

DOSSIER DE VALIDATION

Concepteur Développeur d'Applications Numériques

Nom Prénom	Hilaire Sirika
Nom(s) Prénom(s) du ou des tuteurs	Perez José, Pascal Ollier
Acronyme de la certification IPI visée	CDAN
Niveau visé	Niveau II
Date de la soutenance	Mercredi 27 Novembre 2019 à 11h
Lieu de la soutenance	EPSI/WIS BORDEAUX – 114 rue Lucien Faure – 33000 BORDEAUX





Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon alternance et qui m'ont accompagnée et aidée lors de cette formation.

Tout d'abord, je tiens à remercier l'entreprise CGI dans laquelle j'ai eu l'opportunité d'effectuer mon alternance. C'est une entreprise accueillante, dans laquelle il fait bon vivre et ce fut très agréable d'y travailler.

Ensuite, je souhaiterai remercier l'organisme de formation EPSI qui possède une équipe pédagogique à l'écoute ainsi que des intervenants très qualifiés pour dispenser les cours.

Je remercie également l'équipe de la TMA grâce à laquelle j'ai pris beaucoup de plaisir à venir travailler.

Plus particulièrement, j'aimerais remercier les membres de mon équipe : Pascal OLLIER, José PEREZ, Samuel DULHOSTE, Jennifer MASUREUR, Ceddyk ALICHE, Benjamin BARBE, Pierre LORSON et Romain MATHIEU car ils ont toujours fait preuve de bienveillance à mon égard et m'ont toujours transmis leurs compétences de façon pédagogique et agréable.

Enfin, je remercie également tous mes collègues de la promotion U'DEV2 avec qui j'ai passé d'agréable moments et partagé cette belle aventure.



Table des matières

1. Présentation de mon entreprise	10
1.1. Histoire.....	10
1.2. CGI – International	11
1.2.1. Les membres de la Haute Direction de CGI	12
1.2.2. Les valeurs de CGI	13
1.3. CGI – France	14
1.3.1. L'agence de Bordeaux	15
2. Présentation du client.....	16
2.1. La TEAM AM	17
2.1.1. L'équipe	17
2.1.2. Nos missions.....	18
2.1.3. Organisation interne.....	19
2.1.4. Notre environnement technique.....	20
3. Mes missions	21
3.1. La résolution d'incidents	21
3.2. Rédiger de la documentation.....	22
3.3. Effectuer les contrôles quotidiens.....	22
3.4. Produire du code.....	24
4. Projet réalisé en entreprise	29
4.1. Contexte	29
4.2. Explications de l'évolution, définition du besoin.....	32
4.3. Spécifications fonctionnelles (SFD)	33
4.4. Spécifications techniques (STD)	37
4.5. Pré-rédaction des tests ALM	39
4.5.1. Requirements	39
4.5.2. Test plan	39
4.5.2.1. Cas de rejets.....	40
4.5.2.2. E-mails	41
4.5.2.3. Traitement de la répartition	41
4.5.2.4. Historisation	45

4.5.2.5. Mon test plan	46
4.5.2.6. Mon test lab	49
4.6. Script SQL.....	50
4.7. Code JAVA	51
4.8. Tests sur ALM.....	51
4.8.1. Tests unitaires	51
4.8.2. Tests globaux.....	52
5. Complément de compétences.....	53
6. Conclusions	56
6.1. Bilan du projet	56
6.2. Bilan personnel et avenir professionnel	56
6.3. Compréhension du métier visé	57
7. Glossaire	59
8. Annexes.....	62
8.1. Annexe 1 - Présentation des tickets Jira	62
8.2. Annexe 2 - Rédaction de la procédure pour effectuer un split de serial manuellement	64
8.3. Annexe 3 - Rédaction de la procédure pour comprendre les flags APP_DOC	65
8.4. Annexe 4 - Requêtes SQL.....	65
8.4.1. Récupérer le pourcentage de distribution.....	65
8.4.2. Calculer la somme des pourcentages de distribution	66
8.4.3. Récupérer le multiple de conditionnement.....	66
8.4.4. Récupérer le type de distribution.....	67
8.4.5. Compter le nombre de réceptions partielles	68
8.4.6. Compter le nombre de répartition automatique	68
8.4.7. Récupérer les lignes T_QUANTITIES	69
8.4.8. Requête d'insertion pour historisation.....	70
8.4.9. Récupérer tous les sites d'un contrat	71
8.5. Annexe 5 - Code Java FEB 2628	72
8.6. Annexe 6 – Gabarit de compétences CDAN IPI.....	79
8.7. Annexe 7 – Fiche d'évaluation en entreprise	84
8.8. Annexe 8 – Rédaction de règles de gestion	85
8.9. Annexe 9 – Développement d'une application N-Tiers	88
8.10. Annexe 10 – Réalisation de maquettes	89

8.11. Annexe 11 – Réalisation d'une IHM	90
8.12. Annexe 12 – Réalisation de MCD	93
8.13. Annexe 13 – Exemple de Javadoc	95
8.14. Annexe 14 – Exemple procédure utilisateur	96
8.15. Annexe 15 – Exemple correspondance utilisateur anglophone.....	96
8.16. Annexe 16 – Machines virtuelles	97
8.17. Annexe 17 – CV à jour	98



Table des illustrations

Figure 1- Serge Godin	10
Figure 2 - Le rêve de CGI.....	10
Figure 3 - Logo CGI	11
Figure 4- Organigramme de l'équipe de Direction et ses services corporatifs de CGI	12
Figure 5- Organigramme des opérations mondiales de CGI	12
Figure 6- Implantation de CGI en France.....	14
Figure 7 - CGI Bordeaux, Le Haillan.....	15
Figure 8- Organigramme Sub-BU FGDC Bordeaux.....	15
Figure 9- Organigramme TEAM AM	17
Figure 10- Cycle en V	19
Figure 11 - Environnement technique du projet	20
Figure 12- Schéma échanges Baan – APP_Suivi- APP_Reception.....	21
Figure 13- Schéma explicatif Flux POS	23
Figure 14- Impression écran d'un e-mail suite au développement effectué (1)	24
Figure 15- Impression écran d'un e-mail suite au développement effectué (2)	24
Figure 16- Schéma explicatif Flux Despatch Advices	25
Figure 17- Bon expédition 800 pièces - partie 2	26
Figure 18 - Bon expédition 800 pièces - partie 1	26
Figure 19 - Bon expédition 800 pièces - partie 3	26
Figure 20- Résultat analyse SonarQube suite au commit de mon code	27
Figure 21- Détail analyse technique pour chiffrage FEB 3119	28
Figure 22- Structure fichier ESNPOS	30
Figure 23- Règles de gestion des multi-sites	33
Figure 24- Processus à suivre pour la répartition automatique	33
Figure 25- Processus à suivre pour la répartition manuelle ou non initialisée	34
Figure 26- Règles de gestion partie1	34
Figure 27- Règles de gestion partie 2	35
Figure 28- Modifiactions à faire en base de données - partie 1	35
Figure 29 - Modifiactions à faire en base de données - partie 2	36
Figure 30- STD partie 1	37
Figure 31- STD partie 2	38
Figure 32- STD partie 3	38
Figure 33 - Cas de tests FEB 2628.....	46
Figure 34 - Design steps d'un cas de test FEB 2628	47
Figure 35 - Design steps d'un cas de test FEB 2628 - 2	48
Figure 36 - Test lab FEB 2628.....	49
Figure 37- Tests unitaires d'une fonction.....	51
Figure 38- Exemple résultat tests unitaires.....	51
Figure 39- Jeux de données tests globaux	52
Figure 40- Requirements couverts et validés.....	52

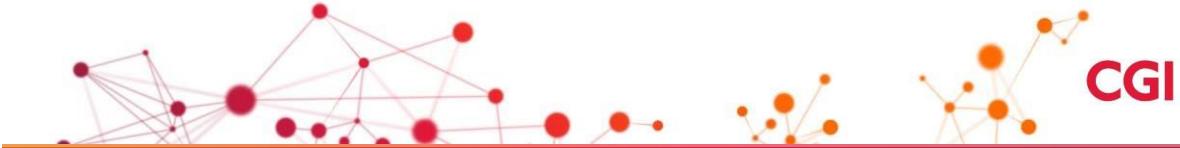


Figure 41- IHM FEB 2628 - Ecran de répartition des sites	53
Figure 42- Architecture n-tiers.....	53
Figure 43 - Exemple d'import de controller dans une jsp (couche Vue)	54
Figure 44 - Exemple d'un controller et de l'appel à la couche Modèle	55
Figure 45 - Exemple couche Modèle.....	55
Figure 46- Satisfaction client team AM septembre 2019	56



Abstract

In order to validate my professional level II certificate « Concepteur Développeur d'Applications Numériques », I followed an internship from December 2018 to December 2019 at CGI, which is a Canadian global information technology consulting, systems integration, outsourcing, and solutions company headquartered in Montreal, Quebec.

I have been placed into a Third Party Application Management team for an international group which designs, makes and maintains engines for commercial and military aircraft.

This year, I worked on four applications. My missions consisted of resolving incidents and problems and developing evolutions. I had to analyze and design technical solutions, constantly evaluate my remaining to do, correct anomalies, prepare and carry out test cases.

In this document, you will see how I evolved throughout this period, all the skills that I have learned and mastered, and how they made me a better developer.

Afin de valider mon titre professionnel de niveau II « Concepteur Développeur d'Applications Numériques », j'ai effectué une alternance de décembre 2018 à décembre 2019 à CGI, une entreprise canadienne de services-conseils en technologie de l'information, d'intégration de systèmes, d'impartition et de solutions, dont le siège social est établi à Montréal, Québec.

J'ai été placée dans une équipe de Tiers Maintenance Applicative pour un groupe international qui conçoit, fabrique et entretient des moteurs pour les avions commerciaux et militaires.

Cette année, j'ai travaillé sur quatre applications. Mes missions consistaient à résoudre des incidents et des problèmes et participer au développement des évolutions. Je devais analyser et concevoir des solutions techniques, constamment réévaluer mon reste à faire, corriger des anomalies, préparer et effectuer des cas de test.

Dans ce document, vous verrez comment j'ai évolué durant cette période, toutes les compétences que j'ai apprises et maîtrisées et comment elles ont fait de moi une meilleure développeuse.



Référentiel

Dans ce dossier de validation du titre de niveau II "Concepteur Développeur d'Applications Numériques", je vais vous présenter les missions que j'ai menées en entreprise et qui m'ont permis de mettre en pratique les compétences à valider dans le référentiel de certification.

Le référentiel est constitué de 5 grands blocs de compétences :

**Qualité et sécurisation
du code réalisé**

**Audit, conception,
méthode de projet**

**Réalisation
d'applications
logicielles**

**Communiquer avec
les acteurs du projet**

**Adapter
l'environnement
d'exécution, échanger
des données entre
logiciels**

Le portefeuille de preuves précisant les circonstances d'acquisition de ces compétences est représenté par le gabarit des compétences CDAN complété en annexe n°6. Vous pourrez y trouver les annexes et numéro de pages de ce dossier qui sont en lien avec les compétences à valider.

Ce gabarit des compétences sera suivi de la fiche d'évaluation complétée par mon tuteur, en annexe n°7.

1. Présentation de mon entreprise

1.1. Histoire



Figure 1- Serge Godin
CGI (Conseillers en Gestion et Informatique) a été fondé par Serge Godin en 1976 au Québec et il a été rejoint quelques mois plus tard par André Imbeau. Ils avaient tous les deux comme rêve de créer une entreprise où tout le monde prendrait plaisir à travailler et pourrait en devenir actionnaire-propriétaire.

Dès les premières années d'existence de CGI, les premières valeurs s'instaurent : aider les clients à atteindre le succès et offrir des opportunités de carrière à ses membres. Tout est mis en œuvre pour que les occasions d'affaires soient saisies et le développement du secteur des services en TI, notamment l'impartition*, a permis à CGI de connaître une forte croissance.

A partir de 1986, CGI commence à fusionner avec des sociétés d'impartition pour pouvoir répondre à la demande croissante de ses clients mais également pour s'étendre aux autres pays.

Parmi les plus importantes fusions nous pouvons notamment retenir :

- En 1998, la fusion de CGI et de Bell Sygma
- En 2001, CGI fusionne avec IMRGlobal pour se doter de bureaux en Inde.
- En 2004, CGI fusionne avec American Management System (AMS)
- En 2010, CGI fusionne avec Stanley Inc., et ses filiales Oberon et Techrizon.
- En 2012, CGI a réalisé sa plus grande acquisition à ce jour en fusionnant avec Logica, faisant passer sa taille de 31 000 à 68 000 membres.
- En 2016, plusieurs fusions stratégiques : JSL, Alcyane et Collaborative Consulting.
- En 2017, plusieurs fusions axées sur les marchés métropolitains, dont quatre firmes de services-conseils stratégiques établies aux États-Unis : CTS (Birmingham, AL) et ECS Team (Denver, CO), Summa Technologies (Pittsburgh, PA) et Paragon Consulting (Philadelphia, PA/New Jersey/New York). En Finlande, CGI a procédé à une fusion avec Affecto Plc.
- En 2018, fusion avec ckc AG.
- En 2019, CGI annonce la fusion avec Acando AB.

Toutes ces fusions permettent à CGI d'agrandir l'éventail de ses compétences afin de répondre au mieux aux divers besoins de ses clients. CGI continue à s'adapter pour mieux tenir compte des changements du marché des TI et satisfaire les attentes de ses clients, ses membres et de ses actionnaires.

Notre rêve

“ Créer un environnement où nous avons du plaisir à travailler ensemble et où, en tant que propriétaires, nous participons au développement d'une entreprise dont nous sommes fiers. ”

Figure 2 - Le rêve de CGI



1.2. CGI – International

Le siège social de CGI est établi à Montréal au Québec, Canada. A l'heure actuelle, la société est composée de près de 77.000 employés et réalise un chiffre d'affaire de 11,5 milliards de dollars canadien.



Figure 3 - Logo CGI

CGI est établi sur 4 continents différents : Europe, Amérique, Afrique, Asie.

CGI propose différents services :

- Services-conseils en management
Conseils pour favoriser la transformation numérique de l'entreprise.
- Intégration de systèmes
Mettre en place la synergie entre les systèmes et les technologies en garantissant transparence et sécurité.
- Services d'impartition IT
Outsourcing IT.
- Services applicatifs
Optimiser et développer les applications d'entreprise.
- Services d'infrastructure
Aider les clients à devenir des organisations numériques.
- Gestion des processus d'affaires.
Aider les clients à devenir des organisations numériques axées sur le client tout en améliorant leurs fonctions d'affaires.

On peut trouver plusieurs secteurs d'activités différents au sein de CGI : Assurance, commerce de détail et services aux consommateurs, communications, gouvernements, pétrole et gaz, santé, sciences de la vie, secteur manufacturier, services publics, services bancaires et marchés des capitaux, transport et logistique...

1.2.1. Les membres de la Haute Direction de CGI

La structure organisationnelle de CGI s'articule autour de trois pôles : l'équipe de direction, les services corporatifs qui gèrent l'organisation et le fonctionnement du groupe à l'échelle mondiale et les opérations mondiales qui représentent les différents centres d'opérations à travers le monde.

L'équipe de direction est composée de Serge Godin et George Schindler. Les services corporatifs se composent de 9 membres de la Haute Direction.

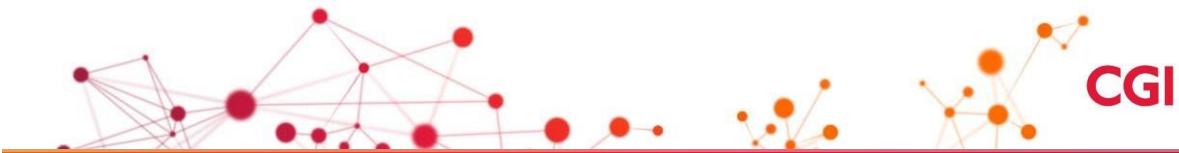


Figure 4- Organigramme de l'équipe de Direction et ses services corporatifs de CGI

Les opérations mondiales se constituent de 9 membres de la Haute Direction.



Figure 5- Organigramme des opérations mondiales de CGI



1.2.2. Les valeurs de CGI

Elles permettent à tous les employés de vivre selon les mêmes normes éthiques.

- Partenariat et qualité

Comprendre les activités des clients, développer de meilleures pratiques de gestion et être à l'écoute afin d'établir des partenariats solides et durables.

- Objectivité et intégrité

Recommander des choix, services et solutions en toute indépendance aux clients.

- Intrapreneurship et partage

Travail d'équipe et partage des connaissances entre les employés. De plus, tous les employés ont accès à l'actionnariat pour participer aux profits de l'entreprise.

- Respect

En toute situation, il y a le respect des membres, clients, partenaires et concurrents.

- Solidité financière

Tout le monde travaille à conserver une croissance à long terme pour soutenir les bénéfices des membres et actionnaires.

- Responsabilité sociale

Le modèle d'affaire a été conçu pour développer des liens étroits avec les clients et les communautés.

1.3. CGI – France

CGI compte 18 agences en France, rassemblant 12000 collaborateurs.

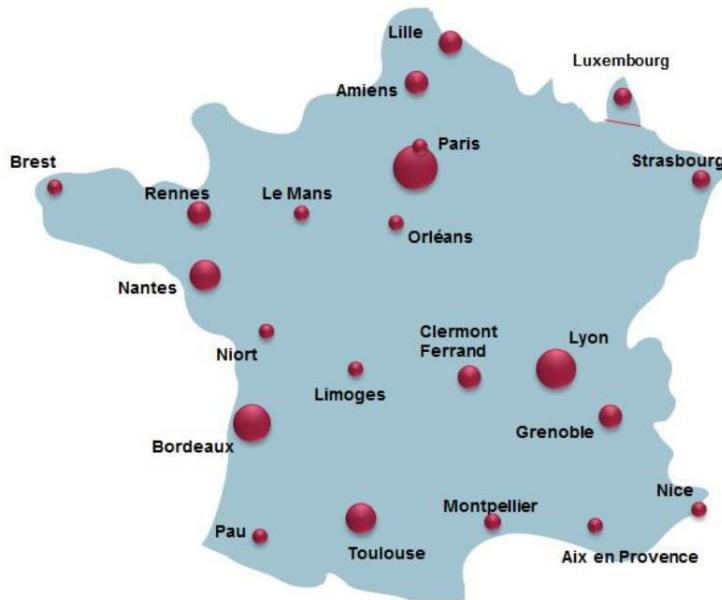


Figure 6- Implantation de CGI en France

Ces collaborateurs sont répartis dans 11 Business Unit* (BU) :

- Business Consulting
- Financial Services
- Energy Utilities Télécom Média
- CGP Retail et manufacturing
- TPSHR (Transport, Public Services, Human Ressources)
- Nord
- Grand Est
- Grand Ouest
- Grand Sud
- France GDC (Global Delivery Center)
- I2CE (Innovation & Industrialisation Center of Excellence)

1.3.1. L'agence de Bordeaux

L'agence CGI à Le Haillan (Bordeaux) compte plus de 900 employés et est divisée en trois BU : FGDC, Grand-Ouest et TPS/HR.



Figure 7 - CGI Bordeaux, Le Haillan

Au sein de l'entreprise, je fais partie de la BU (Unité d'affaire) FGDC (France Global Delivery Center), et plus particulièrement de la Sub-BU FGDC Bordeaux.

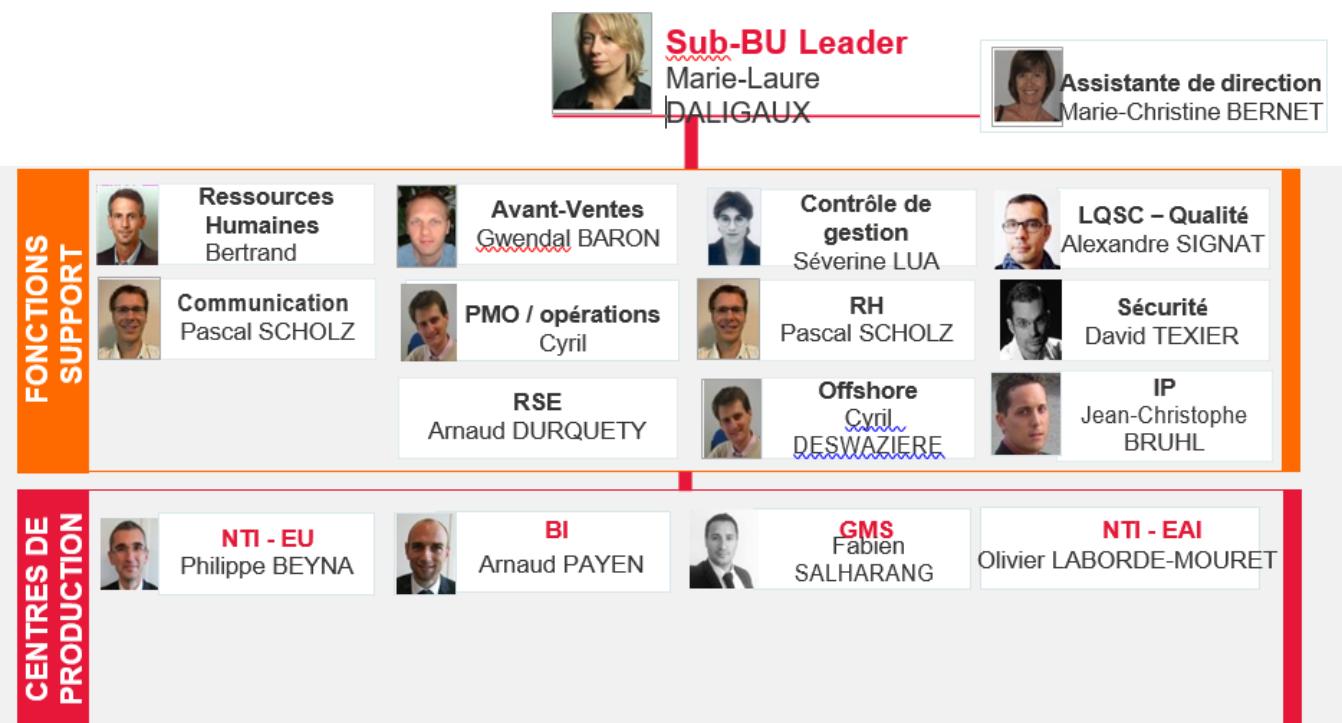


Figure 8- Organigramme Sub-BU FGDC Bordeaux

2. Présentation du client

Le client pour lequel j'ai travaillé dans le cadre de mon alternance est un grand groupe industriel et technologique français présent au niveau international dans les domaines de l'aéronautique, de l'espace et de la défense. Il a été créé en 2005. Depuis septembre 2011, il est coté au CAC40*. Ses métiers sont la conception et la production de moteurs d'avions, d'hélicoptères et de fusées, d'équipements aéronautiques, et de défense. Il occupe des positions de leader mondial sur ces marchés et emploie plus de 58.000 personnes dans près de trente pays.

Au sein de CGI, il est rattaché à l'entité FGDC. Un plateau sécurisé lui est entièrement dédié pour protéger ses données confidentielles.

L'équipe en charge de sa TMAM (Tierce Maintenance Applicative Massive) s'occupe du maintien en condition opérationnelle (MCO) et des évolutions pour une centaine d'applications depuis 2016. Elle est composée d'une trentaine de personnes réparties sur cinq pôles :

- Le pôle Applications Métier.
- Le pôle BI (Business Intelligence).
- Le pôle PIM.
- Le pôle Portail Web.
- Le pôle Support DSI.

2.1. La TEAM AM

Applications Métier (AM) est le pôle dans lequel j'ai effectué mon année d'alternance et qui s'occupe des applications métiers de notre client.

Pour des raisons de confidentialité, j'ai nommé autrement les applications. Voici les principales :

- **APP_Suivi:** application par laquelle les fournisseurs gèrent l'envoi de leurs pièces, consultent et négocient leurs programmes...
- **APP_Doc:** application par laquelle les fournisseurs peuvent accéder à toute la documentation relative au client, aux procédures, au montage des pièces...
- **APP_Reception:** application qui permet d'enregistrer les pré-réceptions des pièces qui ont été expédiées par les fournisseurs ainsi que l'enregistrement des réceptions de celles qui ont été réceptionnées.
- **FLUX :** il s'agit de tous les flux qui font le lien entre les applications, notamment pour gérer l'accès aux bases de données, la partie métier, le transfert des fichiers, l'envoi des e-mails, les vérifications de données...

2.1.1. L'équipe

Nous sommes 7 à travailler dans la TEAM AM. Cependant, des collaborateurs venant des autres pôles peuvent être amenés à intervenir sur notre projet et inversement.

Voici l'organigramme de notre équipe :

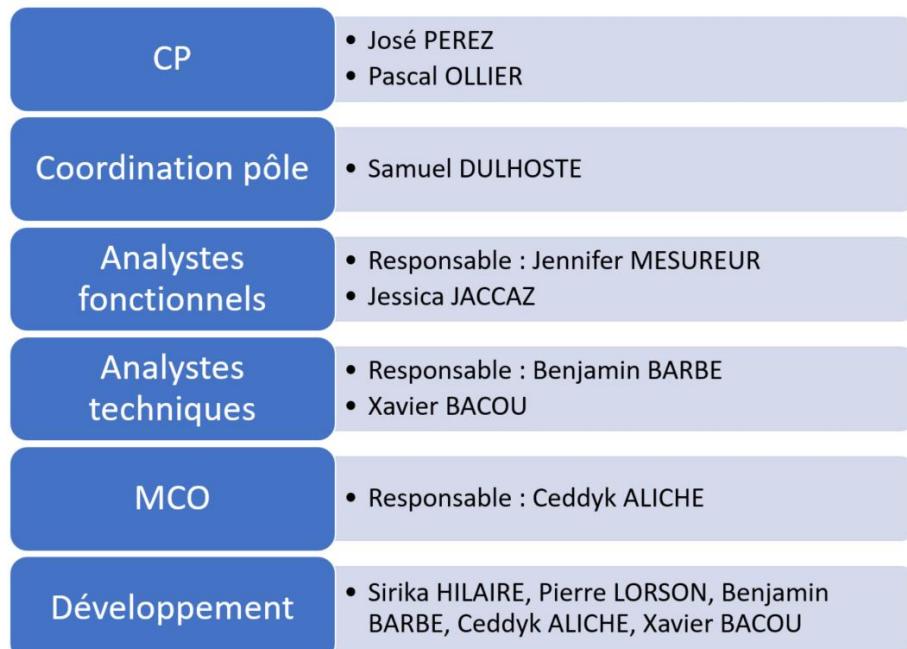


Figure 9- Organigramme TEAM AM

Les membres que je côtoie quotidiennement sont Pascal Ollier, Samuel Dulhoste, Jennifer Mesureur, Benjamin Barbe, Ceddyk Aliche et Pierre Lorson.

Xavier Bacou, expert technique, fait partie de la TEAM PIM mais intervient en cas de besoin sur des spécifications techniques ainsi que des développements sur l'application APP_Reception.

Nous avons également Matthieu Mingo qui est architecte et expert technique sur notre plateau et qui intervient sur nos projets.

Enfin, depuis quelques mois maintenant, la TMA a recruté un prestataire externe : Romain Mathieu, qui travaille principalement au sein de notre équipe mais qui intervient à l'occasion sur les autres pôles.

Notre équipe possède donc deux chefs de projet, un pilote (Samuel DULHOSTE), deux analystes fonctionnelles, deux analystes techniques, un responsable MCO et cinq développeurs.

2.1.2. Nos missions

Dans le cadre de nos missions nous sommes régulièrement en relation avec les différents métiers* des applications ainsi que les employés côté client qui effectuent et vérifient diverses opérations côté Baan.

Qu'est-ce Baan ?

Je me permets de parler ici de Baan car j'utiliseraï souvent ce terme dans le document. Baan est l'ERP (Entreprise Resource Planning) côté client. C'est le progiciel principal avec lequel tous les pôles communiquent. Il regroupe toutes les données pour toutes les applications. Il y a donc énormément d'échanges quotidiens qui se produisent entre Baan et toutes les applications que nous avons en charge. Cependant, nous n'avons pas en charge Baan et nous n'y avons pas non plus accès en consultation. C'est pourquoi, lorsqu'il y a des divergences entre les données, nous devons entrer en relation avec les employés côté client qui s'occupent de vérifier les données dans Baan afin que nous puissions apporter les corrections nécessaires.

Nous devons quotidiennement traiter les incidents qui peuvent survenir et qui viennent de toutes les applications. Ces incidents sont créés soit par les utilisateurs soit par les métiers ou les employés côté client par l'intermédiaire de la hotline informatique. Il s'agit le plus régulièrement de rattrapage à faire en base de données ou de logs* à analyser pour trouver la cause d'un problème.

Il y a toujours des améliorations ou des évolutions à apporter aux applications. Ces demandes peuvent venir de nos garants métiers ou des utilisateurs ou des propositions de notre part. De ce fait, nous avons constamment des choses à modifier, ajouter dans le code. On parle alors de

release*. Nous mettons en production tous les 2-3 mois (selon la quantité de travail) des releases qui englobent des évolutions et des correctifs.

Les métiers peuvent parfois détecter des anomalies et nous demander la création d'un problème. Les problèmes sont des fonctionnalités qui n'existent pas ou ne fonctionnent pas correctement selon les spécifications fonctionnelles en place. Nous devons alors analyser la cause du problème, rédiger des tests et corriger le code afin de pouvoir proposer une correction lors de la prochaine release.

La production de code a donc plusieurs origines :

- Ajouter de nouvelles fonctionnalités, on appelle ça une évolution (FEB : Fiche d'évolution du besoin)
- Corriger des problèmes ou ajouter des choses manquantes par rapport aux spécifications fonctionnelles, on parle alors de problème.
- Fiabiliser le code en ajoutant des logs, en améliorant les messages d'erreurs, d'exception*, en améliorant la structure du code actuel. On parle de fiabilisation (FIAB).

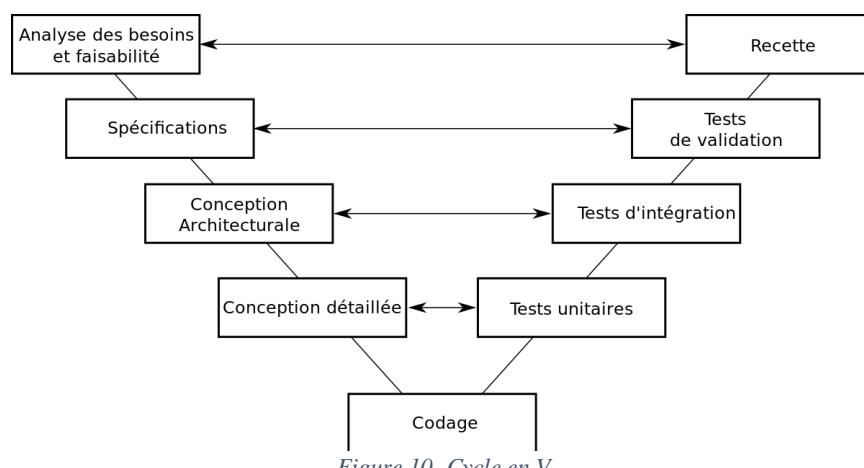
Outre les incidents et les développements, nous veillons à améliorer les procédures en place afin de gagner du temps, du confort et de la fiabilité aux yeux du client.

2.1.3. Organisation interne

Les temps de développement sont initialement estimés par la personne qui effectue le chiffrage*. Il s'agit le plus souvent du référent technique. Cependant, il m'est déjà arrivé d'estimer des temps de développement dans le cadre d'un chiffrage (voir partie 3.4).

Tous les matins, nous effectuons des mornings* afin de partager à l'équipe sur quoi nous travaillons. Tous les problèmes et retards y sont rapidement remontés et discutés afin de ne pas occasionner de retard à la livraison.

Nous suivons le cycle en V (voir figure 10), et j'ai, au cours de cette année, pris part aux phases Conception Architecturale, Conception détaillée, Codage et Tests unitaires. Il m'est également arrivé d'anticiper certains problèmes non spécifiés dans les spécifications fonctionnelles.



Nous devons également, chaque jour, renseigner ce que nous avons fait sur l'outil de travail collaboratif Jira. Jira est un système de suivi de bugs*, de gestion d'incidents et de gestion de projet. Chaque incident que nous devons résoudre et chaque développement que nous devons faire est créé sous forme d'un ticket. A chaque étape de la résolution de l'incident ou de l'avancée du développement, nous devons écrire ce qui a été fait et indiquer le temps travaillé ainsi qu'estimer le temps restant. Vous pourrez trouver en annexe n° 1 des exemples de ticket Jira ainsi que la présentation du reste à faire.

2.1.4. Notre environnement technique

Afin d'assurer la maintenance applicative, nous disposons de plusieurs outils :

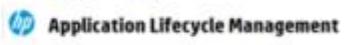
<i>Domaine / Fonctions</i>	<i>Outils</i>	<i>Logo</i>
<i>Gestion des tâches</i>	JIRA	
<i>Partage de documents</i>	CONFLUENCE	
<i>Développement IDE</i>	ECLIPSE JUNO ECLIPSE NEON Workshop	
<i>Langages</i>	JAVA >= 1.4 SQL HTML, CSS, JS	
<i>Framework</i>	SPRING HIBERNATE Weblogic	
<i>Serveur d'application</i>	Tomcat	
<i>Testing</i>	HP ALM	
<i>Qualité du code</i>	SONARQUBE	
<i>Gestion de version</i>	SVN	
<i>Intégration Continue</i>	JENKINS	
<i>Base de données</i>	ORACLE	

Figure 11 - Environnement technique du projet

Tous les développeurs disposent d'un environnement de développement avec serveur d'application (weblogic, bea), serveur web (Tomcat), Base de données (Oracle SQLDeveloper) qu'ils doivent installer en suivant les procédures à disposition.

3. Mes missions

Lors de mon année d'alternance, du 3 décembre 2018 à aujourd'hui, on m'a confié de nombreuses missions très variées !

Quand je suis arrivée au sein de l'équipe, j'ai tout d'abord commencé par lire les nombreuses documentations sur l'application APP_Suivi et les flux WLI. Il est à noter que ce sont des applications complexes et qui nécessitent beaucoup de connaissances. C'est pourquoi j'ai d'abord beaucoup travaillé avec notre analyste fonctionnelle Jennifer.

3.1. La résolution d'incidents

Afin de mettre en pratique l'acquisition de ces nouvelles connaissances, on m'a petit à petit initiée à la résolution d'incidents. Comme énoncé plus haut, les incidents tombent quotidiennement et, en ce qui concerne APP_Suivi, consistent le plus souvent à faire du ratrappage de données en base.

L'incident le plus courant était le ratrappage de transit. C'est-à-dire que les approvisionneurs constataient sur le site APP_Suivi que leur reste à livrer, besoin ou nombre de pièces en transit ne coïncidaient pas avec leurs besoins et expéditions réelles. Il fallait donc que je modifie, par des scripts SQL, les données en base pour qu'elles soient cohérentes par rapport à leurs données, à celles de Baan et aux nôtres. Ces incidents m'ont permis de m'habituer aux différentes tables de la base de données.

Un incident qui est courant également est de modifier les serials* attribués aux pièces d'une expédition car l'approvisionneur s'est trompé en les renseignant. Pour cet incident il fallait, après avoir changé les serials, reflaguer* l'expédition pour qu'elle soit renvoyée à APP_Reception.

En effet, APP_Suivi communique avec Baan et APP_Reception :

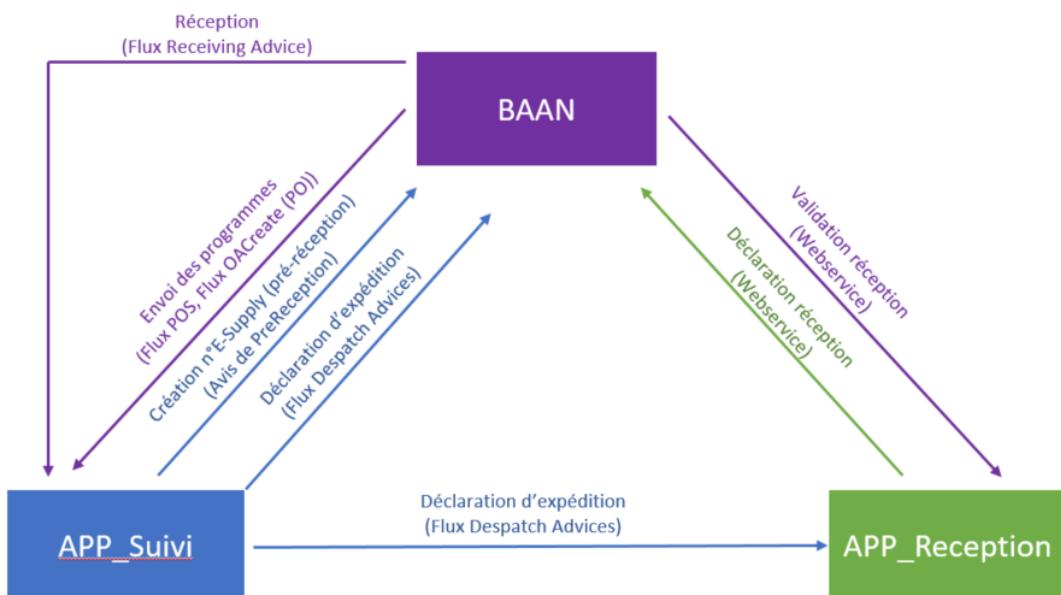


Figure 12- Schéma échanges Baan – APP_Suivi- APP_Reception

Ainsi, si on modifie les numéros de série de pièces qui ont été expédiées, nous modifions l'expédition. Nous devons donc la reflaguer pour qu'elle soit renvoyée à Baan et APP_Reception afin qu'ils aient les données à jour.

Un autre type d'incident que j'ai rencontré plusieurs fois et qui est assez complexe à prendre en main au début est le problème de split* de serials. Lorsqu'un fournisseur soumis aux contrôles qualité (appelés fournisseur V2) expédie des pièces, celles-ci doivent être identifiées par des numéros de série uniques. S'il en expédie plus de 350 en une seule fois, étant donné que les contrôles côté CASQ-IT (application qui effectue les contrôles qualité) ne supportent pas les fichiers de plus de 350 serials, le code prévoit de diviser l'expédition en plusieurs par lot de 350 pièces. A une expédition correspond un numéro de pré-réception. Parfois, pour une raison qui n'a pas encore pu être identifiée car c'est un cas isolé, le split ne s'effectuent pas complètement et des expéditions et serials sont "perdus". Nous devons alors les réintégrer manuellement en demandant au fournisseur ses bons de livraisons (BL) et la liste des serials.

En dehors des incidents APP_Suivi j'ai également résolu des incidents APP_Doc. Ils consistent majoritairement à aider les utilisateurs à trouver leurs documents sur le site APP_Doc ou à reflaguer des documents en base de données pour qu'ils se mettent à jour.

J'ai également participé à la résolution d'incidents APP_Reception. Ils consistent, le plus souvent, à lire des logs pour identifier la raison pour laquelle une pré-réception ne s'y est pas enregistrée.

3.2. Rédiger de la documentation

Quand j'ai commencé à résoudre des incidents, notre analyste fonctionnelle m'avait transmis les compétences pour effectuer les splits de serials manuellement. J'avais donc rédigé une documentation interne à destination de mes collègues afin de leurs transférer la compétence sur notre outil de collaboration : Confluence. Vous pourrez le trouver en annexe n° 2.

J'avais également réalisé une documentation concernant l'application APP_DOC afin d'expliquer les différents types de flags d'une table ainsi que la procédure à appliquer suivant leur état. Vous pourrez la trouver en annexe n° 3.

Enfin, il m'est également arrivé d'écrire des procédures aux utilisateurs pour leurs expliquer comment utiliser une fonctionnalité de l'application APP_DOC. Cela se faisait dans le cadre d'une résolution d'incident. Vous pourrez en trouver un exemple en annexe n°14.

3.3. Effectuer les contrôles quotidiens

Cette tâche est arrivée 2-3 mois après mon arrivée et est devenue une tâche de routine pour tous les membres de l'équipe.

Elle consiste essentiellement à anticiper les écarts de données en base avant que cela n'atteigne les utilisateurs et deviennent bloquant.

Dans l'idée, c'est qu'il existe des dysfonctionnements dans le code, qui ne sont pas encore couverts par une évolution ou qui sont en cours de résolution et qui créent des anomalies en base de données. En attendant de trouver des solutions, nous corrigéons les problèmes pour que ça ait un impact minimum sur les utilisateurs. C'est donc plus correctif que curatif.

Nous avons donc mis en place un batch* qui scanne, à l'aide de requêtes SQL que nous avons créées, les différentes tables de la base de données à la recherche d'anomalies.

Comme exemples d'anomalies nous avons :

- Une expédition a été validée mais ses contrôles CASQ-IT (qualité) sont flagués comme étant KO. Ce qui n'est pas possible car quand une expédition ne valide pas ses contrôles, elle n'est pas validée.
- Des pièces ont été réceptionnées alors qu'elles ne possèdent pas de date d'expédition. Ce n'est pas possible car si des pièces n'ont pas de date d'expédition elles ne sont pas envoyées à APP_Reception.
- Des expéditions ont été réceptionnées sur APP_Reception mais pas dans APP_Suivi. Nous devons alors effectuer la réception dans APP_Suivi en reprenant les informations présentes dans APP_Reception. Il y a dû avoir un problème de communication entre les deux applications.

Nous passons donc un premier scripts SQL avec toutes les requêtes de correction pour les anomalies détectées.

Ensuite, nous devons analyser les e-mails envoyés par le flux WLI POS. C'est le flux qui se situe entre Baan et APP_Suivi :

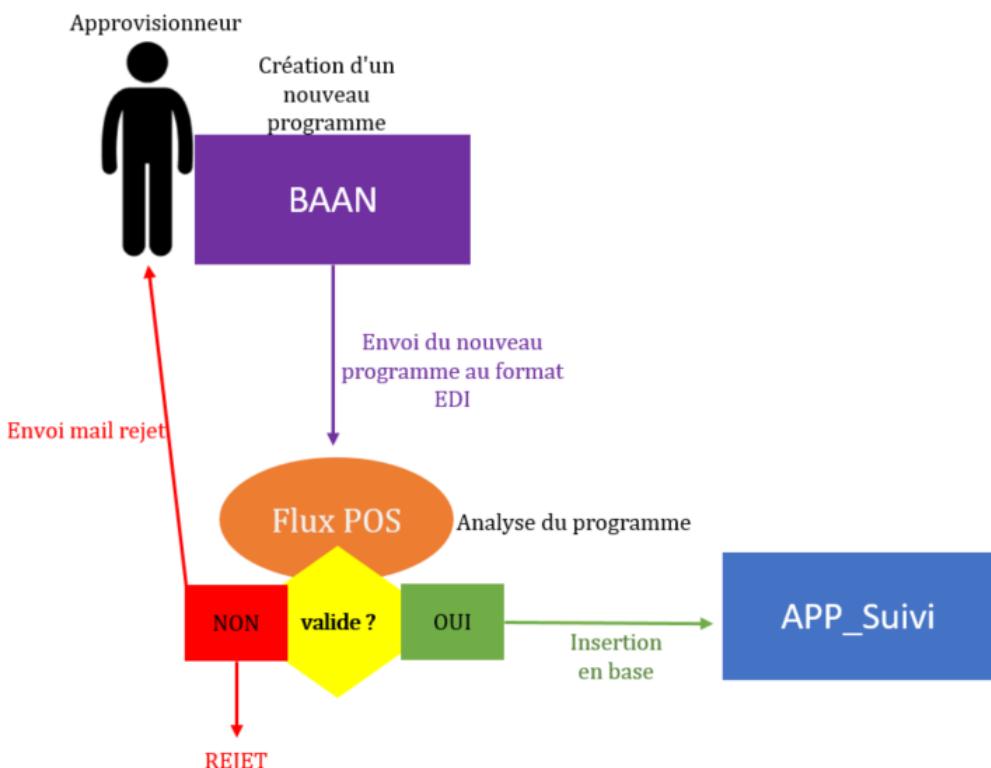


Figure 13- Schéma explicatif Flux POS

Dans le flux POS se trouve une fonction qui analyse les erreurs de transit en base de données pour les échéances du contrat renseigné dans le programme envoyé. S'il y en a, nous créons un incident et un ticket sur Jira pour les traiter.

3.4. Produire du code

J'en arrive à ma mission préférée : concevoir et produire du code.

Au cours de cette année, j'ai participé à plusieurs missions de développement :

- **Release 10.58 - FEB 2282** : Ce fut ma première mission, je devais rajouter la référence article des contrats dans l'objet des e-mails d'alertes du flux POS car c'était beaucoup plus pratique pour les approvisionneurs qui n'avaient plus l'obligation d'ouvrir le message pour la consulter.

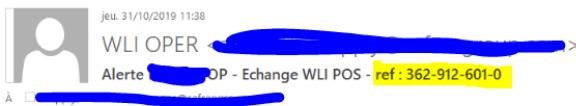


Figure 14- Impression écran d'un e-mail suite au développement effectué (1)

Je devais également rajouter la référence article du contrat dans le corps des e-mails ainsi que rajouter l'état de la validation de l'expédition pour les déclaration d'expédition en SiàSi* des fournisseurs V2 dans l'objet des e-mails envoyés.



Figure 15- Impression écran d'un e-mail suite au développement effectué (2)

Cette première mission m'a permis de prendre mes premières marques dans le code, l'estimation du reste à faire pour Jira et la rédaction des tests dans notre outil ALM.

- **Release 10.59 – FEB 2346** : Mon premier gros développement. Il s'agissait de la refonte complète du code qui gère l'intégration des programmes sur les contrats multisites* du flux POS (voir figure 13). En effet, il y avait beaucoup de blocs de code dupliqués, compliqués et peu clairs qui généraient beaucoup d'incohérences en base de données. Il a donc été décidé avec l'accord du client qu'on allait revoir entièrement cette partie qui produisait un nombre d'incidents conséquents (notamment les incohérences de transit). J'ai donc supprimé des centaines de lignes de code et réécrit quelque chose de plus simple, réutilisable et surtout fonctionnel à l'aide des spécifications fonctionnelles

(SFD) et des spécifications techniques (STD). Aujourd'hui, cette release est en production et nous avons nettement moins d'incidents. Les approvisionneurs en sont également très contents ; ils ont gagné beaucoup de temps car ils ne doivent plus passer des heures à vérifier leurs expéditions, restes à livrer, besoins etc. Il est également à noter que, la même refonte a été demandée par la suite, pour les contrats monosites*, et que mon collègue qui s'est chargé du développement a pu réutiliser mon code en intégralité, ce qui nous a permis de réduire, à nouveau, le surplus de code et de le rendre plus lisible et maintenable.

- **Release 10.61 - Problème 1579 :** Ce problème, qui normalement devait être une évolution car rien dans les spécifications fonctionnelles ne le prévoyait, consistait à faire fonctionner le split de serials pour les fournisseurs V1 (non soumis aux contrôles qualité) en SiàSi (la déclaration d'expédition se fait via l'envoi d'un fichier xml au lieu d'utiliser le portail APP_Suivi). Rien dans les SFD ne le prévoyait car normalement les fournisseurs V1 n'utilisent pas de serials pour leurs expéditions. Cependant, certains en utilisent malgré tout. Le code n'existe pas et vu qu'il s'agissait d'un problème, il n'y avait pas de chiffrage et donc pas de rédaction de SFD et de STD. J'ai beaucoup aimé ce développement car j'ai dû imaginer moi-même un algorithme* pour que cela fonctionne. J'ai également découvert le flux DESPATCH ADVICES :

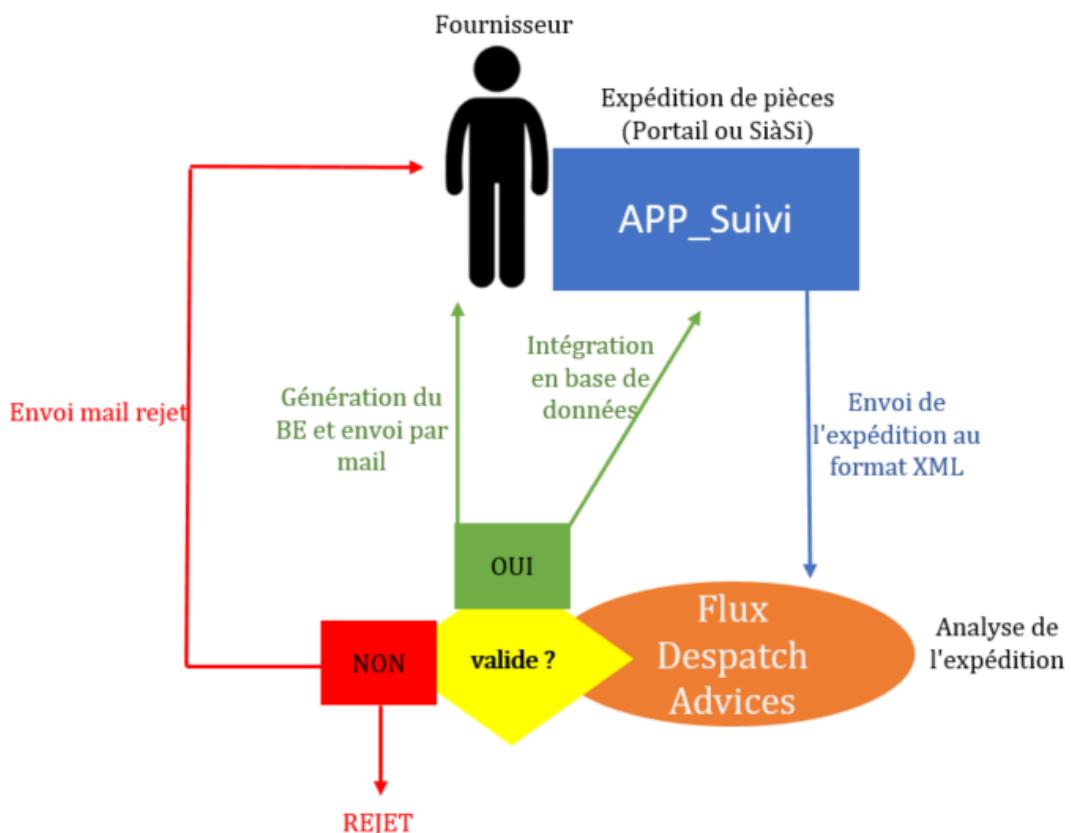


Figure 16- Schéma explicatif Flux Despatch Advices



De plus, j'ai été amenée à modifier la génération du bon d'expédition (format PDF) envoyé au fournisseur suite à la validation de son expédition pour qu'il reflète le split de serials, c'est-à-dire : lister les serials pour chaque numéro de pré-réception créé.

Comme vous pouvez le voir sur les figures 17, 18 et 19, suite à la déclaration d'une expédition de 800 pièces sérialisées, des numéros de pré-réception ont bien été créés pour effectuer les lots de 350 pièces. Le dernier numéro possédant le reste.

N°BE : 204315
N° déclaration de conformité : STI31452
Date d'expédition (JJ/MM/AAAA) : 02/10/2019

N°BE : 204315
N° déclaration de conformité : STI31452
Date d'expédition (JJ/MM/AAAA) : 02/10/2019

Article	Description	Quantité	Code ERP	N° Contract / ligne	N° PO / ligne
362-094-111-0	RESERVOIR_HUILE	350	920	24018 / 1	

Article	Description	Quantité	Code ERP	N° Contract / ligne	N° PO / ligne
362-094-111-0	RESERVOIR_HUILE	350	920	24018 / 1	

N° [REDACTED] 4436355
Date du besoin (JJ/MM/AAAA) : 06/09/2019

N° [REDACTED] 4436368
Date du besoin (JJ/MM/AAAA) : 06/09/2019



99204436355



99204436368

Figure 18 - Bon expédition 800 pièces - partie 1

Figure 17- Bon expédition 800 pièces - partie 2

N°BE : 204315
N° déclaration de conformité : STI31452
Date d'expédition (JJ/MM/AAAA) : 02/10/2019

Article	Description	Quantité	Code ERP	N° Contract / ligne	N° PO / ligne
362-094-111-0	RESERVOIR_HUILE	100	920	24018 / 1	

N° [REDACTED] 4436369
Date du besoin (JJ/MM/AAAA) : 06/09/2019



99204436369

N° Serial	Quantité	N° Coulis
TOTO1	1	
...
TOTO350	1	

Figure 19 - Bon expédition 800 pièces - partie 3

Actuellement, ce code a été testé en intégration par notre analyste fonctionnelle qui m'a validé qu'aucun retour n'est à faire. Il a également été validé en recette par le métier et sera donc mis en production lors de la MEP* de la release 10.61.



- **Release 10.61 - Problème 3117 :** Ce problème concerne le flux DESPATCH ADVICES et le Batch de contrôle qualité des déclarations d'expédition des fournisseurs V2. Lorsque ceux-ci déclarent des expéditions de pièces, elles ne sont pas intégrées directement dans APP_Suivi. Elles sont d'abord transférées dans une table dite 'tampon' - ou temporaire - en attendant le passage du batch qui va effectuer tous les contrôles qualités. Si ces contrôles s'avèrent réussis, l'expédition est intégrée dans APP_Suivi et supprimée de la table temporaire. Sinon, un e-mail est envoyé au fournisseur et l'expédition demeure en table tampon tant que celui-ci ne la débloque pas sur le portail APP_Suivi. Parfois, ils ne le font pas et les expéditions KO s'accumulent dans la table ce qui peut les bloquer s'ils veulent effectuer d'autres expéditions. En effet, APP_Suivi fonctionne de sorte qu'on ne puisse pas expédier de pièces pour une échéance postérieure à une autre qui a toujours du besoin. Ainsi, on devait régulièrement vider manuellement cette table pour les débloquer (cela constituait un incident à la demande du métier). Ma mission a donc été de mettre en place une vérification de toutes les expéditions datant de plus de x jours pour les supprimer automatiquement et d'externaliser ce délai dans les paramètres du site APP_Suivi pour qu'il puisse être changé au besoin par le métier.

De plus, ce code a été réalisé sur des fichiers qui sont analysés par SonarQube, notre logiciel de mesure de qualité du code. En effet, nous sommes obligés de garantir une qualité de code à notre client, c'est-à-dire que nous ne devons pas dégrader l'existant. Ces analyses nous permettent donc de montrer au client la qualité de notre travail, voire mieux, l'amélioration de l'état du code depuis que nous nous en occupons. Sur la figure 20, vous pouvez constater que mon code n'a eu aucun impact négatif sur la qualité existante :

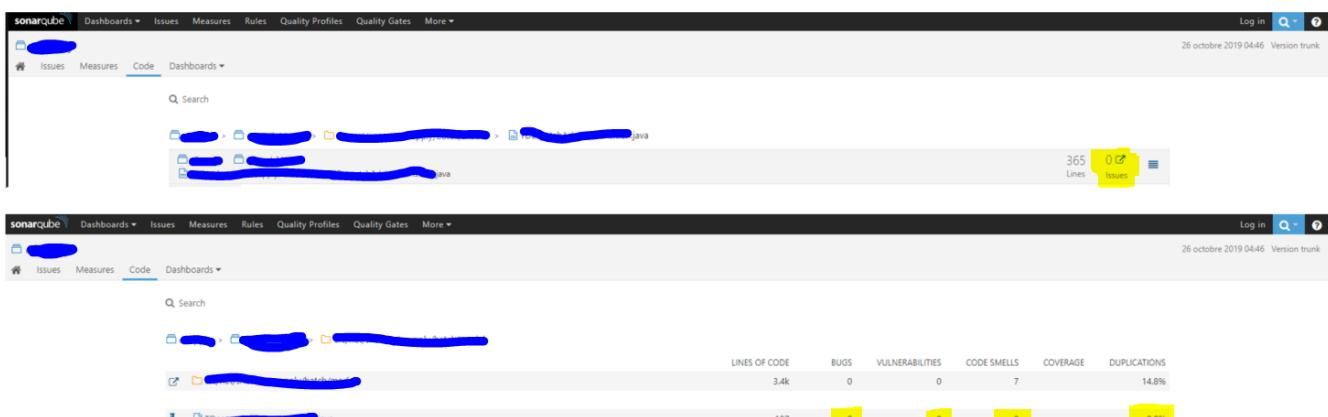


Figure 20- Résultat analyse SonarQube suite au commit de mon code

- **FEB 3119 :** Cette fonctionnalité n'a pas encore constitué un développement de ma part mais une analyse et un chiffrage. Actuellement, tous les fichiers qui sont traités par les différents flux sont envoyés sur un serveur de fichiers. Nous avons des jobs VTOM* qui se chargent d'effectuer tous les jours à des intervalles réguliers des traitements sur ces fichiers. Suite à un dysfonctionnement qui est survenu et qui a laissé les fichiers s'accumuler sur le serveur, nous nous sommes rendu compte que Baan ne peut pas supporter un trop grand afflux (ce qui est arrivé lorsque tout est rentré dans l'ordre) et que les requêtes du batch n'étaient également pas optimisées pour traiter une trop grande quantité de fichiers. Il m'a donc été demandé d'analyser le script qui se charge de déplacer les fichiers afin d'y ajouter une limitation à 300 fichiers par traitement et d'effectuer le chiffrage qui sera proposé au client dans le cadre d'une fiabilisation.

Macro-tâche	Sous-tâche	Tâche élémentaire	Type Module	Complexité Module	Type Modification	Nombre de Modifications	Charge DEV + TU
Description							
	Filtrage des fichiers	Suppression du test actuel qui tente de récupérer tous les fichiers commençant par [REDACTED]	ShellUnix	M	MS	1	0,1
		Filtre sur les 300 fichiers qui commencent par [REDACTED] et qui sont les plus anciens	ShellUnix	M	MS	1	0,4
		Filtre sur les fichiers qui commencent par [REDACTED], qui ne contiennent pas dans leur nom les codes [REDACTED] et [REDACTED] et qui sont les plus anciens	ShellUnix	M	MS	1	0,4
		Test qui vérifie que la récupération des 300 fichiers ne génère aucune erreur	ShellUnix	M	MS	1	0,4
	Copie des 300 fichiers	Création d'une boucle pour récupérer les 300 fichiers	ShellUnix	M	MS	1	0,2
		Copie des 300 fichiers récupérés de [REDACTED] vers [REDACTED]	ShellUnix	M	MS	1	0,4
		Copie temporaire des 300 fichiers à traiter dans le dossier [REDACTED]	ShellUnix	M	MS	1	0,4
	Création des fichiers liste.txt et erexc.liste.txt	Insertion dans liste.txt et [REDACTED].liste.txt des fichiers, parmi les 300 traités qui commencent par [REDACTED] qui ne contiennent pas les codes [REDACTED] et [REDACTED]	ShellUnix	M	MS	1	0,4
	Création des fichiers liste1.txt, liste2.txt, liste3.txt	Insertion dans liste1.txt des fichiers, parmi les 300 traités qui correspondent au pattern [REDACTED]	ShellUnix	M	MS	1	0,2
		Insertion dans liste1.txt des fichiers, parmi les 300 traités qui correspondent au pattern [REDACTED]	ShellUnix	M	MS	1	0,2
		Insertion dans liste1.txt des fichiers, parmi les 300 traités qui correspondent au pattern [REDACTED]	ShellUnix	M	MS	1	0,2
	Post-traitements	Déplacement des listes dans [REDACTED] avec remplacement des existantes + Suppression du contenu du dossier [REDACTED]	ShellUnix	M	MS	1	0,2

Figure 21- Détail analyse technique pour chiffrage FEB 3119

- **Release 10.62 - FEB 2628 :** Ce développement est le plus récent que j'ai effectué, il date de fin octobre 2019 et consiste en l'ajout d'une fonctionnalité de répartition automatique des besoins suite à la déclaration d'un nouveau programme par un approvisionneur. Cela concerne donc le flux POS. C'est le développement que j'ai décidé de détailler plus loin dans ce document.

4. Projet réalisé en entreprise

Comme énoncé précédemment, j'ai choisi de présenter mon développement effectué sur la FEB 2628 de la release 10.62 d'APP_Suivi car c'est celui qui, selon moi, regroupe le plus de compétences.

4.1. Contexte

Sur l'application APP_Suivi, les fournisseurs déclarent des expéditions pour envoyer des pièces. Ces pièces, selon leur type, possèdent une référence. Chaque référence est liée à un numéro de contrat et pour chaque contrat est liée une quantité de besoin que le fournisseur doit satisfaire. Il peut soit envoyer toutes les pièces en une fois soit petit à petit, le besoin se retrouve donc décrémenté à chaque expédition.

Afin de satisfaire le besoin d'un contrat, le fournisseur a plusieurs échéances à respecter pour l'envoi des pièces. Par exemple, si le contrat n°142 a un besoin total de 1000 pièces, voici comment elles peuvent se répartir :

- Echéance du 01/02/2020 : 250 pièces attendues
- Echéance du 01/08/2020 : 600 pièces attendues
- Echéance du 01/10/2020 : 150 pièces attendues

Chaque échéance a donc son propre besoin qui correspond au reste à livrer (RAL).

Le besoin d'un contrat peut évoluer dans le temps. Les fournisseurs passent donc des programmes (POS) sur Baan qui sont ensuite envoyés sur APP_Suivi. Un nouveau programme peut donc être utilisé pour diminuer un besoin, l'augmenter, supprimer un site de livraison... Les POS sont liés à un numéro de contrat, une ligne de contrat et un code baan (ou code ERP, représente les différentes catégories selon l'utilisation des pièces).

Les POS ont des échéances qui peuvent être fermes ou flexibles. Les échéances fermes sont les échéances dont le besoin a été négocié et validé et qui ne peut plus être changé. Les échéances flexibles (souvent les plus éloignées dans le temps) sont les échéances pour des besoins qui peuvent encore être modifiés.

Les pièces peuvent être expédiées sur différents sites de livraisons. Un contrat est dit monosite s'il ne dispose que d'un seul site de livraison et à l'inverse, s'il dispose de plusieurs sites de livraison il est dit multisites. Les contrats multisites possèdent toujours un site de livraison par défaut, c'est en général celui où la majorité des pièces sont envoyées.

Lorsqu'un fournisseur passe un nouveau programme, cela génère un fichier au format EDI* qui contient toutes les informations nécessaires à son intégration sur APP_Suivi (voir figure n° 22). Le fournisseur passe son nouveau programme via Baan qui génère un fichier EDI et l'envoie au flux POS. Celui-ci analyse le programme. S'il contient des erreurs, le programme est rejeté et un e-mail est envoyé à l'approvisionneur pour qu'il corrige son programme et le renvoie. Si le

programme est valide, il est inséré dans la base de données APP_Suivi et sera consultable sur le site.

Ces fichiers EDI sont aussi appelés des fichiers "ESNPOS" car ce sont leurs noms lorsqu'ils sont créés (respect de la charte de nommage mise en place dans les spécifications fonctionnelles)

Voici à quoi ressemble un fichier ESNPOS et son contenu :

```
1192018122601308199304019201WLTIESY|ESNPOS||20190115|1441
219201812260130819930401201901151330-005-505-61V
31920181226013081993040|EA12729|5187671
419201812260130819930401
519201812260130819930401
719201812260130819930401
819201812260130819930401LD011
919201812260130819930401201808021110
919201812260130819930401201808031110
919201812260130819930401201808041110
919201812260130819930401201808051110
919201812260130819930401201808061110
919201812260130819930401201808071110
919201812260130819930401201808081110
919201812260130819930401201808091117
919201812260130819930401201808101110
919201812260130819930401201808111110
919201812260130819930401201808121110
919201812260130819930401201808131110
919201812260130819930401201808141110
919201812260130819930401201808151110
919201812260130819930401201808161117
919201812260130819930401201808171110
919201812260130819930401201808181110
919201812260130819930401201808191110
919201812260130819930401201808201110
919201812260130819930401201808211110
919201812260130819930401201808221110
919201812260130819930401201808231117
919201812260130819930401201808241110
```

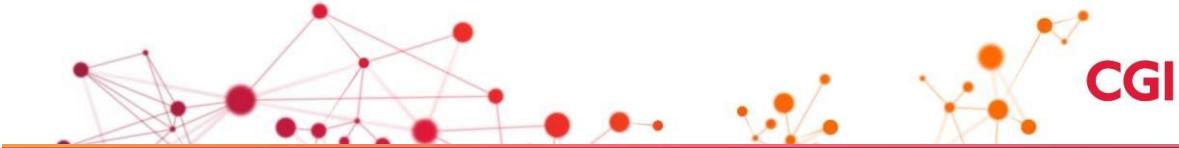
Figure 22- Structure fichier ESNPOS

Le premier numéro tout à gauche est le numéro de ligne. On répète ce numéro lorsque plusieurs données sont renseignées.

Je vais décrire les données essentielles à pour chaque ligne utilisée dans le cadre de cette évolution :

- La ligne 1 : contient les informations sur le POS en cours, le code baan et la date/heure de création.
- La ligne 2 : contient la référence article, sa description, s'il s'agit d'un article sérialisé, le n° du POS précédent...
- La ligne 3 : contient le numéro de contrat, numéro de ligne du contrat, le numéro de contrat achat Baan...
- La ligne 4 : contient les informations relatives au fournisseur : son code, son nom, son adresse...
- La ligne 5 : contient les informations de l'approvisionneur*
- La ligne 8 : contient les informations relatives au site de livraison. Il y a autant de ligne 8 que de sites de livraisons. Autrement dit, un contrat multisites possèdera plusieurs lignes 8.
- La ligne 9 : contient toutes les échéances, leur type (ferme, flexible) et leur besoin. Il y a autant de lignes 9 qu'il y a d'échéance avec un besoin pour un contrat.

En plus des rejets sur la structure du fichier en elle-même, le flux POS fait d'autres contrôles pour vérifier la cohérence des données du fichier avec celles en base avant d'en faire l'intégration. Certains cas provoquent des rejets du fichier et aucune intégration ne se fait. L'approvisionneur devra corriger son programme et le repasser.



Ces cas de rejet actuels sont les suivants :

Pour une échéance ferme :

- Un site (de livraison) sur lequel il existait des pièces en transit n'a pas été repris dans le fichier
- Le besoin (reste à recevoir) est inférieur à la quantité actuellement en transit en base de données.

Si le fichier n'est pas rejeté, des contrôles sont effectués avant l'intégration du nouveau POS sur chaque ligne 9 du fichier EDI :

Pour chaque échéance fermes et flexibles :

- Vérifications des écarts de transit en base de données (non bloquant, envoie un mail à l'approvisionneur pour prévenir).
- Pour chaque nouveau site présent dans le fichier ESNPOS, un ajout est effectué en base pour le renseigner.
- Vérification du site par défaut.

L'intégration d'un fichier ESNPOS dans APP_Suivi consiste en plusieurs chose :

- Clôturer le POS N-1 (celui qui le précède) et intégrer le POS N (le POS du fichier) dans la table prévue à cet effet (T_POS).
- Mettre à jour la liste des sites de livraison selon celle renseignée dans le fichier, en indiquant le site par défaut dans la table prévue à cet effet (T_AVAILABLE_SITES).
- Renseigner les besoins de chaque échéance pour le nouveau POS dans la table prévue à cet effet (je l'appellerai T_LINES_POS).
- Renseigner les sites et leurs besoins/transits dans la table prévue à cet effet (T_QUANTITIES).
- Mettre à jour les besoins restants dans la table prévue à cet effet (T_SHIPMENT_ADVICES).

4.2. Explications de l'évolution, définition du besoin

Pour l'évolution qui m'a été demandée, on veut rajouter comme fonctionnalité une **répartition automatique** des besoins sur les différents sites de livraison d'un contrat. L'évolution concerne donc les POS multisites.

Actuellement, les approvisionneurs passent un premier programme suite auquel tout le besoin est mis sur le site par défaut. Ils doivent ensuite aller sur le site APP_Suivi pour préciser selon les sites, la quantité qu'ils veulent (sous condition que la somme des besoins de tous les sites soit égale au besoin pour l'échéance). Beaucoup d'approvisionneurs envoient toujours le même pourcentage de pièces sur leurs sites de livraison et ont demandé à renseigner ce pourcentage une seule fois pour ne plus avoir à le faire.

L'approvisionneur renseignera un pourcentage de répartition par contrat et par site de livraison ainsi qu'un multiple de conditionnement* (si nécessaire) qui serviront à chaque envoi de programme de calculer la répartition par site.

Contrat	Code site de livraison	Nom Site	Site par défaut	Pourcentage	Multiple conditionnement
452000	LD0	Corbeil	Oui	70	7
452000	M01	Famat	Non	30	7

Le multiple de conditionnement est utilisé pour certains contrat pour lesquels les pièces sont toujours envoyées par lots.

En base, le pourcentage sera renseigné dans la table T_AVAILABLE_SITES. Le multiple de conditionnement sera renseigné dans la table T_SCH_CONTRACT.

L'évolution #2628 pour la release 10.62 disposera donc d'une nouvelle fonctionnalité sur l'IHM APP_Suivi ainsi que de nouveaux traitement dans le flux POS.

Je m'occupe des nouveaux traitements dans le flux POS. Il faudra que celui-ci puisse répartir la quantité totale du besoin présent dans le ESNPOS sur chacun des sites à partir des données renseignées dans la table T_AVAILABLE_SITES ainsi que du multiple de conditionnement s'il existe. Si aucune répartition n'est présente dans cette table, elle se fera comme actuellement.

Actuellement, si le besoin du nouveau programme (POS N) est différent de ce qu'il y a actuellement en base, ou si le site par défaut a été modifié, on ne conserve que les transits (pièces en transit, qui ne sont pas réceptionnées) sur les sites et on met le reste sur le site par défaut. Si le besoin est le même, on conserve la répartition telle qu'elle est.

Enfin, pour nous faciliter le traitement des éventuels futurs incidents relatifs aux transits, nous voulons que tous les besoins répartis automatiquement ou manuellement soient répertoriés dans une table de la base de données.

4.3. Spécifications fonctionnelles (SFD)

Mon équipe ayant une analyste fonctionnelle, les spécifications fonctionnelles pour cette évolution m'ont été fournies. Cependant, comme expliqué plus haut dans la partie 3.4, tous les développements que j'ai faits n'ont pas eu de spécifications fonctionnelles (notamment les problèmes) et c'était à moi de rédiger les requirements à respecter sur notre logiciel de tests (ALM). Les requirements sont l'expression des règles de gestion qu'un code doit satisfaire.

Voici un résumé des spécifications fonctionnelles qui m'ont été données (les schémas ont été fournis par le client) :

Le schéma suivant (Figure 23) indique quel processus suivre selon la présence ou non des pourcentages de répartitions pour les sites d'un contrat dans la table T_AVAILABLE_SITES.

Règles de gestion des multi-sites

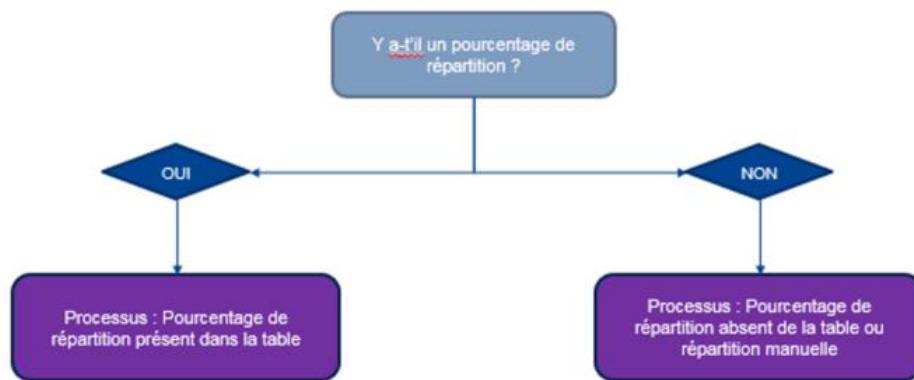


Figure 23- Règles de gestion des multi-sites

La figure 24 représente les règles à suivre selon différents cas de figure dans le cadre d'une répartition automatique

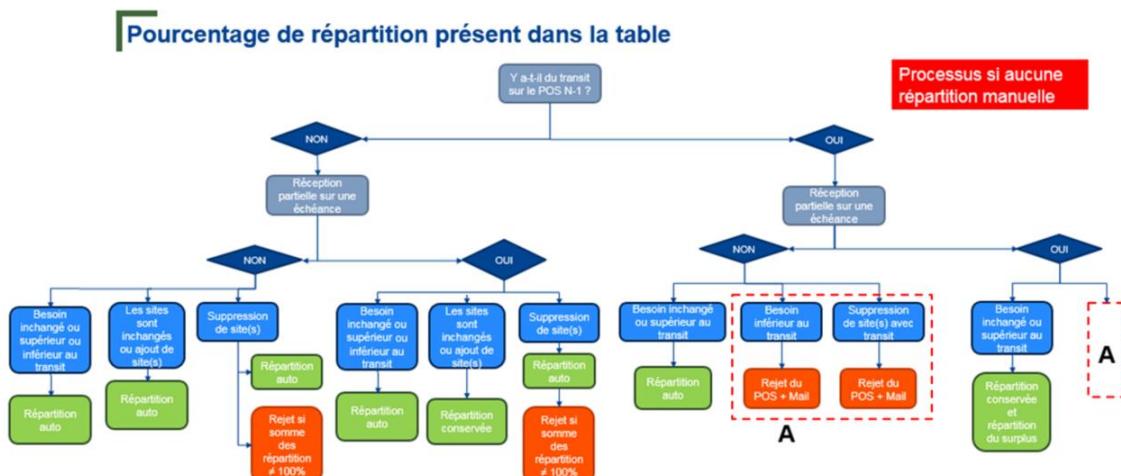


Figure 24- Processus à suivre pour la répartition automatique

La figure 25 représente les règles à suivre selon différents cas de figure dans le cadre d'une répartition manuelle ou non initialisée. C'est la méthode de répartition actuellement en production.

Pourcentage de répartition absent de la table ou répartition manuelle

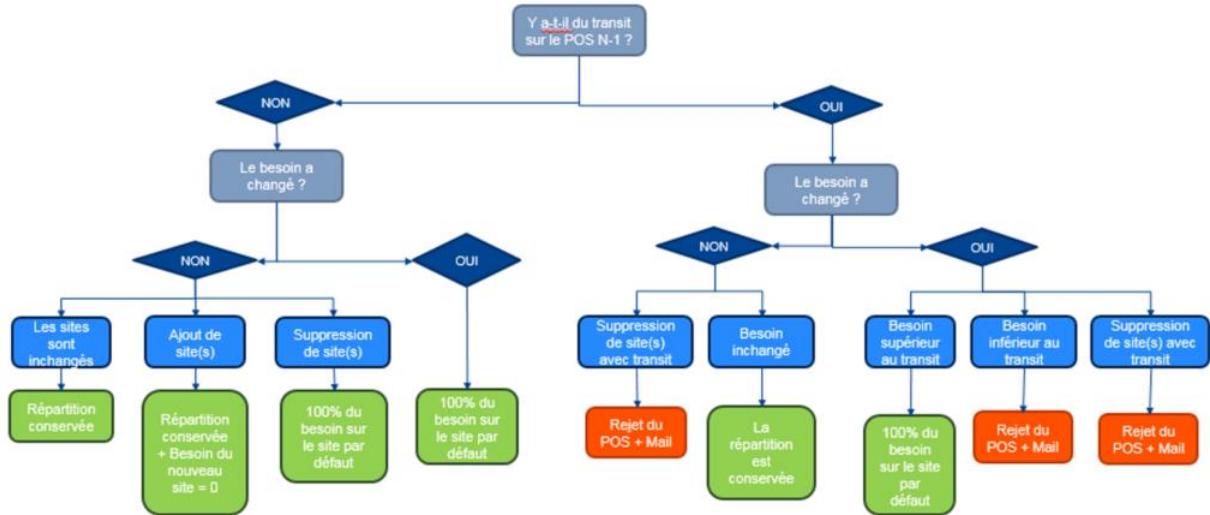


Figure 25- Processus à suivre pour la répartition manuelle ou non initialisée

Voici les différentes règles de gestion qui m'ont été données de respecter :

- RG 1 :** La répartition automatique se fera que si les échéances n'ont pas été modifiées manuellement par l'approvisionneur.
- RG 2 :** La répartition automatique ne se fera que pour les échéances dans le ferme.
- RG 3 :** Le nombre de pièce après répartition doit être un entier.
- RG 4 :** Si le besoin renseigné pour une échéance n'est pas un multiple du multiple de conditionnement renseigné pour le contrat, le message est rejeté.
- RG 5 :** Méthode de calcul de la répartition
Dans le cas où pour une quantité de besoin, l'application des pourcentages de répartition engendre des valeurs décimales : Arrondir les valeurs à l'inférieur et de rajouter la pièce manquante au site par défaut.

Exemple : besoin 462 pièces

site 1 (site par défaut) - 60% : 277,2 => 277
site 2 - 30% : 138,6 => 138
site 3 - 10% : 46,2 => 46

Total : 277 + 138 + 46 = 461, il manque une pièce qui sera mise sur le site par défaut

Figure 26- Règles de gestion partie1



- **RG 6 :** Méthode de calcul de la répartition avec multiple de conditionnement

Dans le cas d'un contrat avec multiple de conditionnement et que les pourcentages entraînent des valeurs décimales, on applique la même règle que la RG 4 et on ajoute toutes les pièces manquantes sur le site par défaut.

Exemple : besoin 161 pièces - multiple de conditionnement : 7

site 1 (site par défaut) - 45% : 72,42 => 72

site 2 - 25% : 40,25 => 40

site 3 - 30% : 48,3 => 48

on applique le multiple de conditionnement à l'inférieur :

site 1 : 70

site 2 : 35

site 3 : 42

=> 70+35+42 = 154 manque 7 pièces

Il manque 7 pièces que l'on rajoute sur le site par défaut => site 1 : 77 / site 2 : 35 / site 3 : 42

- **RG 7 :** Dans le cas de suppression de site dans un programme, si un pourcentage de répartition lui est affecté le message est rejeté et l'approvisionneur devra enlever la répartition avant renvoie de son programme.

- **RG 8 :** Si un nouveau site est ajouté dans un programme, une répartition de 0 lui sera affectée.

Figure 27- Règles de gestion partie 2

Les règles suivantes ne sont pas présentes dans la liste mais font partie des spécifications fonctionnelles complètes pour le flux POS :

- La somme des quantités pour chaque échéance doit être égale à 100% du besoin
- Historisation des répartitions automatiques et manuelles

L'anticipation suivante a été mise en œuvre suite à mon analyse :

- Une requête SQL sera effectuée suite à la mise en AP et la mise en OP afin de passer toutes les échéances actuellement réparties manuellement en répartition MANUELLE. Si on ne le fait pas, nous aurons des rattrapages en masse à effectuer dans le cadre de nouveaux incidents car lorsque les fournisseurs activeront la répartition automatique d'un contrat, toutes les échéances passeront en répartition automatique ce qui aura pour résultat de casser les répartitions manuelles mises en place avant la MEP au bénéfice de la répartition automatique.

Enfin, les éléments de base de données à modifier/créer :

Modification de plusieurs tables :

- [REDACTED] :
Ajoute de la colonne [REDACTED] : pourcentage de répartition affecté au site
- [REDACTED] :
Ajout de la colonne [REDACTED] : multiple de conditionnement attribué au contrat
Ajout d'un flag pour savoir si un contrat a été inactivé ou non par l'approvisionneur
- [REDACTED] :
Ajout d'un flag pour spécifier si la répartition a été faite manuellement
- [REDACTED] :
Ajout d'un flag [REDACTED] pour identifier les manageurs approvisionneur « Convergent »

Figure 28- Modifications à faire en base de données - partie 1

Colonne	Type	Défaut	Desc
...	NUMBER NOT NULL PRIMARY KEY		ID générée par la séquence
...	DATE NOT NULL		
...	VARCHAR2(35 BYTE)		
...	VARCHAR2(35 BYTE)		
...	VARCHAR2(35 BYTE) NOT NULL		SYSTEM s'il s'agit d'une modification effectuée par le flux POS
...	NUMBER NOT NULL		
...	NUMBER NOT NULL		
...	VARCHAR2(3 BYTE) NOT NULL		
...	DATE		Nul lors de l'historisation du multiple de conditionnement ou du premier paramétrage de la répartition auto
...	VARCHAR2(35 BYTE) NOT NULL		
...	NUMBER		Non nul s'il existe une entrée pour le même contrat et le même site
...	NUMBER		Nul lors de l'historisation du multiple de conditionnement
...	NUMBER		Non nul s'il existe une entrée pour le même contrat et le même site
...	NUMBER		Nul lors de l'historisation du multiple de conditionnement
...	NUMBER		Non nul s'il existe une entrée pour le même contrat et le même site
...	NUMBER NOT NULL		
...	CHAR(1 BYTE) NOT NULL	0	
...	VARCHAR(5 BYTE) NOT NULL		'AUTO' ou 'MANUAL'

Figure 29 - Modifications à faire en base de données - partie 2

Les modifications suivantes ont été apportées suite à mon analyse :

- ⇒ Le VARCHAR2(5 BYTE) de la dernière ligne a été changé en VARCHAR2(6 BYTE) car pour contenir la valeur « MANUAL », il faut 6 BYTE.
- ⇒ Le NUMBER NOT NULL de la 3^{ème} ligne en partant du bas est passé à NUMBER car le multiple de conditionnement peut être NULL dans la table T_SCH_CONTRACT.

4.4. Spécifications techniques (STD)

Voici les spécifications techniques qui m'ont été données comme support pour la réalisation de l'évolution. Elles ont été rédigées par notre référent technique Benjamin BARBE à la demande de notre CP.

Cependant, vous pourrez trouver dans la partie 4.5.2 mon analyse technique ainsi que mes algorithmes et fonctions pensées pour cette évolution.

Algorithme de répartition

Généralités

Le code modifié sera principalement dans `ControlCtrlImpl.jcs`.

- `verifierRejetPOS` : Deux nouveaux cas de rejet à intégrer si le POS est en répartition auto :
 - La somme des pourcentages en base est différente de 100%
 - Pour le contrat, faire la somme des `T_AVAILABLE_SITES.POURCENTAGE_DISTRIBUTION`
 - Pour une des échéances du ESNPOS, la quantité n'est pas un multiple du multiple de conditionnement
 - Pour la date, faire `besoin MOD multiple`
 - S'il n'y a pas de multiple, ne pas contrôler

La répartition automatique concernant uniquement les POS fermes, les rejets sont à ajouter uniquement sur les POS fermes : la fonction `verifierRejetPOS` est utilisée uniquement pour les POS fermes.

Il sera nécessaire de créer des fonctions de requêtage, dans le fichier `ESupply.jcx` :

- Requête effectuant la somme des pourcentages pour un contrat
- Requête de récupération des sites/pourcentages
- Requête de récupération du multiple de conditionnement

Figure 30- STD partie 1

⇒ A noter que pour le premier cas de rejet, il faut bien vérifier les données en base par rapport aux sites présents dans le fichier `ESNPOS`.

Globalement, le fonctionnement sera le suivant :

- On initialisera un booléen permettant de déterminer si l'échéance doit être répartie automatiquement :
 - `T_LINES_POS.DISTRIBUTION_TYPE == "AUTO"`
- On isole tous les cas de rejets en premier (besoin < transit, site en transit supprimé, somme des répartitions < 100%). Les nouveaux cas de rejets sont contrôlés si l'échéance est en mode AUTO
- S'il y a des réceptions partielles, on conserve la répartition. L'éventuel surplus sera réparti automatiquement
- S'il n'y a pas de réception partielle, on part directement sur la répartition automatique

Les rejets sont à ajouter avec le code de rejet déjà existant.

Figure 31- STD partie 2

Pour l'action d'historisation et pour récupérer les données (`distribution_before_action`, `poucentage_before_action`, `conditionning_after_action`, etc) on a plusieurs solutions :

- Une requête qui select tout d'un coup (attention à utiliser `NVL` pour éviter les null)
- Tout fusionner en un `INSERT ... SELECT` où on aurait plus qu'à passer contrat/date (et possiblement d'autres valeurs) en paramètre pour gérer en une seule instruction l'historique

Figure 32- STD partie 3

4.5. Pré-rédaction des tests ALM

Au sein de mon service, nous effectuons la rédaction des tests avant l'écriture du code car cela nous permet de nous assurer que nous avons bien compris ce qui est à faire et à respecter.

4.5.1. Requirements

Voici la liste des requirements que j'ai rédigés sur ALM suite à la prise de connaissance des SFD et des STD :

- La répartition automatique ne doit se faire que si les échéances n'ont pas été modifiées manuellement
⇒ *La répartition automatique s'applique sur toutes les échéances d'un contrat tandis que la répartition manuelle ne concerne qu'une échéance ou plusieurs échéances. Sur un même contrat on peut donc avoir 3 échéances manuelles et le reste en automatique.*
- La répartition automatique ne se fait que sur les échéances fermes
- Le nombre de pièces après répartition doit être un entier
- Si le besoin renseigné n'est pas un multiple du multiple de conditionnement, le message est **rejeté**
- Si l'application des pourcentage de répartition engendre des valeurs décimales elles doivent être arrondies à la valeur inférieure et le surplus devra être mis sur le site par défaut
- Dans le cas d'un pourcentage de répartition et d'un multiple de conditionnement, on ajoute les pièces restantes sur le site par défaut
- Si un site est supprimé et qu'un pourcentage de répartition lui était affecté, le message est **rejeté**
- Si un nouveau site est ajouté, une répartition de 0 lui est affectée.
- Pour chaque échéance, ferme et en répartition automatique ou manuelle, traitée dans le ESNPOS, une ligne par site doit être insérée dans T_DISTRIBUTION_HISTORY

4.5.2. Test plan

Afin de rédiger mon test plan, je pense d'abord à l'algorithme général que j'appliquerai au code ainsi qu'aux fonctions que je vais créer.

4.5.2.1. Cas de rejets

Actuellement dans le code, il y a déjà deux cas de rejets qui sont gérés dans la fonction `verifierRejetPOS()`. Les vérifications des nouveaux cas de rejets de cette évolution seront ajoutées à cette fonction afin qu'elles soient toutes rassemblées.

Le premier cas de rejet, « Si un site est supprimé et qu'un pourcentage de répartition lui était affecté, le message est rejeté » ne concerne que les contrats qui ont des échéances en distribution automatique.

- ⇒ Une requête SQL devra être créée pour récupérer le pourcentage de distribution d'un site.

Si la valeur renournée est > 0 alors la fonction suivante sera appelée :

- `verifierDistribution()`

Elle calculera la somme des pourcentages de distribution en base de données pour tous les sites présents dans le fichier ESNPOS.

- ⇒ Une requête SQL devra être créée pour récupérer la somme des pourcentages de distribution des sites concernés.

Si le résultat est de 100, dans ce cas rien n'est appliqué, le traitement poursuit son cours.

Si le résultat est différent de 100, un e-mail sera envoyé et une `POSRejectedException` (Exception que j'ai créée lors d'une précédente évolution) sera lancée pour interrompre le traitement.

Le deuxième cas de rejet, « Si le besoin renseigné n'est pas un multiple du multiple de conditionnement, le message est rejeté », se fera sur chaque ligne 9 présentes dans le ESNPOS.

- `verifierMultiple()`

Cette fonction prendra en paramètre le reste à recevoir indiqué dans le fichier ESNPOS pour une échéance ainsi que la valeur du multiple de conditionnement appliquée au contrat.

Elle ne sera appelée que si le contrat en possède un !

- ⇒ Une requête SQL devra être créée pour récupérer la valeur du multiple de conditionnement appliquée à un contrat.

Si sa valeur vaut 1, la fonction ne sera pas appelée. Si elle vaut autre chose que 1 (car tout entier est multiple de 1, la répartition n'en sera pas impactée) elle calculera, à l'aide d'un modulo* si le RAR est un multiple du multiple de conditionnement. En effet, si le modulo retourne 0 c'est qu'il n'y a pas de reste et donc le besoin est bien un multiple.

Si c'est un multiple, le traitement suit son cours. Si ce n'est pas le cas, un e-mail sera envoyé à l'approvisionneur et une `POSRejectedException` sera lancée pour interrompre le traitement.

4.5.2.2. E-mails

Deux nouveaux e-mails sont à envoyer aux approvisionneurs. Deux nouvelles fonctions seront donc à créer pour les envoyer :

- buildMessageDistributionsNotCorrect()
- buildMessageRarIsNotMultiple()

Ces fonctions auront pour but de générer le message (un en français et un en anglais selon la langue de l'approvisionneur) qui sera envoyé par e-mail. Elles prendront en paramètres toutes les informations nécessaires à la génération de celui-ci.

Par exemple, buildMessageDistributionsNotCorrect() prendra la liste des sites qui ont été supprimés du fichier mais qui possédaient un pourcentage de répartition sur le POS N-1.

Cette liste sera retournée par la méthode getDeletedSitesWithDistribution(). Elle prendra en paramètre les lignes présentes dans la table T_AVAILABLE_SITES pour le POS N-1 et la liste des sites présents dans le fichier ENSPOS. Si un site POS N-1 est présent dans le POS N ou ne possède pas de pourcentage de répartition, il n'est pas considéré comme supprimé. Si un site est considéré comme supprimé il sera ajouté à la liste.

Les messages générés par la fonction seront ensuite envoyés à la fonction sendEmailPosRejected() qui se chargera d'envoyer l'e-mail.

4.5.2.3. Traitement de la répartition

La répartition consiste essentiellement à attribuer les bons transits pour une échéance et un site donné en fonction du besoin, des réceptions et des expéditions non réceptionnées. Il faut également historiser dans la table T_DISTRIBUTION_HISTORY les échéances en distribution automatique et manuelle (un enregistrement par site pour chaque échéance concernée).

En me basant sur les SFD et les STD j'ai donc réalisé l'algorithme suivant :

Dans la fonction actuelle d'intégration des nouvelles lignes dans la table T_QUANTITIES, j'ai rajouté,

```

Si(échéance ferme ET distribution auto){
    performRepartitionAuto() ;
}sinon{
    //Répartition existante (code existant)
    performRepartition() ;
}
Si(distribution auto ou distribution manuelle){
    //Historiser dans la table T_DISTRIBUTION_HISTORY
    Historiser() ;
}

```

La fonction `performRepartition()` sera nouvelle, je mettrai le code existant dedans afin qu'il soit plus lisible.

La fonction `performRepartitionAuto()` aura comme algorithme :

```

Si(réception partielle){
    Si(le RAR ESNPOS < le RAR en base){
        Conserver le transit du POS N-1
    }sinon{
        Conserver les lignes T_QUANTITIES du POS N-1
    }
}sinon{
    Conserver le transit du POS N-1
}
Calcul du delta* entre le RAR du ESNPOS et le RAR des lignes T_QUANTITIES
insérées plus-haut.
Si(delta > 0){
    Pour(chaque site du ESNPOS en commun avec les sites présents dans table
T_QUANTITIES sur le POS N-1 en base){
        Si(Ce n'est pas le site par défaut){
            calculerRepartition(delta, pourcentage, multiple) ;
            Ajout de cette répartition dans la table T_QUANTITIES
            Calcul du delta restant (on soustrait du delta initial ce qui a été
            ajouté à chaque tour de boucle)
        }
    }
    Ajout du delta restant sur le site par défaut.
}

```

Pour respecter les SFD, voici comment j'ai imaginé le calcul de la répartition pour qu'il respecte les pourcentages de répartition de chaque site ainsi que le multiple de conditionnement et l'arrondi à l'entier inférieur :

```

calculerRepartition(delta,pourcentage, multiple){
    result = (delta/100)*pourcentage;
    modulo = result%multiple;
    return (result - modulo);
}

```

On calcule d'abord le besoin restant après application du pourcentage, ensuite on vérifie s'il y a un reste par rapport au multiple de conditionnement en utilisant la fonction modulo '%' et on retourne la différence du premier résultat avec le reste obtenu. De cette façon, l'arrondi à l'entier inférieur se fait grâce au modulo.

Par rapport à la figure 24 des SFD, mis à part les cas de rejets qui sont gérés avant la répartition, on peut distinguer deux grandes règles : soit on effectue une répartition automatique, soit on conserve la répartition actuelle et on répartit l'éventuel surplus.

Il faut partir du principe qu'à partir du moment où il y a eu des réceptions partielles sur une échéance ferme d'un besoin (tout sites confondus), qu'il y ait du transit ou pas, il faut conserver la répartition en l'état. C'est-à-dire, conserver les lignes de la table T_QUANTITIES du POS N-1.

⇒ *Est considéré comme ayant des réceptions partielles un besoin qui, pour une échéance, possède au moins une réception et au moins un besoin supérieur à 0 non réceptionné.*

On conserve donc les répartitions telles qu'elles sont car on ne veut pas les "casser". Cependant, si le besoin du POS N a augmenté par rapport au POS N-1, la différence entre les deux est répartie automatiquement entre les sites selon leur pourcentage de répartition. Le surplus qui peut être engendré sera mis sur le site par défaut. Si le besoin du POS N est inférieur à celui du POS N-1 mais supérieur à la quantité de pièces en transit (obligatoire sinon le POS N est rejeté) alors on ne conserve que les transit et on répartit automatiquement le besoin sur les sites selon le pourcentage de répartition.

Exemples pour illustrer la répartition automatique

Pourcentage de répartition des sites :

VMS (site par défaut)	X05	N09
80%	5%	15%

Cas : réceptions partielles avec transit et augmentation du besoin

Nouveau besoin : **150**

	POS	ECHEANCE	SITE	Reste à livrer	Reste à recevoir	Transit
Avant Répartition	999935	30/08/19	VMS	50	60	10
	999935	30/08/19	X02	10	15	5
	999935	30/08/19	N09	20	40	20

Actuellement le besoin est de $60+15+40 = 115$. Il y a donc 35 pièces à répartir entre les sites.

Pour X02 : $5\% \text{ de } 35 = 1,75$. On arrondit à l'entier inférieur, il reste 1.

Pour N09 : $15\% \text{ de } 35 = 5,25$, on garde 5.

Pour VMS, vu que c'est le site par défaut on lui attribue $35 - 1 - 5 = 29$ pièces.

	POS	ECHEANCE	SITE	Reste à livrer	Reste à recevoir	Transit
Après Répartition	999935	30/08/19	VMS	$50+29 = 79$	$60+29 = 89$	10
	999935	30/08/19	X02	$10+1 = 11$	$15+1 = 16$	5

	999935	30/08/19	N09	$20+5 = 25$	$40+5 = 45$	20
--	--------	----------	-----	-------------	-------------	----

$89+16+45 = 150$, on a bien tout notre besoin.

Cas : réceptions partielles avec transit et diminution du besoin

Nouveau besoin : **50** (doit être supérieur aux 35 pièces en transit pour l'échéance).

Voici l'état de T_QUANTITIES quand on conserve uniquement les transits :

	POS	ECHEANCE	SITE	Reste à livrer	Reste à recevoir	Transit
Conservation des transits	999935	30/08/19	VMS	0	10	10
	999935	30/08/19	X02	0	5	5
	999935	30/08/19	N09	0	20	20

Il reste 15 pièces à répartir par rapport au besoin de 35.

Pour X02 : $5\% \text{ de } 15 = 0,75$. On conserve 0.

Pour N09 : $15\% \text{ de } 15 = 2,25$. On conserve 2.

Pour VMS on attribue $15 - 2$, soit 13.

	POS	ECHEANCE	SITE	Reste à livrer	Reste à recevoir	Transit
Après répartition	999935	30/08/19	VMS	$0+13 = 13$	$10+13 = 23$	10
	999935	30/08/19	X02	0	5	5
	999935	30/08/19	N09	$0+2 = 2$	$20+2 = 22$	20

$23+22+5 = 50$, on a tout notre besoin.

Vous pourrez remarquer que mon algorithme ne prend pas en compte la présence ou non de transit. C'est parce que la fonction existante qui conserve uniquement les transits du POS N-1 (fonction que j'ai créée lors de la refonte de POS multisite pour la FEB 2346) recopie la quantité en transit du POS N-1 dans le POS N. S'il n'y a pas de transit, elles sont à 0 donc la copie se fait quand même. Réutiliser cette fonction me permet donc d'alléger mon algorithme et de créer moins de code.

S'il n'y a pas de réceptions partielles, la répartition automatique s'applique à chaque fois. J'utilise donc uniquement la fonction qui recopie les transits du POS N-1 pour les mêmes raisons qu'énoncées ci-dessus.

A la fin, un calcul de delta est effectué pour connaître la quantité de besoin restante entre le POS N-1 et ce qui a été inséré en base pour le POS N. Ce besoin restant est distribué automatiquement selon les différents pourcentages de répartition et le multiple de conditionnement s'il existe. Le surplus étant toujours remis sur le site par défaut.

4.5.2.4. Historisation

Pour répondre aux SFD, je dois enregistrer en base de données, dans la nouvelle table T_DISTRIBUTION_HISTORY, une ligne pour chaque site de chaque échéance ferme en type de distribution automatique ou manuelle.

⇒ *Cette table sera aussi utilisée pour enregistrer les changements de pourcentage de distribution d'un site et du multiple de conditionnement d'un contrat par l'utilisateur. Ces modifications ne se font que sur le portail APP_Suivi et ne concernent pas le flux POS.*

La table T_DISTRIBUTION_HISTORY reprendra des informations dans plusieurs tables de la base de données afin d'être la plus complète possible.

J'ai donc mis en place l'algorithme suivant :

```
Pour chaque ligne 9 du ESNPOS (sauf celles avec un besoin de 0)
SI ECHEANCE FERME ET type de distribution AUTO OU MANUELLE
    Pour chaque site en commun avec POS N-1
        Insertion dans la table T_DISTRIBUTION_HISTORY
```

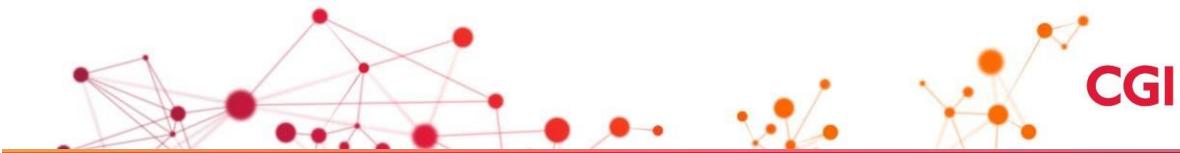
4.5.2.5. Mon test plan

Voici tous les cas de tests que j'ai rédigé et que mon code devra respecter pour remplir les requirements :

The screenshot shows the 'Application Lifecycle Management' interface. The left sidebar has sections for Dashboard, Management, Requirements, Testing (with sub-options like Test Reso..., Test..., and Test...), and Defects. The main area shows a tree view of test cases under 'Testing'. The path selected is 'Release 10.62.x' > '10.62.0' > '#2628' > 'Tests Globaux' > 'REJETS'. Below this, there are numerous detailed test cases, many of which are expanded. At the bottom of the tree view, there is a section for 'Répartition AUTO' and 'Avec réceptions partielles'. A large number of specific test cases are listed under these sections, such as 'Besoins inférieurs au transit pour une échéance - ENG' and 'Site non repris avec du transit - FR'. The bottom of the screen shows a list of methods: 'Méthode calculerRepartition()', 'Méthode calculerSommeDistributions()', 'Méthode compterReceptionsPartielles()', 'Méthode compterRepartitionsAuto()', 'Méthode getDeletedSitesWithDistribution - Aucun site supprimé du ESNPOS', 'Méthode getDeletedSitesWithDistribution - On renseigne un nouveau site', 'Méthode getDeletedSitesWithDistribution - On renseigne un site qui a un pourcentage à null en base', 'Méthode getDeletedSitesWithDistribution - Plusieurs sites ont été supprimés du ESNPOS', 'Méthode getDeletedSitesWithDistribution - Un site a été supprimé du ESNPOS', 'Méthode insertTTDistributionHistory()', 'Méthode selectConditionning()', 'Méthode selectDistribution()', 'Méthode selectTQuantities()', and 'Méthode selectTypeDistribution()'. The top right corner of the interface displays 'Domain: FR_FGDC, Pro'.

Figure 33 - Cas de tests FEB 2628

Ces cas de tests sont ensuite assignés au requirements. Chaque requirement doit donc posséder des tests complétés et validés pour pouvoir être considérés comme couverts.



Exemple de steps à suivre pour tester la méthode selectDistribution() :

Details			
	Step Name	Description	Expected Result
	Step 1	CAS : Le pourcentage récupéré est un entier Appeler la fonction avec les paramètres suivants : Contrat : <<<contrat1>>> Ligne contrat : <<<lcontrat>>> Code baan : <<<baan>>> Site : <<<site1>>>	Résultat espéré : <<<resultat1>>>
	Step 2	CAS : Le pourcentage récupéré est un décimal Appeler la fonction avec les paramètres suivants : Contrat : <<<contrat2>>> Ligne contrat : <<<lcontrat>>> Code baan : <<<baan>>> Site : <<<site2>>>	Résultat espéré : <<<resultat2>>>
	Step 3	CAS : Le pourcentage récupéré vaut 0 Appeler la fonction avec les paramètres suivants : Contrat : <<<contrat3>>> Ligne contrat : <<<lcontrat>>> Code baan : <<<baan>>> Site : <<<site3>>>	Résultat espéré : <<<resultat3>>>
	Step 4	CAS : Le pourcentage récupéré est null en base Appeler la fonction avec les paramètres suivants : Contrat : <<<contrat4>>> Ligne contrat : <<<lcontrat>>> Code baan : <<<baan>>> Site : <<<site4>>>	Résultat espéré : <<<resultat4>>>
	Step 5	CAS : Récupération d'un pourcentage pour un site inexistant pour ce contrat Appeler la fonction avec les paramètres suivants : Contrat : <<<contrat5>>> Ligne contrat : <<<lcontrat>>> Code baan : <<<baan>>> Site : <<<site5>>>	Résultat espéré : <<<resultat5>>>

Figure 34 - Design steps d'un cas de test FEB 2628



Exemple de steps à effectuer pour tester le fonctionnement d'une répartition automatique dans le cas où il existe des réceptions partielles :

Design Steps			
	Step Name	Description	Expected Result
	Step 1	Mettre le fichier ci-joint dans le dossier IN	Il est consommé par le flux POS et mis dans le dossier ARCHIVE
	Step 2	Vérifier la table [REDACTED]	Le nouveau POS <<<new_pos>>> a été créé, et est au statut [REDACTED]. Tandis que l'ancien POS <<<old_POS>>> est au statut [REDACTED].
	Step 3	Vérifier la table [REDACTED]	Le site <<<deleted_site>>> n'apparaît plus. Le site <<<new_site>>> apparaît avec un pourcentage de distribution null. Les sites <<<sites>>> sont également toujours présents. Le site <<<default_site>>> est bien le nouveau site par défaut
	Step 4	Vérifier la table [REDACTED]	Les données correspondent à celles du fichier ENSPOS, pour chaque échéance > 0 en besoin : <<<ligne t_lines_pos>>>
	Step 5	Vérifier la table [REDACTED] pour le pos <<<new_pos>>>	pour l'échéance <<<echeance>>>, et les sites <<<sites>>> on a des lignes t_quantities avec du besoin : <<<ligne t_lines_pos>>> Pour le site <<<new_site>>>, on a un ligne t_quantities à 0. Le site <<<deleted_site>>> n'apparaît plus
	Step 6	Vérifier la table [REDACTED] pour le contrat <<<contrat>>> et l'échéance <<<echeance>>>	Lignes t_lines_pos : <<<lignes_t_lines>>>
	Step 7	Vérifier la table [REDACTED] pour le contrat <<<contrat>>>, ligne <<<lignecontrat>>>, code baan <<<baan>>> et l'échéance <<<echeance>>>	Il existe une entrée par site en commun entre le POS N et le POS N-1 :<<<lignes_t_distrib>>>

Figure 35 - Design steps d'un cas de test FEB 2628 - 2



4.5.2.6. Mon test lab

Le test lab regroupe tous les cas de tests rédigés dans le test plan et qui doivent être joués. Si tous les résultats attendus sont respectés alors le test est validé. Pour que tous les requirements soient respectés, tous les tests doivent être aussi validés.

Execution G...			
Name	Test: Test...	Type	Status
[1]Méthode getDeletedSitesWithDistribution - Aucun site supprimé du ESNPOS	↳ Méthode g...	MANUAL	Passed
[1]Méthode getDeletedSitesWithDistribution - On renseigne un nouveau site	↳ Méthode g...	MANUAL	Passed
[1]Méthode getDeletedSitesWithDistribution - On renseigne un site qui a un pourcentage à null en base	↳ Méthode g...	MANUAL	Passed
[1]Méthode getDeletedSitesWithDistribution - Plusieurs sites ont été supprimés du ESNPOS	↳ Méthode g...	MANUAL	Passed
[1]Méthode getDeletedSitesWithDistribution - Un site a été supprimé du ESNPOS	↳ Méthode g...	MANUAL	Passed
[1]Méthode calculerRepartition()	↳ Méthode c...	MANUAL	Passed
[1]Méthode calculerSommeDistributions()	↳ Méthode c...	MANUAL	Passed
[1]Méthode compterReceptionsPartielles()	↳ Méthode c...	MANUAL	Passed
[1]Méthode compterRepartitionsAuto()	↳ Méthode c...	MANUAL	Passed
[1]Méthode insertTDistributionHistory()	↳ Méthode i...	MANUAL	Passed
[1]Méthode selectConditionning()	↳ Méthode s...	MANUAL	Passed
[1]Méthode selectDistribution()	↳ Méthode s...	MANUAL	Passed
[1]Méthode selectTQuantities()	↳ Méthode s...	MANUAL	Passed
[1]Méthode selectTypeDistribution()	↳ Méthode s...	MANUAL	Passed
[1]Besoin inférieur au transit pour une échéance - ENG	↳ Besoin inf...	MANUAL	Passed
[1]Besoin inférieur au transit pour une échéance - FR	↳ Besoin inf...	MANUAL	Passed
[1]Site non repris avec du transit - ENG	↳ Site non re...	MANUAL	Passed
[1]Site non repris avec du transit - FR	↳ Site non re...	MANUAL	Passed
[1]Site non repris avec un pourcentage de distribution - ENG	↳ Site non re...	MANUAL	Passed
[1]Site non repris avec un pourcentage de distribution - FR	↳ Site non re...	MANUAL	Passed
[1]Besoin pour une échéance n est pas un multiple du multiple de conditionnement - ENG	↳ Besoin po...	MANUAL	Passed
[1]Besoin pour une échéance n est pas un multiple du multiple de conditionnement - FR	↳ Besoin po...	MANUAL	Passed
[1]CAS - réception partielle, transit, suppression de site, ajout de site, site par défaut changé, besoin augmenté	↳ CAS - réc...	MANUAL	Passed
[1]CAS - réception partielle, transit, suppression de site, ajout de site, site par défaut changé, besoin identique	↳ CAS - réc...	MANUAL	Passed
[1]CAS - réception partielle, transit, suppression de site, ajout de site, site par défaut changé, besoin diminué	↳ CAS - réc...	MANUAL	Passed
[1]CAS - réception partielle, sans transit, suppression de site, ajout de site, besoin augmenté	↳ CAS - réc...	MANUAL	Passed

Figure 36 - Test lab FEB 2628

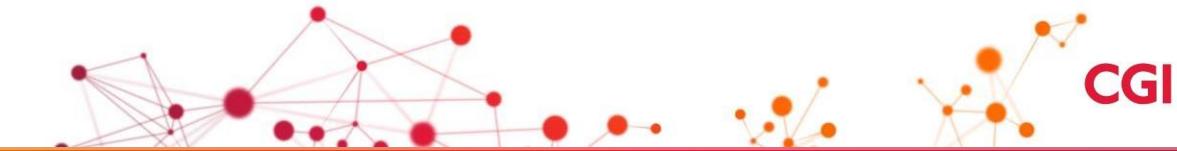
4.6. Script SQL

Voici le script SQL pour la création de la nouvelle table T_DISTRIBUTION_HISTORY et de l'ajout des nouvelles colonnes dans les tables existantes que j'ai rédigé :

```

1. -- Séquence pour DISTRIBUTION_HIST_TECH_ID
2. -- NEXTVAL = 1
3. CREATE SEQUENCE T_DISTRIBUTION_HISTORY_SEQ;
4.
5. CREATE TABLE T_DISTRIBUTION_HISTORY
6. (
7. DISTRIBUTION_HIST_TECH_ID number not null PRIMARY KEY,
8. ACTION_DATE date not null,
9. USER_LAST_NAME varchar2(35 BYTE),
10. USER_FIRST_NAME varchar2(35 BYTE),
11. USER_LOGIN varchar2(35 BYTE) not null,
12. SCH_CONTRACT_NUMBER number not null,
13. SCH_CONTRACT_POSITION number not null,
14. SCH_BAAN_COMPANY_CODE varchar2(3 BYTE) not null,
15. EXPECTED_DELIVERY_DATE date,
16. DELIVERY_SITE_CODE varchar2(35 BYTE) not null,
17. DISTRIBUTION_BEFORE_ACTION number,
18. DISTRIBUTION_AFTER_ACTION number,
19. POURCENTAGE_BEFORE_ACTION number,
20. POURCENTAGE_AFTER_ACTION number,
21. CONDITIONNING_BEFORE_ACTION number,
22. CONDITIONNING_AFTER_ACTION number,
23. INACTIVATED_CONTRACT char(1 BYTE) DEFAULT '0' not null ,
24. DISTRIBUTION_TYPE varchar2(6 BYTE),
25. CONSTRAINT FK_T_DISTRIB_HIST_T_SCH_CNTRCT
26.     FOREIGN KEY(SCH_CONTRACT_NUMBER, SCH_CONTRACT_POSITION, SCH_BAAN_COMPANY_CODE)
27.
28. REFERENCES T_SCH_CONTRACT(SCH_CONTRACT_NUMBER, SCH_CONTRACT_POSITION, SCH_BAAN_C
OMPANY_CODE)
29. );
30. ALTER TABLE T_AVAILABLE_SITES
31. ADD POURCENTAGE_DISTRIBUTION number;
32.
33. ALTER TABLE T_SCH_CONTRACT
34. ADD (
35.     CONDITIONNING number,
36.     IS_DISABLED char(1 BYTE) DEFAULT '0' not null
37. );
38.
39. ALTER TABLE T_LINES_POS
40. ADD DISTRIBUTION_TYPE varchar2(6 BYTE);
41.
42. ALTER TABLE T_EMPLOYEES
43. ADD CONVERGENT char(1 BYTE) DEFAULT '0' not null;
```

Vous pourrez trouver en annexe n°4 toutes les requêtes SQL que j'ai rédigées afin de rendre possible le développement.



4.7. Code JAVA

Vous pourrez trouver en annexe n°5 le code JAVA que j'ai produit et qui est actuellement en attente de passage des tests d'intégration par notre analyste fonctionnelle. J'y ai également mis des explications.

4.8. Tests sur ALM

Les tests que j'ai effectués sont divisés en deux parties, il y a les tests unitaires qui testent le fonctionnement de mes nouvelles fonctions et j'ai également effectué des tests plus globaux du fonctionnement de l'algorithme existant mêlé aux miens. On peut assimiler ça à des tests de non régression ainsi qu'à des tests d'intégration (ils n'empêchent cependant pas que d'autres tests d'intégration sont effectués par notre analyste fonctionnelle avant l'envoi en recette).

4.8.1. Tests unitaires

Mon service ne disposant pas d'un outil adapté pour faire des tests unitaires (tel que JUnit), j'ai créé une classe Java appelée POSUnitTests qui contient l'appel à toutes les fonctions que je veux tester ainsi que les différents jeux de données pour tester les différents cas.

Les résultats attendus et obtenus sont appelés dans un logger, ce qui me permet de vérifier si les fonctions sont opérationnelles.

```
public void tests2628() throws SQLException, ControlException, SocketCommException, Exception {
    logger.debug("-- POS Unit tests BEGIN --");

    logger.debug("-- Récupérer le pourcentage de distribution d'un site dans la table T_AVAIL");
    logger.debug("-- TEST : selectDistribution(int SCH_CONTRACT_NUMBER, int SCH_CONTRACT_POS);

    //Récupération pourcentage entier
    double pourcentage = this.eSupply.selectDistribution(36519, 1, "920", "M91");
    logger.debug("Site passé en paramètres : M91");
    logger.debug("Expected : 70");
    logger.debug("Result : "+pourcentage);
    //Récupération pourcentage décimal
    pourcentage = this.eSupply.selectDistribution(999932, 1, "920", "F75");
    logger.debug("Site passé en paramètres : F75");
    logger.debug("Expected : 20.5");
    logger.debug("Result : "+pourcentage);

    //Récupération pourcentage 0
    pourcentage = this.eSupply.selectDistribution(999932, 1, "920", "F64");
    logger.debug("Site passé en paramètres : F64");
    logger.debug("Expected : 0");
    logger.debug("Result : "+pourcentage);

    //Récupération pourcentage null
    pourcentage = this.eSupply.selectDistribution(999932, 1, "920", "Livraison multisite");
    logger.debug("Site passé en paramètres : Livraison multisite");
    logger.debug("Expected : 0");
    logger.debug("Result : "+pourcentage);

    //Récupération pourcentage d'un site inexistant pour ce POS
    pourcentage = this.eSupply.selectDistribution(999932, 1, "920", "Z05");
    logger.debug("Site passé en paramètres : Z05");
    logger.debug("Expected : 0");
}
```

Figure 37- Tests unitaires d'une fonction

Operation selectDistribution on Control appli:eSupply

Submitted at Thu Oct 24 14:07:55 GMT 2019
Method: controls.appli.ESupply.selectDistribution

Arguments:
SCH_CONTRACT_NUMBER : 36519
SCH_CONTRACT_POSITION : 1
SCH_BAAN_COMPANY_CODE : 920
DELIVERY_SITE_CODE : M91

CallStack:
appli:eSupply.selectDistribution()
appli:control()
subscription()

Returned from selectDistribution on appli:eSupply

Submitted at Thu Oct 24 14:07:55 GMT 2019
Return value: 70.0

Operation debug on Control appli:logger

Submitted at Thu Oct 24 14:07:55 GMT 2019
Method: controls.base.POSLoggerCtrl.debug

Arguments:
message : Site passé en paramètres : M91
CallStack:
appli:logger.debug()
appli:control()
subscription()

Returned from debug on appli:logger

Submitted at Thu Oct 24 14:07:55 GMT 2019

Operation debug on Control appli:logger

Submitted at Thu Oct 24 14:07:55 GMT 2019
Method: controls.base.POSLoggerCtrl.debug

Arguments:
message : Expected : 70
CallStack:
appli:logger.debug()
appli:control()
subscription()

Returned from debug on appli:logger

Submitted at Thu Oct 24 14:07:55 GMT 2019

Operation debug on Control appli:logger

Submitted at Thu Oct 24 14:07:55 GMT 2019
Method: controls.base.POSLoggerCtrl.debug

Arguments:
message : Result : 70.0
CallStack:
appli:logger.debug()

Figure 38- Exemple résultat tests unitaires

4.8.2. Tests globaux

Les tests globaux de cette évolution ont quant à eux été effectués en mettant en scène une déclaration de programme dans l'environnement de développement.

Voici la liste des fichiers ESNPOS que j'ai créé en tant que jeux de données pour vérifier tous mes cas de tests :

- ESNPOS-920-2628 - 007757 - transits - new site - delete site - changement site par defaut - besoin augmenté - 36519.edi
- ESNPOS-920-2628 - 007757 - transits - new site - delete site - changement site par defaut - besoin diminué - 36519.edi
- ESNPOS-920-2628 - 007757 - transits - new site - delete site - changement site par defaut - besoin diminué - 36519.1.edi
- ESNPOS-920-2628 - 007757 - transits - new site - delete site - changement site par defaut - besoin identique - 36519.edi
- ESNPOS-920-2628 - 007757 - manuel et null - transits - new site - delete site - chgmt site par defaut - besoin identique - 36519.edi
- ESNPOS-920-2628 - 007757 - manuel et null - avec transits - new site - delete site - besoin augmenté - 36519.edi
- ESNPOS-920-2628 - 007757 - manuel et null - avec transits - new site - delete site - changement site par defaut - besoin augmenté - 36519.edi
- ESNPOS-920-2628 - 007757 - manuel et null - sans transits - new site - delete site - besoin identique - 36519.edi
- ESNPOS-920-2628 - 007757 - manuel et null - sans transits - new site - delete site - chgmt site par défaut - besoin identique - 36519.edi
- ESNPOS-920-2628 - 007757 - Reception partielle - sans transits - new site - delete site - changement site par defaut - besoin augmenté - 36519.edi
- ESNPOS-920-2628 - 007757 - Reception partielle - sans transits - new site - delete site - changement site par defaut - besoin diminué - 36519.edi
- ESNPOS-920-2628 - 007757 - Reception partielle - transits - new site - delete site - changement site par defaut - Besoin + - 36519.edi
- ESNPOS-920-2628 - 007757 - Reception partielle - transits - new site - delete site - changement site par defaut - besoin diminué - 36519.edi
- ESNPOS-920-2628 - 007757 - Reception partielle - transits - new site - delete site - changement site par defaut - besoin diminué - volée - 36519.edi
- ESNPOS-920-2628 - 007757 - Reception partielle - transits - new site - delete site - changement site par defaut - besoin diminué - volée - 36519.1.edi
- ESNPOS-920-2628 - 007757 - Reception partielle - transits - new site - delete site - changement site par defaut - besoin diminué - volée - 36519.2.edi
- ESNPOS-920-2628 - 007757 - Reception partielle - transits - new site - delete site - changement site par defaut - besoin identique - 36519.edi
- ESNPOS-920-2628 - 007757 - REJET BESOIN INF TRANSIT PLUSIEURS SITES - CAS PASSANT - 36519.edi
- ESNPOS-920-2628 - 007757 - REJET DISTRIBUTIONS KO - 36519.edi
- ESNPOS-920-2628 - 007757 - REJET DISTRIBUTIONS KO - PLUSIEURS SITES - 36519.edi
- ESNPOS-920-2628 - 007757 - REJET MULTIPLE KO - 36519.edi
- ESNPOS-920-2628 - 007757 - REJET SITE NON REPRIS AVEC TRANSIT - 36519.edi
- ESNPOS-920-2628 - 007757 - sans transits - new site - delete site - changement site par defaut - besoin augmenté - 36519.edi
- ESNPOS-920-2628 - 007757 - sans transits - new site - delete site - changement site par defaut - besoin diminué - 36519.edi

Figure 39- Jeux de données tests globaux

Pour savoir si un cas de test est validé, je me réfère aux steps de mon test plan. Les vérifications se font directement en base de données ou sur l'application APP_SUIVI (toujours dans l'environnement de développement).

Une fois que tous mes cas de tests ont été passés et validés, les requirements se retrouvent couverts par le code produit :

La répartition automatique ne doit se faire que si les échéances n'ont pas été modifiées manuellement	1024	Passed	sirika.hilaire
La répartition automatique ne se fait que sur les échéances fermes	1025	Passed	sirika.hilaire
Le nombre de pièces après répartition doit être un entier	1026	Passed	sirika.hilaire
Si le besoin renseigné n'est pas un multiple du multiple de conditionnement, le message est rejeté	1027	Passed	sirika.hilaire
Si l'application des pourcentage de répartition engendre des valeurs décimales elles doivent être arrondies à la valeur inférieure et le surplus devra être mis sur le si...	1028	Passed	sirika.hilaire
Dans le cas d'un pourcentage de répartition et d'un multiple de conditionnement, on ajoute les pièces restantes sur le site par défaut	1029	Passed	sirika.hilaire
Si un site est supprimé et qu'un pourcentage de répartition lui était affecté, le message est rejeté	1030	Passed	sirika.hilaire
Si un nouveau site est ajouté, une répartition de 0 lui est affectée	1031	Passed	sirika.hilaire
Pour chaque échéance, ferme et en répartition automatique ou manuelle, traitée dans le ESNPOS, une ligne par site doit être insérée dans T_DISTRIBUTION_HIS...	1048	Passed	sirika.hilaire

Figure 40- Requirements couverts et validés

5. Complément de compétences

Le projet que je vous ai présenté ne possédant pas d'IHM*, j'ai jugé pertinent de vous parler rapidement d'un développement qui est en cours, pour la même release 10.62. De plus, je vais également vous montrer que le code dans lequel je travaille est organisé en plusieurs couches et que la couche web respecte une architecture MVC*.

On m'a confié le développement qui consiste à modifier l'écran de répartition des sites : des données sont à ajouter et des traitements également.

Je dois rajouter le multiple de conditionnement du contrat dans la cartouche (1), rajouter une colonne dans le tableau des échéances pour afficher le type de répartition (2) et ajouter une vérification des quantités RAL (3) indiquées pour les sites de livraison afin de vérifier que le nombre est un multiple du multiple de conditionnement. Si ce n'est pas le cas, une pop-up informative apparaît mais n'est pas bloquante pour l'utilisateur.

Figure 41- IHM FEB 2628 - Ecran de répartition des sites

Le code est découpé en différentes couches :

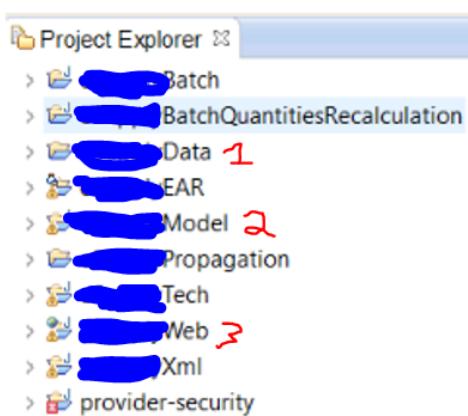


Figure 42- Architecture n-tiers

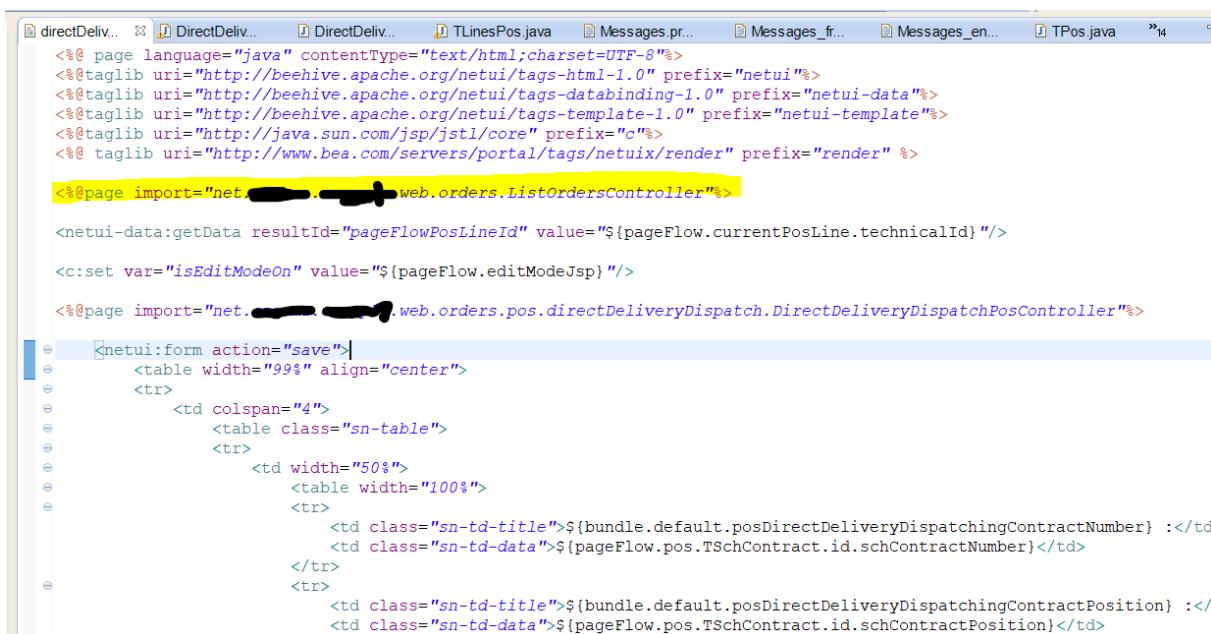
- (1) La couche d'accès aux données en base (DAO).
- (2) Une couche modèle qui contient les services qui contiennent des fonctions faisant appel au DAO pour représenter les différentes règles métier.
- (3) Une couche WEB pour l'affichage des données

Pour ce qui est de la couche DAO ainsi que les objets présents dans la couche modèle, ils ont été mappés* grâce au Framework Hibernate*. Ainsi, chaque objet étend la classe HibernateAbstractPersistent qui permet de faire correspondre les attributs* de la classe à des

colonnes d'une table de la base de données et les DAO étendent la classe HibernateAbstractDao qui permet notamment de récupérer certaines méthode du Framework comme findById() ; pour récupérer un objet en base par la valeur de son id, mais également des fonctionnalités de construction de requêtes SQL comme .add(Restriction) pour ajouter des contraintes, criteria.add pour ajouter des filtres etc...

La couche WEB respecte quant à elle une architecture MVC (Modèle, vue, contrôleur) :

- Les fichiers .jsp* forment la couche des **vues**. C'est dedans que se trouve le code html et javascript.
- Chaque jsp est associé à un **controller***. Vous pouvez voir en figure 43 que la **directDeliveryDispatch.jsp** (jsp de l'IHM que j'ai en charge) est liée au controller **ListOrdersController**.



```

<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<%@taglib uri="http://beehive.apache.org/netui/tags-html-1.0" prefix="netui"%>
<%@taglib uri="http://beehive.apache.org/netui/tags-databinding-1.0" prefix="netui-data"%>
<%@taglib uri="http://beehive.apache.org/netui/tags-template-1.0" prefix="netui-template"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://www.bea.com/servers/portal/tags/netuix/render" prefix="render" %>

<%@page import="net.██████████.web.orders.ListOrdersController"%>
<%@page import="net.██████████.web.orders.pos.directDeliveryDispatch.DirectDeliveryDispatchPosController"%>

<netui-data: getData resultId="pageFlowPosLineId" value="${pageFlow.currentPosLine.technicalId}"/>
<c:set var="isEditModeOn" value="${pageFlow.editModeJsp}"/>
<%@page import="net.██████████.web.orders.pos.directDeliveryDispatch.DirectDeliveryDispatchPosController"%>

<netui:form action="save">
    <table width="99%" align="center">
        <tr>
            <td colspan="4">
                <table class="sn-table">
                    <tr>
                        <td width="50%">
                            <table width="100%">
                                <tr>
                                    <td class="sn-td-title">${bundle.default.posDirectDeliveryDispatchingContractNumber} :</td>
                                    <td class="sn-td-data">${pageFlow.pos.TSchContract.id.schContractNumber}</td>
                                </tr>
                                <tr>
                                    <td class="sn-td-title">${bundle.default.posDirectDeliveryDispatchingContractPosition} :</td>
                                    <td class="sn-td-data">${pageFlow.pos.TSchContract.id.schContractPosition}</td>
                                </tr>
                            </table>
                        </td>
                    </tr>
                </table>
            </td>
        </tr>
    </table>
</netui:form>

```

Figure 43 - Exemple d'import de controller dans une jsp (couche Vue)

Le controller contient les méthodes qui sont appelées dans la jsp. Il fait le lien entre la couche vue et la couche modèle.

- Les méthodes contenues dans les controllers font appel à des méthodes présentes dans la couche modèle :

```

@ControlImplementation
public class LinePosBizManagerImpl implements LinePosBizManager, Serializable {
    // ----- Constants

    /** Auto generated serial uid */
    private static final long serialVersionUID = -7819735073472140318L;

    private List<OrderItemAnalysis> m_oOrderItemAnalysisList;

    /** Logger */
    private static final ILogging __oLogger = ILoggingFactory.getILogging(LinePosBizManagerImpl.class);

    // ----- Implemented methods defined by the LinePosBizManager interface

    /**
     * return the POS with the pos number equal to i_oPosNumber
     * @param i_oPosId id of the chosen Pos
     * @return TPos
     * @throws DatabaseException
     */
    public TPos getTPosAndLinesById(TPosId i_oPosId) throws DatabaseException{
        return TPos.getTPosAndLinesById(i_oPosId);
    }
}

```

Figure 44 - Exemple d'un controller et de l'appel à la couche Modèle

- Exemple d'une classe de la couche modèle (1) on peut voir la méthode appelée par le controller qui fait appel à la couche Data (2).

```

1
public class TPos extends net.s necma.esupply.model.datamodel.TPos {

    // ----- Private attributes

    /** auto generated uid */
    private static final long serialVersionUID = -8005519544247557079L;
    /** value 'true' for the Simulation flag */
    private static final BigDecimal SIMULATION_FLAG_TRUE = new BigDecimal(1);

    /** Logger */
    private static final ILogging __oLogger = ILoggingFactory.getILogging(TPos.class);

    /** lines modified by the supplier */
    private List<TLinesPos> m_oNewLinesPos = null;

    // ----- Static methods

    /**
     * @see TPosDAO#findById(TPoId)
     */
    public static TPos findById(TPosId i_oID) throws DatabaseException { }

    /**
     * Return the TPos found in the database depending with the POS identifier
     */
    public static TPos findForDetailsPOS (TPosId i_oPosId) throws DatabaseException { }

    /**
     * Return the TPos found in the database depending of the contract Id
     */
    public static TPos findPosByContractId(TSchContractId i_oContractId) throws DatabaseException { }

    /**
     * return the POS with the pos number equal to i_oPosNumber
     */
    public static TPos getTPosAndLinesById(TPosId i_oPosId) throws DatabaseException{
        return TPosDAO.findWithTLinesById(i_oPosId);
    }
2

```

Figure 45 - Exemple couche Modèle

6. Conclusions

6.1. Bilan du projet

Malgré le nombre d'applications à maintenir et à faire évoluer, notre équipe AM parvient à satisfaire le client par la connaissance du métier, des applications, l'anticipation des problèmes, la qualité des livrables et la réactivité.

L'un des principaux points forts de notre périmètre est la relation de confiance et de partenariat que nous avons avec le client ainsi que les métiers. Au cours de mon année d'alternance, j'ai pu voir ce lien se fortifier grâce au professionnalisme de tous les membres de l'équipe.

Vous pouvez voir sur la figure suivante (figure 46) la satisfaction du client pour le mois de septembre 2019. La moyenne a été de 9,2/10.

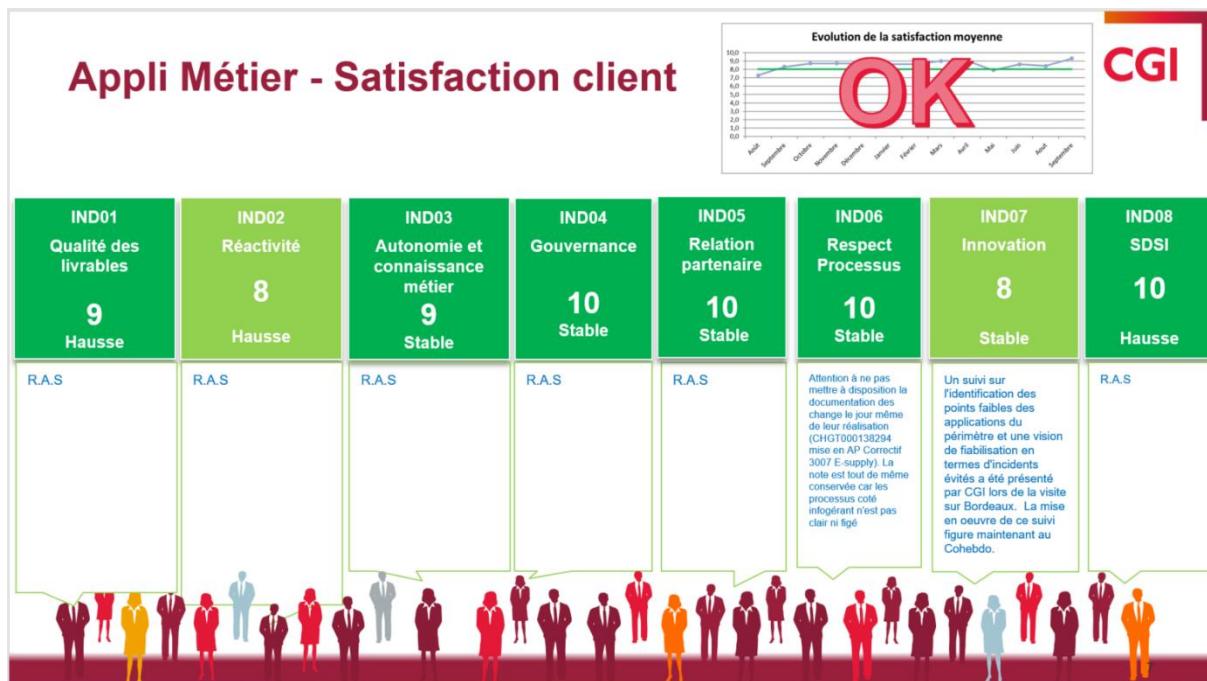


Figure 46- Satisfaction client team AM septembre 2019

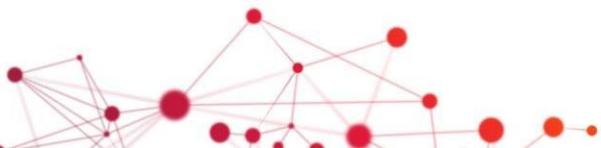
Nous avons également reçu un e-mail interne nous annonçant que la moyenne pour le mois d'octobre 2019 est la plus haute jamais eue sur notre périmètre : 9,7/10.

6.2. Bilan personnel et avenir professionnel

Personnellement et professionnellement, cette alternance m'a beaucoup apporté.

Je sens que j'ai énormément progressé depuis mon arrivée en décembre 2018, je me sens autonome sur plusieurs tâches et je sens que mes collègues ont également développé une confiance en moi, petit à petit, mission après mission.

Je sens également que j'ai trouvé ma vocation car je suis contente d'aller au travail chaque jour et quand on me confie de nouvelles tâches je suis très heureuse d'apprendre de nouvelles choses et surtout, je suis contente de monter en compétence pour aider mes collègues. Je donne toujours



mon maximum et je fais toujours au mieux pour ne pas les décevoir parce qu'ils ont toujours été là pour moi, pour m'encadrer, m'expliquer, me former. De plus, on s'entend parfaitement et on arrive à travailler ensemble et s'entraider dans des conditions agréables avec le même but commun : réussir nos missions.

Les différentes missions que l'on m'a confiées m'ont permis de valider beaucoup de compétences. Vous pourrez trouver la grille des compétences complétée en annexe 6. En effet, pratiqué beaucoup de SQL et de Java.

Pour mon avenir professionnel, j'aimerais poursuivre mes études en alternance en bac+5 au sein de cette même équipe en continuant de monter en compétence sur le langage Java. A plus long terme, j'aimerais également, si j'en ai l'opportunité, pratiquer du Node JS et du Angular.

Vous pourrez trouver en annexe n°7 ma fiche d'évaluation remplie par mon chef de projet.

6.3. Compréhension du métier visé

Pour moi, le métier de Concepteur Développeur d'Applications Numériques est un métier très polyvalent.

Il faut être capable d'analyser, comprendre et retranscrire sous forme de spécifications fonctionnelles le besoin du client. Il faut également bien connaître la ou les applications qu'on a en charge et comprendre leur fonctionnement car il faut, dans la majorité des cas, aider le client à comprendre son besoin et à trouver la solution la plus adaptée.

Il faut également être capable d'évaluer et de planifier le temps qu'il faudra pour mettre en place une solution, une évolution.

Il faut être capable de retranscrire les spécifications fonctionnelles en spécifications techniques. C'est à dire analyser le code existant et concevoir les algorithmes à mettre en place, identifier les classes, objets, méthodes qui devront être créées, réutilisées, adaptées et les données qui seront à récupérer.

Parallèlement aux tâches de spécifications fonctionnelles et techniques, il faut être capable d'identifier et anticiper les potentielles difficultés ou les problèmes qu'on pourrait rencontrer afin de mettre en place des solutions le plus tôt possible.

Dans mon service, nous rédigeons les tests unitaires avant de commencer à écrire le code. Cette phase de tests consiste à vérifier que chaque méthode créée fonctionne bien comme on l'attend. Cela s'accompagne de différents jeux de données afin de tester les différents cas de figure.

Lors de la partie développement pur, il faut mettre en place les algorithmes et méthodes imaginées lors des spécifications techniques en conservant la qualité et le fonctionnement du code existant. Il faut réutiliser le code existant au maximum et rendre réutilisable le nouveau également.

Une fois le code et les tests unitaires réalisés, il faut procéder à des tests d'intégration. Contrairement aux TU, les TI permettent de tester l'intégralité du code selon tous les cas de tests imaginés afin de s'assurer que le code existant remplit toujours ses fonctions (on parlera



de non régression et donc de tests de non régression du code) et que le code ajouté respecte bien les nouvelles spécifications fonctionnelles. Je tiens à préciser l'importance des TU et des TI car en terme de coût, au plus vite on se rend compte que quelque chose ne fonctionne pas correctement, au moins cela n'a d'impact financier sur le projet. En effet, si on se rend compte lors des TU qu'une méthode ne fonctionne pas tout à fait correctement, cela prend moins de temps à corriger que si l'on en fait le constat lorsque le code est en production car on devra repasser par les étapes analyse, recherche de solution, TU, TI, recette pour un problème qu'on aurait pu identifier plus tôt, sans parler de la perte de crédibilité et de confiance aux yeux du client.

Lorsque le code a été validé par les TI, il faut créer le package qui sera installé lors de la mise en environnement de recette. Lors de mon année d'alternance, les recettes étaient effectuées par le métier côté client. Notre analyste fonctionnelle s'occupait de faire de l'assistance recette, c'est-à-dire qu'elle répondait aux différentes questions du métier. S'il valide les fonctionnalités du nouveau code, on peut procéder à la mise en production. Il peut également s'en suivre de l'assistance aux utilisateurs et des formations pour leurs apprendre à utiliser les nouvelles fonctionnalités. Cependant, mon service n'effectue pas ces formations.

Toutes ces choses font, à mes yeux, que ce métier un métier très complet et intéressant. On n'a pas l'impression de faire tous les jours la même chose et la routine ne s'installe pas. Il faut savoir faire preuve de réactivité et d'ingéniosité pour trouver des solutions rapidement et limiter les impacts négatifs pour le client.

7. Glossaire

Algorithm : C'est une suite d'instructions, qui une fois exécutée, donne un résultat ou une solution à un problème indépendamment des données.

Approvisionneur : Personne qui a la responsabilité de commander les matières premières et produits chimiques, nécessaires à la fabrication du produit, et veiller à ce qu'ils arrivent en temps et quantités. Il détermine les quantités nécessaires, ainsi que les dates de livraisons nécessaires.

Attributs : Ce sont les variables d'une classe.

Batch : Batch est un langage de script, il permet d'exécuter une suite d'instructions ayant un but précis et pouvant servir à l'automatisation de tâches.

Bug : Défaut de conception d'un programme informatique à l'origine d'un dysfonctionnement.

Business Unit (Unité d'affaire) : Unité organisationnelle au sein d'une entreprise définie autour d'un domaine d'activité qui est dirigée de façon autonome avec des objectifs et des ressources propres.

CAC40 : C'est un panier composé de 40 valeurs de sociétés françaises. Ces sociétés sont choisies parmi les 100 sociétés françaises dont les volumes d'échanges de titres sont les plus importants. Chaque société a un poids déterminé par rapport à sa capitalisation.

Chiffrage : Document estimatif des coûts pour la réalisation d'une ou plusieurs tâches.

Controller : Contient la logique concernant les actions effectuées par l'utilisateur.

Delta : Différence entre deux grandeurs.

EDI (fichier) : C'est un fichier de données formaté en utilisant un des nombreux standards d'Electronic Data Interchange (EDI). Il contient des données structurées stockées dans un format de texte brut. Il est conçu pour transférer des données entre plusieurs organisations.

Exception : Une exception est une erreur se produisant dans un programme qui conduit le plus souvent à l'arrêt de celui-ci.

Flaguer : Au sein de mon service, flaguer signifie attribuer une valeur booléenne à une entrée en base de données.

Framework : désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel. Un Framework peut à ce titre être constitué de plusieurs bibliothèques. Il peut conduire le développeur à respecter certains patrons de conception.

Gestion de version : Conservation de la chronologie de toutes les modifications qui ont été apportées sur un fichier.

Hibernate : Framework open source gérant la persistance des objets en base de données relationnelle.

IDE : Environnement de Développement Intégré : ensemble d'outils qui permet d'augmenter la productivité des programmeurs qui développent des logiciels¹. Il comporte un éditeur de texte destiné à la programmation, des fonctions qui permettent, par pression sur un bouton, de démarrer le compilateur ou l'éditeur de liens ainsi qu'un débogueur en ligne, qui permet d'exécuter ligne par ligne le programme en cours de construction.

IHM : Interface Homme Machine : interface utilisateur permettant de connecter une personne à une machine, à un système ou à un appareil.

Impartition : Stratégie d'une entreprise qui se procure à l'extérieur des biens matériels ou des services, au lieu de prendre elle-même en charge leur production ou leur fourniture.

Job VTOM : VTOM permet d'ordonner l'automatisation de traitements d'une tâche informatique. C'est une solution utilisée dans mon service. Un job est donc un traitement automatisé.

Jsp : Les JSP (Java Server Pages) sont une technologie Java qui permet la génération de pages web dynamiques. La technologie JSP permet de séparer la présentation sous forme de code HTML et les traitements écrits en Java. Les JSP définissent une syntaxe particulière permettant d'appeler des variables et d'insérer le résultat de son traitement dans la page HTML dynamiquement.

Log : Historisation des évènements liés à un processus.

Mapper : procédé permettant de définir au niveau d'un langage de programmation la correspondance entre deux modèles de données.

MEP : Mise en production du code réalisé.

Métier : Dans mon service, le garant métier est le responsable d'une application côté client. Il a toutes les connaissances fonctionnelles de cette application. C'est avec lui que sont rédigées les spécifications fonctionnelles et c'est également lui qui effectuent les recettes et donne son feu vert pour la mise en production.

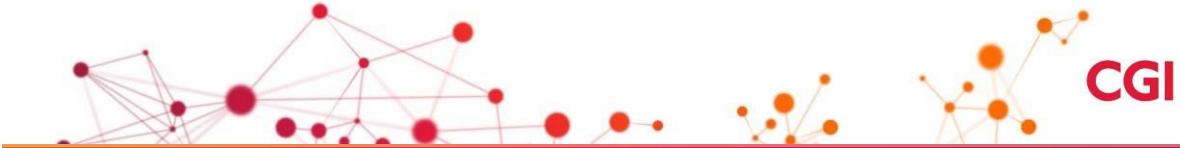
Modulo : Opération qui calcule le reste d'une division.

Monosite (contrat) : Un contrat monosite est un contrat qui ne possède qu'un seul site de livraison.

Morning : Réunions effectuées tous les matins en compagnie du chef de projet et de tous les membres de l'équipe afin d'apporter une vision sur ce qui a été fait, sera fait et est à faire. On parle également des problèmes rencontrés et des solutions apportées.

Multiple de conditionnement : Dans la release 10.62, le multiple de conditionnement est le nombre indiqué par les approvisionneurs pour exprimer la quantité de pièces expédiées par lot.

Multisites (contrat) : Un contrat est multisites s'il possède plusieurs sites de livraison.



MVC (pattern) : signifie Modèle - Vue – Contrôleur. C'est une architecture qui prévoit de séparer le code en plusieurs parties : Les données (Modèle), l'interface graphique (Vue) et la logique (Contrôleur).

Reflaguer : Dans mon service, reflaguer signifie réinitialiser un booléen sur une entrée en base de données.

Release : représente la nouvelle version sur laquelle nous travaillons et qui sera prochainement en production.

Serial : Numéro de série.

SiàSi : Système permettant aux fournisseurs de déclarer des expéditions autrement que par le portail. Ils peuvent déposer directement des fichiers XML contenant les informations à envoyer.

Split (de serials) : Séparation des numéros de série sur plusieurs pré-réceptions au lieu d'une seule.

8. Annexes

8.1. Annexe 1 - Présentation des tickets Jira

Voici la présentation d'un ticket d'incident type Jira :

 TMAM / MANSNETMAM-28932
INCT001021956 : [Medium] Ecart de transit [REDACTED] (priorité Haute) (Incident AM)

[Modifier](#) [Commentaire](#) [Attribuer](#) [Suite ▾](#) [Complete](#) [In Progress](#) [In Review](#) [Workflow ▾](#)

Informations

Type:	! Incident	Etat:	RÉSOLUE (Afficher le workflow)
Priorité:	!! High	Résolution:	Terminé
Affecte la/les version(s):	Aucune	Version(s) corrigée(s):	Aucune
Composants:	[REDACTED]		
Etiquettes:	Aucune		

-----Message d'origine-----
De : [REDACTED] <[REDACTED]>
Envoyé : jeudi 10 octobre 2019 13:08
À : #F_[REDACTED] Hotline Informatique [REDACTED]
Cc : TMAM [REDACTED] - @ [REDACTED] <[REDACTED]>
[REDACTED]

Objet : Ecart de transit [REDACTED] (priorité Haute)

Bonjour,

Merci d'ouvrir un incident à la TMA [REDACTED] pour correction t_quantity

Un message de rejet indique un transit pour le 4 octobre pour le programme 869056 alors que tout est réceptionné dans [REDACTED]

A transit quantity already exists for the ITEM_REFERENCE : 364-600-011-0 :

- POS_DELIVERY_DATE : Fri Oct 04 00:00:00 CEST 2019. DELIVERY_SITE_CODE : VMC This line isn't in the supplied file while it was in POS : 869056

Cordialement,

[REDACTED]
Garant Systèmes d'information [REDACTED] | Direction Industrielle [REDACTED]

Tempo

	Vue: My Data	+ Journal de travail	...
<input checked="" type="checkbox"/> Vérif avant clôture	15m	14/oct/19	...
<input checked="" type="checkbox"/> Correction des données selon les infos de [REDACTED] tout est cohérent en base à présent	2h 15m	11/oct/19	...
<input checked="" type="checkbox"/> Attente retour [REDACTED] sur les restes à expédier.	30m	11/oct/19	...

< < > >> 1 of 1 (3 items)



Voici la présentation d'un ticket Jira pour du développement :

TMAM / MANSNETMAM-27061 #2628 - Livraison multi-site BaaN / MANSNETMAM-27722

Dev & TU #2628 - Rejet POS et contrôle répartition auto ou manuelle

[Modifier](#) [Commentaire](#) [Attribuer](#) [Suite ▾](#) [Close](#) [Resolve](#) [In Review](#) [Workflow ▾](#)

Informations

Type:	Réaliser et tester unitairement	Etat:	EN COURS (Afficher le workflow)
Priorité:	Medium	Résolution:	Non résolu
Affecte la/les version(s):	Release eSupply 10.62.0	Version(s) corrigée(s):	Release [REDACTED] 10.62.0
Composants:	[REDACTED]		
Etiquettes:	Aucune		

Pour cette partie du développement, une charge de travail de 23,3h a été estimée et validée par le client.

Plusieurs personnes peuvent être imputé sur un ticket. Par exemple pour celui-ci je me suis chargée de l'analyse, de l'algorithme, du développement et des tests unitaires. J'ai consacré 24,35h.

Notre analyste fonctionnelle et notre référent technique ont également participé à hauteur de 1h et 2h chacun pour la présentation de l'évolution à faire, du support sur mes retours etc...

Le temps restant est à 0h car le travail est terminé cependant il est revu après chaque jour de développement pour estimer le reste à faire. C'est très important pour le chef de projet qui organise les plannings de connaître nos charges de travail.

Personnes

Responsable:	HILAIRE, Sirika
Rapporteur:	Mesureur, Jennifer
Request participants:	Aucun
Organizations:	Aucun
Votes:	0 Voter pour ce ticket
Gérer les observateurs:	0 Démarrer l'observation de ce ticket

Requête de service d'assistance

Type de requête:	Aucune correspondance
Canal:	Jira

Dates

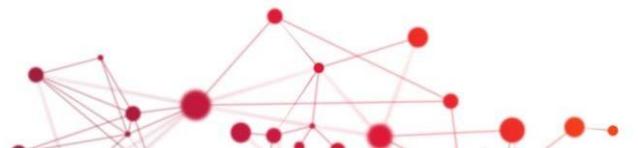
Création:	26/juil./19 11:07 AM
Mise à jour:	29/oct./19 2:32 PM

Suivi temporel

Estimé:	<div style="width: 23.3%;"></div> 23,3h
Restant:	<div style="width: 0%;"></div> 0h
Consigné:	<div style="width: 27h 35m;"></div> 27h 35m

Collaborateurs

BARBE, Benjamin	<div style="width: 2%;"></div> 1h
HILAIRE, Sirika	<div style="width: 24h 35m%;"></div> 24h 35m / 0h
Mesureur, Jennifer	<div style="width: 1%;"></div> 1h



8.2. Annexe 2 - Rédaction de la procédure pour effectuer un split de serial manuellement

Split manuel de serial

ETAPES 1 :

Calculer le nombre de n°E-Supply à créer.

→ 1 n°E-Supply = 350 expéditions.

NB : On ne crée pas un n°E-Supply à proprement parlé, on va rechercher, dans [REDACTED] et pour les mêmes [REDACTED] et DELIVERY_SITE_CODE, des n°E-Supply qui ont été annulé (toutes les valeurs à 0 ou null) et on va les modifier pour que ça coïncide avec le split à réaliser.

Pour ce faire :

Modifier la [REDACTED] à 350 sur le n°E-Supply actif.

En fonction du nombre de n° à recréer :

Bien adapter [REDACTED] à chaque fois, déduire le [REDACTED] qui le précède et recopier toutes les données du premier n°E-Supply.

ETAPES 2 :

Changer le n°E-Supply pour les numéros de série.

Dans [REDACTED] diviser les numéros de série par blocs 350 et un reste. Les 350 premiers garderont le n°E-Supply d'origine mais les suivants vont s'adapter à nos nouveaux numéros. Ainsi, on va DELETE tout ce qui suit les 350 premiers et les recréer avec le bon numéro.

```
SELECT * FROM [REDACTED]
WHERE [REDACTED] = esupplyNumber
AND [REDACTED] IN ('serial1', 'serial2' ...);

DELETE FROM [REDACTED]
WHERE [REDACTED] = esupplyNumber
AND [REDACTED] IN ('serial1', 'serial2' ...);

/* commit toujours appréciable à ce moment du script pour se prémunir d'éventuels bugs */
COMMIT;

insert into [REDACTED] ([REDACTED])
values (esupplyNumber,'serial1','C',1,'');
insert into [REDACTED] ([REDACTED])
values (esupplyNumber,'serial2','C',1,'');
...

```

NB : Pour me faciliter la tâche, quand je copie tous les sériels pour la commande de SELECT et de DELETE, j'utilise notepad++ et les regEXP pour mettre les " et la , autour du numéro de série :

- Coller tous les numéros dans notepad++, ils seront les un en dessous des autres.
- Faire ctrl + F et onglet "remplacer"
- Selon le numéro de série, en général "HA#####", "HB#####", je coche 'expression régulière' et dans recherche je mets (HA[0-9]{6}) et dans remplacer par je mets "\1".
- Remplacer tout

→ [0-9] signifie numéros de 0 à 9 et le {6} signifie qu'il y en a 6. Ensuite les parenthèses servent à réutiliser ce qu'il y a à l'intérieur, à savoir le numéro de série complet. Qui sera traduit par \1 visant le premier bloc entre 0 .

NB2 : Demander à un sachant le fichier excel "Renseignement_Serials_Non_Remontés" pour générer les insert et se faciliter la vie !

ETAPES 3 :

Maintenant il faut reset les flags (sur tous les numéros) pour les [REDACTED] et [REDACTED] afin de faire remonter les informations à BAAN et E-Reception.

```
SELECT * FROM [REDACTED]
WHERE [REDACTED] IN([numeros]);

UPDATE [REDACTED]
SET [REDACTED] = 0,
    [REDACTED] = null,
    [REDACTED] = 0,
    [REDACTED] = null
WHERE [REDACTED] IN([numeros]);
```

ETAPES 4 :

Enfin, pour éviter de futures erreurs d'impression de BL, vérifier que les n°E_Supply sont présents dans la table [REDACTED]

Si aucun n° n'y est présent :

Faire une recherche dans cette table selon le [REDACTED] et récupérer toutes les infos de l'entrée la plus récente ([REDACTED] le plus grand) et les réutiliser pour la création des autres n°.

```
INSERT INTO [REDACTED] ([REDACTED])
VALUES([REDACTED])
```

Si le n°E-Supply qu'on a split y est présent, récupérer toutes ses infos pour l'insert des autres.

8.3. Annexe 3 - Réécriture de la procédure pour comprendre les flags APP_DOC

EDOC : [REDACTED] & [REDACTED]

En base E-DOC, lorsqu'on met le flag à N, lors du passage de la chaîne E-DOC, trois cas sont possibles :

- [REDACTED] nous envoie le nouveau document => Passage du flag à Y
- [REDACTED] ne trouve pas le document et nous renvoie un dossier avec un error.log => Passage du flag à W
- [REDACTED] ne répond pas => Le flag reste à N.

Ainsi, si au bout de 24H on a toujours le flag à N, ça signifie qu'il y a un blocage côté [REDACTED]. Dans ce cas-là, on transfère l'incident sur [REDACTED] au group [REDACTED].

Avant de faire ça, aller via WinSCP sur le serveur EDOC OP dans [REDACTED] et télécharger [REDACTED] pour vérifier si on a un dossier qui porte notre [REDACTED] ou [REDACTED].

S'il n'y en a pas, télécharger dans [REDACTED] le [REDACTED] qui correspond aux fichiers envoyés par [REDACTED] et vérifier si on trouve notre [REDACTED] dedans.

Si pas, on transfère à [REDACTED].

Ce flag est changé lors du passage de la chaîne EDOC. Il passe à Y si un document possède une évolution/mise à jour.

C'est [REDACTED] qui le renseigne à EDOC et EDOC change le flag à Y. Dans la foulée, [REDACTED] nous envoie le document et le flag repasse à N.

Donc dans la pratique, on ne devrait jamais voir le flag à Y en base.

8.4. Annexe 4 - Requêtes SQL

La couche d'accès aux données pour le flux POS se fait par l'intermédiaire d'une interface qui étend (extends) une classe du Framework fourni par Bea (weblogic) afin d'effectuer des requêtes en base de données. Elle contient toutes les fonctions qui, une fois appelées, effectuent des requêtes SQL en base et retournent le résultat obtenu.

Pour mener à bien l'évolution, étant donné l'ajout d'une nouvelle table et de nouvelles colonnes, il m'a fallu créer de nouvelles requêtes SQL pour communiquer avec la base de données pour récupérer, insérer, mettre à jour les données.

8.4.1. Récupérer le pourcentage de distribution

J'ai besoin de récupérer le pourcentage de distribution d'un site pour calculer le besoin qui lui sera attribué lors de la répartition automatique.

⇒ NVL est une fonction qui permet d'attribuer la valeur 0 si le résultat retourné par la requête SQL est null. De cette façon on évite les NullPointerException par exemple.

Un pourcentage de distribution sera de null si, la répartition automatique n'a pas été initialisée par l'utilisateur ou si un nouveau site a été ajouté.

La table T_AVAILABLE_SITES liste tous les sites de livraison disponibles pour un contrat.

Dans les paramètres, il me faut le numéro de contrat, le numéro de ligne de contrat et le code BAAN car ce sont ces trois éléments qui constituent la clé primaire d'un contrat.

```
/*
 * Fonction qui récupère Le pourcentage de distribution d'un site dans La
 * table T_AVAILABLE_SITES
 * @jc:sql
```

```

* statement:::
* SELECT NVL(POURCENTAGE_DISTRIBUTION, 0)
* FROM T_AVAILABLE_SITES
* WHERE SCH_CONTRACT_NUMBER = {SCH_CONTRACT_NUMBER}
* AND SCH_CONTRACT_POSITION = {SCH_CONTRACT_POSITION}
* AND SCH_BAAN_COMPANY_CODE = {SCH_BAAN_COMPANY_CODE}
* AND DELIVERY_SITE_CODE = {DELIVERY_SITE_CODE}
* :::
*/
double selectDistribution(int SCH_CONTRACT_NUMBER, int SCH_CONTRACT_POSITION, String SCH_BAAN_COMPANY_CODE, String DELIVERY_SITE_CODE) throws SQLException, ControlException, SocketCommException, Exception;

```

8.4.2. Calculer la somme des pourcentages de distribution

Afin de vérifier le cas de rejet dans lequel un site n'aurait pas été repris alors qu'un pourcentage de distribution lui était attribué sur le POS N-1, j'ai créé une requête SQL qui va calculer la somme des pourcentages de distribution d'une liste de sites (en l'occurrence les sites du POS N).

```

/**
 * Fonction qui calcule La somme des pourcentages de distribution pour des
sites donnés dans la table T_AVAILABLE_SITES
* @jc:sql
* statement:::
* SELECT NVL(SUM(POURCENTAGE_DISTRIBUTION), 0)
* FROM T_AVAILABLE_SITES
* WHERE SCH_CONTRACT_NUMBER = {SCH_CONTRACT_NUMBER}
* AND SCH_CONTRACT_POSITION = {SCH_CONTRACT_POSITION}
* AND SCH_BAAN_COMPANY_CODE = {SCH_BAAN_COMPANY_CODE}
* AND {sql:fn in("DELIVERY_SITE_CODE", {deliverySites})}
* :::
*/
double calculerSommeDistributions(int SCH_CONTRACT_NUMBER, int SCH_CONTRACT_POSITION, String SCH_BAAN_COMPANY_CODE, String[] deliverySites) throws SQLException, ControlException, SocketCommException, Exception;

```

8.4.3. Récupérer le multiple de conditionnement

Afin de calculer les répartitions selon le multiple de conditionnement, j'ai créé une requête SQL qui va le récupérer dans la table T_SCH_CONTRACT.

Il est à noté que si sa valeur est nulle, je lui assigne une valeur de 1 car tout nombre est multiple de 1, ça ne change donc rien et ça permettra même d'écrire une fonction de moins pour le calcul des répartitions car je ne devrai pas en créer une spécifique pour le cas où aucun multiple de conditionnement n'existe.

Le multiple de conditionnement ne pourra jamais valoir 0. Cet aspect est géré par la partie portail APP_Suivi.

```
/*
 * Fonction qui récupère Le multiple de conditionnement d'un contrat dans
La table T_SCH_CONTRACT
 * @jc:sql
 * statement::
 * SELECT NVL(CONDITIONNING,1)
 * FROM T_SCH_CONTRACT
 * WHERE SCH_CONTRACT_NUMBER = {SCH_CONTRACT_NUMBER}
 * AND SCH_CONTRACT_POSITION = {SCH_CONTRACT_POSITION}
 * AND SCH_BAAN_COMPANY_CODE = {SCH_BAAN_COMPANY_CODE}
 * :::
 */
int selectConditionning(int SCH_CONTRACT_NUMBER, int SCH_CONTRACT_POSITION
, String SCH_BAAN_COMPANY_CODE) throws SQLException, ControlException, SocketC
ommException, Exception;
```

8.4.4. Récupérer le type de distribution

Etant donné que certaines opérations ne doivent avoir lieu que si le type de distribution est dans un état particulier (par exemple l'historisation qui n'a lieu que pour les types AUTO et MANUAL), je dois créer une requête qui peut récupérer ce type.

Cette information est stockée dans la table T_LINES_POS. C'est la table qui retranscrit les besoins attribués pour chaque échéance dans le fichier ESNPOS. Il indique également le type d'échéance – ferme ou flexible – et à présent le type de distribution.

Cette table peut être mise à jour suite à l'action d'un utilisateur. Si celui-ci initialise la distribution automatique pour son contrat, tous les types de distribution pour toutes les échéances du POS en cours passeront à AUTO. Si pour certaines échéances il veut passer en répartition manuelle, les échéances passeront en type MANUEL. Ce changement est effectué par la partie portail d'APP_Suivi.

```
/*
 * Fonction qui récupère Le type de distribution d'un site dans La table T
_LINES_POS
 * @jc:sql
 * statement::
 * SELECT DISTRIBUTION_TYPE
 * FROM T_LINES_POS
 * WHERE POS_NUMBER = {OLD_POS_NUMBER}
 * AND SCH_BAAN_COMPANY_CODE = {SCH_BAAN_COMPANY_CODE}
 * AND POS_DELIVERY_DATE = {POS_DELIVERY_DATE}
 * :::
 */
```



```
String selectTypeDistribution(int OLD_POS_NUMBER, String SCH_BAAN_COMPANY_CODE, Date POS_DELIVERY_DATE) throws SQLException, ControlException, SocketCommException, Exception;
```

8.4.5. Compter le nombre de réceptions partielles

La façon dont la distribution automatique se fera étant dépendante de l'existence de réceptions partielles sur une échéance d'un contrat, j'ai créé une requête SQL qui va les compter.

Comme expliqué plus haut dans ce document, on récupère pour un contrat et une échéance le nombre d'expéditions qui ont été validées (qui ont une date de validation).

Cette requête se fait sur la table T_SHIPMENT_ADVICES qui répertorie toutes les expéditions et les besoins.

```
/**  
 * Fonction qui compte le nombre de réceptions partielles pour un besoin dans la table T_SHIPMENT_ADVICES  
 * @jc:sql  
 * statement:::  
 * SELECT COUNT(E_NUMBER)  
 * FROM T_SHIPMENT_ADVICES  
 * WHERE SCH_CONTRACT_NUMBER = {SCH_CONTRACT_NUMBER}  
 * AND SCH_CONTRACT_POSITION = {SCH_CONTRACT_POSITION}  
 * AND SCH_BAAN_COMPANY_CODE = {SCH_BAAN_COMPANY_CODE}  
 * AND EXPECTED_DELIVERY_DATE = {EXPECTED_DELIVERY_DATE}  
 * AND BL_VALIDATION_DATE IS NOT NULL  
 * :::  
 */  
int compterReceptionsPartielles(int SCH_CONTRACT_NUMBER, int SCH_CONTRACT_POSITION, String SCH_BAAN_COMPANY_CODE, Date EXPECTED_DELIVERY_DATE) throws SQLException, ControlException, SocketCommException, Exception;
```

8.4.6. Compter le nombre de répartition automatique

Pour le cas de rejet sur la vérification qu'un site n'a pas été supprimé alors qu'il avait un pourcentage de répartition assigné, j'ai créé une requête qui va compter le nombre de répartitions automatiques pour un contrat. De cette manière, on ne passe pas systématiquement par cette vérification. Si un contrat ne possède pas de répartition automatique pour ses échéances, c'est qu'il n'a pas de pourcentage de répartitions attribués à ses sites de livraison et il est donc inutile d'appeler ce code.

```
/**  
 * Fonction qui compte le nombre de lignes T_LINES_POS qui ont comme DISTRIBUTION_TYPE 'AUTO' pour un POS donné  
 * @jc:sql  
 * statement:::
```

```

* SELECT COUNT(DISTRIBUTION_TYPE)
* FROM T_LINES_POS
* WHERE POS_NUMBER = {OLD_POS_NUMBER}
* AND SCH_BAAN_COMPANY_CODE = {SCH_BAAN_COMPANY_CODE}
* AND DISTRIBUTION_TYPE = 'AUTO'
* :::
*/
int compterRepartitionsAuto(int OLD_POS_NUMBER, String SCH_BAAN_COMPANY_CODE) throws SQLException, ControlException, SocketCommException, Exception;

```

8.4.7. Récupérer les lignes T_QUANTITIES

Etant donné que je dois historiser les anciens restes à recevoir pour chaque site ayant été réparti manuellement et automatiquement, j'ai créé une requête qui va récupérer les valeurs des lignes de la table T_QUANTITIES pour un contrat et des sites donnés pour le POS N-1.

Je dois faire cette manipulation car la partie du code qui historise est postérieure l'ajout du nouveau POS en base. Lorsque le POS N-1 est remplacé par le POS N, les lignes T_QUANTITIES qu'il possédaient se mettent à 0.

Je devrai donc enregistrer les valeurs avant leur remise à 0 pour pouvoir les exploiter plus tard.

Vous pourrez noter le "@jc:sql iterator-element-type="beans.QuantitiesBean"" qui signifie que le résultat de la requête sera stocké dans un iterator (liste à parcourir) d'objets de type QuantitiesBean. QuantitiesBean est en effet un objet que nous avons qui possèdent en propriétés les différentes colonnes de la table T_QUANTITIES.

```

/**
 * Select des infos T_QUANTITIES pour un POS et des sites donnés
 * @jc:sql iterator-element-type="beans.QuantitiesBean"
 * statement:
 * SELECT DELIVERY_SITE_CODE, POS_DELIVERY_DATE, nvl(REMAIN_TO_DELIVER,0)
AS REMAIN_TO_DELIVER, nvl(REMAIN_TO_RECEIVE,0) AS REMAIN_TO_RECEPT, nvl(TRANSIT_QUANTITY,0) AS TRANSIT_QUANTITY
 * FROM T_QUANTITIES
 * where POS_NUMBER = {POS_NUMBER}
 * and SCH_BAAN_COMPANY_CODE={SCH_BAAN_COMPANY_CODE}
 * and pos_delivery_date = {POS_DELIVERY_DATE}
 * and {sql:fn in("DELIVERY_SITE_CODE", {SITES})}
 * :::
*/
Iterator selectTQuantities(int POS_NUMBER, String SCH_BAAN_COMPANY_CODE, Date POS_DELIVERY_DATE, String[] SITES) throws SQLException, ControlException, SocketCommException, Exception;

```

8.4.8. Requête d'insertion pour historisation

Pour historiser les répartitions automatiques et manuelles, j'ai créé une requête SQL de type INSERT...SELECT.

Etant donné qu'il me faut plein de données éparpillées dans plusieurs tables, j'ai du JOIN plusieurs tables.

Les seules données que je récupère en dehors de la requête sont l'ancien reste à recevoir et le nouveau. Ils seront calculés dans le code.

```
/*
 * insertion de T_DISTRIBUTION_HISTORY
 * @jc:sql
 * statement::
 * INSERT INTO T_DISTRIBUTION_HISTORY (DISTRIBUTION_HIST_TECH_ID, ACTION_DATE,
 * USER_LAST_NAME, USER_FIRST_NAME, USER_LOGIN, SCH_CONTRACT_NUMBER, SCH_CONTRACT_POSITION,
 * SCH_BAAN_COMPANY_CODE,
 * EXPECTED_DELIVERY_DATE, DELIVERY_SITE_CODE, DISTRIBUTION_BEFORE_ACTION, DISTRIBUTION_AFTER_ACTION,
 * POURCENTAGE_BEFORE_ACTION, POURCENTAGE_AFTER_ACTION,
 * CONDITIONNING_BEFORE_ACTION, CONDITIONNING_AFTER_ACTION, INACTIVATED_CONTRACT, DISTRIBUTION_TYPE)
 * SELECT
 * t_distribution_history_seq.NEXTVAL AS DISTRIBUTION_HIST_TECH_ID,
 * sysdate AS ACTION_DATE,
 * 'SYSTEM' AS USER_LAST_NAME,
 * 'SYSTEM' AS USER_FIRST_NAME,
 * 'SYSTEM' AS USER_LOGIN,
 * tship.SCH_CONTRACT_NUMBER,
 * tship.SCH_CONTRACT_POSITION,
 * tship.SCH_BAAN_COMPANY_CODE,
 * tship.EXPECTED_DELIVERY_DATE,
 * tship.DELIVERY_SITE_CODE,
 * {OLD_RAR} AS DISTRIBUTION_BEFORE_ACTION,
 * {NEW_RAR} AS DISTRIBUTION_AFTER_ACTION,
 * tas.POURCENTAGE_DISTRIBUTION,
 * tas.POURCENTAGE_DISTRIBUTION,
 * tsch.CONDITIONNING,
 * tsch.CONDITIONNING,
 * tsch.IS_DISABLED,
 * tlp.DISTRIBUTION_TYPE
 * FROM T_SHIPMENT_ADVICES tship
 * INNER JOIN T_AVAILABLE_SITES tas
 * ON tship.SCH_CONTRACT_NUMBER = tas.SCH_CONTRACT_NUMBER
 * AND tship.SCH_CONTRACT_POSITION = tas.SCH_CONTRACT_POSITION
 * AND tship.SCH_BAAN_COMPANY_CODE = tas.SCH_BAAN_COMPANY_CODE
 * AND tship.DELIVERY_SITE_CODE = tas.DELIVERY_SITE_CODE
 * INNER JOIN T_SCH_CONTRACT tsch
```

```

* ON tas.SCH_CONTRACT_NUMBER = tsch.SCH_CONTRACT_NUMBER
* AND tas.SCH_CONTRACT_POSITION = tsch.SCH_CONTRACT_POSITION
* AND tas.SCH_BAAN_COMPANY_CODE = tsch.SCH_BAAN_COMPANY_CODE
* INNER JOIN T_POS tpos
* ON tsch.SCH_CONTRACT_NUMBER = tpos.SCH_CONTRACT_NUMBER
* AND tsch.SCH_CONTRACT_POSITION = tpos.SCH_CONTRACT_POSITION
* AND tsch.SCH_BAAN_COMPANY_CODE = tpos.SCH_BAAN_COMPANY_CODE
* INNER JOIN T_LINES_POS tlps
* ON tlps.POS_NUMBER = tpos.POS_NUMBER
* AND tship.EXPECTED_DELIVERY_DATE = tlps.POS_DELIVERY_DATE
* AND tship.SCH_BAAN_COMPANY_CODE = tlps.SCH_BAAN_COMPANY_CODE
* WHERE tship.SCH_CONTRACT_NUMBER = {SCH_CONTRACT_NUMBER}
* AND tship.SCH_CONTRACT_POSITION = {SCH_CONTRACT_POSITION}
* AND tship.SCH_BAAN_COMPANY_CODE = {SCH_BAAN_COMPANY_CODE}
* AND tship.EXPECTED_DELIVERY_DATE = {EXPECTED_DELIVERY_DATE}
* AND tlps.POS_NUMBER = {POS_NUMBER}
* AND tas.DELIVERY_SITE_CODE = {DELIVERY_SITE_CODE}
* AND rownum < 2
*::
*/
void insertTDistributionHistory(int OLD_RAR,int NEW_RAR, int SCH_CONTRACT_
NUMBER, int SCH_CONTRACT_POSITION, String SCH_BAAN_COMPANY_CODE, Date EXPECTED
_DELIVERY_DATE, int POS_NUMBER, String DELIVERY_SITE_CODE) throws SQLException
, ControlException, SocketCommException, Exception;

```

8.4.9. Récupérer tous les sites d'un contrat

Enfin, toujours pour le cas de rejet sur le site supprimé avec un pourcentage de répartition assigné, il faudra, comme pour T_QUANTITIES, que je récupère la liste des sites présents dans la table T_AVAILABLE_SITES ainsi que leur pourcentage de répartition pour le POS N-1 car lors du passage du nouveau programme (POS N), celui-ci met directement à jour cette table selon ce qui est indiqué dans le fichier. Je dois donc récupérer en amont cette liste afin de pouvoir effectuer mes comparaisons pour vérifier mon rejet plus tard dans le code.

J'utilise l'objet AvailableSite qui possède en propriétés les colonnes de la table T_AVAILABLE_SITES pour stocker les informations. J'ai utilisé une liste de AvailableSite pour varier de l'iterator.

```

/**
 * Récupère une liste de T_AVAILABLE_SITES
 * @jc:sql
 * statement="SELECT * FROM T_AVAILABLE_SITES WHERE SCH_CONTRACT_NUMBER =
{i_iSCH_CONTRACT_NUMBER} AND SCH_CONTRACT_POSITION = {i_iSCH_CONTRACT_POSITION}
AND SCH_BAAN_COMPANY_CODE = {i_sSCH_BAAN_COMPANY_CODE}"
*/
AvailableSite[] getAllTAvailableSites(int i_iSCH_CONTRACT_NUMBER, int i_iS
CH_CONTRACT_POSITION, String i_sSCH_BAAN_COMPANY_CODE) throws SQLException, Co
ntrolException, SocketCommException, Exception;

```



Toutes ces requêtes SQL pourront être réutilisées pour de futurs développements sur le flux POS et pourront constituer un gain de temps.

8.5. Annexe 5 - Code Java FEB 2628

Ci-dessous, le code Java réalisé dans le cadre de mon évolution sur le flux WLI POS pour l'ajout de la fonctionnalité de répartition automatique.

La méthode principale se nomme control. C'est dans cette méthode qu'est vérifiée la cohérence des données présentes dans le fichier ESNPOS ainsi que les données présentes en base de données. Il y a également l'intégration des données en base si tout est correct.

```
1. //Récupération du BeanLigne9 ni ferme ni flexible
2. Ligne9Bean[] ligne9BeanNiFermeNiFlex = ((Ligne9Bean[])listeTempLigne9.toArray(new Ligne9
   Bean[listeTempLigne9.size()]));
3.
4. int numeroPOS = Integer.valueOf(i_oInfo.getNUMERO_POS()).intValue();
5. int ancienNumeroPOS = l_iNumberPOSAncien;
6. String baanCompanyCode = i_oInfo.getPATTERN();
7. int schContract = Integer.parseInt(i_oInfo.getN_CONTRAT_SCH());
8. int schContractLigne = Integer.parseInt(i_oInfo.getN_LIGNE_CONTRAT_SCH());
9. String[] deliverySitesESNPOS = Fonction.getDeliverySitesESNPOS(l_oBeanTab);
10. Date[] deliveryDatesESNPOS = Fonction.getDeliveryDatesESNPOS(ligne9BeanNiFermeNiFlex);
11. Iterator deliverySitesAndDatesWithTransitNotInESNPOS = this.app_suivi.selectDeliverySite
   sAndDatesWithTransitNotInESNPOS(ancienNumeroPOS, baanCompanyCode, deliverySitesESNPOS, d
   eliveryDatesESNPOS);
12. int multiple = this.app_suivi.selectConditionning(schContract, schContractLigne, baanCom
   panyCode);
13.
14. //Contrôles qui donnent lieu à des rejets pour les POS fermes.
15. verifierRejetPOS(l_oBeanTabFerme, i_oInfo, deliverySitesAndDatesWithTransitNotInESNPOS,
   numeroPOS, ancienNumeroPOS, baanCompanyCode, deliverySitesESNPOS, schContract, schContra
   ctLigne, multiple, oldAvailableSites);
16. //Contrôles pour toutes les lignes 9 des POS fermes et flexibles
17. verifierIntegrationPOS(ligne9BeanNiFermeNiFlex, l_oBeanTab, ancienNumeroPOS, baanCompany
   Code, i_oInfo, numeroPOS, l_sCode_Livraison, l_sType_Livraison, previousDefaultSite, del
   iverySitesESNPOS, multiple);
18. logger.debug("Nouveau POS intégré");
```

Dans ce code, il y a principalement la récupération dans des variables de données (Lignes 2 à 12) qui seront utilisées dans plusieurs méthodes pour effectuer des vérifications et des traitements.

Ensuite dans **verifierRejetPOS()** – L15 - on va d'abord vérifier que le contenu du fichier est bon avant d'intégrer les données avec **verifierIntegrationPOS()** – L17.

Dans la méthode **verifierRejetPOS()** qui existait déjà, j'ai rajouté le code suivant à l'algorithme existant (je n'ai repris que les principales lignes du code existant pour encadrer mon code) :

```

1. if(l_oBeanTabFerme.length > 0){
2.     logger.debug("Il y a des expéditions fermes à analyser !");
3.     //Si il existe des transit POS N-1 non repris dans le POS N
4.     for(int i = 0; i <l_oBeanTabFerme.length; i++){
5.         //#2628 Si on a un multiple de conditionnement,
6.         le besoin pour chaque échéance doit être un multiple de celui-ci
7.         if(multiple != 1){
8.             verifierMultiple(i_oInfo, multiple, remainToReceiveESNPOS, numeroPOS, ancien
NumeroPOS, expectedDeliveryDate);
9.         }
10.    }
11.    //#2628 Si le POS N-1 a au moins une échéance en répartition auto,
12.    on vérifie les pourcentages de distribution sur les sites
13.    if(this.app_suivi.compterRepartitionsAuto(ancienNumeroPOS, baanCompanyCode) > 0){
14.        verifierDistributions(i_oInfo, baanCompanyCode, deliverySitesESNPOS, numeroPOS,
ancienNumeroPOS, oldAvailableSites);
15.    }
16. }
```

Si on a des échéances fermes, on va vérifier que leur besoin est bien un multiple du multiple de conditionnement : **verifierMultiple()**. Ensuite, si le contrat contient des répartitions auto, on va vérifier la somme des pourcentages de distribution : **verifierDistributions()**.

```

1. private void verifierMultiple(InfoBean i_oInfo, int multiple, double remainToReceiveESNPOS
, int numeroPOS, int ancienNumeroPOS, Date expectedDeliveryDate) throws POSRejectedException{
2.     boolean isMultiple = remainToReceiveESNPOS%multiple == 0;
3.     logger.debug("Est un multiple ? : "+isMultiple);
4.     if(!isMultiple){
5.         logger.debug("REJET - Le reste à recevoir pour l'échéance "+expectedDeliveryDa
te+" n'est pas un multiple du multiple de conditionnement.");
6.         HashMap messages = buildMessageRarIsNotMultiple(i_oInfo, expectedDeliveryDate,
multiple, remainToReceiveESNPOS);
7.         sendEmailPosRejected(i_oInfo, messages.get("fr").toString(), messages.get("eng
").toString());
8.         logger.debug("REJET effectué");
9.         throw new POSRejectedException("Le reste à recevoir pour l'échéance "+expected
DeliveryDate+" n'est pas un multiple du multiple de conditionnement.", numeroPOS, ancienNu
meroPOS);
10.    }
11. }
```

La méthode **verifierMultiple()** prend simplement le besoin de l'échéance et vérifie à l'aide d'un modulo s'il y a un reste. S'il y en a un, le besoin n'est pas un multiple et donc le nouveau programme est rejeté, une **POSRejectedException** est levée pour interrompre le traitement et un e-mail est envoyé à l'approvisionneur.

J'ai créé une méthode « générique » pour envoyer les e-mails aux approvisionneurs (**sendEmailPosRejected**) car on commence à avoir pas mal de cas de rejets différents et je ne voulais pas qu'on duplique le code d'envoi de l'e-mail à chaque fois comme c'était prévu initialement par le code. Cependant, chaque rejet a son propre corps de message, j'ai donc créé une fonction pour les construire en anglais et en français car l'application est bilingue.

```

1.  private HashMap buildMessageRarIsNotMultiple(InfoBean i_oInfo, Date expectedDeliveryDat
e, int multiple, double remainToReceiveESNPOS){
2.      String refArticle = i_oInfo.getREFERENCE_ARTICLE_SNECMA();
3.      HashMap messages = new HashMap();
4.
5.      String messageFr ="Le POS n'a pas été intégré dans App_suivi.\n\n"
6.      +"POS_Number : " +i_oInfo.getNUMERO_POS()+" , ITEM_REFERENCE : " +refArticle+".\n"
7.      +"Le besoin doit être un multiple du conditionnement renseigné.\n"
8.      +"Le besoin de "+remainToReceiveESNPOS+" pour l'échéance "+expectedDeliveryDate+" n
'est pas un multiple de "+multiple+".\n\n"
9.      +"Merci de corriger votre programme et de le renvoyer.";
10.
11.     String messageEng ="The POS has not been integrated in App_suivi.\n\n"
12.     +"POS_Number : " +i_oInfo.getNUMERO_POS()+" , ITEM_REFERENCE : " +refArticle+".\n"
13.     +"The need has to be a multiple of the conditioning multiple given.\n"
14.     +"The need of "+remainToReceiveESNPOS+" for the expected date "+expectedDeliveryDat
e+" isn't a multiple of "+multiple+".\n\n"
15.     +"Please correct your program and send it back.";
16.
17.     messages.put("fr", messageFr);
18.     messages.put("eng", messageEng);
19.     return messages;
20. }

```

Voici la méthode qui construit le message pour le cas de rejet où le reste à recevoir du fichier ESNPOS n'est pas un multiple du multiple de conditionnement. Elle prend en paramètre les informations nécessaires à afficher dans le message et retourne un **HashMap** (tableau associatif) des messages anglais et français.

La méthode **sendEmailPosRejected()** reprend le code qui existait déjà, j'y ai simplement glissé les variables messageFr et messageEng et rajouté en paramètre ces messages.

La méthode **verifierDistributions()** calcule la somme des pourcentages de distribution et si elle est différente de 100, une exception est levée, le nouveau programme est rejeté et un e-mail est envoyé à l'approvisionneur. Il y a une méthode qui récupère la liste des sites qui ont un pourcentage de distribution en base et qui n'ont pas été repris dans le fichier afin de les lister dans l'e-mail : **getDeletedSitesWithDistribution()**.

```

1.  private void verifierDistributions(InfoBean i_oInfo, String baanCompanyCode, String[] de
liverySitesESNPOS, int numeroPOS, int ancienNumeroPOS, AvailableSite[] oldAvailableSites)
throws SQLException, SocketCommException, Exception{
2.      int schContract = Integer.parseInt(i_oInfo.getN_CONTRAT_SCH());
3.      int schContractLigne = Integer.parseInt(i_oInfo.getN_LIGNE_CONTRAT_SCH());
4.
5.      double sommePourcentagesDistribution = this.app_suivi.calculerSommeDistributions
(schContract, schContractLigne, baanCompanyCode, deliverySitesESNPOS);
6.      logger.debug("Somme des pourcentages de distribution : "+sommePourcentagesDistri
bution);
7.      if( sommePourcentagesDistribution!= 100){
8.          logger.debug("REJET - La somme des pourcentages de distribution des sites du
fichier ESNPOS n'est pas égale à 100%");
9.
10.         //Récupération des sites supprimés s'il y en a
11.         List deletedSitesWithDistribution = getDeletedSitesWithDistribution(oldAvail
ableSites, deliverySitesESNPOS);
12.
13.         HashMap messages = buildMessageDistributionsNotCorrect(i_oInfo, deletedSites
WithDistribution);

```



```
14.         sendEmailPosRejected(i_oInfo, messages.get("fr").toString(), messages.get("e
    ng").toString());
15.         logger.debug("Message FR : "+messages.get("fr").toString());
16.         logger.debug("Message ENG : "+messages.get("eng").toString());
17.         logger.debug("REJET effectué");
18.         throw new POSRejectedException("La somme des pourcentages de distribution de
    s sites du fichier ESNPOS n'est pas égale à 100%", numeroPOS, ancienNumeroPOS);
19.     }
20. }
```

La méthode **getDeletedSitesWithDistribution()** prend en paramètres les sites qui ont un pourcentage de distribution sur le POS N-1 et les sites présents dans le POS N. Une double boucle simple vérifie si chaque site du POS N-1 est présent dans le POS N.

Il est important de souligner que je compare mes chaînes de caractère avec un « **equals()** » et pas un « **==** » car ce dernier ne compare pas les caractères mais bien l'objet String, ce qui n'est pas ce que je recherche et ça peut même causer des erreurs car deux String peuvent contenir les mêmes caractères sans pour autant être considérés comme étant le même objet.

Si le site du POS N-1 correspond au site du POS N ou si le site du POS N-1 n'avait pas de pourcentage de distribution (donc ce n'est pas grave s'il n'est pas repris car rien ne devait lui est attribué), il est considéré comme repris et ne sera pas retourné par la fonction.

```
1. private List getDeletedSitesWithDistribution(AvailableSite[] oldAvailableSites, String[] de
    liverySitesESNPOS){
2.     boolean isDeleted = true;
3.     List deletedSites = new ArrayList();
4.     for(int i=0; i<oldAvailableSites.length;i++){
5.         isDeleted = true;
6.         for(int j=0; j<deliverySitesESNPOS.length; j++){
7.             if(oldAvailableSites[i].getDELIVERY_SITE_CODE().equals(deliverySitesESNPOS[
                j]) || oldAvailableSites[i].getPOURCENTAGE_DISTRIBUTION() == null || new Double(0).eq
                uals(oldAvailableSites[i].getPOURCENTAGE_DISTRIBUTION())){
8.                 isDeleted = false;
9.             }
10.            }
11.            if(isDeleted){
12.                deletedSites.add(oldAvailableSites[i].getDELIVERY_SITE_CODE());
13.            }
14.        }
15.    }
16. }
```

La méthode **verifierIntegrationPOS()** n'est appelée que si aucun rejet n'a eu lieu et elle va intégrer toutes les échéances fermes et flexibles du nouveau programme en base pour remplacer l'ancien (**integrationLigne9()**).

```
1. private void verifierIntegrationPOS(Ligne9Bean[] ligne9BeanNiFermeNiFlex, Ligne8Bean[] l_o
    BeanTab, int ancienNumeroPOS, String baanCompanyCode, InfoBean i_oInfo, int numeroPO
    S, String l_sCode_Livraison, String l_sType_Livraison, String previousDefaultSite, S
    tring[] deliverySitesESNPOS, int multiple) throws Exception{
2.     if(ligne9BeanNiFermeNiFlex.length > 0){
3.         afficherResultatStringTab(deliverySitesESNPOS);
4.         integrationLigne9(ligne9BeanNiFermeNiFlex, l_sCode_Livraison, i_oInfo, numeroP
        OS, baanCompanyCode, l_sType_Livraison, ancienNumeroPOS, deliverySitesESNPOS, previo
        usDefaultSite, multiple);
5.     }
6. }
```

Dans la méthode **integrationLigne9()** qui existait déjà, j'ai ajouté une méthode pour effectuer la répartition automatique si le type de distribution de l'échéance en cours sur le POS N-1 est « AUTO » et si l'échéance est ferme : **performRepartitionAuto()** ; A l'inverse, j'ai mis l'ancien code pour la répartition classique dans **performRepartition()** pour gagner en visibilité. Suite aux intégrations, si la répartition est « AUTO » ou « MANUELLE », on historise, **historiser()**, dans la nouvelle table T_DISTRIBUTION_HISTORY.

```

1.      //2628 Répartition auto
2.      String typeDistribution = this.app_suivi.selectTypeDistribution(ancienNumeroPOS, baanCompanyCode, expectedDeliveryDate);
3.      logger.debug("Type de distribution : "+typeDistribution);
4.      if("1".equals(ligne9BeanNiFermeNiFlex[i].getTYPE_BESOIN()) || "4".equals(ligne9BeanNiFermeNiFlex[i].getTYPE_BESOIN()) && REPARTITION_AUTO.equals(typeDistribution)){
5.          logger.debug("Echéance ferme et distribution AUTO");
6.          performRepartitionAuto(rarESNPOS, rarDB, ancienNumeroPOS, baanCompanyCode, expectedDeliveryDate, numeroPOS, sameDeliverySites, l_sType_Livraison, sitesInESNPOS, ligne9BeanNiFermeNiFlex[i], deliverySite, schContract, schContractLigne, multiple);
7.      }else{
8.          logger.debug("Echéance flexible ou ferme et distribution manuelle ou non initialisée");
9.          performRepartition(rarESNPOS, rarDB, newDefaultSite, ancienNumeroPOS, baanCompanyCode, expectedDeliveryDate, numeroPOS, sameDeliverySites, l_sType_Livraison, sitesInESNPOS, l_sCode_Livraison, ligne9BeanNiFermeNiFlex[i]);
10.     }
11.     if(REPARTITION_AUTO.equals(typeDistribution) || REPARTITION_MANUELLE.equals(typeDistribution)){
12.         historiser(numeroPOS, baanCompanyCode, expectedDeliveryDate, sameDeliverySites, oldTQuantities, i, schContract, schContractLigne);
13.     }

```

La méthode **performRepartitionAuto()** respecte l'algorithme mis en place lors de la phase de conception et expliqué dans la partie 4.5.2.3. On peut voir l'appel aux différentes méthodes qui récupèrent les données en base présentées en annexe n°4.

```

1. private void performRepartitionAuto(double rarESNPOS, double rarDB, int ancienNumeroPOS, String baanCompanyCode, Date expectedDeliveryDate, int numeroPOS, String[] sameDeliverySites, String l_sType_Livraison, String[] sitesInESNPOS, Ligne9Bean ligne9BeanNiFermeNiFlex, String deliverySite, int schContract, int schContractLigne, int multiple) throws SQLException, SocketCommException, Exception{
2.     logger.debug("REPARTITION AUTO - IN");
3.     int receptionsPartielles = this.app_suivi.compterReceptionsPartielles(schContract, schContractLigne, baanCompanyCode, expectedDeliveryDate);
4.     if(receptionsPartielles > 0){
5.         //TODO ligne inexiste? Vérifier dans les tests que ça fonctionne quand aucune ancienne ligne T_QUANTITIES n'existe
6.         if(rareSNPOS < rarDB){
7.             this.app_suivi.copyTQuantitiesToNewPOSOnlyWithTransit(ancienNumeroPOS, baanCompanyCode, expectedDeliveryDate, numeroPOS, sameDeliverySites, l_sType_Livraison);
8.         }else{
9.             this.app_suivi.copyTQuantitiesToNewPOS(ancienNumeroPOS, baanCompanyCode, expectedDeliveryDate, numeroPOS, sameDeliverySites, l_sType_Livraison);
10.        }
11.    }else{
12.        this.app_suivi.copyTQuantitiesToNewPOSOnlyWithTransit(ancienNumeroPOS, baanCompanyCode, expectedDeliveryDate, numeroPOS, sameDeliverySites, l_sType_Livraison);
13.    }
14.    //Calcul du delta
15.    logger.debug("Calcul du delta");

```

```

16.             Double rarESNPOSDouble = Double.valueOf(ligne9BeanNiFermeNiFlex.getQUANTITE
    _BESOIN());
17.             double delta = this.app_suivi.compareRemainToReceive(numeroPOS, baanCompany
    Code, expectedDeliveryDate, sitesInESNPOS, rarESNPOSDouble);
18.             logger.debug("Delta = "+delta);
19.             if(delta > 0){
20.                 logger.debug("Le Delta est > 0 : répartition automatique sur les sites
    selon les pourcentages");
21.                 Double deltaDouble = new Double(delta);
22.                 Double deltaRestant = new Double(delta);
23.                 logger.debug("Delta : "+deltaDouble);
24.                 logger.debug("Multiple de conditionnement du contrat : "+multiple);
25.                 for(int i=0; i<sameDeliverySites.length; i++){
26.                     if(!sameDeliverySites[i].equals(deliverySite)){
27.                         double pourcentage = this.app_suivi.selectDistribution(schC
    ontract, schContractLigne, baanCompanyCode, sameDeliverySites[i]);
28.                         Double toAdd = calculerRepartition(deltaDouble, pourcentage
    , multiple);
29.                         logger.debug("Remain To Deliver pour le site "+sameDelivery
    Sites[i]+" = "+toAdd);
30.                         this.app_suivi.addRemainToDeliver(numeroPOS, baanCompanyCod
    e, expectedDeliveryDate, sameDeliverySites[i], toAdd);
31.                         double doubleDeltaRestant = deltaRestant.doubleValue();
32.                         double doubleToAdd = toAdd.doubleValue();
33.                         deltaRestant = new Double(doubleDeltaRestant - doubleToAdd)
    ;
34.                         logger.debug("Delta restant = "+deltaRestant);
35.                     }
36.                 }
37.                 //Ajout de tout le delta restant sur le site par défaut :
38.                 this.app_suivi.addRemainToDeliver(numeroPOS, baanCompanyCode, expectedD
    eliveryDate, deliverySite, deltaRestant);
39.             }
40.             logger.debug("REPARTITION AUTO - END");
41.         }

```

De la ligne 19 à la ligne 38, on peut voir qu'on calcule un premier delta qu'on va répartir sur les différents sites qui ont un pourcentage de distribution en base. J'effectue sur cette opération sur tous les sites qui ne sont pas celui par défaut car dans tous les cas, le site par défaut récupère le surplus. J'ai donc un second delta dans lequel est mis les restes de l'application du multiple de conditionnement sur le premier delta ainsi que le reste à recevoir restant après déduction des pourcentages de distribution.

Enfin, la dernière méthode **historiser()**, va insérer dans la table **T_DISTRIBUTION_HISTORY**, par l'intermédiaire d'une boucle, une entrées pour chaque site en commun entre le POS N et le POS N-1 (L10 – L31).

De la ligne 4 à la ligne 9, je récupère les transits du POS N.

De la ligne 15 à la ligne 22, je récupère le reste à recevoir pour le POS N-1 et le site j à l'échéance i. Etant donné que mon objet **oldTQuantities** contient tous les restes à recevoir pour toutes les échéances et tous les sites du POS N-1, je le parcours à l'aide d'une boucle et lorsque l'échéance et le site correspondent à l'échéance i et le site j, je récupère la valeur dans **oldRarForSite** et je **break** la boucle pour ne pas continuer à tous vérifier car ça serait inutile.

De la ligne 23 à 30 je fais pareil mais pour le POS N.

```

1.      private void historiser(int numeroPOS, String baanCompanyCode, Date expectedDeliveryDate, String[] sameDeliverySites, List oldTQuantities, int i, int schContract, int schContractLigne) throws SQLException, SocketCommException, Exception{
2.          logger.debug("Enregistrement dans T_DISTRIBUTION_HISORY - IN");
3.          //Récupération des lignes T_QUANTITIES du POS N-
4.          1 pour l'insertion dans T_DISTRIBUTION_HISTORY
5.          List newTQuantities = new ArrayList();
6.          Iterator iteratorTQuantities = this.app_suivi.selectTQuantities(numeroPOS, baanCompanyCode, expectedDeliveryDate, sameDeliverySites);
7.          while (iteratorTQuantities.hasNext()){
8.              newTQuantities.add((QuantitiesBean) iteratorTQuantities.next());
9.          }
10.         //Il faut faire un enregistrement par site
11.         for(int j=0; j<sameDeliverySites.length; j++){
12.             logger.debug("Site en cours : "+sameDeliverySites[j]);
13.             String oldRarForSite = "";
14.             String newRarForSite = "";
15.
16.             for(int k=0; k<oldTQuantities.size(); k++){
17.                 QuantitiesBean q = (QuantitiesBean) oldTQuantities.get(k);
18.                 if(sameDeliverySites[j].equals(q.getDELIVERY_SITE_CODE())){
19.                     oldRarForSite = q.getREMAIN_TO_RECEP();
20.                     logger.debug("Ancien RAR : "+oldRarForSite);
21.                     break;
22.                 }
23.             }
24.             for(int k=0; k<newTQuantities.size(); k++){
25.                 QuantitiesBean q = (QuantitiesBean) newTQuantities.get(k);
26.                 if(sameDeliverySites[j].equals(q.getDELIVERY_SITE_CODE())){
27.                     newRarForSite = q.getREMAIN_TO_RECEP();
28.                     logger.debug("Nouveau RAR : "+newRarForSite);
29.                     break;
30.                 }
31.             this.app_suivi.insertTDistributionHistory(Integer.parseInt(oldRarForSite), Integer.parseInt(newRarForSite), schContract, schContractLigne, baanCompanyCode, expectedDeliveryDate, numeroPOS, sameDeliverySites[j]);
32.         }
33.         logger.debug("Enregistrement dans T_DISTRIBUTION_HISORY - OUT");
34.     }

```

La complication principale de cette méthode était de réussir à conserver les données du POS N-1 après l'insert du POS N par les méthodes précédentes car les transits du POS N-1 sont remis à 0, les sites sont remplacés par ceux du nouveau programme etc... J'ai donc dû créer des variables dans mon code avant l'intégration du nouveau programme pour récupérer les données du POS N-1 en base afin de pouvoir les réutiliser.

8.6. Annexe 6 – Gabarit de compétences CDAN IPI



CDAN

Commencer par le référentiel :



Bloc de compétences : Qualité et sécurisation du code réalisé

Compétences ou capacités qui seront évaluées	Critères d'évaluation	Exemples d'activités et tâches	Activités pratiquées	Origine de l'acquisition	Preuves apportées & ref. annexe
- Formaliser, identifier les résultats attendus.	La liste de contrôle des attendus fonctionnels est paraphée.	Étude de l'existant. Rédaction du cahier des spécifications fonctionnelles.	Etude de l'existant Rédaction de règles de gestion	E(6M) F(2M) B(3M)	Mes missions réalisées Pages 24 -> 28 ont nécessité des études de l'existant cahier des spécifications : Annexe 8
Respecter des contraintes. Respecter les recommandations qualité de la norme en vigueur pour l'architecture des logiciels. Anticiper les évolutions. Qualifier les risques	Un plan d'assurance qualité est observé. L'application est organisée en couches indépendantes. Les règles métier sont encapsulées dans des services logiciels. L'accès aux données est réalisé par des services logiciels indépendants du mode de stockage. L'exécution de l'application est répartie	Conception/architecture d'applications logicielles. Conception de services métiers. Conception de services d'accès aux données. Détermination du nombre de tiers de l'application.	Respect des règles de gestion établies Conception de services dans la couche modèle d'une application Conception de classes d'accès aux données dans la couche Données d'une application Réalisation d'une application en plusieurs couches, organisation MVC	E(1A) E(2 S) F(3 M) E(2 S) F(3M) E(2S) F(3M)	Mes missions réalisées Pages 24 -> 28 respectent les spécifications fonctionnelles Page 53 + Annexe 9 Page 53,54,55 + Annexe 9 Page 53,54,55 + Annexe 9

IPI - Comment faire mon dossier de validation ?



CDAN

Respecter une norme de présentation des écrans et documents de sortie.	entre un nombre d'ordinateurs adapté au contexte. Un formulaire d'estimation des risques est rempli. Une norme de présentation des données est respectée. Les interfaces Homme/Machine sont validées.	Réalisation d'une interface homme/machine (IHM) Estimation, qualification des risques sécurité. Réalisation des maquettes de sorties interactives. Réalisation des maquettes de sortie imprimée.	Adaptation d'une vue (Ajout de données, alertes) Création d'une IHM Android Création d'un site de gestion de budget Réalisation de maquettes pour une application de gestion de budget	E(2 S) F(3M) F(2M) F(1M)	Page 53 + Annexe 11 Annexe 10
Concevoir des programmes avec une orientation objets. Garantir un accès sécurisé aux données.	Une programmation orientée objets est utilisée. Le taux de réutilisation du code utile est > 80 %. Des gabarits sont utilisés. Une charte de nommage est utilisée.	Programmation de logiciels.	Programmation en langage Java orienté objet Réutilisation du code existant Respect de la charte de nommage des fichiers	E(1A) E(1A) E(1A)	Page 53,54,55 + Annexe 5, Annexe 9 Page 25, 44, 72 Page 30

IPI - Comment faire mon dossier de validation ?

Livrer le logiciel déverminé.	<p>Le taux de documentation interne du code est > 8 % et < 15 %.</p> <p>Les anomalies d'accès aux données ne génèrent pas d'interruption de l'exécution et sont répertoriées.</p>	<p>Programmation de l'accès aux données de l'entreprise.</p> <p>Tests unitaires. Préparation des jeux de tests.</p> <p>Contrôles de l'existence d'anomalies.</p> <p>Recettage du logiciel. Validation d'une étape du projet.</p>	<p>Mise en place de javadoc pour documenter le code</p> <p>Réalisation de TU et jeux de données</p> <p>Réalisation de tests d'intégration avec jeux de données.</p> <p>Envoi du code en recette</p> <p>Respect des règles de gestion établies</p> <p>Utilisation d'un logiciel pour assurer la conservation de la qualité du code</p> <p>Envoi du code en recette</p> <p>Envoi en production lorsque recette validée</p> <p>Envoi du code en production</p>	<p>E(6M)</p> <p>E(1 A)</p> <p>E(1A)</p> <p>E(1A)</p> <p>E(1A)</p> <p>E(1A)</p> <p>E(1A)</p>	<p>Annexe 13</p> <p>Pages 46, 47, 48, 49, 51</p> <p>Page 52</p> <p>Pages 51, 58</p> <p><small>des missions réalisées Pages 24 à 28 respectent les spécifications fonctionnelles</small></p> <p>Page 27</p> <p>Page 58</p> <p>Page 58</p> <p>Pages 19, 25, 26, 58</p>
Livrer le logiciel conforme aux attentes.	<p>Des outils de contrôle automatique du code sont utilisés.</p> <p>Aucun défaut visible ne persiste.</p> <p>Les contraintes spécifiques au projet sont respectées.</p> <p>Un manuel d'assurance qualité est respecté.</p> <p>Une méthode de recettage est utilisée.</p> <p>L'étape du projet est validée.</p>				
Clôturer une mission.	Le PV de réception du logiciel est validé.	Mise en exploitation.		E(1A)	

IPI - Comment faire mon dossier de validation ?

Bloc de compétences : Audit, conception, méthode de projet

Compétences ou capacités qui seront évaluées	Critères d'évaluation	Exemples d'activités et tâches	Activités pratiquées	Origine de l'acquisition	Preuves apportées & ref. annexe
Formaliser des processus	<p>La procédure du service utilisateur est formalisée et validée.</p> <p>La procédure du service utilisateur est conforme aux règles du système de management des services de l'entreprise.</p> <p>La circulation du document résultat du traitement prévu est matérialisée dans un diagramme de workflow.</p>	<p>Étude de l'existant. Identification des procédures en place.</p> <p>Contrôle de la conformité des procédures utilisées avec la gouvernance de l'entreprise.</p> <p>Recensement des documents utilisés, identification de leur circulation et des acteurs concernés.</p> <p>Reconfiguration de procédure.</p>			
Formaliser les règles de gestion et d'organisation des données de l'entreprise.	<p>La proposition de reconstruction de la procédure est validée.</p> <p>La base de données est modélisée.</p>	<p>Conception d'une base de données.</p>	<p>Conception de base de données</p> <p>Réalisation de MCD/MLD</p>	<p>E(6M) F(6M) B(3M)</p>	<p>Annexe 12, annexe 4 Page 50</p>

IPI - Comment faire mon dossier de validation ?



Une méthode de conception par objets est utilisée.	Concevoir des éléments logiciels réutilisables.	Conception de l'architecture applicative.	Programmation orientée objets	E (1A) F(1A)	Pages 53,54 Annexe 5, annexe 9
Une méthode AGILE est utilisée. Absence de signaux d'alertes au point de contrôle du projet.	Produire du logiciel en équipe. Remonter les alertes au(x) décideur(s).	Programmation en équipe. Écriture de code. Coordination de l'avancement.	Travail sur une évolution à plusieurs. Travail sur du code à plusieurs Communication de mon avancement. Remontée de potentiels problèmes	E(6M) F(2M) E(1A)	Pages 25, 32 Page 19, 20
Les étapes du projet sont planifiées.	Estimer des délais.	Planification des tâches du projet.	Estimation de mon reste à faire	E(1A)	Page 20 + Annexe 1
Le projet est conforme au schéma directeur de l'entreprise et respecte les principes d'urbanisation du S.I. Les spécifications fonctionnelles produites respectent le cahier des charges fourni. L'impact de modification est acceptable.	Concevoir une solution logicielle. Anticiper des répercussions.	Conception de la solution logicielle.	Conception d'algorithmes Pré-reflexion sur des méthodes, classes à créer, modifier, réutiliser Analyser l'existant et remonter les problèmes ressentis	E(6M) E(2M)	Pages 40 > 45 Page 35

IPI - Comment faire mon dossier de validation ?



Bloc de compétences : Réalisation d'applications logicielles

Compétences ou capacités qui seront évaluées	Critères d'évaluation	Exemples d'activités et tâches	Activités pratiquées	Origine de l'acquisition	Preuves apportées & ref. annexe
Encapsuler des solutions logicielles spécifiques dans des services logiciels génériques.	Le service d'accès aux données est opérationnel.	Programmation. Investigations documentaires fonctionnelles ou techniques complémentaires.	Création d'une couche accès aux données fonctionnelle	E(2S) F(3M)	Pages 53, 54 Annexe 9
Produire du logiciel générique réutilisable.	Des services logiciels internes sont réutilisables.	Transcription des spécifications fonctionnelles en algorithmes.	Conception d'algorithme suite à la lecture des spec. fonctionnelles	E(6M)	Pages 40 > 45
Produire du logiciel partageable.	Des services logiciels sont partageables en local.	Transcription des algorithmes en code source. Compilation, déverminage du code source.	Adaptation des algorithmes pensés en code commut du code après tests validés	E(6M)	Annexe 5
Intégrer des éléments logiciels hétérogènes et produire des exécutables livrables.	Des services logiciels sont partageables à distance.	Agglomération des différents éléments logiciels en unités de traitement, réalisation des tests unitaires.	Réalisation de tests unitaires sur les méthodes créées	E(6M)	Page 27
Modifier un algorithme sans générer de dysfonctionnements.	Le logiciel est livrable, prêt pour la mise en production.	Réalisation des tests unitaires sur les méthodes créées	Réalisation de tests d'intégration sur le code final	E(1A)	Pages 46,47,48,49, 51
Contrôler des délais.	La modification n'entraîne pas de régression fonctionnelle. Le compte-rendu d'activité est renseigné,	Mise à jour du planning de réalisation.	Evaluation quotidienne du reste à faire	E(1A)	Page 52

IPI - Comment faire mon dossier de validation ?

	les écarts sont constatés.		Le logiciel utilisé indique les écarts	E(1A)	Annexe 1
--	----------------------------	--	--	-------	----------

Bloc de compétences : Communiquer avec les acteurs du projet

Compétences ou capacités qui seront évaluées	Critères d'évaluation	Exemples d'activités et tâches	Activités pratiquées	Origine de l'acquisition	Preuves apportées & ref. annexe
User d'une communication professionnelle tant en français qu'en anglais. Interagir efficacement dans un environnement de travail collaboratif.	Le compte-rendu de la réunion est validé. Le score du TOEIC est > 749 Le document collectant l'expression des besoins des utilisateurs est validé. L'aide du logiciel est rédigée. La documentation du livrable est diffusée. La présentation est appréciée.	Élaboration et rédaction de documents techniques, commerciaux ou internes à destination, des utilisateurs, des clients ou des collaborateurs, ... Rédaction des spécifications fonctionnelles de la solution informatique. Écriture des interfaces homme/machine. Relations avec les clients. Animation de réunions de travail et interviews d'utilisateurs. Démonstrations, recettage de livrables.	Rédaction de documentation interne Rédaction de procédures pour les utilisateurs Réponses au utilisateurs (parfois anglais) Rédaction de règles de gestion Relation avec client, utilisateurs Recettage des livrables Assistance recette	E(2S) E(1S) E(1A) F(2M) B(3M) E(1A)	Annexes 2,3 Annexe 14 Annexe 15 Annexe 8 Pages 18, 58 Annexes 14,15 Page 58

IPI - Comment faire mon dossier de validation ?

	Les utilisateurs sont opérationnels, le transfert des nouvelles compétences est validé.	Formation des utilisateurs au logiciel.	Formation des utilisateurs au logiciel Explication du fonctionnement d'une app aux utilisateurs	E(1A)	Page 58 Annexe 14
Réaliser des échanges de données informatisés (EDI). Automatiser des traitements.		Réalisation d'un procédé d'échange de données informatisées. Rétro-documentation de logiciels et de bases de données. Consolidation, agrégation de données. Programmation de l'interface d'échange de données.	Manipulation de fichier .edi, .xml, json Utilisation de fonctions d'agrégation Conception d'une API WEB	E(1A) F(6M) E(1A) F(3M) F(3M)	edi - Page 30 xml - Page 25 json - Annexe 9 Annexe 4 Annexe 9
Programmer des scripts systèmes.		Réalisation d'un environnement de tests. Création, configuration de machines virtuelles. Installation, configuration de serveurs d'applications, Web et base de données. Écriture de scripts systèmes pour adapter l'environnement d'exécution.	Mise en place d'un environnement de tests Création et configuration de machines virtuelles (linux, windows) Installation, configuration serveur web, BDD, serveur d'application Modification de scripts	E(2M) F(3M) B(2A) E(2M) F(3M) B(2A) E(1M)	Page 20 Annexe 16 Page 20 Page 28

IPI - Comment faire mon dossier de validation ?

Bloc de compétences : Adapter l'environnement d'exécution, échanger des données entre logiciels

Compétences ou capacités qui seront évaluées	Critères d'évaluation	Exemples d'activités et tâches	Activités pratiquées	Origine de l'acquisition	Preuves apportées & ref. annexe
Réaliser des échanges de données informatisés (EDI).	Les données sont consolidées.	Réalisation d'un procédé d'échange de données informatisées.	Manipulation de fichier .edi, .xml, json	E(1A) F(6M)	edi - Page 30 xml - Page 25 json - Annexe 9
Automatiser des traitements.	La base de données tierce est accédée. L'interface d'échange de données est opérationnelle.	Rétro-documentation de logiciels et de bases de données. Consolidation, agrégation de données. Programmation de l'interface d'échange de données.	Utilisation de fonctions d'agrégation Conception d'une API WEB	E(1A) F(3M) F(3M)	Annexe 4 Annexe 9
Programmer des scripts systèmes.	L'environnement de tests est opérationnel.	Réalisation d'un environnement de tests. Création, configuration de machines virtuelles. Installation, configuration de serveurs d'applications, Web et base de données. Écriture de scripts systèmes pour adapter l'environnement d'exécution.	Mise en place d'un environnement de tests Création et configuration de machines virtuelles (linux, windows) Installation, configuration de serveur web, BDD, serveur d'application Modification de scripts	E(2M) F(3M) B(2A) E(2M) F(3M) B(2A) E(1M)	Page 20 Annexe 16 Page 20 Page 28

Le tableau ci-dessus peut constituer une annexe de votre mémoire de soutenance.

IPI - Comment faire mon dossier de validation ?

8.7. Annexe 7 – Fiche d'évaluation en entreprise



CERTIFICATION PROFESSIONNELLE EVALUATION ENTREPRISE

STAGIAIRE

Nom : Hilaine
Prénom : Sinha

Certification visée :

ENTREPRISE

Nom de la société : CGI
Tuteur(trice) : Pascal Ollier
Courriel : pascal.ollier@cgi.com

«tuteur_Civilité»,

Afin de parfaire la validation de la certification professionnelle de votre stagiaire, nous vous remercions de remplir exhaustivement cet original et nous le retourner dans les meilleurs délais. Ce document sera examiné par le jury en vue de l'attribution de la certification professionnelle de votre stagiaire.

L'intégration dans l'équipe de travail est-elle satisfaisante ?

Oui Non

Le comportement en situation professionnelle est-il adapté à la culture de l'entreprise ?

Oui Non

Le comportement relationnel est-il adapté au métier visé ?

Oui Non

La capacité de travail fournie correspond-elle au niveau d'un professionnel du métier ?

Oui Non

Le niveau de responsabilités attendu est-il satisfait ?

Oui Non

Des questions sont-elles posées à bon escient ?

Oui Non

Des difficultés en relation avec le niveau de responsabilités sont-elles surmontées ?

Oui Non

Le niveau d'obligation de réserve demandé est-il pris en compte ?

Oui Non

Les activités effectuées concourent-elles à la couverture du champ de compétences ?

Oui Non

Le métier associé à la certification visée est-il compris ?

Oui Non

Le stagiaire est-il apte à occuper la fonction visée même comme débutant ?

Oui Non

Appréciation générale suite au stage en entreprise :

Sinha a su s'intégrer avec satisfaction dans notre milieu professionnel. Nous sommes satisfait de son travail et nous lui avons proposé, en conséquence, un CDI
Fait à Le Havre, le 08/11/2019

Signature du tuteur
et cachet de l'entreprise

Signature du stagiaire

Association ADIP régie
par la loi du 1^{er} juillet 1901
Siret : 409 801 677 00017 Code
APE : 85.59 A
N° organisme : 11 75 27 99 775

Centre de formation
44 bis, quai de Jemmapes 75010
Paris
Siège social
12, rue Alexandre Parodi 75010 Paris

Tel : 01 80 97 35 00
info@ipi-ecoles.com
www.ipi-ecoles.com

GROUPE IGS

© Institut de Polytechnique (2011)

8.8. Annexe 8 – Rédaction de règles de gestion

Dans le cadre d'un projet personnel de création d'une application de planning en ligne pour les guildes de jeux vidéo de type MMO, j'ai rédigé des règles de gestion. Je vous ai mis quelques exemples ci-dessous :

2. Règles de gestion

2.1. Visiteur

Un visiteur est une personne qui n'est pas inscrite sur "GRAPIKI".

Un visiteur qui s'inscrit doit obligatoirement renseigner un pseudo, un e-mail (unique), un mot de passe.

Un visiteur inscrit devient un utilisateur.

Un visiteur n'a accès à aucun planning.

2.2. Le planning

Les différents rôles au sein d'un planning sont :

- **Propriétaire** (il ne peut y en avoir qu'un seul)
- **Modérateur** (plusieurs possibles)
- **Membre** (plusieurs possibles)
- **Ancien membre** (plusieurs possibles)

Propriétaire hérite de modérateur qui hérite de membre qui hérite de l'utilisateur.

Il faut obligatoirement être connecté (utilisateur) à l'application pour voir un planning.

On ne peut rejoindre un planning qu'en indiquant *le path* de celui-ci.

Le path est renseigné par le créateur du planning à la création de celui-ci, il est unique.

Quand on y accède pour la première fois, on ne fait pas partie du planning. Si un utilisateur veut devenir membre d'un planning il doit envoyer une requête. Cela fait, une notification apparaît sur le Dashboard des modérateurs pour pouvoir accepter ou refuser d'attribuer le statut de membre à ce dernier.

L'exclusion est possible également ([voir blocage d'adresse IP et/ou liste noire sur l'id de l'utilisateur](#)).

On peut créer et rejoindre autant de planning qu'on le souhaite.

Par défaut, le pseudonyme qui apparaît est celui renseigné lors de l'inscription mais celui-ci peut être changé par l'utilisateur ou un modérateur/propriétaire. Le changement ne s'opère que pour le



▲ 2.1. Les requêtes

Un utilisateur doit obligatoirement envoyer une requête d'admission à un planning pour espérer le rejoindre.

Une requête peut avoir 4 statuts :

- En cours : l'utilisateur a soumis sa demande et elle n'a pas encore été traitée
- Annulé : l'utilisateur a annulé sa demande

-
- Accepté : un modérateur a accepté sa requête
 - Rejeté : un modérateur n'a pas accepté sa requête

Quand un utilisateur envoie une requête elle est au statut "En cours". Toutes les requêtes dont le statut est "En cours" sont visibles par les modérateurs du planning pour être traitées.

Si un utilisateur annule sa demande, elle passe au statut "annulée" et disparaît de la vue des modérateurs. Il pourra la renvoyer plus tard en changeant le message s'il le souhaite. Ainsi la requête repassera au statut "En cours".

Si la requête est rejetée par un modérateur, celle-ci passe au statut "Rejetée".

Lorsqu'une requête est rejetée, l'utilisateur doit attendre 48h à compter de la date de rejet pour pouvoir la renouveler.

Si l'utilisateur a été blacklisté par un modérateur du planning, il n'aura plus la possibilité d'envoyer une requête et donc de rejoindre le planning.

Si le modérateur accepte la requête d'un utilisateur, celle-ci passe au statut "accepté" et un membre est créé.

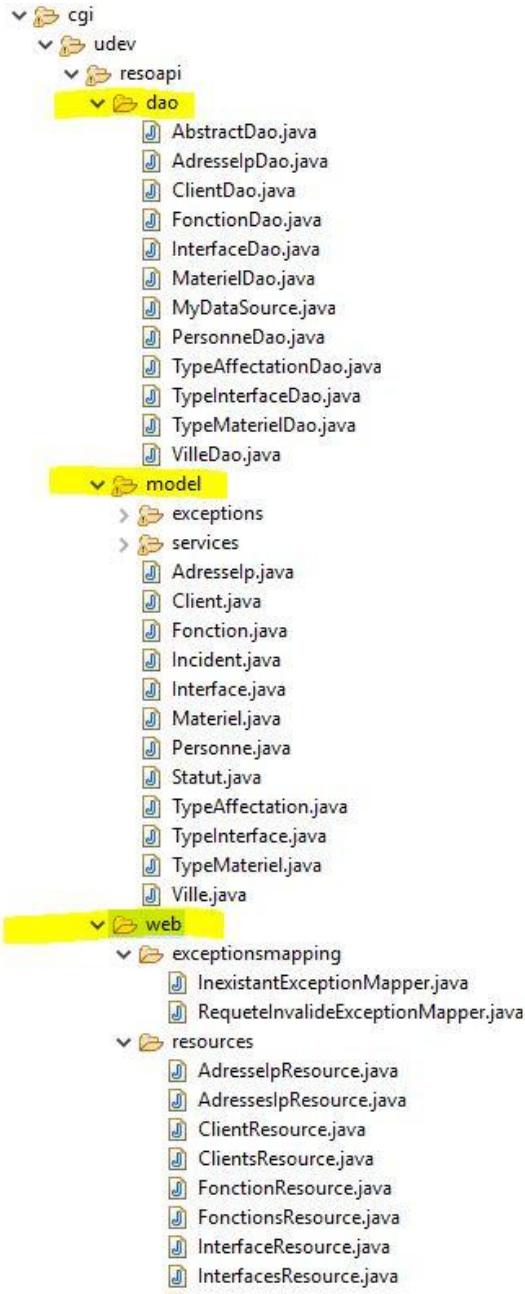


4. Les différents écrans

Rôle	Ecran accessible APP	Ecran accessible planning
Visiteur	<p>Page d'accueil APP</p> <p>Ecran Inscription</p> <p>Ecran Connexion</p>	Pas d'accès (redirection sur l'accueil du site principal)
Utilisateur	<p>Dashboard général (contient notamment : chercher un planning, créer un planning, liste des derniers planning visités, liste des plannings où il est membre, liste des plannings dont il est proprio)</p> <p>>> si on peut imaginer un truc sexy pour ça qui ressemble un peu à discord, <u>les planning</u> sur le côté qui restent même quand on navigue de planning en planning.</p> <p>Ecran modification informations personnelles, paramètres.</p>	<p>Écran d'accueil (éventuellement avec présentation de la guilde, infos publiques générale de la guilde (nombre de membres, date de création...), bouton de demande de droit)</p> <p>Ecran modification pseudo (Pas sûre qu'on fasse un écran, peut-être juste une fonction s'il clique sur son pseudo)</p> <p>Ecran de demande de droit (+modification message si la requête avait été annulée)</p>
Membre		<p>Écran d'accueil membre avec notamment récap des 10 dernières activités sur le planning (nouveau membre, événement créé...), + récap sur <u>les activités</u> sur ses propres événements + récap des</p>

8.9. Annexe 9 – Développement d'une application N-Tiers

Même si je vous ai montré dans la partie 5 de ce dossier que j'ai travaillé sur des applications organisées sur plusieurs couches, je voulais vous ajouter l'organisation de l'API Web pour l'application Android que j'ai faite dans le cadre de ma formation, en cours :



La partie dao est la couche d'accès aux données. Elle contient la classe qui se connecte à la base de données (MyDataSource) ainsi que toutes les classes qui font le lien entre la base de données et les objets Java présents dans la couche Modèle (model)

La partie model est la couche modèle. Elle contient tous les objets java et les méthodes qui permettent d'appliquer les différentes règles métier.

La partie web est la couche web. Elle contient les vues, IHM, les ressources qui font le lien entre la couche web et la couche model.

Les échanges de données entre les différentes couches se font au format json.

L'API Web permet de récupérer et modifier/insérer/supprimer des données en base de données depuis l'application Android et depuis une application Web. C'est l'interface commune aux deux. On pourrait même rajouter d'autres applications pour communiquer avec la base de données.

8.10. Annexe 10 – Réalisation de maquettes

Voici des exemples de maquettes que j'ai réalisées pour mon application de gestion de budget dans le cadre de ma formation à l'Epsi :

The image displays two wireframe mockups of a budget management application interface, showing the 'Budget' screen and the 'Historique' screen.

Budget Screen Mockup:

- Header:** Bienvenue Prénom NOM, Logo App, Profil, Se déconnecter.
- Left Sidebar:** APP Historique, APP Budget - Dépenses, APP Budget - Dépenses, APP Profil, APP Profil - Mon profil.
- Main Content:**
 - Budget Section:** Dépenses, Revenus, Octobre 2018, Budget prévu, Solde actuel, Prévision solde.
 - Category Circles:** Logement, Santé, Achats & Shopping.
 - Information Box:** Afficher les comptes concernés? + petit pop-up ou onglet/fenêtre liée au clic pour réajuster les comptes sélectionnés? (ceux de l'accueil).
 - Pop-up:** Pop-up modif montant budget.
- Bottom:** Facebook, Twitter, Instagram icons.

Historique Screen Mockup:

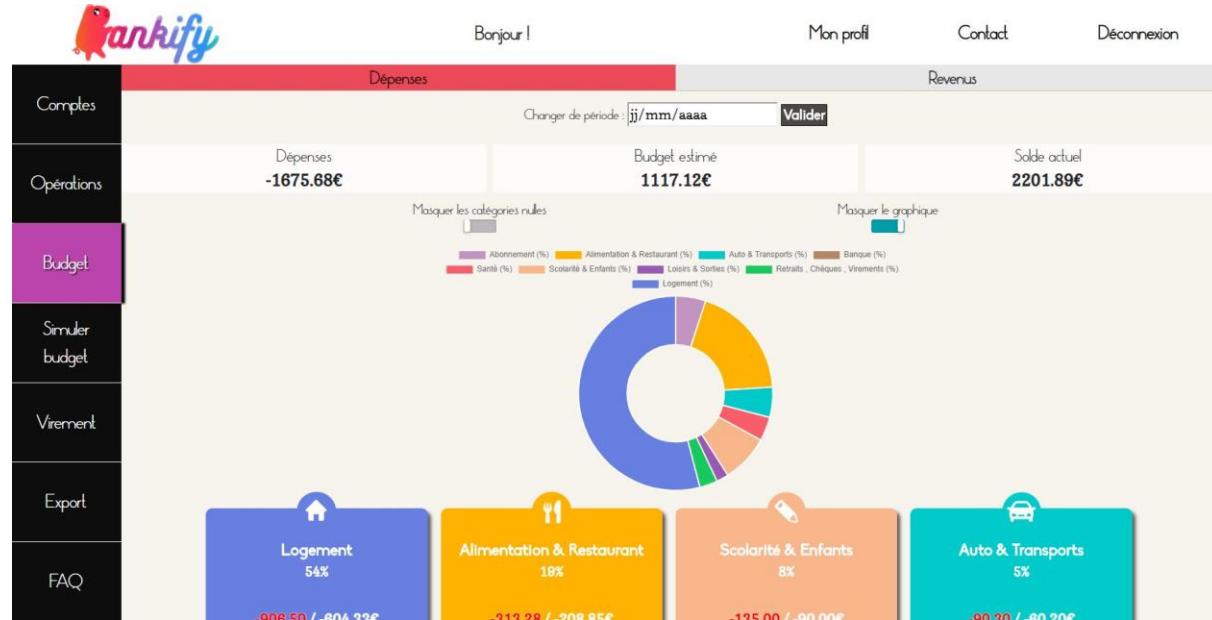
- Header:** Bienvenue Prénom NOM, Logo App, Profil, Se déconnecter.
- Left Sidebar:** APP Accueil - Historique c, APP Accueil - Historique, APP Accueil - Historique, APP Accueil - Historique, APP Accueil - Historique, APP Accueil - Historique.
- Main Content:**
 - Historique Section:** Historique des opérations de NOMCOMpte NUMCOMpte, + Ajouter une entrée, - Ajouter une sortie, rechercher une opération.
 - Operations List:** NomOpération, Montant devise, Date Format : Jour numjour Mois Annee.
 - Buttons:** Rajouter icone catégorie, Aujourd'hui Hier.
- Bottom:** Facebook, Twitter, Instagram icons.

8.11. Annexe 11 – Réalisation d'une IHM

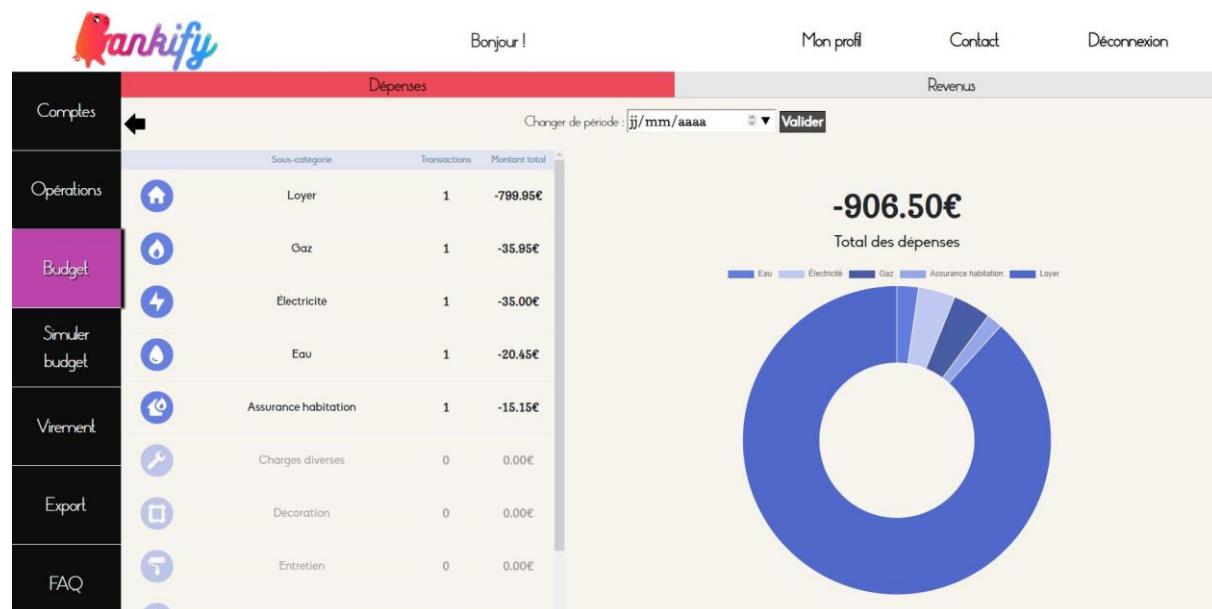
Je vous ai présenté une IHM modifiée dans le cadre professionnel. Je vais également vous montrer des impressions écran d'IHM réalisées pour deux projets de formation cette années.

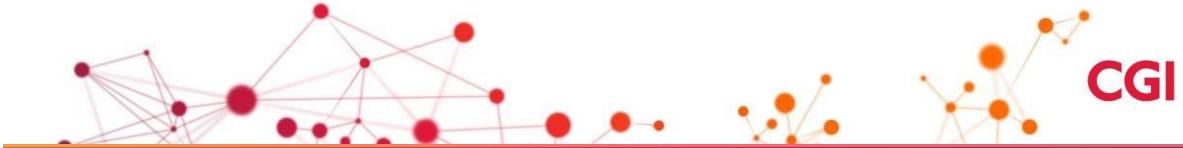
Réalisation d'une application de gestion de budget :

Visualisation des dépenses et regroupement par catégories :



Visualisation des dépenses d'une catégorie regroupées en sous-catégories :





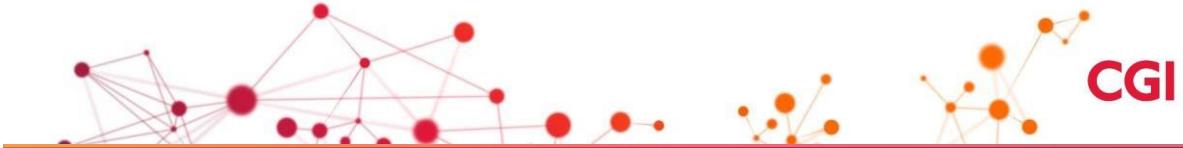
Possibilité d'exporter toutes les transactions par période et par compte :

The screenshot shows the Bankify web interface. On the left, a sidebar menu includes 'Comptes', 'Opérations', 'Budget', 'Simuler budget', 'Virement', 'Export' (which is highlighted in blue), and 'FAQ'. The main content area features a large blue background image of a network of lines and dots. A central modal window has a dark header with the text 'Ici, vous avez la possibilité d'exporter vos opérations au format CSV.' and 'Sélectionnez un compte et une période et on s'occupe du reste !'. It contains two sections: one for selecting a bank account ('Un de mes comptes') with 'Banque Populaire' selected, and another for selecting a period ('Un mois') with fields for 'Du' and 'Au'. At the bottom right of the modal is a blue 'Exporter' button.

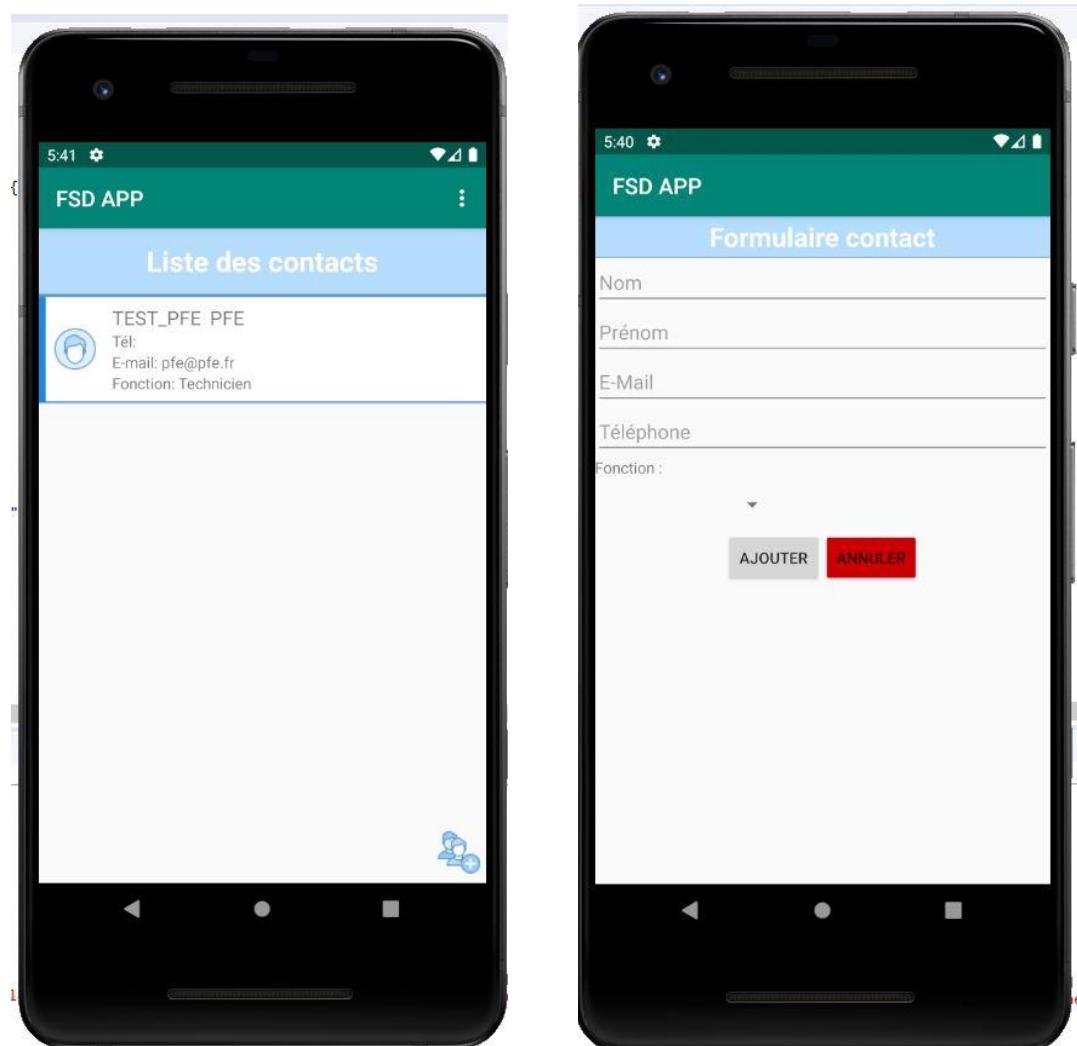
J'ai également réalisé une application Android permettant de gérer les matériels réseau d'un parc informatique. L'apparence n'est pas très jolie car je m'étais beaucoup concentrée sur la fonctionnalité du code.

Visualisation des matériels, possibilité de les trier par type, Ajout/Modification d'un matériel réseau (nom, numéro de série, type, interfaces, adresses ip...) :

The image displays two screenshots of the FSD APP mobile application. The left screenshot shows a list of network equipment under the heading 'Matériels'. The list includes 'Tous', 'Dell 1520', 'NETGEAR DS1008', 'NETGEAR DS1005', 'TP-LINK GS108', 'PASSERELLE LINUX', and 'DEV NICOLAS'. The right screenshot shows a detailed form for managing a specific piece of equipment. The top part is labeled 'Formulaire matériel' and includes fields for 'Libellé' (set to 'PASSERELLE LINUX'), 'N° de série' (set to 'fhh64zeer64fg'), and 'Type' (set to 'Serveur'). Below this are buttons for 'ENREGISTRER' (blue) and 'SUPPRIMER' (red). The bottom part is titled 'Interfaces' and lists 'eth0' with the MAC address '01:02:02:02:02:02'. There is also a 'Type' field set to 'LAN'. The bottom section is titled 'Adresses IP'.



Ajout d'un contact, listing des contacts d'un client :

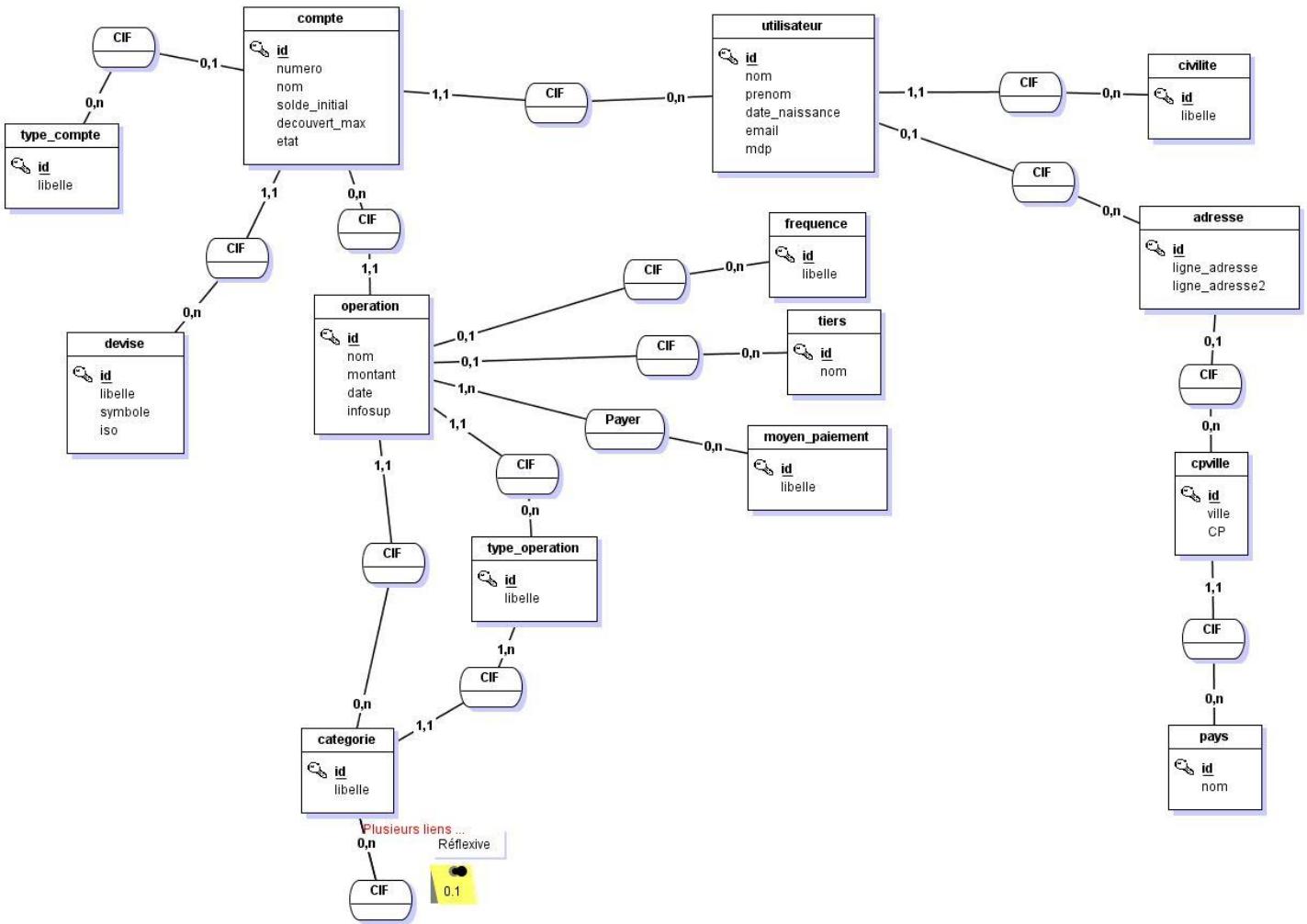


8.12. Annexe 12 – Réalisation de MCD

Lors de ma formation, j'ai pu réaliser plusieurs Modèles Conceptuels de Données (qui ont abouti à un script SQL et à la création d'une base de données).

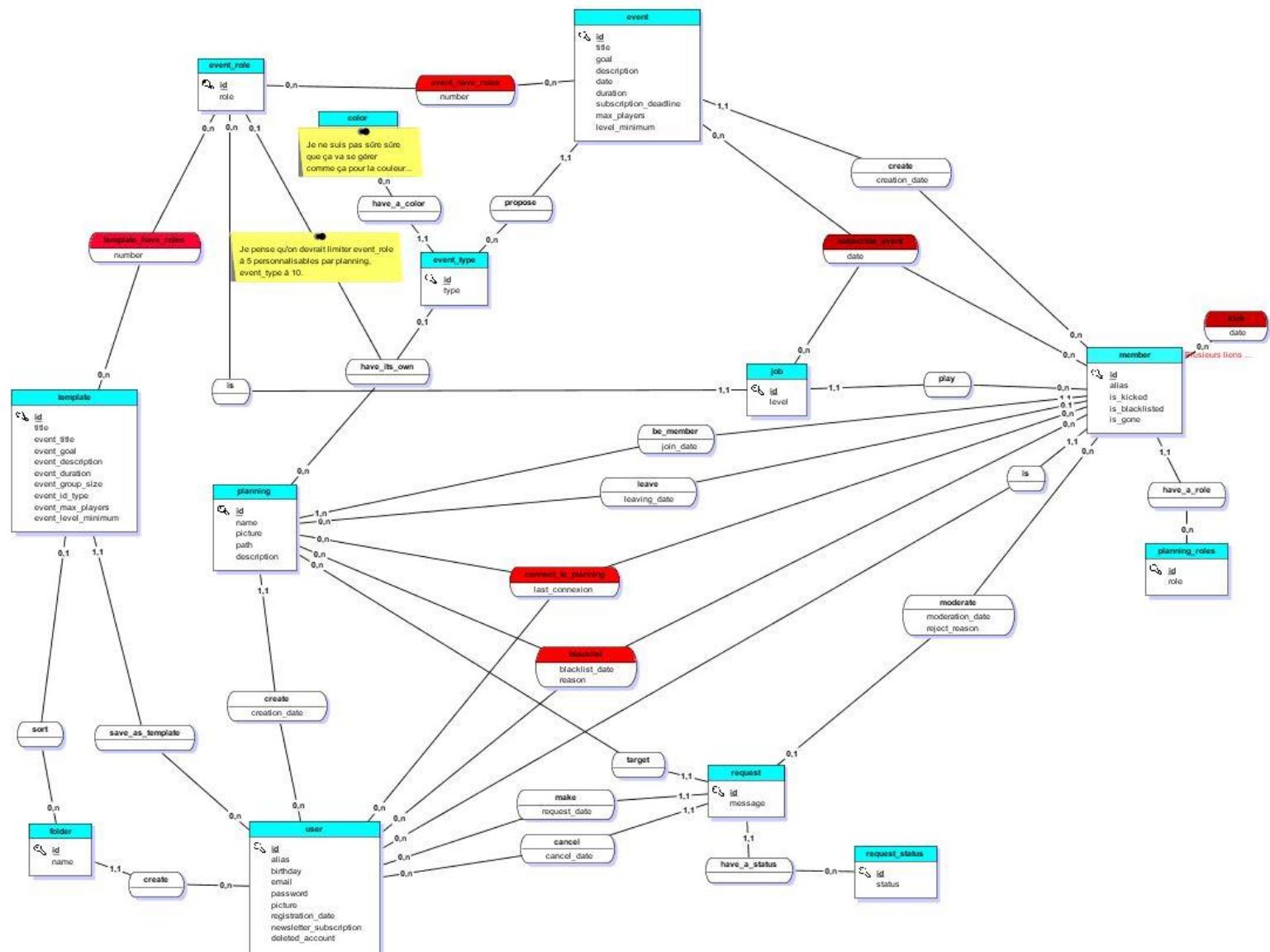
Je vais vous en présenter deux.

Le premier a été réalisé en formation pour mon application de gestion de budget (Bankify) :





Le second a été réalisé pour mon projet personnel (il est toujours susceptible d'être modifié car le projet est en cours) de gestion de planning/événements pour les guildes de jeux vidéo en ligne de type MMO :



8.13. Annexe 13 – Exemple de Javadoc

En entreprise, nous renseignons au maximum la Javadoc de toutes nos méthodes car ça permet de faciliter leur réutilisation et surtout de rendre compréhensible le code par les débutants ou nouveaux arrivants.

```
/**  
 * then set the boolean to false. A message should only be displayed once  
 * @return the _isDataNotAReal  
 */  
public Boolean getIsDataNotAReal() {  
    if(DirectDeliveryDispatchPosController.DISPATCHING_STATE_DATA_NOT_REAL.equals(m_ocurrentState)){  
        m_ocurrentState = DirectDeliveryDispatchPosController.DISPATCHING_STATE_OK;  
        return true;  
    }  
    else  
        return false;  
}  
/**  
 * then set the boolean to false. A message should only be displayed once  
 * @return the Ral was changed  
 */  
public Boolean getWasRalChanged() {  
    if(DirectDeliveryDispatchPosController.DISPATCHING_STATE_RAL_CHANGED.equals(m_ocurrentState)){  
        m_ocurrentState = DirectDeliveryDispatchPosController.DISPATCHING_STATE_OK;  
        return true;  
    }  
    else  
        return false;  
}  
/**  
 * @return the pos was closed  
 */  
public Boolean getIsPosClosed() {  
    return DirectDeliveryDispatchPosController.DISPATCHING_STATE_CLOSED.equals(m_ocurrentState);  
}
```

8.14. Annexe 14 – Exemple procédure utilisateur

Répondre Répondre à tous Transférer

 jeu. 31/10/2019 11:30
HILAIRE, Sirika
 INCT001065704 : [Medium] [REDACTED] - pb doc (Incident AM)
 À [REDACTED]
 Cc FR FNC MANUF [REDACTED] TMAM AM

Bonjour,

Après analyse du problème, vous semblez tenter d'importer le fichier .pdf via la fonctionnalité « Importer à partir d'un fichier ». Une erreur vous indique que le type de fichier n'est pas valide car c'est un fichier .txt qui est attendu.

Afin d'importer votre .pdf, vous pouvez utiliser la fonctionnalité « Ajouter un nouveau document » :

Création d'un groupe de documents

Référence	[REDACTED]
Désignation	[REDACTED]

Liste des documents associés

Référence	Désignation	Source	Original	Traduit	Supp.
					» Ajouter un document existant
					» Ajouter un nouveau document
					» Supprimer documents
					» Importer à partir d'un fichier

» OK

Pourriez-vous me tenir au courant du résultat obtenu ?

Merci d'avance,

Cordialement,

8.15. Annexe 15 – Exemple correspondance utilisateur anglophone

Répondre Répondre à tous Transférer

 jeu. 03/01/2019 10:10
HILAIRE, Sirika
 RE: Rights in [REDACTED] - [REDACTED]
 À [REDACTED] Cc [REDACTED] FR FNC [REDACTED] TMAM AM

Hello [REDACTED]

Concerning your request,
I can advise you to contact your supplier contact to help [REDACTED] to access the necessary document in [REDACTED]

Best Regards,

HILAIRE Sirika
FGDC | CGI
Immeuble Andromède, 6 rue des comètes, 33187 Le Haillan

De : [REDACTED] >
 Envoyé : mercredi 2 janvier 2019 15:57
 À : #F_ Hotline Informatique <hotline.informatique@[REDACTED]>
 Cc : [REDACTED] <[REDACTED]>; [REDACTED] <[REDACTED]>; [REDACTED] <[REDACTED]>; [REDACTED] <[REDACTED]>
 Objet : Rights in [REDACTED]

Hello Hotline,

I am the Administrator for [REDACTED] Company with a Cleveland, Ohio USA address. Our Manager of Manufacturing Assembly has a problem in generating the necessary document in [REDACTED] needed to obtain a DSQR signature. The manager's name is [REDACTED].

[REDACTED] replaced [REDACTED] as Manager of Manufacturing Assembly. [REDACTED] was able to generate the necessary document needed to obtain a DSQR

8.16. Annexe 16 – Machines virtuelles

Dans le cadre de mon travail en entreprise, je n'ai pas eu à manipuler des machines virtuelles. Cependant, j'ai effectué 2 années de formation en alternance en tant que Gestionnaire en Maintenance et Système Informatique au CESI. J'ai donc déjà créé et configuré de multiples machines virtuelles (serveur de stockage linux, serveur Windows avec annuaire LDAP + configuration de réseau, serveur web (apache, tomcat), serveur de base de données (MySQL)).

Lors de mon année de formation à l'EPSI, j'ai créé des machines virtuelles hébergeant un serveur de base de données.

J'ai également, dans le cadre d'un projet personnel, manipulé Docker. Docker ne sert pas à faire des machines virtuelles mais de la virtualisation applicative. J'avais donc stocké dans des containers plusieurs outils : Tomcat, Jenkins, Maven et MySQL. Tout communiquait parfaitement :

Mon serveur MySQL hébergeait ma base de données.

Tomcat était mon serveur web et contenait tout mon code Java.

Maven me permettait build mes images lorsque je voulais mettre à jour mon application. J'utilisais Jenkins (configuré pour utiliser Maven) pour automatiser la création et le déploiement d'images sur mon serveur Tomcat à chaque fois qu'il détectait un commit sur Git (Une vérification était faite toutes les 5 minutes sur mon dépôt).

8.17. Annexe 17 – CV à jour

Concepteur Développeur d'Applications Numériques

SIRIKA
HILAIRE



27 ans

370 Avenue de Tivoli
33110 - Le Bouscat



hilaire.sirika@hotmail.com



06 98 61 81 06

CENTRES D'INTÉRÊT

Cinéma
Musique
Jeux
Cuisine
Animaux

COMPETENCES

Techniques	Qualités
Conception et gestion de base de données	Curiosité
Bases HTML, CSS, PHP, JS, Jquery	Ecoute
Programmation orientée objets (java)	Relationnel
Notions Framework Hibernate	Adaptabilité
Conception d'API Web	Réactivité
Conception d'applications n-tiers	Logique
Architecture MVC	Organisation
Bases NodeJS et Angular	Joie de vivre
Notions scripts batch	Passion
Bureautique : Word, Excel, PowerPoint	
Gestion de projet	
Anglais – Scolaire et technique (lu, écrit, parlé)	

EXPÉRIENCES PROFESSIONNELLES

DEVELOPPEUR JUNIOR – CGI

Décembre 2018 – Décembre 2019 • BORDEAUX

Cycle en V, évaluation du reste à faire, analyse de l'existant, conception d'algorithmes techniques, correction anomalies, problèmes et participation à des évolution (langage Java). Rédaction de tests unitaires et d'intégration. Résolution d'incidents, rattrapage de données en base.

TECHNICIENNE INFORMATIQUE – ORDI DEPANNE

Novembre 2015 – Juillet 2017 • BORDEAUX

A l'agence Docteur Ordinateur de Bordeaux : Conseils, vente, réparation

FORMATIONS

BAC+3 CONCEPTEUR DEVELOPPEUR D'APPLICATIONS NUMÉRIQUES

2018 - 2019 • EPSI BORDEAUX

BAC+2 GESTIONNAIRE EN MAINTENANCE ET SUPPORT INFORMATIQUE

2015 - 2017 • CESI BLANQUEFORT

ETUDES UNIVERSITAIRES – BIOLOGIE MEDICALE

2010 - 2012 • UMONS MONS (BELGIQUE)

BAC LITTERAIRE

2010 • IND CHARLEROI (BELGIQUE)