

Android Cycling Trainer



Lechaire Thomas – FIN2
ETML - Lausanne
102 heures
M. Patrick Chenaux

Table des matières

1	Spécifications.....	3
1.1	Spécifications de départ et documentation associée.....	3
1.1.1	Objectifs et portée du projet	3
1.1.2	Fonctionnalités requises (du point de vue de l'utilisateur)	3
2	Planification.....	4
2.1	Planning(s) initial	4
3	Analyse.....	6
3.1	Faisabilité	6
3.1.1	Les compétences à acquérir :.....	6
3.1.2	Matériel/Logiciels	6
3.1.3	Recherches d'informations/documentation	6
3.1.4	Gestion du temps.....	6
3.1.5	Choix des logiciels.....	7
3.2	Document d'analyse et conception	9
3.3	Conception des tests	18
3.4	Planning détaillé.....	18
4	Réalisation	18
4.1	Dossier de réalisation.....	18
4.2	Modifications	31
5	Tests.....	31
5.1	Dossier des tests	31
6	Conclusion	31
6.1	Bilan des fonctionnalités demandées.....	31
6.2	Bilan de la planification.....	32
6.3	Bilan personnel	32
7	Divers	32
7.1	Journal de travail de chaque participant	32
7.2	Bibliographie.....	32
7.3	Webographie.....	32
8	Annexes.....	32

1 Spécifications

1.1 Spécifications de départ et documentation associée

1.1.1 Objectifs et portée du projet

- Analyser attentivement le cahier des charges.
- Proposer une interface graphique simple et intuitive.
- Réaliser l'application sous Android.
- Tester l'application.

1.1.2 Fonctionnalités requises (du point de vue de l'utilisateur)

- L'utilisateur doit pouvoir ...:
- Créer de façon simple une séance d'entraînement par exemple :
- Entraînement route « force » : 6 sprints 8''/52'' + 6 sprints 12''/48'' + 5 x 1' + 3 x 3' + 1 x 5' + 3 x 3' + 6 sprints 10''/50''.
- Entraînement VTT « côtes » : 2 tours de 3 côtes de 5' une doublée (7 côtes) : 4^e et 5^e en 15/15 et 6^e et 7^e en 20/20.
- Entraînement route « rythme » : 30' échauffement avec 10 sprints de 6 à 8'', 10' 20/20 côte 160-165 bpm, 15' vitesse 130 bpm, 3' 30/30 côte à bloc 180 bpm, 15' vitesse 130bpm, 20' contre-la-montre 165 bpm souplesse, 15' vitesse 130 bpm, 20' contre-la-montre 165 bpm braquet + gros, 20' retour au calme.
- Afficher un résumé de la séance d'entraînement, cela permet d'avoir un aperçu global de l'entraînement.
- Démarrer la séquence d'entraînement, chaque partie de la séquence doit être affichée de façon claire, au moyen de couleurs et contrôles graphiques adéquats, barres de défilement, boutons, clignotements etc.
- Mettre en pause la séquence d'entraînement et la redémarrer au moment souhaité.
- Sauvegarder une séance d'entraînement créée.
- Charger une séance d'entraînement précédemment sauvegardée.
- Insérer sa fréquence cardiaque de repos et maximale, elle pourra être utilisée pour travailler ou afficher une certaine zone cardiaque.
- Calculer un indice de récupération sur le vélo pour une date précise :
- Fcmax – FC 1'30'' après effort, cette valeur augmente lorsque la forme physique augmente, afficher l'historique des valeurs enregistrées.

2 Planification

2.1 Planning(s) initial

- Dates de réalisation : du lundi 11.05.2015 au lundi 08.06.2015
- Horaire de travail :
 - Lundi** 08h00-12h15 13h10-15h40 Pentecôte le 25 mai
 - Mardi** 08h00-11h25 13h10-16h35
 - Mercredi** - 13h10-16h35
 - Jeudi** 08h00-11h25 12h20-15h40 Ascension le 14 mai
 - Vendredi** 08h00-11h25 12h20-15h40 Pont de l'Asc. le 15 mai

- Nombres d'heures : 102

Tâches - objectifs	Nb 1/4 heure	Semaine 1
Absence - Imprévu	72	
Planification	0	
Analyse	24	
Documentation	0	
Maquette de réalisation	21	
Programmation	0	
Tests	41	
Debugging	0	
Rédaction Rapport et Journal de travail	116	
Zone Tampon	16	
Total planifié	456	
Total réalisé	0	

2.1 Semaine 1

Tâches - objectifs	Semaine 2
Absence - Imprévu	
Planification	
Analyse	
Documentation	
Maquette de réalisation	
Programmation	
Tests	
Debugging	
Rédaction Rapport et Journal de travail	
Zone Tampon	
Total planifié	
Total réalisé	

2.2 Semaine 2

Tâches - objectifs	Semaine 3
Absence - Imprévus	<div><div></div></div>
Planification	
Analyse	
Documentation	
Maquette de réalisation	
Programmation	<div><div></div></div>
Tests	<div><div></div></div>
Debugging	
Rédaction Rapport et Journal de travail	<div><div></div></div>
Zone Tampon	<div><div></div></div>
Total planifié	
Total réalisé	

2.3 Semaine 3

Tâches - objectifs	Semaine 4
Absence - Imprévus	
Planification	
Analyse	
Documentation	
Maquette de réalisation	
Programmation	<div><div></div></div>
Tests	<div><div></div></div>
Debugging	<div><div></div></div>
Rédaction Rapport et Journal de travail	<div><div></div></div>
Zone Tampon	<div><div></div></div>
Total planifié	
Total réalisé	

2.4 Semaine 4

Tâches - objectifs	
Absence - Imprévus	
Planification	
Analyse	
Documentation	
Maquette de réalisation	
Programmation	
Tests	
Debugging	
Rédaction Rapport et Journal de travail	<div><div></div></div>
Zone Tampon	<div><div></div></div>
Total planifié	
Total réalisé	

2.5 Semaine 5

Le lien vers le fichier de planification initiale est disponible en annexe

3 Analyse

3.1 Faisabilité

Dans le cadre de ce projet, une analyse est nécessaire afin de pouvoir anticiper les problèmes qui pourraient survenir durant la réalisation ou la programmation. Il faut vérifier certains points comme :

- les compétences à acquérir ou approfondir
- le matériel nécessaire ou à exploiter
- les recherches d'informations particulières
- la gestion du temps de travail
- sélection des logiciels
- les difficultés potentielles et les solutions envisagées

3.1.1 Les compétences à acquérir :

- Compétences en java et POO
- Compétences en programmation Android
- Bases en XML
- Compétences en Sql ou Base de données
- Connaissances de l'environnement Android Studio

3.1.2 Matériel/Logiciels

- Un Ordinateur PC
- Microsoft Office
- Android Studio
- Virtual Box
- Gimp ou InkScape pour le traitement d'image
- Navigation Internet

3.1.3 Recherches d'informations/documentation

- Recherches sur les bases de données Android
- Documentation sur l'environnement Android Studio
- Recherches sur les solutions pour les Tests ou Debugging
- Recherches sur les interfaces Android
- Documentation de Google pour le développement Android

3.1.4 Gestion du temps

La gestion du temps se fait grâce à la planification Initiale ainsi qu'au journal de bord, qui permettent de suivre un fil rouge. La gestion du temps peut facilement devenir problématique. Il est donc important de respecter au maximum les délais fixé dans la planification (initiale et détaillée) afin de mener le projet à son terme.

3.1.5 Choix des logiciels

Concernant le logiciel de développement. Deux grands logiciels peuvent être envisagés. Eclipse projet d'Eclipse Foundation qui possède un plugin (ADT) qui s'intègre très facilement à Eclipse et qui permet de développer sous Android, ainsi que Android Studio qui est développé par Google et qui se base sur IntelliJ IDEA (autre logiciel de développement).

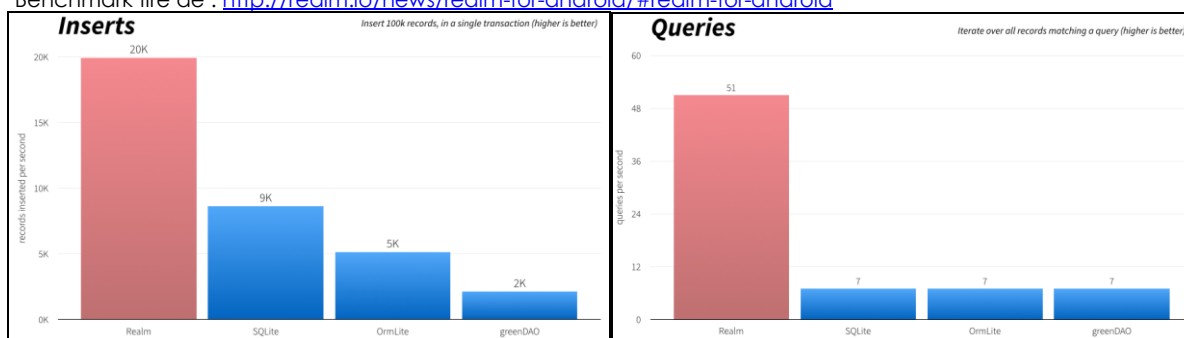
Note: If you have been using Eclipse with ADT, be aware that [Android Studio](#) is now the official IDE for Android, so you should migrate to Android Studio to receive all the latest IDE updates. For help moving projects, see [Migrating to Android Studio](#).

3.1 Note de Google concernant l'arrêt du développement d'éclipse pour Android

ADT n'étant plus mise à jour et Google recommandant son produit pour obtenir les dernières mises à jour, Android Studio a donc été choisi pour développer le projet.

Pour la partie base de données, SQL Lite est la solution de base implémentée pour Android. Cependant les recherches ont mises en avant un grand nombre de développeurs ayant des problèmes et principalement des problèmes liés à la vitesse d'exécution des requêtes ainsi qu'à la difficulté d'implémentation des bases de données avec SQL Lite. Comme solutions, beaucoup de réponses parlaient et mettaient en avant Realm qui une librairie permettant la création de base de données sur mobile. Pourquoi choisir Realm plutôt que SQL Lite ou un ORM (cf. [Lexique](#)) comme greenDAO ? Premièrement, car Realm est facile à implémenter, il utilise son propre mécanisme de persistance (cf. [Lexique](#)) et est capable de sauvegarde n'importe quel objet implémenté dans le code. Deuxièmement, car Realm est multiplateformes. Un fichier *.realm contenant une base de données peut très facilement être importée dans un autre projet que ce soit un projet Android ou IOS. Et enfin, car il est, d'après le benchmark trouvé, plus rapide qu'un ORM ou qu'une base de données SQL Lite. Realm est capable d'insérer 2000 enregistrements à la seconde contre 900 pour SQL Lite et 500 pour OrmLite et exécuter 51 requêtes secondes contre 7 pour les autres plateformes même s'il faut bien sûr prendre du recul sur ces résultats, car ils ont été effectués par les créateurs de Realm et non par un tiers.

Benchmark tiré de : <http://realm.io/news/realm-for-android/#realm-for-android>



3.2 Requêtes avec Realm (51 par sec)

3.3 Insertions avec Realm (2k par sec)

Pour la compilation du projet, un Samsung S3 version 4.3 a été utilisé ainsi que GenyMotion pour émuler des autres terminaux. L'émulateur Android permet l'installation de nouveau terminal, mais la mise en place est longue et complexe pour un résultat et une rapidité très moyenne. GenyMotion est une solution rapide et facile à mettre en place puisqu'un plugin est directement téléchargeable depuis Android Studio. Il s'intègre dans la barre des tâches et propose une liste de plus de plus de 80 devices dans 7 versions Android différentes. Pour les installer rien de plus simple car se sont de simples Machines Virtuelles gérées grâce à VirtualBox. Le téléchargement de la machine virtuelle suffit à avoir un émulateur prêt à être lancé. Un gain de temps énorme pour développer et tester son application sur un grand nombre de devices et ainsi augmenter la compatibilité, sans mettre en avant la possibilité de prendre des screenshots, de simuler des positions GPS, d'installer des applications par Drag and Drop ou encore de simuler les capteurs comme l'accéléromètre.

	Genymotion	Physical device
Deploy a 30 MB App	7s	21s
Start a device	15s	40s

3.4 Comparaison GenyMotion

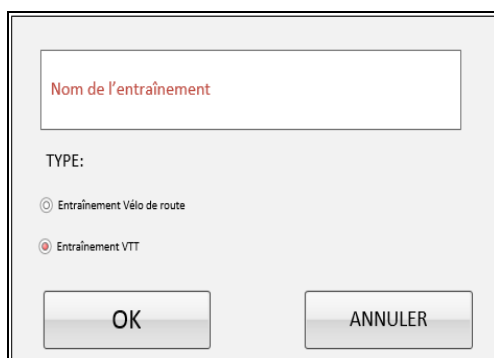
Concernant le reste des logiciels utilisés, ils sont les suivants : Microsoft Office 2016 en preview pour la gestion de la partie administrative et rapport du projet, Gimp et Inkscape pour tout ce qui est la partie design et retouche (principalement pour les interfaces), Chrome pour la navigation et educanet, qui est la plateforme de l'ETML pour la correspondance. Tous ces outils sont des outils gratuits et ils ont été choisis pour cela.

3.2 Document d'analyse et conception

3.2.1 Maquette graphique



3.5 Activité entraînements



3.6 Boîte de dialogue

Dans cette activité (cf. [Lexique](#)), chaque entraînement affichera le nom choisi pour l'entraînement, l'icône du vélo selon le type d'entraînement et la date du jour de création de l'entraînement.

Deux Type d'entraînements :

- VTT (en Bleu) (1)
- Vélo de route (en rouge) (2)

(6) Le menu paramètre permet d'ajouter les fréquences cardiaques de repos et maximum.

Voir image 3.6

(3) Un geste (Swipe cf. [Lexique](#)) vers la gauche laissera apparaître un bouton permettant d'effacer l'entraînement.

(4) Chaque entraînement (ligne) est cliquable. Le click ouvre l'entraînement et laisse apparaître les séquences contenu dans chaque entraînement.

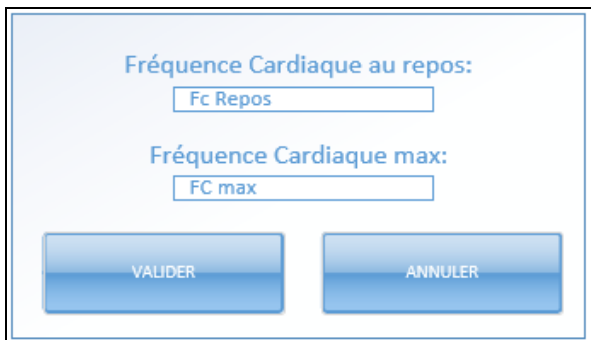
Voir Image 3.7 Eléments entraînement

(5) Le Bouton « Ajouter Entraînement » ouvre une boîte de dialogue permettant l'ajout d'un entraînement.

Voir : Image 3.5 Boîte de dialogue.

La boîte de dialogue apparaît lorsqu'on ajoute un entraînement.

- Une Zone pour ajouter le nom de l'entraînement
- 2 boutons pour définir le type d'entraînement VTT ou Route.
- 2 boutons pour valider ou annuler.



3.7 Insertion des fréquences cardiaques

Au clic sur le bouton paramètres de l'activité entraînement (Image 3.4) une boîte de dialogue s'ouvre et permet à l'utilisateur d'introduire dans la base de données sa fréquence cardiaque de repos et sa fréquence cardiaque maximum (qui sera utilisé pour calculer un indice de récupération)



3.8 Éléments entraînements

(1) Un bouton pour ajouter un élément à l'entraînement.

L'action sur ce bouton ouvre l'activité détails.

Voir : Image 3.8 Détails activité

Chaque ligne représente un élément de l'entraînement, comme des sprints, de la récupération, un contre-la-montre. L'élément indique à l'utilisateur le temps de la séquence, les [Bpm](#) et les [Rpm](#).

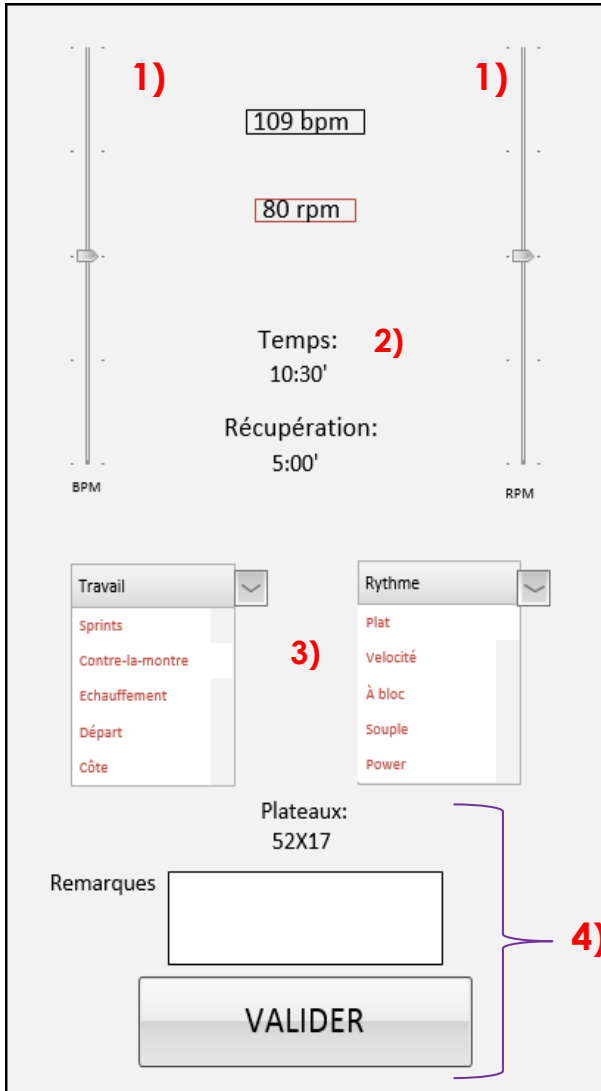
Pour les termes techniques voir Annexes : [Lexique](#)

(2) Comme pour l'entraînement. Un geste (Swipe) vers la gauche laissera apparaître un bouton permettant d'effacer un élément.

Chaque élément (ligne) est aussi cliquable permettant de modifier les valeurs de l'élément choisi.

(3) Le bouton démarrer ici, lance l'activité Timer qui démarre l'entraînement de vélo.

Voir : image 3.9 Timer.



3.9 Détails activité

Les détails de l'activité sont accessibles soit en ajoutant un élément soit en cliquant sur un élément de l'activité précédent (3.7) permettant ainsi la modification.

Dans cette activité, il est possible de choisir plusieurs informations.

Les deux barres verticales permettent de choisir les Bpm et les Rpm (1) et mettent à jour automatiquement les valeurs dans les deux TextView (cf. [Lexique](#)) prévues à cet effet.

(2) Le temps et la récupération sont validés par des TimePicker (cf. [Lexique](#)), présent dans Android. Le choix d'afficher le TimePicker directement ou dans une boîte de dialogue (cf. [Lexique](#)), est encore à faire.

(3) Deux listes déroulantes ou Spinners seront implémentés. Le premier pour choisir le type de travail à effectuer, comme sprint, contre-la-montre ou côtes et le second pour choisir le rythme de la séquence (à bloc, Vitesse etc...)

(4) Un élément pour choisir les plateaux avant et arrière, une boîte de texte pour ajouter des remarques personnelles et un bouton « valider » pour enregistrer la séquence et l'ajouter à l'entraînement.

3.10 Timer

3.11 Récupération

Tous les éléments présents ici (1) proviennent de la séquence en cours et ont été entrés par l'utilisateur. L'affichage permet à l'utilisateur de savoir ce qu'il doit faire durant la séquence. Le type de travail, le rythme, les Bpm et Rpm, les plateaux les remarques et même la séquence suivant seront affichés à l'écran.

Cette activité permet de lancer les chronomètres et le temps des séquences (2) et de l'entraînement(3).

Le point 2 représente donc le temps de la séquence. Il a été défini par l'utilisateur dans l'activité 3.8 (Détails activité) et le point 3 représente lui l'entraînement au complet.

(4) Une série de contrôles permettant d'arrêter, de mettre sur pause ou de stopper l'entraînement.

En fin d'entraînement lorsque ce dernier est terminé, une boîte de dialogue permettra à l'utilisateur de rentrer sa fréquence cardiaque après 1m30 de récupération. Grâce à cette fréquence et la fréquence maximum entrée dans les paramètres, un indice de récupération (pour l'entraînement effectué) pourra être calculé.

3.2.2 Développement

Les recherches et la documentation sur internet ont permis de trouver des solutions rapides pour la base de données, la gestion du Swipe lors de l'effacement ou encore le choix du timing.

Méthodologie :

La première étape consistera à découper le travail en tâches plus petites. La méthode la plus efficace est de travailler par activité. Création d'une activité, création de son interface, puis création des méthodes et des fonctionnalités requises pour cette activité et ainsi de suite. Ce choix est motivé par le fait qu'il y a une hiérarchie dans les activités et qu'il n'est pas possible de naviguer dans une application comme sur un site internet.

Fonctionnalités à développer :

Pour activité illustrée à l'image 3.4 qui représente la première activité avec laquelle l'utilisateur peut interagir. L'utilisateur doit pouvoir ajouter un entraînement grâce au bouton « Ajouter Entraînement », lors de l'ajout d'un entraînement un nom et le type de l'entraînement sera demandé à l'utilisateur au moyen d'une boîte de dialogue lui permettant de remplir ces informations. Il doit pouvoir supprimer l'entraînement lors du swipe ou voir les éléments présents dans l'entraînement en cliquant sur la ligne de l'entraînement. Chaque ajout d'un nouvel entraînement entrainera automatiquement la sauvegarde de celui-ci dans la base de données.

L'activité de l'image 3.7 (activité éléments d'entraînement) représente les séquences d'un entraînement. La même structure d'activité en ligne est utilisée pour afficher les éléments que celle présente dans l'activité précédente (image 3.4), le swipe pour effacer un élément est présent lui aussi, cependant pour ajouter une séquence il faut utiliser la croix bleu ou bouton « ajouter » situé en haut à droite dans l'Action Bar (cf [Lexique](#)). Le clic sur cet élément de l'interface lance l'activité Détails permettant d'ajouter une séquence à notre entraînement. Le bouton démarrer l'entraînement situé en bas de l'activité permet quant à lui de lancer l'activité Timer et donc de démarrer l'entraînement. La modification d'une séquence enregistrée se fait par clic sur la ligne que l'utilisateur souhaite modifier. S'ouvre alors l'activité Détails avec les données de notre séquence.

L'activité Détails présentée à l'image 3.8 est l'activité qui s'ouvre si l'on souhaite ajouter une séquence à l'entraînement. Il est possible aussi possible d'ouvrir cette activité si l'utilisateur désire modifier une des séquences de son entraînement. Ajouter une séquence ouvre l'activité détails avec des valeurs vides. Modifier une séquence ouvre la même activité avec les valeurs de la séquence que l'on souhaite modifier. Il faudra, lors de la validation, vérifier qu'il s'agit bien de la modification d'une séquence déjà existante afin de ne pas rajouter une nouvelle séquence.

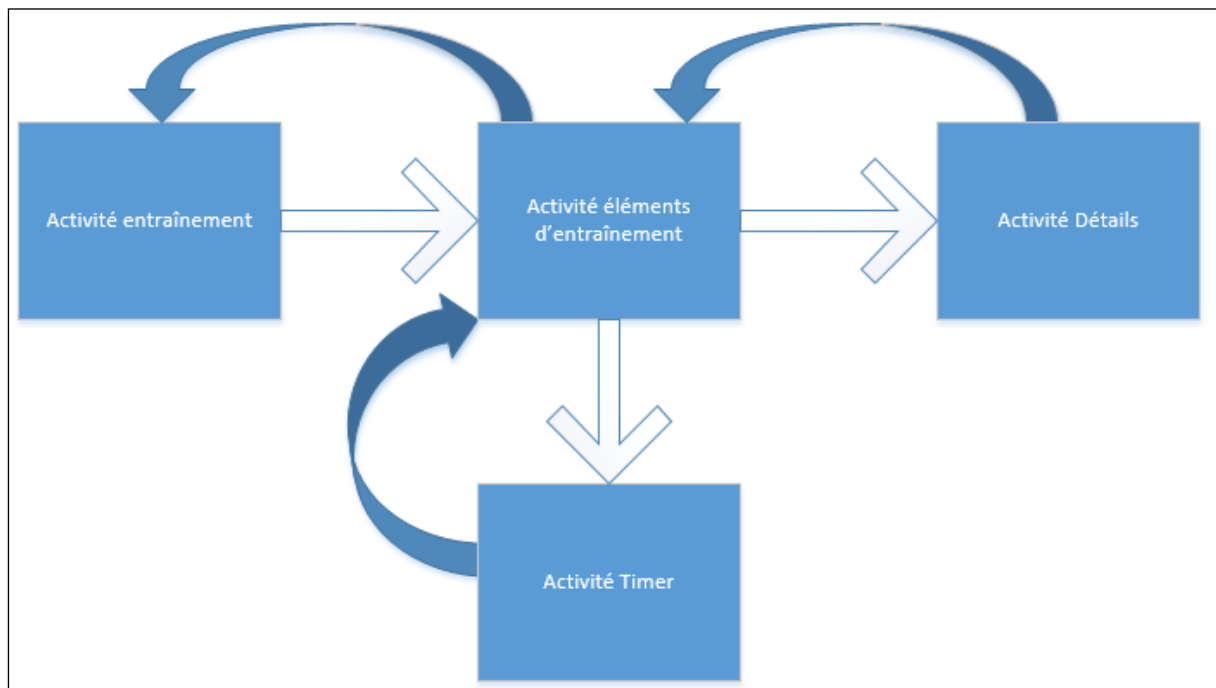
La dernière activité est l'activité timer présentée par l'image 3.9. C'est l'activité principale de l'application, car c'est cette activité qui permet à

L'utilisateur (cycliste) d'effectuer son entraînement. Quand l'utilisateur lance son entraînement, l'activité Timer prend alors le temps total de l'entraînement, ainsi que le temps et les informations issues de la première séquence de l'entraînement. Elle affiche alors les remarques, plateaux, Bpm, Rpm et toutes les autres informations présentes dans la séquence et démarre les timers du temps total de l'entraînement et du temps de la séquence. Une fois la séquence terminée, les données et le temps de la séquence suivante sont affichées mettant à jour les informations déjà présentes dans l'activité. Le timer de l'entraînement continue de tourner normalement, mais le timer de la nouvelle séquence se met à jour avec le temps correspondant et redémarre à zéro. La partie de gestion des timers est quelque chose de compliqué qu'il faudra bien gérer afin de ne pas laisser des timers tourner alors que l'application n'est plus active.

Les paramètres de l'application permettront d'ouvrir une boîte de dialogue. Cette boîte est nécessaire pour l'enregistrement de deux valeurs dans la base de données. La fréquence cardiaque de repos et la fréquence cardiaque maximale.

A la fin de l'entraînement, une boîte de dialogue s'ouvre et permet à l'utilisateur d'entrer sa fréquence de récupération (Fréquence obtenue 1min30 '' après la fin de l'entraînement). Un indice est calculé à partir de cette fréquence et de la fréquence max entrée dans les paramètres.

3.2.3 Hiérarchie des activités/vues

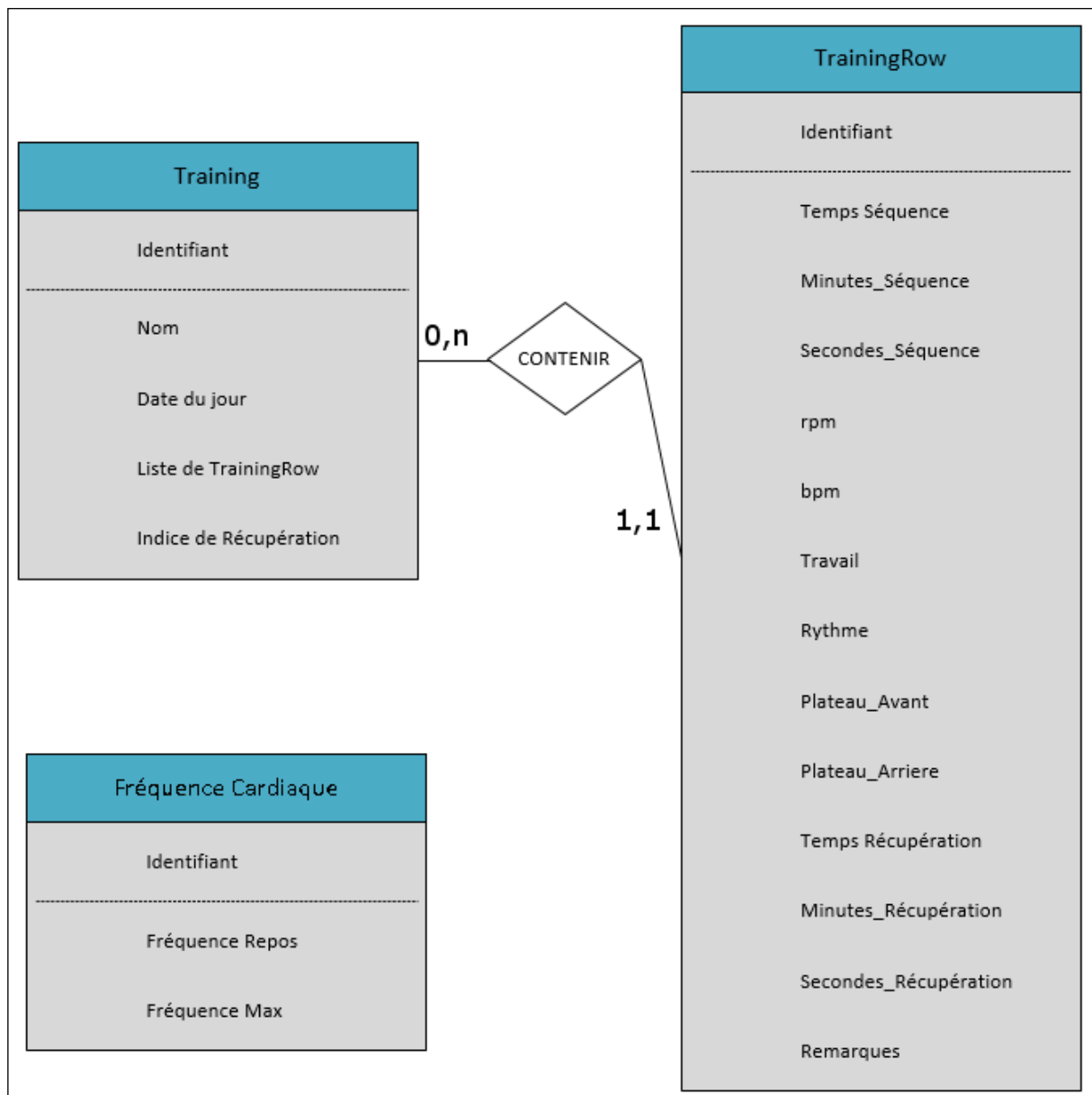


3.12 Hiérarchie des activités

La navigation ne s'effectuant pas comme sur un site internet, l'image ci-dessus représente les activités qui peuvent être appelées ou déclenchées par l'activité en cours. Le schéma permet de voir qu'il est, par exemple, impossible pour l'activité entraînement de déclencher (ouvrir) l'activité timer ou détails. Chaque activité a une fonction retour, qui, sur Android est gérée par le bouton « retour » présent sur chaque téléphone. L'activité Timer aura cependant un retour vers l'activité éléments d'entraînement qui mettra fin à l'entraînement en cours entraînant la perte de la progression déjà effectuée par l'utilisateur.

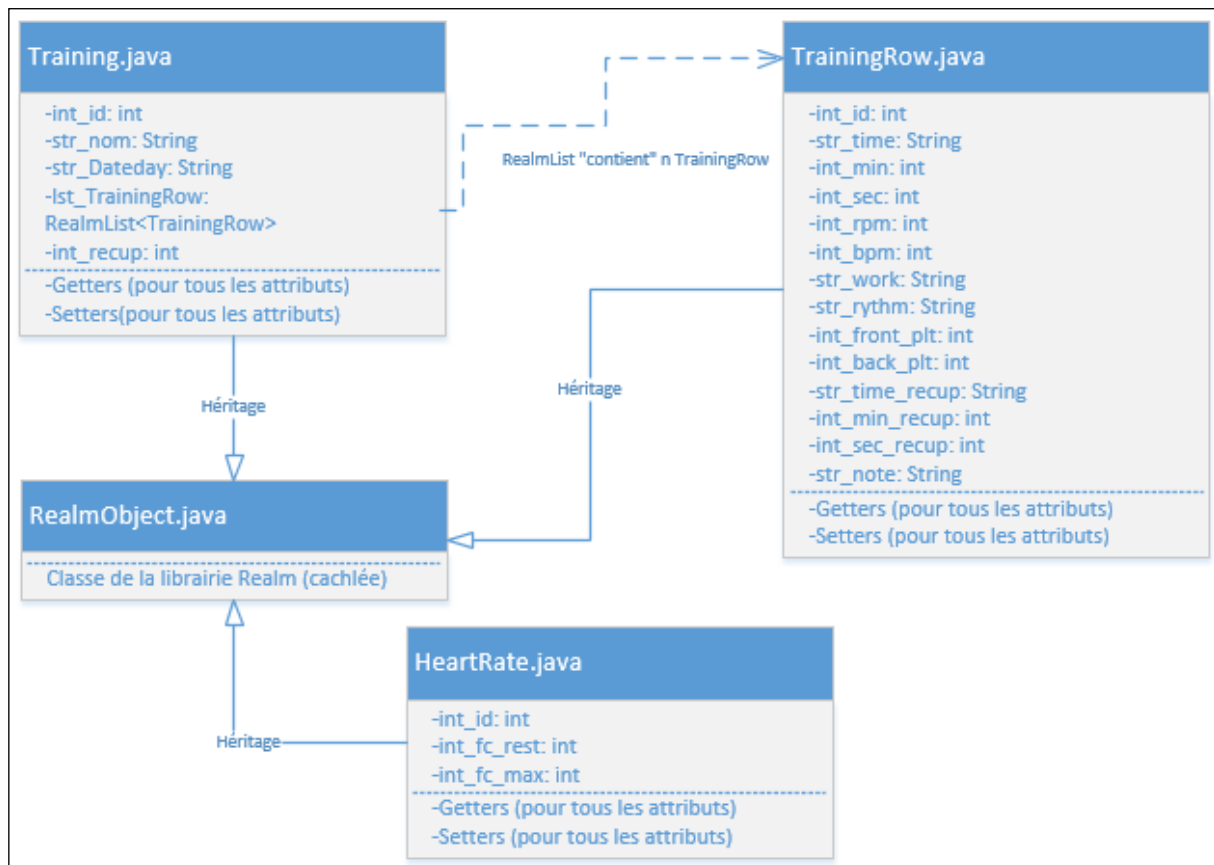
3.2.4 Base de Données

Realm sera utilisé pour la base de données. Cette utilitaire/librairie crée à partir d'une classe Java une table base sur la classe automatiquement. Cette partie n'étant pas modifiable, seul le MLD est présenté ici : Une seule base de données avec 3 tables. Une table pour les entraînements (Training) une table pour les séquences (TrainingRow) et une table pour les fréquences Cardiaques. La table fréquence cardiaque n'a pas de lien avec les deux autres. La table Training a un lien avec la table TrainingRow, car Training peut contenir 0 ou plusieurs TrainingRow. TrainingRow appartient à un et un seul entraînement.



3.13 MCD de la base de données

Cependant, la base de données étant gérée grâce aux classes, un model UML de Design de classes permettra de compenser l'absence de MLD, MPD



3.14 Design de classes UML

Le design de classes UML, remplace le MLD, MPD. Trois classes java héritent de la super classe RealmObject provenant de la librairie Realm. Realm s'occupe de convertir les classes qui héritent RealmObject.java. Cet héritage permet de faire le lien entre la classe créée et la sauvegarde des données dans la base.

Training.java gère les entraînements. TrainingRow.java gère les séquences dans un entraînement. Pour représenter un lien 0 à plusieurs (0 à n) avec Realm, il suffit d'instancier un tableau/liste (RealmList) contenant des objets.

Dans l'UML, la classe Training.java (entraînements) possède une RealmList (lst_TrainingRow) d'objets TrainingRow (séquence), au même titre qu'un entraînement possède 0 ou plusieurs séquences.

3.3 Conception des tests

La partie des tests est complexe et beaucoup de fonctionnalités devront être testées sur le moment, car la hiérarchie des vues sous-entend que si une fonctionnalité n'est pas implémentée correctement, celles dépendantes de la première ne pourront pas non plus fonctionner. Le temps imparti en fin de planification pour les tests finaux devra être utilisé correctement, mais il est réservé pour les tests finaux à savoir le comportement général de l'application.

Aucune méthodologie spécifique n'est prévue. Chaque bouton a une action spécifique, chaque partie de l'application doit se comporter comme décrit dans la partie analyse graphique et fonctionnalité. Si l'une des fonctionnalités ne s'exécute pas correctement lors d'une vérification (compilation de l'application pour vérifier le comportement de l'application), cela pourra être assimilé à un bug qu'il faudra résoudre. De par le fait que ce genre de contrôle fait partie du développement de l'application, elle s'intègre dans la planification sous l'élément « programmation ». En effet, si une fonctionnalité de clic, d'ouverture de nouvelle activité ne fonctionne pas, il faut résoudre le problème avant d'avancer, voilà pourquoi il n'y a pas de temps réservé aux tests avant la dernière semaine du projet.

3.4 Planning détaillé

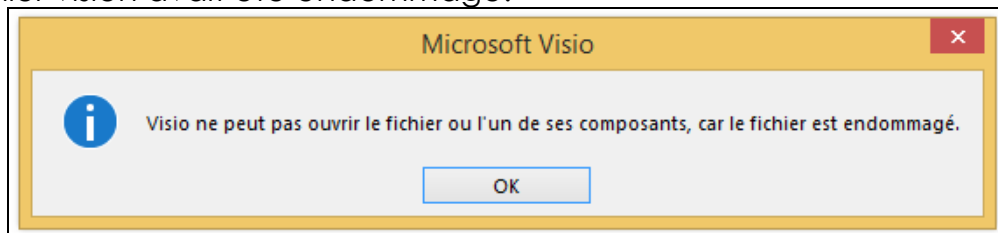
Selon les indications du chef de projet, le lien vers la planification détaillée se trouve en annexe.

4 Réalisation

4.1 Dossier de réalisation

4.1.1 Installation et préparation de l'environnement de travail

- Office Visio étant absent de l'environnement de travail il a été nécessaire de trouver une Machine Virtuelle ayant la suite office installée dessus. La création des schémas s'est faite à l'intérieur de l'environnement virtuel. Un problème est survenu lors de la reprise du travail, le lendemain, car le fichier vision avait été endommagé.



4.1 Erreur du fichier Visio

- Une partie du travail a donc dû être refaite et environ dix quarts d'heures ont été utilisés pour refaire les schémas et interfaces.
- Installation de Android Studio. Version 24.2
- Téléchargement de L'IDE (cf. [Lexique](#)) & du SDK (cf. [Lexique](#))
- Fichier *.exe. Installation Basique.
- Création d'un nouveau projet sous Android Studio.
- Installation des libraires pour débiter le développement.
Pour l'installation des librairies dans Android Studio il suffit d'ajouter une ligne dans le fichier build.gradle (cf. [Lexique](#)) de l'application.
- Exemple : compile 'io.realm:realm-android.0.80.1'

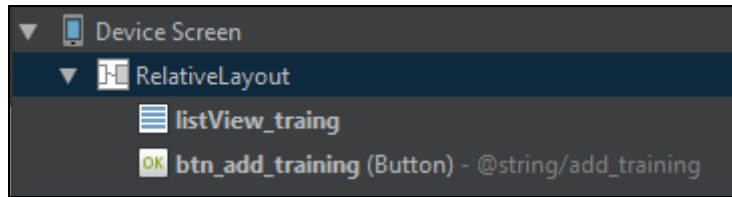
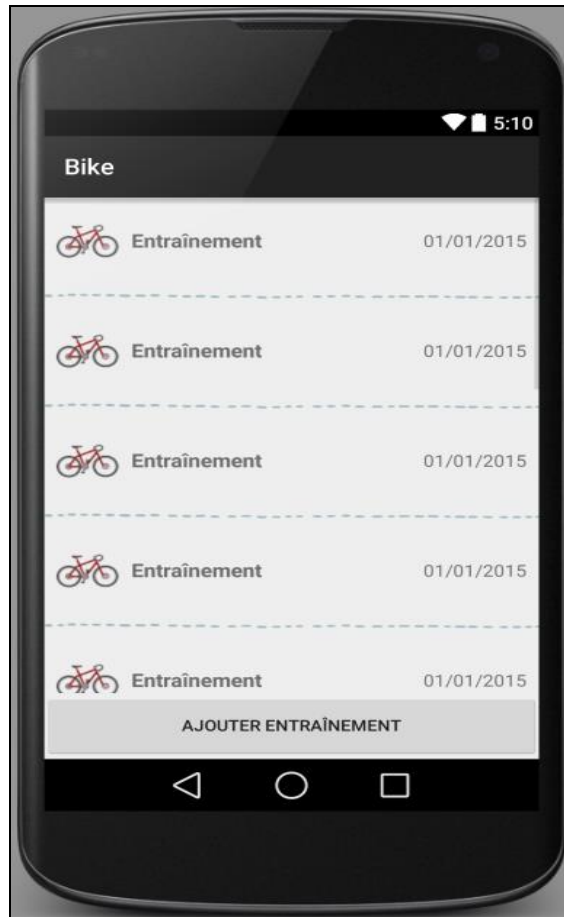
```
dependencies {  
    compile fileTree(dir: 'libs', include: ['*.jar'])  
    compile 'com.android.support:appcompat-v7:22.0.0'  
    //android Realm Library  
    compile 'io.realm:realm-android:0.80.1'  
    /* android library for swipe gesture */  
    compile 'com.android.support:recyclerview-v7:21.0.0'  
    compile 'com.android.support:support-v4:22.+'  
    compile "com.daimajia.swipelayout:library:1.2.0@aar"  
    /* android Yoyo Library */  
    compile 'com.nineoldandroids:library:2.4.0'  
    compile 'com.daimajia.easing:library:1.0.1@aar'  
    compile 'com.daimajia.androidanimations:library:1.1.3@aar'  
    //android Better Picker Library  
    compile ("com.doomonafireball.betterpickers:library:1.6.0") {  
        exclude group: 'com.android.support', module: 'support-v4'  
    }  
}
```

4.2 Installation des librairies

Le fichier build.gradle contient aussi les versions d'android sur lesquelles le programme pourra fonctionner. Chaque version ou « API Level » correspond à une version de la plateforme. Par exemple, la plus petite version de l'api est la 1 et la plus grande la 22. Une table de mise en relation entre le niveau de l'api et la version d'Android est disponible en annexe. Cette application a pour minimum une version 11 et une version cible à 22. Ce choix c'est fait car certaines libraires requièrent une version minimum. Dans notre cas une des libraires utilisée n'aurait pas fonctionnée sur une version 8, le choix c'est donc imposé à nous. Cela n'est en rien grave, car l'application reste compatible avec plus de 95% des smartphones utilisant Android comme système d'exploitation.

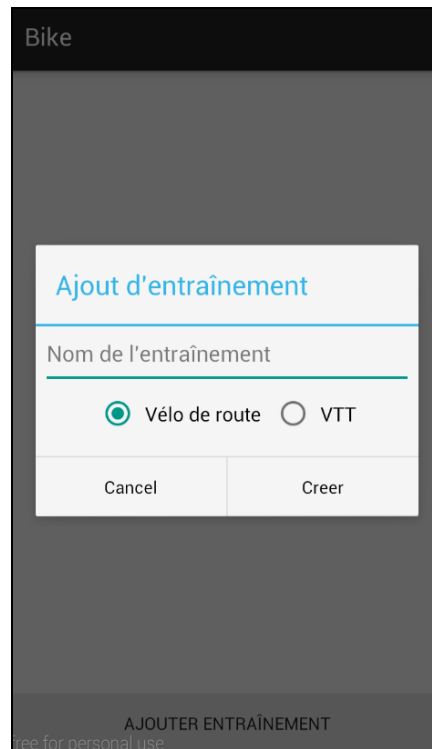
4.1.2 Design des premières interfaces

Création du layout pour la classe Training_activity.java. Un container pour le layout de type « RelativeLayout » contenant une liste de type « ListView » pour afficher les entraînements au fur et à mesure. Création d'un Bouton qui va lancer la boîte de dialogue.

**4.3 Hiérarchie des éléments de l'activité****4.4 Interface Training_activity.java**

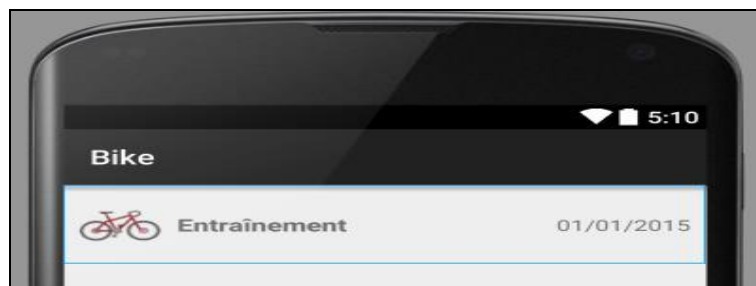
Dans un layout dit « Relative » (RelativeLayout) les éléments sont positionnés les uns par rapport aux autres. C'est l'avantage principal de ce layout. Il permet d'être sûr qu'un élément sera toujours en dessus ou à droite d'un autre et cela peut importer, la taille de l'écran ou le téléphone utilisé. La ListView contient, elle un autre layout (Voir Image : 4.6) qui représente une ligne. C'est la ListView qui se charge de prendre le layout pour la ligne et de le multiplié par rapport au nombre d'élément qu'il y a dans la liste.

La boîte de dialogue pour entrer un entraînement est implémenté aussi. Il faut mettre les deux radios boutons dans un seul radio groupe afin que lorsqu'un des éléments est sélectionné, cela désélectionne automatiquement l'autre. Le layout est créer et le test d'affichage fonctionne.

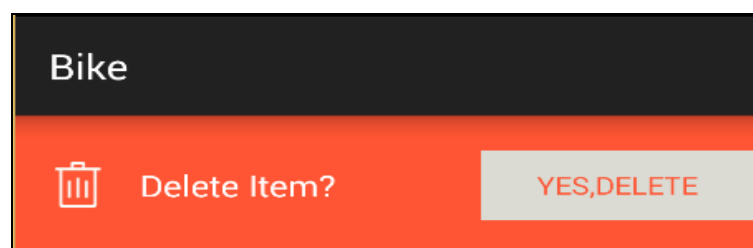


4.5 Boîte de dialogue pour l'ajout d'entraînement

Le layout pour une seule ligne d'entraînement. Comporte un Custom Layout nommé swipe. Ce type de layout est issu de la librairie AndroidSwipeLayout qui permet de faire apparaître un second layout par un geste de swipe vers la gauche.

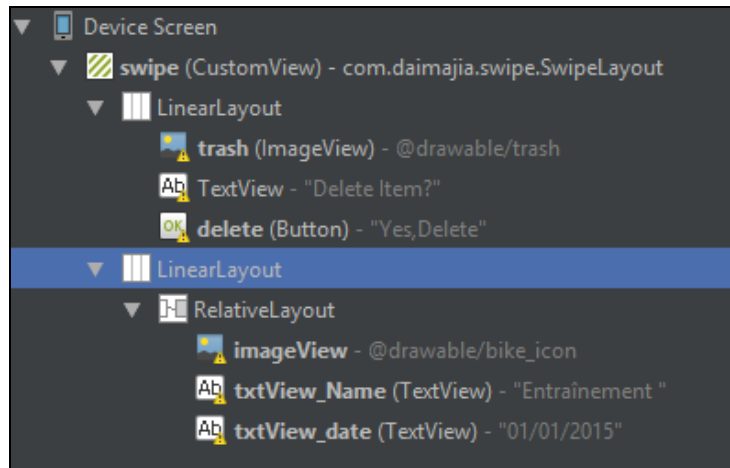


4.6 Layout d'un élément de la liste



4.7 Layout qui apparaît lors du swipe

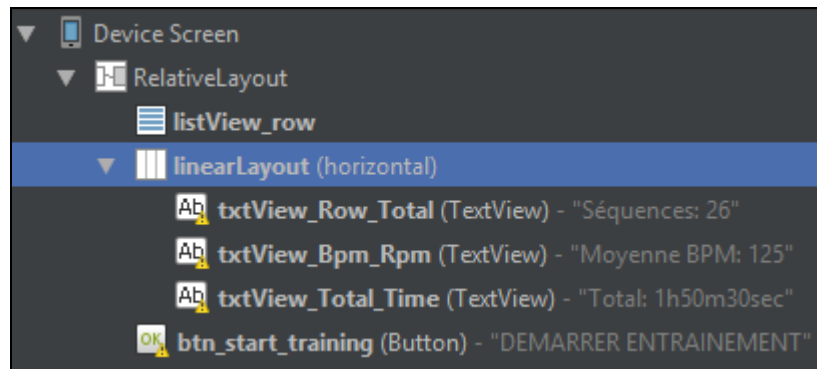
Le layout « swipe » contient ici un premier LinearLayout avec 3 éléments. « trash » qui est l'icône de la poubelle. Une TextView pour la partie Texte, qui demande si l'on veut vraiment supprimer l'entraînement et un « Button » pour lancer l'action et effacer l'entraînement. Cette partie n'est pas visible, car c'est la partie qui apparaît lors du swipe.



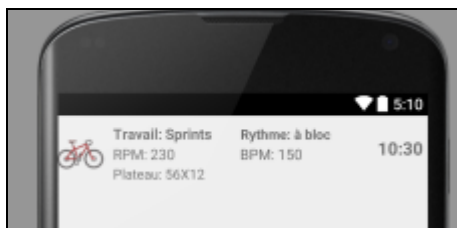
4.8 Hiérarchie des éléments

Le layout qui recouvre ce premier layout est lui aussi un « LinearLayout » qui contient une « ImageView » qui permet d'afficher soit le vélo Bleu pour le vtt ou le vélo rouge pour le vélo de route. Deux « TextView » permettent l'affichage du nom de l'entraînement et de la date. Le tout est contenu dans un « RelativeLayout » afin que tous les éléments s'affichent les uns par rapport aux autres. Le « RelativeLayout » a été rajouté ici car, lorsque l'on utilise seulement le « LinearLayout » les dates ne s'alignent pas à droite, mais se collent au nom de l'entraînement, ce qui donne un effet peut esthétique.

Le layout de l'activité TrainingRow.java est similaire à celui de la classe Training.java. Une liste, contenant toutes les séquences d'un entraînement et permettant aussi l'effacement grâce au swipe. Le fonctionnement est identique à l'interface affichant les entraînements. L'implémentation est problématique, car il faut faire attention au type de Layout que l'on va utiliser. Le layout contient un « RelativeLayout » qui lui contient 3 éléments qui sont, une « ListView » un « LinearLayout » et un « Button » pour démarrer l'entraînement. Ce bouton lance l'activité Timer.java. Le « LinearLayout » permet d'afficher une barre de résumé affichant, 3 « TextView » qui affichent respectivement, le nombre de séquences dans l'entraînement, la moyenne des BPM et le temps total pour l'entraînement. Cette partie du layout n'était pas présente dans la maquette graphique du départ. Cet ajout a été motivé pour améliorer l'expérience utilisateur. En effet, il est pratique pour un utilisateur souhaitant s'entraîner de connaître quelques informations sur l'entraînement qu'il désire effectuer avant de le démarrer. Si le temps est trop long, il pourrait alors choisir de faire un autre entraînement.



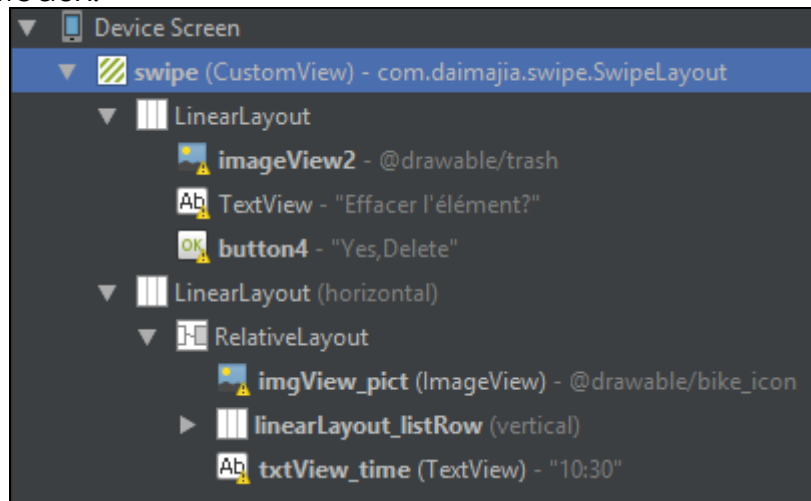
4.9 Eléments du layout de l'activité TrainingRow.java



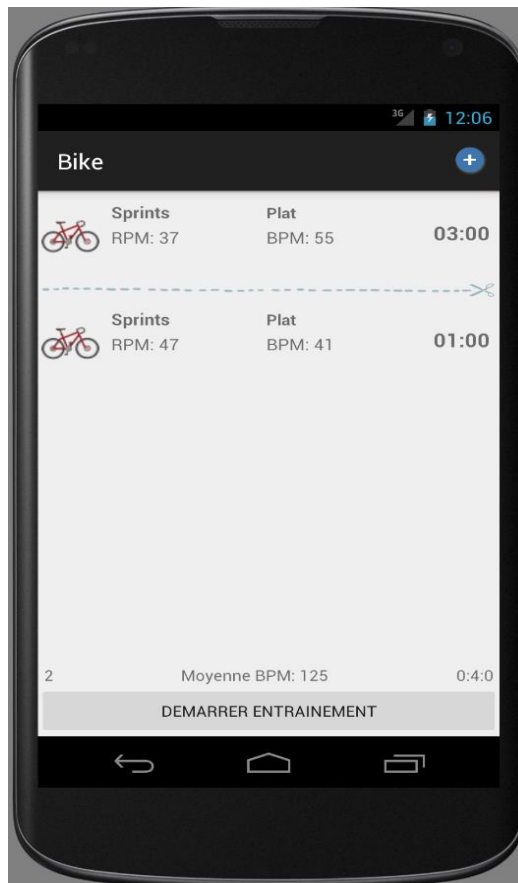
4.10 Layout d'une ligne de la liste des séquences d'entraînement

L'image 4.10, représente une ligne de la « ListView ». Comme pour la « ListView » de l'Activité Training, une « CustumView » de la librairie AndroidSwipeLayout. Le premier « LinearLayout » est celui de l'élément du dessous qui ne sera visible qu'après avoir swipé. Le second « LinearLayout » est celui de l'affichage des données avec un « RelativeLayout » à l'intérieur pour faciliter l'affichage des éléments les uns par rapport aux autres. Une « ImageView » pour l'image, un sous-layout de type « LinearLayout » (LinearLayout_listRow) permettant l'affichage de certaines valeurs propres à

la séquences, comme le type de « Travail », le « rythme », les BPMs, les RPMs, et/ou les plateaux.



4.11 Hiérarchie des éléments d'une séquence d'entraînement



4.12 Layout de l'activité TrainingRow.java

L'activité TrainingRow.java a besoin de deux boutons. Un pour démarrer l'entraînement (btn_start_training)(cf. Image 4.9) et un autre bouton pour ajouter une séquence d'entraînement et lancer l'activité TrainingRowDetailsActivity.java. Ce second bouton n'est pas matérialisé dans

les éléments de l'écran, car il s'affiche sur l'ActionBar qui est la barre où le titre est affiché. Pour implémenter ce bouton il faut créer trois choses. En premier un fichier XML pour le bouton. Il utilise deux images correspondant aux deux états du bouton. Une lorsqu'on appuie dessus, l'autre dans son état normal.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:state_pressed="true"
        android:state_enabled="true"
        android:drawable="@drawable/rond_pressed" />
    <item
        android:state_enabled="true"
        android:drawable="@drawable/rond_over" />
</selector>
```

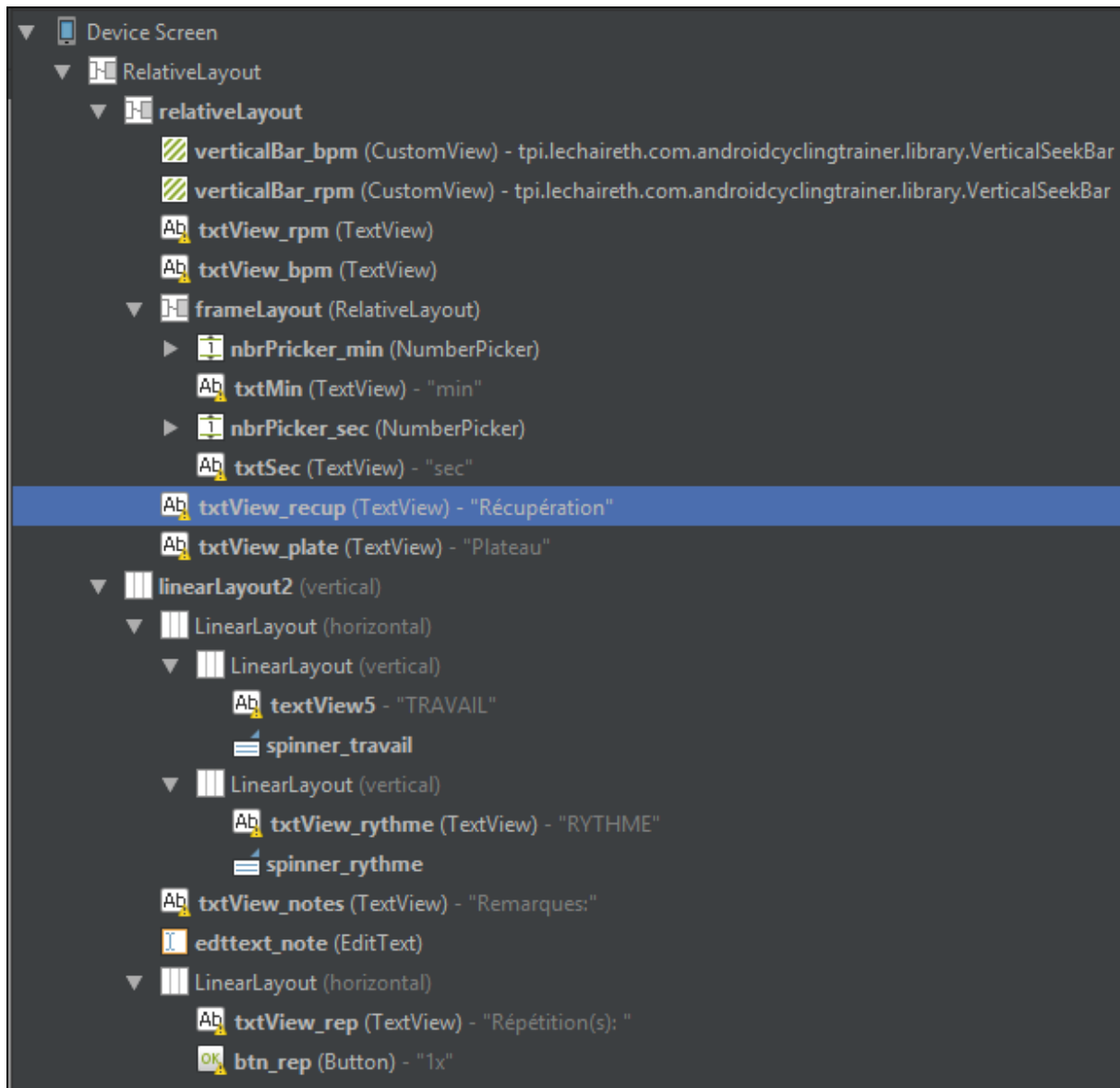
4.13 Fichier XML pour le bouton

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:compat="http://schemas.android.com/apk/res-auto">
    <item android:id="@+id/action_add" android:title="Ajouter"
        android:icon="@drawable/add_button" compat:showAsAction="always" />
</menu>
```

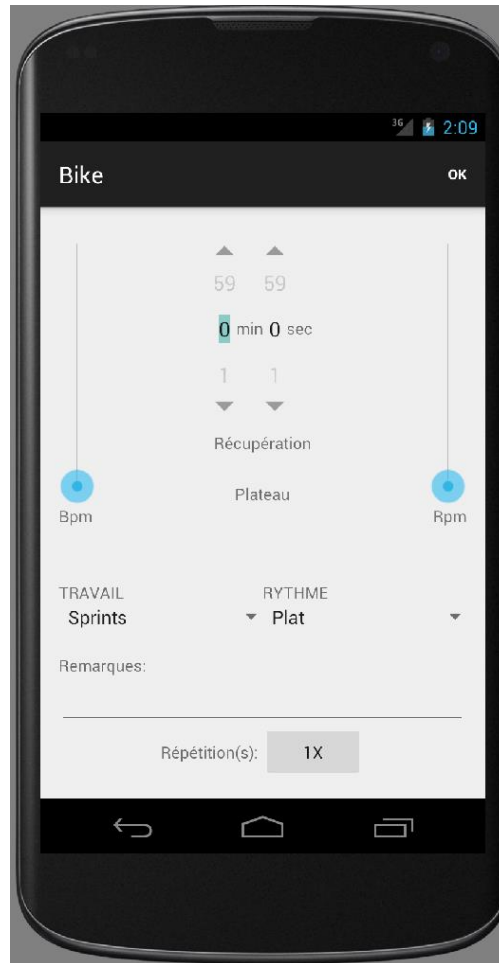
4.14 Menu du bouton ajouter

Le bouton est ajouté dans le code lors de la compilation et non pas prédéfinie dans le layout, il a pour attribut « icon » le fichier XML créé avant « add_button.xml ». C'est pourquoi en plus du fichier XML pour le bouton et pour le menu, il faut ajouter deux méthodes dans l'Activité en question. Une pour dire au téléphone quel est le fichier qui contient le menu (onCreateOptionsMenu) et l'autre pour donner les actions à exécuter lors du clic sur ce menu (onOptionsItemSelected). Lors du test, le bouton ne voulait pas apparaître. L'ajout de l'attribut compat:showAsAction= « Always » dans le fichier du menu a résolu le problème d'affichage.

C'est lors de l'ajout d'une séquence que le layout de l'Activité TrainingRowDetailsActivity.java est utilisé. Il comprend tous les éléments pour enregistrer une séquence. Plusieurs Layout sont imbriqués les uns dans les autres afin de garantir la structure sur n'importe quel téléphone Android. Tous les éléments de l'image 4.16 sont présents dans la hiérarchie.

**4.15 Hiérarchie de l'Activité TrainingRowDetailsActivity**

Deux « VerticalBar » de la librairie Vertical-SeekBar-Android. Deux TextView pour afficher les valeurs des BPM et de RPM choisies grâce aux « SeekBar ». Deux « NumberPicker » permettant de sélectionner les minutes et les secondes de la séquence. Deux « TextView » cliquables pour ajouter le temps de récupération et le choix des vitesses du vélo. Le travail et le rythme ouvrent des « Spinners » ou plus communément appelés menus déroulant. Ils permettent une sélection parmi plusieurs types de travail ou rythme prédéfinis. Le bas de l'écran laisse la place à l'utilisateur d'ajouter une remarque sur la séquence et grâce à un bouton de choisir le nombre de fois qu'il souhaite répéter une séquence.



4.16 Layout de l'activité TrainingRowDetailsActivity

Une modification a été effectuée sur ce layout par rapport à la maquette graphique. (cf. modification du 26.05.2015).

Le bouton pour valider la séquence se trouve dans l' « ActionBar » et n'apparaît donc pas dans la hiérarchie des éléments du layout. Comme pour l'activité précédente un fichier de menu est nécessaire, ainsi que l'implémentation des deux méthodes vues précédemment. (onOptionsItemSelected et onCreateOptionsMenu)

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:compat="http://schemas.android.com/apk/res-auto">
    <item android:id="@+id/action_validate" android:title="OK" compat:showAsAction="always"/>
</menu>
```

4.17 Fichier XML du menu de l'activité TrainingRowDetailsActivity

Pas besoin d'un fichier pour faire apparaître une icône (cf. image 4.13) car ici seul du texte est affiché et non pas une image comme précédemment.

4.1.3 Implémentation de la base de données

Création de 4 classes permettant l'interaction avec la base de données Realm.

Classe Training.java

Cette classe est celle permettant la sauvegarde des entraînements. (cf. Voir Annexe). Elle est implémentée selon le modèle de classe présent sur l'image 3.14. Une modification a été cependant apportée suite à un problème rencontré (cf. Modification du 28.05.2015). En effet, il faut sauvegarder une information supplémentaire concernant l'entraînement, afin de savoir s'il s'agit d'un entraînement de vélo de route ou de VTT.

Classe TrainingRow.java

Classe représentant les séquences d'entraînements contenues dans un entraînement (cf. Voir Annexe).

Classe HeartRate.java

Classe représentant la fréquence maximum et minimum de l'utilisateur.

Classe RealmDB.java

Classe contenant les méthodes d'interaction avec la base de données.

Les classes Training, TrainingRow et HeartRate héritent de la classe RealmObject. La librairie Realm vient alors utiliser les méthodes (getters et setters) des classes en questions, permettant ainsi de sauvegarder, de restaurer ou encore modifier les objets de ces classes.

RealmDB contient de nombreuses méthodes. Durant l'implémentation de ces méthodes il faut faire attention car chaque méthode doit récupérer une instance de Realm avant de pouvoir interagir avec la base de données. Pour chaque interaction avec la base de données Realm, il faut entourer les interactions de deux lignes de codes.

`Realm.beginTransaction()` ; et `Realm.commitTransaction()` ou `Realm.cancelTransaction()` si l'on ne souhaite pas sauvegarder les interactions. Durant les premières phases d'essais de sauvegarde des entraînements un problème avec ces méthodes est survenu. L'entraînement était sauvegarder correctement et sans erreur. Cependant, lorsque l'on souhaitait récupérer les entraînements sauvegardés l'application avait tendance à s'arrêter.

Modification/suppression à rédiger.....

La classe est disponible en annexe.

4.1.4 Implémentation des deux « ListView »

Les deux « ListView » sont les deux listes présentes dans l'Activité TrainigActivity et dans TrainingRowActivity. Ces deux classes font appel à ce qu'on appelle un

Adapter. Un adaptateur est une classe java qui va servir au programme pour traiter toutes les lignes d'une liste et permettre d'avoir une affichage et des interactions identique sur chaque une des lignes présentes dans la liste. Il a fallut ici utiliser deux Adapter personnalisés afin de pouvoir intégrer la gestion du swipe ou encore celle du clic sur une ligne. La librairie gérant le swipe avait deux méthodes à implémenter automatiquement. Il est important alors de comprendre la différence entre les méthodes de bases d'un Adapter personnalisé et celle issues de cette librairie.

Le premier adapter, celui lié a la liste des entraînements, gère trois actions. Le clic sur l'entraînement, qui va alors ouvrir l'Activité TrainigRowActivity avec la seconde « ListView » et le second adapter. Le swipe vers la gauche, qui va laisser apparaitre le second layout caché, mais présent dans la hiérarchie de l'image 4.8 et enfin, le clic sur le bouton apparu suite au swipe, effaçant l'entraînement choisi. La récupération de l'identifiant de la ligne sélectionnée est important, car il permet d'être sur que l'on efface ou ouvre le bon entraînement. Cependant, en commençant a implémenter la librairie gérant le swipe un problème avec la récupération de l'identifiant est apparu. La méthode pour le clic était correcte, mais l'identifiant prenait la valeur zéro au lieu de prendre la valeur de la ligne sélectionnée. C'est sur le forum de la librairie que la solution a été trouvée. En effet, cette librairie a découpé en deux méthodes (NOM DES MÉTHODES) la méthode getView qui est la méthode principale permettant le remplissage des lignes de la liste. Dans notre cas, la position était égal a zéro car les listeners n'étaient pas implémentés dans la bonne méthode parmi les deux. Déplacer les listener dans la seconde méthode a suffi à résoudre ce problème.

La seconde « ListView » présente dans l'Activité TrainingRow permet l'affichage de toutes les séquences de l'entraînement sur lequel l'utilisateur clic dans l'Activité précédente. Un Toast (cf. [Lexique](#)) s'affiche lorsque l'utilisateur n'a pas encore ajouté de séquence dans l'entraînement en question. L'adapter ici est mis en place rapidement, car les erreurs lors d'implémentation de la première « ListView » on permis d'éviter une nouvelle perte de temps.

- Un problème avec le swipe est apparu.
Une mauvaise reconnaissance du geste et un problème lors de l'affichage. Problème résolu en enlevant la ligne de code :
`SwipeLayout.setShowMode(SwipeLayout.ShowMode.LayDown);`

4.1.5 Timer

Lors de la création du timer, beaucoup de problèmes ont été rencontré. Les premiers timers pour l'entraînement ont été fait grâce à une AsyncTask qui

tournait en tâche de fond. Cela permet d'éviter d'utiliser la mémoire et le Thread principal. Cependant, l'interaction avec le Timer dans des actions comme, pause, stop ou reprise était impossible. En effet, les boutons déclenchant les commandes de pause ou de reprise font partie du thread principal et devaient interagir avec un thread en background ce qui est compliqué. De plus, la classe AsyncTask n'est utilisée que pour effectuer des tâches continues et sans interruptions. Pour corriger cela, le Timer a été implémenté avec la classe `CountDownTimer` qui est une classe possédant deux méthodes de base qui sont, `onTick` – qui prend en paramètre la variable passée en paramètre, et `onFinish` qui est appelée lorsque le décompte de `onTick` se termine. Lors de la première implémentation de cette classe privée, le timer a fonctionné. Cependant, lors de l'essai de mettre le timer en pause cela s'est compliqué. Un timer ne se met pas en pause. Il s'arrête et se recrée. Il a fallu donc créer un second timer, car un timer que l'on vient de tuer ne peut pas être recréé directement derrière. Puis des problèmes d'affichage sont apparus car, les objets Realm ne sont accessibles que dans le thread où ils ont été créés. Il a fallu trouver un moyen pour sauvegarder toute la séquence dans des variables. Neuf variables ont été créées et prennent la valeur contenue dans la séquence en cours avant d'êtreinstanciées à nouveau lors du changement de séquence. En testant les fonctionnalités de pause et de reprise deux erreurs sont apparues. La première était la sauvegarde de la progression déjà effectuée avant le déclenchement du bouton pause. La sauvegarde des informations est la partie importante de cette fonctionnalité, car sans ces informations il est impossible de pouvoir recréer un Timer qui démarre là où l'ancien Timer s'est arrêté. Des variables de sauvegarde ont permis de récupérer correctement toutes les données afin de pouvoir recréer un nouveau timer avec les données issues de la précédente Timer. La seconde erreur concernait l'affichage du nouveau Timer, en effet après plusieurs périodes de recherches il semble que la classe `CountDownTimer` implémentée par Android ne gère pas correctement l'arrêt du timer lors de l'appel de la méthode `cancel()`, ce qui a pour conséquence de laisser tourner le premier Timer et de causer un double affichage (l'un sur l'autre) du temps des deux timers. Une solution proposée pour éviter ce problème est de modifier la classe `CountDownTimer` selon l'exemple trouvé ici : <https://code.google.com/p/android/issues/detail?id=58668>. La modification n'a rien changé au problème d'arrêt du premier timer. Il y a eu donc un conflit empêchant le timer de fonctionner correctement. Parfois le compte à rebours stoppait avant la fin du temps, parfois il dépassait la limite. Deux journées ont déjà été utilisées pour implémenter/tester ces méthodes et toujours aucun Timer ne fonctionne.

Ce paragraphe permet de reproduire ou reprendre le projet par un tiers.

- Pour chaque étape, il faut décrire sa mise en œuvre. Typiquement :
 - Versions des outils logiciels utilisés (OS, applications, pilotes, librairies, etc.)

- Configurations spéciales des outils (Equipements, PC, machines, outillage, etc.)
- Code source des éléments logiciels développés.
- – Modèle physique d'une base de données.
- Arborescences des documents produits.
- Il faut décrire le parcours de réalisation et justifier les choix.

4.2 Modifications

- 26.05.2015 - Modification de l'interface pour l'activité TrainingRowActivity.java. Une barre de résumé en bas de la liste avec le nombre de séquence dans l'entraînement, la moyenne des BPM et le temps total a été rajoutée. Cela permet à l'utilisateur d'avoir un résumé visuel de son entraînement améliorant ainsi.
- 26.05.2015 – Modification de l'interface pour l'activité TrainingRowDetailsActivity.java. Le bouton « Valider » a été remplacé suite à la demande du mandataire par un bouton permettant d'indiquer le nombre de fois que l'on désire répéter une séquence d'entraînement. Le bouton pour valider la séquence se trouve maintenant dans l'ActionBar.
- 28.05.2015 – Modification de la classe Training.java servant à la sauvegarde des entraînements par rapport au schéma de base proposé au point 3.14. Ajout d'un Boolean bln_isVtt étant égal à « true » si c'est un entraînement de Vtt et « false » s'il s'agit d'un entraînement sur route. Cet ajout permet de connaître le type d'entraînement et de modifier l'image du vélo (bleu pour Vtt et rouge pour la route) en conséquence.
-
-
- Historique des modifications demandées (ou nécessaires) aux spécifications détaillées.
 - Date, raison, description, etc.

5 Tests

5.1 Dossier des tests

- On dresse le bilan des tests effectués (qui, quand, avec quelles données...) sous forme de procédure (tableau).
- Si des tests prévus dans la stratégie n'ont pas pu être effectués :
 - raison, décisions, etc.
- Liste des bugs répertoriés avec la date de découverte et leur état :
 - Corrigé, date de correction, corrigé par, etc.

6 Conclusion

6.1 Bilan des fonctionnalités demandées

- Il s'agit de reprendre point par point les fonctionnalités décrites dans les spécifications de départ et de définir si elles sont atteintes ou pas, et pourquoi.

- Si ce n'est pas le cas, mesuré en « % » ou en « temps supplémentaire » le travail qu'il reste à accomplir pour terminer le tout.

6.2 Bilan de la planification

- Distinguer et expliquer les tâches qui ont généré des retards ou de l'avance dans la gestion du projet.

6.3 Bilan personnel

- Si c'était à refaire:
 - Qu'est-ce qu'il faudrait garder ? Les plus et les moins ?
 - Qu'est-ce qu'il faudrait gérer, réaliser ou traiter différemment ?
- Qu'est-ce que ce projet m'a appris ?
- Suite à donner, améliorations souhaitables, ...
- Remerciements, signature, etc.

7 Divers

7.1 Journal de travail de chaque participant

- Date, activité (décrit afin de reproduire le cheminement du projet), durée.

7.2 Bibliographie

- Références des livres utilisés durant le projet.

7.3 Webographie

- <http://Wikipedia.fr>
- <http://developer.android.com/>
- <https://github.com/daimajia/AndroidSwipeLayout>

8 Annexes

8.1 Version d'android

Platform Version	API Level	VERSION_CODE
Android 5.1	22	LOLLIPOP_MR1
Android 5.0	21	LOLLIPOP
Android 4.4W	20	KITKAT_WATCH
Android 4.4	19	KITKAT
Android 4.3	18	JELLY_BEAN_MR2
Android 4.2, 4.2.2	17	JELLY_BEAN_MR1
Android 4.1, 4.1.1	16	JELLY_BEAN
Android 4.0.3, 4.0.4	15	ICE_CREAM_SANDWICH_MR1
Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH
Android 3.2	13	HONEYCOMB_MR2
Android 3.1.x	12	HONEYCOMB_MR1
Android 3.0.x	11	HONEYCOMB
Android 2.3.4 Android 2.3.3	10	GINGERBREAD_MR1
Android 2.3.2 Android 2.3.1 Android 2.3	9	GINGERBREAD
Android 2.2.x	8	FROYO
Android 2.1.x	7	ECLAIR_MR1
Android 2.0.1	6	ECLAIR_0_1
Android 2.0	5	ECLAIR
Android 1.6	4	DONUT
Android 1.5	3	CUPCAKE
Android 1.1	2	BASE_1_1
Android 1.0	1	BASE

8.2 Code source

- Build.gradle exporté d'Android Studio

build.gradle

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 22
    buildToolsVersion "21.1.2"

    defaultConfig {
        applicationId "tpi.lechaireth.com.androidcyclingtrainer"
        minSdkVersion 11
        targetSdkVersion 22
    }
}
```

```
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.0.0'
    //android Realm Library
    compile 'io.realm:realm-android:0.80.1'
    /* android library for swipe gesture */
    compile 'com.android.support:recyclerview-v7:21.0.0'
    compile 'com.android.support:support-v4:22.+'
    compile "com.daimajia.swipelayout:library:1.2.0@aar"
    /* android Yoyo Library */
    compile 'com.nineoldandroids:library:2.4.0'
    compile 'com.daimajia.easing:library:1.0.1@aar'
    compile 'com.daimajia.androidanimations:library:1.1.3@aar'
    //android Better Picker Library
    compile ("com.doomonafireball.betterpickers:library:1.6.0") {
        exclude group: 'com.android.support', module: 'support-v4'
    }
}
```

- Training.java classe pour la base de données

Training.java

```
/* *****
 * Programm   : Android Cycling Trainer
 * Society    : ETML
 * Author     : Thomas Léchaire
 * Date       : 26.05.2015
 * Goal       : Class used to backup in the realm data base all Training
created
 *             by the user.
 * ***** //
 * Modifications:
 * Date         : XX.XX.XXXX
 * Author       :
 * Purpose      :
 * *****//
package tpi.lechaireth.com.androidcyclingtrainer.DB;

/* Import for the class */
import io.realm.RealmList;
import io.realm.RealmObject;

/* *****
 * Beginning of the TrainingRow class
 * *****//
public class Training extends RealmObject {
```

```
//id for the row
private int int_id;
//name of the training
private String str_name;
//creation date of the training
private String str_day;
//list of all row for the training
private RealmList<TrainingRow> rlst_row;
//int for the recup indice
private int int_recup;
//boolean to tell if the training is VTT or not
private boolean bln_isVtt;

/* Getters and Setters for all attributs */
public int getInt_id() {return int_id;}

public void setInt_id(int int_id) {this.int_id = int_id;}

public int getInt_recup() {return int_recup;}

public void setInt_recup(int int_recup) {this.int_recup = int_recup;}

public RealmList<TrainingRow> getRlst_row() {return rlst_row;}

public void setRlst_row(RealmList<TrainingRow> rlst_row) {this.rlst_row = rlst_row;}

public String getStr_day() {return str_day;}

public void setStr_day(String str_day) {this.str_day = str_day;}

public String getStr_name() {return str_name;}

public void setStr_name(String str_name) {this.str_name = str_name;}

public boolean isBln_isVtt() {return bln_isVtt;}

public void setBln_isVtt(boolean bln_isVtt) {this.bln_isVtt = bln_isVtt;}
}
```

- TrainingRow.java classe pour la base de données

TrainingRow.java

```
/* *****
 * Programm   : Android Cycling Trainer
 * Society    : ETML
 * Author     : Thomas Léchaire
 * Date       : 26.05.2015
 * Goal       : Class used to backup in the realm data base all TrainingRow
created
 *             by the user for one and only one Training.
 * ***** //
 * Modifications:
 * Date        : XX.XX.XXXX
 * Author      :
 */
```

```
* Purpose :
*****/
package tpi.lechaireth.com.androidcyclingtrainer.DB;

/* Import for the class */
import io.realm.RealmObject;

*****
* Begining of the TrainingRow class
*****/
public class TrainingRow extends RealmObject {

    //id for the row
    private int int_id;
    // Time for the display
    private String str_time;
    //minutes for the timer
    private int int_min;
    //seconds for the timer
    private int int_sec;
    //Revolution per minute
    private int int_rpm;
    //Hear Beat per minute
    private int int_bpm;
    //Type of Work
    private String str_work;
    //Intansity of the work
    private String str_rythm;
    //String for the front and back plates
    private String str_gear;
    //time for the recuperation
    private String str_time_rest;
    //minutes fot the rest time
    private int int_min_rest;
    //seconds for the test time
    private int int_sec_rest;
    //String for the notes
    private String str_note;

    /* Getters and Setters for all attributs of the class */

    public String getStr_gear() {return str_gear;}

    public void setStr_gear(String str_gear) {this.str_gear = str_gear;}

    public int getInt_bpm() {return int_bpm;}

    public void setInt_bpm(int int_bpm) {this.int_bpm = int_bpm; }

    public int getInt_id() {return int_id;}

    public void setInt_id(int int_id) {this.int_id = int_id;}

    public int getInt_min() {return int_min;}

    public void setInt_min(int int_min) {this.int_min = int_min;}

    public int getInt_min_rest() {return int_min_rest;}
```

```
    public void setInt_min_rest(int int_min_rest) {this.int_min_rest =
int_min_rest;}

    public int getInt_rpm() {return int_rpm;}

    public void setInt_rpm(int int_rpm) {this.int_rpm = int_rpm;}

    public int getInt_sec() {return int_sec;}

    public void setInt_sec(int int_sec) {this.int_sec = int_sec;}

    public int getInt_sec_rest() {return int_sec_rest;}

    public void setInt_sec_rest(int int_sec_rest) {this.int_sec_rest =
int_sec_rest;}

    public String getStr_note() {return str_note;}

    public void setStr_note(String str_note) {this.str_note = str_note;}

    public String getStr_rythm() {return str_rythm;}

    public void setStr_rythm(String str_rythm) {this.str_rythm =
str_rythm;}

    public String getStr_time() {return str_time;}

    public void setStr_time(String str_time) {this.str_time = str_time;}

    public String getStr_time_rest() {return str_time_rest;}

    public void setStr_time_rest(String str_time_rest) {this.str_time_rest
= str_time_rest;}

    public String getStr_work() {return str_work;}

    public void setStr_work(String str_work) {this.str_work = str_work;}
}
```

- Classe RealmDB permettant les interactions avec la base de données.
- Listing du code source (partiel ou, plus rarement complet)
-
- Guide(s) d'utilisation et/ou guide de l'administrateur
-
- Etat ou « dump » de la configuration des équipements (routeur, switch, robot, etc.).
- Extraits de catalogue, documentation de fabricant, etc.
- Photocopies diverses, etc.
-
- Planification initiale.

- Planification détaillée.
-
-

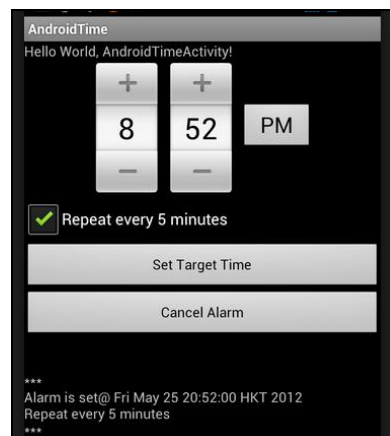
8.3 Lexique

- **ORM** : Un ORM (Object-relational mapping) ou en français mapping objet-relationnel est une technique de programmation qui crée l'illusion d'une base de données orientée objet à partir d'une base de données relationnelle. (Wikipedia)
- **Persistence** : La persistance en programmation informatique se réfère au mécanisme responsable de la sauvegarde et de la restauration des données.
- **Activité** : Une activité est une fenêtre, un élément visuel ou plus précisément l'Interface graphique de l'application. Elle est parfois appelée UI pour User Interface (Interface Utilisateurs).
- **Swipe** : est le nom d'un geste utilisé sur les téléphones portables et qui consiste à faire glisser son doigt de droite à gauche ou de gauche à droite.
- **TextView** : Une TextView est un élément présent dans Android et qui permet de recevoir du texte à l'intérieur. Il n'est pas modifiable et pourrait être assimilé à une étiquette ou un label.
- **Bpm** : Battements par minutes. Concerne le rythme Cardiaque
- **Rpm**, Rotation par minute, révolution par minute, tour par minute. Concerne le cycliste et plus précisément les tours de pédales effectuées.
- **TimePicker** : un timePicker est un élément présent dans Android qui permet d'ouvrir un pop-up afin de nous permettre de choisir le temps. Le temps peut être une date une heure ou les deux.

Exemples :



8.1 TimePicker intégré dans l'activité



8.2 TimePicker avec boîte de dialogue

- **Boîte de dialogue** : Une boîte de dialogue est une sorte de pop-up qui apparaît sur l'écran de l'utilisateur et qui permet d'interagir avec. Ci-dessus un exemple d'une boîte de dialogue contenant un TimePicker
- **Action Bar** : L'Action Bar est la barre de l'application dans laquelle le titre de l'application ou le logo est affiché. Elle est modifiable et il est donc possible d'ajouter un menu ou des boutons spécifiques.

- **IDE** : Environnement de développement. L'IDE correspond au logiciel permettant de développer notre application.
- **SDK** : pour « SoftWare Development Kit », kit de développement logiciel. Contient les compilateurs et tous les éléments nécessaires au développement (spécifique au langage et à la plateforme sur laquelle l'application est développée)
- **build.gradle** : Le fichier Build.gradle est le fichier de configuration de l'application Android. Il gère les librairies, la version de l'application ainsi que la version pour le minSdkVersion (Version d'Android la plus petite avec laquelle l'application sera compatible) et le targetSdkVersion (Version pour laquelle l'application est développée)
- **Toast** : Est un texte qui apparaît, pour une courte durée, en bas de l'écran sur un fond noir et qui permet d'offrir une information (importante ou non) à l'utilisateur.