

ISPORTIVE

*Réalisation d'un programme pour Windows Phone 7
pour le suivi de l'activité physique*



Thibaud Cournollet et Samuel Fobis

2010 - 2011



ISPORTIVE

REALISATION D'UN PROGRAMME POUR WINDOWS PHONE 7
POUR LE SUIVI DE L'ACTIVITE PHYSIQUE

SOMMAIRE

1. Remerciements	2
2. Introduction.....	3
3. Contexte	4
4. Présentation synthétique.....	5
4.1 Objectifs	5
4.2 Environnement et technologies	5
5. Réalisations	8
5.1 Analyse de l'environnement.....	8
5.1.1 Contraintes	8
5.2 Réalisations fonctionnelles.....	9
5.3 Réalisations techniques.....	13
5.3.1 L'interface de développement.....	13
5.3.2 Le développement.....	15
5.3.3 L'implémentation des modules.....	15
5.4 Améliorations futures	18
6. Bilan Technique	19
7. Conclusion	20
8. Annexes	21

1. REMERCIEMENTS

Dans un premier temps, nous tenons à remercier notre responsable de projet pour l'INRA, Mademoiselle Sylvie ROUSSET.

Nous tenons aussi à remercier Monsieur Philippe Lacomme de l'ISIMA qui a participé au bon déroulement de notre mission grâce à son encadrement, son soutien et sa confiance.

2. INTRODUCTION

Durant notre troisième année de Licence Informatique à l'Université Blaise Pascal, nous avons effectué un projet pendant environ 50h de travail.

Il avait pour but, la réalisation d'un programme sur Smartphone capable d'inciter les utilisateurs à pratiquer une activité physique.

Nous avons décidé d'implémenter cette application sur le système d'exploitation mobile développé par Microsoft, Windows Phone 7 (WP7).

Pour se faire, il nous a fallu apprendre à maîtriser le langage de programmation C#.

Suite à cela, nous avons cherché à savoir quels détecteurs possédait notre téléphone et quelles précisions ceux-ci pouvaient avoir.

En effet, il était nécessaire pour que notre projet soit réalisable que l'on puisse analyser les mouvements et déplacements d'une personne très précisément.

Une fois la maîtrise de ces composants du téléphone nous avons pu chercher à traiter les données obtenues sur un serveur distant.

Dans un premier temps, après avoir présenté notre contexte de travail, nous commencerons par détailler chacun des objectifs que nous devons atteindre.

Puis nous parlerons des technologies que nous avons utilisées.

Nous passerons ensuite à l'explication détaillée du programme, de son analyse à sa conception.

Enfin, après avoir montré les améliorations possibles à ce projet, nous ferons un bilan de nos difficultés et des connaissances que nous avons acquises.

3. CONTEXTE

Dans un souci de clarté de notre démarche nous avons décidé de faire état du contexte de notre projet.

Monsieur Lacomme professeur à l'ISIMA et Sylvie Rousset qui travail pour l'INRA ont décidé de proposer comme projet, la réalisation d'un programme pour Android, Iphone et Windows Phone 7 pour le suivi de l'activité physique.

Nous avons donc mis en place, trois groupes de travail pour chacun de ces trois environnements.

Chaque semaine, de manière indépendante ou avec les autres groupes nous allons rendre compte de l'avancement de notre projet et établir de nouveaux objectifs.

Il était très important pour nous d'être en communication avec les autres groupes pour faire un seul et unique groupe de travail. En effet le but final de notre projet était d'avoir une démarche commune pour chacun des environnements.

Suite à cela, nous sommes allé présenter au CNRH les applications que nous avons mis en place afin d'étudier la faisabilité d'une étude de la dépense énergétique faite à l'aide de smartphone.

Lors de cette présentation des personnes non expertes en informatique étaient présentes et nous avons donc rendu accessible notre projet afin qu'il soit compris par tous.

Notre présentation a été bien accueillie et a suscitée un intérêt certain.

Voilà comment s'est déroulé notre projet nous allons maintenant vous présenter notre réalisation.

4. PRESENTATION SYNTHETIQUE

4.1 OBJECTIFS

Notre projet, avait pour but :

- 1) D'effectuer une collecte des données en rapport aux mouvements et déplacements d'une personne.
- 2) D'afficher les résultats de cette collecte sur un téléphone ainsi qu'un site internet.

Les objectifs fonctionnels que nous devons atteindre étaient donc :

- Identifier la position GPS du téléphone.
- Mesurer l'accélération du téléphone.
- Transmettre les données du téléphone vers un serveur distant.
- Créer un site internet permettant de consulter ces données.

4.2 ENVIRONNEMENT ET TECHNOLOGIES

Voici l'environnement général de notre projet :



Figure 1 : Représentation générale du projet

Comme vous pouvez le voir sur cette figure, une fois que le téléphone aura collecté les données de déplacement et d'accélération, il enverra par internet cette collecte à un serveur distant.

Celui-ci sera ensuite chargé de stocker ces données. Il hébergera un site sur lequel l'utilisateur pourra visualiser ses données via un ordinateur de bureau.

Afin de mettre en place ce réseau dans le projet il nous a fallu apprendre à utiliser plusieurs environnements de travail et technologies.

Dans un premier temps, nous devons procéder à la collecte des données sur le téléphone. Pour se faire, le choix des technologies nous a été imposé par l'environnement sur lequel nous devons développer l'application.

Technologies nécessaire au développement sur Windows Phone 7 :

- **Windows seven :**
Dernier système d'exploitation de la société Microsoft. Il succède Windows Vista. Il faut savoir que pour développer sur Windows Phone 7, il est nécessaire de travailler avec le logiciel Visual Studio 2010. Hors celui-ci n'est compatible qu'avec la dernière version du système d'exploitation de Microsoft.
- **Microsoft Visual Studio 2010 Express for Windows Phone :**
Microsoft Visual Studio Express est un ensemble d'environnements de développement intégrés (IDE) gratuits développé par Microsoft. Cet environnement est prévu pour le développement sur Windows Phone 7. Il possède un émulateur permettant de simuler l'intégration du programme au téléphone. De plus, il supporte comme langage de programmation le C#.
- **C# :**
Le C# est un langage de programmation, orienté objet, conçu par Microsoft pour la plateforme .NET (.NET Framework). C'est un descendant de C++ avec des caractéristiques de Java et plusieurs autres langages. Il se compile en langage intermédiaire, le MSIL (Microsoft Intermediate Language), et utilise une librairie multi-langages, le CLR (Common Language Runtime).
- **Windows Phone 7 :**
Windows Phone 7 (WP7) est la nouvelle plateforme de développement de Microsoft destinée aux smartphones.

Dans un second temps, nous avons dû mettre en place un site internet sur un serveur distant afin d'effectuer les tâches de collecte des données, de communication avec l'application et d'affichage des résultats.

La réalisation du site a été effectuée par un étudiant appartenant au groupe iPhone.

Pour notre part, nous avons tenté de proposer une nouvelle interface de ce site.

Technologies nécessaire au développement d'un site internet :

- **HTML, CSS :**

L'*Hypertext Markup Language* (HTML), est un format de données conçu pour représenter les pages web. C'est un langage de balisage qui permet d'écrire de l'hypertexte. Il permet également de structurer sémantiquement et de mettre en forme le contenu des pages.

Ce langage est interprété par les navigateurs actuels afin de représenter les pages web.

Le CSS (*Cascading Style Sheets*), est un langage qui sert à décrire la présentation des documents HTML et XML. Les standards définissant le CSS sont publiés par le World Wide Web Consortium (W3C).

- **PHP5 :**

Le PHP (*Hypertext Preprocessor*) est un langage de scripts libre principalement utilisé pour produire des pages Web dynamiques.

Un code PHP est interprété par le serveur qui va traiter et générer le code final de la page (constitué d'HTML ou de XHTML mais aussi souvent de CSS et de JS). Ensuite le navigateur interprétera chez le client le contenu renvoyé par le serveur.

- **Serveur PHP (WampServer 2) :**

Comme vous avez pu le voir ci-dessus, pour travailler en local avec des scripts PHP, nous avons dû mettre en place sur nos ordinateurs un serveur PHP capable d'interpréter le code.

C'est le rôle que va jouer le logiciel WampServer 2.

5. REALISATIONS

5.1 ANALYSE DE L'ENVIRONNEMENT

5.1.1 CONTRAINTES

Les contraintes que nous avons pu rencontrer dans ce projet étaient des contraintes matérielles.

En effet, un smartphone possède beaucoup moins de capacités qu'un ordinateur classique, tant sur la taille de l'espace mémoire, que sur la vitesse du processeur.

De plus, le smartphone est avant tout un appareil qui doit fonctionner en autonomie et donc à l'aide d'une batterie. Si l'application venait à en consommer une trop grande quantité, celle-ci elle deviendrait obsolète.

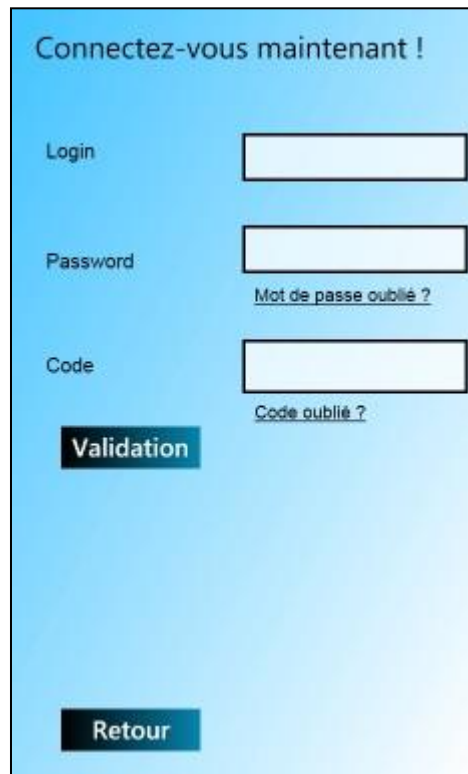
En conclusion dans notre travail il était important de faire en sorte de limiter les calculs, le stockage et les transferts de données via internet. C'est pourquoi nous avons fait le choix de laisser un serveur distant se charger de conserver et de traiter les données.

Les tâches que réalise le téléphone sont donc la collecte et l'envoi de données. Elles engendrent malgré tout, un coup non négligeable sur la batterie notamment à cause de l'utilisation du GPS et du transfert des données.

5.2 REALISATIONS FONCTIONNELLES

L'application que nous avons développée se décompose en plusieurs modules principaux dont nous allons brièvement décrire ici les différentes fonctionnalités.

- L'authentification :



Connectez-vous maintenant !

Login

Password
[Mot de passe oublié ?](#)

Code
[Code oublié ?](#)

Validation

Retour

Figure 2 : Ecran d'authentification

Cette page permet de vérifier que l'utilisateur est bien un utilisateur enregistré sur le site. Ses identifiants lui ont été envoyés au préalable lors de son inscription.

- Ecran d'accueil :



Figure 3 : Ecran d'accueil

Ici, après s'être identifié, l'utilisateur a la possibilité de naviguer entre les 4 principales fonctionnalités de l'application qui sont détaillées ci-après.

- Le suivi sportif :

Le suivi sportif, fait état des différentes activités physiques pratiquées et donne un point de vue positif ou négatif vis-à-vis de la dépense énergétique.

Ce module ne possède qu'une interface graphique statique, car le traitement de données collectées n'a pas été effectué.

- Les amis :

Permet l'affichage précis de la position des amis sur une carte. La gestion des amis est gérée via le site internet par l'utilisateur lui-même.

- Le mode Tracking :



Figure 4 : Ecran de tracking

Cette fonctionnalité permet de choisir au préalable l'activité pratiquée. Cela garantit une meilleure précision du calcul de la dépense.

Ici, une fois que l'utilisateur aura renseigné son activité, appuiera sur le bouton « start » au début de celle-ci et une fois terminé il appuiera sur le bouton « stop ».

Durant toute son activité le téléphone va enregistrer et envoyer les données collectées au serveur distant.

- La configuration :

Elle est décomposée en 2 sous-modules :

- **Personnelle :**

L'utilisateur peut gérer ici ses informations nominatives (nom, prénom, taille, âge, poids, login, mot de passe, ...).

- **Système :**

Cette sous-partie va garantir la pérennité de l'application. Il sera possible de changer l'adresse du site vers lequel le smartphone envoie les données, ainsi que de réinitialiser les paramètres.

Grâce à cela si le site venait à changer d'adresse il ne serait pas nécessaire de soumettre une nouvelle fois l'application pour qu'elle puisse fonctionner.



Figure 5 : Ecran de configuration Système

5.3 REALISATIONS TECHNIQUES

Dans cette partie, nous commenterons l'aspect technique de l'application qui se trouve derrière chacun des modules présentés ci-dessus. Nous commencerons par une analyse du logiciel que nous avons utilisé pour le développement de ce projet.

5.3.1 L'INTERFACE DE DEVELOPPEMENT

- Microsoft Visual Studio 2010 :
La prise en main de Visual Studio est assez rapide. C'est un logiciel bien conçu, qui offre quasiment tout les outils nécessaires à l'utilisateur pour créer ses premières applications sans pour autant avoir une parfaite maîtrise des langages utilisés.

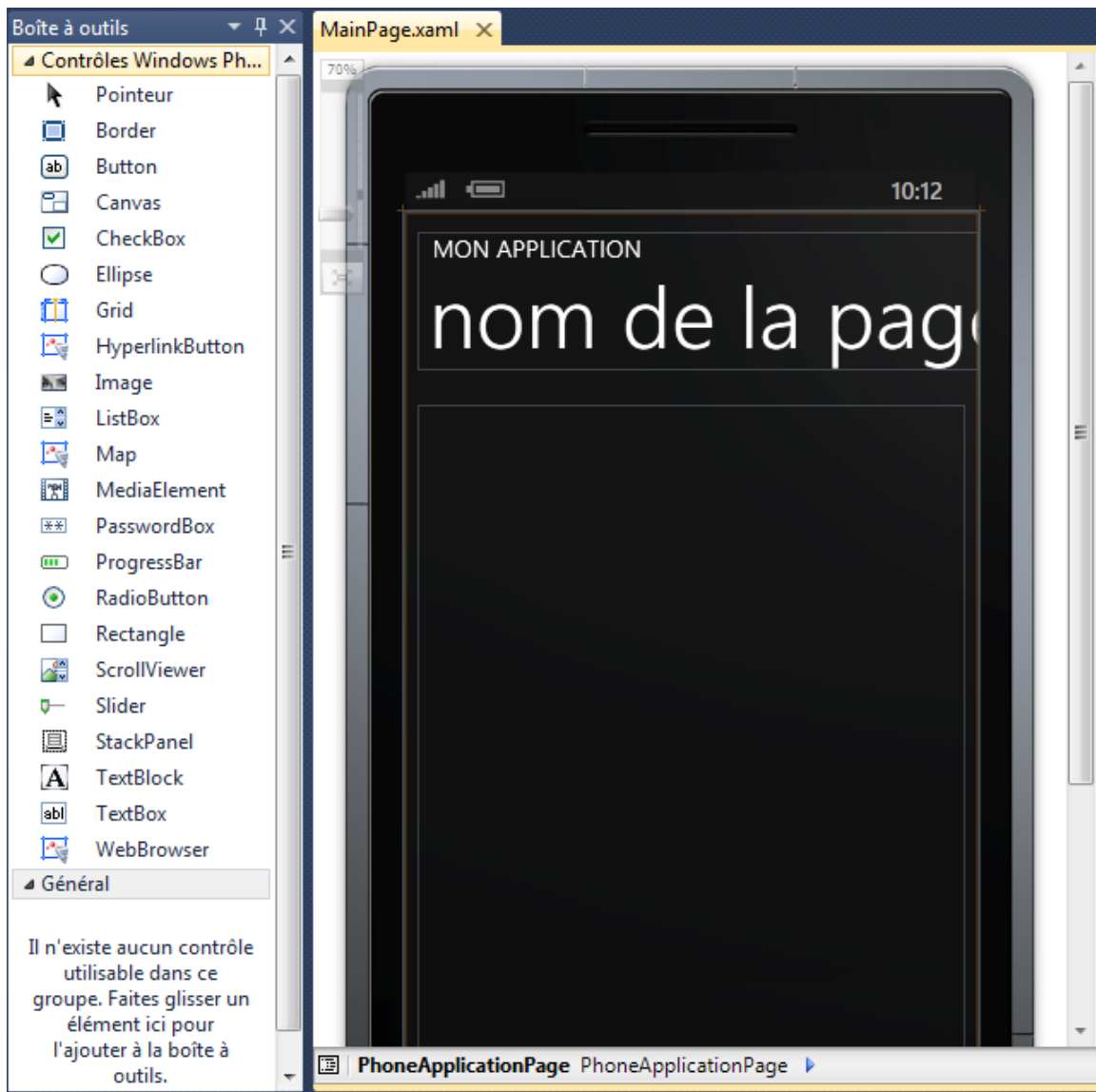


Figure 6 : Créer une interface

Comme vous pouvez le voir sur la figure, il est possible de réaliser une grande partie de l'interface en déplaçant les outils (boutons, images, ...) dans notre fenêtre du téléphone à l'aide de la souris.

Néanmoins, il est nécessaire d'apporter des modifications aux éléments générés par Visual Studio pour élaborer l'interface.

Pour se faire il faut comprendre ce que réalise Visual Studio quand on place un élément dans l'interface.

En effet, derrière cette utilisation graphique, Visual Studio va générer du code XAML. C'est ce même code qui une fois interprété va mettre en place l'interface graphique sur le téléphone.

Ce code XAML, correspond à du code XML bien spécifique qui permet l'instanciation à l'exécution d'objets issus des plateformes .Net.

```
<phone:PhoneApplicationPage
  x:Class="WindowsPhoneApplication1.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
  xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
  FontFamily="{StaticResource PhoneFontFamilyNormal}"
  FontSize="{StaticResource PhoneFontSizeNormal}"
  Foreground="{StaticResource PhoneForegroundBrush}"
  SupportedOrientations="Portrait" Orientation="Portrait"
  shell:SystemTray.IsVisible="True">
```

Figure 7 : Exemple de code XAML généré

Pour obtenir les modules présentés ci-dessus, il a été nécessaire de modifier ce code manuellement afin d'affiner et de personnaliser notre interface.

5.3.2 LE DEVELOPPEMENT

Ce n'est habituellement pas par l'interface, que le développement d'une application débute, mais Visual Studio nous force un peu la main. C'est en effet l'interface qui va nous servir de base.

Comme dit précédemment, la création d'une interface sous Visual Studio ne présente pas de réelle difficulté. Il suffit pour la plupart des cas d'un simple « Drag'n'Drop » d'un composant depuis la fenêtre d'outils vers le modèle graphique du smartphone présent, et le code XML correspondant est généré automatiquement.

Pour certains composants comme les boutons, la modification directe dans le code est nécessaire. La gestion des événements s'effectuera dans la partie C#, mais il faut tout de même attribuer une variable dans la partie XML pour modifier ces composants.

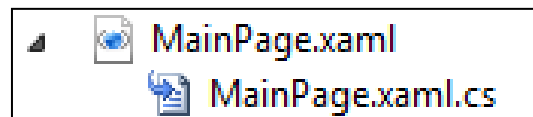


Figure 8 : Exemple d'Arborescence

Comme vous pouvez le voir, dans un projet Visual Studio pour chaque fichier ".xaml" correspond un fichier ".cs" qui va contenir le code C# permettant la gestion des événements de l'interface.

5.3.3 L'IMPLEMENTATION DES MODULES

- Détecteurs du téléphone :

Le but premier de notre application est de détecter lorsqu'un utilisateur se déplace. Il fallait donc pour cela pouvoir utiliser les différents outils mis à notre disposition dans le smartphone, c'est-à-dire, les capteurs d'accélération et le GPS.

```

watcher = new System.Device.Location.GeoCoordinateWatcher();
accelerometer = new Microsoft.Devices.Sensors.Accelerometer();
  
```

Figure 9 : Objets qui gèrent le GPS et l'accélération

Ces deux objets sont instanciés dans la classe « modeTracking.xaml.cs ». Ils possèdent des événements qui vont permettre de détecter la position du téléphone pour le « watcher » et l'accélération pour l'« accelerometer ».

- Enregistrement dans le téléphone :

Une fois que nous avons récupéré les valeurs rapportés par les détecteurs du téléphone, il nous a fallu les stocker avant de pouvoir effectuer le transfert de données vers le serveur.

Nous avons donc décidé d'utiliser des fichiers de sauvegardes comme des files. Les données sont stockées puis envoyées au fur et à mesure.

La classe qui est chargée d'effectuer la gestion des fichiers sur le téléphone dans notre application est une classe qui ne possède pas d'interface. Elle se nomme « GestionFichier.cs ».

```
// Permet d'avoir le dossier virtuel de l'application isolé de l'utilisateur
myStore = IsolatedStorageFile.GetUserStoreForApplication();

// On crée un nouveau dossier
myStore.CreateDirectory(nomDossier);
```

Figure 10 : Gestion des fichiers dans le téléphone

Comme vous pouvez le voir, afin d'avoir le chemin de stockage des fichiers propres aux applications sur le téléphone, il faut faire appel à une méthode « GetUserStoreForApplication() ». Une fois cet aspect maîtrisé, le reste de la gestion des fichiers est très classique. Avec des méthodes « read() » et « write() » pour la lecture et l'écriture des fichiers.

- Transfert de données :

Connecter notre application à internet était d'une grande importance, que ce soit pour authentifier l'utilisateur, pour l'envoi des données collectées sur le serveur, ou encore pour l'affichage des amis.

L'application devait donc pouvoir communiquer avec un serveur PHP. Quasiment tous ces processus utilisent la méthode GET. Ce que nous avons mis en place dans l'application est plutôt simple mais requiert de s'y attarder.

Voici comment notre application va communiquer avec le serveur.

1. Connexion du téléphone
2. Envoi d'une requête de méthode GET
3. Réponse du serveur HTTP
4. Le serveur ferme la connexion pour signaler la fin de la réponse.

La méthode GET est une méthode du protocole HTTP qui va permettre de faire passer des variables à une page web par l'intermédiaire de son adresse.

Nous allons utiliser l'exemple du transfert des données de mouvement pour aider à la compréhension.

Ces méthodes se trouvent à nouveau dans la classe « modeTracking.xaml.cs »

Dans un premier temps, on va générer une adresse de site, qui va contenir des variables.

```
string site = lien + "mobile/trackingWP7.php?getx=" + X + "&gety=" + Y + "&getz=" + Z;
HttpRequest req = (HttpRequest)HttpRequest.Create(site);
req.Method = "GET";
IAsyncResult token = req.BeginGetResponse(new AsyncCallback(GetStatusesCallBack), req);
```

Figure 11 : Méthode GET

C'est la phase de connexion du téléphone et d'envoi de la requête au serveur. Nous utilisons ensuite la méthode HttpRequest qui va nous servir à atteindre cette adresse.

La réponse du serveur va être récupéré dans la méthode « GetStatusesCallBack() ». Afin de recevoir une réponse asynchrone du serveur, cette méthode sera appelée dans un thread.

```
public void GetStatusesCallBack(IAsyncResult result)
{
    WebResponse response = ((HttpRequest)result.AsyncState).EndGetResponse(result);
    StreamReader reader = new StreamReader(response.GetResponseStream());
    string responseString = reader.ReadToEnd();
    reader.Close();
    response.Close();

    responseString = responseString + " " + enregistrement;
    Deployment.Current.Dispatcher.BeginInvoke(() => { textStatus.Text = responseString; });
    enregistrement++;
}
```

Figure 12 : Réponse du serveur

On transforme ensuite la réponse en string puis on va l'afficher. Pour se faire, on doit retourner dans le thread précédent.

Dans les spécifications, il nous a été demandé d'effectuer à l'aide de la méthode POST l'upload du fichier de sauvegarde vers le serveur distant.

Hors, bien qu'il soit possible d'utiliser la méthode POST pour envoyer un flux de bits, nous n'avons trouvé aucun moyen à l'heure actuel pour effectuer l'upload d'un fichier.

Habituellement en C# il existe une méthode « WebClient.UploadFile() » qui n'est pas présente pour windows phone.

5.4 AMELIORATIONS FUTURES

Nous allons maintenant discuter des améliorations auxquelles notre application doit être soumise et d'autres qui pourraient voir le jour.

- Améliorations primaires :

En effet, il ne faut pas oublier le but final de l'application qui est de calculer les dépenses énergétiques des usagers.

Pour le moment, elle permet de collecter les données, puis de les envoyer à un serveur.

Si l'étude des données collectées aboutissait à une évaluation assez précise des dépenses énergétiques de l'utilisateur, nous pourrions directement mettre en place un suivi dans l'application, consultable par l'utilisateur.

De plus, il pourrait donner lieu à un suivi effectué par des médecins qui émettraient leurs avis sur les dépenses des utilisateurs. Cet avis serait consultable via le site internet.

Une autre amélioration importante serait la détection automatique de l'activité effectuée par l'utilisateur.

En effet, à l'aide des données d'accélération du téléphone nous pensons qu'il serait possible de dire si un utilisateur marche ou cours, etc. Cela découlerait de l'étude approfondie sur des données préalablement enregistrées par le smartphone et permettrait ainsi aux usagers d'enregistrer leurs données sans se soucier de choisir ou non l'activité qu'ils pratiquent.

- Améliorations secondaires :

Des gadgets ont été pensés par les différents groupes de projet.

Une idée qui avait retenu l'attention était de mettre en place un IRC (discussion relayée par Internet) entre les différents utilisateurs.

L'idée d'une amélioration sur la recherche et l'affichage des amis a été abordée, mais finalement non implantée par manque de temps.

6. BILAN TECHNIQUE

Dès lors que nous arrivons au terme de ce projet et que nous pouvons regarder l'ensemble du travail réalisé, nous sommes fiers de la solution logicielle que nous avons développée et des difficultés surmontées.

Une des difficultés majeures à laquelle nous avons été confrontées, fut l'absence de support sur lequel tester notre programme. En effet, malgré l'émulateur de Visual Studio, certaine partie de l'application aurait nécessité un support physique pour effectuer des tests. Nous avons continué nos travaux sans certitude qu'ils soient corrects.

Visual Studio fut notre outil principal de développement, nous avons donc dû apprendre à nous en servir. Sa prise en main a été plutôt rapide, mais sa maîtrise demande du temps. Nous pouvons dire à ce jour que nous commençons à bien nous en servir.

Comme dans beaucoup de projet, certains langages nous ont été imposés. Nous avons passé du temps à voir ou revoir certains d'entre eux.

Techniquement, nous sommes restés bloqué quelques temps sur un problème, la communication entre le téléphone et le serveur. Nous avons dû analyser les différentes manières de réaliser cette connexion, de plus, Windows Phone 7 existant depuis peu de temps, la documentation était difficile d'accès. Ce fut finalement grâce à la méthode « HttpWebRequest » que ce problème fut résolu.

Enfin, à l'origine, il était prévu la création d'une seule application qui fonctionnerait sous 3 systèmes différents. Il nous a donc fallu travailler en coopération avec les autres groupes pour pouvoir s'accorder sur la mise en place d'un seul serveur qui serait utilisé pour communiquer avec les 3 applications.

7. CONCLUSION

Au final nous pouvons dire que nous sommes arrivés au terme des objectifs qui nous ont été fixés.

Ce projet nous a permis d'apprécier la réalisation d'une application de son analyse à sa conception.

Nous avons pu présenter notre travail à un public de non expert et ainsi réaliser son impacte.

Nos connaissances en JAVA nous ont été utiles pour la programmation en C#.

Nous avons pu approfondir des technologies de développement proposées par Microsoft ainsi que le développement web.

Le fait d'effectuer ce travail avec d'autres groupes nous a poussé à aller jusqu'au bout de nos objectifs.

Ce travail en équipe était très intéressant et n'était pas sans rappeler un le contexte professionnel dans lequel nous évoluerons.

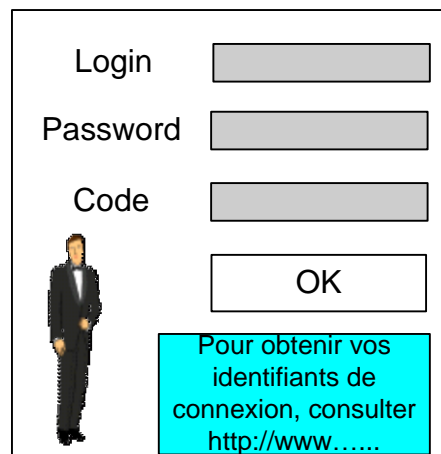
Enfin l'idée du projet est pour nous une grande réussite.

8. ANNEXES

Maquettes des spécifications :

Voici les maquettes proposées par Monsieur Lacomme. Nous avons essayé de les recréer au mieux.

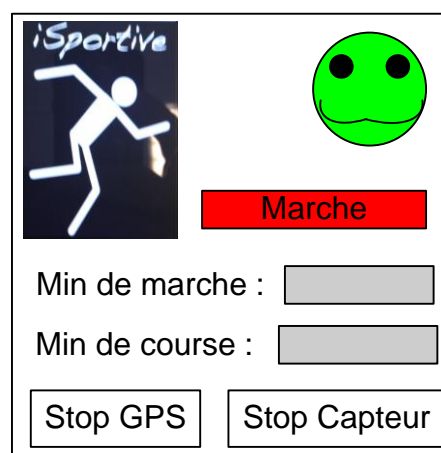
- Écran d'authentification:



Maquette de l'écran d'authentification. Elle contient :

- Champs de saisie pour "Login", "Password" et "Code".
- Un bouton "OK".
- Un petit pictogramme d'un homme en costume.
- Un message d'information en surbrillance : "Pour obtenir vos identifiants de connexion, consulter <http://www.....>".

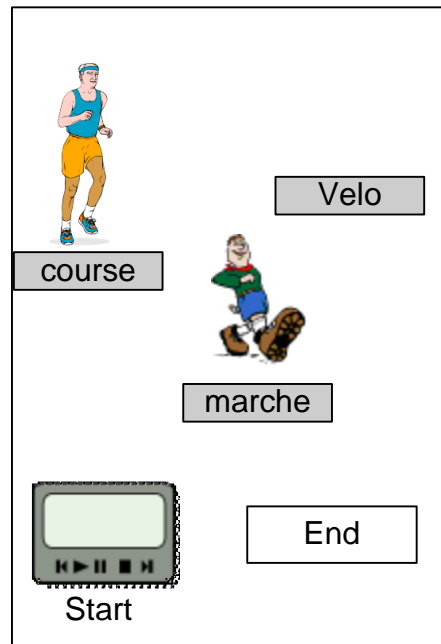
- Écran principal:



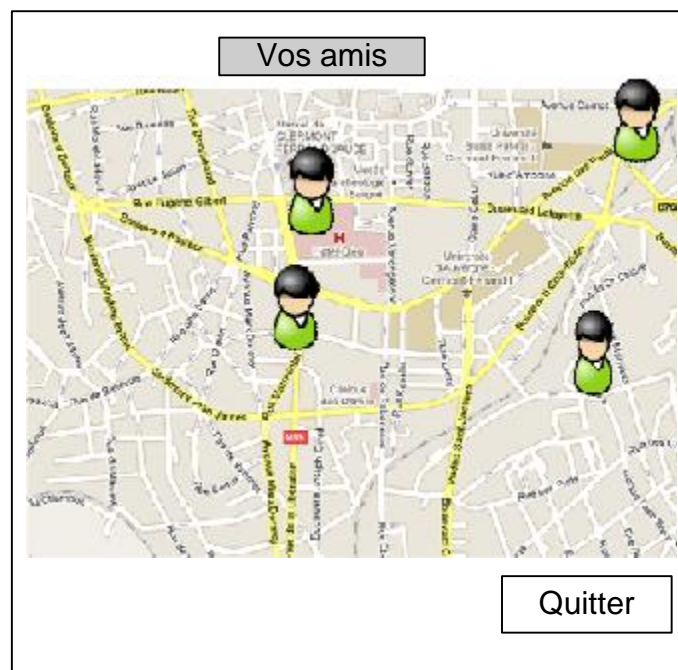
Maquette de l'écran principal. Elle contient :

- Le logo "iSportive" et un pictogramme d'un coureur.
- Un pictogramme d'un visage vert souriant.
- Un bouton "Marche" en surbrillance.
- Champs de saisie pour "Min de marche :" et "Min de course :".
- Deux boutons : "Stop GPS" et "Stop Capteur".

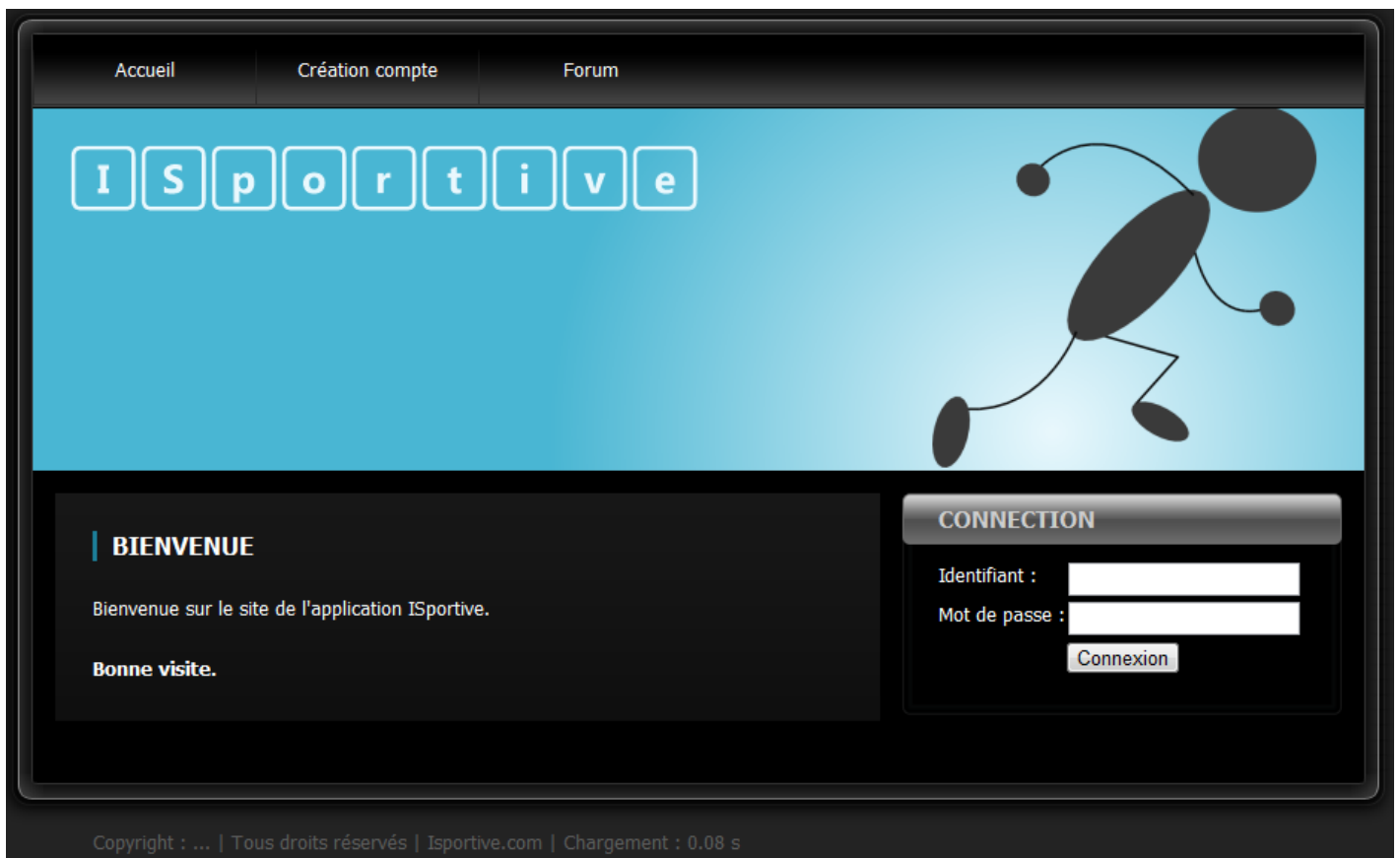
• Écran du mode Tracking:



• Écran des Amis:



Interface du serveur proposé :



Code de la réponse du serveur après authentification :

Voici la méthode qui va traiter la réponse du serveur après l'envoi des données d'authentification.

```

/// <summary>
/// Méthode appelé par un thread permettant de recevoir la réponse à la méthode GET
/// </summary>
/// <param name="result">Résultat de la réponse</param>
public void GetStatusesCallBack(IAsyncResult result)
{
    WebResponse response = ((HttpRequest)result.AsyncState).EndGetResponse(result);
    StreamReader reader = new StreamReader(response.GetResponseStream());
    string responseString = reader.ReadToEnd();
    reader.Close();
    response.Close();

    // Si identification ok
    if (responseString == "Id_OK")
    {
        Uri uri = new Uri("/ecranPrincipal.xaml", UriKind.Relative);
        Deployment.Current.Dispatcher.BeginInvoke(() =>
        {
            GestionFichier gestionFichier = MainPage.getGestionFichier();
            gestionFichier.setInfoConnection(log, mdp, num);

            this.NavigationService.Navigate(uri);
        });
    }
    else
    {
        Deployment.Current.Dispatcher.BeginInvoke(() =>
        {
            responseString = responseString.Replace(".", ".\n");
            textBlockErreur.Text = responseString;
        });
    }
}

```

Méthode de Lecture des fichiers du téléphone :

```
private string lire(string fichier)
{
    // Permet d'avoir le dossier virtuel de l'application
    IsolatedStorageFile myStore = IsolatedStorageFile.GetUserStoreForApplication();
    StreamReader readFile = null;
    string fileText = null;

    try
    {
        readFile = new StreamReader(new IsolatedStorageFileStream(nomDossier + "\\\" + fichier,
                                                                    FileMode.Open, myStore));

        while(!readFile.EndOfStream)
        {
            fileText += readFile.ReadLine();
            fileText += "\n";
        }
        readFile.Close();
        return fileText;
    }

    catch
    {
        fileText = erreur;
        return fileText;
    }
}
```