REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement et de la Recherche Scientifique

Université Mouloud Mammeri de Tizi-Ouzou

Faculté Génie Electrique et Informatique

Département Informatique

Mémoire

de Fin d'Etudes

En vue de l'obtention du Diplôme de master 2 en Informatique
Option : Systèmes informatiques

Thème

Conception et réalisation d'une application mobile avec interconnexion à une base de Données multimédia à distance dans le domaine de la médecine avancée

Réalisé par :

M^elle^ SAIBI Floria

M^elle^ TOUMERT Tassadit

Proposé par :

M^r^ HAMRIOUI

2014-2015

Remerciements

Nous remercions tout d'abord le bon dieu de nous avoir données la force, la volonté et la patience pour l'élaboration de ce travail.

Nous tenons à exprimer notre profonde gratitude et nos sincères remerciements à monsieur HABET, monsieur RAHMANI et madame HADAOUI pour avoir accepté d'honorer par leur jugement notre travail.

Nous tenons aussi à remercier notre promoteur, monsieur HAMRIOUI pour sa confiance aveugle qui nous a donné.

Nos sincères sentiments vont à tous ceux qui, de prés ou de loin, ont contribué à la réalisation de ce projet. En particulier nos chères familles et nos amis (es).

SAIBI&TOUMERT

Dédicace

Je dédie ce modeste travail :

A mes parents, ce travail vous doit beaucoup, qu'il soit pour vous le témoignage de ma reconnaissance infinie pour ces années de compréhension et d'efforts communs.

A mes grandes mères.

A mes frères Syphax et jugurtha et sur tout mohand et ma sœur liticia en leur souhaitant beaucoup de Chances et de réussites.

A tous mes cousins et cousines, oncles et tantes.

A tous mes amis, qu'elles trouvent ici l'expression de mon estime.

A ma famille et à tous ceux qui m'aime et qui ont souhaité ma réussite.

Mary

Dédicace

« Louange à Dieu, le seul et unique »

Je dédie ce modeste travail aux êtres qui me sont les plus
chers au monde; ma chère mère et mon père à qui je dois mon existence et mes succès, Que
Dieu le tout puissant les protège.

A toute ma famille,
Mes très chers frères et sœurs

A mon grand père et mes grandes mères

A Mes Oncles et Tantes
A tous mes amis (es),

A tous ceux que j'aime
Avec l'expression de tous mes sentiments de respect,

Zazie

⋆⋆⋆⋆⋆⋆⋆Sommaire⋆⋆⋆⋆⋆⋆⋆

Chapitre 1 : Généralités

Chapitre 2 : Analyse et Conception.

Chapitre 3 : Réalisation

Liste de figures

Liste de figure

Introduction générale

Introduction générale :

L'informatique est une science qui est impliquée aujourd'hui dans la plupart des domaines de la vie, son utilisation connait de nos jours une grande expansion. Elle intervient dans les différents secteurs tels que l'enseignement, l'administration, le commerce…et le secteur de la santé ne fait pas défaut.

Face à l'évolution rapide des moyens d'informations, de communication et des réseaux de transmission (INTERNET en particulier), une masse importante d'informations multimédias circule tous les jours, ce qui rend leur manipulation difficile. De plus, cette masse de données serait bien inutile sans la capacité de la classer, de l'archiver et d'y accéder rapidement et sélectivement.

Les services multimédia combinent plusieurs formes d'informations (texte, image, audio, vidéo), ainsi ils sont caractérisés par le gros volume de données générées qui doit être pris en charge par les réseaux d'accès et de transport sur lesquels ils sont déployés.

Malgré l'avancement technologique des réseaux cellulaires, le débit offert par ces derniers restes toujours insuffisant pour garantir le bon fonctionnement de tels services, et le coût revient souvent très cher, ce qui rend ce genre de services inaccessibles pour tout le monde.

L'arrivée des nouvelles technologies mobiles tels que les Smartphones, qui sont équipés généralement d'interfaces de communication Wi-Fi et caractérisés par un débit élevé a permis le transfert fiable des fichiers de gros volume.

.

L'informatique prend une place de plus en plus importante au sein du monde médical. On considère généralement que l'application de l'informatique au domaine de la santé est restreinte ou cloisonné à un ensemble de techniques et d'outils.

Notre travail a pour objectif la mise au point d'un système permettant l'échange des données multimédia, à l'aide de la technologie wifi, entre un serveur et des Smartphones. Ce système va permettre aux médecins d'une part de télécharger et visualiser des fichiers multimédia stockés sur le serveur et d'une autre part ils pourront charger des fichiers multimédia vers ce dernier.

Afin de mener à bien notre projet, nous supposons de le structurer en trois chapitres dont :

- Le premier chapitre : « *Généralités* »
- Le deuxième chapitre : « *Analyse et conception* »
- Le troisième chapitre : « *Réalisation* »

.

Chapitre 1

I.1 Introduction :

Aujourd'hui, la majorité des ordinateurs et la quasi-totalité des appareils mobiles (tels que les téléphones portables, les assistants personnels,...) disposent de moyens de connexion à un ou à plusieurs types de réseaux sans fil comme le Wifi, le Bluetooth ou l'Infrarouge. Ainsi, il est très facile de créer en quelques minutes un réseau « sans fil » permettant à tous ces appareils de communiquer entre eux.

Dans ce chapitre nous allons présenter une catégorie particulière de réseaux qui sont les réseaux sans fil, l'architecture client/serveur, ainsi nous allons générer les bases de données multimédia et les services mobiles.

I.2 Généralité sur les réseaux :

I.2.1 Définition d'un réseau :

Un réseau est un ensemble de moyens matériels et logiciels géographiquement dispersés afin d'offrir des services et de permettre le transport de données.

Un réseau est un ensemble des nœuds reliés entre eux par des liens ou canaux de communication dans le but d'échanger des informations.

I.2.2 Classification des réseaux :

A. Selon le type des nœuds, on distingue :

- les réseaux de télécommunication dans lesquels les nœuds sont les stations mobiles, les stations de base (BTS), les contrôleurs des stations de base (BSC), les commutateurs (MSC) ;
- les réseaux informatiques dans lesquels les nœuds sont les ordinateurs, les imprimantes, les routeurs, les switches ou tout autre équipement informatique.

B. Selon l'étendue (la taille), on distingue :

- **PAN :** quelques mètres (une dizaine).
- **LAN :** entre quelques mètres à quelques kilomètres.
- **MAN :** ensemble de l'LAN
- **WAN :** s'étend à l'échelle d'un pays ou de la planète entière.

C. Selon le mode de fonctionnement, on distingue :

- **Commutation de circuits :** dans ce mode, un lien physique est établit par réservation de différent supports physiques afin de constituer une liaison de bout en bout entre une source et une destination.
- **Commutation de message :** ce mode n'établit aucun lien physique entre les deux entités communiquant. Le message est transféré de nœud en nœud et mise en attente si le lieu entre nœud est occupé. Chaque bloc d'information (message) constitue une unité de transfert acheminé individuellement. Le message est mémorisé intégralement par chaque nœud et retransmis en nœud suivant dès qu'un lien se libère (quand le transfert est réalisé le lien est libéré).

Remarque **:** la mémorisation intermédiaire de l'intégralité des messages nécessite des mémoires importantes et augmente le temps de transfert.

- **Commutation de paquets :** un message émis sera découpé en paquets et par la suite chaque paquet est commuté à travers le réseau comme dans le cas de commutation des messages. Les paquets sont envoyés indépendamment les uns des autres, chaque nœud rédige chaque paquet vers la bonne liaison grâce à une table de routage. La reprise par erreur est donc plus simple que la commutation des messages.

Remarque **:** dans ce cas le récepteur final doit être capable de reconstituer le message émis en réassemblant les paquets

D. Selon le type de canaux de communication, on distingue :

- les réseaux filaires : utilisent un canal de transmission matériel (le câble coaxial, les paires torsadées, la fibre optique),
- les réseaux sans fils : sont ceux qui utilisent le canal air pour communiquer en utilisant les ondes hertziennes, les infrarouges ou le laser.

I.3 Les réseaux sans fil :

I.3.1 Définition :

Un réseau sans fil est un réseau dans lequel au moins deux équipements (ordinateur, imprimante, assistant personnel,...) peuvent communiquer sans liaison filaire. Les réseaux sans fil utilisent des ondes radioélectriques à la place des câbles habituels. Il existe plusieurs technologies se distinguant par la fréquence d'émission utilisée ainsi que le débit et la portée des transmissions.

I.3.2 Intérêt des réseaux sans fil :

Grâce aux réseaux sans fil, un utilisateur a la possibilité de rester connecté tout en se déplaçant dans la zone de couverture. On parle alors de mobilité ou d'itinérance. Les réseaux sans fil permettent de relier facilement des équipements d'une distance de quelques dizaines mètres à plusieurs kilomètres. De plus, l'installation de tels réseaux ne demande pas de lourds aménagements des infrastructures existantes, comme c'est le cas avec les réseaux filaires.

I.3.3 Catégories de réseaux sans fil :

On distingue habituellement plusieurs catégories de réseaux sans fil, selon le périmètre géographique offrant la connectivité (appelée zone de couverture), comme le montre la figure ci-après :

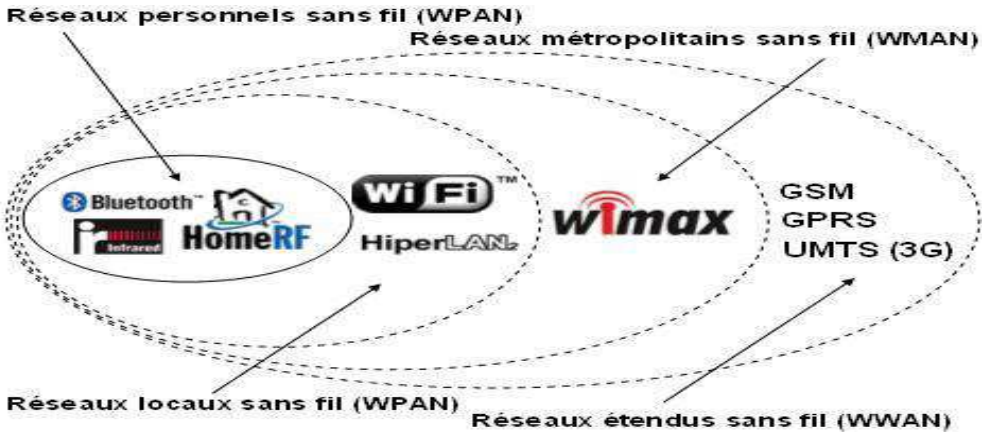

Figure I.1 **:** Catégories des réseaux sans fil.

I.3.3.1 Réseaux personnels sans fil (WPAN) :

Réseau individuel sans fil (Wireless Personal Area Network), Faible portée (quelques dizaines de mètres) Permet la connexion de périphériques (PDA, imprimante,….), d'ordinateurs, ses technologies :

- Bluetooth
- HomeRF
- Zigbee
- infrarouge

I.3.3.2 Réseaux locaux sans fil (WLAN) :

Réseau local d'entreprise (Wireless Local Area Network), Couvre l'équivalent d'un réseau local d'entreprise (100 m), Relie entre eux les équipements présents dans la zone de couverture, ses technologies :

- WiFi
- Hiper LAN

I.3.3.3 Réseaux métropolitains sans fil (WMAN) :

Réseau métropolitain (Wireless Metropolitan Area Network), Plus connu sous le nom de Boucle Local Radio (BLR). Permet à un particulier ou une entreprise d'être relié à son opérateur (téléphonie fixe, Internet, télévision...) via les ondes radio Basé sur la norme 802.16, ses technologies :

- Local Multipoint Distribution Service (LMDS)
- Multichannel Multipoint Distribution Service (MMDS)

I.3.3.4 Réseaux étendus sans fil (WMAN) :

Réseau étendu sans fil (Wireless Wide Area Network) Plus connu sous le nom de « réseaux cellulaire mobile » Utilisé par les téléphones mobiles ses technologies :

- lobGSM : Global System for Mobile communication
- GPRS : General Packet Radio Service

- UMTS : Universal Mobile Telecommunication System

I.4 Le modèle client/serveur :

C'est un modèle informatique basé sur le traitement distribué selon lequel, un utilisateur lance un logiciel client, à partir d'un ordinateur relié à un réseau, déclenchant simultanément le lancement d'un logiciel serveur situé dans un autre ordinateur possédant les ressources souhaitées par l'utilisateur.

I.4.1 Définition de l'architecture client/ serveur :

L'architecture client-serveur est un modèle de fonctionnement logiciel qui peut se réaliser sur tout type d'architecture matérielle (petites ou grosses machines), à partir du moment où ces architectures peuvent être interconnectées**.**

On parle de fonctionnement logiciel dans la mesure où cette architecture est basée sur l'utilisation de deux types de logiciels, à savoir un logiciel serveur et un logiciel client s'exécutant normalement sur deux machines différentes.

L'élément important dans cette architecture est l'utilisation de mécanismes de communication entre les deux applications. Le dialogue entre les applications peut se résumer par :

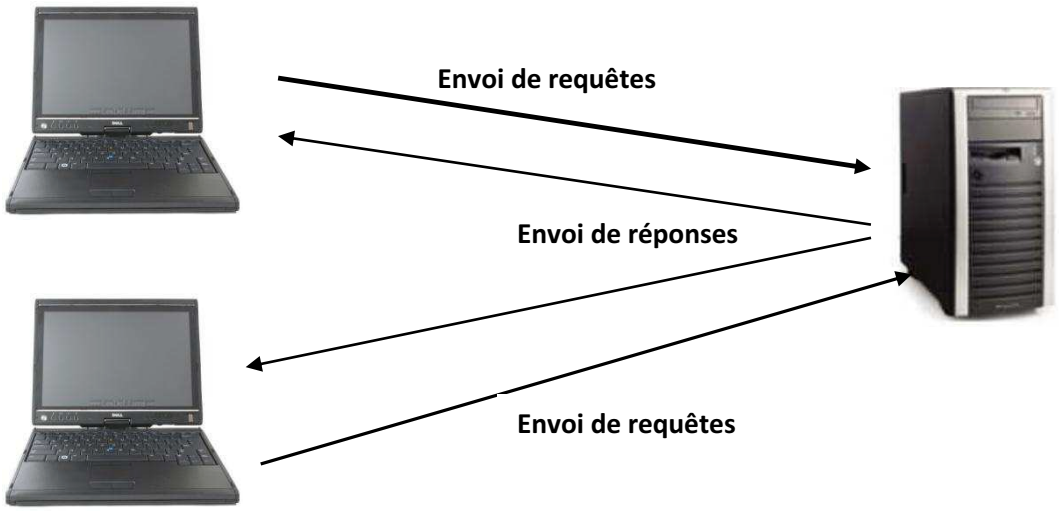

***Figure I.2* :** Le dialogue client /serveur.

- ✓ Le client demande un service au serveur.
- ✓ Le serveur réalise ce service et renvoie le résultat au client.

- **Client :** C'est une entité (processus, programme, ordinateur, etc.) qui demande l'exécution d'une opération à une autre entité par envoie d'un message contenant le descriptif de l'opération à exécuter et attendant la réponse à cette opération par un message en retour.

- **Serveur :** C'est une entité (processus, programme, etc.) qui accomplit une opération sur demande d'un client et transmet la réponse à ce client.

- **Réponse :** message transmis par un serveur à un client suite à l'exécution d'une opération contenant les paramètres de l'opération.

- **Requête :** Message transmis par un client à un serveur décrivant l'opération à exécuter pour le compte du client.

I.4.2 Les architectures Client/serveur :

Le terme « niveau » est utilisé pour décrire un choix architectural grâce à un découpage de la logique applicative entre des clients et des serveurs. Cette répartition des charges a donné naissance à trois modèles d'architectures de base pour le client serveur.

I.4.2.1 L'architecture à deux niveaux :

L'architecture à deux niveaux (aussi appelée architecture 2-tier) : caractérise les systèmes clients/serveurs dans lesquels le *client* demande une ressource et le *serveur* la lui fournit directement. Cela signifie que le serveur ne fait pas appel à une autre application afin de fournir le service.

I.4.2.2 L'architecture à trois niveaux :

Dans l'architecture à 3 niveaux (appelée architecture 3-tier), il existe un niveau intermédiaire, c'est-à-dire que l'on a généralement une architecture partagée entre :

- **Le client** : le demandeur de ressources.
- **Le serveur d'application** : (appelé aussi *middleware*) le serveur chargé de fournir la ressource mais faisant appel à un autre serveur.
- **Le serveur secondaire** : (généralement un serveur de base de données), fournissant un service au premier serveur.

I.4.2.3 L'architecture multi-niveaux :

Dans l'architecture à 3 niveaux, chaque serveur (niveaux 1 et 2) effectue une tâche (un service) spécialisée. Ainsi, un serveur peut utiliser les services d'un ou plusieurs autres serveurs afin de fournir son propre service. Par conséquence, l'architecture à trois niveaux est potentiellement une architecture à N niveaux.

I.4.3 Caractéristiques du Clients/Serveur :

Sachant que le Client/serveur est une forme d'informatique distribuée, il n'est supposé d'appeler système Client/serveur que ceux qui partagent les caractéristiques suivantes :

- **Le service :** c'est le travail fourni par le serveur suite à la requête du client. Le serveur est un fournisseur de services et le client consommateur de ces services. Le modèle Client/serveur établit ainsi une séparation claire des rôles à partir de la notion du service.
- **Les ressources partagées :** le serveur est capable de servir de nombreux clients simultanément et réguler leur accès.
- **Les protocoles asymétriques :** la relation entre clients et serveur est de type « plusieurs vers un », toutefois le client est le déclencheur du dialogue en demandant un service alors que le serveur attend passivement les requêtes.
- **La transparence à la localisation :** le serveur peut résider sur la même machine que le client ou, par l'intermédiaire d'un réseau, sur une machine différente. Le logiciel Client/serveur masque aux clients la localisation du serveur en redirigeant les demandes de services si nécessaire.
- **La compatibilité :** l'application Client-serveur est indépendante du matériel.
- **L'échange de messages :** clients et serveurs sont des systèmes qui interagissent au moyen de messages. Le message est un mécanisme d'émission de demandes de services et de réponses à celles-ci.
- **L'adaptabilité :** des machines peuvent être ajoutées ou retirées du réseau ; les performances des machines peuvent aussi évoluer.

I.5 Les bases de données multimédia :

I.5 .1 Définition d'une base de données multimédia :

Par rapport à un autre type de base de données, une base de données multimédia est une base de données qui peut héberger un ou plusieurs types de données sous différents formats multimédias tels que fichiers sons, photos graphiques, séquences vidéo, données textes, …

I.5.2 Stockage de données multimédia :

Les différents documents (texte, image, audio, vidéo) peuvent être stockés soit à l'intérieur de la base de données ou n'être que référencés par celle-ci. Le choix de stockage dépend de l'application.

A l'extérieur de la base de données l'objet est stockés dans un dossier et dans la base de données on stocke que le chemin vers ce dossier exemple Images/image.jpg telle que Images est le dossier qui contient image.

A l'intérieure de la base de données, les objets sont stockés dans la base de données elle-même et pour ça on utilise le type LOBs (en anglais Large Objects) pour stocker les objets de grande taille, on distingue :

- **BLOBs :** Ce sont des objets binaires de grande taille (Binary Large Objects) utilisés pour le stockage des données multimédias : Images, sons, vidéo.
- **CLOBs :** Ce sont Objets de grande taille de type caractère utilisés pour le stockage des documents : PDF, WORD, XML, TXT...
- **BFILEs :** Ce type permet de stocker des références pour des données externes à la BDD (en général des fichiers du système d'exploitation).

I.6 Développement d'une application mobile :

Pour développer une application il est nécessaire de définir :

I.6.1 Les langages pour le développement :

En fonction de la plateforme utilisée, le langage de programmation sera différent :

- Les applications mobiles Android sont majoritairement développées en *Java*.
- Les applications iOS sont principalement réalisées avec de l'objectives (une sorte de surcouche du C, spécifique à Apple), qui sera bientôt remplacé par le langage Swift. Sur

Windows Phone, on peut développer en *C#* (développement dit « natif ») ou en HTM5/ JavaScript (mode dit « hybride »).

***Remarque** :* Ces choix sont principalement guidés par les outils de développement disponibles, mais il est cependant possible de développer avec d'autres langages, par exemple en Python sur Android.

I.6.2 Les terminaux mobiles :

I.6.2.1 Les terminaux PC mobiles :

Les PC portables *(laptops)* sont des terminaux conçus pour des applications informatiques, Par conséquent, ils présentent des capacités similaires voire même plus élevées que celles des PC de bureau.

Les terminaux laptops ont une plus grande capacité de traitement et de mémoire, leurs écrans sont de grande dimension. Ils sont équipés de systèmes d'exploitation et des interfaces utilisateurs conviviales et des interfaces d'accès aux réseaux fixes (LAN, Internet, etc.) tels qu'une carte réseau et un modem téléphonique, en plus des interfaces pour les périphériques (écrans externes, périphériques USB…).

Actuellement, les terminaux laptops sont équipés des interfaces de connectivité sans fil (WIFI, Bluetooth, etc.), Ces interfaces permettent des applications de type bureau mobile, l'accès à l'Internet mobile et en fin des applications VOIP (Voice IP).

Ces terminaux ont des contraintes de poids et d'autonomie d'énergie, en effet, ils sont gourmands en énergie et leurs batteries ne permettent que de 2 à 9 heures de travail (en utilisant une seconde batterie de réserve).

I.6.2.2 Les assistants personnels (PDA) et les Smartphones :

Les terminaux PDA sont un compromis entre la portabilité (en termes de dimension et de poids) et la puissance de calcul. Un PDA est un ordinateur de poche composé d'un processeur, de mémoire vive, d'un écran tactile avec un stylet et de fonctionnalités réseau (Buletooth, WiFi, etc.), le tout mis dans un boîtier compact d'une petite taille. Les PDA possèdent généralement les possibilités de communication classique (téléphone, navigation internet) mais ils sont plus adaptés à l'organisation des tâches quotidiennes (gestion des rendez-vous, des contacts personnels, des dépenses, etc.).

Un **Smartphone**, également appelé **Ordiphone**, **téléphone intelligent** ou **Intelliphone**, est un téléphone mobile disposant aussi des fonctions d'un PDA. Il peut aussi fournir les fonctionnalités d'agenda/calendrier, de navigation web, de consultation de courrier électronique, de messagerie instantanée, de GPS, etc.

Actuellement il n'y a presque pas de différence entre un Smartphone et un PDA.

I.6.3 Les systèmes d'exploitation :

Il existe plusieurs systèmes d'exploitation spécifiques aux Smartphones. Selon l'article publié en 2013 par l'International Data Corporation (IDC) les systèmes d'exploitation les plus utilisés parmi les autres sont Android, iOS et Windows Phone. L'usage des systèmes Android et iOS était dominant constituent 92.3% des ventes mondialement à la fin de 2013.

I.6.3.1 Android :

Android est un système d'exploitation en partie Open Source pour Smartphones. Dans le cas de notre travail nous nous intéressons plus particulièrement à ce système d'exploitation donc on va détailler beaucoup plus.

I.6.1.2 iOS :

iOS, anciennement iPhone OS, est le système d'exploitation mobile développé par Apple pour l'iPhone, l'iPod touch et l'iPad. Il est dérivé de Mac OS X dont il partage les fondations, le système d'exploitation occupe moins d'un demi-gigaoctet (Go) de la capacité mémoire totale de l'appareil, ce qui est plutôt un avantage étant donné que les iPhones ne disposent pas d'emplacement de carte SD.

I.6.1.3 Windows Mobile :

Windows Mobile est le nom générique donné à différentes versions de Microsoft Windows conçues pour des appareils mobiles tels que les Smartphones ou les Pockets PC. Ces systèmes d'exploitation permettent à des logiciels Microsoft tels que Microsoft Office ou Windows Live Messenger de fonctionner sur un téléphone ou un ordinateur portable.

L'avantage majeur est de pouvoir recevoir des courriels en temps réel. En 2008, Microsoft a vendu 20 millions de licences Windows Mobile aux constructeurs d'assistant personnel (PDA).

1.7 Android :

Android été développé par la startup Android Inc. en 2003, puis racheté par Google en 2005. Pour pouvoir réalisé un système complet, ouvert et gratuit dans le monde du mobile, une coalition de 35 entreprise évoluant dans l'univers du mobile, dont Google, a été créé. Ce rassemblement se nommé l'Open Handset Alliance (OHA) et composé aujourd'hui de 80 membres. Cette alliance a pour but de développer un système open source ‹‹ c'est-à-dire dont les sources sont disponible librement sur internet ›› pour l'exploitation sur mobile.

Android est à l'heure actuelle le système d'exploitation pour smartphones et tablettes le plus utilisé. Les terminaux visés par Android inclus les téléphones portables, Netbook/Smartbook, tablettes multimédia, automobile, GPS, Réfrigérateur, etc.

I.7.1 Les Versions :

Android a débuté avec la sortie de la version 1.0 en septembre 2008. Android a connu plusieurs mises à jour depuis sa première version. Ces mises à jour servent généralement à corriger des bugs et à ajouter de nouvelles fonctionnalités. Dans l'ensemble, chaque version est développée sous un nom de code basé sur des desserts. Ces nom de codes suivent une logique alphabétique, en voici quelques-unes :

- **1.0 :** Apple Pie (Tarte aux pommes) ou Alpha, version connue uniquement ou presque des développeurs car c'est la version du Sdk distribuée avant la sortie du premier téléphone Android, Sdk distribué fin 2007 ;
- **1.5 :** Cupcake (Petit Gâteau), sortie en avril 2009, dernière révision officielle en mai 2010;
- **2.3 (2.3.7) :** Gingerbread (Pain d'épice), sortie le 6 décembre 2010, dernière version dédiée uniquement aux Smartphones. Cette version est parfois utilisée sur de petites tablettes.
- **4.2 :** (Aussi appelée Jelly Bean) introduit photo sphère permettant une prise des photos à 360° type Street View, un système multi-compte sur tablette uniquement

I.7.2 Architecture d'Android :

L'environnement de développement est la base sur une architecture autour du noyau Linux. La plateforme Android est compose de cinq couches principales :

- Un noyau Linux qui lui confère des caractéristiques multitâches.

- Des bibliothèques graphiques, multimédias.
- La Dalvik Virtuel Machine, une machine virtuelle adaptée pour java.
- Une plateforme applicative pour la gestion des fenêtres, du contenu, de téléphonie, etc.
- Des applications.

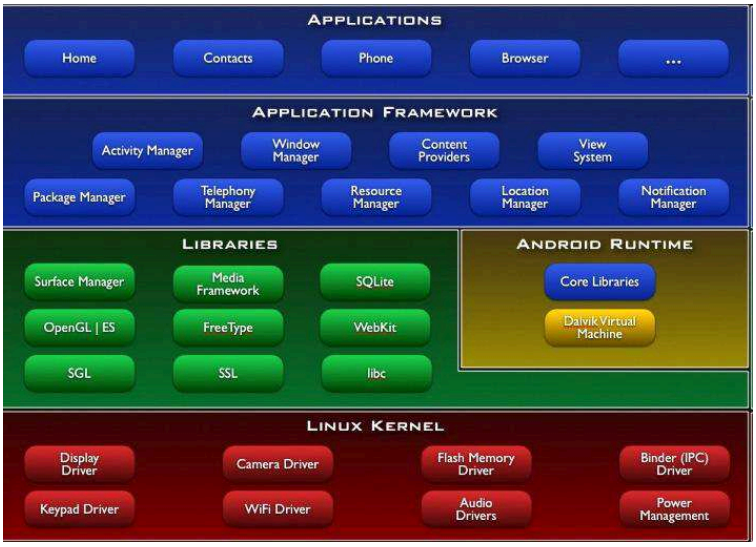

Figure I.3 : L'Architecture de la plateforme Android.

Pour pouvoir développer sur la plateforme Android, il faut accepter la licence Apache 2.0 associe lorsque vous téléchargez le développement kit. La licence autorise la modification sous forme libre ou non et permet d'en faire un usage commercial (car il est open source).

Android offre un système standard de téléchargement d'applications sur les Play Store. Les applications sont classées sur Play Store selon différents critères comme par exemple: catégorie d'âge. Le Play Store offre aussi une possibilité de rendre les applications payantes. Pour mettre une application sur Play Store, il suffit de payer $25 ce qui permet de publier autant d'applications que vous le souhaitez à vie.

I.7.2 Un projet Android :

I.7.2.1 Contenu d'un programme Android :

a. Activity (Activité) :

La brique de base de l'interface utilisateur s'appelle *activity* (activité). Vous pouvez la considérer comme l'équivalent Android de la fenêtre ou de la boîte de dialogue d'une application classique.

Bien que des activités puissent ne pas avoir d'interface utilisateur, un code "invisible" sera délivré le plus souvent sous la forme de fournisseurs de contenus (*content provider*) ou de services.

b. Content providers (fournisseurs de contenus) :

Les fournisseurs de contenus offrent un niveau d'abstraction pour toutes les données stockées sur les terminales et accessibles aux différentes applications. Le modèle de développement Android encourage la mise à disposition de ses propres données aux autres programmes –construire un *content provider* permet d'obtenir ce résultat tout en gardant un contrôle total sur la façon dont on accédera aux données.

c. Intents (intentions) :

Les intentions sont des messages système. Elles sont émises par le terminal pour prévenir les applications de la survenue de différents événements, que ce soit une modification matérielle (comme l'insertion d'une carte SD) ou l'arrivée de données (telle la réception d'un SMS), en passant par les événements des applications elles-mêmes (votre activité a été lancée à partir du menu principal du terminal, par exemple). Vous pouvez non seulement répondre aux intentions, mais également créer les vôtres afin de lancer d'autres activités ou pour vous prévenir qu'une situation particulière a lieu (vous pouvez, par exemple, émettre l'intention X lorsque l'utilisateur est à moins de 100 mètres d'un emplacement Y).

d. Services :

Les activités, les fournisseurs de contenus et les récepteurs d'intentions ont une durée de vie limitée et peuvent être éteints à tout moment. Les services sont en revanche conçus pour durer et, si nécessaire, indépendamment de toute activité. Vous pouvez, par exemple, utiliser un service pour vérifier les mises à jour d'un flux RSS ou pour jouer de la musique, même si l'activité de contrôle n'est plus en cours d'exécution.

I.7.2.2 Structure d'un projet :

Le système de construction d'un programme Android est organisé sous la forme d'une arborescence de répertoires spécifique à un projet, exactement comme n'importe quel projet Java. Les détails, cependant, sont spécifiques à Android et à sa préparation de l'application qui s'exécutera sur le terminal ou l'émulateur.

La création d'un projet Android place plusieurs éléments dans le répertoire racine du projet :

- **AndroidManifest.xml** est un fichier XML qui décrit l'application à construire et les composants – activités, services, etc. – fournis par celle-ci.
- **build.xml** est un script Ant1 permettant de compiler l'application et de l'installer sur le terminal (ce fichier n'est pas présent avec un environnement de programmation adapté, tel Eclipse).
- **default.properties** et **local.properties** sont des fichiers de propriétés utilisés par le script précédent.
- **bin/** contient l'application compilée.
- **gen/** contient le code source produit par les outils de compilation d'Android.
- **libs**/ contient les fichiers JAR extérieurs nécessaires à l'application.
- **src/** contient le code source Java de l'application.
- **res/** contient les ressources – icônes, descriptions des éléments de l'interface graphique (*layouts*), etc. – empaquetées avec le code Java compilé.
- **tests/** contient un projet Android entièrement distinct, utilisé pour tester celui que vous avez créé.
- **assets/** contient les autres fichiers statiques fournis avec l'application pour son déploiement sur le terminal.

I.7.2.3 Contenu du manifeste :

Le point de départ de toute application Android est son fichier manifeste, Android Manifest.xml, qui se trouve à la racine du projet. C'est dans ce fichier que l'on déclare ce que contiendra l'application – les activités, les services, etc. On y indique également la façon dont ces composants seront reliés au système Android lui-même en précisant, par exemple, l'activité (ou les activités) qui doivent apparaître dans le menu principal du terminal (ce menu est également appelé "lanceur" ou "launcher").

La création d'une application produit automatiquement un manifeste de base. Pour une application simple, qui offre uniquement une activité, ce manifeste automatique conviendra sûrement ou nécessitera éventuellement quelques modifications mineures. En revanche, le manifeste de l'application de démonstration Android fournie contient plus de 1 000 lignes. Vos applications se situeront probablement entre ces deux extrémités. AndroidManifest.xml sert à :

- Préciser le nom du package java utilisant l'application. Cela sert d'identifiant unique !
- Il décrire les composants de l'application
- Lister des activités, services, broadcast receivers
- Préciser les classes qui les implémentent
- Précise leurs capacités (`a quels intents ils réagissent), Ceci permet au système de savoir comment lancer chaque partie de l'application afin de satisfaire au principe de réutilisabilité.
- Définir les permissions de l'application :
  • Droit de passer des appels
  • Droit d'accéder `a Internet
  • Droit d'accéder au GP
- Préciser la version d'Android minimum nécessaire
- Déclarer les librairies utilisées
- Déclarer des outils d'Instrumentation (uniquement pour le développement)

I.7.2.4 Cycle de vie d'une activité (ActivityLife cycle) :

Pour développer une application sur Android, on doit comprendre le cycle de vie d'une activité.

- **L'état Active/courant (Running) :** C'est un état que l'activité marche en avant
- **L'état Paused (il est en pause)** : Cette activité est visible mais elle n'est pas active.
- **L'état Stopped :** Cette activité n'est pas visible. Si une activité est complètement masquée par une autre activité, elle est arrêtée et conservée tous les états. Cependant elle n'est plus visible pour l'utilisateur, sa fenêtre est cachée et elle sera souvent tuée par le système lorsque la mémoire est nécessaire ailleurs.
- **L'état Dead :** Cette activité est terminée ou elle n'a jamais été démarrée. Si une activité est en pause ou arrêtée, le système peut supprimer l'activité de la mémoire, soit par lui demandant de se terminer, ou tout simplement tuer le processus. Quand

il est affiché de nouveau à l'utilisateur, il doit être redémarré et restauré à son état antérieur.

Il existe trois boucles principales :

- La durée de vie d'une activité se passe entre le premier appel à **onCreate ()** par l'appel à **onDestroy ()**. Une activité met en place tous les états globaux dans la méthode **onCreate ()** et libère toutes les ressources restantes à **onDestroy ()**.
- La durée de vie visible d'une activité se passe entre un appel à **onStart ()** jusqu'à ce qu'un appel correspondant à **onStop ()**. Dans ce temps, l'utilisateur peut voir l'activité sur l'écran, même s'elle n'est pas à l'avant et à l'interaction avec l'utilisateur. Entre ces deux méthodes, les ressources qui sont nécessaires pour montrer l'activité de l'utilisateur sont conservées.
- La durée de vie d'une activité en avant-plan se passe entre un appel à **onResume ()** jusqu'à ce qu'un appel correspondant à **onPause ()**. Dans ce temps, l'activité est en face de toutes les autres activités afin d'interagir avec l'utilisateur. Une activité peut souvent changer son état entre l'état de reprise et l'état en pause.

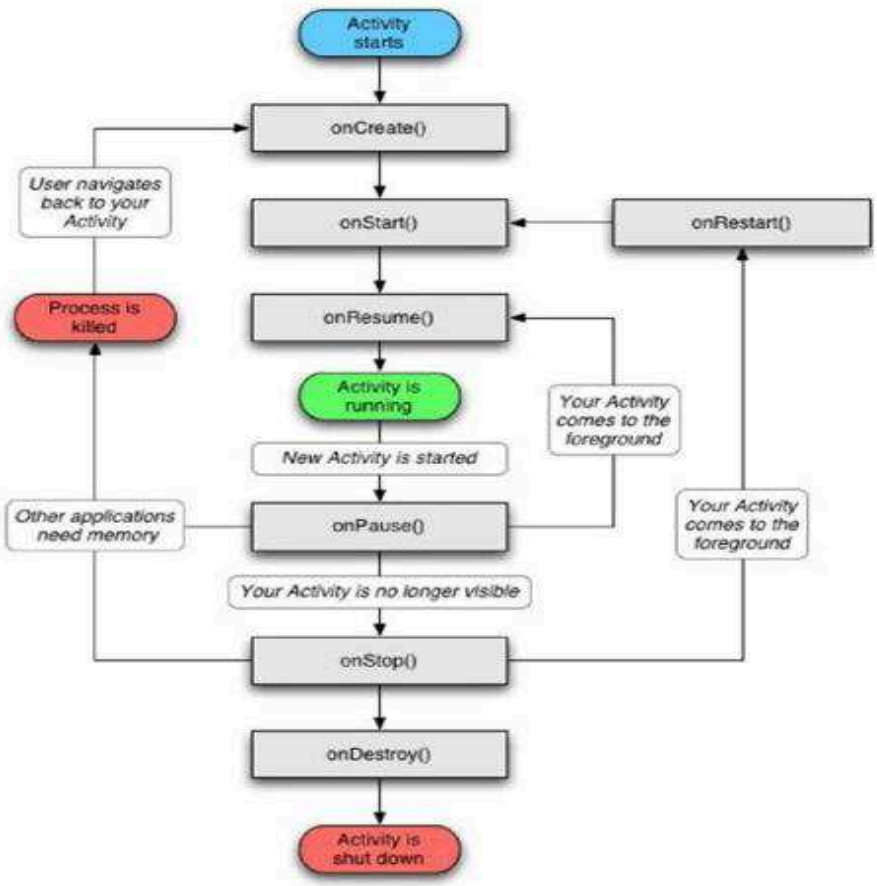

Figure I.4 : Cycle de vie d'une activité.

I.7.2.5 Les fichiers XML sous Android :

XML (eXtensible Markup Language, soit « Langage de balisage extensible ») est un langage de balisage définissant un format universel de représentation des données, permettant de décrire la structure hiérarchique d'un document.

Un document XML contient à la fois les données et les indications sur le rôle que jouent ces données, ces indications (ou balises) permettent de déterminer la structure des documents. Lorsqu'on crée un projet Android, un dossier res/layout est créé d'office, dans ce dernier on trouve les fichiers qui déterminent l'interface graphique de chaque activité (son architecture et les vues qu'elle contient), bien sur la plupart de ce qui peut être fait avec des fichiers de positionnement XML peut également être réalisé avec du code Java. Ces fichiers s'ajoutant à tous ceux que vous devez déjà gérer, il faut donc qu'il y ait de bonnes raisons de les utiliser :

- Ils permettent de créer des outils de définition des vues.

- Il est plus simple de relire et modifier la définition d'une interface graphique exprimée dans un format structuré comme XML plutôt que codée dans un langage de programmation.
- Séparer ces définitions XML du code Java réduit les risques qu'une modification du code source perturbe l'application.

De ce fait l'utilisation de XML pour la définition des interfaces graphiques est devenue monnaie courante.

I.7.2.6 Le résultat d'un projet créé :

Lorsque vous compilez un projet (avec ant ou un IDE), le résultat est placé dans le répertoire bin/, sous la racine de l'arborescence du projet :

- bin/classes/ contient les classes Java compilées.
- bin/classes.dex contient l'exécutable créé à partir de ces classes compilées.
- bin/votreapp.ap_ (où votreapp est le nom de l'application) contient les ressources de celle-ci, sous la forme d'un fichier ZIP.
- bin/votreapp-debug.apk ou bin/votreapp-unsigned.apk est la véritable application Android.

Le fichier .**apk** est une archive ZIP contenant le fichier **.dex**, la version compilée des ressources (resources.arsc), les éventuelles ressources non compilées (celles qui se trouvent sous res/raw/, par exemple) et le fichier AndroidManifest.xml. Cette archive est signée : la partie -debug du nom de fichier indique qu'elle l'a été à l'aide d'une clé de débogage qui fonctionne avec l'émulateur alors que -unsigned précise que l'application a été construite pour être déployée (ant release) : l'archive APK doit alors être signée à l'aide de jarsigner et d'une clé officielle.

I.8 Conclusion :

A travers ce chapitre nous avons vu que le modèle client/serveur est devenu l'architecture d'application la plus utilisée. Nous retenons aussi que les réseaux sans fil offrent des avantages remarquables en évitant le câblage souvent difficile à mettre en œuvre et à maintenir, et que les services mobiles ont connu une évolution remarquable dans le monde. Et puisque dans notre cas nous nous intéressons plus particulièrement au transfert de données multimédia, on a donné aussi quelques notions qui concernent ce domaine.

Le chapitre prochain sera consacré pour l'analyse et conception

Chapitre 2

II.1 Introduction :

La conception de toute solution informatique doit être traitée avec rigueur et précision, car elle constitue la base du système à développer. Avant de s'engager dans la conception, il est impératif de passer par la phase d'analyse qui permet d'identifier les différents acteurs qui interagissent avec le système ainsi que leurs besoins. Puis on passera à la conception qui utilisera les résultats de la phase d'analyse, on donnera aussi la description détaillée du système cible et les objectifs à atteindre. Pour ce faire, notre démarche va s'appuyer sur le langage UML, qui offre une bonne représentation des aspects statique et dynamique d'une application.

II.2 Le langage de modélisation UML :

UML est un langage de modélisation fondé sur les concepts objet standard conçu pour l'écriture de plans d'élaboration de logiciels. L'objectif d'UML est de fournir une notation standard utilisable dans le développement de systèmes informatiques basés sur l'objet.

UML est adapté à la modélisation de systèmes, depuis les systèmes informatiques d'entreprise jusqu'aux applications distribuées basées sur le Web, c'est un langage très expressif qui couvre toutes les perspectives nécessaires au développement et au déploiement de tels systèmes. Pour apprendre à s'en servir efficacement, il faut d'abord s'appuyer sur une représentation conceptuelle de ce langage, ce qui nécessite l'assimilation de trois éléments fondamentaux qui sont :

Les briques de base d'UML, (Des éléments. Des relations. Des diagrammes) et les règles qui déterminent la manière de les assembler et quelques mécanismes généraux qui s'appliquent à ce langage.

UML s'articule autour de neuf diagrammes différents, chacun d'eux étant dédié à la représentation des concepts particuliers d'un système logiciel. Par ailleurs, UML modélise le système suivant deux modes de représentation : l'un concerne la structure du système pris "au repos" **modèle statiques** (diagramme de cas d'utilisation, de classe…etc.) l'autre concerne sa **dynamique** de fonctionnement (diagramme de séquence, d'états transitions, d'activités…etc.). Les deux représentations sont nécessaires et complémentaires pour schématiser la façon dont est composé le système et comment ses composantes fonctionnent entre elles.

II.3 Analyse :

Cette partie a pour objectif la spécification d'une manière claire de notre application. Pour ce faire, il est nécessaire de déterminer globalement ce que ce trouve dans le champ de l'application en s'intéressant à la définition des besoins ainsi que les interactions entre les différents acteurs impliqués.

II.3.1 Identification des besoins :

Notre projet consiste à mettre en œuvre un service multimédia, qui a pour objectif d'assurer aux utilisateurs (médecins) le chargement, le téléchargement, ainsi que la lecture des fichiers multimédia à partir du serveur via un Smartphone.

Dans le but d'une meilleure organisation et une bonne maîtrise du travail, tout processus de développement d'applications ou systèmes informatiques doit suivre une méthode ou une démarche bien définie.

La figure ci-dessous montre la représentation graphique de la démarche de modélisation choisie pour concevoir notre application :

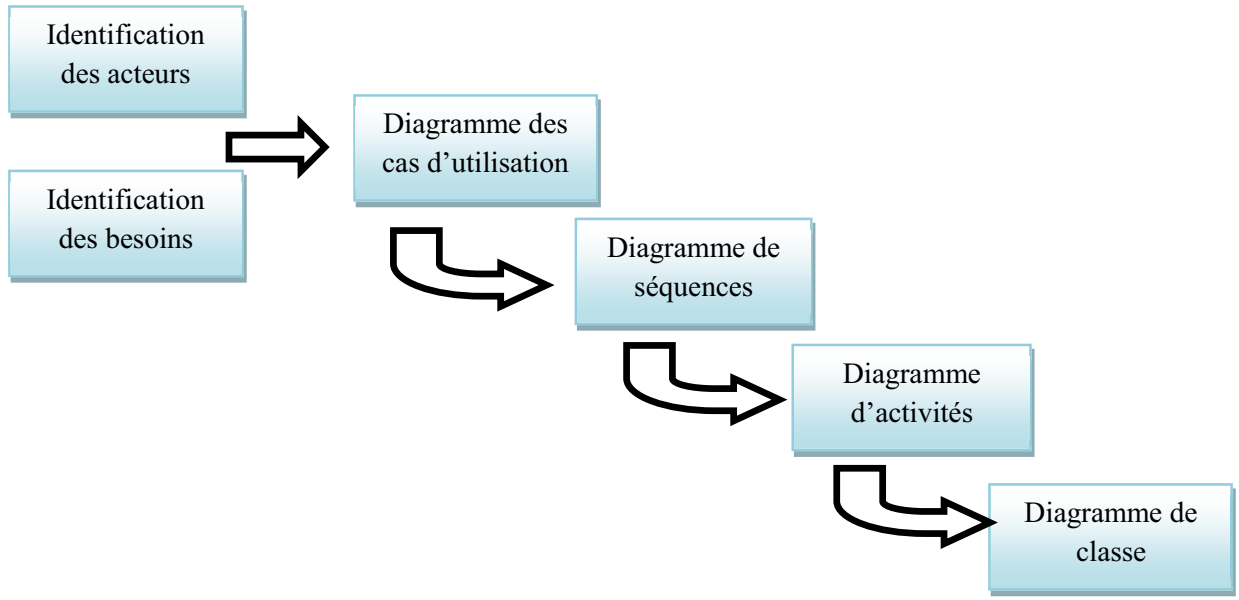

Figure II.1 : La démarche de modélisation de l'application.

II.3.2 Identification des acteurs :

Après avoir étudié les besoins de notre système, nous avons procédé à l'identification de ces principaux acteurs :

- **Utilisateur :** Tout médecin disposant d'un compte qui peut charger, télécharger, ou lire un fichier multimédia, c.à.d. il a comme tâches :
  - S'authentifier par un identifiant et un mot de passe.
  - Charger des fichiers multimédia.
  - Télécharger des fichiers multimédia.
  - Lire des fichiers multimédia.

Remarque : L'utilisateur est le médecin.

- **Administrateur :** celui qui gère les utilisateurs, il effectue les tâches suivantes :
  - S'authentifier par un identifiant et un mot de passe.
  - Gestion des utilisateurs (Ajout, modification, suppression).
  - Gestion des maladies (Ajout, modification, suppression).

II.3.3 Diagramme de contexte :

Le diagramme de contexte est un modèle conceptuel de flux qui permet d'avoir une vision globale des interactions entre le système et l'environnement extérieur, notre diagramme de contexte est donné par la figure suivante :

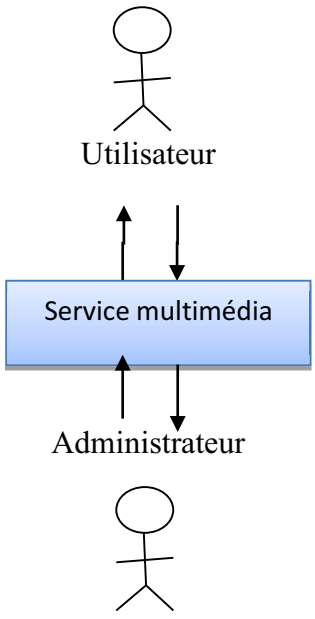

Figure II.2 : Diagramme de contexte de l'application.

II.3.4 Les cas d'utilisation :

Un cas d'utilisation représente un ensemble de séquence d'actions qui sont réalisées par les systèmes et qui produit un résultat observable intéressant pour un acteur particulier. Il permet de décrire ce que le système devra faire, sans spécifier comment le faire.

Dans notre cas nous distinguons les cas d'utilisation suivants :

II.3.4.1 Cas d'utilisation relatifs à l'administrateur :

La figure suivante montre le cas d'utilisation général de l'administrateur

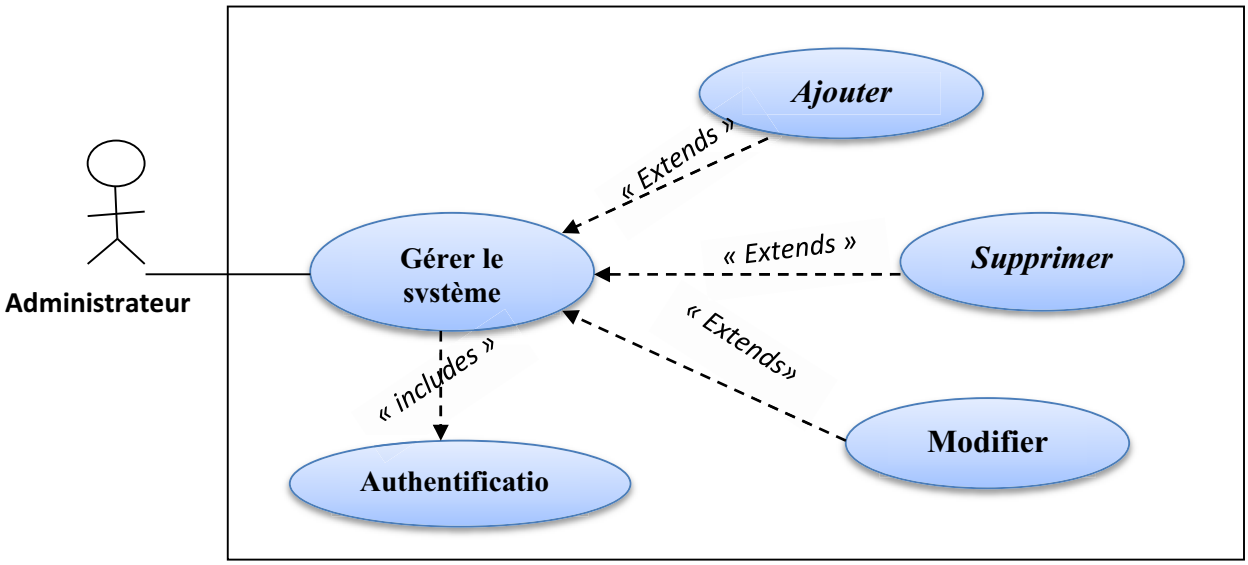

Figure II.3 : Diagramme de cas d'utilisation général de l'administrateur.

II.3.4.2 Cas d'utilisation relatifs à l'utilisateur :

La figure suivante montre le cas d'utilisation général de l'utilisateur :

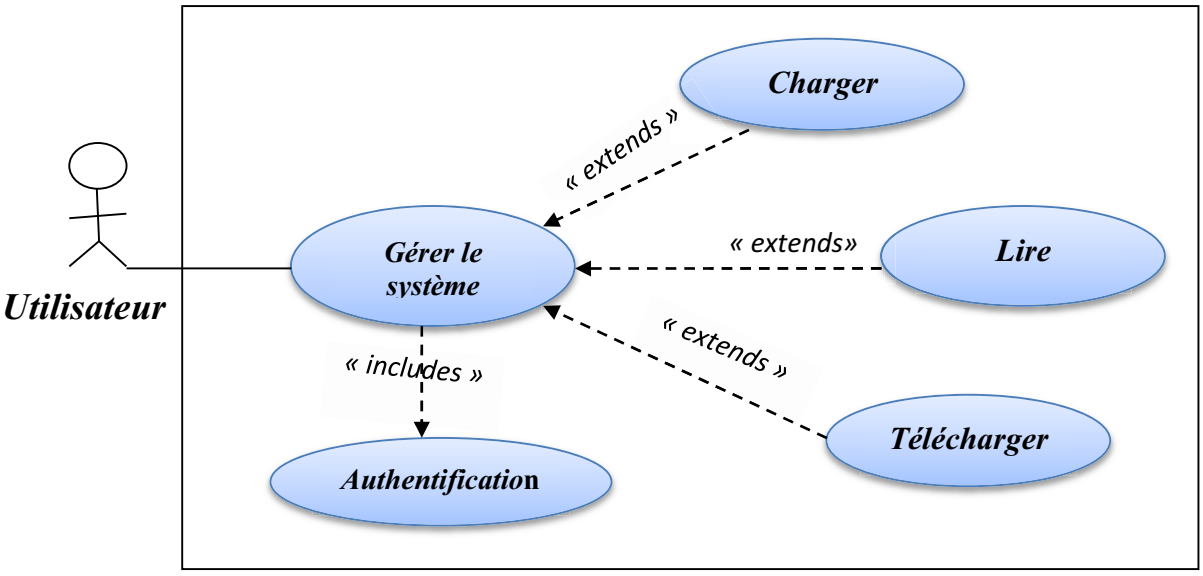

Figure II.4 : Diagramme de cas d'utilisation général de l'utilisateur.

II.3.5 Description des cas d'utilisation avec des scénarios :

Pour détailler le déroulement d'un cas d'utilisation, la procédure la plus évidente consiste à recenser de façon textuelle toutes les interactions entre les acteurs et le système. Dans ce qui suit nous décrivons donc quelques cas d'utilisation de notre système.

II.3.5.1Cas d'utilisation «Authentification Administrateur » :

Use Case : Authentification

Acteur : Administrateur

Description :

1. Atteindre la page «Authentification »
2. Saisie les informations d'identification (Pseudo, mot de passe)
3. Le système affiche un message d'erreur d'identification si les informations saisies sont incorrectes ou si elles ne correspondent pas à l'administrateur
4. Affichage de la page principal de l'espace administrateur

Figure II.5 : Description du cas d'utilisation «Authentification Administrateur »

II.3.5.2 Cas d'utilisation « Ajouter utilisateur coté administrateur » :

Use Case : Ajouter utilisateur

Acteur : Administrateur

Description

1. Cliquer sur le bouton «Gérer les médecins»
1. Cliquer sur le bouton «Ajouter médecin»
2. Remplir le formulaire d'ajout et valider
3. Le system affiche une erreur si les champs obligatoire n'est pas remplis
4. Sinon une page de confirmation est affichée suivie de l'identifiant, nom d'utilisation et un mot de passe concernant le médecin qui vient d'être ajouté

Figure II.6 : Description du cas d'utilisation « Ajout d'un utilisateur ».

II.3.5.3 Cas d'utilisation «Authentification coté utilisateur » :

Use Case : Authentification

Acteur : Utilisateur

Description :

1. Atteindre la page «Authentification »
2. Saisie des informations d'identification (nom d'utilisateur, mot de passe)
3. Le système affiche un message d'erreur d'authentification si les informations saisies sont incorrectes ou si elles ne lui correspondent pas
4. Affichage de la page principal

Figure II.7 **:** Description du cas d'utilisation « Authentification utilisateur ».

II.3.5.4 Cas d'utilisation «Télécharger une image coté utilisateur » :

Use Case : Télécharger une image

Acteur : Utilisateur

Description :

Télécharger un fichier

Télécharger un fichier

1. L'utilisateur atteint la page principale (les types de fichier).
2. L'utilisateur sélectionne le bouton « Image».
3. Le système affiche la page de choix de type gestion de l'mage (charger ou rechercher).
4. L'utilisateur sélectionne rechercher.
5. Le système affiche la page de choix de critère de recherche.
6. L'utilisateur sélectionne un critère soit nom de maladie.
7. Le système affiche une liste des images qui correspondent au critère de recherche.
8. L'utilisateur clique sur une l'image qu'il veut.
9. Le système affiche la page de téléchargement dans laquelle il agrandit l'image choisie.
10. L'utilisateur clique sur télécharger.
11. Le système envoie un message de confirmation.

Figure II.8 : Description du cas d'utilisation « Téléchargement d'une image ».

II.4 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.

II.4.1 Les diagrammes de séquence :

Le diagramme de séquence représente la succession chronologique des opérations réalisées par un acteur : saisir une donnée, consulter une donnée, …, il met en évidence les objets manipulés ainsi que les opérations qui font passer d'un objet à l'autre. Dans notre cas on

s'intéresse seulement à effectuer la représentation des diagrammes de séquence pour les cas d'utilisation déjà présentés auparavant.

II.4.1.1 Diagramme de séquence «Authentification Administrateur» :

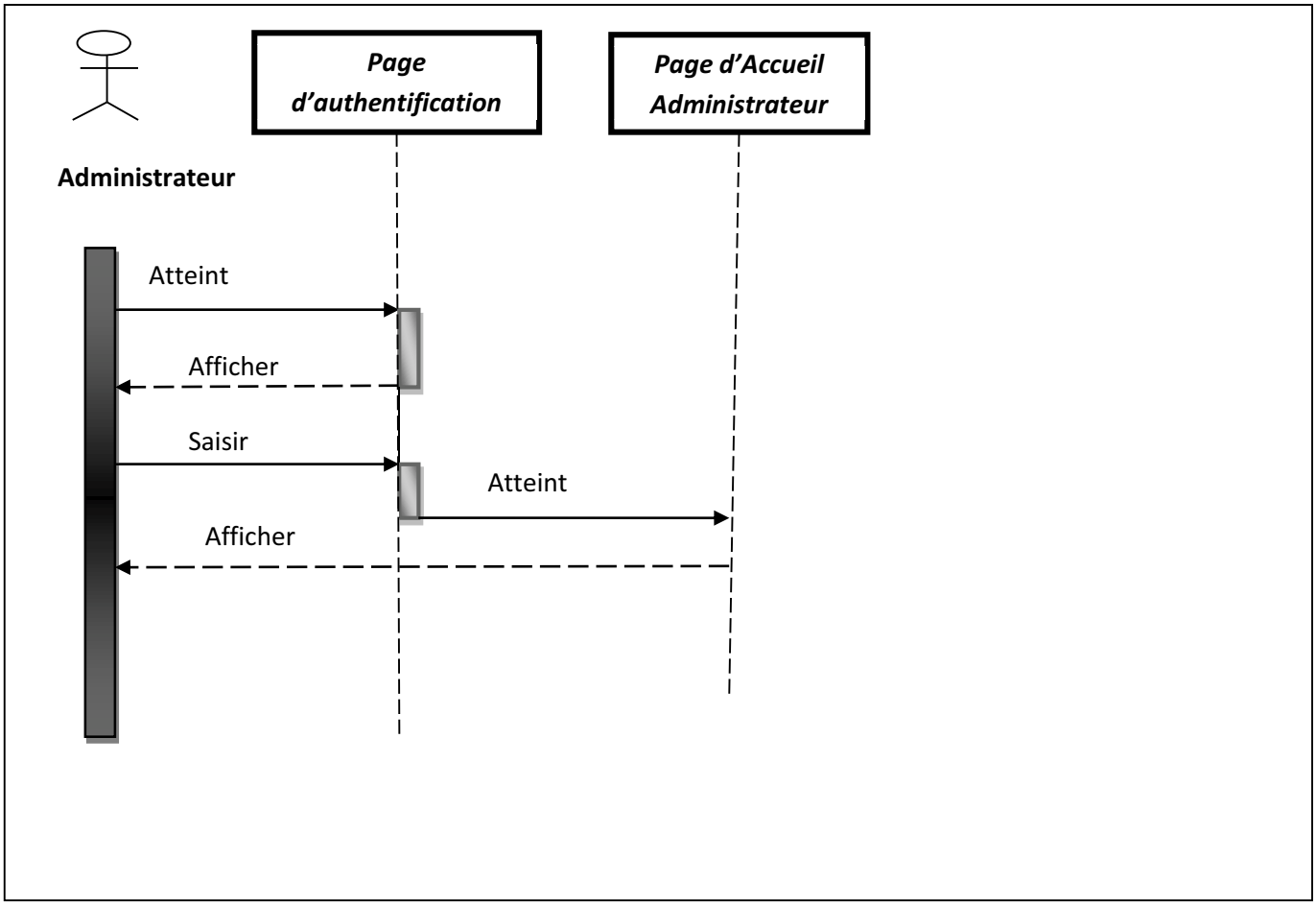

Figure II.9 : Diagramme de séquence de cas utilisation : « Authentification utilisateur ».

II.4.1.2 Diagramme de séquence «Ajouter un utilisateur par l'administrateur » :

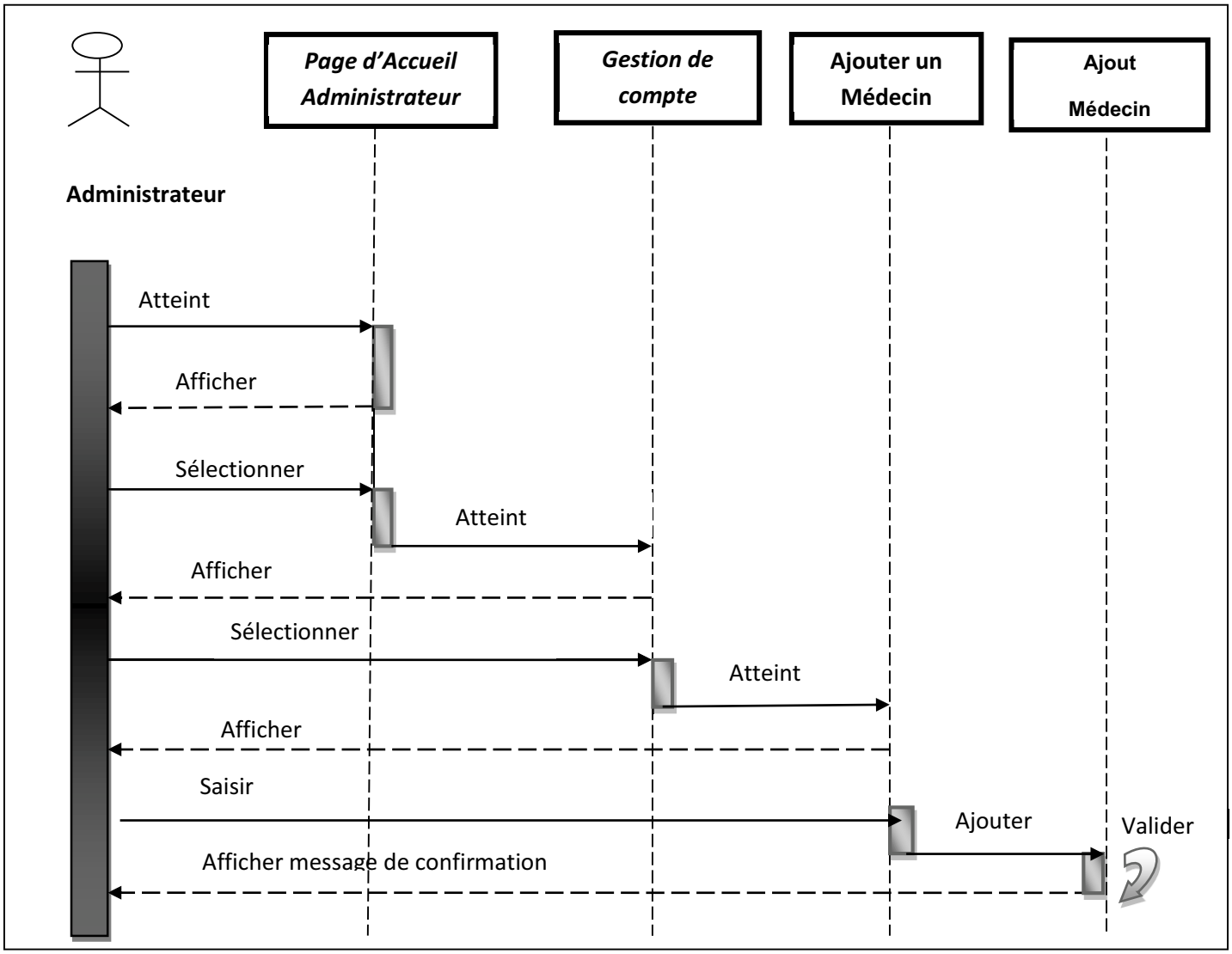

Figure II.10 : Diagramme de séquence : « Ajout d'un médecin par l'administrateur **».**

II.4.1.3 Le diagramme de séquence «Télécharger une image» :

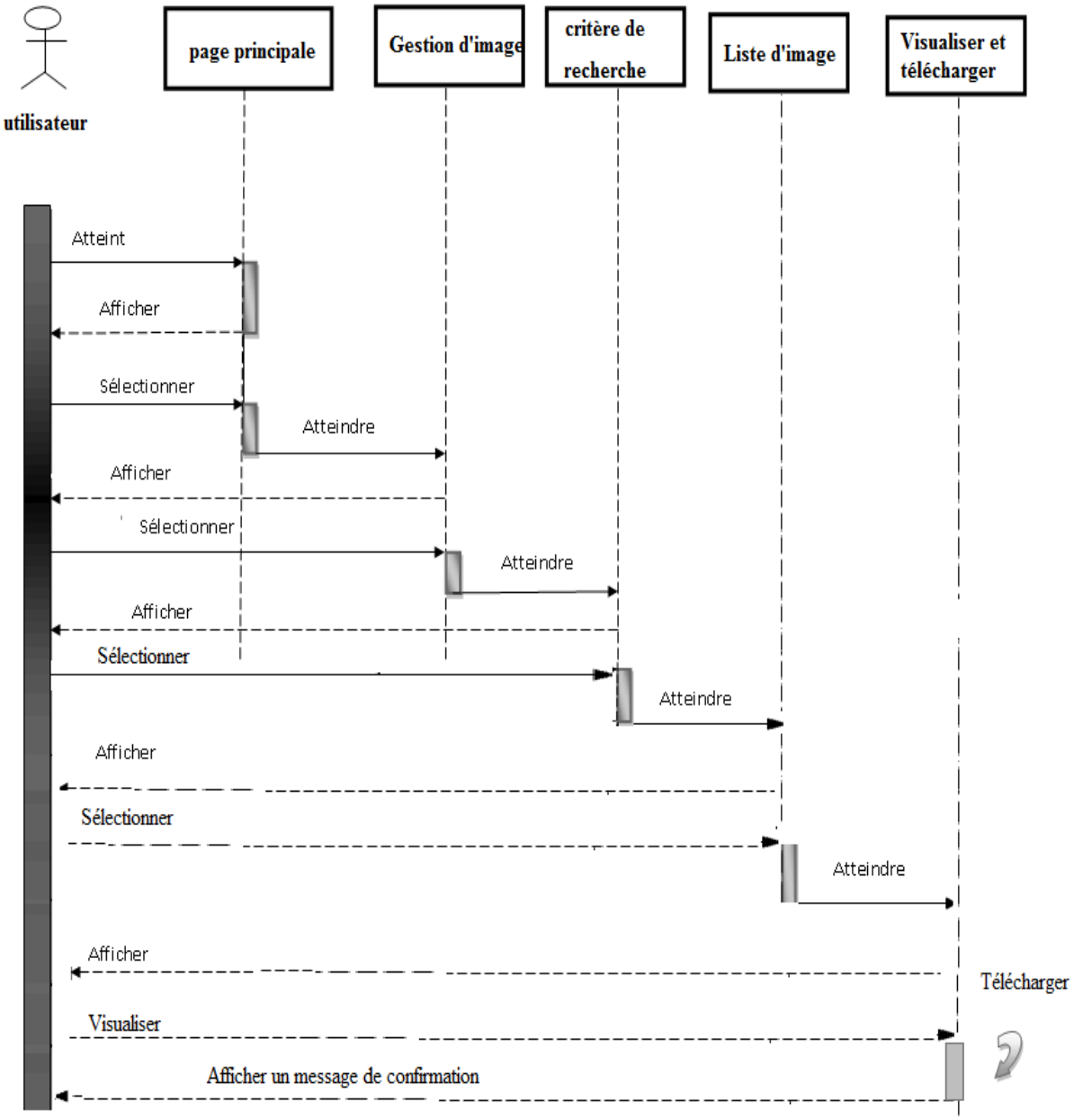

Figure II.11 : Diagramme de séquence : «Téléchargement d'une image»**.**

.

II.4.2 Le diagramme d'activité :

Une activité représente une exécution d'un mécanisme, un déroulement d'étapes séquentielles. Le passage d'une activité vers une autre est matérialisé par une transition. Les transitions sont déclenchées par la fin d'une activité et provoquent le début immédiat d'une autre (elles sont automatiques).

II.4.2.1 Le diagramme d'activité «S'authentifier» :

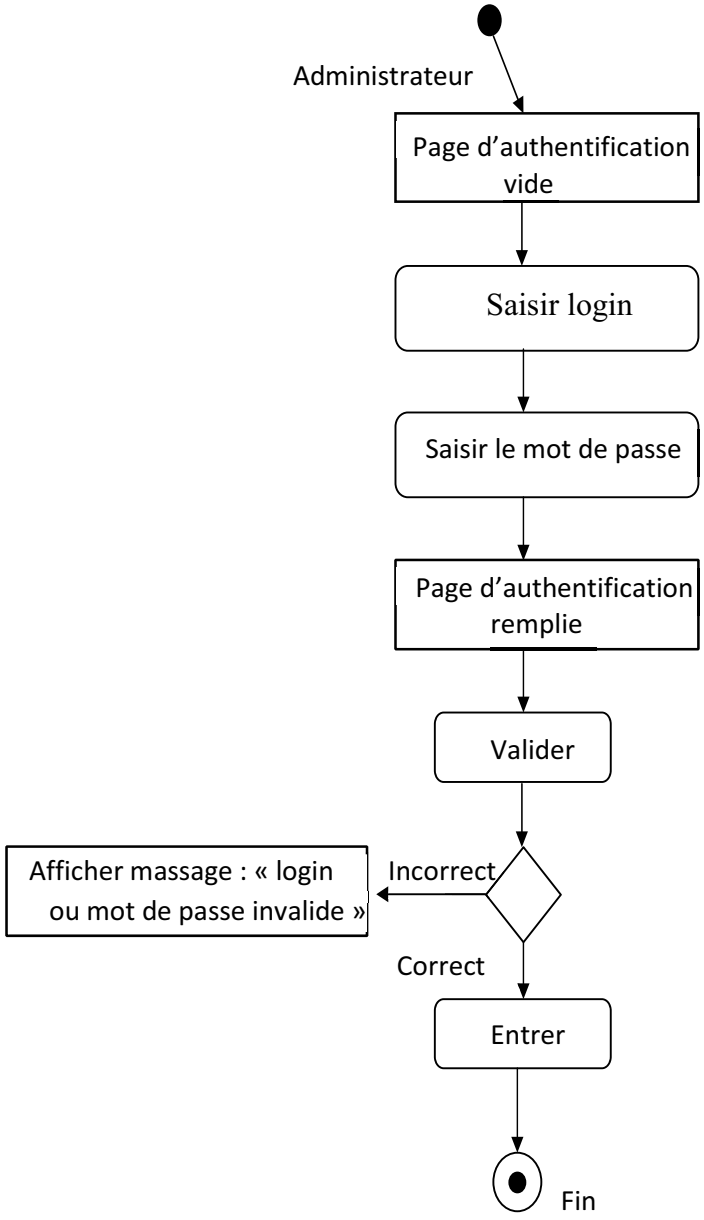

Figure II.12 **:** Diagramme d'activité du cas utilisation : « S'authentifier ».

II.4.2.2 Diagramme d'activité «Ajouter un utilisateur» :

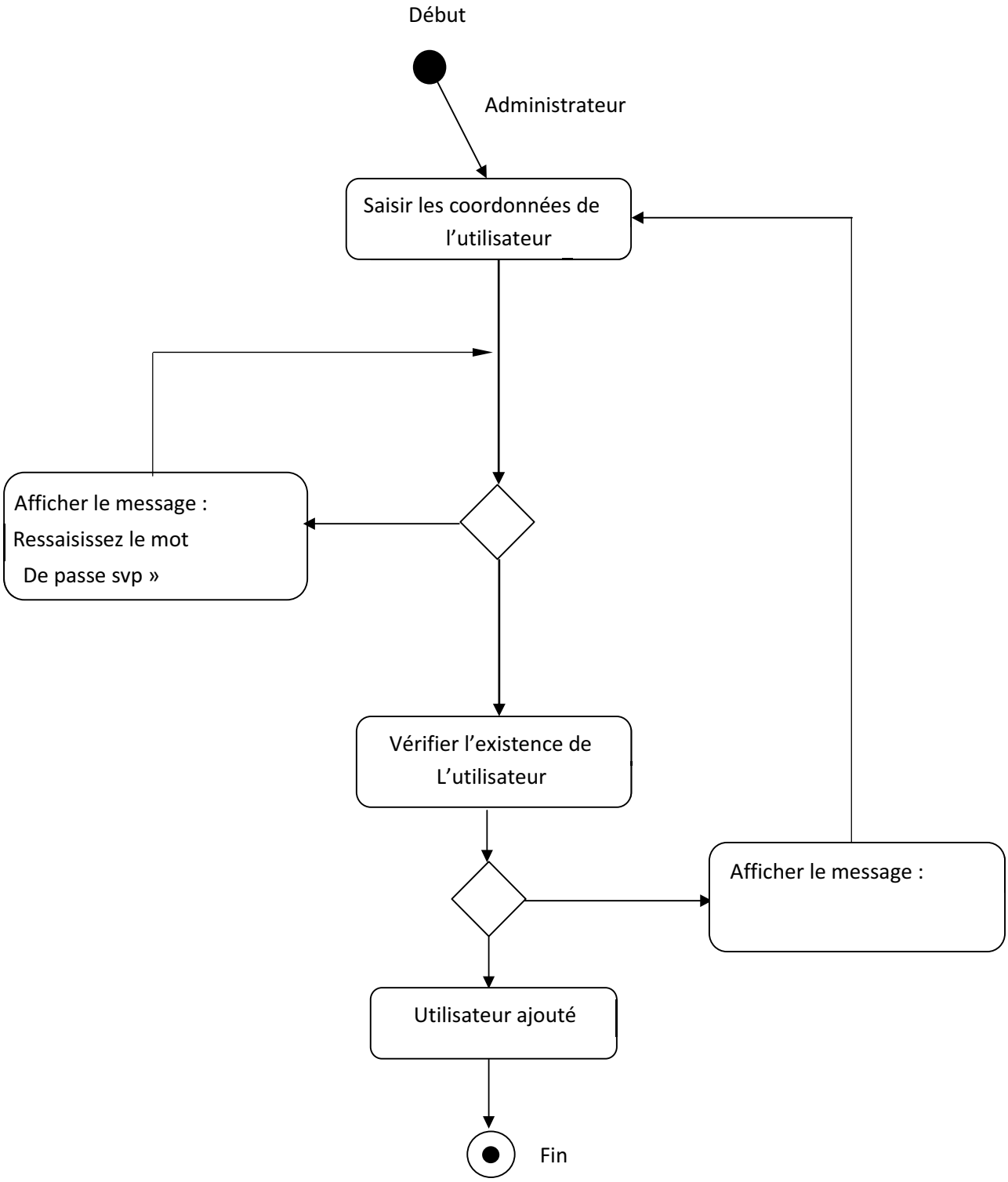

Figure II.13 **:** Diagramme d'activité du cas utilisation : « Ajout d'un utilisateur ».

II.4.2.3 Diagramme d'activité «Télécharger une image par l'utilisateur» :

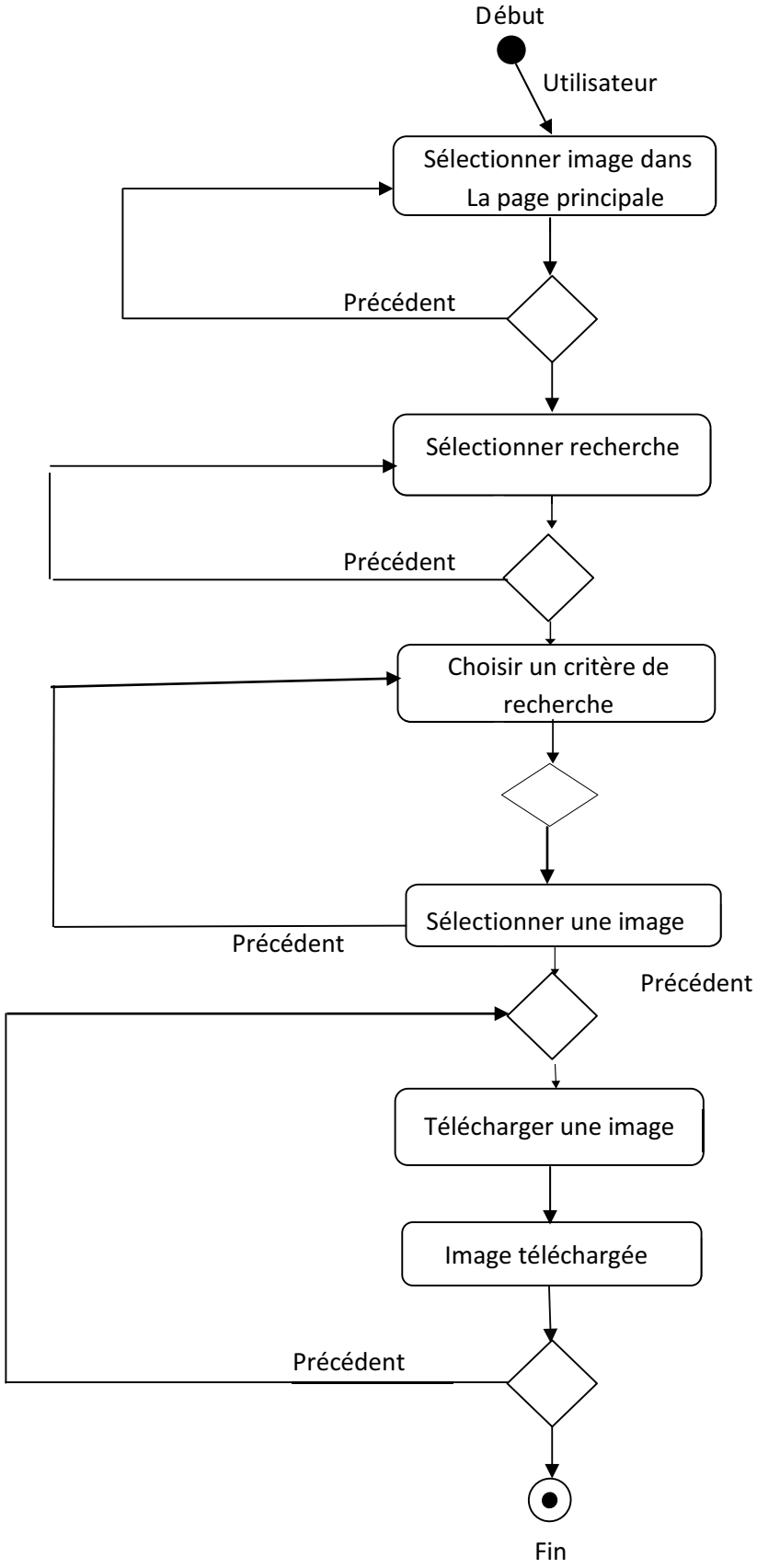

Figure II.14 **:** Diagramme d'activité : « Téléchargement d'une image par l'utilisateur ».

II.4.3 Diagramme de classe :

Le diagramme de classes représente l'architecture conceptuelle du système : il décrit les classes que le système utilise, ainsi que leurs liens, que ceux-ci représentent un emboitage conceptuel (héritage, marqué par une flèche terminée par un triangle) ou une relation organique (agrégation, marqué par une flèche terminée par un diamant).

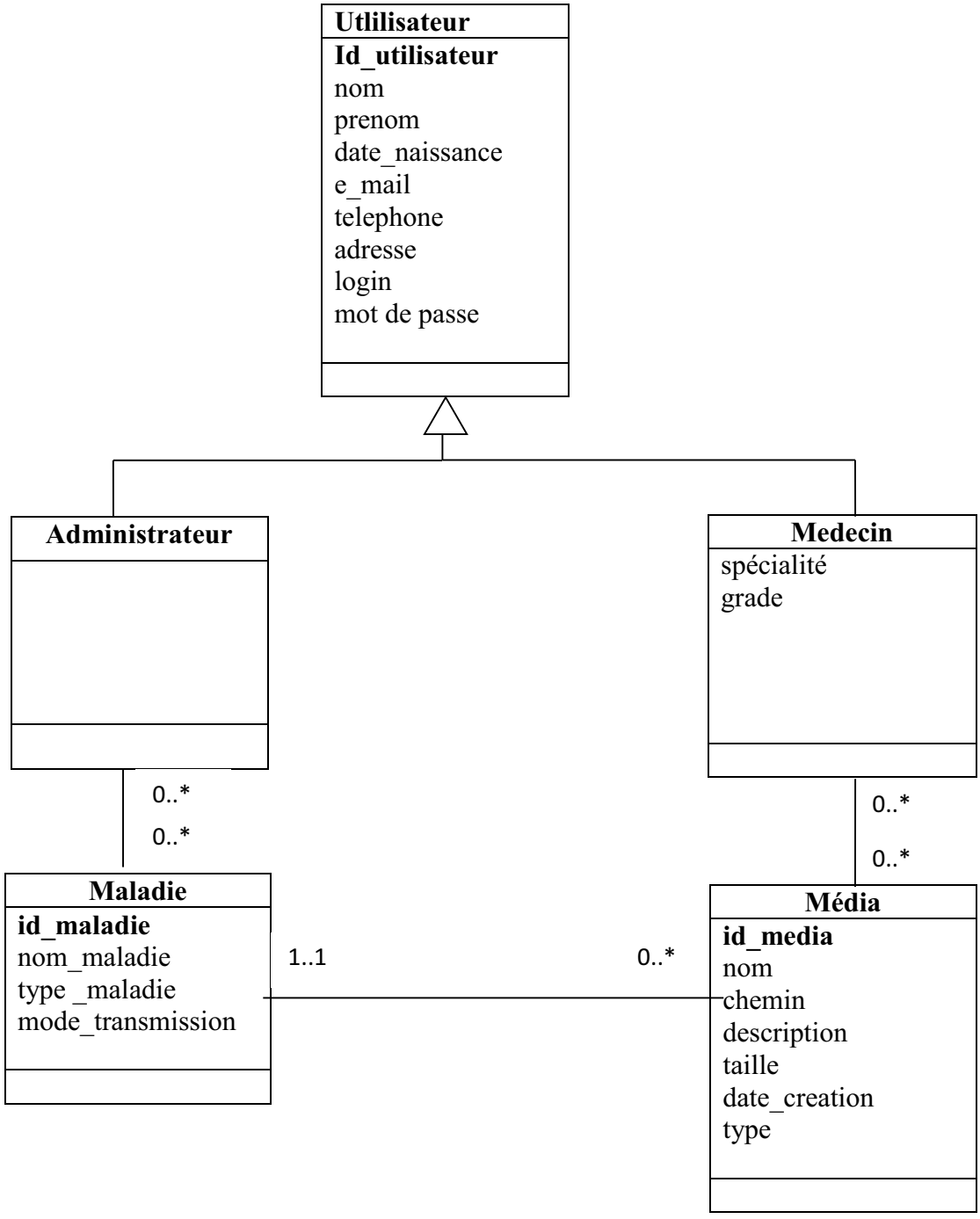

Figure II.15 **:** Diagramme de classe.

II.5 Conclusion :

Dans ce chapitre, nous avons proposé une démarche de modélisation pour développer notre application. Cette démarche est basée sur l'UML, nous avons commencé par la spécification des cas d'utilisations dans un premier temps, suivi d'une élaboration des diagrammes de séquences et d'activités ensuite les diagrammes de classes. L'implémentation et le déploiement du service MULTIMEDIA seront présentés dans le chapitre suivant.

Chapitre 3

III.1 Introduction :

Après avoir présenté dans le chapitre précédent les différentes étapes d'analyse et de conception, nous allons présenter dans ce dernier chapitre l'environnement de développement, les outils qui ont servi à la réalisation de notre application, et nous terminerons par la présentation de ses fonctionnalités à travers quelques interfaces.

III.2 Réalisation de l'application mobile :

III.2.1 Matériels utilisés :

- Smartphone Samsung.
- PC portable doté d'une carte WI-FI.
- Un point d'accès WI-FI.

III.2.2 Logiciels utilisés :

Pour pouvoir réaliser une application dans les bonnes conditions il faut en tout premier lieu bien choisir l'environnement de travail et ce selon nos besoins, nous avons opté pour la réalisation d'une application mobile sous-système Android ce qui nous impose de travailler sous Eclipse avec le langage JAVA et le SDK Android pour son développement, afin d'obtenir un fichier APK qui sera par la suite installé sur les terminaux mobiles fonctionnant sous système Android version 2.3.3 et plus.

III.2.3 Environnement de développement :

III.2.3.1 Le langage JAVA :

Le langage Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy. Il fut présenté officiellement le 23 mai 1995 au SunWorld. La société Sun a été ensuite rachetée en 2009 par la société Oracle qui possède désormais Java.

La particularité et l'objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou Linux, avec peu ou pas du tout de modifications. Pour cela, diverses plateformes associées visent à garantir la portabilité des applications développées en Java.

III.2.3.2 L'IDE Eclipse :

L'IDE Eclipse est un projet, décliné et organisé en un ensemble de sous-projets de développements logiciels, visant à fournir un environnement de production de logiciels libres qui soit extensible, universel et polyvalent, permettant de programmer dans différents langages (principalement Java), et ce grâce à ses nombreux plug-ins et notamment le plug-in Android.

III.2.3.3 Le SDK Android :

Le kit de développement (SDK) d'Android est un ensemble complet d'outils de développement. Il inclut un débogueur, des bibliothèques logicielles, un émulateur, de la documentation, des exemples de code et des tutoriaux. Les plateformes de développement prises en charge par ce kit sont les distributions sous Noyau Linux, Mac OS X 10.5.8 ou plus, Windows XP ou version ultérieure. L'IDE officiellement supporté est Eclipse combiné au plugin d'outils de développement d'Android (ADT). Les développeurs peuvent utiliser n'importe quel éditeur de texte pour modifier les fichiers Java et XML, puis utiliser les outils en ligne de commande (Java Development Kit et Apache sont obligatoires) pour créer, construire et déboguer des applications Android ainsi que contrôler des périphériques Android.

III.2.3.4 L'ADT (Android Development Tools):

C'est un plugin pour l'IDE Eclipse, qui est conçu pour donner un environnement puissant, intégré dans lequel il est possible de construire des applications Android.

Le développement dans Eclipse avec un ADT est fortement recommandé car ce dernier étend les capacités de Eclipse pour permettre la configuration rapide de nouveaux projets Android, créez une interface utilisateur pour l'application, ajouter des packages basés sur l'API Framework Android, déboguer les applications en utilisant les outils SDK Android, et même exporter signé (ou non signée) des fichiers.apk afin de distribuer une application.

III.2.3.5 Le SGBD (MYSQL) :

MySQL est un SGBD relationnel développé dans un souci de performances élevées. Il est multi threads, multiutilisateurs.

MYSQL est un gestionnaire de base de données libre. Il est très utilisé dans les projets libres et dans le milieu industriel.

Les principaux Atouts de MySQL sont la rapidité la robustesse, et la facilité d'utilisation car il est trois a quatre fois plus rapide que la plupart des autres bases de données commerciales

et ceci grâce a l'implémentation des fonctions SQL qui s'est faite a travers des classes librairies extrêmement optimisées.

III.2.4 Préparation de l'environnement de développement :

III.2.4.1 Choix d'une version d'Eclipse :

Nous avons essayé plusieurs versions d'Eclipse et il apparait que la version INDIGO est celle qui fonctionne le mieux pour développer un Projet Android.

III.2.4.2 Installation du plugin ADT pour Eclipse :

1) Installer un nouveau « Software ».

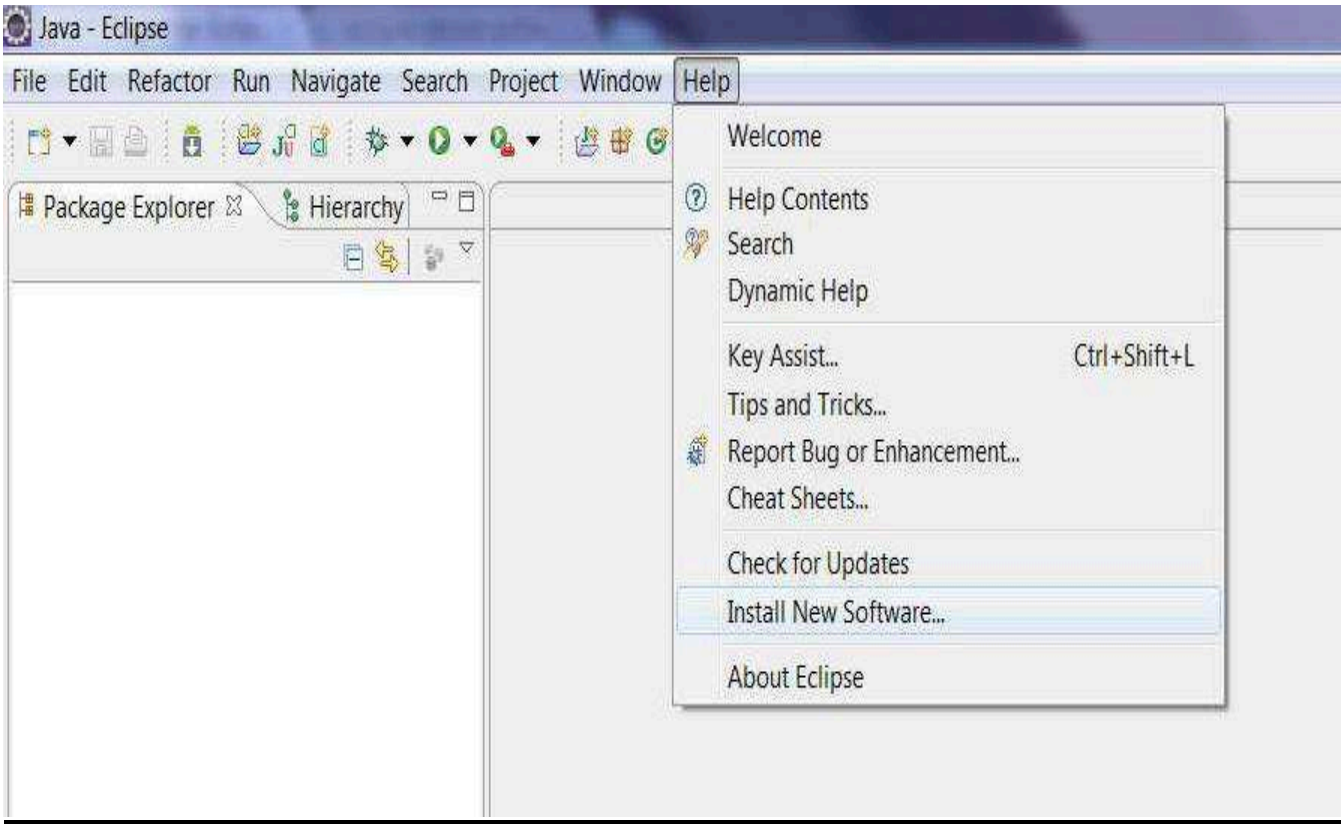

Figure III.1 : Installation d'un Software.

2) Ajouter un nouveau site.

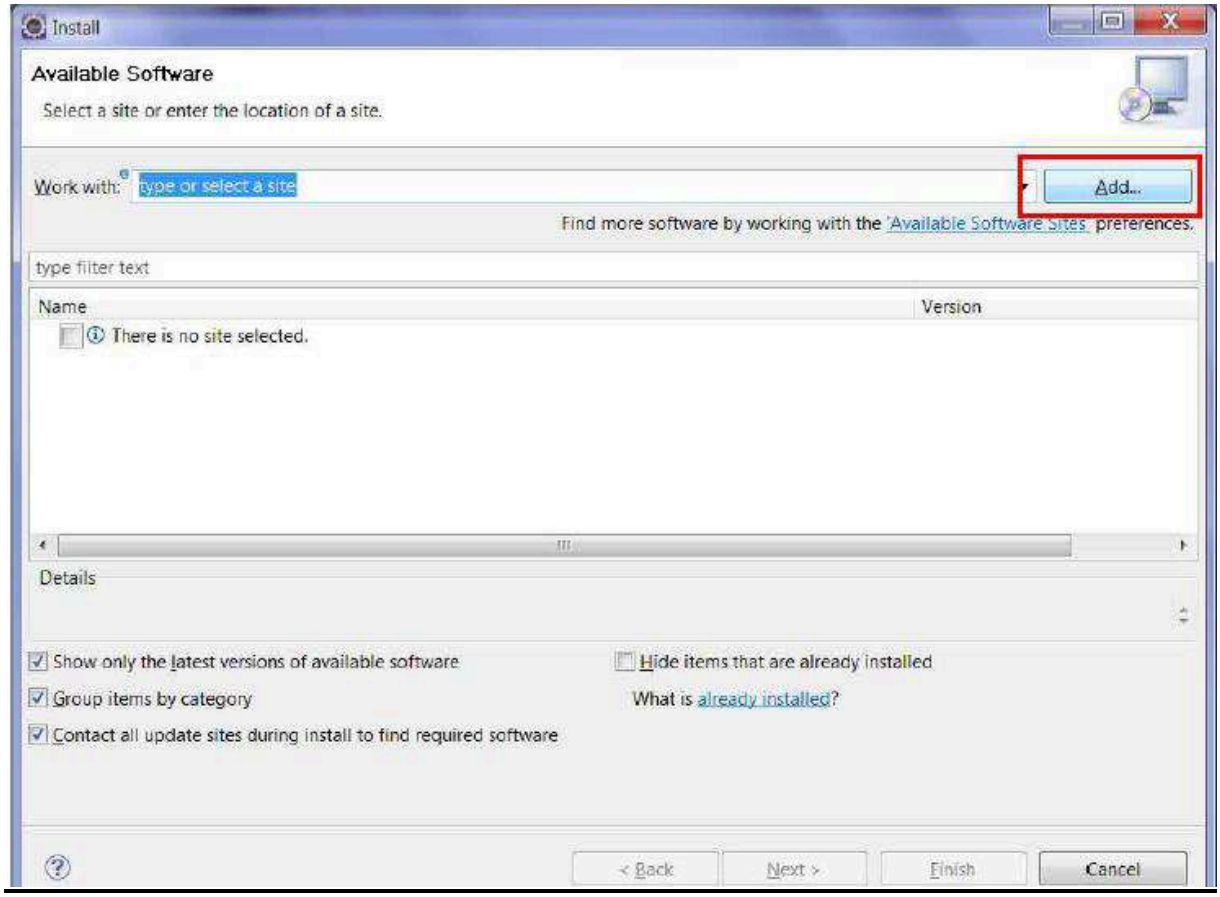

Figure III.2 : Ajout d'un nouveau Site.

3) Localisation de nouveau site.

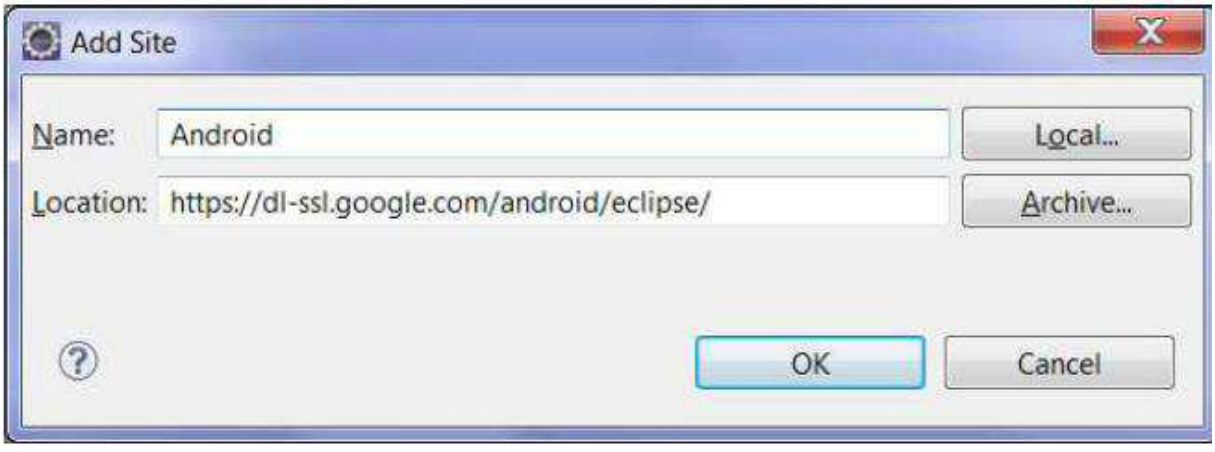

Figure III.3 : Localisation de nouveau site.

4) Sélectionner tout le package à télécharger. Et installer les.

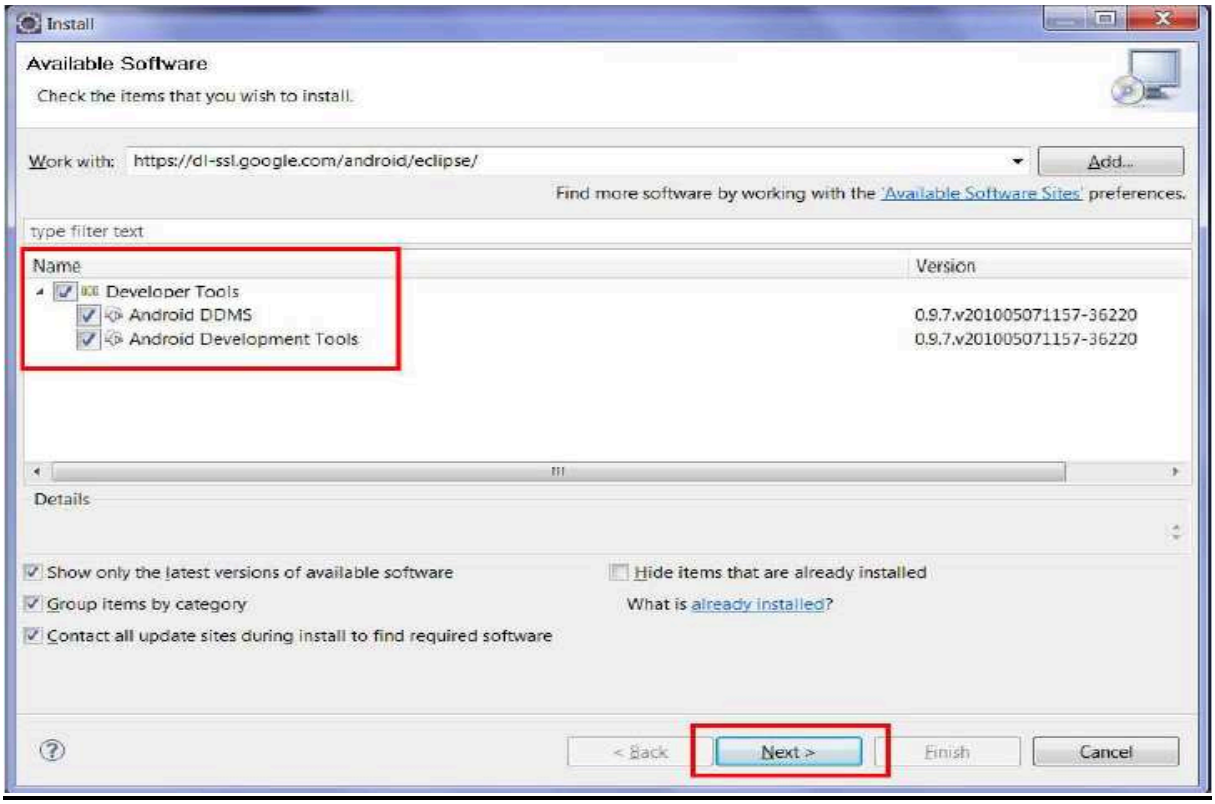

Figure III.4 : Installation de packages.

5) Puis redémarrer Eclipse. Android devrait être installé et visible :

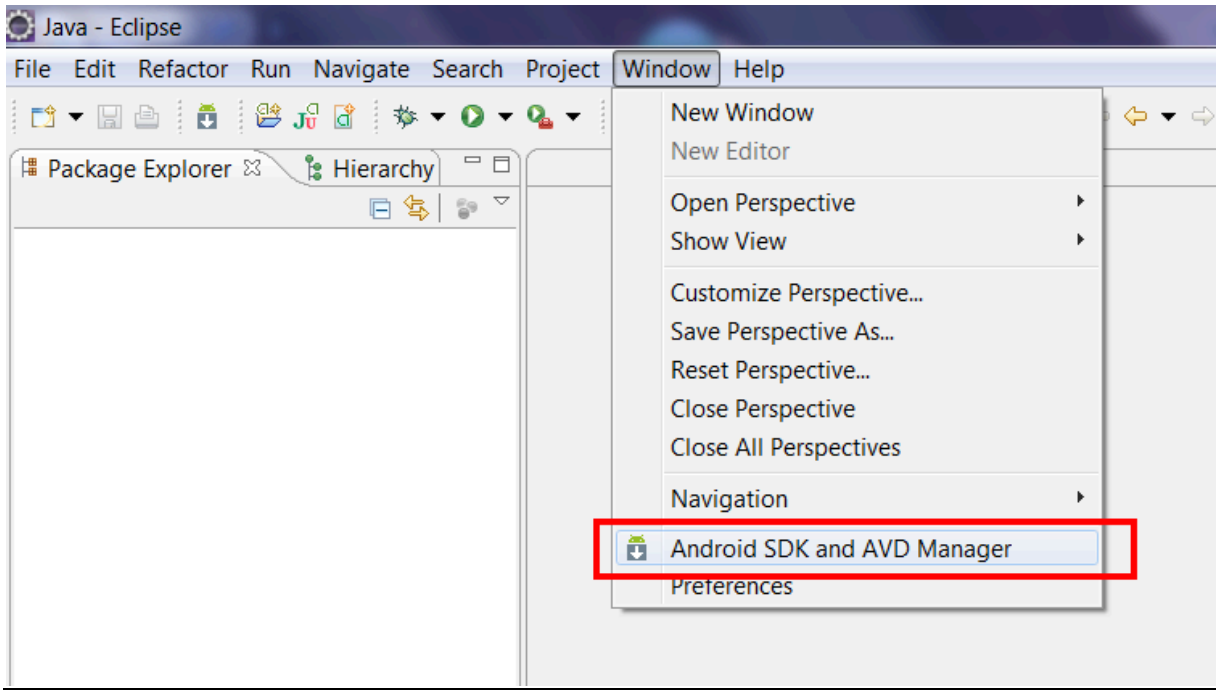

Figure III.5 :L'apparaissions des icônes Android SDK et AVD Manager.

III.2.4.3 Installation d'un téléphone virtuel Android :

III.2.4.3.1 Téléchargement d'Android SDK :

1) Lancer SDK Setup.exe.

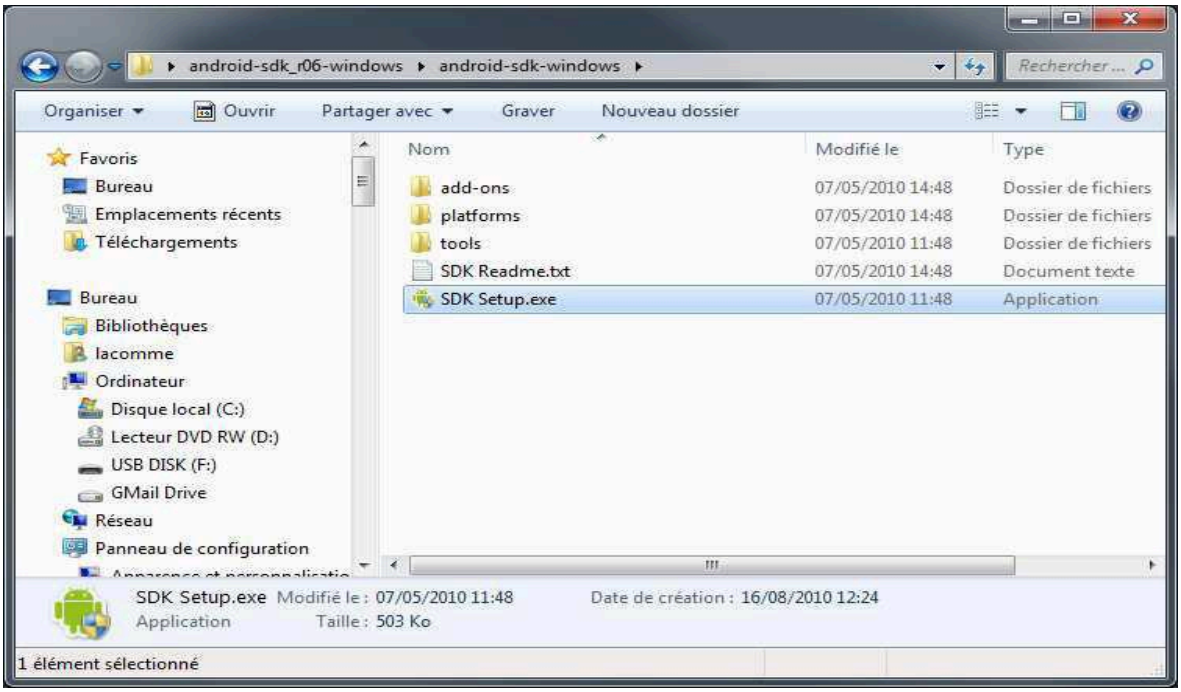

Figure III.6 : Lancement de SDK Setup.exe.

2) Allez dans le sous-menu **Settings** et cocher **Force https.**

3) Aller dans **Available Packages et** Cocher **https**.

4) Appuyer sur Install Selected et ensuite attendre la fin des téléchargements et appuyer sur Close.

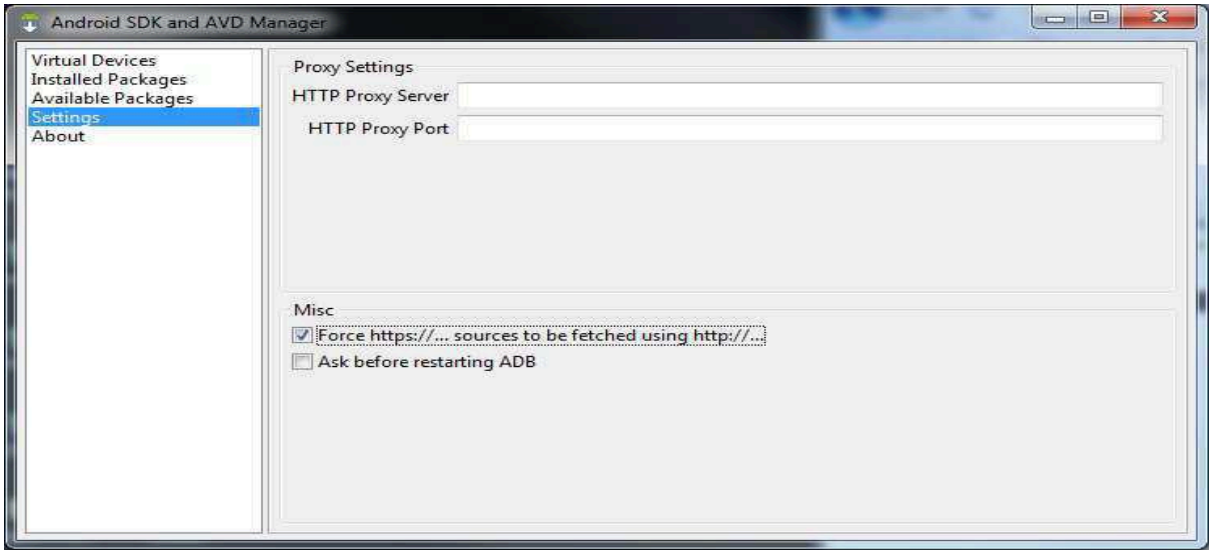

Figure III.7 : Installation de Force https et https.

III.2.4.3.2 Création d'un téléphone Virtuel d'Android (Virtual Devices) :

1) Choisir le menu **Virtual Devices**.

2) Cliquer sur new et remplir les champs qui apparaîtront :

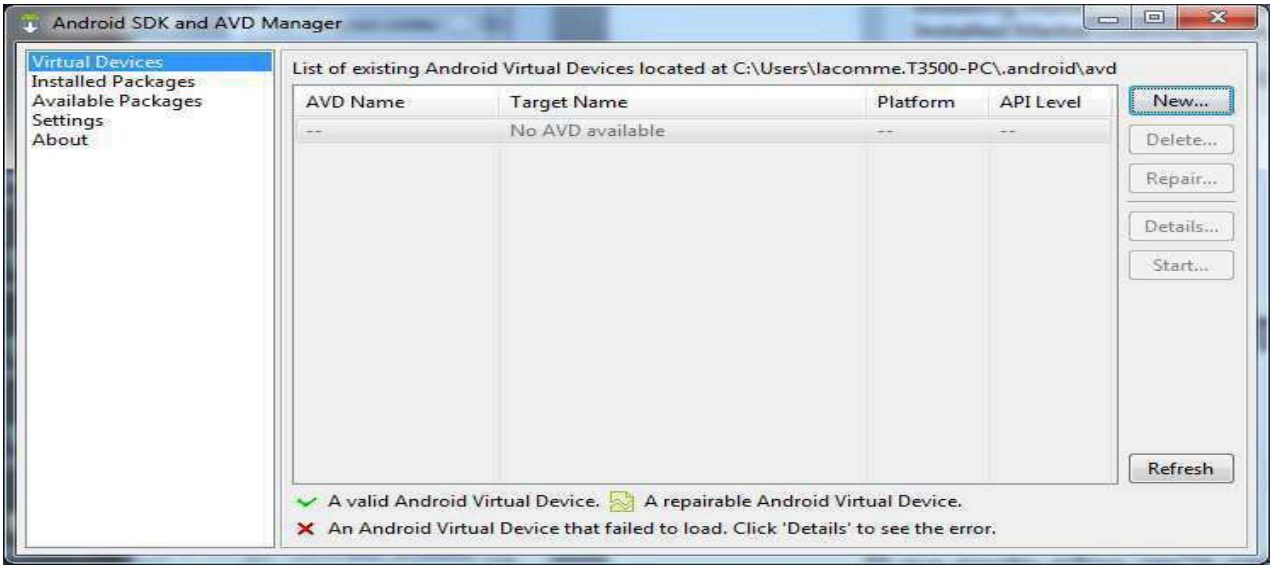

Figure III.8 : Etapes de création d'un virtuel Devices.

3) Cliquer sur start, un emulateur qui apparait comme suit :

Figure III.9 : Obtention d'un téléphone Android virtuel**.**

Remarque : On peut créer autant d'émulateur qu'on veut selon nos besoins.

III.2.5 La connexion à la base de données à distance (Android/MySQL) :

MySQL est un gestionnaire de bases de données relationnelles, basé sur le langage de requête SQL (Structured Query Language), fonctionne en mode client /serveur, le serveur MySQL contrôle l'accès aux bases de données pour assurer que plusieurs utilisateurs peuvent se servir simultanément d'une même base de données.

Il est impossible de se connecter à une base de données directement sous android. La méthode la plus répandue pour se connecter à une base de données MySQL à distance à partir d'un appareil Android, est de mettre en place un script PHP et d'exécuter ce script en utilisant le protocole http.

Ce script PHP convertit (encoder) les données récupérées en format JSON en utilisant la fonction **json_encode()**, Cette méthode est donc très avantageuse, souple et facile à mettre en œuvre. Et pour mieux comprendre la procédure, nous allons la schématiser :

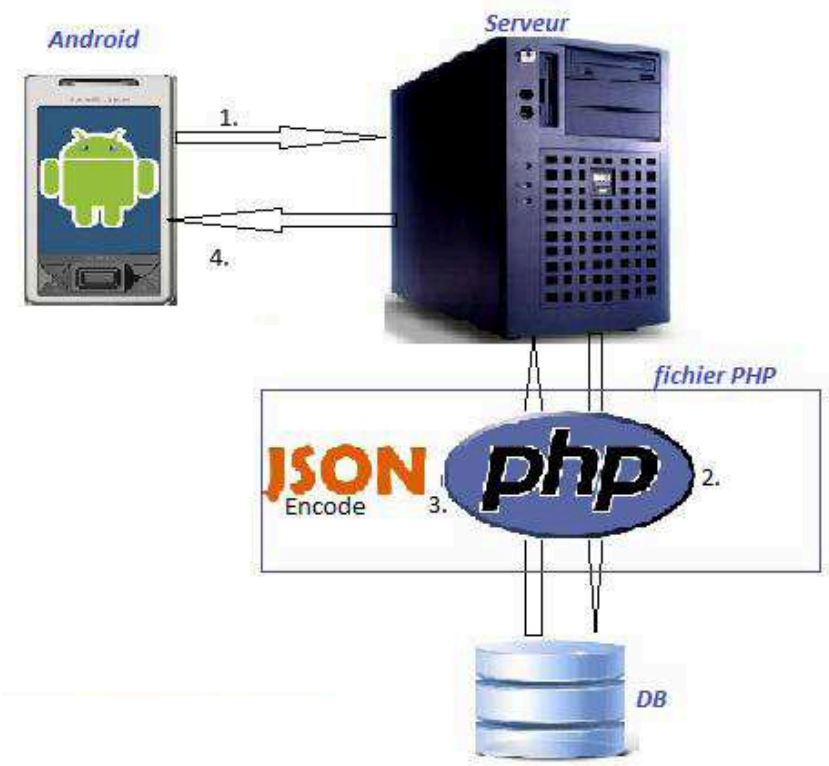

Figure III.10 : La connexion à distance (Android /MySQL)**.**

1. Commande HttpClient.
2. Appel du fichier PHP.
3. Renvoi data en JSON.
4. Récupération et décodage JSON.

III.2.5.1 Coté serveur :

- **Le format JSON *JavaScript Object Notation*) :**

JSON est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme le permet XML par exemple.

JSON est un format texte complètement indépendant de tout langage, mais les conventions qu'il utilise seront familières à tout programmeur habitué aux langages descendant du C, comme par exemple : C lui-même, C++, C#, Java, JavaScript, Perl, Python

et bien d'autres. Ces propriétés font de JSON un langage d'échange de données idéal. Un document JSON ne comprend que deux types d'éléments structurels :

- des ensembles de paires nom / valeur ;
- des listes ordonnées de valeurs.

Ces mêmes éléments représentent trois types de données :

- des objets ;
- des tableaux ;
- des valeurs génériques de type tableau, objet, booléen, nombre, chaîne ou null.

Remarque : JSON ne peut transporter que les chaine de caractère et pour cela, on doit coder nos objets en base 64.

- **Le Format Base 64 :**

Base 64 est un codage de l'information utilisant 64 caractères, choisis pour être disponibles sur la majorité des systèmes. Il est principalement utilisé pour la transmission de messages sur l'internet.

Le principe du codage Base 64 consiste à utiliser des caractères US-ASCII (caractères non accentués) pour coder tout type de données codé sur 8 bits. Les protocoles de courrier électronique ont en effet été prévus à l'origine pour transporter des messages en texte seulement. Or, l'échange de données binaires se traduit la plupart du temps par des transformations du contenu rendant illisible le document original.

Le format Base64, utilisé massivement dans les échanges de courrier électronique, permet ainsi de transmettre n'importe quel document binaire (application, vidéo, fichier audio, etc.) en pièce jointe d'un courrier électronique en les codant à l'aide de caractères classiques.

- **Fichier PHP :**

Le fichier PHP consiste à envoyer des requêtes sql à la base de données. Il récupère le résultat des requêtes, ensuite l'encode en JSON grâce au simple appel de la fonction json_encode() comme dit précédemment.

III.2.5.2 coté client :

Au niveau du Client android, il s'agit de demander au serveur de nous renvoyer le résultat de la requête grâce aux commandes *HttpClient,* Ensuite, il nous appartient de décoder ces données (puisqu'elles sont renvoyées sous forme d'objet JSON) afin de pouvoir les afficher.

Remarque *:* si la requête manipule par exemple un fichier on doit le décoder aussi en base64.

III.2.6 Description de l'application mobile :

Nom d'utilisateur et le mot de passe vont être transmis au serveur et seront vérifiés auprès de la base de données, la réponse pour l'authentification sera transmise, si le nom d'utilisateur ou le mot de passe est invalide alors un message d'erreur sera affiché, sinon le client accède à la page d'accueil « Service Multimédia ».

Figure III.11 : Interface «Authentification».

Lors de l'apparition de la fenêtre « Accueil multimédia » l'utilisateur a le choix de choisir un fichier multimédia (Image, Vidéo, Audio, Document).

Figure III.12 : Interface « Accueil multimédia ».

Lors de l'apparition de la fenêtre « Accueil multimédia », si l'utilisateur a choisi par exemple « Image », l'interface de « Gestion d'une image » sera affichée et l'utilisateur doit sélectionner la tâche à effectuer, puis il valide son choix.

Figure III.13 : Interface « Gestion d'une image ».

Si l'utilisateur a choisi par exemple «Rechercher», l'interface de «critères de recherche» sera affichée et l'utilisateur doit sélectionner un ou plusieurs critères pour effectuer sa recherche, puis valider le choix.

Figure III.14 : Interface « Critères de recherche ».

Lors de l'apparition de la fenêtre « Critères de recherche», Si l'utilisateur a choisi par exemple le critère « Nom de maladie», l'interface de «liste de maladies» sera affichée et l'utilisateur doit sélectionner une maladie.

Figure III.15 : Interface « Liste de maladies ».

Si l'utilisateur a choisi par exemple la maladie «Allergie», l'interface de «liste de maladies» sera affichée et l'utilisateur doit choisir une image.

Figure III.16 : Interface « Liste des images ».

Lors de l'apparition de la fenêtre « Liste des images», l'utilisateur doit sélectionner une image, en cliquant au-dessous pour l'agrandir puis la télécharger.

Figure III.17 : Interface « Téléchargement image ».

III.3 Réalisation de l'application administrateur :

III.3.1 Les logiciels utilisés :

III.3.1.1 L'IDE Netbeans :

En 1997, NetBeans naît de Xelfi, un projet d'étudiants dirigé par la Faculté de mathématiques et de physique de l'Université Charles de Prague. Plus tard, une société se forme autour du projet et édite des versions commerciales de l'IDE NetBeans, jusqu'à ce qu'il soit acheté par Sun en 1999. NetBeans est un environnement de développement intégré (IDE), placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and DistributionLicense). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Développent Kit JDK est requis pour les développements en Java.

III.3.2 Description de l'application gestion :

Cette partie est utilisée par l'administrateur du système multimédia, elle a pour but de gérer les utilisateurs (ajouter un médecin, supprimer un médecin, modifier un compte médecin) les maladies (ajouter une maladie, supprimer une maladie…).

L'administrateur doit tout d'abord s'authentifier par un pseudo et un mot de passe, la figure ci- dessous représente l'interface d'authentification :

Figure III.18 : l'interface d'authentification de l'administrateur.

Une fois que l'administrateur s'est authentifié, l'interface d'accueil de l'application sera affichée comme la montre la figure ci-dessous :

Figure III.19 : Interface d'accueil de l'application de gestion.

Les figures ci-dessous représentent quelques interfaces de tâches qui peuvent être effectuées au niveau de l'application de gestion :

- **Gestion des utilisateurs :**

L'administrateur peut sélectionner le bouton « Gérer les médecins», une interface «Gestion de compte » sera affichée comme la montre la figure ci-dessous :

Figure III.20 : Interface gestion de compte.

L'administrateur peut sélectionner le bouton « Ajouter un médecin », un formulaire d'ajout s'affiche. Il saisit les coordonnés du médecin puis les validées en cliquant sur le bouton «valider », l'identifiant, login et mot de passe seront généré automatiquement

Figure III.21 : Interface d'ajout d'un médecin par l'administrateur.

- **Gestion des maladies :**

L'administrateur peut sélectionner le bouton « Gérer les maladies», une interface «Gestion de maladies » sera affichée comme la montre la figure ci-dessous :

Figure III.22 : Interface gestion de maladies.

L'administrateur peut sélectionner le bouton « Rechercher une maladie», un formulaire de recherche sera affiché. Il saisira les coordonnés du utilisateur puis les validées en cliquant sur le bouton « rechercher » comme la montre la figure ci-dessous :

Figure III.23 : Interface gestion de maladies.

III.4 Conclusion

Dans ce chapitre nous avons en premier lieu présenté l'environnement et les outils utilisés pour implémenter notre application. Puis nous avons présenté des exemples réels de cas d'utilisation de l'application afin de vous donner une idée globale et un premier aperçu de notre projet.

Conclusion générale

Conclusion générale :

En vue de croissance de Smartphone au niveau mondial ces dernières années, la téléphonie mobile est sans doute le secteur le plus dynamique, le plus rentable et le plus innovant de toute l'industrie de télécommunication que fait partie de nos quotidiens, et que la plupart de ces Smartphone ont comme système exploitation l'Android. Nous étions menées à explorer ce nouveau système d'exploitation pour mobiles et à faire une application client/serveur qui permet aux médecins de partager les connaissances et les données multimédia concernant les différentes maladies et se mettre en relation avec des professionnels de santé de toutes spécialité médicale afin de faciliter l'étude, enrichir les connaissances ,et garantir une qualité de soin optimal pour les patients.

Ce projet est donc composée de 3 parties, dans le premier nous avons donné des généralités sur les réseaux sans fil, l'architecture client /serveur, les applications mobiles et les bases de données multimédia, puis nous avons consacré le deuxième chapitre pour l'analyse et la conception de notre application, Finalement nous avons décrit notre application et ses fonctionnalités après avoir définir l'environnement de développement.

Ce travail nous a permet :

- d'acquérir des connaissances très importantes dans les bases de données multimédia et les applications android avec une base de données externe.
- De traiter et de convertir les différentes formats de données (json, base 64, byte array, URI …).
- D'approfondir nos connaissance en quelques langage qu'on a déjà utilisé (java php ,mysql, UML…) .

Nous espérons que ce modeste travail a abouti aux objectifs fixés au départ et sera un guide considérable pour les nouvelles promotions.

.

Bibliographie

Sites :

http://www.commentcamarche.net

http://www.wikipedia.org

http://www.developpez.com

http://stackoverflow.com

http://www.apple.com

http://developer.android.com

http://www.json.org

http://easy-android.over-blog.com

http://www.androidhive.info

http://www.cnblogs.com

http://sunzone.iteye.com

http://longluo.github.io

http://www.oschina.net

https://zestedesavoir.com

http://examples.javacodegeeks.com

Livres :

Créer des applications Android(Auteurs : Philippe Lacomme, Raksmey Phan)

L'art du développement Android (Edition PEARSON)

Mémoires :

SIDALI et BELLOUNI, « conception et réalisation d'une application mobile de sauvegarde des répertoires téléphoniques SAVBOX », Mémoire d'ingénieur, UMMTO 2008.

MECHERRI et OULD AMER, «Base de données multimédiassousORACLE10g ». Mémoire de licence, UMMTO 2008.