

## Dédicace

## **Remerciement**

# Table des matières

Dédicace .....	1
Remerciement.....	2
Table des matières .....	3
Liste des figures.....	7
Introduction générale.....	10
Chapitre 1 : Contexte général .....	12
1.    Introduction .....	13
2.    Organisme d'accueil .....	13
3.    Présentation du projet.....	14
3.1.    Problématique.....	14
3.2.    L'étude de l'existant .....	15
4.    Solution proposée.....	18
5.    Méthodologie de travail .....	18
5.1.    Méthodologie Agile pour la BI .....	18
5.2.    Méthodologie Kimball BI .....	19
5.3.    Méthodologie Lean BI .....	19
5.4.    Méthodologie choisie .....	20
6.    Conclusion.....	21
Chapitre 2 : Définition des besoins métiers.....	22
1.    Introduction .....	23
2.    Objectifs stratégiques de l'entreprise .....	23
2.1.    Centralisation des données pour une gestion optimisée.....	23
2.2.    Amélioration des capacités de recherche et d'analyse .....	23
2.3.    Visualisation des données pour faciliter la prise de décision.....	23
2.4.    Amélioration de l'efficacité opérationnelle .....	23
2.5.    Évolutivité et adaptabilité.....	24
3.    Objectifs généraux et spécifiques du projet .....	24
3.1.    Intégration des données des issues Jira .....	24
3.2.    Collecte et centralisation des commentaires Jira.....	25

3.3.	Gestion des données des projets Jira .....	26
3.4.	Amélioration de la productivité et de la prise de décision .....	27
3.5.	Objectifs stratégiques liés aux données :.....	28
4.	Métriques clés et indicateurs de performance .....	28
5.	Technologies utilisées .....	29
5.1.	Environnement matériel .....	29
5.2.	Environnement logiciel .....	30
	Conclusion .....	30
	<b>Chapitre 3 : Conception des données .....</b>	<b>32</b>
1.	Introduction .....	33
2.	Choix des données à extraire.....	33
2.1.	Source des données extraites : Hibernate Jira .....	33
3.	Dimensions de la data warehouse .....	36
3.1.	La dimension « Dim Project ».....	37
3.2.	La dimension « Dim Issue ».....	37
3.3.	La dimension « Dim Comment » .....	38
3.4.	La table de faits « Fact Table » .....	38
4.	Schéma de la data warehouse et relations .....	39
4.1.	Schémas des entrepôts de données .....	39
4.2.	Le choix du schéma en étoile .....	41
4.3.	Relations table de faits et dimensions .....	42
5.	Conclusion.....	43
	<b>Chapitre 4 : Chargement des données .....</b>	<b>44</b>
1.	Introduction .....	45
2.	Extraction et transformation des données .....	45
2.1.	ET_Projects .....	45
2.2.	ET_Fields .....	47
2.3.	ET_Comments.....	50
2.4.	Transformation pour ET_Comments .....	55
3.	Chargement des données dans la staging area .....	59
3.1.	Dim Projet .....	60
3.2.	Dim Issue.....	62
3.3.	Dim Comment.....	64

4.	Chargement de la data warehouse .....	65
4.1.	Dimensions.....	65
4.2.	Fact table .....	66
5.	Mise en place de l'environnement Elasticsearch et Kibana .....	68
5.1.	Configuration de elasticsearch.yml .....	68
5.2.	Installation de Kibana.....	71
6.	Connection de la base de données avec Elasticsearch .....	72
6.1.	Création du connecteur et de l'index Elasticsearch.....	72
6.2.	Configuration du connecteur "pgconnector" .....	73
6.3.	Configuration des fichiers config PostgreSQL .....	74
6.4.	Exécution du connecteur "pgconnector" .....	76
7.	Conclusion.....	78
	Chapitre 5 : Accès aux données et analyse.....	79
1.	Introduction .....	80
2.	Dashboards Power BI.....	80
2.1.	Dashboard Issue 1 .....	80
2.2.	Dashboard Issue 2 .....	85
2.3.	Dashboard Projet.....	90
3.	Application Python Elasticsearch.....	96
3.1.	Indexation des embeddings dans Elasticsearch.....	96
3.2.	Application de recherche Elasticsearch.....	98
4.	Conclusion.....	102
	Chapitre 6 : Maintenance et optimisation.....	103
1.	Introduction .....	104
2.	Fine tuning du modèle.....	104
2.1.	Préparation des données .....	104
2.2.	Training du modèle .....	107
2.3.	Evaluation du modèle.....	113
3.	Front end du projet .....	116
3.1.	Flask-Python.....	117
3.2.	Plateforme front end.....	117
4.	Conclusion.....	119
	Conclusion et Perspective.....	121

Bibliographie .....	123
---------------------	-----

# Liste des figures

Figure 1 Logo Zapier .....	15
Figure 2 Logo Jira Plugin Insight .....	16
Figure 3 Objectifs généraux .....	24
Figure 4 Intégration des données des issues Jira .....	25
Figure 5 Collecte et centralisation des commentaires Jira .....	26
Figure 6 Gestion des données des projets Jira .....	27
Figure 7 Amélioration de la productivité et de la prise de décision .....	28
Figure 8 Logo Hibernate .....	33
Figure 9 Projets Hibernate .....	34
Figure 10 Issues Hibernate .....	35
Figure 11 Commentaires Hibernate .....	36
Figure 12 Dimension Projet .....	37
Figure 13 Dimension Issue .....	37
Figure 14 Dimension Commentaire .....	38
Figure 15 Fact Table .....	39
Figure 16 Représentation star schema .....	40
Figure 17 Représentation snowflake schema .....	40
Figure 18 Représentation galaxy schema .....	41
Figure 19 Schéma de la data warehouse .....	42
Figure 20 Schéma de la data warehouse complet .....	42
Figure 21 Code ET_projects 1 .....	45
Figure 22 Code ET_Projects 2 .....	46
Figure 23 Excel des projets .....	47
Figure 24 Code ET_Fields 1 .....	47
Figure 25 Code ET_Fields 2 .....	48
Figure 26 Code ET_Fields 3 .....	49
Figure 27 Excel des issues .....	50
Figure 28 Code ET_Comments 1 .....	50
Figure 29 Code ET_Comments 2 .....	51
Figure 30 Code ET_Comments 3 .....	53
Figure 31 Code ET_Comments 4 .....	54
Figure 32 Excel des commentaires .....	55
Figure 33 Code Comments ID .....	56
Figure 34 Excel Comment ID .....	57
Figure 35 Code de Transformation des commentaires .....	57
Figure 36 Excel final des commentaires .....	59
Figure 37 Création de la DB SA .....	60
Figure 38 Job Dim_Project .....	61
Figure 39 Composante Tmap Project .....	61
Figure 40 Composante DBOutput Projects .....	62

Figure 41 Table Projects SA.....	62
Figure 42 Composante Tmap Issues.....	63
Figure 43 Table Issues SA.....	63
Figure 44 Composante Tmap Comment.....	64
Figure 45 Table Comments SA .....	64
Figure 46 Création de la DB DW .....	65
Figure 47 Job Dim_DW_Issues.....	66
Figure 48 Job Fact_Table .....	66
Figure 49 Composante Tmap Fact Table .....	67
Figure 50 Fact Table.....	68
Figure 51 Config elasticseach 1 .....	69
Figure 52 Config elasticseach 2 .....	69
Figure 53 Config elasticseach 3 .....	70
Figure 54 Config elasticseach 4 .....	70
Figure 55 Config elasticseach 5 .....	71
Figure 56 Interface Kibana .....	71
Figure 57 Index pgindex.....	72
Figure 58 Connecteur pgconnector.....	72
Figure 59 Connection pgindex .....	73
Figure 60 Config connecteur 1 .....	74
Figure 61 Config connecteur 2 .....	74
Figure 62 Config postgresql 1 .....	75
Figure 63 Config postgresql 2 .....	75
Figure 64 Connection réussite .....	76
Figure 65 Synchronisation complète .....	77
Figure 66 Documents dans pgindex .....	78
Figure 67 Dashboard Issue 1 .....	80
Figure 68 Top 5 Assignés les Plus Performants .....	81
Figure 69 Nombre Total d'Issues et Issues Résolues par Type .....	82
Figure 70 KPI Issues Résolues .....	82
Figure 71 Nombre Total d'Issues par Composants et Type.....	83
Figure 72 Temps Moyen de Résolution par Priorité .....	84
Figure 73 KPI Temps Moyen de Résolution .....	84
Figure 74 Dashboard Issue 2 .....	85
Figure 75 KPI Nombre Moyen de Commentaire par Issue .....	85
Figure 76 Mesure Estimated Comment Count .....	86
Figure 77 Tableau Issues .....	86
Figure 78 Tooltip Issue.....	87
Figure 79 Évolution du Nombre Total d'Issues, d'Issues Résolues et du Nombre de Commentaires.....	87
Figure 80 Evolution du Nombre Total d'Issue, d'Issues Résolues et du Nombre de Commentaire .....	88
Figure 81 Évolution du Taux de Résolution avec le Taux de Résolution Cible .....	89

Figure 82 KPI Taux de Résolution .....	89
Figure 83 Nombre de Commentaires par Type et Priorité .....	90
Figure 84 Dashboard Projet.....	91
Figure 85 Nombre Total d'Issues et Temps Moyen de Résolution par Projet.....	91
Figure 86 Top 5 Projets par Score de Commentaire .....	92
Figure 87 KPI Score de Commentaire.....	93
Figure 88 Mesure Valid Comment Count .....	93
Figure 89 Nombre Total d'Issues par Projet et Priorité .....	94
Figure 90 Top 5 Projets par Score .....	94
Figure 91 Tooltip Projet .....	95
Figure 92 KPI Score du Projet.....	95
Figure 93 Code d'indexation des embeddings 1 .....	96
Figure 94 Code d'indexation des embeddings 2.....	97
Figure 95 Code d'indexation des embeddings 3 .....	98
Figure 96 Code de recherche Elasticsearch 1 .....	99
Figure 97 Code de recherche Elasticsearch 2 .....	99
Figure 98 Code de recherche Elasticsearch 3 .....	100
Figure 99 Code de recherche Elasticsearch 4 .....	101
Figure 100 Code de recherche Elasticsearch 5 .....	102
Figure 101 Code de préparation de la data 1 .....	105
Figure 102 Code de préparation de la data 2 .....	106
Figure 103 Excel du jeu de données.....	107
Figure 104 Code de l'entraînement du modèle 1 .....	107
Figure 105 Code de l'entraînement du modèle 2.....	108
Figure 106 Code de l'entraînement du modèle 3 .....	109
Figure 107 Résultat de l'entraînement .....	110
Figure 108 Métrique train/learning_rate.....	111
Figure 109 Métrique train/loss .....	111
Figure 110 Métrique eval/validation_set_evaluator_pearson_cosine .....	112
Figure 111 Métrique eval/validation_set_evaluator_spearman_cosine .....	113
Figure 112 Code de l'évaluation du modèle 1 .....	114
Figure 113 Code de l'évaluation du modèle 2 .....	115
Figure 114 Résultat de l'évaluation .....	116
Figure 115 Code Flask-Python .....	117
Figure 116 Front end : Aperçu du projet .....	118
Figure 117 Front end : Recherche Elasticsearch .....	118
Figure 118 Front end : Résultat de la recherche .....	119
Figure 119 Front end : Dashboard Power BI.....	119

# Introduction générale

Dans un monde où la gestion des données joue un rôle central, l'optimisation des processus liés à l'analyse et à la recherche devient indispensable pour assurer une prise de décision rapide et efficace. Les entreprises sont de plus en plus confrontées à des volumes massifs de données qu'elles doivent exploiter pour en extraire des informations précieuses. Dans ce contexte, ce projet vise à développer une solution intégrée pour la collecte, la gestion, l'analyse et la recherche des données issues de systèmes tels que Jira Service Management.

L'objectif principal de ce projet est de construire une architecture unifiée comprenant un entrepôt de données robuste, des visualisations intuitives via Power BI, et un moteur de recherche performant grâce à Elasticsearch. Ce système doit permettre de répondre aux besoins spécifiques liés à l'analyse des tickets Jira et à l'évaluation des performances des projets tout en optimisant la recherche avancée sur des données structurées et non structurées.

Ce rapport est structuré en six chapitres, précédés d'une introduction générale et suivis d'une conclusion générale qui résume les résultats obtenus et les perspectives d'amélioration.

**Contexte Général :** Ce chapitre fournit une vue d'ensemble des sujets abordés, présente l'organisme d'accueil et le cadre du projet. Il introduit également la problématique et la solution proposée, tout en justifiant la méthodologie adoptée pour ce projet.

**Définition des Besoins Métiers :** Cette section explore les besoins métiers spécifiques au projet, en identifiant les objectifs de recherche et de visualisation des données. Elle présente également les types de recherches et analyses à réaliser via Elasticsearch.

**Conception des Données :** Ce chapitre traite de la modélisation des données, en expliquant comment elles seront structurées et organisées pour garantir une intégration optimale dans l'entrepôt de données.

**Chargement des Données (ETL) :** Cette partie aborde le processus d'extraction, transformation et chargement des données depuis leurs sources jusqu'à l'entrepôt de données. Les techniques et outils employés sont détaillés.

**Accès aux Données et Analyse :** Ce chapitre se concentre sur les tableaux de bord Power BI pour la visualisation des données ainsi que sur l'implémentation d'une application Python pour des recherches avancées avec Elasticsearch.

**Maintenance et Optimisation :** Ce dernier chapitre discute des processus de maintenance de la solution et des optimisations nécessaires pour garantir sa performance à long terme. Il présente également l'intégration d'une interface front-end pour une expérience utilisateur complète.

**Conclusion et Perspectives :** Enfin, ce rapport se conclut par une synthèse des résultats obtenus et des recommandations pour les travaux futurs, mettant en lumière les contributions du projet et ses axes d'amélioration potentiels.

Avec cette structure, notre ambition est d'offrir une vue complète et détaillée du projet, de la conceptualisation à la mise en œuvre, en mettant en avant l'intégration des outils modernes tels qu'Elasticsearch et Power BI pour une gestion et une analyse avancée des données.

# Chapitre 1 : Contexte général

1.	Introduction .....	13
2.	Organisme d'accueil .....	13
3.	Présentation du projet.....	14
3.1.	Problématique.....	14
3.2.	L'étude de l'existant .....	15
4.	Solution proposée.....	18
5.	Méthodologie de travail .....	18
5.1.	Méthodologie Agile pour la BI .....	18
5.2.	Méthodologie Kimball BI .....	19
5.3.	Méthodologie Lean BI .....	19
5.4.	Méthodologie choisie .....	20
6.	Conclusion.....	21

## 1. Introduction

L'objectif de ce premier chapitre est de présenter le contexte général de mon projet de fin d'études. Nous aborderons dans un premier temps l'augmentation constante des tickets de support IT, ainsi que les défis liés à leur gestion et à leur résolution. Par la suite, nous décrirons les systèmes existants pour la gestion des tickets et les limites rencontrées. Enfin, nous formulerons la problématique et présenterons la solution proposée, qui combine une analyse approfondie des données via Power BI et une recherche avancée grâce à l'intégration d'Elasticsearch.

## 2. Organisme d'accueil

La société Vermeg, basée aux Berges du Lac à Tunis, a vu le jour en 2002 en tant qu'entité autonome issue de BFI, une entreprise fondée en 1994. Spécialisée dans le domaine de la monétique et dans le développement de solutions logicielles bancaires et financières, Vermeg offre des outils performants dédiés à la gestion des titres et des capitaux. Depuis sa création, l'entreprise s'est continuellement investie dans l'approfondissement de son expertise en finance, proposant une gamme variée de logiciels innovants qui répondent aux besoins des middle et back-offices des institutions traitant des titres.

À ce jour, Vermeg dessert plus de 550 clients répartis dans une quarantaine de pays, parmi lesquels figurent 15 des 20 plus grandes banques au monde. L'entreprise consacre plus de 30 % de son effectif aux activités de recherche et développement. Elle est implantée dans plusieurs régions du globe, notamment en Australie, Belgique, Brésil, France, Allemagne, Hong Kong, Japon, Luxembourg, Pays-Bas, Singapour, Espagne, Tunisie, Royaume-Uni et États-Unis.

Ses clients dans le domaine financier incluent des institutions renommées telles que la Banque de France, la Bank of England, Natixis, Santander Securities Services, SEK, Raiffeisen Bank International, Société Générale, Nordea, CACEIS Investor Services, One Savings Bank, Deutsche Bank et Northern Trust.

Dans le secteur des assurances, Vermeg collabore avec des acteurs majeurs tels qu'ABN-AMRO, AG2R La Mondiale, Allianz, AXA, Nationale-Nederlanden, MILLESIS Banque, Cardiff Lux Vie (groupe BNP Paribas), MAIF, Generali, Suravenir et Zurich.

Pour répondre aux attentes les plus exigeantes de ses clients, Vermeg s'appuie sur une série de distinctions de marché, certifications et accréditations de produits.

En 2023, Vermeg a été reconnue comme leader dans la gestion de garanties par *Chartis RiskTech Quadrant* et a reçu une distinction pour son innovation. En 2021, elle a remporté le prix de la meilleure initiative de gestion des données aux *American Financial Technology Awards*. Cette même année, les *Central Banking Awards* ont salué sa solution de gestion de garanties, tandis que *FStech* a récompensé son utilisation innovante des données. Vermeg a également été reconnue parmi les fournisseurs de solutions de transformation numérique et son produit d'optimisation de garanties a été désigné *Produit de l'année* par les *Risk Markets Technology Awards*. Toujours en 2021, *Waters Technology* a distingué Vermeg pour la meilleure solution de reporting. En 2018, les *Asian Private Banker Awards for Distinction* ont mis en avant sa solution de reporting réglementaire, et en 2017, les *Custody Risk Global Awards*

ainsi que *Lombard Risk* ont respectivement reconnu Vermeg comme le meilleur fournisseur et la meilleure plateforme de garanties.

Au niveau des certifications, Vermeg est certifiée **Great Place to Work** depuis 2023 et figurait parmi les **Best Places to Work** en 2022. En 2023, elle a obtenu la certification **Ecovadis** pour ses performances en matière de responsabilité sociale, ainsi que la qualification **FSQS-NL** pour le système de qualification des fournisseurs. Vermeg est également partenaire officiel **AWS**, détentrice des certifications **SOC 2 Type 2** et **AICPA SOC**, et membre actif du **Pacte Mondial des Nations Unies** avec une reconnaissance particulière de **EY**.

Côté produits, Vermeg propose des solutions conformes aux normes **Swift**, couvrant les domaines des actions corporatives, de la gestion des garanties, de la réconciliation et du règlement des titres.

### 3. Présentation du projet

Nous allons à présent exposer la problématique du projet ainsi que l'analyse de l'existant.

#### 3.1. Problématique

Le projet "Jira Data" vise à optimiser la gestion des tickets de Jira Service Management en utilisant l'analyse et la recherche de données. Avant la mise en œuvre de ce projet, l'environnement de gestion des tickets reposait principalement sur des processus manuels et une intervention humaine intensive. Voici un aperçu de l'état actuel avant l'introduction du projet "Jira Data" :

1. Processus Manuels : La gestion des tickets repose principalement sur des processus manuels, ce qui entraîne des inefficacités en raison de la nécessité d'une intervention humaine à chaque étape du processus, depuis la réception initiale des tickets jusqu'à leur résolution.
2. Surcharge de Tickets : L'équipe fait face à une forte réception de tickets, ce qui entraîne une surcharge de travail et une dispersion des ressources. Cette situation conduit souvent à des retards dans le traitement des requêtes et à une baisse de la satisfaction client.
3. Redondance des Tickets : Un problème récurrent est la redondance des tickets, où plusieurs utilisateurs signalent les mêmes problèmes ou demandent des solutions similaires. Cela entraîne un gaspillage de ressources, car les agents doivent résoudre les mêmes problèmes à plusieurs reprises.
4. Manque d'Analyse des Données : Il y a un manque d'analyse approfondie des données de Jira pour identifier les tendances, les schémas récurrents et les opportunités d'automatisation. Sans cette analyse, il est difficile d'optimiser efficacement les processus de gestion des tickets.
5. Temps de Réponse Prolongé : En raison de la surcharge de tickets et de la redondance des tâches, les temps de réponse aux requêtes des utilisateurs sont souvent prolongés, ce qui nuit à la satisfaction client et à la réactivité de l'équipe de support.

L'environnement de gestion des tickets se caractérisait par des processus manuels, une surcharge de tickets, une redondance des tâches, un manque d'analyse des données et des temps de réponse prolongés. Ces défis ont un impact négatif sur l'efficacité opérationnelle et la satisfaction client, ce qui souligne la nécessité d'une solution innovante comme "Jira Data" pour améliorer la gestion des tickets de manière significative.

### 3.2. L'étude de l'existant

Actuellement, le marché propose diverses solutions d'optimisation de la gestion de Jira service management. Examinons de plus près certaines de ces solutions et identifions leurs avantages et inconvénients par rapport à notre demande :

#### Zapier



Figure 1 Logo Zapier

#### Avantages :

- Facilité d'utilisation : Zapier offre une interface conviviale et intuitive, ce qui permet aux utilisateurs de créer facilement des intégrations entre différentes applications sans avoir besoin de compétences techniques avancées.
- Large gamme d'intégrations : Zapier prend en charge des centaines d'applications populaires, offrant ainsi une grande flexibilité pour automatiser divers processus et flux de travail.
- Automatisation puissante : Zapier permet de créer des automatisations avancées en utilisant des règles conditionnelles et des actions multiples, ce qui permet d'automatiser efficacement les tâches répétitives.

#### Inconvénients :

- Limitations de fonctionnalités : Bien que Zapier soit polyvalent, certaines intégrations peuvent avoir des limitations en termes de fonctionnalités disponibles, ce qui peut limiter la complexité des automatisations.
- Coût : Les plans payants de Zapier peuvent devenir coûteux à mesure que le nombre d'intégrations et d'automatisations augmente, ce qui peut être un inconvénient pour les petites entreprises avec un budget limité.

## Jira Plugin Insight

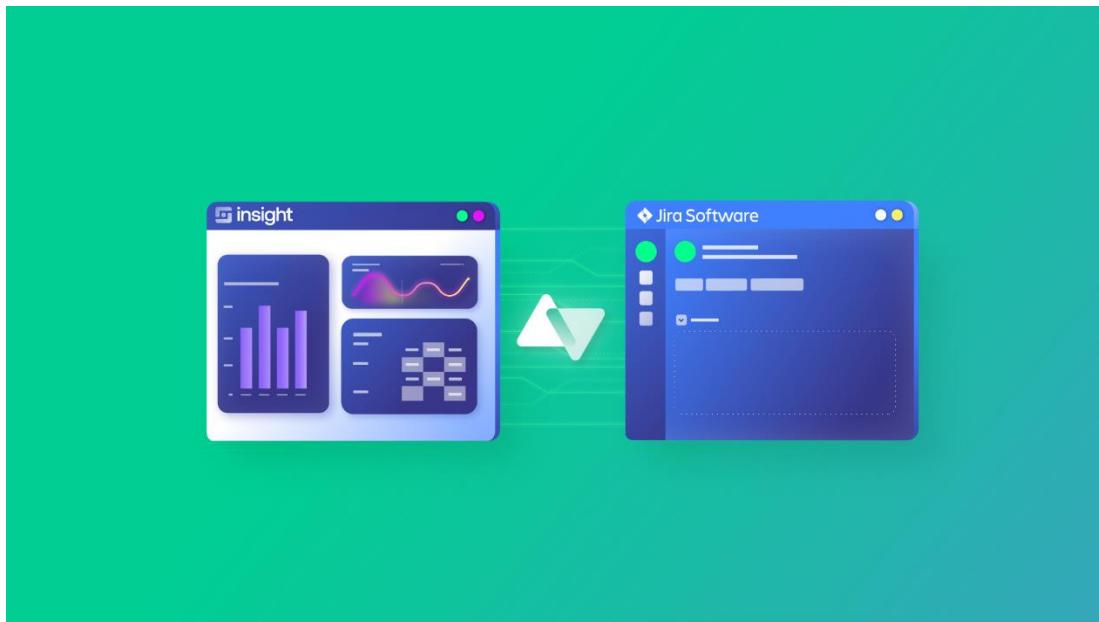


Figure 2 Logo Jira Plugin Insight

### Avantages :

- Gestion centralisée des actifs : Insight permet de gérer les actifs et les configurations au sein de Jira, offrant ainsi une vue centralisée et unifiée de tous les actifs de l'entreprise.
- Personnalisation : Insight offre une grande flexibilité en termes de personnalisation des schémas et des champs, ce qui permet de créer des structures de données adaptées aux besoins spécifiques de l'entreprise.
- Intégration avec Jira Service Management : Insight s'intègre parfaitement avec Jira Service Management, ce qui permet aux utilisateurs de créer et de gérer des tickets liés aux actifs de manière transparente.

### Inconvénients :

- Complexité : La configuration initiale d'Insight peut être complexe, en particulier pour les entreprises avec des besoins de personnalisation avancés, ce qui peut nécessiter une expertise technique.
- Coût : Les licences Insight peuvent être coûteuses, en particulier pour les grandes organisations avec de nombreux utilisateurs, ce qui peut être un inconvénient pour les entreprises avec un budget limité.

## **Externalisation du support client à un centre d'appels**

Avantages :

- Expertise spécialisée : Les centres d'appels spécialisés dans le support client offrent une expertise et une formation spécifiques dans la gestion des tickets et la satisfaction client, ce qui garantit un service de haute qualité.
- Évolutivité : Les centres d'appels peuvent être facilement adaptés pour répondre aux fluctuations de la demande, ce qui permet de maintenir des temps de réponse courts même en cas de pic d'activité.
- Concentration sur le cœur de métier : Externaliser le support client permet à l'entreprise de se concentrer sur ses activités principales, en déchargeant les ressources internes des tâches de support.

Inconvénients :

- Perte de contrôle : Externaliser le support client signifie confier une partie importante de l'expérience client à un tiers, ce qui peut entraîner une perte de contrôle sur la qualité du service.
- Coût : Les services du centre d'appels peuvent être coûteux, en particulier pour les entreprises avec un volume élevé de tickets, ce qui peut être un inconvénient pour les entreprises ayant un budget limité.
- Risque de communication : La communication entre le centre d'appels et l'entreprise peut parfois être difficile, ce qui peut entraîner des retards ou des malentendus dans la résolution des tickets.

## **Conclusion**

L'analyse des solutions disponibles pour optimiser Jira Service Management montre que chaque option a ses avantages et inconvénients. Zapier se distingue par sa facilité d'utilisation et sa flexibilité, mais son coût et ses limitations fonctionnelles peuvent poser un problème pour des besoins complexes.

Le plugin Jira Insight offre une gestion centralisée des actifs et une bonne intégration avec Jira, mais il est plus adapté aux entreprises disposant de ressources techniques et d'un budget conséquent, car sa configuration peut être complexe et coûteuse.

Enfin, l'externalisation du support client à un centre d'appels garantit une expertise et une grande capacité d'adaptation, mais cela peut entraîner une perte de contrôle, des coûts élevés et des difficultés de communication.

## 4. Solution proposée

Il est donc primordial de concevoir une solution capable de répondre aux besoins spécifiques liés à l'analyse des tickets et des projets Jira. Ce projet doit permettre d'effectuer des analyses approfondies et ciblées, offrant une visibilité claire sur les tickets, leur progression et leur gestion au sein des différents projets. De plus, cette solution doit garantir une gestion efficace des tickets grâce à une recherche rapide et performante, notamment en identifiant les tickets redondants ou similaires. L'intégration d'Elasticsearch dans cette solution sera un atout majeur pour accélérer les recherches et améliorer la précision des résultats, contribuant ainsi à une optimisation globale du système de gestion des tickets et à une meilleure prise de décision.

Ainsi, cette solution permettra de garantir un contrôle efficace, une adaptabilité aux besoins spécifiques, une configuration simple et intuitive, tout en restant économiquement avantageuse.

## 5. Méthodologie de travail

Dans le cadre de ce projet, plusieurs méthodologies adaptées à la Business Intelligence (BI) ont été explorées. Ces méthodologies permettent de structurer efficacement les étapes de collecte, d'intégration, de recherche et de visualisation des données. Les méthodologies principales retenues sont : **Agile pour la BI**, **Kimball**, et **Lean BI**. Une analyse comparative a ensuite été réalisée afin de sélectionner la méthodologie la plus appropriée, celle qui permettra d'assurer une progression efficace, une meilleure collaboration et des résultats en adéquation avec les objectifs fixés.

### 5.1. Méthodologie Agile pour la BI

#### Présentation

Agile pour la BI repose sur une approche itérative où le projet est divisé en phases successives, appelées sprints. Chaque sprint permet de développer un composant fonctionnel (par exemple, l'intégration ou la recherche avec Elasticsearch) et d'itérer en fonction des besoins.

#### Avantages

- **Flexibilité** : Permet d'ajuster rapidement les priorités en fonction des défis techniques rencontrés.
- **Livraison progressive** : Convient pour tester chaque composant indépendamment avant de les intégrer.
- **Organisation en étapes claires** : Aide à structurer le projet même sans collaboration externe.

#### Inconvénients

- **Moins pertinent pour un projet individuel** : Sans feedback externe, certaines itérations peuvent être inutiles.

- **Risque de surcharge :** L'organisation en sprints peut devenir contraignante si les phases sont mal équilibrées.

## 5.2. Méthodologie Kimball BI

### Présentation

Kimball propose une approche orientée sur les besoins fonctionnels et privilégie la création des entrepôts de données (Data Warehouse) optimisés pour des usages spécifiques. Dans ce projet, cela correspond à la structuration des données pour la recherche dans Elasticsearch et leur visualisation.

### Avantages

- **Conception modulaire :** Permet de structurer les données de manière claire et optimisée pour des tâches spécifiques (recherche ou visualisation).
- **Rapide à mettre en œuvre :** Les fonctionnalités peuvent être développées indépendamment.
- **Adapté à des objectifs bien définis :** La méthode est idéale pour un projet technique où les besoins sont déjà clairs.

### Inconvénients

- **Complexité d'intégration :** Peut nécessiter un effort supplémentaire pour harmoniser les différents sous-ensembles.
- **Moins stratégique :** Se concentre sur les besoins actuels et non sur une vision globale à long terme.

## 5.3. Méthodologie Lean BI

### Présentation

La méthodologie Lean BI vise à réduire les processus inutiles et à se concentrer uniquement sur les éléments essentiels pour livrer une solution fonctionnelle. Cette approche convient bien aux projets avec des ressources limitées ou des délais serrés.

### Avantages

- **Efficace pour un projet individuel :** Se focalise sur les tâches critiques sans surcharger le processus.
- **Réduction des gaspillages :** Permet d'optimiser le temps et les efforts en priorisant les fonctionnalités les plus importantes.
- **Rapidité d'exécution :** Favorise une livraison rapide grâce à une approche minimaliste.

### Inconvénients

- **Risque de négliger des éléments :** Peut entraîner une perte de profondeur dans certains aspects de la solution.
- **Peu adapté aux besoins complexes :** Limite l'exploration des données si les objectifs deviennent plus sophistiqués.

## 5.4. Méthodologie choisie

Après analyse, la méthodologie **Kimball** a été choisie comme la plus adaptée au contexte du projet.

La nature du projet, qui nécessite une structuration claire des données pour leur intégration dans Elasticsearch et leur visualisation, s'aligne parfaitement avec l'approche modulaire de Kimball. La méthode permet de traiter chaque composant indépendamment tout en garantissant une cohérence globale.

Agile pour la BI, bien que flexible, dépend fortement des interactions avec des parties prenantes, absentes dans ce projet.

Lean BI, bien qu'efficace pour des projets simples, pourrait limiter la profondeur et la structuration nécessaires pour un système complexe impliquant des recherches avancées et des visualisations interactives.

### **Kimball**

La méthodologie Kimball se divise en plusieurs étapes essentielles, qui permettent de structurer un projet BI de manière efficace, tout en répondant aux besoins des utilisateurs. Voici les principales étapes suivies dans ce projet.

#### **Définition des besoins métiers (Business Requirements)**

Cette étape consiste à définir les objectifs de recherche et de visualisation des données collectées, ainsi que les types de recherches et analyses qui seront réalisées via Elasticsearch.

#### **Conception des données (Data Design)**

Cette étape concerne notamment la modélisation des données, définir la manière dont les données seront structurées et organisées pour faciliter l'intégration dans l'entrepôt de données.

#### **Chargement des données (ETL : Extraction, Transformation, Chargement)**

Lors de cette étape, Nous allons extraire les données de différentes sources, les transformer pour les rendre compatibles avec le modèle de données, puis les charger dans La Staging Area et la Data Warehouse ainsi que l'indexation des données dans Elasticsearch

#### **Accès aux données et analyse (Data Access and Analysis)**

Cette étape inclut la création des visualisations dans des outils comme Power BI, et la création de la recherche sur les données intégrées dans Elasticsearch. Les utilisateurs ainsi peuvent explorer les résultats de recherche, filtrer les données et générer des rapports interactifs.

#### **Maintenance et optimisation (Maintenance and Optimization)**

Dans la dernière étape, nous allons assurer l'optimisation continue des index dans Elasticsearch pour répondre efficacement aux requêtes ainsi que la mise en place de la plateforme FrontEnd du projet.

## 6. Conclusion

Dans ce chapitre, nous avons présenté l'organisme d'accueil, réalisé une étude préalable de la problématique ainsi qu'une analyse de l'existant. Enfin, nous avons comparé différentes méthodologies et exposé celle qui a été retenue comme la plus adaptée.

Ensuite, nous aborderons la définition des besoins métier, la présentation des outils utilisés, ainsi que les objectifs spécifiques du projet.

## Chapitre 2 : Définition des besoins métiers

1.	Introduction .....	23
2.	Objectifs stratégiques de l'entreprise .....	23
2.1.	Centralisation des données pour une gestion optimisée.....	23
2.2.	Amélioration des capacités de recherche et d'analyse .....	23
2.3.	Visualisation des données pour faciliter la prise de décision.....	23
2.4.	Amélioration de l'efficacité opérationnelle .....	23
2.5.	Évolutivité et adaptabilité.....	24
3.	Objectifs généraux et spécifiques du projet .....	24
3.1.	Intégration des données des issues Jira .....	24
3.2.	Collecte et centralisation des commentaires Jira.....	25
3.3.	Gestion des données des projets Jira .....	26
3.4.	Amélioration de la productivité et de la prise de décision .....	27
3.5.	Objectifs stratégiques liés aux données :.....	28
4.	Métriques clés et indicateurs de performance .....	28
5.	Technologies utilisées .....	29
5.1.	Environnement matériel .....	29
5.2.	Environnement logiciel .....	30
	Conclusion .....	30

## 1. Introduction

Dans ce chapitre, nous allons aborder plusieurs aspects fondamentaux pour la réussite du projet. Tout d'abord, nous identifierons les objectifs stratégiques de l'entreprise afin de nous aligner sur ses priorités globales. Ensuite, nous définirons les objectifs spécifiques du projet en fonction des données à collecter, tout en déterminant les métriques clés et les indicateurs de performance qui permettront de mesurer son succès. Par ailleurs, une priorisation des besoins sera effectuée en fonction de la valeur qu'ils apportent à l'entreprise. Enfin, nous présenterons les outils qui seront utilisés pour atteindre ces objectifs de manière efficace et structurée.

## 2. Objectifs stratégiques de l'entreprise

Le projet s'inscrit dans une démarche stratégique visant à optimiser la gestion des données et à faciliter la prise de décision. Les principaux objectifs stratégiques sont les suivants :

### 2.1. Centralisation des données pour une gestion optimisée

- Rassembler toutes les données provenant de différentes sources (des données sur les Issues, commentaires, projets, etc.) dans une plateforme unique, facilitant ainsi leur accès et leur utilisation.
- Garantir une intégration fluide des données pour réduire la fragmentation et les silos d'information.

### 2.2. Amélioration des capacités de recherche et d'analyse

- Mettre en place un système de recherche performant avec Elasticsearch, permettant d'interroger les données efficacement grâce à des fonctionnalités avancées (recherche full-text, filtres).
- Optimiser l'exploration des données pour répondre rapidement aux besoins spécifiques, tels que le suivi des projets, l'analyse des performances, ou la résolution d'incidents.

### 2.3. Visualisation des données pour faciliter la prise de décision

- Développer des tableaux de bord interactifs (via Power BI) pour transformer les données brutes en informations exploitables.
- Proposer des indicateurs clés de performance (KPI) personnalisés pour suivre l'évolution des projets, la résolution des problèmes, et la satisfaction des utilisateurs.

### 2.4. Amélioration de l'efficacité opérationnelle

- Réduire le temps consacré à la recherche d'informations grâce à une organisation claire et à des outils performants.
- Automatiser les processus d'intégration et d'analyse pour limiter les tâches répétitives et améliorer la productivité.

## 2.5. Évolutivité et adaptabilité

- Créer un système flexible qui peut évoluer pour répondre à de nouveaux besoins ou intégrer de nouvelles sources de données.
- Prévoir la capacité d'adapter les fonctionnalités de recherche et de visualisation aux besoins futurs, sans nécessiter une refonte complète du système.

## 3.

### Objectifs généraux et spécifiques du projet

Voici le schéma que nous avons créé pour les objectifs du projet :

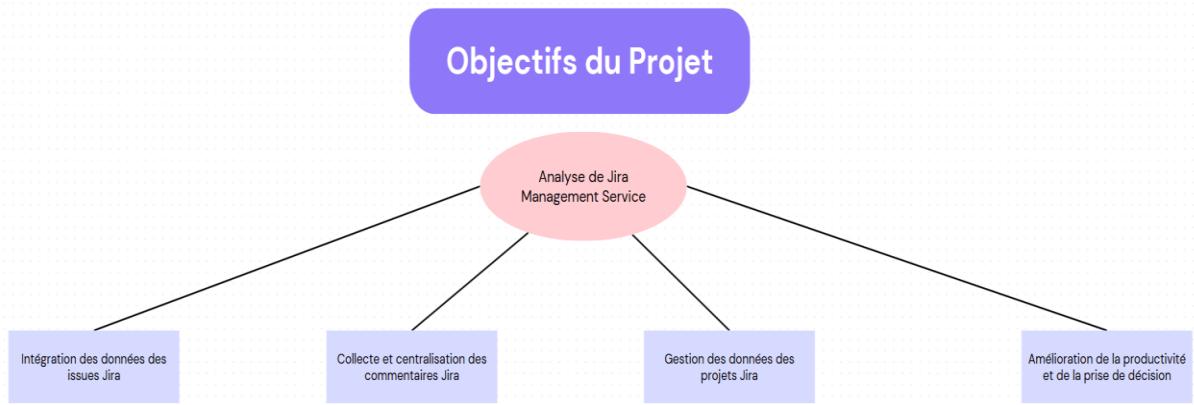
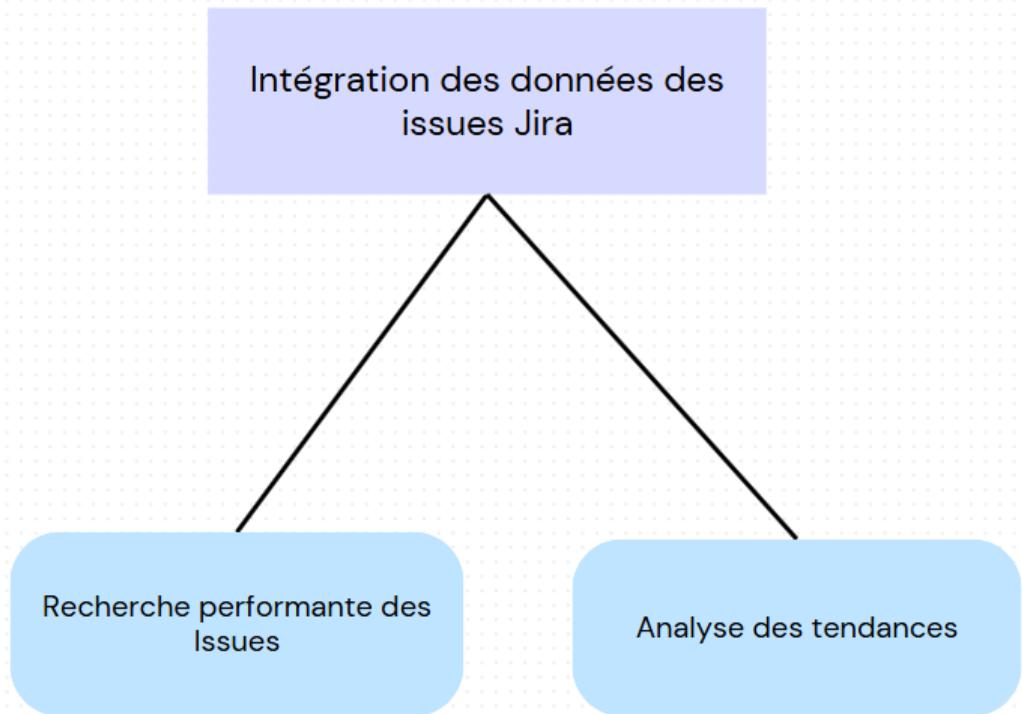


Figure 3 Objectifs généraux

#### 3.1. Intégration des données des issues Jira

Le projet vise à collecter et organiser les informations détaillées des issues Jira, telles que leur nom, description, résumé, priorité, et type, etc... Ces données permettent de suivre précisément les incidents en cours, de détecter les problèmes récurrents, et d'effectuer des recherches ciblées pour identifier rapidement les issues critiques. Une structuration efficace de ces données contribue également à une analyse approfondie pour optimiser les processus de résolution.



*Figure 4 Intégration des données des issues Jira*

#### **Données collectées :**

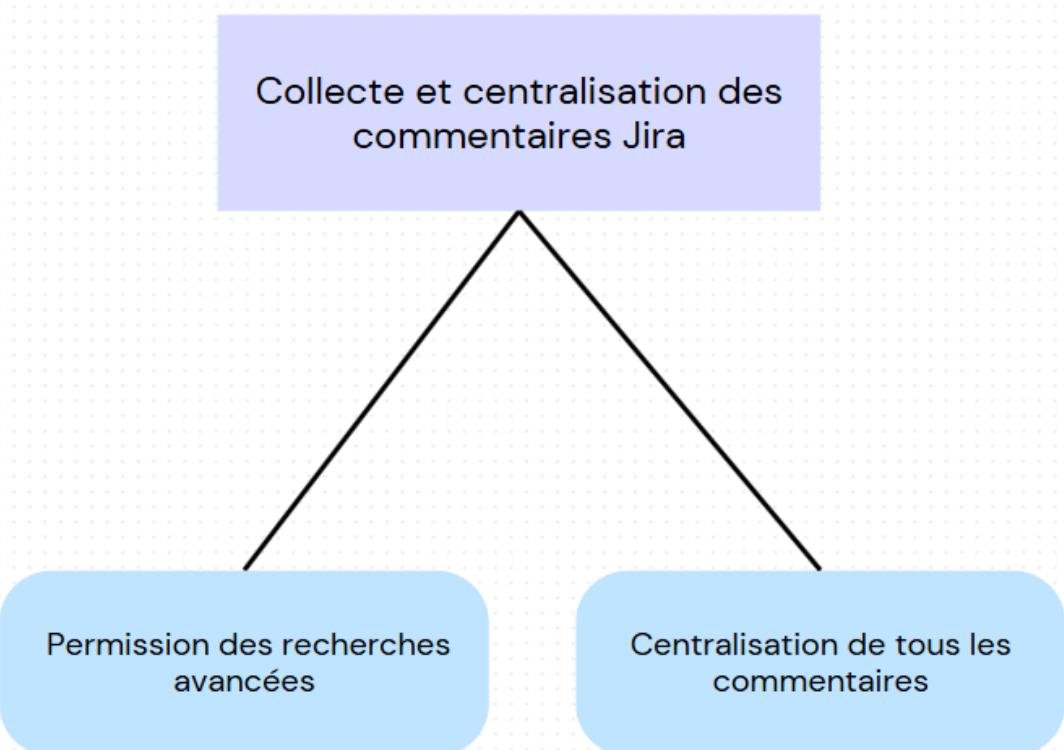
- Nom, description, résumé, priorité, type des issues, date de création et résolution.

#### **Objectifs :**

- **Analyse des tendances :** Analyser les types de problèmes récurrents pour proposer des améliorations des processus ou outils.
- **Recherche performante :** Mettre en place un système de recherche permettant de rechercher les Issues en se basant sur la description et le résumé de chacune.

### **3.2. Collecte et centralisation des commentaires Jira**

L'un des objectifs clés du projet est de rassembler tous les commentaires associés aux issues Jira dans une base de données centralisée. Cela permet de disposer d'un historique complet des discussions, décisions et interactions pour chaque problème. Cette centralisation facilite la recherche rapide et l'analyse des commentaires par Issue offrant ainsi une meilleure compréhension des échanges autour des incidents.



*Figure 5 Collecte et centralisation des commentaires Jira*

#### **Données collectées :**

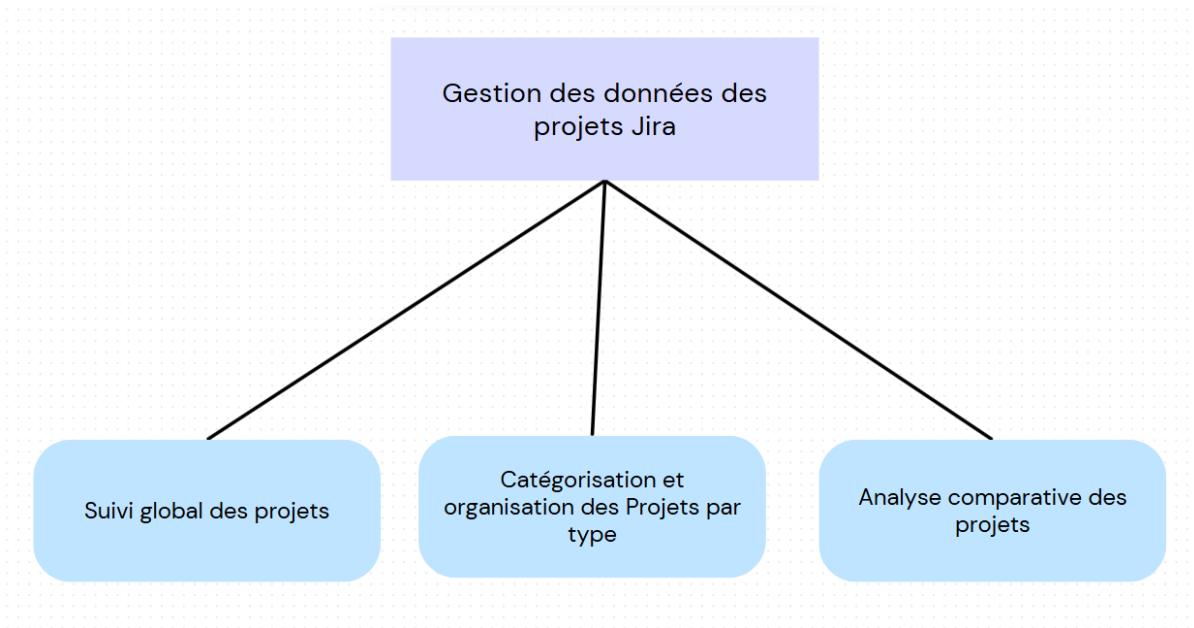
- Les commentaires associés aux issues, incluant le contenu textuel, les auteurs et les dates.

#### **Objectif :**

- Centraliser tous les commentaires pour fournir une vue complète de l'historique des interactions sur chaque issue, facilitant ainsi la compréhension des discussions et des décisions prises.
- Permettre des recherches avancées (par Issue) pour une analyse rapide et efficace des conversations.

### **3.3. Gestion des données des projets Jira**

Les données des projets Jira, incluant leur nom, description, type et catégorie, sont intégrées pour fournir une vue d'ensemble claire sur les initiatives en cours. Cette gestion des projets aide à mieux organiser les informations, à comparer les performances des différents projets, et à identifier les priorités stratégiques. Une telle organisation assure une meilleure lisibilité et une exploitation optimale des données.



*Figure 6 Gestion des données des projets Jira*

#### **Données collectées :**

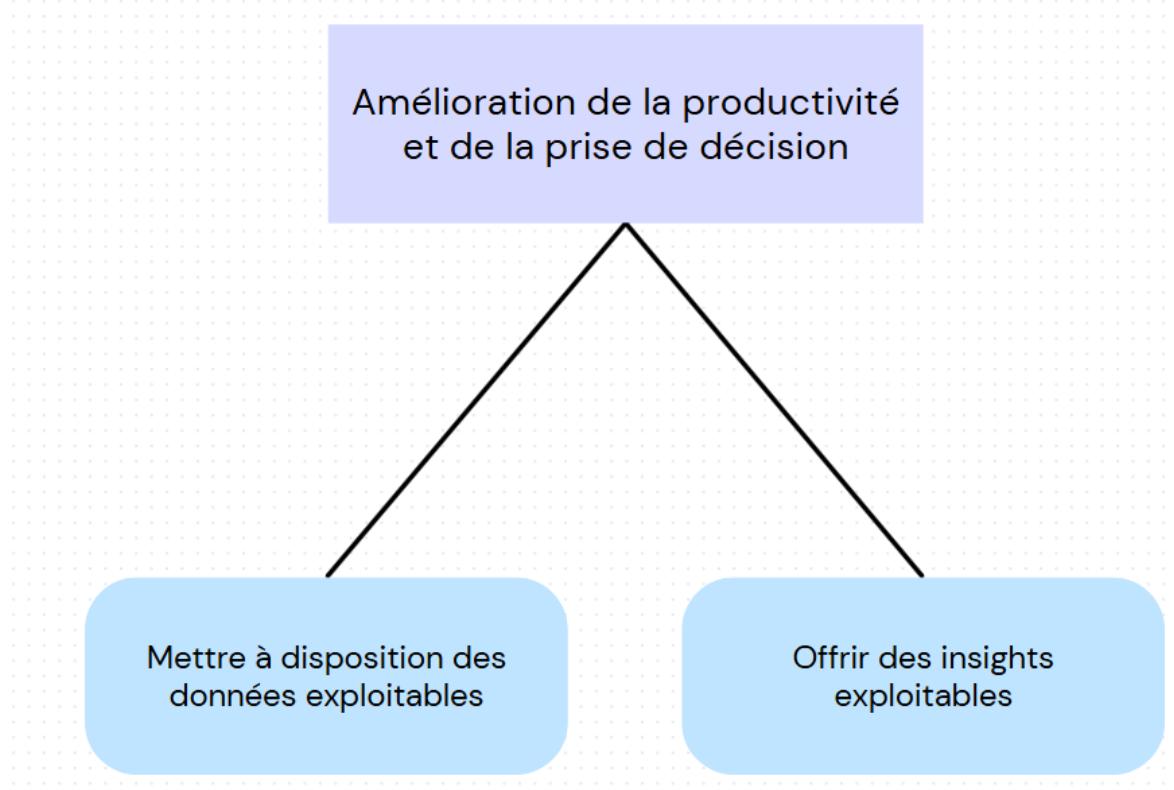
- Nom, description, type et catégories des projets.

#### **Objectifs :**

- **Suivi global des projets** : Fournir une vue d'ensemble sur l'état et les caractéristiques des différents projets gérés dans Jira.
- **Catégorisation et organisation** : Structurer les projets par catégorie ou type pour une meilleure organisation des données.
- **Analyse comparative** : Comparer les projets en termes de volume d'issues, de priorité des problèmes, ou de performance (résolution des issues).

### **3.4. Amélioration de la productivité et de la prise de décision**

En intégrant et structurant toutes ces données dans un système centralisé et performant, le projet contribue à accélérer la recherche d'informations et à réduire le temps passé sur des tâches répétitives.



*Figure 7 Amélioration de la productivité et de la prise de décision*

Les données organisées et accessibles permettent également de générer des insights exploitables via des visualisations et des indicateurs clés, facilitant ainsi la prise de décision éclairée et l'amélioration des performances globales.

### 3.5. Objectifs stratégiques liés aux données :

- Mettre à disposition des données exploitables pour réduire le temps nécessaire à la recherche d'informations spécifiques (par exemple, identifier rapidement les issues à haute priorité).
- Offrir des insights exploitables via des indicateurs clés, tels que :
  - Le nombre d'issues ouvertes par projet.
  - Les commentaires par type d'issue.
  - La répartition des priorités.

## 4. Métriques clés et indicateurs de performance

Nous allons maintenant présenter brièvement les métriques clés et les indicateurs de performance, en fournissant une brève description de chacun. Ces éléments serviront à évaluer l'efficacité du projet et à mesurer son impact sur les objectifs de l'entreprise.

1. **Taux de résolution des Issues** : Cet indicateur mesure le pourcentage d'Issues résolues par rapport au nombre total d'Issues ouvertes. Il permet d'évaluer l'efficacité du support et la gestion des demandes.
2. **Temps moyen de résolution** : Cette métrique calcule le temps moyen nécessaire pour résoudre une Issue, offrant une vue sur la rapidité du processus de gestion d'Issue.
3. **Nombre moyen de commentaires par issue** : Cette métrique calcule le nombre moyen de commentaires ajoutés à chaque Issue. Elle permet d'évaluer l'interaction et l'implication des parties prenantes dans la résolution des problèmes.
4. **Score de performance de l'assignée** : Cette métrique évalue la performance des assignés en fonction de critères tels que le nombre d'Issues résolues, le respect des délais et la qualité des résolutions. Elle aide à suivre l'efficacité individuelle au sein de l'équipe.
5. **Score du projet** : Cette métrique mesure la performance globale d'un projet, en prenant en compte des facteurs comme le respect des délais, le nombre d'issues ouvertes et résolus, et la satisfaction des parties prenantes. Elle permet de suivre la progression et la réussite d'un projet.
6. **Score du commentaire pour chaque issue** : Cette métrique attribue un score aux commentaires associés à chaque issue, basé sur leur nombre. Elle aide à évaluer la qualité des échanges et communication au sein des issues.
7. **Tickets résolus** : Cet indicateur suit le nombre total d'issues résolues sur une période donnée. Il permet de mesurer l'efficacité du traitement des demandes et d'évaluer la charge de travail gérée par l'équipe.
8. **Tickets à haute priorité** : Cette métrique suit le nombre d'issues classées comme étant de haute priorité. Elle permet de suivre la gestion des incidents critiques et de s'assurer qu'ils sont traités en priorité.

Ces métriques sont essentielles pour suivre l'efficacité du système de gestion des tickets et pour identifier les domaines où des améliorations peuvent être apportées.

## 5. Technologies utilisées

Pour notre projet, nous avons utilisé des technologies diverses et robustes, conçues pour assurer une analyse efficace des données ainsi qu'une visualisation claire et pertinente.

### 5.1. Environnement matériel

- ❖ **Processeur** : Intel(R) Core(TM) i5-10300H CPU @ 2,50 GHz 2,50 GHz
- ❖ **RAM installée** : 8,00 Go (7,84 Go utilisables)
- ❖ **Disque** : SSD 512 Go
- ❖ **Système d'exploitation** : Windows 11 Home Single Language
- ❖ **Type de système** : Système d'exploitation 64 bits, processeur basé sur x64

## 5.2. Environnement logiciel

- ❖ **Jupyter Notebook (Python)** : Utilisé pour les processus ETL (Extract, Transform, Load), il permet d'écrire et d'exécuter des scripts Python pour l'extraction, la transformation et le chargement des données dans des formats adaptés à leur exploitation.
- ❖ **Microsoft Excel** : Employé pour le stockage initial des données, notamment lors des phases de collecte ou de préparation des informations avant leur intégration dans des outils plus complexes.
- ❖ **Talend** : Une plateforme puissante pour l'intégration des données, utilisée pour automatiser l'import et la transformation des informations provenant de multiples sources vers une base de données ou un entrepôt de données.
- ❖ **PostgreSQL** : Une base de données relationnelle robuste, utilisée pour le stockage structuré des données, permettant un accès rapide et efficace pour les étapes ultérieures du projet.
- ❖ **WSL (Windows Subsystem for Linux)** : Utilisé pour établir une connexion entre PostgreSQL et Elasticsearch, facilitant ainsi les interactions et l'intégration des données entre les deux systèmes.
- ❖ **Elasticsearch** : Un moteur de recherche et d'analyse distribué, utilisé pour indexer et rechercher rapidement les données collectées, offrant des performances optimales pour les besoins du projet.
- ❖ **Kibana** : L'interface graphique d'Elasticsearch, utilisée pour visualiser, explorer et analyser les données indexées à travers des tableaux de bord interactifs.
- ❖ **Visual Studio Code (VSCode)** : Un éditeur de code léger et performant, utilisé pour développer la solution de recherche en Python, intégrant Elasticsearch pour garantir des performances optimales.
- ❖ **Jira Hibernate** : Source principale des données, utilisée pour collecter les informations sur les projets, les issues, et les commentaires.
- ❖ **SentenceTransformers (Python Library)** : Une librairie Python puissante utilisée pour le fine-tuning du modèle SBERT et pour générer les embeddings nécessaires à la recherche sémantique.
- ❖ **Google Colab** : Utilisé pour l'entraînement du modèle SBERT nécessitant une puissance de calcul plus élevée que celle disponible localement.
- ❖ **Power BI** : Une plateforme de visualisation de données utilisée pour créer des tableaux de bord interactifs à partir des données de l'entrepôt de données.
- ❖ **Flask (Python Framework)** : Utilisé pour intégrer l'application de recherche Elasticsearch au front-end, permettant des requêtes et des réponses dynamiques via une API.
- ❖ **HTML/CSS/JavaScript** : Technologies utilisées pour développer l'interface front-end de votre plateforme, offrant une navigation intuitive entre les fonctionnalités de recherche et les tableaux de bord Power BI.

## Conclusion

Dans ce chapitre, nous avons défini les éléments essentiels pour garantir le succès du projet. Nous avons identifié les objectifs stratégiques de l'entreprise, assurant une cohérence avec ses priorités globales. Nous avons également établi les objectifs spécifiques du projet, les métriques clés, et les indicateurs de performance, permettant une évaluation précise des résultats. La priorisation des besoins a permis de concentrer les efforts sur les aspects à forte valeur ajoutée. Enfin, nous avons présenté les outils sélectionnés pour leur pertinence et leur efficacité, offrant ainsi une base solide pour la mise en œuvre du projet.

## Chapitre 3 : Conception des données

1.	Introduction .....	33
2.	Choix des données à extraire.....	33
2.1.	Source des données extraites : Hibernate Jira .....	33
3.	Dimensions de la data warehouse .....	36
3.1.	La dimension « Dim Project ».....	37
3.2.	La dimension « Dim Issue ».....	37
3.3.	La dimension « Dim Comment » .....	38
3.4.	La table de faits « Fact Table » .....	38
4.	Schéma de la data warehouse et relations .....	39
4.1.	Schémas des entrepôts de données.....	39
4.2.	Le choix du schéma en étoile .....	41
4.3.	Relations table de faits et dimensions .....	42
5.	Conclusion.....	43

## 1. Introduction

Dans ce chapitre consacré à la conception des données, nous définirons la manière dont les données ont été structurées et organisées pour assurer l'intégration optimale dans l'entrepôt de données. Nous commencerons par détailler le choix des données à extraire, en justifiant leur pertinence pour le projet. Ensuite, nous présenterons les dimensions et relations retenues pour le Data Warehouse, afin de garantir une organisation claire et cohérente. Enfin, nous proposerons le schéma de l'entrepôt de données, qui servira de base pour la modélisation et l'analyse des informations collectées.

## 2. Choix des données à extraire

Nous commencerons par introduire l'environnement à partir duquel nous avons extrait les données pour le projet.

### 2.1. Source des données extraites : Hibernate Jira

**Hibernate Jira** est une couche d'abstraction de la base de données utilisée dans Jira, qui repose sur le framework Hibernate pour faciliter l'accès aux données et la gestion des transactions. Hibernate permet de mapper les objets Java aux tables de la base de données, ce qui simplifie l'interaction avec les systèmes de stockage tout en garantissant des performances optimisées.



Figure 8 Logo Hibernate

Dans le cadre de notre projet, nous avons opté pour **Hibernate Jira** afin d'extraire les données provenant de **Jira Service Management**, et plus précisément les données liées aux issues de chaque projet. Hibernate Jira est une solution open source qui nous permet d'interagir de manière simplifiée avec l'API de Jira, facilitant ainsi l'extraction des informations nécessaires à notre projet.

L'utilisation de **Hibernate Jira** présente plusieurs avantages, notamment la possibilité d'intégrer directement les données des issues dans notre processus de collecte et d'analyse, tout en profitant des fonctionnalités avancées offertes par l'API de Jira. Étant open source, cette solution nous offre un accès libre et flexible à l'API de Jira, ce qui réduit les coûts et les contraintes liées aux licences propriétaires et permet une personnalisation selon les besoins spécifiques du projet.

## Aspects de Jira Service Management :

Voici quelques-uns des projets de Hibernate.

The screenshot shows the Jira Service Management interface. At the top, there is a navigation bar with icons for 'Projects', 'Filters', 'Dashboards', and 'Apps'. Below the navigation bar, the word 'HIBERNATE' is displayed in large, bold, yellow capital letters. The main title 'Projects' is centered above a table. The table has two columns: 'Name' and 'Key'. The 'Name' column is sorted by name (indicated by a downward arrow). The 'Key' column shows the project keys. The table lists six projects:

Name	Key
Hibernate Commons Annotations	HCANN
Hibernate ORM	HHH
Hibernate Search	HSEARCH
Hibernate Tools	HBX
Hibernate Validator	HV
Hibernate Websites	WEBSITE

Figure 9 Projets Hibernate

Ainsi, pour chaque projet, nous avons accès à la liste des issues, qu'elles soient ouvertes ou fermées, accompagnées de diverses informations essentielles. Ces informations incluent le nom de l'issue, l'assignee (la personne assignée à la tâche), la priorité, le statut, la date de création, ainsi que la date de dernière mise à jour, et bien d'autres éléments. Cette richesse de données

permet une gestion détaillée de chaque issue, facilitant le suivi de leur progression et l'identification des points nécessitant une attention particulière.

Type	Key	Summary	Assignee	Reporter	Priority	Status	Resolution	Created	Updated
<input checked="" type="checkbox"/>	HHH-18884	Cleanup of the new Maven...	Unassigned	Koen Aers	Major	OPEN	Unresolved	26 Nov 2024	26 Nov 2024
<input type="checkbox"/>	HHH-18883	When initializing...	Unassigned	klaasd	Major	OPEN	Unresolved	26 Nov 2024	26 Nov 2024
<input type="checkbox"/>	HHH-18882	Regression: unquoted cas...	Unassigned	ConsuL	Major	OPEN	Unresolved	26 Nov 2024	26 Nov 2024
<input type="checkbox"/>	HHH-18881	In MySQL, array of dates are n...	Unassigned	Davide D'Alto	Major	OPEN	Unresolved	25 Nov 2024	26 Nov 2024
<input type="checkbox"/>	HHH-18880	@ManyToOne with a ...	Gavin King	Gavin King	Major	WAITIN...	Unresolved	24 Nov 2024	24 Nov 2024
<input type="checkbox"/>	HHH-18879	Refreshing an entity to...	Unassigned	Stefano Gianelli	Major	OPEN	Unresolved	22 Nov 2024	25 Nov 2024
<input type="checkbox"/>	HHH-18878	Bad O( $n^2$ ) performance...	Unassigned	Felix König	Major	OPEN	Unresolved	22 Nov 2024	26 Nov 2024
<input checked="" type="checkbox"/>	HHH-18877	Update Oracle driver to 23.6	Christian Beikov	Christian Beikov	Major	WAITIN...	Unresolved	22 Nov 2024	22 Nov 2024
<input type="checkbox"/>	HHH-18876	ArrayInitializer# resolveInstanc...	Unassigned	Marco Belladelli	Major	AWAITI...	Unresolved	22 Nov 2024	22 Nov 2024
<input type="checkbox"/>	HHH-18875	Stop using 'Array.newInstance...	Sanne Grinovero	Yoann Rodière	Major	AWAITI...	Unresolved	22 Nov 2024	23 Nov 2024

Figure 10 Issues Hibernate

En cliquant sur une issue, il est possible de consulter les commentaires associés ainsi que la description complète de l'issue. Ces deux éléments sont essentiels pour une analyse approfondie et une recherche avancée dans Elasticsearch. Les commentaires offrent des informations contextuelles et des échanges détaillés entre les membres de l'équipe, ce qui peut être crucial pour comprendre les décisions prises, les problèmes rencontrés, et les solutions proposées. De plus, la description complète de l'issue permet de saisir le problème dans son intégralité, incluant les spécifications, les étapes pour reproduire l'incident, et d'autres détails techniques. L'intégration de ces données dans Elasticsearch permet de mettre en place des recherches flexibles et efficaces, facilitant la consultation des informations pertinentes en fonction de critères spécifiques tels que les mots-clés, la priorité, ou les dates de mise à jour.

 HHH-18856

## Incorrect Dialect auto selection when using JTOpen driver with DB2 for i

[+ Add](#)

### Description

When using JTOpen JDBC driver to connect to DB2 for i, Hibernate auto detects DB2Dialect instead of DB2iDialect.

Problems comes from org.hibernate.dialect.Database, that selects DB2iDialect only if databaseVersion starts with QSQ which is never the case when using JTOpen.

It then defaults to DB2Dialect

### Activity

Show: [All](#) [Comments](#) [History](#) [Work log](#)

Newest first ↴



Gavin King 17 November 2024 at 09:34

Please submit a fix for this as a PR on GitHub. Make sure you test it thoroughly because we won't be able to, we will just have to take your word that it works.

PS the workaround for situations like this is to set `hibernate.dialect` explicitly.

*Figure 11 Commentaires Hibernate*

Les projets eux-mêmes contiennent également des informations cruciales pour l'analyse et la gestion. Parmi ces données figurent le type de projet (par exemple, développement, maintenance, etc.), la catégorie (qui permet de classer les projets selon leur domaine ou leur priorité), ainsi que le Lead de chaque projet, c'est-à-dire la personne responsable de sa gestion. Ces éléments sont essentiels pour avoir une vision d'ensemble de l'organisation des projets, de leur portée et de leur avancement.

Ainsi, nous pouvons conclure que la collecte des données provenant de Jira Service Management couvre plusieurs aspects essentiels pour une gestion efficace des projets et des issues. D'un côté, les données relatives aux issues (ouvertes et fermées) nous permettent de suivre l'état des problèmes, leurs priorités, leurs assignées, leurs statuts, et d'autres informations pertinentes. D'un autre côté, les commentaires et la description complète des issues offrent des éléments contextuels indispensables pour une analyse approfondie et une recherche avancée. En parallèle, les données des projets (type, catégorie, lead) contribuent à une vision d'ensemble sur la gestion des projets, permettant de mieux organiser et structurer l'information.

## 3. Dimensions de la data warehouse

Après une analyse approfondie de l'environnement **Jira Service Management**, nous pouvons conclure que trois dimensions principales doivent être prises en compte pour structurer les données et garantir une gestion efficace des informations :

### 3.1. La dimension « Dim Project »

Cette dimension regroupe toutes les données relatives aux projets. Elle permet d'organiser les projets de manière claire et de suivre leur progression en fonction de leurs spécifications et de leurs priorités stratégiques. L'intégration de cette dimension permet une gestion structurée des projets et facilite les analyses globales à l'échelle de l'entreprise.

Elle contient les éléments suivants :

- ❖ **Project\_Key** : L'identificateur du projet
- ❖ **Name** : Le nom complet du projet
- ❖ **Type** : Le type du projet (software, maintenance, etc...)
- ❖ **Lead** : La personne responsable du projet
- ❖ **Category** : La catégorie du projet (Actif ou EOL)
- ❖ **Description** : Une description bref sur le projet

Dim Project
+Project_Key {id, unique}
+Name
+Type
+Lead
+Category
+Description

Figure 12 Dimension Projet

### 3.2. La dimension « Dim Issue »

Cette dimension concerne toutes les données associées aux issues Jira, qu'elles soient ouvertes ou fermées. Cette dimension permet de suivre de manière détaillée l'évolution de chaque issue, d'analyser les tendances et de prendre des décisions éclairées en fonction de leur impact sur les projets.

Elle contient les éléments suivants :

- ❖ **Issue\_Key** : L'identifiant unique de chaque issue, utilisé pour la distinguer des autres issues.
- ❖ **Project\_Key** : La clé du projet auquel l'issue est associée, permettant de lier l'issue à son contexte projet.
- ❖ **Summary** : Le résumé ou titre de l'issue, décrivant brièvement le problème ou la tâche.
- ❖ **Status** : L'état actuel de l'issue (par exemple, "Open", "In Progress", "Resolved", etc.).
- ❖ **Assignee** : La personne à qui l'issue est assignée pour résolution.
- ❖ **Created** : La date de création de l'issue, indiquant quand elle a été enregistrée dans Jira.
- ❖ **Reporter** : L'utilisateur ou la personne ayant signalé l'issue.

Dim Issue
+Issue_Key {id, unique}
+Project_Key
+Summary
+Status
+Assignee
+Created
+Reporter
+Resolution
+Priority
+Updated
+Type
+Affects_Version
+Components
+Fix_Versions
+Status Category
+Description

Figure 13 Dimension Issue

- ❖ **Resolution** : La manière dont l'issue a été résolue (par exemple, "Fixed", "Won't Fix", etc.).
- ❖ **Priority** : Le niveau de priorité de l'issue (par exemple, "High", "Medium", "Low").
- ❖ **Updated** : La date de dernière mise à jour de l'issue, indiquant le dernier moment où des modifications ont été apportées.
- ❖ **Type** : Le type de l'issue (par exemple, "Bug", "Task", "Story", etc.).
- ❖ **Affects\_Version** : La ou les versions du produit concernées par l'issue.
- ❖ **Components** : Les composants ou modules du projet touchés par l'issue.
- ❖ **Fix\_Versions** : La ou les versions dans lesquelles l'issue a été corrigée ou traitée.
- ❖ **Status Category** : La catégorie générale du statut de l'issue (par exemple, "To Do", "In Progress", "Done").
- ❖ **Description** : Une description détaillée de l'issue, expliquant le problème ou la tâche en profondeur.

### 3.3. La dimension « Dim Comment »

Les commentaires constituent une dimension à part entière, car ils contiennent des informations contextuelles cruciales pour comprendre les discussions autour des issues. Ils permettent de retracer l'historique des échanges, de comprendre les décisions prises et d'analyser les solutions proposées. Cette dimension est particulièrement importante pour une analyse qualitative des issues et pour une recherche avancée via Elasticsearch.

Elle contient les éléments suivants :

- ❖ **Comment\_Id** : L'identifiant unique attribué à chaque série de commentaires appartenant à une Issue. Cet Identifiant est incrémenté.
- ❖ **Issue\_Key** : La clé de l'issue à laquelle la série de commentaires est associée, établissant le lien entre les commentaires et le problème ou la tâche correspondante.
- ❖ **Project\_Key** : La clé du projet auquel appartient l'issue liée aux commentaires.
- ❖ **Comments** : Le contenu de la série de commentaires, incluant les discussions échangées autour de l'issue, sous forme de dialogue entre différents contributeurs.

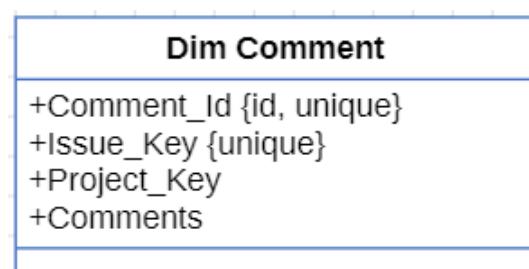


Figure 14 Dimension Commentaire

### 3.4. La table de faits « Fact Table »

La Fact Table est au cœur du modèle de données et joue un rôle central dans les analyses. Elle relie les différentes dimensions du projet tout en contenant des informations essentielles pour les métriques et les indicateurs de performance. Elle comprend les éléments suivants :

- ❖ **Issue\_Key** : L'identifiant unique de chaque issue, utilisé pour établir une relation avec les détails disponibles dans la dimension Issue. Ce champ permet de mesurer et d'analyser les performances associées à chaque issue.
- ❖ **Project\_Key** : La clé unique du projet, qui connecte les données de la fact table à la dimension Project, offrant une vue contextuelle sur les issues et les commentaires à l'échelle des projets.
- ❖ **Comment\_Id** : L'identifiant unique de chaque série de commentaires, permettant d'établir un lien direct avec la dimension Commentaire et de suivre les échanges associés à chaque issue.

En plus de ces clés relationnelles, la fact table contient des métriques et des indicateurs de performance (KPIs), tels que :

- ❖ **Issue\_Time** : Temps de résolution de l'Issue.
- ❖ **Avg Resolution Time** : Temps moyen de résolution.
- ❖ **Issue Resolution Rate** : Taux de résolution des Issues.
- ❖ **Total Issue** : Volume de tickets par projet.
- ❖ **Avg Resolution Time for Fixed Issues** : Temps moyen de résolution pour les Issues résolues.
- ❖ **Comment Count** : Nombre moyen de commentaires par issue.
- ❖ **Assignee Score** : Score de performance de l'assignée.
- ❖ **Project Score** : Score du projet.
- ❖ **Comment Score** : Score des commentaires pour chaque issue.
- ❖ **Resolved Issues** : Issues résolues.
- ❖ **High Priority Issues** : Issues à haute priorité.

Fact Table
+Issue_Key
+Comment_Id
+Project_Key
+Issue_Time
+Avg Resolution Time
+Issue Resolution Rate
+Total Issues
+Avg Resolution Time for Fixed Issues
+Comment Count
+Assignee Score
+Project Score
+Comment Score
+Resolved Issues
+High Priority Issues

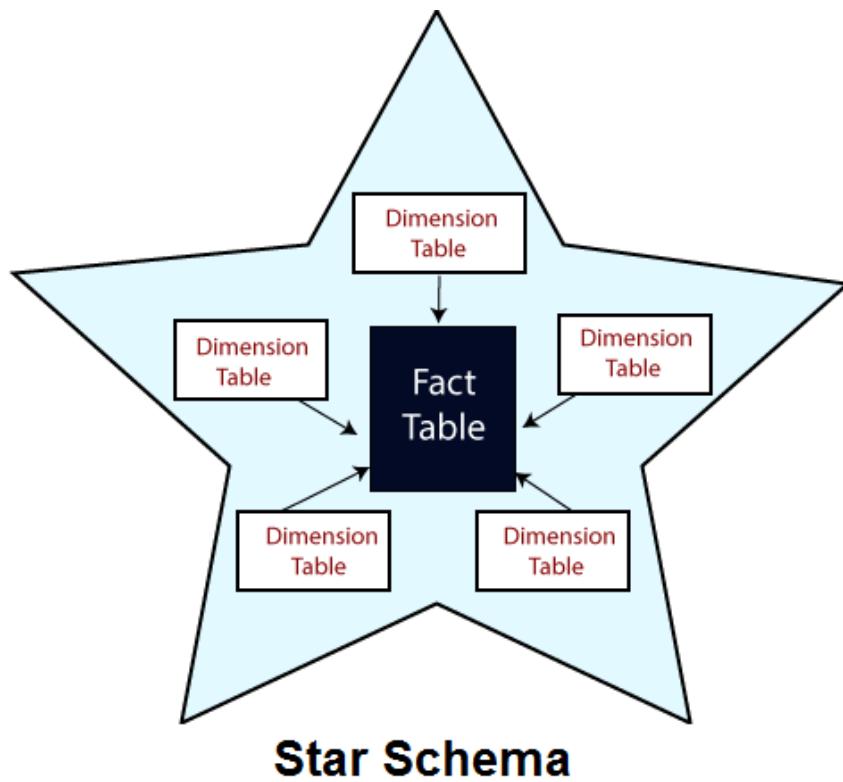
Figure 15 Fact Table

## 4. Schéma de la data warehouse et relations

### 4.1. Schémas des entrepôts de données

Dans le contexte des entrepôts de données, la structuration des informations joue un rôle essentiel pour assurer leur organisation, leur accessibilité et leur performance. Plusieurs schémas sont couramment utilisés pour concevoir un entrepôt de données :

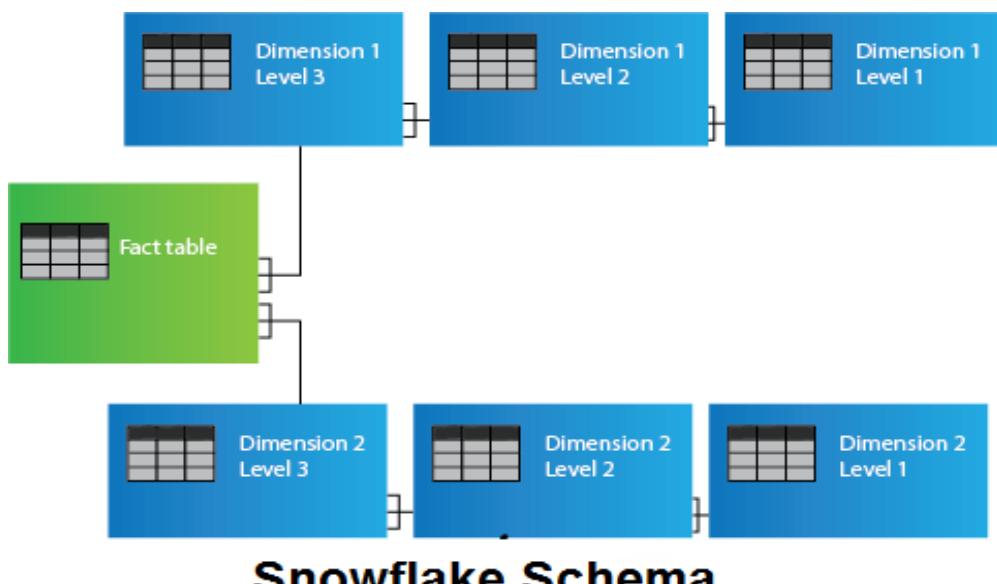
1. **Le Schéma en Étoile (Star Schema) :**



*Figure 16 Représentation star schema*

Ce schéma est centré sur une table de faits unique, qui contient les métriques et les indicateurs clés, reliée à plusieurs tables de dimensions. Chaque table de dimension fournit des informations contextuelles pour analyser les faits. Cette structure simple et intuitive permet des requêtes rapides et efficaces, particulièrement adaptée aux analyses multidimensionnelles.

## 2. Le Schéma en Flocon (Snowflake Schema) :



*Figure 17 Représentation snowflake schema*

Ce schéma est une version normalisée du schéma en étoile. Les tables de dimensions sont décomposées en sous-tables pour éviter la redondance des données. Bien qu'il réduise l'espace de stockage, ce schéma est plus complexe à gérer et peut ralentir les performances des requêtes en raison du grand nombre de jointures nécessaires.

### 3. Le Schéma en Galaxie (Galaxy Schema) :

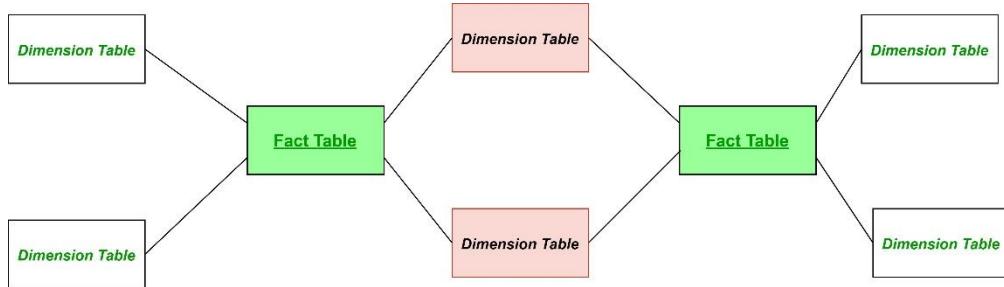


Figure 18 Représentation galaxy schema

Aussi appelé constellation de faits, ce schéma repose sur plusieurs tables de faits qui partagent des dimensions communes. Il est utilisé pour les systèmes complexes nécessitant l'analyse de plusieurs processus métier interconnectés. Bien qu'il soit puissant, ce schéma peut devenir lourd et difficile à gérer.

## 4.2. Le choix du schéma en étoile

Pour ce projet, le schéma en étoile a été retenu comme modèle de conception. Ce choix s'explique par la simplicité et l'efficacité qu'il offre. Toutes les dimensions principales (Projets, Issues, Commentaires) sont directement reliées à une table de faits unique, ce qui facilite la navigation et l'analyse des données. Cette approche garantit des performances optimales pour les requêtes analytiques tout en offrant une structure claire et facile à maintenir.

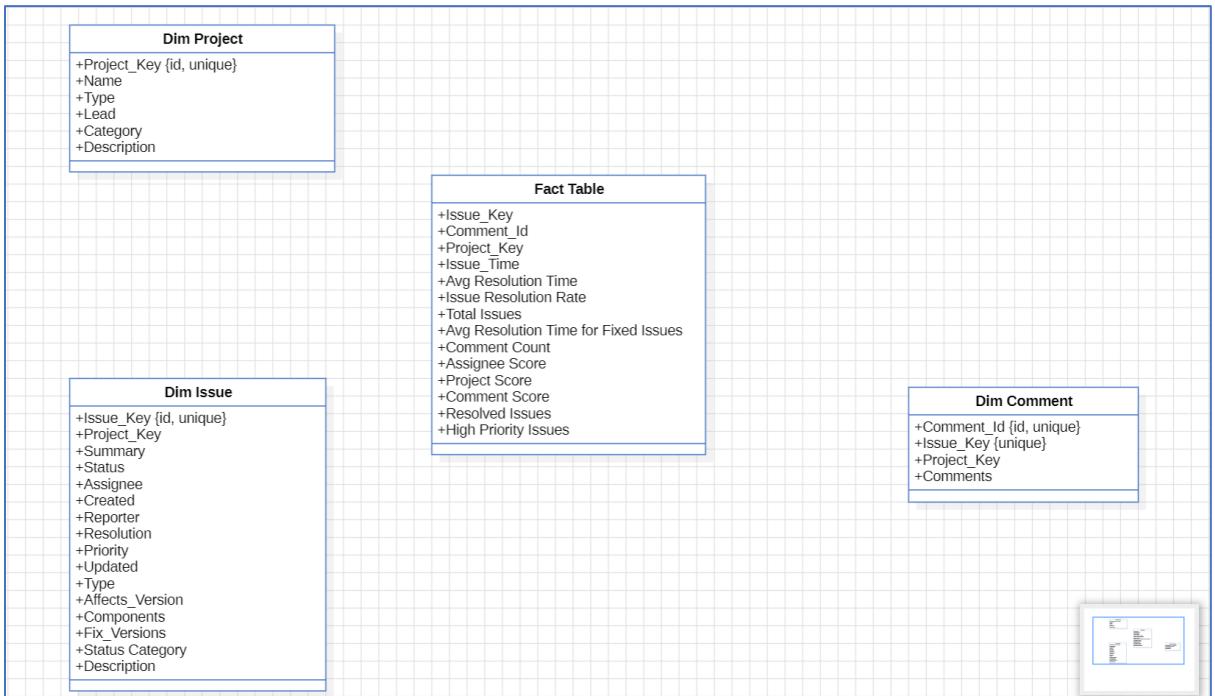


Figure 19 Schéma de la data warehouse

Le schéma en étoile répond parfaitement aux besoins du projet, notamment pour les analyses de performance, les visualisations dans des outils comme Power BI, et l'intégration dans Elasticsearch pour la recherche avancée. Ce choix permet de combiner simplicité, flexibilité, et performance dans le traitement des données.

### 4.3. Relations table de faits et dimensions

Voici notre entrepôt de données avec toutes les relations :

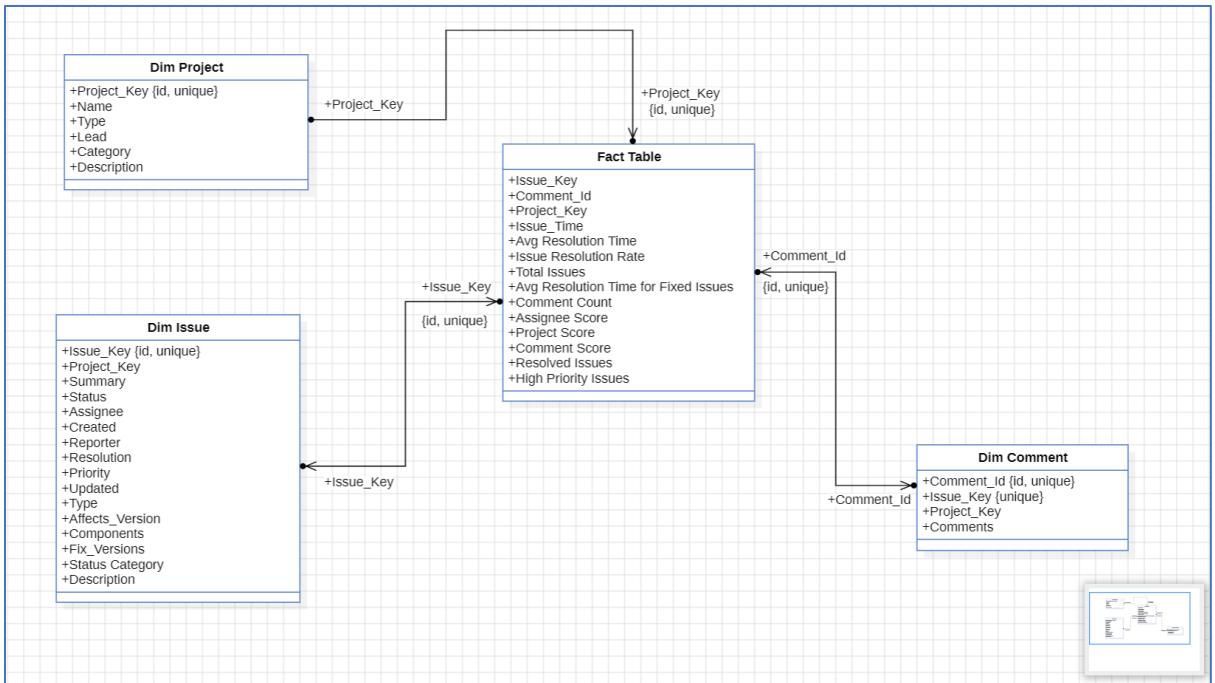


Figure 20 Schéma de la data warehouse complet

## 1. Relation entre la Dimension Issue et la Table de Faits

- **Type :** One-to-One
- **Attribut :** Issue\_Key - Issue\_Key
- **Description :** Pour chaque clé unique d'issue dans la dimension Issues, il existe une clé correspondante dans la table de faits. Cette relation assure un lien direct entre les détails des issues et les métriques associées dans la table de faits, permettant d'analyser chaque issue individuellement.

## 2. Relation entre la Dimension Commentaire et la Table de Faits

- **Type :** One-to-One
- **Attribut :** Comment\_Id - Comment\_Id
- **Description :** De manière similaire à la dimension Issue, chaque clé unique de commentaire dans la dimension Commentaire est directement reliée à une clé correspondante dans la table de faits. Cette relation garantit que chaque série de commentaires peut être tracée et analysée en lien avec les issues et projets associés.

## 3. Relation entre la Dimension Projet et la Table de Faits

- **Type :** One-to-Many (de la dimension Projet vers la table de faits)
- **Attribut :** Project\_Key – Project\_Key
- **Description :** Puisqu'un projet peut contenir plusieurs issues, la table de faits inclut la clé des projets (Project\_Key), qui peut se répéter pour les différentes issues appartenant à un même projet. Cette relation permet d'associer les métriques et les analyses des issues et commentaires à leur contexte projet.

## 5. Conclusion

Ce chapitre a permis de poser les bases solides pour la structuration et l'organisation des données dans le cadre de notre projet. En identifiant les données pertinentes à extraire depuis Jira Service Management, nous avons établi une sélection ciblée couvrant les issues, les commentaires, et les projets, en fonction de leur utilité pour l'analyse et la recherche avancée.

Avec cette conception en place, l'entrepôt de données est prêt à jouer un rôle central dans la suite du projet, en facilitant l'intégration dans Elasticsearch et les visualisations à travers Power BI. Ce travail garantit une base robuste pour les étapes suivantes, axées sur l'analyse et l'exploitation des données.

## Chapitre 4 : Chargement des données

1.	Introduction .....	45
2.	Extraction et transformation des données .....	45
2.1.	ET_Projects .....	45
2.2.	ET_Fields .....	47
2.3.	ET_Comments.....	50
2.4.	Transformation pour ET_Comments .....	55
3.	Chargement des données dans la staging area .....	59
3.1.	Dim Projet .....	60
3.2.	Dim Issue.....	62
3.3.	Dim Comment.....	64
4.	Chargement de la data warehouse .....	65
4.1.	Dimensions.....	65
4.2.	Fact table .....	66
5.	Mise en place de l'environnement Elasticsearch et Kibana .....	68
5.1.	Configuration de elasticsearch.yml .....	68
5.2.	Installation de Kibana.....	71
6.	Connection de la base de données avec Elasticsearch .....	72
6.1.	Création du connecteur et de l'index Elasticsearch.....	72
6.2.	Configuration du connecteur "pgconnector" .....	73
6.3.	Configuration des fichiers config PostgreSQL .....	74
6.4.	Exécution du connecteur "pgconnector" .....	76
7.	Conclusion.....	78

## 1. Introduction

Ce chapitre est consacré au processus ETL (Extraction, Transformation, Chargement) et à la mise en place de notre environnement Elasticsearch, deux étapes essentielles dans la mise en œuvre de notre projet. L'objectif principal est d'assurer une intégration fluide et cohérente des données collectées à partir de différentes sources, en les adaptant au modèle conçu précédemment pour garantir leur exploitabilité.

Nous détaillerons dans un premier temps le processus de chargement des données depuis la staging area vers la data warehouse, en mettant en avant les transformations finales nécessaires pour structurer et optimiser les données. Ensuite, nous aborderons la mise en place d'Elasticsearch, en expliquant comment établir une connexion entre PostgreSQL et Elasticsearch afin de créer un index centralisé, permettant des recherches rapides et performantes sur les documents stockés.

## 2. Extraction et transformation des données

Pour le processus d'extraction et transformation, nous avons choisi d'utiliser Python en combinaison avec Jupyter Notebook et l'API de Jira. Python, grâce à ses bibliothèques riches et sa flexibilité, permet de manipuler les données avec efficacité. Jupyter Notebook offre un environnement interactif idéal pour organiser le code en segments clairs et reproductibles. Enfin, l'API de Jira joue un rôle central dans l'extraction des données, en fournissant un accès programmatique aux informations des projets, issues, commentaires, et pièces jointes stockées dans Jira Service Management.

### Organisation des Notebooks

Ce processus a été structuré en quatre notebooks distincts pour faciliter la gestion des données et maintenir une séparation logique entre les différentes entités :

- **ET\_Projects**
- **ET\_Fields**
- **ET\_Comments**

### 2.1. ET\_Projects

Ce notebook est dédié à l'extraction et la transformation des données des projets Jira, comme leurs noms, types, catégories et responsables.

```
In [1]: import requests
from requests.auth import HTTPBasicAuth
import pandas as pd

projects_url = "https://hibernate.atlassian.net/rest/api/2/project?expand=lead,description"

api_token = "ATATT3XFFGF04d15m_J8u7sB8QL46-gV-_UFgq4bT7LLIFBoesVF7NCq9AincZF6xAnYlRdReBo88Xf-tluj1KgkrV5EZRoa70ns0zTZ6DXxZq1dc4yr"
username = "omar.khabthani123@gmail.com"
```

Figure 21 Code ET\_projects 1

Ce code configure l'accès à l'API Jira pour récupérer les données des projets. Il importe les bibliothèques nécessaires (requests et pandas), définit l'URL pour accéder aux projets Jira, et

configure l'authentification à l'aide d'un nom d'utilisateur et d'un jeton API. Ces éléments permettent de se connecter et d'interagir avec l'API de manière sécurisée.

```
auth = HTTPBasicAuth(username, api_token)

response = requests.get(projects_url, auth=auth)

if response.status_code == 200:
    projects_data = response.json()

    projects = []
    for project_data in projects_data:
        project_name = project_data['name']
        project_key = project_data['key']
        project_type = project_data['projectTypeKey']
        project_lead = project_data['lead']['displayName']
        project_category = project_data['projectCategory'][0]['name']
        project_description = project_data['description']

        projects.append({
            "Project Name": project_name,
            "Project Key": project_key,
            "Project Type": project_type,
            "Project Lead": project_lead,
            "Project Category": project_category,
            "Project Description": project_description
        })

    df = pd.DataFrame(projects)

    excel_filename = "jira_projects.xlsx"
    df.to_excel(excel_filename, index=False)
    print(f"Project information exported to {excel_filename} successfully.")

else:
    print("Failed to fetch project information:", response.text)
```

Project information exported to jira\_projects.xlsx successfully.

Figure 22 Code ET\_Projects 2

Ce code extrait les données des projets Jira à l'aide d'une requête API et les enregistre dans un fichier Excel :

- Une requête est envoyée à l'API Jira pour récupérer les informations des projets.
- Si la requête réussit, les données sont transformées en un tableau contenant des informations clés (nom, clé, type, responsable, catégorie, description).
- Le tableau est ensuite converti en un fichier Excel nommé jira\_projects.xlsx, qui est exporté avec succès.
- Si la requête échoue, un message d'erreur est affiché avec les détails de la réponse.

Ce processus organise efficacement les données des projets pour les analyser ou les intégrer dans d'autres outils.

Grâce à l'ajout de ?expand=lead,description à l'URL de la requête projects\_url, nous avons pu extraire des informations détaillées sur chaque projet, notamment la description et le responsable. La description, en particulier, ne peut être récupérée que sous format JSON via cette requête. Cela permet d'obtenir des données complètes et structurées sur les projets Jira, incluant des éléments spécifiques comme la description textuelle, qui ne seraient autrement pas disponibles dans une requête standard.

Voici le résultat dans une partie du fichier Excel.

A	B	C	D	E	F
Project Key	Name	Type	Lead	Category	Description
2 HCANN	Hibernate Commons Annotations	software	Sanne Grinovero	Active	Utility project for annotation handling
3 ANN	z - Hibernate Annotations	software	Emmanuel Bernard	EOL	Hibernate Java5 annotations for metadata description (EJB3 compliant and Hibernate specific extensions)*NOTE : Now a component of the Hibernate ORM
4 HVAL	z - Hibernate Validator 3	software	Hardy Ferentschik	EOL	
5 STYLE	Hibernate jDocBook Style	software	Steve Ebersole	Active	
6 BVAL	Bean Validation	software	Gunnar Morling	Active	Bean Validation spec
7 HV	Hibernate Validator	software	Guillaume Smet	Active	Annotations based domain model constraint validator
8 HSEARCH	Hibernate Search	software	Marko Bekhta	Active	Hibernate Search: full-text indexing and search through Apache Lucene, Elasticsearch and OpenSearch integration
9 HBX	Hibernate Tools	software	Koen Aeris	Active	Bug area for Hibernate Tools(hibernate plugins for Eclipse, ANT tools, etc...)
10 HHH	Hibernate ORM	software	Steve Ebersole	Active	Project for tracking issues with Hibernate ORM, from version 3 onward.
11 OGM	Hibernate OGM	software	Emmanuel Bernard	Active	Hibernate Object/Grid Mapper
12 HB	z - Hibernate 2	software	GavinG	EOL	Issues pertaining to Hibernate 2.<em><strong>NOTE : No longer supported</strong></em>
13 HREACT	Hibernate Reactive	software	Davide D'Alto	Active	
14 JPA	Java Persistence API	software	Steve Ebersole	Active	Project for issues pertaining to the JPA API itself.
15 BVTC	Bean Validation TCK	software	Gunnar Morling	Active	Issues related to TCK bugs and appeal process.
16 HSHARDS	Hibernate Shards	software	Max Ross	Active	Horizontal partitioning for Hibernate
17 EJB	z - Hibernate Entity Manager	software	Emmanuel Bernard	EOL	Hibernate-based EJB 3.0 Persistence Provider</p><em><strong>NOTE : Now a component of the Hibernate Core project</strong></em>
18 HQLPARSER	HQL/JPQL Parser	software	Sanne Grinovero	Active	Common parse and normalization functionality for translating HQL and JP-QL queries.
19 HBI	z - Hibernate 1.2	software	GavinG	EOL	Issues relating to Hibernate 1.2.<em><strong>NOTE : No longer supported</strong></em>
20 WEBSITE	Hibernate Websites	software	Steve Ebersole	Active	Project for tracking issues regarding the Hibernate websites such as hibernate.org, CI, wiki, blog, etc
21 SQM	Hibernate Semantic Query	software	Steve Ebersole	EOL	Semantic query model and interpreter for HQL/JPQL and JPA Criteria queries.
22 METAGEN	z - Hibernate Metamodel Generator	software	Hardy Ferentschik	EOL	A Java 6 annotation processor used to generate JPA 2 StaticMetamodel classes, intended mostly for use in JPA 2 criteria queries. The processor (JPAMetaMc

Figure 23 Excel des projets

## 2.2. ET\_Fields

Conçu pour traiter les données des issues, ce notebook gère les informations clés telles que les résumés, priorités, statuts, et dates de mise à jour.

```
In [14]: import requests
import json
from bs4 import BeautifulSoup
import pandas as pd
import re

def clean_text(text):
    cleaned_text = re.sub(r'[^\\x00-\\xF]+', '', text)
    cleaned_text = re.sub(r'[\\x00-\\x10][\\x13-\\x14][\\x16-\\x37]', '', cleaned_text)
    return cleaned_text

def fetch_all_project_keys():
    url = "https://hibernate.atlassian.net/rest/api/3/project"
    auth = ("omar.khabthani12@gmail.com", "ATATT3xFgF04dl5m_8u7sB8Ql46-gV_UFgq4bT7LLIFBoesVF7NCq9AincZF6xAnYlRdReBo88xf-tluj1k")
    headers = {"Accept": "application/json"}
    response = requests.get(url, headers=headers, auth=auth)
    response.raise_for_status()
    data = response.json()
    return [project["key"] for project in data]
```

Figure 24 Code ET\_Fields 1

Les bibliothèques requests, json, BeautifulSoup, pandas, et re sont importées pour envoyer des requêtes, traiter les données, et nettoyer les textes.

La fonction clean\_text supprime les caractères non imprimables et les caractères spéciaux qui peuvent perturber l'analyse des données. Cela permet de garantir que les textes extraits des issues sont propres et prêts à être analysés.

La fonction fetch\_all\_project\_keys envoie une requête à l'API Jira pour obtenir toutes les clés des projets. L'authentification se fait via un jeton API et l'adresse e-mail de l'utilisateur.

Les données renvoyées par l'API sont ensuite converties en JSON, et les clés des projets sont extraites et retournées sous forme de liste.

```

def fetch_issue_data(project_key):
    url = "https://hibernate.atlassian.net/rest/api/3/search"
    auth = ("omar.khabthani123@mail.com", "ATATT3xFfGF04d15m_J8u7sB8QL46-gV-_UFgq4bTLLIFBoesVF7NCq9AincZF6xAnY1RdReBo88Xf-tluj1")
    headers = {"Accept": "application/json", "Content-Type": "application/json"}

    all_issues = []
    start_at = 0
    page_size = 50

    while True:
        params = {
            "jql": f"project={project_key}",
            "startAt": start_at,
            "maxResults": page_size,
            "expand": "renderedFields"
        }
        response = requests.get(url, headers=headers, params=params, auth=auth)
        response.raise_for_status()
        data = response.json()

        for issue in data["issues"]:
            description = None
            if "renderedFields" in issue and "description" in issue["renderedFields"]:
                description = BeautifulSoup(issue["renderedFields"]["description"], 'html.parser').get_text()
                description = clean_text(description)

            affects_version = None
            if "versions" in issue["fields"] and issue["fields"]["versions"]:
                affects_version = issue["fields"]["versions"][0]["name"]
            fix_versions = None
            if "fixVersions" in issue["fields"] and issue["fields"]["fixVersions"]:
                fix_versions = [version["name"] for version in issue["fields"]["fixVersions"]]

            all_issues.append({
                "Project": project_key,
                "Key": issue["key"],
                "Summary": issue["fields"]["summary"],
                "Status": issue["fields"]["status"]["name"],
                "Assignee": issue["fields"]["assignee"]["displayName"] if issue["fields"]["assignee"] else None,
                "Created": issue["fields"]["created"],
                "Reporter": issue["fields"]["reporter"]["displayName"] if issue["fields"]["reporter"] else None,
                "Resolution": issue["fields"]["resolution"]["name"] if issue["fields"]["resolution"] else None,
                "Priority": issue["fields"]["priority"]["name"] if issue["fields"]["priority"] else None,
                "Updated": issue["fields"]["updated"],
                "Type": issue["fields"]["issuetype"]["name"],
                "Affects Version": affects_version,
                "Components": [component["name"] for component in issue["fields"]["components"]],
                "Fix Versions": fix_versions,
                "Status Category": issue["fields"]["status"]["statusCategory"]["name"],
                "Description": description,
            })
            start_at += page_size
            if start_at >= data["total"]:
                break

    return all_issues

```

Figure 25 Code ET\_Fields 2

Ce morceau de code est responsable de l'extraction des données détaillées pour chaque issue d'un projet Jira via l'API :

### Récupération des données des issues :

La fonction `fetch_issue_data` prend la clé d'un projet comme argument et récupère les issues associées à ce projet. Elle envoie une requête à l'API de Jira, en utilisant des paramètres comme `startAt` pour paginer les résultats, et `maxResults` pour limiter le nombre d'issues récupérées à chaque appel.

### Traitement des données des issues :

Pour chaque issue, le code extrait divers champs :

- Résumé (Summary), Statut (Status), Assignee, Création (Created), Référenceur (Reporter), Résolution, Priorité, Mise à jour (Updated), Type, Versions affectées, Composants, Versions de correction, Catégorie de statut, et Description.

La description est spécialement nettoyée et transformée en texte brut à l'aide de BeautifulSoup, en supprimant les balises HTML, et en appliquant la fonction de nettoyage de texte clean\_text pour supprimer les caractères indésirables.

### Pagination des résultats :

Le code continue à récupérer des issues par lots de 50, et s'arrête lorsque toutes les issues ont été récupérées (lorsque start\_at atteint ou dépasse le nombre total d'issues).

### Création de la liste d'issues :

Les données récupérées pour chaque issue sont ajoutées à une liste sous forme de dictionnaire, avec des clés correspondant aux champs extraits. Ces données peuvent ensuite être converties en un DataFrame Pandas ou exportées vers un fichier Excel pour une utilisation ultérieure.

Ce code permet donc d'extraire efficacement une grande quantité d'informations détaillées sur les issues Jira, en prenant soin de nettoyer et formater les données pour une analyse plus approfondie.

```
if __name__ == "__main__":
    username = "omar.khabthani123@gmail.com"
    api_token = "ATATT3XFfGF04dl5m_J8u7sB8QL46-gV-_UFgq4bT7LLIFBoesVF7NCq9AincZF6xAnYlRdReBo88xf-tluj1KgkrV5EZRoa70ns0ZTZ6DXXzq1c

    all_project_keys = fetch_all_project_keys()
    all_issues = []

    for project_key in all_project_keys:
        project_issues = fetch_issue_data(project_key)
        all_issues.extend(project_issues)

    df = pd.DataFrame(all_issues)
    excel_filename = "jira_fields1.xlsx"
    df.to_excel(excel_filename, index=False)
    print(f"DataFrame exported to {excel_filename} successfully.")

DataFrame exported to jira_fields1.xlsx successfully.
```

Figure 26 Code ET\_Fields 3

Dans cette section finale, le code organise l'extraction et l'exportation des données dans un fichier Excel :

### Initialisation des variables :

Le code commence par définir les informations d'authentification (nom d'utilisateur et jeton d'API pour Jira) nécessaires pour l'accès aux API.

### Récupération des clés de projets :

La fonction fetch\_all\_project\_keys est appelée pour récupérer toutes les clés des projets Jira disponibles. Ces clés sont ensuite stockées dans la liste all\_project\_keys.

### Extraction des issues pour chaque projet :

Pour chaque clé de projet, la fonction fetch\_issue\_data est appelée pour récupérer les données des issues associées à ce projet. Les données extraites sont ajoutées à la liste all\_issues à l'aide de la méthode extend, qui permet d'ajouter les issues de chaque projet à la liste principale.

### Création et exportation du DataFrame :

Une fois que toutes les données des issues ont été collectées, elles sont converties en un

DataFrame Pandas. Ce DataFrame est ensuite exporté sous forme de fichier Excel (jira\_fields1.xlsx), et un message de succès est affiché.

Voici le résultat dans une partie du fichier Excel.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Issue Key	Project Key	Summary	Status	Assignee	Created	Reporter	Resolution	Priority	Updated	Type	Affects Version	Components	Fix Versions	Status Category	Description
HCANN-134	HCANN	Switch project license Open	Sanne Grinovero	2024-03-14' Sanne Grinovero			Major	2024-03-15' Epic			["7.0"]	To Do	See https://ir		
HCANN-133	HCANN	Remove optimisation Open	Sanne Grinovero	2024-03-14' Sanne Grinovero			Minor	2024-03-14' Task				To Do			
HCANN-132	HCANN	Annotating an entity' Open		2023-09-21' Hamza Hathout			Minor	2023-09-22' Bug	6.0.6.Final			To Do	In Java 17, wl		
HCANN-131	HCANN	JavaXPropertyTest w' Resolved	Sanne Grinovero	2023-08-15' Sanne Grinovero			Fixed	2023-08-15' Task			["6.0.0.next"]	Done	This is a follo		
HCANN-129	HCANN	After Spring boot up' Closed	Sanne Grinovero	2023-03-09' Sharath Jayarakash			Rejected	2023-06-28' Bug				Done	H1 Team, we i		
HCANN-128	HCANN	Reflection does not u' Closed		2023-03-01' Marcus Igner			Cannot Reproduce	2023-03-07' Bug	5.1.2			Done	In a setup wit		
HCANN-127	HCANN	Approximation of a Closed	Christian Beikov	2023-01-26' Christian Beikov			Fixed	2023-01-26' Bug			["6.0.6.Final"]	Done	While debugg		
HCANN-126	HCANN	For pageable query w' Closed	Sanne Grinovero	2022-12-12' Jakub Radlka			Rejected	2022-12-12' Bug	6.0.5.Final			Done	Duplicate of I		
HCANN-125	HCANN	The type of a multipli' Closed	Andrea Boriero	2022-11-15' Andrea Boriero			Rejected	2022-11-17' Bug			["6.0.5.Final"]	Done	Given public c		
HCANN-124	HCANN	Rename access type' Resolved	Christian Beikov	2022-08-29' Christian Beikov			Fixed	2022-08-29' Improvement	6.0.3.Final			["6.0.4.Final"]	Done	After discuss	
HCANN-123	HCANN	Support accessing rec' Resolved	Christian Beikov	2022-08-11' Christian Beikov			Fixed	2022-08-15' Improvement			["6.0.3.Final"]	Done	In order to pr		
HCANN-122	HCANN	Make java.lang.reflec' Closed	Christian Beikov	2022-05-18' Christian Beikov			Fixed	2022-08-15' Improvement			["6.0.2.Final"]	Done			
HCANN-121	HCANN	Introduce toType() m' Closed	Christian Beikov	2022-04-11' Christian Beikov			Fixed	2022-05-03' Improvement			["6.0.1.Final"]	Done	We currently		
HCANN-120	HCANN	Introduce getContain' Closed	Gavin King	2022-01-17' Gavin King			Fixed	2022-01-19' New Feature			["6.0.0.CR1"]	Done	as suggested		
HCANN-119	HCANN	Introduce a real mod' Closed	Sanne Grinovero	2021-12-22' Sanne Grinovero			Fixed	2021-12-22' Task			["6.0.0.Beta1"]	Done			
HCANN-117	HCANN	Require JDK 11' Closed	Sanne Grinovero	2021-12-22' Sanne Grinovero			Fixed	2021-12-22' Task			["6.0.0.Beta1"]	Done	We're no lon		
HCANN-116	HCANN	Remove dependency' Closed	Sanne Grinovero	2021-12-22' Sanne Grinovero			Fixed	2021-12-22' Task			["6.0.0.Beta1"]	Done	The Logger di		
HCANN-115	HCANN	Upgrade to Gradle v' Closed	Sanne Grinovero	2021-12-22' Sanne Grinovero			Fixed	2021-12-22' Task			["6.0.0.Beta1"]	Done			
HCANN-114	HCANN	Add copyright header' Closed	Sanne Grinovero	2020-11-18' Sanne Grinovero			Fixed	Minor	2021-12-22' Task	5.1.1		["6.0.0.Beta1"]	'Done	Files 'TypeEn	
HCANN-113	HCANN	Reduce retained mem' Closed	Sanne Grinovero	2020-10-30' Sanne Grinovero			Fixed	Major	2020-10-30' Task			["6.0.0.Beta1"]	'Done		
HCANN-112	HCANN	Favour releasing ref' Closed	Sanne Grinovero	2020-10-30' Sanne Grinovero			Fixed	Major	2020-10-30' Task			["6.0.0.Beta1"]	'Done	This project f	
HCANN-111	HCANN	Improve hashing and' Closed	Sanne Grinovero	2020-10-30' Sanne Grinovero			Fixed	Major	2020-10-30' Task			["6.0.0.Beta1"]	'Done	The JavaRefl	
HCANN-110	HCANN	Improve encapsulat' Closed	Sanne Grinovero	2020-10-23' Sanne Grinovero			Fixed	Trivial	2021-12-22' Task			["6.0.0.Beta1"]	'Done		
HCANN-109	HCANN	TypeEnvironmentFac' Closed	Sanne Grinovero	2020-10-23' Sanne Grinovero			Fixed	Trivial	2021-12-22' Task			["6.0.0.Beta1"]	'Done		
HCANN-108	HCANN	Never use a LinkedLi' Closed	Sanne Grinovero	2020-10-23' Sanne Grinovero			Fixed	Trivial	2020-10-25' Task			["6.0.0.Beta1"]	'Done	It's rarely effi	
HCANN-107	HCANN	Introduce a method' Closed	Sanne Grinovero	2020-10-23' Sanne Grinovero			Fixed	Major	2020-10-25' Improvement			["6.0.0.Beta1"]	'Done	It turns out th	
HCANN-106	HCANN	Remove unused help' Closed	Sanne Grinovero	2020-10-22' Sanne Grinovero			Fixed	Major	2021-12-22' Remove Feature			["6.0.0.Beta1"]	'Done		
HCANN-105	HCANN	Remove capability' Closed	Sanne Grinovero	2020-10-22' Sanne Grinovero			Fixed	Trivial	2021-12-22' Remove Feature			["6.0.0.Beta1"]	'Done		
HCANN-104	HCANN	Deprecate capability' Closed	Sanne Grinovero	2020-10-22' Sanne Grinovero			Fixed	Trivial	2020-10-25' Deprecation			["5.1.1"]	'Done	I believe all c	

Figure 27 Excel des issues

## 2.3. ET\_Comments

Ce notebook prend en charge les commentaires associés aux issues, en les nettoyant et en les formatant pour leur intégration dans le modèle de données.

Pour l'extraction des commentaires, nous avons initialement tenté de les extraire directement dans Jupyter Notebook. Cependant, en raison du grand nombre de commentaires à traiter et des ressources limitées en termes de puissance de traitement, nous avons opté pour l'utilisation de Google Colab. Ce dernier offre plus de puissance de calcul et de mémoire, ce qui est nécessaire pour gérer efficacement de grandes quantités de données et accélérer le processus d'extraction des commentaires, en particulier dans un environnement de traitement intensif comme celui-ci.

```
import requests
from requests.auth import HTTPBasicAuth
import json
from bs4 import BeautifulSoup
import pandas as pd

def fetch_all_project_keys():
    url = "https://hibernate.atlassian.net/rest/api/3/project"
    auth = HTTPBasicAuth("omar.khabthani123@gmail.com", "ATATT3xFfGF04d15m_J8u7sB8QL46-gV-_UFgq4bT7L")
    headers = {"Accept": "application/json"}
    response = requests.get(url, headers=headers, auth=auth)
    response.raise_for_status()
    data = response.json()
    return [{"project_key": project["key"], "project_name": project["name"]} for project in data]
```

Figure 28 Code ET\_Comments 1

Cette première partie du code permet d'extraire les clés et les noms de tous les projets disponibles dans Jira à l'aide de l'API. Voici un résumé de son fonctionnement :

## Importation des bibliothèques nécessaires :

Les modules requests, json, BeautifulSoup, et pandas sont importés pour gérer les requêtes API, analyser les données et les structurer en DataFrame.

### Fonction fetch\_all\_project\_keys():

- La fonction se connecte à l'API Jira pour récupérer la liste des projets.
- URL de l'API : Utilise l'URL <https://hibernate.atlassian.net/rest/api/3/project> pour accéder aux informations des projets.
- Authentification : L'authentification est effectuée via HTTPBasicAuth en utilisant un email et un jeton d'API.
- Requête GET : Envoie une requête avec l'en-tête approprié pour recevoir les données au format JSON.
- Traitement des données : Les données récupérées sont filtrées pour extraire uniquement la clé du projet (project\_key) et son nom (project\_name).

### Retour des données :

La fonction renvoie une liste de dictionnaires contenant les clés et noms des projets, ce qui servira de base pour l'extraction des commentaires liés à chaque projet.

```
def fetch_issue_keys(project_key):  
    url = "https://hibernate.atlassian.net/rest/api/3/search"  
    auth = HTTPBasicAuth("omar.khabthani123@gmail.com", "ATATT3xFfGF04dl5m_J8u7sB8QL46-gV-_UFgq4b  
    headers = {"Accept": "application/json"}  
    all_issue_keys = []  
  
    start_at = 0  
    max_results = 50  
    total = max_results  
  
    while start_at < total:  
        params = {  
            "jql": f"project={project_key}",  
            "fields": "key",  
            "startAt": start_at,  
            "maxResults": max_results  
        }  
        response = requests.get(url, headers=headers, params=params, auth=auth)  
        response.raise_for_status()  
        data = response.json()  
  
        for issue in data.get("issues", []):  
            all_issue_keys.append({"project_key": project_key, "issue_key": issue["key"]})  
  
        start_at += max_results  
        total = data.get("total", 0)  
  
    return all_issue_keys
```

Figure 29 Code ET\_Comments 2

Cette fonction, fetch\_issue\_keys, est utilisée pour extraire les clés de toutes les issues (problèmes ou tickets) associées à un projet donné dans Jira.

### **Initialisation des paramètres :**

- Une liste all\_issue\_keys est créée pour stocker les clés des issues.
- Des variables de pagination, start\_at et max\_results, permettent de gérer les réponses paginées de l'API.

### **Boucle de pagination :**

- Tant que start\_at est inférieur au nombre total d'issues (total), une requête est envoyée avec les paramètres adéquats :
  - jql : filtre pour limiter les résultats au projet donné.
  - fields : demande uniquement les clés des issues.
  - startAt et maxResults : gèrent les limites de pagination.
- La réponse est analysée pour récupérer les clés des issues. Chaque clé est ajoutée à la liste all\_issue\_keys avec la clé du projet correspondant.

### **Mise à jour de la pagination :**

- start\_at est incrémenté pour passer à la page suivante.
- La variable totale est mise à jour avec le nombre total d'issues retournées par l'API, permettant de contrôler la boucle.

### **Retour des données :**

La fonction retourne une liste de dictionnaires contenant les clés des projets (project\_key) et les clés des issues (issue\_key).

```

def fetch_comments(issue_keys, url):
    auth = HTTPBasicAuth("omar.khabthani123@gmail.com", "ATATT3xXfGF04d15mJ8u7sB8QL46-gV-_UFgq4bT7LLIFBoesVF7NC")
    headers = {"Accept": "application/json"}
    all_comments = []

    for issue_key in issue_keys:
        response = requests.get(url.format(issueIdOrKey=issue_key["issue_key"]), headers=headers, auth=auth)
        response.raise_for_status()
        data = response.json()

        comments = data.get("comments")
        if comments:
            for comment in comments:
                rendered_body = BeautifulSoup(comment.get("renderedBody", ""), 'html.parser').get_text()
                all_comments.append({
                    "project_key": issue_key["project_key"],
                    "issue_key": issue_key["issue_key"],
                    "comment_id": comment["id"],
                    "rendered_body": rendered_body
                })
        else:
            all_comments.append([
                "project_key": issue_key["project_key"],
                "issue_key": issue_key["issue_key"],
                "comment_id": None,
                "rendered_body": None
            ])

    return all_comments

```

Figure 30 Code ET\_Comments 3

Cette fonction, `fetch_comments`, est utilisée pour extraire les commentaires associés à une liste d'issues (tickets) provenant de Jira. Voici une synthèse :

#### **Initialisation :**

Une liste `all_comments` est créée pour stocker tous les commentaires des issues.

#### **Parcours des issues :**

Pour chaque clé d'issue fournie dans `issue_keys`, une requête est envoyée à l'API de Jira pour récupérer les commentaires.

#### **Traitement des commentaires :**

- Les commentaires sont récupérés à partir du champ `comments` dans la réponse JSON.
- Si des commentaires sont présents :
  - Le corps de chaque commentaire est analysé via `BeautifulSoup` pour extraire le texte brut de `renderedBody`.
  - Les données sont stockées sous forme de dictionnaire avec les champs :
    - `project_key` : clé du projet.
    - `issue_key` : clé de l'issue.
    - `comment_id` : identifiant du commentaire.
    - `rendered_body` : texte brut du commentaire.

- Si aucun commentaire n'est présent, des valeurs None sont ajoutées pour comment\_id et rendered\_body.

### Retour des données :

Une fois tous les commentaires récupérés et traités, la fonction retourne la liste all\_comments, contenant les données organisées de chaque commentaire.

```
if __name__ == "__main__":
    url = "https://hibernate.atlassian.net/rest/api/3/issue/{issueIdOrKey}/comment?expand=renderedBody"

    all_project_keys = fetch_all_project_keys()
    all_comments = []

    for project in all_project_keys:
        issue_keys = fetch_issue_keys(project["project_key"])
        project_comments = fetch_comments(issue_keys, url)
        all_comments.extend(project_comments)

    df = pd.DataFrame(all_comments)
    excel_filename = "jira_comments2.xlsx"
    df.to_excel(excel_filename, index=False)
    print(f"DataFrame exported to {excel_filename} successfully.")

    ➜ DataFrame exported to jira_comments2.xlsx successfully.
```

Figure 31 Code ET\_Comments 4

Ce code réalise un processus complet pour extraire les commentaires associés à chaque issue de différents projets Jira, en utilisant l'API REST de Jira avec l'extension ?expand=renderedBody.

### Récupération des projets :

La fonction fetch\_all\_project\_keys récupère les clés et noms de tous les projets disponibles dans Jira via l'API REST. Ces informations servent de point de départ pour parcourir les issues liées à chaque projet.

### Récupération des issues :

La fonction fetch\_issue\_keys prend en entrée la clé d'un projet et retourne toutes les clés des issues associées à ce projet. Le code utilise la pagination pour garantir la récupération de toutes les issues, même lorsqu'elles sont nombreuses.

### Récupération des commentaires :

La fonction fetch\_comments utilise les clés des issues pour interroger l'API Jira et extraire les commentaires associés. Avec l'extension ?expand=renderedBody, elle récupère le champ renderedBody pour chaque commentaire, qui contient le texte enrichi en format JSON. Ce champ est ensuite transformé en texte brut à l'aide de BeautifulSoup pour une exploitation plus pratique.

### Agrégation des données :

Toutes les informations sont collectées et organisées dans une structure tabulaire (DataFrame Pandas), avec des colonnes telles que project\_key, issue\_key, comment\_id, et rendered\_body.

## Exportation des résultats :

Les données finales sont exportées dans un fichier Excel (jira\_comments2.xlsx) pour une utilisation ultérieure.

Voici le résultat dans une partie du fichier Excel.

A	B	C	D
project_key	issue_key	comment_id	rendered_body
1 HCANN	HCANN-134		
2 HCANN	HCANN-133		
3 HCANN	HCANN-132		
4 HCANN	HCANN-131		
5 HCANN	HCANN-129	111044	hi Sharath, sorry this is not the right project - I'll have to close it.If you can create a reproducer which doesn't use Spring please open an issue on the Hibernate ORM project; otherwise it might be best
6 HCANN	HCANN-128	111000	Hi Marcus, at that point the Class reference is already resolved so I'm not understanding the problem. Would you be able to share more details, such as the error message? A reproducer which we can
7 HCANN	HCANN-128	111010	I'm not sure I understand what the issue might be, unfortunately. In the meantime I have managed to work around this by loading the class through a different classloader. So it's not acute for me and i
8 HCANN	HCANN-128	111023	Right that sounds like the contextual classloader wasn't setup correctly, which would indeed be fixed by setting it upfront. I'll close this - if you still have problems feel free to re-open, add a comment c
9 HCANN	HCANN-127	110757	Fixed via <a href="https://github.com/hibernate/hibernate-commons-annotations/commit/ace6228b9b0ddd74604776380c5e7e01b0bfab46">https://github.com/hibernate/hibernate-commons-annotations/commit/ace6228b9b0ddd74604776380c5e7e01b0bfab46</a>
10 HCANN	HCANN-127	110757	Wrong project, see HHH-15852 Closed
11 HCANN	HCANN-126	110444	
12 HCANN	HCANN-125		
13 HCANN	HCANN-124		
14 HCANN	HCANN-123		
15 HCANN	HCANN-122		
16 HCANN	HCANN-121		
17 HCANN	HCANN-120		
18 HCANN	HCANN-118		
19 HCANN	HCANN-117		
20 HCANN	HCANN-116		
21 HCANN	HCANN-115		
22 HCANN	HCANN-114		
23 HCANN	HCANN-113		
24 HCANN	HCANN-112		
25 HCANN	HCANN-111		
26 HCANN	HCANN-110		
27 HCANN	HCANN-109		
28 HCANN	HCANN-108		
29 HCANN	HCANN-107		
30 HCANN	HCANN-106		

Figure 32 Excel des commentaires

## 2.4. Transformation pour ET\_Comments

Nous avons décidé d'aller plus loin dans la collecte des données des commentaires en incluant le nom de chaque personne ayant écrit un commentaire. Ces informations seront ensuite fusionnées avec le contenu des commentaires (rendered\_body) pour former un texte unique contenant une série de commentaires par issue.

### Raisons de cette décision :

- Réduction de la granularité des données :**  
La structure actuelle du Data Warehouse ne peut pas supporter un volume élevé de commentaires individuels pour chaque issue en raison des limitations en termes de stockage ou de performance.
- Conservation du contexte des échanges :**  
En combinant les noms des auteurs et les messages, nous maintenons le contexte conversationnel des commentaires tout en simplifiant leur stockage.
- Optimisation pour les analyses futures :**  
Cette fusion facilite les analyses textuelles sur les discussions complètes par issue, plutôt que sur des fragments individuels.

Pour collecter les noms des auteurs de chaque commentaire, nous avons modifié la fonction fetch\_comments afin de récupérer et inclure le champ creator\_display\_name (nom de l'auteur).

```

def fetch_comments(issue_keys, url):
    auth = HTTPBasicAuth("omar.khabthani123@gmail.com", "ATATT3xFfGF04dl5m_J8u7sB8QL46-gV-_UFgq4bT7LLlFBoesVF7NCqS")
    headers = {"Accept": "application/json"}
    all_comments = []

    for issue_key in issue_keys:
        response = requests.get(url.format(issueIdOrKey=issue_key["issue_key"]), headers=headers, auth=auth)
        response.raise_for_status()
        data = response.json()

        comments = data.get("comments")
        if comments:
            for comment in comments:
                creator_display_name = comment["author"]["displayName"]
                all_comments.append({
                    "project_key": issue_key["project_key"],
                    "issue_key": issue_key["issue_key"],
                    "comment_id": comment["id"],
                    "creator_display_name": creator_display_name
                })
        else:
            all_comments.append({
                "project_key": issue_key["project_key"],
                "issue_key": issue_key["issue_key"],
                "comment_id": None,
                "creator_display_name": None
            })

    return all_comments

```

Figure 33 Code Comments ID

#### **Extraction du nom de l'auteur :**

Nous avons ajouté la récupération du champ comment["author"]["displayName"], qui contient le nom de la personne ayant écrit le commentaire.

Ce champ est ensuite stocké sous le nom creator\_display\_name.

#### **Structure des données retournées :**

La sortie de la fonction inclut désormais creator\_display\_name avec les autres informations du commentaire.

#### **Gestion des commentaires absents :**

En cas d'absence de commentaires pour une issue, la fonction retourne None pour les champs comment\_id et creator\_display\_name.

Voici le résultat dans une partie du fichier Excel.

	A	B	C	D
1	project_key	issue_key	comment_id	creator_display_name
2	HCANN	HCANN-134		
3	HCANN	HCANN-133		
4	HCANN	HCANN-132		
5	HCANN	HCANN-131		
6	HCANN	HCANN-129	111044	Sanne Grinovero
7	HCANN	HCANN-128	111000	Sanne Grinovero
8	HCANN	HCANN-128	111010	Marcus Ilgner
9	HCANN	HCANN-128	111023	Sanne Grinovero
10	HCANN	HCANN-127	110757	Christian Beikov
11	HCANN	HCANN-126	110444	Sanne Grinovero
12	HCANN	HCANN-125		
13	HCANN	HCANN-124		
14	HCANN	HCANN-123		
15	HCANN	HCANN-122		
16	HCANN	HCANN-121		
17	HCANN	HCANN-120		
18	HCANN	HCANN-118		
19	HCANN	HCANN-117		
20	HCANN	HCANN-116		
21	HCANN	HCANN-115		
22	HCANN	HCANN-114		
23	HCANN	HCANN-113		
24	HCANN	HCANN-112		
25	HCANN	HCANN-111		
26	HCANN	HCANN-110		
27	HCANN	HCANN-109		
28	HCANN	HCANN-108		
29	HCANN	HCANN-107		
30	HCANN	HCANN-106		

Figure 34 Excel Comment ID

Afin de regrouper les informations issues des deux fichiers Excel, jira\_comments\_name.xlsx (contenant les noms des auteurs) et jira\_comments2.xlsx (contenant les commentaires), nous avons utilisé le code suivant pour les fusionner :

```
import pandas as pd

df_display_name = pd.read_excel("jira_comments_name.xlsx")

df_comment_body = pd.read_excel("jira_comments2.xlsx")

merged_df = pd.merge(df_display_name, df_comment_body, on=['project_key', 'issue_key', 'comment_id'])

grouped_df = merged_df.groupby(['project_key', 'issue_key']).apply(lambda x: '\n'.join(f"- {row['creator_display_name']}:{row['rendered_body']}" if not pd.isna(row['rendered_body']) else f"none" for idx, row in x.iterrows()))

flattened_df = pd.DataFrame(grouped_df, columns=['comment'])

flattened_df.reset_index(inplace=True)

flattened_df.to_excel("merged_comments.xlsx", index=False)
```

Figure 35 Code de Transformation des commentaires

Ce code a pour objectif de fusionner deux fichiers Excel, jira\_comments\_name.xlsx et jira\_comments2.xlsx, contenant respectivement les noms des auteurs des commentaires et les corps des commentaires associés. Il regroupe ensuite ces informations dans un seul texte par problème (*issue*) et les exporte dans un fichier Excel final, merged\_comments.xlsx.

### **Chargement des fichiers Excel :**

Les fichiers jira\_comments\_name.xlsx et jira\_comments2.xlsx sont chargés dans des DataFrames Pandas nommés respectivement df\_display\_name et df\_comment\_body.

### **Fusion des fichiers :**

- Les deux DataFrames sont fusionnés à l'aide de la méthode pd.merge() sur les colonnes communes : project\_key, issue\_key, et comment\_id.
- Cela garantit que chaque commentaire est enrichi des informations sur son auteur et son projet.

### **Regroupement des commentaires :**

- Le DataFrame fusionné, merged\_df, est regroupé par project\_key et issue\_key pour regrouper tous les commentaires associés à une même *issue*.
- La méthode .apply() est utilisée pour parcourir chaque groupe et concaténer les noms des auteurs avec leurs commentaires respectifs dans un format clair et lisible
- Si le champ rendered\_body est vide ou inexistant, le texte "none" est utilisé.

### **Conversion en DataFrame aplatie :**

Le résultat du regroupement est transformé en un nouveau DataFrame nommé flattened\_df, où chaque ligne correspond à une *issue* avec tous ses commentaires consolidés dans une seule cellule.

### **Réinitialisation des index :**

Les indices du regroupement sont réinitialisés pour convertir project\_key et issue\_key en colonnes normales.

### **Exportation vers Excel :**

Le DataFrame final est exporté dans un fichier Excel nommé merged\_comments.xlsx, prêt à être intégré dans une base de données ou utilisé pour d'autres analyses.

Voici le résultat final :

A	B	C
Project Key	Issue Key	Comments
1 ANN	ANN-1	- Emmanuel Bernard:Done partially. See HBX-302 Closed- Emmanuel Bernard:Delay this won't have time for end of month- Emmanuel Bernard:Done through @or.hibernate.annotations.MapKeyMap<Element,...>Ma
2 ANN	ANN-100	- Dave Brondsema:Actually you could probably keep all the method signatures the same but if it is a ParameterizedType then recursively call extractType until you get a Class-. Emmanuel Bernard:Where you able to map SupportLi
3 ANN	ANN-101	- Emmanuel Bernard:Year, it's actually impossible to use @MapKey in a many to many in Hibernate Core- Christian Bauer:I've got the same mapping working in XML with a formula: <map name="items" table="CATEGORY_ITEM"
4 ANN	ANN-102	- Emmanuel Bernard:it does already, the user was crazy
5 ANN	ANN-103	- Amit Pashazadeh:But I believe the optional="false" is a necessity too. Cause with optional="false" and @SecondaryTables and SingleTable inheritance strategy we could have joined subclasses with discriminator values.- Feybess
6 ANN	ANN-104	- Emmanuel Bernard:The rest is already supported since 3.2.0- Emmanuel Bernard:done, note that R is actually not supported by hibernate core
7 ANN	ANN-105	- Max Rydahl Andersen:done after emmanuel's blessing
8 ANN	ANN-106	- GavinG:"Critical"?I Sheesh Read the EJB spec and the documentation - mike perham:Would it have been that tough to just type one word "@EmbeddableSuperclass" instead of the sarcasm and the generic advice?- GavinG:Yes,
9 ANN	ANN-108	- Emmanuel Bernard:Tested, it works.- MichaelM:The problem is the SchemaUpdate (I use <property name="hibernate.hbm2ddl.auto">update</property>).See HB-1458 Open for details.- Emmanuel Bernard:The fate
10 ANN	ANN-109	none
11 ANN	ANN-111	none
12 ANN	ANN-110	- Emmanuel Bernard:HA raise an exception and I'm -1 on checking the actual signature.- Dennis C. Byrne:I take it this is in HEAD? betas5 let's me navigate the relationship as well.
13 ANN	ANN-111	- Gregor Melhorn:that was too fast. Made a mistake on the first function calling the second one, anyway it should be easily fixed...
14 ANN	ANN-113	- Emmanuel Bernard:MapKey is not supposed to support entity as index, see ANN-1 Resolved for a full support of maps
15 ANN	ANN-114	none
16 ANN	ANN-115	- Emmanuel Bernard:The user is probably wrong on the failing version- THE DPR:I've got similar problem with @JoinColumn and @Index in beta6.@ManyToOne@JoinColumn(name = "parent_id") @Index(name = "category_parent",
17 ANN	ANN-116	none
18 ANN	ANN-117	none
19 ANN	ANN-118	none
20 ANN	ANN-119	- Emmanuel Bernard:So changing the default message to {default.notNull} instead of "this property should not be null"- Emmanuel Bernard:... might be a good idea with a default.message.properties embedded in the JAR- Emmar
21 ANN	ANN-120	- Daniel Fielder:Have you figured anything out yet on this issue? From examining the SQL produced, you can see that it's looking for the mapping column on the join table instead of the table that is connected to the query via the
22 ANN	ANN-121	none
23 ANN	ANN-122	- Max Rydahl Andersen:remember to handle multi property naturalids..- Emmanuel Bernard:"remember to handle multi property naturalids.."The exact reason why it's not supported right now- MathiasM:How about doing some
24 ANN	ANN-123	- Emmanuel Bernard:Partial support: only 1 level of hierarchy, and only superclasses need to do several level and interfaces- Emmanuel Bernard:Multilevel is now supported, interfaces do not make sense.
25 ANN	ANN-124	- Emmanuel Bernard:I had static imports in mind. But I'll change that if it helps the doc readability- Andrew C. Oliver:Yeah probably so cause while an IDE will help find AccessType or @Entity, it won't help find FIELD.- GavinG:Stati
26 ANN	ANN-125	- Emmanuel Bernard:Done through getPropertyName(), and getRootBean()
27 ANN	ANN-126	none
28 ANN	ANN-127	- Emmanuel Bernard:Nice idea. Provide a patch, I'll apply it.- Emmanuel Bernard:Done, I'm concerned by the day after etc, cause it imply something much more complex like day comparison, month comparison, year comparison
29 ANN	ANN-128	- Emmanuel Bernard:Still need to push path instead of propertyName- Emmanuel Bernard:Done.

Figure 36 Excel final des commentaires

### 3. Chargement des données dans la staging area

Maintenant, nous allons passer au chargement des données dans la staging area. Pour cela, nous avons utilisé des outils tels que Talend et PostgreSQL. Talend a permis d'extraire les données depuis les fichiers Excel et de les transférer de manière structurée dans une base de données PostgreSQL. Cette étape a facilité le nettoyage et la transformation des données avant leur intégration finale dans le data warehouse. PostgreSQL a été choisi pour sa capacité à gérer efficacement les volumes de données tout en garantissant leur intégrité et leur préparation pour les étapes suivantes.

La staging area, ou zone de préparation, est un espace intermédiaire crucial dans le processus d'intégration des données. C'est là que les données brutes sont collectées, nettoyées et transformées avant d'être chargées dans le data warehouse. Cette étape est essentielle car elle garantit la qualité et la cohérence des données qui seront utilisées pour l'analyse et la prise de décision. Dans ce rapport, nous allons explorer comment nous allons démarrer la mise en place de la staging area pour notre projet BI."

Tout d'abord, pour mettre en place notre staging area, nous avons créé la base de données Staging Area dans PostgreSQL. Ensuite, nous avons établi la connexion entre cette base de données et Talend, notre outil d'intégration de données.

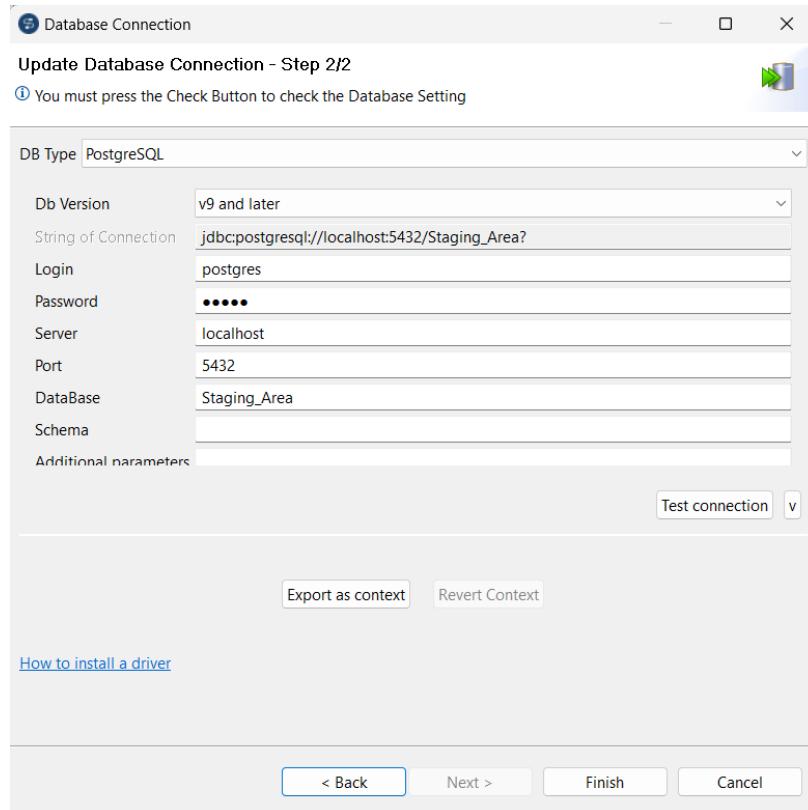


Figure 37 Création de la DB SA

### 3.1. Dim Projet

Le job "dim\_projects" comprend les composantes nécessaires pour charger la table "Projects". Ces composantes sont :

- Le fichier Excel contenant les données des projets.
- La composante TMap pour effectuer les transformations nécessaires sur les données.
- La composante DB Output, qui servira à créer la table dans la base de données et à y charger les données.

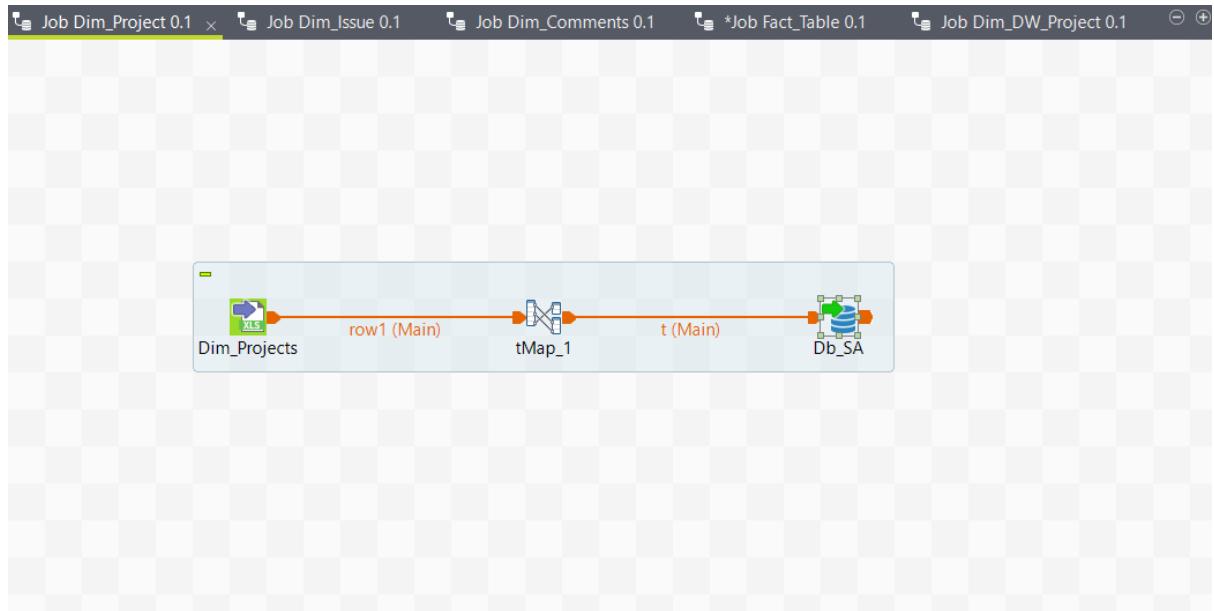


Figure 38 Job Dim\_Project

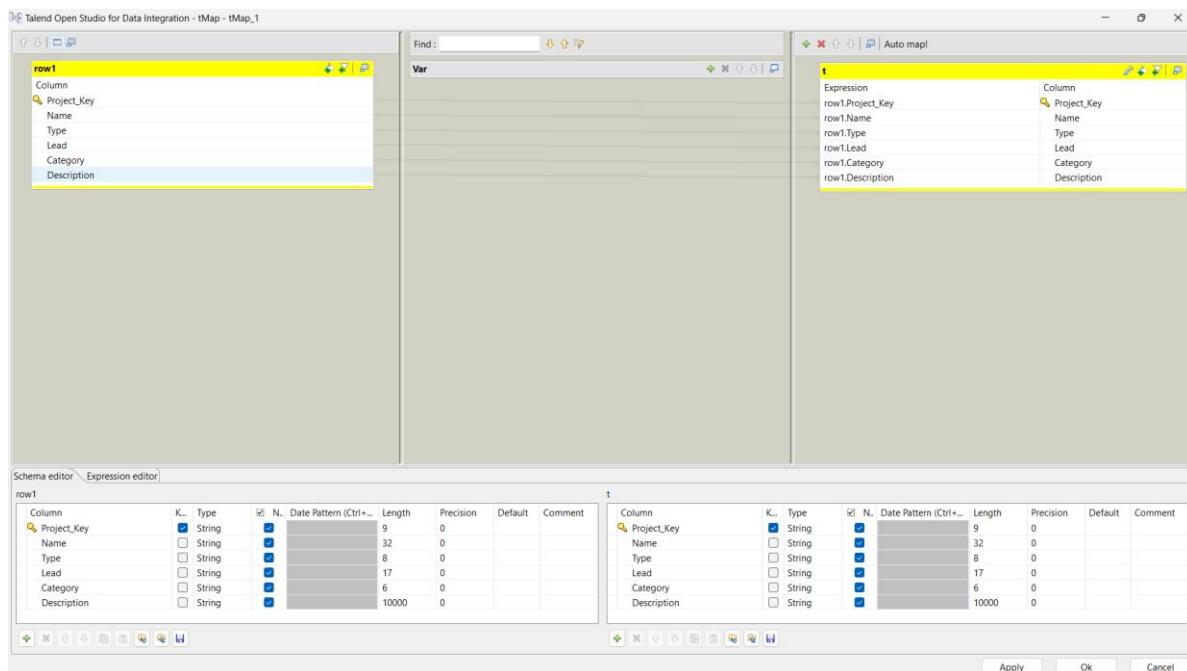


Figure 39 Composante Tmap Project

Dans la composante "DB Output", nous avons modifié le schéma de la table pour refléter les modifications nécessaires. De plus, nous avons également ajusté l'action sur les données à "update or insert". Cette modification permet d'achever la mise à jour de la table si celle-ci change, assurant ainsi que les données sont constamment à jour et cohérentes.

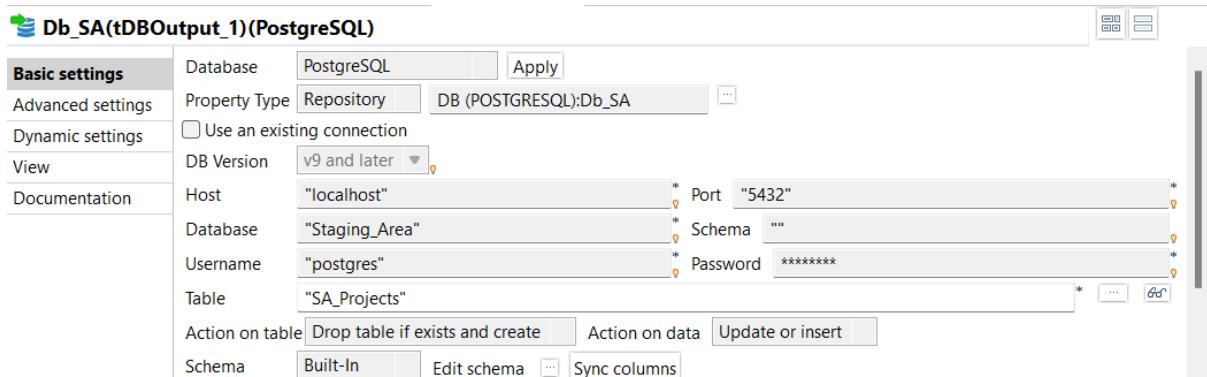


Figure 40 Composante DBOutput Projects

Voici la table créée :

	Project_Key [PK] character varying (9)	Name character varying (32)	Type character varying (8)	Lead character varying (17)	Category character varying (6)	Description character varying (10000)
1	ANN	z - Hibernate Annotations	software	Emmanuel Bernard	EOL	Hibernate Java5 annotations for metadata
2	BVAL	Bean Validation	software	Gunnar Morling	Active	Bean Validation spec
3	BVTCK	Bean Validation TCK	software	Gunnar Morling	Active	Issues related to TCK bugs and appeal prc
4	EJB	z - Hibernate Entity Manager	software	Emmanuel Bernard	EOL	Hibernate-based EJB 3.0 Persistence Prov
5	HB	z - Hibernate 2	software	GavinG	EOL	Issues pertaining to Hibernate 2.x
6	HBI	z - Hibernate 1.2	software	GavinG	EOL	Issues relating to Hibernate 1.2.x
7	HBX	Hibernate Tools	software	Koen Aers	Active	Bug area for Hibernate Tools
8	HCANN	Hibernate Commons Annotations	software	Sanne Grinovero	Active	Utility project for annotation handling
9	HHH	Hibernate ORM	software	Steve Ebersole	Active	Project for tracking issues with Hibernate
10	HQLPARSER	HQL/JPQL Parser	software	Sanne Grinovero	Active	Common parse and normalization functio
11	HREACT	Hibernate Reactive	software	Davide D'Alto	Active	[null]
12	HSEARCH	Hibernate Search	software	Marko Bekhtya	Active	Hibernate Search: full-text indexing and se
13	HSHARDS	Hibernate Shards	software	Max Ross	Active	Horizontal partitioning for Hibernate
14	HV	Hibernate Validator	software	Guillaume Smet	Active	Annotations based domain model constra
15	HVAL	z - Hibernate Validator 3	software	Hardy Ferentschik	EOL	[null]
16	JPA	Java Persistence API	software	Steve Ebersole	Active	Project for issues pertaining to the JPA Af
17	METAGEN	z- Hibernate Metamodel Generator	software	Hardy Ferentschik	EOL	A Java 6 annotation processor used to ge
18	OGM	Hibernate OGM	software	Emmanuel Bernard	Active	Hibernate Object/Grid Mapper
19	SQM	Hibernate Semantic Query	software	Steve E		Successfully run. Total query runtime: 168 msec. 21 rows affected. X

Figure 41 Table Projects SA

### 3.2. Dim Issue

Pour la dimension "issues", nous avons suivi un processus similaire. Cependant, dans ce cas, nous avons utilisé la composante TMap pour transformer les deux champs "created" et "updated" d'un format string "yyyy-MM-dd'T'HH:mm:ss.SSSX" à un format date "yyyy-MM-dd HH:mm". Cette transformation permettra ensuite de calculer le temps nécessaire pour la résolution du ticket.

Voici la fonction utilisée pour la transformation :

```
TalendDate.parseDate("yyyy-MM-dd HH:mm", TalendDate.formatDate("yyyy-MM-dd HH:mm", TalendDate.parseDate("yyyy-MM-dd'T'HH:mm:ss.SSSX", row1.Created)))
```

De plus, nous avons augmenté la longueur de tous les champs string, car de nombreux champs, tels que "summary" et "description", contiennent des valeurs longues.

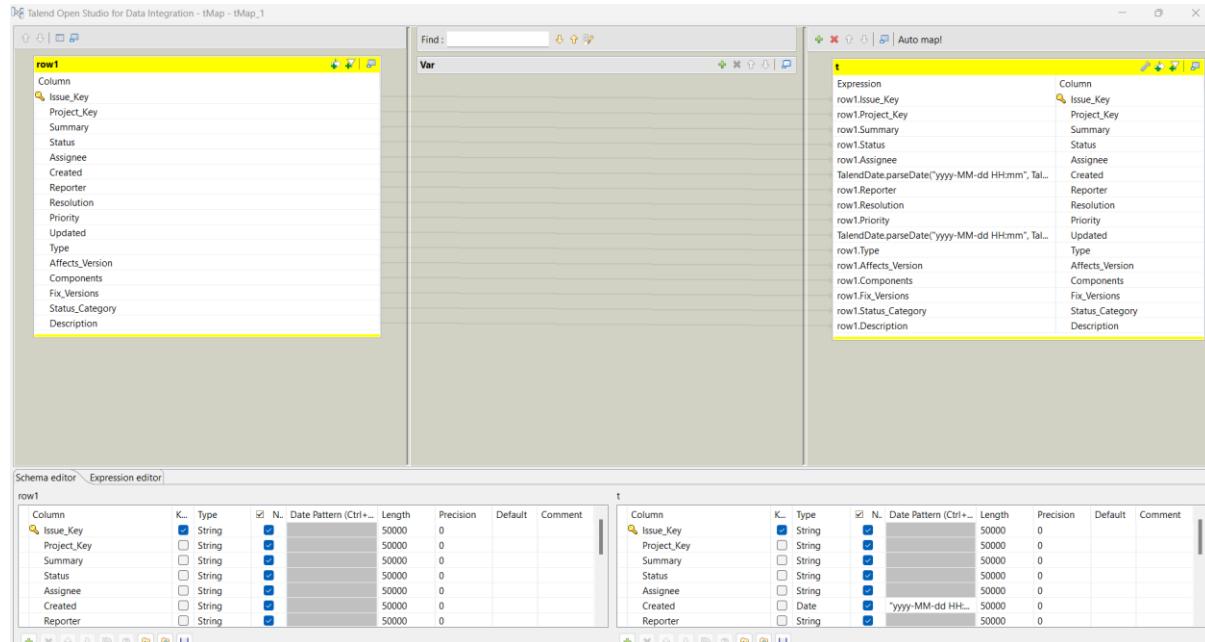


Figure 42 Composante Tmap Issues

Voici la table créée :

	Issue_Key [PK] caracte... character var	Project_Key character var	Summary character var	Status character var	Assignee character var	Created timestamp without ti...	Reporter character varying	Resolution character var	Priority character var	Updated timestamp wi...	Type character var	Affects_Versi... character var	Com... character var
1	ANN-1	ANN	Support ...	Resolved	Emmanue...	2005-04-05 17:49:...	Emmanuel B...	Fixed	Critical	2007-11-1...	New Feat...	3.1beta3	[bin]
2	ANN-100	ANN	cannot m...	Resolved	[null]	2005-09-29 20:29:...	Dave Bronds...	Rejected	Minor	2006-02-1...	Bug	3.1beta5	[bin]
3	ANN-101	ANN	@MapKey...	Resolved	[null]	2005-09-29 23:09:...	Christian Bau...	Rejected	Minor	2005-10-0...	Bug	[null]	[bin]
4	ANN-102	ANN	Play nice...	Resolved	[null]	2005-09-30 11:06:...	Emmanuel B...	Rejected	Minor	2005-10-0...	Improvement...	3.1beta5	[vali
5	ANN-103	ANN	Allow to s...	Resolved	Emmanue...	2005-10-01 02:06:...	Christian Bau...	Fixed	Minor	2007-02-1...	New Feat...	[null]	[bin]
6	ANN-104	ANN	Allow SQ...	Resolved	Emmanue...	2005-10-01 21:28:...	Christian Bau...	Fixed	Major	2007-02-1...	Improvement...	[null]	[bin]
7	ANN-105	ANN	More exc...	Closed	[null]	2005-10-02 20:25:...	Max Rydahl ...	Fixed	Major	2005-10-0...	Improvement...	[null]	[bin]
8	ANN-106	ANN	@Id does ...	Resolved	[null]	2005-10-02 22:32:...	mike perham	Rejected	Critical	2005-10-0...	Bug	3.1beta5	[bin]
9	ANN-108	ANN	@Index d...	Resolved	[null]	2005-10-04 12:22:...	MichaelM	Rejected	Minor	2006-09-0...	Bug	3.1beta5	[]
10	ANN-109	ANN	@Index d...	Resolved	[null]	2005-10-04 20:18:...	Emmanuel B...	Fixed	Minor	2005-10-0...	Bug	[null]	[bin]
11	ANN-11	ANN	Enum sho...	Closed	Emmanue...	2005-07-01 00:37:...	Emmanuel B...	Fixed	Major	2005-07-0...	Improvement...	3.1beta3	[bin]
12	ANN-110	ANN	non-exist...	Resolved	[null]	2005-10-06 19:24:...	Dennis C. Byr...	Rejected	Major	2005-10-0...	Bug	3.1beta4	[]
13	ANN-111	ANN	add functi...	Resolved	[null]	2005-10-07 14:02:...	Gregor Melh...	Fixed	Major	2005-11-1...	New Feat...	[null]	[vali
14	ANN-113	ANN	mapping ...	Resolved	[null]	2005-10-10 17:09:...	Panagiotis K...	Rejected	Major	2006-03-2...	Bug	3.1beta6	[bin]
15	ANN-114	ANN	NPE whe...	Resolved	[null]	2005-10-10 18:28:...	Emmanuel B...	Fixed	Minor	2005-10-1...	Bug	3.1beta6	[bin]
16	ANN-115	ANN	@Index d...	Resolved	Emmanue...	2005-10-12 17:57:...	Petr Matejka	Fixed	Minor	2005-11-1...	Bug	3.1beta6	[]
17	ANN-116	ANN	fetch vali...	Resolved	[null]	2005-10-14 01:15:...	Gregor Melh...	Fixed	Major	2005-11-1...	New Feat...	[null]	[vali
18	ANN-117	ANN	setOrpha...	Resolved	[null]	2005-10-14 12:13:...	Emmanuel B...	Fixed	Major	2005-10-1...	Bug	3.1beta6	[bin]
19	ANN-118	ANN	Duplicate ...	Resolved	Emmanue...	2005-10-14 16:50:...	Emmanuel B...	Fixed	Minor	2006-04-3...	Bug	3.1beta6	[bin]

Figure 43 Table Issues SA

### 3.3. Dim Comment

Pour la dimension "comments", nous avons utilisé la fonction `Numeric.sequence("s1",1,1)` de Talend pour créer une colonne qui contiendra l'identifiant de chaque commentaire. Cela garantit l'unicité de chaque commentaire et facilite leur gestion dans la base de données.

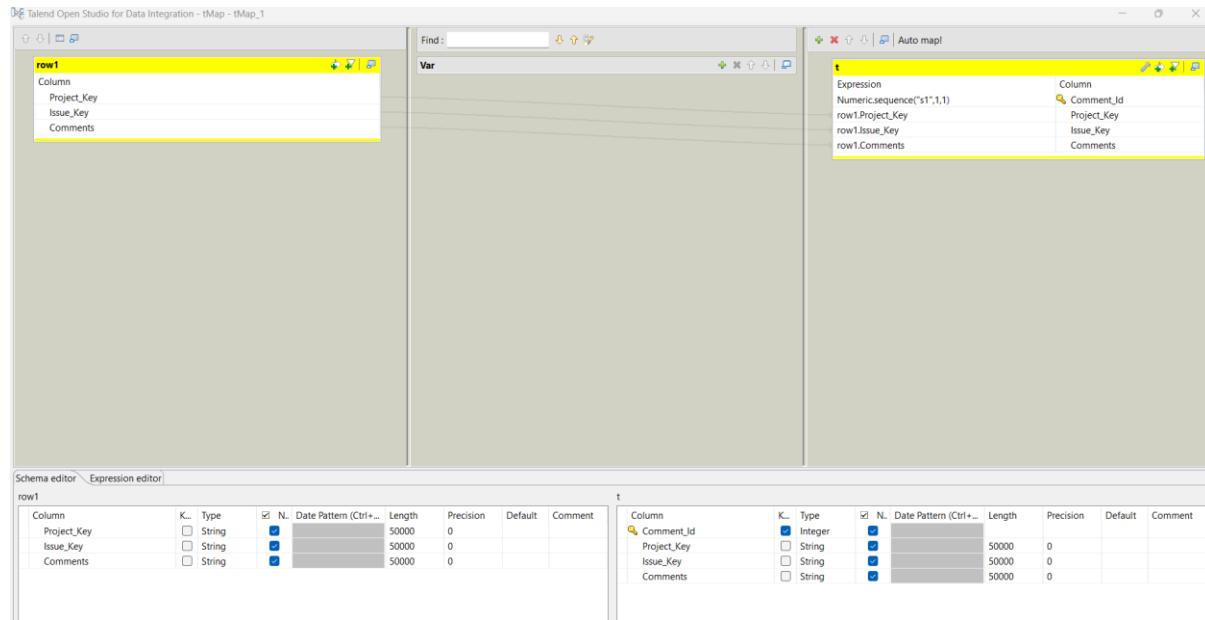


Figure 44 Composante Tmap Comment

Voici la table créée :

	Comment_Id [PK] bigint	Project_Key character varying (50000)	Issue_Key character varying (50000)	Comments character varying (50000)
1	1	ANN	ANN-1	- Emmanuel Bernard:Done partially. See
2	2	ANN	ANN-100	- Dave Brondsema:Actually you could probably keep all the method signatures the same but if it is a Parameterize
3	3	ANN	ANN-101	- Emmanuel Bernard:Year, it's actually impossible to use @MapKey in a many to many in Hibernate Core
4	4	ANN	ANN-102	- Emmanuel Bernard:It does already, the user was crazy
5	5	ANN	ANN-103	- Amir Pashazadeh:But I believe the optional="false" is a necessity too. Cause with optional="false" and @Second
6	6	ANN	ANN-104	- Emmanuel Bernard:The rest is already supported since 3.2.0
7	7	ANN	ANN-105	- Max Rydahl Andersen:done after emmanuel's blessing
8	8	ANN	ANN-106	- GavinG:"Critical"?? Sheesh.
9	9	ANN	ANN-108	- Emmanuel Bernard:Tested, it works.
10	10	ANN	ANN-109	none
11	11	ANN	ANN-11	none
12	12	ANN	ANN-110	- Emmanuel Bernard:HA raise an exception and I'm -1 on checking the actual signature.
13	13	ANN	ANN-111	- Gregor Melhorn:that was too fast. Made a mistake on the first function calling the second one, anyway it should
14	14	ANN	ANN-113	- Emmanuel Bernard:@MapKey is not supposed to support entity as index, see
15	15	ANN	ANN-114	none
16	16	ANN	ANN-115	- Emmanuel Bernard:The user is probably wrong on the failing version
17	17	ANN	ANN-116	none
18	18	ANN	ANN-117	none
19	19	ANN	ANN-119	none

Figure 45 Table Comments SA

## 4. Chargement de la data warehouse

Un entrepôt de données, ou datawarehouse, est un élément clé dans la Business Intelligence. C'est essentiellement une base de données conçue pour stocker et gérer de grandes quantités de données pour l'analyse. Pour notre projet, nous avons créé la base de données DataWarehouse pour la connecter à Talend pour faciliter le transfert de données.

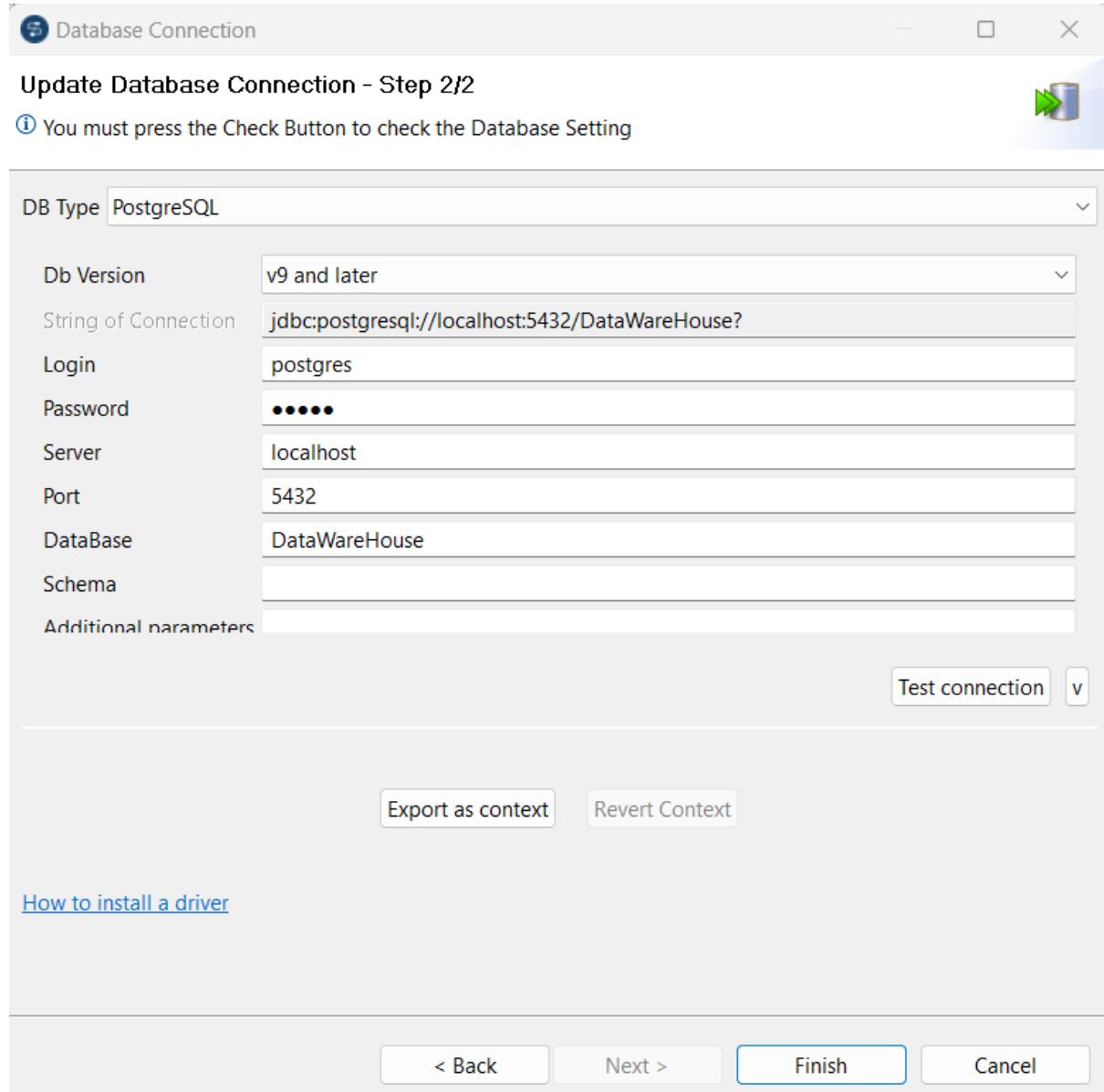


Figure 46 Crédation de la DB DW

### 4.1. Dimensions

Pour chaque dimension (comments, issues et projects), nous avons créé un job dédié. Dans chaque job, nous avons utilisé les composantes suivantes :

- La composante DB Input pour extraire les données de la staging area.
- La composante TMap pour effectuer les transformations nécessaires sur les données.

- La composante DB Output pour charger les données transformées dans la DataWarehouse.

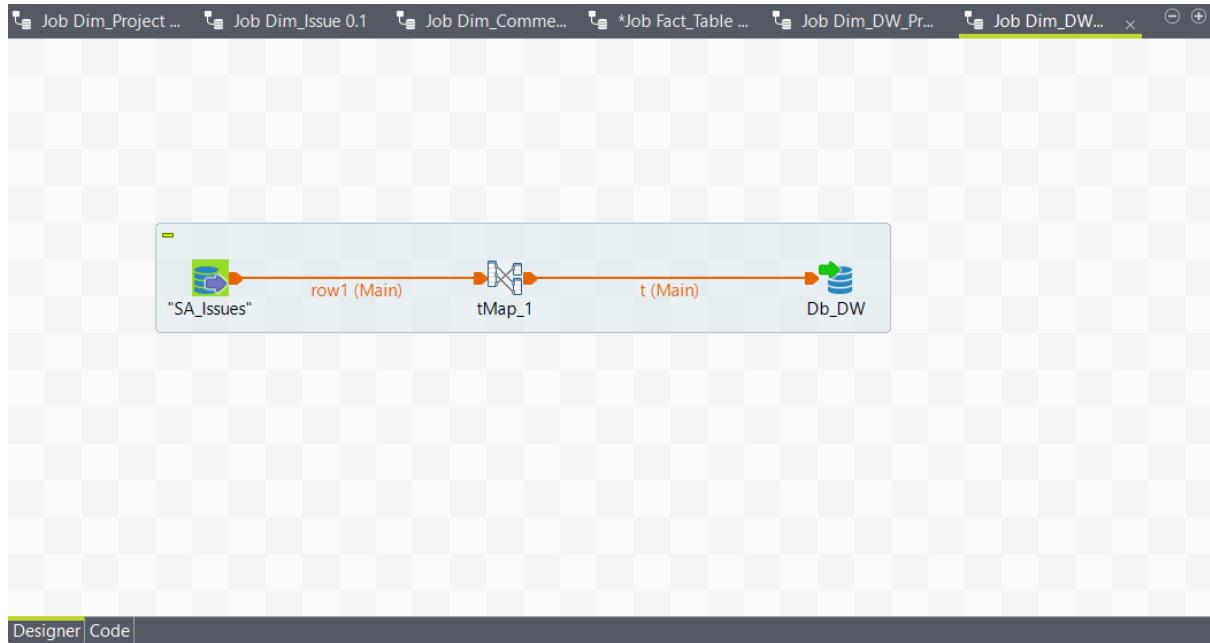


Figure 47 Job Dim\_DW\_Issues

## 4.2. Fact table

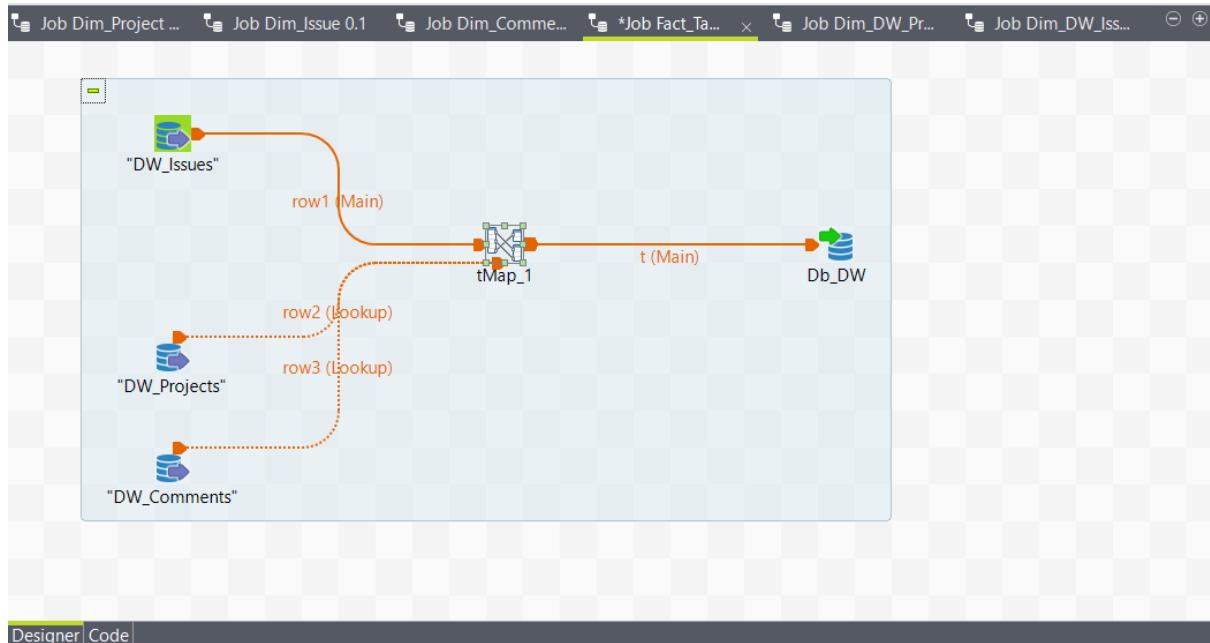


Figure 48 Job Fact\_Table

Pour la table de faits (fact table), nous avons utilisé la composante TMap pour connecter les clés primaires de chaque dimension et les charger dans la table de faits. Ainsi, cette table contiendra chaque clé de chaque issue, son projet et son commentaire, permettant ainsi une analyse complète des données.

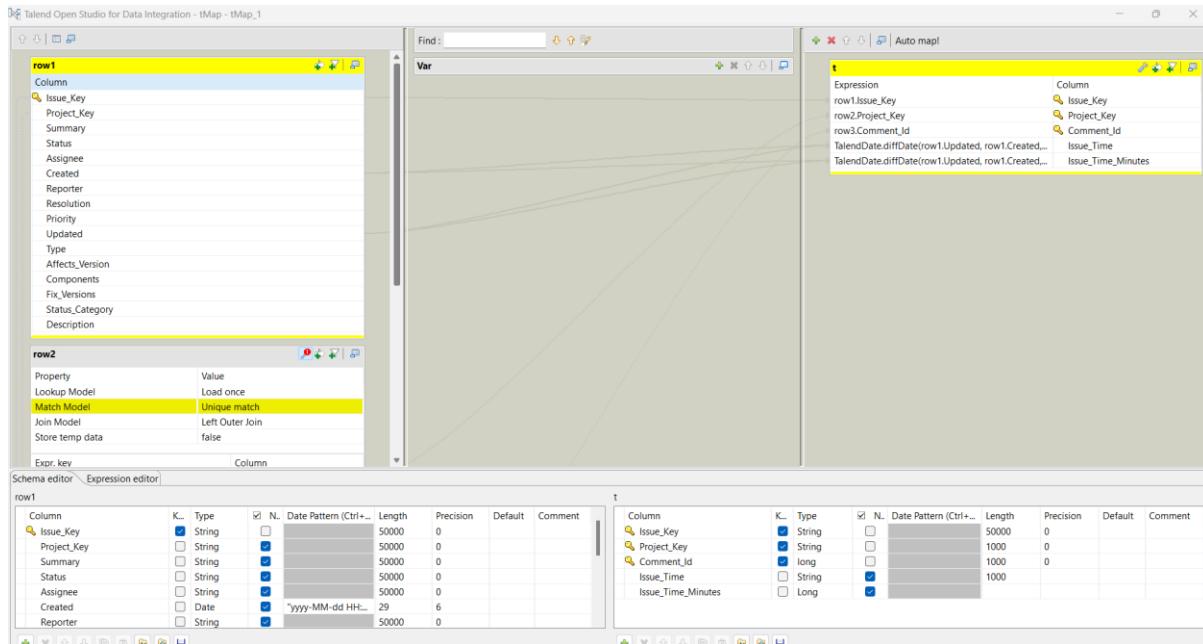


Figure 49 Composante Tmap Fact Table

Pour connecter les clés primaires, nous avons utilisé une méthode de "Left Outer Join" dans la composante TMap. Cela garantit que toutes les clés de la table de faits seront incluses, même si elles ne correspondent pas à une clé dans une dimension. Ensuite, nous avons utilisé une opération de "Lookup Load Once" pour charger les clés primaires de chaque dimension une seule fois, assurant ainsi une performance optimale lors de la création de la table de faits.

En outre, nous avons créé le KPI (Key Performance Indicator) "Issue\_time" qui calcule le temps de résolution de chaque ticket. Ce KPI fournit des informations cruciales sur les performances de résolution des problèmes.

```
TalendDate.diffDate(row1.Updated, row1.Created, "dd") + " days, " +
    (TalendDate.diffDate(row1.Updated,           row1.Created,
"HH") % 24) + " hours, " +
    (TalendDate.diffDate(row1.Updated,           row1.Created,
"mm") % 60) + " minutes";
```

De plus, nous avons également créé le KPI "Issue\_time\_minutes" qui donne le temps en minutes pour la comparaison de chaque ticket. Cette mesure permet une granularité plus fine dans l'analyse du temps de résolution des problèmes.

```
TalendDate.diffDate(row1.Updated, row1.Created, "mm");
```

Le reste des mesures KPI du projet seront créées dans la plateforme Power BI grâce au langage DAX.

Voici donc la table créée :

	Issue_Key [PK] character varying (50000)	Project_Key [PK] character varying (1000)	Comment_Id [PK] bigint	Issue_Time character varying (1000)	Issue_Time_Minutes bigint
1	ANN-1	ANN	1	950 days, 14 hours, 0 minutes	1368840
2	ANN-100	ANN	2	134 days, 9 hours, 28 minutes	193528
3	ANN-101	ANN	3	6 days, 17 hours, 36 minutes	9696
4	ANN-102	ANN	4	6 days, 5 hours, 42 minutes	8982
5	ANN-103	ANN	5	501 days, 1 hours, 12 minutes	721512
6	ANN-104	ANN	6	500 days, 5 hours, 48 minutes	720348
7	ANN-105	ANN	7	0 days, 18 hours, 14 minutes	1094
8	ANN-106	ANN	8	1 days, 6 hours, 21 minutes	1821
9	ANN-108	ANN	9	332 days, 14 hours, 29 minutes	478949
10	ANN-109	ANN	10	1 days, 20 hours, 11 minutes	2651
11	ANN-11	ANN	11	5 days, 21 hours, 7 minutes	8467
12	ANN-110	ANN	12	0 days, 1 hours, 53 minutes	113
13	ANN-111	ANN	13	41 days, 6 hours, 7 minutes	59407
14	ANN-113	ANN	14	170 days, 5 hours, 47 minutes	245147
15	ANN-114	ANN	15	0 days, 0 hours, 1 minutes	1
16	ANN-115	ANN	16	36 days, 21 hours, 9 minutes	53109
17	ANN-116	ANN	17	35 days, 5 hours, 48 minutes	50748
18	ANN-117	ANN	18	0 days, 0 hours, 28 minutes	28
19	ANN-118	ANN	19	198 days, 5 hours, 53 minutes	285473
			20	21 days, 6 hours, 29 minutes	46020

Figure 50 Fact Table

## 5. Mise en place de l'environnement Elasticsearch et Kibana

Elasticsearch est un moteur de recherche et d'analyse distribué, performant et flexible, largement utilisé pour indexer, rechercher et analyser des volumes importants de données en temps réel. Sa structure repose sur des concepts tels que les documents, les index et les champs, offrant une capacité de recherche rapide et efficace.

Kibana, quant à lui, est un outil de visualisation et de gestion conçu pour fonctionner avec Elasticsearch. Il permet de créer des tableaux de bord interactifs, d'explorer les données indexées et de surveiller les performances en temps réel. Ensemble, Elasticsearch et Kibana constituent un écosystème puissant pour la recherche et la visualisation des données, adapté à de nombreux cas d'utilisation, comme l'analyse des journaux ou le suivi des performances.

### 5.1. Configuration de `elasticsearch.yml`

Le fichier `elasticsearch.yml` est le principal fichier de configuration d'Elasticsearch. Il permet de personnaliser le comportement et les paramètres du cluster selon les besoins spécifiques du projet. Ce fichier se trouve dans le répertoire de configuration (`config/`) après l'installation. Voici les principaux paramètres configurables :

#### Paramètres de base :

- `cluster.name` : Définit le nom du cluster Elasticsearch. Cela permet de distinguer différents clusters.
- `node.name` : Définit le nom du nœud au sein du cluster. Chaque nœud doit avoir un nom unique.

```

# ----- Cluster -----
#
# Use a descriptive name for your cluster:
#
cluster.name: Thecluster
#
# ----- Node -----
#
# Use a descriptive name for the node:
#
node.name: Thenode
#
# Add custom attributes to the node:
#
#node.attr.rack: r1
#

```

Figure 51 Config elasticsearch 1

### Chemins des données et des journaux :

- path.data : Spécifie le répertoire où Elasticsearch stockera les données.
- path.logs : Définit le répertoire où les journaux seront enregistrés.

```

# ----- Paths -----
#
# Path to directory where to store the data (separate multiple locations by comma):
#
path.data: /path/to/data
#
# Path to log files:
#
path.logs: /path/to/logs
#

```

Figure 52 Config elasticsearch 2

### Paramètres réseau :

- network.host : Définit l'adresse IP sur laquelle Elasticsearch écoutera. Par défaut, cela peut être localhost, mais pour autoriser l'accès externe, une IP spécifique ou 0.0.0.0 peut être utilisée.
- http.port : Définit le port HTTP (par défaut : 9200).

```

# ----- Network -----
#
# By default Elasticsearch is only accessible on localhost. Set a different
# address here to expose this node on the network:
#
#network.host: 0.0.0.0
#
# By default Elasticsearch listens for HTTP traffic on the first free port it
# finds starting at 9200. Set a specific HTTP port here:
#
#http.port: 9200
#

```

Figure 53 Config elasticsearch 3

#### Paramètres de sécurité :

- `xpack.security.enabled` : Active les fonctionnalités de sécurité comme l'authentification.

```

# Enable security features
xpack.security.enabled: true

xpack.security.enrollment.enabled: true

# Enable encryption for HTTP API client connections, such as Kibana, Logstash, and Agents
▽ xpack.security.http.ssl:
  enabled: true
  keystore.path: certs/http.p12

# Enable encryption and mutual authentication between cluster nodes
▽ xpack.security.transport.ssl:
  enabled: true
  verification_mode: certificate
  keystore.path: certs/transport.p12
  truststore.path: certs/transport.p12
# Create a new cluster with the current node only
# Additional nodes can still join the cluster later
cluster.initial_master_nodes: ["FFX5"]

```

Figure 54 Config elasticsearch 4

#### Paramètres pour un environnement de production :

- `discovery.seed_hosts` : Liste des adresses IP ou noms d'hôte des nœuds pour permettre leur découverte dans le cluster.
- `cluster.initial_master_nodes` : Définit les nœuds éligibles pour devenir maître lors du premier démarrage.

```

# ----- Discovery -----
#
# Pass an initial list of hosts to perform discovery when this node is started:
# The default list of hosts is ["127.0.0.1", "[::1]"]
#
# discovery.seed_hosts: ["host1", "host2"]
#
# Bootstrap the cluster using an initial set of master-eligible nodes:
#
# cluster.initial_master_nodes: ["node-1", "node-2"]
#
#

```

Figure 55 Config elasticseach 5

## 5.2. Installation de Kibana

L'installation de Kibana est une étape clé pour compléter l'environnement Elasticsearch, offrant une interface graphique intuitive pour l'exploration, l'analyse, et la visualisation des données. Kibana est un outil puissant qui permet de transformer des données brutes en insights exploitables grâce à ses tableaux de bord et ses fonctionnalités de recherche avancée.

Lors du premier démarrage de Kibana, Elasticsearch génère automatiquement les paramètres nécessaires pour la configuration de Kibana. Parmi ces paramètres figure le service account token, une clé essentielle qui permet d'établir une communication sécurisée entre Kibana et Elasticsearch. Cette étape simplifie considérablement le processus d'intégration, en éliminant le besoin de configurer manuellement ces paramètres critiques.

Une fois Kibana installé et configuré, il suffit de se connecter à l'adresse <http://localhost:5601> via un navigateur pour accéder à l'interface de gestion et d'analyse d'Elasticsearch.

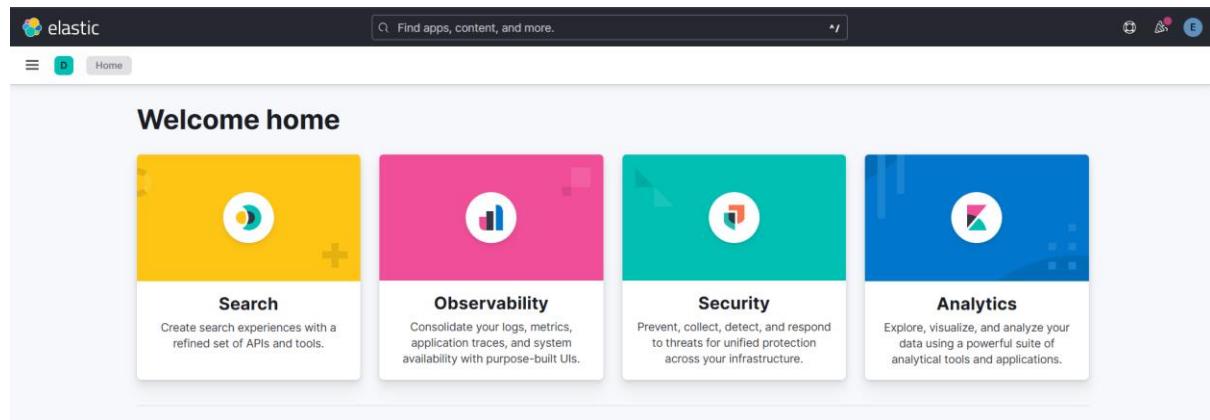


Figure 56 Interface Kibana

## 6. Connection de la base de données avec Elasticsearch

Dans cette partie, nous allons expliquer le processus de connexion entre Elasticsearch et PostgreSQL

### 6.1. Crédit du connecteur et de l'index Elasticsearch

Tout d'abord, nous avons créé notre index "pgindex", dans lequel nous allons stocker tous nos documents.

Index name	Index health	Docs count	Ingestion name	Ingestion method
pgindex	● green	100968	PostgreSQL	Connector

Figure 57 Index pgindex

Ensuite, il a fallu créer le connecteur "pgconnector" qui permettra de relier Elasticsearch à PostgreSQL. À la fois l'index et le connecteur ont été créés via l'interface de Kibana, ce qui a facilité leur configuration et leur gestion.

Connector name	Index name	Docs count	Connector type
pgconnector	pgindex	100968	 PostgreSQL

Figure 58 Connecteur pgconnector

Après cela, nous allons générer la clé API du connecteur, qui permettra d'assurer l'accès et la gestion des permissions nécessaires pour l'indexation des documents dans Elasticsearch.

The screenshot shows the 'Attach an index' configuration page. At the top, there's a note: 'This index will hold your data source content, and is optimized with default field mappings for relevant search experiences. Give your index a unique name and optionally set a default language analyzer for the index.' Below this is a dropdown labeled 'Associated index' containing 'pgindex'. A note below the dropdown says 'You can use an existing index or create a new one.' There's a 'Save configuration' button. Below this, a section titled 'Generate an API key' is expanded, showing a note: 'First, generate an Elasticsearch API key. This pgindex-connector key will enable read and write permissions for the connector to index documents to the created pgindex index. Save the key in a safe place, as you will need it to configure your connector.' There's a 'Generate API key' button.

Figure 59 Connection pgindex

## 6.2. Configuration du connecteur "pgconnector"

On passe maintenant à la configuration du connecteur. Cependant, ce connecteur n'étant compatible qu'avec un environnement Linux, il a été nécessaire d'installer WSL (Windows Subsystem for Linux) sur notre machine Windows.

WSL est une fonctionnalité qui permet d'exécuter une distribution Linux directement sous Windows, offrant un environnement natif pour les applications ou outils nécessitant Linux. Cette installation nous a permis de surmonter la limitation du connecteur et de faciliter le déploiement de notre solution d'intégration de données.

Une fois WSL installé, voici les étapes à suivre pour la configuration du connecteur entre Elasticsearch et PostgreSQL :

- **Cloner le dépôt des connecteurs :**

La première étape consiste à cloner ou forkler le dépôt des connecteurs depuis GitHub sur votre machine locale à l'aide de la commande suivante :

```
git clone https://github.com/elastic/connectors
```

- **Générer le fichier de configuration initial :**

Ensuite, il faut exécuter la commande suivante pour générer le fichier config.yml initial, qui contiendra les paramètres de configuration nécessaires à la connexion entre PostgreSQL et Elasticsearch :

### *make config.yml*

- **Modifier le fichier de configuration :**

Une fois le fichier config.yml généré, ouvrez-le dans l'éditeur de votre choix. Vous devrez alors ajuster certains paramètres spécifiques pour que la configuration corresponde à votre environnement.

Dans le fichier config.yml, il faut remplacer les valeurs suivantes :

- **host** : L'URL de votre endpoint Elasticsearch.
- **api\_key** : La clé API générée pour authentifier la connexion avec Elasticsearch.
- **connector\_id** : L'ID unique du connecteur que vous avez créé dans Kibana.
- **service\_type** : Le type de service (ici, il s'agira de postgresql).

```
## ----- Connectors -----
#
## The list of connector clients/customized connectors configurations.
## Each object in the list requires 'connector_id' and 'service_type'.
## An example is:
connectors:
  -
    connector_id: -vBGWpAB1tm2evHz7ml1 # the ID of the connector.
    service_type: postgresql # The service type of the connector.
    api_key: LV9CSFdwQUIxdG0yZXZIek1XbE86ZkdaV2Z0Q2VUUGVsNfV5UXNqYWZ2UQ==
#
#
```

Figure 60 Config connecteur 1

```
## ----- Elasticsearch -----
#
## The host of the Elasticsearch deployment.
elasticsearch.host: https://192.168.1.8:9200
#
#
## The API key for Elasticsearch connection.
## Using 'api_key' is recommended instead of 'username'/'password'.
elasticsearch.api_key: LV9CSFdwQUIxdG0yZXZIek1XbE86ZkdaV2Z0Q2VUUGVsNfV5UXNqYWZ2UQ==
#
#
```

Figure 61 Config connecteur 2

## 6.3. Configuration des fichiers config PostgreSQL

En deuxième temps, il est essentiel de configurer PostgreSQL pour permettre les connexions distantes, ce qui permet à Elasticsearch (via le connecteur) de se connecter à la base de données. Voici les étapes à suivre :

- **Modification du fichier postgresql.conf :**

Nous avons modifié le fichier de configuration postgresql.conf pour permettre à PostgreSQL d'accepter les connexions depuis des adresses IP distantes.

Figure 62 Config postgresql 1

- Modification du fichier pg\_hba.conf :

Ensuite, il faut mettre à jour le fichier pg\_hba.conf pour autoriser les connexions depuis des adresses IP spécifiques, comme celles de Logstash ou du serveur Elasticsearch.

```
# TYPE DATABASE      USER          ADDRESS         METHOD
# "local" is for Unix domain socket connections only
local  all            all
# IPv4 local connections:
host   all            all            127.0.0.1/32
# IPv6 local connections:
host   all            all            ::1/128
# Allow replication connections from localhost, by a user with the
# replication privilege.
local  replication    all
host   replication    all            127.0.0.1/32
host   replication    all            ::1/128
host   all            all            172.21.88.193/32
host   DataWareHouse  postgres       192.168.1.8/32
                                         md5
                                         md5
```

Figure 63 Config postgresql 2

Nous avons choisi l'adresse 192.168.1.8/32 pour la connexion de l'utilisateur postgres à la base de données DataWareHouse dans le fichier pg\_hba.conf car nous utilisons Windows Subsystem for Linux (WSL).

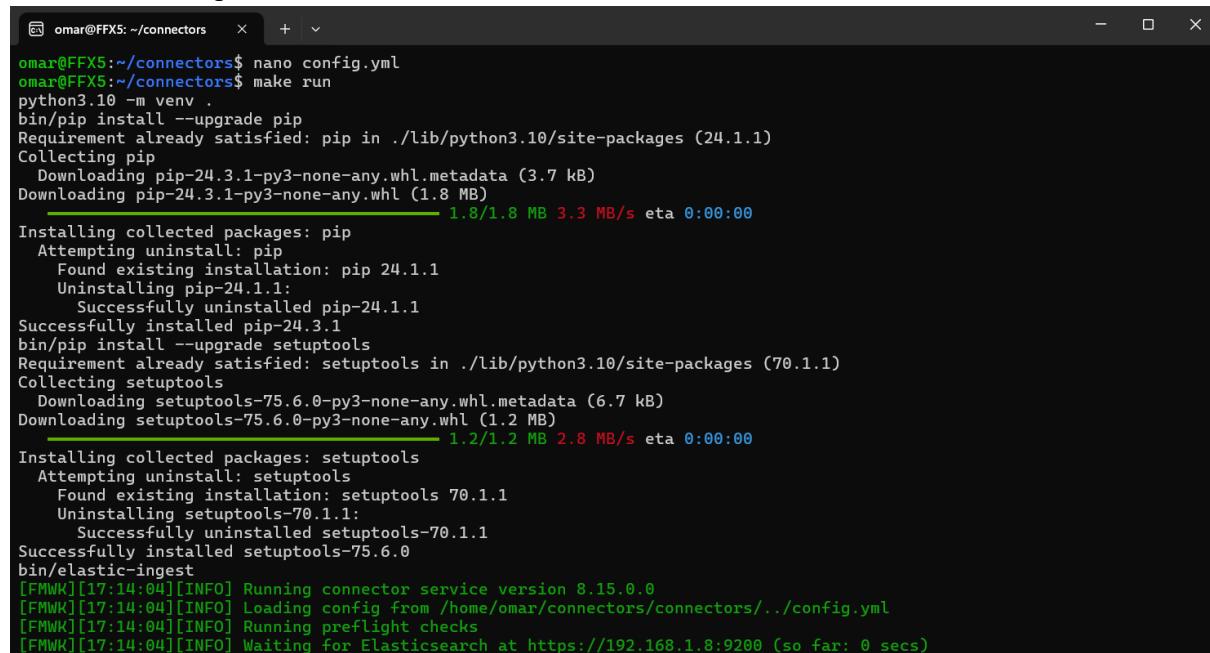
En effet, WSL se comporte comme une machine virtuelle (VM), ce qui signifie que les adresses IP utilisées dans cet environnement peuvent différer de celles de l'hôte Windows. L'adresse 192.168.1.8 correspond à l'IP assignée à la machine virtuelle WSL, permettant ainsi à PostgreSQL d'accepter des connexions provenant de cet environnement.

En attribuant cette adresse IP spécifique dans le fichier pg\_hba.conf, nous garantissons que la connexion à PostgreSQL est autorisée à partir de WSL, en utilisant la méthode d'authentification md5 pour sécuriser les connexions par mot de passe.

## 6.4. Exécution du connecteur "pgconnector"

Maintenant, nous allons exécuter notre connecteur depuis WSL et activer la synchronisation depuis Kibana afin que toutes les données de notre Data Warehouse soient stockées dans des documents dans l'index pgindex.

Dans cette étape, nous avons accédé au répertoire racine du clone/fork du connecteur depuis le terminal ou l'IDE. Nous avons ensuite exécuté les commandes make install pour installer les dépendances nécessaires, puis make run pour démarrer le service du connecteur. Une fois ces commandes exécutées, le service du connecteur a été lancé avec succès, et l'interface utilisateur a confirmé que le connecteur était bien connecté à l'instance Elasticsearch.



```
omar@FFX5: ~/connectors x + v
omar@FFX5:~/connectors$ nano config.yml
omar@FFX5:~/connectors$ make run
python3.10 -m venv .
bin/pip install --upgrade pip
Requirement already satisfied: pip in ./lib/python3.10/site-packages (24.1.1)
Collecting pip
  Downloading pip-24.3.1-py3-none-any.whl.metadata (3.7 kB)
  Downloading pip-24.3.1-py3-none-any.whl (1.8 MB) 1.8/1.8 MB 3.3 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 24.1.1
    Uninstalling pip-24.1.1:
      Successfully uninstalled pip-24.1.1
Successfully installed pip-24.3.1
bin/pip install --upgrade setuptools
Requirement already satisfied: setuptools in ./lib/python3.10/site-packages (70.1.1)
Collecting setuptools
  Downloading setuptools-75.6.0-py3-none-any.whl.metadata (6.7 kB)
  Downloading setuptools-75.6.0-py3-none-any.whl (1.2 MB) 1.2/1.2 MB 2.8 MB/s eta 0:00:00
Installing collected packages: setuptools
  Attempting uninstall: setuptools
    Found existing installation: setuptools 70.1.1
    Uninstalling setuptools-70.1.1:
      Successfully uninstalled setuptools-70.1.1
Successfully installed setuptools-75.6.0
bin/elastic-ingest
[FMWK][17:14:04][INFO] Running connector service version 8.15.0.0
[FMWK][17:14:04][INFO] Loading config from /home/omar/connectors/connectors/.../config.yml
[FMWK][17:14:04][INFO] Running preflight checks
[FMWK][17:14:04][INFO] Waiting for Elasticsearch at https://192.168.1.8:9200 (so far: 0 secs)
```

Figure 64 Connection réussite

Puis, nous avons activé la synchronisation du connecteur depuis l'interface Kibana. Cette étape a permis de configurer le connecteur pour qu'il commence à synchroniser les données entre PostgreSQL et Elasticsearch. Grâce à cette synchronisation, toutes les données de notre Data Warehouse ont été automatiquement indexées dans l'index pgindex dans Elasticsearch.

Kibana a affiché un message de confirmation indiquant que la synchronisation avait bien été activée et que le processus se déroulait sans erreur.

## Event log

### ✓ Sync complete

Completed at Jun 28, 2024 12:12 PM

### ⌚ Sync started manually

Started at Jun 28, 2024 12:11 PM

## Sync

ID	Index name
IH9TXpABhZXU0-yIIEOY	pgindex

## Documents

Upserted ⓘ	Deleted ⓘ	Volume ⓘ
100968	0	About 341mb

Figure 65 Synchronisation complète

Ainsi, voilà notre index, incluant les documents indexés provenant de notre Data Warehouse. Toutes les données ont été correctement transférées et stockées dans l'index pgindex d'Elasticsearch, prêtes à être utilisées pour les recherches et analyses. Grâce à la synchronisation activée via Kibana, nous avons assuré une mise à jour continue des données, garantissant leur disponibilité et leur intégrité dans l'index.

The screenshot shows the pgindex interface with the following details:

- Header:** pgindex, View in Playground, Sync
- Navigation:** Overview, Documents (selected), Index mappings, Configuration, Sync rules, Scheduling, Pipelines
- Browse documents:** Shows 25 of 10,000 documents. A search bar is present.
- Document 1 (Document id: DataWareHouse\_public\_DW\_Issues\_BVAL-444):**

t	public_dw_issues_assignee → "Emmanuel Bernard"
t	schema → "public"
t	public_dw_issues_priority → "Major"
- Document 2 (Document id: DataWareHouse\_public\_DW\_Issues\_BVAL-446):**

t	public_dw_issues_assignee → "Emmanuel Bernard"
t	schema → "public"
t	public_dw_issues_priority → "Major"

Figure 66 Documents dans pgindex

## 7. Conclusion

En conclusion, ce chapitre a permis de décrire de manière détaillée les différentes étapes allant du chargement des données depuis la staging area jusqu'à leur structuration et optimisation dans le data warehouse. Par la suite, nous avons présenté la mise en place de notre environnement Elasticsearch. Nous avons également expliqué le processus de connexion entre PostgreSQL et Elasticsearch, permettant de créer un index dans Elasticsearch pour stocker les documents et assurer des recherches rapides et performantes. Grâce à cette configuration, nous avons réussi à organiser et indexer efficacement les données, optimisant ainsi leur accessibilité et leur utilisation dans le cadre de notre projet.

## Chapitre 5 : Accès aux données et analyse

1.	Introduction .....	80
2.	Dashboards Power BI.....	80
2.1.	Dashboard Issue 1 .....	80
2.2.	Dashboard Issue 2 .....	85
2.3.	Dashboard Projet .....	90
3.	Application Python Elasticsearch.....	96
3.1.	Indexation des embeddings dans Elasticsearch.....	96
3.2.	Application de recherche Elasticsearch.....	98
4.	Conclusion.....	102

# 1. Introduction

Dans ce chapitre, nous allons aborder l'analyse visuelle des données à travers les tableaux de bord Power BI, ainsi que la recherche avancée rendue possible grâce à Elasticsearch et un code Python dédié. L'objectif est de démontrer comment l'exploitation conjointe de ces outils permet de tirer pleinement parti des informations centralisées dans le Data Warehouse.

Nous commencerons par examiner les tableaux de bord Power BI, qui offrent une interface de visualisation intuitive pour explorer les données, identifier des tendances, et faciliter la prise de décision. Ensuite, nous présenterons le code Python responsable d'interroger Elasticsearch afin de réaliser des recherches efficaces et personnalisées sur les données indexées.

## 2. Dashboards Power BI

Après avoir intégré le Data Warehouse, hébergé sur PostgreSQL, dans Power BI, nous avons conçu trois tableaux de bord. Les deux premiers, Dashboard Issue 1 et Dashboard Issue 2, sont dédiés à l'analyse des données relatives aux issues, permettant d'étudier leur répartition, leur évolution, ainsi que leurs caractéristiques principales. Le dernier tableau de bord, Dashboard Projet, est focalisé sur l'examen des données liées aux projets.

### 2.1. Dashboard Issue 1

Le premier tableau de bord fournit une vue d'ensemble sur les issues, en mettant en avant leur répartition par type et par composants, leur temps moyen de résolution, ainsi qu'une analyse des assignés.

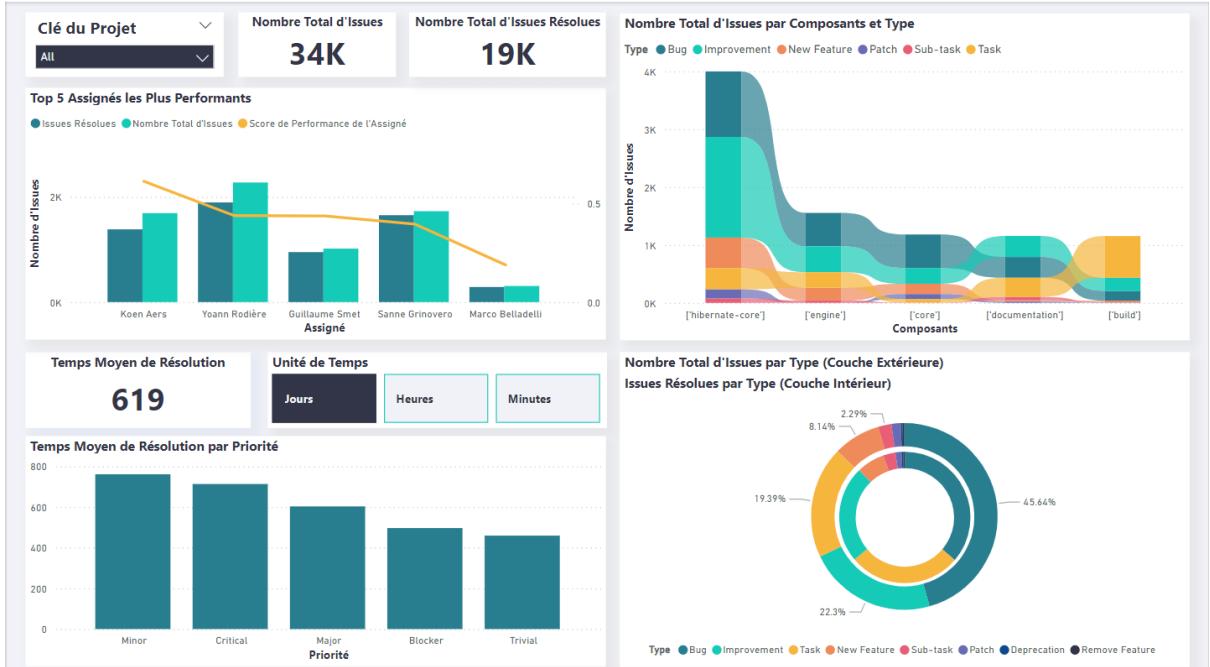


Figure 67 Dashboard Issue 1

L'élément Clé du projet permet de se concentrer sur un projet spécifique. En sélectionnant une clé de projet, tous les autres éléments du tableau de bord s'ajustent automatiquement pour afficher des données exclusivement liées à ce projet.

Ici, nous allons sélectionner le projet HHH afin d'analyser les éléments les plus pertinents de ce tableau de bord.

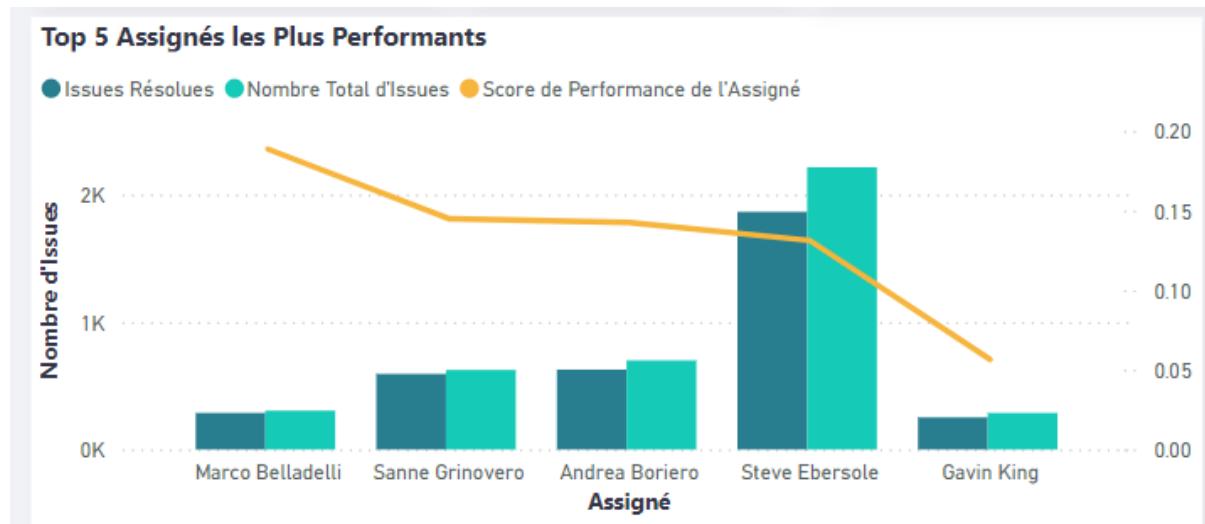


Figure 68 Top 5 Assignés les Plus Performants

### Aperçu :

L'élément Top 5 Assignés les Plus Performants met en avant les cinq assignés ayant obtenu les meilleurs scores de performance. Il fournit des informations sur le nombre total d'issues résolues, le nombre total d'issues assignées, et le score de performance de chaque assigné.

Le score de performance est calculé selon la formule suivante :

```

1 Assignee Performance Score =
2 IF(
3   [Resolved Issues] > 1,
4     DIVIDE(
5       [Resolved Issues],
6       [Avg Resolution Time for Fixed Issues (Hours)],
7       1
8     ),
9     BLANK()
10 )

```

Cette formule attribue un score basé sur le ratio entre le nombre d'issues résolues et le temps moyen de résolution des issues corrigées. Si un assigné a résolu moins de deux issues, le score n'est pas calculé. Ce KPI permet d'évaluer l'efficacité individuelle des membres de l'équipe.

### Conclusion :

Nous concluons que Marco Belladelli est l'assigné le plus performant grâce à son score élevé, reflétant une efficacité remarquable dans la résolution des issues. Cependant, Steve Ebersole se distingue par le nombre total d'issues le plus élevé, bien que son score soit légèrement inférieur à celui de Marco.

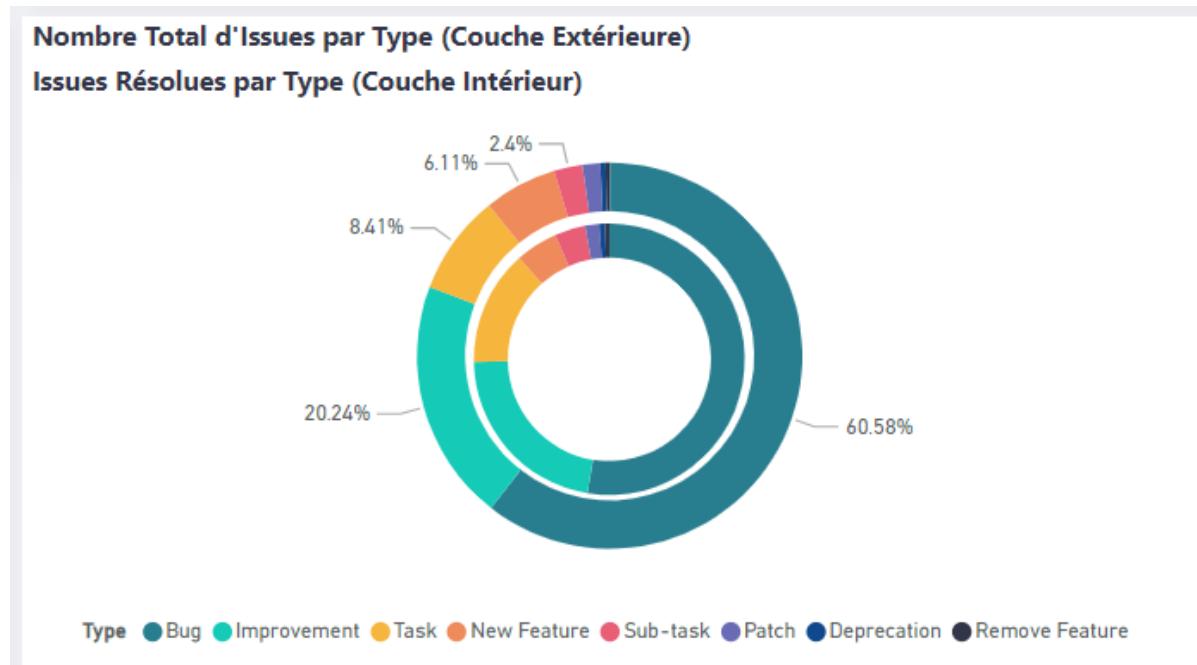


Figure 69 Nombre Total d'Issues et Issues Résolues par Type

### Aperçu :

L'élément Nombre Total d'Issues et Issues Résolues par Type est représenté sous forme d'un diagramme en anneau à deux couches. La couche extérieure illustre le nombre total d'issues par type, tandis que la couche intérieure met en évidence le nombre total d'issues résolues, également réparties par type. Cette visualisation permet de comparer rapidement le volume global des issues avec celles ayant été résolues.

Pour le KPI Issues Résolues, la formule utilisée est la suivante :

---

```

1 Resolved Issues =
2 CALCULATE(COUNT('public DW_Issues'[Issue_Key]), 'public DW_Issues'[Resolution] = "Fixed")

```

---

Figure 70 KPI Issues Résolues

Cette formule calcule le nombre total d'issues ayant une résolution marquée comme "Fixed". Cela nous permet d'identifier clairement le volume d'issues qui ont été corrigées avec succès.

### Conclusion :

L'analyse de l'élément Nombre Total d'Issues et Issues Résolues par Type révèle que les issues de type Bug sont les plus nombreux, suivis par ceux de type Improvement. Toutefois, il est important de noter que les issues sont généralement résolues indépendamment de leur type,

ce qui témoigne d'une gestion équilibrée et efficace des différentes catégories d'issues dans le projet.

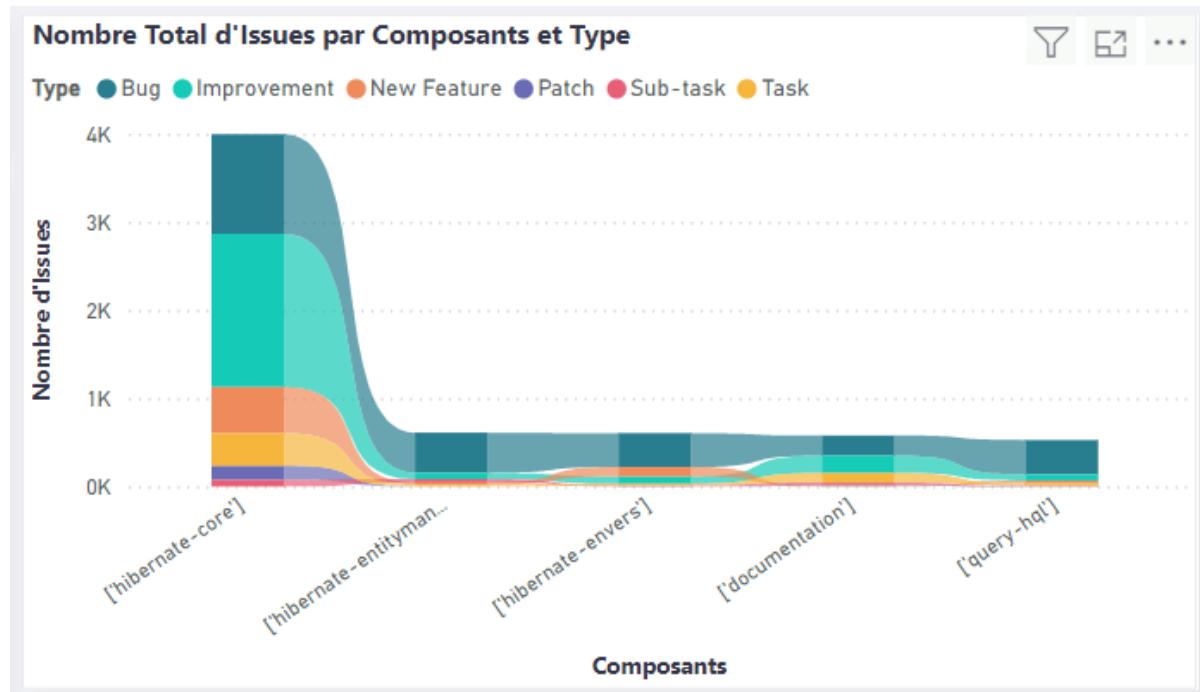


Figure 71 Nombre Total d'Issues par Composants et Type

#### Aperçu :

L'élément Nombre Total d'Issues par Composants et Type est représenté sous forme de Ribbon Chart. Ce graphique permet de visualiser la répartition du nombre total d'issues en fonction des composants et de leur type. Il offre une vue dynamique des variations et des relations entre les composants et les types d'issues.

#### Conclusion :

L'analyse de l'élément Nombre Total d'Issues par Composants et Type montre que le composant [Hibernate-core] est celui qui présente le nombre le plus élevé d'issues, principalement de type Bug et Improvement, comme c'est le cas pour les autres composants. En revanche, le composant [Hibernate-envers] se distingue par un nombre plus important d'issues de type New Feature par rapport aux issues de type Improvement.



Figure 72 Temps Moyen de Résolution par Priorité

### Aperçu :

La série d'éléments Temps Moyen de Résolution - Unité de Temps et Temps Moyen de Résolution par Priorité illustre la répartition du temps moyen de résolution en fonction du type d'issue. En sélectionnant l'unité de temps (jours, heures ou minutes), il est possible d'obtenir une visualisation adaptée de ce temps moyen selon l'échelle choisie.

Formule du KPI Temps Moyen de Résolution :

```

1 Avg Resolution Time (Hours) =
2     AVERAGEX(
3         FILTER(
4             'public DW_FactTable',
5             'public DW_FactTable'[Issue_Time_Minutes] > 0
6         ),
7             'public DW_FactTable'[Issue_Time_Minutes] / 60
8     )

```

Figure 73 KPI Temps Moyen de Résolution

Cette formule utilise la fonction AVERAGEX pour calculer la moyenne du temps de résolution en heures. Elle filtre les issues dont le temps de résolution (Issue\_Time\_Minutes) est supérieur à zéro afin d'exclure les valeurs non pertinentes ou invalides. Ensuite, le temps de résolution en minutes est converti en heures en le divisant par 60.

### Conclusion :

L'analyse de la série d'éléments Temps Moyen de Résolution - Unité de Temps et Temps Moyen de Résolution par Priorité révèle que le temps moyen de résolution dans ce projet est de

879 jours, ce qui semble improbable. Cette anomalie indique que de nombreuses issues restent ouvertes sans être clôturées, faussant ainsi la moyenne globale.

Par ailleurs, nous pouvons conclure que les issues ayant une priorité Minor sont celles qui sont les plus fréquemment résolues, suivies par celles ayant une priorité Critical, ce qui montre une gestion spécifique en fonction de l'urgence ou de l'importance perçue des issues.

## 2.2. Dashboard Issue 2

Le Dashboard Issue 2 présente une analyse temporelle de l'évolution des issues, permettant de suivre leur progression au fil du temps. Il inclut également une section dédiée à l'analyse des commentaires associés aux issues.



Figure 74 Dashboard Issue 2

Dans ce tableau de bord, nous allons poursuivre l'analyse du projet HHH.

Le KPI Nombre Moyen de Commentaire par Issue calcule la moyenne des commentaires associés à chaque issue.

Voici sa formule :

```
1 Avg Comments per Issue =
2 | DIVIDE([Estimated Comment Count], [Total Issues], 0)
```

Figure 75 KPI Nombre Moyen de Commentaire par Issue

La Mesure Estimated Comment Count calcule le nombre de commentaire de chaque série de commentaire dans la base de données comme suit

```

1 Estimated Comment Count =
2 SUMX(
3     'public DW_Comments',
4     LEN('public DW_Comments'[Comments]) - LEN(SUBSTITUTE('public DW_Comments'[Comments], UNICHAR(10) & "-", ""))
5 )

```

Figure 76 Mesure Estimated Comment Count

Tous les commentaires dans les séries de commentaire sont séparés par un " " suivit d'un "-" ce qui nous permet ici de calculer leur nombre.



The screenshot shows a Tableau dashboard titled "Clé de l'Issue". At the top left is a search bar with the placeholder "Search" and a magnifying glass icon. To the right of the search bar is a small edit icon. Below the search bar is a table with the following data:

Clé de l'Issue	Clé du Projet	Assigné	Type
HHH-1	HHH	Steve Ebersole	New Feature
HHH-10	HHH		New Feature
HHH-100	HHH	Steve Ebersole	Patch
HHH-1000	HHH	Steve Ebersole	Bug
HHH-10000	HHH		Improvement
HHH-10001	HHH	Sanne Grinovero	Task
HHH-10002	HHH	Steve Ebersole	Improvement
HHH-10003	HHH	Emmanuel Bernard	New Feature

Figure 77 Tableau Issues

### Aperçu :

Cet élément présente un tableau détaillant la clé de l'issue, la clé du projet, l'assigné de l'issue ainsi que son type. Il inclut également une fonctionnalité de recherche permettant de localiser une issue spécifique à l'aide de sa clé.

Clé de l'Issue	Clé du Projet	Assigné	Type
HHH-1	HHH	Steve Ebersole	New Feature
HHH-10			New Feature
HHH-100			Patch
HHH-1000			Bug
HHH-10000			Improvement
HHH-10001			Task
HHH-10002			Improvement
HHH-10003			New Feature

Evolution du Ta

Clé de l'Issue	Priorité	Composants
HHH-10	Major	['hibernate-core']
Version Affectée	Résolution	
3.1 rc3	Duplicate	
Temps de Résolution		
<b>0 days, 0 hours, 29 minutes</b>		

Figure 78 Tooltip Issue

En passant la souris sur une issue, des informations supplémentaires s'affichent, telles que sa priorité, son composant, sa version affectée, son état de résolution et le temps pris pour la résoudre. Ce tableau offre ainsi une vue complète et interactive pour analyser en détail les issues.

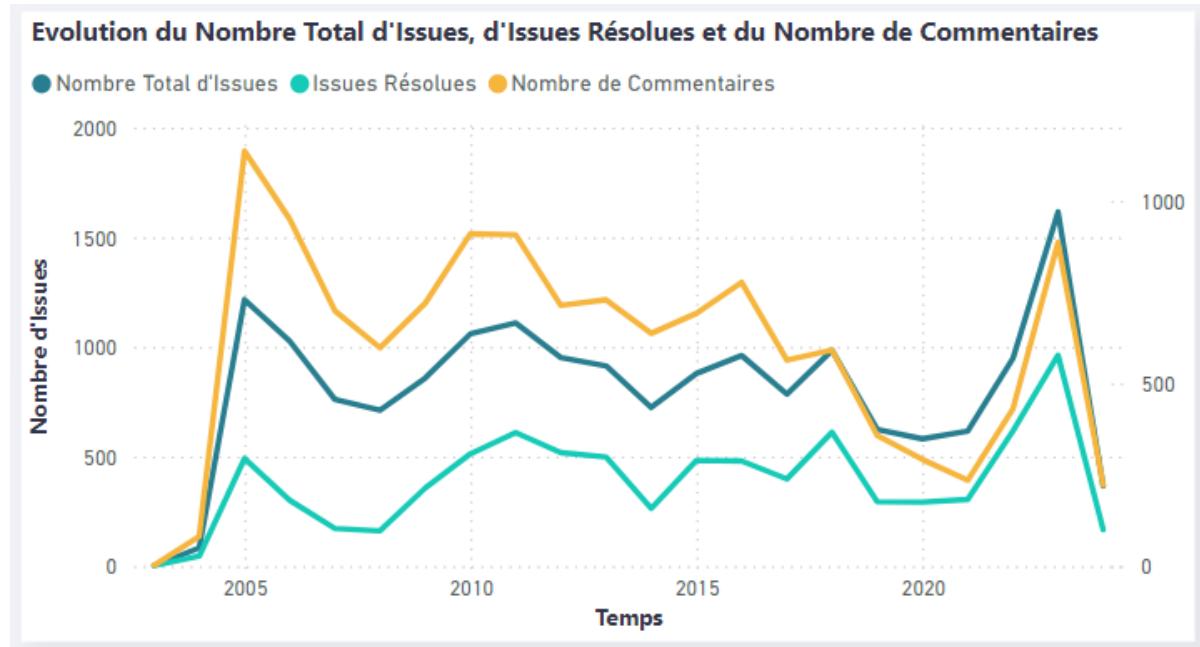


Figure 79 Évolution du Nombre Total d'Issues, d'Issues Résolues et du Nombre de Commentaires

Aperçu :

L'élément Évolution du Nombre Total d'Issues, d'Issues Résolues et du Nombre de Commentaires présente une analyse temporelle par année du nombre total d'issues, du nombre d'issues résolues, et du nombre de commentaires. Grâce à la fonctionnalité de drill down, il est possible d'explorer plus en détail : en cliquant sur une année, l'élément affiche l'évolution par mois de cette année, et en cliquant sur un mois, il montre l'évolution par jour de ce mois.



Figure 80 Evolution du Nombre Total d'Issue, d'Issues Résolues et du Nombre de Commentaire

L'élément Evolution du Nombre Total d'Issue, d'Issues Résolues et du Nombre de Commentaire offre une vue dynamique et détaillée, permettant de comprendre les variations temporelles et les tendances dans la gestion des issues et des commentaires.

### Conclusion :

L'analyse révèle que malgré une forte création d'issues dans certaines périodes (notamment autour de 2005 et 2020), le nombre d'issues résolues reste inférieur, ce qui souligne une accumulation de problèmes non résolus. Par ailleurs, le nombre de commentaires suit une dynamique proche du volume total d'issues, montrant une corrélation entre la collaboration autour des issues et leur croissance.

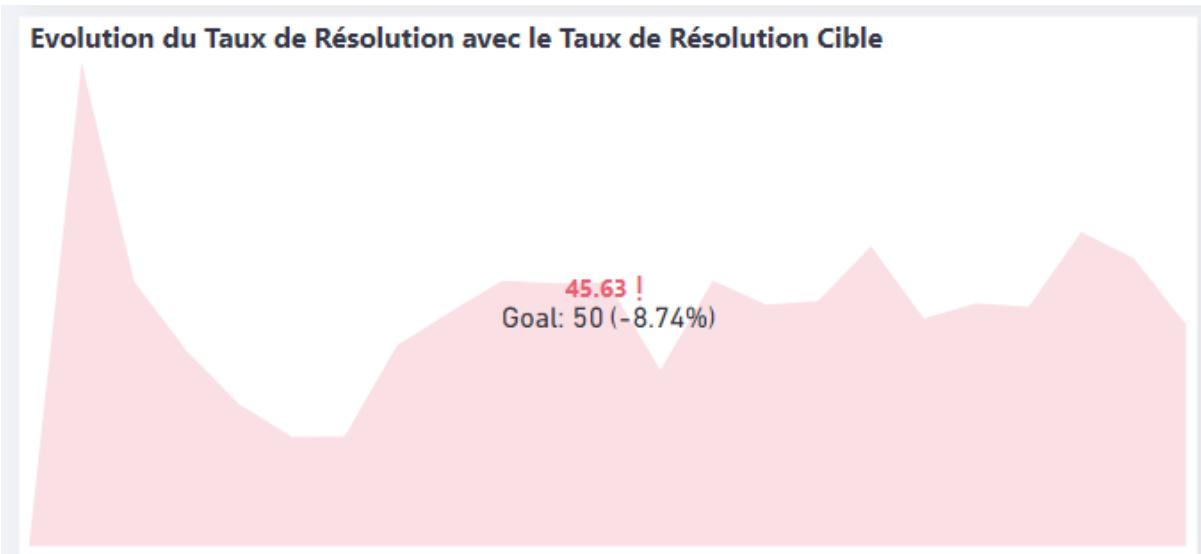


Figure 81 Évolution du Taux de Résolution avec le Taux de Résolution Cible

### Aperçu :

L'élément Évolution du Taux de Résolution avec le Taux de Résolution Cible est un KPI Chart qui affiche l'évolution du taux de résolution des issues sur une période donnée. Il permet de suivre la performance en matière de résolution tout en comparant ce taux à un taux de résolution cible fixé comme objectif à atteindre.

Voici la formule du KPI Taux de Résolution :

```

1 Resolution Rate (%) =
2 | DIVIDE([Resolved Issues], [Total Issues], 0) * 100
  
```

Figure 82 KPI Taux de Résolution

Cette formule calcule le taux de résolution en pourcentage en divisant le nombre d'issues résolues (Resolved Issues) par le nombre total d'issues (Total Issues).

### Conclusion :

L'analyse de l'élément Évolution du Taux de Résolution avec le Taux de Résolution Cible montre que le taux de résolution actuel est de 45.63%, ce qui reste inférieur à l'objectif fixé de 50%, avec un écart de -8.74%. Cette situation met en évidence un retard dans l'atteinte de la cible, nécessitant des efforts supplémentaires pour améliorer l'efficacité de la résolution des issues.

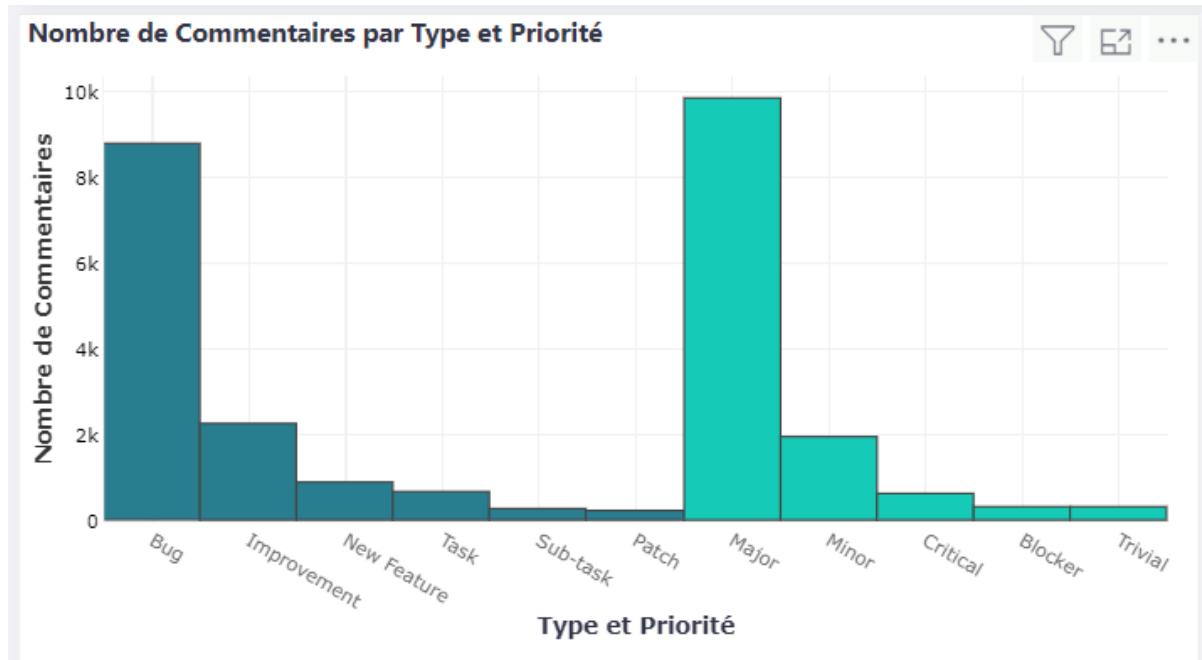


Figure 83 Nombre de Commentaires par Type et Priorité

#### Aperçu :

L'élément Nombre de Commentaires par Type et Priorité est représenté sous la forme d'un double histogramme. Il illustre la répartition du nombre de commentaires en fonction du type des issues ainsi que de leur priorité.

#### Conclusion :

L'analyse de l'élément Nombre de Commentaires par Type et Priorité montre que les issues de type Bug et celles avec la priorité Major accumulent le plus grand nombre de commentaires. Cela indique que ces types d'issues nécessitent davantage d'échanges et de suivi, ce qui témoigne de leur impact et de leur importance dans le projet. Les autres types d'issues, comme New Feature ou Task, ainsi que les priorités Critical et Minor, présentent des volumes de commentaires nettement inférieurs.

### 2.3. Dashboard Projet

Le Dashboard Projet fournit une analyse globale des projets en se basant sur les données relatives aux issues et aux commentaires. Il offre une vue d'ensemble permettant d'évaluer la performance des projets et de comprendre leur progression.

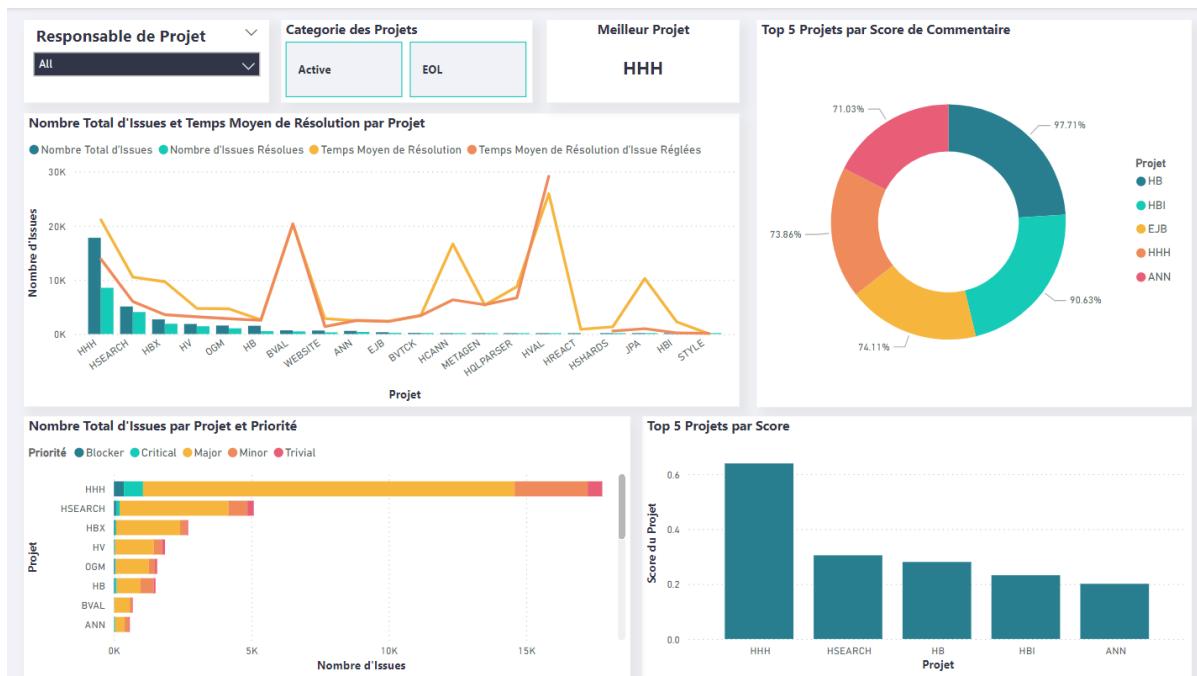


Figure 84 Dashboard Projet

L'élément Responsable du Projet permet de sélectionner les individus responsables des projets, ajustant ainsi le tableau de bord pour afficher uniquement les projets dont ils ont la charge.

Le KPI Meilleur Projet met en avant le projet ayant obtenu le meilleur Project Score, offrant une vue rapide sur les projets les plus performants.

Enfin, grâce à l'élément Catégorie des Projets, il est possible de filtrer les projets en fonction de leur statut, qu'ils soient Actifs ou EOL (End of Life).

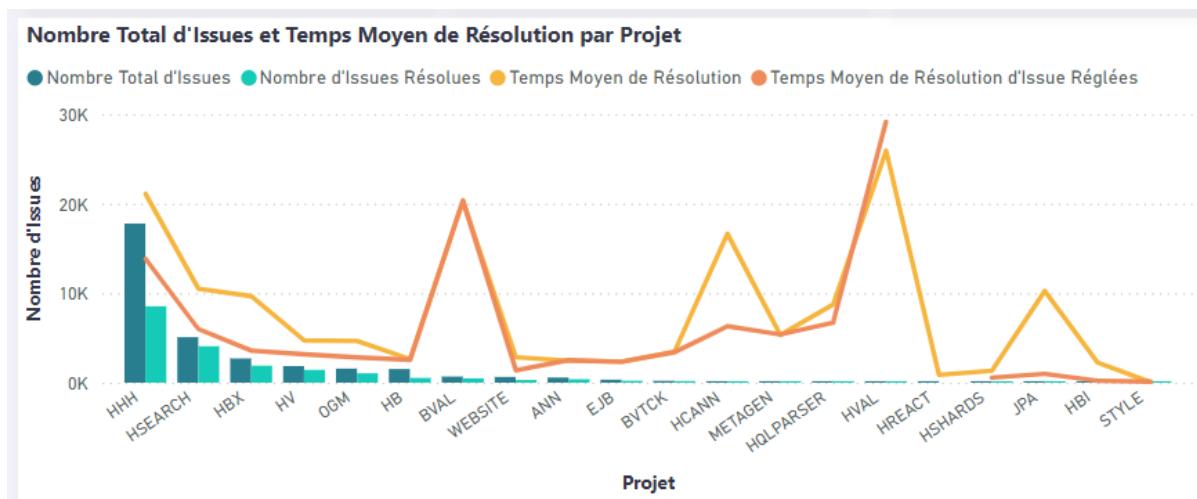


Figure 85 Nombre Total d'Issues et Temps Moyen de Résolution par Projet

Aperçu :

L'élément Nombre Total d'Issues et Temps Moyen de Résolution par Projet offre une représentation visuelle des projets en se basant sur deux indicateurs clés : le nombre total d'issues et Issue Résolues associées à chaque projet et leur temps moyen de résolution.

### Conclusion :

L'analyse de l'élément Nombre Total d'Issues et Temps Moyen de Résolution par Projet montre que les projets HHH et HSEARCH possèdent le plus grand nombre d'issues, bien que leur temps moyen de résolution reste modéré. En revanche, certains projets comme HVAL et JPA affichent un temps moyen de résolution très élevé malgré un nombre plus faible d'issues, indiquant des difficultés dans la résolution des problèmes pour ces projets.

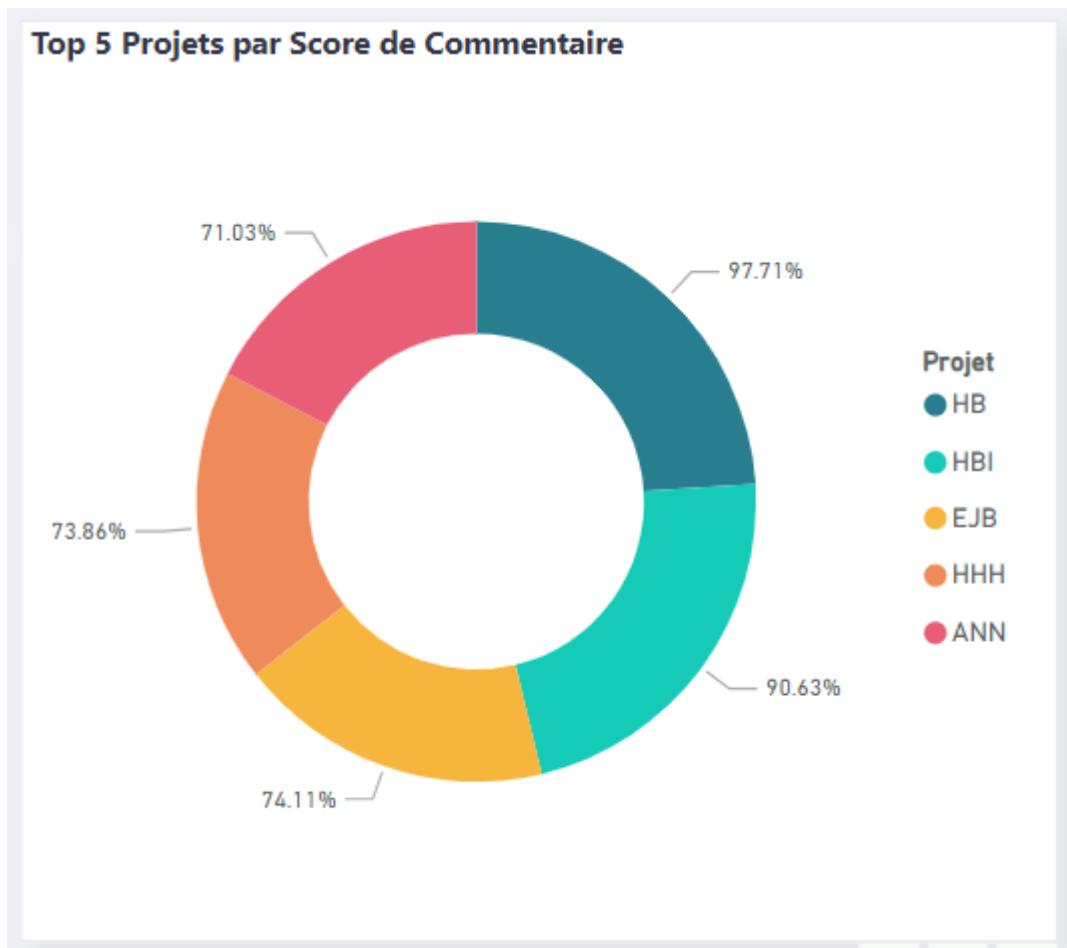


Figure 86 Top 5 Projets par Score de Commentaire

### Aperçu :

L'élément Top 5 Projets par Score de Commentaire est représenté sous forme de donut chart. Il illustre les cinq projets principaux en fonction du pourcentage d'issues ayant des commentaires.

Voici la formule du KPI Score de Commentaire :

```

1 Comment Score =
2   DIVIDE(
3     [Valid Comment Count],
4     COUNT('public DW_Issues'[Issue_Key]),
5     0
6   )

```

Figure 87 KPI Score de Commentaire

Cette formule calcule le Score de Commentaire en divisant le nombre de commentaires valides (Valid Comment Count) par le nombre total d'issues (COUNT('public DW\_Issues'[Issue\_Key])). La fonction DIVIDE garantit que la division est sécurisée en cas de valeur nulle, évitant ainsi les erreurs de calcul.

```

1 Valid Comment Count =
2   COUNTAX(
3     FILTER(
4       'public DW_Comments',
5       'public DW_Comments'[Comments] <> "none"
6     ),
7     'public DW_Comments'[Comment_Id]
8   )

```

Figure 88 Mesure Valid Comment Count

Cette mesure utilise la fonction COUNTAX pour compter les identifiants de commentaires (Comment\_Id) dans la table 'public DW\_Comments'. Elle filtre uniquement les commentaires dont la valeur n'est pas égale à "none", excluant ainsi les commentaires vides ou non pertinents.

### Conclusion :

L'analyse de l'élément Top 5 Commentaires par Score montre que le projet HB possède le meilleur score avec 97.71%, suivi de près par HBI avec 90.63%. Les projets EJB, HHH, et ANN affichent des scores respectifs de 74.11%, 73.86%, et 71.03%, indiquant une bonne interaction autour des issues, mais à un niveau légèrement inférieur.

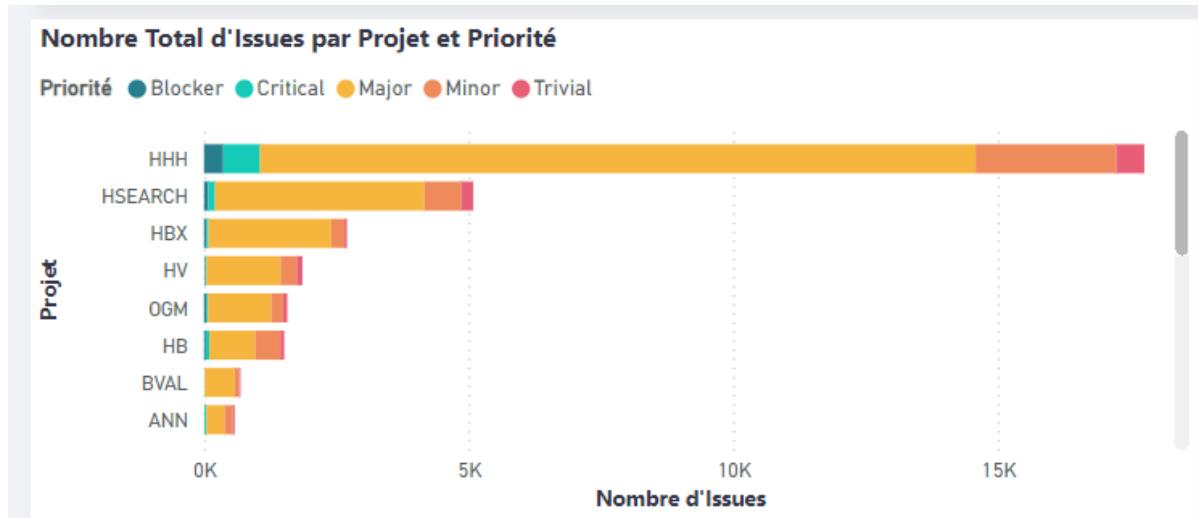


Figure 89 Nombre Total d'Issues par Projet et Priorité

#### Aperçu :

L'élément Nombre Total d'Issues par Projet et Priorité représente la répartition des projets en fonction du nombre total d'issues et de leur priorité.

#### Conclusion :

L'analyse de l'élément Nombre Total d'Issues par Projet et Priorité montre que le projet HHH possède le plus grand nombre d'issues, principalement de priorité Major. Le projet HSEARCH suit, avec une répartition plus équilibrée des priorités, incluant des issues de type Critical et Trivial. Les autres projets, comme HBX, HV, et OGM, présentent un volume d'issues beaucoup plus faible, mais avec une diversité notable dans les priorités.

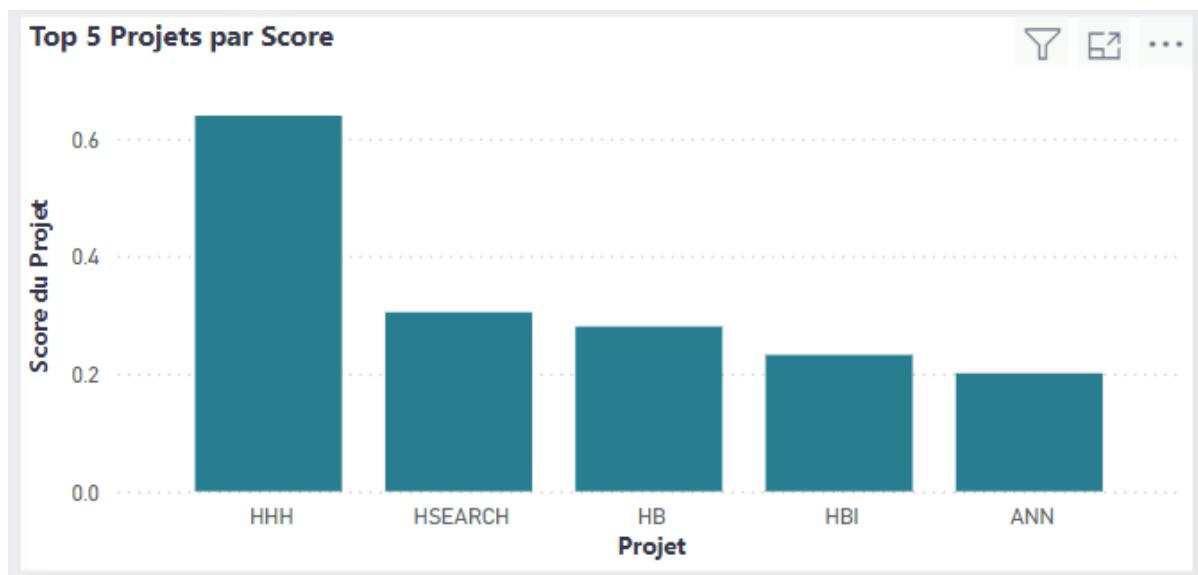


Figure 90 Top 5 Projets par Score

#### Aperçu :

L'élément Top 5 Projets par Score met en avant les cinq projets les plus performants en fonction de leur score.

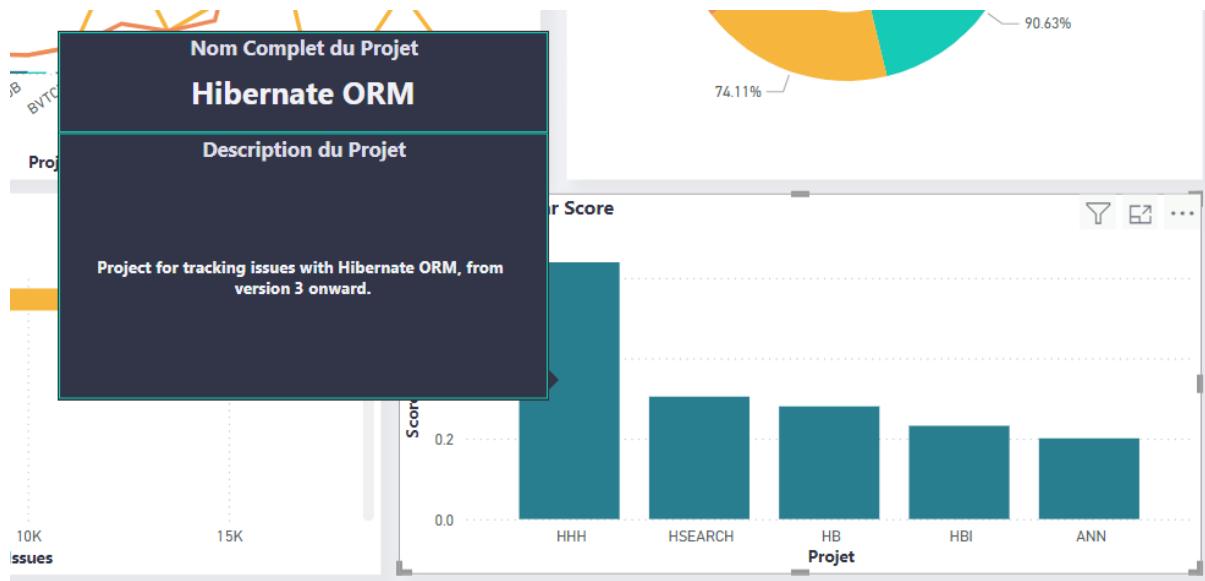


Figure 91 Tooltip Projet

En passant la souris sur un projet, des informations supplémentaires s'affichent, telles que son nom complet et sa description. Cette fonctionnalité interactive permet d'obtenir des détails contextuels sur chaque projet.

La formule du KPI Score du Projet est calculée comme suit :

```

1 Project Score =
2   0.10 * DIVIDE(
3     [Total Issues] - MINX(ALL('public DW_Projects'[Project_Key]), [Total Issues]),
4     MAXX(ALL('public DW_Projects'[Project_Key]), [Total Issues]) - MINX(ALL('public DW_Projects'[Project_Key]), [Total Issues]),
5     0
6   ) +
7   0.35 * DIVIDE(
8     [Resolved Issues] - MINX(ALL('public DW_Projects'[Project_Key]), [Resolved Issues]),
9     MAXX(ALL('public DW_Projects'[Project_Key]), [Resolved Issues]) - MINX(ALL('public DW_Projects'[Project_Key]), [Resolved Issues]),
10    0
11  ) +
12  0.25 * DIVIDE(
13    [Comment Score] - MINX(ALL('public DW_Projects'[Project_Key]), [Comment Score]),
14    MAXX(ALL('public DW_Projects'[Project_Key]), [Comment Score]) - MINX(ALL('public DW_Projects'[Project_Key]), [Comment Score]),
15    0
16  ) +
17  0.30 * DIVIDE(
18    1 / [Avg Resolution Time (Hours)] - MINX(ALL('public DW_Projects'[Project_Key]), 1 / [Avg Resolution Time (Hours)]),
19    MAXX(ALL('public DW_Projects'[Project_Key]), 1 / [Avg Resolution Time (Hours)]) - MINX(ALL('public DW_Projects'[Project_Key]), 1 / [Avg
Resolution Time (Hours)]),
20    0
21 )

```

Figure 92 KPI Score du Projet

La formule Project Score est conçue pour donner une évaluation équilibrée de la performance des projets. Elle prend en compte à la fois la quantité d'issues, la qualité de leur résolution, l'activité des commentaires, et la rapidité de traitement, en attribuant un poids significatif aux issues résolues et au temps de résolution. Cela permet d'identifier les projets les plus performants sur la base de critères objectifs et normalisés.

**Conclusion :**

L'analyse de l'élément Top 5 Projets par Score montre que le projet HHH possède le score le plus élevé, indiquant une performance supérieure en termes de résolution, d'activité des commentaires et de temps de traitement. Les projets HSEARCH, HB, HBI et ANN suivent avec des scores plus faibles, soulignant des performances globales moins optimales par rapport au projet HHH.

### 3. Application Python Elasticsearch

Pour l'application Python utilisant Elasticsearch, nous avons choisi d'effectuer la recherche en nous basant sur les champs description et résumé de l'issue. Un modèle SBERT pré-entraîné est utilisé pour générer des embeddings (représentations vectorielles) des descriptions et résumés saisis par l'utilisateur. Ces embeddings permettent ensuite d'identifier les issues ayant une description et un résumé similaires dans la base de données grâce à des recherches vectorielles optimisées dans Elasticsearch.

#### 3.1. Indexation des embeddings dans Elasticsearch

Les embeddings sont générés pour toutes les descriptions et résumés des issues présents dans les documents d'Elasticsearch, à l'aide du modèle SBERT. Chaque texte est transformé en vecteur numérique qui capture le sens et la sémantique de l'information.

Une fois les embeddings générés, ils sont indexés dans Elasticsearch en tant que documents. Chaque document inclut les vecteurs associés aux champs description et résumé, facilitant ainsi les recherches par similarité.

```

1 1 import warnings
2 2 import time
3 3 from elasticsearch import Elasticsearch, helpers
4 4 from sentence_transformers import SentenceTransformer
5
6 6 warnings.filterwarnings("ignore", category=UserWarning)
7
8 8 ELASTIC_PASSWORD = "elastic"
9 9 client = Elasticsearch([
10 10     "https://localhost:9200",
11 11     ca_certs=r"C:\ElasticStack\elasticsearch-8.14.1\config\certs\http_ca.crt",
12 12     basic_auth=("elastic", ELASTIC_PASSWORD)
13 13 ])
14
15 15 index_name = "pgindex"
16
17 17 model = SentenceTransformer(r'C:\Users\omark\Documents\GitHub\StagePFE\sbert-summary-description-model')
18
19 19 def get_sbert_embedding(text):
20 20     if not text:
21 21         return None
22 22     embedding = model.encode(text, convert_to_numpy=True)
23 23     return embedding.tolist()
24

```

Figure 93 Code d'indexation des embeddings 1

Ce code configure une connexion avec Elasticsearch et utilise SBERT (Sentence-BERT) pour générer des embeddings à partir de textes afin de les indexer pour des recherches basées sur la similarité.

**Fonction get\_sbert\_embedding :**

- Cette fonction prend un texte en entrée.
- Si le texte est vide, elle retourne None.
- Sinon, elle utilise le modèle SBERT pour générer un embedding sous forme de vecteur numpy, puis le convertit en une liste Python pour un stockage compatible avec Elasticsearch.

```
def fetch_all_issues(batch_size=100, retries=3, delay=5):
    search_query = {
        "query": {
            "match_all": {}
        }
    }
    for attempt in range(retries):
        try:
            return helpers.scan(client, query=search_query, index=index_name, size=batch_size, scroll="10m")
        except Elasticsearch.NotFoundError:
            print(f"Scroll context lost. Retrying... Attempt {attempt+1}/{retries}")
            time.sleep(delay)
    raise Exception("Failed to retrieve issues after multiple retries.")
```

Figure 94 Code d'indexation des embeddings 2

La fonction `fetch_all_issues` permet de récupérer toutes les issues indexées dans Elasticsearch par lot (batch) en utilisant la méthode de scrolling pour parcourir de grandes quantités de données.

```

def index_issues_with_sbert_embeddings(issues):
    actions = []

    for issue in issues:
        source = issue['_source']
        issue_id = issue['_id']

        summary_text = source.get('public_dw_issues_summary', "")
        description_text = source.get('public_dw_issues_description', "")
        combined_text = f"{summary_text} {description_text}".strip()

        combined_embedding = get_sbert_embedding(combined_text)

        source['sbert_embedding'] = combined_embedding

        action = {
            "_op_type": "index",
            "_index": index_name,
            "_id": issue_id,
            "_source": source
        }
        actions.append(action)

    helpers.bulk(client, actions)
    print(f"Indexed {len(actions)} documents with SBERT embeddings.")

issues = fetch_all_issues()
index_issues_with_sbert_embeddings(issues)

```

Figure 95 Code d'indexation des embeddings 3

Ce code récupère les documents existants depuis Elasticsearch, combine les champs summary et description pour chaque issue, et génère des embeddings SBERT à l'aide du modèle pré-entraîné. Les embeddings sont ajoutés aux documents sous le champ sbert\_embedding, puis réindexés en masse dans Elasticsearch grâce à la fonction helpers.bulk.

### 3.2. Application de recherche Elasticsearch

Une fois les documents indexés avec les embeddings SBERT, nous passons à la mise en place de l'application Python de recherche. Cette application utilise Elasticsearch pour effectuer des requêtes basées sur la similarité des embeddings. Lorsqu'un utilisateur saisit un texte (comme un résumé ou une description d'issue), le modèle SBERT génère son embedding, qui est ensuite comparé aux embeddings déjà indexés dans Elasticsearch. Cela permet de retrouver rapidement les issues ayant un contenu sémantiquement similaire, offrant ainsi une recherche avancée et pertinente pour l'utilisateur.

```

ELASTIC_PASSWORD = "elastic"
client = Elasticsearch(
    "https://localhost:9200",
    ca_certs=r"C:\ElasticStack\elasticsearch-8.14.1\config\certs\http_ca.crt",
    basic_auth=("elastic", ELASTIC_PASSWORD)
)

model = SentenceTransformer(r"C:\Users\omark\Documents\GitHub\StagePFE\sbert-summary-description-model")

def get_sbert_embedding(text):
    if not text:
        return None
    embedding = model.encode(text, convert_to_numpy=True)
    return embedding.tolist()

```

Figure 96 Code de recherche Elasticsearch 1

Ce code permet d'établir une connexion sécurisée à Elasticsearch et prépare l'environnement pour générer des embeddings SBERT qui serviront dans des recherches sémantiques avancées. La fonction `get_sbert_embedding` joue un rôle clé en transformant les textes en vecteurs pour faciliter leur indexation ou comparaison dans Elasticsearch.

```

def fetch_comments_for_issue(issue_key):
    search_query_comments = {
        "query": {
            "bool": {
                "must": [
                    {
                        "match_phrase": {
                            "public_dw_comments_issue_key": issue_key
                        }
                    }
                ]
            },
            "_source": ["public_dw_comments_comments"]
        }
    }

    try:
        response_comments = client.search(index="pgindex", body=search_query_comments)
        return [hit["_source"] for hit in response_comments["hits"]["hits"]]
    except Exception as e:
        print(f"Error searching comments for issue key {issue_key}: {e}")
        return []

```

Figure 97 Code de recherche Elasticsearch 2

La fonction `fetch_comments_for_issue` permet de rechercher et récupérer les commentaires associés à une issue spécifique dans Elasticsearch, en utilisant l'index pgindex.

#### Requête Elasticsearch (`search_query_comments`) :

- `query` : Une clause booléenne avec une condition `must` pour assurer que les commentaires recherchés correspondent exactement à l'`issue_key`.

- **match\_phrase** :
  - Cette méthode garantit que la valeur du champ public\_dw\_comments\_issue\_key correspond exactement à l'issue\_key spécifiée. Cela évite les correspondances partielles.
- **\_source** : Limite les champs récupérés aux seuls public\_dw\_comments\_comments pour réduire le volume de données retourné.

```
def search_issues_with_embedding(query_summary, query_description, project_key=None, issue_type=None, num_results=5):
    combined_text = f"{query_summary} {query_description}".strip()
    query_embedding = get_sbert_embedding(combined_text)

    if query_embedding is None:
        print("No valid embeddings found for the input query.")
        return

    search_query = {
        "_source": [
            "public_dw_issues_summary",
            "public_dw_issues_description",
            "public_dw_issues_issue_key",
            "public_dw_issues_type",
            "public_dw_issues_status",
            "public_dw_issues_assignee",
            "public_dw_issues_reporter",
            "public_dw_issues_created",
            "public_dw_issues_updated",
            "public_dw_issues_priority",
            "public_dw_issues_components",
            "public_dw_issues_resolution",
            "public_dw_issues_fix_versions",
            "public_dw_issues_affects_version",
            "public_dw_issues_status_category"],
        "size": num_results,
        "query": {
            "bool": {
                "must": [
                    {"match": {"public_dw_issues_project_key": project_key}}]
                if project_key else [],
                "should": [
                    {"knn": {"field": "sbert_embedding", "query_vector": query_embedding, "num_candidates": 2000}}]
            }
        }
    }
```

Figure 98 Code de recherche Elasticsearch 3

La fonction `search_issues_with_embedding` permet d'effectuer une recherche avancée dans Elasticsearch en utilisant une combinaison de filtres et d'une recherche KNN (k plus proches voisins) basée sur les embeddings SBERT.

Les paramètres `query_summary` et `query_description` sont combinés pour créer une chaîne de texte `combined_text`.

Le texte combiné est ensuite converti en embeddings à l'aide de la fonction `get_sbert_embedding`.

**Requête Elasticsearch (search\_query) :**

- **\_source** : Spécifie les champs à récupérer dans les résultats de la recherche, comme le résumé, description, type, priorité, etc. Cela optimise la taille des résultats en excluant les champs inutiles.
- **size** : Définit le nombre de résultats à retourner (par défaut 5, via num\_results).
- **query** :
  - **bool** : Combine plusieurs clauses pour affiner la recherche :
  - **must** : Ajoute un filtre pour le projet (public\_dw\_issues\_project\_key) si un project\_key est fourni.
  - **should** : Utilise une recherche KNN (k-nearest neighbors) pour retrouver les documents ayant des embeddings proches de l'embedding de la requête.
  - **knn** :
    - **field** : Spécifie le champ sbert\_embedding contenant les vecteurs indexés dans Elasticsearch.
    - **query\_vector** : Embedding généré pour la requête.
    - **num\_candidates** : Nombre de candidats à évaluer avant de retourner les meilleurs résultats. Pour les indexes ayant >100000 documents ce qui est notre cas, On doit choisir un nombre entre 2000 et 5000.

```

if project_key:
    search_query["query"]["bool"]["must"].append({
        "match": {
            "public_dw_issues_project_key": project_key
        }
    })

if issue_type:
    search_query["query"]["bool"]["should"].append({
        "match": {
            "public_dw_issues_type": issue_type
        }
    })

```

Figure 99 Code de recherche Elasticsearch 4

Ces lignes de code ajoutent des filtres supplémentaires dans la requête Elasticsearch pour affiner les résultats en fonction des paramètres facultatifs project\_key et issue\_type.

```

hits = response['hits']['hits']
if hits:
    for hit in hits:
        source = hit["_source"]
        issue_key = source.get('public_dw_issues_issue_key')

        comments = fetch_comments_for_issue(issue_key)

        print("===== Détails de l'issue =====")
        print(f"Clé de l'issue: {issue_key}")
        print(f"Statut: {source['public_dw_issues_status']}")
        print(f"Type: {source['public_dw_issues_type']}")
        print(f"Assigné: {source['public_dw_issues_assignee']}")
        print(f"Rapporteur: {source['public_dw_issues_reporter']}")
        print(f"Date de création: {source['public_dw_issues_created']}")
        print(f"Dernière mise à jour: {source['public_dw_issues_updated']}")
        print(f"Priorité: {source['public_dw_issues_priority']}")
        print("\n--- Informations complémentaires ---")
        print(f"Composants: {source['public_dw_issues_components']}")
        print(f"Résolution: {source['public_dw_issues_resolution']}")
        print(f"Versions corrigées: {source['public_dw_issues_fix_versions']}")
        print(f"Versions affectées: {source['public_dw_issues_affects_version']}")
        print(f"Catégorie de statut: {source['public_dw_issues_status_category']}")
        print("\n--- Résumé et Description ---")
        print(f"Résumé: {source['public_dw_issues_summary']}")
        print(f"Description:\n{source['public_dw_issues_description']}")

    if comments:
        print("\n--- Commentaires de l'issue ---")
        for comment in comments:
            comment_content = comment["public_dw_comments_comments"]
            if comment_content.lower() != "none":
                print(f"Contenu du commentaire:\n{comment_content.strip()}")
        if all(comment["public_dw_comments_comments"].lower() == "none" for comment in comments):
            print("Pas de commentaire dans cette issue")
        else:
            print("Pas de commentaire dans cette issue")

```

Figure 100 Code de recherche Elasticsearch 5

Enfin, Ce code récupère les résultats de la recherche Elasticsearch, extrait les informations détaillées de chaque issue, puis affiche ces informations dans un format clair et organisé. Si des commentaires sont associés à une issue, ils sont également affichés.

## 4. Conclusion

Dans ce chapitre, nous avons présenté les différents tableaux de bord Power BI pour l'analyse visuelle des données, ainsi que le code Python permettant d'interroger Elasticsearch afin d'effectuer des recherches avancées. Cette combinaison d'outils offre une solution complète pour visualiser et explorer les données de manière efficace.

Nous allons maintenant passer au dernier chapitre, où nous aborderons l'optimisation de l'application Elasticsearch ainsi que la création du front-end.

## Chapitre 6 : Maintenance et optimisation

1.	Introduction .....	104
2.	Fine tuning du modèle.....	104
2.1.	Préparation des données .....	104
2.2.	Training du modèle .....	107
2.3.	Evaluation du modèle.....	113
3.	Front end du projet .....	116
3.1.	Flask-Python.....	117
3.2.	Plateforme front end.....	117
4.	Conclusion.....	119

## 1. Introduction

Dans ce chapitre, nous allons aborder deux aspects essentiels pour finaliser et améliorer notre projet. Tout d'abord, nous présenterons le fine-tuning du modèle SBERT, une étape cruciale pour améliorer les performances de la recherche sémantique en ajustant le modèle aux spécificités de nos données. Ce fine-tuning permettra d'obtenir des résultats plus pertinents et mieux adaptés aux besoins du projet.

Ensuite, nous décrirons le développement du front-end du projet, une interface utilisateur intégrant la fonctionnalité de recherche avancée basée sur Elasticsearch ainsi que les tableaux de bord Power BI. Cette interface facilitera la visualisation des données et l'interaction avec les résultats de recherche.

## 2. Fine tuning du modèle

Pour améliorer la performance de la recherche sémantique, nous avons choisi d'utiliser la perte contrastive comme technique de fine-tuning du modèle SBERT. Cette méthode permet d'ajuster le modèle en rapprochant les représentations vectorielles (embeddings) des textes similaires tout en éloignant celles des textes différents.

### 2.1. Préparation des données

La première étape consiste à préparer et configurer les données nécessaires pour le fine-tuning :

- Paires positives : Textes ayant un sens similaire, comme les résumés et descriptions d'issues proches ou liés.
- Paires négatives : Textes ayant un sens différent, permettant au modèle de mieux distinguer les contenus non pertinents.

Ces données seront organisées sous forme de paires d'exemples (positifs et négatifs), ce qui servira d'entrée pour l'entraînement avec la perte contrastive. Une fois les données configurées, elles seront utilisées pour ajuster les poids du modèle SBERT, afin de maximiser la précision des recherches par similarité dans Elasticsearch.

```

print("Loading dataset...")
data = pd.read_excel("/content/drive/MyDrive/jira_fields_repaired.xlsx")

data["Summary"] = data["Summary"].fillna("").astype(str)
data = data[data["Summary"] != ""].reset_index(drop=True)

print("Generating summary embeddings...")
model = SentenceTransformer("all-MiniLM-L6-v2")
summary_embeddings = model.encode(data["Summary"].tolist(), batch_size=32)

print("Computing cosine similarity matrix...")
cosine_sim_matrix = cosine_similarity(summary_embeddings)

print("Generating positive pairs...")
positive_pairs = []
top_n_similar = 5

```

*Figure 101 Code de préparation de la data 1*

Ce code prépare un jeu de données pour le fine-tuning d'un modèle SBERT en utilisant la perte contrastive. Il génère des paires positives et négatives de textes basées sur les similarités cosinus calculées entre les résumés d'issues.

La similarité cosinus est calculée entre tous les embeddings pour mesurer leur degré de similitude.

La matrice obtenue donne une vue d'ensemble des textes les plus proches les uns des autres.

```

print("Generating positive pairs...")
positive_pairs = []
top_n_similar = 5

for i, row in data.iterrows():
    similar_indices = np.argsort(-cosine_sim_matrix[i])
    for idx in similar_indices[1:top_n_similar + 1]:
        positive_pairs.append({
            "text1": row["Summary"],
            "text2": data.iloc[idx]["Summary"],
            "label": 1
        })

print("Generating negative pairs...")
negative_pairs = []
num_positive_pairs = len(positive_pairs)
all_indices = list(range(len(data)))

for i, row in data.iterrows():
    dissimilar_indices = np.argsort(cosine_sim_matrix[i])
    dissimilar_indices = [idx for idx in dissimilar_indices if idx != i]

    sampled_negative_indices = np.random.choice(
        dissimilar_indices, size=min(top_n_similar, len(dissimilar_indices)), replace=False
    )
    for idx in sampled_negative_indices:
        negative_pairs.append({
            "text1": row["Summary"],
            "text2": data.iloc[idx]["Summary"],
            "label": 0
        })

negative_pairs = negative_pairs[:num_positive_pairs]

```

Figure 102 Code de préparation de la data 2

### Génération des paires positives :

- Pour chaque résumé, les 5 résumés les plus similaires (en excluant lui-même) sont sélectionnés.
- Ces paires sont marquées avec un label 1 (paires positives).

### Génération des paires négatives :

- Pour chaque résumé, les résumés les moins similaires sont sélectionnés.
- 5 exemples dissemblables sont choisis aléatoirement pour chaque résumé.
- Ces paires sont marquées avec un label 0 (paires négatives).
- Le nombre de paires négatives est équilibré avec le nombre de paires positives.

### Résultat final :

- Un jeu de données équilibré contenant :
  - Des paires positives (textes similaires).

- Des paires négatives (textes dissimilaires).
- Ce jeu de données est prêt pour l'entraînement d'un modèle SBERT avec la perte contrastive, permettant au modèle d'apprendre à rapprocher les embeddings des textes similaires et à éloigner ceux des textes différents.

	A	B	C
	text1	text2	label
1	text1		
2	Micro optimisations for MappingMetamodelImpl.getEntityDes	mapped composite is is always assumed as transient	0
3	Switch project license to Apache License 2.0	Use correct license for final release	1
4	When 2LC enabled, flush session and then refresh entity cat in Gradle build, better account for non-standard local maven repo cache	0	0
5	Remove optimisation HCANN-77 which was specific for Java Remove remaining pre-Java-8 code		1

Figure 103 Excel du jeu de données

## 2.2. Training du modèle

Après avoir préparé les paires de textes (positives et négatives) avec leurs labels dans le fichier summary\_summary\_pairs\_balanced.csv, nous allons entraîner notre modèle SBERT en utilisant ces données pour la perte contrastive.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sentence_transformers import SentenceTransformer, InputExample, losses, evaluation
from torch.utils.data import DataLoader

print("Loading and preprocessing the dataset...")
data = pd.read_csv("/content/drive/MyDrive/summary_summary_pairs_balanced.csv")

data = data.dropna(subset=["text1", "text2", "label"]).reset_index(drop=True)

print("Splitting the dataset into train, validation, and test sets...")
train_data, temp_data = train_test_split(data, test_size=0.2, random_state=42)
val_data, test_data = train_test_split(temp_data, test_size=0.5, random_state=42)
```

Figure 104 Code de l'entraînement du modèle 1

Pour entraîner notre modèle SBERT en utilisant la perte contrastive, nous avons utilisé la librairie Sentence Transformers, une solution adaptée pour générer et affiner des embeddings sémantiques.

### Division du jeu de données :

- Le jeu de données est divisé en 3 ensembles :
  - Ensemble d'entraînement (80%) : Utilisé pour ajuster les poids du modèle.
  - Ensemble de validation (10%) : Permet de surveiller les performances du modèle pendant l'entraînement et d'éviter le surapprentissage (overfitting).
  - Ensemble de test (10%) : Utilisé pour évaluer les performances finales du modèle après l'entraînement.
- Cette répartition est réalisée avec la fonction train\_test\_split de scikit-learn pour garantir une séparation aléatoire mais reproductible (via random\_state=42).

```

def create_input_examples(data):
    return [
        InputExample(texts=[row["text1"], row["text2"]], label=float(row["label"]))
        for _, row in data.iterrows()
    ]

train_examples = create_input_examples(train_data)
val_sentences1 = val_data["text1"].tolist()
val_sentences2 = val_data["text2"].tolist()
val_scores = val_data["label"].astype(float).tolist()

train_dataloader = DataLoader(train_examples, shuffle=True, batch_size=32)

print("Loading SBERT model...")
model = SentenceTransformer("all-MiniLM-L6-v2")
train_loss = losses.CosineSimilarityLoss(model=model)

evaluator = evaluation.EmbeddingSimilarityEvaluator(
    sentences1=val_sentences1,
    sentences2=val_sentences2,
    scores=val_scores,
    name="validation_set_evaluator"
)

```

Figure 105 Code de l'entraînement du modèle 2

Cette partie du code prépare les données d'entraînement et de validation pour le fine-tuning du modèle SBERT en utilisant la perte de similarité cosinus.

### **Création des exemples d'entrée pour l'entraînement :**

- La fonction `create_input_examples` convertit les paires de textes (`text1` et `text2`) et leurs labels (`label`) en objets `InputExample` compatibles avec `Sentence Transformers`.
- Chaque exemple comprend deux textes (`texts=[...]`) et une étiquette numérique représentant leur similarité (`label=float(label)`).
- Le jeu de données d'entraînement est ensuite transformé en une liste d'exemples.

### **Préparation des données de validation :**

- Les phrases du jeu de validation sont extraites en deux listes distinctes :
  - `val_sentences1` : Contient les premiers textes des paires de validation.
  - `val_sentences2` : Contient les seconds textes des paires de validation.
- `val_scores` : Les labels (similarité) sont convertis en float.

### **Définition de la fonction de perte :**

- La fonction CosineSimilarityLoss est utilisée comme fonction de perte. Elle ajuste les poids du modèle pour maximiser la similarité cosinus entre les embeddings des paires positives et minimiser celle des paires négatives.

#### Définition de l'évaluateur :

- EmbeddingSimilarityEvaluator est configuré pour évaluer les performances du modèle sur le jeu de validation après chaque époque.
- Il compare les embeddings générés pour les deux listes de phrases (val\_sentences1 et val\_sentences2) et mesure leur similarité en fonction des scores fournis (val\_scores).

```

print("Training the model...")
output_path = "/content/drive/MyDrive/sbert-summary-description-model"
model.fit(
    train_objectives=[(train_dataloader, train_loss)],
    evaluator=evaluator,
    epochs=3,
    warmup_steps=100,
    evaluation_steps=1000,
    output_path=output_path,
    save_best_model=True,
    use_amp=True
)

print("Model training completed!")

print("Evaluating the model on the test set...")
test_sentences1 = test_data["text1"].tolist()
test_sentences2 = test_data["text2"].tolist()
test_scores = test_data["label"].astype(float).tolist()

test_evaluator = evaluation.EmbeddingSimilarityEvaluator(
    sentences1=test_sentences1,
    sentences2=test_sentences2,
    scores=test_scores,
    name="test_set_evaluator"
)

test_evaluator(model)

print("Evaluation on the test set completed!")

print(f"Model saved successfully to Google Drive at '{output_path}'!")

```

Figure 106 Code de l'entraînement du modèle 3

Cette dernière partie finalise le fine-tuning du modèle SBERT en l'entraînant, en l'évaluant sur l'ensemble de test, puis en sauvegardant le modèle optimisé.

### Hyperparamètres :

- train\_objectives : Le DataLoader d'entraînement (train\_dataloader) est associé à la fonction de perte train\_loss pour entraîner le modèle.
- evaluator : L'évaluateur de validation est utilisé pour suivre les performances du modèle après un certain nombre d'étapes.
- epochs=3 : L'entraînement s'effectue sur 3 époques (passages complets sur les données).
- warmup\_steps=100 : Définit un nombre d'étapes de chauffe pour éviter un démarrage brusque du taux d'apprentissage.
- evaluation\_steps=1000 : Le modèle est évalué toutes les 1000 étapes pour suivre ses performances.
- output\_path : Chemin où le modèle entraîné sera sauvegardé.
- save\_best\_model=True : Sauvegarde automatiquement le meilleur modèle basé sur les performances sur le jeu de validation.
- use\_amp=True : Active le calcul mixte de précision (AMP) pour accélérer l'entraînement tout en réduisant l'utilisation mémoire.
- Learning rate : Dans ce processus d'entraînement, le learning rate n'est pas spécifié explicitement. En effet, la bibliothèque Sentence Transformers utilise par défaut un learning rate scheduler pour gérer automatiquement l'évolution du taux d'apprentissage pendant l'entraînement.

### Évaluation finale sur l'ensemble de test :

EmbeddingSimilarityEvaluator compare les embeddings des textes de test pour mesurer leur similarité par rapport aux scores réels fournis.

Cela permet d'évaluer les performances finales du modèle sur des données **inédites**.

Step	Training Loss	Validation Loss	Validation Set Evaluator Pearson Cosine	Validation Set Evaluator Spearman Cosine
1000	0.056300	No log	0.918329	0.859103
2000	0.050300	No log	0.922372	0.858988
3000	0.048400	No log	0.926709	0.859808
4000	0.046500	No log	0.925774	0.858915
5000	0.045500	No log	0.925909	0.857932
6000	0.045200	No log	0.929006	0.859216
7000	0.043700	No log	0.930414	0.859694
8000	0.043000	No log	0.931837	0.859773
8411	0.043000	No log	0.932359	0.859984
9000	0.042000	No log	0.932205	0.859461
10000	0.040200	No log	0.932897	0.859926

Figure 107 Résultat de l'entraînement

## Métriques de l'entraînement :

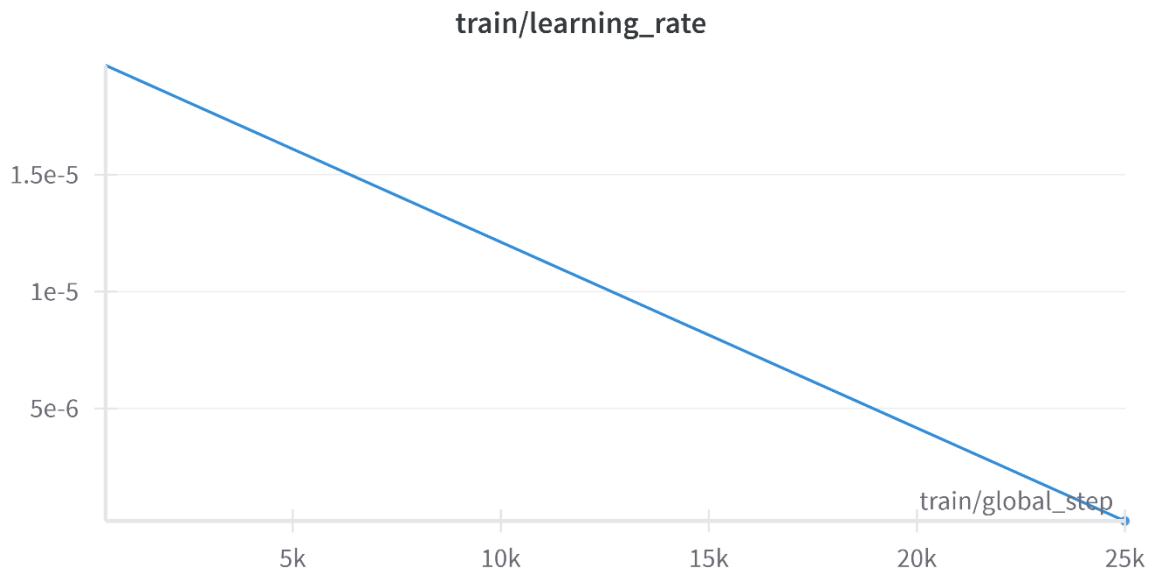


Figure 108 Métrique train/learning\_rate

### train/learning\_rate

- Signification : Le taux d'apprentissage utilisé au cours de l'entraînement. Il diminue linéairement grâce à un scheduler.
- Interprétation : La réduction progressive du taux d'apprentissage aide le modèle à converger en ajustant plus finement ses poids dans les dernières étapes sans modifications drastiques.



Figure 109 Métrique train/loss

## train/loss

- Signification : La perte calculée pendant l'entraînement.
- Interprétation : Une diminution constante de la perte indique que le modèle apprend correctement. Ici, la perte diminue de manière régulière, ce qui est un bon signe.

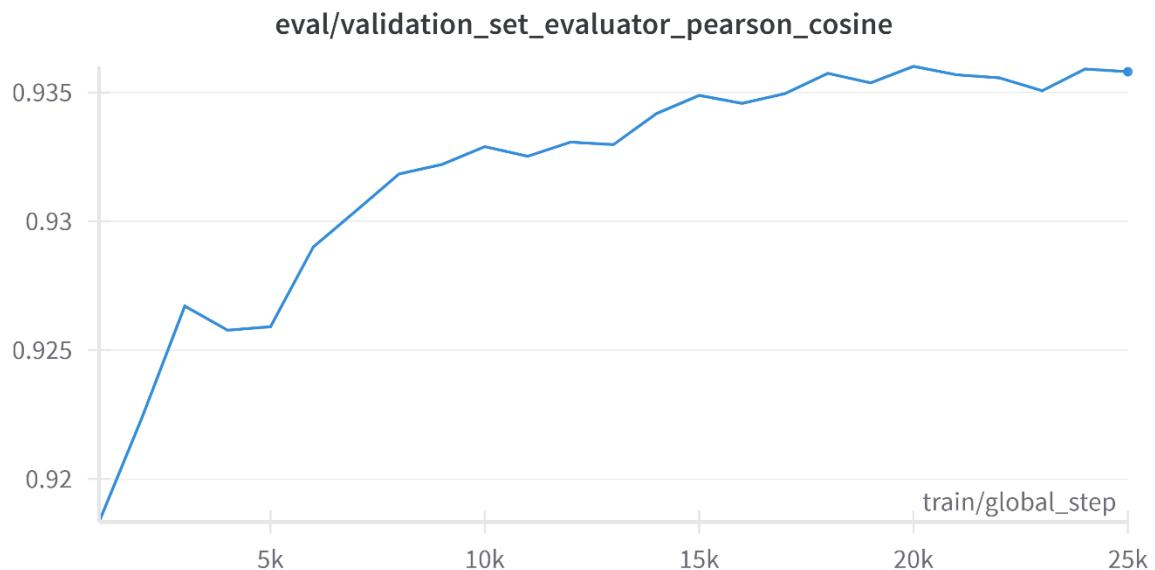


Figure 110 Métrique eval/validation\_set\_evaluator\_pearson\_cosine

## eval/validation\_set\_evaluator\_pearson\_cosine

- Signification : La corrélation de Pearson entre les similarités des embeddings prédicts et les scores réels sur l'ensemble de validation.
- Interprétation : Des valeurs proches de 1 indiquent que les embeddings du modèle s'alignent bien avec les similarités de vérité terrain.

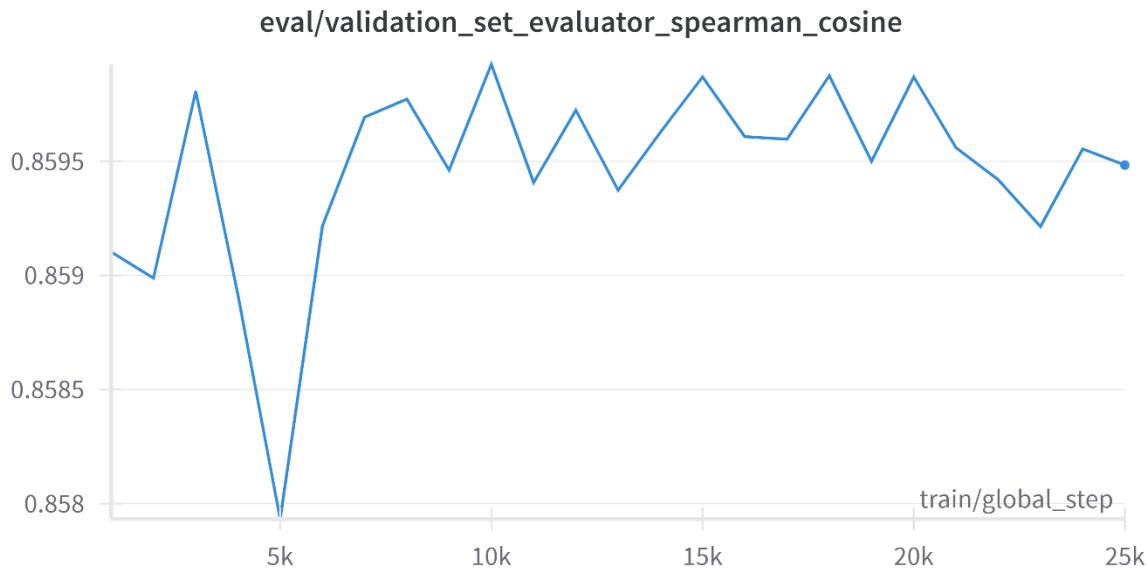


Figure 111 Métrique eval/validation\_set\_evaluator\_spearman\_cosine

### eval/validation\_set\_evaluator\_spearman\_cosine

- Signification : La corrélation de Spearman entre les similarités des embeddings prédicts et les scores réels.
- Interprétation : Spearman évalue la corrélation de rang, en se concentrant sur l'ordre relatif plutôt que sur les valeurs exactes.

Ces métriques indiquent que le modèle a été correctement entraîné, avec une diminution régulière de la perte et des corrélations élevées sur les ensembles de validation, confirmant la stabilité et l'efficacité du fine-tuning. Nous allons maintenant passer à l'évaluation finale du modèle pour analyser ses performances sur l'ensemble de test.

## 2.3. Evaluation du modèle

Pour évaluer les performances du modèle fine-tuné, nous avons suivi une approche structurée :

### Création des ensembles d'évaluation :

- Fields Set : Contient trois issues sélectionnées manuellement, représentant des exemples de référence pour l'évaluation.
- Reference Set : Contient un grand ensemble d'issues, incluant des issues similaires en contexte (créées pour servir de ground truth) et un mélange d'autres issues pour simuler des données réelles.

### Comparaison des modèles :

- Un code a été conçu pour comparer les performances du modèle pré-entraîné (modèle de base) et du modèle fine-tuné.

- Cette comparaison vise à analyser la capacité des modèles à identifier correctement les issues similaires dans le Reference Set en se basant sur les embeddings générés.

```

print("Loading the data...")
reference_data = pd.read_excel("/content/reference_set.xlsx")
query_data = pd.read_excel("/content/fields_set.xlsx")

reference_data["combined_text"] = reference_data["Summary"].fillna("") + " " + reference_data["Description"].fillna("")
query_data["combined_text"] = query_data["Summary"].fillna("") + " " + query_data["Description"].fillna("")

print("Generating embeddings...")
base_model = SentenceTransformer("all-MiniLM-L6-v2")
fine_tuned_model = SentenceTransformer("/content/drive/MyDrive/sbert-summary-description-model")

reference_embeddings_base = np.array(base_model.encode(reference_data["combined_text"].tolist(), batch_size=16))
reference_embeddings_fine_tuned = np.array(fine_tuned_model.encode(reference_data["combined_text"].tolist(), batch_size=16))
query_embeddings_base = np.array(base_model.encode(query_data["combined_text"].tolist(), batch_size=16))
query_embeddings_fine_tuned = np.array(fine_tuned_model.encode(query_data["combined_text"].tolist(), batch_size=16))

print("Performing KNN search...")
knn_base = NearestNeighbors(n_neighbors=4, metric="cosine")
knn_fine_tuned = NearestNeighbors(n_neighbors=4, metric="cosine")

knn_base.fit(reference_embeddings_base)
knn_fine_tuned.fit(reference_embeddings_fine_tuned)

distances_base, indices_base = knn_base.kneighbors(query_embeddings_base)
distances_fine_tuned, indices_fine_tuned = knn_fine_tuned.kneighbors(query_embeddings_fine_tuned)

reference_keys = reference_data["Issue Key"].tolist()

results_base = [
    [reference_keys[idx] for idx in neighbor_indices] for neighbor_indices in indices_base
]
results_fine_tuned = [
    [reference_keys[idx] for idx in neighbor_indices] for neighbor_indices in indices_fine_tuned
]

```

Figure 112 Code de l'évaluation du modèle 1

### Génération des embeddings :

Les embeddings (représentations vectorielles) sont générés pour les issues dans les ensembles de référence et de requêtes pour les deux modèles.

### Recherche KNN :

- n\_neighbors=4 : Retourne les 4 issues les plus proches pour chaque requête.
- Les indices des voisins trouvés pour chaque requête sont stockés dans indices\_base et indices\_fine\_tuned.

```

print("Example Query Comparison:")
for i, query in query_data.iterrows():
    print(f"Query {i+1}: {query['combined_text']}")
    print(f"Base Model Results: {results_base[i]}")
    print(f"Fine-Tuned Model Results: {results_fine_tuned[i]}")
    print("-" * 40)
    if i == 2:
        break

print("Evaluating models...")
ground_truth = {
    "HHH-18920": ["HHH-17485", "HHH-11", "HHH-12", "HHH-17487"],
    "HHH-18919": ["HHH-17412", "HHH-21", "HHH-22", "HHH-16898"],
    "HHH-18918": ["HHH-17315", "HHH-31", "HHH-32", "HHH-15758"],
}

def precision_at_k(retrieved, relevant, k=5):
    retrieved_k = retrieved[:k]
    relevant_set = set(relevant)
    retrieved_set = set(retrieved_k)
    return len(retrieved_set & relevant_set) / len(retrieved_set)

precision_base = []
precision_fine_tuned = []

for i, query in query_data.iterrows():
    query_key = query["Issue Key"]
    relevant_keys = ground_truth.get(query_key, [])

    precision_base.append(precision_at_k(results_base[i], relevant_keys, k=5))
    precision_fine_tuned.append(precision_at_k(results_fine_tuned[i], relevant_keys, k=5))

print(f"Average Precision@5 for Base Model: {np.mean(precision_base)}")
print(f"Average Precision@5 for Fine-Tuned Model: {np.mean(precision_fine_tuned)}")

```

Figure 113 Code de l'évaluation du modèle 2

### Évaluation des performances avec la ground truth :

- Ground truth : Définit, pour chaque requête, les issues réellement pertinentes (similaires).
- Precision@k : Mesure la proportion des résultats retournés qui sont pertinents dans les k premiers résultats.

Pour chaque requête, le code calcule Precision@5 pour les modèles de base et fine-tuné, puis calcule la moyenne pour chaque modèle.

Les moyennes des Precision@5 sont affichées pour les deux modèles.

Une précision plus élevée pour le modèle fine-tuné indique une meilleure capacité à retrouver des issues pertinentes.

### Résultat de l'évaluation des modèles :

```

Loading the data...
Generating embeddings...
Performing KNN search...
Example Query Comparison:
Query 1: Inconsistent hashCode generation across implementations. The issue originates from a bug in NHibernate.
Bug report: NHibernate Issue
Commit: NHibernate Commit
The problematic code in NHibernate was fixed by checking the last token before appending a dot.
NHibernate Test Case.
Base Model Results: ['HHH-12', 'HHH-11', 'HHH-17485', 'HHH-17487']
Fine-Tuned Model Results: ['HHH-12', 'HHH-17485', 'HHH-11', 'HHH-17487']
-----
Query 2: Unexpected method resolution causes type mismatch in Java compilation. This issue highlights a bug in JpaPredicate.equalTo(Expression<T> that);
JpaPredicate.equalTo(T that);
When executing the code:

java
Copy code
final JpaParameterExpression<Contact.Gender> parameter = cb.parameter(Contact.Gender.class);
root.get("gender").equalTo(parameter);
The compiler unexpectedly chooses equalTo(T) instead of equalTo(Expression<T>). To resolve this issue, use the generic version.
Base Model Results: ['HHH-17412', 'HHH-22', 'HHH-21', 'HHH-12760']
Fine-Tuned Model Results: ['HHH-17412', 'HHH-21', 'HHH-22', 'HHH-16898']
-----
Query 3: Limit Oracle database user privileges to prevent schema conflicts. Current Oracle database has the RESOURCE role.
CREATE VIEW
CREATE SESSION
CREATE SYNONYM
CREATE ANY INDEX (for XMLType tables)
EXECUTE ANY TYPE (avoids grant issues post-creation)
For Oracle 23c, the DB_DEVELOPER_ROLE is suggested as an ideal alternative, offering sufficient privileges.
Base Model Results: ['HHH-17315', 'HHH-31', 'HHH-32', nan]
Fine-Tuned Model Results: ['HHH-17315', 'HHH-31', 'HHH-32', nan]
-----
Evaluating models...
Average Precision@5 for Base Model: 0.8333333333333334
Average Precision@5 for Fine-Tuned Model: 0.9166666666666666

```

Figure 114 Résultat de l'évaluation

Le fine-tuning du modèle SBERT a permis d'améliorer considérablement ses performances pour la recherche sémantique. Avec une précision moyenne de 92% pour le modèle ajusté, cela montre que le modèle fine-tuné est mieux adapté pour identifier les issues similaires dans un contexte spécifique. Ce résultat valide l'efficacité du fine-tuning dans ce projet.

### 3. Front end du projet

Pour le front-end, nous avons conçu un site web utilisant HTML, CSS, et JavaScript. Ce site offre une interface utilisateur intuitive et conviviale, intégrant deux fonctionnalités principales :

#### Application de recherche avancée :

- L'application de recherche Python, développée avec Flask, a été intégrée au site pour permettre des recherches sémantiques.
- Les utilisateurs peuvent saisir des résumés ou descriptions, et le site affiche les issues similaires grâce à l'interaction avec Elasticsearch.

## Intégration des tableaux de bord Power BI :

- Les tableaux de bord Power BI, conçus pour analyser les issues et les projets, ont été intégrés au site pour permettre une visualisation directe des données.
- Cela permet aux utilisateurs d'accéder facilement aux insights clés sans quitter l'interface.

Ce site web centralise ainsi la recherche avancée et la visualisation des données.

### 3.1. Flask-Python

```
@app.route('/search', methods=['POST'])
def search_issues():
    data = request.json
    query_summary = data.get('query_summary', '')
    query_description = data.get('query_description', '')
    project_key = data.get('project_key', None)
    issue_type = data.get('issue_type', None)
```

Figure 115 Code Flask-Python

Pour intégrer l'application de recherche sémantique Python au front-end, nous avons utilisé la méthode POST via l'API de Flask. Cette méthode permet de transmettre des données saisies par l'utilisateur depuis l'interface web vers le backend pour effectuer la recherche dans Elasticsearch.

### 3.2. Plateforme front end

Voici donc notre Front End :

#### Aperçu du Projet :

Le site présente tout d'abord un aperçu du projet, offrant une explication claire et concise des différentes fonctionnalités disponibles.



Figure 116 Front end : Aperçu du projet

### Recherche Elasticsearch :

Ensuite, la plateforme propose une fonctionnalité de recherche sémantique avancée.

Résumé de l'Issue:	Description de l'Issue:
NodeName value not set in HAN, causing an NPE in DOM4J mode	The EntityManager.DOM4J cannot be used with business entities that rely solely on an

Clé du Projet:	Type de l'Issue:
ANN	Entrez le type de l'Issue (optionnel)

Recherche terminée avec succès !

- Top 1 Issue
- Top 2 Issue
- Top 3 Issue
- Top 4 Issue

Figure 117 Front end : Recherche Elasticsearch

En saisissant un résumé et une description d'une issue dans l'interface de recherche, l'application envoie ces informations via une requête POST à l'API Flask. La recherche sémantique utilise les embeddings SBERT pour analyser et comparer le contenu des issues.

Le moteur de recherche, connecté à Elasticsearch, retourne alors les 5 issues les plus similaires, classées selon leur pertinence. Ces résultats permettent d'identifier rapidement les issues proches en contexte, facilitant ainsi l'analyse et la résolution des problèmes.

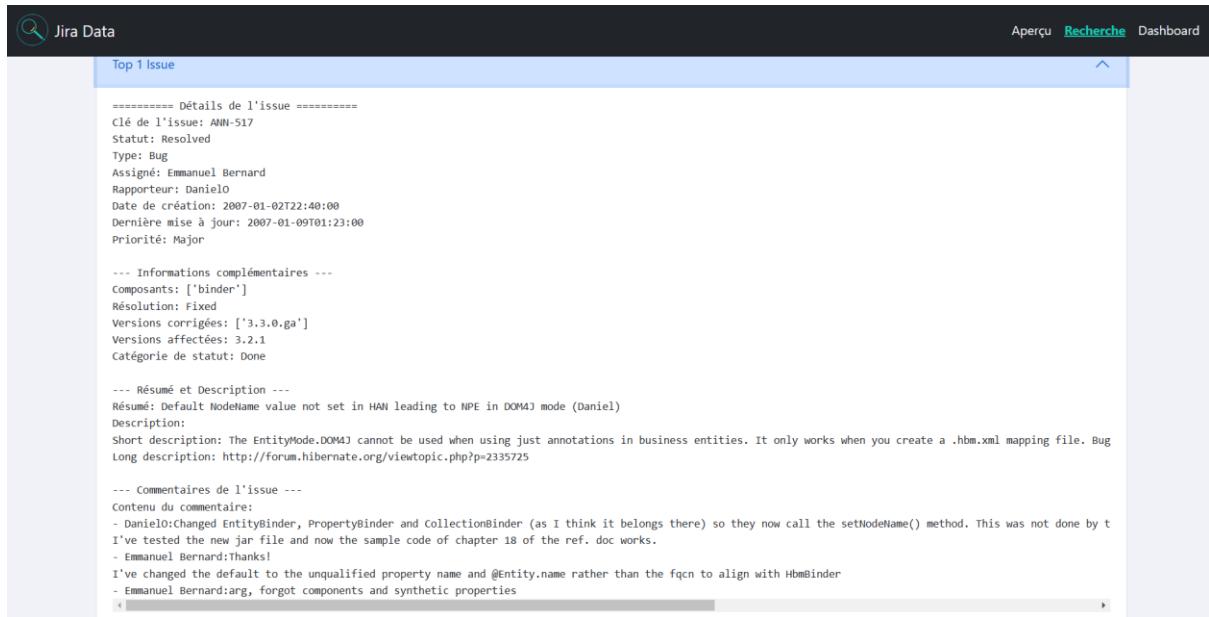


Figure 118 Front end : Résultat de la recherche

## Dashboard Power BI :

Enfin, notre plateforme présente le dashboard Power BI afin de permettre aux utilisateurs d'accéder directement aux visualisations des données sans quitter l'interface.

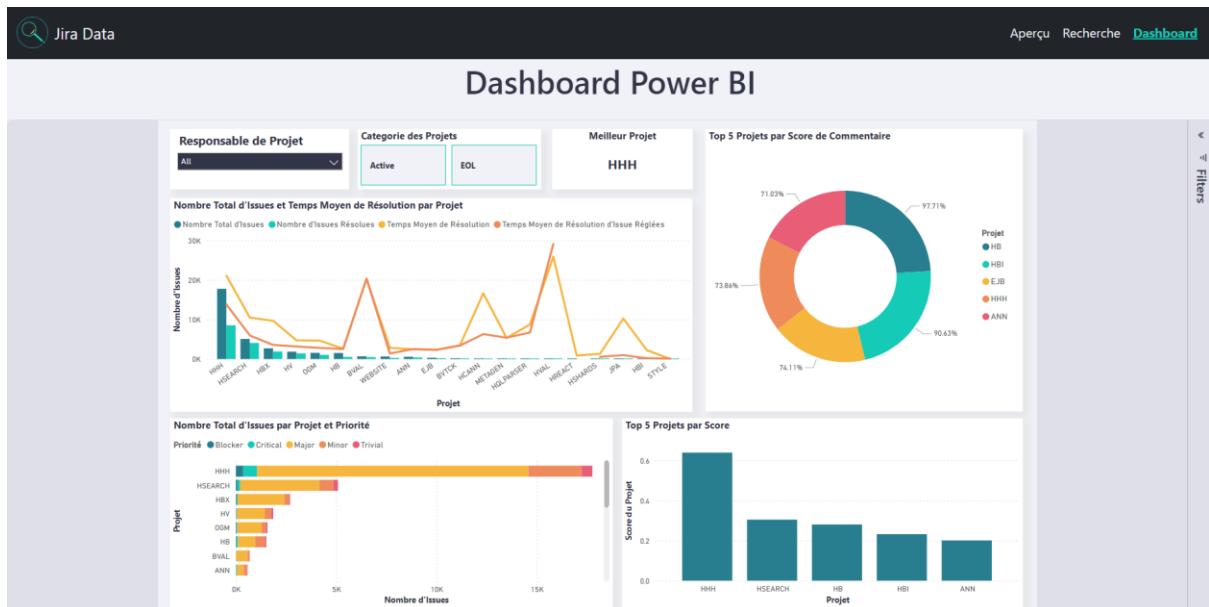


Figure 119 Front end : Dashboard Power BI

## 4. Conclusion

Dans ce chapitre, nous avons abordé le fine-tuning de notre modèle SBERT afin d'améliorer la précision des recherches sémantiques. Ce processus a permis d'adapter le modèle aux spécificités des données du projet. Ensuite, nous avons présenté notre plateforme front-end, qui intègre l'application de recherche développée avec Flask et les tableaux de bord Power BI. Cette

plateforme offre une interface intuitive pour effectuer des recherches avancées et visualiser les données, centralisant ainsi toutes les fonctionnalités essentielles pour une analyse efficace et complète.

# Conclusion et Perspective

Ce rapport offre une vue globale et détaillée des travaux réalisés dans le cadre de ce projet, allant de l'analyse des besoins métiers jusqu'au développement, à la mise en œuvre et à l'optimisation des solutions. Le projet a suivi une approche méthodique et structurée, débutant par la présentation du contexte et la définition des objectifs, suivie par la conception des données, le chargement via des processus ETL, et culminant avec la visualisation des données dans Power BI et l'intégration d'une solution de recherche avancée basée sur Elasticsearch.

Le projet a apporté plusieurs contributions significatives. Premièrement, un entrepôt de données a été conçu et implémenté pour centraliser les informations issues de Jira Service Management, garantissant ainsi une organisation structurée et une exploitation efficace des données. Deuxièmement, des tableaux de bord interactifs ont été développés dans Power BI, offrant une visualisation intuitive pour analyser les performances des projets et des tickets Jira. Ces visualisations permettent de dégager des tendances clés, facilitant ainsi la prise de décision.

En outre, l'intégration d'Elasticsearch avec un modèle SBERT finement ajusté a permis de répondre aux besoins de recherche avancée et sémantique, en fournissant des résultats précis et rapides.

Enfin, l'intégration de cette application dans une interface front-end conviviale a permis de regrouper toutes les fonctionnalités principales comme la visualisation, la recherche et gestion des données dans un seul espace accessible.

## Perspectives

Pour l'avenir, plusieurs axes d'amélioration et de développement peuvent être envisagés afin de renforcer les contributions de ce projet. Une première perspective concerne l'amélioration continue du modèle SBERT utilisé pour la recherche sémantique. Cela pourrait inclure l'entraînement sur des ensembles de données plus vastes et variés pour mieux couvrir les cas complexes et rares, ainsi que l'exploration de modèles de deep learning encore plus performants. L'intégration d'approches basées sur l'apprentissage par transfert permettra également d'améliorer la précision et l'efficacité des recherches.

Un second axe porte sur l'automatisation des processus et la scalabilité du système. La mise en place de pipelines automatisés pour les tâches ETL, l'indexation des données et le déploiement des modèles peut considérablement réduire le temps et les efforts nécessaires à la gestion des données. L'intégration de flux de données en temps réel permettrait une mise à jour continue des données et des analyses.

Les solutions développées dans ce projet peuvent également être adaptées et étendues à d'autres domaines. Par exemple, les techniques utilisées pourraient être appliquées à la maintenance prédictive, à l'analyse de performances dans d'autres systèmes de gestion ou même à des secteurs comme la santé, les finances, ou l'immobilier. Cela ouvre la voie à de nouvelles opportunités d'innovation et d'application.

En conclusion, ce projet constitue une avancée significative dans la gestion et l'analyse des données issues de Jira Service Management. Les méthodologies et outils adoptés jettent les bases d'une solution robuste et évolutive, ouvrant la voie à une amélioration continue des services et à l'exploration de nouvelles opportunités d'innovation dans le domaine de la gestion des données et de la recherche avancée.

## Bibliographie

Jira. (2024, Mars 11). Jira REST API Documentation. Récupéré sur Site web Atlassian: <https://developer.atlassian.com/cloud/jira/platform/rest/v3/intro/>

Elasticsearch. (2024, May 21). Elasticsearch Documentation. Récupéré sur Site web Elastic: <https://www.elastic.co/guide/en/elasticsearch/reference/8.14/index.html>

Elastic Stack. (2024, May 21). Elastic Stack Overview. Récupéré sur Site web Elastic: <https://www.elastic.co/elastic-stack>

Power BI. (2024, May 16). Power BI Documentation. Récupéré sur Site web Microsoft: <https://learn.microsoft.com/en-us/power-bi/>

SentenceTransformers. (2024, Juin 25). SBERT: Sentence-Transformers Library Documentation. Récupéré sur Site web SentenceTransformers: <https://www.sbert.net/>

Python Flask. (2024, Novembre 10). Flask Documentation. Récupéré sur Site web Flask: <https://flask.palletsprojects.com/>