Application de messagerie simple sur Android :

Rapport de projet de VAP RSM

Projet encadré par : **M. Hossam AFFIFI**

Elaboré par : **Khaoula MRABET & Nessrine TRABELSI**

Responsable de la VAP : **M. Patrice AUBRY**

AU : 2010/2011

Sommaire

Introduction

Le marché de la téléphonie portable connaît actuellement une véritable révolution, menée par Apple et son iPhone. . Apple a su mettre en avant son produit en ajoutant au téléphone de nouvelles fonctionnalités et en créant de nouveaux besoins.

Le marché des Smartphones connaît donc un véritable essor dans lequel les acteurs habituels (Windows et Symbian) essaient de s'engouffrer.

Google, ayant réalisé le potentiel de ce marché, a décidé de s'y introduire en rachetant une startup travaillant sur un système d'exploitation ouvert pour terminal mobile : Android.

Dans le cadre de notre projet de Voie d'Approfondissement Réseaux et Services Mobiles, nous étions menées à explorer ce nouveau système d'exploitation pour mobiles, Android, et de faire une application de messagerie simple.

Ainsi, nous articulons notre rapport autour de quatre chapitres : Le premier chapitre consiste en une étude de l'état de l'art d'Android. Le deuxième chapitre aura comme but de définir l'environnement de travail, hardware et software. La troisième partie portera sur la création d'un projet Android simple, intitulé HelloAndroid. Finalement, nous décrirons dans le dernier chapitre le fonctionnement d'une application de messagerie instantanée tout en mettant l'accent sur la nouvelle fonctionnalité implémentée.

Chapitre 1: État de l'art

Introduction

Nous présenterons dans ce chapitre une description du système d'exploitation Android, son historique, ses fonctionnalités et finalement son architecture.

1. Description

Android est un système d'exploitation open-source pour smartphones, PDA et autres terminaux mobiles, conçu par Android, une start-up rachetée par Google en juillet 2005. Il existe d'autres types d'appareils possédant ce système d'exploitation tels que les téléviseurs et les tablettes.

Afin de promouvoir ce nouveau système d'exploitation ouvert, Google a su fédérer autour de lui un consortium d'une trentaine d'entreprises : l'Open Handset Alliance (OHA) créée officiellement le 5 novembre 2007. Toutes ces entreprises interviennent, plus ou moins directement, dans le marché de la téléphonie mobile.

Le but de cette alliance est de mettre en place des normes ouvertes dans le domaine de la téléphonie mobile. Ce qui veut dire que les développeurs d'application Android pourront accéder aux fonctionnalités du cœur de téléphone via une API très fournie.

Android aura comme principaux concurrents Apple avec l'iPhone, Microsoft et son Windows Mobile et Nokia avec Symbian mais également des solutions libres telles que LIMO ou OpenMoko.

2. Historique d'Android

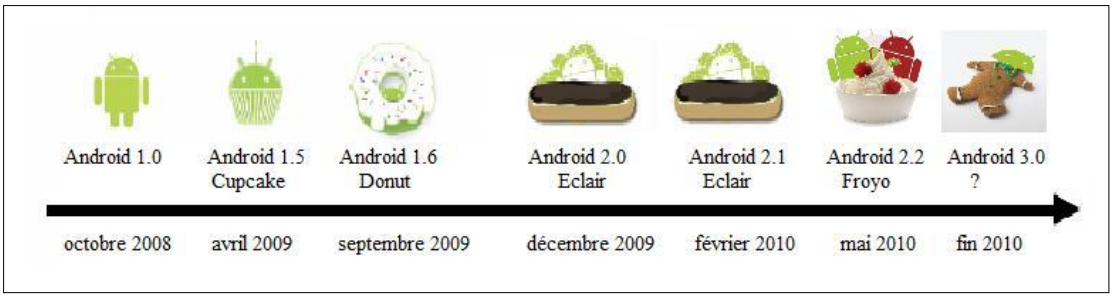


Figure : évolution des versions d'Android.

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.

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.

La version 1.5 **Cupcake** corrigea le manque d'API et rendit le système plus utilisable.

Depuis, **Android 1.6, 2.0** et **2.1** ont apporté d'importantes améliorations respectivement sur les fonctionnalités et sur l'interface graphique du système.

Android 2.2 Froyo a fortement mis l'accent sur la synergie avec Internet. L'envoi d'applications et de liens instantanés depuis un ordinateur est désormais possible. Aussi, Google annonce-t-elle que le navigateur chrome intégré à Android 2.2 est le navigateur mobile le plus rapide au monde grâce à l'intégration du moteur JavaScript V8.

Android 3.0 Honeycomb est spécialement étudié pour les tablettes tactiles. Les premiers modèles devraient être annoncés au CES 2011.
On y apprend quelques nouveautés comme la prise en charge de la vidéo-conférence via Gtalk, la nouvelle interface Gmail ou encore le lecteur de livre électronique Google.
La refonte graphique de l'interface utilisateur est assez réussie, plus d'informations devraient suivre dont surement des éclaircissements sur l'intégration ou non de l'interface de cette version d'Android sur les futurs smartphones.

Android 4.0 devrait arriver très vite (mi 2011) pour rajouter encore plus de fonctionnalités aux terminaux. Pour le développement, ces nouvelles versions d'Android devraient proposer de nouveaux composants permettant de réaliser des applications avec une ergonomie plus adaptée aux tablettes tactiles.

Android 3.0 et **Android 4.0** devraient apporter plus d'outils aux constructeurs leur permettant de proposer des tablettes tactiles, qui seront capables de rivaliser (surtout au niveau de l'ergonomie) avec **Ipad**.

3. 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 Calendar ou encore la cartographie Google Maps.

Voici quelques fonctionnalités proposées par Android classées par version :

✓ Android version 1.5 (Cupcake)

- Enregistrement et lecture des vidéos.
- Mise en ligne directe des vidéos sur YouTube.
- Mise en ligne directe des photos Picasa.
- Prise en charge du Bluetooth A2DP.
- Dossiers dynamiques et widgets pour le home.
- Copier/coller étendu aux pages web.
- Nouvelle version du clavier virtuel.

✓ Android version 1.6 (Donut)

- L'application Galerie permet d'effacer plusieurs photos à la fois.
- Amélioration de l'Android Market.
- Amélioration de la vitesse de la recherche vocale et intégration étendue à plus d'applications natives.
- Prise en charge sur une seule application de la prise de photo et de l'enregistrement vidéo.
- Possibilité de rechercher simultanément dans les favoris, les historiques, les contacts et sur Google depuis le home via le widget recherche.
- Moteur Text-to-speech.
- Prise en charge de plusieurs résolutions d'écran.

✓ Android version 2.0/2.1 (Éclair)

- Interface utilisateur revue (lock screen et lanceur d'application).

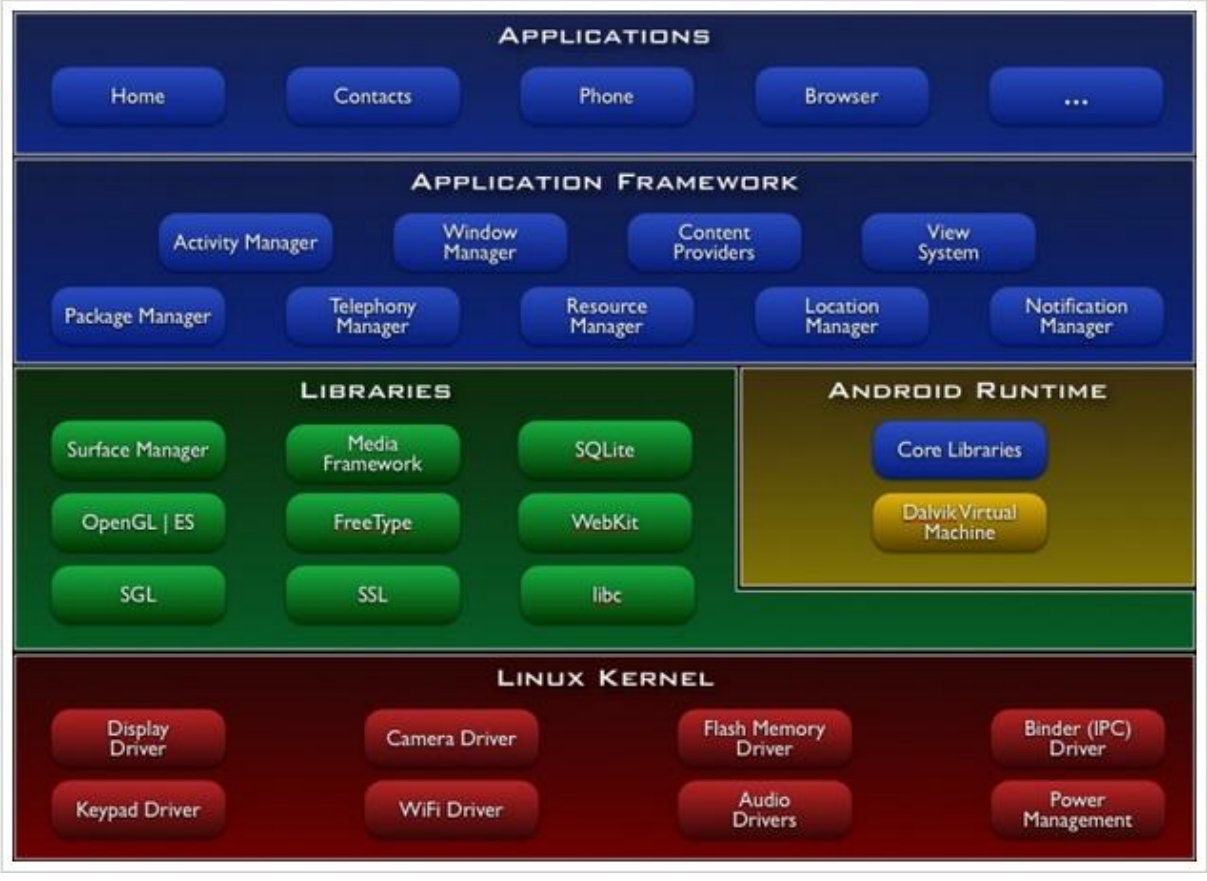
- Fonds d'écran animés.
- New browser interface avec prise en charge du HTML5.
- Prise en charge du protocole Microsoft Exchange.
- New contact lists.
- Prise en charge du Bluetooth 2.1.
- Amélioration du clavier virtuel.
- Prise en charge en natif du flash et du zoom numérique pour des appareils photos.
- Amélioration du ratio blanc/noir sur les fonds.
- Gestion multi-comptes Gmail et ajout de la synchronisation avec Facebook.

✓ Android 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.

4. Architecture Android

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



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.

4.1) Applications

Android est fourni avec un ensemble d'applications dont un client email, une application SMS, un calendrier, un service de cartographie, un navigateur… toutes écrites en JAVA.

4.2) Framework de développement

En fournissant une plateforme de développement ouverte, Android offre aux développeurs la possibilité de créer des applications extrêmement riches et innovantes. Les développeurs sont libres de profiter du matériel périphérique et informations sur la localisation d'accès, exécuter des services d'arrière-plan, définir des alarmes, ajouter des notifications à la barre d'état, etc.

Les développeurs ont un accès complet au même framework API utilisé par les applications de base. L'architecture d'application est conçue pour simplifier la réutilisation des composants; n'importe quelle application peut publier ses capacités et n'importe quelle autre application peut alors faire usage de ces capacités (soumis à des contraintes de sécurité appliquées par le framework). Ce même mécanisme permet aux composants d'être remplacés par l'utilisateur.

Toutes les applications sous-jacentes forment un ensemble de services et de systèmes, y compris:

- Un jeu extensible de vues qui peuvent être utilisées pour construire une application.
- Des fournisseurs de contenu qui permettent aux applications d'accéder aux données d'autres applications (telles que les Contacts), ou de partager leurs propres données
- Un gestionnaire de ressources.

- Un gestionnaire de notification qui permet à toutes les demandes d'afficher des alertes personnalisées dans la barre d'état.
- Un gestionnaire d'activité qui gère le cycle de vie des applications et propose une navigation commune.

4.3) Bibliothèques

Android dispose d'un ensemble de librairies C / C++ utilisées par les différents composants du système Android. Elles sont offertes aux développeurs à travers le framework Android. En voici quelques unes:

Système de bibliothèque C – une mise en œuvre dérivée de BSD de la bibliothèque C standard du système (libc), destinés aux systèmes embarqués basés sur Linux.

Comme cela a été dit précédemment, Android ne supporte pas la glibc, donc les ingénieurs d'Android ont développé une librairie C (libc) nommé Bionic libc . Elle est optimisée pour les appareils mobiles et a été développée spécialement pour Android.

Les ingénieurs d'Android ont décidé de développer une libc propre à la plateforme Android car ils avaient besoin d'une libc légère (la libc sera chargée dans chaque processus) et rapide (les appareils mobiles ne disposent pas de CPU puissant).

La Bionic libc a été écrit pour supporter les CPU ARM, bien que le support x86 soit présent. Il n'y pas de support pour les autres architectures CPU telles que PowerPC ou MIPS. Néanmoins, pour le marché des appareils mobiles, seulement l'architecture ARM est importante.

Cette libc est sous licence BSD. Elle reprend une grande partie du code des glibc issue d'OpenBSD, FreeBSD et NetBSD.

Ces caractéristiques importantes :

Elle pèse environ 200Ko, soit la moitié de la glibc

L'implémentation des pthreads (POSIX thread) a été complètement réécrite pour supporter les threads de la machine virtuelle Dalvik. De ce fait, la Bionic libc ne supporte pas les threads POSIX

Les exceptions C++ et les "wide char" ne sont pas supportés

Médiathèques – basée sur PacketVideo de OpenCore; les librairies permettant la lecture et l'enregistrement audio et vidéo, ainsi que la gestion des fichiers image, y compris MPEG4, H.264, MP3, AAC, AMR, JPG et PNG.

Le schéma ci-dessous décrit tous les éléments de l'architecture de ces médiathèques:

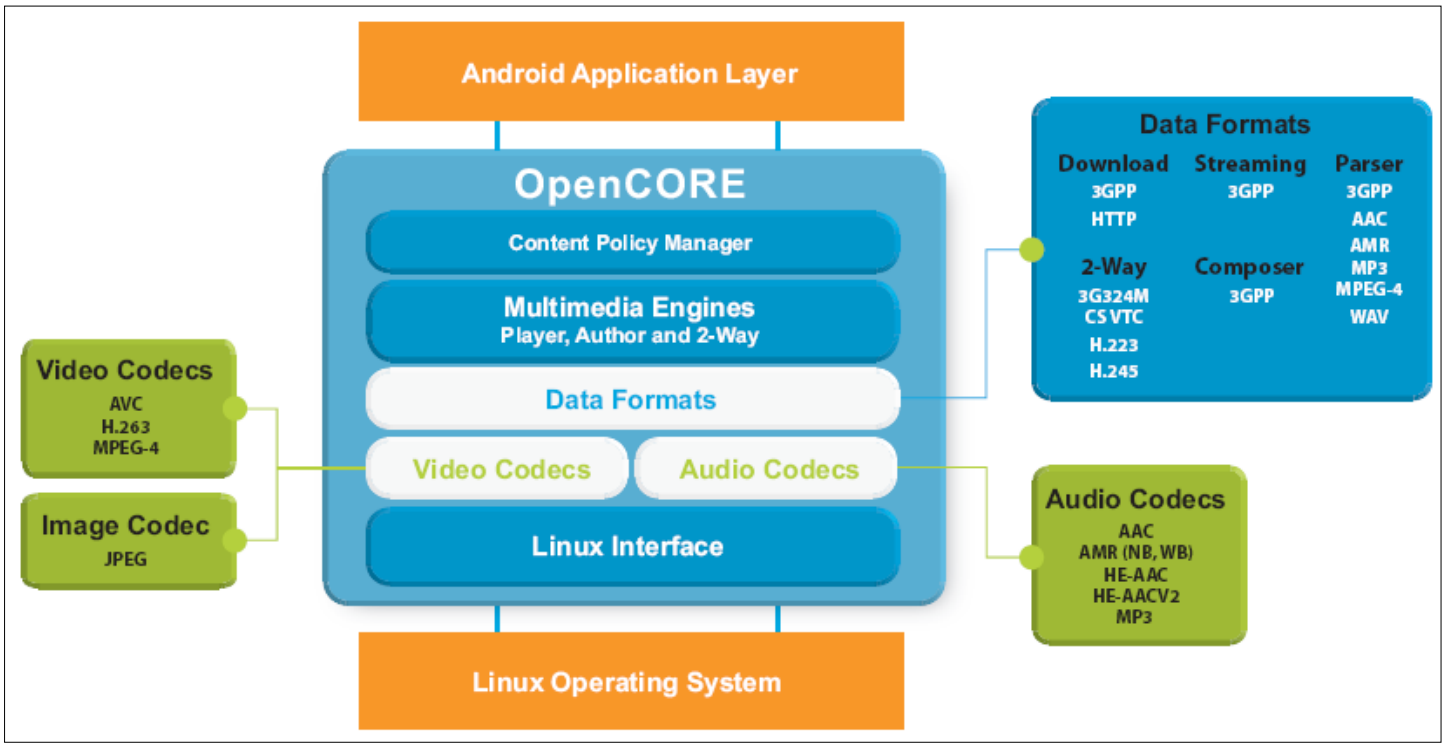


Figure : architecture de ces médiathèques

- Surface Manager – gère l'accès au sous-système d'affichage et de façon transparente.
- LibWebCore – Le navigateur web présent dans Android est basé sur le moteur de rendu sous licence BSD WebKit.
- WebKit est moteur de rendu, qui fournit une "fondation" sur laquelle on peut développer un navigateur web. Il a été originellement dérivé par Apple du moteur de rendu KHTML pour être utilisé par le navigateur web Safari et maintenant il est développé par KDE project, Apple, Nokia, Google et d'autres. WebKit est composé de deux librairies : WebCore et JavascriptCore qui sont disponibles sous licence GPL.

  WebKit supporte le CSS, Javascript, DOM, AJAX. La dernière version a obtenu 100% au test Acid 3. La version de WebKit présent dans Android à été légèrement modifiée pour s'adapter aux appareils mobiles. Ainsi, le moteur de rendu basé sur WebKit présent dans Android supporte l'affichage sur une colonne.

- SGL – le moteur graphique 2D.
- Bibliothèques 3D – une implémentation basée sur OpenGL ES 1.0 API; les bibliothèques utilisent l'accélération 3D matérielle (si disponible).
- FreeType – bitmap et vectoriel de rendu de police.

- SQLite – un moteur de base de données relationnelles puissant et léger, disponible pour toutes les applications.

4.4) Android Runtime

Android inclut un ensemble de librairies de base offrant la plupart des fonctionnalités disponibles dans les bibliothèques de base du langage de programmation Java.

Chaque application Android s'exécute dans son propre processus, avec sa propre instance de la machine virtuelle Dalvik. Dalvik a été écrit pour que le dispositif puisse faire tourner plusieurs machines virtuelles de manière efficace. La machine virtuelle Dalvik exécute des fichiers dans l'exécutable Dalvik (. DEX), un format optimisé pour ne pas encombrer la mémoire. La machine virtuelle est la base de registres et fonctionne grâce aux classes compilées par un compilateur Java et transformées dans le format DEX.

La machine virtuelle Dalvik s'appuie sur le noyau Linux pour les fonctionnalités de base telles que le filetage et la gestion de la mémoire de bas niveau.

4.5) Linux Kernel

Android est basé sur un kernel linux 2.6 mais ce n'est pas linux. Il ne possède pas de système de fenêtrage natif (X window system). La glibc n'étant pas supportée, Android utilise une libc customisée appelée Bionic libc.

Enfin, Android utilise un kernel avec différents patches pour la gestion de l'alimentation, le partage mémoire, etc. permettant une meilleure gestion de ces caractéristiques pour les appareils mobiles.

Android n'est pas linux mais il est basé sur un kernel linux. Pourquoi sur un kernel linux ?

Le kernel linux a un système de gestion mémoire et de processus reconnu pour sa stabilité et ses performances.

Le model de sécurité utilisé par linux, basé sur un système de permission, est connu pour être robuste et performant. Il n'a pas changé depuis les années 70

- Le kernel linux fournit un système de driver permettant une abstraction avec le matériel. Il permet également le partage de librairies entre différents processus, le chargement et le déchargement de modules à chaud.

- le kernel linux est entièrement open source et il y a une communauté de développeurs qui l'améliorèrent et rajoutent des drivers.

C'est pour les points cités ci-dessus que l'équipe en charge du noyau a décidé d'utiliser un kernel linux.

Conclusion

Dans ce chapitre, nous avons fait une étude de l'art d'Android tout en présentant un bref historique, les fonctionnalités que nous pouvons trouver sur ce système d'exploitation et l'architecture d'Android, à savoir les principaux composants du système.

Chapitre 2: Outils de réalisation d'un projet Android

Introduction

Dans ce chapitre, nous présenterons l'environnement de travail qui inclut les outils de développement (logiciels et technologies exploités) ainsi que l'outil matériel.

1. Outils logiciels : Environnement technique

1.1) Installation d'Android SDK sous Windows – Déploiement

1 .1.1) Téléchargement des outils

Nous avons commencé par le téléchargement des outils nécessaires :

- ✓ Android SDK : téléchargé depuis le site officiel Android SDK.
- ✓ Eclipse Version: 3.3.1.1 (Europa) ou 3.5 (Galileo).
- ✓ JDK 5 or JDK 6.

1 .1.2) Installation des outils

- Nous avons installé SDK 7.
- Nous avons dé zippé le contenu du fichier Android SDK sous le chemin C:\android-sdk\.
- Nous avons dé zippé la version Eclipse sous le chemin C:\eclipse\.

1 .1.3) Téléchargement des différents composants d'Android SDK

- Sous C:\android-sdk\ nous lançons SDK Setup.exe.
- Sous « Available Packages », nous trouvons la liste des packages et Archives du Android SDK.
- Nous choisissons les différents packages et nous validons.

Une fois ceci terminé, nous nous retrouvons avec les différentes versions d'Android installées sur le système comme le montre la capture suivante :

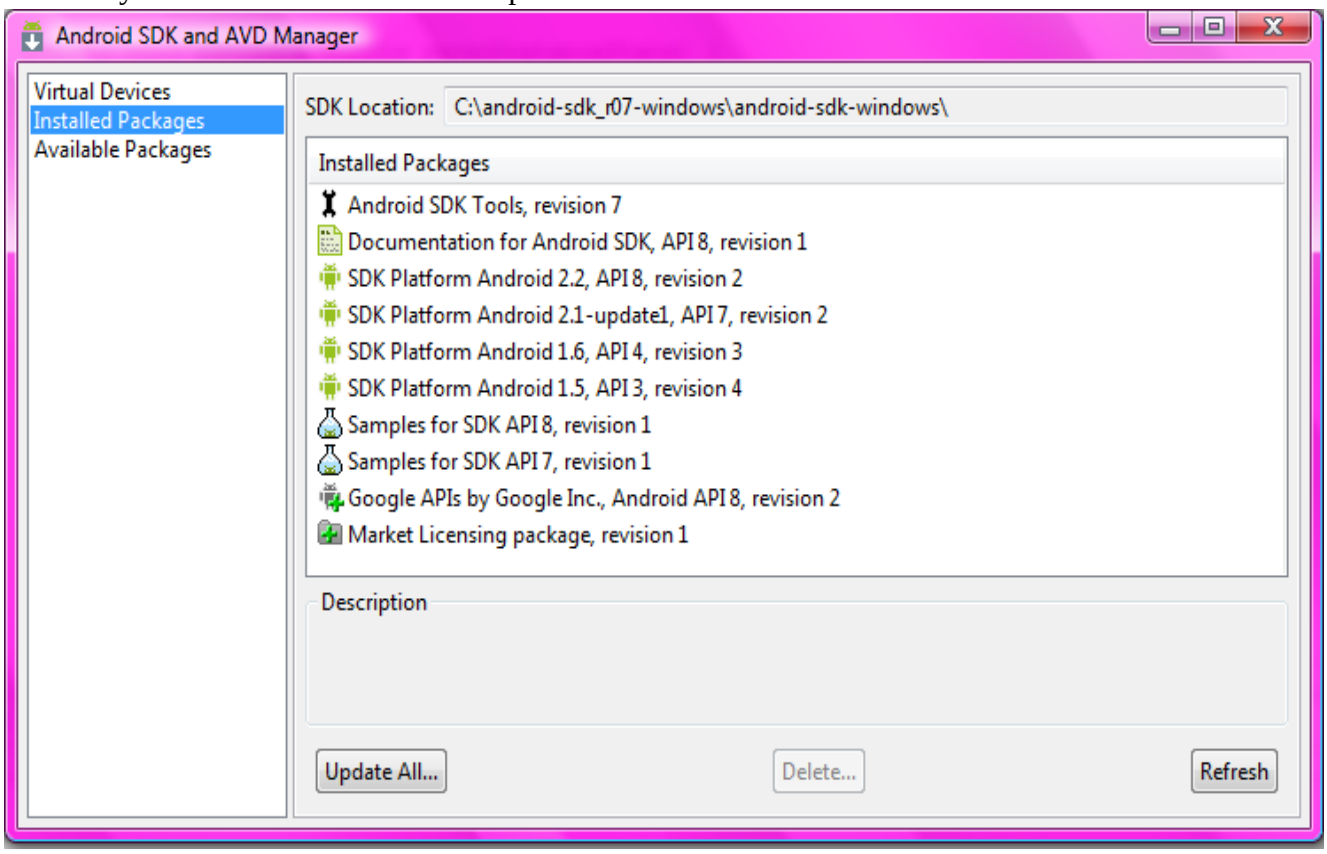


Figure : Android SDK AVD Manager

- Dans le menu Virtual Devices, nous cliquons sur le bouton « new », ce qui nous permettra de créer notre toute première machine virtuelle contenant l'OS Android pour permettre le développement d'application mobile.
- Une taille de 128 MB de données pour la SD Card (simulation de la carte SD) suffira amplement pour le développement que nous aurons à faire, une fois les champs complétés, comme le montre la capture suivante, on clique sur « Create AVD ».

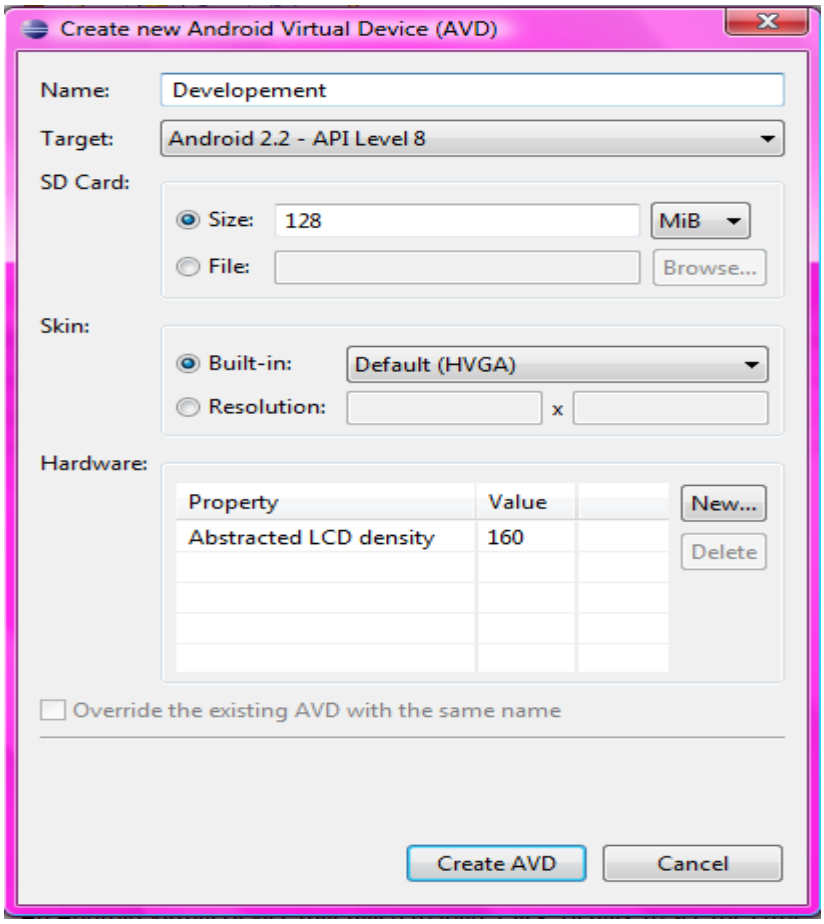


Figure : Création d'une VM

1 .1.4) Paramétrage d'Eclipse et installation du plugin ADT

- Nous lançons Eclipse puis dans le menu « Help » >> « Install new software », une fenêtre s'ouvre alors et nous configurons les dépôts Google Android pour avoir le plugin AVD intégré à notre IDE.
- Nous cliquons sur le bouton « Add » en haut à droite de la fenêtre.
- Nous renseignons les champs suivants :
    - Name : Android (ou autre).
    - Location : http://dl-ssl.google.com/android/eclipse/.

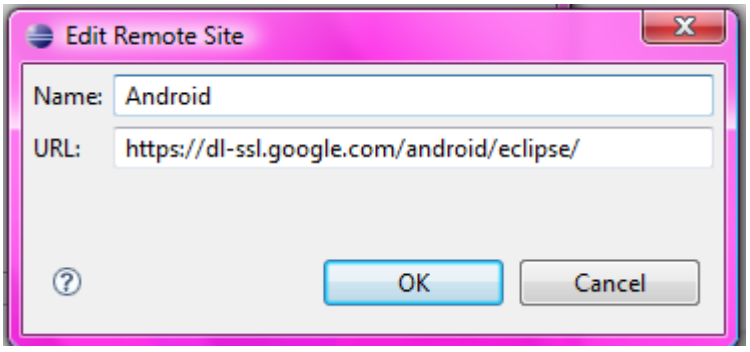


Figure : URL Plugin

- Après rafraichissement, Eclipse affiche les plugins disponibles. Nous sélectionnons le « Android DDMS » et le « Android development Tools ».
- Nous validons les étapes suivantes et nous redémarrons Eclipse.
- Maintenant que Eclipse reconnait les plugins, sous Window-> Preferences->Android nous indiquons le chemin d'Android SDK : C:\android-sdk\ dans notre cas :

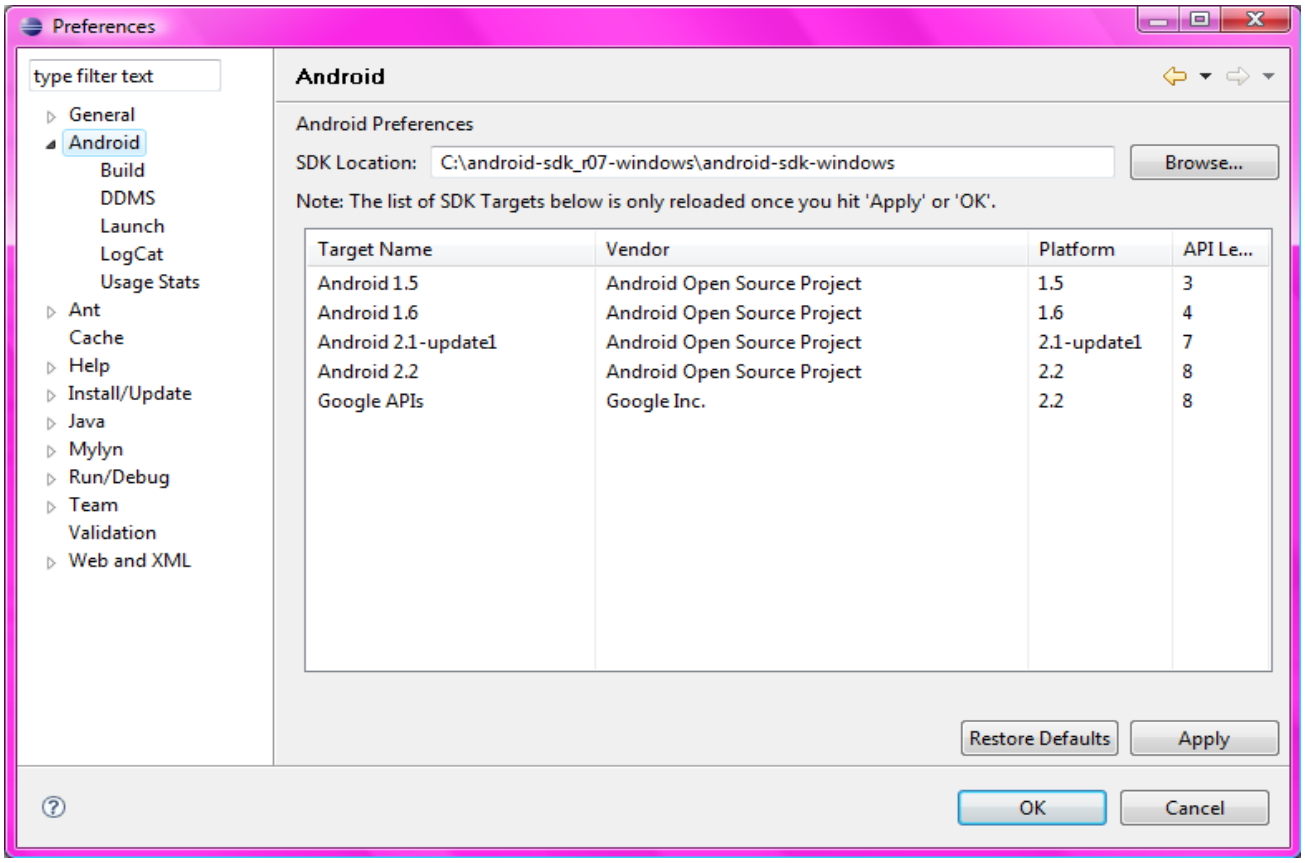


Figure : liste des targets

Nous voilà avec une configuration d'Android et d'Eclipse opérationnelle.

1.2) Installation des applications sur téléphone

1.2.1) Installation du pilote USB

Pour pouvoir utiliser un vrai téléphone android depuis l'éditeur de code éclipse, nous avons besoin d'installer un driver USB, puisqu'en branchant l'USB, windows, par défaut, ne connait pas le type de matériel.

En général, le pilote à installer se trouve dans le dossier du sdk installé et plus exactement dans un dossier qui s'appelle google-usb-driver. Selon le processeur que nous avons sur le téléphone, nous choisissons le bon pilote.

Si le pilote correspondant au téléphone n'existe pas dans le dossier indiqué, nous devrons le télécharger depuis Internet.

Nous branchons le téléphone sur le port USB et l'assistant d'ajout de nouveau matériel détecté apparait. Nous spécifions alors le chemin du pilote et procédons à son installation.

Une fois l'installation du driver terminée, nous pouvons alors commencer à utiliser le téléphone depuis éclipse.

Dans éclipse, nous ouvrons la perspective DDMS et dans l'onglet device à gauche, nous pourrons voir tous les émulateurs existants ainsi que le téléphone branché.

Nous pouvons ainsi utiliser le téléphone avec eclipse.

1.2.2) Paramétrage du téléphone

Nous allons configurer le téléphone pour qu'il puisse accepter le débogage et l'installation d'application de l'environnement de développement.

Les commandes à exécuter sont alors les suivantes:

On clique sur menu puis on choisit paramètres (ou bien settings)

On choisit applications

On coche sources inconnues

Cliquer sur OK quand le warning s'affiche (puisque c'est bien une application de confiance)

Passer dans Développement

Activer Débogage USB, Rester activé et Positions fictives

2. Outil matériel

La programmation a été effectuée sur deux ordinateurs dotés des capacités suivantes:

- Marque : HP
- Microprocesseur : Intel core 2 duo.
- Mémoire vive 3 Go et 2 Go.
- Disque dur 250 Go et 160 Go.

Conclusion

Nous avons présenté dans ce chapitre l'environnement software et hardware utilisé pour la programmation Android.

Chapitre 3 : Création d'un Projet Android

Introduction

Dans ce chapitre, nous allons décrire les étapes de création d'un simple projet Android, à savoir HelloAndroid et nous expliquerons la manière de le tester sur un émulateur Android.

1. Création d'un AVD

Afin de tester notre application, nous allons utiliser l'émulateur Android. Il faudra donc créer un Android Virtual Device (AVD). Un AVD décrit les paramètres systèmes et les composants de notre émulateur.

Pour créer un AVD:

1. Nous lançons Eclipse
2. Nous allons sous « Window > Android SDK and AVD Manager »
3. Nous sélectionnons « Virtual Device » dans le panneau à gauche
4. Nous cliquons sur « New ». La boite de dialogue « Create New AVD » apparaîtra
5. Nous tapons le nom de notre AVD, « hello_avd » par exemple
6. Nous choisissons la cible (the target). La cible est la version de la plateforme Android SDK que nous avons téléchargé.
7. Nous ignorons les autres champs pour le moment et nous cliquons sur « Create AVD »

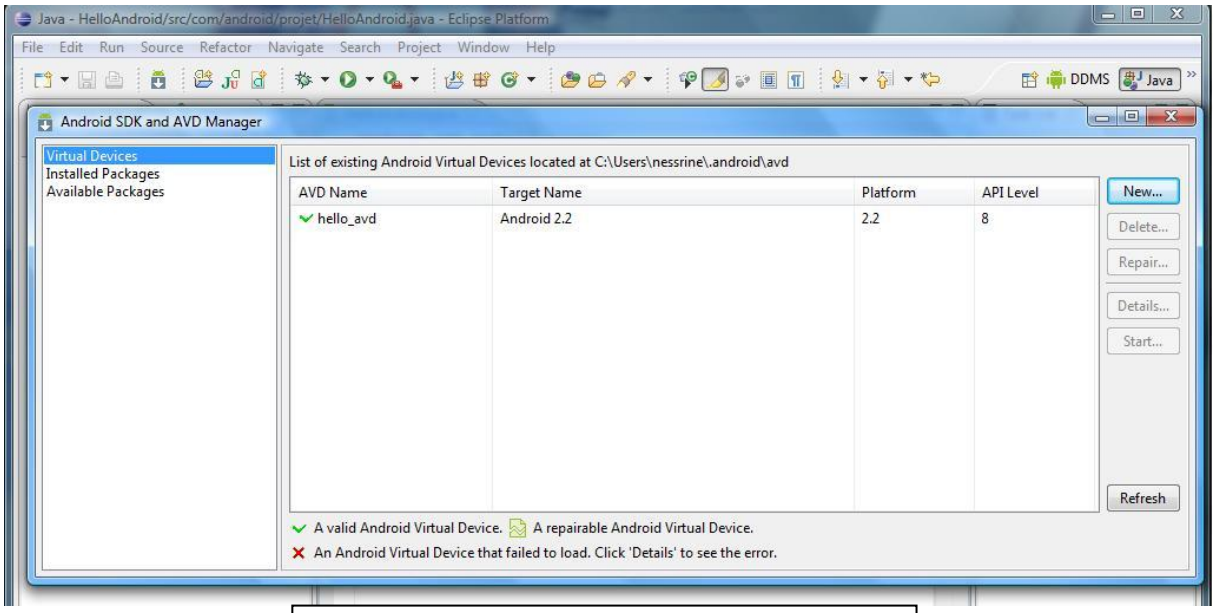


Figure : liste des AVD crées.

2. Création d'un projet Android

Après avoir créé un émulateur Android, nous passons à la création du projet sous Eclipse.

- Nous lançons Eclipse et nous allons sous **File -> New->Project** et nous sélectionnons « **Android Project** ».

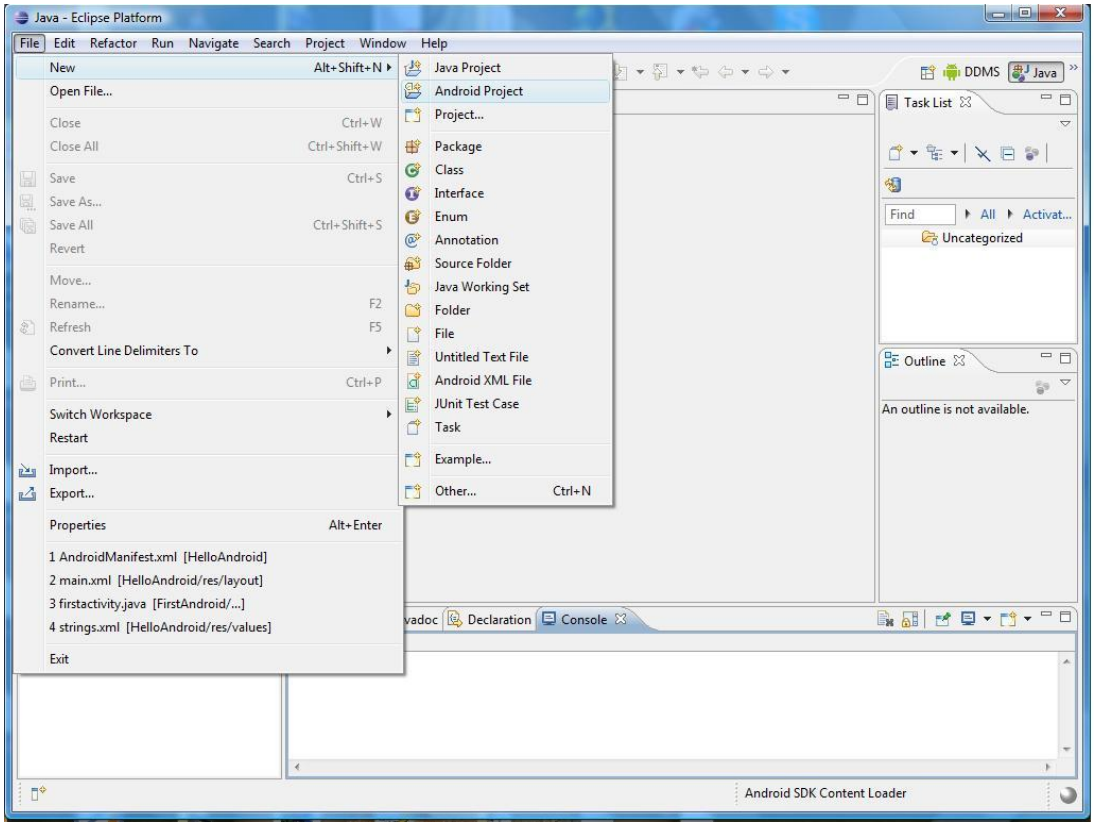


Figure : Création d'un nouveau projet

- Nous renseignons les détails à propos de notre projet comme suit :

  - ***Project name***: HelloAndroid.
  - ***Build Target***: Android 2.2.
  - ***Application name:*** Hello, Android.
  - ***Package name***: com.android.projet.
  - ***Create Activity***: HelloAndroid.

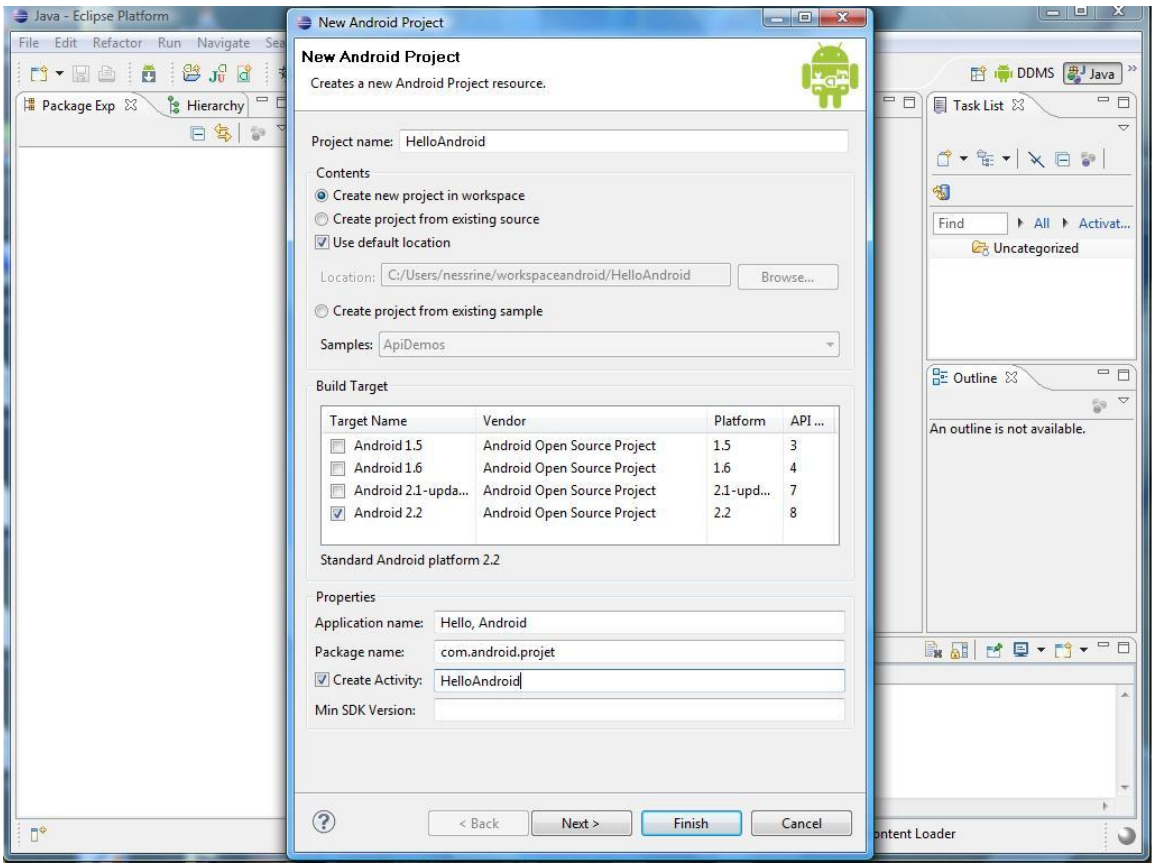


Figure : Paramètres du projet HelloAndroid

- Nous cliquons sur **Finish.**

2.1) Explication des paramètres du projet

- *Project name* : C'est le nom du projet Eclipse. Tous les fichiers seront créés sous un dossier portant le même nom.
- *Application Name* : C'est le nom de l'application tel qu'il va apparaître sur le smartphone Android.
- *Package Name* : C'est le package namespace (suivant les mêmes règles de programmation Java) qui regroupera tout le code source qu'on va écrire. D'une manière générale, le nom du package doit être unique. Dans notre exemple, on a utilisé com.android.projet.
- *Create Activity* : C'est le nom du stub class qui va être générée par le plugin. Elle va être une sous-classe de la classe Activity d'Android.

2.2) Explication du code

Notre projet est maintenant prêt. Examinons le code en navigant dans le Package Explorer à gauche. Nous ouvrons le fichier HelloAndroid.java situé sous HelloAndroid->src->com.android.projet qui devra ressembler à ça :

```
package com.android.projet;


import android.app.Activity;
import android.os.Bundle;


public class HelloAndroid extends Activity {
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
   }
}
```

Nous notons que cette classe est basée sur la classe Activity que nous avons mentionnée tout à l'heure. Une Activity est une entité de l'application permettant d'exécuter des actions. Une application peut avoir plusieurs Activities, mais l'utilisateur interagit avec elles une à une.

La méthode ***onCreate()*** sera appelée par le système Android lors du démarrage de l'application. C'est donc l'endroit idéal pour faire toutes les initialisations et préparer l'interface utilisateur. Cependant, il n'est pas obligatoire d'avoir une interface utilisateur pour chaque Activity.

3. Codage et exécution de HelloAndroid

Modifions le code comme suit :

```
package com.android.projet;


import android.app.Activity;

import android.os.Bundle;

import android.widget.TextView;


public class HelloAndroid extends Activity {

  /** Called when the activity is first created. */

  @Override

  public void onCreate(Bundle savedInstanceState) {

     super.onCreate(savedInstanceState);

     TextView tv = new TextView(this);

     tv.setText("Hello, Android");

     setContentView(tv);

  }

}
```

Une Interface Utilisateur Android est composée d'une hiérarchie d'objets appelés Views (Vues). Une View est un objet à dessiner, utilisé comme un élément de l'interface utilisateur. Cela peut être un bouton, une image ou tout simplement du texte comme dans notre cas. Chacun de ces objets est une sous-classe de la classe View. Et la sous-classe qui prend en charge le texte est TextView.

Nous venons de créer un TextView avec le constructeur de classe qui prend comme paramètre une instance Context Android. Un Context fournit des services comme l'accès aux ressources, l'obtention d'un accès à la base de données, etc… La classe Activity hérite du Context et comme

la classe HelloAndroid est une sous classe Activity, elle est donc un Context. C'est pourquoi nous pouvons passer un « ***this*** » comme référence au TextView.

Nous avons défini après le contenu texte avec setText().

Finalement, nous avons passé le TextView à setContentView() pour l'afficher comme un contenu de l'interface utilisateur de l'Activity. Bien entendu, si nous ne faisons pas appel à cette méthode, le système affichera un écran vide.

C'est tout. Exécutons notre application maintenant.

Le plugin Eclipse facilite l'exécution de nos applications:

- Nous sélectionnons Run -> Run.
- Nous sélectionnons « Android Application ».

Le plugin crée automatiquement une configuration d'exécution pour notre application et lance l'émulateur Android. Ca risque d'être vraiment lent. Après le démarrage du système, le plugin installe la nouvelle application et exécute notre Activity par défaut. Et voici le résultat:

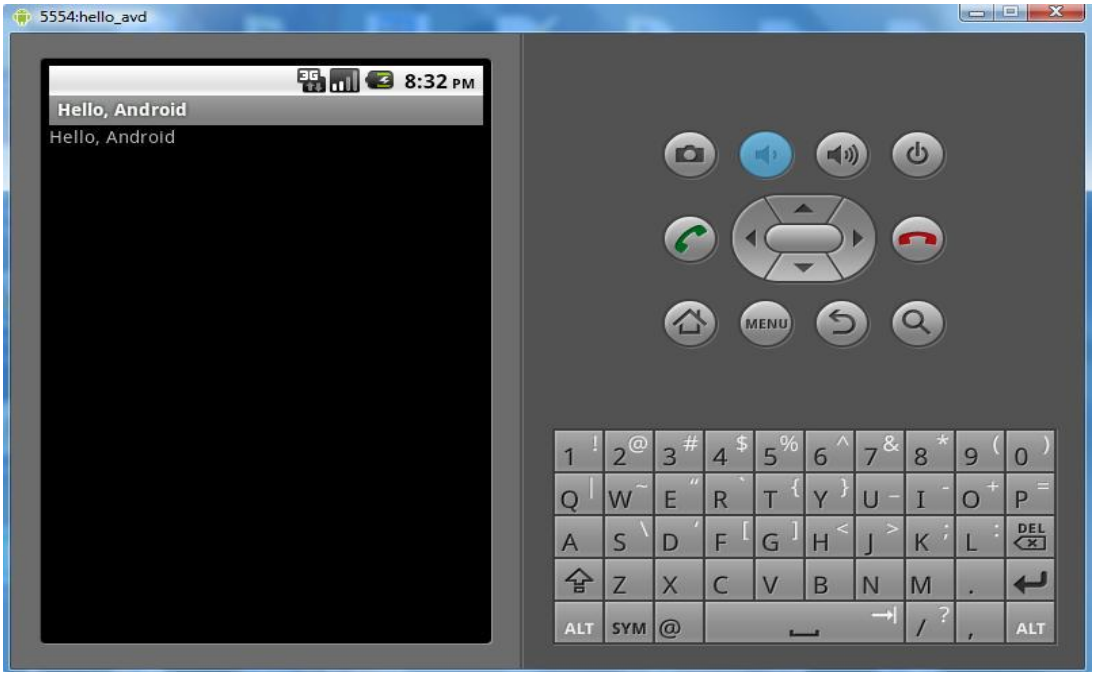


Figure : Résultat de l'exécution de l'application

Conclusion

Après avoir présenté les étapes de création du projet HelloAndroid, nous passerons dans la suite à l'explication d'une application de messagerie simple.

Chapitre 4 : Application Android : Messagerie Instantanée

Introduction

Dans ce chapitre, nous décrirons le fonctionnement d'une application de messagerie instantanée que nous avons prise du site http://code.google.com/p/simple-android-instant-messaging-application/ , expliquerons en détails les étapes de sa mise en marche et décrirons la fonctionnalité que nous avons ajoutée à l'application.

1. Principe du fonctionnement

Il s'agit d'une simple application de messagerie instantanée fonctionnant sur Android. Cette application permet l'enregistrement d'un utilisateur et assure son authentification. La recherche d'un nouvel ami peut être effectuée par le biais du nom d'utilisateur. On peut soit accepter soit refuser les invitations des autres usagers. Cette application permet également à chaque utilisateur d'échanger des messages instantanés avec les amis qui apparaissent dans la liste. Elle lance aussi un service d'arrière-plan afin de faire passer des messages même lorsque l'application est fermée. Une zone de notification d'utilisation s'affiche lors de la réception d'un nouveau message.

2. Etapes de la mise en marche de l'application

- ✓ Téléchargement et installation de WampServer
- ✓ Copier tous les fichiers existant sous le dossier Server vers un dossier dans le répertoire du serveur web (C:\wamp\www), ce dossier sera nommé par exemple android_im. Nous pouvons accéder à ce dossier par http://192.168.7.5/android_im/ (192.168.7.5 est l'adresse IP de l'ordinateur qui exécute Wampserver et Mysql).
- ✓ Ouvrir le fichier index.php et entrer les paramètres de connectivité de base de données telles que le host, le nom d'utilisateur, le mot de passe et le nom de la base de donnée.
- ✓ Ecrire error_reporting (0) en haut de l'indice.Php
- ✓ Le début du fichier index.php devient comme suit :

```
error_reporting(0);

require_once("mysql.class.php");

$dbHost = "localhost";

$dbUsername ="android-im_user";

$dbPassword = "root";

$dbName = "android-im";
```

✓ Créer la base de données android_im :

Créer une nouvelle base " android_im " dans laquelle nous travaillerons dans toute la suite en utilisant le formulaire dans la page http://localhost/phpmyadmin.

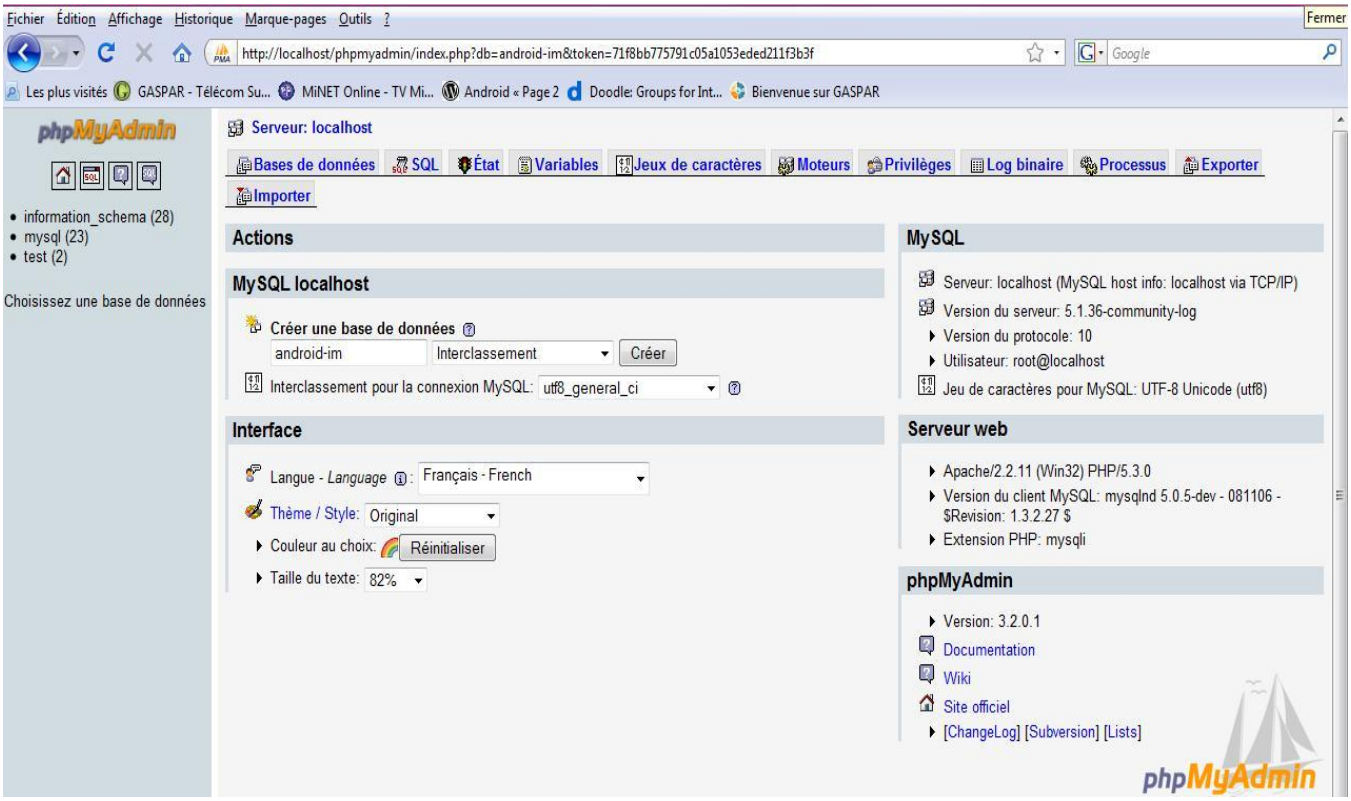


Figure : Création de la base de données.

✓ Créer les tables de base de données mysql à l'aide du script mysql suivant :

```
CREATE TABLE  `friends` (
  `Id` int(10) unsigned NOT NULL auto_increment,
  `providerId` int(10) unsigned NOT NULL default '0',
  `requestId` int(10) unsigned NOT NULL default '0',
  `status` binary(1) NOT NULL default '0',
  PRIMARY KEY  (`Id`),
  UNIQUE KEY `Index_3` (`providerId`,`requestId`),
  KEY `Index_2` (`providerId`,`requestId`,`status`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='providerId is the Id of the
users who wish to be friend with';

CREATE TABLE  `users` (
  `Id` int(10) unsigned NOT NULL auto_increment,
  `username` varchar(45) NOT NULL default '',
  `password` varchar(32) NOT NULL default '',
  `email` varchar(45) NOT NULL default '',
  `date` datetime NOT NULL default '0000-00-00 00:00:00',
  `status` tinyint(3) unsigned NOT NULL default '0',
  `authenticationTime` datetime NOT NULL default '0000-00-00 00:00:00',
  `userKey` varchar(32) NOT NULL default '',
  `IP` varchar(45) NOT NULL default '',
  `port` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`Id`),
  UNIQUE KEY `Index_2` (`username`),
  KEY `Index_3` (`authenticationTime`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Cette étape est effectuée en écrivant le script ci-dessus dans la zone approprié et qui est montrée dans la figure suivante et puis en cliquant sur exécuter :

Figure : Création des tables.

La figure ci-dessous montre les tables friends et users crées :

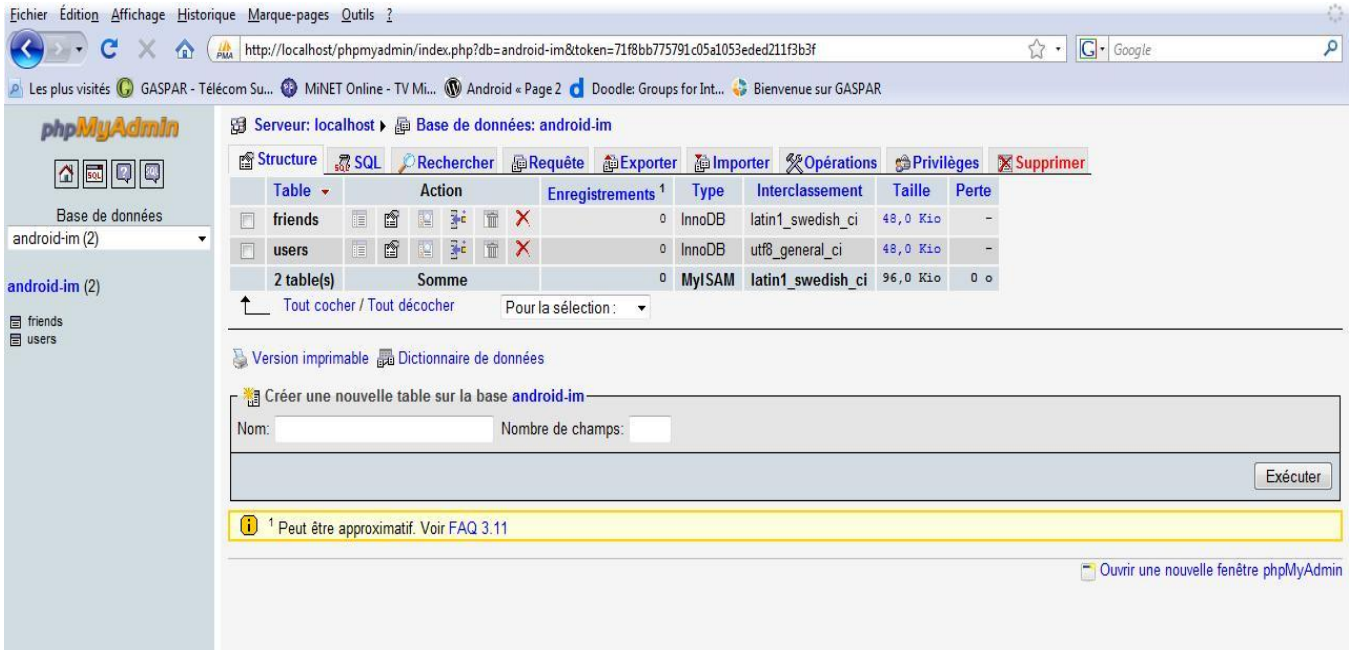


Figure : tables users et friends créés.

3. Fonctionnement détaillé de l'application

Nous présenterons en premier lieu l'application en sa première version telle qu'elle existe sur Internet puis dans un second lieu nous décrirons la nouvelle fonctionnalité ajoutée.

3.1) Première Version

La première étape consiste à s'enregistrer. A travers une première vue de l'application, l'utilisateur est amené à entrer un login, un mot de passe (deux fois) et une adresse mail.

Par exemple dans notre cas, le login sera "nessrine", le mot de passe "nessrine" et l'adresse mail nessrine@nessrine.com.

Notons que l'application affichera des messages d'erreur dans les cas suivants:

- Le login ou le mot de passe sont très courts (moins de cinq caractères).
- Le mot de passe n'est pas le même dans les deux emplacements adéquats.
- Le login existe déjà dans la base de données contenant tous les utilisateurs enregistrés.

Figure : Sign Up

Une fois nous appuyions sur "Sign up", et si tout fonctionne normalement, une nouvelle ligne dans la table "users" apparait contenant certaines informations de l'utilisateur.

L'utilisateur tape après son login et mot de passe pour accéder à l'application.

Figure : Login

En cliquant sur "Login", une requête est envoyée au serveur pour vérifier les données entrées par l'utilisateur. Dans le cas idéal, l'utilisateur est authentifié et une mise à jour du temps d'authentification est faite dans la base de données.

Aussi, l'application cherche-t-elle les informations des amis de "nessrine", dans notre cas, pour les afficher dans la vue suivante.

Figure : Liste des amis (1)

Nous lançons l'application sur un autre émulateur android et entrons les informations de khaoula

L'icône indiquant son statut changera en vert.

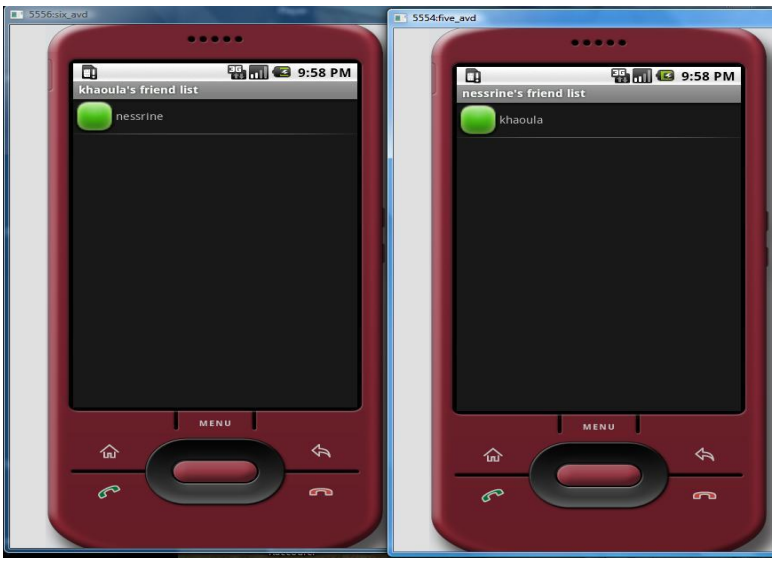


Figure : Liste des amis (2)

Pour pouvoir communiquer entre les deux émulateurs, nous devons faire un port-forwarding.

Le port correspondant à chaque utilisateur est indiqué dans la table "users" dans la base de données "android_im".

Ouvrons un exécuteur de commandes et tapons cmd.

Par la suite, nous devons accéder au dossier plateforme-tools contenu dans la sdk.

Puis nous tapons ces deux lignes de commandes :

```
adb -s emulator-5554 forward tcp:19179 tcp:19179
```

```
adb -s emulator-5556 forward tcp:25777 tcp:25777
```

Ainsi, les deux amies peuvent s'envoyer des messages instantanés.

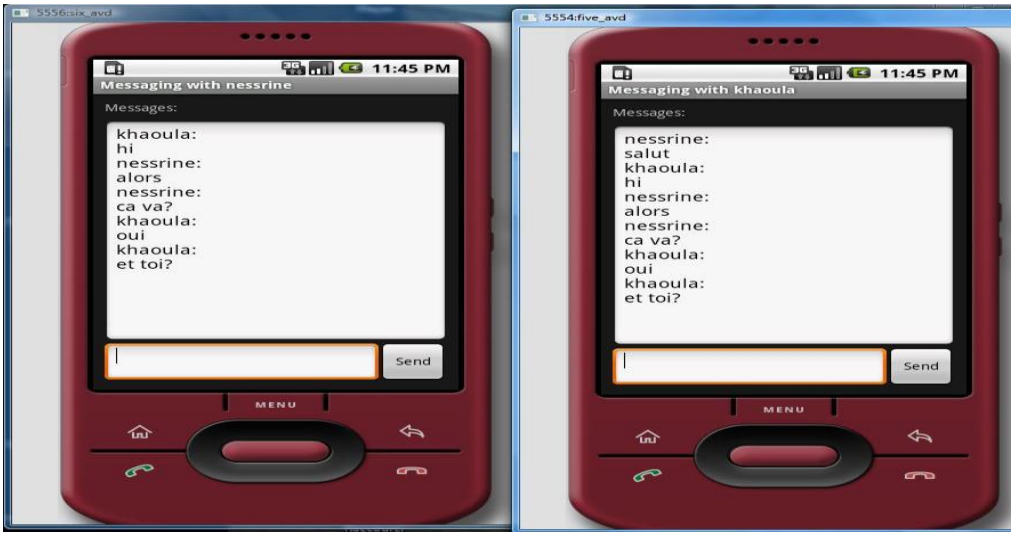


Figure : Envoi de messages

Une fonctionnalité essentielle dans l'application est l'envoi d'invitation d'amitié. La recherche d'amis s'effectue en entrant son login.

Quand un utilisateur reçoit une notification de demande d'ajout, il a la possibilité d'accepter ou de refuser.

Figure : Envoi demande d'ajout d'amis

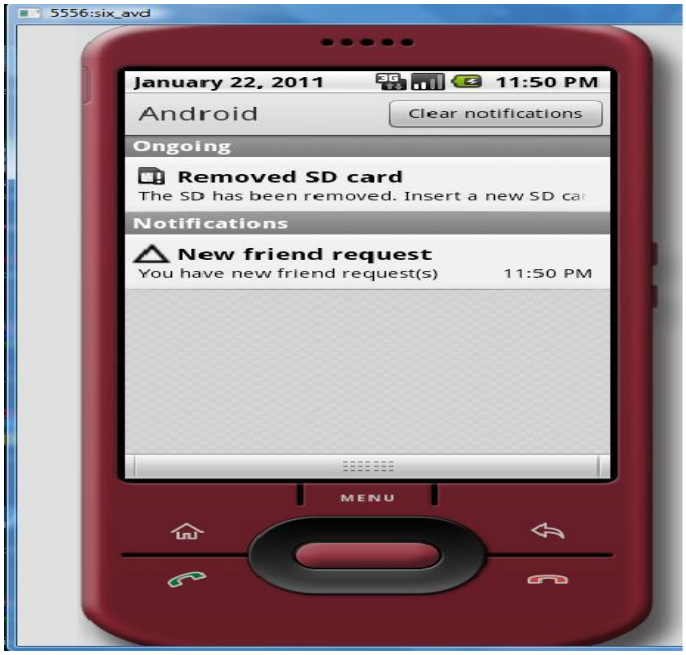


Figure : Réception demande d'ajout

Figure : Accepter/refuser les amis

Même en sortant de l'application, elle ne ferme pas et tourne en arrière plan. Dans le cas de réception d'un message, une notification apparait en haut de l'écran.

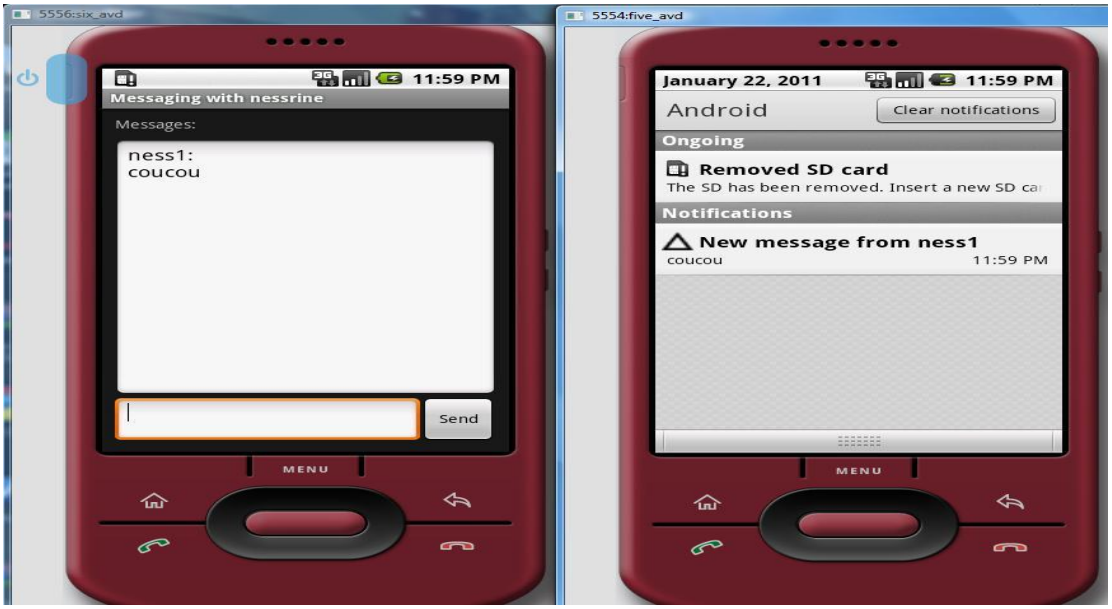


Figure : Application en arrière plan

3.2) Deuxième Version

L'application dans sa première version classait les amis selon le critère de disponibilité, donc un utilisateur est soit en ligne, soit hors ligne.

Nous sommes partis de cette constatation pour ajouter une nouvelle fonctionnalité permettant à l'utilisateur de choisir son statut, à savoir : en ligne, absent ou bien occupé.

Dans la vue listant les amis connectés ou pas, en cliquant sur menu, nous voyons apparaitre un item "set status".

Figure : Set Status

En cliquant sur cet item, un sous menu contenant trois items apparait. L'utilisateur a donc la possibilité de choisir son statut.

Figure : Choix du statut

Par exemple, choisissons "busy". L'icône deviendra alors rouge, indiquant à tous ses amis qu'il est désormais occupé.

Figure : Statut Occupé

4. Difficultés rencontrées

Au cours de la réalisation de ce projet nous avons rencontré plusieurs difficultés liées à l'émulateur d'éclipse pour Android. Parmi ces difficultés, on peut citer le fait qu'il ne supporte pas le Bluetooth, ni le wifi ce qui nous a obligé à tester nos codes d'application sur des téléphones Android réels.

Lors de l'installation de l'application de la messagerie instantanée sur le téléphone, l'application n'a pas fonctionné correctement parce qu'elle utilise les librairies de la version 1.5 et le téléphone utilise celle de la version 2.2. Vu cette incompatibilité l'application n'a pas pu s'exécuter de manière normale et montrer le résultat qu'on a eu sur l'émulateur.

Conclusion

Dans ce chapitre, nous avons décrit le fonctionnement de l'application de messagerie instantanée et mis l'accent sur la fonctionnalité ajoutée.

Conclusion

Ces dernières années, la téléphonie mobile a été sans doute le secteur le plus dynamique, le plus rentable et le plus innovant de toute l'Industrie des Télécommunications.
Le marché des Smartphones, essentiellement, connaît un véritable essor dans lequel les acteurs habituels essaient de s'engouffrer.

Google, ayant réalisé le potentiel de ce marché, a décidé de s'y introduire en sortant un nouveau système d'exploitation Android.

Dans le cadre de notre projet de Voie d'Approfondissement Réseaux et Services Mobiles, nous étions menées à explorer ce nouveau système d'exploitation pour mobiles et à faire une application de messagerie simple.

Ce rapport est donc composé de quatre parties. Dans la première, nous avons fait une étude de l'art d'Android. Puis nous avons présenté l'environnement technique du travail. Le troisième chapitre a porté sur la création d'un simple projet Android, HelloAndroid. Finalement, nous avons décrit l'application de messagerie simple et mis l'accent sur la fonctionnalité ajoutée.