République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Djilali BOUNAAMA de Khemis Miliana

Faculté des Sciences et de la Technologie

Département de Mathématiques et d'Informatique

Mémoire présenté

Pour l'obtention du diplôme de

Licence en Informatique

Spécialité : Systèmes Informatiques

Thème :

Réalisation d'un système de messagerie instantanée pour les smartphones (Android).

Réalisé par :

IRADUKUNDA Valentin

MIEN Ange Habathé Yannick

Encadré par :

Mme. BOUDALI

Année Universitaire 2017 - 2018

Dédicaces

Je dédie ce travail :

Tout d'abord à Ma famille parce que c'est eux ma motivation, ma force, sans eux je ne serai pas là où je suis maintenant.

A ma Mère : Maman ta force et tes sacrifices m'accompagnent dans tout ce que je fais.

A mon Père : Papa je me nourris de tes conseilles chaque jours.

A ma sœur préférée Vanessa : tu es la personne qui me comprend le plus, il n'y a pas Vanessa sans Valentin.

A mes grands frères Fabrice et Gustave : vous êtes toujours là pour moi avec vous ma vie parait facile.

A ma sœur Ingrid, je t'aime beaucoup

A mes petits frères et sœurs Joshua, Caleb, Milkah et Tehirah vous êtes ma force, je vous souhaite d'accomplir ce qu'un homme ne peut pas rêver.

A mes deux compatriotes avec lesquels j'ai passé trois ans merveilleux Gilbert et Daniel et Mon ami Burundais Aimé

Et enfin à Ange Yannick Mien mon Binôme.

IRADUKUNDA Valentin

Dédicaces

Je dédie ce travail :

A mes chers parents qui ne cessent de contribuer à ma réussite et de m'encourager.

La personne que je suis aujourd'hui c'est grâce à votre amour et vos innombrables sacrifices. Je vous aime.

Ma tendre mère qui par ses conseils m'a permis de faire de mes études une priorité.

A mes frères : Mathurin, Léon et mon petit frère Elisé. Vous êtes une force pour moi et l'amour qui nous lie est un réconfort.

A mes amis Flora Tarbangdo un trésor que la vie m'a accordé, Banaon Caleb et Diasso Hemann mes meilleurs amis. Je vous porte tous dans mon cœur ainsi que dans mes pensées.

De là mes pensées vont à Yasmine Gueddoun une amie de valeur, aimable et avec qui j'ai passé de merveilleux moments durant ces dernières années en Algérie.

A Eddy, un ami, un frère avec qui je partage ma passion pour l'informatique.

A tous mes amis qui m'ont toujours encouragé, Daniel, Isaac, à mon binôme Valentin Iradukunda.

Et à tous ceux qui m'ont soutenu de près ou de loin.

MIEN Ange Habathé Yannick

Remerciements

Tout d'abord nous remercions le bon Dieu pour son aide dans l'accomplissement de ce travail.

Cette œuvre n'aurait pas pu être possible sans le soutien inconditionnel de nombreuses personnes, que nous tenons à remercier :

Tout d'abord nos pensées et nos remerciements vont à nos parents et nos familles qui de loin ne cessent de nous soutenir. A tous nos amis qui d'une manière ou d'une autres on apporter leur plus pour que ce projet puisse voir le jour.

Nous remercions Mme. BOUDALI notre encadreur pour ses conseils, ses directives, ses remarques constructives, sa disponibilité et pour son encadrement.

Enfin nos remerciements à toute l'équipe pédagogique du département de Math-Informatique de l'Université de khemis Miliana, à tout l'ensemble de nos professeurs qui ont participé à notre formation tout le long de ce cursus de Licence, notamment à Mr Haniche pour sa sympathie et sa disponibilité.

IRADUKUNDA Valentin et MIEN Ange Habathé Yannick

Liste des figures

Liste des Tableaux

Liste des Abréviations

2D	**2 D**imensions
3D	**3 D**imensions
API	**A**pplication **P**rogramming **I**nterface
CSS	**C**ascading **S**tyle **S**heets**.**
GPS	**G**lobal **P**ositioning **S**ystem
HTML	**H**yper**T**ext **M**arkup **L**anguage
IRC	**I**nternet **R**elay **C**hat
JVM	**J**ava **V**irtual **M**achine
OHA	**O**pen **H**andset **A**lliance
OS	**O**perating **S**ystem
PDF	**P**ortable **D**ocument **F**ormat
SDK	**S**oftware **D**evelopment **K**it
SMS	**S**hort **M**essage **S**ervice
XML	e**X**tensible **M**ark-up **L**anguage
XMPP	e**X**tensible **M**essaging and **P**resence **P**rotocol

Sommaire

Résumé :

La disponibilité croissante des smartphones représente un atout considérable pour les usagers comme pour les développeurs d'applications. Le secret de ces smartphones réside dans les systèmes d'exploitation qu'ils utilisent.

Notre travail consiste principalement à concevoir une application de messagerie instantanée pour smartphones notamment pour Android. L'application a été développée en utilisant différentes technologies et Outils de développement, parmi lesquelles nous pouvons citer : Android Studio qui est l'environnement de développement recommandé pour la création d'application Android utilisant principalement le langage Java et le Kotlin. Le langage UML pour réaliser l'étude conceptuelle du système, la librairie SMACK pour l'implémentation du protocole XMPP et le langage XML.

Mots clés : Android, Application Mobiles, Messageries instantanée, XMPP.

Abstract:

The always growing smartphone availability represents a huge considerable asset for not only users but even for software developers. The secret of smartphones resides in their operating systems.

Our work consists principally of developing an instant messaging smartphone application that runs on android. We developed the application using the following technologies and development tools: Android studio which is the recommended software development environment for Android application creation which uses principally the Java programming language and Kotlin, UML language for the realization of the system's Conceptual study, the SMACK library which is an implementation of the XMPP protocol and then the XML language.

Key words: Android, Mobile applications, Instant Messaging, XMPP.

Introduction générale

Partie intégrante de notre quotidien, les téléphones mobiles subissent une avancée considérable dans le monde d'aujourd'hui. Le Smartphone est devenue un objet quasi-indispensable en permettant à quiconque de pouvoir contacter n'importe qui et n'importe où. Il génère des libertés et facilite certaines tâches chez ses utilisateurs. Ces différentes tâches sont possibles et évoluent de jour en jour grâce aux applications mobiles : à savoir les applications de loisirs aux applications de communication, voir courrier électronique et messagerie instantanée en passant par les applications d'apprentissages qui facilitent notre quotidien. Ainsi smartphone et application mobile vont de pair et ses appareils aux ressources limitées ne cessent d'être de plus en plus performants.

Notre projet s'inscrit dans un cadre général du développement d'applications mobile, notamment le développement d'une application de messagerie instantanée pour smartphone précisément tournant sous Android. Application permettant de manière générale à plusieurs utilisateurs d'échanger des messages textes, des émoticônes des fichiers en quasi temps réel.

Dans quelle mesure pouvons-nous assurer une communication en temps réel sous Android entre deux utilisateurs et quelles sont les outils disponibles pour cette réalisation ?

Tout au long de ce rapport il sera question de voir comment cette réalisation peut être mise en place. Ainsi notre mémoire est structuré en cinq chapitres : le premier chapitre porte sur le smartphone et les applications mobiles en général.

Le deuxième chapitre est dédié au système d'exploitation mobile Android. Il sera question dans ce chapitre de faire une étude détaillée d'Android.

Dans le chapitre trois nous ferons une étude du concept de messagerie instantanée, comprendre son fonctionnement, les protocoles utilisés et procéder par une étude de l'existant en présentant les applications mobiles de messagerie instantanée les plus populaires.

Le chapitre quatre sera consacré à la conception qui est une étude en amont du projet permettant de décrire les besoins de notre système par le biais d'une modélisation UML.

La réalisation de notre application fera l'objet du cinquième chapitre où nous présenterons les outils de développement, les langages utilisées et où nous ferons une présentation de notre application. Nous conclurons notre mémoire par exposer l'ensemble des connaissances acquises au cours de la réalisation de notre projet et nous exposerons quelques perspectives.

Chapitre 1 : Les smartphones

1- Introduction

Le smartphone est passé en l'espace de quelques années d'un outil professionnel à un objet de divertissement grand public, emportant avec lui un secteur mobile en pleine reconversion. Ces nouveaux téléphones et tablettes sont généralement tactiles, bien que certains modèles proposent encore un clavier et ils fonctionnent avec des systèmes d'exploitation (OS) nouveaux désormais partie intégrante de notre quotidien.

2- Définition

Le smartphone ou « téléphone intelligent » désigne un téléphone mobile doté de fonctionnalités évoluées qui s'apparentent à celles d'un ordinateur, à savoir : navigation sur Internet, lecture de vidéos, de musique, jeux vidéo, courrier électronique, vidéoconférence, bureautique légère, etc.

Muni d'un processeur puissant, souvent multi-cœur, il embarque une série de capteurs (boussole, accéléromètre[1], gyroscope[2], GPS) qui lui permettent de faire fonctionner des applications dédiées à l'activité physique, de navigation assistée ainsi que des jeux que l'on peut contrôler d'un simple mouvement. Les smartphones sont généralement dotés d'un appareil photo-vidéo et d'une caméra frontale dont les performances ne cessent de progresser.

Pour pouvoir réaliser toutes ses fonctions, que ça soit envoyer des messages, faire des appels, utiliser le GPS etc., un smartphone a besoin d'un système d'exploitation pour contrôler l'utilisation des différentes ressources et contrôler le fonctionnement générales des applications installées.

[1] *L'**accéléromètre** est un appareil qui permet de mesurer l'accélération subie par un mouvement*

[2] *Le **gyroscope** est un appareil qui sert à indiquer une direction invariable.*

3- Les systèmes d'exploitation (OS) pour smartphones

Il existe plusieurs systèmes d'exploitation dédiés aux smartphones :

- **Android** : Android est le système d'exploitation le plus répandu de nos jours (2018). Développé par Google, il n'est pas rattaché à une marque de smartphones (contrairement à iOS). Android est Open Source utilisant le noyau Linux. On retrouve des télévisions, des montres, des fours à micro-ondes et pleins d'autres objets qui fonctionnent à l'aide de ce système.

- **iOS de la compagnie Apple** : Après Android, iOS est le deuxième système d'exploitation le plus répandu. Il équipe uniquement les « iPhone », c'est-à-dire les smartphones de la marque Apple.

- **BlackBerry OS** : BlackBerry OS est un système d'exploitation propriétaire pour téléphone mobile de la gamme BlackBerry, conçu par la société canadienne Research In Motion, maintenant connue sous le nom de BlackBerry.

- **Symbian OS** : Symbian OS est un système d'exploitation pour téléphones mobiles et assistants personnels conçu par Symbian ltd (fondée d'un partenariat de la société mère PSION avec Nokia, Ericsson, Motorola et Matsushita). Il est acheté en 2008 à 100 % par Nokia. En 2013 l'équipe de Symbian OS annonce l'arrêt du système d'exploitation. Il est récemment passé en open source.

D'autres systèmes d'exploitation existaient aussi comme :

- MeeGo, développé par Intel et Nokia
- Bada, développé par Samsung, abandonné en 2013
- WebOS, développé par Palm, puis HP, puis LG

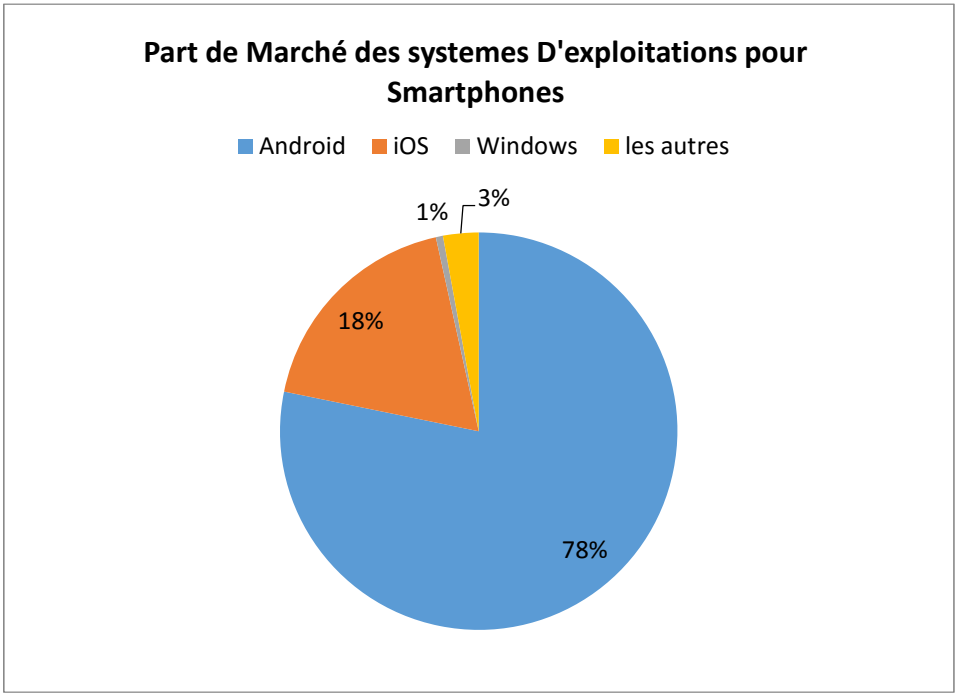


Figure 1 Part de marché des smartphones par OS, année 2017
d'après le site Stat counter global stats

4- Les Applications Mobiles

Une application mobile est un logiciel développé pour un appareil électronique mobile, tel qu'un assistant personnel, un téléphone portable, un smartphone, un baladeur numérique, une tablette tactile, etc.

Il existe de nombreuses applications mobiles pour :

- Les jeux : exemple Angry Birds, Candy Crush
- Le GPS et les services permettant la localisation : exemple Google Maps
- Les suivis des commandes, l'achat de billets ;
- Des applications médicales mobiles ;
- La réalité virtuelle ;
- L'écoute de musiques ou de radios ;
- La visualisation de vidéos ou de chaines de télévision ;
- La consultation d'Internet ; exemple Firefox, Opéra Mini
- Les réseaux sociaux généraux (type Facebook) ;
- La messagerie instantanée : exemple Facebook Messenger, Whatsapp, Skype, Telegram

Il existe différents types d'applications mobiles : [1]

- **Applications Natives** : Les applications natives sont celles qui sont développées avec le langage de programmation native propre au Système. (Par exemple le langage Java, et Kotlin pour Android).

- **Applications Hybrides** : Ce sont des applications utilisant le navigateur web intégré du support (Smartphone, tablette) et les technologies du web (HTML, CSS et JavaScript) pour fonctionner sur différent OS (Android, iOS, Windows Phone, etc.)

5- Conclusion

Dans ce chapitre, nous avons présenté le smartphone et fait un tour d'horizon de certains systèmes d'exploitation qui rendent ces smartphones de plus en plus intelligents. Egalement nous avons donné un petit aperçu sur les applications mobiles. Dans le chapitre qui suit, nous ferons une présentation plus détaillée du Système Android.

Chapitre 2 : Le Système d'exploitation (OS) Android

1. Introduction

Plusieurs Systèmes d'exploitation pour smartphones existent. Mais dans cette partie nous portons notre attention sur le système Android, ses avantages, ses fonctionnalités ainsi que son architecture.

2. Présentation d'Android

Android est un OS (Operating System ou Système d'exploitation en français) Open Source pensé pour les téléphones mobiles et développé par l'Open Handset Alliance (OHA) sous l'autorité de Google. L'OHA est composé d'environ 80 sociétés telles que Samsung, HTC, SFR, Orange, Asus, Qualcom etc. Android est basé sur un kernel (noyau) Linux, et aussi est un Framework[3] et en tant que développeur, il y a un accès libre au SDK (Software Development Kit – Kit de développement) et à tout le code source de la plateforme. Ce qui permet de comprendre son fonctionnement, de créer des versions personnalisées de l'OS et bien sûr, de développer des applications. Par conséquent, plusieurs appareils exotiques tournant sous Android ont vu le jour : autoradio, montre, télévision, etc.

La première version du SDK Android 1.0 sort en 2008 avec le premier téléphone sous Android (HTC Dream).

En Avril 2009, la version 1.5 (API 3) d'Android sort. Cette version baptisée Cupcake (Petit gâteau) inaugure les nouveaux noms des versions d'Android.

Ce qui donnera pour les futures versions : [2]

- **Donut** (Beignet) 1.6 (API 4) : septembre 2009
- **Eclair** (Eclair) 2.0/2.2 (API 7) : octobre 2009
- **Froyo** (Yaourt glacé) 2.2.x (API 8) : mais 2010
- **Gingerbead** (Pain d'épice) 2.3.x (API 10) : décembre 2010
- **Honeycomb** 2.3.x (API 11-12-13) : février 2011

[3] *Un **framework** est un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel, ou d'une application. Et peut être composé de plusieurs bibliothèques chacune spécialisée dans un domaine.*

- **Ice Cream Sandwich** (Sandwich à la crème glacée) 4.0.x (API 14-18)
- **Jelly Bean** (Dragibus) 4. (1-3).x (API 16-17-18) : juin 2012
- **KitKat** 4.4.x : novembre 2013
- **Lollipop** 5.x novembre 2014
- **Marshmallow** 6.x : courant du troisième trimestre 2015
- **Nougat** 7.x
- **Oreo** 8.x version sortie le 21 août 2017

Le personnage représentant Android est nommé « Bugdroid ».

Figure 2 Image de Bugdroid [10]

3. Architecture d'Android

L'architecture d'Android se compose de cinq parties distinctes : [7]

- **Application** : représente l'ensemble des applications fournies avec Android (e-mail, SMS, calendrier, etc.)
- **Framework Android** : représente le Framework permettant aux développeurs de créer des applications en accédant à l'ensemble des APIs et fonctionnalités disponibles sur le téléphone (fournisseur de contenu, gestionnaire de ressources, gestionnaire de notification, gestionnaire d'activité, etc.)
- **Bibliothèque** : Android dispose d'un ensemble de bibliothèques utilisées par les différents composants du système pour de nombreuses fonctions telles que : Les rendus graphiques (Open GL), le stockage de données (SQLite[4]), la navigation web (WebKit), etc.

[4] ***SQLite*** *est une BDD relationnelle légère et open source supportée nativement par Android*

- **Android Runtime** : contient entre autres la JVM Dalvik. Android s'appuie sur une machine virtuelle particulière que l'on appelle Dalvik. Mais à partir de la version 5.0 (Lollipop) ART (Android Run Time) remplace la machine virtuelle Dalvik.
- **Linux Kernel** : le noyau linux fournissant une interface avec le matériel, gérant la mémoire, les ressources et les processus Android.

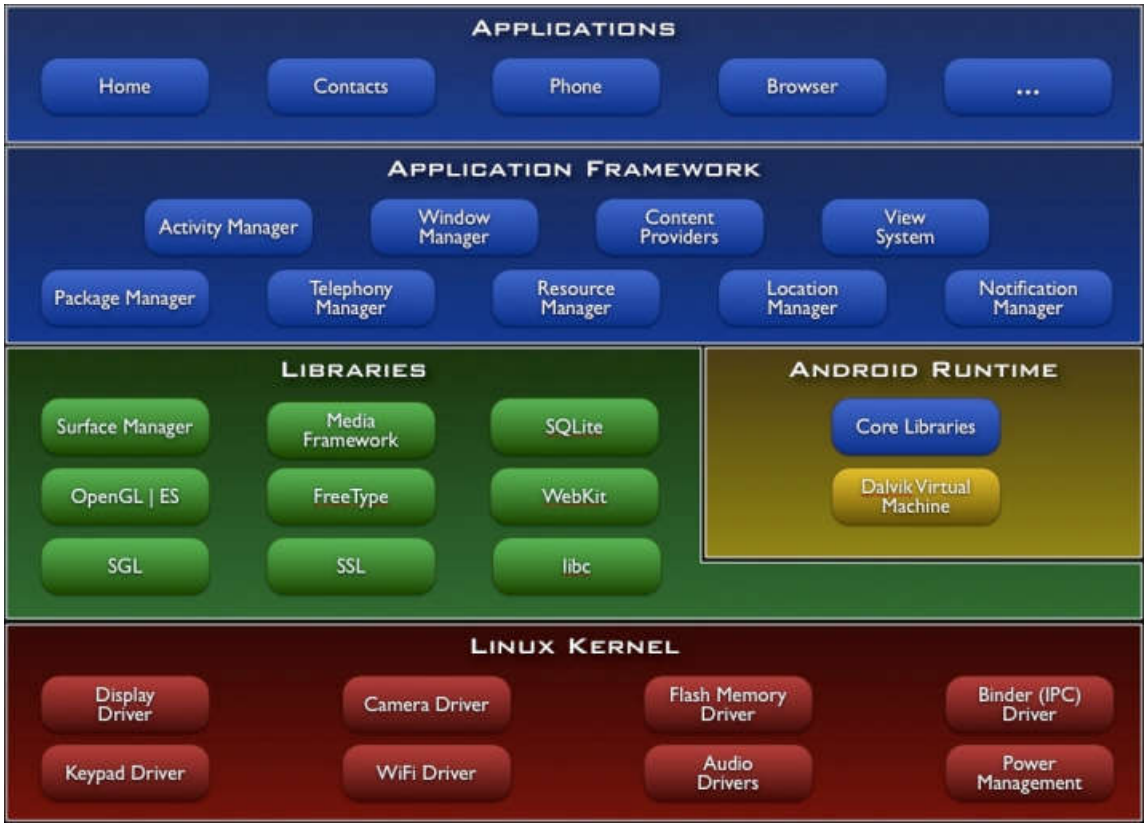


Figure 3 Architecture de l'OS Android [7]

4- Les Avantages d'Android

La plateforme Android présente plusieurs atouts : [7]

- **Open Source :** Le contrat de licence pour Android respecte les principes de l'open source, c'est-à-dire que les codes sources peuvent être téléchargées et modifiées. Notez au passage qu'Android utilise des bibliothèques open source puissantes, comme par exemple SQLite pour les bases de données et OpenGL pour la gestion d'images 2D et 3D (pour les jeux).

- **Facile à Développer :** Toutes les APIs mises à disposition facilitent et accélèrent grandement le travail. Ces APIs sont très complètes et très faciles d'accès.

- **Facile à vendre :** Le Play Store (anciennement appelé Android Market) est une plateforme immense et très visitée ; c'est donc une mine d'opportunités pour quiconque veut diffuser une application dessus.

- **Flexible** : Le système est extrêmement portable, il s'adapte à beaucoup de structures différentes (les smartphones, les tablettes, la présence ou l'absence de clavier). Non seulement c'est une immense chance d'avoir autant d'opportunités, mais en plus Android est construit de manière à faciliter le développement et la distribution en fonction des composants en présence dans le terminal (si une application nécessite d'utiliser le Bluetooth, seuls les terminaux équipés de Bluetooth pourront la voir sur le Play Store).

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

5- Les composantes Android

Le Framework Android se compose de quelques briques essentielles pour la création d'applications. Dans cette partie, nous allons expliquer ces différents composants.

5.1- Activité (Activity)

Une activité est la composante principale d'une application Android. Elle représente l'implémentation et les interactions de vos interfaces. [2]

5.2- Fragment

Il permet de construire des interfaces utilisateurs plus souples (smartphone/tablette), plus dynamiques et plus faciles à utiliser. Un fragment peut être considéré comme une portion d'interface. Donc une interface (activité) peut être constituée de plusieurs fragments. [2]

5.3- Les services

Un service à la différence d'une activité, ne possède pas d'interface mais permet l'exécution d'un traitement en tâche de fond (opération longue ou un appel distant). Un service ne s'arrêtera pas tant qu'il n'est pas interrompu ou terminé. [2]

5.4- Récepteur d'évènement (Broadcast receiver)

C'est une composante qui réagit à un évènement système. Il ne peut pas effectuer des modifications sur vos interfaces, il peut seulement afficher une notification, lancer une activité ou un service, etc. [2]

5.5- Fournisseur de contenu (Content provider)

Il permet de partager les données d'une application. Ces données peuvent être stockées en base SQLite, dans les fichiers ou sur le web. Le but étant de permettre à d'autres applications de requêter ces données.

Le système Android propose des contents providers (présent par défaut) qui peuvent être utilisé par vos applications : Contact, Agenda, Média etc. [2]

5.6- Intentions (Intent)

Les composants Android (Activité, service et Broadcast receiver) communiquent via des messages système que l'on appelle *« Intent ».* Il existe deux types d'Intents

- **Explicite** : appeler, par exemple, une activité connue pour exécuter une action particulière.
- **Implicite** : demander au système quelle activité peut exécuter une action spécifique (par exemple, ouvrir un fichier PDF). [2]

6- Cycle de vie d'une activité

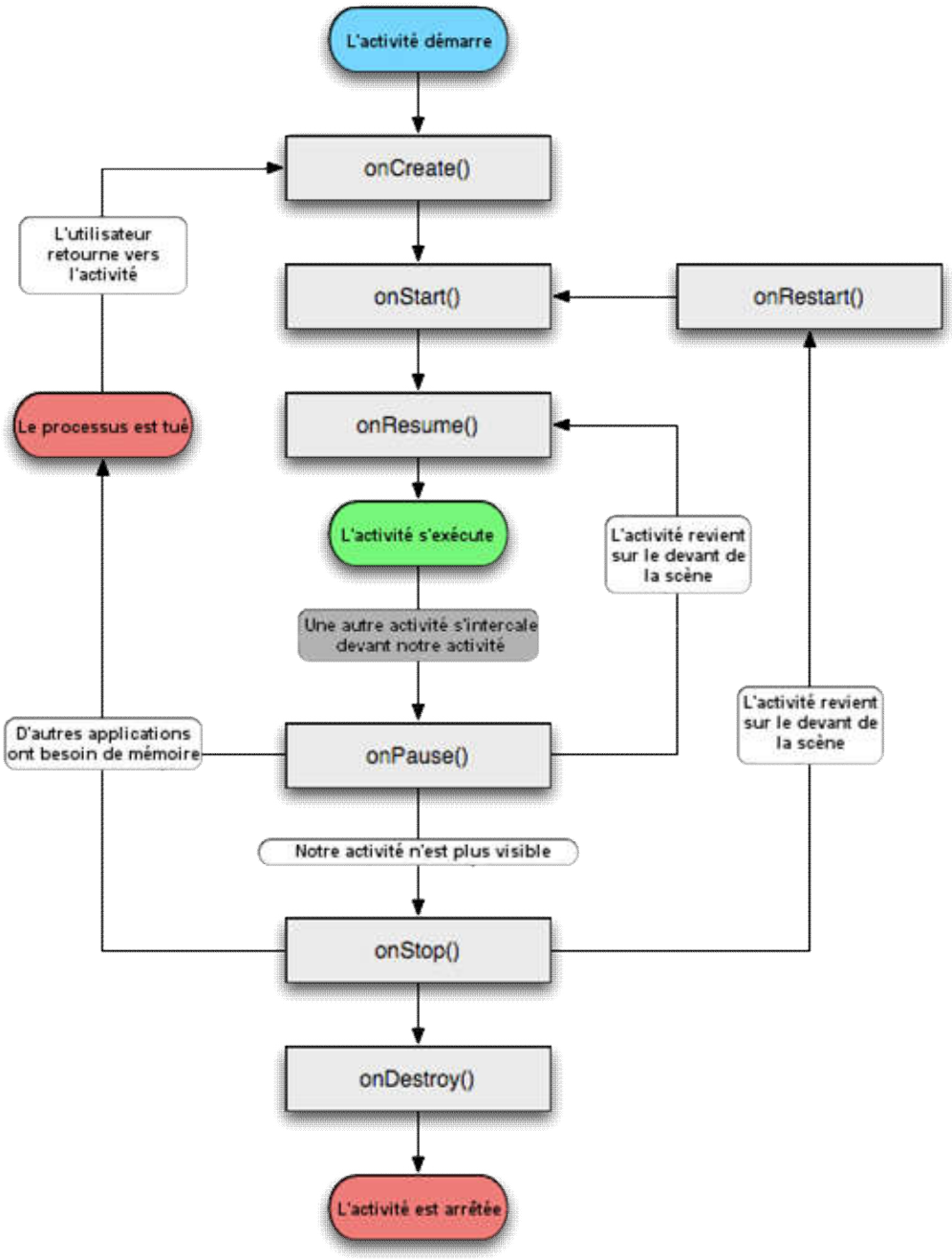


Figure 4 Cycle de vie d'une Activité Android [7]

7- Conclusion

Il ressort de ce chapitre dans lequel nous avons fait une présentation d'Android, sa description, ses avantages, ses fonctionnalités et son architecture, qu'Android est un système d'exploitation bien structuré, et flexible, qui se base sur la réutilisation des éléments et qui facilite le développement des applications. Comprendre le fonctionnement d'Android est important car c'est sur ce système que notre application doit être déployée. Dans le chapitre suivant nous explorerons le monde de la messagerie instantanée.

Chapitre 3 : La messagerie Instantanée

1- Introduction

De nos jours, plusieurs services que l'on peut qualifier d'artisanaux se sont trouvés automatisés et informatisés par l'avènement de l'informatique et de la télécommunication. Cela dans un but de gain en temps, d'argent et également pour le prestige de l'évolution. Parmi cette panoplie de services se trouve la messagerie, et la forme la plus utilisé de ce service aujourd'hui est la messagerie instantanée.

Dans ce chapitre nous sillonnons le concept de messagerie instantanée, passant par sa définition, son historique et également son fonctionnement et les protocoles utilisés. Une étude des applications de messagerie instantanée les plus utilisés aujourd'hui, et un aperçu sur l'aspect sécurité de ce service est inclus évidemment.

2- Définitions

La messagerie instantanée souvent appelé « Chat » est une méthode de communication qui permet l'échange instantané de messages textuels, vocaux, de fichiers en mode interactif entre plusieurs personnes par l'intermédiaire d'ordinateurs, de téléphones connectés au même réseau informatique, et plus communément celui d'internet. [4]

3- Historique

La messagerie instantanée existe depuis **1982.** Lancée par une compagnie de presse appelés « **les derniers nouvelles d'Alsaces** (DNA) ». L'idée fut abandonnée à cause d'un piratage informatique.

- **En 1988** un développeur Finlandais appelé **Jarkko Oirkarinen** développe un protocole[5] de messagerie instantanée appelé **IRC** (Internet Relay Chat)
- **En 1988** un protocole ouvert **Zephyr** fut aussi développée au MIT

[5] *Dans les réseaux informatiques et les télécommunications, un protocole de communication est une spécification de plusieurs règles pour un type de communication particulier*

- Les grands progrès modernes fut réaliser en **1996** par une entreprise israélienne, Mirabilis qui développe un logiciel de messagerie instantanée appelé ICQ (I seek You), l'innovation était la gestion d'une liste de contacts personnels.
- **En 1998** un autre protocole de communication standard et ouvert appelé **Jabber** fut créé.
- **En 2004** le protocole **Jabber** dit **XMPP** fut normalisé par **IETF** [6] (Internet Engineering Task Force).
- **En 2006** les conventions de nommage pour les identifiants de messagerie instantanée « **IRI/URI scheme** »[7] sont adoptées par l'IETF : elles sont basées sur le protocole **Jabber**.

4- Les protocoles de communications

Pour développer une application de messagerie instantanée on a besoin d'établir un protocole de communication. Les protocoles les plus connue ou disons les plus utilisés sont Jabber/XMPP, IRC et SIMPLE.

4.1- Le protocole XMPP

Créé en 1998 par **Jeremy Miller** le XMPP : eXtensible Messaging and Presence Protocol (ou Protocole extensible de messagerie et de présence en français) est un protocole standard de l'IETF basé sur XML. Issu du projet open source Jabber, ce protocole promet une grande interopérabilité entre les différentes messageries instantanées. Il permet l'échange en temps quasi-réel de données structurées (messages, fichiers). [5]

La particularité de cette technologie réside en 3 points :

- Elle permet d'avoir des adresses de même type que l'email :

  Afin de permettre la communication entre toutes les entités constituant une architecture Jabber, des **"Jabber ID"** sont utilisés. Un Jabber ID (ou JID) est composé de 2 à 3 parties à savoir :

[6] ***IETF:*** *un groupe informel d'individu ouvert à tous qui définit des standards d'internet.*

[7] ***IRI/URI:*** *identifiant uniforme des ressources sur internet pour respecter les normes*

- 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 est utilisés lorsque l'utilisateur se connecte de plusieurs endroit ou avec plusieurs clients Jabber différents.

Exemple : *utilisateur@serveur/ressource*

❖ Elle est entièrement basée sur XML

Lors d'une connexion client / serveur, 2 flux XML unidirectionnels sont créés: l'un du client vers le serveur, l'autre du serveur vers le client. Ainsi, la communication entre ce client et ce serveur Jabber repose exclusivement sur ces deux flux. Les données au format XML échangées durant la communication sont appelées **« stanzas»**.

Il existe 3 paquets de base :

- **Présence :** Pour indiquer le statut du client considéré, en l'occurrence du client « A ». L'envoi de ce paquet est automatiquement effectué par le client vers le serveur.

```
<presence
    from="clientA@welldone.org/travail"
    to="clientB@Fritalk.com/maison">

    <Show>away</show>
    <status>absent</status>
    <priority>5</priority>

</presence>
```

Figure 5 « Stanza » de présence XMPP [14]

On remarque la présence d'une notion de "priorité". Cette dernière, comprise entre -128 et +127, caractérise une ressource. La ressource qui a la plus haute priorité est considérée comme la ressource par défaut. Lorsque 2 ressources ont la même priorité, le message est envoyé à la dernière ressource connectée.

- **Message :** Lorsque le client A communique avec le client B.

```
<message
    from='clientA@jabber.org' to='clientB@Fritalk.com'>

    <body>Comment vas-tu?</body>

</message>
```

Figure 6 « Stanza » de message XMPP [14]

- **IQ :** (Info/Query) Ces paquets sont utilisés pour échanger des données entre les applications Jabber (authentification, récupération d'une liste de contact, ou encore découvrir, accéder et interagir aux services proposés par le serveur). Ces données sont exprimées dans la balise "query", dans l'exemple ci-dessous, il s'agit d'un utilisateur demandant à s'authentifier.

```
<iq
    type='set' id='auth'>

    <query xmlns='jabber:iq:auth'>
        <username>user</username>
        <password>azerty</password>
        <resource>TelnetClient</resource>
    </query>
</iq>
```

Figure 7 « Stanza » d'IQ XMPP [14]

Le serveur répond alors par :

```
<iq type='result' id='auth'/>
```

Figure 8 « Stanza » de réponse du serveur XMPP [14]

Si l'authentification a échoué, le serveur renverra un paquet de type **« error »**. En effet, aux types de paquets **« set »** (pour la mise à jour d'une donnée) et **« result »** (pour signifier que le message considéré est le résultat d'une requête précédente), s'ajoutent les messages de types : **« get »** pour effectuer une requête

- Le standard est ouvert et bien documenté.

  Dans la mesure où le protocole XMPP implémenté par l'IETF est ouvert, chacun est libre d'implémenter son propre protocole et de lui attribuer la licence de son choix, libre ou propriétaire. Cela a permis l'explosion de clients Jabber et contribuera très certainement à son expansion.

4.2- Le protocole IRC (Internet Relay Chat)

Le plus ancien des protocoles de messagerie instantanée : le protocole IRC a été développé en 1998 par **Jarkko Oirkarinen.** Il est basé sur le model réseaux Client/Serveur pour faciliter essentiellement la communication en groupe. La différence notable qui le différencie de XMPP est qu'il n'a pas l'aspect extensible pour pouvoir communiquer avec d'autre protocole de communication.

4.3- Le protocole SIMPLE

SIMPLE, qui signifie Protocole d'initialisation de session pour les extensions de messagerie instantanée et de levier de présence, est un protocole de présence et de messagerie instantanée standard ouvert. Comme son nom l'indique, SIMPLE est basé sur le célèbre protocole SIP (Session Initiation Protocol).

Le protocole SIP (Session Initiation Protocol) est un protocole de contrôle de la couche application qui permet d'établir, de modifier et de terminer des sessions multimédia telles que les appels de téléphonie Internet. Les mécanismes fournis par SIP sont plus utiles pour les applications de présence que pour la messagerie instantanée. Pour la messagerie instantanée, SIP définit deux modes, à savoir le mode page (qui envoie des messages sans établir de session) et le mode session (qui commence par l'établissement d'une session avant l'envoi de messages instantanés). L'IETF s'est efforcé de développer un protocole de messagerie instantanée standard basé sur SIP qui répond aux exigences du groupe de travail IMPP (Instant Messaging and Presence Protocol), et par conséquent, ils se sont retrouvés avec SIMPLE. [5]

5- Fonctionnement de la messagerie instantanée

La messagerie instantanée requiert l'emploi d'un logiciel client qui se connecte à un serveur de messagerie instantanée. Le client se connecte au serveur qui contient les informations sur tous les utilisateurs inscrits, connectés ou non. Deux personnes peuvent communiquer en direct si elles sont simultanément connectées au serveur. Sinon, elles ont la possibilité de consulter leurs messages dans leur boîte aux lettres au moment où elles se connectent. Comme logiciel/application client et serveur nous avons par exemple des logiciels Client/Serveur IRC, et Client/Serveur Jabber/XMPP.

5.1- Le modèle Client Serveur

L'environnement client-serveur désigne un mode de communication à travers un réseau entre plusieurs programmes : l'un, qualifié de client, envoie des requêtes ; l'autre ou les autres, qualifiés de serveurs, attendent les requêtes des clients et y répondent. Par extension, le client désigne également l'ordinateur ou la machine virtuelle sur lequel est exécuté le logiciel client, et le serveur, l'ordinateur ou la machine virtuelle sur lequel est exécuté le logiciel serveur.

Un serveur offre des services, à un ou plusieurs clients (parfois des milliers). Les services les plus courants sont :

- la messagerie instantanée
- l'accès aux informations du World Wide Web ;
- le courrier électronique ;
- le partage d'imprimantes ;
- le commerce électronique ;
- le stockage en base de données ;
- la gestion de l'authentification et du contrôle d'accès.

Un serveur fonctionne en permanence, répondant automatiquement à des requêtes provenant d'autres dispositifs informatiques (les clients), selon le principe dit client-serveur.

Le mode Pair-à-pair (anglais peer-to-peer ou P2P) est l'opposé du mode client/serveur

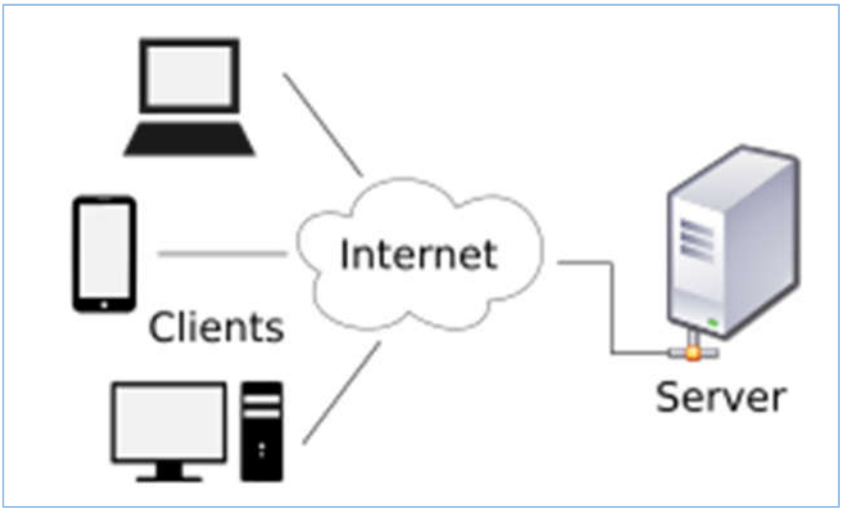


Figure 9 Modèle Client – Serveur

5.2- Client IRC et Serveur IRC

Un client IRC implémente la partie client du protocole IRC et se connecte à un serveur IRC qui est un serveur sur lequel est installé un programme (appelé IRCD) permettant aux utilisateurs connectés de discuter en temps réel par le biais du protocole IRC.

Quelques IRCD les plus utilisés :

- Bahamut développé par le réseau de discussion DALnet
- IRCu développé par le réseau de discussion Undernet
- UnrealIRCd IRCD fonctionnant sous Windows et Unix
- Hybrid (IRCd) pour héberger son propre serveur de discussions, librement.

5.3- Client XMPP et Serveur XMPP

Un client XMPP est un logiciel/application de messagerie instantanée qui utilise le protocole XMPP pour se connecter à un serveur XMPP qui est un logiciel serveur qui implémente le protocole de communication XMPP.

Avec le XMPP 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.

La flexibilité est une composante cruciale du protocole XMPP. Ainsi, sur un serveur Jabber, 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 Jabber de base présente 3 fonctionnalités :

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

Il existe deux catégories de serveur XMPP :

Les serveurs libres : Comme exemple de serveurs nous avons :

- **Ejabberd** : qui utilise le langage Erlang.
- **Openfire** : utilise le langage Java
- **Tigase** : qui utilise le langage Java.
- **DJabberd** : utilise le langage Perl

Les serveurs propriétaires : Comme exemple nous avons

- Apple Messages Server (anciennement iChat Server)
- Jabber XCP
- Sun Java System Instant Messaging Server

6- La sécurité dans la messagerie instantanée

Les développeurs de logiciels se concentrent généralement sur les fonctionnalités qui attirent les utilisateurs finaux plutôt que sur les problèmes de sécurité lors de la conception et de l'implémentation de leurs produits logiciels. L'industrie du logiciel est en proie à de telles pratiques qui traitent la sécurité comme une réflexion après coup lors de la planification et de la production d'un nouveau produit. Avec la prévalence de la messagerie instantanée et les millions d'utilisateurs bénéficiant de ses services, il est très probable que les attaquants et les créateurs de programmes malveillants profiteront de la situation et exploiteront les vulnérabilités des systèmes de messagerie instantanée pour infecter un large secteur de la communauté Internet. Dans cette section, nous discutons de certaines fonctions de sécurité dans les systèmes de messagerie instantanée et mettons en lumière les menaces qui les entourent. [5]

6.1- L'authentification et mécanisme d'authentification

L'authentification est définie comme le processus de vérification d'une identité revendiquée par ou pour une entité système. Dans le contexte de la messagerie instantanée, une entité de système pourrait être représentée par un utilisateur. Les applications de messagerie instantanée sont supposées fournir un mécanisme de sécurité pour authentifier les utilisateurs lorsqu'ils utilisent des clients de messagerie instantanée pour accéder aux serveurs. En outre, l'application de messagerie instantanée doit s'assurer que les utilisateurs sont autorisés à utiliser les ressources système appropriées. Par exemple, *Yannick* n'est pas censé accéder à la liste des contacts de *Valentin*. Le processus d'authentification est généralement basé sur un nom d'utilisateur et un mot de passe choisis par un utilisateur lors de son inscription pour la première fois. Le nom d'utilisateur est distribué par l'utilisateur à tous ses contacts, alors que le mot de passe reste confidentiel. [5].

Le mécanisme utilisé pour l'authentification est Single Sign-On (SSO), par lequel l'utilisateur s'authentifie une seule fois et est automatiquement connecté aux fournisseurs de services (serveur MI) selon les besoins sans nécessiter d'interaction manuelle. Une méthode d'authentification utilisée par MI est SASL (Simple Authentication and Security Layer) qui fournit une méthode généralisée d'ajout de la prise en charge de l'authentification aux protocoles basés sur la connexion et est publiée par l'IETF. SASL est pris en charge par XMPP pour l'authentification.

6.2- Gestion de la confidentialité

D'un autre côté, il est important d'assurer la confidentialité des messages, car une grande partie de la messagerie instantanée se produit sur des réseaux publics et non fiables tels qu'internet.

- **Secure Socket Layer (SSL) :** est un protocole de gestion de la sécurité d'une transmission de messages sur Internet. Il a été développé par Netscape Communications Corporation. Plus tard, l'IETF a utilisé la version 3.0 de SSL pour développer un protocole standard, qui est devenu le protocole TLS (Transport Layer Security). Le protocole TLS permet aux applications client / serveur de communiquer de manière à éviter l'écoute indiscrète, la falsification de messages. Il fonctionne sur la couche de transport pour authentifier les points de terminaison et chiffrer les messages entre les entités authentifiées. XMPP utilise TLS pour chiffrer et sécuriser le flux XML sur le réseau, et le protocole SIMPLE s'appuie également sur TLS.

- **ORT (Off-the-Record) :** c'est une fonction permettant de sécuriser la messagerie instantanée, elle permet aux clients d'avoir des conversations privées sur le réseau en fournissant des méthodes d'authentification et de cryptage robustes. En outre, OTR utilise des clés de chiffrement temporaires par message, ce qui rend impossible la compromission des conversations précédentes même si les pirates parviennent à capturer une clé actuellement utilisée.

- **CAPTCHA :** Une menace courante dans les environnements de messagerie instantanée est lorsque les pirates informatiques utilisent des programmes informatiques pour s'inscrire automatiquement pour des comptes illimités afin de consommer les ressources des serveurs de messagerie instantanée. Les serveurs de messagerie instantanée surmontent cette menace en utilisant la technique connue sous le nom de test de Turing Public complètement automatisé pour raconter aux ordinateurs et aux humains (CAPTCHA). CAPTCHA est un test de réponse aux défis qui invite le créateur du compte de messagerie instantanée à entrer une chaîne de caractères requise en fonction d'une image affichée à l'écran. Habituellement, l'image est affichée d'une manière que seuls les humains peuvent reconnaître.

7- Utilisation de la messagerie instantanée

Les motivations à l'utilisation de la messagerie instantanée sont nombreuses :

- Permet de rester en contact en permanence ;
- Moins coûteux que les autres outils de communication;
- Plus rapide que le mail pour le transfert des fichiers lourds (y compris des photos) ;
- Discuter à plusieurs facilement ;
- Enrichir l'échange avec une communication multimédia (Webcam)
- Réaliser un travail collaboratif
- Echanger et partager des fichiers

8- Etude de l'existant sur la messagerie instantanée

Plusieurs applications de messagerie instantanée pour smartphone Android existe. Dans cette partie nous présenterons quelques une d'entre eux à savoir : WhatsApp Messenger, Facebook Messenger et Telegram.

8.1- WhatsApp Messenger pour Android

WhatsApp (ou WhatsApp Messenger) est une application mobile multiplateforme qui fournit un système de messagerie instantanée via Internet et via les réseaux mobiles. Ecrit en Erlang, elle est utilisée, en 2017, par plus d'un milliard de personnes quotidiennement

L'entreprise WhatsApp, fondée en 2009 par Jan Koum et Brian Acton, deux anciens ingénieurs de Yahoo est rachetée par Facebook en février 2014.

WhatsApp utilise une version personnalisée du protocole de communication XMPP, et se sert du numéro de téléphone comme identifiant unique et utilise un Serveur XMPP : Ejabberd. Depuis avril 2016, l'application a intégré le chiffrement de bout en bout pour toutes les communications. Elle utilise les numéros de téléphone de vos contacts, vous n'avez donc pas à ajouter des contacts manuellement. Vous pouvez envoyer des messages écrits, des messages

vocaux, des stickers, des emojis et des fichiers. WhatsApp vous permet également d'écrire à plusieurs contacts en même temps (groupes), ainsi que d'appeler gratuitement vos contacts (son et/ou vidéo). [8]

Figure 10 Page d'accueil de l'application WhatsApp [13]

8.2- Facebook Messenger sous Android

Ecrit en Erlang, PHP, JavaScript et C++, Facebook Messenger est un système de messagerie instantanée créé par la société Facebook lancé en 2012, et incorporé au réseau social Facebook.

Client XMPP, Messenger vous permet également de communiquer avec vos amis qui ne sont pas sur Facebook, il faut tout simplement utiliser leurs numéros de téléphone. Vous retrouverez sur l'application l'ensemble de votre messagerie Facebook sans avoir à passer par l'application ou le site internet. Parmi les nombreuses fonctionnalités possibles, nous trouvons les emojis, les groupes, les enregistrements vocaux, les données de géolocalisation, les appels gratuits, le partage de photos etc.

L'application est multiplateforme : disponible sous Android, iOS, Windows Phone

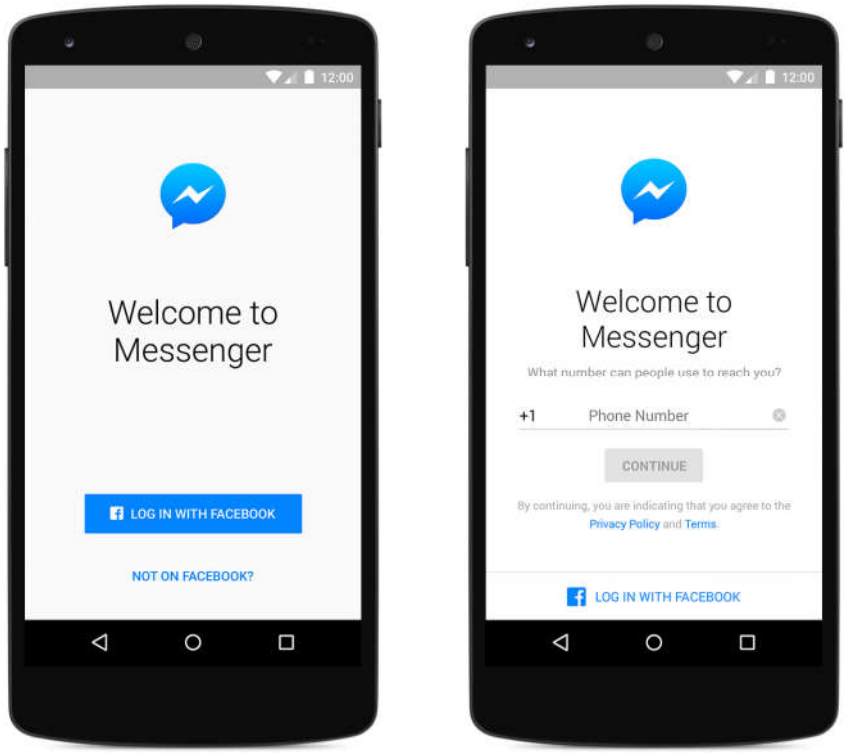


Figure 11 Page d'accueil de l'application Messenger [11]

8.3- Telegram sous Android

Telegram Messenger est une application de messagerie sécurisée hébergée sur le cloud. Créée en 2013 par les frères Nikolaï et Pavel Dourov, fondateurs de VKontakte, le réseau social dominant en Russie. L'application gratuite est disponible sur Android, iOS, Windows Phone ainsi que sur ordinateur (Windows, OS X et Linux). Les utilisateurs peuvent échanger des messages, photos, vidéos et documents sans limite de taille. Il est possible d'envoyer des messages chiffrés de bout en bout qui ne sont pas stocké sur les serveurs de Telegram. La partie cliente est open source tandis que la partie serveur reste propriétaire.

Telegram est une solution qui plaira à ceux qui veulent garder leurs messages privés. Elle est basée sur un chiffrement AES-256, mais demeure fermé et propriétaire. Les conversations sont chiffrées et vous pouvez même utiliser une option de destruction après lecture du message. L'interface est aussi simple et les fonctionnalités sont similaires à WhatsApp. Les options sont nombreuses et vous pouvez choisir la couleur de la LED de notification pour les messages Telegram. [8]

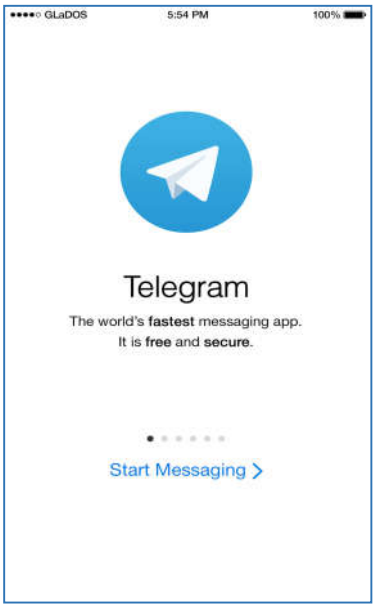


Figure 12 Page d'accueil de l'application Telegram [13]

9- Conclusion

Communiquer (Echange de messages, de fichiers etc.) de manière interactif en temps réel, entre plusieurs utilisateurs connectés au même réseau informatique, ou en deux mots « la messagerie instantanée », fait partis de notre quotidien. Dans ce chapitre nous avons eu à faire une étude de ce service, son historique, son fonctionnement et également une étude de l'existant, en présentant certaines applications qui existent. Comprendre comment fonctionne la messagerie instantanée, est capital pour nous pour la conception de notre application.

Chapitre 4 : Conception

1- Introduction

La phase de conception est la première étape dans la réalisation d'un projet de développement informatique. Elle doit décrire de manière non ambiguë le fonctionnement futur du système, afin d'en faciliter la réalisation. Le but de ce chapitre est donc de présenter les objectifs de notre application, ce qui nous a amené à identifier les besoins des utilisateurs que nous avons essayé de projeter dans des diagrammes de cas d'utilisations.

2- Spécification des Besoins

Les besoins fonctionnelles :

- Inscription des utilisateurs
- Authentification
- Envoyer et Recevoir des messages (messages textes et émoticônes)
- Gestion de la liste des contacts, à savoir : Consulter la liste des contacts, l'ajout de nouveaux contact, rechercher un contact, bloquer un contact, Accepter/Refuser une invitation.
- Envoi de fichiers.
- Disponible en Français et en Anglais.
- Se déconnecter.

Les besoins non fonctionnels :

- La sécurité : Assurer la confidentialité des données, et le respect de la vie privée.
- Ergonomie : interface facile à prendre en main, l'esthétique n'est pas également en laisse

3- Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation représente la structure des grandes fonctionnalités nécessaires aux utilisateurs du système. C'est le premier diagramme du modèle UML, celui où s'assure la relation entre l'utilisateur et les objets que le système met en œuvre.

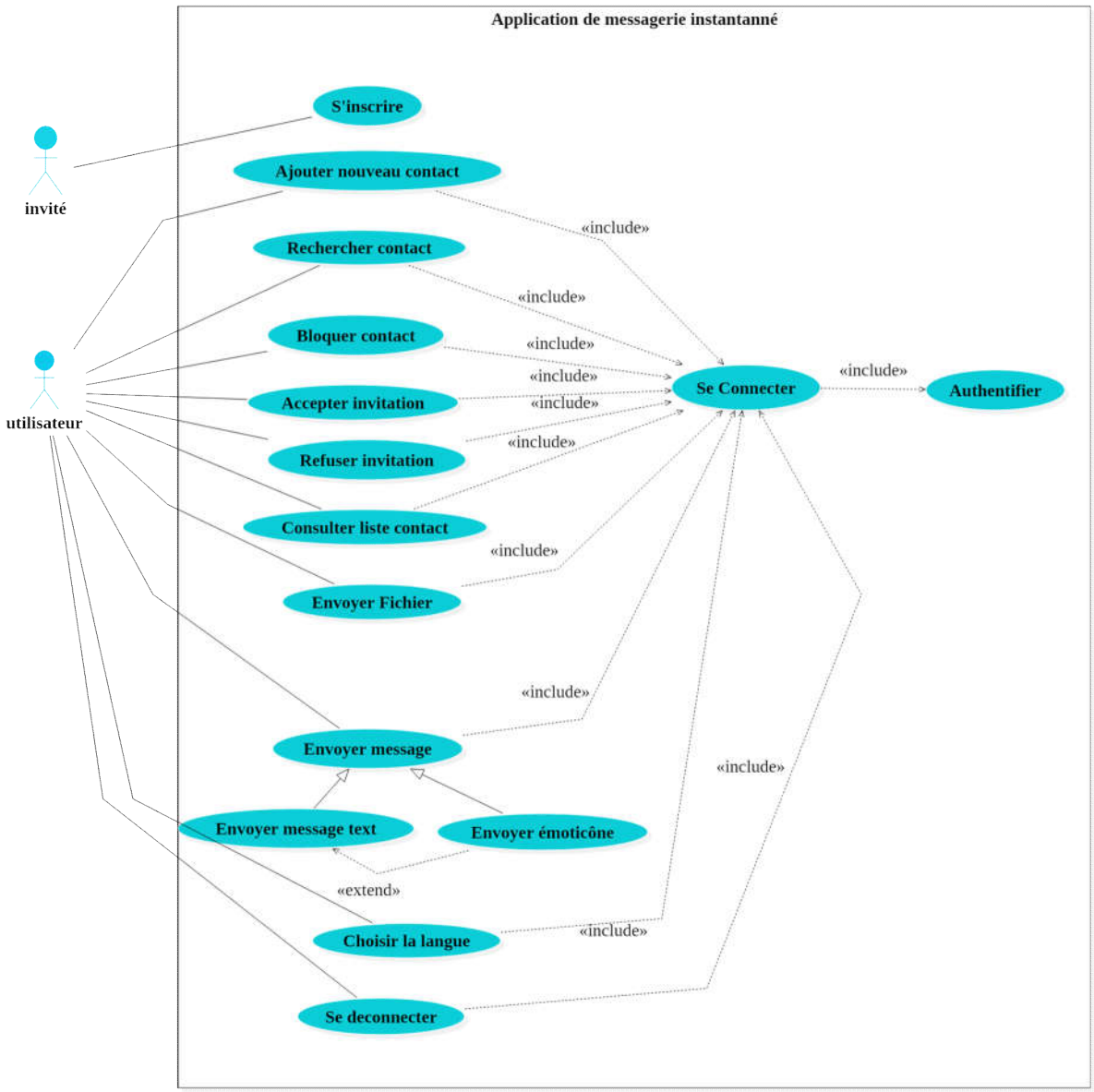


Figure 13 Diagramme de cas d'utilisation global du système

4- Diagrammes de séquences

Le diagramme de séquence est un diagramme parmi les diagrammes UML qui permet de décrire les interactions des éléments du système entre eux et avec les acteurs.

- Les objets au cœur d'un système interagissent en s'échangeant des messages.
- Les acteurs interagissent avec le système au moyen d'IHM (Interfaces Homme-Machine). [13]

4.1- Diagramme de séquence du cas d'utilisation « Inscription ».

Le diagramme de séquence suivant illustre les interactions nécessaires pour l'inscription d'un utilisateur.

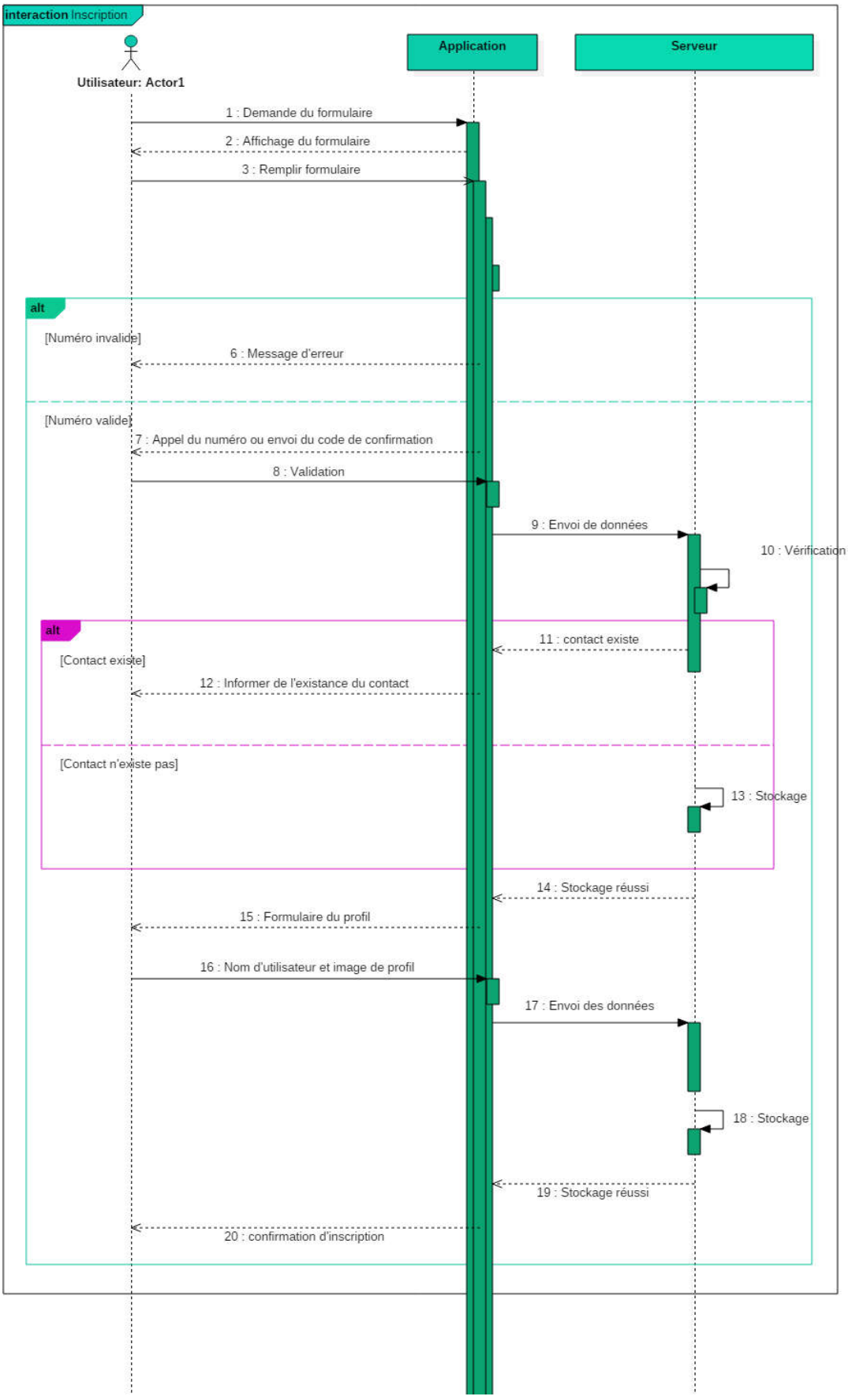


Figure 14 Diagramme de séquence du cas Inscription

4.2- Diagramme de séquence du cas d'utilisation « Authentifier »

Le diagramme de séquence suivant illustre les interactions nécessaires pour l'authentification d'un utilisateur.

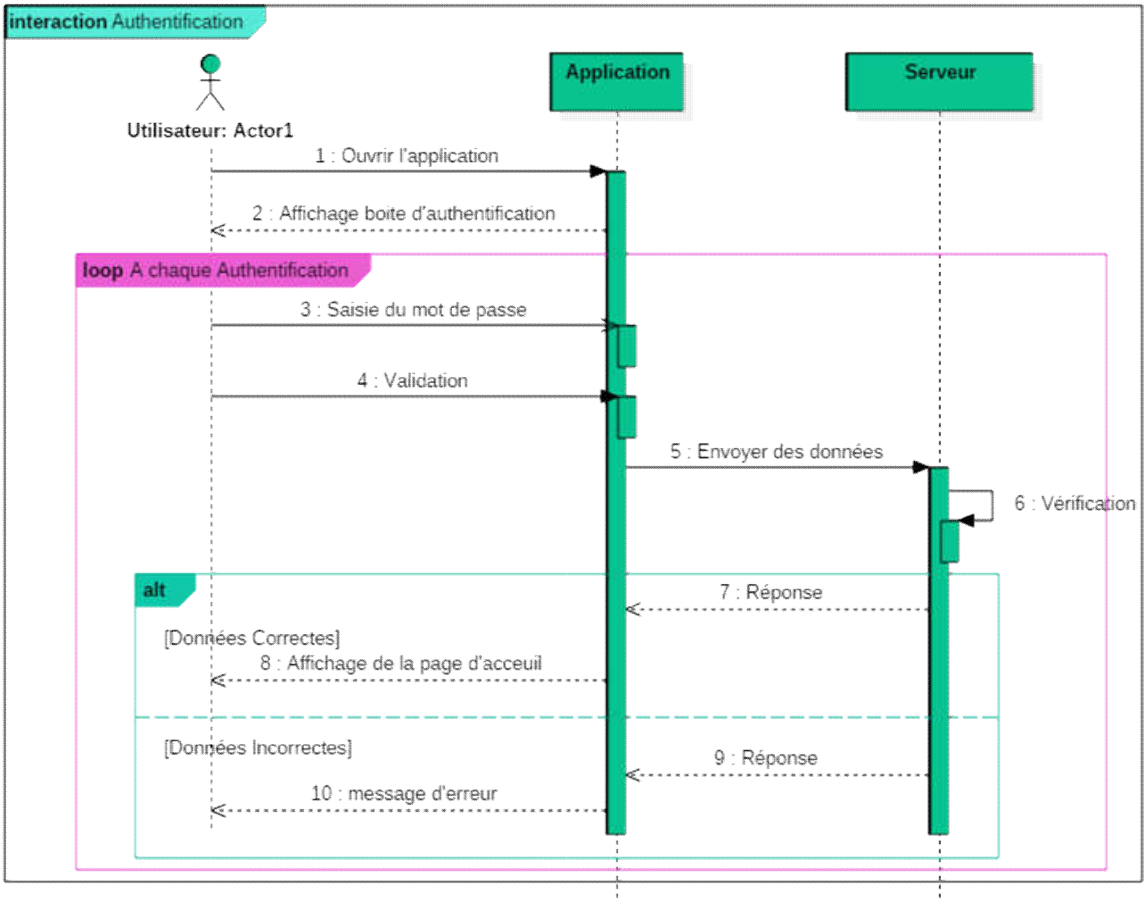


Figure 15 Diagramme de séquence du cas d'utilisation "Authentification"

4.3- Diagramme de séquence du cas d'utilisation « Envoyer un message »

Le diagramme de séquence suivant illustre les interactions nécessaires pour l'envoie de message.

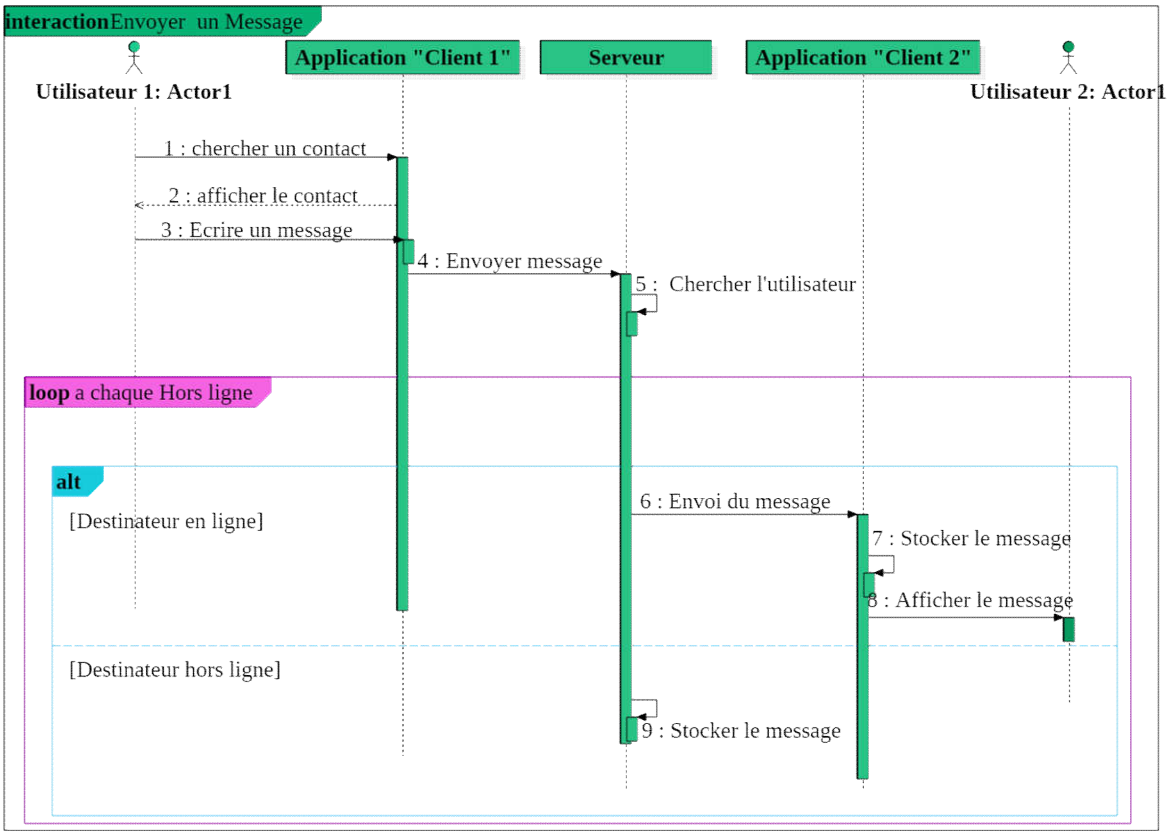


Figure 16 Diagramme de séquence du cas d'utilisation "Envoyer un message"

4.4- Diagramme de séquence du cas d'utilisation « Envoyer fichier»

Le diagramme de séquence suivant illustre les interactions nécessaires pour l'envoie de fichier.

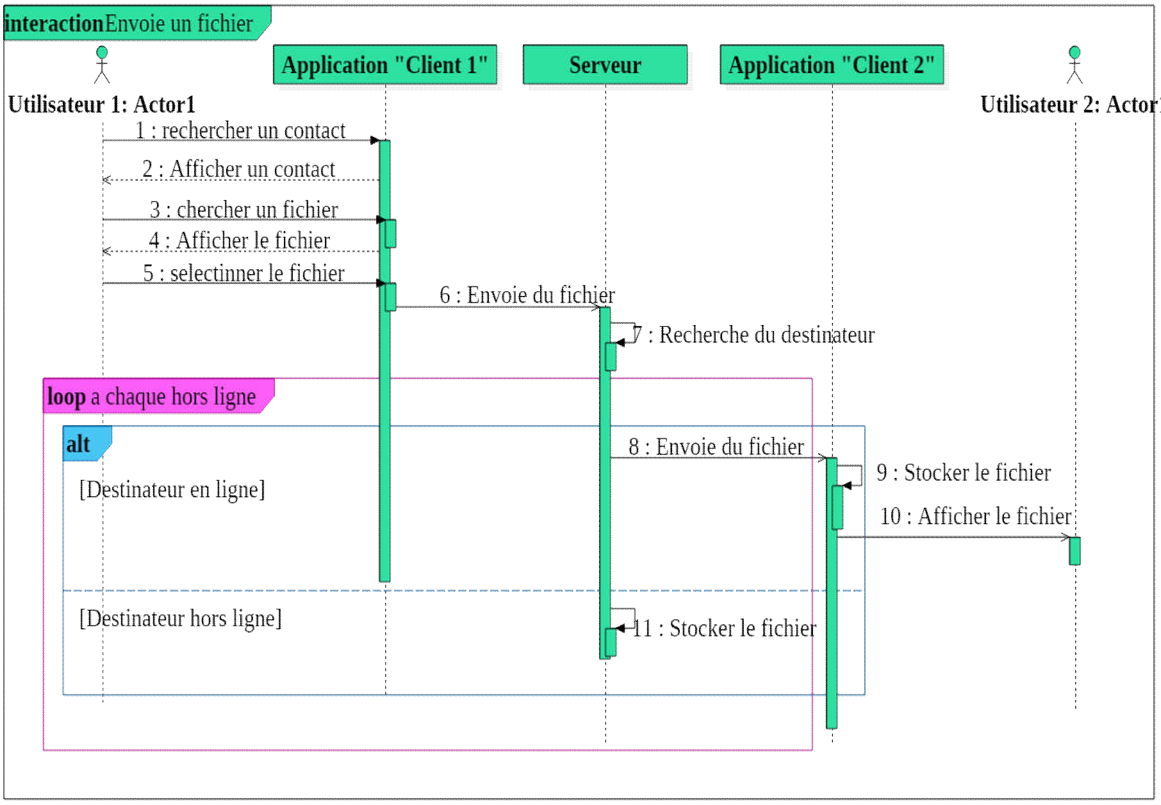


Figure 17 Diagramme de séquence du cas d'utilisation "Envoyer un fichier"

4.5- Diagramme de séquence du cas d'utilisation « Ajout de contact »

Le diagramme de séquence suivant illustre les interactions nécessaires pour l'ajout d'un nouveau contact.

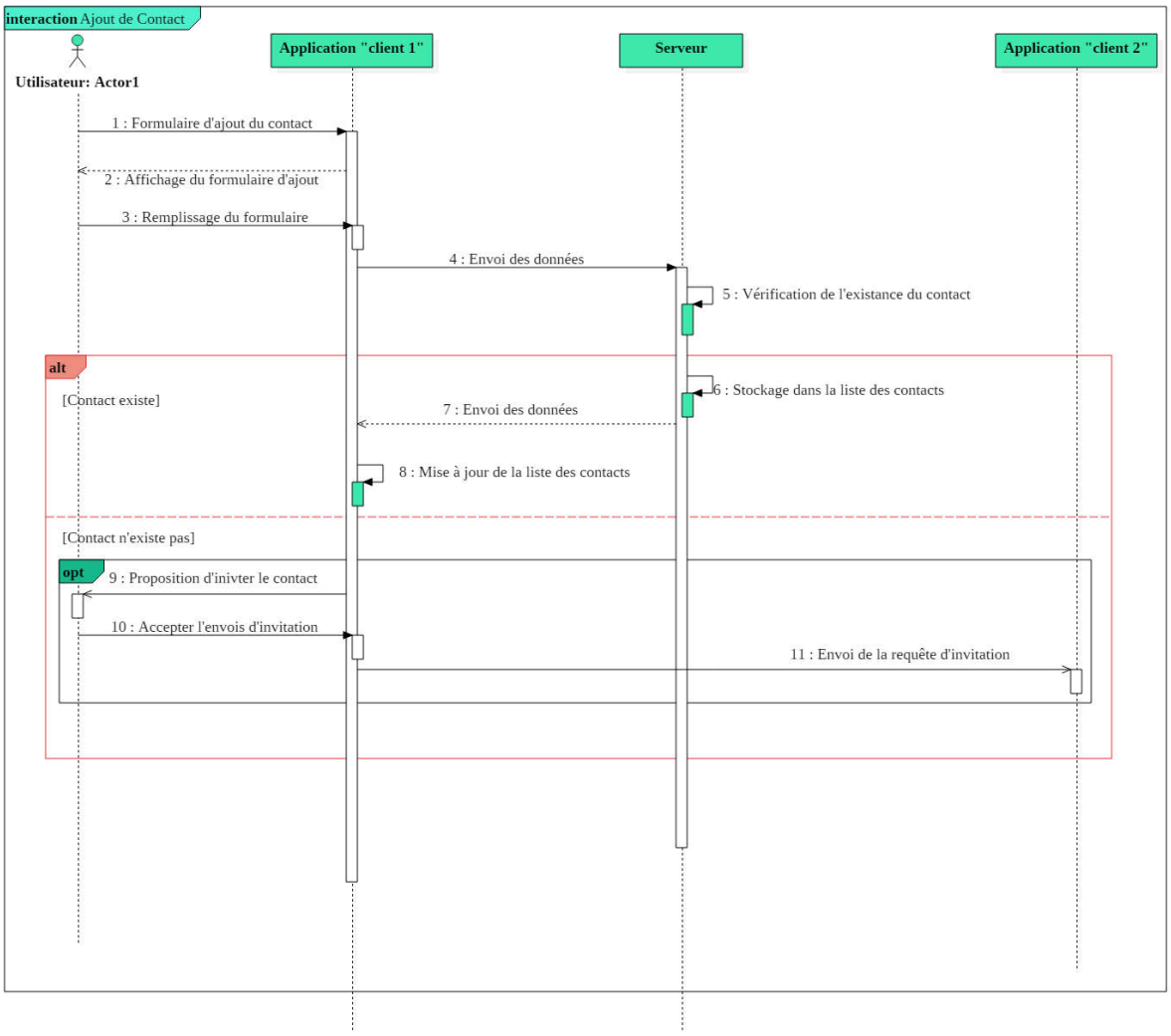


Figure 18 Diagramme de séquence du cas d'utilisation "Ajout de Contact"

4.6- Diagramme de séquence du cas d'utilisation « Bloquer un contact »

Le diagramme de séquence suivant illustre les interactions nécessaires pour bloquer un contact.

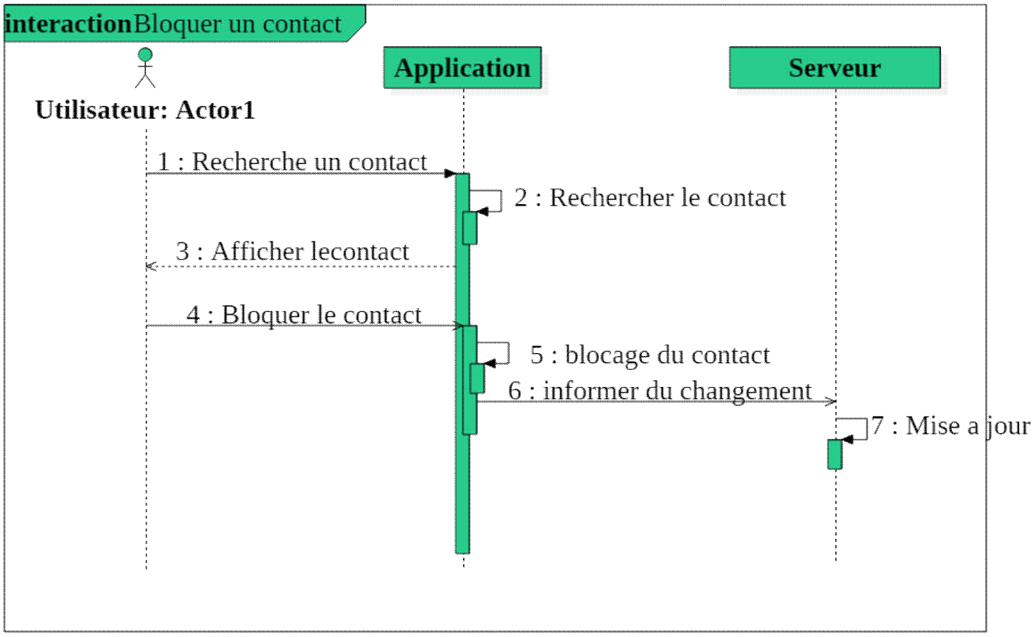


Figure 19 Diagramme de séquence du cas d'utilisation "Bloquer un contact"

4.7- Diagramme de séquence du cas d'utilisation « Déconnexion »

Le diagramme de séquence suivant illustre les interactions nécessaires pour la déconnexion d'un utilisateur.

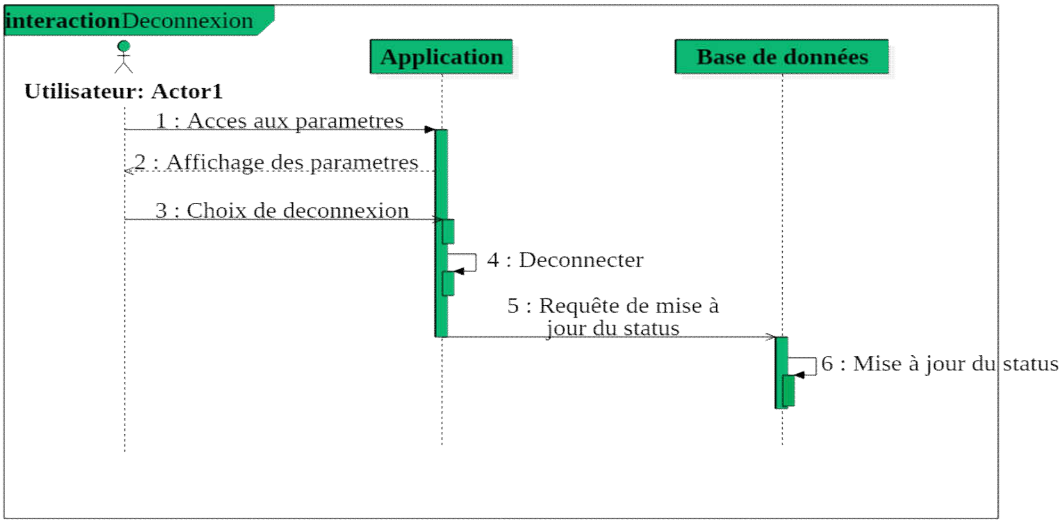


Figure 20 Diagramme de séquence du cas d'utilisation "Déconnexion"

5- Diagramme de classe de la base de donnée coté client

La figure suivante représente le diagramme de classe de la base de données côté client.

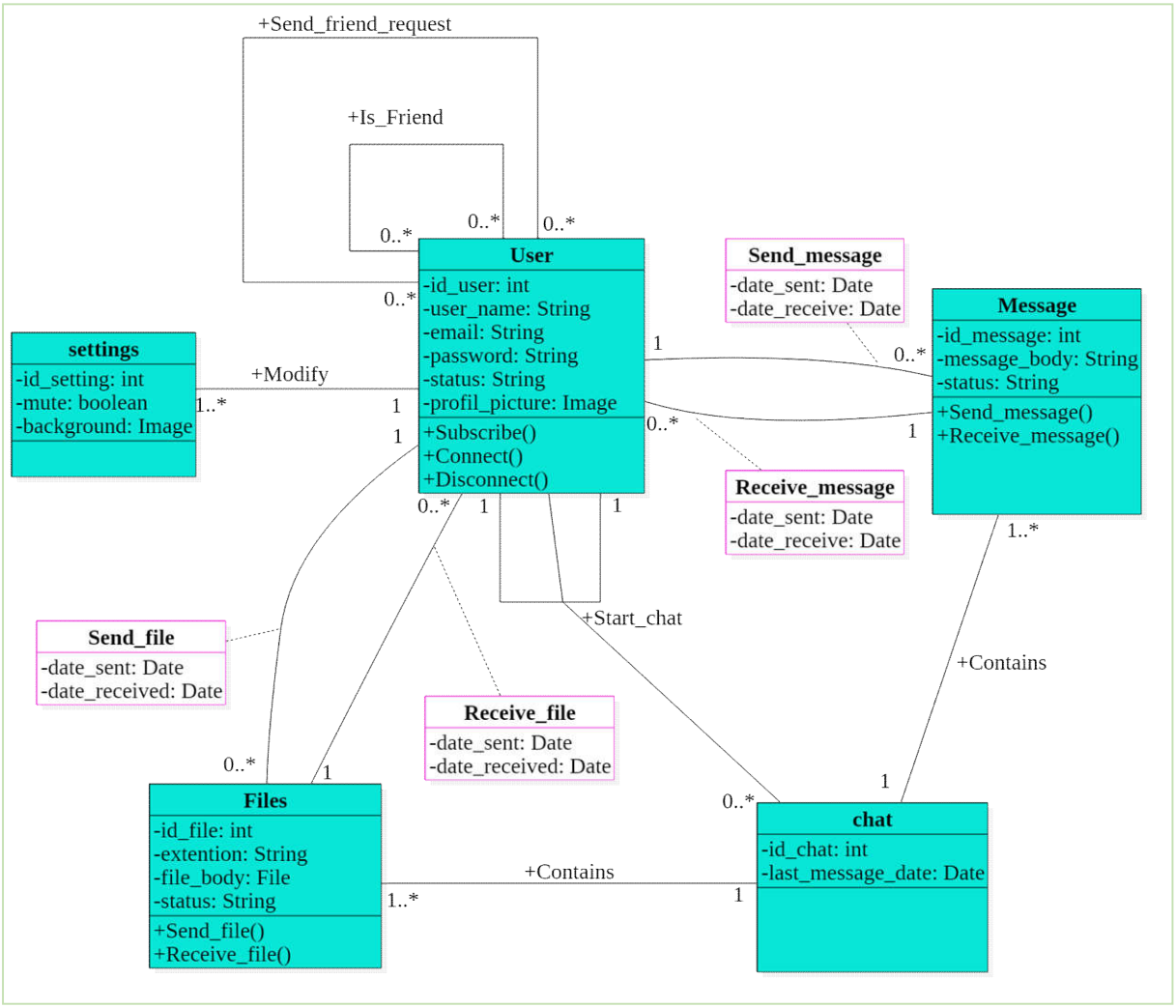


Figure 21 Diagramme de classe de la base de données coté client

6- Modèle Relationnel de la base de donnée coté client

Le modèle relationnel représente la base de données comme un ensemble de relations. Chaque relation est une table de valeurs ou fichier d'enregistrements. Lorsqu'une relation est conçue comme une table de valeurs, chaque ligne de la table représente un ensemble de valeurs reliées. Dans la terminologie du modèle relationnel, une ligne est appelée tuple, les entêtes de colonnes sont appelés attributs et les tables sont appelées relations.

Dans notre cas, nous disposons des relations :

- **User (id_user, user_name, email, password, status, profil_picture)**

Status = En ligne / Hors ligne. Cette relation contient les informations relatives à un utilisateur

- **Message (id_message, message_body, status, send_date, reveive_date, #id_user_from, #id_user_to, #id_chat)** Status peut prendre les valeurs : Lu ou non Lu

- **Settings (id_setting, mute, background, #id_user)** Mute peut prendre les valeurs : oui ou non

Cette relation comporte le paramétrage défini par l'utilisateur, à savoir le choix de l'arrière-plan et la mise en silence ou non des notifications.

- **Chat (id_chat, last_message_date, #id_user_from, #id_user_to) :** représente une discussion entre deux utilisateurs

L'association "Is_friend" devient la relation Friend_List. Cette relation permet de mettre en place une liste de contact pour un utilisateur donnée

- **Friend_list (#id_user1, #id_user2)**
- **Friend_request (#id_user1, #id_user2) :** Cette relation nous permettra dans la base de données de stocker toutes les invitations d'amitié adressé à un utilisateur.
- **File (id_file, file_body, status, send_date, reveive_date, #id_user_from, #id_user_to, #id_chat)** Status peut prendre les valeurs : lu / non lu

7- Conclusion

Dans ce chapitre nous avons déterminé le cadre du projet de réalisation de l'application et défini les besoins en identifiant les différents cas d'utilisations, ainsi que les ensembles des séquences d'actions. Tous ces éléments de conception, ainsi que l'architecture de notre application que nous avons eu à déterminer nous serviront de base pour la phase de réalisation. Cette conception nous servira de repère, pour la mise en place de l'application.

Chapitre 5 : Réalisation

1- Introduction

Ce chapitre est consacré à la réalisation et à la mise en œuvre de notre application de messagerie instantanée que nous avons appelé « WellDone ». Nous entamons ce chapitre par la description de l'environnement matériel et logiciel qui nous a permis la réalisation du projet et ensuite avant la phase d'implémentation dans laquelle nous aurons une présentation des différents modules de l'application nous ferons une description de l'architecture technique de l'application et nous terminerons ce chapitre par présenter les différentes interfaces de « WellDone » ainsi que son utilisation.

2- Environnement matériel

Le développement de l'application a été effectué sur un Ordinateur portable dont les caractéristiques sont consignées dans le tableau suivant :

Modèle	HP 250 G4 Notebook PC
Processeur	Intel® Core™ i3-5005U CPU @ 2.00GHz (4 CPUs)
Mémoire	4096 MB RAM
Carte graphique	Intel® HD Graphics 5500

Tableau 1 Tableau des caractéristiques du matériel utilisé [13]

3- Environnements Logiciels

Nous avons travaillé pour la réalisation de l'application sous Windows 10 Professionnel 64 bits (10.0 version 10240).

3.1- Environnement de développement

Les applications Android utilisent principalement la plate-forme Java pour s'exécuter. Il existe une alternative au Java pour le développement native d'applications Android : il s'agit du langage Kotlin. Notre choix s'est porté sur le Java.

Le développement Android requiert un environnement de développement ou IDE[8] (Integrated developement environment). Pour notre projet nous avons utilisé Android Studio. Il existe des alternative à Android Studio, par exemple Eclipse qui était utilisé au paravent mais de nos jours deviens de plus en plus délaissé. Android Studio est l'IDE privilégié par Google pour le développement Android.

3.1.1- Installation et Configuration d'Android Studio

Ayant choisi de faire le développement avec le langage Java il est indispensable au préalable d'installer et configurer la plate-forme Java.

Après le téléchargement et l'installation de l'Android Studio il faut par la suite installer le SDK.

- **SDK** : Un appareil sous Android ne comprend pas le Java tel quel, il comprend une variante du Java adaptée pour Android. Un SDK est un Kit de développement c'est-à-dire un ensemble d'outils permettant de développer pour une cible particulière. Par exemple pour développer pour une console de jeu vidéo, on utilise un SDK spécifique pour développer des applications pour cette console. Le SDK Android est donc un ensemble d'outils nécessaires pour créer, compiler et déployer les applications Android. Comme outils nous avons :
  - **L'AVD** : Android Virtual Device

    Appareil virtuel que le développeur peut utiliser dans l'émulateur pour tester ses applications.
  - **L'ADB** : Android Debug Bridge

    Permet de connecter un appareil Android virtuel ou réel, dans le but de gérer le périphérique ou de déboguer votre application.
  - **L'AAPT**: Android Assets Packaging Tool

  Sert à regrouper toutes les classes et ressources d'un projet dans un fichier .apk.

[8] *Un IDE ou environnement de développement est une suite d'applications et d'outils aidant à développer des applications (logiciels), gérer les fichiers sources, déboguer du code et enfin procéder à des tests.*

3.1.2- Création d'un projet Android avec Android Studio et son exécution

Pour créer un projet : ouvrez Android Studio, puis cliquez sur ***New Poject***. Une nouvelle fenêtre s'ouvrira. (Pour la version que nous utilisons à savoir Android Studio 3.0.1 cette fenêtre contient trois champs).

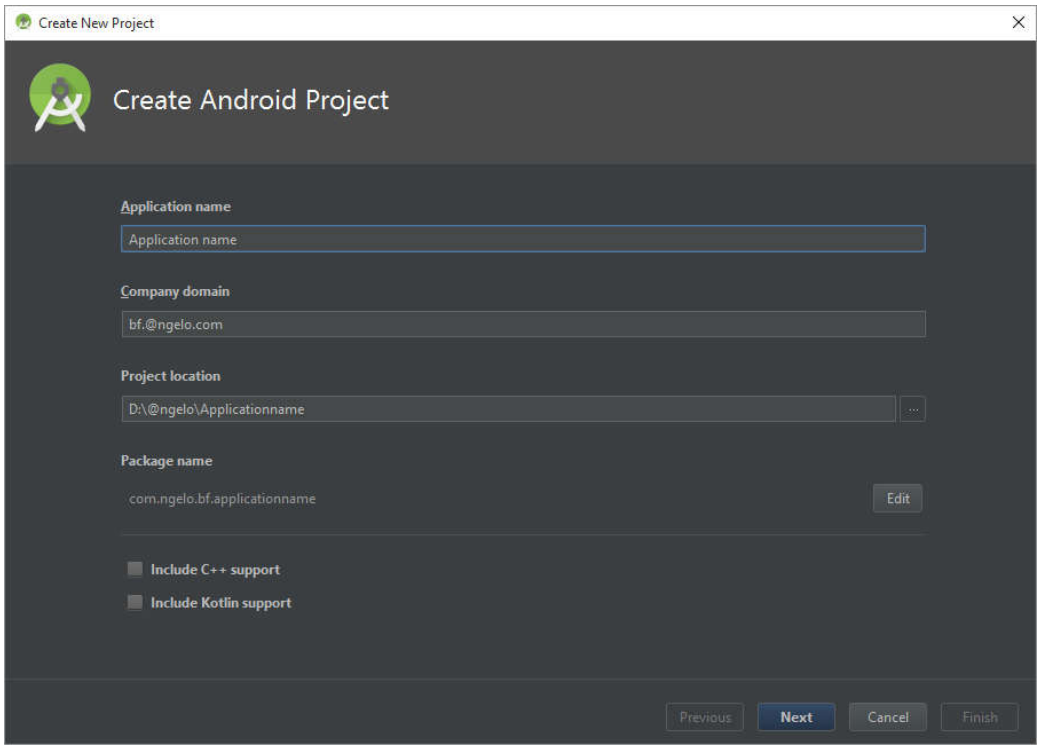


Figure 22. Première étape de la création d'un nouveau projet [13]

Rôle des différents champs :

- **Application Name** : Il s'agit du nom de l'application. Ce nom apparaîtra sur l'appareil et sur le Google Play.
- **Company Domain** : On se base sur le nom de domaine de son entreprise pour constituer ce champ. Ce champ permet à Android Studio de déduire automatiquement un « Package Name » qui sera utilisée pour déterminer dans quel package se trouvera votre projet. Ce package agira comme une sorte d'identifiant pour l'application sur le marché d'applications, celui-ci doit être unique et ne pourra pas être changé une fois l'application publiée.
- **Projet Location** : ici on indique l'emplacement ou les fichiers de notre projet seront créés.

Après avoir cliqué sur ***Next*** On passe à l'écran suivant.

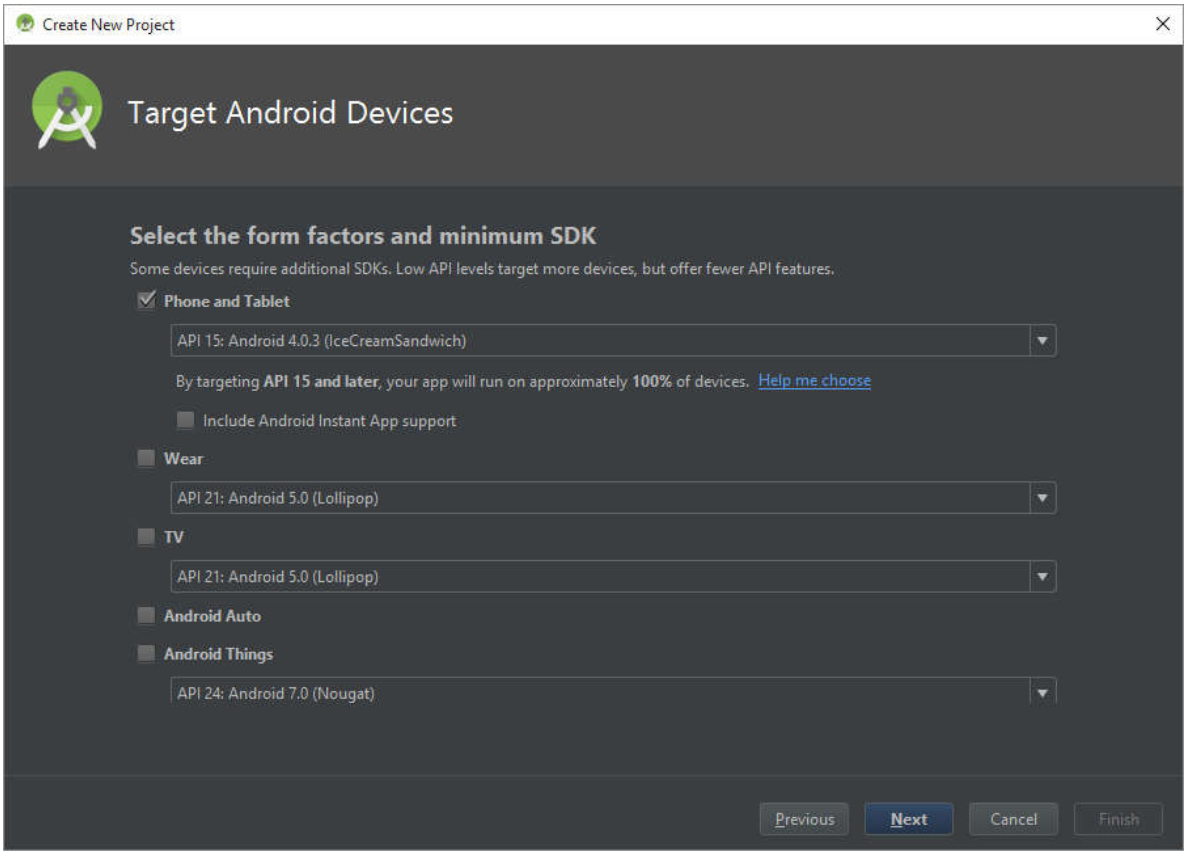


Figure 23 Sélection de l'API [13]

Cet écran permet de sélectionner le matériel de destination (Téléphone, ou tablette, ou Télévision etc.) de notre application, ainsi que la version de l'Android minimal que doit utiliser ce matériel.

Une fois la sélection faite, on clique sur ***Next.***

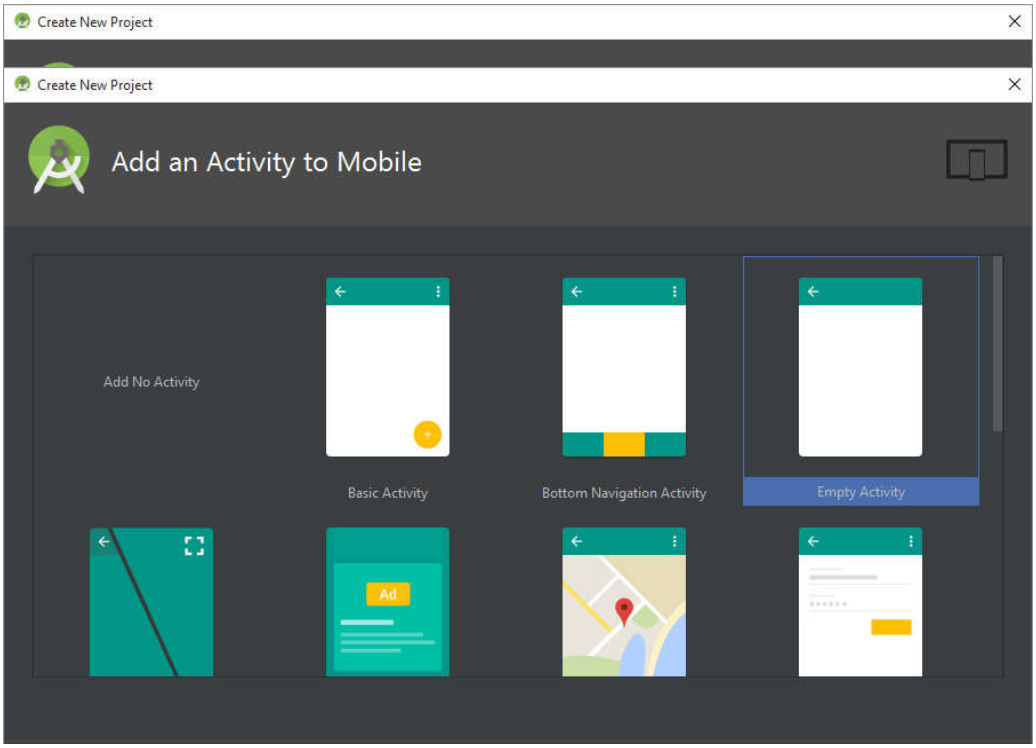


Figure 24 Création de la première activité de l'application [13]

Ici nous sélectionnons déjà le type d'interface graphique qu'on voudra pour le premier écran de notre application, puis ***Next.***

Ensuite on définit les informations sur la première activité de notre application.

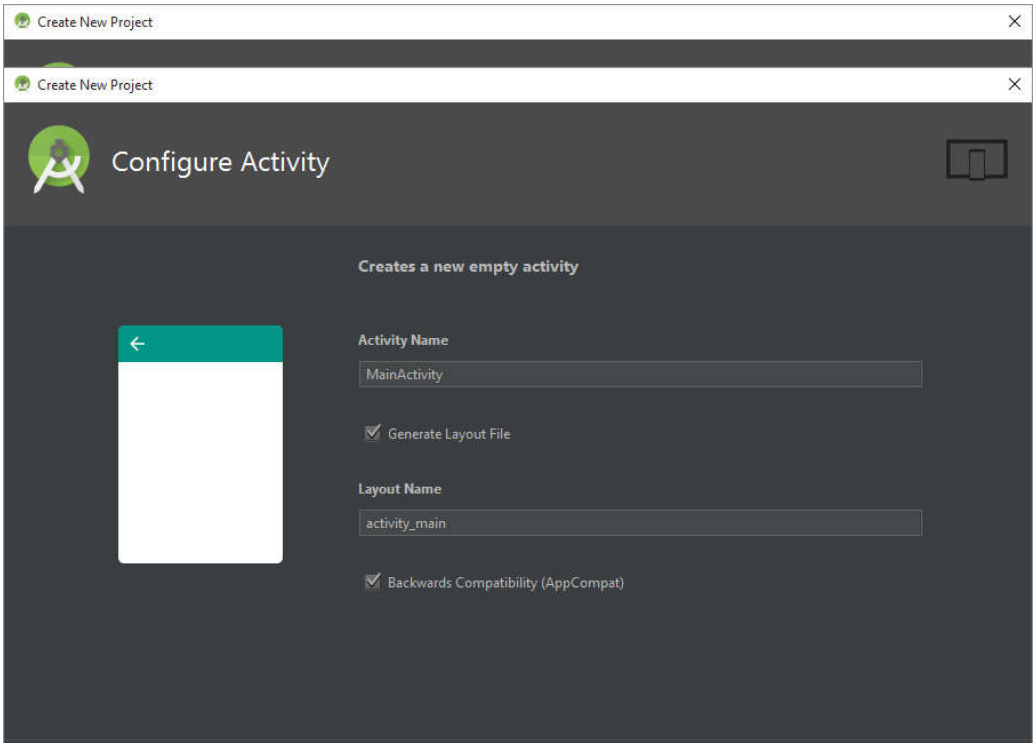


Figure 25 Informations sur la première activité de l'application [13]

Pour la configuration du vrai terminal :

- ❖ Tout d'abord nous devons télécharger les drivers adaptés à notre terminal.
- ❖ Configurer le téléphone pour qu'il puisse exécuter les applications que nous développerons et exécuter un debugger : pour cela il faut activer les options de développeurs sur le téléphone, ainsi que sélectionner l'option Débogage USB et activer les sources inconnues.

Une autre alternative à l'utilisation d'un vrai terminal est d'utiliser une Machine Virtuelle ou AVD (Android Virtual Device) qui va simuler une machine physique avec ses caractéristiques propres.

Pour le lancement de l'application :

- ❖ Utiliser la barre d'outils : où se trouvera les boutons servant à compiler, exécuter ou déboguer l'application.

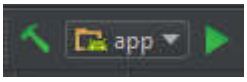


Figure 26 Groupe de buttons servant à compiler, exécuter ou déboguer une application avec Android Studio [13]

- ❖ Enfin on clique sur le bouton *play* et choisir le terminal sur lequel l'application sera exécutée.

3.2- Openfire (version 4.2.2) : le serveur de messagerie instantanée

Openfire est un serveur Jabber/XMPP écrit en Java et distribué sous licence publique générale GNU (GNU General Public License GNU GPL)[9]

Figure 27 Logo d'Openfire [13]

[9] ***GPL*** *est une licence qui fixe les conditions légales de distribution d'un logiciel libre du projet GNU (GNU est un système d'exploitation libre créé en 1983, il reprend les concepts et le fonctionnement d'UNIX)*

Openfire a pour caractéristiques :

- Il dispose d'une interface web d'administration
- Il dispose d'une interface pour les plugins
- Il est personnalisable
- Support SSL/TLS
- Permet une Connectivité avec une base de données (par exemple HSQLDB embarqué, MySQL …) pour conserver les messages et les informations des utilisateurs
- Il est indépendant de la plate-forme utilisée, Java pur.

L'essentiel de la configuration et de l'administration du serveur se fait à travers une interface web. Les administrateurs peuvent se connecter de n'importe où et éditer la configuration du serveur, ajouter ou supprimer des utilisateurs, des salons de conversation…

Pour se connecter à l'interface il faut s'authentifier avec les paramètres d'authentification choisis lors de l'installation et la configuration d'Openfire.

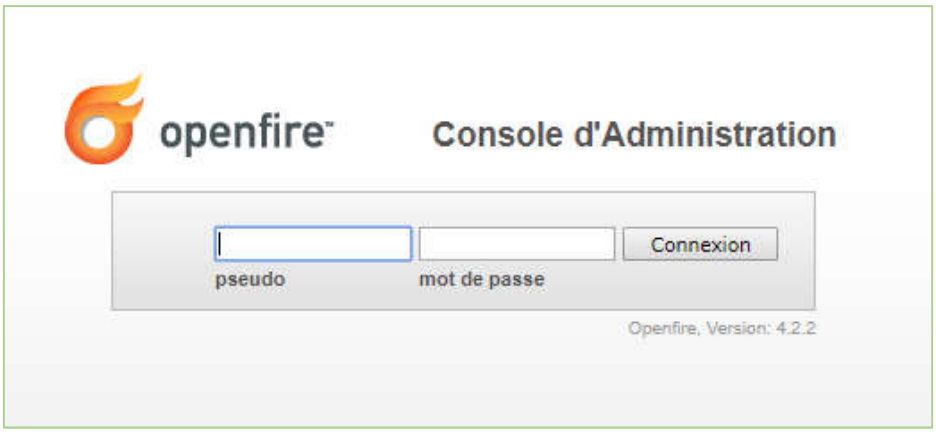


Figure 28 Interface d'authentification pour accéder à la console d'administrateur d'Openfire [13]

3.3- Wamp Server (MySQL server)

Pour l'implémentation de la base de données, nous avons utilisé l'environnement de création de base de données PHPMyAdmin et le système de gestion de base de données MySQL.

WampServer est une plateforme de développement Web sous Windows. Il permet de développer des applications Web dynamiques à l'aide du serveur Apache2, du langage PHP et d'une base de données MySQL. [12]

Notre base de données sous MySQL est connectée au serveur Openfire.

3.4- Technologies utilisées

3.4.1- Le Langage Java

Java est un langage de programmation orienté objet, développé par Sun Microsystems. Il permet de créer des logiciels compatibles avec plusieurs environnements (Windows, Linux, Solaris …). Egalement donne la possibilité de programmer des applications mobiles.

Le Java est un langage pouvant également être utilisé sur internet pour des petites applications intégrées à des pages web (applet) ou encore comme langage côté serveur (jsp).

3.4.2- Le langage XML

Le langage XML (eXtanded Markup Language) est un langage de balisage, et est un format général de documents orienté texte. Il s'est imposé comme un Standard incontournable de l'informatique et est aussi bien utilisé pour le stockage de documents que pour la transmission de données entre applications. Sa simplicité, sa flexibilité et ses possibilités d'extension ont permis de l'adapter à de multiples domaines allant des données géographiques aux dessins vectoriels en passant par les échanges commerciaux. Le XML est utilisé sous Android pour :

- Définition des interfaces graphiques
- Définition des ressources : couleurs, chaines de caractères, dessin vectoriel etc.

4- Présentation de l'Architecture de l'application

Notre application se base sur une architecture centralisée client-serveur.

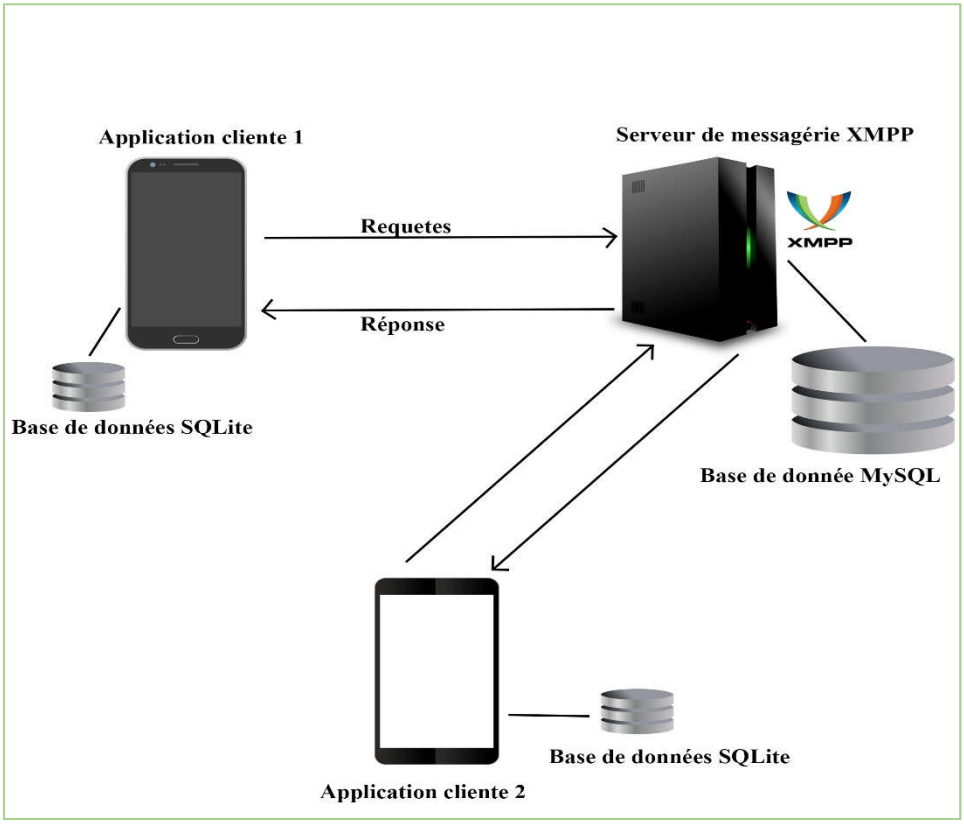


Figure 29 Architecture de l'application [13]

4.1- Serveur de la messagerie instantanée

Coté serveur, nous avons une base de donnée, dans laquelle sera stockées :

- Les informations concernant un utilisateur lors de son inscription à savoir (son numéro de téléphone, son nom d'utilisateur, et un mot de passe).
- Les messages et fichiers transmis entre les utilisateurs, quand le récepteur est hors ligne. Et dès que celui-ci sera en ligne, les messages et fichiers lui seront transmis et, ceux-ci seront supprimé coté serveur. Ainsi au sein de notre base de données il n'y aura pas de sauvegarde de message. Les messages seront juste stockés coté client dans une base de données, pour une consultation hors ligne.

La base de données doit respecter un schéma de base dans le but d'être relier au serveur XMPP. Si la nécessité de changer le schéma de la base de données à savoir ajout, renommage, ou modification de table il faut pour cela modifier le code source du serveur pour l'adapter à ce

changement. Pour rappel le serveur XMPP est libre ce qui veut dire que le code source est téléchargeable et modifiable.

Dans notre cas on utilise le serveur XMPP Openfire, et la plupart des noms des tables de la base de données MySQL commencent tous par « of » pour *openfire.*

Modèles relationnels de quelques tables de la base de données MySQL sont :

- **ofuser** (**username**, storedKey, serverKey, encryptedPassword, name, email, creationDate, modificationDate).

  encryptedPassword représente le mot de passe crypté de l'utilisateur

- **uerstatus (username, ressource**, online, presence, lastIpAddress, lastLogoffDate). Cette contient le statut d'un utilisateur, à savoir si celui-ci est en ligne ou non
- **ofoffline (username, messageID**, creationDate, messageSize, stanza).

  C'est dans cette table que sont stocké les messages envoyés à un utilisateur lorsque celui n'est pas en ligne c'est à dire lorsqu'il est offline.

- **ofpresence (username,offlinePresence**,offlineDate).

  Dans cette table sont stockées les différentes présences des utilisateurs. La présence est un texte qui sera affiché chez vos contacts, par exemple pour dire que vous êtes disponible ou non disponible.

- **ofroster** // Dans cette table est stocké la liste de contact d'un utilisateur
- **ofvcard** // C'est ici que les informations complémentaires d'un utilisateur sont stocké : à savoir la photo de profil, le nom, le numéro de téléphone, etc.

4.2- Client de messagerie instantanée

Dans notre application nous avons une base de données locale (SQLite). Cette base de données sert à stocker :

- Les informations nécessaires pour amorcer l'application, c'est-à-dire enregistrer les données nécessaire à l'authentification de l'utilisateur, pour un nouveau lancement de l'application sans devoir s'authentifier encore. (*cette fonctionnalité est implémentée en*

utilisant un autre système de stockage de données sous Android appelé « SharedPreference[10]» ou préférence partagée).

- ❖ L'historique des messages d'un utilisateur, pour une consultation hors ligne des discussions.
- ❖ La liste des contacts

5- Implémentation des modules

Notre application est structurée en plusieurs modules. Les plus importants sont :

5.1- Le module « XMPP »

Ce module assure l'envoie et la réception des requêtes entre l'application et le serveur de messagerie, en utilisant le protocole XMPP. Nous avons au sein de ce module deux classes :

- ❖ **La classe « Connection_welldone »** : qui assure tous les éléments de connexion avec le serveur. Elle permet de gérer la connexion avec le Server Openfire. C'est à l'intérieur de cette classe que le protocole XMPP a été implémenté en utilisant la librairie Smack[11].

  L'implémentation du protocole fait référence à :

  - La Connexion au serveur
  - L'envoi de messages
  - La Gestion des contacts
  - La Gestion de la présence, etc.

- ❖ **La classe « ConnectionService_welldone »**: qui est un service dans laquelle la connexion au serveur est exécutée dans un Thread. Le service est lancé en tâche de fond donc pas dans le même Thread que le thread de l'interface graphique appelé UI Thread. Le service continu de tourné même si l'application est fermée.

---

*[10] Le **sharedPreference** stock les données sous forme de **clé / valeur**. Ces données sont stocké dans un fichier XML et peuvent être des booléens, des réels des chaines de caractère. Egalement Ces données persistent jusqu'à la désinstallation de l'application et sont accessible partout dans l'application.*

*[11] **Smack** : est une bibliothèque client Open Source XMPP (Jabber) pour la messagerie instantanée et la présence. Une bibliothèque Java pure.*

5.2- Le module « Database »

Ce module est chargé de la Création et la gestion de la base de données SQLite : Pour créer une base de données SQLite sous Android, il faut définir une nouvelle classe (DatabaseBackend par exemple) qui héritera de la super classe SQLiteOpenHelper.

C'est dans ce module que les différentes tables de la base de données embarquée SQLite sont créer en utilisant les requêtes SQL[12]

5.3- Le module « Nexmo Verify »

L'inscription d'un utilisateur a été implémentée dans ce module en utilisant l'API Verify de la plateforme d'API Nexmo. Dans ce module nous utilisons cet API pour la vérification des numéros de téléphone et gérer l'envoi de code de confirmation lors de l'inscription.

Le numéro de l'utilisateur est récupéré et envoyé sur le serveur Nexmo. Après vérification du numéro un code de confirmation est envoyé sur le numéro par sms. Ce code reste valide pendant un temps déterminé (nous avons choisi pour notre application un temps de 5 minutes) et si l'utilisateur ne valide pas le code, un appel sera effectué sur le numéro avec le code.

Dès que le numéro sera vérifié et le code confirmé. Le numéro est envoyé au niveau de notre serveur pour être inscrit dans la base de données.

[12] ***SQL*** *: Structured Query Language. C'est un langage informatique normalisé servant à exploiter des bases de données relationnelles.*

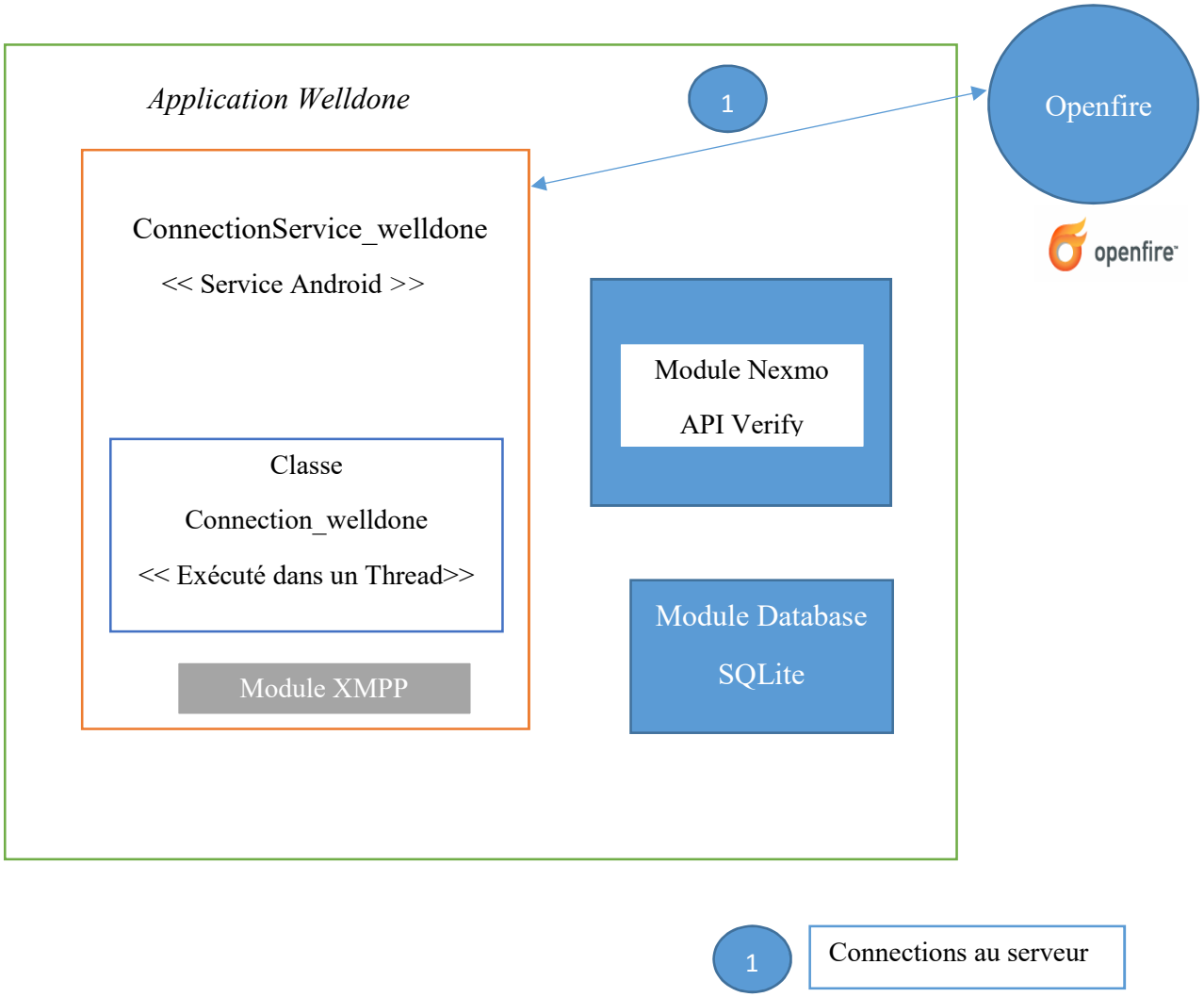
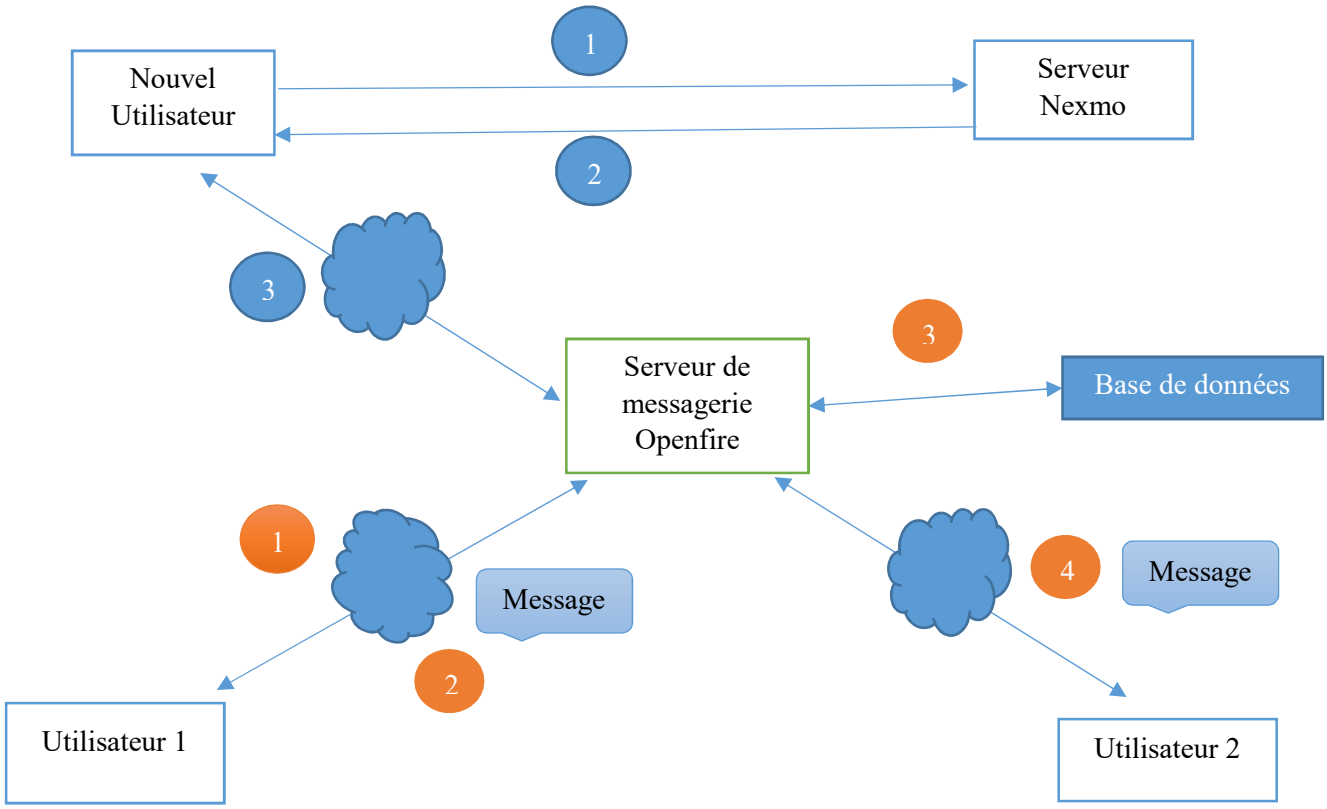


Figure 30 les modules de l'application [13]



1	Envoi du numéro de téléphone sur le serveur Nexmo pour vérification (API verify)
2	Numéro vérifié et envoi du code de confirmation
3	Finalisation de l'inscription.
1	Démarrage d'un chat XMPP après avoir créé une session
2	Envoi d'un message
3	Si utilisateur non connecté, le message est sauvegardé dans la base de données et sera redistribué dès qu'il sera en ligne
1	Réception de message

Figure 31 Schéma fonctionnelle de l'application [13]

6- Présentation de l'application « WellDone »

Dans cette partie nous procèderons à la présentation de WellDone en montrant les interfaces des différentes activités qui constituent l'application.

6.1- Interface d'accueille

Cette interface s'affiche en premier lors du lancement de l'application, souhaitant la bienvenue aux nouveaux utilisateurs. Après l'inscription cette interface ne s'affichera plus aux prochains lancements.

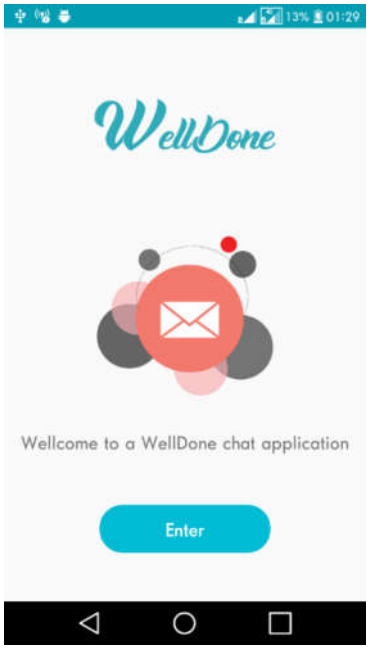
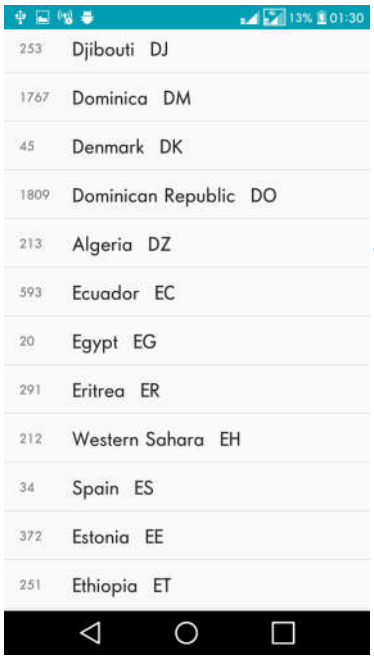


Figure 32 Interface d'accueil de WellDone

6.2- Interfaces d'inscription

Les figures suivantes offrent l'aperçu des interfaces pour la création d'un nouveau compte WellDone.

- Choix du pays et entrer du numéro de téléphone.



Choix du pays

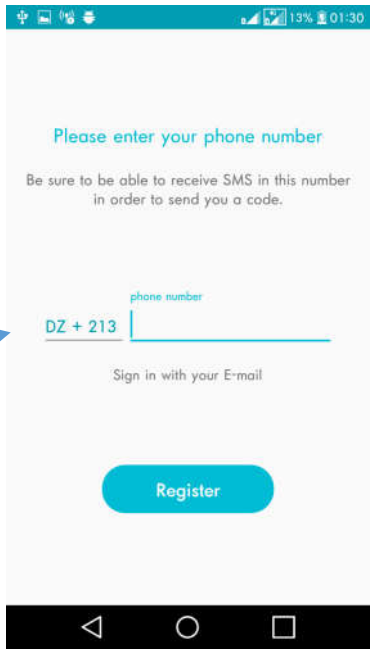


Figure 37 Interface pour le choix du pays

Figure 33 Interface pour le renseignement du numéro de téléphone

- Après avoir renseigné le numéro de téléphone, et cliquer sur le Bouton « **Register** ». Une boite de dialogue s'ouvre pour la confirmation de la validité du numéro par l'utilisateur.

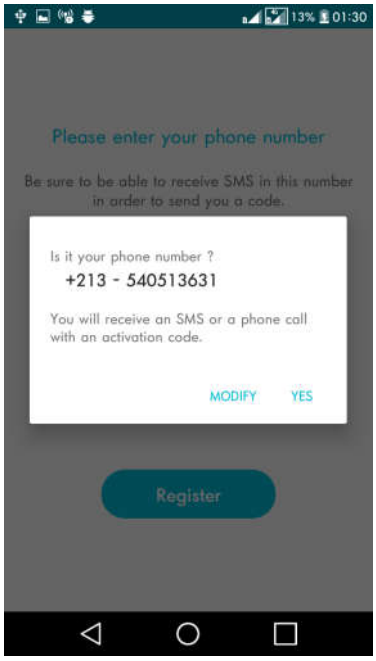


Figure 41 Interface de validation du numéro de téléphone par l'utilisateur

❖ Réception du code de confirmation par SMS

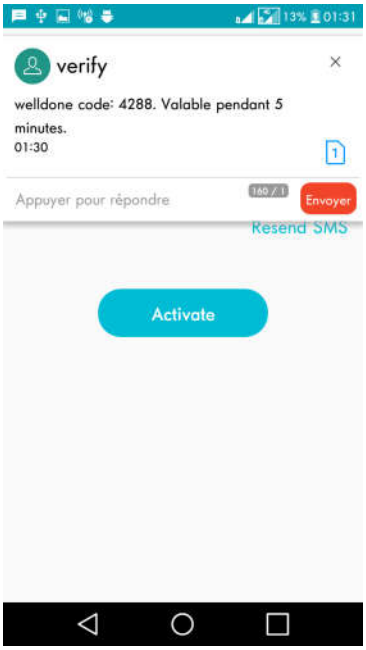


Figure 45 Réception du message avec le code de vérification

❖ L'utilisateur après avoir saisie le code de vérification et validé, doit maintenant finaliser son inscription en choisissant une photo de profil (optionnel), et un nom d'utilisateur (obligatoire).

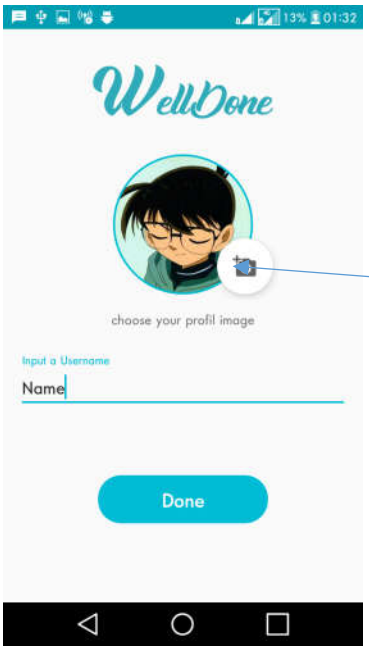


Figure 46 Interface pour le choix de la photo de profil et le nom d'utilisateur

6.3- Interface liste des discussions (« Chat list »)

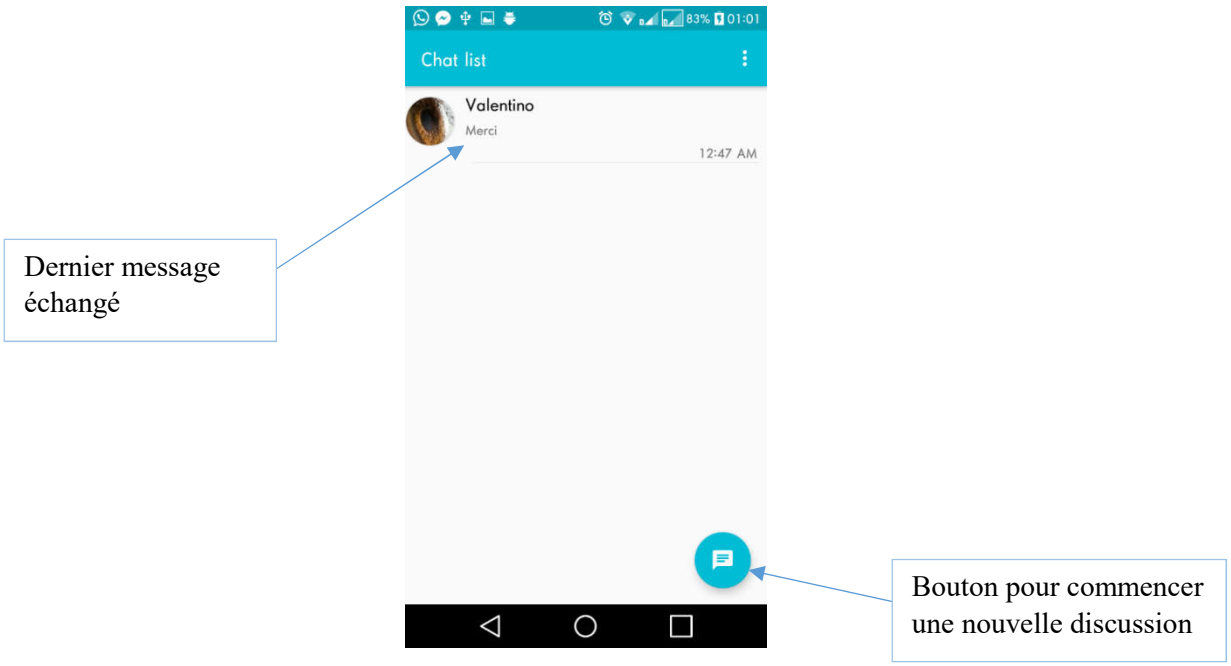


Figure 47 Interface liste des discussions

6.4- Interface de Discussion (Envoi/Réception de messages/fichiers)

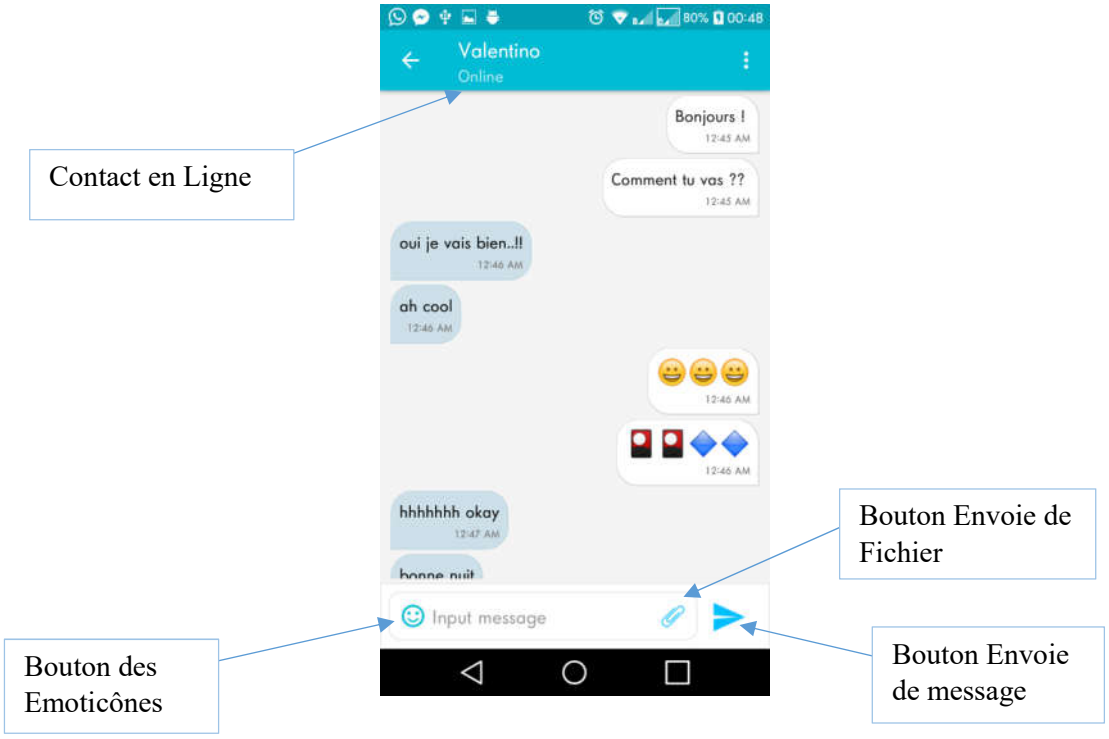


Figure 48 Interface d'une discussion

6.5- Interface profil

Dans cette interface l'utilisateur a la possibilité de changer sa photo de profil et de voir son Statut vis-à-vis du serveur, à savoir s'il est connecté ou déconnecté ou en cours de connexion.

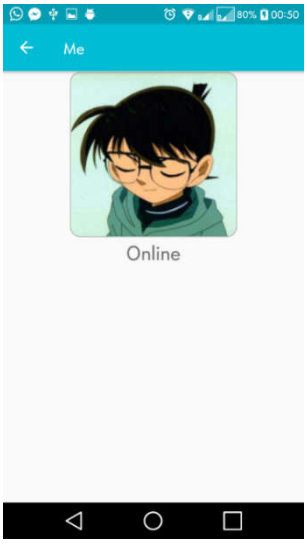


Figure 49 Interface de profil

6.6- Interface profil d'un contact (« Contact details »)

Dans cette interface on a le profil d'un contact, et la possibilité de se souscrire à sa présence c'est-à-dire faire une requête à l'utilisateur afin de voir sa présence (en ligne ou hors ligne).

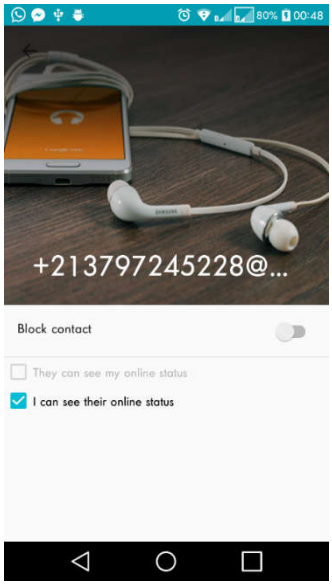


Figure 50 Interface du profil d'un contact

6.7- Interface Liste des contacts

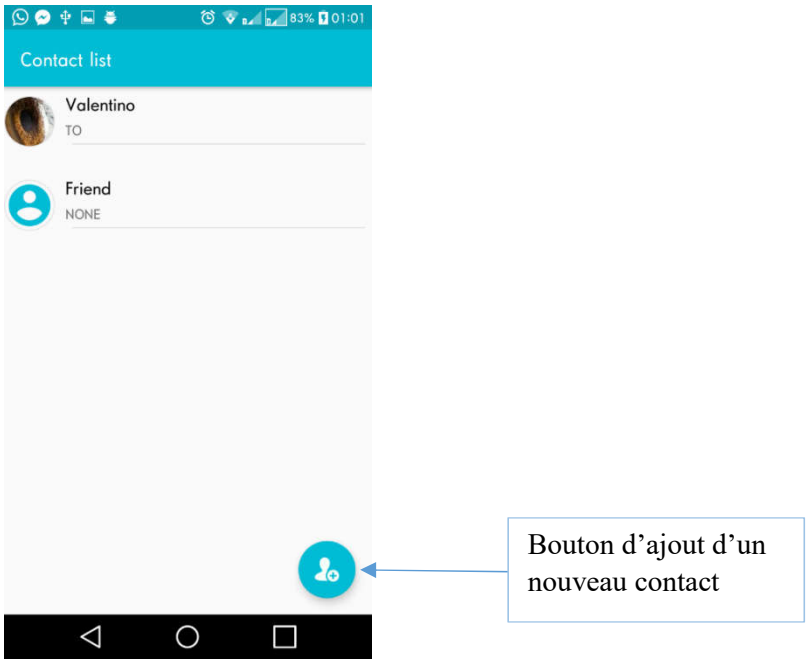


Figure 54 Interface liste des contacts

6.8- Interface d'ajout de contact

Dans cette Activité l'utilisateur a la possibilité d'ajouter un nouveau contact à sa liste des contacts.

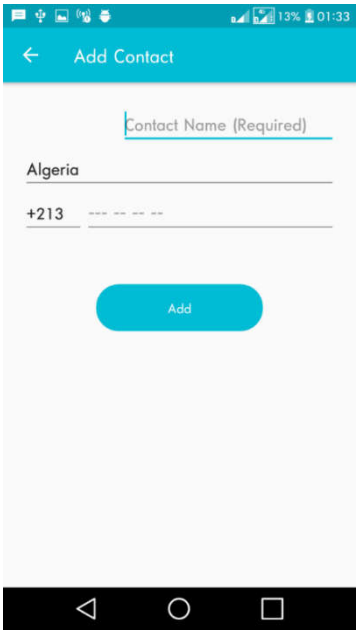


Figure 55 Interface d'ajout de contact

Conclusion générale et Perspectives

La place de la télécommunication dans notre quotidien ne fait que prendre de l'ampleur. Ceci grâce à la grande gamme d'applications : à savoir les applications de messageries instantanées qui disposent de nombreuses fonctionnalités et qui ne font qu'évoluer.

Nous sommes parvenus, par le biais de ce projet, à réaliser une application qui fonctionne sous Android qui permet aux utilisateurs d'échanger des messages de façon instantanée.

Au cours de la phase d'analyse nous avons structuré et défini les besoins du système en analysant la pluparts des cas d'utilisations par les diagrammes UML qui est suivie directement par une phase de conception.

Grâce à certains Outils d'implémentation et de technologies appropriés, nous avons pu procéder à la réalisation de l'application. Ainsi la réalisation étant faite notre application donne la possibilité aux utilisateurs de :

- S'envoyer des messages de manière instantanée
- S'échanger des fichiers
- Envoyer des Emoticônes
- Voir la présence de leur contact (en ligne, hors ligne)
- Utiliser des photos de profils
- Ajouter de nouveaux contacts.
- Envoyer à des personnes des invitations à utiliser l'application.
- Consulter l'historique de ses messages
- Après s'être connecté, recevoir les messages envoyés quand l'application était hors ligne.

Tout cela après avoir créé un compte.

Ce travail nous a permis d'apprendre énormément de choses concernant le développement sous Android, et ce mémoire présente notre première expérience pour la mise en œuvre d'une application mobile. Egalement nous avons appris à manipuler toute une panoplie d'outils : Android Studio, Openfire, et quelques langages de programmation telles que JAVA, XML, SQL. De plus à comprendre le fonctionnement de base du protocole XMPP, ainsi que son implémentation. A rappeler qu'il existe peu de documentation détailler sur son implémentation, ce qui nous amener à nous efforcé de comprendre en ayant comme ressource principale de recherche la documentation officiel.

Ce fut une occasion pour nous de compléter de manière transversale nos compétences en informatique, d'élargir et d'approfondir nos connaissances.

Cependant des perspectives d'améliorations de notre application restent envisageables pour être enrichie car avant de commencer la réalisation de notre application, nous avons eu à nous fixer certains objectifs. Mais pour des raisons de manque de temps et de manques de ressources, quelques-uns de ces objectifs n'ont pas été misent en place. A savoir :

- Crypter l'envoi de message et de fichier par un algorithme fiable et sur
- Utiliser le Cloud pour la synchronisation de l'application
- Ajouter la discussion de groupe
- Le partage de la localisation, ainsi que la fonctionnalité d'appel vidéo et audio.

Pour une prochaine version ces fonctionnalité seront prisent en compte.

Bibliographie

[1] KEDJAR Lahcene; OUMAKHLOUF Mokhtar. *Conception et réalisation d'une application de messagerie instantanée sous Android*. Master Informatique : Administration et sécurité des réseaux. Bejaïa : Université A/Mira de Bejaïa, 2015. 71p.

[2] BENBOURAHLA, Nazim. *La plateforme Android*. **In** : *Android 4 les fondements du développement d'application Java*. Editions ENI, septembre 2012. p. 13-14. Collection **Ressources Informatiques**. ISBN 978-2-7460-7560-3

[3] BENBOURAHLA, Nazim. *Principes de programmation*. **In** : *Android 4 les fondements du développement d'application Java*. Editions ENI, septembre 2012. p. 45-54. Collection **Ressources Informatiques**. ISBN 978-2-7460-7560-3

[4] Anca Boboc, « Le point sur la messagerie instantanée. Solutions grand public (im) et solutions d'entreprise (eim) », *Réseaux* 2005/6 (no 134), p. 223-261.
DOI 10.3917/res.134.0223

[5] Bazara Barry; Fatima M. Tom. *INSTANT MESSAGING: STANDARDS, PROTOCOLS, APPLICATIONS, AND RESEARCH DIRECTIONS.* **In:** Internet Policies and Issues, Volume 7. Edition B. G. Kutais, July 2010. Chapiter 8. ISBN 978-1-61668-745-8

[6] Omar BOUKADOUM. *Introduction à UML 2 : Modélisation Orientée Objet de Systèmes Logiciels. Diagramme de séquence*. Université de Khémis Miliana, Licence 3, Cours, 2017/2018, 18p

[13] Réalisation personnelle (par capture d'écran, ou en utilisant le logiciel de dessin Photoshop)

Webographie

[7] OpenClassrooms. *Créer des applications pour Android* [**en ligne**]. Consulter le 18 Janvier 2018. Disponible sur https://openclassrooms.com/courses/creez-des-applications-pour android/

[8] Pierre Vitré. *Les meilleures applications de messagerie instantanée sur Android*. **In** : *ANDROIDPIT* [**en ligne**]. (Modifié le 14 novembre 2017). Consulté le 22 Janvier 2018. Disponible sur : https://www.androidpit.fr/meilleures-applications-messagerie-instantanee-android

[10] http://images.frandroid.com/wp-content/uploads/2017/02/bugdroid.png

[11] https://www.cnews.cz/galerie/pictures/novinky/2015/06cerven/messenger-bude-fungovat-i-bez-uctu-na-facebooku/messenger_bude_fungovat_i_bez_uctu_na_facebooku.png

[12] https://www.wampserver.com.

[14] https://xmpp.org

Annie-Claude Coze ; Martine le Grand. *La messagerie instantanée* [**en ligne**]. (Modifier le Jeudi 13 septembre 2004). Consulté le 23 Janvier 2018. Disponible sur : https://creg.ac-versailles.fr/la-messagerie-instantanee

https://developer.android.com/

https://www.messenger.com/features

Table des matières