

EPSI Arras
23-25 Rue du Dépôt
62000 Arras



MiPih Amiens
45 Bd Ambroise Paré
80000 Amiens



Dossier de projet

Concepteur développeur Full stack



ROGER Gabriel

Développement et amélioration d'un ERP destinés à la gestion RH des hôpitaux

2022 – 2023

Table des matières

I.	Remerciements	3
II.	Entreprise	4
A.	Présentation de l'entreprise	4
B.	Vie de l'entreprise	5
III.	Mise en place du projet	6
A.	Project Summary	6
B.	Compétences couvertes par le projet	7
C.	Expressions des besoins	8
D.	Gestion de projets	12
E.	Spécifications fonctionnelles	15
F.	Spécifications techniques	18
IV.	Réalisations du projet	25
A.	Conception d'une fonctionnalité d'ajout et de modification de réponses types	26
a.	Maquettage des écrans	26
b.	Règles de gestion des Réponses types	27
c.	Intégration des écrans et des règles de gestion	28
d.	Création de la table en base et des entités Hibernate	30
B.	Conception d'une fonctionnalité de comparaison des bulletins de paies à l'import	31
a.	Analyse et recherche	31
b.	Mise en place du Batch de comparaison des bulletins de paie	32
c.	Création de la table en base et des entités Hibernate	34
d.	Créations des requêtes HQL de comparaison, clone et mise à jour des bulletins	35
e.	Déploiement du Batch	36
C.	Mis en place d'un super utilisateur sur un services web REST en XML	37
a.	Analyse du besoin	37
b.	Implémentation de la fonctionnalité	37
c.	Test	38
D.	Priorisation des demandes d'un services web	39
a.	Analyse du besoin	39
b.	Implémentation de la fonctionnalité	39
c.	Test	40
V.	Jeu d'essai	40
VI.	Test	41
VII.	Veille sur la sécurité	43
VIII.	Recherche de Solutions	45
IX.	Avenir et évolutions du projet	46
X.	Conclusion	47
XI.	Annexes	48

I. Remerciements

Je tiens à exprimer ma gratitude envers le MiPih et l'équipe EPSI pour m'avoir offert l'occasion de faire mon alternance cette année. Je remercie tous ceux qui ont rendu cette expérience enrichissante.

Un grand merci à M. Wilfried Barreaux, mon manager, pour sa bienveillance, son encadrement et ses précieux conseils. Sa présence attentive et son soutien m'ont permis de m'épanouir au sein de l'entreprise.

Je suis également reconnaissant envers mes collègues de travail, M. Nicolas Nowinski, M. Kevin Fleurit, M. Cyril Delicourt, M. Pierre-Marie Boutteau et M. Mickael Bimbard, pour leur accueil chaleureux et leur soutien inconditionnel. Leur expertise, leur disponibilité et leur collaboration ont été essentiels pour le succès de mes projets et ma croissance professionnelle.

Un remerciement particulier à M. Mickael Bimbard, le Product Owner de notre équipe, pour son implication, ses orientations éclairées et sa confiance en moi. Sa vision et son leadership ont joué un rôle déterminant dans l'atteinte de nos objectifs.

J'exprime ma profonde reconnaissance envers toute l'équipe du MiPih pour leur accueil bienveillant et leur contribution à mon apprentissage. Chaque membre a joué un rôle important en partageant leur expertise et en créant un environnement de travail stimulant.

Je tiens également à remercier l'EPSI et ses enseignants pour leur pédagogie. Leurs enseignements de qualité et leur encadrement attentif ont grandement contribué à ma préparation pour le monde professionnel. Je suis vraiment reconnaissant envers toutes les personnes qui m'ont aidé pendant cette année universitaire.

Finalement, je veux dire un grand merci à l'équipe pédagogique de l'EPSI pour leur encadrement et leur soutien. Leur aide tout au long de cette année a facilité mon insertion dans le monde professionnel.

II. Entreprise

A. Présentation de l'entreprise

Le MiPih est un GIP (Groupement d'intérêt public) spécialisée dans l'informatique médicale. En tant qu'éditeur et intégrateur informatique, il offre des solutions innovantes pour les hôpitaux publics. Fondée à Toulouse, en France, il répond aux besoins qui augmentent pour gérer les systèmes d'information des établissements de santé. L'entreprise se concentre sur l'innovation et le développement de solutions adaptées aux problèmes médicaux.

Avec des locaux à Toulouse, Amiens, Bordeaux et Reims, le MiPih joue un rôle crucial dans le secteur de la santé nationale. Il crée des solutions adaptées à chaque établissement, couvrant différents domaines tel que les dossiers médicaux, le parcours patients, les ressources humaines et financières.

Le MiPih est reconnu pour son professionnalisme et la satisfaction de ses clients. Il comprend les défis de la santé et améliore les soins ainsi que l'efficacité des établissements grâce à des solutions applicatives.

L'entreprise participe à moderniser les infrastructures et les services de santé en aidant les établissements à se numériser. Elle propose des applications optimisées pour améliorer la gestion des établissements de santé.

Le MiPih vise également à améliorer le parcours des patients en développant des bornes d'admission qui facilitent et accélèrent leur prise en charge dès leur arrivée (Annexe N° 1).

Les sites d'Amiens et de Toulouse sont équipés de centres de données (DataCenter) qui permettent de stocker de manière sécurisée les solutions applicatives du MiPih et les données de santé des patients. En effet, le MiPih est certifiée en tant qu'Hébergeur de données de santé (HDS), ce qui prouve sa capacité à stocker et gérer des données de santé hautement confidentielles (Annexe N° 2).

Le MiPih évolue constamment pour s'adapter aux besoins du secteur médical public. A l'aide de son expertise et de son engagement, MiPih contribue à l'amélioration des établissements de santé publics.

B. Vie de l'entreprise

Chez MiPih, je suis développeur dans l'équipe "Prodige-1" chargée de développer l'ERP "Prodige". Nous suivons la méthode Scrum. Nous sommes six développeurs et un Product Owner.

Notre journée commence par une réunion matinale appelée "stand-up" à 9 h 30. Chacun partage ce qu'il a accompli, ses progrès, difficultés et tâches prévues. Cela aide à coordonner nos actions et à favoriser la collaboration.

En tant qu'équipe Scrum, nous travaillons en cycle de trois semaines appelés "sprints". Nous développons les fonctionnalités définies par le Product Owner et nous nous engageons à les livrer à temps. Nous adaptons nos priorités en fonction des retours des utilisateurs.

Nous livrons deux versions majeures de l'application chaque année, avec de nouvelles fonctionnalités et améliorations. Entre ces versions, nous pouvons réaliser des livraisons mineures selon les besoins.

La qualité est importante chez MiPih. Nous testons et validons les fonctionnalités développées. La satisfaction des clients est notre priorité. Nous recueillons sans cesse les besoins de ces derniers pour concevoir une solution au plus proche de leurs attentes.

Nous privilégions la collaboration et l'échange au sein de l'entreprise. La communication et l'entraide sont au centre de notre travail. Nous partageons nos connaissances et notre expérience pour progresser ensemble.

Nos journées sont structurées en suivant différents horaires. A 9 h 30, nous avons une réunion matinale de quinze minutes. Ensuite, nous avons une pause déjeuner entre midi et quatorze heures. Enfin, nous avons la possibilité de terminer notre journée à partir de 16 h 00. Les horaires sont souples, car au MiPih, l'accent est mis sur la qualité du travail plutôt que sur la quantité.

III. Mise en place du projet

A. Project Summary

During my work-study project at MiPih, I was part of the "Prodige-1" team as a developer and application designer. Our focus was on improving the ERP "Prodige" for HR management in hospitals.

My main task was to develop user-friendly screens and robust jobs for hospital managers in three key areas: processing agent requests, registering agents and checking payslips.

For processing agent requests, I implemented features that allow agents to submit various types of requests, such as leave applications or schedule adjustments, through the application.

Regarding the registration of agents, interns, and students, I worked on creating functionalities that capture relevant information like personal details, qualifications, and employment history to maintain accurate records within the ERP.

In terms of checking payslips, I developed modules that enable the verification and management of payslips within the ERP, ensuring accurate calculations and reliable payment processes.

To enhance functionality and improve the user experience, I focused on developing user-friendly screens and addressing any bugs in the application to ensure smooth operation.

The ERP "Prodige" was developed using Java Web Applications for the backend and GWT (Google Web Toolkit) for the frontend. MiPih decided to use their in-house Java framework called "M" for security and maintenance purposes.

The main goal of the ERP "Prodige" was to streamline HR management processes in hospitals, enabling efficient processing of agent requests, accurate registration of personnel, and reliable verification of payslips. The developed screens empowered hospital managers to oversee and manage these critical tasks seamlessly.

B. Compétences couvertes par le projet

Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité
Maquetter une application Développer des composants d'accès aux données Développer la partie front-end d'une interface utilisateur web Développer la partie back-end d'une interface utilisateur web

Concevoir et développer la persistance des données en intégrant les recommandations de sécurité
Mettre en place une base de données

Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité
Concevoir une application Développer des composants métier Construire une application organisée en couches

C. Expressions des besoins

Le MiPih souhaite développer et améliorer l'ERP "Prodige" pour la gestion des ressources humaines dans les hôpitaux. Ce projet vise à optimiser et simplifier le quotidien des gestionnaires et des agents de santé.

L'ERP Prodige regroupe toutes les solutions applicatives du MiPih, que ce soit pour la gestion des agents, des patients ou des finances.

Ce projet s'inscrit dans un processus de digitalisation des hôpitaux et des établissements de santé publics. Il a pour principales fonctions :

- Améliorer les échanges entre les établissements et leurs agents.
- Simplifier les tâches administratives.
- Faciliter le "parcours patient" en facilitant la prise en charge des patients et leur suivi via des solutions informatiques telles que des applications mobiles et des bornes d'admission.
- Centraliser et numériser les informations des agents et des patients.
- Améliorer la gestion des ressources humaines et financières.

Pour cela, le MiPih propose un portail réservé aux agents de santé (DigiHosp RH) et un autre pour les patients (DigiHosp Patient). Ces portails sont des applications web et mobiles qui permettent la collecte d'informations administratives et médicales.

Dans notre cas, nous allons nous concentrer sur le portail DigiHosp RH, qui permet aux agents de santé de gérer leurs informations personnelles et professionnelles.

Le portail agent permet aux établissements de santé de réaliser les actions suivantes :

- Gérer les congés et les absences (demande de congés, de récupération, consultation du solde de congés, etc.).
- Gérer les plannings et les horaires.
- Publier des communications institutionnelles (informations sur l'établissement, actualités, etc.).
- Publier des documents (décrets, etc.).
- Effectuer diverses demandes auprès des services RH (demandes de documents, de congés, etc.).
- Consulter les informations personnelles et professionnelles (situation administrative, situation familiale, etc.).
- Consulter les duplicatas de bulletins de paie.
- Interagir avec les services RH (messagerie, etc.).

Certaines fonctionnalités du portail font appel aux services back-end de Prodigie, tels que les demandes d'agents ou les inscriptions.

Prenons le cas des demandes d'agents sur lesquelles nous avons travaillé pour ce projet. Les demandes d'agents comprennent les demandes de congés, de récupération, de documents, de changement de situation familiale, de changement de situation administrative, de changement de situation professionnelle, etc.

Actuellement, les hôpitaux utilisent des formulaires papier ou des courriels pour la gestion de ces demandes. Ces formulaires et courriels sont remplis par les agents et envoyés aux services RH pour traitement. Ce processus est très long et fastidieux. De plus, les agents ne peuvent pas suivre l'état d'avancement de leurs demandes. La diversité des canaux utilisés pour la gestion des demandes rend la tâche des services RH très difficile, car ils doivent tout centraliser eux-mêmes.

C'est pourquoi le MiPih a décidé de développer une application web et mobile qui permettra aux agents de faire leurs demandes directement depuis leurs smartphones ou un navigateur. Cette application permettra aux agents de suivre l'état d'avancement de leurs demandes et aux services RH de centraliser toutes les demandes.

En plus du portail, qui constitue la partie client de notre projet, nous avons la partie gestionnaire qui est une application web (Prodige) permettant aux services RH de gérer les demandes des agents. Cette application permet aux services RH, sans s'y limiter, de :

- Gérer les demandes des agents (visualiser, accepter, refuser, traiter, etc.).
- Gérer les agents (ajouter, modifier, supprimer, etc.).
- Transmettre des documents.
- Échanger avec les agents (messagerie, etc.).
- Gérer le pilotage des demandes à travers un workflow de validation.

Pour le workflow mentionné précédemment, la solution a été récemment mise en place par le MiPih. Elle permet de gérer les demandes des agents à travers un processus de validation défini par les services RH, pouvant être modifié à tout moment. Ce processus permet de définir les étapes de validation d'une demande et les personnes chargées de valider chaque étape. Le processus de validation peut varier d'une demande à une autre. Par exemple, pour une demande de congé, le processus de validation peut être le suivant :

- L'agent fait sa demande de congé.
- Le chef de service valide la demande.
- Le service RH valide la demande.
- Le directeur valide la demande.
- La demande est traitée.

En raison de sa jeunesse, les workflows nécessitent beaucoup d'ajustements. Mon rôle a été d'accompagner le déploiement de la solution sur un site pilote (prise en compte des retours et demandes du site).

Parallèlement à cela, j'ai également travaillé sur le développement d'un nouvel outil de contrôle de la paie. Cet outil voit le jour pour répondre à l'exigence grandissante des commissaires aux comptes en charge des établissements de santé.

Le souhait du MiPih est de mettre à disposition un outil permettant aux établissements de contrôler la paie de leurs agents de manière optimale. Ce nouvel outil est réalisé en co-construction avec certains établissements de santé. Ce mode de fonctionnement nous permet de développer des fonctionnalités au plus près des besoins fonctionnels des différents utilisateurs. En effet, les participants de ces Co-construction sont les futurs utilisateurs de la solution (Gestionnaire, Directrice des affaires médicales, directrice des ressources humaines, Directeur des systèmes d'informations, ...).

Une fois la fonctionnalité développée, elle est déployée dans un premier temps sur un site pilote pour recueillir l'avis des utilisateurs avant de la mettre sur le marché.

Le MIPIH produit plus de 320 000 bulletins de paie par mois en collaboration avec le moteur de paie PH7. Ce moteur de paie nous permet de générer les bulletins de paie à partir des données de paie fournies par le MiPih. Après chaque calcul de paie quotidien (paie journalière), PH7 produit des bulletins de paie temporaires qu'il nous transmet à l'aide de fichiers XML qui sont ensuite intégrés dans Prodiges depuis un batch géré par un ordonnanceur Jenkins.

Pour contrôler ces bulletins de paie, nous utilisons actuellement Microsoft Accès. Cependant, cet outil n'est pas très flexible et son utilisation n'est pas conviviale. De plus, plusieurs établissements de santé génèrent leurs bulletins de paie quotidiennement. Cela permet de réduire le temps de traitement des bulletins de paie et les risques d'erreurs. Cependant, cette méthode de travail n'est pas compatible avec notre solution actuelle. C'est pourquoi le MiPih a décidé de développer un outil de contrôle de la paie modulable et simple d'utilisation.

Ce dernier remplace l'ancienne méthode de contrôle de la paie. L'objectif de ce projet est de développer un outil permettant aux services RH de contrôler la paie de manière plus efficace et plus rapide. Cet outil permettra également de générer des rapports de contrôle de la paie à destination des commissaires aux comptes.

Grâce à cette nouvelle solution, les utilisateurs peuvent désormais comparer les bulletins de n'importe quel mois entre eux et commenter l'ensemble des contrôles qu'ils effectuent. Tous ces commentaires sont conservés dans Prodiges et feront l'objet d'un compte rendu mensuel qui pourra être transmis au commissaire aux comptes. De plus, Dans un contexte de paie journalière, nous pouvons dorénavant conservés l'état des bulletins anciennement importé et mettre à jour l'état des bulletins présentant des modifications entre les deux imports.

Tout cela dans le but d'informer les gestionnaires de la nécessité de recontrôler ou non les bulletins.

D. Gestion de projets

Mon projet d'entreprise est basé sur les principes de l'agilité. Nous utilisons une méthode itérative de développement appelée "SCRUM". Cette méthode agile nous permet de mieux gérer les temps de développement et les livraisons des différentes fonctionnalités.

Notre objectif principal est de livrer au minimum deux versions de l'application chaque année, une par semestre. Ces versions majeures incluent de nouvelles fonctionnalités, des améliorations et des correctifs importants. Elles sont le fruit de notre travail d'équipe et de la volonté du MiPih à toujours améliorer sa solution.

En plus de ces deux versions majeures, nous livrons aussi des versions mineures de l'application. La livraison ou non de ces versions dépend de l'avancement de nos missions ainsi que de la nécessité d'apporter ou non des correctifs.

Nous gérons nos projets au travers de cycles itératifs appelés "Sprints". Chaque cycle dure 3 jours. Lors de ces Sprints, nous nous concentrons sur le développement de fonctionnalités spécifiques. Ces fonctionnalités sont définies par le Product Owner de notre équipe. Il a pour mission de transformer les besoins fonctionnels des utilisateurs en tâches de développement appelées "User Story". Il doit également choisir les missions à effectuer pour un cycle en fonction de leurs priorités.

Une fois les missions d'un cycle choisies, il nous les explique lors d'une réunion appelée "Sprint Planning". Pendant cette réunion, toutes les US sont expliquées brièvement et les développeurs leur donnent un poids en fonction de leur complexité.

Le poids d'une US est une unité qui nous permet de retranscrire la difficulté de l'US et de déterminer par conséquent le temps de développement nécessaire à sa réalisation.

Dans notre équipe, nous utilisons la suite de Fibonacci pour déterminer le poids d'une US. Ainsi, en suivant ce raisonnement, nos tâches peuvent avoir une complexité de 0.5, 1, 2, 3, 5, 8, 13, 21, etc. Généralement, nos tâches ne durent pas plus de 8 jours consécutifs. Si jamais c'est le cas, alors le Product Owner divise l'US en deux.

Les Sprints Plannings sont un élément crucial de la méthode Scrum. Ils ont lieu au début de chaque cycle.

En plus de cela, nous effectuons une fois tous les deux mois une réunion appelée "Rétrospective". Cette réunion qui est censée se dérouler à chaque fin d'itération nous permet de revenir sur notre ressenti durant le sprint. On réfléchit dans notre coin aux points positifs et négatifs auxquels nous avons dû faire face entre deux rétrospectives. Ensuite, nous mettons en commun ces points. Les retours des membres de l'équipe sont ensuite analysés et nous mettons par écrit une solution possible aux problèmes rencontrés. Pour les points positifs, nous veillons à ne pas régresser entre deux rétrospectives. La liste des actions à effectuer sera examinée lors de notre prochaine réunion pour vérifier si ces points ont été corrigés ou non.

En plus de suivre une méthodologie de développement agile, nous nous aidons d'outils :

- **Rally :**

Nous utilisons Rally pour orchestrer et suivre notre mise en place de la méthode Scrum. Ce dernier nous permet de visualiser nos tâches en temps réel pour un cycle. Ces tâches sont affichées sur un tableau appelé "ScrumBan" (Annexe N° 3). Il nous permet de visualiser de manière globale la liste des tâches ainsi que l'état d'avancement de chacune d'entre elles. Chaque tâche peut être assignée à un membre de l'équipe. Rally nous permet également d'accéder aux détails d'une User Story. Pour chaque US, nous avons différents onglets nous permettant par exemple d'accéder et de valider les cas de tests réalisés par le Product Owner. Ou encore de laisser un commentaire sur l'US. Enfin, il nous permet de suivre l'avancement du sprint en définissant la durée de ce dernier.

- **GitLab :**

Nous utilisons GitLab en tant qu'outil de gestion de version. Le code des solutions applicatives du MiPih y est hébergé (Annexe N° 4). Nous créons des branches correspondant à chaque version et sous-version des applications. Ce dernier nous permet de travailler en parallèle sur le même projet sans conflit. Il nous permet également de visualiser les dernières modifications effectuées, ce qui facilite le travail de relecture du code. Enfin, GitLab nous permet de veiller à l'intégration continue de notre application en l'associant à d'autres outils tels que SonarQube et Jenkins.

- **Discourse :**

Discourse nous sert pour la résolution de problèmes. C'est notre StackOverflow interne (Annexe N° 5). Ce dernier nous permet de poser des questions techniques pour qu'une personne ayant les connaissances sur le sujet puisse y répondre. Son avantage est que chaque personne peut à la fois poser des questions et répondre à celles des autres. Ainsi, nous pouvons partager notre savoir et faciliter la résolution des problèmes rencontrés. De plus, Discourse nous permet d'effectuer des recherches dans les discussions précédentes. Ainsi, les questions ne sont pas posées de manière répétitive. Pour les problèmes les plus courants, cela nous permet de trouver la réponse rapidement en nous appuyant sur les questions qui ont déjà fait l'objet d'une réponse.

- **Teams :**

Nous utilisons Teams en tant que plateforme de communication principale. Il nous permet de planifier et d'organiser des réunions à distance. Sa messagerie instantanée nous permet également de communiquer de manière rapide et efficace avec nos collègues sans avoir à envoyer de mail. Teams nous permet aussi de voir le statut de nos collègues. S'ils sont en communication ou absents, nous savons alors que la personne est susceptible de ne pas répondre. Teams nous permet de rester en contact avec nos collègues peu importe notre localisation.

- **Draw.io :**

Nous utilisons Draw.io pour réaliser des maquettes techniques entre développeurs. Son interface web basée sur le Cloud nous permet de modifier facilement nos maquettes. De plus, il offre la possibilité d'exporter et d'importer notre travail, ce qui facilite le partage de nos maquettes.

En utilisant ces outils de manière complémentaire, nous pouvons améliorer la gestion des projets du MiPih. Chaque outil nous permet ainsi de couvrir la plupart des aspects de la gestion de projets. L'utilisation de ces derniers a un impact direct sur le développement des différentes solutions applicatives proposées par le MiPih.

E. Spécifications fonctionnelles

Dans le cadre de ma mission d'amélioration de l'ERP "Prodige" utilisé dans les hôpitaux pour la gestion des ressources humaines, nous devons prendre en considération plusieurs aspects. L'utilisation de l'ERP est divisée en deux parties principales. Une première partie prend la forme d'un portail web et mobile, permettant aux agents et aux patients de communiquer et de transmettre des informations aux gestionnaires de leurs établissements respectifs. Cette communication est possible grâce aux applications "DigiHosp RH" et "DigiHosp Patients".

D'autre part, l'ERP possède une seconde partie réservée aux gestionnaires des établissements de santé publics. Cette partie se présente sous la forme d'une application web, leur permettant de consulter les informations transmises par les agents et les patients via le portail client, ainsi que de réaliser des contrôles sur ces données.

Concernant l'application patients "DigiHosp Patients", nous n'en parlerons pas, car nous n'avons pas participé à son développement et nous n'avons eu aucune interaction avec celle-ci. En ce qui concerne l'application dédiée aux agents, "DigiHosp RH", nous n'avons pas directement développé de fonctionnalités, mais nous avons néanmoins été amenés à communiquer avec elle lors du développement de la partie réservée aux gestionnaires. C'est pourquoi nous développerons plus en détail la partie dédiée aux gestionnaires, tandis que nous aborderons plus brièvement la partie dédiée aux agents, puisque notre travail sur cette dernière a été minimal.

Interface Utilisateur

L'application "DigiHosp RH" est destinée aux agents des établissements de santé publics affiliés au MiPih. Elle possède une interface utilisateur intuitive, permettant aux agents de gérer leur carrière professionnelle. Les agents peuvent gérer leurs données personnelles, consulter leur emploi du temps ainsi que leurs congés. Ils peuvent également soumettre diverses demandes aux gestionnaires.

Dans l'application "DigiHosp RH", les agents ont accès à un tableau de bord personnalisé où ils peuvent voir différentes informations relatives à leur emploi (Annexe N° 6). Par exemple, ils peuvent consulter l'ensemble de leurs données à caractère personnel auxquelles les

gestionnaires ont accès. En plus de les visualiser, ils peuvent également les modifier, par exemple en demandant une modification de leur adresse ou de leur statut marital.

"DigiHosp RH" permet également aux agents de gérer facilement leurs demandes d'absence. Ils peuvent demander tout type d'absence liée à leur emploi, comme des RTT ou des congés payés, directement depuis l'interface web ou mobile de l'application (Annexe N° 7).

Un des avantages principaux de l'application est la possibilité de suivre l'état de ces demandes. Ainsi, lorsqu'un agent émet une demande via l'application, il est notifié à chaque étape importante de la demande, comme la validation par un supérieur hiérarchique ou l'ajout d'une pièce jointe manquante.

L'application permet enfin aux agents de consulter l'historique de leur carrière professionnelle au sein de l'établissement. Ils ont accès à leurs bulletins de salaire, leurs promotions, leurs postes, etc.

L'objectif principal de "DigiHosp RH" est de simplifier la partie administrative pour les agents des hôpitaux grâce à des écrans interactifs. Elle permet également de collecter des informations et de les transmettre aux gestionnaires de manière ordonnée.

Interface Gestionnaire

Une fois que les agents ont entré leurs données dans le portail client, les gestionnaires peuvent accéder à ces informations et les gérer via une application web dédiée. Cette interface leur permet d'effectuer des contrôles sur les données saisies par les agents. L'application est organisée de manière hiérarchique. Dans l'ordre de grandeur, nous avons :

Ilot -> Onglet -> Écrans -> Sous-onglet -> Écrans imbriqués.

Ainsi, la partie gestion des ressources humaines de l'ERP est organisée en ilots. Chaque ilot correspond à un pan particulier de la gestion des établissements de santé. Dans notre cas, nous avons travaillé sur les modules "DED" pour les demandes d'agents, "AUD" pour le workflow et "CTP" pour le contrôle de la paie. Chacun de ces ilots contient un ou plusieurs

onglets. Ici, pour "AUD", nous avons les onglets "Agents", "Hiérarchie" et "Approbation des demandes" (Annexe N° 8).

Ensuite, dans chacun de ces onglets se trouve un écran principal qui peut à son tour comporter des sous-onglets. Par exemple, dans le cas du module relatif à la gestion de la paie, l'onglet "Contrôle de la paie" contient une dizaine de sous-onglets (Annexe N° 9). Pour l'onglet "Demandes agents", il ne contient aucun sous-onglet au niveau de l'écran principal qui prend la forme d'un tableau de recherche (Annexe N° 10). Cependant, lorsque l'on clique sur l'une des lignes du tableau, on accède à un écran détaillant la demande. Sur cet écran, on peut trouver des sous-onglets (Annexe N° 11).

De cette manière, chaque écran de Prodiges peut abriter d'autres écrans sous forme de pop-up contenant eux-mêmes des sous-onglets.

L'interface de gestion permet aux gestionnaires de gérer efficacement les requêtes ainsi que les données des agents, le tout en centralisant les données. De plus, les données sont bien hiérarchisées, permettant ainsi aux gestionnaires de mieux s'y retrouver. Cette architecture hiérarchique est indispensable à l'efficacité de la solution Prodiges. En effet, celle-ci contient un très grand nombre de modules et d'onglets, puisqu'elle tend à couvrir l'ensemble des besoins en termes de gestion hospitalière.

L'interface permet également aux gestionnaires de rechercher rapidement des informations dans l'ERP via des critères de recherche précis (Annexe N° 12). Ces informations sont ensuite retranscrites à l'écran sous différentes formes, telles que des tableaux. L'un des principaux avantages de la solution est la centralisation des données, y compris les documents. En effet, les agents peuvent transmettre des documents aux gestionnaires. Ces derniers pourront accéder à ces documents directement depuis l'interface de Prodiges.

L'interface web de l'ERP permet également aux gestionnaires de converser avec les agents. Ainsi, si un agent effectue une demande de changement d'adresse, le gestionnaire prenant en charge sa demande peut converser avec l'agent via une messagerie instantanée (Annexe N° 13). Cette connexion directe entre agents et gestionnaires fluidifie les démarches administratives.

Enfin, les gestionnaires ont par ailleurs la possibilité de créer des formulaires personnalisés à destination des agents. Ils peuvent par exemple concevoir un formulaire d'inscription pour les étudiants. Ceci se montre grandement utile puisqu'ils peuvent récupérer directement les données de ces formulaires dans l'interface de gestion.

L'ERP n'est pas limité à ces seules fonctionnalités et comprend un très large éventail de modules, cependant j'ai essayé de décrire ceux pour lesquels l'équipe dont je fais partie a été chargée.

F. Spécifications techniques

Technologies utilisées

Pour développer et améliorer l'ERP Prodiges, nous avons utilisé diverses technologies. Étant donné que Prodiges est conçu pour les hôpitaux et les établissements de santé, le MiPih a choisi des technologies stables et sécurisées. Les données traitées par l'application sont sensibles et nécessitent une protection adéquate. Les solutions du MiPih sont cruciales pour le bon fonctionnement des hôpitaux, donc leur stabilité est primordiale.

Le MiPih a choisi principalement Java et GWT pour développer l'ERP Prodiges. Toutefois, une migration de GWT vers Angular est en cours pour moderniser l'application. Java a été choisi pour sa stabilité et sa sécurité. Ces facteurs sont très importants pour les établissements traitant des informations sensibles.

Pour simplifier et accélérer le développement, MiPih a mis en place un socle technique. Ce socle est essentiel pour le développement des applications de l'ERP Prodiges. Il fournit un ensemble de bibliothèques techniques pour standardiser le code. Grâce à une approche orientée services (SOA), les développeurs ont accès à des services avec des contrats standardisés. Dans le contexte de l'approche orientée services (SOA), les contrats standardisés permettent de définir les obligations et les responsabilités des fournisseurs et des consommateurs de services. Cela améliore la productivité en permettant un couplage souple entre les composants et en abstrayant les fonctionnalités techniques. Nous pouvons nous concentrer sur la logique métier en utilisant des services prêts à l'emploi.

Nous avons également utilisé Maven pour le développement de l'ERP Prodiges. C'est un outil de gestion de projets qui organise les fichiers sources. Son architecture est composée de répertoires comme `src/main/java`, `src/main/resources`, `src/test/java`, `src/test/resources` et `src/main/webapp`. Maven automatise le packaging pour générer l'application finalisée. Il permet par ailleurs l'héritage du POM (Project Object Model) parent pour chaque type de projet. Ainsi, nous pouvons utiliser de méthodes définies par le SOCLE en héritant de leurs POM. Maven est utilisé pour structurer le développement de l'ERP Prodiges avec ces modules correspondants.

Base de données

Pour gérer la base de données de l'ERP Prodiges, nous utilisons l'ORM "Hibernate". Avec Hibernate, nous pouvons manipuler la base de données en utilisant la notation objet de Java. Cela nous offre des avantages en séparant la communication avec la base de données du code applicatif. Ainsi, en cas de changement de serveur de base de données, Hibernate s'occupe de modifier les requêtes en conséquence. Actuellement, nous utilisons Oracle, mais nous prévoyons une migration vers Postgres à l'avenir pour répondre aux besoins du projet.

De plus, pour faciliter la conception du système, nous utilisons MagicDraw, un outil de modélisation. MagicDraw nous permet de créer des diagrammes détaillés pour représenter visuellement les entités, les relations et les flux de données du système. Cela nous aide à comprendre la structure du système et à identifier les dépendances entre les éléments (Annexe N° 14).

L'utilisation de MagicDraw facilite la communication avec l'équipe et les parties prenantes du projet. Les diagrammes créés servent de référence pour discuter des fonctionnalités, des flux de données et des interactions entre les composants, favorisant ainsi la collaboration. Cela nous permet de prendre des décisions éclairées tout au long du développement et de l'amélioration de l'ERP Prodiges.

MagicDraw nous permet également de générer nos entités Hibernate et nos classes Java correspondantes aux diagrammes UML conçus. Il génère également des classes de tests pour chaque composant, ainsi que des interfaces et des locators. Cette utilisation de MagicDraw nous permet d'économiser beaucoup de temps et de réduire les erreurs de rédaction manuscrite.

Architecture métier de l'ERP

L'ERP du MiPih est construit selon une architecture spécifique avec des zones, des quartiers et des ilots (Annexe N° 15).

Les zones sont des découpages généraux qui regroupent plusieurs quartiers. Chaque quartier correspond à un sous-système fonctionnel, comme la gestion des ressources humaines ou la logistique.

Les ilots, quant à eux, sont des regroupements fonctionnels au sein des quartiers. Ils sont représentés par des sigles tels que CTP, DED, IPS, etc. On peut les comparer à des maisons, où chaque maison représente un onglet spécifique, par exemple, pour le traitement des demandes des agents ou les paramétrages.

L'objectif de cette architecture est de permettre une livraison indépendante des ilots, sans interdépendances au niveau de la base de données.

En ce qui concerne le découpage par capacité fonctionnelle, il s'agit de regrouper des fonctions simples et réutilisables essentielles au bon fonctionnement du système d'information. Par exemple, pour le processus de "poser des congés", on peut identifier trois capacités fonctionnelles distinctes :

- Vérifier son solde de congés
- Consulter le planning du service
- Faire une demande de congés

Au sein d'un quartier, les ilots ont la possibilité de communiquer entre eux. Cependant, un EBS (Entité de Base du Système) ne peut pas appeler directement un autre EBS.

Cette architecture spécifique au MiPih permet d'organiser les différents éléments de l'ERP en zones, quartiers et ilots, favorisant ainsi une gestion modulaire et indépendante des fonctionnalités, tout en facilitant la communication entre les ilots au sein d'un même quartier.

Architecture technique de l'ERP

Du point de vue technique, l'ERP Prodige sépare chaque application d'un îlot en plusieurs module : ebs, application, pc, ws, message et batch (Annexe N° 16).

Chaque module au sein de l'architecture d'une application d'un îlot remplit un rôle technique spécifique qui contribue au fonctionnement global du système. Voici une explication technique du rôle de chaque module :

EBS (Service Métier d'Entreprise) : Ce module est responsable de la gestion des services métier de l'entreprise. Il gère les requêtes vers la base de données et contient toutes les entités Hibernate associées à l'ERP Prodige.

- Il s'agit d'un service métier réutilisable.
- Chaque application d'un îlot ne contient qu'un seul module EBS.
- Il offre deux niveaux de visibilité : niveau quartier pour tous les îlots du quartier et niveau SI pour tous les quartiers de Prodige.

EBR :

- Il s'agit d'un EBS spécialisé dans la gestion des référentiels.
- Il est très réutilisable.
- Il peut gérer des référentiels stables ou variables.
- Les règles sont les mêmes que pour l'EBS général.

AS (Service Applicatif) : Le module d'application est chargé de l'implémentation des fonctionnalités spécifiques de l'ERP Prodige pour la partie visuelle. Ce module permet la communication entre l'interface utilisateur et l'EBS.

- Il s'agit d'un service dédié à l'application.
- Il n'est pas réutilisable.
- Il offre une seule visibilité pour l'UI de l'îlot.

PC (Processus Métier ou Workflow) : Le module PC joue un rôle clé dans la communication inter-îlots au sein de l'ERP Prodige. Il assure la gestion des interactions entre les îlots, en veillant à ce que les opérations entre les modules se déroulent de manière ordonnée et fiable.

- Il représente un processus métier ou un workflow.
- Il permet l'appels d'EBS/R.
- Les appels entre PC sont autorisés.

WS (Web Services) : Le module WS est responsable de la gestion des services web au sein de l'ERP Prodige. Il utilise Java Web Services (JWS) pour fournir des fonctionnalités d'intégration et de communication avec d'autres systèmes externes. Il expose des API web sécurisées et facilite l'échange de données avec des applications tierces, en utilisant le protocoles REST.

Message : Le module Message utilise Java Message Service (JMS) pour la gestion des messages et la communication asynchrone entre les composants de l'ERP Prodige. Il utilise des files d'attente JMS pour acheminer les messages entre les modules, permettant ainsi une communication fiable et efficace. Il assure également la gestion des notifications et des événements au sein du système.

Batch : Le module Batch est couramment utilisé pour contrôler l'intégrité et la cohérence des données dans l'ERP Prodige. Il exécute des tâches en lots planifiées ou automatisées, telles que le traitement des fichiers de données, les calculs de masse et les mises à jour de la base de données. Il utilise des mécanismes de batch processing, tels que des scripts ou des jobs planifiés, pour optimiser les performances et automatiser les tâches répétitives. Le lancement des batchs dans l'ERP Prodige est orchestré par Jenkins, un ordonnanceur utilisé dans le domaine du développement logiciel. Son rôle principal est de planifier et d'exécuter de manière automatique et contrôlée les tâches en lots.

Dans le contexte de l'ERP Prodige, l'ordonnanceur Jenkins permet de programmer le lancement des batchs à des moments spécifiques, en fonction des besoins et des contraintes du système. Il offre une interface simple pour configurer les horaires d'exécution des batchs, que ce soit à des intervalles réguliers, à des heures précises ou en réponse à des événements particuliers.

Communication entres les couches

Pour organiser les différentes communications au sein des services de l'ERP, nous utilisons différentes technologies. Les principales technologies sont les suivantes :

- **WS :**

Des services web REST en XML sont utilisés pour réaliser des actions dans une base de données. Par exemple, DigiHosp RH communique avec Prodige en utilisant des services web pour effectuer des actions dans la partie backend. Les WebServices agirh nous permettent de récupérer des informations sur le dossier des agents (adresses, RIB, etc.) au format XML.

- **JMS :**

La messagerie JMS nous permet d'appeler un autre processus de manière asynchrone. Par exemple, les demandes du côté DED appellent le workflow de validation en utilisant une file JMS. Un message est déposé dans une file d'attente qui est consommé par un autre service. Les files JMS nous permettent de gérer l'exécution de nos processus à travers une file d'attente. De plus, nous pouvons exécuter les processus de manière synchrone ou asynchrone.

- **Batch :**

Les batchs sont utilisés pour effectuer des traitements périodiques de manière asynchrone. En général, ces traitements nécessitent un temps de calcul important. De plus, nous pouvons gérer leur exécution en utilisant un ordonnanceur Jenkins.

- **Flux MIRTH :**

Les flux MIRTH nous permettent de déclencher des actions lors du dépôt d'un fichier sur un serveur. Ainsi, pour l'importation des bulletins de paie, le moteur de paie PH7 dépose des fichiers contenant des bulletins de paie au format XML sur un serveur. Ensuite, le flux MIRTH appelle le batch d'importation des bulletins.

Ces technologies nous permettent de gérer les communications inter-modules et quartier. En effet, les normes de développement de l'ERP imposées par le MiPih nous empêchent certaines communications directes. Ainsi, l'EBS d'un module ne peut pas communiquer directement avec l'EBS d'un autre module. Pour pallier cela, on peut utiliser un web service ou encore une file JMS.

Voici un tableau résumant les principales règles de communication de l'ERP :

A	>>>	B	[condition]	[possible interdit]
EBSSI	>>>	EBSQU	[si iii == iii']	v
UI AS PC	>>>	EBSQU	[si qqg == qqg']	v
EBSSI EBSQU	>>>	EBSSI		x
UI AS PC	>>>	PC		v
UI	>>>	AS	[si iii == iii']	v
UI	>>>	UI		v

IV. Réalisations du projet

Durant mon alternance au MiPih, j'ai eu pour mission d'améliorer l'ERP Prodige en lui ajoutant de nouvelles fonctionnalités. Étant donné que l'ERP était déjà développé à mon arrivée, je n'ai pas travaillé sur un seul projet de bout en bout, mais sur plusieurs fonctionnalités. En effet, j'ai dû développer des fonctionnalités qui ont été ajoutées au fur et à mesure à l'ERP. À chaque Sprint, je réalisais deux ou trois User Stories.

Dans ce contexte, je ne suis pas en mesure de décrire la réalisation d'un nouveau projet. Je ne peux pas non plus détailler une grosse fonctionnalité que j'aurais développée, car nos sprints ne font que trois semaines et les tâches réalisées durant ces sprints ne dépassent généralement pas 8 jours.

Pour pallier cela, je vais détailler ici quatre fonctionnalités que j'ai développées dans le cadre de l'amélioration de Prodige. Je vais faire en sorte de choisir des fonctionnalités qui couvrent l'ensemble des compétences du titre CDA.

J'ai donc choisi de parler dans un premier temps de la fonctionnalité d'ajout de réponses types pour les questionnaires dans la partie "demandes agents". Cette dernière couvre plusieurs compétences, puisqu'elle nécessite la mise en place d'une interface utilisateur, d'une table en base de données et d'un développement back-end.

Dans un second temps, j'expliquerai en détail la réalisation d'une fonctionnalité de contrôle de la paie. Cette dernière nécessite le développement de composants d'accès aux données. Elle nécessite aussi la création d'un batch. Cette fonctionnalité intervient sur plusieurs modules et il nous est donc primordial de bien gérer la communication entre ces couches.

Dans un troisième temps, je détaillerai la mise en place d'un super-utilisateur pour un services web existant. Cette mission nécessite des connaissances dans le développement de services web REST.

Enfin, je parlerai de la priorisation des demandes transmises par un services web en fonction de certains critères. Cela nécessite de bonne connaissance en java et l'utilisation de predicat.

A. Conception d'une fonctionnalité d'ajout et de modification de réponses types

Lors du développement de l'ERP Prodiges pour le MiPih, il nous a été demandé par les gestionnaires d'ajouter la possibilité de créer des réponses types sur la partie relative aux demandes agents nommée "DED". Le but est de faciliter le travail des gestionnaires en leur permettant de créer et modifier des réponses prédéfinies pour répondre aux demandes adressées par les agents.

a. Maquettage des écrans

Dans le cadre du développement de nouvelle fonctionnalité pour l'ERP Prodiges, les maquettes jouent un rôle essentiel pour traduire les besoins des gestionnaires en tâches concrètes de développement.

Pour cela, notre Product Owner (PO) se charge de réaliser des maquettes fonctionnelles qui servent de référence pour la conception des écrans à développer. Ces maquettes sont créées en collaboration avec les gestionnaires lors de la rédaction des Users Stories (US) du Sprint. Grâce à l'interface utilisateur simple et claire de Prodiges, le PO est en mesure de créer ces maquettes fonctionnelles sans difficultés.

De notre côté en tant que développeurs, il nous arrive également de réaliser des maquettes techniques, surtout pour les User Stories plus complexes. Pour cela, nous utilisons souvent l'outil en ligne "Draw.io", qui nous permet de créer facilement et rapidement des maquettes tout en facilitant la collaboration avec les autres membres de l'équipe (Annexe N° 17).

Le maquettage de l'application est une étape cruciale dans le processus de développement de l'ERP Prodiges. Il nous permet de visualiser et de valider les fonctionnalités attendues par les gestionnaires, tout en facilitant la communication au sein de l'équipe de développement. Les maquettes fonctionnelles réalisées par le PO et les maquettes techniques réalisées par les développeurs nous aident à aligner nos efforts et à garantir la qualité du produit final.

Dans le cadre de la User Story concernant l'ajout d'un système de réponses types, notre PO a conçu 3 maquettes correspondant aux 3 écrans à développer (Annexe N° 18). Chaque maquette détaille précisément les éléments à développer. En tant que développeurs, notre PO nous donne la possibilité de lui faire des suggestions d'amélioration sur les maquettes.

Enfin, si la maquette présente des éléments complexes à développer qui pourraient dépasser le temps prévu pour la tâche, alors le PO ajuste la maquette en collaboration avec le développeur pour respecter les contraintes techniques.

b. Règles de gestion des Réponses types

Une fois que nous avons terminé les maquettes fonctionnelles, nous étudions les règles d'utilisation des réponses types définies par le PO. Ainsi, nous devons concevoir les écrans d'édition des réponses types en respectant le cheminement suivant :

D'abord, on va trier les réponses par ordre alphabétique pour faciliter la recherche. Chaque réponse pourra être sélectionnée pour obtenir plus d'infos.

Quand on choisira une réponse spécifique, le message correspondant s'affichera dans une zone de texte dédiée. Cette zone de texte pourra être modifiée selon nos besoins. On aura aussi la possibilité de supprimer ou de dupliquer une réponse grâce aux boutons correspondants.

Pour créer une nouvelle réponse, on pourra utiliser une fonctionnalité de pop-up qui facilitera la saisie du nom. Une fois le nom saisi, on enregistrera la réponse en cliquant sur "Ok". La liste des réponses se mettra automatiquement à jour pour afficher la nouvelle réponse. Il est important de noter que la réponse sera enregistrée dans le SIREN de la demande en cours.

Par défaut, les boutons "Enregistrer" et "Annuler" seront désactivés. Mais dès qu'on modifiera le contenu du message dans la zone de texte, ces boutons deviendront accessibles. On pourra donc sauvegarder les modifications ou annuler les changements si nécessaire.

Si on essaie de fermer la fenêtre ou de choisir une autre réponse alors qu'on a modifié le message et que les boutons "Enregistrer" et "Annuler" sont accessibles, une pop-up de confirmation s'affichera pour nous avertir que les modifications seront perdues. On pourra appuyer sur "Ok" pour continuer ou sur "Annuler" pour revenir à l'écran d'édition.

Pour supprimer une réponse, une autre pop-up de confirmation s'affichera pour demander de confirmer notre choix. On pourra choisir entre "Oui" et "Non" pour décider en toute connaissance de cause.

Si on duplique une réponse existante, une nouvelle pop-up apparaîtra pour saisir le nom de la nouvelle réponse. Le nom de la réponse dupliquée sera déjà rempli pour simplifier le processus. Une fois validé, la nouvelle réponse sera enregistrée avec le contenu de la réponse dupliquée.

Enfin, lors de la création ou de la duplication d'une réponse, on vérifiera qu'il n'y a pas déjà une réponse avec le même nom. Si un nom similaire est trouvé, une alerte s'affichera indiquant "Une réponse avec le même nom existe déjà". Seul le bouton "Ok" sera disponible pour continuer.

Il est important de noter que la longueur d'un message ne pourra pas dépasser 1000 caractères, pour gérer les réponses efficacement.

c. Intégration des écrans et des règles de gestion

Une fois les maquettes réalisées et les besoins définis clairement, nous pouvons commencer le développement de la fonctionnalité. Pour cela, nous devons respecter les technologies utilisées sur ce projet : Java 8, Hibernate, Magiw Draw et GWT. Nous allons commencer par nous occuper de la partie graphique en intégrant les écrans à partir des maquettes réalisées précédemment.

Pour la partie graphique, nous utilisons GWT (Google Web Toolkit). C'est un framework qui nous permet de créer des écrans en utilisant des éléments graphiques en Java. Pour faciliter notre travail et respecter l'identité graphique du projet, le SOCLE du MiPih met à notre disposition un ensemble d'éléments graphiques développés par eux-mêmes. Ainsi, nous avons la possibilité de mettre en place des éléments graphiques complexes sans difficulté.

Dans le cadre de l'ajout des réponses types, nous avons utilisé les composants suivants :

- **MClientDataGrid** : un tableau qui nous permet d'afficher les données récupérées en base de manière harmonieuse (Annexe N° 19).
- **MToolBar** : une barre d'outils placée au-dessus du tableau qui nous permet d'effectuer des actions sur les éléments du tableau, comme la suppression ou l'ajout d'un élément (Annexe N° 20).
- **PageView** : la classe qui contient l'ensemble des éléments graphiques.
- **PagePresenter** : la classe qui contient l'ensemble des événements associés aux éléments graphiques du PageView.
- **WorkPanelPresenter** : la fenêtre graphique (écran pop-up) dans laquelle les éléments seront affichés.
- Divers autres composants tels que des zones de texte, des compteurs, etc.

Dans notre cas, l'écran d'ajout, de modification et de suppression des réponses types doit être créé car il n'existe pas encore. Nous avons utilisé un "WorkPanel" dans lequel nous avons placé un "MClientDataGrid" à droite pour visualiser la liste des réponses types et les ajouter ou les supprimer. À gauche du "WorkPanel", nous avons mis en place une zone de texte dans laquelle nous pouvons créer ou modifier le texte d'une réponse type sélectionnée. Juste en dessous de cette zone de texte, nous avons ajouté un compteur de caractères pour limiter le message à 1000 caractères (Annexe N° 21). Pour récupérer le nombre de caractères en temps réel, nous avons mis en place un "KeyUpHandler" qui nous permet d'exécuter des actions à chaque fois qu'une touche du clavier est pressée (Annexe N° 22). Dans notre cas, lorsqu'une touche du clavier est pressée, nous récupérons l'ensemble du texte contenu dans la zone de texte, puis nous comptons le nombre de caractères pour ensuite les afficher à l'écran.

Pour remplir la zone de texte avec le contenu d'une réponse type existante, nous allons chercher en base le contenu de la réponse type aux cliques sur cette dernière, et nous l'injectons dans la zone de texte. Ensuite, il est possible de modifier les réponses types et de les mettre à jour. En cliquant sur le bouton enregistrer, on vérifie que le texte des réponses types est différent de celui en base, puis le cas échéant, nous l'enregistrons (Annexe N° 23).

Enfin, si une réponse type est en cours de modification et que l'on clique sur la croix pour fermer l'écran, nous affichons un message d'avertissement en informant l'utilisateur qu'il perdra ces modifications (Annexe N° 24).

Maintenant que nous avons mis en place l'écran permettant d'ajouter, de modifier et de supprimer des réponses types, il nous faut pouvoir accéder à cet écran. Pour cela, nous avons ajouté une icône de clé anglaise contenant un "ClickHandler" (Annexe N° 25). Ainsi, lorsque l'on clique sur cette icône, l'écran des réponses types s'affiche sous forme de pop-up.

Enfin, il nous reste à ajouter une liste déroulante dans l'onglet "Discussion" déjà existant, contenant l'ensemble des réponses types précédemment créées. Lorsque la réponse type est choisie, son texte remplit la zone de texte permettant d'envoyer des messages aux agents. Cette même liste déroulante est également disponible lorsque qu'un gestionnaire refuse une demande. Dans ce cas, il lui est demandé de saisir un motif de refus et il peut choisir une réponse type existante.

d. Création de la table en base et des entités Hibernate

Nous venons de réaliser la partie graphique de la fonctionnalité demandée, mais nous devons également pouvoir stocker les réponses types dans une base de données pour assurer la persistance des données. Pour cela, nous avons créé une table dans la base de données appelée "MTDE" (Message Type Demande Evolution) qui contient les champs suivants (Annexe N° 26) :

- MTDE_DATE_CREATION (Date de création de la réponse type)
- MTDE_DATE_MAJ (Date de mise à jour de la réponse type)
- MTDE_CONTENU_MESSAGE (Contenu de la réponse type)
- MTDE_NOM_MESSAGE (Nom de la réponse type)
- MTDE_UTI_CREATION (Utilisateur qui a créé la réponse type)
- MTDE_UTI_MAJ (Utilisateur qui a mis à jour la réponse type)
- EJ_CREATION (Numéro de SIREN de l'établissement)

Pour créer cette table, nous avons généré une entité Hibernate à l'aide de l'outil de modélisation "Magic Draw". Cet outil nous permet de définir des entités via une interface graphique, puis de générer le code Java associé à ces entités UML, ainsi que les fichiers de mapping Hibernate au format XML (Annexe N° 27). Une fois les fichiers de mapping

Hibernate configurés en adéquation avec la classe Java, nous générons le script de création de la table à l'aide d'un plugin mis en place par le MiPih appelé Orast. Ce plugin nous permet de générer un script SQL en fonction d'un fichier de mapping Hibernate. Une fois les scripts SQL générés, nous les envoyons à un job Jenkins qui les exécutera sur les bases de données sélectionnées.

Une fois les scripts SQL exécutés sur la bonne base de données, il ne nous reste plus qu'à inclure les dépendances vers les fichiers de mapping Hibernate dans nos fichiers de configuration. Nous pouvons maintenant interroger la base de données en utilisant la classe Java associée à l'entité Hibernate. Nous pouvons également construire des requêtes HSQL compatibles avec plusieurs dialectes de bases de données tels qu'Oracle et Postgres.

B. Conception d'une fonctionnalité de comparaison des bulletins de paies à l'import

Pour faciliter la gestion de la paie aux gestionnaires des hôpitaux, nous avons créé un module appelé "CTP" qui met à disposition un ensemble d'onglets pour comparer et gérer les bulletins de paie. Ce module permet aux gestionnaires de contrôler les bulletins de paies des agents travaillant dans des établissements de santé public.

a. Analyse et recherche

Comme énoncé dans la partie "Expressions des besoins", le MiPih propose une solution de gestion de la paie en collaboration avec le moteur de paie PH7. Actuellement, un batch est utilisé pour récupérer un fichier au format XML contenant un ensemble de bulletins de paie pour un mois donné. Une fois ce fichier récupéré, les données des bulletins sont extraites et stockées dans la base de données de Prodige. Notre mission est de comparer les bulletins nouvellement importés avec ceux déjà présents en base afin d'avertir les gestionnaires en cas de modification ou de suppression d'un bulletin déjà contrôlé entre deux imports.

Avant de commencer le développement de cette fonctionnalité, il est important de noter que le batch d'importation doit parfois traiter plusieurs centaines de milliers de bulletins dans un temps raisonnable. Nous devons donc veiller à ne pas rallonger le temps de traitement.

Dans ce contexte, nous avons réalisé une étude préalable pour déterminer le moment idéal de déclenchement de la comparaison. Nous avons également effectué des tests de

performance en comparant différentes méthodes de comparaison des bulletins. De plus, nous devons réfléchir à la meilleure façon de conserver les bulletins déjà présents en base afin de les comparer avec les nouveaux bulletins importés.

Nous nous sommes réunis, l'Agile Master, le référent technique et moi, pour planifier notre mission en tenant compte des contraintes énoncées précédemment. Voici les décisions que nous avons prises :

- **Stockage des bulletins existants** : L'Agile Master a suggéré d'utiliser des files JMS pour communiquer la liste des bulletins avec des différences aux modules de contrôle de la paie. De cette façon, nous pourrions comparer les bulletins du côté d'IPS et transmettre la liste des bulletins contenant des différences à CTP pour l'envoi des commentaires. Ainsi, nous n'aurions pas besoin de stocker les bulletins dans une table temporaire, car ils seraient comparés directement aux nouveaux bulletins importés. Cette idée n'a pas été adoptée en raison du temps de traitement plus long du batch et des limitations techniques des files JMS. En effet, les files JMS ont une capacité de transmission limitée et dans notre cas, la liste de bulletins contenant des différences peut contenir plusieurs centaines de milliers de lignes.
- **Comparaison des bulletins** : Le référent technique a suggéré d'utiliser des requêtes SQL pour une meilleure performance. De mon côté, j'ai envisagé de comparer les bulletins côté Java en les découpant en lots et en les comparant de manière concurrente sur plusieurs threads, ce qui serait plus efficace. Pour déterminer la meilleure solution, nous avons chacun réalisé un POC de notre côté. Après comparaison, les deux solutions ont montré des temps de traitement similaires. Cependant, ma solution semblait plus complexe à maintenir en raison de l'utilisation de concepts de développement avancés. C'est pourquoi nous avons décidé d'utiliser des requêtes HSQL pour la comparaison des bulletins.
- **Communication** : Nous avons envisagé d'utiliser des files JMS et un batch pour la communication entre les différents modules. Cependant, après une analyse approfondie, nous avons constaté que l'utilisation de files JMS pourrait entraîner un allongement du temps de traitement du batch et rencontrer des limitations techniques liées à la capacité de transmission des files JMS. Par conséquent, nous avons décidé d'opter pour une autre approche de communication plus efficace, afin d'éviter ces problèmes potentiels.

b. Mise en place du Batch de comparaison des bulletins de paie

Afin de dissocier le temps de traitement de la comparaison des bulletins de celui de l'importation, nous avons décidé de déclencher la comparaison à partir d'un job distinct.

Ainsi, le batch d'importation des bulletins de paie utilise le PC de son module pour cloner la table contenant les bulletins actuels. Les bulletins sont ensuite importés sans modification dans cette partie. Ensuite, le Batch de comparaison des bulletins est appelé en lui transmettant la date du mois de paie à comparer et le numéro d'identification unique de l'établissement (SIREN) en tant que paramètres. Une fois le batch de comparaison appelé, son exécution est gérée par l'ordonnanceur Jenkins, et le batch d'importation peut se terminer. Ainsi, le temps de traitement du batch d'importation est peu impacté, car seul le clone de la table est ajouté. La comparaison s'exécute de manière indépendante.

Pour créer le batch de comparaison des bulletins, nous avons utilisé un outil interne appelé Castor (Annexe N° 28). Cet outil nous permet de créer un batch dans le module de notre choix en automatisant les différentes étapes nécessaires à son initialisation. Il suffit de spécifier le module cible et le type de batch.

Dans l'environnement Prodige, il existe deux types de batch :

- Traitement simple (jobs) : exécute un script shell et permet d'appeler des programmes ou des commandes système.
- Traitement complexe (Batch) : déploie un conteneur JBoss pour exécuter une application Java et peut être lancé avec différents paramètres.

Ces batchs peuvent être exécutés dans l'un des trois contextes suivants :

- Coopération : les batchs de type coopération sont exécutés sur le même serveur pour tous les établissements en coopération. Ainsi, si un établissement exécute un tel batch, les autres établissements partageant ce serveur devront attendre la fin de son exécution.
- EJ : les batchs en mode EJ sont gérés établissement par établissement. Si un établissement lance un batch en mode EJ, un autre établissement n'aura pas à attendre la fin de son exécution, car ils sont traités sur deux nœuds différents. Chaque établissement possède son propre nœud (serveur) sur lequel le batch est exécuté.
- Technique : les batchs techniques sont exécutés sur un seul nœud pour tout le monde. Ils servent à réaliser des actions techniques telles que la maintenance ou les corrections. Ils sont généralement exécutés dans un contexte particulier, comme le batch de renommage des logins.

Une fois le batch configuré sur Castor, celui-ci est automatiquement stocké dans un dépôt GitLab. Il ne nous reste plus qu'à récupérer ce dépôt et à finaliser la configuration du batch manuellement. En effet, quelques configurations restent à effectuer. Nous devons ajouter les EAR (Enterprise Application Archive) des projets avec lesquels le batch doit communiquer. Dans notre cas, nous devons ajouter le PC et l'EBS de CTP ainsi que l'EBS d'IPS. Ensuite, nous devons également ajouter les paramètres du batch (datePaie et Siren).

Une fois la configuration terminée, nous appelons le PC de CTP, où l'ensemble des méthodes de comparaison sera appelé depuis le batch.

c. Création de la table en base et des entités Hibernate

Maintenant que le batch est mis en place, nous pouvons nous occuper de la partie base de données. Nous devons créer une table pour sauvegarder les bulletins présents avant l'importation. Pour cela, nous utilisons Magic Draw. Nous définissons une entité UML contenant les champs de la table des bulletins dont nous avons besoin pour la comparaison (Annexe N° 29). Ces champs sont les suivants :

- COBP_MATRICULE_AGENT (Matricule de l'agent)
- COBP_TOTAL_BRUT (Total brut pour l'attestation CERFA)
- COBP_TOTAL_COTIS_SAL (Total cotisations salariales pour l'attestation CERFA)
- COBP_TOTAL_COTIS_PAT (Total cotisations patronales pour l'attestation CERFA)
- COBP_MONTANT_PRECOMPTE (Total précomptes)
- COBP_DATE_PAIE (Mois de la paie)
- EJ_CREATION (Numéro de SIREN de l'établissement)
- COBP_ETAT_CTRL (État du contrôle du bulletin)

Pour créer cette table, nous avons généré une entité Hibernate à l'aide de l'outil de modélisation "Magic Draw". Cet outil nous permet de définir des entités via une interface graphique, puis de générer le code Java associé à ces entités UML, ainsi que les fichiers de mapping Hibernate au format XML (Annexe N° 30). Une fois les fichiers de mapping Hibernate configurés en adéquation avec la classe Java, nous générons le script de création de la table à l'aide d'un plugin mis en place par le MiPih appelé Orast. Ce plugin nous permet de générer un script SQL en fonction d'un fichier de mapping Hibernate. Une fois les scripts SQL générés, nous les envoyons à un job Jenkins qui les exécutera sur les bases de données sélectionnées (Annexe N° 31).

Une fois les scripts SQL exécutés sur la bonne base de données, il ne nous reste plus qu'à inclure les dépendances vers les fichiers de mapping Hibernate dans nos fichiers de configuration. Nous pouvons maintenant interroger la base de données en utilisant la classe Java associée à l'entité Hibernate. Nous pouvons également construire des requêtes HSQL compatibles avec plusieurs dialectes de bases de données tels qu'Oracle et Postgres.

d. Créations des requêtes HQL de comparaison, clone et mise à jour des bulletins.

Comme précédemment mentionné, nous avons choisi de comparer les bulletins de paie en utilisant la puissance du système de gestion de base de données Oracle. Pour ce faire, nous avons utilisé des requêtes HQL.

Premièrement, nous avons conçu une requête pour cloner la table des bulletins. Pour cela, nous avons sélectionné l'ensemble des bulletins de la table d'origine (ENBP) ayant le SIREN de l'établissement et la date du mois de paie importé. Ensuite, nous les avons insérés dans la table clone (COBP).

Une fois les bulletins clonés, les données de la table ENBP ont été écrasées par la nouvelle importation. Ensuite, nous comparons les bulletins en vérifiant l'égalité des champs présents dans la table COBP avec leurs homologues de la table ENBP. Puis, nous récupérons la liste des bulletins comportant des modifications entre les deux imports.

Après quoi, nous mettons à jour l'état des bulletins présentant des différences en les passant à l'état "A recontrôler" dans la base de données. Ensuite, nous récupérons l'état des bulletins de la table clone et nous les attribuons à leurs homologues nouvellement importés.

Une fois toutes ces étapes réalisées, nous générons une liste de commentaires à partir de la liste des bulletins présentant des différences. Nous insérons ces commentaires dans la base de données. Enfin, nous supprimons les données de la table clone (COBP).

Revenons plus en détails sur certaines requêtes :

- Pour le clonage des bulletins, nous utilisons une insertion groupée. La requête insère dans le clone de la table des bulletins les données des bulletins sélectionnés dans cette même requête. Ainsi, les bulletins sont insérés en base de manière dynamique en sélectionnant directement dans la requête les bulletins à cloner de la table d'origine (Annexe N° 32).

- Pour la requête permettant de comparer les bulletins entre les deux tables, nous avons dû prendre en compte les bulletins qui auraient été supprimés entre deux imports. Pour ce faire, nous avons utilisé une jointure externe qui nous permet également de récupérer les bulletins présents dans la table clone mais absents de la table d'origine. Ainsi, les bulletins récupérés qui contiennent un ID null sont ceux qui ont été supprimés entre deux imports. Nous pouvons donc maintenant insérer un commentaire en base pour spécifier cette suppression (Annexe N° 33).
- Pour la mise à jour de l'état des bulletins, nous utilisons simplement la fonction SQL "UPDATE" en spécifiant à la table d'origine de prendre l'état du bulletin de son homologue dans la table clone (Annexe N° 34).
- Pour la requête permettant la suppression des données de la table, nous utilisons la syntaxe SQL "DELETE FROM <Table>" qui nous permet de vider entièrement les données d'une table sans avoir à la supprimer (Annexe N° 35).
- Pour l'insertion des commentaires, nous utilisons un insert pour chaque commentaire en base en utilisant la méthode "Store" sur notre entité Hibernate des commentaires (Annexe N° 36).

e. Déploiement du Batch

On peut déployer le batch sur l'ordonnanceur Jenkins en utilisant la commande shell "p jobs deploy". Cette commande permet de mettre en place le batch sur l'ordonnanceur en lui fournissant les informations nécessaires pour son exécution. Cela comprend le nom du batch, les éventuels paramètres et la planification des exécutions.

Une fois la commande exécutée, le batch est déployé sur l'ordonnanceur Jenkins et peut être programmé pour une exécution périodique. L'ordonnanceur se charge alors d'exécuter ce dernier selon la fréquence spécifiée, en utilisant les ressources disponibles du système.

Il est important de souligner qu'au moment du déploiement, il est essentiel de s'assurer que toutes les dépendances requises par le lot sont disponibles et correctement configurées. Cela permettra le bon fonctionnement du batch lors de son exécution.

Le déploiement du lot sur Jenkins permet de simplifier la gestion des traitements. Nous pouvons programmer l'exécution du lot de manière périodique, mais nous pouvons aussi le lancer manuellement. Pour ce faire, il nous est possible d'utiliser l'interface web de Jenkins (Annexe N° 37). Nous pouvons également appeler l'ordonnanceur depuis notre application grâce aux méthodes mises en place par le SOCLE (Annexe N° 38).

C. Mis en place d'un super utilisateur sur un services web REST en XML

a. Analyse du besoin

Il existe actuellement des super-utilisateurs pour la GTA (Gestion des Temps et Activités) qui communique avec Prodige à l'aide de services web (WS). Cependant, cette notion de super-utilisateurs n'existe pas encore au niveau des services web que nous mettons à disposition. Nous avons donc besoin de gérer un matricule spécifique dans ces service web pour identifier ces utilisateurs. Ce matricule permettra d'identifier les "super-utilisateurs" tels que la DRH ou l'informaticien, qui ont des droits d'accès à tous les plannings et aux demandes de congés de tous les agents. L'objectif est de pouvoir utiliser un mot-clé à la place du matricule dans l'URL du WS pour afficher toutes les demandes de congés, en ignorant certains paramètres et en définissant les balises appropriées.

b. Implémentation de la fonctionnalité

Pour implémenter cette fonctionnalité, nous avons ajouté une méthode appelée "getDemandesCongesATraiter" à la classe correspondante. Cette méthode est annotée avec les rôles autorisés pour accéder au WS. Elle prend en compte différents paramètres tels que le siren, le matricule, les dates de début et de fin, le nom de la demande, le type de personnel, ainsi que les paramètres "asRemplacant" et "asVisualisateur" (Annexe N° 39).

La première étape de l'implémentation consiste à gérer le cas du super-utilisateur. Nous vérifions si le matricule correspond au matricule prédéfini pour les super-utilisateurs. Ensuite, nous définissons les dates de début et de fin en fonction des valeurs passées en paramètre. Nous utilisons également un objet appelé "DemandeDTOCriteria" pour définir les critères de recherche des demandes de congés (Annexe N° 40).

En fonction des différents paramètres, nous appliquons des filtres et effectuons des recherches spécifiques pour obtenir les demandes de congés correspondantes. Nous utilisons des méthodes du SOCLE pour obtenir les agents à partir de leur matricule, pour rechercher les entêtes des demandes de congés dans un autre système appelé DED, et pour filtrer les congés en fonction des dates.

Une fois les demandes de congés obtenues, nous les convertissons en objets appelés "DemandeCongeDTO" et nous effectuons des opérations supplémentaires telles que la suppression du template du processus pour la GTA et la mise à jour des balises.

Enfin, nous renvoyons la réponse appropriée en fonction du résultat de la recherche des demandes de congés.

c. Test

Nous avons réalisé des tests pour cette fonctionnalité. Le test simule l'appel au WS avec le mot-clé "SUPER" et vérifie si les demandes de congés sont correctement renvoyées et traitées.

Dans ce test, nous utilisons des objets mocké pour simuler les résultats des appels aux différentes méthodes utilisées dans la fonctionnalité. Nous vérifions le statut de la réponse, la présence des demandes de congés et d'autres éléments spécifiques.

Nous avons réalisé des tests unitaires et fonctionnels pour vérifier le bon fonctionnement de cette fonctionnalité (Annexe N° 41).

D. Priorisation des demandes d'un services web

a. Analyse du besoin

Il y a au sein de l'ERP un services web qui permet d'exposer une liste de formulaires. En ce moment, si le WS est appelé avec un type de personnel et un matricule spécifique, tous les formulaires correspondants sont renvoyés, même s'ils ont le même code technique. L'objectif de notre mission est d'éviter cela en donnant la priorité à certains formulaires. En particulier, si un formulaire est associé aux populations "Personnel non médical" ou "Personnel médical", il doit être renvoyé en premier. Sinon, si plusieurs formulaires ont le même code technique, le premier formulaire trié selon un critère spécifique sera renvoyé par le WS.

b. Implémentation de la fonctionnalité

Pour implémenter cette fonctionnalité, nous avons ajouté une méthode appelée "filter". Elle prend une liste de formulaires (ParametrageDemandeAgent) et un type de personnel (strTypePersonnel) en paramètres. Cette méthode effectue différentes étapes de filtrage pour obtenir les formulaires prioritaires.

Nous utilisons des prédicats (Predicates) pour définir les conditions de filtrage. Le premier prédicat, "codeNotInList", vérifie si le code du formulaire n'a pas déjà été ajouté à la liste des formulaires trouvés. Le deuxième prédicat, "typePersonnelNotPNMAndPMD", vérifie si le type de personnel du formulaire n'est pas "Personnel non médical" ou "Personnel médical". Le troisième prédicat, "typePersonnelInQuery", vérifie si le type de personnel du formulaire correspond au type de personnel passé en paramètre.

En utilisant ces prédicats, nous parcourons la liste des formulaires et ajoutons les formulaires prioritaires à la liste des formulaires trouvés. Les formulaires sont ajoutés dans l'ordre où ils sont rencontrés.

Enfin, nous trions la liste des formulaires trouvés en utilisant un comparateur (Comparator) basé sur le type de demande (TypeDemande) des formulaires (Annexe N° 42).

La méthode renvoie la liste des formulaires prioritaires filtrés et triés.

c. Test

Nous avons réalisé un cas de test pour cette fonctionnalité. Le test simule l'appel au WS avec un type de personnel (PNM) et un matricule (MATRIC). Nous vérifions si les formulaires renvoyés sont ceux attendus après le filtrage et le tri.

Dans ce test, nous avons configuré des formulaires avec différents types de demandes et types de personnel. Nous vérifions la présence des formulaires prioritaires ("DEMANDE_PNM" et "DEMANDE_PMD_CONTRAT_VALIDE") dans la liste renvoyée, ainsi que l'absence des formulaires non prioritaires ("DEMANDE_PNM_CONTRAT_VALIDE" et "DEMANDE_PMD"). Pour réaliser ces tests, nous avons utilisé le framework de test AssertJ ainsi que Mockito (Annexe N° 43).

V. Jeu d'essai

J'ai réalisé un test complet pour évaluer les fonctionnalités que j'ai développées. J'ai utilisé des données spécifiques et comparé les résultats attendus avec les résultats réels. L'objectif était de vérifier si le système ERP "Prodige" fonctionne correctement dans différents scénarios de gestion de la paie des agents hospitaliers.

Voici le scénario de test que j'ai mis en place :

Fonctionnalité testée : Importation et comparaison des bulletins de paie

Données d'entrée :

- Fichiers XML contenant les bulletins de paie du mois en cours

Résultats attendus :

- Les bulletins de paie importés doivent être clonés pour préserver les données existantes.
- Chaque bulletin de paie importé doit recevoir l'état approprié en fonction de la table clonée.
- Les bulletins de paie des deux tables doivent être comparés. Si un bulletin diffère dans la table clonée, son état doit être mis à jour en "À recontrôler".
- Un commentaire doit être envoyé à l'agent pour les bulletins présentant des différences ou ayant été supprimés.

Résultats obtenus :

- Les bulletins de paie ont été clonés avec succès avant l'importation, préservant ainsi les données existantes.
- Chaque bulletin de paie importé a reçu l'état approprié en fonction de la table clonée.
- Les bulletins de paie des deux tables ont été comparés. Si des différences ont été trouvées ou des bulletins manquants identifiés, leur état a été mis à jour en "À reconstruire".
- Des commentaires ont été envoyés aux agents concernés, leur fournissant les informations nécessaires.

En effectuant différents tests avec des données variées, j'ai confirmé le bon fonctionnement de l'importation et de la comparaison des bulletins de paie. Les résultats obtenus étaient conformes aux attentes, démontrant la fiabilité et l'efficacité de cette fonctionnalité.

Ces tests ont validé l'intégration réussie de cette fonctionnalité dans le système ERP "Prodige". Ils ont également permis d'identifier d'éventuels problèmes ou domaines d'amélioration, contribuant ainsi à l'amélioration continue du système et à la facilitation de la gestion de la paie pour les administrateurs hospitaliers.

VI. Test

Aux MiPih, les tests sont importants dans notre travail. En tant que développeur, je suis responsable de faire des tests unitaires et fonctionnels pour chaque fonctionnalité que je développe.

Pour assurer la qualité du code, nous utilisons JUnit pour les tests unitaires. Chaque fonctionnalité que nous développons est accompagnée d'un test unitaire correspondant. Notre objectif est d'atteindre un taux de couverture de code de 80% pour les nouvelles parties du code, tel que requis par SonarQube. Si nous n'atteignons pas ce seuil, nous ne pouvons pas intégrer notre développement dans l'ERP. Pour les tests unitaires, nous utilisons les bibliothèques assertj et Arquillian. De plus, notre Socle technique a mis en place une surcouche à la bibliothèque Arquillian qu'ils ont nommée "MArquillian".

En complément des tests unitaires, nous réalisons des tests fonctionnels pour chaque User Story, en nous basant sur les cas de test rédigés par le Product Owner (PO). Ces tests nous permettent de vérifier la conformité des nouvelles fonctionnalités aux spécifications avant de considérer notre User Story comme terminée.

Une fois qu'un développeur a achevé sa User Story et que le code a passé avec succès les critères de qualité définis par SonarQube, un autre développeur procède à une revue du code et l'intègre à l'application. Après cela, ce même développeur effectue des tests supplémentaires pour valider les nouvelles fonctionnalités en se basant sur les cas de test prévus par le PO. Lorsque tout est terminé, la User Story est marquée comme "terminée" et le PO procède à une nouvelle série de tests.

Si la revue de code n'est pas satisfaisante ou si les tests fonctionnels ne sont pas concluants, nous faisons un retour au développeur responsable du développement pour qu'il corrige les problèmes identifiés. Ensuite, nous relançons un nouveau cycle de tests pour cette tâche.

Pour faciliter les tests, nous utilisons des jeux de données spécifiques. Ces jeux de données sont adaptés aux différentes situations que nous devons tester, pour couvrir de nombreux scénarios. Cela nous permet de vérifier si les fonctionnalités développées répondent aux besoins de gestion des ressources humaines dans les hôpitaux.

Les tests sont très importants pour notre développement. Ils nous aident à trouver et résoudre les problèmes, et à s'assurer que notre ERP Prodige fonctionne bien et que les utilisateurs soit satisfait. Nous utilisons une approche itérative et collaborative pour tester rigoureusement chaque fonctionnalité avant de l'ajouter au produit final.

VII. Veille sur la sécurité

Pendant le développement de l'ERP Prodige, nous avons accordé une attention particulière à la sécurité. Nous nous sommes informés sur les bonnes pratiques de développement et les vulnérabilités liées aux services, langages et frameworks que nous utilisons. Pour assurer cette surveillance, nous respectons les recommandations de sécurité de l'ANSI. Nous surveillons aussi de près les référentiels OWASP, notamment le Top 10 OWASP. Ce dernier identifie chaque année les dix vulnérabilités de sécurité les plus exploitées (Annexe N° 44). Ce document nous permet de mieux identifier les risques les plus critiques.

Pour nous aider à respecter les bonnes pratiques de sécurité lors de nos développements, nous utilisons également des outils. Dans notre cas, nous utilisons SonarQube pour gérer les bonnes pratiques et la qualité du code. Chaque nouveau développement effectué est scanné. Ainsi, les mauvaises pratiques et les failles de sécurité sont évitées. Cela nous offre une vue d'ensemble sur la qualité du projet. De cette manière, nous pouvons diminuer les failles de sécurité courantes, les éventuels bugs et les mauvaises pratiques (Annexe N° 45).

SonarQube est un outil efficace. Cependant, il n'est pas suffisant à lui seul. Nous avons donc décidé d'utiliser CyberWatch, un outil de surveillance de sécurité qui s'intègre parfaitement à notre cas d'utilisation.

CyberWatch est un outil complet qui détecte les vulnérabilités et les menaces émergentes dans notre système. Il nous fournit des informations à jour sur les nouvelles failles de sécurité, les correctifs disponibles et les recommandations pour maintenir notre ERP à jour et protégé. Grâce à CyberWatch, nous restons constamment informés des dernières menaces et vulnérabilités qui pourraient affecter les services de l'ERP Prodige.

Ce qui rend CyberWatch particulièrement adapté à notre cas d'utilisation, c'est sa capacité à analyser en profondeur notre infrastructure et nos applications. Il identifie les points faibles potentiels et nous propose des recommandations précises pour les renforcer. Ainsi, nous prenons des mesures proactives pour corriger les vulnérabilités avant qu'elles ne soient exploitées.

CyberWatch propose également une interface conviviale et des fonctionnalités de rapport avancées (Annexe N° 46).

Avec CyberWatch, nous pouvons facilement vérifier la sécurité de notre système. Il nous permet de scanner les services installés sur les serveurs du MiPih et de détecter ceux qui sont vulnérables. Il génère des rapports détaillés sur les services vulnérables et leur impact potentiel sur le système.

Son intégration à notre processus de sécurité améliore considérablement notre détection des failles. Ainsi, nous pouvons évaluer et résoudre activement les problèmes de sécurité. Cette approche renforce la protection des solutions du MiPih. Elle garantit la confidentialité et l'intégrité des solutions applicatives proposée.

En tant que développeurs, il nous arrive de trouver des failles de sécurité. Ça m'est arrivé récemment en examinant les requêtes réseau pour se connecter à l'ERP lors d'une session de débogage. J'ai vu que les données d'identification étaient transmises sans être cryptées. Sur notre serveur de dev, les communications ne sont pas sécurisées et on utilise le simple HTTP. Du coup, quelqu'un qui surveille le réseau peut facilement intercepter les identifiants de connexion des utilisateurs de l'ERP. Pire encore, ces identifiants sont ceux de l'Active Directory. Donc si quelqu'un vole les informations d'authentications d'un agent Prodige, il peut se connecter à son ordinateur, sa boîte mail et tout autre service de l'Active Directory.

Dès l'identification de la faille, j'en ai informé le service de sécurité informatique. J'ai travaillé en étroite collaboration avec l'équipe SSI pour régler rapidement cette faille et renforcer la sécurité globale de l'ERP.

L'équipe SSI fait souvent des tests d'intrusions pour trouver d'éventuelles failles de sécurité. Avec ces tests, ils découvrent les vulnérabilités et les points faibles de notre système. Quand ils trouvent une faille dans l'ERP Prodige, ils nous préviennent pour que nous la corrigions.

Nous prenons au sérieux les retours de l'équipe SSI et nous faisons notre possible pour réparer les failles de sécurité qu'ils trouvent. Nous suivons de près leurs recommandations en matière de sécurité, ce qui nous permet de corriger rapidement les vulnérabilités identifiées.

Cette collaboration étroite avec l'équipe SSI nous aide à améliorer en permanence la sécurité de notre ERP Prodige. On sait à quel point c'est important de maintenir un niveau élevé de sécurité pour protéger les données sensibles des hôpitaux et pour que notre système de gestion des ressources humaines soit stable.

VIII. Recherche de Solutions

Pendant mon alternance, j'ai dû comparer les bulletins de paie importés dans l'ERP "Prodige". Le système actuel importe les bulletins quotidiennement, ce qui écrase les bulletins existants du mois en cours. Mon objectif était de trouver une solution efficace pour comparer les nouveaux bulletins importés avec les anciens dans la base de données. Nous avons deux options : comparer les données en Java ou utiliser des requêtes SQL. Nous devons choisir la solution la plus rapide et performante.

Pour mes recherches, j'ai utilisé des mots clés comme "comparaison tables bases de données Java vs SQL" et "requêtes SQL vs Java pour comparer données". J'ai consulté des sites spécialisés comme Stack Overflow, Oracle Documentation et DigitalOcean.

J'ai sélectionné les sites qui répondaient à ma problématique avec des discussions actives, des réponses détaillées et des exemples de code. J'ai lu attentivement les solutions proposées et j'ai comparé les différentes approches.

Pour vérifier la viabilité et les performances des deux options, j'ai réalisé un POC en comparant les bulletins de paie avec Java et en utilisant le multithreading pour une comparaison concurrente sur plusieurs threads. Le référent technique de mon équipe a également réalisé un POC en utilisant des requêtes SQL. J'ai comparé les résultats obtenus avec des jeux de données fictifs.

Après avoir analysé les deux options, j'ai choisi d'utiliser des requêtes HQL plutôt que la comparaison en Java. Les performances étaient similaires (avec une différence de quelques millisecondes), mais le POC en Java était plus complexe à comprendre en raison de notions algorithmiques avancées et de la gestion des ressources et des threads concurrents. De plus, pour de grandes quantités de données, la comparaison avec le moteur de la base de données était plus performante.

IX. Avenir et évolutions du projet

Pour améliorer Prodige, nous envisageons quelques changements techniques. D'abord, nous allons migrer vers Angular, un framework moderne et solide pour les applications web. Cette migration rendra Prodige plus facile à utiliser et plus réactif, offrant une meilleure expérience aux utilisateurs.

Nous prévoyons également de passer à PostgreSQL pour la base de données de Prodige. PostgreSQL est un système fiable, performant et compatible avec les normes SQL. Cette migration améliorera la stabilité et la gestion des données, répondant aux besoins des hôpitaux qui utilisent Prodige.

En ce qui concerne les outils de modélisation, nous avons décidé de remplacer Magic Draw par des solutions plus adaptées et modernes. Magic Draw, bien qu'efficace, présente des limitations en termes de collaboration et de compatibilité avec les nouvelles technologies. Les nouveaux outils de modélisation permettront une meilleure visualisation, une documentation plus claire et une collaboration facile entre les parties impliquées.

À l'avenir, nous améliorerons Prodige en ajoutant de nouvelles fonctionnalités pour aider les hôpitaux à gérer leur personnel. Nous automatiserons les processus, gérerons les plannings et les compétences du personnel, et intégrerons Prodige avec d'autres systèmes d'information hospitaliers.

En parallèle, le projet DigiHosp, qui vise à numériser les services hospitaliers, sera étroitement lié à l'évolution de Prodige. DigiHosp permettra une gestion numérique complète des dossiers médicaux, des rendez-vous et des prescriptions, offrant une interface patient intuitive et accessible. La collaboration entre Prodige et DigiHosp créera un écosystème hospitalier numérique cohérent et performant, améliorant l'efficacité des processus hospitaliers.

Le futur de Prodige au sein du MiPih est prometteur. Grâce aux changements techniques tels que la migration vers Angular et PostgreSQL, ainsi qu'à l'utilisation d'outils de modélisation plus adaptés, Prodige pourra répondre aux besoins croissants de la gestion des ressources humaines dans les hôpitaux. La collaboration avec DigiHosp ouvrira de nouvelles perspectives dans la numérisation des services hospitaliers, offrant un écosystème performant répondant aux différents besoins des établissements de santé.

X. Conclusion

Pendant mon travail au MiPih, j'ai contribué au développement de l'ERP Prodige pour la gestion des ressources humaines dans les hôpitaux. Cela m'a apporté une expérience précieuse sur le plan académique et professionnel, en appliquant mes connaissances dans un environnement réel.

Mon rôle était de développer de nouvelles fonctionnalités pour Prodige en collaboration avec l'équipe. J'ai travaillé avec des développeurs et des experts métier pour comprendre les besoins et proposer des solutions adaptées.

J'ai acquis une expérience dans la conception et le développement de logicielle. De même, j'ai été formé au rudiment de la gestion de projets. J'ai utilisé des technologies comme Java et GWT, couramment utilisées pour les applications web. J'ai aussi compris les défis de la gestion des données sensibles en milieu médical.

J'ai appris à communiquer et à travailler en équipe. Travailler avec des équipes pluridisciplinaires m'a montré l'importance de collaborer et de coordonner nos efforts pour réussir un projet complexe comme l'amélioration de Prodige.

Je suis reconnaissant envers MiPih de m'avoir donné cette opportunité en tant qu'étudiant en informatique. Cette expérience a donné un sens concret à mes connaissances théoriques.

Je suis convaincu que mon travail contribuera à l'amélioration continue de Prodige et à la gestion des ressources humaines dans les hôpitaux. Je remercie l'équipe pour son soutien et ses enseignements précieux.

En conclusion, mon travail au MiPih a été stimulant et formateur. J'ai pu mettre en pratique mes compétences techniques et en acquérir de nouvelles tout en améliorant la gestion des ressources humaines dans les hôpitaux. Je suis reconnaissant envers MiPih et l'EPSI pour cette opportunité.

XI. Annexes

Annexe 01 : Bornes admissions patients

Annexe 02 : Datacenter du MiPih

Annexe 03 : ScrumBan Rally

Annexe 04 : GitLab

Annexe 05 : Discourse

Annexe 06 : Tableau de bord DigiHosp RH

Annexe 07 : Demande d'absences DigiHosp RH

Annexe 08 : Onglets AUD

Annexe 09 : Sous-onglets écran contrôle de la paie

Annexe 10 : Ecran principale onglet "Demandes agents"

Annexe 11 : Sous-onglets détails d'un agent

Annexe 12 : Critères de recherches

Annexe 13 : Ecran de discussion entre gestionnaire et agents pour une demande

Annexe 14 : Diagramme Magic Draw

Annexe 15 : Schéma architecture Zone, Quartier, Ilots

Annexe 16 : Modules d'un ilot (GitLab)

Annexe 17 : Maquette technique comparaison de la paie

Annexe 18 : Maquettes fonctionnelles réponses types

Annexe 19 : Composants graphiques MClientDataGrid

Annexe 20 : Composants graphiques MToolBar

Annexe 21 : Zone de texte édition réponses types

Annexe 22 : Code handler compteur de caractères

Annexe 23 : Code enregistrement, modification et suppression réponses types

Annexe 24 : Message avertissement perte des modifications

Annexe 25 : Code handler clique affichage écrans des réponses types

Annexe 26 : Table de données réponses types

Annexe 27 : Mapping Hibernate de la table réponses types et class Java

Annexe 28 : Création du batch de différences des bulletins avec Castor

Annexe 29 : Entité Magic Draw de la table de clone des bulletins

Annexe 30 : Mapping hibernate de la table de clone des bulletins et class Java

Annexe 31 : Table de clone des bulletins

Annexe 32 : Requête HSQL de clone des bulletins

Annexe 33 : Requête HSQL de comparaison des bulletins

Annexe 34 : Requête HSQL de mise à jour de l'état des bulletins

Annexe 35 : Requête HSQL de suppression des données la table clone des bulletins

Annexe 36 : Enregistrement des commentaires en bases avec la méthode store de Hibernate

Annexe 37 : Lancement du batch de comparaison des bulletins depuis Jenkins

Annexe 38 : Lancement du batch de comparaison des bulletins depuis le code Java

Annexe 39 : Code méthode getDemandesCongesATraiter pour le WS des congés

Annexe 40 : Critères de recherches des demandes de congés

Annexe 41 : Test unitaire du ws des congés

Annexe 42 : Code méthode filter du ws des formulaires

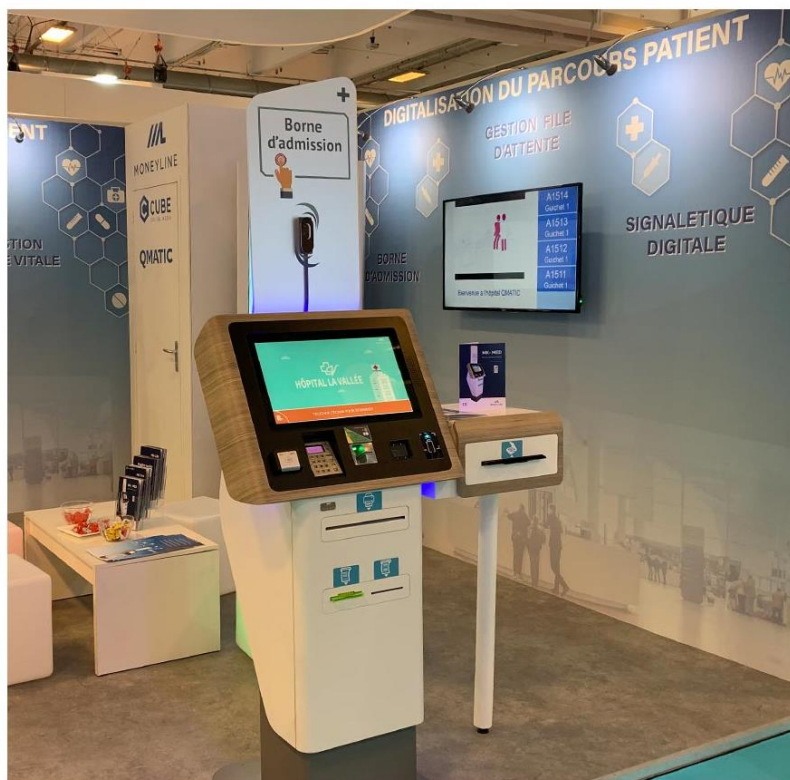
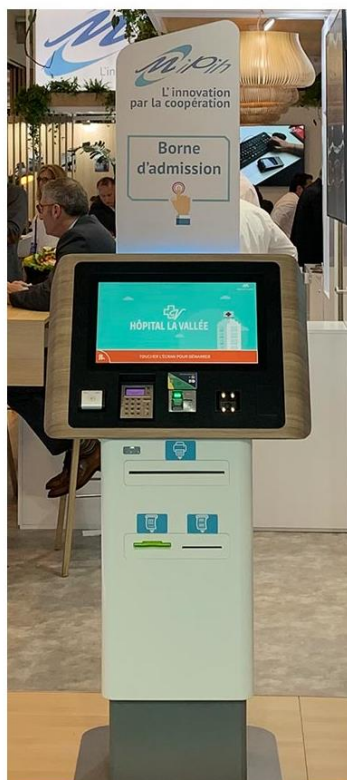
Annexe 43 : Test unitaire du ws des formulaires

Annexe 44 : OWASP Top 10

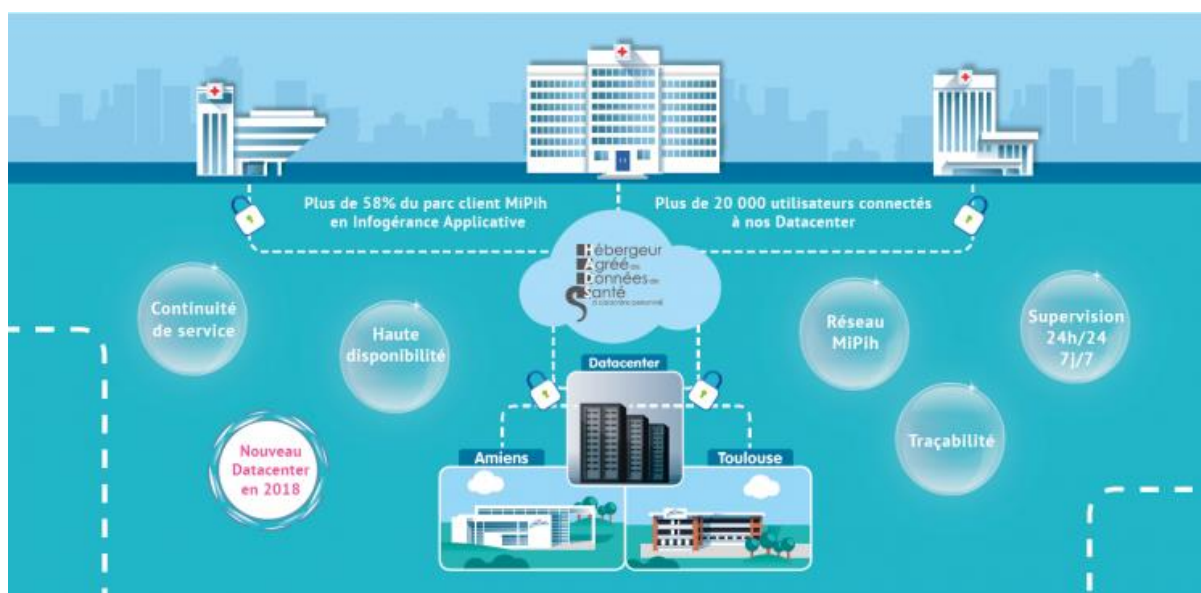
Annexe 45 : Analyse SonarQube d'un nouveau code

Annexe 46 : Interface utilisateur de CyberWatch

Annexe 01 : Bornes admissions patients



Annexe 02 : Datacenter du MiPih



Annexe 03 : ScrumBan Rally

The screenshot displays the ScrumBan Rally interface, a Kanban board for project management. The board is organized into columns representing workflow stages: None, En Cours, En Attente Revue de code, En Cours Revue de code, En Attente TA Dev, En Cours TA Dev, Fini, En Cours Acceptation, and Accepté. Each column contains task cards with details such as task ID, title, assignee, and progress. The interface includes a top navigation bar with 'Rally' logo, user profile, and search bar. The left sidebar shows 'Add Filter' and 'Add New' options. The right sidebar shows 'Show Agreements' and 'Advanced Filters'.

Annexe 04 : GitLab

The screenshot shows the GitLab Projects page. The left sidebar contains navigation links for 'Your work', 'Projects', 'Groups', 'Issues', 'Merge requests', 'To-Do List', 'Milestones', 'Snippets', and 'Activity'. The main content area displays a list of projects under the 'Projects' heading. Each project entry includes a profile picture, project name, description, and last update date. The projects are listed in a table-like format with columns for project details and actions.

Annexe 05 : Discourse

The screenshot shows the Discourse forum interface. At the top, there's a search bar and a user profile icon. Below the header, there are tabs for 'toutes les catégories', 'toutes les étiquettes', 'Catégories', 'Récents', 'Nouveaux (17)', and 'Top'. A '+ Créer un sujet' button is also present. The main content area is divided into two columns: 'Catégorie' and 'Sujets'. The 'Catégorie' column lists various categories with their respective counts and descriptions. The 'Sujets' column shows a list of recent posts with their titles, authors, and timestamps.

Catégorie	Sujets	Récents
Aide et Fonctionnement Si vous êtes perdu, ou que vous cherchez à faire quelque chose c'est ici !	8	Problème avec la version du repo scripts-bdd Forge harmony 0 1 min
Release Suivis des packaging socle - socle graphique - prodige Partage d'information de livraison	2/mois	Version Release release 20 12 min
Design System Questions et demandes pour la CCE et aux garants du design system	6/mois 1 nouveau	Erreur script sql tablespace inexistant Exploitation Interne socle plateforme tablespace 11 19 min
Socle Graphique Angular Socle Angular - NGRX - Génération des stubs	25/mois 2 nouveaux	Plateforme d'exploitation interne down très régulièrement Exploitation Interne aps 12 2 h
Socle Socle technique - Citron - GWT - PostgreSQL - MagicDraw - S2E Migration Oracle PostgreSQL 1 nouveau Quarkus	21/mois 2 nouveaux	Packaging interminable Forge release 3 2 h
Fondamentaux Technique Echange- GAD GDF GCO - Ordonnanceur / traitements métiers - ParamétrageSI - Métamodèle- Sécurité - Arche - Keywi - Adresse - Banque	22/mois 3 nouveaux	Packaging en erreur Forge 5 17 h
Fondamentaux Fonctionnel Nomenclature - Personne Physique - Structures	5/mois	OutOfMemory generation par Magic Draw Socle socle magicdraw 3 17 h

Annexe 06 : Tableau de bord DigiHosp RH

The screenshot shows the DigiHosp RH dashboard. At the top, there's a header with the 'digi hosp RH' logo, a user profile picture, and a 'Contact' button. Below the header, there's a navigation menu on the left with links to 'Accueil', 'Mes demandes', 'Mes Liens', 'Mes bulles RH', 'Espace documentaire', 'Mon Dossier RH', 'Mon compte', 'FAQ', 'CGU', and 'Mentions légales'. The main content area is titled 'Bonjour Gabriel ROGER' and includes a 'Rafraîchir les données' button. Below this, there's a section for 'Mes Bulles RH' with a notification about 'Astreintes du 14 novembre au 21 novembre' published on 14/11/2022. A 'Voir mes Bulles RH' button is also present. The 'Mes actions' section at the bottom features six icons representing different functionalities: 'Mes demandes', 'Mes Liens', 'Mes bulles RH', 'Espace documentaire', 'Mon Dossier RH', and 'Mon compte'.

Annexe 07 : Demande d'absences DigiHosp RH

digihosp RH
Une solution développée par le MIPH

MIPH
Contact

Accueil
Mes demandes
Mes Liens
Mes bulles RH
Espace documentaire
Mon Dossier RH
Mon compte
FAQ CGU Mentions légales
Se déconnecter

Mes demandes

Ma nouvelle demande

Selectionner une demande

Congés non rémunérés

Premier jour de congés -
12/12/2023

Dernier jour de congés -
12/12/2023

Motif -
Titre CDA

Commentaire
Passage de la soutenance du titre CDA

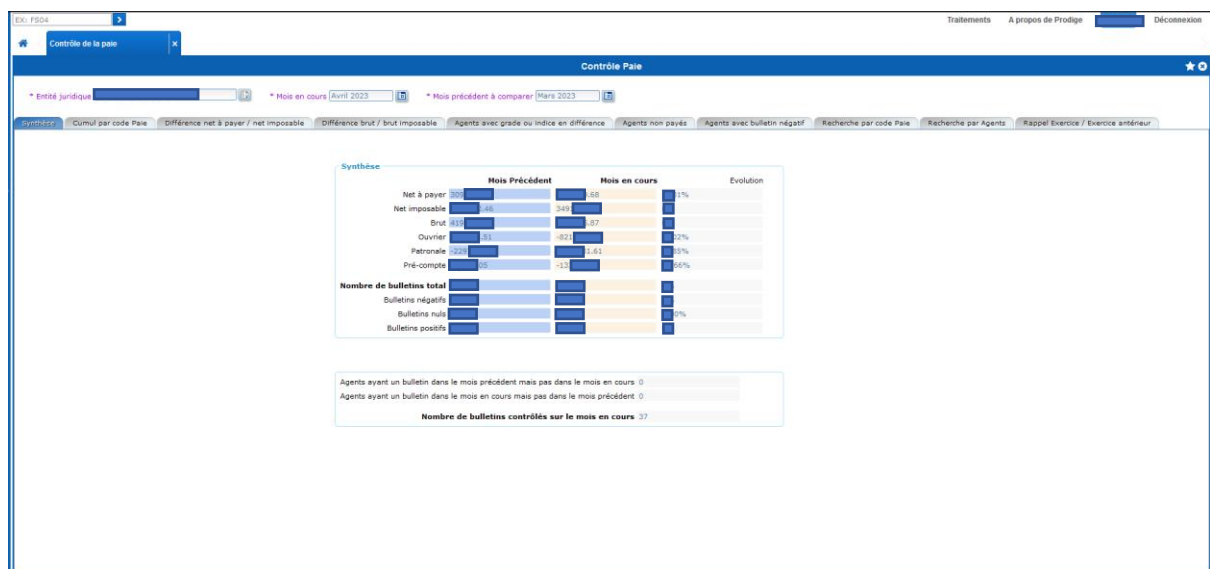
Retour Annuler Envoyer

FAQ CGU Mentions légales

Annexe 08 : Onglets AUD



Annexe 09 : Sous-onglets écran contrôle de la paie



Annexe 10 : Ecran principale onglet "Demandes agents"

EX: P804 2

Traitements A propos de Prodigie Déconnexion

Traitements des demandes agents

Liste des demandes Agents

Critères

Entité juridique

Etat

Date demande du 14 au 25

En cours par ROGER Gabriel

Domaine d'activité

Type

Matricule

Grade

Nom

Statut

Prénom

UF

Pôle

Avec notification

Effacer les critères Rechercher

Résultat

Résumé des critères : Etat : En attente , En cours En cours par ROGER Gabriel

Total : 13 Sé : 0

13 us Restrictions d'accès Agrh Mes demandes en cours : 1 Mes demandes en attente : 50

Etat	Maj Agrh	Date demande	Type	Nom	Prénom	Matricule	Date limite	Etablissement	En cours par	Dernière mise à jour le
		04/05/2023	Demande de changement de coordonnées bancaires				14/05/2023			
		16/03/2023	Demande de changement de coordonnées bancaires				26/03/2023			
		22/03/2023	Demande de document				28/03/2023			
		16/06/2023	Changement d'adresse				26/06/2023			
		10/05/2023	Test champ multiligne							
		09/06/2023	Demande de document							
		16/05/2023	Déclaration de naissance ou d'adoption							
		16/05/2023	Demande de document							
		15/05/2023	demande de document							
		04/04/2023	demande de document							
		04/04/2023	Demande de document							
		29/03/2023	Demande de temps partiel sur autorisation							
		29/03/2023	Demande de temps partiel sur autorisation							

Annexe 11 : Sous-onglets détails d'un agent

EX: P804 2

Traitements A propos de Prodigie Déconnexion

Traitements des demandes agents

Liste des demandes Agents

Critères

Entité juridique

Etat

Date demande du 14 au 25

En cours par ROGER Gabriel

Domaine d'activité

Type

Matricule

Grade

Nom

Statut

Prénom

UF

Pôle

Avec notification

Effacer les critères Rechercher

Résultat

Résumé des critères : Etat : En attente , En cours En cours par ROGER Gabriel

Total : 13 Sé : 1

13 us Restrictions d'accès Agrh Mes demandes en cours : 1 Mes demandes en attente : 50

Demande de changement de coordonnées bancaires

Etat : En attente Prendre en charge

Contenu

Prise-jointe Discussions Suivi interne

Date de la demande 04/05/2023 Date limite de traitement 14/05/2023

Export en PDF

Changement demandé

* Date d'effet 01/07/2021

* IBAN

* BIC

Libellé Banque CHICAM BRIE PICARDIE

* Titulaire NAI

Situation actuelle

* Date d'effet 10/10/2020

* IBAN

* BIC

Libellé Banque Banque inconnue

* Titulaire VANUYDOORV VOFPOOL

Information sur l'agent

Identité

Etablissement

Matricule

Nom / Prénom

Point de gestion

Situation à la date du jour

UP 2144 - E.H.U.R.A.D SOINS

Pôle 0060 - POLE GERIATRIE

Grade 3037 - Aide-Soignant CN

Statut 30 - Contractuel CDI Emp Perm.

Modifié le 14/06/2023 à 18:34 par

Annexe 12 : Critères de recherches

EX: P804 2

Traitements A propos de Prodigie Déconnexion

Traitements des demandes agents

Liste des demandes Agents

Critères

Entité juridique

Etat

Date demande du 14 au 25

En cours par ROGER Gabriel

Domaine d'activité

Type

Matricule

Grade

Nom

Statut

Prénom

UF

Pôle

Avec notification

Effacer les critères Rechercher

Résultat

Résumé des critères : Etat : En attente , En cours En cours par ROGER Gabriel

Total : 13 Sé : 1

Annexe 13 : Ecran de discussion entre gestionnaire et agents pour une demande

Changement de coordonnées ★ ⓘ ⌵

Etat: En cours Prendre en charge Orienter la demande En cours de traitement par [nom] depuis le 01/09/2022

Demande Pièce-jointe Discussions Suivi interne

Réponse

Caractères : 0/1000

Bonjour,

Votre demande est refusée parce qu'on ne vous aime pas

Cordialement,

Le service RH

Réponses types: Refus par défaut ✕ Envoyer

Ajouter un document à la demande

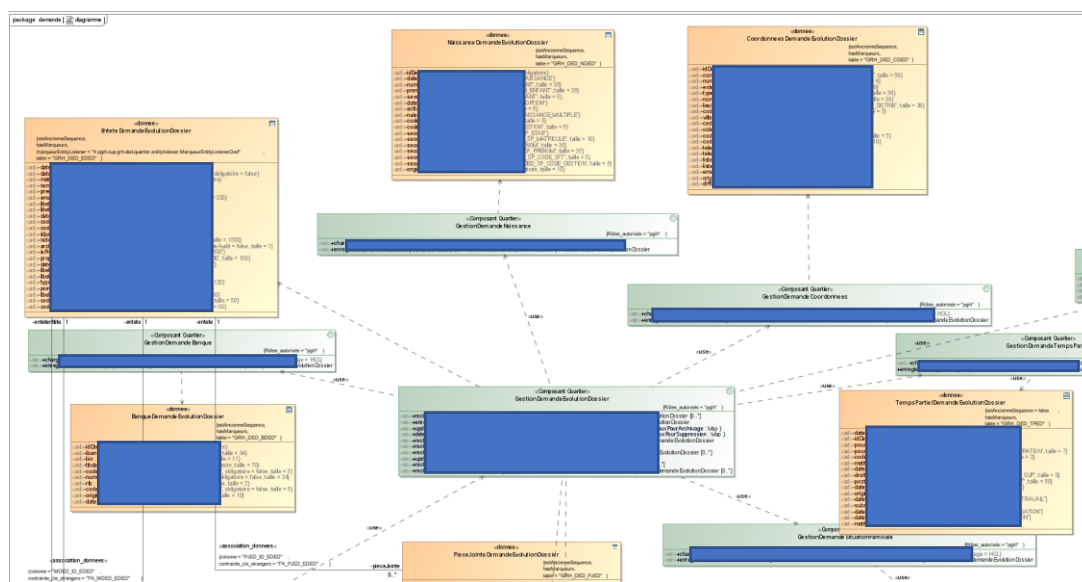
Choisir un fichier Aucun fichier choisi Ajouter le document

Historiques des échanges

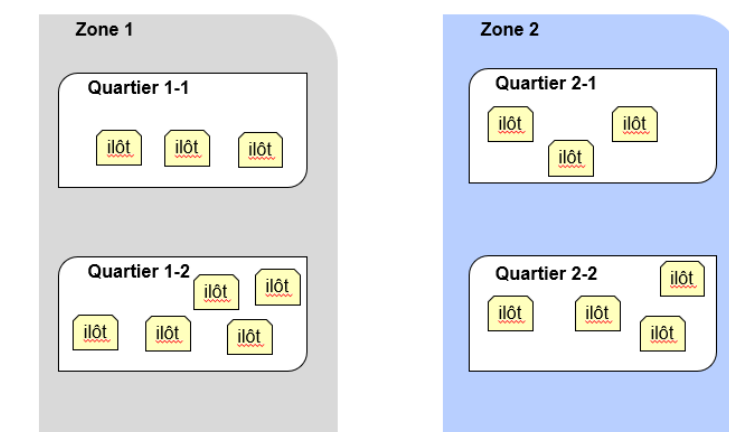
zedredzedredzed

[nom] envoyé le 01/09/2022 à 10:51

Annexe 14 : Diagramme Magic Draw







Annexe 15 : Schéma architecture Zone, Quartier, Ilots





























Annexe 16 : Modules d'un ilot (GitLab)

Prodige > sup-grh-ded

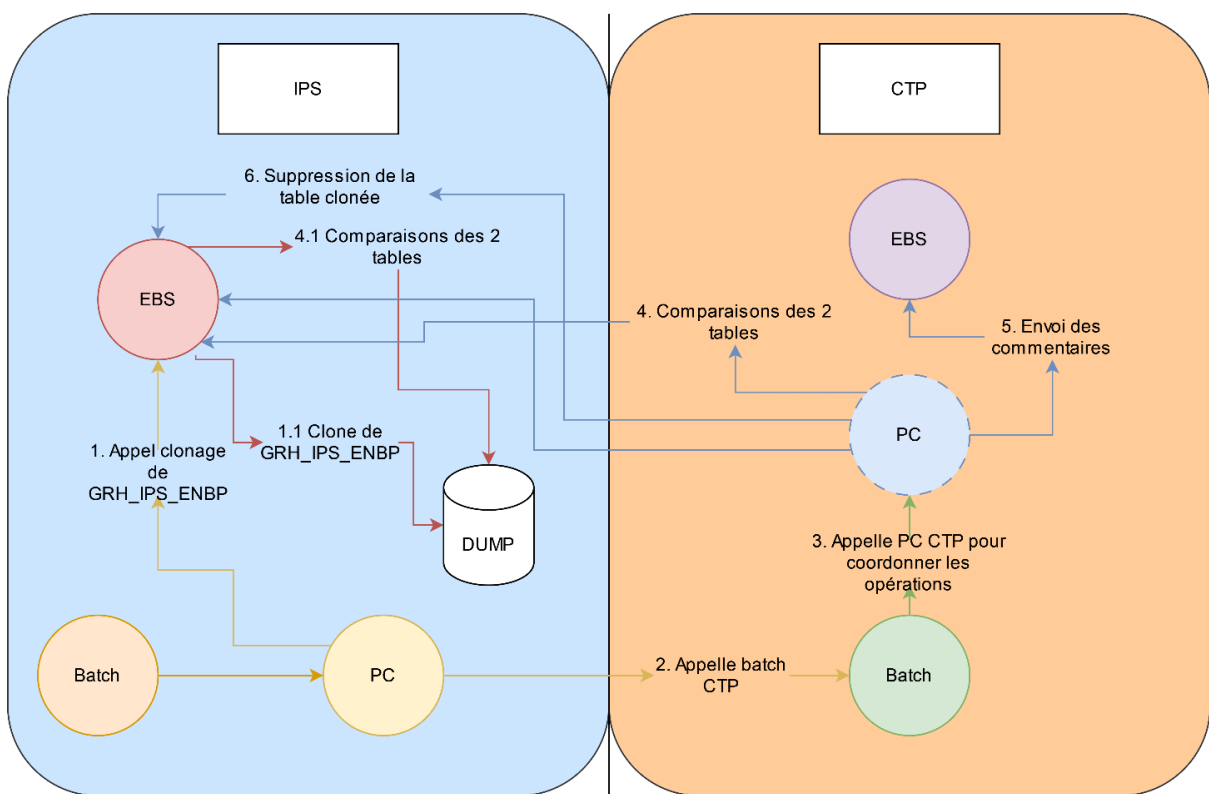
sup-grh-ded  Group ID: 326 

  [New subgroup](#) [New project](#)

Subgroups and projects Shared projects Archived projects Name ▾ ↓ ▴

		sup-grh-ded-application 	★ 3	1 day ago
		sup-grh-ded-batch-archivage-suppression 	★ 3	1 day ago
		sup-grh-ded-batch-import-analyse-map   Créé par Castor le 21 October 2022 à 13:17:01	★ 1	1 week ago
		sup-grh-ded-batch-maj-informations-agents 	★ 3	1 month ago
		sup-grh-ded-ebs 	★ 3	1 day ago
		sup-grh-ded-message   Créé par Castor le 19 May 2022 à 14:45:24	★ 3	1 month ago
		sup-grh-ded-pc 	★ 3	1 week ago
		sup-grh-ded-ws 	★ 3	2 days ago

Annexe 17 : Maquette technique comparaison de la paie



Annexe 18 : Maquettes fonctionnelles réponses types

✓

Changement de coordonnées

★ ⓘ ⌵

Etat

En cours

Prendre en charge

Orienter la demande

En cours de traitement par [redacted] depuis le 01/09/2022

Demande

Pièce-jointe

Discussions

Suivi interne

Réponse

Caractères : 0/1000

Bonjour,
Votre demande est refusée parce qu'on ne vous aime pas
Cordialement,
Le service RH

Réponses types

Refus par défaut

✕

Envoyer

Ajouter un document à la demande

Choisir un fichier

Aucun fichier choisi

Ajouter le document

Historiques des échanges

zedzedzedzedddzed

envoyé le 01/09/2022 à 10:51

Gestion des réponses types

✕

+

+

+

Refus par défaut

Refus motivé pour demande de temps partiel

Demande traitée - réponse simple

Contenu du message

Caractères : 0/ 1000

Enregistrer

Annuler

Motif de refus

Saisissez un motif de refus si nécessaire

Caractères : 0/1000

Réponses types

▼

OK

Annuler

Annexe 19 : Composants graphiques MClientDataGrid

Total : 20 Sél : 0

Demo DataGrid Client

Résultat affiché : 1 - 15

Id Technique	Civilité	Nom	Prenom	Date de naissance	Age	Genre	Montant Random	OK
4	Mme			07/11/1955	137	♀	320,71	✓
10	Mme			14/01/2005	88	♀	313,33	✓
12	Mme			14/01/2005	88	♀	145,64	✓
23	Mme			09/10/1974	118	♀	227,35	✓
3	Mlle			03/04/2004	89	♀	392,18	✓
15	Me			01/08/2008	84	♂	66,28	✓
1	M			01/08/2008	84	♂	92,00	✓
2	M			03/04/2007	86	♂	363,19	✓
5	M			03/04/2002	91	♂	214,41	✓
8	M			16/03/1965	128	♂	124,35	✓
11	M			01/08/2008	84	♂	220,15	✓
13	M			01/08/2008	84	♂	321,54	✓
14	M			01/08/2008	84	♂	271,79	✓
16	M			01/08/2008	84	♂	263,66	✓
17	M			01/08/2008	84	♂	150,78	✓

Moyenne : 16/06/2023 Age : 70 Moyenne : 215.22

Annexe 20 : Composants graphiques MToolBar

Exemples

MToolBar sans séparateur

Autres barres d'outils

Label de test

Bouton 1 Bouton 2 Bouton 3

Bouton 1 Bouton 3

test

Label de test Zone de test

Ajouter | Modifier | Effacer | Detail | Rechercher | Imprimer | Tout sélectionner | Aide | Filtrer | Ajouter | Modifier | Effacer | Detail | Rechercher | Imprimer | Tout sélectionner | Aide | Filtrer | Ajouter | Modifier | Effacer | Detail

Total : 3 Sél : 0

Non action

Résultat affiché : 1 - 3

Test

Total : 3 Sél : 0

Non action

Résultat affiché : 1 - 3

Test2

Annexe 21 : Zone de texte édition réponses types

Gestion des réponses types

Refus par défaut

Refus motivé pour demande de temps partiel

Demande traitée - réponse simple

Contenu du message

Caractères : 0/ 1000

Enregistrer Annuler

Annexe 22 : Code handler compteur de caractères

```
/**
 * Enregistre le handler de type change
 */
private void registerHandlerChange()
{
    registerHandler(dateLimiteTraitementChange());
    registerHandler(display.getDisplayDiscussionComposite().getRepondreAgentTextArea())
}
```

```

.addKeyUpHandler(new
RepondreAgentTextHandler(display.getDisplayDiscussionComposite())));

        registerHandler(display.getDisplayDiscussionComposite().getDialogBoxRe
ponsesTypesTextArea()
                .addKeyUpHandler(new RepondreAgentTextHandler(
                        display.getDisplayDiscussionComposite())));

        registerHandler(display.getDisplayDiscussionComposite().getDialogBoxRe
ponsesTypesTextArea()
                .addKeyUpHandler((KeyUpEvent event) ->
enableDialogBoxReponsesTypesButton()));
        registerHandler(display.getRefusDemandeMotifTextArea()
                .addKeyUpHandler(new RepondreAgentTextHandler(this)));

        // compteur de caractère d'ajout manuel de suivi
        registerHandler(display.getSuiviInterne().getAjoutSuiviTextArea()
                .addKeyUpHandler(new
AjoutManuelSuiviHandler(display.getSuiviInterne())));

        registerHandler(
                display.getDisplayDiscussionComposite().getDialogBoxReponsesTy
pesListBox()
                        .addValueChangeHandler((ValueChangeEvent<ReponsesTypes
MessageDED> event) -> {

                                if (event.getValue() != null &&
StringUtils.isEmpty(event.getValue().getMessages())) {
                                        display.getDisplayDiscussionComposite().getRep
ondreAgentTextArea()
                                                .setText(event.getValue().getMessages(
));
                                        display.getDisplayDiscussionComposite().getRep
ondreAgentTextArea().setFocus(true);

                                        getDisplay().getDisplayDiscussionComposite().g
etCompteur()
                                                .setText(event.getValue().getMessages(
).length() + "/1000");
                                } else {
                                        display.getDisplayDiscussionComposite().getRep
ondreAgentTextArea()
                                                .setText("");
                                        getDisplay().getDisplayDiscussionComposite().g
etCompteur()
                                                .setText(COMPTEUR_ZERO);
                                }
                        }
                }));
}

```

Sélection d'une ligne :

```
public void handleReponseTypesDataRowSelection() {
    registerHandler(reponsesTypesMessagesDEDDataGrid
        .addRowSelectionModelHandler(event -> {
            final ReponsesTypesMessageDED reponseType =
reponsesTypesMessagesDEDDataGrid
                .getReponseTypeDataRowSelectedRow();

            if (reponseType != null) {
                getDisplay().getDisplayDiscussionComposite().getDialogBoxRepon
sesTypesTextArea()
                    .setEnabled(true);

                if (StringUtils.isEmpty(reponseType.getMessages())) {
                    getDisplay().getDisplayDiscussionComposite().getDialogBoxR
eponsesTypesTextArea()
                        .setText(reponseType.getMessages());

                    getDisplay().getDisplayDiscussionComposite().getDialogBoxR
eponsesTypesCompteurFrappe()
                        .setText(reponseType.getMessages().length() +
"/1000");
                } else {
                    getDisplay().getDisplayDiscussionComposite().getDialogBoxR
eponsesTypesTextArea()
                        .setText("");
                    getDisplay().getDisplayDiscussionComposite().getDialogBoxR
eponsesTypesCompteurFrappe()
                        .setText(COMPTEUR_ZERO);
                }

                getDisplay().getDisplayDiscussionComposite().getDialogBoxRepon
sesTypesTextArea().setFocus(true);

            } else {
                getDisplay().getDisplayDiscussionComposite().getDialogBoxRepon
sesTypesTextArea()
                    .setEnabled(false);
                getDisplay().getDisplayDiscussionComposite().getDialogBoxRepon
sesTypesCompteurFrappe()
                    .setText(COMPTEUR_ZERO);
            }
        })
    );
}
```

Annexe 23 : Code enregistrement, modification et suppression réponses types

Ajout :

```
/**
 * Ajoute une réponse type
 */

public void addAjouterAction() {
    getToolBar().addAction(new
MAbstractSelectionAwareAction<ReponsesTypesMessageDED>(
        ToolbarStandardAction.AJOUTER, "Ajouter un message type",
this, TypeUiFonctionnalites.GERER_DEMANDES) {

        @Override
        public void execute() {
            dialogBoxAjoutReponsesTypesTextBox.setText("");
            dialogBoxAjoutReponsesTypes.setText(AJOUTER_UNE_REPONSE_TYPE);
            dialogBoxAjoutReponsesTypes.show();
            dialogBoxAjoutReponsesTypesTextBox.setFocus(true);
        }
    });
}

public void addSupprimerAction() {
    getToolBar().addAction(new
MAbstractSelectionAwareAction<ReponsesTypesMessageDED>(
        ToolbarStandardAction.EFFACER, "Supprimer un message type",
this,
        TypeUiFonctionnalites.GERER_DEMANDES) {

        @Override
        public void execute() {

            confirmerSuppression(getReponseTypeDataGridSelectedRow());
        }

    });
}

/**
 * supplique la reponse type sélectionnée
 */

public void addDupliquerAction() {
    getToolBar().addAction(new
MAbstractSelectionAwareAction<ReponsesTypesMessageDED>(
        ToolbarStandardAction.DUPLIQUER, "Dupliquer un message type",
this,
        TypeUiFonctionnalites.GERER_DEMANDES) {
```

```

        @Override
        public void execute() {

            dialogBoxAjoutReponsesTypesTextBox.setText(getReponseTypeDataGridSelectedRow().getNom() + " - Copie");

            dialogBoxAjoutReponsesTypes.setText("Dupliquer une réponse type");

            dialogBoxAjoutReponsesTypes.show();
            dialogBoxAjoutReponsesTypesTextBox.setFocus(true);

        }

        @Override
        protected State getActionStateInternal() {
            return (getCurrentSelection().isEmpty()) ? State.DISABLED : State.ENABLED;
        }
    });
}

/**
 * supprime la reponse type sélectionnée
 */
public void supprimerReponesType(ReponsesTypesMessageDED reponseType) {
    GwtDedASLocator.getGestionDemandeEvolutionDossierAS().supprimerReponsesTypesById(reponseType.getId(),
        new MAsyncCallback<ReponsesTypesMessageDED>() {

            @Override
            public void onSuccess(ReponsesTypesMessageDED result) {
                removeValue(reponseType);
                reload();
            }

        });
}

public ReponsesTypesMessageDED getReponseTypeDataGridSelectedRow() {
    final List<ReponsesTypesMessageDED> selectedRows = this.getSelectedRowsItems();
    if (selectedRows != null && !selectedRows.isEmpty()) {
        return selectedRows.get(0);
    }
    return null;
}

```

```

        private void confirmerSuppression(final ReponsesTypesMessageDED
reponseType) {
            MMessageBox.showConfirm(null,
ApplicationUIConstants.DEMANDE_CONFIRMATION,
                getMessageConfirmation(), true, result -> {
                    // Si la suppression est confirmée.
                    if (result) {
                        // On supprime
                        supprimerReponseType(reponseType);
                        selectRow(getRowItem(0), result);
                    }
                });
        }

/**
 * Retourne le message à afficher pour la confirmation
 *
 * @param reponseType à supprimer
 */
private String getMessageConfirmation() {
    return FormaterUtils
        .getConfirmationSuppression(ApplicationUIConstants.FEMININ_SIN
GULIER, "reponse type");
}

```

Mise à jour :

```

public void handleReponseTypesMisAJour() {
    registerHandler(
        display.getDisplayDiscussionComposite().getDialogBoxReponsesTypesEnreg
istrerButton()
            .addClickHandler((ClickEvent event) -> {
                ReponsesTypesMessageDED reponseType =
reponsesTypesMessagesDEDDDataGrid
                    .getReponseTypeDataGridSelectedRow();

                if (reponseType != null) {
                    reponseType.setMessages(display.getDisplayDiscussionCompos
ite()
                        .getDialogBoxReponsesTypesTextArea().getValue());
                    reponseType.setDateMaj(new Date());
                    reponseType.setUtilisateurMaj(CurrentUser.getInstance().ge
tName());

                    GwtDedASLocator.getGestionDemandeEvolutionDossierAS()
                        .mettreAJourReponsesTypes(
                            reponseType,

```

```

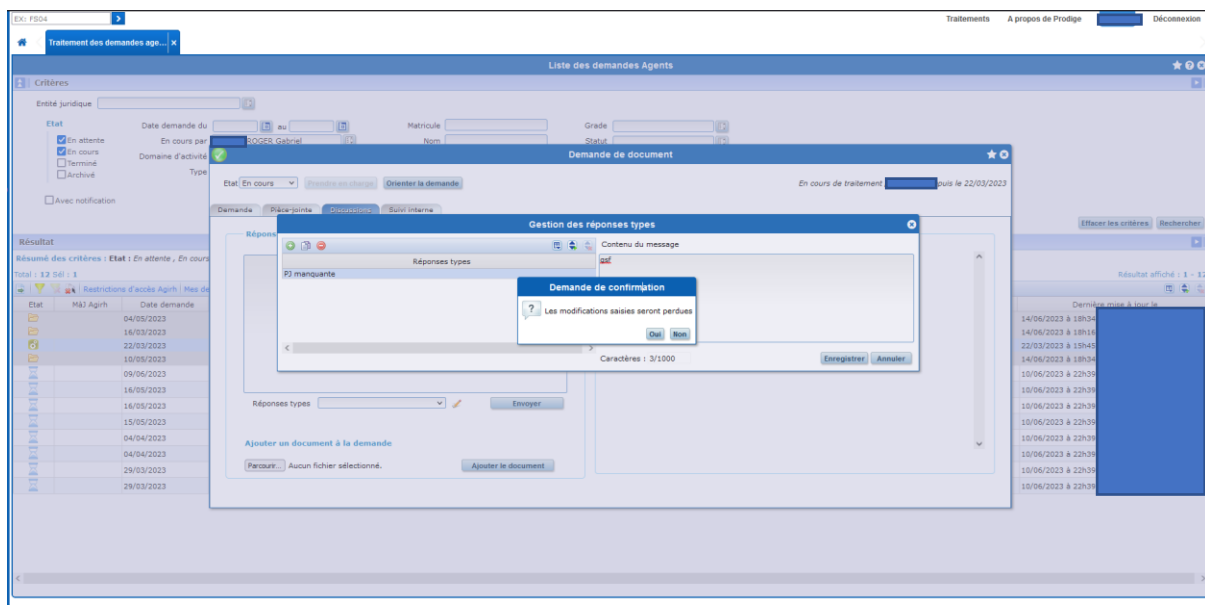
new MAsyncCallback<ReponsesTypesMessageDED>() {
    @Override
    public void onSuccess(ReponsesTypesMessageDED
result) {

        reponsesTypesMessagesDEDDataGrid.reLoad();
        display.getDisplayDiscussionComposite()
            .enableDialogBoxReponsesTypesButton(false);

    }
});
    }
}
);
}
}

```

Annexe 24 : Message avertissement perte des modifications



Annexe 25 : Code handler clique affichage écrans des réponses types

```

private void reponsesTypesMessageDedClickHandler() {

    registerHandler(display.getDisplayDiscussionComposite().getDialogBoxRe
ponsesTypesMenuToolTip()
        .addClickHandler((ClickEvent event) ->
        {
            display.getRepondreAgentTextArea().setText("");
            display.getDisplayDiscussionComposite()
                .getDialogBoxDiscussionReponsesTypes().show();
            reponsesTypesMessagesDEDDataGrid.setValue(listReponsesTypesMes
sagesDed);
        }));

    registerHandler(
        reponsesTypesMessagesDEDDataGrid

```



```

        .getDialogBoxAjoutReponsesTypesAnnulerButton()
        .addClickHandler((ClickEvent event) ->
reponsesTypesMessagesDEDDataGrid
        .getDialogBoxDiscussionAjoutReponsesTypes().hide()));

        registerHandler(
            display.getDisplayDiscussionComposite().getDialogBoxReponsesTypesAnnulerButton()
            .addClickHandler((ClickEvent event) -> {

                if
(reponsesTypesMessagesDEDDataGrid.getReponseTypeDataGridSelectedRow() != null)
{

                    display.getDisplayDiscussionComposite()
                        .getDialogBoxReponsesTypesTextArea().setText(repon
sesTypesMessagesDEDDataGrid
                        .getReponseTypeDataGridSelectedRow().getMessages()
);

                    display.getDisplayDiscussionComposite()
                        .enableDialogBoxReponsesTypesButton(false);

                }
            }));

        registerHandler(reponsesTypesMessagesDEDDataGrid
            .addValueChangeHandler(event -> {
                getDisplay().getDisplayDiscussionComposite().getDialogBoxRepon
sesTypesTextArea()
                    .setText("");
                getDisplay().getDisplayDiscussionComposite().getDialogBoxRepon
sesTypesTextArea()
                    .setEnabled(false);

                display.getDisplayDiscussionComposite().getDialogBoxReponsesTy
pesListBox().clear();

                display.getDisplayDiscussionComposite().getDialogBoxReponsesTy
pesListBox()
                    .addAll(listReponsesTypesMessagesDed);
            }));

        handleAjouterReponsesTypes();

        handleReponseTypesDataGridRowSelection();

        handleReponseTypesMisAJour();
    }

```

Annexe 26 : Table de données réponses types

Propriétés

Données

ER Diagram

GRH_DED_MTDE

Name:GRH_DED_MTDE

Table Type: N/A

Tablespace:

IOT Type:

Comment:Table contenant les messages preenregistres des discussions pour les demandes d'évolution de

IOT Name:

☐ Temporary

☐ Secondary

☐ Nested

Columns

	Nom de la colonne	#	Type	Type ...	Not ...	Déf...	Comment
123	ID	1	NUMBER(19...		<input checked="" type="checkbox"/>		Id de la table
	MTDE_DATE_CREATION	2	DATE		<input type="checkbox"/>		Date de création de la réponse type
	MTDE_DATE_MAJ	3	DATE		<input type="checkbox"/>		Date de mis à jour de la réponse type
abc	MTDE_CONTENU_MESSAGE	4	VARCHAR2(...		<input type="checkbox"/>		Contenu de la réponse type
abc	MTDE_NOM_MESSAGE	5	VARCHAR2(...		<input type="checkbox"/>		Nom de la réponse type
abc	MTDE_UTL_CREATION	6	VARCHAR2(...		<input type="checkbox"/>		Utilisateur qui à conçue la réponse type
abc	MTDE_UTL_MAJ	7	VARCHAR2(...		<input type="checkbox"/>		Utilisateur qui à mis à jour la réponse type
abc	EJ_CREATION	8	VARCHAR2(...		<input checked="" type="checkbox"/>		Numero de SIREN

8 items

GRH_DED_MTDE

Propriétés

Données

ER Diagram

GRH_DED_MTDE

Name:GRH_DED_MTDE

Table Type: N/A

Tablespace:

IOT Type:

Comment:Table contenant les messages preenregistres des discussions pour les demandes d'évolution de

IOT Name:

☐ Temporary

☐ Secondary

☐ Nested

Columns

	Nom	Propriétaire	Type	Condition	Status
>	PK_GRH_DED_MTDE	GRH_DED_MTDE	PRIMARY KEY		ENABLED
>	SYS_C0042320	GRH_DED_MTDE	CHECK	"ID" IS NOT NULL	ENABLED
>	SYS_C0042321	GRH_DED_MTDE	CHECK	"EJ_CREATION" IS NOT NULL	ENABLED

Annexe 27 : Mapping hibernate de la table réponses types et class Java

Entité Hibernate :

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- [GENERATED] -->
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN" "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="fr.pgih.sup.grh.ded.quartier.demande.domaine">

    <class name="ReponsesTypesMessageDED" table="GRH_DED_MTDE">
```

```

        <comment>
            <![CDATA[Table contenant les messages préenregistrés des
discussions pour les demandes d'évolution de dossier]]>
        </comment>

        <!-- Identifiant technique par défaut -->
        <id name="id" type="long" column="ID" unsaved-value="null">
            <generator
class="org.hibernate.id.enhanced.SequenceStyleGenerator">
                <param name="sequence_name">SEQ_GRH_DED_MTDE</param>
                <param name="optimizer">none</param>
            </generator>
        </id>

        <!-- propriété dateCreation -->
        <property name="dateCreation" type="mmdate"
column="MTDE_DATE_CREATION"/>

        <!-- propriété dateMaj -->
        <property name="dateMaj" type="mmdate" column="MTDE_DATE_MAJ"/>

        <!-- propriété messages -->
        <property name="messages" column="MTDE_CONTENU_MESSAGE"
length="1000"/>

        <!-- propriété nom -->
        <property name="nom" column="MTDE_NOM_MESSAGE" length="150"/>

        <!-- propriété utilisateurCreation -->
        <property name="utilisateurCreation" column="MTDE_UTI_CREATION"
length="35"/>

        <!-- propriété utilisateurMaj -->
        <property name="utilisateurMaj" column="MTDE_UTI_MAJ" length="35"/>

        <!-- propriété ejCreation -->
        <property name="ejCreation" >
            <column name="EJ_CREATION" not-null="true" length="20" >
                <comment>
                    <![CDATA[Numéro de SIREN]]>
                </comment>
            </column>
        </property>
        <!-- Filtre MultiEJ -->
        <filter name="filter_by_sirens_and_cooperation" />
    </class>
</hibernate-mapping>

```

Class Java :

```
/**
 * Table contenant les messages préenregistrés des discussions pour les
 * demandes d'évolution de dossier
 */
@XmlRootElement
public class ReponsesTypesMessageDED implements Identifiable<Long>,
MultiEjEntity, Serializable
{
    /** Identifiant par défaut de l'objet. */
    private Long id;

    private Date dateCreation;

    private Date dateMaj;

    @Length(max = 1000)
    private String messages;

    @Length(max = 150)
    private String nom;

    @Length(max = 35)
    private String utilisateurCreation;

    @Length(max = 35)
    private String utilisateurMaj;

    private String ejCreation;

    public ReponsesTypesMessageDED()
    {
        // constructeur sans initialisation
    }

    public ReponsesTypesMessageDED(String nom, String siren, String
userCreation, String userMaj) {

        this.nom = nom;
        this.ejCreation = siren;
        this.dateCreation = new Date();
        this.dateMaj = new Date();
        this.messages = "";
        this.utilisateurCreation = userCreation;
        this.utilisateurMaj = userMaj;
    }

    @Override
```

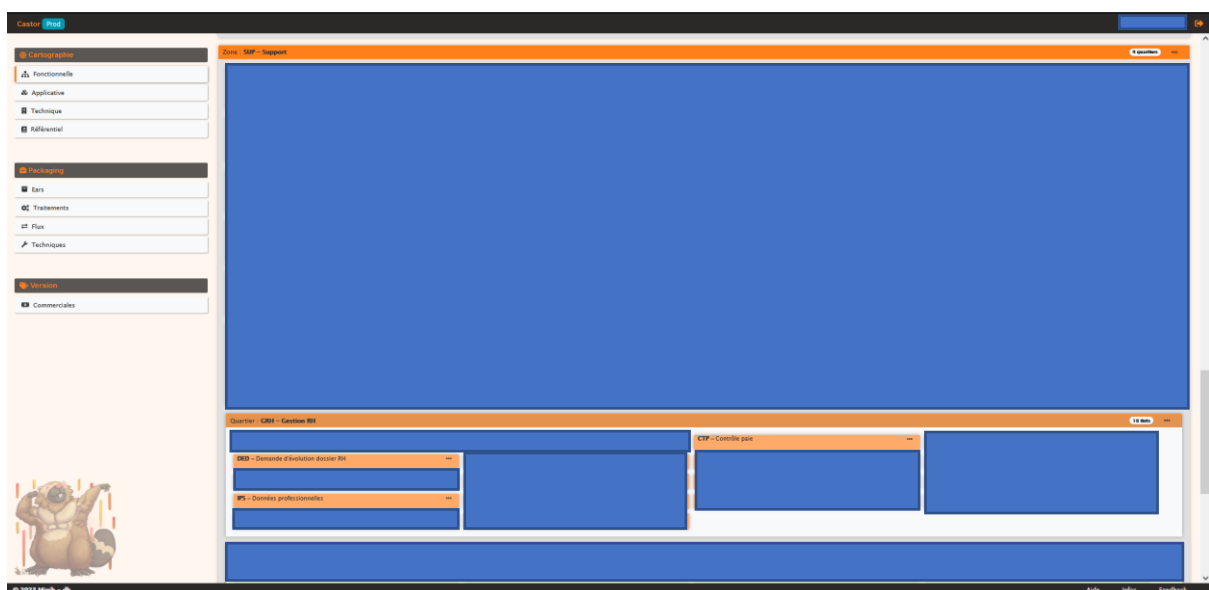
```

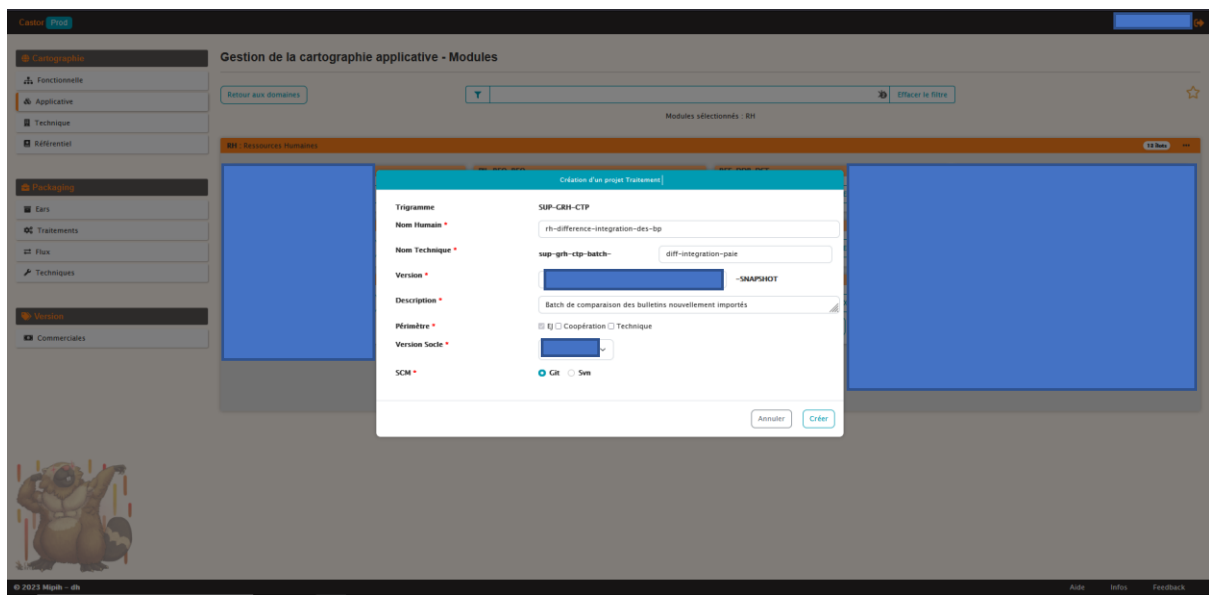
public Long id()
{
    return getId();
}

@XmlID
public String getSID()
{
    if (id() == null)
    {
        return null;
    }
    return id().toString();
}
/**
 * Représentation textuelle de cette classe.
 *
 * @return La représentation textuelle de cette classe
 */
@Override
public String toString()
{
    String toString = "ReponsesTypesMessageDED [id=" + id + ",
dateCreation=" + dateCreation + ", dateMaj=" + dateMaj + ", messages=" +
messages + ", nom="
        + nom + ", utilisateurCreation=" + utilisateurCreation + ",
utilisateurMaj=" + utilisateurMaj + "]";
    return toString;
}
}

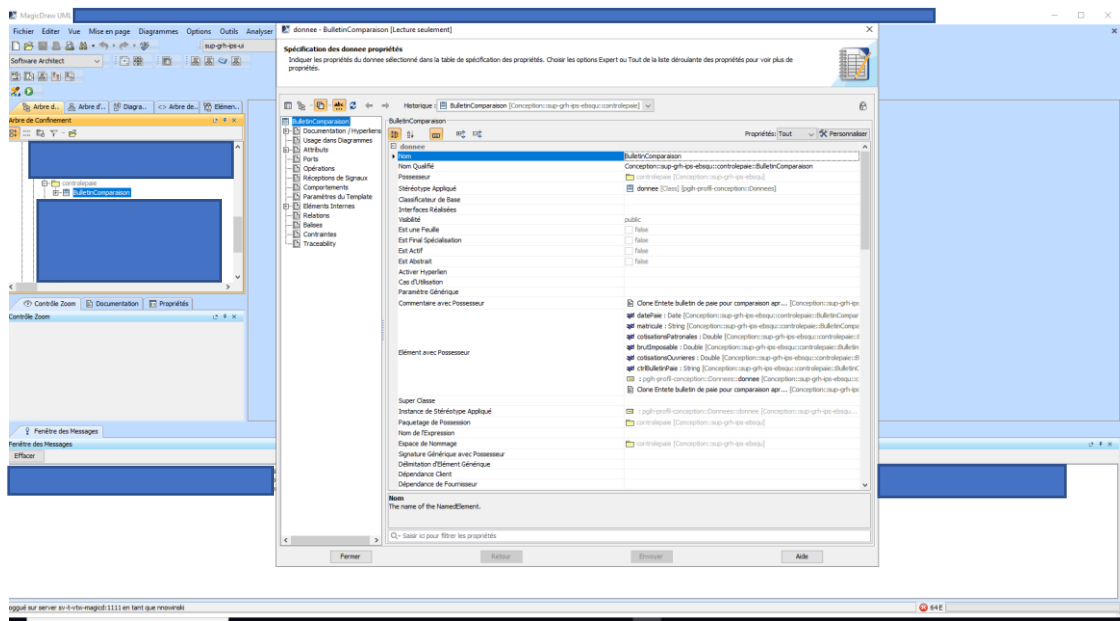
```

Annexe 28 : Creation bu batch de differences des bulletins avec Castor





Annexe 29 : Entité Magic Draw de la table de clone des bulletins



«donnee» BulletinComparaison	
	{estAncienneSequence = false, estMultiEj, hasMarqueurs = false, table = "GRH_IPS_COBP"}
«ad»-datePaie : Date(colonne = "COBP_DATE_PAIE", obligatoire)	
«ad»-matricule : String(nonunique,colonne = "COBP_MATRICULE_AGENT", obligatoire, taille = 14)	
«ad»-cotisationsPatronales : Double(nonunique,colonne = "COBP_TOTAL_COTIS_PAT")	
«ad»-brutImposable : Double(nonunique,colonne = "COBP_TOTAL_BRUT")	
«ad»-cotisationsOuvrieres : Double(colonne = "COBP_TOTAL_COTIS_SAL")	
«ad»-ctrlBulletinPaie : String(colonne = "COBP_ETAT_CTRL", obligatoire = false, taille = 6)	

Annexe 30 : Mapping hibernate de la table de clone des bulletins et class Java

Class Java :

```
/**
 * nouveau mapping pour l'entité ControleBulletinComparaison afin de faire la
 * lecture uniquement pour le controle paie
 */
@XmlRootElement
@Entity
@EntityListeners(MarqueurEntityListener.class)
public class ControleBulletinComparaison implements Identifiable<Long>,
MultiEjEntity, Serializable
{
    private Long id;

    private String matricule; //

    private Double brutImposable; //

    private Double cotisationsOuvrieres; //

    private Double cotisationsPatronales; //

    private Double precomptes; //

    private Date moisPaie; //

    private String ejCreation; //

    private String ctrlBulletinPaie; // etat

    @Override
    public Long id()
    {
        return getId();
    }

    @XmlID
    public String getSID()
    {
        if (id() == null)
        {
            return null;
        }
        return id().toString();
    }
}
```

Annexe 31 : Table de clone des bulletins

Table originelle :

[illegible]

Clone :

Propriétés

Données

ER Diagram

GRH_IPS_COBP

Name:

GRH_IPS_COBP

Table Type:

N/A

Tablesapce:

IOT Type:

Comment:

Entete du bulletin de paie.

IOT Name:

☐ Temporary

☐ Secondary

☐ Nested

	Nom de la colonne	#	Type	Type ...	Not ...	Déf...	Comment
Columns	123 ID	1	NUMBER(19...		<input checked="" type="checkbox"/>		
Constraints	abc COBP_MATRICULE_AGENT	2	VARCHAR2(...		<input type="checkbox"/>		Matricule de l'agent.
Foreign Keys	123 COBP_TOTAL_BRUT	3	NUMBER(19...		<input type="checkbox"/>		Total Brut pour l'attestation Cerfa.
References	123 COBP_TOTAL_COTIS_SAL	4	NUMBER(19...		<input type="checkbox"/>		Total cotisations salariales attestation Cerfa.
Triggers	123 COBP_TOTAL_COTIS_PAT	5	NUMBER(19...		<input type="checkbox"/>		Total cotisations patronales pour l'attestation Cerfa.
Indexes	123 COBP_MONTANT_PRECOMPTE	6	NUMBER(19...		<input type="checkbox"/>		Total precomptes.
Partitions	abc COBP_DATE_PAIE	7	DATE		<input type="checkbox"/>		Mois de la paie
Privileges	abc EI_CREATION	8	VARCHAR2(...		<input checked="" type="checkbox"/>		Numero de SIREN
Statistics / ...	abc COBP_ETAT_CTRL	9	VARCHAR2(6)		<input type="checkbox"/>		Etat du controle du bulletin
DDL							
Virtual							

Annexe 32 : Requête HSQL de clone des bulletins

```
@Override
public void cloneBulletinPaieTable(String sirenEj, Date moisPaie)
{
    StringBuilder hqlBuilder = new StringBuilder();
    hqlBuilder.append("INSERT INTO ControleBulletinComparaison (matricule,
brutImposable, cotisationsOuvrieres,");
    hqlBuilder.append(" cotisationsPatronales, precomptes, moisPaie,
ejCreation, ctrlBulletinPaie ");
    hqlBuilder.append("SELECT t.matricule, t.brutImposable,
t.cotisationsOuvrieres, t.cotisationsPatronales,");
    hqlBuilder.append(" t.precomptes, t.moisPaie, t.ejCreation,
t.ctrlBulletinPaie ");
    hqlBuilder.append("FROM ControleEnteteBulletinPaie t ");
    hqlBuilder.append("WHERE t.moisPaie = :currentDate");
    hqlBuilder.append(" AND t.ctrlBulletinPaie = :state");
    hqlBuilder.append(" AND t.ejCreation = :sirenEj");

    String hql = hqlBuilder.toString();

    Query query = getEntityManager().createQuery(hql);
    query.setParameter("currentDate", Date.from(moisPaie.toInstant()));
    query.setParameter("state", "BULLOK");
    query.setParameter("sirenEj", sirenEj);

    query.executeUpdate();
}
```

Annexe 33 : Requête HSQL de comparaison des bulletins

```
@Override
public List<ControleBulletinComparaison> compareBulletinPaie(Date
dateBulletin)
{
    StringBuilder sb = new StringBuilder();

    sb.append("SELECT gie2 FROM ControleBulletinComparaison gie2");
    sb.append(" LEFT JOIN ControleEnteteBulletinPaie gie ON gie.matricule
= gie2.matricule AND gie.moisPaie = :mois2");
    sb.append(" WHERE gie2.ctrlBulletinPaie = 'BULLOK' AND gie2.moisPaie =
:mois1 AND (gie IS NULL OR ");
    sb.append("      gie2.brutImposable <> gie.brutImposable AND ");
    sb.append("      gie2.cotisationsOuvrieres <> gie.cotisationsOuvrieres
AND ");
    sb.append("      gie2.cotisationsPatronales <>
gie.cotisationsPatronales AND ");
    sb.append("      gie2.precomptes <> gie.precomptes)");
}
```

```

        TypedQuery<ControleBulletinComparaison> createQuery =
getEntityManager().createQuery(sb.toString(),
ControleBulletinComparaison.class);
        createQuery.setParameter("mois2", Date.from(LocalDate.of(2023, 2,
28).atStartOfDay().atZone(ZoneId.systemDefault()).toInstant()));
        createQuery.setParameter("mois1", Date.from(LocalDate.of(2023, 3,
31).atStartOfDay().atZone(ZoneId.systemDefault()).toInstant()));

        return createQuery.getResultList();
    }

```

Annexe 34 : Requête HSQL de mise à jour de l'état des bulletins

```

@Override
public void updateBulletinPaieTable()
{
    StringBuilder sb = new StringBuilder();

    sb.append("UPDATE ControleEnteteBulletinPaie gie1");
    sb.append(" SET gie1.ctrlBulletinPaie = (Select gie2.ctrlBulletinPaie
FROM ControleBulletinComparaison gie2");
    sb.append(" WHERE gie1.ejCreation = gie2.ejCreation AND gie1.moisPaie
= gie2.moisPaie AND gie1.matricule = gie2.matricule)");

    getEntityManager().createQuery(sb.toString()).executeUpdate();
}

```

```

@Override
public void updateBulletinPaieARencontrer(String sirenEj, Date datePaie,
List<String> matricules)
{
    StringBuilder sb = new StringBuilder();

    sb.append("UPDATE BulletinComparaison gie1");
    sb.append(" SET gie1.ctrlBulletinPaie = 'REBULL'");
    sb.append(" WHERE gie1.ejCreation = :siren AND gie1.datePaie =
:currentDate AND gie1.matricule in (:matricules)");

    Query query = getEntityManager().createQuery(sb.toString());
    query.setParameter("currentDate",
DatesUtils.getDateDernierJourMois(datePaie));
    query.setParameter("siren", sirenEj);
    query.setParameter("matricules", matricules);

    query.executeUpdate();
}

```

Annexe 35 : Requête HSQL de suppression des données la table clone des bulletins

```
@Override
public void deleteBulletinPaieTable()
{
    String hql = "DELETE FROM ControleBulletinComparaison";
    getEntityManager().createQuery(hql).executeUpdate();
}
```

Annexe 36 : Enregistrement des commentaires en bases avec la méthode store de Hibernate

```
@Override
public CommentaireBulletin enregistrerCommentaireBulletin(CommentaireBulletin
commentaireBulletin) throws ValidationException
{
    return store(commentaireBulletin);
}

@Override
public List<CommentaireBulletin>
enregistrerCommentaireBulletin(List<CommentaireBulletin> commentaireAEnvoyer)
throws CommentaireBulletinException
{
    List<CommentaireBulletin> comments = new ArrayList<>();

    try {
        for (CommentaireBulletin commentaire : commentaireAEnvoyer)
        {
            CommentaireBulletin savedCommentaire =
enregistrerCommentaireBulletin(commentaire);
            comments.add(savedCommentaire);
        }
    }
    catch (Exception e)
    {
        throw new
CommentaireBulletinException(CommentaireBulletinException.Messages.ERREUR_SAUVEGARDE_COMMENTAIRE);
    }

    return comments;
}
```

Annexe 37 : Lancement du batch de comparaison des bulletins depuis Jenkins

Up
État
Répertoire de travail
Exécution avec paramètres
Supprimer Projet
Configurer
Relancer la dernière exécution
Move

Historique des exécutions **lancement**

Find

23.50	
23.28	

Build

Projet RH - Difference integration des BP

Cette exécution nécessite des paramètres :

jbossTransactionsTimeout: 3600
Durée relative au time out d'une transaction (en sec)

Utilisateur: [redacted]
Utilisateur exécutant ce traitement

Origine: [redacted]
Origine de l'utilisateur exécutant ce traitement (Ordonnanceur, LDAP, PRODIGE)

EjCreation: [redacted]

Siren de l'établissement: [redacted]

DatePaie: [redacted]
Date du mois de la paie à importer.

Annexe 38 : Lancement du batch de comparaison des bulletins depuis le code Java

```
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyyMMdd");

final List<ParametreJob> parametreMap = new ArrayList<>();
parametreMap.add(new ParametreJob("origin", "[redacted]"));
parametreMap.add(new ParametreJob("login", "[redacted]"));
parametreMap.add(new ParametreJob("EJ", sirenEj));
parametreMap.add(new ParametreJob("datePaie", dateFormat.format(datePaie)));

getGestionTraitementBatchSI().lancerJob("diff-integration-paie", parametreMap, sirenEj);
```

Annexe 39 : Code méthode getDemandesCongesATraiter pour le WS des congés

```
@Override
@RolesAllowed({ TypeWsFonctionnalites.WS_BPM })
public Response getDemandesCongesATraiter(String siren, String matricule,
String dateDebut, String dateFin, String nomDemande, String matriculeAgent,
String typePersonnel, boolean asRemplacant, boolean asVisualisateur)
{
    try
    {
        // Gestion du super-utilisateur
        boolean isSuperUtilisateur =
matricule.equalsIgnoreCase(SUPER_UTILISATEUR);

        // Affection de la date de début si renseignée sinon dateMinimum
        final Date dateStart = StringUtils.isNotBlank(dateDebut) ?
checkAndParseDateFormat(dateDebut)
: InfiniteDate.INFINITE_PAST_DATE;

        // Affection de la date de fin si renseignée sinon dateMax
        final Date dateEnd = StringUtils.isNotBlank(dateFin) ?
checkAndParseDateFormat(dateFin)
: InfiniteDate.INFINITE_FUTURE_DATE;
```

```

        final DemandeDTOCriteria criteria = new DemandeDTOCriteria();
        criteria.addToSirenEj(siren);
        // uniquement les demandes de congés en cours d'exécution
        criteria.setEtats(Arrays.asList(ProcessusExecutionStatus.EXECUTING,
ProcessusExecutionStatus.ACCEPTED, ProcessusExecutionStatus.REFUSED));
        criteria.setAsValideur(true);
        if (nomDemande != null)
        {
            criteria.setNomDemande(Arrays.asList(nomDemande));
        }
        criteria.setCodeTypeDemande(TypeDemandeEnum.CONGE.getName());

        if (StringUtils.isNotBlank(matriculeAgent))
        {
            final Agent agent = getAgent(matriculeAgent, siren);

            criteria.setNom(agent.getNom());
            criteria.setPrenom(agent.getPrenom());
        }

        criteria.setTypePersonnel(typePersonnel);

        // Dans le cas ou le matricule est celui d'un super-utilisateur, on
        ignore les paramètres asRemplacant et asVisualisateur
        if (!isSuperUtilisateur)
        {
            criteria.setAsRemplacant(asRemplacant);
            criteria.setAsVisualisateur(asVisualisateur);

            // gestion du responsable
            final Agent responsable = getAgent(matricule, siren);

            criteria.setUserProdige(responsable.getLoginProdige());
            criteria.setMatriculeResponsable(responsable.getMatricule());
        }

        List<DemandeDTO> demandesRetour =
getGestionDemandeComponent().rechercherDemande(criteria, true);

        // on convertit les demandes en demandes de congés
        List<DemandeCongeDTO> demandesConges = new ArrayList<>();

        if (CollectionUtils.isNotEmpty(demandesRetour))
        {
            // on extrait les codes fonctionnels des demandes congés
            final List<String> codesFonctionnels = demandesRetour.stream()
                .map(DemandeDTO::getNomDemande)
                .collect(Collectors.toList());

```

```

        // on va chercher les entêtes dans DED
        final List<CongeEntete> congesEntete =
getGestionWSDemande().getCongeEnteteFromCodeFonc(codesFonctionnels);

        if (CollectionUtils.isNotEmpty(congesEntete))
        {
            // Filtre sur les conges se trouvant à cheval sur date de
début et de fin
            // (date fin conge > date deb paramètre et date deb conge <
date fin paramètre)
            // Ajout des congés et des demandes dans la liste de retour du
WS
            congesEntete.stream()
                .filter(c -> DateUtils.compareDatePartOnly(dateStart,
c.getConge().getDateFin()) <= 0
                    && DateUtils.compareDatePartOnly(dateEnd,
c.getConge().getDateDebut()) >= 0)
                .forEach(cong -> {
                    DemandeCongeDTO demConge = new
DemandeCongeDTO(demandesRetour.stream()
                        .filter(d ->
ObjectUtils.compare(d.getNomDemande(), cong.getEntete().getCodeFonctionnel())
== 0)
                            .findFirst().orElse(null));
                    demConge.setEnteteConge(cong);
                    demandesConges.add(demConge);
                });
        }
    }

    // on retire le template du processus pour la GTA
demandesConges.forEach(d -> {
    d.setProcessus(null);
    if (isSuperUtilisateur)
    {
        d.setAsValideur(true);
        d.setAsVisualisateur(false);
        d.setAsRemplacant(false);
    }
});

    if (CollectionUtils.isEmpty(demandesConges))
    {
        return Response.noContent().build();
    }
}

```

```

        return Response.ok().entity(new
DemandesCongeDTO(demandesConges)).build();
    }
    catch (ClientParamException e)
    {
        return Response.status(e.getStatus()).entity(e.getMessage()).build();
    }
    catch (Exception e)
    {
        LOGGER.error("Erreur durant la recherche des congés à traiter", e);
        return Response.serverError().entity("Erreur durant la recherche des
congés à traiter").build();
    }
}

/**
 *
 * permet de retourner un agent à partir de son matricule
 *
 * @param matricule
 * @return
 */
private Agent getAgent(String matricule, String siren) throws
ClientParamException
{
    final AgentQuartierCriteria agentQuartierCriteria = new
AgentQuartierCriteria();
    agentQuartierCriteria.setMatricule(matricule);
    agentQuartierCriteria.getSirenEj().add(siren);
    final List<Agent> agents =
getGestionAgentQuartier().rechercherAgent(agentQuartierCriteria);

    Agent agent = agents.stream()
        .findFirst()
        .orElse(null);

    if (agent == null)
    {
        LOGGER.warn("L'agent {} n'a pas été trouvé", matricule);
        throw new ClientParamException(Status.BAD_REQUEST, "L'agent " +
matricule + " n'a pas été trouvé");
    }
    return agent;
}
}

// test

```

Annexe 40 : Critères de recherches des demandes agents

The screenshot shows a web application window titled 'Liste des demandes Agents'. It features a search criteria section on the left with the following fields and options:

- Entité juridique: [dropdown]
- Etat: ☒ En attente, ☒ En cours, ☐ Terminé, ☐ Archivé
- Date demande du: [date picker] au: [date picker]
- En cours par: [text input with 'ROGER Gabriel']
- Matricule: [text input]
- Grade: [dropdown]
- Nom: [text input]
- Prénom: [text input]
- Statut: [dropdown]
- UF: [dropdown]
- Pôle: [dropdown]
- Domaine d'activité: [dropdown]
- Type: [dropdown]
- ☐ Avec notification

At the bottom, there is a 'Résultat' section showing a summary: 'Résumé des critères : Etat : En attente , En cours En cours par : ROGER Gabriel'. There are also buttons for 'Effacer les critères' and 'Rechercher'.

Annexe 41 : Test unitaire du ws des congés

```
@Test()
@WithAuthorizations({ @FunctionnalityOnSirens(TypeWsFonctionnalites.WS_BPM) })
public void testgetDemandesCongesATraiterErreur_500()
{
    final DemandeDTO demande = getMockEnteteDemandeDTO();

    Mockito.when(
        getMockGestionDemandeComponent().rechercherDemande(Mockito.any(
            DemandeDTOCriteria.class),
            Mockito.anyBoolean()))
        .thenReturn(Arrays.asList(demande));

    Mockito.when(getMockGestionAgentQuartier().rechercherAgent(Mockito.any(
        AgentQuartierCriteria.class)))
        .thenReturn(Arrays.asList(getMockAgent()));

    final EnteteDemandeEvolutionDossier entete = new
    EnteteDemandeEvolutionDossier();
    entete.setCodeFonctionnel(demande.getNomDemande());
    final CongeDemandeEvolutionDossier c = new
    CongeDemandeEvolutionDossier();
    c.setDateDebut(DateUtils.createDate(2021, Calendar.JANUARY, 1));
    c.setDateFin(DateUtils.createDate(2021, Calendar.DECEMBER, 31));
    final CongeEntete conge = new CongeEntete();
    conge.setEntete(entete);
    conge.setConge(c);

    Mockito.when(
        getMockGestionWSDemande().getCongeEnteteFromCodeFonc(Matchers.
        anyListOf(String.class)))
        .thenReturn(Arrays.asList(conge));

    final Response response = service.getDemandesCongesATraiter("081",
    "SUPER", "20210101", "20211231", "ABCDE12345", "123456", "PNM", false, false);

    assertThat(response.getStatus()).isEqualTo(200);
}
```



```

        final DemandesCongeDTO demandesDTO =
(DemandesCongeDTO)response.getEntity();

        assertThat(demandesDTO).isNotNull();
        assertThat(demandesDTO.getDemandes().size()).isEqualTo(1);

        final DemandeCongeDTO demandeConge =
(DemandeCongeDTO)demandesDTO.getDemandes().get(0);

        assertThat(demandeConge.getEnteteConge()).isNotNull();
    }

```

Annexe 42 : Code méthode filter du ws des formulaires

```

public static List<ParametrageDemandeAgent>
filter(List<ParametrageDemandeAgent> parametrages, String strTypePersonnel)
{
    List<ParametrageDemandeAgent> found = new ArrayList<>();

    Predicate<ParametrageDemandeAgent> codeNotInList = parametrage ->
found.stream()
    .noneMatch(ft ->
ft.getTypeDemande().equals(parametrage.getTypeDemande()));

    Predicate<ParametrageDemandeAgent> typePersonnelNotPNMAndPMD =
codeNotInList.and(parametrage -> parametrage.getTypesPersonnel().stream()
    .noneMatch(typePerso -> Arrays.asList("PNM",
"PMD").contains(typePerso.getCode())));

    Predicate<ParametrageDemandeAgent> typePersonnelInQuery =
codeNotInList.and(parametrage -> parametrage.getTypesPersonnel().stream()
    .anyMatch(typePerso ->
typePerso.getCode().equals(strTypePersonnel)));

    parametrages.forEach(param -> {
        if (typePersonnelInQuery.test(param))
        {
            found.add(param);
        }
    });

    parametrages.forEach(param -> {
        if (typePersonnelNotPNMAndPMD.test(param))
        {
            found.add(param);
        }
    });
}

```

```

        return found.stream()
            .sorted(Comparator.comparing(ParametrageDemandeAgent::getTypeDemande))
            .collect(Collectors.toList());
    }

```

Annexe 43 : Test unitaire du ws des formulaires

```

@Test
public void getParametrageListeTest() throws DemandeEvoDossierException
{
    List<ParametrageDemandeAgent> params = getListParametrage();

    Mockito.when(gestionMockParametrageDemandeAgent().rechercherAvecNational(Mockito.any(ParametrageDemandeAgentCriteria.class))).thenReturn(params);
    Mockito.when(gestionMockNomenclatureProdige().chargerElementReferentielsProdige(Mockito.anyListOf(String.class), Mockito.anyString(),
        Mockito.any(Date.class))).thenReturn(new ArrayList<>());
    Mockito.when(gestionMockDocumentMap().getDocumentsByCriteria(Mockito.any(MapDocumentCriteria.class))).thenReturn(Arrays.asList(new DocumentMap()));

    List<EnteteParametrageDTO> demande =
gestionParametrage.parametrageDemandeListe("SIREN", "PNM", "MATRIC");
    assertNotNull(demande);
    assertTrue(demande.stream().anyMatch(d ->
d.getTypeDemande().equals("DEMANDE_PNM")));
    assertTrue(demande.stream()
        .anyMatch(d ->
d.getTypeDemande().equals("DEMANDE_PMD_CONTRAT_VALIDE") &&
d.getLibelleDemande().equals("Demande pmd contrat valide")));
    assertTrue(demande.stream().anyMatch(d ->
d.getTypeDemande().equals("DEMANDE_PNM_2") &&
d.getLibelleDemande().equals("Demande pnm 02")));

    assertFalse(demande.stream().anyMatch(d ->
d.getTypeDemande().equals("DEMANDE_PNM_CONTRAT_VALIDE")));
    assertFalse(demande.stream().anyMatch(d ->
d.getTypeDemande().equals("DEMANDE_PMD")));
    assertFalse(demande.stream()
        .anyMatch(d -> d.getTypeDemande().equals("DEMANDE_PNM_2") &&
d.getLibelleDemande().equals("Demande pnm 02 pmd contrat valide")));
}

private List<ParametrageDemandeAgent> getListParametrage()
{
    List<ParametrageDemandeAgent> params = new ArrayList<>();

    TypePersonnel pnm = new TypePersonnel();

```

```

pnm.setCode("PNM");

TypePersonnel pmd = new TypePersonnel();
pmd.setCode("PMD");

TypePersonnel pmdContratValideAUneDate = new TypePersonnel();
pmdContratValideAUneDate.setCode("Agents PMD avec contrat valide à une
date");

TypePersonnel pnmContratValideAUneDate = new TypePersonnel();
pnmContratValideAUneDate.setCode("Agents PNM avec contrat valide à une
date");

Set<TypePersonnel> typesPersonnel1 = new HashSet<TypePersonnel>();
typesPersonnel1.add(pnm);
Set<TypePersonnel> typesPersonnel2 = new HashSet<TypePersonnel>();
typesPersonnel2.add(pmd);
Set<TypePersonnel> typesPersonnel3 = new HashSet<TypePersonnel>();
typesPersonnel3.add(pmdContratValideAUneDate);
Set<TypePersonnel> typesPersonnel4 = new HashSet<TypePersonnel>();
typesPersonnel4.add(pnmContratValideAUneDate);

ParametrageDemandeAgent param1 = new ParametrageDemandeAgent();
param1.setId(1L);
param1.setChamp(new HashSet<>());
param1.setTypeDemande("DEMANDE_PNM");
param1.setTypesPersonnel(typesPersonnel1);
param1.setLibelleDemande("Demande pnm");

ParametrageDemandeAgent param2 = new ParametrageDemandeAgent();
param2.setId(2L);
param2.setChamp(new HashSet<>());
param2.setTypeDemande("DEMANDE_PMD");
param2.setTypesPersonnel(typesPersonnel2);
param2.setLibelleDemande("Demande pmd");

ParametrageDemandeAgent param3 = new ParametrageDemandeAgent();
param3.setId(3L);
param3.setChamp(new HashSet<>());
param3.setTypeDemande("DEMANDE_PMD_CONTRAT_VALIDE");
param3.setTypesPersonnel(typesPersonnel3);
param3.setLibelleDemande("Demande pmd contrat valide");

ParametrageDemandeAgent param4 = new ParametrageDemandeAgent();
param4.setId(4L);
param4.setChamp(new HashSet<>());
param4.setTypeDemande("DEMANDE_PMD_CONTRAT_VALIDE");
param4.setTypesPersonnel(typesPersonnel4);

```

```

param4.setLibelleDemande("Demande pnm contrat valide");

ParametrageDemandeAgent param5 = new ParametrageDemandeAgent();
param5.setId(5L);
param5.setChamp(new HashSet<>());
param5.setTypeDemande("DEMANDE_PNM_2");
param5.setTypesPersonnel(typesPersonnel1);
param5.setLibelleDemande("Demande pnm 02");

ParametrageDemandeAgent param6 = new ParametrageDemandeAgent();
param6.setId(6L);
param6.setChamp(new HashSet<>());
param6.setTypeDemande("DEMANDE_PNM_2");
param6.setTypesPersonnel(typesPersonnel1);
param6.setLibelleDemande("Demande pnm 02 pmd contrat valide");

ChampParametrage champ = new ChampParametrage();
champ.setCode("CODE");

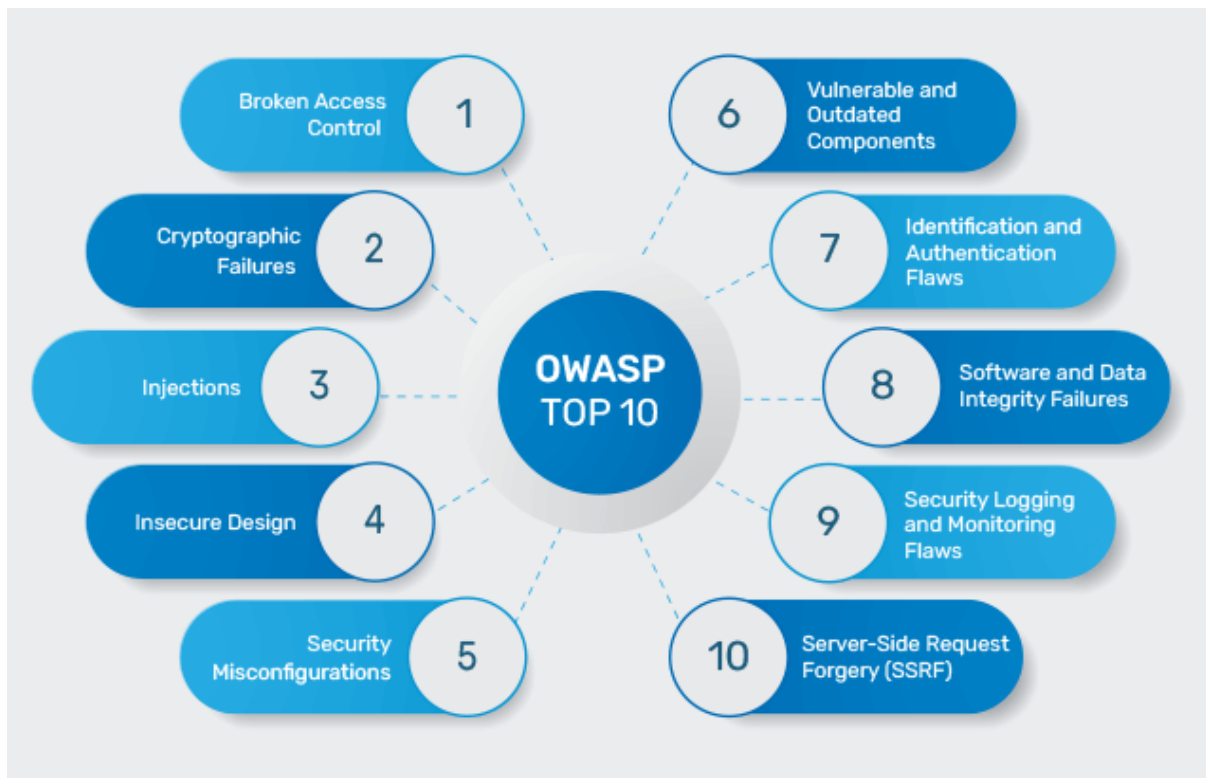
param1.getChamp().add(champ);
param2.getChamp().add(champ);
param3.getChamp().add(champ);
param4.getChamp().add(champ);
param5.getChamp().add(champ);
param6.getChamp().add(champ);

params.add(param1);
params.add(param2);
params.add(param3);
params.add(param4);
params.add(param5);
params.add(param6);

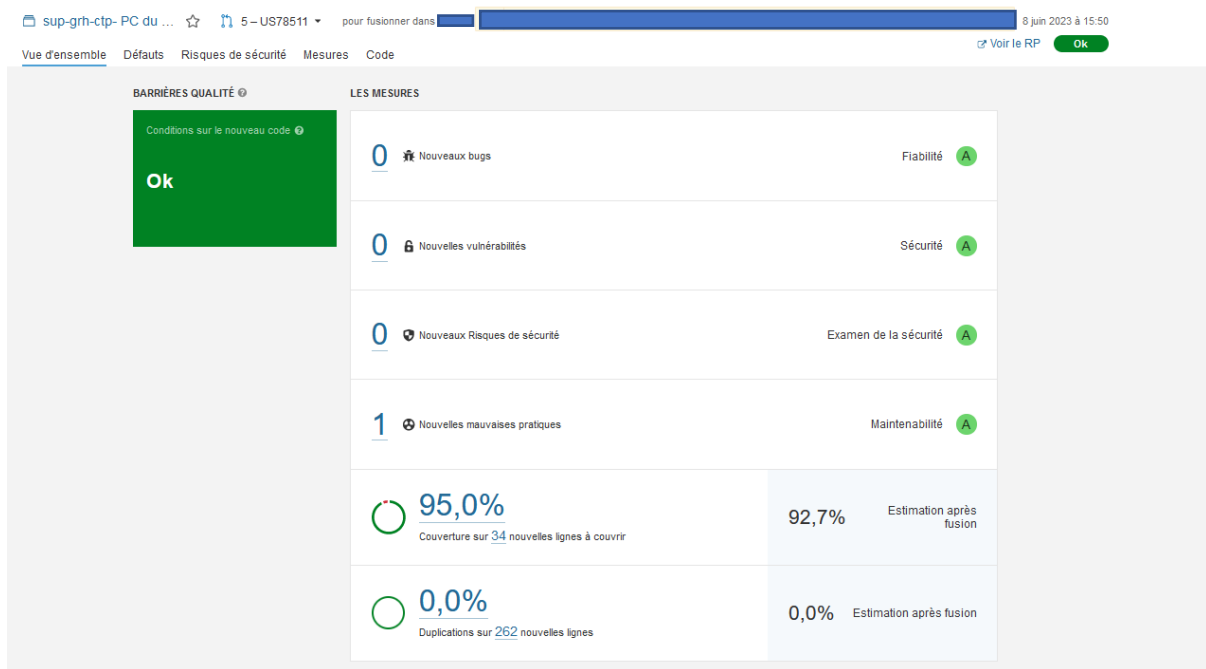
return params;
}

```

Annexe 44 : OWASP Top 10



Annexe 45 : Analyse SonarQube d'un nouveau code



Annexe 46 : Interface utilisateur de cyberwatch

