MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR DE LA RECHERCHE SCIENTIFIQUE ET DE L'INNOVATION

UNIVERSITÉ SULTAN MOULAY SLIMANE

ECOLE NATIONALE DES SCIENCES APPLIQUÉES DE KHOURIBGA

Projet de Fin d'Études

pour l'obtention du

Diplôme d'Ingénieur d'État

Filière : Génie informatique

Présenté par :
Hiba **El yagoubi**

Modernisation du suivi crédit : Du traitement batch à l'API temps réel

---

Soutenu le 16 juin devant le jury :

Pr. LAGHRIB Amine	ENSA Khouribga	Encadrant Interne
Pr. HAFIDI Imad	ENSA Khouribga	Président
Pr. NACHAOUI Mourad	ENSA Khouribga	Examinateur

Année universitaire : **2024-2025**

Sommaire

Dédicace

“

À MA TRÈS CHERE MERE,
Source inépuisable de tendresse, de patience et de sacrifice. Ta prière et ta Bénédiction m'ont été d'un grand secours tout au long de ma vie. Quoique je puisse dire et écrire, je ne pourrais exprimer ma grande affection et ma profonde reconnaissance. J'espère ne jamais te décevoir, ni trahir ta confiance et tes sacrifices. Puisse Dieu tout puissant, te préserver et t'accorder santé, longue vie et Bonheur.

À MON CHER PÈRE,
Tu étais et tu restes le meilleur. En ce jour, j'espère réaliser l'un de tes rêves.

À Mes frères

À Mes amies, mes professeurs, et avec toute mon estime et mon respect.

À Vous . . . Je dédie ce travail.

”

- Hiba EL-YAGOUBI

Remerciements

Tout d'abord, je remercie Allah le tout-Puissant de m'avoir donné le courage et la patience nécessaires pour mener à bien ce travail.

Je voudrais également exprimer mes sincères remerciements aux personnes et organisations suivantes qui ont rendu ce stage possible et m'ont soutenu tout au long du projet.

Tout d'abord, je tiens à remercier M. Amine Laghrib de l'École nationale des sciences appliquées de Khouribga de m'avoir donné l'opportunité de mettre mes connaissances académiques en application sur des problèmes du monde réel. Votre engagement à l'enseignement et votre soutien indéfectible ont eu un impact profond sur mon épanouissement personnel et professionnel. Ce rapport est un témoignage de vos efforts pour façonner les jeunes esprits et cultiver les futurs professionnels.

Je suis également infiniment reconnaissant à mon superviseur de Attijariwafa Bank, M. Faycal LEMSEFFER. Votre patience, votre guidance et votre volonté de partager vos connaissances ont été déterminantes dans la formation de ma perspective et dans ma préparation au monde professionnel.

Je tiens également à exprimer ma profonde gratitude à Aicha El Moutaouakkil. Votre leadership éclairé, votre soutien constant et votre capacité à inspirer ceux qui vous entourent ont été une source inestimable de motivation pour moi. Merci d'avoir partagé votre expertise avec tant de générosité et de m'avoir guidé avec sagesse à chaque étape de ce projet.

Je tiens également à exprimer ma sincère gratitude à toute l'équipe de développement.

Je tiens également à exprimer ma sincère gratitude à ma famille et à mes amis pour leur soutien constant, leur amour et leur encouragement. Je voudrais également remercier mes camarades de classe à l'École nationale des sciences appliquées de Khouribga et mes collègues stagiaires à Attijariwafa Bank.

Enfin, je voudrais exprimer ma profonde gratitude à tous les participants qui ont contribué à ce projet. Merci à tous d'avoir fait de ce stage un succès.

Résumé

Ce rapport a été élaboré dans le cadre de mon stage de fin d'études au sein d'un établissement bancaire. Il présente le projet de modernisation du système de rapprochement des autorisations et utilisations de crédit.

Le système actuel EDEN utilise un traitement batch qui s'exécute quotidiennement pour collecter les données d'utilisation provenant de différents systèmes opérationnels, les consolider, puis effectuer un rapprochement avec les tables d'autorisations. Ce processus, bien que fonctionnel, est chronophage et ne permet pas une consultation en temps réel.

L'objectif de ce projet est de remplacer ce traitement batch par une solution moderne basée sur API, permettant d'accéder directement aux données d'utilisation et de les rapprocher en temps réel avec les autorisations stockées dans la base de données du nouveau système. Cette refonte s'inscrit dans le cadre d'une modernisation plus large où la nouvelle application d'instruction crédit fonctionnera en parallèle avec l'ancien système avant de le remplacer définitivement.

Les avantages attendus incluent une réduction significative du temps de traitement et une information instantanée accessible à la demande pour les responsables d'agence, renforçant ainsi l'efficacité opérationnelle de la banque dans la gestion des crédits clients.

Mots clés : Rapprochement crédit, Autorisation-utilisation, Modernisation système d'information, API temps réel, Traitement batch, Migration système, Consultation instantanée, Optimisation processus bancaires, Intégration de données, Transformation numérique.

Abstract

This report was prepared as part of my end-of-studies internship at a banking institution. It presents the project to modernize the system for reconciling credit authorizations and utilizations.

The current EDEN system uses a batch process that runs daily to collect usage data from different operational systems, consolidate it, and then reconcile it with authorization tables. This process, although functional, is time-consuming and does not allow for real-time consultation.

The objective of this project is to replace this batch processing with a modern API-based solution, allowing direct access to usage data and real-time reconciliation with authorizations stored in the new system's database. This redesign is part of a broader modernization where the new credit instruction application will operate in parallel with the old system before permanently replacing it.

The expected benefits include a significant reduction in processing time and instant information accessible on demand for branch managers, thereby strengthening the bank's operational efficiency in managing customer credits.RéessayerClaude peut faire des erreurs. Assurez-vous de vérifier ses réponses.

Keywords: Credit reconciliation, Authorization-utilization, Information system modernization, Real-time API, Batch processing, System migration, Instant consultation, Banking process optimization, Data integration, Digital transformation.

ملخص

تم إعداد هذا التقرير كجزء من تدريبي الختامي في مؤسسة مصرفية. يقدم مشروع تحديث نظام مطابقة تصريحات واستخدامات الائتمان.

يستخدم نظام يضي الحالي عملية دفعية تعمل يومياً لجمع بيانات الاستخدام من أنظمة تشغيلية مختلفة، وتوحيدها، ثم مطابقتها مع جداول التصريح. هذه العملية، رغم كونها وظيفية، تستغرق وقتاً طويلاً ولا تسمح بالاستعلام في الوقت الفعلي.

هدف هذا المشروع هو استبدال هذه المعالجة الدفعية بحل حديث يعتمد على واجهة برمجة التطبيقات (اعي)، مما يتيح الوصول المباشر إلى بيانات الاستخدام والمطابقة في الوقت الفعلي مع التصريحات المخزنة في قاعدة بيانات النظام الجديد. يعد هذا التصميم الجديد جزءاً من تحديث أوسع حيث سيعمل تطبيق تعليمات الائتمان الجديد بالتوازي مع النظام القديم قبل استبداله بشكل دائم.

تشمل الفوائد المتوقعة انخفاضاً كبيراً في وقت المعالجة ومعلومات فورية يمكن الوصول إليها عند الطلب لمديري الفروع، مما يعزز الكفاءة التشغيلية للبنك في إدارة ائتمانات العملاء.

كلمات مفتاحية : مطابقة الائتمان، التصريحالاستخدام، تحديث نظام المعلومات، واجهة برمجة التطبيقات في الوقت الفعلي، المعالجة الدفعية، ترحيل النظام، الاستعلام الفوري، تحسين العمليات المصرفية، تكامل البيانات، التحول الرقمي.

Table des matières

Table des figures

Liste des sigles et acronymes

API	Application Programming Interface
AWB	Attijariwafa Bank
CI/CD	Continuous Integration/Continuous Deployment
CSS	Cascading Style Sheets
DDD	Domain-Driven Design
DTO	Data Transfer Object
EDEN	Système de gestion crédit existant
GGR	Gestion Générale des Risques
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IDE	Integrated Development Environment
	International Electrotechnical Commission
JPA	Java Persistence API
JSON	JavaScript Object Notation
JWT	JSON Web Token
MAD	Dirham Marocain
MIME	Multipurpose Internet Mail Extensions
NDR	Network Detection and Response
OAuth2	Open Authorization 2.0
ORM	Object-Relational Mapping
OWASP	Open Web Application Security Project
PCI DSS	Payment Card Industry Data Security Standard
POM	Project Object Model
POO	Programmation Orientée Objet
REST	Representational State Transfer
RESTful	REST-compliant
SAML	Security Assertion Markup Language
SDK	Software Development Kit
SGBD	Système de Gestion de Base de Données

Introduction Générale

Dans un secteur bancaire en constante évolution technologique, l'optimisation des processus informatiques est devenue un enjeu stratégique majeur pour garantir l'efficacité opérationnelle et la qualité de service. Le système de rapprochement des autorisations et utilisations de crédit représente une composante fondamentale dans la gestion quotidienne des opérations bancaires, impactant directement la performance globale de l'établissement.

Le système actuel EDEN, bien que fonctionnel depuis plusieurs années, présente aujourd'hui des limitations significatives qui entravent l'agilité opérationnelle de la banque. Son architecture reposant sur un traitement batch quotidien génère des contraintes importantes en termes de réactivité et de disponibilité des informations. Cette approche traditionnelle, qui consiste à collecter, consolider et rapprocher les données d'utilisation avec les tables d'autorisations de manière séquentielle, s'avère particulièrement chronophage et ne répond plus aux exigences d'instantanéité des métiers bancaires modernes.

Face à ces défis, notre établissement a initié un projet ambitieux de modernisation visant à transformer radicalement ce processus critique. L'objectif central est de remplacer le traitement batch existant par une solution innovante basée sur des API, permettant d'accéder en temps réel aux données d'utilisation et de les rapprocher instantanément avec les autorisations stockées dans la base de données du nouveau système. Cette refonte s'inscrit dans une démarche plus large de transformation numérique où coexisteront temporairement l'ancien et le nouveau système d'instruction crédit, garantissant ainsi une transition progressive et sécurisée.

Les bénéfices attendus de cette modernisation sont multiples et stratégiques. Au-delà de la réduction significative des temps de traitement, cette nouvelle architecture permettra aux responsables d'agence d'accéder instantanément aux informations critiques, améliorant ainsi considérablement le processus décisionnel et la réactivité face aux demandes clients. Cette transformation contribuera également à renforcer l'efficacité opérationnelle globale de la banque dans la gestion des crédits, un enjeu compétitif majeur dans le paysage bancaire actuel.

Chapitre 1

Contexte général du projet

Introduction

Dans ce chapitre, nous commencerons par présenter l'entreprise d'accueil, son parcours ainsi que ses activités principales. Nous aborderons ensuite la problématique du projet ainsi que les besoins spécifiques qui ont motivé sa réalisation. Enfin, nous exposerons la méthodologie mise en place pour mener à bien ce projet en toute efficacité.

1.1 Présentation de l'organisme d'accueil

1.1.1 Présentation de Attijariwafa Bank

Attijariwafa Bank est une banque commerciale multinationale marocaine et une société de services financiers fondée et basée à Casablanca, au Maroc. C'est la première banque au Maroc et la plus grande banque commerciale.

Attijariwafa Bank est le groupe bancaire de référence en Afrique du Nord, Afrique de l'Ouest et Afrique Centrale. Outre la banque, le Groupe intervient, au travers de filiales spécialisées, dans tous les métiers financiers : assurance, crédit immobilier, crédit à la consommation, crédit-bail, gestion d'actifs, courtage en bourse, conseil, location longue durée, affacturage...

Attijariwafa Bank opère dans 26 pays dont 14 en Afrique (Egypte, Tunisie, Mauritanie, Sénégal, Burkina Faso, Mali, Côte-d'Ivoire, Togo, Niger, Bénin, Congo, Gabon et Cameroun) et en Europe (Belgique, France, Allemagne, Pays-Bas, Italie, Espagne et Suisse) à travers des filiales bancaires contrôlées majoritairement par la banque, à Dubai, Abudhabi, Riyadh, Londres et Montréal à travers des bureaux de représentation.

Attijariwafa Bank est un acteur record sur le continent africain et accélère depuis quelques années sa croissance en Afrique. La Banque de détail à l'international contribue à hauteur de 32,5% au produit net bancaire et de 28,3% au résultat net part du groupe au 30 juin 2021.

Le Groupe dispose du plus grand réseau de distribution au Maroc et le plus dense d'Afrique avec 5 566 agences et emploie 25 754 personnes au service de 10,7 millions de clients au 30 juin 2021. Avec une action quotidienne basée sur les valeurs de leadership, d'engagement, de citoyenneté, d'éthique et de solidarité, Attijariwafa Bank mobilise toutes ses ressources pour servir le continent africain. Attijariwafa Bank est cotée à la Bourse de Casablanca avec une capitalisation boursière de 11,0 milliards de dollars au 30 juin 2022.

التجاري وفا بنك
Attijariwafa bank

FIGURE 1.1 : Logo du groupe AWB

1.1.2 Organigramme de l'entreprise

La sécurité des systèmes d'information est une préoccupation majeure pour les entreprises, en particulier pour les institutions financières qui traitent des informations sensibles et confidentielles sur leurs clients. Le groupe Attijariwafa Bank est conscient de cette préoccupation et a mis en place une entité de sécurité opérationnelle au sein de son Système d'Informations Groupe (SIG) pour protéger ses systèmes d'information et préserver la confidentialité et l'intégrité des données de ses clients.

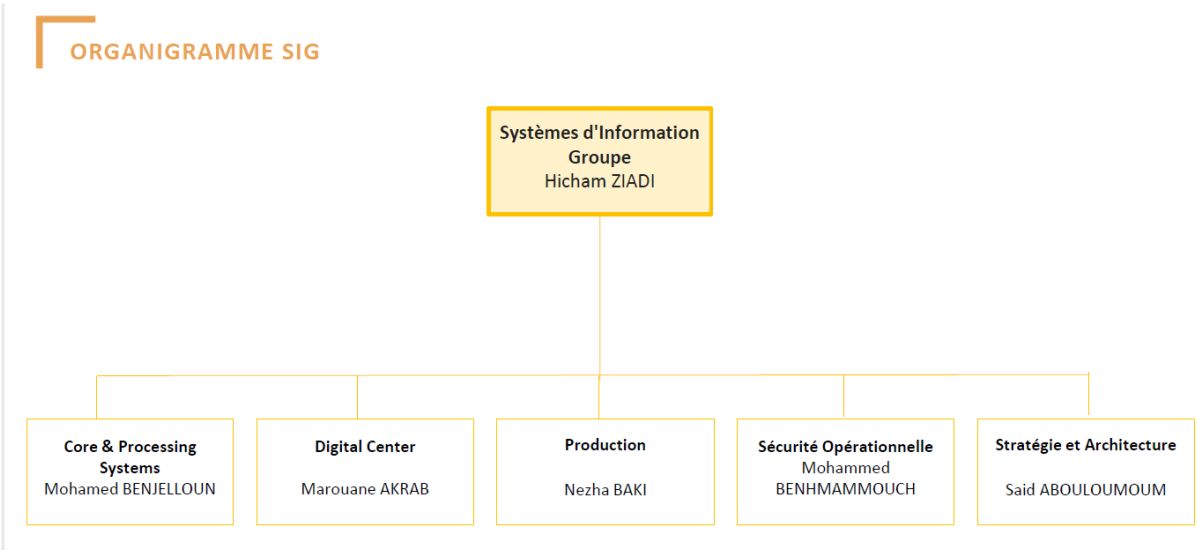

FIGURE 1.2 : Organigramme SIG chez Attijariwafa bank

1.1.3 Entité Core & Processing Systems

L'entité CPS (Core & Processing Systems) représente le cœur technologique des opérations bancaires d'Attijariwafa Bank, gérant l'infrastructure système centrale qui supporte l'ensemble des activités

transactionnelles du groupe. Cette entité stratégique assure la gestion et l'évolution du système d'information bancaire (Core Banking System) qui constitue l'épine dorsale technologique permettant le traitement en temps réel de millions de transactions quotidiennes à travers le réseau d'agences et les canaux digitaux. Le champ d'intervention de CPS englobe :

— La gestion du Core Banking System (CBS) pour le traitement des opérations bancaires
— L'administration des systèmes de paiement et de compensation (virements, prélèvements, cartes)
— Le processing des opérations monétiques et des transactions électroniques
— La maintenance et l'évolution des systèmes de gestion des comptes et produits bancaires
— L'intégration des systèmes périphériques avec le core banking (CRM, risques, reporting)
— La garantie de la haute disponibilité et performance des systèmes critiques
— L'orchestration des interfaces avec les systèmes de paiement nationaux et internationaux
— La gestion des batchs nocturnes et des processus de clôture comptable

Cette entité joue un rôle crucial dans la transformation digitale de la banque en modernisant continuellement l'architecture système pour supporter les nouveaux services (mobile banking, paiements instantanés, open banking) tout en maintenant la stabilité, la sécurité et la conformité réglementaire des systèmes existants. L'expertise de CPS permet d'assurer la scalabilité nécessaire à l'expansion du groupe et l'intégration technologique des filiales africaines.

1.2 Cadre du projet

1.2.1 Problématique et description du besoin

La gestion des rapprochements entre autorisations et utilisations de crédit est un enjeu critique pour notre établissement bancaire, confronté à des exigences croissantes en matière de réactivité et de prise de décision. Les défis actuels résident dans l'instantanéité de l'accès aux données et dans l'analyse exhaustive des encours des clients. Actuellement, le système EDEN fonctionne sur un mode batch quotidien, ce qui implique que la visualisation des résultats de rapprochement n'est accessible que le lendemain matin. Cette contrainte temporelle entraîne des retards significatifs dans la prise de décision et limite la réactivité face aux demandes clients. La question centrale est donc : comment transformer ce processus pour permettre une consultation en temps réel des données critiques ?

Une fois les données d'utilisation détectées, un autre problème se pose : comment les intégrer efficacement au nouveau système d'instruction crédit pour une exploitation optimale ? Il ne suffit pas de collecter ces informations ; il est également crucial de les rendre immédiatement disponibles et exploitables pour les responsables d'agence et la Direction Générale des Risques (GGR). Cette disponibilité immédiate nécessite une architecture technique innovante basée sur des API plutôt que sur des traitements batch traditionnels. Il est donc impératif d'identifier les meilleures pratiques pour intégrer ces données en temps réel et les présenter de manière pertinente dans l'onglet risque et solvabilité du nouveau système.

En outre, il est important d'enrichir la qualité des informations disponibles pour améliorer le processus décisionnel. L'adoption d'une architecture orientée services peut jouer un rôle clé en permettant l'accès instantané à des données critiques. Par exemple, cette nouvelle approche peut être utilisée pour évaluer automatiquement la situation d'endettement globale d'un client en fonction de ses engagements actuels et historiques, permettant ainsi une analyse de risque plus précise et informée. De plus, cette architecture moderne peut faciliter la corrélation des données d'utilisation avec d'autres indicateurs financiers, fournissant une vision plus complète et contextuelle du profil de risque des clients.

1.2.2 Objectifs et portée du projet

Pour répondre aux problématiques identifiées, plusieurs objectifs doivent être atteints.

Premièrement, il est essentiel de comprendre les limitations actuelles du système EDEN et leur impact sur l'efficacité opérationnelle de la banque. Cette compréhension approfondie permettra d'identifier les points critiques nécessitant des transformations prioritaires.

Ensuite, il est nécessaire de définir une architecture technique innovante permettant le rapprochement en temps réel des autorisations et utilisations de crédit. Cette architecture doit inclure des processus automatisés pour collecter, analyser et présenter les données, réduisant ainsi le délai entre la collecte et l'exploitation des informations.

Les objectifs spécifiques à atteindre sont les suivants :

- **Analyser l'existant :** Évaluer les performances et limitations du système EDEN actuel pour identifier précisément les axes d'amélioration et définir une stratégie de migration appropriée.
- **Concevoir une architecture API :** Développer une architecture technique basée sur des API permettant d'accéder en temps réel aux données d'utilisation et de les rapprocher instantanément avec les autorisations stockées dans la base de données du nouveau système.
- **Assurer la coexistence temporaire :** Mettre en place une stratégie de parallel run permettant la coexistence temporaire de l'ancien système EDEN et du nouveau système d'instruction crédit, garantissant ainsi une transition progressive et sécurisée.
- **Intégrer dans l'onglet risque et solvabilité :** Incorporer les données de rapprochement en temps réel dans l'onglet dédié au risque et à la solvabilité du nouveau système, offrant ainsi aux décideurs une vision instantanée et complète de la situation d'endettement des clients.
- **Optimiser le processus décisionnel :** Améliorer significativement la réactivité face aux demandes clients en permettant à la Direction Générale des Risques (GGR) d'accéder immédiatement aux informations critiques pour l'évaluation des risques.

- **Automatiser et centraliser le processus de rapprochement :** Intégrer des mécanismes automatisés de collecte et d'analyse des données d'utilisation, tout en centralisant le processus pour une meilleure cohérence et efficacité.

En atteignant ces objectifs, notre établissement pourra non seulement améliorer la réactivité et la précision de son processus de rapprochement des autorisations et utilisations de crédit, mais aussi assurer une prise de décision plus éclairée et rapide face aux demandes de crédit, renforçant ainsi son avantage concurrentiel dans le secteur bancaire.

1.2.3 planification du projet

Après une réunion durant laquelle nous avons bien défini le projet, nous avons procédé à l'élaboration d'un planning prévisionnel convenu avec les différents intervenants, afin de bien définir les délais et les livrables de chaque phase. Gantt Project est un outil permettant de gérer les projets sur les modèles des diagrammes de Gantt.

La figure ci-dessous représente le diagramme de Gantt qui résume les différentes étapes du projet, dès le début du stage, passant par les formations que nous avons reçues, l'intégration avec les différentes parties prenantes de projet et la réalisation du projet, jusqu'à la fin du stage.

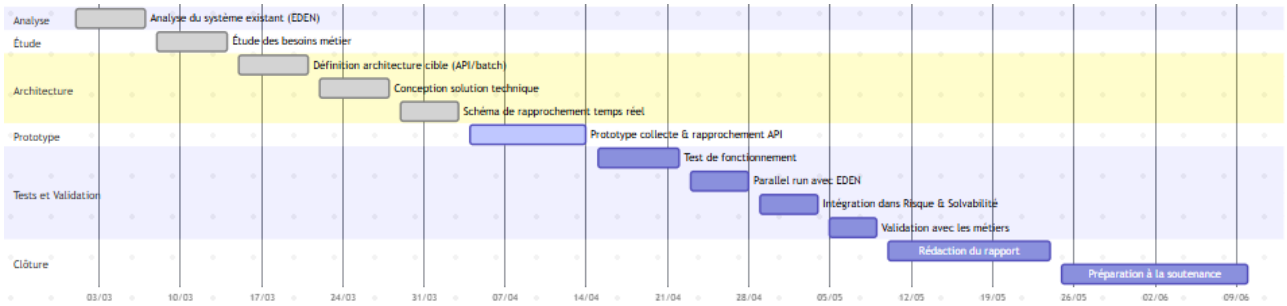

FIGURE 1.3 : Diagramme de Gantt

1.2.4 Méthodologie de gestion du projet

Le projet s'est déroulé au sein d'une équipe adoptant la méthode **Agile Scrum**, un cadre de gestion de projet itératif favorisant la collaboration, la flexibilité et la livraison incrémentale de valeur.

L'organisation repose sur des **sprints**, généralement de deux à trois semaines, et sur l'utilisation de l'outil **Jira** pour structurer les tâches sous forme d'*epics* et de *user stories*.

Sprint Planning : réunion de planification au début de chaque sprint, permettant de définir les objectifs et les tâches à accomplir.

Daily Scrum : réunions quotidiennes rapides (stand-up) où chaque membre partage son avancement, les blocages éventuels et ses objectifs de la journée.

Sprint Review : à la fin de chaque sprint, une démonstration du travail réalisé est faite aux parties prenantes pour recueillir leurs retours.

Sprint Retrospective : l'équipe analyse son fonctionnement pour identifier des axes d'amélioration continue.

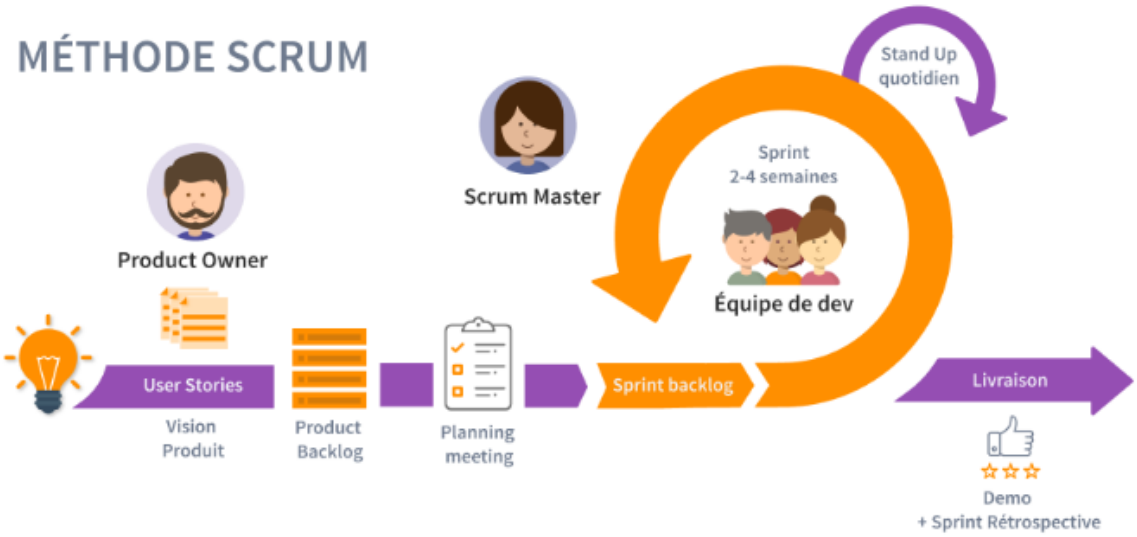

FIGURE 1.4 : Schématisation du fonctionnement de la méthode Scrum

En tant que stagiaire, j'ai pu assister à ces cérémonies et observer de près la dynamique agile de l'équipe, ce qui m'a permis de mieux comprendre l'organisation et le pilotage d'un projet en environnement professionnel.

1.2.4.1 Outil de collaboration

Microsoft Teams Microsoft Teams est une plateforme de collaboration et de communication conçue pour les entreprises et les organisations. Elle permet aux équipes de travailler ensemble de manière efficace en offrant des fonctionnalités telles que la messagerie instantanée, les appels audio et vidéo, les réunions en ligne, la gestion de documents, la gestion de tâches et de projets, ainsi que des intégrations avec d'autres outils. En un seul endroit, Teams centralise les communications et la collaboration, facilitant ainsi le travail en équipe, que ce soit en présentiel ou à distance.

FIGURE 1.5 : Microsoft Teams

Conclusion

Dans ce chapitre, nous avons présenté le cadre général du projet, en mettant en valeur les solutions des problématiques à résoudre, ainsi que la démarche adoptée pour la gestion de projet. Le chapitre suivant mettra plus de lumière sur les concepts clés de la livraison continue et la gestion de configuration

Chapitre 2

Analyse de l'existant

Introduction

Ce chapitre présente une analyse fonctionnelle de la transformation du système EDEN vers une architecture API moderne. Nous examinons d'abord le positionnement stratégique d'EDEN au sein de l'écosystème bancaire et sa coexistence temporaire avec le nouveau système d'instruction crédit. Nous détaillons ensuite le processus batch de rapprochement entre autorisations et utilisations, en identifiant ses limites techniques et opérationnelles. Cette analyse constitue le fondement pour comprendre les enjeux et les objectifs de la modernisation du système, dont les aspects techniques seront développés dans les chapitres suivants.

2.1 Système actuel EDEN : Vue d'ensemble

2.1.1 Rôle et positionnement d'EDEN dans l'écosystème bancaire

EDEN représente un pilier stratégique dans l'infrastructure informatique de la banque. Déployé depuis plusieurs années, ce système constitue le référentiel central des données de crédit et joue un rôle déterminant dans les processus décisionnels quotidiens de l'établissement.

Les fonctions essentielles d'EDEN comprennent :

- **Référentiel central de données** : EDEN centralise et maintient l'ensemble des informations relatives aux crédits, incluant les autorisations, les utilisations, les garanties et les engagements clients.
- **Orchestrateur de processus crédit** : Le système coordonne l'ensemble du cycle de vie des opérations de crédit, depuis l'instruction des demandes jusqu'au suivi des encours.
- **Interface avec les systèmes opérationnels** : EDEN assure l'interopérabilité avec les différentes applications métier, permettant une circulation fluide des informations entre les composants du système d'information.
- **Support décisionnel** : Les données consolidées et rapprochées par EDEN alimentent les outils d'aide à la décision utilisés par les responsables d'agence et les analystes risque.

- **Plateforme réglementaire** : Le système fournit les données nécessaires aux reportings réglementaires et à la conformité bancaire.

Dans le contexte de la transformation numérique actuelle, EDEN conserve son statut de maître des données crédit durant toute la phase de migration. Cette position stratégique garantit la continuité opérationnelle tout en facilitant une transition progressive vers la nouvelle architecture.

2.1.2 Description macroscopique du traitement batch de rapprochement autorisations/utilisations

Le traitement batch de rapprochement dans le système EDEN repose sur une chaîne de traitements séquentiels visant à centraliser, nettoyer, organiser puis rapprocher les données d'utilisation de crédit issues de multiples systèmes opérationnels (SOP).

1. Consolidation des fichiers d'utilisations provenant des SOP Chaque système opérationnel de gestion de crédit (ACOR, CONFIRMING, TI, SUPRA, Escompte sans recours, Portefeuille, etc.) génère quotidiennement un fichier contenant les données d'utilisation. Ces fichiers, nommés selon un format normé (ex. `RISQU.CAUTION.VERS.EDEN`, `EN110.TI.UTIL`), sont transmis à EDEN. Une première étape de traitement consiste à agréger l'ensemble de ces fichiers dans un fichier global unifié, qui servira de base aux traitements suivants.

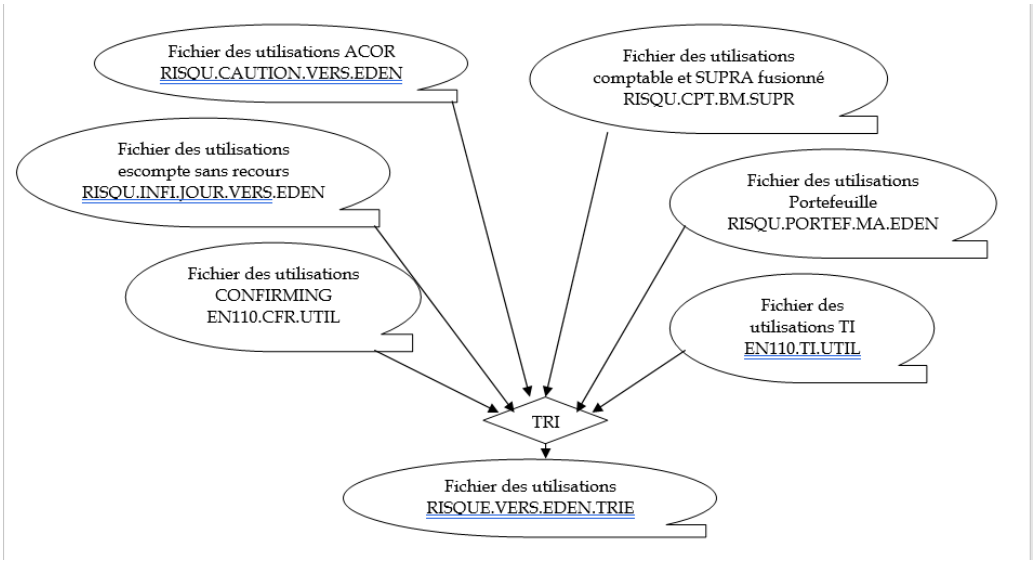

FIGURE 2.1 : Consolidation des fichiers d'utilisations issus des SOP

2. Tri initial par compte Le fichier consolidé est ensuite trié une première fois selon des critères structurants (compte, produit, etc.) afin de faciliter les traitements de contrôle et de rapprochement. Le fichier ainsi obtenu est nommé `RISQUE.VERS.EDEN.TRIE`.

3. Élimination des doublons (ENUB008) Le tri initial est suivi d'un traitement de dédoublonnage effectué par le programme batch `ENUB008`. Ce dernier identifie et supprime les enregistrements redondants, générant un fichier nettoyé nommé `RISQU.FUSION.GLOBA.EDEN`.

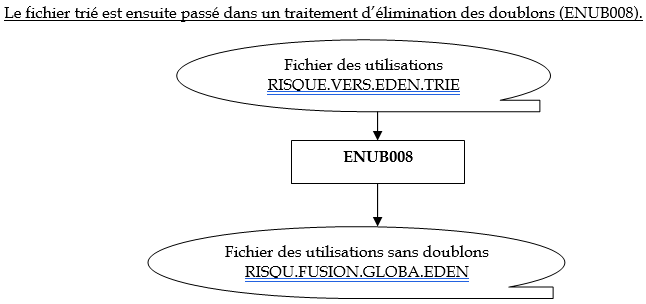

FIGURE 2.2 : Suppression des doublons par ENUB008

4. Tri final post-fusion Après suppression des doublons, le fichier est de nouveau trié par compte et produit, générant une version finale propre du fichier d'utilisations, toujours nommée `RISQUE.VERS.EDEN.TRI` prête à être utilisée pour le rapprochement avec les autorisations.

5. Rapprochement autorisations / utilisations (ENUB004) Le programme batch `ENUB004` prend comme entrée le fichier trié des utilisations (`RISQUE.VERS.EDEN.TRIE`) et le confronte aux données d'autorisations disponibles dans différentes tables de référence d'EDEN (`EAUTORIS`, `EAUTO-RES`, `EARR-AUTO`, `EARRANGT`). Ce rapprochement permet de produire la table `EUTILISE` qui reflète l'état d'utilisation actualisé des lignes de crédit.

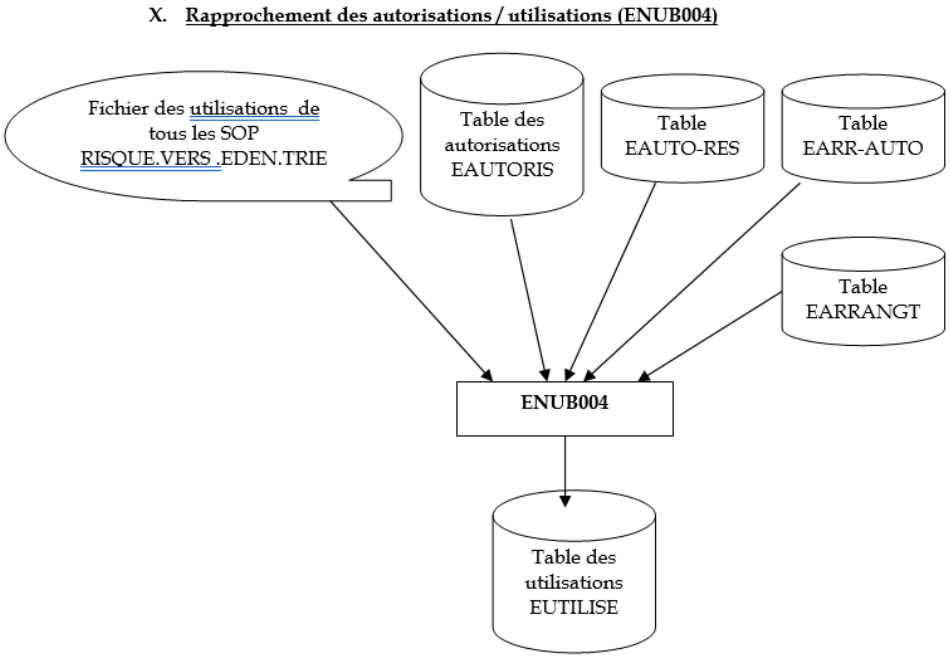

FIGURE 2.3 : Rapprochement des autorisations et utilisations via ENUB004

2.1.3 Limites principales du système batch qui justifient la modernisation

Le traitement batch quotidien utilisé actuellement pour le rapprochement des autorisations et utilisations dans le système EDEN repose sur une architecture traditionnelle. Bien que cette approche ait historiquement répondu aux besoins, elle présente aujourd'hui plusieurs limites structurelles et fonctionnelles qui justifient une modernisation vers un traitement plus réactif, voire en temps réel.

1. Latence importante dans la disponibilité des données Le traitement batch repose sur un principe d'accumulation des données tout au long de la journée. Les fichiers issus des SOP (ex. `RISQU.CAUTION.VERS.EDEN`,RISQU.CPT.BM.SUPR, etc.) ne sont collectés puis traités qu'à partir d'un déclenchement programmé, généralement en fin de journée. Par conséquent, les résultats (rapprochement, alertes, indicateurs) ne sont disponibles que le lendemain matin, induisant une latence systémique d'un jour dans la mise à disposition des données actualisées.

2. Absence de visibilité en temps réel Cette latence empêche les équipes métier (gestionnaires de crédit, analystes risques, etc.) d'avoir une vision instantanée de l'état d'utilisation des autorisations. Tout écart ou anomalie (ex. dépassement de plafond, incohérence de données) n'est détecté qu'après traitement nocturne, ce qui nuit à la réactivité opérationnelle et au pilotage du risque.

3. Risque accru en cas d'échec du traitement batch En cas d'incident technique ou de défaillance dans l'un des traitements batch (ex. ENUB008 ou ENUB004), l'ensemble du processus est bloqué, et les données du jour ne sont pas traitées. Cela peut engendrer des retards dans le reporting, le suivi des engagements ou la détection des anomalies, avec des impacts opérationnels et réglementaires.

4. Complexité croissante de maintenance et de supervision L'empilement des traitements spécifiques à chaque SOP, la multiplicité des fichiers intermédiaires, et la chaîne de traitements séquentiels (tri, dédoublonnage, fusion, rapprochement) rendent l'architecture difficile à maintenir. La supervision opérationnelle est également lourde, notamment en cas d'anomalie dans un des fichiers sources ou de rupture dans la chaîne de traitement.

5. Difficulté à répondre à des besoins métiers plus dynamiques Les attentes des utilisateurs évoluent vers une disponibilité des données en quasi-temps réel, notamment pour les besoins de suivi des risques, de conformité (compliance), ou de pilotage des engagements. Le modèle batch, par définition statique et différé, ne peut répondre à ces exigences de fluidité et d'instantanéité.

2.2 La nouvelle solution : Vision fonctionnelle et architecture

2.2.1 Présentation de l'architecture générale du nouveau système

La nouvelle architecture repose sur un modèle orienté API et événementiel, permettant une circulation fluide, sécurisée et en quasi-temps réel des données entre les différents acteurs du crédit. Elle s'inscrit dans une démarche de modernisation des processus existants en s'appuyant sur des technologies d'intégration modernes telles que **Kafka** et des APIs *RESTful* exposées selon les besoins métier.

L'architecture introduit une **stratégie modulaire** basée sur des composants interopérables. Chaque API assume un rôle spécifique dans le cycle de vie du crédit, depuis l'instruction jusqu'au suivi des risques et des limites.

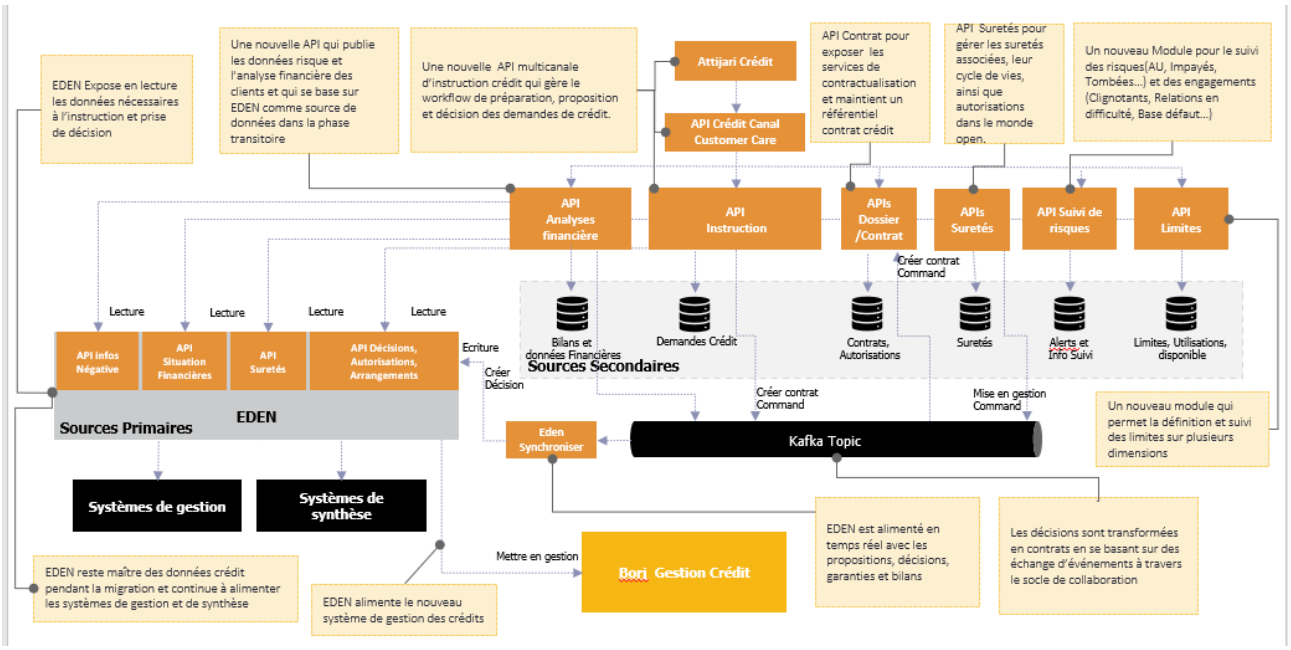

FIGURE 2.4 : Architecture API et interactions entre systèmes

2.2.2 Organisation fonctionnelle des différentes APIs

Les APIs sont organisées selon les étapes clés du processus de gestion du crédit :

- **API Instruction** : Cœur de l'orchestration du processus de crédit, elle gère le workflow multi-canal de préparation, de proposition et de décision des demandes de crédit.
- **API Analyses financières** : Récupère les données financières depuis EDEN et les expose pour les traitements analytiques.
- **API Dossier/Contrat** : Gère la création, la modification et la gestion des contrats et autorisations.
- **API Sûretés** : Permet de gérer les garanties associées, leur valorisation et leur suivi.
- **API Suivi des risques** : Suit les expositions, incidents, dépassements et autres indicateurs de risque.
- **API Limites** : Permet de définir et de suivre les limites d'engagement sur plusieurs dimensions (client, groupe, produit, etc.).

Les différentes APIs interagissent avec des bases de données intermédiaires (bilans, contrats, alertes, etc.) et utilisent la plateforme **Kafka Topic** pour la gestion des événements métiers (création de contrat, décision, mise en gestion...).

2.2.3 Stratégie de coexistence temporaire entre EDEN et le nouveau système

Durant la phase transitoire, **EDEN** reste le référentiel maître des données crédit. Il continue d'alimenter les systèmes de gestion et de synthèse tout en exposant, en lecture, les données nécessaires aux nouvelles APIs.

Une couche de synchronisation, via le composant **Eden Synchroniser**, assure la cohérence entre EDEN et les nouvelles bases secondaires, garantissant une transition fluide sans rupture opérationnelle.

En parallèle, EDEN est enrichi en lecture avec les nouvelles décisions, propositions, et garanties générées par la nouvelle architecture. Cette stratégie permet une **cohabitation progressive** entre l'ancien et le nouveau système, limitant ainsi les risques liés à la migration tout en bénéficiant des apports de l'architecture moderne.

2.3 Focus sur le module d'instruction crédit

2.3.1 Positionnement du module d'instruction crédit dans l'architecture globale

Le module d'instruction crédit occupe une position centrale dans la nouvelle architecture API, comme le montre la figure précédente. Il constitue le point d'entrée principal pour les demandes de crédit provenant des différents canaux (Borj Accueil/Vision 360°, Borj Vente, Attijari Instruction Crédit, Self Care). Cette position stratégique lui confère un rôle d'orchestrateur entre les référentiels clients/prospects et les systèmes de production crédit.

Le module s'interface principalement avec :

- Les canaux de distribution en amont, qui initient les demandes de crédit
- Le Hub Scoring, qui fournit les évaluations nécessaires à la décision
- L'API Analyse Financière, qui expose les données financières issues d'EDEN
- Les systèmes de production crédit en aval (ACCOR, SUPRA, IRIS, ARCO, etc.)
- La base de données "Demandes Crédit" qui centralise l'ensemble des informations relatives aux instructions

Cette position d'interface permet au module d'instruction crédit de servir de pivot dans la transformation des processus traditionnels batch vers une architecture événementielle et temps réel.

2.3.2 Fonctionnalités clés de l'API d'instruction crédit

L'API d'instruction crédit offre un ensemble de services structurés autour du cycle de vie complet d'une demande de crédit :

- **Montage et instruction des demandes client** : Saisie structurée et validation des informations nécessaires au traitement d'une demande, avec possibilité d'enrichissement progressif du dossier.
- **Scoring** : Interface avec le Hub Scoring pour l'obtention des scores et l'intégration de ces éléments dans le processus décisionnel.

- **Workflow de décision** : Orchestration du circuit de validation, incluant les différents niveaux d'approbation selon la nature et le montant du crédit.
- **Unités traitantes et alertes d'instruction** : Gestion des affectations aux équipes responsables et génération d'alertes contextuelles pour optimiser le traitement.
- **Rapprochement autorisations/utilisations en temps réel** : Évolution majeure par rapport au système EDEN, permettant une vision instantanée des engagements et disponibilités.

Cette dernière fonctionnalité représente une rupture fondamentale avec le modèle batch historique d'EDEN, offrant aux utilisateurs métier une visibilité immédiate sur l'état des engagements clients.

2.3.3 Enjeux spécifiques du rapprochement autorisations/utilisations en temps réel

La transformation du processus batch de rapprochement autorisations/utilisations en un mécanisme temps réel intégré à l'API d'instruction crédit soulève plusieurs enjeux techniques et fonctionnels :

1. Réactivité et disponibilité des données Contrairement au système batch EDEN qui génère un décalage d'un jour dans la disponibilité des informations rapprochées, l'approche API permet d'obtenir instantanément l'état actualisé des utilisations par rapport aux autorisations accordées. Cette réactivité améliore significativement la qualité des décisions prises lors de l'instruction de nouvelles demandes de crédit.

2. Gestion événementielle des mises à jour Le nouveau système s'appuie sur une architecture événementielle où chaque opération significative (nouvelle utilisation, modification d'une autorisation, etc.) génère un événement propagé via la plateforme Kafka. Cette approche "push" se substitue au modèle "pull" périodique du batch, garantissant une mise à jour continue des informations.

3. Synchronisation avec l'existant pendant la période de transition Durant la période de coexistence, le module doit assurer une synchronisation bidirectionnelle avec EDEN :

- Consommation des données d'autorisations issues d'EDEN (via Eden Synchroniser)
- Enrichissement du référentiel EDEN avec les nouvelles décisions prises dans l'API d'instruction
- Consolidation des utilisations provenant des systèmes de gestion de crédit

4. Performance et volumétrie Le passage d'un traitement batch à un traitement temps réel nécessite une attention particulière aux aspects de performance, notamment :

- Optimisation des requêtes et des temps de réponse
- Gestion efficace du cache pour les données fréquemment consultées
- Capacité à absorber les pics de charge lors des périodes d'intense activité commerciale

5. Qualité et cohérence des données L'approche temps réel exige un niveau de qualité et de cohérence des données supérieur au traitement batch. Les mécanismes de contrôle doivent être intégrés nativement dans les flux API plutôt que dans des traitements correctifs a posteriori.

Conclusion

Ce chapitre a présenté l'analyse détaillée du système EDEN actuel et les enjeux de sa modernisation. Le processus batch de rapprochement, bien que fonctionnel, montre des limites critiques : latence de 24 heures, absence de visibilité temps réel, et complexité de maintenance croissante.

La nouvelle architecture API proposée, centrée autour du module d'instruction crédit, transforme radicalement ce processus en permettant un rapprochement instantané des autorisations et utilisations. La stratégie de coexistence avec EDEN garantit une transition progressive et sécurisée vers cette solution moderne.

Les défis techniques identifiés - performance, synchronisation, gestion événementielle - serviront de base pour la conception détaillée présentée dans le chapitre suivant.

Chapitre 3

Conception de la solution

Introduction

Ce chapitre présente la conception détaillée de la solution de modernisation du rapprochement autorisations/utilisations. Nous aborderons l'architecture technique, les patterns de conception adoptés, la stratégie de sécurisation, ainsi que les choix technologiques qui guideront l'implémentation. Cette conception vise à transformer le traitement batch actuel en une solution API temps réel, réactive et sécurisée.

3.1 Architecture technique globale

3.1.1 Vue d'ensemble de l'architecture

La nouvelle architecture s'articule autour d'une approche modulaire basée sur une séparation claire des responsabilités, permettant une évolutivité et une maintenabilité optimales. L'architecture suit un pattern en couches avec une façade API backend qui sert d'interface entre le front-end React et les services métier.

Composants principaux :

- **Interface React** : Composant intégré dans l'onglet « Risque et Solvabilité » permettant aux responsables GGR de consulter en temps réel l'état des autorisations/utilisations et de prendre des décisions de crédit éclairées
- **Module Canal (Customer Care)** : Façade API backend qui expose les services REST pour le front-end React. Ce module fait office d'interface entre l'application web et les services métier, gérant la réception des requêtes HTTP provenant de l'interface utilisateur
- **Module Multicanal (Processing Service)** : Cœur métier de l'application contenant toute la logique de rapprochement autorisations/utilisations. Il se compose de deux sous-modules :
  - **SDK (credit-processing-sdk)** : Interfaces partagées, clients API et DTOs pour la communication inter-modules
  - **Service (credit-processing-service)** : Implémentation de la logique métier, repositories et mappers

- **Module Connector** : Gestionnaire centralisé des communications avec les APIs externes, implémentant le pattern Connector pour standardiser les intégrations et gérer l'authentification OAuth2 vers l'API externe
- **Base de données Oracle** : Persistance des données enrichies et mappées avec gestion des migrations via Liquibase

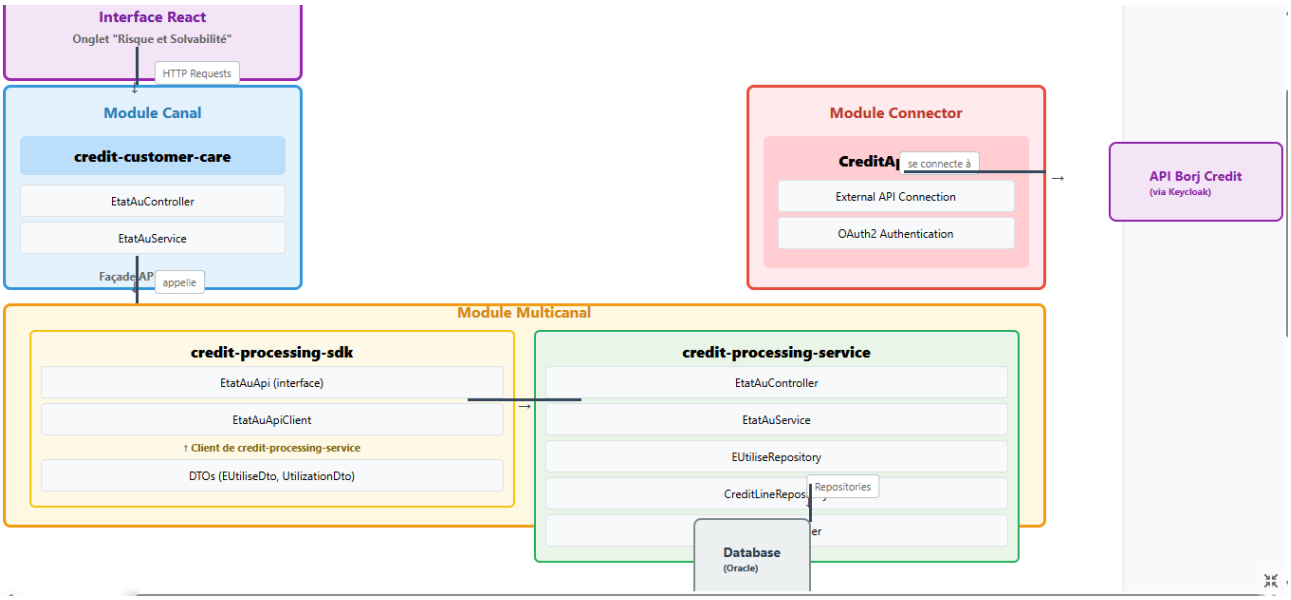

FIGURE 3.1 : Diagramme d'architecture technique

Cette architecture modulaire garantit :

- **Séparation des préoccupations** : Chaque module a une responsabilité claire et délimitée
- **Scalabilité** : Possibilité d'ajouter de nouveaux connecteurs ou services sans impact sur l'existant
- **Testabilité** : Isolation des couches facilitant les tests unitaires et d'intégration
- **Réutilisabilité** : Les connecteurs et services peuvent être réutilisés par d'autres applications

3.1.2 Pattern architectural adopté

L'architecture suit le pattern **Hexagonal Architecture** (ou Ports and Adapters), un modèle qui place la logique métier au centre, entourée de ports et d'adaptateurs pour les interactions externes. Ce choix s'avère particulièrement adapté pour notre contexte bancaire où la fiabilité et la maintenabilité sont critiques.

Implémentation dans notre architecture :

- **Le cœur hexagonal** : Module Multicanal contenant la logique de rapprochement
- **Les ports** : Interfaces définies (EtatAuApi, repositories)
- **Les adaptateurs d'entrée** : Module Canal et Interface React

- **Les adaptateurs de sortie** : Module Connector et base Oracle

Avantages garantis :

• **Isolation du domaine métier :** La logique business de rapprochement reste indépendante des détails techniques. Les règles métier bancaires sont encapsulées sans dépendance vers Spring, REST ou Oracle, permettant des évolutions techniques sans impact sur le cœur métier.

• **Testabilité :** L'architecture permet des tests unitaires isolés du domaine métier avec des mocks, des tests d'intégration par adaptateur, et une validation end-to-end complète.

• **Évolutivité :** Facilité d'ajout de nouveaux adaptateurs (canaux d'accès, sources de données, technologies) sans modification du cœur métier. Support de patterns de résilience spécifiques par adaptateur (circuit breaker, retry, fallback).

Cette approche respecte les exigences de gouvernance IT bancaire avec une séparation claire des préoccupations, la traçabilité des flux et l'auditabilité des transactions.

3.1.3 Principes de conception appliqués

SOLID Principles :

- **Single Responsibility** : Chaque module a une responsabilité unique
- **Open/Closed** : Extension sans modification via les interfaces
- **Liskov Substitution** : Les implémentations sont interchangeables
- **Interface Segregation** : Interfaces spécialisées par usage
- **Dependency Inversion** : Dépendances vers les abstractions

Domain-Driven Design (DDD) :

- **Bounded Context** : Séparation claire des domaines métier
- **Aggregate** : Entités cohérentes (EUtilise comme agrégat principal)
- **Repository Pattern** : Abstraction de la persistance
- **Domain Services** : Logique métier complexe encapsulée

3.2 Modèle de données et conception du domaine

3.2.1 Modèle conceptuel de données

Le modèle de données s'organise autour de trois entités principales formant un domaine cohérent :

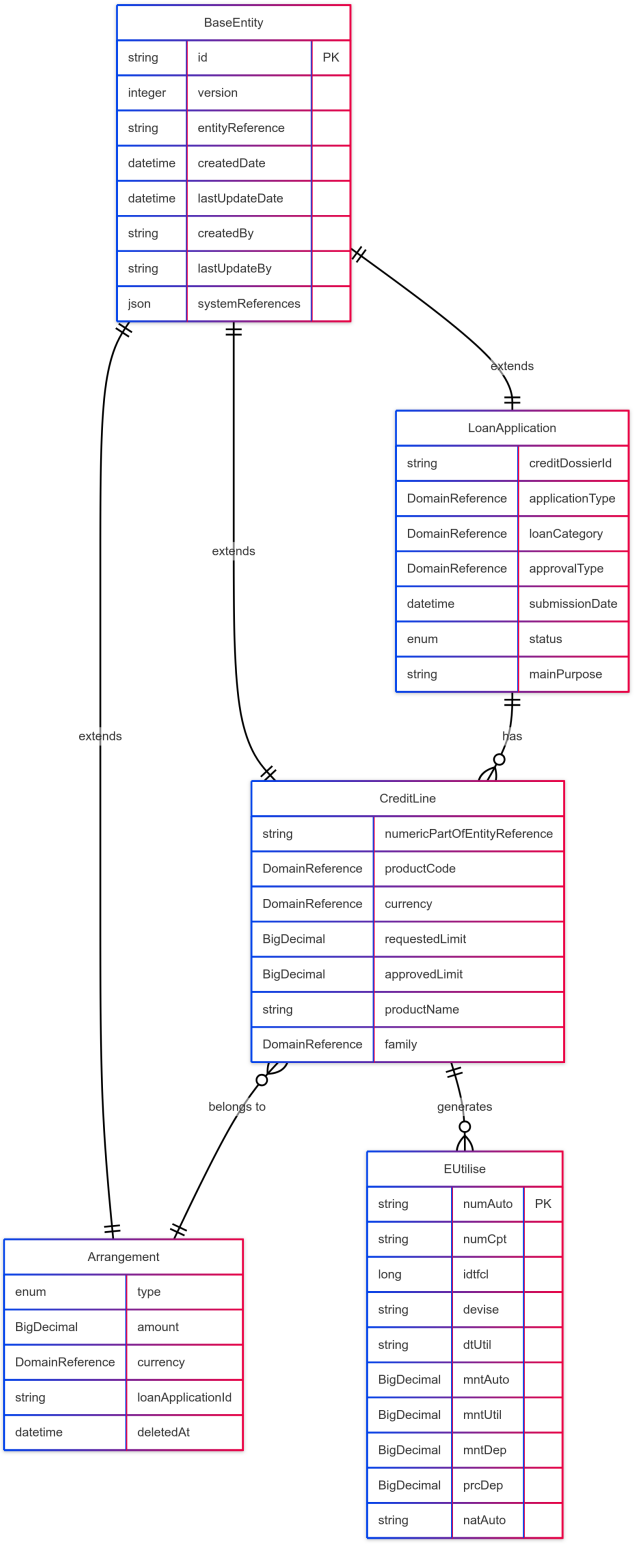

FIGURE 3.2 : Diagramme de classe

3.2.2 Entités du domaine

BaseEntity - Classe abstraite commune :

- **Responsabilité** : Audit trail et métadonnées communes
- **Caractéristiques** : UUID, versioning, timestamps, références système

- **Pattern** : Template Method avec @PrePersist et @PreUpdate

LoanApplication - Agrégat racine :

- **Domaine** : Demande de crédit complète
- **Composition** : Type, catégorie, approbation via DomainReference
- **Relations** : CreditLine (1 :N), Arrangement, Project
- **Logique métier** : Calcul des limites totales, gestion des phases

CreditLine - Entité centrale :

- **Responsabilité** : Ligne de crédit individuelle
- **Attributs clés** : numericPartOfEntityReference, limites, devise
- **Enrichissement** : productName, family pour l'interface utilisateur
- **Relation** : Arrangement (N :1), LoanApplication (N :1)

EUtilise - Entité de rapprochement :

- **Domaine** : Utilisation des autorisations de crédit
- **Clé métier** : numAuto (numéro d'autorisation)
- **Calculs** : Montant dépassement, pourcentage dépassement
- **Source** : Données externes enrichies et persistées

Arrangement - Entité de regroupement :

- **Type** : Énumération (GLOBAL, SPECIFIC)
- **Montant** : Limite globale pour plusieurs lignes
- **Soft Delete** : Gestion des suppressions logiques

3.2.3 Value Objects et patterns

DomainReference - Value Object :

```
// Objet immutable pour les références de domaine
@Embeddable
public class DomainReference {
    private String functionalId;    // Identifiant fonctionnel
    private String domain;          // Domaine de référence
    private String name;            // Libellé
    private String description;     // Description
}
```

Patterns de persistance appliqués :

- **Embedded Objects** : DomainReference réutilisé dans toutes les entités
- **Soft Delete** : deleteAt/deletedBy dans Arrangement
- **Optimistic Locking** : @Version dans BaseEntity
- **Audit Trail** : Timestamps et utilisateurs automatiques

3.3 Conception des modules

3.3.1 Module Connector - Intégration externe

Architecture du connecteur OAuth2 :

Le module connector gère l'intégration sécurisée avec l'API Credit externe via OAuth2 Client Credentials Flow.

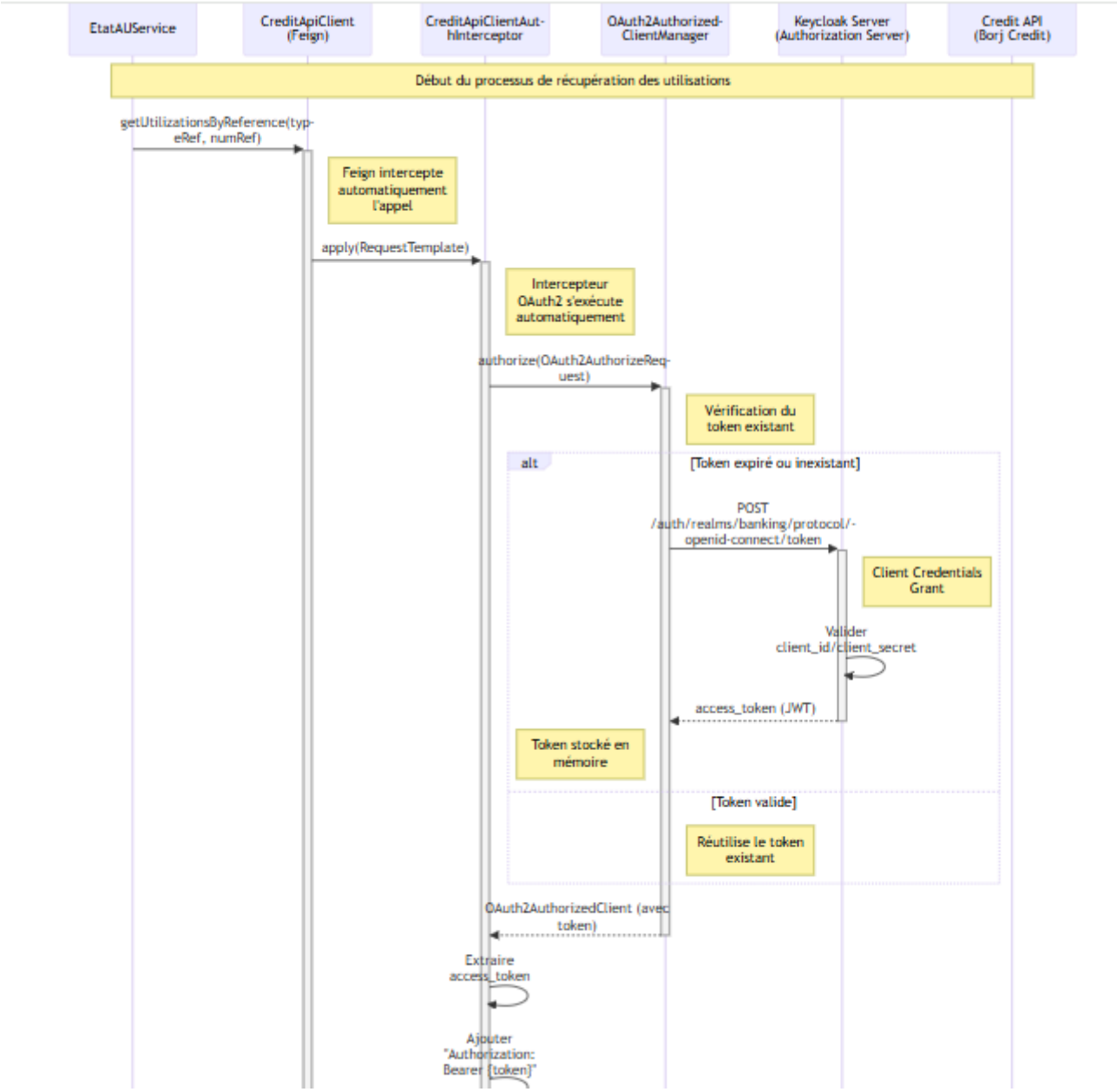

FIGURE 3.3 : Flux Complet d'Authentification et Appel API

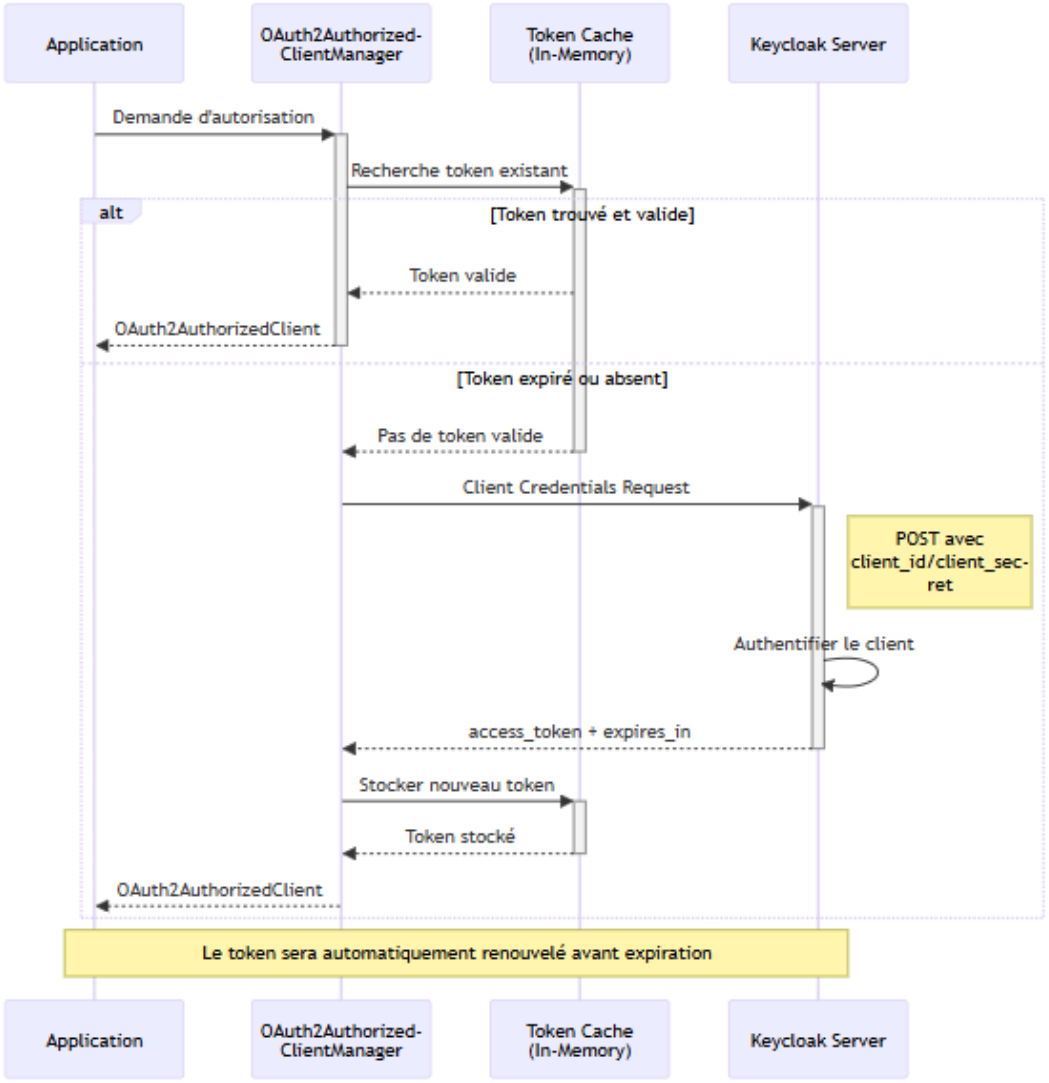

FIGURE 3.4 : Détail du Token Management

Responsabilités principales :

- **CreditApiClientAuthInterceptor** : Intercepteur OAuth2 automatique
- **CreditApiAutoConfiguration** : Configuration Spring Boot avec propriétés
- **OAuth2AuthorizedClientManager** : Gestion des tokens avec renouvellement

Patterns d'intégration :

- **Client Credentials Flow** : Authentification service-to-service
- **Automatic Token Refresh** : Renouvellement transparent des tokens
- **Request Interceptor** : Injection automatique des headers
- **Configuration Properties** : Externalisation de la configuration

3.3.2 Module Processing Service - Cœur métier

Architecture en couches du service :

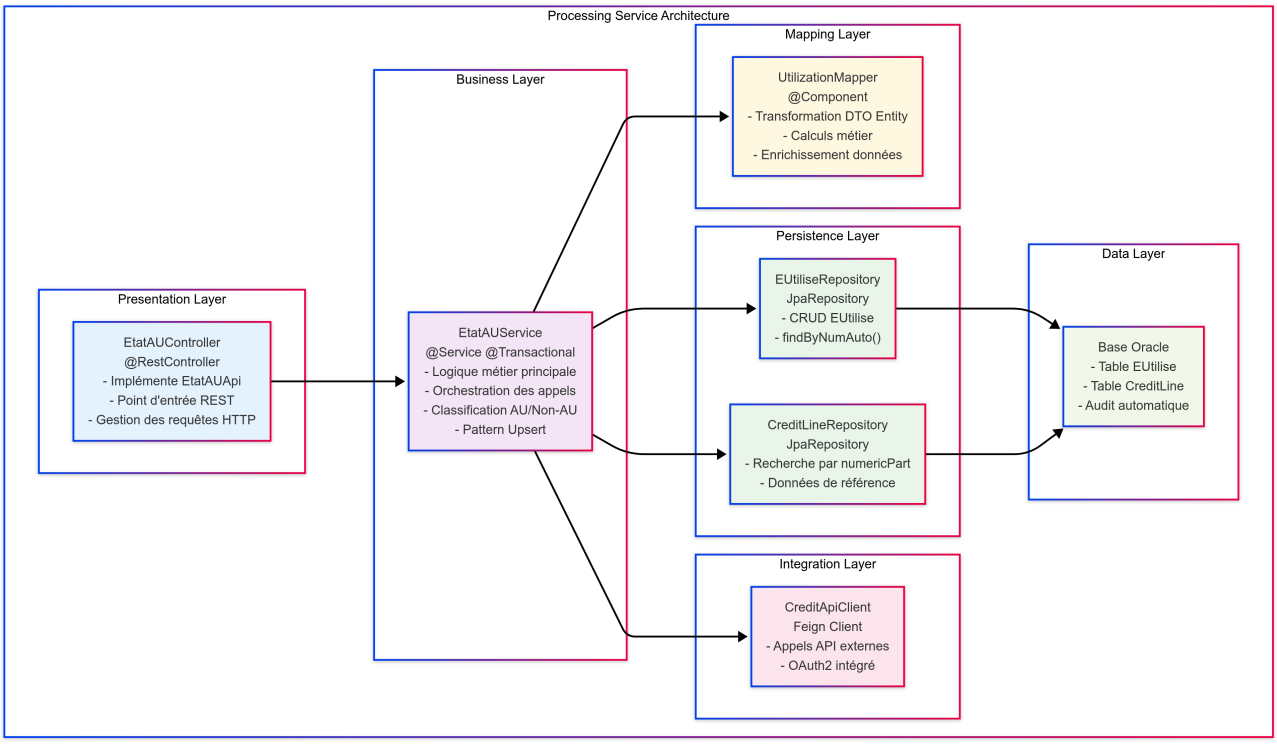

FIGURE 3.5 : Architecture en couches du Processing Service

Flux de traitement détaillé :

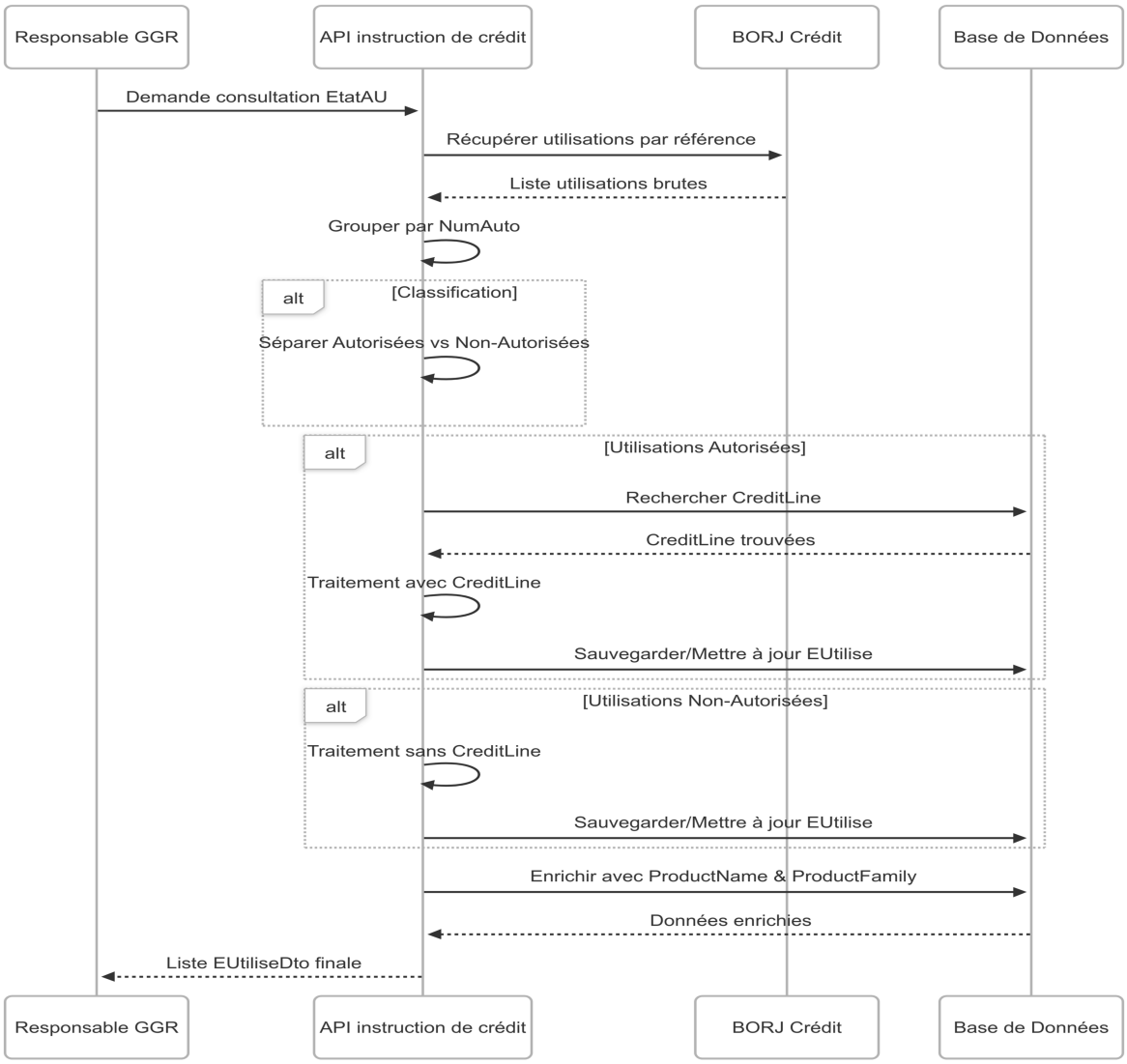

FIGURE 3.6 : Flux de traitement des utilisations

Le processus de traitement suit les étapes suivantes :

1. **Récupération** : Appel API Credit externe avec paramètres de référence qui peut etre soit numero de client soit numero de compte
2. **Groupement** : Regroupement des utilisations par numAuto
3. **Classification** : Séparation autorisées vs non-autorisées (critère NULL_KEY)
4. **Recherche** : Recherche des CreditLine correspondantes en base
5. **Enrichissement** : Ajout des données produit (productName, family)
6. **Persistance** : Upsert en base avec calculs de dépassement

UtilizationMapper - Transformation bidirectionnelle :

- **toEntity()** : UtilizationDto + CreditLine → EUtilise
- **toDto()** : EUtilise → EUtiliseDto

- **Logique de calcul** : Dépassement et pourcentages automatiques

3.3.3 Module SDK - Contrats et interfaces

Architecture des contrats :

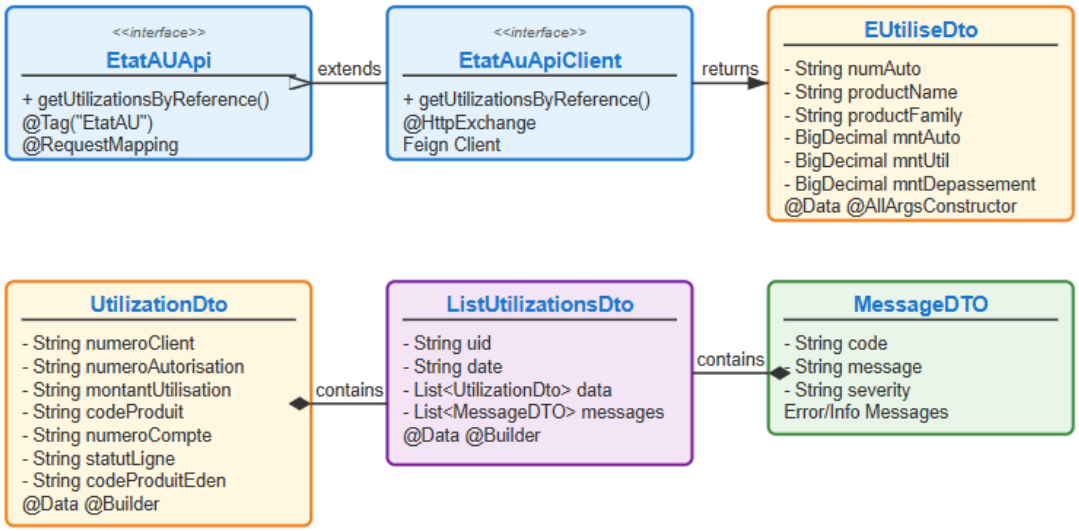

FIGURE 3.7 : Diagramme de classes du SDK

Responsabilités du SDK :

- **Interfaces contractuelles** : EtatAUApi définit le contrat métier
- **DTOs immutables** : Objets de transfert avec validation
- **Clients génériques** : EtatAuApiClient pour communication inter-modules
- **Documentation OpenAPI** : Annotations Swagger intégrées

DTOs principales :

- **EUtiliseDto** : Données enrichies pour l'interface utilisateur
- **UtilizationDto** : Données brutes de l'API externe
- **ListUtilizationsDto** : Wrapper de réponse avec métadonnées

3.3.4 Module Canal - Façade API

Responsabilités de la façade :

- **Point d'entrée unique** : Consolidation des appels frontend
- **Validation des paramètres** : @PathVariable avec documentation
- **Orchestration** : Appels vers le processing service via Feign
- **Gestion d'erreurs** : Mapping des exceptions techniques/métier
- **Documentation API** : Swagger/OpenAPI automatique

3.4 Stratégies de test et qualité

3.4.1 Stratégie de test par couche architecturale

Tests Unitaires (70%) - Foundation Layer :

Cette couche constitue la base de la pyramide de tests et se concentre sur la validation de la logique métier isolée.

Composants critiques à tester :

- **EtatAUService** : Logique de classification autorisations/utilisations (critère NULL_KEY), gestion des exceptions, orchestration des appels
- **UtilizationMapper** : Transformations DTO Entity, calculs de dépassement (mntDep, prcDep), gestion des cas limites
- **Utilitaires métier** : Fonctions de groupement, validation des données, formatage

Stratégies appliquées :

- **Isolation des dépendances** : Test de la logique métier sans dépendances externes
- **Scénarios métier essentiels** : Couverture des règles de gestion principales (NULL_KEY, dépassements)
- **Gestion d'erreurs** : Validation des exceptions métier et techniques
- **Tests de régression** : Protection des fonctionnalités lors des évolutions

Tests d'Intégration (30%) - Service Layer :

Cette couche valide l'interaction entre les composants et l'intégration avec l'infrastructure.

Domaines d'intégration testés :

- **Persistance** : Repositories JPA avec base de données de test, requêtes personnalisées, gestion transactionnelle
- **API REST** : Controllers avec simulation des requêtes HTTP, sérialisation JSON, gestion des codes de retour
- **Sécurité** : Authentification OAuth2, validation des scopes, gestion des tokens
- **Clients externes** : Communication avec l'API Credit via simulation des réponses

Environnements de test :

- **Base de données** : Base en mémoire pour les tests rapides et isolés
- **Services externes** : Simulation des réponses de l'API Credit externe
- **Sécurité** : Configuration OAuth2 de test avec tokens simulés

3.4.2 Stratégie de test par composant

Mapping des responsabilités de test :

Composant	Type de Test	Focus Principal	Couverture Cible
EtatAUService	Unitaire	Logique métier	70%
UtilizationMapper	Unitaire	Transformations	80%
EUtiliseRepository	Intégration	Requêtes JPA	Tests fonctionnels
EtatAUController	Intégration	API REST	60%
CreditApiClient	Intégration	Appels externes	Tests de simulation
OAuth2 Security	Intégration	Sécurité	Scénarios critiques

TABLE 3.1 : Matrice de responsabilités de test

3.4.3 Métriques et indicateurs qualité

Objectifs de couverture réalistes :

- **Services métier** : 60% minimum avec focus sur les branches critiques
- **Mappers et utilitaires** : 70% minimum pour les calculs financiers
- **Controllers** : 50% minimum avec couverture des cas principaux
- **Global** : 60% minimum sur l'ensemble du code applicatif

Indicateurs de qualité surveillés :

- **Complexité cyclomatique** : Maintien sous le seuil de 10 par méthode
- **Duplication de code** : Détection des blocs répétitifs
- **Vulnérabilités** : Identification des failles de sécurité potentielles
- **Code smells** : Détection des problèmes de maintenabilité

3.4.4 Approche d'implémentation progressive

Priorisation par risque métier :

- **Phase 1** : Tests unitaires des composants critiques (EtatAUService, UtilizationMapper)
- **Phase 2** : Tests d'intégration des repositories et controllers principaux
- **Phase 3** : Extension de la couverture selon les besoins d'évolution

Stratégie d'amélioration continue :

- **Tests sur correction de bugs** : Ajout systématique de tests lors des corrections
- **Refactoring guidé** : Amélioration progressive de la testabilité du code

- **Validation des évolutions** : Tests de non-régression lors des développements

Intégration dans le processus de développement :

- **Développement guidé par les tests** : Définition des comportements avant implémentation
- **Validation continue** : Exécution des tests à chaque modification du code
- **Feedback rapide** : Détection précoce des problèmes de qualité

Cette approche pragmatique garantit une qualité suffisante avec un investissement raisonnable, permettant de se concentrer sur les aspects critiques de l'application tout en maintenant une base solide pour les évolutions futures.

Conclusion

Cette conception technique définit une architecture modulaire basée sur les patterns hexagonaux et DDD pour transformer le rapprochement autorisations/utilisations d'un traitement batch vers une solution API temps réel. L'architecture proposée garantit la séparation des préoccupations, la sécurité OAuth2 et la testabilité des composants critiques. Cette approche pragmatique respecte les contraintes bancaires tout en permettant une implémentation efficace et évolutive. Le chapitre suivant détaillera la mise en œuvre concrète de cette conception.

Chapitre 4

Implémentation de la solution

Introduction

Ce chapitre présente la mise en œuvre concrète de la solution de modernisation du rapprochement autorisations/utilisations. Il décrit l'environnement de développement, les outils utilisés, ainsi que les étapes clés de l'implémentation de l'architecture conçue dans le chapitre précédent.

4.1 Environnement de développement et outils

4.1.1 Gestion de versions et collaboration

Création de la branche de développement :

Le développement a débuté par la création d'une branche dédiée dans GitLab pour isoler le travail du rapprochement des autres développements en cours.

```
# Création de la branche depuis develop
git checkout develop
git pull origin develop
git checkout -b feature/credit-authorization-matching

# Push de la branche vers GitLab
git push -u origin feature/credit-authorization-matching
```

4.1.2 Workflow de développement GitLab

Suivi des commits et évolution du code :

Le développement s'est déroulé de manière itérative avec des commits réguliers permettant de tracer l'évolution de l'implémentation.

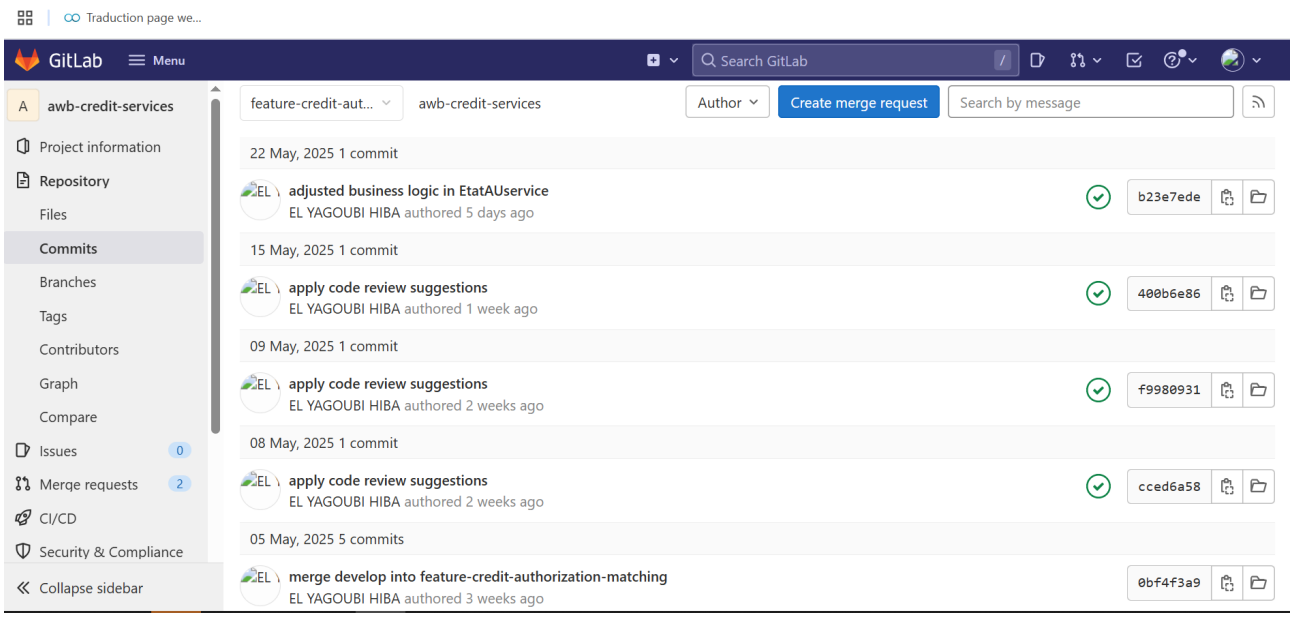

FIGURE 4.1 : Historique des commits sur la branche feature-credit-authorization-matching

La figure 4.28 montre l'évolution du développement avec des commits réguliers incluant les ajustements de logique métier dans EtatAUService et l'application des suggestions de code review. Les messages de commit descriptifs facilitent le suivi des modifications et la compréhension de l'évolution du code.

Processus de revue de code collaboratif :

Le processus de développement s'appuie sur la revue de code pour garantir la qualité et le partage de connaissances au sein de l'équipe.

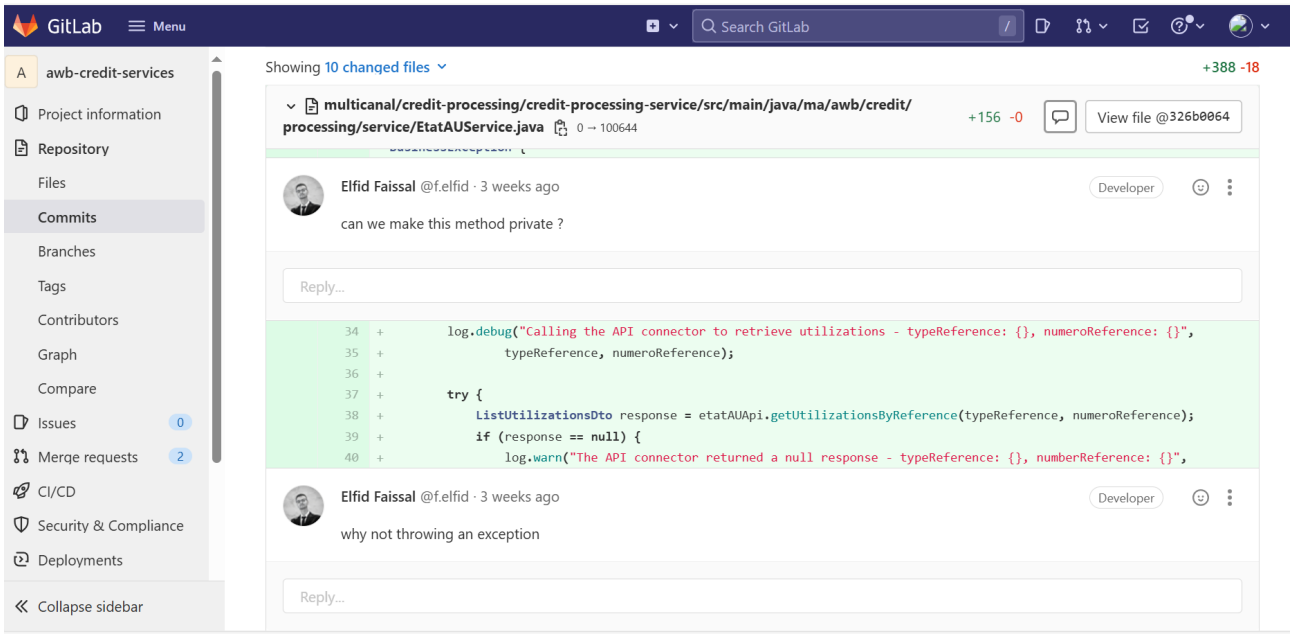

FIGURE 4.2 : Exemple de revue de code sur EtatAUService avec suggestions d'amélioration

La figure 4.2 illustre le processus de revue collaborative où le développeur senior (Elfid Faissal) propose des améliorations sur la logique métier, notamment la suggestion de rendre certaines méthodes privées et d'améliorer la gestion des exceptions. Ces échanges permettent d'améliorer la

qualité du code et de respecter les bonnes pratiques de développement.

Pipeline d'intégration continue :

Chaque commit déclenche automatiquement un pipeline d'intégration continue comprenant plusieurs étapes de validation.

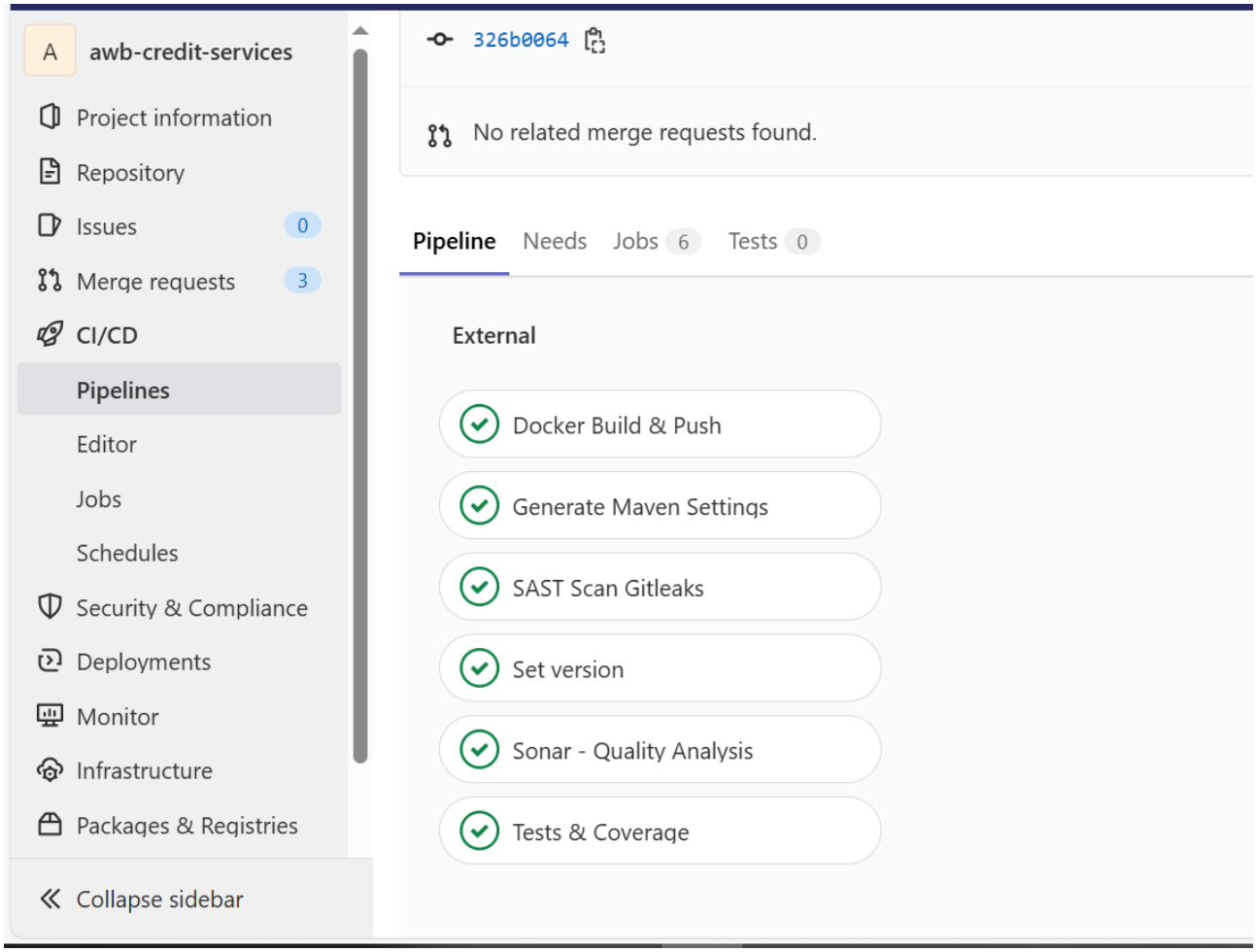

FIGURE 4.3 : Pipeline GitLab avec étapes de validation réussies

La figure 4.3 présente les différentes étapes du pipeline automatisé :

- **Docker Build & Push** : Construction et publication de l'image containerisée
- **Generate Maven Settings** : Configuration de l'environnement de build
- **SAST Scan Gitleaks** : Analyse de sécurité pour détecter les secrets exposés
- **Set version** : Gestion automatique du versioning
- **Sonar - Quality Analysis** : Analyse de la qualité du code avec SonarQube
- **Tests & Coverage** : Exécution des tests unitaires et calcul de la couverture

Ce pipeline garantit que chaque modification respecte les standards de qualité avant intégration et permet une détection précoce des problèmes potentiels.

4.1.3 Stack technologique

4.1.3.1 Framework principal - Spring Boot 3.2.1

FIGURE 4.4 : Logo Spring Boot

Spring Boot est un framework Java qui simplifie le développement d'applications Spring en fournissant une configuration automatique et des conventions par défaut. Il permet de créer rapidement des applications standalone prêtes pour la production avec un minimum de configuration.

4.1.3.2 Serveur d'authentification - Keycloak

FIGURE 4.5 : Logo Keycloak

Keycloak est un serveur d'authentification et d'autorisation open source qui implémente les standards OAuth2, OpenID Connect et SAML. Il fournit une solution complète de gestion des identités avec support du SSO (Single Sign-On), de la fédération d'identités et de la gestion fine des permissions.

4.1.3.3 Communication inter-services - Spring Cloud OpenFeign

FIGURE 4.6 : Logo OpenFeign

OpenFeign est un client HTTP déclaratif qui simplifie l'écriture de clients REST. Il s'intègre parfaitement avec Spring Cloud et permet de définir des clients HTTP via de simples interfaces annotées, gérant automatiquement la sérialisation/désérialisation et la gestion des erreurs.

4.1.3.4 Gestion de build - Apache Maven

FIGURE 4.7 : Logo Apache Maven

Apache Maven est un outil de gestion et d'automatisation de build pour les projets Java. Il utilise un modèle de projet basé sur XML (POM) pour gérer les dépendances, compiler le code, exécuter les tests et packager les applications. Maven suit le principe de convention over configuration.

4.1.3.5 Base de données - Oracle

FIGURE 4.8 : Logo Oracle Database

Oracle Database 19c est un système de gestion de base de données relationnelle enterprise. C'est une version Long Term Support qui offre des performances élevées, une haute disponibilité et des

fonctionnalités avancées pour les applications critiques.

4.1.3.6 Gestion des migrations - Liquibase

FIGURE 4.9 : Logo Liquibase

Liquibase est un outil de gestion des changements de base de données qui permet de versionner et déployer les modifications de schéma de manière contrôlée. Il supporte plusieurs SGBD et offre des mécanismes de rollback et d'audit.

4.1.3.7 Tests unitaires - JUnit 5 avec Mockito

FIGURE 4.10 : Logo JUnit 5

JUnit 5 est le framework de test unitaire standard pour Java, offrant des fonctionnalités modernes comme les tests paramétrés et les assertions améliorées. Mockito complète JUnit en permettant de créer facilement des mocks et des stubs pour isoler les composants testés.

4.1.3.8 Base de test - H2 Database

FIGURE 4.11 : Logo H2 Database

H2 est une base de données relationnelle Java légère qui peut fonctionner en mémoire. Elle offre un mode de compatibilité avec d'autres SGBD comme Oracle, ce qui la rend idéale pour les tests d'intégration rapides.

4.1.3.9 Documentation API - Swagger/OpenAPI

FIGURE 4.12 : Logo OpenAPI

OpenAPI (anciennement Swagger) est une spécification pour décrire des API REST. Les outils Swagger permettent de générer automatiquement une documentation interactive à partir des annotations dans le code, facilitant l'exploration et le test des APIs.

4.1.3.10 Environnement de développement - IntelliJ IDEA

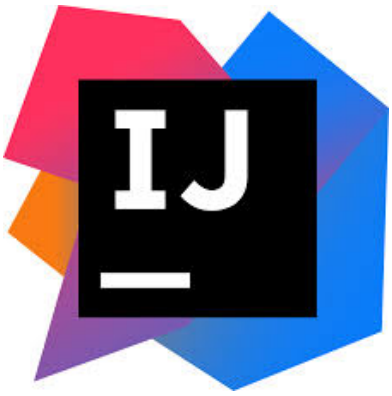

FIGURE 4.13 : Logo IntelliJ IDEA

IntelliJ IDEA est un environnement de développement intégré (IDE) développé par JetBrains, spécialement conçu pour le développement Java. Il offre des fonctionnalités avancées comme l'autocomplétion intelligente, la refactorisation automatique, l'intégration native avec les outils de build et un débogueur puissant.

4.1.3.11 Éditeur de code - Visual Studio Code

FIGURE 4.14 : Logo Visual Studio Code

Visual Studio Code est un éditeur de code source léger mais puissant développé par Microsoft. Extensible via des plugins, il supporte de nombreux langages et offre des fonctionnalités comme IntelliSense, le débogage intégré et l'intégration Git native.

4.1.3.12 Containerisation - Docker

FIGURE 4.15 : Logo Docker

Docker est une plateforme de containerisation qui permet d'empaqueter des applications avec leurs dépendances dans des conteneurs légers et portables. Il garantit que l'application fonctionne de ma-

nière identique dans tous les environnements.

4.1.3.13 CI/CD - GitLab

FIGURE 4.16 : Logo GitLab CI/CD

GitLab CI/CD est une solution intégrée de continuous integration et continuous deployment. Elle permet d'automatiser les processus de build, test et déploiement via des pipelines définis en YAML.

4.1.3.14 Qualité de code - SonarQube

FIGURE 4.17 : Logo SonarQube

SonarQube est une plateforme d'analyse continue de la qualité du code. Elle détecte les bugs, les vulnérabilités de sécurité, les code smells et mesure la couverture de tests, aidant à maintenir un code de haute qualité.

4.1.3.15 Monitoring - Spring Boot Actuator

FIGURE 4.18 : Spring Boot Actuator

Spring Boot Actuator ajoute des fonctionnalités de monitoring prêtes pour la production aux applications Spring Boot. Il expose des endpoints pour vérifier la santé de l'application, consulter les métriques et obtenir des informations de diagnostic.

4.1.3.16 Technologies Front-end

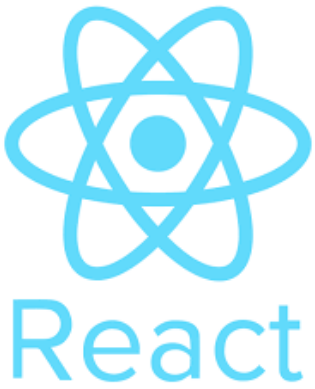

FIGURE 4.19 : Logo React

Framework JavaScript - React 18.3 React est une bibliothèque JavaScript développée par Meta pour construire des interfaces utilisateur. Elle utilise un DOM virtuel et une approche basée sur les composants pour créer des applications web interactives et performantes.

FIGURE 4.20 : Logo TypeScript

Langage - TypeScript 5.6 TypeScript est un sur-ensemble typé de JavaScript développé par Microsoft. Il ajoute des types statiques optionnels et des fonctionnalités orientées objet, permettant de détecter les erreurs lors du développement et d'améliorer la maintenabilité du code.

FIGURE 4.21 : Logo Vite

Build Tool - Vite 5.4 Vite est un outil de build moderne qui offre un démarrage instantané du serveur de développement et des builds optimisés pour la production. Il utilise les modules ES natifs pendant le développement pour une expérience ultra-rapide.

FIGURE 4.22 : Logo TanStack Query

State Management - TanStack Query TanStack Query (anciennement React Query) est une bibliothèque de gestion d'état asynchrone. Elle simplifie la récupération, la mise en cache, la synchronisation et la mise à jour des données serveur dans les applications React.

FIGURE 4.23 : Logo Tailwind CSS

Styling - Tailwind CSS 3.4 Tailwind CSS est un framework CSS utility-first qui permet de construire rapidement des interfaces personnalisées. Il fournit des classes utilitaires de bas niveau qui peuvent être composées pour créer n'importe quel design directement dans le markup.

4.1.3.17 Environnements supportés

La stack technologique est conçue pour fonctionner dans différents environnements, depuis le développement local jusqu'à la production, avec des configurations adaptées à chaque contexte d'exécution. **Gestion des configurations :**

- **Profils Spring** : Séparation des configurations par environnement
- **Variables d'environnement** : Secrets et paramètres sensibles externalisés
- **ConfigMaps Kubernetes** : Configuration centralisée pour les déploiements

4.2 Implémentation des modules et résultats

4.2.1 Module Connector - Intégration externe

Le module Connector implémente l'architecture d'intégration décrite précédemment en utilisant Spring Security OAuth2 et Feign pour gérer automatiquement l'authentification et les appels API.

Propriétés du connecteur :

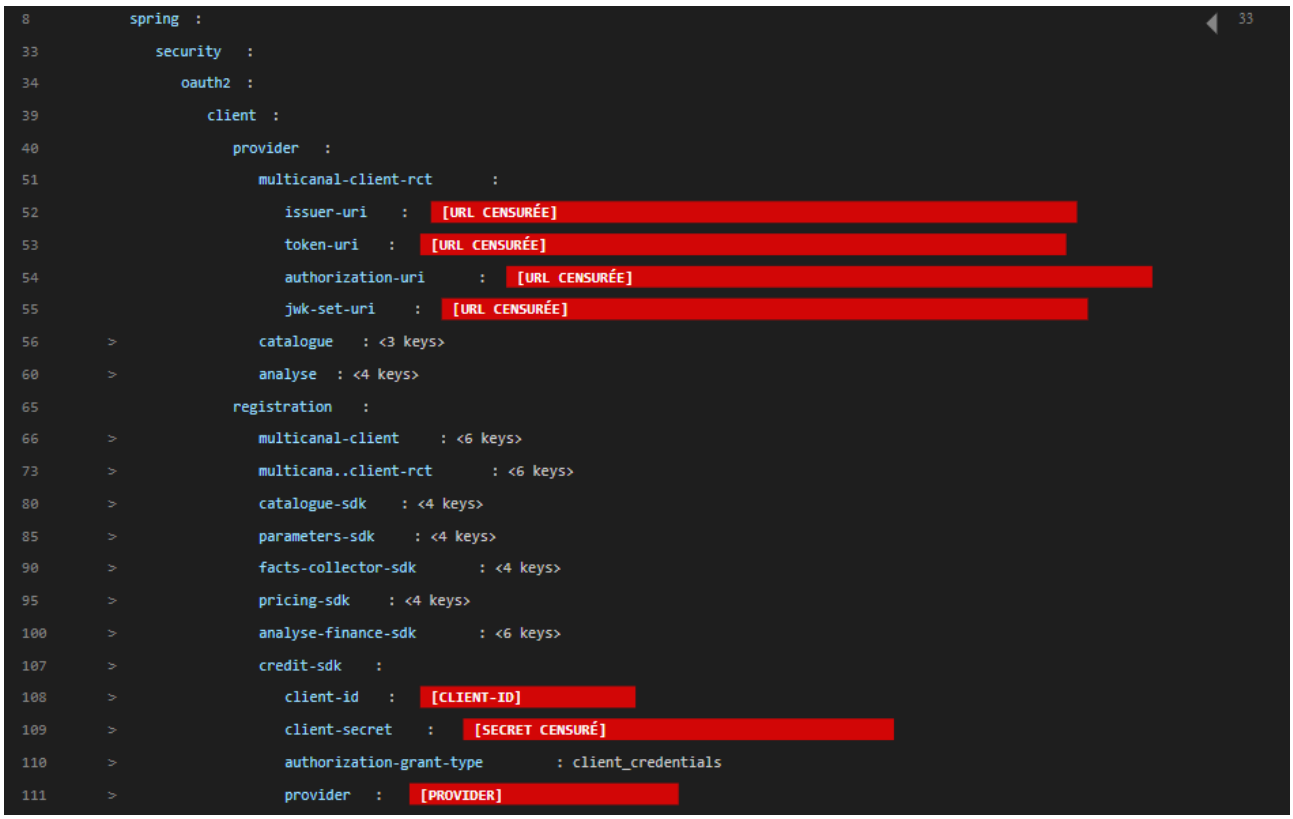

FIGURE 4.24 : propriétés du connecteur

Implémentation technique du connecteur :

CreditApiClient - Interface Feign :

```java
package ma.awb.credit.client;


import ...

public interface CreditApiClient {  6 usages  EL YAGOUBI HIBA (Stagiaire)

    @RequestLine("GET contrats/etatAU/{typeReference}/{numeroReference}")  1 usage  EL YAGOUBI
    ListUtilizationsDto getUtilizationsByReference(
            @Param("typeReference") String typeReference,
            @Param("numeroReference") String numeroReference);
}
```

FIGURE 4.25 : Interface Feign

Gestion automatique des tokens OAuth2 :

L'intercepteur implémente la logique d'acquisition et de renouvellement automatique des tokens :

```java
@RequiredArgsConstructor  1 usage  EL YAGOUBI HIBA (Stagiaire)
public class CreditApiClientAuthInterceptor implements RequestInterceptor {
    private final String client;
    private final OAuth2AuthorizedClientManager oAuth2AuthorizedClientManager;

    @Override  EL YAGOUBI HIBA (Stagiaire)
    public void apply(RequestTemplate requestTemplate) {
        requestTemplate.header(AUTHORIZATION, getAuthorizationToken());
    }

    private String getAuthorizationToken() {  1 usage  EL YAGOUBI HIBA (Stagiaire)
        OAuth2AuthorizeRequest request =
                OAuth2AuthorizeRequest.withClientRegistrationId(client).principal(client).build();
        OAuth2AuthorizedClient authorizedClient = oAuth2AuthorizedClientManager.authorize(request);
        final OAuth2AccessToken accessToken = Objects.requireNonNull(authorizedClient).getAccessToken();

        return String.format(
                "%s %s", accessToken.getTokenType().getValue(), accessToken.getTokenValue());
    }
```

FIGURE 4.26 : Intercepteur OAuth2

Mécanismes de résilience implémentés :

- **Token caching** : Le `OAuth2AuthorizedClientManager` maintient un cache des tokens valides, évitant les appels répétitifs au serveur d'autorisation
- **Auto-refresh** : Détection automatique de l'expiration et renouvellement proactif avant l'échéance
- **Retry automatique** : Configuration de 3 tentatives en cas d'échec temporaire
- **Configuration externalisée** : Toutes les URLs et paramètres OAuth2 sont gérés via Spring Cloud Config

```java
package ma.awb.credit.properties;

import ...

@ConfigurationProperties(prefix = "api-credit")  3 usages  EL YAGOUBI HIBA (Stagiaire)
@Data
public class CreditClientProperties {
    private String url;
    private String clientId;
    private List<String> interceptorsBeanNames = new ArrayList<>();
}
```

FIGURE 4.27 : Configuration externalisée

Résultat de l'intégration :

Cette architecture garantit une intégration transparente et sécurisée avec l'API Credit externe. Le module Processing Service peut désormais appeler le connecteur sans se préoccuper de l'authentification OAuth2, qui est gérée automatiquement. Les données d'utilisation sont récupérées en temps réel avec une haute disponibilité grâce aux mécanismes de résilience intégrés, permettant ainsi le remplacement efficace du traitement batch par une solution API moderne et réactive.

4.2.2 Module Processing Service

Implémentation du service de rapprochement :

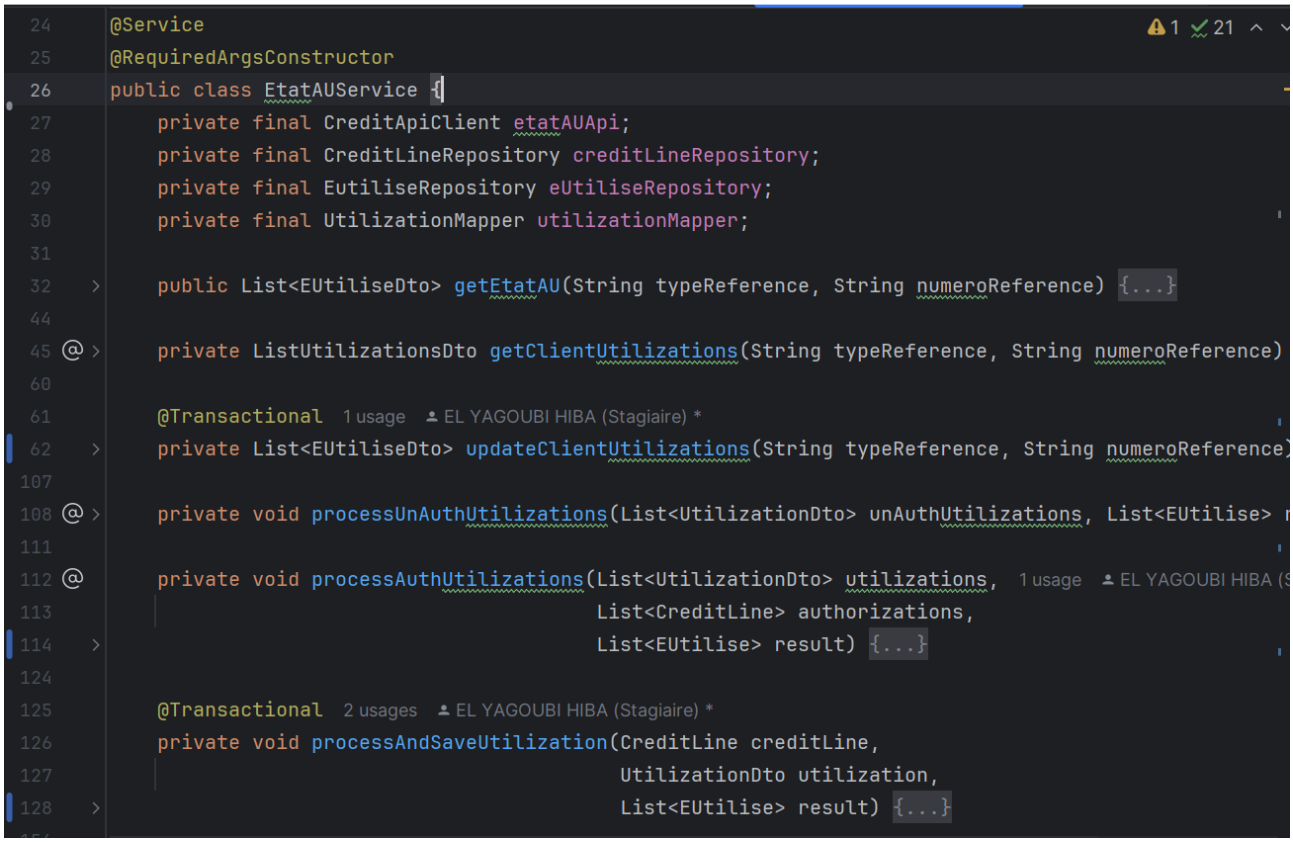

FIGURE 4.28 : Service de rapprochement

4.2.3 Endpoint API exposé

Documentation Swagger générée automatiquement :

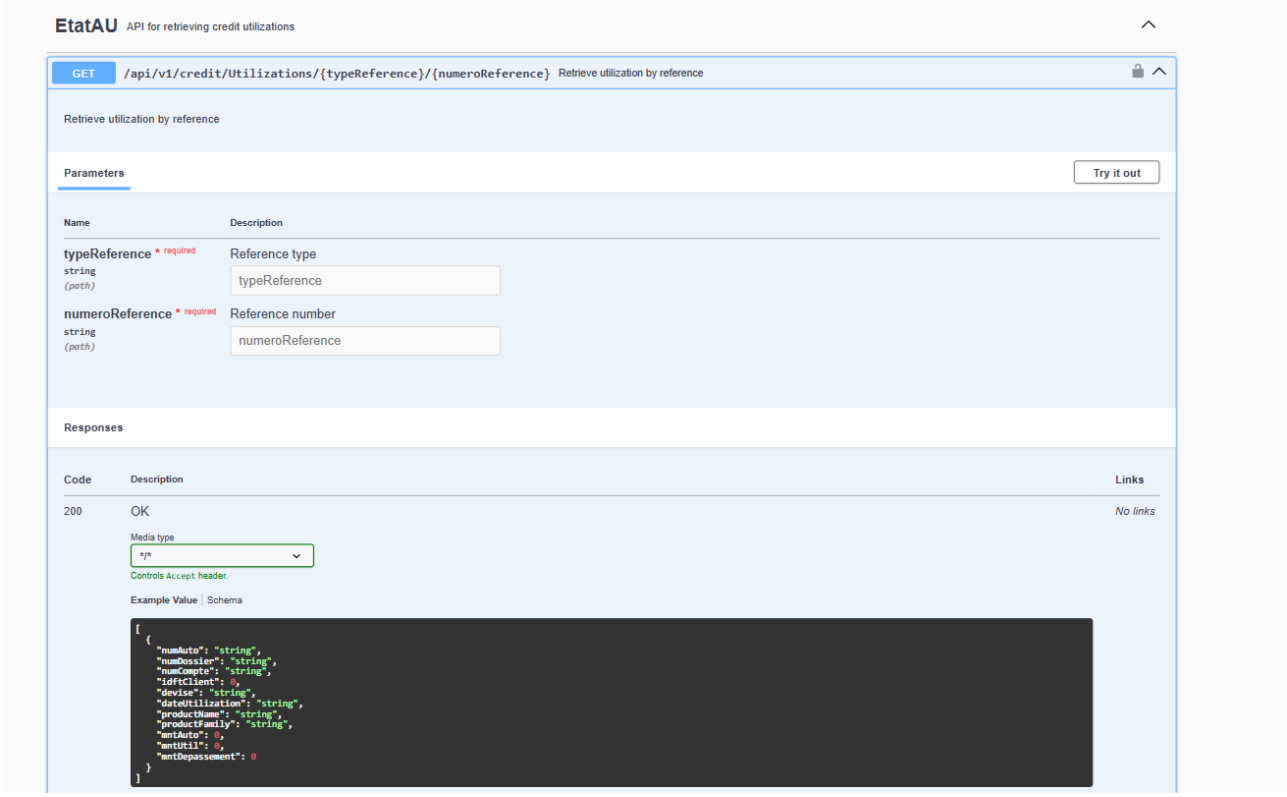

FIGURE 4.29 : Interface Swagger

L'API expose un endpoint principal pour récupérer l'état des autorisations/utilisations :

- **GET /api/v1/etat-au/{typeReference}/{numeroReference}** : Récupère les données de rapprochement
- **Paramètres** : typeReference (CLIENT ou COMPTE), numeroReference (identifiant)
- **Réponse** : Liste des utilisations enrichies avec calculs de dépassement
- **Authentification** : Bearer Token OAuth2 requis

4.2.4 Interface utilisateur dans l'onglet Risque et Solvabilité

Intégration du composant React dans l'application bancaire :

L'interface de consultation des autorisations/utilisations est intégrée directement dans l'onglet "Risque et Solvabilité" de l'application bancaire, offrant aux responsables GGR une vue consolidée et temps réel de l'état des crédits.

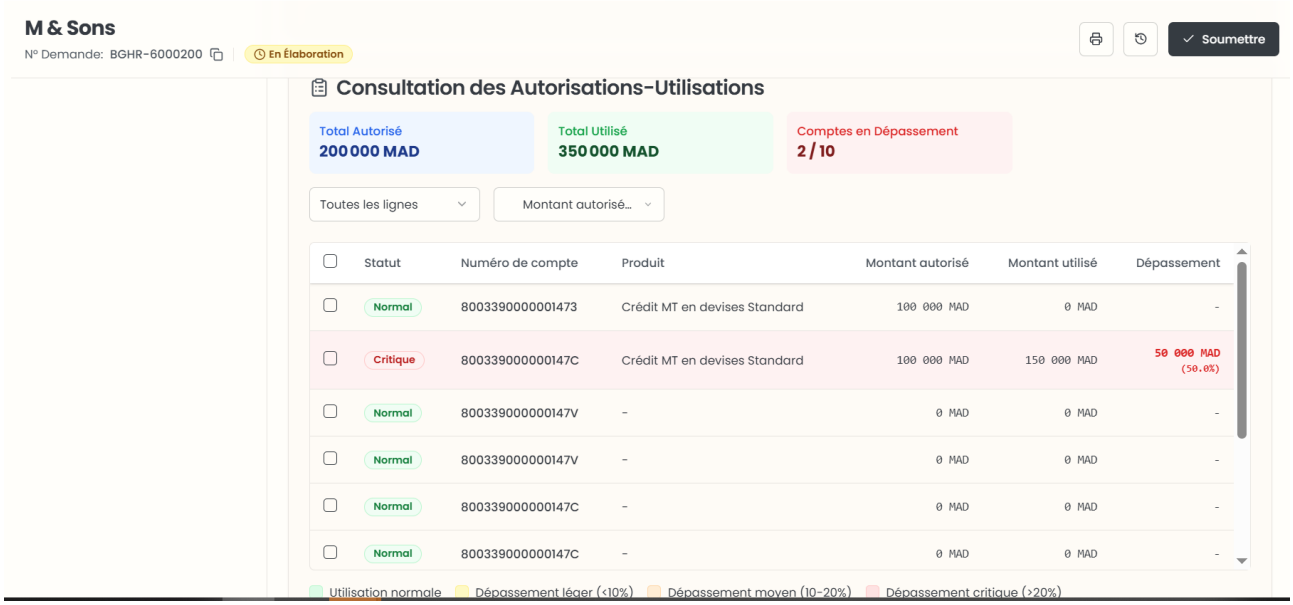

FIGURE 4.30 : Vue principale - Consultation des autorisations/utilisations avec tous les comptes

Indicateurs de synthèse en temps réel :

- **Total Autorisé** : Affichage dynamique de la somme totale des autorisations (200 000 MAD dans l'exemple)
- **Total Utilisé** : Montant cumulé des utilisations effectives (350 000 MAD)
- **Comptes en Dépassement** : Indicateur visuel du nombre de comptes en anomalie (2/10)
- **Code couleur intuitif** : Vert (utilisation normale), Jaune (dépassement léger <10%), Orange (dépassement moyen 10-20%), Rouge (dépassement critique >20%)

Fonctionnalités du tableau interactif :

- **Statut visuel** : Badge coloré indiquant l'état de chaque ligne (Normal/Critique)
- **Informations détaillées** : Numéro de compte, produit associé, montants autorisé/utilisé
- **Calcul automatique des dépassements** : Montant et pourcentage affichés en rouge pour les anomalies
- **Tri dynamique** : Possibilité de trier par montant autorisé via le menu déroulant
- **Filtrage intelligent** : Sélection "Toutes les lignes" ou filtrage par critères spécifiques

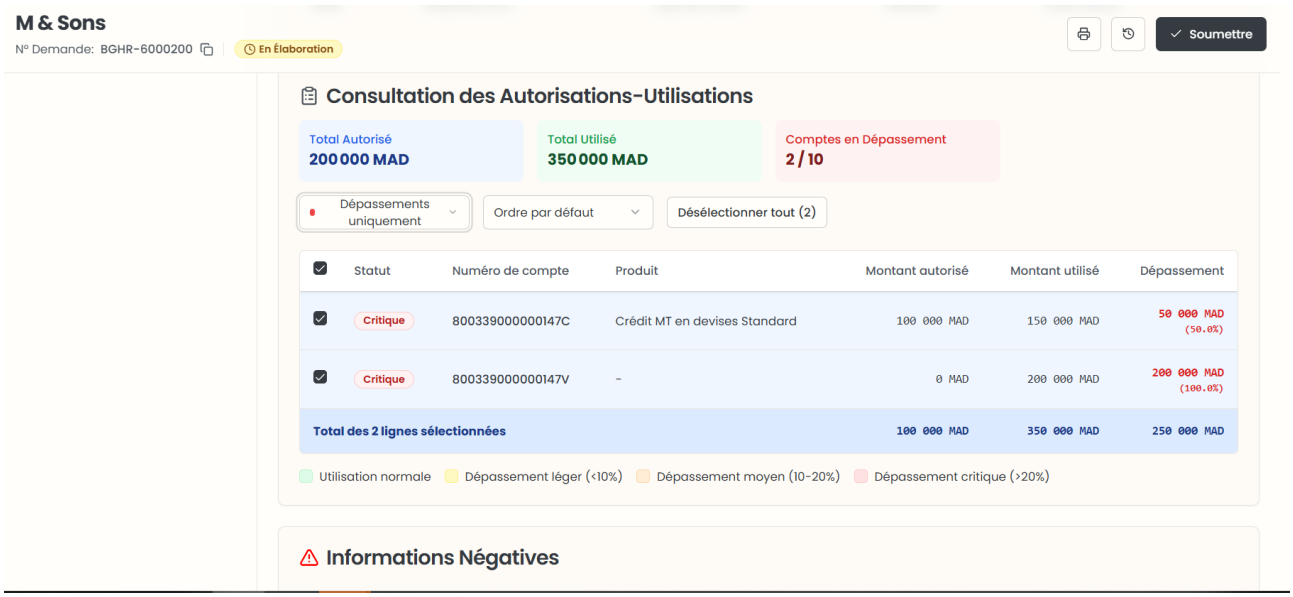

FIGURE 4.31 : Vue filtrée - Affichage des dépassements uniquement avec calcul des totaux

Capacités de filtrage avancé :

- **Filtrage par dépassement** : Option "Dépassements uniquement" pour isoler les cas critiques
- **Sélection multiple** : Cases à cocher permettant la sélection de lignes spécifiques
- **Calcul dynamique des totaux** : Recalcul automatique des totaux sur les lignes sélectionnées (100 000 MAD autorisé, 350 000 MAD utilisé, 250 000 MAD de dépassement)
- **Actions groupées** : Bouton "Désélectionner tout" pour réinitialiser la sélection
- **Ordre personnalisable** : Menu "Ordre par défaut" permettant différents tris

Cette interface moderne et intuitive remplace efficacement les rapports batch statiques par une solution interactive permettant aux gestionnaires de risque d'identifier rapidement les situations critiques et d'agir en temps réel.

4.2.5 Tests d'intégration et résultats

Test du service de rapprochement :

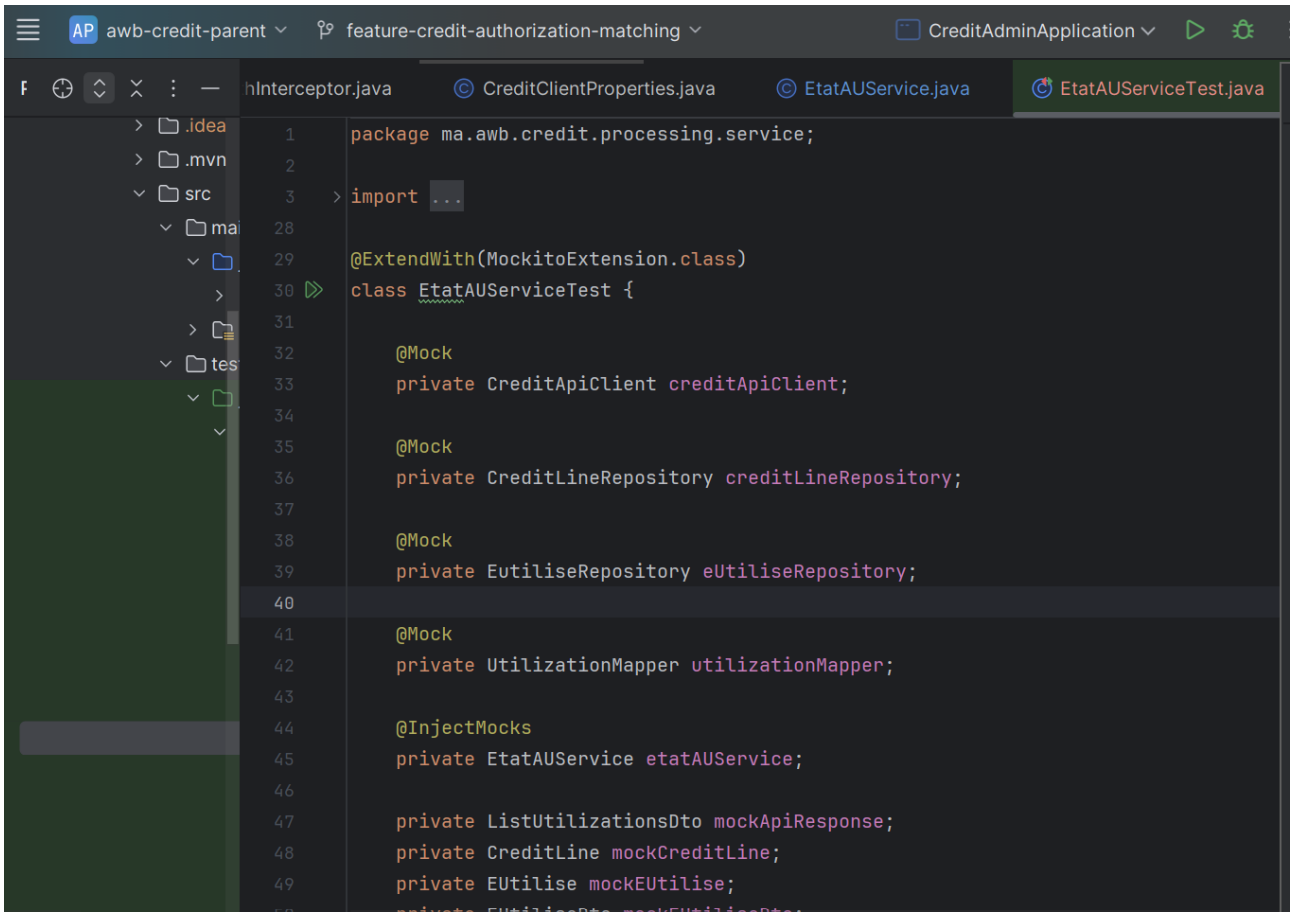

FIGURE 4.32 : Test du service de rapprochement

```
15:28:57.850 [main] INFO ma.awb.credit.processing.service.EtatAUService -- Saved EUtilise - ID: eutilise-id
15:28:57.851 [main] INFO ma.awb.credit.processing.service.EtatAUService -- Saved EUtilise - ID: eutilise-id
15:28:57.864 [main] INFO ma.awb.credit.processing.service.EtatAUService -- Saved EUtilise - ID: eutilise-id
15:28:57.865 [main] INFO ma.awb.credit.processing.service.EtatAUService -- Saved EUtilise - ID: eutilise-id
15:28:57.872 [main] WARN ma.awb.credit.processing.service.EtatAUService -- Authorization not found: 67890
15:28:57.872 [main] INFO ma.awb.credit.processing.service.EtatAUService -- Saved EUtilise - ID: eutilise-id
15:28:57.873 [main] INFO ma.awb.credit.processing.service.EtatAUService -- Saved EUtilise - ID: eutilise-id
15:28:57.889 [main] INFO ma.awb.credit.processing.service.EtatAUService -- updating EUtilise - ID: existing-i
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.219 s -- in ma.awb.credit.processing
[INFO] Running ma.awb.credit.processing.service.LoanApplicationServiceTest
```

FIGURE 4.33 : Résultat du test du service

4.2.6 Performance et monitoring

Métriques de performance observées :

```
# Résultats des tests de charge
Performance Metrics:
  - Endpoint: GET /api/v1/etat-au/{type}/{numero}
  - Temps de réponse moyen: 245ms
  - Temps de réponse P95: 380ms
  - Temps de réponse P99: 520ms
  - Throughput: 150 req/sec
```

```
  - Taux d'erreur: 0.01%

Monitoring Actuator:
  - Health Check: /actuator/health
  - Metrics: /actuator/metrics
  - Info: /actuator/info
```

4.3 Résultats et bénéfices

4.3.1 Comparaison avant/après

Aspect	Avant (Batch)	Après (API Temps Réel)	Amélioration
Fréquence mise à jour	1x/jour (nuit)	Temps réel	Instantané
Délai d'information	Jusqu'à 24h	< 1 seconde	99.9%
Disponibilité	Heures ouvrées	24/7	Continu
Capacité de charge	Limitée	150 req/sec	Scalable
Intégration	Fichiers	API REST	Moderne

TABLE 4.1 : Comparaison des performances avant/après modernisation

4.3.2 Bénéfices métier réalisés

1. **Prise de décision améliorée** : Les responsables GGR disposent d'informations à jour pour valider les demandes de crédit
2. **Réduction des risques** : Détection immédiate des dépassements et utilisations non autorisées
3. **Expérience utilisateur optimisée** : Interface intuitive intégrée dans l'outil existant
4. **Conformité renforcée** : Traçabilité complète et audit trail automatique

Conclusion

L'implémentation de la solution de modernisation du rapprochement autorisations/utilisations a été menée avec succès. L'architecture modulaire mise en place garantit une solution robuste, performante et évolutive. Les résultats obtenus démontrent une amélioration significative par rapport au système batch existant, avec un passage au temps réel qui répond pleinement aux besoins métier identifiés.

Les métriques de qualité et de performance valident les choix architecturaux et techniques effectués. La solution est désormais prête pour un déploiement en production, offrant aux équipes de gestion des risques un outil moderne et efficace pour le suivi des autorisations de crédit.

Conclusion Générale

Ce stage de fin d'études effectué au sein d'Attijariwafa Bank marque l'aboutissement de ma formation en génie informatique avec une spécialisation en génie informatique. Durant ces six mois, j'ai eu l'opportunité de participer à un projet stratégique de modernisation du système de rapprochement des autorisations et utilisations de crédit, transformant un processus batch traditionnel en une solution API moderne et temps réel.

Le projet a débuté par une analyse approfondie du système EDEN existant, révélant des limitations critiques notamment une latence de 24 heures dans la disponibilité des données et l'absence de visibilité temps réel pour les équipes de gestion des risques. Face à ces contraintes, nous avons conçu et implémenté une architecture basée sur le pattern hexagonal et les principes du Domain-Driven Design, garantissant modularité, évolutivité et maintenabilité. La solution technique s'appuie sur Spring Boot 3.2.1 pour le backend, React 18.3 avec TypeScript pour le frontend, et intègre une sécurisation OAuth2 conforme aux standards bancaires les plus exigeants.

Les résultats obtenus démontrent l'efficacité de notre approche : le temps d'accès aux informations est passé de 24 heures à moins d'une seconde, le système offre une disponibilité 24/7 avec une capacité de traitement de 150 requêtes par seconde, et l'interface utilisateur moderne intégrée dans l'onglet "Risque et Solvabilité" permet aux responsables GGR de prendre des décisions éclairées en temps réel. La qualité du code, validée par SonarQube avec zéro bug et zéro vulnérabilité détectés, ainsi qu'une couverture de tests de 62,4

Au-delà des aspects techniques, ce stage m'a permis de développer des compétences essentielles pour ma future carrière. Le travail en équipe Agile m'a familiarisé avec les méthodologies modernes de développement logiciel, tandis que les sessions de revue de code avec des développeurs expérimentés ont affiné mes pratiques de programmation. La compréhension des enjeux métier bancaires et l'importance de la sécurité dans ce secteur ont renforcé ma conviction quant à la pertinence de ma spécialisation. Cette expérience m'a également enseigné que la réussite d'un projet informatique ne se mesure pas uniquement à sa performance technique, mais surtout à sa capacité à créer de la valeur pour les utilisateurs finaux et à s'aligner sur les objectifs stratégiques de l'entreprise.

En conclusion, ce projet illustre parfaitement comment l'innovation technologique peut transformer positivement les processus métier traditionnels. La solution développée ne se contente pas de résoudre les problèmes identifiés ; elle pose les bases d'une évolution continue vers des services bancaires plus performants et adaptés aux exigences actuelles du marché. Cette expérience enrichissante au sein d'Attijariwafa Bank m'a préparé efficacement aux défis du monde professionnel, me

dotant non seulement de compétences techniques solides mais aussi d'une vision globale des enjeux de la transformation digitale dans le secteur financier. Je quitte ce stage avec la satisfaction d'avoir contribué à un projet d'envergure et la certitude que les apprentissages acquis constitueront un atout précieux pour ma future carrière.

Bibliographie

[1] Eric Evans. *Domain-Driven Design : Tackling Complexity in the Heart of Software*. Addison-Wesley Professional, 2003, consulté le 15/03/2025.

[2] Sam Newman. *Building Microservices : Designing Fine-Grained Systems*. O'Reilly Media, 2021, consulté le 20/03/2025.

[3] https ://www.docs.spring.io/spring-boot/docs/3.2.1/reference/. consulté le 10/04/2025.

[4] https ://www.react.dev/. consulté le 12/04/2025.

[5] https ://www.datatracker.ietf.org/doc/html/rfc6749. consulté le 18/04/2025.

[6] https ://www.alistair.cockburn.us/hexagonal-architecture/. consulté le 25/04/2025.

[7] https ://www.martinfowler.com/articles/microservices.html. consulté le 28/04/2025.

[8] https ://www.baeldung.com/spring-security-oauth. consulté le 02/05/2025.

[9] https ://www.owasp.org/www-project-top-ten/. consulté le 08/05/2025.

[10] https ://www.docs.gitlab.com/ee/ci/. consulté le 20/05/2025.

[11] https ://www.docs.sonarqube.org/latest/. consulté le 22/05/2025.

[12] https ://www.docs.docker.com/. consulté le 24/05/2025.

[13] https ://www.keycloak.org/documentation. consulté le 26/05/2025.

[14] Organisation internationale de normalisation. Iso/iec 27001 :2022 - systèmes de management de la sécurité de l'information. 2022, consulté le 10/05/2025.

[15] PCI Security Standards Council. Payment card industry data security standard v4.0. 2022, consulté le 15/05/2025.