République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté desSciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Licence en Informatique

Thème

Application pour système Android

Transfert du répertoire téléphonique vers un serveur.

Réalisé par :

- **BENSAHLA TALET Wafâe.**

Présenté le 10 Juin 2013 devant la commission d'examination composée de MM.

- *Mr. Smahi* *(Examinateur)*
- *Mr. Merzoug* *(Examinateur)*
- *Mr. Benazzouz* *(Encadreur)*

Année universitaire :2013-2014

Remerciements

Avant tout, je remercie le bon DIEU de m'avoir aidé à accomplir ce modeste travail.

Je voudrais témoigner ma reconnaissance sincère à mon encadreur Mr. BENAZZOUZ pour ses conseils et ses encouragements tout au long de ce projet, ainsi qu'à Mr. BENNAMAR Abdelkrim, Chef de département d'informatique.

Je remercie les examinateurs pour avoir accepté d'examiner ce travail et pour leurs participations au jury.

Je tiens tout particulièrement à exprimer ma gratitude à Mme. EL YEBDRI Zeyneb, HADJOUI Meriem, EL GHAZI Walid et MESBAHI Fatima Zohra pour m'avoir accordé leur temps.

Enfin, je ne saurais terminer ces remerciements sans y associer toute personne qui, de près ou de loin, m'a apporté son aide ou sa sympathie.

Dédicaces

Je dédie ce modeste travail :

A mes chers parents qui ont contribué à ma réussite et m'ont encouragé, en reconnaissance de leur présence et de leurs sacrifices.

A mes sœurs.

A ma famille.

A tous ceux qui me sont chers.

Wafae.

Introduction générale

Aujourd'hui, les Smartphones ont envahi nos vies. Ils offrent des applications variées qui nous permettent de nous divertir et nous simplifient la vie. Par ailleurs, sa capacité de plus en plus grande permet de stocker musique, photos, vidéos, contacts...

Les téléphones sont beaucoup plus personnalisables et personnels.

Ainsi, dans ce projet, nous sommes intéressés à une application mobile permettant de transférer ses données téléphoniques vers un serveur afin de pouvoir les restituer en cas de changement de téléphone ou de perte. L'application permettrait ainsi d'exporter ses contacts, puis de les importer du serveur vers n'importe quel téléphone doté d'un système d'exploitation Android sur lequel l'application est installée. Une interface comprenant un identifiant et un mot de passe permettrait l'authentification de l'utilisateur afin de protéger les données d'un accès inopportun et de pouvoir les restituer à leur propriétaire respectif. Il serait également possible de transférer la liste des contacts directement d'un téléphone vers un autre.

Ce mémoire se divise en trois chapitres :

- Après cette introduction, le premier chapitre présente le système d'exploitation Android.
- Le deuxième chapitre présente la conception du système basé sur le processus de développement UP.
- Dans le troisième chapitre, nous nous intéressons à l'implémentation de l'application.

Chapitre I: Android

I.1. Android OS

I.1.1. Définition

Android est un système d'exploitation (SE) open source basé sur Unix dédié aux appareils mobiles, Smartphones et tablettes [1].

Apparu sur le marché en 2007, Android est l'un des principaux systèmes d'exploitation pour mobile sur le marché avec l'iOS d'Apple.

Alors que la barre du milliard de Smartphones vendus a été franchie l'an passé, Android aurait capté 79% du marché au niveau mondial, selon StrategyAnalytics[1].

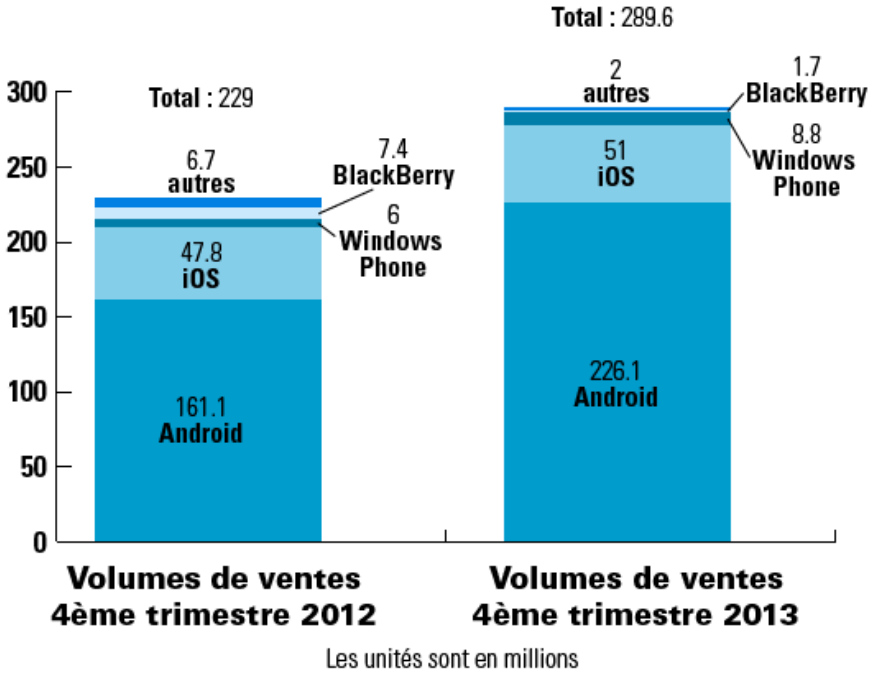

Figure I.1: Volume des ventes des systèmes d'exploitation présents sur le marché [2]

[1]StrategyAnalyticsinc: société qui fournit des services de recherche et de consulting aux Etats Unis et à l'international.

Le système a en effet connu une croissance exponentielle en quelques années. Les atouts ayant fait le succès du système sont principalement le fait qu'il est un système libre, évolutif et son développement accessible.

I.1.2. Sous Licence Open Source

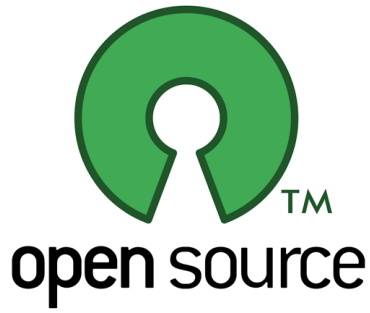

Figure I.2: Logo Open Source [3]

L'appellation open source désigne les logiciels qui respectent les critères de libre redistribution et d'accès au code source. La communauté de développement peut ainsi étudier le code source pour localiser d'éventuels problèmes de sécurité et les corriger.

Le système d'exploitation de Google est effectivement sous licence Apache, ce qui permet au code, en plus d'être consultable, modifiable et réutilisable, d'être intégré dans des applications propriétaires.

Basé sur le noyau Unix**, il est associé à un large ensemble de bibliothèques open-source également.** Nous pouvons citer OpenGL pour la gestion des images 2D et environnements 3D, SQL Lite pour la gestion des bases de données embarquées ou encore WebKit, est employé pour le navigateur internet.

I.1.3. Historique des versions

Depuis sa création, Android n'a cessé de s'enrichir de nouvelles fonctionnalités.

Version	Date de sortie	Nom de code	Fonctionnalités principales ajoutées
1.0 et 1.1	Fin 2007	Apple Pie	Barre de notification et apparition des wigets. Création de l'Android Market.
1.5	Avril 2009	Cupcake	Premier clavier tactile.
1.6	Septembre 2009	Donut	Gestion des résolutions d'écran allant jusqu'à la HD.
2.0	Octobre 2009	Eclair	Prise en charge du Bluetooth. Arrivée de Google Maps.
2.2.x	Mai 2010	Froyo	Ecran de verrouillage à mot de passe
2.3.x	Décembre 2010	Gingerbread	Outils fournis par Google pour permettre le développement d'applications Android en natif.
3.x.x	Février 2011	Honeycomb	Mise-à-jour réservée aux tablettes.
4.0.x	Octobre 2011	IceCream Sandwich	Capture d'écran. Prise en charge de l'enregistrement full HD.
4.x	Juillet 2012	Jelly Bean	Système multi-utilisateurs pour tablette. Bluetooth Smart. Restriction au niveau des comptes utilisateurs.
4.4.x	Novembre 2013	KitKat	Ecran tactile amélioré. Utilisation de la mémoire optimisée.

Table I.1: Les différentes versions d'Android.

I.2. Programmer pour Android

I.2.1. Contraintes du développement mobile

Le développement sur Smartphone doit prendre en compte les spécificités de ce type de terminaux comme la gestion des ressources matérielles et réseaux limitées et la fragmentation importante des systèmes.

Effectivement, contrairement à un ordinateur de bureau, les capacités d'un téléphone portable sont des paramètres importants à considérer lors du processus de programmation:

- Les Smartphones étant généralement connectés à internet via les réseaux Wi-fi ou la 3G, l'accès au réseau de données peut être interrompu. Le mode «déconnecté» doit alors être géré au niveau de l'application.

- Le développeur doit rester vigilent sur les ressources mémoires et processeurs nécessaires au fonctionnement de l'application. De manière générale, l'architecture logicielle doit être économe, les algorithmes optimisés.

- Pour une même plateforme, les caractéristiques techniques des mobiles sont très variables en termes de processeurs, de mémoire, de taille d'écran. Il faut être sensible à ces points afin de garantir la compatibilité, l'accessibilité et la visibilité de son application au plus grand nombre.

Par ailleurs, Android implique encore une autre précaution à prendre qui lui est propre. Près de huit versions de son système ont été publiées en l'espace de deux ans et demi, et les anciennes versions restent très présentes. Il faut donc penser à concevoir une application compatible avec le maximum de systèmes.

Version	Codename	API	Distribution
2.2	Froyo	8	1.1%
2.3.3 - 2.3.7	Gingerbread	10	17.8%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	14.3%
4.1.x	Jelly Bean	16	34.4%
4.2.x		17	18.1%
4.3		18	8.9%
4.4	KitKat	19	5.3%

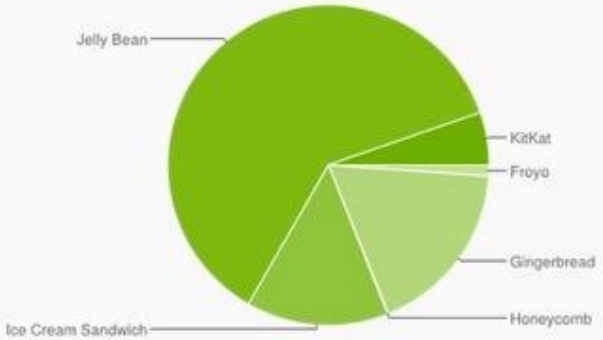

Figure I.3: Répartition des différentes versions d'Android [4].

I.2.2. Outils

Depuis 2010, Google met à la disposition des utilisateurs les outils nécessaires au développement d'applications destiné à Android. Le constructeur recommande d'utiliser l'environnement de développement intégré (EDI) Eclipse[2] et les plugins *Android Developer Tools* (ADT) et *Software Development Kit* (SDK). Le SDK et l'AVD contiennent des fonctionnalités propres à Android ainsi que l'*Android Virtual Device*(AVD), un émulateur qui va permettre de tester notre application au fur et à mesure du développement.

Une documentation complète est mise à disposition sur internet et encourage les particuliers à enrichir le système d'applications innovantes.

En effet, initialement proposées pour la productivité et la récupération de données, les applications voient rapidement leur domaine d'application s'élargir. Les applications proposent désormais des services de géolocalisation, d'achat, des jeux…

[2] Un EDI dédié entièrement à Android, Android Studio, est également en cours de développement.

I.2.3. Architecture d'une application

Lors de l'ouverture d'une nouvelle application via Eclipse, l'EDI construit l'arborescence suivante:

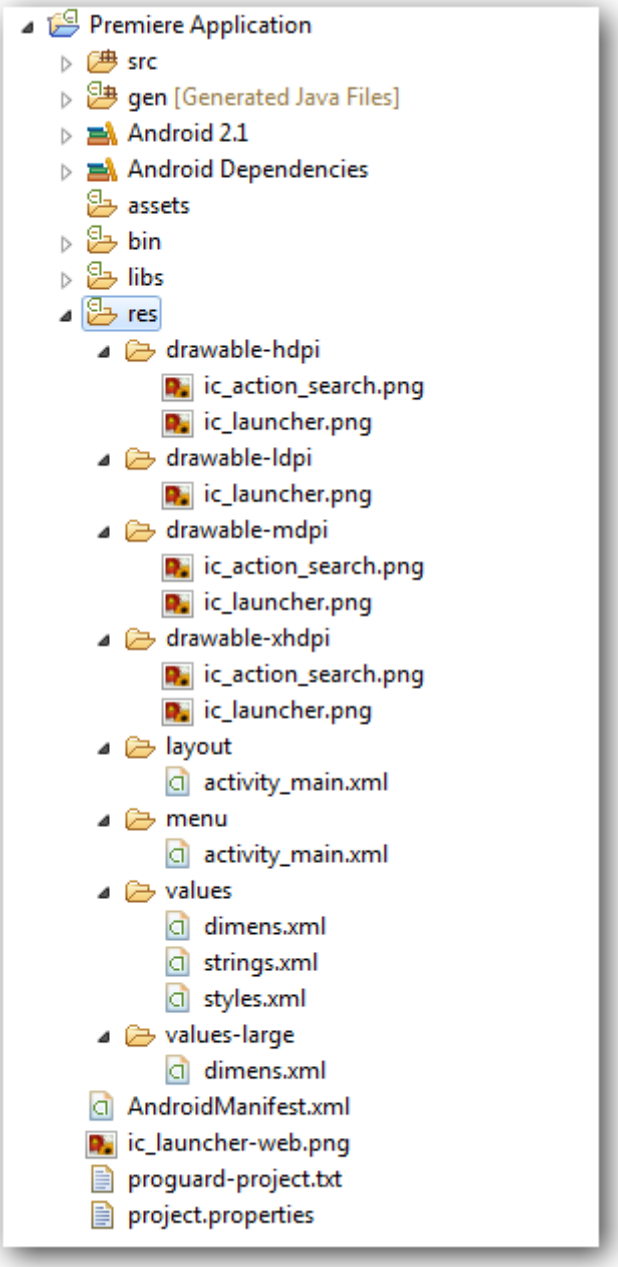

Figure I.4: Composition d'une application [5].

- Le dossier «src» est le dossier contenant le code source Java de l'application.
- Le dossier «gen» les fichiers générés automatiquement lors de la compilation.

- Le dossier «res» celui qui contient les ressources de l'application. Ces ressources sont de type différent: le type «drawable» contient les images, «layout» la description de l'interface graphique en format XML, «menu» les fichiers définissant le menu de l'application et «value» contient les valeurs et constantes pouvant être utilisées dans l'application. Par exemple, le fichier «string.xml» contient les chaînes de caractères.

Il est possible de créer plusieurs dossiers ressources par type de ressource, un pour chaque configuration. Le dossier «drawable» existe en quatre fois pour quatre résolutions d'écran différentes. Par exemple «drawable_ldpi» pour une résolution moyenne et «drawable_xdpi» pour une très grande résolution.

- Le fichier « AndroidManifest.xml » est chargé de décrire l'application au système Android. On y trouve le nom du package Java, les versions minimum et maximum du SE avec lesquelles l'application est compatible, les déclarations des permissions _pour autoriser l'application à accéder à Internet par exemple, ou encore les différentes composantes de l'application qui vont être utilisées.

1.2.4. Composantes d'une application

Les applications Android disposent de six composantes:

- *Activities:* une activité est un composant fournissant une interface avec laquelle l'utilisateur peut interagir dans le but de lancer une action comme envoyer un SMS ou passer un appel. Une application est généralement composée de plusieurs activités (une par interface graphique) liées entre elles. Chaque activité peut alors en lancer une nouvelle, ce qui définit un cycle de vie.

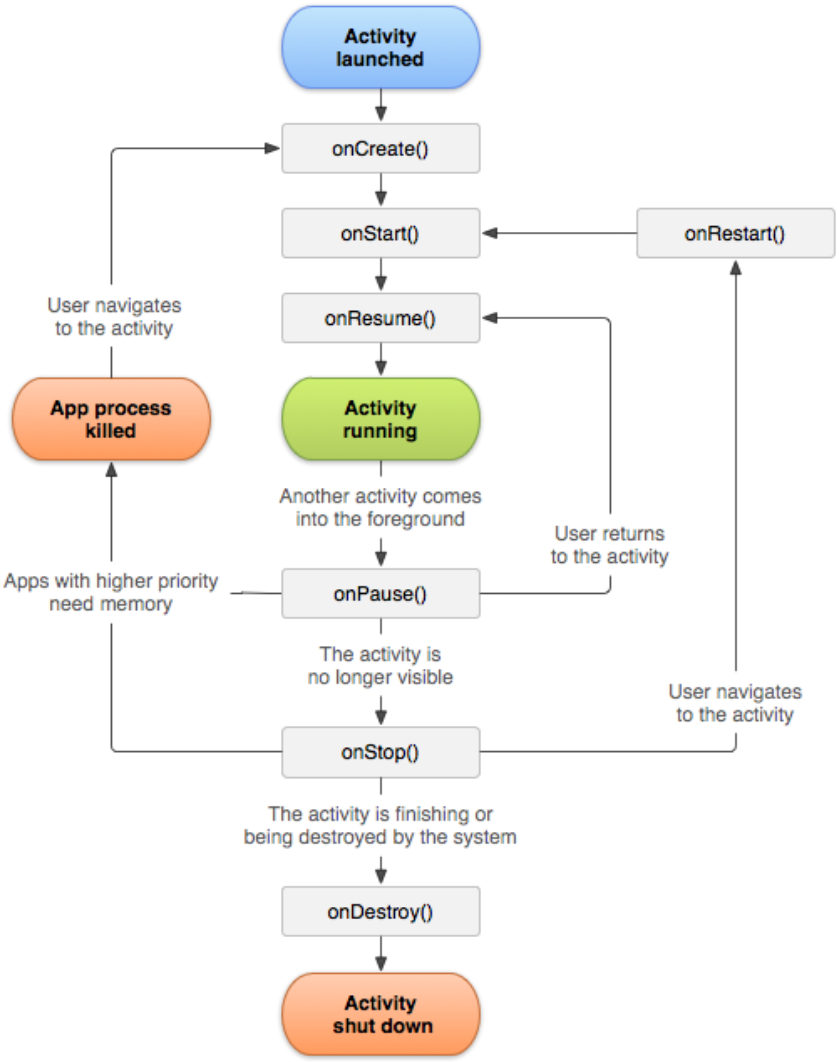

Figure I.5: Cycle de vie d'une activité [6].

- *Services*: un service est un composant permettant d'exécuter des actions en arrière-plan. Contrairement à l'activité, il ne fournit pas d'interface. Un service continue d'opérer même si l'utilisateur passe à une autre application.

- *Content providers*: ils servent à accéder à une structure de données organisée, encapsulent ces données et fournissent les mécanismes de sécurité nécessaires à leur manipulations.

- *App widgets*: un widget est une application miniature pouvant être intégrée dans d'autres applications telle que la page d'accueil. Par exemple, un lecteur audio.

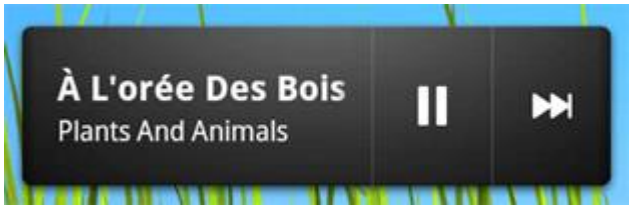

Figure I.6: App widget [6].

o *Processes andThreads*: par défaut, tous les composants d'une même l'application s'exécutent dans un même processus et thread dit thread principal. Cependant, l'application peut avoir besoin d'exécuter des traitements coûteux en temps en arrière-plan pour ne pas bloquer l'application. Il est ainsi possible d'exécuter ces traitements dans un autre thread.

o *Intents and intentfilters*: un intent est une composante pouvant être utilisée pour envoyer des messages aux autres composantes de l'application afin de lancer des actions. Il peut être utilisé dans trois cas fondamentaux:

- Lancer une activité: il est possible de lancer une nouvelle activité en envoyant un intent. Cet intent décrit l'activité à commencer et contient les données nécessaires.
- Lancer un service: ce service servira à exécuter une action ponctuelle comme télécharger un fichier.
- Envoyer un *Broadcast*: un broadcast est un message pouvant être envoyé à d'autres applications dans le but de communiquer avec celles-ci.

Pour conclure, Android est un système d'exploitation pour mobile open source. Son propriétaire, Google, met l'ensemble des outils nécessaires au développement d'applications pour Android à disposition des utilisateurs. Cependant, le développement pour téléphones portables et tablettes présente des contraintes de programmation particulières liées à la limitation des ressources qu'il faut garder en tête tout au long du processus de développement.

Chapitre II: Conception UML et UP

Pour l'analyse et la conception de cette application, nous nous somme basé sur la méthode UP, qui elle-même utilise le langage de modélisation UML.

I.1. UML

II.1.1. Introduction

UML, traduit par langage de modélisation unifié, est un modèle de modélisation orienté objet.

Face au nouveau mode de programmation Orienté Objet (POO), les méthodes de modélisation classiques séquentielles montrent rapidement certaines limites. Dans ce contexte, de nouveaux langages de modélisation voient le jour tels Booch, OMT... Devant ce foisonnement, l'*Object Management Group*[3] (OMG) se fixe comme objectif de définir une notation standard. C'est ainsi que naît UML de la fusion des méthodes objets dominantes.

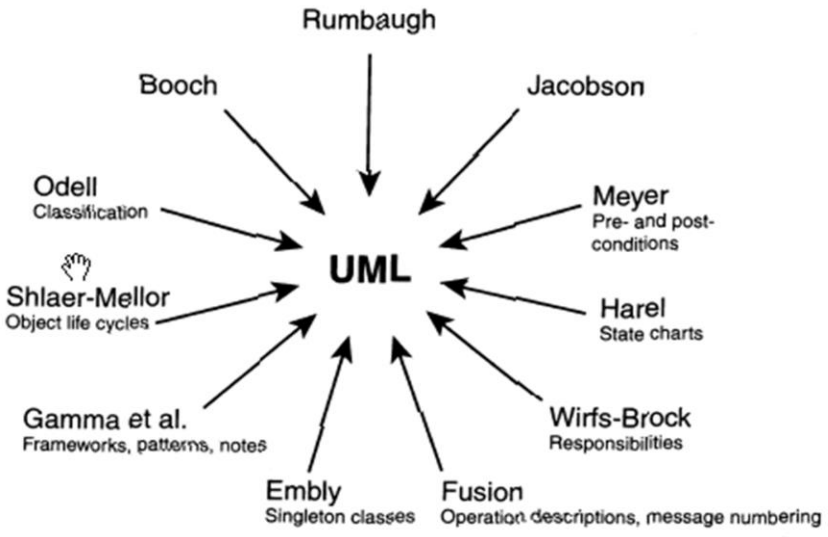

Figure II.1 : Les langages de modélisation.

.

UML s'impose rapidement comme un standard international.

[3] L'OMG est un organisme à but non lucratif créé en 1989 à l'initiative de grandes multinationales (HP, Sun...) dont le but est de promouvoir des standards qui garantissent l'interopérabilité entre applications orientées objet.

En effet, UML sert à faciliter la compréhension des problèmes et des besoins et la communication entre les personnes (utilisateur, analyste, concepteur…).

C'est un support de raisonnement qui vise à améliorer la lisibilité des schémas de conception, prépare la documentation et les programmes et facilite la maintenance.

Il permet ainsi de définir :

- Les éléments de modélisation (les concepts manipulés par le langage) ;
- La sémantique de ces éléments (leurs définitions et leur sens d'utilisation) ;
- Une notation qui permet de représenter graphiquement les éléments de modélisation. Cette notation est le support d'utilisation d'UML.

UML permet ainsi d'exprimer des modèles objet en faisant abstraction de leur implémentation, c'est-à-dire que le modèle fourni est indépendant du langage de programmation. C'est un langage formel possédant les caractéristiques suivantes :

- Un langage sans ambiguïtés ;
- Un langage universel pouvant servir de support pour tout langage orienté objet ;
- Un moyen de définir la structure d'un programme ;
- Une représentation visuelle permettant la communication entre acteurs d'un même projet.

II.1.2. Modéliser avec UML

Un processus de modélisation consiste à identifier les caractéristiques intéressantes d'une entité en vue d'une utilisation précise.

Le caractère abstrait d'un modèle doit notamment permettre de faciliter la compréhension du système étudié et réduire la complexité. Il doit permettre de simuler le système : un modèle représente le système étudié et reproduit ses comportements. Le modèle est une décomposition de la réalité, le but étant de disposer d'éléments de travail exploitables par des moyens mathématiques et informatiques.

Bien qu'UML permette de représenter des modèles, il ne définit pas de processus d'élaboration de ces modèles. Cependant, dans le cadre de la modélisation informatique, les auteurs d'UML préconisent d'utiliser :

- *Une démarche itérative et incrémentale* :

  Pour modéliser un système complexe, il vaut mieux s'y prendre en plusieurs fois, en affinant son analyse par étapes. Cette étape devrait aussi s'appliquer au cycle de développement dans son ensemble, en favorisant le prototypage. Le but est de mieux maitriser la part d'inconnu et d'incertitudes qui caractérisent les systèmes complexes.

- *Une démarche pilotée par les besoins utilisateurs :*

  Ce sont les utilisateurs qui guident la définition des modèles (les utilisateurs définissent ce que doit être le système). Le but est de répondre aux besoins des clients. A chaque itération de la phase d'analyse, on clarifie, affine et valide les besoins des utilisateurs. A chaque étape de la phase de conception et de réalisation, on veille à la prise en compte des besoins utilisateurs. A chaque itération de la phase de test, on vérifie que les besoins des utilisateurs sont satisfaits.

- *Une démarche centrée sur l'architecture* :

  Une architecture adaptée est la clé du succès d'un développement. Elle décrit des choix stratégiques qui déterminent en grandes parti les qualités du logiciel (adaptabilité, performance, fiabilité…). Philippe Kruchten[4] propose différentes perspectives, indépendantes et complémentaires, qui permettent de définir un modèle d'architecture. Cette vue (dite 4+1) a fortement inspiré UML.

  - La vue logique : elle identifie les éléments du domaine ainsi que les relations et les interactions entre ces éléments.

  - La vue des composantes : cette vue montre l'allocation des éléments de modélisation dans des modules (fichiers sources, bibliothèques…).

[4] Né en 1952, Ph. Kruchten est un professeur en Génie Logiciel à l'université British Columbia au Canada. Il est l'auteur du modèle de vue « 4+1 ».

- La vue des processus : dans les environnements multitâches, elle montre la décomposition du système en termes de processus, ainsi que les interactions entre ces processus.

- La vue de déploiement : dans les environnements distribués, cette vue décrit les ressources matérielles et la répartition du logiciel entre ces ressources.

- La vue des cas d'utilisation : elle définit les besoins des clients du système et centre la définition de l'architecture du système sur la satisfaction de ces besoins.

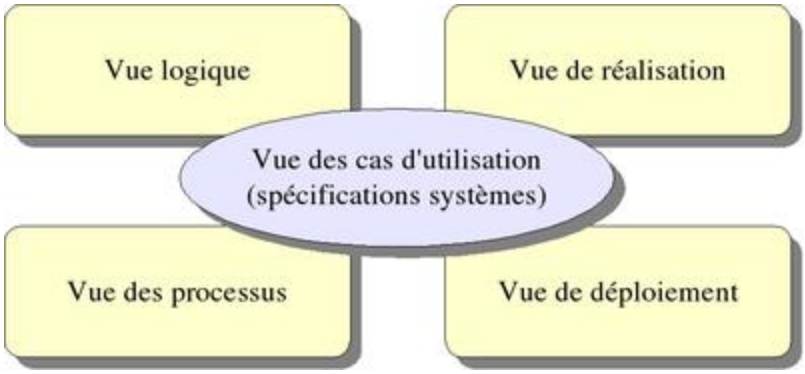

Figure II.2 : Vue 4+1 [1].

II.1.3 : Les diagrammes UML

Un diagramme UML est une représentation graphique qui s'intéresse à un aspect précis du modèle. Chaque type de diagramme possède une structure et véhicule une sémantique précise.

Nous distinguons deux types de diagrammes : les diagrammes statiques et les diagrammes dynamiques.

- Les diagrammes statiques :
  - *Diagramme de cas d'utilisation* : permet de modéliser les acteurs principaux du système et met en évidence les cas d'utilisation typiques.
  - *Diagramme d'objets* : modélise des objets et les interactions entre ces objets.
  - *Diagramme de classes* : exprime de manière générale la structure statique d'un système en termes de classes et de relations entre ces classes.
  - *Diagramme de composants* : étudie l'implantation du système en mémoire.

- *Diagramme de déploiement* : complémentaire du diagramme de composants, il décrit la répartition physique des instances de composants, de processus et d'objets.

- Les diagrammes dynamiques :
  - *Diagramme de collaboration* : permet de mettre en évidence les interactions entre les différents objets du système étudié, ainsi que les messages qu'ils échangent entre eux.
  - *Diagramme de séquence* : permet de représenter des collaborations entre objets selon un point de vue temporel, on y met l'accent sur la chronologie des envois des messages.
  - *Diagramme d'activité* : montre le flux d'activité associé à un scénario de cas d'utilisation.
  - *Diagramme d'état transition* : décrit les changements d'état d'un objet en réponse aux interactions avec d'autres objets ou composantes.

II.2. Processus Unifié

II.2.1. Introduction

Dans une démarche traditionnelle, le processus de développement était caractérisé par un processus de type séquentiel :

- Le développement était composé de plusieurs phases elles-mêmes divisées en plusieurs étapes.

- La fin d'une phase correspond à la conclusion de ses étapes.

Dans une approche objet :

- Le développement est itératif.

- Les activités se déroulent dans plusieurs dimensions.

La maîtrise des processus de développement implique une organisation et un suivi des activités, c'est ce à quoi s'attachent les différentes méthodes qui s'appuient sur l'utilisation du langage UML pour modéliser un système.

UP est donc une méthode générique de développement de logiciel qui fournit un processus de développement regroupant les activités à mener pour transformer les besoins d'un utilisateur en système logiciel.

II.2.2. Caractéristiques du Processus Unifié

Le processus unifié utilise l'ensemble des outils et des diagrammes fournis par UML.

- *UP est piloté par les cas d'utilisation.*

  L'objectif principal d'un système logiciel est de rendre service à ses utilisateurs, il faut par conséquent bien comprendre leurs désirs et leurs besoins. Le processus de développement sera donc centré sur l'utilisateur.

  L'utilisateur représente une personne ou bien une chose dialoguant avec le système en cours de développement. Ce type d'interaction est appelé cas d'utilisation.

- *UP est centré sur l'architecture.*

  L'architecture d'un système logiciel peut être décrite comme les différentes vues du système qui doit être construit. L'architecture logicielle équivaut aux aspects statiques et dynamiques les plus significatifs du système. Elle émerge des besoins de l'entreprise, tels qu'ils sont exprimés par les utilisateurs et autres intervenants et tels qu'ils sont reflétés par les cas d'utilisation.

  Elle subit également l'influence d'autres facteurs tels que la plateforme sur laquelle devra s'exécuter le système ou encore les briques de base réutilisables disponibles pour le développement.

  L'architecture doit prévoir la réalisation de tous les cas d'utilisation.

- *UP est itératif et incrémental*.

  Un processus itératif et incrémental présente de nombreux avantages :

- Limiter les coûts, en terme de risque, aux strictes dépenses liées à une itération ;
- Limiter les risques de retard de mise sur le marché du produit développé _l'identification des problèmes se faisant dès les premiers stades de développement ;
- Accélérer le rythme de développement en fixant des objectifs clairs et à court terme ;
- Prendre en compte le fait que les besoins des utilisateurs et les exigences correspondantes ne peuvent être intégralement définies à l'avance et se dégagent peu à peu des itérations successives.

Ainsi, l'architecture fournit la structure qui servira de cadre au travail effectué tandis que les cas d'utilisation définissent les objectifs et orientent le travail de chaque itération.

II.2.3. Cycle de vie du processus unifié

UP gère le processus de développement par deux axes :

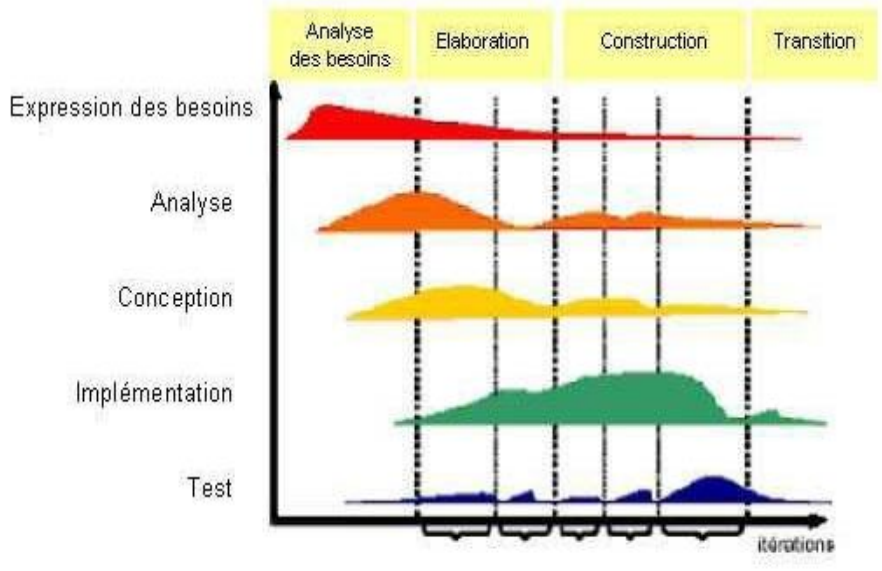

Figure 1I.2 : Cycle de vie et activités [7].

L'axe vertical représente les principaux enchainements d'activités, qui regroupent les activités selon leur nature.

Modèle des cas d'utilisation	Expose les cas d'utilisation et leurs relations avec les utilisateurs.
Modèle d'analyse	Détaille les cas d'utilisation et procède à une première répartition du comportement du système entre divers objet.
Modèle de conception	Définit la structure statique du système sous forme de sous-système, classes et interfaces.
Modèle d'implémentation	Intègre les composants et la correspondance entre les classes et les composants.
Modèle de déploiement	Définit les nœuds physiques des ordinateurs et l'affectation des composants sur ces nœuds.
Modèle de test	Décrit les cas de test vérifiant les cas d'utilisation.
Représentation de l'architecture	Description de l'architecture.

Table II.1 : Définition des différentes activités du processus UP.

L'axe horizontal représente le temps et montre le déroulement du cycle de vie du processus.

Phase	***Description***
Expression des besoins	_Inventorier les besoins principaux. _Recenser les besoins fonctionnels (utilisateurs). _Appréhender les besoins non fonctionnels (aspect technique).
Analyse	_Préciser les cas d'utilisation _Construction d'une architecture de référence. _Estimer les ressources nécessaires à l'achèvement du projet.
Construction	_Construction du projet. _Implémentation du système sous forme de composants (codes sources, scripts…).
Transition	_Obtention du produit en version bêta. _Un groupe d'utilisateurs essaye le produit et détecte les anomalies.

Table II.2 : Description des phases du cycle de vie du processus unifié.

II.3. Application au projet

Dans le but de modéliser notre application, nous avons utilisé trois diagrammes UML : le diagramme de cas d'utilisation, le diagramme de séquence et le diagramme de classes.

II.3.1. Cas d'utilisation

Commençons par identifier les acteurs ainsi que les activités principales de notre application. L'utilisateur peut être toute personne propriétaire d'un téléphone doté du système d'exploitation Android. L'application propose trois activités : exporter ses données vers le serveur, récupérer les données du serveur sur le téléphone, transférer les données vers un autre téléphone.

Nous obtenons alors le diagramme de cas d'utilisation suivant :

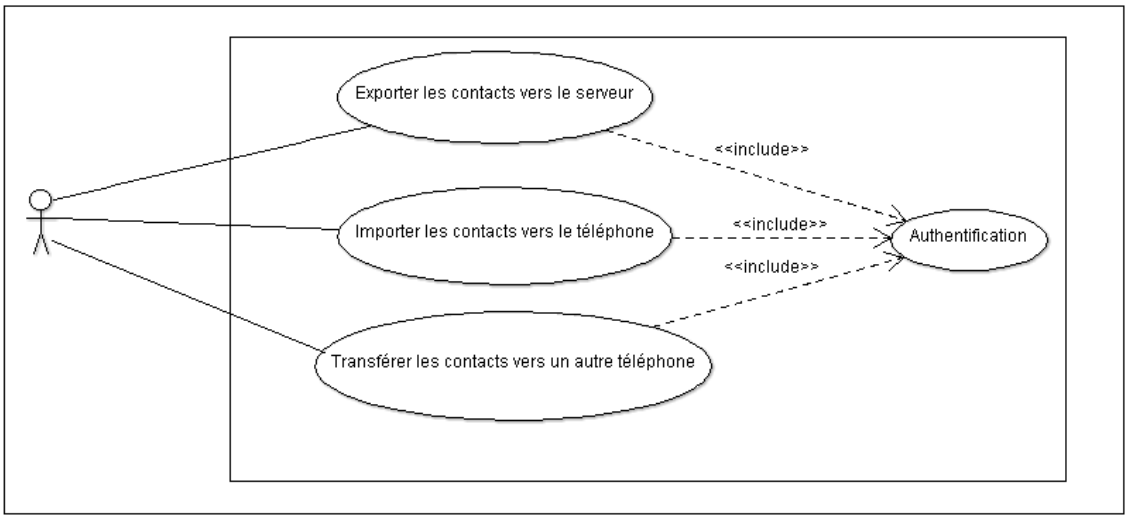

Figure II.3 : Diagramme de cas d'utilisation.

II.3.2. Diagramme de séquence

Détaillons les cas d'utilisation.

L'utilisateur interagit avec deux entités principales : le serveur ou bien un autre téléphone.

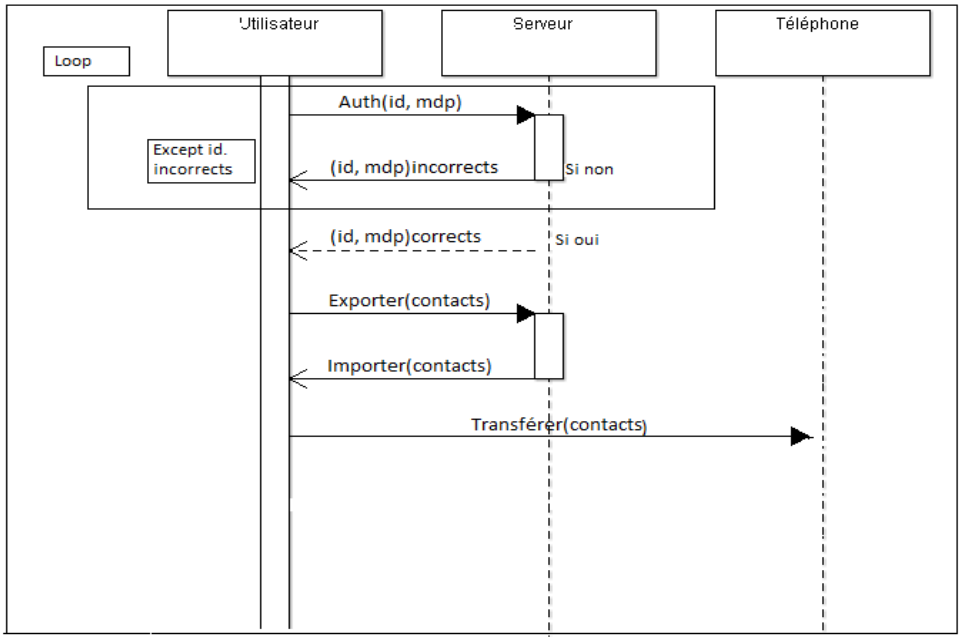

Figure II.4 : Diagramme de séquence.

L'utilisateur a besoin d'un identifiant et d'un mot de passe pour se connecter. Si ceux-ci sont corrects, alors il accède à l'application et peut choisir l'action qu'il veut effectuer.

III.3.3. Diagramme des classes

Nous avons élaboré le diagramme des classes final de notre système. La figure III.5 illustre ce diagramme.

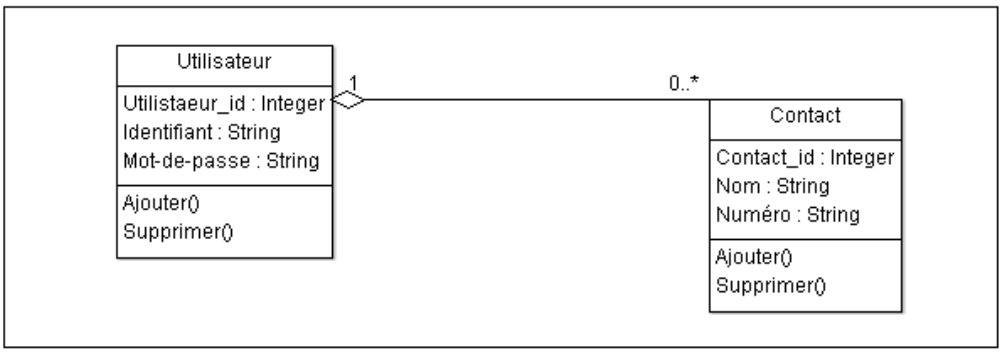

Figure III.5 : Diagramme de classe.

En conclusion, le processus unifié met en place un cadre général (Framework) permettant le développement d'un produit logiciel. Il prend en considération les cycles, les enchaînements d'activités, la réduction des risques, le contrôle qualité et la gestion de projet.

Chapitre III : Mise-en-œuvre

Notre application a pour but :

- De mettre en place une page d'authentification.
- D'accéder au répertoire téléphonique.
- De le transférer les contacts vers le répertoire d'un autre téléphone doté du SE Android.
- D'exporter les données vers un serveur.
- D'importer les données du serveur vers le téléphone.

Pour pouvoir répondre à ces objectifs, nous avons mis en place WAMPServeur, que nous avons connecté à l'application via les composantes proposées par le SDK Android.

III.1. XAMPPServer

XAMPP est un acronyme pour Windows, Apache, MySQL et PHP.

III.1.1. Apache

Le logiciel libre Apache est un serveur HTTP. C'est donc ce dispositif qui va nous permettre de stocker les données nécessaires à l'application tels les identifiants et les mots de passe des utilisateurs et les contacts.

Cependant, pour des raisons de sécurité, Apache est configuré pour n'accorder la permission d'accès au serveur qu'à la machine sur laquelle il est installé.

Pour permettre à notre application de se connecter, il suffit d'ajouter la permission dans le fichier de configuration d'Apache (httpd.conf) et de remplacer la directive « ALLOW FROM 127.0.0.1 » par « ALLOW FROM ALL ».

Le serveur va ainsi pouvoir répondre automatiquement aux requêtes provenant de l'application.

III.1.2. MySQL

MySQL est un système de gestion de base de données relationnel (SGBDR).

Nous allons donc pouvoir créer une base de données pour l'application qui va contenir les tables :

- UTILISATEUR (Utilisateur_id, Identifiant, Mot-de-passe) ;
- CONTACT (Contact_id, #Utilisateur_id, Nom, Numéro).

L'interface « phpMyAdmin » fournie avec WAMPServeur permet de gérer la base de données : la création de la base, la création des tables, les utilisateurs de la base de données (administrateurs) et leurs privilèges.

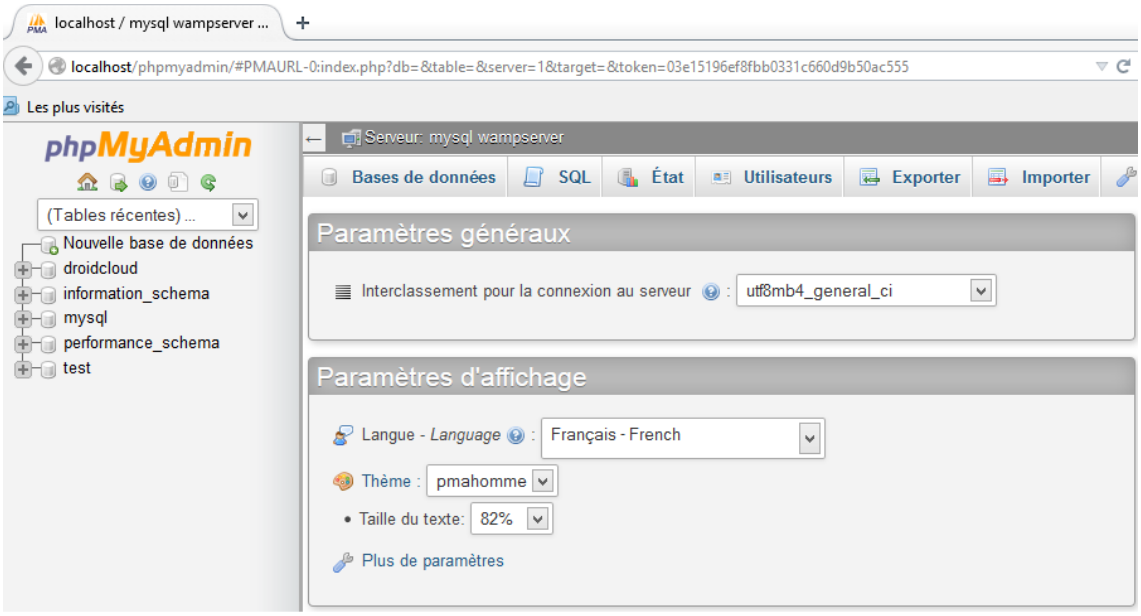

Figure III.1 : phpMyAdmin, interface permettant la gestion de la base de données.

III.1.3. PHP

Hypertext Preprocessor (PHP) est un langage impératif orienté objet utilisé principalement pour produire des pages Web dynamiques via un serveur HTTP.

Dans notre cas, les scripts vont servir à faire la liaison entre l'application et la base de données.

Pour chaque opération à effectuer, nous avons un script.

Ainsi, si nous décomposons chaque objectif en plusieurs opérations, nous obtenons :

- L'utilisateur crée un compte (première utilisation) :
  - Connexion à la base de données.
  - Vérification de l'unicité du pseudonyme (sélection).
  - Insertion des lignes dans la table.
  - Déconnexion.
- L'utilisateur se connecte (vérifier que l'identifiant existe et que le mot de passe correspond) :
  - Connexion à la base de données.
  - Sélection de l'identifiant (s'il existe) et du mot de passe correspondant.
  - Déconnexion.
- L'utilisateur exporte ses contacts sur le serveur :
  - Connexion à la base de données.
  - Insertion des contacts dans la table CONTACT.
  - Déconnexion.
- L'utilisateur importe ses contacts vers le téléphone.
  - Connexion à la base de données.
  - Sélection des contacts dont le champ Utilisateur_id est identique à celui de l'utilisateur.
  - Déconnexion.

En regroupant les opérations identiques, nous obtenons six scripts : un script pour insérer de nouveaux utilisateurs, un pour faire une sélection sur la table UTILISATEUR, un pour insérer des contacts, un pour sélectionner les lignes de la table CONTACT et enfin un pour ouvrir et fermer la base de données.

III.2. Eclipse

Eclipse est un EDI s'appuyant principalement sur le langage de programmation JAVA. Additionné du SDK Android, nous disposons ainsi de toutes les ressources nécessaires à l'implémentation de l'application.

III.2.1. Accès au serveur

Il s'agit d'implémenter la partie logicielle qui va permettre de se connecter au serveur afin de communiquer avec la base de données.

Nous implémentons tout d'abord les interfaces graphiques permettant à l'utilisateur de s'identifier ou bien de créer un compte.

Figure III.3 : Capture d'écran : connexion au compte.

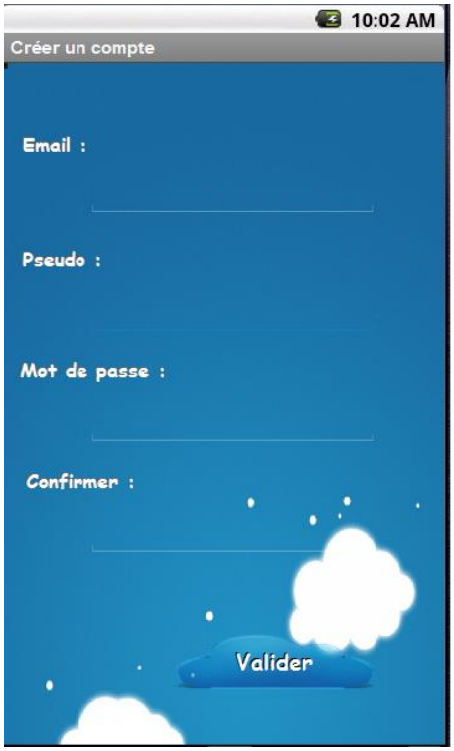

Figure III.4 : Capture d'écran : créer un compte.

Voilà ce qui se passe lorsque l'utilisateur clique sur l'un des boutons « Se connecter » ou « Valider » :

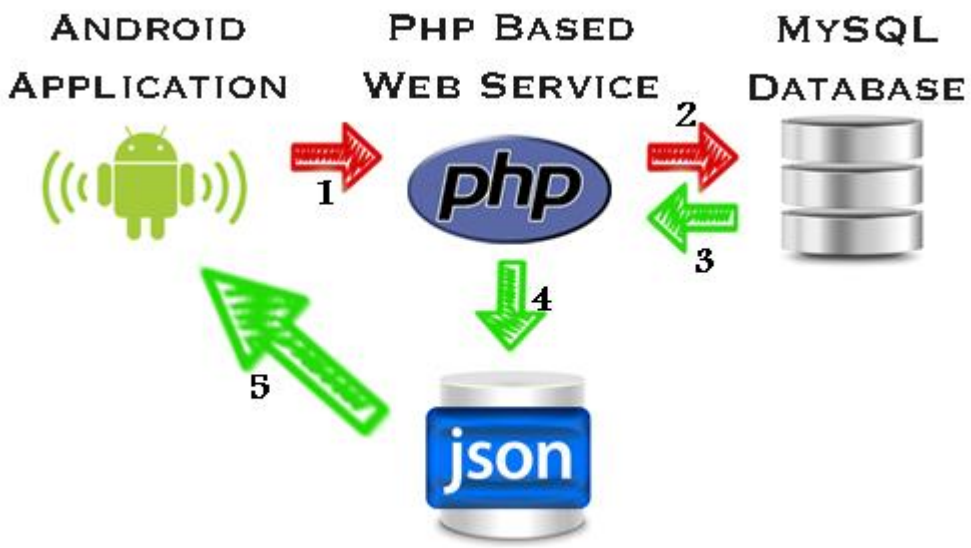

Figure III.2 : Architecture du système.

1. L'application envoie une requête HTTP au serveur avec l'adresse du script php dont elle a besoin. Le serveur se charge de trouver le script en question.

2. L'accès à la base de données se fait via les fichiers php (WebServices).

3. La base de données se charge d'insérer les données dans les tables (méthodes POST) ou bien de renvoyer le résultat d'une sélection (méthodes GET).

4. Le résultat retourné est en format *Java Script Object Notation* (JSON). Ce format permet de représenter de l'information structurée.

5. Le résultat est transféré à l'application. Il suffit ensuite de convertir le résultat pour ensuite le réutiliser (cette conversion est dite *parsing*).

III.2.2. Transfert de données entre deux téléphones

Figure III.5 : Capture d'écran : Menu.

Pour permettre à une application Android de transférer des données à un autre téléphone doté de la même application, nous avons utilisé WiFiManager. Cela va permettre de communiquer via Wi-Fi sans point d'accès intermédiaire.

Ce système va pouvoir nous permettre de déclarer le téléphone en charge de transférer le répertoire comme HotSpot (serveur), celui qui va recevoir les données est le client.

Après que les téléphones aient été connectés, l'envoie des données se fait via l'utilisation de sockets.

CONCLUSION GENERALE

L'objectif de ce projet est de sauvegarder les données d'un téléphone mobile afin de pouvoir les récupérer en cas de perte ou échange de portable

Pour se faire, nous avons conçu et implémenté une application mobile destinée aux systèmes d'exploitation Android.

Cette application permet l'envoie de données (le répertoire téléphonique) vers un serveur ainsi leur restitution, et l'échange de ces données entre deux téléphones.

Pour la conception de notre application, nous avons utilisé le processus de développement UP basé sur le langage de modélisation UML, et pour la mise-en-place, nous avons programmé avec Java grâce à l'EDI Eclipse.

Ce projet nous a permis de mettre en œuvre les connaissances que nous avons acquises en menant le développement d'un produit logiciel de la conception à l'implémentation. Nous avons pu faire le lien entre tous les modules que nous avons étudié : génie logiciel, réseau, base de données et algorithmique.

La principale difficulté de ce projet a été de mettre en place un serveur local et de le configurer pour pouvoir accéder à la base de données à partir de l'application.

L'idéal serait cependant de pouvoir envoyer des données vers un Cloud (Cloud Computing).

Nous pourrions également utiliser le principe de « wifiManager » pour développer une application proposant une galerie de partage (contacts mais aussi photos, musiques…).

Référence :

[1] : Wikipedia.com

[2] : Linformaticien.com

[3] : opensource.org

[4] : générationmobile.net

[5] : tutos.android.com

[6] : developer.android.com

[7] : Developper.com