



République Tunisienne

Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique

École Supérieure Privée d'ingénierie et de technologie

TEK-UP



RAPPORT DE PROJET DE FIN D'ÉTUDES

Présenté en vue de l'obtention du

Diplôme National d'Ingénieur en Sciences Appliquées et Technologiques

Spécialité : Génie Logiciel et Système d'Informations

Réalisé par

Mohamed Amine Ayed

REFONTE DU MODULE DE GESTION DES CANDIDATS : DE L'OUTIL BYBLOS VERS UNE ARCHITECTURE MICROSERVICES

Encadrant professionnel : **Madame Mariem Chater**

Scrum Master

Encadrant académique : **Madame Asma Mabrouk**

Maître Assistant(e)

J'autorise l'étudiant à faire le dépôt de son rapport de stage en vue d'une soutenance.

Encadrant professionnel, **Madame Mariem Chater**

Signature et cachet

J'autorise l'étudiant à faire le dépôt de son rapport de stage en vue d'une soutenance.

Encadrant académique, **Madame Asma Mabrouk**

Signature

Dédicace

Je dédie ce travail, comme preuve de respect, de gratitude, et de reconnaissance à :

Ma chère famille, pour son affection, sa patience, et ses prières.

Mes meilleurs amis pour leur aide, leur temps, leurs encouragements, leur assistance et leur soutien.

A tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

Toute ma reconnaissance...

Remerciements

C'est avec un grand plaisir que je réserve cette page en signe de gratitude et de profonde reconnaissance à tous ceux qui ont bien voulu apporter l'assistance nécessaire au bon déroulement de ce travail.

Je remercie tout d'abord **Madame Asma Mabrouk**, maître assistante à TEK-UP, pour son soutien continue et son attention qu'elle n'a cessé de m'apporter tout au long de ce projet en tant qu'encadrante académique. Je tiens à lui exprimer ma gratitude pour apport crucial à ce travail.

Ce projet a été effectué au sein de Talan Tunisie. J'adresse mes sincères remerciements à **Madame Mariem Chater**, le scrum master de l'équipe Byblos, et mon encadrante professionnelle, pou m'avoir aidé à bien mener ce projet, pour son soutien scientifique et sa disponibilité.

Je tiens à remercier également **Monsieur Hassen Besbes**, Ingénieur senior en java, et mon encadrant technique, lors de ce projet, pour m'aider aux lacunes et aux difficultés techniques et aussi à l'esprit logique de savoir-faire à la face des tâches.

Merci aussi à mes amis qui se reconnaîtront ici, merci pour les moments d'amitié que nous avons partagés. Vous avez toujours été présents pendant les moments les plus difficiles de ma vie professionnelle, j'ai eu beaucoup de chance de vous avoir comme amis.

Table des matières

Introduction générale	1
Chapitre 1: État de l'art	2
I Cadre général du projet	2
I.1 Présentation de l'organisme d'accueil	2
I.1.1 Présentation générale	2
I.1.2 Secteurs d'activité	3
I.2 Contexte du projet	4
I.2.1 Présentation de l'outil BYBLOS	4
II Problématique	4
III Solution proposée : Migration vers une architecture microservices	6
IV Architecture microservices	6
IV.1 Caractéristiques d'un microservice	7
IV.2 Architecture monolithique Vs microservices	8
V Méthodologie de développement adoptée : Scrum	9
V.1 Présentation de la méthode SCRUM	9
V.2 Les rôles dans SCRUM	10
V.3 Cycle de vie de Scrum	10
Chapitre 2: Analyse et spécification des besoins	12
I Identification des acteurs	12
II Identification des besoins	12
II.1 Les besoins fonctionnels	13
II.2 Les besoins non fonctionnels	13
III Diagramme de cas d'utilisation global	13
IV Diagrammes des cas d'utilisation détaillés	14
IV.1 Raffinement des cas d'utilisation pour le back-office	14
IV.1.1 Cas d'utilisation "Gérer une fiche candidat"	14
IV.1.2 Cas d'utilisation "Gérer un entretien"	16
IV.2 Raffinement des cas d'utilisation pour le recruteur	19

IV.2.1	Cas d'utilisation "Valider un entretien"	19
IV.2.2	Cas d'utilisation "Rechercher un candidat"	21
Chapitre 3:	Conception	23
I	Conception générale	23
I.1	Diagramme de déploiement de l'application	23
I.2	Architecture logique de l'application	24
I.3	Diagramme de paquetage	25
II	Conception détaillée	26
II.1	Diagramme de classes	26
II.2	Diagrammes de séquences	28
II.2.1	Ajouter une fiche candidat	28
II.2.2	Rechercher un candidat	29
II.2.3	Planifier un entretien	29
II.2.4	Accepter un entretien	30
II.2.5	Rejeter un entretien	31
II.2.6	Modifier une fiche candidat	32
Chapitre 4:	Réalisation	33
I	Environnement de travail	33
I.1	Environnement matériel	33
I.2	Environnement logiciel	33
I.3	Technologies utilisées	34
II	Les interfaces utilisateur	35
II.1	Interface d'authentification	35
II.2	Interface d'accueil	36
II.3	Interface d'ajout d'une fiche candidat	37
II.4	Interface de la recherche d'un candidat	41
II.5	Interface du résultat de la recherche	41
II.6	Interface du fichier excel contenant le résultat de la recherche	42
II.7	Interface de la planification d'un entretien	43
II.8	Interface contenant l'e-mail de validation de l'entretien	43
II.9	Interface contenant l'e-mail du rejet de l'entretien	43
II.10	Interface contenant les différents statuts des entretiens	44

II.11	Interface de la consultation du Workflow d'un entretien en attente	44
II.12	Interface de la consultation du Workflow d'un entretien accepté	45
II.13	Interface de la consultation du Workflow d'un entretien rejeté	45
Conclusion générale		47
Nétographie		48

Table des figures

1.1	Implantation de Talan dans le monde	3
1.2	Architecture monolithique	5
1.3	Architecture monolithique de BYBLOS	6
1.4	Comparaison entre les architectures microservices et monolithique	7
1.5	Cycle de vie de SCRUM	10
2.1	Diagramme de cas d'utilisation global	14
2.2	Diagramme de cas d'utilisation "Gérer une fiche candidat"	15
2.3	Diagramme de cas d'utilisation "Gérer un entretien"	16
2.4	Diagramme de cas d'utilisation "Valider un entretien"	20
2.5	Diagramme de cas d'utilisation "Rechercher un candidat"	21
3.1	Diagramme de déploiement de l'application	24
3.2	Architecture logique du module de gestion des candidats	25
3.3	Diagramme de paquetage	26
3.4	Diagramme de classes	27
3.5	Diagramme de séquence "Ajouter une fiche candidat"	28
3.6	Diagramme de séquence "Rechercher un candidat"	29
3.7	Diagramme de séquence "Planifier un entretien"	30
3.8	Diagramme de séquence "Accepter un entretien"	31
3.9	Diagramme de séquence "Rejeter un entretien"	31
3.10	Diagramme de séquence "Modifier une fiche candidat"	32
4.1	Interface de Eureka Discovery Service	35
4.2	Interface d'authentification	36
4.3	Interface d'accueil	37
4.4	Interface Identité	37
4.5	Interface Contact	38
4.6	Interface Expérience	38
4.7	Interface Documents	39
4.8	Interface Planification des entretiens	39
4.9	Interface Contrat	40

4.10	Interface Fonction	40
4.11	Interface Rémunération	41
4.12	Interface de la recherche d'un candidat	41
4.13	Interface du résultat de la recherche	42
4.14	Interface du fichier excel contenant le résultat de recherche	42
4.15	Interface de la planification d'un entretien	43
4.16	Interface contenant l'e-mail de validation de l'entretien	43
4.17	Interface contenant l'e-mail du rejet de l'entretien	44
4.18	Interface contenant les différents statuts des entretiens	44
4.19	Interface de la consultation du Workflow d'un entretien en attente de validation	45
4.20	Interface de la consultation du Workflow d'un entretien accepté	45
4.21	Interface de la consultation du Workflow d'un entretien rejeté	46

Liste des tableaux

1.1	Architecture microservices Vs monolithique	9
2.1	Description du cas d'utilisation "Ajouter une fiche candidat"	15
2.2	Description du cas d'utilisation "Modifier une fiche candidat"	16
2.3	Description du cas d'utilisation "Planifier un entretien"	17
2.4	Description du cas d'utilisation "Modifier un entretien"	18
2.5	Description du cas d'utilisation "Consulter le KPI des avis des entretiens"	18
2.6	Description du cas d'utilisation "Consulter le workflow d'un entretien"	19
2.7	Description du cas d'utilisation "Accepter un entretien"	20
2.8	Description du cas d'utilisation "Rejeter un entretien"	21
2.9	Description du cas d'utilisation "Exporter le résultat dans un fichier excel"	22

Introduction générale

De nos jours, l'évolution des technologies de l'information et de la communication a été accompagnée de profonds changements ayant affecté les processus métier au sein de l'entreprise qui doit dorénavant s'adapter rapidement aux nouvelles technologies pour assurer son évolution et sa pérennité.

En effet, l'équipe d'intégration doit suivre les évolutions technologiques qui ont apporté des changements dans les architectures logicielles. Ces changements ont pour objectif de développer des applications plus performantes, maintenables et extensibles.

À cet égard, devant un développement exponentiel et continu de ses processus métier, tels que le processus de gestion des candidats qui est indispensable pour toute entreprise, il faut faire face à certains problèmes qui peuvent ralentir le fonctionnement de ses applications existantes tel que : l'augmentation du temps de réponse, la redondance des modules et la difficulté de maintenance.

C'est dans ce cadre que s'inscrit notre projet de fin d'études du cycle des ingénieurs à l'École Supérieure Privée de Technologies et d'Ingénierie(**TEK-UP**) réalisé à **Talan Tunisie Consulting**, société de services en ingénierie informatique. Notre tâche consiste à assurer une refonte architecturale du module de gestion des candidats de l'ERP Byblos de Talan, en passant de l'architecture monolithique vers une architecture à base de microservices. L'objectif étant d'avoir une structure robuste et fiable qui allège la complexité de l'architecture existante.

Le présent rapport décrit les différentes étapes de notre travail, et s'articule autour de quatre chapitres :

- Le premier chapitre comporte une brève présentation de l'organisme d'accueil et du cadre général de ce projet. Il expose aussi l'étude de l'existant et met l'accent sur la solution proposée et explique l'architecture microservices et la méthodologie de travail Scrum.
- Le deuxième chapitre présente, en premier lieu, une analyse détaillée des acteurs, des besoins fonctionnels et non fonctionnels du module de gestion des candidats de Byblos. En second lieu, il décrit le cas d'utilisation général ainsi que l'architecture globale de notre solution.
- Le troisième chapitre détaille la conception de ce module.
- Le quatrième chapitre illustre l'intégration du micro-service tout en exposant les choix technologiques utilisés pour la réalisation de notre solution, ainsi que les résultats obtenus selon ces technologies.
- Nous clôturons par une conclusion générale qui présente une récapitulation du travail réalisé et ouvre quelques perspectives.

ÉTAT DE L'ART

Plan

I	Cadre général du projet	2
II	Problématique	4
III	Solution proposée : Migration vers une architecture microservices	6
IV	Architecture microservices	6
V	Méthodologie de développement adoptée : Scrum	9

Introduction

Dans ce chapitre introductif, nous présentons, dans un premier temps, l'organisme d'accueil : la société **Talan Tunisie** et ses domaines d'activité. En se basant sur une étude de l'existant, nous définissons ensuite, la problématique et notre solution proposée. Dans la quatrième partie, nous explorons les concepts clés du domaine de travail et nous finissons par présenter la méthodologie adoptée lors de la réalisation du projet.

I Cadre général du projet

Cette première section du rapport donne une vision globale du projet, sa problématique ainsi que son cadre général.

I.1 Présentation de l'organisme d'accueil

Commençons d'abord par présenter l'organisme d'accueil et l'ERP (Entreprise ressource planning) BYBLOS utilisé en particulier pour la gestion des candidats.

I.1.1 Présentation générale

Créée en 2007, **Talan Tunisie** est le centre de développement « Nearshore » du groupe Talan regroupant à ce jour plus de 200 ingénieurs de développement en nouvelles technologies, plus

particulièrement autour des technologies Java J2EE, Open Source, issus des plus grandes écoles d'ingénieurs tunisiennes et européennes, et travaillant pour les plus grands clients européens.

En huit ans, **Talan Tunisie** a acquis un savoir faire spécifique dans la mise en œuvre de projets au forfait en mode Nearshore et de centres délocalisés de services pour le compte de plusieurs clients grands comptes en France, mais aussi pour des éditeurs en forte croissance qui souhaitent bénéficier des avantages que peut offrir un centre de développement Nearshore en Tunisie (baisse des coûts de production, proximité géographique et culturelle, maîtrise du français, ressources humaines de qualité et en profusion, flexibilité), tout en en minimisant les risques.

Son offre a ainsi permis aux entreprises qui lui ont fait confiance d'accélérer la mise en opération de leurs projets, d'en réduire les coûts tout en les soulageant des contraintes du "sourcing" et de la gestion des ressources humaines.

La Figure 1.1 montre les implantations de Talan dans le monde.



FIGURE 1.1 : Implantation de Talan dans le monde

I.1.2 Secteurs d'activité

Talan Tunisie couvre essentiellement :

- Le secteur de la finance à travers une collaboration avec des banques d'investissement et des sociétés d'assurances.
- Le secteur de la télécommunication à travers des projets destinés aux opérateurs télécom et un ensemble de fournisseurs d'accès à internet.

- Le secteur du transport et de la logistique.
- Le secteur de l'énergie à travers des projets de développement visant des opérateurs de services liés à l'électricité, au gaz, à l'eau, etc.

I.2 Contexte du projet

Les entreprises utilisent un système "**Enterprise Resource Planning**" (ERP) en tant que progiciel permettant la gestion de l'ensemble des processus opérationnels d'une entreprise en intégrant plusieurs modules de gestion : module RH, module de gestion des fournisseurs, module de gestion de la comptabilité...

Autrement dit, l'ERP représente la « colonne vertébrale » d'une entreprise. **Talan Tunisie** a fait le choix d'investir dans la refonte de son système ERP, appelé **BYBLOS**, dont nous allons présenter la version existante. Nous exposerons aussi ses limites et l'impact de son architecture monolithique pour justifier la migration vers une architecture microservices.

I.2.1 Présentation de l'outil BYBLOS

BYBLOS est un ERP développé par les équipes de Talan Tunisie consulting, contenant plusieurs modules à savoir :

- Un module de gestion des ressources humaines : qui gère les fiches RH du personnel, ainsi que les soldes de congés.
- Un module de gestion des candidats : qui gère les fiches des candidats et leurs entretiens.
- Un module de gestion des collaborateurs : qui gère les fiches des collaborateurs.
- Un module de gestion des fournisseurs : qui gère les données personnelles et la situation des fournisseurs.
- Un module de gestion des contrats : qui expose le type de contrats et leurs détails (date d'effet, date de fin, durée ...).

Dans le cadre de ce projet, nous nous intéressons au module de **gestion des candidats**. Cette solution existante, basée sur une architecture monolithique, est développée avec Java Enterprise Edition (JEE) et Java Server Faces (JSF).

II Problématique

Dans cette section, nous allons présenter l'architecture monolithique, ainsi que son impact sur l'outil BYBLOS. L'architecture monolithique est l'architecture traditionnelle où toutes les tâches sont

réalisées dans une seule et grande application. Tous les services individuels accèdent à une grande base de données et sont édités via une interface utilisateur, tous implémentés dans une seule application.

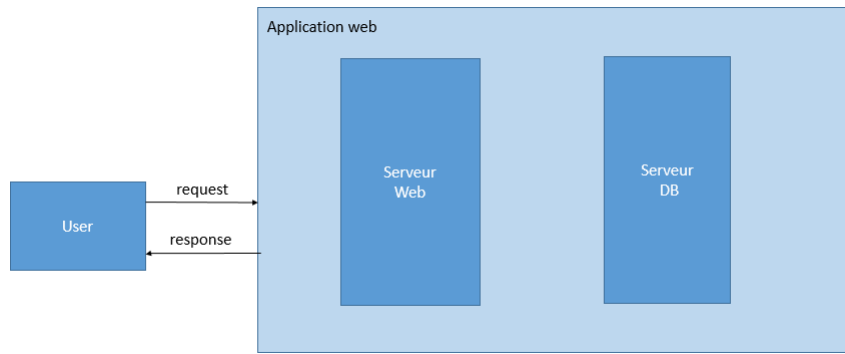


FIGURE 1.2 : Architecture monolithique

Une architecture monolithique représente le modèle traditionnel unifié de conception d'un programme informatique. En se basant sur une telle architecture, la maintenance de l'outil BYBLOS est devenue trop difficile à cause de la complexité de l'application qui augmente à chaque itération.

En outre, une légère modification apportée à une petite partie de l'application nécessite la construction et le déploiement d'une version entièrement nouvelle et requiert une durée des tests automatiques et des builds plus longue, ce qui engendre un coût supplémentaire. De plus, l'application est constituée d'un seul bloc, un dysfonctionnement sur une partie du système affecte toute l'application. Cette architecture réduit l'agilité de l'équipe et la fréquence de livraison à cause du fort couplage entre les composants de l'application.

L'outil BYBLOS est une application web composée de trois couches comme le montre la Figure 1.3 :

- **Couche présentation** : cette couche est composée des pages JSF et des servlets.
- **Couche métier** : cette couche représente l'ensemble métier de l'application. Elle est composée des EJBs.
- **Couche persistance** : c'est la couche Data Access Objects (DAO) qui transforme les objets métiers en données sérialisées, et inversement pour les échanges avec la base de donnée PostgreSQL. Elle est réalisée en se basant sur le framework Hibernate/JPA.

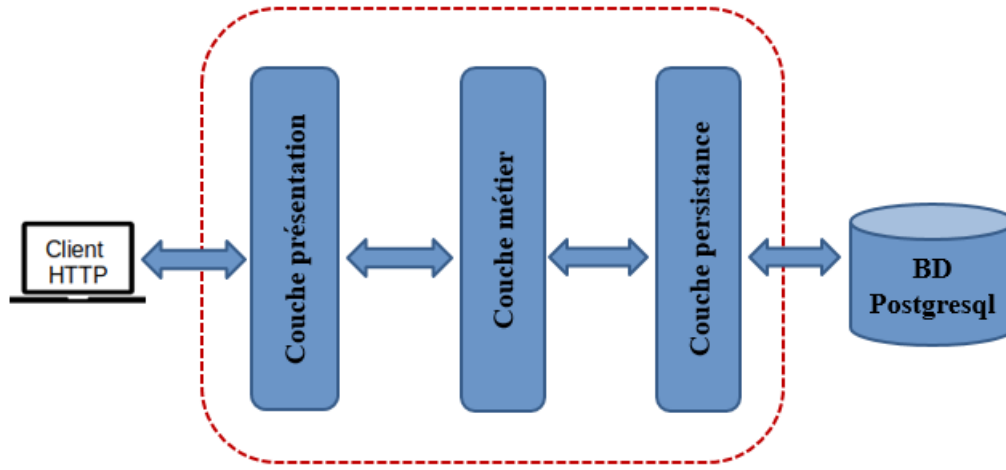


FIGURE 1.3 : Architecture monolithique de BYBLOS

III Solution proposée : Migration vers une architecture microservices

Talan prévoit de renforcer son effectif d'ici fin 2020 en intégrant 3000 nouveaux collaborateurs et d'ouvrir, dès septembre 2020, un siège international à Londres ainsi que des bureaux à l'étranger, notamment en Belgique, en Italie et aux Pays-Bas.

De ce fait, le module de gestion des candidats dans l'ERP BYBLOS doit être disponible et performant à cette période. Après une étude approfondie de la solution existante, et dans le but de surmonter les limites de l'architecture monolithique ainsi que dans un souci de performance, l'équipe BYBLOS a décidé de faire une refonte de l'ERP BYBLOS existant vers une architecture microservices tout en améliorant les IHMs.

C'est dans ce cadre que s'inscrit notre projet de fin d'études, qui consiste à réaliser la migration du module de gestion des candidats de la solution existante vers une architecture microservices.

Notre travail consiste donc à :

- Concevoir et implémenter le microservice de gestion des candidats ;
- Enregistrer le microservice dans le serveur de découverte (Eureka) et dans le Gateway (zuul) ;
- Ajouter le fichier de configuration du microservice dans le serveur de configuration.

IV Architecture microservices

Dans cette section, nous définirons l'architecture microservices. Par la suite, nous allons présenter les caractéristiques d'un microservice, et enfin les avantages de cette architecture par rapport à l'architecture monolithique comme le montre la figure 1.4.

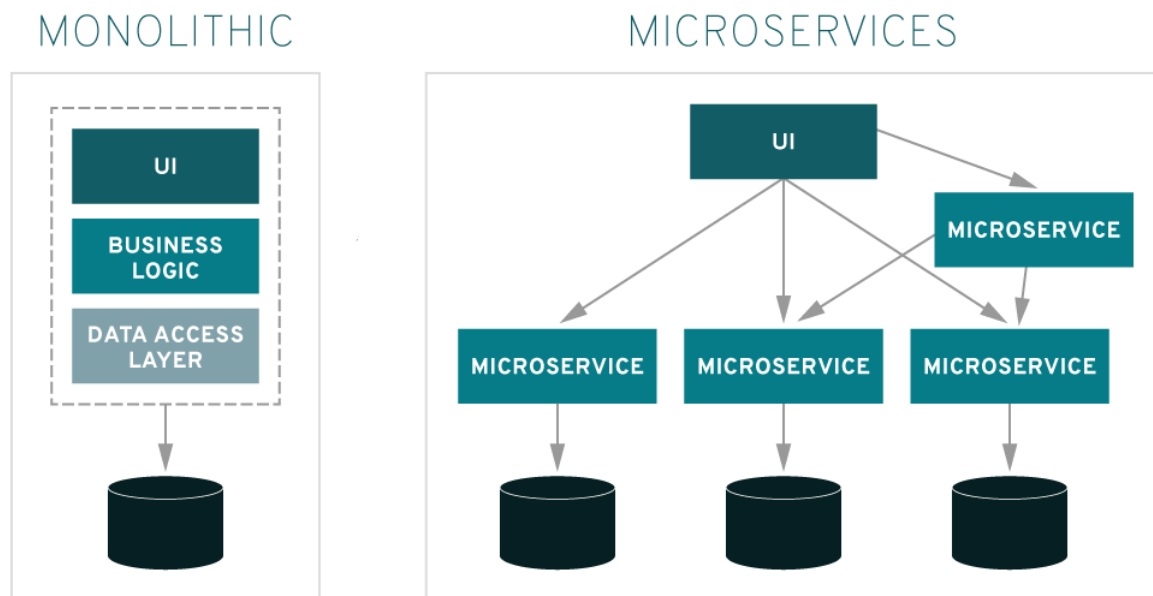


FIGURE 1.4 : Comparaison entre les architectures microservices et monolithique

L'architecture microservices est une méthode distinctive de développement des systèmes logiciels, qui a connu une popularité grandissante ces dernières années. Il s'agit d'une méthode de développement d'applications logicielles en tant que suite de services modulables et indépendants.

Ce style d'architecture présente plusieurs avantages comme l'hétérogénéité technologique, la résistance contre l'échec, la scalabilité sur mesure, la facilité de déploiement, l'alignement organisationnel et la réutilisabilité.

IV.1 Caractéristiques d'un microservice

L'architecture microservices possède plusieurs caractéristiques qu'il est essentiel d'appliquer durant la conception et le développement d'une application basée sur les microservices.

- 1 **Concentration sur une seule fonctionnalité :** Un microservice implémente une seule fonctionnalité dont il est responsable dans tout le système. Une fonctionnalité peut être une fonctionnalité métier contribuant à l'objectif du système, ou une fonctionnalité technique utilisée par plusieurs fonctionnalités métier.
- 2 **Déploiement individuel des microservices :** Chaque microservice doit être déployable indépendamment des autres. Ce point est important : tout changement ou mise à jour d'un microservice doivent être réalisés sans toucher aux autres microservices ni affecter leur exécution.
Le système doit continuer à fonctionner correctement durant la mise à jour du microservice en question, ainsi qu'après l'avoir redéployé.
- 3 **Indépendance des microservices :** Un microservice peut correspondre à un ou plusieurs processus. Il doit être obligatoirement hébergé sur des processus indépendants. En d'autres

termes, il ne doit pas partager un processus avec un autre microservice.

En revanche, il peut être constitué de plusieurs processus, notamment dans le cas d'un accès à une base de données. Autrement dit, seul ce service peut accéder à cette base de données. C'est une façon d'éviter au maximum les interdépendances entre les microservices.

Par exemple si on développe deux microservices sur le même processus, nous risquons d'avoir des interdépendances qui vont conduire à l'obligation de déployer les deux en même temps, ce qui est contradictoire avec les caractéristiques des microservices.

4 **Taille des microservices** : Comme l'indique le nom "micro", les microservices doivent être concentrés sur une seule et unique fonctionnalité, de façon à ce que leur développement et leur maintenance puisse se faire par des équipes de petites tailles.

5 **Remplaçabilité** : On doit pouvoir remplacer n'importe quel microservice du système dans un temps raisonnablement court.

6 **Conception pour l'échec** : L'un des atouts majeurs des microservices est qu'ils sont conçus pour être tolérants aux pannes. Dans une application en microservices, si un service échoue, les autres services ne sont pas affectés et adaptent leurs fonctionnements selon l'état du système dans lequel ils évoluent.

IV.2 Architecture monolithique Vs microservices

Le tableau 1.1 ci-dessous présente les principales différences entre les architectures microservice et monolithique.

	Architecture Microservice	Architecture Monolithique
Mise sur le marché	Déploiement et mises à jour en microservices sont plus rapides et plus agiles.	Déploiement et mises à jour en un seul bloc, nécessitent plus de temps et sont moins agiles.
Haute évolutivité	La possibilité d'étendre les déploiements sur plusieurs serveurs et infrastructures.	Le déploiement se fait sur un seul serveur.
Résilience	Les microservices sont indépendants. Lorsqu'un service tombe en panne, l'ensemble de l'application continue à fonctionner.	Lorsqu'un module tombe en panne, l'intégralité de l'application cesse de fonctionner.

Facilité de déploiement	Les applications basées sur des microservices sont plus modulaires et légères.	Les applications monolithiques sont déployées en un seul bloc et sont plus lourdes.
Ouverture	Les développeurs ont la liberté de choisir la technologie et le langage qui leur conviennent le mieux pour chaque fonction.	Les développeurs ne peuvent utiliser qu'une seule technologie et un seul langage pour toute l'application.

TABLEAU 1.1 : Architecture microservices Vs monolithique

V Méthodologie de développement adoptée : Scrum

Le choix de la méthodologie à utiliser durant le développement d'un logiciel doit répondre aux critères suivants :

- Exprimer au mieux les besoins des futurs clients.
- Permettre de développer une application robuste et évolutive.
- Élaborer une application qui répond aux besoins des clients dans des délais respectables.

Nous avons travaillé avec la méthode Scrum, déjà utilisée par l'équipe BYBLOS. Nous présentons, dans la suite, la méthode Scrum, les différents intervenants dans notre projet ainsi que son cycle de vie.

V.1 Présentation de la méthode SCRUM

Scrum est une méthode qui implémente les bonnes pratiques du Manifeste Agile dont l'utilisation s'étend au-delà du domaine informatique. Elle vise à produire les meilleurs logiciels possibles, le plus rapidement possible. Le cycle d'un projet Scrum est illustré dans la Figure 1.5.

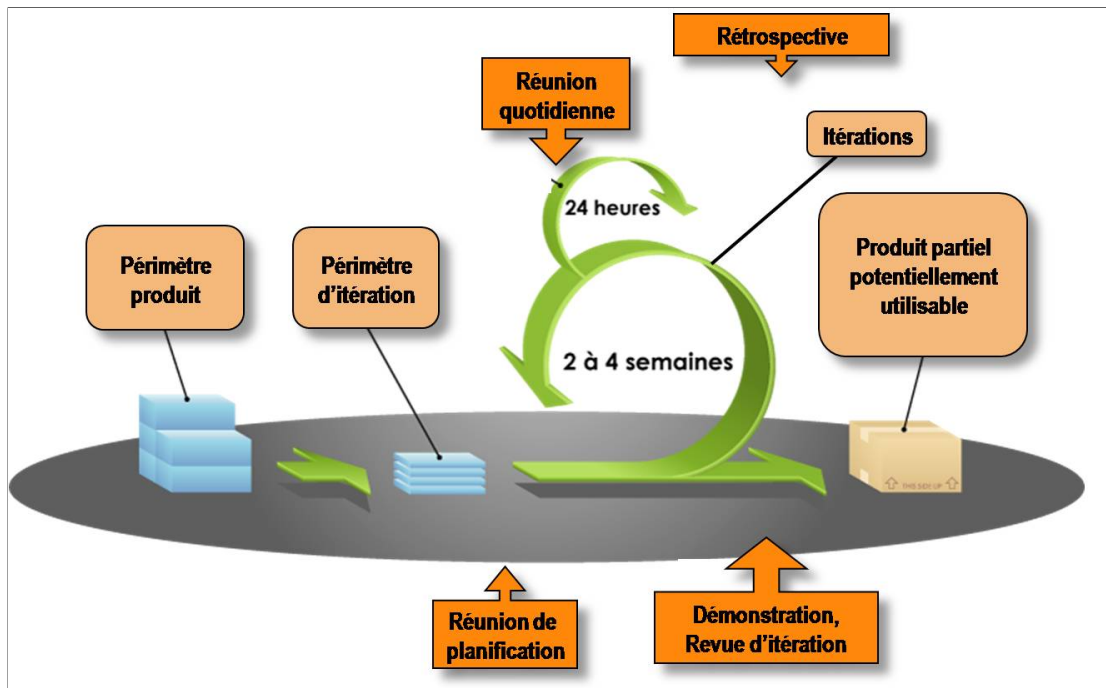


FIGURE 1.5 : Cycle de vie de SCRUM

V.2 Les rôles dans SCRUM

La méthodologie SCRUM fait intervenir trois rôles principaux qui sont :

- **Le Scrum Master** : C'est le coach de l'équipe, il assure plusieurs tâches :
 - Assister l'équipe pour appliquer la philosophie et les pratiques de Scrum.
 - S'assurer que l'équipe bénéficie des meilleures conditions pour accomplir les tâches nécessaires.
 - Surmonter les obstacles éventuels : prendre en compte les problèmes qui surviennent à tout moment sur le projet pour les éliminer.
 - Faire en sorte que l'équipe reste concentrée sur le véritable objectif du projet, en s'assurant que chacun participe pleinement aux travaux de l'équipe.
- **Le Product Owner** : Il est le propriétaire du **Product Backlog**. Il représente les utilisateurs et est fortement impliqué dans les réunions de planification et d'avancement. Il a aussi la capacité d'annuler le sprint avant échéance.
- **La Scrum Team** : Il s'agit de l'équipe de développement. Elle produit chaque itération du projet, et ses membres sont les mêmes tout au long du projet.

V.3 Cycle de vie de Scrum

- **Backlog produit** : lister toutes les fonctionnalités en les triant par priorité.
 - Besoins priorisés par le product owner.

- Besoins évalués par l'équipe.
- **Backlog de sprint** : extraire les fonctionnalités qui vont être développées au prochain sprint.
 - Extrait du backlog produit.
 - Besoins éclatés en tâches.
- **Sprint** : la scrum team commence à développer les fonctionnalités du Sprint
 - Développement des fonctionnalités du backlog de sprint.
 - Aucune modification du backlog de sprint possible.
- **Mêlée quotidienne** : une petite réunion journalière pour identifier les problèmes rencontrés par chaque membre de l'équipe de développement.
 - Point de contrôle quotidien de l'équipe.
 - Interventions régulées : 2 minutes par personne.
- **Incrément logiciel** : Après avoir achevé le sprint et testé les fonctionnalités, un livrable est prêt à être déployé.
 - Livrable remis au Product Owner à la fin du sprint.

Conclusion

Dans ce chapitre, nous avons défini le cadre général de notre projet, en présentant l'organisme d'accueil et la solution proposée. Nous avons aussi défini les principales notions qui constituent la base de ce travail. Le prochain chapitre sera consacré à la spécification des besoins fonctionnels et non-fonctionnels de notre système.

ANALYSE ET SPÉCIFICATION DES BESOINS

Plan

I	Identification des acteurs	12
II	Identification des besoins	12
III	Diagramme de cas d'utilisation global	13
IV	Diagrammes des cas d'utilisation détaillés	14

Introduction

Dans toute application informatique, les fonctionnalités doivent être mises en relation avec un ensemble de besoins définis par l'utilisateur du système. Les fonctions que le système est appelé à accomplir découlent essentiellement des besoins exprimés. La spécification des besoins constitue l'élément de base sur lequel sera construite l'architecture du système. Nous commencerons par identifier les acteurs du système. Nous définirons ensuite les besoins exprimés par ces derniers, et à partir desquels seront définis les cas d'utilisations de l'application.

I Identification des acteurs

Un acteur est l'idéalisation d'un rôle joué par une entité distincte du système et qui interagit directement avec lui. Notre module comporte deux acteurs majeurs :

- **Le Back-office** : cet acteur est responsable de la gestion de la fiche du candidat et de l'assignation d'un entretien à un recruteur.
- **Le recruteur** : c'est celui qui va passer l'entretien d'embauche au candidat.

II Identification des besoins

L'application offre à chaque acteur un ensemble de fonctionnalités. Cette partie sera donc consacrée aux besoins fonctionnels et non fonctionnels relatifs à l'application.

II.1 Les besoins fonctionnels

Dans cette partie, nous citons les fonctionnalités pour chaque acteur.

- Pour le back-office, l'application doit lui permettre de :
 - Gérer une fiche candidat : le back-office peut ajouter ou modifier une fiche candidat.
 - Gérer un entretien : le back-office peut planifier un entretien, modifier certains de ses éléments ou consulter son Workflow(historique).
 - Rechercher un candidat : le back-office peut rechercher un candidat selon plusieurs critères.
 - Consulter le KPI des avis des entretiens : le back-office peut consulter le graphe(en secteurs) qui indique le nombre total,par année, de candidats correspondant à chacun des avis listés.
- Pour le recruteur , l'application doit lui permettre de :
 - Modifier un entretien : le recruteur peut modifier certains éléments de l'entretien.
 - Rechercher un candidat : le recruteur peut rechercher un candidat selon plusieurs critères.
 - Valider un entretien : le recruteur peut accepter ou rejeter un entretien selon sa disponibilité.
 - Consulter le KPI des avis des entretiens : le recruteur peut consulter le graphe(en secteurs) qui indique le nombre total,par année, de candidats correspondant à chacun des avis listés.

II.2 Les besoins non fonctionnels

Dans l'intention de réussir notre tâche, l'application doit vérifier quelques propriétés et doit tenir compte de certaines contraintes et exigences :

- **Ergonomie** : les interfaces graphiques de notre module de gestion des candidats doit avoir une ergonomie identique à celle des modules déjà migrés vers l'architecture microservices.
- **Sécurité** : notre application doit assurer la sécurité pour chaque authentification.
- **Maintenabilité** : notre module de gestion des candidats doit être évolutif et extensible.

III Diagramme de cas d'utilisation global

Cette partie permet de montrer le comportement de l'application. Nous utilisons le diagramme de cas d'utilisation pour visualiser les fonctionnalités du système et les interactions entre les acteurs et le système.

La Figure 2.1 présente le diagramme des cas d'utilisation global de notre application montrant les fonctionnalités offertes pour chacun des deux principaux acteurs(back-office et recruteur).

Nous présentons dans la suite le raffinement des cas d'utilisation pour chaque acteur.

Dans cette section nous présentons, pour chaque acteur, les diagrammes des cas d'utilisation détaillés et leur description textuelle.

Dans cette partie, nous développons chaque cas d'utilisation du back-office avec une description détaillée.

Nous nous intéressons, dans cette partie, au raffinement du cas d'utilisation "Gérer une fiche candidat". Les fonctionnalités relatives à ce diagramme sont présentées dans la Figure 2.2.

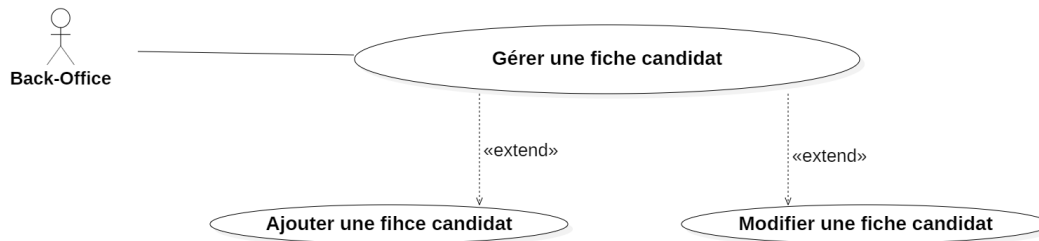


FIGURE 2.2 : Diagramme de cas d'utilisation "Gérer une fiche candidat"

Nous détaillons dans ce qui suit les deux cas d'utilisation : "Ajouter une fiche candidat" et "Modifier une fiche candidat". Après authentification, le back-office peut toujours créer une nouvelle fiche candidat.

Titre	Ajouter une fiche candidat
Résumé	Le back-office peut ajouter une nouvelle fiche candidat.
Pré-condition	Le back-office doit être authentifié.
Scénario normal	<ul style="list-style-type: none"> — A1 : Le back office clique sur le bouton "Ajouter fiche candidat". — A2 : Le système affiche le formulaire à remplir. — A3 : Le back-office introduit les informations nécessaires. — A4 : Le système vérifie les informations saisies et ajoute le candidat.
Enchaînement d'erreur	E1 : Le système affiche un message d'erreur mentionnant la présence d'un champ obligatoire vide.
Post-condition	Une nouvelle fiche candidat est ajoutée.

TABLEAU 2.1 : Description du cas d'utilisation "Ajouter une fiche candidat"

Le back-office peut aussi modifier une fiche candidat.

Titre	Modifier une fiche candidat
Résumé	Le back-office peut modifier une fiche candidat déjà créée.
Pré-condition	Le back-office doit être authentifié.
Scénario normal	<ul style="list-style-type: none"> — A1 : Une fois la recherche est effectuée, le back office clique sur le bouton "Consulter ou Modifier une fiche candidat" pour le candidat choisis. — A2 : Le système affiche le formulaire contenant les informations du candidat concerné. — A3 : Le back-office modifie les informations nécessaires. — A4 : Le système vérifie les informations saisies et enregistre les modifications.
Enchaînement d'erreur	E1 : Le système affiche un message d'erreur mentionnant la présence d'un champ obligatoire vide.
Post-condition	Une fiche candidat est modifiée.

TABLEAU 2.2 : Description du cas d'utilisation "Modifier une fiche candidat"

IV.1.2 Cas d'utilisation "Gérer un entretien"

Dans cette partie, nous nous concentrons sur le raffinement du cas d'utilisation "Gérer un entretien". Les fonctionnalités relatives à ce diagramme sont représentées dans la Figure 2.3.

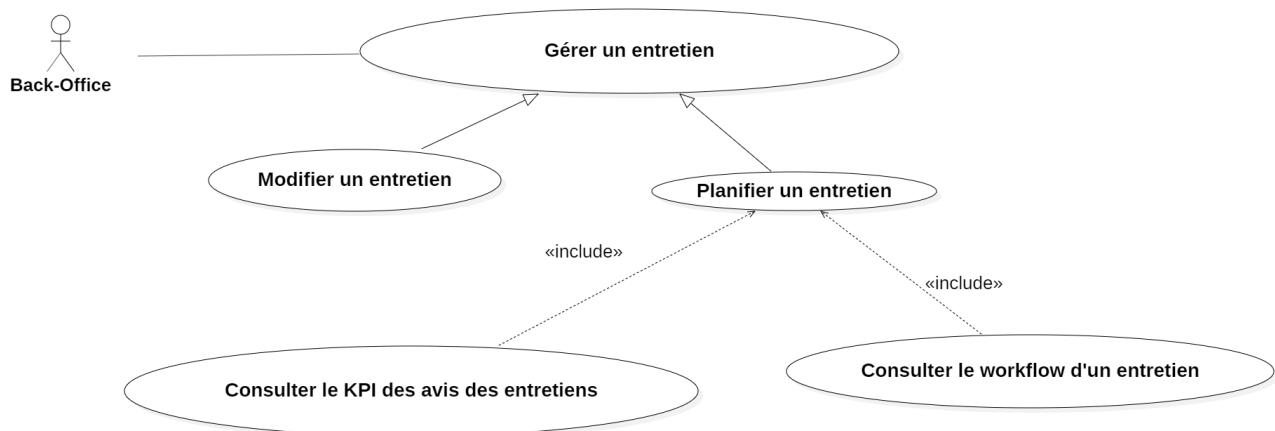


FIGURE 2.3 : Diagramme de cas d'utilisation "Gérer un entretien"

Nous détaillons dans ce qui suit les quatres cas d'utilisation : "Planifier un entretien" , "Modifier

un entretien", "Consulter le KPI des avis des entretiens" et "Consulter le workflow d'un entretien".

Le back-office peut toujours planifier un entretien après avoir été authentifié.

Titre	Planifier un entretien
Résumé	Le back-office peut planifier un nouvel entretien.
Pré-condition	Le back-office doit être authentifié.
Scénario normal	<ul style="list-style-type: none">— A1 : Une fois la fiche du candidat est créée, le back-office clique sur le bouton "Planifier un entretien".— A2 : Le système affiche le formulaire de l'entretien.— A3 : Le back-office introduit les informations nécessaires de l'entretien.— A4 : Le back-office clique sur le bouton "Enregistrer".— A5 : Le système vérifie les informations saisies et ajoute l'entretien.— A6 : Le système attribue le statut "En attente" pour cet entretien.— A7 : Le système envoie un e-mail, généré automatiquement, au recruteur sélectionné, contenant la date, l'heure, les nom et prénom du candidat et un lien qui mène vers la fiche de ce candidat.
Enchaînement d'erreur	E1 : Le système affiche un message d'erreur mentionnant la présence d'un champ obligatoire vide.
Post-condition	Un nouvel entretien est ajouté.

TABLEAU 2.3 : Description du cas d'utilisation "Planifier un entretien"

Le back-office peut aussi modifier un entretien.

Titre	Modifier un entretien
Résumé	Le back-office peut modifier un entretien déjà planifié.
Pré-condition	Le back-office doit être authentifié.

Scénario normal	<ul style="list-style-type: none"> — A1 : Une fois un entretien est créé, le back-office clique sur le bouton "Modifier un entretien". — A2 : Le système affiche le formulaire de l'entretien déjà créé. — A3 : Le back-office introduit les modifications d'informations nécessaires. — A4 : Le back-office clique sur le bouton "Enregistrer". — A5 : Le système vérifie les informations saisies et enregistre les modifications.
Enchaînement d'erreur	E1 : Le système affiche un message d'erreur mentionnant la présence d'un champ obligatoire vide.
Post-condition	Un entretien est modifié.

TABLEAU 2.4 : Description du cas d'utilisation "Modifier un entretien"

Le back-office peut aussi consulter le KPI des avis des recruteurs lors des entretiens, sachant que ces avis changent d'une implantation à une autre de Talan après qu'un entretien ait été planifié.

Titre	Consulter le KPI des avis des recruteurs lors des entretiens
Résumé	Le back-office peut consulter le KPI des avis des entretiens.
Pré-condition	Le back-office doit être authentifié et l'entretien déjà créé.
Scénario normal	<ul style="list-style-type: none"> — A1 : Le back-office navigue vers l'interface d'accueil(Dashboard). — A2 : Le système affiche l'interface contenant un graphe à secteurs montrant le nombre total de candidats par avis(qui concerne l'implantation de Talan du back-office connecté) pour une année sélectionnée au préalable.
Enchaînement d'erreur	
Post-condition	Le KPI des avis des entretiens est consulté.

TABLEAU 2.5 : Description du cas d'utilisation "Consulter le KPI des avis des entretiens"

Le back-office peut aussi consulter le Workflow d'un entretien qui représente son historique :

- **Au niveau du back-office** : la date de création de l'entretien, les nom et prénom de l'intervenant au back-office qui a créé l'entretien.
- **Au niveau du recruteur** : après l'acceptation ou le rejet de l'entretien par ce dernier : la date de la décision du recruteur, les nom et prénom de celui-ci.

Titre	Consulter le Workflow d'un entretien
Résumé	Le back-office peut consulter le workflow d'un entretien déjà créé.
Pré-condition	Le back-office doit être authentifié et un entretien déjà créé.
Scénario normal	<ul style="list-style-type: none">— A1 : Une fois un entretien est créé, le back-office clique sur le bouton "Consulter Workflow"— A2 : Le système affiche un modal (Popup : une petite fenêtre) contenant la date de création , les nom et prénom de l'intervenant au back-office qui a créé l'entretien, ainsi que la date de l'acceptation ou du rejet de l'entretien avec le nom et le prénom du recruteur concerné.
Enchaînement d'erreur	
Post-condition	La fenêtre du Workflow est affichée.

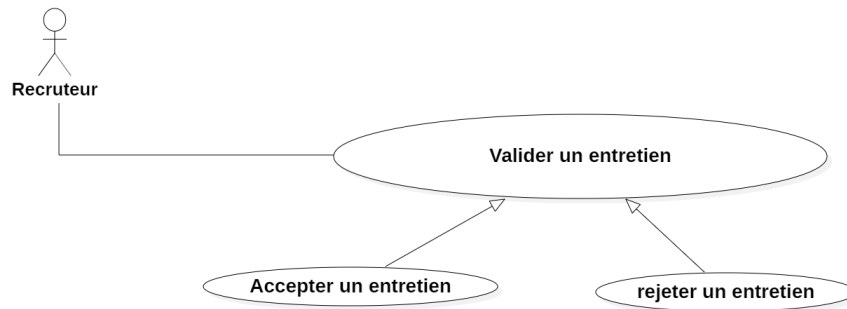
TABLEAU 2.6 : Description du cas d'utilisation "Consulter le workflow d'un entretien"

IV.2 Raffinement des cas d'utilisation pour le recruteur

Nous développons, dans cette partie, chaque cas d'utilisation du recruteur avec une description détaillée.

IV.2.1 Cas d'utilisation "Valider un entretien"

Nous nous intéressons, dans cette partie, au raffinement du cas d'utilisation "Valider un entretien". Les fonctionnalités relatives à ce diagramme sont représentées dans la Figure 2.4

**FIGURE 2.4 :** Diagramme de cas d'utilisation "Valider un entretien"

Nous détaillons dans ce qui suit les deux cas d'utilisation : "Accepter un entretien" et "Rejeter un entretien". Après la planification d'un entretien et l'authentification du recruteur, un e-mail généré automatiquement sera envoyé à ce dernier pour lui notifier qu'un entretien est prévu à une date et à une heure précises, et lui transmettre un lien vers la fiche du candidat sur laquelle il indique sa décision d'accepter ou de rejeter l'entretien.

Titre	Accepter un entretien
Résumé	Le recruteur peut accepter un entretien déjà créé.
Pré-condition	Le recruteur doit être authentifié et un entretien déjà créé.
Scénario normal	<ul style="list-style-type: none"> — A1 : Une fois un entretien est créé et l'e-mail est envoyé, le recruteur clique sur le lien dans l'e-mail et l'interface de la fiche du candidat sera chargée. — A2 : le recruteur clique sur le bouton "Accepter un entretien". — A3 : Le système modifie le statut de l'entretien de "En attente" en "Accepté".
Post-condition	Le statut de l'entretien est modifié en "Accepté"

TABLEAU 2.7 : Description du cas d'utilisation "Accepter un entretien"

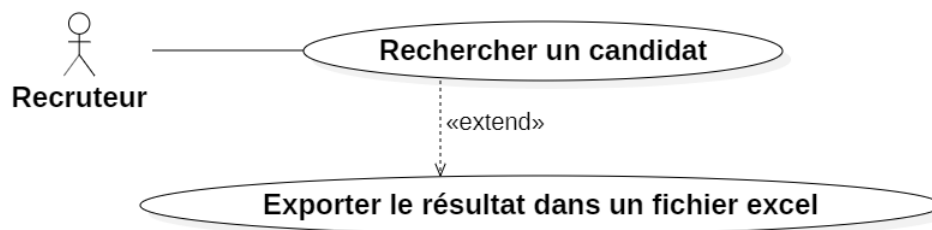
Le recruteur peut aussi rejeter un entretien.

Titre	Rejeter un entretien
Résumé	Le recruteur peut rejeter un entretien déjà créé.
Pré-condition	Le recruteur doit être authentifié et un entretien déjà créé.
Scénario normal	<ul style="list-style-type: none"> — A1 : Une fois un entretien est créé et l'e-mail est envoyé, le recruteur clique sur le lien dans l'e-mail et l'interface de la fiche du candidat sera chargée. — A2 : le recruteur clique sur le bouton "Rejeter un entretien". — A3 : Le système modifie le statut de l'entretien de "En attente" en "Rejeté".
Post-condition	Un e-mail est envoyé au back-office pour le notifier.

TABLEAU 2.8 : Description du cas d'utilisation "Rejeter un entretien"

IV.2.2 Cas d'utilisation "Rechercher un candidat"

Dans cette partie, nous nous intéressons au raffinement du cas d'utilisation "Rechercher un candidat". Les fonctionnalités relatives à ce diagramme sont représentées dans la figure 2.5.

**FIGURE 2.5** : Diagramme de cas d'utilisation "Rechercher un candidat"

Nous détaillons dans ce qui suit le cas d'utilisation "Exporter le résultat dans un fichier excel". Après que le recruteur ait effectué une recherche des candidats, le résultat est affiché sous forme d'un tableau. Le recruteur peut sélectionner un ou plusieurs candidats dans le tableau, et clique sur le bouton "Exporter" pour télécharger un fichier excel qui contient toutes les informations concernant les candidats sélectionnés.

Titre	Exporter le résultat dans un fichier excel
Résumé	Le recruteur peut exporter le résultat de la recherche de candidats dans un fichier excel.
Pré-condition	Le recruteur doit être authentifié et une recherche doit être effectuée.
Scénario normal	<ul style="list-style-type: none">— A1 : Une fois la recherche est effectuée, le recruteur sélectionne le(s) candidat(s) en cliquant sur la case à cocher pour chaque candidat.— A2 : le recruteur clique sur le bouton "Exporter".— A3 : Le système génère un fichier excel contenant toutes les informations relatives aux candidats sélectionnés, et le télécharge à travers le navigateur .
Post-condition	Un fichier excel est téléchargé.

TABLEAU 2.9 : Description du cas d'utilisation "Exporter le résultat dans un fichier excel"

Conclusion

Dans ce chapitre, nous avons mis en évidence les acteurs, ainsi que les besoins fonctionnels et non fonctionnels du système. D'autre part, nous avons détaillé les différents cas d'utilisation en précisant une description textuelle pour chaque cas. En effet, ce chapitre nous a permis de mieux comprendre les fonctionnalités à développer. L'étape suivante concerne la conception de notre application qui sera présenté dans le prochain chapitre.

CONCEPTION

Plan

I	Conception générale	23
II	Conception détaillée	26

Introduction

Ce chapitre est consacré, en premier lieu, à visualiser la conception générale de l'application en particulier son architecture logique. En deuxième lieu, nous présenterons l'enchaînement de conception qui a pour but de mieux expliciter les fonctionnalités en se basant sur les diagrammes de séquences. L'objectif recherché étant de façonner le système et de lui donner une forme répondant à tous les besoins et exigences recensés.

I Conception générale

Dans cette section, nous présentons, en premier lieu, les interactions des composants de notre application dans un diagramme de déploiement. En deuxième lieu, nous exposons l'architecture logique ainsi que le diagramme de paquetage.

I.1 Diagramme de déploiement de l'application

La Figure 3.1 décrit l'infrastructure selon laquelle notre application est déployée. Cette infrastructure est constituée des éléments suivants :

- **Serveur web** : il contient un seul composant "Angular Frontend" qui est la partie frontend de notre application.
- **Serveur** : il contient plusieurs composants :
 - **Gateway** : c'est le composant qui est responsable de l'acheminement des requêtes reçues du serveur web vers les différents microservices, dont le module "Gestion Candidat".

- **Eureka discovery service** : c'est un service de découverte des microservices qui permet de signaler un microservice défectueux.
- **Configuration** : c'est le composant qui contient toute la configuration des différents microservices.
- **Gestion candidat** : c'est le microservice objet de ce travail qui est composé principalement de trois parties :
 - **Gestion candidat Controller** : qui contient les différents web services REST.
 - **Gestion candidat Service** : qui contient le métier de notre module.
 - **Gestion candidat DAO** : qui contient la partie de persistance des données dans la base de données.
- **Serveur Base de données** : il contient un seul composant, qui est le Système de gestion de base de données Postgresql.

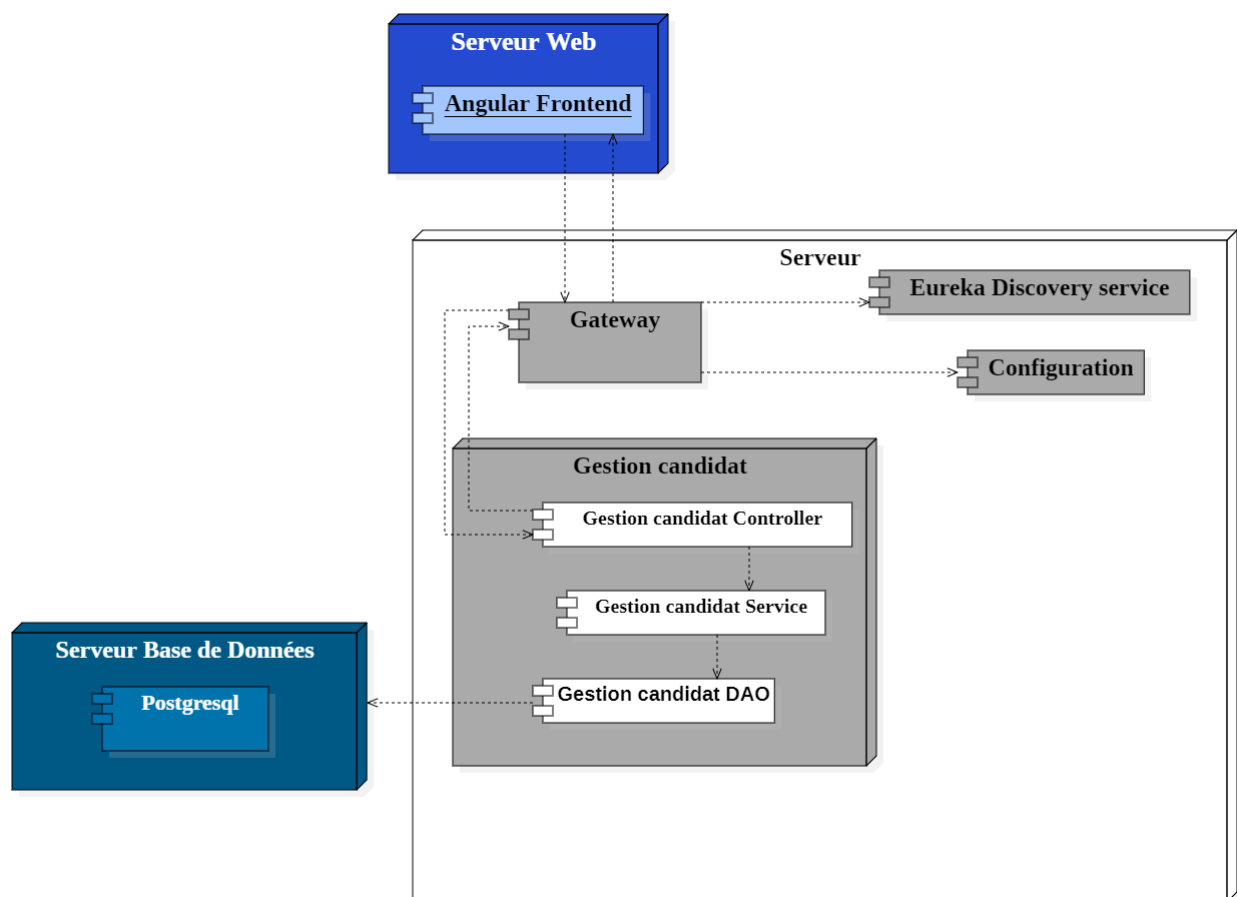


FIGURE 3.1 : Diagramme de déploiement de l'application

I.2 Architecture logique de l'application

L'architecture logique décrit les différentes couches d'une application, leurs interrelations et leurs interactions. Notre module de gestion des candidats est composé des couches suivantes présentées dans la Figure 3.2.

- **Couche présentation** : cette couche contient les interfaces graphiques qui permettent la visualisation des données pour l'utilisateur.
- **Couche contrôleur** : cette couche contient les différents web services REST qui permettent de relier la couche présentation avec la partie métier de notre application.
- **Couche Service** : cette couche contient les différents services métier de l'application.
- **Couche Mapper** : cette couche permet la transformation des objets sous forme DTO (provenant de la couche présentation) en des entités.
- **Couche persistance** : cette couche permet l'interaction avec la base de données pour ajouter, modifier, supprimer ou bien consulter des données.

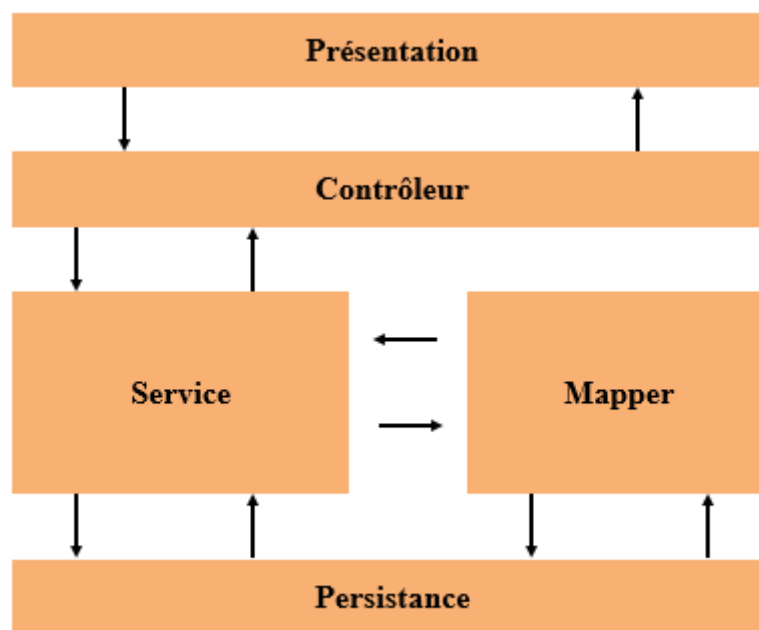


FIGURE 3.2 : Architecture logique du module de gestion des candidats

I.3 Diagramme de paquetage

Notre application doit garantir une indépendance dans les rôles entre ses différents modules qui sont représentés dans la Figure 3.3 :

- Le paquetage **Controller** contient les web services REST.
- Le paquetage **Dto** contient les différentes classes Dto qui représentent les objets envoyés et reçus de la partie frontend.
- Le paquetage **Services** contient les différentes classes de la logique métier.
- Le paquetage **Mapper** contient les différentes classes qui permettent la transformation des objets Dto en des entités pour les enregistrer, les modifier ou les supprimer de la base de données.

- Le paquetage **Dao** contient les différentes interfaces contenant les méthodes qui permettent la persistance des données dans la base de données.
- Le paquetage **Entities** contient les différentes classes des entités relatives aux tables de la base de données.

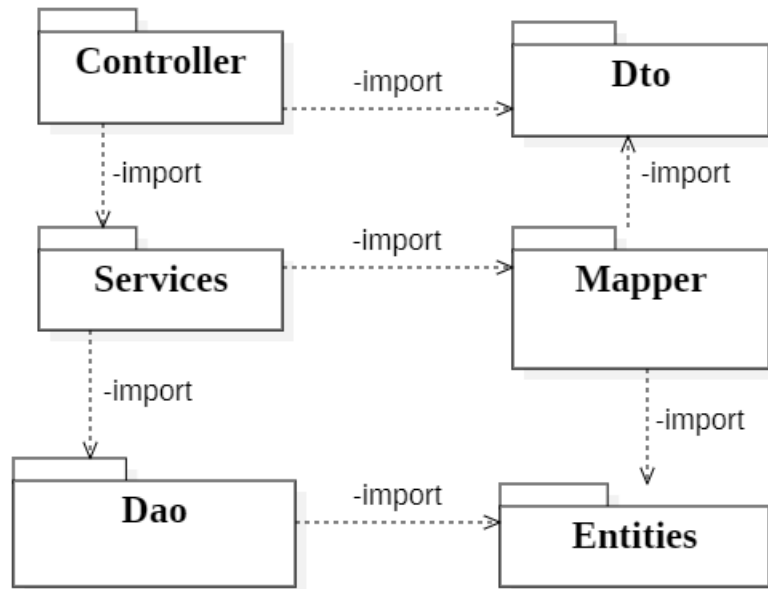


FIGURE 3.3 : Diagramme de paquetage

II Conception détaillée

Après avoir précisé les différents aspects de la conception générale de notre application, nous passons à la conception détaillée. Dans cette section, nous présentons le diagramme de classe ainsi que les diagrammes de séquences.

II.1 Diagramme de classes

Les diagrammes de classes sont l'un des types de diagrammes les plus utiles, car ils décrivent clairement la structure d'un système particulier en modélisant ses classes, ses attributs, ses opérations et les relations entre ses objets. La Figure 3.4 représente le diagramme de classes relatif à notre application.

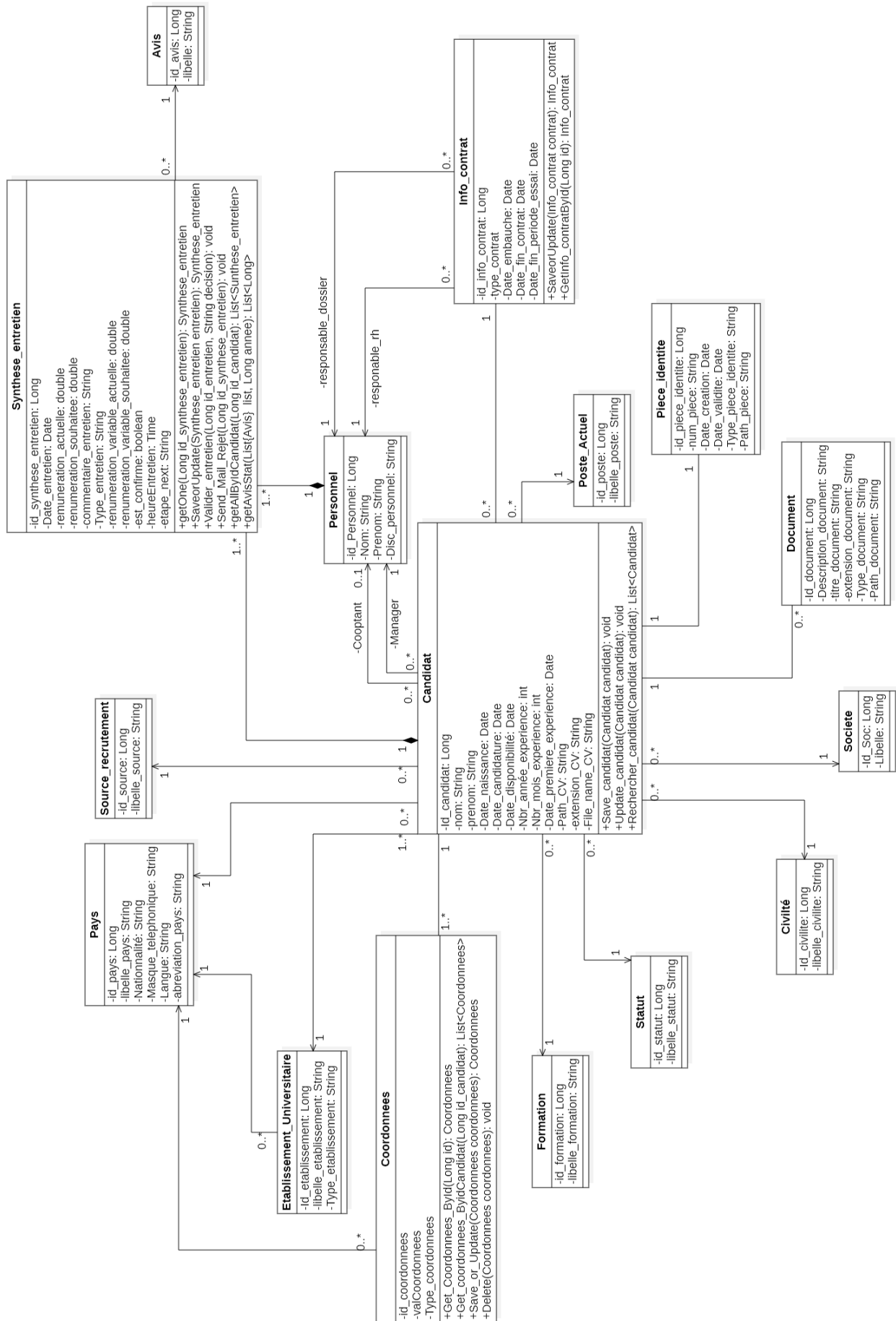


FIGURE 3.4 : Diagramme de classes

Le diagramme représenté par la Figure 3.4 est constitué des classes suivantes :

- **Classe coordonnées** : cette classe permet de gérer les coordonnées relatives à un candidat (un e-mail ou bien un numéro de téléphone).
- **Classe Info_contrat** : cette classe permet de gérer les informations concernant le contrat (salaire, durée, poste, régime...) à signer par candidat s'il sera accepté.
- **Classe Synthèse_entretien** : cette classe permet de gérer les entretiens du candidat.
- **Classe Candidat** : cette classe permet de gérer la fiche du candidat.

II.2 Diagrammes de séquences

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique. Dans cette partie, nous présentons les principaux diagrammes de séquences relatifs aux fonctionnalités de notre application.

II.2.1 Ajouter une fiche candidat

La Figure 3.5 représente le diagramme de séquence relatif à l'ajout d'une nouvelle fiche candidat.

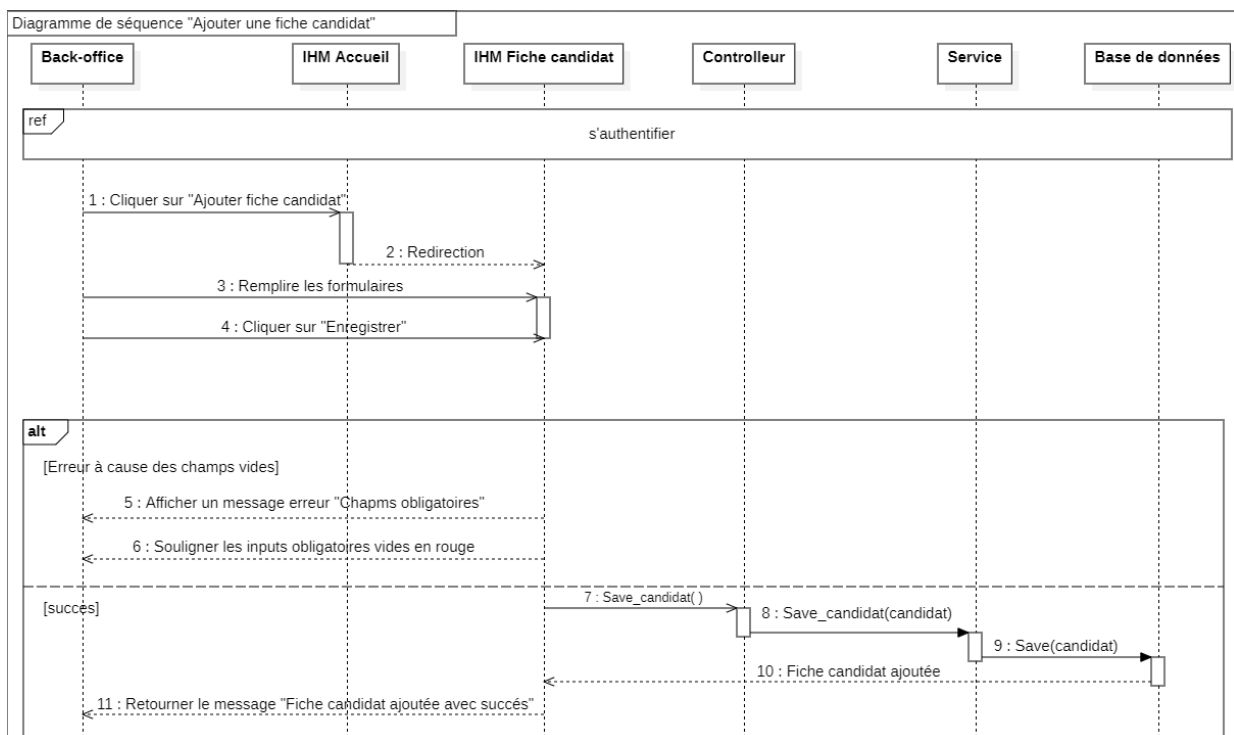


FIGURE 3.5 : Diagramme de séquence "Ajouter une fiche candidat"

Le back-office doit s'authentifier pour accéder à l'interface de la fiche candidat en cliquant sur "Ajouter fiche candidat" à partir de l'interface d'accueil. Il doit ensuite remplir le formulaire de la fiche et cliquer sur "Enregistrer". Un objet candidat est envoyé au contrôleur qui le passe, à son tour, au service pour l'enregistrer dans la base de données.

II.2.2 Rechercher un candidat

Le diagramme de séquence du cas d'utilisation "Rechercher un candidat" est représenté par la Figure 3.6

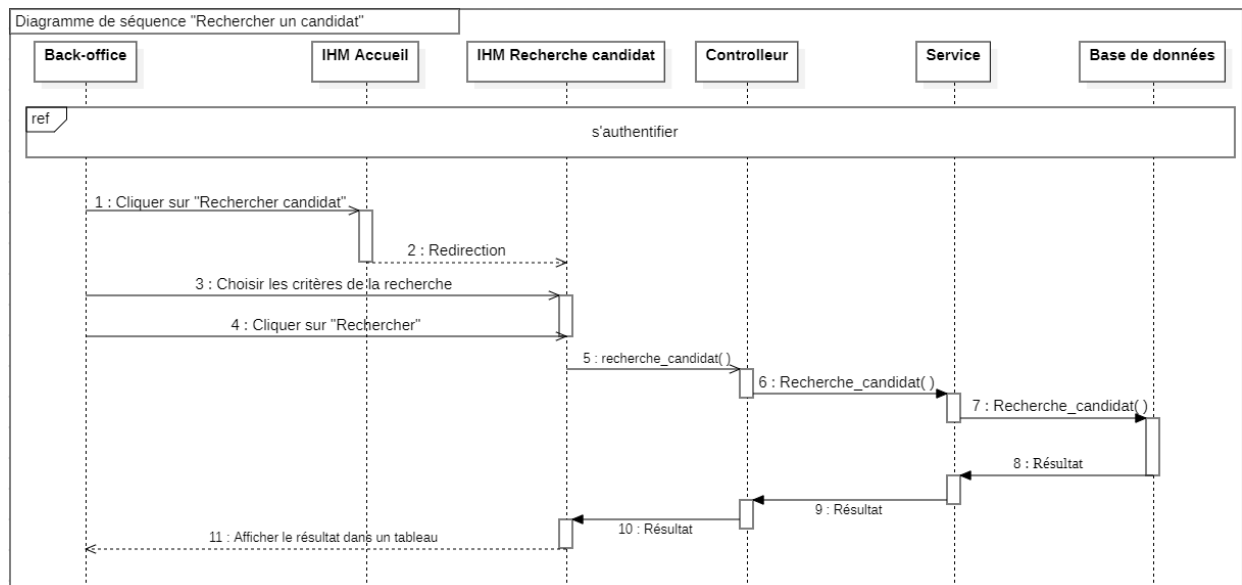


FIGURE 3.6 : Diagramme de séquence "Rechercher un candidat"

Après authentification, le back-office clique sur le bouton "Rechercher un candidat". Une fois l'interface de la recherche est chargée, le back-office choisit les critères de recherche (liste déroulante, case à cocher...) et clique sur le bouton "Rechercher". Un objet contenant les critères de recherche est envoyé au contrôleur, puis passé au service afin d'exécuter la méthode de la recherche. Le résultat est envoyé sous forme d'une liste des candidats à l'interface de la recherche où le résultat est affiché sous forme d'un tableau. Chaque ligne du tableau contient quelques informations qui concernent le candidat ainsi qu'un bouton pour consulter la fiche du candidat.

II.2.3 Planifier un entretien

La Figure 3.7 illustre le diagramme de séquence relatif à la planification d'un nouvel entretien.

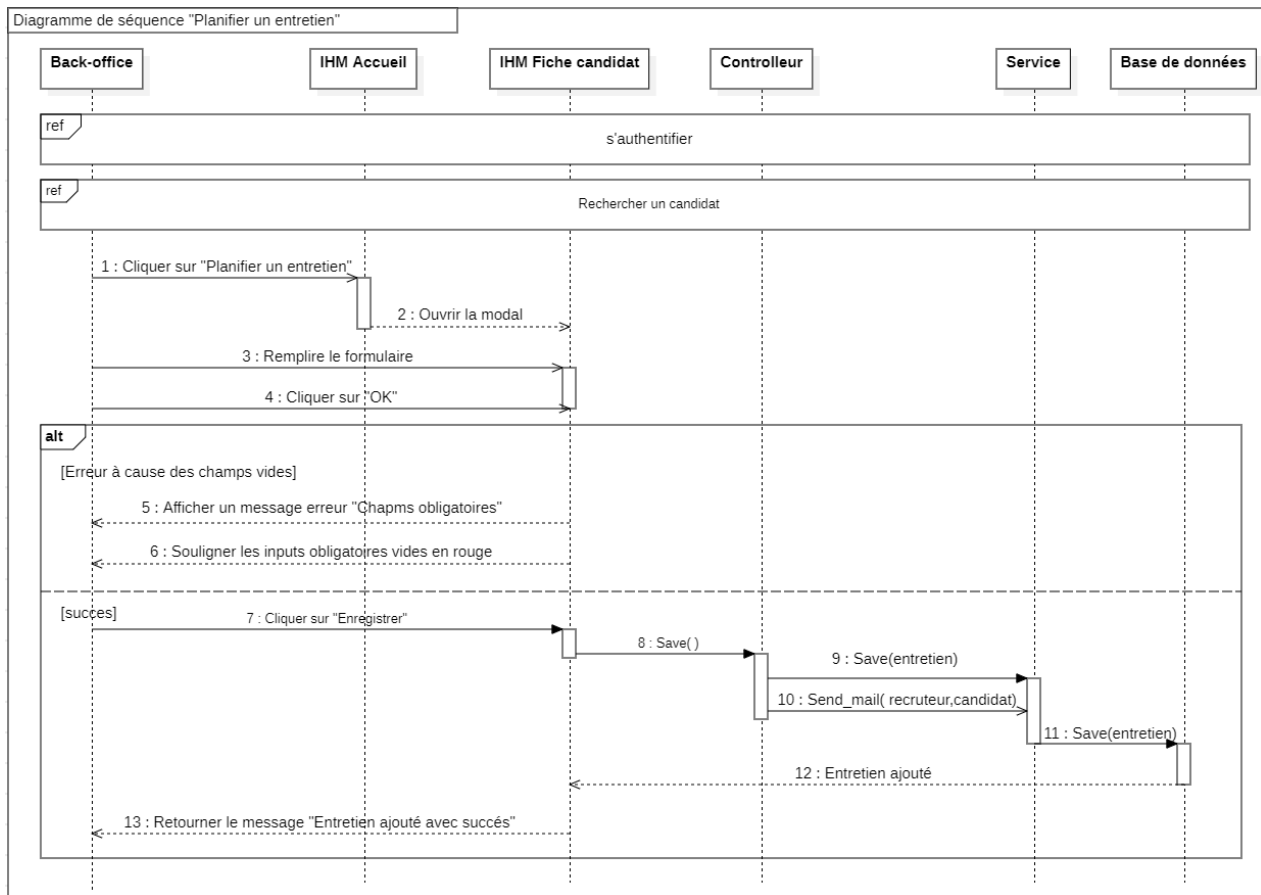


FIGURE 3.7 : Diagramme de séquence "Planifier un entretien"

Une fois le back-office s'est authentifié et a effectué une recherche, il planifie un nouvel entretien en cliquant sur le bouton "Planifier un entretien". Après, il remplit le formulaire et clique sur "OK". Le système vérifie qu'il n'y a pas des champs obligatoires vides et enregistre l'entretien. Enfin, un e-mail est envoyé pour notifier le recruteur sélectionné lors de la planification de l'entretien.

II.2.4 Accepter un entretien

Le diagramme de séquence relatif au cas d'utilisation "Accepter un entretien" est représenté dans la Figure 3.8.

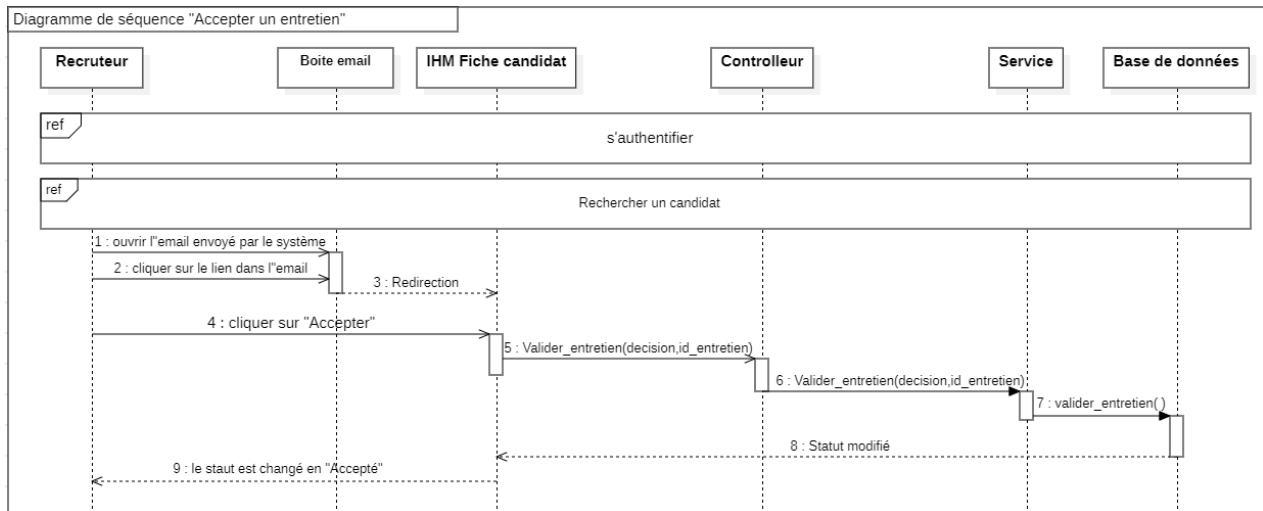


FIGURE 3.8 : Diagramme de séquence "Accepter un entretien"

Une fois un entretien est planifié, le recruteur consulte l'e-mail reçu dans son courrier électronique et contenant les informations relatives à l'entretien. Il clique alors sur le lien qui lui est envoyé dans l'e-mail et qui le redirige vers la fiche du candidat. Il clique sur le bouton "Accepter", ce qui modifie le statut de l'entretien en "Accepté".

II.2.5 Rejeter un entretien

La Figure 3.9 représente le diagramme de séquence relatif au cas d'utilisation "Rejeter un entretien".

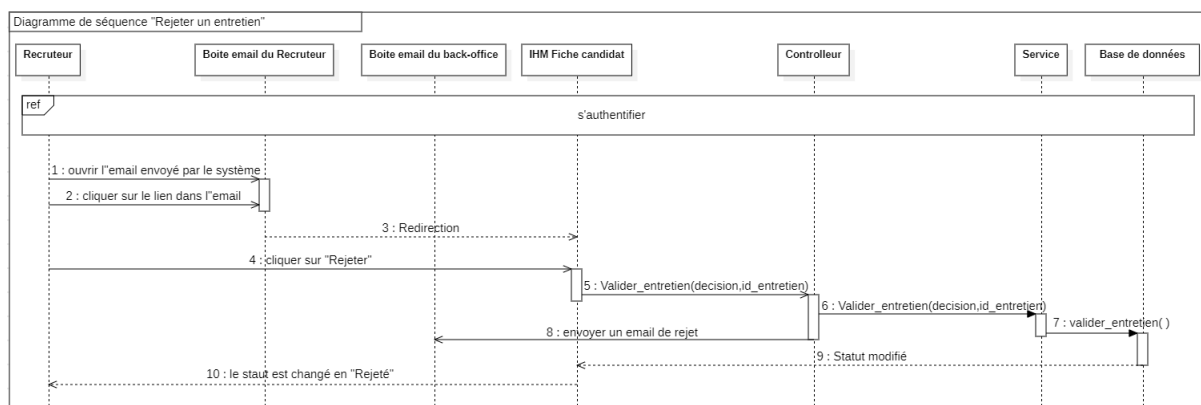


FIGURE 3.9 : Diagramme de séquence "Rejeter un entretien"

Une fois un entretien est planifié, le recruteur consulte l'e-mail reçu dans son courrier électronique et contenant les informations relatives à l'entretien. Il clique alors sur le lien qui lui est envoyé dans l'e-mail et qui le redirige vers la fiche du candidat. Il clique sur le bouton "Rejeter", un e-mail est alors envoyé au back-office pour le notifier du rejet de l'entretien par le recruteur. Ainsi, le statut de l'entretien est modifié en "Rejeté".

II.2.6 Modifier une fiche candidat

Le diagramme de séquence relatif à la modification d'une fiche candidat est représenté dans la Figure 3.10.

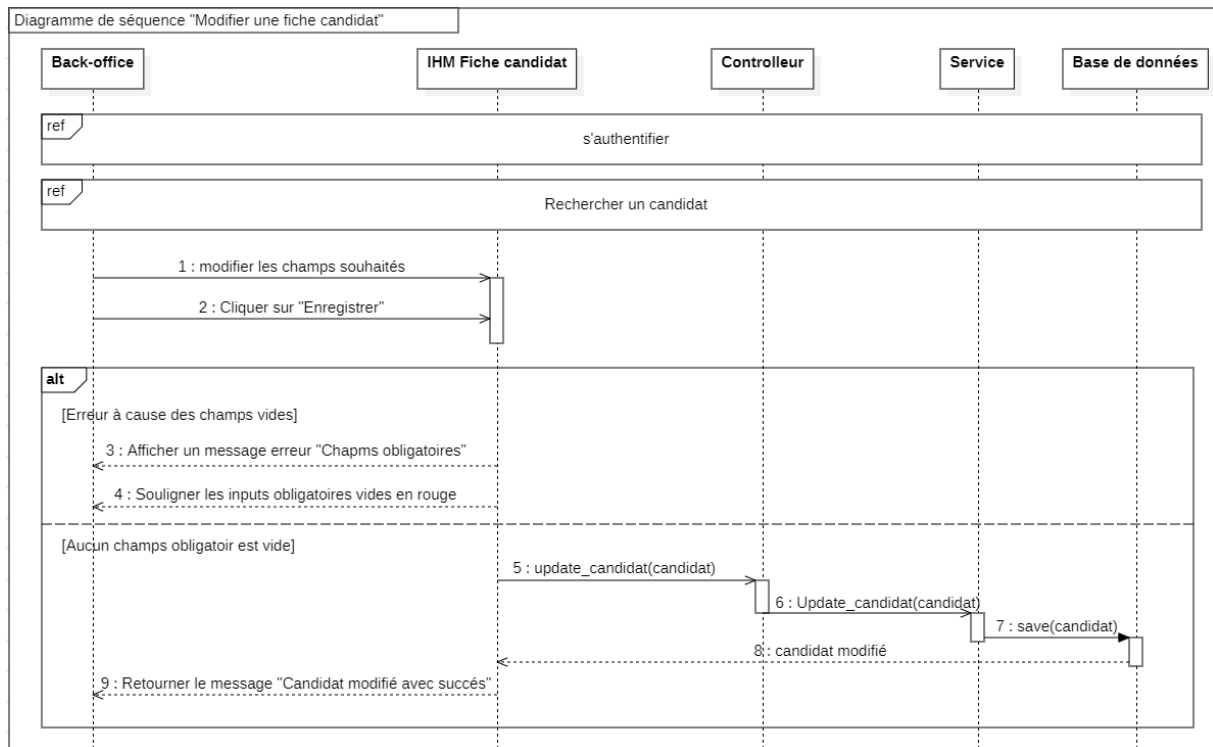


FIGURE 3.10 : Diagramme de séquence "Modifier une fiche candidat"

Après son authentification, et la recherche de la fiche d'un candidat, le back-office procède aux modifications de champs qu'il juge nécessaires dans le formulaire affiché, puis clique sur le bouton "Enregistrer". Le système vérifie que les champs obligatoires sont remplis et enregistre les modifications. Enfin, un message contenant "Candidat modifié avec succès" est affiché.

Conclusion

Dans le présent chapitre, nous avons présenté la conception générale de notre application. Par la suite, nous avons passé en revue la conception détaillée, en enchaînant par le diagramme de classes ainsi que les diagrammes de séquences pour mieux visualiser l'interaction des acteurs avec le système. Dans le prochain chapitre, nous allons montrer comment nous avons traduit cette étude conceptuelle et à l'aide de quels outils nous avons pu mettre en place notre application.

RÉALISATION

Plan

I	Environnement de travail	33
II	Les interfaces utilisateur	35

Introduction

Tenant compte des besoins fixés et des choix conceptuels effectués, nous consacrons ce chapitre à l'exposition du travail réalisé. Nous commençons par la description de l'environnement matériel et logiciel utilisés. Nous terminons par des captures d'écrans traduisant le déroulement du projet et présentant les fonctionnalités développées.

I Environnement de travail

Dans ce paragraphe, nous présentons les outils matériels et logiciels adoptés pour l'achèvement de notre projet.

I.1 Environnement matériel

Notre application a été réalisée sur un ordinateur ayant les caractéristiques suivantes :

- Un processeur Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz 2.20 GHz.
- Une mémoire RAM de 12 Go.
- Un disque dur de 1 To.
- Un système d'exploitation Windows 10.

I.2 Environnement logiciel

Dans cette partie, nous présentons les différents logiciels et outils utilisés :

- **StarUML**

C'est un logiciel de modélisation UML qui gère la plupart des diagrammes spécifiés dans la norme

UML 2.0.

— **PostgreSQL**

C'est un puissant système de gestion de bases de données open source. PostgreSQL fonctionne sur tous les principaux systèmes d'exploitation.

— **Intellij IDEA**

C'est un environnement de développement intégré de technologie Java. Il est destiné au développement de logiciels informatiques.

— **Visual Studio Code**

C'est un éditeur de code source, léger mais puissant. Il dispose d'un riche écosystème d'extensions pour d'autres langages (tels que C ++, C#, Java, Python, PHP, Go).

— **Docker**

C'est un logiciel libre permettant de lancer des applications dans des conteneurs logiciels. Un conteneur est une unité logicielle standard qui regroupe le code et toutes ses dépendances, et qui permet à l'application de s'exécuter rapidement et de manière fiable, d'un environnement informatique à un autre.

— **Overleaf**

C'est un éditeur LaTeX en ligne, collaboratif en temps réel. Il est utilisé pour la rédaction, l'édition et la publication de documents scientifiques.

I.3 Technologies utilisées

Dans cette partie, nous nous intéressons aux langages et aux bibliothèques utilisés tout au long de la réalisation de notre application tout en justifiant notre choix.

— **Spring Boot**

C'est un framework de développement applicatif JAVA open Source. Il est particulièrement recommandé pour le développement des microservices. Ses points forts sont l'auto-configuration ainsi que les annotations.

— **Spring Cloud Eureka Client**

Il permet aux différents microservices de s'enregistrer dans l'annuaire des services Eureka. La Figure 4.1 montre la liste des microservices détectés par le service Eureka.

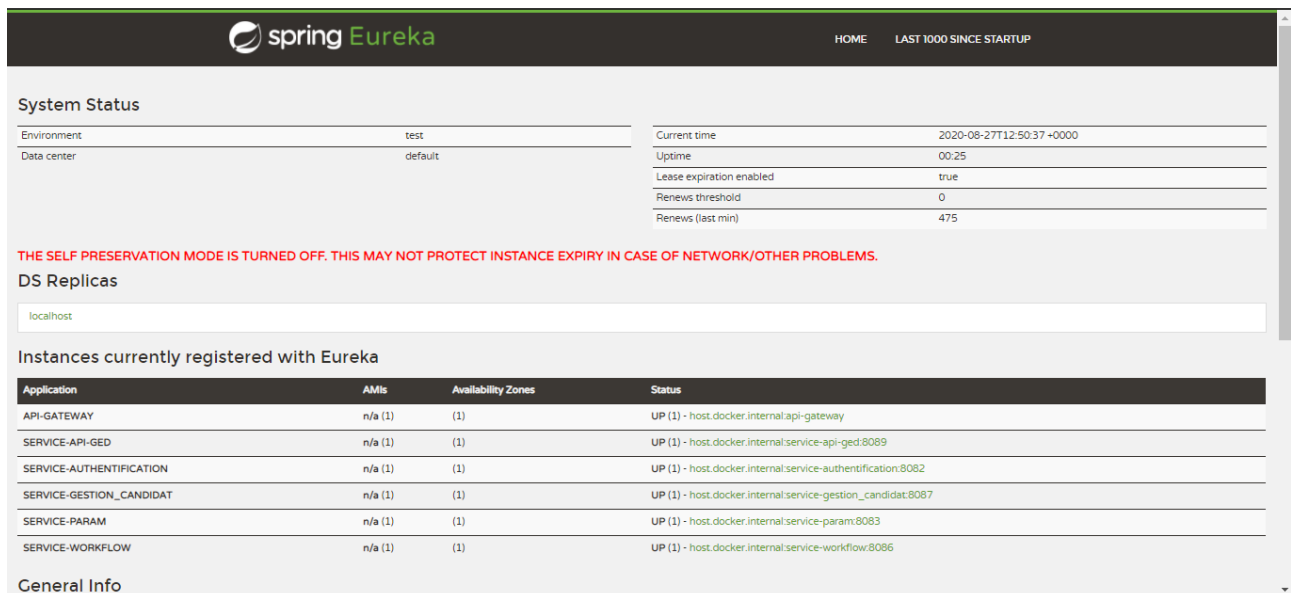


FIGURE 4.1 : Interface de Eureka Discovery Service

— Spring Data JPA

Il vise à améliorer d'une manière significative la mise en oeuvre des couches d'accès aux données en réduisant la quantité de code à écrire.

— QueryDsl

C'est un framework qui permet la construction des requêtes SQL typées dynamiquement via son API fluide.

— Angular 7

C'est framework JavaScript libre, développé par Google et utilisé pour créer des applications Web basées sur une seule page (Single Page Application). Il utilise des fonctionnalités de plate-forme Web modernes pour offrir de nouvelles expériences, ayant une installation à haute performance avec zéro-étape.

II Les interfaces utilisateur

Nous exposerons dans cette section les interfaces homme-machine qui sont un élément important pour la réussite d'une application.

II.1 Interface d'authentification

Pour pouvoir accéder aux différentes fonctionnalités de l'application, l'utilisateur doit saisir son login et son mot de passe comme le montre la Figure 4.2.

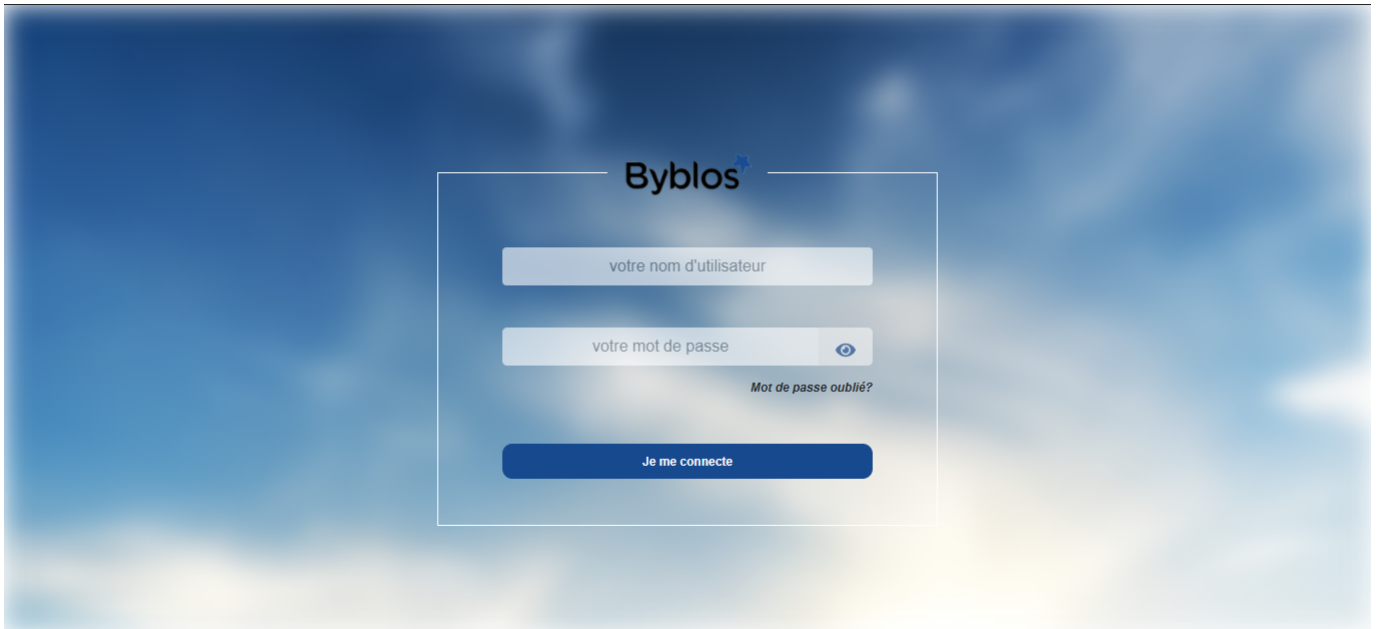


FIGURE 4.2 : Interface d'authentification

II.2 Interface d'accueil

Après l'authentification de l'utilisateur, la page d'accueil est chargée comme le montre la Figure 4.3. Elle contient le KPI des avis des entretiens composé des éléments suivants :

- Une liste prédéfinie des avis distingués par des couleurs différentes.
- Un graphe en secteurs de différentes couleurs. Chaque secteur contient le nombre total de candidats correspondant à un avis donné.
- L'année choisie ainsi que deux boutons pour sélectionner l'année suivante ou bien l'année précédente.
- Deux boutons en bas, l'un dirige vers une nouvelle fiche candidat, l'autre vers l'interface de recherche de candidats.

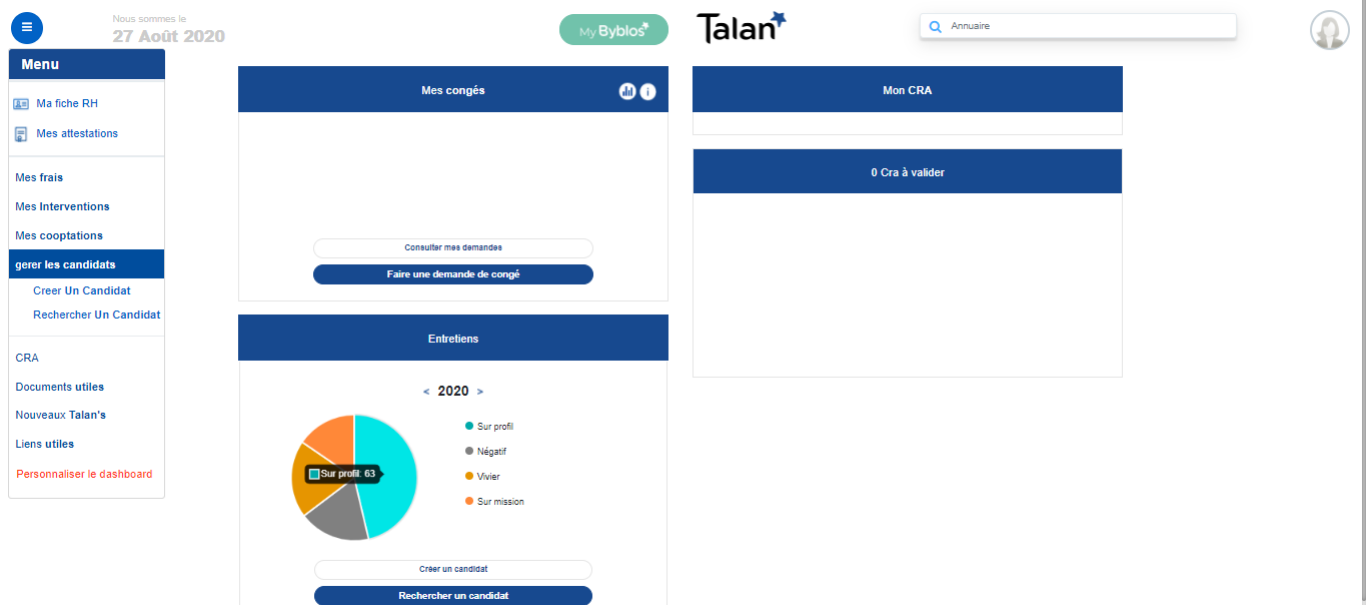


FIGURE 4.3 : Interface d'accueil

II.3 Interface d'ajout d'une fiche candidat

Cette interface est composée de trois parties :

1. **Fiche candidat** : elle contient les informations qui concernent le candidat. Cette partie se compose de quatre formulaires listés ci-dessous :

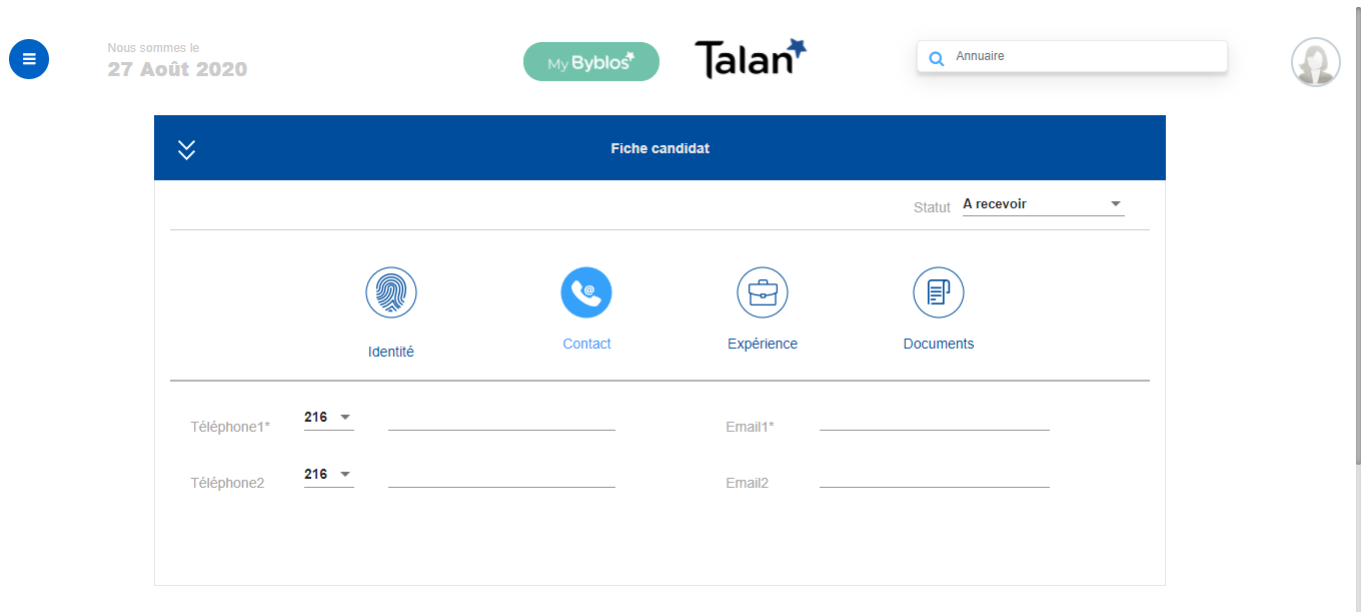
- **Interface identité :**

Cette interface, représentée dans la Figure 4.4, contient les informations liées à l'identité du candidat : son nom, son prénom, sa date de naissance ...

FIGURE 4.4 : Interface Identité

- **Interface contact :**

Cette interface, représentée dans la Figure 4.5, contient les coordonnées du candidat : son numéro du téléphone et son e-mail.

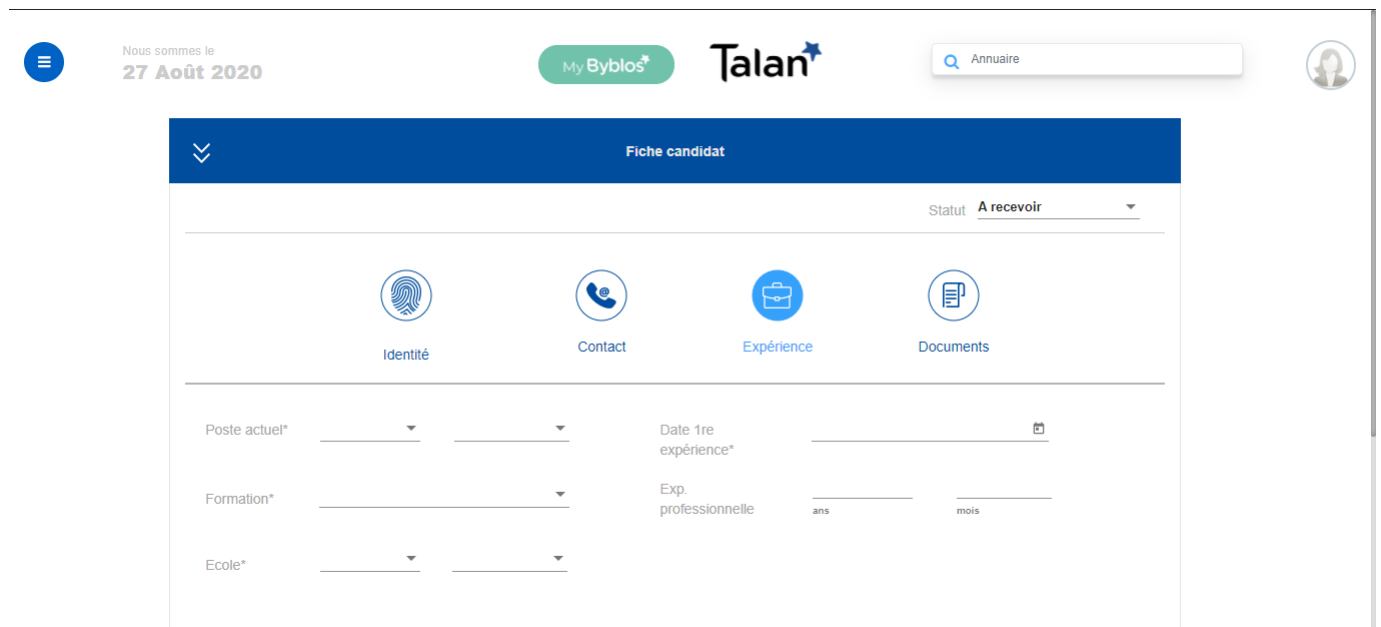


The screenshot shows the 'Fiche candidat' (Candidate Profile) interface. The header includes a date '27 Août 2020', a 'My Byblos' logo, and a 'Talan' logo. A search bar labeled 'Annuaire' is on the right. The main content area has a blue header 'Fiche candidat' and a 'Statut' dropdown set to 'A recevoir'. Below this are four tabs: 'Identité', 'Contact', 'Expérience', and 'Documents'. The 'Contact' tab is active, displaying two rows of input fields. The first row has 'Téléphone1*' and 'Email1*'. The second row has 'Téléphone2' and 'Email2'. The 'Téléphone' fields have a dropdown menu showing '216'.

FIGURE 4.5 : Interface Contact

- **Interface expérience :**

Cette interface, représentée dans la Figure 4.6, contient les informations concernant l'expérience professionnelle du candidat.



The screenshot shows the 'Fiche candidat' (Candidate Profile) interface. The header includes a date '27 Août 2020', a 'My Byblos' logo, and a 'Talan' logo. A search bar labeled 'Annuaire' is on the right. The main content area has a blue header 'Fiche candidat' and a 'Statut' dropdown set to 'A recevoir'. Below this are four tabs: 'Identité', 'Contact', 'Expérience', and 'Documents'. The 'Expérience' tab is active, displaying several input fields. On the left, there are three rows: 'Poste actuel*', 'Formation*', and 'Ecole*', each with a dropdown menu. On the right, there are two rows: 'Date 1re expérience*' with a calendar icon, and 'Exp. professionnelle' with 'ans' and 'mois' sub-fields.

FIGURE 4.6 : Interface Expérience

- **Interface Documents :**

Cette interface, représentée dans la Figure 4.7, contient les informations concernant la pièce

d'identité du candidat ainsi que son CV.

FIGURE 4.7 : Interface Documents

2. Planification des entretiens :

Cette interface, représentée dans la Figure 4.8, contient la liste des entretiens déjà planifiés, avec un bouton permettant de planifier un nouvel entretien.

FIGURE 4.8 : Interface Planification des entretiens

3. Fiche de pré-embauche

Cette interface contient les informations concernant le contrat de pré-embauche à signer par le candidat s'il est accepté. Cette interface est composée de trois formulaires listés ci-dessous :

- Interface contrat

Cette interface, représentée dans la Figure 4.9, contient les différentes informations contenues dans le contrat à signer.

FIGURE 4.9 : Interface Contrat

- **Interface fonction**

Cette interface, représentée dans la Figure 4.10, contient les informations concernant la fonction que le candidat va occuper.

FIGURE 4.10 : Interface Fonction

- **Interface rémunération**

Cette interface, représentée dans la Figure 4.11, contient les informations concernant la rémunération qui sera attribuée au candidat.

FIGURE 4.11 : Interface Rémunération

II.4 Interface de la recherche d'un candidat

Cette interface, représentée dans la Figure 4.12, comporte un filtre contenant les critères de recherche sélectionnés, avec la possibilité d'omettre certains d'entre eux ainsi que le formulaire de recherche et un tableau contenant le résultat obtenu.

FIGURE 4.12 : Interface de la recherche d'un candidat

II.5 Interface du résultat de la recherche

Cette interface, représentée dans la Figure 4.13, comporte une partie qui contient le nombre total de candidats répondant aux critères de la recherche effectuée, le nombre de candidats ayant un

II.7 Interface de la planification d'un entretien

Cette interface, représentée dans la Figure 4.15, contient le formulaire utilisé pour planifier un nouvel entretien.

FIGURE 4.15 : Interface de la planification d'un entretien

II.8 Interface contenant l'e-mail de validation de l'entretien

Cette interface, représentée dans la Figure 4.16, contient l'e-mail envoyé au recruteur après la planification de l'entretien par le back-office, pour qu'il soit accepté ou refusé.

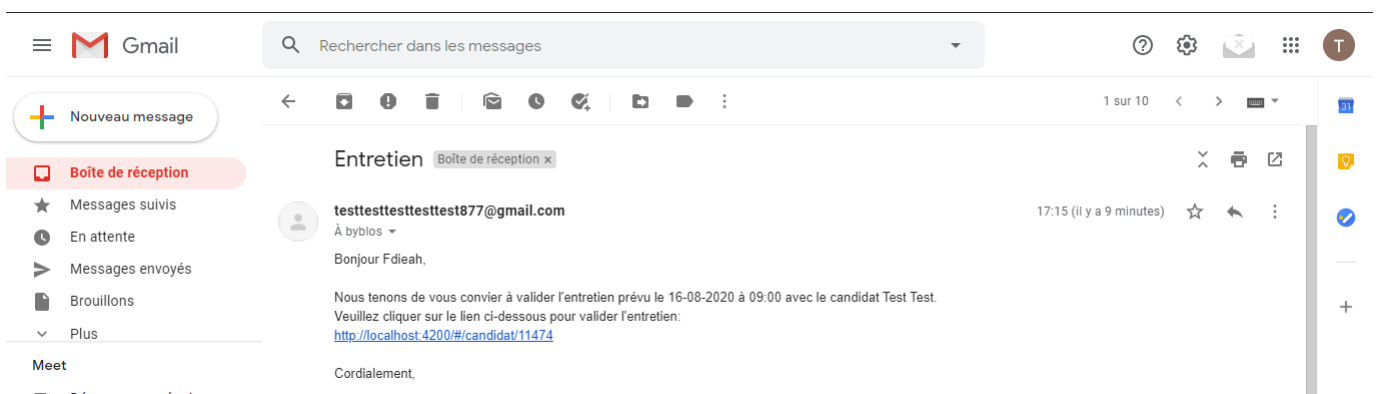


FIGURE 4.16 : Interface contenant l'e-mail de validation de l'entretien

II.9 Interface contenant l'e-mail du rejet de l'entretien

Si le recruteur rejette l'entretien, un e-mail sera envoyé au back-office pour lui notifier ce refus. Cette interface est représentée dans la Figure 4.17.

II.10 Interface contenant les différents statuts des entretiens

- "**Accepté**" : représenté sous forme d'une icône verte.
- "**Rejeté**" : représenté sous forme d'une icône rouge.
- "**En attente**" : représenté sous forme d'une icône bleue.

Planification des entretiens								
TYPE ENTRETIEN	DATE ENTRETIEN	BU	MANAGER	STATUT	AVIS			
RDV MNG 1	16/08/2020	TTC	DDIEAH Fdieah	✓				
RDV MNG 1	25/08/2020	TTC	LDEHJJ Edehjj	✗				
RDV MNG 3	10/08/2020	TTC	CDCADE Ldcade	ⓘ				🔍 📅 ✓ ✗
RDV MNG 1	25/08/2020	TTC	LDEHJJ Edehjj	✗				

II.11 Interface de la consultation du Workflow d'un entretien en attente

44

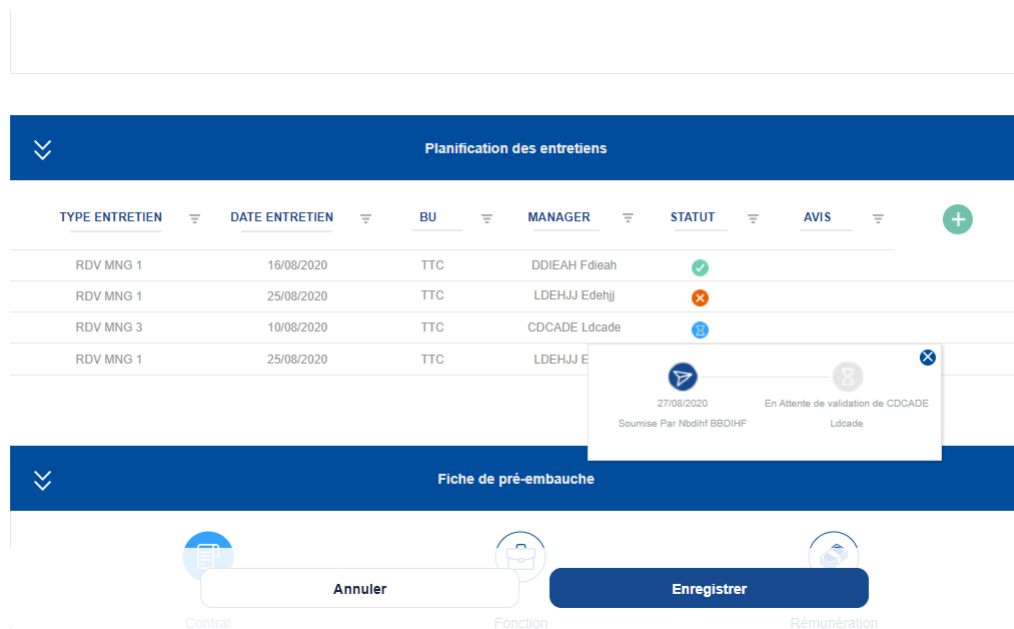


FIGURE 4.19 : Interface de la consultation du Workflow d'un entretien en attente de validation

II.12 Interface de la consultation du Workflow d'un entretien accepté

Cette interface, représentée dans la Figure 4.20, montre le Workflow d'un entretien accepté.

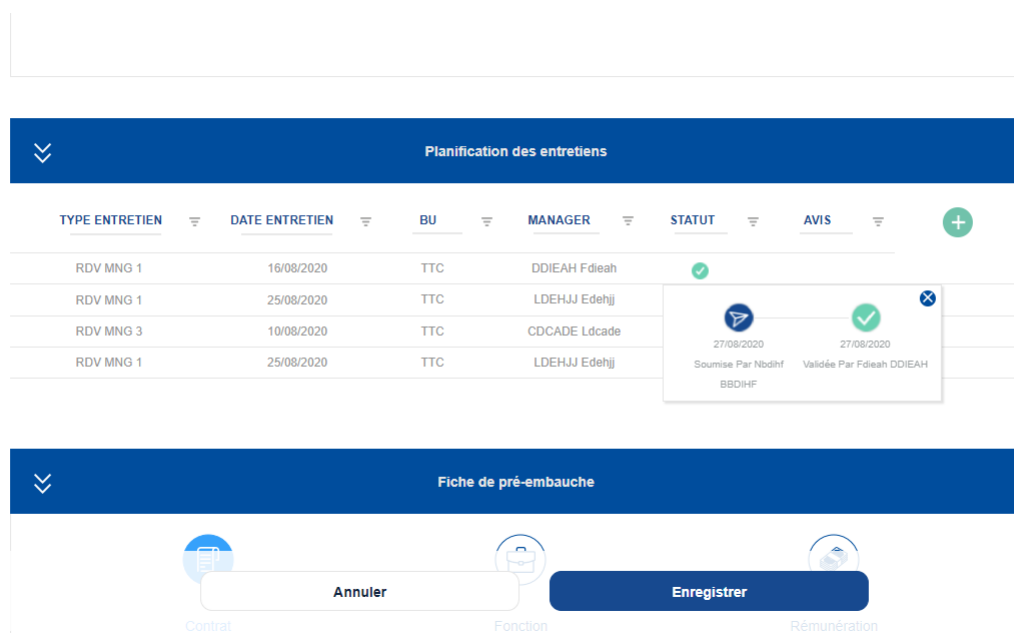


FIGURE 4.20 : Interface de la consultation du Workflow d'un entretien accepté

II.13 Interface de la consultation du Workflow d'un entretien rejeté

Cette interface, représentée dans la Figure 4.21, montre le Workflow d'un entretien rejeté.

The screenshot displays a software interface for interview scheduling. The top section, titled 'Planification des entretiens', contains a table with the following data:

TYPE ENTRETIEN	DATE ENTRETIEN	BU	MANAGER	STATUT	AVIS
RDV MNG 1	16/08/2020	TTC	DDIEAH Fdieah	✓	
RDV MNG 1	25/08/2020	TTC	LDEHJJ Edehij	✗	
RDV MNG 3	10/08/2020	TTC	CDCADE Ldcade		
RDV MNG 1	25/08/2020	TTC	LDEHJJ Edehij		

A pop-up window on the right shows a timeline of actions: '27/08/2020 Soumise Par Nbdihf BBDIHF' (marked with a blue circle) and '27/08/2020 Rejetée Par Edehij LDEHJJ' (marked with a red X). Below the table is a 'Fiche de pré-embauche' section with three tabs: 'Contrat', 'Fonction', and 'Rémunération'. The 'Fonction' tab is active, showing 'Annuler' and 'Enregistrer' buttons.

FIGURE 4.21 : Interface de la consultation du Workflow d'un entretien rejeté

Conclusion

La phase de réalisation est l'étape la plus exigeante en termes d'efforts et de temps. Elle nous a permis de concrétiser et d'approfondir nos connaissances académiques grâce à l'opportunité qui nous a été offerte pour la réalisation d'un projet informatique réel au sein d'une grande entreprise. Ce chapitre constitue en fait l'aboutissement des phases précédentes.

Conclusion générale

Adopter l'approche des micro-services n'est pas un sujet prospectif, mais une réalité pour les entreprises, d'autant plus que des géants du web comme Netflix et Amazon le trouvent efficace et fiable. En effet, les entreprises devraient accorder dorénavant une grande importance au choix de l'architecture de leur système en vue d'améliorer la performance de leurs activités.

Le projet de refonte architecturale de l'ERP "BYBLOS" réalisé par **Talan Tunisie**, particulièrement le module de gestion des candidats, est un pas en avant vers une agilité de bout en bout. L'objectif final, n'est qu'une pure volonté stratégique, vise à faire évoluer l'organisation de l'entreprise, mais aussi à lui apporter de nouveaux projets plus robustes et plus efficaces.

Ainsi, **Talan Tunisie** a jugé utile d'opter pour l'architecture microservices de son ERP qui permet de pallier à certaines insuffisances de l'architecture monolithique, notamment la lourdeur de la maintenance et l'affectation du système par la défaillance d'une seule partie de ce dernier. Quelques modules ont déjà été migrés vers cette architecture (le module d'authentification, le module RH...). Notre travail a consisté à suivre toutes les étapes nécessaires (analyse préalable, conception, réalisation, intégration) pour assurer une telle migration et bénéficier des ses avantages.

Signalons toutefois que l'intégration du microservice de gestion des candidats dans le service de découverte et dans le Gateway nous a demandé un effort particulier pour trouver le meilleur moyen de réaliser une telle opération qui constitue pour nous une tâche tout à fait nouvelle.

Ce travail est certes perfectible. Certaines fonctionnalités pourront être intégrées ultérieurement dans le module de gestion des candidats. Citons en particulier :

- L'intégration d'un système de notification pour prévenir le recruteur de la planification d'un entretien, et informer le back-office de la validation de l'entretien par le recruteur.
- L'intégration d'une liste de questions qui peuvent être posées aux candidats par les recruteurs à chaque étapes de l'entretien.
- L'intégration d'un module de Machine Learning qui permet d'analyser les réponses des candidats.

Nétographie

- [1] <https://tn.talan.com/> (consulté le 21/02/2020)
- [2] <https://www.occitech.fr/blog/2015/02/les-microservices/> (Consulté le 25/02/2020)
- [3] <https://blog.codeinsider.fr/architecture-monolith-ou-microservices-laquelle-choisir-et-pourquoi/> (Consulté le 15/03/2020)
- [4] <https://www.ionos.fr/digitalguide/sites-internet/developpement-web/architecture-microservices/> (Consulté le 22/03/2020)
- [5] <https://openclassrooms.com/fr/courses/4668056-construisez-des-microservices/5122300-apprenez-larchitecture-microservices> (Consulté le 12/05/2020)
- [6] <https://www.redhat.com/fr/topics/microservices/what-are-microservices> (Consulté le 26/05/2020)
- [7] <https://cloud.google.com/solutions/migrating-a-monolithic-app-to-microservices-gke?hl=fr> (Consulté le 13/06/2020)
- [8] <https://juliendubreuil.fr/blog/developpement/de-application-monolithique-aux-architectures-microservices-ou-orientees-composants/> (Consulté le 23/06/2020)

Résumé

Le présent rapport synthétise le travail effectué dans le cadre du projet de fin d'études pour l'obtention du diplôme national d'ingénieur en informatique au sein de l'entreprise Talan Tunisie Consulting. Le travail consiste à une refonte du module de gestion des candidats de BYBLOS qui est un ERP (Entreprise Ressource Planning) de Talan Tunisie Consulting, en se basant sur une architecture en microservices ainsi que trouver des alternatifs pour quelques technologies qui ont présenté leurs limites.

Mots clés : Microservices, Spring Boot, Spring Cloud, Spring Data, QueryDsl, Angular

Abstract

This report summarizes the work carried out as part of the end-of-study project for obtaining the national software engineering degree within the company Talan Tunisia Consulting. The work consists in redesigning the BYBLOS candidate management module, which is an ERP (Enterprise Resource Planning) of Talan Tunisia Consulting, based on a microservices architecture and finding alternatives for some technologies that have presented their limitations.

Keywords : Microservices, Spring Boot, Spring Cloud, Spring Data, QueryDsl, Angular
