

Rapport de projet

KeepIt



Sommaire

| | |
|--|----|
| 1 Introduction | 3 |
| 1.1 Présentation de l'application | 3 |
| 1.1 But de l'application | 3 |
| 2 Choix technologiques..... | 4 |
| 2.1 IDE & Langage..... | 4 |
| 2.2 GreenDao | 4 |
| 3.Base de données | 5 |
| structure de base de données..... | 5 |
| 4. Interface..... | 6 |
| 5 Stratégie d'implémentation..... | 10 |
| 6 Problème rencontré..... | 11 |
| Timer : | 11 |
| Base de donnée :..... | 12 |
| 7 Amélioration possible | 13 |
| Base de donnée :..... | 13 |
| Interfaces pour utilisateurs : | 13 |
| Calendrier : | 13 |
| Modification et supprime des séances : | 13 |
| Timer : | 13 |
| 8 Conclusion | 14 |

1 Introduction

1.1 Presentation de l'application

Nous avons développé une application pour le CODEP25 dans le cadre de l'organisation de ses stages de perfectionnement annuels. En effet, comme vous le savez tous, il existe une multitude d'applications permettant de gérer des entraînements mais aucune capable de gérer des entraînements intelligents et aléatoires.

1.1 But de l'application

Cette application doit permettre de s'inscrire et s'authentifier.

L'application doit permettre d'afficher et saisir tous les entraînements, son nom, sa durée, son type de public (adulte, jeune ou les deux), son niveau de difficulté, son groupe de niveau (1 à n) et ses thèmes.

L'application doit permettre de saisir des entraînements, L'application doit permettre de saisir ses séances d'entraînement sur le téléphone/tablette de manière textuelle ou photo, définir pour chaque élément constituant la séance, sa durée, son intensité et son nombre de répétitions.

L'application doit permettre de chronométrer chaque séance effectuée par entraîneur, indiquer le temps qu'il reste par paquet de multiples de 30. pendant de chronométrer, L'entraîneur peut stopper le chrono pour donner des explications.

2 Choix technologiques

On va ici présenter les techniques de notre équipe pour réaliser ce projet.

2.1 IDE & Langage

On utilise l'IDE Android Studio, avec le langage Java.

2.2 GreenDao



GreenDAO est un ORM Android open source. Il évite aux développeurs de gérer les exigences de base de données de bas niveau tout en économisant du temps de développement. SQLite est une formidable base de données relationnelle intégrée. Pourtant, l'écriture de SQL et l'analyse des résultats des requêtes sont des tâches assez fastidieuses et chronophages. greenDAO vous en libère en mappant des objets Java à des tables de base de données (appelées ORM, «mappage objet / relationnel»). De cette façon, vous pouvez stocker, mettre à jour, supprimer et interroger des objets Java à l'aide d'une API orientée objet simple.

Caractéristique:

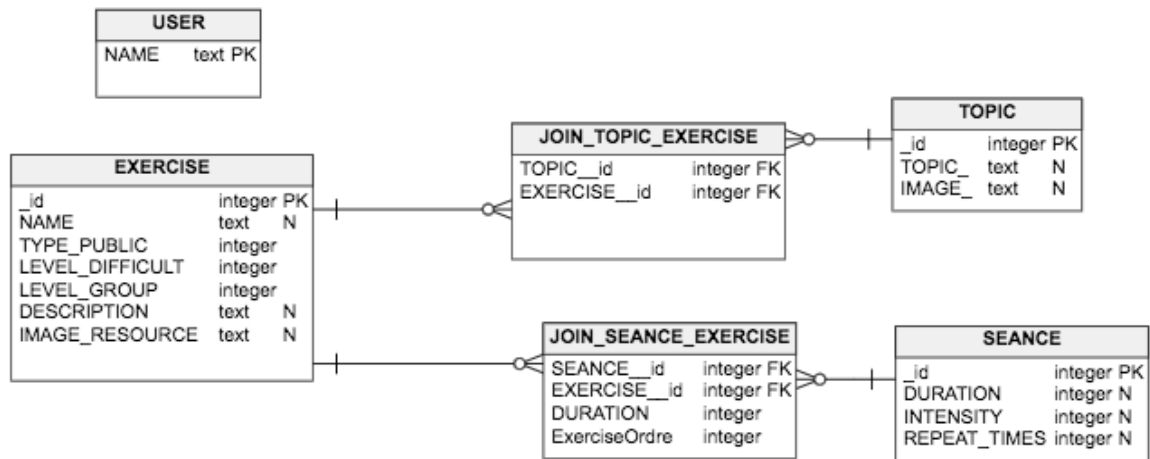
1. Performances maximales (probablement l'ORM le plus rapide pour Android).
2. API puissantes et faciles à utiliser couvrant les relations et les jointures.
3. Consommation de mémoire minimale.
4. Cryptage de la base de données: greenDAO prend en charge SQLCipher pour protéger les données de votre utilisateur

3.Base de données

structure de base de données

-4 table principale : User,Exercise, Topic, Seance

-2 table de jointure: Join_Topic_Exercise et Join_seance_exercise



4. Interface

Comme les graphiques suivant montrent, on a trois parties : Thèmes, Séance, profile.

Thèmes: thème de entraînement.

Séance: pour créer une séance

Profile : les informations totale pour utilisateur

Tout d'abord, le client se connecte dans l'application si il a déjà une compte. Si non il peut s'inscrire un nouveau compte.

Il crée un nouveau thème/ entraînement dans la partie topic

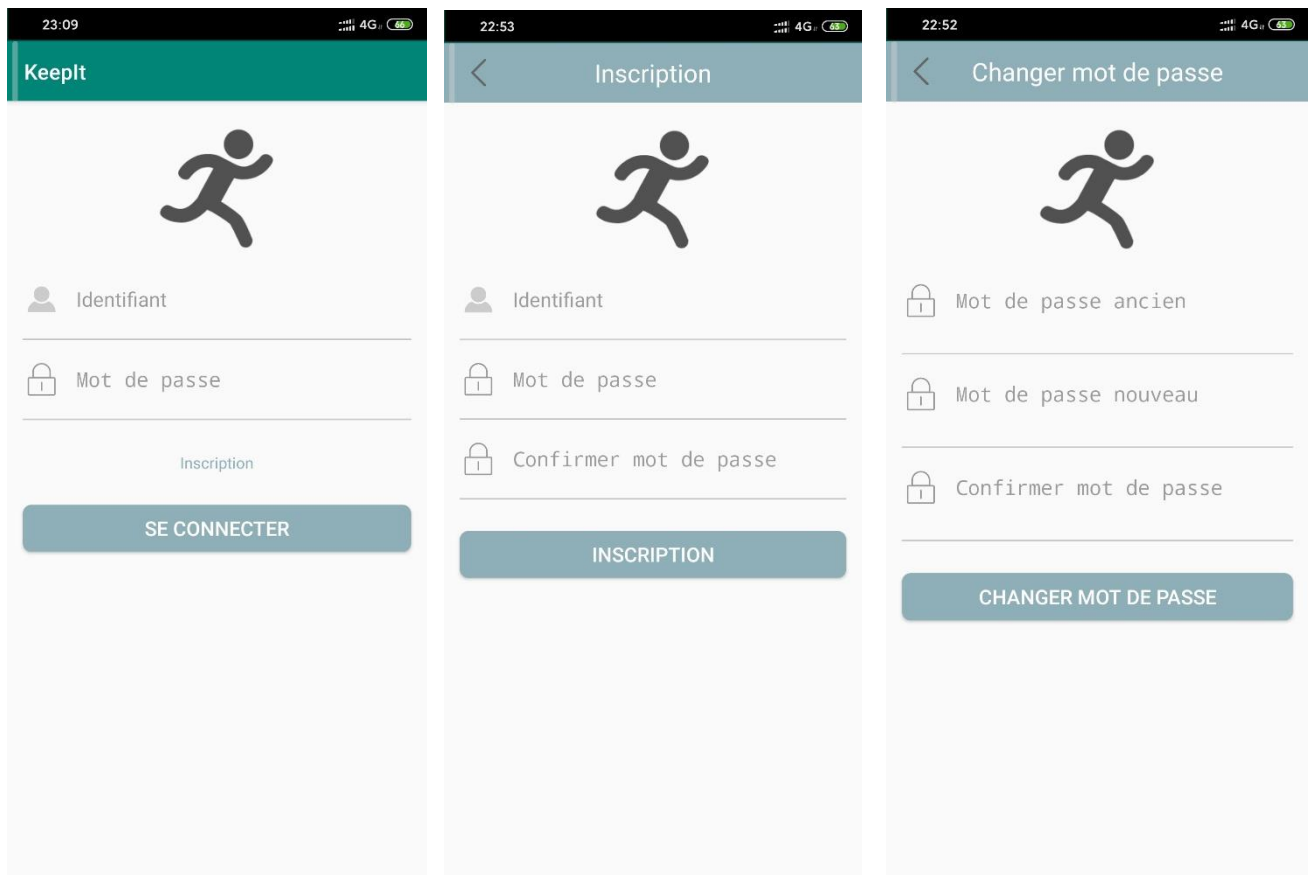
Chaque entraînement a un type de public (adulte, jeune ou les deux), un niveau de difficulté, un groupe de niveau (1 à n) et d'un ou plusieurs thèmes

Un entraîneur doit pouvoir saisir ses séances d'entraînement sur le téléphone/tablette de manière textuelle ou bien d'un scan / photo de document.

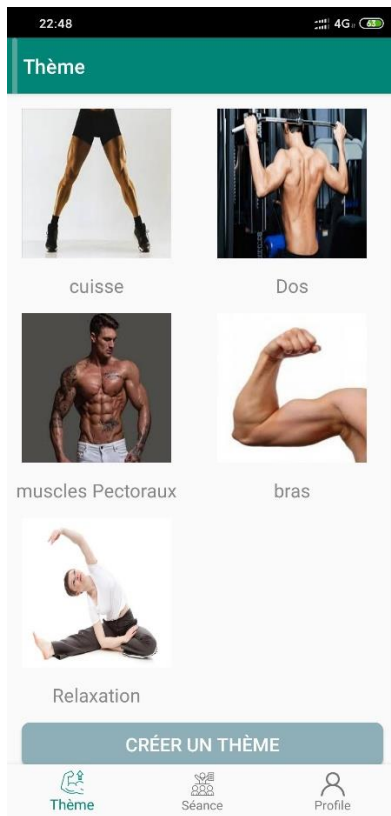
Pour creer une séance, un entraîneur doit pouvoir définir pour chaque élément constituant la séance, sa durée, son intensité et son nombre de répétitions.

Quand l'utilisateur lancer une séance, il y un timer pour recorder le temps et pour notifier l'utilisateur.

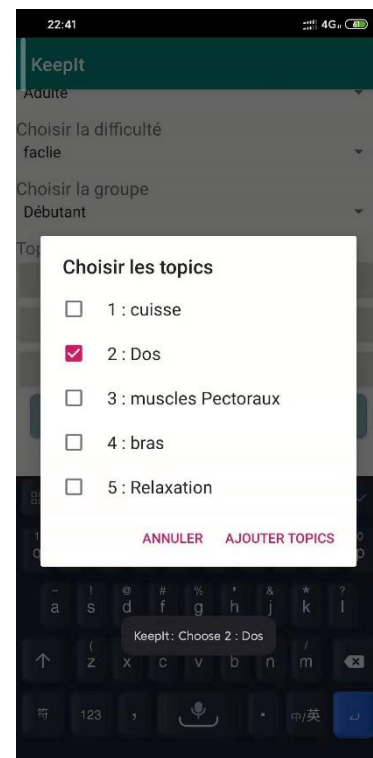
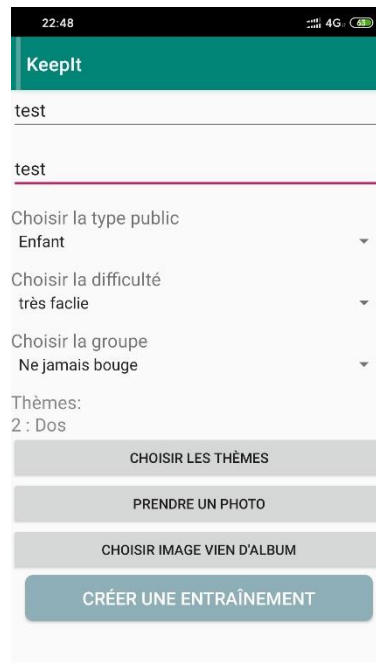
Authentication :



Thèmes



Entraînement:



Create séance :


22:49 4G 63

Créer une séance

seance 1


2

3




Courir
Type:Enfant
Level:Ne jamais bouge
Difficulté:très facile

Duration:2M



Courir
Type:Enfant
Level:Ne jamais bouge
Difficulté:très facile

Duration:2M



pour bras
Type:Adulte
Level:Amateur
Difficulté:normal

Duration:1M

+ ENTRAÎNEMENT

VALIDER

ANNULER

Thème Séance Profile

22:42 4G 61

Créer une séance

test1

2

2

Choisir Entraînement et enter la duration

3: pour bras Adulte Amate..

00 H 03 M 00 S

01 04 01

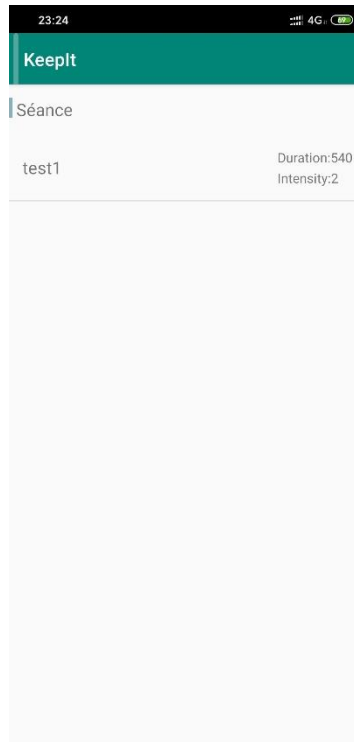
CANCEL CONFIRM

Thème Séance Profile

Profile :



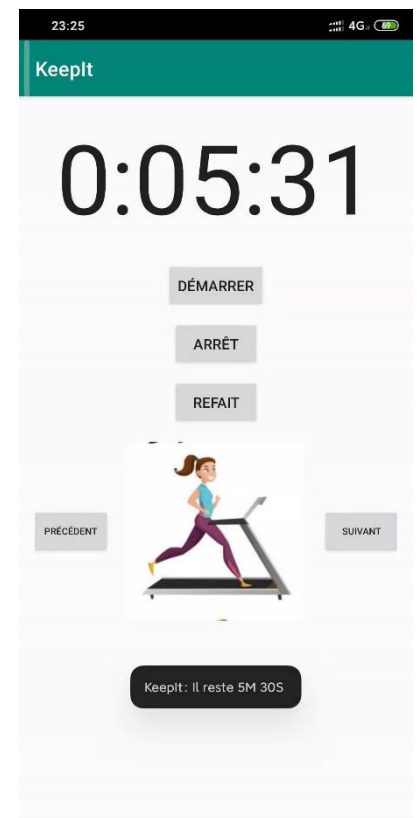
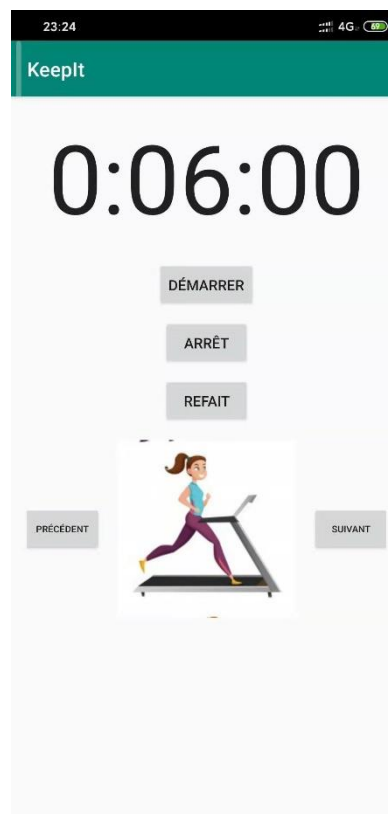
Mes séances:



Les entraînements dans un séance



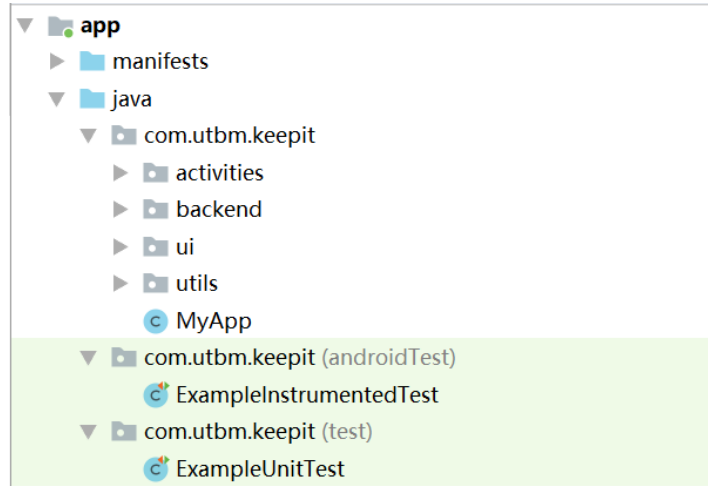
Page pour lancer séance



5 Stratégie d'implémentation

Dans un premier temps, on a eu l'occasion de mettre en place l'architecture du code exécutif. Celui-ci a été pensé en tant qu'objet et se découpe en plusieurs parties :

- activities : qui correspondent aux tous les activités
- backend : qui gère tous fonctions de recherche pour la base de donnée
- ui: qui gère les interfaces
- utils : qui gère les fonctions des outils
- .



6 Problème rencontrée

Timer :

On utilise un thread pour réaliser la fonction de chronométrage.

On rencontre des problèmes de itération d'une liste de séances pour un thread.

On ne peut pas utiliser la boucle for ou while dans le thread.

On définit une variable privée (private int i) pour réaliser la boucle d'itération d'une séance.

```
final Handler handler = new Handler();

handler.post(() -> {
    Exercise exe = listExerciseData.get(i).e;
    String uri = listExerciseData.get(i).e.getImageResource();
    System.out.println(uri);
    int hours = seconds / 3600;
    int minutes = (seconds % 3600) / 60;
    int secs = seconds % 60;
    String time = String.format("%d:%02d:%02d", hours, minutes, secs);
    timeView.setText(time);
    if (running) {
        seconds--;
    }
}, 1000);

if (running) {
    seconds--;
    if (seconds > 0) {
        //TODO : 每30s 每 15s
        if (seconds % 30 == 0) {
            int h, m, s;
            h = seconds / 3600;
            m = seconds % 3600 / 60;
            s = seconds - 3600 * h - 60 * m;
            String th = "", tm = "", ts = "";
            if (h != 0) {th = h + "H ";}
            if (m != 0) {tm = m + "M ";}
            if (s != 0) {ts = s + "S ";}
            String timeLeft = "Il reste " + th + tm + ts;
            Toast.makeText(context, StopwatchActivity.this, timeLeft, Toast.LENGTH_SHORT).show();
            Vibrator vib = (Vibrator) getSystemService(VIBRATOR_SERVICE);
            vib.vibrate(milliseconds: 1000);
        }
    }
} else { // Seance 结束的时候震动
```

```

else { // Seance 结束的时候震动
    if(i==listExerciseData.size()-1){
        Vibrator vib = (Vibrator) getSystemService(VIBRATOR_SERVICE);
        vib.vibrate( milliseconds: 2000);
        Toast.makeText( context: StopwatchActivity.this, "Félicitation", Toast.LENGTH_SHORT).show();
        running = false;
    }
    else if (i < listExerciseData.size()) {
        Vibrator vib = (Vibrator) getSystemService(VIBRATOR_SERVICE);
        vib.vibrate( milliseconds: 1000);
        i++;
        seconds = listExerciseData.get(i).jse.getDuration();
        changeImageView();
    }
}
}

```

Base de donnée :

Problème pour initialisation de bdd. Il faut reconstruire la DaoMaster de GreenDao. Nous n'avons pas beaucoup temps pour étudier et modifier la classe DaoMaster.

```

package com.utbm.KeepIt.backend.dao;

import ...

// THIS CODE IS GENERATED BY greenDAO, DO NOT EDIT.
/**
 * Master of DAO (schema version 1): knows all DAOs.
 */
public class DaoMaster extends AbstractDaoMaster {
    public static final int SCHEMA_VERSION = 1;

    /** Creates underlying database table using DAOs. */
    public static void createAllTables(Database db, boolean ifNotExists) {
        ExerciseDao.createTable(db, ifNotExists);
        JoinSeanceExerciseDao.createTable(db, ifNotExists);
        JoinTopicExerciseDao.createTable(db, ifNotExists);
        SeanceDao.createTable(db, ifNotExists);
        TopicDao.createTable(db, ifNotExists);
        UserDao.createTable(db, ifNotExists);
    }
}

```

7 Amélioration possible

Base de donnée :

On peut créer une base donnée externe pour stocker les informations des utilisateurs. Et puis on doit stocker les informations d'entraînement directement dans le smartphone.

Interfaces pour utilisateurs :

Maintenant, nos interfaces pour clients sont un peu complexes pour lancer une séance des d'entraînements. On peut simplifier les opérations. En même temps, ces interfaces ne sont pas assez belles. Presque tous les layouts sont LinearLayout. On peut embellir convenablement notre interface

Calendrier :

On n'a pas réaliser la fonction dans sujet : '*planifier ses séances dans la journée / semaine*'.

Modification et supprime pour tous les objets :

On n'a pas pas réaliser cette fonction de modification comme requis non plus. C'est une fonction importante et fondamentale pour les clients s'ils veulent modifier quelques choses.

Timer :

On utilise des threads pour implémenter la fonction timer, donc à chaque démarrage d'une nouvelle entraînement. On n'a pas pas défini de délai d'expiration ou d'autres conditions pour terminer le thread. On doit concevoir un fil qui tue cette minuterie lorsque l'utilisateur veut quitter l'entraînement

8 Conclusion

Pour conclure sur ce projet, il faut d'abord souligner que notre implémentation est fonctionnelle. La gestion de stockage des informations est la fonction plus importantes dans ce projet. Le timer pour recorder le temps est une autre fonction importante à réaliser. On a presque tous finis ces deux fonctions. Mais on n'a pas le temps pour les fonctions de calendrier et modifications.

Pendant ce projet, on a réalisé de l'application Android. On a appris comment construire une architecture basique de Android application, réaliser la connection entre l'interface et la base de donnée, réaliser un chronographe. Sauf les technologies qu'on a appris dans les cours, le chose plus important que le projet nous apprend c'est une méthodologie de raisonnement logique pour résoudre des problématiques algorithmiques qu'on peut représenter à l'aide d'un autre langage de programmation, et surtout, la persistance de rechercher et résoudre les problèmes pour l'amélioration continue.