



Université de la Manouba
École Nationale des Sciences de l'Informatique



RAPPORT DU PROJET DE STAGE D'INGÉNIEUR

Sujet : Analyse de fichiers logs

Auteur :

M. Youssef CHRAIEF

Encadrant :

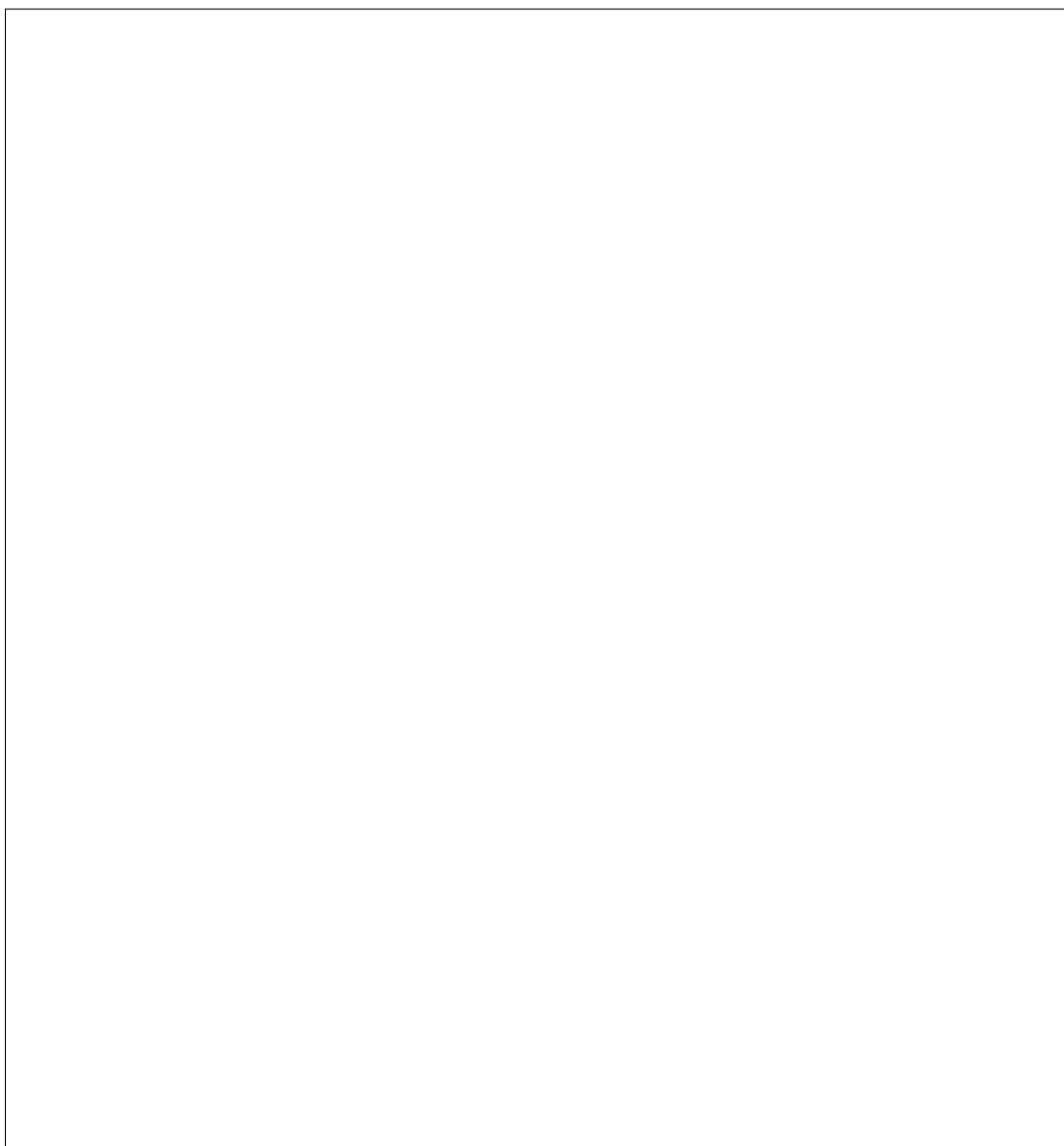
M. SAIFALLAH JRAD

Entreprise :



Année Universitaire : 2022 /2023

Appréciations et signature de l'encadrant

A large, empty rectangular box with a thin black border, intended for the appraiser's and signatory's comments and signature.

Remerciements

Avant de commencer la présentation de notre travail, nous tenons à remercier tous ceux et celles qui nous ont apporté leur aide, leurs encouragements et leur soutien pour nous permettre de mener à bien ce projet, et nos remerciements s'adressent en particulier à Monsieur Saifallah Jrad pour son assistance, sa disponibilité et ses précieux conseils durant toute cette période.

Nous exprimons, également, tout notre respect et notre haute considération aux membres du jury qui ont accepté d'évaluer ce travail en espérant qu'ils trouvent notre rapport clair et exprimant toute la motivation qu'ils attendent.

Finalement, nos remerciements s'adressent à tous les enseignants de l'école nationale des sciences de l'informatique qui nous ont fait bénéficier de leurs connaissances et leur savoir-faire en sciences de l'informatique.

Table des matières

Introduction	1
1 Présentation générale du projet	2
1.1 Présentation de l'entreprise	2
1.1.1 Activités et produits de Vermeg	3
1.1.2 Organisation de Vermeg	4
1.2 Présentation du projet	4
1.2.1 Cadre du projet	4
1.2.2 Contexte	4
1.2.3 Critique de l'existant	5
1.2.4 Solution proposée	5
1.3 Méthodologie adoptée	6
1.3.1 Choix de méthodologie	6
2 Analyse et spécification des besoins	9
2.1 Contexte Statique	9
2.1.1 Notions de Base	9
2.1.1.1 Soliam	9
2.1.1.2 Fichier log	9
2.2 Branche Technique	10
2.2.1 Etude du progiciel Soliam	10
2.2.1.1 Architecture de Soliam	10
2.2.1.2 Contraintes techniques	11
2.3 Choix technologiques	11
2.3.1 Choix de Logstash et Elasticsearch	11
2.3.2 Choix de Talend	12
2.3.3 Choix de Power BI	12
2.3.4 Choix du serveur Tomcat	12
2.4 Branche Fonctionnelle	13
2.4.1 Identification des acteurs	13
2.4.2 Besoins fonctionnels	13

2.4.3	Besoins non fonctionnels	13
2.4.4	Besoins de domaine	14
2.5	Spécification des besoins	14
2.5.1	Diagramme de cas d'utilisation global	14
2.5.2	Cas d'utilisation détaillés	16
2.5.2.1	Cas d'utilisation « Configurer les flux de collecte automatisés »	16
2.5.2.2	Cas d'utilisation « Intégrer les rapport dans Soliam » . . .	17
2.5.2.3	Cas d'utilisation « Appliquer des transformations ETL » .	18
3	Conception	21
3.1	Conception globale	21
3.1.1	Architecture physique	21
3.1.2	Architecture logicielle	22
3.1.2.1	Couche de Présentation :	22
3.1.2.2	Couche Logique :	22
3.1.2.3	Couche de Données :	23
3.1.2.4	Intégration :	23
3.2	Aspect statique du système	23
3.2.1	Diagramme de Paquetages	23
3.2.2	Déploiement de l'Application	24
3.2.2.1	Description du Déploiement de l'Application	24
4	Réalisation	25
4.1	Environnement de travail	25
4.1.1	Environnement matériel	25
4.1.2	Environnement logiciel	25
4.1.2.1	Technologies utilisées	25
4.1.2.2	Environnement de développement intégré	27
4.2	Aperçu des interfaces	27
4.2.1	Interfaces de Collecte	27
4.2.1.1	Configuration des Sources	27
4.2.1.2	Pipelines de Collecte	29
4.2.2	Interfaces de Transformation ETL	30
4.2.2.1	Création de Jobs ETL	30
4.2.2.2	Suppression des doublons	30
4.2.2.3	Analyse des performances et du temps d'exécution	31
4.2.3	Interfaces de Stockage	32
4.2.3.1	Tables Oracle	32
4.2.4	Interfaces d'Intégration	33
4.2.4.1	Connexions à Power BI	33

4.2.4.2	Affichage des Rapports	34
Conclusion		35
Webographie		36

Table des figures

1.1	Schéma explicatif de la solution proposée.	6
1.2	le processus de développement 2TUP.	7
2.1	Architecture Monolithique.	11
2.2	Diagramme de cas d'utilisation global.	15
2.3	Diagramme de cas d'utilisation « Configurer les flux de collecte ».	16
2.4	Diagramme de séquence « Configurer les flux de collecte ».	17
2.5	Diagramme de cas d'utilisation « Intégrer les rapport dans Soliam ».	17
2.6	Diagramme de séquence « Configurer les flux de collecte ».	18
2.7	Diagramme de cas d'utilisation « Appliquer des transformations ETL ».	19
2.8	Diagramme de séquence « Appliquer des transformations ETL ».	20
3.1	Architecture physique de l'application.	22
3.2	Diagramme de paquetages.	23
4.1	Script logback-spring.xml.	28
4.2	Dossier logback des logs.	28
4.3	Serveur Logstash.	29
4.4	Serveur Logstash.	29
4.5	Interface Talend.	30
4.6	Job Talend.	31
4.7	Temps d'execution	32
4.8	Table des logs	33
4.9	configuration d'un rapport Power BI	33
4.10	un rapport Power BI	34

Liste des tableaux

2.1	Cas d'utilisation : Configuration des flux de collecte	16
2.2	Cas d'utilisation : Intégration des rapports Power BI dans Soliam	18
2.3	Cas d'utilisation : Appliquer des transformations ETL	19

Introduction

Les fichiers logs sont le pouls de toute infrastructure informatique moderne. Ils contiennent une mine d'informations précieuses sur le fonctionnement des systèmes, les performances, les erreurs et les incidents. Cependant, la gestion de ces fichiers logs, en particulier lorsque leur taille devient considérable, peut rapidement devenir une tâche complexe et chronophage.

Dans un monde où la rapidité de détection et la réactivité aux problèmes sont cruciales, il est impératif de disposer d'une solution efficace pour collecter, nettoyer, transformer et analyser ces fichiers logs. C'est précisément l'objectif de notre projet : mettre en place une solution complète de gestion de fichiers logs qui permettra une gestion plus intelligente et proactive de ces données critiques.

Ce rapport documente en détail chaque étape de notre projet, depuis la collecte initiale des fichiers logs jusqu'à la création de tableaux de bord interactifs pour une visualisation claire des données. Nous avons opté pour une approche intégrée, utilisant des technologies telles que Logstash, Elasticsearch, Talend, Oracle, Power BI et Tomcat pour créer une solution robuste et polyvalente.

Dans la suite, nous détaillerons toutes les phases de développement de cette solution, il s'étale sur quatre chapitres organisés comme suit :

- Le premier chapitre est réservé à la présentation générale du cadre du projet et une étude des méthodologies de travail.
- Quant au deuxième chapitre, il est dédié à l'analyse et la spécification des besoins.
- Le troisième chapitre est consacré à l'architecture physique et logique adoptées. Ensuite nous allons effectuer une étude statique du système en schématisant le diagrammes de paquets.
- Le dernier chapitre contient les choix techniques utilisés ainsi que des interfaces de la solution qui décrivent les scénarios.
- À la fin, ce rapport sera clôturé par une conclusion générale qui résume le travail effectué.

Chapitre 1

Présentation générale du projet

Introduction

Le présent chapitre a pour but de définir le contexte général afin de situer le projet dans son environnement organisationnel, méthodologique et stratégique. Il présente les principales problématiques qui ont appelé à proposer une solution.

1.1 Présentation de l'entreprise

Le projet a été réalisé au sein de la société VERMEG située aux Berges du lac à Tunis. C'est une entreprise « off-shore », fondée en 2002 comme entreprise indépendante de la société mère BFI, établie depuis 1994. VERMEG est dotée d'une ingénierie experte en monétique et en édition de logiciels bancaires et financiers pour la gestion des titres et des capitaux. Depuis sa création, elle n'a pas cessé de développer son expertise en finance pour offrir une large gamme de logiciels innovants pour les middle et back-offices des institutions financières concernées par le traitement des titres. Son atout majeur est sa capacité de capitalisation d'expertise utilisée à des fins d'innovation, ce qui explique l'utilisation de ses produits par de grandes institutions financières dans quinze pays à travers trois continents.

VERMEG a plusieurs clients de renommée à travers le monde englobant les banques, les gestionnaires de fonds, les assurances telles que le Banco Santander, la Banque de France, BNPParibas, HSBC, Nordea, Société Générale...

VERMEG compte aujourd'hui plus de 700 collaborateurs, principalement des ingénieurs et des consultants. Ses bureaux sont répartis à Bruxelles, Paris, Luxembourg, Amsterdam et Tunis. Leur mission est d'ouvrir des services spécialisés dans le domaine de la finance, de conseiller les clients et d'être à l'écoute de leurs besoins afin de les satisfaire dans la mesure du possible.

Dans le cadre de son expansion, VERMEG a pris le contrôle de BSB, une entreprise belge spécialisée en édition de logiciels d'assurances et de gestion d'actifs financiers. Elle compte parmi ses clients les plus grandes banques, assurances et institutions financières de renommée mondiale : Santander, Société Générale, BNP Paribas, Crédit Agricole, Allianz, Barclays, Generali, Aviva, Zurich, Banque de France, Banque d'Angleterre, Nordea, National Bank of Abu Dhabi, etc.

1.1.1 Activités et produits de Vermeg

L'activité de Vermeg s'articule principalement autour de quatre axes :

- **Assurance** : couverture d'assurance individuelle ou de groupe pour l'épargne et la santé.
- **Gestion de richesse et d'actifs** : : gestion de position et des décisions financières dédiée aux besoins spécifiques de gestion de patrimoine.
- **Services financier digitaux** : numérisation des processus financier et analyse des données sensibles.

Vermeg offre pour ses clients une large gamme de logiciels spécialisés dans le domaine financier, parmi lesquels nous citons la suite Megara , Omega FA, Solife et Soliam , une plateforme de développement Palmyra ainsi que le logiciel d'assurance vie Vermeg Life.

- **MEGARA (Securities processing)** : La suite MEGARA est une plateforme modulaire pour le traitement des titres destinée essentiellement aux institutions financières qui propose un nombre de modules pouvant être implémentés séparément.
- **Omega FA** : Omega FA est une application orientée métier, fournissant les fonctionnalités Front-office, Middle-office et Back-office qui sont destinées aux gestionnaires d'actifs. Elle est conçue aussi bien pour s'intégrer facilement avec les systèmes d'informations des clients que pour assurer une meilleure productivité. Omega FA est compatible avec l'environnement Windows.
- **SOLIFE** : Une solution d'administration de polices d'assurance vie et un portail web à destination des clients finaux/brokers.
- **SOLIAM** : est une solution de gestion de portefeuilles pour les gestionnaires d'actifs institutionnels de fortune. Ayant une présence internationale en Belgique, France, Irlande, Luxembourg, Pays-Bas, Suisse, Royaume-Uni, Tunisie et des clients dans plus de 23 pays, VermegLife et Soliam sont considérés comme les deux produits phares de Vermeg BSB.
- **PALMYRA** : est un Framework JEE compatible SOA (architecture orientée service) qui incorpore des composants ainsi que des services web réutilisables. L'implémentation de services web réutilisables permet de développer des logiciels de meilleure qualité en temps réduit dont l'architecture est basée sur un client léger.

- **Vermeg Life** : C'est un progiciel d'administration de police d'assurance-vie. Il offre la gestion du cycle de vie du contrat depuis la souscription, en passant par la création jusqu'à la clôture. La première version était développée en 2005. D'un point de vue technologique, Vermeg Life utilise une architecture orientée services (SOA) et est développé en technologie Java.

1.1.2 Organisation de Vermeg

Vermeg est composé de quatre unités appelées Business Unit (BU) qui sont les suivantes :

- **Pension and Insurance** : c'est l'unité chargée de développer la solution SoLife pour la gestion des polices d'assurance.
- **Wealth and Asset Management** : C'est l'unité chargée de développer la solution SOLIAM pour la gestion des portefeuilles des gestionnaires d'actifs institutionnels de fortune.
- **Financial Markets and Securities Services** : Il s'agit de l'unité qui développe la solution Megara dédiée au traitement des titres.
- **Digital Financial Services** : Il s'agit de l'unité qui développe la solution Megara dédiée au traitement des titres.

1.2 Présentation du projet

Dans cette section, le projet sera mis au départ dans son cadre général, ensuite nous allons déposer la problématique ainsi que le travail demandé.

1.2.1 Cadre du projet

Ce projet rentre dans le cadre d'un projet d'été, effectué durant deux mois au regard d'un stage d'Ingénieur. Il est intitulé « Analyse de fichiers logs de la couche publique de Soliam ».

1.2.2 Contexte

Soliam est une solution de gestion d'actifs et de patrimoines. Suite à l'augmentation du nombre des clientèles de Soliam, Vermeg cherche à optimiser encore plus le temps de réponse de plusieurs processus de cette solution pour satisfaire les besoins de ses clients. Plusieurs processus de la solution Soliam prennent plus de temps par rapport aux exigences des clients, pour cette raison Vermeg cherche à améliorer la qualité de son produit en termes de performance pour fidéliser ses clientèles.

La quantité de fichiers de logs (fichiers journaux) générées a considérablement augmenté, atteignant des niveaux qui rendent difficile leur gestion manuelle, cela engendre une inefficacité qui dépasse la capacité de l'entreprise à réagir rapidement aux problèmes, à prendre des décisions éclairées et à optimiser ses performances opérationnelles. Dans ce contexte, ce projet vise à mettre en place une solution complète, allant de la collecte initiale des fichiers journaux à la création de tableaux de bord interactifs, afin d'améliorer la gestion des données de journal, d'optimiser les processus décisionnels et d'accroître l'efficacité opérationnelle globale de l'entreprise.

1.2.3 Critique de l'existant

Actuellement, l'entreprise se trouve dans une situation où la gestion des fichiers journaux repose entièrement sur des processus manuels. Les équipes responsables de l'analyse des fichiers journaux doivent consacrer un temps considérable à la collecte, à la vérification et à l'interprétation des données de manière manuelle.

Cette approche présente plusieurs inconvénients notables :

- Entraîner des retards dans la détection et la résolution des problèmes potentiels.
- Absence d'automatisation dans la collecte et la préparation des données peut donner lieu à des erreurs humaines, ce qui affecte la fiabilité des analyses effectuées.
- Imposible d'établir une vue d'ensemble consolidée de la performance du système sur le long terme
- Absence de rapports pertinents qui décrivent les performances opérationnelles et identifient les tendances, les anomalies et les opportunités d'amélioration.

Pour maintenir sa compétitivité dans un environnement en constante évolution, l'entreprise doit passer d'une approche manuelle fragmentée à une solution moderne et automatisée qui non seulement simplifiera la gestion des fichiers journaux, mais permettra également une analyse approfondie et une prise de décision éclairée grâce à des rapports et des tableaux de bord interactifs.

1.2.4 Solution proposée

Afin de remédier aux problèmes précédemment cités, une solution complète et intégrée est proposée. Cette solution vise à automatiser la collecte, le nettoyage, la transformation et l'analyse des données de journal, tout en fournissant des rapports et des tableaux de bord interactifs pour une prise de décision éclairée.

Dans le cadre de cette solution, nous comptons sur l'utilisation d'outils avancés tels que Logstash pour la collecte initiale des fichiers journaux, Talend pour les étapes de nettoyage et de transformation ETL, Power BI pour la création de rapports visuels interactifs, et l'intégration de ces éléments au sein d'une application web Java déployée sur le serveur Tomcat.

Cette solution est illustrée dans la figure 1.1.

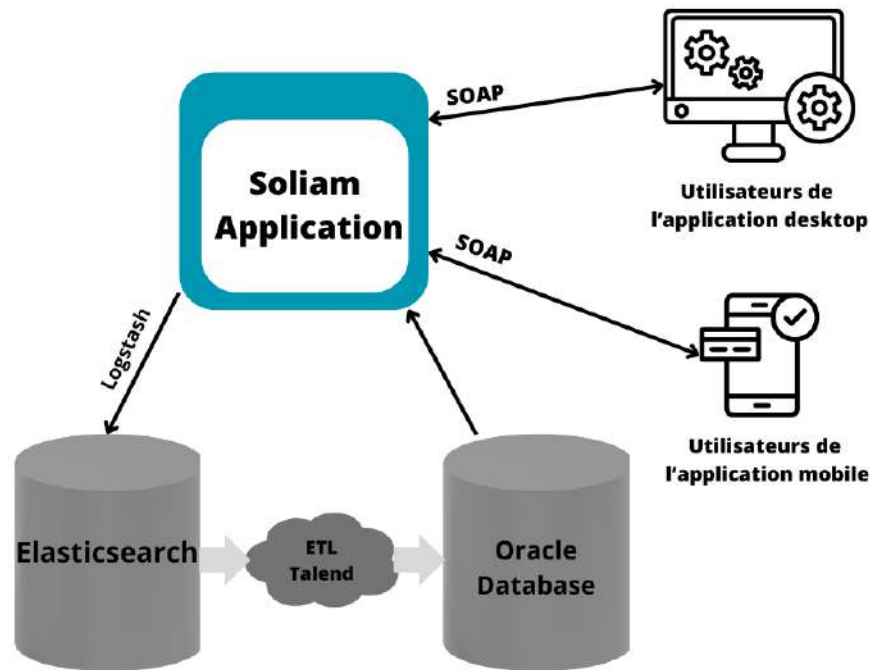


FIGURE 1.1 – Schéma explicatif de la solution proposée.

Ce projet vise à améliorer le temps de réparation des bugs des services web Soliam et à générer les rapports et les tableaux de bord afin de détecter les tendances, analyser les anomalies et de prendre des mesures relatives pour améliorer les performances à long terme.

1.3 Méthodologie adoptée

Pour pouvoir entamer le développement du projet, il est essentiel de sélectionner une méthodologie appropriée pour diriger efficacement notre projet, garantir un efficace processus de développement et assurer la réalisation optimale en matière de qualité et de productivité.

1.3.1 Choix de méthodologie

Suite à l'étude et à la comparaison des principaux processus de développement et afin de contrôler les risques et de mener à bon terme notre projet, vu sa complexité, nous avons opté pour le processus 2TUP pour plusieurs raisons :

- Le projet représente une certaine complexité au niveau technique d'où la nécessité de prévoir une phase pour étudier toutes les exigences techniques qu'il faut prendre en considération avant d'attaquer les spécifications fonctionnelles.
- A un moment donné, une liaison entre le modèle fonctionnel et le modèle technique doit être faite pour vérifier l'adaptation de l'architecture technique avec les besoins de l'utilisateur. Se basant sur un cycle en Y, la méthode 2TUP est la plus adéquate pour le contexte de ce projet, surtout qu'il ne représente pas une grande complexité au niveau fonctionnel.

La figure 1.2 permet d'exprimer toutes les étapes nécessaires pour la réalisation d'un projet.

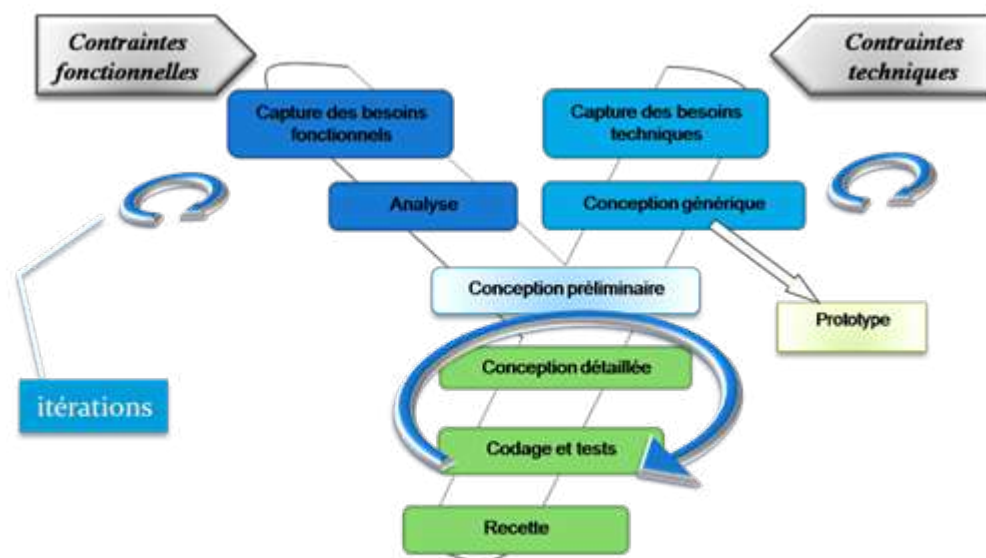


FIGURE 1.2 – le processus de développement 2TUP.

En se basant sur un cycle en Y, la méthode 2TUP est la plus adéquate pour le contexte de ce projet. Le projet à réaliser dans le cadre de ce stage est de nature assez spéciale puisqu'il nécessite l'adéquation entre les exigences techniques et les besoins en termes de fonctionnalités chez son utilisateur.

La méthodologie 2TUP est un processus de développement unifié qui préconise un cycle de développement en Y et qui dissocie les aspects fonctionnels de l'application des aspects techniques et architecturaux.

Nous avons opté pour la méthodologie 2TUP pour plusieurs raisons. Elle garantit une évolution flexible de notre application en exploitant l'axe fonctionnel et l'axe technique en parallèle. Ainsi, si une nouvelle fonctionnalité se présente, seule la branche fonctionnelle

sera modifiée. De même, si un nouveau besoin technique surgit, la branche technique peut être traitée puis réintégrée dans le projet facilement. La réalisation du logiciel consiste en la fusion des résultats de ces deux branches.

Le processus 2TUP est composé de trois branches qui sont les suivantes :

La branche technique : cette branche se concentrera sur les aspects techniques du projet, incluant la mise en œuvre et l'intégration de Logstash, Talend, Power BI et la configuration de Tomcat pour l'application web. Les activités de cette branche seront axées sur la conception et le développement des systèmes et des infrastructures nécessaires à la collecte, au nettoyage, à la transformation, à l'analyse et à la visualisation des données de journal. En garantissant que les solutions techniques sont robustes, performantes et sécurisées, cette branche s'aligne directement sur l'objectif d'automatisation et de gestion efficace des fichiers journaux.

La branche fonctionnelle : La branche fonctionnelle sera responsable de la capture, de la compréhension et de la documentation détaillée des besoins de l'entreprise. Elle établira des cas d'utilisation, des scénarios d'utilisation et des spécifications fonctionnelles qui guideront le développement. La communication étroite avec les parties prenantes internes et externes assurera que les solutions techniques répondent précisément aux besoins de l'entreprise. Cette branche contribuera à l'adoption réussie des solutions en garantissant qu'elles résolvent les problèmes réels et apportent de la valeur ajoutée.

La branche conception et développement logiciel : Cette branche facilitera la conception détaillée des fonctionnalités et des interfaces utilisateur de l'application web, tout en coordonnant le développement et les tests. Elle veillera à ce que les solutions techniques soient mises en œuvre conformément aux spécifications fonctionnelles et techniques établies dans les deux autres branches. En alignant la conception et le développement, cette branche contribuera à la création d'une application web conviviale et performante, parfaitement adaptée aux besoins de l'entreprise.

Conclusion

Ce chapitre a servi à présenter l'entreprise, puis le cadre du projet, ensuite la problématique, l'étude de l'existant et enfin notre solution et à expliquer la méthodologie du travail adoptée.

Chapitre 2

Analyse et spécification des besoins

Introduction

La phase d'analyse et spécification des besoins est une étape analytique contenant une description complète du comportement du cycle de développement d'un projet.

2.1 Contexte Statique

2.1.1 Notions de Base

Dans cette section, nous exposons les divers concepts qui forment la base du projet, dans le but de parvenir à une meilleure compréhension de celui-ci.

2.1.1.1 Soliam

Le progiciel Soliam de Vermeg est une solution intégrée de gestion d'investissement front-to-back prenant en charge plusieurs normes comptables, des entités multiples, des dépositaires multiples et permettant à chaque investisseur institutionnel (compagnies d'assurances, instituts de prévoyance, caisses de retraites) de gérer l'ensemble de sa chaîne de valeur d'investissement en y intégrant le cycle de souscription rachat de son passif.

Les services fournis par Soliam dépassent l'informatique : ils fournissent à nos clients plus de souplesse, plus de sécurité et plus d'autonomie.

Nous y parvenons en leur fournissant une plate-forme intégrée, s'appuyant sur une base de données unique, du back-office au front-office et dans toutes les catégories d'actifs et mettant à disposition les outils et métriques nécessaires à leurs décisions d'investissement.

2.1.1.2 Fichier log

Dans un contexte informatique, un journal (log) désigne la documentation automatiquement générée et horodatée des événements concernant un système particulier. Pratiquement

tous les systèmes et logiciels produisent des fichiers journaux.

Dans le contexte d'un logiciel de gestion d'investissement comme Soliam, un "fichier log" fait référence à un enregistrement électronique qui contient des informations détaillées sur les événements, les actions et les opérations effectuées dans le système. Les fichiers de log jouent un rôle essentiel dans la surveillance, la détection d'erreurs, le suivi des activités et la sécurité du logiciel. Ils enregistrent généralement des informations telles que :

- **Horodatage (timestamp)** : L'heure et la date exactes auxquelles un événement s'est produit.
- **Type d'événement** : Une description de l'action ou de l'événement qui s'est produit, par exemple, une opération de souscription, de rachat ou une mise à jour de portefeuille.
- **Utilisateur ou entité** : L'identifiant de l'utilisateur ou de l'entité qui a effectué l'action. Cela peut être un gestionnaire d'investissement, un administrateur système, etc.
- **Détails de l'événement** : Des informations spécifiques sur l'événement, telles que les valeurs entrées, les modifications apportées, les transactions effectuées, etc.
- **Niveau de gravité** : Une indication du degré d'importance de l'événement, pouvant aller de simple information à une erreur critique.
- **Codes d'erreur** : Si une opération échoue ou rencontre un problème, un code d'erreur peut être enregistré pour faciliter le diagnostic.
- **Adresse IP** : Dans le cas de transactions en ligne, l'adresse IP de l'ordinateur à partir duquel l'action a été effectuée peut être enregistrée.
- **Trace d'exécution** : Une séquence d'étapes ou de processus qui ont conduit à l'événement enregistré.

2.2 Branche Technique

Au sein de cette section, nous exposons l'architecture technique de notre solution, puis énumérons les outils que nous avons utilisés pour le développement de notre application.

2.2.1 Etude du progiciel Soliam

2.2.1.1 Architecture de Soliam

Soliam se base sur une architecture monolithique c'est-à-dire qu'elle est développée en un seul bloc « war », déployée d'une manière unitaire dans un seul serveur d'application « Tomcat » et communique avec une seule base de données « Oracle ».

Le schéma 2.1 ci-dessous représente l'architecture de la solution Soliam :

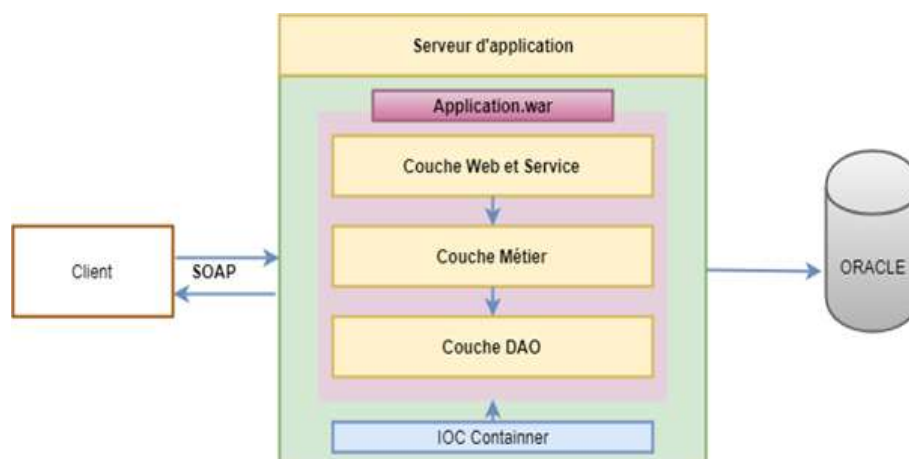


FIGURE 2.1 – Architecture Monolithique.

2.2.1.2 Contraintes techniques

Le projet à réaliser nécessite une étude de l'existant de Vermeg à savoir la solution Soliam sauf que cette étude nous a pris beaucoup de temps vue l'existence de plusieurs contraintes techniques dont nous pouvons citer :

- L'absence de documentation et d'informations disponibles sur le progiciel Soliam, ce qui engendre des difficultés au niveau de la compréhension métier et technique de la solution existante Soliam.
- Architecture complexe : Soliam se base sur une architecture monolithique qui présente plusieurs limites.
- Décentralisation des fichiers logs : En effet, les journaux sont réparties sur plusieurs sources tel que : la base de données oracle, des fichiers .txt, .log etc.

Toutes ces contraintes montrent le degré de difficulté de notre mission, et pour cette raison l'étude du progiciel Soliam nécessite assez du temps afin de choisir les outils technologiques adéquats.

2.3 Choix technologiques

Les choix technologiques ont été minutieusement évalués pour chaque étape du projet.

2.3.1 Choix de Logstash et Elasticsearch

Pour la collecte initiale des fichiers journaux, nous avons opté pour Logstash, un outil de traitement des données en temps réel qui offre des fonctionnalités robustes pour extraire et transformer les données brutes. Logstash présente l'avantage d'une intégration fluide avec

notre environnement existant et offre une flexibilité pour traiter différents types de fichiers journaux.

Une autre raison déterminante dans le choix de Logstash est son intégration native avec Elasticsearch, un moteur de recherche et d'analyse distribué. Cette synergie permet d'alimenter Elasticsearch avec les données collectées, créant ainsi une base solide pour des recherches et des analyses ultérieures. Les données sont indexées et stockées de manière optimisée

2.3.2 Choix de Talend

En ce qui concerne le nettoyage et le transformation des données, notre choix s'est orienté vers la plateforme ETL polyvalente, Talend. Sa capacité à gérer différents types de sources de données, qu'elles proviennent de bases de données SQL, NoSQL (Elasticsearch) ou de fichiers divers, offre une flexibilité essentielle à notre projet. Talend nous permettra d'appliquer des règles prédéfinies pour nettoyer et enrichir les données, garantissant ainsi leur cohérence et leur qualité avant leur utilisation dans les analyses ultérieures

De plus, un autre facteur décisif dans notre choix de Talend est sa capacité à alimenter les données finales dans une base de données Oracle. Cette caractéristique répond précisément à nos besoins, car notre progiciel Soliam est configuré pour lire exclusivement à partir des tables Oracle.

2.3.3 Choix de Power BI

Power BI a été sélectionné pour la visualisation des données et la création de tableaux de bord interactifs. En combinant des sources de données provenant de diverses étapes du processus, Power BI permettra aux parties prenantes de visualiser et d'explorer les données en temps réel, facilitant ainsi la prise de décision éclairée.

2.3.4 Choix du serveur Tomcat

Pour le déploiement de l'application web, nous avons choisi Tomcat comme serveur d'application. Tomcat offre une plateforme stable et bien établie pour héberger notre application, assurant ainsi une accessibilité optimale pour les utilisateurs.

Un autre avantage majeur de Tomcat réside dans sa compatibilité avec notre système existant. Notamment, il est le même serveur sur lequel le progiciel Soliam a déjà été déployé avec succès.

2.4 Branche Fonctionnelle

2.4.1 Identification des acteurs

Les acteurs clés impliqués dans notre projet comprennent les administrateurs de système, les analystes de données et les utilisateurs finaux.

Développeur : Il est responsable de la configuration et de la maintenance des flux de collecte.

Analyste de données : Il effectue des transformations et des analyses avancées.

L'utilisateur final : Il interagit avec les tableaux de bord pour obtenir des informations pertinentes.

2.4.2 Besoins fonctionnels

- Pour le développeur :

- Configurer les flux de collecte automatisés à partir de différentes sources de fichiers journaux.
- Surveiller en temps réel l'état des collectes en cours, détecter les éventuelles erreurs et prendre des mesures correctives si nécessaire.
- Intégrer les rapports Power BI générés dans l'application web déployée sur Tomcat.

- Pour l'analyste de données :

- Appliquer des transformations ETL aux données collectées, en utilisant des règles de nettoyage, de filtrage et de mise en forme.
- Stocker les données finales dans des tables Oracle.
- Générer des rapports pour présenter les résultats de leurs analyses de manière claire et concise.

- Pour l'utilisateur Final :

- Accéder aux rapports Power BI intégrés.

2.4.3 Besoins non fonctionnels

- **Performance :** L'utilisateur s'attend à une réponse rapide lors de l'accès aux données et à la navigation dans les tableaux de bord.
- **Sécurité :** Les données doivent être sécurisées à toutes les étapes du processus, de la collecte à la visualisation, avec des mécanismes d'authentification et de protection des données.

- **Évolutivité** : Le système doit être capable de gérer une augmentation du volume de données sans compromettre les performances.

2.4.4 Besoins de domaine

- **Conformité Réglementaire** : Les données collectées et traitées doivent être conformes aux réglementations de l'industrie et aux normes de sécurité en vigueur.
- **Intégration avec 'Soliam'** : Étant donné que l'application web 'Soliam' ne peut lire que depuis des tables Oracle, le système doit être capable d'intégrer les données transformées dans une base de données Oracle pour une utilisation transparente avec 'Soliam'.
- **Flexibilité pour les Flux de Données Spécifiques** : Le système doit être suffisamment flexible pour prendre en charge des flux de données spécifiques à l'entreprise, en s'adaptant aux différentes sources de fichiers journaux et en permettant des transformations personnalisées.

2.5 Spécification des besoins

Après avoir précisé les acteurs du système et ses besoins, il est primordial de donner une vision globale sur le comportement fonctionnel de notre application. Nous procédons dans cette section à modéliser les fonctions et modules que doit fournir le système à l'aide du diagramme de cas d'utilisation.

2.5.1 Diagramme de cas d'utilisation global

Le diagramme de cas d'utilisation global du candidat et de l'administrateur est illustré dans La figure 2.2. Ce diagramme sera détaillé dans la section suivante.

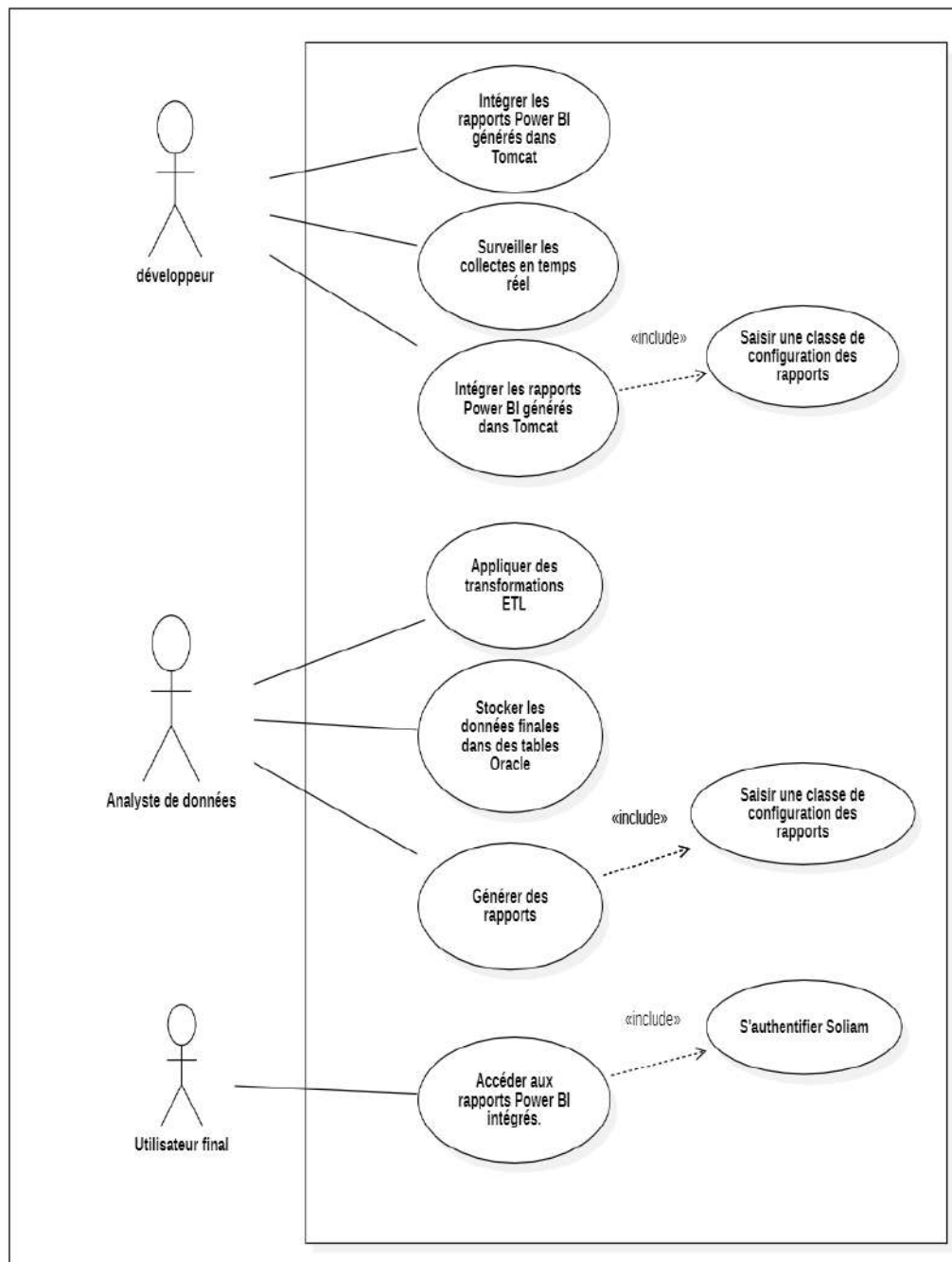


FIGURE 2.2 – Diagramme de cas d'utilisation global.

2.5.2 Cas d'utilisation détaillés

2.5.2.1 Cas d'utilisation « Configurer les flux de collecte automatisés »

La configuration des flux de collecte est une fonctionnalité très importante. La figure 2.3 ci-dessous permet à un développeur de collecter les fichiers de logs à partir de serveur Tomcat.

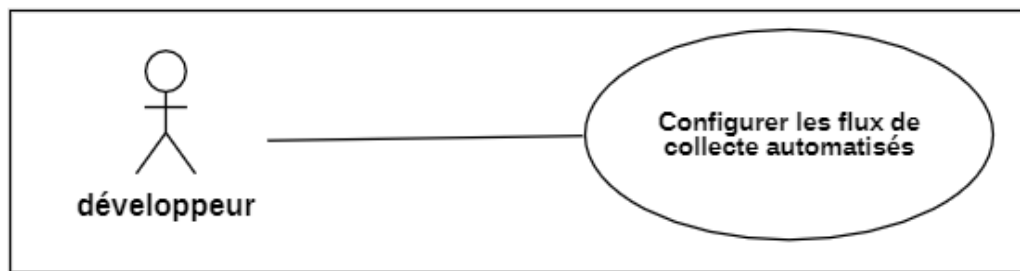


FIGURE 2.3 – Diagramme de cas d'utilisation « Configurer les flux de collecte ».

Le tableau 2.1 décrit l'enchaînement du cas d'utilisation « Configurer les flux de collecte ».

Description	Configuration des flux de collecte
Acteur	Développeur
Précondition	Développeur authentifié.
Postcondition	Logstash activé, Tomcat activé
Scénario de base	1 : Le développeur ajoute le code de collecte logstash dans le code source de Soliam. 2 : Le système valide logstash à utiliser lors de l'exécution des web services.

TABLE 2.1 – Cas d'utilisation : Configuration des flux de collecte

La figure 2.4 présente le diagramme de séquence système du cas d'utilisation « Configurer les flux de collecte ».

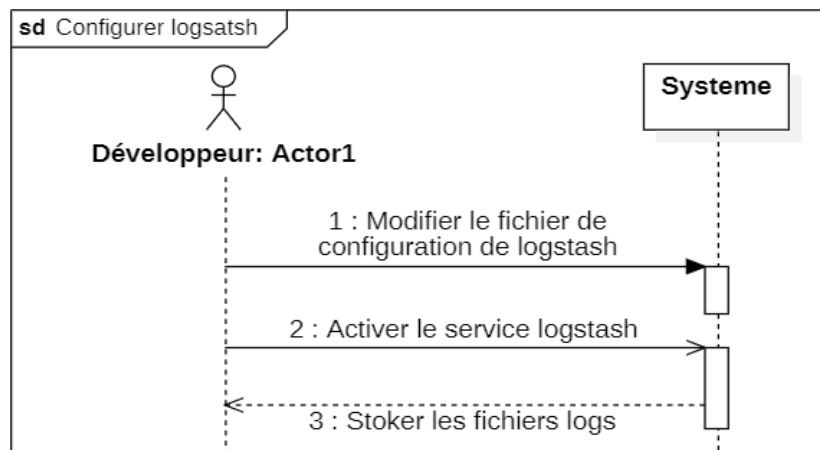


FIGURE 2.4 – Diagramme de séquence « Configurer les flux de collecte ».

2.5.2.2 Cas d'utilisation « Intégrer les rapport dans Soliam »

La figure 2.5 décrit le besoin d'intégrer les rapport Power BI dans Soliam.

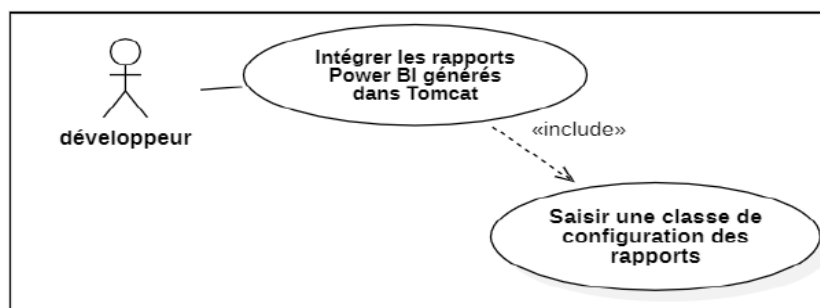


FIGURE 2.5 – Diagramme de cas d'utilisation « Intégrer les rapport dans Soliam ».

Le tableau 2.2 décrit l'enchaînement du cas d'utilisation « Intégrer les rapport dans Soliam ».

Description	Ce cas d'utilisation présente les étapes pour intégrer les rapport dans le site web du produit Soliam .
Acteur	Développeur
Précondition	Développeur authentifié.
Postcondition	Tomcat activé
Scénario de base	1 : Le développeur ajoute les informations de chaque rapport dans une classe java. 2 : Le système valide ces informations et affiche ces rapports d'une façon interactive comme un service web .

TABLE 2.2 – Cas d'utilisation : Intégration des rapports Power BI dans Soliam

La figure 2.6 présente le diagramme de séquence système du cas d'utilisation « Intégration des rapports Power BI dans Soliam ».

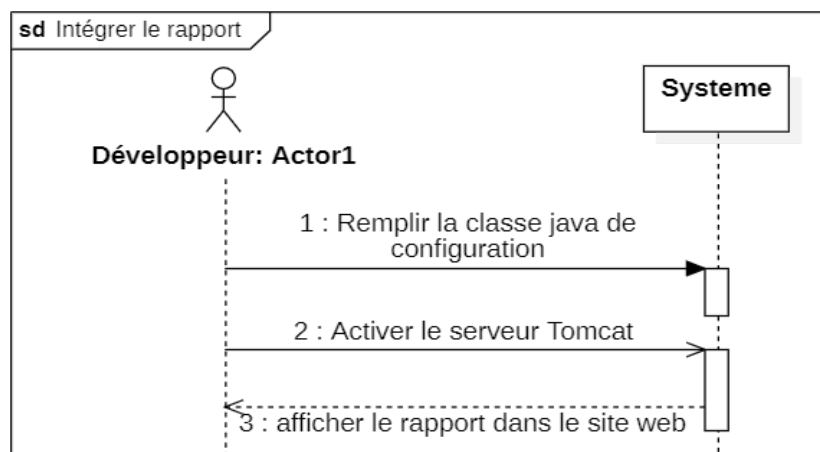


FIGURE 2.6 – Diagramme de séquence « Configurer les flux de collecte ».

2.5.2.3 Cas d'utilisation « Appliquer des transformations ETL »

La figure 2.7 représente le besoin fonctionnel d'appliquer des transformations ETL avec Talend.

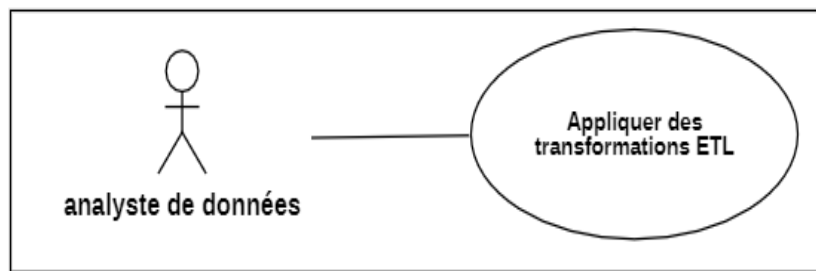


FIGURE 2.7 – Diagramme de cas d'utilisation « Appliquer des transformations ETL ».

Le tableau 2.3 décrit l'enchaînement du cas d'utilisation « Appliquer des transformations ETL ».

Description	Ce cas d'utilisation présente les étapes pour nettoyer les données exportées depuis de Elasticsearch.
Acteur	Analyste de données
Précondition	Elasticsearch activé.
Postcondition	Les données doivent être cohérentes.
Scénario de base	1 : L'analyste entre la configuration de base de données Elasticsearch dans Talend. 2 : L'analyste ajoute les tables de fichiers logs comme Data Input. 3 : L'analyste crée des composants de nettoyage pour les lignes nulles et les lignes doublantes dans Talend.

TABLE 2.3 – Cas d'utilisation : Appliquer des transformations ETL

La figure 2.8 présente le diagramme de séquence système du cas d'utilisation « Appliquer des transformations ETL ».

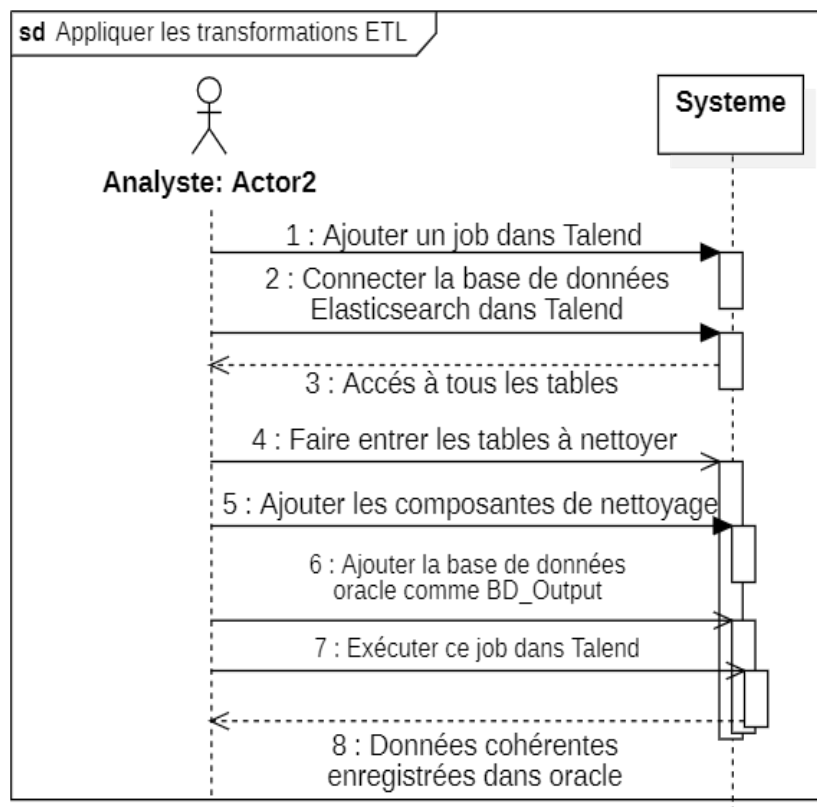


FIGURE 2.8 – Diagramme de séquence « Appliquer des transformations ETL ».

Conclusion

Dans ce chapitre, nous avons fourni une analyse détaillée des besoins fonctionnels et non fonctionnels grâce à l'ensemble des diagrammes de cas d'utilisation que nous avons détaillé. Dans le chapitre suivant, nous présentons l'architecture ainsi la conception logicielle détaillée du projet.

Chapitre 3

Conception

Introduction

Dans cette section, nous explorerons en profondeur l'architecture globale du système que nous avons élaborée pour répondre aux exigences identifiées dans les chapitres précédents. Cette architecture englobe à la fois l'infrastructure matérielle et logicielle qui constitue notre solution complète.

3.1 Conception globale

Dans cette partie, il est important de mentionner le choix de notre architecture physique et logicielle.

3.1.1 Architecture physique

Compte tenu des composants et des exigences que nous avons mentionnés pour notre solution (collecte, transformation ETL, stockage de données, génération de rapports, intégration avec une application web), une architecture adaptée pourrait être une architecture distribuée à trois niveaux (3-Tiers).

Le figure 3.1 illustre cette architecture.

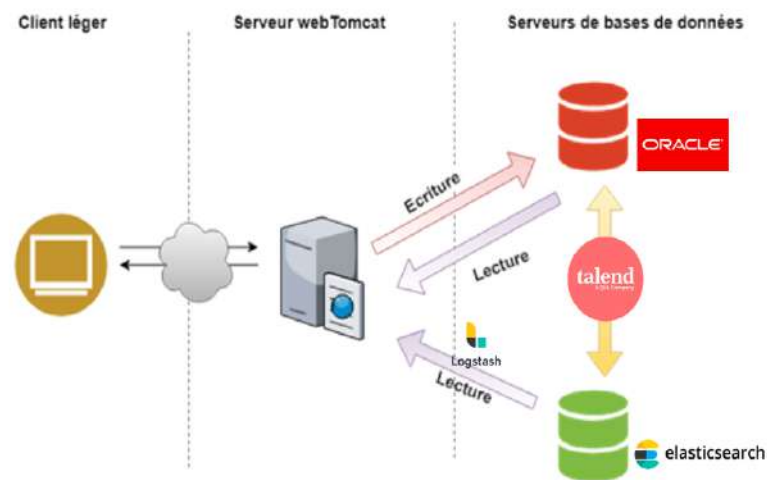


FIGURE 3.1 – Architecture physique de l'application.

3.1.2 Architecture logicielle

L'architecture logicielle de notre solution, qui inclut les composants logiciels et leur interaction, pourrait être structurée comme suit :

3.1.2.1 Couche de Présentation :

- **Application Web (Déployée sur Tomcat)** : Cette application offre une interface utilisateur conviviale aux utilisateurs finaux pour accéder aux fonctionnalités du système. Elle inclut les tableaux de bord, les options d'exploration des données, et l'intégration du rapport Power BI.

3.1.2.2 Couche Logique :

- **Collecte des Fichiers Journaux (Logstash)** : Logstash est utilisé pour collecter des fichiers journaux à partir de différentes sources. Chaque source est configurée comme un pipeline de collecte distinct.
- **Transformation ETL (Talend)** : Talend gère les processus de transformation ETL. Il applique des règles prédéfinies pour nettoyer, filtrer et formater les données collectées avant de les envoyer à la couche de données.

3.1.2.3 Couche de Données :

- **Base de Données Oracle** : La base de données Oracle stocke les données transformées de manière organisée. Elle comprend des tables dédiées pour chaque type de données, garantissant l'intégrité et la cohérence des données.

3.1.2.4 Intégration :

- **Intégration de Power BI** : Power BI est intégré dans l'application web via une API. Les rapports générés par Power BI sont accessibles aux utilisateurs finaux via l'application web.

3.2 Aspect statique du système

Dans cette section, nous explorerons l'aspect statique de notre système, en mettant l'accent sur la structure des composants et leur organisation au sein du système.

3.2.1 Diagramme de Paquetages

Le diagramme de paquetages est un outil essentiel pour comprendre l'organisation globale de notre système en termes de modules, de composants et de dépendances entre eux. Il permet de visualiser les différents paquetages logiciels et la manière dont ils sont regroupés pour assurer une gestion efficace du système.

La figure 3.2 représente clairement le diagramme de paquetages de notre solution.

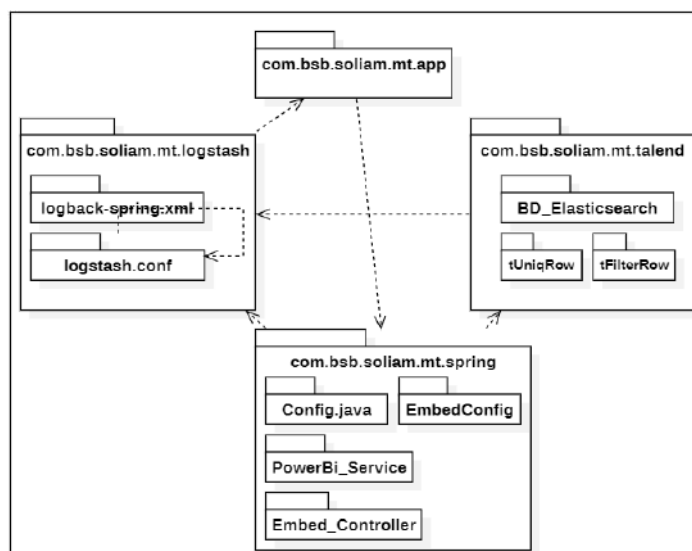


FIGURE 3.2 – Diagramme de paquetages.

- `com.bsb.soliam.mt.app` : Ce paquetage englobe tous les composants liés à l'interface utilisateur de notre application web. Il contient les vues, les contrôleurs et les ressources graphiques nécessaires pour fournir une expérience utilisateur fluide.
- `com.bsb.soliam.mt.logstash` : Dans ce paquetage, nous regroupons tous les composants associés à la collecte des fichiers journaux. Cela inclut les configurations spécifiques à chaque source de données, les pipelines de collecte et les modules de gestion des erreurs.
- `com.bsb.soliam.mt.talend` : Ce paquetage englobe les composants de transformation ETL. Il contient les jobs, les règles de transformation, les routines de nettoyage et les mécanismes de gestion des flux de données.
- `com.bsb.soliam.mt.spring` : Ce paquetage contient les composants requis pour intégrer Power BI dans notre application web. Il gère les connexions, les rapports et les mécanismes de visualisation.

3.2.2 Déploiement de l'Application

Le déploiement de l'application est une étape cruciale de notre projet, car elle permet de mettre en production notre solution complète. Dans cette section, nous discuterons des détails du déploiement de l'application web.

3.2.2.1 Description du Déploiement de l'Application

Serveurs Cibles : Nous détaillerons les serveurs sur lesquels notre application web sera déployée. Cela inclut le serveur Tomcat qui hébergera l'interface utilisateur et les éventuels serveurs nécessaires pour l'intégration de Power BI.

Configuration du Serveur : Nous expliquerons les configurations spécifiques qui doivent être appliquées sur chaque serveur, y compris les prérequis logiciels, les paramètres de sécurité et les configurations réseau.

Déploiement de l'Application : Nous décrirons les étapes du processus de déploiement, depuis la préparation des fichiers d'application jusqu'à leur déploiement sur les serveurs cibles. Nous aborderons également les mécanismes de sauvegarde et de restauration.

Tests Post-Déploiement : Nous discuterons des tests qui seront effectués après le déploiement pour s'assurer que l'application fonctionne correctement. Cela inclut les tests d'intégration, de performance et de sécurité.

Conclusion

Au cours de ce chapitre, nous avons présenté l'architecture de notre projet et les aspects statiques grâce aux diagrammes de classe, séquence et activités. Le chapitre suivant sera consacré à l'étape de réalisation.

Chapitre 4

Réalisation

Introduction

Nous dédions ce chapitre à la présentation du travail accompli au cours de la phase d'implémentation. Dans la première section, nous décrivons le contexte dans lequel le projet a été réalisé, notamment l'environnement matériel et logiciel. Dans la seconde section, nous offrons un aperçu du travail effectué en illustrant quelques captures d'écran accompagnées de commentaires explicatifs.

4.1 Environnement de travail

Cette partie est dédiée à la présentation des environnements matériels et logiciels qui nous permet de réaliser notre projet.

4.1.1 Environnement matériel

Pour mener à bien ce projet, nous avons utilisé un ordinateur portable dont les caractéristiques sont les suivantes :

- **Système d'exploitation** : Windows 11.
- **Processeur** : Intel Core i5-8250U, jusqu'à 3.4 GHz.
- **Mémoire** : 16 Go de RAM.
- **Disque dur** : 1 To.

4.1.2 Environnement logiciel

4.1.2.1 Technologies utilisées

Afin d'atteindre les finalités de notre projet, nous avons choisi de travailler avec les outils suivants :

Collecte de Fichiers Journaux (Logstash)

Logstash : Un outil open source de collecte, de traitement et de transfert de données en temps réel. Il est couramment utilisé pour collecter des fichiers journaux de différentes sources.

Transformation ETL (Talend)

Talend : Une plateforme ETL open source puissante et polyvalente. Elle permet de nettoyer, transformer et enrichir les données à partir de diverses sources.

Base de Données (Oracle)

Oracle Database : Une base de données relationnelle hautement évolutive et robuste pour le stockage des données transformées. Elle est couramment utilisée dans les entreprises.

Application Web (Tomcat)

Apache Tomcat : Un serveur d'application Web open source qui peut être utilisé pour héberger votre application web.

Intégration de Rapports (Power BI)

Power BI : Une suite d'outils d'analyse et de business intelligence développée par Microsoft. Elle permet de créer des tableaux de bord interactifs et des rapports visuels à partir des données.

Les Frameworks

Java : Java est un langage typé et orienté objet. Il est compilé et basé sur une architecture logicielle très particulière nécessitant une machine virtuelle Java. Il utilise les notions usuelles de la programmation orientée objet. Il est accompagné d'un ensemble énorme de bibliothèques standard couvrant de très nombreux domaines, notamment des bibliothèques graphiques. [URL1]

Spring Framework : SPRING est un framework libre pour construire et définir l'architecture d'une application java. Il est également un conteneur léger permettant de simplifier l'intégration des différentes couches. En plus Spring est totalement portable sur tous les serveurs d'application et totalement intégrable avec toutes les technologies choisies pour le développement de ce projet. [URL2]

Maven : Maven est un outil de build, de gestion des dépendances et du cycle de vie pour des projets Java. Il utilise un fichier de configuration XML appelé POM (Project Object Model) afin de décrire un projet logiciel, ses dépendances avec des modules externes et l'ordre à suivre pour sa production. [URL3]

Angular :Angular 15 est un Framework frontend qui permet de créer des pages web en utilisant HTML, SCSS et Type Script. Il implémente les fonctionnalités de base et facultatives sous la forme d'un ensemble de bibliothèques Type Script que vous importez dans vos applications [URL4]

4.1.2.2 Environnement de développement intégré

Visual Studio Code

Visual Studio Code est un éditeur de code extensible développé par Microsoft. Il intègre des extensions pour plusieurs langages de programmation, notamment Java, JavaScript et C++ et des environnements d'exécution tels que .Net et Unity. [URL5]

Postman

Postman est une plateforme collaborative de développement des API. Il permet d'envoyer des requêtes REST, SOAP pour tester vos API et Surveiller leurs performances. [URL6]

IntelliJ Idea

IntelliJ Idea est un environnement de développement intégré (IDE) pour le développement de logiciels en Java, Kotlin surtout. Il offre l'intégration avec des outils de construction tels que Maven et Gradle. Aussi, il prend en charge l'intégration des frameworks de développement tels que Spring, Hibernate et Struts. [URL7]

Star UML

Star UML est un logiciel qui permet de dessiner des diagrammes ou des organigrammes. Il vous propose de concevoir toutes sortes de diagrammes, de dessins vectoriels, de les enregistrer sur plusieurs formats XML, PNG, JPG puis de les exporter. [URL8]

4.2 Aperçu des interfaces

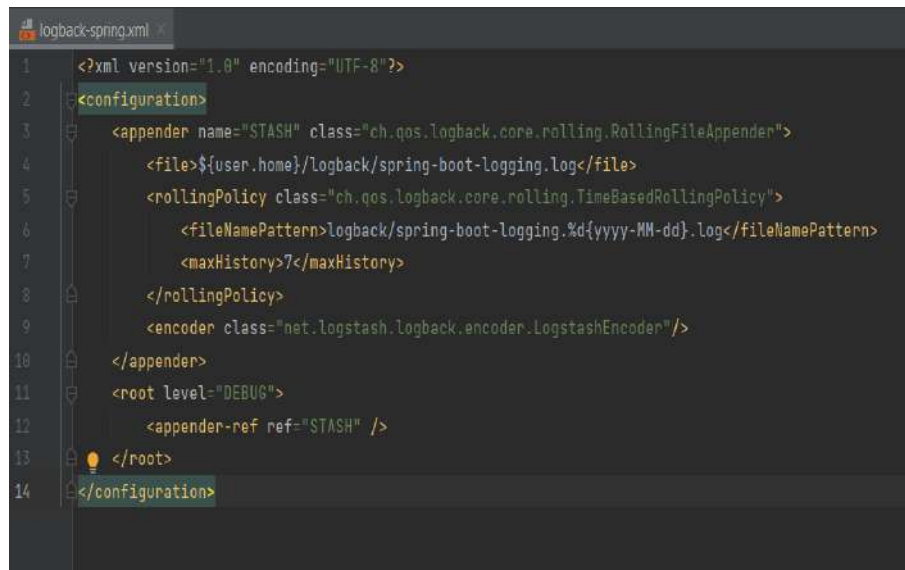
Dans cette partie, nous allons dévoiler les interfaces de la solution.

4.2.1 Interfaces de Collecte

Dans cette section, nous explorerons les interfaces utilisées pour collecter des fichiers journaux à partir de différentes sources. Cela inclut :

4.2.1.1 Configuration des Sources

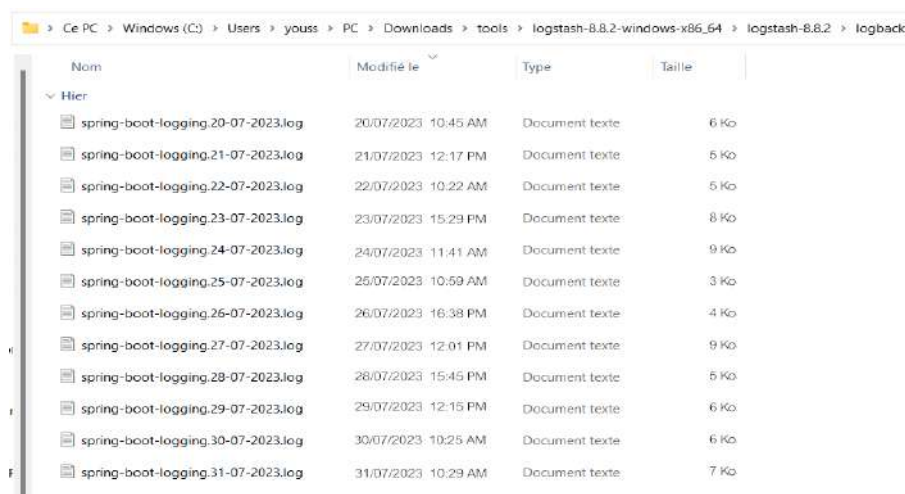
Les développeurs configurent les sources de collecte, y compris les paramètres spécifiques à chaque source avec le fichier "logback-spring.xml".



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <configuration>
3   <appender name="STASH" class="ch.qos.logback.core.rolling.RollingFileAppender">
4     <file>${user.home}/logback/spring-boot-logging.log</file>
5     <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
6       <fileNamePattern>logback/spring-boot-logging.%d{yyyy-MM-dd}.log</fileNamePattern>
7       <maxHistory>7</maxHistory>
8     </rollingPolicy>
9     <encoder class="net.logstash.logback.encoder.LogstashEncoder"/>
10  </appender>
11  <root level="DEBUG">
12    <appender-ref ref="STASH" />
13  </root>
14 </configuration>
```

FIGURE 4.1 – Script logback-spring.xml.

Dans la configuration de notre service Spring on a spécifié la manière dont les logs seront stockés dans un fichier log à l'aide d'un code XML de configuration "logback-spring.xml".



File Explorer path: Ce PC > Windows (C:) > Users > youss > PC > Downloads > tools > logstash-8.8.2-windows-x86_64 > logstash-8.8.2 > logback

Nom	Modifié le	Type	Taille
spring-boot-logging.20-07-2023.log	20/07/2023 10:45 AM	Document texte	6 Ko
spring-boot-logging.21-07-2023.log	21/07/2023 12:17 PM	Document texte	5 Ko
spring-boot-logging.22-07-2023.log	22/07/2023 10:22 AM	Document texte	5 Ko
spring-boot-logging.23-07-2023.log	23/07/2023 15:29 PM	Document texte	8 Ko
spring-boot-logging.24-07-2023.log	24/07/2023 11:41 AM	Document texte	9 Ko
spring-boot-logging.25-07-2023.log	25/07/2023 10:59 AM	Document texte	3 Ko
spring-boot-logging.26-07-2023.log	26/07/2023 16:38 PM	Document texte	4 Ko
spring-boot-logging.27-07-2023.log	27/07/2023 12:01 PM	Document texte	9 Ko
spring-boot-logging.28-07-2023.log	28/07/2023 15:45 PM	Document texte	5 Ko
spring-boot-logging.29-07-2023.log	29/07/2023 12:15 PM	Document texte	6 Ko
spring-boot-logging.30-07-2023.log	30/07/2023 10:25 AM	Document texte	6 Ko
spring-boot-logging.31-07-2023.log	31/07/2023 10:29 AM	Document texte	7 Ko

FIGURE 4.2 – Dossier logback des logs.

En particulier les logs seront enregistrés dans des fichiers .log dans un dossier logback sous un nom "spring-boot-logging.yyyy-MM-dd.log".

4.2.1.2 Pipelines de Collecte

ces interfaces sont utilisées pour définir et gérer les pipelines de collecte qui transforment les données brutes en données exploitables.

```

0.9.1-java/lib/manticore/client.rb:536: warning: already initialized constant Manticore::Client::StringEntity
C:/Users/youss/PC/Downloads/tools/logstash-8.8.2-windows-x86_64/logstash-8.8.2/vendor/bundle/jruby/2.6.0/gems/manticore-
0.9.1-java/lib/manticore/client.rb:536: warning: already initialized constant Manticore::Client::StringEntity
C:/Users/youss/PC/Downloads/tools/logstash-8.8.2-windows-x86_64/logstash-8.8.2/vendor/bundle/jruby/2.6.0/gems/manticore-
0.9.1-java/lib/manticore/client.rb:536: warning: already initialized constant Manticore::Client::StringEntity
C:/Users/youss/PC/Downloads/tools/logstash-8.8.2-windows-x86_64/logstash-8.8.2/vendor/bundle/jruby/2.6.0/gems/manticore-
0.9.1-java/lib/manticore/client.rb:536: warning: already initialized constant Manticore::Client::StringEntity
{
  "event" => {
    "original" => "2023-07-31 10:25:23,678 INFO com.example.MyController - Request received: GET /api/users\r"
  },
  "message" => "2023-07-31 10:25:23,678 INFO com.example.MyController - Request received: GET /api/users\r",
  "loglevel" => "INFO",
  "log" => {
    "file" => {
      "path" => "C:/Users/youss/PC/Downloads/tools/logstash-8.8.2-windows-x86_64/logstash-8.8.2/1_log.txt"
    }
  },
  "logmessage" => "Request received: GET /api/users\r",
  "class" => "com.example.MyController",
  "timestamp" => "2023-07-31 10:25:23,678",
  "host" => {
    "name" => "LAPTOP-QQBAR0DC"
  },
  "@timestamp" => 2023-07-31T10:25:23.678Z,
  "@version" => "1",
  "tags" => [
    [0] "_grokparsefailure"
  ]
}

```

FIGURE 4.3 – Serveur Logstash.

le figure 4.3 représente comment les logs sont capturés en temps réel depuis les fichiers de dossier backlog.

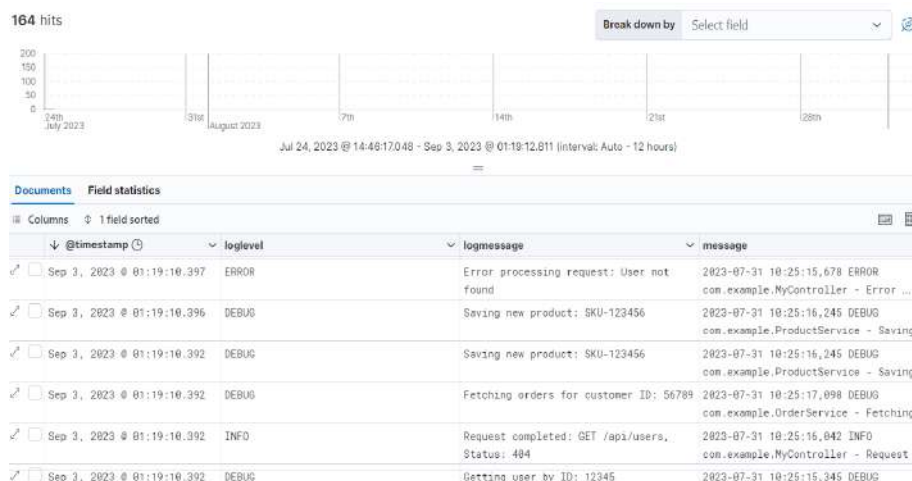


FIGURE 4.4 – Serveur Logstash.

le figure 4.4 pourrait decrire des filtres et des règles de traitement spécifiques que nous avons définis, les tables enregistrés ont été créés avec des colonnes de "timestamp", "logLevel", "logMessage".

4.2.2 Interfaces de Transformation ETL

Nous examinerons les interfaces qui permettent aux analystes de données de définir les règles de transformation ETL. Cela comprend :

4.2.2.1 Création de Jobs ETL

Dans cette partie, nous allons aborder comment les analystes créent des jobs ETL en spécifiant les étapes de transformation.

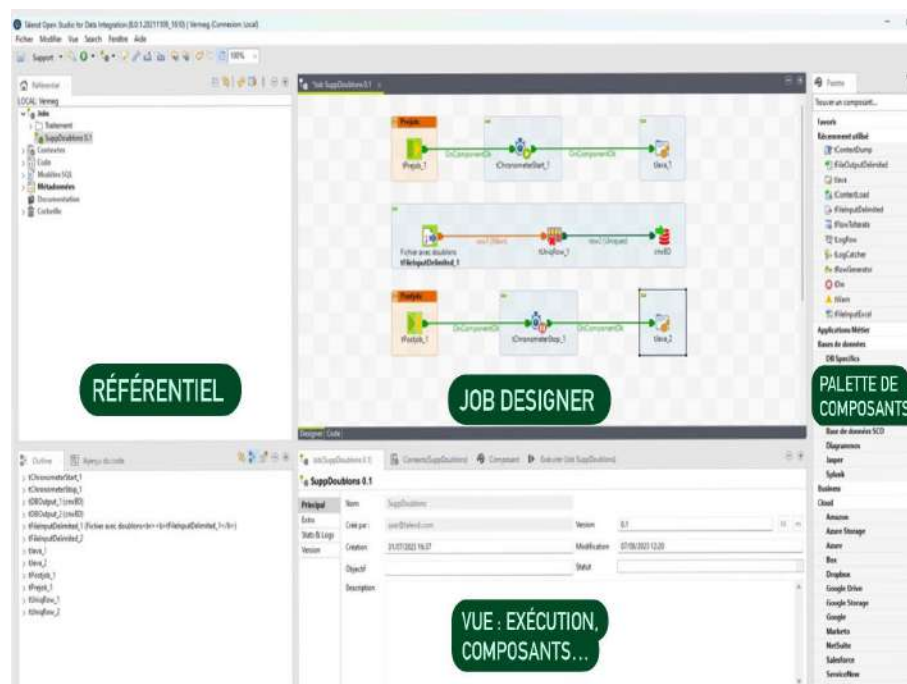


FIGURE 4.5 – Interface Talend.

L'utilisation combinée du Job Designer, de la palette des composants ETL, du référentiel et des différentes vues offre une approche holistique pour concevoir, développer et déboguer des jobs ETL de manière efficace et structurée dans l'environnement Talend.

4.2.2.2 Suppression des doublons

Pour la suppression des doublons dans le fichier d'entrée en utilisant le composant tUniqueRow de Talend for Data Integration. Ce composant permet d'éliminer les enregistrements en double en se basant sur des colonnes spécifiques définies par l'utilisateur.

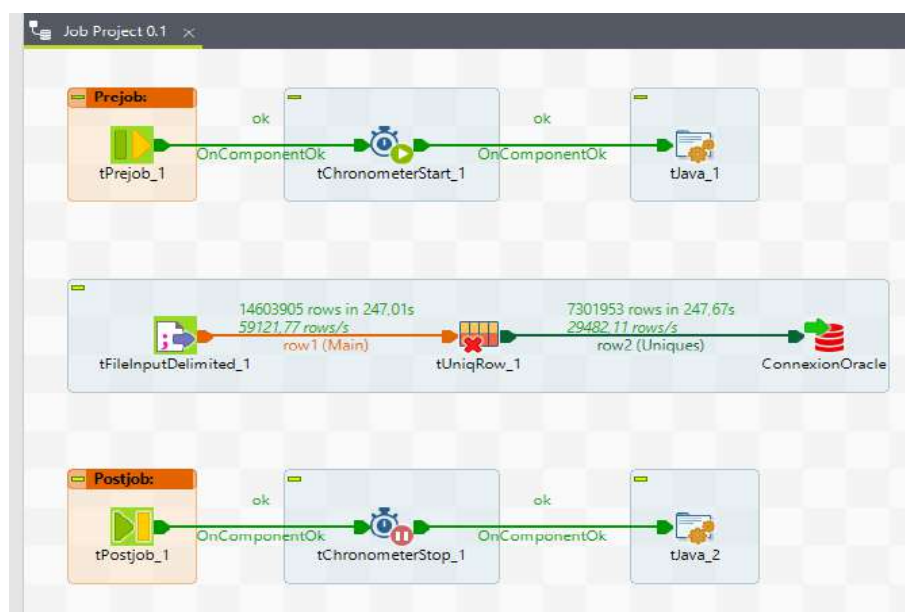


FIGURE 4.6 – Job Talend.

Le composant `tUniqRow` de Talend joue un rôle essentiel dans la gestion de la qualité des données en détectant et en supprimant les doublons d'un jeu de données. Ce composant identifie les enregistrements en double en fonction des critères définis et ne permet qu'à un seul enregistrement unique de passer, éliminant ainsi les redondances potentielles. Cette fonctionnalité est cruciale pour garantir la cohérence et l'intégrité des données dans nos processus ETL.

4.2.2.3 Analyse des performances et du temps d'exécution

L'efficacité et la rapidité de la suppression des doublons avec `tUniqRow` sont des aspects cruciaux pour nos flux de données. Nous avons mesuré les performances en utilisant un jeu de données de taille significative. Les résultats ont montré que le composant `tUniqRow` a réussi à traiter rapidement les doublons, contribuant ainsi à l'amélioration de la qualité des données tout en minimisant l'impact sur les performances globales du processus ETL.

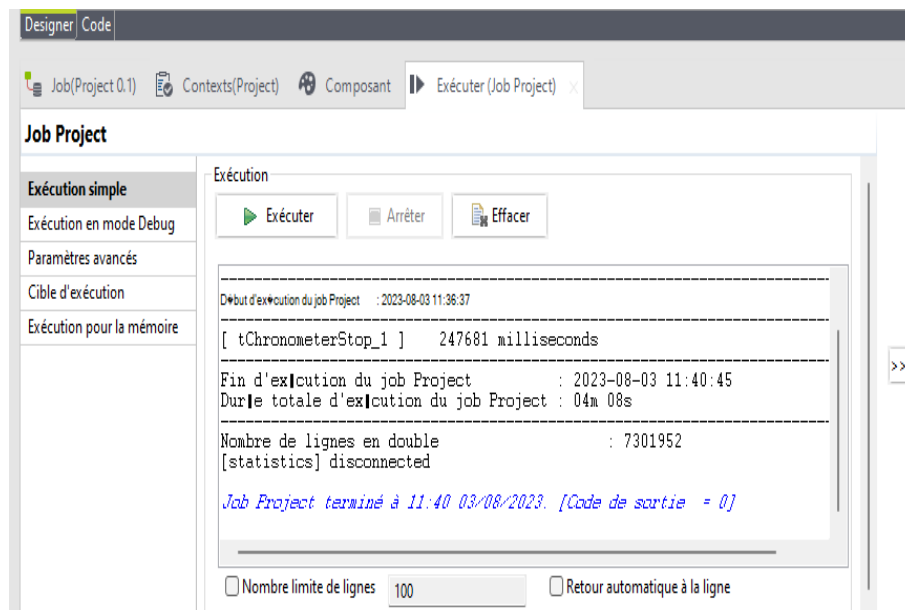


FIGURE 4.7 – Temps d'exécution

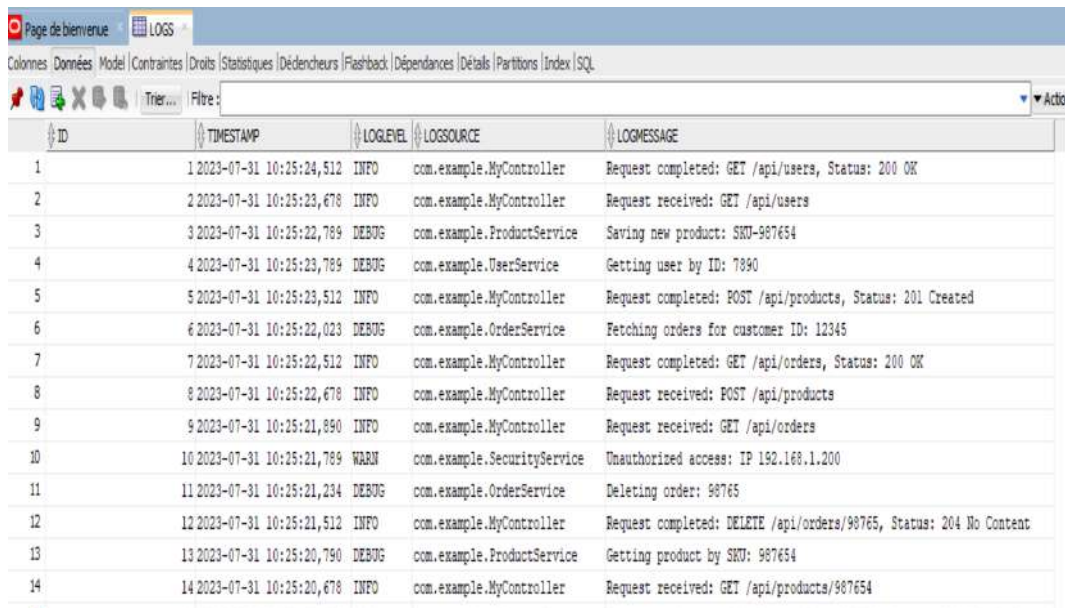
En utilisant le composant `tUniqRow`, nous sommes en mesure de garantir l'intégrité de nos données en éliminant les doublons, ce qui se traduit par des analyses plus précises et des rapports fiables dans notre environnement de traitement de données.

4.2.3 Interfaces de Stockage

Nous expliquerons les interfaces liées au stockage des données transformées dans la base de données Oracle, notamment :

4.2.3.1 Tables Oracle

Cette partie décrit les interfaces pour définir la structure des tables qui stockent les données finales.



ID	TIMESTAMP	LOGLEVEL	LOGSOURCE	LOGMESSAGE
1	2023-07-31 10:25:24,512	INFO	com.example.MyController	Request completed: GET /api/users, Status: 200 OK
2	2023-07-31 10:25:23,678	INFO	com.example.MyController	Request received: GET /api/users
3	2023-07-31 10:25:22,789	DEBUG	com.example.ProductService	Saving new product: SKU-987654
4	2023-07-31 10:25:23,789	DEBUG	com.example.UserService	Getting user by ID: 7890
5	2023-07-31 10:25:23,512	INFO	com.example.MyController	Request completed: POST /api/products, Status: 201 Created
6	2023-07-31 10:25:22,023	DEBUG	com.example.OrderService	Fetching orders for customer ID: 12345
7	2023-07-31 10:25:22,512	INFO	com.example.MyController	Request completed: GET /api/orders, Status: 200 OK
8	2023-07-31 10:25:22,678	INFO	com.example.MyController	Request received: POST /api/products
9	2023-07-31 10:25:21,890	INFO	com.example.MyController	Request received: GET /api/orders
10	2023-07-31 10:25:21,789	WARN	com.example.SecurityService	Unauthorized access: IP 192.168.1.200
11	2023-07-31 10:25:21,234	DEBUG	com.example.OrderService	Deleting order: 98765
12	2023-07-31 10:25:21,512	INFO	com.example.MyController	Request completed: DELETE /api/orders/98765, Status: 204 No Content
13	2023-07-31 10:25:20,790	DEBUG	com.example.ProductService	Getting product by SKU: 987654
14	2023-07-31 10:25:20,678	INFO	com.example.MyController	Request received: GET /api/products/987654

FIGURE 4.8 – Table des logs

4.2.4 Interfaces d'Intégration

Enfin, nous décrirons les interfaces d'intégration qui permettent à notre application web de communiquer avec Power BI, notamment :

4.2.4.1 Connexions à Power BI

Avec Power BI Embedded, on peut intégrer un rapport Power BI dans notre application Java en utilisant des API et des composants de Power BI. Cela offre une personnalisation avancée, une interaction en temps réel et la possibilité d'intégrer des filtres et des actions. Les utilisateurs peuvent interagir avec le rapport comme s'ils étaient dans l'application Power BI.

```

user: youssefch@dataVisualisationEnsi.onmicrosoft.com
password: *****
Application Name : testDashboard
Application Id:1bccdc15-6d34-41d3-a8a7-cf3fec140f28
Tenant Id:63643b42-f4d7-4ba7-bfa6-5bf0c2741338
key:NIG8Q~SSDdl19ICqtCJyylyQB2jISygaiCnGqcl9
Name gorup:PowerBIEmbeddedGroup
https://app.powerbi.com/groups/236605eb-e644-40e6-b910-63d85274f0cd/reports/d2c6fba1-976c-44fd-83ab-42d525aa096e/ReportSection?experience=power-bibi
Workspace id: 236605eb-e644-40e6-b910-63d85274f0cd
Report id: d2c6fba1-976c-44fd-83ab-42d525aa096e
d97f980d-ba86-455d-b864-8d578d9de2af

```

FIGURE 4.9 – configuration d'un rapport Power BI

L'intégration d'un rapport Power BI dans une application Java à l'aide de Power BI Embedded est une procédure en plusieurs étapes qui nécessite une configuration minutieuse à la fois dans Azure Active Directory et Power BI.

4.2.4.2 Affichage des Rapports

Après l'ajout la configuration des rapports, ces derniers seront affichés dans notre interface utilisateur d'une façon interactive et dynamique.

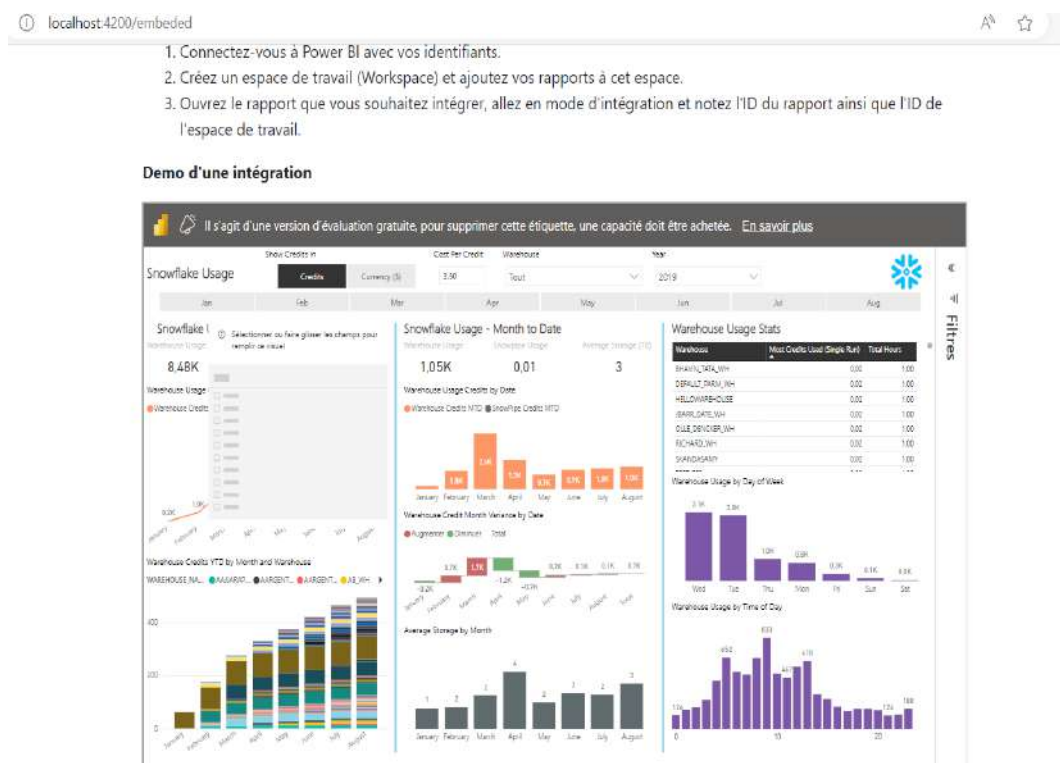


FIGURE 4.10 – un rapport Power BI

Conclusion

Dans ce chapitre, nous avons présenté les différents éléments techniques de notre projet. Nous avons commencé par une étude des langages de programmation et bibliothèques utilisés dans notre environnement de développement. Ensuite, nous avons décrit les fonctionnalités offertes par notre solution à travers des captures d'écran.

Conclusion

En conclusion, ce projet a réussi à transformer notre gestion des fichiers logs de grande taille, améliorant la qualité de nos données, la réactivité aux incidents et la prise de décision. Nous nous engageons à maintenir et à développer cette solution pour répondre aux besoins futurs de notre entreprise. Ce projet a montré que l'analyse des logs peut avoir un impact significatif sur la performance de nos opérations informatiques.

En regardant en arrière sur nos objectifs initiaux, nous constatons que nous les avons accomplis de manière satisfaisante. Nous avons réussi à :

- Collecter efficacement des fichiers logs de grande taille à l'aide de Logstash et à les stocker dans Elasticsearch pour une recherche rapide et précise.
- Appliquer des processus de nettoyage et de transformation ETL sur ces logs à l'aide de Talend, créant ainsi une base de données Oracle bien structurée.
- Créer des rapports dynamiques et des tableaux de bord interactifs avec Power BI pour permettre à nos équipes d'analyser les données de manière approfondie.
- Intégrer ces rapports et tableaux de bord dans une application web conviviale hébergée sur Tomcat pour une accessibilité facile.

Webographie

- [URL1] Java : qu'est-ce que c'est? <https://www.futura-sciences.com/tech/definitions/informatique-java-485/>, [consulté le 15/07/2023].
- [URL2] Spring : définition. <https://spring.io/projects/spring-framework>, [consulté le 01/08/2023].
- [URL3] Maven - Présentation. <https://www.tutorialspoint.com/maven/maven>, [consulté le 02/08/2023].
- [URL4] Introduction aux concepts angulr. <https://angular.io/guide/architecture/>, [consulté le 01/08/2023].
- [URL5] Qu'est-ce que Visual Studio Code? L'éditeur de code extensible de Microsoft. <https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html>, [consulté le 01/08/2023].
- [URL6] Qu'est-ce que le facteur? <https://www.postman.com/product/what-is-postman/>, [consulté le 01/08/2023].
- [URL7] Présentation d'IntelliJ IDEA. <https://www.jetbrains.com/help/idea/discover-intellij-idea.html>, [consulté le 01/08/2023].
- [URL8] StarUML Présentation du logiciel. <https://dictionnaire.sensagent.com/STARUML/fr-fr/>, [consulté le 01/08/2023].