Équipe #15

Engin de recherche et plateforme de maillage dans le domaine de l’IA en santé

Document d'architecture logicielle

Version 3.0

Historique des révisions

Date	Version	Description	Auteur
2023-09-28	2.0	Rédaction des sections 1 et 3	Antoine Déry
2023-09-30	2.1	Rédaction des sections 2	Antoine Déry
2023-10-02	2.2	Rédaction des sections 4 et 6	Augustin Lompo Antoine Déry
2023-10-04	2.3	Rédaction de la section 7	Adam Halim
2023-10-06	2.4	Rédaction des sections 5	Antoine Déry Hichem Lamraoui Augustin Lompo
2023-10-06	2.5	Révision pour prototype	Équipe #15
2023-11-24	3.0	Correction selon les rétroactions de la remise de mi-session	Équipe #15

Table des matières

Document d'architecture logicielle

1. Introduction

Ce document décrit l'architecture logicielle du répertoire des expertises de la Communauté de Pratique IA en Santé (CPIAS). Il fait notamment état des décisions prises quant à l'interaction entre les différentes composantes, autant à bas qu'à haut niveau. Le document est divisé en sections qui présentent les objectifs et contraintes architecturaux, les principaux cas d'utilisation ainsi que la logique de l'application. De plus, les processus, sous forme de diagrammes de séquences, le diagramme de déploiement et les caractéristiques relatives à la taille de à la performance du logiciel sont présentés.

2. Objectifs et contraintes architecturaux

L'architecture d'un logiciel est guidée par un ensemble d'objectifs et de contraintes qui déterminent notamment sa fiabilité, ses coûts et sa sécurité. Ces éléments clés influencent les décisions prises tout au long du processus de conception.

2.1. Fiabilité

L'objectif de fiabilité est essentiel pour assurer le bon fonctionnement du système. Il se traduit par des exigences de disponibilité élevées. Tel qu'énoncé dans le document de spécification des requis, l'objectif est d'offrir une disponibilité de 99,9 %. Pour atteindre cet objectif, le système intègre des mécanismes de sauvegarde et de récupération en cas de défaillance. Les stratégies de gestion des erreurs sont également mises en œuvre pour garantir une expérience utilisateur fluide. De plus, des tests de résistance et de performance sont effectués pour valider la fiabilité du système.

2.2. Coûts

Les contraintes liées aux coûts définissent les ressources financières disponibles pour le projet. Dans le cadre de ce projet, un budget de 500$ nous a été alloué , ce qui nous permettra d'accéder aux ressources adéquates pour le développement et le déploiement de notre solution.

2.3. Sécurité

La sécurité est une contrainte dans le projet. Nous devons mettre en place des mesures robustes pour protéger les données sensibles et garantir la confidentialité, l'intégrité et la disponibilité de l'information. La sécurité doit être intégrée à tous les niveaux de l'architecture, principalement en ce qui concerne l'hébergement du serveur. En effet, le serveur doit être hébergé sur les serveurs du fournisseur en infonuagique AWS.

2.4. Échéancier

Respecter l'échéancier du projet est essentiel pour son succès. Nous devons planifier l'architecture de manière à garantir que les étapes clés sont réalisées en temps voulu pour répondre aux besoins du projet. En effet, ce projet est relativement court puisqu'il doit être réalisé en moins de quatre mois. Cela a notamment des implications sur le choix du modèle d'intelligence artificielle à utiliser. En effet, la création et l'entraînement de notre propre modèle n'était pas une option envisageable en raison du faible nombre de données et de la contrainte de temps.

3. Vue des cas d’utilisation

Les cas d’utilisation illustrés ci-dessous mettent en évidence les différentes fonctionnalités auxquelles ont accès les utilisateurs du répertoire des expertises de la CPIAS.

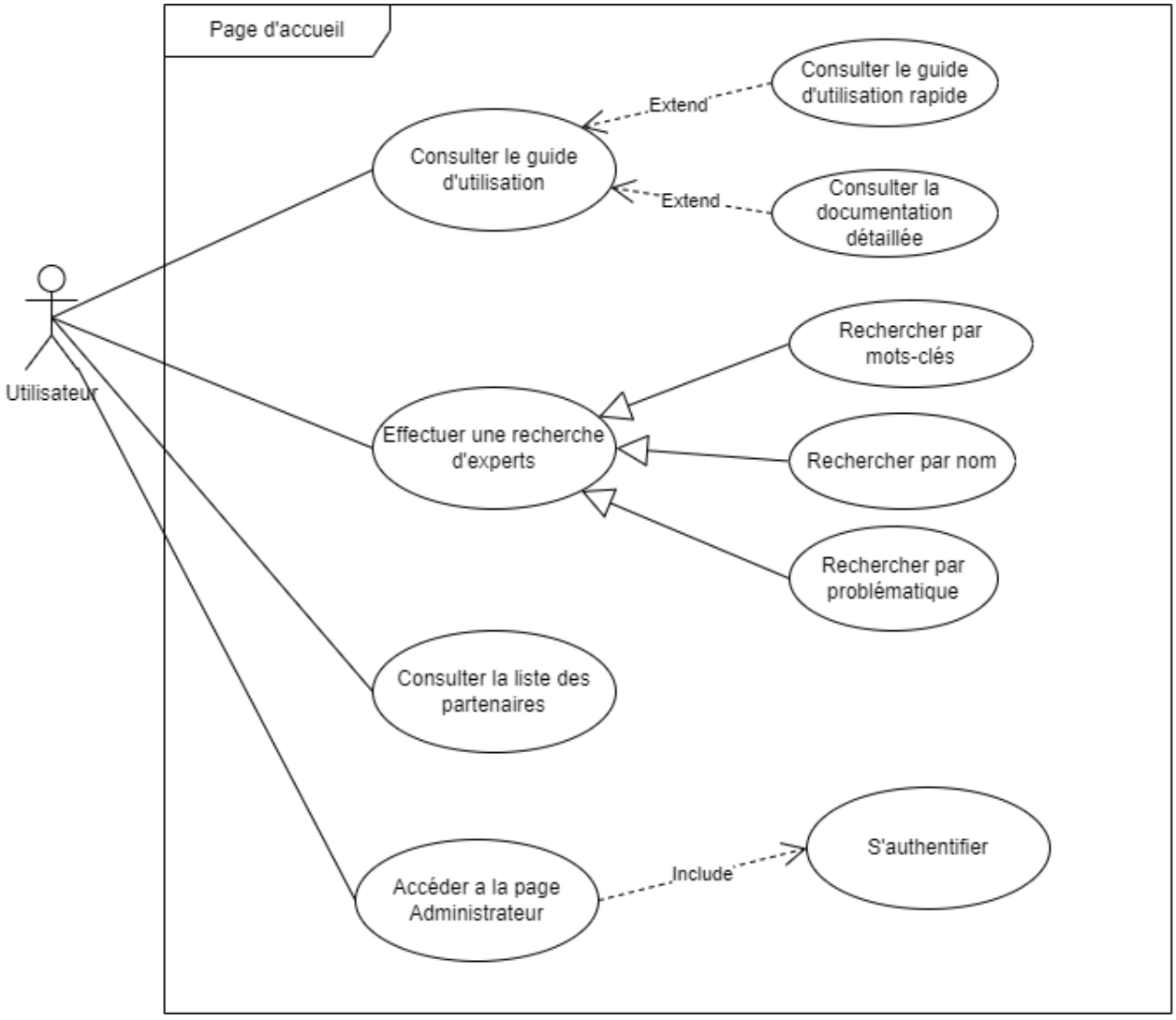

Figure 3.1 : Cas d’utilisation de la page d’accueil

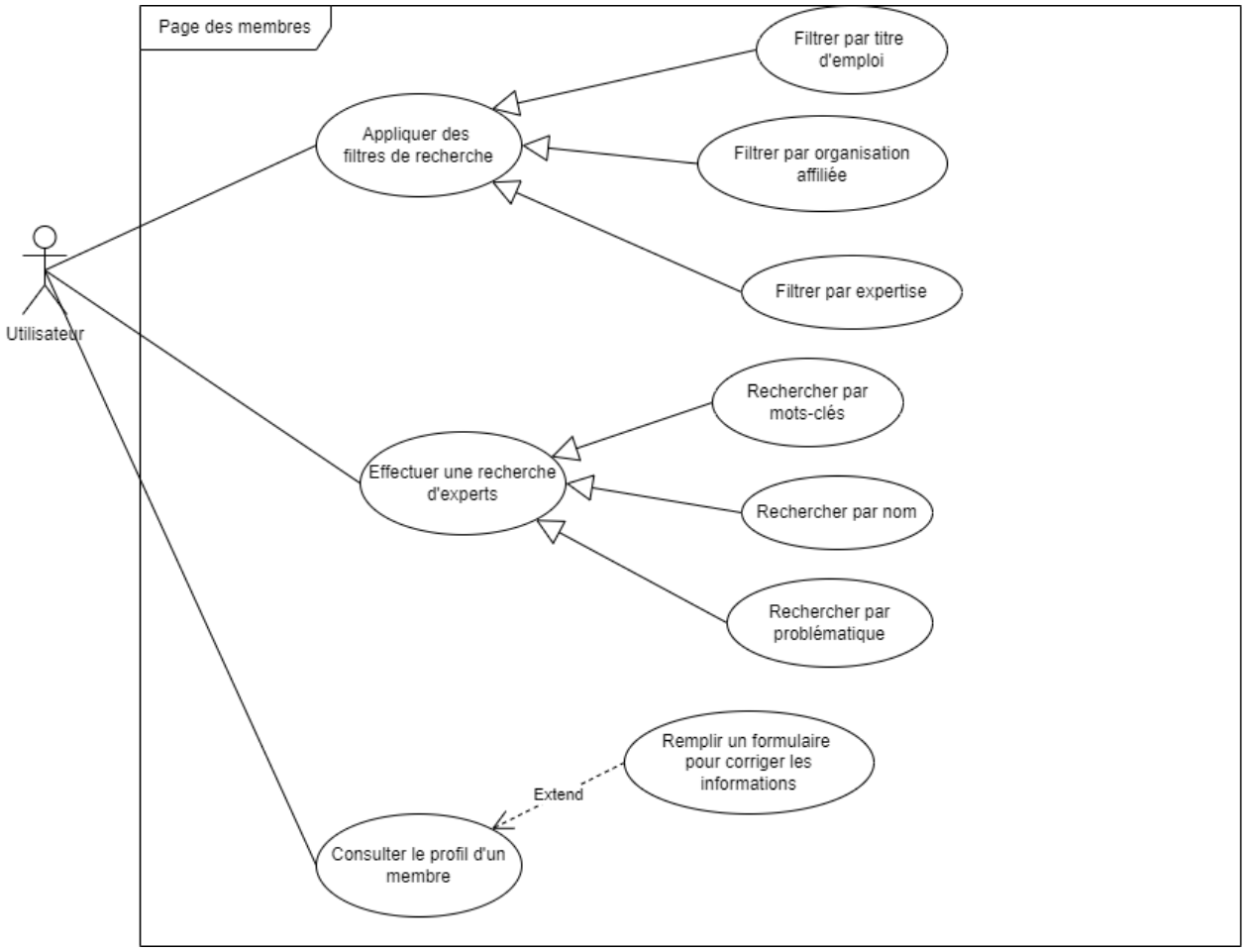

Figure 3.2 : Cas d'utilisation de la page des membres

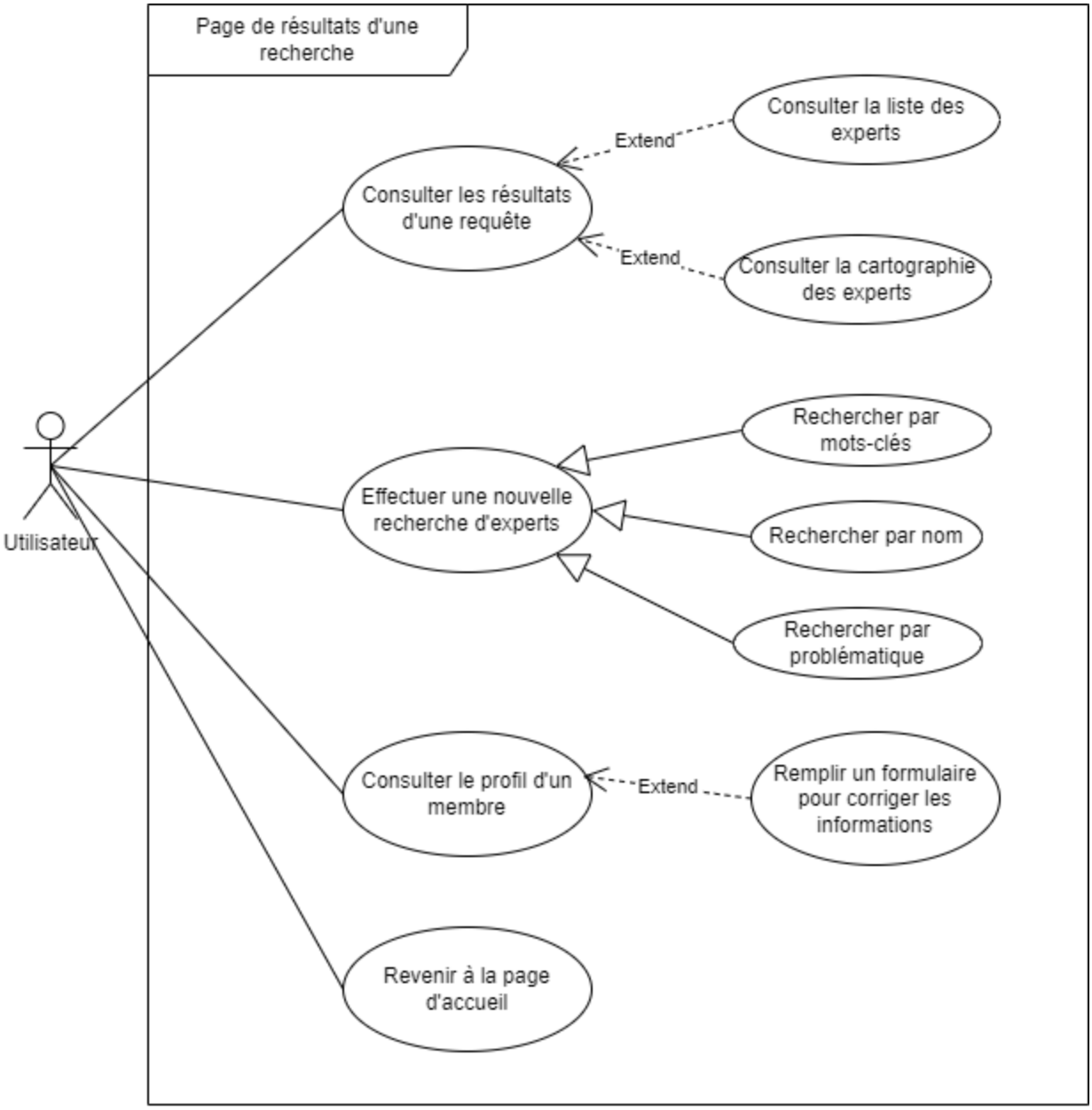

Figure 3.3 : Cas d'utilisation de la page de résultats des membres

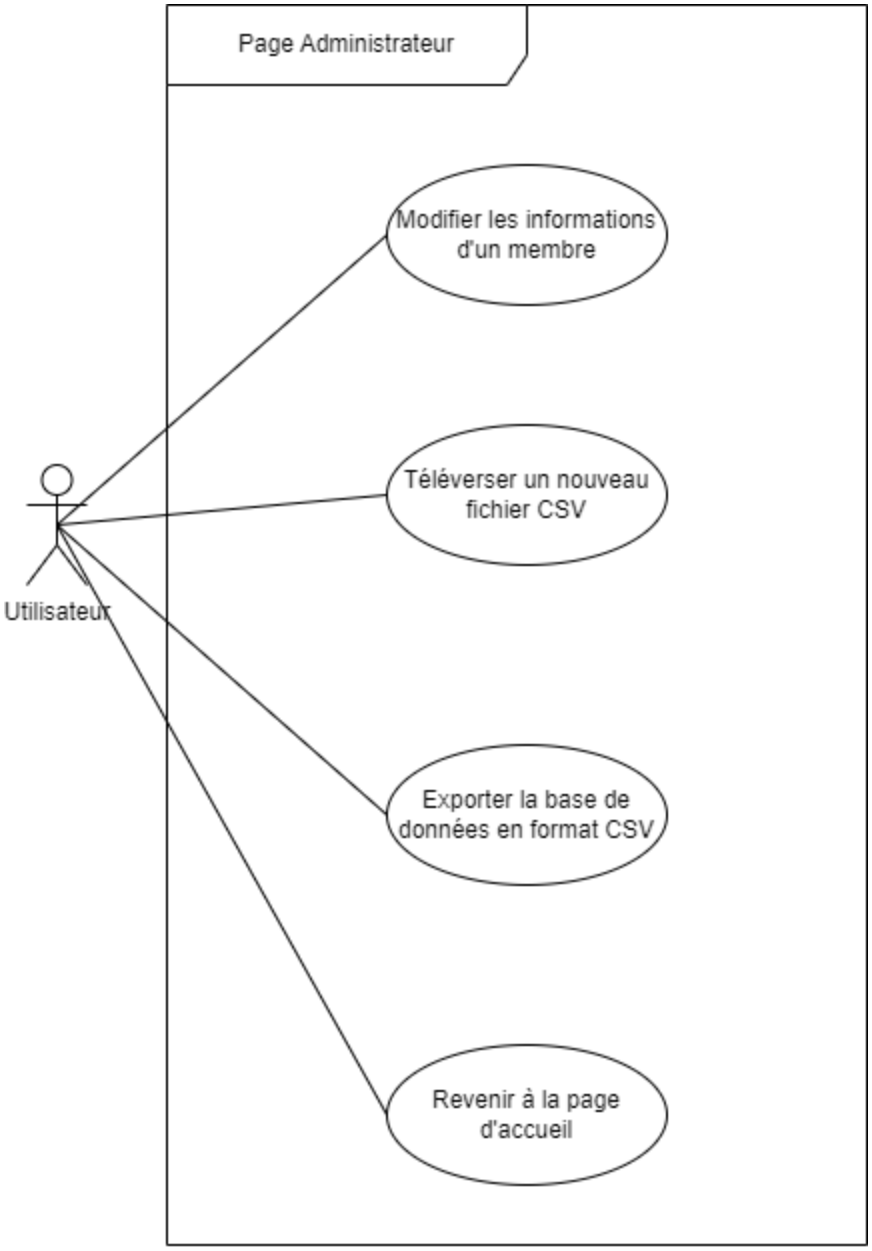

Figure 3.3 : Cas d’utilisation de la page administrateur

4. Vue logique

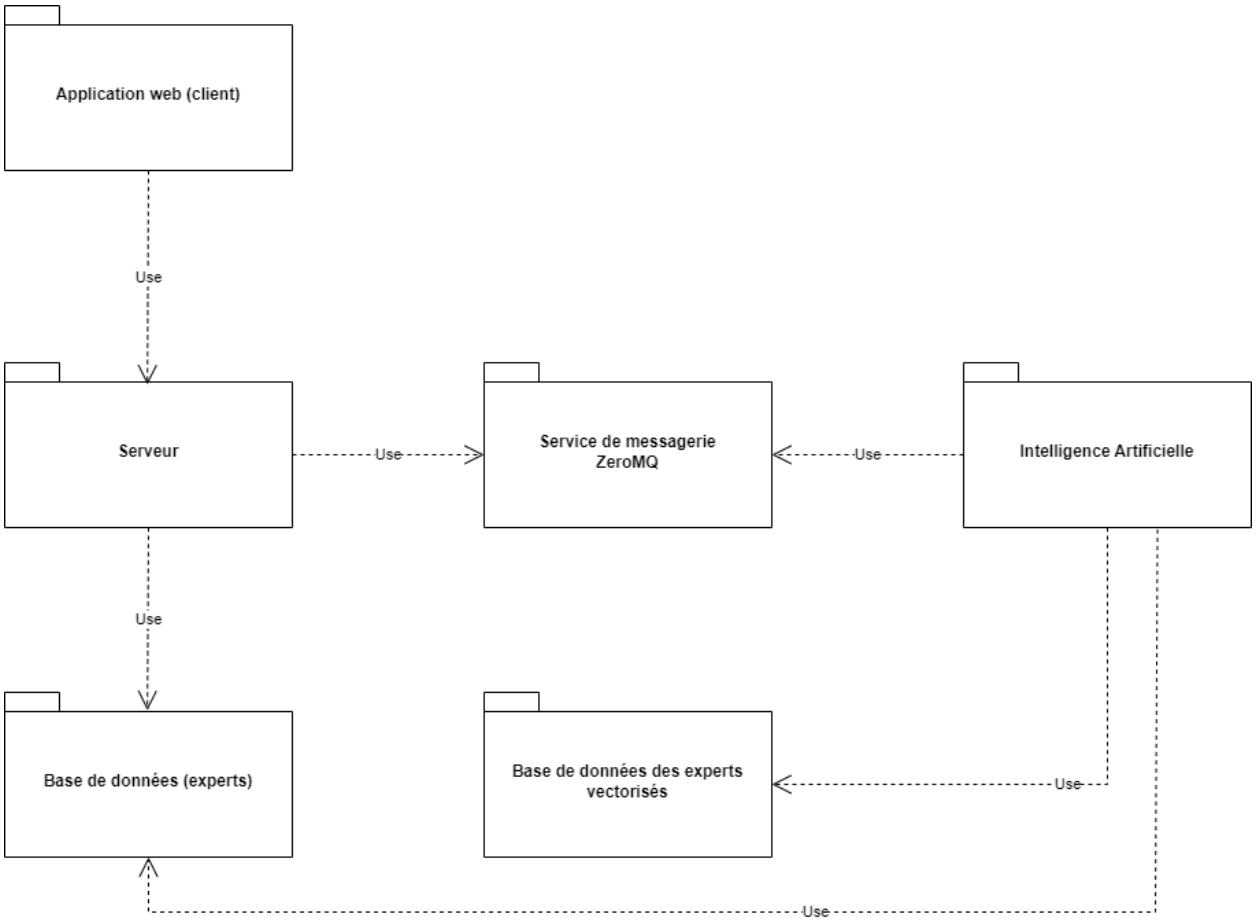

Figure 4.1 : Diagramme de paquetage du système global

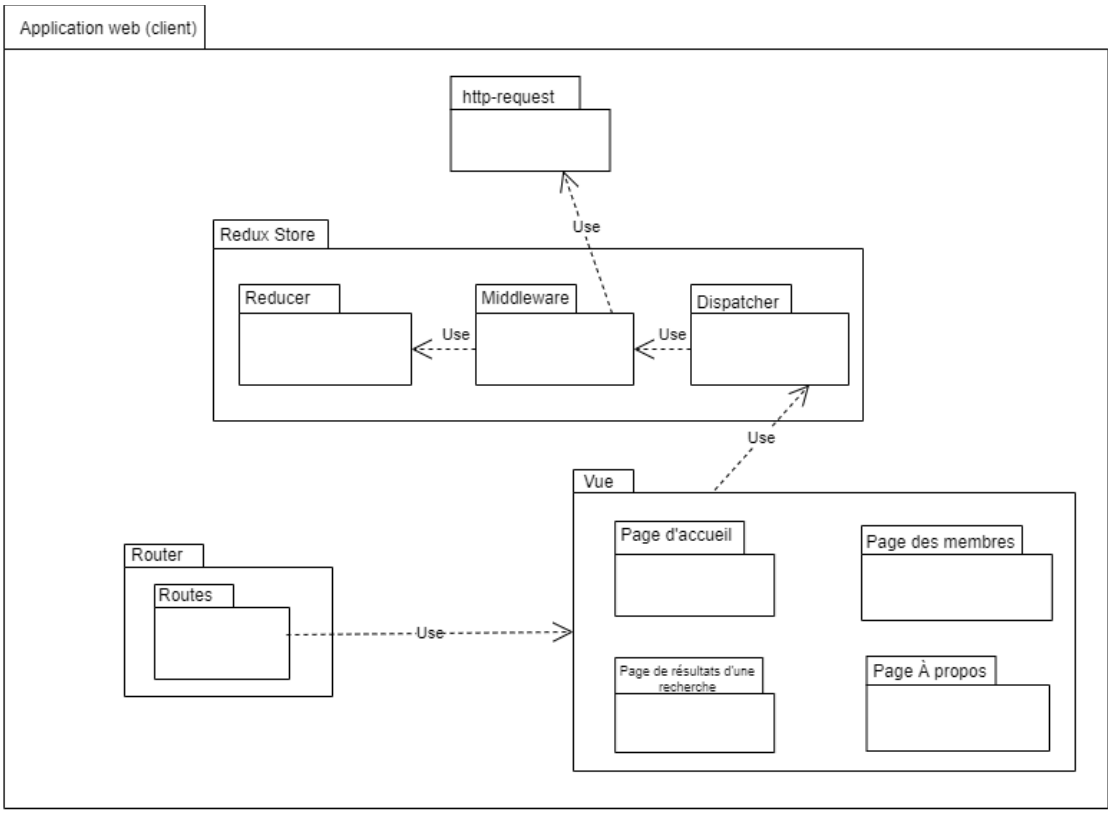

Figure 4.2 : Diagramme de paquetage du Client

http-request
Ce paquet est responsable d'envoyer des requêtes HTTP vers le serveur. Il est également responsable de recevoir les réponses et de les diriger vers le *middleware* du Redux Store.

Redux Store
Le Redux Store est un objet central qui stocke l'état global de votre application React. Il agit comme un réservoir de données centralisé que les composants peuvent interroger pour accéder aux données dont ils ont besoin.
Middleware Le *middleware* est une couche intermédiaire qui intercepte les actions avant qu'elles n'atteignent les *reducers*. Il gère également les requêtes et les réponses provenant du serveur.
Reducer Le *reducer* décrit comment l'état de l'application change en réponse à des actions. Il utilise l'état actuel et une action en entrée et sauvegarde le nouvel état. Le *reducer* est utilisé par le *middleware*.
Dispatcher Lorsqu'une action est déclenchée dans l'application, elle est envoyée au *dispatcher*, qui la transmet aux *reducers* appropriés pour mettre à jour l'état global.

Router
Le paquet "Router" représente le point d'entrée de la gestion de la navigation dans l'application. Il est responsable de la synchronisation de l'URL avec l'état de l'application. Le paquet utilise la vue pour afficher la bonne composante en fonction de l'URL.
Routes La composante de Routes associe une URL spécifique à un composant à afficher.

Vue
Le paquet "Vue" est responsable de la présentation de l'interface utilisateur de l'application. Il contient l'ensemble des composants, des pages et des éléments visuels nécessaires pour rendre l'application fonctionnelle. La vue utilise le *dispatcher* pour envoyer des actions vers le Redux Store.
Page d'accueil Ce paquet contient tous les composants et le contenu nécessaire pour afficher la page d'accueil.
Page de résultats d'une recherche Ce paquet contient tous les composants et le contenu nécessaire pour afficher la page de résultats d'une recherche.
Page des membres Ce paquet contient tous les composants et le contenu nécessaire pour afficher la page des membres.
Page à propos Ce paquet contient tous les composants et le contenu nécessaire pour afficher la page "à propos".

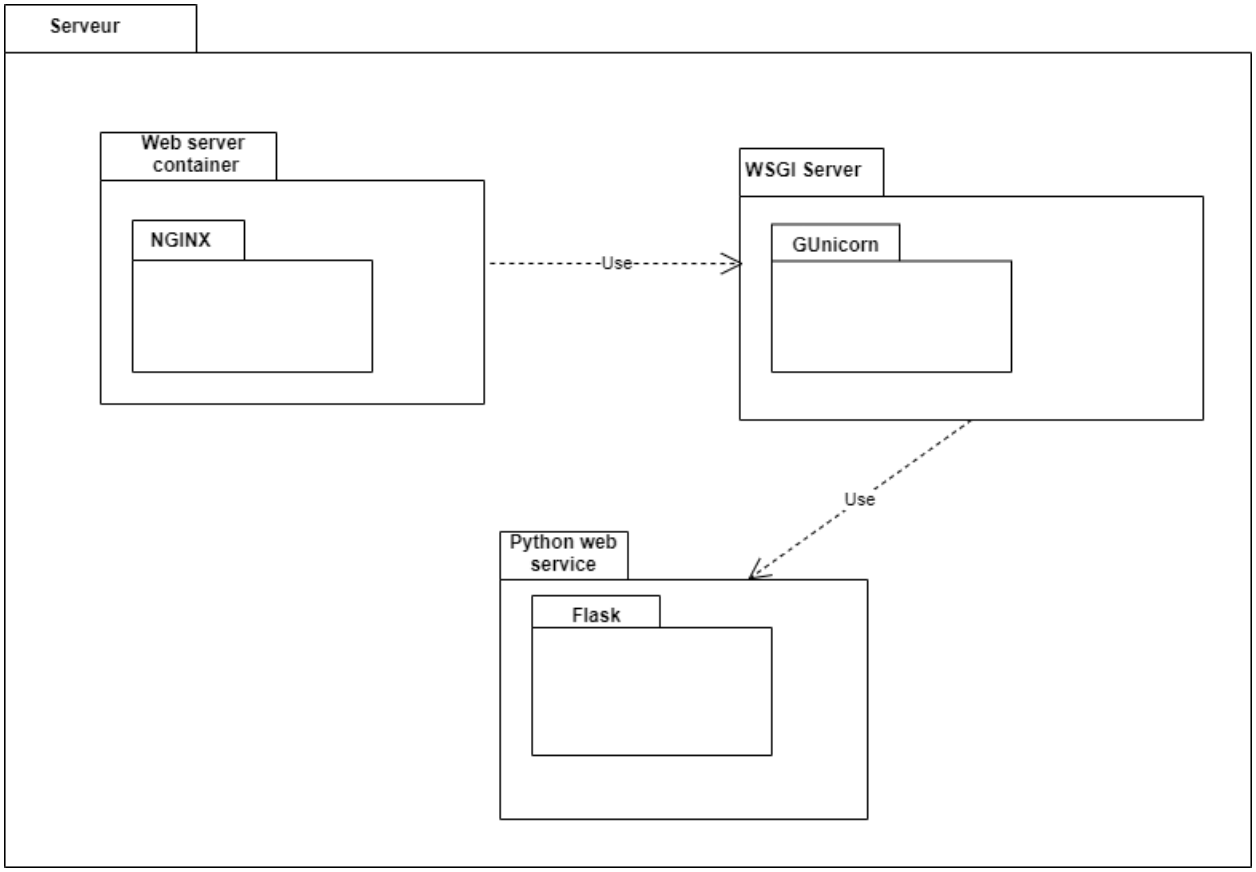

Figure 4.3 : Diagramme de paquetage du Serveur

Web server container
Le paquet "Web server container" est responsable de la gestion appropriée des requêtes d'utilisateur et de leur redirection vers le web service Flask.
NGINX Ce paquet agit en tant que proxy inversé et redirige les requêtes vers le web service Flask . Il joue donc le rôle d' équilibreur de charge . Il joue également le rôle de couche de sécurité, cachant la structure interne du web service.

WSGI server
Le paquet "WSGI server" est responsable de la configuration adéquate du web service .
GUnicorn Ce paquet permet de configurer le webservice Flask. Il assure la gestion des processus et permet la mise en place de plusieurs processus worker afin d'augmenter la tolérance aux pannes et permettre un meilleur traitement des erreurs.

Python web service
Le paquet "Python web service" est responsable de la gestion des requêtes http entrant et de leur traitement.
Flask Ce paquet permet de gérer les requêtes http entrant de les traiter et de retourner les réponses au proxy **NGINX** qui se chargera de retourner la réponse au client.

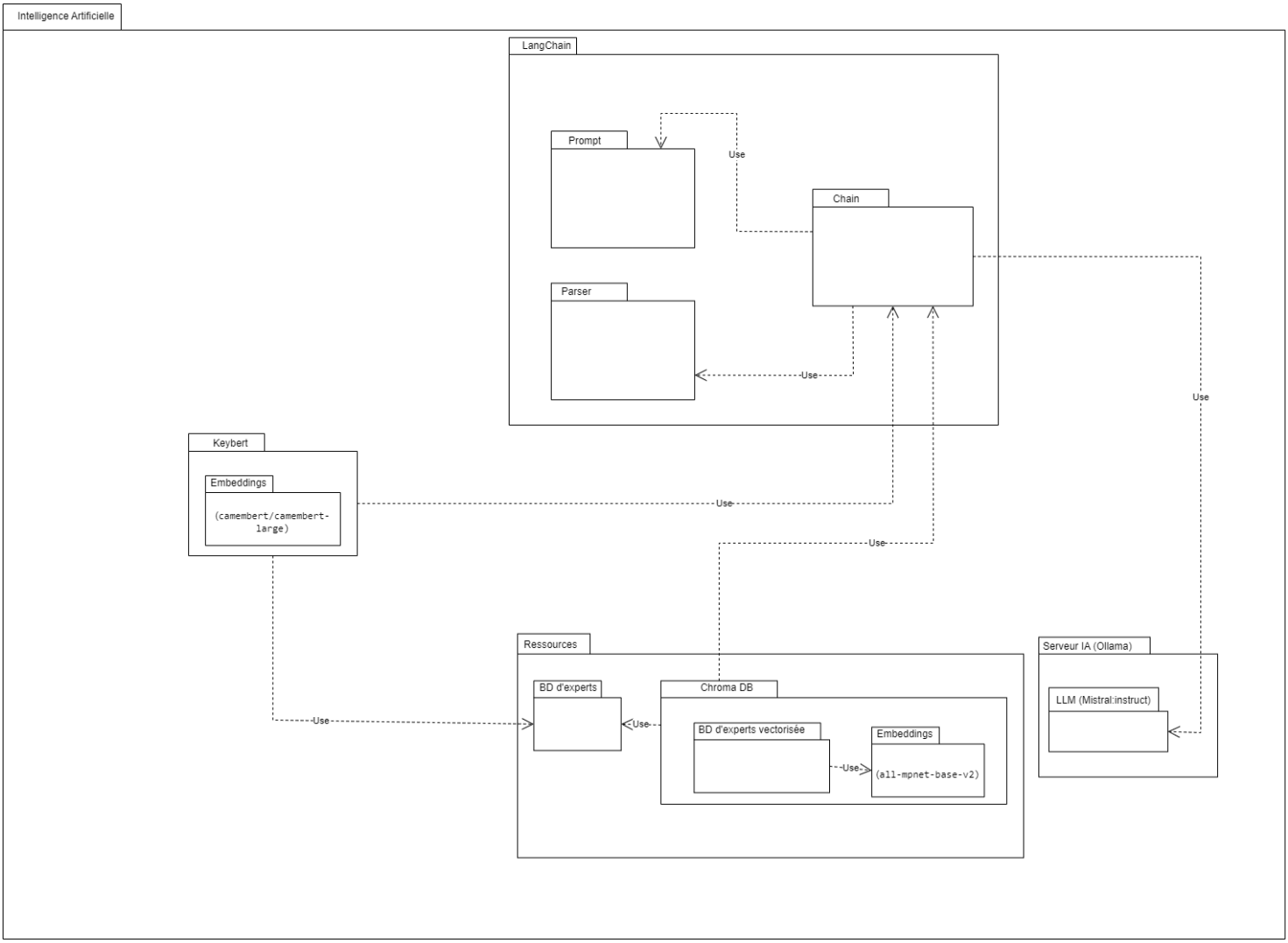

Figure 4.4 : Diagramme de paquetage de l'intelligence artificielle

Langchain
Le paquet "Langchain" contient toutes les classes permettant de construire de nouvelles fonctionnalités autour d'un Large Langage Model (LLM) .
Chain Cette classe permet la construction de la pipeline permettant l'interaction avec le Large Langage Model (LLM).
Prompt Cette classe permet de formater les instructions qui seront envoyés au LLM
Parser Cette classe s'occupe de formater la réponse qui sera retourné par le LLM dans un format bien déterminé

Keybert
Cette classe s'occupe d'extraire les mots clés de la description d'expertise de chaque expert
Embeddings (`camembert/camembert-large`) Il s'agit des SentenceTransformers qui seront utilisés afin d'effectuer la transformation des mots clés candidats en vecteur.

Serveur IA (Ollama)
Le module "Serveur IA" ou "Ollama" est un serveur local sur lequel est déployé le Large Langage Model(LLM).
Mistral Ce Large Langage Model (LLM) est responsable de deux principales tâches: dans un premier temps il est responsable de retourner les mots clés qui correspondent le mieux aux compétences d'un expert. Sa seconde tâche consiste à retourner les profils génériques qui sont nécessaires à la réalisation d'un projet, pour se faire il se base sur un ensemble d'exemples de projets et de leur listes de profils génériques.

Ressources
Le paquet "Ressources" regroupe les bases de données utilisées par le Retrieval chain et Large Langage Model .
Base de données d'experts Il s'agit de la base de données dans laquelle sont stockées toutes les données d'experts.
Chroma DB
Le module "Chroma DB" est une base de données intelligente permettant de faire de la recherche de similarité à l'aide de différentes techniques mathématiques.

Base de données d'experts vectorisés Il s'agit de la collection dans laquelle sera stockée la représentation vectorielle de toutes les données de chaque expert. Pour se faire des modèles appelés SentenceTransformers seront utilisés afin d'effectuer la transformation de texte en vecteur.
Embeddings (`all-mpnet-base-v2`) Il s'agit des SentenceTransformers qui seront utilisés afin d'effectuer la transformation de texte en vecteur.

5. Vue des processus

Cette section offre une représentation visuelle des interactions et des flux de données entre les processus clés du système, permettant ainsi de comprendre comment ils travaillent ensemble pour atteindre les objectifs définis. Cette vue séquentielle des actions et des échanges de données facilite la compréhension du fonctionnement interne du système. Les diagrammes de séquence et les composantes sont utilisés pour illustrer ces interactions de manière claire et concise. Il est à noter que, dans le but d'alléger, nous avons volontairement omis les séquences liées aux erreurs.

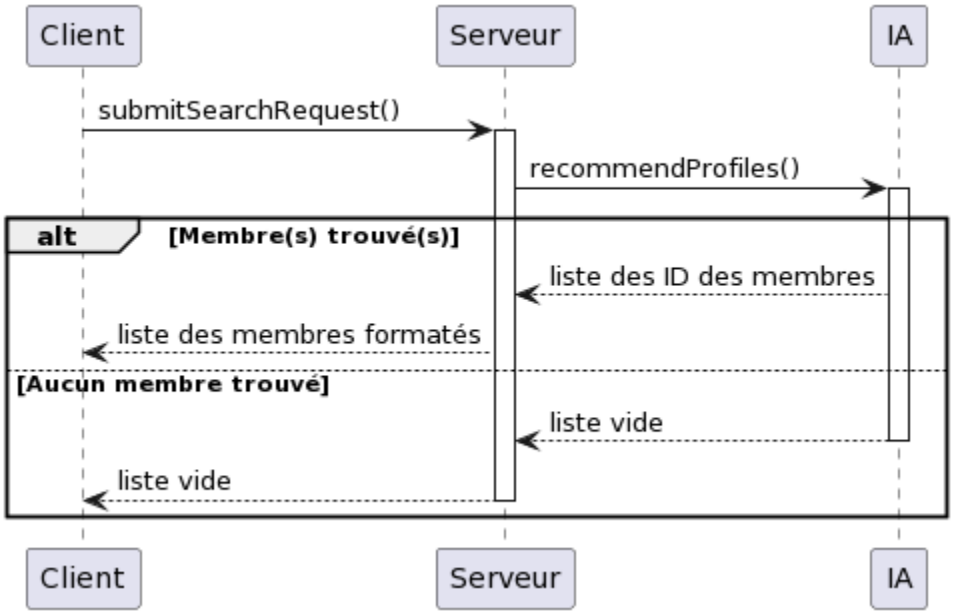

Figure 5.4 : Diagramme de séquence pour la recherche de membre avec l'écriture d'une requête

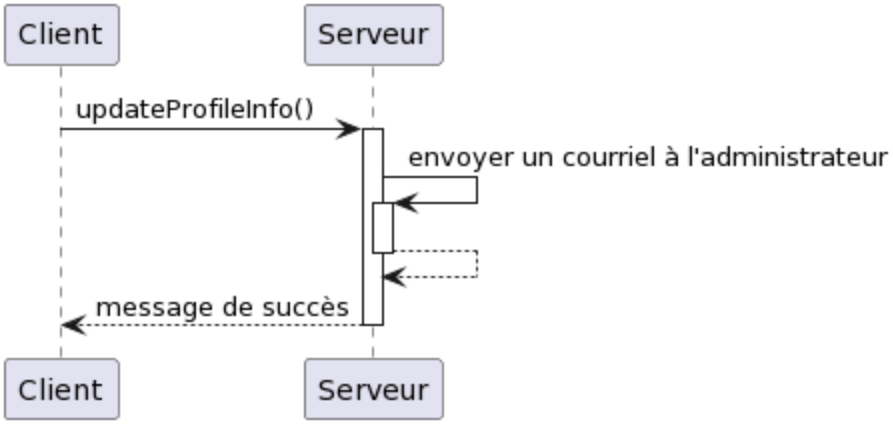

Figure 5.5 : Diagramme de séquence de la correction d'information d'un membre

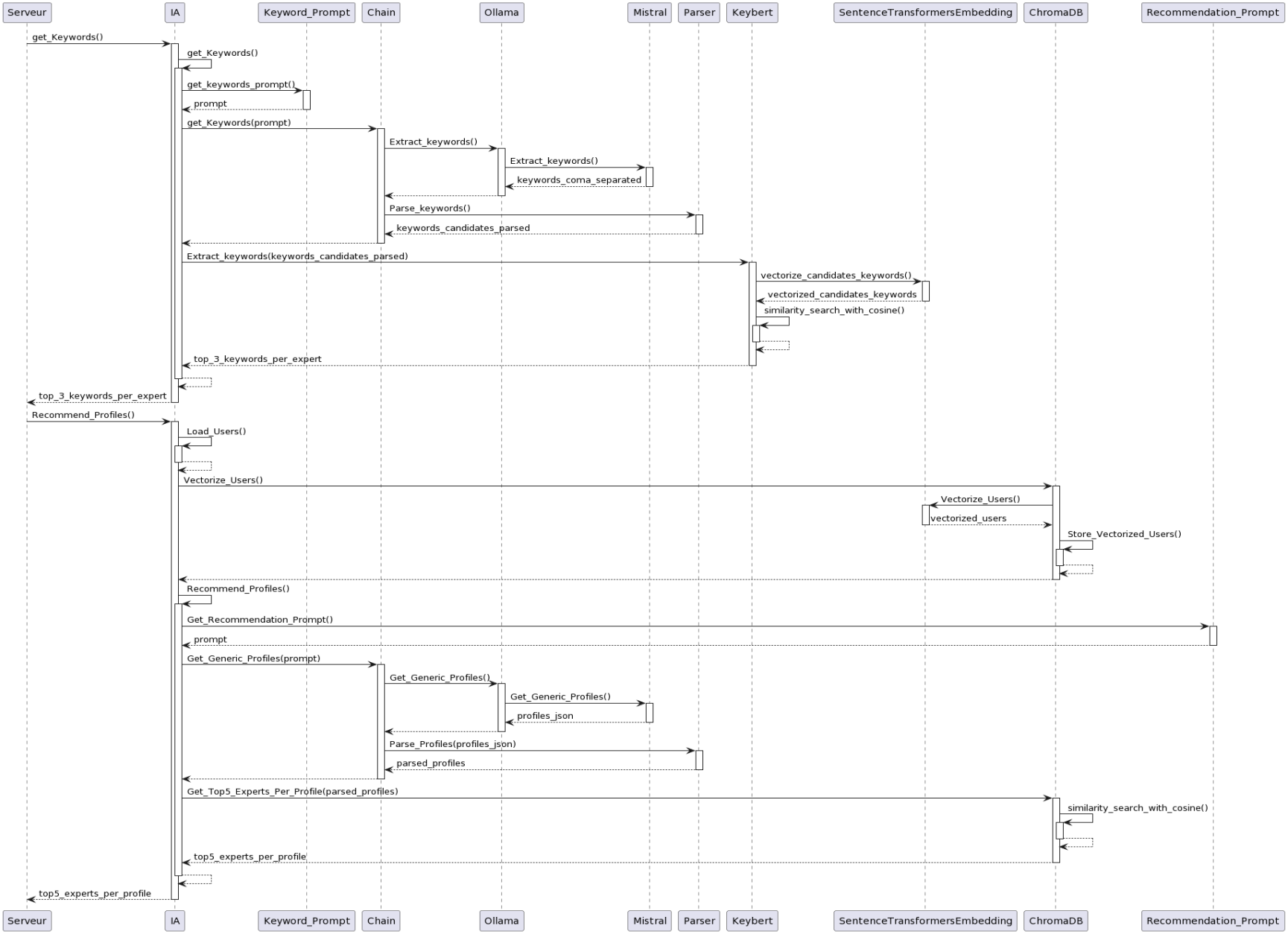

Figure 5.6 : Diagramme de séquence du fonctionnement interne du module de l'intelligence artificielle

6. Vue de déploiement

Le diagramme de déploiement est présenté ci-dessous. Chaque nœud de ce diagramme représente une composante physique. À l'intérieur de chaque nœud se trouvent l'environnement d'exécution et l'artéfact exécuté sur la composante physique.

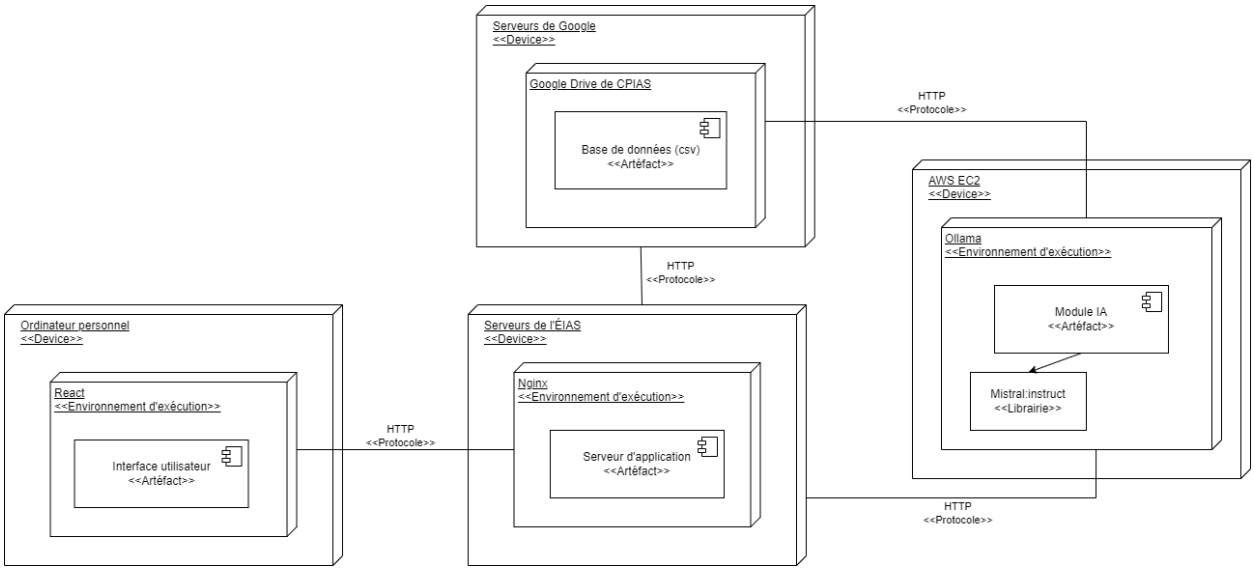

Figure 6.1 : Diagramme de déploiement

7. Taille et performance

La capacité de la base de données est de 15 Go, ce qui est plus qu'assez pour nos besoins, considérant que le fichier .csv contenant les données des utilisateurs est de l'ordre de quelques centaines de Ko. Cette taille n'est qu'une minime fraction de la capacité maximale de 15 Go. Le serveur d'application web est hébergé sur les serveurs du fournisseur en infonuagique AWS. Celui-ci est déployé sur une instance EC2 d'AWS, ayant une mémoire d'au moins 16 Go de mémoire vive. Cette instance devrait suffire pour répondre aux besoins des utilisateurs et traiter les requêtes de ceux-ci dans des délais raisonnables.

Cependant, le modèle IA, s'exécutant sur la seule et unique instance du serveur, sera inévitablement surchargée lorsque plusieurs utilisateurs feront une requête. Néanmoins, le service EC2 d'Amazon est redimensionnable et il est possible d'ajuster le nombre d'instances pour servir plus d'utilisateurs à la fois et même d'augmenter les capacités de calcul afin de répondre aux requêtes plus rapidement. L'architecture sera évaluée selon le nombre d'utilisateurs qu'elle peut servir et selon le temps de réponse. Le modèle utilisé (Mistral) requiert au minimum 8 Go de mémoire vidéo (*VRAM*) pour avoir des résultats acceptables. Cependant, il est recommandé d'avoir 16 Go de VRAM.