

République Algérienne Démocratique et Populaire  
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique  
Université Constantine 2- Abdelhamid Mehri



Faculté des Nouvelles Technologies de l'Information et la Communication  
Département des Technologies des Logiciels et Systèmes d'Information

---

Projet de fin d'études pour l'obtention du diplôme de  
Master en Informatique

Option : Génie Logiciel

Thème

Application Android pour le rapprochement des  
entreprises et des demandeurs d'emplois et de stages

Dirigé par :  
Mr. Saadi Adel

Réalisé par :  
Chokri TOUAHRIA  
Merwan TADJ

## *Remerciements*

Nous tenons tout d'abord à remercier DIEU le tout puissant de nous avoir donné la force et le courage pour réaliser notre projet.

On tient à remercier sincèrement notre encadreur docteur Saadi Adel, qui a été toujours à l'écoute et très disponible tout au long de la réalisation de ce projet, ainsi pour l'inspiration, l'aide et le temps qu'il a bien voulu nous accorder.

Nous exprimons également notre sincère gratitude aux membres de notre jury et qui nous honoreront par leur évaluation de notre travail.

## Dédicaces

À tous ceux qui me sont chers

Phokri et Merwan

## Résumé

Les applications mobiles occupent une grande partie du marché des applications et sont considérées aujourd’hui comme étant les plus utilisées. Cette mise en vogue est due au facteur de la mobilité qui a permis à des utilisateurs de profiter de ces applications n’importe où et n’importe quand, d’une manière très facile et efficace.

Dans ce cadre nous avons développé une application Android pour le rapprochement des entreprises et des demandeurs d'emplois/stages, dans le but de maîtriser certains aspects du développement mobile, en relation avec la modélisation (l’analyse et conception) et l’implémentation.

**Mots clés : Application mobile, Système Android, modélisation.**

## **Abstract**

Mobile applications occupy a large part of the application market and are now considered the most widely used. This popularity is due to the mobility factor that allowed users to take advantage of these applications anywhere and anytime, in a very easy and efficient way.

In this context, we have developed an Android application to bring together companies and job/internships seekers, with the aim of mastering certain aspects of mobile development, in relation to modeling (analysis and design) and implementation.

**Keywords:** Mobile application, Android system, modeling.

## **الملخص**

تحتل التطبيقات النقالة جزأً كبيراً من سوق التطبيقات وتعتبر الآن الأكثر استخداماً. ويرجع هذا الانتشار إلى طبيعتها المتنقلة التي تسمح للمستخدمين بالاستفادة من هذه التطبيقات بكل حرية في أي مكان وأي زمان بطريقة سهلة وفعالة.

وفي هذا السياق قمنا بتطوير تطبيق أندرويد خاص للنقارب بين المؤسسات والباحثين عن عمل/تربيص، من أجل التعرف وإتقان جوانب تطوير التطبيقات النقالة بالعلاقة مع النمذجة (التحليل والتصميم) والتنفيذ.

**الكلمات المفتاحية:** تطبيق نقال، نظام أندرويد، نمذجة.

## Table des matières

Introduction générale :.....	1
<b>Chapitre 01 : Présentation de quelques approches de développement itératives et incrémentales</b>	
1. Introduction .....	4
2. Langage de modélisation UML.....	4
3. Approche de développement.....	6
4. UP (Unified Process).....	7
5. Méthodes Agiles.....	9
5.1. Les valeurs fondamentales de l'agilité .....	9
5.2. Les principes généraux de l'agilité .....	10
6. eXtreme Programming (XP) .....	11
6.1. Cycle XP .....	11
6.2. Les Valeurs de XP .....	12
6.3. Les pratiques d'XP.....	13
7. Agile Modeling (AM) .....	15
8. L'approche de développement de Pascal Roques .....	16
8.1. Identification des besoins et spécification des fonctionnalités .....	17
8.2. Phases d'analyse .....	18
8.3. Phase de conception .....	20
9. Conclusion.....	21
<b>Chapitre 02: Introduction aux applications et S.E Android</b>	
1. Introduction .....	23
2. Application mobile.....	23
3. Le système d'exploitation Android .....	24
3.1. Les points forts du Système d'exploitation Android .....	24
3.2. L'architecture d'Android .....	25
4. Les applications Android .....	27
4.1. Fonctionnement des applications Android.....	27
5. Les approches de réalisation des applications Android .....	29
5.1. L'approche Application Web.....	29

5.2.	L'approche Native .....	30
5.3.	Approche Hybrid .....	30
6.	Développement sous l'approche Native.....	32
6.1.	Les contraintes d'environnement :.....	32
6.2.	Outils nécessaires pour le développement Native .....	32
6.3.	Les composants d'une application Android Native.....	34
7.	Conclusion.....	37

### **Chapitre 03 : Analyse et conception d'une application mobile pour la recherche d'emplois et de stages**

1.	Introduction .....	39
2.	Présentation du notre projet.....	39
3.	Identification des besoins et spécification des fonctionnalités.....	41
3.1.	Elaborations des diagrammes des cas d'utilisation.....	41
3.2.	Description textuelle des cas d'utilisation .....	43
3.3.	Spécification détaillée des besoins (Les Diagrammes de séquence système) .....	47
4.	La phase d'analyse (Diagrammes d'activités de navigation).....	55
5.	Phase de conception .....	62
5.1.	Diagrammes de séquence system détaillé.....	62
6.	Conclusion.....	73

### **Chapitre 04 : L'implmentation de l'application**

1.	Introduction .....	75
2.	Description de l'environnement du travail.....	75
2.1.	Plateformes matérielles.....	75
2.2.	Environnements logiciels.....	76
2.3.	Langages utilisés.....	78
3.	Choix technique d'implémentation .....	78
3.1.	Implémentation en modèle MVC .....	78
3.2.	L'architecture de l'application.....	79
3.3.	Cryptage des données .....	80
3.4.	Les bibliothèques utilisées .....	81
3.5.	Le Choix de niveau de l'API : .....	81

3.6.	Proguard : .....	82
3.7.	Material design .....	82
3.8.	Constraint Layout : .....	83
4.	Description des interfaces de l'application .....	83
5.	Conclusion.....	89
	Conclusion générale et perspectives .....	91
	Bibliographie.....	93

## Liste des figures

Figure 1. 1. L'évolution des versions d'UML [2].....	5
Figure 1. 2. Les phases d'UP.....	8
Figure 1. 3. Cycle de XP [7].....	12
Figure 1. 4 Portée d'Agile Modeling.....	15
Figure 1. 5. Les besoins donnent lieu à des cas d'utilisation et à une maquette [11]. .....	18
Figure 1. 6. Passage de l'analyse à la conception préliminaire [11]. .....	20
Figure 1. 7. Chaîne complète de la démarche de modélisation sous L'approche Pascal Roques [11]. .....	21
Figure 2. 1. Les composants d'architecture d'Android OS [16]. .....	26
Figure 2. 2. La création d'un code intermédiaire à partir du code source.....	28
Figure 2. 3. Compilation et déploiement d'une application mobile. ....	29
Figure 2. 4. Les approches de réalisation d'une application mobile [19].....	29
Figure 2. 5. Interface d'IDE Android Studio.....	33
Figure 2. 6. Interface d'Emulateur Android.....	33
Figure 2. 7. Cycle de vie d'Activity [16]. .....	34
Figure 2. 8. Cycle de vie de Fragment (avec une comparaison avec la Cycle d'Activity) [16]....	35
Figure 2. 9.Cycle de vie de Service [16]. .....	36
Figure 3. 1. Diagrammes de cas de d'utilisation pour l'acteur « Candidat ». .....	41
Figure 3. 2. Diagrammes de cas de d'utilisation pour l'acteur « Recruteur »......	42
Figure 3. 3. Diagrammes de cas de d'utilisation pour l'acteur « Administrateur ».....	42
Figure 3. 4. Diagramme de séquence système du cas « S'authentifier ».....	47
Figure 3. 5. Diagramme de séquence système du cas « Incrire ». .....	48
Figure 3. 6. Diagramme de séquence système du cas « Gérer profil ». .....	49
Figure 3. 7. Diagramme de séquence système du cas « Ajouter compétence ». .....	50
Figure 3. 8. Diagramme de séquence système du cas « Consulter la liste des offres ».....	50
Figure 3. 9. Diagramme de séquence système du cas « Postuler pour une offre ». .....	51
Figure 3. 10. Diagramme de séquence système du cas « Ajouter une offre aux favoris ».....	51
Figure 3. 11. Diagramme de séquence système du cas « Consulter les informations d'entreprise ». .....	52
Figure 3. 12. Diagramme de séquence système du cas « Publier une offre ». .....	52
Figure 3. 13. Diagramme de séquence système du cas « Supprimer une offre ». .....	53
Figure 3. 14. Diagramme de séquence système du cas « Confirmer une demande d'inscription ». .....	53
Figure 3. 15. Diagramme de séquence système du cas « Rejeter une demande d'inscription ». ..	54
Figure 3. 16. Diagramme d'activité du cas « S'authentifier ».....	55
Figure 3. 17. Diagramme d'activité du cas « Incrire ». .....	56

Figure 3. 18. Diagramme d'activité du cas « Gérer profil » .....	56
Figure 3. 19. Diagramme d'activité du cas « Ajouter compétence ».....	57
Figure 3. 20. Diagramme d'activité du cas « Consulter liste des offres » .....	57
Figure 3. 21. Diagramme d'activité du cas « postuler pour une offre » .....	58
Figure 3. 22. Diagramme d'activité du cas « Ajouter une offre aux favoris ».....	58
Figure 3. 23. Diagramme d'activité du cas « Consulter les informations d'entreprise ».....	59
Figure 3. 24. Diagramme d'activité du cas « Publier une offre » .....	59
Figure 3. 25. Diagramme d'activité du cas « Supprimer offre ».....	60
Figure 3. 26. Diagramme d'activité du cas « Confirmer demande d'inscription ».....	60
Figure 3. 27. Diagramme d'activité du cas « Rejeter demande d'inscription ».....	61
Figure 3. 28. Diagramme de séquence détaillé du cas « s'authentifier » .....	62
Figure 3. 29. Diagramme de séquence détaillé du cas « Incrire ».....	63
Figure 3. 30. Diagramme de séquence détaillé du cas « Gérer profil ».....	64
Figure 3. 31. Diagramme de séquence détaillé du cas « Ajouter Compétence » .....	65
Figure 3. 32. Diagramme de séquence détaillé du cas « Consulter liste des offres ».....	65
Figure 3. 33. Diagramme de séquence détaillé du cas « Postuler pour une offre ».....	66
Figure 3. 34. Diagramme de séquence détaillé du cas « Ajouter offre aux favoris ».....	66
Figure 3. 35. Diagramme de séquence détaillé du cas « Consulter les informations d'entreprise ».	67
Figure 3. 36. Diagramme de séquence détaillé du cas « Publier offre » .....	67
Figure 3. 37. Diagramme de séquence détaillé du cas « Supprimer offre ».....	68
Figure 3. 38. Diagramme de séquence détaillé du cas « Confirmer demande d'inscription ».....	68
Figure 3. 39. Diagramme de séquence détaillé du cas « Rejeter demande d'inscription ».....	69
Figure 3. 40. Diagramme de classe du cas « S'authentifier » .....	69
Figure 3. 41. Diagramme de classe du cas « Incrire » .....	70
Figure 3. 42. Diagramme de classe de conception du cas « Gérer profil » .....	70
Figure 3. 43. Diagramme de classe de conception du cas « Consulter liste offres ».....	70
Figure 3. 44. Diagramme de classe de conception du cas « Ajouter compétence ».....	71
Figure 3. 45. Diagramme de classe de conception du cas « Postuler pour une offre ».....	71
Figure 3. 46. Diagramme de classe de conception du cas « Ajouter offre aux favoris » .....	71
Figure 3. 47. Diagramme de classe de conception du cas « Consulter informations d'entreprise ».	72
Figure 3. 48. Diagramme de classe de conception du cas « Publier offre » .....	72
Figure 3. 49. Diagramme de classe de conception du cas « Supprimer offre » .....	72
Figure 3. 50. Diagramme de classe de conception du cas « Confirmer une demande d'inscription ».....	73
Figure 3. 51. Diagramme de classe de conception du cas « Rejeter une demande d'inscription ».	73

Figure 4. 1. Passage de conception, vers Code selon Modèle MVC.....	79
Figure 4. 2. Architecture de notre application.....	80
Figure 4. 3. Fonctionnement de cryptage dans notre application.....	80
Figure 4. 4. Distribution globale des versions Android juin 2017.....	82
Figure 4. 5. Exemple de contrôles de Material Design.....	82
Figure 4. 6. Interface de l'authentification.....	83
Figure 4. 7. Interfaces de demande d'inscription de candidat.....	84
Figure 4. 8. Interfaces de demande d'inscription de recruteur.....	84
Figure 4. 9. Message pour les utilisateurs non validés.....	85
Figure 4. 10. Interfaces représentent le menu principal et le profil de candidat.....	85
Figure 4. 11. Différentes interfaces des compétences (consulter, modifier, ajouter).....	86
Figure 4. 12. Interface présente la liste des offres et consulter les informations d'une offre.....	86
Figure 4. 13. Interfaces de profil de recruteur (informations d'entreprise).....	87
Figure 4. 14. Interface décrit la fonctionnalité publier une offre.....	88
Figure 4. 15. Interfaces présentent la liste des offres et les demandes.....	89
Figure 4. 16. Interface liste de demandes d'inscription et exemple de vérification.....	89

## Liste des tableaux

Table 2. 1. Tableau comparatif entre les approches de réalisation d'App Mobile(Android).....	31
Table 2. 2. La différence entre l'Activity et le Fragment .....	35
Table 3. 1. Fiche descriptif du cas « S'authentifier ».....	43
Table 3. 2. Fiche descriptif du cas « Incrire ».....	43
Table 3. 3. Fiche descriptif du cas « Gérer profil ».....	44
Table 3. 4. Fiche descriptif du cas « Ajouter compétence ».....	44
Table 3. 5. Fiche descriptif du cas « Consulter liste des offres ».	44
Table 3. 6. Fiche descriptif du cas « Postuler pour une offre ».	45
Table 3. 7. Fiche descriptif du cas « Ajouter offre aux favoris ».	45
Table 3. 8. Fiche descriptif du cas « Consulter les informations d'entreprise ».	45
Table 3. 9. Fiche descriptif du cas « Publier une offre ».....	46
Table 3. 10. Fiche descriptif du cas « supprimer une offre ».	46
Table 3. 11. Fiche descriptif du cas « Confirmer une demande d'inscription ».	46
Table 3. 12. Fiche descriptif du cas « Rejeter une demande d'inscription ».	47

---

## **Introduction générale**

---

## **Introduction générale :**

Le développement des technologies de l'information et de la communication, a évolué avec le temps. Dans les dernières années, le paysage mobile était à bien des égards, un monde à part entière où nous sommes maintenant. Les smartphones, ces périphériques ressemblant beaucoup plus aux PDA (personal digital assistant) avec un téléphone intégré ont un impact énorme sur la vie des gens tout en la rendant intéressante.

Les applications mobiles ont pu occuper une grande partie du marché des applications et sont considérées aujourd'hui comme étant les plus utilisées. Cette mise en vogue est due au facteur de la mobilité qui a permis à des utilisateurs de profiter de ces applications n'importe où et n'importe quand, d'une manière très facile et efficace.

Les applications mobiles ont trouvé leurs utilisations dans plusieurs domaines tels que la communication, l'éducation, la santé, les loisirs, etc. Le domaine d'emploi est aussi un domaine qui a pu bénéficier de cette évolution technologique. De grands réseaux professionnels comme LinkedIn (le plus grand réseau professionnel au monde), Opportunity et Emploitic (en Algérie) sont émergés à travers leurs applications mobiles.

La recherche d'emplois et l'accessibilité aux stages restent toujours parmi les tâches les plus difficiles de nos jours même avec la multiplicité des solutions actuels. C'est dans ce cadre, qu'intervient notre projet où nous allons développer une application Android dédiée à l'emploi, au recrutement et aussi à la recherche des stages. Cette solution est conçue pour faciliter la recherche d'emplois aux candidats et permettre aux recruteurs de trouver les profils les plus convenables pour un poste dans les plus brefs délais.

Notre présent mémoire est composé de quatre chapitres dont :

**Le premier chapitre** porte sur quelques approches de développement logiciel itératifs et incrémentales à savoir l'approche UP et les méthodes Agile. Nous allons présenter aussi dans ce chapitre l'approche de développements que nous allons suivre au cours de la réalisation de ce projet.

**Le deuxième chapitre** présente un état de l'art sur le système d'exploitation Android, son apparition et son architecture. Il présente également, une vue plus technique sur les applications Android, leur développement et fonctionnement.

**Le troisième chapitre** présente le résultat de l'analyse et de la conception de notre application.

**Le quatrième chapitre** présente les outils de développement que nous avons utilisés, nos choix de développement et leurs impacts sur notre projet. Enfin nous illustrons l'utilisation de notre application à travers quelques interfaces graphiques.

# **CHAPITRE 01**

---

**Présentation de quelques approches de développement itératives et incrémentales**

---

## 1. Introduction

Modéliser un système avant sa réalisation permet de mieux comprendre le fonctionnement du système. C'est également un bon moyen de maîtriser sa complexité et d'assurer sa cohérence. Un modèle est un moyen commun, précis, qui est connu par tous les membres de l'équipe et il est donc, à ce titre, un vecteur privilégié pour communiquer. Cette communication est essentielle pour aboutir à une compréhension commune aux différentes parties prenantes et précise d'un problème donné.

Un projet informatique, quelle que soit sa taille et la portée de ses objectifs, nécessite la mise en place d'un planning organisationnel tout au long de son cycle de vie. C'est ainsi qu'est apparue la notion de méthode. Une méthode, dans le contexte informatique, peut être définie comme une démarche fournissant une méthodologie et des notations standards qui aident à concevoir des logiciels de qualité. Dans ce chapitre nous allons présenter le langage de modélisation UML, aussi nous allons présenter quelques méthodologies de développements et nous allons choisir une de ces méthodologies pour l'utiliser dans notre projet.

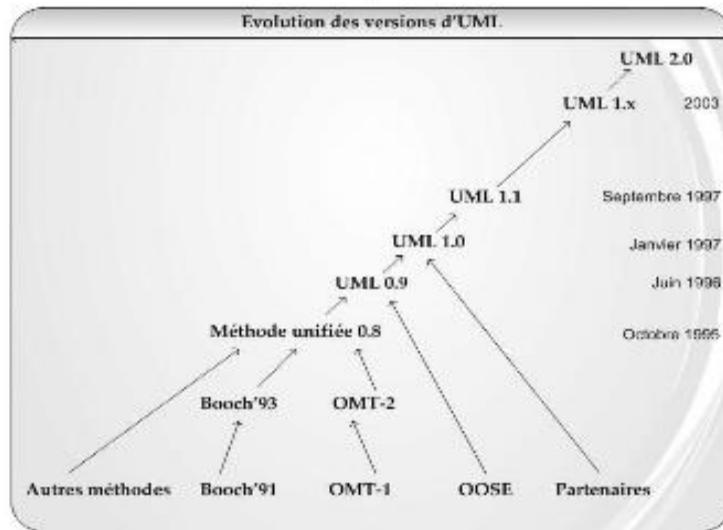
## 2. Langage de modélisation UML

UML (Unified Modeling Language) se définit comme un langage de modélisation graphique et textuelle destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. C'est un formalisme très abouti et non propriétaire de la modélisation objet [1].

UML unifie les concepts orientés objet et les notations nécessaires aux différentes activités d'un processus de développement et offre, par ce biais, le moyen d'établir le suivi des décisions prises, depuis la spécification jusqu'au codage [1].

Ce langage a démarré avec la version 0.8 intégrant les méthodes BOOCH 93 et OMT. Par la suite il y a eu l'avènement de la version 0.9 ayant intégré la méthode OOSE. La version 1.0, proposé à l'OMG en 1996, fut finalement standardisé en 1997 sous la version 1.1.

Depuis cette année il y a eu quatre révisions du standard (d'UML 1.1 à UML 1.5 en 2003). Les dernières améliorations étant conséquentes, UML est passé à une nouvelle version : UML 2.0 (ou UML 2), abrégé souvent en U2. La figure 1.1 ci-dessous montre l'évolution des versions d'UML depuis sa genèse.



**Figure 1. 1.** L'évolution des versions d'UML [2].

UML propose différents modèles pour représenter les différents points de vue de la modélisation. Les vues sont :

- **La vue logique (intégrité de conception)** : perspective abstraite de la solution (classes, relations, machines états-transitions, etc.)
- **La vue des cas d'utilisation (intégrité de conception)** : qui guide et justifie les autres: moyen rigoureux et systématique pour guider la modélisation (cas d'utilisation, scénarios, collaborations d'objets, machines à états).
- **La vue des composants (intégrité de gestion du code)** : perspective physique de l'organisation du code (modules, composants, concepts du langage ou de l'environnement d'implémentation).
- **La vue des processus (intégrité d'exécution)** : perspective sur les activités concurrentes et parallèles (tâches et processus).
- **La vue de déploiement (intégrité de performance)** : répartition du système (logiciel) à travers un réseau.

UML dans sa version2 propose treize diagrammes qui peuvent être utilisés dans la description d'un système. Ces diagrammes sont regroupés dans deux grands ensembles [3] :

- Diagrammes structurels ou diagrammes statiques (UML Structure) :
  - Diagramme de classes (class diagram).

- Diagramme d'objets (Objects diagram).
- Diagramme de composants (Component diagram).
- Diagramme de déploiement (Deployment diagram).
- Diagramme de paquetages (Package diagram).
- Diagramme de structures composites (Composite structure diagram).
- Diagrammes comportementaux ou diagrammes dynamiques (UML Behavior):
  - Diagramme de cas d'utilisation (Use case diagram).
  - Diagramme d'activités (Activity diagram).
  - Diagramme d'états-transitions (State machine diagram).
  - Diagramme global d'interaction (Interaction overview diagram).
  - Diagramme de séquence (Sequence diagram).
  - Diagramme de communication (Communication diagram).
  - Diagramme de temps (Timing diagram).

### **3. Approche de développement**

Un processus définit une séquence d'étapes, en partie ordonnées, qui concourent à l'obtention d'un système logiciel ou à l'évolution d'un système existant. L'objet d'un processus de développement est de produire des logiciels de qualité qui répondent aux besoins de leurs utilisateurs dans des temps et des coûts prévisibles.

Dans les cycles de vie classique (cycle linéaire), les phases du développement se suivent dans l'ordre et sans retour en arrière (effet tunnel). Cela fait que les risques sont élevés et non contrôlés, la découverte tardive des anomalies, preuve tardive de bon fonctionnement et défauts qui engendre beaucoup de dégâts (dépassement de budgets et de délais, arrêt de projet, etc.).

Pour pallier les problèmes causés par les cycles linéaires, des améliorations ont été introduites, à savoir [4] :

- Distinction, entre phases et activités.
- Construction itérative du système
- Chaque itération produit un nouvel incrément.
- Chaque nouvel incrément a pour but de maîtriser une partie des risques et apporte une preuve tangible de faisabilité ou d'adéquation.

- Enrichissement par une série de prototypes.
- Les versions livrées correspondent à une étape de la chaîne des prototypes.

Il existe plusieurs approches de développement itératif et incrémental tel qu'UP (2TUP, RUP, OpenUP etc.), les méthodes Agile (XP, Scrum, Crystal clear etc.).

### 4. UP (Unified Process)

Le processus unifié UP (Unified Process) est un processus de développement logiciel né de la fusion des travaux d'Ivar Jacobson, Grady Booch et de James Rumbaugh, les trois concepteurs du langage UML. Fruit des meilleures pratiques de l'ingénierie logicielle [3].

Il s'agit d'une démarche s'appuyant sur la modélisation UML pour la description de l'architecture du logiciel (fonctionnelle, logicielle et physique) et la mise au point de cas d'utilisation permettant de décrire les besoins et exigences des utilisateurs.

Le Processus Unifié est décrit par ses auteurs comme une méthode pilotée par les cas d'utilisation, centrée sur l'architecture, itérative et incrémentale.

Les itérations d'UP s'inscrivent dans quatre phases successives dont la validation constitue des jalons importants du processus de développement (figure 1.2.) : initialisation (inception), élaboration, construction, transition [2].

#### – Initialisation :

Première phase du cycle de vie du processus unifié, permet de traduire une idée en vision de produit fini et présente l'étude de rentabilité pour ce produit. Elle essaie de répondre à un certain nombre de questions : que va faire le système pour les utilisateurs ? à quoi peut ressembler l'architecture d'un tel système ? Quels sont l'organisation et les coûts du développement de ce produit ? C'est à ce niveau où les principaux cas d'utilisation seront spécifiés. L'identification des risques majeurs, la mise sur place d'une architecture provisoire du système à concevoir et la préparation de la phase d'élaboration seront les principales tâches à effectuer durant cette étape de la création.

#### – Elaboration :

Elle permet de préciser la plupart des cas d'utilisation et de concevoir l'architecture du système. L'architecture doit être exprimée sous forme de vue de chacun des modèles. Lors de cette

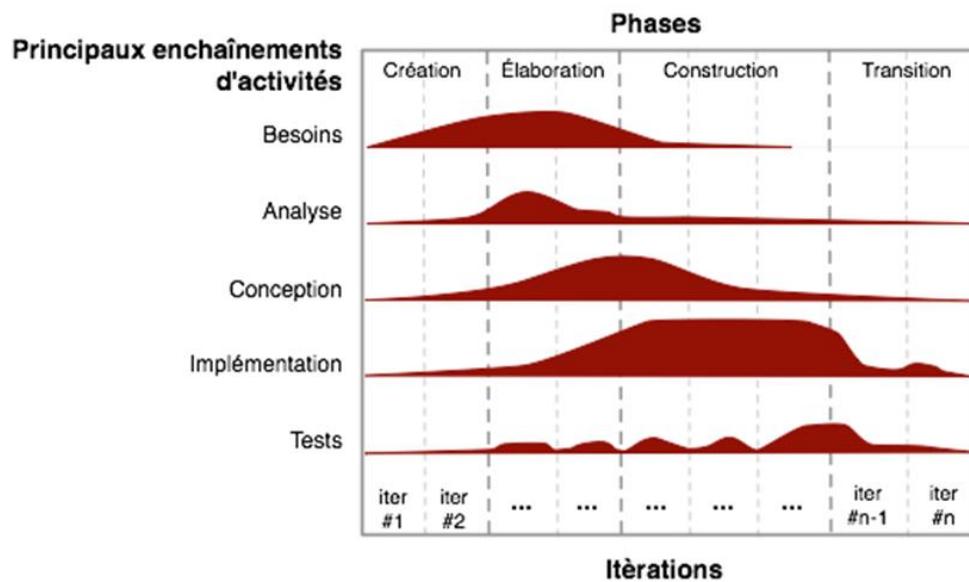
phase une architecture de référence sera conçue. Au terme de cette étape, le chef de projet doit être en mesure de prévoir les activités et d'estimer les ressources nécessaires à l'achèvement du projet.

- **Construction :**

C'est le moment où l'on construit le produit. L'architecture de référence se métamorphose en produit complet, elle est maintenant stable. Le produit contient tous les cas d'utilisation que les chefs de projet, en accord avec les utilisateurs ont décidé de mettre au point pour cette version. Celle-ci doit encore avoir des anomalies qui peuvent être en partie résolue lors de la phase de transition.

- **Transition :**

Le produit est en version bêta. Un groupe d'utilisateurs essaye le produit et détecte les anomalies et défauts. Cette phase suppose des activités comme la fabrication, la formation des utilisateurs clients, la mise en œuvre d'un service d'assistance et la correction des anomalies constatées (ou le report de leur correction à la version suivante).



**Figure 1. 2.** Les phases d'UP.

Bien entendu, la philosophie du Processus Unifié est tout à fait compatible avec la pratique des méthodes Agiles. Le UP est cependant plus complet que la plupart des autres méthodes Agiles, comme XP ou Scrum. Par exemple ces dernières n'abordent pas ou très peu la notion de discipline [3]. Pour certains types de logiciels, tels que les systèmes critiques, où une analyse complète du système est essentielle, l'approche basée modélisation (plan-driven

approach) tel que UP est l'approche la plus appropriée [5] Toutefois, dans un environnement dynamique et changeant, dans le cas de petit/moyen projet et équipes de développement (comme dans le contexte des applications mobiles), cela peut causer des problèmes réels. En effet, l'utilisation des approches basée modélisation dans ce contexte engendre un cout de développement et de maintenance très important. Ce problème a causé l'apparition de nouveaux processus axés sur le développement et la livraison rapides de logiciels. Parmi ces processus on trouve les méthodes Agile.

## 5. Méthodes Agiles

Une méthode Agile est une méthode de développement itérative et incrémentale permettant de concevoir des logiciels en impliquant au maximum le demandeur (client), ce qui permet une grande réactivité à ses demandes. Les méthodes agiles se veulent plus pragmatiques que les méthodes traditionnelles. Elles visent la satisfaction réelle du besoin du client, et non d'un contrat établi préalablement. Ces méthodes sont menées dans un esprit collaboratif avec juste ce qu'il faut de formalisme. La mise au premier plan du facteur humain constitue 1' une des facettes fondamentales et originales des méthodes Agiles [3].

La notion de méthode agile est née à travers un manifeste signé par 17 personnalités (parmi lesquelles Ward Cunningham, l'inventeur du Wiki), créateurs de méthodes ou dirigeants de sociétés.

### 5.1. Les valeurs fondamentales de l'agilité

Le manifeste prône quatre valeurs fondamentales [6] :

#### **Individus et interactions plutôt que processus et outils**

Dans l'optique Agile, l'équipe est bien plus importante que les moyens matériels ou les procédures. La communication au sein de l'équipe ainsi qu'avec le client est une notion essentielle.

**Des fonctionnalités opérationnelles plutôt qu'une documentation exhaustive :**

Il est primordial que l'application fonctionne, le reste, comme la documentation technique, étant secondaire. Une documentation concise reste indispensable, mais une documentation trop détaillée représente une charge de travail importante qui peut devenir contre-productive si elle n'est pas tenue à jour.

**Collaboration avec le client plutôt que contractualisation des relations**

La collaboration avec le client ne se limite pas à la négociation du contrat au début du projet. Le client doit être impliqué dans le processus de développement en fournissant un retour continu sur l'adaptation du logiciel à ses attentes.

**Acceptation du changement plutôt que conformité aux plans**

La planification initiale et la structure du logiciel doivent être flexibles afin de permettre l'évolution de la demande du client tout au long du projet. D'ailleurs, l'utilisation des premières versions du logiciel suscite généralement des demandes d'évolution de la part du client.

## **5.2. Les principes généraux de l'agilité**

Les quatre valeurs citées précédemment se déclinent en douze principes généraux communs à toutes les méthodes Agiles :

- La priorité est de satisfaire le client en livrant tôt et régulièrement des logiciels de qualité.
- Le changement est bienvenu, même tardivement dans le développement. Un processus Agile considère le changement comme un avantage compétitif pour le client.
- Livrer fréquemment des versions fonctionnelles, toutes les deux semaines à deux mois. Avec une préférence pour les périodicités courtes.
- Assurer une coopération quotidiennement entre les clients et les développeurs.
- Construire le projet autour d'individus motivés, leur donner l'environnement et le soutien dont ils ont besoin, leur faire confiance.

- Une conversation en tête-à-tête constitue la méthode la plus efficace pour transmettre l'information à et au sein d'une équipe de développement.
- Les fonctionnalités effectives d'un logiciel constituent la meilleure unité de mesure de l'avancement du projet.
- Faire avancer le projet à un rythme raisonnable, qui doit donc pouvoir être théoriquement maintenu indéfiniment.
- Porter une attention continue à l'excellence technique et à la qualité de conception.
- La simplicité est essentielle.
- Les meilleures architectures, spécifications et conceptions sont issues d'équipes qui s'auto-organisent.
- À intervalle régulier, l'équipe réfléchit aux moyens de devenir plus efficace, puis ajuste son comportement et ses procédures en conséquence.

Parmi les méthodes Agiles les plus utilisées actuellement XP. La section suivante détaille cette dernière méthode.

### **6. eXtreme Programming (XP)**

eXtreme Programming (XP) est une méthodologie légère de développement itératif mise au point à la fin des années 90 par trois des signataires du Manifeste Agile, Kent Beck, Ward Cunningham et Ron Jeffries. XP doit son nom au fait qu'elle place l'activité de programmation au centre du projet, et qu'elle obtient ses résultats en combinant et en poussant à l'extrême les meilleures pratiques de développement. XP est adaptée aux projets mettant en œuvre des équipes de taille moyenne, constituées d'une dizaine de personnes au maximum.

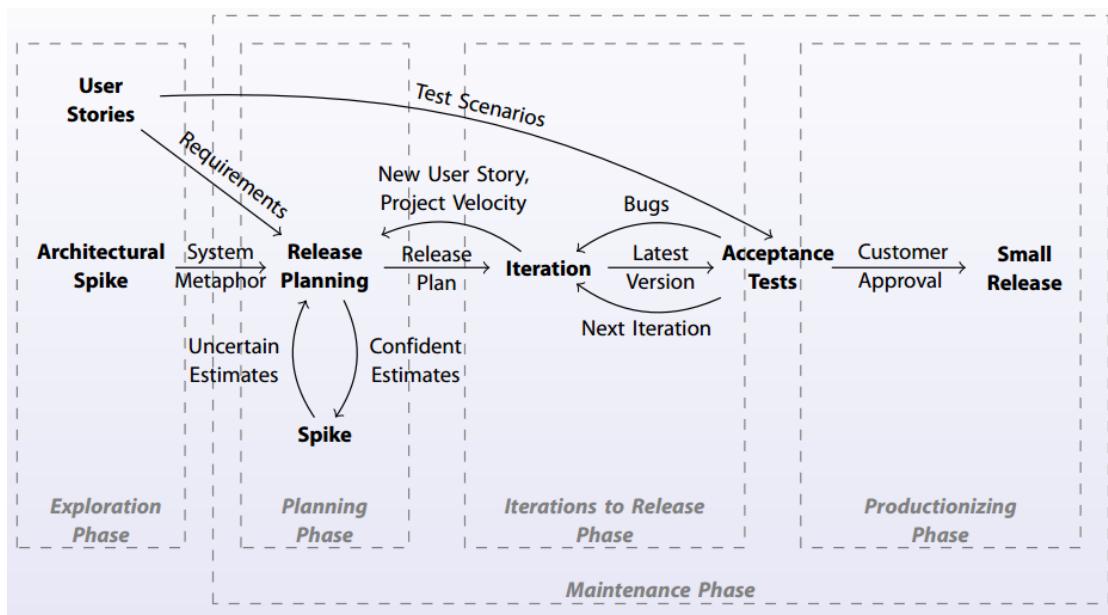
#### **6.1. Cycle XP**

L'eXtreme Programming repose sur des cycles rapides de développement (des itérations de quelques semaines) dont les étapes sont les suivantes :

- Une phase d'exploration qui détermine les scénarios clients (User Stories) qui seront fournis pendant cette itération.
- Une phase de planning où l'équipe transforme les scénarios en tâches à réaliser et en tests fonctionnels.

- Une phase de mise en production où chaque développeur s'attribue des tâches et les réalise avec un binôme.
- Une phase de maintenance lorsque tous les tests fonctionnels passent avec succès, le produit est livré.

Le cycle se répète tant que le client peut fournir des scénarios à livrer. La première livraison est généralement plus importante et n'est réalisée qu'après quelques itérations. Après la première mise en production, les itérations peuvent devenir plus courtes (une semaine par exemple).



**Figure 1. 3.** Cycle de XP [7].

## 6.2. Les Valeurs de XP

L'EXtreme Programming s'appuie sur les valeurs suivantes [8] :

- **La communication** : c'est le moyen fondamental d'éviter les erreurs. Le moyen à privilégier est la conversation directe, en face à face. Les moyens écrits ne sont que des supports et des moyens de mémorisation.
- **Le courage** : il est nécessaire à tous les niveaux et de la part de tous les intervenants, notamment chez les développeurs (quand des changements surviennent à un stade avancé du projet, ou quand des défauts apparaissent) et chez le client (qui doit pouvoir prioriser ses demandes).

- **Le retour d'information (feedback)** : les itérations sont basées sur les retours d'informations du client, permettant une adéquation totale entre l'application finale et sa demande.
- **La simplicité** : XP érige la simplicité en véritable discipline, tant au niveau de la conception que du processus lui-même.

### 6.3. Les pratiques d'XP

Les quatre valeurs de la section précédente se déclinent en treize pratiques qui se renforcent mutuellement [3]. Ces pratiques ont déjà des racines bien profondes et anciennes en génie logiciel que l'XP vient juste pour les confirmés et les mettre bien en valeur, en exposant tout ce qui relatif avec ces pratiques.

- **Client sur site** : Un représentant des clients, présent à plein temps pendant toute la durée du projet, permet une communication intensive et permanente avec les développeurs, tant pour l'expression des besoins que pour la validation des livraisons. Il s'agit là de l'une des exigences d'XP les plus complexes à réaliser dans la pratique.
- **Jeu du Planning** : Le client exprime ses besoins au travers de scénarios utilisateur décrivant les fonctionnalités qu'il souhaite obtenir. L'équipe évalue le temps nécessaire pour les implémenter puis, client et développeurs s'accordent sur les scénarios à réaliser durant l'itération.
- **Tests de recette (ou tests fonctionnels)** : L'équipe crée des procédures de tests associées à chaque scénario pour mesurer l'avancement du développement. Une itération est terminée lorsque tous les tests fonctionnels sont passés avec succès.
- **Tests unitaires** : Un test unitaire permet de vérifier qu'un module de code fonctionne correctement. Pour s'assurer que les tests unitaires ne soient jamais négligés, XP impose leur écriture avant le code qu'ils doivent tester. Ces tests sont conservés jusqu'à la fin du projet. À chaque modification du code, ils sont exécutés pour se prémunir de toute régression.
- **Intégration continue** : L'intégration continue des incrément au produit complet permet de ne pas repousser à la fin du projet le risque majeur de l'intégration simultanée des modules développés par des équipes différentes.
- **Petites livraisons** : Partant du constat que ce sont les clients qui sont les plus aptes à juger de la qualité du produit, les concepteurs d'XP prônent un processus basé sur des itérations courtes conduisant à de petites livraisons fréquentes. Dans un premier temps, le produit livré ressemble à

un squelette vide qui s'enrichit au fur et à mesure des itérations.

- **Rythme soutenable :** En favorisant une élimination progressive des risques, XP vise à se prémunir des coups de bourre et à établir un rythme de travail régulier sans heures supplémentaires. L'objectif est de réduire stress et tensions qui sont nuisibles à la performance de l'équipe.
- **Conception simple :** XP bannit tout investissement en matière de développement destiné à anticiper d'éventuelles évolutions. Le développeur doit se focaliser sur la solution la plus simple qui permette de passer les tests fonctionnels. Plus l'application est simple, plus il sera facile de la faire évoluer lors des prochaines itérations
- **User des métaphores :** Les métaphores et les analogies sont aussi bien utiles pour la communication avec le client que pour le partage des schémas intellectuels au sein de l'équipe.
- **Amélioration du code par la réécriture (Refactoring) :** Comme le code doit être le plus simple possible, il est inévitable de devoir procéder à des réécritures pour prendre en compte de nouvelles fonctionnalités. L'impact de cette réécriture est maîtrisé grâce à l'existence des tests automatiques de non régression. XP est favorable aux techniques de réécriture visant une amélioration progressive et régulière de la qualité du code sans en modifier le comportement.
- **Appropriation collective du code :** L'équipe est collectivement responsable du fonctionnement ou du dysfonctionnement de l'application. Chaque développeur est en droit d'apporter des modifications dans toutes les portions du code, personne n'est propriétaire d'une partie de l'application. Encore une fois, les tests permettent de se prémunir des régressions. Aucun développeur ne doit devenir indispensable au projet. La programmation en binôme va également dans ce sens.
- **Convention de codage :** Puisque tous les développeurs interviennent sur tout le code, XP préconise la définition de conventions de codage en début de projet.
- **Programmation en binôme :** La programmation en binôme est l'arme secrète d'XP. Elle se fait à quatre mains, avec un seul poste de travail pour deux personnes. Le premier appelé pilote (driver), est au clavier et travaille sur la portion de code à écrire. Le second, appelé copilote (partner), est là pour l'aider en suggérant de nouvelles possibilités ou en décelant d'éventuels problèmes. Cette pratique favorise l'échanger d'idées et la correction mutuelle. La qualité des programmes et des solutions atteint ainsi des niveaux très élevés tout en diminuant les délais. D'un point de vue

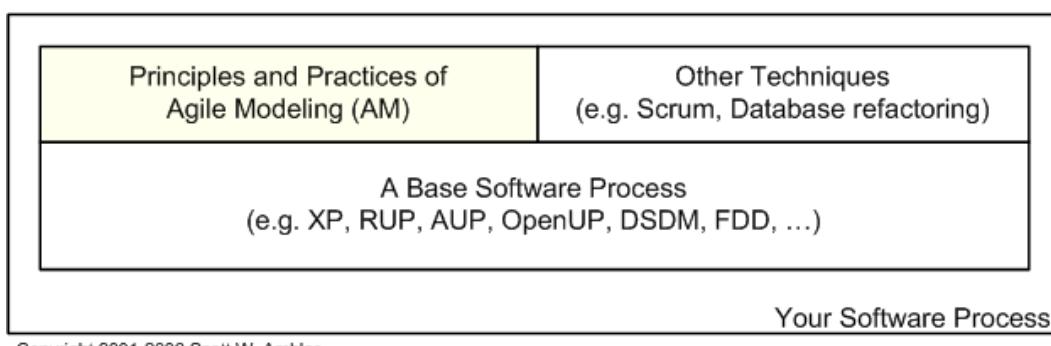
budgétaire, le surcoût engendré par une programmation en binôme est largement compensé par le gain obtenu lors de la correction d'erreurs après mise en production. Pour favoriser la connaissance collective de l'application et améliorer la communication au sein de l'équipe, les développeurs doivent régulièrement changer de partenaire. Enfin, le travail de concert permet de limiter le temps consommé par des activités individuelles non productives (consultation du courrier électronique, d'Internet, communications téléphoniques personnelles, etc.).

Dans la section suivante nous allons présenter une approche qui a essayé de réconcilier les deux courants précédents (Approche basée modélisation, Méthodes Agile). Cette approche est nommée par AM (Agile Modeling). Dans AM on suggère qu'un modèle bien fait favorise et facilite la communication au sein de l'équipe de développement d'une part et avec le client d'autre part (la communication est un aspect important dans les méthodes Agile) [9]. Donc, l'inclusion d'une modélisation bien faite dans les méthodes Agile ne peut être que bénéfique.

### **7. Agile Modeling (AM)**

Agile Modeling (AM) est une méthodologie pratique pour la modélisation et la documentation efficaces des systèmes logiciels. Autrement dit, Agile Modeling (AM) est une collection de valeurs, de principes et de pratiques pour le logiciel de modélisation qui peut être appliquée sur un projet de développement logiciel de manière efficace et légère.

La figure 1.4, AM est destiné à être adapté à d'autres méthodologies à part entière telles que XP ou RUP, cette méthodologie permettant de développer un processus logiciel qui répond vraiment au besoin [10].



**Figure 1. 4** Portée d'Agile Modeling.

Agile Modeling a trois principaux objectifs [9] :

- Définir et montrer comment mettre en pratique une collection de valeurs, de principes et de pratiques pour une modélisation efficace et légère.
- Décrire comment appliquer les techniques de modélisation sur les équipes de projets logiciels Agiles.
- Pour décrire comment vous pouvez améliorer vos activités de modélisation suite à une approche « Near Agile » du développement de logiciels, en particulier, des équipes de projet après le RUP (Rational Unified Process) ou AUP (Agile Unified Process).

Dans ce qui suite nous allons présenter l'approche de Pascal Roques qui a été choisi pour la modélisation de notre projet, cette approche est conformément à l'esprit de Agile Modeling.

## **8. L'approche de développement de Pascal Roques**

L'approche Pascal Roques est une méthode simple et générique qui se situe à mi-chemin entre UP (Unified Processus), qui constitue un cadre général très complet de processus de développement, et XP (eXtreme Programming) qui est une approche minimaliste à la mode centrée sur le code [3].

Le processus de Rascal Roques a été utilisé à l'origine pour les applications web, mais à notre avis il peut être utilisé les applications mobiles cela est dû à la similitude des caractéristiques des deux types d'applications, aussi la similitude d'environnement (les deux dynamiques et changeant rapidement), dans ce qui suite on cite quelques points forts de ce processus [11] :

- Conduit par les cas d'utilisation, comme UP, mais beaucoup plus simple.
- Relativement léger et restreint, comme les méthodes agiles, mais sans négliger les activités de modélisation en analyse et conception.
- Fondé sur l'utilisation d'un sous-ensemble nécessaire et suffisant du langage UML, conformément à Agile Modeling.

Cette approche est composée des étapes suivantes :

### **8.1. Identification des besoins et spécification des fonctionnalités**

#### **– Identification et représentation des besoins (diagramme de cas d'utilisation)**

Les cas d'utilisation sont utilisés tout au long du projet. Dans un premier temps, on les crée pour identifier et modéliser les besoins des utilisateurs. Ces besoins sont déterminés à partir des informations recueillies lors des rencontres entre informaticiens et utilisateurs.

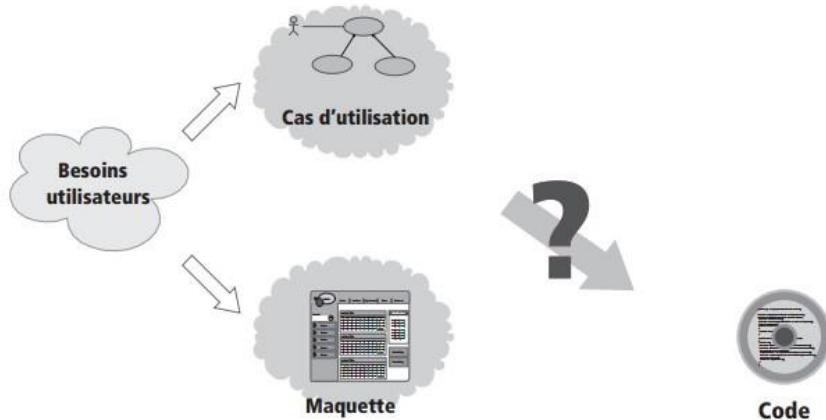
#### **– Spécification détaillée des besoins (Diagrammes de séquence système)**

Dans cette étape, on cherche à détailler la description des besoins par la description textuelle des cas d'utilisation et la production de diagrammes de séquence système illustrant cette description textuelle. Le diagramme de séquence système représente l'interaction entre les acteurs et le système.

Cette étape amène souvent à mettre à jour le diagramme de cas d'utilisation puisque nous sommes toujours dans la spécification des besoins.

#### **– Maquette de l'IHM de l'application (non couvert par UML) :**

Une maquette d'IHM (Interface Homme-Machine) est un produit jetable permettant aux utilisateurs d'avoir une vue concrète, mais non définitive de la future interface de l'application. La maquette peut très bien consister en un ensemble de dessins produits par un logiciel de présentation ou de dessin. Par la suite, la maquette pourra intégrer des fonctionnalités de navigation permettant à l'utilisateur de tester l'enchaînement des écrans ou des menus, même si les fonctionnalités restent fictives. La maquette doit être développée rapidement afin de provoquer des retours de la part des utilisateurs.



**Figure 1. 5.** Les besoins donnent lieu à des cas d'utilisation et à une maquette [11].

## 8.2. Phases d'analyse

L'analyse permet une formalisation du système à développer en réponse à l'expression des besoins formulée par l'utilisateur. L'analyse est concrétisée par l'élaboration de tous les diagrammes donnant une représentation du système tant statique (diagramme de classes principalement), que dynamique (diagramme de séquence, d'activité).

### – Analyse du domaine (modèle du domaine)

La phase d'analyse du domaine permet d'élaborer la première version du diagramme de classes, les classes du modèle du domaine ne doivent pas contenir d'opération, mais seulement des attributs. Les étapes à suivre pour établir ce diagramme sont :

- Identifier les entités ou concepts du domaine.
- Identifier et ajouter les associations et les attributs.
- Organiser et simplifier le modèle en éliminant les classes redondantes et en utilisant l'héritage.
- Le cas échéant, structurer les classes en paquetage selon les principes de cohérence et d'indépendance.

### – Diagramme de classes participantes :

Le diagramme de classes participantes est particulièrement important puisqu'il effectue la jonction entre, d'une part les cas d'utilisation le modèle du domaine et la maquette et d'autre part, les diagrammes de conception logicielle que sont les diagrammes d'interaction et le diagramme de

classes de conception. Le diagramme de classes participantes modélise trois types de classes d'analyse :

- **Les classes de dialogues :**

Les classes qui permettent les interactions entre l'IHM et les utilisateurs sont qualifiées de dialogues. Ces classes sont directement issues de l'analyse de la maquette présentée. Il y a au moins un dialogue pour chaque association entre un acteur et un cas d'utilisation du diagramme de cas d'utilisation. En général, les dialogues vivent seulement le temps du déroulement du cas d'utilisation concerné.

- **Les classes de contrôles :**

Les classes qui modélisent la cinématique de l'application sont appelées contrôles. Elles font la jonction entre les dialogues et les classes métier en permettant aux différentes vues de l'application de manipuler des informations détenues par un ou plusieurs objets métier. Elles contiennent les règles applicatives et les isolent à la fois des dialogues et des entités.

- **Les classes entités :**

Les classes métier, qui proviennent directement du modèle du domaine, sont qualifiées d'entités. Ces classes sont généralement persistantes, c'est -à-dire qu'elles survivent à l'exécution d'un cas d'utilisation particulier et qu'elles permettent à des données et des relations d'être stockées dans des fichiers ou des bases de données.

- **Diagramme d'activité de navigation**

Les IHM modernes facilitent la communication entre l'application et l'utilisateur en offrant toute une gamme de moyens d'action et de visualisation comme des menus déroulants ou contextuels, des palettes d'outils, des boîtes de dialogues, des fenêtres de visualisation, etc. Cette combinaison possible d'options d'affichage, d'interaction et de navigation aboutit aujourd'hui à des interfaces de plus en plus riches et puissantes.

Les diagrammes d'activités de navigation sont à relier aux classes de dialogue du diagramme de classes participantes. Les différentes activités du diagramme de navigation peuvent être stéréotypées en fonction de leur nature : « Dialogue », « Contrôle », « entité ».

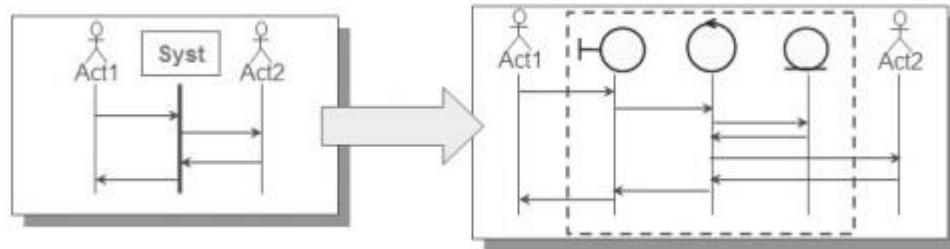
### **8.3. Phase de conception**

#### **– Diagramme d'interaction (diagramme de séquence détaillé)**

Par rapport aux diagrammes de séquence système du chapitre 4, nous allons remplacer le système vu comme une boîte noire par un ensemble d'objets en interaction. Pour cela, nous utiliserons encore dans ce chapitre les trois types de classes d'analyse, à savoir les dialogues, les contrôles et les entités. Nous respecterons également les règles que nous avions fixées sur les relations entre classes d'analyse, mais en nous intéressant cette fois-ci aux interactions dynamiques entre objets :

- Les acteurs ne peuvent interagir (envoyer des messages) qu'avec les dialogues.
- Les dialogues peuvent interagir avec les contrôles.
- Les contrôles peuvent interagir avec les dialogues, les entités, ou d'autres contrôles.
- Les entités ne peuvent interagir qu'entre elles.

Le changement de niveau d'abstraction par rapport au diagramme de séquence système peut ainsi se représenter comme sur la figure 1.6 .



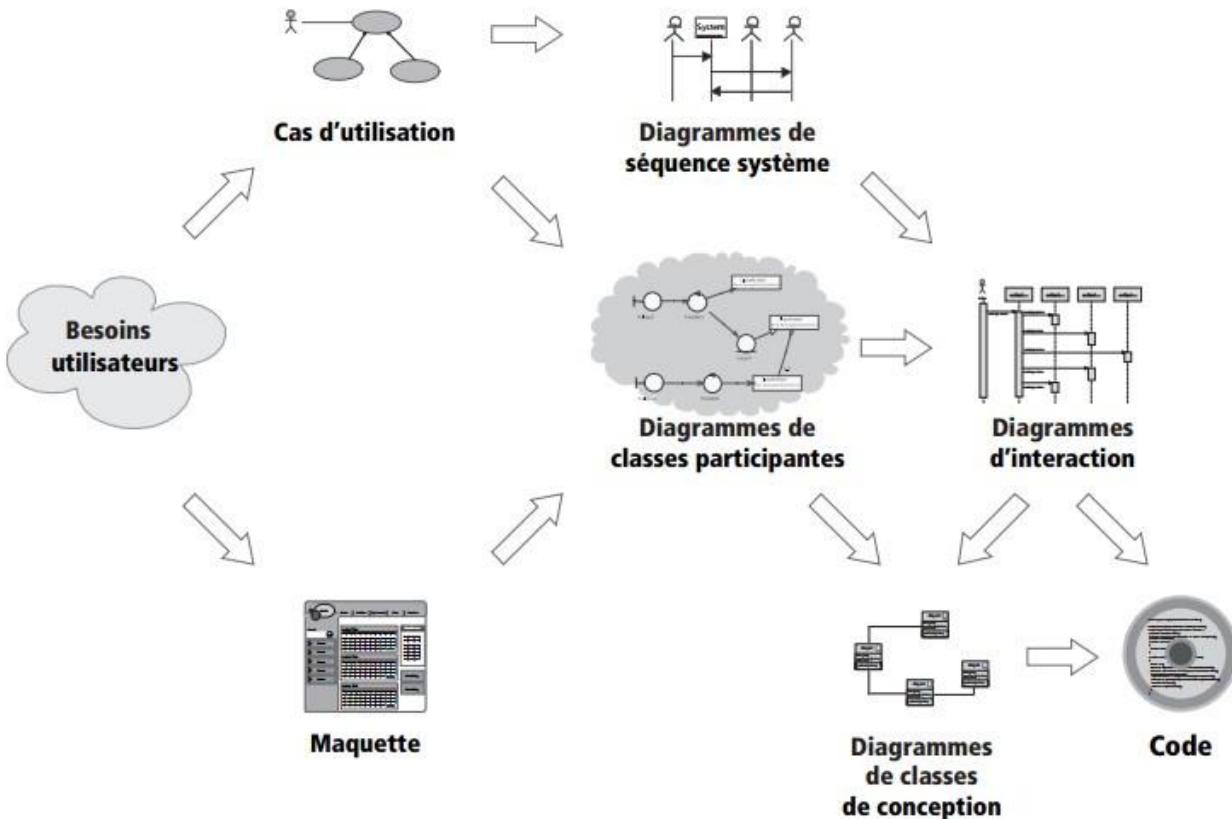
**Figure 1. 6.** Passage de l'analyse à la conception préliminaire [11].

#### **– Diagramme de classes de conception :**

L'objectif de cette étape est de produire le diagramme de classes qui servira pour l'implémentation. Une première ébauche du diagramme de classes de conception a déjà été élaborée en parallèle du diagramme d'interaction. Il faut maintenant le compléter en précisant [3] :

- Les attributs et les opérations privées nécessaires aux différentes classes ;
- Le type des attributs, des opérations et de leurs éventuels paramètres ;

- La navigabilité des associations qui n'ont pas de raison d'être bidirectionnelles (ce qui devrait être le cas de la grande majorité des associations).



**Figure 1. 7.** Chaîne complète de la démarche de modélisation sous L'approche Pascal Roques [11].

## 9. Conclusion

Dans ce chapitre nous avons présenté l'approche de développement de Pascal Roque que nous allons suivre pour développer notre application mobile. Pour bien comprendre les ingrédients qui la composent nous avons également présenté quelques méthodologies de développement à savoir UP, les méthodes Agile et Agile Modeling. Cette dernière rapproche les deux premières et constitue une voie prometteuse pour le développement logiciels en général et des applications mobiles en particulier.

## **CHAPITRE 02**

---

**Introduction aux applications et S.E Android**

---

## 1. Introduction

Le téléphone portable est devenu un équipement indispensable dans la vie humaine et ce grâce à la nouvelle technologie de smartphone. Le téléphone intelligent ou smartphone a remplacé plusieurs équipements qui ont été utilisés dans la vie quotidienne comme : le PC, Télévision, Radio etc., ce qui a poussé les fournisseurs des services et des applications à développer des nouveaux systèmes et programmes pour ces nouveaux équipements.

Dans ce chapitre nous allons présenter le système d'exploitation des appareils mobile Android, les applications le concernant et comment ils fonctionnent. Enfin nous allons aussi expliquer comment développer des applications sous Android.

## 2. Application mobile

Les applications mobiles sont apparues dans les années 1990, elles sont liées aux développements d'Internet et des télécommunications, des réseaux sans fil, et à l'apparition et la démocratisation des terminaux mobiles : smartphones, tablettes tactiles etc.

Une application mobile est un logiciel applicatif développé pour un appareil électronique mobile, tel qu'un assistant personnel, un téléphone portable, un « smartphone », un baladeur numérique, une tablette tactile, ou encore certains ordinateurs fonctionnant avec le système d'exploitation Windows Phone [12].

Une application mobile est un programme téléchargeable de façon gratuite ou payante et exécutable à partir du système d'exploitation du téléphone [13].

Les applications mobiles sont adaptées aux différents environnements techniques des smartphones et à leurs contraintes et possibilités ergonomiques (écran tactile notamment). Elles permettent généralement un accès plus confortable et plus efficace à des sites accessibles par ailleurs en versions mobile ou web. L'application mobile peut avoir une vocation commerciale (m-commerce), marketing et / ou publicitaire [13], Elle peut aussi offrir un service public.

Il y a plusieurs systèmes d'exploitation orientés mobile et les plus connus sont : Android, iOS, BlackBerry OS, Microsoft Windows Phone. Dans la suite nous présentons le système Android qui est le plus utilisé.

### 3. Le système d'exploitation Android

Android est un système d'exploitation mobile open source, basé sur le noyau Linux et développé actuellement par Google. Lancé en juin 2007 à la suite du rachat par Google en 2005 de la startup du même nom. Le système avait d'abord été conçu pour les smartphones et tablettes tactiles, puis s'est diversifié dans les objets connectés comme les télévisions (Android TV), les voitures (Android Auto), les ordinateurs (Android-x86) et les smartwatch (Android Wear).



Android propose une approche unifiée pour le développement d'applications pour les appareils mobiles, ce qui signifie que les développeurs doivent développer uniquement pour Android, et leurs applications devraient pouvoir fonctionner sur différents appareils alimentés par Android [14].

Google lance une nouvelle version d'Android chaque année avec de nouvelles fonctionnalités et améliorations. Les différentes versions d'Android ont toutes des noms de desserts ou plus généralement des sucreries depuis la sortie de la version 1.5 et suivent une logique alphabétique (la version la plus récente est Android 7.1 dénommée "Nougat").

#### 3.1. Les points forts du Système d'exploitation Android

Android offre plusieurs avantages dont :

- **Open-source :**

Le contrat de licence pour Android respecte l'idéologie open-source [15], c'est-à-dire que vous pouvez à tout moment télécharger les sources et les modifier selon vos goûts.

- **Gratuit (ou presque)**

Android est gratuit, autant pour les utilisateurs, que pour les constructeurs [15]. Si les utilisateurs prenaient l'envie de produire son propre téléphone sous Android, alors ils n'auraient même pas à ouvrir ses portes monnaie. En revanche, pour poster leurs applications sur le Play Store, ils en coûteront la modique somme de 25\$. Ces 25\$ permettent de publier autant d'applications qu'ils le souhaitent, à vie.

- **Facile à développer**

Toutes les API mises à disposition facilitent et accélèrent grandement le travail [14]. Ces APIs sont très complètes et très faciles d'accès.

- **Facile à vendre**

Le Play Store (anciennement Android Market) est une plateforme immense et très visitée; c'est donc une mine d'opportunités pour quiconque possédant une idée originale ou utile.

- **Flexible**

Le système est extrêmement portable, il s'adapte à beaucoup de structures différentes. Les smartphones, les tablettes, la présence ou l'absence de clavier ou de trackball, différents processeurs... On trouve même des micro-ondes qui fonctionnent à l'aide d'Android.

- **Ingénieux**

L'architecture d'Android est inspirée par les applications composites, et encourage par ailleurs leur développement. Ces applications se trouvent essentiellement sur internet et leur principe est que vous pouvez combiner plusieurs composants totalement différents pour obtenir un résultat surpuissant. Par exemple, si on combine l'appareil photo avec le GPS, on peut poster les coordonnées GPS des photos prises.

### 3.2. L'architecture d'Android

Android est défini comme étant une pile de logiciels, c'est-à-dire un ensemble de logiciels destinés à fournir une solution clé en main pour les appareils mobiles – smartphones et tablettes tactiles l'ensemble est organisé en cinq couches distinctes [15] :

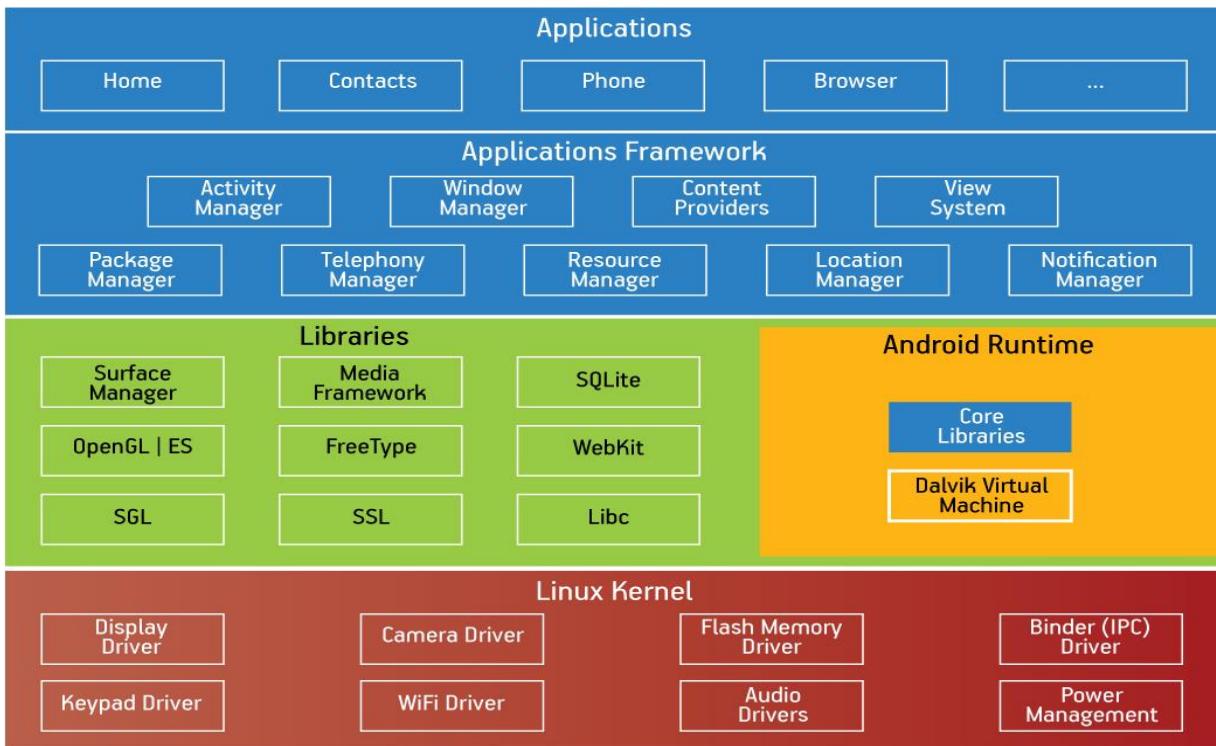


Figure 2. 1. Les composants d'architecture d'Android OS [16].

- **Un noyau Linux** qui contient les pilotes pour les interactions avec le matériel (radios, camera, sensors, etc) et les gestionnaires de mémoire, d'alimentation et de gouverneur CPU [15].
- **Des bibliothèques standard** : en interne, Android inclut un ensemble de bibliothèques C et C++ utilisées par de nombreux composants de la plateforme Android. Ces bibliothèques sont en réalité accessibles au développeur par l'intermédiaire du Framework Android. En effet, le Framework Android effectue de façon interne des appels à des fonctions C/C++ beaucoup plus rapides à exécuter que des méthodes Java standard. La technologie JNI (Java Native Interface) permet d'effectuer des échanges entre le code Java et le code C et C++.
- **Une machine virtuelle Java adaptée (Moteur d'exécution d'Android)** [15] : la Dalvik Virtual Machine (remplacée par ART à partir de la version 5.0, Android RunTime est plus performant, les applications prennent ainsi plus de place (+20 %), mais les gains en performance et en autonomie des batteries sont conséquents (+20 à 30 %).), c'est le responsable de l'exécution des applications écrites pour Android, conçues pour des systèmes contraints en termes de mémoire et de vitesse du processeur.

- **Application Framework** [17] sous la forme de l'API java qui permet aux applications d'exploiter les différentes fonctionnalités comme la gestion de fenêtres, de téléphonie, gestion de contenu, les informations de localisation, exécuter les services d'arrière-plan, définir des alarmes, ajouter des notifications de la barre d'état, etc.
- **Les Application** visible à l'utilisateur contiennent les interfaces graphiques, comme les applications basiques incluent (le navigateur web, une gestion des contacts, un calendrier, etc.) ou les applications écrit par les développeurs tiers (Facebook, twitter, etc.).

## 4. Les applications Android

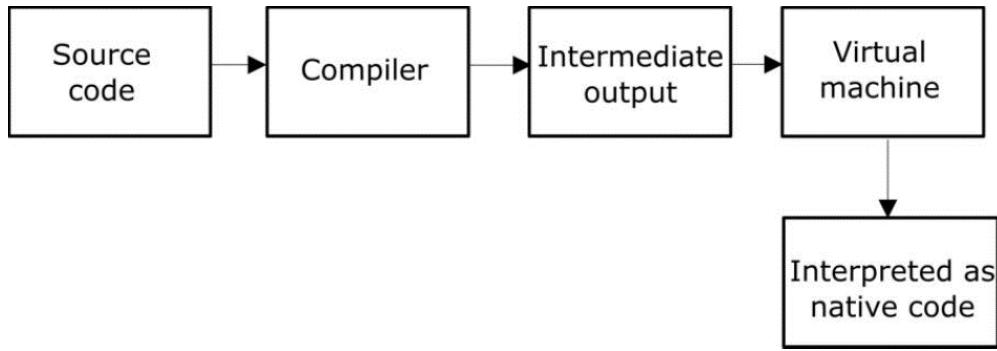
Les applications Android sont généralement développées en langage Java en utilisant **Android Software Development Kit**. Une fois développées, les applications Android peuvent être emballées facilement et vendues soit à travers un store tel que Google Play ou Amazon Appstore [14].

Android gère des centaines de millions d'appareils mobiles dans plus de 190 pays à travers le monde. C'est la plus grande base installée de toute plate-forme mobile et se développe rapidement. Chaque jour, plus d'un million de nouveaux appareils Android sont activés dans le monde entier.

### 4.1. Fonctionnement des applications Android

Certaines applications comme les applications JAVA besoin une machine virtuelle pour s'exécutent. Si la machine virtuelle est en marche alors le logiciel d'application s'exécute sur l'ordinateur indépendamment du matériel et du système d'exploitation, L'avantage évident de développer des applications qui s'exécutent sur des machines virtuelles peut alors être déclaré comme suit : "développer une fois et courir sur toutes les plates-formes". Toutefois, les applications exécutées sur des machines virtuelles sont plus lentes que les autres applications.

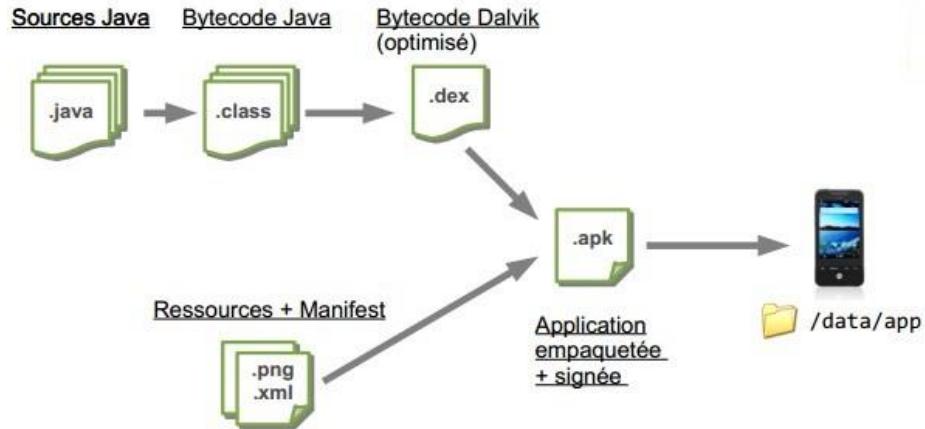
Le processus de développement général des applications de machines virtuelles est résumé à La Figure 2.2.



**Figure 2. 2.** La création d'un code intermédiaire à partir du code source.

Comme pour les applications Java, les applications Android fonctionnent également sur une machine virtuelle. Il existe deux machines virtuelles spéciales utilisées dans Android : Dalvik Virtual Machine (DVM) et Android RunTime (ART). Il s'agit de JVM spécialisées qui peut fonctionner sur des ressources système faibles [18]. Les fichiers APK (exécutables d'applications Android) sont effectivement exécutés sur ces machines virtuelles. DVM a été l'environnement d'exécution par défaut (machine virtuelle) jusqu'à la version de Lollipop (Android 5.0).

ART est introduit par Android 4.4 et a été la VM par défaut à partir d'Android 5.0 [18]. DVM et ART font essentiellement le même travail : exécuter des applications Android indépendamment de la plate-forme (Figure 2.3). Le principal avantage de ART sur DVM est l'utilisation d'un concept appelé la compilation « Ahead of Time » (la compilation anticipée) au lieu de compilation « Just in Time » (la compilation à la volée) [18]. Dans AOT, les applications sont compilées lors de l'installation, donc elles chargent plus rapidement avec une utilisation CPU plus faible. D'autre part, la compilation JIT offre une consommation d'espace de stockage plus faible avec des temps de chargement relativement plus longs [18].

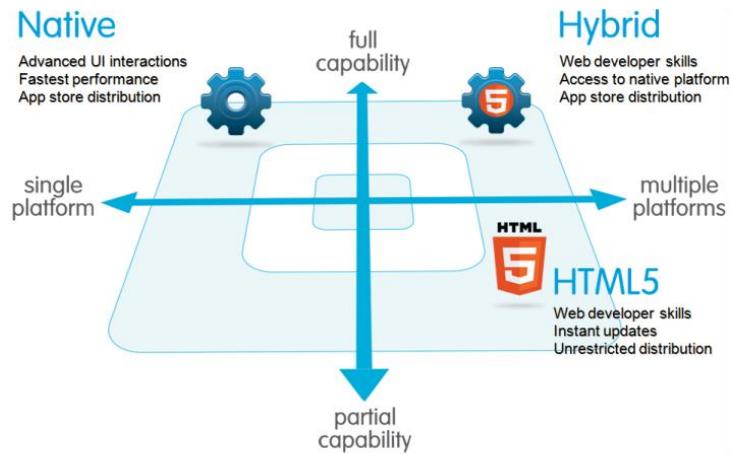


**Figure 2. 3.** Compilation et déploiement d'une application mobile.

Il y a plusieurs approches pour développer des applications Android. Nous les présentons dans la suite.

## 5. Les approches de réalisation des applications Android

Pour développer une application Android on trouve trois différentes approches, chaque approche a ses avantages et inconvénients.



**Figure 2. 4.** Les approches de réalisation d'une application mobile [19].

### 5.1. L'approche Application Web

Une application mobile web n'est rien d'autre qu'un site internet conçu pour être accessible et affichable correctement sur les terminaux mobiles [20].

**Avantages :**

- Accessibles depuis n'importe quel terminal mobile.
- Simples à développer.
- Compatibilité multiplateforme.
- Mise à jour simple et instantanée.

**Inconvénients :**

- Moins de fonctionnalités et de performances.
- Inaccessible hors ligne.
- Esthétique moins agréable et pas toujours ergonomique.

### 5.2. L'approche Native

Cette approche est supportée officiellement par Google et elle s'appuie sur le langage JAVA et le Android SDK (software development kit), cette approche est la meilleure pour la performance et l'exploitation des capacités complètes du plateau [21].

**Avantages :**

- Meilleures performances et un accès avancé au hardware.
- Une expérience entièrement offline pour répondre aux exigences des utilisateurs.
- Une expérience riche, intuitive et inégalée en termes de fonctionnalité, résolution et de qualité.

**Inconvénients :**

- Plus complexe par rapport aux autres types d'applications.
- Le coût de la maintenance et de la mise à jour plus élevé.

### 5.3. Approche Hybrid

Une application hybride combine des éléments HTML5 sous forme de web application et des éléments d'une application native. Ceux-ci permettent d'utiliser les fonctionnalités natives des smartphones [22].

**Avantages :**

- Un coût et un délai de développement réduits.
- Un accès à toutes les fonctions présentes sur le mobile.
- Facilité de maintenance.

### Inconvénients :

- Moins de performances que l'approche native.
- Pas à jour avec les dernières APIs.
- Outils de développement sont encore relativement immatures.
- Vous avez besoin d'une plateforme tierce (comme Cordova , Ionic , Onsen UI, pas toujours gratuit).

L'approche	Native	HTML5	Hybrid
<b>Fonctionnalité de l'application</b>			
Graphique	Native APIs	HTML, Canvas, SVG	HTML, Canvas, SVG
Performance	Rapide	Lent	Lent
Aspect et sensation	Native	Emulé	Emulé
Distribution	PlayStore	Web	PlayStore
<b>Accès aux périphériques</b>			
Camera	Oui	Non	Oui
Notifications	Oui	Non	Oui
Contacts, Calendrier	Oui	Non	Oui
Stockage hors ligne	Secure file storage	Shared SQL	Secure file system, Shared SQL
Géolocalisation	Oui	Oui	Oui
<b>Connectivité</b>	En ligne et hors ligne	Généralement en ligne	En ligne et hors ligne
<b>Développement</b>	Java, Kotlin	HTML5, CSS, Js	HTML5, CSS, Js

Table 2. 1. Tableau comparatif entre les approches de réalisation d'App Mobile(Android).

Ci-dessous nous allons présenter comment développer une application Android sous l'Approche Native.

## 6. Développement sous l'approche Native

### 6.1. Les contraintes d'environnement :

Pour les développements, il faut garder à l'esprit que les périphériques mobiles ont souvent [23]:

- Une puissance processeur plus faible.
- Une RAM limitée.
- Des capacités de stockage permanent limitées.
- De petits écrans avec de faibles résolutions.
- Des coûts élevés de transfert de données.
- Des taux de transfert plus lents avec une latence élevée.
- Des connexions réseau moins fiables.
- Des batteries à autonomie limitée.

### 6.2. Outils nécessaires pour le développement Native

On présente dans ce qui suit les outils nécessaires pour développer des applications Android sous l'approche Native :

#### – **Android SDK :**

Le kit de développement (SDK) d'Android est un ensemble complet d'outils de développement. Il inclut un débogueur, des bibliothèques logicielles, un émulateur basé sur QEMU (un logiciel libre de machine virtuelle, pouvant émuler un processeur et, plus généralement, une architecture différente si besoin), de la documentation, des exemples de code et des tutoriaux [24].

L'IDE officiellement supporté était Eclipse combiné au plugin d'outils de développement d'Android (ADT), mais depuis 2015, Google officialise Android Studio qui devient alors l'IDE officiel pour le SDK Android.

#### – **Android Studio :**

Android Studio est un environnement de développement pour développer des applications Android. Il est basé sur IntelliJ IDEA (IDE Java commercial développé par JetBrains) et permet principalement d'éditer les fichiers Java et les fichiers de configuration d'une application Android

et propose entre autres des outils pour gérer le développement d'applications multilingues et permet de visualiser la mise en page des écrans sur des écrans de résolutions variées simultanément [15].

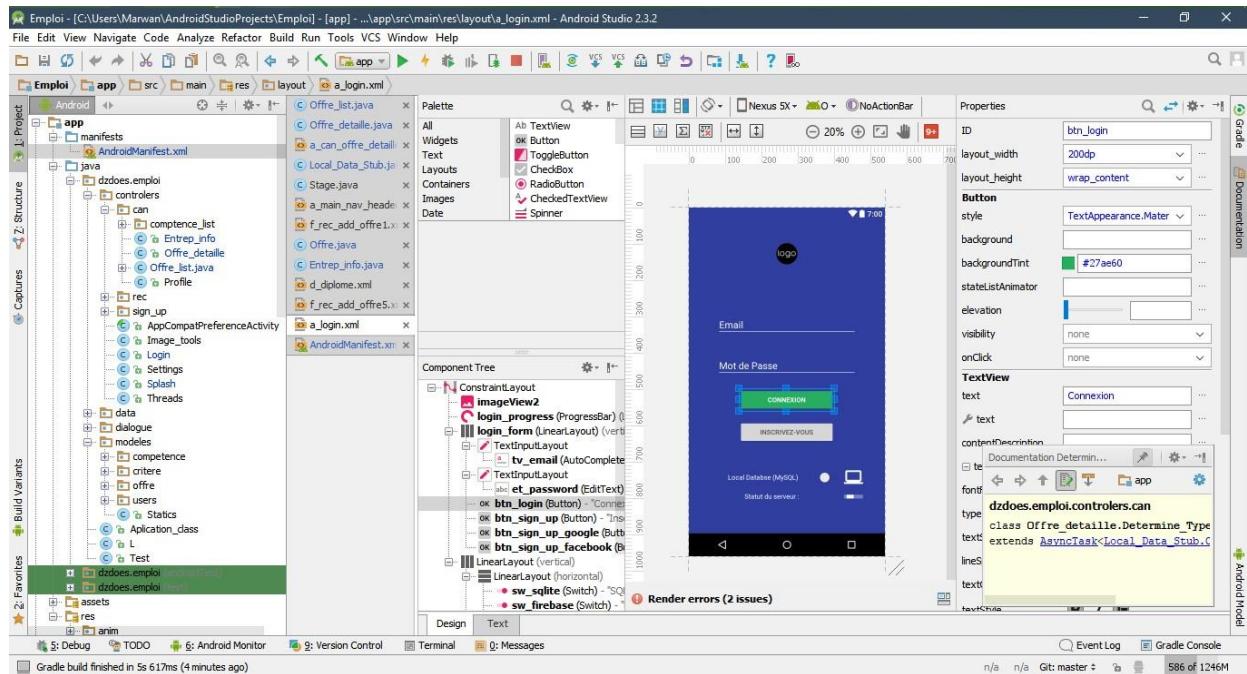


Figure 2. 5. Interface d'IDE Android Studio.

### Emulateur Android

Le SDK Android inclut un émulateur de périphérique mobile virtuel qui s'exécute sur votre ordinateur. L'émulateur vous permet de développer et de tester des applications Android sans utiliser un périphérique physique [15].



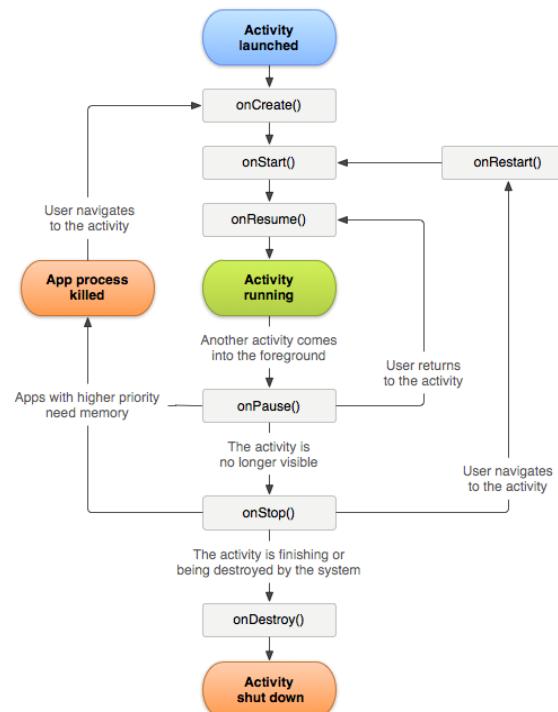
Figure 2. 6. Interface d'Emulateur Android.

### 6.3. Les composants d'une application Android Native

Une application Android peut être construit en utilisant les éléments suivants :

- **Activity :**

Une Activity représente la fenêtre ou tout simplement l'écran qui sera affiché à l'utilisateur. Elle permet également de gérer des fonctionnalités telles que l'appui sur la touche [MENU] ou l'affichage de messages d'alerte (Toast) etc.



**Figure 2. 7.** Cycle de vie d'Activity [16].

- **Fragment :**

Un fragment est un sous-contrôleur d'une activité, contenant une vue ainsi que son propre cycle de vie (voir figure 2.8.), capable d'être ajouté/retiré/déplacé d'une activité [25]. Le principal avantage d'un fragment est le fait d'être facilement déplaçable. Le développeur a donc tout intérêt à mettre la logique d'une vue dans un fragment, au cas où il voudrait déplacer cette vue, ou bien la réutiliser sur un autre écran (exemple : afficher une liste d'utilisateurs).

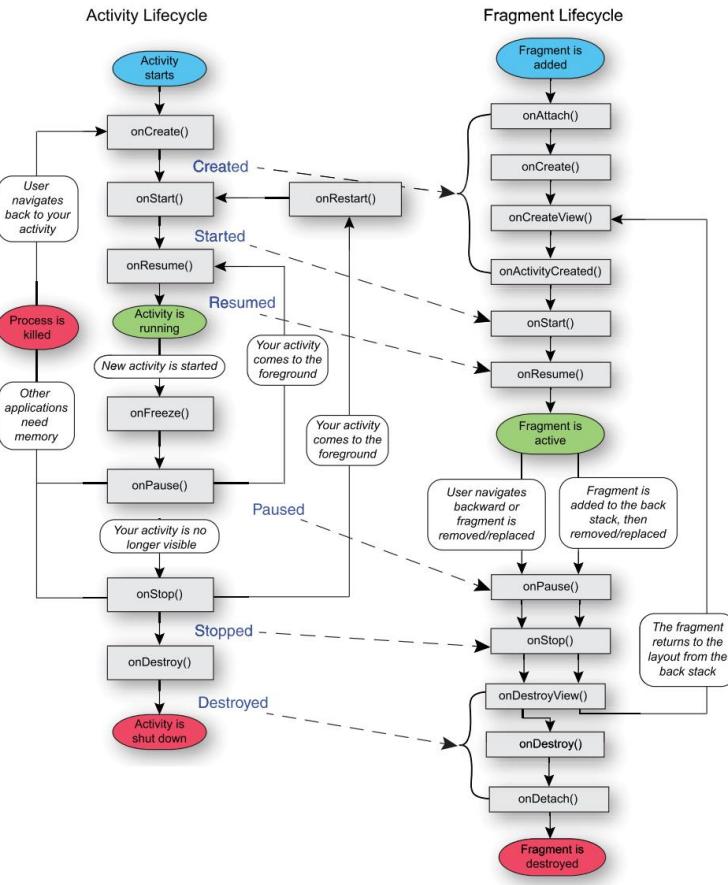


Figure 2. 8. Cycle de vie de Fragment (avec une comparaison avec la Cycle d'Activity) [16].

Le tableau suivant présente la différence entre l' Activity et le Fragment :

	Activity	Fragment
<b>Doit être déclaré dans le Manifest</b>	Oui	Non
<b>Possède un cycle de vie</b>	Oui	Oui
<b>Peut être utilisé dans une Activity</b>	Oui	Oui
<b>Peut être utilisé dans un Fragment</b>	Oui	Non
<b>Sensible à l'orientation</b>	Oui	Oui
<b>Affiche un Layout</b>	Oui	Oui

Table 2. 2. La différence entre l'Activity et le Fragment.

#### – Service :

Un Service est en fait un programme tournant en tâche de fond et n'ayant pas d'interface graphique. L'exemple commun illustrant cette notion est celui du lecteur MP3. Un lecteur MP3 ne nécessite pas pour la plupart du temps d'interface graphique et doit tourner en tâche de fond, laissant

la possibilité aux autres applications de s'exécuter librement. Un service peut être lancé à différents moments [23] :

- Au démarrage du téléphone.
- Au moment d'un événement (arrivée d'un appel, SMS, mail, etc.).
- Lancement de l'application.
- Action particulière dans application.

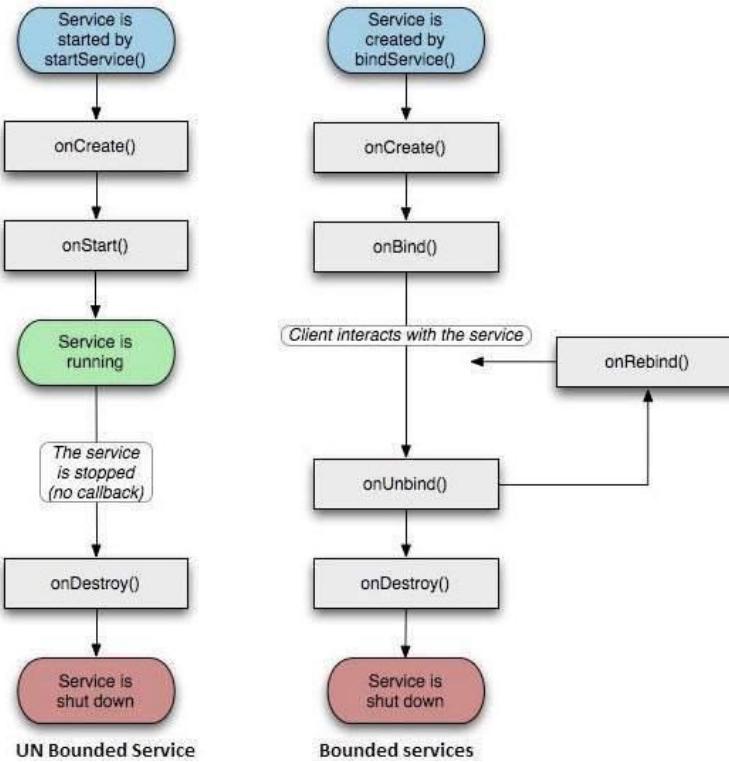


Figure 2. 9.Cycle de vie de Service [16].

#### – Intent :

Les Intents sont des objets permettant de faire passer des messages contenant de l'information entre les principaux composants. La notion d'Intent peut être vue comme une demande de démarrage d'un autre composant ou d'une action à effectuer. La raison d'être des Intents provient du modèle de sécurité d'Android. Chaque application est en effet sandboxée. Cela veut dire qu'une application A ne peut accéder aux données d'une application B. Grâce aux Intents, les applications ont la possibilité de fournir leurs services ou données si elles le souhaitent [23].

**– Broadcast Receivers :**

Un Broadcast Receiver est un composant qui sert à écouter ce qui se passe sur le système ou sur les applications et de déclencher une action pré définie. C'est souvent par ce mécanisme que les services sont lancés.

**– Content Providers:**

Les Content Providers sont, comme l'expriment leurs noms, des gestionnaires de données. Ils permettent de partager l'information entre les applications tel que les contacts sur téléphone, la music, les photos, etc.

## 7. Conclusion

Dans ce chapitre nous avons présenté le système d'exploitation Android et son architecture. Nous avons également présenté le fonctionnement, les approches et les outils utilisés pour la réalisation des applications Android.

## **CHAPITRE 03**

---

**Analyse et conception d'une application mobile pour la recherche d'emplois et de stages**

---

## **1. Introduction**

Pour la réalisation d'un projet logiciel on a besoin de suivre une méthodologie de développement. Cette dernière rend le développement plus facile et sert à documenter notre projet et partager le travail.

Dans ce qui suit, on donne un bref aperçu sur le projet à réaliser, ainsi que les résultats obtenus par l'analyse et la conception par l'application de l'approche du Pascal Roques à notre projet.

## **2. Présentation du notre projet**

Un des problèmes habituels des étudiants en fin de cycle est de trouver 'le stage' qui leurs permettra de valider leurs années et enfin obtenir leurs diplômes. Les jeunes diplômés aussi trouvent des difficultés pour avoir un emploi selon leur compétence. D'ici vient l'idée de créer une application Android pour les aider dans leurs recherches et de leur faciliter la communication avec les entreprises.

- **Les objectifs de projet :**

- Permettre aux étudiants et jeunes diplômés de créer leurs profils, permettant ainsi de valorise les expériences.
- Aider les entreprises à trouver des gens compétents et idéals pour les postes proposés.

- **Le public ciblé :**

- Candidats (étudiants et jeunes diplômés).
- Les recruteurs (représentants des entreprises).
- Administrateur de l'application.

- **Les fonctionnalités de notre application :**

Cette application doit permettre au candidat de faire :

- Une inscription gratuite (mais il doit charger des documents nécessaires pour la validation par le gérant de l'application comme la pièce d'identité).
- Une authentification qui permet à l'utilisateur d'accéder au compte et utiliser les différentes fonctionnalités.
- Un profil qui contient des informations personnelles de l'utilisateur (photo de profil, nom, prénom, Compétences).
- Gérer Profil (modifier la photo de profil, le nom, le prénom et gérer aussi les compétences).

- Gérer les compétences (ajouter, modifier ou supprimer une section).
- Explorer les offres (voir la liste des offres et consulter les informations d'une offre).
- Sauvegarder une offre dans la liste des favoris.
- Consulter la liste des offres favoris.
- Postuler pour une offre.
- Consulter les informations d'une entreprise.

Le recruteur en plus les 2 fonctionnalités : l'inscription et l'authentification, il bénéficie aussi des fonctionnalités suivantes :

- Publier une offre.
- Consulter les demandes sur une offre.
- Accepter une demande.
- Rejeter une demande.
- Consulter la liste de ses offres publiées.
- Modifier les informations concernant l'entreprise.

L'administrateur à la possibilité de faire :

- L'authentification.
- La consultation des demandes d'inscription.
- La confirmation et la validation d'une demande d'inscription.
- Le rejet d'une demande d'inscription.

- **Les besoins non fonctionnels :**

### **Exigences de qualité**

Pour attirer un client sur une application et ensuite le fidéliser, il est important de répondre aux exigences de qualité suivantes :

- L'application doit offrir une interface conviviale et ergonomique exploitable par l'utilisateur.
- **Maintenabilité et évolutivité :** Le code de l'application doit être lisible, compréhensible et bien structuré afin d'assurer son état évolutif et extensible par rapport aux besoins du marché.
- **Contraintes de conception :** Toutes les données personnelles doivent être protégées et leur confidentialité doit être garantie.

### 3. Identification des besoins et spécification des fonctionnalités

#### 3.1. Elaborations des diagrammes des cas d'utilisation

Maintenant voici les diagrammes des cas d'utilisation de notre application :

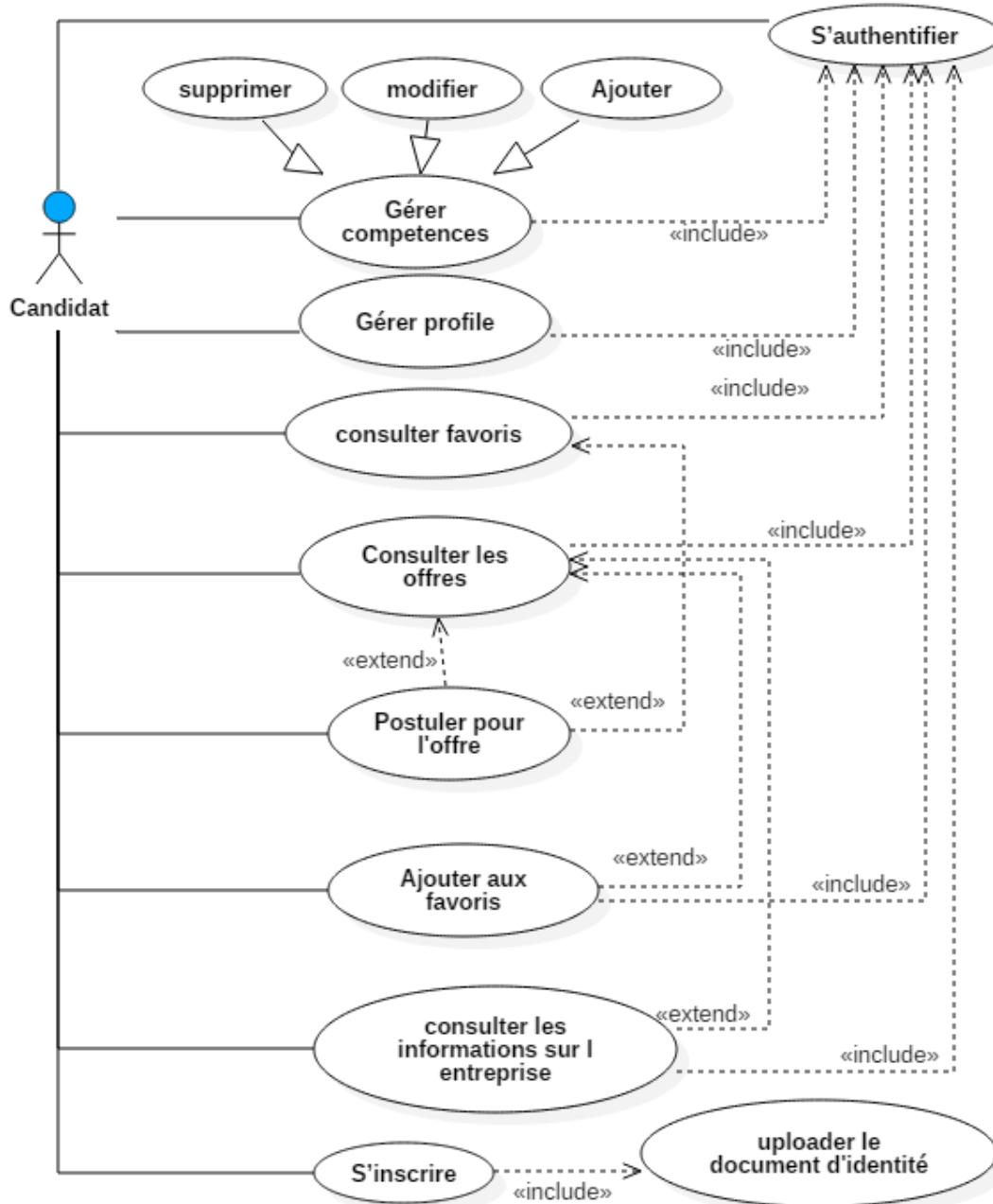


Figure 3. 1. Diagrammes de cas de d'utilisation pour l'acteur « Candidat ».

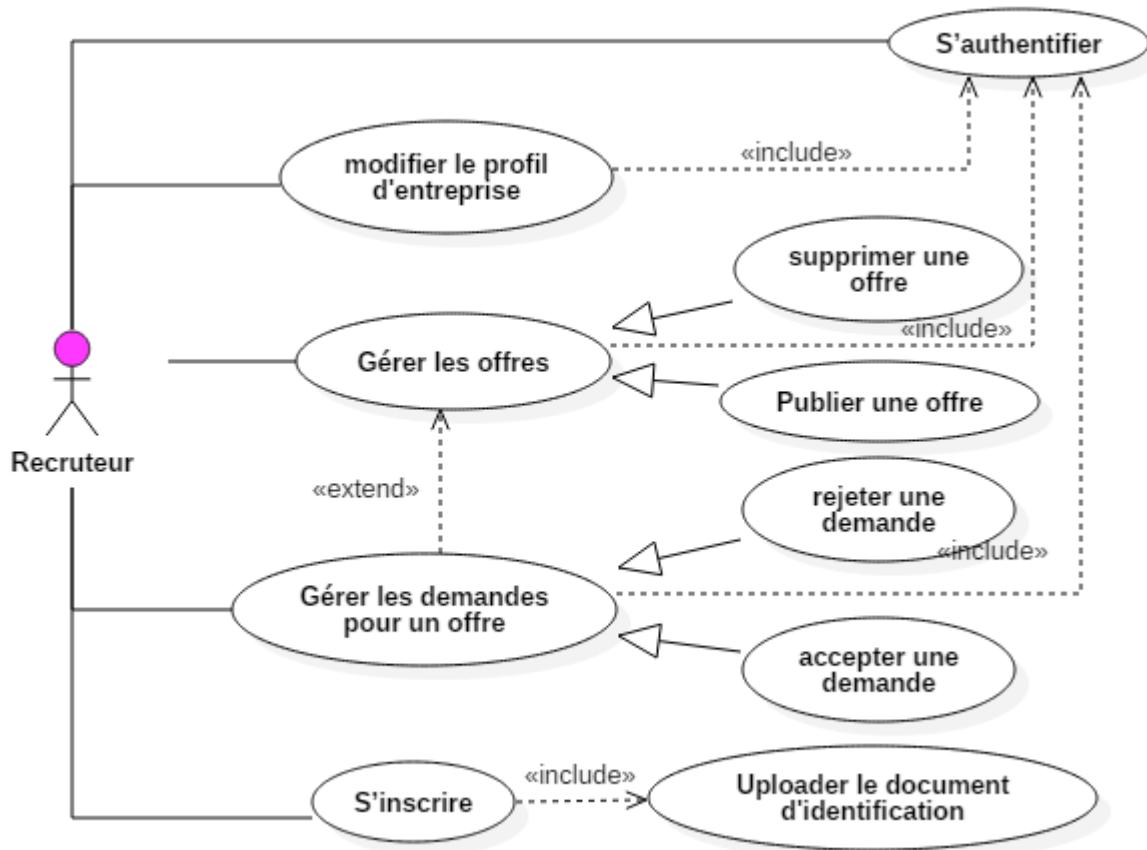


Figure 3. 2. Diagrammes de cas de d'utilisation pour l'acteur « Recruteur ».

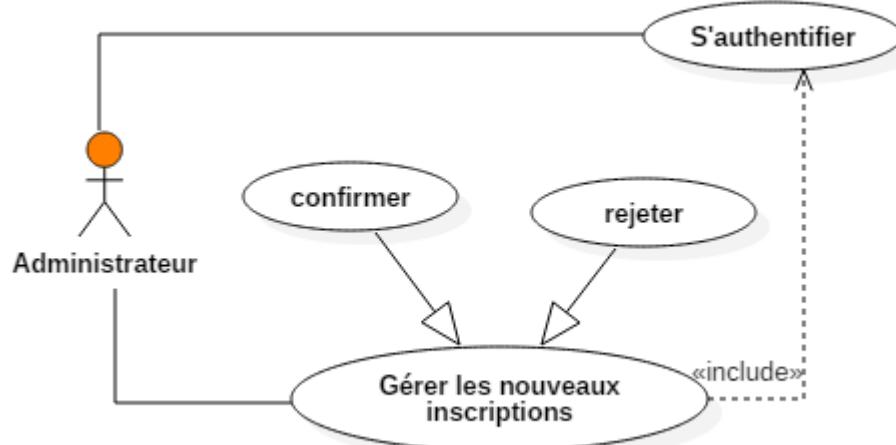


Figure 3. 3. Diagrammes de cas de d'utilisation pour l'acteur « Administrateur ».

### 3.2. Description textuelle des cas d'utilisation

<b>Cas d'utilisation :</b>	S'Authentifier
<b>Acteurs principaux :</b>	Utilisateur (Candidat, Recruteur, Administrateur.).
<b>Acteurs secondaires :</b>	/
<b>Objectifs :</b>	Permettre à l'utilisateur d'accéder et d'utiliser les différents services d'application.
<b>Précondition :</b>	/
<b>Post-condition :</b>	L'utilisateur est authentifié et a accès aux fonctionnalités du système selon son profil (Candidat, Recruteur, Administrateur).
<b>Scenario nominal :</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur saisit ses informations (E-mail et mot de passe).</li> <li>2. L'utilisateur envoie le formulaire.</li> <li>3. Le système accède à la base de données et vérifie le contenu du formulaire.</li> <li>4. Le système affiche la page d'accueil de l'utilisateur.</li> </ol>
<b>Scenarios alternatifs :</b>	<ol style="list-style-type: none"> <li>3. a) Si L'utilisateur saisit une information invalide le système affiche un message d'erreur, <b>Aller à 1</b>.</li> <li>3. b) formulaire vide, <b>Aller à 1</b>.</li> </ol>

Table 3. 1. Fiche descriptif du cas « S'authentifier ».

<b>Cas d'utilisation :</b>	Inscrire
<b>Acteurs principaux :</b>	Utilisateur (Candidat, Recruteur)
<b>Acteurs secondaires :</b>	/
<b>Objectifs :</b>	Donner une identification pour différents types d'utilisateur.
<b>Précondition :</b>	/
<b>Post-condition :</b>	Inscrire l'utilisateur provisoirement en attendant la validation par l'administrateur.
<b>Scenario nominal :</b>	<ol style="list-style-type: none"> <li>1. L'Utilisateur remplit le formulaire d'inscription</li> <li>2. L'Utilisateur envoie le formulaire.</li> <li>3. Le système vérifie le contenu du formulaire.</li> <li>4. Le système la suite de formulaire d'inscription</li> <li>5. L'utilisateur compléter la suite, charger la pièce d'identité (registre d'entreprise) et envoyer.</li> <li>6. Le système vérifier et enregistrer la demande d'inscription.</li> </ol>
<b>Scenarios alternatifs :</b>	<ol style="list-style-type: none"> <li>3.a) Si l'utilisateur saisit E-mail invalide le système affiche un message d'erreur. <b>Aller à 1</b>.</li> <li>3. b) Si l'utilisateur n'a pas rempli un champ obligatoire le système affiche un message d'erreur. <b>Aller à 1</b>.</li> <li>3.c) Si l'utilisateur saisit E-mail existe déjà le système affiche un message d'erreur <b>Aller à 1</b>.</li> </ol>

Table 3. 2. Fiche descriptif du cas « Incrire ».

<b>Cas d'utilisation :</b>	Gérer profil
<b>Acteurs principaux :</b>	Candidat.
<b>Acteurs secondaires :</b>	/
<b>Objectifs :</b>	Décrire les étapes permettant à un utilisateur authentifié de mettre à jour les informations de son profil.
<b>Précondition :</b>	Le candidat est authentifié.
<b>Post-condition :</b>	/
<b>Scenario nominal :</b>	<ol style="list-style-type: none"> <li>1. Le candidat active la modification.</li> <li>2. Le système affiche le formulaire de modification</li> <li>3. Le candidat modifie l'information.</li> <li>4. Le système vérifie la cohérence des données et enregistre.</li> <li>5. Le système affiche un message de succès.</li> </ol>
<b>Scenarios alternatifs :</b>	<p>4.a) Si le candidat saisit une information invalide ou laisse un champs vide le système affiche un message d'erreur. <b>Aller à 2.</b></p>

Table 3. 3. Fiche descriptif du cas « Gérer profil ».

<b>Cas d'utilisation :</b>	Ajouter compétence
<b>Acteurs principaux :</b>	Candidat.
<b>Acteurs secondaires :</b>	/
<b>Objectifs :</b>	Décrire les étapes permettant à un utilisateur authentifié de mettre à jour les informations de ses compétences.
<b>Précondition :</b>	Le candidat est authentifié.
<b>Post-condition :</b>	/
<b>Scenario nominal :</b>	<ol style="list-style-type: none"> <li>1. Le candidat choisit le type de compétence.</li> <li>2. Le système affiche le formulaire d'ajout</li> <li>3. Le candidat rempli et envoyer les informations.</li> <li>4. Le système vérifie les informations et enregistre.</li> <li>5. Le système affiche un message de succès.</li> </ol>
<b>Scenarios alternatifs :</b>	<p>4.a) Si le candidat saisit une information invalide ou laisse un champs vide le système affiche un message d'erreur. <b>Aller à 2.</b></p>

Table 3. 4. Fiche descriptif du cas « Ajouter compétence ».

<b>Cas d'utilisation :</b>	Consulter la liste des offres.
<b>Acteurs principaux :</b>	Candidat
<b>Acteurs secondaires :</b>	/
<b>Objectifs :</b>	Permettre au candidat de consulter la liste des offres
<b>Précondition :</b>	Le candidat est authentifié.
<b>Post-condition :</b>	/
<b>Scenario nominal :</b>	<ol style="list-style-type: none"> <li>1. Le candidat demande d'accéder à la liste des offres.</li> <li>2. Le système affiche la liste des offres.</li> </ol>
<b>Scenario alternatif :</b>	/

Table 3. 5. Fiche descriptif du cas « Consulter liste des offres ».

<b>Cas d'utilisation :</b>	<b>Postuler pour une offre.</b>
<b>Acteurs principaux :</b>	Candidat
<b>Acteurs secondaires :</b>	/
<b>Objectifs :</b>	Permet au candidat de postuler pour une offre.
<b>Précondition :</b>	Le candidat est authentifié.
<b>Post-condition :</b>	/
<b>Scenario nominal :</b>	<ol style="list-style-type: none"> <li>1. Le candidat demande de postuler pour l'offre.</li> <li>2. Le système afficher un message de confirmation.</li> <li>3. Le Candidat confirme sa demande.</li> <li>4. Le système affiche un message de succès.</li> </ol>
<b>Scenario alternatif :</b>	3.a) Si le candidat click d'annuler sa demande. <b>Retourner au page d'information d'offre.</b>

Table 3. 6. Fiche descriptif du cas « Postuler pour une offre ».

<b>Cas d'utilisation :</b>	<b>Ajouter offre aux favoris.</b>
<b>Acteurs principaux :</b>	Candidat.
<b>Acteurs secondaires :</b>	/
<b>Objectifs :</b>	Permettre au candidat d'ajouter une offre aux liste favoris.
<b>Précondition :</b>	Le candidat est authentifié et accès à la liste des offres.
<b>Post-condition :</b>	/
<b>Scenario nominal :</b>	<ol style="list-style-type: none"> <li>1. Le candidat demande d'ajouter offre aux favoris.</li> <li>2. Le système affiche la liste des offres.</li> </ol>
<b>Scenario alternatif :</b>	/

Table 3. 7. Fiche descriptif du cas « Ajouter offre aux favoris ».

<b>Cas d'utilisation :</b>	<b>Consulter les informations d'entreprise.</b>
<b>Acteurs principaux :</b>	Candidat.
<b>Acteurs secondaires :</b>	/
<b>Objectifs :</b>	Permettre au candidat de voir le profil d'une entreprise.
<b>Précondition :</b>	Le candidat est authentifié et accès à la liste des offres.
<b>Post-condition :</b>	/
<b>Scenario nominal :</b>	<ol style="list-style-type: none"> <li>1. Le candidat demande de consulter les informations d'entreprise.</li> <li>2. Le système récupère les informations d'entreprise.</li> <li>3. Le système affiche les informations d'entreprise.</li> </ol>
<b>Scenario alternatif :</b>	/

Table 3. 8. Fiche descriptif du cas « Consulter les informations d'entreprise ».

<b>Cas d'utilisation :</b>	<b>Publier une offre.</b>
<b>Acteurs principaux :</b>	Recruteur.
<b>Acteurs secondaires :</b>	/
<b>Objectifs :</b>	Permet au candidat de postuler pour un stage
<b>Précondition :</b>	Le recruteur est authentifié
<b>Post-condition :</b>	/
<b>Scenario nominal :</b>	<ol style="list-style-type: none"> <li>1. Le recruteur rempli et envoyer le contenu d'offre.</li> <li>2. Le système vérifie le contenu et publier l'offre.</li> <li>3. Le système affiche l'offre dans la liste des offres.</li> </ol>
<b>Scenario alternatif :</b>	<ol style="list-style-type: none"> <li>3.a) Si le recruteur saisit une information invalide (champs vide) le système affiche message d'erreur. <b>Aller à 1.</b></li> </ol>

Table 3. 9. Fiche descriptif du cas « Publier une offre ».

<b>Cas d'utilisation :</b>	<b>Supprimer une offre.</b>
<b>Acteurs principaux :</b>	Recruteur.
<b>Acteurs secondaires :</b>	/
<b>Objectifs :</b>	Permet au recruteur de supprimer une offre.
<b>Précondition :</b>	Le recruteur est authentifié.
<b>Post-condition :</b>	/
<b>Scenario nominal :</b>	<ol style="list-style-type: none"> <li>1. Le recruteur demander de supprimer l'offre.</li> <li>2. Le system afficher un message de confirmation.</li> <li>3. Le recruteur confirme sa demande.</li> <li>4. Le système supprime l'offre.</li> </ol>
<b>Scenario alternatif :</b>	<ol style="list-style-type: none"> <li>3.a) Si le candidat annulait sa demande. <b>Retourner à la liste des offres.</b></li> </ol>

Table 3. 10. Fiche descriptif du cas « supprimer une offre ».

<b>Cas d'utilisation :</b>	<b>Confirmer une demande inscription.</b>
<b>Acteurs principaux :</b>	Administrateur.
<b>Acteurs secondaires :</b>	/
<b>Objectifs :</b>	Permet à l'administrateur de confirmer une demande d'inscription.
-	L'administrateur est authentifié.
<b>Post-condition :</b>	/
<b>Scenario nominal :</b>	<ol style="list-style-type: none"> <li>1. L'administrateur demander de confirmer la demande d'inscription.</li> <li>2. Le system afficher un message de confirmation.</li> <li>3. L'administrateur confirme sa demande.</li> <li>4. Le système confirme la demande et envoie une notification au demandeur.</li> </ol>
<b>Scenario alternatif :</b>	<ol style="list-style-type: none"> <li>3.a) Si l'administrateur annulait sa demande. <b>Retourner à la liste des demandes d'inscription.</b></li> </ol>

Table 3. 11. Fiche descriptif du cas « Confirmer une demande d'inscription ».

<b>Cas d'utilisation :</b>	Confirmer une demande inscription.
<b>Acteurs principaux :</b>	Administrateur.
<b>Acteurs secondaires :</b>	/
<b>Objectifs :</b>	Permet à l'administrateur de confirmer une demande d'inscription.
-	L'administrateur est authentifié.
<b>Post-condition :</b>	/
<b>Scenario nominal :</b>	<ol style="list-style-type: none"> <li>1. L'administrateur demander de rejeter la demande.</li> <li>2. Le system afficher un message de confirmation.</li> <li>3. L'administrateur confirme sa demande.</li> <li>4. Le système rejet la demande et affiche un message de succès.</li> </ol>
<b>Scenario alternatif :</b>	3.a) Si l'administrateur annulait sa demande. <b>Retourner à la liste des demandes d'inscription.</b>

Table 3. 12. Fiche descriptif du cas « Rejeter une demande d'inscription ».

### 3.3. Spécification détaillée des besoins (Les Diagrammes de séquence système)

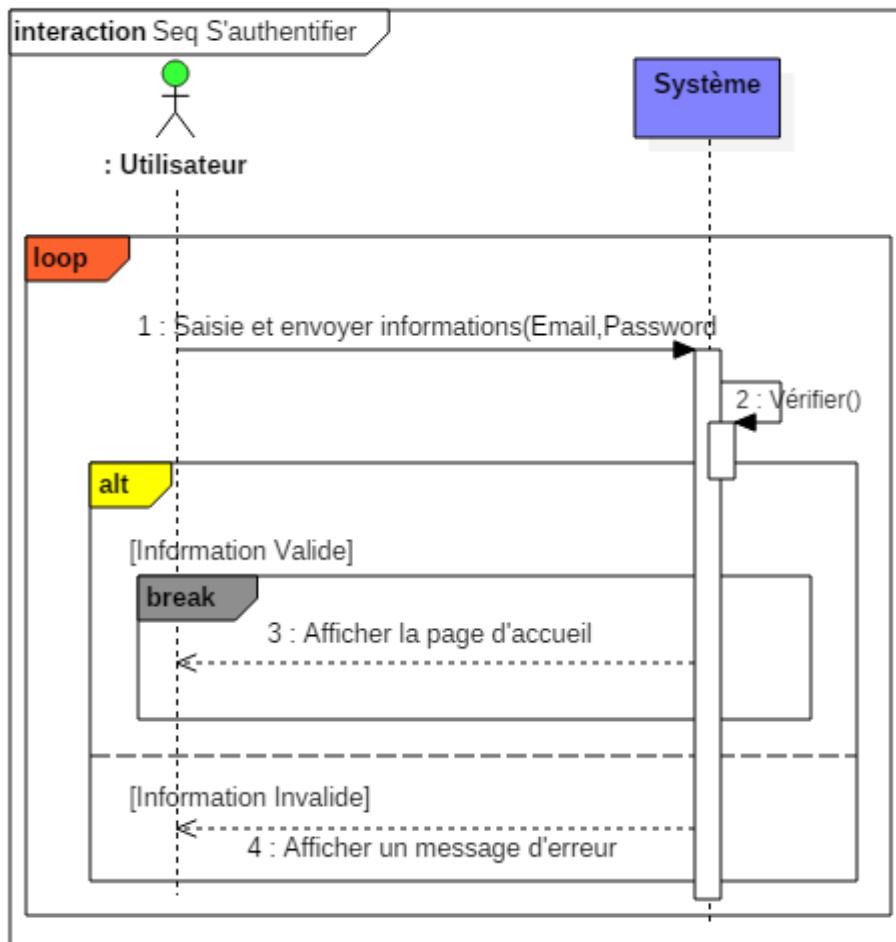


Figure 3. 4. Diagramme de séquence système du cas « S'authentifier ».

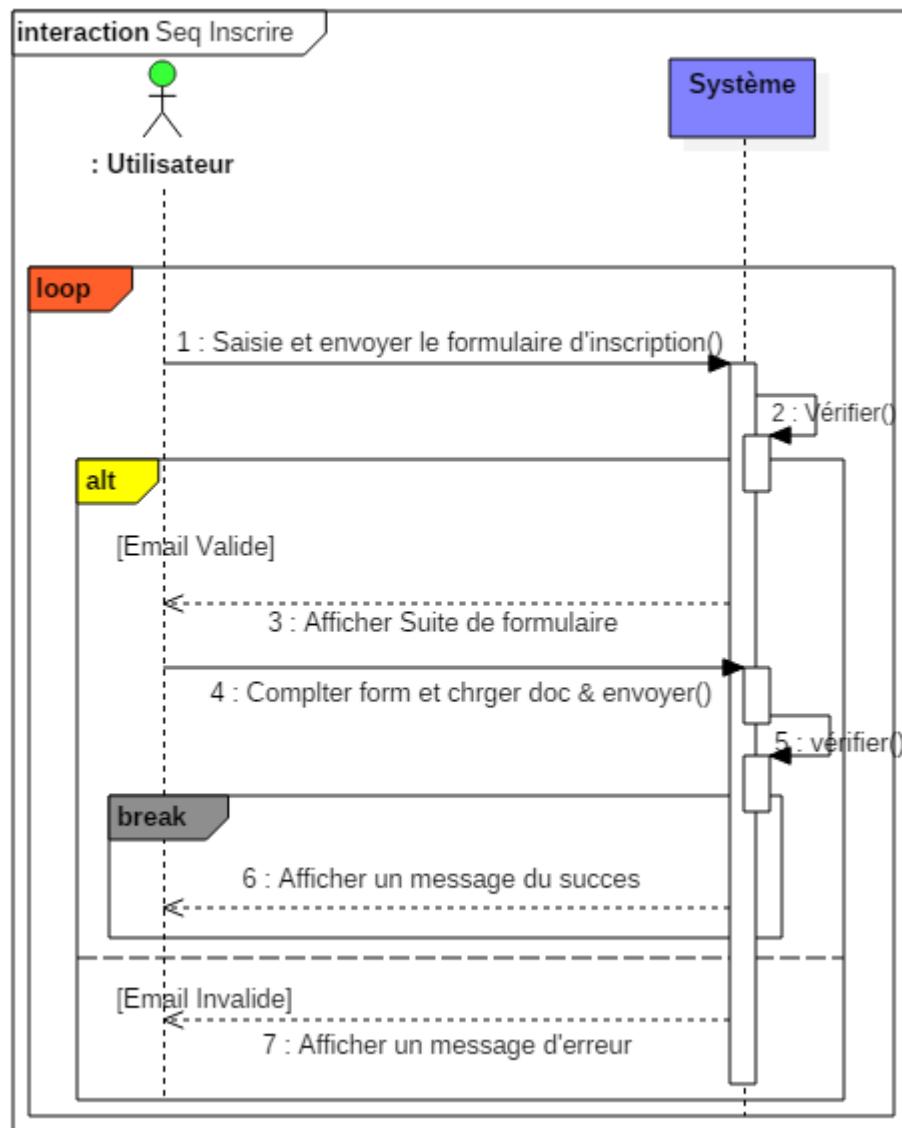


Figure 3. 5. Diagramme de séquence système du cas « Incrire ».

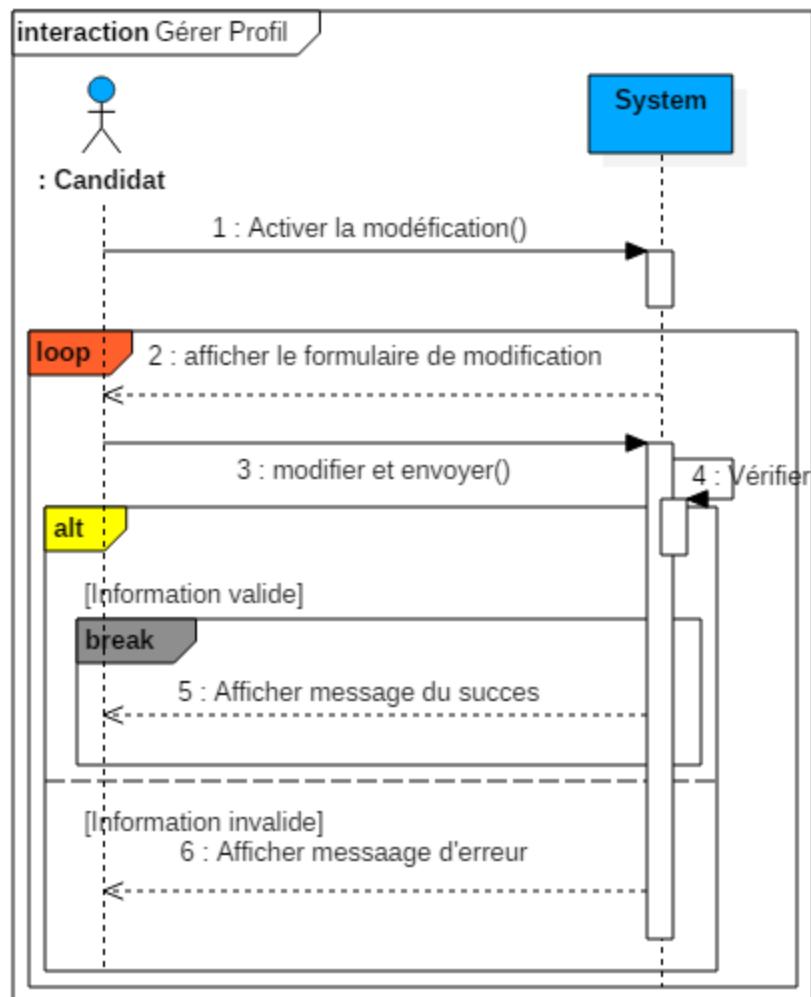


Figure 3. 6. Diagramme de séquence système du cas « Gérer profil ».

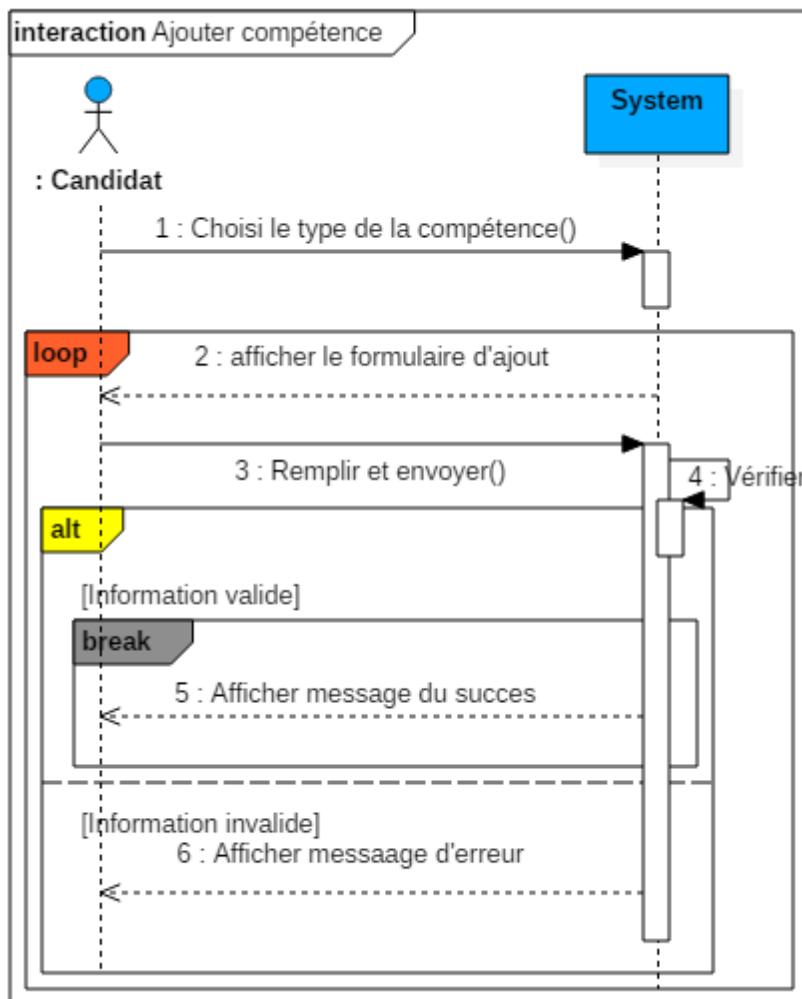


Figure 3. 7. Diagramme de séquence système du cas « Ajouter compétence ».

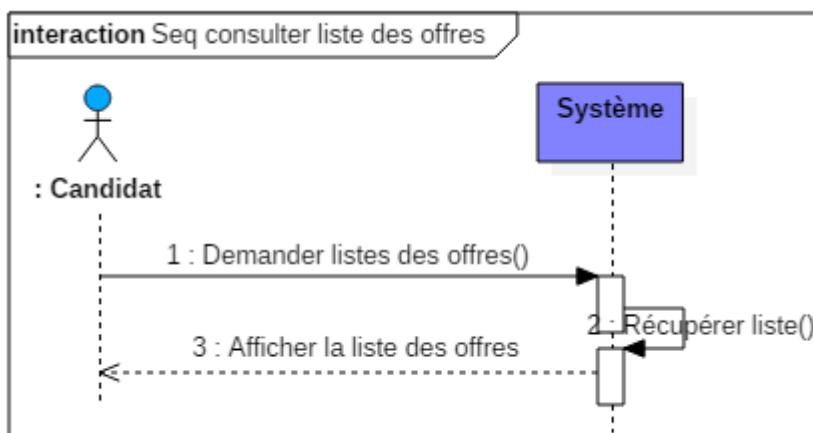


Figure 3. 8. Diagramme de séquence système du cas « Consulter la liste des offres ».

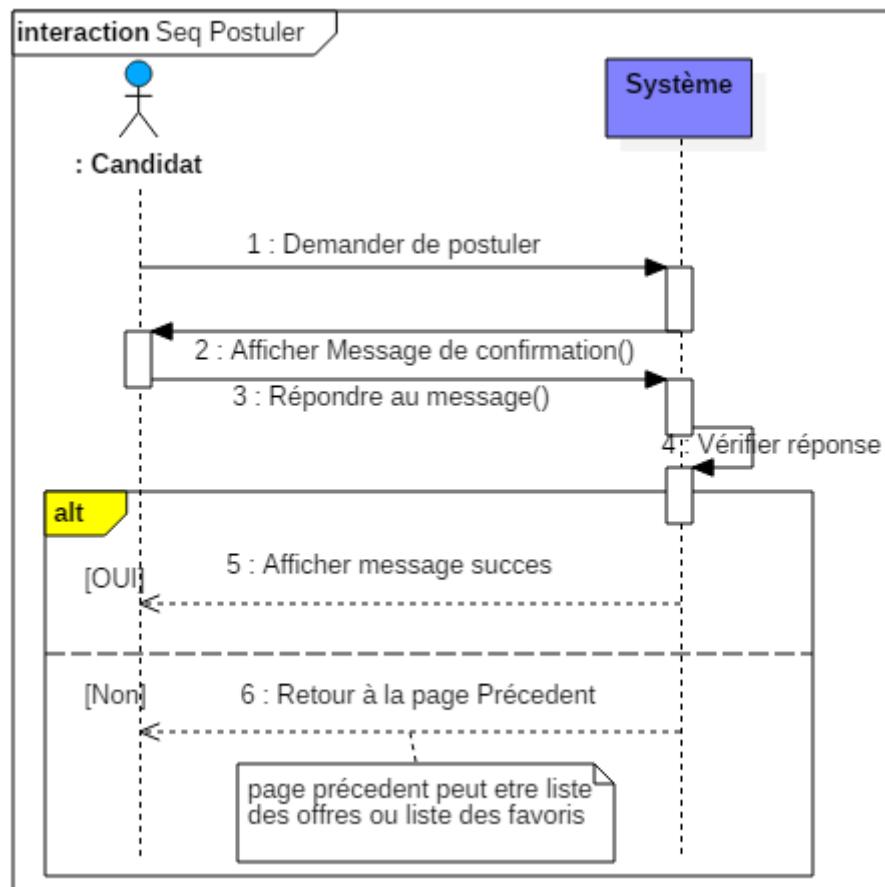


Figure 3. 9. Diagramme de séquence système du cas « Postuler pour une offre ».

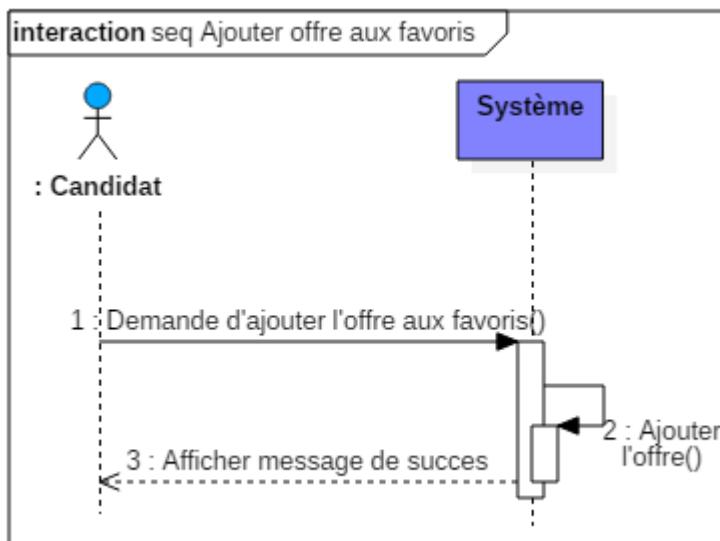


Figure 3. 10. Diagramme de séquence système du cas « Ajouter une offre aux favoris ».

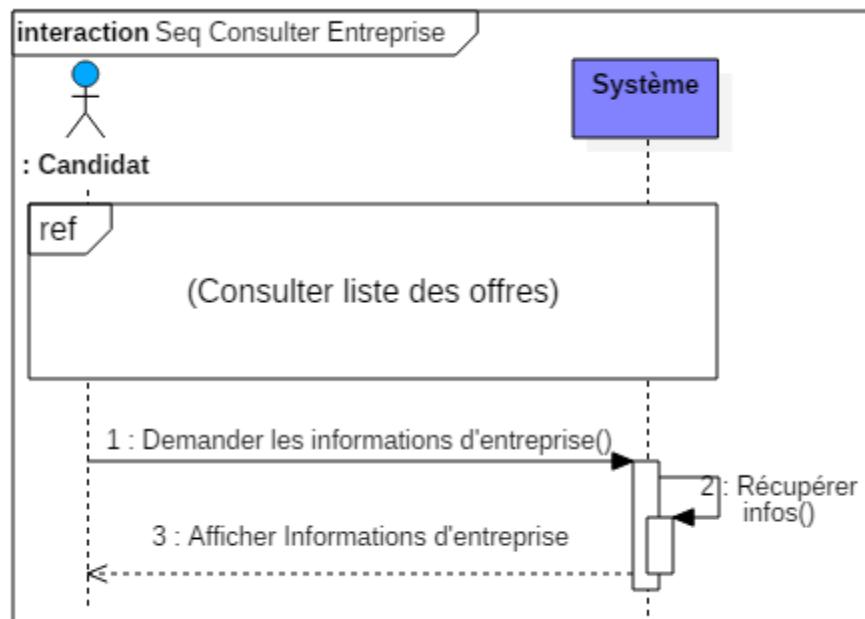


Figure 3. 11. Diagramme de séquence système du cas « Consulter les informations d'entreprise ».

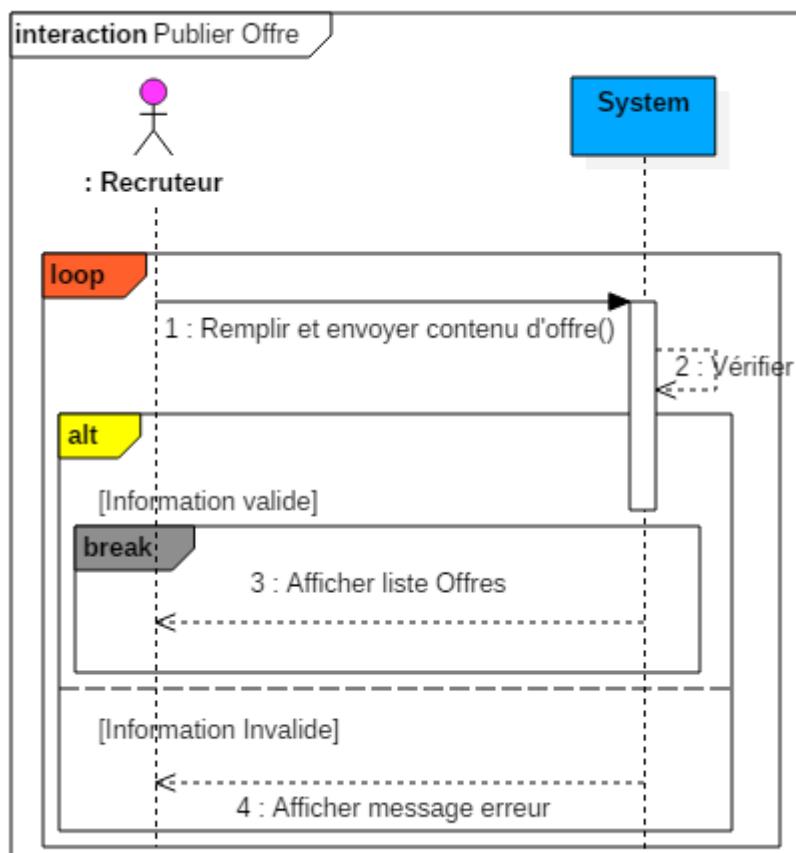


Figure 3. 12. Diagramme de séquence système du cas « Publier une offre ».

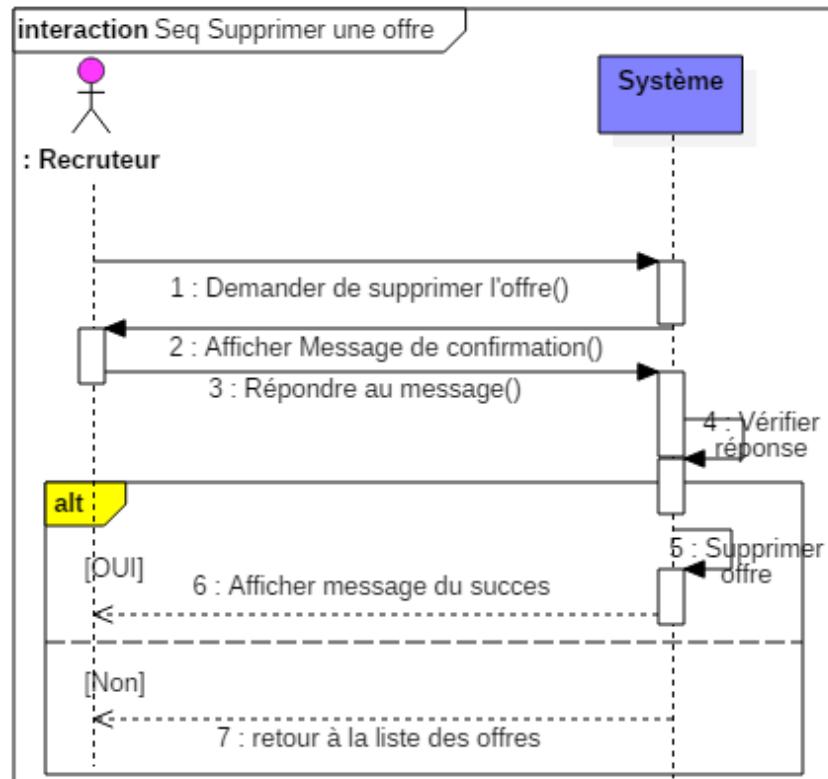


Figure 3. 13. Diagramme de séquence système du cas « Supprimer une offre ».

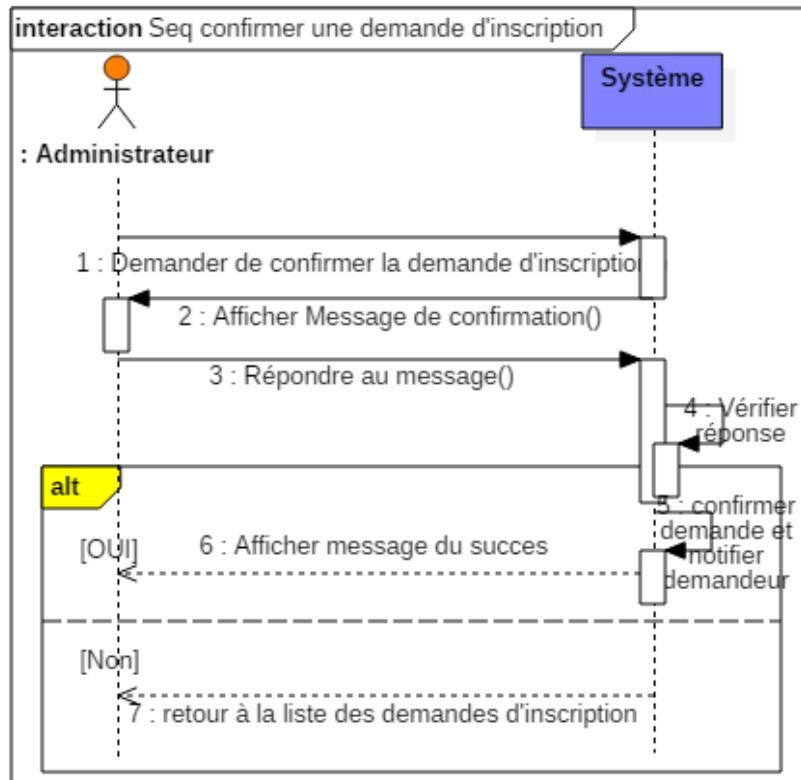
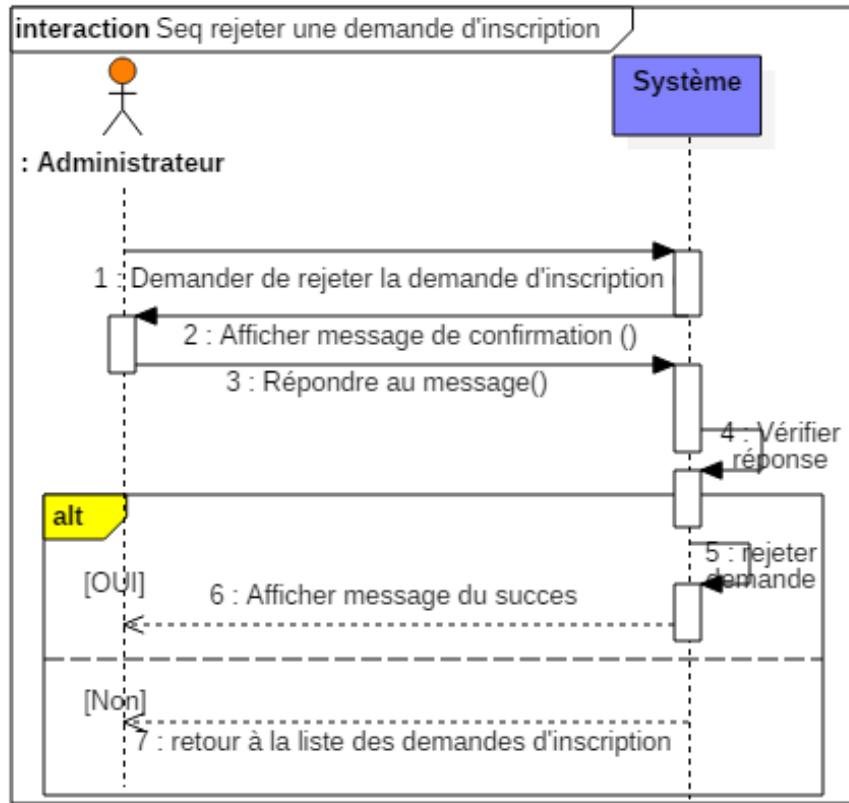


Figure 3. 14. Diagramme de séquence système du cas « Confirmer une demande d'inscription ».



**Figure 3. 15.** Diagramme de séquence système du cas « Rejeter une demande d'inscription ».

#### Remarque

Nous avons choisi seulement ces cas pour la présentation d'étude préliminaire et spécification des besoins parce que les autres cas d'utilisation ont le même scénario. Exemple : Le cas « Consulter liste des favoris » a le même scénario que le cas « Consulter liste des offres » qu'il est déjà présenté.

#### 4. La phase d'analyse (Diagrammes d'activités de navigation)

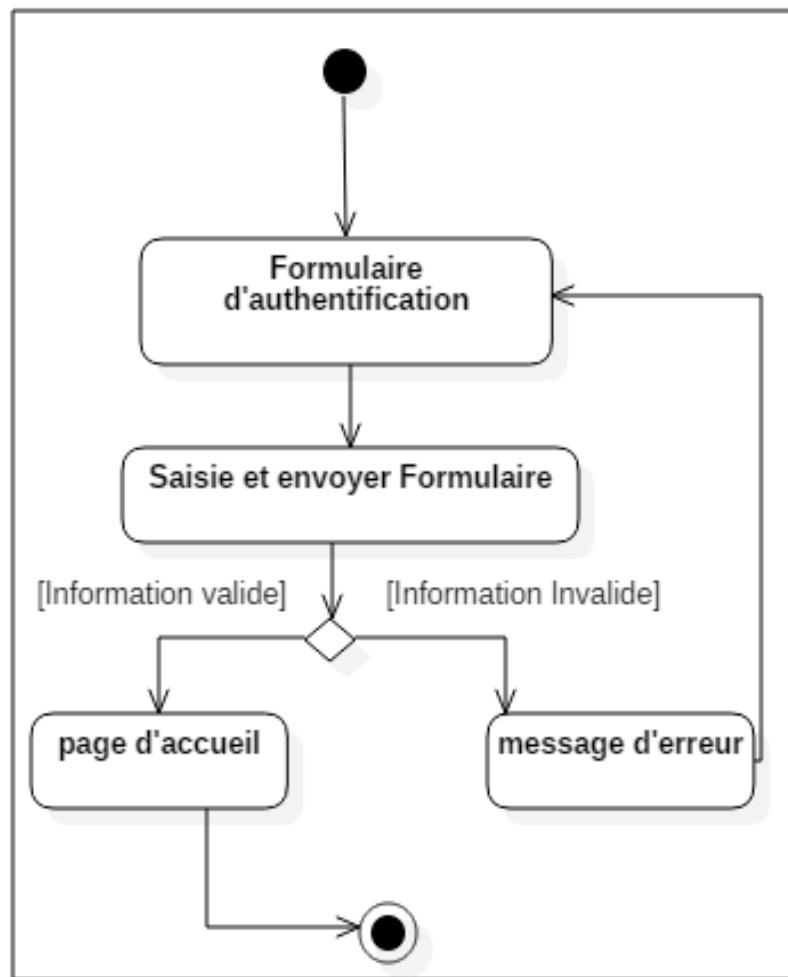


Figure 3. 16. Diagramme d'activité du cas « S'authentifier ».

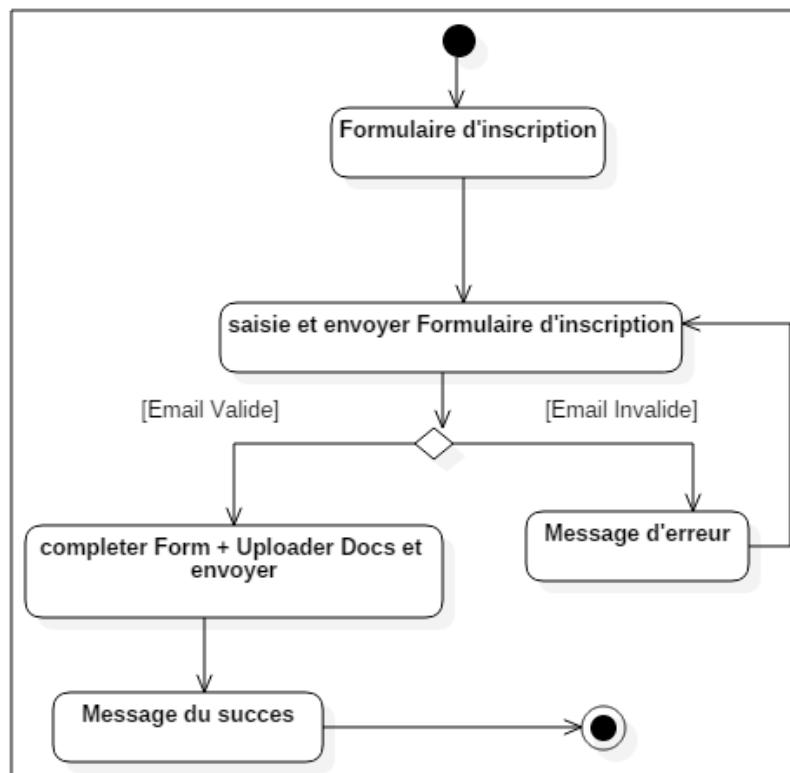


Figure 3. 17. Diagramme d'activité du cas « Incrire ».

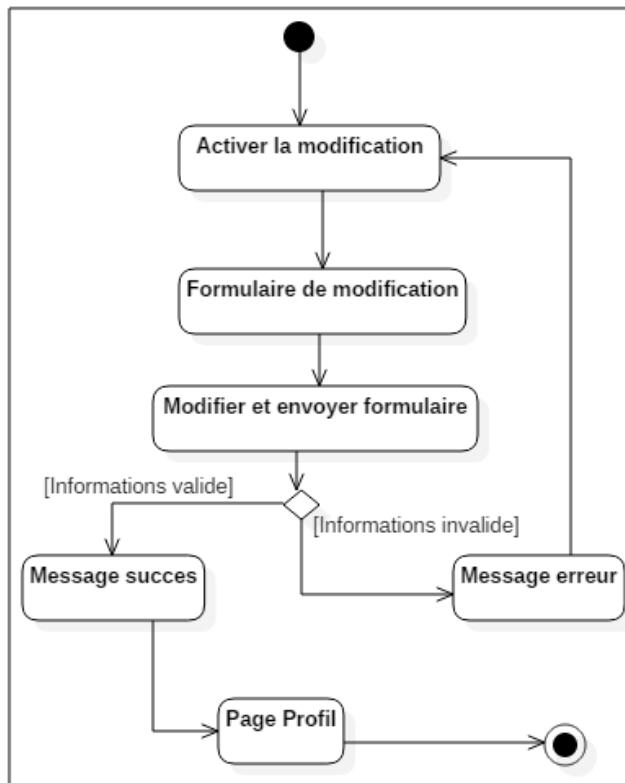


Figure 3. 18. Diagramme d'activité du cas « Gérer profil ».

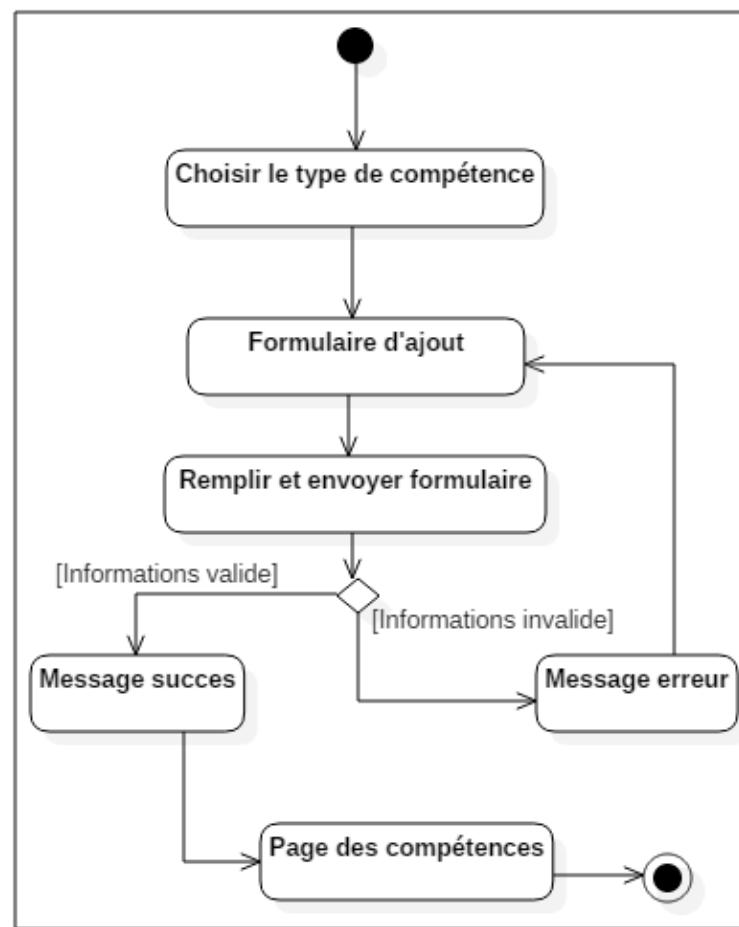


Figure 3. 19. Diagramme d'activité du cas « Ajouter compétence ».

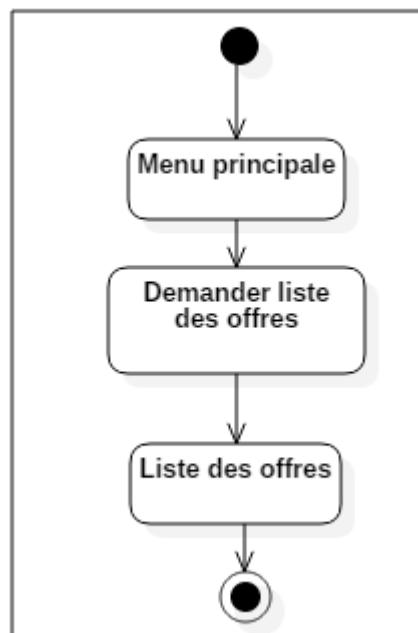


Figure 3. 20. Diagramme d'activité du cas « Consulter liste des offres ».

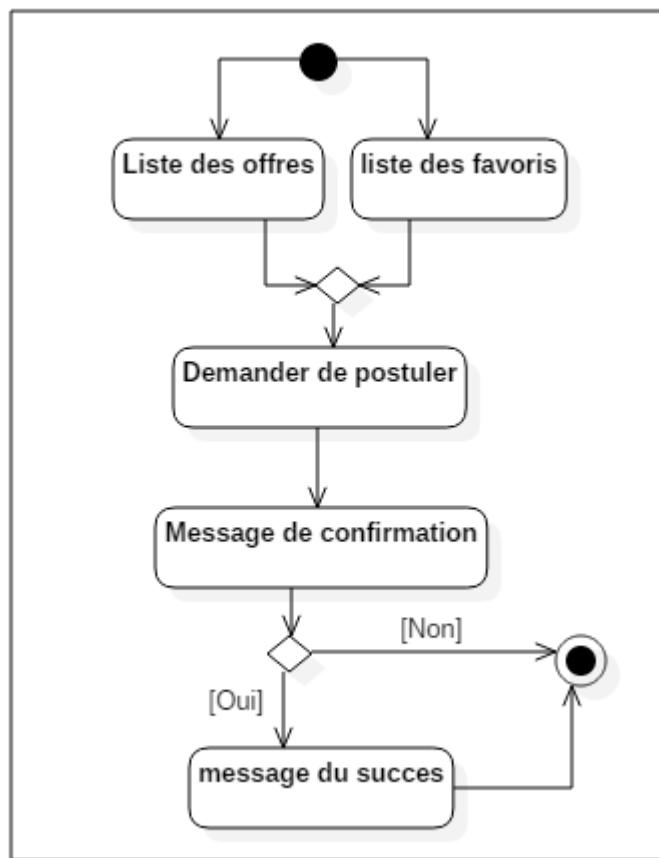


Figure 3. 21. Diagramme d'activité du cas « postuler pour une offre ».

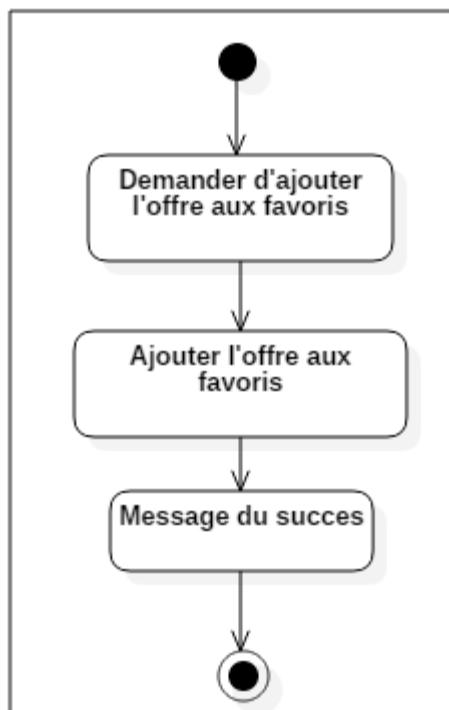
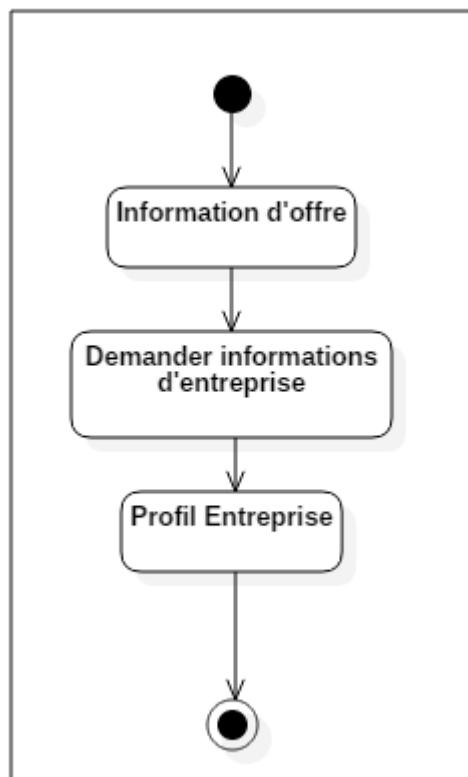
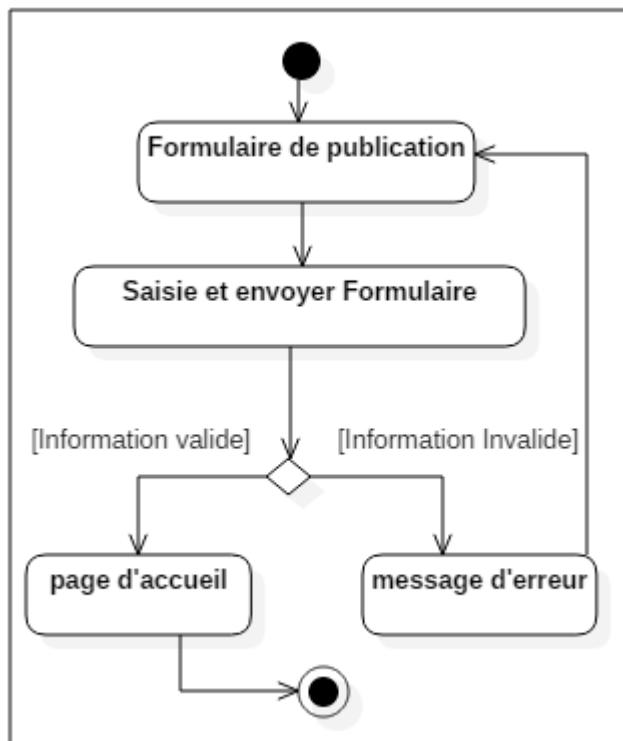


Figure 3. 22. Diagramme d'activité du cas « Ajouter une offre aux favoris ».



**Figure 3. 23.** Diagramme d'activité du cas « Consulter les informations d'entreprise ».



**Figure 3. 24.** Diagramme d'activité du cas « Publier une offre ».

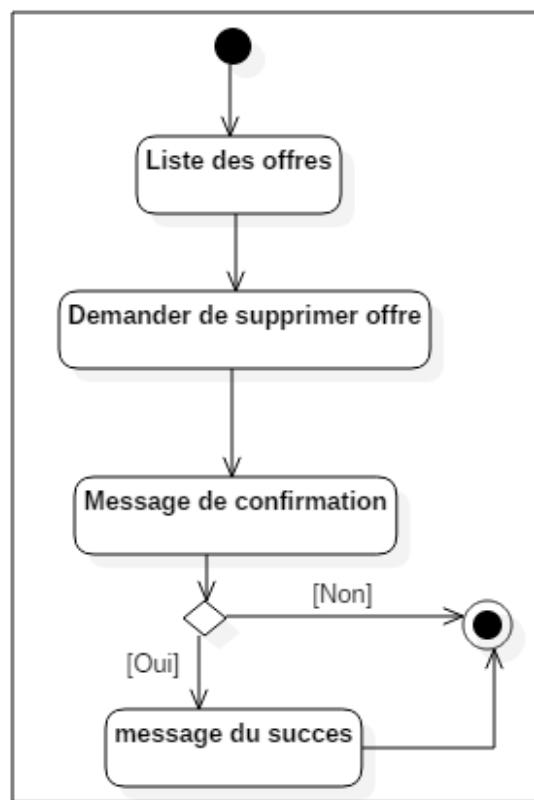


Figure 3. 25. Diagramme d'activité du cas « Supprimer offre ».

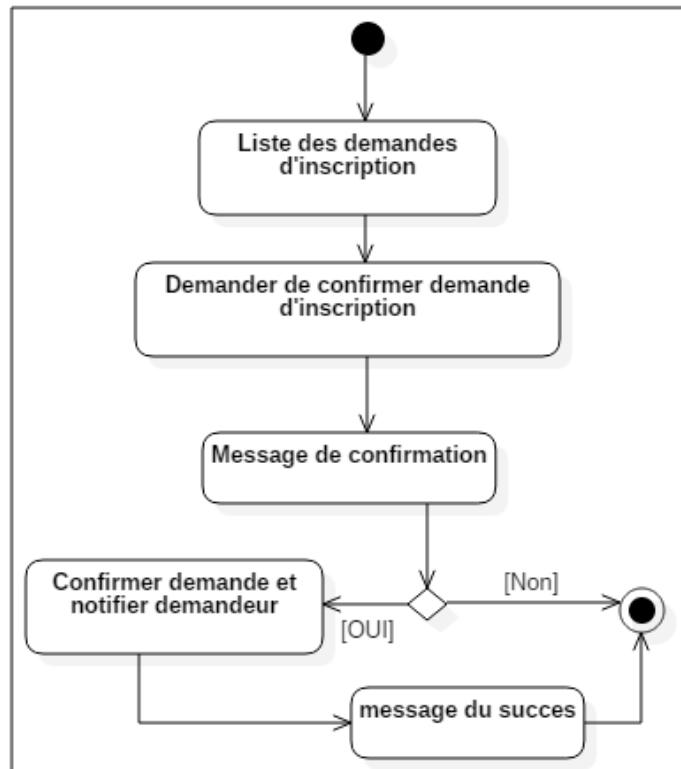


Figure 3. 26. Diagramme d'activité du cas « Confirmer demande d'inscription ».

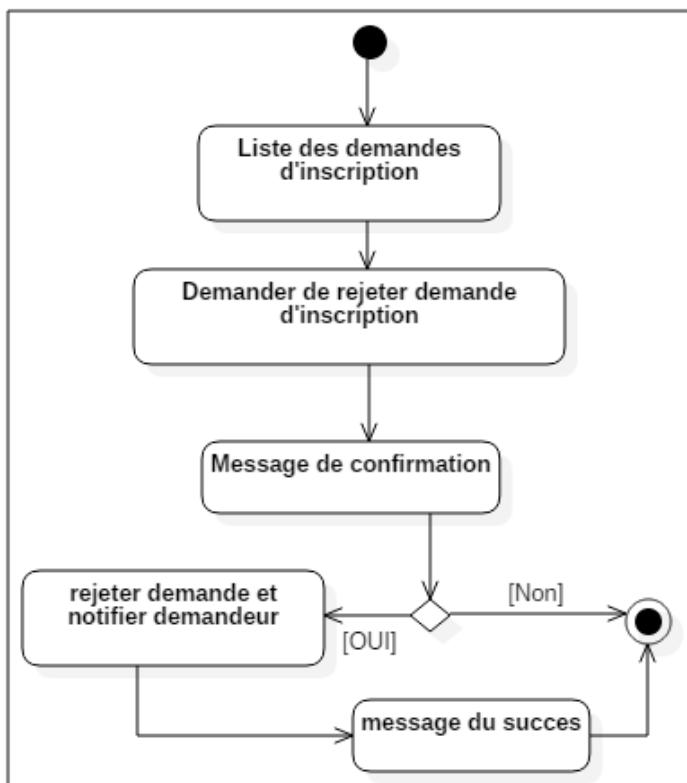


Figure 3. 27. Diagramme d'activité du cas « Rejeter demande d'inscription ».

## 5. Phase de conception

### 5.1. Diagrammes de séquence system détaillé

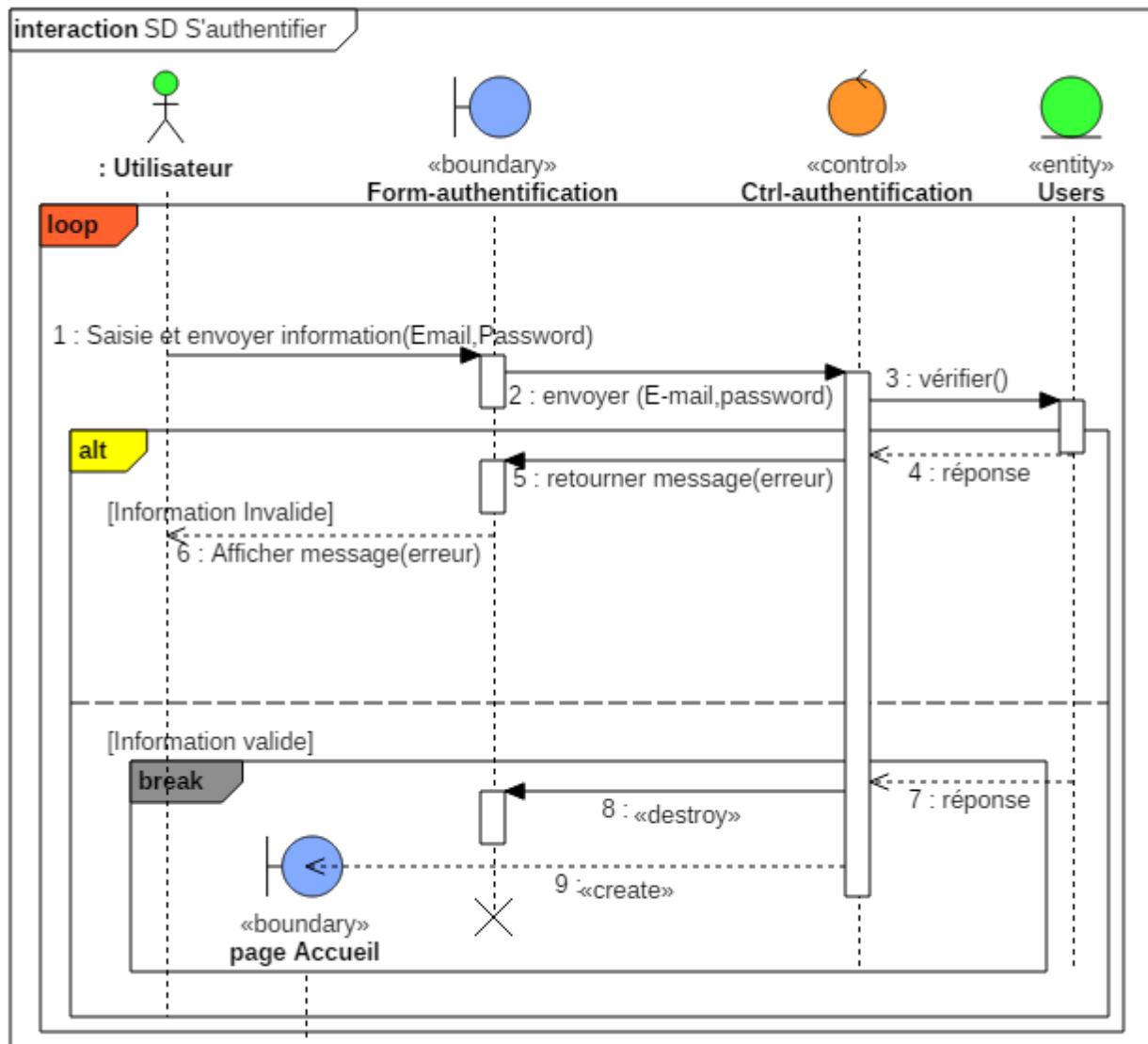


Figure 3. 28. Diagramme de séquence détaillé du cas « s'authentifier ».

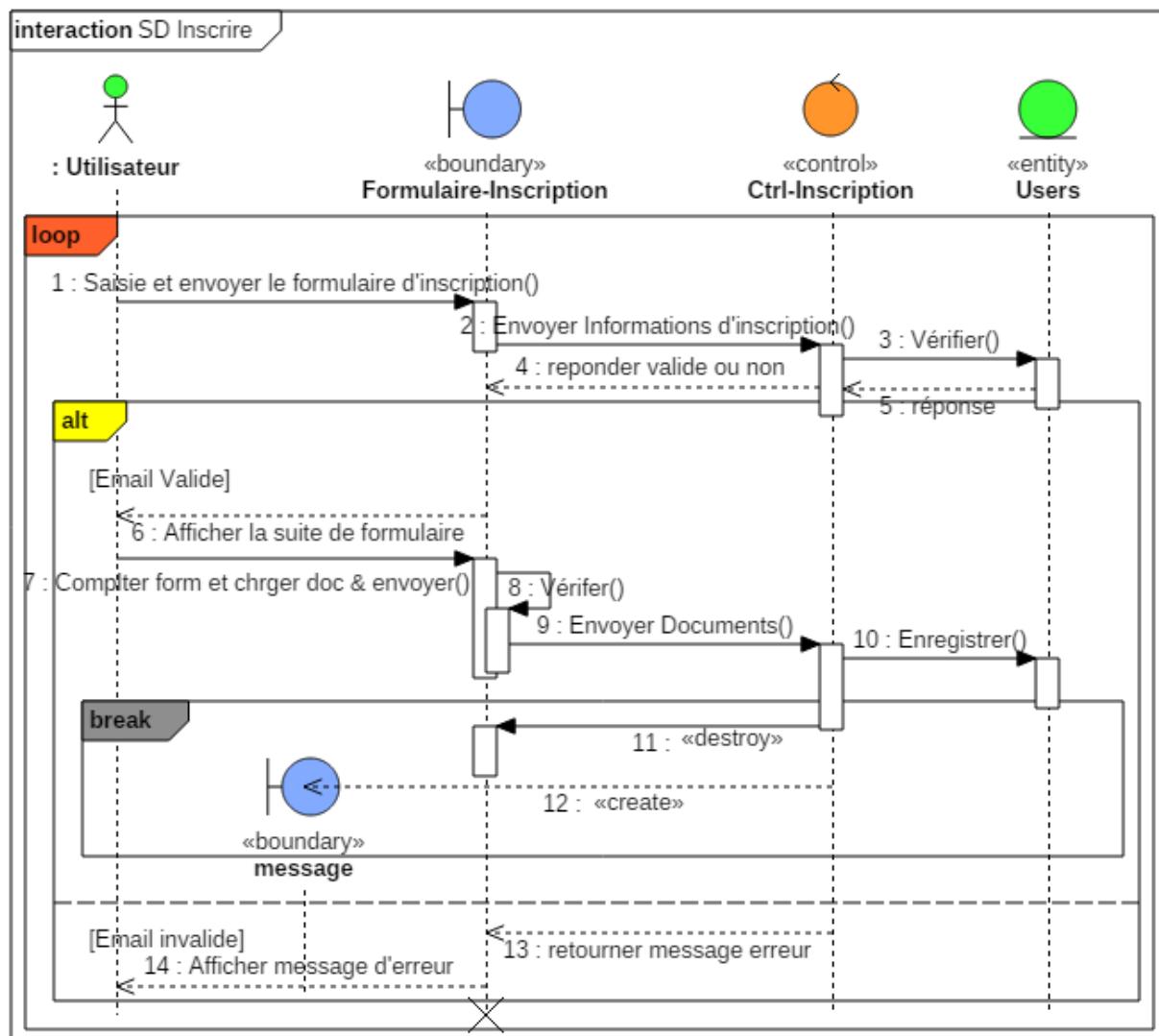


Figure 3. 29. Diagramme de séquence détaillé du cas « Incrire ».

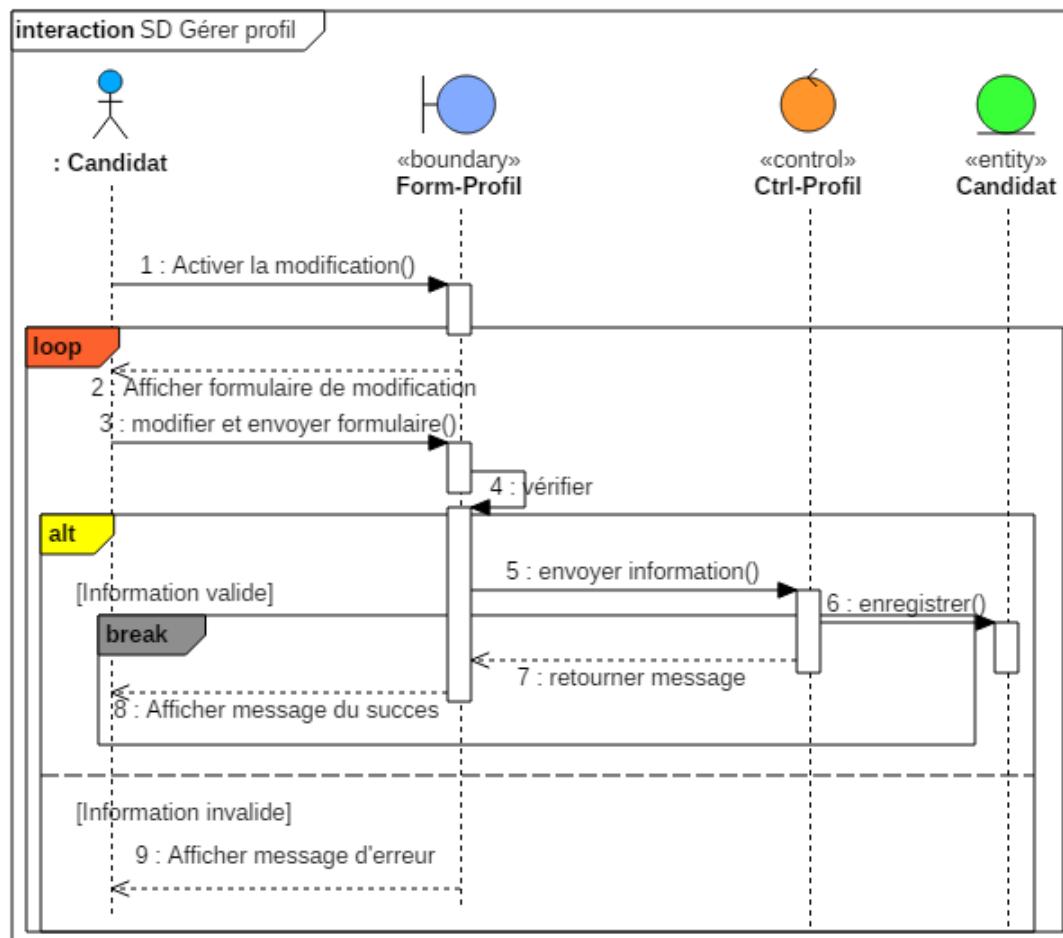


Figure 3. 30. Diagramme de séquence détaillé du cas « Gérer profil ».

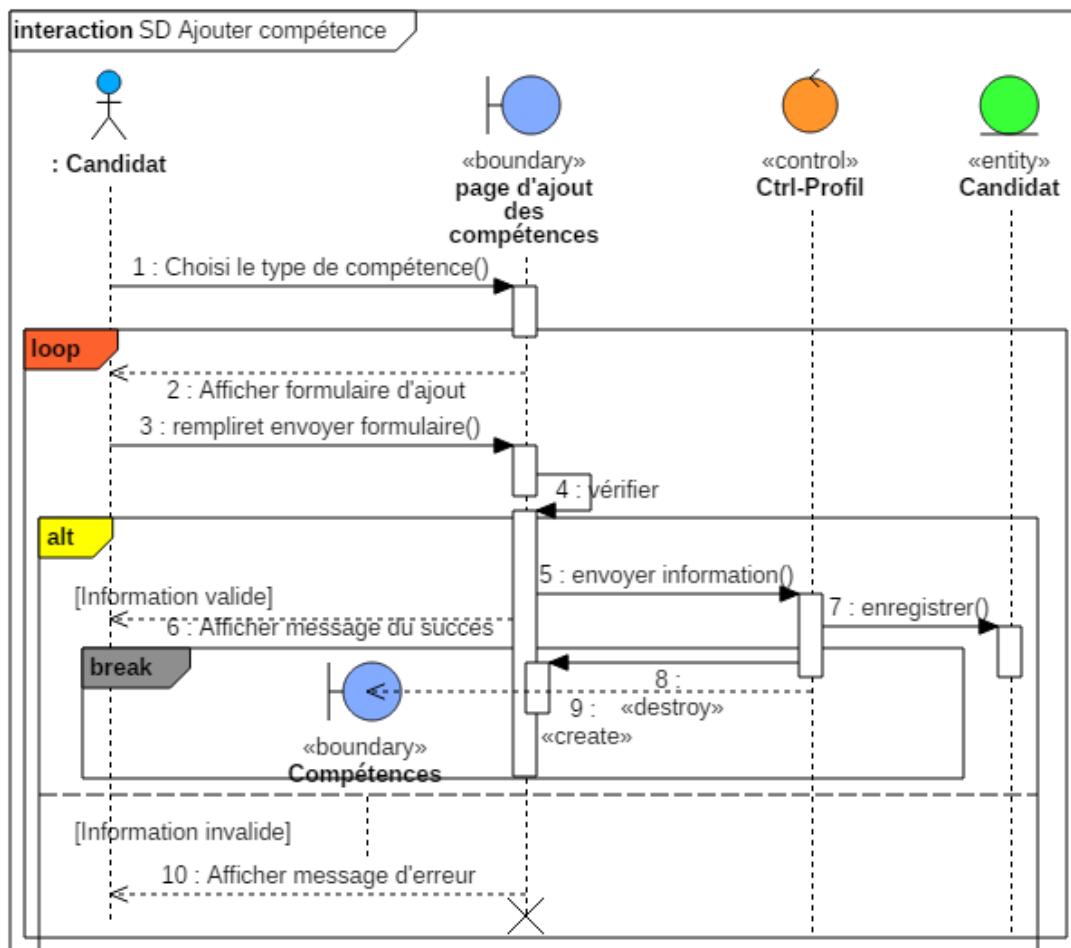


Figure 3.31. Diagramme de séquence détaillé du cas « Ajouter Compétence »

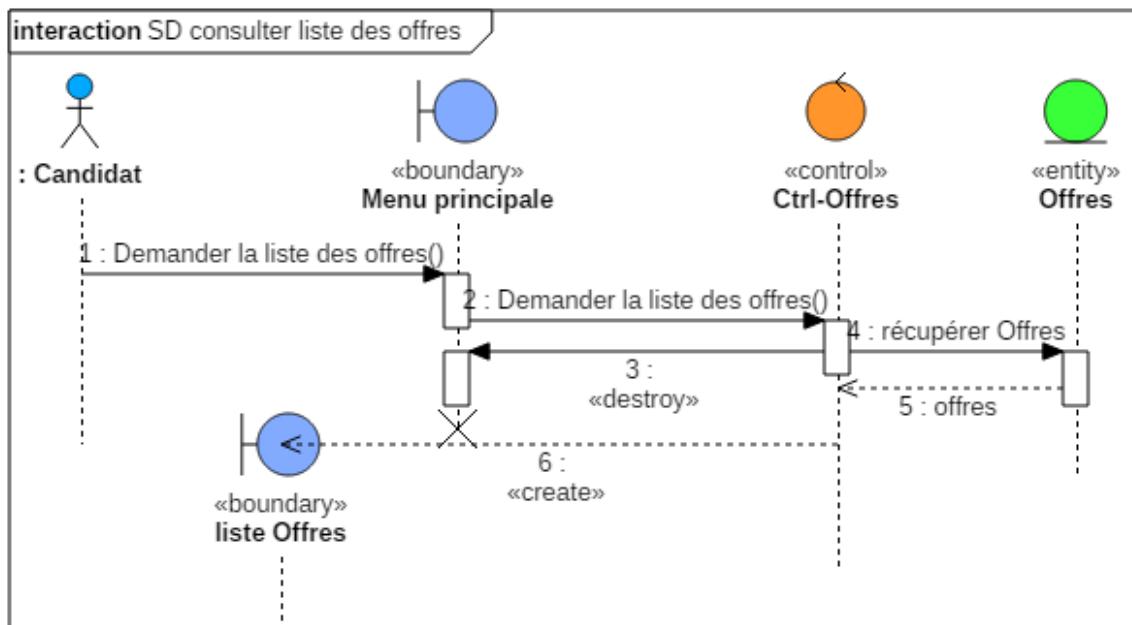


Figure 3.32. Diagramme de séquence détaillé du cas « Consulter liste des offres ».

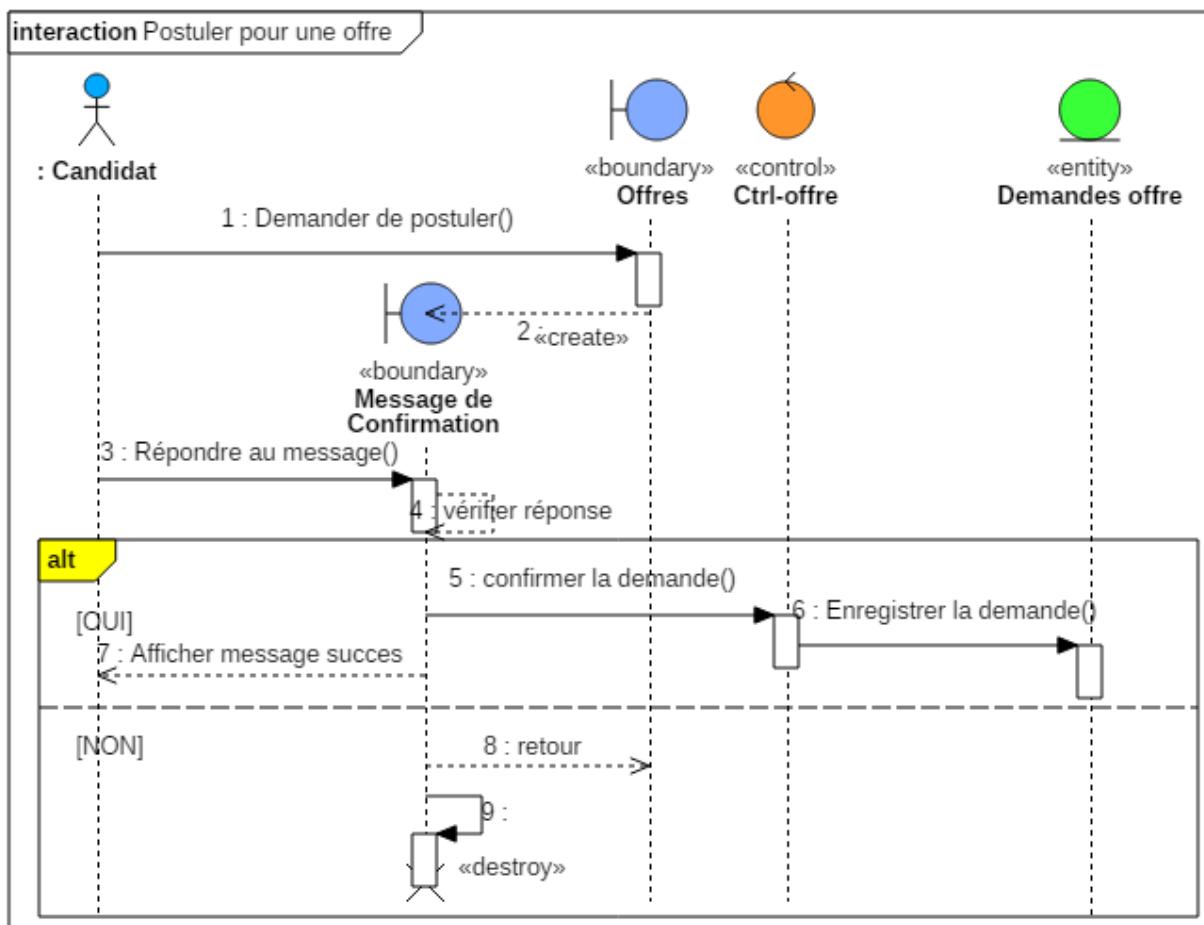


Figure 3. 33. Diagramme de séquence détaillé du cas « Postuler pour une offre ».

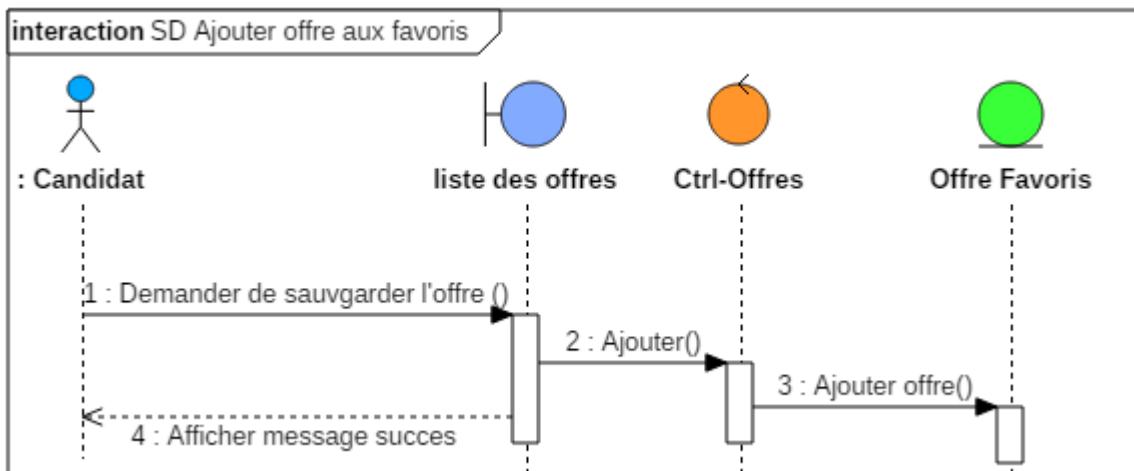


Figure 3. 34. Diagramme de séquence détaillé du cas « Ajouter offre aux favoris ».

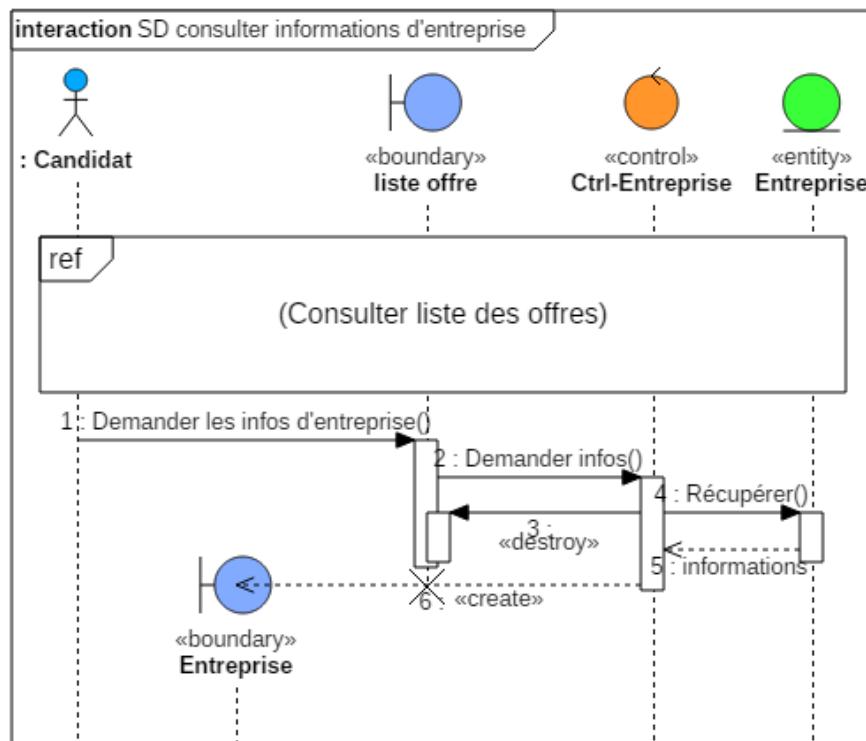


Figure 3.35. Diagramme de séquence détaillé du cas « Consulter les informations d'entreprise ».

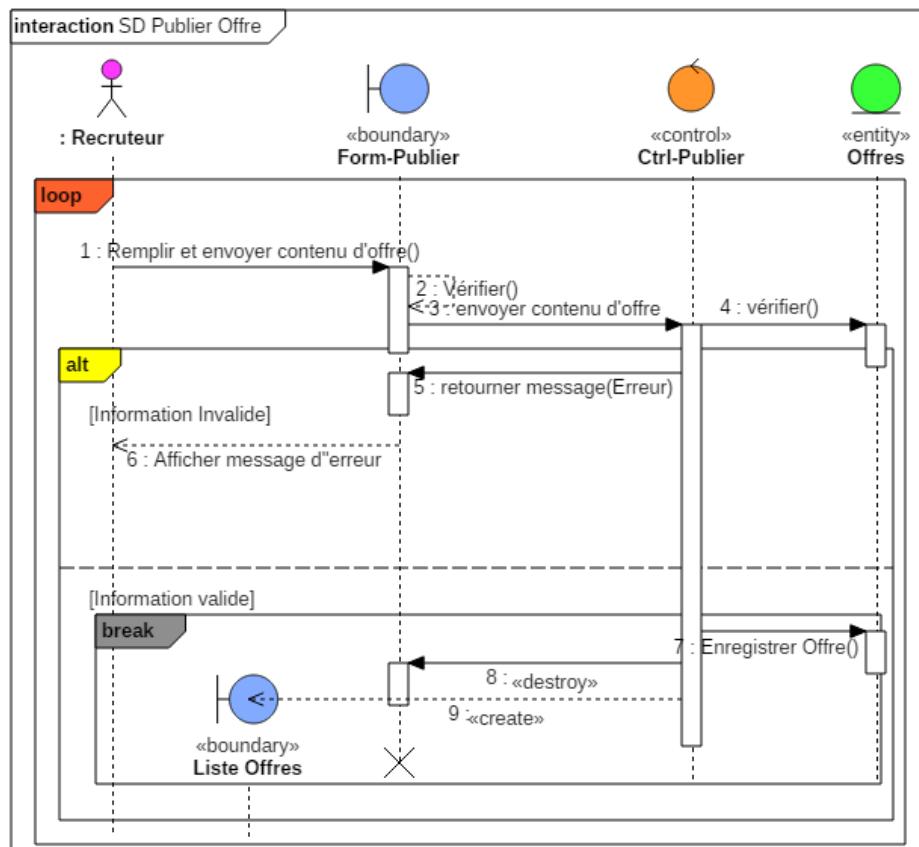


Figure 3.36. Diagramme de séquence détaillé du cas « Publier offre ».

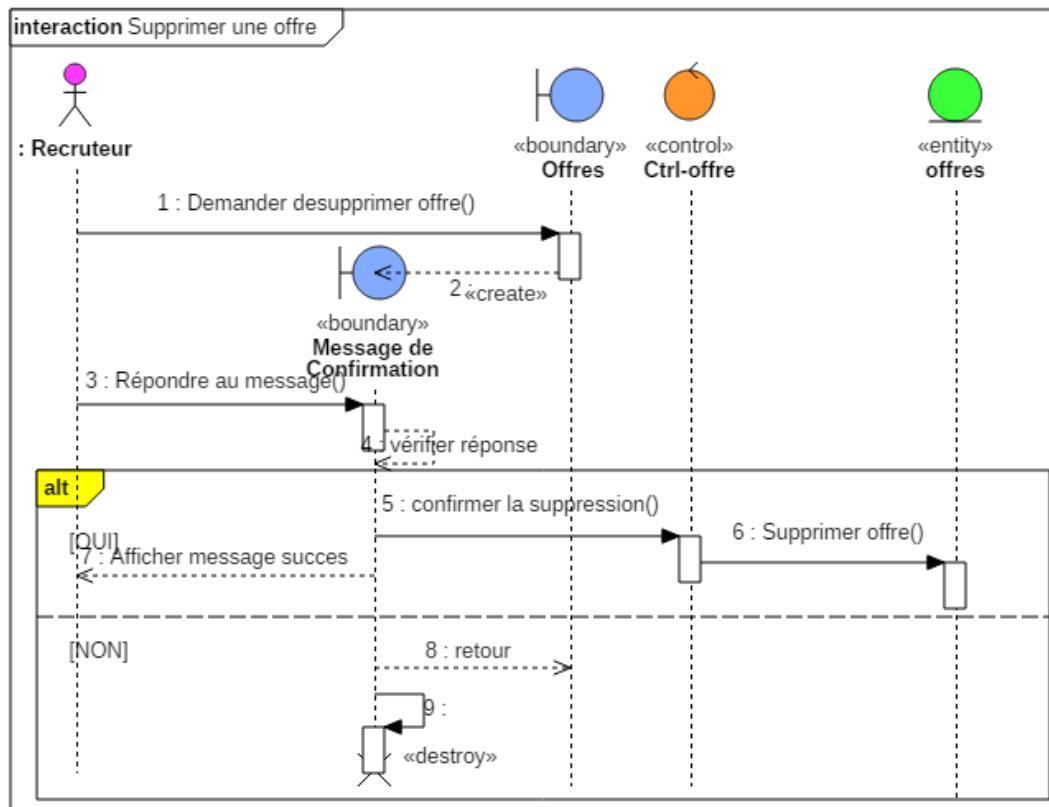


Figure 3.37. Diagramme de séquence détaillé du cas « Supprimer offre ».

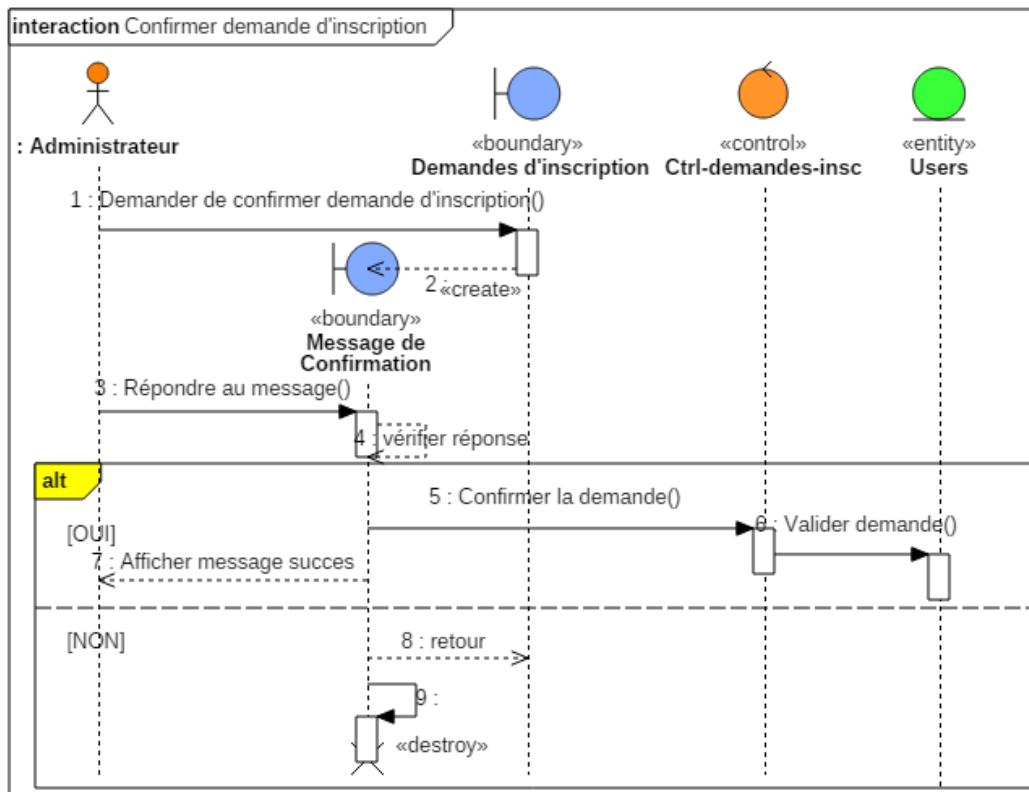


Figure 3.38. Diagramme de séquence détaillé du cas « Confirmer demande d'inscription ».

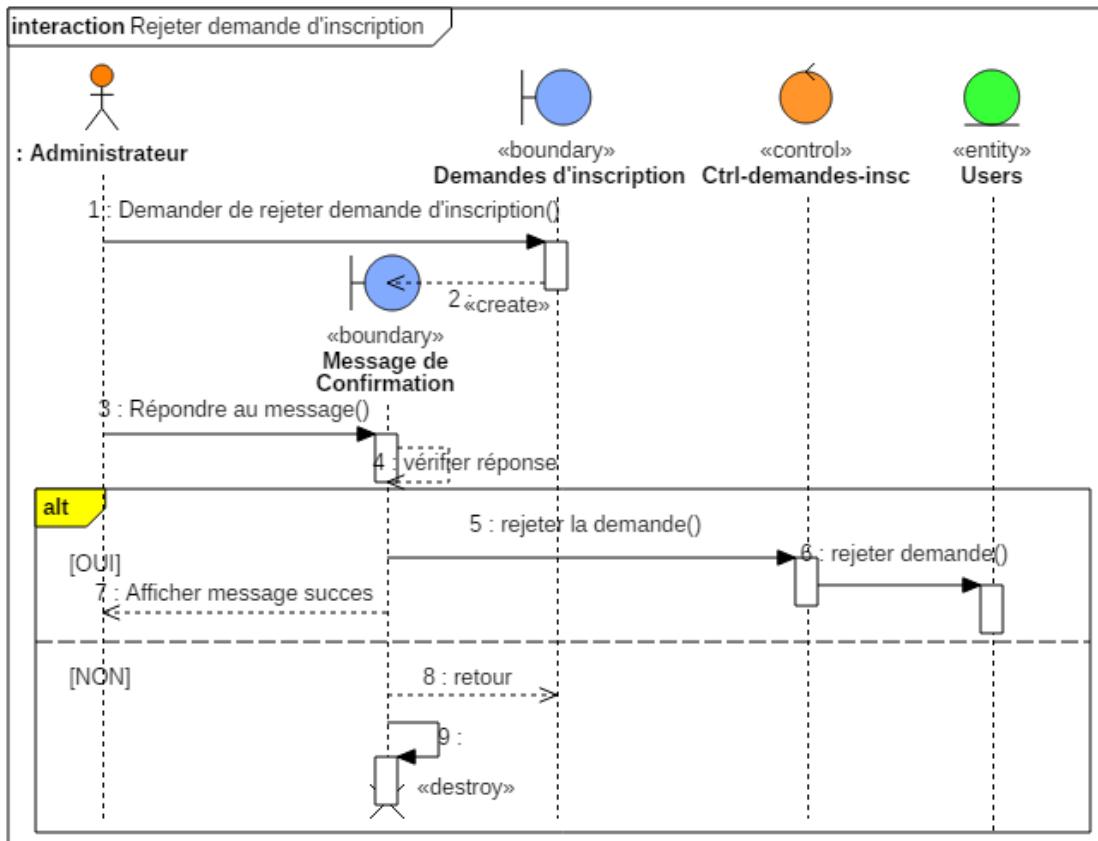


Figure 3.39. Diagramme de séquence détaillé du cas « Rejeter demande d'inscription ».

## 5.2. Diagrammes de classes de conception

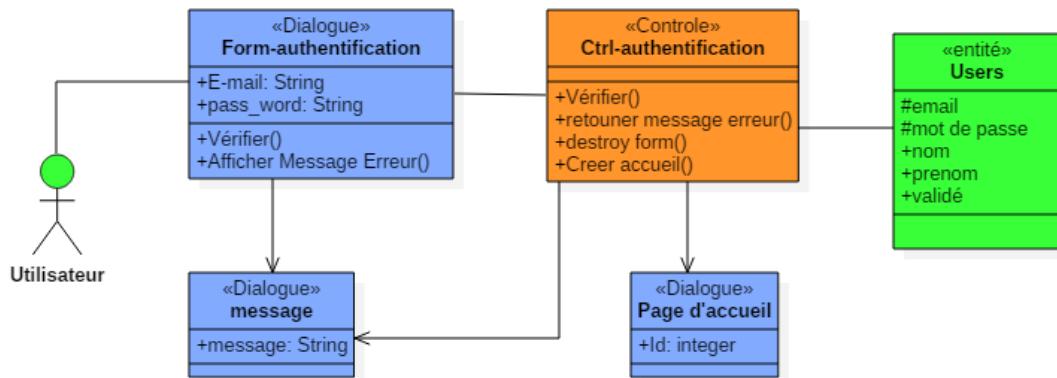


Figure 3.40. Diagramme de classe du cas « S'authentifier ».

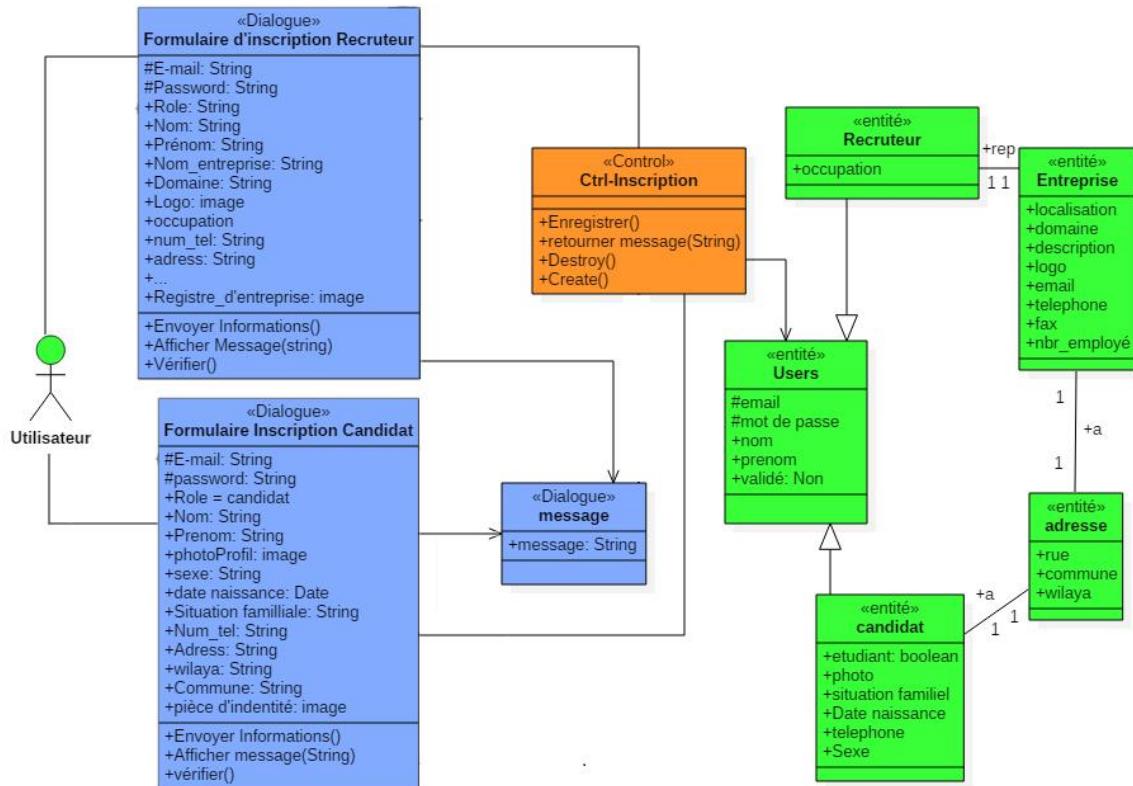


Figure 3.41. Diagramme de classe du cas « Incrire ».

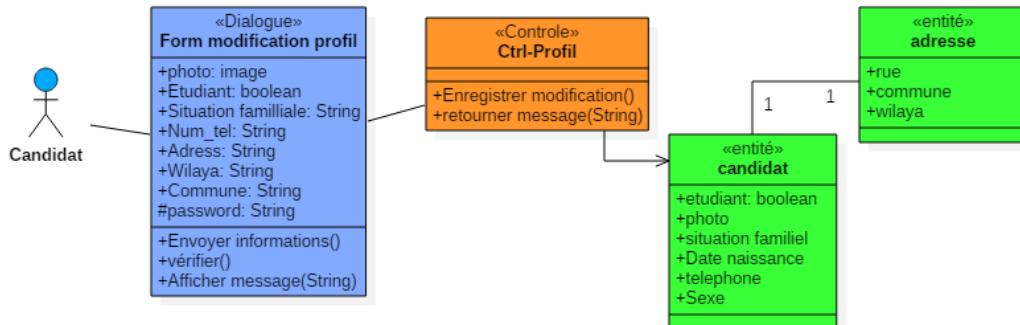


Figure 3.42. Diagramme de classe de conception du cas « Gérer profil ».

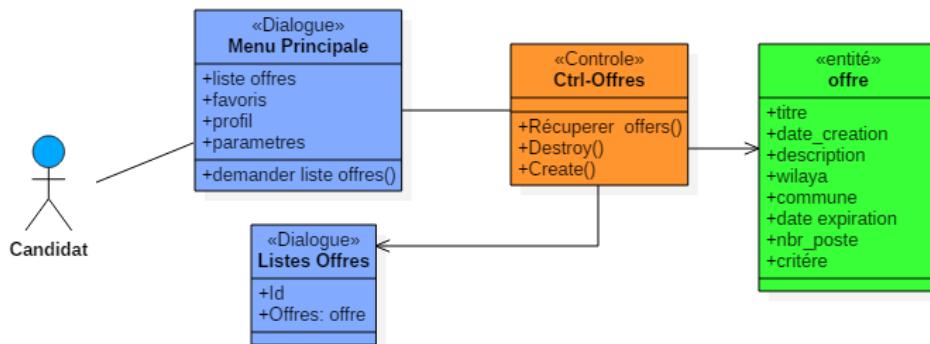


Figure 3.43. Diagramme de classe de conception du cas « Consulter liste offres ».

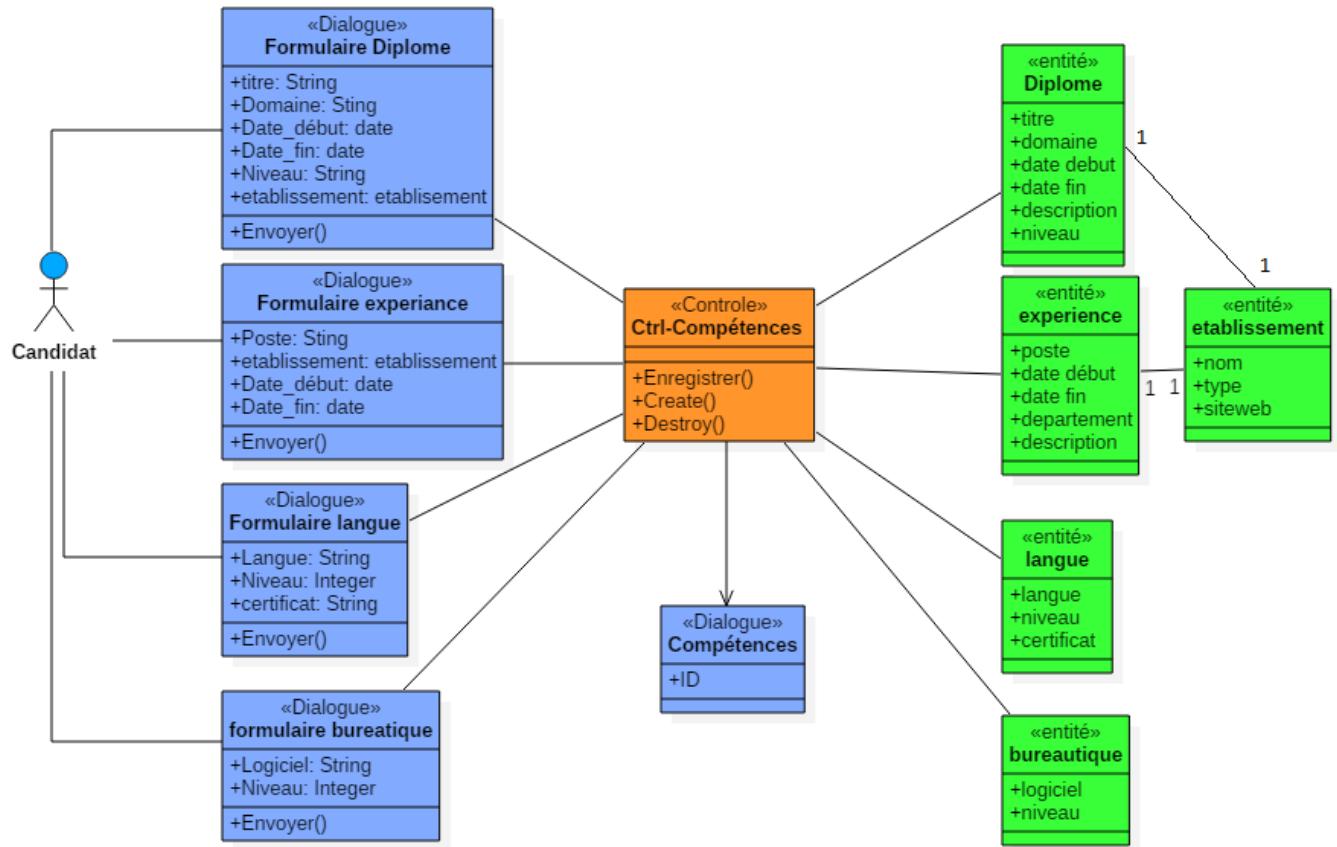


Figure 3. 44. Diagramme de classe de conception du cas « Ajouter compétence ».

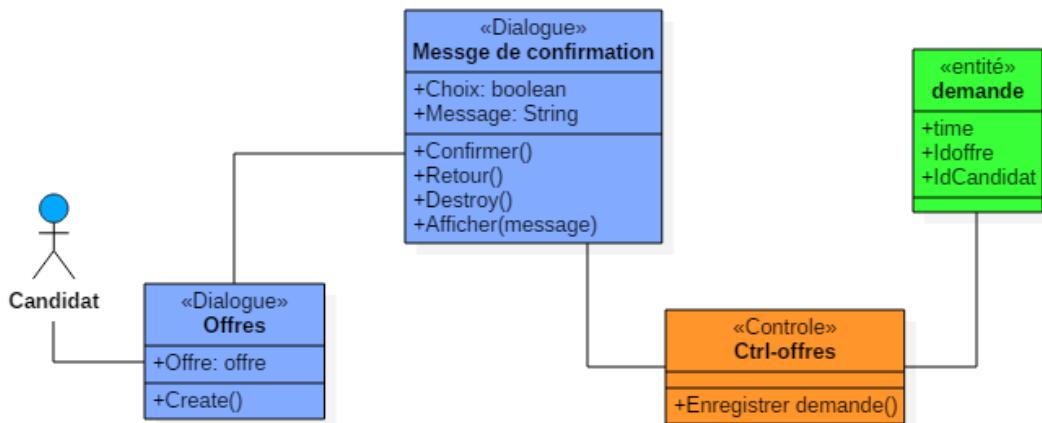


Figure 3. 45. Diagramme de classe de conception du cas « Postuler pour une offre ».

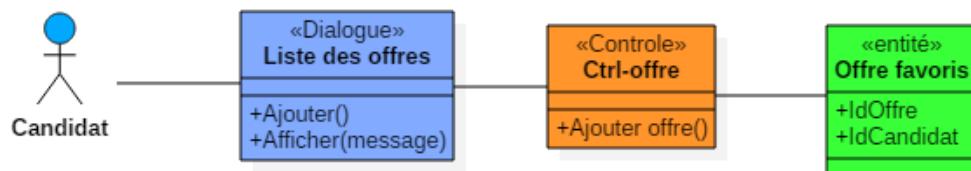


Figure 3. 46. Diagramme de classe de conception du cas « Ajouter offre aux favoris ».

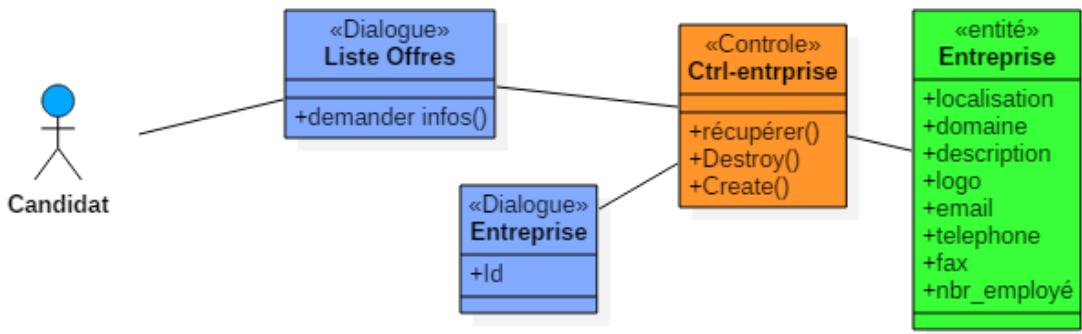


Figure 3.47. Diagramme de classe de conception du cas « Consulter informations d'entreprise ».

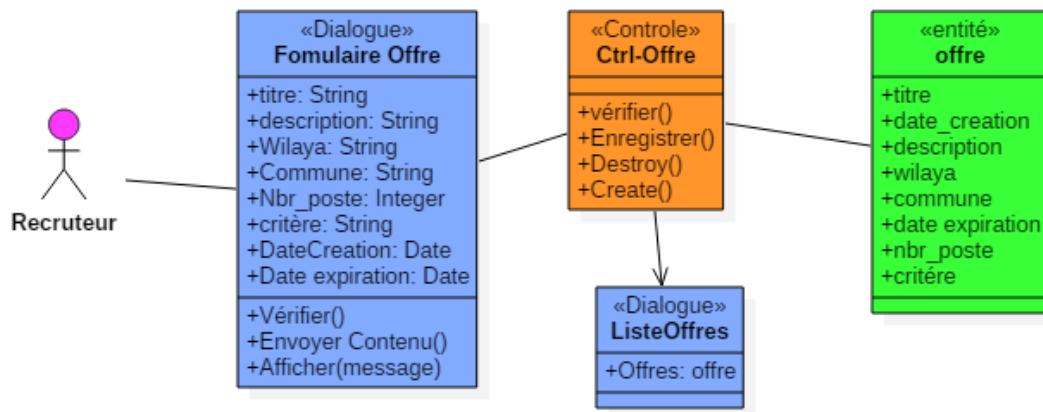


Figure 3.48. Diagramme de classe de conception du cas « Publier offre ».

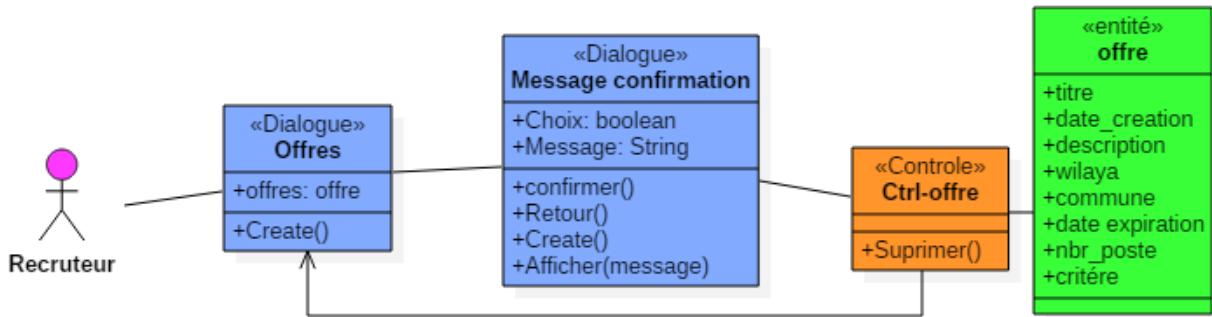


Figure 3.49. Diagramme de classe de conception du cas « Supprimer offre ».

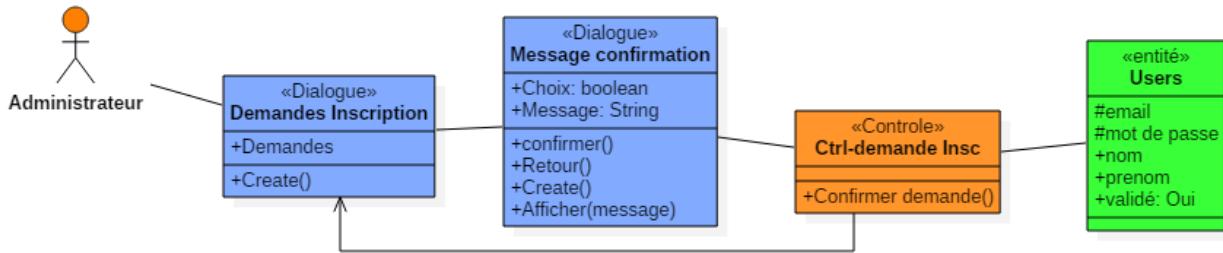


Figure 3. 50. Diagramme de classe de conception du cas « Confirmer une demande d'inscription ».

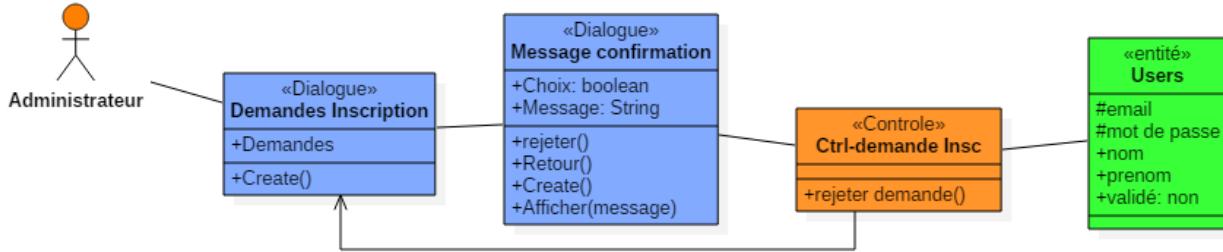


Figure 3. 51. Diagramme de classe de conception du cas « Rejeter une demande d'inscription ».

## 6. Conclusion

Dans ce chapitre nous avons présenté le résultat de l'analyse et la conception de notre projet. Ce résultat va nous permettre d'entamer l'étape de l'implémentation qui est l'objet du chapitre suivant.

# **CHAPITRE 04**

---

## **L'implémentation de l'application**

---

## 1. Introduction

Pour Réaliser nos objectifs, il est nécessaire de choisir les outils et les technologies permettant de les réaliser. Pour cela, après avoir complété l'étude conceptuelle dans le chapitre précédent, nous allons dans ce qui suit abordait la partie implémentation.

## 2. Description de l'environnement du travail

### 2.1. Plateformes matérielles

PC 1 : Dell Inspiron 3521		
Composant	Nom et version	Capacité
Processeur	Intel Core i5 3317U	2 * 1.70 GHz
RAM	DDR3	8 GB @ 665MHz
Disque Dur	ADATA SP550 (SSD)	128 GB
Carte Graphique	AMD Radeon HD 7670M	1 GB
Système d'Exploitation	Windows 10 Pro 64-bit	/

Smartphone 1 : Meizu M2		
Composant	Nom et version	Capacité
Chipset	Mediatek MT6735 (Cortex-A53 + Mali-T720MP2)	4* 1.3 GHz
RAM	LPDDR3	2GB
Mémoire Interne	UFS	16 GB
Système d'Exploitation	Android 5.1 (Lollipop) + Surcouche Flyme 6.0	/
Connectivité	2G,3G,4G,Wi-Fi, Bluetooth, GPS	/
Écran d'affichage	IPS LCD Capacitif tactile	5.0 pouces, Resolution de 720 x 1280 p

Smartphone 2 : Doogee T6		
Composant	Nom et version	Capacité
Chipset	MediaTek MT6735P (Cortex-A53 + Mali-T720MP2)	4* 1.0 GHz
RAM	LPDDR3	2GB
Mémoire Interne	UFS	16 GB
Système d'Exploitation	Android 5.1 (Lollipop)	/
Connectivité	2G,3G,4G,Wi-Fi, Bluetooth, GPS	/
Écran d'affichage	IPS LCD Capacitif tactile	5.5 pouces ,Resolution de 720 x 1280 p

## 2.2. Environnements logiciels

Outil	Version	Description
 <b>Android Studio</b>	Stable 2.3.2	L'IDE officiel pour le développement d'applications Android.
 <b>Android SDK</b>	Platforme 26	Le kit de développement (SDK) d'Android
 <b>Genymotion</b>	v2.9 free Edition	Un émulateur Android pour Windows, plus rapide que l'émulateur intégré avec le Android SDK

	1.12.1	Un éditeur de code source orienté application Web et Cloud.
	3.0.6 64bit	Plate-forme de développement Web sous Windows, contient un serveur Apache2, PHP et d'une base de données MySQL.
	4.6.4	Une interface d'administration pour SGBD MySQL. Il est écrit en langage PHP et s'appuie sur le serveur HTTP Apache.
	2.10.2	Un logiciel de gestion de versions décentralisé
	CC 2015 Release	Un logiciel de retouche, de traitement et de dessin assisté par ordinateur.

 <b>StarUML</b>	2.8.0	Un outil de modélisation UML.
---	-------	-------------------------------

### 2.3. Langages utilisés

**Java** : utilisé comme langage de programmation principale pour la logique de l'application.

**XML** : langage de balisage extensible, est un métalangage informatique de balisage générique, cette syntaxe est dite « extensible » car elle permet de définir différents espaces de noms, c'est-à-dire des langages avec chacun leur vocabulaire et leur grammaire, il est utilisé par Android studio pour décrire les Ressources comme les fichier Layouts (les vues), les ressources texte, les thèmes etc.

**JSON** : un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme le permet XML, il est utilisé pour échanger les données entre le client et le serveur car il est très léger.

**PHP** : est un langage de script côté serveur conçu principalement pour le développement web, mais aussi utilisé comme langage de programmation à usage général. Nous avons utilisé PHP dans le côté de serveur de notre application.

**SQL** : (sigle de StructuredQueryLanguage, en français langage de requête structurée) est un langage informatique normalisé servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.

## 3. Choix technique d'implémentation

### 3.1. Implémentation en modèle MVC

Notre application Android est organisée selon le modèle MVC :

- La couche Modèle est constituée d'objets Java.
- La couche Vue est constituée de fichier Layouts en XML.
- La couche Contrôle est constituée des classes activity et fragments.

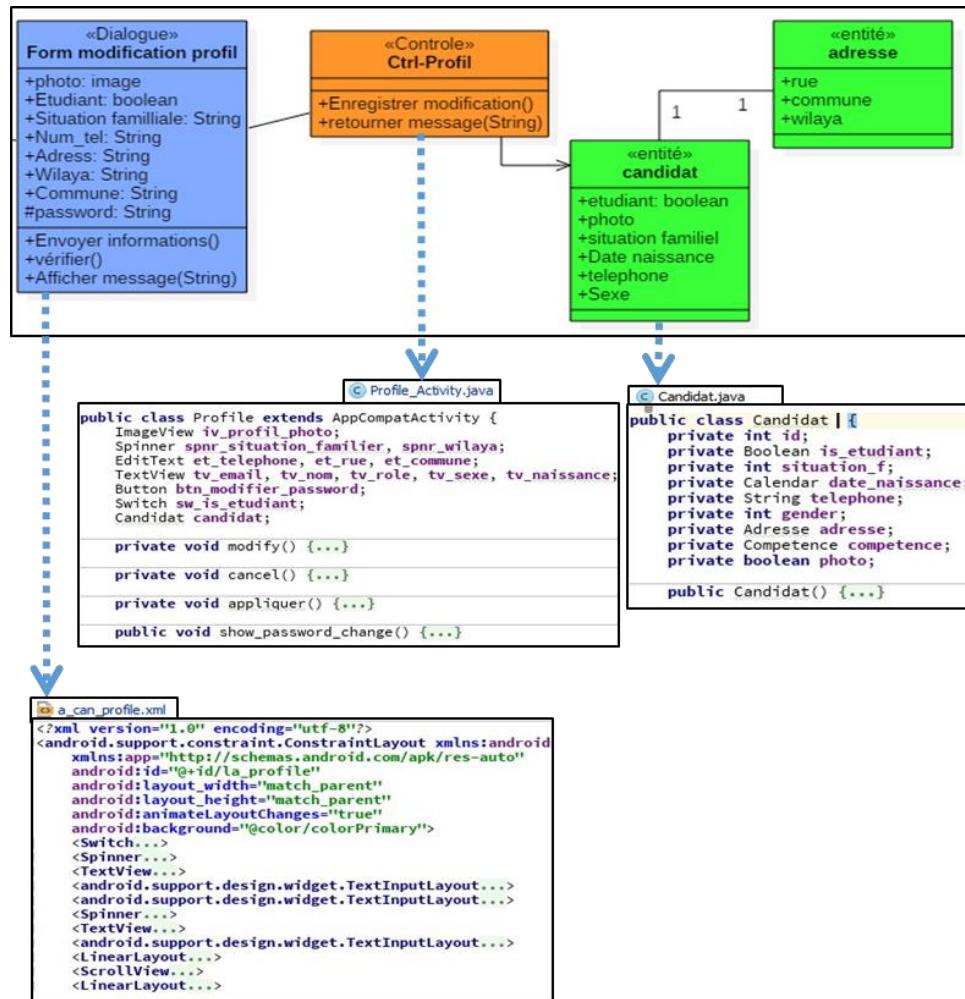


Figure 4. 1. Passage de conception, vers Code selon Modèle MVC.

### 3.2.L'architecture de l'application

La connexion d'un périphérique Android à une BDD distante n'est pas prévue (ni souhaitable), la méthode utilisée pour se connecter à la BDD MySQL à distance à partir d'un appareil Android, est de mettre en place un serveur qui contient des scripts PHP et d'exécuter ces scripts en utilisant le protocole http.

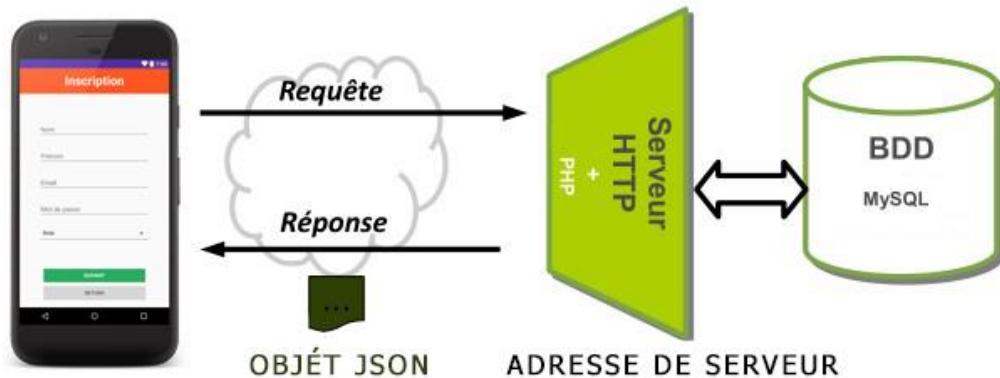


Figure 4. 2. Architecture de notre application.

### 3.3. Cryptage des données

Pour une sécurité optimale nous avons utilisé le cryptage pour sécuriser notre application et ces communications. Par défaut Android sécurise les données locales de chaque application dans un répertoire inaccessible à un utilisateur normal (il faut Rooter l'appareil pour les manipuler). Dans notre cas les données sont généralement des fichiers temporaires générés par Android pour la performance et des valeurs de configurations (pas de BDD locale). Par conséquent, on a crypté les communications entre l'application et le serveur là où il y a des informations sensibles faciles à lire. Nous avons utilisé comme algorithme AES128Bit pour le cryptage des objets JSON.

Pour les mots de passe, ils sont sauvegardés dans la BDD après une opération de hashage avec l'algorithme SH1, utilisant Salt qui est la date de naissance pour plus de sécurité.

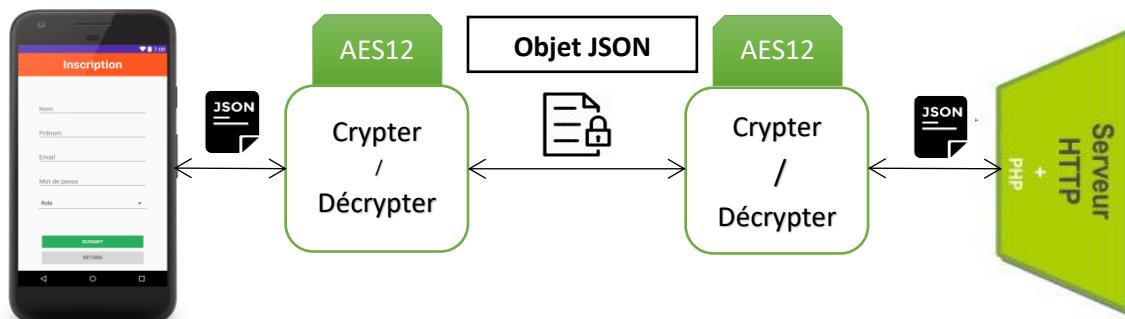


Figure 4. 3. Fonctionnement de cryptage dans notre application.

### Les Documents Attaché :

Lors de la vérification du compte est terminé par l'administrateur, les documents sensibles attachés par l'utilisateur seront automatiquement supprimés du serveur, pour respecter la confidentialité.

#### 3.4. Les bibliothèques utilisées

- **AESCrypt-Android**

Nom de package : com.scottyab:aescrypt

Rôle : API pour effectuer le cryptage AES sur Android.

- **Calligraphy**

Nom de package : uk.co.chrisjenx:calligraphy

Rôle : Permet d'utiliser les polices personnalisées dans Android

- **Gson**

Nome de package : com.google.code.gson:gson

Rôle : Utilisé pour convertir des objets Java en leur représentation JSON.

#### 3.5. Le Choix de niveau de l'API :

Chaque application Android doit spécifier le niveau d'API minimum requis pour que l'application s'exécute. Ce choix est un compromis entre l'utilisation de nouvelles fonctionnalités de l'API et le support pour plus d'utilisateurs, dans notre application on a choisi comme niveau minimal la version 5.0 (Lollipop API level 21), ce qui indique que notre application peut fonctionner sur 71.5% des appareils Android dans le monde, selon les statistiques du Google (juin 2017).

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.8%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.8%
4.1.x	Jelly Bean	16	3.1%
4.2.x		17	4.4%
4.3		18	1.3%
4.4	KitKat	19	18.1%
5.0	Lollipop	21	8.2%
5.1		22	22.6%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	8.9%
7.1		25	0.6%

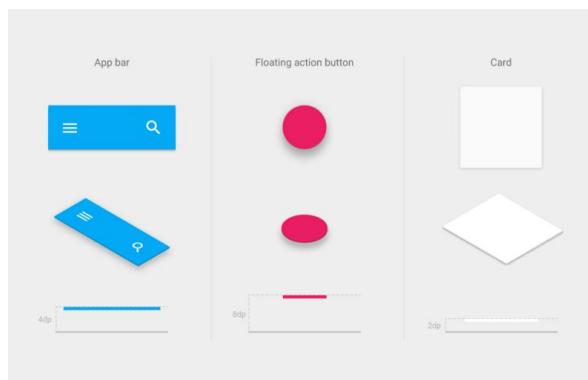
**Figure 4. 4.** Distribution globale des versions Android juin 2017.

### 3.6. Proguard :

**Proguard** est un optimiseur pour le bytecode Java. Cela rend l'application Android jusqu'à 90% plus petites et jusqu'à 20% plus rapidement. ProGuard offre également une protection minimale contre l'ingénierie inverse (reverse engineering) en changeant les noms des classes, des champs et des méthodes.

### 3.7. Material design

Material design Est un ensemble de règles de design proposées par Google et qui s'appliquent à l'interface graphique des logiciels et applications. Il est utilisé notamment à partir de la version 5.0 du système d'exploitation Android.



**Figure 4. 5.** Exemple de contrôles de Material Design.

### 3.8. Constraint Layout :

Constraint Layout (introduit en février 2017) vous permet de créer des mises en page étendues et complexes avec une hiérarchie de vue plate (pas de groupes de vues imbriquées). Ça améliorait la performance.

## 4. Description des interfaces de l'application



Figure 4. 6. Interface de l'authentification.

À partir de cette interface (Figure 4.6), si l'utilisateur est déjà inscrit, il pourra se connecter. Il suffit d'entrer son E-mail et son mot de passe et cliquer sur le bouton « Connexion » pour ouvrir son compte.

Bien évidemment, un nouvel utilisateur doit demander l'inscription. Les interfaces de Figure 4.7 décrivent les étapes qui permettent au candidat de demander l'inscription. Figure 4.8 décrit les étapes de demande d'inscription du recruteur.

The figure consists of four screenshots of a mobile application interface for candidate registration, arranged horizontally. Each screenshot shows a different step in the process:

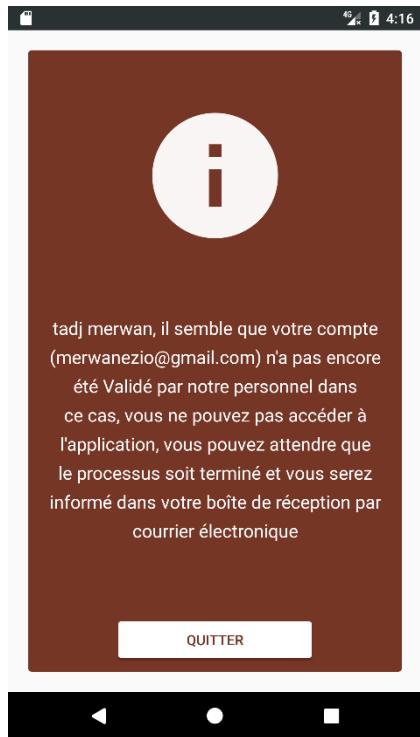
- Inscription:** Contains fields for Nom, Prénom, Email, Mot de passe, and Role. Buttons for SUIVANT and RETURN are at the bottom.
- Profile:** Features a placeholder for a photo labeled "PHOTO". Below it are gender selection buttons (Male/Female), a date of birth field, and a dropdown for Familiar situation. Buttons for SUIVANT and RETURN are at the bottom.
- Adresse:** Includes fields for Téléphone, Adresse, Wilaya, and Commune. A note in a red box at the top right says: "Pour éviter le spam et les inscriptions indésirables, vous devez attacher une photo frontal de votre carte d'identité afin de vérifier votre identité." A placeholder for a "CARTE ID" document is shown with an "ATTACH" button below it. Buttons for SUIVANT and RETURN are at the bottom.
- Validation:** Shows a note about attaching a photo of an identity card. Below it is a placeholder for a "CARTE ID" document with an "ATTACH" button. Buttons for SUIVANT and RETURN are at the bottom.

Figure 4. 7. Interfaces de demande d'inscription de candidat.

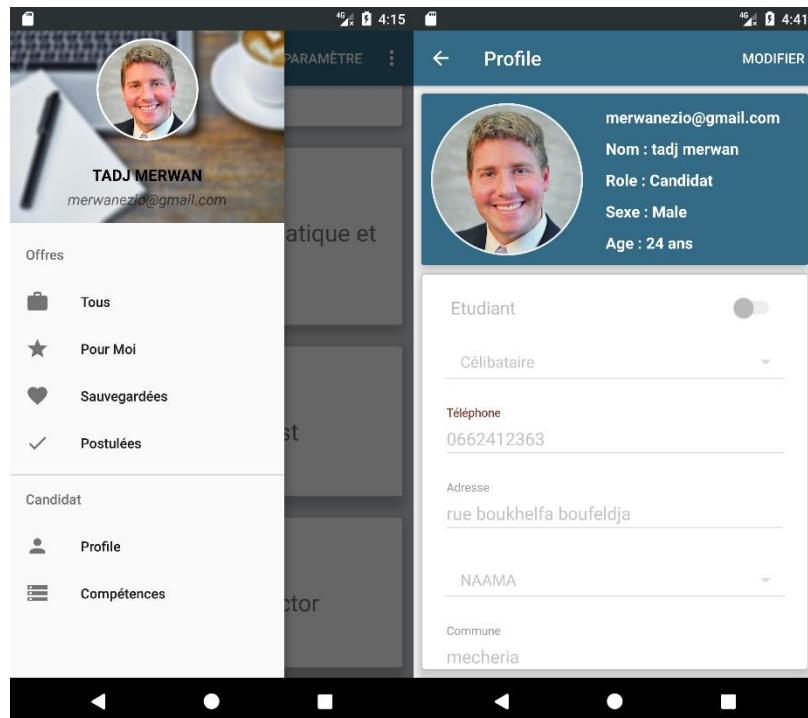
The figure consists of four screenshots of a mobile application interface for employer registration, arranged horizontally. Each screenshot shows a different step in the process:

- Inscription:** Contains fields for Nom, Prénom, Email, Mot de passe, and Role. Buttons for SUIVANT and RETURN are at the bottom.
- Entreprise:** Features a placeholder for a logo. Below it are fields for Nom, Domaine, and Type. Buttons for SUIVANT and RETURN are at the bottom.
- Contact:** Includes fields for Email, Téléphone, FAX, and Site Web. Below it are fields for Adresse, Wilaya, and Commune. A note in a red box at the top right says: "Pour éviter le spam et les inscriptions indésirables, vous devez attacher une photo du registre commercial ou un document similaire (montrer le nom et l'activité) pour vérifier votre entreprise." A placeholder for a "DOCUMENT" is shown with an "ATTACH" button below it. Buttons for SUIVANT and RETURN are at the bottom.
- Validation:** Shows a note about attaching a photo of a commercial register or similar document. Below it is a placeholder for a "DOCUMENT" with an "ATTACH" button. Buttons for SUIVANT and RETURN are at the bottom.

Figure 4. 8. Interfaces de demande d'inscription de recruteur.

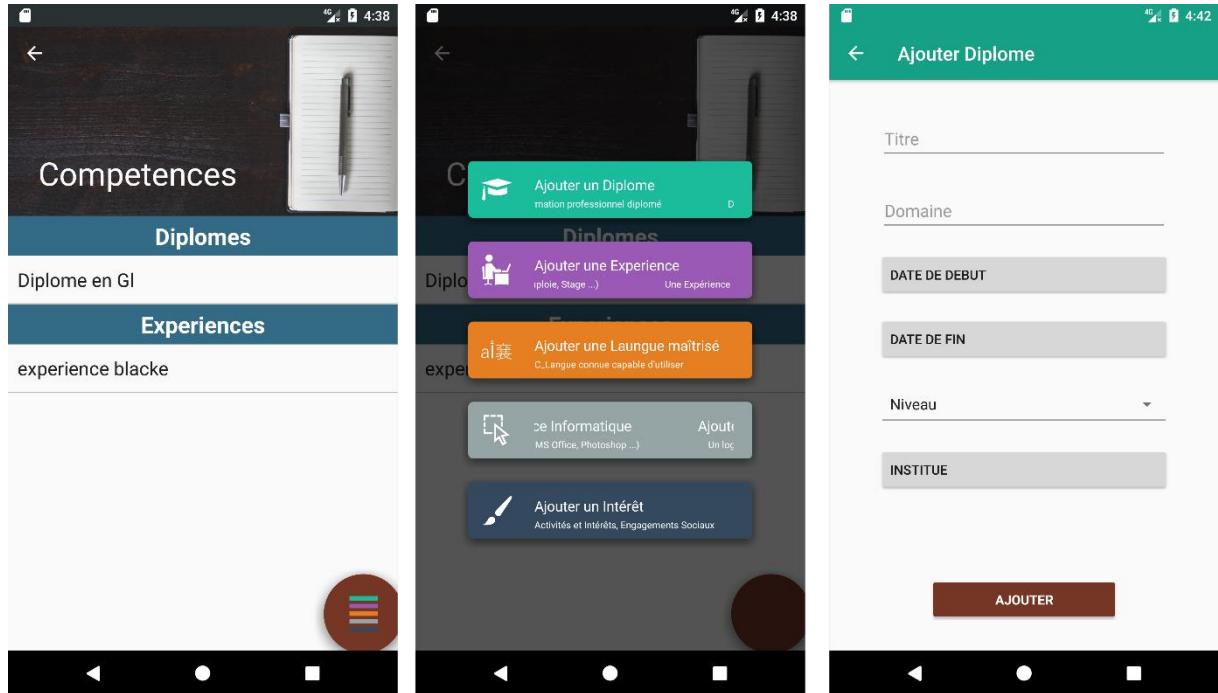


**Figure 4. 9.** Message pour les utilisateurs non validés.



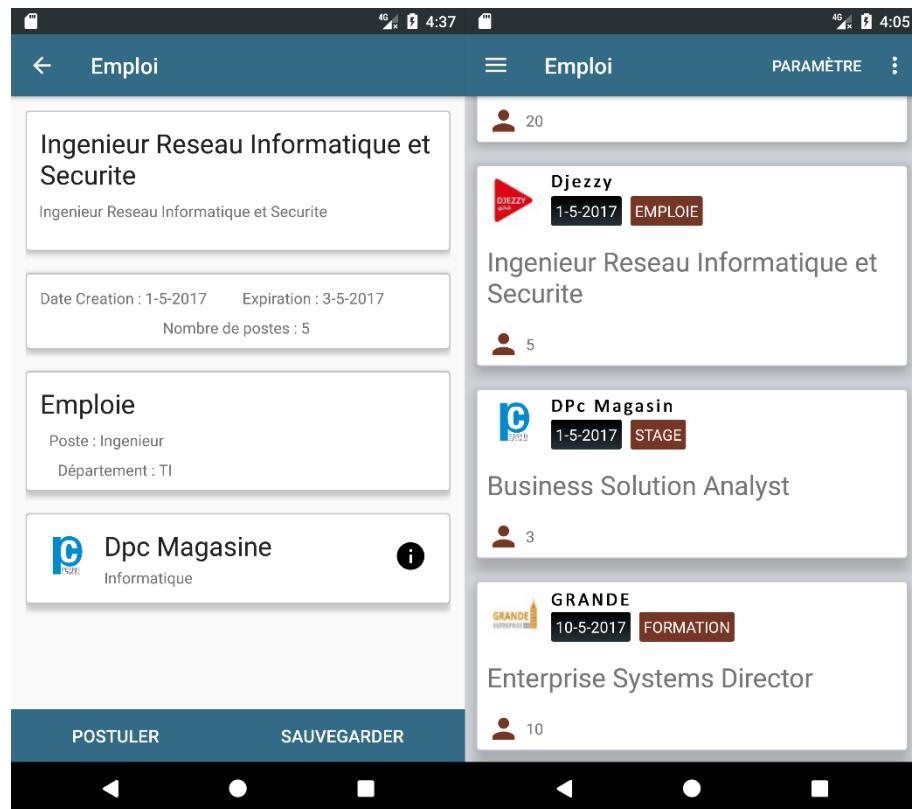
**Figure 4. 10.** Interfaces représentent le menu principal et le profil de candidat.

A partir de menu principal le candidat peut accéder à toutes les fonctionnalités, dans l'interface du profil, il peut gérer ces informations personnelles.



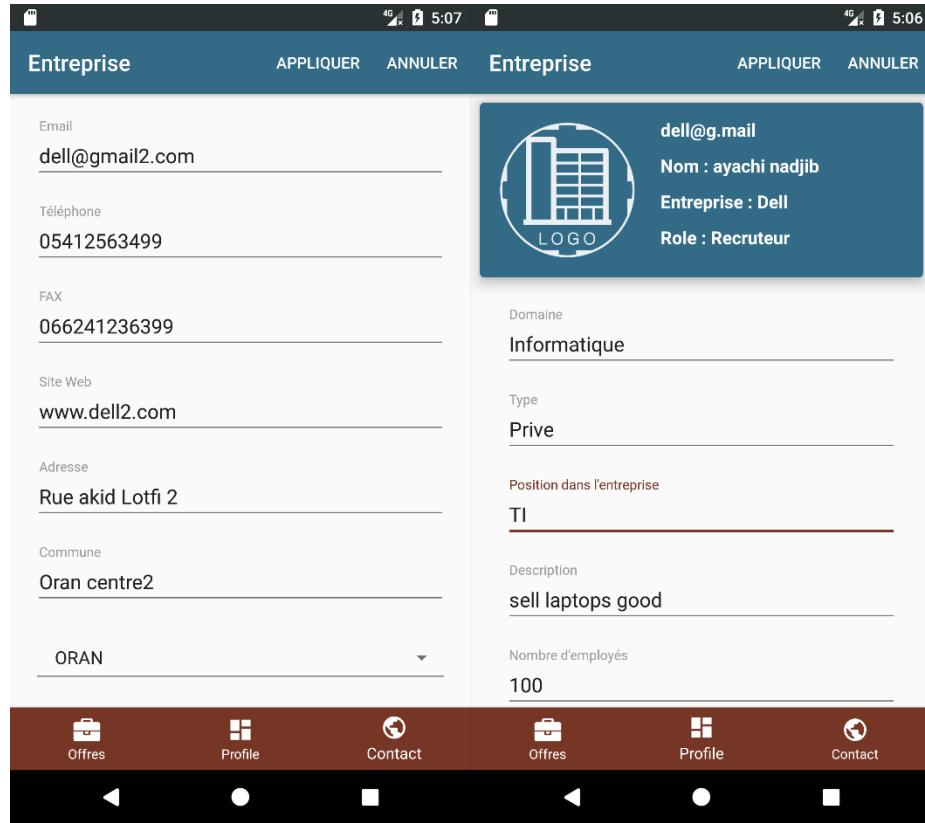
**Figure 4. 11.** Différentes interfaces des compétences (consulter, modifier, ajouter).

Ces interfaces permettent au candidat de gérer ses compétences.



**Figure 4. 12.** Interface présente la liste des offres et consulter les informations d'une offre.

à partir de ces interfaces le candidat peut visualiser la liste des offres et voir leur détails avec la possibilité de postuler ou d'ajouter aux favoris.



**Figure 4. 13.** Interfaces de profil de recruteur (informations d'entreprise).

D'après ces interfaces le recruteur peut gérer les informations d'entreprise.

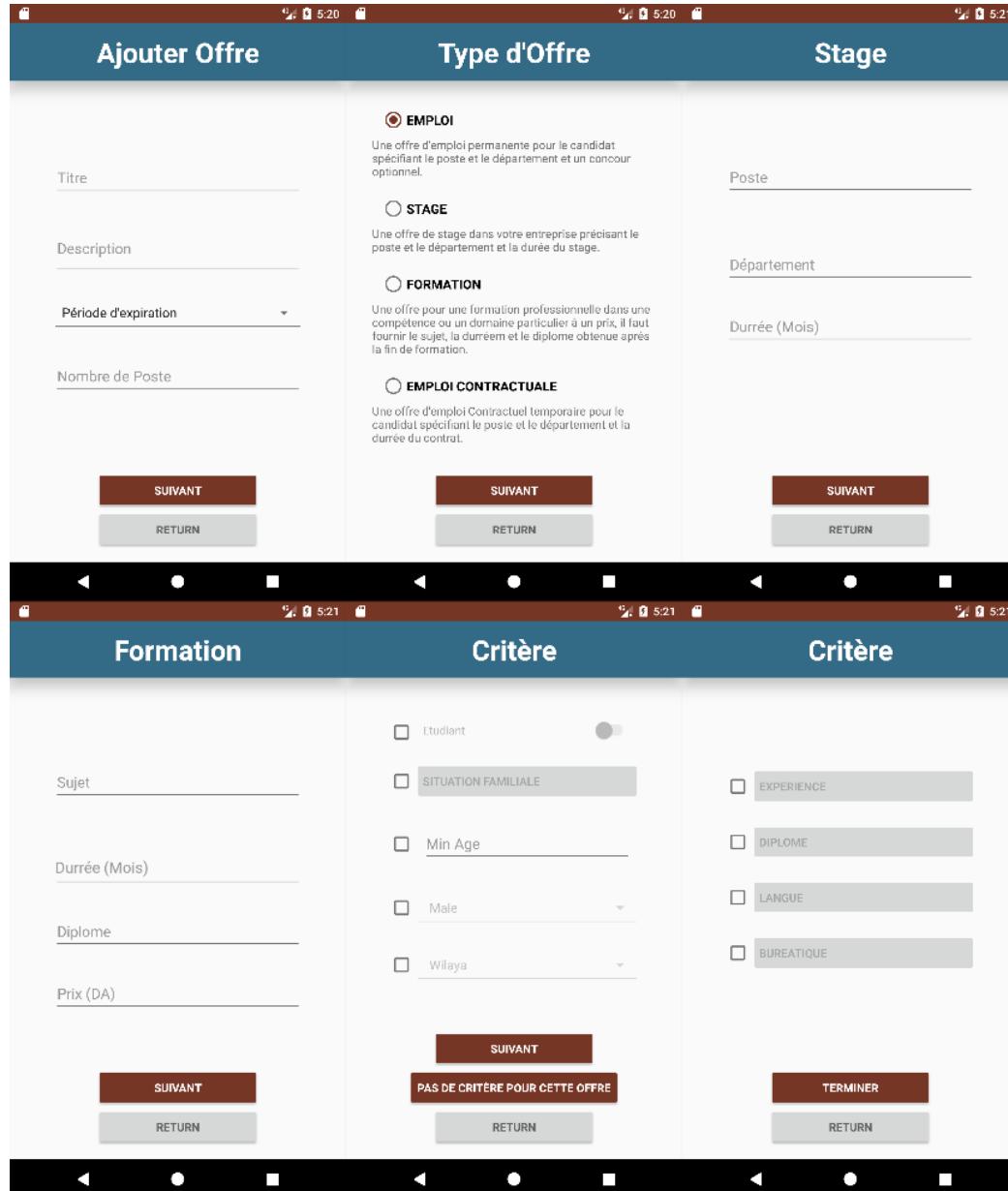


Figure 4. 14. Interface décrit la fonctionnalité publier une offre.

Ces interfaces (Figure 4.14) décrivent les différentes étapes permettant au recruteur de publier une nouvelle offre.

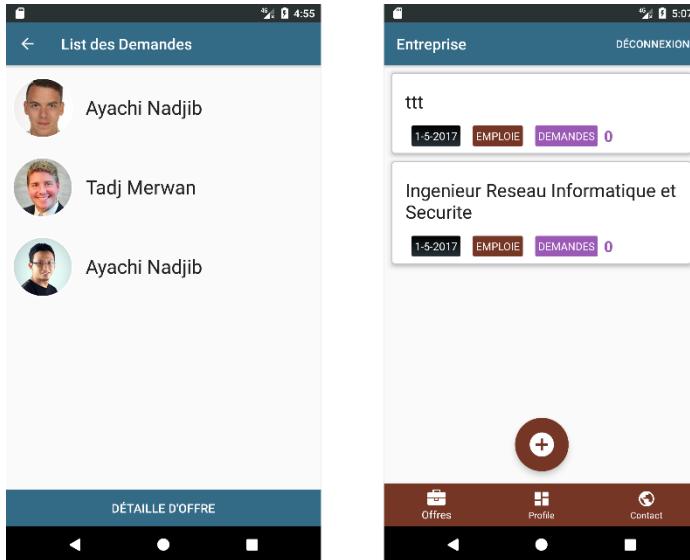


Figure 4. 15. Interfaces présentent la liste des offres et les demandes

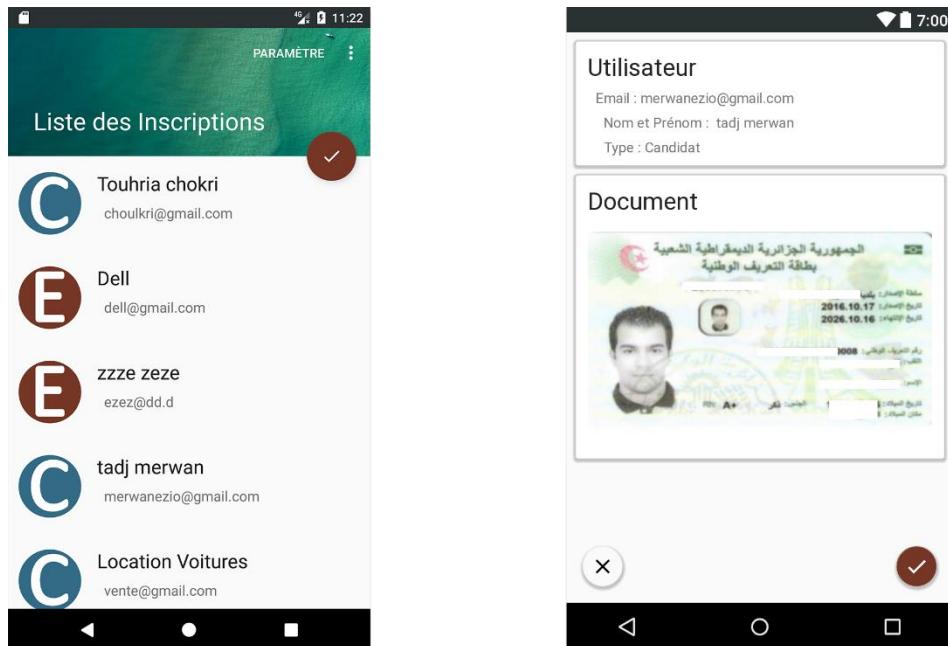


Figure 4. 16. Interface liste de demandes d'inscription et exemple de vérification.

L’administrateur peut voir les nouvelles demandes d’inscription non validées, puis faire une vérification pour assurer que les informations des demandeurs sont valables.

## 5. Conclusion

Dans ce chapitre, nous avons présenté l’environnement matériel et logiciel, les langages et outils utilisés, les différents choix d’implémentation et la structure de l’application. On a terminé ce chapitre par quelques interfaces graphique de notre application.

---

## **Conclusion générale et perspectives**

---

## Conclusion générale et perspectives

Ce projet représente le fruit de 5 ans d'études, au sein du département Technologie de Logiciels et Système d'Information (TLSI). Dans ce travail, nous avons essayé d'appliquer les notions théoriques et les pratiques acquises au cours de ce cycle didactique relatives à la discipline du génie logiciel.

Le but de ce projet est de nous permettre de maîtriser certains aspects du développement mobile, en particulier la modélisation (l'analyse et conception) et l'implémentation, à travers la réalisation d'une application Android pour le rapprochement des entreprises et des demandeurs d'emplois et de stages.

Pour le développement de l'application nous avons utilisé des technologies et des techniques récentes et performantes tel que l'approche Native pour les applications Android. Nous avons également utilisé le modèle MVC (à travers l'approche de développement de Pascal Roques) pour la modélisation et l'implémentation de notre projet pour arriver à un code source propre, lisible, bien arrangé, extensible et maintenable. Sans oublier le côté de la sécurité et de la confidentialité des utilisateurs, on a appliqué la technique de cryptage des données (AES) et de Hachage (SHA1) pour chiffrer toute information circulant entre l'application et le serveur de base de données.

L'application que nous avons développée pourrait être enrichie par plusieurs fonctionnalités avancées telles que :

- Ajouter une fonctionnalité permettant de recommander une offre de la part d'un candidat à un autre, ainsi la disposition d'un espace d'échange d'idées.
- Enrichir la fonctionnalité permettant de gérer les compétences en ajoutant l'option de l'attachement de CV du candidat sous différents formats (par exemple le format PDF).
- Ajouter une fonctionnalité permettant aux recruteurs d'explorer et de rechercher les profils adéquats aux offres proposées.
- Améliorer la fonctionnalité permettant la consultation des informations d'une entreprise, par l'ajout de la localisation géographique (intégration de Google Maps, etc.).

- Le traitement et le filtrage des demandes pour les offres est réalisés actuellement d'une manière manuelle par le recruteur ou le responsable d'entreprise. On compte rendre cette tache automatisée ou semi-automatisée par l'application.
- Utiliser la base de données Cloud (FireBase) pour enrichir notre application avec des nouvelles options (vérification de l'e-mail, authentification avec un autre réseau social par exemple Facebook, Google+, etc.)
- Ajouter une fonctionnalité permettant la diffusion des alertes RSS, pour gérer les annonces et les actualités.
- Déployer l'application dans un environnement réel (PlayStore).

Finalement, on espère que ce travail serait utile et bénéfique pour d'autres personnes, et que l'application soit exploitée.

## Bibliographie

- [1] P. Roques, *SysML par l'exemple, Un langage de modélisation pour systèmes complexes*, Eyrolles, 2009.
- [2] D. GUEYE, "Plateformes de services intégrés pour mobiles, Diplôme d'Ingénieur de Conception 2008," Université Cheikh Anta Diop de Dakar - , Dakar, 2008.
- [3] L. Audibert, *UML 2 - de l'apprentissage à la pratique*, éditions Ellipses , 2009.
- [4] P. Gérard, *Processus de Développement Logiciel*, Paris: Université de Paris 13, 2007/2008.
- [5] I. Sommerville, *SOFTWARE ENGINEERING*, Library of Congress Cataloging-in-Publication Data, 2011.
- [6] . B. Kent, B. Mike , B. Arie van, C. Alistair , C. Ward , F. Martin and G. James , *Manifesto for Agile Software Development*, 2001.
- [7] D. Wells, "Extreme Programming: A gentle introduction," Don Wells, [Online]. Available: <http://www.extremeprogramming.org/>. [Accessed 20 03 2017].
- [8] K. Beck and A. Cynthia, *Extreme Programming Explained: Embrace Change*, Second Edition, Addison Wesley Professional, 2004.
- [9] S. Ambler, *The Object Primer, Agile Model-Driven development with UML 2.0*, Cambridge University Press, 2004.
- [10] S. Ambler, "Introduction To AM," [Online]. Available: <http://www.agilemodeling.com/essays/introductionToAM.htm>. [Accessed 23 03 2017].
- [11] P. Roques, *Les cahiers du programmeur UML 2, Modéliser une application Web.*, Eyrolles, 2008.
- [12] wikiversity, "Applications pour mobiles : Introduction et définition," 20 03 2017. [Online]. Available: [https://fr.wikiversity.org/wiki/Applications\\_pour\\_mobiles/Introduction\\_et\\_d%C3%A9finition](https://fr.wikiversity.org/wiki/Applications_pour_mobiles/Introduction_et_d%C3%A9finition).
- [13] B.Bathelot, "Définition : Application mobile," 08 03 2017. [Online]. Available: <http://www.definitions-marketing.com/definition/application-mobile/>.
- [14] Tutorials Point, *Android application development*, Tutorials Point (I) Pvt.Ltd., 2014.
- [15] E. Frédéric, *Créez des Applications pour android, le Site de zéro*, 2013.

- [16] Google, "API Guide," [Online]. Available: <https://developer.android.com/guide/components/index.html>. [Accessed 20 04 2017].
- [17] G. Damien, C. Julien and R. Emmanuel, *Programmation Android: De la conception au déploiement avec le SDK Google Android 2*, Edition Eyrolles, 30 avril 2010.
- [18] J. P. Cardle, *Android App Development in Android Studio.*, Manchester Academic Publishers, 2017.
- [19] servicesmobiles.fr, "Développement d'applications mobiles : web, natif, hybrides," 2016. [Online]. Available: <http://www.servicesmobiles.fr/developpement-dapplications-mobiles-web-natif-hybrides-32198/>. [Accessed 20 04 2017].
- [20] J. Summerfield, 2016. [Online]. Available: <https://www.hwsolutions.com/services/mobile-web-development/mobile-website-vs-apps/>. [Accessed 20 04 2017].
- [21] generationmobiles.net, 2014. [Online]. Available: <http://generationmobiles.net/2014/11/les-differents-types-dapps-mobiles-leurs-avantages-et-inconvenients/>. [Accessed 20 04 2017].
- [22] C. d. R. Dujardin, 2013. [Online]. Available: <http://www.journaldunet.com/developpeur/expert/53610/application-mobile-native--web-ou-hybride---6-points-a-considerer-pour-faire-le-bon-choix.shtml>. [Accessed 20 04 2017].
- [23] O. L. Goaer, *Programmation des systèmes mobiles & sans fil [Android]*, Developpez.com, 2016.
- [24] H. Sylvain and P. Sébastien, *Android: Guide de développement d'applications Java pour SmartPhone et Tablettes*, 2ième édition, France: Edition ENI, 2014.
- [25] F. Champigny, "Introduction aux Fragments," 11 08 2015. [Online]. Available: <http://tutos-android-france.com/fragments/>. [Accessed 20 04 2017].