Remerciements

Nous remercions tout d'abord notre tuteur de projet **M. Yann Mathet**, enseignant-chercheur à l'université de Caen et responsable des projets de deuxième année du Master, pour ses conseils et sa disponibilité.

Ensuite nous tenons à remercier tous particulièrement **Jean Boudet**, notre collègue de promotion pour sa participation lors de nos tests de réception des signaux GPS.

Merci aussi aux **administrateurs système** de l'université d'avoir été présents lorsque nous avions besoin d'eux.

Enfin, nous tenons à remercier tous **nos collègues de promotion** dans laquelle règne une atmosphère agréable de solidarité et d'entraide.

Table des matières

Introduction

Ce projet prend place dans le cadre de notre Master Document Numérique en Réseau Ingénierie de l'Internet (DNR2I). Il sera fait au cours de ce rapport l'avancement de notre projet concernant le développement d'une application servant à la gestion de trajets pour les sportifs amateurs de course à pieds. Nous verrons au cours de ce rapport les enjeux du projet, nous analyserons ensuite les technologies employées ainsi que nos choix d'implémentations. Puis nous verrons en détail la réalisation du projet. Et enfin, nous terminerons par un bilan généralisant notre travail sur ce projet.

Première partie

Présentation et objectifs

Chapitre 1

Présentation du projet

1.1 Contexte

De nos jours, les smartphones disposent tous d'une puce GPS permettant leur géolocalisation à chaque instant avec une précision de plus en plus grande. C'est donc l'outil idéal pour les joggers et cyclistes qui souhaitent enregistrer et analyser leurs performances. De nombreuses applications mobiles ont ainsi vues le jour ces dernières années pour répondre à ces besoins. On peut notamment évoquer le leader sur le marché des applications mobiles dans la catégorie santé et remise en forme qu'est Runkeeper ou encore un de ses concurrents Runtastic.

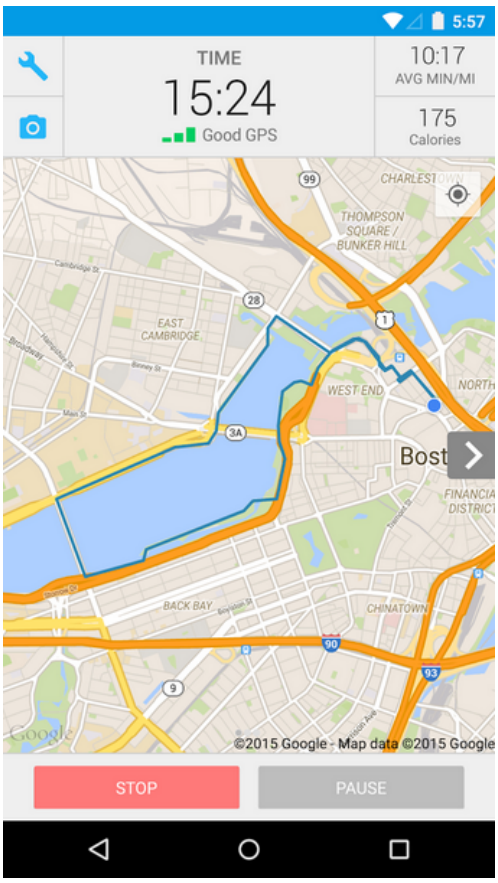

FIGURE 1.1 – Application RunKeeper

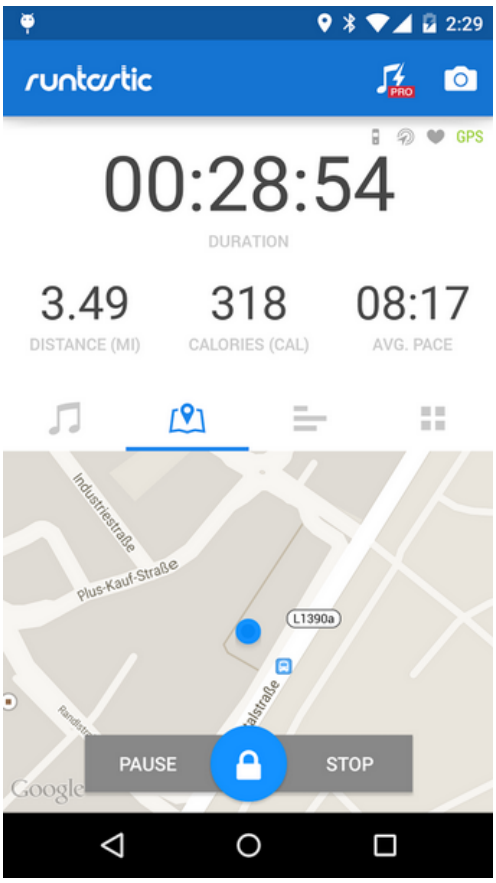

FIGURE 1.2 – Application Runtastic

Classiquement, ces deux applications propose d'enregistrer les trajets de l'utilisateur, d'afficher son parcours sur une carte et de calculer en temps réel les différentes données de la course. Mais certaines fonctionnalités qui pourrait se montrer très intéressantes pour l'utilisateur ne sont pas proposées.

C'est dans ce contexte que notre tuteur, M. Mathet nous a proposé de concevoir et de développer une application mobile permettant d'enregistrer les tracés GPS des parcours de l'utilisateur et d'analyser ses performances. Mais aussi, et c'est ici que tous l'enjeu du projet et la nouveauté se situe, de comparer en temps réel les performances de l'utilisateur par rapport aux précédentes effectuées sur le même parcours. Concrètement, il s'agit d'afficher l'avance ou le retard en secondes par rapport à une performance antérieur définie (la meilleure ou la moyenne des dernières).

Chapitre 2

Objectifs de l'application

2.1 Les trajets

Les objectifs de ce projet sont multiples. D'abord il s'agit de réaliser une application mobile pour smartphone Android qui dispose des fonctionnalités suivantes :

- Enregistrement des coordonnées GPS du trajet
- Visualisation du trajet sur une carte
- Consultation de l'historique des trajets
- Suppression d'un trajet
- Analyse des données pour obtenir :
  - le temps du trajet
  - la distance parcourue
  - la vitesse moyenne en km/h
  - l'allure en min/km (temps moyen pour parcourir un kilomètre)

2.2 Les parcours

Le second objectif est certainement le cœur du projet puisque c'est dans celui-ci que réside la nouveauté. Il s'agit de regrouper les trajets identiques pour pouvoir les comparer. Un parcours représente alors un ensemble de trajets qui suivent à peu de chose près la même trajectoire.

Avec la mise en place de cette nouvelle structure de données nous pourrons extraire des statistiques intéressantes pour l'utilisateur à savoir :

- La meilleure performance sur le parcours
- Les différentes moyennes du parcours :
  - le temps moyen
  - la distance
  - la vitesse moyenne
  - l'allure moyenne

Mais nous pourrons aussi comparer plusieurs trajectoires entre elles pour permettre à l'utilisateur de connaître en temps réel le retard ou l'avance qu'il possède par rapport à un autre trajet. L'utilisateur devra donc pouvoir fixer le trajet à prendre en référence pour chaque parcours. Ce trajet de référence

2.3 Le diagramme de cas d'utilisation

A partir de ces besoins, nous avons élaboré un diagramme de cas d'utilisation qui permet de visualiser globalement le comportement de l'application. Il décrit les enchainements d'actions que les différents acteurs peuvent effectuer. Dans notre application il n'existe qu'un seul acteur : l'utilisateur.

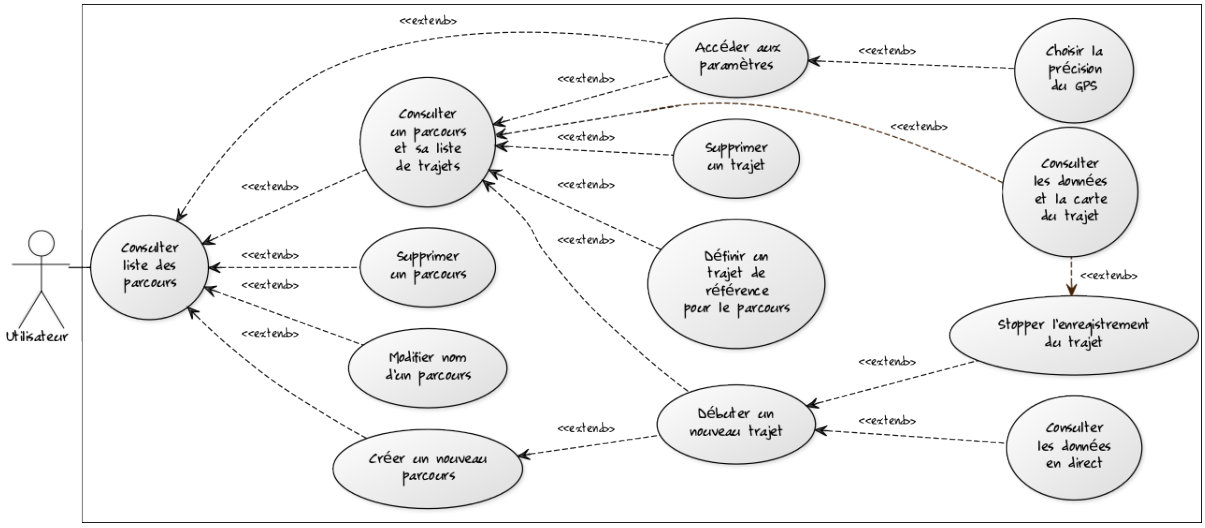

FIGURE 2.1 – Diagramme de cas d'utilisation de l'application

2.4 Les différents écrans

Avec ce diagramme on peut facilement imaginer la structure de l'application à réaliser et le contenu des différents écrans. Quatres écrans principaux en découle :

— La liste des parcours
— Le détail d'un parcours contenant sa liste de trajet
— Le détail d'un trajet avec l'affichage du tracé sur une carte
— L'écran d'enregistrement du trajet avec les données en temps réel

2.5 Les paramètres

On remarque un dernier cas d'utilisation nécessitant la création d'un nouvel écran, la gestion des paramètres, ou l'utilisateur pourra régler la précision du GPS. Il s'agit la de définir l'intervalle d'actualisation des coordonnées GPS pour optimiser la consommation de la batterie ou la précision de son trajet. Plus la précision seras élevée, plus la consommation de batterie de son appareil sera grande et inversement.

Deuxième partie

Analyse et implémentations

Chapitre 3

Détails sur l'implémentation

3.1 Représentation du modèle

3.1.1 Les classes

Parcours

Comme expliqué au chapitre précédent un parcours est constitué de plusieurs trajets. Donc la classe `Parcours` possédera une liste de trajets. Il regroupe également des informations qui lui sont propres tel qu'un nom et un identifiant. On remarquera également qu'un parcours possède un trajet de référence. Ce trajet est cruciale car il permettra la comparaison avec le trajet qui est effectué. Comme celui-ci pourra être modifié, on le représentera dans la classe `Parcours` par son identifiant. On ajoutera à cette classe tout un tas d'attributs servant à déterminer le meilleur temps, la vitesse moyenne, la vitesse maximale sur l'ensemble des trajets effectués.

Trajet

La classe `Trajet` représente le chemin qu'un utilisateur emprunte lorsqu'il effectue une course. On y retrouve une liste de points (`Location`) nécessaire à la visualisation du trajet ainsi qu'à la comparaison avec le trajet de référence. On gardera également des informations concernant la date à laquelle le trajet a été réalisé ainsi qu'un identifiant pour faciliter le stockage en base. Un trajet possède également une distance ainsi qu'un temps total. Ces deux attributs sont modifiés lors de l'ajout d'un point via la méthode `addLocation` (détaillée dans l'Annexe A) et permettent le calcul de la vitesse moyenne.

Location

Lors de implémentation du modèle, nous avions créé une classe Point permettant de stocker la latitude, la longitude et la date à laquelle le point est captée. Nous avions également établie la formule permettant de calculer la distance entre deux points (voir l'Annexe B). Mais après avoir fait des recherches sur le fonctionnement du GPS sous Android, nous nous sommes aperçu que le `LocationManager` retournait des objets `Location` qui contiennent des attributs pour la latitude, la longitude et la date. De plus, une méthode `distanceTo(Location l)` permet de calculer la distance entre deux points géolocalisés.

3.1.2 Le diagramme de classe

Afin de résumer les parties précédentes, nous avons élaboré un diagramme de classes (Figure 3.1)

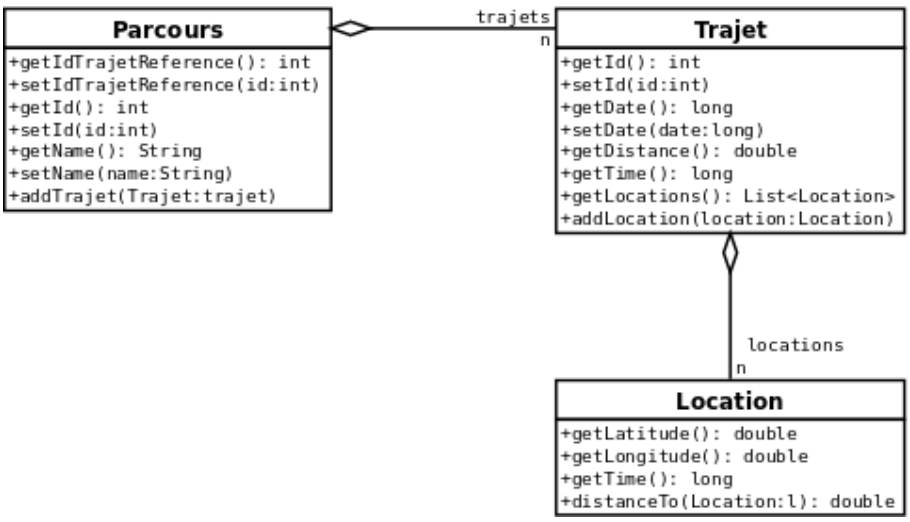
Parcours
+getIdTrajetReference(): int
+setIdTrajetReference(id:int)
+getId(): int
+setId(id:int)
+getName(): String
+setName(name:String)
+addTrajet(Trajet:trajet)
trajets
n
Trajet
+getId(): int
+setId(id:int)
+getDate(): long
+setDate(date:long)
+getDistance(): double
+getTime(): long
+getLocations(): List<Location>
+addLocation(location:Location)
locations
n
Location
+getLatitude(): double
+getLongitude(): double
+getTime(): long
+distanceTo(Location:l): double

FIGURE 3.1 –

Chapitre 4

Les technologies utilisées

4.1 Le choix du langage

4.1.1 Javascript

Nous avons réalisé une étude sur les différentes technologies que nous pouvions employées pour réaliser ce projet. En effet, avec les technologies Javascript émergentes, l'utilisation de Cordova et Phonegap permettait de réaliser une application compatible avec l'ensemble des smartphones présent sur le marché actuel. Malheureusement, ces technologies sont plutôt orientées sur des technologies web. Le développement du code avec ces technologies nécessite une compilation afin d'être lisible par le smartphone sur lequel il est déployé.

4.1.2 Java

Le développement sous Android se fait dans un langage très populaire dans les systèmes embarqués, le Java. De plus, la compilation du langage est optimisée en fonction de l'appareil sur lequel il est déployé. De ce fait, l'exécution d'une application est plus performante si elle est conçue dans le langage natif au smartphone(Java pour Android, Objectif-C pour iOS). Notre application utilisant le GPS du smartphone, nous avons opté pour un développement en Java afin d'optimiser les connexions avec le satellite servant à géolocaliser le téléphone. Android étant mis à disposition par Google, il existe une documentation extrêmement fournie par l'API Google [1]

4.2 Le stockage de données

Pour un bon fonctionnement, l'application doit stockées des informations concernant les parcours et les trajets.

4.2.1 SQLite

Sous Android (et autres smartphones), le stockage des données se fait via une base de données en SQLite. SQLite est une bibliothèque proposant un moteur de base de données relationnelles utilisant le langage SQL. Elle est plus légère que ses homologues MySQL et PostgreSQL car elle n'intègre pas le schéma habituel Client-Serveur. En effet, l'ensemble des données est stocké dans un fichier indépendant d'Android et directement intégré à l'application.

1. Google API

4.2.2 Le format GPX

Le format GPX est un format XML conçu spécialement pour représenter un ensemble de points GPS ayant une latitude et une longitude à un instant t donné. L'utilisation de fichier GPX est très intéressant dans notre projet dans la mesure où le GPS d'Android nous fournit des points devant d'être stockés mais sans encombrer la base. En effet, SQLite n'est pas très adapté au stockage de grosses données (big data). De plus, cette séparation permet d'extraire facilement les données liées à la représentation du trajet effectué.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<gpx xmlns="http://www.topografix.com/GPX/1/1" creator="MapSource 6.9.2"
   version="1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.topografix.com/GPX/1/1 http://www.
   topografix.com/GPX/1/1/gpx.xsd">
  <metadata>
    <time>2011-04-13T15:58:51Z</time>
    <bounds maxlat="49.746009" maxlon="-1.372054" minlat="49.644963" minlon=
    "-1.925924"/>
  </metadata>
  <trk>
    <trkseg>
      <trkpt lat="49.645132" lon="-1.620045">
        <ele>-46.486572</ele>
        <time>2011-04-06T08:47:37Z</time>
      </trkpt>
     ...
      <trkpt lat="49.645703" lon="-1.619700">
        <ele>-3.227295</ele>
        <time>2011-04-06T08:49:37Z</time>
    </trkseg>
  </trk>
</gpx>
```

Code 4.1 – Exemple de fichier GPX

4.2.3 L'analyseur syntaxique

L'utilisation de fichier GPX implique forcement un système de lecture et d'écriture de fichier XML (parseur). En Java, il existe deux grandes familles de parseurs XML, les parseurs utilisant SAX et les parseurs utilisant le DOM.

Les parseurs DOM

Les parseurs DOM analysent la structure même d'un document.Ils stockent en mémoire l'ensemble des balises XML (le DOM) afin de pouvoir retrouver n'importe quel élément du document. l'inconvénient vient du fait qu'un document ne peut pas être plus gros que la mémoire vive.

Les parseurs SAX

Les parseurs SAX sont événementiels. En effet, ils analysent le document et déclenchent un événement lorsqu'une balise est construite ou détruite. Cette façon de lire un document est nettement moins coûteuse que la précédente dans la mesure où rien n'est stockée par le parseur. La lecture du document se fait au fur et à mesure. Cette méthode est beaucoup plus adaptée pour notre projet dans la mesure où les fichiers GPX analysés sont volumineux et les appareils n'ont pas beaucoup de mémoire.

Troisième partie

Détail sur la réalisation

Chapitre 5

La gestion des points

5.1 La capture de points

5.1.1 Les *providers*

5.1.2 Le GPS

5.1.3 Le LocationManager

5.2 Interpolation d'un point

5.2.1 Calcul des coordonnées de H

5.2.2 Interpolation temporelle

5.3 Détermination d'un segment

5.3.1 Les problèmes rencontrés

5.3.2 L'algorithme

Chapitre 6

Développement Android

6.1 Android en bref

Java, activity, fragment, services, permissions

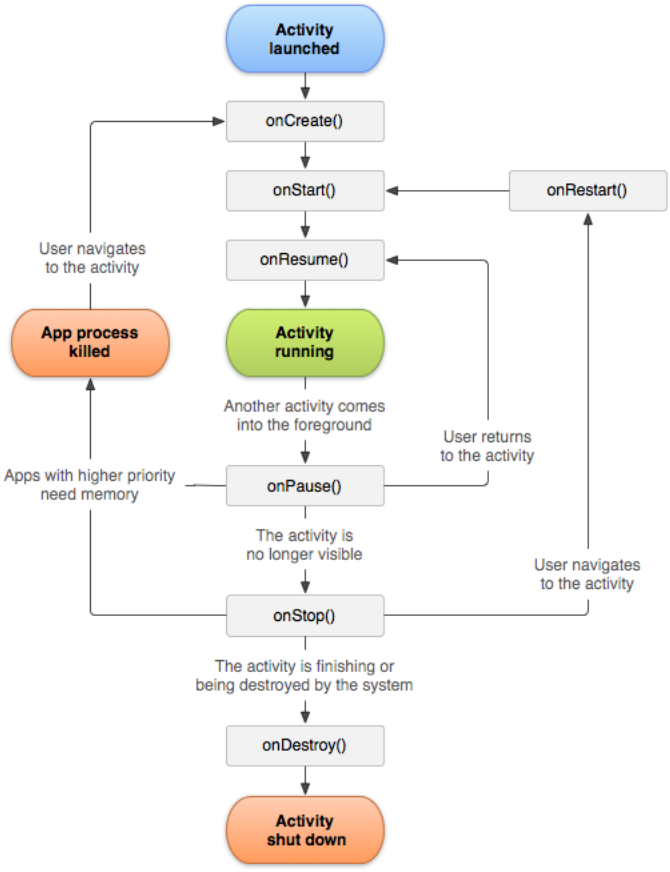

FIGURE 6.1 – Cycle de vie d'une Activity

6.2 Structure du projet

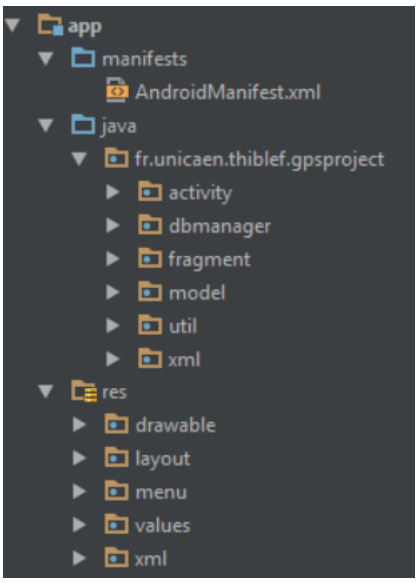

FIGURE 6.2 – Structure de l'application

6.3 Fonctionnement des layouts

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/textView"
        android:layout_weight="0.5"
        android:text="@string/temps" />

    <fragment
        android:id="@+id/map_container"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="0.5"
        class="com.google.android.gms.maps.SupportMapFragment"
        android:name="com.google.android.gms.maps.SupportMapFragment" />
</LinearLayout>
```

Code 6.1 – Exemple d'un layout

6.4

Quatrième partie

Bilan du projet

Glossaire

Base de données : c'est un ensemble de données stockées qui peuvent être accédées et modifiées via un langage de manipulation de données.

Bibliothèque / Librairie : désigne un groupe de fonctionnalités dont les caractéristiques sont éditées, et donc à la disposition de différentes applications.

JavaScript : c'est un langage de programmation orienté objet, principalement utilisé dans les pages web.

Modélisation : Action de décrire un système réel de façon formelle, de façon à pouvoir le manipuler par ordinateur.

SAX (Simple API for XML) : API basée sur un mode événementiel permettant de réagir à des événements (comme le début d'un élément, la fin d'un élément) et de renvoyer le résultat à l'application.

Sources

Appendices

Annexe A

L'ajout de points

```
public void addLocation(Location l) {
  //Si il n'y a pas de points d'enregistres
  if (locations.size() == 0) {
    distance = 0;
    locations.add(l);
    temps = 0;
  } else {

  //On recupere la derniere position
  Location last = getLastPosition();

  //On augmente la distance total du trajet
  distance += last.distanceTo(l);
  locations.add(l);

  //On incremente egalement le temps
  if (locations.size() == 1) {
    temps = 0;
  }
  temps += ((l.getTime() / 1000) - (last.getTime() / 1000));
  }
}
```

Code A.1 – L'ajout de point à un trajet

Annexe B

Calcul de la distance entre deux points