**ABSTRACT**

This work was carried out at the company VERMEG as part of the final project for obtaining a national engineering degree in software engineering at the EPI Digital School of Sousse for the academic year 2023/2024.

Our project involves creating a dedicated software platform for automanaging client emails, developed using Angular for the front end, Spring Boot for the back end, and Kafka for asynchronous communication.

\*\***Keywords**\*\*: Development, Angular, Spring Boot, API, JSON, Kafka, Microservice, MVC, JavaScript.

**RESUME**

Le présent travail a été effectué au sein de l’entreprise VERMEG dans le cadre du projet de fin d’études pour l’obtention d’un diplôme national d’ingénieur en informatique spécialité génie logiciel à l’EPI Digital School de Sousse pour l’année académique 2023/2024.

Notre projet consiste à réaliser une plate-forme logicielle dédiée à la gestion automatique des emails clients, développée en utilisant la technologie Angular pour le front end, Spring Boot pour le back end, et Kafka pour la communication asynchrone.

**\*\*Mots clés\*\***: Développement, Angular, Spring Boot, API, JSON, Kafka, Microservice, MVC, JavaScript.

**DEDICACES**

Je dédie ce travail à ma mère, dont l'amour inconditionnel, la patience infinie et le soutien indéfectible ont été une source constante de motivation et d'inspiration. Son exemple de force et de dévouement m'a guidée et soutenue tout au long de ce projet.

À mes autres frères et sœurs, merci pour votre amour et votre soutien continus.

À tous mes amis et tous ceux qui me sont chers, merci pour votre compréhension, vos encouragements et votre soutien. Vous avez tous contribué, d'une manière ou d'une autre, à la réalisation de ce projet.

Enfin, je tiens à exprimer ma gratitude à toutes les personnes qui ont, de près ou de loin, contribué à la réussite de ce travail.

**REMERCIEMENTS**

En premier lieu, je tiens à exprimer mes sincères remerciements à mon encadrant, Monsieur **Ben Amor Sami**, mon encadreur pédagogique, pour son soutien constant et ses conseils pertinents qui ont enrichi mon expérience tout au long de ce projet.

Je tiens à exprimer ma profonde gratitude envers mes **professeurs** pour leur précieuse formation tout au long de mon parcours. Leurs enseignements et leur soutien ont joué un rôle crucial dans mon développement académique et professionnel.

Enfin, je tiens à adresser mes sincères remerciements aux **membres du jury** pour avoir accepté d'évaluer mon travail. Leur temps et leur expertise sont grandement appréciés.

Avec toute ma reconnaissance,

**Table des matières**

[Introduction générale 1](#_Toc170539812)

[Chapitre 1 : Présentation générale 4](#_Toc170539813)

[1.1. Introduction 4](#_Toc170539814)

[1.2. Présentation de la société d’accueil 4](#_Toc170539815)

[1.2.1. Organisme d’accueil 4](#_Toc170539816)

[1.2.2. Activités et produits de VERMEG 5](#_Toc170539817)

[1.2.3. Organisation de VERMEG 5](#_Toc170539818)

[1.3. Concepts de base 6](#_Toc170539819)

[1.3.1. Problématique 6](#_Toc170539820)

[1.3.2. Objectifs du projet 7](#_Toc170539821)

[1.3.3. Processus de développement 7](#_Toc170539822)

[a. Comparaison des approches classiques et agiles 7](#_Toc170539823)

[b. Méthode de gestion de projet Agile SCRUM 8](#_Toc170539824)

[1.4. Étude théorique 9](#_Toc170539825)

[1.4.1. Prospection de solutions existantes 9](#_Toc170539826)

[a. UiPath 9](#_Toc170539827)

[b. Microsoft Power Automate 9](#_Toc170539828)

[c. Google Cloud Natural Language Processing (NLP) 10](#_Toc170539829)

[d. Camunda 10](#_Toc170539830)

[1.4.2. Critiques et recommandations 10](#_Toc170539831)

[1.4.3. Solution proposée 12](#_Toc170539832)

[1.5. Planification des tâches 12](#_Toc170539833)

[1.6. Conclusion 13](#_Toc170539834)

[Chapitre 2 : Capture et spécification des besoins 16](#_Toc170539835)

[2.1. Introduction 16](#_Toc170539836)

[2.2. Vision du projet 16](#_Toc170539837)

[2.2.1. Exigences fonctionnelles 16](#_Toc170539838)

[2.2.2. Exigences non fonctionnelles 17](#_Toc170539839)

[2.3. Langage de modélisation et de conception 19](#_Toc170539840)

[2.4. Spécification des exigences 19](#_Toc170539841)

[2.4.1. Identification des acteurs 19](#_Toc170539842)

[2.4.2. Identification des cas d’utilisation 20](#_Toc170539843)

[2.4.3. Diagramme des cas d’utilisation global 20](#_Toc170539844)

[2.4.1. Diagrammes des cas d’utilisation raffinés 21](#_Toc170539845)

[a. Raffinement de cas d’utilisation « Traitement automatique des courriels » 22](#_Toc170539846)

[b. Raffinement de cas d’utilisation « Consulter les permissions d’accès » 23](#_Toc170539847)

[c. Raffinement de cas d’utilisation « Consulter les comptes » 24](#_Toc170539848)

[d. Raffinement de cas d’utilisation « Consulter les Types de traitement » 25](#_Toc170539849)

[e. Raffinement de cas d’utilisation « Consulter les catégories » 26](#_Toc170539850)

[2.5. Pilotage du projet avec SCRUM 27](#_Toc170539851)

[2.5.1. Identification des Rôles 27](#_Toc170539852)

[2.5.2. Le Backlog Produit 27](#_Toc170539853)

[2.6. Planification des sprints 31](#_Toc170539854)

[2.7. Diagramme de séquence système 33](#_Toc170539855)

[a. Diagramme de séquence « Traitement automatique des courriels » 34](#_Toc170539856)

[b. Diagramme de séquence « Traitement des Emails Non Structurés » 35](#_Toc170539857)

[2.8. Conclusion 36](#_Toc170539858)

[Chapitre 3 : Etude conceptuelle 38](#_Toc170539859)

[3.1. Introduction 38](#_Toc170539860)

[3.2. Diagramme de classes 38](#_Toc170539861)

[3.3. Dictionnaire de données 40](#_Toc170539862)

[3.4. Modélisation Relationnelle 44](#_Toc170539863)

[3.4.1. Description des relations entre les tables 44](#_Toc170539864)

[3.4.2. Description des Relations 44](#_Toc170539865)

[3.5. Diagrammes de séquences de conception 46](#_Toc170539875)

[3.5.1. Diagramme de séquence du cas d’utilisation « S’authentifier » 47](#_Toc170539876)

[3.5.2. Diagramme de séquence « Gérer les Comptes » 49](#_Toc170539877)

[3.6. Diagramme de composants 51](#_Toc170539878)

[3.7. Conclusion 54](#_Toc170539879)

[Chapitre 4 : Présentation technique 56](#_Toc170539880)

[4.1. Introduction 56](#_Toc170539881)

[4.2. Choix technologiques 56](#_Toc170539882)

[4.2.1. Environnement technique 56](#_Toc170539883)

[a. Environnement matériel 56](#_Toc170539884)

[b. Environnement logiciel 56](#_Toc170539885)

[4.2.2. Schéma de développement 61](#_Toc170539886)

[a. Le Modèle (modèle de données) 61](#_Toc170539887)

[b. La Vue (Présentation, Interface utilisateur) 61](#_Toc170539888)

[c. Le Contrôleur 61](#_Toc170539889)

[4.2.3. Architecture logique de l’application 62](#_Toc170539890)

[4.2.4. Architecture logicielle 66](#_Toc170539891)

[4.2.5. Architecture Technique 67](#_Toc170539893)

[4.3. Processus DevOps 69](#_Toc170539896)

[4.3.1. Contrôle du code source 69](#_Toc170539897)

[4.3.2. Design Patterns 69](#_Toc170539898)

[4.3.3. Tests continus 69](#_Toc170539899)

[4.4. Jeu d'essais et manuel d’utilisateur 71](#_Toc170539900)

[4.4.1. API – Interfaces de programmation 71](#_Toc170539901)

[4.4.2. Parties Front-end/Back-end 72](#_Toc170539902)

[4.5. Conclusion 75](#_Toc170539903)

[Conclusion générale et perspectives 77](#_Toc170539904)

[WEBOGRAPHIE 78](#_Toc170539905)

**Table des figures**

[Figure 1 : Dispersion géographique de ses différents clients. 4](#_Toc170535412)

[Figure 2 : Le cycle de vie du SCRUM 9](#_Toc170535413)

[figure 3 : diagramme de gantt 13](#_Toc170535414)

[Figure 4: langage de modélisation unifié UML 19](#_Toc170535415)

[Figure 5 : Diagramme de cas d'utilisation global 21](#_Toc170535416)

[Figure 6 : Diagramme de cas d’utilisation « Traitement automatique des courriels » 22](#_Toc170535417)

[Figure 7 : Diagramme de cas d’utilisation « Consulter les permissions d’accès » 23](#_Toc170535418)

[Figure 8 : Diagramme de cas d’utilisation « Consulter les comptes » 24](#_Toc170535419)

[Figure 9 : Diagramme de cas d’utilisation « Consulter les Types de traitement » 25](#_Toc170535420)

[Figure 10 : Diagramme de cas d’utilisation « Consulter les catégories » 26](#_Toc170535421)

[Figure 11 : Diagramme de séquence « Traitement automatique des courriels » 34](#_Toc170535422)

[Figure 12 : Diagramme de séquence « Traitement des Emails Non Structurés » 35](#_Toc170535423)

[Figure 13 : Diagramme de classes 38](#_Toc170535424)

[Figure 14 : Diagramme de séquence du cas d’utilisation « S’authentifier » 47](#_Toc170535425)

[Figure 15 : Diagramme de séquence « Gérer les Comptes » 49](#_Toc170535426)

[Figure 16 : Diagramme de composants du cas d'utilisation « Traiter les Emails Non Structurés » 51](#_Toc170535427)

[Figure 17 : Logo draw.io 56](#_Toc170535428)

[Figure 18 : Logo IntellijIDEA 57](#_Toc170535429)

[Figure 19 : Logo Visual Studio Code 57](#_Toc170535430)

[Figure 20 : Logo Postman 57](#_Toc170535431)

[Figure 21 : Logo MongoDB Compass 57](#_Toc170535432)

[Figure 22 : LOGO JAVA 58](#_Toc170535433)

[Figure 23 : Logo JavaScript 58](#_Toc170535434)

[Figure 24 : Logo HTML 58](#_Toc170535435)

[Figure 25: Logo Springboot 60](#_Toc170535436)

[Figure 26: LOGO angular 60](#_Toc170535437)

[Figure 27 : Architecture MVC 61](#_Toc170535438)

[Figure 28 : Architecture Microservices 62](#_Toc170535439)

[Figure 29 : Aperçu de l'architecture KAFKA 64](#_Toc170535440)

[Figure 30 : ARCHITECTURE LOGICIELLE DE MICROSERVICE EMAILPROCESSOR 67](#_Toc170535441)

[Figure 31 : ARCHITECTURE TECHNIQUE 68](#_Toc170535442)

[Figure 32: la réussite des tests unitaires du microservice "EmailProcessor-Service" 70](#_Toc170535443)

[Figure 33 : la réussite des tests d'intégration du microservice "EmailProcessor-Service" 71](#_Toc170535444)

[Figure 35 : EndPoint pour authentification 71](#_Toc170535445)

[Figure 34: Code de l'endpoint de connexion dans le contrôleur 71](#_Toc170535446)

[Figure 36 : EndPoint pour authentification 71](#_Toc170535447)

[Figure 37: Formulaire de connexion utilisateur 72](#_Toc170535448)

[Figure 38: Interface du tableau de bord administrateur 72](#_Toc170535449)

[Figure 39 : Interface de la boîte de réception du service de messagerie 73](#_Toc170535450)

[Figure 40 : Interface de gestion des catégories 73](#_Toc170535451)

[Figure 41: Interface de Gestion des Contacts 74](#_Toc170535452)

[Figure 42: Interface de gestion des rôles 75](#_Toc170535453)

[Figure 43 : Interface de gestion du profil utilisateur 75](#_Toc170535454)

**Listes des tableaux**

[Tableau 1 : Tableau comparatif 7](#_Toc170486309)

[Tableau 2 : Tableau des cas d'utilisation 20](#_Toc170486310)

[Tableau 3 : Scénario du cas d’utilisation « Traitement automatique des courriels » 22](#_Toc170486311)

[Tableau 4 : Scénario du cas d’utilisation « Consulter les permissions d’accès » 23](#_Toc170486312)

[Tableau 5: Scénario du cas d’utilisation « Consulter les comptes » 24](#_Toc170486313)

[Tableau 6: Scénario du cas d’utilisation « Consulter les Types de traitement » 25](#_Toc170486314)

[Tableau 7 : Scénario du cas d’utilisation « Consulter les catégories » 26](#_Toc170486315)

[Tableau 8 : Tableau de Backlog 28](#_Toc170486316)

[Tableau 9 : Tableau de planification des releases 33](#_Toc170486317)

[Tableau 10 : Tableau descriptif du diagramme de classes 39](#_Toc170486318)

[Tableau 11: Table « Email » 40](#_Toc170486319)

[Tableau 12: Table « EmailProcessingResult » 41](#_Toc170486320)

[Tableau 13: Table « Attachement » 41](#_Toc170486321)

[Tableau 14: Table « Client » 41](#_Toc170486322)

[Tableau 15: Table « RelatedData » 41](#_Toc170486323)

[Tableau 16: Table « Notification » 42](#_Toc170486324)

[Tableau 17: Table « User » 42](#_Toc170486325)

[Tableau 18: Table « securityRole » 42](#_Toc170486326)

[Tableau 19: Table « Action » 43](#_Toc170486327)

[Tableau 20: Table « ActionParam » 43](#_Toc170486328)

[Tableau 21: Table « Category » 43](#_Toc170486329)

[Tableau 22: Table « Keyword » 43](#_Toc170486330)

[Tableau 23: Relations entre les Classes 44](#_Toc170486331)

[Tableau 24 : Scénario du diagramme de séquence « Authentification » 47](#_Toc170486332)

[Tableau 25 : Scénario du diagramme de séquence « Gérer les Comptes » 49](#_Toc170486333)

Introduction générale

Les méthodes traditionnelles d'apprentissage de la conduite, bien qu'efficaces, présentent plusieurs défis en termes de coût, de temps, et de sécurité. La nécessité de pratiquer sur des véhicules réels entraîne des dépenses considérables, notamment pour le carburant, l'entretien, et le temps des moniteurs. De plus, l'encombrement du trafic urbain et les coûts élevés associés aux nombreuses heures de conduite représentent des obstacles majeurs. Certains candidats, nécessitant plus de temps pour s'adapter à la conduite, peuvent voir leurs frais considérablement augmenter, ce qui impose des contraintes supplémentaires pour les auto-écoles en termes de ressources.

C'est dans ce contexte que s'inscrit notre projet de fin d'études, baptisé "Simulateur VR de test de conduite". Ce projet innovant a été conçu pour répondre aux besoins croissants des auto-écoles. L'objectif est de développer un simulateur VR immersif, capable de reproduire fidèlement les conditions de conduite réelles, tout en intégrant des règles de circulation strictement respectées ainsi que des tests de code de la route.

Le projet se déroule en plusieurs phases, mettant en lumière les différentes étapes de sa conception et de son développement. Nous commencerons par une introduction posant le contexte et les objectifs du projet, suivie d'une analyse approfondie et d'une spécification des exigences, afin d'identifier les besoins essentiels et les fonctionnalités clés du simulateur. La phase de conception sera suivie de la phase de réalisation, où nous appliquerons la méthodologie Scrum pour garantir une gestion efficace du projet. Enfin, nous analyserons les fonctionnalités majeures qui en découlent, ainsi que les avantages et les perspectives futures de cette solution.

Le rapport est organisé en quatre chapitres distincts afin de fournir une vue d'ensemble complète du projet :

1. Présentation générale :

Ce premier chapitre présente la société d’accueil, expose la problématique, et définit les objectifs du projet. Nous y analysons également les solutions existantes sur le marché, les besoins fonctionnels et non fonctionnels du projet, et nous expliquons la méthodologie de travail utilisée. Ce chapitre inclut une étude des approches adoptées par d'autres solutions similaires et détaille les exigences spécifiques que notre plateforme vise à satisfaire.

1. Analyse et spécification des besoins :

Ce chapitre est consacré à la collecte et à la spécification des besoins du projet. Nous détaillerons les processus de capture des exigences des utilisateurs finaux et des parties prenantes, ainsi que la documentation de ces besoins, pour garantir que la solution développée répond précisément aux attentes identifiées.

1. Analyse et conception :

Ce chapitre se concentre sur la conception détaillée du projet. Il s'agit d'une phase préparatoire à l'implémentation où nous explorons l'architecture globale de la plateforme, les modèles de données, et les choix de conception effectués pour répondre aux besoins identifiés lors de l'analyse.

1. Réalisation :

Le dernier chapitre aborde les aspects techniques du développement du projet. Nous présenterons les technologies utilisées et les scénarios expérimentaux permettant de démontrer le fonctionnement pratique de la plateforme.

L'objectif ultime de ce rapport est d’offrir une vue exhaustive du projet et de démontrer son potentiel pour révolutionner la formation à la conduite au sein des auto-ecoles.

CHAPITRE 1 :

PRESENTATION GENERALE

Chapitre 1 : Présentation générale

Introduction

Ce chapitre de présentation générale vise à fournir une vue d'ensemble du contexte dans lequel s'inscrit notre projet "Emails Processor". Il couvre la présentation de la société d'accueil, les concepts de base liés à notre projet, et une étude théorique des solutions existantes sur le marché. Il s'agit de situer le projet dans son environnement spécifique, d'identifier les problématiques auxquelles il répond, de préciser les objectifs poursuivis, et de décrire le processus de développement adopté. Enfin, ce chapitre propose une analyse critique des solutions existantes et des recommandations pour améliorer l'efficacité de la gestion des e-mails dans le secteur bancaire.

Concepts de base

* + 1. Problématique

Face à des points de friction majeurs, dont la réception quotidienne de 40 000 courriels par une équipe de 700 personnes, soit environ 572 courriels par individu, émerge un risque évident. Même en tenant compte d'un effectif localisé dans une région à faible coût, où chaque employé représente un investissement de 40 000 $, les coûts atteignent 28 millions de dollars. L'objectif clé réside dans l'arrêt de la croissance de l'équipe, avec une finalité axée sur la simplification des processus (STP) et une réduction ultérieure des coûts.

Les risques opérationnels augmentent avec l'accroissement des processus manuels, accroissant les probabilités d'erreurs et d'omissions, ce qui impacte directement la satisfaction du client. Le poids de la gestion manuelle se fait sentir au sein d'une équipe de 700 personnes, où même le simple routage des courriels représente une charge significative, consommant un temps considérable.

Actuellement, Santander gère cette tâche en faisant tourner environ 50 personnes, chacune y consacrant environ 2 heures par jour. Le simple acte de routage engendre un coût important – 100 heures par jour, équivalant à plus de 12 équivalents temps plein (FTE) uniquement pour cette fonction. Même avec un coût unitaire relativement bas de 40 000 $ par employé, cela représente près de 500 000 $ annuellement, avec une tendance à la hausse[3][4].

La solution envisagée se concentre sur l'amélioration de la vitesse du processus. En automatisant ne serait-ce que le routage, des économies substantielles de centaines de milliers de dollars peuvent être réalisées. En augmentant l'efficacité au-delà de cette étape, le projet vise à générer des économies de plusieurs millions de dollars, à améliorer la satisfaction client (CSAT), tout en permettant une expansion continue de l'activité, le tout accompagné d'une réduction des coûts.

* + 1. Objectifs du projet

Notre projet s'inscrit dans le domaine du traitement automatisé des courriels. Il a pour objectif principal de concevoir et de mettre en œuvre une solution permettant de réduire les coûts opérationnels liés à la gestion des courriels au sein des institutions financières, en particulier dans le secteur bancaire.

Les objectifs spécifiques du projet incluent :

* Automatisation du traitement des courriels entrants, afin de diminuer le nombre d'opérations manuelles et de réduire les erreurs associées.
* Mise en place d'un système de routage intelligent des courriels, capable de classer et de diriger automatiquement les messages vers les destinataires appropriés.
* Intégration de la solution avec les systèmes existants de l'entreprise, afin de garantir une compatibilité et une interopérabilité maximales.
* Amélioration de la satisfaction des clients en fournissant des réponses plus rapides et plus précises à leurs demandes.
* Réduction des coûts opérationnels liés à la gestion des courriels, grâce à une diminution du nombre de personnes nécessaires pour effectuer ces tâches."
  + 1. Processus de développement

1. Comparaison des approches classiques et agiles

On peut distinguer deux grandes approches pour la gestion des projets, les approches classiques et agiles. Pour choisir une approche particulière, nous avons dressé le tableau ci-dessous qui présente une comparaison entre eux [5] [6] :

Tableau 1 : Tableau comparatif

|  |  |  |
| --- | --- | --- |
| Critères | Approche classique | Approche Agile |
| Cycle de vie | Phase séquentielles ou itératives | Itératif et incrémentale |
| Planification | Prédictive | Adaptative |
| Changement | Résistance au changement. Processus lourds de gestion des changements acceptés. | Accueil favorable au changement intégré dans le processus. |
| Gestion des risques | Processus district et rigoureux de gestion des risques. | Gestion des risques intégrée dans le processus global. |
| Mesure des succès | Respect des engagements initiaux en termes de coût, de budget et de niveau de qualité. | Satisfaction du client par la livraison de valeur souhaitée. |
| Equipe | Equipe avec ressources spécialisés, dirigées par un chef de projet. | Equipe responsabilisée, soutenue par un responsable de projet. |
| Qualité | Contrôle de qualité à la fin de cycle de développement | Contrôle de qualité permanant au niveau du produit et du processus. |

D’après la comparaison faite au tableau ci-dessus, nous pouvons remarquer qu’une approche agile, menée de manière collaborative et incrémentale, produit des solutions de haute qualité en répondant aux besoins évolutifs des clients. Elle assure une meilleure communication avec le client et une visibilité accrue du produit livrable, permet de gérer la qualité en continu et détecte rapidement les problèmes, facilitant ainsi les actions correctrices avec un impact minimal sur les coûts et les délais.

1. Méthode de gestion de projet Agile SCRUM

Pour notre projet, nous avons choisi le cadre de travail SCRUM pour ses réunions quotidiennes qui renforcent la communication et l'esprit d'équipe, augmentant ainsi la productivité. SCRUM aide à respecter les délais en divisant le travail en itérations de deux à quatre semaines appelées Sprints, chaque Sprint aboutissant à la livraison d’un produit partiel fonctionnel.

Une image contenant texte, capture d’écran, Police, Graphique

Description générée automatiquementLa figure ci-dessous illustre les étapes à suivre dans le cycle de vie du SCRUM :

Figure 2 : Le cycle de vie du SCRUM

Étude théorique

* + 1. Prospection de solutions existantes

Actuellement, le marché propose diverses solutions d'automatisation des courriels visant à améliorer l'efficacité opérationnelle. Examinons de plus près certaines de ces solutions et identifions leurs avantages et inconvénients par rapport à notre demande.

1. UiPath

UiPath est une plateforme d'automatisation des processus robotisés (RPA) qui permet d'automatiser des tâches complexes avec une interface utilisateur intuitive [7].

* **Avantages** : Interface utilisateur intuitive, automatisation des tâches complexes.
* **Inconvénients** : Nécessite une certaine expertise en automatisation.
* **Critique** : Bien que UiPath soit puissant pour l'automatisation générale des processus, son niveau d'expertise requis peut être un obstacle, limitant son accessibilité à un public non technique. De plus, son modèle de licence payante peut engendrer des coûts élevés.

1. Microsoft Power Automate

Microsoft Power Automate est une solution d'automatisation qui permet de créer des flux de travail automatisés entre différentes applications et services [8].

* **Avantages** : Intégration étroite avec les produits Microsoft, facilité d'utilisation.
* **Inconvénients** : Certaines fonctionnalités peuvent nécessiter une licence payante.
* **Critique** : Bien que Power Automate soit bien intégré dans l'écosystème Microsoft, sa dépendance aux licences payantes pour des fonctionnalités avancées peut entraîner des coûts variables et représenter un défi financier.

1. Google Cloud Natural Language Processing (NLP)

Google Cloud NLP offre des outils avancés pour l'analyse sémantique, l'extraction d'entités et la détection de sentiments dans les courriels [9].

* **Avantages** : Analyse sémantique avancée, extraction d'entités et de sentiments.
* **Inconvénients** : Peut nécessiter une intégration spécifique pour le traitement des courriels.
* **Critique** : Alors que Google Cloud NLP excelle dans l'analyse linguistique, il peut nécessiter des ajustements importants pour répondre précisément aux exigences du client, entraînant une complexité supplémentaire et des coûts associés.

1. Camunda

Camunda est une plateforme de gestion des processus métier (BPM) permettant de modéliser, exécuter et optimiser des processus métier complexes.

* **Avantages** : Gestion complète des processus métier, intégration aisée.
* **Inconvénients** : Peut demander une courbe d'apprentissage pour une utilisation avancée.
* **Critique** : Bien que Camunda soit robuste pour la gestion des processus métier, il peut être perçu comme complexe pour des utilisateurs non techniques. Son niveau d'apprentissage peut représenter un obstacle à une adoption rapide et généralisée, ce qui pourrait également entraîner des coûts liés à la formation.
  + 1. Critiques et recommandations

Après avoir examiné les solutions existantes sur le marché, plusieurs considérations critiques et recommandations générales émergent.

Critiques générales des solutions existantes :

* **Complexité et expertise requise** : La plupart des solutions d'automatisation des courriels nécessitent une expertise significative pour la configuration et l'optimisation, ce qui peut limiter leur accessibilité à un large éventail d'utilisateurs au sein de l'organisation.
* **Coûts de licence et d'implémentation** : Les solutions disponibles peuvent être accompagnées de coûts de licence élevés pour accéder à des fonctionnalités avancées, ce qui représente un défi financier pour les entreprises cherchant à optimiser leurs opérations.
* **Intégration et adaptation** : L'intégration des solutions dans l'infrastructure existante peut nécessiter des ajustements complexes et prolongés, entraînant des dépenses supplémentaires et des retards dans la mise en œuvre.

Recommandations pour la future solution

* **Évaluation des besoins spécifiques** : Avant de choisir une solution d'automatisation des courriels, il est crucial d'effectuer une analyse approfondie des besoins spécifiques de ce projet. Cela inclut la prise en compte des volumes de courriels traités, des types de processus à automatiser et des exigences de sécurité.
* **Stratégie de mise en œuvre progressive** : Adopter une approche de déploiement progressif et modulaire peut réduire les risques associés à l'implémentation d'une nouvelle solution. Commencer par des tests pilotes permet de valider l'efficacité de la solution avant un déploiement à grande échelle.
* **Formation et support utilisateur** : Investir dans la formation des utilisateurs finaux et dans un support continu est essentiel pour maximiser l'adoption et l'utilisation efficace de la nouvelle technologie. Cela contribue également à réduire la courbe d'apprentissage et à minimiser les erreurs opérationnelles.
* **Évaluation continue des performances** : Mettre en place un processus d'évaluation continue pour surveiller les performances de la solution choisie est recommandé. Cela permet d'identifier les domaines d'amélioration potentiels et d'optimiser l'efficacité opérationnelle au fil du temps.
  + 1. Solution proposée

Suite à cette étude précédente, nous proposons de concevoir et de développer une application intitulée "Email Processor" qui permet de :

* Automatiser la réception, le routage et la classification des courriels entrants.
* Intégrer des algorithmes d'analyse linguistique pour extraire les informations pertinentes et identifier les priorités des courriels.
* Mettre en place un tableau de bord pour le suivi des performances et des indicateurs clés de traitement des courriels.
* Assurer une intégration fluide avec les systèmes existants pour garantir une interopérabilité maximale.
* Fournir des outils de gestion des rôles et des permissions pour un accès sécurisé et contrôlé aux fonctionnalités du système.

En conclusion, le choix et la mise en œuvre d'une solution d'automatisation des courriels doivent être soigneusement planifiés pour répondre aux besoins spécifiques du projet "Emails Processor" tout en minimisant les défis potentiels associés à l'intégration et à l'adoption de nouvelles technologies.

Planification des tâches

La planification des tâches pour notre projet se déroule selon la méthodologie Agile Scrum, favorisant une approche itérative et incrémentale. Chaque phase du projet est divisée en sprints, permettant des cycles de développement courts et des réajustements continus basés sur les retours d'expérience.

Un diagramme de Gantt est un outil de gestion de projet aidant à la planification et à l'ordonnancement de projets de toutes tailles, bien qu'ils soient particulièrement utiles pour simplifier des projets complexes. Les calendriers et les tâches de gestion de projet sont convertis en un graphique à barres horizontales, indiquant les dates de début et de fin, ainsi que les dépendances, la planification et les échéances. Ceci est utile pour garder les tâches sur la bonne voie lorsqu'il y a une grande équipe et plusieurs parties prenantes lorsque la portée change [11].

Une image contenant capture d’écran, texte, ligne, Police

Description générée automatiquement

figure 3 : diagramme de gantt

Le projet est structuré en quatre grandes parties : l'étude, le développement, les tests et le rapport.

* **Étude préalable :** Consiste à effectuer une analyse approfondie du projet, en évaluant les besoins et les exigences spécifiques. Cette phase permet de définir clairement les objectifs du projet et d'identifier les meilleures approches à suivre.
* **Étude conceptuelle :** Cette étape consiste à concevoir l'architecture de l'application, y compris la modélisation des données, la définition des composants et des interactions entre eux, et la validation de l'architecture proposée avec les parties prenantes.
* **Développement :** C'est à ce stade que l'application est créée, en mettant en place les fonctionnalités nécessaires en accord avec les spécifications établies lors de l'étude.
* **Tests :** Cette phase vise à vérifier le bon fonctionnement de l'application en effectuant différents types de tests, tels que des tests unitaires, des tests d'intégration et des tests de validation. L'objectif est de s'assurer que l'application répond aux exigences et fonctionne de manière fiable.
* **Rapport :** C'est ici que nous documentons les différentes étapes du projet, les résultats des tests et les éventuels problèmes rencontrés. Le rapport offre une vue d'ensemble du projet, résume les principales réalisations et peut servir de référence pour les futurs travaux ou présentations.

Conclusion

Ce chapitre a fourni une vue d'ensemble du projet "Emails Processor", présentant l'entreprise VERMEG, ses activités, et son organisation. Nous avons exploré les problématiques de gestion des courriels dans les institutions financières, soulignant les enjeux et les coûts des processus manuels inefficaces.

Les objectifs du projet ont été définis avec un accent sur l'automatisation du traitement des courriels, la mise en place d'un système de routage intelligent et l'amélioration de la satisfaction client. Nous avons comparé les approches de gestion de projet classiques et agiles, soulignant les avantages de la méthode SCRUM choisie pour ce projet.

L'étude théorique a examiné les solutions existantes sur le marché, leurs avantages et inconvénients, et proposé des recommandations pour une solution adaptée aux besoins du projet. La solution "Email Processor" a été décrite avec ses principales fonctionnalités et bénéfices.

Enfin, la planification des tâches selon la méthodologie Agile SCRUM a été détaillée, assurant une approche itérative et incrémentale pour le développement du projet. Ce cadre servira de base pour les chapitres suivants, où nous aborderons la conception, le développement et la mise en œuvre de notre solution.

CHAPITRE 2 :

CAPTURE ET SPECIFICATION DES BESOINS

Chapitre 2 : Capture et spécification des besoins

Introduction

Dans ce chapitre, nous mettons en place le cadre de travail SCRUM en présentant les profils utilisateurs, l’équipe SCRUM, les User Stories et le Backlog du produit. Nous clôturerons le chapitre par la planification des sprints et des releases. Nous détaillerons également la vision du projet, les exigences fonctionnelles et non fonctionnelles, ainsi que les différents diagrammes UML nécessaires à la modélisation du système. Enfin, nous fournirons une explication des diagrammes de séquence pour illustrer les interactions dynamiques du système.

Vision du projet

Ce projet a pour objectif de créer un système de traitement automatisé des courriels pour les institutions financières, visant à optimiser la gestion des communications électroniques, améliorer l'efficacité opérationnelle et réduire les coûts. La vision détaillée du projet est déclinée en exigences fonctionnelles et non fonctionnelles.

* + 1. Exigences fonctionnelles
* **Traitement Automatisé des Courriels :**
  + Le système doit pouvoir automatiser le traitement des courriels entrants provenant de diverses sources.
  + Une fonctionnalité de classification automatique des courriels basée sur le contenu doit être mise en place.
* **Intégration Polyvalente :**
  + Le projet doit permettre l'intégration transparente de différentes sources de courriels, assurant une gestion centralisée des communications électroniques.
  + La conception modulaire doit permettre l'ajout facile de nouvelles sources sans impact sur les fonctionnalités existantes.
* **Analyse Linguistique Avancée :**
  + L'utilisation d'algorithmes de traitement du langage pour effectuer une analyse sémantique avancée, y compris l'extraction d'entités et la compréhension contextuelle.
  + L'analyse du ton et du contenu des courriels pour faciliter une réponse plus précise aux demandes des clients.
* **Gestion des Priorités et des Urgences :**
  + Le système doit détecter automatiquement les courriels nécessitant une action urgente ou une priorité élevée.
  + Une fonctionnalité de gestion des priorités pour assurer un traitement rapide des demandes critiques.
* **Tableau de Bord de Gestion :**
  + Un tableau de bord intuitif qui présente des indicateurs clés tels que le volume de courriels traités, les temps de réponse, et les statistiques par catégorie doivent être disponibles.
* **Adaptabilité et Personnalisation :**
  + Le système doit être facilement adaptable aux besoins spécifiques du client, permettant la personnalisation des catégories, des règles de traitement, etc.
  + Des mécanismes de configuration conviviaux pour permettre des ajustements sans intervention technique complexe.
* **Sécurité et Confidentialité :**
  + Mise en place de mécanismes de sécurité robustes pour garantir la confidentialité des données clients.

L’accès à l’application est sécurisé moyennant des paramètres crypté et des rôles bien définis.

* + 1. Exigences non fonctionnelles
* **Performance :** 
  + Critère : Le système doit traiter les courriels de manière efficace, assurant une réponse rapide et précise.
  + Exigence : Le temps de traitement des courriels ne doit pas dépasser X secondes pour garantir une expérience utilisateur optimale.
* **Sécurité des Données :**
  + Critère : Assurer la confidentialité et l'intégrité des données clients.
  + Exigence : Le projet doit être conforme aux normes de sécurité des données, notamment le respect du Règlement général sur la protection des données (GDPR).
* **Facilité d'Utilisation :**
  + Critère : Le système doit être convivial pour les utilisateurs finaux.
  + Exigence : Une formation minimale ne doit pas excéder X heures pour permettre une adoption rapide par l'équipe opérationnelle.
* **Adaptabilité :**
  + Critère : Le système doit s'adapter facilement aux changements de sources de courriels ou aux nouvelles fonctionnalités.
  + Exigence : Les mises à jour du système ne doivent pas entraîner d'interruption significative du service.
* **Audit et Traçabilité :**
  + Critère : Le système doit permettre le suivi des actions effectuées sur les courriels.
  + Exigence : Un journal d'audit détaillé doit enregistrer toutes les activités liées au traitement des courriels.
* **Interopérabilité :**
  + Critère : Le système doit s'interfacer sans heurts avec d'autres systèmes internes de l'entreprise.
  + Exigence : Le projet doit prendre en charge les protocoles d'intégration standards, tels que les API REST.
* **Récupération en Cas de Sinistre :**
  + Critère : Le système doit avoir une stratégie de sauvegarde et de récupération en cas de perte de données.
  + Exigence : Les données doivent être sauvegardées régulièrement et la récupération en cas de sinistre doit être testée périodiquement

Langage de modélisation et de conception

Une image contenant Graphique, graphisme, Police, conception

Description générée automatiquementLe succès ou l’échec d’un développement se réfère en grande partie à la phase de modélisation. Avant d’attaquer aveuglement le code, la modélisation du système facilite énormément sa mise en œuvre et élimine le risque de naufrage du projet. C’est pourquoi, de nos jours, les outils de modélisation de processus métier s’étoffent chaque année et les suites logicielles sont de plus en plus nombreuses, donc nous avons choisi le langage de modélisation unifié UML (Unified Modeling Langage), vu qu’il est reconnu comme étant le standard industriel par excellence de la modélisation objet. De plus, son indépendance par rapport aux langages de programmation, aux domaines de l’application et aux processus, son caractère polyvalent et sa souplesse ont fait de lui un langage universel [12].

Figure 4: langage de modélisation unifié UML

Spécification des exigences

* + 1. Identification des acteurs
* **Utilisateur Final (Opérateurs/Équipe Opérationnelle) :**
* Rôle : Utilisateur principal du système pour gérer les courriels manière efficace.
* **Administrateur Système :**
* Rôle : Responsable de la gestion des comptes, gestion des paramètres système, de la maintenance et de la résolution des problèmes techniques
* **Technicien :**
* Rôle : Responsable de la maintenance et de la résolution des problèmes techniques
* **Analyste :**
* Rôle : Responsable de l'analyse et de l'optimisation des processus métier, ainsi que de la génération de rapports sur les performances du système.
  + 1. Identification des cas d’utilisation

Tableau 2 : Tableau des cas d'utilisation

|  |  |
| --- | --- |
| **Cas d’utilisation** | **Description** |
| Traiter les emails | Automatiser le traitement et la classification des emails entrants. |
| Analyser les emails | Analyser le contenu des emails pour en extraire des informations pertinentes. |
| Archiver les emails | Stocker les emails de manière organisée pour une consultation ultérieure. |
| Gérer les utilisateurs | Ajouter, modifier et supprimer des utilisateurs dans le système. |
| Gérer les rôles | Définir et attribuer des rôles aux utilisateurs pour contrôler les accès. |
| Configurer les sources | Ajouter et configurer les différentes sources d’emails à traiter par le système. |
| Visualiser les statistiques | Afficher des statistiques sur les performances du traitement des emails. |
| Générer des rapports | Créer des rapports sur les activités et performances du système de gestion d’emails. |

* + 1. Diagramme des cas d’utilisation global

Pour présenter les fonctionnalités de notre système de maniéré formelle, nous utilisons le diagramme de cas d’utilisation du langage de modélisation UML.

Une image contenant texte, diagramme, dessin, croquis

Description générée automatiquementDans la figure ci-dessous, nous illustrons le diagramme de cas d’utilisation global de notre application. Ce diagramme représente les différents acteurs ainsi que les cas d’utilisations affectés à chacun de ces acteurs.

Figure 5 : Diagramme de cas d'utilisation global

* + 1. Diagrammes des cas d’utilisation raffinés

Les diagrammes de cas d'utilisation modélisent le comportement d'un système et permettent de capturer les exigences du système. Ils identifient également les interactions entre le système et ses acteurs.

1. Raffinement de cas d’utilisation « Traitement automatique des courriels »

Une image contenant texte, diagramme, capture d’écran, ligne

Description générée automatiquement

Figure 6 : Diagramme de cas d’utilisation « Traitement automatique des courriels »

* Description textuelle de cas d’utilisation « Traitement automatique des courriels »

Tableau 3 : Scénario du cas d’utilisation « Traitement automatique des courriels »

|  |  |
| --- | --- |
| **Cas d’utilisation** | Traitement automatique des courriels |
| **Objectif** | Automatiser le traitement et la classification des courriels entrants |
| **Précondition** | L'utilisateur est authentifié et dispose des permissions nécessaires. |
| **Post \_condition** | Les courriels sont traités et classifiés automatiquement. |
| **Scénario nominal** | 1. Le déclencheur initie le processus lors de la réception d'un nouvel email. 2. Le système enregistre l'email dans la base de données. 3. Le système classifie automatiquement l'email selon les règles définies. |
| **Scénario alternatif** | * Si le système ne parvient pas à récupérer les courriels, un message d'erreur est affiché. |

Explication des éléments du diagramme de cas d'utilisation

* **Enregistrement de l'email dans la base de données** :

Cette étape consiste à stocker l'email reçu dans la base de données du système pour un traitement ultérieur.

* **Classifier automatiquement l'email :**

Une fois l'email enregistré, le système applique des règles de classification prédéfinies pour trier et organiser les emails.

1. Raffinement de cas d’utilisation « Consulter les permissions d’accès »

Une image contenant texte, diagramme, Police, cercle

Description générée automatiquement

Figure 7 : Diagramme de cas d’utilisation « Consulter les permissions d’accès »

* Description textuelle de cas d’utilisation « Consulter les permissions d’accès »

Tableau 4 : Scénario du cas d’utilisation « Consulter les permissions d’accès »

|  |  |
| --- | --- |
| **Cas d’utilisation** | Consulter des permissions d'accès |
| **Acteurs** | Administrateur |
| **Objectif** | Consulter les permissions d'accès aux fonctionnalités du système |
| **Précondition** | L'administrateur est authentifié dans le système. |
| **Post \_condition** | Les permissions d'accès sont mises à jour. |
| **Scénario nominal** | 1. L'administrateur accède à la section de gestion des permissions. 2. L'administrateur sélectionne l'utilisateur et les permissions à modifier. 3. Le système met à jour les permissions de l'utilisateur. |
| **Scénario alternatif** | * Si la mise à jour des permissions échoue, le système affiche un message d’erreur. |

1. Une image contenant texte, diagramme, cercle, Police

   Description générée automatiquementRaffinement de cas d’utilisation « Consulter les comptes »

Figure 8 : Diagramme de cas d’utilisation « Consulter les comptes »

* Description textuelle de cas d’utilisation « Consulter les comptes »

Tableau 5: Scénario du cas d’utilisation « Consulter les comptes »

|  |  |
| --- | --- |
| **Cas d'utilisation** | Gérer les Comptes |
| **Acteurs** | Administrateur |
| **Objectif** | Gérer les comptes des utilisateurs |
| **Précondition** | L'administrateur est authentifié dans le système. |
| **Post-condition** | Les comptes des utilisateurs sont ajoutés, modifiés, supprimés ou désactivés selon l'action effectuée. |
| **Scénario nominal** | 1. L'administrateur accède à la section de gestion des comptes. 2. L'administrateur sélectionne l'option pour ajouter/modifier/supprimer/désactiver un compte. 3. L'administrateur entre les informations nécessaires ou confirme l'action. 4. Le système enregistre l'action. 5. Le compte est ajouté/modifié/supprimé/désactivé et l'utilisateur concerné reçoit une notification. |
| **Scénario alternatif** | * Si l'action échoue, le système affiche un message d’erreur et propose de réessayer ou de contacter le support technique. |

1. Raffinement de cas d’utilisation « Consulter les Types de traitement »

Une image contenant diagramme, texte, croquis, cercle

Description générée automatiquement

Figure 9 : Diagramme de cas d’utilisation « Consulter les Types de traitement »

* Description textuelle de cas d’utilisation « Consulter les Types de traitement »

Tableau 6: Scénario du cas d’utilisation « Consulter les Types de traitement »

|  |  |
| --- | --- |
| **Cas d'utilisation** | Consulter les Types de traitement |
| **Acteurs** | Administrateur, Technicien |
| **Objectif** | Gérer les types de traitement des emails pour organiser le workflow |
| **Précondition** | L'administrateur ou le technicien est authentifié dans le système. |
| **Post-condition** | Les types de traitement des emails sont ajoutés, modifiés ou supprimés. |
| **Scénario nominal** | 1. L'utilisateur accède à la section de gestion des types de traitement. 2. L'utilisateur sélectionne l'option pour ajouter/modifier/supprimer un type de traitement. 3. L'utilisateur entre les informations nécessaires ou confirme l'action. 4. Le système enregistre l'action. 5. Le type de traitement est ajouté/modifié/supprimé et les emails concernés sont mis à jour. |
| **Scénario alternatif** | * Si l'action échoue, le système affiche un message d’erreur et propose de réessayer ou de contacter le support technique. |

1. Raffinement de cas d’utilisation « Consulter les catégories »

Une image contenant texte, diagramme, croquis, cercle

Description générée automatiquementFigure 10 : Diagramme de cas d’utilisation « Consulter les catégories »

* Description textuelle de cas d’utilisation « Consulter les catégories »

Tableau 7 : Scénario du cas d’utilisation « Consulter les catégories »

|  |  |
| --- | --- |
| **Cas d'utilisation** | Consulter les Catégories des emails |
| **Acteurs** | Administrateur, Technicien |
| **Objectif** | Gérer les catégories des emails pour une meilleure organisation |
| **Précondition** | L'administrateur ou le technicien est authentifié dans le système. |
| **Post-condition** | Les catégories des emails sont ajoutées, modifiées ou supprimées. |
| **Scénario nominal** | 1. L'utilisateur accède à la section de gestion des catégories des emails. 2. L'utilisateur sélectionne l'option pour ajouter/modifier/supprimer une catégorie. 3. L'utilisateur entre les informations nécessaires ou confirme l'action. 4. Le système enregistre l'action. 5. La catégorie est ajoutée/modifiée/supprimée et les emails concernés sont mis à jour. |
| **Scénario alternatif** | * Si l'action échoue, le système affiche un message d’erreur et propose de réessayer ou de contacter le support technique. |

Pilotage du projet avec SCRUM

* + 1. Identification des Rôles

SCRUM définit trois rôles : le Propriétaire du produit (Product Owner), le SCRUM Master et l’équipe de développement.

* **Le propriétaire du produit (Product Owner)** **:** porte la vision du produit à réaliser et travaille en interaction avec l’équipe de développement. Il s’agit généralement d’un expert du domaine métier du projet.
* **Le SCRUM Master :** Maîtrise SCRUM et s’assure de son application correcte. Il a donc un rôle de Coach à la fois auprès du Product Owner et auprès de l’équipe de développement. Il doit donc faire preuve de pédagogie.
* **L’équipe de développement :** Transforme les besoins exprimés par le Product Owner en fonctionnalités utilisables. Elle est pluridisciplinaire, incluant des développeurs, des architectes logiciels, des administrateurs de base de données, des analystes fonctionnels, des graphistes/ergonomes et des ingénieurs systèmes.

Pour notre projet nous avons réparti les rôles suivants :

* Product Owner: Mr. Anaya Akram
* SCRUM Master: Mr. Anaya Akram
* Equipe de développement : Sara Mejbri.
  + 1. Le Backlog Produit

Le Product Bаcklog est l’ensemble des caractéristiques fonctionnelles qui constituent le produit souhaité Les caractéristiques fonctionnelles sont appelées des histoires utilisateurs (user story).

Les user stories sont caractérisés par :

* Features : elle contient les noms des sprints.
* Identifiant : Il détermine un identifiant unique pour l’histoire en question.
* Description : Elle décrit le besoin d’un acteur.
* Difficulté : de la complexité, elle est une valeur entière qui appartient à la suite de Fibonacci.
* Priorité : Les priorités sont utilisées pour définir l’ordre de réalisation, elles permettent de constituer le flux de stories qui va alimenter l’équipe.

Pour prioriser nos user stories, nous avons pris en compte les critères suivants :

* La valeur apportée (Business Value)
* La fréquence d’utilisation
* La réduction des risques
* L’incertitude sur des besoins des utilisateurs qu’un user story permettra de diminuer
* La contribution à la qualité. Les travaux visant à garantir la qualité du produit devraient être prioritaires
* Les dépendances entre stories

Le Backlog de produit consiste à énoncer la liste des spécifications fonctionnelles exigés par le client, et est récapitulé dans le tableau ci-dessous. L’échelle d’évaluation de la difficulté contient les valeurs : facile, moyen, assez difficile et difficile.

* Facile : tâches basiques.
* Moyen : tâches avec une certaine complexité.
* Assez difficile : tâches avec une complexité importante.
* Difficile : tâches très complexes.

Tableau 8 : Tableau de Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Fonctionnalité** | **User Story** | **Acteurs** | **Priorité** |
| Consulter le reporting Système | En tant qu'analyste, je peux consulter le reporting système | Analyste | Haute |
| Tableau de bord | En tant que membre de l'équipe opérationnelle, je veux un tableau de bord. | Haute | Difficile |
| Gérer les types de traitement | En tant qu'administrateur, je peux consulter les types de traitement des emails | Administrateur, Technicien | Haute |
| En tant qu'administrateur, je peux ajouter un type de traitement | Administrateur, Technicien | Moyenne |
| En tant qu'administrateur, je peux supprimer un type de traitement | Administrateur, Technicien | Moyenne |
| En tant qu'administrateur, je peux modifier un type de traitement | Administrateur, Technicien | Moyenne |
| Configuration des sources de courriels | En tant qu'administrateur système, je veux configurer les sources de courriels | Administrateur | Moyenne |
| Classification automatique des courriels | En tant qu'utilisateur final, je veux consulter la classification automatiquement des courriels. | Utilisateur final | Haute |
| Interface utilisateur | En tant qu’acteur, je veux une interface utilisateur | Tous acteurs | Haute |
| Gérer les Catégories des emails | En tant qu’administrateur, je peux consulter les catégories des emails | Administrateur, Technicien | Moyenne |
| En tant qu’administrateur, je peux ajouter des catégories des emails | Administrateur, Technicien | Moyenne |
| En tant qu’administrateur, je peux modifier des catégories des emails | Administrateur, Technicien | Moyenne |
| En tant qu’administrateur, je peux supprimer des catégories des emails | Administrateur, Technicien | Moyenne |
| En tant qu’administrateur, je peux gérer les keywords associés aux catégories des emails | Administrateur, Technicien | Moyenne |
| Gérer les Keywords | En tant qu’administrateur, je peux consulter les keywords | Administrateur, Technicien | Moyenne |
| En tant qu’administrateur, je peux ajouter des keywords | Administrateur, Technicien | Moyenne |
| En tant qu’administrateur, je peux modifier des keywords | Administrateur, Technicien | Moyenne |
| En tant qu’administrateur, je peux supprimer des keywords | Administrateur, Technicien | Moyenne |
| Consulter les Emails Urgentes | En tant qu'agent, je peux consulter les emails urgents | Agent | Haute |
| Consulter les Emails Non Structuré | En tant qu'analyste, je peux consulter les emails non structurés | Analyste, Technicien, Agent, Administrateur | Haute |
| En tant qu'administrateur, je peux ajouter un résultat d'email non structuré | Administrateur | Moyenne |
| En tant qu'administrateur, je peux supprimer un résultat d'email non structuré | Administrateur | Moyenne |
| En tant qu'administrateur, je peux modifier un résultat d'email non structuré | Administrateur | Moyenne |
| Authentification et gestion des rôles | En tant qu’administrateur, je veux consulter les permissions d'accès aux fonctionnalités du système | Administrateur | Haute |
| En tant qu'administrateur, je peux ajouter un rôle | Administrateur | Moyenne |
| En tant qu'administrateur, je peux annuler un rôle | Administrateur | Moyenne |
| En tant qu'administrateur, je peux modifier un rôle | Administrateur | Moyenne |
| Tableau de bord | En tant que membre de l'équipe opérationnelle, je veux un tableau de bord | Équipe opérationnelle | Haute |
| Consulter les Courriels | En tant qu'agent, je peux consulter les courriels | Agent, Analyste, Technicien, Administrateur | Haute |
| Gérer les Comptes | En tant qu'administrateur, je peux consulter les comptes | Administrateur | Moyenne |
| En tant qu'administrateur, je peux ajouter un compte | Administrateur | Moyenne |
| En tant qu'administrateur, je peux supprimer un compte | Administrateur | Moyenne |
| En tant qu'administrateur, je peux modifier un compte | Administrateur | Moyenne |
| En tant qu'administrateur, je peux désactiver un compte | Administrateur | Moyenne |

Notre Backlog du produit a subi plusieurs modifications tout au long de la réalisation de notre projet suite aux changements des besoins du client, auxquels nous avons pu nous adapter grâce au cadre de travail SCRUM, ce qui prouve l’aspect agile du projet.

En effet, plusieurs fonctionnalités, non mentionnés dans le cahier des charges, ont été ajoutés à savoir la fonctionnalité d’essayage des produit digitaux à partir de la plateforme aussi que la gestion des mails aussi.

Planification des sprints

Nous avons d'abord dégagé le maximum des fonctionnalités à réaliser pour former le Backlog du produit, puis défini les priorités des fonctionnalités et choisi lesquelles seront dans chaque sprint. Un sprint aboutit toujours à la livraison d’un produit partiel fonctionnel appelé incrément. Après une réunion avec l’équipe du projet, nous avons identifié 7 sprints qui constituent notre projet.

**Sprint 1** : Définition des exigences et Conception de l'architecture

* Identification des besoins et attentes des utilisateurs.
* Élaboration des user stories et des critères d'acceptation.
* Priorisation des fonctionnalités.
* Création des schémas d'architecture logicielle et matérielle.
* Définition des composants clés et de leurs interactions.
* Validation de l'architecture avec les parties prenantes.

**Sprint 2 :** Modèle de données et Mise en place de l'infrastructure

* Conception du modèle de données relationnel et/ou NoSQL.
* Définition des schémas de base de données.
* Mise en place des politiques de sécurité et de gouvernance des données.
* Configuration de l'environnement de développement.
* Déploiement initial des environnements de test et de production.

**Sprint 3 :** Développement des modules de configuration et de l'interface utilisateur

* Implémentation des modules de configuration de base.
* Écriture et exécution des tests unitaires des composants.
* Tests d'intégration pour assurer la cohérence entre frontend et backend.

**Sprint 4 :** Développement des modules de configuration et de l'interface utilisateur

* Développement des interfaces de gestion des configurations.
* Développement des composants frontend.
* Écriture et exécution des tests unitaires des composants.
* Tests d'intégration pour assurer la cohérence entre frontend et backend.

**Sprint 5 :** Tests unitaires et d'intégration

* Réalisation des tests d'intégration pour vérifier l'interaction entre les modules.
* Correction des bugs identifiés durant les tests.
* Réalisation des tests en conditions réelles.
* Collecte des retours des utilisateurs bêta.
* Ajustements finaux avant le lancement.

**Sprint 6** : Déploiement sur l'infrastructure Cloud

* Préparation des scripts de déploiement pour l'infrastructure Cloud.
* Migration des données initiales.
* Validation de l'installation sur l'infrastructure Cloud.
* Connexion et configuration des sources de données externes.
* Validation de l'intégrité et de la cohérence des données.
* Mise en place des mécanismes de synchronisation des données.

**Sprint 7** : Tests en environnement réel et documentation

* Rédaction de guides d'utilisation pour les utilisateurs finaux.
* Validation et révision de la documentation.

La planification des releases est représentée dans le tableau ci-dessous :

Tableau 9 : Tableau de planification des releases

|  |  |  |
| --- | --- | --- |
| Release | Sprint | Période de sprint |
| 1 | 1 | 13 jours (du 01/03/2024 au 14/03/2024) |
| 2 | 4 semaines (du 14/03/2024 au 10/04/2024) |
| 3 | 2 semaines (du 11/04/2024 au 24/04/2024) |
| 2 | 4 | 4 semaines (du 25/04/2024 au 23/05/2024) |
| 5 | 1 semaine (du 24/05/2024 au 01/06/2024) |
| 6 | 1 semaine (du 01/06/2024 au 08/06/2024) |
| 7 | 2 semaines (du 08/06/2024 au 22/06/2024) |

Diagramme de séquence système

Le diagramme de séquence permet de représenter une vue dynamique du système d’information. Il permet de décrire les scénarios de chaque cas d'utilisation en mettant l'accent sur la chronologie des opérations en interaction avec les objets.

* 1. Une image contenant texte, diagramme, Parallèle, ligne

     Description générée automatiquementDiagramme de séquence « Traitement automatique des courriels »

Figure 11 : Diagramme de séquence « Traitement automatique des courriels »

* Description textuelle de diagramme de séquence « Traitement automatique des courriels»  
  **Explication du diagramme de séquence**

1. **Client :**

* L'utilisateur envoie un email au système.

1. **Email Monitoring microservice :**

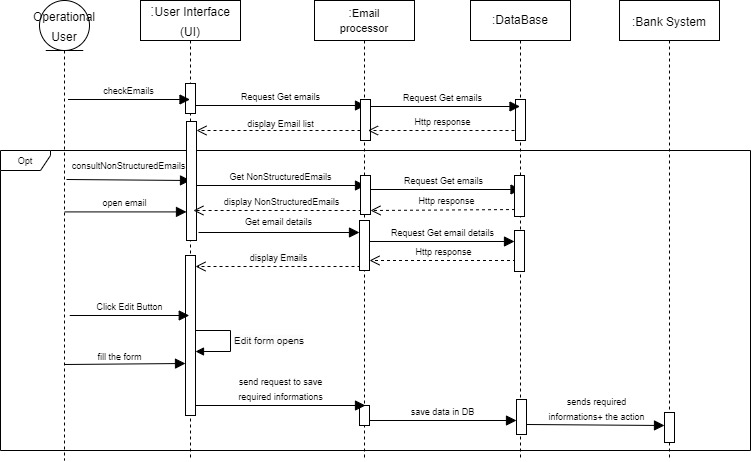
* Reçoit l'email entrant et le stocke dans une file de messages.
* Envoie l'email au microservice de traitement d'email.

1. **Email processor microservice :**

* Extrait les informations pertinentes de l'email.
* Identifie la catégorie et les actions requises.
* Envoie les informations nécessaires et les actions au système bancaire.

1. **Bank System :**

* Traite la demande.
  1. Diagramme de séquence « Traitement des Emails Non Structurés »

Figure 12 : Diagramme de séquence « Traitement des Emails Non Structurés »

* Description textuelle de cas d’utilisation « Traitement des Emails Non Structurés »

**Explication du diagramme de séquence**

1. L'opérateur initie la vérification des emails via l'interface utilisateur.
2. L'interface utilisateur reçoit l'action de l'utilisateur et envoie une requête pour obtenir les emails.
3. Le contrôleur reçoit cette requête de l'interface utilisateur et demande les emails à la base de données.
4. La base de données reçoit la requête, récupère les emails, et les envoie au contrôleur.
5. Le contrôleur affiche ensuite les emails dans l'interface utilisateur.
6. L'utilisateur choisit de consulter les emails non structurés.
7. Le système affiche ces emails en fonction des données disponibles dans l'interface.
8. L'opérateur ouvre un email, choisit de le modifier, et un formulaire s'ouvre.
9. L'opérateur saisit les nouvelles données et les envoie au contrôleur.
10. Le contrôleur envoie la requête pour sauvegarder la modification dans la base de données.
11. La base de données renvoie une confirmation de l'action au contrôleur de traitement des emails.

Conclusion

Ce chapitre a permis d'établir les bases nécessaires à la réalisation de notre projet en identifiant et spécifiant les besoins fonctionnels et non fonctionnels. Nous avons défini les rôles et responsabilités des acteurs impliqués, détaillé les User Stories et constitué le Backlog du produit.

De plus, nous avons planifié les sprints en alignement avec la méthodologie SCRUM pour garantir une livraison incrémentale et fonctionnelle de notre application. Les diagrammes de cas d'utilisation et de séquence ont été utilisés pour modéliser et visualiser les interactions et les processus clés du système, assurant ainsi une compréhension claire et partagée des exigences du projet.

CHAPITRE 3 :

ETUDE CONCEPTUELLE

Chapitre 3 : Etude conceptuelle

Introduction

Ce chapitre se concentre sur l'analyse et la conception orientée objet de notre système. Nous allons réaliser les diagrammes de cas d'utilisation majeurs, les diagrammes de classes de conception, les diagrammes de séquences, la modélisation relationnelle, ainsi que d'autres modèles UML nécessaires pour une conception complète et cohérente.

Diagramme de classes

**Une image contenant texte, diagramme, Plan, schématique

Description générée automatiquement**Ce diagramme de classes et ses descriptions montrent les relations et les interactions essentielles entre les différentes entités du système "Emails Processor", assurant une compréhension claire de la structure et du fonctionnement du système.

Figure 13 : Diagramme de classes

* Description textuelle

Tableau 10 : Tableau descriptif du diagramme de classes

|  |  |
| --- | --- |
| **Classe** | **Description** |
| **Category** | Gère les catégories d'emails. Permet de consulter, ajouter, supprimer et mettre à jour les catégories, facilitant ainsi leur traitement et classification automatique. |
| **Keyword** | Gère les mots-clés associés aux emails. Permet l'ajout, la suppression, la mise à jour et la consultation des mots-clés, essentiels pour la classification et l'analyse sémantique des emails. |
| **Email** | Représente un email avec ses attributs et états. Permet de mettre à jour le traitement des emails, vérifier les emails, et consulter les emails non structurés ou urgents. |
| **Client** | Gère les informations des clients envoyant des emails. Permet la consultation, la création, la mise à jour et la suppression des clients, facilitant la gestion des relations client. |
| **EmailProcessingRes-ult** | Stocke les résultats du traitement des emails, tels que les scores de pertinence. Permet d'évaluer l'efficacité des traitements appliqués et d'améliorer les processus de classification et de réponse. |
| **Action** | Gère les actions à exécuter sur les emails. Permet la consultation, l'ajout, la suppression et la mise à jour des actions, incluant des réponses automatiques et des mises à jour de systèmes en fonction du contenu des emails. |
| **Notification** | Gère les notifications et alertes liées aux emails. Permet de gérer les alertes envoyées aux utilisateurs pour des emails spécifiques ou des actions nécessitant une attention immédiate. |
| **SecurityRole** | Gère les rôles de sécurité et les permissions d'accès. Permet de définir et gérer l'accès aux différentes fonctionnalités du système en fonction des rôles des utilisateurs. |
| **User** | Gère les utilisateurs du système. Permet l'inscription, la connexion et la consultation du profil des utilisateurs, assurant que seuls les utilisateurs autorisés peuvent accéder et interagir avec le système. |
| **Attachment** | Gère les pièces jointes aux emails. Stocke les informations sur les fichiers joints, permettant leur consultation et gestion. |
| **RelatedData** | Stocke les données relatives associées aux emails, telles que les informations sur les comptes et les transactions. Essentielles pour le traitement contextuel des emails et les réponses automatiques précises. |
| **ResponseTemplates** | Stocke les modèles de réponse utilisés pour générer des réponses automatiques aux emails. Permet de standardiser les réponses et de gagner du temps dans le traitement des emails. |
| **ActionParam** | Stocke les paramètres des actions exécutées sur les emails. Permet de configurer les actions de manière flexible et adaptable aux besoins spécifiques du traitement des emails. |

Dictionnaire de données

* **Classe Email**

Tableau 11: Table « Email »

|  |  |
| --- | --- |
| **Attribut** | **Description** |
| emailId | Identifiant unique pour chaque courriel |
| sender | Expéditeur du courriel |
| subject | Sujet du courriel |
| content | Contenu du courriel |
| originalContent | Contenu original du courriel |
| date | Date du courriel |
| isRead | Indique si le courriel a été lu |
| isTreated | Indique si le courriel a été traité |
| urgent | Indique si le courriel est urgent |
| important | Indique si le courriel est important |
| draft | Indique si le courriel est un brouillon |
| spam | Indique si le courriel est un spam |
| archived | Indique si le courriel est archivé |

* **Classe EmailProcessingResult**

Tableau 12: Table « EmailProcessingResult »

|  |  |
| --- | --- |
| **Attribut** | **Description** |
| id | Identifiant unique pour chaque résultat |
| score | Score du traitement du courriel |

* **Classe Attachement**

Tableau 13: Table « Attachement »

|  |  |
| --- | --- |
| **Attribut** | **Description** |
| attachementId | Identifiant unique pour chaque pièce jointe |
| fileName | Nom de la pièce jointe |
| fileId | Identifiant de fichier pour la pièce jointe |
| fileContent | Contenu du fichier de la pièce jointe |

* **Classe Client**

Tableau 14: Table « Client »

|  |  |
| --- | --- |
| **Attribut** | **Description** |
| senderId | Identifiant unique pour chaque client |
| firstName | Prénom du client |
| lastName | Nom de famille du client |
| gender | Genre du client |
| SenderEmail | Adresse email de l'expéditeur |
| phone\_nbr | Numéro de téléphone de l'expéditeur |
| rib | Numéro de compte bancaire de l'expéditeur |
| iban | Numéro IBAN de l'expéditeur |
| address | Adresse du client |
| priority\_int | Niveau de priorité de l'expéditeur |

* **Classe** **RelatedData**

Tableau 15: Table « RelatedData »

|  |  |
| --- | --- |
| **Attribut** | **Description** |
| relatedDataId | Identifiant unique pour chaque donnée associée |
| account\_number | Numéro de compte associé |
| account\_type | Type de compte associé |
| période | Période associée aux données |
| amount | Montant associé aux données |
| currency | Devise associée aux données |

* **Classe Notification**

Tableau 16: Table « Notification »

|  |  |
| --- | --- |
| **Attribut** | **Description** |
| notificationId | Identifiant unique pour chaque notification |
| content | Contenu de la notification |
| urgency | Niveau d'urgence de la notification |

* **Classe User**

Tableau 17: Table « User »

|  |  |
| --- | --- |
| **Attribut** | **Description** |
| userId | Identifiant unique pour chaque utilisateur |
| firstName | Prénom de l'utilisateur |
| lastName | Nom de famille de l'utilisateur |
| email | Adresse email de l'utilisateur |
| activated | Indique si le compte utilisateur est activé |
| phone\_nbr | Numéro de téléphone de l'utilisateur |
| password | Mot de passe de l'utilisateur |
| imageUrl | URL de l'image de profil de l'utilisateur |

* **Classe securityRole**

Tableau 18: Table « securityRole »

|  |  |
| --- | --- |
| **Attribut** | **Description** |
| roleId | Identifiant unique pour chaque rôle de sécurité |
| roleName | Nom du rôle de sécurité |
| description | Description du rôle |

* **Classe** **Action**

Tableau 19: Table « Action »

|  |  |
| --- | --- |
| **Attribut** | **Description** |
| actionId | Identifiant unique pour chaque action |
| action | Description de l'action |
| descriptionAct | Détails supplémentaires décrivant l'action |
| action\_date | Date de l'action |
| updated\_at | Date de la dernière mise à jour de l'action |
| affected | Indique si l'action a été affectée |
| state | État de l'action |
| endPoint | Point de terminaison associé à l'action |
| params | Paramètres supplémentaires pour l'action |

* **Classe ActionParam**

Tableau 20: Table « ActionParam »

|  |  |
| --- | --- |
| **Attribut** | **Description** |
| **idActionParam** | Identifiant unique pour chaque action |
| **action** | **Nom de l'action** |
| **params** | **Paramètres supplémentaires pour l'action** |

* **Classe Category**

Tableau 21: Table « Category »

|  |  |
| --- | --- |
| **Attribut** | **Description** |
| categoryId | Identifiant unique pour chaque catégorie |
| title | Titre de la catégorie |
| description | Description de la catégorie |

* **Classe Keyword**

Tableau 22: Table « Keyword »

|  |  |
| --- | --- |
| **Attribut** | **Description** |
| keywordId | Identifiant unique pour chaque mot-clé |
| word | Le mot-clé utilisé |
| weight | Poids du mot-clé pour la classification |

Modélisation Relationnelle

Il est important de comprendre comment les données sont organisées et structurées pour assurer une gestion efficace et cohérente. La modélisation relationnelle peut aider à conceptualiser les relations entre différentes entités et à planifier la structure de la base de données, même dans un contexte NoSQL [14].

* + 1. Description des relations entre les tables

Tableau 23: Relations entre les Classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table Source** | **Relation** | **Table Cible** | **Type** | **Nom de l'Association** |
| Keyword | 1 -> N | Category | One-to-Many | associé |
| EmailProcessingResult | 1 -> 1 | Email | One-to-One | avoir |
| Attachement | 1 -> N | Email | One-to-Many | inclure |
| Notification | 1 -> N | Email | One-to-Many | générer |
| RelatedData | 1 -> 1 | Email | One-to-One | associées |
| Action | 1 -> 1 | Category | One-to-One | définir |
| User | 1 -> N | SecurityRole | One-to-Many | attribué |
| ResponseTemplates | 1 -> N | Action | One-to-Many | assigné |
| ActionParam | 1 -> N | Action | One-to-Many | avoir |
| Email | 1 -> N | Client | One-to-Many | envoyé par |
| Notification | 1 -> N | User | One-to-Many | envoyée à |

* + 1. Description des Relations
  + Keyword et Category
  + Relation : Une catégorie (Category) peut avoir plusieurs mots-clés (Keyword) associés.
  + Type : One-to-Many (1)
  + Description : Une catégorie de courriels peut être liée à plusieurs mots-clés pour améliorer la classification et l'analyse des courriels.
  + EmailProcessingResult et Email
  + Relation : Un courriel (Email) peut avoir un résultat de traitement (EmailProcessingResult) associés.
  + Type : One-to-One (1)
  + Description : Un courriel peut générer un résultat de traitement, reflétant différentes étapes ou méthodes de traitement appliquées.
  + Attachment et Email
  + Relation : Un courriel (Email) peut avoir plusieurs pièces jointes (Attachment) associées.
  + Type : One-to-Many (1)
  + Description : Un courriel peut inclure plusieurs pièces jointes, telles que des documents, images ou autres fichiers pertinents.
  + Notification et Email
  + Relation : Un courriel (Email) peut générer plusieurs notifications (Notification) associées.
  + Type : One-to-Many (1)
  + Description : Différentes notifications peuvent être envoyées en fonction de l'état ou du contenu du courriel.
  + RelatedData et Email
  + Relation : Un courriel (Email) peut avoir des données associées (RelatedData).
  + Type : One-to-One (1)
  + Description : Des données supplémentaires, telles que des informations de compte ou des transactions, peuvent être liées à un courriel pour fournir un contexte supplémentaire.
  + Action et Category
  + Relation : Un category doit avoir relie à une action (Action).
  + Type : One-to-one (1)
  + Description : Une action peut être définie pour chaque catégorie de courriels afin de standardiser les réponses ou traitements automatiques.
  + User et SecurityRole
  + Relation : Un rôle de sécurité (SecurityRole) peut être attribué à plusieurs utilisateurs (User).
  + Type : One-to-Many (1)
  + Description : Différents utilisateurs peuvent partager un même rôle de sécurité, définissant leurs permissions et accès dans le système.
  + ActionParam et Action
  + Relation : Une action (Action) peut avoir plusieurs paramètres (ActionParam) associés.
  + Type : One-to-Many (1)
  + Description : Différents paramètres peuvent être associés à une action pour personnaliser son exécution.
  + Email et Client
  + Relation : Un client (Client) peut être associé à plusieurs courriels (Email).
  + Type : One-to-Many (1)
  + Description : Plusieurs courriel peuvent être liés à un client, permettant une gestion efficace des interactions client.

Ces relations permettent de structurer efficacement les données et de garantir l'intégrité référentielle dans le système de gestion des courriels.

Diagrammes de séquences de conception

Les diagrammes de séquence montrent la séquence des interactions entre objets selon un point de vue temporel (chronologique). Ce diagramme permet de représenter les scénarios d’un cas d’utilisation, il permet de mieux visualiser la séquence des messages par une lecture de bas en haut. Un scénario est une instance d’un cas d’utilisation [15].

* + 1. Diagramme de séquence du cas d’utilisation « S’authentifier »

Une image contenant texte, diagramme, Parallèle, ligne

Description générée automatiquementFigure 14 : Diagramme de séquence du cas d’utilisation « S’authentifier »

* Description textuelle

Tableau 24 : Scénario du diagramme de séquence « Authentification »

|  |  |
| --- | --- |
| **Objectif** | Traiter les emails envoyés par les clients et exécuter les actions nécessaires |
| **Précondition** | L'utilisateur accède à l'écran de connexion. |
| **Post \_condition** | L'utilisateur est authentifié et reçoit un jeton de session s'il a réussi l'authentification. |
| **Scénario nominal** | 1. L'utilisateur clique sur le bouton de connexion. 2. L'écran de connexion appelle la fonction ValidateUser(). 3. Vérification des informations d'identification dans la base de données. 4. Si l'utilisateur est trouvé et le mot de passe correspond, la base de données répond avec les informations d'identification. 5. La fonction ValidateUser() vérifie le succès de l'authentification et, en cas de succès, génère un jeton basé sur les informations d'identification. 6. La fonction ValidateUser() renvoie le jeton de réponse à l'écran de connexion. 7. L'écran de connexion affiche un message de résultat à l'utilisateur. |
| **Scénario alternatif** | * Si l'authentification échoue, l'écran de connexion affiche un message d'erreur et propose de réessayer. |

**Explication du diagramme de séquence**

1. **Utilisateur :**

* L'utilisateur initie le processus d'authentification en cliquant sur le bouton de connexion.

1. **Écran de connexion :**

* L'écran de connexion reçoit l'action de l'utilisateur et appelle la fonction ValidateUser() pour vérifier les informations d'identification.

1. **Service de validation :**

* Reçoit la requête de l'écran de connexion et envoie une requête à la base de données pour vérifier l'ID utilisateur et le mot de passe.

1. **Base de données :**

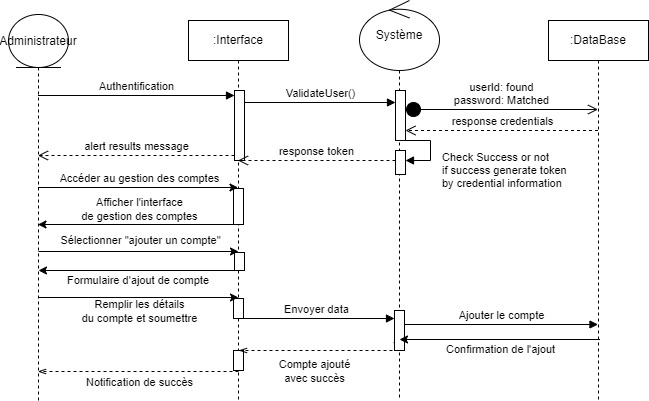
* La base de données recherche l'ID utilisateur et vérifie que le mot de passe correspond.
* Renvoie les informations d'identification trouvées à ValidateUser().

1. **Service de validation :**

* Vérifie si les informations d'identification sont valides.
* Si l'authentification est réussie, génère un token basé sur les informations d'identification.
* Renvoie le token de réponse à l'écran de connexion.

1. **Écran de connexion :**

* Reçoit le token de réponse et affiche un message de succès à l'utilisateur.
* Si l'authentification échoue, affiche un message d'erreur à l'utilisateur.
  + 1. Diagramme de séquence « Gérer les Comptes »

Figure 15 : Diagramme de séquence « Gérer les Comptes »

* Description textuelle

Tableau 25 : Scénario du diagramme de séquence « Gérer les Comptes »

|  |  |
| --- | --- |
| **Objectif** | Gérer les comptes des utilisateurs |
| **Précondition** | L'administrateur est authentifié dans le système. |
| **Post \_condition** | Les comptes des utilisateurs sont ajoutés, modifiés, supprimés ou désactivés selon l'action effectuée. |
| **Scénario nominal** | 1. L’administrateur accède à la section de gestion des comptes. 2. L’administrateur sélectionne l'option pour ajouter/modifier/supprimer/désactiver un compte. 3. L’administrateur entre les informations nécessaires ou confirme l'action. 4. Le système enregistre l'action. 5. Le compte est ajouté/modifié/supprimé/désactivé et l'utilisateur concerné reçoit une notification. |
| **Scénario alternatif** | * Si l'action échoue, le système affiche un message d’erreur et propose de réessayer ou de contacter le support technique. |

**Explication du diagramme de séquence :**

1. **Administrateur :**

* L'administrateur initie le processus en accédant à la section de gestion des comptes et en choisissant l'action souhaitée (ajouter/modifier/supprimer/désactiver un compte).

1. **Interface :**

* L'interface présente les options de gestion des comptes à l'administrateur.
* L'interface reçoit les informations saisies par l'administrateur et envoie les données au système pour traitement.

1. **Système :**

* Le système valide les informations d'identification de l'administrateur via la fonction ValidateUser().
* Si la validation est réussie, le système reçoit les données des comptes à partir de l'interface.
* Le système envoie une requête à la base de données pour ajouter/modifier/supprimer/désactiver le compte.
* La base de données confirme l'action effectuée.

1. **Base de données :**

* La base de données reçoit les requêtes du système et effectue les actions nécessaires (ajout, modification, suppression ou désactivation de compte).
* La base de données renvoie une confirmation de l'action au système.

1. **Interface :**

* L'interface affiche un message de succès à l'administrateur une fois l'action complétée.
* Si l'action échoue, l'interface affiche un message d’erreur et propose de réessayer ou de contacter le support technique.

Diagramme de composants

Nous présentons dans cette section un diagramme de composant afin d'illustrer l'organisation du système du point de vue des éléments logiciels comme les modules, les données ou encore d'éléments de configuration [16].

* **Diagramme de composants du cas d'utilisation « Traiter les Emails Non Structurés »**

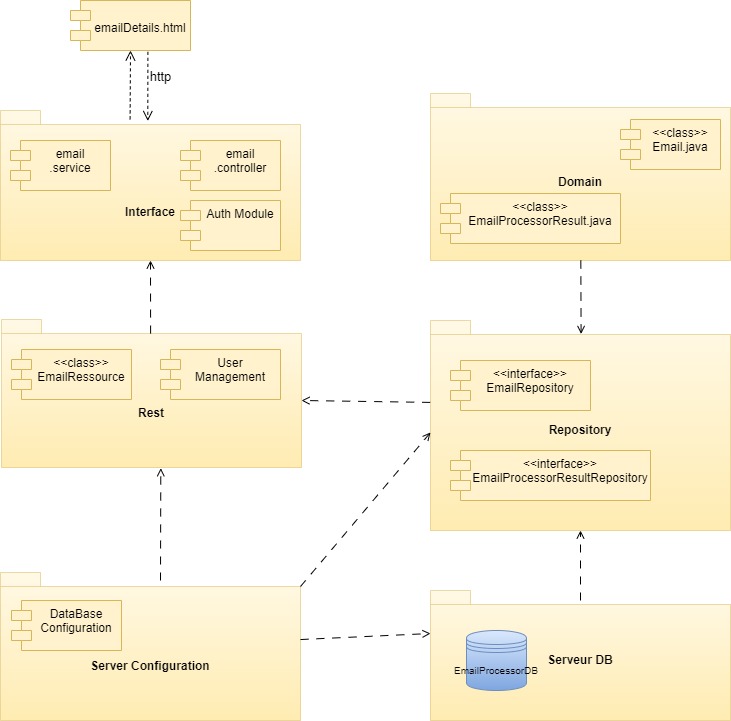


Figure 16 : Diagramme de composants du cas d'utilisation « Traiter les Emails Non Structurés »

* **Description des Composants :**

1. **Frontend**

* emailDetails.html : Interface utilisateur pour visualiser les détails des emails. Utilise HTTP pour communiquer avec le composant Interface.

1. **Interface**

* email.service : Fournit des services de traitement des emails, y compris la récupération, l'envoi et la gestion des emails.
* email.controller : Contrôleur qui gère les requêtes HTTP liées aux emails et dirige les actions appropriées vers les services correspondants.
* Auth Module : Module d'authentification gérant la connexion et la gestion des utilisateurs, incluant la vérification des identités et la gestion des sessions.

1. **Rest**

* EmailRessource : Expose les ressources email via une API RESTful, permettant la gestion et la récupération des emails à travers des endpoints REST.
* User Management : Gère les opérations CRUD (Create, Read, Update, Delete) pour les utilisateurs via des endpoints RESTful, facilitant la gestion des comptes utilisateurs.

1. **Domain**

* Email.java : Classe de domaine représentant l'entité Email, incluant ses attributs et comportements.
* EmailProcessorResult.java : Classe de domaine représentant les résultats du traitement des emails, incluant les scores et les décisions de traitement.

1. **Repository**

* EmailRepository : Interface pour accéder aux données des emails dans la base de données. Utilisée pour les opérations CRUD sur les entités Email.
* EmailProcessorResultRepository : Interface pour accéder aux résultats de traitement des emails dans la base de données. Utilisée pour les opérations CRUD sur les entités EmailProcessorResult.

1. **Server Configuration**

* DataBase Configuration : Configure l'accès et les paramètres de la base de données, assurant une connexion stable et sécurisée aux services de stockage des données.

1. **Serveur DB**

* EmailProcessorDB : Base de données où sont stockés les données.
* **Relations et Interactions :**

1. **Frontend avec Interface :**

* emailDetails.html utilise HTTP pour envoyer des requêtes à email.service et email.controller pour récupérer et afficher les détails des emails.

1. **Interface avec Rest :**

* email.controller et Auth Module utilisent les ressources exposées par EmailRessource et User Management pour gérer les emails et les utilisateurs respectivement.

1. **Rest avec Repository :**

* EmailRessource et User Management utilisent EmailRepository et EmailProcessorResultRepository pour effectuer des opérations CRUD sur les entités Email et EmailProcessorResult.

1. **Domain avec Repository :**

* Les classes de domaine Email.java et EmailProcessorResult.java définissent les structures des entités gérées par EmailRepository et EmailProcessor-ResultRepository.

1. **Server Configuration avec Serveur DB :**

* DataBase Configuration assure la configuration et la connexion de EmailRepository et EmailProcessorResultRepository avec la base de données EmailProcessorResultDB.

Ce diagramme de composants fournit une vue d'ensemble des principaux éléments de notre système et de la manière dont ils interagissent.

Conclusion

Ce chapitre a détaillé l'analyse et la conception orientée objet de notre système de gestion des courriels. En passant par les cas d'utilisation majeurs, les diagrammes de classes, les diagrammes de séquences, et la modélisation relationnelle, nous avons fourni une vue complète et cohérente du système. Cette analyse servira de base solide pour la mise en œuvre et le développement du système.

Chapitre 4 :

PRESENTATION TECHNIQUE

Chapitre 4 : Présentation technique

Introduction

Ce chapitre se concentre sur l'analyse et la conception orientée objet de notre système. Nous allons réaliser les diagrammes de cas d'utilisation majeurs, les diagrammes de classes de conception, les diagrammes de séquences, la modélisation

Choix technologiques

* + 1. Environnement technique

Dans cette partie nous présentons l'environnement matériel et l'environnement logiciel pour la réalisation du notre application.

* + - 1. Environnement matériel

Pour mener à bien la réalisation, nous avons utilisé comme environnement matériel, un poste de travail ayant les caractéristiques suivantes :

* Système d’exploitation : Windows 11
* Disque dur : 8 Go
* Ram : 16 Go
* Processeur: Intel(R) Core(TM) i5-8365U CPU @ 1.60GHz
  + - 1. Environnement logiciel

Dans cette partie, nous nous intéressons aux langages, aux bibliothèques et aux techniques de programmation utilisées tout au long de la réalisation de notre application en justifiant notre choix.

* **Outils et logiciels**
* **Draw.io** : Utilisé pour créer des diagrammes UML et autres schémas techniques, facilitant la visualisation des architectures et des flux de travail [18].

Une image contenant texte, Police, Graphique, logo

Description générée automatiquement

Figure 17 : Logo draw.io

* **IntelliJ IDEA :** Utilisé comme environnement de développement intégré (IDE) principal pour le développement Java, offrant des outils avancés pour le refactoring, le débogage, et l'intégration de systèmes de contrôle de version[19].

Une image contenant Graphique, capture d’écran, symbole, Police

Description générée automatiquement

Figure 18 : Logo IntellijIDEA

* **Visual Studio Code** : Utilisé pour le développement front-end, offrant une flexibilité avec une multitude d'extensions et une excellente prise en charge de JavaScript et HTML [20].

Une image contenant symbole, logo, Graphique, conception

Description générée automatiquement

Figure 19 : Logo Visual Studio Code

* **Postman :** Postman est une plate-forme de collaboration pour le développement d'API. Les fonctionnalités de Postman simplifient chaque étape de la création d'une API et rationalisent la collaboration afin que vous puissiez créer de meilleures API, plus rapidement [21].

Une image contenant clipart, cercle, illustration, conception

Description générée automatiquement

Figure 20 : Logo Postman

* **MongoDB Compass :** Utilisé pour interagir avec la base de données MongoDB, permettant de visualiser et d'analyser les données stockées.



Figure 21 : Logo MongoDB Compass

* **Langages :**
* Une image contenant Graphique, Police, graphisme, texte

  Description générée automatiquement**Java :** Utilisé pour le développement back-end, assurant la robustesse et la scalabilité de l'application.

Figure 22 : LOGO JAVA

* **JavaScript :** Utilisé pour le développement front-end, permettant de créer des interfaces utilisateur interactives.



Figure 23 : Logo JavaScript

* **HTML :** Utilisé pour structurer le contenu des pages web.

Une image contenant Graphique, orange, rouge, conception

Description générée automatiquement

Figure 24 : Logo HTML

* **Technologies**
* **Spring Boot :** Framework avancé qui simplifie le démarrage et le développement de nouvelles applications Java EE. Il réduit les configurations grâce à des conteneurs embarqués, permettant aux applications web de s'exécuter indépendamment sans déploiement sur un serveur web.
* **Spring Cloud :** Basé sur Spring Boot, offre une boîte à outils pour créer des systèmes d'applications distribuées rapidement.
* **Spring Data JPA :** Simplifie la mise en œuvre des couches d'accès aux données en réduisant la quantité de code à écrire. Les développeurs n'ont qu'à écrire leurs interfaces de référentiel, y compris les méthodes de recherche personnalisées, et Spring fournit automatiquement l'implémentation.
* **Spring Data MongoDB :** Fournit une couche d'abstraction pour les bases de données NoSQL, facilitant l'intégration avec le framework Spring.
* **Spring Cloud Config :** Permet de centraliser les configurations du système distribué, rendant la maintenance des microservices plus aisée.
* **Kafka :** Utilisé pour la messagerie asynchrone entre les microservices, assurant une communication fiable et scalable.
* **ModelMapper :** Utilisé pour le mapping des objets, simplifiant la conversion entre les entités de domaine et les DTOs (Data Transfer Objects).
* **MapStruct :** Un générateur de mappers code à compile-time qui simplifie les conversions entre les différentes couches de l'application.
* **Lombok :** Bibliothèque Java qui permet de réduire le code boilerplate grâce à des annotations pour générer automatiquement des getters, setters, constructeurs, et autres méthodes utilitaires.
* **SpringDoc OpenAPI :** Pour documenter les API RESTful, facilitant ainsi la compréhension et l'utilisation des services par les développeurs.
* **Stanford CoreNLP :** Utilisé pour le traitement du langage naturel (NLP), permettant d'analyser et d'extraire des informations des emails de manière automatisée.
* **Redis :** Utilisé pour la mise en cache des données et l'amélioration des performances de l'application.
* **Swagger Documentation** : Utilisé pour générer une documentation interactive des API, facilitant l'exploration et le test des endpoints pour les développeurs.
* **Framework**
* **Choix de framework Back-end :**

Avec une architecture Microservices, chaque Microservice peut être développé avec une technologie différente.

En raison de l’expérience de l’équipe, Nous avons décidé de développer les Microservices avec la technologie Java/JEE.

Par ailleurs, il existe trois Framework populaires pour développer une application en Microservices : Dropwizard, Spring et WildFlySwarm. Dans notre étude, nous somme intéresser seulement au Framework le plus mature SpringBoot.



Figure 25: Logo Springboot

C’est un Framework java créé par l’équipe Pivotal qui permet de simplifier le démarrage et le développement de nouvelles applications Spring en réduisant la complexité de configuration.

**Spring Boot offre les avantages suivants :**

* Faciliter la création des applications Spring
* Offrir un serveur d’application intégré (Tomcat, Jetty), donc nous n’avons pas besoin de la déployer en tant que fichier war.
* Fournir un "starter" POM pour simplifier la configuration Maven.
* Pas de fichiers XML à configurer.
* **Choix de Framework Front-end**

En raison de l’expérience de l’équipe, Nous avons décidé d’utiliser le Framework Angular. Par ailleurs, il existe trois Framework populaires : Angular, VueJs et React. Dans notre étude, nous nous somme intéresser seulement au Framework Angular.

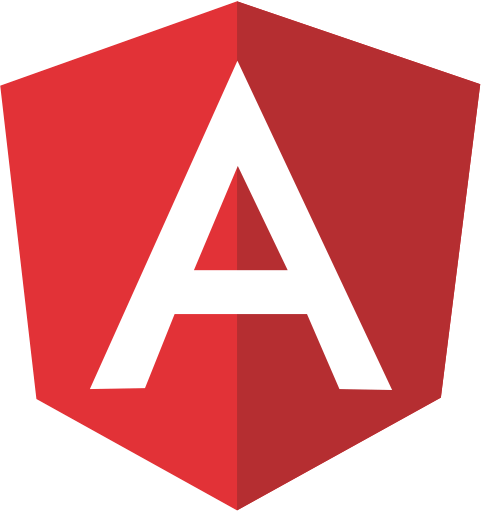


Figure 26: LOGO angular

* + 1. Schéma de développement

Dans cette partie nous expliquons le choix de l’architecture logiciel de notre travail qui se base sur le modèle MVC. L’architecture MVC est l’une des architectures les plus utilisée pour les applications web, elle se compose de trois modules [17] :

Une image contenant texte, capture d’écran, diagramme, conception

Description générée automatiquement

Figure 27 : Architecture MVC

* 1. Le Modèle (modèle de données)

Cette partie gère les données de l’application, qui eux-mêmes peuvent être exploitées sous forme de classes. Son rôle est d’aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu’elles puissent ensuite être traitées par le contrôleur. On y trouve donc les requêtes SQL.

* 1. La Vue (Présentation, Interface utilisateur)

Cette partie se concentre sur l’affichage. Elle ne fait presque aucun traitement et se contente juste de récupérer des variables pour savoir ce qui doit être affiché. On y trouve essentiellement du code HTML.

* 1. Le Contrôleur

Cette partie gère la logique du code qui prend des décisions. C’est en quelque sorte l’intermédiaire entre le Modèle et la Vue : le Contrôleur va demander au Modèle les données pour les analyser, prendre des décisions et renvoyer le texte à afficher à la Vue.

L’architecture MVC a été utilisée ici à cause de la clarté de l’architecture qu’elle impose. En effet, la modification des traitements ne change en rien la Vue.

* + 1. Architecture logique de l’application

Les micro-services représentent un style d’architecture logicielle où un ensemble complexe d’applications est décomposé en plusieurs processus indépendants et faiblement couplés, souvent spécialisés dans une seule tâche.

* **Processus indépendants :** Chaque micro-service fonctionne indépendamment et communique avec d'autres services via des API.
* **API REST :** Utilisées pour relier les micro-services, permettant à chaque service de fonctionner de manière autonome et de suivre ses propres cycles de développement.

Le tableau suivant décrit les principaux avantages des architectures micro-services par rapport à d’autres types d'architectures :

Tableau 26 : Tableau comparatif des différentes architectures

|  |  |  |  |
| --- | --- | --- | --- |
| Architecture | Micro-services | Monolithique | Modulaire |
| Liaison des services | Faiblement liés | Fortement liés | Fortement liés |
| Mise à jour | Simple | Compliqué | Simple |
| Langages | Multiples | Un seul langage | Un seul langage |

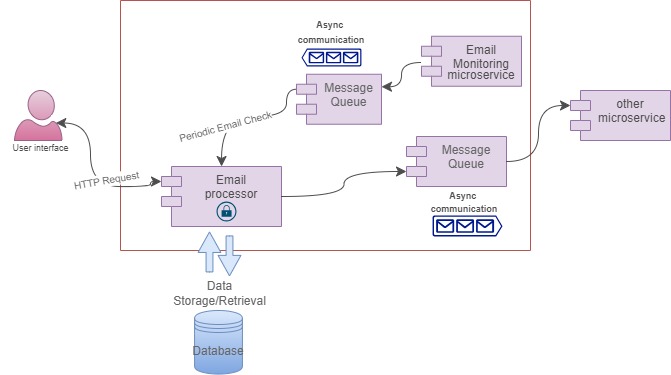


Figure 28 : Architecture Microservices

* **Description Composants principaux du système (les micro-services) :**

1. **User Interface** : Interface utilisateur permettant l’authentification et l’interaction avec le système.
   * **Rôle** : Gérer les demandes d’authentification et d’interaction utilisateur.
   * **Communication** : Synchrone avec l'Email Processor via des requêtes HTTP.
2. **Email Processor** : Service central traitant les emails.
   * **Rôle** :
     1. Surveillance Périodique : Les nouveaux e-mails sont récupérés périodiquement à l'aide de la tâche planifiée.
     2. Traiter les emails entrants, les analyser et les stocker dans la base de données.
   * **Communication : Asynchrone avec les autres micro-services via un message queue.**
3. **Email Monitoring Micro-service** : Service de collecte des emails.
   * **Rôle :**
4. Surveille la boîte de réception, extrait les nouveaux emails et envoie les données traitées via un message queue.
5. Maintient la connexion IMAP active en envoyant périodiquement une commande NOOP pour éviter la fermeture de la connexion.
   * **Communication : Asynchrone avec l'Email Processor via un message queue.**

**Workflow général :**

* **Authentification** : Les utilisateurs s’authentifient via l’interface utilisateur, recevant un jeton JWT.
* **Traitement des Emails** : L'Email Processor traite les emails et envoie les données pertinentes aux autres micro-services via un message queue.
* **Communication Asynchrone** : un message queue est utilisé pour la communication asynchrone entre les micro-services, assurant la scalabilité et la fiabilité du système.
* **Communication entre les micro-services**

Les micro-services, bien qu'autonomes, nécessitent des échanges de données pour réaliser certaines fonctionnalités. Dans une architecture en micro-services, chaque service gère ses propres données et ne les partage pas directement avec d’autres services. Les intercommunications se font de deux manières :

* **Communication synchrone**

Spring Cloud offre trois façons d’appels clients Rest :

* **Rest Template :** Méthode traditionnelle où le micro-service client de l’API doit connaître l’adresse complète du micro-service producteur.
* **Rest Template avec Eureka :** Utilisation de Netflix Ribbon et du service d’annuaire Eureka pour résoudre le problème de connaissance de l'hôte.
* **Netflix Feign et Eureka :** Solution simplifiée offrant une interface simple et des annotations paramétrées, tout en maintenant des performances similaires à Ribbon.
* **Communication asynchrone**

Utilisation de systèmes de messagerie pour des communications asynchrones, favorisées pour répondre aux exigences de temps de réponse. Nous avons choisi Apache Kafka pour sa performance supérieure par rapport à RabbitMQ et ActiveMQ. Kafka nécessite l'utilisation de Zookeeper pour la communication entre les instances. Ce service permet aux instances Kafka de communiquer entre eux. Par conséquent, nous avons utilisé le service de file d’attente Apache Kafka qui répond suffisamment aux besoins.

* **Architecture de Kafka**

Kafka est composé de plusieurs composants clés [26] :

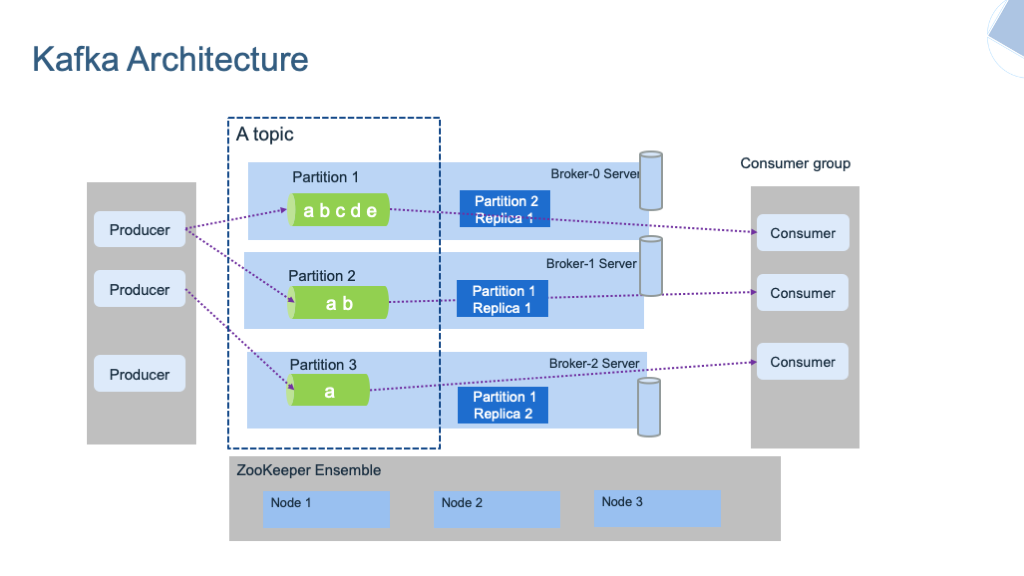
****

Figure 29 : Aperçu de l'architecture KAFKA

* **Brokers Kafka :** Les brokers sont les serveurs Kafka. Un cluster Kafka se compose de plusieurs brokers qui stockent les données et servent les producteurs et consommateurs de données.
* **Topics :** Les données sont organisées en topics. Chaque topic est partitionné, et chaque partition est répliquée pour assurer la tolérance aux pannes.
* **Producers :** Les producteurs publient des messages dans les topics de Kafka. Ils envoient les données aux partitions de manière équilibrée.
* **Consumers :** Les consommateurs lisent les messages des topics. Ils peuvent lire les données en temps réel ou rejouer les messages depuis le début du topic.
* **ZooKeeper :** Kafka utilise ZooKeeper pour la gestion de la configuration et la coordination entre les brokers. Il est responsable de l’élection du leader et du suivi des offsets.
* **Fonctionnement de Kafka**

Kafka fonctionne selon un modèle de publication-abonnement avec une persistance durable des messages.

1. **Production** de Messages : Les producteurs envoient les messages à un broker Kafka. Les messages sont écrits dans une partition spécifique du topic, soit de manière aléatoire, soit selon une clé spécifique.
2. **Stockage** et Réplication : Chaque message est stocké de manière durable sur le disque et répliqué sur d'autres brokers pour assurer la tolérance aux pannes. Le nombre de répliques est configurable par topic.
3. **Consommation de Messages :**
   * Les consommateurs lisent les messages des partitions de topics. Kafka conserve un offset pour chaque consommateur, indiquant la position de lecture actuelle. Les consommateurs peuvent relire les messages depuis n'importe quel point.
4. **Message Queue Asynchrone :**
   * Kafka utilise une file d'attente de messages pour permettre la communication asynchrone entre les microservices. Cela signifie que les producteurs peuvent envoyer des messages sans attendre que les consommateurs les traitent immédiatement, ce qui améliore la performance et la scalabilité.

* **Utilisation de Kafka dans notre Architecture**

Dans notre système, Kafka joue un rôle crucial pour assurer la communication asynchrone entre les microservices. Voici comment nous utilisons Kafka :

1. **Production d'Événements :**

Les microservices produisent des événements à chaque fois qu'une action significative se produit (par exemple, la réception d'un email).

1. **Consommation d'Événements :**

Les microservices consomment ces événements pour effectuer des traitements supplémentaires ou déclencher d'autres actions. Par exemple, un service de monitoring peut consommer les événements produits par un service d'ingestion d'emails.

1. **Réplication et Tolérance aux Pannes :**

Grâce à la réplication des partitions, nous assurons que les données sont toujours disponibles même en cas de panne d'un broker. Cela augmente la fiabilité de notre système.

1. **Scalabilité :**

Kafka permet de scalabilité horizontale. En ajoutant plus de brokers et de partitions, nous pouvons gérer une plus grande charge de travail sans dégrader les performances.

* + 1. Architecture logicielle

L’application utilise une architecture basée sur des micro-services. Cette approche permet de décomposer l'application en plusieurs services indépendants, chacun ayant une responsabilité distincte. La communication entre ces micro-services se fait de manière asynchrone à l'aide d’un message queue.

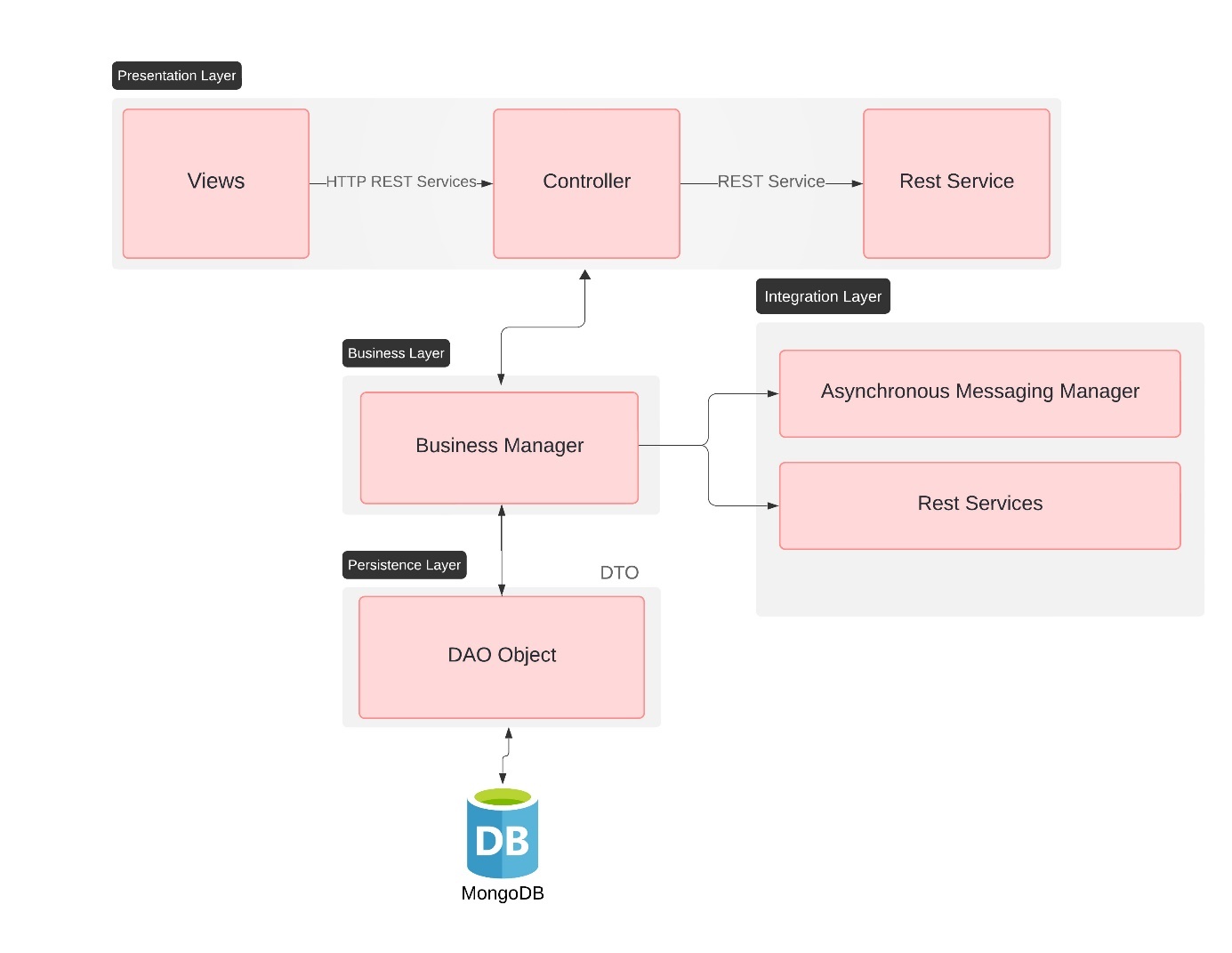


Figure 30 : ARCHITECTURE LOGICIELLE DE MICROSERVICE EMAILPROCESSOR

* + 1. Architecture Technique

Pour déployer les micro-services, nous avons opté pour les choix technologiques présentés dans l’architecture technique de la figure suivante. Chacun ayant une responsabilité distincte. La communication entre ces micro-services se fait de manière asynchrone à l'aide d’une file d'attente de messages (message queue).

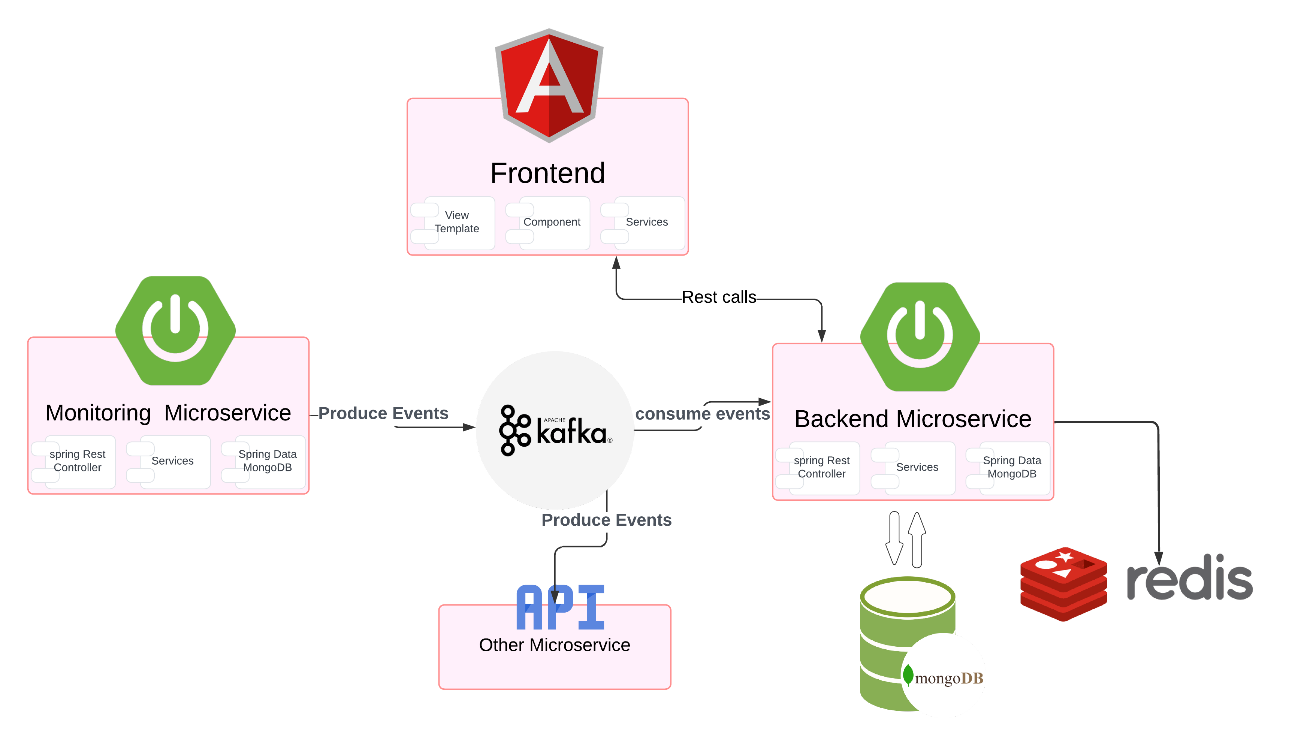


Figure 31 : ARCHITECTURE TECHNIQUE

**Composants principaux du système :**

1. **Frontend :**
   * **Technologie :** Angular
   * **Communication :** Utilise des appels REST pour interagir avec le Backend Microservice.
2. **Backend Microservice :**
   * **Technologie :** Spring Boot
   * **Communication :**
     + Consomme les événements produits par le Monitoring Microservice via Kafka.
     + Envoie des réponses aux requêtes du Frontend via des appels REST.
3. **Monitoring Microservice :**
   * **Technologie :** Spring Boot
   * **Communication :**
     + Produit des événements à Kafka pour chaque email reçu.
     + Interagit avec MongoDB pour stocker les informations des emails.
4. **Message Queue :**
   * **Technologie :** Kafka
5. **Base de données :**
   * **Technologie :** MongoDB
   * **Communication :** Les microservices interagissent avec MongoDB pour lire et écrire des données.
6. **Cache :**
   * **Technologie :** Redis
   * **Communication :** Le Backend Microservice interagit avec Redis pour mettre en cache les données fréquemment utilisées.

Processus DevOps

* + 1. Contrôle du code source

Le contrôle du code source est géré à l'aide de Git, un système de gestion de versions décentralisé. Nous utilisons des plateformes telles que GitHub et GitLab pour héberger notre code source, faciliter la collaboration en équipe, et assurer le suivi des modifications. Ces outils permettent également de gérer les branches de développement, les pull requests, et les revues de code.

* + 1. Design Patterns

L'application utilise plusieurs design patterns pour améliorer la maintenabilité et la lisibilité du code. Parmi les principaux design patterns utilisés, on trouve :

* **Singleton** : Pour garantir qu'une classe n'a qu'une seule instance et fournir un point d'accès global à cette instance.
* **Factory** : Pour créer des objets sans spécifier la classe exacte de l'objet qui sera créé.
* **Repository** : Pour encapsuler la logique de stockage, de récupération et de recherche qui émane directement du modèle.
* **Service** : Pour contenir la logique métier et assurer une séparation claire des préoccupations.
  + 1. Tests continus

Les tests sont une partie cruciale de notre processus de développement. Nous utilisons :

* **Tests unitaires** : Pour vérifier le bon fonctionnement des composants individuels de l'application. Nous utilisons JUnit pour écrire et exécuter des tests unitaires.
* **Tests d'intégration** : Pour vérifier l'interaction entre plusieurs composants de l'application. Nous utilisons des frameworks comme Spring Test pour écrire des tests d'intégration.
* **Tests de microservice "EmailProcessor-Service" :**
  + **Tests Unitaires : Test de chargement du contexte (contextLoads) :** Ce test unitaire vérifie que le contexte de l'application se charge correctement et que les composants principaux, notamment « EmailMonitoringService » et « Session », sont bien instanciés et injectés. Cela garantit que l'environnement de test est correctement configuré et que les dépendances nécessaires sont disponibles.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 32: la réussite des tests unitaires du microservice "EmailProcessor-Service"

* + **Tests d’Intégration : Test de récupération des nouveaux emails :   
    Ce test d'intégration valide la fonctionnalité principale du service, à savoir la récupération des nouveaux emails. Le service est testé pour s'assurer qu'il peut correctement se connecter au serveur de messagerie configuré et récupérer les emails non lus. Des assertions et des vérifications supplémentaires peuvent être ajoutées pour valider le contenu et le format des emails récupérés, garantissant ainsi le bon fonctionnement de l'intégration avec le serveur de messagerie.**

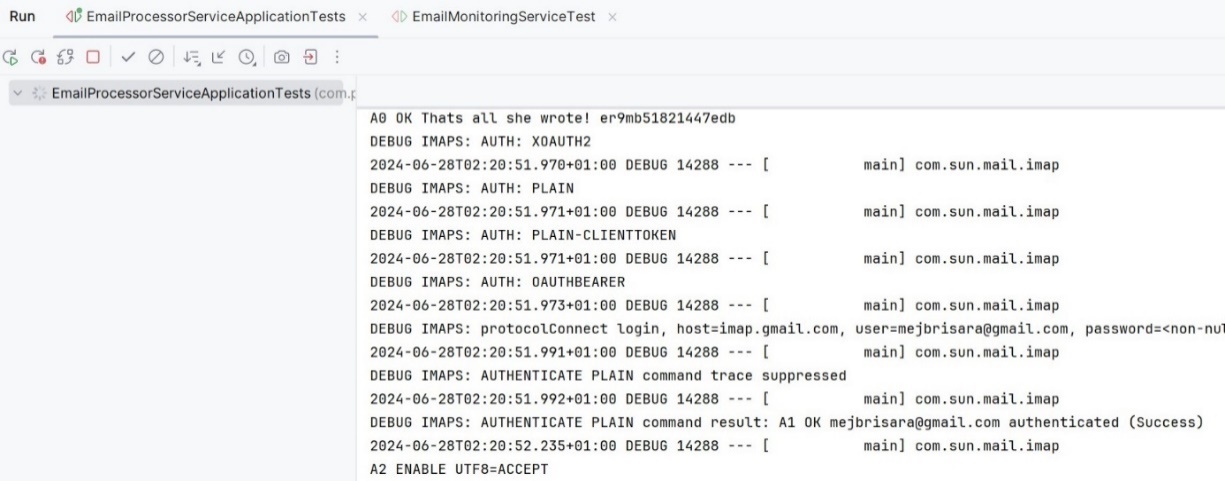




Figure 33 : la réussite des tests d'intégration du microservice "EmailProcessor-Service"

Jeu d'essais et manuel d’utilisateur

* + 1. API – Interfaces de programmation

Une image contenant texte, capture d’écran, nombre, logiciel

Description générée automatiquementUne image contenant texte, Police, ligne, capture d’écran

Description générée automatiquementCes captures d'écran montrent une requête POST envoyée à l'endpoint /login avec les informations d'identification de l'utilisateur. La réponse JSON inclut les détails de l'utilisateur, tels que l'email, le numéro de téléphone, les rôles attribués, et le jeton JWT

Figure 34 : EndPoint pour authentification

Figure 35: Code de l'endpoint de connexion dans le contrôleur

Figure 36 : EndPoint pour authentification

* + 1. Parties Front-end/Back-endUne image contenant texte, capture d’écran, conception

       Description générée automatiquement

Figure 37: Formulaire de connexion utilisateur

L'interface d'authentification permet aux utilisateurs de se connecter à l'application en saisissant leur nom d'utilisateur (adresse email) et leur mot de passe. Si les informations d'identification saisies existent dans la base de données du module, l'utilisateur sera redirigé vers le tableau de bord. Sinon, l'accès est refusé et un message d'erreur est affiché.

Une image contenant capture d’écran, texte, logiciel, Icône d’ordinateur

Description générée automatiquement

Figure 38: Interface du tableau de bord administrateur

Le tableau de bord offre une vue d'ensemble des statistiques et des actions disponibles pour l'administrateur.

Une image contenant capture d’écran, texte, logiciel, nombre

Description générée automatiquement

Figure 39 : Interface de la boîte de réception du service de messagerie

Une image contenant texte, logiciel, capture d’écran, Icône d’ordinateur

Description générée automatiquement

Figure 40 : Interface de gestion des catégories

L'interface de gestion des catégories permet à l'administrateur de voir, ajouter et modifier des catégories d'actions. Chaque catégorie est présentée avec son nom, le nombre de mots-clés associés, et les actions qui lui sont attribuées.

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

L'interface affiche un formulaire pour ajouter ou modifier une catégorie, comprenant les champs pour le titre, une description courte et l'action associée. Les boutons "add a Keyword" et "Submit" permettent de finaliser l'ajout ou la modification de la catégorie. En dessous, les catégories existantes sont affichées sous forme de cartes, chacune indiquant le nombre de mots-clés et l'action associée.

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Figure 41: Interface de Gestion des Contacts

L'interface de gestion des contacts permet à l'administrateur de gérer les informations de contact. Chaque contact est affiché avec son prénom, nom de famille, email, numéro de téléphone, et adresse. L'administrateur peut ajouter, modifier ou supprimer des contacts.

Une image contenant capture d’écran, logiciel, ligne, texte

Description générée automatiquement

Figure 42: Interface de gestion des rôles

L'interface de gestion des rôles permet aux administrateurs de créer, modifier et supprimer des rôles pour les utilisateurs de l'application. Cette section présente une vue d'ensemble des rôles existants ainsi qu'un formulaire pour ajouter ou modifier un rôle.

Une image contenant texte, capture d’écran, nombre

Description générée automatiquement

Figure 43 : Interface de gestion du profil utilisateur

L'interface de profil utilisateur permet aux administrateurs de créer, modifier et gérer les profils des utilisateurs de l'application

Conclusion

Ce chapitre a présenté une vue détaillée de l'architecture technique et des choix technologiques de notre système. Nous avons exploré l'architecture logicielle en nous appuyant sur le modèle MVC et l'architecture trois tiers, détaillant les niveaux client, serveur d'application et serveur de base de données. Nous avons également mis en lumière les processus DevOps, incluant le contrôle du code source, les tests continus et l'approche CI/CD, qui sont essentiels pour une intégration et un déploiement fluide. Les technologies utilisées, telles que Spring Boot, Spring Cloud, Kafka, et Redis, ont été soigneusement sélectionnées pour garantir la robustesse, la scalabilité et la maintenabilité du système. Cette présentation technique fournit une base solide pour la mise en œuvre et le développement du système, assurant ainsi sa performance et son adaptabilité aux besoins futurs.

Conclusion générale et perspectives

Notre projet de fin d'études a été réalisé dans le cadre de l'obtention du diplôme d'ingénierie en génie logiciel, effectué au sein de la société VERMEG. Ce projet a consisté en la conception et le développement d'une application web pour le traitement automatisé des courriels dans les institutions bancaires, visant à optimiser la gestion des communications électroniques, améliorer l'efficacité opérationnelle et réduire les coûts.

Malgré les difficultés rencontrées, notamment la complexité des systèmes de supervision, nous avons réussi à atteindre les objectifs établis. En adoptant la méthodologie Scrum, nous avons pu gérer efficacement les différentes étapes du projet. Notre rapport a détaillé ces étapes, en commençant par l'étude de l'existant, l'identification des objectifs du projet, et la proposition de notre solution.

Nous avons ensuite planifié le travail en identifiant les différents modules de notre projet et en respectant la priorité des besoins. Les sprints ont été accompagnés d'études spécifiques et conceptuelles lorsque nécessaire. Le rapport a également inclus une description de l'environnement de travail matériel et logiciel, des outils utilisés, des technologies adoptées, et de l'architecture de notre application.

En termes de réalisation, nous avons répondu amplement aux objectifs établis par le service. Les limites rencontrées, notamment la contrainte de temps, ont été un obstacle à l'ajout de certaines fonctionnalités supplémentaires. Toutefois, ce projet nous a permis de mettre en pratique et de renforcer nos connaissances en informatique, tout en développant des compétences précieuses en gestion de projet et en développement logiciel.

Comme perspectives, nous proposons de poursuivre l'amélioration de l'application en intégrant des fonctionnalités avancées pour la gestion des contrôles, ainsi que l'optimisation continue des performances du système. Nous envisageons également d'explorer de nouvelles techniques de traitement du langage naturel (NLP) pour améliorer encore la précision et l'efficacité du traitement des courriels.

En conclusion, ce projet a été une expérience enrichissante qui nous a permis de démontrer notre capacité à concevoir, développer et déployer une solution logicielle complexe dans un environnement professionnel.

WEBOGRAPHIE

1. [RESEARCHGATE], [https://www.researchgate.net/publication/365233503\_ The\_International\_Journal\_of\_ Business\_and\_Management\_Research](https://www.researchgate.net/publication/365233503_%20The_International_Journal_of_%20Business_and_Management_Research) Consulté le 05/02/2024
2. [VERMEG] <https://www.vermeg.com/fr/a-propos-de-nous/> Consulté le 07/02/2024
3. [MCKINSEY], <https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/response-and-resilience-in-operational-risk-events> Consulté le 09/02/2024
4. [MDPI], <https://www.mdpi.com/2674-1032/2/1/8> Consulté le 09/02/2024
5. [METHODES-AGILES], [https://www.ideematic.com/actualites/2015/01/ methodes-agiles-definition/](https://www.ideematic.com/actualites/2015/01/%20methodes-agiles-definition/) Consulté le 10/02/2024
6. [PENTALOG], <https://www.pentalog.fr/notre_demarche/methode-agile-scrum.htm> Consulté le 10/02/2024
7. [UIPATH], <https://www.uipath.com/> Consulté le 15/02/2024
8. [MICROSOFT], <https://www.microsoft.com/fr-fr/power-platform/products/power-automate> Consulté le 15/02/2024
9. [GOOGLE], <https://cloud.google.com/natural-language/?hl=fr> Consulté le 15/02/2024
10. [CAMUNDA], <https://camunda.com/solutions/industry/insurance/> Consulté le 15/02/2024
11. [APPGANTT], <https://app.gantt.io/> Consulté le 20/02/2024
12. [MEDIUM], <https://medium.com/@sadikarahmantanisha/the-mvc-architecture-97d47e071eb2> Consulté le 02/03/2024
13. [WAYOFCODE], <http://wayofcode.fr/implmentation-de-services-web-rest-avec-le-framework-spring/> Consulté le 05/03/2024
14. [DEVELOPPEZ], <http://laurent-audibert.developpez.com/Cours-UML/?page=mise-en-oeuvre-uml> Consulté le 05/03/2024
15. [DOCPLAYER] <http://docplayer.fr/13702624-Chapitre-3-le-modele-relationnel.html> Consulté le 15/03/2024
16. [DEVELOPPEZ], <https://laurent-audibert.developpez.com/Cours-UML/?page=diagrammes-composants-deploiement> Consulté le 15/05/2024
17. [WIKIMONDE], <https://wikimonde.com/article/Architecturetroistiers> Consulté le 05/03/2024
18. [DRAWIO] <https://www.drawio.com/> Consulté le 05/04/2024
19. [JETBRAINS], <https://www.jetbrains.com/idea/> Consulté le 18/04/2024
20. [VISUALSTUDIO], <https://code.visualstudio.com/> Consulté le 18/04/2024
21. [POSTMAN], <https://www.postman.com/company/about-postman/> Consulté le 28/04/2024
22. [MONGODB]<https://www.mongodb.com/products/tools/compass> Consulté le 05/04/2024
23. [SLIDESHARE], <https://fr.slideshare.net/slideshow/rapport-de-pfe-du-diplme-de-mastre-pro-en-modlisation-bases-de-donnes-et-intgration-des-systmes/103496281#13> Consulté le 05/06/2024
24. [ACADEMIA], [https://www.academia.edu/36171921/Rapport\_de\_ Projet\_de\_Fin\_dEtudes\_Conception\_et\_r%C3%A9alisation\_dune\_application\_Web](https://www.academia.edu/36171921/Rapport_de_%20Projet_de_Fin_dEtudes_Conception_et_r%C3%A9alisation_dune_application_Web) Consulté le 15/06/2024
25. [SLIDESHARE], [https://fr.slideshare.net/slideshow/rapport-pfe- talan2018doniahammami/103959267#28](https://fr.slideshare.net/slideshow/rapport-pfe-%20talan2018doniahammami/103959267#28) Consulté le 17/06/2024
26. [GITHUB], <https://ibm-cloud-architecture.github.io/refarch-eda/technology/kafka-overview/> Consulté le 17/06/2024