

# Rapport de Projet de fin d'études

Pour l'Obtention du Diplôme

LICENCE SCIENCES ET TECHNIQUES

Spécialité : Génie Informatique

**Implémentation de microservices pour la gestion des  
ressources humaines basés sur l'intelligence artificielle  
avec Gemini 2.5, Firebase Studio et Flask**

**Réalisé par :**

Chouaib Yakine

**Sous la direction de :**

**Mme Doha Malki**, professeur à la FST de Settat

**Soutenu le :12 juin 2025**

**JURY**

**Pr. Doha MALKI**, professeur à la FST de Settat

**Pr. Haitam ETTAZI**, professeur à la FST de Settat

**Pr. Fatima Ezzahraa BENBOUAZZA**, professeur à la FST de Settat

**Année Universitaire : 2024-2025**

## Dédicaces

Je dédie ce rapport à tous ceux qui m'ont accompagné de près ou de loin durant ce stage. Merci à ma famille pour son soutien constant, et à mes amis pour leurs encouragements.

## Remerciements

Je tiens à exprimer ma profonde gratitude à l'ensemble du personnel de HumantechSolutions qui m'a accueilli chaleureusement et m'a permis d'évoluer dans un environnement professionnel enrichissant.

Je remercie tout particulièrement M. Ahmed Assalih, mon encadrant de stage, pour sa disponibilité, ses conseils avisés et la confiance qu'il m'a accordée tout au long de cette expérience.

J'adresse également mes sincères remerciements à Mme Doha Malki pour son accompagnement et ses orientations tout au long de mon projet.

Mes remerciements vont également à tous les enseignants de FSTS pour la qualité de la formation qu'ils m'ont dispensée.

Enfin, je souhaite remercier ma famille et mes proches pour leur soutien constant, leurs encouragements et leur patience durant cette période décisive de mon parcours.

# Sommaire

Dédicaces .....	2
Remerciements .....	3
Sommaire .....	4
Résumé .....	6
Introduction générale.....	11
<b>CHAPITRE 1 Contexte générale du projet .....</b>	<b>12</b>
1. Introduction .....	13
2. Présentation de la société d'accueil : HumanTechSolution.....	13
3. Etude de l'existant .....	14
a. Description de l'existant .....	14
b. Critique de l'existant .....	14
c. Solution proposée .....	15
4. Choix de modelé de développement.....	15
5. Planning prévisionnel.....	16
6. Conclusion.....	17
<b>CHAPITRE 2 Spécification des besoins.....</b>	<b>18</b>
1. Introduction .....	19
2. Spécification des besoins fonctionnels .....	19
a. Authentification des utilisateurs .....	19
b. Dépôt de CV pour les candidats .....	19
c. Création et gestion des offres par les recruteurs.....	20
d. Postuler à une offre .....	20
e. Analyse automatique des CV .....	20
f. Matching candidat-offre.....	20
g. Système de notifications .....	21
3. Spécification des besoins non fonctionnels .....	21
4. Présentation des cas d'utilisation .....	21
a. Présentation des acteurs .....	22
b. Description des cas d'utilisation .....	22
c. Diagramme des cas d'utilisation global .....	33
<b>CHAPITRE 3 Conception du système .....</b>	<b>34</b>
1. Introduction .....	35
2. Modélisation dynamique.....	35
a. Diagramme de séquences.....	35

b.	Diagramme d'activité .....	40
3.	Modélisation statique .....	42
a.	Diagramme de classes .....	42
b.	Modèle relationnel .....	43
c.	Architecture de l'application .....	47
4.	Conclusion .....	48
<b>CHAPITRE 4 Réalisation du système .....</b>		<b>49</b>
1.	Introduction .....	50
2.	Environnement de développement .....	50
a.	Environnement matériel.....	50
b.	Environnement logiciel.....	51
3.	Principales interfaces graphiques .....	53
a.	Login et Sign Up.....	53
b.	Candidat (Homme) .....	54
c.	Gestion et analyse du CV .....	55
d.	Suivi de Candidature .....	57
e.	Offres Recommandées .....	57
f.	Tableau de Bord .....	58
g.	Gestion des offres d'emploi .....	59
h.	Suivi des candidatures .....	60
i.	Suivi des candidatures .....	60
j.	Tableau de bord .....	61
4.	Conclusion .....	62
<b>Conclusion générale .....</b>		<b>63</b>
<b>Références bibliographiques .....</b>		<b>64</b>
<b>ANNEXES.....</b>		<b>65</b>
	Initialisation Firebase ET Flask.....	65
	POST : Création d'une candidature.....	65
	GET : Candidatures d'un candidat.....	66
	GET : Candidatures pour une offre .....	66
	PUT : Mise à jour du statut d'une candidature .....	66
	DELETE : Suppression d'une candidature .....	67
	GET : Candidatures pour un recruteur.....	67
	Lancement du service.....	67

## Résumé

Ce rapport présente le stage de fin d'études effectué au sein de l'entreprise HumanTechSolutions, spécialisée dans la transformation digitale des ressources humaines, sur une durée de deux mois.

L'objectif principal de ce stage était de développer une architecture de microservices RH intelligente en utilisant Gemini 2.5, Firebase Studio et Flask, afin d'automatiser le traitement des candidatures et d'optimiser le matching entre profils et offres d'emploi.

Durant cette période, j'ai analysé les besoins fonctionnels du système, conçu et développé des microservices backend avec Flask, intégré l'intelligence artificielle via Gemini 2.5 pour l'analyse sémantique des données RH, géré l'authentification et la base de données grâce à Firebase Studio, puis développé une interface utilisateur moderne et réactive avec React. J'ai également réalisé des tests de performance afin de garantir la fiabilité, la fluidité et la cohérence globale de la solution.

En utilisant des outils tels que React pour le frontend, Flask pour le backend, Firebase Studio pour la gestion des utilisateurs et des données, Gemini 2.5 pour l'intelligence artificielle, ainsi que des outils complémentaires comme Axios, Postman, GitHub et VS Code, j'ai pu concevoir une solution complète, performante et évolutive dédiée à la gestion intelligente des processus RH.

Ce stage m'a permis d'approfondir mes compétences en développement web full-stack, en conception de microservices, en intégration d'IA appliquée aux ressources humaines, et en gestion sécurisée des données. Il a également renforcé mes connaissances en React et développé mon sens de l'organisation.

Les résultats obtenus incluent la mise en place d'une architecture complète de microservices RH, une interface utilisateur fonctionnelle, des services backend performants, une intégration réussie de Gemini 2.5 pour l'analyse intelligente des CV et des offres, ainsi qu'un système sécurisé de gestion des utilisateurs et des données via Firebase. Cette solution a permis d'automatiser et d'optimiser efficacement le processus de recrutement, tout en garantissant sa scalabilité.

Ce stage a été une expérience professionnelle riche et formatrice, constituant une étape clé dans mon parcours.

## Abstract

This report presents the final-year internship carried out at HumanTech Solutions, located in Technopark, a company specialized in the digital transformation of human resources, over a period of two months.

The main objective of this internship was to develop an intelligent HR microservices architecture using Gemini 2.5, Firebase Studio, and Flask to automate candidate processing and optimize the matching between profiles and job offers.

During this period, I analyzed the functional requirements of the system, designed and developed backend microservices with Flask, integrated artificial intelligence through Gemini 2.5 for semantic analysis of HR data, managed authentication and the database using Firebase Studio, and developed a modern and responsive user interface with React. I also performed performance testing to ensure the solution's reliability, smoothness, and overall coherence.

By using tools such as React for the frontend, Flask for the backend, Firebase Studio for user and data management, Gemini 2.5 for AI integration, as well as complementary tools like Axios, Postman, GitHub, and VS Code, I was able to design a complete, efficient, and scalable solution dedicated to intelligent HR process management.

This internship allowed me to deepen my skills in full-stack web development, microservices design, AI integration in human resources, and secure data management. It also strengthened my knowledge of React and improved my organizational skills.

The outcomes include the implementation of a comprehensive HR microservices architecture, a functional user interface, high-performance backend services, successful integration of Gemini 2.5 for intelligent CV and job offer analysis, and a secure user and data management system via Firebase. This solution effectively automated and optimized the recruitment process while ensuring scalability.

This internship was a rich and formative professional experience, marking a key milestone in my academic and career path.

## Liste des tableaux

- Tableau 1 : Description du cas d'utilisation S'authentifier pour Utilisateur (*page 23*)
- Tableau 2 : Description du cas d'utilisation Gérer profil pour Utilisateur (*page 24*)
- Tableau 3 : Description du cas d'utilisation UploadCV pour Candidat (*page 25*)
- Tableau 4 : Description du cas d'utilisation Analyser CV pour Système AI (*page 26*)
- Tableau 5 : Description du cas d'utilisation Extraire compétence pour Système AI (*page 27*)
- Tableau 6 : Description du cas d'utilisation Publier offre pour recruteur (*page 28*)
- Tableau 7 : Description du cas d'utilisation Modifier offre pour recruteur (*page 29*)
- Tableau 8 : Description du cas d'utilisation Lister candidats compatibles (*page 30*)
- Tableau 9 : Description du cas d'utilisation Suggérer offres pertinentes (*page 31*)
- Tableau 10 : Description du cas d'utilisation Notification (*page 32*)
- Tableau 11 : modèle relationnel Users (*page 44*)
- Tableau 12 : modèle relationnel Jobs (*page 44*)
- Tableau 13 : modèle relationnel Applications (*page 45*)
- Tableau 14 : modèle relationnel CV\_Analyses (*page 46*)
- Tableau 15 : modèle relationnel Matching\_Result (*page 46*)
- Tableau 16 : modèle relationnel Notifications (*page 46*)



## Liste des figures

- Figure 1 : Diagramme de Gantt (*page 16*)
- Figure 2 : Diagramme des acteurs (*page 22*)
- Figure 3 : Diagramme des cas d'utilisation (*page 33*)
- Figure 4 : Diagramme de séquence Authentification (Inscription) (*page 35*)
- Figure 5 : Diagramme de séquence Authentification (connexion) (*page36*)
- Figure 6 : Diagramme de séquence Upload CV (*page36*)
- Figure 7 : Diagramme de séquence calcul des scores de compatibilité entre un CV et les offres (*page37*)
- Figure 8 : Diagramme de séquence Publication d'une offre et mise à jour de la liste des offres (*page38*)
- Figure 9 : Diagramme de séquence Candidat postule, recruteur consulte et décide (Candidat) (*page39*)
- Figure 10 : Diagramme de séquence Candidat postule, recruteur consulte et décide (Recruteur) (*page40*)
- Figure 11 : Diagramme d'état-transition Candidature (*page41*)
- Figure 12 : Diagramme d'état de l'accès aux offres avec tri par score (*page42*)
- Figure 13 : Diagramme d'état de la notification (*page43*)
- Figure 14 : Diagramme d'activité Postuler à une offre (*page44*)
- Figure 15 : Diagramme d'activité Traitement d'une candidature par un recruteur (*page45*)
- Figure 16 : Diagramme de classes (*page45*)
- Figure 17 : Schéma relationnel (*page46*)

## Liste des abréviations

<b><i>Abréviation</i></b>	<b><i>Signification complète</i></b>
<b>AI</b>	Artificial Intelligence (Intelligence Artificielle)
<b>API</b>	Application Programming Interface
<b>CV</b>	Curriculum Vitae
<b>HR</b>	Human Resources (Ressources Humaines)
<b>GUI</b>	Graphical User Interface
<b>ISO</b>	International Organization for Standardization
<b>IDE</b>	<i>Integrated Development Environment</i>
<b>CMMI</b>	<i>Capability Maturity Model Integration</i>
<b>UML</b>	<i>Langage de Modélisation Unifié</i>
<b>NLP</b>	<i>Natural Language Processing</i>
<b>NLTK</b>	<i>Natural Language Toolkit</i>

# Introduction générale

Dans un contexte où la digitalisation transforme profondément le secteur des ressources humaines, les entreprises sont confrontées à la nécessité de moderniser leurs processus de recrutement. La gestion manuelle des candidatures, chronophage et souvent inefficace, engendre des retards et une mauvaise adéquation entre les compétences des candidats et les besoins réels des postes à pourvoir. Face à ces enjeux, l'intégration de technologies avancées telles que l'intelligence artificielle et les architectures à base de microservices constitue une réponse innovante et performante.

C'est dans cette optique que HumanTech Solutions, entreprise spécialisée dans la transformation digitale des ressources humaines, a lancé un projet ambitieux visant à automatiser le traitement des candidatures tout en optimisant le matching entre les profils et les offres d'emploi. Ce projet s'inscrit dans une dynamique globale d'innovation RH, avec pour objectif d'améliorer l'expérience utilisateur tant pour les recruteurs que pour les candidats.

Mon stage de deux mois s'est inscrit dans ce cadre, avec pour mission principale la conception et la mise en œuvre d'un système intelligent reposant sur une architecture de microservices. Ce système s'appuie sur plusieurs technologies complémentaires : **Gemini 2.5** pour l'analyse sémantique des CV et des offres via intelligence artificielle, **Firebase** pour la gestion de l'authentification et des données, **Flask** pour le développement des microservices backend, et **React** pour la réalisation d'une interface utilisateur moderne, responsive et intuitive.

Le projet a été mené en plusieurs étapes. Tout d'abord, une **analyse approfondie des besoins fonctionnels** a été réalisée afin d'identifier les différents cas d'usage du système. Cette phase a été suivie d'une **modélisation UML** comprenant un **diagramme de cas d'utilisation** (interactions système/utilisateur), un **diagramme de classes** (structure des entités métiers), un **diagramme de séquence** (flux d'échange entre les composants), ainsi qu'un **diagramme d'activités** pour illustrer les différents scénarios de traitement.

Sur la base de cette modélisation, j'ai ensuite développé les **microservices backend**, intégré l'intelligence artificielle pour analyser les documents de candidature, conçu l'interface frontend, puis procédé à des **tests de performance et de cohérence**. L'ensemble visait à garantir la fluidité, la fiabilité et l'interopérabilité de la solution.

Ce travail m'a permis de mettre en pratique mes compétences en architecture logicielle, en développement web, en intégration d'IA et en gestion de projet technique. Il constitue également une contribution concrète à l'amélioration des pratiques de recrutement à l'ère du numérique.

# **C** **HAPITRE** **1**

## **Contexte générale du projet**

## **1. Introduction**

Ce chapitre présente le cadre dans lequel s'inscrit le projet de stage réalisé chez HumanTechSolution. Il expose l'environnement professionnel, décrit les pratiques existantes avant le début du projet, et identifie les principales limites rencontrées. Il introduit également la solution envisagée, les choix méthodologiques retenus, ainsi que le planning prévu. Ce contexte permet de mieux comprendre les motivations du projet et prépare à l'étude détaillée qui suivra dans les prochains chapitres.

## **2. Présentation de la société d'accueil : HumanTechSolution**

HumanTechSolution évolue dans le secteur des technologies de l'information, plus précisément dans la programmation, le conseil en gestion et les activités informatiques diverses. L'entreprise accompagne les organisations dans leur transformation digitale en combinant innovation technologique et compréhension des comportements humains pour optimiser les performances et la croissance.

Créée en 2009, HumanTechSolution a connu une expansion rapide grâce à une stratégie d'acquisitions et à une vision ambitieuse de la transformation digitale. Initialement une société technologique de premier plan, elle s'est imposée comme un acteur global dans le domaine de la transformation numérique, avec des certifications ISO 9001:2015 et CMMI Level 5, et une présence internationale en Asie, en Amérique du Nord et en Europe.

HumanTechSolution est structurée autour de plusieurs départements clés :

- Développement technologique et innovation
- Conseil en gestion et transformation digitale
- Gestion de projets et accompagnement au changement
- Support client et formation
- Recherche et développement en intelligence artificielle et data intelligence

L'étudiant a effectué son stage au sein du département de développement technologique, plus précisément dans l'équipe dédiée à l'automatisation et à l'intelligence artificielle. Ce service est en charge de la conception et du déploiement d'outils digitaux innovants visant à améliorer l'expérience utilisateur et à optimiser les processus métiers.

Dans ce département, plusieurs outils informatiques sont utilisés pour la gestion de projets et le développement :

- Plateformes de développement low-code/no-code pour accélérer la création d'applications
- Outils d'automatisation et d'Intelligence Artificielle pour la data intelligence et l'analyse prédictive
- Logiciels de gestion de projet collaboratifs pour le suivi des tâches et la coordination d'équipe
- Environnements de programmation modernes adaptés à la transformation digitale et à la gestion des données

### **3. Etude de l'existant**

#### **a. Description de l'existant**

Avant la mise en place de l'application développée durant ce stage, le processus de recrutement chez HumanTechSolution se faisait de manière classique. Les offres d'emploi étaient publiées sur des plateformes comme LinkedIn, et de nombreux candidats y postulaient en envoyant leur CV.

Le tri des candidatures était effectué manuellement par les responsables RH ou les recruteurs. Ces derniers devaient lire l'ensemble des CV reçus pour identifier les profils correspondant aux postes à pourvoir. Cette méthode demandait beaucoup de temps et d'efforts, notamment lorsqu'un grand nombre de candidats postulaient à une même offre. Il était difficile de savoir rapidement quels candidats étaient réellement qualifiés, car aucun système de filtrage ou de classement automatique n'était utilisé.

Ce fonctionnement générait une charge de travail importante et augmentait le risque de passer à côté de bons profils, tout en ralentissant l'ensemble du processus de recrutement.

#### **b. Critique de l'existant**

Le processus de recrutement manuel présentait plusieurs inconvénients majeurs. Premièrement, lorsqu'un recruteur recevait un grand nombre de CV pour une même offre, il était difficile d'identifier rapidement les profils réellement adaptés. Il n'existait aucun mécanisme permettant de

comparer objectivement les compétences des candidats, ce qui pouvait entraîner des erreurs de sélection. Par exemple, un candidat très qualifié pouvait être ignoré au profit d'un autre, simplement parce que son CV avait été lu en dernier ou mal interprété.

Deuxièmement, certains candidats postulaient sans lire attentivement les exigences du poste, ce qui entraînait la réception de nombreuses candidatures non pertinentes. Cela compliquait encore plus le travail du recruteur, qui devait passer du temps à écarter les profils inadaptés.

En résumé, l'absence d'un système intelligent pour filtrer, analyser et classer les candidatures créait un gaspillage de temps et augmentait le risque de choix non optimaux dans le processus de recrutement.

### **c. Solution proposée**

Pour pallier les faiblesses relevées dans le processus de sélection des candidats, une application innovante a été conçue durant ce stage.

L'application offre aux postulants la possibilité de candidater à une annonce via une plateforme unique. Lorsqu'un CV est soumis, le système procède automatiquement à une évaluation sémantique des compétences et les confronte aux critères de l'annonce. Il établit ensuite un score de compatibilité, ce qui permet de classer les candidats selon leur adéquation au poste.

Avec ce système de filtrage intelligent, les recruteurs ont la possibilité de se concentrer principalement sur les profils les plus adaptés, ce qui réduit considérablement le temps nécessaire pour trier les candidatures et améliore la qualité des recrutements. Cette automatisation aide également à minimiser les erreurs humaines et à se focaliser sur les candidats véritablement qualifiés.

## **4. Choix de modèle de développement**

Pour le développement de l'application, nous avons choisi le modèle de développement itératif et incrémental. Ce choix vient du fait que le projet est évolutif et contient plusieurs microservices comme AuthService pour l'authentification, AIService pour l'analyse des CV, le calcul du score, et aussi une interface frontend faite avec React.

Avec ce modèle, on avance petit à petit, en développant une partie du projet à chaque étape. Par exemple, au début on a fait le système d'authentification avec

Firestore. Ensuite, on a travaillé sur l'analyse des CV avec l'IA, puis sur le calcul du score entre le CV et l'offre d'emploi. À la fin, on a construit l'interface pour que le recruteur voit les résultats facilement.

Chaque microservice a été testé avec Postman, ce qui nous a permis de vérifier leur bon fonctionnement à chaque étape, de détecter rapidement les erreurs et de les corriger avant d'avancer.

Cette méthode nous a permis de tester et corriger à chaque fois, sans attendre la fin du projet. Elle nous a aussi aidés à organiser notre travail, à gérer le temps du stage, et à faire évoluer l'application étape par étape tout en gardant chaque partie bien séparée. Ce modèle est donc très adapté pour un projet comme celui-ci.

## 5. Planning prévisionnel

Pour une meilleure visibilité de l'avancement du projet et de l'organisation des tâches, un diagramme de Gantt a été réalisé. Il permet de suivre les différentes étapes du projet, leur durée ainsi que les liens entre elles.

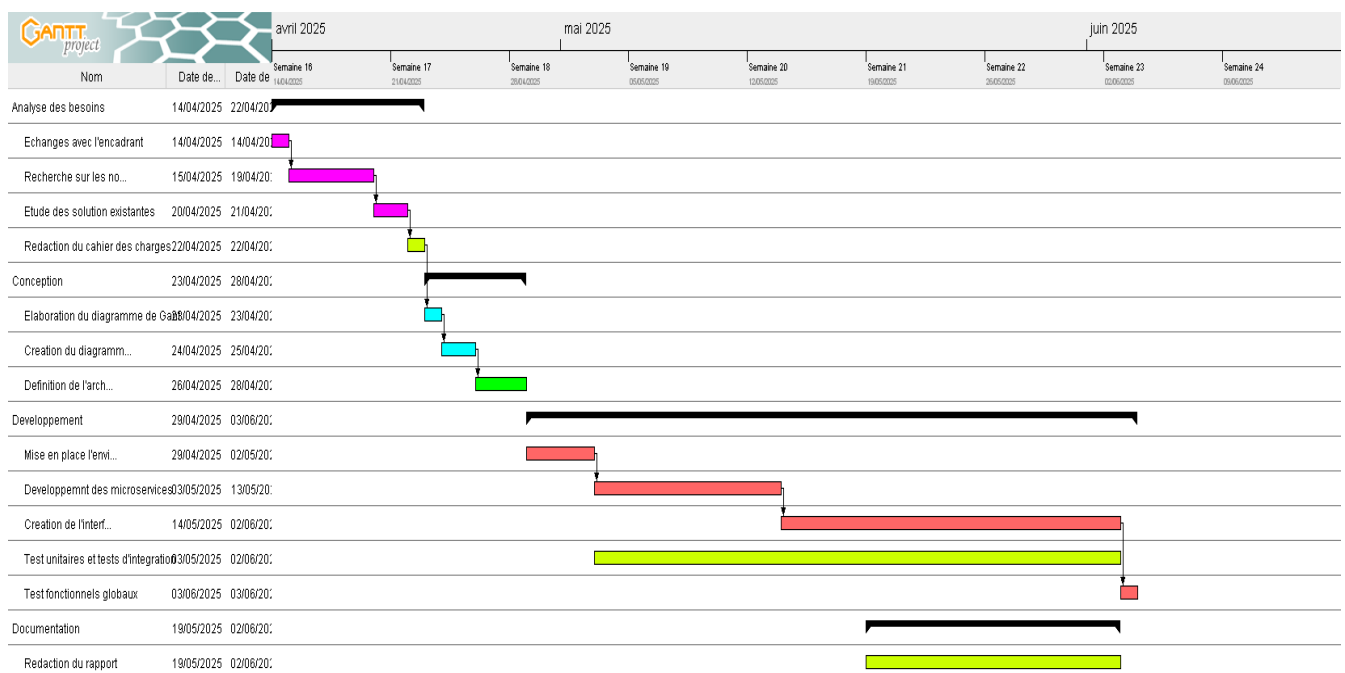


Figure 1 : Diagramme de Gantt



## 6. Conclusion

Ce chapitre a permis de situer le projet dans son contexte professionnel chez HumanTechSolution. On y a présenté l'environnement de travail, les méthodes utilisées avant le début du stage, ainsi que les limites du processus de recrutement manuel. Ces difficultés ont montré qu'il fallait une solution plus efficace pour gérer les candidatures.

La solution proposée est une application innovante composée de plusieurs microservices, comme AuthService pour l'authentification, AIService pour l'analyse des CV et le calcul du score, ainsi qu'une interface frontend développée avec React. Cette organisation en microservices permet d'automatiser et d'améliorer le tri des candidatures, pour que les recruteurs gagnent du temps et choisissent mieux les profils.

Le choix du modèle de développement itératif et incrémental a été judicieux pour ce projet qui évolue au fil du temps. Ce modèle nous a permis d'avancer étape par étape, en développant et en testant chaque partie séparément. Chaque microservice a été testé avec Postman, ce qui nous a aidés à vérifier leur bon fonctionnement, détecter les erreurs rapidement et les corriger avant de continuer.

Cette méthode a facilité l'organisation du travail, la gestion du temps pendant le stage, et la progression régulière du projet. Elle assure aussi que chaque partie reste bien indépendante tout en s'intégrant dans l'ensemble.

Les chapitres suivants détailleront la conception, le développement, les tests et la mise en place de cette application.

# **C** **HAPITRE** **2**

## **Spécification des besoins**

## 1. Introduction

Dans ce chapitre, on va présenter les besoins qui ont été identifiés pendant la phase d'analyse. Le but, c'est de structurer les choses de façon claire pour bien préparer la suite du projet.

Les besoins ont été classés en deux parties : les besoins fonctionnels et les besoins non fonctionnels. Cette séparation permet d'avoir une meilleure organisation et de savoir ce qu'on doit faire concrètement, et dans quelles conditions ça doit fonctionner.

Ce travail va servir de base pour tout ce qui vient après, que ce soit la conception, le développement ou les tests.

## 2. Spécification des besoins fonctionnels

Cette partie regroupe les besoins fonctionnels de l'application. Ce sont les fonctionnalités que le système doit obligatoirement assurer. Pour mieux organiser les choses, les besoins ont été divisés en besoins globaux, puis détaillés sous forme de sous-besoins.

### a. Authentification des utilisateurs

Ce besoin permet à tout utilisateur de s'inscrire et se connecter à la plateforme.

- i. L'utilisateur peut créer un compte avec son adresse email et un mot de passe.*
- ii. Lors de l'inscription, il peut choisir s'il est candidat ou recruteur.*
- iii. L'utilisateur peut se connecter à son compte.*
- iv. L'utilisateur peut se déconnecter.*

### b. Dépôt de CV pour les candidats

Le candidat peut déposer et gérer son CV depuis son compte.

- i. Il peut envoyer un CV au format PDF.*
- ii. À chaque nouvel envoi, l'ancien CV est remplacé automatiquement. Seul le dernier CV envoyé est pris en compte.*
- iii. Le CV est lié à son profil pour les candidatures.*

### **c. Création et gestion des offres par les recruteurs**

Le recruteur peut publier des offres et les gérer.

- i. Il peut créer une offre en ajoutant un titre, une description, des critères, des compétences recherchées, une localisation, et d'autres détails liés au poste.*
- ii. Il peut modifier ou supprimer ses offres à tout moment.*
- iii. Les offres sont visibles pour tous les candidats connectés.*

### **d. Postuler à une offre**

Le candidat peut postuler aux offres qui l'intéressent.

- i. Il sélectionne une offre et clique sur "postuler".*
- ii. La candidature est enregistrée avec son CV.*
- iii. Il peut consulter ses candidatures et leur statut.*

### **e. Analyse automatique des CV**

Chaque CV est analysé pour détecter les compétences.

- i. Le système envoie le CV à un service d'analyse IA.*
- ii. Les compétences extraites sont stockées pour le matching.*
- iii. L'analyse du CV est aussi affichée au candidat sous forme de feedback, avec un résumé de ses compétences, ses expériences détectées, et un aperçu général de son profil.*

### **f. Matching candidat-offre**

Un score est calculé pour chaque candidature.

- i. Le score dépend des compétences du candidat et des exigences de l'offre.*
- ii. Il aide les recruteurs à trier les candidatures.*

## g. Système de notifications

Les utilisateurs reçoivent des alertes importantes.

- i. *Le candidat reçoit une notification si sa candidature est acceptée ou refusée.*
- ii. *Le recruteur est notifié lorsqu'un candidat postule à une de ses offres.*
- iii. *Quand une nouvelle offre est créée, les candidats reçoivent une notification pour être informés des nouvelles opportunités.*

## 3. Spécification des besoins non fonctionnels

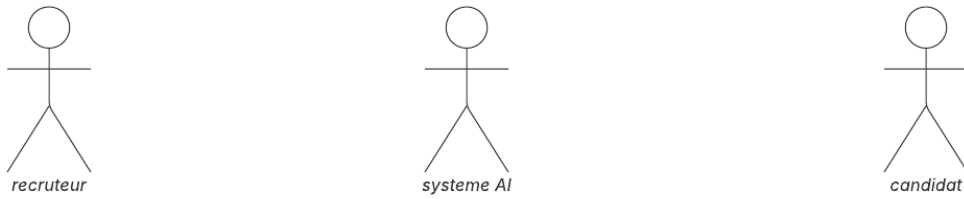
Ce sont les besoins qui servent à améliorer la qualité de l'application. Même si ce n'est pas des fonctionnalités directes pour l'utilisateur, ils sont importants pour que tout fonctionne bien.

- ✓ **Sécurité des données** : les infos des utilisateurs (comme le mot de passe et le CV) doivent être protégées. On utilise Firebase Auth pour sécuriser l'accès.
- ✓ **Temps de réponse rapide** : les actions comme la connexion, la postulation ou l'envoi de CV doivent se faire sans attendre trop longtemps
- ✓ **Convivialité de l'interface** : l'interface doit être simple et facile à comprendre pour tous les utilisateurs.
- ✓ **Compatibilité** : l'application doit marcher sur les navigateurs récents et différents écrans (ordi, tablette...).
- ✓ **Modularité** : chaque microservice est indépendant, ce qui permet de modifier une partie sans casser les autres.
- ✓ **Sauvegarde des données** : les données doivent être sauvegardées pour éviter les pertes en cas de bug ou coupure.

## 4. Présentation des cas d'utilisation

Dans cette partie, on présente comment les différents utilisateurs interagissent avec l'application. Chaque utilisateur a ses propres actions. les cas d'utilisation permettent de mieux comprendre le rôle de chacun et les fonctionnalités principales de la plateforme.

## a. Présentation des acteurs



*Figure 2 : Diagramme des acteurs*

## b. Description des cas d'utilisation

Dans cette partie, on va décrire quelques cas d'utilisation importants de l'application. Chaque cas montre comment un acteur (candidat, recruteur ou système) interagit avec le système pour accomplir une tâche bien précise.

Chaque fiche contient l'acteur, l'objectif, les conditions avant/après, et le déroulement normal du scénario, avec aussi un scénario alternatif

**Tableau 1 : Description du cas d'utilisation S'authentifier pour Utilisateur**

<b>Cas n°</b>	1
<b>Acteur</b>	Candidat / Recruteur
<b>Objectif</b>	Se connecter à la plateforme pour accéder à son espace personnel
<b>Préconditions :</b>	<ul style="list-style-type: none"> <li>▪ L'utilisateur a déjà un compte</li> <li>▪ Il dispose de ses identifiants ( email et mot de passe )</li> </ul>
<b>Post-conditions :</b>	<ul style="list-style-type: none"> <li>▪ L'utilisateur est redirigé vers son tableau de bord</li> <li>▪ Son rôle ( candidat ou recruteur ) est reconnu par le système</li> </ul>
<b>ScENARIO nominal</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur accède à la page de connexion</li> <li>2. Il entre son email et son mot de passe</li> <li>3. Il clique sur le bouton "Se connecter"</li> <li>4. Le système vérifie les informations</li> <li>5. L'utilisateur est connecté et redirigé selon son rôle (candidat ou recruteur)</li> </ol>
<b>ScENARIO alternatif :</b>	<ul style="list-style-type: none"> <li>▪ Si l'email ou le mot de passe est incorrect, <b>un message d'erreur s'affiche</b></li> <li>▪ Si l'utilisateur n'a pas encore de compte, <b>il peut cliquer sur "Créer un compte"</b></li> <li>▪ Si le serveur d'authentification est indisponible, <b>un message d'erreur est affiché</b></li> </ul>

**Tableau 2 : Description du cas d'utilisation Gérer profil pour Utilisateur**

<b>Cas n°</b>	2
<b>Acteur</b>	Candidat / Recruteur
<b>Objectif</b>	Mettre à jour les infos personnelles et les préférences de l'utilisateur
<b>Préconditions :</b>	<ul style="list-style-type: none"> <li>▪ L'utilisateur est déjà connecté</li> <li>▪ Le profil existe dans la base de données</li> </ul>
<b>Post-conditions :</b>	<ul style="list-style-type: none"> <li>▪ Les nouvelles infos sont enregistrées</li> <li>▪ Le profil est affiché avec les changements</li> </ul>
<b>Scenario nominal</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur va dans la section "Mon profil"</li> <li>2. Il clique sur "Profil"</li> <li>3. Il change ses infos : nom, rôle, email</li> <li>4. Il clique sur "Enregistrer"</li> <li>5. Le système met à jour la base de données</li> <li>6. Un message de confirmation s'affiche et le profil actualisé est montré</li> </ol>
<b>Scenario alternatif :</b>	<ul style="list-style-type: none"> <li>▪ Champs obligatoires manquants, <b>un message d'erreur s'affiche</b></li> <li>▪ Format d'email invalide, <b>le système demande de corriger</b></li> <li>▪ Problème de connexion ou de serveur, <b>l'enregistrement échoue et un message d'erreur apparaît</b></li> </ul>



**Tableau 3 : Description du cas d'utilisation UploadCV pour Candidat**

<b>Cas n°</b>	3
<b>Acteur</b>	Candidat
<b>Objectif</b>	Permettre au candidat de déposer son CV sur la plateforme et recevoir un feedback automatique
<b>Préconditions :</b>	<ul style="list-style-type: none"> <li>▪ Le candidat est connecté</li> <li>▪ Aucun problème de connexion au serveur</li> </ul>
<b>Post-conditions :</b>	<ul style="list-style-type: none"> <li>▪ Le CV est stocké dans le système</li> <li>▪ Analyse est lancée automatiquement</li> <li>▪ Les compétences sont extraites et enregistrées dans Firestore</li> <li>▪ Un feedback est affiché dans l'espace du candidat</li> </ul>
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. Le candidat va dans la section "Mon CV"</li> <li>2. Il clique sur "Déposer un CV"</li> <li>3. Il choisit un fichier PDF</li> <li>4. Le système enregistre le fichier (en remplaçant l'ancien s'il y en a)</li> <li>5. Le CV est envoyé à l'IA pour analyse</li> <li>6. Les compétences et autres infos sont extraites automatiquement</li> <li>7. Les résultats sont enregistrés dans Firestore</li> </ol>
<b>Scénario alternatif :</b>	<ul style="list-style-type: none"> <li>▪ Si le fichier n'est pas un PDF, <b>un message d'erreur s'affiche</b></li> <li>▪ Si le système ne peut pas lire le fichier, <b>l'analyse est annulée</b></li> </ul>

**Tableau 4 : Description du cas d'utilisation Analyser CV pour Système AI**

<b>Cas n°</b>	4
<b>Acteur</b>	Système d'IA (Gemini 2.5)
<b>Objectif</b>	Analyser automatiquement le CV du candidat pour extraire les informations importantes
<b>Préconditions :</b>	<ul style="list-style-type: none"> <li>▪ Le candidat a déjà déposé un CV</li> <li>▪ Le fichier est au bon format (PDF) et lisible</li> </ul>
<b>Post-conditions :</b>	<ul style="list-style-type: none"> <li>▪ Les compétences, expériences, et un résumé sont extraits</li> <li>▪ Le feedback est affiché au candidat</li> </ul>
<b>Scenario nominal</b>	<ol style="list-style-type: none"> <li>1. Le système détecte qu'un nouveau CV a été déposé</li> <li>2. Le CV est envoyé automatiquement au service d'analyse IA</li> <li>3. L'IA scanne le document avec NLP</li> <li>4. Elle extrait les compétences, expériences, diplômes, etc</li> <li>5. Un résumé du profil est généré</li> <li>6. Le feedback est affiché au candidat dans son espace</li> </ol>
<b>Scenario alternatif :</b>	<ul style="list-style-type: none"> <li>▪ Si le CV est vide, mal formaté ou corrompu, <b>le système affiche un message d'erreur</b></li> <li>▪ Si l'analyse échoue partiellement, <b>seules certaines données sont extraites</b></li> </ul>

**Tableau 5 : Description du cas d'utilisation Extraire compétence pour Système AI**

<b>Cas n°</b>	5
<b>Acteur</b>	Système d'IA (Gemini 2.5)
<b>Objectif</b>	Extraire automatiquement les compétences du candidat à partir de son CV
<b>Préconditions :</b>	<ul style="list-style-type: none"> <li>▪ Elles seront utilisées plus tard pour le matching et l'affichage du profil</li> <li>▪ Le fichier est lisible et analysable</li> </ul>
<b>Post-conditions :</b>	<ul style="list-style-type: none"> <li>▪ Les compétences sont détectées par l'IA</li> <li>▪ Elles sont enregistrées dans Firestore</li> <li>▪ Elles seront utilisées plus tard pour le matching et l'affichage du profil</li> </ul>
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. Le système détecte un nouveau CV dans le compte du candidat</li> <li>2. Le fichier est envoyé au service d'analyse IA</li> <li>3. L'IA utilise le traitement NLP pour lire le contenu</li> <li>4. Les compétences techniques et générales sont extraites automatiquement</li> <li>5. Le système enregistre les compétences extraites dans Firestore, liées au compte du candidat</li> </ol>
<b>Scénario alternatif :</b>	<ul style="list-style-type: none"> <li>▪ Si le CV est vide, mal formaté ou corrompu, <b>le système affiche un message d'erreur</b></li> </ul>

**Tableau 6 : Description du cas d'utilisation Publier offre pour recruteur**

<b>Cas n°</b>	6
<b>Acteur</b>	recruteur
<b>Objectif</b>	Permettre au recruteur de créer et publier une nouvelle offre d'emploi sur la plateforme
<b>Préconditions :</b>	<ul style="list-style-type: none"> <li>▪ Le recruteur est connecté à son compte</li> <li>▪ Il a accès à l'espace de gestion des offres</li> </ul>
<b>Post-conditions :</b>	<ul style="list-style-type: none"> <li>▪ L'offre est enregistrée dans Firestore</li> <li>▪ Elle devient visible par tous les candidats</li> <li>▪ Une notification est envoyée aux candidats concernés</li> </ul>
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. Le recruteur accède à l'espace "Mes offres"</li> <li>2. Il clique sur "créer offre"</li> <li>3. Il remplit les champs nécessaires : titre, description, compétences recherchées, localisation, etc.</li> <li>4. Il clique sur "Publier"</li> <li>5. L'offre est enregistrée dans Firestore</li> <li>6. Le système envoie une notification aux candidats selon les critères</li> </ol>
<b>Scénario alternatif :</b>	<ul style="list-style-type: none"> <li>▪ Un champ obligatoire est vide, <b>le système bloque la publication et affiche un message</b></li> <li>▪ Si Firestore est indisponible, <b>l'enregistrement échoue et un message d'erreur s'affiche</b></li> <li>▪ Si la notification ne part pas , <b>l'offre reste publiée mais sans alerte pour les candidats</b></li> </ul>

**Tableau 7 : Description du cas d'utilisation Modifier offre pour recruteur**

<b>Cas n°</b>	7
<b>Acteur</b>	recruteur
<b>Objectif</b>	Permettre au recruteur de modifier les détails d'une offre qu'il a déjà publiée
<b>Préconditions :</b>	<ul style="list-style-type: none"> <li>▪ Le recruteur est connecté</li> <li>▪ L'offre existe déjà dans Firestore et lui appartient</li> </ul>
<b>Post-conditions :</b>	<ul style="list-style-type: none"> <li>▪ Les nouvelles informations de l'offre sont mises à jour dans Firestore</li> <li>▪ Les candidats voient l'offre modifiée avec les nouvelles données</li> </ul>
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. Le recruteur accède à la liste de ses offres publiées</li> <li>2. Il clique sur "Modifier" sur une offre spécifique</li> <li>3. Il met à jour un ou plusieurs champs (titre, description, compétences, localisation, etc)</li> <li>4. Il clique sur "Enregistrer"</li> <li>5. Le système met à jour les infos de l'offre dans Firestore</li> </ol>
<b>Scénario alternatif :</b>	<ul style="list-style-type: none"> <li>▪ Si le recruteur essaye de modifier une offre qui ne lui appartient pas, <b>accès refusé</b></li> <li>▪ Si un champ obligatoire est supprimé, <b>le système bloque la mise à jour</b></li> </ul>

**Tableau 8 : Description du cas d'utilisation Lister candidats compatibles**

<b>Cas n°</b>	8
<b>Acteur</b>	Recruteur / Système de Matching
<b>Objectif</b>	Lister automatiquement les candidats ayant postulé à une offre, triés par compatibilité
<b>Préconditions :</b>	<ul style="list-style-type: none"> <li>▪ L'offre d'emploi est créée et enregistrée dans Firestore</li> <li>▪ Des candidats ont postulé à l'offre</li> <li>▪ Les profils analysés des candidats (compétences extraites via IA) sont stockés</li> <li>▪ Le système de matching est opérationnel</li> </ul>
<b>Post-conditions :</b>	<ul style="list-style-type: none"> <li>▪ Une <b>liste filtrée et triée des candidats ayant postulé</b> est affichée au recruteur</li> <li>▪ Chaque candidat est associé à un <b>score de compatibilité</b></li> <li>▪ Le recruteur peut <b>accepter</b> ou <b>refuser</b> chaque candidature</li> </ul>
<b>Scenario nominal</b>	<ol style="list-style-type: none"> <li>1. Le recruteur sélectionne une de ses offres d'emploi publiées</li> <li>2. Le système récupère les <b>compétences requises</b> de l'offre</li> <li>3. Le système récupère tous les <b>candidats ayant postulé</b> à cette offre</li> <li>4. Pour chaque candidat, le système récupère le <b>profil IA analysé</b> (compétences)</li> <li>5. Il calcule un <b>score de compatibilité</b> pour chaque candidat</li> <li>6. Il trie les candidats du <b>score le plus élevé au plus faible</b></li> <li>7. Il affiche cette liste au recruteur</li> <li>8. Le recruteur peut <b>consulter</b> chaque profil et <b>accepter ou refuser</b> les candidatures</li> </ol>
<b>Scenario alternatif :</b>	<ul style="list-style-type: none"> <li>▪ Si aucun candidat ne correspond aux critères, <b>le système affiche un message error</b></li> <li>▪ Si certains critères sont manquants dans l'offre, <b>le système adapte le calcul du score en conséquence</b></li> <li>▪ Si la Firestore des candidats est indisponible, <b>le système affiche un message d'erreur</b></li> </ul>

**Tableau 9 : Description du cas d'utilisation Suggérer offres pertinentes**

<b>Cas n°</b>	9
<b>Acteur</b>	Système IA (Gemini 2.5) / Candidat
<b>Objectif</b>	Proposer automatiquement des offres d'emploi compatibles avec le profil du candidat à partir de CV
<b>Préconditions :</b>	<ul style="list-style-type: none"> <li>▪ Le candidat a déjà déposé un CV au format PDF</li> <li>▪ Le CV a été analysé par l'IA (compétences extraites)</li> <li>▪ Des offres d'emploi sont disponibles dans la base Firestore</li> </ul>
<b>Post-conditions :</b>	<ul style="list-style-type: none"> <li>▪ Une liste d'offres pertinentes, triées selon le score de compatibilité, est affichée</li> <li>▪ Le candidat peut postuler à l'offre de son choix</li> </ul>
<b>Scenario nominal</b>	<ol style="list-style-type: none"> <li>1. Le candidat accède à la section "Offres suggérées"</li> <li>2. Le système récupère les compétences extraites du CV (profil IA)</li> <li>3. Il compare ces compétences avec celles exigées dans chaque offre</li> <li>4. Il calcule un score de compatibilité pour chaque offre</li> <li>5. Il trie les offres du score le plus élevé au plus bas</li> <li>6. Il affiche la liste au candidat</li> <li>7. Le candidat peut consulter chaque offre et postuler s'il le souhaite</li> </ol>
<b>Scenario alternatif :</b>	<ul style="list-style-type: none"> <li>▪ Si le profil IA n'existe pas, <b>le système affiche un message d'erreur</b></li> <li>▪ Si certaines offres n'ont pas de données complètes (compétences manquantes), <b>elles sont ignorées ou reçoivent un score de 0</b></li> </ul>

**Tableau 10 : Description du cas d'utilisation Notification**

<b>Cas n°</b>	10
<b>Acteur</b>	Recruteur / Candidat
<b>Objectif</b>	Envoyer automatiquement des notifications lors de la création d'offres ou lors de candidatures
<b>Préconditions :</b>	<ul style="list-style-type: none"> <li>▪ Chaque utilisateur (candidat ou recruteur) a un token FCM valide enregistré dans Firestore</li> </ul>
<b>Post-conditions :</b>	<ul style="list-style-type: none"> <li>▪ Le ou les utilisateurs ciblés reçoivent une notification en temps réel</li> </ul>
<b>Scenario nominal</b>	<p>A. Offre créée :</p> <ol style="list-style-type: none"> <li>1. Un recruteur crée une nouvelle offre</li> <li>2. Le système envoie une notification aux candidats avec un message type</li> </ol> <p>B. Postulation :</p> <ol style="list-style-type: none"> <li>1. Un candidat postule à une offre</li> <li>2. Le système identifie le recruteur propriétaire de l'offre</li> <li>3. Il envoie une notification personnalisée au recruteur</li> </ol>
<b>Scenario alternatif :</b>	<ul style="list-style-type: none"> <li>▪ Si Firestore ou FCM est indisponible, <b>une erreur est logée</b></li> <li>▪ Si le candidat a désactivé les notifications, <b>rien n'est affiché côté client</b></li> </ul>



### c. Diagramme des cas d'utilisation global

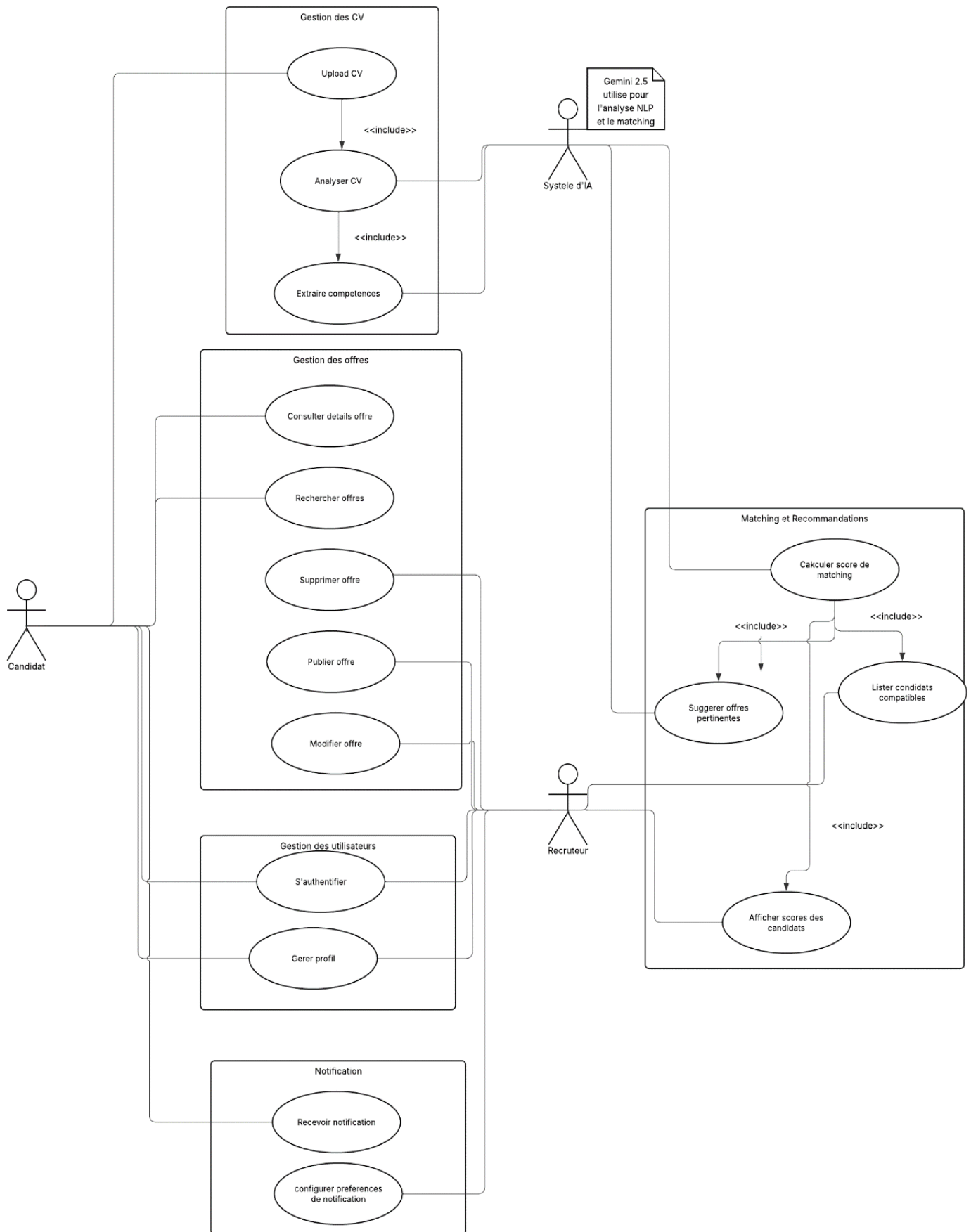


Figure 3 : Diagramme des cas d'utilisation

# **C** **HAPITRE** **3**

## **Conception du système**

# 1. Introduction

La conception du système est une étape clé qui permet de transformer les besoins fonctionnels en une structure claire et bien organisée. Dans notre cas, on utilise Firebase, notamment Firestore, ce qui influence certains choix techniques, surtout au niveau de la modélisation des données.

Ce chapitre est divisé en deux parties : d'abord la modélisation dynamique pour décrire le comportement du système (comme les séquences, les activités, etc.), puis la modélisation statique pour présenter la structure (diagramme de classes, modèle de données NoSQL, architecture de l'appli...). L'objectif, c'est d'avoir une vision globale et précise du système avant d'entrer dans la phase d'implémentation.

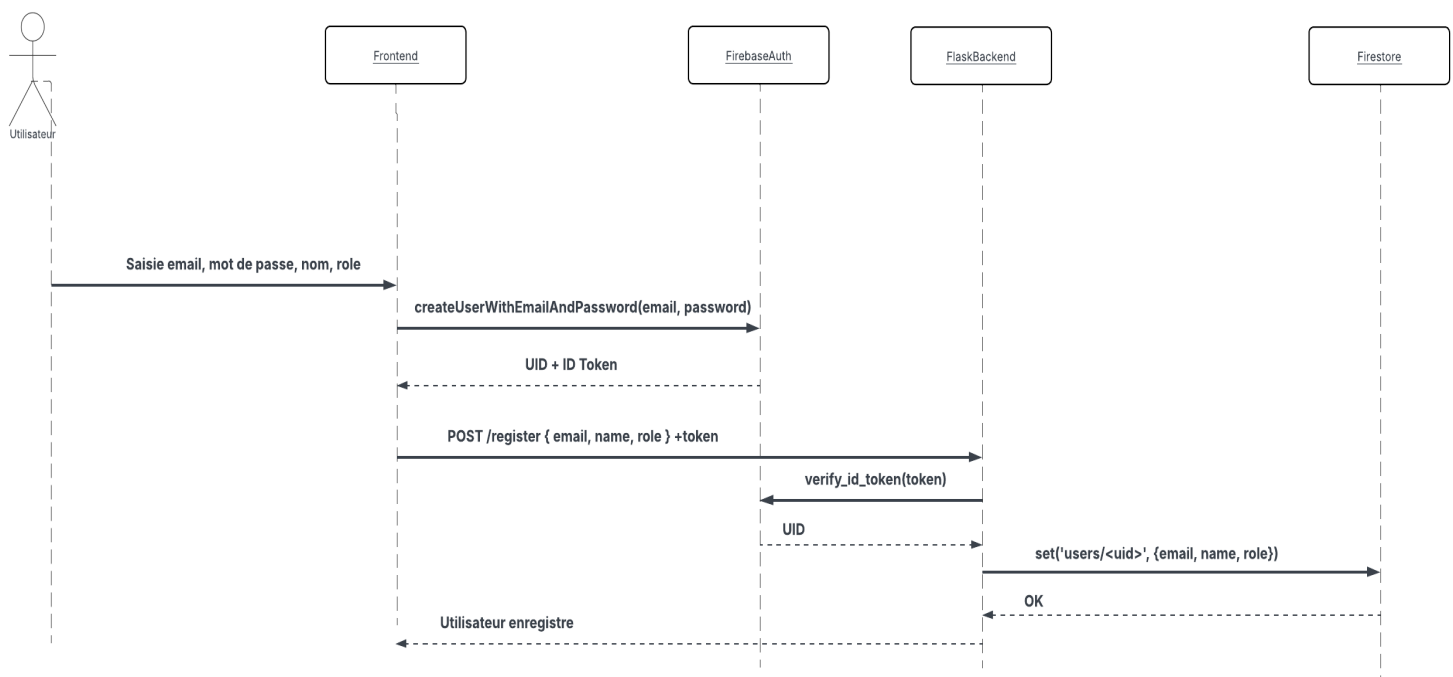
## 2. Modélisation dynamique

### a. Diagramme de séquences

Les diagrammes de séquence décrivent les interactions entre les objets au cours du temps pour un scénario donné. Nous avons choisi de représenter les cas d'utilisation principaux de manière à couvrir l'essentiel du comportement du système.

- **Diagramme séquence Authentification**

*Inscription :*



**Figure 4 : Diagramme de séquence Authentification (Inscription)**

## Connexion :

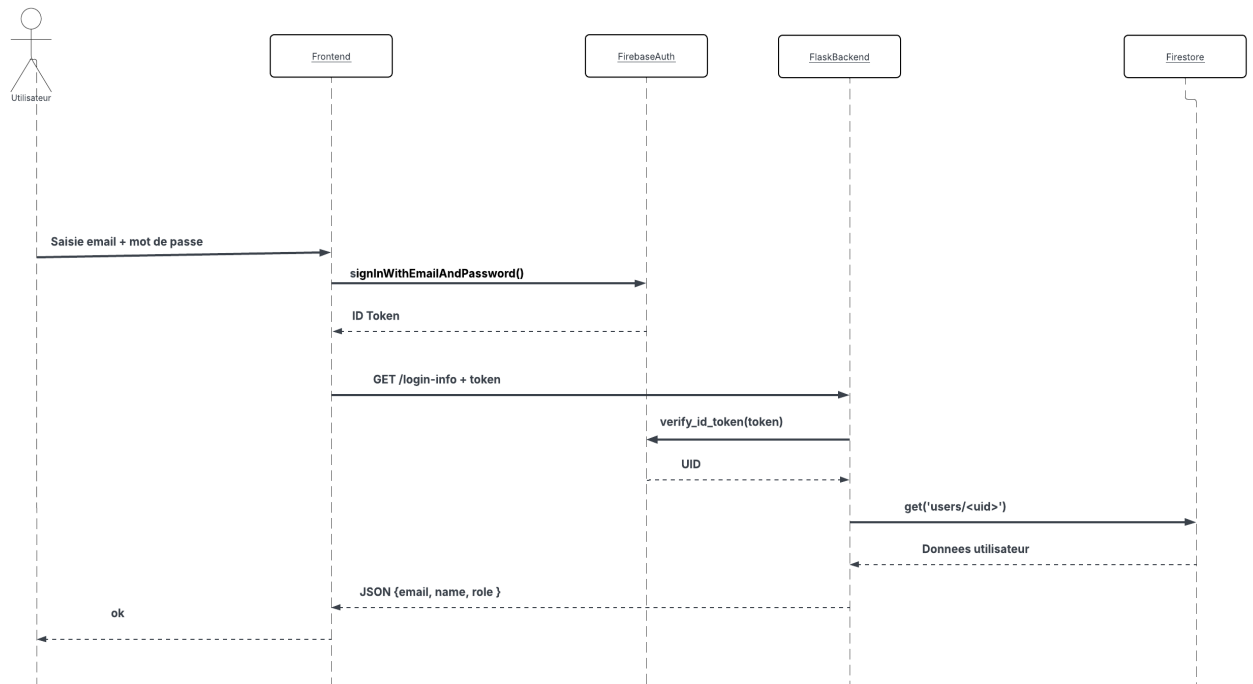


Figure 5 : Diagramme de séquence Authentification (connexion)

## • Diagramme de séquence Upload CV

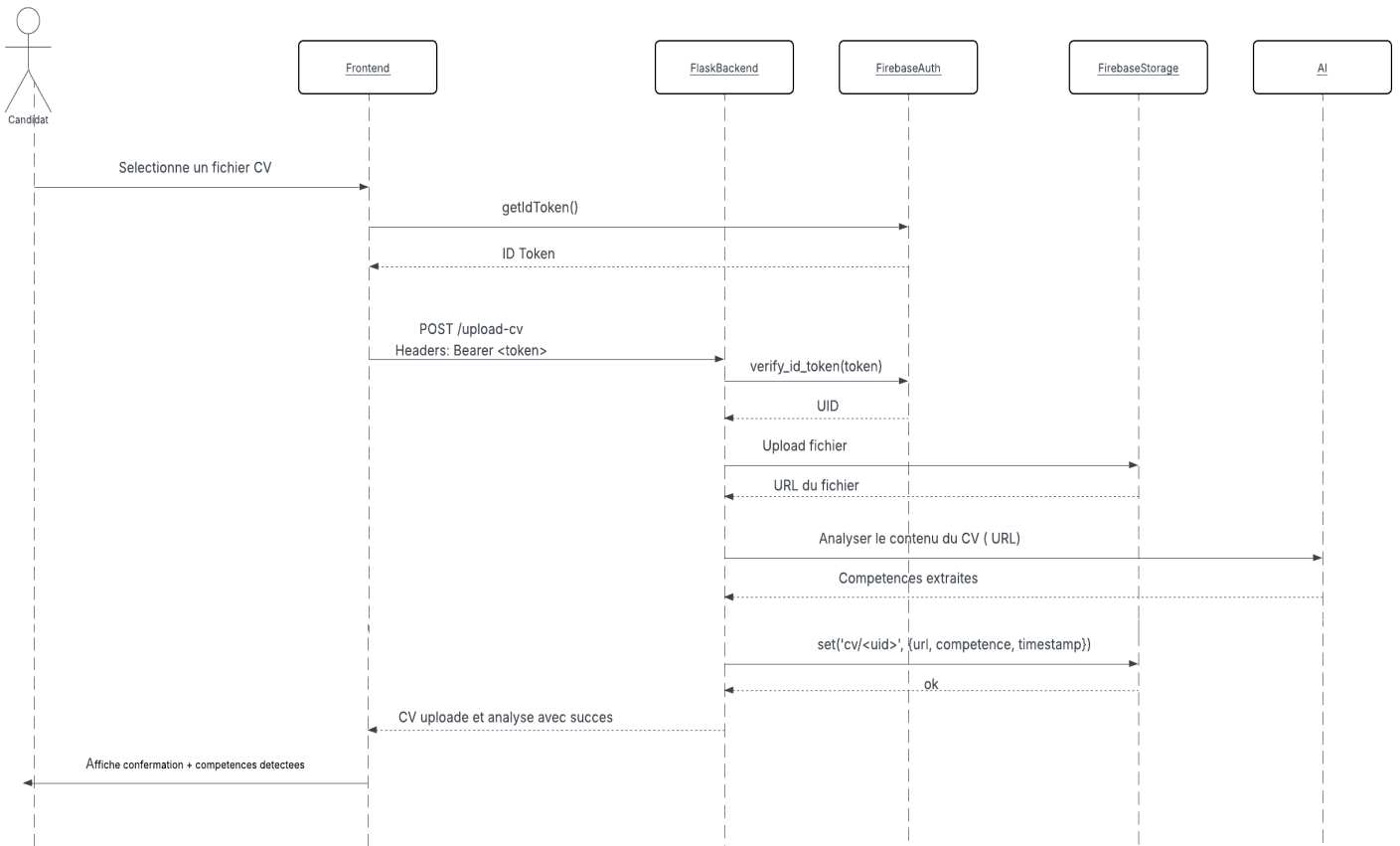
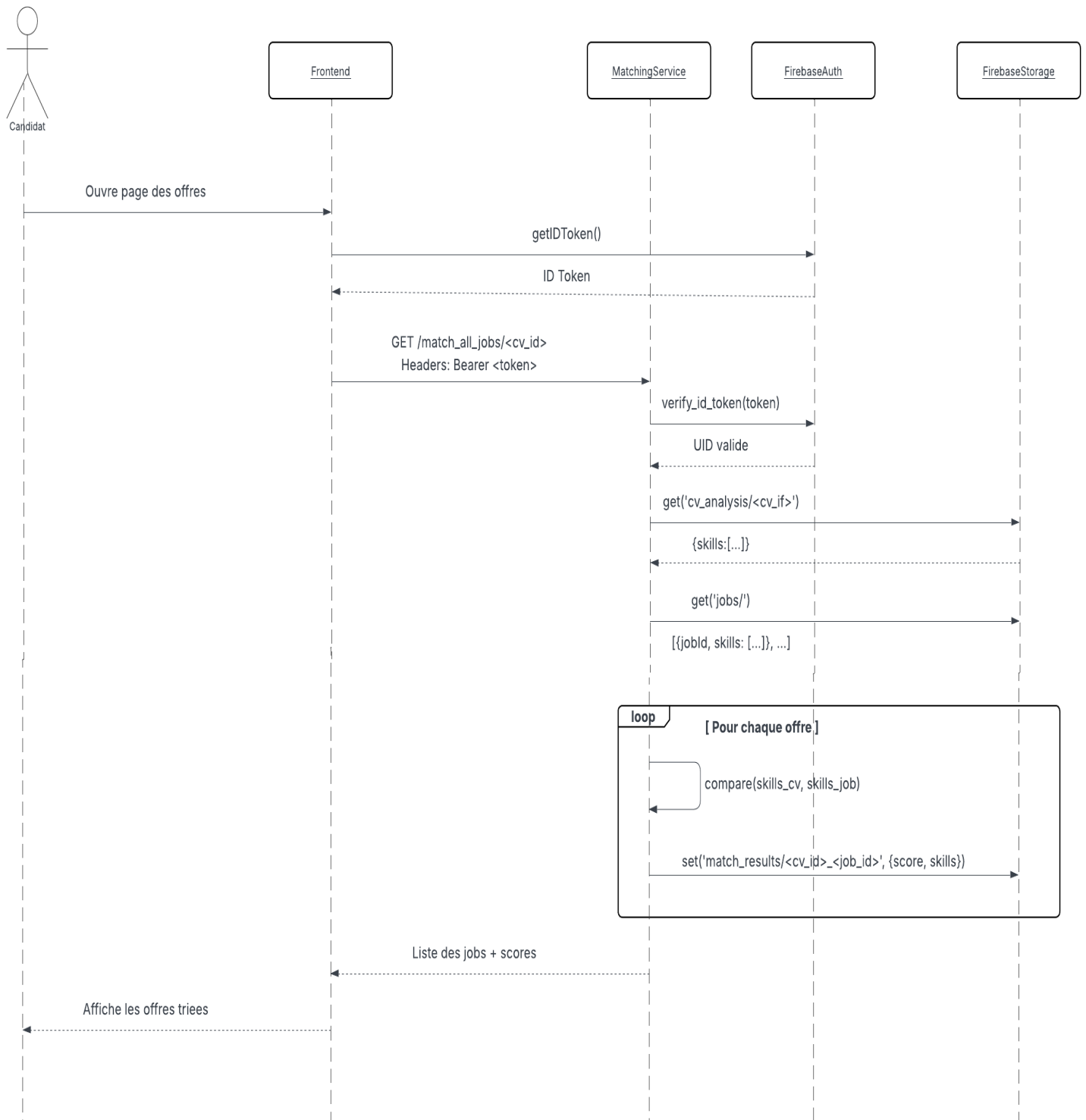


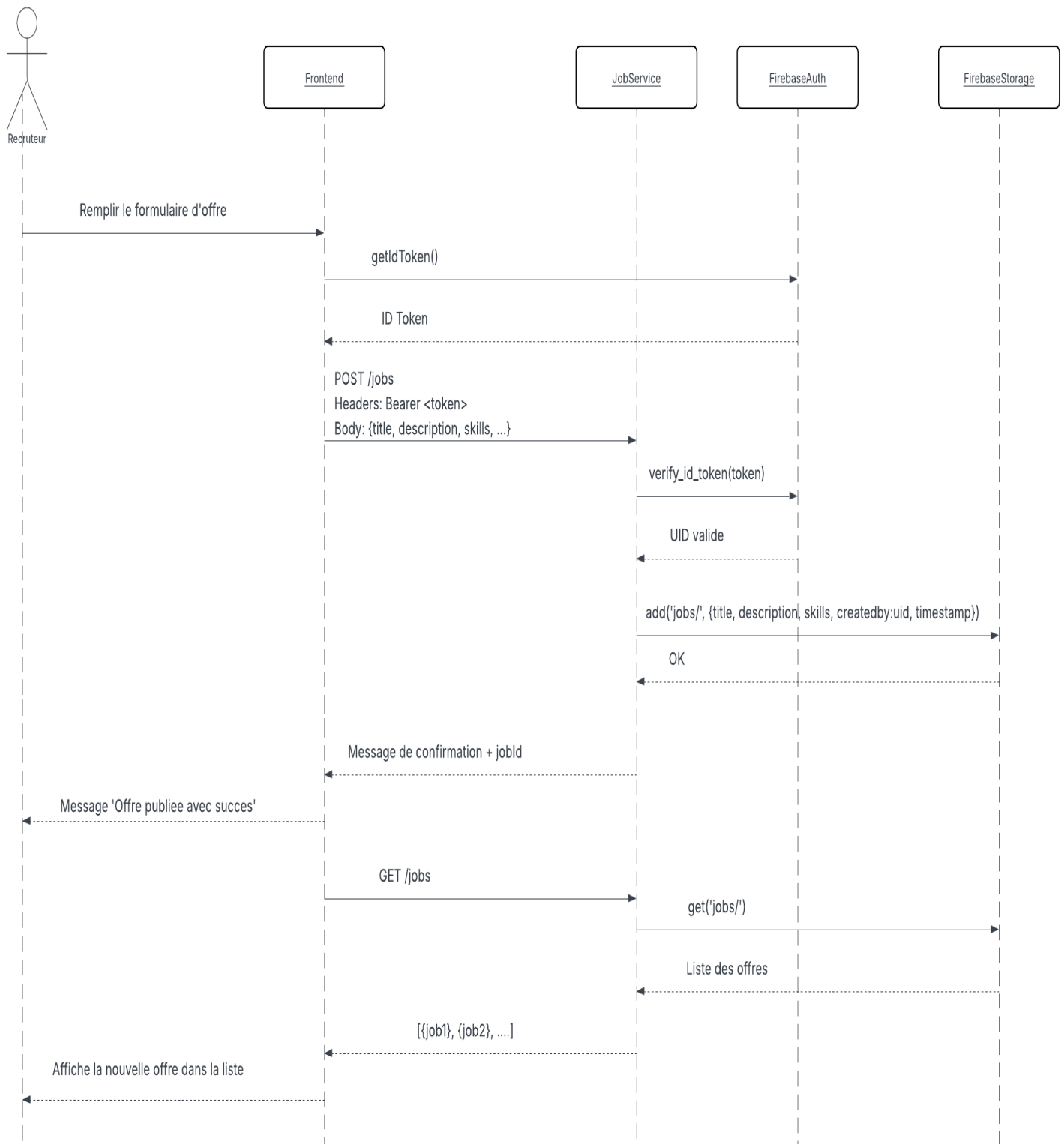
Figure 6 : Diagramme de séquence Upload CV

- **Diagramme de séquence calcul des scores de compatibilité entre un CV et les offres**



**Figure 7 : Diagramme de séquence calcul des scores de compatibilité entre un CV et les offres**

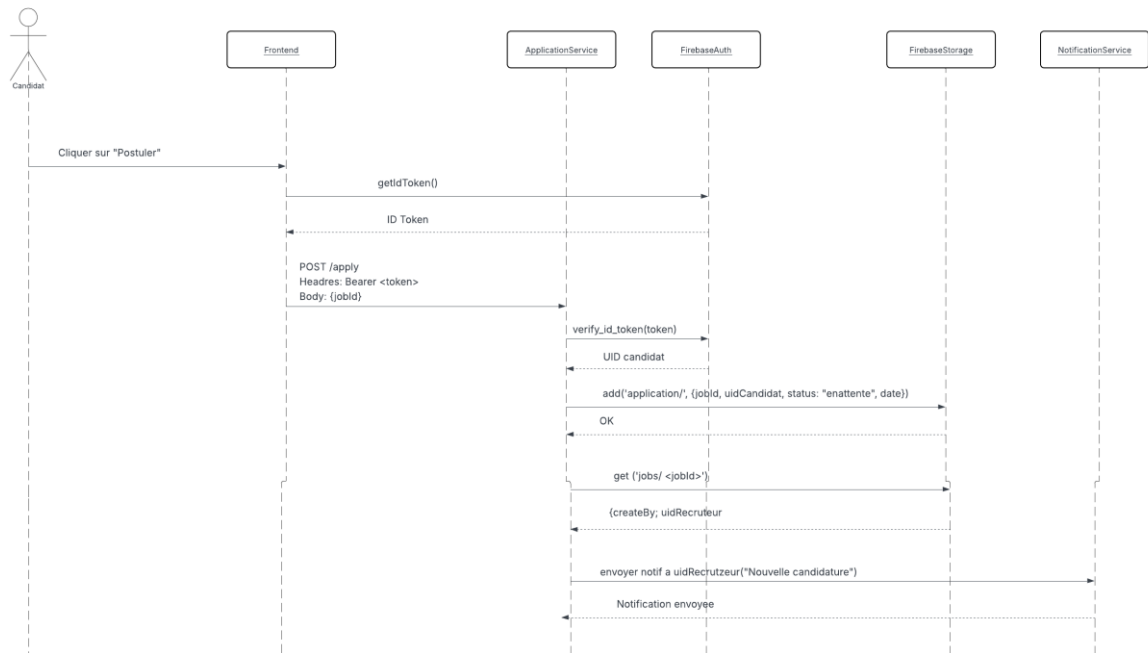
- **Diagramme de séquence Publication d'une offre et mise à jour de la liste des offres**



**Figure 8 : Diagramme de séquence Publication d'une offre et mise à jour de la liste des offres**

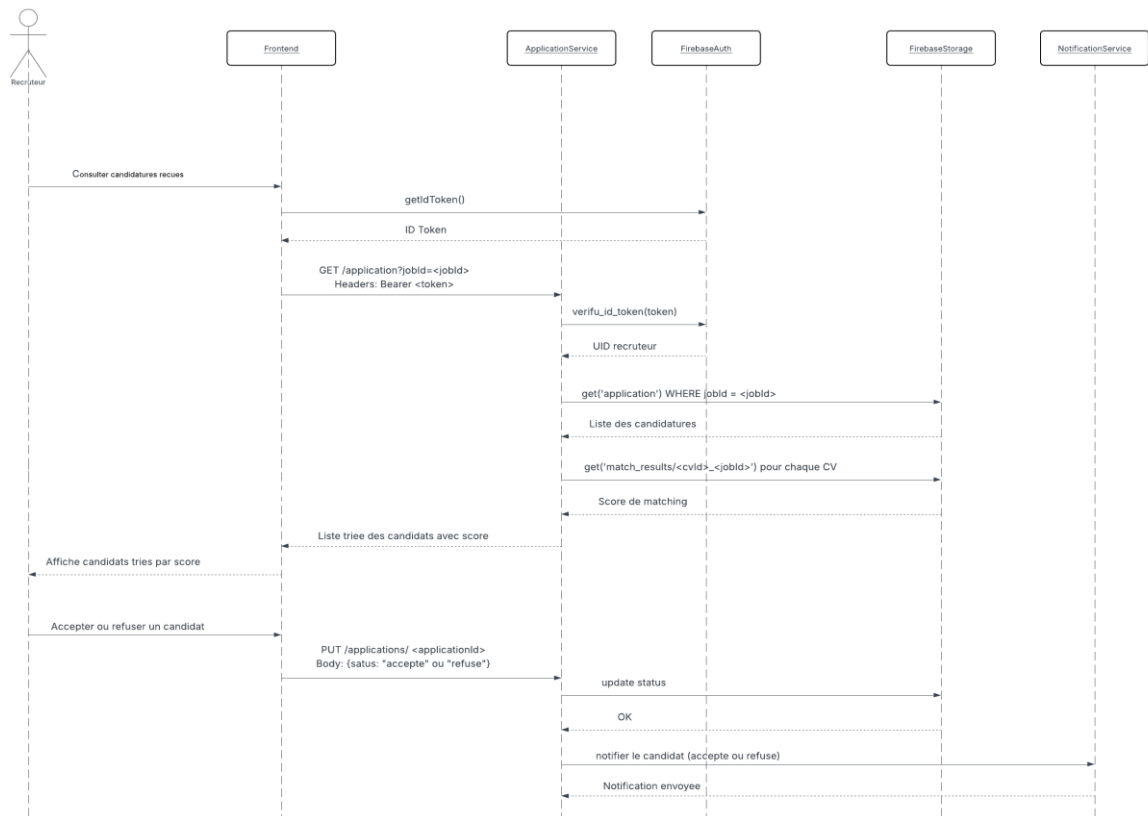
- **Diagramme de séquence Candidat postule, recruteur consulte et décide**

Pour Candidat:



**Figure 9 : Diagramme de séquence Candidat postule, recruteur consulte et décide (candidat)**

Pour Recruteur:



**Figure 10 : Diagramme de séquence Candidat postule, recruteur consulte et décide (Recruteur)**

## b. Diagramme d'activité

- **Diagramme d'activité Postuler à une offre**

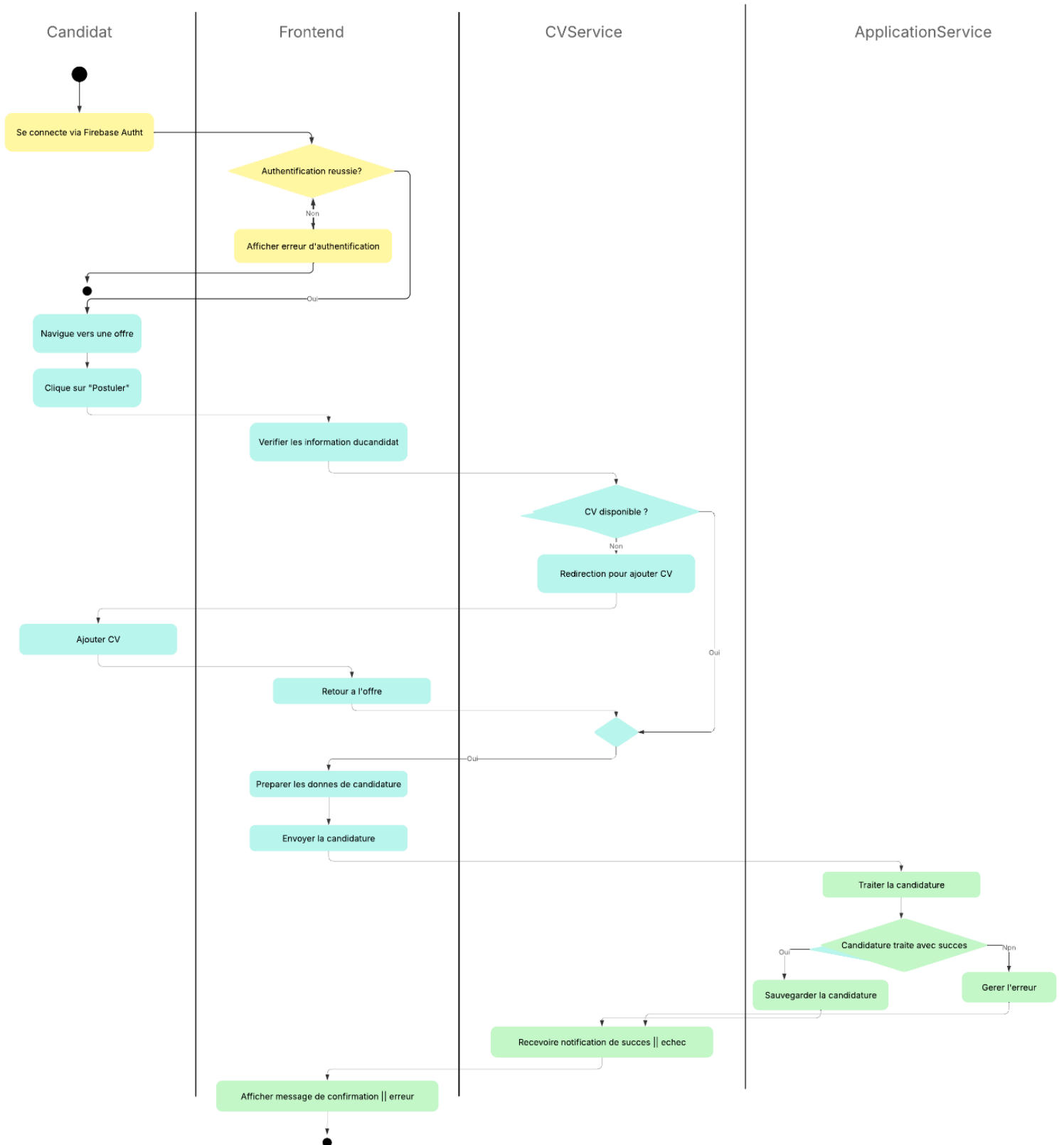
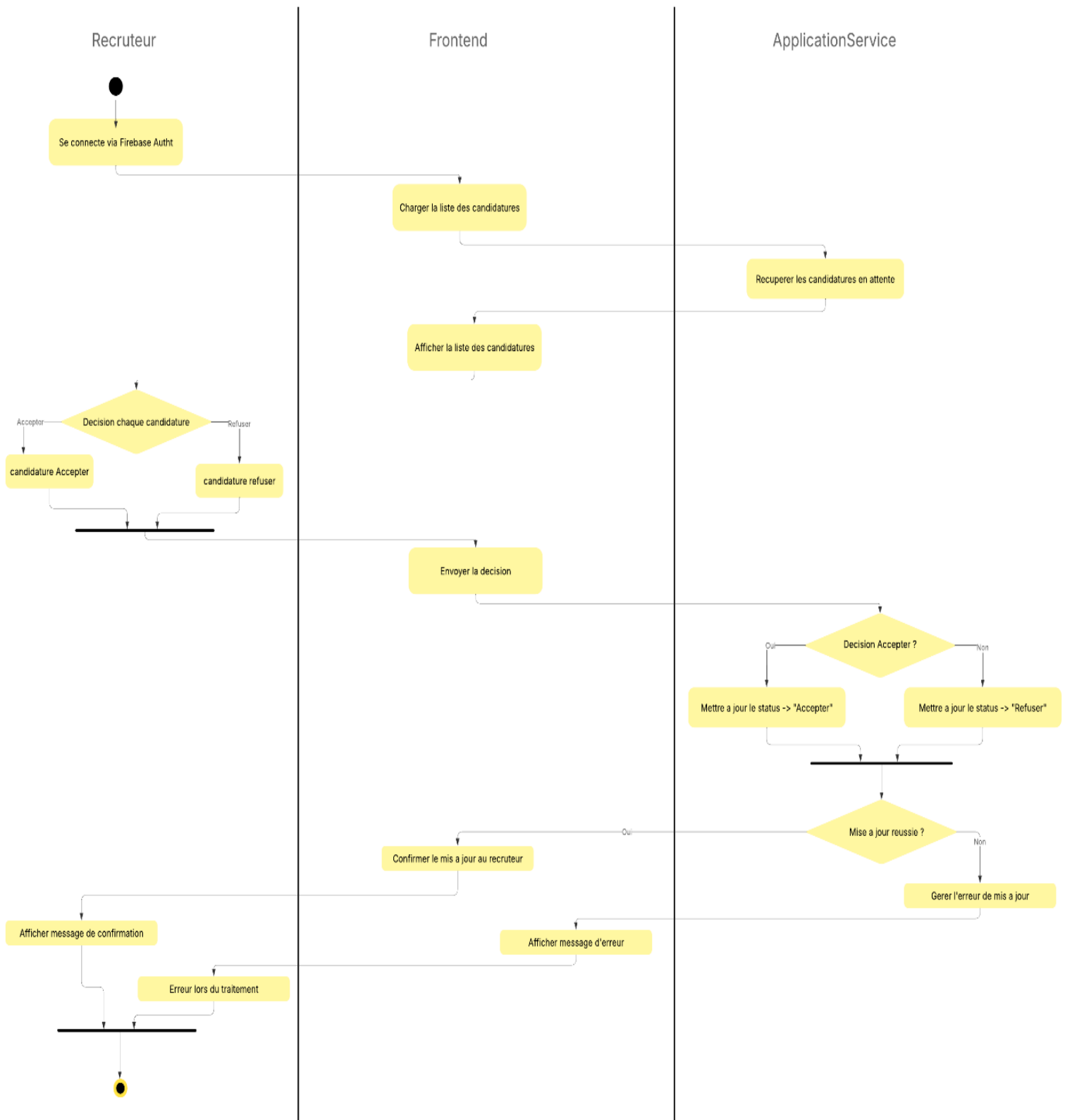


Figure 14 : Diagramme d'activité Postuler à une offre



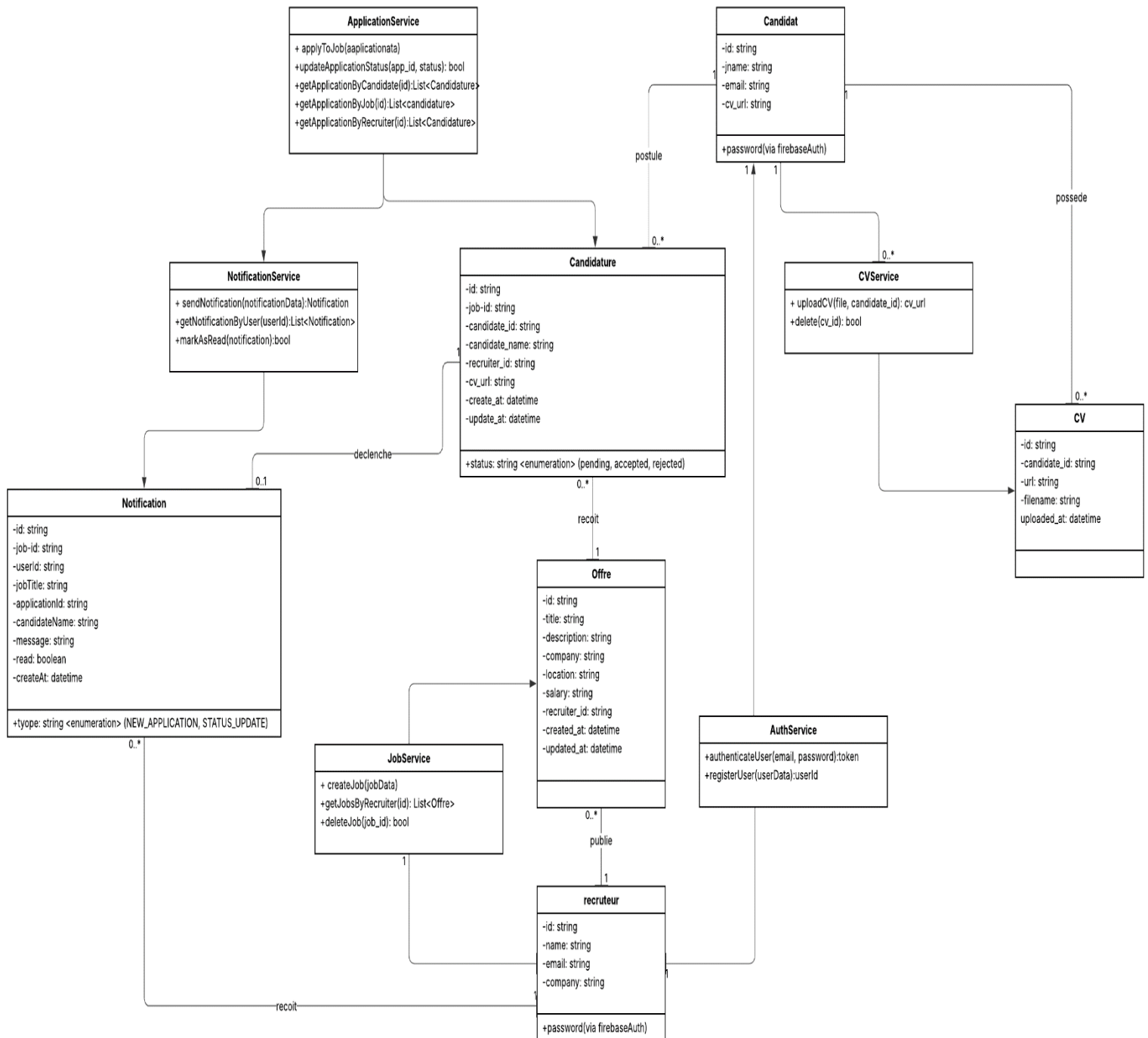
- **Diagramme d'activité Traitement d'une candidature par un recruteur**



**Figure 15 : Diagramme d'activité Traitement d'une candidature par un recruteur**

### a. Diagramme de classes

Le diagramme de classe ci-dessous représente la structure logique du système de gestion des candidatures. Il met en évidence les entités principales (Candidat, Offre, Candidature, Recruteur) ainsi que leurs relations. Chaque classe possède ses attributs essentiels pour assurer le bon fonctionnement du processus de postulation et de gestion des candidatures via nos microservices connectés à Firebase.



**Figure 16 : Diagramme de classes**

## b. Modèle relationnel

Bien que Firebase Firestore soit une base de données NoSQL orientée documents et non relationnelle, nous présentons ici un équivalent du modèle relationnel afin de clarifier la structure logique des collections, sous-collections et les relations entre les entités de notre application.

### • Schéma relationnel global

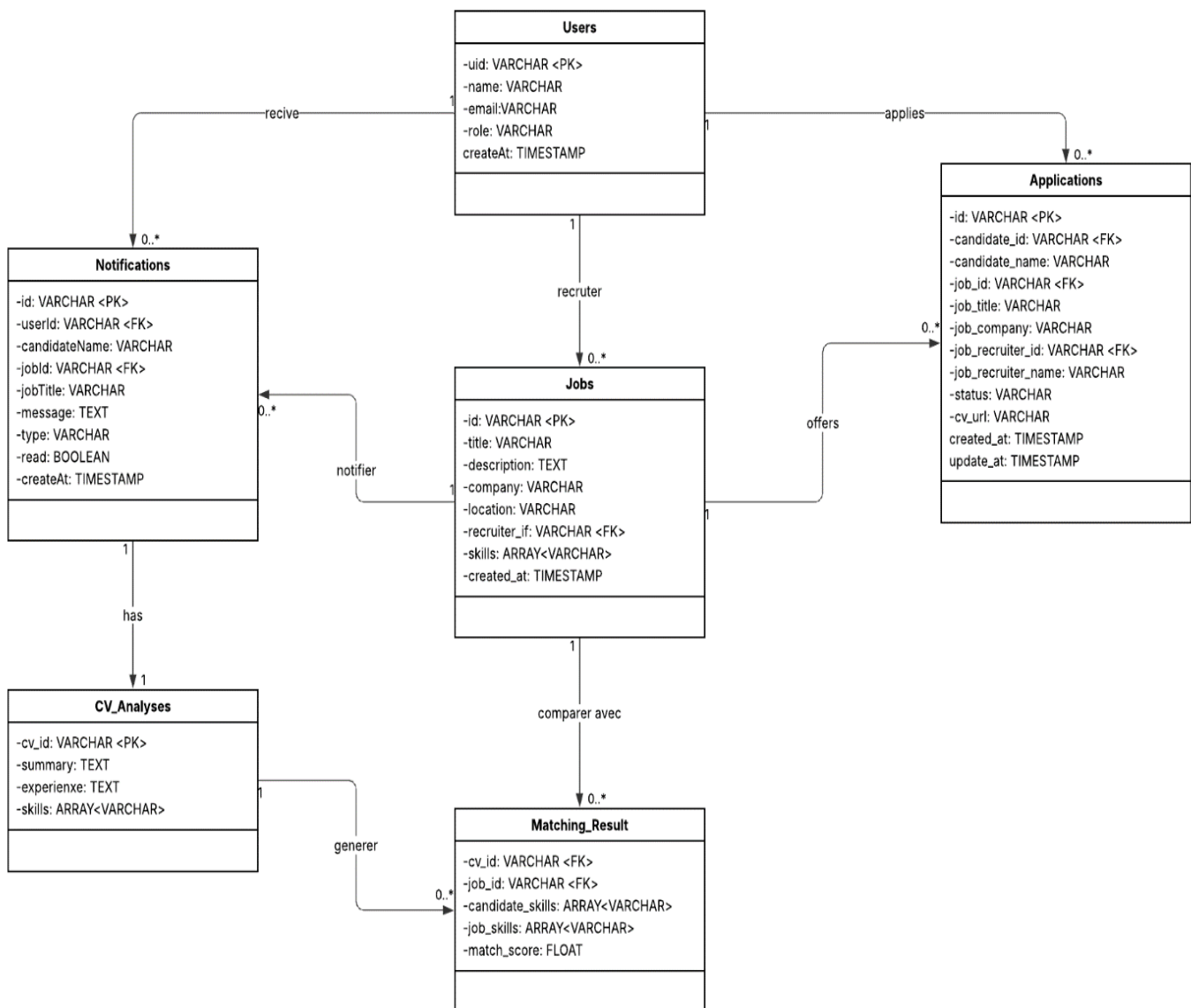


Figure 17 : Schéma relationnel

- **Tableaux des entités**

**Tableau 11 : modèle relationnel Users**

Nom du champ	Type	Description	Clé	Contraintes
Uid	VARCHAR(255)	Identifiant unique de l'utilisateur	Clé primaire	Génère par Firebase Auth
Name	VARCHAR(255)	Nom complet de l'utilisateur	—	Obligatoire
Email	VARCHAR(255)	Adresse email de l'utilisateur	Unique	Format email
Role	VARCHAR(255)	Role de l'utilisateur	—	Valeurs : « candidat », « recruter »
createAt	TIMESTAMP	Date d'inscription	—	Défaut : NOW()

**Tableau 12 : modèle relationnel Jobs**

Nom du champ	Type	Description	Clé	Contraintes
id	VARCHAR(255)	Identifiant unique du job	Clé primaire	Génère automatiquement
Title	VARCHAR(255)	Titre de l'offre	—	Obligatoire
Description	TEXT	Description de l'offre	—	Optionnel
Company	VARCHAR(255)	Nom de l'entreprise	—	Obligatoire
Location	VARCHAR(255)	Localisation	—	Obligatoire
Recruiter_id	ARRAY<VARCHAR>	Identifiant du recruteur (référence vers Users)	Clé étrangère	Obligatoire
Skills	ARRAY<VARCHAR>	Compétences requises	—	Optionnel (liste de chaînes)
Creates_at	TIMESTAMP	Date de création de l'offre	—	Défaut : NOW()

**Tableau 13 : modèle relationnel Applications**

Nom du champ	Type	Description	Clé	Contraintes
id	VARCHAR(255)	Identifiant unique de la candidature	Clé primaire	Génère automatiquement
Created_at	TIMESTAMP	Date de création de la candidature	—	Default ; NOW()
Updated_at	TIMESTAMP	Date de dernière mise à jour de la candidature	—	Optionnel
status	VARCHAR(255)	Statut de la candidature	—	Obligatoire
Cv_url	VARCHAR(255)	Lien ou identifiant du cv stoke	—	Obligatoire
Candidate_id	VARCHAR(255)	Reference vers l'utilisateur candidat (Users.uid)	Clé étrangère	Obligatoire
Candidate_name	VARCHAR(255)	Nom du candidat	—	Optionnel(utilise pur éviter jointures)
Job_id	VARCHAR(255)	Reference vers l'utilisateur candidat (users.uid)	Cle etrangere	Optionnel
Job_title	VARCHAR(255)	Titre de l'offre (copie locale)	—	Optionnel(utilise pour affichage rapide)
Job_company	VARCHAR(255)	Reference vers le recruteur (users.uid)	Clé étrangère	Optionnel
Job_recruiter_name	VARCHAR(255)	Nom du recruteur (copie locale, peut être vide)	—	Optionnel

**Tableau 14 : modèle relationnel CV\_Analyses**

Nom du champ	Type	Description	Clé	Contraintes
Cv_id	VARCHAR(255)	Identifiant unique du cv ( référence au fichier ou a la candidature liée)	Clé primaire	Génère automatiquement
Summary	TEXT	Résumé extrait du CV ( profil du candidat)	—	Optionnel
Expérience	TEXT	Expérience professionnelle extraite du CV	—	Optionnel
Skills	ARRAY<VARCHAR>	Compétences techniques extraites du CV (java, Python, etc.)	—	Obligatoire

**Tableau 15 : modèle relationnel Matching\_Result**

Nom du champ	Type	Description	Clé	Contraintes
Cv_id	VARCHAR(255)	Identifiant unique du cv ( référence vers CV_Analyses.cv_id)	Clé étrangère	Obligatoire
Job_id	TEXT	Identifiant de l'offre correspondante (référence vers job.id)	Clé étrangère	Obligatoire
Candidate_skills	ARRAY<VARCHAR>	Compétences extraites du cv	—	Obligatoire
Job_skills	ARRAY<VARCHAR>	Compétence requises pour l'offre	—	Obligatoire
Match_score	FLOAT	Score de correspondance	—	Obligatoire

**Tableau 16 : modèle relationnel Notifications**

Nom du champ	Type	Description	Clé	Contraintes
id	VARCHAR(255)	Identifiant unique de la notification	Clé primaire	Génère automatiquement
userId	VARCHAR(255)	Identifiant de l'utilisateur concerne (référence vers users.uid)	Clé étrangère	Obligatoire
candidateName	VARCHAR(255)	Nom du candidat lie à l'évènement (copie locale)	—	Optionnel
jobId	VARCHAR(255)	Identifiant de l'offre concernée (référence vers job.id)	Clé étrangère	Optionnel
jobTitle	VARCHAR(255)	Titre de l'offre liée a la notification	—	Obligatoire
Message	TEXT	Contenu de la notification	—	Obligatoire
Type	VARCHAR(255)	Type de notification (NEW_APPLICATION, STATUS_UPDATE, etc.)	—	Obligatoire
Read	BOOLEAN	Status de lecture ( true si lue, false sinon)	—	Défaut : false
createAt	TIMESTAMP	Date de creation de la notification	—	Défaut : NOW()

## c. Architecture de l'application

### i. Architecture logiciel

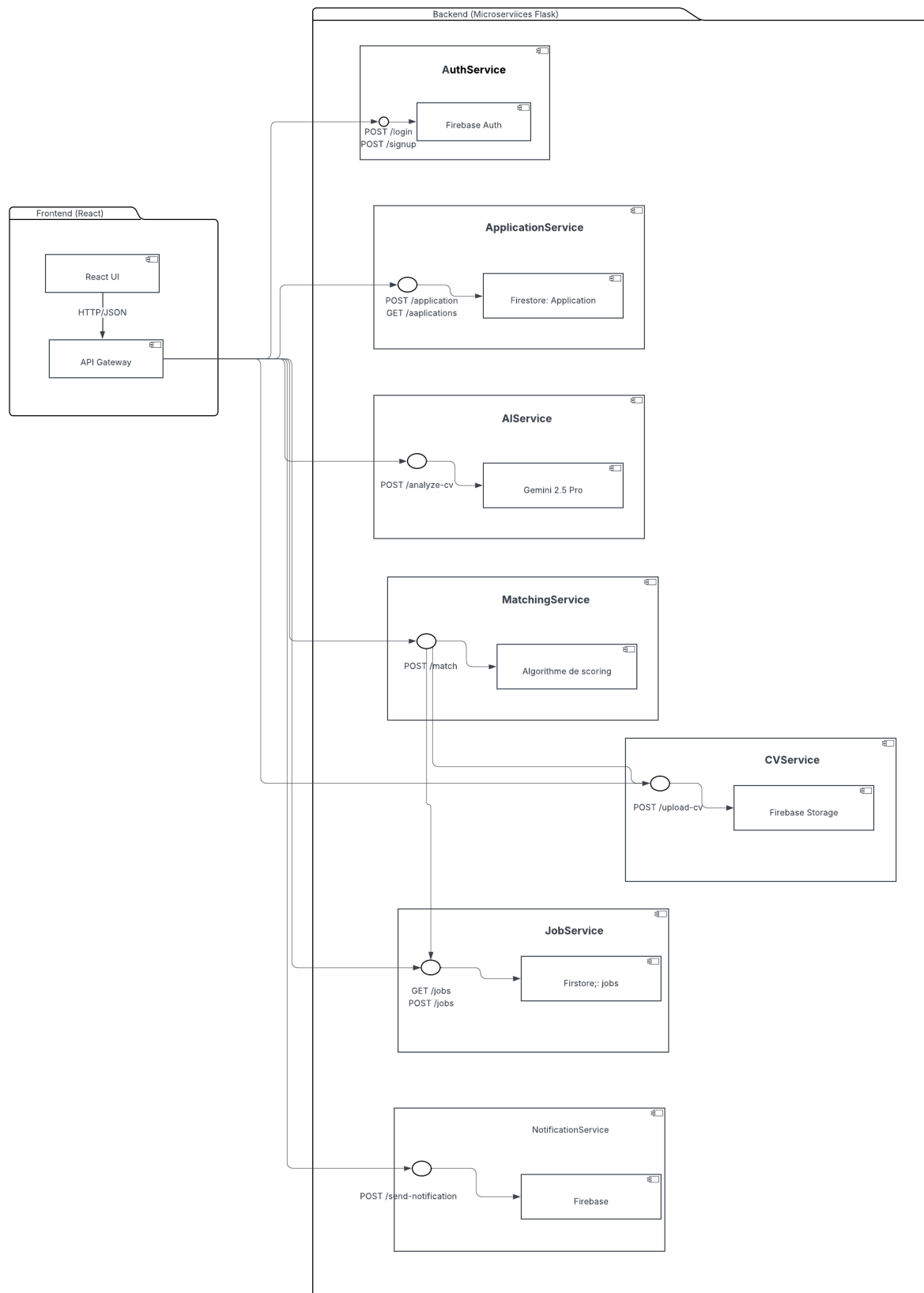


Figure 18 : Diagramme de composant

## 4. Conclusion

À travers ce chapitre, on a pu poser les bases concrètes du système à mettre en place, en traduisant les besoins métiers en représentations claires, cohérentes et exploitables pour la phase de développement.

La **modélisation dynamique** nous a permis de simuler les comportements du système dans différents scénarios (authentification, postulation, traitement des candidatures...) en utilisant des diagrammes de séquence, d'état-transition et d'activité. Cela nous a aidés à mieux comprendre le déroulement logique des interactions entre les utilisateurs et les microservices.

La **modélisation statique**, quant à elle, a défini la structure interne des données et des entités manipulées. Grâce au diagramme de classes, au modèle relationnel et au dictionnaire de données, on a pu structurer nos collections Firebase de façon claire, ce qui facilitera les opérations de lecture, d'écriture et de recherche dans la base.

Enfin, l'**architecture de l'application** a mis en évidence la répartition des responsabilités entre le frontend en React et les différents microservices Flask, tous connectés à Firebase (Auth, Firestore, Storage, FCM). L'objectif est d'assurer un système bien organisé, modulaire, maintenable et évolutif.

Cette phase de conception nous donne une vision d'ensemble précise, à la fois fonctionnelle et technique, qui guidera toutes les étapes suivantes du projet.



# **C** **HAPITRE** **4**

## **Réalisation du système**

# 1. Introduction

Dans ce chapitre, on commence par présenter l'environnement **matériel** (les équipements utilisés) et **logiciel** (outils, technologies, bibliothèques, services cloud comme Firebase) mobilisés tout au long du projet. Ensuite, on décrit les **principales interfaces graphiques** développées côté frontend en React, avec une attention particulière sur l'ergonomie, la navigation et les fonctionnalités proposées aux différents types d'utilisateurs (candidat, recruteur).

Ce cadre technique et visuel constitue la base sur laquelle reposera toute la logique applicative décrite dans les chapitres suivants.

## 2. Environnement de développement

### a. Environnement matériel

Pour le développement de notre application, on a utilisé une machine personnelle avec une configuration assez confortable, ce qui nous a permis de travailler dans de bonnes conditions, surtout avec Firebase et les différents microservices.

- **Ordinateur** : HP Pavilion
- **Processeur** : Intel Core i7, 11<sup>e</sup> génération, 2.8 GHz
- **RAM** : 16 Go
- **Stockage** : SSD de 512 Go
- **Système d'exploitation** : Windows 11
- **Connexion Internet** : Fibre optique (nécessaire pour les tests en ligne et la communication avec Firebase)

*Côté navigateurs, l'application a été testée sur :*

- **Google Chrome** (dernière version)
- **Microsoft Edge** (dernière version)

Cette configuration matérielle nous a permis de coder, tester, déployer et simuler les différents scénarios utilisateurs sans problème.

## b. Environnement logiciel

Pour mener à bien le développement de notre application, on a utilisé un ensemble d'outils et de technologies modernes, adaptés à un système basé sur l'architecture microservices et l'intégration avec Firebase. Ces outils ont été choisis pour leur compatibilité, leur simplicité d'utilisation et leur efficacité en termes de développement et de déploiement.



**Visual Studio** éditeur de code principal utilisé pour le développement côté frontend (React) et backend (Flask).



**Node.js & npm** nécessaires pour gérer le projet React et installer les différentes dépendances.



**React.js** utilisé pour le développement de l'interface utilisateur. Il m'a permis de construire une application frontend dynamique, réactive et bien structurée grâce aux composants.



**Python** langage utilisé pour le développement des microservices côté backend avec Flask.



**Flask** framework web Python léger, utilisé pour créer les microservices RESTful de l'application (authentification, gestion des candidatures, offres, scoring, etc.).



**Postman** pour tester les endpoints des API REST créés (GET, POST, PUT, DELETE).



**Firebase Authentication** pour gérer l'inscription et la connexion des utilisateurs.



**Firebase Firestore** base de données cloud NoSQL pour stocker les informations des utilisateurs, des offres et des candidatures.



**L'API REST** permet d'échanger des données entre applications via des requêtes simples. Elle facilite la création, lecture, mise à jour et suppression d'informations, tout en assurant des échanges rapides et structurés au format JSON.



**Axios** bibliothèque HTTP utilisée dans React pour communiquer avec les microservices Flask.



**L'API Gemini** nous a permis d'analyser automatiquement le contenu des CVs grâce à l'intelligence artificielle. On l'a intégrée dans un microservice dédié pour extraire les compétences, les expériences et d'autres infos utiles afin d'améliorer le matching entre candidats et offres.



**Git** est l'outil qu'on a utilisé pour gérer les versions du projet et suivre les modifications faites tout au long du développement. **GitHub**, de son côté, nous a servi à héberger le code en ligne, centraliser le travail et faciliter la collaboration entre les membres de l'équipe.

### 3. Principales interfaces graphiques

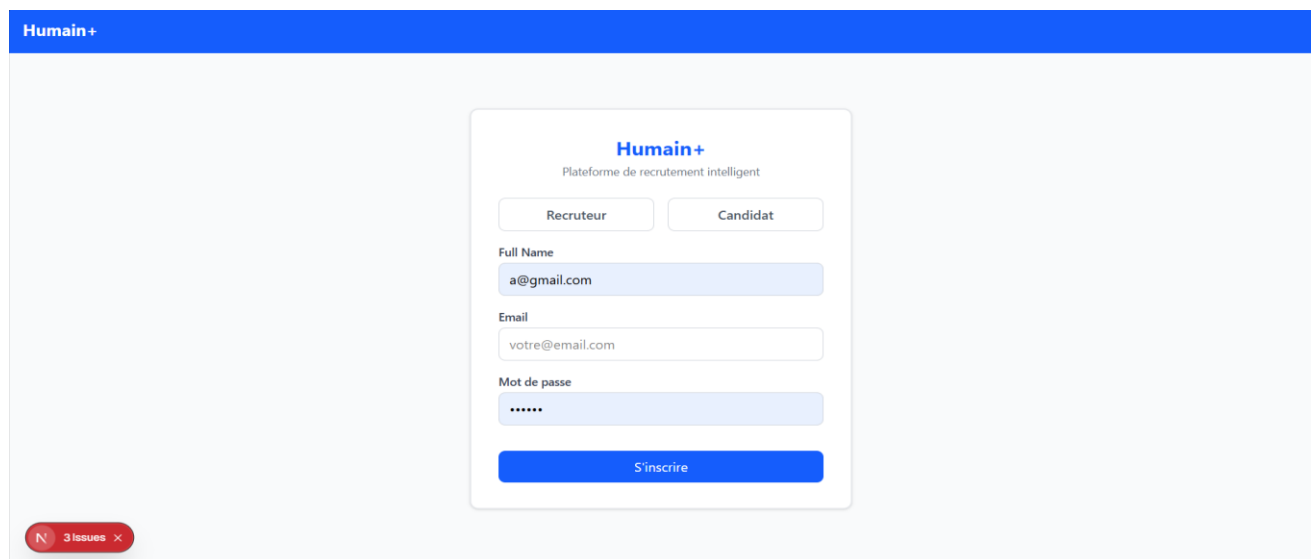
#### a. Login et Sign Up

Dans cette partie, l'utilisateur a la possibilité de créer un compte s'il est nouveau (Sign Up) ou se connecter s'il a déjà un compte (Login).

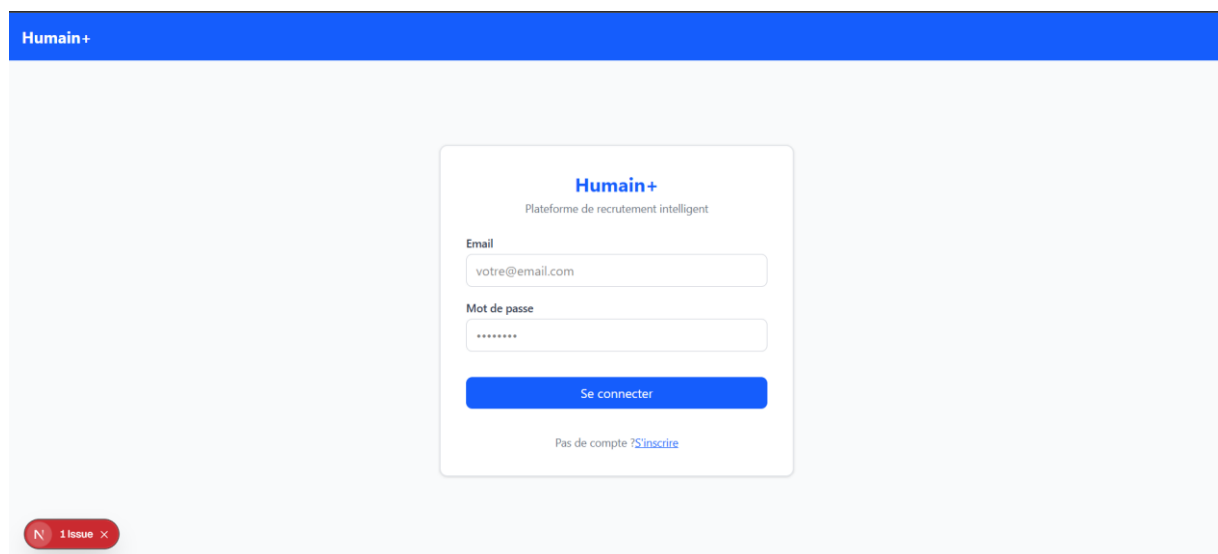
Le formulaire de Sign Up demande les informations de base comme le nom, prénom, email, mot de passe, et le rôle (candidat ou recruteur).

Une fois inscrit, l'utilisateur peut se connecter avec son email et son mot de passe via le formulaire de Login.

J'ai utilisé Firebase Authentication pour gérer la création des comptes, la connexion sécurisée, et la gestion des sessions.



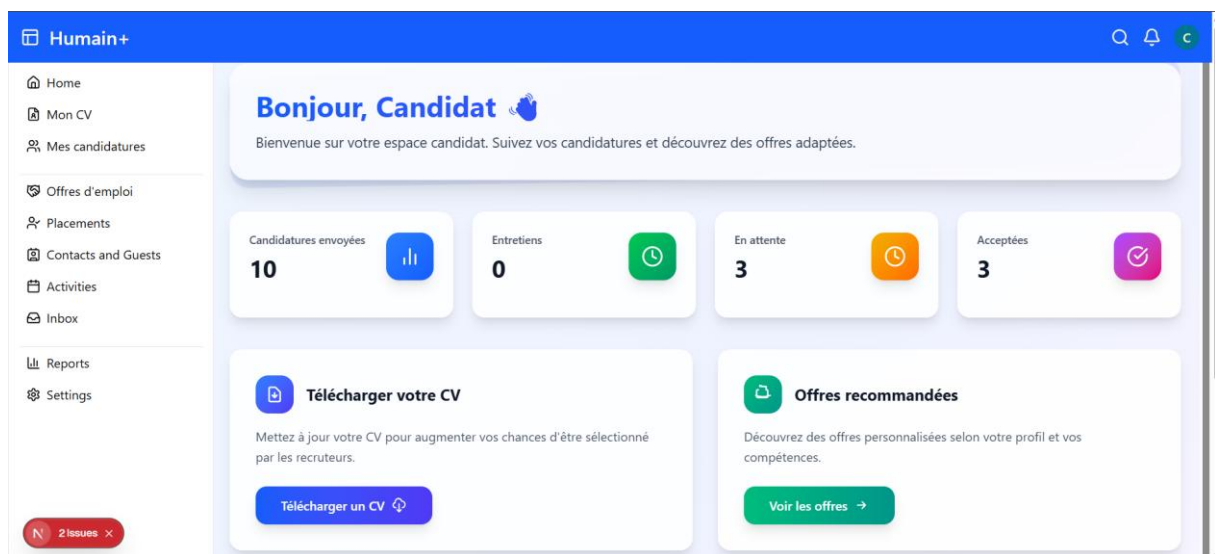
The screenshot shows the 'Humain+' Sign Up interface. At the top is a blue header with the 'Humain+' logo. Below it, a white card contains the 'Humain+' logo and the tagline 'Plateforme de recrutement intelligent'. There are two buttons: 'Recruteur' and 'Candidat'. The form fields are: 'Full Name' (with 'a@gmail.com' entered), 'Email' (with 'votre@email.com' entered), and 'Mot de passe' (with masked characters '\*\*\*\*\*'). A blue 'S'inscrire' button is at the bottom of the card. In the bottom left corner, there is a red notification bubble that says '3 issues'.



The screenshot shows the 'Humain+' Login interface. It has the same blue header and white card layout as the Sign Up page. The card contains the 'Humain+' logo and tagline. The form fields are: 'Email' (with 'votre@email.com' entered) and 'Mot de passe' (with masked characters '\*\*\*\*\*'). A blue 'Se connecter' button is at the bottom of the card. Below the button, there is a link that says 'Pas de compte ? S'inscrire'. In the bottom left corner, there is a red notification bubble that says '1 issue'.

## b. Candidat (Homme)

Cette interface représente le tableau de bord d'un candidat homme après sa connexion. Il peut voir un résumé de ses candidatures : combien il en a envoyées, combien sont en attente, acceptées ou s'il a des entretiens. Il peut aussi téléverser son CV pour augmenter ses chances, et accéder rapidement aux offres recommandées selon son profil. Le menu à gauche lui permet de naviguer facilement entre les différentes pages comme les candidatures, les offres, son CV, les placements, ou encore les paramètres.



### c. Gestion et analyse du CV

Dans cette partie, le candidat peut téléverser son CV au format PDF pour qu'il soit analysé automatiquement. Une fois le fichier déposé, le système extrait les compétences techniques (comme Java, Python, React...), l'expérience professionnelle et un résumé du profil. Cette analyse permet de mieux comprendre le profil du candidat et d'améliorer la correspondance avec les offres. L'interface est simple : on choisit un fichier, on l'upload, puis on obtient un aperçu clair et organisé du contenu du CV.

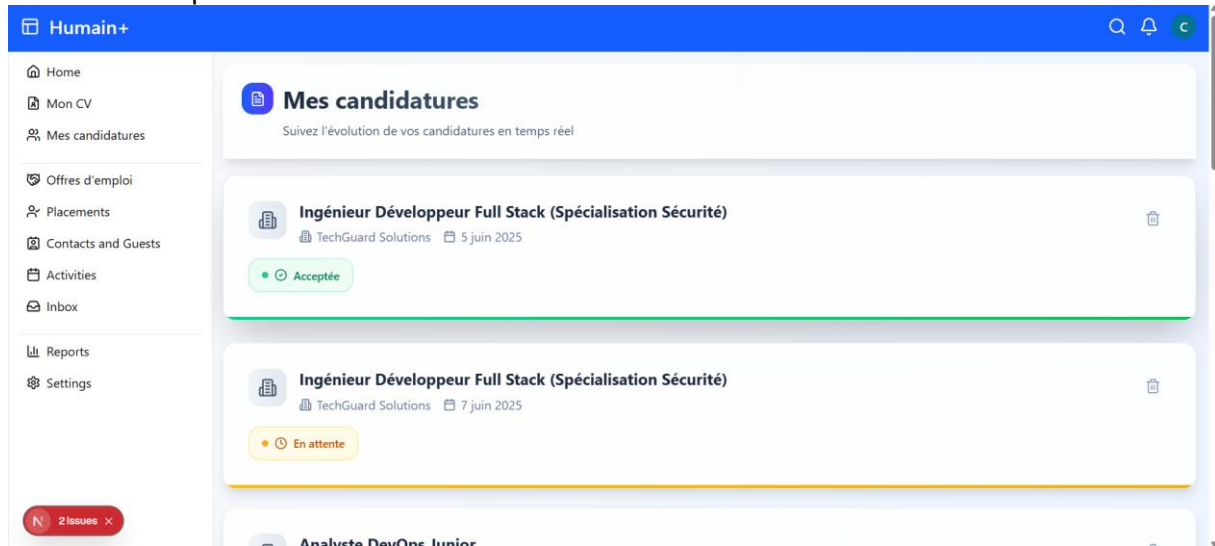






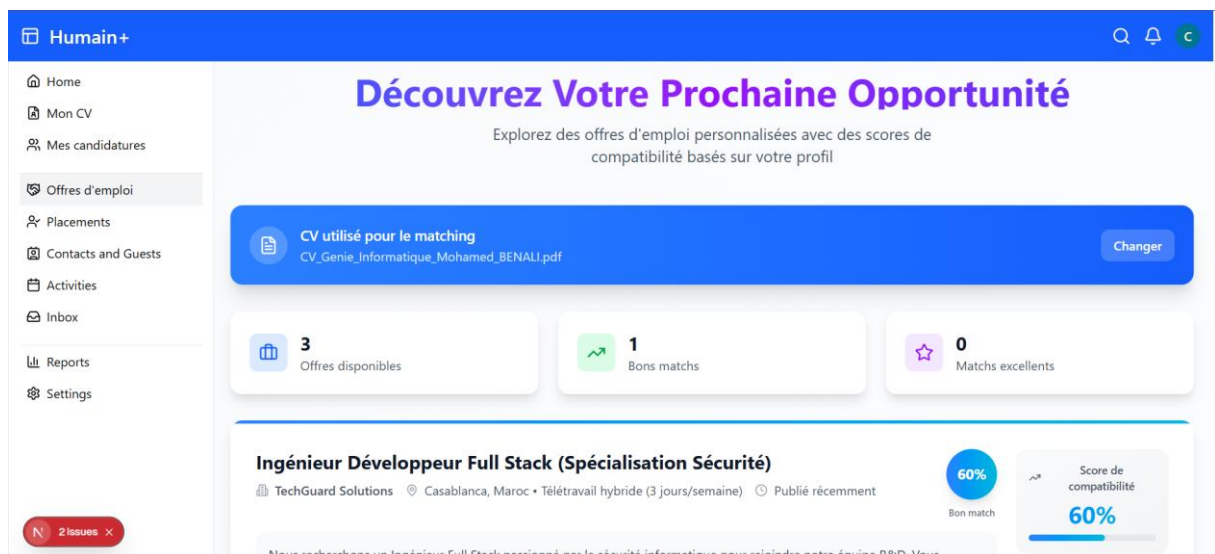
## d. Suivi de Candidature

Dans cette section, le candidat peut suivre l'évolution de sa candidature pour un poste d'Analyste Développeur Junior. L'interface permet de visualiser le statut de la candidature (en attente, en cours, acceptée...) ainsi que des détails clés comme le nom de l'entreprise et la date de dépôt. Un suivi en temps réel est disponible pour rester informé des mises à jour. L'utilisation est simple : consultez l'état actuel et recevez des notifications pour toute nouvelle activité liée à votre candidature.



## e. Offres Recommandées

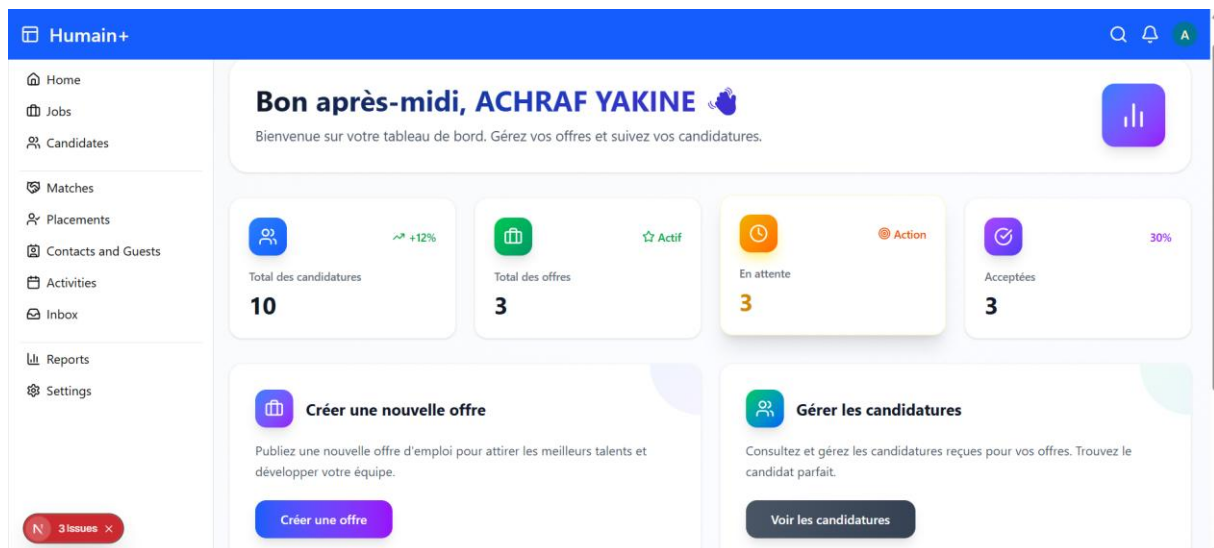
Cette interface affiche les offres d'emploi adaptées au profil du candidat, évaluées grâce à un système de matching intelligent basé sur son CV. Chaque offre présente un score de compatibilité indiquant son adéquation avec le parcours du candidat. Il peut rapidement repérer les "bons matchs" et les "matchs excellents", avec des détails clés comme la localisation, le type de travail et un résumé du poste. Des filtres sont disponibles pour affiner la recherche, ainsi qu'une option pour changer de CV et améliorer la pertinence des résultats.





## f. Tableau de Bord

Cette section permet au recruteur de garder une vue d'ensemble sur ses activités de recrutement. Il peut suivre l'évolution des candidatures, gérer les offres publiées et accéder facilement aux fonctionnalités essentielles. Tout est centralisé pour lui offrir un pilotage simple et efficace de ses recrutements.



## g. Gestion des offres d'emploi

Dans cette section, le recruteur peut gérer toutes les offres qu'il a déjà publiées. Il a une vue claire sur les postes en cours, les villes, les compétences demandées et les missions du poste. Il peut aussi créer une nouvelle offre facilement en remplissant un formulaire rapide. Tout est centralisé pour lui permettre de modifier, publier ou suivre ses offres de manière simple et efficace.

The screenshot displays the 'Offres d'emploi' (Job Offers) section of the Humain+ platform. The interface is divided into a left sidebar with navigation links (Home, Jobs, Candidates, Matches, Placements, Contacts and Guests, Activities, Inbox, Reports, Settings) and a main content area. The main area features a 'Créer une offre' (Create a job offer) button and a 'Vos offres d'emploi' (Your job offers) section. This section contains three job listings:

- Ingénieur Développeur Full Stack (Spécialisation Sécurité)**  
TechGuard Solutions • Casablanca, Maroc • Télétravail hybride (3 jours/semaine)  
Nous recherchons un Ingénieur Full Stack passionné par la sécurité informatique pour rejoindre notre équipe R&D. Vous participerez au développement d'applications sécurisées et à la mise en place de solutions innovantes pour nos clients internationaux.  
Missions principales : Développement d'applications web avec React/Node.js Intégration de fonctionnalités de cybersécurité Gestion de bases de données (Firestore/MySQL) Participation à la conception
- Développeur Mobile Flutter**  
MobileApp Solutions • Marrakech  
Développement d'applications mobiles cross-platform en Flutter (Dart).  
Compétences : Connaissance du cycle de vie d'une application mobile, Intégration, Test unitaire et debug sur Android/iOS, Connaissance des principes UI/UX mobile et services backend via Firebase (Auth, Firestore)
- Analyste DevOps Junior**  
CloudOps Maroc • Rabat  
Participation à la mise en place de pipelines CI/CD et à la surveillance des environnements cloud.  
Compétences : Docker, Jenkins, GitLab CI/CD, Python, Bash, Curiosité technique, rigueur, et capacité à documenter

A red notification bubble in the bottom left corner indicates '3 issues'.

## h. Suivi des candidatures

Cette section est dédiée au suivi des candidatures de personnes en situation de handicap. Elle permet au recruteur de gérer ces candidatures de façon simple et structurée. Il peut suivre le nombre total de candidatures, voir celles qui sont en attente ou déjà traitées, et prendre les décisions nécessaires. C'est un outil qui facilite le processus de recrutement inclusif tout en assurant un bon suivi des profils concernés.

**Humain+**

**Candidatures reçues**  
Gérez vos candidatures en temps réel

**TOTAL CANDIDATURES**  
10  
Toutes vos opportunités

**EN ATTENTE**  
3  
Action requise

**TRAITÉES**  
7  
Décisions prises

**Candidatures en attente** 3

**Ingénieur Développeur Full Stack (Spécialisation Sécurité)** 60% Match

**CANDIDAT**  
chouaib yakine

**ENTREPRISE**  
TechGuard Solutions

**COMPÉTENCES CLÉS:**  
Java Python C/C++ JavaScript PHP HTML5 CSS3 React Node.js MySQL

**RÉSUMÉ DU PROFIL:**  
Mohamed BENALI est un ingénieur en génie informatique diplômé de l'École Nationale Supérieure d'Informatique de Rabat, spécialisé en réseaux et sécurité et développement logiciel. Il possède une expérience professionnelle en tant que développeur full stack, démontrée par un stage chez SoftTech Solutions. Son expertise technique couvre plusieurs langages de programmation, des bases de données, des outils DevOps et des systèmes d'exploitation. Il est également certifié Google IT Support Professional et a des connaissances en cybersécurité. Ses centres d'intérêt incluent la sécurité informatique, l'IA et les technologies web.

Postulé le 7 juin 2025

**Accepter**  
**Refuser**  
**Voir CV**

## i. Suivi des candidatures

Cette section permet au recruteur de suivre l'avancement des candidatures déjà traitées. Il peut voir combien ont été acceptées ou refusées, avec les informations nécessaires pour garder une trace claire des profils retenus. C'est un moyen simple pour gérer les retours et avoir une vue d'ensemble sur l'état du recrutement.

## j. Tableau de bord

## 4. Conclusion

Pour mener à bien le développement de notre application, il était essentiel de s'appuyer sur un environnement solide, cohérent et adapté à nos besoins spécifiques. De l'infrastructure matérielle jusqu'aux outils logiciels, chaque élément a été choisi de manière réfléchie pour garantir efficacité, productivité et fluidité tout au long du projet.

Nous avons opté pour un stack moderne : React.js pour un frontend interactif et responsive, Flask avec Python pour construire des microservices backend légers, et Firebase Firestore pour une base de données temps réel, flexible et scalable. Des outils comme Postman, Git/GitHub, ou encore Visual Studio Code ont renforcé notre capacité à tester, versionner et maintenir le projet de manière professionnelle.

En somme, cet environnement de développement nous a permis de transformer nos idées en une application fonctionnelle, robuste et bien structurée.

## Conclusion générale

Ce projet m'a permis de concevoir et de développer une plateforme complète de gestion des candidatures, répondant aux besoins des candidats comme des recruteurs. À travers l'intégration de fonctionnalités clés telles que le tableau de bord personnalisé, le système de matching intelligent basé sur le CV, le suivi des candidatures en temps réel et la gestion des offres publiées, l'ensemble du processus de recrutement a été centralisé et simplifié pour une meilleure expérience utilisateur.

Ce stage a été une réelle opportunité de mettre en pratique mes compétences techniques (React, Flask, Firebase...) tout en approfondissant ma compréhension des enjeux liés aux systèmes d'information RH. Il m'a également permis de renforcer mes capacités en gestion de projet, et en résolution de problèmes concrets en contexte professionnel.

En somme, cette expérience a été à la fois formatrice et enrichissante, et marque une étape importante dans mon parcours en génie informatique. Elle ouvre la voie à de futures améliorations du système et à de nouvelles perspectives professionnelles dans le domaine du développement d'applications web intelligentes et orientées utilisateur.

# Références bibliographiques

## Front-end & UI

[React.js] : <https://reactjs.org/docs/getting-started.html>

[Tailwind CSS] : <https://tailwindcss.com/docs/installation>

## Back-end & Microservices

[Flask Framework] : <https://flask.palletsprojects.com/en/latest/>

[REST API Design] : <https://restfulapi.net/>

[Microservices Architecture] : <https://microservices.io/>

## Firebase

[Firebase Authentication] : <https://firebase.google.com/docs/auth>

[Firebase Firestore] : <https://firebase.google.com/docs/firestore>

## Outils et bibliothèques complémentaires

[Axios (HTTP client for React)] : <https://axios-http.com/docs/intro>

[Postman (REST Client)] : <https://www.postman.com/>

## Traitement de texte et NLP

[Python NLP - NLTK] : <https://www.nltk.org/>

## Diagrammes et modélisation

[UML Diagrams] : <https://lucid.app/>

## Autres

[Gemini API] : [https://aistudio.google.com/prompts/new\\_chat](https://aistudio.google.com/prompts/new_chat)



# ANNEXES

## Annexe A : Code source du microservice ApplicationService.py (gestion des candidatures)

Ce microservice Flask gère les candidatures dans l'application : création, lecture, mise à jour, suppression, et notifications.

### Initialisation Firebase ET Flask

```
from flask import Flask, request, jsonify
import firebase_admin
from firebase_admin import credentials, firestore
from flask_cors import CORS
import uuid
from datetime import datetime

cred = credentials.Certificate("chemin/vers/firebase_admin_key.json")
firebase_admin.initialize_app(cred)
db = firestore.client()

app = Flask(__name__)
CORS(app)
```

### POST : Création d'une candidature

```
@app.route('/applications', methods=['POST'])
def apply_to_job():
    data = request.get_json()

    # Vérification des champs obligatoires
    required = ['job_id', 'job_title', 'candidate_id', 'candidate_name',
'cv_url', 'skills']
    if not all(f in data for f in required):
        return jsonify({'error': 'Données manquantes'}), 400

    # Vérification de l'offre d'emploi
    job = db.collection('jobs').document(data['job_id']).get()
    if not job.exists:
        return jsonify({'error': 'Job non trouvé'}), 404

    job_data = job.to_dict()

    application = {
        'id': str(uuid.uuid4()),
        'job': {
            'id': data['job_id'],
            'title': data['job_title'],
            'recruiter_id': job_data['recruiter_id'],
            'recruiter_name': job_data.get('recruiter_name', ''),
            'company': job_data.get('company', '')
        },
        'candidate': {
            'id': data['candidate_id'],
            'name': data['candidate_name']
        },
        'cv_url': data['cv_url'],
```

```

        'skills': data['skills'],
        'summary': data.get('summary', ''),
        'created_at': datetime.utcnow().isoformat(),
        'status': 'pending',
        'match_score': data.get('match_score', 0)
    }

    # Enregistrement de la candidature

db.collection('applications').document(application['id']).set(application)

# Notification pour le recruteur
notification = {
    'id': str(uuid.uuid4()),
    'userId': job_data['recruiter_id'],
    'type': 'NEW_APPLICATION',
    'applicationId': application['id'],
    'jobId': data['job_id'],
    'jobTitle': data['job_title'],
    'candidateName': data['candidate_name'],
    'message': f"Nouvelle candidature pour {data['job_title']}",
    'read': False,
    'createdAt': datetime.utcnow()
}

db.collection('notifications').document(notification['id']).set(notification)

return jsonify(application), 201

```

### GET : Candidatures d'un candidat

```

@app.route('/applications/candidate/<candidate_id>', methods=['GET'])
def get_candidate_applications(candidate_id):
    query = db.collection('applications').where('candidate.id', '==',
candidate_id).stream()
    return jsonify([doc.to_dict() for doc in query]), 200

```

### GET : Candidatures pour une offre

```

@app.route('/applications/job/<job_id>', methods=['GET'])
def get_job_applications(job_id):
    job = db.collection('jobs').document(job_id).get()
    if not job.exists:
        return jsonify({'error': 'Job non trouvé'}), 404

    query = db.collection('applications').where('job.id', '==',
job_id).stream()
    return jsonify([doc.to_dict() for doc in query]), 200

```

### PUT : Mise à jour du statut d'une candidature

```

@app.route('/applications/<application_id>/status', methods=['PUT'])
def update_application_status(application_id):
    data = request.get_json()
    new_status = data.get('status')

    if new_status not in ['accepted', 'rejected', 'pending']:

```

```

        return jsonify({'error': 'Statut invalide'}), 400

    app_ref = db.collection('applications').document(application_id)
    if not app_ref.get().exists:
        return jsonify({'error': 'Candidature non trouvée'}), 404

    app_ref.update({'status': new_status, 'updated_at':
datetime.utcnow().isoformat()})
    return jsonify({'message': 'Statut mis à jour'}), 200

```

### DELETE : Suppression d'une candidature

```

@app.route('/applications/<application_id>', methods=['DELETE'])
def delete_application(application_id):
    ref = db.collection('applications').document(application_id)
    if not ref.get().exists:
        return jsonify({'error': 'Candidature non trouvée'}), 404

    ref.delete()
    return jsonify({'message': 'Candidature supprimée'}), 200

```

### GET : Candidatures pour un recruteur

```

@app.route('/applications/recruiter/<recruiter_id>', methods=['GET'])
def get_recruiter_applications(recruiter_id):
    query = db.collection('applications')\
        .where('job.recruiter_id', '==', recruiter_id)\
        .stream()
    return jsonify([doc.to_dict() for doc in query]), 200

```

### Lancement du service

```

if __name__ == '__main__':
    app.run(port=5005, debug=True)

```

## Annexe B : Extraits du code des microservices d'analyse de CV, extraction de texte et matching des compétences

### Analyse du CV avec Gemini (AIService)

```

def analyze_with_ai(cv_text):
    prompt = f"""Analyse le CV ci-dessous et retourne les informations dans
ce format EXACT : ..."""
    response = model.generate_content(prompt)
    return {'analysis': getattr(response, 'text', 'Aucune réponse')}

```

Cette fonction envoie le contenu texte du CV à Gemini et récupère une réponse structurée contenant les compétences, l'expérience et un résumé.

### Extraction du texte du PDF

```

def extract_text_from_pdf(filepath):
    with open(filepath, 'rb') as f:
        reader = PyPDF2.PdfReader(f)
        return ''.join(page.extract_text() or '' for page in
reader.pages).strip()

```

Ce bloc lit le fichier PDF page par page et en extrait tout le texte brut.

#### Matching des compétences (MatchingService)

```
def calculate_score(candidate_skills, job_skills):  
    match_count = len(set(candidate_skills) & set(job_skills))  
    return round((match_count / len(job_skills)) * 100, 2)
```

Ce calcul donne un **score de compatibilité (%)** entre les compétences du candidat et celles demandées dans l'offre.

#### Exemple de route Flask pour matcher un CV avec une offre

```
@app.route('/match/<cv_id>/<job_id>', methods=['GET'])  
def match_candidate(cv_id, job_id):  
    # Récupération des compétences + calcul du score  
    ...  
    return jsonify({'match_score': score})
```

Cette route est appelée pour obtenir un score de matching entre un candidat (CV) et une offre précise.