



MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
DE LA RECHERCHE SCIENTIFIQUE ET DE L'INNOVATION
UNIVERSITÉ SULTAN MOULAY SLIMANE
ECOLE NATIONALE DES SCIENCES APPLIQUÉS
DE KHOURIBGA



Rapport du Projet

Filière : Informatique et Ingénierie des Données



Réalisé par :

Oumaima El alami

Salma Adsaoui

Oumayma Ennamous

Application de Recommandation des Offres de Stage utilisant les LLMs et un Chatbot

Encadré par :

Pr. Nidal Lamghari ENSA Khouribga

Année Académique : 2025

Sommaire

Résumé	ii
Remerciement	iii
Table des matières	iv
Liste des figures	viii
Introduction	1
1 Analyse des besoins et conception fonctionnelle	2
2 Système d'Authentification	13
3 Système de recommandation des offres de stage	20
4 Prédiction de la filière adaptée	34
5 Développement du chatbot intelligent	41
6 Technologies et Outils Utilisés	52
7 Interface graphique	60
Conclusion	79

Résumé

Ce projet tutoré s'inscrit dans une démarche d'innovation visant à accompagner les étudiants dans leur parcours académique et professionnel à travers le développement d'une application web intelligente. Exploitant les capacités des Large Language Models (LLMs), cette application propose un système de recommandation personnalisé des offres de stage, un module de prédition de la filière universitaire la plus adaptée, ainsi qu'un chatbot conversationnel intelligent.

L'objectif principal est d'offrir aux étudiants un outil d'aide à la décision, capable d'analyser leurs compétences, leurs préférences et leurs performances académiques afin de leur proposer des recommandations pertinentes et ciblées. Le chatbot intégré permet, quant à lui, d'enrichir l'expérience utilisateur en fournissant des informations sur les métiers, les compétences clés, la préparation aux entretiens, ainsi que sur les formations proposées par l'établissement.

Par ailleurs, l'application met à disposition une base de données de travaux académiques (rapports, projets, mémoires), classés par filière, année et entreprise, facilitant ainsi l'accès à des ressources utiles et inspirantes.

Ce projet combine des approches modernes en intelligence artificielle, en traitement du langage naturel et en développement web, dans le but de créer une plateforme complète et évolutive répondant aux besoins concrets des étudiants.

Mots clés :LLMs,module de prédition, système de recommandation , chatbot conversationnel intelligent.

Remerciement

Nous tenons à exprimer notre profonde gratitude à **Madame Lamghari Nidal** pour son encadrement exemplaire et son engagement constant tout au long de cette année universitaire.

Votre passion pour l'enseignement, votre disponibilité, ainsi que la rigueur dont vous avez fait preuve, ont été pour nous une véritable source d'inspiration. Vous avez su nous guider avec bienveillance et professionnalisme, en nous apportant des conseils précieux à chaque étape de notre projet.

Nous vous remercions sincèrement pour le temps que vous nous avez consacré, pour vos remarques constructives, et pour votre soutien sans faille, qui ont grandement contribué à la réussite de ce travail. Grâce à votre encadrement attentif, nous avons pu surmonter les difficultés rencontrées et mener ce projet à terme dans les meilleures conditions.

Nous sommes honorés d'avoir pu bénéficier de votre accompagnement, et nous vous en sommes profondément reconnaissants.

Table des matières

Résumé	ii
Remerciement	iii
Table des matières	iv
Liste des figures	viii
Introduction	1
1 Analyse des besoins et conception fonctionnelle	2
1.1 Diagramme des cas d'utilisation	2
1.2 Diagramme d'activités	6
1.3 Diagramme de classes	7
1.3.1 Description des classes	8
users	8
cv_analysis	8
user_preferences	8
internship_listings	8
user_recommendations	8
1.3.2 Relations entre les classes	8
1.4 Diagramme de Séquence	8
1.4.1 L'authentification des étudiants	8
1.4.2 Le système de recommandation des offres de stage	9
1.4.3 La prédiction de la filière adaptée	10
1.4.4 L'interaction avec un chatbot	11
1.5 Diagramme de Communication	11
1.5.1 Diagramme de Communication du systeme	11
1.5.2 Explication du Diagramme de Communication	12
2 Système d'Authentification	13
2.1 Technologies Utilisées	13
2.2 Processus d'Inscription (Registration)	14
2.3 Processus de Connexion (Login)	14
2.4 Accès aux Ressources Protégées	15
2.5 Gestion de la Sécurité	16
2.6 Sécurité des Routes et Gestion de l'Authentification	16
2.6.1 Gestion des Tokens dans les Routes API	16

2.6.2	Protection des Routes avec Middleware	17
2.6.3	Gestion du Contexte d'Authentification côté Client	17
2.6.4	Bonnes Pratiques et Limitations	17
2.7	Architecture de Sécurité du Système d'Authentification	18
3	Système de recommandation des offres de stage	20
3.1	Système de Recommandation Hybride	20
3.1.1	Filtrage Basé sur le Contenu (Content-Based Filtering)	21
3.1.1.1	Concept et Fonctionnement	21
3.1.1.2	Implémentation dans notre Système	21
3.1.1.3	Avantages et Limites du Filtrage Basé sur le Contenu	22
	Avantages	22
	Limites	22
3.1.2	Filtrage Collaboratif (Collaborative Filtering)	22
3.1.2.1	Concept et Fonctionnement	22
3.1.2.2	Implémentation dans notre Système	23
3.1.2.3	Avantages et Limites du Filtrage Collaboratif	23
	Avantages	23
	Limites	24
3.1.3	L'Approche Hybride : Synergie pour des Recommandations Optimales	24
3.1.3.1	Justification de l'Approche Hybride	24
3.1.3.2	Méthode de Combinaison : Pondération Linéaire	24
	Normalisation des Scores	25
	Combinaison Pondérée	25
3.1.3.3	Filtrage et Classement des Recommandations	25
3.1.3.4	Sauvegarde des Recommandations	25
3.1.3.5	Avantages de l'Approche Hybride	25
3.2	Processus d'Upload et d'Analyse de CV	25
3.2.1	Étapes du Processus	26
3.2.1.1	Téléchargement du fichier par l'utilisateur (Frontend)	26
3.2.1.2	Réception et validation du fichier (Backend)	26
3.2.1.3	Extraction du texte du CV	26
3.2.1.4	Analyse sémantique du texte (Intégration Groq)	26
3.2.2	Sauvegarde des Résultats et Déclenchement des Recommandations	26
3.2.3	Réponse au frontend	27
3.3	Processus de Web Scraping	27
3.3.1	Configuration et Initialisation du Scraper (EnhancedInternshipScraper)	27
3.3.2	Processus de Scraping par Plateforme	28
3.3.3	Nettoyage et Normalisation des Données (DataCleaner)	28
3.3.4	Sauvegarde des Données dans la Base de Données	29
3.3.5	Fonctions Utilitaires du Scraper (scraper_utils.py)	29
3.4	Gestion des Préférences Utilisateur	29
3.4.1	Structure des Préférences	30
3.4.2	Processus de Définition et de Mise à Jour des Préférences	30
3.4.3	Utilisation des Préférences dans le Moteur de Recommandation	30

3.5	L'architecture de notre système de recommandations	32
4	Prédiction de la filière adaptée	34
4.1	Construction du dataset synthétique	34
4.1.1	Motivation	34
4.1.2	Données simulées	34
4.1.3	Logique d'attribution de la filière	35
4.2	Développement du modèle de prédiction	36
4.2.1	Prétraitement des données	36
4.2.2	Choix de l'algorithme	36
4.2.3	Évaluation du modèle	36
4.3	Mise en place d'une API et d'une interface web	37
4.3.1	Création de l'API avec FastAPI	37
4.3.2	Interface utilisateur développée avec Next.js	38
4.4	Architecture Générale du Système	38
4.4.1	Description de l'Architecture	39
5	Développement du chatbot intelligent	41
5.1	Objectif du Chatbot	41
5.2	Architecture Générale du Système	43
5.3	Importation des Bibliothèques	45
5.4	Préparation et Structuration des Données	45
5.5	Indexation intelligente avec le traitement du langage naturel (NLP)	47
5.6	Recherche contextuelle et extraction ciblée	48
5.6.1	Recherche sémantique (retrieval contextuel)	48
5.6.2	Recherche filtrée par type	48
5.7	Génération de réponse intelligente avec LLaMA 4 (via GROQ API)	49
6	Technologies et Outils Utilisés	52
6.1	Outils de développement	52
6.2	Base de données	53
6.3	Backend	53
6.4	Frontend	55
6.5	Web Scraping	56
6.6	Analyse, IA et Recommandation	57
7	Interface graphique	60
7.1	Page d'Accueil	60
7.2	Écran d'Authentification	63
7.2.1	Écran d'Inscription (Formulaire complet)	63
7.2.2	Écran de Connexion (Formulaire standard)	63
7.2.3	Sécurité des Routes – Accès Non Autorisé	64
7.3	Accueil Stages	65
7.3.1	Écran d'Upload et d'Analyse du CV	65
7.3.2	Écran de Préférences	66
7.3.3	Écran de Recommandations de Stages	67

7.4	Écran de Prédiction de Filière	68
7.4.1	Étape 1 – Informations personnelles	68
7.4.2	Étape 2 – Notes des Semestres 1 et 2	68
7.4.3	Étape 3 – Notes des Semestres 3 et 4	69
7.4.4	Étape 4 – Centres d’intérêt	70
7.4.5	Résultat de la prédiction	71
7.4.6	Aperçu de la filière recommandée	72
7.4.7	Découvrez la filière – Cours principaux	73
7.4.8	Découvrez la filière – Compétences	73
7.4.9	Découvrez la filière – Carrières	73
7.5	Écran du Chatbot	74
	Conclusion	79

Liste des figures

1.1	Diagramme des cas d'utilisation	3
1.2	Diagramme d'activités	6
1.3	Diagramme de classes	7
1.4	Diagramme de séquence - Authentification - partie 1	9
1.5	Diagramme de séquence - Authentification - partie 2	9
1.6	Diagramme de séquence - Recommandation de stage	10
1.7	Diagramme de séquence - Prédiction de la filière	10
1.8	Diagramme de Séquence – Chatbot RAG (Meta-LLaMA + FastAPI + FAISS)	11
1.9	Diagramme de Communication - Interaction des composants	12
2.1	Architecture de Sécurité du Système d'Authentification	18
3.1	Les différentes approches des systèmes de recommandation.	20
3.2	L'architecture du système de recommandation	32
4.1	Matrice de confusion du modèle CatBoost	37
4.2	Architecture de Sécurité du Système d'Authentification	39
5.1	Architecture du système basée sur l'approche RAG	44
6.1	Visual Studio Code	52
6.2	GitHub	53
6.3	PostgreSQL	53
6.4	FastAPI	53
6.5	Pydantic	54
6.6	JWT (JSON Web Token)	54
6.7	React	55
6.8	Next.js	55
6.9	Tailwind CSS	56
6.10	Lucide React	56
6.11	Selenium	56
6.12	BeautifulSoup	57
6.13	FAISS	57
6.14	Scikit-learn	58
6.15	CatBoost	58
6.16	Groq API (Meta LLaMA 4)	58
7.1	Interface visuelle de la page-d'accueil de l'application - partie 1	61
7.2	Interface visuelle de la page-d'accueil de l'application - partie 2	61

7.3	Interface visuelle de la page-d'accueil de l'application - partie 3	62
7.4	Formulaire d'inscription avec validation des champs	63
7.5	Formulaire de connexion avec lien vers la récupération de mot de passe	63
7.6	Redirection vers /login?redirect=/stages lorsque l'utilisateur tente d'accéder à la page des stages sans être connecté.	64
7.7	Redirection vers /login?redirect=/prediction lors d'un accès non autorisé à la page de prédiction de filière.	64
7.8	Redirection vers /login?redirect=/chatbot pour un accès non autorisé à l'écran chatbot.	64
7.9	Page de connexion affichée après redirection automatique en cas d'accès non autorisé.	64
7.10	Interface principale de la page d'accueil des stages avec les différentes étapes du parcours utilisateur	65
7.11	Interface de téléchargement et d'analyse du CV avec modification des domaines et mots-clés extraits - partie 1	66
7.12	Interface de téléchargement et d'analyse du CV avec modification des domaines et mots-clés extraits - partie 2	66
7.13	Interface de personnalisation des préférences de recommandation	67
7.14	Interface des recommandations de stages personnalisées	67
7.15	Interface des recommandations de stages personnalisées	68
7.16	Notes des Semestres 1 et 2	69
7.17	Notes des Semestres 3 et 4	70
7.18	Centres d'intérêt	71
7.19	Résultat de la prédiction	72
7.20	Aperçu de la filière recommandée	72
7.21	Cours principaux	73
7.22	Découvrez la filière – Compétences	73
7.23	Découvrez la filière –Carrières	74
7.24	Interaction sur les questions en Data Analytics	74
7.25	Interaction sur les questions en Data Analytics	75
7.26	Interaction sur les questions en Data Analytics	75
7.27	Interaction sur les questions en Data Analytics	76
7.28	Informations sur la formation	77
7.29	Réponse concernant le Codex Club	78

Introduction Générale

Dans un contexte académique en constante évolution, les étudiants sont confrontés à de nombreux défis dans la construction de leur parcours universitaire et professionnel. Parmi ceux-ci, la recherche de stages adaptés à leur profil, l'orientation vers une filière correspondant à leurs compétences, ou encore la préparation aux exigences du monde professionnel représentent des enjeux majeurs.

Afin de répondre à ces besoins croissants, notre projet tutoré s'inscrit dans une démarche d'innovation pédagogique et technologique, en mettant à profit les avancées récentes en intelligence artificielle, notamment les Large Language Models (LLMs). Le projet consiste en la conception et le développement d'une application intelligente d'aide à l'orientation et à la recherche de stages, spécialement dédiée aux étudiants.

Cette application multifonctionnelle intègre un système de recommandation des offres de stage, personnalisé selon le profil de chaque utilisateur (compétences, préférences, niveau d'étude), ainsi qu'un chatbot intelligent. Ce dernier joue un rôle d'assistant virtuel, capable de fournir des informations sur les compétences requises dans divers domaines, d'accompagner les étudiants dans la préparation de leurs entretiens, et de répondre à leurs questions sur les formations et l'établissement.

En parallèle, un module de prédiction de la filière adaptée a été intégré, se basant sur les résultats académiques et les compétences individuelles des étudiants, afin de leur proposer des orientations pertinentes et personnalisées. L'application offre également un accès structuré à une base de données de travaux académiques, classés par filière, année et entreprise, favorisant ainsi le partage de connaissances et l'inspiration entre étudiants.

Ce rapport présente l'ensemble des étapes de conception, de développement et d'évaluation de cette application. Il met en lumière les choix techniques, les approches algorithmiques et les outils utilisés, tout en soulignant l'impact potentiel de ce type de solution sur l'accompagnement des étudiants dans leur parcours académique et professionnel.

Analyse des besoins et conception fonctionnelle

Introduction

Avant toute phase de développement logiciel, il est essentiel de procéder à une analyse approfondie des besoins fonctionnels et non fonctionnels du système à concevoir. Cette étape permet de bien comprendre les attentes des utilisateurs, les contraintes du projet, ainsi que les objectifs à atteindre, afin de concevoir une architecture cohérente et efficace.

Dans le cadre de notre application, l'analyse des besoins vise à identifier clairement les fonctionnalités principales attendues par les utilisateurs finaux, telles que l'inscription, la connexion, l'accès personnalisé aux ressources, ou encore la sécurité des données. Cette compréhension fine des exigences permet ensuite de définir une conception fonctionnelle précise, traduisant ces besoins sous forme de cas d'utilisation, d'interactions entre les composants, et d'organisation logique du système.

La présente section décrit donc les différentes étapes de cette analyse, les méthodes utilisées pour recueillir les besoins, ainsi que les schémas et modèles fonctionnels qui guideront le développement de l'application.

1.1 Diagramme des cas d'utilisation

Le diagramme des cas d'utilisation est un outil fondamental de modélisation dans l'Unified Modeling Language (UML). Il permet de représenter visuellement les interactions entre les utilisateurs (ou acteurs) et le système, en identifiant les différentes fonctionnalités offertes. Ce type de diagramme aide à mieux comprendre les besoins des utilisateurs, à structurer les exigences fonctionnelles et à faciliter la communication entre les différentes parties prenantes du projet. Il se compose généralement d'acteurs, de cas d'utilisation et de relations illustrant les interactions entre eux.

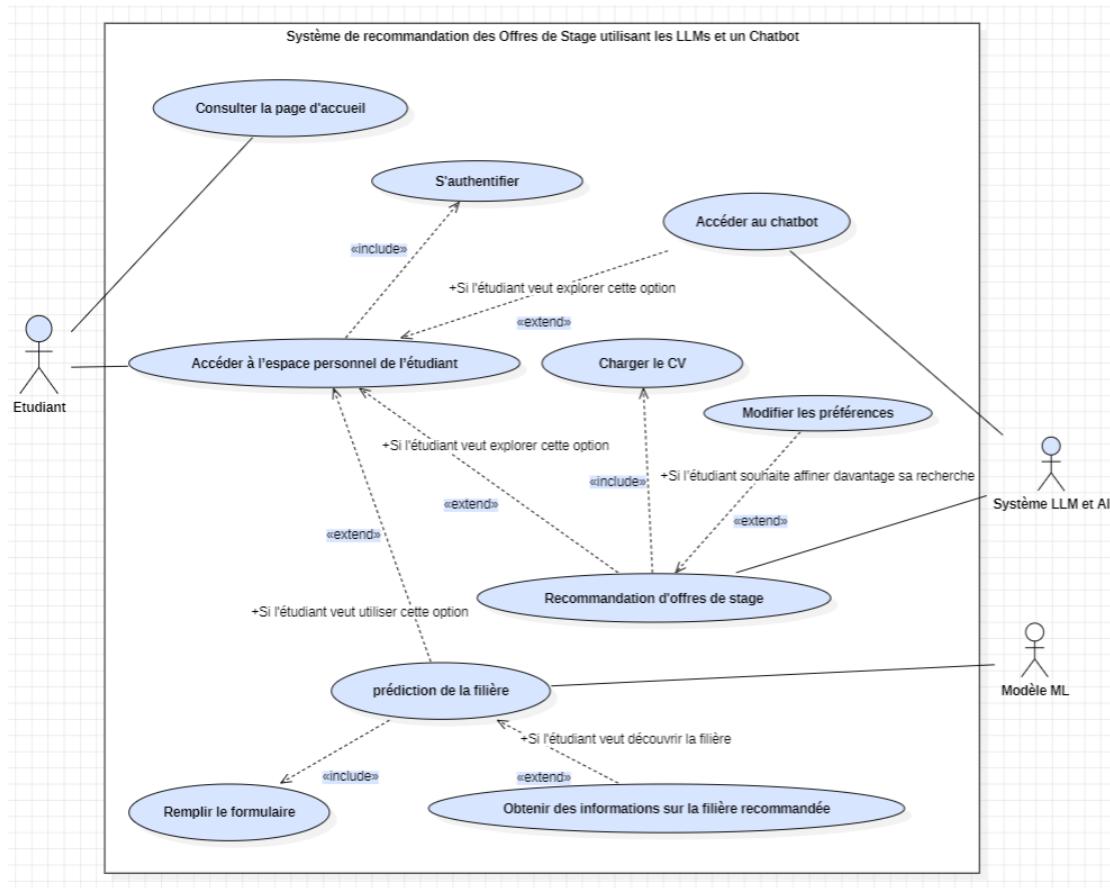


FIGURE 1.1 : Diagramme des cas d'utilisation

Description du diagramme de cas d'utilisation

Le diagramme ci-dessus représente les principales interactions entre les acteurs et les fonctionnalités du **Système de recommandation des Offres de Stage utilisant les LLMs et un Chatbot**. Il modélise les différentes possibilités offertes à un étudiant, ainsi que les interactions avec les composants intelligents du système, notamment le **Modèle de Machine Learning (ML)** et le **Système LLM et AI**.

L'acteur principal est l'étudiant, qui peut :

- Consulter la page d'accueil.
- S'authentifier pour accéder à son espace personnel.
- Utiliser un chatbot basé sur des LLMs pour poser des questions,
- Charger son CV et modifier ses préférences pour personnaliser les recommandations de stage.
- Recevoir des recommandations d'offres de stage basées sur ses informations personnelles, ses préférences et son CV.
- Déclencher un processus de prédition de la filière à partir d'un formulaire, si l'étudiant ne connaît pas encore la spécialisation qui lui correspond.
- Obtenir des informations sur la filière recommandée.

Les cas d'utilisation sont enrichis par des relations d'*inclusion* (*include*) et d'*extension* (*extend*), permettant de modéliser les dépendances logiques entre les actions :

- Par exemple, l'action *Charger le CV* est incluse dans le processus de recommandation.
- De même, *Modifier les préférences* est une extension optionnelle permettant d'affiner les résultats.

Les acteurs secondaires sont :

- Le **Système LLM et AI**, qui intervient dans les tâches de recommandation de stages et dans le fonctionnement du chatbot.
- Le **Modèle ML**, utilisé spécifiquement pour la prédiction de la filière.

Ce diagramme met en évidence la **modularité du système** et la **personnalisation de l'expérience utilisateur**, rendues possibles grâce à l'intégration de l'intelligence artificielle, aussi bien dans le processus de recherche de stage que dans l'assistance fournie par le chatbot.

Description Textuelle

Cette description textuelle détaille le cas d'utilisation **Recommandation d'offres de stage**.

Partie 1 : Identification

- **Titre** : Recommandation d'offres de stage
- **Résumé** : Ce cas d'utilisation décrit le processus par lequel le système génère des recommandations de stages adaptées à l'étudiant.
- **Acteurs** :
 - Acteur principal : Étudiant
 - Acteurs secondaires : Système LLM et IA
- **Dates** :
 - Date de création : 27/03/2025
 - Date de mise à jour : 27/03/2025
- **Responsable** : Équipe de développement du projet
- **Version** : 1.0

Partie 2 : Description des scénarios

Préconditions

- L'étudiant est authentifié sur la plateforme.
- L'étudiant a accédé à sa page principale.

Scénario nominal

1. L'étudiant accède à la fonctionnalité "Recommandation de stage" depuis sa page principale.
2. L'étudiant charge son CV.
3. Extraction des informations à partir du CV.
4. L'étudiant lance le scraping via un bouton.
5. Sauvegarde des informations extraites du CV.
6. Scraping des offres à partir de sites web de stages.
7. Enregistrement des offres trouvées dans la base de données.
8. L'étudiant modifie ses préférences : pays, plateforme, pondération de certains critères.
9. Les préférences sont mises à jour dans la base de données.
10. Le moteur de recommandation hybride traite les données pour effectuer le matching.
11. Le système génère une liste d'offres de stages recommandées et les affiche à l'étudiant.

Scénarios alternatifs

A1 : L'étudiant ne modifie pas ses préférences

- Ce scénario commence au point 8 du scénario nominal.
- Les préférences par défaut sont prises en considération.
- Ces préférences sont enregistrées dans la base de données.
- Le scénario reprend au point 10 du scénario nominal.

Scénarios d'exception

E1 : Aucune offre trouvée

- Ce scénario commence au point 10 du scénario nominal.
- Le système affiche "Aucune offre trouvée".
- L'étudiant est invité à réessayer ultérieurement.
- Le cas d'utilisation se termine en échec.

Postconditions

- Le système conserve un historique des offres de stage affichées.

Partie 3 : Exigences non fonctionnelles

- L'interface de recommandation doit être intuitive et facile à utiliser pour l'étudiant.
- Le système doit garantir un temps de réponse optimal pour générer les recommandations.

1.2 Diagramme d'activités

Les diagrammes d'activités permettent de modéliser le déroulement d'un processus métier ou d'un cas d'utilisation en représentant graphiquement les différentes étapes et transitions. Ils facilitent la compréhension du flux de travail en identifiant les actions et les décisions.

Le diagramme d'activité suivant illustre le processus du cas d'utilisation **Recommandation d'offres de stage**, en détaillant les différentes étapes permettant de suggérer des offres adaptées aux utilisateurs en fonction de leurs profils et préférences.

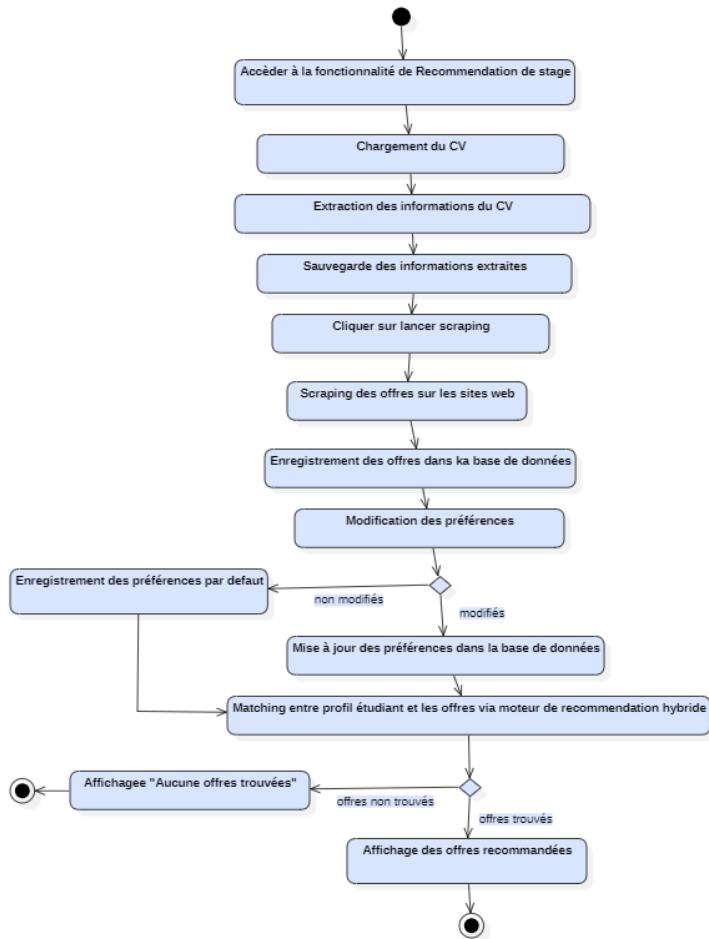


FIGURE 1.2 : Diagramme d'activités

Description du diagramme d'activité – Recommandation d'offres de stage

Ce diagramme d'activité illustre le processus complet de la fonctionnalité de **recommandation d'offres de stage** destinée aux étudiants. Le processus débute par l'accès à cette fonctionnalité, suivi du **chargement du CV** de l'étudiant. Une fois le CV chargé, les **informations pertinentes sont extraites automatiquement**, puis sauvegardées.

L'étudiant peut alors lancer une opération de **scraping** qui permet de collecter des offres de stage depuis divers sites web. Ces offres sont ensuite **enregistrées dans la base de données**.

L'étudiant a la possibilité de **modifier ses préférences de recherche** (pays, plateforme et pondération de certaine critères). S'il ne modifie rien, des préférences par défaut sont enregistrées. Sinon, les nouvelles préférences sont mises à jour dans la base.

Le moteur de recommandation hybride entre ensuite en jeu : il effectue un **matching entre le profil de l'étudiant (CV + préférences) et les offres disponibles**. Si aucune correspondance n'est trouvée, un message l'indique. Sinon, une liste d'offres recommandées est affichée à l'étudiant.

Ce processus vise à **automatiser et personnaliser la recherche de stages**, en facilitant la mise en relation entre les étudiants et les offres les plus adaptées à leur profil.

1.3 Diagramme de classes

Le diagramme de classes suivant modélise l'architecture de la base de données du système de recommandation de stages. Il met en évidence les différentes entités manipulées par la plateforme, leurs attributs principaux ainsi que les relations existantes entre elles. Cette structure vise à garantir la cohérence, la scalabilité et la performance du système, notamment pour les opérations d'analyse de CV et de génération de recommandations.

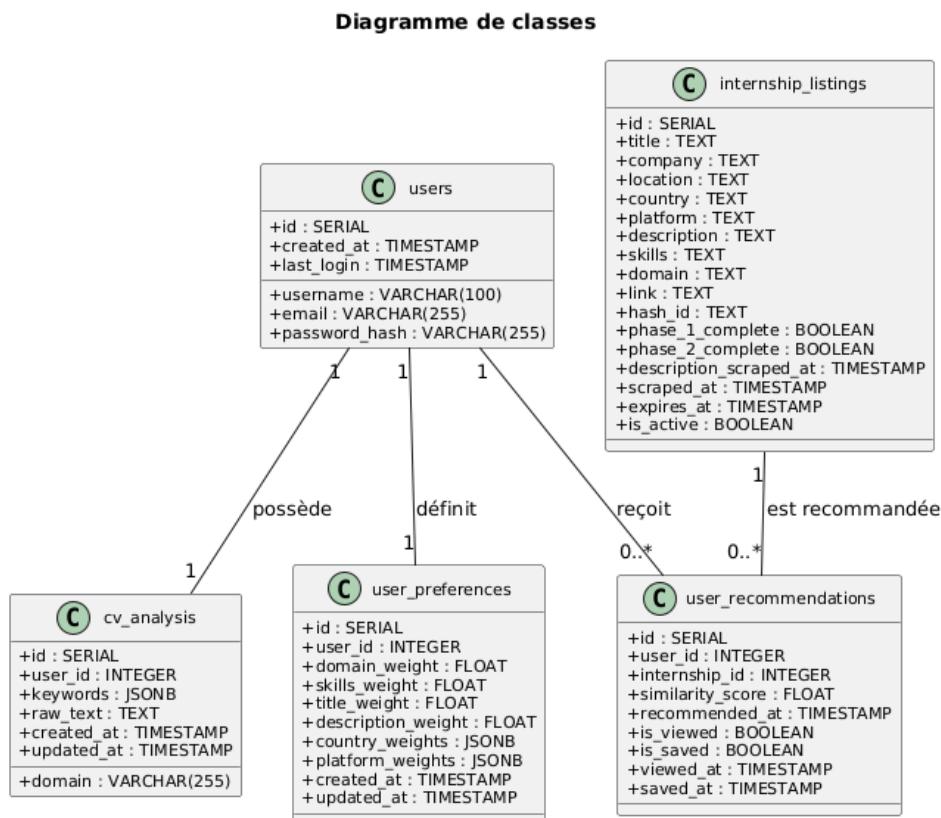


FIGURE 1.3 : Diagramme de classes

1.3.1 Description des classes

users Cette classe représente les utilisateurs de la plateforme. Elle contient les informations d'identification, d'authentification et les métadonnées liées à leur activité (date de création, dernière connexion).

cv_analysis Elle stocke les résultats de l'analyse automatique du CV de l'utilisateur. Ces données incluent le texte brut du CV, les mots-clés extraits et le domaine identifié. Cette classe alimente directement le moteur de recommandation.

user_preferences Cette classe contient les préférences de chaque utilisateur quant aux critères de recommandation. Des poids sont attribués à différents facteurs (domaine, compétences, titre, description), ainsi qu'à des filtres géographiques ou de plateformes. Ces paramètres personnalisent le score de correspondance.

internship_listings Elle regroupe les offres de stage extraites de différentes plateformes. Chaque offre inclut des détails sur l'entreprise, le poste, les compétences recherchées, ainsi que des métadonnées temporelles (date de collecte, d'expiration). Elle constitue la base des recommandations.

user_recommendations Cette classe enregistre les recommandations générées pour chaque utilisateur. Elle associe un utilisateur à une offre de stage via un score de similarité, et permet de suivre les interactions (consultation, sauvegarde) avec l'offre.

1.3.2 Relations entre les classes

- Un utilisateur **possède** une analyse de CV.
- Un utilisateur **définit** ses préférences.
- Un utilisateur peut **recevoir** plusieurs recommandations.
- Une offre de stage peut **être recommandée** à plusieurs utilisateurs.

1.4 Diagramme de Séquence

1.4.1 L'authentification des étudiants

Ce diagramme met en évidence les interactions entre les différents composants du système d'authentification : l'utilisateur interagit via le frontend (Next.js), qui envoie les requêtes au backend (FastAPI). Ce dernier gère l'authentification, le hachage des mots de passe (via passlib et bcrypt), la génération et vérification des tokens JWT (avec python-jose), et communique avec la base de données PostgreSQL pour stocker les informations utilisateur.

Trois phases principales sont représentées :

1. **Inscription** : le mot de passe est haché puis stocké dans la base de données avec les autres informations utilisateur.
2. **Connexion** : si les identifiants sont valides, un token JWT est généré et renvoyé au frontend, sinon une erreur est renournée.

3. Accès aux ressources protégées : le token JWT est vérifié pour authentifier l'utilisateur avant d'autoriser l'accès aux ressources sécurisées.

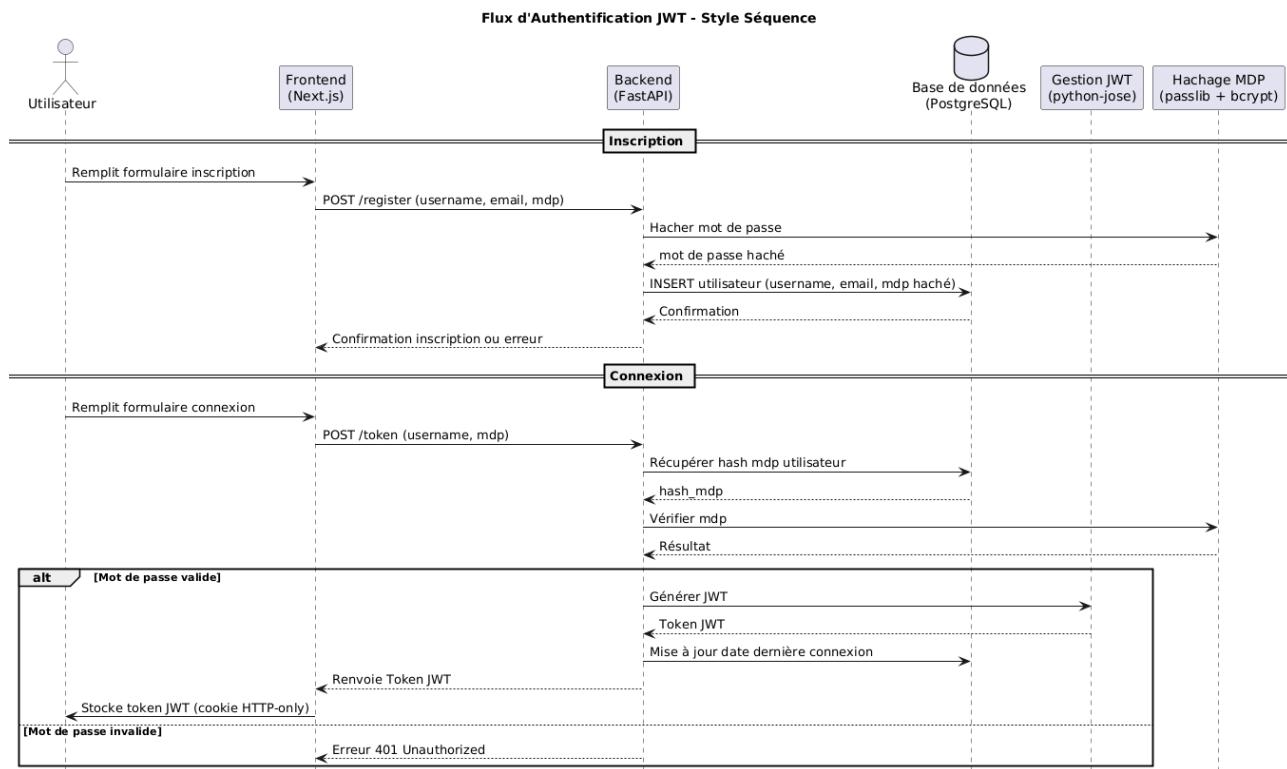


FIGURE 1.4 : Diagramme de séquence - Authentification - partie 1

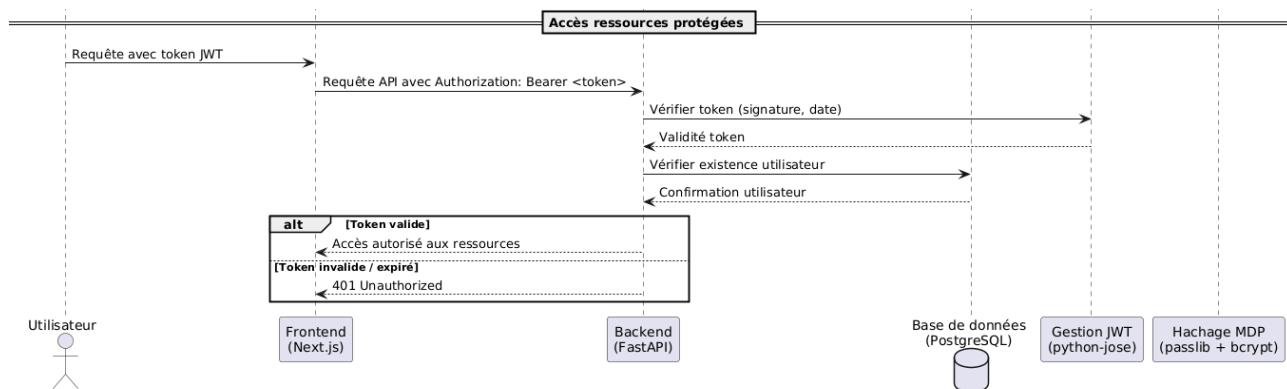


FIGURE 1.5 : Diagramme de séquence - Authentification - partie 2

1.4.2 Le système de recommandation des offres de stage

Ce diagramme de séquence illustre le fonctionnement global du système de recommandation de stages, depuis l'authentification de l'utilisateur jusqu'à la génération des recommandations personnalisées. Il met en lumière l'interaction entre les principaux composants du projet : le frontend (Next.js), le backend (FastAPI), la base de données (PostgreSQL), le module d'analyse de CV (via l'API Groq), et le scraper automatisé d'offres de stages. Le backend orchestre la communication entre ces éléments pour extraire les mots-clés des CV, récupérer les préférences de l'utilisateur, scraper des offres en ligne, et appliquer un moteur de recommandation hybride combinant filtrage collaboratif et basé sur le contenu.

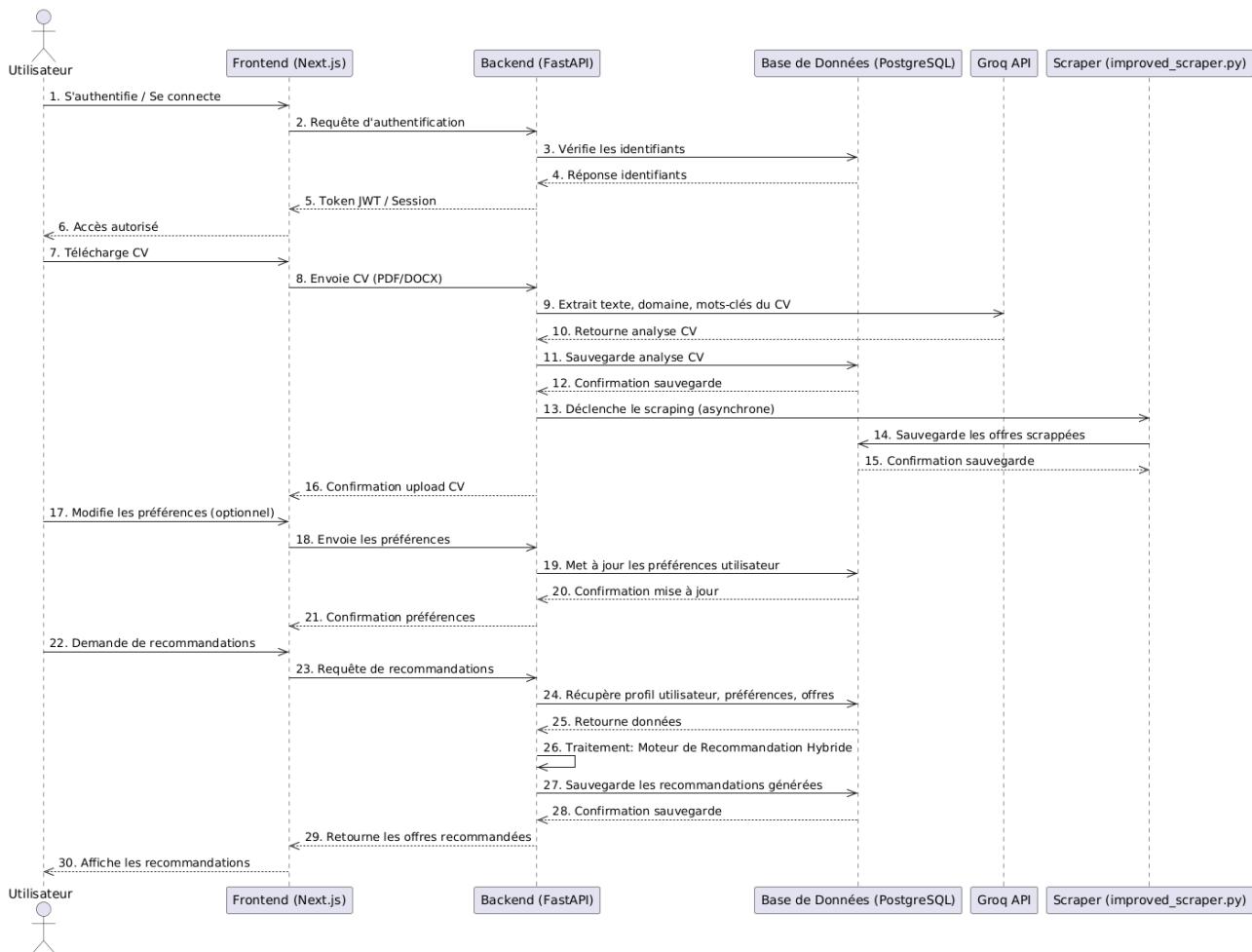


FIGURE 1.6 : Diagramme de séquence - Recommandation de stage

1.4.3 La prédiction de la filière adaptée

Ce diagramme illustre le processus de prédiction de la filière la plus adaptée à un étudiant. L'utilisateur saisit ses données académiques et socio-démographiques via l'interface frontend (Next.js). Ces données sont ensuite envoyées au backend (FastAPI), qui appelle un modèle de Machine Learning (CatBoost). Le modèle traite les données et renvoie une prédiction de filière accompagnée de probabilités. Enfin, le backend transmet les résultats au frontend pour affichage à l'utilisateur.

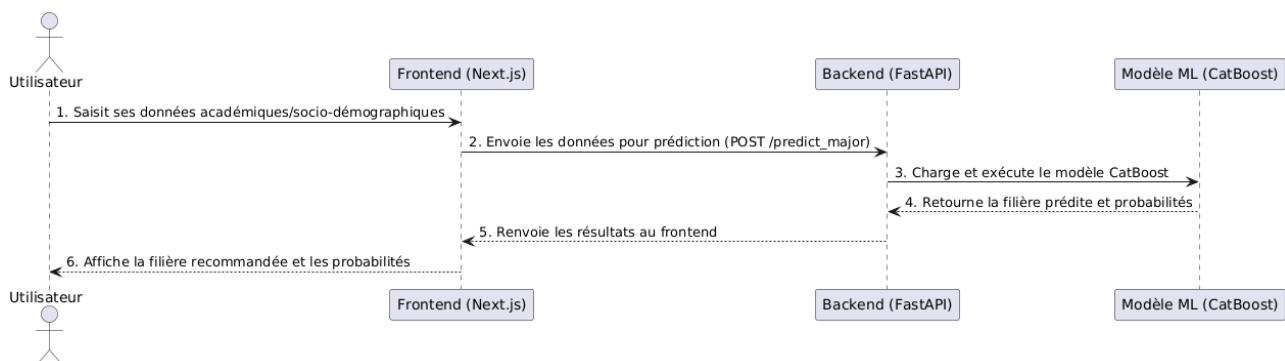


FIGURE 1.7 : Diagramme de séquence - Prédiction de la filière

1.4.4 L'interaction avec un chatbot

Ce diagramme décrit le fonctionnement d'un chatbot intelligent basé sur un modèle de langage (LLM) et la recherche sémantique vectorielle. Lorsqu'un utilisateur saisit une question, celle-ci est envoyée au backend FastAPI qui l'encode via SentenceTransformers en vecteur dense. Ce vecteur est comparé aux documents stockés dans une base documentaire à l'aide de l'index FAISS pour retrouver les plus pertinents. Si des documents sont trouvés, ils sont fournis au LLM (Meta-LLaMA 4 via Groq API) pour générer une réponse enrichie. Sinon, le LLM répond uniquement sur la base de la question. La réponse est ensuite renvoyée à l'utilisateur via l'interface frontend (Next.js).

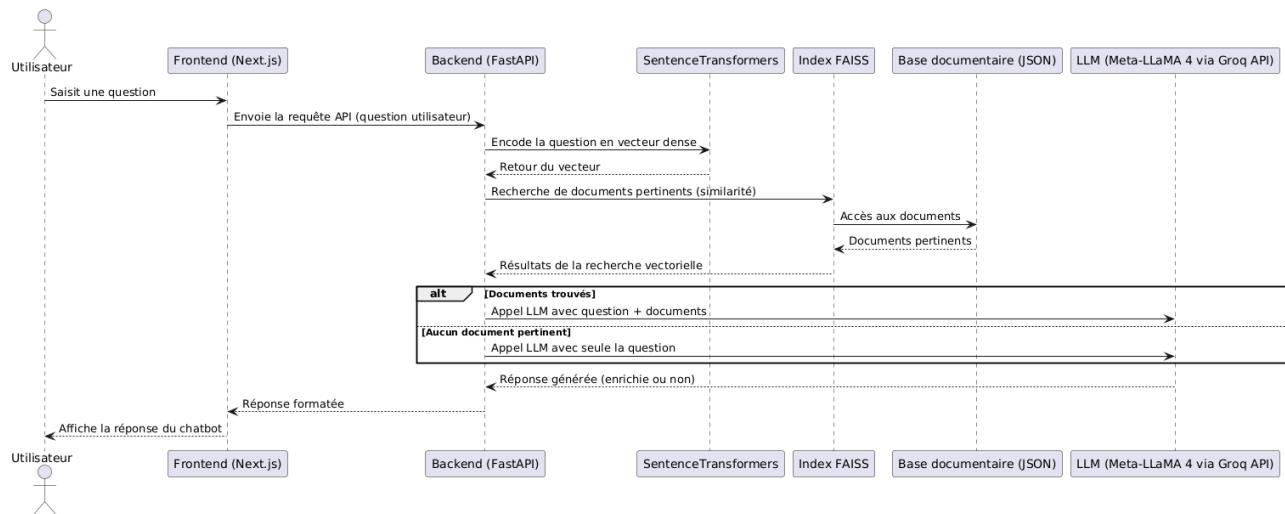


FIGURE 1.8 : Diagramme de Séquence – Chatbot RAG (Meta-LLaMA + FastAPI + FAISS)

1.5 Diagramme de Communication

Le diagramme de communication illustre les interactions entre les différents participants du système, en représentant les échanges de messages entre les acteurs et les composants internes. Contrairement au diagramme de séquence, qui est chronologique, le diagramme de communication se concentre sur la structure des messages échangés dans le système.

Dans ce projet, le diagramme de communication modélise l'ensemble des fonctionnalités de l'application, en mettant en évidence les relations entre les étudiants, le système de recommandation, le chatbot, la base de données, et d'autres composants du système.

1.5.1 Diagramme de Communication du système

Ce diagramme montre comment les différentes entités interagissent pour réaliser les fonctionnalités essentielles du système, telles que l'authentification des étudiants, la recommandation de stage, la prédiction de la filière adaptée et l'interaction avec le chatbot.

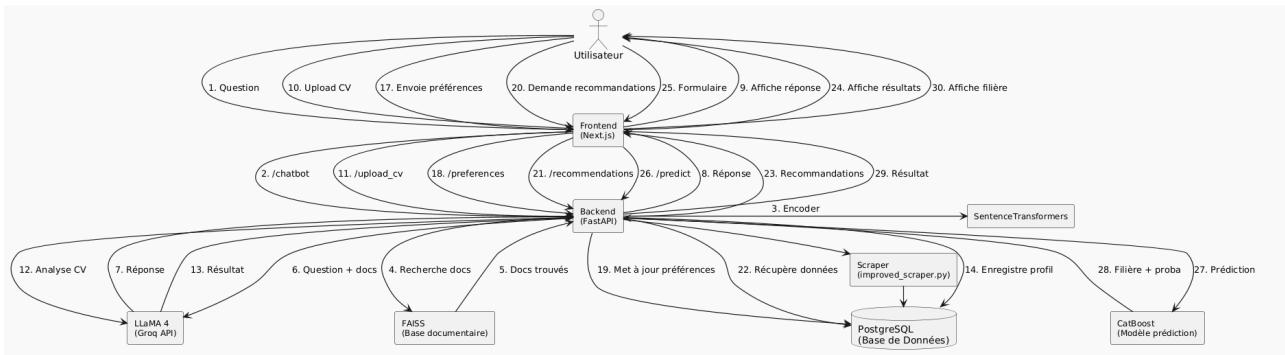


FIGURE 1.9 : Diagramme de Communication - Interaction des composants

1.5.2 Explication du Diagramme de Communication

Ce diagramme de communication illustre le fonctionnement global d'une application intelligente destinée à l'orientation et à l'insertion professionnelle des étudiants. Le système repose sur une architecture modulaire combinant des composants frontend (Next.js), backend (FastAPI), des modèles d'IA, et une base de données PostgreSQL.

L'interaction commence lorsqu'un utilisateur envoie une question via l'interface. Cette requête est transmise au backend qui encode la question à l'aide de SentenceTransformers, interroge une base documentaire FAISS, et transmet le tout au modèle LLaMA 4 via l'API Groq pour générer une réponse pertinente. Celle-ci est ensuite restituée à l'utilisateur.

En parallèle, le système permet d'analyser un CV soumis par l'utilisateur. Ce CV est traité par le modèle LLM qui en extrait des informations clés, stockées dans la base de données. Un scraper Python récupère alors les offres de stages disponibles, qui sont filtrées et recommandées à l'utilisateur selon ses préférences.

Le système propose également une prédiction de la filière la plus adaptée à l'étudiant. Cette prédiction est réalisée à partir des données saisies dans un formulaire, analysées par un modèle CatBoost, et renvoyées à l'utilisateur avec un score de probabilité.

Ce processus intègre des technologies modernes de traitement du langage naturel, de machine learning, et de développement web, garantissant une expérience utilisateur personnalisée et efficace.

Système d'Authentification

Le système d'authentification est une composante fondamentale du projet, assurant la sécurité et la personnalisation de l'accès aux fonctionnalités de l'application. Il est principalement géré par le fichier `main.py` du backend (FastAPI) et interagit avec la table `users` de la base de données (PostgreSQL) définie dans `database_schema.py`. Le frontend (Next.js) fournit les interfaces utilisateur pour l'inscription et la connexion, et gère l'état d'authentification.

2.1 Technologies Utilisées

Le système d'authentification repose sur un ensemble de technologies modernes assurant la sécurité, la rapidité et la maintenabilité de l'application.

- **FastAPI** : Framework web Python permettant d'exposer des endpoints API pour l'authentification.
- **JWT (JSON Web Tokens)** : Utilisé pour créer des tokens d'accès sécurisés et sans état. Un JWT contient des informations sur l'utilisateur et est signé numériquement, permettant au backend de vérifier son intégrité et son authenticité sans interroger la base de données à chaque requête.
- **python-jose** : Bibliothèque Python pour la manipulation des JWT (encodage, décodage, vérification).
- **passlib** : Bibliothèque de hachage de mots de passe, utilisée pour hacher les mots de passe des utilisateurs avant leur stockage en base de données, garantissant la protection des données sensibles même en cas de fuite.
- **bcrypt** : Algorithme de hachage spécifique utilisé par `passlib`. Il est réputé pour sa robustesse contre les attaques par force brute et les tables arc-en-ciel, grâce à sa lenteur et à l'utilisation d'un *salt* pour chaque hachage.
- **OAuth2PasswordBearer** : Schéma de sécurité proposé par FastAPI qui facilite la mise en œuvre de l'authentification basée sur les tokens OAuth2 (Bearer Token).
- **PostgreSQL** : Base de données relationnelle où sont stockées les informations des utilisateurs (nom d'utilisateur, email, hachage du mot de passe, date de création, dernière connexion) dans la table `users`.

- **asyncpg** : Pilote PostgreSQL asynchrone permettant des interactions non bloquantes avec la base de données.

2.2 Processus d'Inscription (Registration)

L'inscription d'un nouvel utilisateur suit un enchaînement d'étapes côté frontend et backend, assurant la sécurité des données et l'intégrité de la base de données :

1. **Frontend (app/inscription/page.tsx)** : L'utilisateur remplit un formulaire d'inscription avec un nom d'utilisateur, une adresse email et un mot de passe.
2. **Requête au Backend** : Ces informations sont envoyées via une requête HTTP POST à un endpoint FastAPI du backend.
3. **Hachage du Mot de Passe** : Dans le backend (`main.py`, dossier `/users`), le mot de passe est haché à l'aide de la commande `pwd_context.hash(user.password)`. Seul le hachage est stocké en base de données — le mot de passe en clair n'est jamais conservé.
4. **Création de l'Utilisateur en Base de Données** : La fonction `create_user` (dans `database_schema.py`) est appelée pour insérer le nouvel utilisateur (nom d'utilisateur, email, mot de passe haché) dans la table `users`.
5. **Gestion des Doublons** : La base de données impose des contraintes d'unicité sur le nom d'utilisateur et l'email. Si ces valeurs existent déjà, une `HTTPException` avec le statut `400 Bad Request` est levée, empêchant la création du compte et informant l'utilisateur de l'erreur.
6. **Réponse au Frontend** : Si l'inscription est réussie, les données du nouvel utilisateur (sans le mot de passe) sont renvoyées au frontend.

2.3 Processus de Connexion (Login)

La connexion d'un utilisateur se déroule selon les étapes suivantes, garantissant l'authentification sécurisée via un token JWT :

1. **Frontend (app/login/page.tsx)** : L'utilisateur saisit son nom d'utilisateur et son mot de passe dans le formulaire de connexion.
2. **Requête au Backend** : Le frontend envoie ces informations à l'endpoint `/token` du backend via une requête HTTP POST. Cet endpoint utilise le schéma `OAuth2PasswordRequestForm` de FastAPI, qui attend les champs `username` et `password`.
3. **Vérification du Mot de Passe** : Le backend (dans `main.py`) utilise le nom d'utilisateur pour récupérer le hachage du mot de passe depuis la base de données.

La fonction `verify_password(plain_password, hashed_password)` de `passlib` est utilisée pour comparer de manière sécurisée le mot de passe fourni avec le mot de passe haché stocké.

4. Génération du Token d'Accès :

- La date de dernière connexion de l'utilisateur est mise à jour dans la base de données (`update_last_login`).
- Un token JWT est généré via la fonction `create_access_token`, contenant :
 - le nom d'utilisateur comme `sub` (sujet du token),
 - une date d'expiration définie par `ACCESS_TOKEN_EXPIRE_MINUTES` (24 heures),
 - une signature sécurisée à l'aide de la clé `SECRET_KEY` et l'algorithme `HS256`.

5. Réponse au Frontend :

Le token d'accès (`access_token`) et le type de token ("bearer") sont renvoyés au frontend.

Celui-ci stocke le token (dans le `localStorage` ou les cookies) et l'utilise dans les requêtes suivantes pour accéder aux ressources protégées.

2.4 Accès aux Ressources Protégées

L'accès aux endpoints sécurisés repose sur l'utilisation de tokens JWT valides et leur vérification par le backend. Voici le processus détaillé :

1. Requêtes Protégées (Frontend) :

Pour accéder aux endpoints nécessitant une authentification (par exemple, `/users/me`, `/upload_cv`, `/get_saved_analysis`), le frontend ajoute le token JWT dans l'en-tête `Authorization` de la requête HTTP, au format suivant :

```
Authorization: Bearer <token>
```

2. Vérification du Token (Backend) :

Dans le backend (FastAPI), les endpoints protégés incluent la dépendance `get_current_user` ou `get_current_active_user`, assurant les vérifications suivantes :

- Extraction du token depuis l'en-tête `Authorization`.
- Décodage et vérification du JWT à l'aide de la `SECRET_KEY` et de l'`ALGORITHM` spécifiés.
- Vérification de la validité du token : signature, expiration.
- Récupération du nom d'utilisateur à partir du `sub` du token, et confirmation de son existence dans la base de données.
- En cas de token invalide, expiré ou d'utilisateur introuvable, une exception `HTTPException` avec un statut 401 `Unauthorized` est levée.

3. Accès Autorisé :

Si toutes les vérifications sont satisfaites, l'objet utilisateur est injecté dans la fonction associée à l'endpoint, permettant ainsi l'exécution de la logique métier de manière sécurisée.

2.5 Gestion de la Sécurité

La sécurité est un aspect fondamental de cette application, particulièrement en ce qui concerne l'authentification, la gestion des mots de passe et la configuration des échanges client-serveur.

- **Hachage des Mots de Passe :**

L'utilisation de `bcrypt` via la bibliothèque `passlib` garantit que les mots de passe ne sont jamais stockés en clair, même si la base de données venait à être compromise. Le hachage est lent et utilise un *salt* unique pour chaque mot de passe, rendant les attaques par force brute ou les tables arc-en-ciel inefficaces.

- **Tokens JWT :**

Les JWT sont *stateless*, ce qui signifie que le serveur n'a pas besoin de conserver l'état des sessions utilisateurs. Cela simplifie l'architecture et permet une bonne scalabilité. Cependant, cette nature sans état implique que les tokens ne peuvent pas être révoqués manuellement avant leur expiration. Il est donc important de bien choisir leur durée de vie (24 heures dans ce projet) pour équilibrer sécurité et praticité.

- **Clé Secrète :**

La `SECRET_KEY` utilisée pour signer les tokens JWT est un élément critique de sécurité. Elle doit être protégée et jamais stockée en clair dans le code source. En environnement de production, elle devrait être stockée dans une variable d'environnement ou un gestionnaire de secrets (comme HashiCorp Vault, AWS Secrets Manager, etc.).

- **CORS (Cross-Origin Resource Sharing) :**

Le backend FastAPI utilise le `CORSMiddleware` pour autoriser les requêtes provenant d'autres origines. Durant le développement, toutes les origines sont permises (`allow_origins=["*"]`), ce qui est pratique.

Le système d'authentification mis en place respecte les bonnes pratiques modernes en matière de sécurité, assurant une gestion fiable et robuste des accès utilisateurs.

2.6 Sécurité des Routes et Gestion de l'Authentification

Dans notre application, la sécurité des routes est assurée principalement via un mécanisme de gestion des tokens d'authentification stockés en cookies `HTTP-only`, combiné à un middleware de protection des routes sensibles.

2.6.1 Gestion des Tokens dans les Routes API

Lors de la connexion (`auth/route.ts`), un token JWT est reçu côté serveur et placé dans un cookie `authToken` avec les attributs suivants pour renforcer la sécurité :

- **httpOnly** : empêche l'accès au cookie depuis le JavaScript côté client, ce qui limite les risques de vol via les attaques XSS (Cross-Site Scripting).

- **secure** : le cookie est transmis uniquement sur une connexion HTTPS en production, assurant la confidentialité des échanges.
- **sameSite=lax** : protège contre certaines attaques *CSRF* (Cross-Site Request Forgery) tout en autorisant la navigation normale entre sites.
- **maxAge** : durée de validité du cookie fixée à 7 jours.

La déconnexion (`logout/route.ts` et suppression via la méthode `DELETE`) consiste à invalider ce cookie en le supprimant (mise à zéro du `maxAge` et suppression explicite), garantissant la déconnexion effective de l'utilisateur.

2.6.2 Protection des Routes avec Middleware

Un middleware global (`middleware.ts`) intercepte toutes les requêtes vers des routes sensibles comme `/stages`, `/prediction`, `/chatbot` et `/pfe`.

Pour ces routes protégées, la présence d'un token d'authentification valide dans les cookies est vérifiée.

En cas d'absence ou de token invalide (token vide ou non présent), l'utilisateur est automatiquement redirigé vers la page de connexion avec un paramètre `redirect` qui conserve l'URL initiale pour faciliter la navigation après authentification.

Cette approche garantit que seules les requêtes authentifiées peuvent accéder aux ressources sensibles.

2.6.3 Gestion du Contexte d'Authentification côté Client

Le contexte React (`context/AutoConfig.tsx`) gère l'état global d'authentification de l'utilisateur, incluant la vérification initiale du token stocké en `localStorage` au chargement de l'application.

Ce contexte propose des fonctions `login` et `logout` qui interagissent avec les routes backend pour obtenir, valider, stocker ou supprimer le token, tout en synchronisant les informations utilisateur.

Le token est envoyé dans les en-têtes HTTP `Authorization: Bearer` pour les requêtes nécessitant une identification, ce qui permet au backend de vérifier la validité du token.

2.6.4 Bonnes Pratiques et Limitations

Le choix du cookie `HTTP-only` améliore la sécurité contre les attaques côté client mais nécessite une gestion prudente côté backend pour valider la validité et l'intégrité du token (non implémentée ici mais recommandée en production).

Le middleware pourrait être étendu pour décoder et valider le JWT, vérifier les permissions utilisateur, et éviter l'accès non autorisé.

La gestion sécurisée du cookie `secure` uniquement en production garantit une meilleure flexibilité lors du développement local.

Cette architecture assure une séparation claire entre les responsabilités frontend et backend, protège les routes sensibles grâce à un contrôle d'accès basé sur les tokens, et sécurise le stockage du token via des cookies `HTTP-only` avec des attributs adaptés.

2.7 Architecture de Sécurité du Système d'Authentification

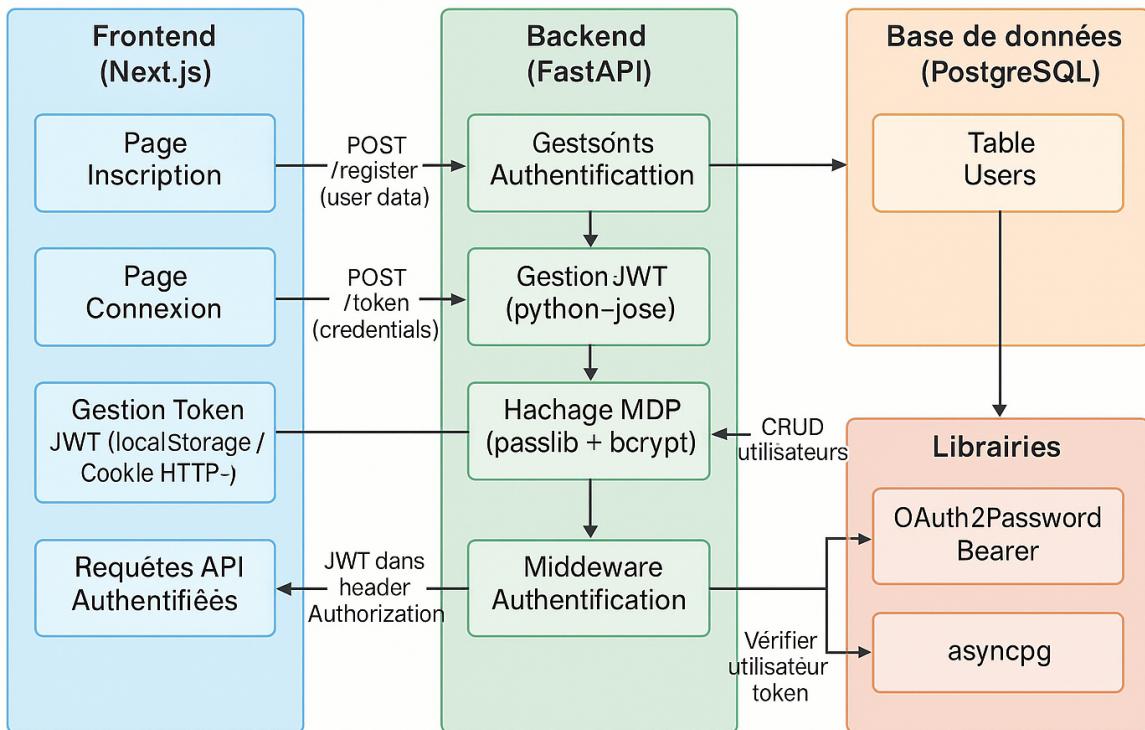


FIGURE 2.1 : Architecture de Sécurité du Système d'Authentification

Le schéma ci-dessous illustre une architecture de sécurité typique basée sur un système d'authentification via **JWT** et **cookies HTTP-only**, utilisée dans notre projet.

- **Inscription / Connexion (Frontend) :**

L'utilisateur saisit ses identifiants via un formulaire. Ces données sont envoyées au backend pour vérification.

- **Vérification et Génération de JWT (Backend) :**

Le backend (FastAPI) vérifie les identifiants, hache les mots de passe avec `bcrypt`, et génère un token `JWT` signé si les informations sont valides.

- **Stockage Sécurisé du Token :**

Le `JWT` est renvoyé au frontend et stocké dans un **cookie HTTP-only** pour prévenir les attaques XSS. Le cookie a les attributs `secure`, `httpOnly` et `sameSite=lax`.

- **Middleware de Protection des Routes (Frontend) :**

Un middleware React (dans `middleware.ts`) intercepte les requêtes vers les pages sensibles (`/stages`, `/chatbot`, etc.).

Si aucun token valide n'est détecté, l'utilisateur est redirigé vers la page de connexion.

- **5. Accès Authentifié aux Ressources (Backend) :**

Le token est envoyé automatiquement via le cookie dans les requêtes. Le backend vérifie le JWT (validité, signature, expiration) avant de donner accès à la ressource.

- **6. Déconnexion Sécurisée :**

Une route dédiée permet de supprimer le cookie, invalidant ainsi la session utilisateur.

Cette architecture assure une séparation claire des responsabilités entre le frontend et le backend, tout en garantissant une protection robuste contre les accès non autorisés.

Conclusion

Le système d'authentification développé dans ce projet repose sur une architecture robuste, moderne et sécurisée, combinant les meilleures pratiques du développement web full-stack. Grâce à l'intégration de FastAPI pour la gestion des requêtes backend, de Next.js pour l'interface utilisateur, et à l'utilisation de technologies comme JWT, bcrypt, et les cookies HTTP-only, ce système garantit la confidentialité des données, la résistance aux attaques courantes (XSS, CSRF, brute force) et une expérience utilisateur fluide.

La séparation des responsabilités entre le frontend et le backend, ainsi que l'utilisation d'un middleware pour contrôler l'accès aux routes sensibles, permettent d'assurer un contrôle rigoureux des accès utilisateurs. Ce système est à la fois évolutif, maintenable et prêt à être déployé en environnement de production, tout en offrant un haut niveau de sécurité.

Système de recommandation des offres de stage

3.1 Système de Recommandation Hybride

Dans un monde numérique où l'information et les choix abondent, la capacité à filtrer et à présenter le contenu le plus pertinent aux utilisateurs est devenue primordiale. Que ce soit pour des produits, des films, des articles ou, dans notre cas, des offres de stage, les systèmes de recommandation jouent un rôle crucial en améliorant l'expérience utilisateur et en optimisant la découverte.

Plutôt que de s'appuyer sur une approche unique, notre système de recommandation adopte une **stratégie hybride**, combinant les forces de plusieurs paradigmes pour offrir des suggestions à la fois précises, personnalisées et diversifiées. Cette section détaille les fondements théoriques et l'implémentation pratique de notre approche, en explorant le **filtrage basé sur le contenu**, le **filtrage collaboratif**, ainsi que la manière dont leur synergie aboutit à un système de recommandation robuste et efficace.

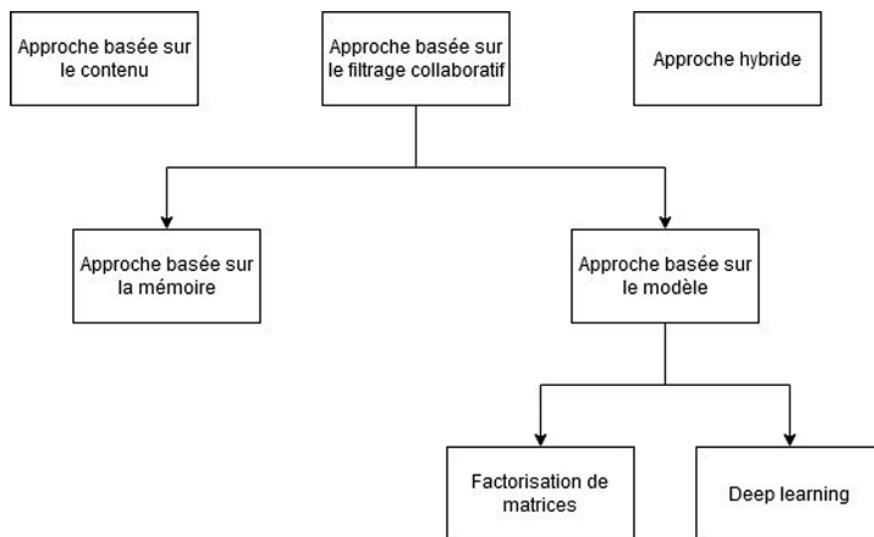


FIGURE 3.1 : Les différentes approches des systèmes de recommandation.

3.1.1 Filtrage Basé sur le Contenu (Content-Based Filtering)

Le filtrage basé sur le contenu est une méthode de recommandation qui se concentre sur les attributs des éléments et les préférences d'un utilisateur spécifique. L'idée fondamentale est de recommander à un utilisateur des éléments similaires à ceux qu'il a appréciés ou consommés par le passé. Cette approche est particulièrement efficace pour fournir des recommandations personnalisées, car elle construit un profil unique pour chaque utilisateur en fonction de ses interactions historiques et de ses préférences explicites.

3.1.1.1 Concept et Fonctionnement

Au cœur du filtrage basé sur le contenu se trouve la notion de **profil utilisateur** et de **profil d'élément**. Chaque élément (dans notre cas, une offre de stage) est décrit par un ensemble de caractéristiques ou d'attributs, tels que le titre, la description, les compétences requises, le domaine d'activité, etc. Simultanément, un profil est construit pour chaque utilisateur, reflétant ses intérêts et ses préférences. Ce profil est généralement dérivé des caractéristiques des éléments avec lesquels l'utilisateur a interagi positivement (par exemple, les stages qu'il a sauvagardés, consultés en détail, ou pour lesquels il a postulé).

Le processus de recommandation implique ensuite de comparer le profil de l'utilisateur avec les profils des éléments non encore vus. Les éléments dont les caractéristiques correspondent le mieux aux préférences de l'utilisateur sont alors recommandés. La similarité entre le profil utilisateur et les profils des éléments est généralement mesurée à l'aide de techniques de similarité textuelle ou vectorielle.

3.1.1.2 Implémentation dans notre Système

Dans notre système de recommandation, la fonction `calculate_content_scores` est l'épine dorsale de l'approche basée sur le contenu. Cette fonction est conçue pour évaluer la pertinence de chaque offre de stage pour un utilisateur donné en se basant sur son profil personnel. Voici les étapes clés de son implémentation :

- **Extraction et traitement des mots-clés du profil utilisateur** : Le profil est enrichi par une analyse du CV de l'utilisateur, d'où sont extraits des mots-clés représentant les compétences, domaines d'intérêt et expériences professionnelles. Ces mots-clés sont traités (conversion JSON en liste Python, gestion des absences, etc.). Si aucun mot-clé valide n'est trouvé, des scores nuls sont attribués à cette partie du filtrage.
- **Préparation des textes des offres de stage** : Pour chaque offre, un texte combiné est créé (titre, description, compétences requises, domaine), permettant une comparaison cohérente avec le profil utilisateur. Les champs manquants sont remplacés par des chaînes vides.
- **Vectorisation TF-IDF et calcul de similarité cosinus** :
 - **TF-IDF (Term Frequency-Inverse Document Frequency)** : utilisé pour transformer les textes combinés des offres et les mots-clés de l'utilisateur en vecteurs numériques. Cette pondération met en valeur les termes distinctifs.
 - **Similarité cosinus** : mesure l'angle entre les vecteurs du profil utilisateur et ceux des offres. Une valeur proche de 1 indique une grande similarité, tandis qu'une valeur proche de 0 indique une faible similarité.

- **Application des pondérations basées sur les préférences utilisateur** : Le système intègre des préférences explicites de l'utilisateur (comme `country_weights` et `platform_weights`) qui modulent les scores finaux. Par exemple, une forte préférence pour un pays entraîne une pondération plus élevée pour les offres correspondantes.

3.1.1.3 Avantages et Limites du Filtrage Basé sur le Contenu

Avantages

- **Personnalisation élevée** : recommandations adaptées aux goûts uniques de chaque utilisateur.
- **Indépendance des autres utilisateurs** : utile pour les nouveaux utilisateurs ou les éléments de niche.
- **Transparence** : il est souvent possible d'expliquer pourquoi une offre a été recommandée.
- **Recommandation de nouveaux éléments** : même si ces éléments n'ont pas encore été évalués.

Limites

- **Sur-spécialisation (over-specialization)** : tendance à recommander uniquement des éléments très similaires.
- **Problème des nouveaux éléments** : difficulté à recommander si les attributs sont absents ou faibles.
- **Qualité des attributs** : dépendance à des descriptions riches et complètes.
- **Diversité limitée** : difficile de proposer des stages dans des domaines inattendus mais pertinents.

En dépit de ces limites, le filtrage basé sur le contenu constitue une base solide pour la personnalisation des recommandations, en particulier lorsqu'il est combiné avec d'autres approches comme le filtrage collaboratif.

3.1.2 Filtrage Collaboratif (Collaborative Filtering)

Le filtrage collaboratif est une technique de recommandation puissante qui s'appuie sur les comportements et les préférences d'un groupe d'utilisateurs pour faire des prédictions ou des recommandations à un utilisateur individuel. L'hypothèse fondamentale est que si deux utilisateurs ont eu des goûts similaires par le passé (par exemple, ils ont aimé les mêmes films, acheté les mêmes produits, ou sauvegardé les mêmes offres de stage), ils sont susceptibles d'avoir des goûts similaires à l'avenir. Cette approche permet de découvrir des éléments qui ne seraient pas nécessairement liés au profil direct de l'utilisateur, mais qui sont populaires parmi des utilisateurs aux préférences similaires.

3.1.2.1 Concept et Fonctionnement

Il existe principalement deux types de filtrage collaboratif : basé sur les utilisateurs (*user-based*) et basé sur les éléments (*item-based*). Notre système utilise une approche **user-based**. Le principe repose sur les étapes suivantes :

1. **Collecte des interactions** : on recueille les interactions des utilisateurs avec les éléments (par exemple : clics, sauvegardes, notes). Dans notre cas, il s'agit des offres de stage que les utilisateurs ont sauvegardées.
2. **Construction de la matrice utilisateur-élément** : les interactions sont organisées dans une matrice où les lignes représentent les utilisateurs, les colonnes les offres de stage, et les cellules indiquent si une interaction a eu lieu (1 si l'utilisateur a sauvégarde le stage, 0 sinon).
3. **Recherche de voisins similaires** : pour un utilisateur donné, le système identifie d'autres utilisateurs ayant des comportements similaires. La similarité est mesurée à l'aide de métriques comme la similarité cosinus ou la corrélation de Pearson.
4. **Prédiction et recommandation** : les préférences de l'utilisateur cible sont estimées à partir des comportements de ses voisins les plus similaires, puis les stages les plus pertinents sont recommandés.

3.1.2.2 Implémentation dans notre Système

La fonction `calculate_collaborative_scores` est responsable de l'implémentation de cette approche dans notre moteur de recommandation. Elle suit rigoureusement les étapes décrites ci-dessus.

- **Récupération des interactions utilisateur-élément :**

Les interactions utilisateur-stage sont extraites de la base de données. Une interaction est définie ici par l'action de `sauvegarder` une offre de stage. Ces données sont stockées sous forme de paires (`user_id, internship_id`).

- **Construction de la matrice d'interaction :**

Les interactions sont transformées en un `DataFrame` Pandas, puis pivotées en une matrice utilisateur-élément. Chaque ligne représente un utilisateur, chaque colonne une offre, et chaque cellule prend la valeur 1 (sauvegardé) ou 0 (non sauvegardé). Cette matrice reflète les préférences implicites des utilisateurs.

- **Modélisation k-NN et recherche de voisins :**

Un modèle `NearestNeighbors` (k plus proches voisins) utilisant la similarité cosinus est entraîné sur la matrice utilisateur-élément. Pour un utilisateur cible, les voisins les plus proches sont identifiés, à l'exclusion de lui-même.

- **Prédiction des scores basée sur les voisins :**

Les scores pour les stages non encore sauvegardés par l'utilisateur sont prédits en agrégant les interactions de ses voisins, pondérées par leur similarité. Plus un voisin est proche, plus il influence la prédiction finale.

3.1.2.3 Avantages et Limites du Filtrage Collaboratif

Avantages

- **Découverte de nouveaux éléments** : recommande des stages que l'utilisateur n'aurait pas trouvés seul.

- **Indépendant des attributs** : ne nécessite pas de métadonnées sur les stages.
- **Capture des goûts complexes** : détecte des corrélations subtiles entre utilisateurs.

Limites

- **Démarrage à froid :**
 - **Utilisateurs** : difficile de recommander à un nouvel utilisateur sans historique.
 - **Stages** : un nouveau stage ne peut être recommandé s'il n'a pas encore été sauvegardé.
- **Rareté des données (sparsity)** : peu d'interactions rendent difficile la recherche de voisins fiables.
- **Problèmes de scalabilité** : coûteux pour les systèmes très volumineux.
- **Vulnérabilité aux attaques (shilling attacks)** : manipulation possible via de faux profils ou fausses interactions.

Malgré ces limitations, le filtrage collaboratif reste un pilier des systèmes de recommandation. Il permet de proposer des suggestions inédites et pertinentes, d'où l'intérêt de le combiner avec d'autres méthodes dans une approche hybride.

3.1.3 L'Approche Hybride : Synergie pour des Recommandations Optimales

Bien que le filtrage basé sur le contenu et le filtrage collaboratif soient des méthodes puissantes, chacune présente des forces et des faiblesses. Le filtrage basé sur le contenu excelle dans la personnalisation et la gestion du démarrage à froid pour les nouveaux éléments, mais peut souffrir de sur-spécialisation. Le filtrage collaboratif, quant à lui, est efficace pour la découverte de nouveaux éléments, mais il est vulnérable au démarrage à froid pour les nouveaux utilisateurs et à la rareté des données. Pour pallier ces limitations, notre système adopte une stratégie hybride.

3.1.3.1 Justification de l'Approche Hybride

L'objectif principal est d'améliorer la qualité globale des recommandations en combinant les deux approches :

- **Atténuer le Démarrage à Froid** : Le filtrage par contenu permet des recommandations basées sur le profil (ex : mots-clés du CV), même sans historique.
- **Améliorer la Précision et la Couverture** : Le collaboratif découvre des éléments que le contenu seul aurait négligés, et inversement.
- **Augmenter la Diversité** : La combinaison enrichit la variété des suggestions, évitant la sur-spécialisation.
- **Renforcer la Robustesse** : Le système devient plus résilient face aux données manquantes ou bruitées.

3.1.3.2 Méthode de Combinaison : Pondération Linéaire

Nous utilisons une pondération linéaire implémentée dans la fonction `generate_hybrid_recommendation`. Chaque méthode contribue à un score final pondéré.

Normalisation des Scores Les scores (cosinus pour le contenu, autres pour le collaboratif) peuvent avoir des échelles différentes. Une normalisation préalable (via `normalize_scores`) ramène tous les scores entre 0 et 1, assurant une contribution équitable et évitant les divisions par zéro.

Combinaison Pondérée Les scores normalisés sont combinés selon la formule suivante :

$$\text{Score}_{\text{Hybride}} = \alpha \times \text{Score}_{\text{Contenu}} + (1 - \alpha) \times \text{Score}_{\text{Collaboratif}} \quad (3.1)$$

Le paramètre α (par défaut 0.6) détermine l'importance de chaque composante. Une valeur de 0.6 signifie que 60% du score vient du contenu, et 40% du collaboratif. Ce paramètre peut être ajusté selon les performances souhaitées (ex : plus de personnalisation ou de découverte).

3.1.3.3 Filtrage et Classement des Recommandations

Deux étapes clés suivent le calcul des scores hybrides :

1. **Filtrage des Offres Déjà Interagies** : Les stages déjà consultés ou sauvagardés sont exclus.
2. **Classement et Sélection des Top-N** : Les offres restantes sont triées par score décroissant. Les N meilleures (par défaut 10) sont sélectionnées, en éliminant les valeurs non valides (NaN ou trop faibles).

3.1.3.4 Sauvegarde des Recommandations

Les recommandations générées sont stockées dans la table `user_recommendations` de la base de données, permettant de suivre l'historique des recommandations et d'éviter de les proposer inutilement.

3.1.3.5 Avantages de l'Approche Hybride

- **Précision Améliorée** : Fusion des sources pour des recommandations plus fiables.
- **Robustesse au Démarrage à Froid** : Utilise le profil utilisateur ou les attributs d'un stage en l'absence d'interactions.
- **Diversité et Nouveauté** : Encourage la découverte de stages variés et inattendus.
- **Flexibilité** : α permet d'adapter le système à divers objectifs (personnalisation vs. exploration).

Conclusion : L'approche hybride combine le meilleur des deux mondes — contenu et collaboratif — offrant des recommandations plus pertinentes, variées et robustes, adaptées aux profils uniques des utilisateurs.

3.2 Processus d'Upload et d'Analyse de CV

La fonctionnalité d'upload de CV est une composante essentielle du projet, car elle alimente le système de recommandation en informations clés sur le profil de l'étudiant. Ce document décrit en détail les différentes étapes du processus d'upload et d'analyse d'un CV.

3.2.1 Étapes du Processus

3.2.1.1 Téléchargement du fichier par l'utilisateur (Frontend)

- L'utilisateur accède à la page d'upload de CV (`app/cv-upload/page.tsx`) sur le frontend.
- Il sélectionne un fichier CV depuis son appareil.
- Le frontend accepte les formats PDF et DOCX, validés côté backend.
- Le fichier est envoyé au backend via une requête HTTP POST vers l'endpoint `/upload_cv`.

3.2.1.2 Réception et validation du fichier (Backend)

- L'endpoint `/upload_cv` de FastAPI reçoit le fichier (`UploadFile`) envoyé par le frontend.
- Validation du format : seuls les fichiers PDF (`.pdf`) et DOCX (`.docx`) sont acceptés. Sinon, une erreur HTTP 400 est renvoyée.
- Lecture du contenu binaire du fichier via `await file.read()`.

3.2.1.3 Extraction du texte du CV

La fonction `extract_text_from_CV_bytes` convertit le contenu binaire en texte brut :

- Pour les PDF : utilisation de la bibliothèque `pdfplumber` pour ouvrir le fichier en mémoire (`io.BytesIO`) et extraire le texte page par page.
- Pour les DOCX : utilisation de la bibliothèque `python-docx` pour extraire le texte de chaque paragraphe.
- Gestion des erreurs : en cas de problème d'extraction, une exception est levée et une erreur HTTP 500 est retournée.

3.2.1.4 Analyse sémantique du texte (Intégration Groq)

- Le texte extrait est envoyé à l'API Groq via un prompt conçu pour identifier :
 - Le domaine de stage ciblé (ex. : *Développement Logiciel, Data Science, etc.*).
 - Une liste de mots-clés pertinents (ex. : Python, Machine Learning, SQL).
- La fonction `query_groq` envoie le prompt à l'API en utilisant une clé API et spécifie un modèle avec une température basse (0.3) pour garantir la précision.
- La réponse est nettoyée pour extraire un bloc JSON contenant le domaine et les mots-clés, avec gestion d'erreurs de parsing JSON.

3.2.2 Sauvegarde des Résultats et Déclenchement des Recommandations

- **Stockage dans la Base de Données** : Le domaine et les mots-clés extraits par l'analyse Groq sont sauvegardés dans la table `cv_analysis` de la base de données. Cette opération est réalisée par la fonction `save_keywords_to_db` (ou `save_cv_analysis` dans `database_schema.py`), qui met à jour l'entrée existante pour l'utilisateur ou crée une nouvelle entrée si c'est la première analyse de son CV.

- **Déclenchement Asynchrone des Recommandations :** Après la sauvegarde réussie de l'analyse du CV, le système déclenche de manière asynchrone la génération de recommandations hybrides pour l'utilisateur. Cela est réalisé en utilisant `background_tasks.add_task` et en appelant la fonction `asyncio.create_task` pour exécuter `generate_hybrid_recommendations` du module `recommendation.py`. Cette approche asynchrone permet à l'API de répondre rapidement à l'utilisateur sans attendre la fin du processus de recommandation, qui peut être plus long.

3.2.3 Réponse au frontend

Une fois le processus terminé, une réponse est renvoyée au frontend, indiquant que le CV a été traité avec succès et incluant le domaine ainsi que les mots-clés extraits. Le frontend peut alors utiliser ces informations pour afficher un résumé à l'utilisateur ou pour mettre à jour l'interface.

Ce processus illustre une chaîne complète, de la réception du fichier CV jusqu'à l'analyse sémantique avancée, permettant d'enrichir le profil utilisateur et de déclencher des recommandations personnalisées.

3.3 Processus de Web Scraping

Le web scraping est une fonctionnalité cruciale du backend, implémentée principalement dans `improved_scraper.py` et assistée par `scraper_utils.py`. Son objectif est de collecter des offres de stage à jour à partir de diverses plateformes en ligne, telles que LinkedIn, Indeed et Glassdoor. Ce processus est essentiel pour alimenter le système de recommandation avec un flux continu de données pertinentes. L'implémentation utilise `undetected_chromedriver` et Selenium pour naviguer sur les sites web et extraire les informations, tout en tentant de contourner les mécanismes anti-bot.

3.3.1 Configuration et Initialisation du Scraper (`EnhancedInternshipScraper`)

- **Initialisation :** La classe `EnhancedInternshipScraper` est initialisée avec des options telles que le mode `headless` (pour exécuter le navigateur sans interface graphique) et une liste de pays cibles (par défaut : "Morocco", "France", "Canada").
- **Configuration du Driver Chrome :** La méthode `_setup_driver` configure `undetected_chromedriver`, une version modifiée de ChromeDriver qui tente d'éviter la détection par les sites web en modifiant certaines propriétés du navigateur souvent utilisées pour identifier les bots (par exemple, `navigator.webdriver`). Des arguments sont ajoutés pour désactiver le sandbox, les notifications, le blocage des pop-ups et les extensions, ce qui est courant pour les opérations de scraping.
- **Outils Utilitaires :** Le scraper utilise des utilitaires internes comme `LocalStorage` pour sauvegarder temporairement les données scrappées, et `DataCleaner` pour nettoyer et normaliser les informations extraites.

3.3.2 Processus de Scraping par Plateforme

Le scraper implémente des méthodes spécifiques pour chaque plateforme cible, adaptées à leur structure HTML et à leurs mécanismes de chargement de contenu. Bien que les détails varient, le flux général est similaire :

- **Navigation :** Le driver Selenium navigue vers l'URL de recherche d'offres de stage pour un mot-clé (`kw`) et un pays (`country`) donnés. Les URLs sont construites dynamiquement pour cibler les pages de recherche pertinentes (par exemple, `linkedin.com/jobs/search`, `indeed.com/jobs`, `glassdoor.fr/Emploi`).
- **Attente et Défilement :** Après avoir chargé la page, le scraper attend un certain temps (`_sleep`) pour permettre au contenu de se charger. Pour les sites qui chargent du contenu dynamiquement (ex. défilement infini), le scraper exécute des scripts JavaScript (`window.scrollTo`) pour simuler le défilement de l'utilisateur et charger plus d'offres.
- **Gestion des Consents (Cookies) :** Pour des plateformes comme Indeed, le scraper tente de gérer les bannières de consentement aux cookies en cliquant sur les boutons d'acceptation, afin de ne pas bloquer l'accès au contenu.
- **Extraction des Données :** Le scraper identifie les éléments HTML correspondant aux cartes d'offres d'emploi (ex. `.job-search-card` pour LinkedIn, `.job_seen_beacon` pour Indeed) en utilisant des sélecteurs CSS (`By.CSS_SELECTOR`). Pour chaque carte, il extrait des informations clés comme le titre, l'entreprise, la localisation, le lien de l'offre, et parfois le salaire, en utilisant des sélecteurs spécifiques (`By.CLASS_NAME`, `By.TAG_NAME`).
- **Génération de Hash ID :** Pour chaque offre extraite, un `Hash_ID` unique est généré (par exemple, via SHA256 sur une combinaison du titre, de l'entreprise, de la localisation et de la plateforme). Ce `Hash_ID` est crucial pour la déduplication des offres dans la base de données.
- **Stockage Temporaire :** Les données extraites sont stockées temporairement dans une liste de dictionnaires en mémoire, puis sauvegardées par lots dans des fichiers JSON locaux par la classe `LocalStorage`.

3.3.3 Nettoyage et Normalisation des Données (DataCleaner)

Avant d'être sauvegardées dans la base de données, les données brutes extraites sont nettoyées et normalisées par la classe `DataCleaner` pour assurer la cohérence et la qualité :

- `clean_text` : supprime les espaces blancs excessifs et certains caractères spéciaux des chaînes de texte.
- `clean_company_name` : nettoie les noms d'entreprise en supprimant les suffixes courants (Inc., LLC, Ltd., etc.).
- `clean_location` : standardise les données de localisation en supprimant les préfixes comme "Remote", "Hybrid", "On-site".
- `extract_skills` : identifie et extrait des compétences techniques clés à partir de la description de l'offre, en utilisant des expressions régulières et une liste prédéfinie de motifs de compétences. Les compétences trouvées sont ensuite formatées et jointes en une seule chaîne.

3.3.4 Sauvegarde des Données dans la Base de Données

- **Connexion à la Base de Données :** Le scraper interagit avec la base de données PostgreSQL via `asyncpg`. Une pool de connexions (`get_pool`) est utilisée pour gérer efficacement les connexions.
- `save_internship_listing` : Cette fonction insère ou met à jour les offres dans la table `internship_listings`. Elle utilise le `hash_id` pour détecter les doublons : si une offre avec le même `hash_id` existe déjà, elle est mise à jour ; sinon, une nouvelle entrée est créée.
- **Gestion des Phases :** Le schéma de la base comprend les champs `phase_1_complete` et `phase_2_complete`. `phase_1_complete` est défini à TRUE après le scraping initial (collecte des informations de base). `phase_2_complete` est initialement FALSE et sera mis à jour après enrichissement de la description (par exemple, en visitant le lien de l'offre pour extraire la description complète).

3.3.5 Fonctions Utilitaires du Scraper (`scraper_utils.py`)

Ce fichier fournit des fonctions pour la maintenance et l'analyse des données scrappées :

- `get_scraping_stats` : statistiques sur les offres dans la base (nombre total, ajouts récents, répartition par plateforme et pays).
- `cleanup_duplicates` : supprime les doublons dans `internship_listings` en se basant sur `hash_id`, en conservant l'entrée la plus récente.
- `get_phase_statistics` : statistiques détaillées sur l'état d'avancement des phases de scraping (phase 1 et phase 2).
- `get_records_needing_phase_2` : identifie les offres scrappées en phase 1 mais sans description détaillée (phase 2 incomplète).
- `mark_phase_2_complete` : met à jour le statut `phase_2_complete` pour les offres enrichies.
- `cleanup_phase_data` : nettoyage général incluant suppression des doublons, des enregistrements incomplets, et normalisation des champs vides.

Le processus de scraping forme ainsi un pipeline robuste qui collecte les données, assure leur qualité et organisation dans la base, préparant efficacement le terrain pour le système de recommandation.

3.4 Gestion des Préférences Utilisateur

La gestion des préférences utilisateur est une fonctionnalité clé qui permet aux étudiants de personnaliser davantage les recommandations de stages qu'ils reçoivent. Implémentée principalement via la table `user_preferences` dans `database_schema.py` et utilisée par le moteur de recommandation dans `recommendation.py`, cette section offre une flexibilité pour affiner la pertinence des offres.

3.4.1 Structure des Préférences

La table `user_preferences` stocke les paramètres de personnalisation pour chaque utilisateur. Ces préférences incluent :

- **Poids des Attributs** : Des valeurs flottantes (`domain_weight`, `title_weight`, `skills_weight`, `description_weight`) qui déterminent l'importance relative de chaque attribut d'une offre de stage lors du calcul de la similarité. Par exemple, un utilisateur peut accorder plus d'importance au domaine qu'aux compétences.
- **Poids par Pays** (`country_weights`) : Un objet JSON (JSONB) qui permet à l'utilisateur de spécifier une préférence pour les stages dans certains pays. Chaque pays peut avoir un poids associé (par exemple, {"Morocco" : 1.0, "France" : 0.8}). Un poids de 1.0 signifie une préférence neutre ou par défaut, tandis qu'une valeur inférieure peut réduire la pertinence des offres de ce pays.
- **Poids par Plateforme** (`platform_weights`) : Similaire aux poids par pays, cet objet JSON permet de privilégier ou de déprioriser les offres provenant de plateformes spécifiques (par exemple, {"LinkedIn" : 1.0, "Indeed" : 0.7}).
- Ces préférences sont stockées sous forme de JSONB dans la base de données, offrant une flexibilité pour ajouter de nouvelles préférences sans modifier le schéma de la table.

3.4.2 Processus de Définition et de Mise à Jour des Préférences

- **Interface Utilisateur (Frontend)** : Le frontend (`app/preferences/page.tsx`) fournit une interface où l'utilisateur peut visualiser et modifier ses préférences. Cela peut inclure des sliders pour ajuster les poids des attributs, et des listes déroulantes ou champs de saisie pour définir les poids par pays et par plateforme.
- **Envoi au Backend** : Lorsque l'utilisateur sauvegarde ses modifications, le frontend envoie ces préférences au backend via un endpoint API (probablement dans `main.py` ou un module dédié aux préférences). Les données sont validées par un modèle Pydantic (`UserPreferences` dans `main.py`).
- **Sauvegarde dans la Base de Données** : Le backend utilise la fonction `save_user_preferences` (définie dans `database_schema.py`) pour stocker ou mettre à jour les préférences de l'utilisateur dans la table `user_preferences`. Si l'utilisateur n'a pas encore de préférences enregistrées, une nouvelle entrée est créée avec les valeurs par défaut. Sinon, l'entrée existante est mise à jour.

3.4.3 Utilisation des Préférences dans le Moteur de Recommandation

Les préférences utilisateur jouent un rôle direct dans le calcul des scores de recommandation, spécifiquement dans la fonction `calculate_content_scores` du module `recommendation.py`.

- **Récupération des Préférences** : Avant de calculer les scores de contenu, le moteur de recommandation récupère les préférences de l'utilisateur à partir de la base de données en utilisant `get_user_preferences`.

- **Application des Poids des Attributs :** Les poids (`skills_weight`, `title_weight`, `domain_weight`, `description_weight`) sont utilisés pour pondérer la contribution de chaque partie du texte combiné de l'offre de stage (titre, description, compétences, domaine) lors de la vectorisation ou du calcul de similarité. Bien que l'implémentation actuelle dans `calculate_content_scores` combine le texte avant la vectorisation, une application plus granulaire des poids pourrait être envisagée en pondérant les scores de similarité pour chaque attribut individuellement avant de les agréger.
- **Application des Multiplicateurs Pays/Plateforme :** Après le calcul initial de la similarité cosinus, les scores sont multipliés par des facteurs basés sur le pays et la plateforme de l'offre. Par exemple, si un pays ou une plateforme a un poids de 0.5, les offres correspondantes verront leur score de recommandation réduit de moitié. Si un pays ou une plateforme n'est pas spécifié dans les préférences de l'utilisateur, un poids par défaut de 1.0 est appliqué, ce qui signifie qu'il n'y a pas d'impact sur le score.

Cette approche permet au système de recommandation de s'adapter dynamiquement aux préférences explicites de l'utilisateur, offrant ainsi une expérience plus personnalisée et pertinente. Les utilisateurs peuvent affiner les résultats pour correspondre précisément à leurs critères de recherche et à leurs priorités.

3.5 L'architecture de notre système de recommandations

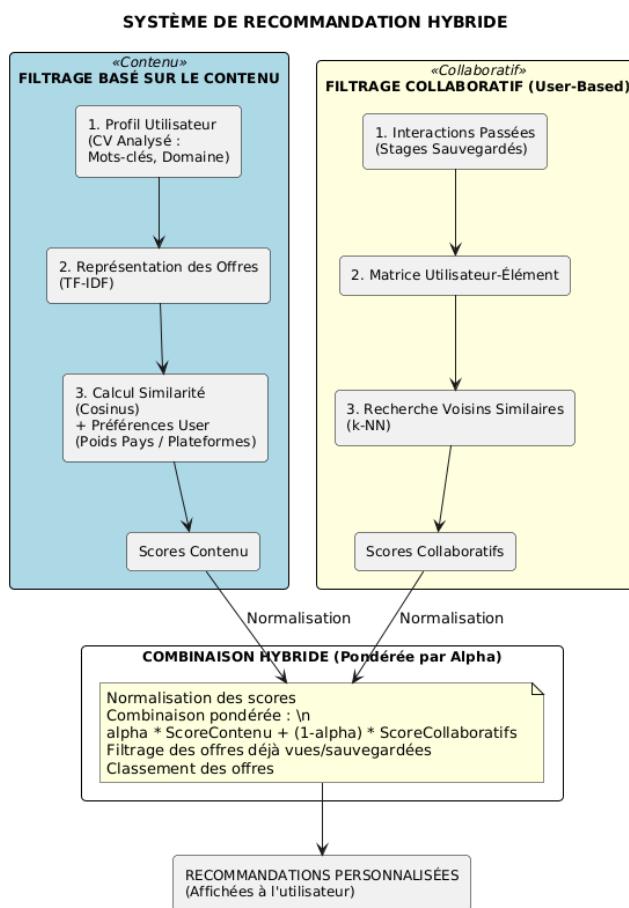


FIGURE 3.2 : L'architecture du système de recommandation

Explication du Schéma

Ce schéma illustre les deux piliers du système de recommandation hybride et comment leurs résultats sont combinés pour générer les recommandations finales.

Filtrage Basé sur le Contenu :

- Prend en entrée le Profil Utilisateur (mots-clés et domaine du CV analysé) et les Offres de Stage.
- Les offres sont transformées en représentations numériques (vecteurs TF-IDF).
- La Similarité Cosinus est calculée entre le profil utilisateur et chaque offre.
- Les Préférences Utilisateur (poids pour les pays, plateformes, etc.) sont appliquées pour ajuster la pertinence.
- Le résultat est un ensemble de Scores Contenu pour chaque offre.

Filtrage Collaboratif :

- Se base sur les Interactions Passées des utilisateurs (stages qu'ils ont sauvegardés).
- Ces interactions sont utilisées pour construire une Matrice Utilisateur-Élément.
- Le système identifie les Voisins Similaires à l'utilisateur actuel via l'algorithme k-NN.
- Le résultat est un ensemble de Scores Collaboratifs basés sur les préférences des utilisateurs similaires.

Combinaison Hybride :

- Les Scores Contenu et les Scores Collaboratifs sont normalisés pour être comparables.
- Ils sont combinés de manière pondérée avec un paramètre *alpha* permettant de moduler l'importance de chaque méthode.
- Les offres déjà vues ou sauvegardées par l'utilisateur sont filtrées.
- Les offres restantes sont classées selon leur score hybride.

Recommendations Personnalisées :

La liste finale des offres les mieux classées est présentée à l'utilisateur, offrant une recommandation robuste et adaptée, même en cas de données limitées pour une des approches (démarrage à froid).

Ce schéma met en évidence la complémentarité des deux méthodes pour fournir des recommandations pertinentes et personnalisées.

Conclusion

Le développement d'un système de recommandation hybride constitue une réponse pertinente aux limites inhérentes aux approches individuelles que sont le filtrage basé sur le contenu et le filtrage collaboratif. En combinant leurs forces respectives, notre système parvient à offrir des recommandations plus précises, plus diversifiées et mieux adaptées aux profils variés des utilisateurs.

Grâce à l'analyse des mots-clés extraits des CVs et à l'apprentissage des comportements collectifs des utilisateurs, le moteur de recommandation hybride améliore l'expérience de recherche de stages, tout en tenant compte des préférences explicites et implicites. La stratégie de pondération linéaire permet une adaptation fine du système en fonction des besoins, qu'ils soient orientés vers la personnalisation ou l'exploration.

Ainsi, notre système constitue une solution robuste, évolutive et personnalisée, essentielle pour guider efficacement les étudiants dans la découverte d'opportunités professionnelles pertinentes. Il s'intègre naturellement dans un environnement intelligent d'aide à l'orientation et à l'insertion professionnelle.

Prédiction de la filière adaptée

Introduction

L'un des objectifs majeurs de ce projet est d'accompagner les étudiants dans leur choix de spécialité après les années préparatoires. Pour cela, nous avons conçu un système intelligent capable de recommander la filière la plus adaptée à chaque étudiant en se basant sur son profil académique, socio-démographique et ses préférences personnelles. Ce chapitre décrit en détail le processus complet mis en œuvre : depuis la génération d'un dataset synthétique, jusqu'au développement du modèle de machine learning, et son intégration via une API accessible à travers une interface web.

4.1 Construction du dataset synthétique

4.1.1 Motivation

En raison de l'inexistence d'un dataset public ou institutionnel recensant les performances des étudiants et leurs choix de spécialité, il a été nécessaire de construire un dataset synthétique réaliste. L'objectif était de simuler un environnement éducatif fidèle aux classes préparatoires intégrées de notre école.

4.1.2 Données simulées

Chaque étudiant est décrit par :

- **24 notes** correspondant aux modules suivis durant les 4 semestres (ex. : Algèbre, Analyse, Informatique, Électronique, Physique, Langues, etc.);
- **6 variables socio-démographiques** : sexe, âge, statut social, mention au baccalauréat, orientation lycée, situation géographique ;
- **6 préférences personnelles** représentant des centres d'intérêt en ingénierie : Programmation, Data/IA, Réseaux, Électronique embarquée, Génie des procédés, Management des SI;
- **La filière cible (Filiere)** : déterminée par une logique métier basée sur les moyennes pondérées des notes et des préférences dominantes.

4.1.3 Logique d'attribution de la filière

La génération du dataset s'est basée sur une logique d'attribution automatique de la filière la mieux adaptée à chaque étudiant synthétique. Cette logique combine deux aspects essentiels : les performances académiques et les centres d'intérêt personnels.

Chaque filière est définie par un vecteur de coefficients pondérant l'importance des différentes matières. Pour chaque étudiant, un score académique est calculé en effectuant un produit scalaire entre ses notes et les coefficients de chaque filière, normalisé par la somme des coefficients. Parallèlement, un score basé sur les intérêts spécifiques de l'étudiant (programmation, réseaux, data, électronique embarquée, génie des procédés, management des SI) est attribué à chaque filière selon une correspondance prédéfinie.

La filière finale est celle qui maximise la somme pondérée (ici 50% académique, 50% centres d'intérêt) de ces deux scores.

Extrait de la fonction d'attribution :

```
def predict_filiere(notes, interests_scores):
    scores = {}
    for filiere in filiere_names:
        coef = coefficients[filiere]
        academic_score = np.dot(notes, coef) / sum(coef)
        interest_score = 0
        if filiere == "IID":
            interest_score = interests_scores[2]
        elif filiere == "GI":
            interest_score = interests_scores[0]
        elif filiere == "GE":
            interest_score = interests_scores[3]
        elif filiere == "GP":
            interest_score = interests_scores[4]
        elif filiere == "IRIC":
            interest_score = interests_scores[1]
        elif filiere == "MGSI":
            interest_score = interests_scores[5]
        total_score = 0.5*academic_score + 0.5*interest_score
        scores[filiere] = total_score
    return max(scores, key=scores.get)
```

Ce mécanisme a permis de générer un jeu de données réaliste et cohérent de 6000 étudiants, comprenant leurs caractéristiques personnelles, leurs notes simulées, leurs intérêts, et la filière correspondant le mieux à leur profil.

4.2 Développement du modèle de prédiction

4.2.1 Prétraitement des données

Avant l’entraînement, les étapes suivantes ont été réalisées :

- Encodage des variables catégorielles avec `get_dummies()`;
- Encodage de la cible `Filiere` avec `LabelEncoder`;
- Séparation en ensembles d’entraînement (80%) et de test (20%), avec stratification pour équilibrer les classes.

4.2.2 Choix de l’algorithme

Après expérimentation de plusieurs modèles (Random Forest, XGBoost, LightGBM), le **CatBoostClassifier** a été retenu pour :

- Sa capacité à gérer efficacement les données catégorielles;
- Son excellente précision sans besoin intensif de tuning;
- Sa rapidité d’entraînement;
- La clarté des probabilités renvoyées par classe.

4.2.3 Évaluation du modèle

L’évaluation du modèle de classification a été réalisée à l’aide de plusieurs métriques, dont l’accuracy globale, le rapport de classification (précision, rappel et F1-score) ainsi que la matrice de confusion. Le modèle CatBoostClassifier a atteint une accuracy de **89,42%** sur l’ensemble de test, ce qui indique une performance globale très satisfaisante.

Matrice de confusion :

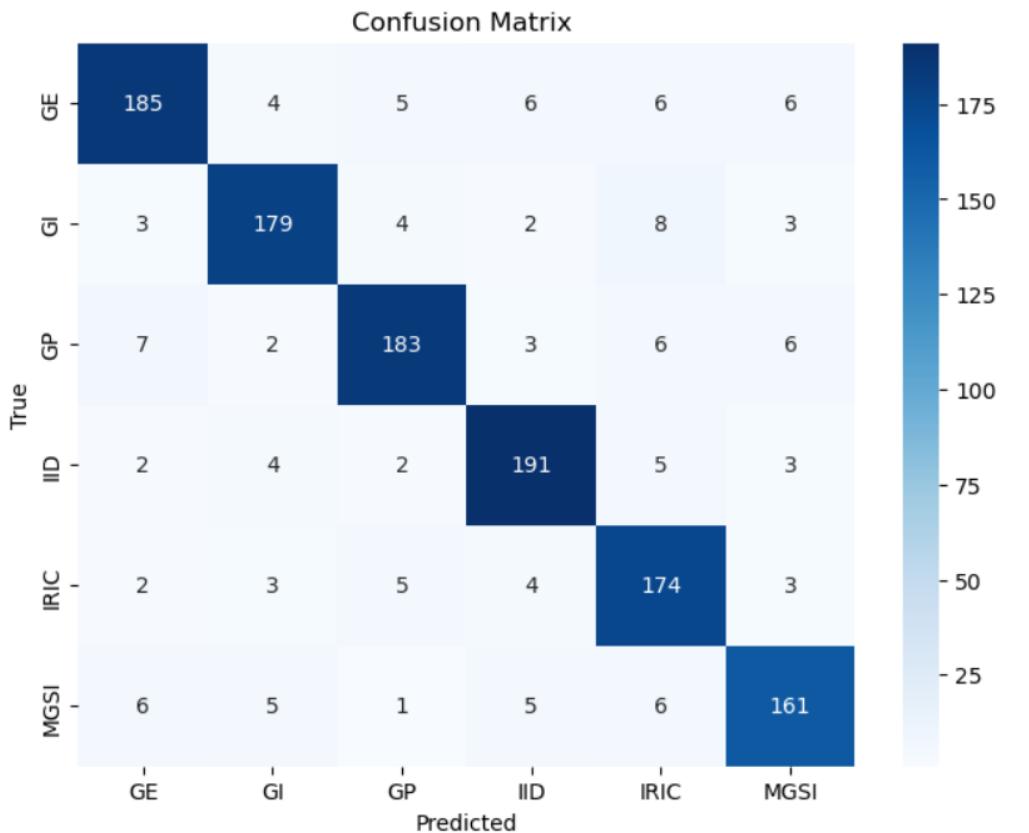


FIGURE 4.1 : Matrice de confusion du modèle CatBoost

Cette matrice permet de visualiser la capacité du modèle à bien classer chaque filière :

- La diagonale principale représente les bonnes prédictions. Par exemple, **191 étudiants** de la filière **IID** ont été correctement classés comme **IID**.
- Les valeurs hors diagonale correspondent aux erreurs de classification. Par exemple, 6 étudiants de la filière **GE** ont été classés comme **MGSI**.
- La majorité des valeurs sont concentrées sur la diagonale, ce qui confirme une bonne séparation des classes par le modèle.
- Certaines confusions récurrentes existent entre des filières proches en compétences, telles que **GE** et **GP**, ou encore **MGSI** et **GI**, probablement en raison d'une proximité dans les profils et les matières pondérées.

La visualisation de la matrice de confusion montre donc que le modèle ne souffre pas d'un biais majeur et qu'il est globalement robuste et équilibré sur l'ensemble des classes.

4.3 Mise en place d'une API et d'une interface web

4.3.1 Création de l'API avec FastAPI

Une API REST a été développée avec le framework **FastAPI**. Cette API :

- reçoit les données d'un étudiant via une requête POST;
- applique les mêmes transformations (encodage, alignement de colonnes) que lors de l'entraînement;
- retourne la filière prédite ainsi que les probabilités associées à chaque filière.

Le middleware CORS a été ajouté pour permettre les appels croisés depuis une application front-end.

4.3.2 Interface utilisateur développée avec Next.js

Afin de rendre le système de recommandation de filière accessible aux étudiants en cycle préparatoire, une interface web a été développée à l'aide du framework **Next.js**. Ce choix permet de combiner les avantages du rendu côté serveur (SSR) avec une navigation fluide et moderne, tout en assurant la scalabilité et la maintenabilité de l'application.

Objectif de l'interface : Fournir aux étudiants une restitution claire, interactive et explicative de la filière recommandée par le modèle d'intelligence artificielle, tout en leur offrant des éléments d'orientation académique et professionnelle.

Informations affichées après la prédiction :

- Filière recommandée
- Pourcentage de compatibilité avec chaque filière
- Points forts identifiés du profil de l'étudiant
- Débouchés professionnels associés à la filière recommandée
- Détails sur la filière recommandée incluant :
 - Aperçu du cycle ingénieur
 - Cours principaux
 - Compétences développées
 - Débouchés professionnels

Design et ergonomie : L'interface est conçue de manière responsive, compatible avec tous les types d'appareils (desktop, mobile, tablette). Elle adopte un design moderne et intuitif facilitant la navigation, l'interprétation des résultats et l'accès aux informations détaillées sur chaque filière.

4.4 Architecture Générale du Système

Pour permettre une intégration fluide entre l'utilisateur final et le moteur de prédiction, nous avons adopté une architecture logicielle basée sur une séparation claire des responsabilités. Cette architecture suit un modèle en trois couches : **interface utilisateur**, **API REST**, et **modèle de prédiction**. Chaque composant joue un rôle précis dans la chaîne de traitement de la prédiction de la filière.

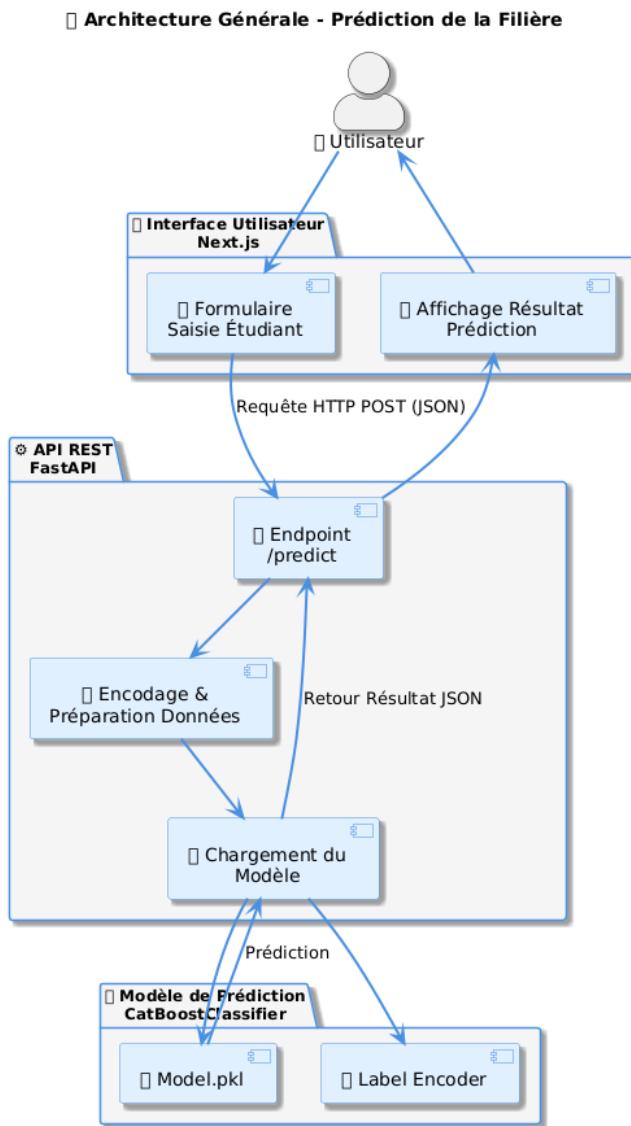


FIGURE 4.2 : Architecture de Sécurité du Système d'Authentification

4.4.1 Description de l'Architecture

L'architecture globale repose sur trois couches :

- **Interface Utilisateur (Next.js)** : Cette couche frontale permet aux étudiants de saisir leurs données (notes, informations personnelles, centres d'intérêt) à travers une interface conviviale et accessible via un navigateur web.
- **API REST (FastAPI)** : Ce backend agit comme un intermédiaire. Il reçoit les données utilisateur, les encode selon le même format que celui utilisé lors de l'entraînement du modèle, et transmet les données au modèle. Il retourne ensuite la filière prédictive et les probabilités associées.
- **Modèle de Prédiction (CatBoostClassifier)** : Ce composant, préalablement entraîné et sauvegardé au format .pkl, effectue la prédiction de la filière la plus adaptée. Il repose sur l'analyse combinée des notes académiques et des centres d'intérêt.

Conclusion

Ce chapitre a présenté l'ensemble du processus de conception d'un système intelligent de recommandation de filière. En l'absence de données réelles, un dataset synthétique a été généré avec des hypothèses pédagogiques robustes. Le modèle CatBoost s'est avéré performant, atteignant une précision de 89.42%. Finalement, l'intégration via une API FastAPI et une interface Next.js permet de mettre à disposition une solution fonctionnelle, ergonomique et extensible. Ce système pourra être enrichi avec des données réelles et s'adapter à d'autres contextes éducatifs.

Développement du chatbot intelligent

Introduction

Dans un contexte où l'accès rapide à l'information devient un levier essentiel pour l'efficacité et l'autonomie des étudiants, la mise en place d'un chatbot intelligent représente une solution innovante. Ce chapitre présente en détail la conception du chatbot GenieusEnsakh, une solution basée sur l'approche Retrieval-Augmented Generation (RAG), spécialement pensée pour répondre aux besoins des étudiants de l'École Nationale des Sciences Appliquées de Khouribga. À travers une architecture modulaire combinant encodage sémantique, recherche vectorielle et génération de texte par un LLM (LLaMA 4 via API GROQ), GenieusEnsakh vise à offrir des réponses fiables et contextuelles. Ce chapitre explore les objectifs du projet, les choix technologiques adoptés, ainsi que le fonctionnement interne du système, de la structuration des données à l'interaction des composants logiciels.

5.1 Objectif du Chatbot

Problématique que le chatbot cherche à résoudre :

De nombreux étudiants se retrouvent confrontés à un manque d'accessibilité ou de clarté concernant certaines informations relatives à leur parcours académique, aux activités extrascolaires ou aux opportunités offertes par l'école. Ils doivent souvent chercher manuellement dans plusieurs documents, poser des questions à l'administration ou attendre des réponses sur les réseaux sociaux internes.

Le chatbot vise à centraliser et automatiser l'accès à ces informations en fournissant des réponses instantanées, claires, et adaptées aux besoins spécifiques des étudiants.

Contexte d'utilisation :

Le chatbot développé s'inscrit dans un contexte académique, au sein de l'École Nationale des Sciences Appliquées de Khouribga (ENSAKH). Il a pour vocation de répondre aux besoins d'information des étudiants en leur offrant un accès rapide, structuré et automatisé à une large base de connaissances.

Deux grandes catégories de questions peuvent être traitées par le chatbot :

1. Questions spécifiques à l'ENSAKH

Ces questions portent sur les aspects institutionnels, académiques et associatifs propres à l'École Nationale des Sciences Appliquées de Khouribga. Le chatbot agit ici comme un assistant dédié à l'environnement ENSAKH, permettant aux étudiants d'obtenir des réponses précises sur les filières, les départements, les clubs et leurs activités, les événements organisés à l'école, les enseignants et leurs informations, les partenariats avec d'autres établissements, ainsi que diverses statistiques clés concernant l'école. Par exemple :

- *Peux-tu me donner des informations sur Dataverse ?*
- *donner une description du club Codex ?*
- *Quels sont les débouchés de la filière Génie Électrique ?*
- *Peux-tu me donner l'e-mail du professeur Lamghari Nidal ?*
- *Quelles sont les écoles partenaires où existe la modalité double diplôme ingénieur-ingénieur ?*
- *Quel est le nombre d'étudiants inscrits à l'ENSAKH ?*

2. Questions générales

Le chatbot peut ainsi répondre à tout type de question générale. Il est capable de traiter des interrogations transversales portant sur les compétences ou les entretiens professionnels. Ces questions dépassent le cadre de l'école et s'inscrivent dans une démarche de développement personnel et professionnel. Par exemple :

- *Quelles sont les questions les plus fréquentes lors d'un entretien en Data Analytics ?*
- *Quelles sont les étapes pour devenir un expert en cybersécurité ?*
- *Quelles sont les compétences nécessaires pour maîtriser le domaine des systèmes embarqués ?*

Grâce à cette double fonctionnalité, le chatbot joue à la fois le rôle d'un assistant académique et d'un conseiller d'orientation numérique, au service de la réussite étudiante.

Public cible :

Le chatbot s'adresse principalement à l'ensemble des étudiants de l'ENSA Khouribga, tous niveaux confondus (du cycle préparatoire au cycle ingénieur) et toutes filières comprises. Il peut également être utile aux nouveaux inscrits souhaitant accéder rapidement à des informations sur l'ENSAKH.

Objectifs fonctionnels :

Les objectifs fonctionnels de ce chatbot sont multiples :

- **Gain de temps** : Fournir instantanément des réponses à des questions fréquemment posées sans intervention humaine.

- **Automatisation de l'accès à l'information** : Permettre aux étudiants d'obtenir des informations vérifiées et structurées sans passer par plusieurs sources disparates.
- **Accessibilité** : Offrir un accès 24/7 à une base de connaissances complète, via une interface simple d'utilisation.
- **Orientation académique et professionnelle** : Accompagner les étudiants dans leurs choix de parcours, la préparation aux entretiens, et le développement de leurs compétences.

Multilinguisme du système

L'un des atouts majeurs du chatbot GenieusEnsakh réside dans sa capacité à comprendre et répondre aux questions dans plusieurs langues, notamment le français, l'anglais et l'arabe. Cette polyvalence linguistique permet à un large éventail d'étudiants d'interagir avec le système dans la langue de leur choix, favorisant ainsi l'inclusion et l'accessibilité.

5.2 Architecture Générale du Système

Le chatbot repose sur une architecture modulaire qui implémente l'approche **RAG (Retrieval-Augmented Generation)**. Cette approche combine la récupération d'information (*retrieval*) depuis une base documentaire avec la génération de texte (*generation*) à l'aide d'un grand modèle de langage (LLM). Cela permet de fournir des réponses précises, contextualisées et informées.

L'architecture est structurée autour des composants suivants :

- **Client / Utilisateur** : L'étudiant interagit avec le chatbot via une interface web intuitive.
- **Front-end** : Une interface web (développée avec **Next.js**) communique avec le backend via des requêtes HTTP.
- **API FastAPI** : Le backend est construit avec **FastAPI**, un framework rapide et léger en Python. Il gère les requêtes, déclenche le processus RAG, et renvoie la réponse au client.
- **SentenceTransformers** : Ce module encode les questions des utilisateurs sous forme de vecteurs denses.
- **Index FAISS** : Utilisé pour rechercher les documents les plus pertinents dans une base de connaissances vectorisée.
- **Base de données documentaire (JSON)** : Contient les documents utilisés pour l'enrichissement des réponses.
- **Modèle LLM (Meta-LLaMA 4 via API GROQ)** : Ce modèle génère la réponse finale en se basant à la fois sur la question et sur les documents récupérés.
- **Réponse générée** : La réponse synthétisée est renvoyée à l'interface web pour affichage à l'utilisateur.

Interaction entre les Composants

La figure ci-dessous illustre le flux global d'interaction entre les différents composants du système basé sur l'approche RAG (Retrieval-Augmented Generation). Chaque étape du traitement, depuis la saisie de la question par l'utilisateur jusqu'à la génération de la réponse finale, est représentée de manière schématique pour faciliter la compréhension du processus.

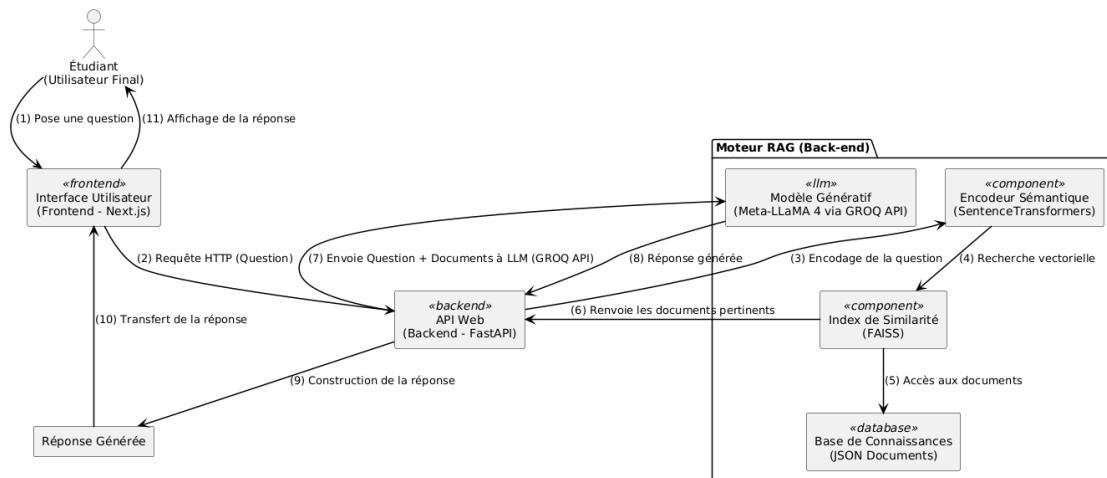


FIGURE 5.1 : Architecture du système basée sur l'approche RAG

Cette architecture modulaire repose sur une série d'étapes numérotées :

1. **L'étudiant pose une question** via l'interface web (développée avec Next.js).
2. **La requête est transmise** au backend via une API REST construite avec FastAPI.
3. **La question est encodée** sous forme de vecteur sémantique à l'aide du modèle *SentenceTransformers*.
4. **Une recherche vectorielle** est effectuée dans l'index *FAISS*.
5. **L'index interroge la base documentaire JSON** afin de retrouver les documents les plus pertinents.
6. **Les documents pertinents sont renvoyés** au backend.
7. **Le backend appelle le modèle LLM (Meta-LLaMA 4)** via l'API GROQ, en fournissant la question initiale et les documents retrouvés.
8. **Le modèle génère une réponse** contextuelle et informative.
9. **Le backend construit et formate la réponse**.
10. **La réponse est transmise** au frontend.
11. **La réponse est affichée** à l'utilisateur sur l'interface web.

Cette approche permet d'obtenir des réponses précises, contextualisées et fondées sur une base documentaire fiable tout en exploitant la puissance d'un grand modèle de langage.

5.3 Importation des Bibliothèques

Dans cette section, les bibliothèques nécessaires au bon fonctionnement du chatbot sont importées. Elles couvrent divers aspects du système, depuis le traitement du langage naturel jusqu'à la création d'API web :

- **sentence_transformers** : pour l'encodage sémantique des questions en vecteurs.
- **faiss** : pour l'indexation et la recherche rapide de similarité entre vecteurs.
- **numpy** : pour les manipulations de données numériques.
- **json, re, unicodedata, difflib** : pour la gestion de données, le nettoyage de texte, et la comparaison de chaînes.
- **requests** : pour l'envoi de requêtes HTTP à l'API du LLM.
- **fastapi, pydantic, typing** : pour la construction d'une API web rapide, la validation des données, et la gestion des types.
- **fastapi.middleware.cors** : pour permettre les échanges entre le front-end (Next.js) et le backend.

5.4 Préparation et Structuration des Données

Sources de données

Le chatbot s'appuie principalement sur des fichiers .json regroupant les données spécifiques à l'École Nationale des Sciences Appliquées de Khouribga (ENSA Khouribga). Ces données proviennent en grande partie du site web officiel <http://ensak.usms.ac.ma>, enrichies manuellement dans les cas où certaines informations n'étaient pas disponibles en ligne.

- `event_ENSAKH.json` : événements organisés par l'école,
- `chiffres_ENSAKH.json` : statistiques clés de l'établissement,
- `clubs_ENSAKH.json` : liste des clubs et associations étudiants,
- `ecoles_modalites_DD_ENSAKH.json` : écoles partenaires et leurs modalités,
- `enseignants_ENSAKH.json` : informations sur les enseignants,
- `departement_ENSAKH.json` : informations sur les départements pédagogiques,
- `filières_ENSAKH.json` : informations sur les filières disponibles.

Pour récupérer ces données, des scripts de scraping ont été développés. Par exemple, le script ci-dessous permet d'extraire automatiquement les informations des départements depuis le site de l'ENSAK :

```

import requests
from bs4 import BeautifulSoup
import json

def get_department_links(BASE_URL="http://ensak.usms.ac.ma"):
    url = f"{BASE_URL}/ensak/departements/"
    response = requests.get(url)
    soup = BeautifulSoup(response.content, "html.parser")
    departments = []

    divs = soup.select("div.col-sm-6")
    for div in divs:
        a_tag = div.find("a")
        if a_tag:
            name = a_tag.find("button").text.strip() if
            a_tag.find("button") else "Nom non trouvé"
            link = a_tag["href"]
            if not link.startswith("http"):
                link = BASE_URL + link
            departments.append({"Nom": name, "Lien": link})
    return departments

```

Chargement des données

Les fichiers .json obtenus sont ensuite chargés et structurés pour permettre l'indexation vectorielle et la recherche sémantique. Le chargement se fait via une fonction `load_json_file()`, qui lit le contenu du fichier et transforme chaque entrée en un format standardisé avec les clés : `type`, `title`, `content`, et éventuellement `raw_data`.

Voici un extrait de code utilisé pour charger les données des départements :

```

load_json_file("departement_ENSAKH.json", lambda depts: sources.extend([
    {
        "type": "Département",
        "title": dept.get("Nom Département", "Nom inconnu"),
        "content": (
            f"Département : {dept.get('Nom Département', 'Nom inconnu')}\n"
            f"Chef de département : {dept.get('Chef', 'N'existe pas')}\n"
            f"Adjoint : {dept.get('Adjoint', 'N'existe pas')}\n\n"
            f"Filières proposées :\n" +
            "\n".join([f"* {f.strip()}" for f in dept.get("Filières", [])]) +
            "\n\nListe des professeurs :\n" +
            "\n".join([f"* {prof['Nom']} (Email: {prof['Email']})" +
                      for prof in dept.get("Professeurs", [])])
    )
}
for dept in depts

```

```
]) )
```

Ce formatage permet d'unifier la structure des contenus pour qu'ils soient directement exploitable dans la phase d'indexation et de recherche, assurant ainsi une cohérence dans les réponses du chatbot.

5.5 Indexation intelligente avec le traitement du langage naturel (NLP)

Pour permettre une recherche efficace et pertinente parmi les données textuelles relatives à l'ENSA Khouribga, deux techniques fondamentales de traitement du langage naturel (NLP) ont été utilisées : l'encodage sémantique et la recherche vectorielle.

a. Encodage sémantique avec SentenceTransformer

La première étape consiste à transformer les descriptions textuelles (`content`) en représentations numériques appelées **vecteurs d'embedding**. Ces vecteurs capturent la signification des textes, permettant de mesurer la similarité entre phrases sur la base de leur sens plutôt que de simples mots-clés.

Le modèle utilisé est `paraphrase-multilingual-MiniLM-L12-v2`, un encodeur multilingue performant, proposé par la bibliothèque `sentence-transformers`. Il permet de générer des embeddings denses et compacts adaptés à la recherche sémantique.

Extrait de code Python :

```
embedding_model = SentenceTransformer(
    "sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2"
)
text_embeddings = embedding_model.encode(texts, show_progress_bar=True)
```

b. Recherche vectorielle avec FAISS

Après l'encodage des textes, les vecteurs sont indexés dans une structure optimisée pour la recherche rapide : FAISS (Facebook AI Similarity Search). Cette bibliothèque est conçue pour effectuer des recherches efficaces dans de grands ensembles de vecteurs en haute dimension.

Ici, l'index `IndexFlatL2` est utilisé. Il permet une recherche exacte basée sur la distance euclidienne (L^2) entre les vecteurs.

Extrait de code Python :

```
import faiss
import numpy as np

dimension = text_embeddings[0].shape[0]
index = faiss.IndexFlatL2(dimension)
index.add(np.array(text_embeddings))
```

Ainsi, lorsqu'un utilisateur saisit une requête, celle-ci est encodée de la même manière, puis comparée à l'ensemble des vecteurs indexés pour retrouver les contenus les plus sémantiquement pertinents.

5.6 Recherche contextuelle et extraction ciblée

La phase de recherche permet au chatbot de récupérer les données pertinentes en fonction de la question posée par l'utilisateur. Cette étape repose sur deux composantes principales : la recherche sémantique contextuelle à l'aide d'index FAISS, et l'extraction ciblée par filtre selon des critères précis.

5.6.1 Recherche sémantique (retrieval contextuel)

Lorsqu'un utilisateur soumet une requête, celle-ci est encodée par le même modèle de transformation de phrases utilisé lors de l'indexation (paraphrase-multilingual-MiniLM-L12-v2). Ce vecteur est ensuite comparé à l'index FAISS pour identifier les descriptions les plus proches sur le plan sémantique. Les contenus les plus pertinents sont alors extraits et proposés comme contexte à la réponse.

```
def retrieve_context_faiss(user_input, top_k=4):
    query_embedding = embedding_model.encode([user_input])
    D, I = index.search(np.array(query_embedding), top_k)
    top_contexts = []
    for idx in I[0]:
        if idx < len(data_sources):
            src = data_sources[idx]
            top_contexts.append(f'{src["type"]} - {src["title"]}\n{src["content"]}')
    return '\n\n'.join(top_contexts)
```

Cette fonction encode la question utilisateur, recherche les `top_k` documents les plus similaires, puis assemble un contexte pertinent à partir des contenus indexés.

5.6.2 Recherche filtrée par type

Outre la recherche globale, le système permet de filtrer les données par catégorie ou critère spécifique, afin de répondre plus précisément à certaines demandes structurées (ex. : "Quels sont les débouchés de la filière Génie Informatique ?").

Voici un exemple de fonction de filtrage selon un critère appliqué aux filières :

```
def filter_filières_by_criteria(criteria):
    """Filtre les filières selon différents critères"""
    results = []

    for src in data_sources:
        if src["type"] == "Filière":
```

```

        section_content = extract_specific_section(src, criteria)
        if section_content and "aucun" not in section_content.lower():
            results.append(section_content + "\n")

    if results:
        return "\n".join(results)
    else:
        return f"Aucune information trouvée pour le critère : {criteria}"

```

Cette approche permet une extraction ciblée de sections spécifiques telles que les compétences, débouchés ou prérequis d'une filière, tout en évitant le bruit ou les réponses génériques.

5.7 Génération de réponse intelligente avec LLaMA 4 (via GROQ API)

Une fois le contexte pertinent extrait par la recherche sémantique, le backend envoie une requête à l'intelligence artificielle générative afin de produire une réponse à la fois naturelle, fluide et pertinente. Cette étape utilise le modèle LLaMA de Meta via l'API **GROQ**.

a. API utilisée : GROQ

Pour communiquer avec LLaMA, on utilise l'API de GROQ, configurée de la manière suivante :

- **Clé API** : API_KEY = "votre_clé_API"
- **URL** : API_URL = "https://api.groq.com/openai/v1/chat/completions"
- **Modèle** : MODEL = "meta-llama/llama-4-scout-17b-16e-instruct"

b. Exemple de requête à l'API avec *system_prompt*

Avant d'appeler l'API, on définit un message système ('system_prompt') qui guide le modèle : il l'incite à utiliser prioritairement les informations relatives à l'ENSA Khouribga, tout en restant capable de répondre à toute autre question généraliste.

```

system_prompt = (
    "Tu es un assistant intelligent spécialisé dans les informations concernant \""
    "l'École Nationale des Sciences Appliquées de Khouribga (ENSAKH).\\n\""
    "Utilise uniquement les informations suivantes :\\n\\n"
    f"{context}\\n\\n"
    "Réponds à la question de l'utilisateur de manière claire, concise et"
    "uniquement "
    "basée sur les descriptions ci-dessus."
)

```

Le backend assemble ensuite les messages de conversation (message système + historique + question utilisateur), puis envoie la requête :

```

headers = {
    "Content-Type": "application/json",
    "Authorization": f"Bearer {API_KEY}"
}

payload = {
    "model": MODEL,
    "messages": [
        {"role": "system", "content": system_prompt},
        *history,
        {"role": "user", "content": user_input}
    ]
}

response = requests.post(API_URL, headers=headers, json=payload)
response.raise_for_status()
reply = response.json()["choices"][0]["message"]["content"].strip()

```

Grâce à ce ‘system_prompt’, le modèle est d’abord orienté vers les données ENSAKH, mais sans être restreint exclusivement à ce domaine : s’il ne trouve pas de réponse dans le contexte fourni, il peut néanmoins mobiliser ses connaissances générales pour répondre de manière pertinente à d’autres questions.

5. Déploiement via une API web (FastAPI)

a. Backend FastAPI

Une API REST est mise en place avec FastAPI pour :

- recevoir la question de l’utilisateur,
- exécuter les recherches (FAISS ou filtrées),
- appeler l’IA GROQ,
- retourner la réponse.

b. CORS activé

Pour permettre l’appel depuis un frontend , le CORS (*Cross-Origin Resource Sharing*) est activé pour toutes les origines à l’aide du middleware suivant :

```

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"], # Pour tous les domaines
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=[ "*"],
)

```

9. Évaluation des performances

L'évaluation des performances de l'API a été réalisée à travers l'analyse des temps de réponse pour les requêtes envoyées à l'endpoint /chatbot. Voici un extrait des logs illustrant ces temps :

```
GET /chatbot 200 in 180ms
GET /chatbot 200 in 181ms
GET /chatbot 200 in 63ms
GET /chatbot 200 in 61ms
GET /chatbot 200 in 174ms
GET /chatbot 200 in 108ms
GET /chatbot 200 in 92ms
```

Chaque ligne indique que l'API a traité une requête GET avec succès (code 200) et dans un délai compris entre **61 ms et 181 ms**. Ces résultats montrent que l'application offre une bonne réactivité et que le temps de génération de réponse, incluant l'accès au contexte et l'appel à l'IA via l'API GROQ, reste raisonnablement bas.

Ce suivi permet d'assurer une expérience fluide pour l'utilisateur et de détecter d'éventuelles latences en cas de surcharge ou d'erreurs.

Conclusion

La mise en œuvre du chatbot ENSAKH illustre la capacité des technologies de pointe en traitement du langage naturel à répondre à des problématiques concrètes dans un cadre académique. Grâce à l'approche RAG, il devient possible de coupler une base de connaissances locale (liée à l'ENSAKH) à la puissance d'un modèle génératif de dernière génération, pour fournir des réponses à la fois précises et adaptées. L'architecture développée démontre une intégration cohérente entre collecte de données, indexation intelligente et génération de contenu, tout en restant évolutive et maintenable. Ce chatbot constitue ainsi une base solide pour d'éventuelles extensions futures, comme l'élargissement du corpus, la prise en charge de la voix ou encore la personnalisation par profil utilisateur.

Technologies et Outils Utilisés

Introduction

Le développement d'une application moderne, performante et adaptée aux besoins des utilisateurs nécessite le recours à un ensemble cohérent de technologies. Ce chapitre présente les outils et frameworks sélectionnés pour concevoir notre solution, depuis l'environnement de développement jusqu'aux techniques d'analyse intelligente. Le choix de chaque technologie a été motivé par sa robustesse, sa compatibilité avec l'architecture globale du projet, et sa capacité à répondre efficacement aux exigences fonctionnelles et techniques.

6.1 Outils de développement

Visual Studio Code



FIGURE 6.1 : Visual Studio Code

Description : Éditeur de code open-source très populaire.

Pourquoi ce choix : Pour son interface intuitive, ses extensions puissantes et sa compatibilité multi-technologies.

GitHub

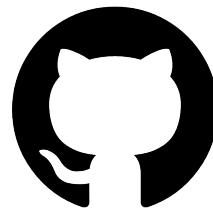


FIGURE 6.2 : GitHub

Description : Plateforme de versionnage et de collaboration basée sur Git.

Pourquoi ce choix : Pour gérer efficacement le code source, les versions et le travail collaboratif.

6.2 Base de données

PostgreSQL

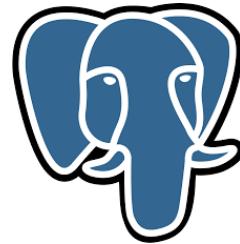


FIGURE 6.3 : PostgreSQL

Description : Base de données relationnelle open-source, robuste et extensible.

Pourquoi ce choix : Très performante pour le stockage sécurisé de données utilisateurs et d'analyse.

6.3 Backend

FastAPI

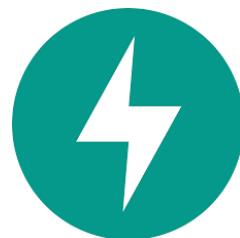


FIGURE 6.4 : FastAPI

Description : Framework web Python moderne conçu pour développer des APIs rapides et typées.

Pourquoi ce choix : Rapide, conforme aux standards OpenAPI, idéal pour créer une API REST performante et bien documentée.

Asyncpg

Description : Pilote asynchrone performant pour PostgreSQL.

Pourquoi ce choix : Optimise les performances lors des requêtes concurrentes vers la base de données.

Pydantic



FIGURE 6.5 : Pydantic

Description : Librairie Python pour la validation de données à l'aide des types.

Pourquoi ce choix : Garantit la validité des données transmises entre l'API et le frontend.

CORSMiddleware

Description : Middleware FastAPI permettant de gérer les politiques CORS.

Pourquoi ce choix : Pour autoriser les requêtes HTTP entre le frontend (Next.js) et l'API FastAPI sans restrictions.

JWT (JSON Web Token)



FIGURE 6.6 : JWT (JSON Web Token)

Description : JWT est une méthode d'authentification sécurisée utilisant des tokens portables et signés.

Pourquoi ce choix : Permet de gérer des sessions utilisateurs sans stocker d'état côté serveur, tout en garantissant l'intégrité du token.

Passlib

Description : Bibliothèque Python dédiée au hachage sécurisé des mots de passe.

Pourquoi ce choix : Garantit la sécurité des mots de passe stockés et facilite leur vérification lors de l'authentification.

pdfplumber & python-docx

Description : Extraction de texte depuis des fichiers PDF ou Word.

Pourquoi ce choix : Pour analyser automatiquement les CVs des utilisateurs.

6.4 Frontend

React

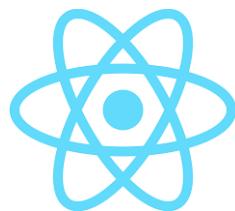


FIGURE 6.7 : React

Description : Bibliothèque JavaScript pour la création d'interfaces interactives.

Pourquoi ce choix : Pour concevoir une interface utilisateur moderne et modulaire.

Next.js



FIGURE 6.8 : Next.js

Description : Framework React pour le rendu côté serveur (SSR) et le SEO.

Pourquoi ce choix : Pour un rendu rapide, dynamique et optimisé pour les moteurs de recherche.

Tailwind CSS



FIGURE 6.9 : Tailwind CSS

Description : Framework CSS utilitaire.

Pourquoi ce choix : Pour un design épuré, personnalisable et rapide à développer.

Lucide React



FIGURE 6.10 : Lucide React

Description : Librairie d'icônes open-source intégrable à React.

Pourquoi ce choix : Pour enrichir visuellement l'interface utilisateur avec des icônes modernes.

6.5 Web Scraping

Selenium



FIGURE 6.11 : Selenium

Description : Selenium est un outil d'automatisation permettant de simuler les interactions d'un utilisateur avec un navigateur web (clics, saisies, navigation, etc.).

Pourquoi ce choix : Il facilite le contrôle du navigateur pour automatiser la collecte d'informations sur des sites nécessitant une interaction dynamique.

undetected_chromedriver

Description : Variante de ChromeDriver conçue pour contourner les systèmes de détection anti-bot mis en place sur certains sites web.

Pourquoi ce choix : Pour éviter les blocages lors du scraping sur des plateformes protégées.

BeautifulSoup



FIGURE 6.12 : BeautifulSoup

Description : Bibliothèque Python de parsing HTML/XML, permettant d'extraire et structurer les données contenues dans les pages web.

Pourquoi ce choix : Pour structurer et extraire facilement les informations pertinentes à partir du HTML récupéré avec `requests..`

6.6 Analyse, IA et Recommandation

SentenceTransformers

Description : Modèles de type BERT utilisés pour transformer des textes en vecteurs sémantiques.

Pourquoi ce choix : Utilisés dans le chatbot pour évaluer la similarité sémantique entre les questions des utilisateurs et les réponses disponibles, garantissant ainsi une réponse pertinente. .

FAISS



FIGURE 6.13 : FAISS

Description : Librairie développée par Facebook pour la recherche rapide de similarité entre vecteurs.

Pourquoi ce choix : Utilisée dans le chatbot pour retrouver rapidement, parmi plusieurs documents vectorisés, les réponses les plus pertinentes à une question posée.

scikit-learn



FIGURE 6.14 : Scikit-learn

Description : Outils de machine learning classiques : TF-IDF, similarité cosinus, etc.

Pourquoi ce choix : Pour extraire et comparer des représentations vectorielles simples de textes.

CatBoost



FIGURE 6.15 : CatBoost

Description : Algorithme de boosting performant, spécialement adapté aux données catégorielles.

Pourquoi ce choix : Utilisé pour prédire la filière la plus adaptée à un étudiant en fonction de ses préférences, ses notes et de son profil académique.

Groq API (Meta LLaMA 4)



FIGURE 6.16 : Groq API (Meta LLaMA 4)

Description : API permettant d'interroger des modèles de langage avancés comme LLaMA 4.

Pourquoi ce choix : Intégrée pour deux usages principaux : l'analyse sémantique des CV afin d'améliorer la pertinence des recommandations, et la communication avec le modèle LLaMA dans le cadre du fonctionnement du chatbot intelligent.

RAG (Retrieval-Augmented Generation)

Description : Architecture combinant la recherche de documents et la génération via LLM.

Pourquoi ce choix : Pour enrichir la compréhension du modèle en l'orientant avec un contexte pertinent.

Conclusion

L'écosystème technologique mis en place a permis d'assurer une application cohérente, réactive et sécurisée, tant sur le plan fonctionnel que technique. L'intégration de technologies modernes comme FastAPI, React ou encore SentenceTransformers a facilité le développement d'une solution à la fois robuste et intelligente. L'approche modulaire adoptée garantit une maintenabilité et une évolutivité du système, tout en assurant une expérience utilisateur fluide. Ainsi, le choix judicieux de ces outils constitue un fondement essentiel à la réussite du projet.

Interface graphique

Introduction :

L'interface graphique constitue un élément central dans l'expérience utilisateur d'une application web. Elle doit à la fois être intuitive, fluide, responsive et visuellement cohérente avec les besoins des utilisateurs cibles. Dans le cadre de ce projet, une attention particulière a été portée à la conception des différentes pages afin de rendre la navigation simple et agréable, tout en facilitant l'accès aux fonctionnalités principales.

Ce chapitre présente de manière détaillée les différents écrans qui composent l'application, depuis la page d'accueil jusqu'aux modules spécifiques tels que l'écran de prédition de filière ou encore le chatbot. Chaque section met en lumière les composants visuels, les choix de navigation ainsi que les mesures de sécurité mises en place pour encadrer l'accès aux données.

7.1 Page d'Accueil

La page d'accueil joue un rôle central dans l'orientation de l'utilisateur. Elle combine un slogan accrocheur, des boutons d'accès rapide aux fonctionnalités clés, des éléments de preuve sociale (témoignages et chiffres), et une explication simplifiée du fonctionnement de l'application. Ce design vise à offrir une navigation fluide et à inciter l'engagement dès le premier contact avec la plateforme.

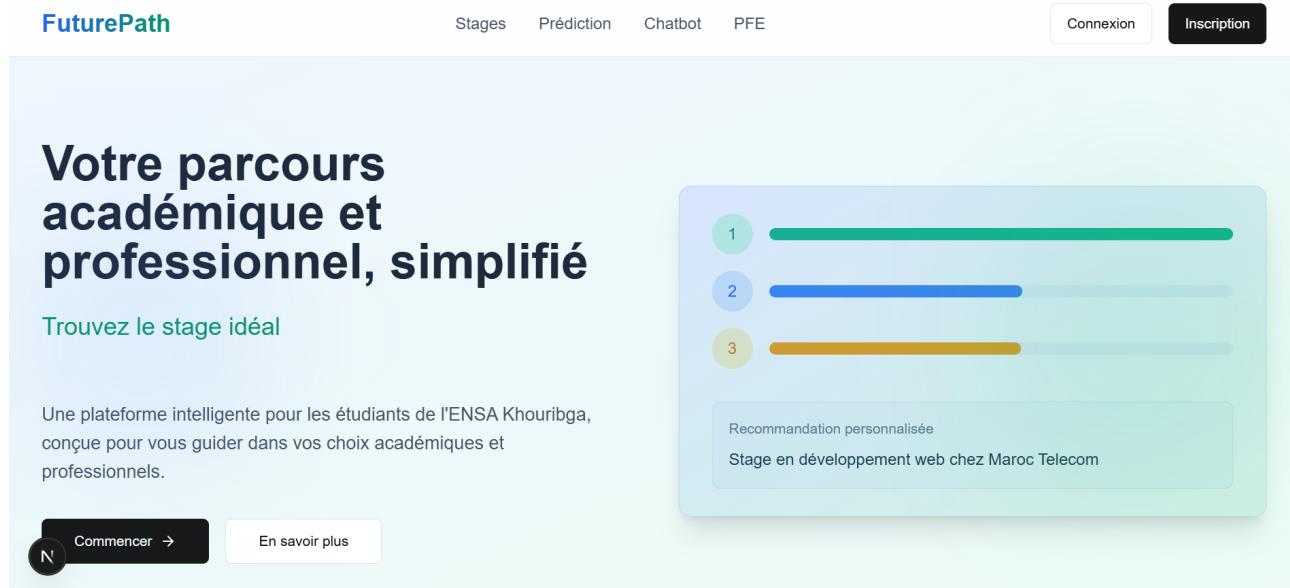


FIGURE 7.1 : Interface visuelle de la page-d'accueil de l'application - partie 1

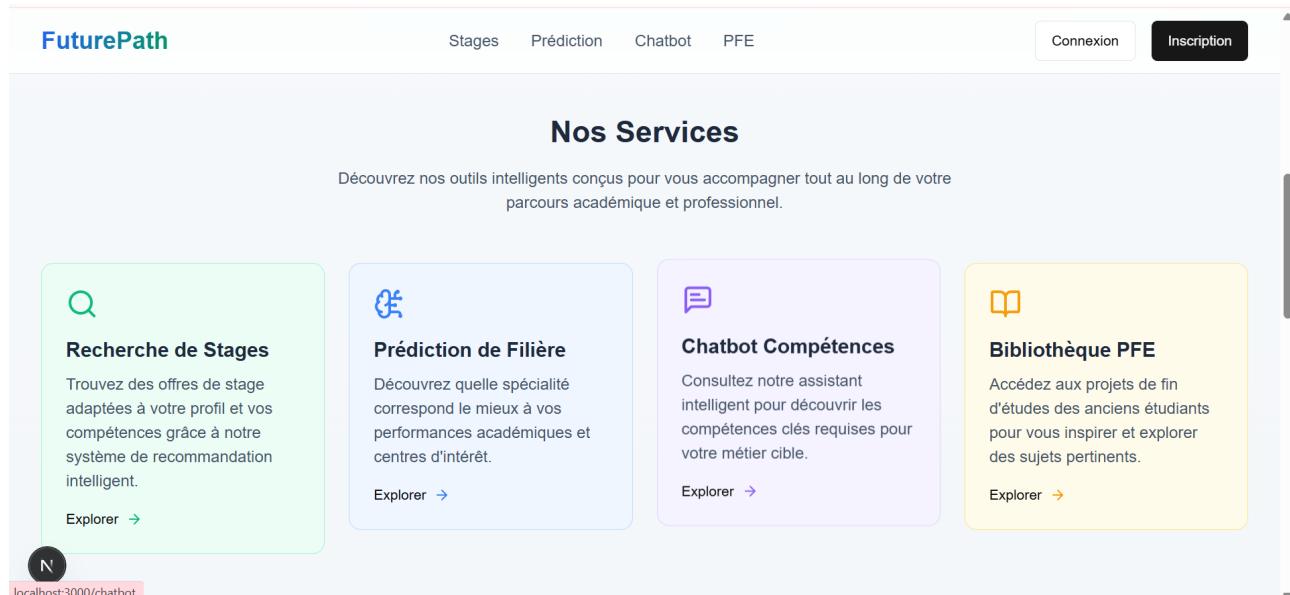


FIGURE 7.2 : Interface visuelle de la page-d'accueil de l'application - partie 2

The screenshot shows the FuturePath application's home page. At the top, there is a navigation bar with the logo "FuturePath" on the left, followed by links for "Stages", "Prédiction", "Chatbot", and "PFE". On the right side of the navigation bar are two buttons: "Connexion" and "Inscription". Below the navigation bar, there is a section titled "À propos de notre plateforme" (About our platform) which contains a brief description of the platform's purpose and a "En savoir plus" button. To the right of this section is a "Témoignages" (Testimonials) section featuring two quotes from students: Fatima and Ahmed. At the bottom of the page is a footer section for "NextStep" containing social media icons, a copyright notice, and links to "SERVICES" and "LIENS UTILES".

À propos de notre plateforme

Notre plateforme a été développée pour répondre aux besoins spécifiques des étudiants de l'ENSA Khouribga. Grâce à l'intelligence artificielle et aux modèles de langage avancés, nous offrons des solutions innovantes pour faciliter votre parcours académique et professionnel.

En savoir plus

Témoignages

"Grâce à cette plateforme, j'ai trouvé un stage parfaitement adapté à mes compétences et à mes aspirations professionnelles."
— Fatima, étudiante en 4ème année

"Le système de prédiction de filière m'a aidé à faire un choix éclairé pour ma spécialisation. Je suis très satisfait de mon parcours."
— Ahmed, étudiant en 3ème année

NextStep

Une plateforme intelligente pour accompagner les étudiants de l'ENSA Khouribga dans leur parcours académique et professionnel.

f g o in

SERVICES

- Recherche de stages
- Prédiction de filière
- Chatbot compétences
- Bibliothèque PFE

LIENS UTILES

- À propos
- Contact
- FAQ
- ENSA Khouribga

© 2025 NextStep. Tous droits réservés.

Politique de confidentialité Conditions d'utilisation

FIGURE 7.3 : Interface visuelle de la page-d'accueil de l'application - partie 3

7.2 Écran d'Authentification

7.2.1 Écran d'Inscription (Formulaire complet)

L'écran d'inscription permet aux nouveaux utilisateurs de créer un compte sur la plateforme. Il comprend les champs nécessaires à la création d'un profil utilisateur : nom complet, adresse email, mot de passe et confirmation du mot de passe. Le bouton « Créer un compte » déclenche la validation des données saisies. En cas d'erreur (champ vide, format invalide, mot de passe non confirmé), des messages d'alerte s'affichent sous les champs concernés pour guider l'utilisateur dans la correction de ses informations.

The screenshot shows a registration form titled "Welcome back" with fields for "Username or Email" (containing "ouaimma") and "Password". Below the password field is a red error message: "The password must be at least 6 characters long and contain both uppercase and lowercase letters". A "Sign up" button is present. To the right is a "Future Path" logo and social media links.

FIGURE 7.4 : Formulaire d'inscription avec validation des champs

7.2.2 Écran de Connexion (Formulaire standard)

L'écran de connexion offre un accès rapide et sécurisé à l'espace personnel de l'utilisateur. Il comporte deux champs : l'email et le mot de passe. Un lien « Mot de passe oublié » est disponible pour déclencher la procédure de récupération d'accès en cas d'oubli. En cas de saisie incorrecte ou de tentative échouée, un message d'erreur s'affiche (par exemple : « Email ou mot de passe incorrect »). Une fois la connexion réussie, l'utilisateur est redirigé vers la page d'accueil ou son tableau de bord personnalisé.

The screenshot shows a registration form titled "Create your account" with fields for "Username" (containing "ouaimma"), "Email" (containing "Oumaima.elalam12@usms.ma"), and "Password". Below the password field is a red error message: "The password must be at least 6 characters long and contain both uppercase and lowercase letters". A "Sign up" button is present. To the right is a "Future Path" logo and social media links. At the bottom, there is a link "Forgot your password?" which is underlined.

FIGURE 7.5 : Formulaire de connexion avec lien vers la récupération de mot de passe

7.2.3 Sécurité des Routes – Accès Non Autorisé

Cette série de captures illustre le comportement du système lorsqu'un utilisateur non authentifié tente d'accéder à différentes fonctionnalités protégées. Dans chaque cas, la requête est interceptée et l'utilisateur est automatiquement redirigé vers la page de connexion (login), avec un paramètre indiquant la page d'origine. Cette mesure garantit que seules les personnes authentifiées peuvent accéder aux ressources sécurisées, renforçant ainsi la sécurité globale de l'application.



FIGURE 7.6 : Redirection vers /login?redirect=/stages lorsque l'utilisateur tente d'accéder à la page des stages sans être connecté.



FIGURE 7.7 : Redirection vers /login?redirect=/prediction lors d'un accès non autorisé à la page de prédition de filière.



FIGURE 7.8 : Redirection vers /login?redirect=/chatbot pour un accès non autorisé à l'écran chatbot.

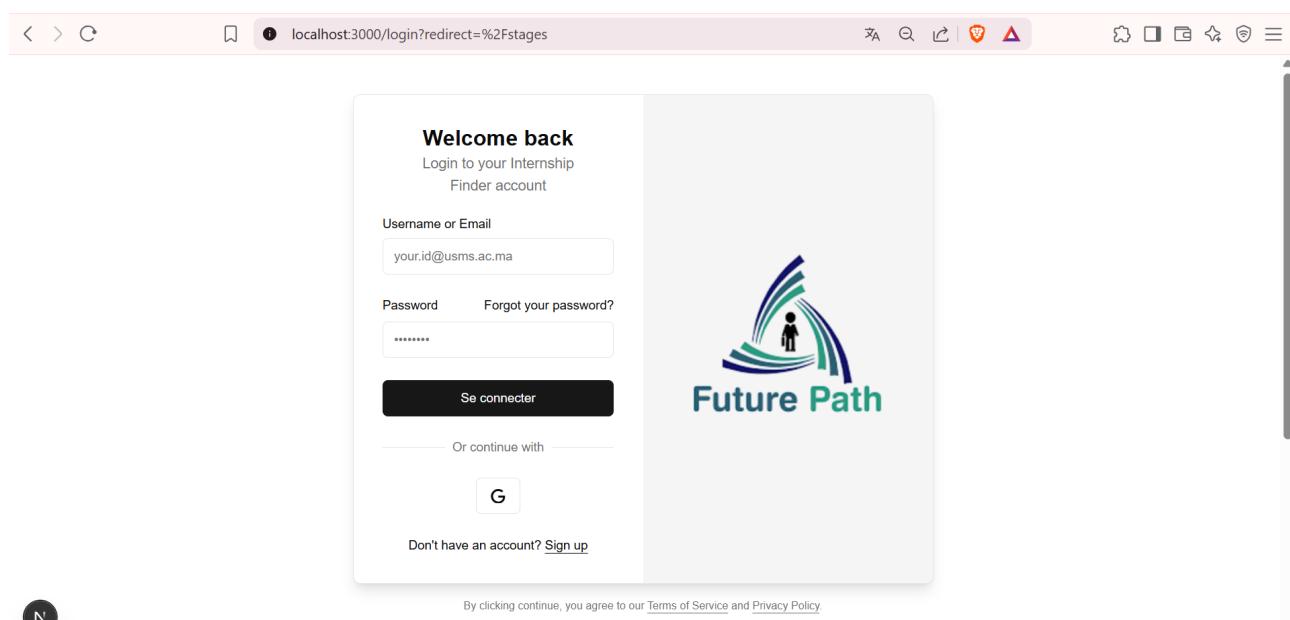


FIGURE 7.9 : Page de connexion affichée après redirection automatique en cas d'accès non autorisé.

7.3 Accueil Stages

La page d'accueil des stages présente à l'utilisateur une interface simple et intuitive pour découvrir des offres adaptées à son profil. Grâce à l'analyse automatisée du CV et à la prise en compte des préférences personnelles, la plateforme propose des recommandations personnalisées. Le parcours utilisateur est guidé en trois étapes claires : téléchargement du CV, définition des préférences, puis consultation des offres recommandées. Des boutons d'action facilitent l'accès aux fonctionnalités clés, garantissant une expérience fluide et efficace.

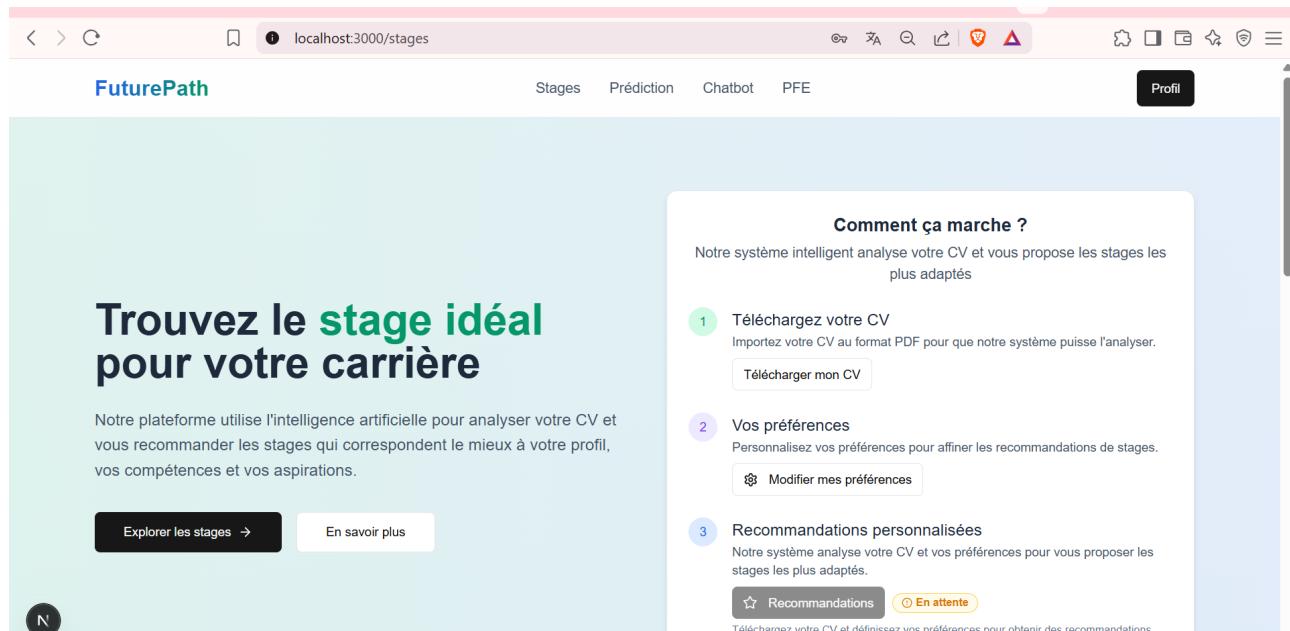


FIGURE 7.10 : Interface principale de la page d'accueil des stages avec les différentes étapes du parcours utilisateur

7.3.1 Écran d'Upload et d'Analyse du CV

Cet écran permet à l'utilisateur de télécharger son CV au format PDF afin d'extraire automatiquement le domaine principal et les mots-clés associés. Ces informations sont affichées pour validation et modification avant sauvegarde. L'utilisateur peut également lancer un scraping avancé des offres correspondant à son profil pour enrichir les recommandations. Un bouton de sauvegarde permet de finaliser cette étape avant de passer à la suite.

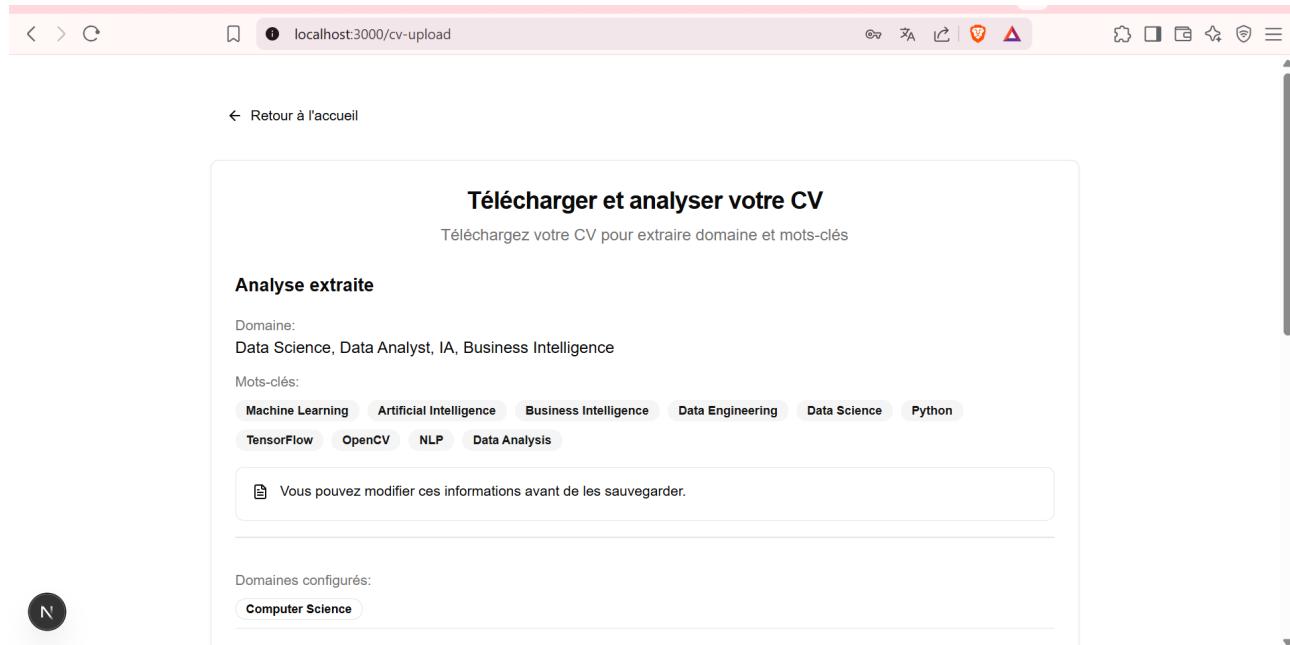


FIGURE 7.11 : Interface de téléchargement et d'analyse du CV avec modification des domaines et mots-clés extraits - partie 1

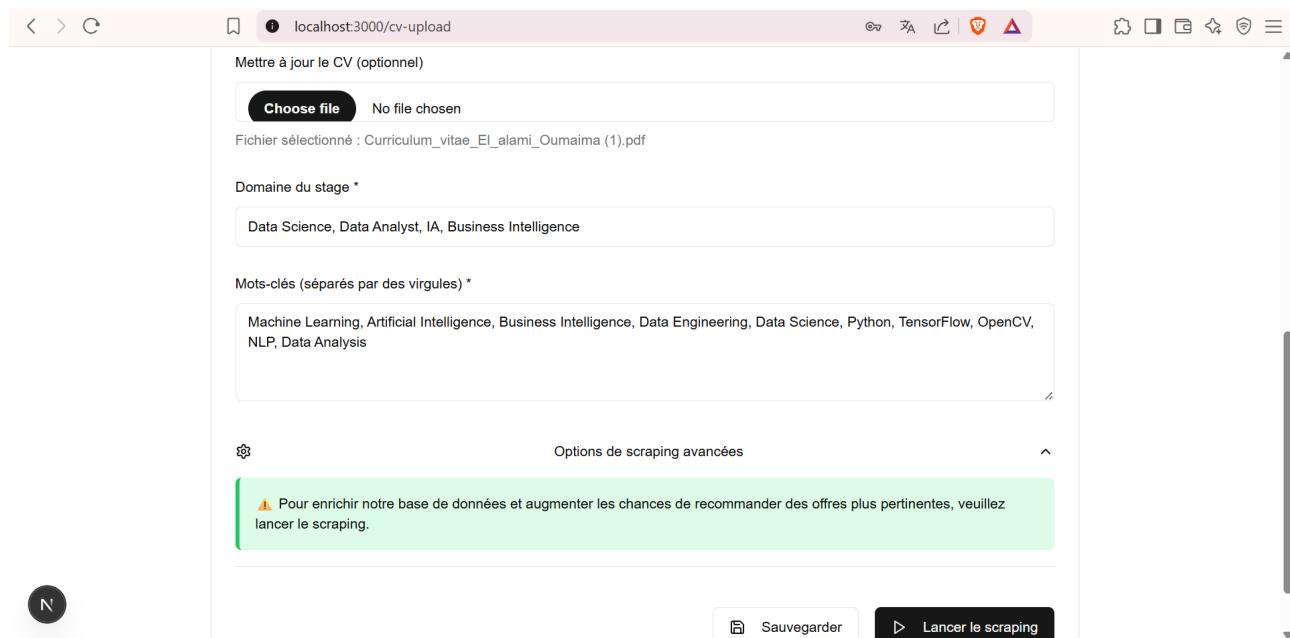


FIGURE 7.12 : Interface de téléchargement et d'analyse du CV avec modification des domaines et mots-clés extraits - partie 2

7.3.2 Écran de Préférences

Cet écran permet à l'utilisateur de régler l'importance des critères utilisés pour les recommandations de stages. Il peut ajuster la pondération des critères comme le domaine, les compétences, le titre et la description, ainsi que ses préférences géographiques (pays) et sources d'offres (LinkedIn, Indeed, Glassdoor).

Des boutons sont disponibles pour réinitialiser, annuler ou sauvegarder les modifications, offrant un contrôle simple et efficace pour personnaliser les résultats.

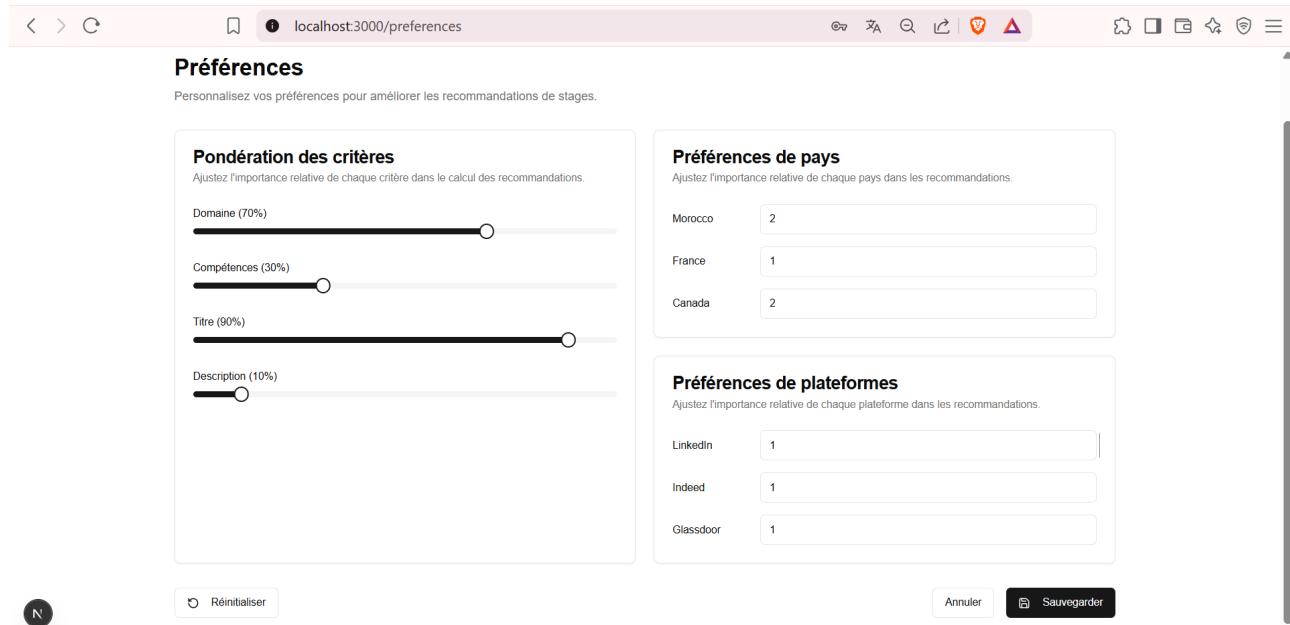


FIGURE 7.13 : Interface de personnalisation des préférences de recommandation

7.3.3 Écran de Recommandations de Stages

Cet écran affiche des offres de stage recommandées en fonction du profil et des compétences de l'utilisateur. Il propose également des filtres (mots-clés, localisation, plateformes, statut) pour affiner la recherche. Chaque offre présente des informations détaillées comme le titre, l'entreprise, le lieu, la durée et la description.

L'utilisateur peut consulter, sauvegarder ou marquer les offres comme vues, facilitant ainsi la gestion des opportunités.

FuturePath

Stages Prédiction Chatbot PFE Profil

Recommendations

Voici les offres de stage recommandées en fonction de votre profil.

Filtres

- Mots-clés: Rechercher...
- Localisation: Toutes
- Plateformes: LinkedIn, Indeed, Glassdoor
- Statut: Nouveau, Vu, Sauvegardé
- Trier par: Pertinence

Data Analyst Intern - X Delivery
Entreprise : Boston Consulting Group 0.41%
Lieu : Casablanca, Morocco
Durée : N/A
Plateforme : Indeed
Job details Job type Full-time Location Casablanca Full job description Who We Are Who We Are Boston Consulting Group partners with leaders in business and society to tackle their most important challenges and capture their greatest opportunities. BCG was the pioneer in business strategy when it was founded in 1963. Today, we help clients with total transformation-inspiring...
Analytics, Data Science, Excel, Python, React, Sql, Statistics

Specialist HR Analytics & Controlling
Entreprise : Leoni 0.41%
Lieu : Agadir, Morocco
Durée : N/A
Plateforme : Indeed

FIGURE 7.14 : Interface des recommandations de stages personnalisées

Note : Les recommandations sont automatiquement mises à jour à chaque connexion de l'utilisateur. Toute modification des préférences ou du CV entraîne la génération de nouvelles recommandations adaptées aux informations actualisées.

7.4 Écran de Prédiction de Filière

L'interface utilisateur de notre système de prédiction d'orientation académique est conçue pour être intuitive, épurée et centrée sur l'expérience utilisateur. Elle est divisée en plusieurs étapes successives, facilitant la saisie des informations et la compréhension des résultats.

7.4.1 Étape 1 – Informations personnelles

Cet écran permet à l'utilisateur de renseigner ses données sociodémographiques : sexe, âge, statut socio-économique, lieu de résidence, orientation au lycée et mention obtenue au baccalauréat. Cette étape est essentielle pour cerner le profil de l'étudiant.

FuturePath

Stages Prédiction Chatbot PFE Profil

Prédiction de Filière

Notre système d'intelligence artificielle analyse vos performances académiques et vos centres d'intérêt pour vous recommander la filière qui correspond le mieux à votre profil.

1 2 3 4

Informations personnelles Modules Semestre 1-2 Modules Semestre 3-4 Centres d'intérêt

Informations personnelles

Sexe Femme	Âge 19
Statut socio-économique Élevé	Situation géographique Urbain
Orientation au Lycée Science Physique	Mention au Bac Très Bien

Précédent Suivant

FIGURE 7.15 : Interface des recommandations de stages personnalisées

7.4.2 Étape 2 – Notes des Semestres 1 et 2

L'utilisateur saisit ses notes (sur 20) pour les modules fondamentaux du premier cycle universitaire, tels que Algèbre 1, Analyse 1, Physique, Informatique, Langue et Communication, etc.

The screenshot shows the FuturePath software interface. At the top, there is a navigation bar with tabs: Stages, Prédiction, Chatbot, PFE, and a black button labeled "Profil". Below the navigation bar, there is a horizontal progress bar divided into four segments, each containing a number (1, 2, 3, 4) and a corresponding label: Informations personnelles, Modules Semestre 1-2, Modules Semestre 3-4, and Centres d'intérêt. The second segment (Modules Semestre 1-2) is highlighted with a blue circle around the number 2.

Notes des modules (Semestre 1-2)

Veuillez saisir vos notes pour les modules du premier et deuxième semestre (sur 20).

Algèbre 1 14	Analyse 1 14
Physique 1 (Mécanique) 15	Mécanique du Point 12
Informatique 1 13	Langue et Communication 1 15
Algèbre 2 13	Analyse 2 15
Physique 2 14	Chimie 15
Informatique 2 13	Langue et Communication 2 18

Précédent **Suivant**

FIGURE 7.16 : Notes des Semestres 1 et 2

7.4.3 Étape 3 – Notes des Semestres 3 et 4

Suite logique de la précédente, cet écran recueille les notes des modules avancés du cycle préparatoire, notamment Analyse 3, Mécanique, Électronique, Mathématiques Appliquées, etc.

The screenshot shows the FuturePath software interface. At the top, there is a navigation bar with the logo "FuturePath" and links for "Stages", "Prédiction", "Chatbot", "PFE", and "Profil". Below the navigation bar, a progress bar indicates four steps: "Informations personnelles" (step 1), "Modules Semestre 1-2" (step 2), "Modules Semestre 3-4" (step 3, highlighted in blue), and "Centres d'intérêt" (step 4). The main content area is titled "Notes des modules (Semestre 3-4)" and contains a message: "Veuillez saisir vos notes pour les modules du troisième et quatrième semestre (sur 20)". Below this, there are two columns of input fields for module grades:

Module	Note
Algèbre 3	14
Analyse 3	12
Mécanique 2	13
Électronique 1	12
Informatique 3	13
Langue et Communication 3	15
Analysse 4	14
Mathématiques Appliquées	13
Physique 3	12
Physique 4	15
Électronique 2	13
Langue et Communication 4	17

At the bottom left is a "Précédent" button, and at the bottom right is a "Suivant" button.

FIGURE 7.17 : Notes des Semestres 3 et 4

7.4.4 Étape 4 – Centres d'intérêt

L'étudiant évalue son intérêt pour différents domaines sur une échelle de 1 à 5 : Programmation, Réseaux, Data Science, Électronique, Management, etc. Cette étape est déterminante dans la recommandation de filière.

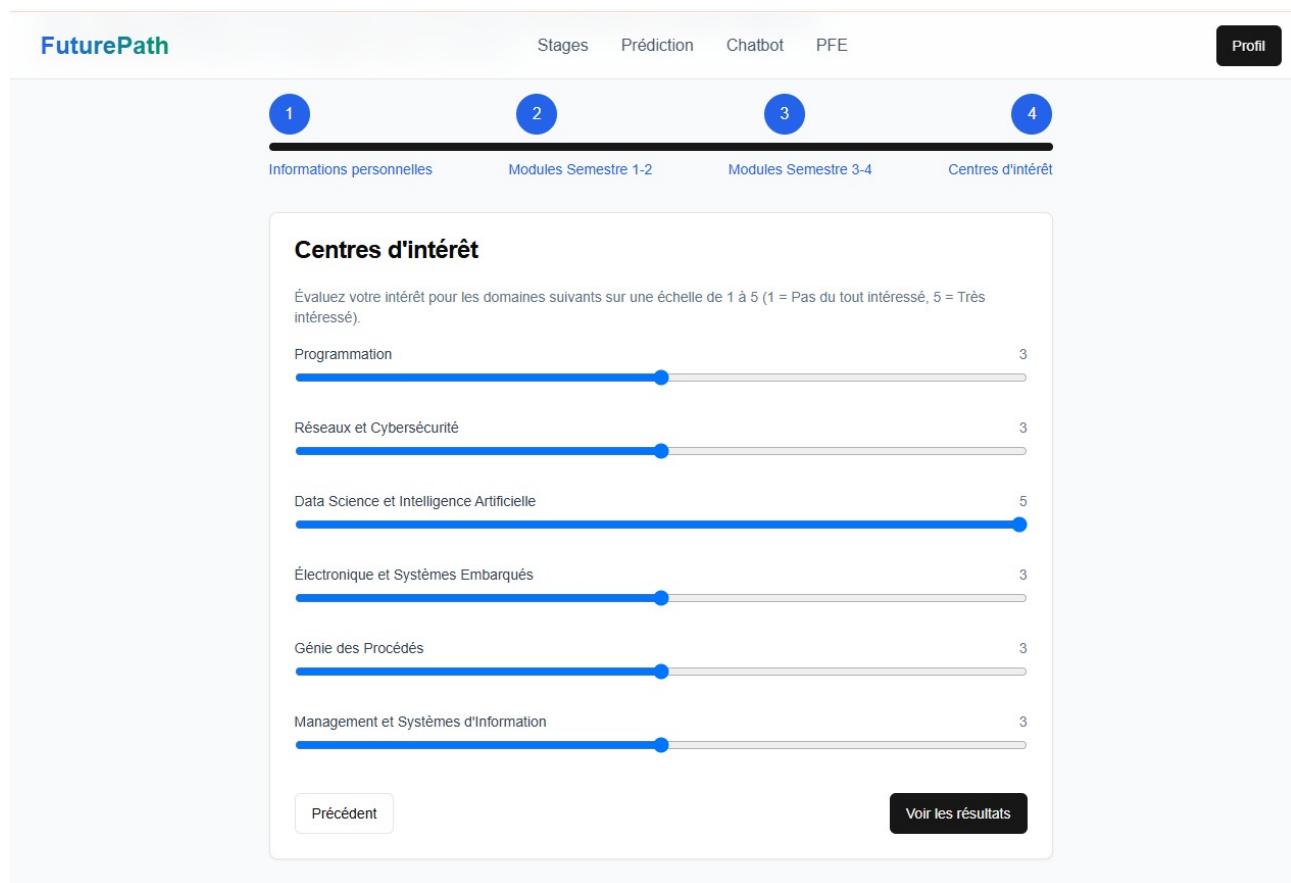


FIGURE 7.18 : Centres d'intérêt

7.4.5 Résultat de la prédition

Après soumission, le système affiche :

- La filière recommandée
- Un graphique de compatibilité avec les autres filières
- Les points forts détectés
- Les débouchés professionnels associés

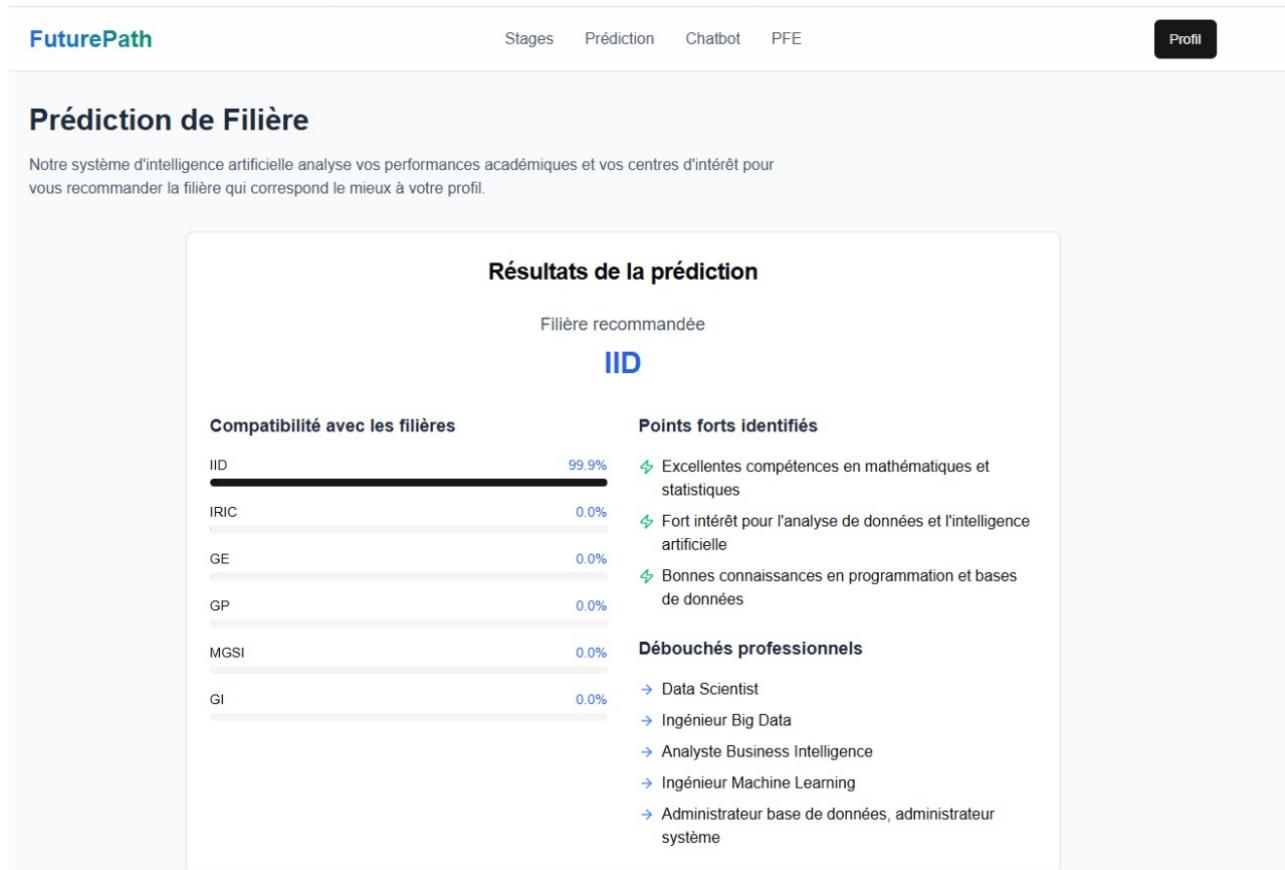


FIGURE 7.19 : Résultat de la prédition

7.4.6 Aperçu de la filière recommandée

Cette section présente un résumé de la filière choisie : durée de formation, infrastructures, projets pratiques et partenariats industriels.

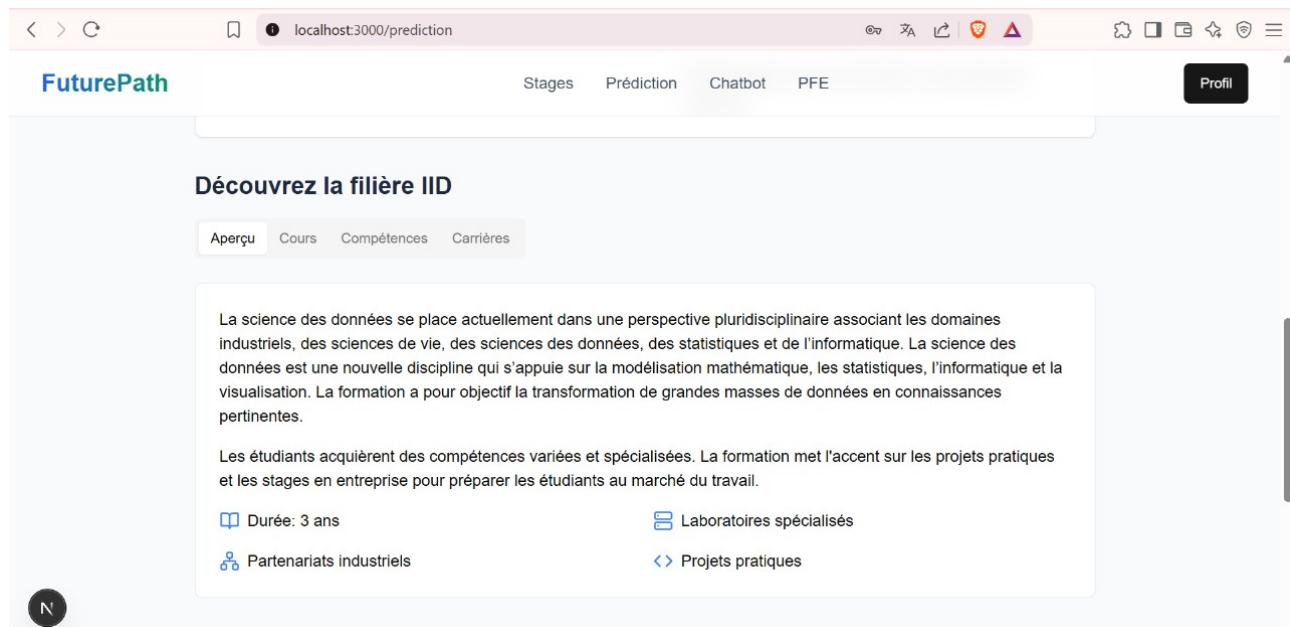


FIGURE 7.20 : Aperçu de la filière recommandée

7.4.7 Découvrez la filière – Cours principaux

Liste des cours principaux dispensés dans la filière, chacun avec un titre et une description détaillée, afin de donner un aperçu du contenu pédagogique.

The screenshot shows the FuturePath interface for the IRIC pathway. At the top, there is a navigation bar with tabs for 'Stages', 'Prédition', 'Chatbot', 'PFE', 'Connexion', and a black 'Inscription' button. Below the navigation bar, a breadcrumb trail indicates the current page: '→ Testeurs d'Intrusions (Pentester)'. The main content area is titled 'Découvrez la filière IRIC' and features a tab navigation with 'Aperçu', 'Cours' (which is selected), 'Compétences', and 'Carrières'. A sub-section titled 'Cours principaux' lists several courses with their descriptions:

- Réseaux informatiques avancés
Protocoles, routage et commutation
- Sécurité des systèmes et réseaux
Cryptographie, pare-feu et détection d'intrusion
- Administration système
Linux, Windows Server et services réseau
- Virtualisation et cloud computing
Technologies de virtualisation et services cloud
- Réseaux sans fil et mobiles
Technologies 5G, WiFi et IoT

At the bottom of this section are two buttons: 'Modifier mes réponses' and 'Découvrir les compétences requises'.

FIGURE 7.21 : Cours principaux

7.4.8 Découvrez la filière – Compétences

Compétences développées : description des compétences techniques et comportementales que les étudiants acquièrent.

The screenshot shows the FuturePath interface for the IID pathway. At the top, there is a navigation bar with tabs for 'Stages', 'Prédition', 'Chatbot', 'PFE', and a black 'Profil' button. Below the navigation bar, a breadcrumb trail indicates the current page: '→ Testeurs d'Intrusions (Pentester)'. The main content area is titled 'Découvrez la filière IID' and features a tab navigation with 'Aperçu', 'Cours', 'Compétences' (which is selected), and 'Carrières'. A sub-section titled 'Compétences développées' lists several competencies with their descriptions:

Analyse de données Techniques statistiques, préparation et nettoyage de données, interprétation des résultats.	Intelligence artificielle Algorithmes de machine learning, deep learning et traitement du langage naturel.
Big Data Technologies de traitement distribué, bases de données NoSQL et data lakes.	Business Intelligence Tableaux de bord, KPIs et aide à la décision basée sur les données.

At the bottom of this section are two buttons: 'Modifier mes réponses' and 'Découvrir les compétences requises'.

FIGURE 7.22 : Découvrez la filière – Compétences

7.4.9 Découvrez la filière – Carrières

Carrières : tableau des débouchés organisés par secteur (développement, administration réseaux, ingénierie, etc.).

The screenshot shows the FuturePath platform interface. At the top, there's a navigation bar with links for Stages, Prédiction, Chatbot, PFE, and Profil. Below the navigation, a section titled "Découvrez la filière IID" is displayed. Under this, there are tabs for Aperçu, Cours, Compétences, and Carrières, with "Carrières" being the active tab. A sub-section titled "Débouchés professionnels" lists various career paths. It includes three main categories: Data Science, Big Data, and Business Intelligence, each with a list of specific roles.

Data Science	Big Data	Business Intelligence
→ Data Scientist	→ Ingénieur Big Data	→ Analyste BI
→ Ingénieur Machine Learning	→ Architecte de données	→ Data Analyst
→ Spécialiste NLP	→ Ingénieur ETL	→ Consultant en analytique
→ Chercheur en IA	→ Administrateur de bases de données	→ Chef de projet BI

FIGURE 7.23 : Découvrez la filière –Carrières

7.5 Écran du Chatbot

1. Questions d'entretien en Data Analytics

The screenshot shows a web-based chatbot interface for "GenieusEnsa". The top navigation bar includes links for Stages, Prédiction, Chatbot, PFE, and Profil. The main content area features a heading "GenieusEnsa – L'intelligence qui éclaire votre avenir". A sidebar on the left contains a "Suggestions" section with three items: "Donner la liste de tous les filières de l'ENSAKH", "Peux-tu me donner des informations sur Dataverse ?", and "Peux-tu me donner une description du club Codex à l'ENSAKH ?". The main conversation area shows a message from the bot: "Bonjour ! Je suis GenieusEnsa, votre assistant intelligent. Je vous guide à travers tout ce qui concerne l'ENSAKH (événements, clubs, départements, enseignants, etc.) et je réponds aussi à vos questions générales : compétences clés, entretiens... Posez votre question, je suis là pour vous !" A user message in a purple box asks: "Quelles sont toutes les questions les plus courantes posées lors d'un entretien en Data Analytics ?". The bot responds with: "Voici quelques-unes des questions les plus courantes posées lors d'un entretien en Data Analytics".

FIGURE 7.24 : Interaction sur les questions en Data Analytics

The screenshot shows a web-based chatbot interface titled "FuturePath". At the top, there are navigation tabs: "Stages", "Prédiction", "Chatbot", and "PFE". A "Profil" button is located in the top right corner. The main area has a sidebar on the left labeled "Suggestions" containing several text input fields. The main content area displays a purple header box with the question: "Quelles sont toutes les questions les plus courantes posées lors d'un entretien en Data Analytics ?". Below this, a message from the bot says: "Voici quelques-unes des questions les plus courantes posées lors d'un entretien en Data Analytics :". Underneath, two sections are listed: "Questions Générales" and "Questions Techniques", each containing a numbered list of questions.

FIGURE 7.25 : Interaction sur les questions en Data Analytics

This screenshot shows the same "FuturePath" interface. The sidebar "Suggestions" contains the same four text input fields. The main content area now displays a section titled "Questions sur les Outils et les Technologies" with a numbered list of three questions. Below this is another section titled "Questions de Résolution de Problèmes" with a numbered list of three questions. At the bottom of the main content area, there is a text input field with the placeholder "Posez votre question sur l'ENSAKH..." and a small "Envoyer" button.

FIGURE 7.26 : Interaction sur les questions en Data Analytics

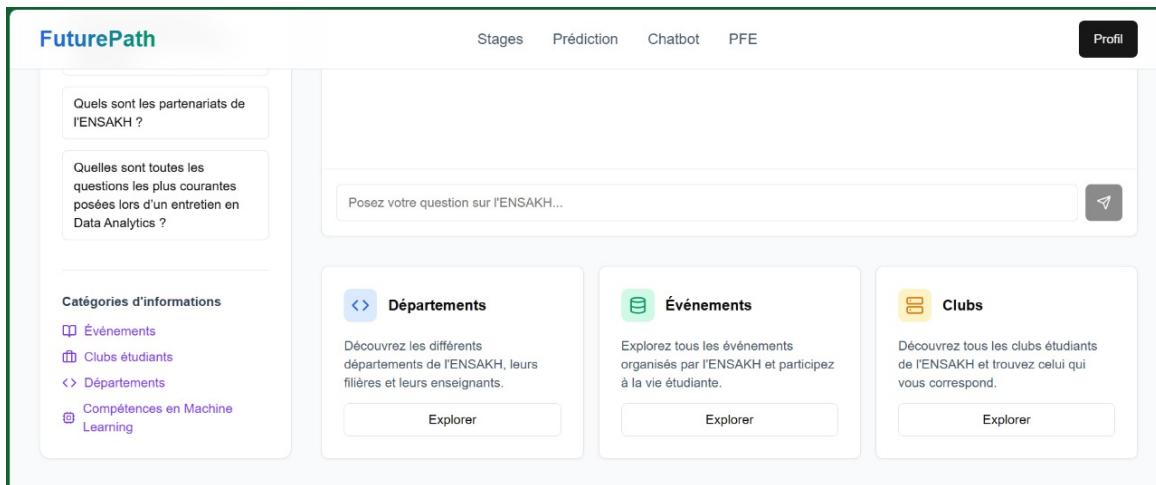


FIGURE 7.27 : Interaction sur les questions en Data Analytics

Description de l'interface –Quelles sont toutes les questions les plus courantes posées lors d'un entretien en Data Analytics ?

L'interface du chatbot **GenieusEnsaKh**, illustrée à travers la question sur les entretiens en Data Analytics, démontre une structuration claire et intuitive pensée pour optimiser l'expérience utilisateur. Elle débute par un en-tête présentant le nom du chatbot et un slogan accrocheur, suivi d'une brève description de ses fonctionnalités principales, telles que l'assistance sur les filières, clubs, enseignants ou encore les compétences professionnelles. Une section de suggestions cliquables permet à l'utilisateur de poser rapidement d'autres questions fréquentes. La réponse apportée par le chatbot est bien organisée selon différentes catégories pertinentes (questions générales, techniques, outils, résolution de problèmes), illustrant ainsi sa capacité à délivrer des informations précises, utiles et adaptées au contexte professionnel. L'interface est renforcée par un style visuel épuré et ergonomique, avec des titres, listes à puces et séparateurs clairs, facilitant la lisibilité. Enfin, des éléments interactifs complémentaires comme les filtres thématiques et les cartes de navigation enrichissent l'exploration, permettant d'accéder à d'autres domaines comme les départements ou les événements de l'ENSAKH. Cette interface reflète l'objectif du chatbot : agir comme un véritable assistant intelligent, capable d'informer, orienter et accompagner les étudiants dans leur parcours académique et professionnel.

2. Informations sur la filière Génie Électrique

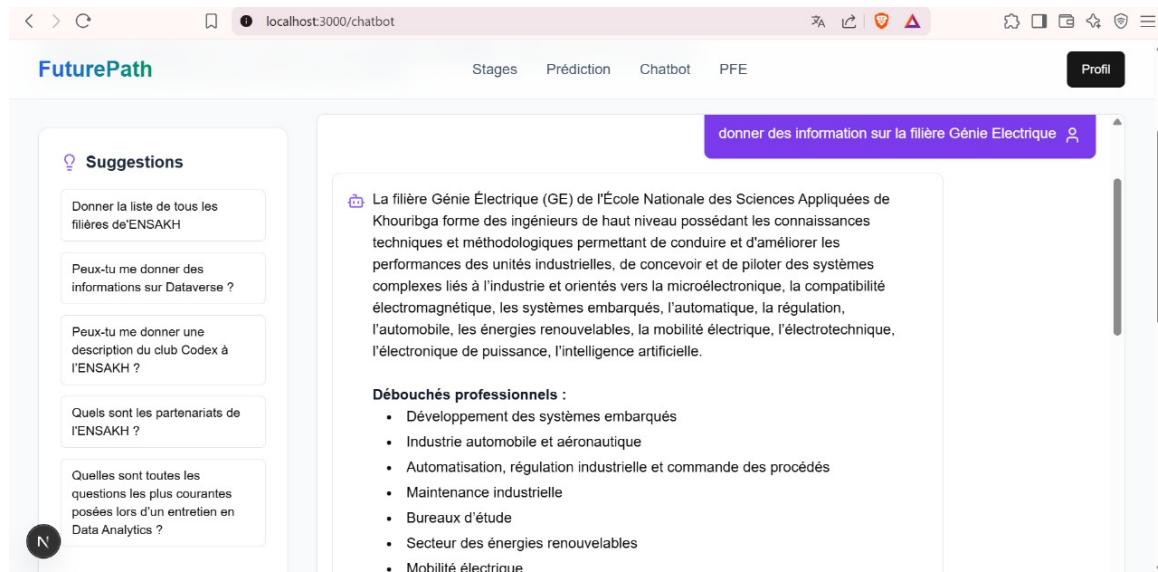


FIGURE 7.28 : Informations sur la formation

Description de l'interface – Donner des informations sur la filière Génie Electrique

Lorsque l'étudiant pose la question « Donner des informations sur la filière Génie Électrique », le chatbot fournit une réponse claire et bien structurée. L'information est présentée de manière logique, en commençant par une description générale de la formation, suivie des compétences développées au cours du cursus, des débouchés professionnels par secteur, ainsi que des prérequis académiques et des modalités d'admission.

Les données affichées proviennent directement des sources officielles de l'ENSA Khouribga, ce qui garantit leur fiabilité et leur exactitude. Cette structuration permet aux étudiants d'accéder facilement aux informations essentielles et d'obtenir une vision complète et cohérente de la filière.

L'interface s'avère particulièrement utile pour les étudiants du cycle préparatoire qui sont en phase de réflexion sur leur orientation. En regroupant l'ensemble des éléments clés nécessaires à la prise de décision — contenu pédagogique, débouchés, compétences, conditions d'accès — elle favorise une meilleure compréhension et aide à faire un choix éclairé.

Enfin, cette fonctionnalité s'intègre harmonieusement dans l'écosystème *FuturePath*, en complément du modèle de prédiction. Après avoir reçu une suggestion de filière adaptée à leur profil, les étudiants peuvent approfondir leur exploration grâce au chatbot, validant ainsi leur choix ou envisageant d'autres pistes en toute autonomie.

3. Présentation du Codex Club

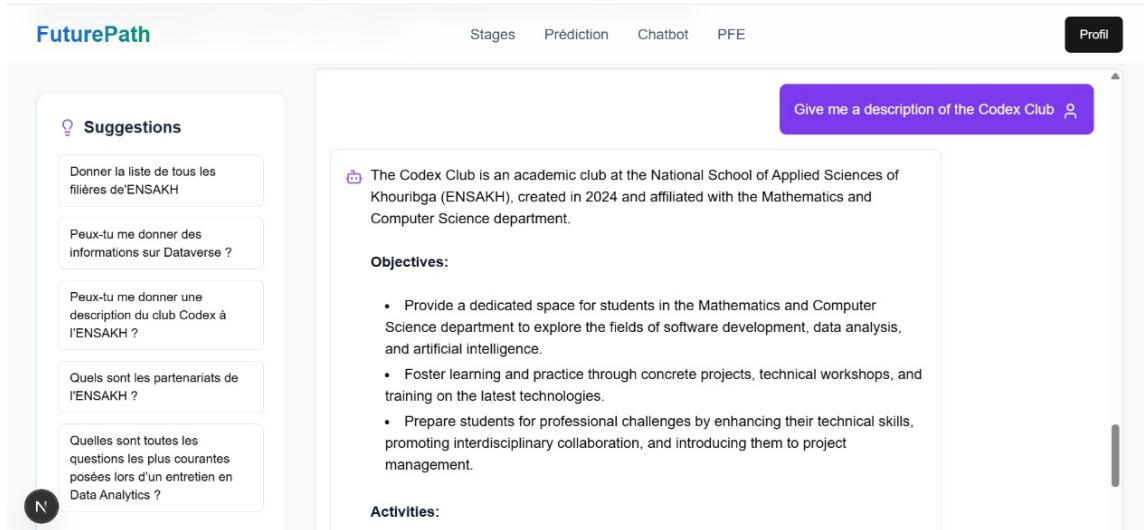


FIGURE 7.29 : Réponse concernant le Codex Club

Description de l'interface – Give me a description of the Codex Club

Lorsqu'un étudiant pose la question « Give me a description of the Codex Club », le chatbot démontre sa capacité multilingue en fournissant une réponse complète et bien structurée en anglais, prouvant ainsi sa maîtrise de plusieurs langues, dont l'anglais, le français et l'arabe. Cette fonctionnalité rend le système accessible à un public diversifié et élimine les barrières linguistiques. La réponse affichée présente avec clarté le Club Codex, en mentionnant sa création en 2024, son rattachement au département Mathématiques et Informatique, ainsi que ses objectifs axés sur le développement technologique, la formation pratique et la préparation aux défis professionnels. Elle détaille également les activités organisées, comme les workshops et les projets concrets. Cette interface est particulièrement utile pour les nouveaux étudiants, car elle leur permet de découvrir la vie associative de l'école, d'identifier les clubs disponibles, de comprendre leurs objectifs et d'évaluer leur adéquation avec leurs centres d'intérêt.

Conclusion

La conception de l'interface graphique a été guidée par le souci d'ergonomie, de simplicité et de performance. Chaque écran a été réfléchi pour répondre à un besoin spécifique de l'utilisateur : s'authentifier, consulter des recommandations de stages, découvrir une filière adaptée à son profil, ou encore interagir avec un chatbot intelligent.

Grâce à une organisation modulaire, des composants réutilisables et une attention particulière à l'expérience utilisateur, l'interface graphique permet de valoriser les fonctionnalités développées dans les chapitres précédents, tout en assurant une cohérence visuelle et fonctionnelle de l'application dans son ensemble.

Conclusion Générale

À l'issue de ce projet tutoré, nous avons conçu et développé une application intelligente dédiée à l'accompagnement des étudiants dans leur orientation académique et leur insertion professionnelle. En intégrant des technologies avancées telles que les Large Language Models (LLMs) et le traitement du langage naturel, nous avons pu proposer des solutions innovantes répondant à des besoins réels : la recommandation personnalisée d'offres de stage, la prédiction de la filière universitaire adaptée, l'assistance via un chatbot conversationnel, et l'accès facilité à des ressources académiques.

Ce projet nous a permis de mobiliser des compétences pluridisciplinaires allant du développement web à l'intelligence artificielle, en passant par la gestion de base de données et l'ergonomie d'interface. Il a également renforcé notre capacité à travailler en équipe, à gérer un projet de bout en bout et à concevoir une solution concrète orientée utilisateur.

L'application développée constitue une base solide pouvant être enrichie à l'avenir par de nouvelles fonctionnalités, telles que l'intégration d'un système de suivi des candidatures ou des tableaux de bord analytiques pour les établissements. Elle ouvre également des perspectives d'évolution vers une plateforme plus large de soutien à la réussite étudiante, au cœur des enjeux de l'enseignement supérieur.