Optimisation de la Gestion des Plaintes Clients chez GE Healthcare : Développement et Implémentation d'IBTool

Livret Blanc de Master 1

Présenté par

Sana ALINIA

**Établissement:** L’École La Passerelle des Métiers du Numérique – La PMN

**Entreprise d’accueil**: GE Healthcare, 283 Rue de la Minière, 78530 Buc

Septembre 2023

Paris, France

# Remerciement

Je tiens à exprimer ma gratitude envers L'École La Passerelle des Métiers du Numérique - La PMN pour leur confiance et soutien dans mon parcours académique. Un grand merci à mon manager, Magali Wissocq, à l'équipe RH de GE Healthcare, ainsi qu'à mes collègues pour leurs précieux conseils durant mon alternance. Mes professeurs de la PMN ont également joué un rôle clé par leurs enseignements. Je suis reconnaissante envers le jury pour leur évaluation et mes proches pour leur soutien inconditionnel tout au long de ce projet.

# Résumé

La maintenance logicielle est une étape clé post-livraison dans le domaine du génie logiciel, impliquant la rectification des défauts et l'amélioration des performances. Pour GE Healthcare, gérer les problèmes d'Installation Base (IB) soulevés par les clients est un défi conséquent qui nécessite une surveillance attentive. Malgré l'existence d'outils commerciaux comme ClearQuest d'IBM pour cette gestion, ils présentent des limites en termes de sécurité, de coûts et d'adaptabilité aux besoins spécifiques du secteur.

Dans ce contexte, ce rapport décrit le développement d'IBTool, une solution interne créée pour optimiser le suivi et l'analyse des problèmes IB chez GE Healthcare. Réalisé en étroite collaboration avec les ingénieurs de GE, le projet englobe le développement de fonctionnalités frontend et backend, visant à améliorer substantiellement l'interface utilisateur et l'analyse des réclamations.

La phase récente du développement s'est concentrée sur l'amélioration des performances via la mise en cache et l'indexation de la base de données, et sur l'initiation de la migration vers un environnement React pour une meilleure réutilisabilité des composants et optimisation des performances. Parallèlement, le déploiement sur le cloud a été amorcé avec l'utilisation d'un éventail de services AWS, offrant plus de sécurité et de contrôle.

Pour les étapes à venir, il est prévu de compléter la migration vers React et d'approfondir l'intégration au cloud, incluant des optimisations de coûts et l'incorporation de fonctionnalités telles que le déploiement continu et une surveillance accrue.

# Table des matières

[Introduction 6](#_Toc145849839)

[Définition du problème et contributions 6](#_Toc145849840)

[Développement 7](#_Toc145849841)

[Outils et technologies utilisés 7](#_Toc145849842)

[Structure et pages de l'interface utilisateur d'IBTool 8](#_Toc145849843)

[Schéma de base de données 9](#_Toc145849844)

[Amélioration des performances 10](#_Toc145849845)

[Déploiement d'IBTool 11](#_Toc145849846)

[Déploiement sur cloud privé 11](#_Toc145849847)

[Déploiement sur AWS 11](#_Toc145849848)

[Déploiement d'IBTool sur Amazon EC2 12](#_Toc145849849)

[Déploiement via AWS EKS et EC2 13](#_Toc145849850)

[Discussion sur les orientations futures 13](#_Toc145849851)

[Déploiement Continu 13](#_Toc145849852)

[Absence d'un système d'authentification robuste 14](#_Toc145849853)

[Mise en cache avec AWS ElastiCache 14](#_Toc145849854)

[Fiabilité et Récupération en cas de Sinistre 14](#_Toc145849855)

[Test Pilote 14](#_Toc145849856)

[Conclusion 15](#_Toc145849857)

[Bibliographie 15](#_Toc145849858)

# Introduction

Dans le secteur de la santé dynamique et en constante évolution, il est primordial de disposer d'outils de gestion à la fois performants et sécurisés. Reconnaissant cela, GE Healthcare a entrepris le projet ambitieux de moderniser son ancien outil de gestion des réclamations afin d'améliorer divers aspects, y compris l'expérience utilisateur et la sécurité.

L'année précédente, j'ai grandement contribué à ce progrès en participant au lancement de la première version du nouvel outil, marquant ainsi le début d'une transformation radicale dans la gestion des réclamations de l'entreprise. Cette année, j'ai maintenu cette dynamique, notamment en orientant le projet vers une migration vers React pour une base de code plus propre et une expérience utilisateur optimisée.

Parallèlement, j'ai œuvré à l'amélioration substantielle des performances du logiciel, notamment en intégrant du caching via Redis pour accélérer les temps de réponse du système. Un autre volet crucial de mon travail a été la migration stratégique du projet vers AWS, initialement sur une instance EC2, puis en évoluant vers un déploiement plus sophistiqué en utilisant AWS EKS, apportant ainsi une sécurité renforcée et une gestion plus efficace des ressources.

Dans ce mémoire, je m'efforce de fournir une vue complète de cette transformation en cours, en exposant les différentes phases de développement et les avantages stratégiques et techniques qu'elles apportent à GE Healthcare. Je détaille également les interdépendances des services AWS utilisés, avec l'objectif ultime de garantir une transition réussie vers un système plus sécurisé et optimisé, prêt à répondre aux besoins évolutifs de l'entreprise.

## Définition du problème et contributions

La thèse se propose de traiter des carences significatives du précédent outil de gestion des plaintes utilisé par GE Healthcare, l’IBTool. Ces lacunes, engendrant des difficultés en termes de performances, sécurité des données et expérience utilisateur, sont principalement dues à une absence de système de mise en cache efficace et d’une stratégie de sauvegarde robuste.  
L’objectif est de concevoir une version optimisée de l’IBTool qui surmonte ces défaillances en utilisant les technologies contemporaines de containerisation et les services AWS. Cette démarche vise une amélioration substantielle de l’efficacité opérationnelle et de la gestion du système.

Les contributions de ce mémoire sont les suivantes :

**1. Cohérence et facilité d’utilisation**

Pour résoudre les problèmes d’intégration et d’utilisation liés à l’outil précédent, le nouvel IBTool offre une meilleure visibilité et gestion des dossiers de sécurité/réglementaires et une interface utilisateur améliorée, garantissant ainsi une productivité et une compréhension de la qualité des produits accrues.

**2. Conteneurisation et déploiement sur AWS**

Le déploiement envisagé est fait en deux étapes : une initialisation via Docker-Compose, puis une migration vers AWS EC2 et EKS, permettant une orchestration de conteneurs plus efficace. Cette stratégie favorise la scalabilité, la performance et la sécurité, facilitant ainsi la maintenance et l’évolution futures du système.

**3. Mise en œuvre du Caching**

L’intégration d’un système de mise en cache utilisant d’abord Redis, puis à terme Amazon ElasticCache, vise à optimiser les performances du système en offrant un accès rapide aux données fréquemment utilisées, améliorant ainsi l’expérience utilisateur.

**4. Stratégie de sauvegarde robuste**

Pour garantir la sécurité et l’intégrité des données, une stratégie de sauvegarde efficace est mis en place à l’aide d’AWS Backup. Ce système permet des récupérations de données rapides et une conformité réglementaire renforcée, offrant ainsi une tranquillité d’esprit accrue aux utilisateurs.

À travers cette thèse, nous visons à transformer l’IBTool en une solution plus robuste, flexible et efficace, prête à relever les défis futurs avec agilité et compétence, tout en améliorant significativement l’expérience utilisateur et la sécurité des données.

# Développement

## Outils et technologies utilisés

Dans cette section, les parties techniques du développement d'IBTool sont expliquées. Le développement comprend à la fois le front-end et le back-end et pour chaque partie, différentes techniques sont utilisées.

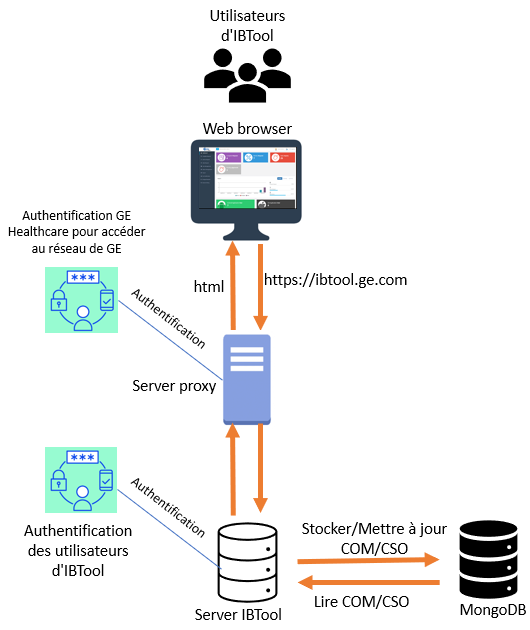


Figure 1 Accès à IBTool par les utilisateurs

Les utilisateurs d'IBTool utilisent un navigateur Web pour se connecter au service. Les utilisateurs devront être authentifiés pour accéder au réseau GE Healthcare, puis se connecter à IBTool à l'aide de leurs informations d'identification. La figure 1 illustre cette conception au niveau du système.

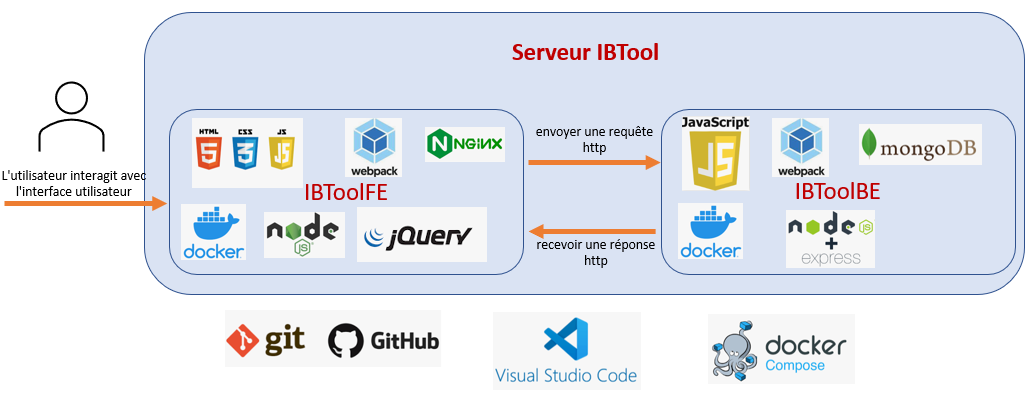


Figure 2: Composants IBTool et technologies de développement

Le serveur IBTool comprend à la fois des services frontend et backend. La figure 2 montre différents composants de l'IBTool, leur interaction et les différentes technologies utilisées dans le projet.

La partie frontend du projet traite de l'interface utilisateur et de l'envoi de certaines requêtes au backend afin de récupérer des données à afficher dans l'interface utilisateur.

Comme on peut le voir sur la Fig. 4, les outils et technologies suivants sont utilisés dans la partie frontend :

* HTML/CSS : pour la structure et le style des pages
* Javascript : comme langage de programmation frontal
* NodeJS et npm : pour exécuter javascript en dehors du navigateur et d'installation des librairies
* Devextreme [2] : une toolkit de javascript basée sur JQuery
* Webpack [3] : une librairie d'emballages
* Docker [4] : pour créer une image frontend
* NGINX [5] : pour créer un serveur local et lancer l'application
* Git/Github : pour la gestion des versions et revue de code
* VS code : éditeur de code

## Structure et pages de l'interface utilisateur d'IBTool

La structure des pages dans IBTool est illustrée à la figure 3. Une brève explication de chaque page sera donnée plus loin dans cette section.

Le backend se compose de deux parties principales : a) traiter les requêtes http reçues dans ses endpoints et b) se connecter à la base de données et effectuer des opérations CRUD. Les principales technologies utilisées dans la partie backend sont les suivantes :

* NodeJS
* Express
* MongoDB
* Webpack
* Git/Github
* Docker

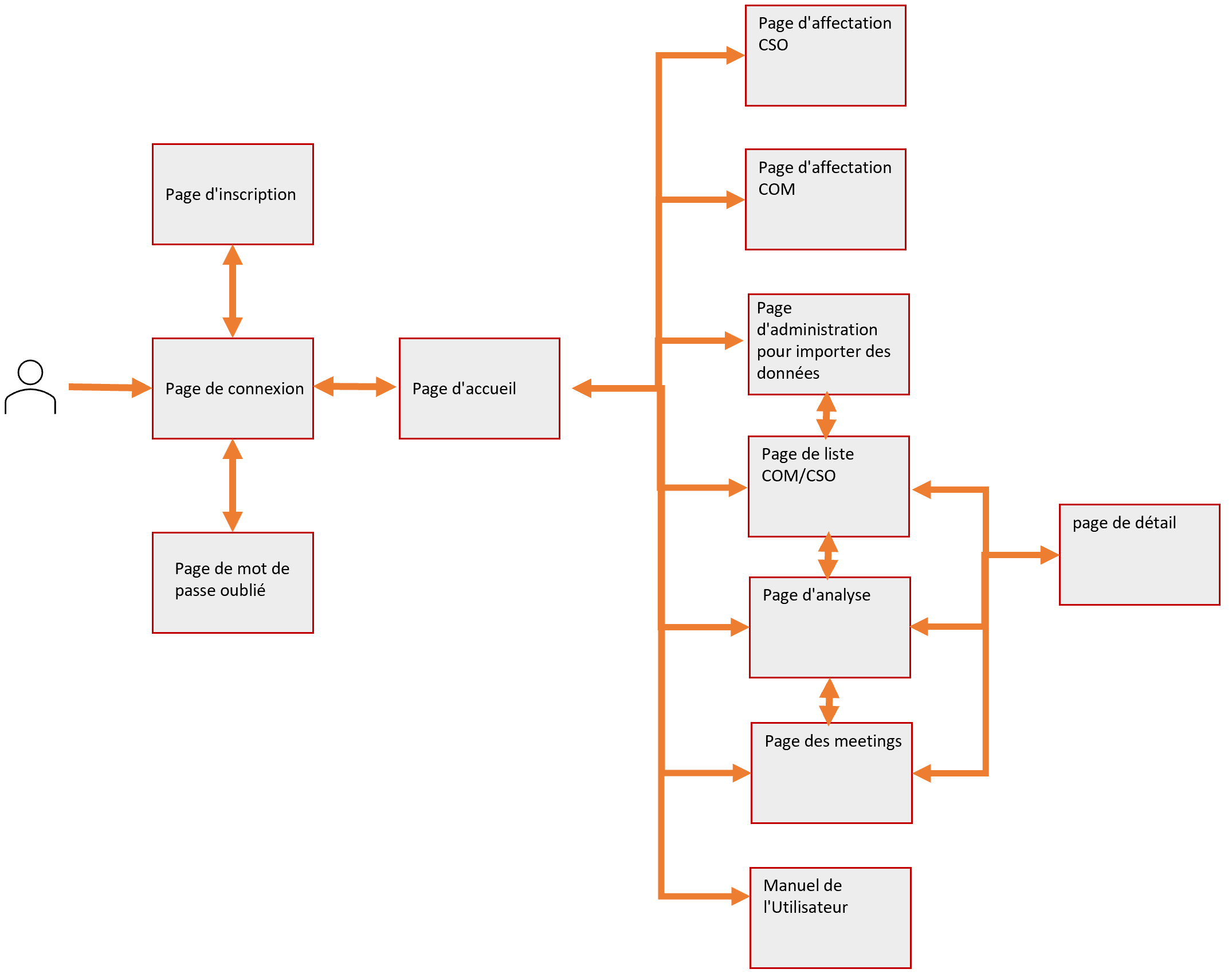


Figure 3: Pages de IBTool. Certains liens ne sont pas affichés pour plus de simplicité

J'ai également utilisé différentes libraires de npm pour encoder/déchiffrer les informations d'identification des utilisateurs, se connecter à la base de données, builder le projet, convertir json en csv, etc.

## Schéma de base de données

Pour la base de données, j'ai choisi MongoDB (une base de données NoSQL) pour les raisons suivantes : Tout d'abord, les données que nous devons stocker (par exemple, la liste des COM/CSO) contiennent de nombreux champs, mais certains ou plusieurs d'entre eux peuvent être vides. L'utilisation d'une base de données de relations comme MySql peut gaspiller de l'espace tandis qu'une base de données basée sur des documents comme MongoDB ne stocke que les informations remplies et économise de l'espace. Deuxièmement, MongoDB est évolutif et plus rapide. Troisièmement, il existe des outils et des libraires javascript qui soutiennent MongoDB et facilitent le development. Le schéma de la base de données est illustré à la figure 4.

De nombreux champs de la collection Record ne sont pas affichés en raison d'un manque d'espace. Les collections "User" et "Record" sont connectées à l'aide de COM-ID / CSO-ID. La collection "Shared" stocke toutes les informations qui ne sont pas destinées à un utilisateur spécifique (par exemple, la dernière mise à jour COM). La collection "Valid SSO" stocke la liste des utilisateurs SSO autorisés à s'enregistrer et à utiliser IBTool.

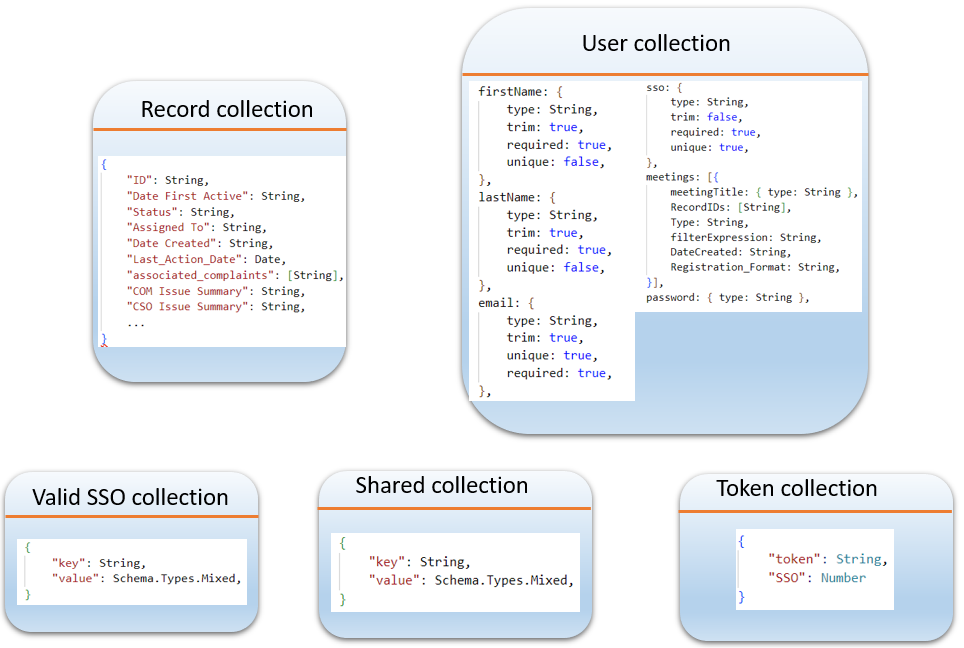


Figure 4: Schéma de base de données pour IBTool.

## Amélioration des performances

J'ai utilisé deux stratégies pour améliorer les performances de l'application : utiliser le cache et migrer vers React.

Afin d'optimiser la réactivité de l'application, j'ai intégré des mécanismes de mise en cache côté client et backend avec l'utilisation de Redis. La mise en cache côté client réduit les requêtes inutiles au serveur en stockant les données courantes dans la mémoire du navigateur. Du côté backend, Redis est utilisé pour stocker les données fréquentes de MongoDB. Cette stratégie diminue la sollicitation du serveur, améliorant ainsi les performances de l'IBTool.

La transition vers React pour le front-end est en cours, offrant une meilleure performance et une réutilisabilité du code. React optimise le temps de chargement des pages et la fluidité de l'interface utilisateur grâce à sa méthode de rendu virtuel, tout en favorisant une architecture modulaire pour le code. Sa communauté active et l'écosystème fournissent des ressources précieuses pour le développement. Toutefois, cette migration, bien que bénéfique, est un processus minutieux qui sera complété dans des travaux ultérieurs pour garantir la stabilité de l'application.

# Déploiement d'IBTool

## Déploiement sur cloud privé

Dans cette méthode de déploiement, IBTool est déployé sur les serveurs de l'entreprise sur son propre cloud privé.

Pour déployer IBTool sur le cloud privé de GE Healthcare, il suffit de déployer des images docker sur un serveur de l'entreprise et de partager l'adresse IP et le port du service frontal aux utilisateurs. IBTool se compose de deux repsitories Git différents pour le frontend et le backend, et a également besoin de MongoDB.

Avec l'aide de docker, nous pouvons créer des images docker pour les parties frontend et backend. Et nous n'avons pas besoin d'installer MongoDB. Au lieu de cela, nous pouvons utiliser une image docker officielle de MongoDB. Plus important encore, nous n'avons pas à nous soucier de la machine de déploiement. Enfin, nous pouvons utiliser docker-compose pour connecter les services frontend et backend entre eux et à la base de données et lancer l'IBTool par une seule commande de docker-compose.

## Déploiement sur AWS

Le déploiement d'IBTool sur un cloud public comme Amazon AWS présente des avantages :

* **Scalabilité et Flexibilité** : AWS facilite la mise à l'échelle d'IBTool grâce à des services variés comme EC2, Lambda, ECS et EKS.
* **Rentabilité** : La tarification d'AWS est basée sur l'utilisation, ce qui élimine les coûts initiaux élevés et permet une adaptation selon les besoins du projet en croissance.
* **Portée mondiale** : L'infrastructure d'AWS permet un déploiement mondial, garantissant une faible latence et le respect des réglementations locales sur la souveraineté des données.
* **Sécurité et Conformité** : AWS est reconnu pour sa sécurité stricte et sa conformité à différentes normes industrielles, offrant des outils intégrés pour gérer l'accès aux ressources.
* **Déploiement et Maintenance Simplifiés** : Des services comme Elastic Beanstalk et OpsWorks facilitent le déploiement, permettant à l'équipe de se concentrer sur le développement de l'application.
* **Surveillance et Journalisation Intégrées** : Avec CloudWatch et CloudTrail, AWS offre une surveillance centralisée, aidant à l'analyse et au dépannage d'IBTool.

Top of Form

AWS a été privilégié pour sa maturité, la diversité de ses services, son support, sa flexibilité tarifaire, ses intégrations et sa sécurité, se distinguant d'autres fournisseurs comme Azure et Google Cloud. Fourni par Amazon, AWS offre des services de cloud variés, dont la puissance de calcul, le stockage et le réseau. Des composants clés incluent les serveurs virtuels, le stockage, les bases de données, le réseau et la sécurité. AWS offre une scalabilité, une rentabilité, une portée mondiale, une sécurité, et facilite le déploiement et la maintenance pour des projets comme IBTool.

### Déploiement d'IBTool sur Amazon EC2

Le déploiement d'IBTool sur Amazon EC2 a été orchestré avec une attention particulière aux détails pour garantir à la fois la performance et la sécurité. Voici une vue d'ensemble du processus, étape par étape, pour déployer l'IBTool sur Amazon EC2:

**Préparation de l'environnement EC2**: Avant de commencer, il était essentiel de préparer l'environnement EC2. Les instances EC2 ont été lancées, en veillant à ce que chacune d'elles corresponde aux besoins spécifiques de chaque composant d'IBTool, à savoir le frontend, le backend, et la base de données MongoDB. L'AMI (Amazon Machine Image) sélectionnée possédait Docker et Docker Compose pré-installés, facilitant ainsi la mise en place et le déploiement des conteneurs nécessaires.

**IP Elastic :** L'adresse IP élastique configurée à l'aide de la console AWS et associée aux instances EC2 d'IBTool. Cela donne une adresse IP fixe, même si l'instance est arrêtée ou terminée, garantissant ainsi une connectivité constante et évitant les changements fréquents d'adresse IP.

**Déploiement des conteneurs Docker**: Une fois les instances EC2 opérationnelles, les fichiers de l'application, notamment Dockerfiles et docker-compose.yaml, ont été transférés sur les instances pertinentes. Sur chaque instance, ces fichiers ont été utilisés pour construire et exécuter les conteneurs Docker, permettant ainsi à chaque composant de l'IBTool de fonctionner dans son propre environnement isolé.

**Configuration de la base de données MongoDB**: La base de données, un élément central de l'IBTool, a été soigneusement configurée. MongoDB a été initialisé avec des variables d'environnement spécifiques et une stratégie de sauvegarde a été mise en place pour garantir l'intégrité et la disponibilité des données.

**Équilibrage de charge et surveillance**: Vu l'importance de la disponibilité et de la fiabilité, un Équilibreur de Charge Élastique (ELB) a été intégré, dirigeant le trafic vers les instances EC2 appropriées. De plus, pour surveiller l'état du système et assurer sa stabilité, AWS CloudWatch a été utilisé pour recueillir des journaux d'activité et AWS X-Ray a été envisagé pour analyser les requêtes des utilisateurs.

**Gestion des coûts et des ressources**: Afin de gérer efficacement les coûts, des alertes de facturation ont été mises en place via les Budgets AWS. Par ailleurs, une revue régulière des ressources, comme les instances EC2 non utilisées, a été établie pour éviter tout frais superflu.

En somme, le déploiement d'IBTool sur Amazon EC2 a été une tâche complexe qui a nécessité une approche méthodique. Chaque étape a été soigneusement planifiée et exécutée pour garantir une application performante, sécurisée et économiquement viable.

### Déploiement via AWS EKS et EC2

En tant que méthode plus avancée, j'ai déployé IBTool à l'aide d'AWS EKS, dans le but d'améliorer la sécurité et l'évolutivité en utilisant des instances EC2 au lieu d'AWS Fargate, en raison de ses limites et de ses coûts élevés.

#### Étapes de Déploiement

Le processus se déroule en plusieurs étapes :

1. **Préparation des Ressources Kubernetes** :
   * Définition des ressources dans des fichiers yaml.
   * Création de trois déploiements Kubernetes pour gérer le frontend, le backend, et MongoDB.
   * Mise en place des ressources additionnelles comme le stockage secret et persistant, et les services pour chaque pod.
2. **Création du Cluster Kubernetes** :
   * Réalisée via la console AWS pour un contrôle optimal de la configuration.
   * Configuration des VPC et des subnets, suivi de l’établissement des groupes de nœuds.
3. **Provisionnement des Instances EC2** :
   * Similaire à la méthode décrite dans la section précédente.
4. **Mise en place des Groupes de Nœuds** :
   * Établissement de groupes distincts (public et privé) pour améliorer la sécurité.
5. **Déploiement de l’Application** :
   * Utilisation de kubectl pour déployer les ressources Kubernetes selon les configurations précédemment définies dans les fichiers yaml.
   * Installation et configuration de kubectl pour la communication avec le cluster.

#### Résultat

Après avoir suivi ces étapes détaillées, l’application IBTool est déployée avec succès, garantissant une meilleure sécurité et flexibilité grâce à l’usage de AWS EKS et des instances EC2.

# Discussion sur les orientations futures

## Déploiement Continu

Actuellement, l'IBTool manque de système CI/CD, rendant le déploiement de nouvelles fonctionnalités lent et vulnérable à des erreurs. La solution est d'intégrer un pipeline CI/CD avec AWS CodePipeline et AWS CodeDeploy, facilitant ainsi l'intégration et le déploiement continus. Cela comprend l'utilisation d'AWS CodeStar pour une gestion centralisée, l'automatisation des tests pour éviter les bugs dans la production, et des itérations rapides pour une amélioration continue. Ce système surpasserait des outils tels que Jenkins en étendant l'intégration continue avec un déploiement automatique des modifications dans des environnements de test et production.

## Absence d'un système d'authentification robuste

La version précédente de l'IBTool souffrait d'un système d'authentification défaillant. La réponse est l'intégration d'Amazon Cognito pour contrôler l'accès sécurisé, caractérisé par une configuration initiale distincte des utilisateurs, une intégration avec IBTool pour un processus de connexion sécurisé, et des politiques de sécurité robustes pour une transmission sécurisée des mots de passe. Cette démarche garantirait un environnement sécurisé tout en optimisant l'expérience utilisateur.

## Mise en cache avec AWS ElastiCache

Pour accélérer l'accès aux données dans l'IBTool, AWS ElastiCache sera intégré pour gérer la mise en cache des données fréquemment consultées, réduisant ainsi les temps de chargement et la charge sur la base de données tout en améliorant l'expérience utilisateur. Cette intégration comprendrait une analyse des données pour identifier les éléments à cacher et une surveillance pour assurer l'efficacité du système.

## Fiabilité et Récupération en cas de Sinistre

L'ancienne installation de l'IBTool n'avait pas de mécanismes de sauvegarde et de récupération fiables, mettant en danger la continuité du service. La nouvelle approche utilisera Amazon S3 pour des sauvegardes régulières et Amazon Glacier pour l'archivage à long terme. En outre, un plan bien structuré pour la récupération après sinistre sera développé. Pour augmenter la disponibilité, un déploiement Multi-AZ sera mis en œuvre avec une répartition de charge efficace et une surveillance proactive de l'infrastructure.

## Test Pilote

Actuellement, l'IBTool est déployé sans tests pilotes suffisamment rigoureux, risquant des bugs dans l'environnement de production. Pour y remédier, une phase de tests pilotes sera introduite, comprenant l'élaboration d'un plan de test, la formation d'une équipe de testeurs diversifiée, la création d'un environnement de test isolé, et l'intégration d'outils de suivi des bugs. Suite à l'exécution des tests, une analyse complète sera menée pour identifier et corriger les problèmes avant le déploiement final, assurant ainsi un système robuste et fiable.

# Conclusion

Au terme de notre étude pour perfectionner l'outil de gestion des plaintes chez GE Healthcare, nous constatons des avancées notables et cruciales visant une efficacité et une résilience augmentées dans un environnement technologique constamment en évolution.Les principales évolutions comprennent l'intégration d'une solution de mise en cache puissante grâce à l'adoption de Redis et la future utilisation d'ElasticCache, ainsi que le début de la migration vers une application React pour optimiser l'expérience utilisateur. L'initiation du déploiement de l'IBTool sur une plateforme AWS EC2 plus flexible et dynamique s'est avérée être un choix judicieux, une démarche amplifiée en embrassant les services AWS EKS pour une meilleure sécurité et contrôle.

L'adoption d'un environnement containerisé et d'une stratégie de sauvegarde solide grâce à AWS Backup renforce la fiabilité du système, promettant une maintenance facilitée et une récupération rapide en cas de panne. Ces innovations, couplées avec des méthodes de déploiement modernes, préparent l'IBTool à répondre aux défis présents et futurs, garantissant sa pertinence à long terme.

En regardant vers l'avenir, les opportunités pour optimiser davantage sont considérables, incluant la considération pour le déploiement continu et l'optimisation des coûts. La réalisation d'un projet pilote est également prévue pour assurer une mise en œuvre en douceur à grande échelle.

En synthèse, cette refonte majeure de l'IBTool ne symbolise pas seulement une avancée en termes d'innovation et d'efficacité; elle pave également la voie pour un développement agile, anticipant les défis futurs avec une base solide et optimiste, prête à évoluer au gré des besoins futurs chez GE Healthcare.

# Bibliographie

1. <https://www.gehealthcare.com/>
2. <https://js.devexpress.com/>
3. <https://webpack.js.org/>
4. <https://www.docker.com/>
5. <https://www.nginx.com/>