DEPARTEMENT INFORMATIQUE – IUT 2 GRENOBLE

Année Universitaire 2023-2024

MEMOIRE DE STAGE

RAPPORT DE STAGE

Hardis-Group

Présenté par

Matys Marrot-Fourralo

Jury

IUT : M. Blanchon

IUT : Mme. Soualmia

Société : Mme. Juré

Déclaration de respect des droits d'auteurs

Par la présente, je déclare être le seul auteur de ce rapport et assure qu'aucune autre ressource que celles indiquées n'ont été utilisées pour la réalisation de ce travail. Tout emprunt (citation ou référence) littéral ou non à des documents publiés ou inédits est référencé comme tel.

Je suis informé qu'en cas de flagrant délit de fraude, les sanctions prévues dans le règlement des études en cas de fraude aux examens par application du décret 92-657 du 13 juillet 1992 peuvent s'appliquer. Elles seront décidées par la commission disciplinaire de l'UGA.

A Grenoble Le 2 juin 2024

Remerciement

Je tiens à remercier toutes les personnes qui ont pu contribuer de près ou de loin à la réalisation de ce stage et de ce rapport.

Tout d'abord, je remercie chaleureusement Madame Corinne Juré, ma tutrice de stage pour son accueil, ses conseils et son accompagnement tout au long de cette expérience. Son professionnalisme, sa patience et sa disponibilité ont été essentielles pour la réussite de ce stage.

Je tiens également à exprimer ma reconnaissance à tous mes collègues de la R&D Reflex pour leur accueil chaleureux. Leur bienveillance et leur amicalité ont grandement facilité mon intégration. Leurs conseils et leurs expériences m'ont permis de mener à bien ce projet.

Enfin, je remercie ma famille et mes amis pour leur soutien moral tout au long de ce stage.

Résumé long

Hardis Group a été créé en 1984 par Christian Balmain et trois associés. Initialement, la société offrait des services dans l'environnement IBM/38 et des formations en micro-informatique. Rapidement, Hardis s'est diversifié dans l'édition de progiciels, notamment avec la création d'Adélia en 1986. Aujourd'hui, Hardis est une Entreprise de Services Numériques (ESN) qui opère dans quatre domaines : l'édition de logiciels, le conseil, le développement et l'intégration des infrastructures, et le Cloud. En 2023, Hardis comptait 1550 employés, réalisait un chiffre d'affaires de 190 millions d'euros, et avait plus de 1000 clients. Reflex, leur progiciel phare, est utilisé par plus de 350 clients dans 25 pays et a généré 40% du chiffre d'affaires total de l'entreprise.

Le progiciel Reflex WMS est une plateforme de gestion d'entrepôts qui optimise la gestion complète des entrepôts, depuis la réception et le stockage jusqu'à la préparation, l'expédition et le suivi des stocks, tout en intégrant les processus intermédiaires. Le logiciel est modulaire et configurable pour répondre aux besoins spécifiques de chaque client, ce qui lui permet de s'adapter à plus de 10 secteurs d'activités différents, tels que la gestion des matières dangereuses, la pharmacie, la cosmétique et les produits de consommation courante. Le logiciel est principalement développé en Adelia, le langage de programmation propriétaire de Hardis, offrant ainsi une indépendance technologique. Adelia comprend un environnement de développement intégré (IDE) appelé Adelia Studio. Le langage est compilé en Java pour améliorer sa portabilité.

Le projet sur lequel j'ai travaillé concernait la refonte graphique du logiciel Reflex WMS. Cette refonte, débutée il y a deux ans, impliquait la modification des interfaces et le remplacement des anciennes images par des icônes SVG plus élaborées et cohérentes. Pour faciliter cette transition, j'ai utilisé des règles de gestion, ce sont des segments de programme appelés dans le programme principal pour charger les images de manière plus pérenne.

Chaque modification de programme nécessitait la création d'une correction dans Adélia Studio, portant le nom du ticket Jira associé au changement. Une fois les modifications effectuées, j'ajoutais des commentaires dans l'en-tête du programme pour documenter les changements.

Les programmes étaient classés par priorités dans un document Excel, j'ai donc créé une correction par priorité pour éviter de bloquer les programmes trop longtemps. Chaque correction devait être validée par la responsable des designs UX/UI, Caroline FAURE, avant d'être testée en recette par ma tutrice, Corinne JURE.

En cas de conflit sur un programme détenu par quelqu'un d'autre, différentes stratégies étaient adoptées, soit j'attendais que la personne libère le programme soit je modifiais le programme dans sa correction et en recette.

Des réunions régulières, comme les daily hebdomadaires dirigés par Corinne JURE et les plannings Poker mensuels, permettaient de suivre l'avancement des tâches, discuter des problèmes rencontrés et planifier les sprints à venir.

Sommaire

Table des figures

Introduction

Dans le cadre de ma formation de BUT Informatique, il m'était nécessaire de réaliser un stage de 10 semaines au sein d'une entreprise. Hardis Group, acteur majeur des services du numériques en Europe, m'a offert une mission axée sur la refonte graphique de leur produit Reflex WMS, un progiciel* de gestion d'entrepôt.

Reflex, créé en 1992, est un progiciel imposant qui comporte aujourd'hui plus de 15000 programmes uniquement en Adélia, le langage propriétaire de Hardis Group. Parmi tous ces programmes, certains n'ont été que très peu modifiés depuis leur création. Il était nécessaire d'engager une refonte graphique du logiciel Reflex WMS. C'est dans le cadre de ce projet que mes services ont pu être utiles. En effet, depuis 2 ans, Hardis Group et plus précisément le service R&D en charge du développement de Reflex œuvre à retravailler ces interfaces afin d'améliorer l'expérience client. J'ai donc été chargé de modifier les images et icônes d'environ 200 programmes afin de rafraichir l'interface, d'améliorer la compatibilité avec le Web en mettant des images au format SVG et de charger les images de manière plus pérenne en utilisant des règles de gestion.

Pour réaliser ce projet, j'ai dû apprendre à utiliser les outils à ma disposition. La découverte majeure a été celle du langage propriétaire Adélia. C'est dans ce langage que la grande majorité du progiciel Reflex WMS est développé. Il a aussi fallu que je découvre le progiciel Reflex. C'est un logiciel pointu et complexe en raison de sa modularité importante et des détails dans chaque fonctionnalité. En temps normal, les nouveaux associés de la R&D Reflex doivent suivre une formation d'un mois sur le produit Reflex WMS afin de s'en approprier le fonctionnement. Cependant, dans le cadre du stage, il m'était impossible de suivre cette formation. J'ai donc dû apprendre au fur et à mesure du stage l'utilisation complexe du logiciel. J'ai également été initié aux méthodes agiles avec des pratiques telles que les daily et les planning poker dirigés par Corinne Juré.

Dans un premier temps, je présenterai l'entreprise qui m'a accueilli, son histoire, ses produits et services, ses objectifs et son organisation. Dans un second temps, je traiterai l'environnement qui m'a permis de réaliser mon stage dans de bonnes conditions, les outils techniques et les outils de gestion de projet. Enfin, j'expliquerai comment j'ai utilisé tous les outils à ma disposition pour mener à bien ce projet et également tout ce qui est mis en place pour permettre à toute l'équipe R&D de se développer dans un environnement serein. Pour conclure, je résumerai ce que ces 10 semaines de stage au sein d'une entreprise importante telle que Hardis Group m'ont apporté.

I. L'entreprise : Hardis Group

1. *Historique de l'entreprise*

En 1984, Hardis-Group est créé par Christian Balmain et trois associés. Le premier secteur d'activité de l'entreprise est la prestation de service dans l'environnement IBM/38* ainsi que des formations dans le domaine de la micro-informatique* [1]. L'environnement IBM/38*, aussi appelé System/38*, est un ordinateur milieu de gamme conçu par IBM en 1978 pour une utilisation commerciale [2]. La micro-informatique désigne les micro-ordinateurs ce qui correspond aujourd'hui aux ordinateurs personnels. Un micro-ordinateur est composé d'un micro-processeur et de tous les périphériques intégrés (clavier, écran, stockage).

Très vite, la société diversifie ses activités vers l'édition de progiciels* et, en 1986, Hardis décide de créer son propre environnement de programmation : Adélia*. La sortie de l'AS/400* en 1989 va permettre à l'entreprise d'imposer son environnement de programmation sur le marché français [1]. L'AS/400* est l'évolution du System/38* par IBM devenant un ordinateur professionnel plus avancé [2].

À partir de 1992, Hardis se positionne dans le domaine prometteur de la logistique avec la création de leur progiciel* Reflex. Reflex est développé à partir d'Adélia*. C'est un progiciel* extrêmement polyvalent par sa modularité à différents secteurs tels que l'industrie, le pharmaceutique, le e-commerce ou encore les matières dangereuses.

Aujourd'hui, en tant qu'Entreprise de Services Numériques (ESN), anciennement connue sous le nom de Société de Services et d'Ingénierie en Informatique (SSII), Hardis est positionné dans 4 domaines d'activités : l'édition de logiciel ; l'activité de conseil ; le développement et l'intégration des infrastructures ; le Cloud* et l'infogérance. (Voir annexe A)

En 2023, Hardis Group affiche 1550 employés, 39 années de croissance rentable, 190 millions d'euros de chiffre d'affaires, plus de 1000 clients et 16% de croissance. L'entreprise a son siège social à Seyssinet-Pariset, possède des filiales à Lille, Lyon, Nantes, Paris, Bordeaux et est également présente en Espagne, aux Pays-Bas et en Pologne. Le progiciel* Reflex est utilisé par plus de 350 clients dans 1900 sites et 25 pays différents, il est le leader européen dans le domaine. Il a généré à lui seul 78 millions d'euros de chiffres d'affaires en 2023, soit 40% du chiffres d'affaires total de l'entreprise. Reflex est utilisé par des clients tels que Renault, Intersport, Celio, Castorama ou encore Decathlon, ce qui permet à l'entreprise d'être présente à l'internationale. La société est partenaire avec des géants du numérique tels que Google, Microsoft et IBM. (Voir annexe B et C)

2. *Services de l'entreprise*

Le principal service de l'entreprise est le progiciel* Reflex. Reflex est un WMS (Warehouse Management System), c'est un logiciel qui rationalise tous les aspects de la gestion des entrepôts, des processus de réception et de stockage à la préparation, au conditionnement,

à l'expédition et au suivi des stocks, tout en intégrant les étapes intermédiaires. Les avantages de ce type de logiciel pour les clients sont nombreux : il permet d'assurer la traçabilité complète des produits, de la réception des composants jusqu'à l'expédition du produit fini. Il permet de réduire les erreurs qui peuvent être faites lors de la préparation. L'entreprise peut gérer plus efficacement ses ressources humaines et matérielles, elle obtient des informations précieuses en temps réel sur l'état de son activité. Tous ses avantages permettent d'accélérer l'acheminement de tous les produits, réduire les pertes et finalement augmenter les productivités de l'entreprise. Reflex se distingue de ses concurrents par ses 30 années d'expériences. Le logiciel est modulaire et paramétrable pour s'adapter aux spécificités de chaque client. Au total, le logiciel peut s'adapter à plus de 10 secteurs d'activités tels que la gestion des matières dangereuses, la pharmacie et la cosmétique, ou encore les produits de consommation courante.

La firme propose un service de Cloud* avec des applications cloud* natives et des hébergements de données. Le principal avantage du cloud* est de bénéficier d'un accès permanent à tous les logiciels et données qui y sont installés [3]. Le cloud* désigne des serveurs accessibles sur Internet ainsi que des logiciels. Ces serveurs sont hébergés au sein d'un "Datacenter", soit un centre de données souvent coûteux et contraignants à entretenir, il est donc utile pour les entreprises de déléguer ce genre de service.

Hardis propose également des conseils en transformation, innovation et management digital. L'objectif est d'aider les entreprises à sécuriser et à accélérer leurs projets de transformation digitale et d'évolution de leur système d'information. Hardis accompagne ses clients dans le choix des technologies, dans leur mise en œuvre et jusqu'à la conduite du changement de ses équipes [4].

3. Objectifs de l'entreprise

L'objectif principal de l'entreprise est d'accélérer la transformation numérique du commerce, de la supply chain* afin d'en faire un véritable avantage concurrentiel pour ses clients. La supply chain*, qui peut se traduire par chaîne logistique, définit les différentes étapes liées à la chaîne d'approvisionnement, de l'achat des matières premières à la livraison d'un service ou d'un produit fini au client. Cela comprend donc le transport, la distribution, la transformation, la vente, les fournisseurs, le stock de tous les produits finis ou nécessaires à la fabrication du produit et/ou du service final. Le but est d'optimiser toute la chaîne d'approvisionnement afin de réduire les dépenses de l'entreprise et donc le coût final du produit et/ou du service.

4. Organisation de l'entreprise

D'un point de vue organisationnel, l'entreprise est présidée par Nicolas Odet ainsi que le directeur général Yvan Coutaz. L'organisation de l'entreprise ressemble à une organisation divisionnelle subdivisée par activités et par directions fonctionnelles. L'objectif de ce type d'organisation est de se spécialiser par segment stratégique d'activités. Les activités et les directions fonctionnelles sont clairement séparées et autonomes dans leurs domaines respectifs.

Plus spécifiquement, au sein de la R&D Reflex dirigé par Florent Boizard, la structure montre une approche divisée par produits et plateformes. Nous pouvons également retrouver cette structure physiquement. En effet, toute la R&D Reflex se trouve dans le même bâtiment et les divisions au sein du bâtiment sont les mêmes avec l'équipe de documentation du logiciel, l'équipe responsable du développement de l'environnement Adelia et enfin l'équipe de développement du logiciel Reflex WMS. Cette proximité physique permet à toute la R&D Reflex de communiquer facilement et de gagner en efficacité. (Voir annexe D)

II. L'environnement de travail

1. Outils techniques

Le logiciel est Reflex WMS est en partie accessible depuis une interface web. Cela signifie qu'il est possible d'y accéder depuis un navigateur web. Il est en particulier plus adapté à Chrome ou Firefox. Pour le développement, le logiciel tourne sur plusieurs machines virtuelles* : une pour le développement, une autre pour la recette, une pour chaque version encore supportée du logiciel et encore d'autres que je n'ai pas eu l'occasion de découvrir. Veuillez-vous référer à l'annexe E afin de voir comment le logiciel se présente lors de son lancement depuis un navigateur Web. Cependant, le logiciel étant multiplateforme, il y a aussi des programmes dits embarqués qui fonctionnent sur des appareils Android. En effet, des appareils comme les Zebra permettent de scanner des codes-barres avec une sorte de téléphone sous Android. Il est donc important que le logiciel soit supporté sur ce type d'appareil afin de simplifier le travail du personnel travaillant en entrepôt (cariste, réceptionniste...). Il faut savoir que l'entreprise sort 2 mises à jour majeurs par an, soit une tous les 6 mois et qu'elle assure le support d'au minimum les quatre dernières mises à jour, en plus des problèmes urgents (hotfix*) qui sont résolus entre les mises à jour. Dans le domaine de la logistique, cette régularité est importante car c'est un domaine qui bouge très lentement à cause notamment des grosses quantités de données qui sont gérées et de l'impact que peut avoir un dysfonctionnement du logiciel au sein d'un entrepôt. Pour finir, le logiciel est en grande partie développé sous Adelia, le langage de programmation propriétaire de Hardis, ce qui permet à l'entreprise d'être technologiquement indépendante. Le langage Adélia permet également au logiciel Reflex de supporter diverses bases de données : SQL Server, PostgreSQL ou encore Oracle car le langage reste le même quelle que soit la base de données. Avoir un langage propriétaire permet de s'adapter plus facilement aux différents portages. En effet, lorsque les technologies évoluent, les modifications sont directement apportées au langage ce qui permet de ne pas avoir à réécrire les programmes concernés. Le langage Adélia est donc un atout majeur pour Hardis et le logiciel Reflex.

Adelia comprend tout un environnement de programmation propriétaire à Hardis. Il s'agit d'un langage de programmation mais aussi d'un environnement de développement intégré (IDE) Adelia Studio. Le langage ayant été majoritairement développé pour l'environnement de l'AS/400*, il était d'abord compilé en RPG*. Le RPG* est un langage de programmation développé par IBM en 1959. Il est le langage qui a été utilisé pour développer les environnements du System/38* et de l'AS/400*. Par la suite, pour rendre le langage plus polyvalent et utilisable notamment sur les machines Windows et Linux, il a été compilé en C++*. Aujourd'hui, le langage est compilé en Java* afin d'améliorer sa portabilité. Tous ces changements ont permis entre autres de porter de logiciel Reflex sur les différentes plateformes sans modifier les programmes du logiciel. Tout programme Adelia est composé d'un bloc de déclaration et d'un bloc d'initialisation. Il est possible d'ajouter des fenêtres créées à partir de la maquette d'Adelia Studio. Voici la forme typique d'un programme que j'ai eu à modifier. Au début, il y a toujours l'en-tête du programme qui contient l'historique de toutes les modifications qui ont été faites sur le programme depuis sa création. Cet en-tête est dans la

zone de déclaration, il s'agit en réalité de code commenté. La zone de déclaration contient les déclarations de toutes les variables utilisées dans le programme. La zone d'initialisation attribue aux variables les valeurs nécessaires au bon fonctionnement du programme. Dans les programmes que j'ai eu à modifier, il y avait généralement tout le temps une fenêtre voire plusieurs. Une fenêtre est un ensemble d'objet graphique tels que des images, des listes, des boutons, des champs... Un objet graphique peut se voir attribuer un événement, c'est à dire par exemple lorsque le bouton `IDC_CHOISIR` va recevoir un clic gauche de la souris, le code dans l'événement va alors s'exécuter. Enfin, il est possible de créer des fonctions locales au programme afin de factoriser le code. Une fonction possède un bloc de déclaration pour ses variables et notamment ses paramètres.

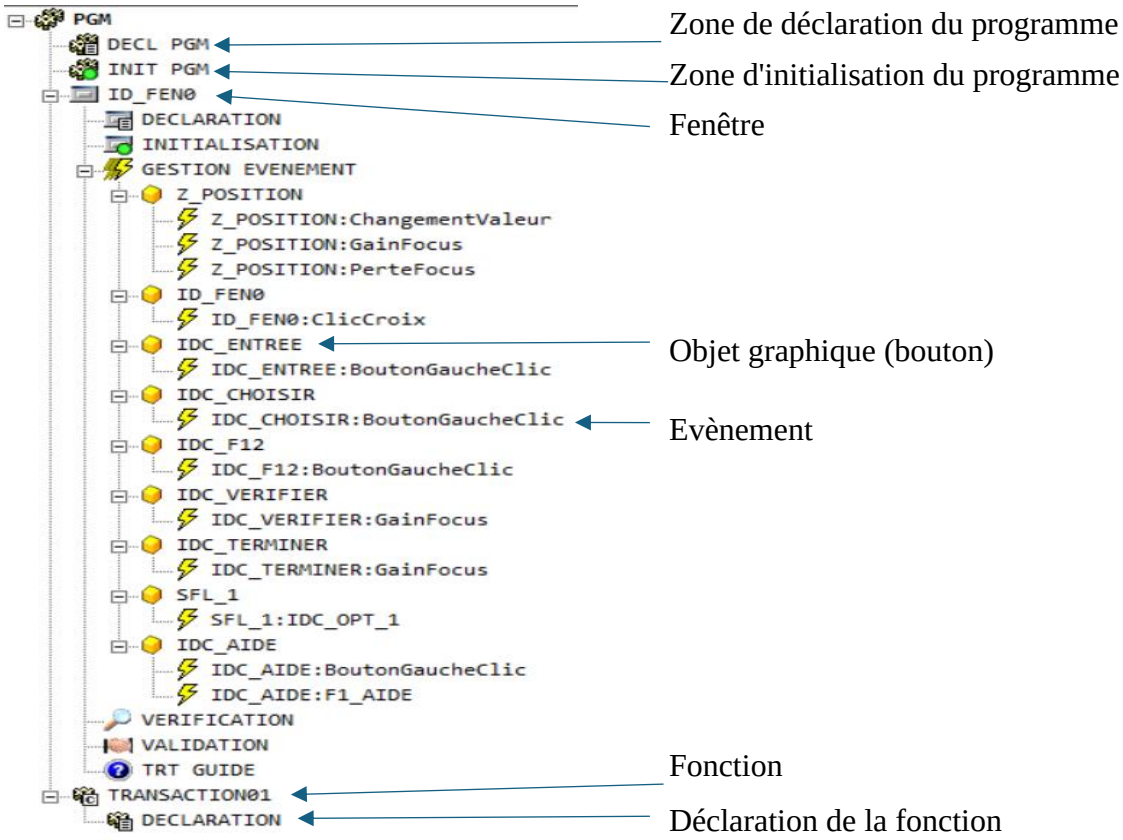


Figure 1 : Différentes parties d'un programme Adélia

Etant donné que git* n'existait pas lors de la création d'Adélia*, Adélia Studio* intègre un système de correction afin de gérer les différentes versions des programmes. Lorsqu'un développeur doit développer une nouvelle fonctionnalité, il doit créer une correction. Un fois sa correction créée, il peut se mettre à l'intérieur et "détenir" des programmes afin de les modifier. Quand un programme est détenu dans une correction qui n'est pas la nôtre, on peut seulement le consulter et non le modifier. Cela engendre des problèmes de productivités parce qu'un seul développeur peut travailler sur un programme et donc il n'est possible d'ajouter qu'une modification à la fois. Git* est tout de même utilisé pour stocker les ressources nécessaires au programmes Adélia* comme les images ou bien les codes qui ne sont pas en Adélia* tel que le CSS, le Python ou le Go.

KListe est un logiciel développé en interne. Parmi toutes ses fonctions, il y a notamment l'arborescence remontante qui permet pour un programme donné d'avoir la liste de tous les programmes qui l'appelle. Cela m'a beaucoup servi pour trouver les programmes que je cherchais car Reflex est un logiciel très complexe avec de nombreux programmes. (Voir annexe F)

Pour accéder à tous ces outils, nous utilisons plusieurs machines virtuelles*. Une machine virtuelle ou VM* est un environnement entièrement virtualisé qui fonctionne sur une machine physique. C'est comme un ordinateur dans un ordinateur. Elle exécute son propre système d'exploitation (OS) et possède les mêmes équipements qu'une machine physique : CPU, mémoire RAM, disque dur et carte réseau. L'avantage d'une machine virtuelle* est de pouvoir faire fonctionner plusieurs VM* sur un seul serveur. Grâce à l'utilisation de machines virtuelles*, chaque application fonctionne dans un environnement dédié, ce qui minimise les risques de conflits et d'interférences. Cette isolation permet une gestion plus efficace des ressources et assure une meilleure sécurité et stabilité des systèmes. Au cours de mon stage, j'ai utilisé 3 machines virtuelles*, bien qu'il en existe d'autres.

La première machine virtuelle* est celle de développement. Elle possède l'environnement Adélia* Studio, KListe et un outil développé en interne qui permet de rendre plus intuitive l'utilisation de Git*. Cet outil facilite la gestion des ressources liées au programme comme les images (Voir annexe G). C'est donc sur cette machine virtuelle* que j'ai passé la plupart de mon temps.

La deuxième machine virtuelle* est celle où tourne le logiciel Reflex en environnement de développement. C'est sur cette machine que nous avons accès au débuggeur. Il permet d'afficher le code compilé avec notamment tous les appels vers les règles de gestion. On peut également mettre des points d'arrêt sur le programme pour analyser les variables et voir les étapes une par une. Le débuggeur est assez complexe à utiliser mais très pratique pour comprendre le logiciel et reproduire certains cas très pointus dans le programme. Sur cette machine, nous pouvons retrouver les images du logiciel, ce qui est particulièrement utile dans le cadre de mon projet.

La troisième machine virtuelle* que j'ai utilisée est celle de l'environnement de recette. Sur cette machine, le logiciel Reflex tourne avec les dernières modifications finies, ajoutées et testées. C'est la version avant la constitution du paquet de livraison. Plusieurs vérifications, déploiements et tests automatiques sont encore réalisés sur une autre machine virtuelle avant la livraison finale.

2. Outils de gestion de projet

Pour organiser toutes ses tâches, l'entreprise utilise Jira, développé par Atlassian. Jira est un outil de développement pour les équipes agiles. Il utilise un système de ticket pour chaque tâche. Une tâche peut avoir des sous tâches et donc des sous tickets. Chaque ticket a un numéro unique. Cela permet de suivre et de gérer efficacement les tâches, les incidents et les bugs rencontrés tout au long du cycle de développement. Jira permet également de mieux planifier et exécuter les sprints afin qu'ils soient structurés tout en visualisant la progression en

temps réel. De plus, chaque ticket possède un lien vers une page Confluence, un outil que décris juste après.

Chaque ticket Jira passe au moins par 4 phases : la qualification ; l'analyse ; le développement ; la recette. La qualification consiste à comprendre et décrire les besoins d'un client. L'analyse consiste à réfléchir à la conception de la solution, de déterminer le choix des technologies à utiliser et écrire une description technique pour réaliser la tâche. Ensuite, on réalise le développement de la solution. Enfin, la recette consiste à vérifier les modifications faites durant le développement, faire des tests unitaires afin de s'assurer qu'il n'y a pas de bugs ou de régressions.

Confluence, également développé par Atlassian, est une plateforme de collaboration et de gestion de contenu conçue pour faciliter la création, le partage et l'organisation de documents et d'informations au sein d'une équipe. Il est l'outil de référence documentaire dans Hardis Group. C'est sur cet outil que nous allons retrouver toutes les informations techniques : guide d'installation, aide au développement, les bonnes pratiques, l'aide en ligne mais aussi des informations RH comme le guide à suivre pour le télétravail, les informations concernant l'entreprise, etc... On peut également y retrouver l'historique des modifications apportées aux programmes à travers les tickets Jira. Chaque ticket Jira est documenté par une page Confluence qui va regrouper toutes les informations le concernant, soit le problème de base, les modifications apportées, les tests réalisés.

Pour communiquer avec tous les membres de la R&D Reflex, nous utilisons Microsoft Teams. C'est un outil particulièrement puissant pour communiquer, notamment lorsque nous ne sommes pas tous physiquement dans les locaux. En effet, Hardis autorise jusqu'à 3 jours de télétravail par semaine, il y a donc rarement tout le monde au bureau.

Tout au long du stage, nous avons utilisé le tableur Excel pour renseigner tous les programmes à traiter ainsi que leur avancement. Il y avait environ 150 programmes à modifier. Il fallait donc les organiser selon leur priorité. Excel et toute la suite Office en général offre la possibilité de collaborer sur les documents, ce qui permet d'avoir plusieurs personnes en même temps qui le modifie.

Sharepoint est une application Web également développée par Microsoft. En tant que plateforme de stockage Cloud, Sharepoint garantit que tous les documents et fichiers sont centralisés et accessibles de manière sécurisée par tous les membres autorisés de l'équipe, réduisant ainsi les risques de pertes de données et les problèmes de versionnage. Les fichiers Excel et les images que nous avons utilisé sont stockés sur cette plateforme.

L'ensemble des outils de la suite Microsoft créent un écosystème intégré qui améliore la communication, la collaboration et la gestion de projets.

Forticlient est un Virtual Private Network (VPN). Il permet d'avoir accès au réseau d'entreprise depuis chez soi par exemple. C'est donc un outil indispensable pour le télétravail.

III. Réalisation

1. *Développement*

Dès mon arrivée, on m'a prêté un ordinateur portable avec la plupart des logiciels nécessaires à mon travail installés. On m'a d'abord donné accès à tout l'environnent Microsoft : la boîte mail professionnelle Outlook, la plateforme de communication Teams, la plateforme de partage de document SharePoint ainsi que toute la suite Office et notamment Excel. J'ai également eu accès au portail d'Hardis Group regroupant toutes les procédures et informations administratives.

On m'a ensuite expliqué le projet. Reflex est un logiciel particulièrement vieux qui nécessitait une refonte graphique (Voir annexe H). Cette refonte graphique a commencé il y a deux ans : les interfaces ont été retravaillées, un mode light a été ajouté, etc... Il fallait alors changer les images par des icônes plus travaillées, plus représentatives et plus cohérentes. Avant, les images étaient au format GIF, PNG ou JPG. Il fallait également modifier le format des images en SVG afin de les rendre plus compatibles avec le web. En effet, les images au format SVG sont redimensionnables sans perte de qualité et également plus légères. Les images étaient chargées en dur dans les programmes ce qui signifiait que leur modification était compliquée. Il fallait entrer dans le programme et modifier directement l'image, tout en sachant que la cohérence avec les autres images n'allait pas être respectée. Pour pouvoir charger les images de manière plus pérenne, nous avons décidé d'utiliser des règles de gestion. Une règle de gestion est un segment de programme que l'on appelle dans un programme parent, avec ou sans paramètre, et qui lors de la compilation vient directement s'insérer mot pour mot dans le programme parent, comme une macro en C par exemple.

```
* Modif MMARROT 16/04/24 - RD-90192
INSERER_RG HF_IMG_UTIL(ID_FEN0; IMG_PERSONNE_CALCUL)
INSERER_RG HF_IMG_UTIL(ID_FEN0; IMG_PERSONNE_CREATION)
INSERER_RG HF_IMG_UTIL(ID_FEN0; IMG_PERSONNE_MAJ)
* APPELER_METHODE ID_FEN0.IMG_PERSONNE_CALCUL    CHARGER_BITMAP NomFichier
* APPELER_METHODE ID_FEN0.IMG_PERSONNE_CREATION CHARGER_BITMAP NomFichier
* APPELER_METHODE ID_FEN0.IMG_PERSONNE_MAJ      CHARGER_BITMAP NomFichier
* Fin modif MMARROT 16/04/24 - RD-90192
```

```
* Modif MMARROT 16/04/24 - RD-90192
* INSERER_RG HF_IMG_UTIL(ID_FEN0; IMG_PERSONNE_CALCUL)
*-----------------------------------------------------------*
*> Chargement de l'image IMG_UTIL                          *
*-----------------------------------------------------------*
* Créé par IG le 09/04/24 - RD_90192                       *
*-----------------------------------------------------------*
* 01 nom de la fenêtre
* 02 nom de l'objet image dans la fenêtre
*
* INSERER_RG HF_INIT_CHG_IMG
*-----------------------------------------------------------*
*> Chargement des images - Initialisation                  *
*-----------------------------------------------------------*
* Créé par IG le 09/04/24 - RD_90192                       *
*-----------------------------------------------------------*
*
SI X_CHG_IMG_RECUP_CUR_DIR = *BLANK
  X_CHG_IMG_lg = 256
  APPELER_DLL 'HFToolBox' 'HFTB_GetCurrentDirectory' X_CHG_IMG_CurrentDir X_CHG_IMG_lg
  X_CHG_IMG_RECUP_CUR_DIR = '1'
FIN
*
SI X_CHG_IMG_THEME = *BLANK
  APPELER_DLL 'HFToolBox' 'HFTB_GetWagonCurrentTheme' X_CHG_IMG_THEME
FIN
********** fin de la règle de gestion HF_INIT_CHG_IMG **********
*
X_CHG_IMG_NomFichier = X_CHG_IMG_CurrentDir /// '\IMG_UTIL.svg'
APPELER_METHODE ID_FEN0.IMG_PERSONNE_CALCUL CHARGER_BITMAP X_CHG_IMG_NomFichier
*
********** fin de la règle de gestion HF_IMG_UTIL **********
```

Figure 2 : Exemple d'un programme avant et après l'insertion d'une règle de gestion

Dans le cas où l'on veut modifier une image, ce mode de chargement permet de le faire très facilement en modifiant juste le nom de l'image dans la règle de gestion et de recompiler tous les programmes utilisant cette règle de gestion. Nous avons donc commencé par faire une règle de gestion par image, voici le code d'une règle de gestion :

```
INSERER_RG HF_INIT_CHG_IMG
X_CHG_IMG_NomFichier = X_CHG_IMG_CurrentDir /// '\IMG_CROIX.svg'
APPELER_METHODE :01.:02 CHARGER_BITMAP X_CHG_IMG_NomFichier
```

Cette règle de gestion permet de charger l'image d'une croix rouge. Elle appelle une autre règle de gestion qui attribue le lien vers le répertoire qui comporte les images à la variable X_CHG_IMG_CurrentDir, ensuite nous ajoutons au lien vers le répertoire le nom de l'image afin d'obtenir le lien vers l'image. Enfin, nous appelons la méthode CHARGER_BITMAP de l'objet graphique passer en paramètre :01.:02 . Ici :01 permet d'appeler le premier paramètre et :02 le deuxième. En premier paramètre nous plaçons le nom de la fenêtre où se trouve l'objet graphique et en deuxième paramètre le nom de l'image ou du bouton, ce qui donne dans le programme parent un appel comme ceci :

```
INSERER_RG HF_IMG_CROIX (ID_FEN0; OBJET_IMAGE)
```

Avant d'appeler cette règle de gestion dans un programme, il faut initialiser les variables tels que le nom de l'image, le nom du répertoire courant avec une autre règle de gestion. Pour initialiser des variables en Adélia*, il suffit de mettre le type de la variable et son nom :

```
ALPHA(256)  X_CHG_IMG_CurrentDir
ALPHA(256)  X_CHG_IMG_NomFichier
BOOL        X_CHG_IMG_CodeRetour
NUM_BIN_2   X_CHG_IMG_lg
```

Avant de modifier un programme, il faut créer une correction dans Adélia Studio* pour que pouvoir détenir le programme et le modifier. La correction doit porter le même nom que le ticket Jira, soit RD_XXXXX. Une fois ceci fait, nous pouvons détenir le programme pour changer les images. Dès lors que nous apportons une modification dans un programme, nous devons compléter son en-tête avec du code commenté contenant notre nom, la date de modification, le numéro du ticket Jira et une brève description de ce que nous faisons, comme ceci :

```
*-------------------------------------------*
* Modifié par MMARROT le 15/04/24 - RD_90192 *
*     Chargement des images via l'utilisation*
*     de règles de gestion                  *
*-------------------------------------------*
```

Ensuite, il faut appeler la règle de gestion qui déclare les variables nécessaires au chargement de l'image. Quand nous modifions, ajoutons ou supprimons une partie du programme, il faut aussi annoncer ses modifications avec son nom, la date et le numéro du ticket Jira, comme dans l'en-tête.

```
* Ajout MMARROT le 15/04/24 - RD_90192
INSERER_RG HF_DCL_CHG_IMG
* Fin ajout MMARROT le 15/04/24 - RD_90192
```

Maintenant que les variables sont déclarées, nous pouvons charger l'image en utilisant la règle de gestion. Les images peuvent être chargées de différentes manières : soit depuis la maquette soit depuis le programme. Si l'image est chargée depuis la maquette, il n'y a alors aucune trace de l'image dans le programme. Il faut donc utiliser la règle de gestion dans la zone d'initialisation de la fenêtre. En revanche, si l'image est chargée dans le programme, il suffit de trouver où et de la charger avec la règle de gestion au même endroit. Les images peuvent être chargées sur différents objets graphiques : il peut y avoir des images sur des boutons, sur les barres d'outils, dans les listes, dans des listes arborescentes. Toutes ces images sur des objets graphiques sont chargées grâce à des fonctions qui n'ont pas le même nom et qui ne prennent pas les mêmes paramètres. Donc charger une image sur un bouton, dans une liste ou dans une barre d'outils requiert des règles de gestion différentes. En règle générale, nous n'avons pas eu à traiter les images dans les barres d'outils car elles étaient déjà chargées correctement. Cependant, nous avons dû modifier les icônes, les listes, les boutons et quelques listes arborescentes.

Lorsque nous chargeons les images avec la règle de gestion, il faut veiller à ce que le code ne soit pas exécuté côté serveur. En effet, en Adélia* il est possible de déterminer quelle partie du code est exécutée côté serveur et quelle partie du code est exécutée côté client. Cela permet d'optimiser les échanges entre client et serveur et donc de gagner des performances. Par défaut, le programme est exécuté côté client sauf lorsqu'il a besoin du serveur. Imaginons qu'il y ait deux requêtes vers la base de données d'affilée, par défaut le programme va faire deux échanges avec le serveur, alors que si nous mettons les deux requêtes côté serveur, le programme n'aura qu'un seul échange entre le client et le serveur. Dans le cadre du chargement des images, il faut que la règle de gestion soit exécutée côté client car c'est un écran qui permet l'interface avec l'utilisateur. Il arrive que l'initialisation de la fenêtre soit faite côté serveur et lorsque nous ajoutons la règle de gestion, cela génère une anomalie. Il faut alors veiller à remettre la règle de gestion côté client tout en ne provoquant pas d'aller-retour inutile.

Pour tester les modifications que nous avons effectuées sur un programme, nous commençons par compiler le programme afin de tester qu'il n'y ait pas de problèmes. Les erreurs qu'il peut y avoir sont souvent des erreurs d'inattention. Par exemple, il peut y avoir une erreur parce que la règle de gestion est appelée côté serveur, parce que nous avons oublié d'appeler la règle de gestion qui déclare les variables, ou tout simplement parce qu'il y a eu une faute de frappes en appelant les règles de gestion. Si le programme compile correctement, nous pouvons alors lancer l'exécution. Une fois l'exécution terminée, nous allons sur l'interface Web de Reflex afin d'appeler le programme et voir les modifications effectuées. Sur l'interface Web, le seul problème qu'il peut y avoir est l'image qui ne s'affiche pas, cela est généralement dû à une erreur dans son nom.

Sur le document Excel, les programmes étaient triés par priorités. Nous avons créé une correction par priorité pour éviter de bloquer les programmes trop longtemps. En effet, pour modifier les programmes, nous sommes obligés de les mettre dans notre correction. Tant que

les programmes sont dans notre correction, il est impossible pour les autres de travailler dessus. Etant donné que le projet allait durer plusieurs semaines, il était inconcevable de bloquer une centaine de programmes pendant tout ce temps. C'est pourquoi nous avons séparé la tâche en 5 priorités que nous réalisions une par une. Une fois tous les programmes d'une priorité traités, nous pouvions libérer la correction, libérer les programmes et passer à la priorité suivante.

Pour livrer les images dans le produit, nous utilisons Gitlab. Pour faciliter l'utilisation de Git, nous avons un outil interne à l'entreprise qui permet en quelques mouvements d'envoyer nos ressources sur le serveur Gitlab tout en assurant la traçabilité (voir annexe G). Lorsque nous lançons l'outil, nous devons renseigner notre numéro de ticket Jira pour assurer la traçabilité. Ensuite, nous cliquons sur "Commencer le développement", cela va créer une branche Git et ramener (clone) le projet sur notre machine. Nous pouvons ensuite ajouter nos images dans le répertoire. Ensuite, il suffit de cliquer sur "Terminer le développement", il faudra renseigner le message du commit puis toutes nos modifications seront ajoutées à Gitlab. Pour lancer la recette, il faut alors cliquer sur "Commencer la recette" et toutes les ressources que j'ai apportées dans mon ticket serons alors envoyées sur la machine virtuelle de recette afin d'effectuer les tests.

2. Validation

Pour qu'une correction soit validée, elle devait d'abord être validée par Caroline FAURE, responsable des designs UX/UI. Elle était chargée de vérifier que l'image avait les bonnes dimensions, qu'elle était alignée avec le texte et qu'elle avait la bonne signification. Il est arrivé que, à la suite de sa demande, nous ayons à modifier les dimensions de l'image depuis la maquette, à changer sa position ou tout simplement la supprimer.

Lorsqu'une correction était validée, ma tutrice Corinne JURE s'occupait de passer les programmes en recette afin de vérifier qu'il n'y avait pas de problème. Lorsque la correction était passée en recette, elle changeait de machine virtuelle*. Les programmes détenus dans cette correction étaient alors libérés sur la machine de développement et étaient donc disponibles pour les autres.

Lorsque nous devions modifier un programme, il est arrivé qu'il soit déjà détenu par un autre développeur. Il y avait alors plusieurs cas. Premier cas, la personne n'avait pas modifié le programme et n'avait pas besoin de le modifier, elle pouvait alors annuler la mise en correction afin que nous puissions le détenir. Deuxième cas, la personne avait modifié le programme et devais conserver ses modifications, nous devions alors nous mettre dans sa correction afin de modifier le programme tout en conservant ses modifications. Si la personne était sur une modification importante et nécessitant beaucoup de temps alors nous devions aller sur la VM* de recette afin de reproduire nos modifications en recette. Cela permettait à Corinne de pouvoir tester nos modifications sans transférer la correction de la VM* de développement à la VM* de recette. Si les modifications ne nécessitaient peu de temps, soit nous faisions nos modifications dans sa correction et la recette des images était faite en même temps que le sienne, soit nous attendions qu'elle fasse sa recette pour qu'elle libère le programme afin que

nous puissions le détenir dans notre correction et faire nos modifications. Vous trouverez en annexe J le logigramme représentant tout le processus de correction.

3. Documentation

Après avoir modifié tous les programmes qui avaient une image, il a fallu que nous mettions à jour la documentation. Il existe deux documentations : une externe pour les clients et les intégrateurs et une autre interne pour les développeurs du logiciel. Les intégrateurs sont des entreprises qui sont payés par Hardis pour vendre le logiciel et le mettre en place chez le client. Les intégrateurs peuvent également concevoir des programmes sur mesure pour les clients. IBM est un de nos intégrateurs.

Pour modifier la documentation externe, il fallait se rendre sur la page Confluence appropriée et la modifier. Cela allait en réalité dupliquer la page Confluence afin d'enregistrer l'ancienne tout en modifiant la copie. Ensuite, nous devions mettre le lien de la page Confluence dupliquée et modifiée sur le ticket Jira correspondant. Le service documentation allait ensuite pouvoir valider et traduire la documentation modifiée. Sur cette documentation, nous avions seulement rajouté qu'il était préférable d'utiliser des images au format SVG afin d'éviter la déformation des images. Il n'était pas nécessaire d'expliquer les règles de développement mise en place car ni les clients ni les intégrateurs n'avaient accès au code et au règle de gestion.

Pour ce qui est de la documentation interne pour les développeurs Hardis, nous pouvions modifier directement la page Confluence sans la faire valider. Ici, nous devions expliquer le fonctionnement avec les deux règles de gestion, le dimensionnement des images et les règles de nommage pour les règles de gestion.

4. Méthodes agiles

Les méthodes agiles, particulièrement adoptées dans le domaine de l'informatique et du développement logiciel, se caractérisent par une approche itérative et collaborative de la gestion de projet. Elles visent à offrir une meilleure flexibilité et une adaptation rapide aux changements, tout en communicant et en collaborant au sein de l'équipe.

Parmi ces méthodes, nous trouvons les sprints. Ce sont des itérations courtes (4 semaines) qui permettent la livraison fréquente de versions du logiciel et qui assurent que le produit développé répond bien aux besoins des clients. Ce principe de sprint assure la transparence car l'avancement de chacun est visible, la responsabilisation de l'équipe car chacun a ses propres tâches, et l'amélioration continue en prenant en compte l'avis de tous à la fin du sprint.

Les sprints sont rythmés par plusieurs réunions, notamment les daily meetings et le planning poker. Ces réunions sont essentielles pour le bon déroulement des projets. Les daily ont lieu tous les mardis et vendredis matin dans l'équipe Vercors, dirigée par Corinne JURE. Ces réunions permettent à chaque membre de l'équipe de partager son avancement, d'exposer les obstacles rencontrés et de définir les priorités jusqu'au prochain daily. Cette pratique

favorise la transparence, permet de détecter rapidement si quelqu'un est en difficulté et encourage la collaboration et la communication entre les membres de l'équipe.

Le planning poker a lieu toutes les quatre semaines. Cette réunion permet de planifier les tâches des 4 prochaines semaines. Durant ce rendez-vous, les tickets Jira sont distribués à chacun en fonction de leurs préférences et de leurs compétences. Nous parlons également des activités en cours et futures, de nos ressentis par rapport au dernier sprint afin d'identifier les potentiels problèmes et de s'améliorer continuellement. Les performances de l'équipe sont également abordées avec le nombre d'heures travaillées, des graphiques générés par Jira montrant l'évolution de la complétion de tickets, ou encore le nombre de tickets restants ou pas.

L'outil Jira est également important pour la mise en place de méthodes agiles. Le backlog Jira est un tableau regroupant tous les tickets qui nous ont été attribués. Vous pouvez retrouver dans l'annexe I le backlog de ma responsable Corinne Juré. Dans la colonne de gauche "To do", il y tous les tickets qui nous sont attribués mais qui ne peuvent pas être commencés car les tâches antérieures n'ont pas encore été validées. Ensuite, dans la colonne "Ready to start", il y a les tickets où le développement peut être commencés. Dans la colonne "In progress", il y a les tickets commencés mais pas finis. Enfin, dans la colonne "Done", il y a les tickets finis. Sur chaque ticket, il est possible de saisir son temps. La barre bleue indique le temps prévu pour faire cette tâche, la barre verte le temps passé à travailler sur cette tâche et la barre orange le temps qu'il est censé rester.

Ainsi, les méthodes agiles et leurs réunions régulières, comme les daily et le planning poker, poussent l'équipe à collaborer, à s'entraider et améliorent la communication. L'outil Jira est parfait pour organiser l'ensemble des tâches auprès de toute l'équipe. S'informer sur les ressentis de chacun est essentiel pour pouvoir s'améliorer continuellement. Cela optimise la productivité et l'efficacité de chacun, tout en garantissant un milieu de travail serein.

Conclusion

Au terme de ce stage de dix semaines au sein de l'entreprise Hardis Group, j'ai eu l'opportunité de me plonger dans un environnement professionnel et formateur, qui m'a permis de développer à la fois des compétences techniques, des compétences relationnelles et des compétences personnelles, essentielles pour mon futur dans le domaine de l'informatique.

L'expérience acquise lors de ce projet de refonte graphique du produit Reflex WMS a été particulièrement enrichissante. J'ai pu appréhender les enjeux et les défis liés à la gestion et à la maintenance d'un progiciel* de gestion d'entrepôt utilisé par de nombreuses entreprises. Cette mission m'a permis de me familiariser avec des technologies et des outils spécifiques tels qu'Adelia Studio et Jira, et d'appliquer des méthodes de gestion de projet agiles, notamment à travers les plannings poker et les daily.

Travailler en collaboration avec l'équipe R&D Reflex, sous la direction de Mme Corinne Juré, m'a offert une perspective précieuse sur l'organisation interne et les dynamiques de travail au sein de Hardis Group. J'ai pu observer l'importance de la communication et de la coordination au sein d'une équipe pour le succès des projets. Les séances de feedback régulières sont cruciales pour identifier et résoudre rapidement les problèmes, ainsi que pour améliorer continuellement les processus de travail.

L'utilisation d'Adelia, un langage de programmation propriétaire, a été une découverte significative. Adelia, avec son environnement de développement intégré (IDE) Adelia Studio, est un outil puissant et flexible, malgré sa complexité. Comprendre et manipuler ce langage m'a permis d'acquérir une autonomie, renforçant ma capacité à gérer des projets de développement logiciel.

De plus, l'expérience de travail au sein d'une entreprise comme Hardis Group m'a exposé à des pratiques avancées, telles que la gestion des versions avec d'autres technologies que Git* et des corrections dans un environnement de développement collaboratif. Cette immersion m'a aidé de développer une compréhension approfondie des processus de développement logiciel et de la gestion des infrastructures nécessaires pour soutenir un progiciel* complexe comme Reflex WMS.

Sur un plan personnel, ce stage a renforcé ma capacité à m'adapter à des situations nouvelles et parfois complexes. J'ai appris à gérer mon temps et mes priorités, à travailler et à trouver des solutions aux problèmes rencontrés. Cette expérience m'a également donné l'opportunité de tisser des liens professionnels avec mes collègues, enrichissant ainsi mon réseau et ouvrant de nouvelles perspectives pour mon avenir professionnel.

Pour conclure, ce stage chez Hardis Group a été une expérience formatrice dans mon parcours universitaire et professionnel. Il m'a permis de mettre en pratique mes connaissances théoriques et d'acquérir de nouvelles compétences. Je suis convaincu que les enseignements tirés de cette expérience me seront d'une grande utilité dans la poursuite de mes études et dans ma carrière professionnelle et personnelle à venir.

Glossaire

IBM/38 ou System/38 : ordinateur milieu de gamme conçu par IBM en 1978 pour une utilisation commerciale

Micro-informatique : désigne les micro-ordinateurs ce qui correspond aujourd'hui aux ordinateurs personnels

Progiciel : logiciel généraliste paramétrable

AS/400 : évolution du System/38 par IBM devenant un ordinateur professionnel plus avancé

Cloud : serveur accessible sur Internet permettant de stocker des données et/ou d'héberger des logiciels

Supply chain : différentes étapes liées à la chaîne d'approvisionnement, de l'achat des matières premières à la livraison d'un service ou d'un produit fini au client

Adélia : langage de programmation propriétaire à Hardis Group

Adélia Studio : environnement de développement intégré (IDE) utilisé pour la programmation en langage Adélia

RPG (Report Program Generator) : langage de programmation développé par IBM en 1959, utilisé pour les systèmes IBM comme le System/38 et l'AS/400.

C++ : langage de programmation orienté objet

Java : langage de programmation de haut niveau, orienté objet, connu pour sa portabilité

Git : logiciel de gestion de versions

Machine Virtuelle (VM) : logiciel qui émule un ordinateur physique et exécute des programmes comme un ordinateur réel

Débuggeur : outil qui permet de tester et de déboguer des programmes, en affichant le code compilé et en permettant d'analyser les variables et de suivre les étapes du programme

Hotfix : correctif urgent apporté à un logiciel pour résoudre un problème sans attendre la prochaine mise à jour majeure

Sitographie

[1] Page Wikipédia de Hardis Group : https://fr.wikipedia.org/wiki/Hardis (consulté le 28/05/2024)

[2] Blog de l'entreprise Fortra sur l'histoire d'IBM : https://www.fortra.com/fr/blog/histoire-et-chronologie (consulté le 30/05/2024)

[3] Site officiel de Hardis Group, onglet "Solutions Cloud" : https://www.hardis-group.com/services-solutions/services/solutions-cloud-applications-cloud-native-et-hebergement-en-cloud (consulté le 01/05/2024)

[4] Site officiel de Hardis Group, onglet "Transformation Digitale" : https://www.hardis-group.com/services-solutions/services/transformation-digitale (consulté le 01/05/2024)

Résumé :

L'entreprise Hardis Group est un acteur majeur en Europe dans le domaine des Entreprises de Services du Numérique. Le produit phare de Hardis est Reflex WMS, un progiciel de gestion d'entrepôt développé en Adélia, langage propriétaire de l'entreprise. C'est dans le cadre de la refonte graphique du progiciel Reflex WMS que j'ai été chargé de modifier les images du logiciel afin de rafraichir son design. Cette refonte graphique était l'occasion de changer le format des images en format SVG pour les rendre plus compatible avec le Web mais aussi de les charger de manière plus pérenne. Pour aboutir à ce résultat, nous avons utilisé les règles de gestions, des segments de code qui sont directement insérés dans celui-ci une fois compilé. Pour charger une image à partir d'une règle de gestion, il faut initialiser les variables nécessaires grâce à une autre règle de gestion puis utiliser la règle de gestion qui charge l'image en passant en paramètre l'objet dans lequel on veut charger l'image. Pour s'assurer du résultat, il suffit d'afficher le programme sur l'interface Web de Reflex et de vérifier que l'image est bien apparente.

Mots-clés : Hardis Group, Reflex WMS, Adélia, Image, Refonte graphique, Règle de gestion, Programme, Design.

Abstract :

Change of Images for a Graphic Redesign of Reflex WMS

Matys Marrot-Fourralo

Abstract: Reflex WMS is a 30 year old software. It was necessary to graphically rethink the software to better match customer expectations. In this project, the images had to be changed. The objective is to change the images to refresh the software, modify the image format to SVG to be more responsive and use management rules to load the images in a more sustainable way. Management rules are like macro in C. They are segments of code that are directly inserted into the program but are declared out of the program. This allows you to factor the loading of images into a single program which is called by other programs that want to use the image. To change images in the program, it's necessary to comment out the old image loading. Then, you must use management rules to initialize the variables used to load the image. Finally, you call the management rules that load the image with the parameter in which you want to load the image. To test, you view the program in the web interface and verify that the image has changed. Now, if you want to change an image, you just have to modify the management rule and reload the program.

Keywords: Image, Management rule, Reflex WMS, Software, Graphic design, Interface, Program.

Annexe

1. *Annexe A : Positionnement de l'entreprise Hardis Group*

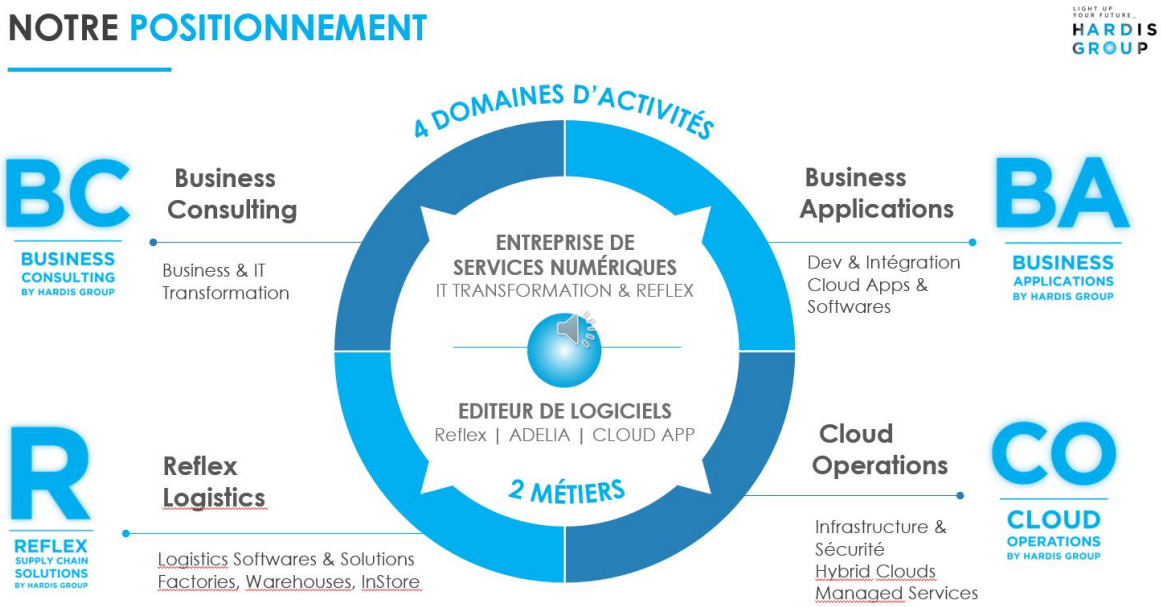


Figure 3 : Positionnement de l'entreprise Hardis Group

2. *Annexe B : Bilan de l'entreprise Hardis Group pour l'année 2023*

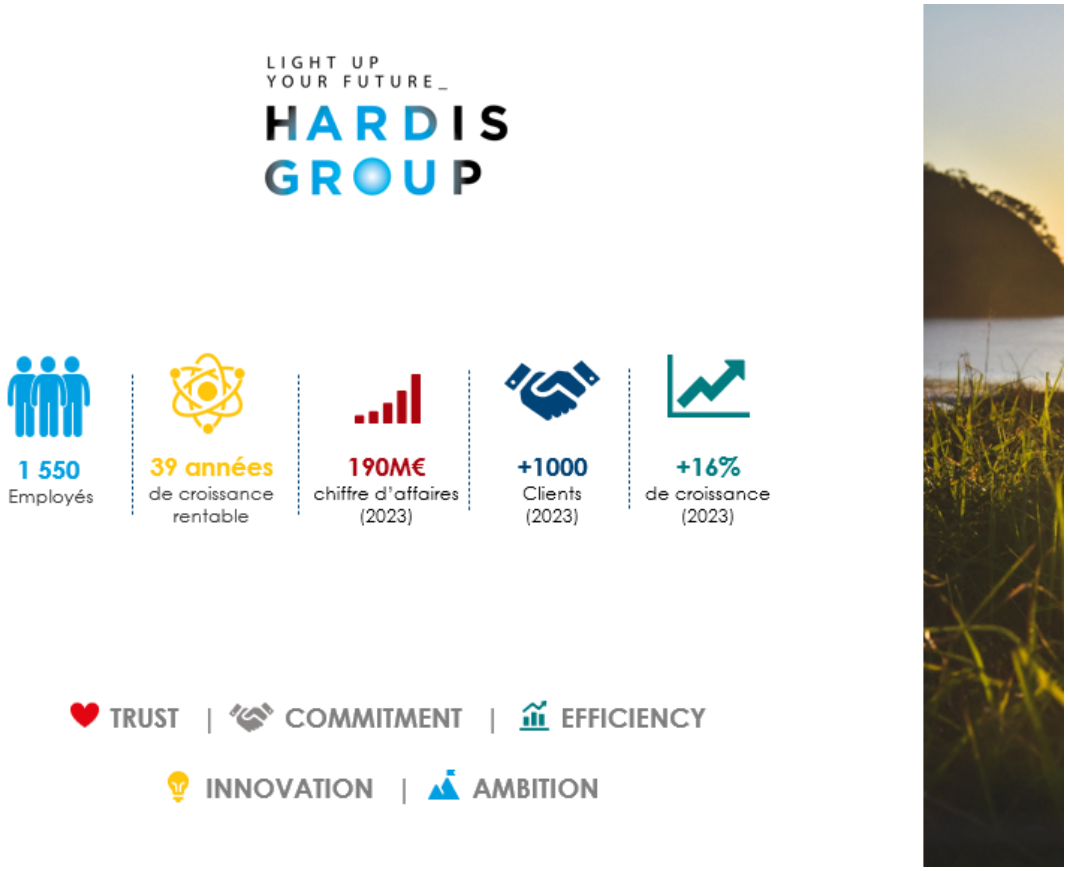
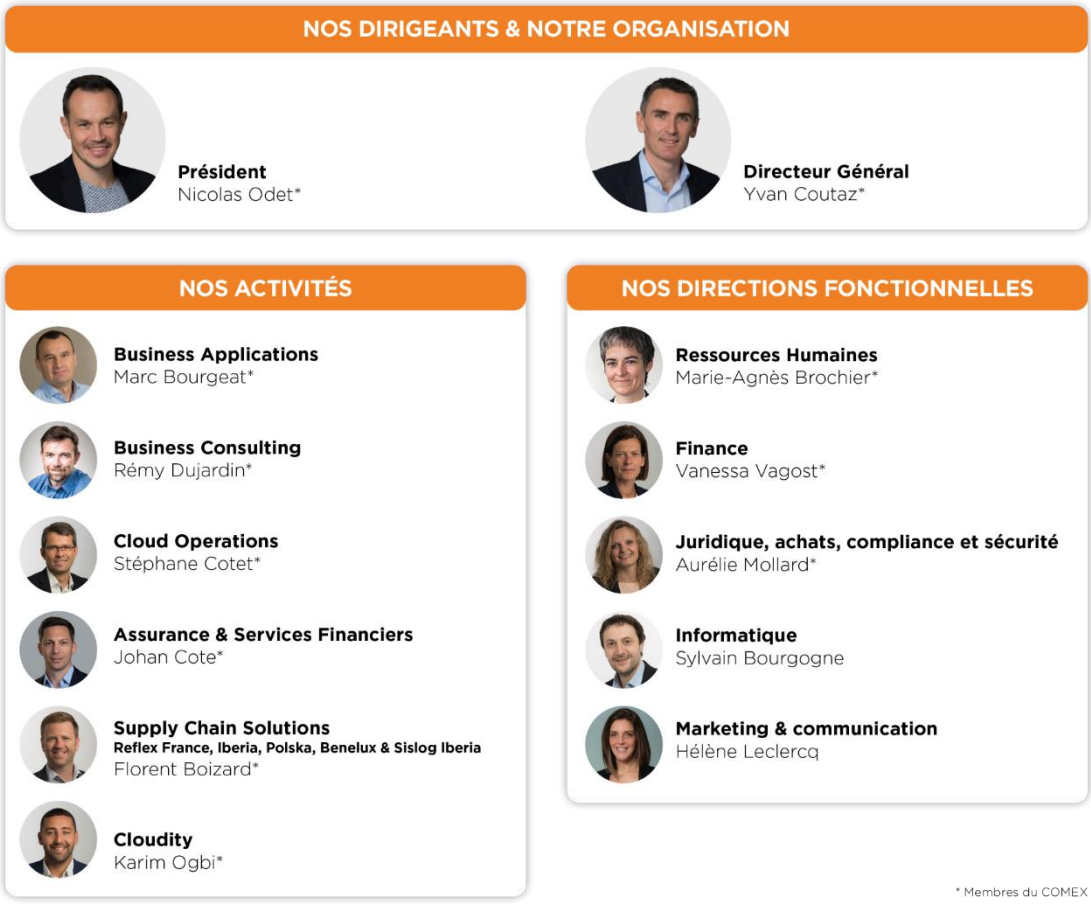


Figure 4 : Bilan de l'entreprise Hardis Group pour l'année 2023

3. *Annexe C : Bilan du progiciel Reflex WMS pour l'année 2023*

Figure 5 : Bilan du progiciel Reflex WMS pour l'année 2023

4. *Annexe D : Organigramme de l'entreprise Hardis Group*



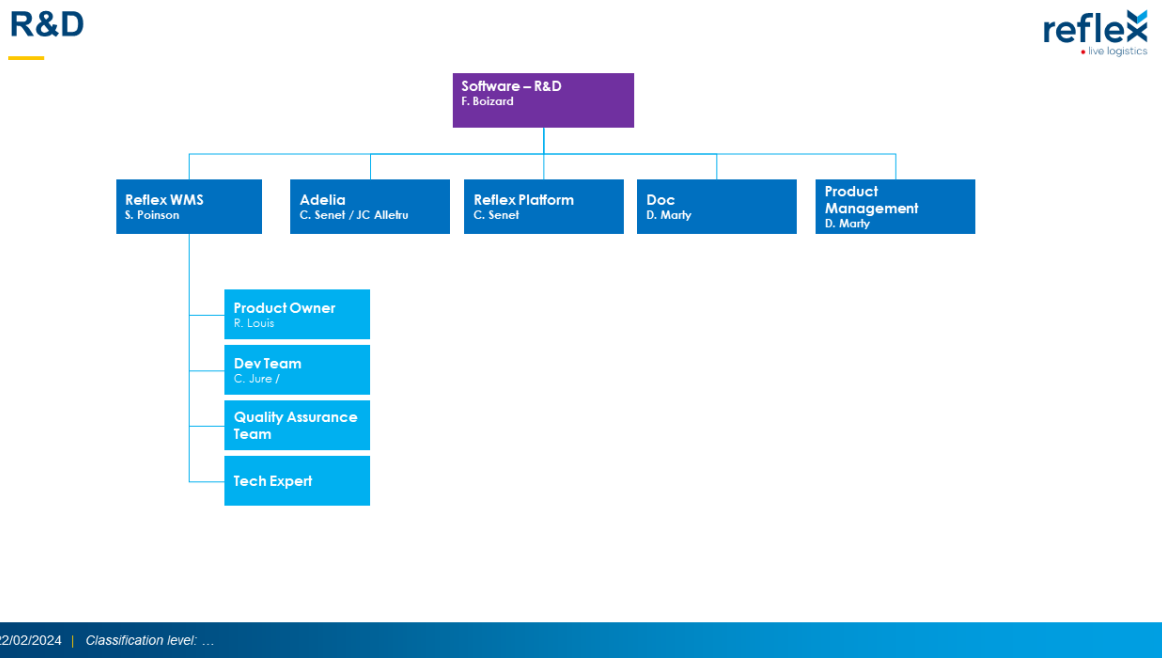


Figure 6 : Organigramme de l'entreprise Hardis Group

5. *Annexe E : Interface Web de Reflex*

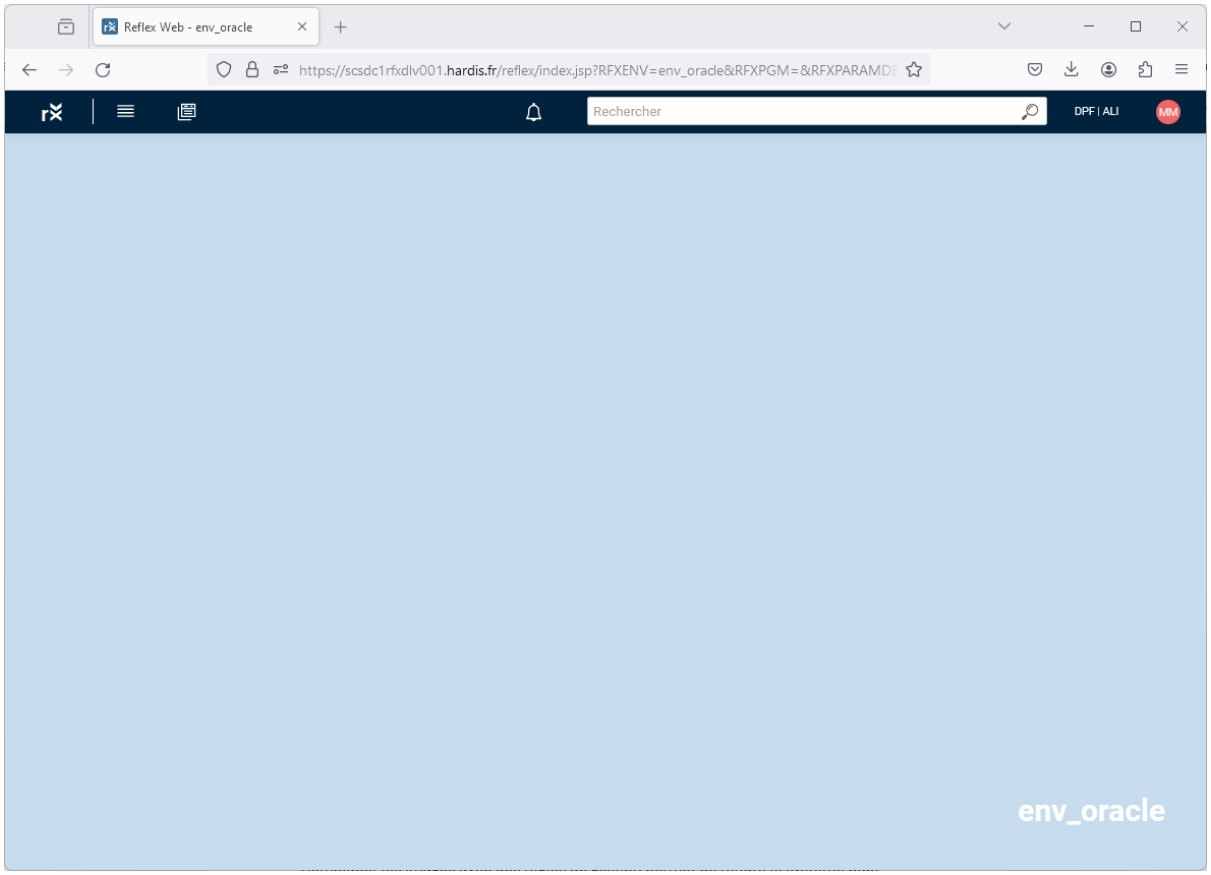


Figure 7 : Interface Web de Reflex

6. *Annexe F : Arborescence remontante du logiciel KListe*

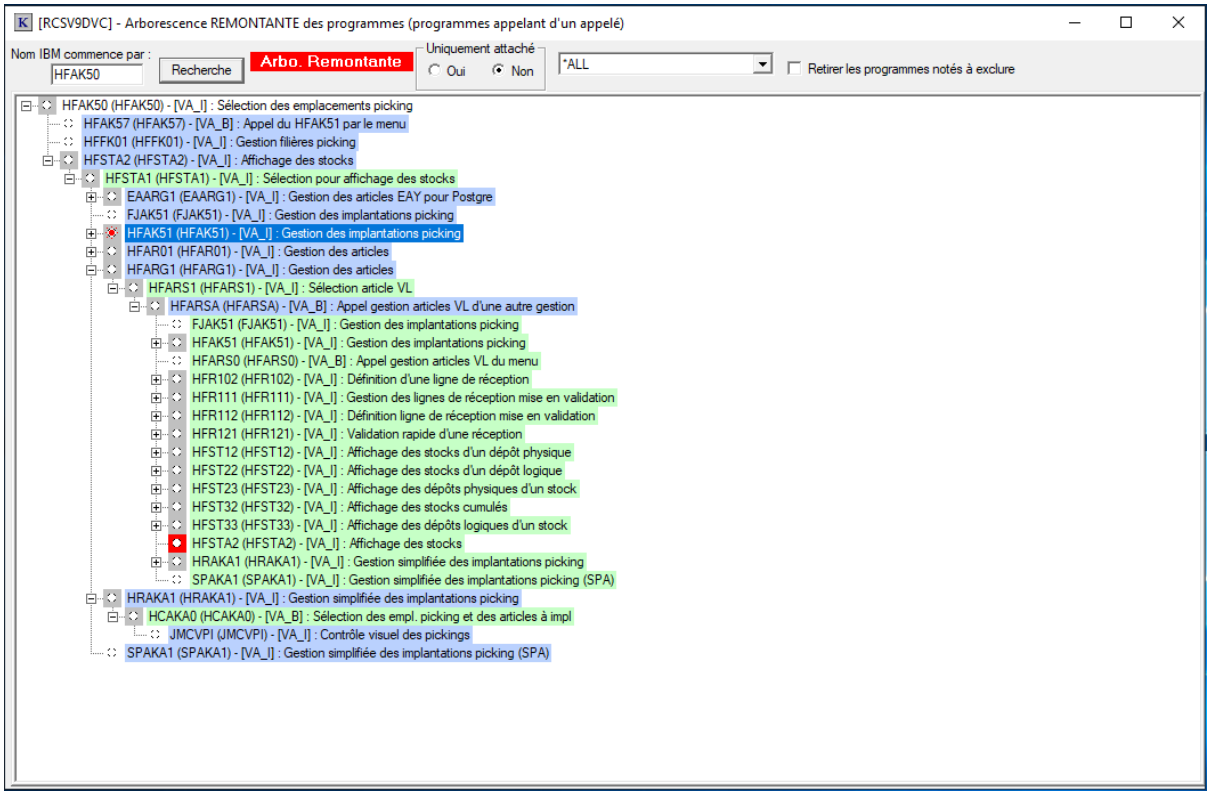


Figure 8 : Arborescence remontante du logiciel KListe

7. *Annexe G : Outil pour simplifier l'utilisation de Git*

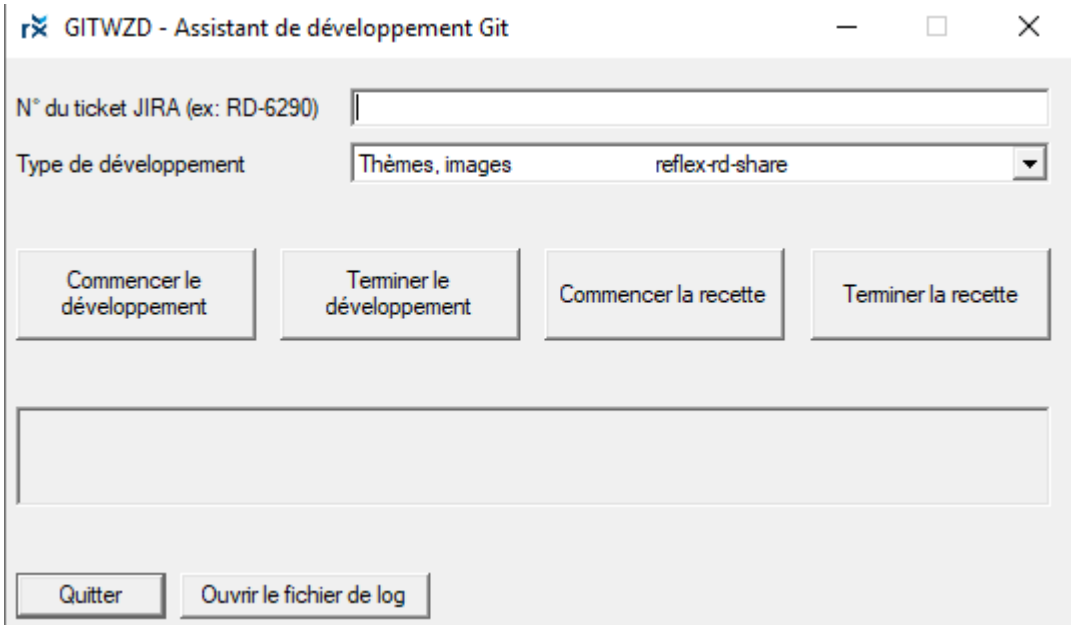


Figure 9 : Outil pour simplifier l'utilisation de Git

8. *Annexe H : Interface Web de Reflex en version 9.20 (il y a environ 2 ans)*

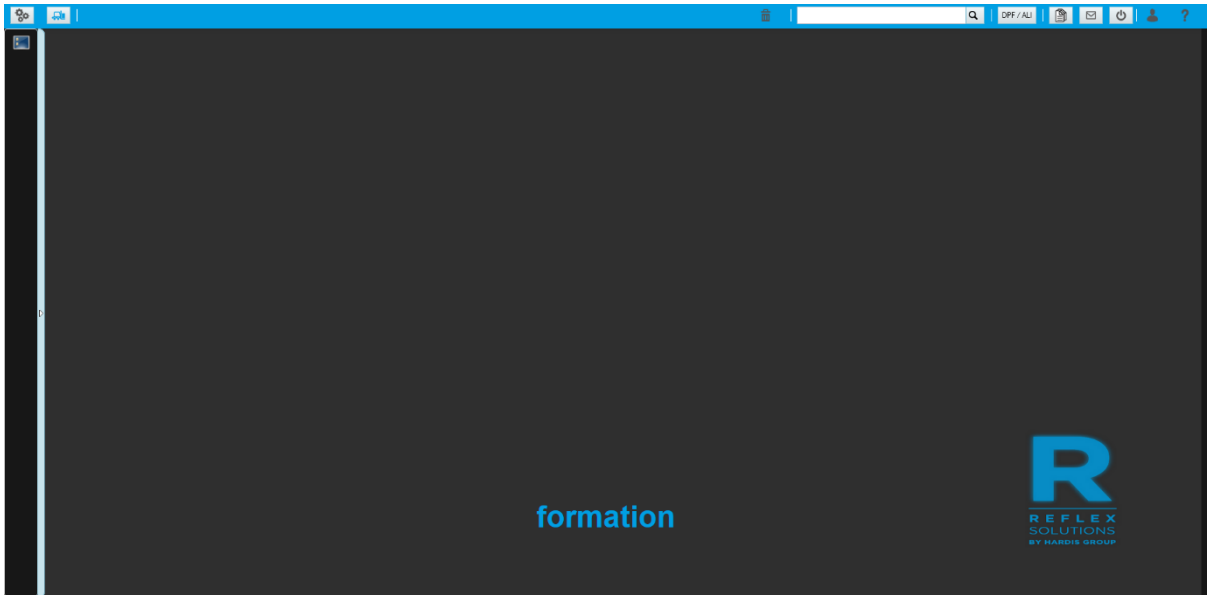


Figure 10 : Interface Web de Reflex en version 9.20 (il y a environ 2 ans)

9. *Annexe I : Backlog Jira de Corinne Juré*

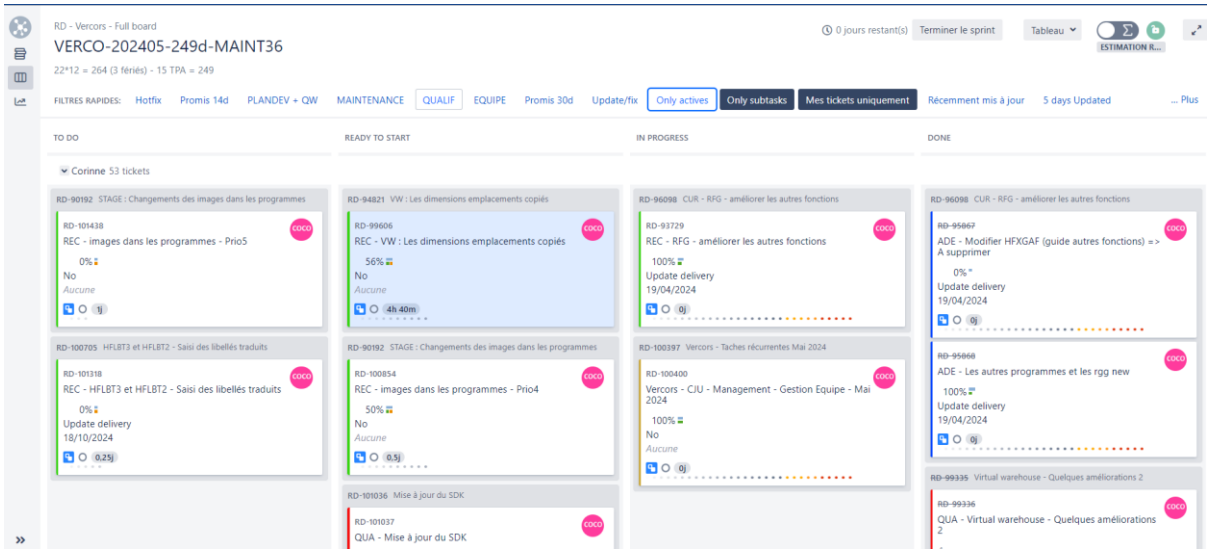


Figure 11 : Backlog Jira de Corinne Juré

10. Annexe J : Logigramme du processus de correction

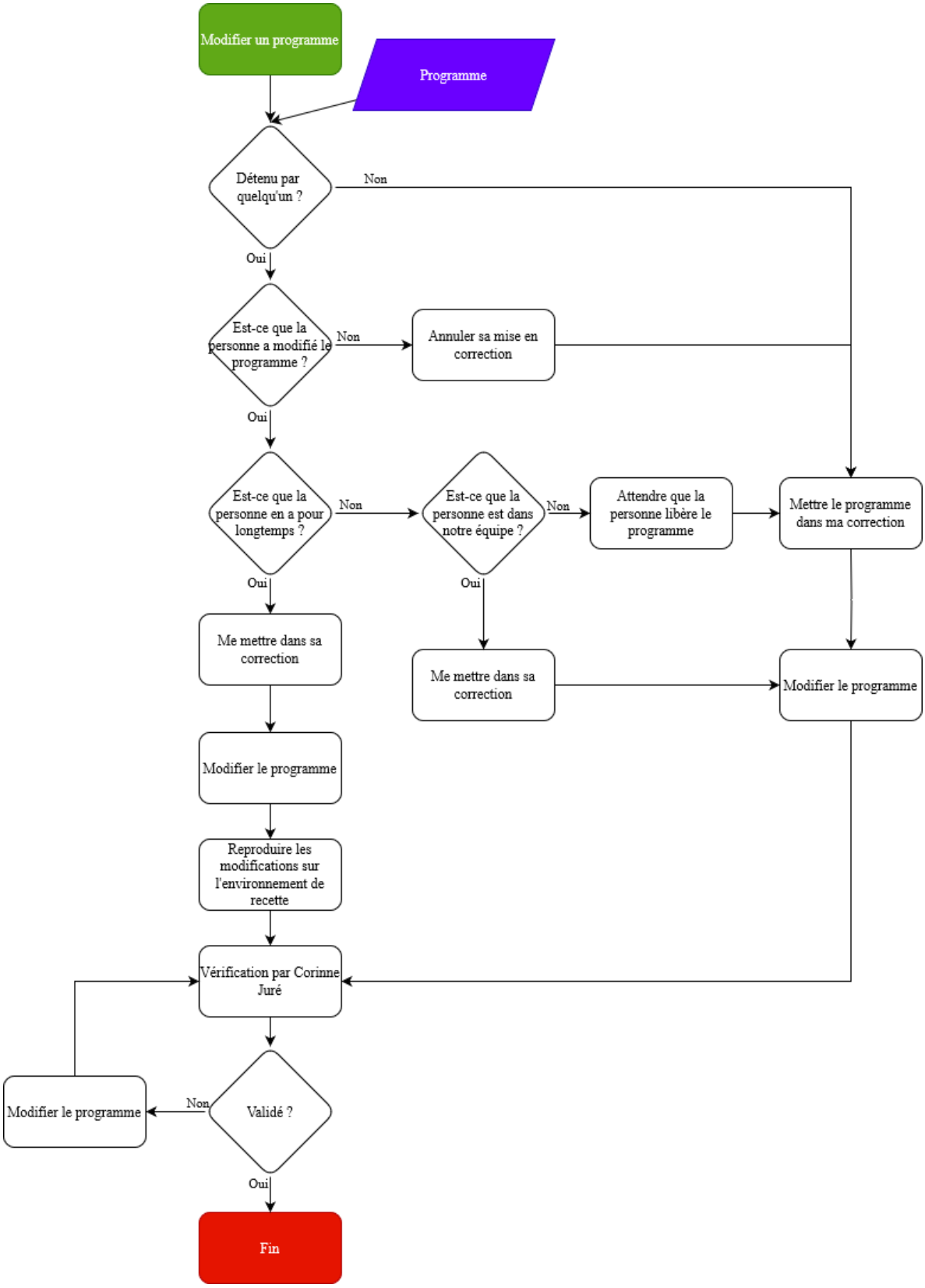


Figure 12 : Logigramme du processus de correction