

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	2
1.1 1. Qui ?	2
1.2 2. Quoi ? Pourquoi ?	2
1.3 3. Description de l'application	2
1.3.1 Fonctionnalités clés	2
1.4 4. Analyse de Concurrence	2
2 Ingénierie et Fonctionnalités	4
2.1 5. Éléments de gestion de projet et Architecture technique	4
2.2 6. Diagramme de classes	4
2.2.1 6.1 But des features	4
2.2.2 6.2 Scénarios	6
2.2.3 6.3 Wireframe et screenshots	6
2.3 7. Résumé des features	7
3 8. Annexe API REST	7
3.1 Documentation de l'API Mails	7
3.1.1 1. Actions Globales (Collection)	7
3.1.2 2. Actions Spécifiques (Ressource Unique)	8
3.1.3 3. Actions de Délégation	8
3.2 Documentation des Contrôleurs (Application Web)	8
3.2.1 1. Vues Principales (Navigation)	9
3.2.2 2. Actions sur les Mails	9
3.2.3 Délégation	9
3.2.4 3. Événements et Planification	10
3.2.5 4. Base de Connaissances (Knowledge)	10
3.2.6 5. Gestion des Tags (Tags & API)	11

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

2.2.1 6.1 But des features

F1 : 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*.

F2 : Traitement et fusion de fichier markdown * Objectif : Centraliser et assembler dynamiquement plusieurs fichiers Markdown fragmentés en un document unique et structuré pour générer le livrable final. *
Implémentation : Un script parcourt les dossiers cibles, nettoie les métadonnées (frontmatter) et concatène les fichiers .md dans un ordre défini avant de passer le résultat au compilateur Pandoc.

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.

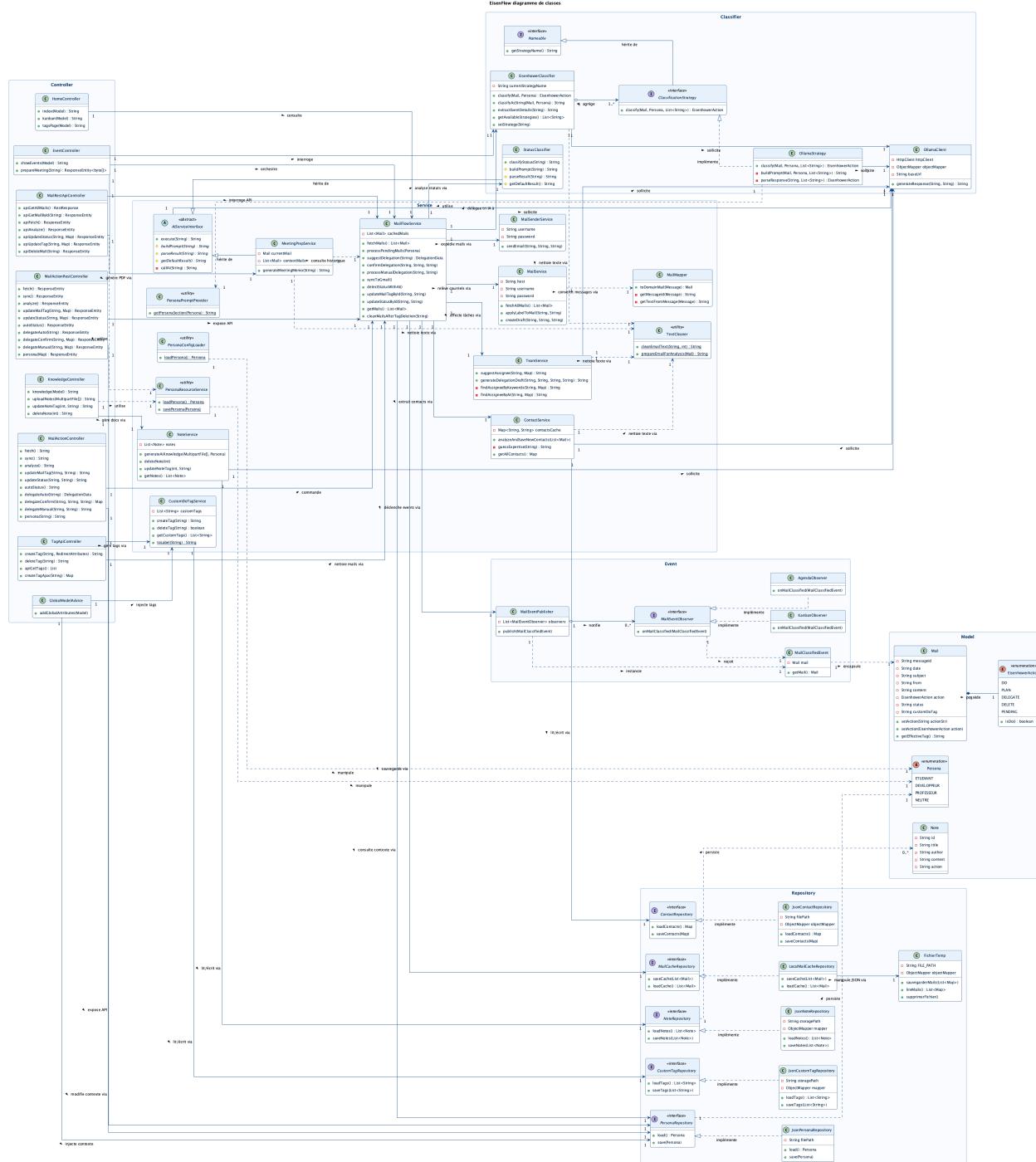


Figure 1: Diagramme de classes

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

F1 : Traitement et fusion de fichier markdown F2 : Tri Intelligent (IA) F3 : Tags Personnalisés

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.
-

3 8. Annexe API REST

3.1 Documentation de l'API Mails

URL de base : /api-mails

Format des requêtes et réponses : application/json

3.1.1 1. Actions Globales (Collection)

Ces routes agissent sur l'ensemble de la boîte mail ou sur la configuration globale.

Méthode	Route	Description	Corps (Body)
GET	/	Récupère tous les emails avec leurs liens d'actions (HATEOAS).	<i>Aucun</i>
POST	/fetch	Récupère les nouveaux emails depuis la source.	<i>Aucun</i>
POST	/sync	Synchronise avec Gmail puis récupère les nouveaux emails.	<i>Aucun</i>
POST	/analyze	Analyse et classifie automatiquement les emails en attente (PENDING).	<i>Aucun</i>
POST	/auto-status	Met à jour automatiquement les statuts grâce à l'IA.	<i>Aucun</i>
PUT	/persona	Met à jour le Persona de l'utilisateur.	{"persona": "NOM_DU_PERSONA"}

3.1.2 2. Actions Spécifiques (Ressource Unique)

Ces routes ciblent un email précis grâce à son identifiant (`{id}`).

Méthode	Route	Description	Corps (Body) Requis
GET	<code>/{{id}}</code>	Récupère les détails d'un email spécifique.	<i>Aucun</i>
PUT	<code>/{{id}}/status</code>	Modifie le statut d'un email (ex: TODO, DONE).	<code>{"status": "DONE"}</code>
PUT	<code>/{{id}}/tag</code>	Ajoute ou modifie un tag sur l'email.	<code>{"tag": "URGENT"}</code>
DELETE	<code>/{{id}}</code>	Marque l'email comme supprimé (Action = DELETE).	<i>Aucun</i>

3.1.3 3. Actions de Délégation

Routes dédiées au processus de délégation (Eisenhower : Delegate).

Méthode	Route	Description	Corps (Body) Requis
POST	<code>/{{id}}/delegate-auto</code>	Génère une suggestion de délégation via l'IA.	<i>Aucun</i>
POST	<code>/{{id}}/delegate-confirm</code>	Valide la délégation avec l'assigné et le brouillon.	<code>{"assignee": "email@...", "draftBody": "texte..."}</code>
POST	<code>/{{id}}/delegate-manual</code>	Force une délégation manuelle sans brouillon IA.	<code>{"assignee": "email@..."}</code>

Note sur le format de réponse (HATEOAS) : > La plupart des requêtes GET (et certaines PUT) renvoient l'objet modifié accompagné d'un noeud `_links`. Ces liens indiquent au front-end quelles sont les prochaines actions possibles en fonction de l'état actuel de l'email.

3.2 Documentation des Contrôleurs (Application Web)

Ce document liste l'ensemble des routes exposées par les contrôleurs de l'application. La majorité des routes utilisent le pattern MVC classique (redirection vers des vues ou d'autres pages), avec quelques endpoints spécifiques renvoyant des données brutes (JSON ou PDF).

3.2.1 1. Vues Principales (Navigation)

Géré par `HomeController`. Ces routes permettent d'afficher les différentes pages de l'interface utilisateur.

Méthode	Route	Description	Paramètres
GET	/	Affiche la page d'accueil (Boîte de réception / Inbox).	<i>Aucun</i>
GET	/kanban	Affiche la vue Kanban (tâches en cours et finalisées).	<i>Aucun</i>
GET	/tags	Affiche la page de gestion des tags personnalisés.	<i>Aucun</i>

3.2.2 2. Actions sur les Mails

Géré par `MailActionController`. Ces routes traitent les formulaires d'action sur les emails et redirigent vers les vues.

Méthode	Route	Description	Paramètres attendus
POST	/fetch	Récupère les nouveaux mails depuis la source.	<i>Aucun</i>
POST	/sync	Synchronise avec Gmail et récupère les mails.	<i>Aucun</i>
POST	/analyze	Lance l'analyse IA des mails en attente.	<i>Aucun</i>
POST	/auto-status	Met à jour les statuts automatiquement avec l'IA.	<i>Aucun</i>
POST	/update-mail-tag	Met à jour le tag d'un mail spécifique.	<i>messageId, tag</i>
POST	/update-status	Met à jour le statut d'un mail.	<i>messageId, status</i>
POST	/persona	Modifie le Persona global de classification.	<i>persona</i>

3.2.3 Délégation

Méthode	Route	Description	Paramètres attendus	Retour
POST	<code>/delegate-auto</code>	Suggère une délégation via l'IA.	<code>messageId</code>	JSON
POST	<code>/delegate-confirm</code>	Confirme la délégation avec le brouillon.	<code>messageId,</code> <code>assignee,</code> <code>draftBody</code>	JSON
POST	<code>/delegate-manual</code>	Force une délégation sans brouillon IA.	<code>messageId,</code> <code>assignee</code>	Redirection

3.2.4 3. Événements et Planification

Géré par `EventController`. Permet de gérer les mails classifiés comme “À planifier”.

Méthode	Route	Description	Paramètres attendus	Retour
GET	<code>/events</code>	Affiche la liste des événements et tâches à faire.	<i>Aucun</i>	Vue HTML
POST	<code>/events/prepare</code>	Génère une fiche de préparation de réunion.	<code>messageId</code>	Fichier PDF

3.2.5 4. Base de Connaissances (Knowledge)

Géré par `KnowledgeController`. Permet de gérer les notes et fichiers de contexte pour l'IA.

Méthode	Route	Description	Paramètres attendus
GET	<code>/knowledge</code>	Affiche la base de connaissances.	<i>Aucun</i>
POST	<code>/knowledge/upload</code>	Upload et génère des notes IA depuis des fichiers.	<code>files</code> (Multipart)
POST	<code>/update-note-tag</code>	Modifie le tag d'une note spécifique.	<code>index, tag</code>
POST	<code>/knowledge/delete</code>	Supprime une note de la base.	<code>index</code>

3.2.6 5. Gestion des Tags (Tags & API)

Géré par `TagApiController`. Gère la création et la suppression des étiquettes personnalisées.

Méthode	Route	Description	Paramètres attendus	Retour
GET	<code>/api/tags</code>	Récupère la liste de tous les tags.	<i>Aucun</i>	JSON
POST	<code>/tags/create</code>	Crée un nouveau tag via le formulaire classique.	<code>label</code>	Redirection
POST	<code>/tags/create-ajax</code>	Crée un nouveau tag de manière asynchrone (AJAX).	<code>label</code>	JSON
POST	<code>/tags/delete</code>	Supprime un tag et nettoie les mails associés.	<code>tagName</code>	Redirection