

Rapport SAE LaPoste



i-TEAM

Direction des services IT Groupe

IUT du Limousin
BUT Informatique



**Université
de Limoges**

18/03/2024

Mohamed MESRI, Yassine SADDIKI, Jimmy LEGG, Leo CONDAT, Cyril ROMERO

Sommaire :

I.	Présentation détaillée de la problématique et de son contexte	3
II.	Synthèse de notre analyse	4
II.1.	Compréhension du sujet.....	4
II.2.	Maquettage.....	4
III.	Présentation des pistes envisagées et leurs justifications	5
IV.	Justification de nos choix techniques	8
	Conclusions individuelles	11
	Glossaire	13
	Annexes.....	16

I. Présentation détaillée de la problématique et de son contexte

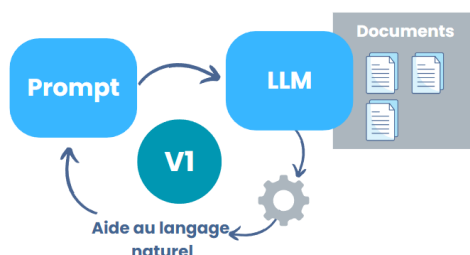
Dans le cadre d'une démarche de veille technologique, nous avons eu l'opportunité de collaborer avec le Groupe La Poste, plus précisément avec *M. Alain Janicot*, sur un projet innovant concernant le service informatique i-TEAM. Cette direction des services IT, qui compte aujourd'hui 1000 collaborateurs répartis sur 11 sites en France, joue un rôle crucial dans le maintien et l'évolution du système d'information du groupe.

La problématique centrale qui a motivé ce projet émane d'un constat opérationnel : la gestion des incidents en production au sein du système d'information d'i-TEAM fait face à plusieurs défis majeurs. Tout d'abord, l'analyse du contexte d'un incident technique complexe nécessite actuellement un temps considérable, ralentissant ainsi sa prise en charge. Cette complexité est amplifiée par la nécessité de rechercher les informations pertinentes dans une vaste documentation technique. De plus, la résolution des incidents dépend fortement de la disponibilité des experts, nécessitant la mise en place de groupes de résolution pour sélectionner les experts appropriés. L'ensemble de ces facteurs contribue à prolonger significativement le temps de résolution des incidents, impactant directement l'efficacité du support SI et, par extension, la performance globale des services informatiques.

Pour répondre à cette problématique, nous avons développé une première version du projet sous la forme d'une [application web](#) permettant de communiquer avec un assistant passif : **RAGAdmin**. Cet outil, basé sur la technologie [RAG](#) (*Retrieval-Augmented Generation*), combine intelligemment la recherche d'informations dans une base de données documentaire avec la génération de réponses contextuelles via un modèle de langage ([LLM](#)). Les capacités de cette version initiale de **RAGAdmin** sont multiples :

- La compréhension et la réponse aux questions posées en [langage naturel](#), permettant une interaction fluide et intuitive avec les utilisateurs
- L'assistance technique ciblée pour la résolution de problèmes spécifiques, notamment dans le cadre du Maintien en Condition Opérationnelle ([MCO](#)), en proposant des solutions adaptées
- L'accompagnement personnalisé des utilisateurs dans leur progression technique, en fournissant des ressources et des explications adaptées à leur niveau de compétence

Une particularité importante du projet réside dans son architecture locale et autonome, alimentée par la documentation technique interne de l'entreprise. Cette approche garantit non seulement la sécurité des données sensibles mais assure également des réponses pertinentes et adaptées au contexte spécifique de La Poste.



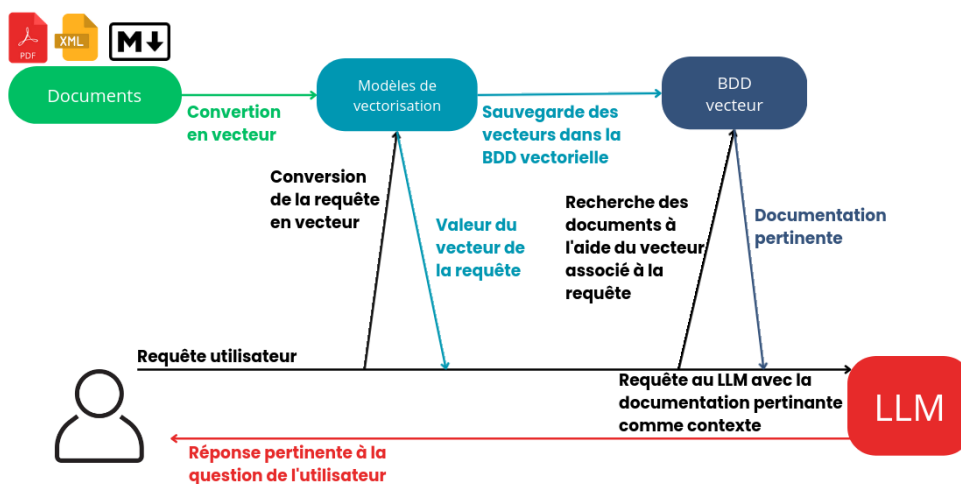
Bien que notre travail porte sur la **V1**, nous avons conçu le projet avec une vision évolutive. Cette première version, axée sur un mode passif, constitue une base solide pour des évolutions futures, avec des perspectives comme : La surveillance proactive des systèmes pour détecter les anomalies et l'exécution de scripts automatisés pour résoudre des incidents de manière autonome.

II. Synthèse de notre analyse

II.1. Compréhension du sujet

Afin de concevoir une [application web](#) avec un assistant intelligent performant, nous avons débuté par une analyse approfondie des mécanismes de fonctionnement d'un modèle de langage de grande taille ([LLM](#)). Nous avons étudié en détail comment intégrer une approche [RAG](#) (*Retrieval-Augmented Generation*) à ce type de modèle et comment implémenter cette solution en local, en tirant parti des ressources disponibles.

Nous avons élaboré un schéma décrivant le processus global que nous avons compris et mis en œuvre. Ce schéma (illustré ci-dessous) détaille les différentes étapes clés de recherche et de récupération de documents pertinents à l'aide de [vecteurs](#) (processus derrière [RAGAdmin](#)) :



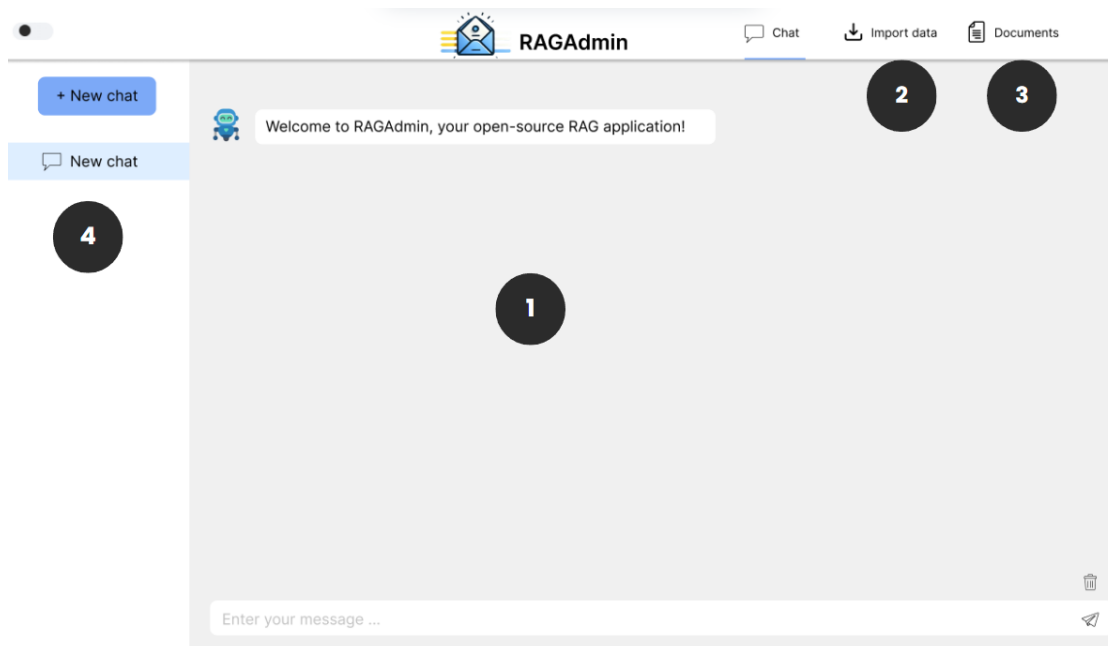
L'objectif de cette implémentation est non seulement de fournir des réponses pertinentes, mais également de mettre en place une structure évolutive qui pourra être adaptée et améliorée dans des versions futures.

II.2. Maquettage

Après avoir bien assimilé les objectifs du projet, nous avons procédé à l'élaboration d'un [Parcours utilisateur](#) et d'un [Story Mapping](#). Ces deux outils ont joué un rôle essentiel dans nos choix fonctionnels :

- **Parcours Utilisateur** : Cet outil nous a permis de visualiser l'expérience des futurs utilisateurs à travers différentes étapes d'interaction avec l'application. En nous mettant à leur place, nous avons pu identifier leurs attentes, leurs besoins, et les fonctionnalités prioritaires à intégrer.
- **Story Mapping** : En décomposant les fonctionnalités en blocs d'interaction, cette approche nous a aidés à structurer les fonctionnalités en fonction des objectifs du projet et des besoins des utilisateurs. Cela nous a également permis de prioriser les tâches et de concevoir une progression cohérente dans le développement de l'application.

Grâce à ces analyses, nous avons pu adopter une approche centrée sur l'utilisateur pour concevoir une application intuitive et adaptée aux besoins réels du support SI.



La maquette de l'application **RAGAdmin** illustre une interface simple, fonctionnelle et alignée avec les objectifs du projet. En s'inspirant des interfaces des [LLM](#) existants (comme *ChatGPT* et *Mistral*), elle répond aux besoins spécifiques des utilisateurs. Voici les points essentiels qui montrent la cohérence de cette maquette avec notre projet :

- **1) Zone centrale de discussion** : Cette zone est dédiée aux interactions en [langage naturel](#) entre l'utilisateur et l'assistant. Elle est au cœur de l'application, ce qui reflète l'objectif principal : fournir des réponses rapides et pertinentes grâce à l'intégration du [LLM](#).
- **2) Bouton « Import data »** : Ce bouton permet aux utilisateurs d'être redirigé sur une page permettant d'importer facilement des documents (PDF, XML, Markdown) dans le système, (voir [annexe](#) pour la maquette de cette page). Cela répond au besoin d'enrichir la base documentaire pour garantir des réponses toujours adaptées et basées sur des informations à jour.
- **3) Bouton « Documents »** : Cette section offre un accès direct aux documents disponibles, permettant aux utilisateurs de consulter manuellement des ressources spécifiques si nécessaire. Elle complète l'approche [RAG](#) en proposant une navigation supplémentaire dans les données. (Voir [annexe](#) pour la maquette de cette page).
- **4) Option "New chat"** : Ce bouton permet de lancer une nouvelle conversation. Il garantit une expérience fluide en séparant chaque interaction, ce qui est particulièrement utile pour gérer plusieurs sujets ou demandes différentes.

Cette maquette reflète parfaitement les objectifs du projet **RAGAdmin** grâce à sa simplicité et son efficacité : les éléments numérotés (1 à 4) sont disposés de manière intuitive, garantissant une prise en main rapide pour les administrateurs SI. Chaque fonctionnalité répond à un besoin clairement identifié lors de l'analyse, comme la recherche documentaire ou la génération de réponses contextuelles. De plus, sa structure évolutive permet d'intégrer facilement des fonctionnalités futures, telles que des outils d'analyse ou d'automatisation. Elle constitue ainsi une solution intuitive et performante, parfaitement adaptée aux attentes des équipes SI.

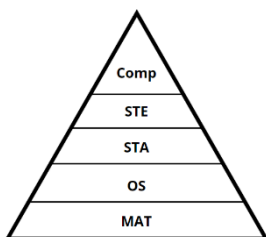
III. Présentation des pistes envisagées et leurs justifications

Pour répondre aux besoins de notre projet, nous avons étudié diverses technologies réparties en plusieurs catégories : [Frontend](#), [Backend](#), [CI/CD](#), [Bases de données](#), [Infrastructure](#), [Frameworks](#) et [Modèles LLM](#). Nos choix ont été guidés par des critères clés, notamment la gratuité, l'open-source, et la possibilité de fonctionner en local, garantissant ainsi l'indépendance vis-à-vis d'une connexion Internet et une meilleure sécurisation des données. Le tableau ci-dessous synthétise les technologies envisagées :

Catégories	Technologie	Description
Front End	Flask ~	Bien que fonctionnel, pour notre projet, il nécessiterait l'installation et la configuration de multiples extensions et une gestion manuelle de la structure du projet.
	React +	Framework performant et populaire pour des interfaces dynamiques.
	Streamlit -	Conçu pour des applications data-driven simples, limité en possibilités.
	OpenWebUI -	Application de Chat LLM fournie mais limitée pour les solutions RAG .
	Prochat +	Interface spécialisée pour des chats LLM .
	Tailwind (CSS) +	Framework utilitaire pour des designs modernes et réactifs.
	SASS +	Préprocesseur CSS structuré et réutilisable, améliore la gestion du style.
Back End	FastAPI ~	Framework rapide pour API simples mais configuration initiale complexe.
	Node.js +	Plateforme légère et flexible pour des applications performantes.
	Django +	Framework robuste pour projets complexes.
	Spring Boot -	Solution puissante mais lourde pour des infrastructures complexes en plus de l'utilisation de Java qui reste plus verbeux et moins rapide à prototyper que Python.
CI/CD	Jenkins + Gitea +	Pipelines personnalisables et auto-hébergement.
	GitHub Actions -	CI/CD cloud intégré, mais payant et compliqué à déployer sur Proxmox.
	GitLab CI/CD -	Avancé, mais gourmand en ressources et payant.
Base de données relationnelle	SQLite +	Léger en ce qui concerne la configuration, l'administration et le stockage. Facile à utiliser.
	Weaviate ~	Base de données vectorielle open-source qui simplifie le stockage des objets de données et des embeddings vectoriels des modèles ML .

Base de données vectorielles	FAISS ~	Outil puissant pour accélérer les recherches de similarités vectorielles denses, offrant une gamme de fonctionnalités et d'optimisations pour des opérations de recherche efficaces et efficientes. Solution proposée par Facebook.
	Chroma DB +	Capacité à traiter sans effort des documents textuels, à transformer du texte en enchâssements et à effectuer des recherches de similarité. Open source.
Infrastructure	Proxmox +	Gestion intuitive, faible consommation. Technologie étudié lors de la 2 ^{ème} année de BUT informatique.
	RedHat -	Solution premium mais coûteuse.
	Kubernetes -	Scalable mais nécessite des compétences trop avancées.
	Docker -	Léger mais insuffisant pour gérer une Infrastructure complète. Fortement déconseillé par notre intervenant.
LLM Framework	Hugging Face ~	Large bibliothèque mais gourmand en ressources.
	Ollama +	Optimisé pour des LLM spécifiques et local, simple à utiliser.
	LangChain +	Chaînes complexes et modulaires.
Modèle LLM	LLama 3.1 & 3.2 +	Open-source, performant et efficace.
	Mistral +	Léger et rapide pour des besoins spécifiques.
	GPT-4 -	Très puissant mais payant, et pas utilisable localement.

Pyramide des couches d'infrastructure



Dans le cadre de nos analyses, nous nous sommes appuyés sur la pyramide des couches d'[Infrastructure](#), un modèle de référence essentiel qui a structuré notre compréhension et l'organisation des différents aspects techniques du projet. Présentée par M. Alain Janicot, cette pyramide décrit les [infrastructures](#) en cinq niveaux distincts, chacun jouant un rôle clé dans l'écosystème global de **RAGAdmin** :

1. **MAT** (Matériel) : Infrastructure physique (serveurs, stockage, réseaux) constituant la base.
2. **OS** (Système d'exploitation) : Cette couche sert d'interface entre le matériel et l'hyperviseur.
3. **STA** (Stationnement/virtualisation) : Cette couche caractérise l'hyperviseur qui gère les machines virtuelles, nous permettant de faire de la mise en réseau virtuelle et gérer les ressources des [VMs](#) (le cœur de l'infrastructure virtualisée).
4. **STE** (Services techniques) : Couche dédiée la gestion des données, réseaux et environnements. Elle est cruciale dans notre infrastructure pour garantir la protection des machines et des données en cas de panne, de corruption ou de défaillance matérielle.

5. **COMP** (Composants applicatifs) : Au sommet de la pyramide des couches d'infrastructure se trouvent les machines virtuelles ([VM](#)) ou physique, qui exécutent les services visibles et directement utilisés par les utilisateurs finaux.

La hiérarchisation du projet en couches a permis d'organiser efficacement l'intégration des différentes technologies : le [Frontend](#) , [Backend](#) , [bases de données](#) et [Frameworks LLM](#) dans la couche **COMP**, les outils de conteneurisation dans **STA** et **STE** assurant ainsi la cohérence technique de l'ensemble.

IV. Justification de nos choix techniques

Avec toutes ces pistes envisagées, pour concevoir notre solution nous avons structuré nos choix techniques en suivant la hiérarchie des couches de la pyramide d'infrastructure. Voici les justifications détaillées, en partant de la base jusqu'au sommet.

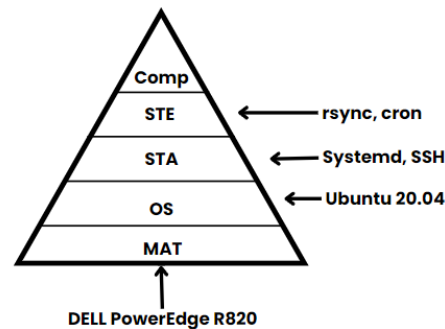
Matériel (MAT) : Nous n'avons pas eu à choisir cette couche, car le matériel nous a été fourni : un serveur **DELL PowerEdge R820** avec 196 Go de [RAM](#) et sans [GPU](#).

Système d'exploitation (OS) : *Proxmox (version 7.4)* étant un [hyperviseur de type 1](#), fait qu'il fonctionne directement sur le matériel, sans qu'un système d'exploitation complet soit installé en premier. Etant aussi basé sur Debian, veut dire que son noyau (kernel) et son environnement d'exécution sont construits à partir de Debian Linux, avec des adaptations spécifiques pour inclure les fonctionnalités de [virtualisation](#) (comme *QEMU/KVM* et *LXC*). Dans un premier temps, notre serveur utilisait *Proxmox* comme système d'exploitation (basé sur *Debian*). Cependant, lors de la phase d'expérimentation, nous avons rencontré des problèmes de performances majeurs avec le [LLM](#), avec des temps de réponse atteignant 50 minutes. Bien que nous ayons initialement soupçonné que l'absence de [GPU](#) sur le serveur puisse être à l'origine de ces lenteurs, M. Alain Janicot nous a confirmé qu'il était possible de réaliser ce projet sans carte graphique. Nous avons alors investigué d'autres pistes et avons décidé de faire un [dualboot](#) avec *Ubuntu*, qui offre un meilleur support des [bibliothèques](#) nécessaires à l'optimisation des modèles [LLM](#). Actuellement notre [OS](#) est *Ubuntu 20.04*.

Infrastructure de virtualisation (STA) : *Proxmox 7.4* permet de gérer efficacement les machines virtuelles et les sauvegardes grâce à son interface intuitive et son faible impact supposé sur les ressources. Il offre également une modularité et une isolation des environnements critiques ([WebAPP](#), [LLM](#), [CI/CD](#)). Toutefois, face aux problèmes de performance mentionnés précédemment, nous avons pris la décision de désactiver la [virtualisation](#) et de travailler directement sur le serveur physique. Ce choix a permis une allocation directe des ressources matérielles et a contribué à réduire significativement les temps de réponse. Pour gérer et orchestrer les différents services dans cet environnement sans [virtualisation](#), nous avons adopté [systemd](#), qui nous permet de superviser et contrôler efficacement les processus critiques (démarrage automatique, redémarrage en cas d'erreur, etc.). L'accès et la gestion à distance du serveur ont été facilités par l'utilisation de [SSH](#), garantissant une administration sécurisée et une grande flexibilité dans nos interventions techniques.

STE (Services Techniques): Pour les services techniques, nous avons initialement utilisé les fonctionnalités de sauvegarde intégrées de *Proxmox*. Avec l'abandon de la [virtualisation](#), nous avons ajusté notre approche de sauvegarde pour nous adapter à l'environnement sans hyperviseur, tout en assurant la continuité des données critiques. Avec l'abandon de la [virtualisation](#), nous avons ajusté notre approche de sauvegarde pour nous adapter à l'environnement sans hyperviseur, tout en assurant la continuité des données critiques. Nous avons mis en place une solution basée sur [rsync](#) et [cron](#) pour automatiser les sauvegardes. (Voir annexe exemple concret).

Ainsi, nous obtenons la pyramide suivante :



Services finaux (COMP) :

Frontend : Initialement, nous avons envisagé d'utiliser *Streamlit* pour le projet. Cependant, en expérimentant avec cet outil, nous nous sommes rapidement rendu compte qu'il présentait des limitations importantes, notamment pour implémenter la maquette et les fonctionnalités prévues. Sa grande simplicité d'utilisation est à la fois son point fort mais également son point faible, car la personnalisation avancée de l'interface ainsi que les éléments complexes ne sont pas disponibles. C'est pourquoi nous avons opté pour *React*. Ce choix nous permet de concrétiser toutes les fonctionnalités établies au préalable et offre une grande flexibilité pour l'intégration avec des solutions [Backend](#) comme *Django* en plus de disposer d'une large documentation. Pour ceux qui est du style pur de l'application, nous avons opté pour *Tailwind* qui permet très facilement d'obtenir un joli rendu tout en possédant une grande possibilité de personnalisation, ainsi que *SCSS* qui est son parfait complément pour les petites choses ardues à faire à l'aide de *Tailwind* tel que le mode sombre.

Backend: Le choix de *Django* s'est imposé car il s'agit d'un [framework](#) *Python*, un langage avec lequel nous sommes familiers grâce à notre formation universitaire. De plus, nous prévoyons d'utiliser des [bibliothèques](#) *Python* pour réaliser les opérations vectorielles indispensables dans le cadre d'un [LLM](#). Enfin, en plus de s'intégrer parfaitement avec *React*, *Django* bénéficie d'une documentation riche et complète, facilitant son utilisation.

Bases de données : Nous avons choisi *ChromaDB* comme [base de données vectorielle](#) car, en l'expérimentant, nous l'avons trouvé simple d'utilisation et efficace pour les opérations vectorielles. De plus, étant une [bibliothèque](#) *Python* (un langage que nous maîtrisons bien), *ChromaDB* permet l'intégration d'autres [bibliothèques](#), telles que *SentenceTransformers*, pour utiliser des modèles encore plus performants et spécifiques, notamment pour la vectorisation de texte. Par ailleurs, pour répondre à notre besoin de persistance des vecteurs, nous avons opté pour *SQLite* en raison de sa simplicité de mise en œuvre.

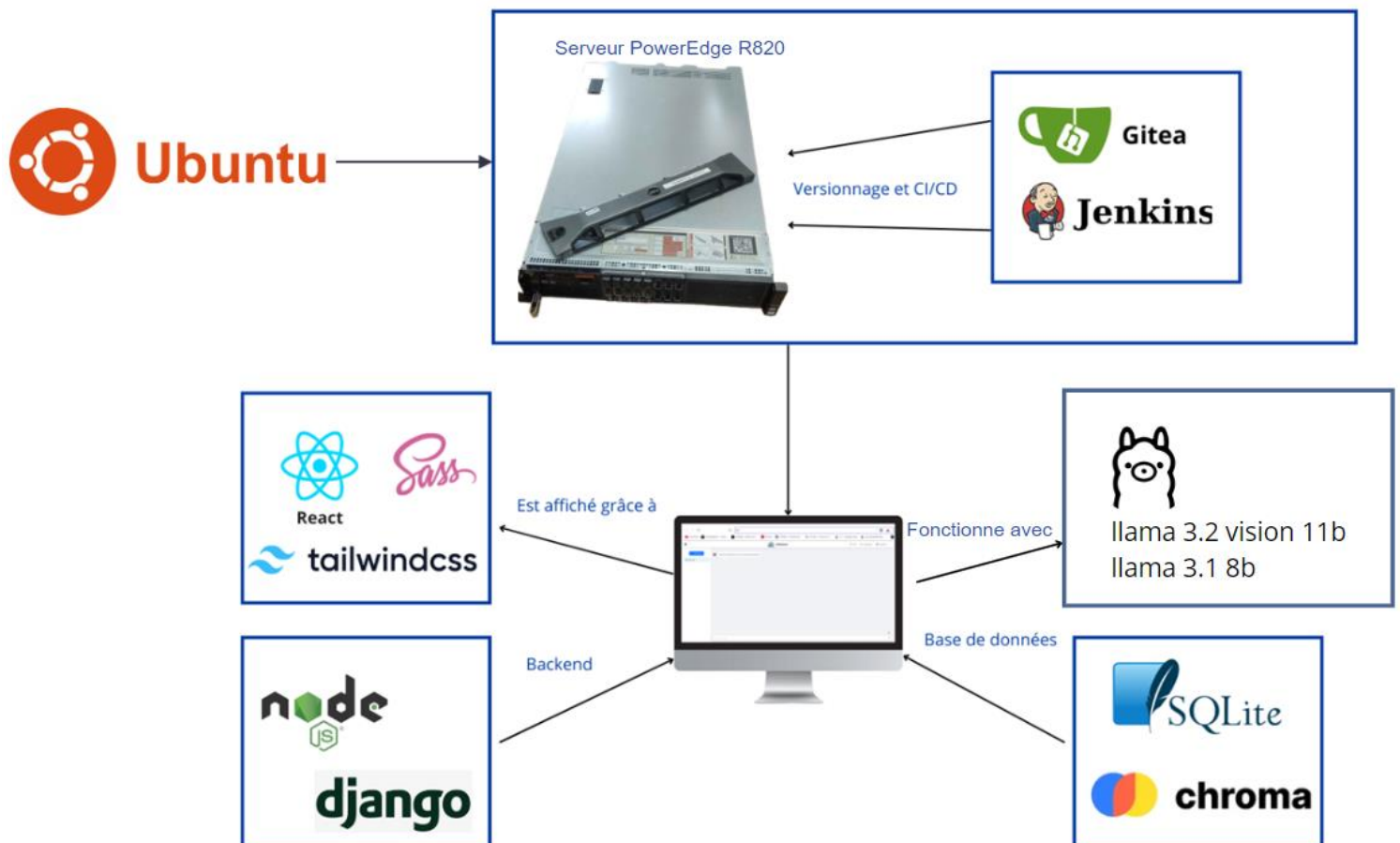
LLM : Nous avons développé une fonctionnalité permettant à l'utilisateur de choisir le modèle de langage qu'il souhaite utiliser parmi ceux qu'il a installés. Cela garantit une flexibilité maximale et permet d'intégrer un large éventail d'utilisateurs. En ce qui concerne le serveur fourni par La

Poste, après avoir expérimenté un grand nombre de [LLM](#), les modèles préinstallés et optimaux pour **RAGAdmin** sont *LLama 3.2 Vision version 11B* et *LLama 3.1 version 8B*.

Gestionnaire de version : Nous avons choisi d'utiliser *Gitea* car il s'agit d'une solution open-source, facile à déployer, ce qui est un atout essentiel étant donné que nous devons l'installer sur un serveur. Son caractère open-source et son interface intuitive en font un outil particulièrement adapté à nos besoins. De plus, *Gitea* s'intègre parfaitement avec notre outil [CI/CD](#), *Jenkins*, renforçant ainsi son rôle central dans notre workflow de développement.

CI/CD : Nous avons choisi *Jenkins* car il est complètement open-source, dispose d'une large documentation, s'intègre parfaitement avec *Gitea*, et offre une compatibilité idéale pour notre workflow. De plus, sa simplicité de mise en place sur un serveur, combinée à son interface intuitive, en fait une solution adaptée à nos besoins pour l'automatisation des [pipelines CI/CD](#).

Pour conclure cette partie et résumer les choix technologiques effectués, le schéma ci-dessous présente une vue d'ensemble des technologies utilisées et leur interaction au sein de notre infrastructure. Il illustre clairement les différents composants et leur rôle dans la solution mise en place.



Des images de notre solutions fonctionnelles sont disponibles en [annexes](#).

Conclusions individuelles

Jimmy : Cette SAE m'a permis de développer de nombreuses compétences techniques. J'ai appris à installer et configurer des systèmes comme *Proxmox*, gérer les ressources serveur, configurer des réseaux, créer un [RAG](#) utilisant une IA, et mettre en place des [RAID](#). La configuration des [RAID](#) m'a fait comprendre l'importance de la redondance des données, le choix du type de [RAID](#) adapté aux besoins, et leurs implications sur le stockage et la performance. Bien que le réseau ne soit pas un domaine que j'appréciais initialement, cette expérience m'a aidé à mieux comprendre son rôle et à résoudre des problèmes de connectivité, renforçant ainsi mes compétences pratiques. Ce projet a amélioré ma maîtrise de la gestion des systèmes, des réseaux et du stockage, tout en me donnant une vision claire de l'importance d'une configuration et d'une gestion rigoureuse des ressources pour garantir la stabilité et la performance d'un environnement virtuel complexe.

Léo : Au cours de ce projet, nous avons configuré un serveur en mettant en place la virtualisation avec *Proxmox*, l'installation de l'[OS \(Ubuntu\)](#), l'environnement, et la configuration réseau. Cette expérience m'a permis de renforcer mes compétences en configuration et déploiement d'infrastructure, notamment sur les réseaux complexes locaux. Passionné par l'IA, j'ai particulièrement apprécié cette SAE, qui m'a permis de travailler avec différents modèles, d'optimiser leur utilisation (prompt engineering, comparaison de modèles) et de découvrir des [bases de données vectorielles](#) ainsi que la [vectorisation](#) de documents, des éléments clés en [Machine Learning](#) et Intelligence Artificielle. Le développement d'une solution moderne avec des technologies récentes, pertinentes pour le monde professionnel, a été particulièrement enrichissant, sachant que ces apprentissages seront utiles pour mon avenir. Enfin, la dimension réelle de ce projet, réalisé pour une grande entreprise comme La Poste, a renforcé son intérêt en offrant une expérience concrète et professionnelle.

Mohamed : Ce projet a été une expérience particulièrement formatrice. Sur le plan technique, j'ai développé une expertise significative dans la configuration d'infrastructure serveur, notamment à travers la gestion du [BIOS](#), du [RAID](#) et de l'[IDRAC](#). J'ai également approfondi mes connaissances en Intelligence Artificielle, particulièrement dans les domaines des [LLM](#) et du [RAG](#), tout en maîtrisant de nouvelles technologies comme *ChromaDB* et *OLLAMA*. Ces apprentissages sont venus compléter mes compétences existantes en *Django*, *React*, *SCSS* et *Tailwind* pour le développement d'[application web](#). Sur le plan organisationnel, l'application de la méthodologie [SCRUM](#) avec une rotation du rôle de chef de projet toutes les deux semaines s'est révélée être une approche efficace, me permettant de développer mes compétences en gestion d'équipe et en coordination de projet. La principale erreur technique a été le manque d'évaluation approfondie de *Proxmox* avant son implémentation, ce qui a temporairement impacté les performances de notre [LLM](#). Du côté organisationnel, nous avons constaté qu'une analyse plus poussée des outils et technologies en phase initiale aurait été bénéfique. Ces expériences ont contribué à renforcer mes compétences en gestion de projet informatique et m'ont enseigné l'importance cruciale de l'adaptabilité dans notre domaine.

Yassine : Cette SAE, en partenariat avec La Poste, a été une opportunité très enrichissante et formatrice dans le cadre de mon BUT. Un point clé de cette expérience a été la découverte et l'expérimentation des bases de données vectorielles via des [bibliothèques Python](#) (*ChromaDB*, *FAISS*, *Weaviate*), essentielles au processus de **RAGAdmin**. Configurer une infrastructure virtualisée et déployer RAGAdmin ont renforcé mes compétences en virtualisation et chaîne de

production (*Proxmox*, *Jenkins*, configuration réseau). Les échanges avec des professionnels de La Poste m'ont permis d'approfondir ma maîtrise des outils utilisés. D'un point de vue organisationnel, travailler avec M. Janicot m'a permis d'expérimenter les méthodes *AGILE*, avec des réunions bihebdomadaires et des points quotidiens en groupe. Après une phase d'analyse, nous avons réalisé des [sprints](#) pour implémenter les fonctionnalités. Le changement régulier de chef de projet, combiné à l'utilisation de *Jira* et *Confluence*, a renforcé mes compétences en gestion de projet, communication et travail d'équipe. Pour les projets futurs, je devrai réfléchir méthodiquement en gardant à l'esprit le principe de la pyramide d'infrastructure, en veillant à construire une base solide afin de garantir la réussite du produit final et visible. Finalement, avoir produit une solution fonctionnelle de **RAGAdmin**, adaptable à divers secteurs, représente pour moi une véritable réussite.

Cyril : Lors de ce projet, j'ai découvert plusieurs facettes du métier de *DevOps*, notamment des aspects que je n'avais pas abordés durant mon cursus scolaire. J'ai eu l'opportunité d'explorer de nouvelles technologies, telles qu'*Ollama*, et de les intégrer dans un programme. J'ai également approfondi mes compétences en [bases de données vectorielles](#) avec *ChromaDB*. La configuration d'un serveur neuf pour accueillir un système d'exploitation spécifique a enrichi mes connaissances, tout comme l'utilisation de *Proxmox* pour la gestion des ressources et des [machines virtuelles](#). Par ailleurs, j'ai appris à manipuler des [frameworks](#) tels que *LangChain* et *Jupyter*, ainsi que des outils de gestion de projet comme *Jira* et *Confluence*, que je n'avais jamais utilisés auparavant. Ce projet m'a aussi permis d'identifier des erreurs à éviter, telles que les problèmes liés aux versions *OS/serveur* ou à l'évaluation des performances d'un système d'exploitation. Globalement, cette expérience a été très enrichissante, tant en termes de compétences techniques que d'organisation, tout en me rapprochant des exigences du cadre professionnel grâce à un sujet passionnant lié à l'IA.

Glossaire

API (Application Programming Interface) est un ensemble de protocoles, de routines et d'outils de programmation permettant la communication entre différents logiciels ou composants logiciels.

Application web : Logiciel accessible et utilisable directement depuis un navigateur internet.

Backend : La partie d'une application qui s'exécute sur le serveur, gérant la logique métier, les opérations de base de données et la performance de l'application.

Bases de données : Un système organisé de stockage de données qui permet la saisie, la consultation, la mise à jour et l'analyse des données.

Bases de données vectorielles : Base de données spécialisée qui stocke les informations sous forme de vecteurs mathématiques pour faciliter les recherches par similarité.

Bibliothèque : Collection de code réutilisable fournissant des fonctionnalités prêtes à l'emploi pour faciliter le développement d'applications.

BIOS (Basic Input/Output System): Programme intégré à l'ordinateur qui gère le démarrage et l'initialisation du matériel.

CI/CD (Continuous Integration / Continuous Deployment): Méthodes et outils permettant d'automatiser les étapes de développement et de déploiement d'applications.

Cron : Outil sur Linux permettant de programmer l'exécution automatique de tâches à des moments précis.

Data-driven : Approche basée sur l'analyse des données pour prendre des décisions plutôt que sur l'intuition.

Dualboot : Configuration permettant d'avoir deux systèmes d'exploitation sur un même ordinateur.

Embeddings (ou vecteurs) : Représentations numériques d'objets (comme des mots, des phrases, des images ou des documents) sous forme de vecteurs dans un espace multidimensionnel. Ces vecteurs capturent les relations et les similarités entre les objets en fonction de leur contexte ou de leur signification.

Frameworks : Ensemble d'outils et de règles fournissant une base pour développer des applications.

Frontend : La partie d'une application web que l'utilisateur voit et avec laquelle il interagit directement.

Gestionnaire de version : Outil qui permet de suivre et de gérer les modifications apportées à des fichiers, principalement dans le développement de logiciels. Il garde un historique des

changements, permettant aux développeurs de collaborer efficacement, de revenir à des versions antérieures et de résoudre les conflits de modifications.

GPU (Graphics Processing Unit): Processeur spécialisé pour le traitement rapide des calculs graphiques et d'intelligence artificielle..

Hyperviseur de type 1 : Logiciel qui s'exécute directement sur le matériel physique d'un serveur ou d'un ordinateur, sans système d'exploitation hôte. Il permet de créer et de gérer des machines virtuelles (VM) en partageant les ressources matérielles.

IDRAC : Interface de gestion à distance des serveurs Dell permettant leur administration sans présence physique.

Infrastructure : Désigne l'ensemble des ressources matérielles (serveurs, réseaux, stockage) et logicielles (systèmes d'exploitation, outils de gestion) nécessaires pour faire fonctionner des applications, des sites web ou des services. Elle peut être physique (sur site) ou virtuelle (cloud), et elle constitue la base sur laquelle reposent les systèmes et les applications.

Langage naturel : Langage utilisé naturellement par les humains pour communiquer, par opposition au langage informatique.

LLM (Large Language Model) : Un grand modèle linguistique (Large Language Model, LLM) est un type de programme d'intelligence artificielle (IA) capable, entre autres tâches, de reconnaître et de générer du texte. Les LLM sont entraînés sur de vastes ensembles de données, d'où l'emploi du terme « large » (grand) dans la dénomination anglaise.

ML (Machine Learning) : Technologie permettant aux ordinateurs d'apprendre à partir de données sans être explicitement programmés.

Machine Virtuelle (VM): Une machine virtuelle est un environnement virtualisé qui fonctionne sur une machine physique. Elle permet d'émuler un OS (Operation System) sans l'installer physiquement sur l'ordinateur.

MCO (Maintenance en Condition Opérationnelle) : Ensemble des actions permettant de maintenir un système informatique en bon état de fonctionnement.

Modèle LLM : Un système d'intelligence artificielle entraîné sur de grandes quantités de textes, capable de comprendre et générer du langage naturel pour diverses tâches comme la rédaction, la traduction ou la réponse aux questions.

OS (operating system): Logiciel essentiel qui permet de gérer le matériel d'un ordinateur (comme le processeur, la mémoire, le stockage) et de fournir des services aux applications. Il sert d'interface entre l'utilisateur et le matériel.

Pipelines : Suite d'étapes automatisées pour tester et déployer des applications.

RAG (Retrieval-Augmented Generation) : technique qui améliore les capacités d'un modèle de langage (**LLM**) en lui permettant d'accéder à une base de connaissances externe avant de générer une réponse. Le processus se déroule en trois étapes principales : d'abord, la requête de l'utilisateur est analysée pour rechercher les informations pertinentes dans la base de connaissances. Ensuite, ces informations sont combinées avec la requête pour créer un contexte enrichi. Enfin, le **LLM** utilise ce contexte pour générer une réponse plus précise et mieux documentée. Cette approche permet d'obtenir des réponses plus fiables et adaptées à un domaine

spécifique sans avoir à réentraîner le modèle, tout en réduisant les risques d'hallucinations du [LLM](#).

RAID (Redundant Array of Independent Disks): Technologie de virtualisation du stockage qui combine plusieurs disques durs physiques en une seule unité logique pour assurer la redondance des données et/ou améliorer les performances.

RAM (Random Access Memory): Mémoire temporaire de l'ordinateur qui stocke les données des programmes en cours d'utilisation..

Rsync : Outil de synchronisation qui copie efficacement des fichiers entre différents emplacements.

SCRUM : Méthodologie agile de gestion de projet qui organise le travail en cycles courts appelés "[sprints](#)". Elle met l'accent sur la collaboration, la flexibilité et la livraison incrémentale de fonctionnalités.

Sprint : Période de travail courte et définie (généralement 2 semaines) pendant laquelle une équipe doit réaliser des objectifs précis dans une méthode agile.


SSH (Secure SHELL) : Protocole sécurisé permettant de se connecter et contrôler un ordinateur à distance

Systemd : Système de gestion des services et du démarrage sur Linux.

Virtualisation : La virtualisation est une technologie que vous pouvez utiliser pour créer des représentations virtuelles de serveurs, de stockage, de réseaux et d'autres machines physiques. Le logiciel virtuel imite les fonctions du matériel physique pour exécuter plusieurs machines virtuelles sur une seule machine physique.

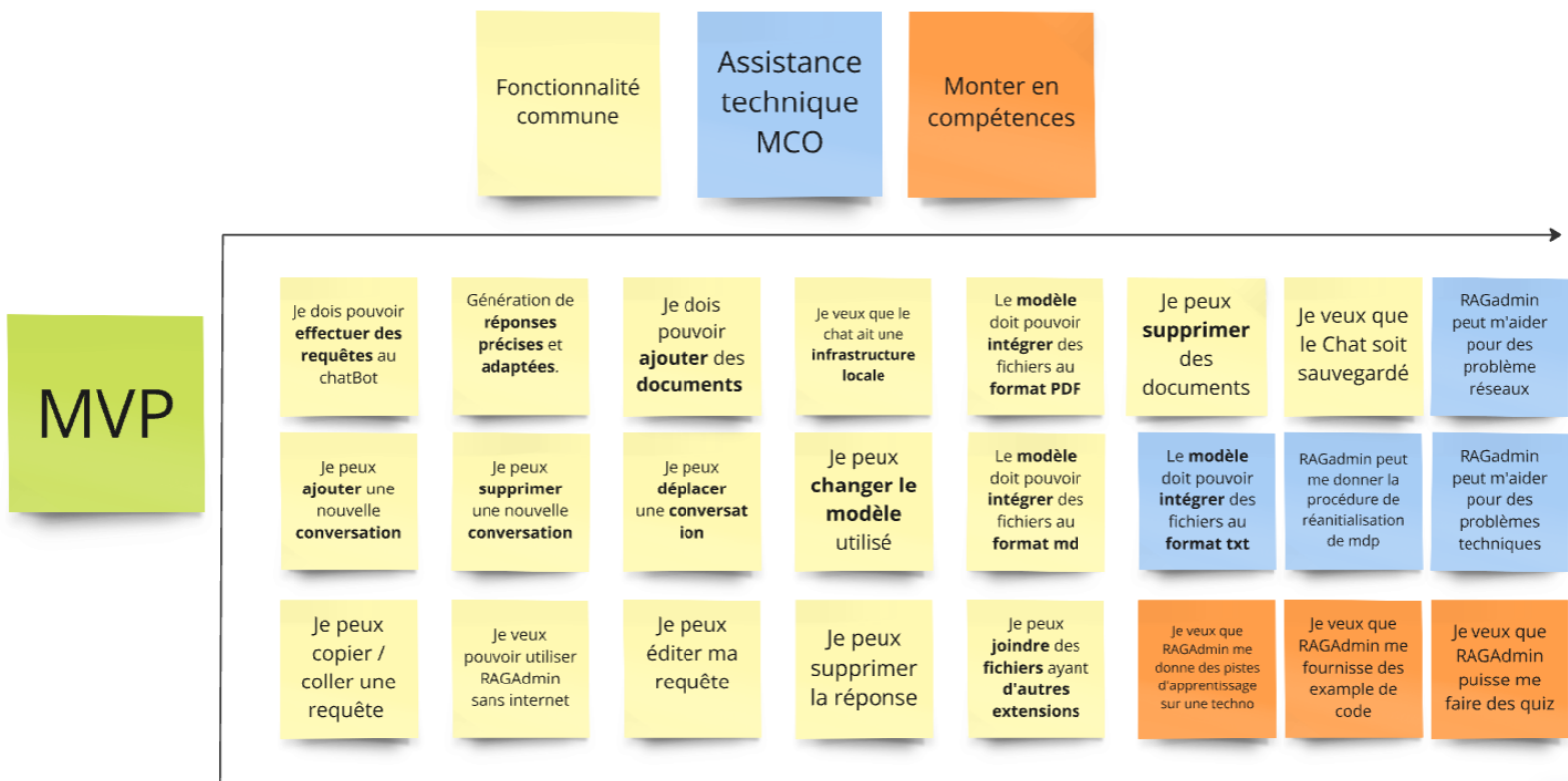
Annexes

1. Exemple d'une partie du Parcours utilisateurs

	Insertion de Documents		
	Choix des documents	Interaction fonctionnalité	Document ajouté
ACTIONS	<ul style="list-style-type: none"> -Se rend dans l'explorateur de fichier -Repère / créer un dossier 	<ul style="list-style-type: none"> -Clique sur Insertion Documents -Sélectionne les fichiers à insérer 	<ul style="list-style-type: none"> -Vérifie que le message indique un succès -Va discuter avec le rag
TOUCH POINTS	-Ordinateur	-ragadmin/insert-data	-ragadmin/insert-data
PAIN POINTS	<ul style="list-style-type: none"> -Il peut lui manquer des documents -Volume de documents difficile à filtrer -Incertitude sur l'impact de leur choix de document 	<ul style="list-style-type: none"> -L'insertion peut prendre un peu de temps (fichier volumineux) -Un format peut ne pas être pris en charge 	<ul style="list-style-type: none"> -Le nom du fichier est trop grand pour être affiché en entier -Une erreur est affichée
OPPORTUNITY	-Libre de choisir les documents qu'il veut	<ul style="list-style-type: none"> -Prise en charge du drag and drop -Format pris en charge explicitement écrits -Ouverture rapide de l'explorateur de fichier 	<ul style="list-style-type: none"> -Affichage d'un message clair indiquant le succès de l'upload -Possibilité d'enchaîner les insert à la suite -Possibilité d'importer plusieurs documents en 1 fois
EXPERIENCE	<div>Positive</div> <div>Negative</div> 	curieux, excité	satisfait, heureux

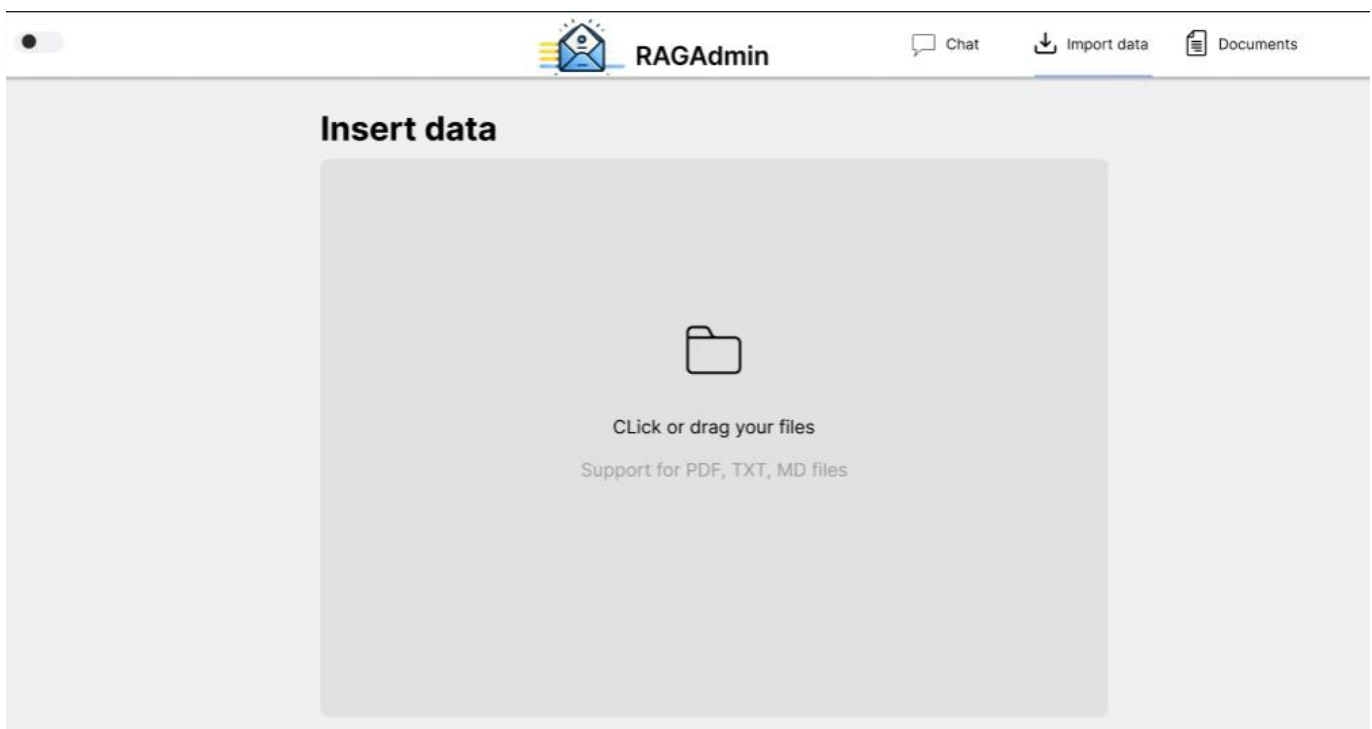
<https://www.canva.com/design/DAGXwSL8jQA/FoV3MG3QYSHdiqCjEpL6-g/edit>

2. Story Mapping (focus MVP)

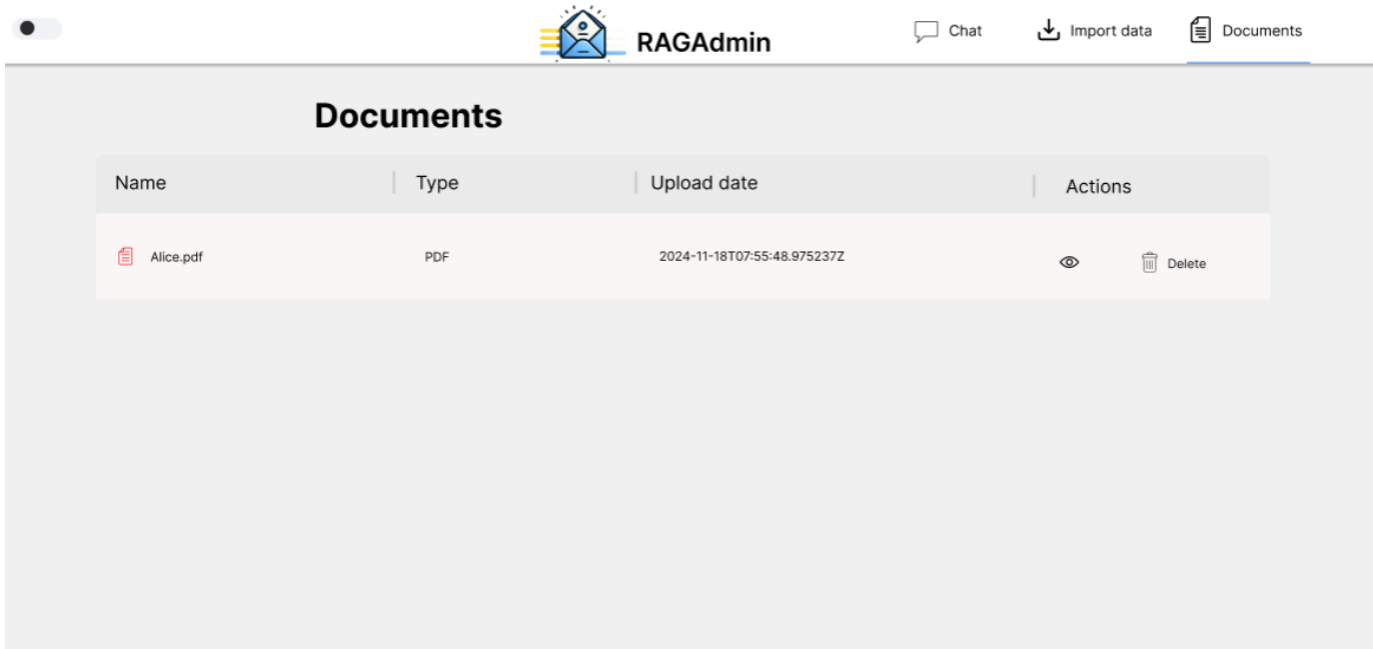


<https://miro.com/welcomeonboard/SDI4STNRNFZVbFUzRO9PMG5qUjhnQOtCUIUvbWRBLzFZUjE2RUlnZVBTUdIcGFrSTg5TWWh1aENHRFRkdEJmTXVqUmE3amNuSnVZanVYNTJGROFYUS9PRzFnM2EzMmJiVOFYdXpjVmMrRzY5Vy9NK3RpRnJnRTB1NOJpajVJRm4hZQ==?share link id=761361176147>

3. Maquette de la page « import data »

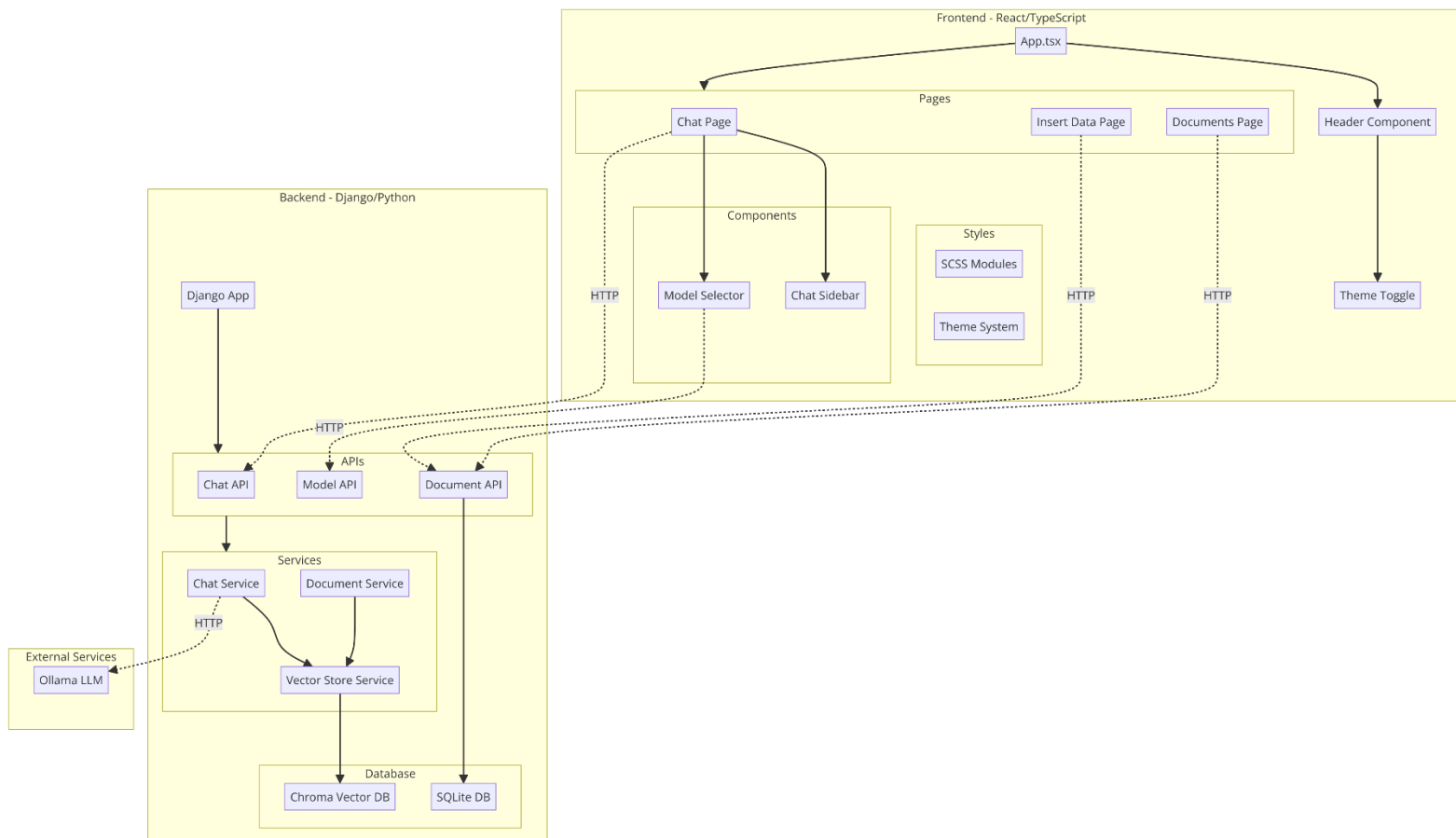


4. Maquette de la page « Documents »



<https://www.figma.com/design/gSeZb4E7p7ITCrwv4mr98F/RagAdmin?node-id=0-1&p=f&t=pQfn0Fu6eU66nF6j-0>

5. Cartographie de l'Architecture RAGAdmin



6. Notre solution RAGAdmin fonctionnelle

