République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane Mira -Bejaïa
Faculté des sciences exactes
Département d'Informatique

Mémoire de fin d'étude
En vue de l'obtention de diplôme de Master en Informatique
Option : Administration et Sécurité des Réseaux

Thème

Conception et Réalisation d'une Application Réseau pour le Système ANDROID
Cas : Gestion du Personnel de l'Université A. Mira de Bejaia GUAMB.

Réalisé par :
M^r^ DJEBARA Yacine
M^r^ MESSAOUDI Abdelmadjid

Encadré par :
M^r^ OUZEGGANE Redouane

Examinateurs :
M^r^ ATMANI Mouloud
M^lle^ BOUADEM Nassima

Président de jury :
D^r^ SIDER Abderrahmane

Promotion 2012

Dédicaces

Je dédie ce modeste travail à

Mes parents,
Mes frères et Mes sœurs,
Ma petite amie,
Et tous mes amis.

Djebara Yacine.

Dédicaces

Je dédie ce travail à

Mes parents,
Mes frères et Mes sœurs,
Et tous mes amis.

Messaoudi Abdelmadjid.

Remerciements

Au terme de ce travail, on tient à remercier

Monsieur, **D^r** SIDER Abderrahmane

pour avoir accepté de présider le jury de notre soutenance.

Ainsi les examinateurs **M^r** ATMANI Mouloud et **M^lle** BOUADEM Nassima.

Notre encadreur Monsieur, **M^r** OUZEGGANE Redouane, pour ses qualités humaines et professionnelles, pour son encadrement, ses directives, ses remarques constructives, et sa disponibilité.

Djebara Yacine.
Messaoudi Abdmadjid

Table des Matières

Liste des Figures

Liste des Abréviations

ADT	Abstract Data Type, plugin Android pour Eclipse.
API	Application Programming Interface.
APK	Android Package.
AVD	Android Virtual Device, Terminal Android Virtuel.
BSD	Berkeley software distribution license, abrégé en BSD, désigne en informatique une famille de systèmes d'exploitation Unix.
FTP	File Transfer Protocol, protocole de transfert des fichiers d'un ordinateur vers autre.
GPRS	GPRS est une norme pour la téléphonie mobile dérivée du GSM permettant un débit de données plus élevé.
GUAMB	Gestion d'Université Abderrahmane Mira de Bejaia.
GUI	Graphical User Interface, c'est l'interface graphique utilisateur.
HTTP	Hyper Text Transfer Protocol, protocole de communication entre serveur web et utilisateur.
IDE	Integrated Development Environment (Environnement de Développement Intégré).
IP	Internet Protocol, protocole de communication utilisé par l'ordinateur relié à internet.
JDBC	Java Data Base Connectivity.
LIBC	Library (libc) est la bibliothèque standard C écrite par Roland Mc Grath pour le projet GNU.
MVC	Model Vue Contrôleur.
MySQL	est un système de gestion de base de données (SGBD).Selon le type d'application, sa licence est libre ou propriétaire.
OHA	Open Handset Alliance, consortium d'entreprises soutenant le projet Android.
OS	Operating System.
PDA	Personal Digital Assistant, appareil numérique portable.
PHP	**P**ersonal **H**ome **P**age, langage utilisé pour produire des pages Web dynamiques
SDK	Software Development kit, Kit de développement permettant la création de type défini.
SGBD	Système Gestion Base de Données.
SGL	Skia Graphics Library, Bibhitoque graphique 2D utilisée par Android.

SMTP	Simple Mail Transfer Protocol (littéralement « Protocole simple de transfert de courrier »).
SQL	Structured Query Langage.
SSL	Secure Sockets Layer, un protocole de sécurisation des échanges sur internet.
TCP	Transmission Control Protocol.
UDP	Le protocole UDP fournit un service en mode sans connexion et sans reprise sur erreur.
UML	Unified Modeling Language, Langage de modélisation Graphique dans le monde du génie logiciel.
WiFi	Wireless fidélité, réseaux sans fil.
WLAN	Wireless Local Area Network, le réseau wifi local.
XML	Extensible Markup Language, Langage informatique de balisage générique.

Introduction Générale

Introduction Générale

Les progrès technologiques récents ont permis l'apparition d'une grande variété de nouveaux moyens permettant à un utilisateur d'accéder et d'utiliser l'information qui l'intéresse en tout lieu couvert par le réseau et à tout moment. L'accès au contenu ne s'effectue plus exclusivement de la même façon ni par les mêmes appareils qu'il y a quelques années. Ces nouveaux appareils, fruits d'une véritable révolution technologique, ont pour nom : assistants personnels, téléphones cellulaires, Smartphones, etc. Le nombre d'utilisateurs de ces nouveaux appareils continue sa croissance exponentielle. Les moyens d'accès au contenu ont également évolué, avec de nouveaux réseaux tels que les réseaux sans fil (Wifi, GPRS, etc.)

L'université de Bejaia est entrain de développer son propre système d'information automatisé en se basant sur son propre personnel. Ce système porte le nom de GUAMB : Gestion de l'Université A.Mira Bejaia. Ce dernier est développé d'une façon à utiliser uniquement des technologies open-sources ou libres, notamment le système d'exploitation *Linux*, langage de programmation *Java*, environnement de développement *Eclipse*.

Notre travail consiste à la conception et la réalisation d'une application réseau pour le système d'exploitation ANDROID d'un appareil mobile en vue de l'utilisation dans la gestion du personnels au sein de l'université A/Mira de Béjaia. Pour ce faire, notre choix est porté sur une architecture Client-serveur à 3 niveaux.

Le présent mémoire est structuré en quatre chapitres, le premier chapitre sera consacré à l'état de l'art sur le système Android, le second se focalise sur l'étude du système GUAMB (Gestion de l'Université A. Mira-Bejaia). Le troisième chapitre, présente l'analyse et la conception de l'application **GUAMB-Mobile.** Le quatrième chapitre illustre l'implémentation et la réalisation de l'application.

Enfin, nous concluons notre mémoire par une conclusion générales et quelques perspectives.

Chapitre I : Etat de l'Art sur les systèmes Android

I.1 Introduction

Nous présenterons dans ce chapitre une définition globale de l'informatique mobile et du système d'exploitation Android, ainsi que ses fonctionnalités, son architecture et le cycle de vie d'une activité Android.

I.2 L'informatique mobile

I.2.1 Présentation

L'informatique mobile s'inscrit sous l'approche globale de l'informatique ubiquitaire. Historiquement, elle a été autour depuis 1992. Depuis, elle constitue un outil de communications très puissant pour les entreprises et les utilisations personnel. **[07]**

Les supports qui lui ont été développés ont repris l'industrie sans fil. Il s'agissait des entités mobiles communicantes et parfois de très petites tailles permettant de se connecter aux différents types de réseaux tant qu'ils disposent des périphériques appropriés dont les Smartphones font partie. **[07]**

I.3 Les Smartphones

Un Smartphone est un téléphone « intelligent », appelé aussi ordi-phone, c'est un appareil dédié aux communications mobiles, disposant d'un système d'exploitation ouvert et adoptant des applications tierces développées par le fabricant, par l'opérateur ou par un éditeur de logiciel.

I.4 Le système d'exploitation Android

I.4.1 Présentation

L'histoire de la naissance d'Android a commencé en aout 2005, lorsque Google a acquis Android INC, une startup[1] qui développent des applications pour téléphones mobiles. Depuis lors, Andy Rubin, étant un ancien d'Android INC, a entamé son travail sur un système d'exploitation basé sur un noyau linux dédié aux appareils mobiles. En 2007, le 5 novembre, l'Open Handset Alliance fut officiellement annoncée. C'est un consortium qui réunit un

ensemble de sociétés, ayant pour but de développer des standards open sources pour terminaux mobiles. Le 15 novembre, le premier standard a été annoncé. Il s'agissait d'Android, une plateforme pour appareils mobiles.

En effet, Android représente un système d'exploitation open source dédié pour Smartphones, PDA et terminaux mobiles.

Il est basé essentiellement sur la simplicité d'utilisation et surtout sur une capacité de personnalisation importante présentant un argument commercial de poids. Pour promouvoir ce système d'exploitation open source, Google lui a conféré des alliés puissants réunis au sein de l'Open Handset Alliance tel que Samsung, Motorola, Sony Ericsson et LG. Les principaux concurrents d'Android sont Apple avec iPhone OS qui équipe l'iPhone, Research In Motion avec BlackBerry OS, Palm avec Nova ou Web OS, Microsoft et son Windows Mobile, etc.

la figure I.1 montre l'évolution des versions d'OS Android:

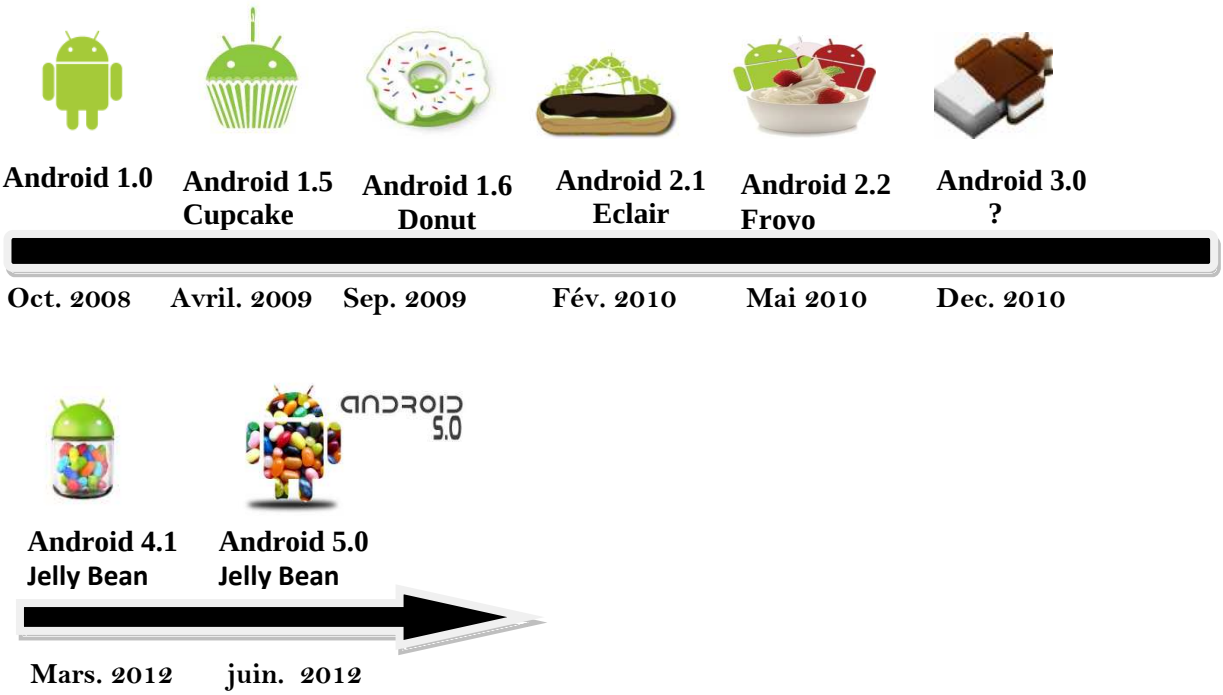

Figure I.1 : *Evolution des versions d'Android*

I.4.2 Fonctionnalités d'Android

Android a été conçu pour intégrer au mieux les applications existantes de Google comme le service de courrier Gmail, l'agenda Google Calendrier ou encore la cartographie Google Maps.

Les fonctionnalités proposées par Android différents d'une version à une autre, on peut citer celle de la version utilisée dans le cadre de notre travail (version 2.2) :

- Augmentation de la performance et de la vitesse ;
- Fonctionnalité de Hot spot Wifi ;
- Partage de contact sur Bluetooth ;
- Mise à jour automatique des applications.

I.4.3 Architecture Android

Android bénéficie d'une architecture en couche complète faisant de lui une plateforme riche, dédiée aux appareils mobiles. Il est à base de noyau linux profitant des services système de base tels que la sécurité, la gestion mémoire, gestion de processus, etc. À un niveau supérieur se trouvent un ensemble de librairies écrites en C/C++ jouant le rôle d'un middleware (on en cite le système de bibliothèque C, les médiathèques, le SGL, etc.). C'est sur cette couche que se greffe l'Android Runtime, comprenant la machine virtuelle de Java et ses bibliothèques. Vient ensuite la plateforme logicielle, nommée aussi framework de développement, écrite en java et permettant de mutualiser les ressources entre applications Java. Elle offre aux développeurs la possibilité de produire des applications diverses et innovantes à travers un ensemble d'API. Enfin, à un niveau plus supérieur se situe un ensemble d'applications sous forme de paquets ***APK (Android PacKage)***. Les applications fournies par Android sont telles qu'un navigateur web, un client-email, un calendrier, un gestionnaire de contacts, etc. La figure I.2, illustre l'architecture du système *Android*. **[03]**

Les couches constituantes cette architecture sont décrites comme suit: **[N1]**

- ***Linux Kernel (Linux le cœur du système)***

La couche la plus basse de l'architecture repose sur un noyau linux 2.6. Cependant, il ne s'agit pas d'un linux proprement dit car X-windows n'est pas implémenté (gestion de fenêtrage) tout comme la librairie glibc qui n'est pas supportée (Android utilise une bibliothèque libc customisée nommée Bionic).

Le choix d'un noyau linux a été réalisé pour sa stabilité et sa performance, pour son modèle de sécurité, pour ses capacités d'abstraction avec le matériel et enfin pour son aspect open source et communautaire fort.

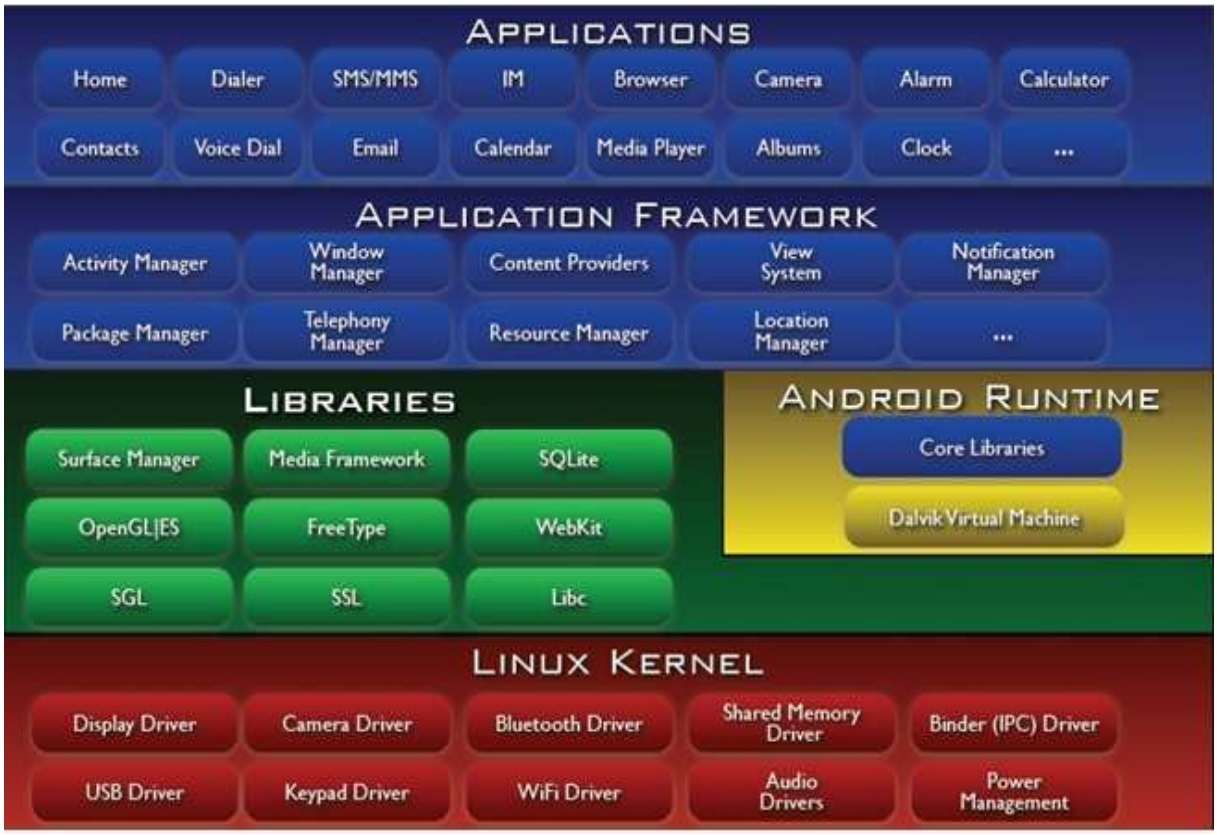

Figure I.2 : *Architecture d'Android*

A noter enfin que le noyau linux a été patché offrant ainsi de nouvelles fonctionnalités. Le tableau ci-dessous liste quelques exemples de patchs :

Alarm	Mise en place de timers permettant par exemple de réveiller l'appareil quand il est en veille.
Ashmem	Permet le partage de la mémoire entre application.
Binder	Permet la communication entre processus.
Logger	Offre un système de logs.
Wakelogs	Permet la gestion de l'alimentation.

- ***Les librairies***

Ecrites en C/C++, les librairies sont utilisées par les composants du système Android et sont utilisables par le développeur via le Framework Android, et voici quelques-unes :

- Surface Manager gérant l'accès au sous-système d'affichage ;
- Media Framework permettant la gestion de fichier multimédia (Audio et vidéo) ;

- SQLite le moteur de base de données relationnel ;
- OpenGL la bibliothèque graphique 3D ;
- FreeType gérant les bitmap et le rendu des polices ;
- WebKit permettant la navigation internet ;
- SGL le moteur graphique 2D ;
- SSL gérant le protocole SSL ;
- Libc gérant la librairie C.

❖ ***Android Runtime (Le moteur d'exécution linux)***

Chargé d'exécuter les applications, le moteur android comporte deux éléments :

- Des bibliothèques de base.
- La machine virtuelle Dalvik.

Les bibliothèques de bases fournissent les fonctionnalités du langage JAVA (langage utilisé pour le développement d'applications Android) et des bibliothèques spécifiques à Android.

Au lieu et place d'utiliser une JVM (Java Virtual Machine) classique, Android dispose de sa propre machine virtuelle nommée Dalvik. Basé sur une architecture de registre Dalvik, exécute les applications Android une fois celles-ci compilées au format requis (.dex) à l'aide de l'outil dx (Un fichier java étant compilé en .class, ce fichier étant ensuite compilé en .dex)

Il est important de noter que chaque application Android s'exécute dans son propre processus ayant sa propre instance de Dalvik car la VM Android a été écrite à des fins de multiples instanciations sans perte de performances.

Enfin, Dalvik s'appuie sur le noyau linux pour la réalisation des taches dites de bas niveau telles que le threading, la gestion des processus et de la mémoire.

❖ ***Le Framework de développement***

Cette couche intéresse tout particulièrement les développeurs car elle leur permet de créer des applications à l'aide d'une plateforme ouverte.

Utilisant le langage JAVA le framework met à disposition un ensemble de classes utiles à la création d'applications sans oublier la mise à disposition de classes et méthodes

permettant l'accès au matériel, à la gestion de l'interface graphique et aux ressources de l'application.

Enfin, notons que le Framework Android se compose des services applicatifs suivants :

- Activity Manager, gérant le cycle de vie des Activités (Activités) ;
- Views permettant de créer les interfaces graphiques ;
- Notification Manager fournissant un mécanisme d'envois de message aux utilisateurs ;
- Content Providers, permettant aux applications de partager des données entre elles ;
- Resource Manager, offrant l'externalisation de ressources telles que les chaines de caractères pour la notion d'internationalisation, de style, de menus…etc.

❖ ***La couche applicative***

C'est la Couche la plus haute de l'architecture, celle-ci contient l'ensemble des applications natives, tierces ou développées présentes sur l'appareil.

Toutes les applications présentes dans cette couche sont, comme nous l'avons vu précédemment, exécutées par le moteur d'exécution Android.

I.5 Cycle de vie d'une application Android

Les technologies mobiles prennent de plus en plus de place sur le marché. Les Smartphones sont considérés comme des petits ordinateurs dotés d'un système d'exploitation s'appuyant sur un noyau Linux. Cependant, ils diffèrent des ordinateurs classiques par le cycle de vie d'une application. Sous Android, une application est composée d'une ou plusieurs activités. Une activité est la base d'un composant pour la création d'interfaces utilisateur. Afin de faciliter la cinématique de l'application, il est préconisé de n'avoir qu'une interface visuelle par activité. **[06]**

Les différentes étapes de l'organigramme illustré par la *Figure I.3* sont :

- Démarrages de l'Activité : la méthode onCreate est appelée. Pendant l'utilisation d'une activité, l'utilisateur presse la touche Accueil, ou bien l'application

téléphone, qualifiée comme prioritaire et qui interrompt son fonctionnement par un appel téléphonique entrant.

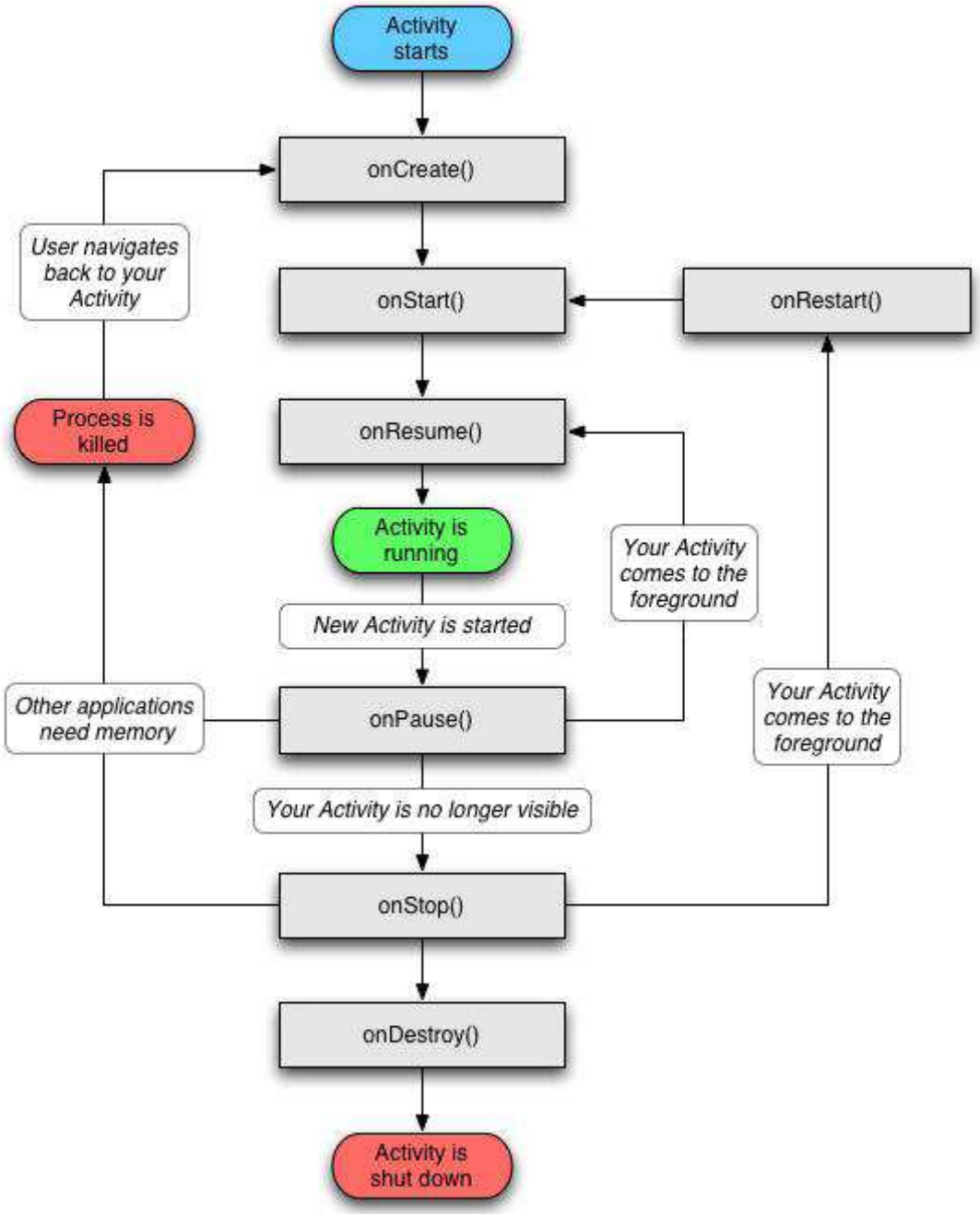

Figure I.3 : *Organigramme de cycle de vie d'une application Android*

- Une fois l'appel téléphonique terminé, le système réveille l'activité précédemment mise en pause (appel de *onRestart*, *onStart*).

- L'activité reste trop longtemps en pause, le système a besoin de mémoire, il détruit l'activité (appel de *onDestroy*).

- onPause et onResume rajoutent un état à l'activité, puisqu'ils interviennent dans le cas d'activités partiellement visibles, mais qui n'ont pas le focus. La méthode onPause implique également que la vie de cette application n'est plus une priorité pour le système. Donc si celui-ci a besoin de mémoire, l'activité peut être fermée. Ainsi, il est préférable, lorsque l'on utilise cette méthode, de sauvegarder l'état de l'activité dans le cas où l'utilisateur souhaiterait y revenir avec la touche Accueil.

I.6 Conclusion

Dans ce chapitre, nous avons fait une étude du Systèmes d'Android tout en présentant un bref historique, les fonctionnalités que nous pouvons trouver sur ce système d'exploitation et son architecture, à savoir les principaux composants du système. Ceci nous a permis de comprendre les capacités de ce système et de les exploiter pour proposer une extension au système GUAMB.

Chapitre II : Etude du système GUAMB

II.1 Introduction

Vu l'importance et la nécessité d'une gestion informatique des différents services d'une organisation, l'Université de Bejaia a concrétisé cet intérêt en créant la section Système d'Information, au sein du Centre de Calcul (*CSRICTED*). Cette section prend en charge le développement d'un nouveau système d'informations pour l'université.

Dans ce chapitre, nous présenterons l'idée principale du système GUAMB, son architecture logicielle, la méthode de travail adoptée par la section système d'information, et enfin, notre apport ou contribution à ce système et qui consiste à ajouter un accès au système via des Smartphones équipé d'un système Android.

II.2 Objectif du GUAMB

L'objectif initial du système GUAMB est d'élaborer une seule base de données pour tous les services de l'université : Scolarité, Stock, Personnel et Budget. Et de fournir un accès réseau à cette base de données à travers un ensemble d'applicatifs.

Vu la nature de l'application, la cellule de développement ont opté pour des langages de programmation qui facilite le développement des applications réseau : le langage *PHP* pour le développement d'une interface web dynamique et le langage Java vu sa puissance et sa portabilité pour la quasi-totalité des systèmes d'exploitation. Le langage *XML* est utilisé pour définir un langage de réponse entre le serveur et les clients.

II.3 Architecture logicielle

Le système GUAMB est constitué de deux volets : un site web dynamique élaboré en langage PHP et accédant à une base de données MySQL, et d'une application conçue selon l'architecture *Client/serveur* à trois niveaux : le client qui représente une interface graphique Java/Swing, le serveur est incarné par une couche de scripts *PHP* et la base de données.

Les deux composantes du système, à savoir le site web et l'application Java-Swing accèdent à la même base de données et utilise, partiellement, la même couche Métier en PHP. Par conséquent, les scripts *PHP* de réponse au requêtes *HTTP* sont de deux types : ceux qui répondent au site web et ceux qui répondent à l'application Java-Swing. Dans ce qui suit, nous nous concentrons sur l'application Java Swing.

Cette application possède l'architecture illustrée dans la figure suivante :

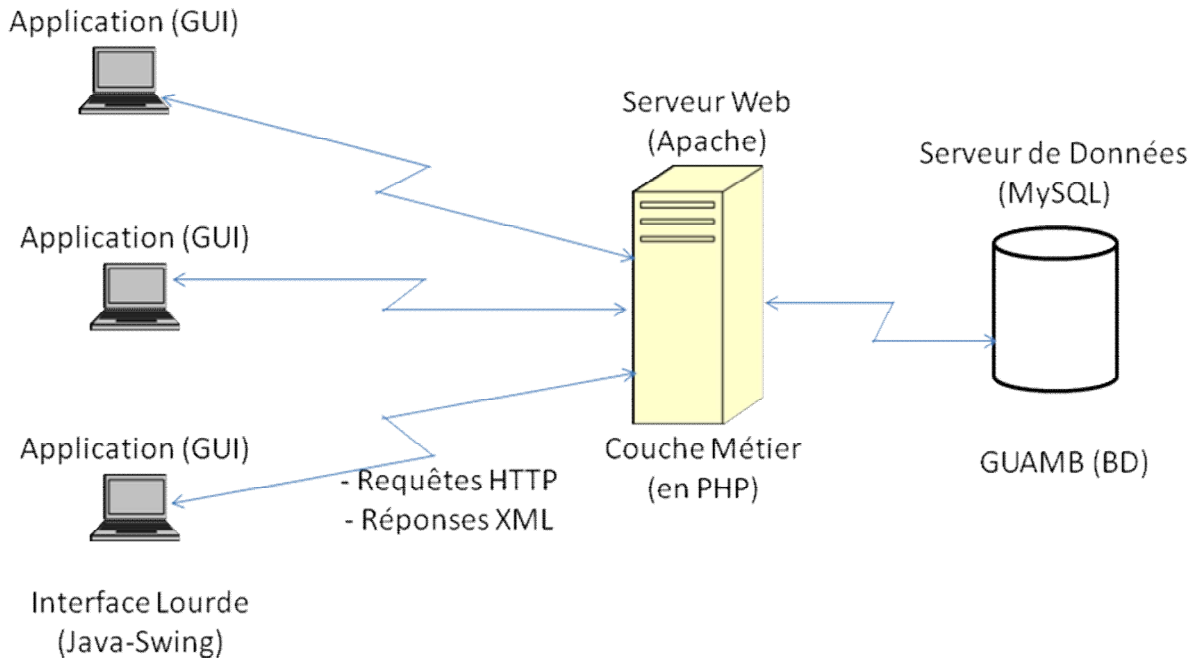

Figure II. 1 : *Architecture et Déploiement du Système GUAMB*

II.3.1 Communiquer un client Java avec un serveur PHP

Comme illustré dans la figure ci-dessus, la partie Java-Swing communique avec le serveur à travers des requêtes HTTP, ces requêtes sont reçues par un serveur Web (le serveur Apache qui est utilisé par la cellule de développement) qui déclenche le script PHP adéquat pour répondre à cette requête. Le script peut éventuellement accéder à la base de données pour recueillir des données et construit une réponse XML. Une fois la réponse est reçue, l'application Java parse l'XML reçu afin de construire les objets correspondant à cette réponse.

II.3.2 Objets Métiers

Les objets métiers permettent de représenter la base de données au niveau du langage de programmation cible, de tels sorte que chaque table dans la base de données est incarnée par une classe au niveau du langage utilisé. Chaque instance d'une classe de la couche métier représente un tuple au niveau de la table correspondante, et les opérations usuelles sur la base de données comme l'insertion, la mise à jour et la sélection de tuples seront réalisés par des méthodes.

Ceci permet de séparer la couche qui communique avec la base de données des autres couches de l'application.

La figure suivante montre les différentes couches de système GUAMB et indique la position de la couche métiers.

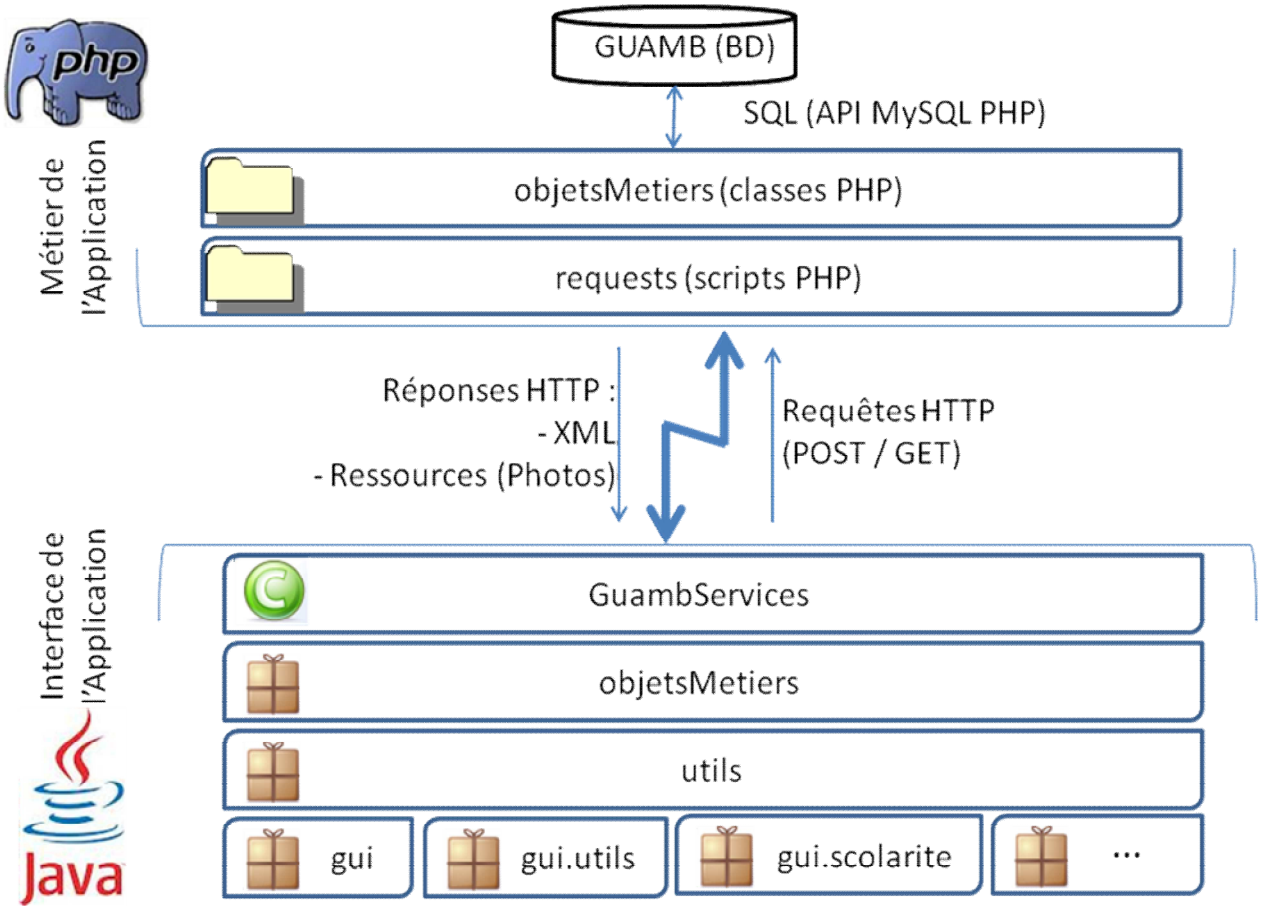

Figure II. 2 : *Structure du code source du système GUAMB*

II.4. Extension du système aux systèmes Android

Le noyau de notre travail consiste à développer une application pour les systèmes Android et qui permet de réaliser certaines tâches de gestion du personnel de l'université de Bejaia.

Au lieu de proposé une application à zéro, il est plus judicieux de se baser sur ce que ce fait au niveau de la section système d'information de l'université et de proposer une extension à l'application GUAMB. Ceci nous évite de refaire la phase de la conception de la base de données et on se concentre premièrement, sur le moyen de faire communiquer un système Android avec un serveur Socket qui joue un rôle d'intermédiaire entre le client et le SGBD MySQL, et deuxièmement, de concevoir une application Android de telle sorte qu'elle suit le même schéma structurel de l'application GUAMB, c'est-à-dire, de réutiliser les objets

métiers pour rendre l'application plus modulaires et facile à maintenir et développer de nouvelles fonctionnalités.

II.5. Conclusion

Dans ce chapitre, nous avons présenté l'architecture globale du système GUAMB, les techniques utilisées pour mettre en place une application réseau en se basant sur les deux langages *PHP* et *Java*. Nous avons proposé une extension à ce système permettant l'accès à la base de données à travers des applications *Android*. Dans les deux chapitres suivants, nous allons voire la conception et l'implémentation de cette extension.

Chapitre III : Analyse et Conception

III.1 Introduction

Dans ce chapitre, nous suivrons les étapes nécessaires pour concevoir une application mobile pour système Android pour la gestion du personnel. Pour cela, dans cette conception, nous nous sommes basés sur la méthode UP (processus unifié), cette dernière qui utilise UML comme langage de modélisation et qui offre une souplesse remarquable qui s'exprime par la possibilité d'obtenir des modèles de systèmes reflétant la réalité en utilisant des diagrammes, et qui est compatible avec les langages de programmation.

III.2 Définition processus unifié

Le processus unifié est un processus de développement logiciel : il regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel. **[05]**

Le processus de développement (UP) d'une application passe par les étapes suivantes :

1) Analyse et spécification des besoins ;
2) Conception ;
3) Implémentation ;
4) Test.

Dans ce chapitre, on développera uniquement les deux premières étapes, à savoir l'analyse et la spécification des besoins et la conception.

III.2.1 Analyse et spécification des besoins

L'analyse et la spécification des besoins représentent la première phase du cycle de développement d'un logiciel. Elle sert à identifier les acteurs réactifs du système et leur associer chacun l'ensemble d'actions avec lesquelles il intervient dans l'objectif de donner un résultat optimal et satisfaisant au client.

III.2.1.1 Spécification des besoins

III.2.1.1.a Description de l'application

Notre projet consiste au développement d'une application sous Android pour la gestion du personnel de l'université de Bejaia. Cette gestion a été définie auparavant par la section Système d'Information, et notre tâche consiste à faire une extension de cette gestion sur le terminal mobile fonctionnant sous Android.

III.2.1.1.b Les besoins fonctionnels

Les services proposés par notre application se résument en deux actions majeures :

b.1) Authentification

L'application doit permettre une authentification pour déterminer le rôle de l'utilisateur et la sécurité des données.

b.2) Gestion des fonctionnaires

Le système doit donner à l'utilisateur la possibilité de gérer des fonctionnaires soit :

a) Création d'un nouveau fonctionnaire ;
b) Lister l'ensemble des fonctionnaires ;
c) Supprimer / Désactiver un fonctionnaire ;
d) Modifier / Mise à jour d'un fonctionnaire.

III.2.1.1.c Les besoins non fonctionnels

Les besoins non fonctionnels décrivent toutes les contraintes auxquelles est soumis le système pour sa réalisation et son bon fonctionnement.

1. Existence d'un réseau *Wifi* (Réseau sans fil) ;
2. Existence d'un serveur de données avec le *SGBD MySQL* ;
3. Ergonomie et souplesse : l'application doit offrir une interface conviviale et ergonomique exploitable par l'utilisateur en envisageant toutes les interactions possibles à l'écran du support tenu ;
4. Efficacité : l'application doit être fonctionnelle indépendamment de toutes circonstances pouvant entourer l'utilisateur ;
5. L'application doit optimiser les traitements et l'utilisation de ses ressources (autonomie batterie, mémoire disponible….) afin de ne pas épuiser ces dernières dans des calculs grands et inutiles.

III.2.1.2 Analyse des besoins

a) Diagramme de cas d'utilisation

- ***Identification des acteurs***
  - ✓ **Les acteurs** : Utilisateur inter-département, Agent du personnel.
    - Administrateur ou Agent du personnel : il alimente la base de données du système en introduisant les ressources nécessaires (la mise à jours enseignants, ATS).
    - Utilisateur inter-département : il peut consulter la liste des fonctionnaires appartenant au département adéquat.

- ***Identification des cas d'utilisations***
  - ✓ ***Cas d'utilisation authentification***

Saisir login et mot de passe : seulement les utilisateurs autorisés peuvent accéder au système cependant, chacun d'eux à un certain nombre de privilèges. C'est pour cela, qu'il faut au début s'identifier en donnant son login, son mot de passe.

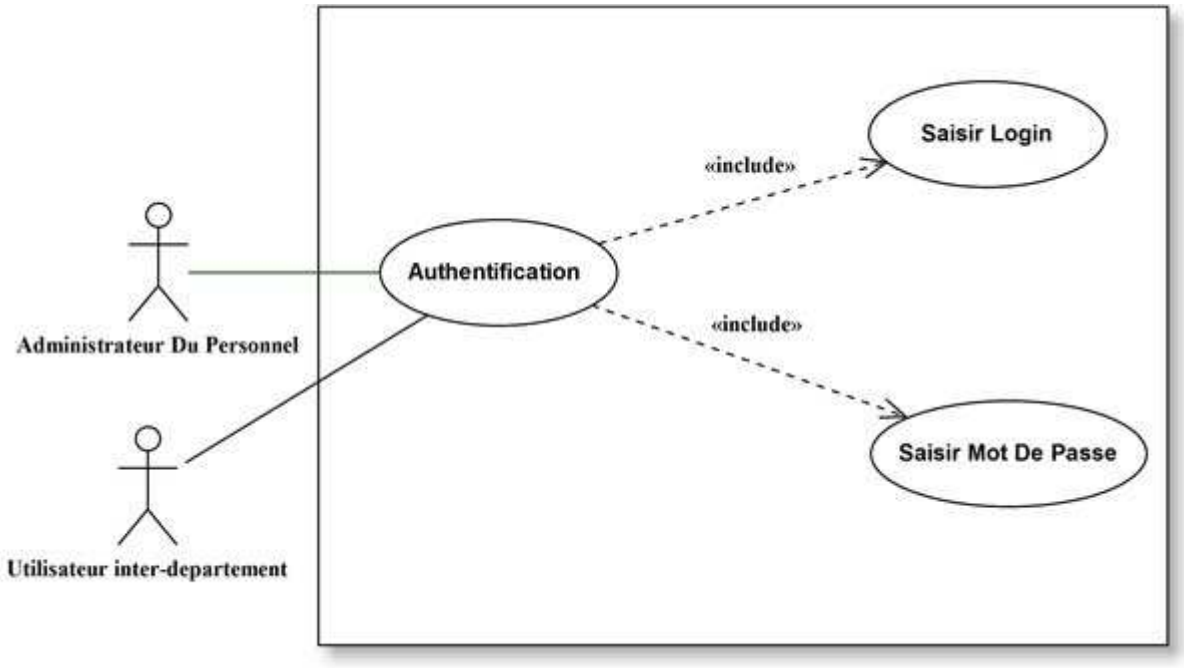

Figure III. 1 : *Diagramme cas d'utilisation authentification.*

Scénario nominal :

1. L'application, après son lancement, affiche un manque pour introduire un login et mot de passe.
2. L'utilisateur introduit son login et son mot de passe.
3. L'application réalise un premier contrôle sur la validité des champs :

a. S'il y a un champ vide, l'application affiche un message d'erreur et on revient vers 2.

b. Si le format de mot de passe n'est pas valide (le mot de passe doit avoir au minimum 6 caractères, dont 4 alphabétiques et 2 non-alphabétique).

4. L'application réalise un deuxième contrôle sur la validité du login/mot de passe :

a. Si le login et/ou mot de passe sont erroné, l'application affiche un message d'erreur et on revient à l'étape 2.

b. Sinon (le login et mot de passe sont valides), l'application affiche l'interface principal.

✓ ***Cas d'utilisation gestion des fonctionnaires***

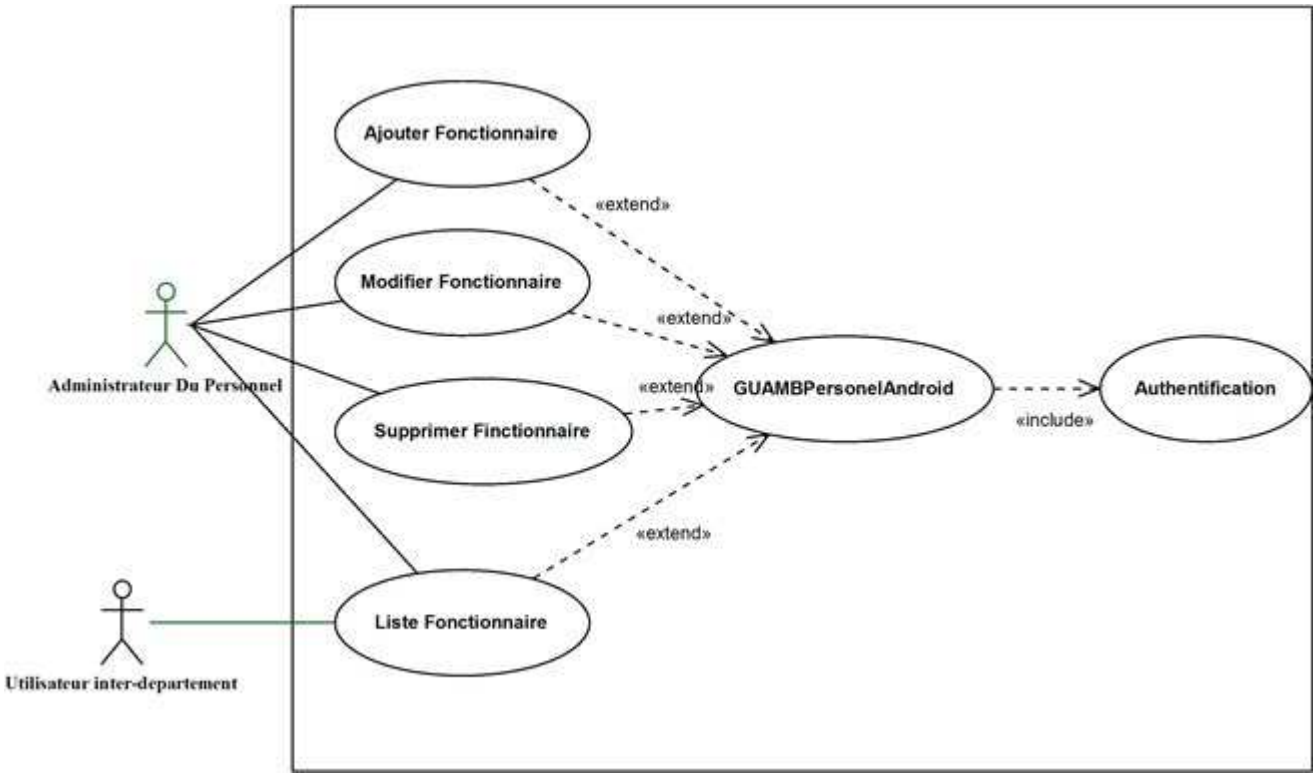

Figure III. 2 : *Diagramme de cas d'utilisation gestion des fonctionnaires.*

Scénario nominal :

1. Après l'authentification de l'utilisateur par le système ;

2. Le système doit renvoyer les informations correspondantes à la session en cours ;

3. L'utilisateur de système doit pouvoir gérer les opérations d'ajout, de modification, et de suppression ;

4. L'utilisateur soumet les modifications au système ;

5. Le système enregistre les modifications et signale le succès de l'opération.

III.2.2 Conception

Nous passons, à travers cette section, à décrire la conception élue pour réaliser convenablement le travail demandé.

Pour ce faire, nous choisissons de donner en premier lieu une idée sur les composants d'une application Android afin de comprendre sa structuration et par la suite choisir le modèle d'architecture adéquat pour sa réalisation.

Après avoir parlé de la conception générale, nous détaillons dans un second lieu la conception de notre application au moyen des diagrammes de collaboration, de séquences et de classes.

III.2.2.1 Conception générale

- ***Les composants d'une application Android***

Une application Android consiste en un assemblage de composants liés via un fichier de configuration, qui présentent en quelque sorte les briques sur lesquelles se repose l'application. Ces concepts fondamentaux à préciser sont :

- **Les vues (**Views**) :** les vues sont les composants basiques de l'interface graphique. Ce sont les éléments de l'interface que l'utilisateur voit et sur lesquels il agit. C'est de la classe View qu'héritent les widgets (exemple de composants graphiques tel que les boutons), les layouts(le plan sur lequel on organise et on place les composants graphiques) et tous les composants graphiques servant à la création d'une interface graphique interactive.
- **Les contrôles :** c'est bien la classe des composants graphiques cités dessus. Les contrôles sont tels que les boutons, les champs de saisie de texte, les cases à cocher, etc.
- **Les activités :** une Activité représente la fenêtre qui sera affichée à l'utilisateur. C'est un ensemble de vues et de contrôles composant une interface logique. Elle permet également de gérer des fonctionnalités telles que l'appui sur une touche, l'affichage de messages, etc. Ce concept repose essentiellement sur l'interaction de l'utilisateur avec l'écran.

- **Les ressources :** chaque application Android a ses propres fichiers ressources. C'est dans ces fichiers que seront puisés les textes, les images, les couleurs, etc.
- **Le fichier de configuration :** c'est fichier auto généré par l'application, qui lui est indispensable. C'est un fichier XML appelé **« AndroidManifest »** qui décrit le point d'entrée de l'application (le code à exécuter), les composants du projet ainsi que les permissions nécessaires pour l'exécution du programme.

Une illustration explicative de ces concepts est représentée par le schéma de la figure ci-dessous :

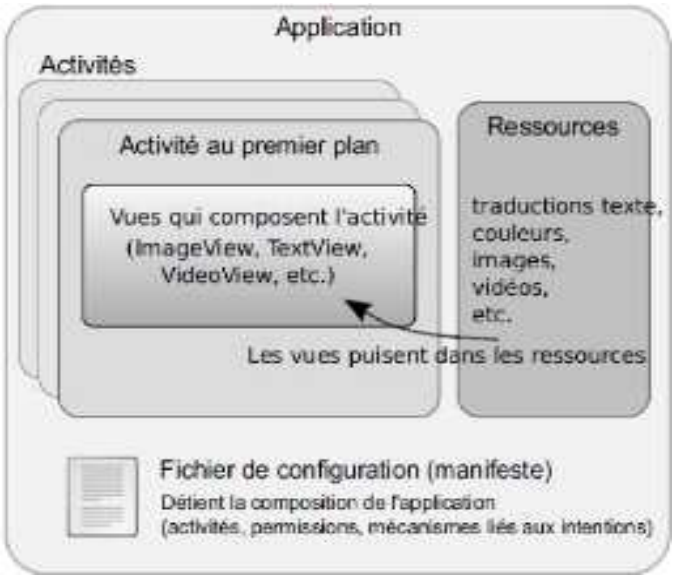

Figure III.3 : *Composition d'une application Android.*

III.2.2.2 Conception architecturale

Il est primordiale à la conception de tout système informatique de choisir le modèle d'architecture qui lui sera adéquat pouvant assurer un bon fonctionnement, des meilleurs performances ainsi que la réutilisation et l'interconnexion fiable de ce système avec d'autres.

C'est à cet effet que nous optons pour le modèle MVC qui sera également très pratique pour gérer l'interaction entre les différents composants de notre application Android. Nous décrivons cette architecture dans la section suivante.

III.1.2.2.a Architecture MVC

L'architecture MVC (modèle, vue et contrôleur) est une architecture à trois couches utilisées pour la programmation de divers types d'applications, notamment les applications

client/serveur et avec interface graphique. Il tire sa puissance de son concept de base qui est la séparation des données (modèle), de l'affichage (vue) et les traitements (contrôleur). Ces trois couches sont décrites comme suit : **[N2]**

- **Modèle :** il correspond aux données stockées généralement dans une base de données. Dans un langage orientée objet ces données sont exploitées sous forme de classes. Le modèle peut aussi agir sur la vue en mettant à jour ses données.
- **Vue :** ne contenant que les informations liées à l'affichage, la vue se contente d'afficher le contenu qu'elle reçoit sans avoir connaissance des données. En bref, c'est l'interface homme machine de l'application.
- **Contrôleur** : le contrôleur sert de base à récupérer les informations, de les traiter en fonction des paramètres demandés par la vue (par l'utilisateur), puis de renvoyer à la vue les données afin d'être affichées. C'est donc l'élément qui va utiliser les données pour les envoyer à la vue.

L'interaction entre ces trois couches est décrite à l'aide de la figure III.4

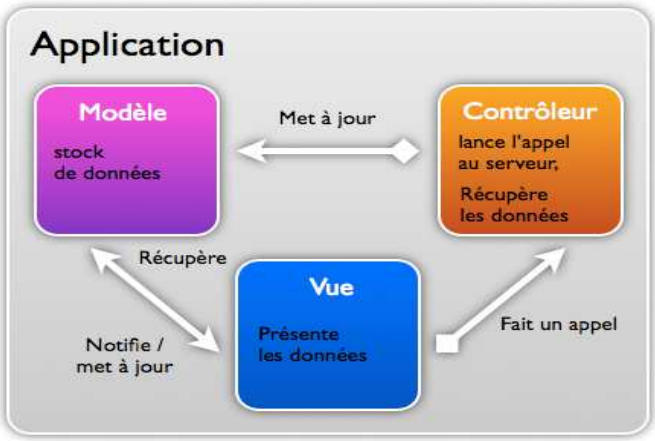

Figure III.4 : *Architecture MVC*

Les avantages apportés par l'architecture MVC sont :

- ✓ La séparation des données de la vue et du contrôleur (ce qui permet une conception claire et efficace de l'application).
- ✓ Une indépendance des données, de l'affichage et des actions (ce qui donne plus de souplesse pour la maintenabilité et l'évolutivité du système).
- ✓ Un gain de temps de maintenance et d'évolution de l'application.

III.2.2.2.b) Approche Orientée Objet

La programmation orienté objet est définie comme un paradigme de programmation informatique basé sur l'interaction de briques appelées objets, qui sont eux-mêmes des abstractions d'objets réels, via leurs relations et les propriétés qui leur sont accordées.

Ce paradigme assure:

- une modularité des programmes résolvant le problème de la complexité des codes ;
- une vitesse d'exécution plus rapide ;
- une souplesse de réutilisation et d'évolution du code.

III.2.2.3 Conception détaillée

III.2.2.3.a) Diagramme de collaboration

Il permet de visualiser des objets, leurs relations et l'ordonnancement des appels de messages.

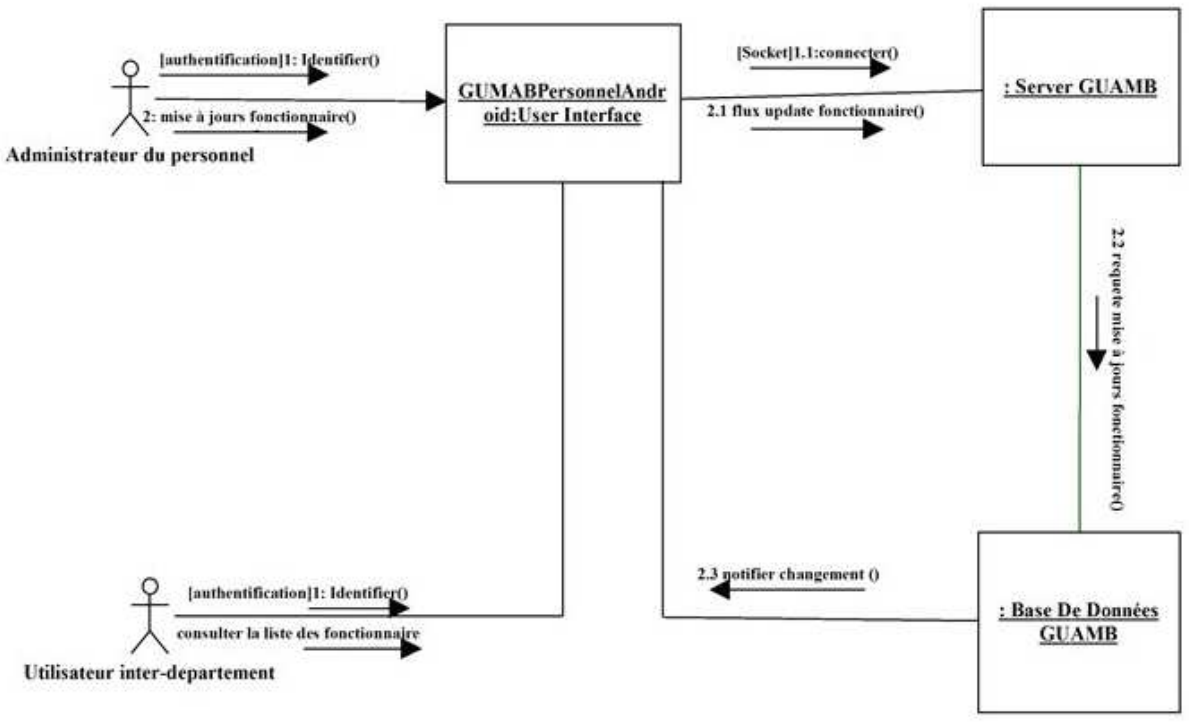

Figure III. 5 : *Diagramme de collaboration.*

III.2.2.3.b) Diagrammes de séquences

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation UML.

Dans ce qui suit, nous présentons le diagramme de séquence pour chaque cas d'utilisation dans notre système.

✓ ***Diagramme de séquence de cas d'utilisation «authentification échec» :***

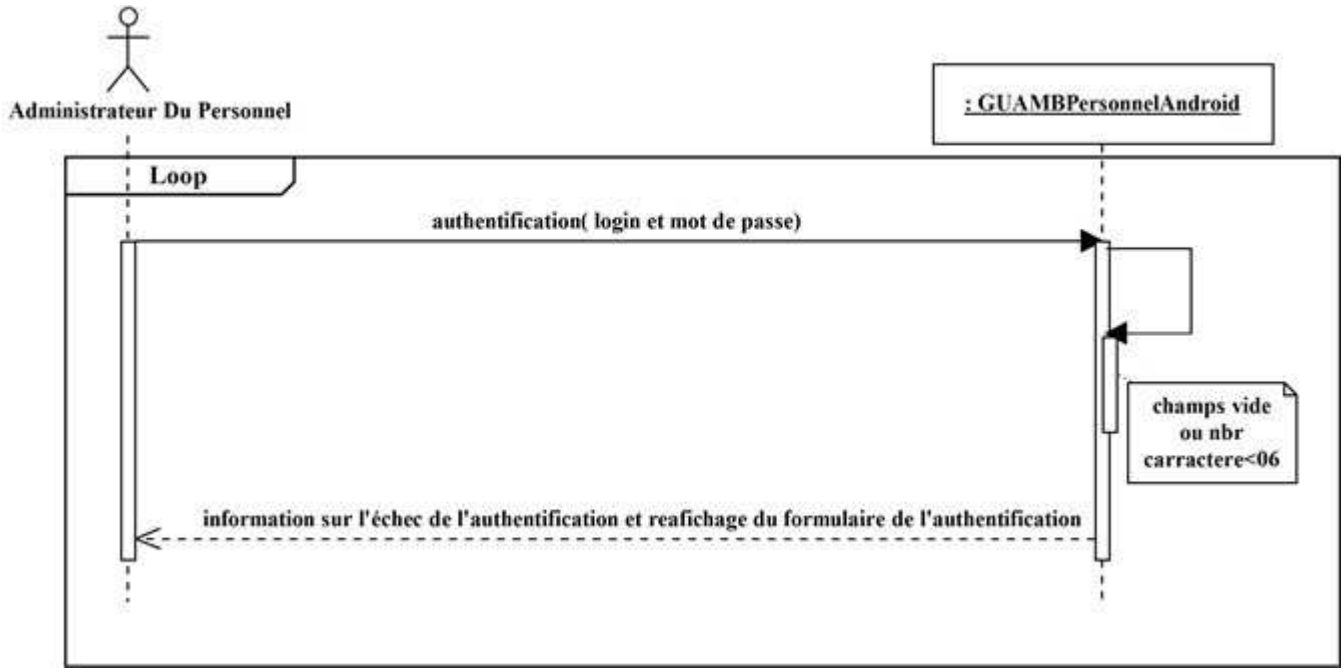

Figure III. 6 : *Diagramme de séquence de cas d'utilisation : authentification échec.*

✓ ***Diagramme de séquence de cas d'utilisation «authentification succès»***

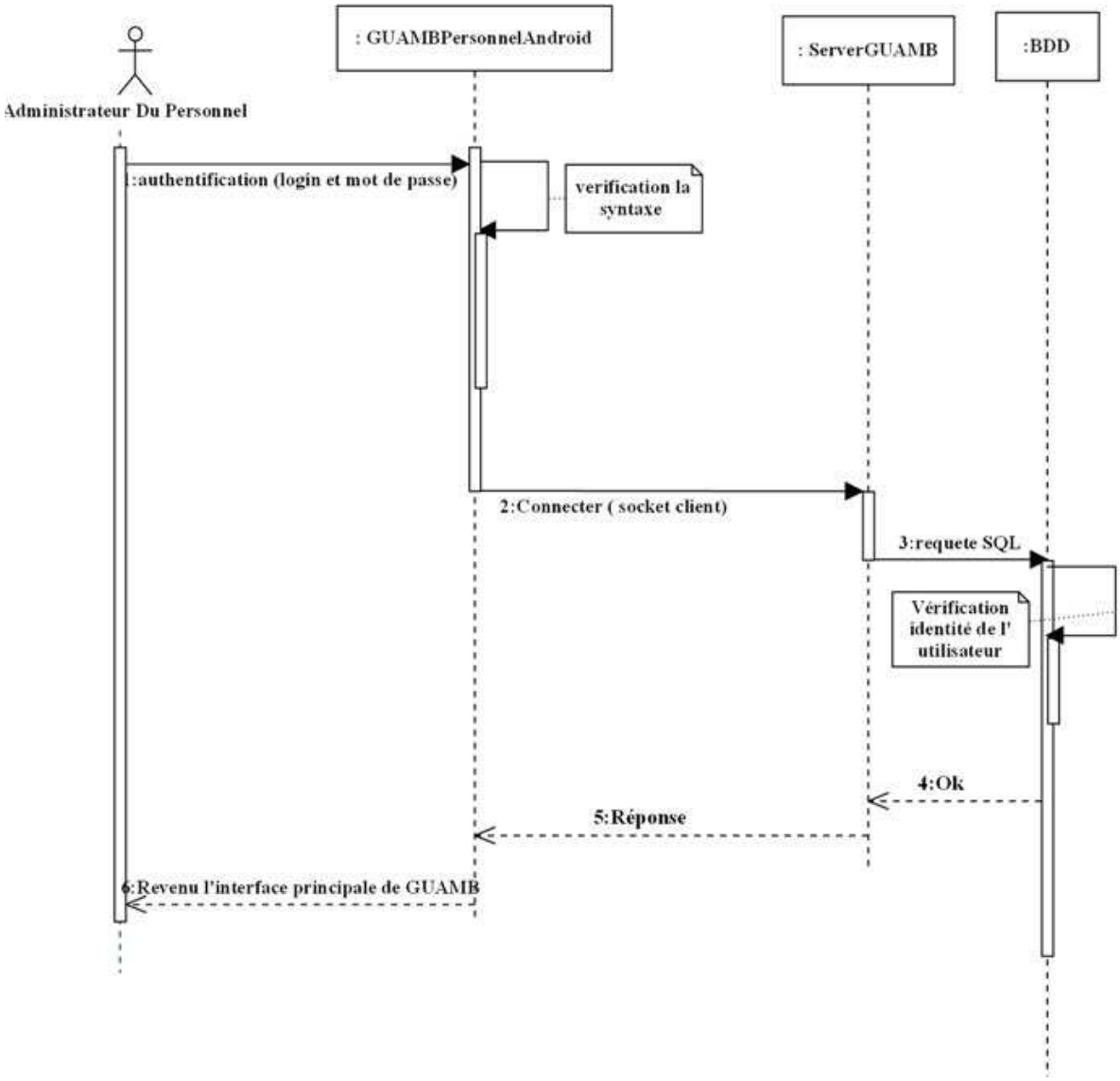

***Figure III. 7** : Diagramme de séquence de cas d'utilisation : authentification succès.*

✓ ***Diagramme de séquence de cas d'utilisation «ajouter fonctionnaire»***

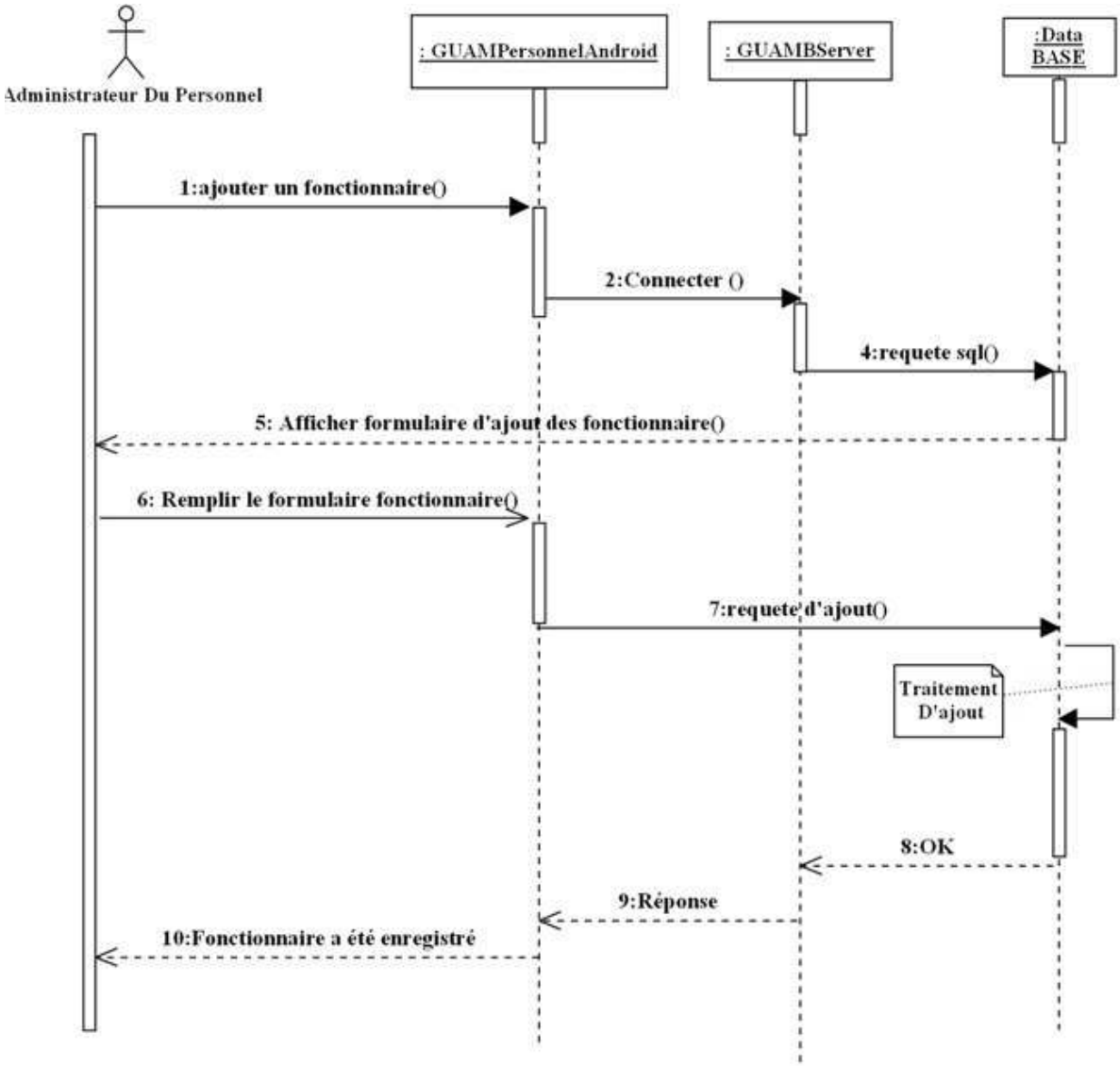

Figure III. 8 : *Diagramme de séquence de cas d'utilisation : ajouter fonctionnaire.*

- ✓ ***Diagramme de séquence de cas d'utilisation «modifier fonctionnaire»***

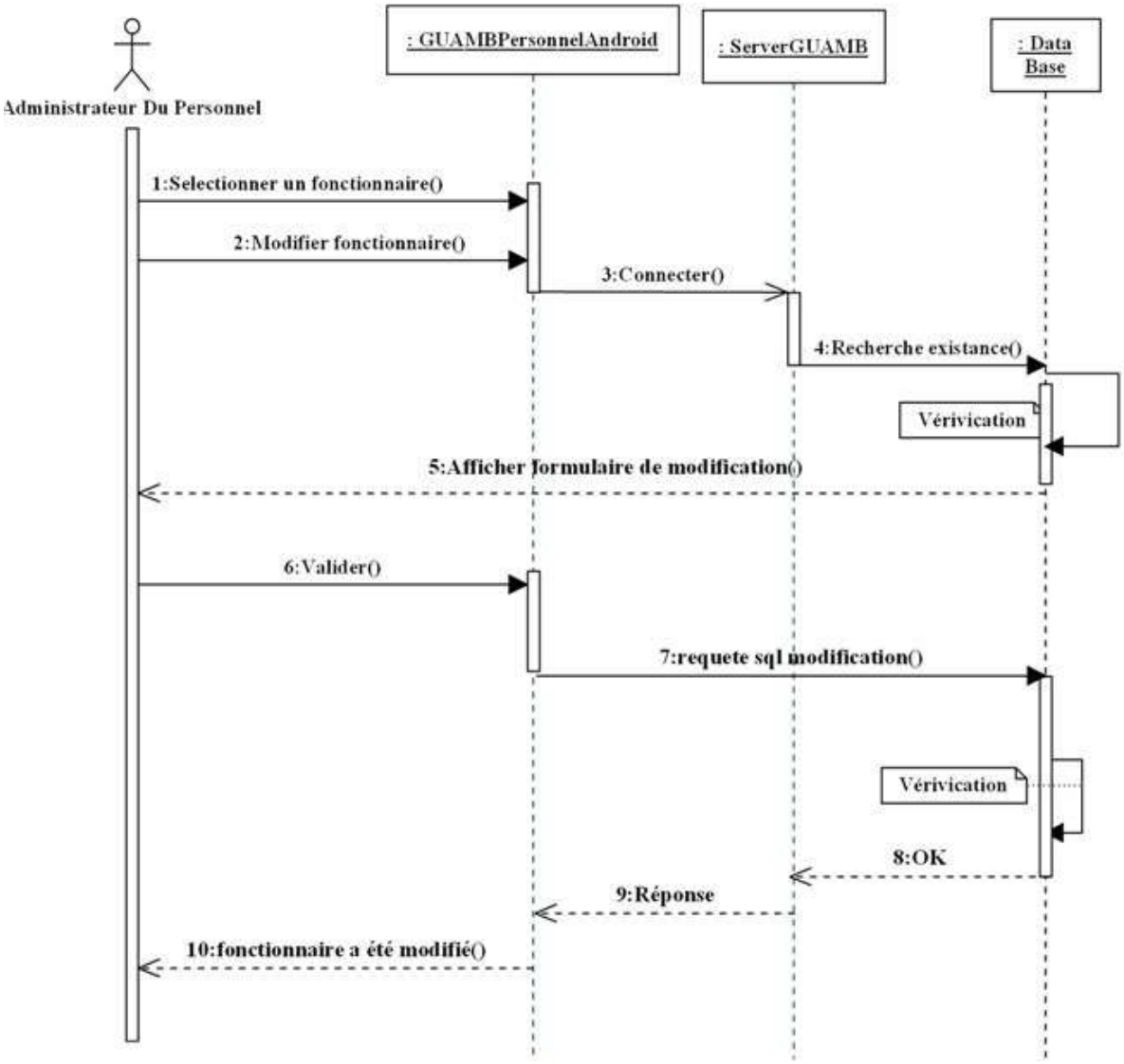

Figure III. 9 : *Diagramme de séquence de cas d'utilisation : modifier fonctionnaire.*

- ✓ ***Diagramme de séquence de cas d'utilisation « supprimer fonctionnaire »***

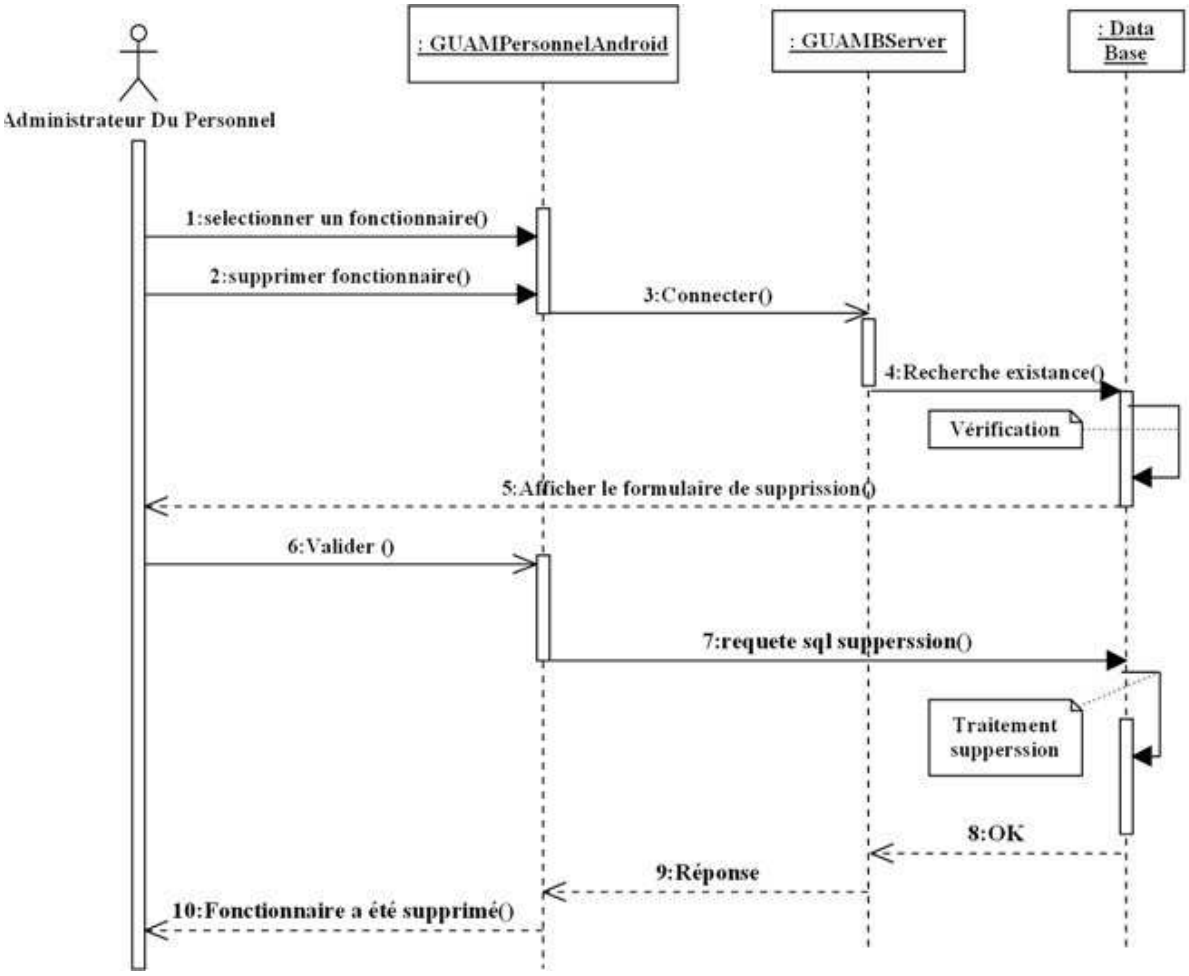

Figure III. 10 : *Diagramme de séquence de cas d'utilisation : Supprimer fonctionnaire.*

III.2.2.3.c) Diagramme de classes

Notre application est constituer de plusieurs packages contenant plusieurs classes. On se focalise sur le package *ObjetMetiers* qui représente les classes du domaine et qui permet d'interagir avec la base de données. La *figure III.11* illustre l'ensemble des classes du domaine :

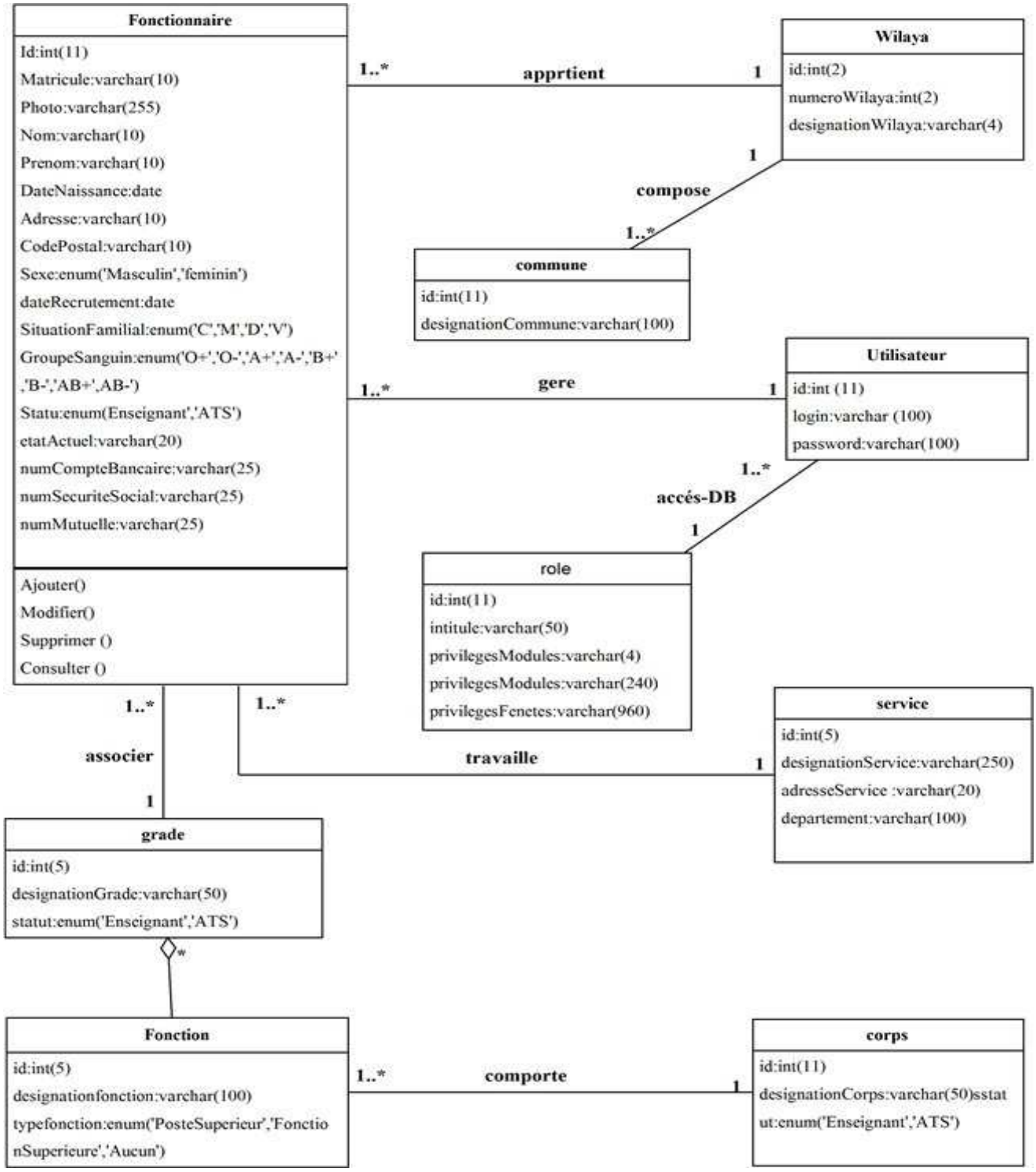

Figure III. 11 : *Diagramme de classes de la couche Model*

III.3. Conclusion

A l'issu de cette étape, nous avons pu exprimer clairement les objectifs attendus du futur système à concevoir, ainsi que les différents diagrammes UML afin de bien entamé l'implémentation et la réalisation de notre application.

Chapitre IV : Implémentation de l'application GUAMB-Personnel-Android

IV.1 Introduction

Dans ce chapitre, nous présentons l'architecture sur laquelle nous avons développé notre application, les différents outils utilisés pour l'implémentation ainsi que les outils de mise en œuvre de notre projet accompagné de quelques interfaces illustrant les fonctionnalités de l'application développée.

IV.2 Environnement de travail

IV.2.1 Environnement matériel

IV.2.1.1 Architecture matérielle

GUAMB-Personnel-Android est une application embarquée qui se connecte à un serveur de bases de données distant, via le réseau wifi local de l'université, afin de récupérer et mettre à jour les données. Ce qui nécessite aussi l'intégration d'un serveur entre l'application client et le serveur de bases de données. D'où l'architecture de notre application est à 3 niveaux (*architecture 3-tiers*), elle est partagée entre :

- Le client Android : Conteneur d'application et demandeur de ressources,
- Le serveur GUAMB-Mobile : Vu que les données seront communiquées entre deux environnements hétérogènes, le rôle principale du serveur est de gérer la communication entre les clients Android et le serveur de base de données,
- Le serveur de base de données fournis les données au serveur GUAMB-Mobile.

La *figure IV.1* schématise l'architecture matérielle de notre application.

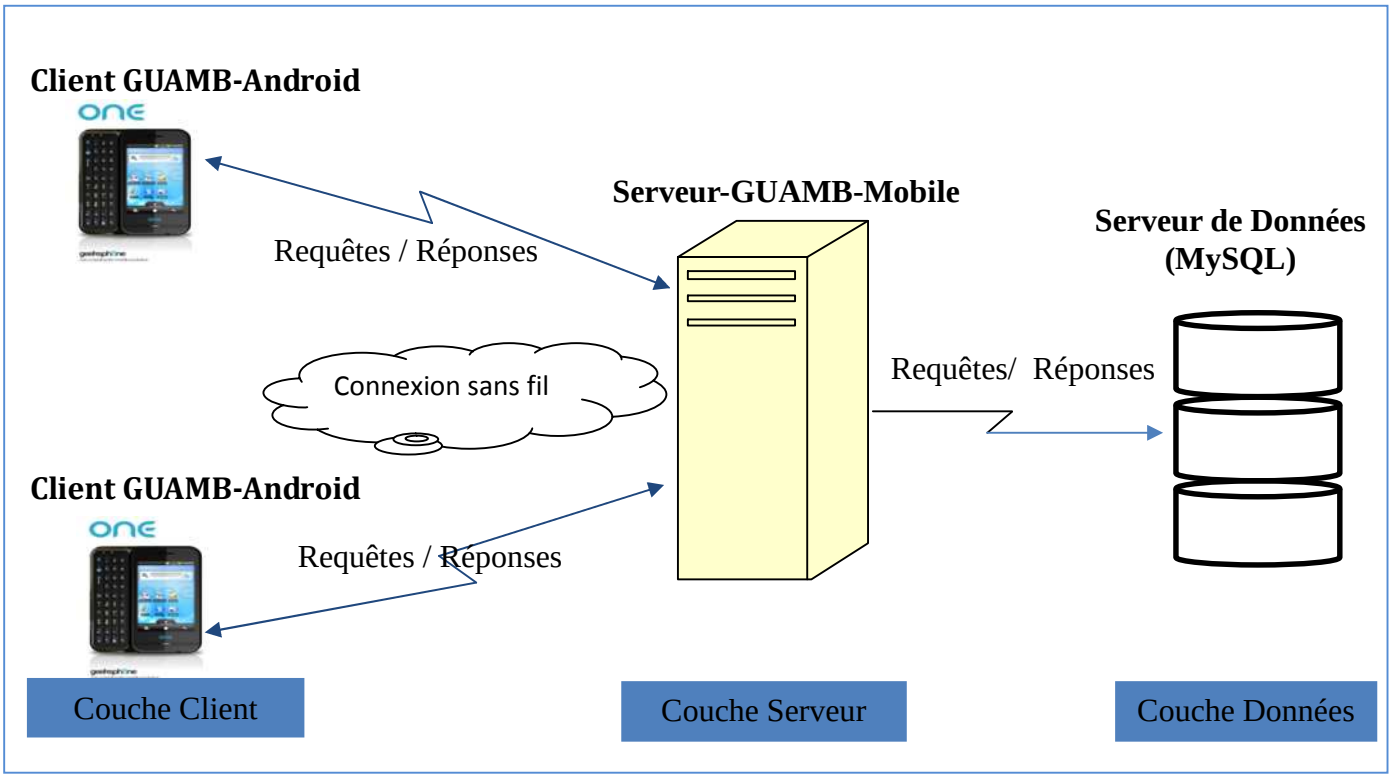

Figure IV. 1 : *Architecture matériel du système.*

IV.2.1.2 Matériels utilisés

Nous avons élaboré ce travail sur un PC dont la configuration est la suivante :

- ✓ PC Portable: Toshiba - RAM: 4 GO
- ✓ Disque dur: 320 GO
- ✓ Microprocesseur : Intel(R) Core(TM)2 Duo 3.00 GHz.

IV.2.2 Environnement logiciels

L'environnement logiciel utilisé s'illustre en :

- ✓ Un Système d'exploitation Linux Ubuntu 11.10.
- ✓ Pacestar UML Diagrammer : utilisé pour le traçage des différents diagrammes.
- ✓ Microsoft Office.

IV.2.3 Technologies utilisées

IV.2.3.1 IDE Eclipse

Eclipse est un environnement de développement Java gratuit, open source et extensible. Il est capable d'intégrer des modules (Plugins) de base permettant de gérer l'ensemble de ressources et faciliter le travail du programmeur. **[02]**

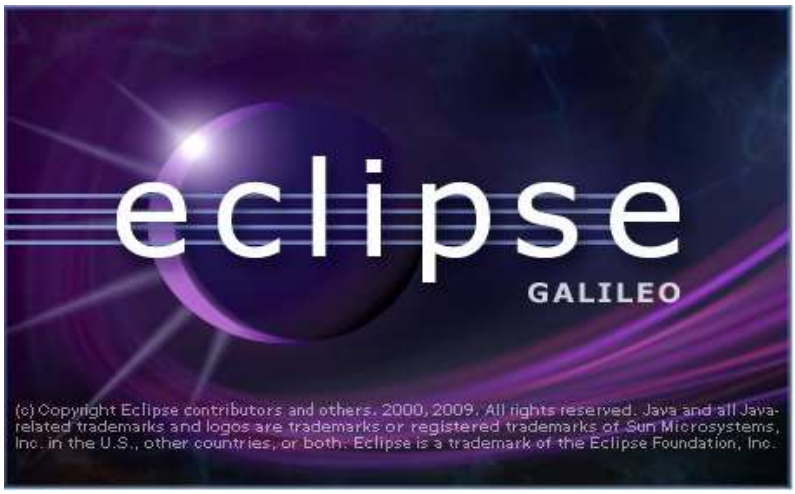

Figure IV. 2 : *IDE Eclipse GALILEO*

IV.2.3.2 Le Plugin ADT (Android Développement Tools)

Pour développer une application Android, nous avons à installer le plugin *Android* qui rajoutera à *Eclipse* les fonctionnalités spécialisées dans le développement sous Android. **[04]**

La *Figure IV.3* représente une capture d'écran montrant l'étape d'intégration du plugin *ADT* dans l'*IDE Eclipse*.

IV.2.3.3 Software Development Kit (SDK)

C'est un kit de développement basé sur le langage Java. Il s'agit des outils que Google a fournis pour interagir avec *Android* pour la réalisation des applications. Le SDK propose de plus, un *émulateur Android*. Ce dernier permet de lancer sur la machine du développeur un terminal virtuel représentant à l'écran un téléphone embarquant *Android*. C'est bien évidemment un outil indispensable pour le développement mobile. A chaque version d'*Android* est associée une version de l'émulateur, permettant au développeur de tester et voir exactement à quoi ressemblera son application sur un matériel réel. **[N3]**

La *Figure IV.4*, montre un émulateur *Android 2.2*.

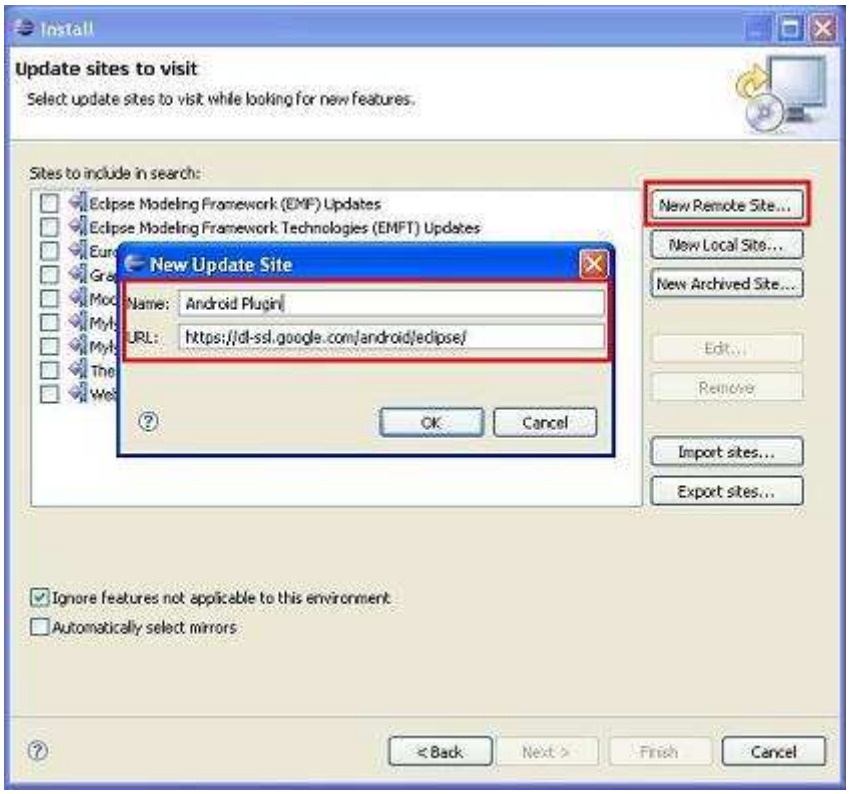

Figure IV. 3 : *Installation de plugin ADT sous Eclipse*

Figure IV. 4 : *Interface du simulateur Android.*

IV.2.3.4 Java DataBase Connectivity

JDBC (*Java DataBase Connectivity*) est une *API* fournie avec Java (depuis sa version 1.1) permettant de se connecter à des bases de données, c'est-à-dire que *JDBC* constitue un ensemble de classes permettant de développer des applications capables de se connecter à des serveurs de bases de données (*SGBD*). **[02]**

L'*API JDBC* a été développée de telle façon à permettre à un programme de se connecter à n'importe quelle base de données en utilisant la même syntaxe, c'est-à-dire que l'*API JDBC* est indépendante du *SGBD*. De plus, *JDBC* bénéficie des avantages de Java, dont la portabilité du code, ce qui lui vaut, en plus d'être indépendant de la base de données, d'être indépendant de la plate-forme sur laquelle elle s'exécute.

Dans un système *Client/serveur*, l'accès aux bases de données avec *JDBC* peut s'effectuer selon un modèle à deux couches ou bien un modèle à trois couches. Pour le premier modèle, une application Java est intimement liée avec une base de données. A cet effet, il faut bien évidemment disposer, pour la base de données concernée, d'un pilote *JDBC* adéquat. Les requêtes *SQL* sont directement envoyées à la base de données, cette dernière renvoyant les résultats par un biais tout aussi direct. La base de données peut être exécutée sur la machine locale (celle sur laquelle l'application Java fonctionne) ou bien sur tout autre ordinateur du réseau (Intranet ou Internet). La figure suivante montre les couches intervenant dans l'utilisation d'un pilote *JDBC* **[02]** :

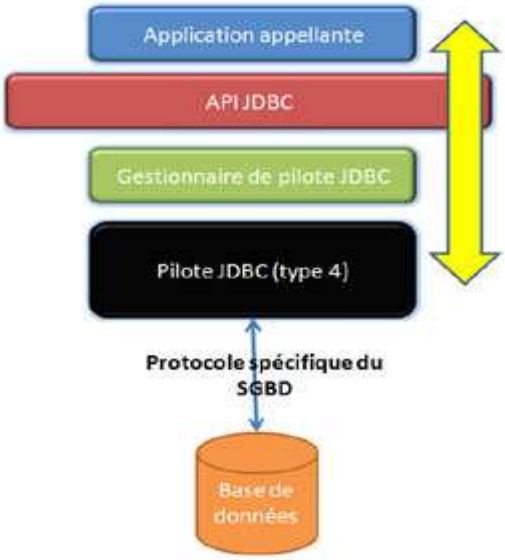

Figure IV. 5 : *Schéma d'un pilote JDBC*

IV.2.3.4 MySQL

MySQL est un système de gestion de base de données relationnel : un langage de requêtes vers les bases de données exploitant le modèle relationnel et utilise le langage SQL comme langage de requête. **[01]**

SQL est un langage de manipulation de bases de données mis au point dans les années 70 par IBM, il permet d'effectuer trois types de manipulations :

1. La manipulation des tables : création, suppression, modification de la structure.
2. La manipulation des données de la base : sélection, modification, suppression d'enregistrement.
3. La gestion des droits d'accès aux tables : Contrôle des données, droit d'accès, validation des modifications.

Figure IV. 6 : *Logo de SGBD MySQL*

IV.3 Les langages de programmation

- ✓ ***JAVA***

Android est un système d'exploitation conçu pour téléphone mobile développé par Google, qui a mis à disposition un kit de développement logiciel (SDK) basé sur le langage Java. **[02]**

Pour justifier ce choix, nous notons qu'au cœur du système *Android* se trouve la machine virtuelle Java, appelée *Dalvik*, qui n'exécute que des fichiers **.dex* qui sont en fait des classes **.java* transformées via le SDK spécialement pour *Dalvik*. En outre, *Dalvik* est liée à un Runtime comportant les principales librairies du Java.

Enfin, nous rappelons que le Java, étant un langage de programmation orienté objet utilisable sur divers systèmes d'exploitation, est un langage assez robuste, portable et à hautes performances.

- ✓ ***XML***

Extensible Markup Language est un langage informatique de balisage générique. Il sert essentiellement à stocker/transférer des données de type texte Unicode structurées en champs arborescents. **[N4]**

En *Android*, grâce à ce langage, nous décrivons les interfaces dans un format spécial, et *Android* le converti automatiquement en objets Java qui seront par la suite disponibles comme tout autre objet dans notre code. Il offre ainsi plus de souplesse de développement, facilite les modifications du code et assure la séparation entre la présentation et le comportement des objets.

IV.4 Le modèle de communication Client / serveur

L'un des objectifs principaux de notre projet est de faire communiquer l'application GUAMB-Personnel-Android (Client) avec le serveur GUAMB-Mobile qui joue le rôle de l'intermédiaire et interagit avec le serveur de base de données GUAMB. Pour cela, nous avons opté pour l'utilisation des sockets qui sont une sorte de point d'ancrage pour les protocoles de transmission de données comme TCP. Les socket sont des objets permettant la gestion de deux flux de données: un flux d'entrée (InputStream), garantissant la réception des données, et un flux de sortie (OutputStream), servant à envoyer les données. En Java, nous distinguerons deux types de socket: les socket simples (dits "clients") et les socket serveurs. Un socket client est tout simplement un socket qui va se connecter sur un socket serveur pour lui demander d'effectuer des tâches.

Le système des sockets est le moyen de communication interprocessus développé pour l'Unix Berkeley (BSD).

Il est actuellement implémenté sur tous les systèmes d'exploitation utilisant TCP/IP. Un socket est le point de communication par lequel un thread peut émettre ou recevoir des informations et ainsi elle permet la communication entre deux applications à travers le réseau.

La communication se fait sur un port particulier de la machine. Le port est une entité logique qui permet d'associer un service particulier à une connexion. Un port est identifié par un entier de 1 à 65535. Par convention les 1024 premiers sont réservés pour des services

standard (80 : *HTTP*, 21 : *FTP*, 25: *SMTP*, ...) Java prend en charge deux protocoles : *TCP* et *UDP*. Les classes et interfaces utiles au développement réseau sont regroupés dans le package *java.net.*

Dans notre cas d'application, on était confronté à l'implémentation d'une application serveur qui représente une interface intermédiaire entre l'utilisateur et le serveur de base de données. Ce serveur qu'on a appelé serveur *GUAMB-Mobile* gère la communication et la répartition des clients de l'application Android et intercepte leurs requêtes grâce à la classe ConfigurationSocket.java sur le serveur qui consiste à exécuter une boucle qui écoute sur un port désigné et récupère le flux de données envoyé par les clients d'une manière non bloquante grâce à un thread qui créé un *Fork* sur l'opération bloquante (accès à distance), par la suite, il fait appel à une méthode de classe SocketReader.java qui s'occupe de traiter la requête reçu afin de généré une requête SQL structuré qui va la communiquer par la suite au serveur de bases de données et ce dernier va lui renvoyer une réponse et renvoie à son tour aux clients.

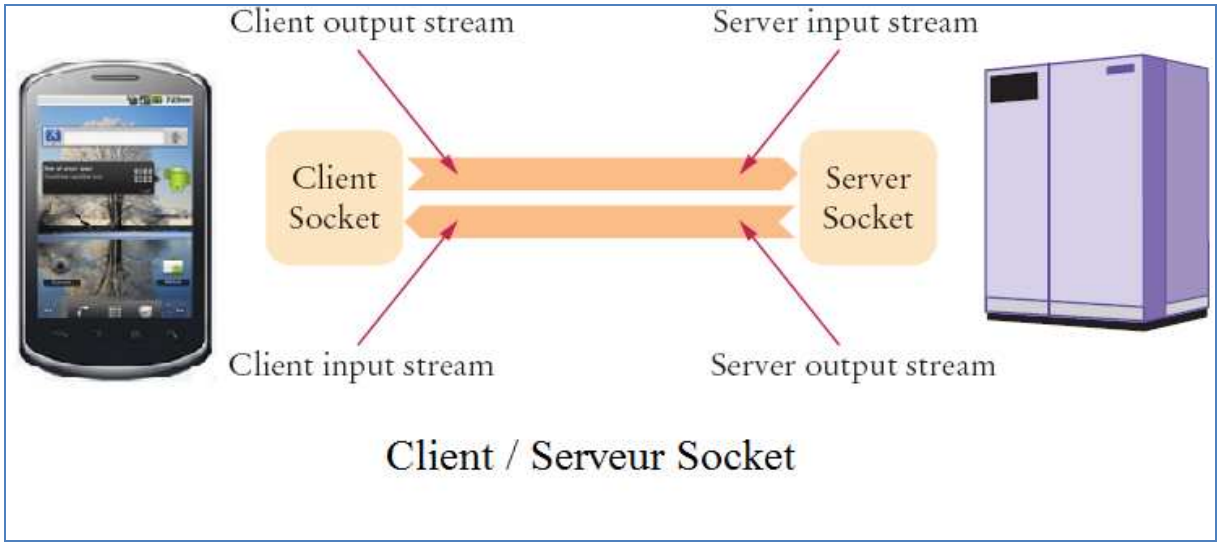

Figure IV. 7 : *Schéma communication Client/serveur avec Socket.*

Dans ce qui précède, on a expliqué brièvement le fonctionnement de notre application et les différentes composantes qui lui permettent de communiquer sur un réseau.

IV.5 L'application serveur de socket GUAMB-Mobile

Pour la gestion des connexions clients (Smartphones Android exécutant l'application GUAMB-PA) à la base de données GUAMB, on a mis en place une application serveur de socket qui joue le rôle d'intermédiaire entre ces clients et le serveur de base de données MySQL.

Elle est chargée de communiquer avec les clients mobiles. Elle récupère les données ou requêtes envoyées par les mobiles ou renvoie aux clients mobiles les données stockées récupérés à partir de la base de données. Elle est aussi chargée d'enregistré les photos des fonctionnaires au travers un formulaire d'ajout ou de mise à jour dans un fichier data de type image à l'intérieur d'un dossier créé manuellement sur l'espace disque de serveur où on a spécifier le chemin d'accès pour ce dernier, en cas de mise à jour de cette photo elle sera automatiquement écrasé dans ce dossier, afin de ne pas avoir des redondances de données inutiles, enfin il lui reste qu'a insérer dans la base de données le nom de fichier image qui est auto généré, composé du nom de fonctionnaire concerné suivi de sa date de naissance, cela pour avoir des noms significatif facile à manipuler .

L'application serveur comme l'application GUAMB-Android-Personnel est développée sous JAVA, elle est composé d'un ensemble de packages structuré en model MVC (Model, Vue, Contrôleurs), chaque package contient ces propre classes nous verrons dans ce qui suit ces packages et leurs classes :

- Le package communication : contient essentiellement les classes qui permet d'établir la communication que ce soit avec les clients ou avec le serveur de base de données.
- Le package contrôleurs : ses classes permettant de gérer des fonctionnalités telles que l'appui sur une touche, et la différente interaction de l'utilisateur sur les vue…etc.
- Le package objets métiers : ses classes sont auto généré par la commande genAll du Shell qui représenté la base de données au niveau de langage de programmation JAVA, de tels sorte que chaque table dans la base de données est incarnée par une classe au niveau du langage JAVA, et qui intègre les opérations usuelles sur la base de données comme l'insertion, la mise à jour et la sélection seront réalisés par des méthodes.
- Le package main : contient la classe principale qui représente le point de démarrage de l'application serveur

- Le package utiles : ses classes sont utilisé pour intégrer des méthodes utilitaire comme la conversion de la date de format anglais au format français …etc.

IV.6 Présentation des interfaces de l'application

IV.6.1 Présentation générale

La figure suivante montre les différentes interfaces de l'application GUAMB-Personnel-Android :

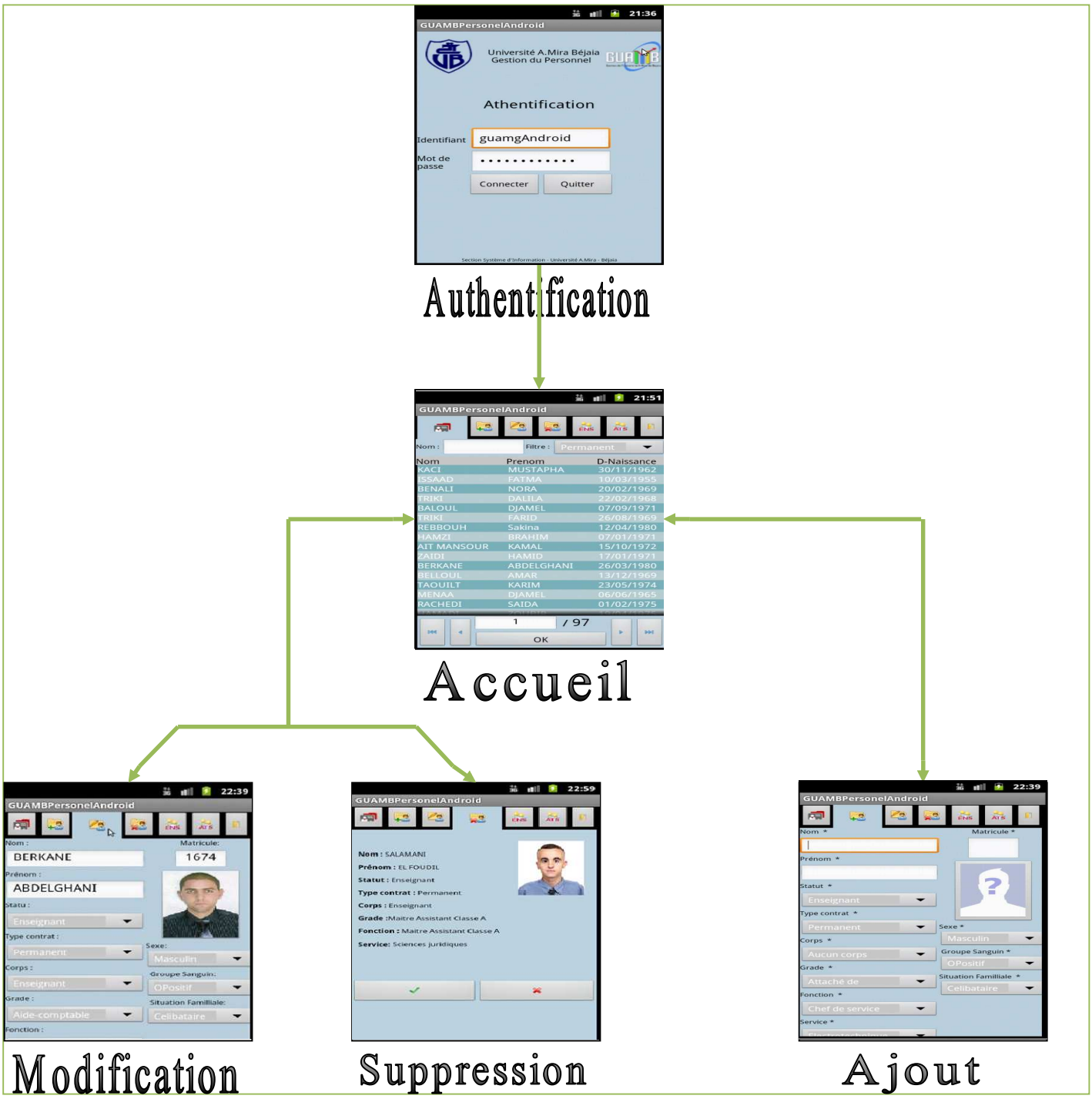

Figure IV. 8 : *Application Android GUAMB – Présentation générale*

IV.6.2 Présentation détaillée

L'application débute par le lancement de la fenêtre d'authentification qui consiste à la sécurisation de notre application des accès non autorisés, une fois l'identifiant et le mot de passe sont vérifier, l'utilisateur peut accéder à toutes les fonctionnalités de l'application .

Figure IV. 9 : *Application Android GUAMB –Authentification*

Une fois connecté, l'utilisateur peut accéder à chaque fonctionnalité de l'application grâce aux différents onglets ci –dessous :

L'onglet « Accueil » Affiche les listes des fonctionnaires avec leurs coordonnées personnelles.

L'onglet permet aussi d'aller directement sur une liste on introduisant sa position ou bien faire la recherche d'un fonctionnaire par son nom.

On pourra aussi filtrer la liste des fonctionnaires par le type de contrat d'emplois, et aussi par leurs statuts (Enseignant, ATS).

Elle assure bien-évidement la navigation entre les différents onglets (ajout, modification, suppression).

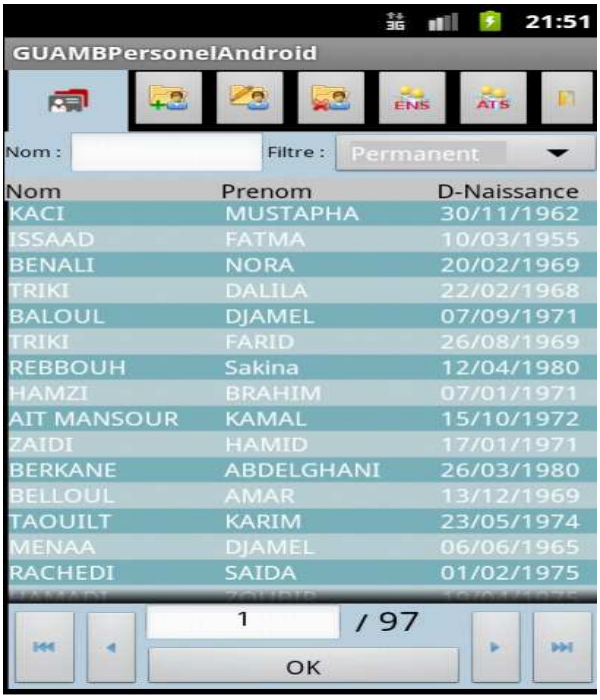

Figure IV. 10 *: Application Android GUAMB - Accueil*

Lorsque l'utilisateur de système voudrait pouvoir ajouter un fonctionnaire un formulaire d'ajout s'affiche à l'écran, une fois le formulaire est rempli, il doit être enregistrés dans la base de données GUAMB.

Figure IV. 11 : *Application Android GUAMB-Ajout*

L'onglet « Modification » permit de mettre à jour un fonctionnaire sélectionné au préalable dans la liste des fonctionnaires à l'onglet « accueil ».

Figure IV. 12 : *Application Android GUAMB Modification*

L'onglet « Suppression » permet de supprimer un fonctionnaire de la base de données.

Figure IV. 13. *Application Android GUAMB –Suppression*

IV.7 Conclusion

Dans ce chapitre de réalisation, nous avons présenté les plates-formes matérielles et logicielles sur lesquelles nous avons développé notre projet, ainsi que les technologies employées et le modèle de communication client/serveur.

Nous avons, par la suite, présenté les interfaces les plus significatives de notre application.

Conclusion Générale et Perspectives

Conclusion générale et perspectives

La mobilité est un élément très important dans les systèmes d'informations, elle permet de réaliser la saisie des données, des consultations et divers tâche sans être obligé d'être physiquement à un endroit donné.

Les systèmes Android, le concurrent principal de *IPhone* de l'entreprise *Apple Inc*, ouvre les portes à la communauté de développement, vu qu'il représente un logiciel libre et open source, à fin construire des applications mobiles dans de multiples domaines.

Par conséquent, et à partir des indications et recommandations de la section Système d'information de l'université de Bejaia, qui consistent à favoriser les logiciels et solutions libres et/ou open source (utilisation de *Linux*, *Java*, *MySQL*, *Apache*, etc.), nous avons opté pour les système Android à fin de développer une application qui contribue à la facilité de la gestion du personnel de l'université. Cette application mobile n'est autonome de l'application en cours de développement dans le sens où nous exploitons la même base de données.

Après analyse et conception, nous avons utilisé pour le développement de l'application le langage Java, vu ses avantages dans la portabilité et la mise sous-réseau. Pour la connexion d'une application Android avec la base de données *GUAMB*, nous avons utilisé un serveur de socket comme un pont pour y accéder.

Notre travail, comme tout travail humain, possède des lacunes et d'insuffisances, pour cela, nous avons dressé quelques perspectives :

- ✓ Ajouter les fonctionnalités manquantes de gestion du personnel ;
- ✓ Optimiser les communications entre l'application et le serveur ;
- ✓ Elargir le champ de cette application pour gérer les autres services de l'université, comme la gestion de Stock, la scolarité, *etc.*

Bibliographie

Bibliographie

Ouvrages

[01] Christian Soutou, « Apprendre SQL avec MySQL », ÉDITIONS EYROLLES, 02/2011.

[02] Jean-Michel, « Développons en Java », EDITION DOUDOUX 05/03/2012.

[03] Lucas Jordan and Pieter Gryling, « Practical Android Projects », Books for Professionals by Professionals, EDITION Apress 22 février, 2011.

[04] Mark Murphy, « Beginning Android *2* », Edition APRESS, New York 2010.

[05] Pascal Roques, « UML 2 par la pratique »,5 éme édition EYROLLES, sept 2006.

[06] Reto Meier, « Professional Android Application Development», Wiley Publishing 2009.

[07] Sayed Y. Hashimi and Satya Komatineni, « Apress.Pro.Android », Jun.2009

Sites Internet

[N1] http://aurelien-vannieuwenhuyze.com/?p=45, date consultation 12/05/2012.

[N2] http://www.lafabrick.com/blog/2009/01/13/786-reflex-1-une-micro-architecture-pour- flex- se/, date consultation 15/05/2012.

[N3] http://developer.android.com/sdk/eclipse-adt.html, date consultation 25/05/2012.

[N4] http://www.ibm.com/developerworks/opensource/library/x-android/, date consultation 05/06/2012.

Résumé

Aujourd'hui, l'informatique mobile a atteint une prodigieuse évolution technologique dans différents domaines (réseaux informatiques, bases de données, le Web, etc.). Cette évolution est nécessaire pour remédier aux problèmes rencontrés dans la vie actuelle.

Notre travail, qui s'inscrit dans ce cadre, est de concevoir et de réaliser une application pour le système Android permettant la gestion du personnel de l'université de Bejaia. Cette gestion a été définie auparavant par la section système d'information, qui prend en charge le développement du système GUAMB, et notre effort sera focalisé sur l'extension de cette gestion sur le terminal mobile embarquant le système d'exploitation *Android*. Cette solution facilitera et augmentera les possibilités de travail pour le personnel de l'université.

Mots-clés :

MySQL, SGBD, UML, UP, Java, Linux, Web, Android, Gestion Personnel.

Abstract

Nowadays, mobile computing has reached a prodigious technological development in different fields (computer networks, databases, Web...). This change is necessary to remedy the problems encountered in the present life.

Our work is to design and implement an application for the *Android* system for personnel management at the University of Bejaia. This management has been previously defined by the system information section, and our effort will be focused on the extension of this management on the mobile embed the Android operating system. This solution will facilitate and increase work opportunities for university staff.

Keywords:

MySQL, SGBC, UML, UP, Java, Linux, Web, Android, Personal Management.