Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

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

Mémoire de fin d'études

Pour l'obtention du diplôme de Licence en Informatique

Thème

Conception et Developpement d'une application Java sous Android

(Donner du sang)

Réalisé par :

- **BOUKLI HACENE abdel hafid**
-

Présenté le 26 mai 2015 devant la commission d'examination composée de MM.

- *Mme Halfaoui A.* *(Examinateur)*
- *Mme El Yebdri Z.* *(Examinateur)*
- *Chouiti S.* *(Encadreur)*

Année universitaire : 2014-2015

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. CHOUITI sidi Mohammed 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.

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

A mon frère et ma sœur

A tout ma famille.

A tous ceux qui me sont chers

Table des matières

Résumé

Notre objectif principal est de développer une application mobile sous Android utile à la santé de tout le monde. Pour cela nous avons du, s'initier à concevoir en UML, manipuler des langages comme java et XML et maitriser un environnement de développement dédié à ce type de d'application.

Nous avons comparé plusieurs plateformes de développement à savoir android studio, windev mobile et nous avons choisi éclipse.

Notre application consiste

- Trouver les donneurs de sang les plus proches.
- Envoyer un message d'alerte aux donneurs par un simple clic de bouton
- Auto-évaluer l'aptitude du donneur par un court questionnaire

Introduction générale

Avec l'avancée et l'émergence des technologies mobiles, les développements embarqués sont de plus en plus demandés sur le marché. Avoir un Smartphone est devenu incontournable pour les jeunes de nos jours. D'où nait l'idée de développer une application utile et conviviale. Il s'agissait au cours de notre projet d'étudier et de développer une application mobile sous Android facilitant le don du sang.

Mis à part le développement proprement dit de l'application, la première étape consistait à nous familiariser avec l'environnement Android, puis de choisir les outils conviviaux et envisageables à l'aboutissement du projet. Par la suite, nous avons conçu et développer cette application.

Aussi, nous avons créé un serveur web pour pouvoir stocker les données dans une base de données.

Ce rapport peut ainsi être subdivisé en quatre parties :

- ✓ Le premier chapitre présente l'application mobiles et Les différents systèmes d'exploitation mobiles
- ✓ Le deuxième chapitre présente le système d'exploitation ANDROID
- ✓ Dans le troisième chapitre, nous nous intéressons à l'environnement de développement et les différents outils utilisés dans cette application
- ✓ le quatrième chapitre sera réservé à présenter l'application et à expliquer son fonctionnement.
- ✓ Enfin nous donnons une conclusion et quelques perspectives.

Chapitre1 : Android et les SE mobiles

1. Définition

Android et un système d'exploitation mobile pour Smartphones, tablettes tactile, smartwatches (version Wear) et terminaux mobiles. C'est un système open source utilisant le noyau Linux

2. Historique

En juillet 2005, Google a acquis Android Inc., une petite startup qui développait des applications pour téléphones mobiles. C'est à ce moment là que des rumeurs sur l'entrée de Google dans le secteur du mobile ont commencé. Mais personne n'avait des données sûres à propos des marchés dans lesquels ils allaient se positionner.

Après ce rachat fait par Google, une équipe dirigée par Andy Rubin, un ancien d'Android Inc, a commencé à travailler sur un système d'exploitation pour appareil mobile basé sur linux. Durant 2 ans, avant que l'OHA soit crée officiellement, un certain nombre de rumeurs ont circulé au sujet de Google. Il a été dit que Google développait des applications mobiles de son moteur de recherche, qu'elle développait un nouveau téléphone mobile, etc.

En 2007, le 5 novembre, l'OHA a été officiellement annoncée, ainsi que son but : développer des standards open sources pour appareil mobile. Le premier standard annoncé a été Android, une plateforme pour appareils mobiles basée sur un kernel linux 2.6. [1]

En octobre 2008, apparait la première version d'Android qui n'avait pas reçu de nom. Cette version s'est avérée être la β du système. (Voir figure1.)

Version	Dernière révision	Nom de code	Date de sortie
1.0	1.0	Apple pie[13]	11 novembre 2007
1.1	1.1	*Bananas split*[13]	22 octobre 2008
1.5	mai 2010	*Cupcake*[13]	30 avril 2009
1.6	mai 2010	*Donut*[13]	15 septembre 2009
2.0	2.1, mai 2010	*Eclair*[13]	26 octobre 2009
2.2.x	2.2.3, 2011	Froyo[13]	20 mai 2010
2.3.x	2.3.7, 2012	*Gingerbread*[13,14]	6 décembre 2010
3.x.x	3.2.6, 2012	*Honeycomb*[15]	22 février 2011
4.0.x	4.0.4, 2012	*Ice Cream Sandwich*[13,17,18]	19 octobre 2011
4.1.x	4.1.2, 2012	*Jelly Bean*[19]	9 juillet 2012
4.2.x	4.2.2, 15 février 2013	*Jelly Bean*	13 novembre 2012
4.3.x	4.3.1, 24 juillet 2013	*Jelly Bean*	24 juillet 2013
4.4.x	4.4.4, 19 juin 2014	*KitKat*[20,21]	31 octobre 2013
5.0.x	5.0.2, 19 décembre 2014	*Lollipop*	3 novembre 2014
5.1.x	5.1.0, 3 avril 2015	*Lollipop*	9 mars 2015[22]

Figure 1 évolution des versions d'Android. [2]

3. Contraintes liées au développement sous Android :

Différentes contraintes sont à prendre en compte lors du développement dans cet Environnement mobile :

- Il faut pouvoir interagir avec un système complet sans l'interrompre. Android fait des choses pendant que votre application est utilisée, il reçoit des SMS et des appels, entre autres.

- Il faut respecter une certaine priorité dans l'exécution des tâches.

- Il faudra exploiter tous les outils fournis afin de débusquer les portions de code qui nécessitent des optimisations

- La taille de l'écran est réduite, et il existe par ailleurs plusieurs tailles et résolutions différentes.
- L'interface graphique doit s'adapter à toutes les tailles et toutes les résolutions, ou il aura des risques de laisser de côté un bon nombre d'utilisateurs.

- Enfin, Près de huit versions Android 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. [3]

4. Architecture Android :

Le diagramme suivant illustre les composants principaux du système d'exploitation Android. Chaque section sera décrite dans ce qui suit :

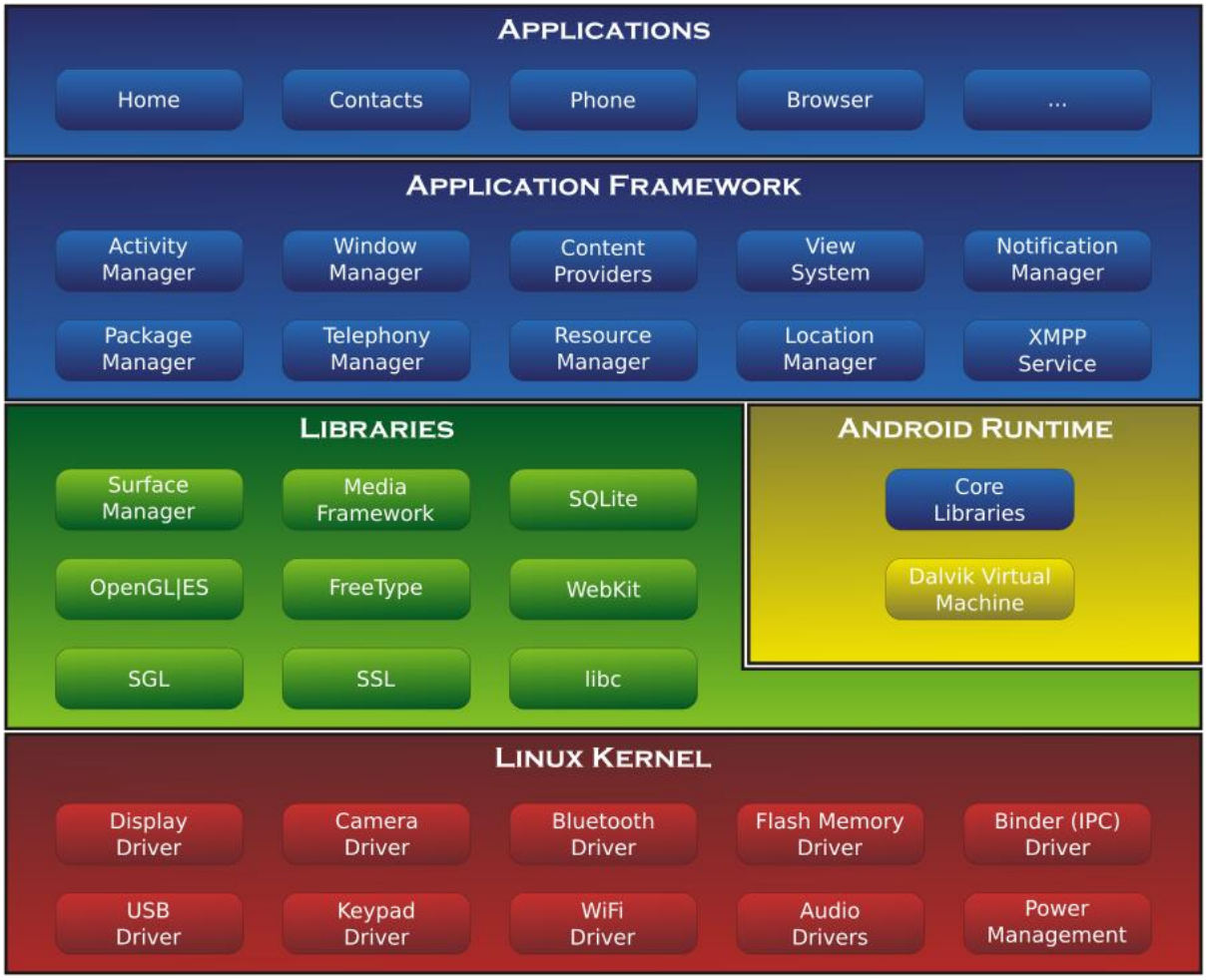

Figure 2 Architecture de la plateforme Android [4]

Android est basé sur un kernel linux 2.6.xx.

Au-dessus de cette couche, on retrouve les librairies C/C++ utilisées par un certain nombre de composants du système Android.

Au-dessus des librairies, on retrouve l'Android Runtime. Cette couche contient les librairies cœurs du Framework ainsi que la machine virtuelle exécutant les applications.

Au-dessus de la couche "Android Runtime" et des librairies cœurs, on retrouve le Framework permettant au développeur de créer des applications.

Enfin au-dessus du Framework, il y a les applications. [4]

5. Les différents systèmes d'exploitation mobiles sur le marché :

Il existe sur le marché des dizaines de systèmes d'exploitation différents : Symbian OS de Nokia, ios d'Apple, BlackBerry OS de RIM, Windows Phone de Microsoft, Bada de Samsung et Android de Google…etc

Symbian OS :

Le Symbian OS est développé par la société éponyme qui est une propriété exclusive de Nokia. Bien que cette plateforme soit crée par la participation de plusieurs fabricants tels que Samsung ou Sony Ericsson, ce système est fortement connoté Nokia, ce qui est un frein à son adoption par d'autres constructeurs. Il est récemment passé en open source. C'est un système libre, open source se base sur un noyau Symbian. [5]

Ios :

IOS (*Internetwork Operating System*), qui était nommé iPhone OS, se trouve non seulement sur les différents générations de iPhone mais également sur d'autres produits de Apple iPad et iPod touch. Il est dérivé de Mac OS X dont il partage les fondations : kernel, les services Unix et Cocoa. Pour Apple, le succès est considérable : début 2009, il n'y avait pas moins de 5 millions de téléchargements par jour. Donc, il s'agit du concurrent numéro un pour Android.[5]

BlackBerry OS

Le système d'exploitation BlackBerry est la plate-forme exclusive mobile développé par RIM (*Research In Motion*) exclusivement pour ses Smartphones BlackBerry et les appareils mobiles. RIM utilise ce système d'exploitation pour soutenir des fonctions spécialisées, notamment le trackball de la marque, molette, le trackpad et l'écran tactile. [5]

Windows Phone

Windows Mobile, est l'OS (système d'exploitation) mobile de Microsoft. C'est une évolution de Windows Pocket PC, ancêtre de Windows CE. Cet OS a réussi au fil des années à s'octroyer une part de marché honorable. Son succès est dû à son affiliation à la famille d'OS Windows, ultra-dominante sur le bureau. Un autre avantage souvent cité est la facilité de développement apportée grâce à l'environnement cliquodrome de Visual Studio qui a su faire venir au développement mobile les développeurs VB (*Visual Basic*). **[5]**

Android :

Android de Google Inc. fut développé par une petite startup qui fut acheté par Google qui poursuit activement son développement. Android distribué sous licence open source, est une variante de Linux. Google a lancé Open Handset Alliance qui regroupe des grands constructeurs et développeurs de logiciels (tel qu'Intel, HTC, ARM, Samsung, Motorola and eBay). Ce système est assez nouveau (relativement parlant) auprès des programmeurs. Il a eu douze versions, chacune portant un « nom de code » spécifique.[5]

➢ En septembre 2014, la part de marché mondiale d'Android est passée à 85 % (voire figure 3).

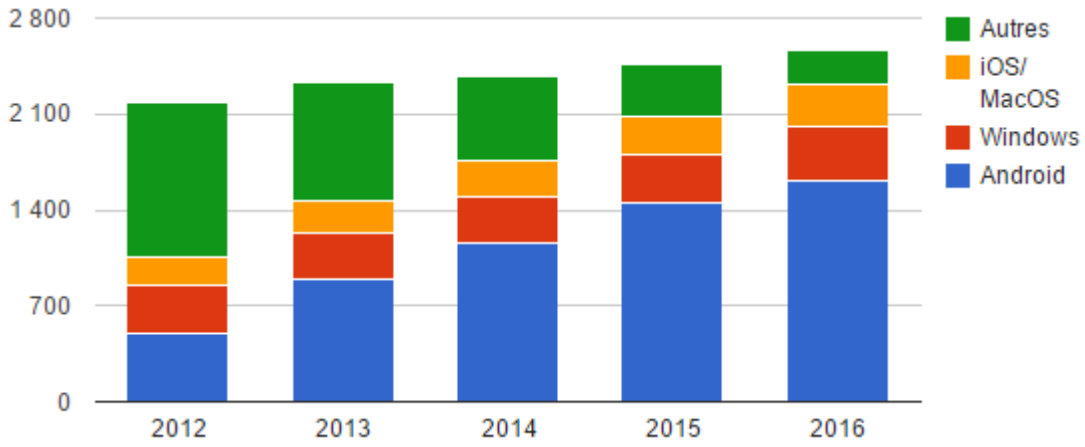

Figure 3 système d'exploitation des terminaux livrés dans le monde (millions d'unités /janvier) [6]

Chapitre 2 : Les applications mobiles

Les technologies mobiles prennent de plus en plus de place sur le marché. 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 Smartphones sont considérés comme des petits ordinateurs et 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.

1 L'application mobile :

Une application mobile est un logiciel applicatif développé pour être installé sur un appareil électronique mobile, comme un Smartphone, une tablette ou un baladeur numérique.

Une application mobile peut être soit installée directement sur l'appareil dès sa fabrication en usine soit téléchargée depuis un magasin d'applications dit « application store » telle que Google Play, l'App Store ou encore le Windows Phone Store. Une partie des applications disponibles sont gratuites tandis que d'autres sont payantes.

Il existe plusieurs systèmes d'exploitation mobiles (OS) dont les plus répandus sont les suivants :

- iOS (Apple) utilisé sur iPhone et iPad,
- Android (Google) qui anime un grand nombre de smartphones tels que Samsung, HTC, LG, Motorola…
- Blackberry OS,
- Windows Phone (Microsoft),
- Symbian (Nokia),
- Bada (Samsung). [7]

1.1 Les avantages d'une application mobile :

Une application mobile revêt de nombreux avantages comparés à un site web mobile, en voici 5 principaux :

- Un confort d'usage et une expérience utilisateur inégalée.
- L'accès direct aux contenus de l'application mobile via l'icône présent sur le Dashboard du téléphone ou de la tablette (mode d'accès sans URL).
- Un fonctionnement en mode déconnecté.
- elle permet d'utiliser et d'intégrer toutes les **fonctionnalités téléphone** (accéléromètre, gyroscope, GPS, caméra…), ce qui n'est pas forcément le cas des WebApps.

- L'implémentation de fonctionnalités natives comme par exemple les notifications « PUSH ». [8]

1.2 Les inconvénients d'une application mobile :

- Le principal inconvénient d'une application mobile est qu'elle doit respecter les règles définies par les différentes sociétés des plateformes mobiles. Que ce soit l'approbation nécessaire des Apps Store pour diffuser l'application ou ses mises à jour
  .
- les conditions tarifaires imposées ou la non compatibilité avec les autres systèmes d'exploitation mobiles.

- Le coût lié au développement d'une application mobile est un frein car généralement plus élevé si elle est portée sur plusieurs plateformes (afin d'être disponible pour un maximum de mobinautes) que le coût d'un site mobile ou d'une Web App. Il faudrait potentiellement prévoir un développement sur chaque technologie, et donc un coût supplémentaire si l'on souhaite se positionner sur tous les modèles.

- Pour que l'utilisateur ait accès à la dernière version, il faut qu'il la mette à jour depuis le store contrairement aux sites mobiles et WebApp qui se mettent à jour directement [8]

2 Présentation de notre application :

Parce qu'un don du sang ne dure qu'une heure et permet de sauver des vies, nous proposons une application qu'on a nommée **Donner du sang.**

Donner du sang est une initiative de sauvetage qui donne une chance d'être un héros par le don du sang à chaque utilisateur de Smartphone Android. Cette application permet à l'utilisateur de trouver et contacter les donneurs du sang qui se trouve dans le même territoire (la même zone géographique) dans les situations d'urgence partout dans le monde. Lors d'un besoin urgent, cette application vous aidera à envoyer des SMS à tout utilisateur inscrit qui possède le même groupe sanguin par un simple clic de bouton.

Notre application a pour but :

- De mettre en place une page d'authentification.
- De mettre en place un profile utilisateur
- De mettre en place un questionnaire prés don
- D'exporter les données vers un serveur et de les importer
- Envoyer un message SMS
- Envoyer un email

3 Conception en UML de notre application:

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.

3.1. Diagramme de 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 : chercher un donneur, envoyer un mail, consulter page profil.

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

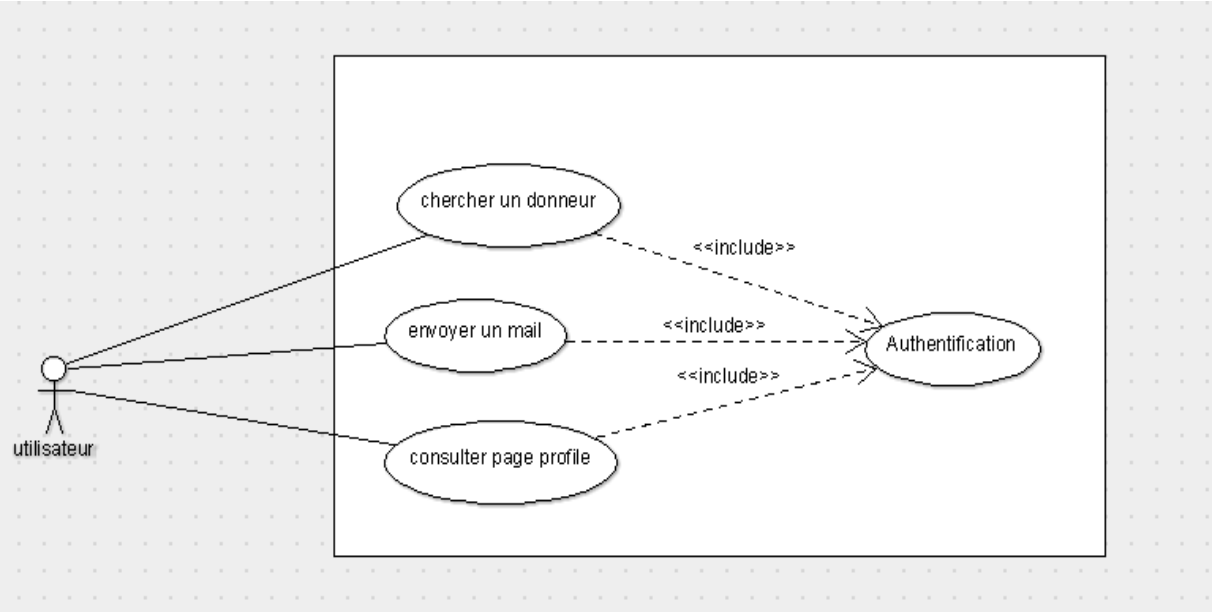

Figure 4 Diagramme des cas d'utilisation

3.2. Diagrammes de séquence :

Détaillons les cas d'utilisation. :

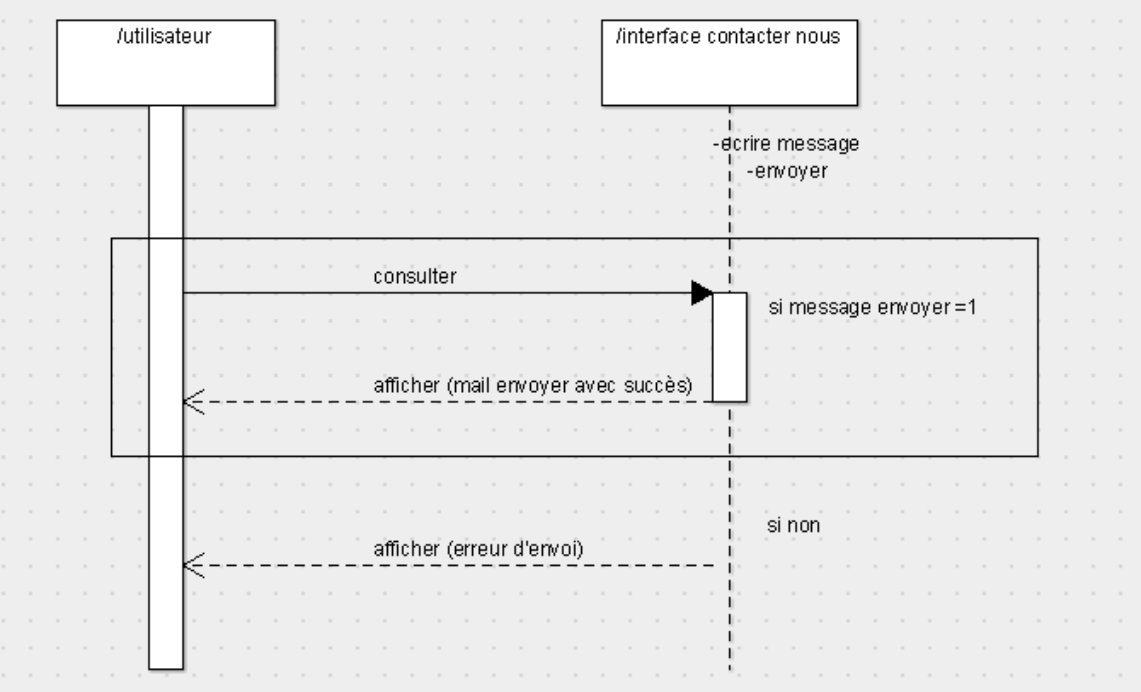

Figure 5 Diagramme de séquence envoyer un mail

Ce diagramme de séquence représente le scénario de l'envoie d'un mail l'utilisateur consulte la page contacter nous dans cette page va écrire et envoyer le mail si le message et envoyer avec succès le message « mail envoyer avec succès » s'affiche si non « erreur d'envoie ».

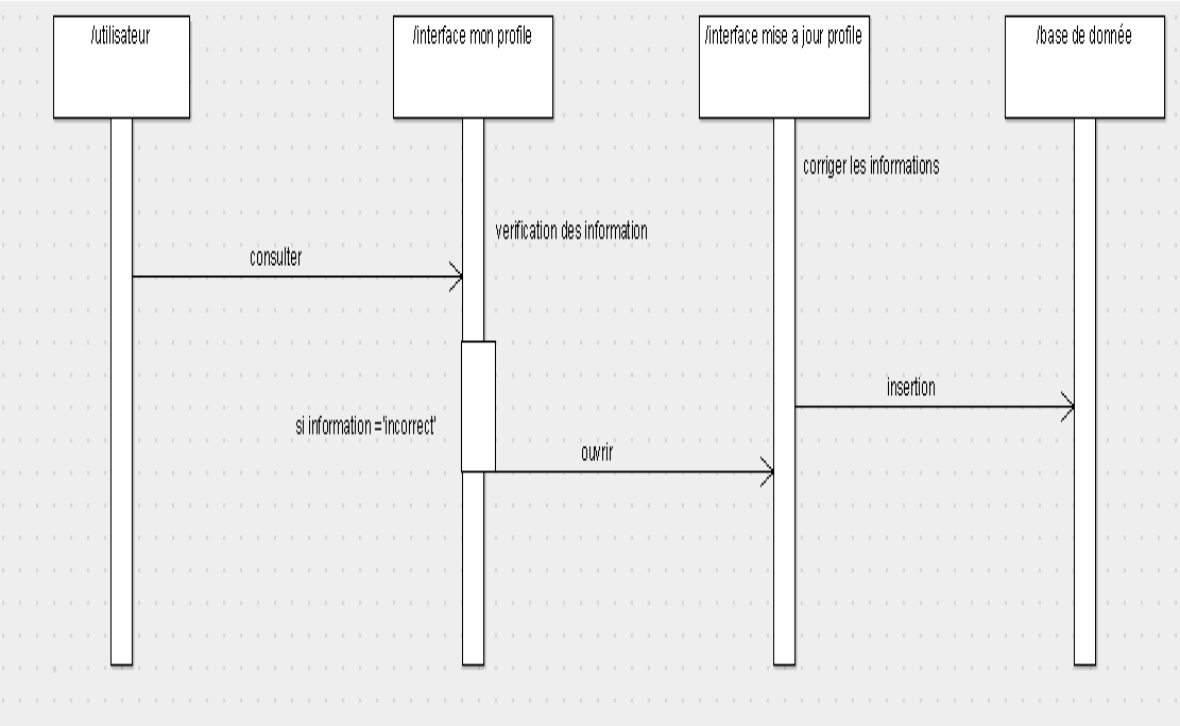

Figure 6 Diagramme de séquence mon profil

Ce diagramme de séquence représente le scénario de consultation du profile utilisateur, l'utilisateur consulte interface mon profile ou il vérifié c'est information personnel, si les informations sont incorrect alors il ouvre interface mise a jour ou il corrige c'est informations et il fait une insertion des donnée dans la base de donnée. si non ne rien faire

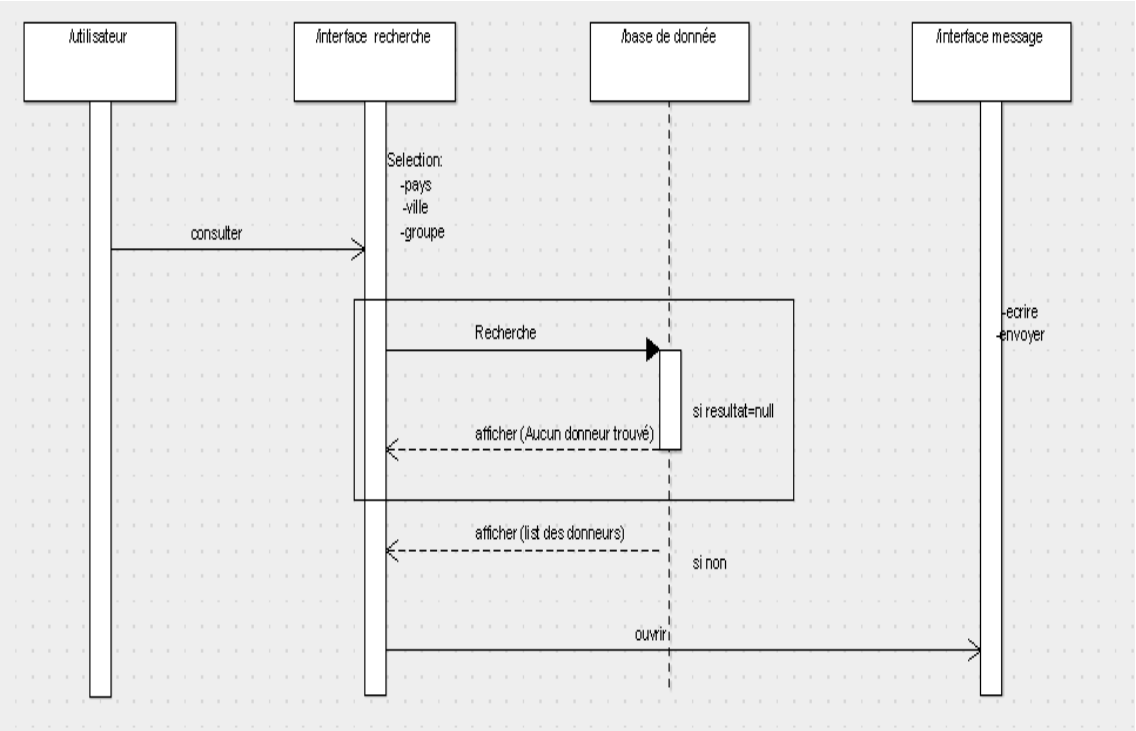

Figure 7 Diagramme de séquence chercher un donneur

Ce diagramme de séquence représente le scénario de la recherche d'un donneur. utilisateur ouvre interface recherche et fait une recherche dans la base de donnée par sélections du pays ville et groupe sanguin si le résultat=nul alors le message aucun donneur trouvé s'affiche si non la liste des donneurs qui s'affiche ,après sa utilisateur ouvre la boite d'envoie du message il tape sont message et il l'envoie

3.3. Diagramme de classe global :

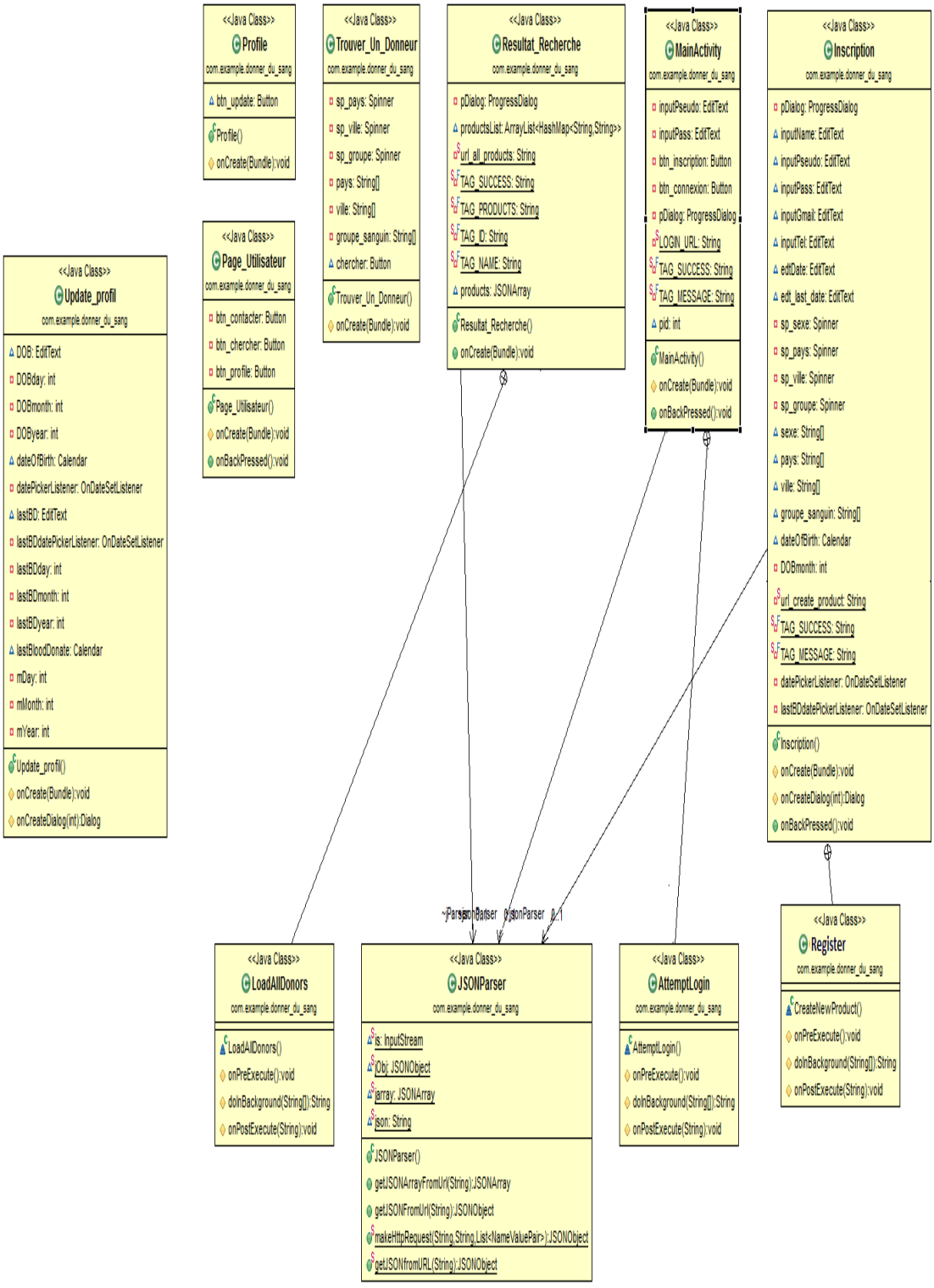

Figure 8 diagramme de classe global

Chapitre 3 : Environnement de développement et langages utilisés

Dans ce chapitre, nous présenterons l'environnement de travail qui inclut les outils de développement (logiciels et technologies exploités). Nous allons ensuit décrire les différentes langages utilisée.

1. Eclipse

1.1. Définition :

Eclipse est un projet, décliné et organisé en un ensemble de sous-projets de développements logiciels, de la Fondation Eclipse visant à développer un environnement de production de logiciels libre qui soit extensible, universel et polyvalent, en s'appuyant principalement sur Java.

Son objectif est de produire et fournir des outils pour la réalisation des Logiciels, englobant les activités de programmation (notamment environnement de développement intégré et frameworks).

Son EDI, partie intégrante du projet, vise notamment à supporter tout langage de programmation à l'instar de Microsoft Visual Studio. [9]

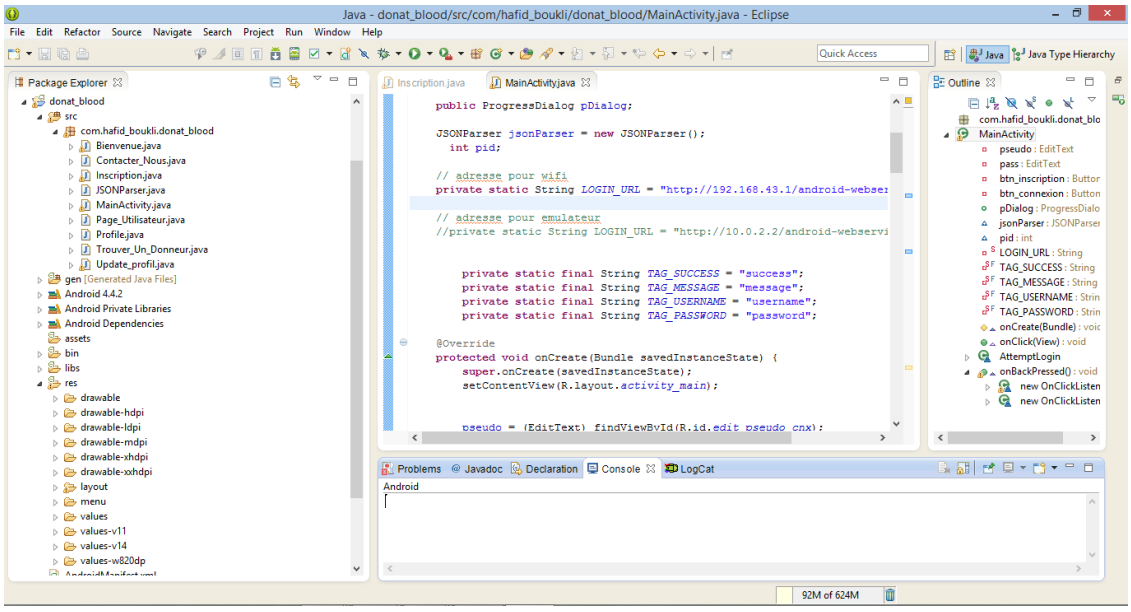

Figure 9 Capture d'écran de l'IDE Eclipse [9]

1.2. Le plugin de développement d'Eclipse : ADT

ADT (Android Developer Tools) est un plugin pour Eclipse qui fournit une suite d'outils qui sont intégrés à l'IDE Eclipse. Il vous offre l'accès à de nombreuses fonctionnalités qui vous aident à développer des applications Android. ADT offre un accès graphique à la plupart des outils SDK ainsi que d'un outil de conception de l'interface utilisateur pour le prototypage rapide, la conception, et la construction de l'interface utilisateur de votre application. [10]

Figure 10 Capture d'écran de l'ADT [10]

2. L'AVD Android:

Android Virtual Device est un dispositif mobile virtuel qui s'exécute sur l'ordinateur et permet de développer et de tester des applications Android sans l'aide d'un dispositif physique.

L'émulateur Android imite toutes les fonctionnalités matérielles et logicielles d'un dispositif mobile typique, tel que la lecture des fichiers audio et vidéo, stockage des données, sauf qu'il ne peut pas passer des appels réels.

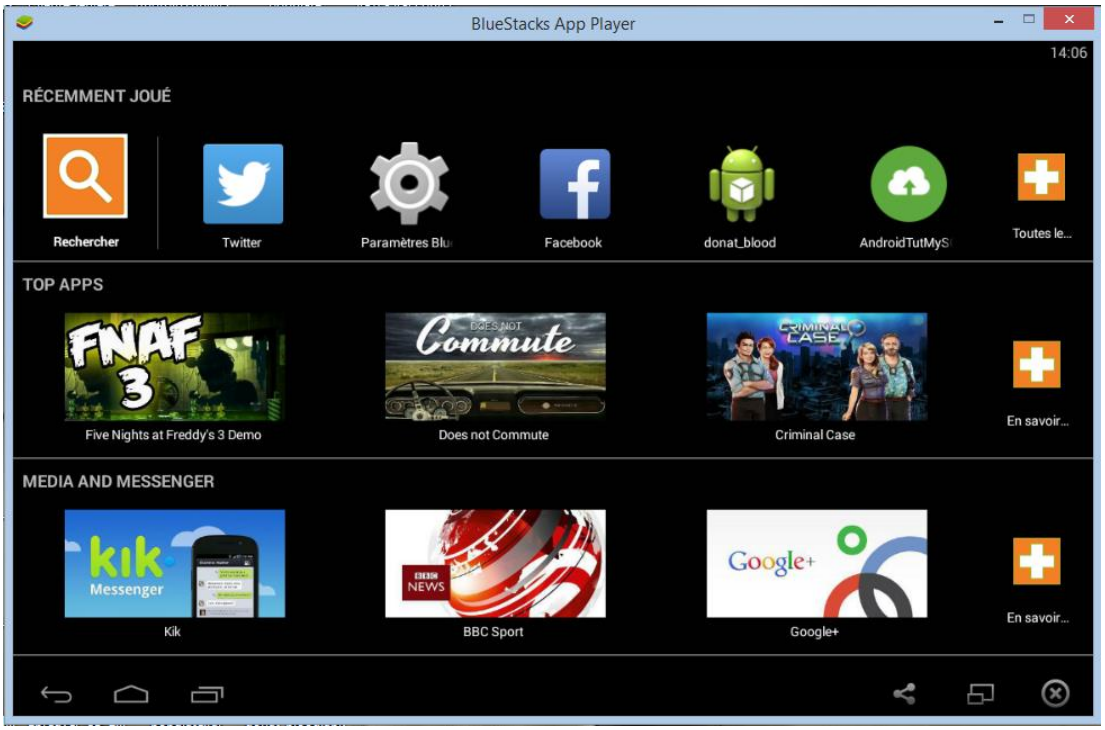

Figure 11 Exemple d'émulateur(BlueStacks)

3. Le SDK Android :

Le Software Developement Kit Android est un ensemble de fichiers d'aide et d'exemples.On y trouve aussi des utilitaires pour la mise au point et les tests. [11]

3.1. Les fonctions du SDK :

- Accès au Hardware, y compris Camera, GPS, et Accéléromètre.
- Base de données SQLite.
- Données et dépôt de données partagées et communication inter application par échange de messages.
- Ecran d'accueil riche par l'utilisation des Widgets, Live Folders, and Live Wallpaper Support Média très riche et graphiques 2D/3D : Rendus graphiques par HW optimisé pour la mobilité, incluant une librairie " path - based " pour les rendus 2D et le support pour les graphiques 3D utilisant OpenGL ES 2.0
- Accès au HW Wifi et librairies pour l'utilisation du Bluetooth pour le transfert Peer to - Peer.
- Technologies réseau GSM, EDGE, et 3G pour la téléphonie ou le transfert de données, permettant de placer des appels téléphoniques, des SMS, et d'envoyer et de recevoir des données en utilisant les réseaux de données mobiles.

- API pour l'utilisation des capteurs HW y compris les accéléromètres et le compas.[11]

4. WampServer :

WampServer est une plateforme de développement Web , permettant de faire fonctionner localement (sans se connecter à un serveur externe) des scripts PHP. WampServer n'est pas en soi un logiciel, mais un environnement comprenant deux serveurs (Apache et MySQL), un interpréteur de script (PHP), ainsi que phpMyAdmin pour l'administration Web des bases MySQL.

5. Langages utilisés :

5.1. Java :

Le principal langage utilisé pour écrire des programmes Android est le Java, un langage orienté objet, utilisé pour développer de nombreux logiciels. Cependant, tous les programmes fonctionnant sur la machine virtuelle de Java (JVM : Java Virtual Machine), ne fonctionnera pas forcément sous Android. Celui-ci étant destiné à des appareils mobiles ayant peu de puissances (par rapport à un ordinateur classique), Google a développé sa propre machine virtuelle : Dalvik. Cette machine se base sur JVM mais certaines classes disponibles sous JVM ne le sont pas sous Dalvik.

Un programme Java est constitué de Classe d'objet, combinant à la fois les données utilisées (appelées Propriétés) et le code manipulant celles-ci (appelé Méthodes). Le code devient logiquement découpé en petites entités cohérentes et devient ainsi plus simple à maintenir et plus facilement réutilisable, étant intrinsèquement modulaire.

```
Public class Toto
{
	int age;
	char sexe;
	float taille;

	Infos(int age, char sexe, float taille)
	{
		System.out.println(" age : %d, sexe : %c et taille : %f",
		age,	sexe, taille);

	}
}
```

Figure 12 Exemple classe Java

Dans cet exemple, la classe est « Toto », les propriétés sont l'âge, le sexe et la taille et la méthode est « Infos » qui va manipuler les propriétés en les affichant.

5.2. XML :

Le second langage nécessaire à connaitre pour développer une application Android est le XML qui signifie eXtensible Markup Language (en français : langage extensible debalisage). XML n'est pas un langage de programmation, il n'y a pas de boucle for, de if, de while,..... . Il est presque exclusivement utilisé pour stocker (ou transférer d'un programme à un autre) des données (du texte) de façon structurée.

Ce langage utilise des balises (comme le HTML). Ces balises sont ouvrantes, <Balise ouvrante> ou fermantes, </balise_fermante>. Chaque balise ouvrante devra être fermée à un moment. Il est possible de mettre du texte entre la balise ouvrante et la balise fermante. Les balises peuvent être imbriquées : on peut insérer un ou plusieurs couples de balises (ouvrante et fermante) entre 2 balises (ouvrante + fermante) voici un exemple permettant de stocker des informations à propos des films d'un cinéma :

```
<cinema>
        <film>
                <nom>Les deux tours</nom>
                <realisateur>P Jackson</realisateur>
                <annee_sortie>2002</annee_sortie>
        </film>
        <film>
                <nom>Bladerunner</nom>
                <realisateur>R Scott</realisateur>
                <annee_sortie>1982</annee_sortie>
        </film>
</cinema>
```

Figure 13 Exemple XML

La balise <cinema> est appelée "élément racine", la balise <film> est le "premier enfant",etc. Un fichier XML doit posséder un "élément racine". Lorsqu'il n'y a pas de texte entre la balise ouvrante et la balise fermante, on peut les remplacer par <balise/> (qui est à la fois la balise ouvrante et la balise fermante). Une balise peut contenir des attributs :

<balise nom="Toto" prénom="Jean Pierre"/>

Nom et prénom sont des attributs qui seront utilisés par un programme tiers. Un fichier XML doit toujours commencer par une ligne appelée prologue :

<?xml version="1.0"encoding="utf-8"?>

Cette ligne précise la version de XML utilisée (ici, c'est la version 1.0) et le type d'encodage pour le document. Voici un exemple de fichier XML utilisé lors du développement d'applications sous Android.

```
<?xml version="1.0"encoding="utf-8"?>
<LinearLayout
      xmlns:android=http://schemas.android.com/apk/res/android
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">

      <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/hello"/>

</LinearLayout>
```

Figure 14 Exemple XML Android

5.3. PHP ET MYSQL :

5.3.1.PHP :

PHP est un open source langage de script rapide et riche en fonctionnalités pour développer des applications Web ou Internet / Intranet Applications.

5.3.2.MYSQL :

MySQL est un serveur puissant de base de données open source intégré basé sur un système de gestion de base de données relationnelle (SGBDR) et est capable de gérer une grande base de données de connexion simultanée.

5.4. JSON (JavaScript Object Notation) :

Format de données textuel, générique, dérivé de la notation des objets du langage ECMAScript.

Chapitre 4 : Implémentation

Dans ce chapitre, nous expliquons l'implémentation des différents services utilisées ensuit nous présenterons les différentes interfaces de notre application et enfin Nous allons décrire les quelque ligne de code .

1. Utilisation de WampServer :

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 et les groupes sanguin des utilisateurs…etc.

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.

2. La base de données MYSQL :

Nous allons donc pouvoir créer une base de données pour l'application qui va contenir une seule table

- INSCRIPTION (Utilisateur_id, pseudo , Mot-de-passe , Nom_Prenom ,Date_de_naissance , Sexe, Num_telephone , Gmail , Groupe_sanguin ,Pays ,Ville ,Date_du_dernier_don) ;

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

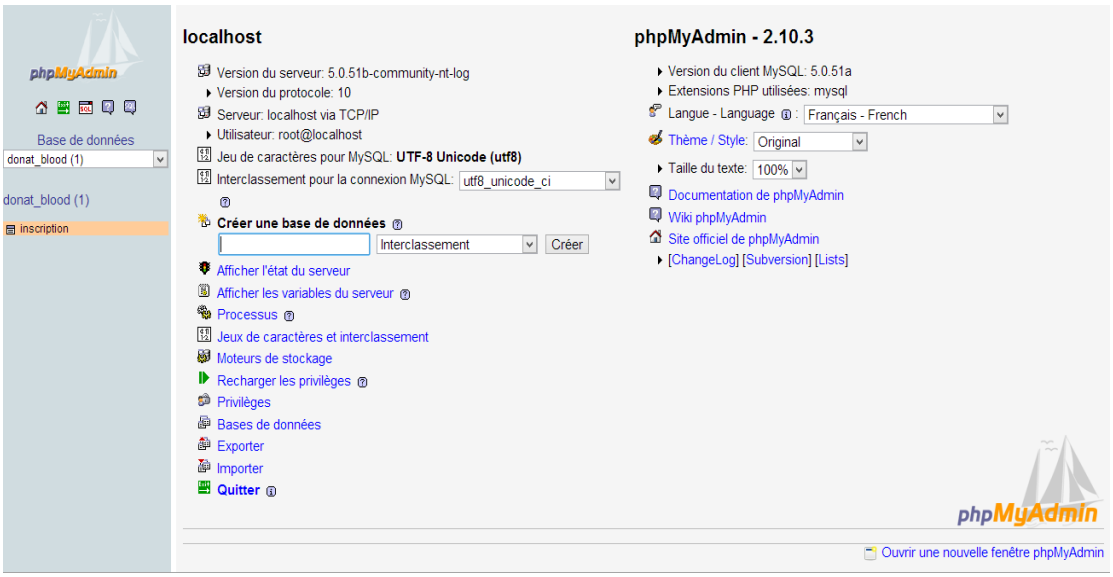

Figure 15 phpMyAdmin, interface permettant la gestion de la base de données. [12]

3. PHP :

Comme on à dit précédemment dans le chapitre 3 le 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 fait une recherche :
    - ✓ Connexion à la base de données.
    - ✓ Sélection des lignes dans la table INSCRIPTION par condition sur : le pays ,la ville ,groupe sanguin et date du dernier don
    - ✓ Déconnexion.

En regroupant les opérations identiques, nous obtenons cinq scripts : un script pour insérer de nouveaux utilisateurs, un pour faire une sélection sur la table INSCRIPTION, un pour sélectionner les lignes de la table inscription (scripte pour login), un pour faire une recherche sur les utilisateurs et enfin un pour ouvrir et fermer la base de données.

4. 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.

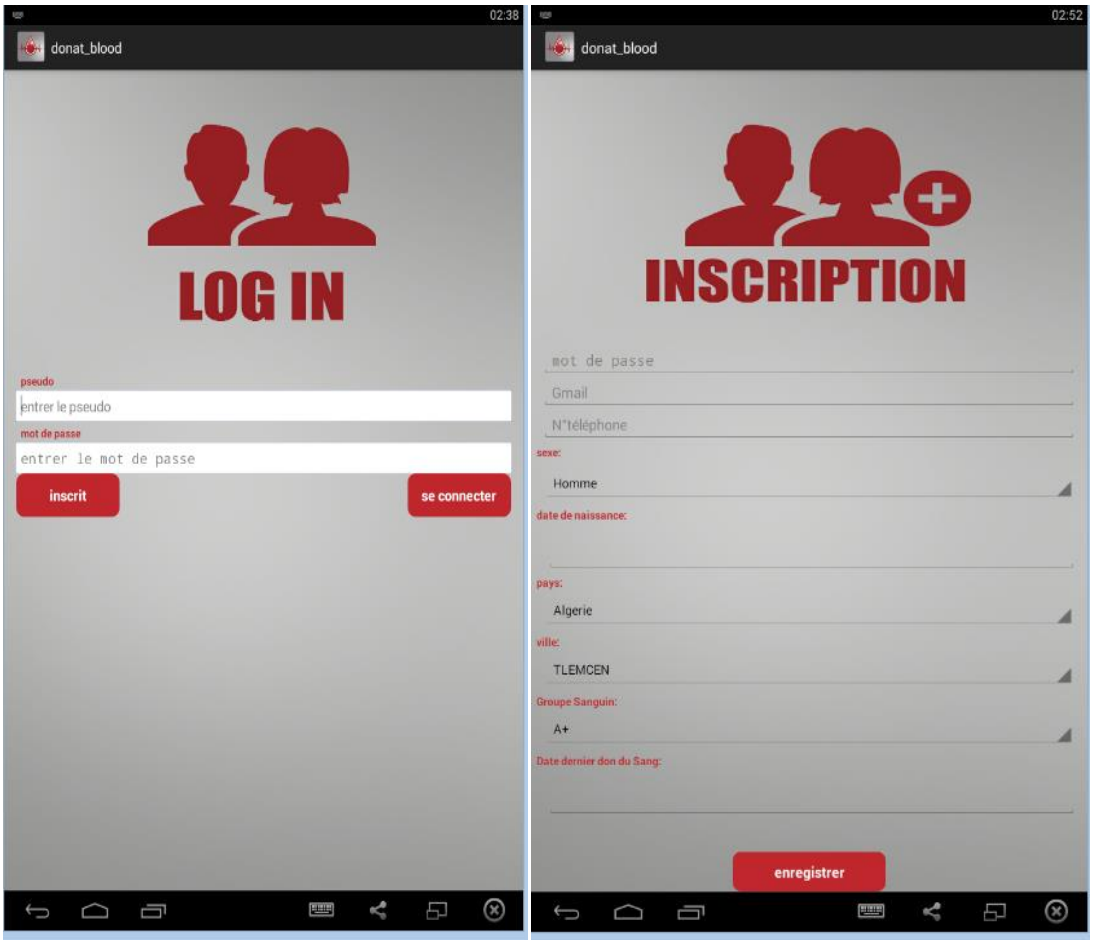

Figure 16 capture d'écran login et inscription

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

Figure 17 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*).

5. Les différentes interfaces de notre application :

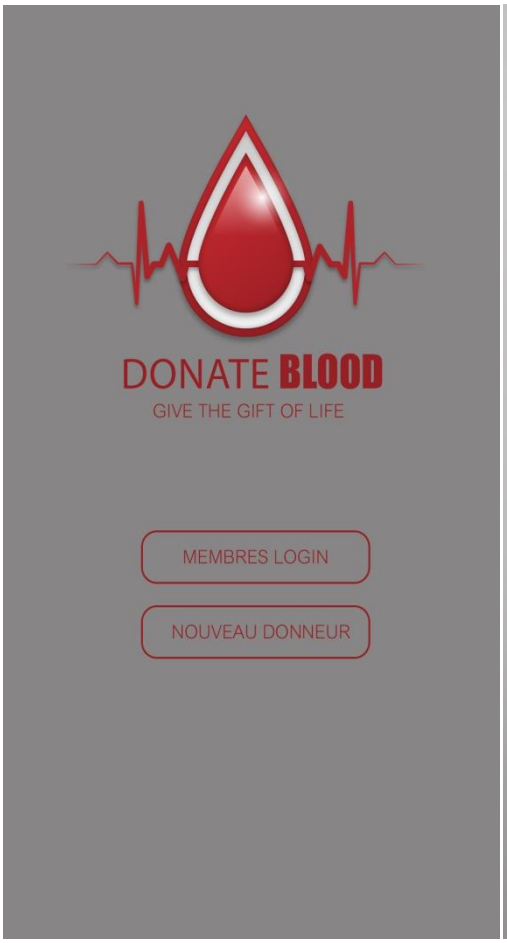

Figure 18 interface bienvenue

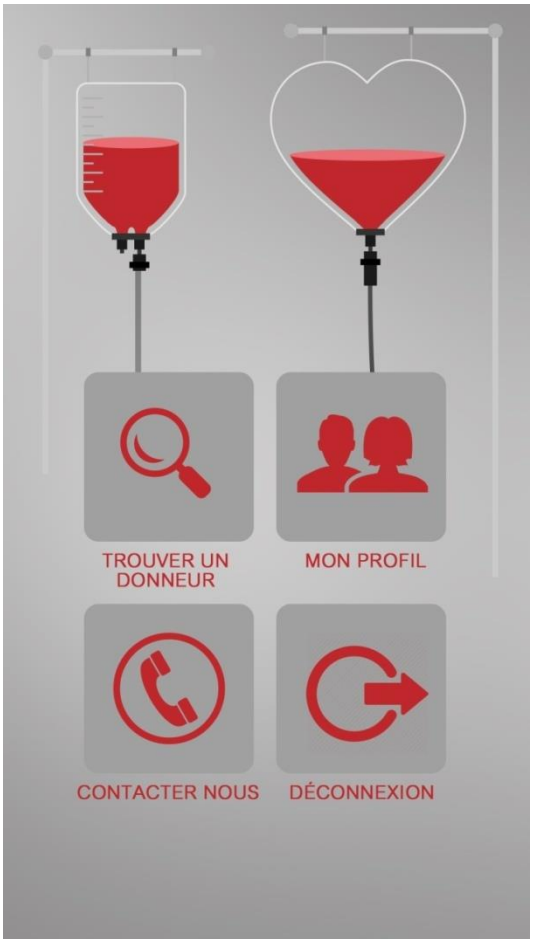

Figure 19 interface page utilisateur

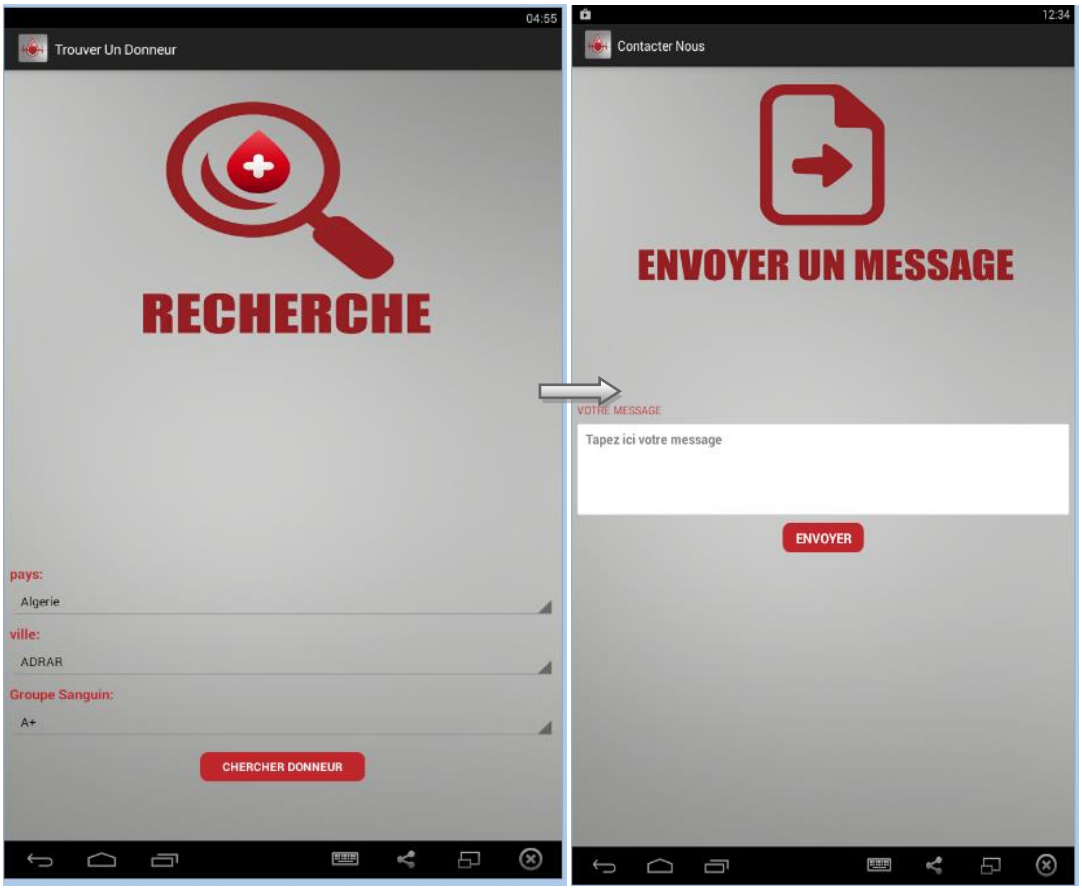

Figure 20 interface chercher donneur et envoyer message

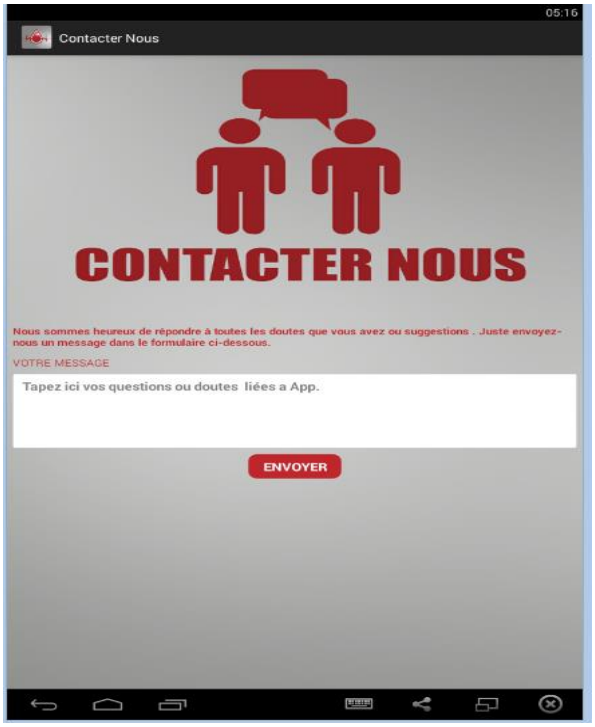

Figure 21 interface contacter Nous

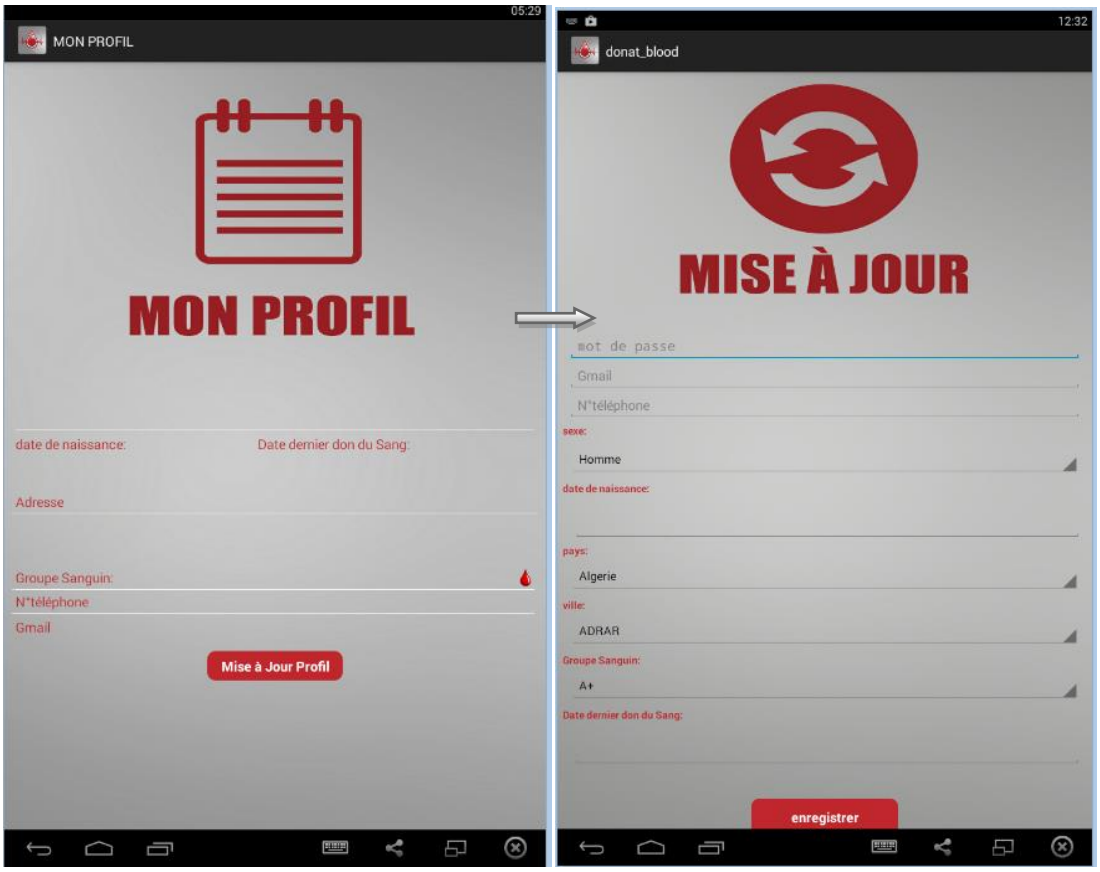

Figure 22 interface Mon profil et mise à jour profil

6. Description du code :

Une Activité Android :

Android gère les applications à travers une classe abstraite appelée une activité : elle est définie par la documentation comme étant « une chose qui à le focus que l'utilisateur peut faire. Presque toutes interagissent avec l'utilisateur, elles prennent donc en chargela création d'une fenêtre où placer l'interface utilisateur (UI). »

Notre programme Android «Mise_a_jour_profile » se présente donc sous la forme d'une classe implémentant la classe abstraite « Activity ».

```
public class Mise_a_jour_profile extends Activity {

    // our program

}
```

De manière générale une activité implémente souvent :

- onCreate(Bundle) : l'endroit où l'activité est initialisée, on y utilise souvent la méthode
- setContentView(int) avec une ressource de contenu qui définit l'UI
- onPause() : ici est traitée le cas où l'utilisateur quitte l'activité

Dans la mesure où il n'y a pas de données à sauvegarder ou de changement à mémoriser nous n'avons pas utiliser onPause().

Notre programme est donc constitué d'un appel à onCreate() qui va alors lancer l'affichage de l'interface, simplement constituée d'un texte et d'un bouton. Ceux-ci sont, comme le veut Android stockés dans un fichier XML nommé update_profil.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bonjour," />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=" enregistrer "
     />

</LinearLayout>
```

Ce fichier XML est la base de stockage des éléments qui vont se greffer à l'interfac graphique, on y reconnaît un texte (TextView) et un bouton (Button). A propos des attributs XML rencontrés on en remarquera quelques un :

- layout : tous les attributs incluant le mot layout font référence à la place en hauteur et en largeur que doit prendre l'application
- id : l'attribut id sert à référencer les éléments d'une façon plus simple et permet de les récupérer facilement grâce à la fonction findViewById(int) pour ainsi les manipuler aisément
- text : cet attribut permet de définir le texte tel que l'utilisateur le verra. Pour le bouton c'est celui qui apparaîtra sur celui-ci
- cette interface et appelée grace a la fonction setContentView(R.layout.update_profil).

```
setContentView(R.layout.update_profil);
```

Notez également que ce texte et ce bouton sont simplement initialisés juste avant l'appel à onCreate() :

```
Button b = null;
TextView myText = null;
```

Nous avons donc un texte suivi d'un bouton qui sont affichés dans l'UI, il est maintenant nécessaire d'associer au bouton l'action qui se produira lorsque l'utilisateur le « cliquera» : il faut le linker. Pour cela on se sert de la méthode setOnClickListener, on demande alors à l'activité d'« écouter » ce bouton et donc le message qui indiquera qu'il a été appelé.

```
b = (Button) findViewById(R.id.button);
b.setOnClickListener(this);
```

Le code décrivant l'action qui s'ensuit est décrit dans une fonction extérieur à onCreate(Bundle) est logiquement appelé onClick(View v).

```
public void onClick(View v) {

    // code

}
```

Le but du bouton est de lancer l'échange de donnée avec la base de données, avant Cela nous allons éditer un nouveau contenu sur l'UI. Cela est assuré par le TextView «myText » initialisé au début du code :

```
TextView myText = null;
```

Cette vue est ensuite « construite » dans le onClick(View v), on lui associe un texte grâce à la méthode de TextView setText(CharSequence text) puis on l'affiche finalement sur l'UI à l'aide de la méthode setContentView(View view) appliqué à notre TextView « myText ».

```
myText = new TextView(this);
myText.setText("Bonjour, vous avez lancé onClick.");
setContentView(myText);
```

Cette partie du code était donc pour informer l'utilisateur du déroulement des opérations, il faut maintenant s'occuper de l'échange de données. C'est à travers la classe ConnectionToServer que nous allons faire cela, nous allons donc en créer une instance et lancer la méthode qui va se charger du transfert (ici nommée execute()) :

```
ConnectionToServer cts = new ConnectionToServer();
cts.execute((Void) null);
```

Le détail des opérations est donné dans la prochaine partie.

Class ConnectionToServer :

C'est donc la classe qui va effectuer le transfert des données entre la base de données et l'application android.

Cette tâche implique utilisation de la connexion, et pour que l'application puisse utiliser le Wifi il lui faut la permission nécessaire. Elle lui est donnée en ajoutant la ligne suivante au fichier automatiquement créé AndroidManifest.xml :

```
<uses-permission android:name="android.permission.INTERNET"></uses-
    permission>
```

Nous avons également appris qu'une connexion ne pouvait être gérée dans ce qui est appeléle "UI thread", le fil principal (celui de l'interface utilisateur). C'est à dire que Android refuse de prendre en charge de telles opérations qu'il considère plus risquées et nécessitant donc d'être exécutées sur un fil séparé (auquel cas s'il survient un problème toute l'application ne plantera pas) et demande à les traiter en tant que "AsyncTask", tâche asynchrone.

```
private class ConnectionToServer extends AsyncTask<Void, Void, Void> {

    // AsyncTask

}
```

Une telle classe déroule ses actions en quatre étapes :

1. onPreExecute () : cette méthode est appelée au tout début de l'exécution de la tâche et est généralement utilisée pour la mettre en place, en montrant par exemple une barre de progression dans l'interface utilisateur
2. doInBackground (Params...) : celle-ci est appelée juste après l'exécution de la précédente en « background »/arrière-plan (en opposition à ce qui est « visible »). On l'utilise pour des tâches relativement longues, elle prend les paramètres passés à AsyncTask et renvoie un résultat à la dernière étape. Cette étape peut se servir de la méthode publishProgress(Progress...) pour envoyer des unités de progression à l'étape suivante.
3. OnProgressUpdate (Progress...) : cette étape n'est appelée que si la précédente a elle-même appelée la méthode publishProgress(Progress...), elle se charge d'afficher dans l'UI thread les unités de progression de la méthode doInBackground(Params...).

4. OnPostExecute (Result) : finalement, cette étape s'exécute à la fin de doInBackground(Params...)

- La seule étape obligatoire est la second.

```
@Override
protected Void doInBackground(Void... params) {

     // connexions process

}
```

```
public void  onPostExecute(Void param) {

     Toast.makeText(UpdateWondervileActivity.this,
           "Connexion terminée, merci d'avoir participé!",
           Toast.LENGTH_SHORT).show();
```

Conclusion et perspectives

L'objectif de ce projet est de stocker les informations des donneurs du sang dans une base de donnée afin de les récupérer en faisant une recherche par groupe sanguin pays et ville et enfin d'envoyer un message d'alerte a tout donneur d'un click de bouton.

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

Pour la conception de notre application, nous avons utilisé 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 oeuvre 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 :

- Ajouter une boite d'envoie et de réception comme les applications des réseaux sociaux (facebook, twitter).
- Ajouter un questionnaire prés don.

Tables des figures

Références bibliographiques

[1] : http://www.phonandroid.com/toute-l-histoire-et-la-chronologie-d-android-dossier.html

[2]: Wikipedia, http://en.wikipedia.org/wiki/Android_version_history

[3]:http://www.experip.com/mobile/les-contraintes-liees-au-developpement-mobile/

[4]: http://openclassrooms.com/courses/creez-des-applications-pour-android/l-architecture-d-android

[5]: http://fr.wikipedia.org/wiki/Syst%C3%A8me_d'exploitation_mobile

[6]: http://www.zdnet.fr/actualites/chiffres-cles-les-os-pour-smartphones-39790245.htm

[7]: http://fr.wikipedia.org/wiki/Application_mobile

[8]: http://www.contenus-en-ligne.com/avantages-inconvenients-dune-application-mobile

[9]:http://fr.wikipedia.org/wiki/Eclipse_(projet)

[10]: http://developer.android.com/tools/help/adt.html

[11]: http://fr.wikipedia.org/wiki/Kit_de_d%C3%A9veloppement

[12]: http://localhost/phpMyAdmin/