième année de BUT Informatique à l'IUT de Villetaneuse, affilié à la Sorbonne Paris Nord, j'ai eu l'opportunité d'effectuer un stage de développeuse logiciel d'une durée de 8 semaines chez Sage, situé à la Garenne-Colombe. Cette entreprise est spécialisée dans le développement logiciel. Sage développe des solutions innovantes et adaptées aux besoins des entreprises, notamment en termes de gestions et d'organisations.

L'objectif principal de ce stage est de nous faire découvrir le monde professionnel, de mettre en pratique les compétences acquises au cours de notre formation et d'acquérir des compétences nouvelles. J'ai eu l'opportunité d'atteindre ces objectifs au sein de l'équipe logistique qui développe la partie applicative du produit X3. Ainsi, l'un des enjeux de cette expérience, était, de quelle manière appliquer et adapter les compétences acquises à l'IUT dans le contexte de l'entreprise ?

I. Présentation de l'entreprise

Sage est une entreprise anglaise spécialisée dans le développement de logiciels de gestion et d'organisation. L'entreprise se place comme l'un des leaders mondiaux du secteur. Depuis sa création en 1981, Sage accompagne les petites et moyennes entreprises en leur proposant des solutions innovantes et performantes pour optimiser leur gestion comptable, financière, commerciale et des ressources humaines.

Depuis sa création, l'entreprise s'adapte constamment aux évolutions technologiques, mais aussi aux besoins de ses clients. Cette évolution constante en fait sa force, et garantit sa fiabilité et la sécurité de ces solutions. Ainsi, Sage développe des outils performants adaptés aux besoins des professionnels, toujours dans l'objectif de leur faire gagner en efficacité et en productivité.

A. Son histoire

L'histoire de Sage commence en 1980 à **Newcastle Upon Tyne** au Nord-Est de l'Angleterre avec **Graham Wylie**. Ce dernier, étudiant en informatique, effectue un stage d'été au sein d'une entreprise comptable subventionnée par le gouvernement britannique pour développer un logiciel d'aide à la comptabilité à destination des PME. C'est dans le cadre de ce stage qu'il rencontre **David Goldman**, un imprimeur qui recherche une solution pour optimiser ses coûts. Les deux hommes se rendent très vite compte du potentiel de ce genre d'outils. En s'associant à **Paul Muller**, un universitaire travaillant pour la NASA, ils décident de fonder **Sage** en **1981**. Le nom de la société leur est inspiré par une affiche publicitaire pour une marque locale de bière à la sauge (sage en anglais) dans un pub où le trio se réunit, d'où le logo évoquant la plante aromatique.



Les travaux de Graham Wylie débouchent sur un premier logiciel, Sage Accounts. Ce premier logiciel obtient un grand succès. Puis, en 1984, la société met sur le marché Sage Software ce qui lui permet de décoller et de faire son entrée à la bourse de Londres en 1989.

Ce succès fulgurant s'explique par la vision innovante du trio, qui diffère de ce qui est proposé par les autres entreprises du secteur à l'époque. En effet, si la plupart des acteurs du marché du logiciel pensent qu'il faut un modèle de logiciel unique pour toutes les entreprises, Sage pense au contraire qu'il faut proposer des solutions spécifiques et adaptées aux différentes formes d'entreprises et aux réglementations de chaque pays. Alors, au lieu d'exporter ses logiciels et de tenter de s'imposer, l'entreprise préfère travailler avec des filiales locales. Ainsi, Sage entre dans une période de grande expansion qui durera jusqu'en 2008. Aujourd'hui, Sage diffuse ses logiciels dans plus de 100 pays est qui sont traduits dans 20 langues, et compte 23 filiales. On peut les retrouver en :

- Afrique de l'ouest
- Afrique centrale
- Maroc
- Algérie
- Allemagne
- Amérique de sud (hors Brésil)
- Belgique et Luxembourg
- DOM-TOM
- Europe centrale
- Europe de l'Est
- Fédération de Russie
- Irlande
- Liban
- Turquie
- Madagascar
- Tunisie
- Pays-Bas
- Royaume Uni

B. Ses produits

Bien que l'entreprise soit à l'origine spécialisée dans les logiciels de comptabilité, son expansion et ses filiales lui ont permis de diversifier ses offres en proposant des solutions logicielles adaptées à tous les besoins des entreprises, de toutes tailles. Ainsi, on retrouve des logiciels de :

- **comptabilité et gestion commerciale** :
  - **Sage Active** : Une solution cloud dotée d'intelligence artificielle, conçue pour les petites entreprises de 2 à 20 salariés. Elle permet d'automatiser les flux financiers, d'achats et de ventes, offrant une vision en temps réel de la trésorerie.
  - **Sage 50** : Destiné aux TPE, ce logiciel offre des fonctionnalités complètes de comptabilité, de gestion des devis et factures, de gestion des stocks et de trésorerie.
  - **Sage 100** : Conçu pour les PME de 10 à 199 salariés, il propose une gestion intégrée de la comptabilité, des finances, des ventes et de la production.
- **paie et ressources humaines** :
  - **Sage Business Cloud Paie** : Une solution en ligne pour réaliser simplement les bulletins de paie et les déclarations sociales, adaptée aux petites et moyennes entreprises.
  - **Sage 100 Paie & RH** : Un outil complet pour gérer la paie et les ressources humaines, incluant la réalisation des bulletins de paie et des déclarations sociales.
- **gestion d'entreprise / ERP** :
  - **Sage Intacct** : Un logiciel de gestion financière cloud, conforme aux normes françaises et internationales, idéal pour les entreprises recherchant une solution financière avancée.
  - **Sage X3** : Une solution ERP offrant une gestion financière et opérationnelle rapide et flexible, adaptée aux PME, ETI et grands groupes.
  - **Sage 100 Entreprise** : Un ERP conçu pour les PME, centralisant les processus de gestion comptable, financière et commerciale.
- **solutions pour les experts-comptables** :
  - **Sage Génération Experts Connect** : Une solution de production comptable et sociale 100 % en ligne, facilitant la digitalisation des cabinets d'expertise-comptable.

- **Sage Service Paie** : Une solution cloud simplifiant la gestion de la paie et la mission sociale des cabinets d'experts-comptables.

II. Organisation/Structure de l'équipe et méthodologie

Pour cette première expérience, j'ai eu la chance de pouvoir intégrer l'équipe Logistique travaillant sur le développement de la partie applicative des opérations de gestion logistique du produit X3.

A. Rôle de l'équipe

L'équipe applicative travaille essentiellement sur la **conception**, **l'optimisation** et **l'adaptation** des fonctionnalités liées à la gestion des flux de marchandises, des stocks, des approvisionnements, etc. Ainsi, elle possède différentes missions. Tout d'abord, l'analyse des processus métiers logistiques afin de définir les besoins des utilisateurs de manière précise, en collaboration avec la Product Manager (PM). Une fois les processus métiers bien définis, ces derniers nécessitent des configurations de l'ERP afin qu'il réponde aux besoins et s'adapte aux particularités des entreprises (Ex : gestion de plusieurs sites, plusieurs entrepôts, plusieurs emplacements, etc.). Enfin, une fois la configuration de l'ERP et le développement des fonctionnalités, vient l'étape des tests et de la validation. Cette dernière étape est importante pour s'assurer de la bonne qualité du développement, mais aussi pour s'assurer que le logiciel répond aux besoins exprimés. Cette étape s'effectue sous la forme de tests fonctionnels et d'intégrations. Ces tests d'intégrations permettent de garantir la cohérence des versions au fur et à mesure des mises à jour du logiciel. Ainsi, les clients peuvent monter en dernière version sans que cela impact trop leur utilisation ou ne créée de bugs. Un des avantages de X3 est qu'il permet aux clients de réaliser des développements spécifiques à leurs besoins, eux même (ou par l'intermédiaire des partenaires), dans le cas où les grands paramétrages ne leur suffisent pas. Or, plus ces développements spécifiques sont importants plus il est compliqué de gérer la cohérence des versions et de faire en sorte que les améliorations de l'ERP n'aient pas d'impacts sur ces développements spécifiques. Il est donc important de vérifier que toutes les améliorations apportées n'aient pas d'impacts négatifs pour les clients.

B. Composition de l'équipe

L'équipe R&D, managée par Emmanuel Bodard (Team Leader), est composée de cinq développeurs : Marie-Ange, Sylvie, Ouissame, François et Thierry. Elle est par ailleurs constituée de deux qualiticiens : Karim et Xavier. Nous retrouvons également la Product Owner (PO), Nadine, et le SCRUM Master, William, dont on détaillera les rôles dans la prochaine partie.

Notez également que, comme dans la plupart des équipes chez Sage, tous les membres ne sont pas situés sur les mêmes sites. Bien que dans l'équipe logistique, tous les membres sont en France, à Paris ou à Annecy, il est possible que les équipes soient réparties dans différents pays. Cette organisation particulière peut-être assez lourde sur le long terme. En effet, travailler à distance et l'absence d'interaction physique, malgré les différents outils et méthodes conçus pour les compenser, ne peuvent pas remplacer le contact humain direct.

Pour mener ces missions à bien, l'équipe logistique doit également travailler avec d'autres équipes de différentes spécialités.

Tout d'abord, la **Product Manager,** dont on détaillera le rôle plus précisément dans la prochaine partie, qui représente le client auprès de l'équipe et donc qui leur donne les priorités de développement.

Ensuite, l'équipe **plateforme**, responsable de la création et de la maintenance de l'infrastructure et des outils nécessaires pour que les équipes applicatives puissent développer correctement et efficacement. Elle va notamment gérer les serveurs, les conteneurs, mettre en place des pipelines (Cf. Lexique) d'intégration, etc.

On retrouve également l'équipe **Support/L3**. Lorsque des clients rencontrent des difficultés, leurs retours passent par différents niveaux. En premier lieu, le premier niveau est pris en charge par les partenaires (qui revendent le logiciel). Ensuite, s'ils n'ont pas la capacité de traiter la demande, celle-ci est envoyée et traitée au deuxième niveau par l'équipe support. Dans la majorité des cas, les situations sont résolues à ce niveau. Dans le cas contraire, elles sont envoyées au troisième niveau, à l'équipe L3 (= Level 3), qui a plus d'expertise et sera donc d'avantage en mesure d'apporter des solutions. Ces solutions ont la forme de hotfix (Cf. Lexique) ou de nouveaux développements intégrés à la prochaine release.

Il y a par ailleurs l'équipe **Xtrem**. Xtrem est comme un module qui améliore les capacités de X3 pour la gestion des données et optimiser les performances des requêtes, le traitement et l'analyse des données.

L'équipe logistique est également en lien avec l'équipe **QA**, chargée des tests automatisés.

Enfin, l'équipe peut être en contact avec l'équipe de la **traduction et la documentation**. Comme son nom l'indique, cette équipe est chargée de rédiger les documentations, les aides en lignes, les informations et tous les textes présents au niveau des interfaces utilisateur, et de les traduire.

Cette liste est non exhaustive, mais représente les principales équipes avec lesquelles j'ai été en contact tout au long de mon expérience. Ce sont notamment ces contacts et cette diversité qui m'ont permis d'avoir une vision globale de la création d'un logiciel et de son développement. De plus, observer les différentes spécialités informatiques m'a permis d'avoir une vision plus précise de leur nature et de la manière dont elles s'agençaient toutes dans un seul et même projet.

C. Méthode de travail

Comme toutes les instances de Sage, l'équipe R&D travaille en suivant la méthode Agile et plus précisément la méthode SCRUM. Cette méthodologie de travail a été adoptée en 2012, dans l'objectif d'adapter le développement au changement permanant, de se concentrer sur les besoins des entreprises et sur les clients, mais aussi afin de favoriser la communication et la gestion des informations.

1. Les acteurs

L'agilité a apporté une organisation généralisée des équipes de développement, composées d'un PO, d'un Scrum Master, de développeurs et de qualiticiens comme présenté précédemment.

La PM (Product Manager) : elle ne fait pas partie de l'équipe, mais est en lien direct avec elle. Elle est la personne responsable du développement et de l'amélioration d'un produit. Elle

définit la vision stratégique du produit en fonction des besoins du marché. Pour cela, elle mène des études pour comprendre les attentes des clients. Ainsi, elle priorise les fonctionnalités. Enfin, elle coordonne les équipes (technique, design, marketing, etc.) pour assurer son succès. Elle agit comme un lien entre les différentes parties prenantes et s'assure que le produit répond aux objectifs de l'entreprise et aux attentes des clients.

La PO (Product Owner) : elle doit définir la vision du produit en collaboration avec la PM (Product Manager) et partager cette vision avec l'équipe afin que tous aillent dans la même direction. Elle traduit la vision du PM en tâches concrètes pour les développeurs. Ainsi, elle gère la création du **Product Backlog** (liste des fonctionnalités et des améliorations à développer), du **Sprint Backlog** et rédige les **Users Stories** (créées à partir des épics/livrables fournit par la PM). Elle réalise un vrai travail de priorisation des tâches en fonctions de leurs importances, des attentes des clients (connues grâce à la PM), et des contraintes techniques. Enfin, elle collabore avec l'équipe pour clarifier les exigences, valider les fonctionnalités et s'assurer qu'elles répondent aux attentes.

Le SCRUM Master : le Scrum Master veille à la mise en place de la méthode scrum et de l'implication de chaque membre. Pour cela, il aide tous les membres de l'équipe à s'organiser, s'améliorer et à suivre les pratiques de la méthode Agile. Il s'assure de l'efficacité de l'équipe, mais aussi que tous se sentent bien au sein de l'équipe.

2. Les pratiques

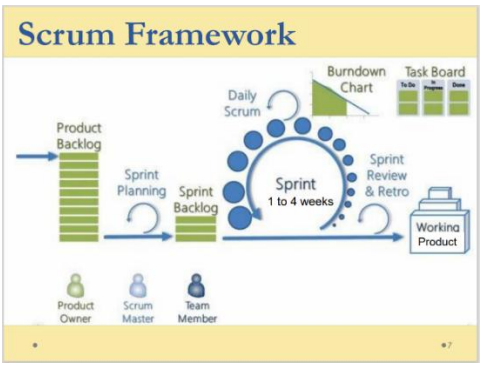
La méthode agile donne lieu à de nombreuses pratiques visant à améliorer la flexibilité, l'efficacité, la communication et la collaboration entre tous les membres d'une équipe. Dans un premier temps, il est important de comprendre les différents cycles de travail.

Le plus grand cycle est la **release**. Elle désigne la livraison d'une version du produit, et dure six mois. Il y a donc deux releases par an. Dans ces releases nous retrouvons les **sprints** d'une période courte et fixe de deux semaines. Ils assurent une livraison fréquente, réduisent les risques, permettent une meilleure réactivité aux changements et aux urgences, et encouragent la collaboration et l'amélioration continue. Un **sprint planning** est réalisé au début de chaque sprint pour définir les tâches et les livrables à livrer à la fin du sprint. Pendant les sprints, des **daily stand-up** (DSU) sont réalisés tous les matins, permettant d'avoir un suivi quotidien. Lors de ces réunions courtes, chaque membre de l'équipe partage ce qu'il a fait la veille et ce qu'il a prévu de faire le jour même. Cela permet à tous d'avoir une visibilité et une transparence sur tout ce qui est réalisé par l'équipe. Ces réunions permettent également de partager les potentielles difficultés rencontrées, afin de profiter de l'expérience des autres membres de l'équipe, en réfléchissant collectivement aux solutions et ainsi profiter d'une collaboration efficace. Puis, à la fin des sprints nous retrouvons deux cérémonies : une **sprint rétrospective** et une **sprint review :**

- La sprint rétrospective a pour objectif d'analyser ce qui s'est bien passé, ce qui peut être amélioré et de définir des actions concrètes pour optimiser les prochains sprints. Le déroulement typique est le suivant :
    - l'équipe partage ses expériences (positives et négatives),
    - elle identifie des points d'amélioration,
    - et elle définit des actions concrètes à appliquer dans le prochain sprint.

Cette cérémonie est aussi une occasion pour s'assurer que tous les membres se sentent à l'aise dans l'équipe et écouté.

- La sprint review a elle pour objectif de présenter les nouveautés réalisées sur le produit aux différentes parties prenantes.
  A ce titre, l'équipe m'a donnée la responsabilité de réaliser une démonstration de ce que j'ai développé lors d'une sprint review. J'ai donc bénéficier d'une expérience complète en termes de gestion de projet. De ce fait, avoir pu évoluer dans cet environnement Agile, m'apportera beaucoup pour mes prochains projets.



Shéma des cycles de la méthode SCRUM

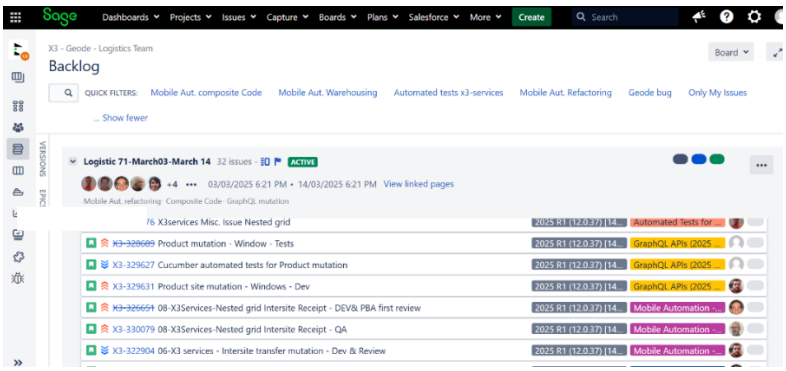
La méthode Agile, bien qu'efficace et utile pour toutes les raisons déjà évoquées, peut parfois sembler lourde, notamment à cause de ses nombreuses réunions. Ces cérémonies peuvent parfois paraître longues et empiéter sur le temps qu'on pourrais consacrer à la réalisation de nos tâches. De plus, dans l'équipe logistique le sprint planning et la sprint rétrospective sont réalisés le même jour (le lundi), en plus du daily stand-up quotidien. Etant en partie réservée pour les cérémonies, il m'est parfois arrivé de penser que c'était une journée « gâchée », car c'est du temps pendant lequel personne ne peux avancer sur ses tâches. Néanmoins, ces réunions restent des outils d'organisation utiles et des moments pendant lesquels l'équipe peut partager des instants plus légers et renforcer ses liens et sa cohésion. D'autant plus, compte tenu de la configuration géographique de l'équipe, je pense que c'est aussi un moyen de pallier la distance.

3. Les outils

Les outils de Scrum sont conçus pour maximiser la transparence des informations clé. Ainsi, tous ceux qui les consultent ont la même base, les mêmes objectifs. Parmi ces outils, nous retrouvons :

- **Jira :** il s'agit d'un outil de suivi des tâches, qui permet de mettre en place les méthodologies Scrum. Cet outil est également compatible avec Confluence et GitHub.
- **Le Product Backlog** : il s'agit de l'objectif du produit. Il liste toutes les tâches, fonctionnalités et améliorations à réaliser pour le produit de manière ordonnée. Il est géré par la PO et évolue en fonction des besoins et priorités du projet. Il est disponible sur Jira.
- **Le Sprint Backlog :** il s'agit de la liste des tâches présentent dans la Product Backlog et que l'équipe s'engage à réaliser pendant un sprint. Il est disponible sur Jira.

- **Confluence :** Il s'agit d'un outil de collaboration qui permet de créer, organiser et partager des documentations. Les documentations peuvent porter sur différents sujets tels que des process, des outils, des méthodes de travail, etc.



Backlog Sprint 71 – Jira

Bien que nous ayons eu à appliquer la méthode agile et utiliser certains de ces outils (tel que les livrables, Jira, la définition de Users Stories) lors de nos projets académiques, nos méthodologies de travail n'étaient pas assez bien appliquées dû à notre manque d'expérience. La taille de nos projets universitaires n'étant pas de cette envergure, il n'était pas simple de nous approprier ces méthodes et de les appliquer. Ainsi, avoir pu appliquer mes connaissances en gestion de projet dans cet environnement avec autant d'aspects de la méthode a été très enrichissant et me permettra de mieux gérer mes futurs projets en me réappropriant ce que j'ai observé.

D. Environnement et espace de travail

1. Infrastructure

Pendant cette période de stage, j'ai eu l'opportunité de travailler dans les locaux de Sage situé à la Garenne-Colombes (cf. Annexe 1). Cet emplacement est stratégique pour les entreprises, car il donne accès à un vaste réseau de transport à quelques minutes de La Défense. Cette proximité avec le centre d'affaire de Paris, offre une visibilité internationale, ainsi que des infrastructures modernes.

En ce qui concerne l'espace de travail en lui-même, il est organisé sous forme d'open-space (cf. Annexe 2) ce qui favorise la communication et la collaboration. En effet, bien que des « secteurs » soient définis pour les différentes activités de l'entreprise, l'emplacement des employés reste libre. Cela bannit toute routine et favorise les rencontres, les discussions et la collaboration. Cette organisation m'a notamment permis de faire de nombreuses rencontres très rapidement, ce qui a encore plus favorisé mon intégration au sein de l'équipe et de l'entreprise en général. La variété et le mélange de toutes les équipes permettent également d'observer les différentes méthodes de travail, les liens entre les équipes, et les dynamiques de travail. Le seul inconvénient de cette organisation est que les distractions sont nombreuses. En effet, les conversations entre collègues, les réunions ou entretiens en visio (parfois dans d'autres langues), le passage et les déplacements peuvent facilement distraire. Néanmoins, pour palier à cette problématique, il existe plusieurs installations, telles que des box antibruit (Cf. Annexes), de nombreuses salles de réunion, des espaces plus isolés, ainsi que d'autres espaces de pauses où il est également possible de s'installer pour travailler. La qualité des espaces de travail est importante notamment pour favoriser le retour au travail. En effet, récemment la politique de

l'entreprise au sujet du télétravail a été modifiée (en rendant plus stricts les critères de dérogation) obligeant certains collègues à venir plus souvent au bureau (Ex : deux jours de télétravail au lieu de quatre auparavant). J'ai moi-même bénéficié de deux jours de télétravail me permettant de gagner en autonomie et d'améliorer mes compétences en organisation.

2. Environnement humain

Si les infrastructures et installations sont importantes, l'environnement humain l'est tout autant. L'une des choses qui m'a le plus été bénéfique durant ce stage sont les échanges que j'ai eus. En effet, chaque personne que j'ai rencontrée avait à cœur de m'expliquer et me présenter son rôle, son équipe, ou encore son parcours. La culture de l'entreprise repose grandement sur la transmission du savoir et des compétences, favorisant un environnement d'apprentissage et de partage. En tant que stagiaire, j'ai particulièrement apprécié la disponibilité et l'écoute de l'équipe. Je pouvais poser mes questions sans crainte, et chacun prenait le temps d'expliquer avec pédagogie, rendant mon intégration et ma montée en compétences à la fois fluides et enrichissantes.

En plus de l'accompagnement de l'équipe, j'ai également pu bénéficier de formations sur l'espace « *Sage Learning* » de l'entreprise. Il existe plusieurs formations à suivre lorsqu'on arrive dans l'entreprise. J'ai eu à suivre trois formations qui portaient principalement sur la protection des données et le code de conduite de l'entreprise. Ces formations visaient à prévenir les collaborateurs sur les risques que représentent les hackers et les fuites de données par exemple. Toutefois, elles abordaient surtout les comportements à avoir afin de limiter au maximum les risques pour l'entreprise. Le code de conduite portait quant à lui sur les comportements à tenir ou non en entreprise (la corruption, la confidentialité, etc.).

J'ai également pu assister à une conférence sur le « feedback » et le travail d'équipe, avec **Madoussou Fall,** joueuse internationale française de rugby, et **Benjamin Kayser** ancien joueur de rugby. Les deux sportifs de haut niveau ont partagé leurs conseils et leur expérience de travail en équipe. C'était également l'occasion d'aborder les différences de popularité et d'évolution entre les rugbys masculin et féminin à l'occasion du tournoi féminin des Six Nations qui se déroulait à la même période. Madoussou Fall a tenu un discours inspirant auquel j'ai moi-même pu m'identifier en évoluant dans un domaine où les femmes sont encore trop peu représentées.

III. Réalisations, résultats et compétences techniques

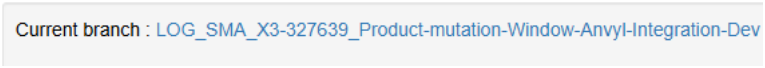
A. Mise en place de l'environnement de travail (Onboarding)

Tout d'abord, j'ai dû installer et configurer mon environnement de travail. Après la réception de mon ordinateur portable, j'ai dû suivre les étapes de l'Onboarding. Cet Onboarding consistait à me donner les accès à tous les environnements dont j'aurais besoin, parmi lesquels on retrouve : le Jira, les équipes Teams, les projets GitHub, Phabricator, Keeper, Azure, etc. (Cf. Lexique).
En effet, dans une grande organisation comme celle de Sage, les accès aux différents services et plateformes sont sécurisés et ne sont pas accessibles facilement. Ces règles strictes de sécurité rallongent considérablement l'installation d'un nouveau collaborateur. À titre d'exemple, il a fallu quatre jours pour que mon environnement de travail soit opérationnel. La création de ticket pour demander les accès aux équipes compétentes, l'attente de la réponse, l'attente que les accès

soient donnés ou encore trouver la bonne personne pour répondre à nos besoins peut s'avérer fastidieux. Heureusement, les équipes sont assez réactives et les demandes sont traitées rapidement. Néanmoins, en dehors des quelques demandes d'accès, l'installation de l'environnement de travail est documentée sur Confluence. J'ai donc pu essayer de faire les installations seule, et bénéficier de l'aide de Thierry.
Mes connaissances en système sont assez limitées et la matière n'est pas celle que je préfère dans le domaine. Ces installations m'ont donc paru longues et compliquées, mais cette expérience m'a tout de même permis de développer mes compétences et de prendre conscience de l'ampleur de l'environnement de travail d'un développeur.

L'environnement de base comprend l'environnement WSL, AWS, Ubuntu, une machine virtuelle, le Docker, les IDE VSCode et Eclipse, Git, Node.js (cf. Lexique).

Comme annoncé précédemment, l'utilisation de système de gestion de versions comme Git et GitHub est de rigueur. Ce système permet un meilleur suivi des modifications, une restauration facile en cas de problèmes et une meilleure collaboration, ce qui améliore l'efficacité et la rapidité. Malgré mon habitude d'utiliser Git et GitHub lors des SAE, pour tous les avantages cités précédemment, l'utilisation que j'en ai fait lors de ce stage m'a en plus familiarisé avec l'utilisation des branches. Une branche est une copie du code principal, sur laquelle nous pouvons travailler sur de nouvelles fonctionnalités, sans modifier directement la version principale. Ainsi, chaque développeur possède ses propres branches sur lesquelles il peut travailler en parallèle sans impacter les autres. J'ai donc dû créer mes propres branches, en respectant les règles de création telles que les règles de nommage.



LOG= Logistic team_SMA= mes initials _N° de l'US _Intitulé de l'US

Ainsi, j'ai dû naviguer entre différentes branches lors des différentes étapes de développement. J'ai également dû manipuler les différentes commandes Git qui permettent de gérer une branche, la mettre à jour, etc. Je pense que ces manipulations de branche ont été mon point faible. Le fait de ne pas avoir l'habitude de travailler de cette façon, en plus de mon manque de maîtrise à ce niveau-là m'a parfois fait défaut et m'a fait perdre en efficacité à cause d'un manque d'anticipation ou d'oubli. En effet, les « **pull** » ou les changements de branche peuvent être long. Néanmoins, ces erreurs m'ont aussi permis de mieux comprendre comment cela fonctionnait et me donne envie de remettre ces procédés en place lors de mes prochains projets pour mieux les maîtriser.

B. Développement

Le processus de développement informatique comprend plusieurs étapes essentielles permettant d'assurer la qualité et la fiabilité d'un projet. La première étape consiste en la **définition du problème**, au cours de laquelle les besoins sont définis clairement. Ensuite, vient la phase de **développement** qui englobe la conception, la programmation, les tests et la documentation. Une fois le code produit, des **tests unitaires** sont effectués pour vérifier le bon fonctionnement du développement de manière isolée. Puis, des **tests automatisés** sont également réalisés pour garantir une meilleure robustesse et détecter d'éventuelles régressions ou erreurs. Enfin, la phase **d'intégration**, durant laquelle le nouveau développement est intégré au projet complet pour son déploiement final.

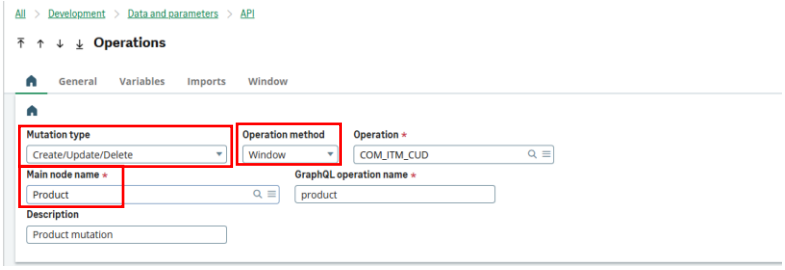
1. Définition du problème

Les améliorations de X3 depuis sa création sont nombreuses. Actuellement, l'une des fonctionnalités développées a pour but d'intégrer la gestion des **codes composites** (Cf. Lexique) par l'ERP. Le besoin de cette fonctionnalité est lié à la création de X3 Mobile Automation. Mobile Automation, permet, comme son nom l'indique, d'utiliser X3 sur mobile ou encore sur les terminaux portables de codes-barres souvent utilisés dans les entrepôts. Mobile Automation est disponible depuis la version R4 de Sage X3 publiée en 2024. La gestion des codes composites, a donc pour objectif de donner la possibilité aux utilisateurs de scanner des codes-barres des produits et de remplir les informations des produits à partir du code-barres (le client scan un produit et les informations relatives aux produits sont complétées automatiquement, à partir du code-barres.). L'intérêt de cette fonctionnalité, est de faciliter la gestion des stocks en entrepôt, la préparation de commandes, et également d'augmenter la productivité et l'efficacité des clients. L'ajout de cette fonctionnalité se fait via l'intégration d'**Anvyl** avec Sage X3. Anvyl est une plateforme de gestion de la chaîne d'approvisionnement (supply chain management). Elle est principalement utilisée pour gérer les fournisseurs, les commandes, la production et la logistique. Elle permet également d'automatiser des flux de travail tout en offrant une visibilité en temps réel. Sage a acquis Anvyl en 2024. L'enjeu de ce projet est donc la gestion des codes composites dans X3. Mais pour cela, il est également nécessaire de pouvoir créer des produits dans X3 à partir d'Anvyl. C'est sur ce dernier point que j'ai pu travailler.

2. Le Développement

a) Créer l'opération de création d'un produit dans X3

Permettre la création d'un produit dans X3 se fait en plusieurs étapes. Dans un premier temps, il est nécessaire de créer cette opération de création dans X3. Pour cela, j'ai donc dû manipuler l'ERP X3.



ERP X3 - Interface de création d'opération

Comme vous pouvez le voir ci-dessus, l'opération est caractérisée par différentes choses, mais il y a trois informations importantes. Tout d'abord, l'opération est de **type « create ».** Puis, elle est simplement basée sur le **Node « Product »** (cf. Lexique), car nous voulons agir sur les produits (création de produit). Enfin, nous remarquons également que la mutation est de **type « Window »**, cela signifie simplement qu'elle se base sur une fenêtre déjà existante dans X3. Mais alors, qu'est-ce qu'une fenêtre dans X3 ? Une fenêtre est simplement un élément de l'interface utilisateur qui permet de structurer l'affichage et l'interaction des données. Cette fenêtre est elle-même composée d'écrans qui définissent la disposition des champs. Ainsi, l'opération que j'ai créée est liée à la structure de la fenêtre. Une fois que nous avons paramétré l'opération, nous venons la valider afin que les artefacts soient générés et que les modifications soient appliquées au niveau du Node product. Concrètement, au niveau du

Node, nous observons l'ajout des lignes « canCreate / canDelete /canUpdate ». Les opérations **CRUD** sont donc autorisées sur ce Node.

```
160     externalStorageManager: new X3StorageManager({
161         joins,
162         compositeReferences,
163     }),
164     isPublished: true,
165     canRead: true,
166     canSearch: true,
167     canCreate: true,
168     canUpdate: true,
169     canDelete: true,
170     canDeleteMany: true,
171 })
```

Node Product from product.ts 1

Le fait que l'opération soit basée sur une fenêtre est encore quelque chose de nouveau et donc encore en cours de développement. Cela ne fonctionne pas toujours comme on le voudrait et ce contexte demande de **l'analyse** et de la **collaboration**. Ainsi, j'ai aussi observé cet aspect du métier qui est un peu moins présent durant notre parcours académique. Les nouvelles techniques de développement évoluent sans cesse, il est alors important de réussir à trouver des solutions et de savoir collaborer, avec diverses personnes de diverses spécialités.

b) Etude des caractéristiques produits et mutation

Maintenant que tout est généré au niveau du Node, les créations de produits sont possibles. Néanmoins, si un produit peut être créés depuis l'interface de X3, pour l'intégration d'Anvyl, il doit aussi pouvoir être créé côté **Xtrem** (le module de gestion et d'optimisation des données de Sage X3), depuis une **mutation graphQL**. Il faut donc récupérer les caractéristiques d'un produit, car s'ils sont directement récupérés dans l'interface de X3 grâce à la fenêtre, ce n'est pas tout à fait le cas pour la mutation. Ainsi, j'ai dû étudier la table (table de base de données) des produits afin de récupérer tous les champs, d'étudier leur type, s'ils sont indispensables ou non, etc. Pour cela, j'ai également commencé à manipuler les queries graphQL (requête de lecture), pour me familiariser avec **l'API graphQL** que je n'avais jamais utilisé. Mes compétences en base de données m'ont grandement servie pour cette étape. Même si je n'avais jamais manipulé graphQL, le fait d'avoir des connaissances en SQL et en gestion de données m'a permis de m'adapter rapidement à ce nouvel outil. Ici, il était essentiel que je sache transposer mes connaissances pour réussir à le prendre en main rapidement. En effet, c'est à travers celui-ci que j'allais principalement travailler par la suite, notamment pour comprendre les différents types de données et leur structure.

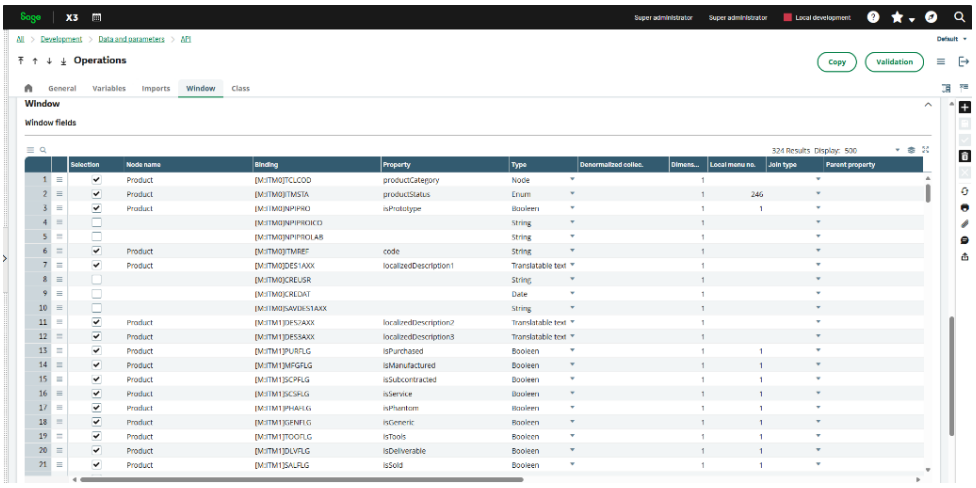
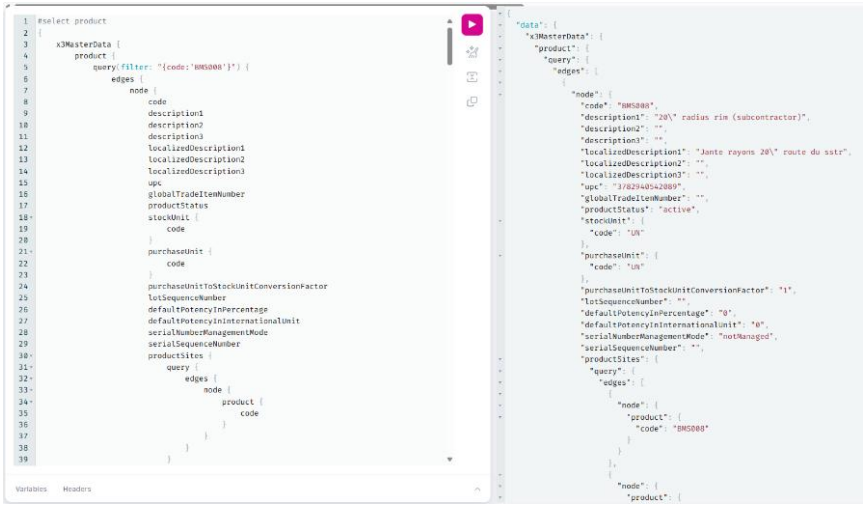


Table de l'opération produit



Query graphQL sur un produit

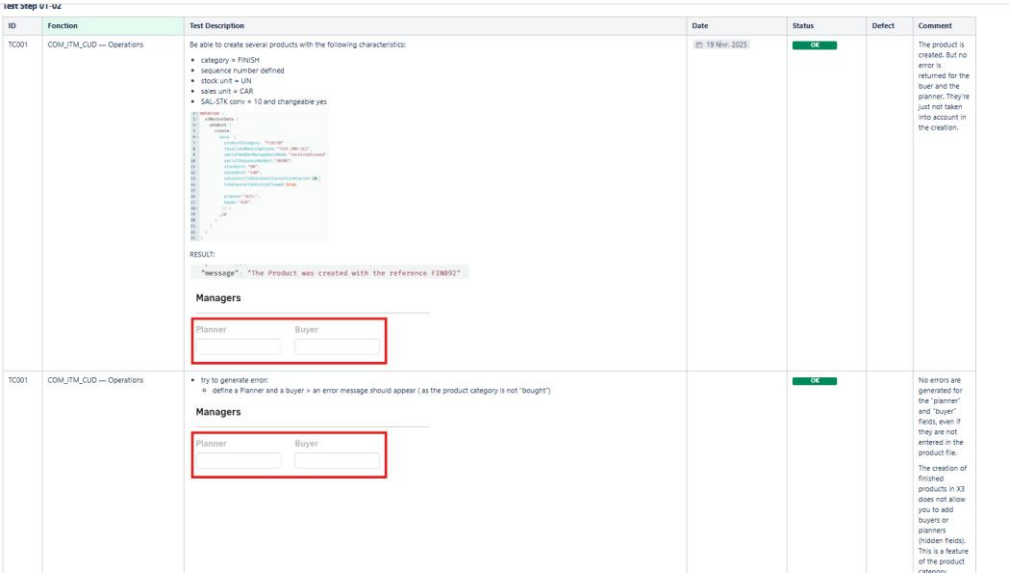
Cette étude de la table m'a permis de mieux comprendre la gestion des produits, bien que cela n'est pas été simple. En effet, n'ayant pas beaucoup de connaissance du secteur logistique, il a parfois été compliqué de comprendre la fonction de certains des champs, mais également de me repérer dans tous les codes et abréviations propres à X3 (Exemple : le champ « *supplier* » est désigné par le code, « *BPSNUM* »). Il faut également noter, que l'usage de l'anglais est de rigueur, même avec le vocabulaire professionnel appris en cours, l'utilisation des mots spécifiques au secteur d'activité logistique ne rendait pas la tâche plus facile. Mais à force de manipuler les produits, d'effectuer les différentes opérations, j'ai réussi à maîtriser la manipulation des produits et à écrire des mutations fonctionnelles.

c) Tests unitaires

Une fois que la mutation est créée, que nous avons observé qu'elle fonctionne, que la création de produit est fonctionnelle, vient l'étape des tests. En effet, lors de l'étape précédente, j'ai pu effectuer des créations de produit, et des premiers bugs ont été observés. Cette phase de test nous permet ainsi de chercher la source des bugs et de les corriger si cela est possible. En revanche, nous ne faisons pas des tests unitaires uniquement pour les bugs détectés à l'étape 2, l'intérêt des tests est aussi de couvrir un maximum de cas. Ainsi, nous pouvons nous assurer de la qualité du développement, soit en trouvant de nouveaux bugs et en les corrigeant immédiatement, soit en observant que tout fonctionne correctement.

Pour ce développement, les tests unitaires consistaient à effectuer les opérations CRUD et à tester un maximum des champs caractéristiques d'un produit. J'ai également dû vérifier les contrôles sur les champs qui en bénéficient (contrôle d'unicité, type de données, accessibilité, dépendances de champs, etc.). Ces contrôles étant définis dans les fichiers sources de X3, j'ai aussi eu à me familiariser avec le langage X3 créé exclusivement pour ce logiciel. Comme dit précédemment les produits peuvent avoir de nombreuses propriétés ce qui rend les tests longs et laborieux. Il était donc important que je sois organisée et que je suive une méthodologie claire et efficace. Pour cela, j'ai pu bénéficier de l'expérience de Xavier, qualiticiens de l'équipe. Il m'a présenté le Test Book, qui est un document regroupant tous les cas de tests effectués. Il est accessible à tous et permet ainsi de s'assurer que toutes les fonctionnalités sont couvertes et conformes aux attentes. Chez Sage, les Test Book ont des modèles standardisés que chaque équipe peut adapter à ses besoins (Les spécifications des équipes sont accompagnées d'une documentation !). Ainsi, nous retrouvons six colonnes qui permettent de guider la rédaction des tests et de faciliter la consultation des tests par tous. Nous retrouvons :

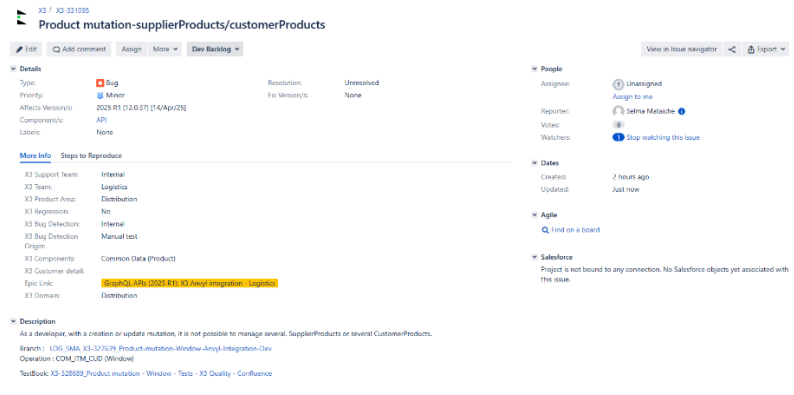
- la **fonction** (l'opération) sur laquelle nous effectuons le test,
- la **description** du test : (comportement attendue, requête effectuée, étapes du test, etc.),
- la **date**,
- le **statut** : OK / KO (réussite du test ou non),
- le "**défaut**" (ce qui pose un problème),
- un **commentaire**.



TestBook Mutation Product

Les tests que j'ai réalisés m'ont permis de trouver plusieurs bugs, qu'il a ensuite fallu corriger. Si la source de certaines erreurs est évidente et peut être corrigée facilement, d'autres peuvent être plus complexe à trouver. Rechercher ce qui pose un problème, lorsque cela peut venir de plusieurs fichiers qui contiennent des centaines de lignes, peut s'avérer compliqué. Une des méthodes consiste à utiliser la fonction de débogage proposée par Eclipse. J'ai principalement utilisé les points d'arrêts, qui permettent de suspendre l'exécution d'un programme en un point donné, et de suivre les étapes d'exécution du programme. J'ai ainsi pu observer quelles instructions posaient un problème, généraient un message d'erreur, etc. Le débogage était une étape longue et fastidieuse. Il fallait redémarrer le serveur assez souvent, ouvrir Eclipse au bon moment ou encore fermer les sessions de travail au fur et à mesure. Toutes ces étapes redondantes rendent les tests peu agréables et ont provoqué une légère baisse de ma

motivation. Le métier de testeur est une vocation et nécessite de toujours avoir en tête l'intérêt du client. Malgré ces difficultés, j'ai pu, avec l'aide de Ouissame et Marie-Ange notamment, trouver la source des bugs. Nous avons ainsi, pu corriger en modifiant les paramétrages des champs via l'opération, ou en modifiant le code source lorsque cela était nécessaire. Néanmoins, certains bugs ne peuvent être corrigés par l'équipe applicative. (Dans notre cas il s'agissait d'un problème pour la gestion de plusieurs fournisseurs ou plusieurs clients d'un produit. Certaines actions, comme la création de plusieurs fournisseurs ou encore la modification, n'étaient pas opérationnelles.) Dans ces cas-là, il faut s'orienter vers les équipes compétentes en créant des tickets de bugs à leur intention (notamment l'équipe plateforme). La création de ces tickets de bugs, ne se fait que lorsque nous avons tout testé de notre côté. Il est important de ne pas diffuser ou transférer des bugs ou autres sans vérification ou à des équipes non concernées, car cela entraîne des pertes de temps, et une baisse de productivités des équipes.



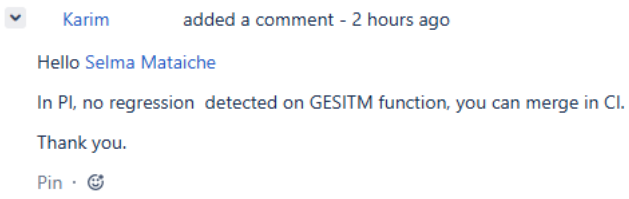
Ticket de bug crée dans Jira

d) Tests automatisés

Une fois les tests unitaires réalisés et les bugs corrigés, nous pouvons débuter les tests automatisés. En réalité, avant de pouvoir me lancer sur les tests automatisés, il a fallu passer par plusieurs étapes. Tout d'abord, installer l'environnement de test. Là encore, toute la documentation nécessaire est disponible sur Confluence. Ainsi, j'ai installé tout ce qu'il fallait en bénéficiant des compétences de Thierry lorsque je rencontrais des difficultés. Lors de ces installations, j'ai également pu utiliser un canal de discussion pour demander de l'aide. En effet, il existe plusieurs équipes Teams, pour différents secteurs, qui permettent aux membres de poser leurs questions, entre autres, et d'obtenir de l'aide rapidement. Pour ma part, j'ai pu exposer un problème que je rencontrais lors de la configuration de mon environnement de test. Avec le grand nombre de personnes ayant accès à ces équipes, j'ai pu obtenir de l'aide rapidement. De plus, étant donné la diversité des équipes, j'ai été en contact avec une jeune diplômée de l'équipe QA, basée en Angleterre à Winnersh. Cette façon de travailler m'a permis de surmonter facilement les difficultés que je rencontrais, mais aussi de rencontrer de nouvelles personnes enthousiastes à l'idée de pouvoir m'aider. (Cette rencontre m'a aussi permis de pratiquer mon anglais et d'échanger avec une jeune diplômée sur son expérience chez Sage.)

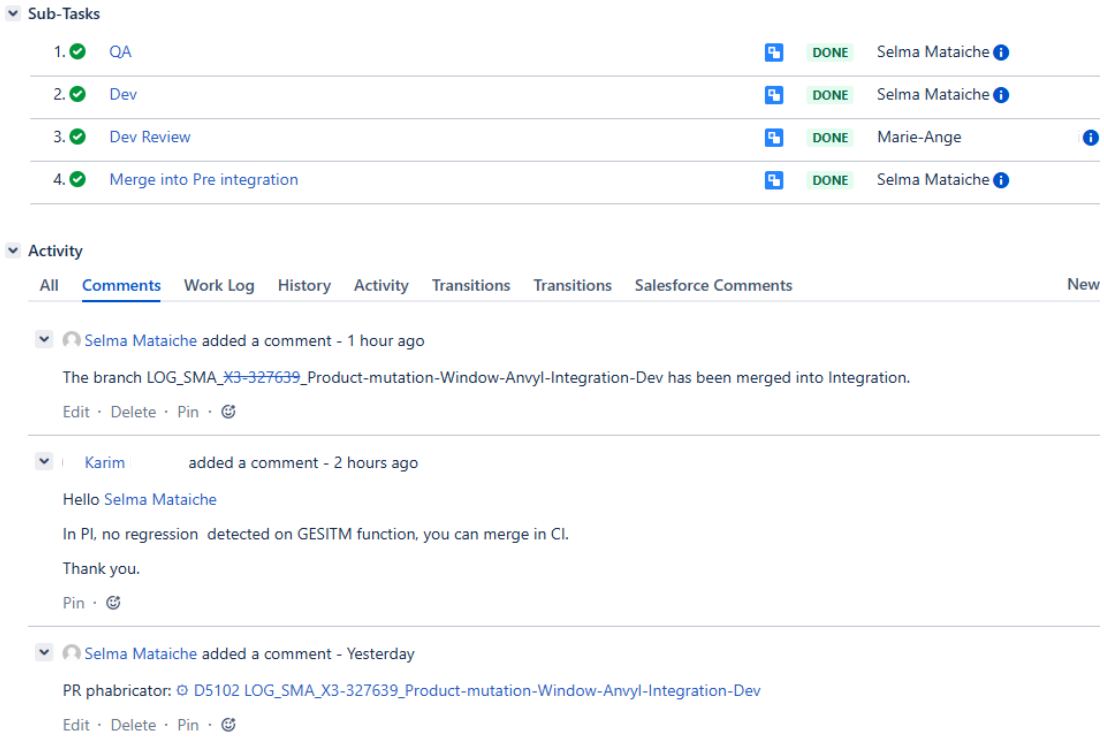
Une fois que tout est installé, il faut encore que le développement réalisé précédemment soit revu par d'autres développeurs, avant de pouvoir passer aux tests automatisés. Pour la revue de code, il faut créer une **PR (Pull Request)**. C'est une demande de comparaison entre ce qui est produit (le nouveau développement) et la branche principale. Alors, d'autres développeurs examinent le code, suggèrent des modifications si besoin, ou approuve la PR, et la fusion du nouveau développement avec la branche principale peut être faite (en réalité, cette première fusion se fait dans une branche nommée « *Pré-intégration* », car elle n'est pas vraiment la branche principale.). Ces PR assurent une meilleure qualité de code et facilitent la collaboration. La dernière étape avant de pouvoir faire les tests automatisés est le contrôle d'un qualiticien de

l'équipe (Karim ou Xavier), pour s'assurer que le nouveau développement ne provoque pas de régression.



Commentaire Jira du qualiticien

C'est uniquement à partir de ce moment que j'ai pu **merger** (Cf. Lexique) dans Intégration mon développement. Il faut noter, que chaque étape doit être suivie, donc un commentaire est ajouté dans Jira au niveau de l'US et dans le canal de discussion de l'équipe pour que tout le monde ait une visibilité sur l'avancement du travail.



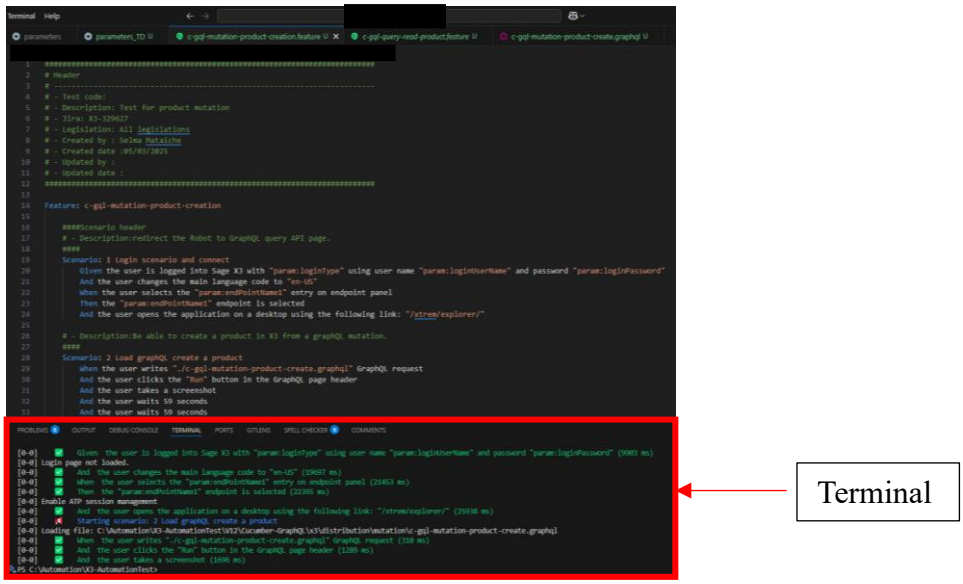
Fiche Jira - Sous-tâches et Commentaires

Concernant les PR, il faut attendre une journée pour que l'image générée par le Docker inclue le nouveau développement (image Docker = modèle pour créer les containers). Cette image est générée chaque nuit ce qui explique que nous ne pouvons effectuer les tests qu'à partir du lendemain.

Là encore même si ces étapes apportent une sécurité pour une bonne qualité de code, elles rendent les livraisons de code un petit peu longue. Pour moi, ce système était nouveau et je n'avais pas conscience de toutes ces étapes dès le début du processus. J'ai donc perdu un petit peu de temps à ce moment-là en pensant après chaque étape que j'allais pouvoir commencer

les tests auto alors que ce n'était pas le cas. Il n'y a rien d'étonnant à cela, car c'était la première fois que j'étais confrontée à cette façon de travailler, mais j'ai maintenant une bonne vision de ces processus et je me sens capable de les remettre en pratique par la suite.

Voyons maintenant, comment effectuer les tests automatisés. Tout d'abord, c'est un test exécuté par un programme ou un robot plutôt que manuellement. Il consiste à simuler des actions utilisateurs. Pour cela, nous utilisons l'outil de test **Cucumber** qui permet d'exécuter des tests automatisés écrit en langage **Gherkin**. C'est un langage de description compréhensible pour les humains et les machines. Il est facile à lire et simple à comprendre, notamment grâce à ses mots-clés (Given, When, Then, And) et au fait qu'il soit écrit en langage naturel. Les tests sont écrits dans des fichiers en « . *feature* » et m'ont permis de tester mes mutations.



Feature de test automatisé

Comme vous pouvez le voir, une *feature* contient plusieurs scénarios qui ont tous un objectif différent. Tout d'abord, le robot se connecte à l'ERP puis est redirigé vers l'API graphQL. Dans le scénario 2, le robot écrit la mutation graphQL qui se trouve dans un autre fichier en « *.graphql »*, puis il exécute la mutation (mutation de création d'un produit, car c'est ce que nous testons.). Les prochains scénarios servent à vérifier la réponse de l'ERP en vérifiant qu'il n'y a pas d'erreurs générées et que les propriétés du produit créé sont bien celles attendues. Pour ces vérifications, et l'écriture de la mutation à tester, le contenu à écrire ou à récupérer depuis l'API est rédigé dans d'autres fichiers. Comme vous pouvez le voir à la ligne 29, ces fichiers sont récupérés grâce à des liens symboliques (chemin à prendre dans l'arborescence des fichiers pour retrouver le fichier souhaité.). Ainsi, il est important pour cette étape de suivre les normes déjà définies pour le nommage des fichiers et leur emplacement dans l'arborescence. Il est important d'avoir une structure cohérente et fixe. En effet, c'est une règle à mettre en place sur tout type de projet pour faciliter son exploitation mais d'autant plus sur des projets aussi grands et manipulé par un grand nombre de personnes. Le fait d'avoir des règles à ce sujet permet à tous de s'adapter rapidement d'un projet à un autre. De plus, nous parlons ici de tests automatisés, qui sont donc exécutés tous les jours automatiquement. Il est donc important de ne pas faire d'erreurs ou de modifications à ce niveau-là, au risque de faire échouer les tests ou de provoquer des erreurs inattendues. Lorsque l'on exécute une *feature*, nous pouvons suivre l'évolution de l'exécution depuis le Terminal où nous voyons le succès ou l'échec de chaque étape des scénarios. Mais, il y a également une page de navigateur qui s'ouvre et grâce à laquelle nous pouvons observer chaque étape que le robot effectue (la connexion avec l'identifiant, le

mot de passe, le changement de langue, etc.), comme si un utilisateur naviguait mais cela est réalisé par le robot sans qu'il soit nécessaire de toucher quoi que ce soit.
L'intérêt de ces tests et aussi de pouvoir les réaliser de manière automatique sans intervention humaine, et de récupérer les résultats par la suite. Cela permet de vérifier à chaque modification de codes que l'application fonctionne correctement. Ainsi, tous les jours à la même heure (environ à 7 h 20), tous les tests automatisés sont exécutés (étant donné le grand nombre de tests existants, cela prend environ quatre heures.). Ces exécutions génèrent des rapports afin de voir le pourcentage de tests réussis, ceux qui ont échoués et surtout pourquoi ils ont échoué. Il est possible de voir à quel moment le test à échouer en regardant l'exécution, parfois même grâce à des captures d'écran prises au moment de l'échec. L'exécution quotidienne des tests permet de détecter rapidement des bugs et de garantir la stabilité du logiciel. Les régressions sont plus facilement détectables, ce qui donne plus de garantie pour livrer un produit stable et de qualité.

La mise en place de ces tests automatisés a été l'une des parties que j'ai préférées. Loin des tests unitaires fastidieux et redondants, les tests automatisés ont comme un aspect plus « ludique » avec l'exécution réalisée par les robots. De plus, le langage utilisé est accessible et simple à comprendre, la structure des *features* l'est aussi ce qui rend les tests plus agréables. Ces caractéristiques permettent de se concentrer plus facilement sur le résultat obtenu, et les potentielles erreurs de développement, sans perdre de temps à vérifier une syntaxe complexe ou autre. Cette expérience me donne même envie de m'intéresser plus en détails à cette spécialité pour la suite de mon parcours.

C. Compétences transversales

En tant que développeuse, les compétences techniques sont importantes, mais ne font pas tous. Des compétences en expression orale et écrite notamment sont appréciées. Que ce soit pour l'écriture de documentation, pour présenter le travail effectué ou encore pour exposer les difficultés rencontrées afin d'obtenir une aide adaptée. J'ai eu à exploiter ses compétences à différents moments. Tout d'abord, lors des DSU. L'objectif de ces réunions étant d'exposer le travail réalisé et celui prévu pour la journée, de manière clair et concise, j'ai travaillé ma capacité à synthétiser et à trier les informations. Ces réunions se déroulaient en français, car tous les membres de l'équipe logistique sont basés en France. Cependant, étant donné le caractère international de l'entreprise, l'équipe réalise une fois par semaine un DSU en anglais pour pratiquer la langue. Ainsi, j'ai travaillé mes capacités en expression orale tant en français, qu'en anglais. Puis, lors d'une sprint review l'équipe m'a confié la responsabilité de faire une démonstration de ce que j'avais développé. Cette présentation est faite à des personnes localisées dans différents pays, et se déroule donc en anglais. Alors, j'ai eu à expliquer le travail que j'avais réalisé et l'accompagner d'une démonstration en direct des opérations CRUD sur un produit, via des mutations graphQL. Cette expérience, bien qu'un peu stressante, m'a permis d'explorer une autre facette du métier de développeuse. J'ai particulièrement apprécié l'exercice de présentation orale, qui m'a apporté un sentiment de satisfaction et de fierté, en voyant que le travail réalisé était reconnu et apprécié. Cette expérience me permet de pratiquer en condition réelle et de confirmer mon appétence pour ce genre d'exercice même en anglais, alors que ce n'est pas la langue dans laquelle je suis la plus à l'aise.

Si l'expression orale est un point fort, l'expression écrite n'en est pas moins indispensable. L'écriture de documentation est essentielle pour communiquer sur ce qui est produit et surtout dans mon cas pour laisser une trace de ce que j'ai fait et permettre à l'équipe

de retrouver facilement ce que j'ai fait même une fois partie. Dans mon cas, la documentation que j'ai créée se limitait principalement à des exemples de mutations GraphQL. Cependant, mon test book peut également être considéré comme une forme de documentation. Et dans ce dernier, l'expression écrite est plus présente, notamment pour l'explication du test en anglais qui doit être claire et reproductible.

Conclusion

C'est ainsi que ce fini ma période de stage. Ce stage a été très enrichissant tant sur le plan professionnel que sur le plan personnel. Comme présenté dans ce rapport, les taches que j'ai pu réaliser sont variées et m'ont permis de découvrir un certain nombre d'aspects du développement logiciel. Durant ces huit semaines, j'ai pu travailler avec des profils variés de développeur et apprendre de leur expérience. J'ai pu moi-même me mettre dans la peau d'une développeuse logiciel en mettant en pratique tout ce que cela implique comme la mise en place et la configuration de l'environnement de travail. Les compétences du BUT que j'ai le plus développées durant cette expérience professionnelle sont la gestion de projet, la gestion de données et de manière plus générale la réalisation d'un développement fonctionnel du développement, aux tests automatisés en passant par la correction de bugs ou encore la création de documentation. J'ai bénéficié d'une expérience extrêmement riche en gestion de projet et application de la méthode Agile. J'ai également eu l'opportunité d'acquérir de nouvelles compétences techniques, en graphQL notamment, de découvrir les bonnes pratiques du développement logiciel et de participer à un projet concret et de grande ampleur. Je sais par avance que je pourrais réinvestir ces compétences lors de mes prochains projets et prochaines expériences. La principale difficulté au début a été de m'adapter à un environnement vaste, où les technologies sont nombreuses et ne m'étaient pas familières, ainsi qu'au langage d'entreprise auquel je n'étais pas habituée. Néanmoins, je pense avoir réussi à surmonter cette difficulté assez rapidement en m'intégrant à cet environnement et à l'équipe. Passer par cette étape m'a permis de développer mes capacités d'adaptation, de surmonter ma timidité en sollicitant les personnes qui m'entouraient, pour avancer. J'ai pu observer que la collaboration et la communication sont les clés pour un travail efficace et de qualité. Ces semaines de travail m'ont également fait prendre conscience que l'environnement humain et ses interactions sont un facteur de réussite bien plus fort que ce que j'imaginais. Pouvoir évoluer dans un environnement de travail où l'échange et l'écoute est encouragé, contribue grandement à l'efficacité d'une équipe et au bien-être de ses membres. La qualité des relations professionnelles est tout autant importante que les compétences techniques. Ainsi, ce stage m'a permis d'évoluer tant au niveau de mon savoir-faire que de mon savoir-être. Je me suis pleinement épanouie à travers cette expérience. Elle a renforcé mon envie de poursuivre une carrière dans le domaine du logiciel.

IV. Lexique

- **AWS** : plateforme cloud complète.
- **Azure DevOps** : plateforme Microsoft permettant de créer, tester et déployer des applications plus facilement.
- **Code composite** : code-barres qui combine **deux types de symboles** en un seul, permettant ainsi d'encoder **à la fois des informations lisibles par un lecteur de codes-barres standard et des données supplémentaires** pouvant être traitées par des systèmes plus avancés (exemple : **GS1 DataBar Composite** (associé au secteur de la distribution et de la pharmacie)).
- **Docker** : c'est un outil open source permettant de livrer, développer et exécuter des applications dans un conteneurs (= ce qui regroupe une application et toutes ses dépendances) Linux. Le Docker permet d'exécuter une application sur n'importe quel système d'exploitation.
- **Eclipse** : IDE : environnement de développement intégré, open-source.
- **Git** : système de gestion de versions permettant de suivre les modifications.
- **Hotfix** : correction rapide pour corriger un bug critique sans attendre la prochaine mise à jour.
- **Integrated Development Environment (IDE)** : environnement de développement intégré (logiciel fournissant des outils pour écrire, tester et déboguer du code efficacement).
- **Keeper** : gestionnaire de mot de passe permettant de gérer est créer des mots de passe de manière sécurisée.
- **Machine Virtuelle** : elle permet d'exécuter une machine virtuelle sur n'importe quel matériel. Elle virtualise toute une machine jusqu'aux couches matériels.
- **Merge** : fusionne une branche dans une autre.
- **Mutation** : requête qui permet de modifier des données. Elle permet de créer, modifier ou supprimer des données dans la base de données.
- **Node** : un node est un élément fondamental d'une structure de données. Il contient généralement **des données** et **des liens** (ou références) vers d'autres nœuds.
- **NodeJS** : environnement permettant d'exécuter du code JavaScript en dehors d'un navigateur.
- Opérations **CRUD** (Create, Update, Delete) : opération de création, de modification et de suppression.
- **Pipeline :** ensemble d'étapes en chaîne, permettant de traiter, transformer ou déployer du code.

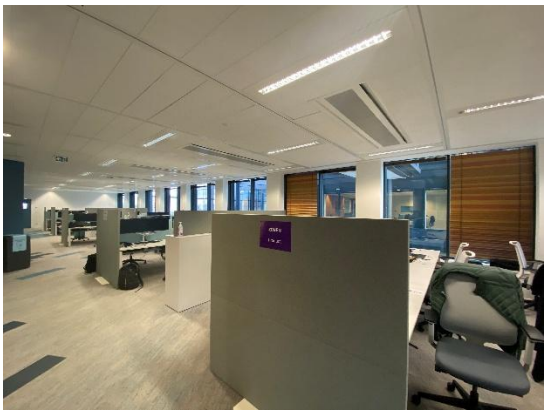
- **Pull Request**: demande de fusion de code dans un système de gestion de version (comme GitHub dans notre cas).
- **Terminal**: interface en ligne de commande qui permet d'interagir avec un système.

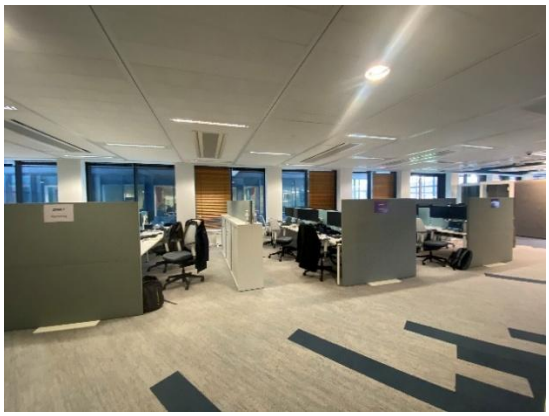
- **WSL** : *Windows Subsystem for Linux* = couche de compatibilité permettant d'exécuter des exécutables binaire Linux directement sous Windows.

Annexes

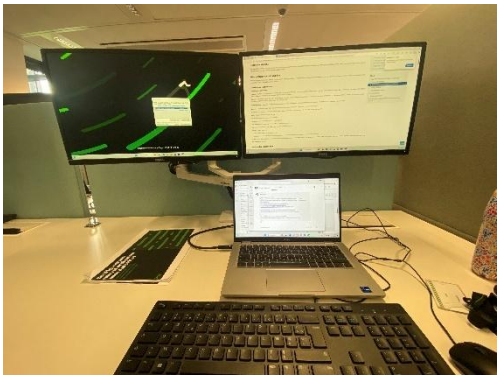
Annexe 1 : Locaux de Sage à La Garenne-Colombes

Annexe 2 : Open-Space



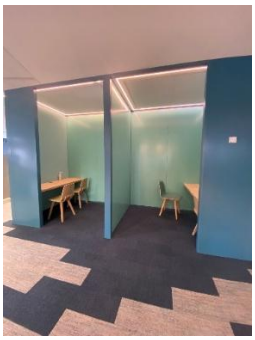
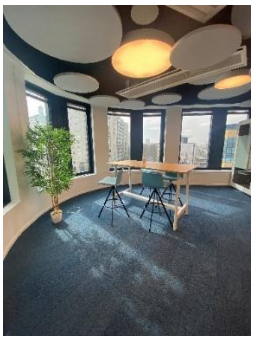


Annexe 3 : Mon poste de travail

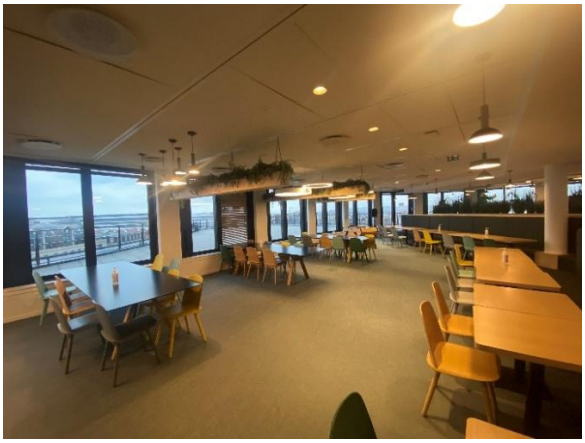
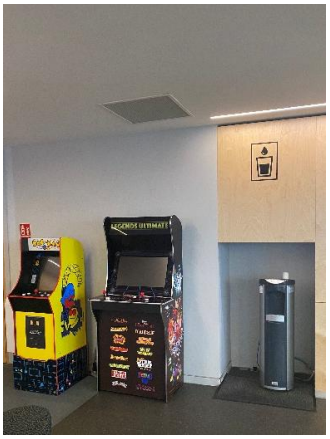


Annexe 4 : Box anti bruit et espace de co-working



Annexe 5 : Espace de pause

Annexe 6 : Coupe du tournoi des 6 Nations