EisenFlow : Gestion Intelligente d'Emails

Projet DevOps 1 - Documentation Technique

Alex BRINDUSOIU Arthur CHAUVEAU Swetha SARAVANAN
Franck ZHENG

Février 2026

Table des matières

1 Présentation Générale

1.1 1. Qui ?

EisenFlow a été créé pour aider les personnes qui reçoivent beaucoup d'emails et qui ont du mal à tout gérer. Le but est de réduire le stress et de mieux organiser le travail grâce à l'Intelligence Artificielle.

1.2 2. Quoi ? Pourquoi ?

L'objectif est d'appliquer la méthode de productivité *Eisenhower* directement dans le flux de travail des emails, en utilisant une IA locale pour garantir la **confidentialité** des données utilisateur.

1.3 3. Description de l'application

Notre application permet à l'utilisateur de charger ses mails dans notre application. Cette application va ensuite donner la possibilité de soit classer automatiquement les mails dans la Matrice Eisenhower pour que les mails se fassent taguer, soit de laisser classer et taguer les mails manuellement par l'utilisateur.

Parmi les fonctionnalités présentes dans notre application il y a la possibilité d'envoyer les résultats dans Gmail directement, classer et taguer les mails.

Dans les tags disponibles, il y a le DO, PLAN, DELEGATE et DELETE. Le **DO** est le tag qui regroupe les tâches urgentes et importantes nécessitant une action immédiate.

L'application offre deux modes d'utilisation : * **Classement automatique :** L'IA analyse le contenu et applique les tags de la Matrice Eisenhower. * **Classement manuel :** L'utilisateur garde le contrôle total sur le tagage.

1.3.1 Fonctionnalités clés

L'application permet d'envoyer les résultats directement dans Gmail. Les tags disponibles sont :

1. **DO** : Tâches urgentes et importantes.
2. **PLAN** : Tâches importantes mais non urgentes.
3. **DELEGATE** : Tâches urgentes mais non importantes.
4. **DELETE** : Tâches ni urgentes ni importantes.

1.4 4. Analyse de Concurrence

Parmi les concurrents de notre application, on retrouve plusieurs applications largement utilisées qui proposent des fonctionnalités similaires. Par exemple, les boîtes mails classiques comme Gmail, Outlook ou AppleMail, qui permettent la gestion centralisée des mails, d'afficher des messages, de créer des dossiers ou encore de créer des règles manuelles. Contrairement à notre application, Gmail vise à faciliter la communication et l'organisation de l'information pour les utilisateurs mais elle ne permet pas le tri de mails automatique (pour l'instant).

Un autre type d'applications parmi les boîtes mails sont celles utilisant l'IA comme SuperHuman. L'IA peut résumer un mail et écrire une réponse. SuperHuman a donc quelques ressemblances par rapport à notre

application (utilisation de l'IA ainsi que réponse qui ressemble à notre DELEGATE automatique) mais contrairement à nous, SuperHuman n'utilise pas une IA locale mais envoie les mails sur des serveurs en ligne. Si l'utilisateur souhaite de la confidentialité, ça serait une problématique.

Il y a aussi les applications de gestion de projet comme Jira qui proposent des tableaux Kanban, permettent de suivre les tâches et sont donc très utiles pour le travail en équipe. Notre application permet de suivre les tâches lues dans les mails, contrairement à Jira qui nécessite de mettre manuellement chaque tâche.

> **Note sur la confidentialité :** Contrairement à SuperHuman, EisenFlow privilégie une **IA locale** pour protéger la vie privée des utilisateurs.

2 Ingénierie et Fonctionnalités

2.1 5. Éléments de gestion de projet et Architecture technique

Dans le cadre de ce projet, nous avons mis en place une organisation d'équipe stricte (réunions hebdomadaires, répartition claire des rôles : Backend, UI, QA) et une approche DevOps robuste :

- **Backend & IA Locale :** Le cœur de l'application repose sur **Java 21** et **Spring Boot 3.x**. L'extraction et la relève sécurisée des mails se font via **Jakarta Mail**. L'IA, garantissant 100% de confidentialité, est propulsée localement par **Ollama** (modèle *TinyLlama*).
- **Collaboration & DevOps :** Nous utilisons la méthodologie Agile (Kanban) avec **GitHub** (Issues, PRs). L'intégration continue (CI/CD) est gérée par GitHub Actions pour automatiser la compilation et les tests. Le suivi strict de la qualité du code et des vulnérabilités est assuré par **SonarCloud**, **Jacoco**, et **JUnit 5**.
- **Architecture & Patterns :** Afin de garantir un découplage strict et une flexibilité maximale (par exemple pour changer de modèle d'IA facilement), nous avons implémenté une architecture **MVC** soutenue par les design patterns **Strategy**, **Facade**, et **Registry**.

2.2 6. Diagramme de classes

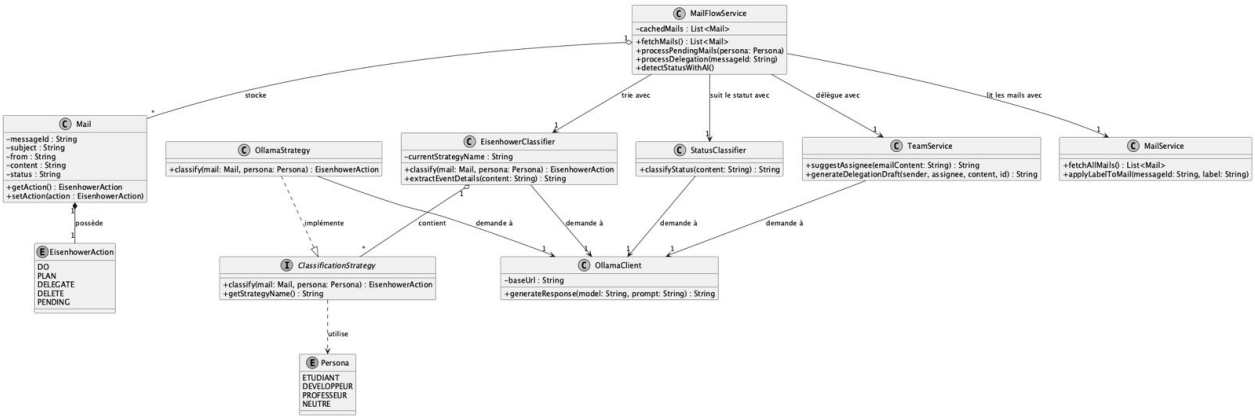

Figure 1: Diagramme de classes

2.2.1 6.1 But des features

F1 : Synchronisation Emails * **Objectif :** Maintenir le Zéro Inbox et lire les messages de façon sécurisée. * **Implémentation :** Utilisation de `jakarta.mail` (IMAP/SMTP) pour la lecture, la création de labels et le déplacement de mails (parsing MimeMultipart).

F2 : Tri Intelligent (IA) * **Objectif :** Automatiser la priorisation selon la matrice d'Eisenhower. * **Implémentation :** Analyse sémantique via IA locale (Ollama) prenant en compte le Persona de l'utilisateur. Utilisation du Pattern *Strategy*.

F3 : Tags Personnalisés * **Objectif :** Adapter l'outil aux processus spécifiques de l'utilisateur. *

Implémentation : Création/suppression de tags sur-mesure validés par contrôles Regex et sauvegardés en JSON via la librairie *Jackson*.

F4 : Délégation Assistée * **Objectif :** Faciliter le transfert des tâches à l'expert adéquat. * **Implémentation :** Routage hybride (mots-clés + IA locale) pour générer un brouillon automatique. Utilisation du Pattern *Adapter*.

F5 : Préparation Réunions * **Objectif :** Fournir un contexte immédiat avant un rendez-vous. * **Implémentation :** Synthèse de l'historique des échanges par l'IA et génération d'un mémo au format PDF à l'aide de flux binaires et de la librairie *OpenPDF*.

F6 : Agenda & Rendez-vous * **Objectif :** Extraire et planifier automatiquement les événements. * **Implémentation :** Détection de dates et lieux via IA locale (NER - *Named Entity Recognition*) et parsing de dates par Regex pour proposer une insertion dans l'agenda.

2.2.2 6.2 Scénarios

- **Scénario F1 & F2 (Synchronisation et Tri) :** L'utilisateur connecte son compte. Le backend relève les mails via Jakarta Mail. L'IA Ollama analyse sémantiquement chaque message en fonction du persona choisi et les répartit dans les colonnes du Kanban (DO, PLAN, etc.).
- **Scénario F4 (Délégation) :** L'utilisateur reçoit une demande technique qu'il ne peut pas traiter. L'application utilise le routage hybride pour identifier l'expert concerné, prépare un brouillon de réponse expliquant le contexte, prêt à être envoyé.
- **Scénario F5 & F6 (Réunion et Agenda) :** Un fil de mails contient une invitation pour un point de synchronisation vendredi à 10h. L'IA (via NER) extrait la date, propose d'ajouter le rendez-vous à l'agenda (F6) et génère simultanément un PDF via OpenPDF résumant les échanges précédents pour préparer la réunion (F5).

2.2.3 6.3 Wireframe et screenshots

(À compléter par vos maquettes et captures d'écran de l'application)

2.3 7. Résumé des features

L'application repose sur un cœur Java/Spring Boot solide orchestrant 6 fonctionnalités principales :

1. **Synchronisation :** Relève sécurisée (IMAP/SMTP) pour tendre vers le Zéro Inbox.
2. **Tri IA :** Classification Eisenhower sémantique gérée localement par TinyLlama.
3. **Tags personnalisés :** Flexibilité du système gérée via JSON.
4. **Délégation :** Routage et brouillons assistés par IA et mots-clés.
5. **Préparation :** Mémos PDF générés automatiquement.
6. **Agenda :** Détection d'entités (NER) pour planifier les évènements.

2.4 8. Annexe API REST

(Détails des endpoints du backend Spring Boot à insérer ici)