République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mouloud MAMMERI de Tizi-Ouzou
X • Θ Λ • ∑ X [:// :V •X [•Λ [•O
Faculté de Génie Électrique et Informatique
Département Informatique

Mémoire

De fin d'études

En Vue de l'Obtention du Diplôme de Master en Informatique

Option : Réseaux, Mobilité, et Systèmes Embarqués

Thème

Conception et Réalisation d'une application mobile sous ANDROID. Pour : La téléphonie IP et la messagerie instantanée

Proposé et dirigé par :
M^me^ : M.Belkadi.

Réalisé par :
M^r^ : SEBAOUI Karim.
M^r^ : SAIDANI Elyas.

Soutenu publiquement le : 23/10/2014, devant le Jury composé de :

M^r^ : M. DAOUI	**Président**
M^me^ : R.AOUDJIT	**Examinatrice**
M^me^ : R.HADAOUI	**Examinatrice**

Promotion 2013/2014

Remerciements

D'abord nous remercions le bon dieu qui nous a donné le courage, la patience et surtout longue vie pour faire ce travail.

Nous tenons à remercier et à exprimer notre profonde gratitude à notre promotrice Madame M.Belkadi qui nous a suivi durant toutes les phases de notre projet où elle nous a orienté et donné de précieux conseils. Nous la prions de bien vouloir agréer le témoignage de notre plus vive reconnaissance.

Nous remercions les membres de jury qui nous ont fait honneur de bien avoir accepter de juger notre travail. Ainsi tous les enseignants qui ont contribués à notre formation.

Enfin. Nous remercions toute personne qui a contribué de près ou de loin à la réalisation de notre travail. Surtout Monsieur Medjber Amirouche.

Dédicaces :

Je dédie ce modeste travail :

A ma très chère mère qui s'est tant sacrifiée pour les besoins de nos études.
A mon meilleur guide qui me montre le chemin dans les nuits obscures, à mon père.
A mes chers frères.
A mes très chères sœurs.
A tous les membres de ma famille.
A tous mes amis particulièrement Menad.

karim

SOMMAIRE

CHAPITRE II : LA TELEPHONIE IP ET LA MESSAGERIE INSTANTANEE

CHAPITRE III : ANALYSE ET CONCEPTION

CHAPITRE IV : REALISATION

Depuis quelques années, le marché des téléphones portables (désormais appelés Smartphones littéralement téléphones intelligents en français) connaît une véritable révolution, menée par Samsung, Apple et Nokia. Les constructeurs ne cessent de redoubler d'efforts pour mettre en avant leurs produits qui tout en assurant la fonctionnalité primaire de leurs appareils en outre la communication par voix ou par messages grâce aux réseaux de téléphonie standard comme GSM, AMPS..etc. Ils rendent leurs produits de plus en plus puissant en augmentant leur capacité mémoire et leur puissance de calcul mais aussi plus légers, les dotant de plus en plus de fonctionnalités et d'applications, notamment en profitant de la révolution technique qu'a connu internet en cette décennie, rendant possible des interactions en temps réel entre les différents utilisateurs de la toile ainsi que les services que cette dernière leur offre : réseaux sociaux (facebook, twitter ….etc), GPS, accès à différentes sources d'informations...etc

En effet, l'évolution en parallèle de ces deux domaines (téléphone et internet) a permis à chacun d'entre eux de profiter des atouts de l'autre, accès à un réseau planétaire sans contraintes physiques et en très haut débit grâce aux réseaux 3G, 4G ou les points d'accès WIFI, la mobilité et la facilité d'utilisation qu'offrent les Smartphones. Ainsi cette nouvelle forme d'interaction entre l'utilisateur et le monde virtuelle a permis l'essor et l'apparition d'applications de communication basées non pas sur les réseaux de téléphonie standard mais sur les réseaux internet sans fils. Parmi ces applications on compte le géant de Microsoft Skype, Viber, What's APP ...etc. Toutes ces applications offrent la possibilité de communiquer par voix ou messages à moindre couts, offrant ainsi une alternative plus agréable et aussi plus fluide que les communications traditionnelles via les opérateurs téléphoniques.

Dans ce mémoire, nous allons réaliser une application de communication à divergence sociale. Un client peut, soit communiquer avec un cercle de contactes par messages grâce à une messagerie instantanée, soit passer des appels par voix IP. Pour cela nous avons choisi comme système d'exploitation Android qui nous permettra de mettre en place l'interface graphique et les différents services pour interconnecter l'application aux serveurs.

Par la suite, nous allons installer et configurer trois serveurs Asterisk (qui gérera la voix IP), Openfire (pour la messagerie instantanée), ainsi que Mysql (ou nous installerons la base de données). Enfin nous allons interconnecter l'ensemble à un réseau local.

Pour mener à terme ce mémoire, nous l'avons organisé en quatre chapitres :

- Le premier chapitre sera consacré pour des généralités sur le système Android ;
- Le deuxième chapitre portera sur la téléphonie IP et la messagerie instantanée ;
- Le troisième chapitre sera consacré pour l'analyse et conception de l'application ;
- Le dernier chapitre portera la réalisation de notre application.

I- Introduction :

Dans ce chapitre nous allons présenter la technologie Android, son historique, son architecture, et les outils nécessaires au développement Android.

II. Présentation générale :

Android est un système d'exploitation édité par Google pour des appareils embarqués et/ou mobiles, comme les Smartphones ou les tablettes. On le retrouve aussi dans certains GPS, ordinateurs de bord de voitures, dans des télévisions, autoradios, et même des montres. De plus, de nombreux prototypes d'appareils électroménagers, comme des réfrigérateurs ou des machines à laver, fonctionnant sous Android ont été présentés ces derniers temps, permettant ainsi de lancer une machine à l'aide de son téléphone, ou encore d'être prévenu par son frigo lorsqu'il manque certaines choses.

Le système Android est basé sur un noyau de Linux. Ce dernier a été modifié pour être plus adapté aux terminaux mobiles ayant peu de puissance de calcul, de mémoire et de batterie. En effet, certaines librairies standards ne sont pas supportées par le système, et des améliorations ont été apportées sur la gestion de l'énergie.

Les applications sont écrites en Java, et fonctionnent au sein d'une machine virtuelle Dalvik. Cette machine virtuelle a été aussi modifiée pour être le plus adapté possible aux appareils de faible puissance. Ainsi, beaucoup d'efforts ont été fournis sur la consommation de mémoire, qui a été largement diminuée par rapport à la machine virtuelle java classique.

D'autre part, ce système est open source, ce qui permet à n'importe qui de lancer sa propre version d'Android. De nombreuses personnes utilisent ainsi des "ROM custom", c'est à dire des versions modifiées par rapport au code de base fourni par l'éditeur. Toutefois, on peut se demander si Google joue le jeu : en effet, le code source de la version 3.0 (Honeycomb), dédié aux tablettes, n'a jamais été fourni. Officiellement, c'était pour éviter que les constructeurs rajoutent une surcouche (c'est à dire un thème différent, d'autres logiciels, etc.).

De plus, certaines librairies sont propriétaires, et les développeurs ne sont pas tous égaux devant le géant de MountainView, un petit groupe est en effet privilégié et bénéficie des dernières mises à jour du SDK, tandis que les autres sont un peu oubliés.

II.1 Historique :

Android commence en octobre 2003, où la société Android Inc. est créée. Officiellement, elle développe des logiciels pour mobiles. Mais en réalité, elle se préparait à sortir un tout nouveau système d'exploitation pour Smartphones. En 2005, Google rachète cette entreprise, et sort une première béta en novembre 2007, avant de lancer la version 1.0 en septembre 2008 avec le HTC Dream. À partir de ce moment-là, le rythme des nouvelles sorties est très élevé : pas moins de 32 versions différentes sont apparues à notre jour.

La version actuelle est la 4.4, nommée " Kit Kat", et est disponible depuis 31 octobre 2013. Toutefois, peu de Smartphones ont été mis à jour, cette tâche étant déléguée aux constructeurs.

Plus de 1 300 000 applications sont disponibles sur le Market en juillet 2014 et huit téléphones sur dix acheté au niveau mondial fonctionnent sous Android. Ce système d'exploitation est, en effet, gratuit pour les constructeurs, ce qui les incite à sortir de nouveaux Smartphones : de fait, plus de 30 marques proposent des Smartphones ou des tablettes : HTC, Samsung, Sony Ericsson, etc.

Android, même s'il est majoritaire, n'est pas le seul sur le marché des Smartphones. Il doit, en effet, faire face à la concurrence d'Apple son principal adversaire avec son iPhone. De plus, Google doit aussi compter sur la concurrence de Microsoft avec Windows Mobile 8, et RIM (Blackberry), qui espère gagner quelques parts du marché. Quant à Nokia et son système Symbian, il est en nette perte de vitesse car il n'a pas beaucoup évolué ces dernières années.

II.2 Bugdroid : [1]

Figure I.1 : Bugdroid

Le personnage nommé Bugdroid est la mascotte verte utilisée par Google et toutes les communautés Android pour représenter le système d'exploitation (OS).

Ce personnage est sous licence « creative commons by (3.0)» et peut donc être utilisé librement. C'est ainsi que l'on peut voir une multitude de petits robots verts. Certains sont juste Fun, d'autres sont le symbole d'une communauté d'internautes passionnés.

Selon certains sites comme Engadget, Bugdroid serait en fait un personnage d'un jeu des années 1990 sur Atari : Gauntlet: The ThirdEncounter.

III- Caractéristiques d'Android :

Le système d'exploitation Android est développé sur un noyau Linux ce qui fait de ce système un open-source. Son interface est développée en java ce qui lui procure une grande souplesse d'interactivité. C'est un interpréteur de type JIT (Just-In-Time) qui exécute les programmes en faisant une compilation à la volée ce qui procure un gain de place énorme pour le système, toutefois il est possible de passer outre cette interface.

Certaines applications ont été développées directement en C ce qui donne beaucoup de souffle au système, libère de la mémoire et accélère les programmes, de plus le développement en C est beaucoup plus efficace et beaucoup plus rapide.

La grande hétérogénéité des coprocesseurs procurera certaines différences d'un appareil à l'autre malgré une architecture ARM (Advanced Risc Machine). Seul le système HAL (Hardware Abstraction Layer), couche d'abstraction matérielle, peut laisser une impression de léger retard pour le système sur ses concurrents car il est déjà considéré comme obsolète ! Par contre, on trouvera dans le système Android tous les utilitaires indispensables à tout téléphone portable.

Android a été conçu pour intégrer au mieux des applications existantes de Google comme le service de courrier Gmail, celui de cartographie, Google Maps, ou encore Google Agenda, Google Talk, YouTube…etc.

III.1 Architecture de la plateforme Android : [2]

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

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

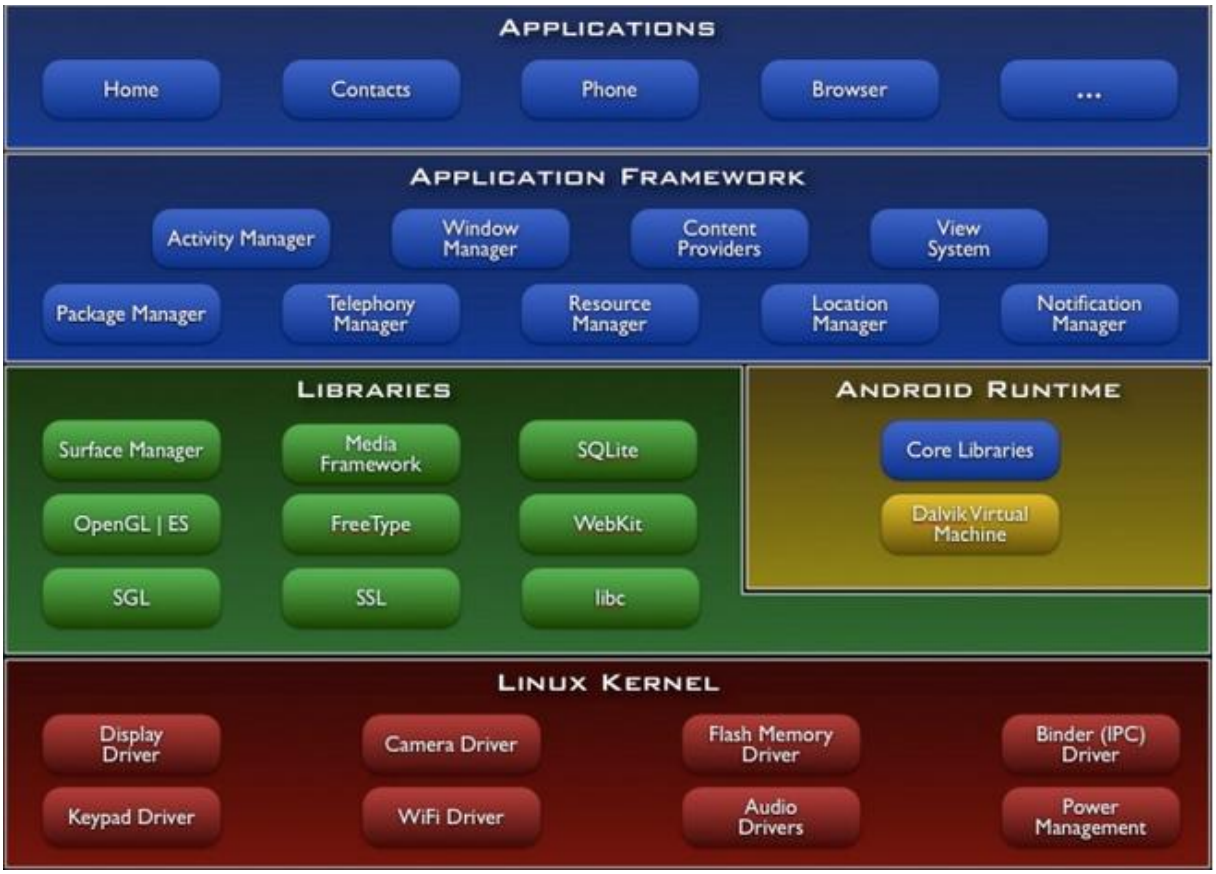


Figure I.2 : Architecture de la plateforme Android.

III.1.1 Premier niveau : noyau linux :

Android s'appuie sur le noyau Linux 2.6. Le noyau est l'élément du système d'exploitation qui permet de faire le pont entre la partie matérielle et la partie logicielle. La version du noyau utilisée avec Android est une version conçue spécialement pour l'environnement mobile, avec une gestion avancée de la batterie et une gestion particulière de la mémoire.

III.1.2 Deuxiéme niveau : librairies et environnement d'exécution :

a) Les librairies :

Ces bibliothèques proviennent de beaucoup de projets open-sources, écrits en c/c++ pour la plupart, comme SQLite pour les bases de données, WebKit pour la navigation web ou encore OpenGL, afin de produire des graphismes en 2D ou 3D.

Malgré que le développement d'applications se fasse en langage JAVA, Android comprend très bien le C et le C++.

b) L'environnement d'exécution :

Le runtime Android est basé sur le concept de machine virtuelle utilisée en java. Etant donné les limitations des dispositifs (peu de mémoire et vitesse du processeur), il n'a pas été possible d'utiliser une machine virtuelle java standard. Google a pris la décision de créer une nouvelle machine virtuelle Dalvik, afin de mieux répondre à ces limitations. Dans cette partie qui est l'environnement d'exécution, on retrouve essentiellement :

b.1) Core Libraries :

Les Corelibraries fournissent le langage java, disponible pour les applications. Le langage java fournit avec Android reprend en grande partie l'API (Application programming Interface) JSE 1.5. Il y a des choses qui ont été mises de côté, car cela n'avait pas de sens pour Android (comme les imprimantes, swing, etc…) et d'autres APIs spécifiques requises pour Android ont été rajoutées.

b.2) Dalvik :

Les applications java, développées pour Android, doivent être compilées au format Dalvik exécutable (.dex) avec l'outil (dx). Cet outil compile les (.java) en (.class) et ensuite il convertit ces (.class) en (.dex).

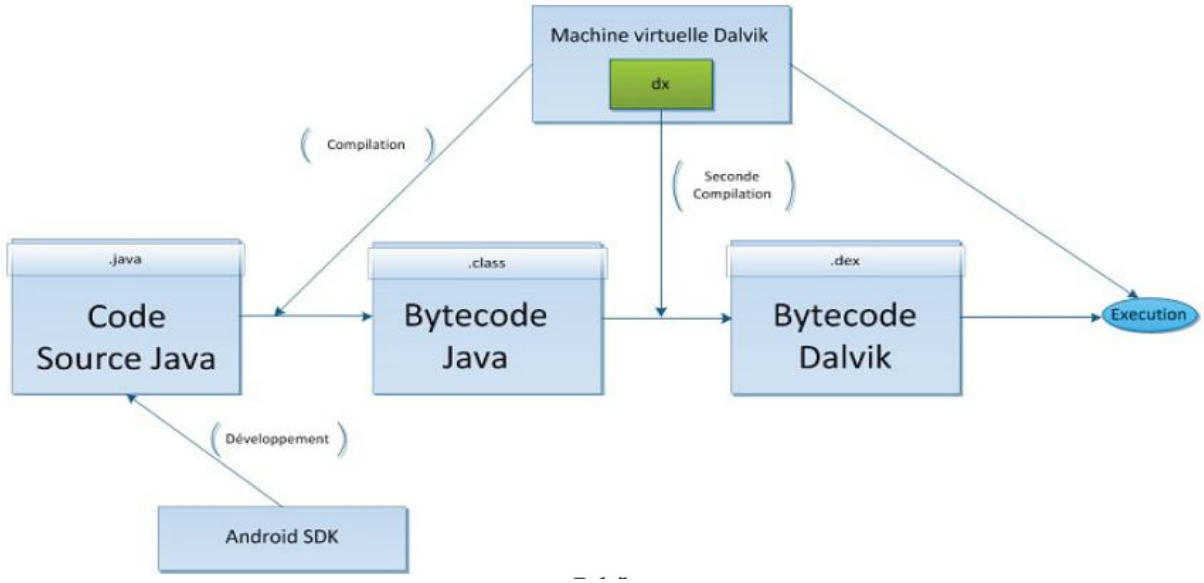


Figure I.3 : la machine virtuelle Dalvik.

III.1.3 Troisième niveau : module de développement d'applications : [3]

Le Framework est situé au-dessus de l'Android Runtime et des librairies. Il fournit des API utilisées par les applications de base permettant aux développeurs de créer des applications extrêmement riches et innovantes. L'architecture d'applications est conçue pour simplifier la réutilisation des composants ; n'importe quelle application peut publier ses capacités et n'importe quelle autre application peut alors faire usage de ces capacités. Ce même mécanisme permet aux composants de base d'être remplacés par l'utilisateur.

Toutes les applications sous-jacentes forment un ensemble de services (un service est une application, qui n'a aucune interaction avec l'utilisateur et qui tourne en arrière-plan) et de systèmes, y compris :

- Activity Manager : gère le cycle de vie des applications et maintient une pile de navigation permettant d'aller d'une application à une autre ;
- Content Provider : gère le partage de données entre applications ;
- View System : fournit tous les composants graphiques tels que : listes, grilles, boutons, etc ;
- Wifi Service : permet d'accéder à l'interface Wifi
- Telephony Service : permet d'accéder aux interfaces téléphoniques (GSM « Global Système for Mobile Communications », 3G, etc.).

III.1.4 Quatrième niveau : applications : [3]

Il s'agit tout simplement d'un ensemble d'applications que l'on peut trouver sur Android, par exemple, les fonctionnalités de base incluent un client, pour recevoir/envoyer des emails, une application pour envoyer/recevoir des SMS, un calendrier, un répertoire, etc.

III.2 SDK Android :

Approchant le fonctionnement de Flex (ActionScript/MXML) ou de .NET, Android fournit un kit de développement SDK (Software Development Kit) qui permet de développer des applications spécifiques de la téléphonie mobile à mettre en œuvre sur la plate-forme.

III.2.1 Développement : [4]

III.2.1.1 Version :

Les différentes versions d'Android ont toutes des noms de desserts « en anglais » depuis la sortie de la version 1.5 et suivent une logique alphabétique (de A vers Z) : Apple Pie (Tarte aux pommes), version connue uniquement ou presque des développeurs car c'est la Sdk distribuée avant la sortie du premier téléphone Android.

- **1.0** : Apple Pie (Tarte aux pommes), sortie en 23 septembre 2008 ;
- **1.1 :** Banana bread (Pain de la banane), sortie en 9 février 2009 ;
- **1.5** : Cupcake (Petit Gâteau), sortie en avril 2009 ;
- **1.6 :** Donut (Beignet), sortie en 15 septembre 2009 ;
- **2.0 (2.0.1) :** sortie le 26 octobre 2009, version appelée Éclair au départ mais, à cause de nombreux bugs, vite remplacée par la 2.0.1 puis par la 2.1. Cette version 2.0 est très peu connue ;
- **2.1 :** Éclair, sortie en janvier 2010 ;
- **2.3 (2.3.7) :** Gingerbread (Pain d'épice), sortie le 6 décembre 2010, support la VOIP et SIP ;
- **3.0 (3.2.6) :** Honeycomb (Rayon de miel), sortie le 2 février 2011, version pour grandes tablettes et télévisions connectées ;
- **4.0.1 (4.0.4) :** IceCream Sandwich (Sandwich à la crème glacée), version unifiée pour Smartphone, tablette et Google TV, fortement inspirée d'Honeycomb, sortie le 19 octobre 2011 ;

- **4.1 (4.3) :** Jelly Bean (dragée), sortie le 27 juin 2012, il ajoute un système de notification améliorée, la reconnaissance vocale sans connexion internet, et le « Project Butter » qui augmente la fluidité d'Android ;
- **4.4 :** Kitkat, sortie le 31 octobre 2013. Consommation en ressource moins élevée nécessitant moins de RAM, nouvelles icones plus soignées, la barre de bas et celle de statut deviennent transparentes sur certains menus et changent de couleur en fonction du contenu affiché.

Selon plusieurs documents officiels, les fichiers SDK de la version developé pre view de **Android L** et le code source Android (Android Open Source Project), la prochaine version majeure de l'OS (Opérating system) de Google pour Smartphones et tablettes connue jusqu'ici sous le nom ou le diminutif Android L, pourrait bien s'appeler **Lemon Meringue Pie**« *LMP* » (Tarte au citron meringuée). La référence **LMP** aurait en effet été découverte dans de nombreux fichiers du **SDK d'Android L** et également dans un document de la WiFi Alliance au sujet de la certification WiFi d'une certaine HTC Volantis (la Nexus 8/9). Pour rappel, la version finale d'Android L est prévue pour la fin d'année 2014.

III.2.1.2 Répartition actuelle des versions d'Android :

Répartition des différentes versions, au 12 août 2014 [4]

Version	Nom de version	API	Distribution
2.2	Froyo	8	0,7%
2.3.3-2.3.7	Gingerbread	10	13,6%
4.0.3-4.0.4	Icecream sandwich	15	1O,6%
4.1. x	Jelly Bean	16	26,5%
4.2. x		17	19,8%
4.3		18	7 ,9%
4.4	Kitkat	19	20,9

Figure : I.4 : Table illustrant la part de chaque version d'Android au 12 août 2014.

III.2.2 Quelques outils que fournit SDK pour les développeurs : [5]

Le SDK Android compte plusieurs outils et utilitaires pour nous aider à créer, tester et déboguer nos projets. On présentera brièvement quelques un de ces outils

❖ III.2.2.1 Android Virtual Device Manager :

Cet outil est utilisé pour créer et gérer les terminaux virtuels (AVD) qui hébergeront les instances de l'émulateur. Ces AVD servent à simuler les versions logicielles et les configurations matérielles disponibles sur différents appareils physiques. Les applications peuvent être ainsi testées sur plusieurs plateformes matérielles sans avoir besoin d'acheter les téléphones correspondants.

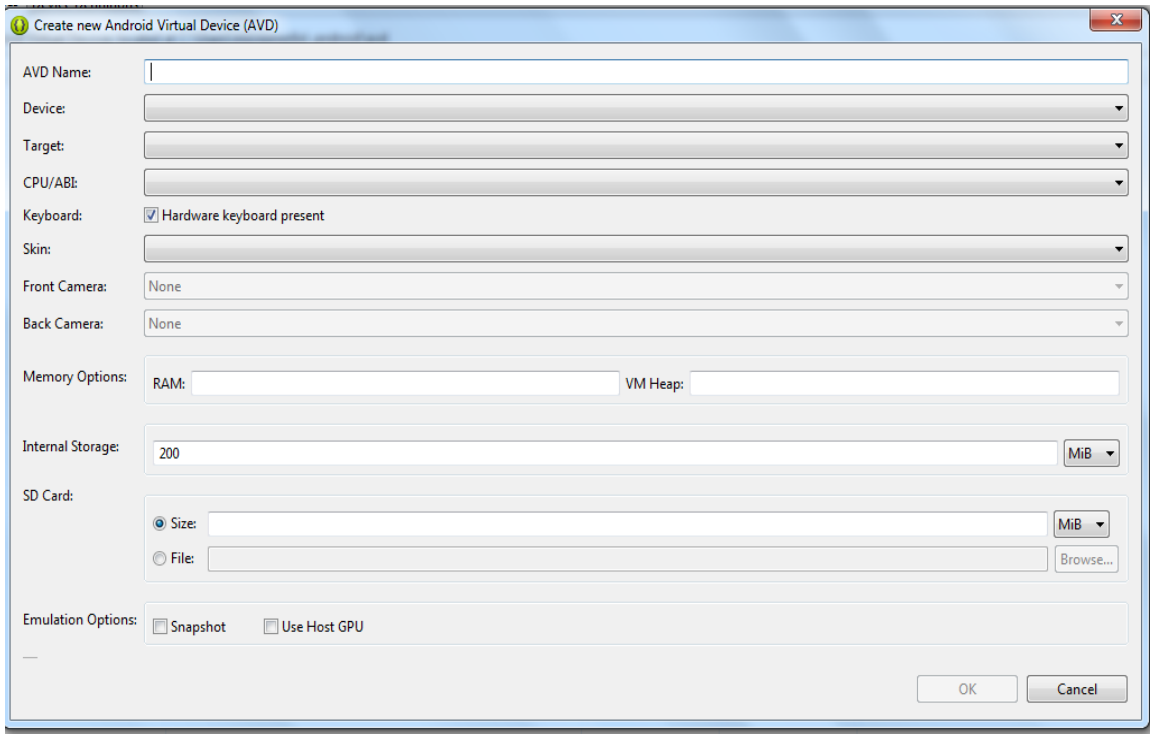


Figure I.5 : Android Virtual Device Manager.

❖ III.2.2.2 SDK Manager :

Le SDK Manager permet de connaitre la version du SDK installée et d'en installer de nouvelles lorsqu'elles sont disponibles.

Outre la version de la plateforme, il affiche les outils et un certain nombre de packages supplémentaires. Chaque version de plateforme comprend le SDK, la documentation, des outils et des exemples correspondant à la version concernée.

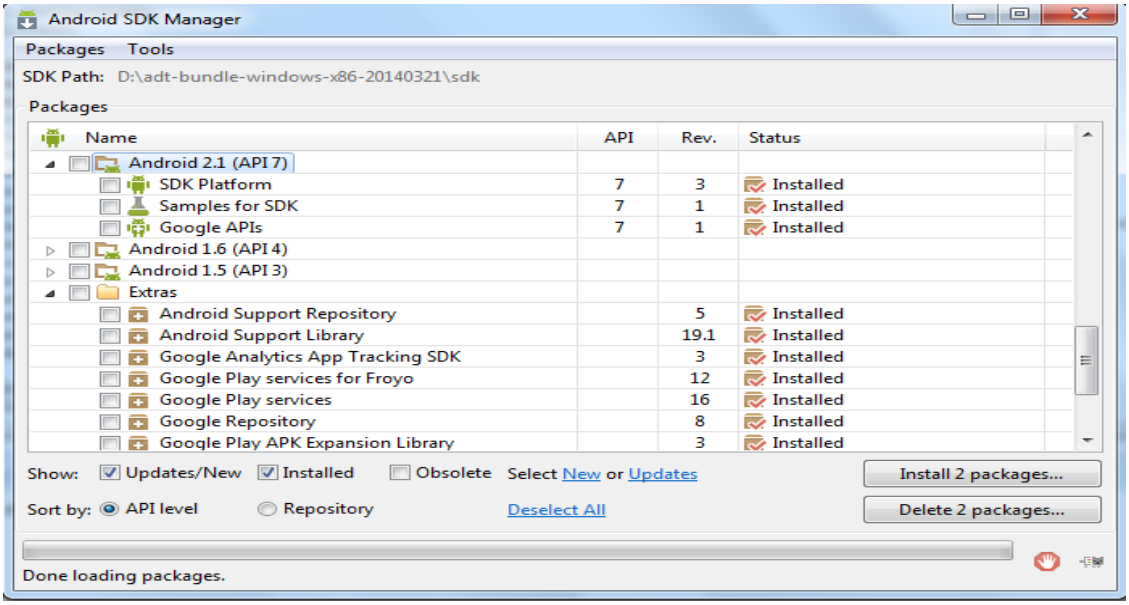


Figure I.6 : SDK Manager.

❖ III.2.2.3 L'émulateur Android :

L'émulateur est le parfait outil de test et de débogage de nos applications. C'est une implémentation de la machine virtuelle Dalvik, faisant de celle-ci une plateforme exécutant des applications Android comme le ferait n'importe quel téléphone Android.
Etant découplé de tout matériel, c'est une excellente base de test de nos applications.

Il fournit une connectivité réseau complète ainsi que la possibilité d'ajouter finement vitesse de connexion et latence au cours de débogage des applications. Nous pouvons également simuler l'envoi et la réception d'appels et de SMS.

Figure I.7 : L'émulateur Android.

❖ III.2.2.4 DDMS (Dalvik Debug Monitor Service) :

L'émulateur permet de voir à quoi ressemblera notre application et la façon dont elle se comportera et interagira, mais c'est le DDMS qui nous permettra de voir ce qui se passe en profondeur. C'est un puissant outil de débogage avec lequel nous pourrons interroger les processus actifs, examinons la pile et le tas, surveillons et mettrons en pause les threads actifs et explorer le système de fichiers de n'importe quel matériel Android connecté.

La perspective DDMS sous Eclipse fournit également un accès simplifié aux captures d'écran de l'émulateur et aux journaux générés par LogCat.

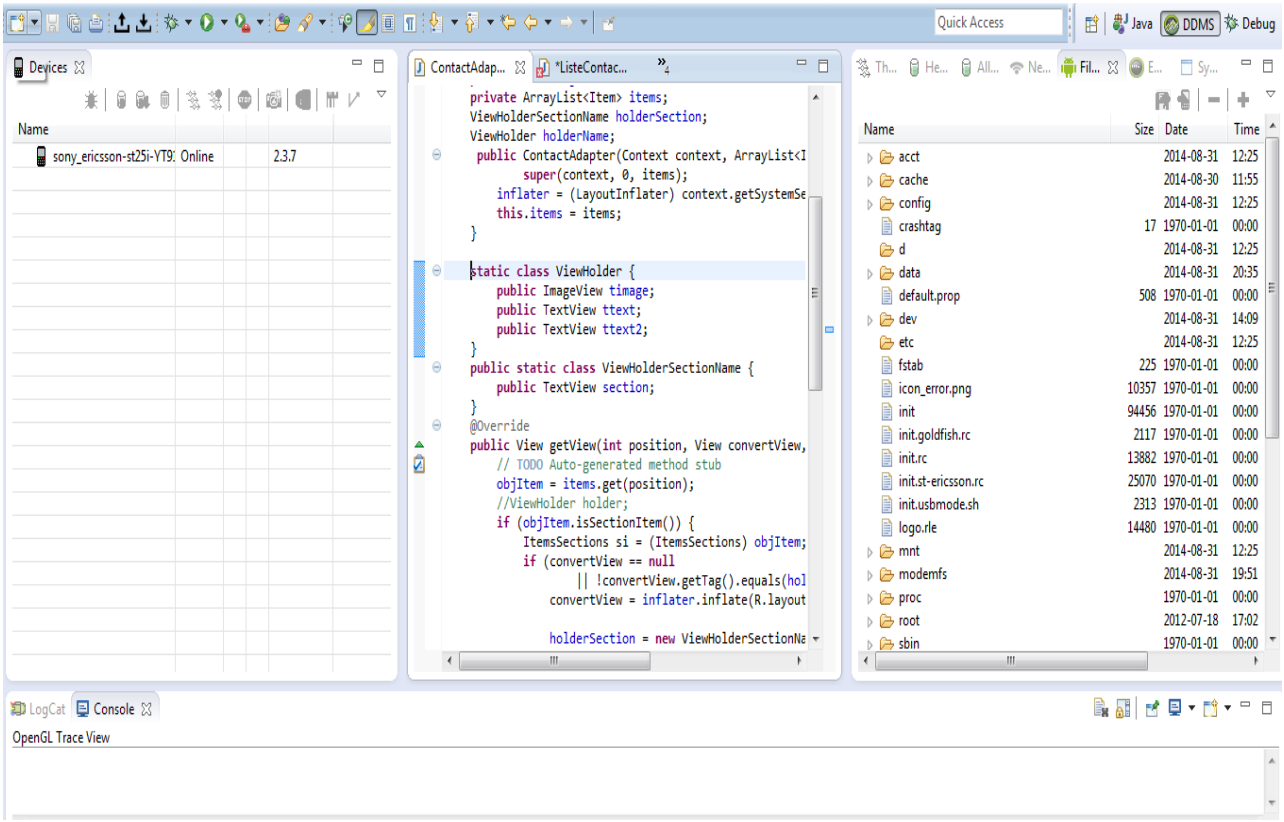


Figure I.8: Dalvik Debug Monitor Service.

❖ III.2.2.5 ADB (Android Debug Bridge) :

L'ADB est à la fois un client et un service qui nous connecte à un émulateur Android ou un appareil. Il est composé de trois parties :

- Une démo exécutée par l'émulateur ;
- Un service exécuté par notre machine de développement ;
- Des applications clientes (comme le DDMS) qui communiquent avec le démo via le service

L'ADB joue le rôle d'un conduit de communications entre la machine de développement et l'émulateur ou le dispositif Android, ADB permet d'installer des applications, de copier ou lire des fichiers et d'exécuter des commandes shell sur le dispositif cible. Grace au shell du dispositif, nous pouvons modifier les réglages des journaux et interroger ou modifier les bases de données SQL qu'il contient.

IV- Concepts et innovations :

IV.1 Bureaux virtuels (le home): [2]

Le home est l'application principale du système. Elle peut s'apparenter au bureau d'un système d'exploitation pour ordinateur. L'équipe de Google ayant développé le système a aussi développé un home sous licence open source. Le home s'étend en général sur plusieurs parties coulissantes de gauche à droite sur lesquels il est possible d'y placer des raccourcis vers les applications installées et des widgets.

L'image de fond s'étend aussi sur toutes les parties et bouge légèrement quand on change de partie, ce qui donne l'impression que le contenu fait partie du décor. Cependant, certains constructeurs comme HTC modifient ce bureau : l'interface HTC Sense est dotée de 7 bureaux, avec un menu personnalisé. Sony, LG et Samsung profitent aussi du caractère ouvert d'Android pour modifier l'interface Android originelle. Plusieurs surcouches de fabricants ont été développées comme HTC Sense, Samsung TouchWiz, LG Optimus UI ou Sony Ericsson UI.

Figure I.9 : le home.

IV.2 Les Widgets : [2]

Comme pour les bureaux des systèmes d'exploitation récents, il est possible de choisir les éléments qui se trouvent sur l'écran d'accueil. On trouve principalement des icônes, mais les utilisateurs d'Android sont aussi friands de ce qu'on appelle les Widgets,

Une Widget est une application miniature destinée à être utilisée dans une autre application, cette Widget permet d'améliorer une application à peu de frais en lui ajoutant un compagnon permanent. De plus, mettre un Widget sur son écran d'accueil permet à l'utilisateur de se rappeler l'existence de notre produit et par conséquent d'y accéder plus régulièrement. Par ailleurs, jeter un œil rapide sur une application installée et lui accordé un accès direct à certaines fonctionnalités de l'application sans avoir à l'ouvrir, ou même ouvrir l'application ou des portions de l'application.

Parmi les plus utilisées, nous avons donc : la galerie photo, le lecteur musical, l'horloge, la météo, l'agenda, la gestion de batterie etc. Et ce qui est génial, c'est de pouvoir les choisir, les modifier, les placer sur la page de notre choix.

IV.3 Centralisation des notifications : [2]

Figure I.10 : Barre de notification

La barre du haut reçoit les statuts (réseaux utilisés, niveau de batterie, modes vibreur/sonnerie/silencieux, alarme, etc.) et les notifications. Celles-ci proviennent des applications, et peuvent avertir de l'arrivée d'emails, de SMS, d'appels en absence, ou bien afficher des informations relatives à Google Talk, à l'avancement des téléchargements en cours, etc.

Lorsque l'utilisateur glisse son doigt sur cette barre afin de la faire coulisser vers le bas, elle affiche les détails des notifications et permet, lors d'une pression sur ces notifications, l'ouverture de l'application concernée.

IV.4 Capteurs divers : [6]

Android prend en charge des capteurs de type gyroscope, gravité, accélération linéaire ou encore baromètre. Ces capteurs peuvent ouvrir une autre dimension au système Android, pouvant le spécialiser dans des domaines de recherche, ou de sport.

Depuis la version 2.3, Android prend en charge la technologie NFC (Near Field Communication), qui ressemble au RFID (Radio Frequency Identification), pour tout ce qui peut être paiement, ticket de métro, et d'autres applications commerciales. Parmi les appareils équipés d'une puce NFC, on peut citer le Nexus S ou le GalaxyNexus.

IV.5 Fils de discussion : [6]

Les fils de discussion sont de plus en plus utilisés sur les différents modes de communication. Sur Android, ils sont utilisés pour les courriels (comme sur Gmail) et les SMS (comme sur les Sony Ericsson).

IV.6 Android Market/Google Play Store: [7]

Android Market a été remplacée par Google Play Store le 6 mars 2012. Cette nouvelle plate-forme rassemble des services Google déjà existants.

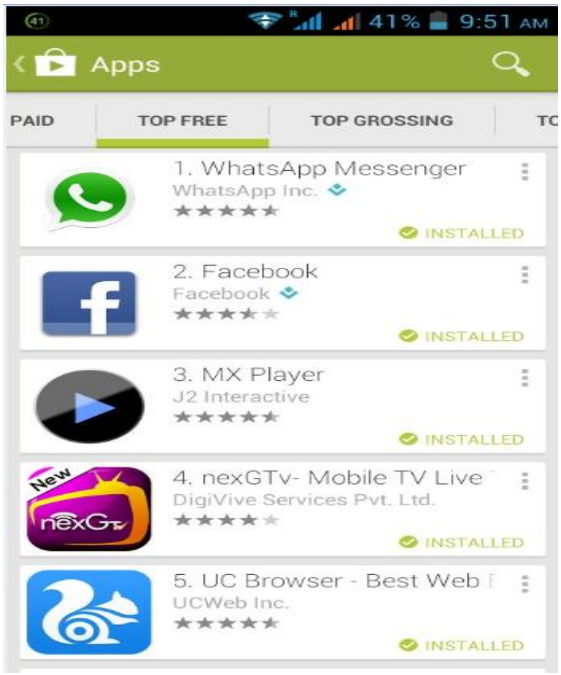


Figure I.11: Google Play Store.

Celui-ci permet de télécharger des logiciels, des livres, des films ou de la musique. Il est aussi possible de les noter et de les commenter. En juillet 2014, il y avait plus de 1.300 .000 applications sur Google Play Store, dont plus d'un million sont gratuites.

L'Android market est aussi le système de publication d'applications officielles d'Android. Néanmoins, contrairement à l'App Store d'Apple, la distribution d'applications au public n'est pas un monopole de Google et techniquement, rien n'empêche d'installer des applications depuis une autre source. D'ailleurs, des alternatives à Android Market, comme par exemple SlideMe [14], émergent peu à peu.

Les Smartphones disposant d'une puce Tegra, comme Nexus 7 avec une puce Tegra 3, ont accès à la Tegra Zone [15], un magasin d'applications permettant de télécharger des applications dédiées aux téléphones sous Tegra. Il contient notamment des applications THD (Très Haute Définition).

Pour proposer son application sur Android Market (Play Store) [16], il faudra détenir un compte Google et s'acquitter des frais d'enregistrement qui s'élèvent à 25 dollars.

Bien sûr, l'application devra être parfaitement packagée et signée à l'aide d'un certificat P12, Le système ne peut pas installer une application qui n'est pas signée. Par ailleurs, Android Market requiert que la date de validité du certificat soit postérieure au 22 octobre 2033, qu'une icône et un intitulé soient précisés dans le fichier « AndroidManifest.xml »de l'application (dans l'attribut android : icon et android : label) et que les champs de version (android : versionCode et android : versionName) soient renseignés.

V- Conclusion :

Dans ce chapitre, nous avons fait une étude sur Android tout en présentant un bref historique, l'architecture de la plateforme Android et les éléments du SDK d'Android, afin de voir les fonctionnalités que nous pouvons trouver sur ce système d'exploitation.

Aujourd'hui, un développeur souhaitant se lancer dans la création d'applications pour mobiles a le choix entre trois plateformes : Android, iOS (iPhone), et Windows Mobile. Développer sous Android concède beaucoup d'avantages par rapport à ses concurrents. Comme nous avons pu le voir, il est gratuit, modulaire, open source, et ne nécessite pas d'apprendre un nouveau langage si l'on connaît déjà le Java (contrairement à l'iPhone et son ObjectiveC).

I- Introduction :

Pour la réalisation d'un système de messagerie instantanée ou de téléphonie IP nous avons besoin d'outils spécifiques plus exactement des serveurs dédiés à ces tâches.

Nous citerons pour la téléphonie IP, les quelques PABX (Private Branch Exchange) open source :

- Asterisk.
- SipXecs.
- allWeaver.
- FreeSwitch.
- GNU Bayonne.
- YATE.

Pour la messagerie instantanée nous citerons les quelques services open source :

- 12 Planet/Instant Messaging Server Standard.
- Ecrio /IMPS Server.
- IBM/Lotus Sametime 7 Server.
- Novell /Groupwise Messenger.
- Sun /Sun Java System Instant Messaging.
- Jabber/OpenFire.

Dans ce chapitre nous présenterons les deux serveurs Asterisk (PABX) et Openfire (serveur de messagerie), leurs descriptions ainsi que leurs fonctionnalités, et les protocoles qu'ils implémentent.

II – La téléphonie IP/Voix sur IP:

On utilise le terme de voix sur IP pour désigner une communication vocale et/ou vidéo d'un point fixe à un autre sur le réseau internet, par l'intermédiaire de routeurs. La VoIP (acronyme de l'anglais Voice over Internet Protocol) est un protocole de communication permettant de transférer la voix sur internet par des réseaux à large bande passante. La voix est alors décomposée en paquets qui circulent sur la toile mondiale. Ce qu'on appelle aussi la téléphonie par internet se distingue des réseaux filaires classiques par les lignes de cuivre des opérateurs.

Depuis les années 80 la VoIP cherche à trouver son marché, mais c'est l'arrivée de larges bandes et donc l'amélioration de la vitesse et de la qualité des transmissions internet qui a permis l'explosion de la voix sur IP.

II -1 Asterisk et la téléphonie IP :

Asterisk est un autocommutateur téléphonique privé (PABX) open source pour les Systèmes d'exploitation UNIX, il est publié sous licence GPL et licence propriétaire. Asterisk permet aux ingénieurs réseau et aux développeurs de s'approprier la téléphonie et de construire des services et des applications innovantes et utiles. Il comprend plusieurs fonctionnalités telles que, les appels téléphoniques la messagerie vocale, la messagerie instantanée, la musique d'attente, les files d'attentes.

Asterisk implémente les protocoles H.320, H.323 et SIP, ainsi qu'un protocole spécifique nommé IAX (Inter-Asterisk eXchange). Ce protocole IAX permet la communication entre deux serveurs Asterisk ainsi qu'entre un client et un serveur Asterisk. Il peut également jouer le rôle de *registrar* « bureau d'enregistrement » (un serveur qui accepte les demandes de registre et met l'information qu'il reçoit de ces demandes dans le service de localisation pour le domaine qu'il gère) et passerelle avec les réseaux publics (RTC, GSM, etc.).

Figure II.1 : Logo d'Asterisk

II-2 Présentation des fonctionnalités d'ASTERISK : [8]

Asterisk a pour principale fonction celle d'autocommutateur téléphonique. Cela consiste essentiellement à gérer les appels téléphoniques pour un ensemble de postes, tout comme pouvait le réaliser une opératrice. Cependant, et contrairement à certaines solutions de ToIP (Telephony over Internet Protocol) propriétaires qui se contentent de reproduire ce schéma classique de la téléphonie d'entreprise, Asterisk concrétise les promesses de la technologie ToIP, en proposant de nouveaux services sans casser sa tirelire.

II-2-1 Serveur d'audioconférences :

La gestion des audioconférences est une fonction majeure d'Asterisk, qui par l'intermédiaire de l'application MeetMe, agit comme un véritable pont de conférence. Le nombre d'utilisateurs par session est illimité sous réserve d'acquérir la licence nécessaire, comme on peut l'attendre de tout logiciel libre, aucune licence ne restreint l'utilisation du service d'audioconférence ainsi déployé, les seules limites d'ordre quantitatif découlant des capacités du matériel.

II-2-2 Messagerie vocale et instantanée :

La messagerie vocale est le complément indispensable à tout système de téléphonie digne de ce nom. Là encore, Asterisk brille par sa souplesse ainsi que par la richesse de ses fonctionnalités. Il est ainsi tout à fait à même de réaliser ce que certains éditeurs ou constructeurs de système de téléphonie appellent parfois la messagerie unifiée. L'envoi de message électronique (protocole SMTP) dans une boîte e-mail la réception d'un message téléphonique (vocale) et naturellement possible, de même que l'envoi d'un message instantané par Jabber (protocole XMPP). [8]

Voici quelques exemples de configuration d'un service de messagerie vocale sous Asterisk :

- Asterisk en tant qu'IPBX incluant la messagerie vocale ;
- Asterisk en tant que serveur de messagerie vocale connecté à un autocommutateur existant, par exemple grâce à SIP ;
- Notification sur une boîte e-mail ou par la messagerie instantanée lors de la réception d'un message téléphonique.

Il n'est pas rare de voir un serveur Asterisk assigné à la seule tâche de la messagerie vocale. En séparant les fonctions sur différents serveurs, on améliore parfois la disponibilité des éléments qui constituent le service téléphonique fourni aux utilisateurs.

II-2-3 Serveur vocal interactif (SVI) :

Un SVI (en anglais IVR, pour *Interactive Voice Response*) est une application téléphonique d'aide à la navigation. L'utilisateur appelant le système est guidé par une voix parmi différents menus qu'il explore ou sélectionne en appuyant sur des touches de son terminal (par exemple un poste téléphonique), chacune émettant alors un code sonore (DTMF) qui sera interprété par le SVI comme un chiffre.

Asterisk dispose des éléments constitutifs d'un SVI évolué : [8]

- Un mécanisme de routage interne des appels reposant sur des tests conditionnels ;
- La possibilité de lire des fichiers préenregistrés dans différents langages ;
- Une application permettant d'enregistrer ses propres fichiers depuis un téléphone.

Le routage des appels est configurable via le *dialplan*, l'équivalent du plan de numérotation d'un PABX classique. Il décrit les combinaisons que les utilisateurs peuvent composer et ce à quoi mène chaque série de touches frappées. Les fichiers préenregistrés sont inclus dans les distributions d'Asterisk et disponibles en plusieurs langues. Ils contiennent les menus vocaux que l'utilisateur entendra au cours d'un appel au SVI. Asterisk permet également d'enregistrer des fichiers vocaux qui pourront servir à établir un menu vocal personnalisé, par le biais de l'application Record.

II-3 Protocoles pour la VoIP sur Asterisk : [2]

Le mécanisme pour la réalisation d'une connexion VoIP implique généralement une série de signalisations entre les terminaux (et des passerelles entre les deux), culminant en deux flux de médias persistant (un pour chaque sens) qui portent la conversation réelle. Il existe plusieurs protocoles pour gérer cela. Dans cette section, nous allons discuter de certains ceux qui sont importants pour la VoIP en général et Astérisque en particulier.

II-3-1 IAX (The "Inter-Asterisk eXchange" Protocol): [9]

Le protocole a été développé par IAX Digium dans le but de communiquer avec d'autres serveurs Asterisk (d'où le protocole d'échange inter-Asterisk). Il est très important à noter qu'IAX n'est pas du tout limité à Asterisk. La norme est ouverte à tous. IAX est soutenu par de nombreux autres projets open source de télécommunication, ainsi que par plusieurs fournisseurs de matériel.

IAX est un protocole de transport (un peu comme SIP) qui utilise un seul Port UDP (4569) à la fois pour la signalisation de la voix et des flux multimédias. Ce qui le rend plus facile à gérer derrière un firewall NATées.

IAX a également la capacité de conduire plusieurs sessions dans un flux de données, cela peut être un avantage énorme dans la bande passante lors de l'envoi d'un grand nombre de canaux simultanés vers une machine distante. Trunking permet plusieurs flux de médias d'être représentés avec une tête de datagramme unique, ainsi il permet de réduire la surcharge associée aux canaux personnels. Cela permet de réduire la latence, la puissance de traitement et la bande passante requise. Le protocole permet de mettre à échelle un grand nombre de canaux actifs entre les extrémités beaucoup plus facilement.

II-3-2 SIP : [9]

La Session Initiation Protocol (SIP) a pris l'industrie des télécommunications par assaut. SIP a à peu près détrôné le H.323. La prémisse de SIP est que chaque extrémité d'une connexion est un pair; le protocole négocie les capacités entre eux. Ce qui rend SIP convaincant, c'est qu'il est un protocole relativement simple, avec une syntaxe similaire à d'autres protocoles connus tels que HTTP et SMTP.

SIP a été soumis à l'Internet Engineering Task Force (IETF) en Février 1996 en tant que «. draft-ietf-MMUSIC-sip-00 » Le projet initial ne ressemblait en rien au SIP actuel et ne contenait qu'un seul type de demande: une demande d'établissement d'appel.

En Mars 1999, après 11 révisions, SIP RFC 2543 est née. Dans un premier temps, SIP a été passé sous silence, comme H.323 est considéré comme le protocole de choix pour la négociation de transport VoIP. Cependant, comme le buzz grandit, SIP a commencé à gagner en popularité, il peut y avoir eu beaucoup de différents facteurs qui ont accéléré sa

croissance, nous aimerions à penser qu'une grande partie de son succès est due à sa libre disposition.

SIP est un protocole de signalisation de couche application qui utilise le port 5060 bien connu pour les communications. SIP peut être transporté soit avec le protocole UDP ou TCP de la couche transport. SIP est utilisé pour établir, modifier et terminer les sessions multimédias telles que les appels de téléphonie sur Internet. Le protocole SIP n'est utilisé que pour configurer et discriminer des appels vocaux et vidéo. Toutes les communications vocales et vidéo utilisent le protocole RTP (Realtime Transport Protocol) [10].

Le protocole RTP ne fait que transporter des données vocales/vidéo sur le réseau. L'établissement et la discrimination des appels sont généralement assurés par le protocole SIP. Le protocole RTP ne requiert pas de port TCP ou UDP standard ou statique avec lequel communiquer. Les communications RTP s'effectuent sur un numéro de port UDP pair, et le numéro impair suivant ce port est utilisé pour les communications TCP. Bien qu'il n'existe pas d'affectations de plage de ports standards, le protocole RTP est généralement configuré pour utiliser les ports 1024 à 65535, Il est difficile pour le protocole RTP de traverser des pare-feu, car il utilise une plage de ports dynamiques [10], ports non privilégiés dans Asterisk (10000 à 20000, par défaut).

II-3-3 H.323 : [9]

Ce protocole Union Internationale des Télécommunications (UIT) a été conçu à l'origine pour fournir un mécanisme de transport IP de la visioconférence. Il est devenu la norme en IP basée sur l'équipement de vidéoconférence, et il jouissait brièvement d'une grande renommée en tant que protocole VOIP par excellence. Ainsi, bien qu'il y ait beaucoup de débats sur ce qui est de SIP ou H.323 (ou IAX) va dominer le monde des protocoles VoIP, Asterisk, H.323 a été largement obsolète en faveur de IAX et SIP.

H.323 n'a pas connu beaucoup de succès auprès des utilisateurs et d'entreprises, même s'il est encore le protocole le plus largement utilisé pour la VoIP entre les transporteurs. Les trois versions de H.323 prises en charge dans Asterisk sont traitées par les modules chan_h323.so (fourni avec Asterisk), chan_oh323.so (disponible en tant que add-on gratuit), et chan_ooh323.so (fourni dans asterisk-addons).

H.323 a été développé par l'UIT en mai 1996 comme un moyen pour transmettre la voix, la vidéo, données et des communications par fax à travers un réseau IP tout en maintenant la connectivité avec le réseau RTC. Depuis ce temps, H.323 a connu plusieurs versions et annexes (qui ajoutent des fonctionnalités au protocole), ce qui lui permet de fonctionner sur les réseaux purs de VoIP et les réseaux plus largement distribués.

II-3-4 MGCP : [9]

Le Media Gateway Control Protocol (MGCP) vient de l'IETF. Bien que le déploiement de MGCP soit plus répandu que l'on pourrait penser, il est en train de perdre rapidement du terrain devant des protocoles tels que SIP et IAX. Pourtant, Asterisk a un support rudimentaire pour ce protocole. MGCP est défini dans la RFC 3435 [30]. Il a été conçu pour les terminaux (comme numéros de téléphone) aussi simples que possible. Contrairement à SIP, MGCP utilise un modèle centralisé. MGCP téléphones ne peuvent pas appeler directement d'autres téléphones MGCP; ils doivent toujours passer par un certain type de contrôleur. Asterisk permet MGCP par le module de chan_mgcp.so, et les critères sont définis dans le fichier de configuration mgcp.conf. Depuis Asterisk fournit un seul agent d'appel de base, il ne peut pas émuler un téléphone MGCP (pour vous inscrire à un autre contrôleur MGCP à titre d'agent d'utilisateur, par exemple). Si vous avez des téléphones MGCP qui traînent, vous serez en mesure de les utiliser avec Asterisk. Si vous prévoyez de mettre les téléphones MGCP en production sur un système Asterisk, garder à l'esprit que la communauté a déménagé sur les protocoles les plus populaires.

II-4 Les avantages d'Asterisk :

1- Aucune limite de nombres de postes, on peut installer autant de postes que l'on souhaite, la seul limite et liée à la capacité de la machine hardware.
2- Un réseau unique plus facile à gérer et à étendre.
3- Indépendance vis-à-vis du fournisseur.
4- La téléphonie peut fonctionner directement par adresse IP (Locale).

II-5 Les différentes versions d'Asterisk : [11]

Ce tableau illustre les différentes versions d'Asterisk :

Series	Type de sortie	Date de sortie	Security Fix Only	EOL (end of licence)
1.2.X		2005-11-21	2007-08-07	2010-11-21
1.4.X	LTS (long term support)	2006-12-23	2011-04-21	2012-04-21
1.6.0.X	Standard	2008-10-01	2010-05-01	2010-10-01
1.6.1.x	Standard	2009-04-27	2010-05-01	2011-04-27
1.6.2.X	Standard	2009-12-18	2011-04-21	2012-04-21
1.8.x	LTS (long term support)	2010-10-21	21/10/2014	21/10/2015
10.X	Standard	2011-12-15	15/12/2012	15/12/2013
11.x	LTS (long term support)	2012-10-25	25/10/2016	25/10/2017
12.x	Standard	20/12/2013	20/12/2014	20/12/2015
13.x	LTS (long term support)	2014-10 (provisoire)	2018-10 (provisoire)	2019-10 (provisoire)
14.x	Standard	2015-10 (provisoire)	2016-10 (provisoire)	2017-10 (provisoire)
15.x	LTS (long term support)	2016-10 (provisoire)	2020-10 (provisoire)	2021-10 (provisoire)

Figure II.2 : Les différentes versions d'Asterisk.

III – La messagerie instantanée :

La messagerie instantanée (ou IM) est aujourd'hui largement utilisée comme un moyen rapide et léger de communication, aussi bien professionnelle que personnelle. La messagerie instantanée est, par définition, immédiate les messages arrivent à destination pratiquement juste après leur envois, ce qui lui donne un avantage sur l'email quand il s'agit de notification rapide. Elle est également «push» plutôt que «pull». Lorsque vous recevez un message, il apparaîtra sur votre écran et attirera votre attention.

III-1 Fonctionnement de la messagerie instantanée :

La messagerie instantanée est un moyen de communiquer en privé avec d'autres personnes de son choix. Le client se connecte à un serveur qui contient les informations sur tous les utilisateurs inscrits, connectés ou non. Chaque personne possède un pseudonyme qui n'est pas forcément unique, et un identifiant unique dans la base de données du serveur. Un mot de passe est aussi indispensable pour connecter au service. Deux personnes peuvent communiquer en direct si elles sont simultanément connectées au serveur.

III-2 Openfire et la messagerie instantanée : [12]

Openfire est un serveur Jabber sous licence libre (GPL) utilisé pour faire de la messagerie instantanée. Connu sous l'appellation de JiveMessenger, puis de Wildfire, il est écrit en java par la Jive Software, l'éditeur du client libre Spark. Bien qu'il ne soit pas très connu, Openfire bénéficie du soutien d'une importante communauté. Il est stable et est réputé pour sa facilité d'installation et d'administration.

Openfire a beaucoup de fonctionnalités, conçues pour rationaliser la communication. Unes des caractéristiques d'Openfire, sa conception sécurisée, qui est due à son protocole Jabber. Jabber utilise Transport Layer Security (TLS) par défaut, et va établir une connexion sécurisée si elle est disponible. Openfire peut être configuré pour autoriser uniquement les connexions sécurisées.

Figure II.3 : Logo d'Openfire.

III-3 Présentation des fonctionnalités d'Openfire:

Alors que la version de base du serveur est disponible gratuitement, une version entreprise payante est aussi disponible, elle a des caractéristiques appropriées pour une grande entreprise multi-offices. Openfire est à double distributions sous licence GPL avec une extension commerciale pour les grandes entreprises. La version gratuite GPL n'est pas obsolète et a une variété de caractéristiques dont a besoin un serveur de messagerie instantanée complet, telles que la centralisation de l'administration de la liste d'utilisateurs, la possibilité de diffuser des messages à des groupes entiers, et la présence personnalisable indiquée (disponible, hors ligne, occupé…etc.), le tout avec de riches fonctionnalités sécurisées. Openfire est conçu pour compléter d'autres services existants en cours d'exécution sur le réseau. Ainsi, par exemple, il peut se brancher sur un serveur d'annuaire pour l'authentification des utilisateurs, ou dans une installation d'Asterisk pour la téléphonie IP.

III.3.1 Interface d'administration web :

Openfire offre une administration Web front-end (interface web), facile à utiliser Indépendamment des réglages du navigateur, l'interface vous offre beaucoup d'informations et propose différentes options à choisir. Par exemple, changer la façon dont Openfire traite les messages hors ligne (message reçu alors que l'utilisateur n'est pas connecter), le serveur offre diverses possibilités de stockage, de renvoi, ou d'ignorer les messages. L'interface permet également un accès plus rapide à des parties pertinentes de la configuration. Par exemple, dans la section « Information Server», avec une liste des ports utilisés, Openfire a un lien vers la section «Paramètres de sécurité» d'où on peut modifier les paramètres de sécurité des ports.

III.3.2 Groupe d'utilisateurs :

Openfire offre la possibilité de créer et de gérer des groupes d'utilisateurs personnalisés (amis, famille, collègues, ….etc.) facilement. Ces groupes d'utilisateurs peuvent être partagés facilement aux nouveaux utilisateurs des listes de contacts appropriés.

III.3.3 Salon de tchat :

Openfire permet la création de salon de tchat (chat room) et la possibilité de les gérer. Il offre plusieurs options comme le contrôle de la modération du salon, l'occupation maximale, informations de présence (disponible, occupé, non connecté),..etc. La page « salon de chat» permet de visualiser et d'éditer les salles de tchats actuelles et d'en créer de nouvelles.

III.3.4 Création d'une passerelle :

Sous Openfire on peut mettre en place une passerelle vers d'autres réseaux publics. Les paramètres de la passerelle permettent de filtrer, avec des critères de sécurité adéquate, les demandes de chaque client afin que seuls les clients autorisés soient admis sur le réseau.

III.4 Protocole de la messagerie instantanée sous Openfire :

Openfire, implémente entièrement le protocole Jabber /XMPP pour la messagerie instantanée. Dans ce qui suit nous verrons ce qu'est Jabber /XMPP et son historique.

III.4.1 Le protocole Jabber/Xmpp : [13]

XMPP (Extensible Messaging and Presence Protocol) est un protocole basé sur XML (Extensible Markup Language) et destiné à la messagerie instantanée (IM) ainsi que la détection de présence en ligne, qui a émergé de la communauté Jabber open-source en 1999. Il fonctionne entre ou parmi les serveurs (il peut être le cœur même du serveur ou juste un lien externe de communication avec d'autres serveurs), et il fonctionne en temps quasi-réel. Le protocole peut éventuellement permettre aux internautes d'envoyer des messages instantanés à n'importe qui d'autre sur Internet, indépendamment des différences dans les systèmes d'exploitation et navigateurs.

XMPP est prévu pour supporter des applications de messagerie instantanée avec authentification, contrôle d'accès, une haute sécurité pour la vie privée, le cryptage hop-by-hop, le cryptage de bout en bout, et la compatibilité avec d'autres protocoles.

XMPP est parfois appelé le protocole Jabber, mais c'est un abus de langage technique. Lorsque la communauté Jabber a soumis le protocole à l'IETF en 2002, elle a choisi d'utiliser le nom "XMPP" au lieu de "Jabber" pour éliminer la possibilité de confusion avec la communauté open-source et la société Jabber.com commerciale (acquis depuis par Cisco Systems, Inc.).

En Janvier 1999, Jeremie Miller a annoncé l'existence de Jabber, une technologie ouverte pour la messagerie instantanée et la gestion de la présence. Tout au long de 1999, le développement s'est déroulé rapidement sur un serveur open-source (jabberd), plusieurs clients open-source et des bibliothèques de code, et des protocoles de fils ouverts pour le streaming XML en temps réel ainsi que des extensions instantanés de base de messagerie et

de présence. Ces mêmes protocoles de base ont été utilisés (avec diverses améliorations et extensions) depuis 1999.

En Août 1999, Jérémie Miller a présenté une déclaration promettant le soutien de la communauté Jabber pour le processus de normalisation à l'IETF. Cette déclaration est conforme à la mission fondatrice du projet Jabber: favoriser la liberté de la conversation et de soutien des standards ouverts et l'interopérabilité dans les communications en temps réel.

En Février 2000, l'IETF a publié les résultats des travaux du groupe IMPP WG pour la messagerie instantanée et le Protocole (IMPP), qui est un modèle pour la messagerie instantanée et les systèmes de présence (RFC 2778 [19]) et un ensemble d'exigences pour de tels systèmes (RFC 2779 [20]). Toutefois, en raison de l'absence de consensus, l'IMPP WG n'a jamais produit de protocoles. En mai 2000, la version 1.0 du serveur jabberd a été libérée et les protocoles Jabber de base (streaming XML, la messagerie, la présence, les listes de contacts, etc) ont été stabilisés. En Juin 2000, Jérémie et les autres membres du projet Jabber présentent un projet de l'Internet (texte | HTML) pour la messagerie instantanée avec le protocole (IMPP) du Groupe de travail IMPP WG. Malheureusement, la communauté Jabber n'était pas bien organisée à ce moment-là et n'a pas été fortement axée sur l'élaboration du protocole en soi. Ainsi, le projet de l'Internet initial a expiré. En Octobre 2000, jabberd 1.2 a été libéré et le protocole de la fonction de rappel du serveur a été introduit pour éviter l'usurpation d'adresse sur le réseau croissant des serveurs Jabber.

En Août 2001, la Jabber Software Foundation (JSF) a été créée pour assurer la coordination entre le nombre croissant de projets open-source et les entités commerciales utilisant les technologies Jabber. Le mandat principal de la fondation est de gérer les protocoles XML utilisés dans la communauté Jabber / XMPP en documentant les protocoles existants et le développement d'extensions du protocole à travers un processus de normalisation ouvert.

À la fin de 2001 et début 2002, des membres éminents de la communauté Jabber ont décidé de soumettre à nouveau les protocoles Jabber de base à l'IETF, cette fois comme une soumission officielle par la JSF. La première présentation a été faite en Février 2002 en tant que projet de l'Internet (texte | HTML). Après le succès de cette présentation, il a été décidé

d'étudier la possibilité de former un groupe de travail IETF consacré à la formalisation des protocoles Jabber de base, sous le nom neutre d'eXtensible Messaging and Presence Protocol (XMPP). En conséquence, trois interdépendants Internet Drafts ont été présentés. Le 15-07-2002, à la 54e réunion de l'IETF à Yokohama, au Japon une session BOF (Birds of a Feather) à eu lieu. Compte tenu de la réaction globalement favorable, il a été décidé d'aller plus loin en soumettant un groupe de travail a charte proposée à l'IESG. Le 31-10-2002, la formation du Groupe de travail XMPP a été approuvée par le groupe de pilotage de l'ingénierie Internet (IESG). En conséquence, la Fondation a contribué formellement au processus de normalisation de l'Internet (RFC 2026 [21]) des protocoles Jabber et le contrôle du changement cédé au cours de ces protocoles à l'IETF sous le nom d'eXtensible Messaging and Presence Protocol (XMPP). Le 03-11- 2002, les mises à jour d'Internet Draft ont été soumises au core XMPP et à «XMPP IM». Le 19-11-2002, la première réunion du groupe de travail XMPP a eu lieu à l'IETF 55, y compris les représentant Jeremie Miller, Joe Hildebrand et Peter Saint-Andre.

En 2003, le XMPP WG a essentiellement achevé ses travaux sur la formalisation des protocoles Jabber de base. L'IETF a approuvé les protocoles de messagerie instantanée et de présence qui répondent pleinement aux exigences de la RFC 2779 [20]. Des réunions du groupe de travail ont eu lieu le 17-03- 2003 à l'IETF 56 et le 18-07- 2003 à l'IETF 57. Les révisions apportées aux protocoles de base principalement centrées sur l'amélioration de la sécurité et de l'internationalisation.

L'IESG a approuvé XMPP et XMPP IM sur les propositions de normes standards proposées respectivement le 29-01-2004 et 05-02-2004, acceptant ainsi XMPP 1.0 comme une messagerie instantanée IETF en conformité avec les exigences de la RFC 2779 [20]. La spécification de cartographie XMPP CPIM a été approuvée le 19-05-2004. La spécification XMPP-CPIM a été approuvée le 26-07-2004. En Octobre 2004, ces documents ont été publiés dans les RFC ci-dessous:

- RFC 3920 [22]: eXtensible Messaging and Presence Protocol (XMPP): core.
- RFC 3921 [23]: eXtensible Messaging and Presence Protocol (XMPP): Messagerie instantanée et présence.

- RFC 3922 [24]: Cartographie du Protocole Extensible Messaging and Presence (XMPP) de présence commune et la messagerie instantanée (CPIM).

- RFC 3923 [25]: End-to-End signature et de cryptage pour objet Extensible Messaging and Presence Protocol (XMPP).

Lors de la publication des XMPP RFCs [26], l'IETF a annoncé la rupture du Groupe de travail XMPP. Cependant, le développement de nouvelles extensions XMPP continu.

Autre que de continuer la publication et la promotion des différentes extensions XMPP, il n'a eu aucun grand effort de normalisation en 2005. Toutefois, la mise en œuvre et le déploiement de services à grande échelle XMPP se poursuivent, mis en évidence par le lancement de Google Talk en Août 2005.

En Juillet 2006, l'IETF a publié RFC 4622 [31] définissant le régime XMPP URI. En Juillet 2006, le premier Sommet XMPP a eu lieu à Portland, Oregon, Etats-Unis. En Octobre 2006, le JSF rebaptisé «Jabber Enhancement Proposals» de la série de spécifications à «Protocoles d'extension XMPP » et les a déplacés sur le site de xmpp.org. En Octobre 2006, les travaux ont commencé sur la révision de la RFC 3920 [22] et RFC 3921 [23] afin d'intégrer l'expérience acquise par plusieurs années de mise en œuvre généralisée et le déploiement de technologies XMPP. En Décembre 2006, le JSF a conclu un partenariat avec StartCom pour offrir des certificats numériques gratuits pour les administrateurs de serveurs Jabber / XMPP par l'autorité de certification intermédiaire XMPP.

III-5 Avantages et inconvénients d'Openfire :

III-5-1 Avantages :

- Support complet de XMPP/Jabber (le protocole).
- Programmer en java, un langage très répondu.
- Délivrer sous licence GPL open source.
- Disponible sur plusieurs plateformes (Windows, Linux, Mac OS).
- Facile à installer.
- Fénéficie d'une interface d'administration très complète et intuitive.
- Peut s'interfacer avec un grand nombre de composants externes (bases de données, annuaires LDAP, serveur de VoIP,… etc.).

- Intègre de nombreux services (serveur de discussion, PubSub, proxy de transfert de fichiers, etc.).
- Permet l'utilisation de nombreux plugins.

III-5-2 Inconvénients :

- Un seul nom de domaine par serveur, impossibilité d'avoir des hôtes virtuels.
- Lenteur de la correction des bugs, openfire ne bénéficie pas d'une importante équipe de développement.
- Pas très répondu et reste peu connu.

III-6 Architecture d'openfire /Jabber:

Openfire fonctionne sur une architecture client/serveur et fonctionne sur le modèle de l'e-mail (amélioré pour permettre la messagerie instantanée). De ce fait, toutes les données envoyées d'un client à un autre passent nécessairement par le serveur openfire (ou un autre serveur Jabber comme : EJabber, etc).

Le client établit une connexion TCP avec le serveur en utilisant le port 5222. La connexion ainsi établie perdure durant toute la durée de la communication, ce qui évite au client de demander confirmation au serveur de la réception des messages.

- Si le client est toujours connecté : Tous les messages destinés au client en question lui sont directement adressés.
- Si le client n'est plus connecté : Le serveur mémorise les messages qui lui sont adressés afin de pouvoir les lui transmettre lors de la prochaine connexion

C'est cette capacité du serveur à savoir si le client est connecté ou pas, qui permet la livraison des messages en temps réel. C'est donc l'état de présence du client qui permet à la messagerie d'être instantanée.

III.6.1 L'adressage :

Les serveurs Jabber, openfire compris utilisent une adresse appelée « Jabber ID » similaire à une adresse e-mail, afin de permettre la communication entre toutes les entités constituant une architecture Jabber. Un Jabber ID (ou JID) est composé de 2 à 3 parties :

- Un nom d'utilisateur (unique sur un serveur)
- Un nom de serveur
- Une ressource qui peut changer, cette partie est optionnelle. La ressource est paramétrée par l'utilisateur qui peut en désigner plusieurs. Généralement, elles permettent de déterminer où l'utilisateur de Jabber est situé. Cette dernière partie est utilisée lorsque l'utilisateur se connecte de plusieurs endroits ou avec plusieurs clients Jabber différents.

L'identifiant formé de ces 3 informations a la forme suivante : utilisateur@serveur/ressource

III.6.2 Le client (l'application de communication) :

Dans une architecture Jabber, le client n'a que très peu de restrictions. En effet, un client Jabber est simple à mettre en œuvre.

Il doit simplement être capable :

- D'établir une connexion TCP avec un serveur XMPP
- D'analyser les messages XML qu'il reçoit afin de pouvoir les interpréter
- Supporter les types de données de bases de XMPP, comme la présence par exemple.

Reporter la complexité sur les serveurs présente deux avantages. D'une part, il est plus facile de mettre à jour les fonctionnalités inhérentes à un tel protocole en évitant de demander au client de mettre à jour lui même son système, d'autre part, les clients Jabber deviennent très faciles à mettre en œuvre. Dans la mesure où l'objectif est de faciliter l'interopérabilité entre les messageries, ce dernier aspect est déterminant.

III.6.3 Le serveur :

La flexibilité est une composante cruciale du protocole XMPP. Ainsi, sur un serveur openfire, les administrateurs ont la possibilité d'ajouter des fonctionnalités au serveur d'origine. Dans le cas d'ajout de tels composants, la simplicité du serveur en lui même n'est pas sacrifiée. En revanche, une certaine complexité est répercutée sur le déploiement du service.

Un serveur Openfire présente trois fonctionnalités de base :

- Gérer les connexions directes avec les clients.
- Communiquer avec les autres serveurs Jabber (openfire ou autres serveurs Jabber).
- Coordonner les différents composants parfois ajoutés par les administrateurs.

La modularité du serveur repose sur le fait que chaque fonctionnalité (enregistrement, présence...etc.) est décrite par une portion de code indépendante.

Cette flexibilité et cette modularité sont à l'origine de l'interopérabilité de ce serveur et de sa simplicité de mise en œuvre.

IV- Conclusion :

Pour la réalisation d'un système de téléphonie IP ou de messagerie instantanée, un développeur a le choix entre plusieurs serveurs open source ou propriétaire, qui offrent les fonctionnalités et les outils nécessaires selon le besoin.

Dans ce chapitre, nous avons présenté deux outils Asterisk (pour la téléphonie IP), et Openfire (pour la messagerie instantanée), leurs fonctionnements, les protocoles qu'ils implémentent. Ces deux serveurs offrent la puissance nécessaire à la conception de telles applications et sont complémentaires et facilement connectables. En outre Asterisk, est une référence dans le domaine de la ToIP qui reste très documenté et Openfire est un choix judicieux vu sa facilité d'utilisation (une administration et une gestion via une interface web) et de cohabitation avec d'autres systèmes.

Dans le chapitre suivant nous nous intéresserons à l'analyse et la conception de notre application.

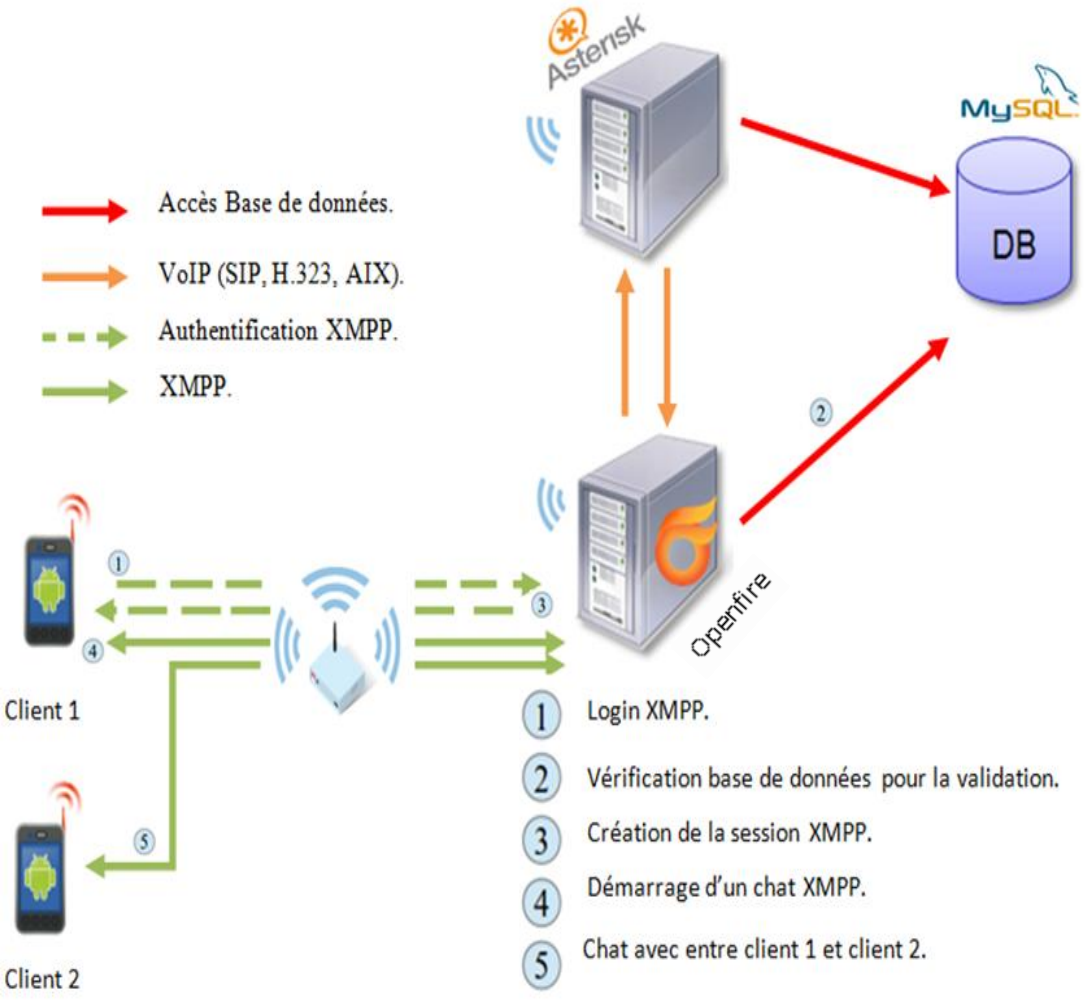
I-Introduction :

Dans cette partie du document, nous présentons notre contribution dans le domaine de la téléphonie IP ainsi que la messagerie instantanée à travers une application Android. Pour ce faire il nous faut d'abord entamer une analyse pour identifier les différents acteurs qui interagissent avec notre application. Puis nous entamerons l'étape de conception en s'appuyant sur les résultats de la phase d'analyse puis nous donnerons la description détaillée du système les classes définies…etc.

L'analyse et la conception de toute solution informatique sont d'une grande importance et elles doivent être traitées avec rigueur et précision, car elles constituent la base du système à développer. Mais avant cela nous donnerons un aperçu de l'architecture de notre solution.

II- Architecture de la solution :

Le schéma ci-dessous illustre l'architecture de notre solution avec les serveurs openfire, Asterisk et Mysql.



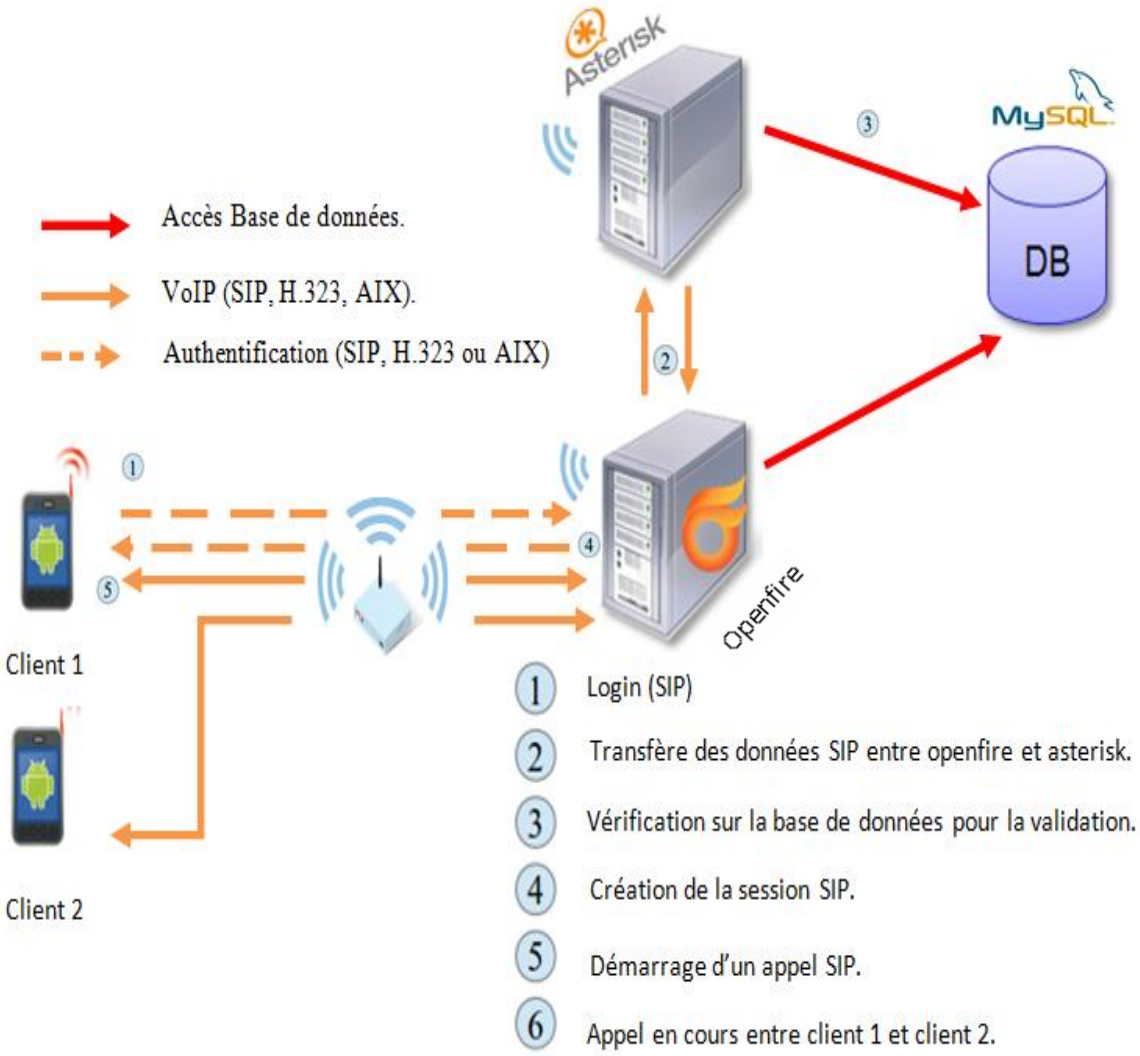


Figure III.1 : Architecture de la solution.

Les clients (applications) sont installés sur les appareils des utilisateurs et connectés avec le serveur Openfire. Celui-ci est en relation avec le serveur Asterisk, tous les deux stockent les données dans le même serveur Mysql.

Asterisk et openfire peuvent jouer le rôle d'une passerelle mutuellement entre eux. Ainsi Asterisk peut faire appel à openfire pour les requêtes XMPP et inversement openfire peut faire appel à Asterisk pour les requêtes de téléphonie IP (SIP, H.323 ou AIX).

Dans la figure IV.1 nous pouvons voir un exemple d'authentification, avec les différents échanges entre le client et le système pour les deux cas appel IP et messagerie instantanée. (Remarque : dans cette figure c'est openfire qui joue le rôle d'une passerelle pour asterisk).

III- Présentation d'UML : [17]

UML (Unified Modeling Language), que l'on peut traduire par « langage de modélisation unifié », est une notation permettant de modéliser un problème de façon standard. Ce langage est né de la fusion de plusieurs méthodes existantes auparavant, il est devenu désormais la référence en terme de modélisation objet, à un tel point que sa connaissance est souvent nécessaire pour obtenir un poste de développeur objet.

UML est un langage de modélisation graphique et textuel destiné à comprendre et à décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. En effet UML est un langage avec une syntaxe et des règles bien définies qui tentent à réaliser les buts décrits grâce à une représentation graphique formée de diagrammes et d'une modélisation textuelle qui vient enrichir la représentation graphique.

III-1 Modélisation avec UML : [18]

La modélisation consiste à créer une représentation simplifiée d'un problème: **le modèle**. Grâce au modèle, il est possible de représenter simplement un problème, un concept et le simuler. La modélisation comporte deux composantes :

- **L'analyse** : c'est-à-dire l'étude du problème ;
- **La conception** : soit la mise au point d'une solution au problème

L'UML permet de représenter des modèles, mais il ne définit pas de processus d'élaboration de ces modèles. Les auteurs d'UML conseillent tout de même une démarche pour favoriser la réussite d'un projet, cette démarche doit être :

- **Une démarche itérative et incrémentale :** Pour comprendre et représenter un système complexe, pour analyser par étapes, pour favoriser le prototypage et pour réduire et maitriser l'inconnu.
- **Une démarche guidée par les besoins des utilisateurs :** Tout est basé sur le besoin de répondre à leur besoin. Chaque étape sera affinée et validée en fonction des besoins des utilisateurs.
- **Une démarche centrée sur l'architecture logicielle :** c'est la clé de succès du développement, les choix stratégiques définiront la qualité du logiciel.

III-2 La démarche de modélisation avec L'UML :

La démarche que nous avons adoptée pour la conception de l'application s'appuie sur quatre étapes :

- ✓ Etape 1 : identification des acteurs et des besoins.
- ✓ Etape 2 : identification et représentation des cas d'utilisation.
- ✓ Etape 3 : élaboration des diagrammes de séquences.
- ✓ Etape 4 : élaboration des diagrammes de classes.

La figure suivante donne la représentation graphique de la démarche de modélisation.

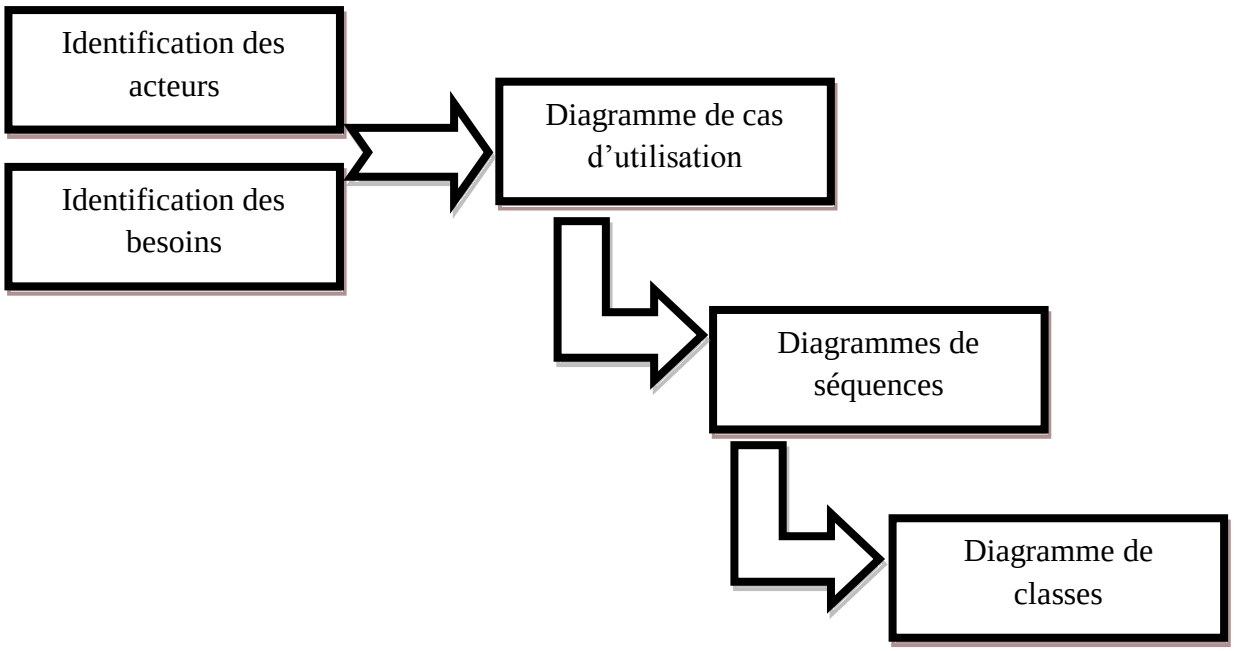


Figure III.2:représentation graphique de la démarche de modélisation.

IV- Analyse :

Cette étape commence par l'étude des cas d'utilisation et de leurs scénarios ainsi que les besoins fonctionnels du système (ce que le système doit faire en réponse à une requête d'un utilisateur), qui aboutira à un ensemble de diagrammes représentant le modèle d'analyse.

IV-1 Identification des besoins :

Les deux objectifs principaux de notre application, c'est de permettre à l'utilisateur de passer des appels IP ainsi que la messagerie instantanée avec un cercle d'amis.

La liste des fonctionnalités de base sur notre application est la suivante :

- Récupérer et afficher la liste des contacts.
- Afficher le profil d'un contact.
- Chater avec un contact.
- Appeler un contact.

IV-2 Identification des acteurs :

Un acteur est une entité externe pouvant interagir avec le système, Le terme acteur ne désigne pas seulement les utilisateurs humains mais également les autres systèmes (matériel et logiciel) externes au système. Un acteur peut avoir les comportements suivants :

- Donner des informations au système.
- Recevoir des informations du système.
- Donner et recevoir des informations du système.

En réponse à l'action d'un acteur, le système fournit un service qui correspond à son besoin. Pour l'identification des acteurs, il faut se concentrer sur le rôle des différentes entités.

Les acteurs de notre système sont toute personne qui utilise une machine (Téléphone, Tablette, etc..) qui fonctionne sous le système Android et utilisent notre application. On les regroupe sous le nom utilisateur « **User** ».

IV-3 Spécification des tâches :

Une tâche est l'ensemble des différentes fonctions qui peut être accédée par un acteur bien spécifié.

L'acteur défini précédemment (**user**) effectue un certain nombre de tâches. Ces dernières sont résumées dans le tableau ci-dessous :

Acteur	Tachés
USER	T1 : Lancer l'application. T2 : Inscription. T3 : Accéder à la liste des contacts. T4 : Rechercher un contact. T5 : Modifier un contact. T6 : Supprimer un contact. T7 : Ajouter des contacts du téléphone. T8 : Accéder au profil d'un contact. T9 : Appeler un contact. T10 : Chater avec un contact. T11 : Accéder au profil User. T12 : Modifier nom User. T13 : Modifier l'image User. T14 : Ajouter un contacte. T15 : Déconnecter/Quitter l'application.

Figure III.3 : Spécification des tâches

IV-4 Spécification des scénarios :

Un scénario est une succession particulière d'enchainements qui s'exécute du début à la fin du cas d'utilisation. A chaque fois qu'une instance d'un acteur déclenche un cas d'utilisation, un scenario est créé. Ce scenario suivra un chemin particulier dans le cas d'utilisation.

Remarque : Durant notre étude les scenarios seront symbolisés par **Si** (**i** : représente le numéro du scenario).

Le tableau suivant résume les tâches et leurs scénarios :

Utilisateur	Taches	Scénarios
User	T01 : Lancer l'application.	S01 : Cliquer sur l'icône de l'application. S02 : Connexion au serveur.
	T02 : Inscription.	S03 : Remplir les champs du formulaire. S04 : Clique sur le bouton « Enregistrer ».
	T03 : Accéder à la liste des contacts.	S05 : Cliquer sur le bouton « Liste Contacts ».
	T04 : Rechercher un contact.	S06 : Placer le curseur sur la zone de recherche, et écrire le nom du contact à rechercher.
	T05 : Modifier un contact.	S07 : Long clique sur le contact. S08 : Cliquer sur le bouton « modifier ». S09 : Saisir les nouvelles informations. S10 : Cliquer sur le bouton « modifier ». S11 : Cliquer sur le bouton « oui » pour confirmer.
	T06 : Supprimer un contact.	S12 : Cliquer sur le bouton « supprimer ». S13 : Cliquer sur le bouton « supprimer » pour la confirmation.
	T07 : Ajouter des contacts du téléphone.	S14 : Cliquer sur le bouton « mettre à jour la liste contacts ».
	T08 : Accéder au profil d'un contact.	S15 : Simple clique sur un contact.
	T09 : Appeler un contact.	S16 : Clique sur le bouton « appeler ».

	T10 : Chater avec un contact.	S17 : Cliquer sur le bouton « message »
	T11 : Accéder au profil User.	S18 : Cliquer sur le bouton « profil »
	T12 : Modifier nom User.	S19 : Cliquer sur le bouton « modifier nom ». S20 : saisir le nouveau nom. S21 : Cliquer sur le bouton « enregistrer».
	T13 : Modifier l'image User.	S22 : Cliquer sur le bouton « modifier image ». S23 : Parcourir le répertoire image/ choisir la nouvelle image. S24 : Enregistrer les modifications.
	T14 : Ajouter un contact.	S25 : Cliquer sur le bouton « ajouter contact ». S26 : Remplir les informations relatives au nouveau contact. S27 : Enregistrer.
	T15 : Déconnecter/Quitter l'application.	S28 : Cliquer sur le bouton « déconnecter ». S29 : Cliquer sur le bouton « oui » pour confirmation.

Figure III.4 : Spécification des scénarios.

IV-5 Les cas d'utilisation :

IV-5-1 Définition :

Un cas d'utilisation permet de mettre en évidence les relations fonctionnelles entre les acteurs et le système étudié. Le format de représentation d'un cas d'utilisation est complètement libre, mais l'UML propose un formalisme et des concepts issus de bonnes pratiques.

L'objectif poursuivi par les cas d'utilisations est de permettre de décrire la finalité des interactions du système et de ces utilisateurs.

IV-5-2 Spécification des cas d'utilisation :

Les figures suivantes présentent des descriptions de quelques cas d'utilisation de notre système :

Cas d'utilisation : Inscription.

Scénarios : S1, S3, S4.

Acteur : User.

Description :

1- L'utilisateur lance l'application après avoir cliqué sur l'icône.
2- Le système affiche un formulaire d'inscription (seulement si c'est le premier lancement de l'application).
3- L'utilisateur remplit le formulaire.
4- L'utilisateur clique sur le bouton « enregistrer ».

Le système enregistre les informations, recherche, enregistre puis affiche la liste des contacts

Figure III.5 : Cas d'utilisation **« Inscription»**

Cas d'utilisation : Supprimer un contacte.

Scénarios : S5, S7, S12, S13.

Acteur : User.

Description :

1- L'utilisateur clique sur le bouton « Liste contacts ».
2- Le système affiche l'interface « Liste contacts ».
3- L'utilisateur fait un long clique sur un contact.
4- Le système affiche une liste de choix (Modifier, supprimer, Ajouter des contacts).
5- L'utilisateur clique sur le bouton « supprimer ».
6- Le système affiche une boite de dialogue de confirmation.
7- L'utilisateur clique sur le bouton « supprimer ».
8- Le système supprime l'entité de la liste des contacts, et réactualise cette dernière avant de l'afficher.

Figure III.6 : Cas d'utilisation **« Supprimer un contacte»**

Cas d'utilisation : Rechercher un contact.

Scénarios : S5, S6.

Acteur : User.

Description :

1- L'utilisateur clique sur le bouton « Liste contacts ».
2- Le système affiche l'interface « Liste contacts ».
3- L'utilisateur place le curseur dans la zone de recherche et tape le nom à rechercher.
4- Le système filtre les noms et affiche le contact correspondant à la recherche.

Figure III.7: Cas d'utilisation **« Rechercher un contacte»**

Cas d'utilisation : Appeler un contact.

Scénarios :S5, S15, S16.

Acteur : User.

Description :

1- L'utilisateur clique sur le bouton « Liste contacts ».
2- Le système affiche l'interface « Liste contacts ».
3- L'utilisateur fait un clic simple sur le contact.
4- Le système affiche le profil du contact.
5- L'utilisateur clique sur le bouton « appeler ».
6- Le système affiche l'interface appel en cours, et lance l'appel.

Figure III.8 : Cas d'utilisation **«Appeler un contacte»**

Cas d'utilisation : Chater avec un contact.

Scénarios :S5, S15, S17.

Acteur : User.

Description :

1- L'utilisateur clique sur le bouton « Liste contacts ».
2- Le système affiche l'interface « Liste contacts ».
3- L'utilisateur fait un clic simple sur le contact.
4- Le système affiche le profil du contact.
5- L'utilisateur clique sur le bouton « message ».
6- Le système affiche l'interface de chat.

Figure III.9: Cas d'utilisation **« Chater avec un contact»**

Cas d'utilisation : Accéder à la liste des contacts.

Scénarios : S1, S5.

Acteur : User.

Description 1:

1- L'utilisateur clic sur l'icône de l'application pour la démarrer.
2- Le système affiche une barre de progression.
3- Le système récupère la liste des contacts du téléphone.
4- Le système affiche la liste des contacts.

Description 2 :

1- L'utilisateur lance l'application.
2- Le système affiche la liste des contacts.

Description 3 :

1- L'utilisateur clique sur le bouton « Liste contacts ».
2- Le système affiche l'interface « liste contacts ».

Figure III.10: Cas d'utilisation **« Accéder à la liste des contacts».**

Cas d'utilisation : Ajouter un contact.

Scénarios :S18, S25, S26, S27.

Acteur : User.

Description :

1- L'utilisateur clique sur le bouton « profil ».
2- Le système affiche le profil user.
3- L'utilisateur clique sur le bouton « ajouter contact ».
4- Le système affiche un formulaire de coordonnées.
5- L'utilisateur remplit les champs d'informations.
6- L'utilisateur clique sur le bouton « ajouter ».
7- Le système enregistre les informations.

Figure III.11 : Cas d'utilisation **« Ajouter un contact».**

Cas d'utilisation : Déconnecter/Quitter l'application.

Scénarios: S18, S28, S29.

Acteur : User.

Description :

1- L'utilisateur clique sur le bouton « Profil ».
2- Le système affiche l'interface « Profil ».
3- L'utilisateur fait un clique sur le bouton « déconnecter ».
4- Le système déconnecte et ferme l'application.

Figure III.12 : Cas d'utilisation **«Déconnecter/Quitter l'application »**

V- Conception :

Le processus de conception de notre application repose sur l'organisation conceptuelle, logique et physique des données collectées durant la phase analyse. En effet, elle s'appuie essentiellement sur quelques diagrammes du langage de modélisation UML.

En effet, après l'identification des différents acteurs ainsi que quelques cas d'utilisation qui sont mis en œuvre par ces acteurs, le diagramme de cas d'utilisation général est dégagé.

Chaque cas d'utilisation se traduit par un ou plusieurs scénarios. Chaque scénario fait l'objet d'une description sous forme graphique à l'aide d'un diagramme de séquence.

Enfin, une identification des classes est fournie par la synthèse des diagrammes de séquences, ainsi le diagramme de classe sera dégagé.

V-1 Le diagramme de cas d'utilisation :

Lors de la phase d'analyse nous avons pu identifier les acteurs ainsi que les cas d'utilisation associés à ces derniers. Ce qui nous donne l'opportunité d'élaborer le diagramme des cas d'utilisation.

Les diagrammes de cas d'utilisation permettent de représenter un ensemble de cas d'utilisation, d'acteurs et leurs relations.

- **La relation d'inclusion (include) :** Elle indique que le cas d'utilisation source contient aussi le comportement décrit dans le cas d'utilisation destinataire. Cette relation permet de décomposer des comportements et de définir les comportements partageables entre plusieurs cas d'utilisations.
- **La relation d'extension (Extend) :** Elle indique que le cas d'utilisation source ajoute son comportement au cas d'utilisation destinataire. L'extension peut être soumise à des conditions.

V-2 Diagrammes de séquences :

Un diagramme de séquences est un diagramme qui représente la séquence de messages entre les objets au cours d'une interaction. Un diagramme de séquences comprend un groupe d'objets, représentés par des lignes de vie, et les messages que ces objets échangent lors de l'interaction. Ils peuvent également représenter les structures de contrôle entre des objets.

Un diagramme de séquences met toujours l'accent sur l'ordre chronologique des messages.

Les composants d'un diagramme de séquences sont les suivants :

Les objets : ils apparaissent dans la partie supérieure, ce qui facilite l'identification des classes qui participent à l'interaction.

Les messages : ils sont représentés par des flèches directionnelles. Au-dessus de ces flèches figurent un texte nous informons du message envoyé entre les objets.

Ci-dessous nous présentons quelques diagrammes de séquences :

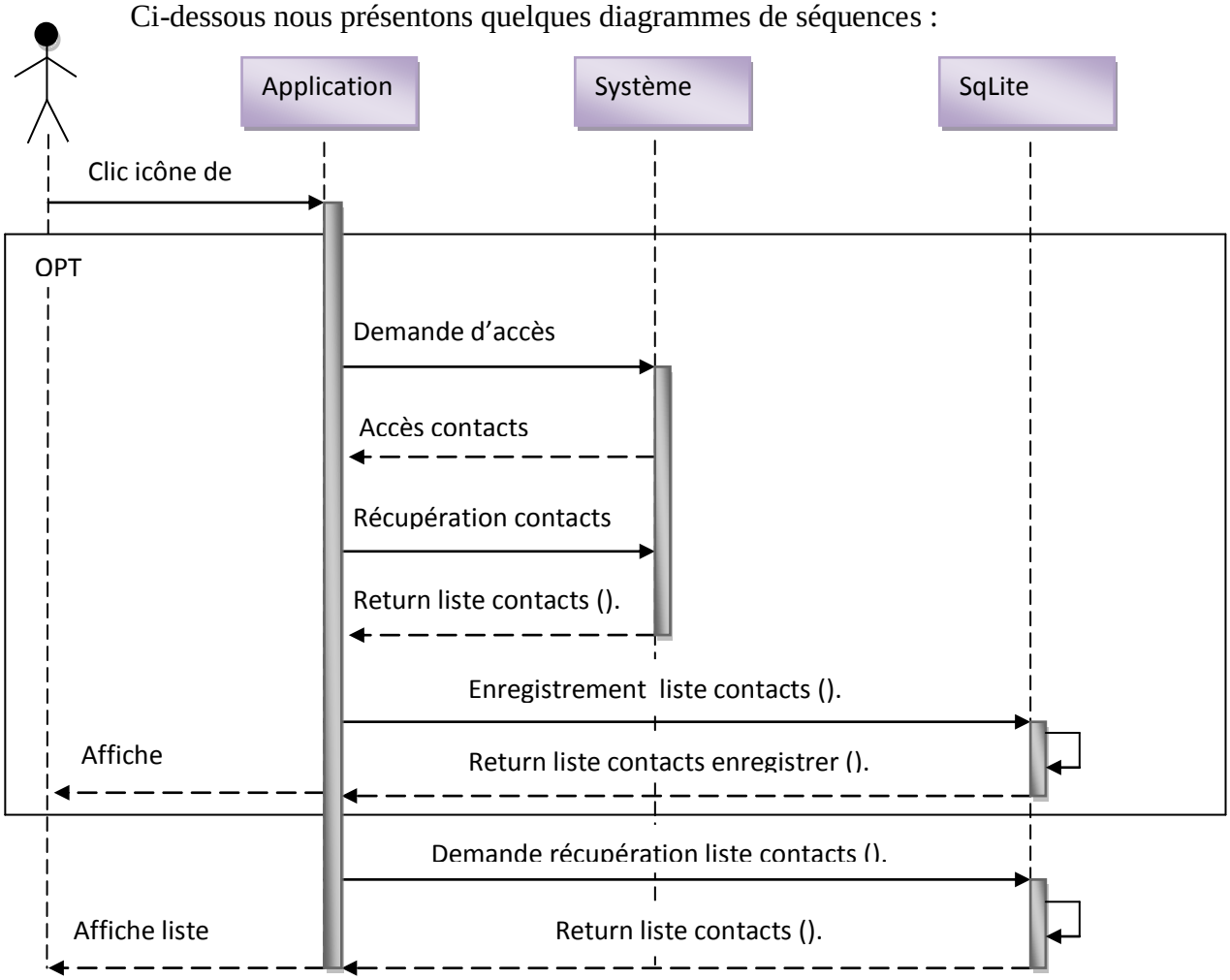


Figure III.14 : Diagramme de séquences du cas d'utilisation « Accéder à la liste de contacts ».

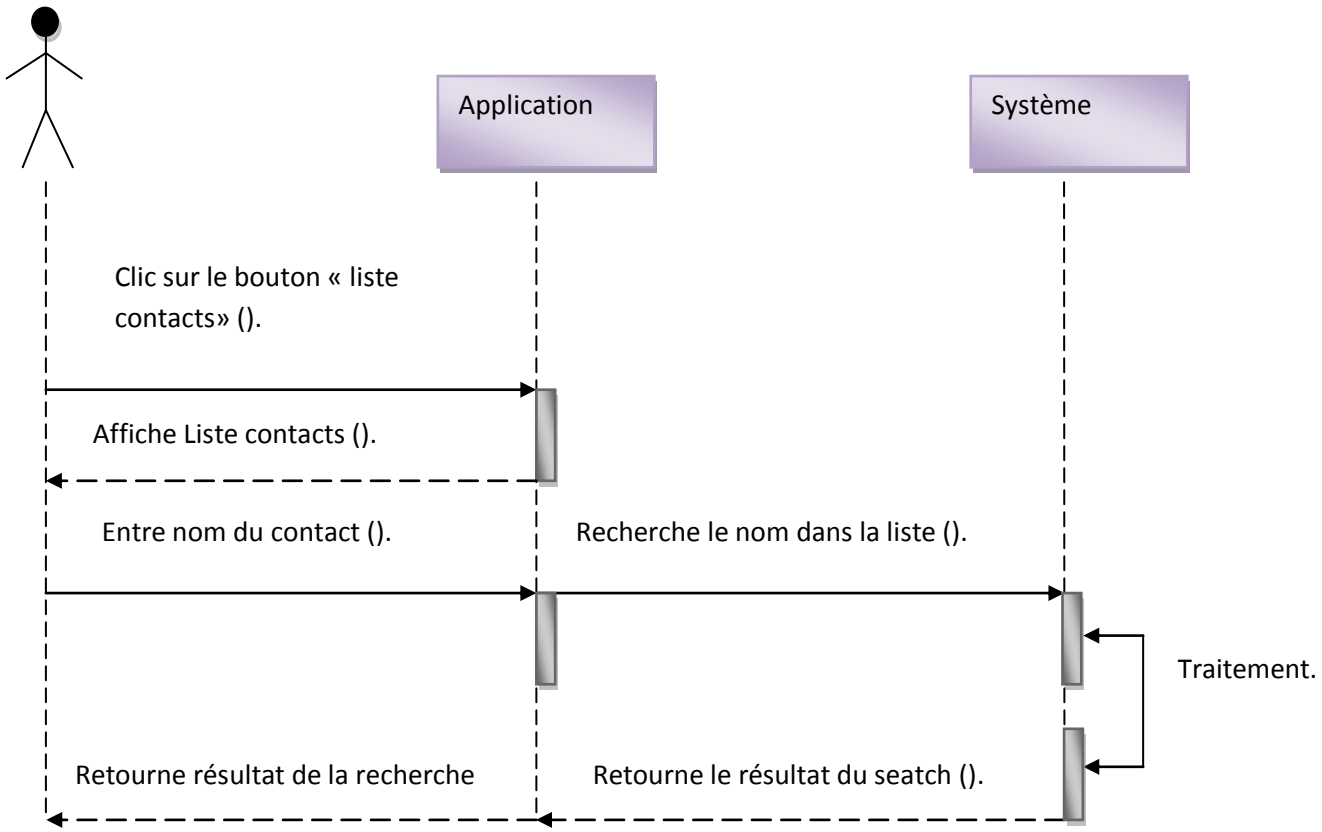


Figure III.15 : Diagramme de séquences du cas d'utilisation « Rechercher un contact ».

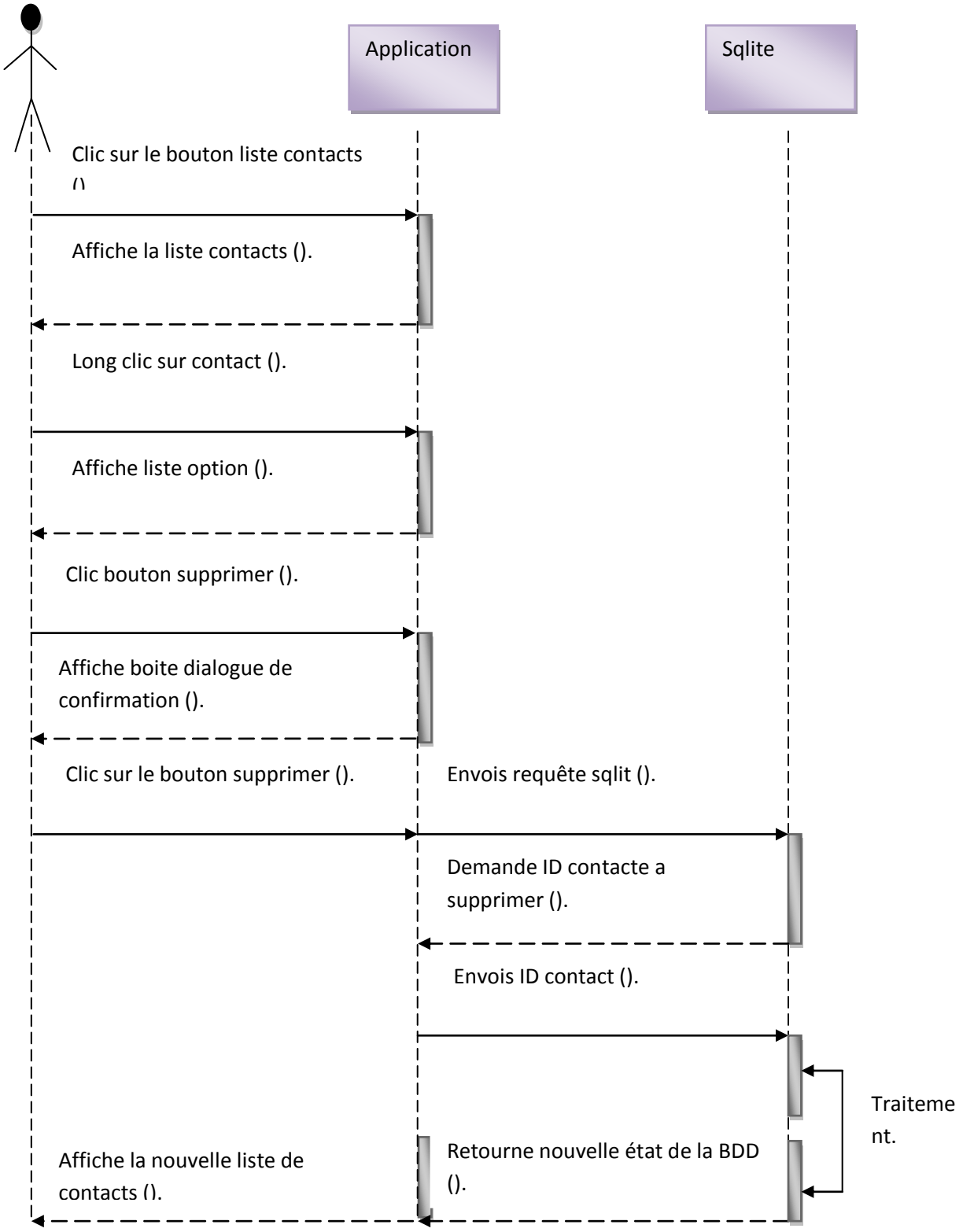


Figure III.16 : Diagramme de séquences du cas d'utilisation « supprimer un contact ».

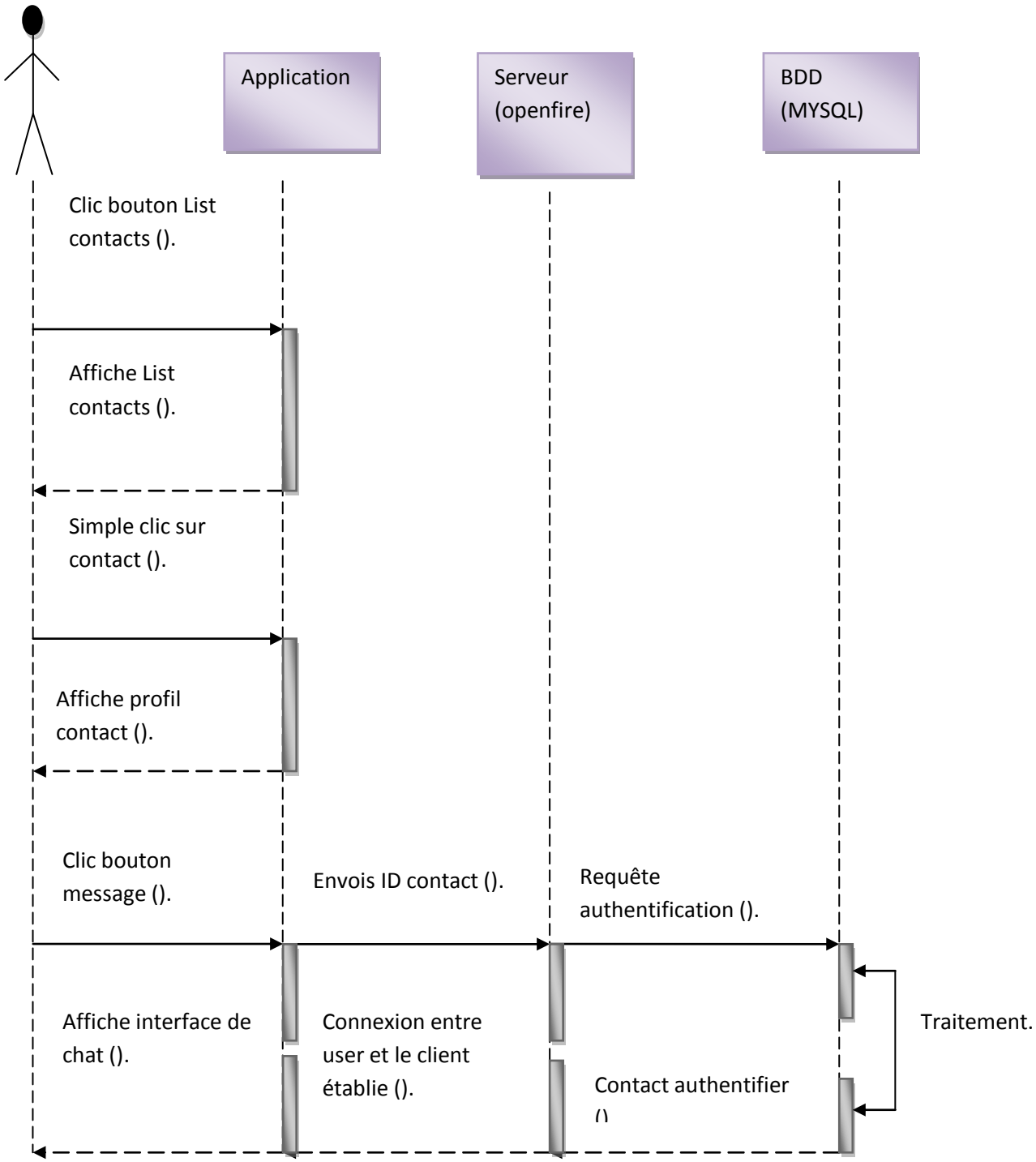


Figure III.17 : Diagramme de séquences du cas d'utilisation « chater avec un contact ».

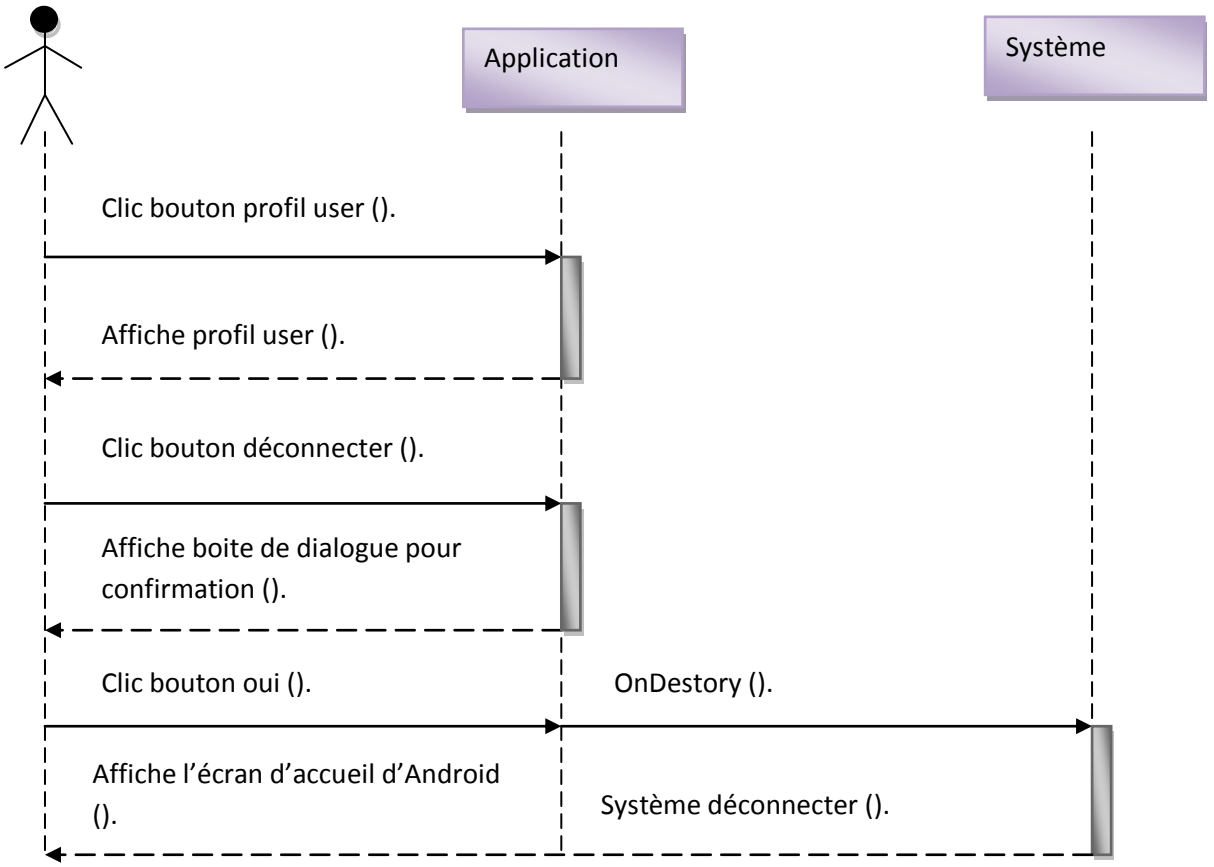


Figure III.18 : Diagramme de séquences du cas d'utilisation « déconnecter/Quitter l'application ».

V-3 Diagramme de classes :

Le diagramme de classe est considéré comme le plus important de la modélisation orienté objet, il est le seul obligatoire lors d'une telle modélisation.

Alors que le diagramme de cas d'utilisation montre un système du point de vus des acteurs, le diagramme de classes en montre la structure interne. Il contient principalement des classes reliées par des associations et chaque classe contient des attributs et des opérations.

V-4 Le niveau de données :

- **Schéma de la base de données Contacts :**

  La base de données contacts ne contient qu'une seule table représentée ci-dessous :

Champ	Signification	Type de donnée	Observation
_ID_CONTACT	Identifiant de contact	INTEGER	Clé primaire
ID_CONTACT_TEL	Identifiant de contact dans le téléphone	STRING	Clé étrangère
NOM_CONTACT	Nom de contact	STRING	
NUMERO_CONTACT	Numéro de téléphone contact	STRING	
MEMBRE	Le contact est membre ou pas	INTERER(1)	
IMAGE_CONTACT	L'image de contact	STRING	
ETAT_SUPP_CONTACT	L'état de contact, supprimé ou pas	INTERER(1)	

Figure III.20 : Schéma de la base de données Contacts.

- **Schéma de la base de données Utilisateur :**

La base de données utilisateur ne contient qu'une seule table représentée ci-dessous :

Champ	Signification	Type de donnée	Observation
_ID_UTILISATEUR	Identifiant de l'utilisateur	INTERER	Clé primaire
NOM_UTILISATEUR	Nom d'utilisateur	STRING	
NUMERO_UTILISATEUR	Numéro de téléphone de l'utilisateur	STRING	
IMAGE_UTILISATEUR	L'image de l'utilisateur	STRING	

Figure III.21 : Schéma de la base de données Utilisateur.

VI- Conclusion :

Nous avons consacré ce chapitre à l'étude, l'analyse et la conception de notre application. Dans le chapitre suivant nous aborderons la réalisation et la mise en œuvre de l'application.

I- Introduction :

Dans ce chapitre, nous présentons notre environnement de travail les étapes d'installation et de configuration des différents outils utilisés ainsi que les composantes applicatives réalisées.

Les différentes implémentations, installations et les configurations ont été réalisées sur une plateforme Windows 7.

II- installation et configuration des serveurs :

II-1 configuration de Mysql :

Une fois que l'installation est démarrée l'interface graphique du gestionnaire d'installation de Mysql propose différentes configurations. Nous avons choisi dans l'ordre les suivantes :

1- Après avoir validé les informations de la licence, puis choisi d'installer Mysql Product une interface nous donne le choix d'installation (seulement le serveur, tous le produit, par défaut ou customisé), nous avons choisi d'installer le serveur seulement.

2- Dans cette étape on nous propose la configuration à adopter pour notre serveur, nous laissons la configuration par défaut (config type : server machine et le port TCP/IP : 3306 et nous cochons la case open firewell Access pour que le port soit activé).

3- Dans l'interface suivante on choisit un mot de passe pour l'administrateur (root) et on clique sur suivant. Dans la fenêtre d'après on a laissé les configurations telles quelles (le nom du système Mysql55 et démarrage du système comme compte standard).

4- L'installation et la configuration sont terminées on peut choisir de copier le log au clipbord et cliquer sur finish.

5- Une fois l'installation terminée on démarre Mysql on tape le mot de passe root et on tape la commande (creat database openfire).

Une fois que l'on a créé la base de données openfire on passe à l'étape suivante qui consiste à Installer et à configurer Openfire.

II-2 Openfire :

II-2-1 Installation d'Openfire :

Openfire est open source et il est facilement téléchargeable. Pour notre part on n'a pas téléchargé de fichier exécutable mais un répertoire complet zipé, pas besoin de l'installer il suffit de le dézipér et le démarrer pour le configurer.

Nous avons téléchargé notre répertoire zipé sur [27], on a choisi la version 3.9.3 sous Windows. Une fois dézipé et démarré on peut passer à la configuration du serveur.

II-2-2 Configuration d'Openfire :

Openfire offre plusieurs configurations et options selon les besoins d'utilisation. Cette étape ainsi que la gestion du serveur se font via une interface web. Dans ce qui suit nous verrons les différentes étapes de notre configuration :

Une fois openfire lancé, une petite fenêtre s'ouvre contenant quelques informations comme le nom de la machine, le port de la console d'administration 9090 ..etc. on clique sur launch pour démarrer l'interface web et commencer la configuration dans l'ordre qui suit :

1- La première interface après le lunch et celle de la langue, on a le choix entre plusieurs langues.

2- Dans l'interface suivante on peut changer le nom du domaine (serveur) ainsi que le numéro du port de la console d'administration, on a choisi comme nom du domaine « gallo.com » et on a laissé les deux ports d'administration sécurisé/non sécurisé a 9090 et 9091.

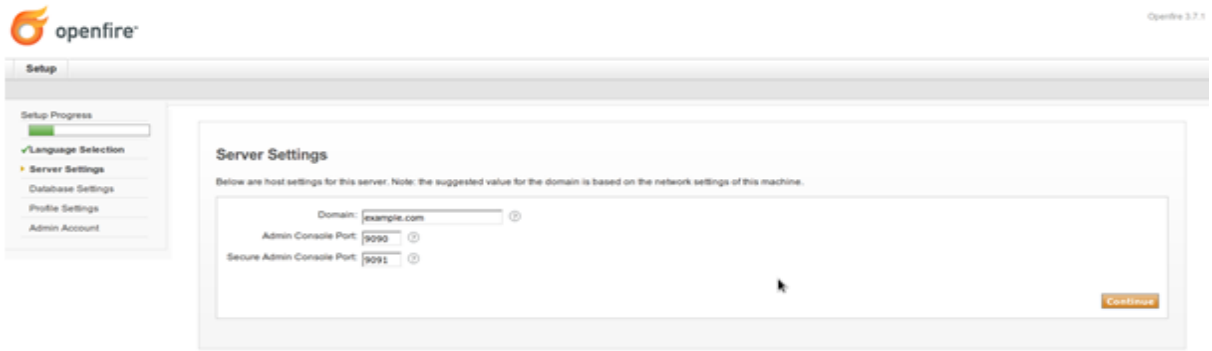


Figure IV.1 : Paramètres du serveur Openfire (domaine / ports)

3- A cette étape on nous demande de choisir la connexion à la base de données. Il ya deux choix :

a- Connexion standard à la base de données (Utiliser une base de données externe avec un pool de connexion interne).

b- Base de données embarquée (Utiliser une base de données embarquée, qui fonctionne grâce à HSQLDB (HyperSQL DataBase), C'est la méthode pour démarrer le plus rapidement. Cependant elle n'offre pas le même niveau de performance qu'une base de données externe).

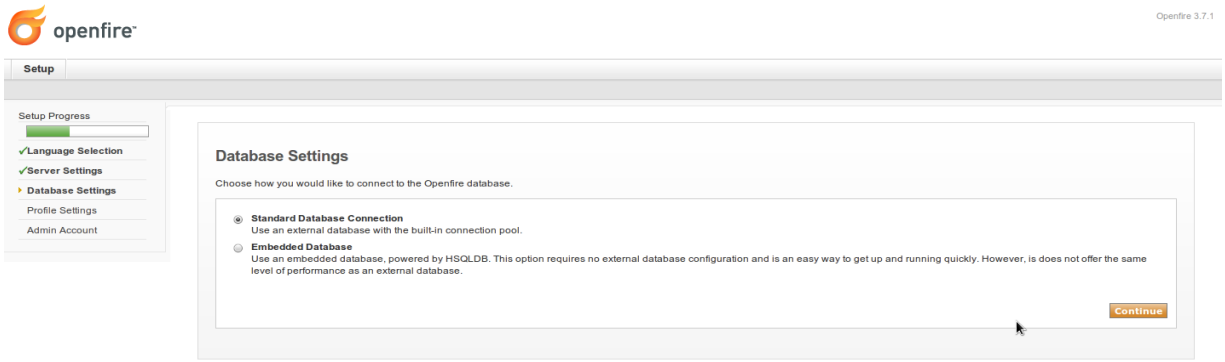


Figure IV.2 : Choix du type de connexions à la base de données.

Nous avons choisi la première option pour une question de performance.

4- Après avoir choisi le type de connexion à la base de données on va paramétrer notre base de données comme suit :

a- Pilote de la base de données prédéfini : « Mysql ».

b- Classe du pilote JDBC : « com.mysql.jdbc.Driver ».

c- URL de la base de données : « jdbc :mysql://[host :name] :3306/(database-name) ? » avec host name = (notre nom de domaine « gallo.com »), 3306= le port de connexion à la base de données (nous l'avons validé lors de la configuration de Mysql) et enfin database-name = (le nom de notre base de données dans notre cas « openfire »).

d- Et enfin le nom d'utilisateur et le mot de passe de la base de données (par défaut l'utilisateur de Mysql est « root ») et on clique sur continuer.

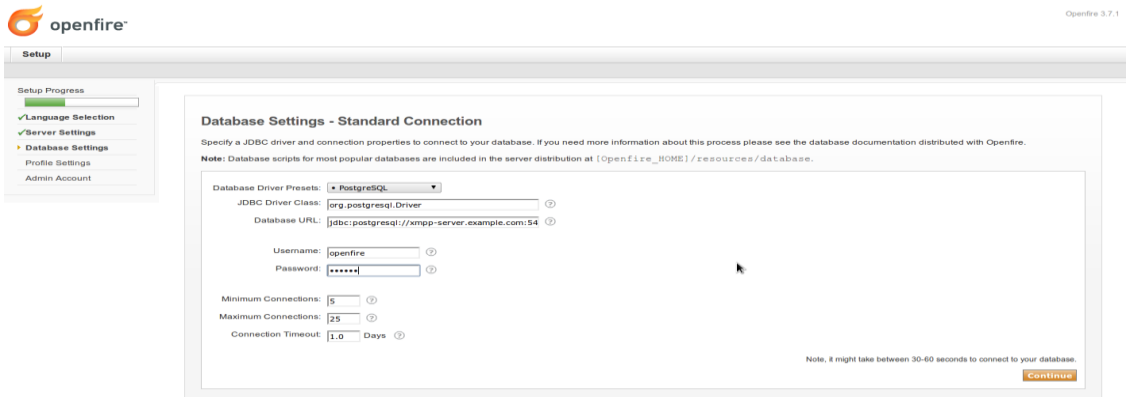


Figure IV.3 : Paramètres de la base de données XMPP.

5- Dans l'interface ci-dessus on peut choisir l'utilisateur et le groupe système à utiliser avec Openfire.

   a- Par défaut (Enregistre les utilisateurs dans la base de données Openfire. Recommander pour les déploiements simples).

   b- Serveur LDAP (Intégrer avec un serveur LDAP comme active directory ou OpenLDAP en utilisant le protocole LDAP. Les utilisateurs et le groupe doivent être présents dans le serveur LDAP et visibles en lecture seule).

   c- ClearspaceIntegration.

Nous avons choisi l'option par défaut car nous n'avons pas besoin d'un serveur LDAP (qui est une sorte d'annuaire nous avons besoin que les utilisateurs puissent s'inscrire directement et être enregistrés sur la base de données).

6- Dans l'interface suivante on choisit un mot de passe pour l'administrateur de openfire et on saisit son adresse e-mail (par défaut le nom d'utilisateur d'administration et « admin ».

7- La dernière interface nous informe que l'installation est terminée et nous demande de nous connecter à la console d'administration avec user : admin et le mot de passe choisi précédemment.

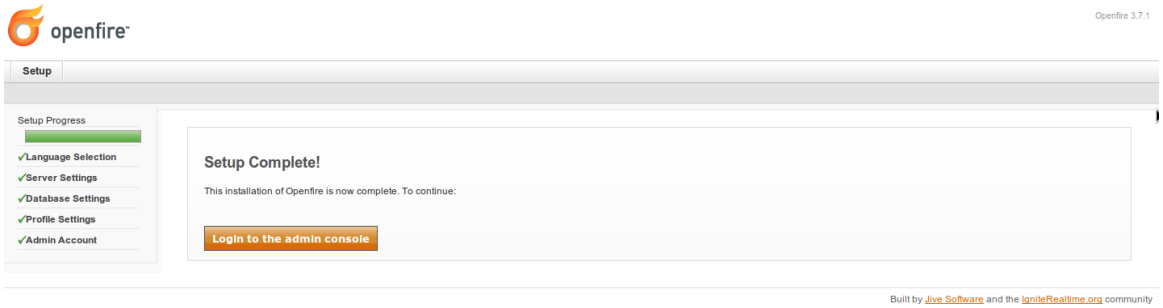


Figure IV.4 : Fin de la configuration d'Openfire et demande de login.

III- Préparation d'Eclipse :

Eclipse est un IDE qui permet de programmer dans différents langages grâce à ses nombreux plug-ins et notamment le plug-in d'Android. Une interface spécifique permet de gérer des fichiers java et compliquer ses programmes. Les fichiers sont organisés selon une arborescence qui correspond aux paquetages java définis. L'analyse syntaxique permet de mettre en valeur les mots clés dans les fichiers java. Eclipse dispose aussi d'un système d'auto compilation des fonctions, de détection des erreurs syntaxiques en temps réel sans oublier un système de débogage permettant d'exécuter ses programmes pas à pas.

Afin de développer sous Android, nous avons installé le plugin ADT (Androit Development Tools) qui ajoutera à Eclipse les fonctionnalités spécialisées dans le développement sous Android.

III-1 Installation du SDK :

Le SDK (Software Developpement Kit) est un ensemble d'outils de développement qui permet de créer de nouvelles applications. Son installation se fait en lançant le SDK-Manager.exe, téléchargé du site officiel d'Android, la fenêtre représentée dans la figure IV.5 s'ouvre.

On fait le choix du SDK puis il suffit de cliquer sur installer packages pour lancer l'installation. Une fois terminée on peut créer notre projet File/new project/android.

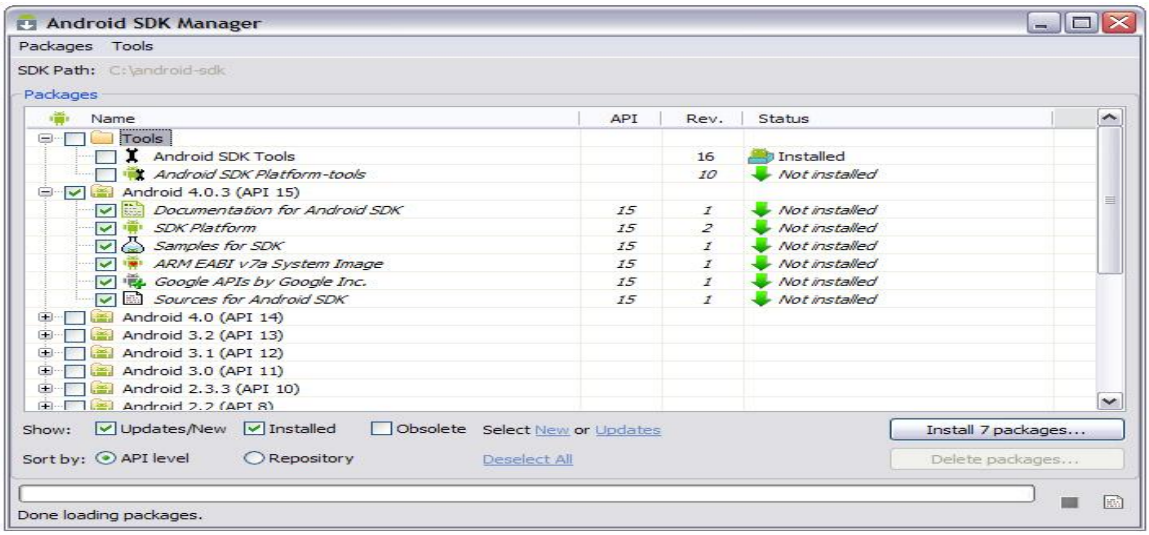


Figure IV.5 : Choix de la version de SDK.

III-2 Télécharger et intégrer la bibliothèque Asmack au projet :

Asmack est un portage non officiel sous Android de Smack API. Smack est une bibliothèque Open Source XMPP (Jabber) cliente pour la messagerie instantanée. Une bibliothèque écrite entièrement en java, elle peut être intégrée dans des applications pour créer un client XMPP complet ou des intégrations XMPP simples comme l'envoi de messages de notification et des dispositifs de présence-favorable. Elle permet aux applications de se connecter plus facilement aux serveurs de messagerie instantanée Jabber/XMPP et elle comporte plusieurs méthodes qui peuvent faciliter grandement toute sorte d'implémentations relative à la messagerie XMPP. Elle est téléchargeable sur le lien suivant [29].Pour l'intégrer à notre projet il suffit juste de la copier dans le répertoire de notre application.

IV- Environnement :

IV-1 Environnement matériel :

Pour accomplir notre travail nous avons à notre disposition l'environnement matériel suivant :

- Un ordinateur portable Intel® I3 U330 @ 1.20GHz 1.20GHz 4Go
- Une clé 3G qui fait office de routeur.
- Deux Smartphones :
  - HTC ONE X avec les caractéristiques suivantes :
    - OS : Android 4.2.2
    - CPU Quadcore NvidiaTegra 3 – 1.5 GHz
    - WIFI 802.11 a/b/g/n
  - SONY
    - Android : 2.3.7
    - CPU : ST-Ericsson NovaThor U8500- 1 GHz
    - WIFI : 802.11 b/g/n + DLNA

IV-2 Environnement cible :

Notre application est destinée aux Smartphones et tablettes équipés d'un système d'exploitation Android 2.1 ou une version ultérieure.

V- Fonctionnement de l'application :

V-1Présentation générale de l'application :

L'application que nous avons développée a une activité principale nommée « Tableactivite.java », cette activité contient deux fils « ListeContact.java » et « profile.java ».On peut permuter entre les deux simplement en cliquant sur le bouton qui correspond à l'interface voulue. L'activité « ListeContacts.java » permet de gérer les contacts, elle fait appel à plusieurs sous activités comme « choixutilisateur.java » qui permet de passer un appel ou chater avec un contact. Quant à « profile.java », elle permet de gérer le profil utilisateur comme ajouter un contact, modifier le profil..etc.

Pour ce faire une modification au niveau du manifest est effectuée comme on peut le voir dans la figure ci-dessous :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.gallo"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="7"
        android:targetSdkVersion="19" />
   <uses-permission android:name="android.permission.READ_CONTACTS"/>
   <uses-permission android:name="android.permission.WRITE_CONTACTS" />
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme"
       >
        <activity
            android:name="com.gallo.Inscription"
            android:label="@string/app_name"
          >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service  android:enabled="true" android:name="com.gallo.Connexion"></service>
       <activity android:name="com.gallo.Tableactivite"></activity>
       <activity android:name="com.gallo.ListeContact"></activity>
       <activity android:name="com.gallo.Choixutilisateur"></activity>
       <activity android:name="com.gallo.XmppChat"></activity>
       <activity android:name="com.gallo.Appler"></activity>
       <activity android:name="com.gallo.Profile"></activity>

   </application>
```

Figure IV.6: Fichier Manifest.xml.

Dans la figure IV.6 on peut voir la présence de nos activités mise entre les balises <activity>et</activity>. Ainsi que le service connexion qui nous permet de nous connecter aux serveurs.

On voit aussi la présence de ces trois permissions :

- Android.permission.READ_CONTACTS : qui nous permet d'avoir accès aux contacts du Smartphone.
- Android.permission.WRITE_CONTACTS : qui nous permet de modifier des contacts sur le Smartphone.
- Android.permission.INTERNET : qui nous permet d'avoir accès à internet.

V-2 Les fonctions de notre application :

❖ **Fonction 1 (Formulaire d'inscription) :**

Une fois l'application installée, lors du premier démarrage le formulaire ci—dessous apparait. Ce formulaire sert à l'inscription de l'utilisateur avec les informations requises sur la base de données centralisées et celle du cellulaire lui-même.

Figure IV.7 : Formulaire d'inscription.

❖ **Fonction 2 (Connexion) :**

Juste après la sauvegarde du formulaire d'inscription, la classe « connexion.java » récupérera les informations de l'utilisateur à partir de la base de données interne et procédera à l'authentification avec le serveur. Cette étape va connecter le client à notre système.

❖ **Fonction 3 (Récupération de la liste des contacts) :**

Lors de la première utilisation de notre application, cela va rechercher puis enregistrer tous les contacts présents sur l'appareil dans une base de données propre à cette application. Le client verra une barre de progression comme le montre la figure suivante.

Figure IV.8 : Chargement de la liste des contacts.

❖ **Fonction 4 (Afficher la liste des contacts) :**

Une fois les contacts récupérés, ils seront affichés.

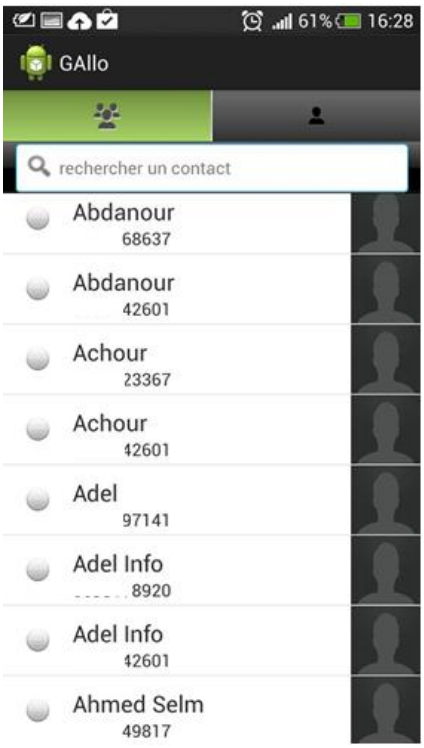


Figure IV.9 : Liste des contacts.

❖ **Fonction 5 (Rechercher un contact) :**

La recherche d'un contact se fait dans l'activité principale « Listecontacts.java » dont l'interface n'est autre que celle de la liste des contacts vue précédemment. Il suffit d'écrire le nom du contact dans la zone dédiée à cet effet pour voir apparaitre le résultat comme sur la figure suivante :

Figure IV.10 : Recherche d'un contact.

❖ **Fonction 6 (Suppression/modification d'un contact) :**

Il suffit d'un clic long sur un contact pour voir s'afficher une liste de choix (option) visibles sur la figure IV.11:

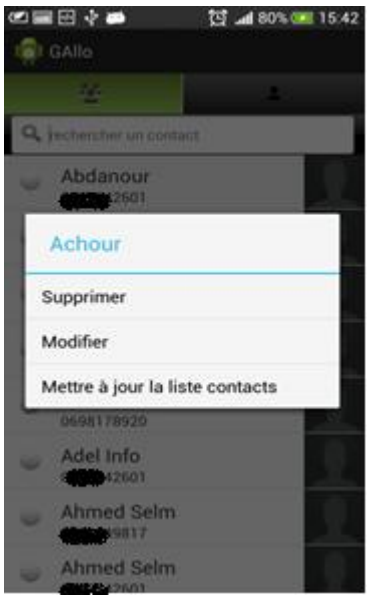


Figure IV.11: Liste d'options pour un contact.

1- **Supprimer un contact :** une fois l'option de suppression choisie une boite de confirmation s'affiche, si elle est confirmée le contact sera supprimé de la liste des contacts.

Figure IV.12 : Suppression d'un contact.

2- **Modification d'un contact :** si on choisit modifier, la boite de dialogue ci-dessous s'affiche nous demandons de rentrer le nouveau nom du contact (on ne peut pas modifier le numéro de téléphone d'un contact).

Figure IV.13 : Modification du nom d'un contact.

Après cela une boite de dialogue s'affiche comme le montre la figure IV.14 si on confirme cette modification les changements seront enregistrés dans la base de données et affichés sur la liste des contacts.

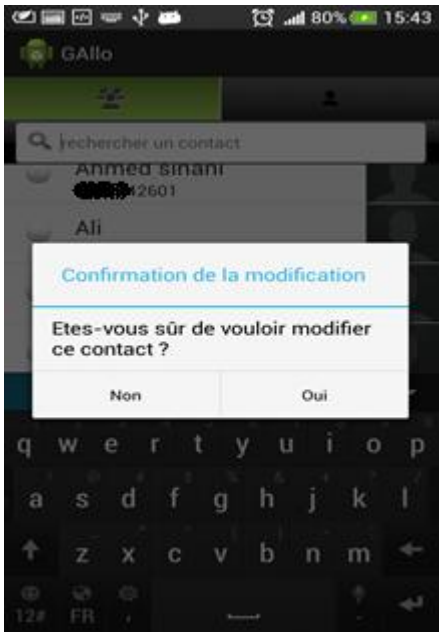


Figure IV.14: Confirmation d'une modification sur un contact.

❖ **Fonction 7 (accéder au profil d'un contact) :**

Il suffit d'un simple clic sur le contact pour voir le profil de ce dernier. Pour réaliser cela nous avons récupéré l'identifiant du contact (son identifiant dans la base de données interne) lors du clic, après on l'envoie dans un Intent à l'activité « choixutilisateur.java » grâce à la méthode « putExtra() » qui permet de transporter des informations au sein des Intents. Ainsi l'activité « choixutilisateur.java » nous affichera les informations du contact comme le montre la figure IV.15.

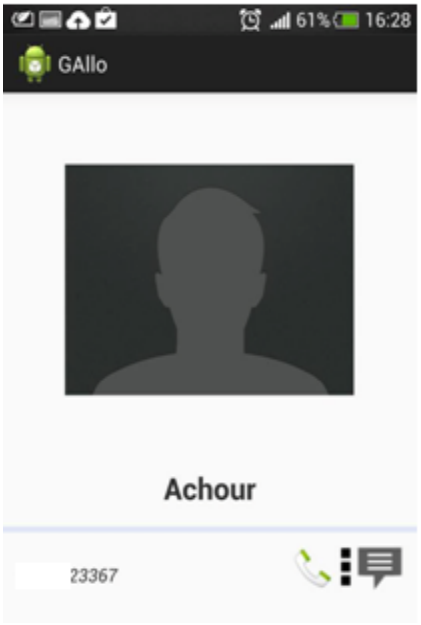


Figure IV.15: Profil d'un contact.

❖ **Fonction 8 : Appeler un contact :**

Une fois sur le profil d'un contact on clique sur le bouton appeler pour passer l'appel.

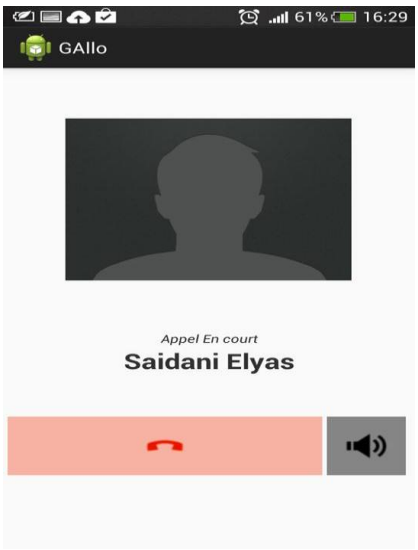


Figure IV.16 : Interface d'appel.

❖ **Fonction 9 : Chater avec un contact :**

Du profil d'un contact il suffit de cliquer sur le bouton message pour accéder à l'interface de chat comme le montre la figure IV.17 ci-dessous. Pour faire cela, en premier lieu nous avons envoyé le numéro du contact à partir de l'activité « choixcontact.java » vers « XmppChat.java » grâce à des Intents qu'on a modifié pour qu'il soit un Jabber ID (JID) en lui ajoutant « **@gallo.com** » (gallo.com étant le nom de notre service). Ce dernier sera utilisé pour envoyer les messages au contact, l'application utilise un auditeur de paquets pour obtenir les messages reçus.

Figure IV.17 : Fenêtre de chat.

❖ **Fonction 10 (Voir le profil utilisateur) :**

L'interface principale de l'application est un Tableactivite, il suffit d'un clic sur l'icône de droite pour basculer sur le profile User ou inversement pour basculer sur la liste des contacts. Le profile User est visible sur la figure IV.18.

Figure IV.18 : Profil User (Utilisateur).

❖ **Fonction 11 (Modifier le nom User) :**

Dans le profile User, il suffit de cliquer sur le bouton modifier nom pour que la boite de dialogue suivante apparait, en nous demandons le nouveau nom user (utilisateur).

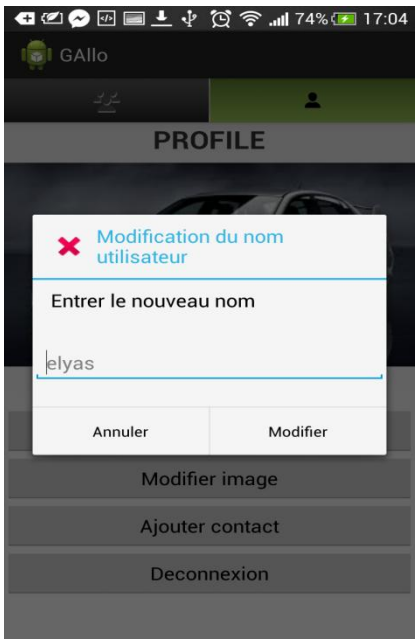


Figure IV.19: Modifier le nom User (utilisateur).

❖ **Fonction 12 (Modifier l'image de profil) :**

Dans le profile User, il suffit de cliquer sur le bouton Modifier image pour accéder à la galerie des images du Smartphone.

❖ **Fonction 13 (Ajouter un contact) :**

Dans le profile User, il suffit de cliquer sur le bouton ajouter un contact pour avoir un formulaire à remplir et à sauvegarder.

Figure IV.20: Ajouter un contact.

❖ **Fonction 13 : Déconnecter/Quitter l'application :**

Dans le profile User, il suffit de cliquer sur le bouton déconnexion, une petite boite de confirmation apparait une fois cette déconnexion est confirmée l'application sera fermée et détruite (elle y'a plus de trace d'elle sur la ram toute fois elle n'est pas désinstallée).

VI- Conclusion :

Nous avons présenté dans ce chapitre l'architecture de notre système, les configurations du serveur Openfire et Mysql server, ainsi qu'une vue globale sur notre application. La partie mise en œuvre traduit les besoins fonctionnels et techniques déjà définies par l'implémentation des différentes interfaces.

L'élaboration de notre projet de fin d'étude était dans le but de concevoir une application dédiée aux Smartphones Android. Cette application sociale permet aux clients de communiquer avec un cercle d'amis à tout moment via un réseau local.

Notre application ainsi réalisée permet de :

- Récupérer la liste des contacts déjà existants sur l'appareil et ainsi éviter au client d'ajouter constamment des amis déjà répertoriés sur son Smartphone.
- Ajouter de nouveaux amis.
- Communiquer avec des amis grâce à la messagerie instantanée.

Ce projet nous a donné la chance de nous intégrer dans le domaine de la recherche et plus encore nous confronter au domaine technique, et ainsi nous offrir l'occasion d'acquérir de nouvelles connaissances dans le domaine de la mobilité, des réseaux et des nouvelles technologies de communication, une meilleure maitrise du langage de programmation java, mais surtout il nous a permis de comprendre pourquoi des applications comme What'sAPP, Skype…etc, Se vendent à plusieurs milliard de dollars et de mieux voir le long chemin qu'ils nous reste à parcourir pour arriver à une vrai performance.

Et enfin, notre travail ainsi présenté reste un prototype sur lequel nous espérons apporter plus de consistance pour le rendre meilleur et plus fonctionnel, ainsi nous aimerions améliorer ce qui suit :

- Régler la téléphonie IP.
- Ajouter les services de notification en arrière plans et permettre à l'application d'être parfaitement en marche même hors utilisation.
- Ajouter un annuaire de contacts.
- Exploiter d'autres Smartphones : Apple, Windows, BlackBerry, et pourquoi pas le nouveau venu sous Linux.
- Passer du réseau local à la grande toile mondiale.

BIBLIOGRAPHIE

[1] :http://www.le-libriste.fr/android/.

[2] :EspiauFrédéric**,** Créer des applications pour Android.s.l : le livre de zero, 2012.

[3] :Alain Menu, Android How to, édition 2.1, septembre 2013.

[4] :http://developer.android.com/

[5] : Reto Meier,Android 4 : développement d'application avancée,Montreuil : Pearson, cop, 2012.

[6] : BOUROUIS Abderrahim, thèse « INTELLIGENT MOBILE HEALTH MONITORING SYSTEMS ».Université de Telmcen , décembre 2013.

[7]:Florent Garin, Développer des applications mobiles pour les Google Phones, Dunod, 21 octobre 2009.

[8] : Philippe Sultan,Asterisk la téléphonie d'entreprise libre,édition EYROLLES, 2010.

[9] : Jim Van Meggelen& Leif Madsen & Jared Smith, Asterisk the future of Telephony 2eme édition, Edition O'REILLY, 2007.

[10] :http://technet.microsoft.com/fr-fr/library/aa998265%28v=exchg.150%29.aspx

[11]:https://wiki.asterisk.org/wiki/display/AST/Asterisk+Versions

[12]: Myank Sharma, Openfire Administration, Edition PACKT, 2008

[13]: http://xmpp.org/

[14] :http://slideme.org

[15] :http://www.tegrazone.com

[16] : https://accounts.google.com

[17]: Grady B James R, « guide de l'utilisateur UML », Edition Eyrolles, 1997.

[18] : Pascal Roques, « UML par la pratique » édition Eyrolles 2003.

[19] : http://www.ietf.org/rfc/rfc2778.txt

[20] : http://www.ietf.org/rfc/rfc2779.txt

[21] : http://www.ietf.org/rfc/rfc2026.txt

[22] : http://www.ietf.org/rfc/rfc3920.txt

[23] : http://www.ietf.org/rfc/rfc3921.txt

[24] : http://www.ietf.org/rfc/rfc3922.txt

[25] : http://www.ietf.org/rfc/rfc3923.txt

[26] : http://xmpp.org/xmpp-protocols/rfcs/

[27] : http://dev.mysql.com/downloads/file.php?id=453071

[28] :http://www.igniterealtime.org/downloads/

[29] :ftp://ftp-developpez.com/florentgarin/android/smack.jar

[30] :https://www.ietf.org/rfc/rfc3435.txt

[31] : http://www.ietf.org/rfc/rfc4622.txt