Conception détaillée

ASADI : ASsistant ADministratif Intelligent

Groupe:
L3W1

Encadrant:
David Janisek
Auteurs:
Djaffer Abdel Malik, Gourmelen Thomas, Ponnoussamy Valentin, Traore Ali

Version du document:
1.0

Résumé
Dans le cadre de l'UE Projet Informatique à l'université Paris Cité, nous devons par équipes de quatre tenir un projet tout le long du semestre 6 sous la tutelle d'un encadrant. Ce projet vise à développer un assistant numérique capable de répondre automatiquement aux questions administratives, d'expliquer des procédures et d'offrir un apprentissage interactif via des scénarios immersifs, en s'appuyant sur une base documentaire centralisée.

Sommaire

1 Introduction

1.1 Contexte

Notre assistant virtuel, construit sur la technologie RAG, a été élaboré dans le but de rendre les démarches administratives plus simples et accessibles. En raison de la complexité des procédures et du grand nombre de documents administratifs, il est important de fournir une solution intelligente et interactive qui puisse aider les utilisateurs de manière efficace dans leur quête d'informations et leur processus d'apprentissage.

L'objectif de la conception détaillée est de fournir les détails nécessaires à la réalisation technique et fonctionnel de notre projet. Pour cela, elle se base sur les fonctionnalités mentionnées dans le cahier de recette ainsi que les besoins du cahier des charges, en les détaillants et en expliquant leurs solutions techniques afin de clarifier la structure du projet et d'assurer la cohérence entre les besoins et le développement du projet.

En plus de simplifier la phase de développement, cela permettra également la planification du projet notamment grâce à un diagramme de Gantt hiérarchisant les fonctionnalités selon leurs priorités de développement. Ainsi grâce à ce document, chaque membre de l'équipe pourra comprendre et structurer son travail au mieux.

Enfin, la conception détaillée permettra également de retrouver et corriger plus facilement l'origine des problèmes rencontrés durant le développement de notre application.

1.2 Objectif

L'objectif principal de ce projet est de développer une application web intégrant un agent intelligent permettant d'apporter des réponses claires aux questions administratives en s'appuyant sur une base documentaire fiable. D'expliquer de manière détaillée et structurée les procédures réglementaires, en guidant les utilisateurs étape par étape. Ainsi qu'intégrer un mode d'apprentissage interactif, avec des quiz et des mises en situation pour renforcer la compréhension des règles administratives.

Afin d'atteindre ces objectifs, plusieurs sous-objectifs sont définis :

1.2.a Intégration d'un système RAG

Une des grandes difficultés du projet réside dans la garantie que l'assistant offre des réponses adéquates et argumentées. Nous mettons en place un système RAG pour extraire des documents appropriés avant de produire une réponse.

- Amélioration de la précision des réponses grâce à une recherche documentaire avancée.
- Citations des sources pour garantir la transparence des informations fournies.
- Mise à jour dynamique de la base de connaissances pour suivre l'évolution des réglementations.

1.2.b Mise en place d'une base de données vectorielle

L'assistant s'aide d'une base de connaissances structurée afin de permettre une recherche rapide et efficace des documents administratifs.

- Utilisation d'une base de données vectorielle pour indexer les documents sous forme d'Embeddings.
- Mise à jour dynamique de la base de connaissances pour suivre l'évolution des réglementations.

1.2.c Développement d'une interface utilisateur ergonomique

Un assistant efficace doit être simple à utiliser, même pour des personnes non techniques. L'interface web devra :

- Être intuitive et accessible pour garantir une navigation fluide et une bonne expérience utilisateur.
- Faciliter l'accès aux fonctionnalités essentielles (recherche d'informations, ajout de documents, accès aux quiz et scénarios).

1.2.d Gestion sécurisée des documents administratifs

L'accès aux documents doit être contrôlé et sécurisé afin d'éviter toute modification non autorisée.

- Gestion des rôles :
  - Les administrateurs peuvent ajouter et supprimer des documents.
  - Les utilisateurs ont la possibilité de visualiser les documents, mais ne peuvent pas les modifier.

1.2.e Espace de travail

Afin d'optimiser les réponses du RAG lors d'une requête d'un utilisateur, l'assistant permettra à un administrateur d'établir des espaces de travail fonctionnant comme une bibliothèque thématique. Pour cela il faudra :

- Catégoriser les ressources en fonction d'un domaine de travail spécifique (ex : RH, comptabilité, etc.).
- Facilité l'accès à ces documents, le RAG devra éviter de se baser sur des documents ne faisant pas partie de l'espace en question.
- Seul les administrateurs ont la possibilité de de créer, supprimer ou modifier un espace de travail.

1.2.f Mode d'apprentissage interactif et pédagogique

L'assistant ne se contentera pas de consulter des documents, il offrira aussi une méthode d'apprentissage interactive destinée à améliorer la compréhension des utilisateurs grâce à :

- La mise en place de quiz permettant aux utilisateurs d'évaluer leur compréhension des procédures.
- L'élaboration de scénarios administratifs, dans lesquels les utilisateurs devront résoudre des situations en mettant en pratique leurs connaissances.
- Un retour sur mesure basé, sur les réponses fournies, pour favoriser leur apprentissage.

1.2.g Garantir un assistant fiable et évolutif

Pour assurer la fiabilité du projet dans le temps, l'architecture du système doit être modulaire et évolutive.

- Développement d'un code modulaire facilitant la maintenance et l'ajout de nouvelles fonctionnalités.
- Utilisation de technologies open-source tel que Python, Django, PostgrelSQL et ChromaDB.
- Déploiement sécurisé n'affectant pas la performance de l'application web ni l'expérience utilisateur.

L'ensemble de ces objectifs guideront l'ensemble du processus de développement du projet, de la conception technique à la mise en production. En combinant RAG, base vectorielle, interface ergonomique et mode d'apprentissage interactif, garantissant un assistant numérique performant, fiable et sécurisé.

2 Logiciel

Nom	Nature	Utilité	Limites
LlamaIndex	Bibliothèque	Permet d'intégrer des sources de données externes à des modèles de génération (RAG). Améliore la recherche d'information et la génération de texte.	Certains documents, comme les images et les PDFs scannés, sont mal extraits.
Pdf2image	Bibliothèque	Convertis des fichiers PDF en images (une page par image).	Ne gère pas les PDF contenant uniquement du texte sans images ; qualité de conversion dépend de la qualité du PDF.
Pytesseract	Bibliothèque	Utilise l'OCR (Reconnaissance Optique de Caractères) pour extraire du texte à partir d'images.	La précision de l'OCR peut être faible sur des images de mauvaise qualité ou avec des polices difficiles à lire.
PostgrelSQL	Système de gestion de bases de données (SGBD)	Base de données relationnelle utilisée pour stocker, organiser et gérer de grandes quantités de données.	Un inconvénient de PostgreSQL est sa consommation mémoire plus élevée par rapport à d'autres SGBD comme MySQL.
Python	Langage de programmation	Langage polyvalent utilisé pour le développement d'applications et de systèmes d'IA.	Peut être moins performant que d'autres langages pour des applications à très haute performance (par exemple, en calcul scientifique intensif).
Ollama	Plateforme/outil d'IA	Permet de déployer des modèles de langage (LLM) localement, avec des outils pour intégrer l'IA dans des applications sans dépendance cloud.	Nécessite une gestion des ressources locales, ce qui peut être limité par la capacité matérielle des systèmes.
Django	Framework	Framework Python pour le développement rapide d'applications web.	Peut être complexe pour les petits projets et nécessite des configurations pour certains environnements.
Django ORM	Technique	Technique permettant la manipulation de base de données relationnelle en utilisant des objets sans écrire de requêtes SQL.	Cette technique ne permet pas de faire des requêtes complexes.

3 Contraintes

Dans le cadre du développement de notre assistant numérique basé sur la technologie RAG, plusieurs contraintes doivent être prises en compte afin d'assurer la faisabilité et la viabilité du projet.

Le projet étant réalisé dans un cadre étudiant sans budget alloué, l'utilisation exclusive de technologies gratuites et open-source est impérative. Ainsi, les solutions cloud, les API payantes et les infrastructures coûteuses sont exclues, et les modèles d'IA utilisés seront issus de l'open-source afin d'éviter toute dépense.

Le développement devra être réalisé dans un délai imparti de 12 semaines, avec une livraison prévue la semaine du 28 avril 2025. Ce planning inclut plusieurs étapes clés :

- La semaine du 24 février : fin de la conception et remise de la documentation.
- La semaine du 7 avril : dernière phase de développement, incluant l'analyse des ressources, la mise en place de l'assistant et du mode interactif, ainsi que les tests et optimisations.
- La semaine du 21 avril : finalisation de l'intégration.
- La semaine du 28 avril : préparation de la soutenance, incluant un rapport de projet, un wiki et des diapositives.

L'absence de budget impose l'utilisation de matériel personnel pour le développement, nécessitant :

- Des ordinateurs adaptés à l'exécution locale du projet et à l'IDE choisi.
- Une connexion Internet stable, essentielle pour accéder aux API et aux outils collaboratifs.
- Une base de données vectorielle open-source, permettant le stockage et la recherche efficace des documents administratifs indexés et des données générées.

Le projet devra répondre à plusieurs exigences en matière de qualité, et d'ergonomie :

- Structuration et qualité des données : Les documents doivent être clairs, organisés et exploitables par l'IA pour garantir des réponses pertinentes.
- Modularité du code : Le développement devra être modulaire, facilitant ainsi la maintenance, la réutilisation et l'ajout de nouvelles fonctionnalités sans impacter l'ensemble du projet.
- Ergonomie: L'interface utilisateur devra être intuitive et fluide.
- Conformité de l'application web : L'application devra respecter les fonctionnalités définies, offrir une expérience utilisateur optimale, et garantir performance et fiabilité. Des tests et validations seront menés pour s'assurer que l'application répond aux attentes.

En résumé, ces contraintes et une gestion agile avec des sprints courts et des objectifs intermédiaires assureront un produit fonctionnel, optimisé et aligné avec les ressources disponibles et les exigences techniques du projet tout en garantissant le respect des échéances.

4 Description des modules

Tout au long de cette section, nous ferons référence aux différentes pages de la maquette de l'application web, qui sont situées en annexe. Nous les mentionnerons en indiquant le numéro de l'annexe ainsi que le nom de la page.

- « ⇒ » : Indique la sous partie concernée.
- « → » : Indique l'annexe concernée.

4.1 Utilisateur

4.1.a Page de connexion :

- **Connexion** :
  - **Objectif** : permet à l'utilisateur de se connecter pour accéder à l'application.
  - **Technologies utilisées** : **Django ORM**, **PostgreSQL**.
  - **Table concernée** : **Utilisateur**.
  - **Éléments présents** :
    - Deux champs de texte :
      - Champ pour l'**identifiant**.
      - Champ pour le **mot de passe**.
    - Deux boutons :
      - Bouton de validation de la **connexion**.
      - Bouton de redirection vers la page d'**inscription**.
  - **Comportement** :
    - Une fois que l'utilisateur renseigne ses identifiants valides, il est dirigé vers ⇒ **4.1.c.Page principal**.
    - Si l'utilisateur ne dispose pas de compte, il est obligé pour lui de créer un compte et il sera dirigé vers ⇒ **4.1.b.Page d'inscription**.
    - L'authentification est vérifiée via **Django ORM** et **PostgreSQL**.
    - En cas d'échec, un message d'erreur est affiché.
  - **Maquette en annexe** → A1: Page de login.

4.1.b Page d'inscription :

- **Inscription** :
  - **Objectif** : Permet à l'utilisateur de créer un compte, et que le compte soit stocké dans la base base de données **PostgreSQL** via **Django ORM**.
  - **Technologies utilisées** : **Django ORM**, **PostgreSQL**.
  - **Table concernée** : **Utilisateur**.
  - **Éléments présents** :
    - Quatre champs de texte :
      - Champs pour le **prénom**.
      - Champs pour le **nom**.
      - Champs pour l'**adresse e-mail**.
      - Champs pour le **mot de passe**.
    - Deux boutons :
      - Bouton de validation de l'**inscription**.
      - Bouton de **retour en arrière**.
  - **Comportements** :
    - Une fois validé, le compte de l'utilisateur est enregistré dans la base de données.
    - Une fois le formulaire complété et validé, l'utilisateur sera redirigé vers ⇒ **4.1.c.Page principale**.
    - Si le compte est déjà existant, l'utilisateur sera dirigé vers ⇒ **4.1.a.Page de connexion**.
  - **Maquette en annexe** → A2: Page d'inscription.

4.1.c Page principale :

Cette page est la page principale où l'utilisateur pourra interagir avec l'assistant administratif. Il pourra également tester ses connaissances administratives en générant des quiz ainsi qu'en réalisant des scénarios.

- **Page principale - partie Prompt** :
  - **Objectif** : cette section permet à l'utilisateur de rédiger un prompt et d'obtenir une réponse générée par le LLM et basée sur le RAG.
  - **Technologies utilisées** : **Django ORM**, **PostgreSQL**, **ChromaDB**, **Ollama**, **LlamaIndex**.
  - **Table concernée** : **Historique**.
  - **Éléments présents** :
    - Un champ de texte pour la rédaction du **prompt**.
    - Un bouton d'**envoi du prompt**.
    - Une zone de **réponse**.
    - Un bouton d'accès aux différents **espaces de travail**.
  - **Comportements** :
    - Une fois que l'utilisateur soumet son prompt à l'assistant, ce dernier génère une réponse en utilisant la technologie RAG pour fournir une réponse pertinente.
    - Si l'utilisateur appuie sur le bouton « Espaces de travail », un pop-up s'affiche lui permettant de choisir un espace de travail ⇒ **4.1.g.Espace de travail**.
    - L'historique de la discussion (prompts & réponses) est stocké dans la base de données.
    - LlamaIndex recherche, dans la base de données vectorielle, des informations pertinentes à la demande.
    - La réponse est générée via le LLM.
  - **Maquette en annexe** → A3 : Page de prompt.

- **Page principale - partie Quiz** :
  - **Objectif** : Cette section permet à l'utilisateur de démarrer, créer ou supprimer un quiz.
  - **Technologies utilisées** : **Django**, **PostgreSQL**.
  - **Tables concernées** : **Quiz**, **QuestionQuiz**.
  - **Éléments présents** :
    - Trois boutons :
      - Bouton de **sélection & démarrage** d'un quiz.
      - Bouton de **création** d'un quiz.
      - Bouton de **suppression** d'un quiz.
    - Une liste de quiz.
  - **Comportements** :
    - Lorsque l'utilisateur sélectionne un quiz, il sera redirigé vers ⇒ **4.1.d.Page démarrage d'un quiz**.
    - Lorsque l'utilisateur clique sur le bouton de création de quiz, il sera redirigé vers ⇒ **4.1.e. Page création de quiz**.
    - Lorsque l'utilisateur clique sur le bouton de suppression, le quiz est supprimé de la base de donnée et de l'interface utilisateur.

- **Page principale - partie Scénario** :
  - **Objectif** : Cette section permet à l'utilisateur de démarrer un scénario.
  - **Technologies utilisées** : **Django**, **PostgreSQL**.
  - **Table concernée** : **Scenario**, **QuestionScenario**.
  - **Éléments présents** :
    - Une liste de scénarios.
  - **Comportements** :
    - Lorsque l'utilisateur sélectionne un scénario, il sera redirigé vers ⇒ **4.1.f.Page démarrage d'un scénario**.

- **Page principale - partie Historique** :
  - **Objectif**: cette section permet à l'utilisateur d'avoir accès aux anciennes discussions avec l'assistant.
  - **Technologies utilisées** : **Django**, **PostgreSQL**.
  - **Table concernée** : **Historique**.
  - **Éléments présents** :
    - Une liste de discussion.
    - Un bouton de **suppression** pour chaque discussion.
  - **Comportement** :
    - Les discussions sont récupérées depuis la table « Historique » présente dans la base de données.
    - Visualisation des anciennes discussions sous forme de liste.
    - Sélection d'une discussion, dans la liste d'historiques de discussion, permettant de continuer cette dernière ⇒ **4.1.c. Page principale - Partie Prompt**.
    - Lorsque l'utilisateur clique sur le bouton de suppression, la discussion est supprimée de la base de données et de l'interface utilisateur.

- **Page principale - partie Profil** :
  - **Objectif** : cette section permet à l'utilisateur d'accéder à son profil utilisateur.
  - **Technologies utilisées** : **Django**, **PostgreSQL**.
  - **Table concernée** : **Utilisateur**.
  - **Éléments présents** :
    - Un bouton permettant l'**accès à la page du profil** utilisateur.
    - Une **barre de niveau**.
  - **Comportement** :
    - Si l'utilisateur appuie sur le bouton **Profil utilisateur**, il sera redirigé vers ⇒ **4.1.h.Page profil utilisateur**.
    - L'utilisateur pourra suivre l'évolution de son niveau.
    - Si l'utilisateur appuie sur le bouton **Déconnexion**, ce dernier est déconnecté et redirigé vers ⇒ **4.1.a.Page de connexion**. La déconnexion implique la suppression de la session de l'utilisateur.

4.1.d Page démarrage d'un quiz :

- **Objectif** : cette section permet à l'utilisateur de démarrer un quiz afin qu'il puisse s'entraîner et améliorer ses connaissance.
- **Technologies utilisées** : **Django**, **PostgreSQL**, **ChromaDB** , **Ollama**, **LlamaIndex**.
- **Tables concernées** : **Quiz**, **QuestionQuiz**.
- **Éléments présents** :
  - Un champ de texte pour **saisir le prompt** de réponse.
  - Un bouton d'**envoi du prompt** de réponse.
  - Une zone de texte permettant d'**afficher la correction** de la réponse donnée.
- **Comportement** :
  - L'assistant sélectionne une question depuis la table « Question » et l'affiche, dans la zone de texte.
  - L'utilisateur saisit et envoie sa réponse.
  - L'assistant compare la réponse avec la réponse prévue, dans la base de données, et fournit un retour détaillé.
  - L'historique des questions/réponses est enregistré au sein du quiz.

4.1.e Page création d'un quiz :

- **Objectif** : cette section permet à l'utilisateur de créer un nouveau quiz.
- **Technologies utilisées** : **Django**, **PostgreSQL**, **ChromaDB**, **Ollama**, **LlamaIndex**.
- **Tables concernées** : **Quiz**, **Question**.
- **Éléments présents** :
  - Un champ de texte pour **rédiger le prompt** servant à donner un thème au quiz.
  - Un bouton d'**envoi du prompt**.
  - Une **zone de réponse** pour l'assistant.
- **Comportements** :
  - L'utilisateur spécifie le domaine ou le sujet sur lequel il souhaite créer un quiz.
  - L'assistant analyse la requête pour créer des questions réponses en lien avec le domaine ou sujet abordé dans la requête.
  - Le quiz généré est accessible dans la section Quiz de la ⇒ **4.1.c.Page Principale - Quiz**.

4.1.f Page démarrage d'un scénario :

- **Objectif** : cette section permet à l'utilisateur de se confronter à un scénario.
- **Technologies utilisées** : **Django**, **PostgreSQL**, **ChromaDB**, **Ollama**, **LlamaIndex**.
- **Table concernée** : **Scenario**.
- **Éléments présents** :
  - Un champ de texte pour que l'utilisateur puisse **entrer une réponse** à la question posée par le scénario.
  - Un bouton pour **envoyer le prompt** à l'assistant.
  - Une **zone de discussion** pour que l'assistant puisse poser les questions, composant le scénario, à l'utilisateur et affichant les réponses détaillées.
- **Comportements** :
  - Sélection d'un scénario depuis la liste des scénarios.
  - L'assistant compare la réponse avec la base de données et fournit un retour détaillé.
  - Si la réponse est incorrecte, l'assistant fournit une correction détaillée avec explications et sources.

4.1.g Fenêtre contextuelle de l'espace de travail

- **Objectif** : fournir un environnement permettant aux utilisateurs de limiter leur recherches au sein du RAG à certains domaines donnés RAG.
- **Technologies utilisées** : **Django**, **PostgreSQL**, **ChromaDB**, **Ollama**, **LlamaIndex**.
- **Tables concernées** : **Workspace**, **Document**.
- **Éléments présents** :
  - Une liste d'**espaces de travail**.
  - Un bouton permettant de **fermer la fenêtre**.
- **Comportements** :
  - Les réponses fournies par l'assistant sont plus précises et délivrées plus rapidement que sans préciser l'espace de travail.
  - Les interactions effectuées dans chaque espace de travail sont enregistrées dans l'Historique.
- **Maquette en annexe** → A4 : Page d'espace de travail.

4.1.h Page profil utilisateur :

- **Objectif** : cette section permet à l'utilisateur de voir ses informations personnelles et de se déconnecter.
- **Technologies utilisées** : **Django**, **PostgreSQL**.
- **Table concernée** : **Utilisateur**.
- **Éléments présents** :
  - **Prénom**.
  - **Nom**.
  - **Email**.
  - **Rôle**
  - **Niveau du profil**.
  - Un bouton de **déconnexion**.
  - Un bouton de **retour en arrière**.
- **Comportements** :
  - L'utilisateur aura accès à ses informations personnelles.
  - En appuyant sur le bouton de déconnexion, l'utilisateur est déconnecté et redirigé vers ⇒ **4.1.a.Page de connexion**

4.2 Admin

4.2.a Page d'accueil :

- **Objectif** : cette section permet à l'administrateur de choisir une des quatre fonctionnalités de l'administrateur.
- **Technologies utilisées** : **Django**.
- **Éléments présents** :
  - Quatre boutons :
    - Bouton **Liste des utilisateurs**.
    - Bouton **Fichiers de l'application**.
    - Bouton **Scénario**.
    - Bouton **Prompt**.
- **Comportements** :
  - Si l'administrateur clique sur le bouton « Liste des utilisateurs », il sera redirigé vers ⇒ **4.2.c.Page liste des utilisateurs**.
  - Si l'administrateur clique sur le bouton « Fichiers de l'application », il sera redirigé vers ⇒ **4.2.d.Page Fichier**.
  - Si l'administrateur clique sur le bouton « Prompts », il sera redirigé vers ⇒**4.1.c.Prompt**.
  - Si l'administrateur clique sur le bouton « Scénario », il sera redirigé vers ⇒ **4.2.f.Page de Scénario**.
- **Maquette en annexe** → A5: Page de menu.

4.2.b Page Liste des utilisateurs :

- **Objectif** : cette section permet à l'administrateur de gérer les utilisateurs de l'application.
- **Technologies utilisées** : **Django**, **PostgreSQL**.
- **Table concernée** : **Utilisateur**.
- **Éléments présents** :
  - Une **liste d'utilisateurs**.
  - Un bouton de **suppression** par utilisateur.
  - Un bouton de **retour en arrière**.
- **Comportements** :
  - Visualisation des utilisateurs de l'application.
  - Suppression d'utilisateurs.
- **Maquette en annexe** → A7: Page contenant la liste des utilisateurs.

4.2.c Page Fichier :

- **Objectif** : cette section permet à l'administrateur de gérer les fichiers. Ces fichiers seront utilisés par l'assistant.
- **Technologies utilisées** : **Django**, **PostgreSQL**, **ChromaDB**.
- **Tables concernées** : **Document**, **Workspace**.
- **Éléments présents** :
  - Une zone de **dépôt de fichiers**.
  - Une **liste des fichiers**.
  - Deux boutons :
    - Bouton de **suppression** par fichier.
    - Bouton de **retour en arrière**.
- **Comportements** :
  - Visualisation des fichiers.
  - Ajout de fichiers pour enrichir la base de données. Lors de l'ajout, l'administrateur peut préciser l'espace de travail voir ⇒ **4.2.e.Espace de travail**.
  - Suppression de fichiers en cliquant sur le bouton de suppression.
- **Maquette en annexe** → A6 : Page de chargement des documents.

4.2.d Espace de travail :

- **Objectif** : cette section permet à l'administrateur de gérer les espaces de travail.
- **Technologies utilisées** : **Django**, **PostgreSQL**.
- **Tables concernées** : **Workspace**.
  - **Éléments présents** :
    - Une **liste des espaces de travail**.
    - Trois boutons :
      - Bouton d'**ajout** d'espace de travail.
      - Bouton de **suppression** par espace de travail.
      - Bouton de **retour en arrière**.
  - **Comportements** :
    - Visualisation des espaces de travail.
    - Ajout d'un espace de travail.
    - Suppression d'un espace de travail en cliquant sur le bouton de suppression.

4.2.e Page de Scénario :

- **Objectif** : Cette section permet à l'administrateur de gérer les scénarios.
- **Technologies utilisées** : **Django**, **PostgreSQL**.
- **Tables concernées** : **Scenario**.
- **Éléments présents** :
  - Une liste des scénarios
  - Deux champs :
    - Un champ pour **saisir une question**.
    - Un champ pour **saisir une réponse**.
  - Trois boutons :
    - Bouton **ajout d'un scénario**.
    - Bouton **suppression d'un scénario**.
    - Bouton de **retour en arrière**.
- **Comportements** :
  - Visualiser des scénarios.
  - Ajout d'un scénario après validation du formulaire dédié.
  - Suppression d'un scénario en cliquant sur le bouton de suppression.
- **Maquette en annexe** → A8 : Page des scénarios.

4.2.f Page de prompt :

Cette page est la même page que la page principale. ⇒ **4.1.c.Page Principale - partie Prompt**.

4.2.g Page de Profil :

Cette page est la même page que la page principale. ⇒ **Page principale - partie Profil**.

5 Implémentation

5.1 Cas d'utilisation

Les cas d'utilisation servent à représenter les actions concrètes des utilisateurs sur le projet et interagissant avec le système. Ils permettent de formaliser les exigences fonctionnelles, d'identifier les interactions entre l'utilisateur et le système, et de guider le développement. Ces schémas aident à anticiper les scénarios possibles, y compris les erreurs ou exceptions.

5.1.a User case utilisateur

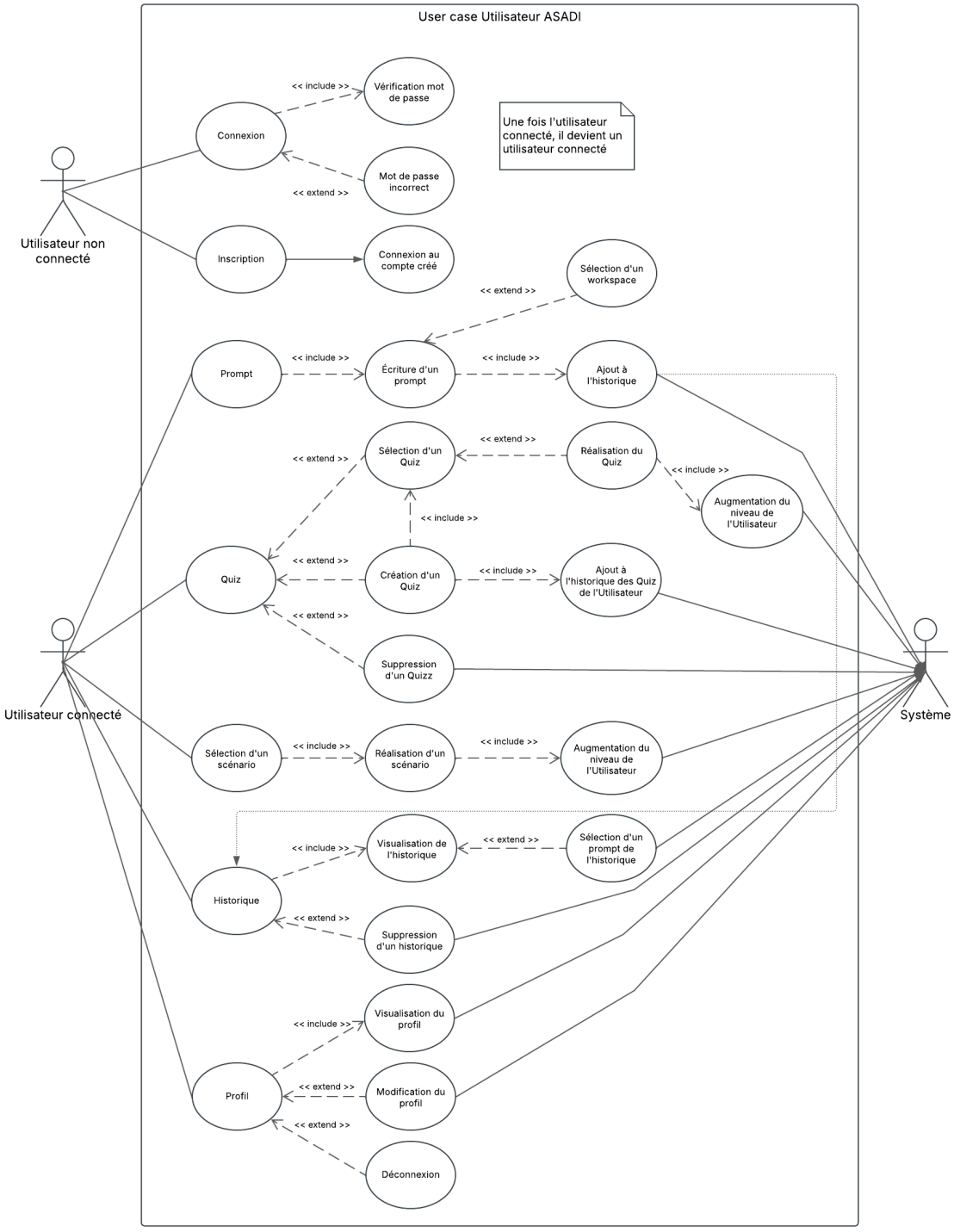


Fig. 1. – Case d'utilisation d'un utilisateur

5.1.b User case administrateur

Fig. 2. – Case d'utilisation d'un administrateur

5.2 Base de données

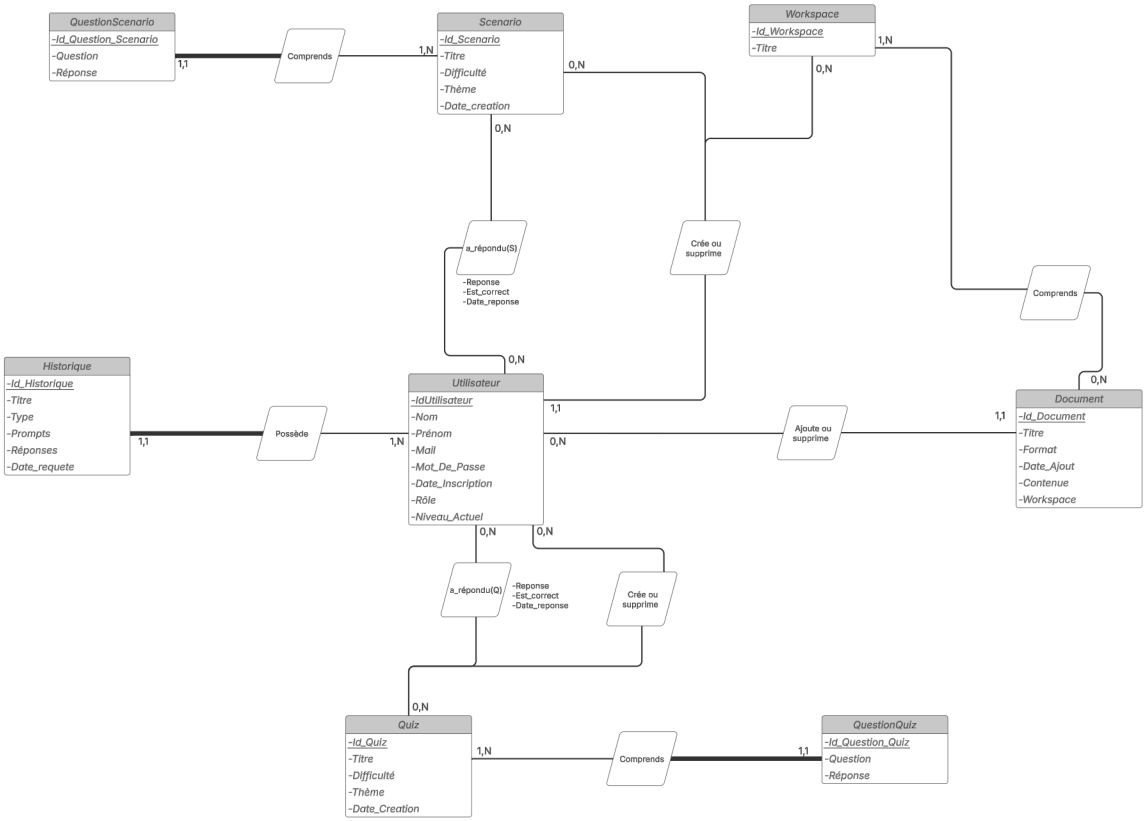


Fig. 3. – Schéma de base de données

5.3 Diagramme de classes

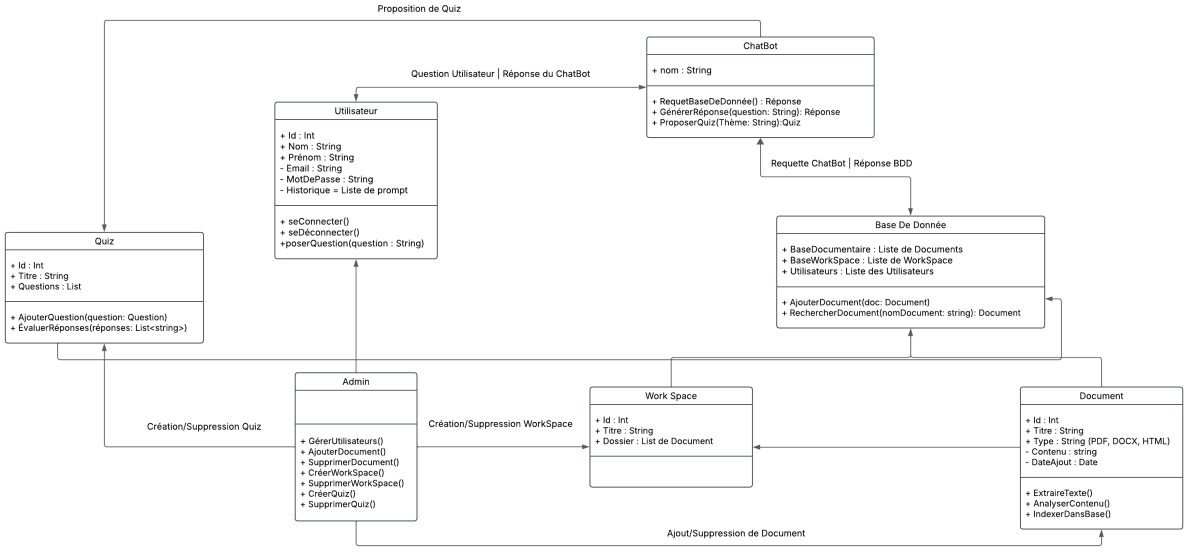


Fig. 4. – Diagramme de classes

6 Organisation

N°	Tâche	Début	Fin
FU1	**Frontend** : Développement de la page de connexion & d'inscription.	3 mars	10 mars
FU2	**Frontend** : Développement de la page des prompts.	3 mars	10 mars
FU3	**Frontend** : Développement de la page profil.	3 mars	10 mars
FA1	**Frontend** : Développement de la page de création de scénario par l'administrateur.	10 mars	17 mars
FA2	**Frontend** : Développement de la page d'insertion de documents.	10 mars	17 mars
FA3	**Frontend** : Développement de la page récapitulative des différents utilisateurs de l'application.	10 mars	17 mars
FA4	**Frontend** : Développement du menu de l'administrateur.	14 mars	28 mars
F	**Frontend** : Implémentation de l'interaction entre les pages de l'application.	28 mars	7 avril
B1	**Backend** : Intégration des fonctionnalités permettant l'extraction de contenu à partir de différents types de documents.	3 mars	10 mars
B2	**Backend** : Intégration d'un modèle de langage (LLM) et d'un embedding NLP pour le traitement et l'analyse des phrases.	3 mars	10 mars
B3	**Backend** : Développement de la structure de la base de données.	10 mars	17 mars
B4	**Backend** : Développement du système de création/connexion d'un compte.	10 mars	17 mars
B5	**Backend** : Développement du système de rôle utilisateur/ administrateur.	17 mars	24 mars
B6	**Backend** : Implémentation de la fonctionnalité d'ajout de documents par un administrateur.	24 mars	1 avril
B7	**Backend** : Mise en place du système de création de scénarios par un administrateur.	1 avril	7 avril
B8	**Backend** : Développement de la fonctionnalité de génération de quiz.	7 avril	14 avril
FB1	**Frontend + Backend** : Connexion entre le Frontend et le Backend pour les connexion et les inscriptions.	14 avril	21 avril
FB2	**Frontend + Backend** : Connexion entre le Frontend et le Backend pour la gestion des prompts.	14 avril	21 avril
FB3	**Frontend + Backend** : Connexion entre le Frontend et le Backend pour l'ajout de documents par un administrateur.	14 avril	21 avril
FB4	**Frontend + Backend** : Connexion entre le Frontend et le Backend pour l'ajout de scénarios par un administrateur.	14 avril	21 avril
D1	**Déploiement** : Déploiement de l'application sur un serveur.	21 avril	28 avril

7 Planification

7.1 Diagramme de Gantt

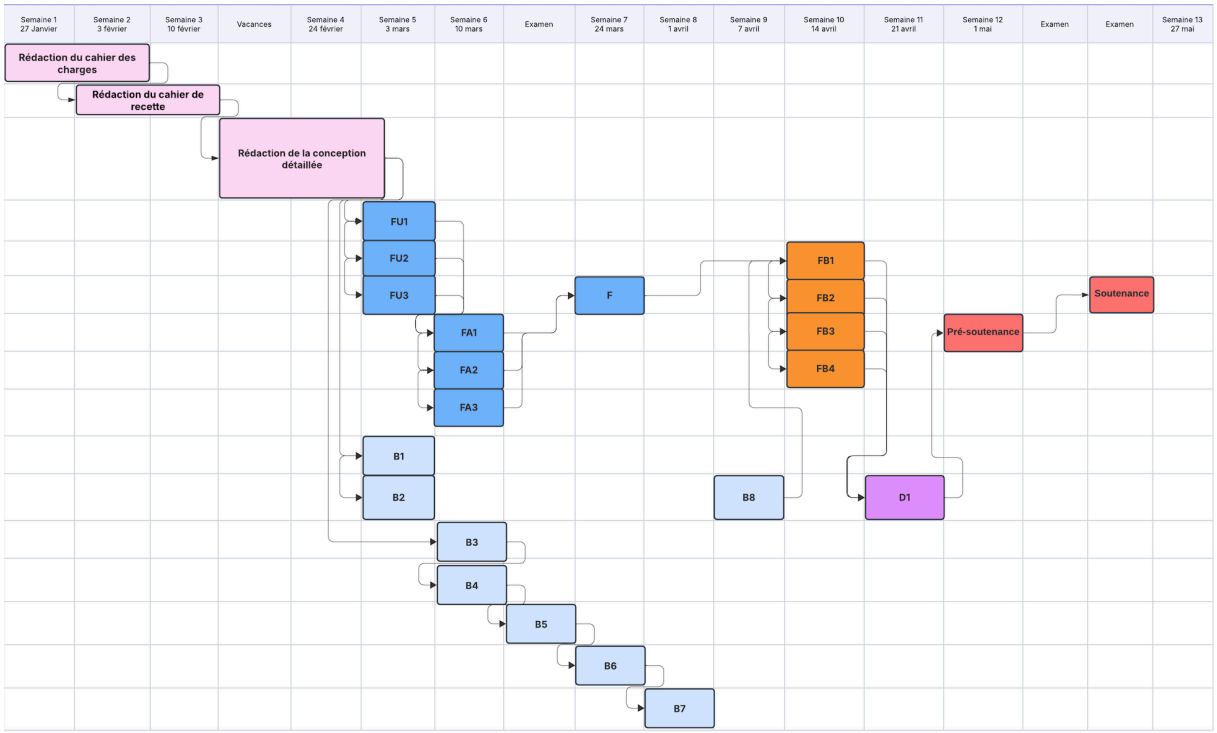
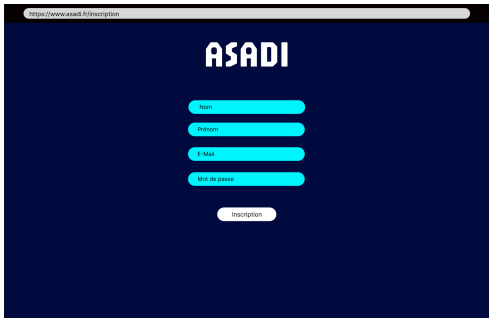


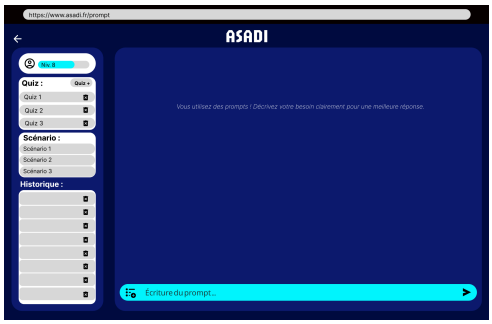
Fig. 5. – Diagramme de Gantt du projet

8 Annexes

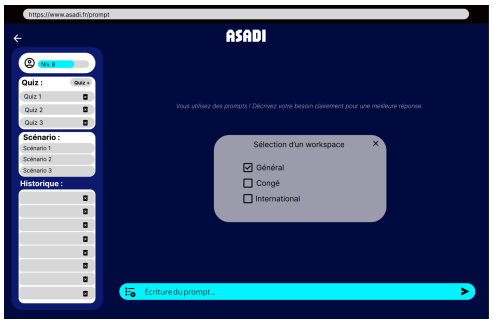
A1 : **Page de login** pour se connecter.



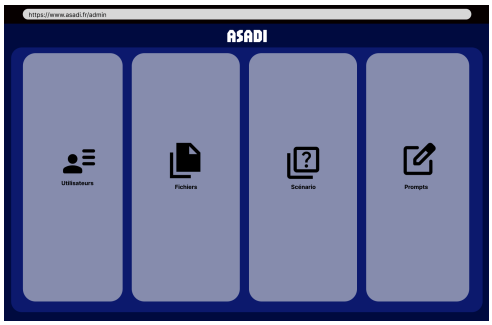
A2 : **Page d'inscription (Utilisateur)** pour créer un compte.



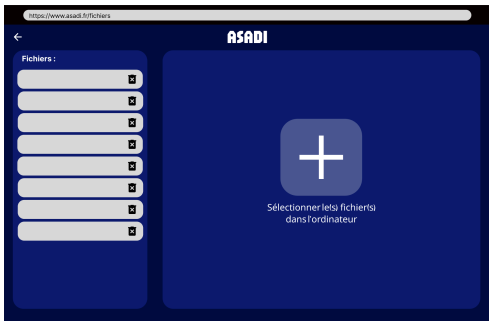
A3 : **Page de prompt (Utilisateur & Admin)** pour rédiger des prompts.



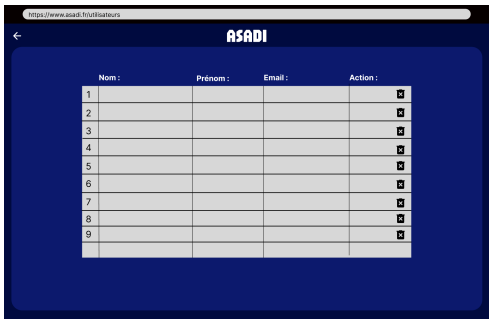
A4 : **Page d'espace de travail (Utilisateur & Admin)** pour spécifier son espace de travail.



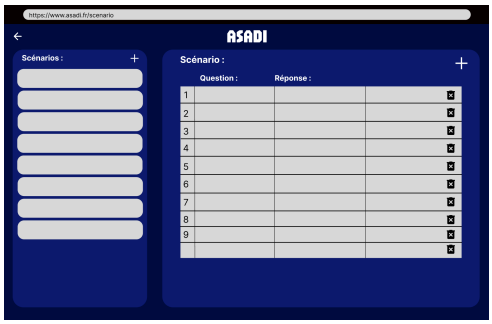
A5 : **Page de menu (Admin)** permettant de naviguer dans l'application.



A6 : **Page de chargement des documents (Admin)** pour déposer des fichiers administratifs.



A7 : **Page contenant la liste des utilisateurs (Admin)** pour visualiser les utilisateurs.



A8 : **Page des scénarios (Admin)** pour visualiser et modifier les scénarios.

9 Glossaire

Administrateur : Utilisateur ayant des droits spéciaux pour gérer la base de connaissances, ajouter/modifier/supprimer des documents et administrer l'application.

API (Application Programming Interface) : Interface permettant à différentes applications de communiquer entre elles.

Application web : Logiciel accessible via un navigateur web, permettant aux utilisateurs d'interagir avec l'assistant numérique.

Base de données : Système de stockage structuré permettant de conserver et d'organiser les informations de l'application.

Base de données vectorielle : Type de base de données optimisée pour stocker et rechercher des embeddings (vecteurs numériques) afin d'effectuer des recherches par similarité.

Backend : Partie du système qui gère la logique métier et les interactions avec la base de données, généralement exécutée sur un serveur.

ChromaDB : Base de données vectorielle open-source utilisée pour stocker et interroger des embeddings.

Cybersécurité : Ensemble des mesures visant à protéger les systèmes informatiques contre les attaques et les accès non autorisés.

Django : Framework Python permettant de développer des applications web rapidement et efficacement.

Documents administratifs : Ensemble des fichiers et textes réglementaires que l'application utilise pour générer ses réponses.

Développement : Processus de conception, de codage et de mise en œuvre du logiciel.

Embedding : Représentation numérique d'un texte ou d'un document sous forme de vecteur, permettant une recherche avancée par similarité.

Ergonomie : Qualité d'une interface permettant une utilisation fluide et intuitive.

Frontend : Partie visible de l'application, avec laquelle l'utilisateur interagit (interface utilisateur).

Framework : Ensemble d'outils et de bibliothèques facilitant le développement d'un logiciel.

IA (Intelligence Artificielle) : Technologie permettant à un programme informatique de réaliser des tâches habituellement effectuées par des humains, comme la compréhension du langage naturel.

Indexation : Processus de transformation des documents en vecteurs pour permettre une recherche rapide.

LlamaIndex : Outil permettant d'intégrer des sources de données externes à des modèles de génération d'IA pour améliorer la recherche d'information.

LLM (Large Language Model) : Modèle de langage de grande taille capable de traiter et générer du texte en fonction des entrées fournies par un utilisateur.

MySQL : Système de gestion de bases de données relationnelles utilisé pour stocker les informations structurées de l'application.

OCR (Optical Character Recognition) : Technologie permettant d'extraire du texte à partir d'images ou de documents scannés.

Ollama : Plateforme permettant d'exécuter des modèles d'IA localement, sans dépendre de services cloud.

Open-source : Logiciel dont le code source est libre d'accès et modifiable par la communauté.

PDF2Image : Bibliothèque permettant de convertir des fichiers PDF en images, utile pour l'extraction de texte via OCR.

Pytesseract : Outil d'OCR utilisé pour extraire du texte à partir d'images ou de PDF scannés.

Prompt : Instruction donnée à un modèle de langage pour générer une réponse ou une action.

Python :Langage de programmation utilisé pour le développement de l'application.

RAG (Retrieval-Augmented Generation) : Approche combinant la recherche d'informations et la génération de texte pour produire des réponses pertinentes basées sur une base documentaire.

Recherche de similarité : Processus permettant de comparer un vecteur donné (ex: une question) avec les vecteurs stockés dans la base de données pour trouver les résultats les plus proches.

Utilisateur : Personne interagissant avec l'application, soit pour poser des questions, soit pour gérer les données.

Vecteur : Représentation numérique utilisée pour la recherche de documents par similarité.

10 Références

- Pour la rédaction de cette documentation, nous avons utilisé le modèle d'intelligence artificielle ChatGPT. Ce modèle nous a aidés sur plusieurs points :
  - Reformulation de phrases et correction de fautes d'orthographe : Nous avons utilisé ChatGPT pour nous assister dans la rédaction de certaines phrases, afin d'améliorer la clarté du texte.
  - Recherche : ChatGPT nous a permis d'explorer différentes technologies que nous devrons utiliser dans ce projet. Il nous a également aidés à mieux comprendre certains concepts abstraits.

11 Index