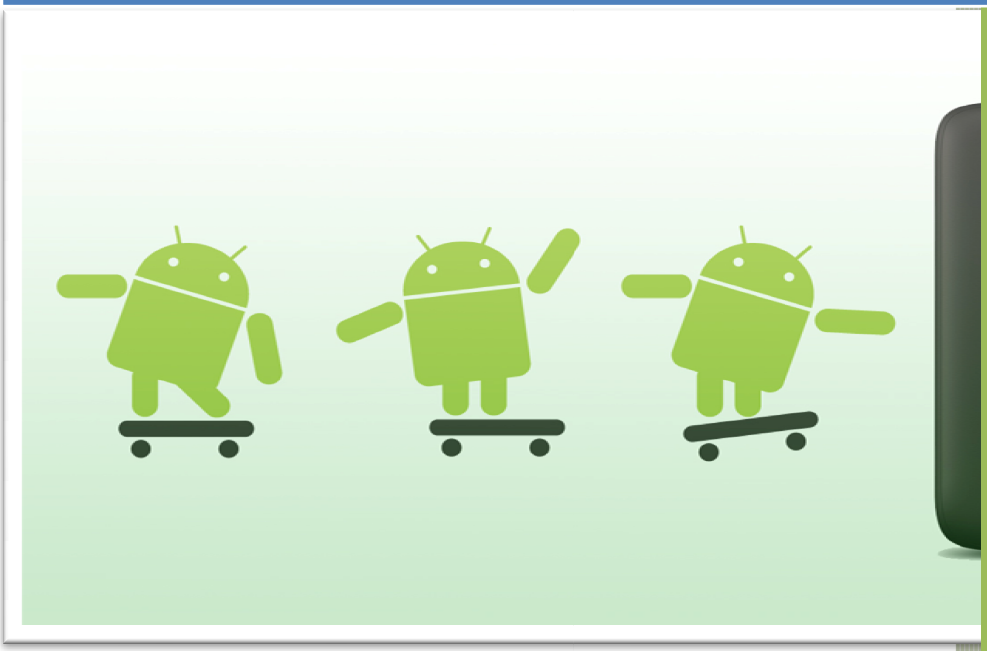
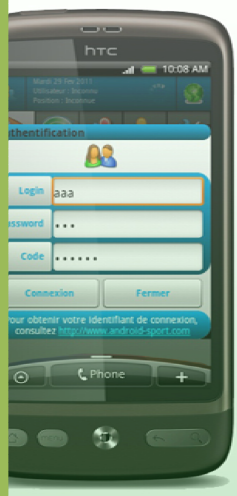
ISIMA

)11

e sous
)id 2.2





Encadré par :

- Philippe Lacomme

Responsable INRA :

- Sylvie Rousset

)na Andrianaina

Sofiane

) Andry

: Blaise Pascal II - Informatique

11

TABLE DES MATIERES

Sommaire

TABLE DES FIGURES

Remerciements

Nous tenons tout d'abord à remercier M. Philippe Lacomme, notre tuteur de stage, de nous avoir proposé ce sujet, de nous avoir aidé et encadré tout au long du projet.

Nous remercions aussi le directeur technique et tout le personnel de l'INRA ayant assisté à la présentation du projet, de nous avoir encouragés pour ce projet

Nous remercions particulièrement chaque étudiant du groupe, d'avoir participé et contribué à atteindre notre objectif

RESUME

Nous avons effectué notre projet de fin d'études de la 3ème année de licence sous l'encadrement de M. Philippe Lacomme. Ce travail a été réalisé en liaison avec Sylvie Rousset Ingénieur de Recherche INRA.

Le projet a pour but de nous initialiser à la plateforme de développement mobile sous Android et de tirer profit des gadgets que possèdent les Smartphones de nos jours. Les principaux gadgets qui nous seraient utiles pour élaborer ce projet sont les capteurs sensoriels et le GPS. Etant donnée l'émergence de technologie mobile, beaucoup d'applications ont été développées dans divers domaines. Cependant nous avons pensé comment utiliser ces gadgets pour aider chaque individu à suivre ses efforts physiques grâce à son Smartphone.

La première étape a été l'étude préalable du logiciel, c'est à dire définir le contexte et le choix des outils à utiliser pour le développement. Etablir les fonctionnalités de base ainsi que les diagrammes nécessaires à la conception du logiciel.

Dans un second temps, nous avons choisi un IDE de développement permettant d'implémenter le logiciel, sachant que celui-ci s'écrit sous le langage JAVA. Entre NetBeans et Eclipse, nous avons choisi NetBeans pour mieux le maitriser.

Dans la dernière partie, nous avons créé un site web puis nous l'avons héberger temporairement sur le serveur web de l'ISIMA. Compte tenu de la faible capacité de la batterie des Smartphones à fournir l'énergie nécessaire à une utilisation intensive des capteurs et du GPS, le site permettra d'alléger et de diminuer la consommation énergétique du téléphone en déléguant tous les gros calculs sur celui-ci.

Enfin, il faut noter que comme tous les projets, celui-ci est destiné à évoluer au fil du temps.

Mots-clés : Android 2.1, Java, Dépense énergétique, maps

INTRODUCTION

Dans le cadre de notre préparation au diplôme de Licence L3 parcours Informatique, nous avons été amenés à effectuer notre projet sous l'encadrement de M. Philippe Lacomme.

Il s'agissait au cours du projet d'étudier et de développer une application mobile sous Android. 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, à installer sur les Smartphones permettant de suivre sa dépense énergétique tout en gardant son téléphone dans sa poche ou accroché à sa ceinture, puis d'être en contact avec ses ami(es) et savoir où ils se trouvent grâce à la Google map intégré au logiciel lui même.

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 entamerons la modélisation et le développement de l'application.

Ensuite, créer un serveur web pour pouvoir stocker les données dans une base de donnée est facile d'accès grâce à un site web que nous avons développé par la suite. Les gros calculs seront implémenter sur le serveur web afin d'éliminer ou de diminuer les contraintes matérielles limitant notre application sur les Smartphones.

Ce rapport peut ainsi être subdivisé en quatre parties. La première consistera à la présentation du projet, des objectifs principaux puis les fonctionnalités de base. La seconde partie sera consacrée au développement proprement dit de l'application. La troisième partie propose une ébauche d'un site web qui servira d'interface entre le serveur web et l'application elle-même et implémentant toutes les fonctionnalités nécessaires à la communication entre elles. Enfin, la quatrième et dernière partie sera réservée à présenter l'application avec les fonctionnalités de base et une notice permettant de comprendre comment utiliser le logiciel.

PRESENTATION DU PROJET

Contexte

Les Smartphones de nos jours sont équipés de plusieurs capteurs capables de mesurer l'accélération, l'altitude ou même de localiser la position du téléphone avec précision. Ces derniers peuvent ainsi être utilisés à des fins médicales.

Interaction client/médecin

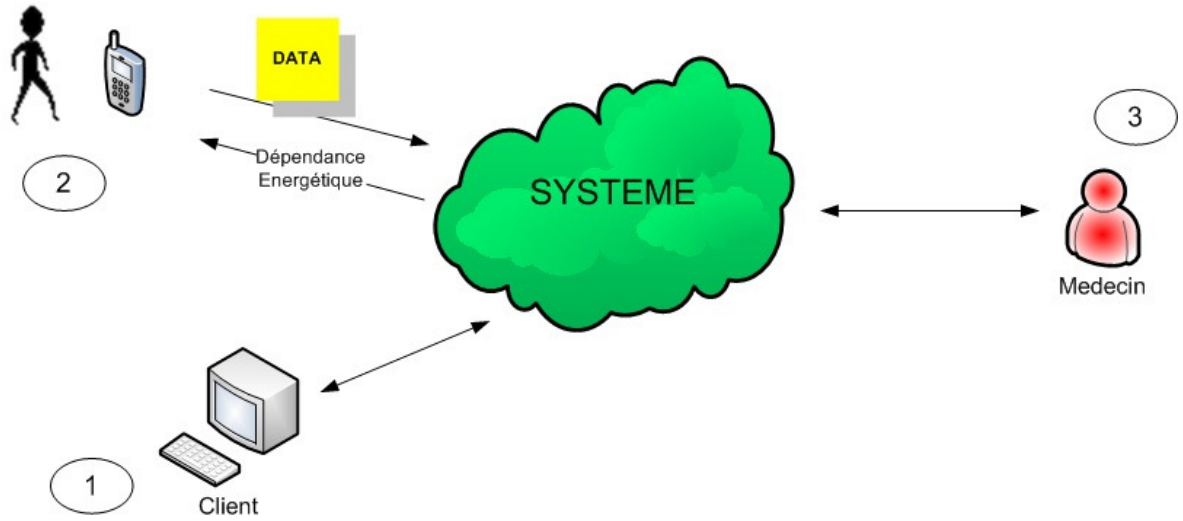


Figure 1 : Aperçu globale du système

Comme on peut le constater dans la figure ci-dessus, l'interaction entre l'individu et son médecin se résume en trois étapes :

Pour commencer la personne utilisant le smartphone doit au préalable créer un compte utilisateur sur le site web, qui permet de classer les individus et les différencier.

En second lieu, le client ayant un compte validé par un administrateur pourra se connecter et disposer des différentes fonctionnalités de l'application Android. Il pourra alors pendant son activité sportive, transmettre les données au sein du système.

Pour terminer, les données transferées par l'utilisateur via le telephone Android seront récupérées par son médecin personnel qui pourra traiter ces dernières.

Objectif du projet

Le travail qui nous a été demandé a été de développer une application mobile sous Android, permettant de mesurer la dépense énergétique d'une personne grâce à un Smartphone. La principale fonctionnalité étant de mesurer la dépense énergétique pendant une marche, une course à pied ou à vélo. Au terme de ce projet, l'application devra être en mesure de deviner automatiquement l'activité de l'utilisateur.

Dans le cadre de ce projet, les fonctionnalités principales demandées sont la récupération des données depuis les capteurs, les coordonnées GPS et l'envoi des données acquises sur un serveur web où ils seront traités plus tard.

Le fonctionnement et la mise en place du serveur sont décrits par le schéma ci-dessous:

Le principe est donc que le Client (ordinateur ou le téléphone mobile), grâce à son navigateur, envoie une requête http au serveur Web via Internet. Le serveur traitera à son tour la requête correspondante et renverra la réponse au Client depuis Internet.

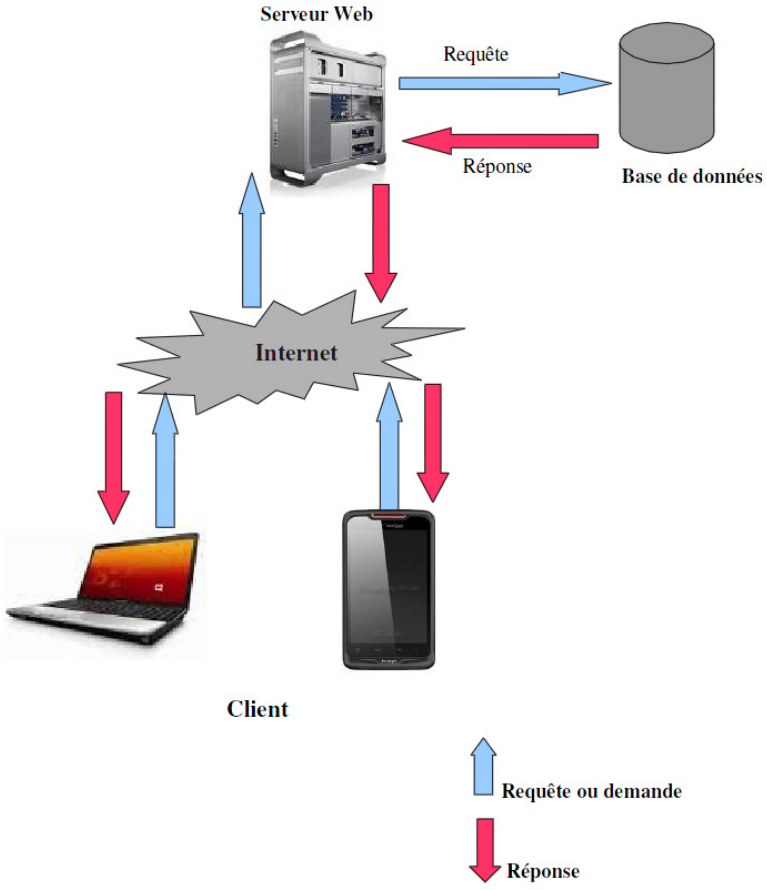


Figure 2 : Communication mobile - Internet - Serveur Web - Base de données

Contrainte

Nous sommes limités à l'autonomie de la batterie et de la taille de la mémoire du téléphone. Pour pouvoir communiquer avec le serveur web et s'échanger des données, le téléphone doit être connecté à Internet (abonnement chez un fournisseur).

Par rapport au téléphone, le serveur présente un grand avantage du point de vue mémoire qu'autonomie. Pour l'instant, le serveur web est hébergé temporairement sur le serveur de l'ISIMA. La difficulté surviendrait lorsque le site web changera de serveur. C'est à dire trouver un moyen de changer automatiquement la nouvelle adresse sur chaque application sans l'intervention de l'utilisateur et sans recompiler l'application.

DEVELOPPEMENT MOBILE

Présentation (environnement de développement)

Les technologies mobiles prennent de plus en plus de place sur le marché. 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. Sous Android, une application est composée d'une ou plusieurs activités. Une activité est la base d'un composant pour la création d'interfaces utilisateur. Afin de facilité la cinématique de l'application, il est préconisé de n'avoir qu'une interface visuelle par activité.

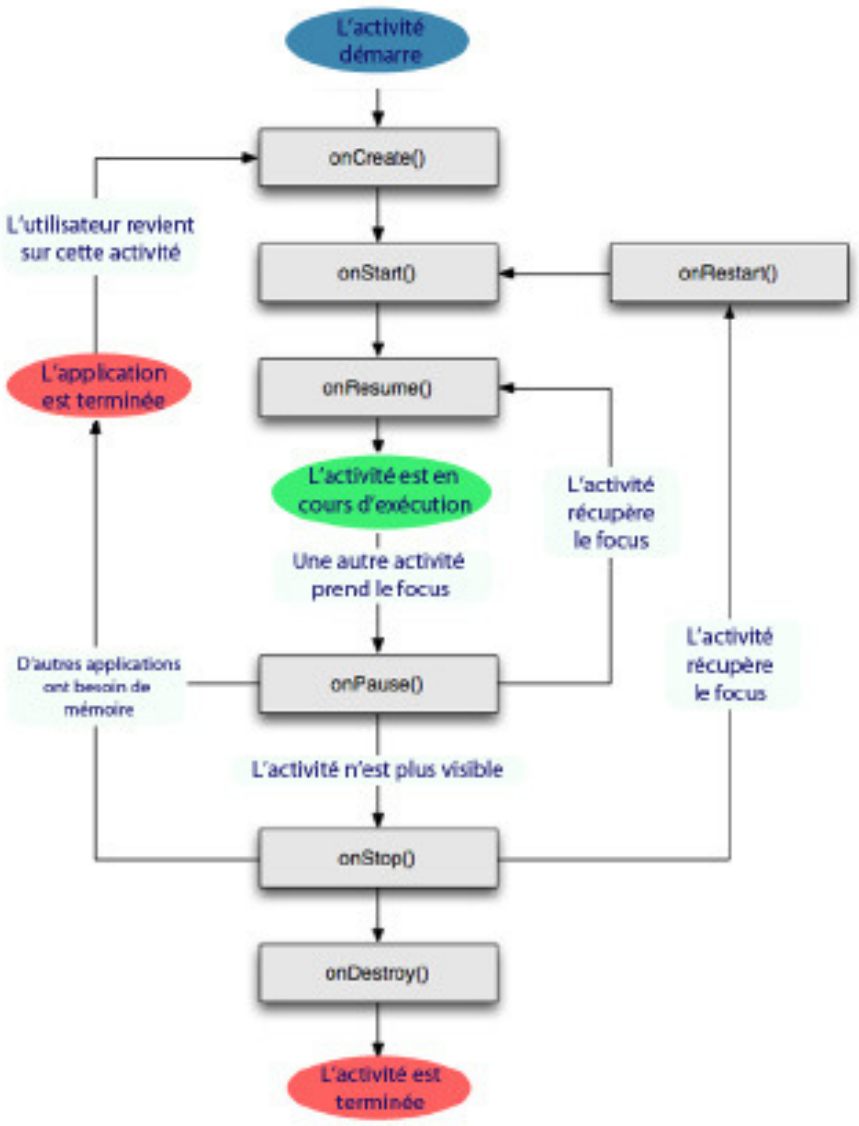


Figure 3 : Cycle de la vie d'une application (LIBRAIRIE.IMMATERIEL.FR/FR/READ_BOOK/9782815002028/DEVELOPPEZ-POUR-ANDROID_SPLIT_001)

- L'activité démarre : la méthode *onCreate* est appelée.

- Pendant l'utilisation d'une activité, l'utilisateur presse la touche Accueil, ou bien l'application téléphone, qualifiée comme prioritaire et qui interrompt son fonctionnement par un appel téléphonique entrant. L'activité est arrêtée (appel de *onStop*), le développeur détermine l'impact sur

l'interface utilisateur, par exemple la mise en pause d'une animation puisque l'activité n'est plus visible.

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

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

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

Android se base sur un noyau Linux 2.6. Le SDK Android possède une bibliothèque de librairie de plusieurs classes java de base pour plusieurs types d'application (exemple : OpenGL|ES pour la 3D, SSL pour les protocoles de sécurité, etc.). Une application Android se repose sur un Framework qui facilite l'utilisation des classes de base et sert d'interface entre les "Librairies" et les applications.

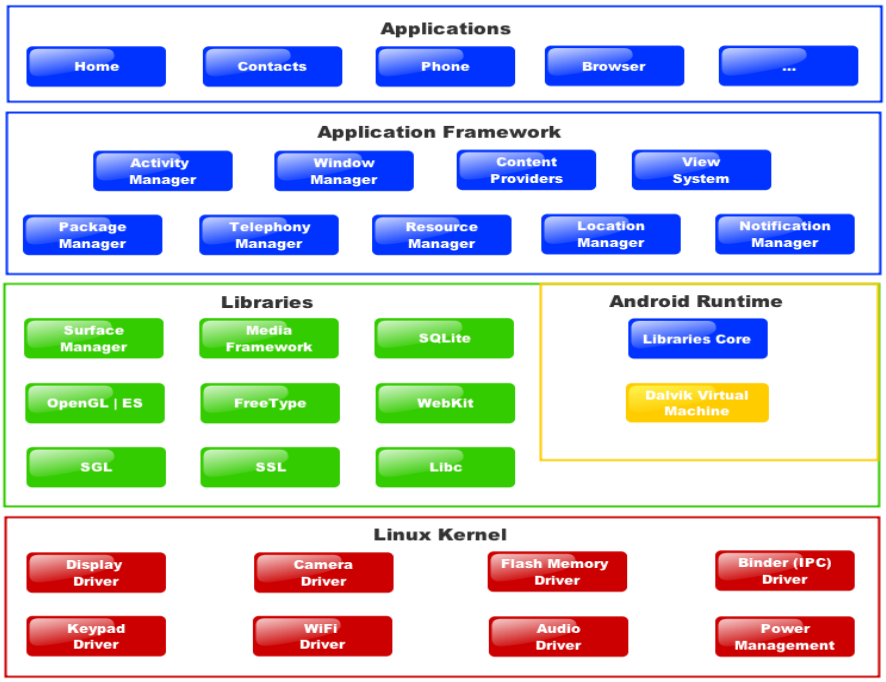


Figure 4 : Architecture plateforme Android (HTTP://FLEXGURUIN.WORDPRESS.COM/CATEGORY/ANDROID)

Android est un système d'exploitation pour téléphone portable de nouvelle génération développé par Google. Celui-ci met à disposition un kit de développement (SDK) basé sur le langage Java.

Nous avons eu l'occasion d'utiliser deux outils de développement : NetBeans et Eclipse. La seule différence que nous avons constatée, se situe au niveau de l'édition de l'interface graphique. Sous éclipse, l'éditeur graphique est un éditeur wysiwyg (What You See Is What You Get) ce qui n'est pas le cas pour NetBeans où il faut saisir le code pour générer l'interface graphique.

Pour la suite du projet, nous utiliserons NetBeans comme cela a été demandé, et pour mieux comprendre le fonctionnement de l'interface graphique à l'aide du code.

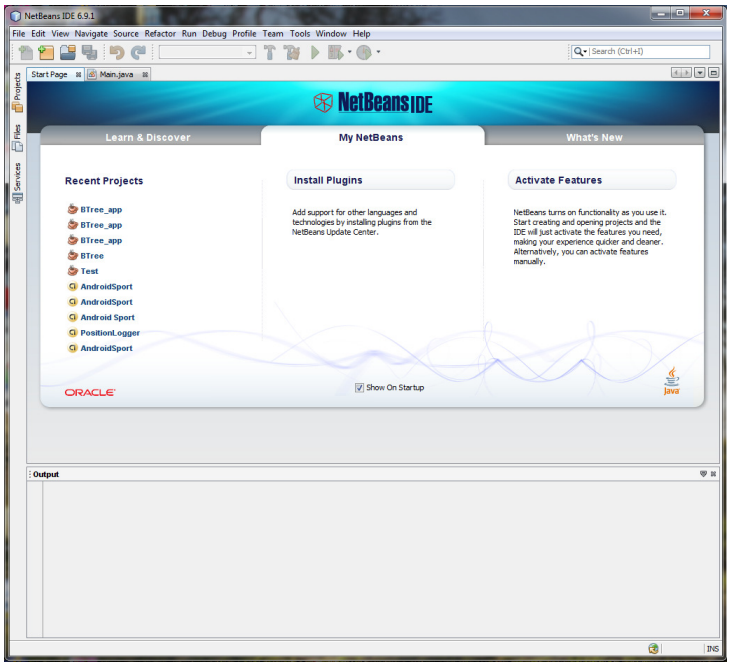


Figure 5 : IDE NetBeans

Nous vous invitons à consulter l'url "HTTP://KENAI.COM/PROJECTS/NBANDROID/PAGES/INSTALL " pour voir les étapes d'installation du plugin Android, sur NetBeans IDE.

Le kit de développement Android est téléchargeable gratuitement sur : "HTTP://DEVELOPER.ANDROID.COM/SDK/INDEX.HTML" ainsi qu'un guide d'installation.

Voici à quoi ressemble le SDK Android (SDK Manager) après l'installation et lancement :

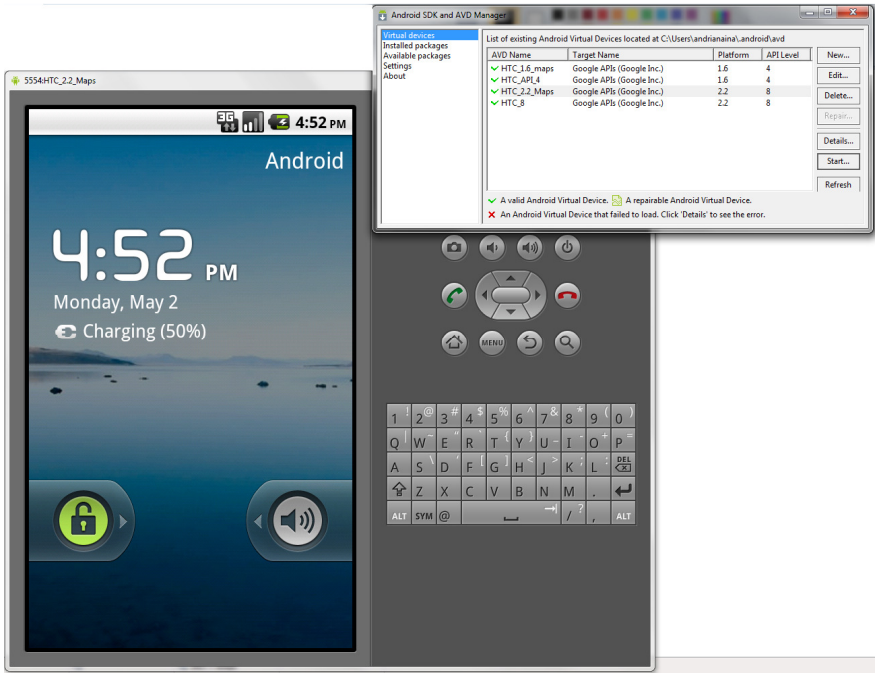


Figure 6 : Android SDK and AVD Manager

La figure (6) montre le lancement d'un émulateur Android sur le port : 5554 du localhost, parmi 4 émulateurs.

Modélisation

Diagramme de cas d'utilisation

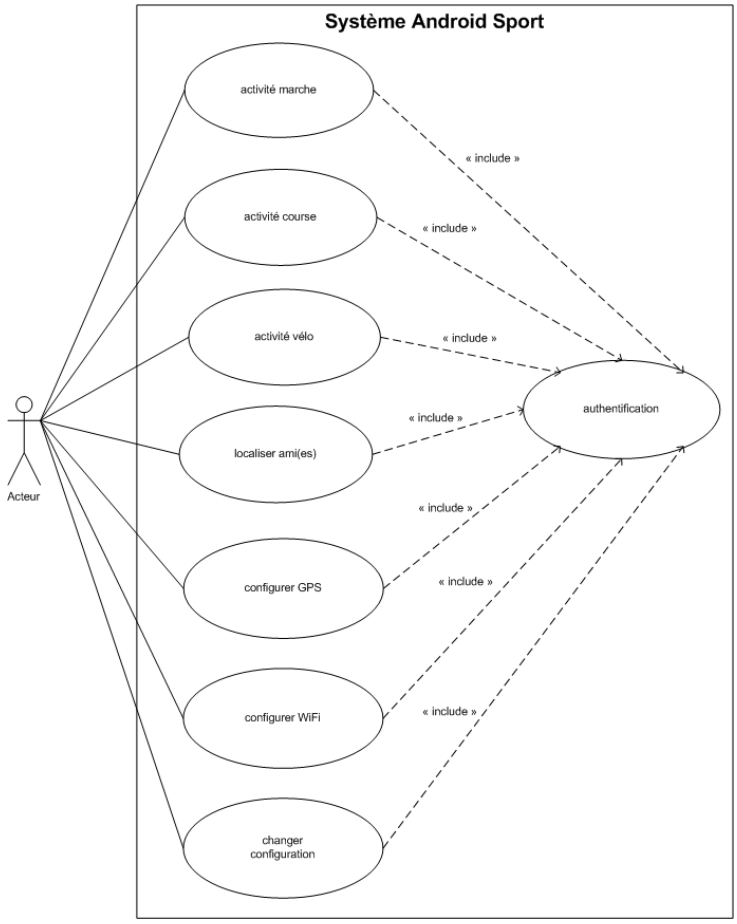


Figure 7 : Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation ci-dessus nous montre l'interaction entre l'utilisateur et l'application. Les fonctionnalités principales sont marche, course et vélo.

L'utilisateur doit s'authentifier sur le serveur web pour pouvoir utiliser l'application. Une fois connecté, il peut configurer les capteurs (activer et désactiver), le GPS ainsi que les paramètres de connexion.

Diagramme de classe

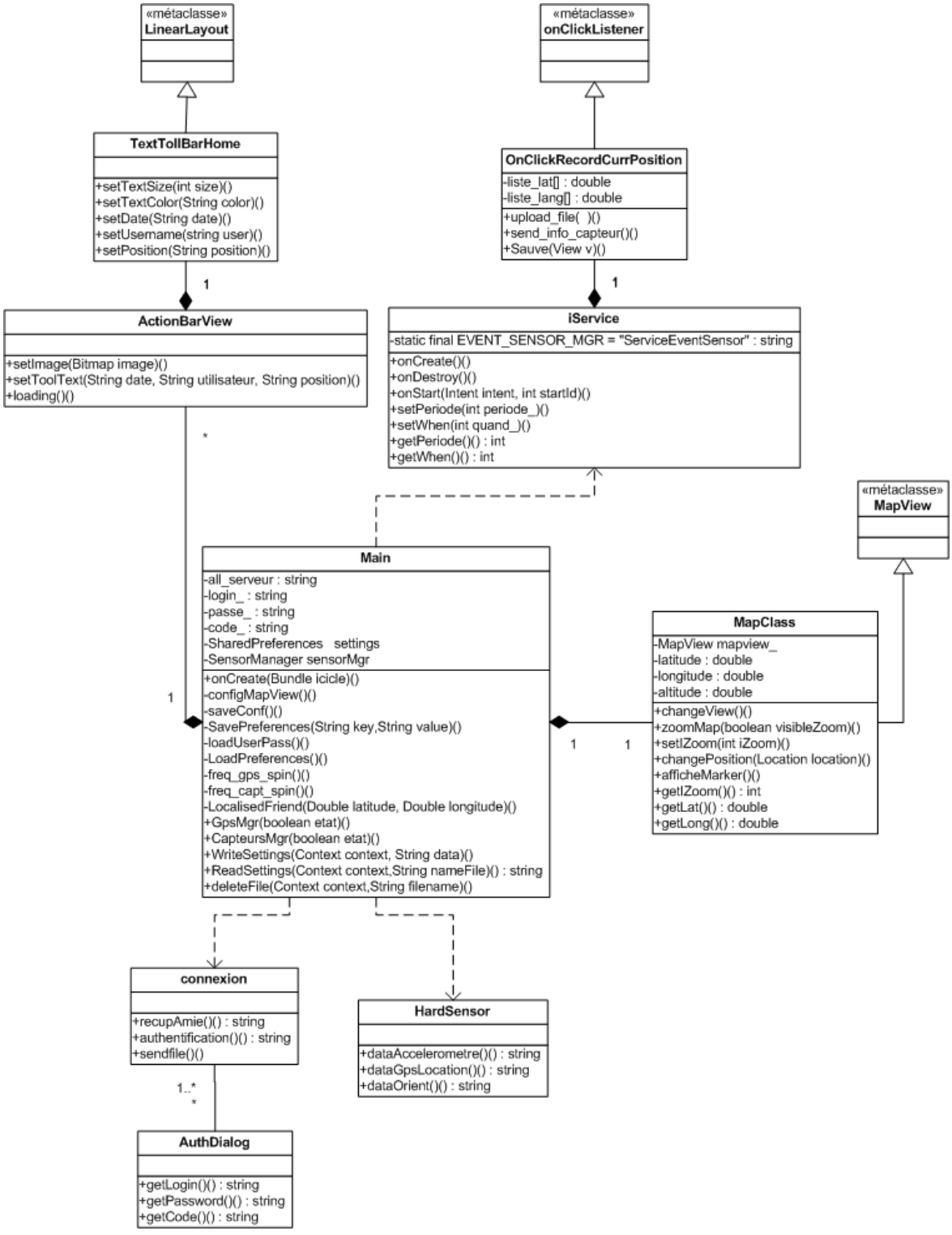


Figure 8 : Diagramme de classe

Implémentation

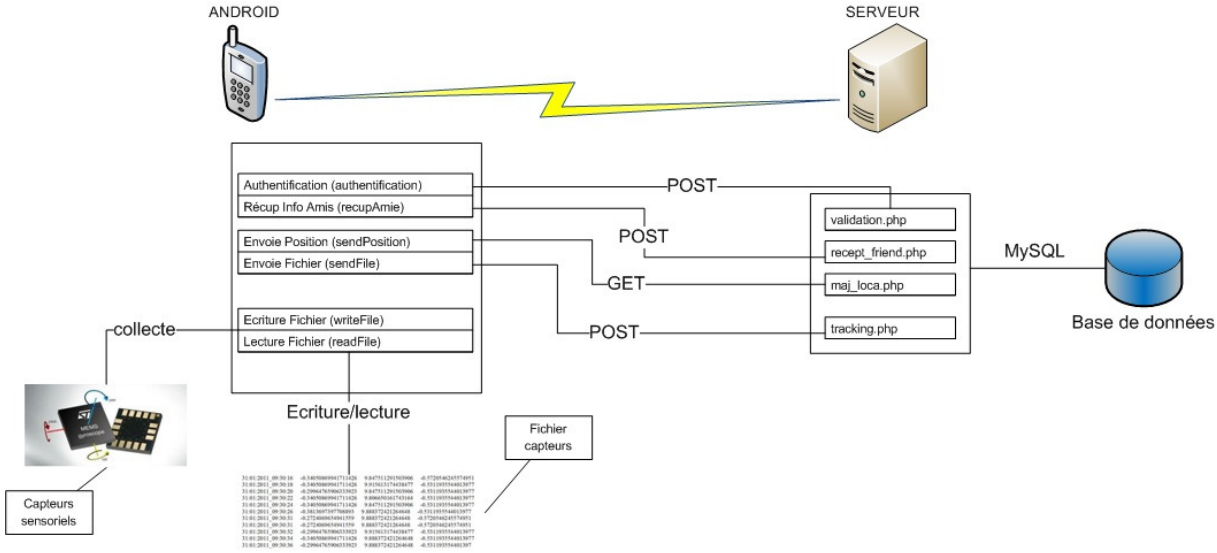


Figure 9 : Communication Android/ Serveur Web/ Base de données

L'environnement de développement sous Android se fait sous JAVA. Nous avons deux choix d'outils pour le faire : NetBeans et Eclipse. Cependant, sur chacun de ces outils doit être installé le plugin qui permettra de développer des applications sous Android.

Pour s'exécuter sur les appareils mobiles Android, le programme en java doit être converti. Dalvik est la machine virtuelle qui gère la plate-forme Java sur les appareils mobiles Android. Elle permet d'exécuter les applications qui ont été converties en un exécutable compact Dalvik (.dex), un format adapté aux systèmes limités en termes de mémoire et de vitesse du processeur.

Dans tout le projet, nous utiliserons NetBeans car nous avons constaté moins de bug.

Pour mieux structurer la logique application, nous l'avons séparé en modules. Une première « classe », qui contient les classes suivantes :

- La classe « connexion », contient les méthodes et les protocoles de connexion au serveur web
- La classe « ActionViewBar », contient les paramètres utilisateur (login, position, état de connexion) ainsi que la date en temps réelle.
- La classe « HardSensor », contient les méthodes qui permettent de manipuler et récupérer les données de chaque capteur ainsi que la position à partir du GPS.
- La classe « iService », contient le service et les méthodes qui permettent d'envoyer régulièrement et périodiquement les données sur le serveur.

En cas de coupure de la connexion Internet, les données sont écrites dans un fichier texte. Grâce à la classe « iService », l'application enverra les données dès que la connexion sera rétablie.

L'interface graphique

Nous vous invitons à consulter un document sur la page personnel de M. Philippe Lacomme : "HTTP://WWW.ISIMA.FR/~LACOMME/DEVPORTABLE/TUTORIAL_ANDROID.PDF".

Android utilise des fichiers ressources xml : "main.xml", pour générer une interface graphique (ou view). Ce fichier est appelé par le fichier principal "main.java" grâce à la commande "setContentView(R.layout.main)". "R" est un fichier ressource de l'application contenant tous les identifiants des widgets (références vers les widgets). Il est généré automatiquement par le SDK.

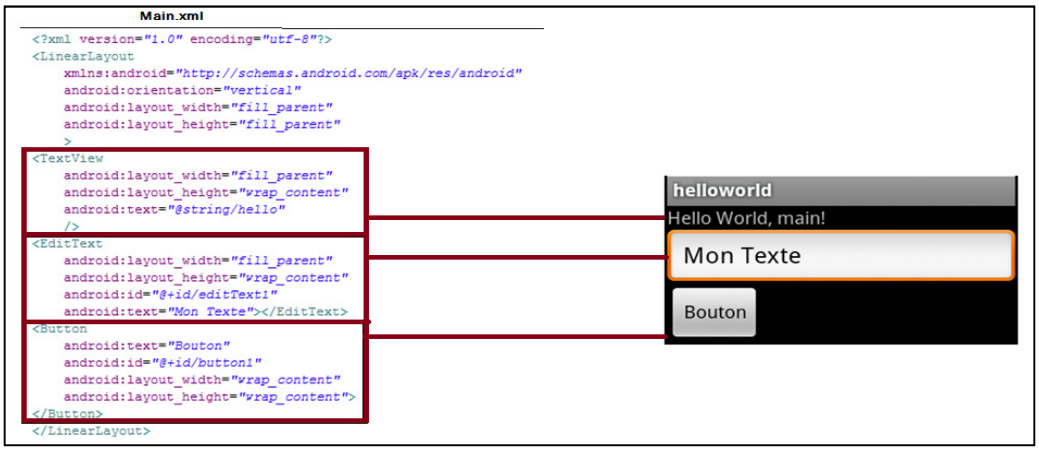


Figure 10 : view xml

La figure ci-dessus nous montre comment générer une interface graphique en xml. Les composants ou widgets sont définis par des balises et les attributs représentent les propriétés. Pour placer les widgets, Android dispose deux types de gestionnaire de disposition (ou layout) : LinearLayout (dispose les widgets horizontalement ou verticalement comme ci-dessus) , TableLayout (layout en forme de tableau).

"<TextView>" permet de créer un widget "label" . La propriété "android:text" récupère et affiche dans le widget, le contenu de la ressource "string" ayant le nom hello référencé par "@string/hello". La propriété "android:id" permet de rajouter automatiquement dans le fichier "R", l'identifiant du widget.

Gestion d'évènement

```
import android.widget.Button;
import android.app.Activity;
import android.widget.EditText;
import android.widget.Button;

public class Main extends MapActivity{
        private EditText zoneText;
        private Button bouton;
        @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
            setContentView(R.layout.main);
            zoneText = (EditText)findViewById(R.id.editText1);
            bouton = (Button)findViewById(R.id.button1);
            bouton.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v){
                switch(v.getId()){
                    case R.id.button1:
                        zoneText.setText("Hello World");
                }
            }
        }
    }
}
```

Figure 11 : Gestion d'évènement click d'un bouton

Le code de la figure ci-dessus permet d'écrire "Hello World" dans la zone de texte "zoneText" lors d'un clique sur le bouton "bouton". "onCreate" créer le view du fichier "main.xml" précédent lors de l'appel. Au bouton est assigné un listener (écouteur d'évènement clic) grâce à

"setOnClickListener". "onClick(View v)" exécutera l'instruction du bloc. "v.getId" récupère l'identification du ou des widgets cliqués.

Gestion des capteurs

```
import android.hardware.SensorManager;
import android.hardware.SensorEvent;

...

private SensorManager sensorMgr;
@Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);

        ......

        sensorMgr = (SensorManager)getSystemService(SENSOR_SERVICE);
        boolean accelSupported = sensorMgr.registerListener( this
                , sensorMgr.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
                 ,sensorMgr.SENSOR_DELAY_FASTEST);
        if (!accelSupported){
            sensorMgr.unregisterListener(this, sensorMgr.getDefaultSensor(
            Sensor.TYPE_ACCELEROMETER));
            Toast.makeText(this, "Pas de capteur d'accéléromètre", Toast.LENGTH_LONG);
        }
    }
     public void onSensorChanged(SensorEvent event) {
        switch(event.sensor.getType()){
            case Sensor.TYPE_ACCELEROMETER:
                onAccelerometerChanged(event);
            break;

        }
    }
     private void onAccelerometerChanged(SensorEvent event){
        float x, y, z;
        x = event.values[0];
        y = event.values[1];
        z = event.values[2];

     }
}
```

Figure 12 : Récupération des valeurs du capteur

Le code de la figure 12 nous montre comment récupérer une instance du capteur de type "TYPE_ACCELEROMETER" afin de récupérer les valeurs (x,y,z) correspondant lorsque la méthode "onSensorChanged" est appelée (la valeur des capteurs change). L'évènement "event" contient alors les valeurs de chaque coordonnée sous forme d'un tableau.

Gestion des services

iService.java

```
import java.util.Timer;
import java.util.TimerTask;
import android.app.Service;
import android.content.Intent;
import android.os.Binder;
import android.os.IBinder;


public class iService extends Service {
    public static final String EVENT_SENSOR_MGR = "ServiceEventSensor";
    private long periode = 3000;
    private long quand = 0;
    TimerTask    tache;
    Timer       timer;

    public class MyBinder extends Binder{
        iService getService(){
            return iService.this;
        }
    }
    @Override
    public IBinder onBind(Intent arg0) {
        return new MyBinder();
    }
    @Override
    public void onCreate(){
        timer = new Timer();
        tache = new TimerTask(){
            @Override
            public void run(){
                Intent intent = new Intent(EVENT_SENSOR_MGR);
                intent.putExtra("Value", System.currentTimeMillis());
                iService.this.sendBroadcast(intent);
            }
        };
     super.onCreate();
     }
    @Override
    public void onStart(Intent intent, int startId) {
        if(active==false){
            timer.scheduleAtFixedRate(tache, quand, periode);
        }
        super.onStart(intent, startId);
    }
}
```

Main.java

```
import android.os.Bundle;
import android.content.Intent;
import com.project.androidsport.services.iService;
import android.content.BroadcastReceiver;
import android.view.View;

...

@Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);

        ......

       @Override
       public void onClick(View v) {
            switch(v.getId()){

                ...

                    Intent bindIntent = new Intent(Main.this, iService.class);
                    startService(bindIntent)

            }
       }

       public class Receiver extends BroadcastReceiver{

           @Override
           public void onReceive(Context context, Intent intent) {
               if(intent.getAction().equals(iService.EVENT_SENSOR_MGR)){
                   Bundle bundle = intent.getExtras();
                   if(bundle!=null){
                   // instruction

                   }
               }
           }
       }
}
```

Figure 13 : Manipulation service

Cette figure nous montre comment créer un service. On peut constater que l'on utilise la méthode « intent », cette dernière est une description abstraite d'une opération à effectuer. Elle permet le dialogue à travers le système à partir des canaux qui lui sont dédiés.

Par exemple : Quand le mobile reçoit un appel, la plateforme lance un intent signalant l'arrivé d'un appel.

Les "Intent" permettent d'envoyer des données dans toute l'application puis récupérer par un "BroadcastReceiver". Le "Timer "timer" permet de fixer une fréquence périodique d'envoyer des données en utilisant la méthode "scheduleAtFixedRate". Le service peut être démarrer (startService) ou arrêter (stopService).

Communication Java / PHP

L'un des objectifs principaux de ce projet étant de faire communiquer l'application Android (Java) avec le serveur web grâce au site web (PHP). Pour cela, nous avons deux possibilités : utiliser directement les sockets ou utiliser des packages de classe manipulant les protocoles Http implémentant les sockets, dans le code java. Nous avons opté pour l'utilisation des packages Http de Apache renfermant tous les protocoles et outils pouvant manipuler les flux de données sur Internet.

Les packages utilisés sont les suivants avec leurs liens de téléchargement et leurs documentations respectifs :

apache-mime4j-0.4.jar	Fournit un analyseur pour des données XML. Il ne traite que la structure des flux de données XML.
commons-io-2.0.1.jar	Utilisée pour aider à développer des fonctionnalités d'E/S.
httpclient-4.0-beta2.jar	Permet d'utiliser les fonctionnalités du protocole http coté client, le rôle de ce package est de transmettre et recevoir des messages http.
httpcore-4.0-beta3.jar	Son objectif est la mise en œuvre des aspects les plus fondamentaux du protocole http et d'éviter un encombrement du fonctionnement des services coté client et serveur.
httpmime-4.0-beta1.jar	Utilisé pour typer des documents transférés par le protocole http.

- apache-mime4j-0.4.jar :
http://grepcode.com/snapshot/repo1.maven.org/maven2/org.apache.james/apache-mime4j/0.4
- commons-io-2.0.1.jar :
http://grepcode.com/snapshot/repo1.maven.org/maven2/commons-io/commons-io/2.0.1
- httpclient-4.0-beta2.jar :
http://grepcode.com/snapshot/repo1.maven.org/maven2/org.apache.httpcomponents/httpclient/4.0-beta2
- httpcore-4.0-beta3.jar :
http://grepcode.com/snapshot/repo1.maven.org/maven2/org.apache.httpcomponents/httpcore/4.0-beta3
- httpmime-4.0-beta1.jar :
http://grepcode.com/snapshot/repo1.maven.org/maven2/org.apache.httpcomponents/httpmime/4.0-beta1

```
package com.project.androidsport;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
```

Figure 14 : import package apache http

```
...

public String authentification(String serveur,String login, String passe, String code){
        String ligneCodeHTML= "";
        String url= serveur + "/mobile/validation.php?log_iconnect=" + login
                + "&pwd_iconnect=" + passe + "&num_iconnect=" + code ;
        BufferedReader bufferedReader = null;

        try{
            // Création d'un DefaultHttpClient et un HttpGet permettant d'effectuer une requete HTTP
            HttpClient httpClient = new DefaultHttpClient();
            HttpGet httpGet = new HttpGet();

            // Création de l'URI et on l'affecte au HttpGet
            URI uri = new URI(url);
            httpGet.setURI(uri);

            // Execution du client HTTP avec le HttpGet
            HttpResponse httpResponse = httpClient.execute(httpGet);

            // On récupère la réponse dans un InputStream
            InputStream inputStream = httpResponse.getEntity().getContent();

            // On crée un bufferedReader pour pouvoir stocker le résultat dans un string
           bufferedReader = new BufferedReader(new InputStreamReader(inputStream));

           // On lit ligne à ligne le bufferedReader pour le stocker dans le stringBuffer
          ligneCodeHTML = bufferedReader.readLine();

        }catch(Exception e){

        }finally{
            // Dans tous les cas on ferme le bufferedReader s'il n'est pas null
            if (bufferedReader != null){
                try{
                    bufferedReader.close();
                }catch(IOException e){

                }
            }
        }
        return ligneCodeHTML;
    }
```

Figure 15 : Fonction authentification méthode Get

La figure 15 montre l'utilisation de la méthode HttpGet. La fonction authentification retourne une chaine de caractère envoyée par le serveur qui est dans notre cas :

- ***"Id_OK"*** si l'utilisateur est membre actif,
- ***"Compte non actif."*** si l'utilisateur n'a pas encore été validé par l'administrateur,
- ***"Mauvais numeros d'acces"*** si le code d'accès est erroné,
- ***"Mauvais mot de passe."*** si le mode de passe est erroné,

- ***"Identifiant ".$log_iconnect." inconnu !"*** si l'utilisateur n'est pas membre du site ($log_iconnect : login de l'utilisateur).

Les chaines de caractères précédemment citées sont utilisées pour tester l'état des utilisateurs qui se connecte et utilise l'application. Remarquons que sans authentification, l'utilisateur ne pourra pas accéder à l'application (l'application reste sur la page d'authentification).

```
public void sendfile(Context context,String adresse, String login, String password, String code, String action, String filename, String paramFile){
    int  serverResponseCode =0;
    String serverResponseMessage="";

    HttpURLConnection connection = null;
    DataOutputStream outputStream = null;
    DataInputStream inputStream = null;
    String pathToOurFile =context.getFileStreamPath(filename).getAbsolutePath();
    String urlServer = adresse + "/mobile/tracking.php?&log_itracking=" + login +
                    "&pwd_itracking=" + password + "&num_itracking="+ code
                    + "&act_itracking=" + action ;
    String lineEnd = "\r\n";
    String twoHyphens = "--";
    String boundary =  "*****";

    int bytesRead, bytesAvailable, bufferSize;
    byte[] buffer;
    int maxBufferSize = 1*1024*1024;

    try{
        FileInputStream fileInputStream = new FileInputStream(new File(pathToOurFile) );
        URL url = new URL(urlServer);
        connection = (HttpURLConnection) url.openConnection();
        // Allow Inputs & Outputs
        connection.setDoInput(true);
        connection.setDoOutput(true);
        connection.setUseCaches(false);

        // Enable POST method
        connection.setRequestMethod("POST");

        connection.setRequestProperty("Connection", "Keep-Alive");
        connection.setRequestProperty("Content-Type", "multipart/form-data;boundary="+boundary);

        outputStream = new DataOutputStream( connection.getOutputStream() );
        outputStream.writeBytes(twoHyphens + boundary + lineEnd);
        outputStream.writeBytes("Content-Disposition: form-data; name=" + paramFile
                + ";filename=\"" + pathToOurFile +"\"" + lineEnd);
        outputStream.writeBytes(lineEnd);

        bytesAvailable = fileInputStream.available();
        bufferSize = Math.min(bytesAvailable, maxBufferSize);
        buffer = new byte[bufferSize];
        // Read file
        bytesRead = fileInputStream.read(buffer, 0, bufferSize);

        while (bytesRead > 0) {
            outputStream.write(buffer, 0, bufferSize);
            bytesAvailable = fileInputStream.available();
            bufferSize = Math.min(bytesAvailable, maxBufferSize);
            bytesRead = fileInputStream.read(buffer, 0, bufferSize);
        }

        outputStream.writeBytes(lineEnd);
        outputStream.writeBytes(twoHyphens + boundary + twoHyphens + lineEnd);

        // Responses from the server (code and message)
        serverResponseCode = connection.getResponseCode();
        serverResponseMessage = connection.getResponseMessage();

        Toast.makeText(context, "Code : " + serverResponseCode + "\nmessage : " + serverResponseMessage,Toast.LENGTH_LONG).show();

        fileInputStream.close();
        outputStream.flush();
        outputStream.close();
    } catch (Exception ex) {
        //Exception handling
        Toast.makeText(context, ex.getMessage(),Toast.LENGTH_LONG).show();
    }

}
```

Figure 16 : Envoie de donnée utilisant la classe connexion (methode POST)

La figure 16 montre comment envoyer un fichier sur le serveur. Les paramètres à renseigner de la fonction par ordre sont : le contexte (l'application parent), l'adresse url du site, le login - mot de

passe - code d'accès de l'utilisateur, l'action ou l'activité de l'utilisateur ("Marche, Course ou Vélo"), le nom du fichier à envoyer (fichier local sur le mobile) et le nom du paramètre ***$_FILES["..."]*** de la page php qui recevra le fichier.

Présentation de l'application

Figure 17 : Android Sport – présentation générale

La figure ci-dessus correspond à la vue d'ensemble des différents onglets présent dans notre programme que nous avons nommé Android Sport.

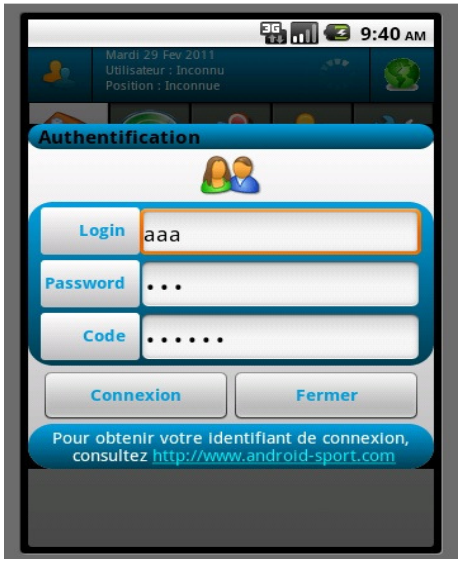


Figure 18 : Android Sport – Authentification

Au premier lancement de l'application, il faut se connecter sur le site web pour créer un compte et obtenir un code de connexion sur son e-mail. Après la validation du compte par l'administrateur, l'utilisateur peut ensuite accéder à toutes les fonctionnalités de l'application.

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

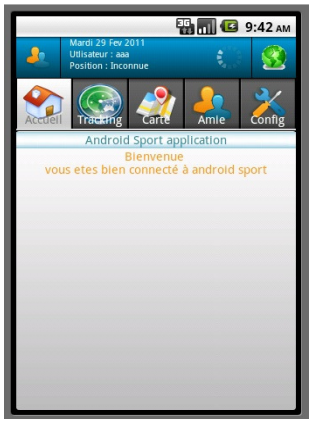


Figure 19 : Android Sport – Accueil

L'onglet "Accueil" affiche un message de bienvenue et assure que l'utilisateur est bien connecté.

Prochainement, sur l'accueil sera affiché l'activité de l'utilisateur et sa dépense énergétique. Ce qui n'est pas le cas, car le cadre du projet consiste à collecter les données puis les envoyer sur le serveur. Sachant que l'interprétation de ces données est hors de notre compétence. (échantillon des données - voir annexe)

Figure 20 : Android Sport – Tracking

Pour le moment, l'onglet "Tracking" permet à l'utilisateur de choisir une activité. Il suffit de cliquer sur l'un des boutons "Marche", "courir" ou "Vélo" pour l'activer. L'application va donc collecter les données des capteurs puis l'envoyer au serveur grâce à ses paramètres de connexions (login, mot de passe et code), périodiquement afin de minimiser l'utilisation simultanée des capteurs du réseau WiFi (contrainte d'autonomie de la batterie).

En cas de coupure de connectivité, les données collectées seront sauvegardées sur un fichier en attendant son rétablissement.

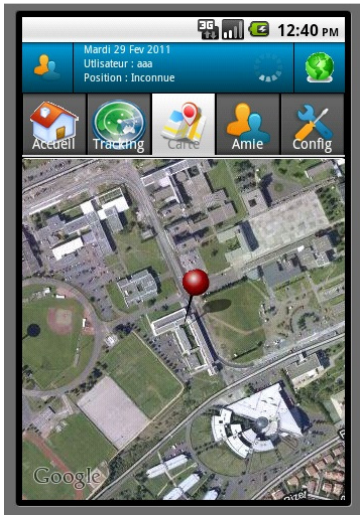


Figure 21 : Android Sport - Fonction Amie

L'onglet "Carte" permet de localiser et de marquer la position de ses amis (es) sur la "google map" de l'application. Dans la version suivante, l'onglet "Amie" affichera la liste des amis(es) ainsi que quelques informations comme l'état de la connexion, le login et leurs positions respectives.

Figure 22 : Android Sport – Config

L'onglet "Config" permet de sauvegarder les paramètres de connexion, à savoir l'adresse du serveur, le login, le mode passe, le code d'accès. Ainsi, l'utilisateur n'aura plus à saisir les paramètres de connexion à chaque lancement de l'application.

La configuration matérielle permet de choisir les fréquences de collecte et d'envoyer des données des capteurs afin d'économiser la batterie du téléphone.

Les problèmes rencontrés

Ce projet est notre première expérience en développement mobile. Plusieurs contraintes nous ont considérablement retardés :

- Limitation de la dimension de l'écran : choisir un minimum de composants essentiels à afficher sur l'écran, pas comme celui des ordinateurs.

- Sous Android, il faut gérer la reprise de l'état de l'application après une perturbation du cycle de vie de celui-ci. C'est-à-dire si au cours de son fonctionnement, nous avons un appel entrant ou un message, par ordre de priorité, le système peut arrêter l'activité en cours pour libérer de la

mémoire. Par conséquent, le programmeur doit donc sauvegarder des paramètres avant la destruction de l'instance de l'application, pour pouvoir par la suite les recharger. Car c'est une nouvelle instance de l'application qui sera lancée après la reprise. Grâce aux paramètres sauvegardés, nous avons l'impression que l'application était juste réduite.

- Au niveau de la compatibilité de certain composant à intégrer notre application.

DEVELOPPEMENT WEB

Présentation

Le serveur Web est l'élément essentiel garantissant le stockage des informations de taille importante, car le téléphone ne dispose pas assez de mémoire suffisante.

L'objectif du serveur Web dans ce projet est d'héberger une application permettant de collecter les informations notamment de l'utilisateur et d'en assurer le traitement des données.

Lors de la première utilisation, l'utilisateur doit se connecter sur le site Web afin d'enregistrer ses informations qui seront stockées dans la base de données du serveur Web. Par la suite le login et le mot de passe de son choix ainsi qu'un code généré automatiquement lui permettra de se connecter sur le serveur et utiliser l'application installée sur le téléphone. Une fois connecté sur le site, l'utilisateur pourra donc voir son profil ou le modifier par la suite en cas de besoin.

Une partie réseau sociale donne à l'utilisateur de chercher des amis (es), comme sur Facebook et twitter, et d'envoyer une demande d'amis (es) et de l'ajouter dans ces contacts. L'utilisateur concerné aura le choix d'accepter ou de refuser la demande en question.

Un compte administrateur permet à un utilisateur privilégié, d'accepter ou de refuser une ou plusieurs demandes de création de compte afin d'assurer facilement la sécurité ainsi que la gestion de capacité.

Modélisation

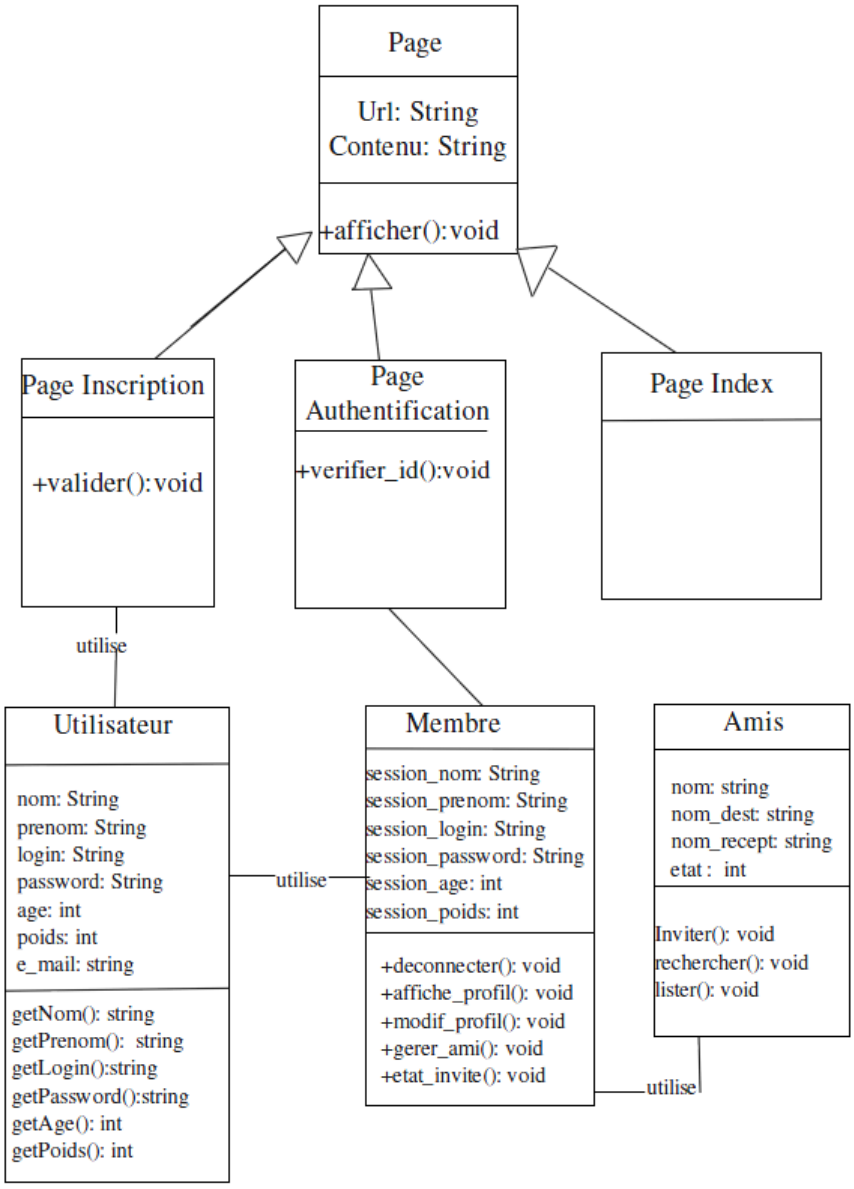


Figure 23 : Diagramme de classe site web

Implémentation

- styles2.css : code css implémentant le design du site Web

- entete.inc : fichier d'inclusion de l'entête de la page web

- pied_page.inc : fichier d'inclusion du pied de page

- index.php : page d'accueil du site web

- inscript.php : inclus le formulaire d'inscription de la page d'inscription

- valid_inscr.php: script qui vérifie si le champ a bien été rempli si oui, enregistrement des informations de l'utilisateur dans la base de données.

- login.php : script vérifiant si le login et le mot de passe entré par l'utilisateur sont corrects. S'ils sont corrects, on accède à l'espace membre en appelant le script '' membre.php ''.

- membre.php: script de l'espace membre incluant:

1. Vérification s'il y a une demande d'ami
2. Affichage du profil utilisateur
3. Modification du profil utilisateur
4. Gestion Ami

- ami.php : gère la recherche de personnes membres et possibilité de demande d'ajout à la liste d'ami, inclus aussi l'acceptation ou le refus d'amis.

Présentation à l'INRA

Durant notre projet nous avons eu l'occasion de présenter notre travail lors d'une réunion à l'INRA (CHU) en présence des personnalités suivantes :

J.M.Chardigny : Directeur de Recherche INRA et Directeur de l'Unité de Nutrition Humaine (UNH).

B.Morio : Directrice de Rechercheur INRA, responsable de l'équipe Contrôle de l'homéostasie lipido-énergétique et obésité (CHLEO).

P.Dusché : Vice présidente de l'Université Blaise Pascal.

P.Lacomme : Maitre de conférence LIMOS/ISIMA.

M.Duclos : Professeur et chef de service de la Médecine du Sport et des Explorations Fonctionnelles au CHU de Clermont.

S.Rousset : Ingénieur de recherche INRA au sein de l'équipe CHLEO, en charge de la modélisation de la dépense énergétique.

La présentation de notre travail lors de cette réunion faisait partie de notre projet, tenant compte que ce dernier est en liaison entre l'ISIMA et l'INRA(CHU).

Ce qui reste à faire

Pour atteindre l'objectif, il nous faut atteindre et comprendre comment interpréter les données collectées à partir de chaque capteur.

- deviner automatiquement l'activité actuelle de l'utilisateur (donc l'évaluation de la dépense énergétique)

- affichage de la liste des amis(es)

- modification des données paramètres de l'utilisateur (sur le serveur web) depuis le mobile

Plusieurs fonctionnalités peuvent encore être améliorées et rajoutées :

- plateforme "Tchat" pour permettre aux utilisateurs membres (et amis (es)) de s'envoyer des messages instantanément.

- tracé de la trajectoire de l'activité de l'utilisateur.

CONCLUSION

Au bout de notre cursus en licence informatique, nous avons été chargés de réaliser un projet de fin d'année. Notre travail s'est basé sur le développement d'un programme sur les technologies mobiles (Smartphone). Ceci nous a amené à découvrir une nouvelle plateforme de développement et à enrichir notre savoir et notre expérience.

L'application que nous avons réalisé permet d'hors et déjà à un utilisateur d'enregistrer au préalable de s'authentifier et ainsi il lui est permis d'accéder aux différentes fonctionnalités de l'application. En outre, la sauvegarde de ses paramètres d'accès, le transfert des données vers le serveur lors d'une quelconque activité. Mais encore, l'utilisateur peut connaitre la position GPS de ses amis pour d'éventuelles activités sportives collectives.

ANNEXE

Processus de compilation d'un code Android

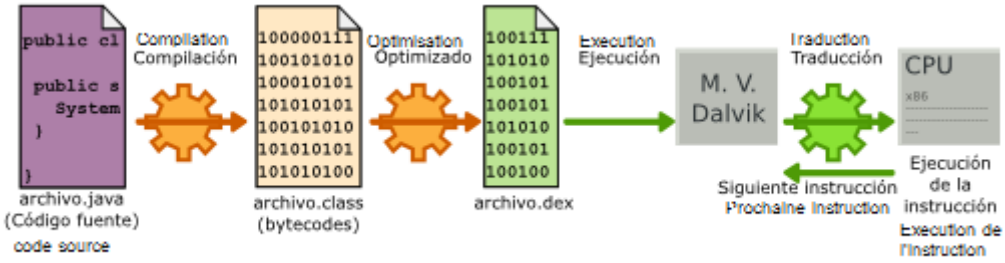


Figure 24 : Compilation Android (http://revistalinux.net/cursos/java-y-android/attachment/android-compilation)

Références et bibliographies

http://fr.wikipedia.org/wiki/Machine_virtuelle_Dalvik

http://developper.android.com

Android – A PROGRAMMER'S Guide [Jerome (J.F) DiMarzio] Mc Graw Hill

The Busy Coder's Guide to Android Develoment [Mark L.Murphy] COMMONSWARE

Developper avec les API Google Maps – Applications web, Iphone/Ipad et Android [Fabien Goblet _Michel Dirix_Loic Goblet_Jean-Philippe Moreux]