Sierre, le 21.10.24

Analyse préliminaire

Table des matières

Introduction

Ce document s'inscrit dans le cadre de la collaboration entre la Digital Team Academy et du site HES DevPro pour la réalisation d'un projet portant sur la réalisation d'un chatbot.

Contexte de réalisation

Le projet consiste en la création d'un chatbot visant à assister les utilisateurs des modules de formation continue (DevPro). Ce chatbot simplifierait les questions des utilisateurs du site dans le but de soulager le secrétariat en répondant aux questions courantes déjà couvertes sur le site lui-même. Grâce à sa connaissance des informations disponibles (conditions, qualifications requises, niveaux, FAQ, etc.), cet assistant rendrait l'accès aux informations plus facile et éviterait des demandes inutiles.

Contenu du document

Ce document présente une analyse visant à identifier une stack technologique théorique pour le développement du projet, en se concentrant sur deux solutions principales : Rasa et MindsDB. L'évaluation inclut un benchmark détaillé et une analyse des risques associés à chacune de ces technologies. Chaque solution est étudiée en fonction de son niveau d'abstraction, des technologies requises, des prérequis techniques, ainsi que des dépendances. Les choix sont justifiés en tenant compte des besoins spécifiques du projet, notamment en termes de performance, de compatibilité et de facilité d'intégration avec les systèmes existants.

Recherche

Cette section examine et évalue les technologies et outils les plus appropriés pour la conception du chatbot en fonction des exigences et contraintes du projet.

Exigences

Exigences

Le projet comporte plusieurs exigences initiales :

Exigence	Potentialité
Doit utiliser idéalement une stack open source, en veillant à respecter les licences.	
Peux tirer parti de l'infrastructure Microsoft en raison des abonnements de la HES.	Pourrait être bénéfique pour la puissance de calcul et l'utilisation de services IA spécifiques (processeurs déjà adapté pour l'IA [NLP, LLM])
Doit proposer une IA personnalisable : Enrichissement de ses connaissances (Scraping, PDF, Web Services) Proposition de scénario conversationnel de réponse et navigation.	L'utilisation de Workflow sur la partie « front-end » du projet pourrait être un avantage pour le client final.
Doit pouvoir être intégrer sur l'infrastructure existante Accès par sous-url (minisite) du site. Ne pas nécessiter de serveur physiquement dédié.	

Remarques initiales

Il est essentiel de prêter attention à certains éléments qui pourraient être limitants :

- **Utilisation des technologies IA et matériel requis**

  L'intégration de l'IA, comme les modèles LLM et le traitement du langage naturel (NLP), peut nécessiter des ressources considérables ainsi que des processus spécialisés (comme les réseaux neuronaux) pour garantir une expérience fluide pour l'utilisateur.

  Il est donc crucial de prendre en compte ces aspects afin d'éviter un service lent. Ces traitements auto-hébergés (serveurs Docker) requièrent une puissance de calcul adéquate et un matériel le soutenant.

  Il pourrait être judicieux d'utiliser les services Microsoft (comme le modèle Azure OpenAI) pour déléguer le calcul des modèles et des embeddings.

- **Niveau d'abstraction**

  Sur le marché, il existe des outils no-code extrêmement bien conçus et puissants. Il est également possible de construire entièrement sa stack en utilisant des framework et du code. En fonction des besoins spécifiques du projet, il peut être judicieux d'opter pour une solution hybride, combinant des éléments low-code pour la partie principalement front-end, tout en intégrant des composants personnalisés en code pour répondre à des exigences plus complexes ou spécifiques. Cette approche permet de bénéficier de la flexibilité et de l'efficacité des outils no-code tout en conservant le contrôle et la personnalisation que le développement traditionnel en code peut offrir.

Analyse

Donnée

Les deux solutions retenues, étudiées et présentées sont un environnement basé sur « Rasa » et un autre basé sur « MindsDB ».

Benchmark

Ce benchmark permet de comparer les performances et les caractéristiques des différentes solutions afin de déterminer celle qui répond le mieux aux besoins du projet.

Introduction

Environnement basé sur « Rasa »

Rasa est une plateforme open source permettant de créer des chatbots et des assistants virtuels avancés en utilisant le traitement du langage naturel et la reconnaissance d'intention, tout en offrant un haut degré de personnalisation pour l'orchestration des conversations.

Environnement basé sur « Mindsdb »

MindsDB est un outil de machine learning automatisé et LLM qui permet d'intégrer des modèles d'IA directement dans des bases de données, facilitant ainsi les prédictions et analyses en temps réel à partir de données structurées sans nécessiter de code complexe. Mindsdb permet aussi de créer ses propres agents, d'automatiser des processus d'intégration de données pour du fine-tuning & embeddings.

Généralité

Rasa	Mindsdb
• **Éditeur** : Rasa Technologies GmbH • **Version en cours** : 3.5.9 (octobre 2024)	• **Éditeur** : MindsDB Inc. • **Version en cours** : 24.8 (octobre 2024)

Licence

Rasa	Mindsdb
Elle est publiée sous la licence **Apache License 2.0**,	Il utilise la licence **Elastic Licence 2.0** pour son module « core », le reste étant sous la licence « MIT ».
Licence open source permissive permettant une utilisation large, y compris commerciale, avec peu de restrictions.	Licence open source qui permet l'utilisation commerciale des logiciels Elastic, tout en imposant des restrictions sur la redistribution pour protéger les intérêts d'Elastic.
Il n'y a pas d'obligation de redistribuer le code modifié sous la même licence.	Il n'y a pas d'obligation de redistribuer le code modifié sous la même licence.
Il faut attribuer (reconnaître) le travail original en incluant un avis de droits d'auteur et une copie de la licence dans toute distribution du logiciel.	Il faut attribuer (reconnaître) le travail original en incluant un avis de droits d'auteur et une copie de la licence dans toute distribution du logiciel.

Infrastructure

Rasa	Mindsdb
Type d'infrastructure	
Framework de traitement du langage naturel (NLP) en Python **Docker**	Plateforme de Machine Learning & LLM orientée sur les bases de données. **Docker**

Rasa peut-être facilement intégré dans des environnements conteneurisés à l'aide de Docker. **Flexibilité** Rasa offre une grande liberté d'exécution grâce à sa capacité à être intégré avec différents backends et services (par exemple, bases de données, APIs), et à interagir avec d'autres frameworks et outils d'IA, ce qui le rend grandement personalisable.	MindsDB s'exécute initialement dans des conteneurs Docker, permettant une installation simple et rapide, ainsi qu'une gestion efficace des dépendances **Flexibilité** MindsDB peut se connecter directement aux bases de données existantes / autres sources de données, ce qui permet aux utilisateurs de créer et d'exécuter des modèles de Machine Learning / LLM sans avoir besoin de déplacer les données.
Gestion des modèles IA	
Prend en compte les principaux modèles du marché notamment OpenAI, Azure OpenAI tant pour l'embeddings que le LLM.	Prend en compte les principaux modèles du marché notamment OpenAI, Azure OpenAI tant pour l'embeddings que le LLM.
API	
L'API de Rasa est proposée sous forme d'API REST, permettant une intégration facile avec diverses applications et services.	L'API de Mindsdb est proposée sous forme d'API REST, Typescript SDK Client, permettant une intégration facile avec diverses applications et services.

Prérequis et performance

Rasa	Mindsdb
Pré-requis	
- Python jusqu'à 3.10 - Dépendances python (rasa) - Possibilité de le mettre dans un Docker	- Docker
Niveau de profondeur	
Bas niveau. Rasa offre un contrôle très granulaire sur chaque aspect de la gestion des dialogues, des intents, et des entities.	Plus haut niveau que Rasa. MindsDB propose une approche simplifiée où l'utilisateur interagit principalement via

L'utilisateur doit configurer manuellement les pipelines NLP, les politiques de dialogue, et les modèles, ce qui permet une personnalisation poussée mais nécessite une expertise technique approfondie pour la mise en place et l'ajustement.	des requêtes SQL, déléguant la gestion du machine learning à l'outil. Cela réduit considérablement la complexité, car les aspects techniques de la formation des modèles et de l'inférence sont pris en charge automatiquement, permettant une utilisation rapide et efficace sans entrer dans les détails du fonctionnement interne.
Scalabilité	
Rasa peut-être facilement mis à l'échelle en fonction du nombre d'utilisateurs, en particulier dans des environnements de cloud ou avec Kubernetes.	MindsDB peut être mis à l'échelle pour traiter des volumes de données importants, surtout si la base de données sous-jacente est bien optimisée. Il s'intègre bien avec des systèmes de bases de données populaires (MySQL, PostgreSQL, MongoDB, etc.), ce qui permet de bénéficier des capacités de ces bases en termes de gestion de volume et de concurrence.
Extensibilité	
Très extensible. Rasa permet d'ajouter de nouveaux composants, d'intégrer des API externes, de personnaliser les pipelines NLP, et de créer des workflows complexes pour les dialogues.	Extensibilité limitée. Bien que MindsDB soit conçu pour intégrer des modèles de machine learning dans des bases de données, ses capacités d'extension se limitent principalement à ce domaine. L'ajout de nouvelles fonctionnalités est plus restreint.

Communauté et support

Rasa	Mindsdb
Communauté	
Grande communauté open-source avec de nombreux contributeurs. De	Communauté en croissance, plus petite que celle de Rasa, mais avec une documentation disponible et une

nombreuses ressources (tutoriels, forums, documentation) sont disponibles, ce qui facilite la résolution des problèmes et l'apprentissage.	présence active sur des forums spécifiques.
Support	
Rasa propose un support commercial payant pour les entreprises qui ont besoin d'assistance professionnelle ou de services supplémentaires.	MindsDB propose également un support payant pour les entreprises, avec des options d'assistance et d'accompagnement pour les projets critiques.

Sierre, le 21.10.24

Conclusion

En conclusion, bien que **MindsDB** et **Rasa** offrent tous deux des avantages intéressants, notre choix doit être orienté selon les besoins spécifiques et des contraintes techniques du projet.

Rasa, avec son **contrôle granulaire** sur la gestion des dialogues, des intents et des pipelines NLP, nous permettrait de personnaliser en profondeur notre solution. Son **extensibilité** nous donnerait la possibilité d'ajouter des composants, d'intégrer des API externes, et de concevoir des workflows complexes adaptés à nos exigences. De plus, étant sous la **licence Apache 2.0**, Rasa nous offrirait une grande liberté d'utilisation, y compris pour l'intégration dans le site de DevPro, sans restriction significatives liées à la licence.

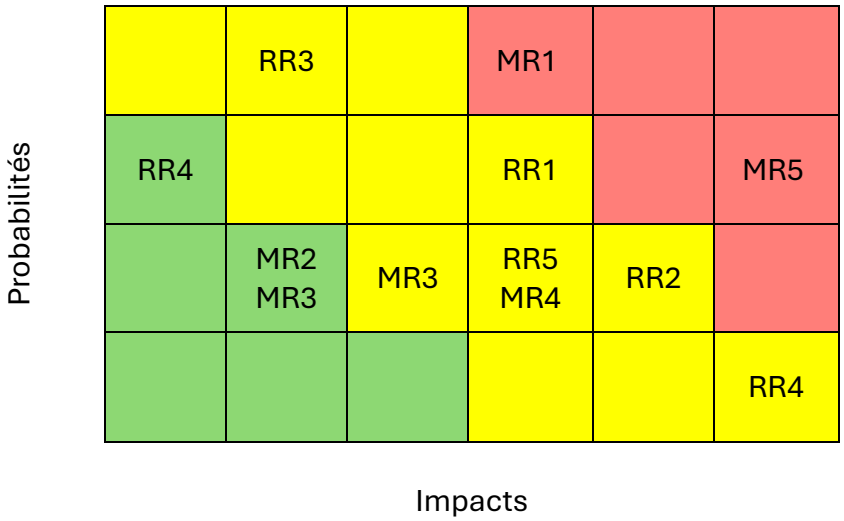
De son côté, **MindsDB** nous permettrait de simplifier l'intégration du machine learning grâce à son approche basée sur des requêtes SQL et son intégration directe avec des bases de données existantes. Cette approche faciliterait le développement de modèles de machine learning sans avoir à déplacer les données. Cependant, son **extensibilité est plus limitée**, et la **licence Elastic 2.0** impose certaines contraintes sur l'utilisation commerciale.

Compte tenu de ces éléments, bien que **MindsDB** présente des avantages pour des cas d'utilisation spécifiques, **Rasa** nous offrirait plus de flexibilité, une personnalisation accrue et une liberté d'utilisation, faisant de lui un choix plus adapté à nos besoins globaux. De plus, Rasa nous permet d'apprendre plus en profondeur les différents concepts des LLM, ce qui est un atout pour notre semestre et un bon compromis entre les parties.

Sierre, le 21.10.24

Matrice des risques

Ci-joint la présentation des risques, illustrée et catégorisée à l'aide de la matrice de probabilité et d'impact ci-dessus.



Risques évalués

RR1 : Rasa Risque 1 : Personnalisation du chatbot

Bien que Rasa soit personnalisable, ajuster finement les réponses du chatbot à partir de des données peut nécessiter du temps et de l'expertise en NLP.

RR2 : Rasa Risque 2 : Réimportation et intégration des nouvelles données

L'importation de nouvelles données et leur réintégration dans le chatbot pour créer de nouveaux embeddings peut entraîner des latences, des défis techniques mais également modifier le comportement de l'exécution du modèle.

RR3 : Rasa Risque 3 : Complexité de gestion des embeddings

Gérer les embeddings pour chaque nouveau cours ou modification via le webservice peut devenir complexe si les mises à jour sont fréquentes.

RR4 : Rasa Risque 4 : Gestion des intentions non couvertes

Avec des données et un scope limité (cours, FAQ, contact), il est possible que certaines Intentions importantes ne soient pas couvertes par le chatbot, entraînant des réponses devant être renvoyé par généralité.

RR5 : Rasa Risque 5 : Confusion dans les cours non détaillés

Avec des données des cours, la sélection en passant par un chemin général comme la recherche des prérequis sur la globalité les cours peuvent créer des confusions et

nécessiter de passer par un scénario sélectionnant initialement le cours dans le processus.

MR1 : Mindsdb Risque 1 : Complexité de personnalisation

Bien que MindsDB permette une personnalisation, configurer les embeddings de manière spécifique aux besoins du projet pourrait nécessiter une expertise et une accessibilité dédiée sur la plateforme.

MR2 : Mindsdb Risque 2 : Réimportation et intégration des nouvelles données

L'importation de nouvelles données et leur réintégration dans le chatbot pour créer de nouveaux embeddings peut entraîner des latences, des défis techniques mais également modifier le comportement de l'exécution du modèle.

MR3 : Mindsdb Risque 3 : Complexité de gestion des embeddings

Gérer les embeddings pour chaque nouveau cours ou modification via le webservice peut devenir complexe si les mises à jour sont fréquentes.

MR4 : Mindsdb Risque 4 : Risques de compatibilité avec d'autres outils NLP / embedding

L'intégration de MindsDB avec d'autres outils de NLP ou d'embeddings personnalisés peut poser des problèmes de compatibilité, nécessitant des adaptations supplémentaires.

MR5 : Mindsdb Risque 5 : Licence de déploiement davantage complexe

Déployer MindsDB dans un contexte public ou commercial tout en respectant les obligations d'une licence peut s'avérer plus complexe.

MR6 : Mindsdb Risque 6 : Dépendance à l'infrastructure Docker

La gestion de l'infrastructure Docker pour chaque mise à jour des données et la réintégration des embeddings peut devenir lourde pour des projets à faible volume.

Sierre, le 21.10.24

Présentation du PoC 1 : RASA

Introduction

Ce Proof-of-Concept (POC) illustre l'intégration de **Rasa**, un framework de traitement du langage naturel, avec une base de données vectorielle et un modèle de langage large (**LLM**). L'objectif est de permettre à un chatbot d'interagir dynamiquement avec les utilisateurs en comprenant leurs intentions et en générant des réponses pertinentes à partir d'une base de connaissances.

Objectifs du POC

L'objectif principal de ce POC est de démontrer la faisabilité de la création d'un chatbot hautement personnalisé capable de répondre efficacement aux besoins des utilisateurs. À travers ce projet, nous visons à établir une solution qui intègre des modèles de langage avancés (LLM) et des bases de données vectorielles pour fournir des réponses pertinentes et précises aux questions posées par les utilisateurs.

Technologies utilisées

Pour la réalisation de ce POC, nous avons intégré plusieurs technologies clés. **Milvus**, une base de données vectorielle, est utilisée pour stocker et rechercher efficacement les embeddings de données non structurées. **Sentence Transformers** nous permettent de convertir les requêtes et les contenus en vecteurs numériques, facilitant ainsi la recherche de similarité. **Rasa** sert de framework pour gérer la compréhension du langage naturel et orchestrer les dialogues avec les utilisateurs. **Pdfminer** est utilisé pour extraire des informations à partir de fichiers PDF, et l'**OpenAI API** permet d'enrichir les réponses du chatbot avec des modèles de langage avancés. Ensemble, ces technologies assurent une interaction fluide et personnalisée pour les utilisateurs.

Différentes étapes

1. **Input utilisateur**

   L'utilisateur interagit avec le système via une requête textuelle, qui est analysée par le module de compréhension du langage naturel (NLU) de Rasa.

2. **Intent (NLU)**

   Le système identifie l'intention de l'utilisateur (FAQ, cours, contact, etc.) et catégorise la demande.

3. **Stories, règles et actions**

   En fonction de l'intention, des **stories** définies dans Rasa guident les **actions** que le chatbot doit exécuter, telles que la récupération d'informations ou la génération d'une réponse.

4. **VectorDB et Embeddings**

   Si l'intention nécessite une recherche d'informations spécifiques, la requête de l'utilisateur est d'abord vectorisée en utilisant un modèle d'embeddings (par exemple, avec des modèles comme ceux de SentenceTransformers). Ce vecteur est ensuite comparé aux embeddings pré-calculés stockés dans la base de données vectorielle (VectorDB), qui contient des vecteurs pour les catégories de FAQ, cours et contacts. En fonction de l'intent identifié, la comparaison est effectuée avec les embeddings correspondants (par exemple, les embeddings des FAQ si l'intent est lié aux questions fréquentes). Le système retourne ensuite le vecteur le plus proche, qui correspond à la réponse ou information la plus pertinente à fournir.

5. **Prompt vers LLM**

   Une fois l'information identifiée, un **prompt** est généré et envoyé au modèle de langage large (LLM), qui renvoie une réponse structurée et adaptée au contexte de la demande utilisateur.

6. **Memory (non intégré dans ce POC)**

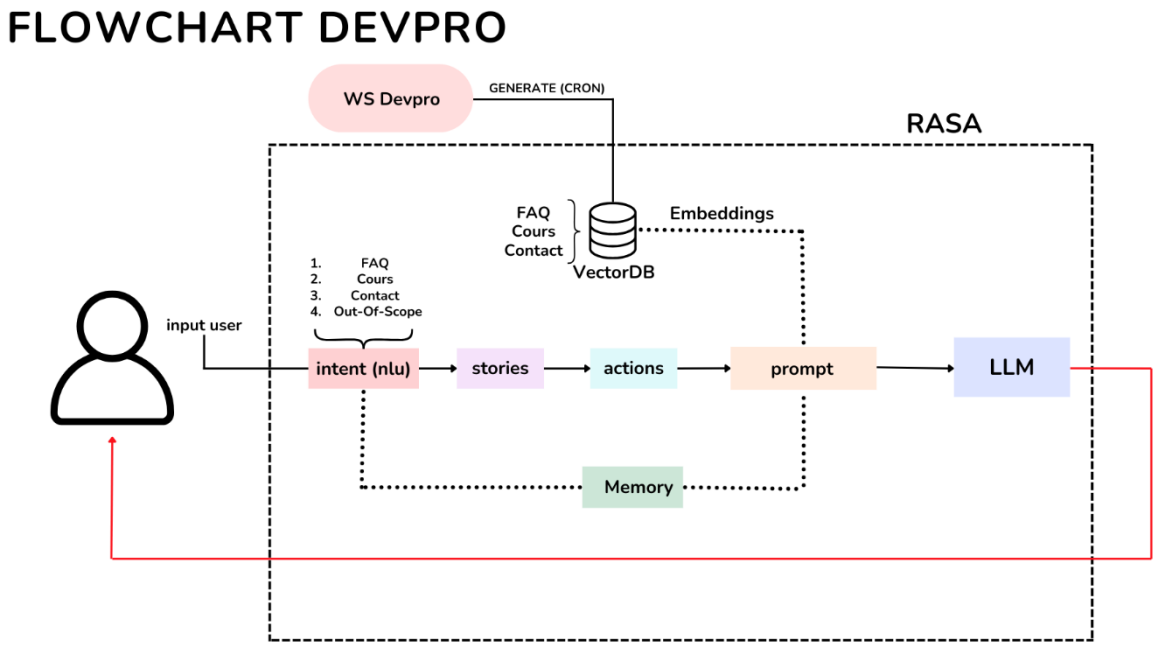
   Le système maintient une mémoire pour gérer le contexte de la conversation et assurer des réponses cohérentes lors de l'interaction continue.

7. **WS DevPro (simulé par PDF / DB SQL)**

   Un service externe (WS Devpro) génère et alimente périodiquement la base de données vectorielle via un processus de génération automatique (CRON).

Sierre, le 21.10.24

Schéma



Le POC en pratique

Le POC est structuré en deux parties, se concentrant principalement sur deux axes : les **embeddings** et **la gestion des intention**s.

Sierre, le 21.10.24

Présentation du PoC 2 : Mindsdb

Introduction de la problématique du projet

La HES-SO, avec plus de 10 000 enseignant·es, fait face à une demande croissante de formation et de gestion administrative via son centre de développement professionnel, DevPro. DevPro offre une variété de cours et assure la gestion des attestations didactiques en collaboration avec les différentes directions. L'afflux constant de questions administratives et de demandes d'information pose un défi pour l'administration, qui cherche à soulager cette pression.

La problématique centrale est donc la suivante

Comment améliorer l'expérience utilisateur sur le site DevPro tout en réduisant la charge administrative liée aux demandes d'information récurrentes ? La solution identifiée est la création d'un chatbot basé sur l'intelligence artificielle (IA), capable de répondre aux questions fréquentes des utilisateur·rices et de pointer vers les ressources appropriées.

Critères principaux

- Technologies open-source : Permet de réduire les coûts et de garantir la flexibilité et la transparence du code.
- Solution self-hosted : Important pour des raisons de contrôle des données et de respect de la vie privée.
- IA customizable : L'outil doit permettre des ajustements spécifiques à l'institution, tels que l'entraînement sur des données propres à la HES-SO.
- Utilisation de technologies d'Embeddings/RAG (Retrieval-Augmented Generation) : Essentiel pour que le chatbot puisse répondre aux questions en recherchant dans une base de connaissances existante et en générant des réponses précises à partir des documents fournis.

Sierre, le 21.10.24

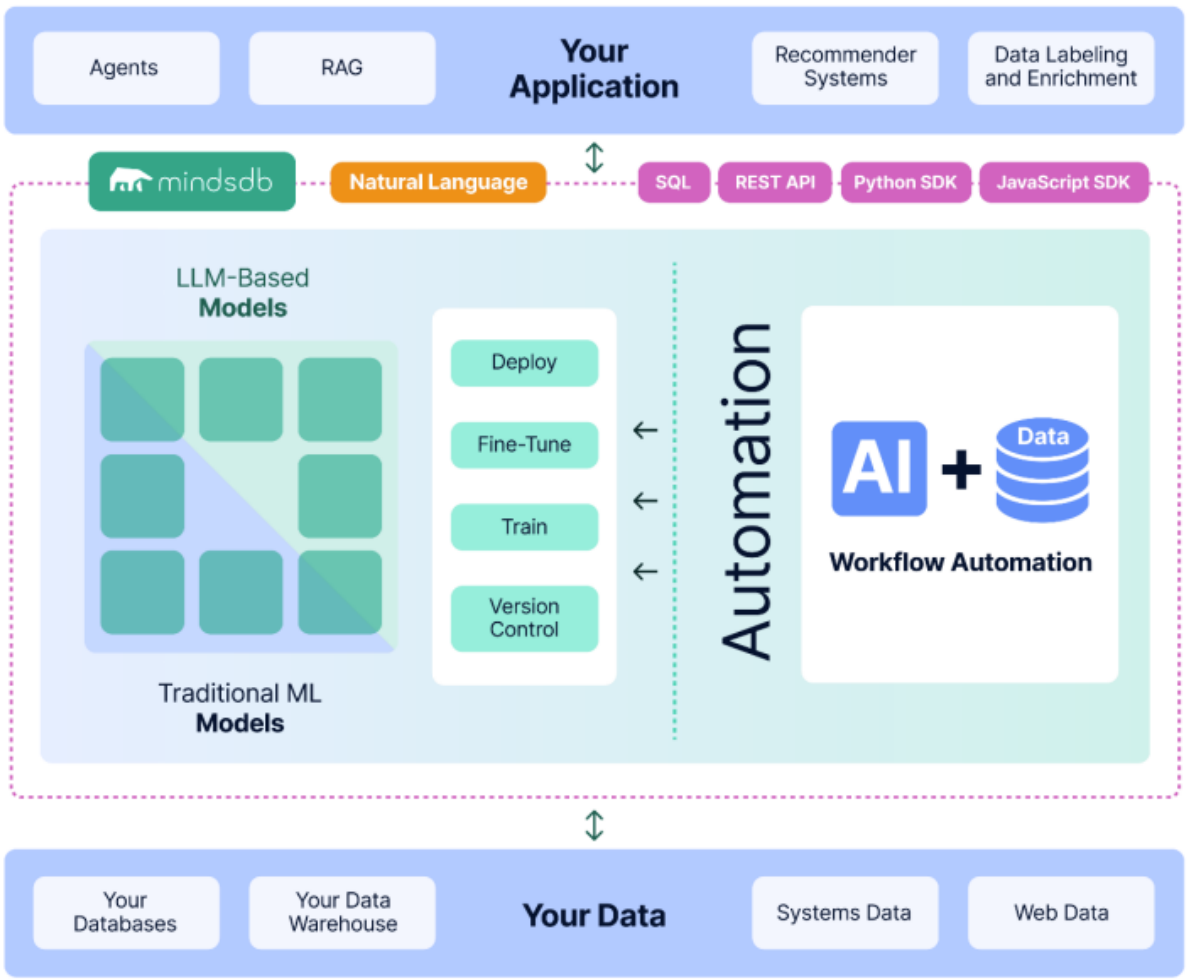
Choix de la technologie MindsDB en backend et Next.js en frontend

Pourquoi cette solution et comment cela fonctionne ?

MindsDB est une plateforme d'IA qui permet de créer, déployer, et ajuster des modèles IA en temps réel en se connectant à des sources de données variées comme des bases de données, des vector stores, ou des applications. Elle propose un ensemble d'outils simples, compatibles avec des technologies standards comme SQL, pour faciliter l'intégration de l'IA dans les applications existantes.

Pourquoi MindsDB ?

MindsDB se distingue par sa capacité à créer des agents d'IA personnalisables, en intégrant des compétences comme le text-to-SQL, la recherche sémantique ou les modèles de langage naturel (LLM). Elle permet d'utiliser des technologies comme les embeddings pour améliorer les réponses du chatbot en combinant génération de texte et recherche dans des documents préexistants, une approche adaptée au besoin de DevPro d'analyser des documents pédagogiques et administratifs.



Sierre, le 21.10.24

Licence de MindsDB

MindsDB utilise deux types de licences :

- Elastic License 2.0 : Cette licence permet une utilisation gratuite de la plateforme à condition que le logiciel ne soit pas fourni sous forme de service hébergé. Cela correspond parfaitement au projet, car la solution sera auto-hébergée sur les serveurs de la HES-SO.
- MIT License : Utilisée pour certaines intégrations, cette licence libre permet une large utilisation, modification, et distribution du logiciel, compatible avec les exigences open-source du projet.

Next.js, qu'est-ce que c'est et pourquoi cette solution ?

Next.js est un framework pour React qui permet de créer des applications web modernes avec un rendu côté serveur (SSR), idéal pour les performances et le SEO. Il offre une grande flexibilité, notamment en ce qui concerne l'intégration avec des API, la gestion des données en temps réel, et l'optimisation des interfaces utilisateurs. Il est également open-source, et peut facilement s'intégrer à MindsDB via des requêtes API pour afficher les réponses du chatbot en direct.

Pourquoi Next.js ?

- Rendu côté serveur : Améliore les performances, essentiel pour une navigation fluide sur le site DevPro.
- Facilité d'intégration : Compatible avec les API de MindsDB et permet d'ajuster rapidement l'interface en fonction des retours utilisateur·rices.
- Modularité : Le framework permet de gérer différents composants du site web tout en assurant une expérience utilisateur optimale.

System Design : Backend + Frontend + RAG/Embeddings

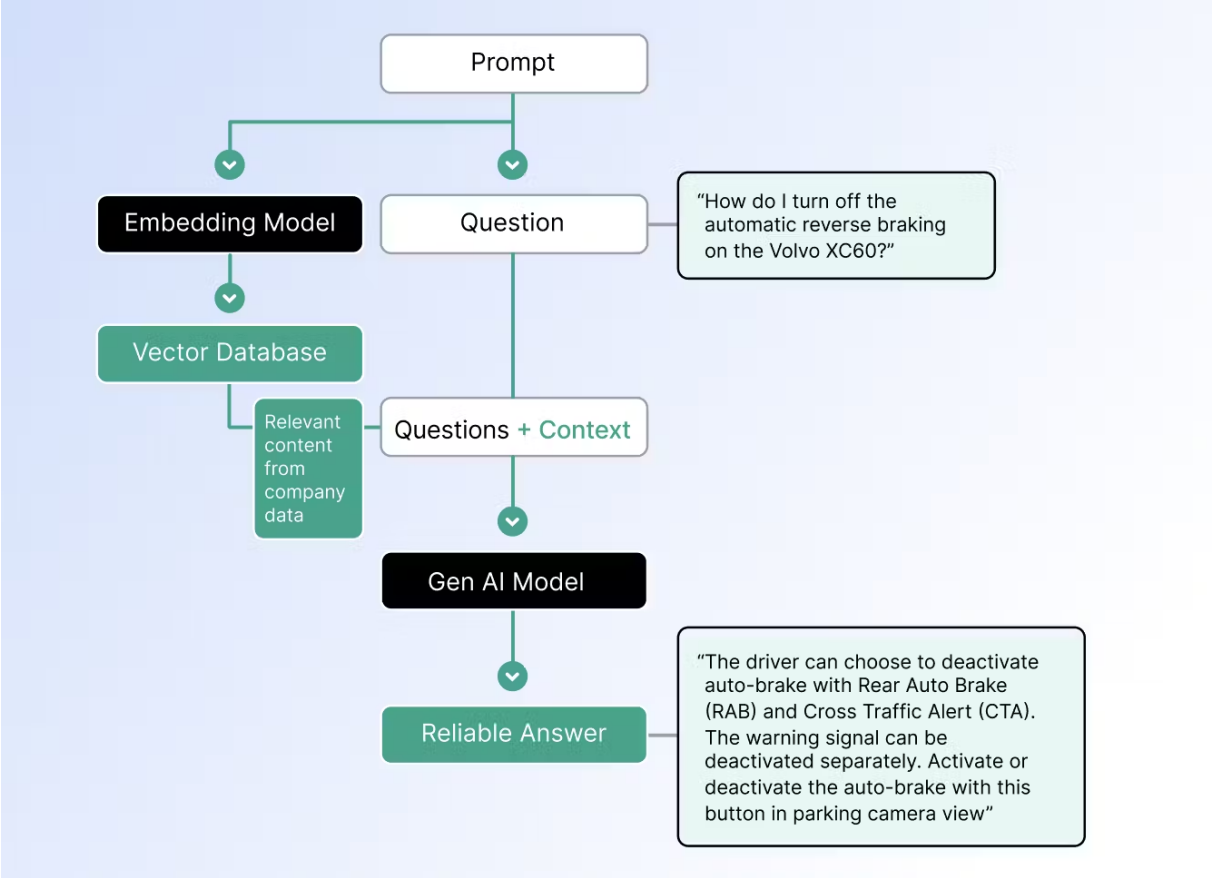
Architecture du système

Le système proposé repose sur une architecture à deux niveaux :

1. **Frontend (Next.js)** : Gère l'interface utilisateur, notamment le formulaire de questions pour le chatbot. Il est responsable de l'affichage des réponses du chatbot et de la navigation fluide entre les différentes pages liées.
2. **Backend (MindsDB)** : C'est ici que se situent les principaux traitements d'IA. Le backend intègre un modèle RAG qui permet d'extraire des informations

pertinentes à partir des documents de la HES-SO (comme les conditions d'obtention des attestations). Le backend répond aux questions des utilisateur·rices en effectuant des recherches dans la base de données et en générant une réponse adéquate.

Voici le processus de RAG/Embeddings



Présentation du POC Mindsdb

Le PoC contient une marche à suivre explicative des différentes étapes du processus et de l'interface, il suffit de suivre l'introduction intéractive.