MICHEL Jérôme
BESSON Julien

L3 Info

Projet L3
Elaboration d'une application iPhone et d'un site Internet pour le suivi de l'activité physique.

Sommaire

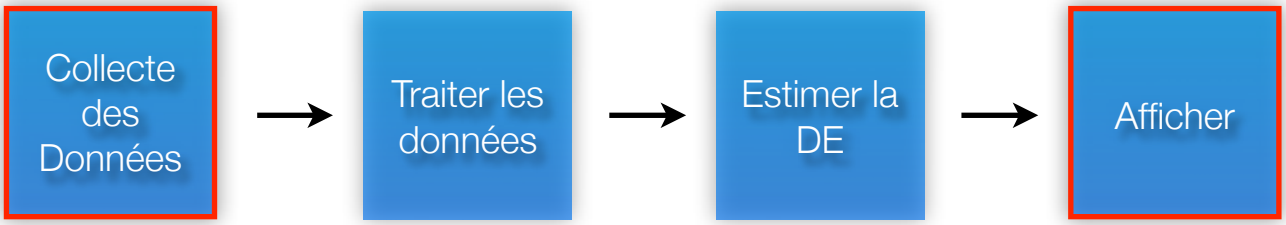
Introduction :
Explication de notre projet

Afin de bien définir le périmètre de notre sujet, partons du document présent sur l'ENT décrivant le sujet et de la première prise de contact avec le responsable de projet : M. Ph. LACOMME.

Sur l'ENT nous avions donc comme informations :

Il s'agit de réaliser un programme pour Android et Iphone permettant :
- *d'identifier la position GPS d'une personne,*
- *de localiser le téléphone sur une carte hébergée sur un serveur,*
- *de mesurer l'accélérométrie du téléphone,*
- *de faire un suivi de son activité sportive grâce au logiciel,*

Pour notre part, nous devions travailler sur la partie iPhone, ainsi que le site web. Après une première rencontre avec notre responsable de projet, nous avons pu éclaircir les choses et donc définir les grandes parties de notre projet, ainsi de ce savoir qui s'y joignait.



Nous pouvons donc définir l'intégralité du travail par ces 4 grandes parties, les deux éléments cerclés de rouge représentant les parties à réaliser par nos soins, et faisant partie de notre projet. Nous y trouvons donc :

- La collecte des données. Consistant à construire une application iPhone qui recueillera les données nécessaires.
- Le traitement des données. A faire par les mathématiciens afin d'avoir ou non la possibilité grâce aux capteurs du téléphone de définir le sport qu'est en train de faire la personne.
- L'estimation de la Dépense Énergétique (DE). A faire par les Biologistes / Experts en énergétique, c'est à dire, à partir d'un sport donné, arriver à estimer la dépense (en kJ, kCal) de la personne.
- Et enfin, l'affichage. Il fallait intégrer la possibilité d'afficher des informationsàa l'utilisateur, à partir d'un site internet ou du téléphone lui-même.

En plus de ces informations, plutôt théoriques, nous avons pu avoir des information plus précises concernant les utilisateur : Mme S. ROUSSET nous a appris que cette application était vouée à être utilisée par des personnes avec de forts problèmes de sédentarité, et donc de surpoids. Il nous fallait donc avoir une application accessible à toutes les tranches d'âge (jeunes, adolescents, adultes voire aussi personnes d'un certain âge). Il y avait donc un réel travail de fond à faire au niveau de l'interface.

Section 1 :
L'environnement de Travail.

Pour commencer à travailler, il nous fallait un environnement de travail qui nous permettrait de développer sur les deux plateformes que nous allions utiliser : Objectif-C pour l'iPhone et PhP pour le site Web.
Le PhP ne pose aucun souci, puisque nous allons travailler directement sous Notepad++ (éditeur de texte qui permet la colorisation suivant le langage utilisé). Donc aucune contrainte de ce côté-là.
L'Objectif-C ou plutôt la programmation sur iPhone impose deux grandes choses : un Mac, et XCode. Personnellement, Jérôme posséde un Mac, ce qui rend les choses plus faciles, mais nous avions aussi accès aux salles de Mac de l'ISIMA au cas où. Le second, XCode, était par chance, encore gratuit quand nous avons commencé notre projet en janvier 2011 (la version 4 sortira en mars 2011 pour un coût de 4,90 €).

C'est donc la partie iPhone qui nous imposera les contraintes de développement. Nous nous sommes donc réparti le travail ainsi : Jérôme s'occuperait de la partie iPhone et Julien de la partie Web, selon les préférences de chacun. A ce moment-là, nous n'avions aucune connaissance sur ces deux langages, il a donc fallu partir de zéro afin d'arriver au résultat final.

Au delà de ces contraintes, il en est apparu une plus tard dans le projet : transférer notre application de XCode à l'iPhone. Un compte développer gratuit chez Apple apporte le droit à télécharger XCode gratuitement, mais dès que l'on veut aller plus loin, les choses se gâtent. Pour pouvoir soumettre son application à l'iTunes Store de Apple, il faut avoir un compte payant ce qui es compréhensible, mais pour pouvoir transférer une application de XCode à un iPhone il faut aussi posséder ce compte payant (ce qui, après réflexion est compréhensible, car en trouvant les sources extraites d'un programme sur Internet il serait alors gratuit de l'installer sur son iPhone comme une application gratuite et ainsi contourner le système de paiement).
Nous avions donc XCode et un téléphone, mais au vu des difficultés à obtenir un compte développer (payant) chez Apple nous avons choisi une voie plus « simple ». Nous avons choisi de Jailbreaker l'iPhone pour pouvoir contourner le besoin d'un compte payant. Pour ce faire il suffit d'utiliser les outils (assez bien faits il faut le dire) de Jailbreak disponibles sur Internet et installer un petit outil sur le téléphone qui acceptera les applications de tous les ordinateurs.

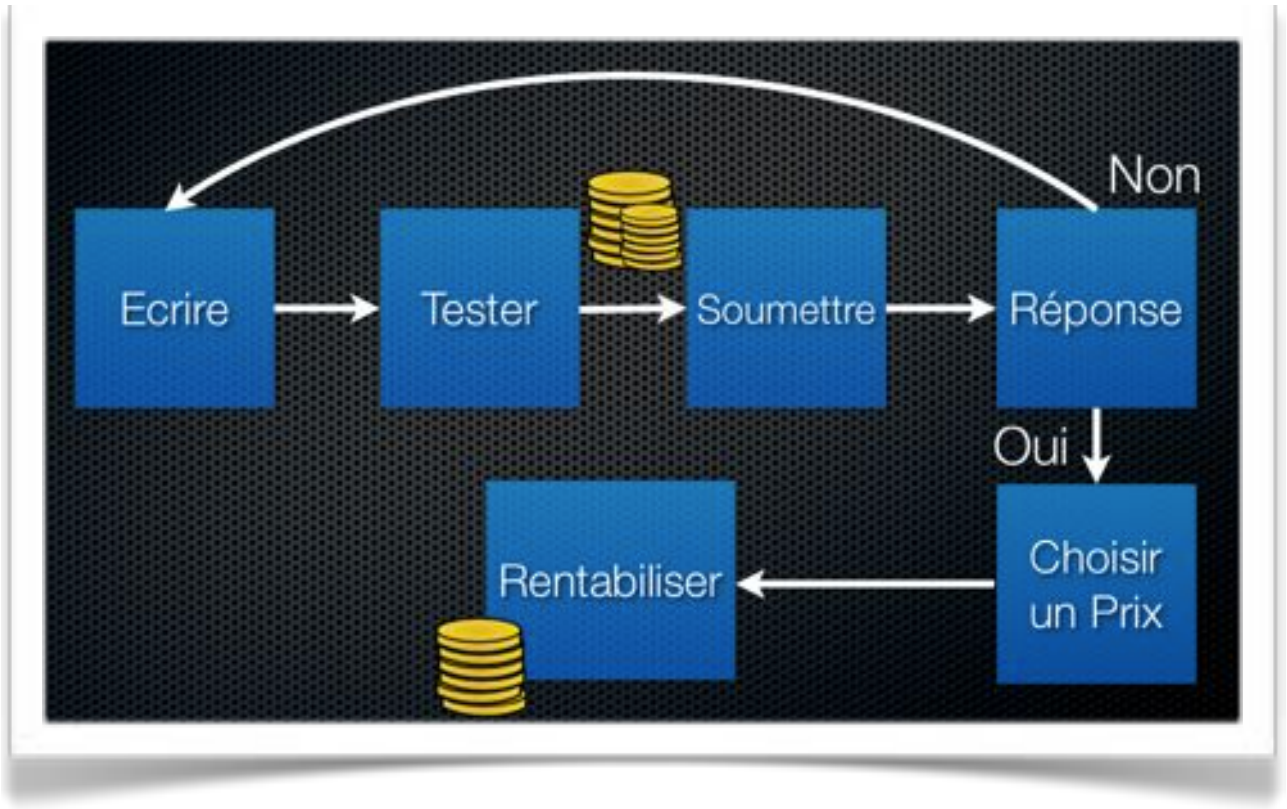
Dans XCode, nous avons accès à trois utilitaires qui nous permettront de créer au mieux notre application :

XCode est le logiciel de « Base » du développement d'un projet, il sert a poser les premières briques d'un projet, en créant des classes et fichiers prédéfinis en fonction de ce que l'on veux faire. Il sert aussi au développement du code à proprement parler avec une réelle interface IDE pour faciliter la vie du développer (tel que Eclipse ou NetBeans).

Interface Builder est le 2ème logiciel phare de notre projet, il permet de poser une interface sur le code. Spécialement mis à jour pour l'iPhone, il permet de gérer très simplement l'interface avec toutes ses composantes : boutons, listes, en passant par le clavier à utiliser, les éléments à cacher et permet aisément de faciliter la navigation par onglet, typique d'iOS.

Enfin, Instruments permet de tester l'efficacité de notre application, il intègre de nombreux instruments de mesure pour juger la mémoire, processeur ou encore les connexions réseaux et autres. Ce logiciel permet de juger à la fois un logiciel en simulateur (mémoire, processeur,...) ou sur un iPhone réel (de développement) pour juger les choses impossibles sur un simulateur (consommation électrique, réseau...).

De plus, même si les outils de développement sont gratuits, il nous semble intéressant de faire une petite mise au point technique concernant la mise à disposition d'une application iPhone.
Diffuser une application iPhone répond à une grande contrainte : l'App Store. Il est certes très pratique car on y trouve toutes les application du téléphone (impossible d'en installer autrement que par ce biais) mais impose que l'on envoie d'abord l'application à Apple qui la validera (ou non). Et bien sûr pour pouvoir accéder à cette mise en ligne un compte développeur payant est nécessaire (99€/an).



Ceci est fondamental quand on sait que chaque mise à Jour de l'application devra suivre ce trajet (assez long) et que si une seule chose ne va pas, il faudra tout reprendre.

D'ailleurs Apple impose certaines choses au niveau du codage d'une application comme la nécessitée d'utiliser une architecture MVC, ne pas animer la première vue etc. Nous avons donc fait l'effort d'essayer de coder comme Apple le veut pour à la fin arriver à une application qui puisse être publiée.

Concernant le site Internet, même si les outils n'étaient pas une préoccupation, il nous fallait quelque part où le stocker et donc forcément un serveur joignable à tout moment. Pour ce faire, nous l'avons temporairement stocké sur le serveur de l'ISIMA auquel nous avions accès grâce à M. LACOMME.

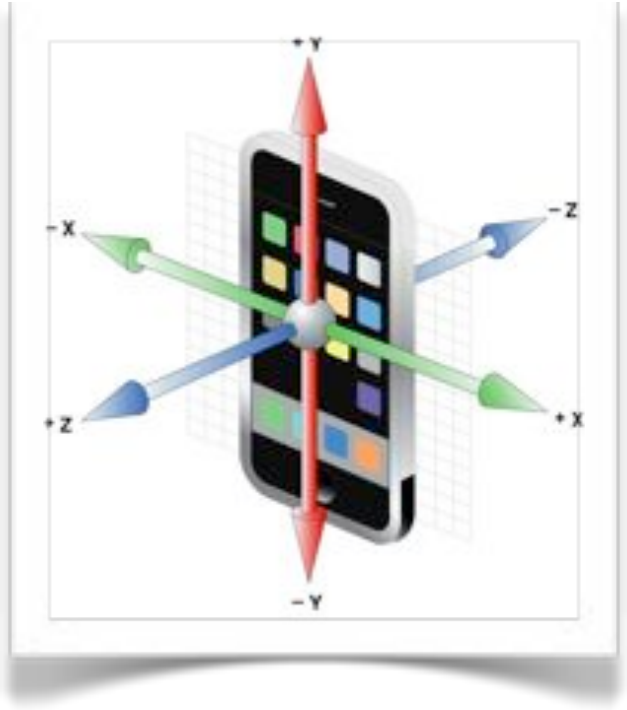
Section 2 :
Application « Capteurs »

La première chose à faire dans notre projet était de mettre en place un système qui permette la récolte des données du téléphone. Dans un iPhone il y a deux capteurs que nous pouvons mettre à contribution : un accéléromètre et un GPS. Le premier nous donnera des informations locales sur le positionnement du téléphone, et le second des informations globales sur le placement du téléphone sur la planète entière.

A ce moment de notre projet nos connaissances sur le sujet se résumaient à l'affichage de zones de texte ainsi que de variables a l'intérieur de celle-ci. C'est donc avec ces maigres connaissances que nous avons bâti la première application : « Capteurs ».

Comme on peux le constater, l'interface était réellement sommaire, mais correspondait à nos connaissances a ce moment là. Nous avons donc inclus deux parties principales, une pour le GPS et une pour l'accéléromètre.
La première affichait la Latitude, la Longitude, et l'Altitude. Nous nous sommes rendu compte plus tard que l'accès à l'altitude se faisait avec l'aide d'un serveur distant (chez Google) et renvoyait l'altitude du point correspondant au couple Altitude/Longitude donné. Nous avons donc remplacé plus loin cette partie par une mesure du nombre de rafraichissements du GPS pour connaitre la vitesse à laquelle l'iPhone pouvait déterminer la position de l'utilisateur.
La seconde affichait les accélérations sur les axes X, Y et Z. Il nous a fallu donc savoir à quoi correspondait ces valeurs, leur précision et bien sûr le positionnement des axes sur le téléphone. Les

valeurs retournées étaient affichées avec 10 chiffres après la virgule, mais après plusieurs tests, il nous est apparu que seuls 2 étaient réellement significatifs.
Pour le positionnement, nous avons trouvé un document qui permettait de les visionner directement :



Maintenant voyons comment nous avons procédé pour nous approprier le flux de données fourni par les deux capteurs, et leur affichage.

1. L'accéléromètre

L'accéléromètre est un capteurs qui n'a besoin que de peux de choses pour fournir les données. en fait ceci vient essentiellement du fait que ce dernier est déjà utilisé pour connaitre l'orientation de la vue a afficher (Portrait ou Paysage).
La récolte de ce flux se fait par l'intermédiaire de seulement trois lignes :

```
    UIAccelerometer *Accelerometre = [UIAccelerometer sharedAccelerometer];
    [Accelerometre setDelegate:self];
    Accelerometre.updateInterval = 10.0f/60.0f;
```

La première définit un objet qui s'avère être un périphérique partagé de type *UIAccelerometer* (accéléromètre), ceci permet de spécifier à l'iPhone que nous allons manipuler l'accéléromètre.
La deuxième permet de définir un délégué[] pour l'accéléromètre, qui est en vérité notre unique classe dans cette application, ceci afin de lui fournir les données.
La troisième permet de donner l'intervalle de temps minimal et maximal auquel on veut les données de l'accéléromètre.

Rien de bien compliqué donc pour l'accéléromètre, il suffit de définir un objet -physique- et la méthode appelée lors de la réception d'un message et les informations seront directement disponibles sous la forme de nombres flottants transmis via le message de l'accéléromètre.

2. Le GPS

Pour le GPS, l'implémentation se corse un peu, le principe reste le même au début : définition d'un objet physique et nomination de son délégué et méthode appelée ; mais après le traitement des données est un peu différent.

Tout d'abord il faut savoir qu'à l'instar de l'accéléromètre, le GPS a besoin d'un framework[] pour fonctionner. En base les outils pour le GPS ne sont pas inclus car ceux-ci ne sont pas utilisés par l'application sans demande du programmeur.

L'import du framework du GPS étant fait, il faut définir de la même manière que pour l'accéléromètre, un objet qui permette d'avoir accès aux informations :

```
    Localisateur = [[CLLocationManager alloc] init];
    [Localisateur setDelegate:self];
```

Dans ce cas, on ne définit pas un objet qui s'avère être physique. Ici on va définir un *CLLocationManager* qui va permettre de gérer le GPS. Ceci est de cette façon car au contraire de l'accéléromètre qui est tout le temps alimenté, le GPS lui n'est alimenté que quand on en a donné l'instruction au *CLLocationManager*.

Pour manipuler le GPS, nous avons donc ajouté un bouton qui commande l'activation et la désactivation du GPS. Ceci entrainera donc le début ou l'arrêt d'envoi de message via le délégué et donc la mise à jour des champs appropriés.

3. Interface

Pour les applications iPhone, il est possible de passer par deux façons, soit directement en ligne de commande, soit par Interface Builder. Dans notre cas, Interface Builder a été le meilleur choix vu sa simplicité de mise en oeuvre.
L'interface de cette application utilise essentiellement 2 types d'objets : les UILabel et les UITextField. Pour tout le texte affiché nous avons utilisé des UILabel. Pour les UITextField, nous les avons utilisé pour l'affichage des informations des capteurs.
Les UITextField ont la capacité à être gérés grâce à du code, avec une méthode, il est possible d'assigner du texte à un UITetField. En mettant le champ non éditable, cela permet de ne pas permettre à l'utilisateur de rentrer des données et à l'application d'afficher les données voulues.

Avec ces deux capteurs, et l'interface, nous avons pu avoir les premières briques de l'application.
Pour permettre l'utilisation de l'application en tant que collecteur de donnée il nous a fallu implanter l'envoi de données depuis l'iPhone vers le serveur de l'ISIMA où le site Internet est stocké temporairement.
Pour ce faire, nous avons besoin de deux choses : écrire dans un fichier texte les données reçues des capteurs et envoyer des fichiers via le réseau (Wifi ou 3G/EDGE).

4. La gestion des données

La gestion des données a amené une réflexion de fond concernant l'envoi de ces données au serveur.
Nous étions face à deux choix :

- Transmettre directement les données à chaque mesure au serveur
- Créer un fichier et l'envoyer à la fin des mesures

Les deux n'étaient pas sans contraintes. Le premier choix nous faisait passer directement par la méthode GET[] en PhP, directement implémentée dans les outils de Apple. Mais le problème était le réseau, les temps de latence comme la consommation batterie ajoutée dus à l'activation des puces réseau.
Le deuxième choix avait un énorme désavantage, l'envoi de données par internet via une page PhP ne passe que par le protocole POST[] mais celui-ci n'était pas directement implémenté dans les outils d'Apple.
Après avoir pesé le pour et le contre, notre choix s'est donc posé sur la deuxième solution et il nous a donc fallu arriver à stocker les informations sur le téléphone et les envoyer dans un fichier à travers le réseau.

Tout d'abord l'écriture des fichiers, celle-ci peut se faire de plusieurs façon selon ce que l'on veut stocker. Il existe par exemple une structure *plist* qui pourrait être assimilée à un XML mais dont les fonctions ont été grandement simplifiés. Mais dans notre cas il s'agissait de données brutes et il était nécessaire que le serveur puisse les traiter le plus simplement possible.
Vu les contraintes, nous avons opté pour un système de fichiers texte simple car aucune donnée modifiée ne serait retouchée par le téléphone. Tout ceci (création du fichier et écriture) se fait grâce à deux objets : *NSFileManager* pour créer/supprimer les fichiers et un *NSFileHandle* pour l'écriture dans celui-ci. L'utilisation de ces deux en parallèle nous a permis de faire ce que nous voulions : remplir un fichier texte à intervalles réguliers21 malgré une utilisation processeur assez élevé (surtout quand l'on approche les 0,1s entre chaque écriture).

La chose qu'il nous a fallu appréhender aussi pour l'écriture de fichiers est les Timers. Ce sont pour faire simple des minuteurs qui déclenchent une action toutes les x secondes avec un système de message comme décrit ci-dessus. La classe *NSTimer* s'avère extrêmement utile et facile à utiliser quand l'on connait le temps entre chaque action et définir par exemple un temps d'attente. Ceci nous a permis de faire l'écriture dans un fichier toutes les x secondes et de laisser l'application tourner ce processus en second plan sans avoir à toucher aux threads plus complexes à utiliser.

La deuxième chose concerne l'envoi de ce fichier. La méthode POST n'étant pas implémentée il nous a fallu construire directement le paquet à envoyer à la main et ensuite le faire transiter par le réseau ce qui nous a pris à vrai dire beaucoup de temps. Il nous a fallu comprendre le système de bundle (paquet) et comment il était construit pour le faire à notre façon. Cette fonction s'avère en fait tenir en une dizaine de lignes malgré sa difficulté à appréhender, et permet des échanges réseaux assez rapide que l'on peut manipuler comme bon nous semble.

5. A quoi cela sert ?

Bien que rudimentaire, cette application nous a servi tout au long du projet, car c'est celle-ci qui nous a permis de récolter les valeurs des capteurs pour la marche, la course, le repos d'un individus. Et ce sera d'ailleurs la seule version, car bien que les autres applications récolteront les données, il n'est d'aucune utilité de les garder sur le téléphone et de les récolter de cette façon.

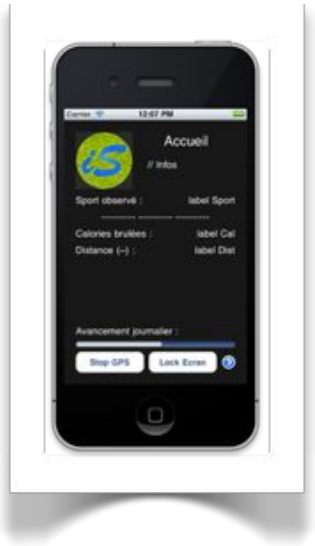
Section 3 :
Application « iSportive »

Il nous fallait maintenant créer une application que les utilisateur pourraient manipuler. Il fallait donc penser à l'architecture générale, ainsi qu'à l'évolution du code très probable et donc adopter un schéma de développement modulable. Mais avant ça, nous avons fait de petites recherches pour savoir comment notre application se nommerait. Nous avions pensé à iSport mais déjà pris dans le iTunes Store, donc nous avons fait le choix de iSportive.

Du coté de l'architecture, nous avons donc développé une application suivant le schéma MVC (Model-View-Controller) qui est d'ailleurs la technique préconisée par Apple pour que l'application soit validée. Ceci nous faciliterait le travail futur et permettrait une très grande modularité dans le développement, et surtout la possibilité de varier les vues si par exemple nous voulions faire une version iPhone et iPad.

L'application se composait donc d'un menu, avec 4 boutons qui permettaient d'accéder aux quatre parties de l'application à savoir : l'accueil, le mode tracking[], les amis et la configuration.

1. L'Accueil



L'idée de départ du projet était qu'à l'aide des différents capteurs, le téléphone devienne « intelligent » et trouve tout seul le sport exercé par la personne pour en déduire sa dépense énergétique. C'est dans cette partie que nous voulions implanter cette fonctionnalité, mais c'était aussi la dernière partie de notre travail car il fallait que nous travaillions avec les mathématiciens pour savoir comment trouver le sport exercé, et que les biologistes nous fournissent des formules pour le calcul de la dépense énergétique.
C'est pour cela que cette partie restera vide dans cette version de l'application.

2. Le mode Tracking

Au début du développement, le mode tracking était un substitut au mode « intelligent ». Ce mode a amené de nombreuses et longues réflexions sur sa réelle utilité, et son utilisation par les utilisateurs. Au départ, nous avions en tête que ce mode serait un substitut au mode « intelligent » jusqu'au moment où une solution pour détecter le sport serait trouvé. Mais ceci impliquait que nous pouvions arriver à détecter tous les sports. Nous avions donc trois possibilités :

- Le mode tracking sera enlevé de la version finale, le mode intelligent pourra détecter tous les sports possibles.
- Le mode tracking sera enlevé de la version finale, le mode intelligent classera les sports en classes qui permettront de détecter tous les sports mais une dépense énergétique moins précise.
- Le mode tracking restera en mixte avec le mode intelligent. Le mode intelligent permettra de détecter les sports de tous les jours et le mode tracking les différents sports dont l'évaluation de la dépense énergétique serait disponible.

Après réflexion et différents tests, il nous semblait impossible de déterminer tous les sports possibles. Il fallait garder dans l'esprit que les personnes susceptibles d'utiliser l'application sont des personnes atteintes de surcharge pondérale, donc il nous semblait plus plausible d'utiliser la troisième proposition. Les sports a détecter seraient donc les sports de tous les jours : repos, marche, course, montée d'escalier. En dehors de ceci, le mode tracking nous permettrait de détecter plus précisément le sport et donc la dépense énergétique. Certes quand la personne ferait du sport, elle serait obligée de fournir les informations au téléphone, mais vu que le mode intelligent ne sera (à notre avis) pas capable de détecter tous les sports, le mode mixte est très avantageux.

3. Le mode Configuration

Le mode configuration permet à l'utilisateur de modifier son compte et l'adresse sur lequel l'iPhone ira chercher les script PhP.

Pour faire au mieux, nous avons à ce moment du développement, remanié intégralement l'application (possible grâce à l'utilisation de l'architecture MVC) pour intégrer dans toutes les parties de celle-ci un système de sauvegarde pour remettre les informations à zéro chaque jour, et garder en mémoire la

dépense journalière. De plus, pour les échanges avec le serveur un Login est obligatoire donc la sauvegarde de ces informations était essentielle pour éviter que l'utilisateur n'ait à les rentrer à chaque lancement de l'application.

Mais l'utilité première de ce mode est de pouvoir modifier l'adresse du serveur : si jamais le serveur change d'adresse, il serait plus facile d'informer tous les utilisateur afin qu'ils mettent l'adresse du serveur à jour que de mettre à jour l'application en la re-proposant à Apple et repasser par tout le processus décrit dans l'introduction. Ce mode a donc été implémenté pour la pérennité de l'application.

4. Le mode Amis

Pendant la première rencontre, avec Mme S. ROUSSET, il est ressorti que le plus important dans l'application est l'interaction sociale entre les personnes. Nous avons donc mis en place un système d'amis (comme sur Facebook). Sur le serveur il est donc possible de demander quelqu'un en ami. Sur l'iPhone, nous avons donc donné la possibilité de voir ces amis sur une carte afin de connaitre leur emplacement. Il a donc fallu mettre en place un système qui toutes les 30 minutes enverrait la localisation du téléphone au serveur, et donc les amis seraient au courant de l'emplacement de la personne. Bien sûr il nous a donc fallu incorporer la possibilité de couper le GPS pour les personnes ne voulant pas être localisées.

Nous avons donc mis en place un système de carte (qui utilise la map de Google) et qui permet après implémentation des méthodes adéquates de fournir une carte à l'utilisateur, et permet au développeur de poser des épingles sur celle-ci.
De plus nous avons simplifié l'échange de données avec le serveur. Au départ nous voulions partir sur une base de XML, certes pratique mais très lourd lors de l'échange de 5 à 10 contacts (sachant qu'il était prévu que l'application soit utilisée par environs 50 personnes). Nous avons donc opté pour un système d'échange de texte directement et de découpage de ce dernier, cela avait l'énorme avantage d'être très léger en termes d'échange réseau.

5. Conclusion sur la première version

Dans cette première version, nous avons pu voir appréhender l'interface utilisateur et les échanges de messages avec le serveur. Avec cette version, nous sommes arrivés à tout ce qui était demandé et les 4 parties fonctionnelles. Mais un problème se posait : la gestion des différentes parties se faisait séparément, or l'application nécessitait des tâches de fond comme l'envoi de la localisation ou encore les calculs permettant de deviner le sport de la personne (même s'ils ne sont pas encore au point il faut y penser lors de l'élaboration de l'application).

Cette version bien que fonctionnelle s'avère assez peu pratique à utiliser tous les jours, ceci amena donc une deuxième version qui amène une refonte complète de l'interface pour quelque chose de plus proche des attributs d'une application iPhone conçue comme celles d'Apple. Et surtout qui permette le calcul en arrière plan et un réel lien entre les parties.

Section 4 :
Application « iSportive 2 »

Dans la version précédente, nous nous sommes essentiellement concentrés sur la partie « Model » qui nous a permis de mettre en place plusieurs choses dont un système d'échange avec le serveur, de sauvegarde des préférences de l'utilisateur ainsi que la manipulation de nombreux objets (carte Google pour les amis, GPS, Accéléromètre ...).

1. Une nouvelle version, mais pourquoi ?

Tout d'abord comme nous l'avons dit, il nous fallait une interface plus accueillante pour l'utilisateur que la première version. Nous voulions aussi une interface plus « typique » d'Apple avec un système d'onglets, et l'utilisation des outils que fournit Apple pour que l'application soit la plus fluide possible. De plus, la première version nous a posé d'énormes problèmes lorsque nous avions voulu intégrer le système qui toutes les 30 minutes enverrait la position de la personne au serveur. Les différentes vues ne communiquant pas, et les Timers étant difficiles à utiliser dans des fichier de Model, le problème n'était pas simple. Le système d'onglets a l'avantage de charger tous les onglets lorsque l'un d'entre eux est affiché, et de laisser tourner en arrière plan certaines méthodes.
Et enfin ce système d'onglets est simple à utiliser car ressemble à l'application téléphone de l'iPhone.

La nouvelle interface ressemble donc à ceci :

(L'accueil a été remplacé par un mode « Cohésion » car le mot accueil n'avait pas réellement de sens dans ce contexte)

2. Pas de nouveautés à part l'interface ?

A ce stade du développement, il nous restait environ 15 jours pour plancher sur quelques fonctionnalités supplémentaires que nous voulions implanter. Et la première venait d'un défaut majeur des iPhone.

2.1. Le LockScreen

Nous devions faire face à un défaut de l'iPhone pour les développeurs (mais avantage pour les utilisateurs) : pour préserver la batterie, quand l'iPhone se met en veille, il coupe tout calcul d'une application autre que celle de base (SMS, Téléphone, Mail). Même si un moyen doit bien exister, nos connaissances ne nous permettent pas de faire ceci « simplement ». Dans l'optique que l'application reste dans la poche de l'utilisateur et tourne toute la journée, cette contre-indication était de taille.

Nous avons donc implanté notre propre Lock-Screen qui désactiverait la veille de l'iPhone et permettrait de faire tous les calculs en arrière plan. De plus ce système nous permettrait d'afficher ce que l'on veut sur l'écran comme la dépense énergétique ou la position etc.
Pour ce faire, nous avons intégré un jeu de doubles boutons qui poussés simultanément réactiveraient l'application (et ne se réactiveraient pas dans une poche par exemple).

2.2. Le lien Social

La seconde chose que nous voulions était d'aller plus loin dans le lien social que nous avions instauré avec le système d'amis. Voir ces amis sur une carte c'est bien, leur parler, c'est mieux. De plus ces gens font peut-être parti du même programme mais ils ne se connaissent pas forcément assez pour s'échanger leur numéro de téléphone ou autre. Un système tiers, propre au programme leur permettrait d'échanger sans pour autant dévoiler leur vie à chacun. Nous avons donc planché sur le sujet, instauré un système de Login/pass au lieu du Nom/pass, et finalement arrivé au résultat : un système de messagerie.

Nous avons donc instauré un système de messagerie propre a notre application et géré par le serveur lui-même. Mais pas quelque chose de standard comme un serveur mail, bel et bien un système hybride entre le SMS et l'eMail de notre création. Ceci était d'autant plus dur qu'il fallait utiliser les technologies que nous avions jugé trop lourdes auparavant : le XML, le système de Liste sur l'iPhone, et de relevé automatique.

Nous avons pu avoir un système opérationnel au sein du site internet, mais sur l'iPhone, tout était implanté dans l'interface ou le Model, il nous manquait un peu de temps pour faire le lien entre les deux. C'est d'ailleurs sur ceci qu'ont porté nos dernier travaux sur ce projet.

3. En résumé

Dans cette nouvelle version, nous avons donc instauré une nouvelle interface plus conviviale et surtout plus intuitive pour l'utilisateur, mais aussi rajouté des petites améliorations en vue des versions futures et surtout une grosse fonction, celle de Messagerie entre amis. Le lien social si important entre les gens était donc renforcé.

Section 5 :
Le Serveur, technologies et logiciels

1. Le Serveur

Comme expliqué dans la partie précédente, un serveur est obligatoire pour ce genre de projet. En effet c'est lui qui administrera et stockera les données que les téléphones enverront. Je vais donc essayé de vous expliquer un peu comment a été créé ce serveur en parlant dans une première partie de technologies et logiciels employées.

Dans une seconde partie, je parlerai plus précisément du site web présent sur le serveur qui servira d'interface pour gérer les données présentes sur ce serveur. A savoir les différentes fonctions que le site possède et permet de faire, et j'expliquerais brièvement quelques soucis de sécurité.

Enfin dans une dernière partie, j'expliquerai les différents scripts que le serveur possède pour interagir avec les téléphones.

1.1. Technologies et Logiciels

1.1.1.Les Technologies

Les langages HTML et XHTML 1.0

L'Hypertext Markup Language, généralement abrégé HTML, est le format de données conçu pour représenter les pages web. Il permet notamment d'implémenter de l'hypertexte dans le contenu des pages et repose sur un langage de balisage, d'où son nom. HTML permet aussi de structurer sémantiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie, et des éléments programmables tels que des applets. Il permet de créer des documents interopérables avec des équipements très variés de manières conformes aux exigences de l'accessibilité du web. Le langage en est aujourd'hui à sa version 5.0.

XHTML 1.0 est une simple reformulation de HTML 4.0 en application XML 1.0. Il s'agit donc uniquement d'un changement de syntaxe, aucune fonctionnalité n'ayant été ajoutée ou retirée. XHTML 1.0 tend à remplacer HTML 4.0 en raison d'une syntaxe plus stricte et de sa compatibilité avec les langages XML. C'est pourquoi nous avons choisi d'utiliser ce langage.

Les feuilles de style CSS

Le langage CSS (Cascading Style Sheet : feuilles de style en cascade) sert à décrire la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le consortium W3C. Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien supporté par les navigateurs web dans les années 2000.

L'un des objectifs majeurs de CSS est de permettre la stylisation hors des documents. Il est préférable de ne décrire que la structuration d'un document en XHTML, et de décrire toute la présentation dans une feuille de Style séparée (fichier .css annexe). Les styles sont appliqués au dernier moment, dans le navigateur web des visiteurs qui consultent le document. Cette séparation fournit un certain nombre de bénéfices, permettant d'améliorer l'accessibilité, de changer plus facilement de structure et de présentation, et de réduire la complexité de l'architecture d'un document.

Le langage PHP

PHP (Hypertext Prepocessor) est un langage de script interprété, gratuit, OpenSource et distribué sous une licence autorisant la modification et la redistribution. PHP est supporté sur plusieurs systèmes d'exploitation.

D'un point de vue exécution, PHP a besoin d'un serveur Web pour fonctionner. Toutes les pages demandées par un client seront reconstruites par le serveur Web, en fonction des paramètres transmis, avant d'être retournées au client. Le schéma ci-dessous illustre le principe de fonctionnement de PHP :

Une des forces du langage PHP est sa richesse en termes de fonctionnalités. En effet, il dispose à l'origine de plus de 3 000 fonctions natives prêtres à l'emploi garantissant aux développeurs de s'affranchirent de temps de développement supplémentaires et parfois fastidieux. Ces fonctions permettent entre de traiter les chaines de caractères, d'opérer mathématiquement sur des nombres, de convertir des dates, de se connecter à un système de base de données, de manipuler des fichiers présents sur le serveur …

Le langage JavaScript

JavaScript est un langage de programmation de scripts principalement utilisé pour les pages web interactives. Le code est directement écrit dans la page HTML, nous n'avons donc aucune confidentialité au niveau de ce code (il est en effet visible par l'utilisateur s'il souhaite le voir).

Ce code qui est intégré directement dans les pages web est exécuté sur le poste client. C'est alors le navigateur qui prend en charge l'exécution de ces scripts. Généralement, JavaScript sert à contrôler les données saisies dans des formulaires HTML, ou à interagir avec le document HTML via l'interface DOM, fournie par le navigateur.

A noter que le JavaScript peut être désactivé par certains utilisateurs, notamment pour des raisons de sécurité. Il faut donc veiller à ce que le site reste accessible et que les contrôles de données soit également effectués en PHP lors de la désactivation de celui-ci.

MySQL

MySQL est un système de gestion de base de données (SGBD). Il fait partie des SGBD les plus utilisés au monde, autant par le grand public (applications web principalement) que par des professionnels, au même titre que Oracle ou Microsoft SQL Server. Il est multi-thread et multi-utilisateurs.

C'est un logiciel libre développé sous double licence en fonction de l'utilisation qui en est faite (produit libre ou produit propriétaire). Dans ce dernier cas, la licence est payante, sinon c'est LGPL qui s'applique. Ce type de licence double est utilisé par d'autres produits.

1.1.2.Les Logiciels utilisés

WAMP Server 2.1

WAMP signifie Windows Apache Mysql PhP. Les rôles de ces quatre composants sont les suivants:

- Apache est le serveur web « frontal », il est devant tout les autres et répond directement aux requêtes du client web (navigateur)
- Le langage de script PHP sert la logique
- MySQL est le SGBD gérant nos données stockées

• Windows assure l'attribution des ressources à ces trois composants (le System d'exploitation, il existe aussi LAMP fonctionnant sur Linux et MAMP pour Macintosh)

WAMP contient donc tous les éléments permettant de développer localement une application web. Il inclut également phpMyAdmin, qui est une interface conviviale, gratuite réalisée en langage PHP pour le SGBD MySQL afin de faciliter la gestion des bases de données sur le server, et est distribué sous licence GNU GPL.

Il s'agit de l'une des plus célèbre interfaces pour gérer une base de données MySQL sur un serveur PHP. De nombreux hébergeurs le proposent, ce qui permet à l'utilisateur de ne pas avoir à l'installer.

Cette interface pratique permet d'exécuter, très facilement de nombreuses requêtes comme la création de tables de données, des insertions, des mis à jours, des suppressions ou encore des modifications de structure de base de données. Ce système permet également de sauvegarder une base de données sous forme de fichier .sql et ainsi transférer facilement ses données. De plus celui-ci accepte la formulation de requête SQL directement en langage SQL, cela permet de tester ses requêtes directement et ainsi gagner un temps précieux.

Notepad++

Pour coder des scripts PHP, JavaScript ou encore des pages HTML, un simple éditeur de texte de type Bloc-notes ou WordPad suffit. Cependant pour plus de confort, Notepad++ a été utilisé. En effet cet éditeur de code source, en plus de supporter plusieurs langages, possède plusieurs avantages :

Une coloration syntaxique modifiable selon les envies et habitudes.

Un système de plugin mis a disposition permettant à l'utilisateur de télécharger et d'installer des options spécifiques (comme une auto complétion de balise html, une validation du document XML, etc…)

D'autres petites fonctions bien utiles comme le Replace dans tout le document, l'explorateur de fichier intégré, le système d'onglet …

JQuery

Il s'agit d'une bibliothèque JavaScript libre permettant une interaction antre le JavaScript et l'Html à l'aide de commandes simplifiées. Elle a notamment été utilisé afin de réaliser un captcha dans le formulaire d'inscription afin d'éviter aux robots de créer des comptes automatiquement. A l'aide d'un drag and drop, l'utilisateur fait coulisser le bouton dans la barre afin de débloqué le bouton final de création de compte.

Pour utiliser cette librairie, il faut spécifier sont emplacement dans la page HTML à l'aide du code suivant :

```
<script type="text/javascript" src="js/captcha/jquery.js"></script>
<script type="text/javascript" src="js/captcha/jquery-ui.js"></script>
<script type="text/javascript" src="js/captcha/QapTcha.jquery.js"></script>
```

L'initialisation ce fait à l'aide de ces quelques lignes (il faut juste avoir une div possédant un id ="QapTcha") :

```
<script type="text/javascript">
		$(document).ready(function(){
			$('#QapTcha').QapTcha({disabledSubmit:true});
		});
</script>
```

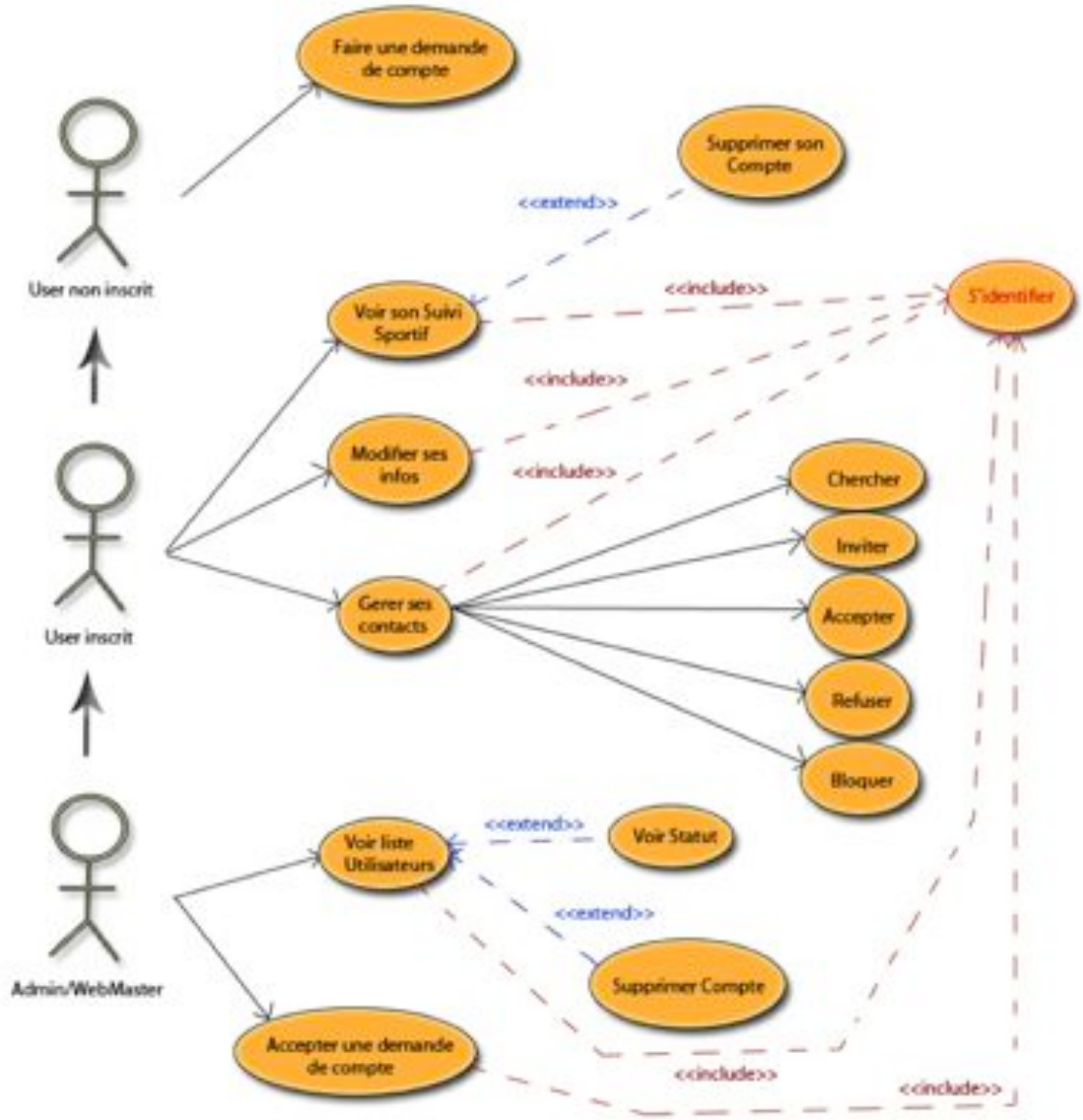
Google maps

L'utilisation de l'API GoogleMaps nous servira afin de montrer la localisation de ses amis (que ce soit sur téléphone ou sur le site web). Cette API possède une multitude de méthode prêtres à l'emploi et sont plus ou moins facile d'utilisation. Sur le site web, on se servira de l'API JavaScript de GoogleMaps.

Section 6 :
La conception du Site

Une des fonctionnalités du server et de donner une interface Homme-Machine permettant aux utilisateurs d'interagir avec les données collectés par le téléphone, c'est dans ce but que nous avons décidé de créer un site web. Ce dernier comportera plusieurs éléments (en plus des scripts de réceptions de données des téléphones).

Afin d'expliquer plus facilement et de montrer les différentes possibilités offertes par le site, voici un diagramme de cas d'utilisations :



2. Quelques fonctionnalités

On s'aperçoit donc, qu'il ya trois niveaux d'accès, à savoir un internaute lambda qui n'est pas inscrit sur le site, un internaute qui a son compte crée et actif, et les admin/webmaster qui gèreront le site.

L'utilisateur qui souhaite créer son compte sur le site web devra remplir un formulaire (voir en annexe), et soumettre sa demande. Pour que le compte soit actif, un admin devra accepter cette demande après avoir pris connaissances des informations. Lors de l'activation du compte, un mail sera envoyé à l'utilisateur afin de spécifier si son compte à été accepté ou non, et dans un cas positif, l'utilisateur recevra un numéro d'accès unique qui est généré sur le serveur (ce numéro permettra à l'utilisateur de se connecter au serveur depuis son mobile, à la manière d'un mot de passe).
Une fois l'utilisateur inscrit sur le site et connecté sur celui-ci, plusieurs options s'offre à lui :

- Il peur modifier ses informations
- Il peut voir son suivi sportif (annexe) qui est en fait un récapitulatif s des données sur lui-même. Dans un premier temps nous affichons juste les localisations des amis, mais cette page est là afin de nous permettre dans un second temps de montrer à l'utilisateur des graphiques de performances en fonction des données collectées sur lui.
- Il peut également gérer ses contacts. Un système d'amis à été mis en place afin d'avoir un aspect communautaire plus important. Ce système permet comme beaucoup d'autres systèmes d'amis, de rechercher, d'inviter, d'accepter une personne, mais aussi de la refuser ou de la bloquer.

En ce qui concerne le grade admin, nous avons une page récapitulatif qui permet à celui-ci d'avoir une vision globale des utilisateurs (voir annexe). A l'aide de ce tableau, l'admin aura une vision rapide des nouvelles demandes d'inscription et pourra d'un simple clic, soit accepter soit refuser. Le mail envoyé, ainsi que le message, étant automatique, l'admin n'aura rien à faire.

Il aura aussi la possibilité de supprimer un utilisateur (par exemple si la personne n'est plus active depuis plusieurs mois). L'admin pourra également voir la page de suivit de sportif de chaque utilisateur comme s'il était connecté à leur place. Cela permettra à l'admin de faire un suivit facilement et rapidement, sans avoir à demander à chaque utilisateur ce qu'ils ont fait dans la semaine.

2.1. Plus loin ...

Toujours dans un souci de rendre ce site web plus attractif pour les utilisateurs, nous avons commencé à travailler sur un système de messagerie interne. Une phase Beta est en marche et une série de test reste à faire, mais cette fonctionnalité permettrait d'envoyer un message à l'un de ses amis en allant sur son suivi sportif (pour l'inviter dans un évènement par exemple). Cette messagerie permettra également aux admins de laisser un message sur le suivi sportif des utilisateurs pour les encourager par exemple.

Nous pensions également faire un lien avec les téléphones et cette messagerie afin de donner la possibilité aux utilisateurs de lire et d'envoyer ces messages directement depuis leur téléphone.

2.2. Sécurité

Une chose essentielle pour ce genre de site web (surtout en php) est la sécurité des accès et des données. Un utilisateur lambda ne doit pas pouvoir trouver un moyen d'avoir les droits d'un admin sur le site auquel cas la pérennité du site est affecté.

Plusieurs moyens existent pour pirater un site, les plus connus étant :

- les injections SQL (perturbe directement les bases de données à l'aide de requête SQL dans des champs non protégés)
- les failles XSS (cross-site scripting, on cherche un endroit vulnérable dans le site comme un champ de formulaire ou alors directement une URL, afin de lancer un script qui peut par exemple voler des informations)

Afin d'éviter ses attaques, plusieurs mesures sont à prendre au niveau du code, mais PHP contient des fonctions permettant de nous aider également. Par exemple pour les injections SQL, le point sensible étant l'utilisation de quotes, des fonctions comme mysql_real_escape_string() ou encore prepare() permettent d'éviter un grand nombre d'injections. De même des fonctions comme htmlspecialchars() ou htmlentities() permet de limiter certaines failles XSS.

Comme nous travaillons également sur un server Apache, certaines fonctionnalités nous sont offertes pour améliorer notre protection, c'est notamment le cas avec les fichiers .htaccess qui sont des fichiers de configuration permettant, entre autre, de définir des règles d'accès pour un répertoire. Cela nous est utile pour protéger un répertoire par un mot de passe ou bien pour éviter une vue sous forme de liste quand on accède à un répertoire par l'url. Mais ces fichiers de configuration permettent également d'autres choses comme des redirections (utile pour les pages d'erreurs), la réécriture d'url, de filtrer les accès au server en fonction du type de machine utilisé pour s'y connecter
Voila une partie de fichier .htaccess :

```
#empêche le listage sur le site
Options -Indexes

ErrorDocument 401 http://www.isima.fr/~lacomme/tel/401.php
ErrorDocument 403 http://www.isima.fr/~lacomme/tel/403.php
ErrorDocument 404 http://www.isima.fr/~lacomme/tel/404.php
ErrorDocument 406 http://www.isima.fr/~lacomme/tel/406.php
ErrorDocument 500 http://www.isima.fr/~lacomme/tel/500.php
ErrorDocument 503 http://www.isima.fr/~lacomme/tel/503.php

#AuthName "Acces securise au site iSportive"
#AuthType Basic

#AuthUserFile C:\wamp\www\iSportive_10_03_2011\secure\.pwdiSportive
```

3. Liaisons avec le téléphone

Une autre fonctionnalité du server est de recevoir et stocker les informations envoyées par les différents téléphones (iphone, android et windows 7). Pour cela nous avons crée plusieurs script PHP permettant de recevoir les données et qui les stocks directement dans la base de données du server. Pour le moment nous avons 5 scripts différents que je vais expliquer :

- Un script permettant de s'identifier. Le téléphone nous envoi un identifiant, un mot de passe, et un numéro d'accès (celui qu'on est censé obtenir lors d'une inscription). Ce script va collecter ces trois informations et regarder dans la base de données si elles sont bien en accord entre elles. Le script renverra des messages d'erreurs ou de confirmation selon les données reçues.

- Un script de tracking. Ce script va recevoir comme tous les scripts un identifiant, un mot de passe et un numéro d'accès, afin de voir si la personne qui envoie les données est bien présente dans la base, mais ce script reçoit également un fichier de données (.txt) contenant les informations de tracking que le téléphone à faite ainsi que le nom de l'activité faite. Ce fichier sera stocké sur le server dans une hiérarchie précise (à savoir un dossier pour la personne, puis un dossier pour le type d'activité). C'est ce fichier qui pourra être ensuite traité par le server afin de travailler sur les données et faire des graphiques récapitulatif. De même que le script précédent, des messages d'erreurs ou de confirmation sont renvoyé au téléphone.

- Un script de mis a jour de localisation. Sur ce script on reçoit une latitude, une longitude, et un temps que l'on stockera dans la base de données (on garde 50 infos par utilisateurs, une fois ce quota atteint, la première information est remplacée par la nouvelle). C'est informations permettront d'afficher la position des amis sur les cartes (site web et/ou téléphone) afin de les retrouver s'ils ne sont pas loin par exemple (bien sur ces données ne sont pas envoyées sans l'accord de l'utilisateur, et il peut à tout moment désactiver l'option GPS).

- Un script qui permet au téléphone de recevoir la liste des amis de la personne faisant la demande. A la réception du login et après vérification de l'existence de la personne dans la base, le script recherche toute les autres personnes amies et cherche également leur dernière position GPS enregistré dans la base. Cette série d'informations sera ensuite transmise au téléphone qui pourra alors afficher les amis sur une carte.

- Et enfin un script permettant de recevoir toute les informations d'une personne connaissant son login, son mot de passe et son numéro d'accès (pour par exemple faire une interface sur téléphone plus conviviale avec le prénom de la personne).

Il est donc aisé de rajouter d'autres scripts afin de réaliser et d'implémenter d'autres fonctionnalités au projet comme la messagerie entre personnes. Il faut cependant penser à faire de bonnes spécifications afin que les trois plateformes de développement (iphone, android, Windows7) envoient les même les données au serveur.

Section 7 :
Le travail accompli, et le reste

Ce que nous avons achevé :
Au moment du rendu de notre projet, nous avons donc complètement terminé :

- Une application iPhone fonctionnelle avec les modes suivants implantés : Login, Tracking, Amis, Configuration et Cohésion (sans calcul de la Dépense Énergetique).
- Un site Web fonctionnel avec les possibilités suivantes : création de compte, modification des données utilisateur, suivi sportif, gestion des amis et des demandes.
- Egalement les transfert Web-iPhone : changement de position toutes les 10 minutes, récupération des fichiers de tracking, envoi/réception des amis, login, modification des informations utilisateurs.

Ce qui reste à faire :
L'essentiel de ce qui reste à faire réside dans les échanges de données pour la fonction Messages. Celle-ci est présente de façon succincte sur le site Web et implémentée graphiquement sur l'iPhone mais aucun échange entre ces deux parties n'est encore possible. Avec un peu plus de temps, cette fonction serait arrivée à son terme.

Nous avons également implanté un embryon de système intelligent pour le téléphone qui se débrouille pour définir le repos/marche/course mais qui est encore en phase de test donc non vraiment concluant. Avec un peu plus de temps et de travail avec les mathématiciens cette partie pourrait réellement arriver à son terme et être concluante.

Section 8 :
En conclusion

Ce projet a été très bénéfique pour nous, tant sur le plan humain que professionnel. Le travail à plusieurs groupes sur un même projet montre l'importance de la communication entre ces groupes et aussi l'importance de la communication au sein du même groupe. Nous avons pu développé nos connaissances sur les deux langages importants à savoir : le PhP et l'Objective-C. Ceci nous a permis d'appréhender la difficulté de monter de toute pièce un site web et une application communiquant entre eux. De plus, un projet qui permet d'appliquer ce que l'on a appris en cours nous permet de vraiment nous mettre en situation quasi professionnelle et donc de préciser au mieux ce que nous voulons faire plus tard.

De plus, ce projet nous a été extrêmement bénéfique sur le plan de l'expression : les deux présentations que nous avons dû faire à l'INRA nous ont permis à la fois de prendre de l'assurance quand à l'expression orale, et nous ont montré l'importance d'explications claires et simples pour les néophytes en informatique afin de se faire comprendre au mieux.

Glossaire :
Définitions & Explications

[1] Délégué :
Le terme délégué (Delegate en anglais) permet de définir une classe qui sera la réceptrice des informations quand l'objet enverra des information.
Par exemple, dans la Section 1, lorsque l'on défini la classe en tant que délégué de l'accéléromètre, ce sera la classe qui recevra les messages de l'accéléromètre. A nous de définir les méthodes qui sont appelés lors de l'envoi de message. Par exemple pour l'accéléromètre, quand un message est transmis, il appelle automatiquement la fonction *accelerometer:didAccelerate,* que l'on pourrais traduire simplement par « l'accéléromètre a bougé ».

[2] framework :
un framework est une boite à outils qui contient toutes les classes pour la bonne utilisation d'un système, par exemple le GPS (CoreLocation.framework) ou bien les graphismes (CoreGraphics.framework) et bien d'autres. Il s'agit ni plus ni moins que d'outils développés par Apple permettant d'importer des fonction sans avoir a inclure les .m et .h dans le code.

[3] tracking (de l'anglais track : suivre):
le mode tracking désigne en fait une partie où l'utilisateur est « suivi » par le téléphone dans ces mouvement grâce au GPS.

Annexe :
Bibliographie

Partie iPhone :

Centre Developper d'Apple :
http://developer.apple.com

Livre : « Développer pour l'iPhone et l'iPad »
De : Etienne Vauterin
Edition : DUNOD

Partie Web :

Manuel de réference PhP :
http://php.net

Annexe :
Les affichages Serveur et iPhone

Vues de l'iPhone :

Figure 1 : vue Cohésion

Figure 2 : vue Tracking

Figure 3 : vue Amis (sans amis)

Figure 4 : vue Messages

Figure 5 : vue Configuration (Menu)

Vues du Serveur :

Figure 6 : Création de compte

Figure 7 : Activation du compte par un Administrateur

Figure 8 : Modification des informations personelles

Figure 9 : Suivi Sportif du site
Note : Aperçu du système de méessage au bas du screen.

Figure 10 : interface de gestion des amis