République Algérienne Démocratique et Populaire
وزارة التعليـم العالـي والبحـث العلمـي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Nº Réf :……

Centre Universitaire
Abd Elhafid Boussouf Mila

Institut des Sciences et Technologie Département de Mathématiques et Informatique

Mémoire préparé en vue de l'obtention du diplôme de Master

En : **Mathématiques**
Spécialité : Mathématiques fondamentales et appliquées
ou
En : Informatique

Spécialité: Sciences et Technologies de l'Information et de la Communication (STIC) .

Réalisation d'une application basée agents pour la réservation de taxi .

Préparé par : Djakour Safa .
Yessad Khatima.

Soutenue devant le jury :

Encadré par :	**Mr** Djaaboub Salim.	C.U.Abd Elhafid Boussouf.
Président :	**Mme** Nardjes Bouchemal.	C.U.Abd Elhafid Boussouf.
Examinateur :	**Mr** Hettab Abdelkamel.	C.U.Abd Elhafid Boussouf .

Année Universitaire : 2020/2021.

Remerciement

Nous remercions d'abord et avant tout Allah qui nous a donné le courage, la santé, la possibilité et la patience pour réaliser ce travail.

Un remerciement particulier à notre encadreur Monsieur **DJAABOUB SALIM** pour son soutient, son sérieux, sa disponibilité, ses précieux conseils et son aide tout au long de l'élaboration de ce travail.

Nous remercions également, les membres du jury d'avoir accepté d'examiner et d'évaluer notre travail.

Sans oublier tous les enseignants du département d'informatique pour la qualité de l'enseignement qu'ils ont bien voulu nous prodiguer durant nos études afin de nous fournir une formation efficiente.

Nous n'aurions garde d'oublier tous ceux qui ont contribué de près ou de loin à la réalisation de ce mémoire et à tous ceux qui ont partagé avec nous les moments les plus difficiles dans la réalisation de ce travail et tous ceux qui nous souhaite le bon courage.

Finalement, nous remercions très sincèrement tous nos familles pour leur encouragement sans limite.

SAFA et KHATIMA

Dédicace

À

Les personnes les plus adorables du monde

Pour la plus belle maman ***ZAHIA,*** et le meilleur papa Mohamed, je veux vous dire " Je suis ce que je suis aujourd'hui à cause de toi je suis reconnaissant pour votre amour, prières, soutien et les précieux conseils. Et J'espère que vous continuerez à guider mes pas de ma vie ".

Mes belles sœurs : ***HALIMA*** , ***SARA*** et ***SAMEH*** .

Mon cher frère : ***WALID*** .

Mon cher Oncle *:* ***FATEH .***

Mon fiancé : ***YACINE LAAMARA*** .

Mes belle amies : ***NESRINE*** , ***SAFA*** , ***AMIRA*** et ***FATIMA*** .

À mes nièces : ***NIHAL, IBTIHEl*** et ***HIDAYA***.

À mes neveux : ***TAKI, ADEM***, ***YOUCEf*** et ***YAKIN***.

À tous les membres de la famille **YESSAD, LAAMARA.**

À toute la promotion 2020/2021 Informatique ET Enfin mon plus profond respect va tout droit à mes aimables professeurs dans Tous les cycles de ma scolarité qui mon éclairé la voie du savoir .

Khatima.

Dédicace

À

La lumière de mes yeux ma très chère mère **WARDA**, qui me guider, m'inspirer et qui est ma donnée un magnifique modèle de labeur et de persévérance.

Mon très chère père **RAMDANE**, qui a toujours été là pour moi, qui a su me protéger, guider mes pas, m'éclairer avec ses conseils avisés.

Mon marie : **BOUSSOUF MALIK.**

Mes chères sœurs : **AMIRA, MOUNA**.

Mon cher frère : **YOUCEF**.

À mes belle amis : **AISSOUS KHAOULA, REMOUGUI BASMA CHAYMA, ASSIA BENKHANOUCHE.**

À mes nièces : **RAMA, ANFEL.**

À mon neveu : **MOHAMED**.

À tous mes collègues du Centre Universitaire de Mila, Mon amie, collègue, et binôme **KHATIMA** qui a partagé avec moi les moments difficiles de ce travail. À tous les membres de la famille **DJAKOUR**, **BOUSSOUF**.

SAFA.

Résumé :

L'émergence de la technologie des téléphones mobiles et les développements technologiques dans le domaine de l'information et de la communication ont conduit à l'apparition de différentes applications mobiles utiles pour différents domaines comme la réservation en ligne de taxi. Les applications de réservation en ligne permettent aux utilisateurs de trouver et de réserver des taxis à tout moment et n'importe où en utilisant leurs appareils mobiles. Dans ce travail nous avons développé une application mobile basée agent qui va permettre aux utilisateurs en Algérie de bénéficier de ces nouvelles technologies dans le domaine de réservation de taxis. L'application réalisée, va donner aux chauffeurs la possibilité de créer et de gérer facilement des comptes, d'ajouter des services, de de proposer leurs services aux clients. Pour les clients, l'application offre plusieurs fonctionnalités comme la recherche de taxis, la géolocalisation automatique, la réservation de taxis, etc. l'application a été développée en utilisant plusieurs méthodes et outils de la technologie mobile et de systèmes multi-agents. .

Mots-clés : Applications mobiles, réservation de taxis, agents, systèmes multi-agents, SMA.

Abstract :

The emergence of mobile phone technology and technological developments in the field of information and communication have led to the emergence of different mobile applications useful for different fields such as online taxi booking. Online booking applications allow users to find and book taxis anytime and anywhere using their mobile devices. In this work we have developed an agent-based mobile application that will allow users in Algeria to benefit from these new technologies in the area of taxi reservation. The realised application will give drivers the ability to easily create and manage accounts, add services, and offer their services to customers. For customers, the application offers several features such as taxi search, automatic geolocation, taxi reservation, etc. The application has been developed using several methods and tools from mobile technology and multi-agent systems.

ملخص :

أدى ظهور تكنولوجيات الهاتف المحمول والتطورات التكنولوجية في مجال المعلومات والاتصالات إلى انتشار تطبيقات مختلفة للهاتف المحمول مفيدة لمجالات مختلفة مثل حجز سيارات الأجرة عبر الإنترنت. تتيح تطبيقات الحجز عبر الإنترنت للمستخدمين العثور على سيارات الأجرة وحجزها في أي وقت وفي أي مكان باستخدام أجهزتهم المحمولة. في هذا العمل ، طورنا تطبيقًا للهاتف المحمول يعتمد على الوكيل والذي سيسمح للمستخدمين في الجزائر بالاستفادة من هذه التقنيات الجديدة في مجال حجز سيارات الأجرة. سيمنح التطبيق المنجز السائقين القدرة على إنشاء حسابات وإدارتها بسهولة وإضافة خدمات وتقديم عروضهم للزبائن. بالنسبة للزبائن، يقدم التطبيق العديد من الميزات مثل البحث عن سيارات الأجرة، التحديد التلقائي للموقع الجغرافي، الحجز عن بعد لسيارات الأجرة، وما إلى ذلك. تم تطوير التطبيق باستخدام العديد من الأساليب والأدوات لتكنولوجيا الهاتف المحمول والأنظمة متعددة الوكلاء.

Table des matières :

Chapitre 2 : Système Multi Agent.

Chapitre 3 : Analyse et Conception .

Chapitre 4 : Implémentation.

Liste des Figures :

Liste des tableaux :

Liste des abréviations :

A

ACC : Agent Communication Channel.

ACL: Agent Communication language.

AIP : Agent Interaction Protocol.

AMS : Agent Management System.

AUML: Agent Unified Modeling Language.

C

CORMAS : Common Resources Multi-Agent System.

D

DA: Dummy Agent.

DAO : Data Access Objects.

DF : Directory Facilitator.

G

GPS : Global Positioning System.

I

IA : Introspector Agent.

IAD : l'Intelligence Artificielle Distribuée.

J

JDBC : Java Database Connectivity.

JADE : Java Agent Développent.

JADE-LEAP: Lightweight and Extensible Agent Platform.

L

LGPL : Lesser General Public License.

R

RMA: Remote Managment Agent.

S

SA: Sniffer Agent.

SMA : Systèmes Multi-Agent

Introduction Général :

1. Contexte générale :

Les technologies de l'information et de la communication peuvent être considérées comme la révolution la plus importante et innovante dans ces dernières années. Cette révolution a permis l'émergence de la notion de la portabilité et de la mobilité qui offre un accès distant, instantanée et un flux sans interruption d'informations. En effet, cela est symbolisé par l'apparition des différents appareils de haute technologie tels que les smartphones et les tablettes qui sont dotées de plusieurs applications pratiques et utiles pour différents domaines comme le domaine de réservations en ligne de transport et de Taxi.

Grâce aux multiples fonctionnalités natives qu'ils offrent, ainsi que l'ergonomie spéciale qui respecte parfaitement leur résolution, les smartphones et les tablettes sont prêts à détrôner les ordinateurs à travers le monde. Cela a amené vers l'apparition des applications mobiles pour la réservation de taxi, qui correspond à l'utilisation de technologies sans fil, et plus particulièrement de la téléphonie mobile, afin d'effectuer des opérations de recherche et de réservations de taxis. Cette tendance offre aux utilisateurs, clients ou chauffeurs, de formidables services comme la réservation en ligne 24h/7jours, de voir exactement à quelle distance se trouve le chauffeur du taxi désigné, etc.

2. Objectif :

Malheureusement, malgré l'augmentation de l'utilisation des appareils mobiles et de leurs applications, la notion de réservation en ligne des taxis est autonome en Algérie dont l'informatisation étant très en retard. Pour cela, nous avons décidé de consacrer notre projet à la réalisation d'une application mobile accessible pour tous afin de palier ce défaut. Cette application var offre en particulier l'option de géolocalisation automatique, la possibilité de rechercher un taxi, de réserver une voiture à tout moment 24h/7jours, de voir tous les informations concernant un chauffeur et sa position. De plus pour les chauffeurs, la possibilité

d'ajouter facilement un service, l'obtention facile des clients, la possibilité de contacter un client en cas d'annulation et d'un retard etc.

Pour la réalisation de notre application, le paradigme multi-agents a été choisi pour la modélisation des différentes entités de notre système. Ce choix est motivé par la nature du système réalisé et par les caractéristiques des systèmes multi-agents qui sont devenus un paradigme dominant dans le domaine du développement des systèmes intelligents, distribués et complexes.

3. Organisation du mémoire :

Le manuscrit est composé de quatre chapitres suivants:

Chapitre 1: Dans ce chapitre, consacré aux applications mobiles, nous commençons par la présentation de quelques notions sur l'informatique mobile. Ensuite, nous allons présenter quelques exemples des applications existant pour la réservation en ligne de taxi.

Chapitre 2: Nous présentons dans ce chapitre les concepts d'agents et de système multi-agents et leurs propriétés et leurs caractéristiques. Nous présentons aussi quelques plateformes de développement des SMA ainsi que leurs principaux domaines d'application.

Chapitre 3: Dans ce chapitre, qui concerne la modélisation de notre application mobile pour la réservation de taxi, nous allons expliquer en détails toutes les phases de la conception de notre système en utilisant la méthodologie voyelle et le langage de modélisation AUML.

Chapitre 4: Ce chapitre concerne la description détaillée du fonctionnement de notre application. Il comporte aussi une description des différents outils et langages de développement utilisés.

Le mémoire se termine par une conclusion où nous avons expliqué l'intérêt du sujet, ce que nous avons réalisé et que quelques perspectives et des directions pour des travaux futurs.

Chapitre 01

Applications mobiles.

1. Introduction :

Dans le cadre de projet de fin d'études, et afin de réaliser ce chapitre, on a fait une étude sur l'application mobile, leurs techniques, leurs systèmes d'exploitation et leurs outils de développement et spécialement avec l'environnement Android sur lequel dépend notre mémoire. Ce chapitre présente aussi quelques exemples des applications mobiles existantes pour la réservation des taxis.

2. L'informatique Mobile :

Afin de résoudre les problèmes et d'atteindre les objectifs mentionnés précédemment, nous avons utilisé l'informatique mobiles qui est définie comme la possibilité pour des usagers munis de périphériques portables ou d'ordinateurs mobiles d'accéder à des services et à des applications évoluées, à travers une infrastructure partagée de réseau, indépendamment de la localisation physique ou du mouvement de ses usagers.

2.1 Application Mobile :

Les applications mobiles ont envahi notre quotidien. Consulter vos mails, jouer, parler à un ami, faire du shopping, consulter les actualités ou la météo, retrouver un trajet ou un hôtel, consulter vos comptes, vous pouvez le faire en un clic via votre téléphone portable. Entreprises ou particuliers, vous avez tous (ou presque), aujourd'hui recours à ces applications qui font partie intégrante de notre vie et sans lesquelles, pour beaucoup, la vie serait presque plus compliquée.

2.1.1 Définition :

Une application mobile est un type de logiciel ou programme conçu pour s'exécuter sur un appareil mobile, tel qu'un Smartphone ou une tablette. Les applications mobiles servent souvent à fournir aux utilisateurs des services similaires à ceux du PC. Les applications mobiles sont des

programmes relativement légers, autonomes, utilisés pour des services de l'information, des médias sociaux, des jeux, etc. **[1].**

2.1.2 Caractéristiques des applications mobiles :

Une application mobile est réussie, elle doit avoir certains critères très importants qui sont les suivants :

- ✓ Rapidité et fluidité, un premier critère pour une application réussie.
- ✓ Le poids de l'application mobile, un facteur important pour les utilisateurs.
- ✓ L'ergonomie, un critère pour fidéliser l'utilisateur
- ✓ Une bonne qualité d'affichage, un critère pour séduire l'utilisateur.
- ✓ Maintenance et évolutivité, la clé pour un bon suivi.

2.1.3 Stratégies de développement :

La conception d'applications mobiles peut se faire suivant trois stratégies de développement distinctes :application web, application native et hybride, comme le montre la figure app Mobile¸ suivante : **[1]**

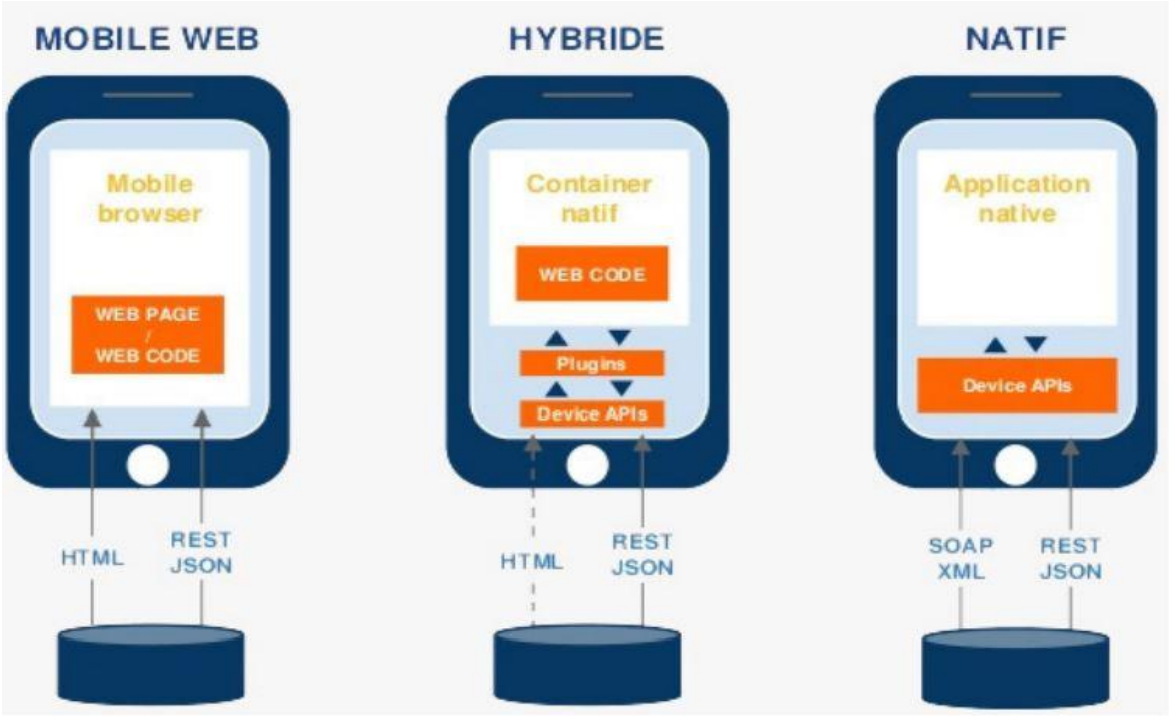


Figure 1.1 : Les différentes applications mobiles.

Dans ce qui suit, nous donnons un bref aperçu de chaque stratégie :

2.1.3.1 Application native :

Une application native est une application mobile spécifiquement développée pour un système d'exploitation mobile. Elle est conçue avec le langage et les outils associés à son système d'exploitation, et installée directement sur le mobile. Cette installation se faisant soit au travers d'un téléchargement via Internet soit par déploiement depuis un ordinateur connecté au mobile. Les tests pour vérifier le comportement de ces applications nécessitent des compétences techniques spécifiques et des appareils très couteux.

2.1.3.2 Application Web :

Une application mobile Web est une application développée en HTML, accessible et exécutable par le biais d'un navigateur Internet pour téléphone mobile. Elle utilise le navigateur du Smartphone et ne nécessite pas forcément de télécharger l'application. Les applications mobiles Web complètent les applications natives qui sont développées spécifiquement pour un système d'exploitation et qui doivent être téléchargées et installées par les mobinautes. Elles s'adressent donc à l'ensemble des utilisateurs de mobiles, et non à une population spécifique utilisant une marque bien précise. Toutefois, les applications Web doivent être testées pour chaque navigateur, résolution et taille d'écran, à l'instar de n'importe quel site Web.

2.1.3.3 Application hybride :

L'application hybride est une application pour mobiles qui combine des éléments HTML5 sous forme d'application mobile Web et des éléments d'une application native permettant l'utilisation des fonctionnalités natives des Smartphones et d'être distribuée en tant qu'application sur les stores des systèmes mobiles (App Store, Play Store, etc.)

2.1.4 Tableau comparatif des applications natives et web :

Le tableau suivant représente une comparaison des applications natives et les applications web. **[2]**

Critères.	**Applications native.**	**Application web.**
Portabilité.	Développement spécifique à chaque plateforme.	Navigateur web, mais une intégration distincte selon la plateforme.
Référencement.	Arriver à se positionner dans une boutique d'applications Accessible par la recherche	Accessible par les moteurs de recherche classiques et liens

	dans une boutique d'applications.	externes éventuels.
Accessibilité technique.	Dépendante de la plateforme et de l'éventuelle validation par une boutique d'applications mode offline possible.	Éventuelle dépendance aux navigateurs mode offline impossible, support HTML5 nécessaire.
Exploitation du mobile.	Utilise toutes les possibilités du mobile (GPS, contacts, camera, voix . . . etc.).	Se limite aux possibilités du navigateur.
Développement/coût.	Plus long, plus fastidieux nécessite un SDK + la connaissance d'un langage spécifique.	Généralement moins onéreux, HTML/JavaScript/CSS.
Effet immersif.	Plus de possibilités, richesse fonctionnelle et multimédia, logique marketing forte.	Limité. Des possibilités plus importantes avec l'arrivée de HTML5.
Expérience utilisateur.	Maximale. Possibilité de notifier l'utilisateur (Push) même quand l'application n'est pas utilisée.	Limitée mais conforme à l'utilisation classique du web.
Mise à jour.	Processus de soumission à un magasin d'application. Contraignant dans le cas de l'App Store d'Apple donc mise à jour en mode par action de l'utilisateur.	Mise à jour instantanée sur le serveur web, en une seule opération.
Développement / Courbe d'apprentissage	Dépends des compétences existantes sur le SDK du mobile visé	. Compétences HTML / CSS / JavaScript plus classique.

Table 1.1 : Tableau comparatif des applications natives et web.

2.1.5 Les avantages et les inconvénients d'une application mobile :

2.1.5.1 Les avantages d'une application mobile :

Une application mobile revêt de nombreux avantages comparés à un site web mobile, en voici 5 principaux :

- ☞ Un confort d'usage et une expérience utilisateur inégalée.
- ☞ L'accès direct aux contenus de l'application mobile via l'icône présent sur le Dashboard du téléphone ou de la tablette (mode d'accès sans URL).

- ☞ Un fonctionnement en mode déconnecté.
- ☞ Elle permet d'utiliser et d'intégrer toutes les fonctionnalités téléphone (accéléromètre, gyroscope, GPS, caméra…), ce qui n'est pas forcément le cas des Web Apps.
- ☞ L'implémentation de fonctionnalités natives comme par exemple les notifications « PUSH ».

2.1.5.2 Les inconvénients d'une application mobile :

- ☞ Le principal inconvénient d'une application mobile est qu'elle doit respecter les règles définies par les différentes sociétés des plateformes mobiles. Que ce soit l'approbation nécessaire des Apps Store pour diffuser l'application ou ses mises à jour.
- ☞ Les conditions tarifaires imposées ou le non compatibilité avec les autres systèmes d'exploitation mobiles.
- ☞ Le coût lié au développement d'une application mobile est un frein car généralement plus élevé si elle est portée sur plusieurs plateformes (afin d'être disponible pour un maximum de mobinautes) que le coût d'un site mobile ou d'une Web App. Il faudrait potentiellement prévoir un développement sur chaque technologie, et donc un coût supplémentaire si l'on souhaite se positionner sur tous les modèles.
- ☞ Pour que l'utilisateur ait accès à la dernière version, il faut qu'il la mette à jour depuis le store contrairement aux sites mobiles et Web App qui se mettent à jour directement.

2.1.6 Objectifs des applications mobiles :

Les applications visaient initialement l'amélioration de la productivité et la facilitation de la récupération d'informations telles que courrier électronique, calendrier électronique, contacts, marché boursier et informations météorologiques. **[3]**

Vers 2005, elles pénètrent les sociétés.

Puis, les développeurs d'application répondent ensuite à une demande du public et la disponibilité d'outils de développement ont conduit à une expansion rapide dans d'autres domaines, comme :

- Les jeux mobiles.
- Les automatismes industriels.
- Le GPS et les services permettant la localisation.
- Les opérations bancaires.

- Les suivis des commandes, l'achat de billets.
- Des applications médicales mobiles.
- La réalité virtuelle.
- L'écoute de musiques ou de radios.
- La visualisation de vidéos ou de chaines de télévision.
- La consultation d'internet.
- Les réseaux sociaux généraux (type Facebook).
- Les réseaux sociaux spécialisés.

Figure 1.2 : Liste des applications.

2.2 Systèmes d'exploitation mobiles :

- **Android :** Racheté par Google, Android se définit comme étant Open Source mobile adapté au Smartphone, PDA, MP3 et tablette.

Ce système d'exploitation mobile est développé en principe avec le langage Java, ce langage de programmation permet à tout logiciel d'être aisément compatible à différents systèmes d'exploitation comme UNIX, Windows et autres en apportant quelques rectifications ou même sens.

- **IOS :** Conçu par Apple uniquement pour ses propres appareils mobiles,

IOS est un système d'exploitation mobile développé par le langage Objective-C, langage faisant partie de C ANSI, reconnu par le fait qu'il transmet rapidement les messages ainsi que le chargement faisable à titre dynamique. Systèmes d'exploitation mobiles.

- **Windows phone :** Conçu par **Microsoft**, Windows Phone est un système d'exploitation mobile non restreint uniquement aux professionnels mais également les particuliers

peuvent s'en servir. En revanche, Microsoft a lancé des offres avancées destinées aux entreprises depuis l'apparition de Windows Phone 8

Windows Phone est développé par C#, c'est un langage de programmation similaire à Java en terme de syntaxe et des notions basiques. **[4]**

Figure 1.3 : Système d'application mobile.

2.3 Tableau Comparatif des différents systèmes d'exploitation mobiles :

Il existe des tonnes de différences entre les différents systèmes d'exploitation existants, dans le tableau suivant nous avons souligné les différences les plus pertinentes entre ces derniers :**[3]**

	Android	**iOs**	**Windows OS Phone**
Appareils compatibles.	Samsung galaxy, HTC.	Iphone,ipod ,ipad.	Windows phone, Nokia Lumia 710.
Dernière version.	7.1.	10. 0.2.	10.0.14393 321.
Date de sortie.	4/10/2016.	23/08/2016.	11/10/2016.
Open source.	✓	X.	X.
Mis à jour.	29/04/2017.	28/04/2017.	28/04/2017.
Support d'adobe flash.	✓ Intégré directement dans les applications.	X.	✓

Place de marché.	Google Play.	App store.	Windows Phone marketplace.
Nombre d'application.	+800 000.	+1000000.	+9 000.

TABLE 1.2 : Tableau comparatif des différents systèmes d'exploitation mobiles .

2.4 Le système utilisé « Android »:

2.4.1 Définition :

Android est un système d'exploitation mobile basé sur une version modifiée du noyau Linux et d'autres logiciels open source , conçu principalement pour les appareils mobiles à écran tactile tels que les smartphones et les tablettes . Android est développé par un consortium de développeurs connu sous le nom d' Open Handset Alliance et sponsorisé commercialement par Google . Il a été dévoilé en novembre 2007, avec le premier appareil Android commercial, le HTC Dream , lancé en septembre 2008. **[4]**

2.4.2 Versions d'Android :

L'historique des versions d'Android a débuté avec la sortie de la version 1.0 en septembre 2008 Android a connu plusieurs mises à jour depuis sa première version. Ces mises à jour servent généralement à corriger des bugs et à ajouter de nouvelles fonctionnalités, le tableau suivant résume quelques versions de l'Android :**[5]**

La version	La date d'apparition
Android 1.0.	Septembre 2008.
Android 1.1.	Février 2009.
Android 1.5 dit « Cupcake ».	Avril 2009.
Android 1.6 dit « Donut ».	Septembre 2009.
Android 2.0 dit « Eclair ».	Octobre 2009.
Android 2.0.1.	Décembre 2009.
Android 2.1, 2.2, 2.3.	Janvier - Décembre 2010.
Android 2.3.3, 2.3.4,2.3.5, 2.3.6,2.3.7.	Février-septembre 2011.
Android 3.0.	Février-septembre 2011.
Android 3.1.	Mai 2011.
Android 3.2.	Juillet 2011.
Android 3.2.1,3.2.2.	Septembre 2011.
Android 4.0.	Octobre 2011.
Android 5.0.	Octobre 2014.

Android 5.1.	Mars 2015.
Android7.0,7.1.1.	Septembre 2016.
Android 8.0 , 8.1.	21 août -25 octobre 2017
Android 9	6 août 2018
Android 10.	7 mai 2019.
Android 11	8 septembre 2020.

Tableau 1.3 : versions du système Android.

2.4.3 Architecture Android :

L'architecture de la plateforme Android se décline, selon une démarche bottom up, en quatre principaux niveaux que sont : **[6]**

- Le noyau linux
- Les librairies et l'environnement d'exécution,
- Le module de développement d'applications .
- Les différentes applications.

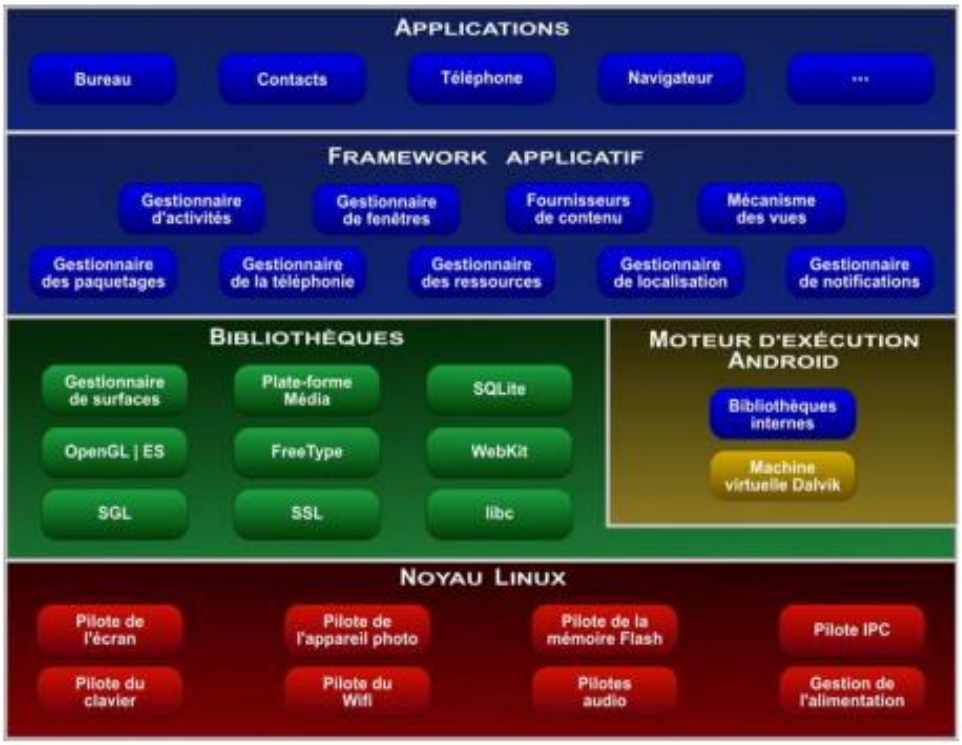


Figure 1.4 : Les composants du système d'exploitation Android.

3. Réservation de taxi :

Depuis quelques années, l'offre de transport public n'a cessé de se développer et de s'améliorer, et avec elle les demandes et les exigences des usagés, rendant ces moyens de transport moins compétitifs. L'utilisation des taxis reste assez répondue dans ces zones de par son confort et son efficacité malgré les quelques inconvénients qu'ils possèdent en l'occurrence sa disponibilité en temps voulu, le temps nécessaire pour trouver un taxi, ainsi que le coût élevé.

Notre travail consiste à explorer les solutions existantes relatives à la réservation de taxi et à concevoir et développer une application mobile basée agents sous Android.Cette application est destinée, à la fois, aux clients et aux chauffeurs de taxis.

Pour cela nous avons pris l'initiative d'améliorer quelques inconvénients et de rajouter des fonctionnalités à notre application qui sont citées parmi les principaux objectifs de cette dernière et qui sont les suivants :

- ⇒ Réservation d'un taxi de manière simple et rapide en temps réel en ayant obligatoirement un Smartphone Android.
- ⇒ Application collaborative entre le client et le taxieur.
- ⇒ C'est le client et le taxieur qui permettent à l'application de s'autogérer.
- ⇒ L'utilisateur peut s'inscrire par téléphone, Facebook ou bien en utilisant son compte Google.
- ⇒ Utilisation de la géolocalisation pour voir les taxis à un rayon de 5Km en temps réel.
- ⇒ La position du client est localisée automatiquement.
- ⇒ Le client peut indiquer un lieu de rendez-vous s'il le souhaite.
- ⇒ Après avoir reçu une demande de réservation et dans le cas où le taxieur accepte la demande, Il indique le prix qu'il trouve adéquat, à son tour le client pourra soit accepter ou refuser.
- ⇒ La demande de réservation expire automatiquement au bout de trois minutes s'il n'y a pas de réponse.
- ⇒ Après avoir envoyé une demande de réservation, le client ne peut pas faire une autre demande.

⇒ Après que la course soit confirmée, tous les taxis seront supprimés de la Map du client hormis celui qu'il a réservé.

⇒ Après avoir accepté une demande de réservation, le taxieur ne peut plus l'annuler.

⇒ Le taxieur devient indisponible pour les autres clients, après une course confirmée.

⇒ Le client peut suivre la position du taxi qu'il a réservé en temps réel.

⇒ Le client sera notifié lorsque le taxieur arrive.

⇒ Le client peut consulter un guide d'utilisation "Comment ça marche" qui est utile lors de la première utilisation de l'application.

⇒ Accéder à l'historique des courses.

⇒ Possibilité au client de noter le taxieur.

3.1 Pourquoi ?

Le mémoire s'intitule «Réalisation d'une application basée agents pour la réservation de taxi».

Notre application nommée « Taxi43 » est destinée, à la fois, aux clients et aux chauffeurs de taxis possédant un Smartphone sous Android.

En résumé,

L'application côté client permet la réservation de taxis en temps réel après une géolocalisation sur la carte.

L'application côté chauffeur permet de recevoir les demandes de réservation et d'afficher la position du client ainsi que sa destination, le chauffeur de taxi indique le prix au client dans le cas où il accepte la demande, et précisera s'il est libre dans l'immédiat ou non en indiquant quand il le sera, et dans le cas où il décide de ne pas accepter la demande il l'annule. A son tour le client aura la possibilité d'accepter ou de refuser le prix indiqué par le taxieur si ce dernier ne lui convient pas.

3.2 Etude de quelques solutions existant pour la réservation de taxi :

3.2.1 Uber :

⇒ Agence Uber est une société née san Francisco qui permet aux clients la réservation des taxis avec chauffeur directement depuis smartphone à l'aide de la géolocalisation

⇒Uber fournit à ses clients des services de haute par exemple : proposer l'eau, des boissons, des sucreries, etc.

⇒Dans Uber, toutes les transactions financières se font en ligne.

⇒La société soigne les liens entre ses chauffeurs et les clients afin de gagner leur confiance.

☞ **Le principe de Uber :**

Avant l'utilisation du Uber vous devez d'abord vous inscrire en utilisant votre numéro de téléphone ou Email et un numéro de compte bancaire et spécifier un mot de passe pour votre compte .

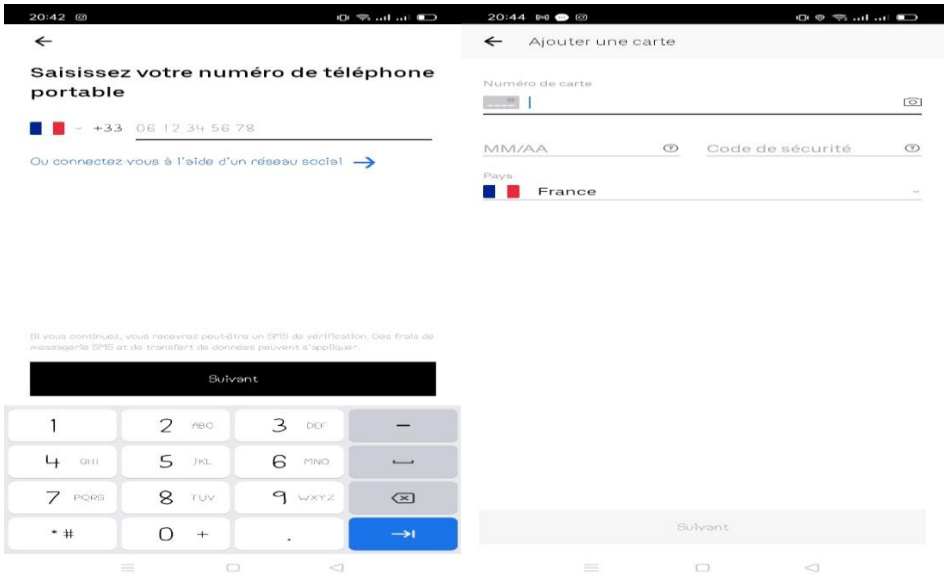


Figure 1.5: Inscription UBER.

Puis Uber détecte automatiquement votre position à l'aide de la GPS . Pour Commander un taxi, indiquez votre point de récupération et validez sur la carte comme le montre la figure 1.6. Uber donne alors instantanément le chauffeur le plus proche de vous .

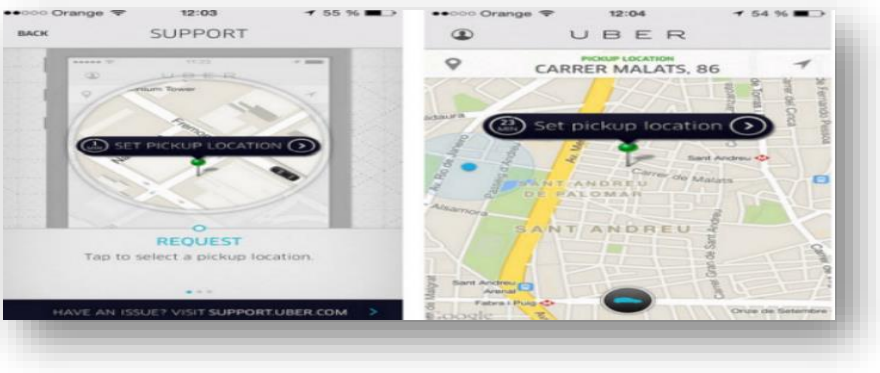


Figure 1.6 : Indication du point de récupération sur la carte.

3.2.2 Yassir :

Yassir est une application qui fonctionne seulement à Alger, qui est mettre les personnes qui ont besoin se déplacer de commander des taxis par leur smartphone.

☞ **Le principe de Yassir :**

Le principe de l'application est simple. Lorsque vous avez besoin de commander un taxi, l'application YASSIR vous indique le nombre et lieux des chauffeurs .

Pour profiter L'accès au service YASSIR nécessite que le client s'inscrive sur l'application, en remplissant Le formulaire préce effet comme le montre la figure 1.7.

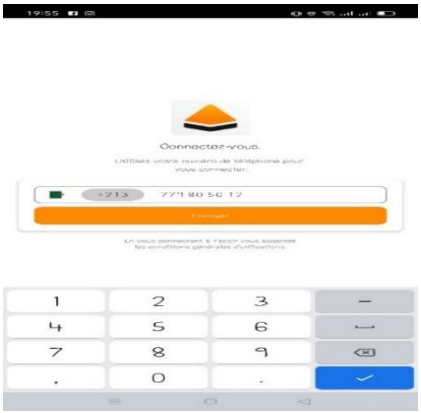


Figure 1.7 : Inscription Yassir .

Après l'inscription, Le client doit préciser votre position et votre destination puis l'application fournissant tout les informations concernant votre commande et le paiement ...etc.

Figure 1.8 : Choisir votre destination.

Figure 1.9 : Commander Taxi Yassir.

3.2.3 ITaxi :

Itaxi la nouvelle application **Smartphone** qui arrive au Maroc le 21/09/2014 uniquement à Casablanca .Commander son taxi en quelques clics via son Smartphone c'est possible. La société Itaxi.ma propose une réservation des taxis par application mobile.

☞ **Le principe de ITxi :**

- Téléchargez l'application iTaxi.ma.
- Réservez votre iTaxi de Ville ou d'Aéroport.
- Suivez votre chauffeur en temps réel.
- Évaluez votre trajet.

3.2.4 Discussion :

3.2.4.1 Tableau comparatif des solutions existantes :

Le tableau suivant représente une comparaison entre les solutions existantes :

Critères solutions	Type de réservation	Disponibilité	Avantages	Limites
Uber	En temps réel.	24h/24. 7j/7.	☞ Simple et facile à utiliser. ☞ Paiement en ligne. ☞ Service rapide. ☞ Véhicules propres. ☞ Application compatible avec le système d'exploitation iOS et Android.	☞ Escroqueries par les conducteurs. ☞ La durée et la distance du trajet sont inconnues.
Yassir	En temps réel.	24h/24. 7j/7.	☞ L'estimation du prix se fait avant l'envoi de la demande de réservation. ☞ Deux types de taxi sont disponibles : Taxi simple et Yassir Food.	☞ La réservation ne peut être annulée après la confirmation de la demande. ☞ La source n'est pas localisée automatiquement lors de l'envoi de la demande de réservation.
ITaxi	Réservation à l'avance ou en temps réel.	24h/24. 7j/7.	☞ Une tarification raisonnable. ☞ La possibilité de payer en cash ☞ Sécurité au sens large. ☞ Application compatible avec le système d'exploitation iOS et Android.	☞ Réservation lente. ☞ Le délai d'attente est très grand.

Tableau 1.4: Tableau comparatif des solutions existantes.

3.2.5 Contribution :

A travers ce travail, nous avons d'abord essayé d'étudier et de comprendre les systèmes existants en matière de réservation de taxis puis nous avons tenté d'apporter un certain nombre d'améliorations, notamment sur les plans suivants :

- Facilité et simplicité d'utilisation de l'interface de l'application, que ce soit pour le client ou pour le chauffeur de taxi.
- Lors d'une demande de réservation, la source du client sera localisée automatiquement et dans le cas où le client décide d'indiquer un lieu de rendez-vous il l'a modifié.
- Le chauffeur de taxi pourra accepter ou refuser une demande de réservation.
- Le client ne peut pas envoyer une demande de réservation à un autre taxieur s'il en a déjà envoyé une.
- Le client sera notifié lorsque le chauffeur de taxi arrive à destination en plus de suivre sa position sur la carte.

4. Conclusion :

Ce chapitre fournit une présentation générale du la problématique de notre Application, ainsi que, nous avons parlé quelque notion concernant l'informatique mobile et ces différents systèmes d'exploitation mobiles.

Chapitre 02

Système Multi-agents.

1. Introduction :

Le Système Multi-Agent (SMA) est une discipline de l'Intelligence Artificielle Distribuée (IAD). Cette discipline a vu le jour par la rencontre de l'IA et des systèmes distribués afin de remédier aux insuffisances de l'IA classique et pour permettre la résolution de certains problèmes qui sont distribués de manière inhérente. Les SMA ont prend de plus en plus une place importante parmi les technologies de développement des systèmes complexes et distribués. Nous présentons dans ce chapitre les principaux concepts de la technologie agent et des SMA, ainsi que leurs caractéristiques et quelques exemples de leurs domaines d'application.

2. Notion d'agent :

2.1. Qu'est-ce qu'un agent ?

Il n'y a pas une définition acceptée en unanimité pour la notion d'agent. En ce qui suit, nous allons présenter quelques définitions de ce concept.

- **Un agent :** est un système informatique, situé dans un environnement, et qui agit d'une façon autonome pour atteindre les objectifs (buts) pour lesquels il a été conçu **[7].**
- **Un agent :** est une entité qui fonctionne continuellement et de manière autonome dans un environnement où d'autres processus se déroulent et d'autres agents existent **[8].**
- **Un agent :** est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents **[9].**

☞ **Les agents intelligents :** sont des entités logiciels qui réalisent des opérations à la place d'un utilisateur ou d'un autre programme, avec une sorte d'indépendance ou d'autonomie, et pour faire cela, ils utilisent une sorte de connaissance ou de représentation des buts ou des désirs de l'utilisateur **[10].**

La structure d'un agent, avec les composantes principales et les fonctions ainsi définies, est représentée dans la figure 2.1.

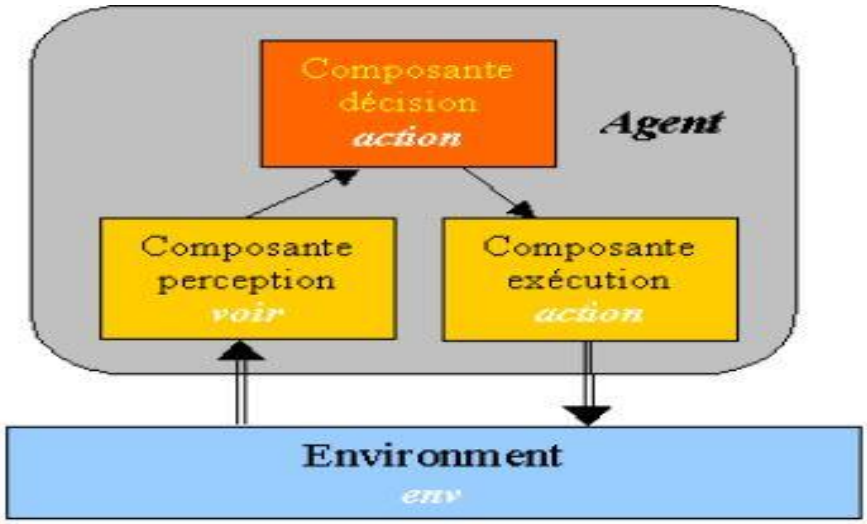


Figure 2.1 : Structure générale d'un agent.

2.2. Caractéristiques d'un agent :

Par la définition de Jennings, Sycara et Woodbridge **[11].,** on peut identifier les caractéristiques suivant pour la notion d'agent :

- **Situé** : l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement.
- **Autonome**: l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne.
- **Flexible :** l'agent dans ce cas est :
  - ***Pro-actif* :** l'agent doit exhiber un comportement pro-actif et opportuniste, tout en étant capable de prendre l'initiative au bon moment
  - ***Capable de répondre à temps*** : l'agent doit être capable de percevoir son environnement et d'élaborer une réponse dans le temps requis.

- ***Social*** **:** l'agent doit être capable d'interagir avec les autres agents (logiciels et humains) quand la situation l'exige afin de compléter ses tâches ou aider ces agents à accomplir les leurs.

2.3. Types d'agents :

2.3.1. Agents réactifs :

Les agents réactifs : sont souvent qualifiés de ne pas être " intelligents " par eux-mêmes. Ils sont des composantes très simples qui perçoivent l'environnement et sont capable d'agir sur celui-ci. Ils n'ont pas une représentation symbolique de l'environnement ou des connaissances et ils ne possèdent pas de croyances, pas de mécanismes d'envoi de messages. Leurs capacités répondent uniquement au mode stimulus/action qui peut être considéré comme une forme de communication. Un SMA constitué d'agents réactifs possède généralement un grand nombre d'agents et présente un comportement global intelligent.

Les agents réactifs sont considérés intelligents au niveau du groupe, du système. En conséquence, l'intelligence est distribuée entre beaucoup d'agents réactifs et le comportement intelligent devrait émerger de l'interaction entre ces agents réactifs et l'environnement. **[12].**

2.3.2. Agents cognitifs :

Les agents cognitifs ont une représentation partielle de l'environnement, des buts explicites, ils sont capables de planifier leur comportement, mémoriser leurs actions passées, communiquer par envoi de messages, négocier, etc. Un SMA constitué d'agents cognitifs possède communément peu d'agents. **[11].**

2.3.3. Agents hybrides :

Une architecture hybride d'un agent intelligent est une architecture composée d'un ensemble de modules organisés dans une hiérarchie, chaque module étant soit une composante cognitive avec représentation symbolique des connaissances et des capacités de raisonnement, soit une composante réactive. De cette manière, on combine le comportement pro-actif de l'agent, dirigé par les buts, avec un comportement réactif aux changements de l'environnement. En plus, on espère obtenir simultanément les avantages des architectures cognitives et réactives, tout en éliminant leurs limitations. **[13].**

3. Système multi-agents :

3.1. Définition :

On appelle système multi-agents (SMA), un système composé des éléments suivants **[14]**:

1. Un environnement E, c'est-à-dire un espace disposant généralement d'une métrique.
2. Un ensemble d'objet O situés, C'est à dire pour tout objet, il est possible, à un moment donné, d'associer une position dans E. De plus, ces objets sont passifs, c'est à dire peuvent être perçus, créés, détruits et modifiés par les agents.
3. Un ensemble A d'agents, représentent les entités actives du système.
4. Un ensemble de relations R qui unissent des agents entre eux.
5. Un ensemble d'opérations Op permettant aux agents de percevoir, produire, consommer, transformer et manipuler des objets.
6. Des opérateurs chargés de représenter l'application de ces opérations et la réaction de l'environnement envers les tentatives de modification.

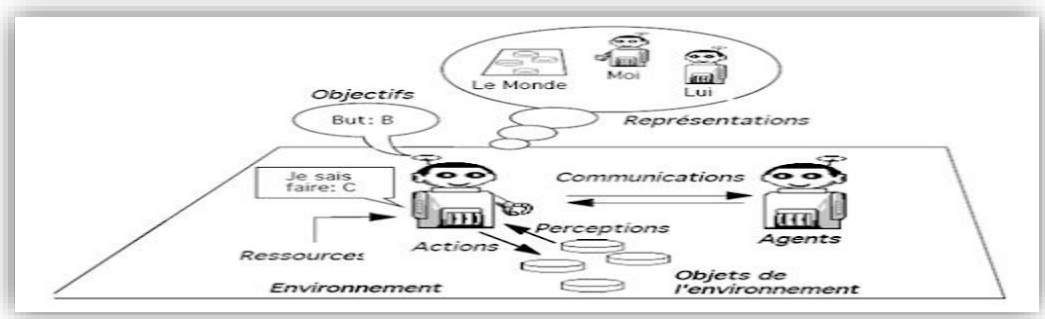


Figure 2.2 : Représentation schématique SMA.

3.2. Caractéristiques :

Un SMA est généralement caractérisé par **[15]** :

- Chaque agent a des informations ou des capacités de résolution de problèmes limitées,
- Ainsi chaque agent a un point de vue partiel.

- Il n'y a aucun contrôle global du système multi agent.
- Les donnés sont décentralisées.
- Le calcul est asynchrone.

3.3. Types :

On peut différencier les SMA selon le nombre d'agents, la nature et la complexité de ses agents. Ainsi, un SMA peut être soit **[16]** :

- **Ouvert :** les agents y entrent et en sortent librement (par exemple : l'utilisation d'un SMA pour développer une application de commerce électronique).
- **Fermé :** où l'ensemble d'agents reste le même.
- **Homogène :** dont tous les agents sont construits sur le même modèle.
- **Hétérogène :** dont les agents de modèles sont différents, c'est-à-dire de granularités différentes.

3.4. Intérêts :

Parmi les principaux intérêts des SMA, nous pouvons citer :

- Ils s'adaptent bien à la réalité dans la mesure où de nombreux problèmes sont de nature distribuée.
- L'utilisation de nombreux agents résolvant le même problème de façon différente, produit généralement des solutions de meilleure qualité en termes de complétude et de précision.
- Ils permettent d'intégrer des connaissances diverses et complexes. Les connaissances insuffisantes peuvent être complétées suivant la résolution.
- Ils permettent de résoudre des problèmes de taille et de complexité telle qu'il n'est pas réaliste d'essayer de les résoudre avec un seul agent.
- La modularité : la complexité d'un système d'IA croit avec la taille de sa base de connaissances. Partitionner ce système en N agents réduit sa complexité par un facteur parfois plus grand que N, et la configuration résultante se trouve plus facile à développer, à tester et à maintenir.

3.5. L'interaction dans les SMAs :

3.5.1. Notion :

Une des principales propriétés de l'agent dans un SMA est celle d'interagir avec les autres agents. Ces interactions sont généralement définies comme toute forme d'action exécutée au sein du système d'agents et qui a pour effet de modifier le comportement d'un autre agent. Elles permettent aux agents de participer à la satisfaction d'un but global. Cette participation permet au système d'évoluer vers un de ses objectifs et d'avoir un comportement intelligent indépendamment du degré de complexité des agents qui le composent.

En général, les interactions sont mises en œuvre par un transfert d'informations entre agents ou entre l'environnement et les agents, soit par perception, soit par communication. Par la perception, les agents ont connaissance d'un changement de comportement d'un tiers au travers du milieu. Par la communication, un agent fait un acte délibéré de transfert d'informations vers un ou plusieurs autres agents. L'interaction peut être décomposée en trois phases non nécessairement séquentielles (Chaïb-draa, 1996) :

- La réception d'informations ou la perception d'un changement,
- Le raisonnement sur les autres agents à partir des informations acquises,
- Une émission de message(s) ou plusieurs actions (plan d'actions) modifiant l'environnement. Cette phase est le résultat d'un raisonnement de l'agent sur son propre savoir-faire et celui des autres agents.

Le degré de complexité des connaissances nécessaires pour traiter les interactions dépend des capacités cognitives (de raisonnement) de l'agent et du fait que l'agent a connaissance ou non de l'objectif du système global. En effet, un agent qui poursuit un objectif individuel au sein du système comme c'est le cas pour les agents dits réactifs ne focalise pas son énergie pour interagir avec les autres même s'il y est amené. Par contre, un agent qui participe à la satisfaction du but global du système tout en poursuivant un objectif individuel, va passer une partie de son temps à coopérer ou à se coordonner avec les autres agents. Pour cela, il doit posséder des connaissances sociales qui modélisent ses croyances sur les autres agents [**11]**.

3.5.2. Types :

On a principalement quatre types d'interactions que les agents peuvent utiliser qui sont : la coordination, la coopération, la communication et la négociation.

3.5.2.1. Coordination :

La coordination est le processus de construction de programmes en assemblant les parties actives. Ainsi, la coordination détermine en quelque sorte quelles sont les règles de bon fonctionnement du système auquel les agents appartiennent. Selon la nature des agents et des interactions, plusieurs formes de coordination sont possibles. La coordination peut être imposée à un agent par une sorte de contrat, comme le respect d'un protocole d'interaction donné. Elle peut aussi être apprise, lorsque l'agent trouve un intérêt à participer à la collectivité **[17].**

3.5.2.2. Coopération :

La coopération est une caractéristique très importante dans les systèmes multi agents. En effet, une résolution distribuée d'un problème est le résultat de l'interaction coopérative entre les différents agents. Dans le cadre d'une telle dynamique collective, un agent doit disposer en plus de la connaissance reflétant son degré d'implication dans cette dynamique (croyances, buts) d'un certain nombre de compétences nécessaires pour la coopération. Il doit pouvoir **[17]** :

- Mettre à jour le modèle du monde environnant.
- Intégrer des informations venant d'autres agents.
- Interrompre un plan pour aider d'autres agents.

3.5.2.3. Communications :

Le système de communication qui lie un ensemble d'agents agit comme un système nerveux qui met en contact des individus parfois séparés. La communication en effet agrandit les capacités perceptives des agents en leur permettant de bénéficier des informations et du savoir-faire des autres agents. Les communications sont indispensables à la coopération et il est difficile de concevoir un système d'agents coopérants s'il n'existe pas un système permettant aux agents d'échanger. **[12]**.

3.5.2.4.Négociation :

La négociation est une discussion entre deux agents ou plus qui essaient d'établir une solution à leur problème. Le participant peut être un seul agent, un groupe d'agents, un system. La négociation est donc une activité qui consiste à échanger des informations entre participants afin d'aboutir à un compromis mutuellement acceptable . **[18]**.

3.6. Plateformes SMA :

Les plateformes multi-agents sont des Framework permettant aux développeurs de réaliser des applications sans perdre de temps à réaliser des fonctions de base pour la création et l'interaction entre agents et éliminent, dans la plupart des cas, la nécessité d'être familier avec les différents concepts théoriques des systèmes multi-agents. Dans la suite de cette sous-section nous allons présenter quelques exemples, les plus connues, des plateformes multi-agents.

3.6.1. JADE :

Jade (Java Agent Développent) est un Framework de développement de systèmes multi-agents, open-source et basé sur le langage Java. Il offre en particulier un support avancé de la norme FIPA-ACL, ainsi que des outils de validation syntaxique des messages entre agents basé sur les ontologies **[19]**.

3.6.2. MAGIQUE :

Une plate-forme pour agents physiquement distribués écrite en Java et fournissant un modèle de communication original d'appel à la cantonade. Dans MAGIQUE, les compétences sont dissociées des agents. L'architecture des agents et les différentes compétences sont développées séparément. Les compétences sont ensuite greffées comme plugin dans les agents au gré du concepteur. Cette plate-forme est développée au sein du LIFL **[19]**.

3.6.3. MACE :

Le premier environnement de conception et d'expérimentation de différentes architectures d'agents dans divers domaines d'application. Dans MACE, un agent est un objet actif qui communique par envoi de messages. Les agents existent dans un environnement qui regroupe tous les autres agents et toutes les autres entités du système. Un agent peut effectuer trois types d'actions : changer son état interne, envoyer des messages aux autres agents et envoyer des requêtes au noyau MACE pour contrôler les évènements internes.

Chaque agent est doté d'un moteur qui représente la partie active de l'agent. Ce moteur détermine l'activité de l'agent et la façon dont les messages sont interprétés. MACE a été utilisé pour développer des simulations d'applications distribuées **[20].**

3.6.4. SemanticAgent :

Basé sur JADE et permet le développement d'agents dont le comportement est représenté en SWRL. SemanticAgent est développé au sein du LIRIS , il est open-source et sous licence GPL V3 **[21]**.

3.6.5. Jadex :

Une plate-forme agent développée en JAVA par l'université de Hambourg qui se veut modulaire, compatible avec de nombreux standards et capable de développer des agents. **[23]**.

3.6.6. CORMAS (Common Resources Multi-Agent System) :

Un Framework de développement de systèmes multiagents, open-source et basé sur le langage de programmation orientée objet SmallTalk. Il est centré sur des problématiques de recherche en sciences du développement et de négociation entre acteurs **[22].**

3.7. Domaines d'application des SMA :

Plusieurs applications bénéficient particulièrement du paradigme d'agents mobiles. Parmi ces applications **[23]**:

- **Le commerce électronique :** les agents mobiles sont bien adaptés pour le commerce électronique. Une transaction commerciale peut nécessiter un accès en temps réel aux ressources distantes. Les agents mobiles représentent leurs propriétaires et peuvent agir et négocier en leurs noms en utilisant différents stratégies pour atteindre leurs buts.
- **Recherche d'information distribuée :** dans cette recherche, l'agent est transféré vers les sources d'informations où il crée localement des indexes et les renvoie à son propriétaire s'il est disponible. Même si le propriétaire est déconnecté, La recherche peut être poursuivie.
- **Services des réseaux de télécommunication :** la gestion de ces services de est caractérisée par la reconfiguration dynamique du réseau et la personnalisation des

utilisateurs. Dans ce cas, L'utilisation de la technologie des agents mobiles maintient ces systèmes flexibles et efficaces en dépit des exigences strictes dans lesquelles ils opèrent.

- **Surveillance et notification :** c'est des applications qui montre la nature asynchrone des agents mobiles. Ces agents m sont capables de surveiller la source d'information et son emplacement. Dans ce cas, la durée de vie des agents mobiles est indépendante du processus qui les crée.
- **Calcul parallèle :** les agents mobiles peuvent utiliser pour administrer des tâches parallèles en créant une cascade de clones dans le réseau. Ainsi, un calcul qui nécessite une grande puissance de traitement, peut être distribué dynamiquement via une infrastructure d'agents mobiles**.**

4. Conclusion :

Dans ce chapitre, nous avons eu un aperçu général sur les notions d'agent et de systèmes multi-agents, ainsi que les différentes plateformes de développement et quelques exemples de domaines d'application des SMA. Les SMA se composent d'un ensemble d'agents organisés dans des sociétés. Ces agents interagissent, communiquent et coopèrent entre eux pour accomplir des tâche bien déterminée. Avec leurs caractéristiques, on conclut que les agents et les SMA, avec leurs propriétés, constituent une bonne approche pour développer notre application de réservation de taxi.

Chapitre 03
Analyse et conception

1. Introduction :

Après avoir présenté les concepts d'applications mobiles et de systèmes multi-agents dans les chapitres précédents, nous présentons dans ce chapitre les phases d'analyse et de conception de notre système. Dans la première section, nous allons faire une description fonctionnelle du système que nous voulons développer. Ensuite, nous allons présenter la méthode voyelle et le langage de modélisation AUML (Agent Unied Modelising Langage) que nous avons utilisé pour la modélisation de notre système. Dans la troisième section, nous allons identifier les agents et les différentes interactions entre ces derniers et nous terminerons par la partie conception ou nous allons présenter les différents diagrammes décrivant notre application.

2. Description du système réalisé :

Dans ce projet de fin d'étude, nous voulons développer un système de réservation en ligne de taxi basé sur les technologies mobile et multi-agents. Le système développé doit contenir en particulier les fonctionnalités suivantes :

- La possibilité de création de comptes soit pour les clients ou les chauffeurs.
- La géolocalisation automatique.
- La recherche facile de taxis.
- La réservation de taxi par les clients.
- Le contacte facile entre les clients et les chauffeurs.

Dans ce système, les utilisateurs (clients (passager), chauffeur et administrateur) doivent être représentés par des agents logiciels capables d'effectuer d'une manière autonome tous les taches au lieu des utilisateurs humains.

3. Choix méthodologique

3.1. AUML :

3.1.1. Définition :

AUML est un langage de modélisation graphique qui a été standardisé par FIPA (Foundation for Intelligent Physical Agents) comité technique de modélisation. Il a été proposé comme une extension du langage de modélisation unifié (UML). Jusqu' à maintenant il n y'a pas de standard reconnu pour la modélisation des systèmes multi-agents et AUML a émergé comme un candidat pour assumer une tel position. Il utilise les caractéristiques de ''décomposition'', ''d'abstraction'', et ''d'organisation'', qui réduisent la complexité de développement de logiciel. AUML décompose le système en petites parties d'objets, de modèles, de cas d'utilisation ou de classes, et d'actions opérationnelles. Concernant ''l'abstraction'', elle offre une vue spécialisée abstraite de la modélisation (classe, cas d'utilisation, diagramme, interface, etc.). **[24].**

3.1.2. Diagrammes AUML :

Comme nous l'avons indiqué dans la définition, AUML est une extension d'UML afin de prendre en compte les notions d'agent que ce dernier n'a pas. AUML hérite il hérite les différents représentations proposées par UML avec des extensions pour permettre de représenter les concepts de la technologie agent. En voici la liste des principaux diagrammes d'AUML**.[24]**

1. Diagrammes de séquence.
2. Diagramme de collaboration.
3. Diagramme d'activité.
4. Diagramme d'état transition.
5. Diagramme de cas d'utilisation.
6. Diagramme de classe.
7. Diagramme d'objets.
8. Packages.
9. Diagramme de composants.

10. Diagramme de déploiement.

3.2. Méthodologie Voyelle :

3.2.1. Méthode Voyelle :

L'approche Voyelles de Demazeau est une méthode de haut niveau (peu de directives techniques sont données) mais très souvent citée car reposant sur des principes purement multi-agents. Elle repose sur la décomposition de la vue d'un système suivant quatre dimensions (ou lettres) Agent, Environnement, Interaction et Organisation.

- **Les agents :** ils concernent les modèles (ou les architectures) utilisés pour la partie active de l'agent, d'un simple automate à des cas plus complexes, comme des systèmes à base de connaissances.
- **Les environnements :** ils sont les milieux dans lesquels sont plongés les agents. Ils sont spatiaux dans la plupart des applications proposées.
- **Les interactions :** ils concernent les infrastructures, les langages et les protocoles d'interaction entre agents, depuis de simples interactions physiques à des interactions par actes de langage.
- **Les organisations :** qui structurent les agents en groupes, hiérarchies, relations... etc **[25].**

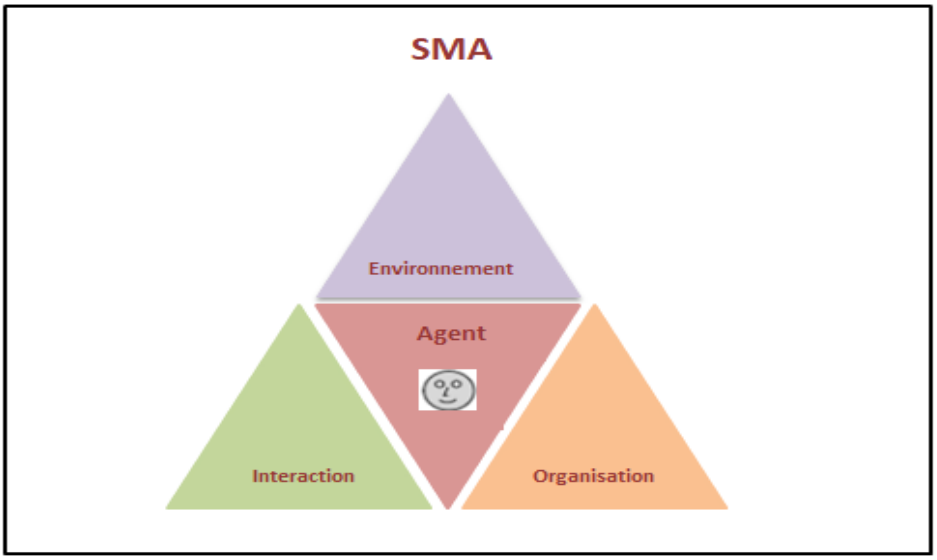


Figure 3.1 : Méthodologie voyelle.

3.2.2. Le processus de développement avec la méthode Voyelles :

- **Analyse :** porte sur le domaine d'application et le type du problème, et consiste en la décomposition du problème en voyelles, c.-à-d. les différentes composantes du système (Agent, Environnement, Interactions, Organisation (O), Utilisateurs (U)).
- **Conception :** correspond à l'identification des modèles à utiliser pour chacune des voyelles.
- **Implémentation :** consiste en l'instanciation des modèles, en utilisant des plates-formes et des langages choisis. Le résultat, le système implémenté, peut alors être exécuté, évalué et repensé en cas d'inadéquation avec les besoins exprimés par le type de problème et le domaine d'application.

4. Analyse :

4.1. Identification des utilisateurs :

Dans le système élaboré, nous identifions trois types d'utilisateurs : Les clients, les chauffeurs et l'administrateur du système.

☞ **Les clients :** ce sont les personnes qui accèdent au système pour chercher et réserver un taxi. Parmi les activités principales des clients dans le système on trouve :

- La recherche et la consultation des taxis.
- La réservation des taxis.
- La consultation du compte et de toutes les informations concernant ses réservations.

☞ **Les chauffeurs :** ce sont les personnes qui utilisent notre système pour conduit les clients à leur destination. Parmi les activités principales des chauffeurs dans le système on trouve :

- La gestion des demandes de réservations.
- La MAJ des services (ajout, modification et suppression de service).
- Consultation des informations concernant du compte.

☞ **L'administrateur :** c'est le responsable de la gestion du système. Les activités principales de l'administrateur dans le système sont :

- Gérer les comptes des utilisateurs : valider les comptes des chauffeurs et supprimer et désactiver les comptes des utilisateurs.

4.2. Identification des agents :

La figure 3.2 exprime l'architecture générale de notre système. Le système est composé de plusieurs types d'agents, chacun joue un rôle bien précis. Comme il est montré dans la figure 7, on peut distinguer deux catégories d'agents :

- Les agents système : ces agents s'exécutent dans le serveur principal.
- Les agents interface qui vont s'exécuter dans les appareilles des utilisateurs.

Dans la suite de cette section nous allons présenter les différents agents de notre système ainsi que les rôles joués par chacun.

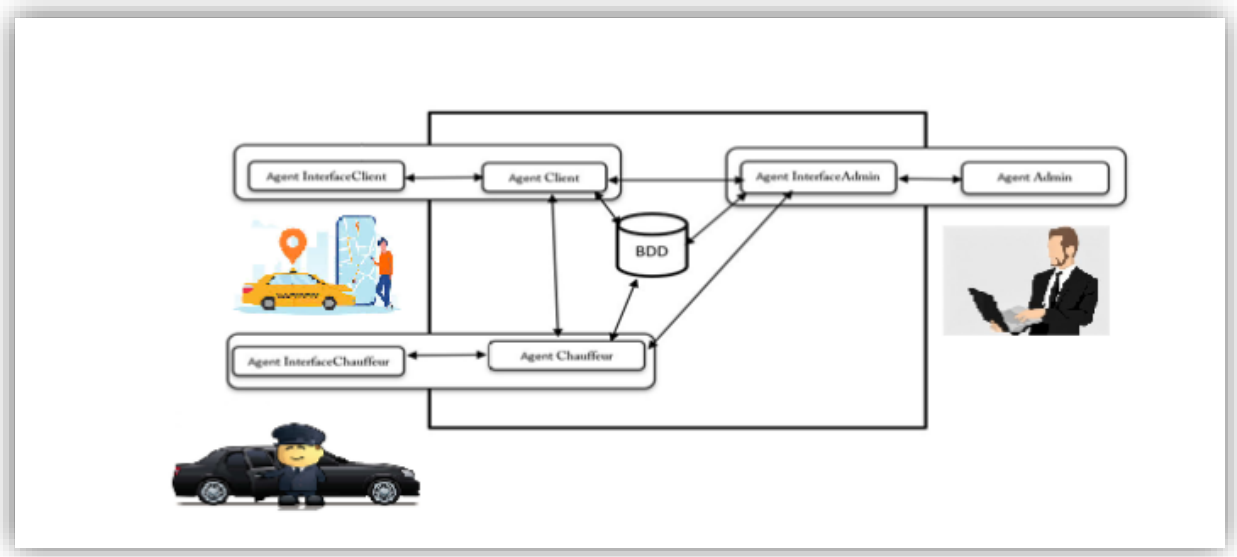


Figure 3.2: Architecture générale du système.

Les agents interfaces

➢ **Agent InterfaceClient :** c'est un agent interface qui s'exécute dans l'appareil du client. Il permet à l'utilisateur de s'interagir avec notre système. Cet agent est doté

d'une interface permettant aux clients d'accéder à toutes les l'information nécessaires et d'effectuer les opérations des réservations (le résultat de la recherche, les informations sur les réservations (les prix, taxi, la localisation des chauffeurs, etc).

- **Agent InterfaceChauffeur :** c'est un agent interface qui s'exécute dans l'appareille du conducteur et lui permet, à travers d'une interface spécifique, en particulier de consulter et répondre aux demandes de réservations, ...etc. Cet agent est le responsable de la géolocalisation automatique du chauffeur.
- **Agent InterfaceAdmin :** c'est un agent interface destiné à l'administrateur du système.

Les agents système

- **Agent Client :** Il représente l'entité virtuelle du client au sein du système. Donc il est responsable des opérations de réservation (rechercher, commander, consulter . . .). Il reçoit les données à partir de l'agent interface et prend un contact avec les autres agents de system pour réaliser ces opérations.
- **Agent Chauffeur :** Il représente l'entité virtuelle du chauffeur au sein du système. Donc il est responsable des opérations de Chauffeur (mettre à jour des Services, gérer et Consulter les commandes des clients), Il reçoit les données à partir de l'agent interface et prend un contact avec les autres agents du system pour réaliser ces opérations.
- **Agent Administrateur :** Il représente l'entité virtuelle de l'administrateur au sein du système, il est le responsable des opérations d'administrateur dans le système ainsi que la création de tous les autres agents du système.

4.3. Identification des interactions

- **Interaction Agent InterfaceClient / Agent administrateur :** L'agent InterfaceClient demande l'authentification à l'agent administrateur. Ce dernier crée un agent client lié à ce client.
- **Interaction Agent InterfaceChauffeur / Agent administrateur :** L'agent InterfaceChauffeur demande l'authentification à l'agent administrateur. Ce dernier crée un agent Chauffeur lié à ce Chauffeur.

- **Interaction Agent InterfaceClient / Agent Client :** L'Agent InterfaceClient peut envoyer à l'agent client des requêtes de recherche de taxis, de réservation. L'agent Client répond par le renvoie les résultats de traitement de ces requêtes.
- **Interaction Agent InterfaceChauffeur /Agent Chauffeur :** L'Agent InterfaceChauffeur envoie une demande Mise à jour des services, consulter des commandes et des informations personnelles. L'agent Chauffeur répond par le renvoie des résultats de traitement de ces requêtes.
- **Interaction Agent Interface Administrateur/Agent administrateur :** L'agent Interface Administrateur demande des informations d'application, Valider et supprimer des comptes à l'agent administrateur. Ce dernier effectuer l'opération.
- **Interaction Agent Client / Agent Chauffeur :** l'agent Client transmet les requêtes de recherche de taxis aux agents Chauffeurs du système. Les agents chauffeurs traiter la requête et répond soit par une acceptation ou un refus. En cas de requête de réservation, l'agent client transmet la requête à l'agent chauffeur concerné qui répond soit par un message de confirmation ou un message d'erreur.
- **Interaction Agent Chauffeur / Agent Interface Chauffeur :** lors de la réception d'une requête de recherche de taxis, l'Agent Chauffeur envoi une requête à Agent InterfaceChauffeur pour voir la localisation géographique du chauffeur et son état (libre ou occupé). En outre, lors de la réception d'une requête de réservation, l'Agent Chauffeur demande à l'Agent InterfaceChauffeur la confirmation du chauffeur.

4.4. L'environnement :

Dans les SMAs, l'environnement d'un agent est constitué des agents du système et des autres entités physique ou logique avec lesquels il s'interagit. Le tableau sous-dessous montre l'environnement de chacun des agents de notre système.

Agent	**Environnement**
Agent Interface Chauffeur	Chauffeur + Agent Chauffeur
Agent Interface Client	Client + Agent Client.
Agent Interface Administrateur	Administrateur + Agent administrateur

Agent Chauffeur	Agent InterfaceChauffeur + Agent Client+ Agent Administrateur + Base de données de système.
Agent client	Agent InterfaceClient + Agent Chauffeur + Agent Administrateur + Base de données de système.
Agent administrateur	Agent InterfaceClient+ Agent InterfaceChauffeur + Agent Interface Administrateur + Base de données de système +

Table 3.1 : L'environnement de chaque agent de notre système.

4.5. L'organisation :

L'organisation est une structure décrivant comment les agents dans l'environnement sont en relation et interagissent afin d'atteindre des buts. L'organisation structurelle est :

- Côté client, chauffeur ou administration on retrouve des agents interfaces qui qui interagir avec les agents du système (l'agent Client, l'agent Chauffeur et l'agent Administrateur). Ces agents représentent les points d'interaction entre le système et les utilisateurs (Client, Chauffeur, Administrateur).
- Dans le serveur on trouve un agent Administrateur, les Agents Client et les agents Chauffeurs. L'agent Administrateur est le responsable du système (inscription, authentification, création des agents).

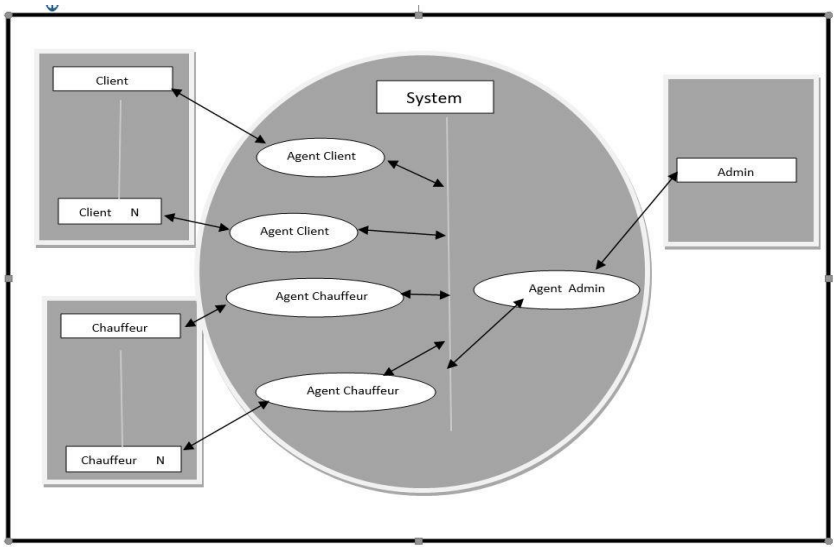


Figure 3.3 : Organisation du system.

5. Conception :

5.1. Identification des cas d'utilisations :

5.1.1. Le diagramme de cas d'utilisation :

Dans la figure 3.4, nous présentons le diagramme des cas d'utilisation qui ne liste que des fonctions générales essentielles et les acteurs principaux de notre System.

❖ Les acteurs principaux sont :

- **Le client :** qui utilise notre système pour recherche et réserver des taxis.
- **Le chauffeur :** qui utilise notre système pour gérer les commandes et MAJ de leurs Services.
- **L'administrateur :** qui est le responsable de la gestion du système.

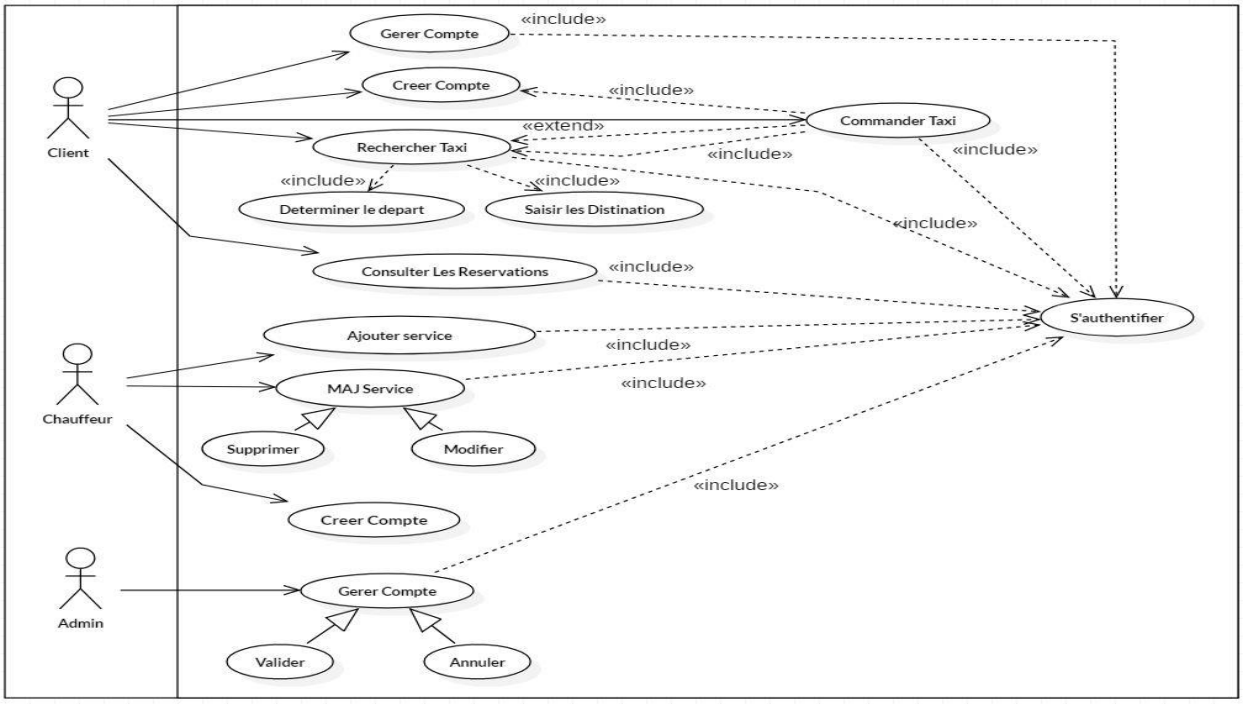


Figure 3.4 : Le diagramme de cas d'utilisation.

5.1.2. Le diagramme de Séquence :

Dans ce qui suit, nous représentons le diagramme de séquence d'un scénario représentatif de chacun des cas d'utilisation décrits précédemment.

Le diagramme de Séquence 'Créer Compte' :

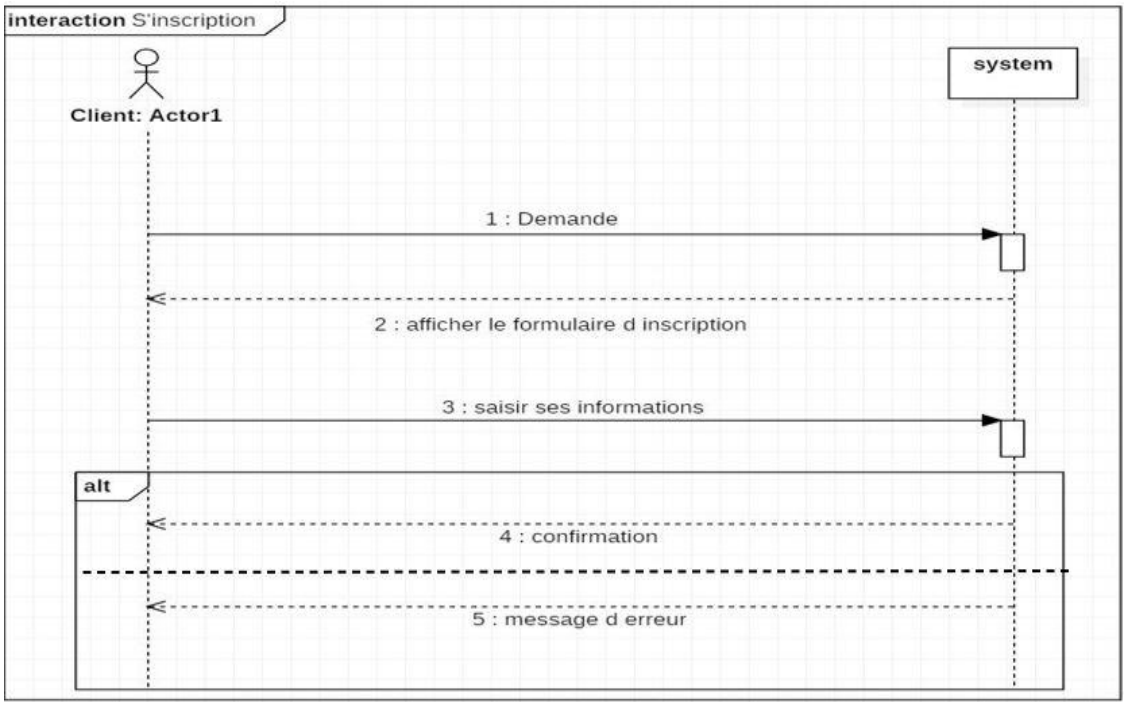


Figure 3.5 : Le diagramme de Séquence ‘ Créer Compte ’.

Le diagramme de Séquence ‘ Rechercher Taxi ’

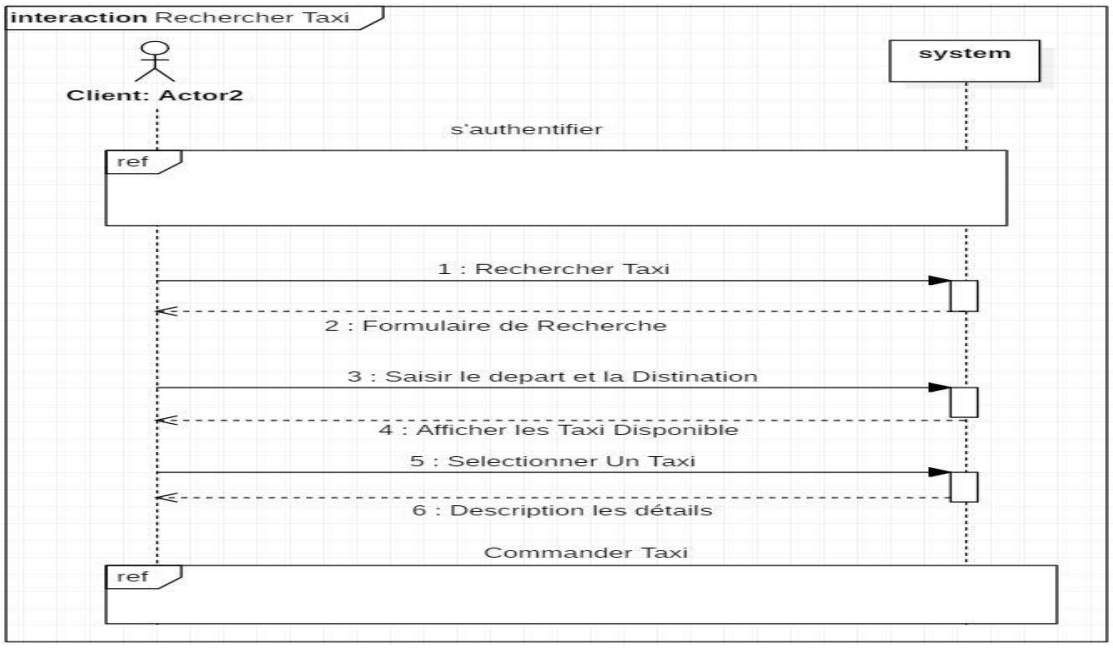


Figure 3.6 : Le diagramme de Séquence ‘ Rechercher Taxi ’.

Le diagramme de Séquence ‘ Réserver Taxi ’ :

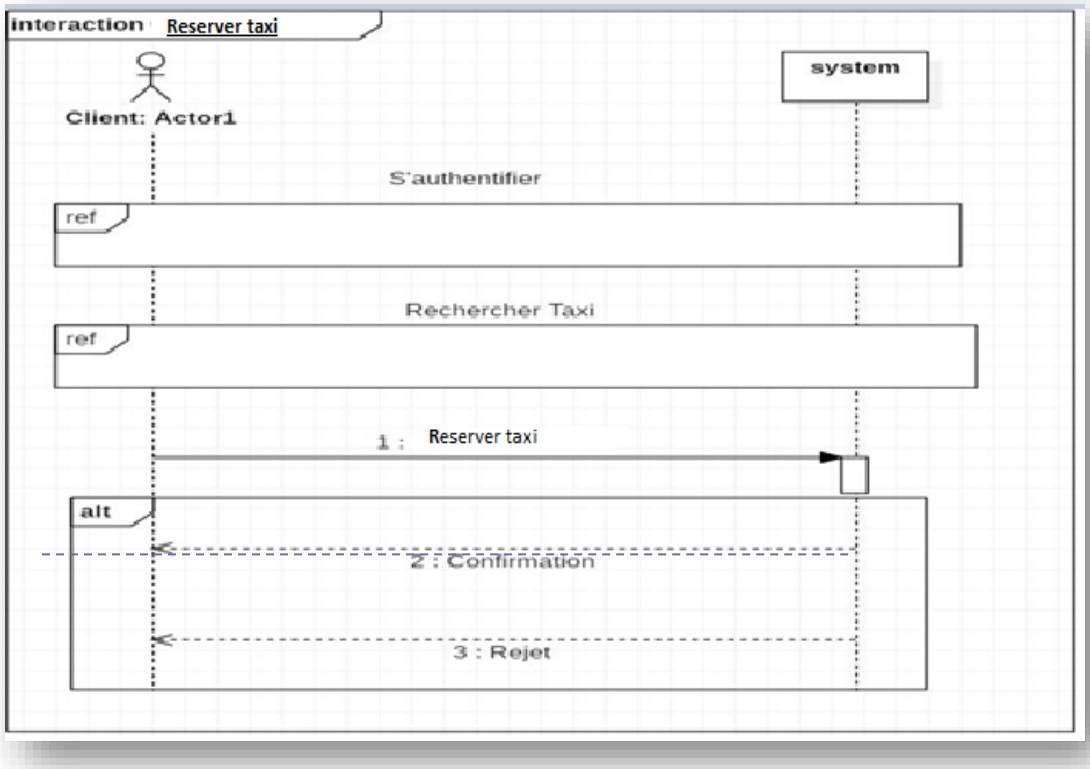


Figure 3.7 : Le diagramme de Séquence ‘ Réserver Taxi ’.

Le diagramme de Séquence ‘ Ajouter service ’ :

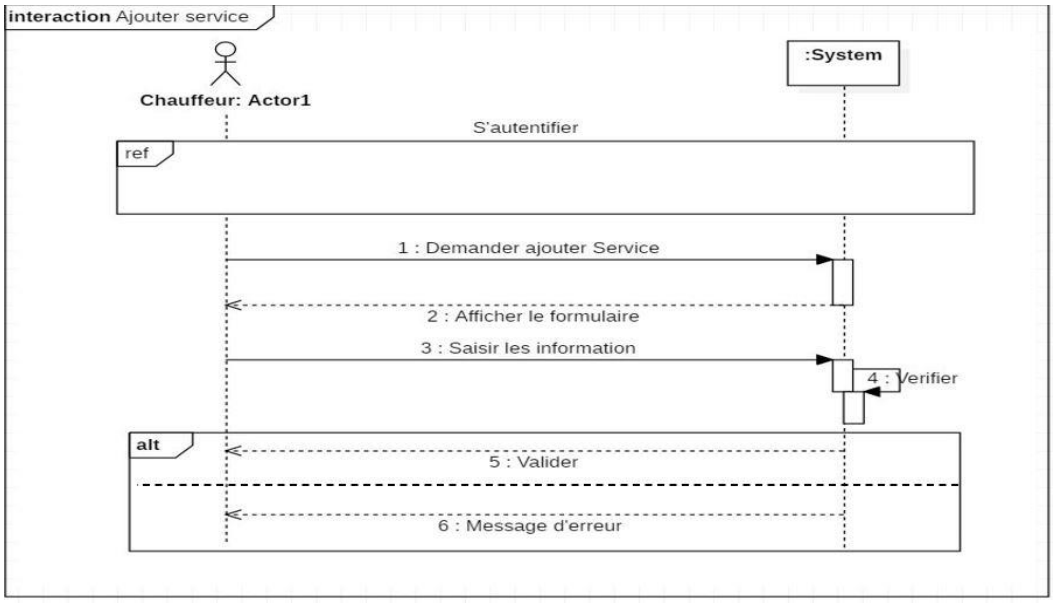


Figure 3.8 : Le diagramme de Séquence ‘ Ajouter Service ’.

Le diagramme de Séquence ‘ MAJ Service ’ :

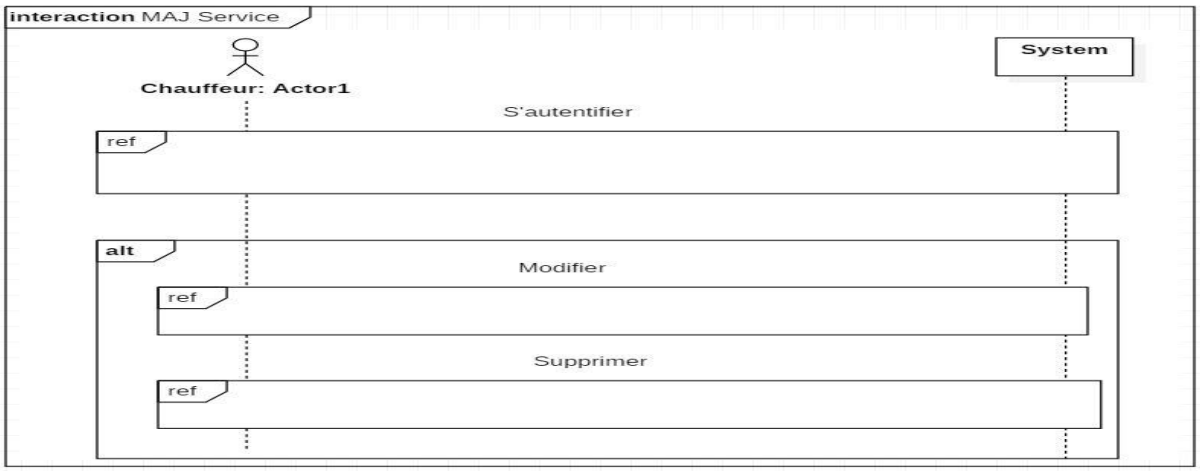


Figure 3.9 : Le diagramme de Séquence ‘ MAJ Service ’.

5.2. Diagramme de protocole d’interactions :

5.2.1. Diagramme de protocole d’interaction de s’authentification :

Dans ce diagramme, l’Agent Interface envoie une requête de s’authentification avec le login et le mot de passe à l’Agent Administrateur. Ce dernier vérifie les données. Si les données sont valides, alors il renvoie le message (CONFIRM) et crée un agent lié à ce client, sinon il renvoie un message d’erreur (FAILURE). La figure 3.10 illustre le diagramme de protocole de s’authentification.

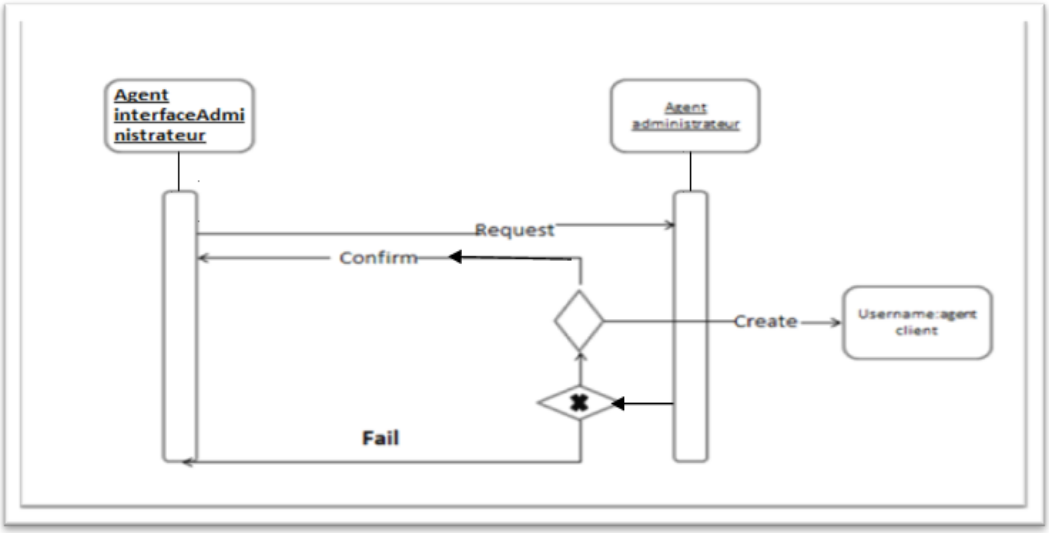


Figure 3.10 : Diagramme de protocole d’interactions authentification.

5.2.2. Diagramme de protocole d’interaction de Rechercher taxi :

Dans ce diagramme, l'Agent interface du client envoie une requête de recherche taxis à son Agent client. L'Agent client transmet cette requête aux différents agents chauffeurs dans le système. Chaque agent chauffeur traite la requête et rependre par un message (INFORM) contenant la liste des commandes trouvés. Lors de la réception de toutes les réponses et après les traitements nécessaires, l'agent client informe l'agent GUI du résultat de recherche.

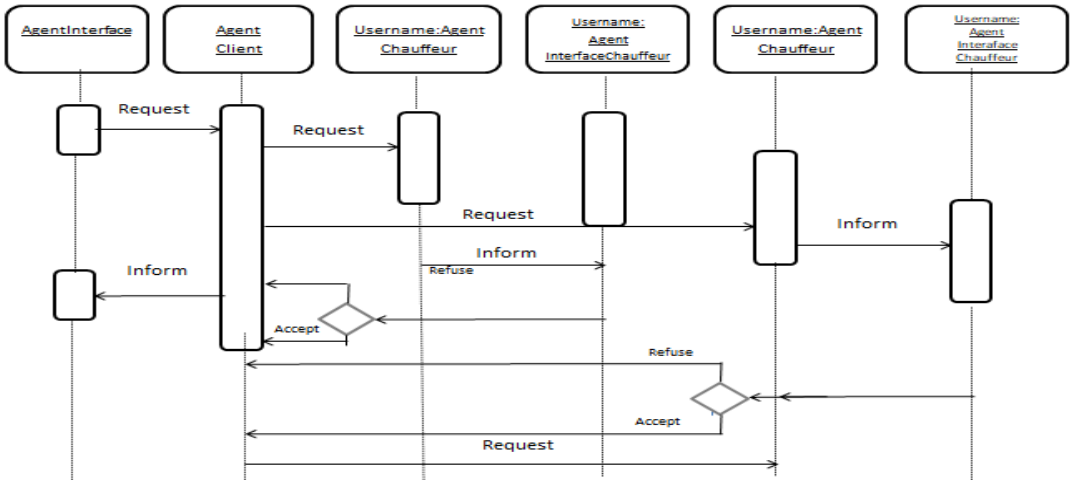


Figure 3.11 : Diagramme de protocole d'interactions Rechercher taxi.

5.2.3. Diagramme de protocole d'interaction de Réserver taxi :

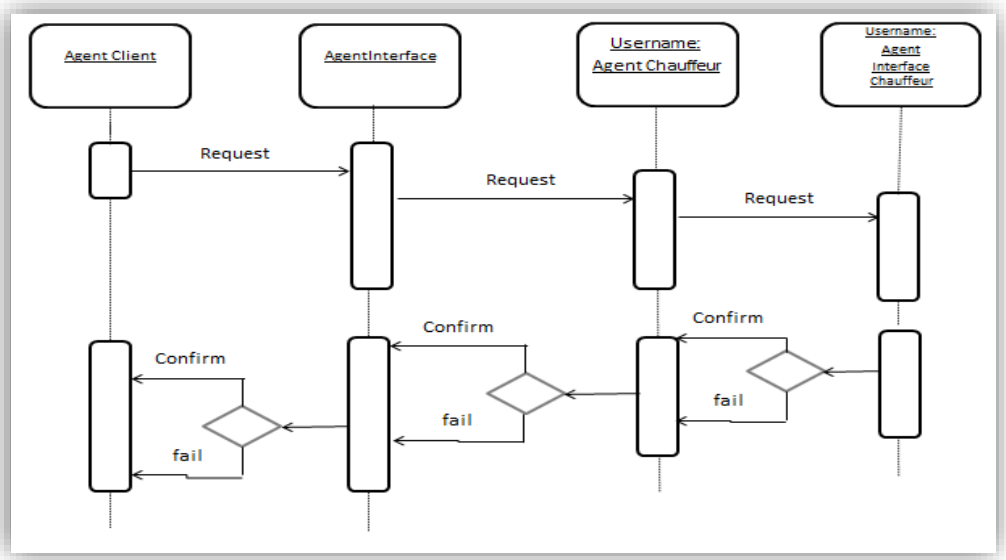


Figure 3.12 : Diagramme de protocole d'interactions Réserver taxi.

5.2.4. Diagramme de protocole d'interaction d'ajouter service :

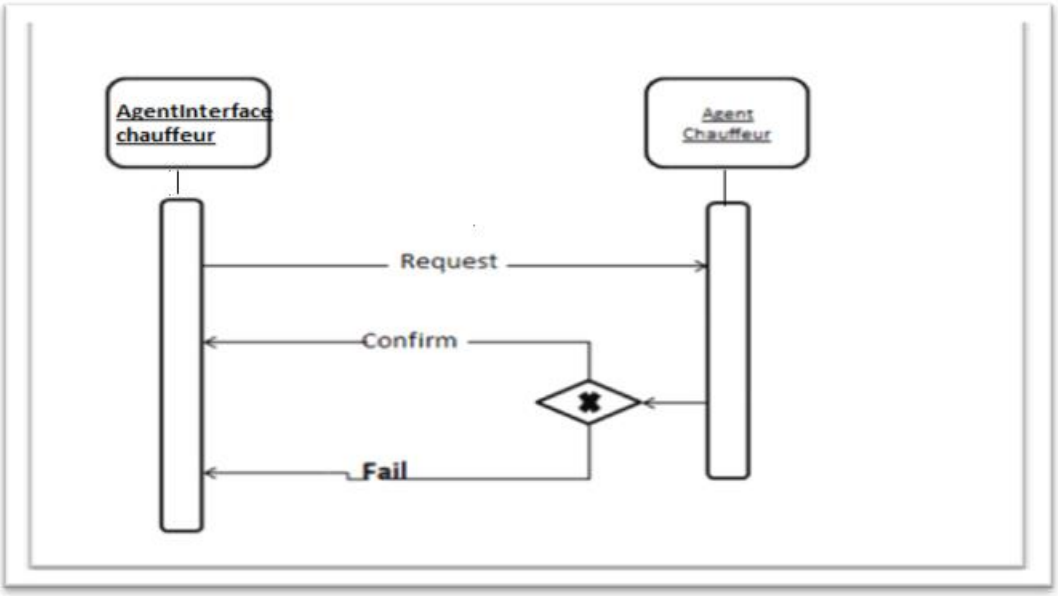


Figure 3.13: Diagramme de protocole d'interactions Ajouter service.

5.3. Diagramme de class :

5.3.1. Diagramme de Classe du système :

La figure 3.14 présente le diagramme de classes de notre système.

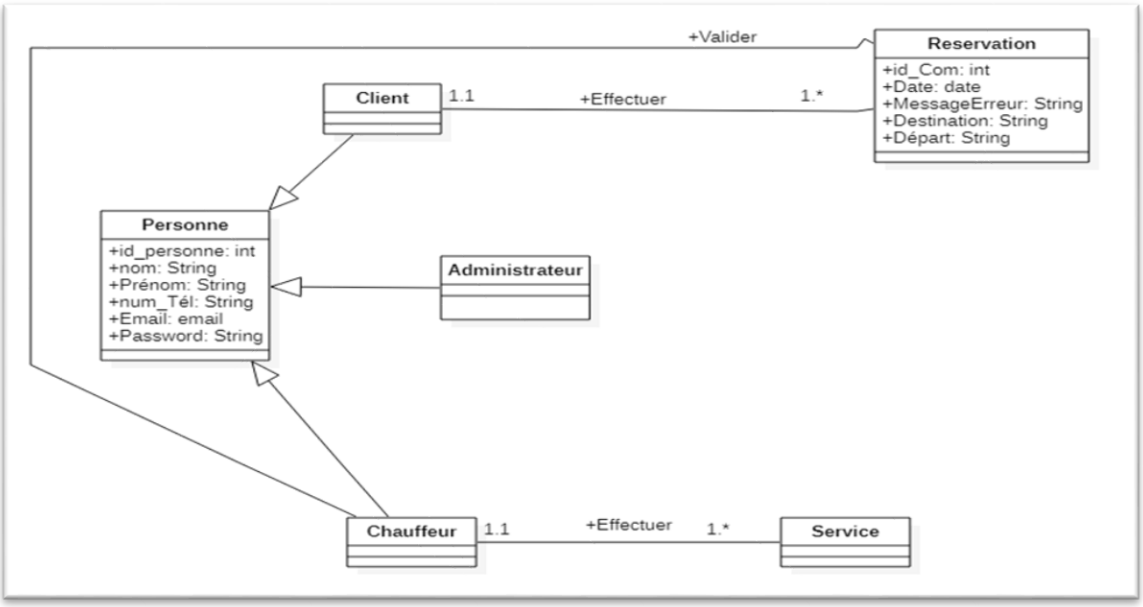


Figure 3.14: Diagramme de classes.

5.3.2. Diagramme de Classe agents :

La Figure 3.15 présentes le diagramme de classes des agents de notre système. Dans ce diagramme nous avons cinq classes d'agent qui héritent de la classe agent.

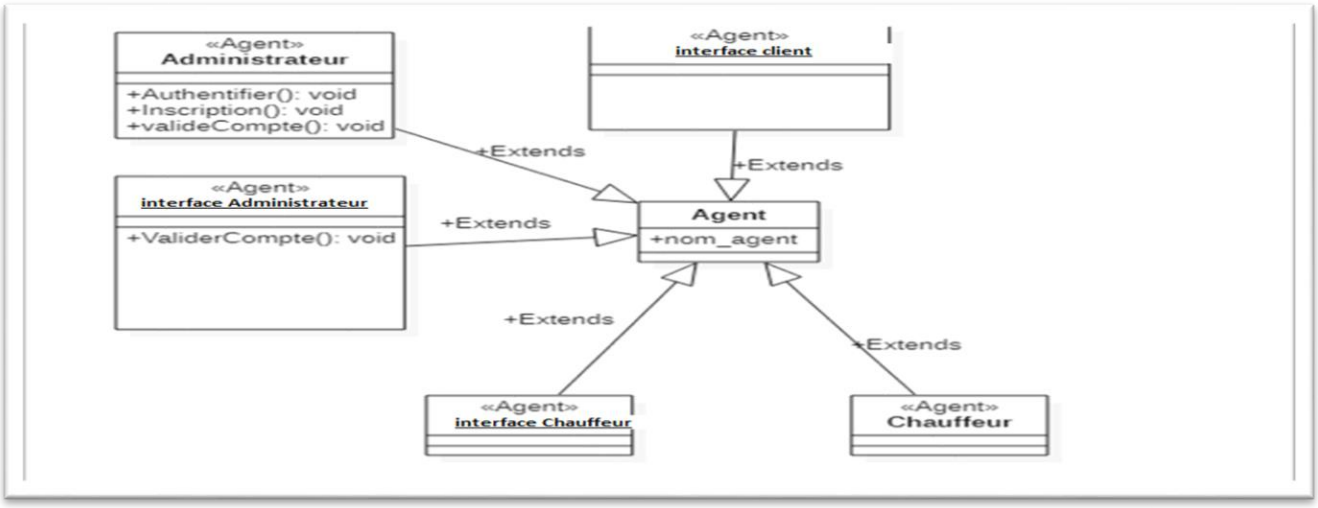


Figure 3.15 : Diagramme de classes agents.

Les figures 3.16 au 3.21 donnent une représentation détaillée de différentes classes d'agents de notre système.

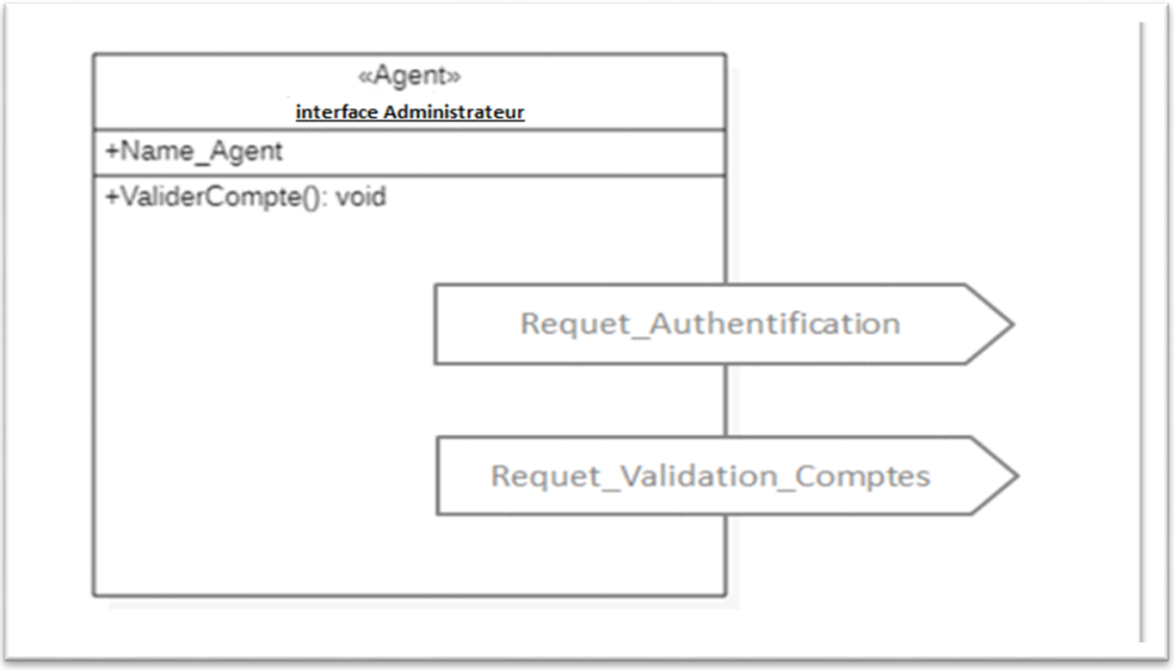


Figure 3.16 : Classe Agent Interface administrateur.

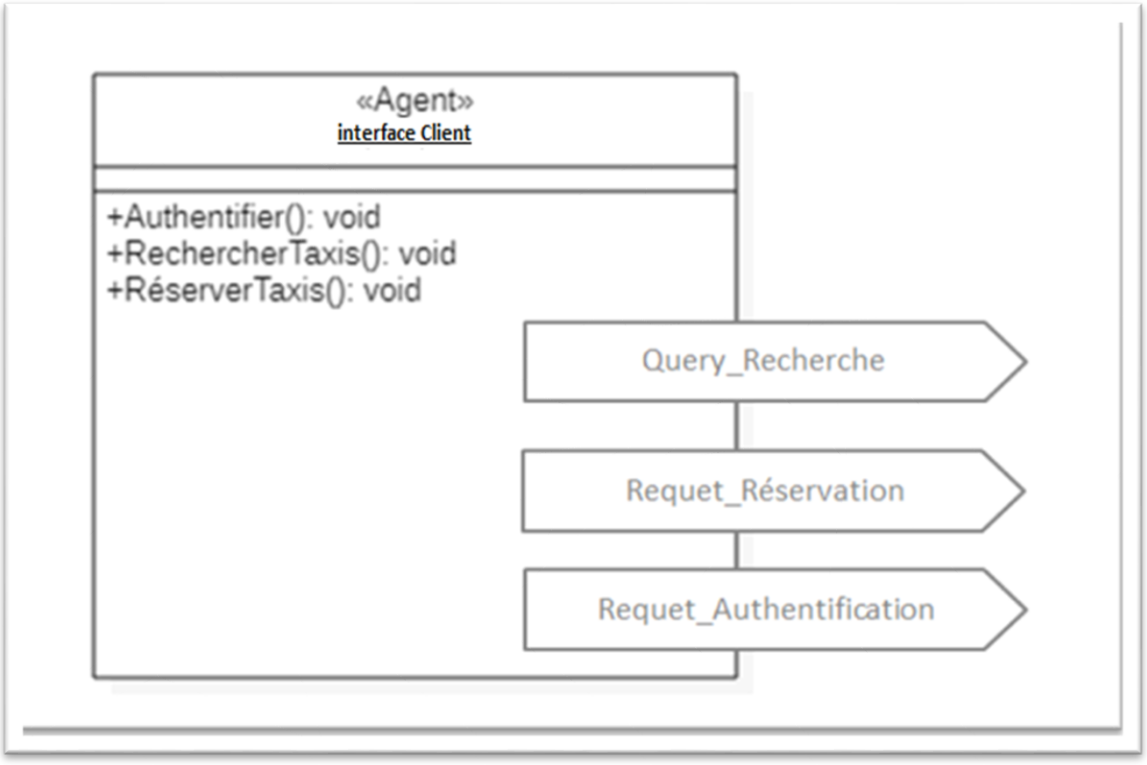
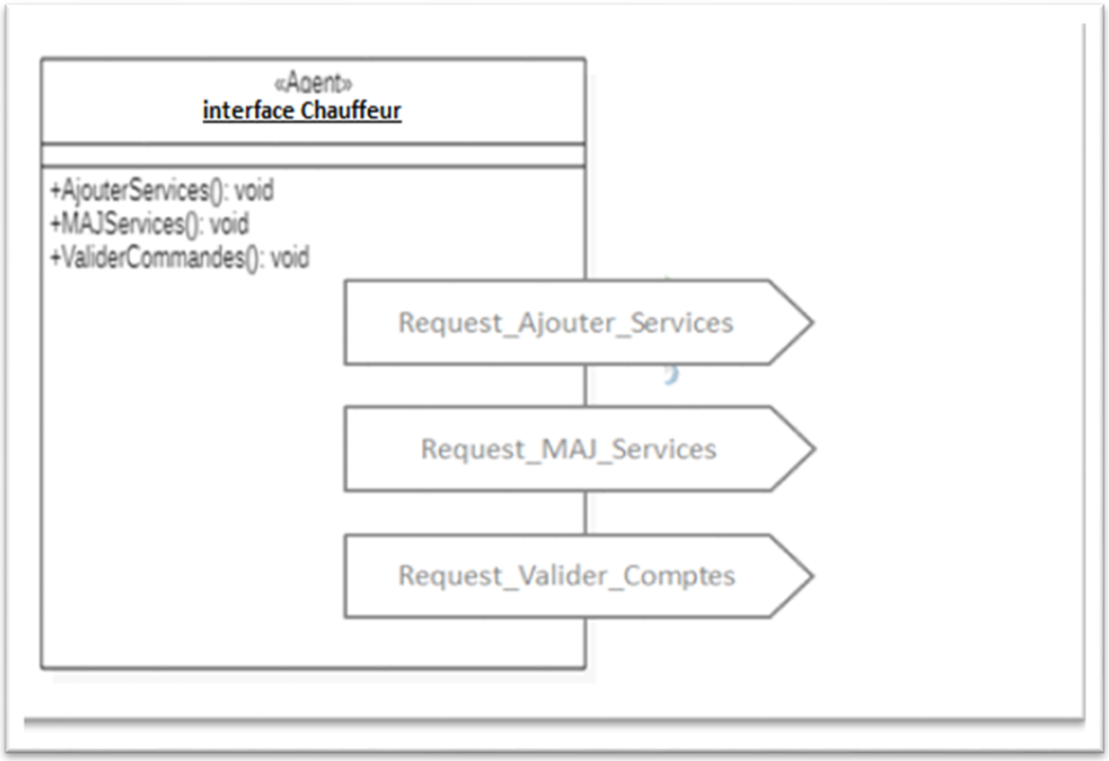


Figure 3.17: Classe Agent Interface Client.



.Figure 3.18 : Classe Agent Interface Chauffeur.

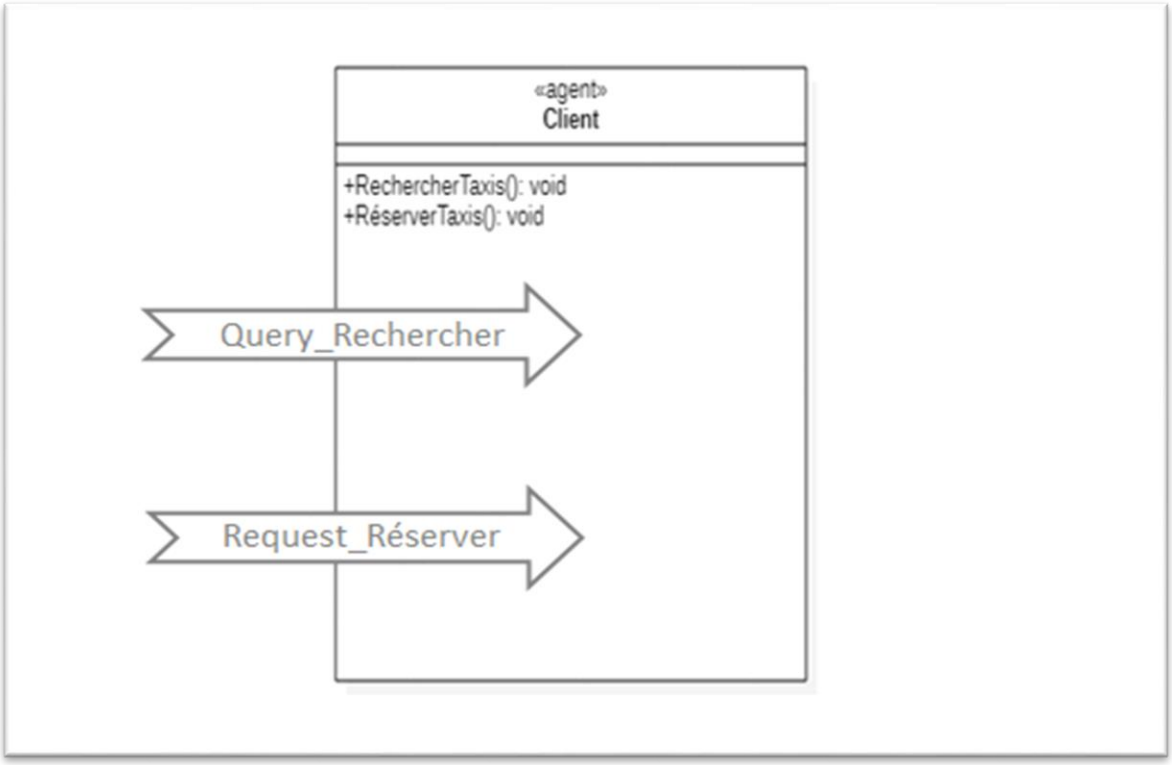


Figure 3.19 : Classe Agent Client.

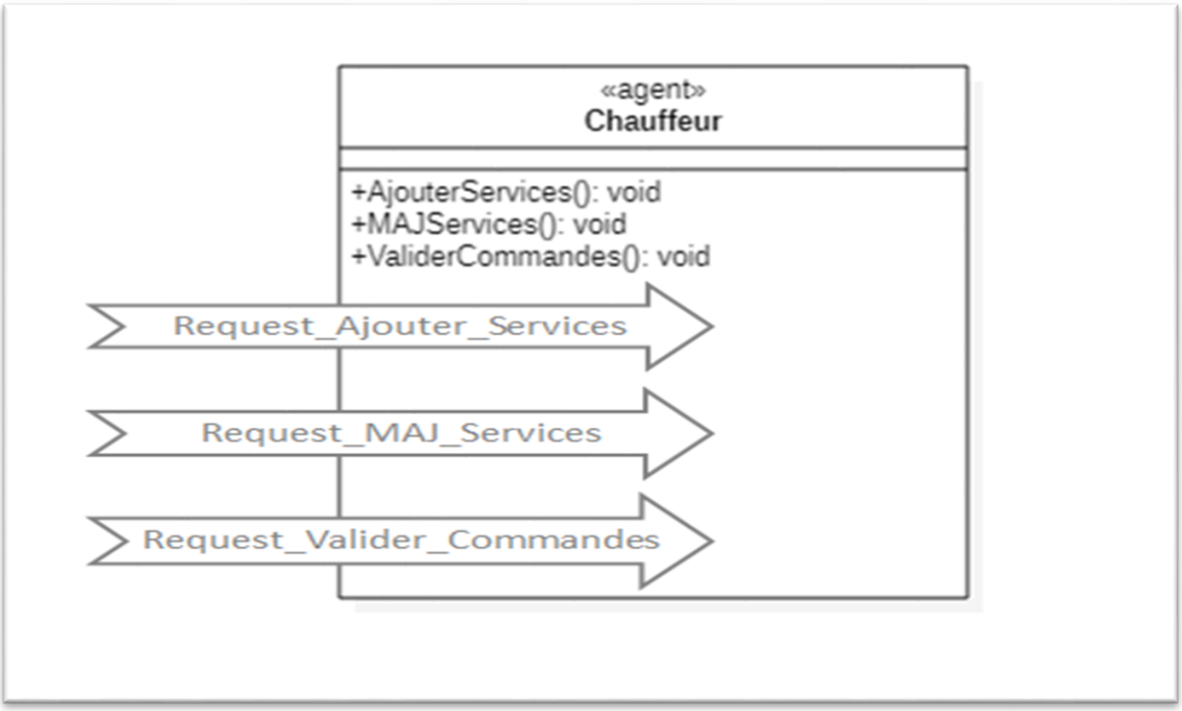


Figure 3.20 : Classe Agent Chauffeur.

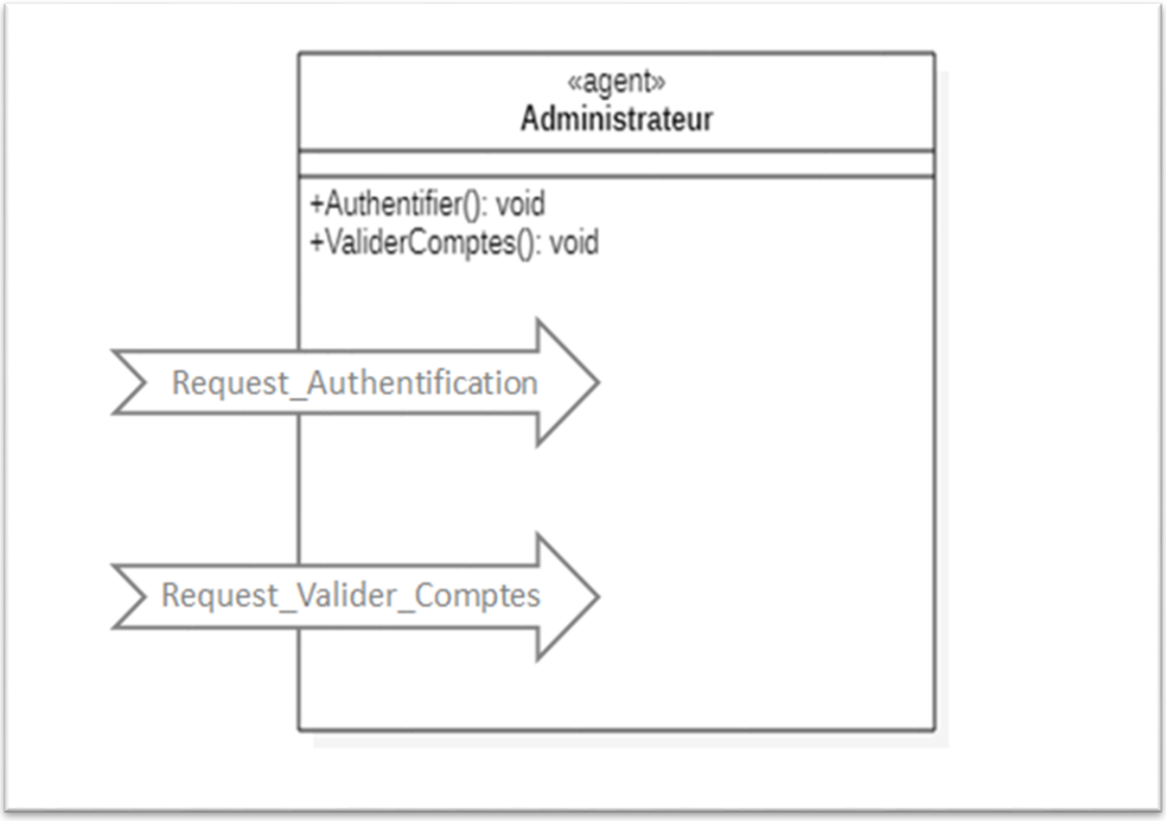


Figure 3.21 : Classe Agent Administrateur.

6. Conclusion :

Dans ce chapitre, nous avons présenté la phase d'analyse et de conception de notre système, en se basant sur la méthodologie voyelle et sur le langage AUML. après l'identification de différentes composantes du système, nous avons élaboré les diagrammes AUML nécessaire pour sa modélisation, à savoir le diagramme de cas d'utilisation, le diagramme de protocole d'interaction et le diagramme de classes. Le chapitre suivant sera consacré à l'implémentation de ce systèm

Chapitre 04

Implémentation.

1. Introduction :

C'est ainsi que nous poursuivons avec la dernière étape de notre travail qui se traduit par la phase de réalisation. Ceci en précisant l'environnement logiciel, les outils de développements utilisés ainsi que les langages de programmation. Ces derniers, nous ont permis d'arriver au résultat voulu, que nous présentons à travers les différentes interfaces réalisées.

2. Langages et outils de développement :

Nous présentons dans cette section les différents langages et outils utiliser pour le développement de notre application :

Langage Java :

JAVA fut le langage de notre choix vu que nous avons choisi préalablement la plateforme JADE, et aussi grâce à sa portabilité et à son aspect Orienté Objet qui facilite la migration d'une conception d'objet vers une implémentation de classe . **[26]**

Plateforme Jade :

Pour développer des agents, il est pratique de choisir une plate-forme. A l'heure actuelle, plusieurs outils ont été développés pour la réalisation d'agents. Si ces plateformes abondent en quantité, elles ne répondent pas toutes aux critères que nous nous sommes fixés. Cette partie présente notre démarche dans le choix d'une plateforme de développement d'agents. La plateforme devra répondre aux contraintes suivantes : **[27]**

- La plateforme doit répondre à plusieurs fonctionnalités et offrir une large gamme de bibliothèques.
- IL faut tenir compte de la nature de l'interface de développement vu le temps court réserver au développement des agents.

- La plateforme doit être répandue. Elle doit avoir été utilisée dans plusieurs projets de développement de systèmes multi-agents pour avoir un maximum d'exemple et de tutoriaux.
- Le langage de programmation sous-jacent et la nature des messages échangés sont aussi un point important.
- Un langage connu et répandu favorisera une meilleure compréhension des codes sources.

Nous avons choisis JADE (Java Agent Developpement Framework) pour les raisons suivantes :

- ✓ Facilité d'installation.
- ✓ Documentation détaillée.
- ✓ Utilise un langage puissant et stable (JAVA).
- ✓ Intégration avec d'autres outils de développement (Intégration dans Eclipse via les plugins EJIP et EJADE).
- ✓ Licence libre (LGPL: (pour GNU Lesser General Public License) : c'est une licence utilisée par certains logiciels libres).

Framework Hibernate :

Framework Hibernate Est un Framework d'ORM pour applications JAVA **. [28]**

- Y a aussi Nhibernate pour les applications .NET
- Appelé une solution de gestion de persistance ou couche de persistance permet de créer une couche d'accès aux données (DAO) plus modulaire,
- Plus maintenable, plus performante en productivité qu'une couche d'accès aux données "classique" reposant sur l'api JDBC.

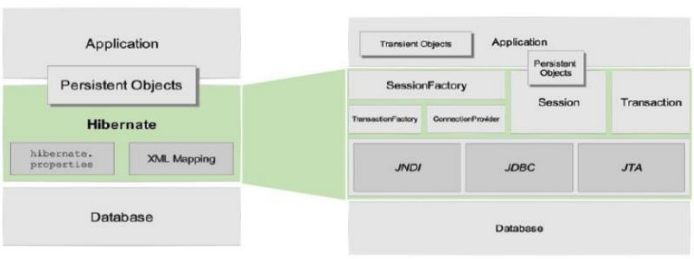


Figure 4.1 : Architecture de Hibernate.

Android studio :

Android Studio est un nouvel environnement pour développement et programmation entièrement intégré qui a été lancé par Google pour les systèmes Android. Il a été conçu pour fournir un environnement de développement et une alternative à Eclipse qui est l'IDE le plus utilisé . **[29]**

- Parmi ses avantages on peut citer les suivants :
  - Un environnement de développement robuste.
  - Permet de voir chacun des changements visuels effectué dans l'application en temps réel.
  - Une manière simple pour tester les performances sur d'autres types d'appareils.
  - Firebase est intégré à l'IDE.
  - Un éditeur complet avec une panoplie d'outils pour accélérer le développement des applications.

MySQL :

MySQL a été lancé à l'origine en 1995. Depuis, il a connu quelques changements de propriétaire et de gestion, avant de se retrouver chez Oracle Corporation en 2010. Alors qu'Oracle est en charge maintenant, MySQL est toujours un **logiciel open source**, ce qui signifie que vous pouvez l'utiliser et le modifier librement. . **[30]**

Figure 4.2 : Interface de MySQL.

L'IDE Eclipse :

L'IDE Eclipse (Integrated Développent Environnent), c'est un environnement de développement intégré (le terme Eclipse désigne également le projet correspondant, lancé par IBM) extensible, universel polyvalent, permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse est principalement écrit en Java, et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions. **[31]**

Eclipse a pour objectif de produire et de fournir des outils pour la réalisation de logiciels, il été conçu uniquement pour produire des environnements de développement, cependant les utilisateurs et les programmeurs se sont rapidement mis à réutiliser ses briques logicielles pour des applications clientes. Alors, on utilise cette Platform pour développer les applications Android, et ça ce qu'on a faits dans notre projet on intégrant ce qu'on appelle le SDK.

Figure 4.3 : Interface d'Eclipse.

3. Implémentation des agents avec Jade :

JADE est un middleware qui facilite le développement des systèmes multi agents (SMA). Dans Jade, les agents sont créés dans des conteneurs où chaque conteneur peut contenir plusieurs agents. L'ensemble de conteneurs constituent une plateforme. Chaque plateforme doit contenir au moins le conteneur principale qui appelé main-container. Ce conteneur qui possède trois modules principaux : **[32]**

• DF « Directory Facilitator » fournit un service de « pages jaunes» à la plate-forme .

• ACC «Agent Communication Channel » gère la communication entre les agents .

• AMS « Agent Management System » supervise l'enregistrement des agents, leur authentification, leur accès et l'utilisation du système. Dans la suite de cette section nous allons présenter brièvement la manière de création des agents du notre système ainsi que le lancement du différent conteneur nécessaire pour l'exécution des agents.

4. Présentation des Interfaces Graphiques :

Interface de connexion à la plateforme jade :

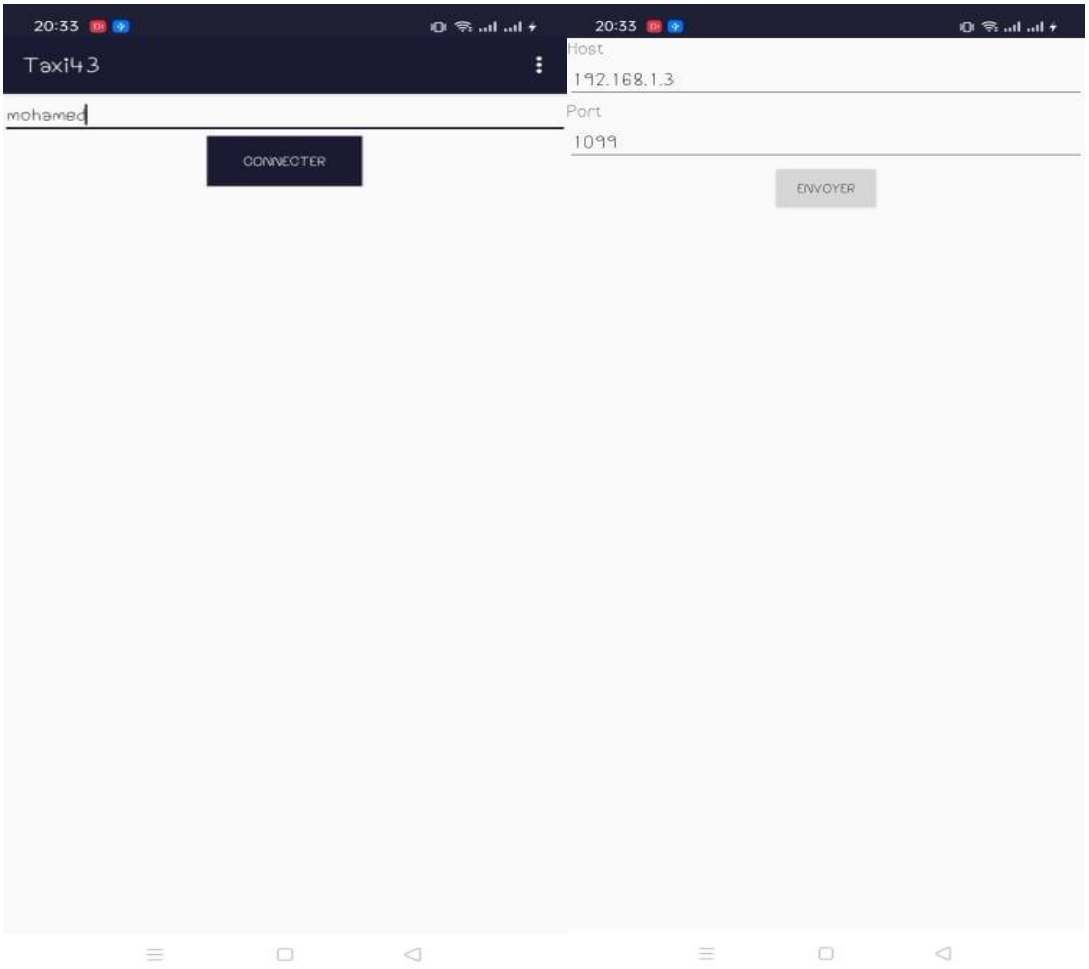


Figure 4.4: Interface de connexion à la plateforme jade.

Interface d'inscription Client :

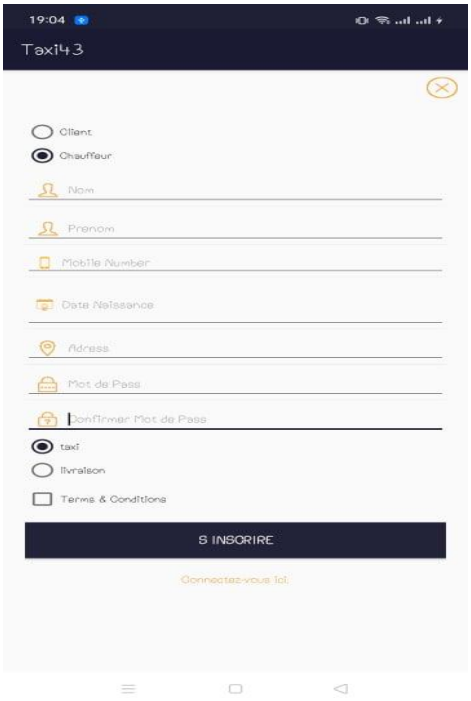


Figure 4.5 : Inscription Client .

Interface d'inscription Chauffeur :

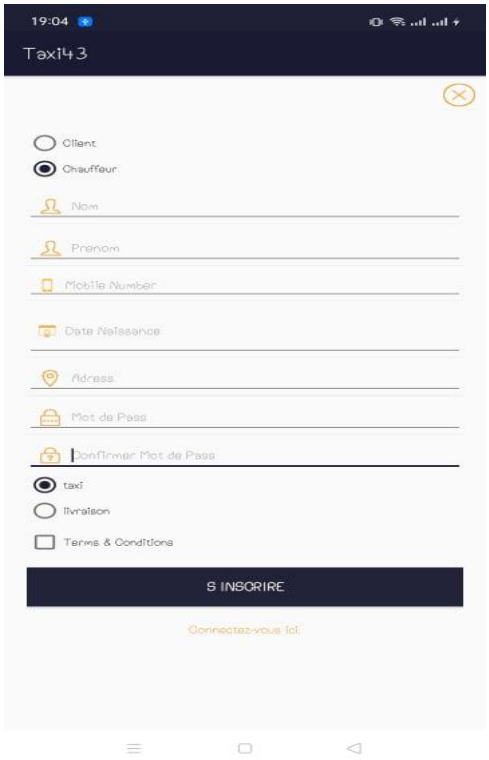


Figure 4.6 : Inscription Chauffeur.

Interface de connexion :

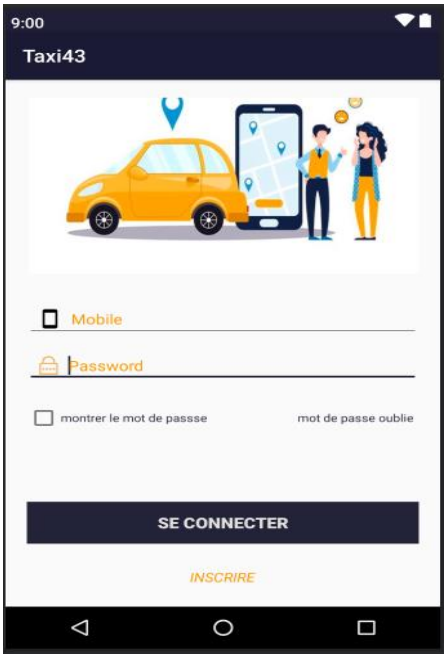


Figure 4.7 : Interface connexion

Interface de Localisation :

Figure 4.8 : Interface de localisation.

Interface de Menu Client :

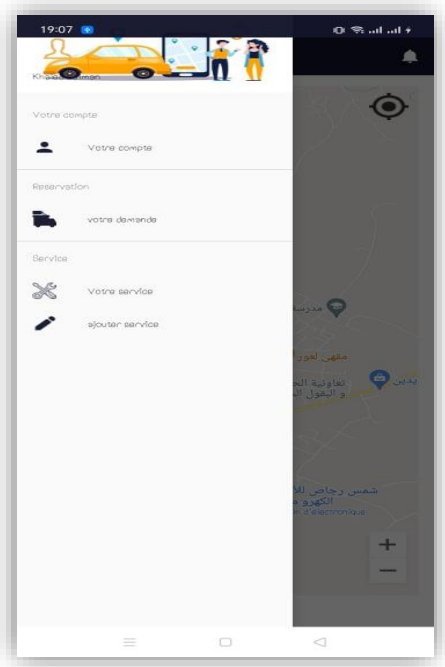


Figure 4.9 : Menu Client .

Interface d'ajouter service:

Figure 4.10 : Interface d'ajouter service .

Interface de Menu Client :

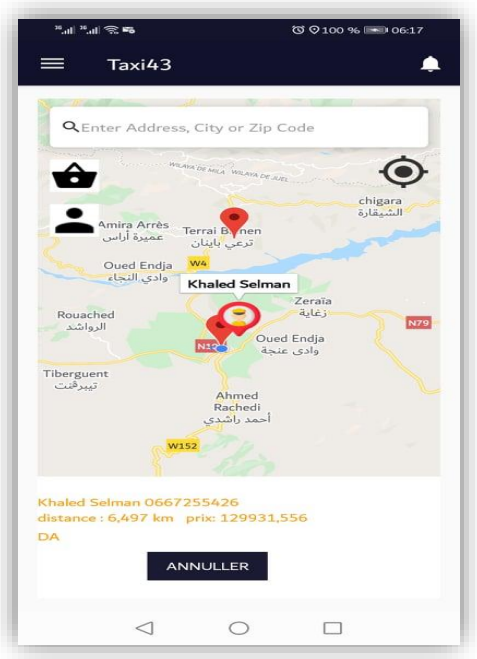


Figure 4.11 : Estimation de prix .

Interface Demande Réservation :

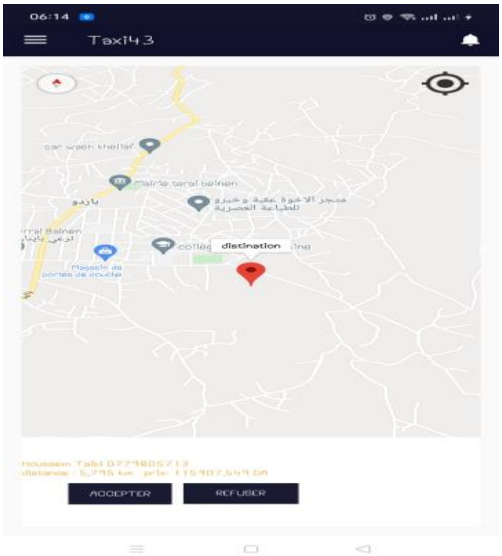


Figure 4.12 : Interface Demande Réservation .

Interface liste de réservation :

Figure 4.13 : Interface Liste de réservation .

5. Conclusion :

Dans ce chapitre, nous avons décrit les aspects pratiques liés à la réalisation de notre application « Taxi43 », à savoir l'environnement, les outils de développement, ainsi que nous avons présenté les principales interfaces de notre application mobile.

Conclusion Générale :

Les innovations technologiques dans le domaine de l'information et de la communication font de plus en plus preuve de leur efficacité et de leur commodité. L'utilisation de ces technologies, notamment la technologie mobile, est devenue incontournable et même indispensable dans plusieurs domaines, comme la réservation en ligne de taxi, qui se présente aujourd'hui comme une tendance prometteuse pour les clients et les chauffeurs de taxi.

L'objectif de notre étude était de proposer une solution efficace pour permettre aux clients d'effectuer des réservations de taxi en utilisant leurs appareilles mobiles. Pour cela, nous avons réalisé dans ce travail une application de réservation en ligne en se basant sur la technologie mobile et sur les systèmes multi-agents. Cette application va permettre en particulier :

- Pour les chauffeurs : de créer des comptes en ligne, d'ajouter des services, de recevoir des notifications et de traiter des commandes passées par les clients.
- Pour les clients : de rechercher des taxis, de faire de réservation en ligne, de voir la localisation géographique des chauffeurs, etc.

Pour la réalisation de ce projet de fin d'étude nous avons suivi les principales étapes suivantes :

- ✓ En premier lieu, nous avons passé en revue les concepts d'application mobile et nous avons extrait d'après cette étude les aspects à prendre en considération lors du développement de notre application.
- ✓ Ensuite, nous avons décrit les notions d'agent et de systèmes multi-agents ainsi que leurs caractéristiques et leurs différents domaines d'application.
- ✓ Pour la conception de notre système, nous avons basé sur la méthode Voyelles et sur le langage AUML et ses différents diagrammes.
- ✓ Enfin, nous avons utilisé la plateforme JADE et plusieurs autres outils de développement pour sa réalisation et nous avons conduit plusieurs tests d'exécution qui ont révélé que le système fonctionne correctement.

Finalement, comme perspectives futures, nous proposons d'étendre l'architecture proposée principalement par :

- ✓ L'intégration des autres types de transports.
- ✓ L'utilisation des mécanismes de payement en ligne.

Bibliographie :

[1] MOUSSOUNI Zobir , RAMDANI Massinissa , Conception et réalisation d'une application mobile pour le service de tourisme, cas d'étude "Wilaya de Bejaia" . Mémoire de Master : Génie Logiciel (2016-2017) . Université Abderrahmane Mira de Béjaïa (1-102 p) .

[2] MOUNSI Massinissa, ZIANI Abid . Conception et réalisation d'une application mobile sous Android pour le m-tourisme« Cas d'étude : Wilaya de Bejaïa » . Diplôme de Master Professionnel en Informatique 2013-2014 .

[3] Dr. KOUAH SOFIA , Chapitre 1 Introduction aux Applications Mobiles 3ieme Année Licence Informatique S.I (2019-2020) . Université larbi ben M'hidi oum EL-bouaghi .(http://www.univ-oeb.dz/fsesnv/wp-content/uploads/2020/04/Chapitre-1-Introduction-aux-applications-mobiles.pdf) .

[4] https://fr.wikipedia.org/wiki/Historique_des_versions_d%27Android, consulte 24/08/2021

[5] https://fr.wikipedia.org/wiki/Historique_des_versions_d%27Android 20/08/2021.

[6] Mr ICHALLALENE Toufik , Conception et réalisation d'une application Android pour la gestion d'unservice de restauration . diplôme de master en Informatique 2014-2015 .

[7] Nicholas Jennings , Katia Sycara , Michael Wooldridge : A Roadmap or agent research and development . Dans les artes de International Conference on Autonomous Agent (AA) Minneapolis , MN ,USA , mai 1998 .ACM

[8] Y. Shoham ``agent-orineted programing ' artificial life 60,1993, (139...159 p) .

[9] M. Wooldridge. An Introduction to Multiagent Systems. John Wiley and Sons, 2002 .

[10] Agent Ibm .

[11] Adina Florea, Daniel Kayser, Stefan Pentiuc et contribution de Amal El Fallah Segrounichi , chapitre 01 - cours en-ligne "Agents Intelligents" , Politechnica University of Bucharest – 2002 .

[12] Tahri Samira , Analyse Hydrodynamique d'un piston de moteur combustion interne par Simulation de système Multi Agent , Mémoire de magister 2005-2006 (1-139p) .

[13] Adina Florea, Daniel Kayser, Stefan Pentiuc et contribution de Amal El Fallah Segrounichi , chapitre 02 - cours en-ligne "Agents Intelligents" , Politechnica University of Bucharest – 2002 .

[14] Ferber.J , " les systèmes multi-agents vers une intelligence collective » interEditions , Paris , 1995 .

[15] B. Chaib-draa, I. Jarras et B. Moulin Article à paraître dans : J. P. Briot et Y. Demazeau « Agent et systèmes multiagents » chez Hermès en 2001

[16] Organisation des Systèmes Multi-Agents Joël Quinqueton LIRMM Montpellier, France (http://www.lirmm.fr/~jq/Cours/3cycle/organisationSMA.pdf) .

[17] Kemoukh Ahmed , Terrouche Salah , Développement d'une application M-commerce à base d'agents Mémoire préparé en vue de l'obtention du diplôme de Master 2019/2020 .

[18] EFACENE Mohamed Lamine ,MODÉLISATION ET RÉALISATION D'UN SYSTÈME D'ENCHERES PAR LES SYSTÈMES MULTI-AGENTS.MEMOIRE DE MASTER :Génie Logiciel, 2016. Université Abderrahmane Mira de Béjaïa (1-7 p) .

[19]https://www.techno-science.net/glossaire-definition/Systeme-multi-agents-page-2.html consulte 06/09/2021

[20] Gasser, L., C. Braganza, N. Herman. MACE: A flexible testbed for distributed AI, Pitman, 1987 .

[21] https://fr.wikipedia.org/wiki/Syst%C3%A8me_multi-agents consulte 06/09/2021

[22] Zerrougui Salim , Une approche pour l'extraction d'Aspects dans les Applications Multi-Agents . MÉMOIRE Pour l'obtention du diplôme de Magister en Informatique .

[23] Mlle. KADDOUR Halima , Mise en œuvre d'une application SMA Dans les réseaux P2P , Diplôme de Master en Informatique 2014-2015 .

[24] https://fr.scribd.com/document/430037622/AUML consulte 20/08/2021.

[25]

[26]http://dspace.univtlemcen.dz/bitstream/112/8178/1/Mise_en_oeuvre_d_une_application_SMA_Dans_les_reseaux_P2P_Supervision%20system.pdf .

[27]http://dspace.univ-tlemcen.dz/bitstream/112/8178/1/Mise_en_oeuvre_d_une_application_SMA_Dans_les_reseaux_P2P_Supervision%20system.pdf .

[28]http://elearning.centre-univ-mila.dz/pluginfile.php/72307/mod_resource/content/1/Persistance%20de%20donn%C3%A9es%20transparente%20Hibernate.pdf .

[29] http ://www.techno-science.net/ ?onglet=glossaire&definition=517. Consulte le 21/07/2021 .

[30]https://kinsta.com/fr/base-de-connaissances/qu-est-ce-que-mysql/#:~:text=MySQL%20a%20%C3%A9t%C3%A9%20lanc%C3%A9%20%C3%A0,utiliser%20et%20le%20modifier%20librement .

[31] Ayoub MKHARBACH , Conception et réalisation d'un site web de gestion des certifications et formations. Faculté des sciences et techniques de settat - Hassan I - Licence Génie informatique 2015 (https://www.memoireonline.com/09/19/10967/m_Conception-et-realisation-d-un-site-web-de-gestion-des-certifications-et-formations23.html Consulte 10/08/2021) .

[32]http://nicolas.thiery.name/Enseignement/CCI-LO/Eclipse.pdf consulte 17/08/2021 . .

[33] **https://www.memoireonline.com/08/11/4782/m_Planification-multi-agents-pour-la-composition-dynamique40.html** consulte 17/09/2021

[34]https://elearning.univ-bejaia.dz/pluginfile.php/273313/mod_resource/content/0/MonCours_JADE.pdf).consulte 17/09/2021

[35]**https://www.memoireonline.com/08/11/4782/m_Planification-multi-agents-pour-la-composition-dynamique40.html).**. 17/09/2021

Annexe A

Agent Unified Modeling Language.

1. Introduction :

Les faiblesses d'UML pour la représentation des systèmes multi-agents ont conduit une équipe de chercheurs travaillant dans différentes entreprises ou universités (Siemens, University Paderborn, Intelligent Automation, Fujitsu...) à concevoir AUML. L'objectif est de mettre au point des sémantiques communes, des méta-modèles et une syntaxe générique pour les méthodologies agents. AUML est un des fruits de la coopération entre FIPA (Fondation of Intelligent Physical Agents) et l'OMG (Object Management Group). Par rapport à UML, AgentUML propose des extensions pour la représentation des agents que nous détaillerons dans cette annexe..

2. AUML :

AUML est base sur la méthode UML (Unified Modeling Language) qui est une méthode de génie logiciel utilisée pour les développements en langages orientes-objets. Elle est déjà largement utilisée par la communauté des concepteurs-objet et son succès continue de croitre. Comme nous l'avons déjà vu, par rapport aux objets, les agents ont des activités autonomes et des buts. C'est cette différence qui entraine l'insuffisance d'UML pour modéliser les agents et les systèmes multi-agents. Aussi AUML remplace-t-il la notion de méthode par celle de service. Ses principales extensions sont :

- Diagramme de classes d'agent qui est une reformulation du diagramme de classes d'objets,
- Diagramme de séquence qui permet une meilleure modélisation des interactions entre agents.
- Diagramme de collaboration qui complémente le diagramme de séquence en proposant une autre lecture et vision des interactions entre agents.

3. Présentation générale des extensions d'AUML :

3.1. Le diagramme de classes d'agent :

Une classe d'agent représente un agent ou un groupe d'agents pouvant jouer un rôle ou avoir un comportement détermine, Une classe d'agent comporte :

— Description de la classe d'agent et des rôles.

— Description de l'état interne.

— Actions, méthodes et services fournis.

— Messages échangés.

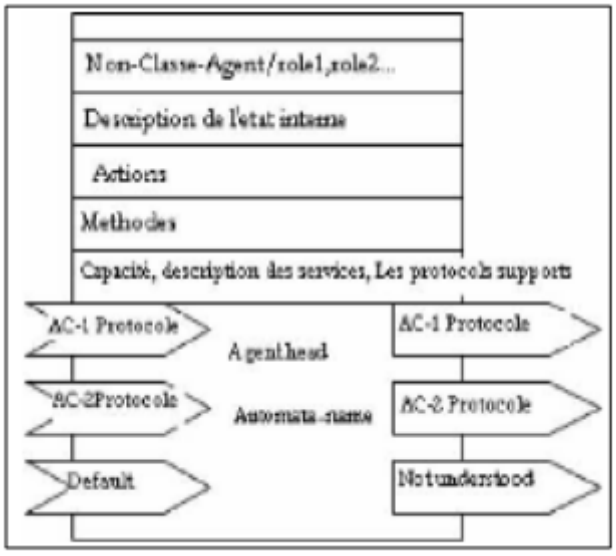


Figure A.1 – Diagramme de classe d'agent.

⇒ **Nom de la classe agent/ role1, role2,... :** Un agent d'une classe donnée peut avoir plusieurs rôles (un détaillent peut être acheteur ou vendeur).

⇒ **Description des états :** Définition de variables d'instance qui reflètent l'état de l'agent.

⇒ **Actions (Plans) :** deux types d'actions peuvent être spécifies : action pro-active exécutée par l'agent lui-même si une pré-condition devient vraie, et réactive résultant

d'un message reçu d'un autre agent. En d'autres termes les actions sont les plans qu'a un agent.

⇒ **Méthodes :** Elles sont définies comme dans UML, avec éventuellement des pré-conditions, post-conditions ou invariants.

⇒ **Envoi et réception de messages :** description des messages émis et reçus par l'agent en précisant les protocoles.

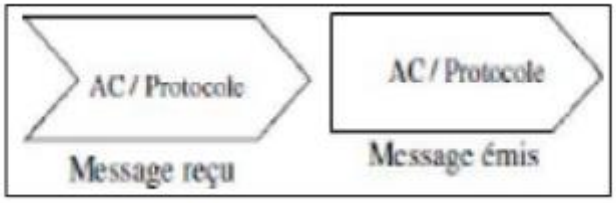


Figure A.2 – messages émis et reçus.

⇒ **Un automate :** représente les changements d'état induits par les échanges de messages.

3.2. Protocoles d'interaction (AIP : Agent Interaction Protocol) :

UML a été étendu afin de représenter la communication simultanée des actes envoyés de l'expéditeur au destinataire.

— Les actes de communications simultanées de CA - 1à CA - n sont envoyés en parallèle.

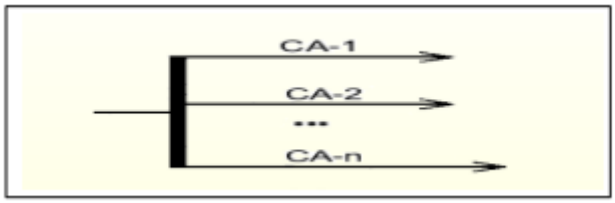


Figure A.3 – Les actes de communication simultanées.

— Une sélection de n actes est envoyé en parallèle (zéro ou plus).

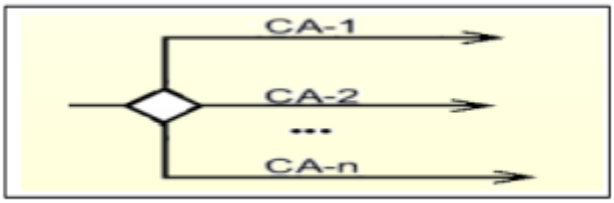


Figure A.4 – Les actes est envoyé en parallèle.

— Choix exclusif : un seul des actes de communication est envoyé.

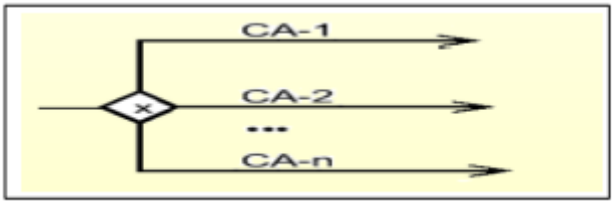


Figure A.5 – Un seul des actes de communication est envoyé.

- **3.2.1 Exemple des interactions :**

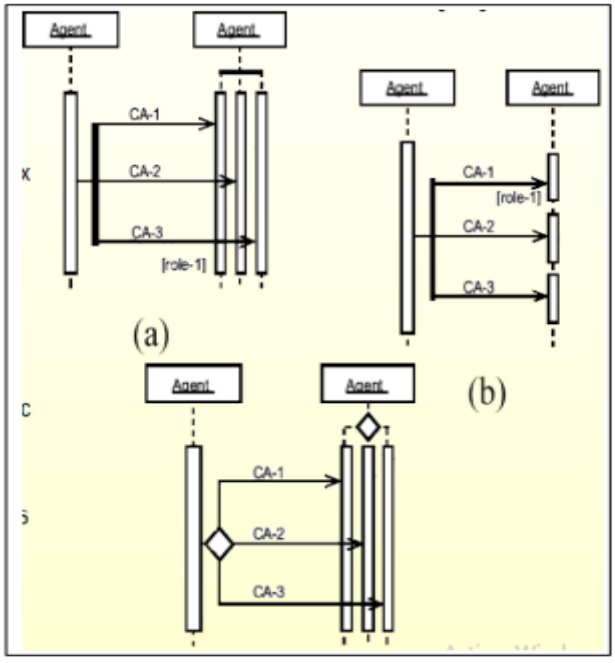


Figure A.6 – Exemple des interactions.

4. AUML et les autres méthodologies :

Les objectifs entre AUML et nombre de méthodologies (Gaia, MaSE, etc.) orientées agent sont communs : partir de méthodologies ou langages de modélisation orientés objet et les étendre au paradigme agent.

5. Limitations d'AUML :

- ✓ Les diagrammes sont des ordonnes et peuvent être mal interprètes.
- ✓ L'expression de toutes les informations nécessaires sur les protocoles peut les rendre illisibles.
- ✓ Les cas de redondance sont difficiles à identifier et corriger ;
- ✓ La description des actions temporelles (telles que le timeout, deadline, ...) est difficile à exprimer.
- ✓ La notion d'historique des échanges n'existe pas.
- ✓ La terminaison des interactions n'est pas toujours spécifiée.

6. Conclusion :

Aujourd'hui AUML n'est pas encore un langage de modélisation fini et adopte par la communauté car aucune spécification finale n'a été publié officiellement mais plusieurs publications sur des extensions d'UML ont êtes publiées par les membres du groupe. Actuellement, leurs travaux se concentrent sur les interactions, plus spécifiquement sur les protocoles d'interaction, mais aussi sur d'autres aspects connexes comme les langages d'interaction, la coordination, les rôles des agents. D'autres travaux portent également sur les architectures, et les ontologies.

Annexe B

Plateformes jade

1. Introduction :

JADE : plate-forme multi-agent créé par le laboratoire TILAB .**[33]**

☞ (Java Agent Development) est un Framework de développement de systèmes multi-agents, open-source et basé sur le langage Java.

☞ Un agent selon JADE est conforme au standard FIPA, possède un cycle de vie, possède un ou plusieurs comportements (*Behaviours*), communique avec des messages de type Agent Communication Langage (*ACL*) et rend des services. Un agent est identifié de manière globale par un nom unique (l'*Agent Identifier* ou *AID*).

☞ JADE possède trois modules principaux (nécessaire aux normes FIPA).

⇒ **AMS «Agent Management System »** : supervise l'enregistrement des agents, leur authentification, leur accès et l'utilisation du système.

⇒ **DF « Directory Facilitator »** : fournit un service de « pages jaunes » à la plateforme.

⇒ **ACC « Agent Communication Channel »** : gère la communication entre les agents.

Ces trois modules sont activés à chaque démarrage de la plate-forme.

2. Architecture Logicielle de JADE :

JADE est un middleware qui facilite le développement des systèmes multi agents (SMA).**[34]**

JADE contient :

- ✓ Un Runtime Environment : l'environnement ou les agents peuvent vivre.
- ✓ Une librairie de classes : que les développeurs utilisent pour écrire leurs agents.
- ✓ Une suite d'outils graphiques : qui facilitent la gestion et la supervision de la plateforme des agents.
- ✓ Chaque instance du JADE est appelée conteneur « **Container** », et peut contenir plusieurs agents.

- ✓ Un ensemble de conteneurs constituent une plateforme.
- ✓ Chaque plateforme doit contenir un conteneur spécial appelé main-container et tous les autres conteneurs s'enregistrent auprès de celui-là dès leur lancement.

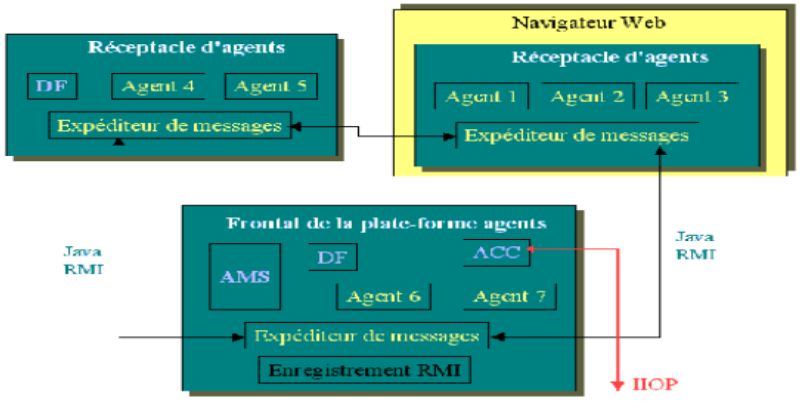


Figure B.1 – Architecture Logicielle de JADE.

3. Outils de JADE :

Pour supporter la tâche difficile du débogage des applications multi-agents, la plateforme possède l'ensemble d'agents prédéfinis suivant : **[35]**

- **RMA «Remote Managment Agent » :** le RMA permet de contrôler le cycle de vie de la plate-forme et tous les agents la composant. Avec son interface, il est possible de créer, supprimer et de migrer des agents. L'agent RMA permet également la création et la Suppression des conteneurs et la fermeture de la plateforme.

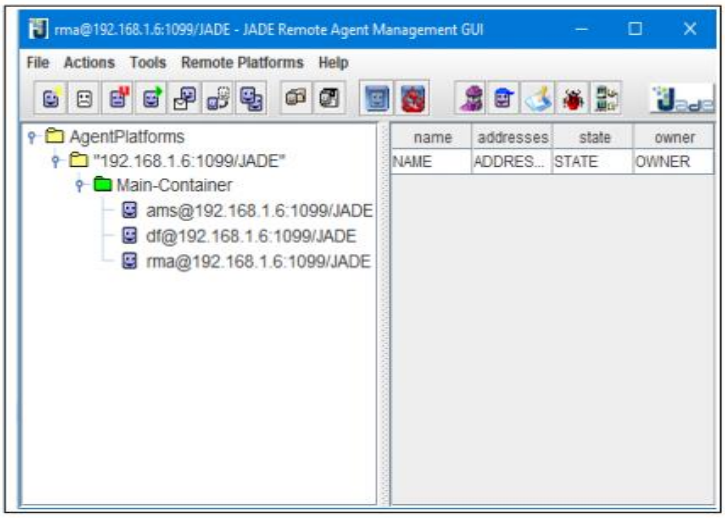


Figure B.2 – L'interface de l'agent RMA.

- **DF « Directory Facilitator » :** enregistre les descriptions et les services des agents et maintien des pages jaunes de services. Grâce à ce service, un agent peut connaître les agents capables de lui fournir les services qu'il requiert pour atteindre son but. L'interface du DF peut être lancée à partir du menu du RMA.

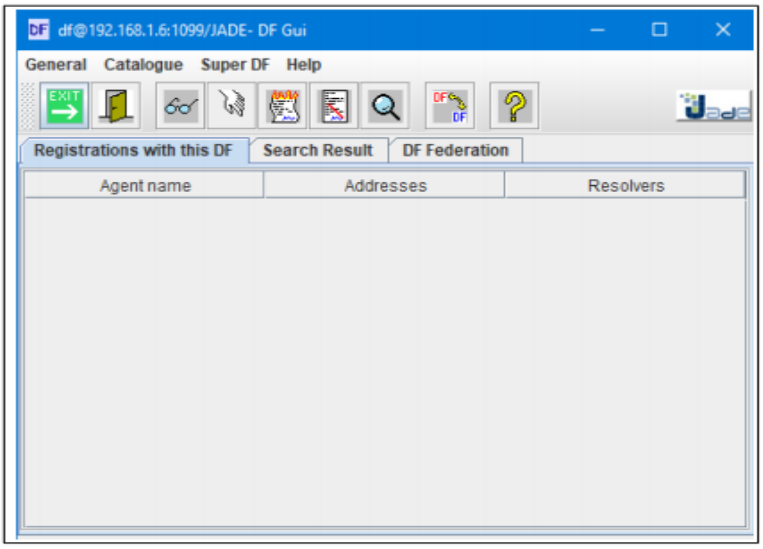


Figure B.3 – L'interface de l'agent DF.

- **DA «Dummy Agent » :** l'outil *DummyAgent* permet aux utilisateurs d'interagir avec les agents JADE d'une façon particulière. L'interface permet la composition et l'envoi de messages ACL et maintient une liste de messages ACL envoyés et reçus. Cette liste peut être examinée par l'utilisateur et chaque message peut être vu en détail ou même édité. Plus encore, le message peut être sauvegardé sur le disque et renvoyé plus tard.

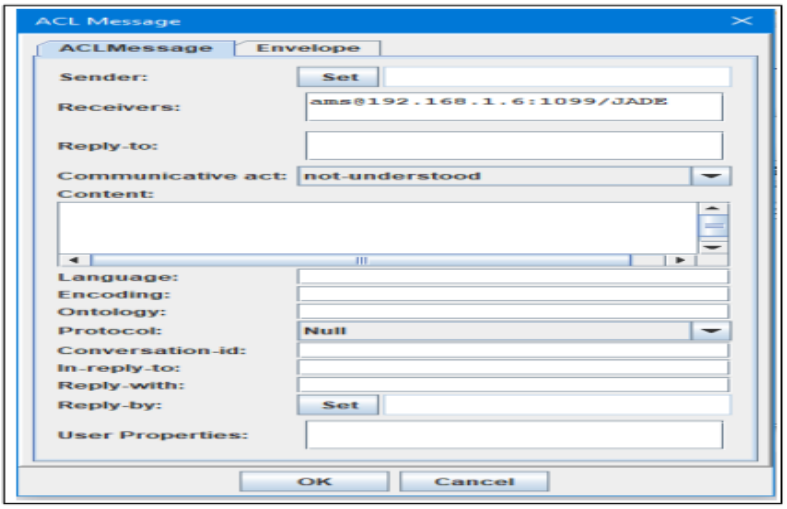


Figure B.4 – L'interface de l'agent Dammy.

- **SA «Sniffer Agent » :** quand un utilisateur décide d'épier un agent ou un groupe d'agents, il utilise un agent Sniffer. Chaque message partant ou allant vers ce groupe est capté et affiché sur l'interface du sniffer. L'utilisateur peut voir et enregistrer tous les messages, pour éventuellement les analyser plus tard.

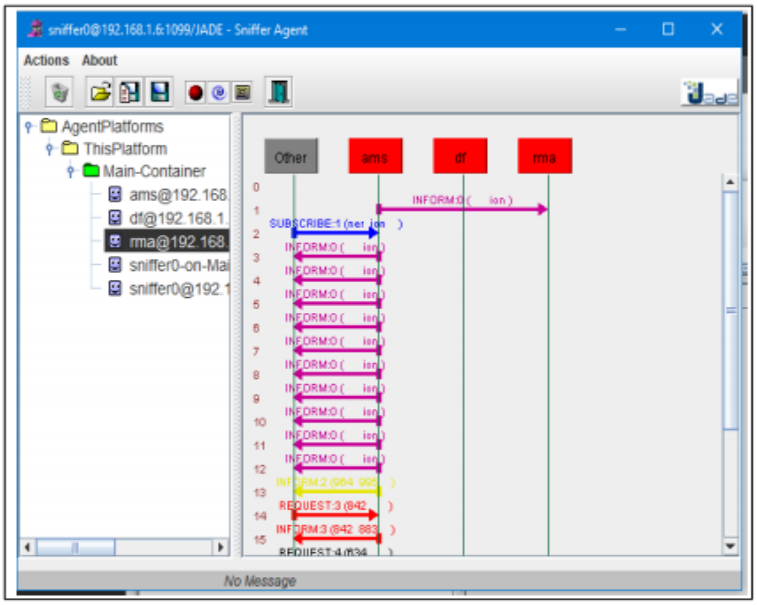


Figure B.5– L'interface de l'agent Sniffer.

- **Introspector Agent :** Cet agent permet de gérer et de contrôler le cycle de vie d'un agent s'exécutant et de la file de ses messages envoyés et reçus.

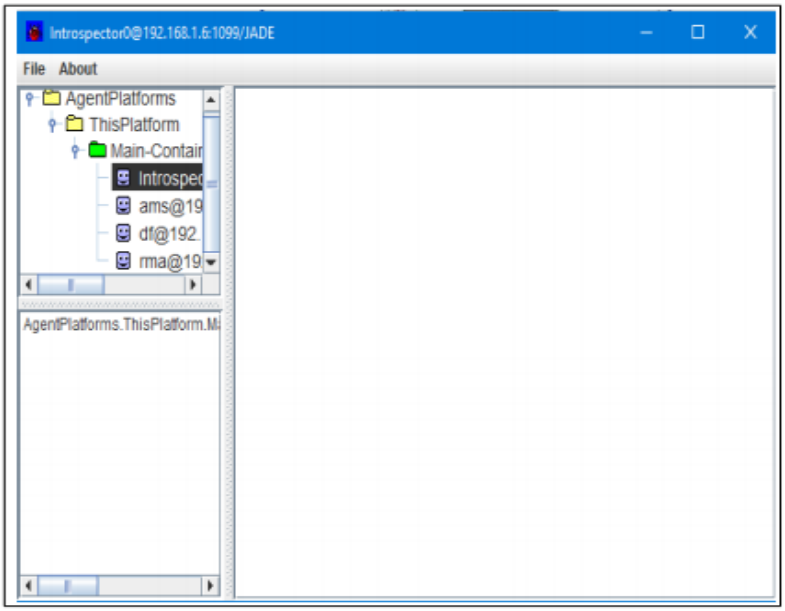


Figure B.6 – L'interface de l'agent Introspector.

4. Lancer la plateforme jade sur ordinateur :

Ajouter jad.jar sur une application java.

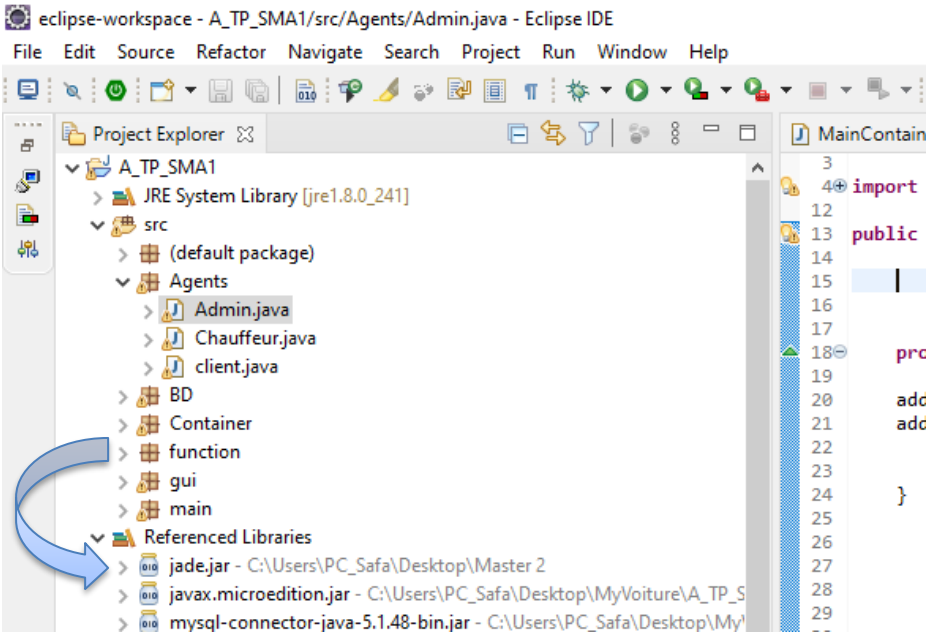


Figure B.7 – Ajouter jad.jar sur une application java.

4.1. Démarrer le MainContainer :

```
package main;

import jade.core.Profile;

public class MainContainer {
    public static void main(String[] args){
        try {
        Runtime runtime= Runtime.instance();
        Properties properties= new ExtendedProperties();
        properties.setProperty(Profile.GUI, "true");

        ProfileImpl profileImpl =new ProfileImpl(properties);
        profileImpl.setParameter(profileImpl.MAIN_HOST, "192.168.1.1");
        AgentContainer agentContainer= runtime.createMainContainer(profileImpl);
        agentContainer.start();

        }catch(ControllerException e) {
            e.printStackTrace();
        }
    }
```

Figure B.8 – Démarrer le MainContainer.

4.2. Démarrer un AgentContainer :

```
package Container;

import Agents.Admin;

public class AdminContainer {
    public static void main(String[] args){

        try {
            Runtime runtime= Runtime.instance();
            ProfileImpl profileImpl =new ProfileImpl(false);
            profileImpl.setParameter(ProfileImpl.MAIN_HOST, "localhost");
            AgentContainer agentContainer= runtime.createAgentContainer(profileImpl);
            AgentController agentController=agentContainer.createNewAgent("admin",
                    Admin.class.getName(), new Object[] {});

            //AgentController agentController1=agentContainer.createNewAgent("Chauffeur",
            //      Chauffeur.class.getName(), new Object[] {});
            agentController.start();
            //agentController1.start();
        } catch (ControllerException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Figure B.9 – Démarrer un AgentContainer.

4.3. Créé premier Agent :

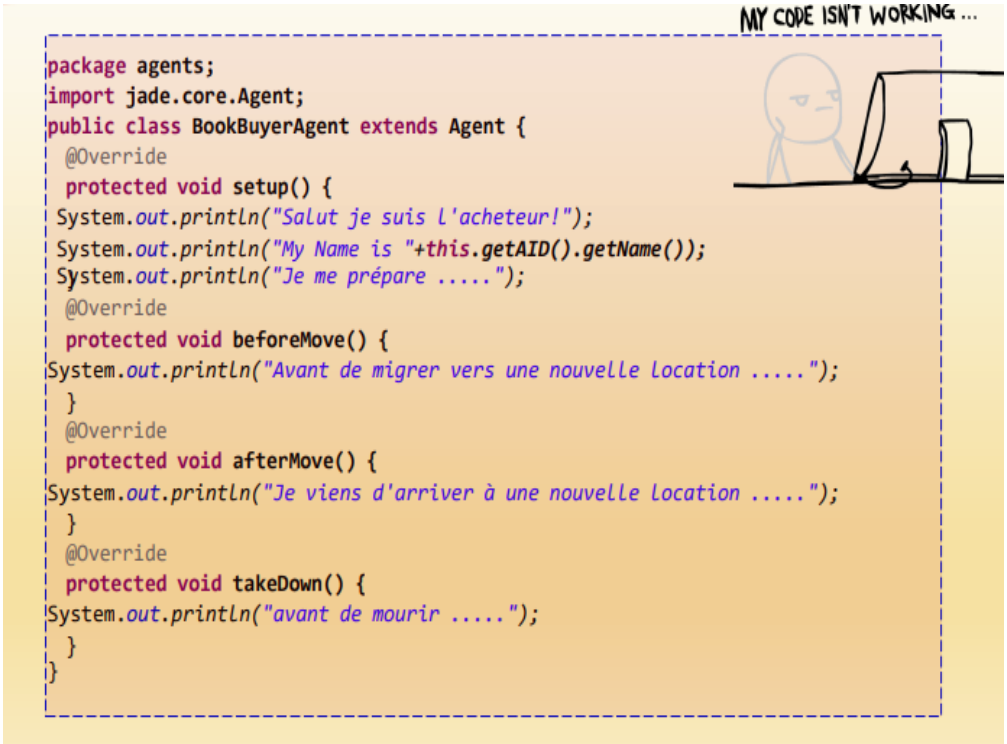
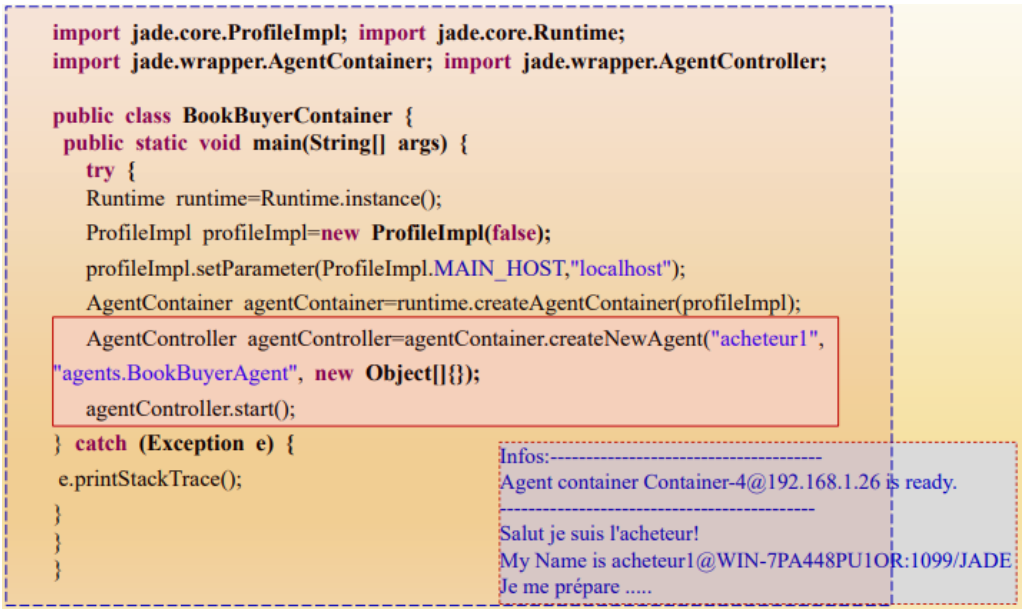


Figure B.10 – créé premier agent.

4.3.1. Déployer l'agent dans une application :



```
import jade.core.ProfileImpl; import jade.core.Runtime;
import jade.wrapper.AgentContainer; import jade.wrapper.AgentController;

public class BookBuyerContainer {
 public static void main(String[] args) {
   try {
   Runtime runtime=Runtime.instance();
   ProfileImpl profileImpl=new ProfileImpl(false);
   profileImpl.setParameter(ProfileImpl.MAIN_HOST,"localhost");
   AgentContainer agentContainer=runtime.createAgentContainer(profileImpl);
   AgentController agentController=agentContainer.createNewAgent("acheteur1",
"agents.BookBuyerAgent", new Object[]{});
   agentController.start();
} catch (Exception e) {
 e.printStackTrace();
}
}
}
```

```
Infos:----------------------------------
Agent container Container-4@192.168.1.26 is ready.
----------------------------------------
Salut je suis l'acheteur!
My Name is acheteur1@WIN-7PA448PU1OR:1099/JADE
Je me prépare .....
```

Figure B.11 – Déployer l'agent dans une application.

4.3.2. Affecter les comportements à un agent :

Pour qu'un agent JADE exécute une tâche, nous avons tout d'abord besoin de définir ces tâches. Les tâches dans JADE (appelées behaviours ou des comportements) sont des instances de la classe jade.core.Behaviours . pour qu'un agent exécute une tâche on doit lui l'attribuer par la méthode addBehaviour (Behaviour b) de la classe jade.core.Agent.

Chaque Behaviour doit implémenter au moins les deux méthodes :

- ✓ **Action () :** qui désigne les opérations à exécuter par le Behaviour .
- ✓ **done() :** qui exprime si le Behaviour a terminé son exécution ou pas.

Il existe deux autres méthodes dont l'implémentation n'est pas obligatoire mais qui peuvent être très utiles :

- ✓ **onStart () :** appelée juste avant l'exécution de la méthode action ().
- ✓ **onEnd () :** appelée juste après la retournement de true par la méthode done ().

Des fois on a besoin de savoir quel est le propriétaire d'un Behaviour, et cela peut être connu par le membre myAgent du Behaviour en question. JADE alloue un thread par agent, pour cela un agent exécute un Behaviour à la fois. L'agent peut exécuter plusieurs Behaviours simultanément en choisissant un bon mécanisme de passation d'un Behaviour à un autre (c'est à la charge du programmeur et non pas à la charge du JADE). Affecter les comportements à

un agent Possibilite d'ajouter/retirer des comportements a un agent en cours d'execution. un agent "dort" s'il n'a pas de comportement à exécuter.

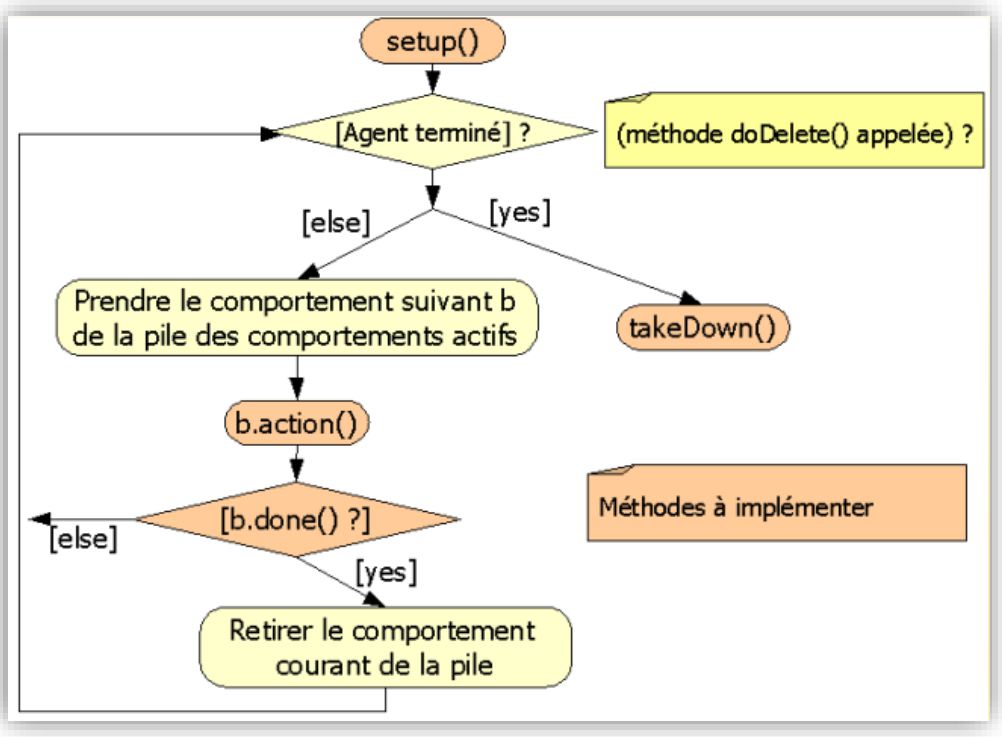


Figure B.12 – Cycle de vie d'un comportement d'un agent.

4.3.3. Behaviours :

JADE offre trois types de Behaviours simple. Ces Behaviours sont :

- ✓ **One-shot Behaviour :**

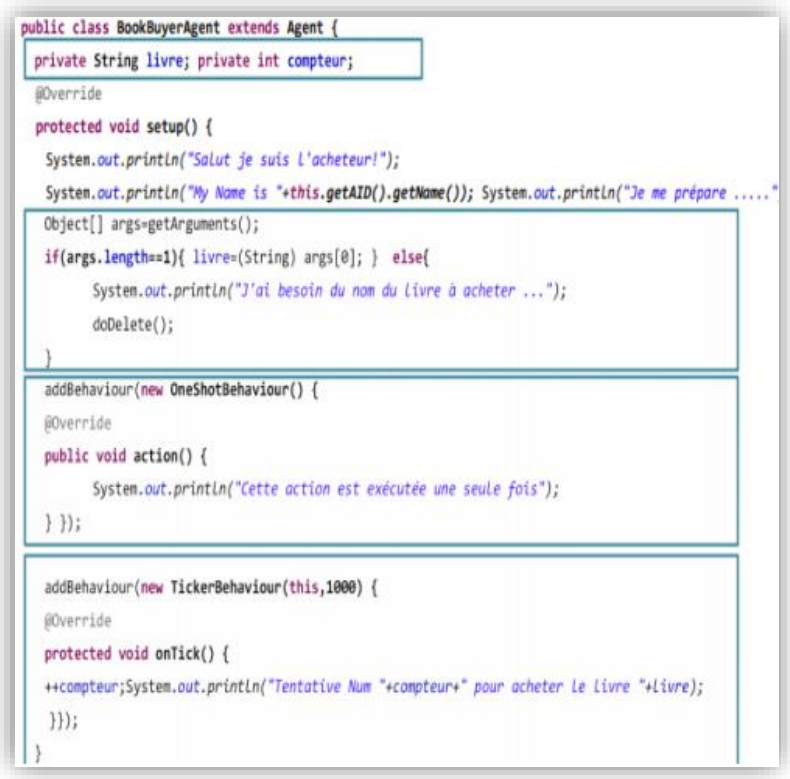
Un one-shot Behaviour est une instance de la classe jade.core.behaviours.OneShotBehaviour. Il a la particularité d'exécuter sa tâche une et une seule fois puis il se termine. La classe OneShotBehaviour implémente la méthode done() et elle retourne toujours true.

- ✓ **Cyclic Behaviour**

Un cyclic Behaviour est une instance de la classe jade.core.behaviours.CyclicBehaviour un cyclic Behaviour exécute sa tâche d'une manière répétitive. La classe CyclicBehaviour implémente la méthode done() qui retourne toujours false.

✓ **Generic Behaviour :**

Un Generic Behaviour est une instance de la classe jade.core.behaviours.Behaviour. Le Generic Behaviour vient entre le One-shot Behaviour et le Cyclic Behaviour de faite qu'il n'implémente pas la méthode done() et laisse son implémentation au programmeur, donc il peut planifier la terminaison de son Behaviour selon ces besoin.



```
public class BookBuyerAgent extends Agent {
private String livre; private int compteur;
@Override
protected void setup() {
  System.out.println("Salut je suis l'acheteur!");
  System.out.println("My Name is "+this.getAID().getName()); System.out.println("Je me prépare ....."
  Object[] args=getArguments();
  if(args.length==1){ livre=(String) args[0]; } else{
        System.out.println("J'ai besoin du nom du livre à acheter ...");
        doDelete();
  }
  addBehaviour(new OneShotBehaviour() {
  @Override
  public void action() {
        System.out.println("Cette action est exécutée une seule fois");
  } });
  addBehaviour(new TickerBehaviour(this,1000) {
  @Override
  protected void onTick() {
  ++compteur;System.out.println("Tentative Num "+compteur+" pour acheter le livre "+livre);
   }));
}
```

Figure B.13 – Exemple Behaviours.

4.3.4. **Communication entre agents** :

La communication entre agents se fait par un langage qui est FIPA-ACL (Agent Communication language). La classe ACLMessage représente les messages qui peuvent être échangés par les agents. La communication de messages se fait en mode asynchrone. Lorsqu'un agent souhaite envoyer un message, il doit créer un nouvel objet ACLMessage, compléter ces champs avec des valeurs appropriées et enfin appeler la méthode send(). Lorsqu'un agent souhaite recevoir un message, il doit employer la méthode receive() ou la méthode blockingReceive().

Un message ACL dispose obligatoirement des champs suivants :

Performative	**Type de l'acte de communication**
Sender	Expéditeur du message.
Receiver	Destinataire du message.
Reply-to	Participant de la communication.
Content	Contenu du message.
Language	Description du contenu.
Encoding	Description du contenu.
Ontology	Description du contenu.
Protocol	Contrôle de la communication.
Conversation-id	Contrôle de la communication.
Reply-with	Contrôle de la communication.
In-reply-to	Contrôle de la communication.
Reply-by	Contrôle de la communication.

Table B.1 : Les champs d'un message ACL

4.3.4.1. L'envoi d'un message :

Pour envoyer un message, il suffit de remplir les champs nécessaires (l'ensemble des récepteurs et le contenu du message et l'acte de communication) d'un message JADE, puis d'appeler la méthode send () de la classe Agent. Voici un exemple d'envoi d'un message JADE.

```
ACLMessage message = new ACLMessage(ACLMessage.INFORM);
message.addReceiver(new AID("vendeu1", AID.ISLOCALNAME));
message.setContent("Livre XML");
send(message);
```

Figure B.14 – L'envoi d'un message.

4.3.4.2. Réception d'un message :

La réception d'un message est aussi simple que l'envoi.

Il suffit d'appeler la méthode receive () de la classeAgent pour récupérer le premier message non encore lu de l'agent.

```
ACLMessage msg = receive();

// Pour envoyer la réponse :
ACLMessage message = new ACLMessage(ACLMessage.INFORM);

message.addReceiver(msg.getSender());

message.setContent("400");

send(message);
```

Figure B.15 – Réception d'un message.

4.3.4.3. L'attente d'un message :

Il se peut qu'un agent doive effectuer un certain traitement ou lancer quelques tâches après avoir reçu un message d'un autre agent.

Il est possible de faire une attente active jusqu'à l'arrivé du message en utilisant la méthode block () :

```
addBehaviour(new CyclicBehaviour() {
 @Override
 public void action() {
  System.out.println("Cyclic behavior");
  ACLMessage msg=receive();
  if(msg!=null){
    System.out.println("Réception du message :"+msg.getContent());
  }
 else{
  block();
}} });
```

Figure B.16 – L'attente d'un message.

4.3.5. Message Template :

- Plusieurs messages peuvent arriver dans la file d'attente.
- Plusieurs agents tentent d'accéder aux messages qui arrivent, il est important de disposer d'un moyen qui permet à un agent de filtrer les messages qui le concernent dans un contexte quelconque.
- Les Template de messages permettent de résoudre ce problème.

- La classe MessageTemplate dispose d'un ensemble de méthodes statiques de fabrique de d'objet MessageTemplate selon différents critères.

```
addBehaviour(new CyclicBehaviour() {
 @Override
 public void action() {
    MessageTemplate mt =
    MessageTemplate.MatchPerformative(ACLMessage.CFP);
    ACLMessage msg = myAgent.receive(mt);
    if (msg != null) {
     // CFP Message received. Process it

    }
    else {
     block();
    }
  }
 }
```

Figure B.17 :Message Template (Filtrer les message).

5. Lancer la plateforme jade sur mobile (JADELEAP) :

JADE-LEAP (Lightweight and Extensible Agent Platform) est une extension de la plateforme JADE qui peut être déployée non seulement sur des PC et des serveurs, mais également sur des périphériques de ressources légers tels que les téléphones mobiles compatibles Java. Pour ce faire, JADE-LEAP peut être mis en forme de différentes manières correspondant aux deux configurations de Java Micro Edition et de la machine virtuelle Java Android Dalvik.

6. Lancer la plateforme JADE-LEAP sur mobile :

— Ajouter JADE-LEAP sur une application java.

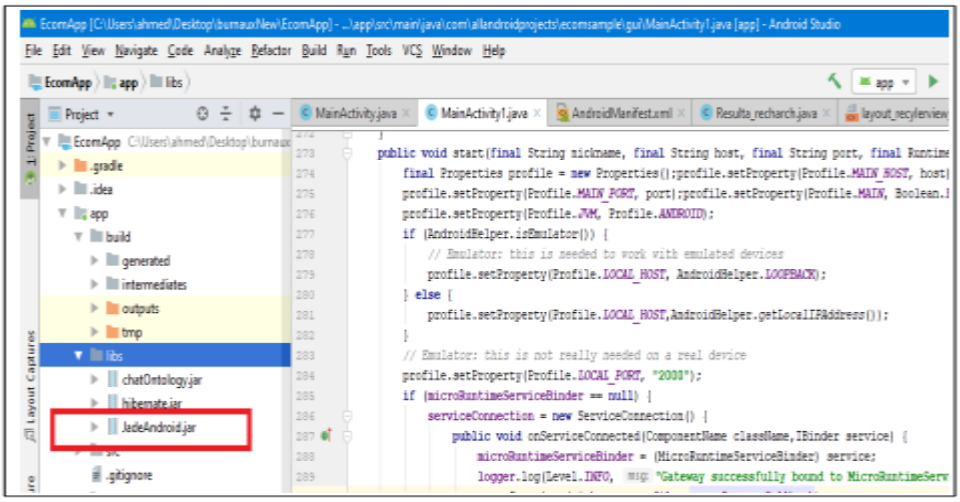


Figure B.18 :Ajouter JADE-LEAP sur une application java

7. Démarrer un AgentContainer et agent :

— Lancer service MicroRuntimeService.

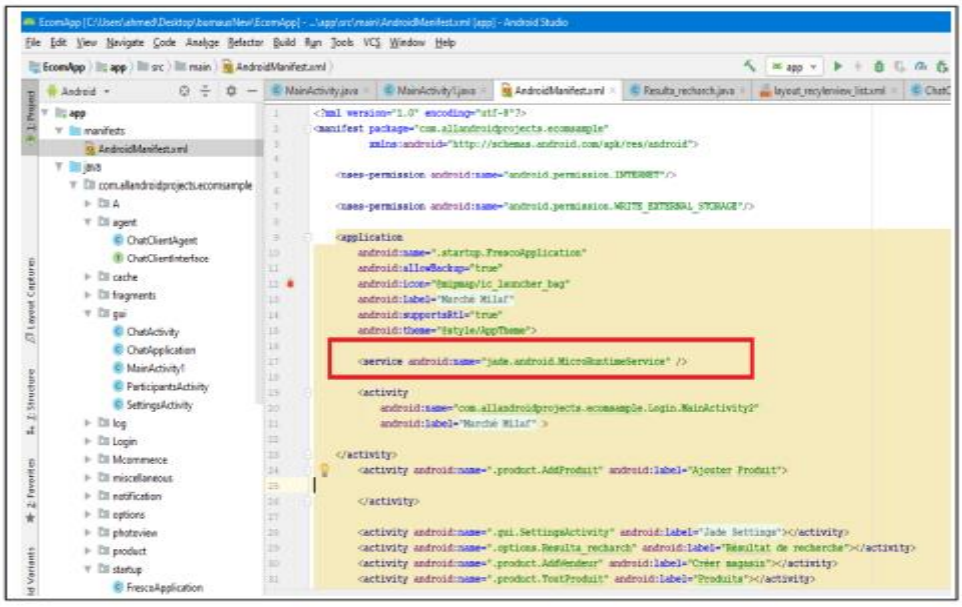


Figure B.19 :Déclarer MicroRuntimeService in AndroidManifest.

— Lancer un Container et Agent.

— Initialisation.

```
public void start(final String nickname, final String host, final String port, final RuntimeCallback<AgentController> agentS
    final Properties profile = new Properties();profile.setProperty(Profile.MAIN_HOST, host);
    profile.setProperty(Profile.MAIN_PORT, port);profile.setProperty(Profile.MAIN, Boolean.FALSE.toString());
    profile.setProperty(Profile.JVM, Profile.ANDROID);
    if (AndroidHelper.isEmulator()) {
        // Emulator: this is needed to work with emulated devices
        profile.setProperty(Profile.LOCAL_HOST, AndroidHelper.LOOPBACK);
    } else {
        profile.setProperty(Profile.LOCAL_HOST,AndroidHelper.getLocalIPAddress());
    }
    // Emulator: this is not really needed on a real device
    profile.setProperty(Profile.LOCAL_PORT, "2000");
    if (microRuntimeServiceBinder == null) {
        serviceConnection = new ServiceConnection() {
            public void onServiceConnected(ComponentName className,IBinder service) {
                microRuntimeServiceBinder = (MicroRuntimeServiceBinder) service;
                logger.log(Level.INFO,  msg: "Gateway successfully bound to MicroRuntimeService");
                startContainer(nickname, profile, agentStartupCallback);
            };
            public void onServiceDisconnected(ComponentName className) {
                microRuntimeServiceBinder = null;
                logger.log(Level.INFO,  msg: "Gateway unbound from MicroRuntimeService");
            }
        };
        logger.log(Level.INFO,  msg: "Binding Gateway to MicroRuntimeService...");
        bindService(new Intent(getApplicationContext(),
                MicroRuntimeService.class), serviceConnection,
                Context.BIND_AUTO_CREATE);
    } else {
        logger.log(Level.INFO,  msg: "MicroRuntimeGateway already binded to service");
        startContainer(nickname, profile, agentStartupCallback);
    }
```

Figure B.20: Initialisation.

— Lancer un Container.

```
private void startContainer(final String nickname, Properties profile,
                            final RuntimeCallback<AgentController> agentStartupCallback) {
    if (!MicroRuntime.isRunning()) {
        microRuntimeServiceBinder.startAgentContainer(profile,
                new RuntimeCallback<Void>() {
                    @Override
                    public void onSuccess(Void thisIsNull) {
                        logger.log(Level.INFO, msg: "Successfully start of the container...");
                        startAgent(nickname, agentStartupCallback);
                    }

                    @Override
                    public void onFailure(Throwable throwable) {
                        logger.log(Level.SEVERE, msg: "Failed to start the container...");
                    }
                });
    } else {
        startAgent(nickname, agentStartupCallback);
    }
}
```

Figure B.21:Lancer un Container.

— Lancer un agent.

```
private void startAgent(final String nickname,
        final RuntimeCallback<AgentController> agentStartupCallback) {
    microRuntimeServiceBinder.startAgent(nickname,
            ChatClientAgent.class.getName(),
            new Object[] { getApplicationContext() },
            new RuntimeCallback<Void>() {
                @Override
                public void onSuccess(Void thisIsNull) {
                    logger.log(Level.INFO, msg: "Successfully start of the "
                            + ChatClientAgent.class.getName() + "...");
                    Intent showChat = new Intent( packageContext: MainActivity1.this,
                            MainActivity2.class);
                    showChat.putExtra( name: "nickname", nickname);
                    MainActivity1.this
                            .startActivityForResult(showChat, CHAT_REQUEST);
                    try {
                        agentStartupCallback.onSuccess(MicroRuntime
                                .getAgent(nickname));
                    } catch (ControllerException e) {
                        // Should never happen
                        agentStartupCallback.onFailure(e);
                    }
                }

                @Override
                public void onFailure(Throwable throwable) {
                    logger.log(Level.SEVERE, msg: "Failed to start the "
                            + ChatClientAgent.class.getName() + "...");
                    agentStartupCallback.onFailure(throwable);
                }
```

Figure B.22 – Lancer un agent.

Note : Déclaration agent dans mobile ressemble déclaration dans ordinateur.

8. Conclusion :

Le rôle de la plate-forme JADE est de réaliser la couche communication inter-agents de façon à pouvoir faire une argumentation sans l'intervention de l'humain et donc dans cette optique de les rendre autonome. On se sert pour cela du langage ACL pour que les agents s'échangent leurs messages