COMPTES RENDUS DES TRAVAUX PRATIQUE

Membres du groupe :

-DJOUKOUO SAHA

- LI Xing

-MEIMOU Sonia-Ruth

-TENG

Supervisé par: Pr. FABIEN BRISSET

INTRODUCTION GENERALE

Android est un système d'exploitation mobile fondé sur le noyau Linux et développé actuellement par Google.

Son système avait d'abord été conçu pour les smartphones et tablettes tactiles, puis s'est diversifié dans les objets connectés et ordinateurs comme les télévisions (Android TV), les voitures (Android Auto), les Chromebook (Chrome OS qui utilise les applications Android) et les smart Watch (Wear OS).

L'utilisation de ce système d'exploitation conduit plus souvent à la réalisation des applications mobiles dont chacun des OS mobile, dispose d'un langage de programmation à lui propre. Les applications pour les terminaux Apple sont développées dans un langage principalement dédié à ces applications mobiles, le Swift4. Celles pour Windows Mobile, sont développées en C#5, langage aussi utilisé pour les programmes exécutables .exe. Le système Android utilise, quant à lui, un langage universel, le Java6, langage pouvant être utilisé pour les ordinateurs, le développement Web (JEE).

Dans cette optique nous avons été à même de suivre l'UV LO52 qui est une unité de valeur intitulé sous le nom de « ***Informatique mobile et communications courtes portées »*** ayant pour objectif d'acquérir :

- Les compétences sur l'architecture et les spécificités du système Android, ainsi que sur les technologies de développement d'application pour Android.
- Acquérir les compétences sur les communications courtes distances appropriées aux systèmes embarqués.

Elle nous a permis à cet effet tout au long de ce semestre d'acquérir les connaissances telles que :

- La mise en place d'un environnement de production de la plateforme Android

- De produire et définir son propre produit Android
- D'ajouter des composants à un produit Android
- D'utiliser et programmer des cartes Arduino utilisant le protocole Zygbee

A l'aide des notions vues en cours telles que :

- Introduction à l'OS Android
- Organisation et compilation du code source d'Android (Android Build System)
- Présentation de l'outil de debug « Android Debug Bridge »
- Chargement d'Android au démarrage (Android Bootloader)
- Spécificités du noyau linux utilisé et système de fichiers Android

Ce présent document comporte le rapport des différents travaux que nous avons effectués à travers les travaux pratiques et le projet en groupe. La présentation du rapport se fera en deux parties. La première portera sur le rapport des travaux pratiques et la seconde portera sur le rapport du projet.

PARTIE I : Rapport des travaux pratiques

I. TP 1 : mise en place de l'environnement de développement

1. Objectifs du TP

Dans ce TP, il est question dans un premier temps de :

- Définir son fichier de configuration Git pour les TPs et le projet.
- Mettre en place un environnement de développement d'applications Android.
- Développer et modifier sa première application Android
- Mettre en place un modèle de données pour le développement de l'application
- Modifier des fichiers de produit Android pour notre tablette spécifique au projet
- Designer l'interface graphique du projet
- Implémenter le code métier relatif à ce projet.

2. Réalisations effectuées

Nous avons tout d'abord installé le Git et cloner le dépôt effectué par l'enseignant. Pour se faire vous avons suivi des étapes suivantes :

- Télécharger GitHub Desktop et Git pour Windows.
- GitHub Desktop -> File -> Clone repository -> coller le lien copié depuis le github

https://github.com/gxfab/LO52_A2019.git pour le clonage

- $git config –global user.name “xxx”;
  $git config –global user.email xxx@example.com
  (Linux)
- GitHub desktop -> Branche -> Name : FGurlsDev -> Create branch based on master

- GitHub desktop -> **Current branch -> FGurlsDev** qui est la branche qui est propre à notre de groupe de projet

- Créer directement un répertoire dans le répertoire cloné s’appelle LO52_2019_FGurlsDev, modifier le fichier README.txt.
- Télécharger et installer AndroidStudio qui a été notre environnement de travail.
  - Java (jdk, jre)
  - SDK

Une fois l’installation effectuée le projet est créé dans Android Studio

- Android Studio → **Start a new Android Studio project** → Configurer les informations du projet -> *Empty Activity (MainActivity.java)*

  Rajouter un bouton dans le lay-out principal (activity_main.xml).

  Configurer l’action de bouton (MainActivity.java, setOnClickListener)

  Créer la deuxième activité (SecondActivity.java)

  Rajouter un label dans le layout secondaire (activity_second.xml)

  - Remarque :
    - Pour utiliser l’activité existée pour activer la nouvelle activité, la nouvelle activité n’est pas encore créée, donc nous devons utiliser newActivity.class mais pas newActivity.this.
    - ‘This’ est une référence d’un membre de l’objet actuel à partir d’une méthode d’instance ou d’un constructeur.

- Problème : Android Studio n'arrive pas à retrouver le divice HUAWEI
  - Raison et Solution : drive de huawei ne peut pas être utilisé par Android Studio, donc télécharger Google USB Driver dans SDK Manager – Google USB Driver.
- Pour changer la nouvelle activité, utiliser la classe Intent.

- Copier le répertoire src dans le répertoire LO52_A2019/LO52_2019_FGurlsDev/Part 1

Compte tenu de la suppression du TP 2, nous allons passer directement au TP3.

II. TP 3 : Création d'un device Android

1. Les différentes étapes de créations d'un device

a. Premièrement, nous allons Merger la branche SnakeTeacher dans la branche FGurlsDev pour récupérer le répertoire Rsc-libusb

1. *GitHub Desktop -> Current Branch -> **FGurlsDev***
2. *Branch -> Merge into current branch -> Merge **SnakeTeacher** into **FGurlsDev***

b. Ensuite, nous allons implémenter la libusb

1. *Fichiers sources : core.c, descriptor.c, io.c, sync.c, os/darwin_usb.c, os/linux_usbfs.c*
   Fichiers headers: libusb.h, libusbi.h, os/darwin_usb.h, os/linux_usbfs.h
2. Modifier le fichier Rsc-libusb/libusb-1.0.3/libusb/Android.mk (MakeFile pour le composant libusb) :

   LOCAL_PATH:=$(call my-dir)

   include $(CLEAR_VARS)

   LOCAL_SRC_FILES:= \

   core.c \

descriptor.c \

io.c \

sync.c \

os/darwin_usb.c \

os/linux_usbfs.c

LOCAL_C_INCLUDES += \

$(LOCAL_PATH) \

$(LOCAL_PATH)/os

LOCAL_MODULE:=libusb

include $(BUILD_SHARED_LIBRARY)

3. Solution pour l'erreur TIMESPEC_TO_TIMEVAL - Rajouter les codes suivant dans le fichier io.c :

*#define TIMESPEC_TO_TIMEVAL(tv, ts) do { *

*(tv)->tv_sec = (ts)->tv_sec; *

*(tv)->tv_usec = (ts)->tv_nsec / 1000; *

} while (0)

4. Solution pour l'erreur library "libusb.so" not in prelink map – Rajouter la ligne suivante dans le fichier Android.mk : LOCAL_PRELINK_MODULE:=false

c. Implémentation d'un nouveau produit Android

1. Renommer le fichier **Rsc-libusb/device/TP3_A_VOUS_JOUEZ.mk** comme **lo52_FGurlsDev.mk**
2. Modifier ce fichier :
   - Configurer les informations simples de produit :

     PRODUCT_NAME:=lo52_FGurlsDev

     PRODUCT_DEVICE:=lo52_FGurlsDev

     PRODUCT_BRAND:=lo52_FGurlsDev

     PRODUCT_MODEL:=lo52_FGurlsDev

- Il héritera du produit hikey de Linaro :

  $(call inherite-product,device/Linaro/hikey/hikey.mk)

- Personnaliser les propriétés :

  *PRODUCT_PROPERTY_OVERRIDES+= *

  *ro.hw=lo52 *

  *net.dns1=8.8.8.8 *

  net.dns2=4.4.4.4

- Surcharger le fichier sym_keyboard_delete.png

  1. Chercher “aosp sym_keyboard_delete.png” sur Google;
  2. Cliquer sur premier lien, le fichier peut être trouvé dans le projet platform_frameworks_base;
  3. Créer un répertoire **overlay** dans le répertoire **Rsc-libusb/device**;
  4. Créer les répertoire avec la même arborescence mirror comme ce que l’on a trouvé dans le projet :

     overlay/platform_frameworks_base/core/res/res/drawable-en-mdpi
  5. Rajouter cette ligne de code dans le fichier lo52_FGurlsDev.mk :

     DEVICE_PACKAGE_OVERLAYS:=overlay

- Ajouter la libusb aux packages du produit : PRODUCT_PACKAGES+=libusb

d. Commit les modifications sur la branche FgurlsDev :

III. TP 4 : Utilisation de JNI Configuration de l’environnement NDK

Sur l’application Android studio on effectue les actions suivantes :

Tools -> SDK Manager -> SDK Tools -> télécharger NDK, CMAKE, LLDB

File -> SDK Location -> Android NDK location : (default)

Copier le chemin par défaut, créer une nouvelle variable d’environnement

« NDK_ROOT », coller le chemin dans la champe « value », rajouter cette %NDK_ROOT% dans Path.

Cmd -> ndk-build pour tester s'il est bien configuré.

1. Création du projet

Créer un projet de type Native C++, choisir le langage Java et « Toolchain Default » comme la version de C++ Standard que nous voulons utiliser pour écrire les code sources de librairie JNI.

Remarques :

Il y a quelques fichiers et répertoires importants sont créés.

src/main/cpp : un répertoire qui contient

- CMakeLists.txt : un fichier .txt généré lors de la création du projet, il permet d'indiquer et d'ajouter les librairies JNIs dans le projet.
- native-lib.cpp : Le fichier .cpp qui contient les codes sources de librairie "native-lib" en langage C/C++.

/app/build.gradle : fichier de configuration qui gère la construction de projet, les lignes de codes suivantes permettent de lancer l'installation de librairie JNI :

```
externalNativeBuild {
        cmake {
            path "src/main/cpp/CMakeLists.txt"
            version "3.10.2"
        }
    }
```

Créer une nouvelle classe NDKTools.java qui contien la déclaration de la librairie personnalisé et déplace les lignes de code de MainActivity.java suivants dans NDKTools.java :

static {

System.loadLibrary("native-lib");

}

public native static String stringFromJNI();

Qui décrit :

- Le nom de librairie : native-lib, ce qui doit être la même avec ce qui est marqué dans le CMakeLists.txt :

```
add_library( # Sets the name of the library.
             native-lib

             # Sets the library as a shared library.
             SHARED

             # Provides a relative path to your source file(s).
           native-lib.cpp)
```

- Et les noms de fonction dans la librairie.

Une Erreur apparaît: **"Cannot resolve corresponding JNI function Java_fr_utbm_fgurlsdev_tp4jni_NDKTools_stringFromJNI"**

Raison : le nom de fonction JNI n'est pas correct ou il n'existe pas dans le code source.

Solution : Aller dans le fichier native-lib.cpp, modifier le nom de fonction comme le suivant : **Java_fr_utbm_fgurlsdev_tp4jni_NDKTools_stringFromJNI**, et modifier le type de son deuxième paramètre comme jclass.

Rajouter les fonctions suivantes :

Public native static String sayHi(); // Celle-ci est pour tester

public native static String readBtn(String numStr);

public native static String writeBtn(String numStr);

public native static String directionBtn(String direction);

Définir les fonctions dans le fichier native-lib.cpp avec les noms suivants :

Java_fr_utbm_fgurlsdev_tp4jni_NDKTools_sayHi

Java_fr_utbm_fgurlsdev_tp4jni_NDKTools_readBtn

Java_fr_utbm_fgurlsdev_tp4jni_NDKTools_writeBtn

Java_fr_utbm_fgurlsdev_tp4jni_NDKTools_directionBtn

//Définit sous le format : Java_package_Classe_nomDeFonction

(Le détail est dans les fichiers sources)

Mettre en place l'interface d'application et les fonctionnalités bouton.

Comme le principe de fonction des boutons "UP", "DOWN", "LEFT", "RIGHT" est le même, on a décidé de créer un nouveau objet **DrectionBtnListener** qui hérite l'interface **OnClickListener**, pour gérer l'action après on clique sur un de ces 4 boutons.

PARTIE II: Rapport du projet

I. Présentation du projet

1. Contexte du projet

Le projet qui nous a été confié dans le cadre de l'unité de valeur LO52, est **F1 levier.** Il s'agit d'une application androïde développée pour le CODEP25 dans le cadre de l'organisation de ses stages de perfectionnement annuels. Lors de ces manifestations, afin de travailler sur un exercice à la fois physique et mental, les entraîneurs utilisent ce qu'on appelle le F1 sorte de course de vitesse/d'obstacles. Lors de cette course, en fonction du nombre de participants répartis en deux groupes de niveaux, des équipes de 2,3 personnes équilibrées sont formées. Pour une bonne réflexion et étude du projet, le responsable de l'UV a mis à notre disposition des informations importantes qui nous a permis d'identifier les besoins du client. De plus les fonctionnalités de base de l'application ont été explicitement citées.

2. Objectifs spécifiques

L'utilisateur de notre système nous avons

- La gestion des coureurs où nous pouvons
  - Créer un coureur
  - Modifier un coureur
  - Supprimer un coureur
- La gestion des équipes où le but est de
  - Créer une équipe
  - modifier une équipe
  - Supprimer une équipe
- Créer course ou compétition
- Visualiser les statistiques selon le filtre appliqué
- Lancer et chronométrer une course

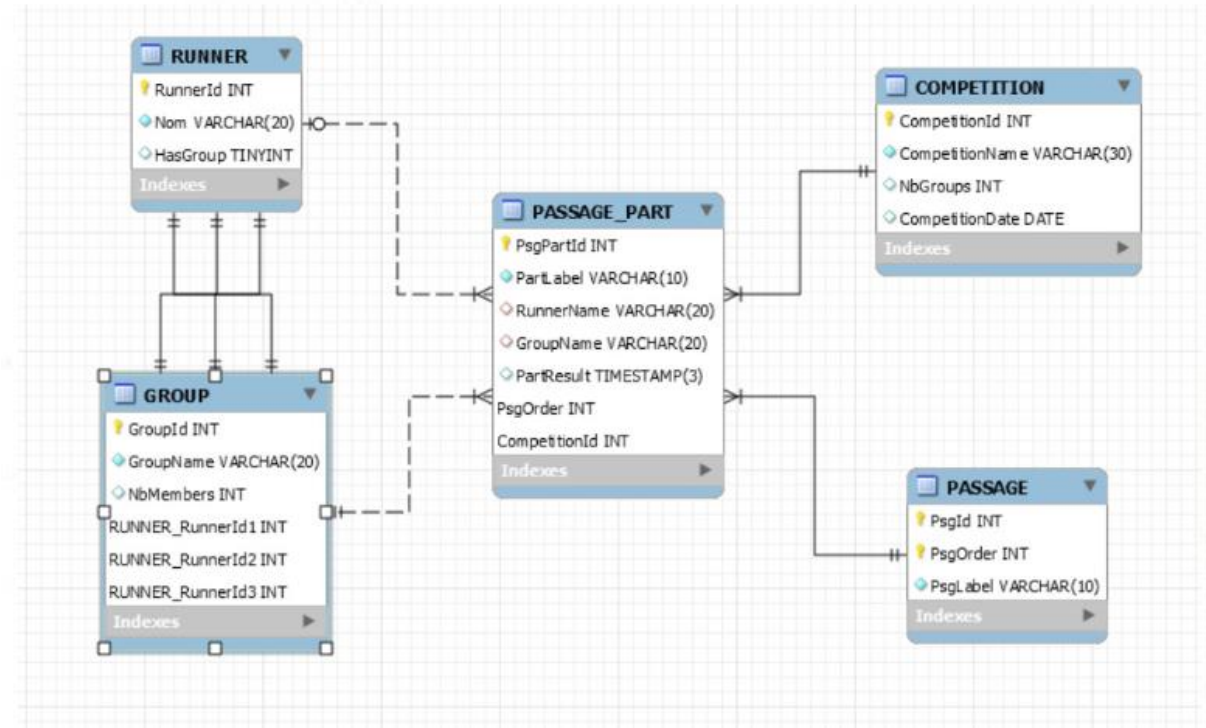
II. CONCEPTION

Pendant cette phase, la collecte des informations ainsi que leur analyse nous a permis d'établir le diagramme de classe suivante

1. Diagramme de classe

Le diagramme de classes montre la structure interne du projet et permet également de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation.

L'outil qui nous a permis de le réaliser est nommé ***mysql Workbench***



III. Environnement développement et implémentation

1. Technologie utilisé et fonctionnement

Avant de présenter les autres outils, nous allons d'abord parler des techniques qui nous ont été d'une très grande utilité dans l'implémentation de l'application. Il s'agit de

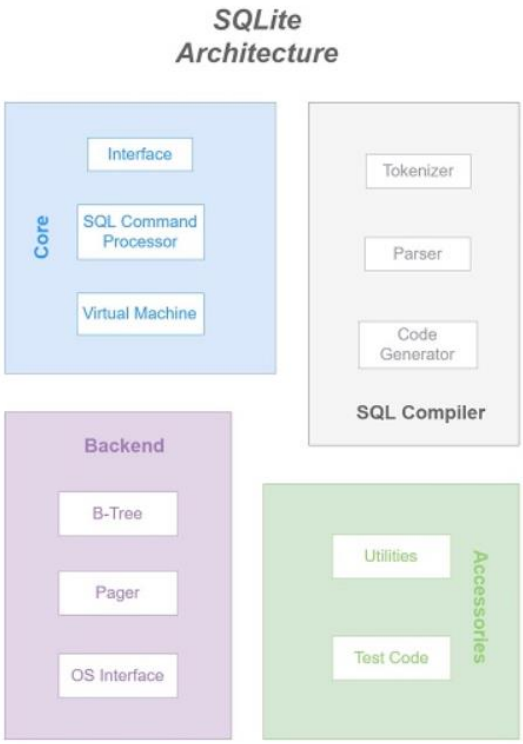
- Fragment + TabLayout + ViewPager (TabFragmentAdapter.java) : pour la représentation des vues sous forme de fragment, avec la possibilité de quitter d'une vue à l'autre soit par le clic soit par les scroll.
- AutoCompleteText (MyAutoCompleteTextAdapter.java) : nous l'avons utilisé pour afficher les noms de coureurs et des équipes lorsque la première lettre est saisie.
- ExpandableListView (MyExpandableListAdapter.java)

a. Base de données SQLite

La base de données sqlite est une base de données relationnelle open source, qui se distingue des autres bases par son indépendante à un système de gestion base de données relationnel. Cette dernière est couramment utilisée dans le contexte des applications mobiles sous Windows phone, IOS et Android. En effet, elle est facilement intégrée dans le code d'un programme ; De plus, elle est déjà implémentée sur la plateforme Android à travers sa SDK et on la retrouve dans le package **android.database.sqlite**

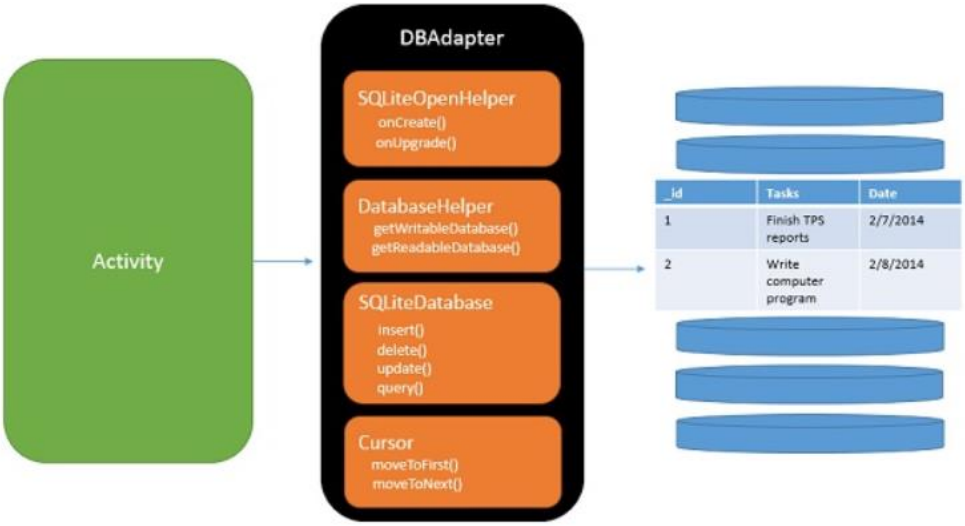
i. Architecture de Sqlite

Sqlite est composé de modules représentant chacun une fonctionnalité spécifique. On sein de ses modules on y retrouve des couches. L'architecture se présente comme suit :



Dans cette architecture nous allons le plus nous intéresser du **SQL Compiler** car il s'agit du compilateur constitué d'un séquenceur, un parseur et un générateur de code.

Le package android.database.sqlite se présente comme suit :



Tout d'abord, nous allons commencer par le **SQLiteOpenHelper** qui est l'élément

de base qui permet de créer une instance de base de données. Cette classe Helper permet aussi de créer et de gérer une base de données SQLite.

Ensuite, le **SQLiteDatabase** qui constitue les méthodes de manipulation, le **Cursor** qui récupère le résultat d'une requête a un objet **SQLiteDatabase** et surtout le **SQLiteQuery** qui représente la requête en elle-même.

ii. Implémentation de SQLite dans un projet Android

A partir d'un **Empty Activity** on crée une classe Helper qui va hériter de **SQLiteOpenHelper**. La nouvelle classe on la nomme **SQLiteDataBaseHelper** puis on commence les modifications en fonction de nos besoins. L'exemple s'illustre à partir de cette image.

```
public class SQLiteDataBaseHelper extends SQLiteOpenHelper {
    public static  final String DATABASE_NAME = "Book.db";
    public static  final String TABLE_NAME = "book_table";
    public static  final String COL_1 = "ID";
    public static  final String COL_2 = "NOM";
    public static  final String COL_3 = "AUTEUR";
    public static  final String COL_4 = "CATEGORY";

    public SQLiteDataBaseHelper(Context context) {
        super(context, DATABASE_NAME, null, 1);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {

    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }
}
```

Une fois la modification des différentes méthodes, on ouvre le fichier d'activité principale, "**MainActivity.java**", et on va faire en sorte qu'à la création de cette activité notre base de données SQLite se crée à l'aide de la classe SQLiteDataBaseHelper.

```
public class MainActivity extends AppCompatActivity {
    SQLiteDataBaseHelper db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        db = new SQLiteDataBaseHelper(this);
    }

}
```

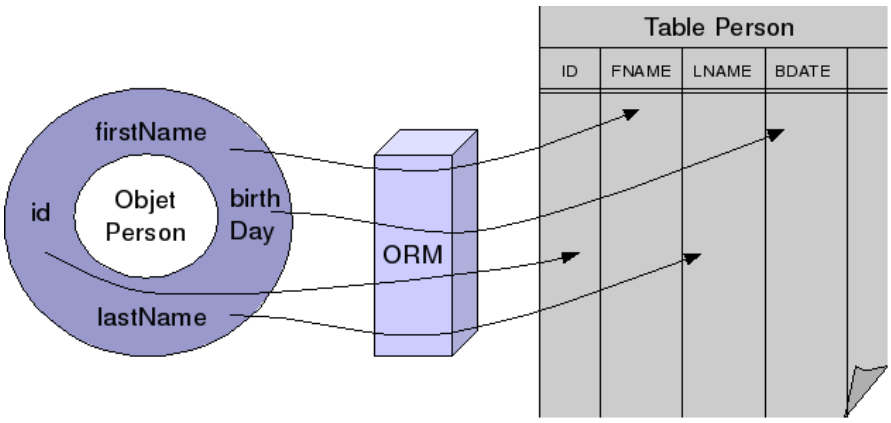
Puis on exécute l'application en cliquant **Run.**

b. Green Dao

i. Qu'est-ce que c'est Green Dao ?

Green Dao est une bibliothèque de mappage objet-relationnel (ORM) sous Android

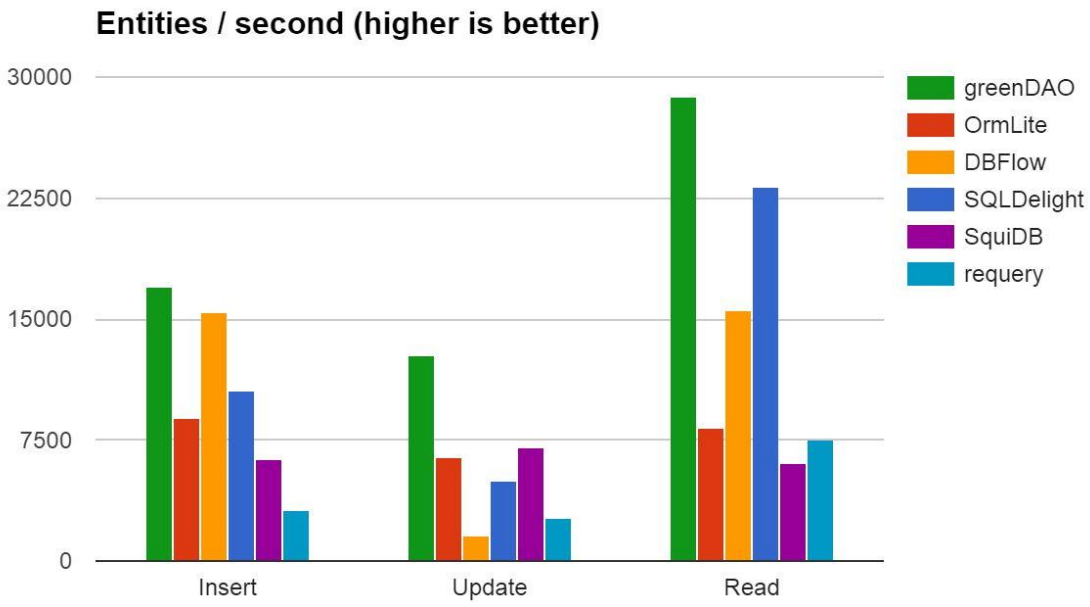
- **Qu'est-ce que c'est ORM ?**



Il permet de **f**aire la correspondance entre la table dans la base de données (Android : SQLite) et l'objet java **f**aciliter la mise en œuvre de base de données

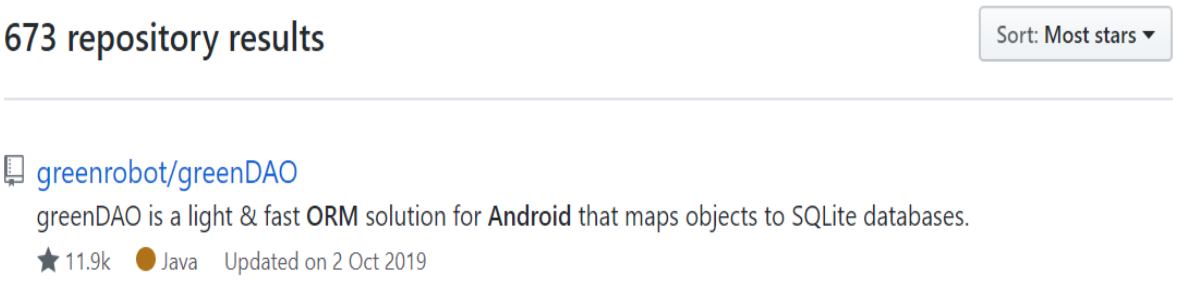
Minimiser les requêtes SQL similaire

❖ **Pourquoi Green Dao ?**



Les raisons de son utilisation sont multiples parmi lesquelles :

- Sa performance
- Dao (Data Access Object) sera généré automatiquement lorsque les modèles de donnés sont créé (voir l'explication dans « mis-en-œuvre Green Dao)



- Le plus approprié dans le cadre de Framework ORM Android sur GitHub, beaucoup de tutoriel

❖ **Mettre en œuvre Green Dao**

Sa configuration est la suivante

➢ Build.gradle sous projet/app

```
apply plugin: 'com.android.application'
apply plugin: 'org.greenrobot.greendao' // Apply plugin in application

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.2"
    defaultConfig {...}
    buildTypes {...}
}

dependencies {
    implementation 'org.greenrobot:greendao:3.2.2'  // Add library
    implementation fileTree(dir: 'libs', include: ['*.jar'])
```

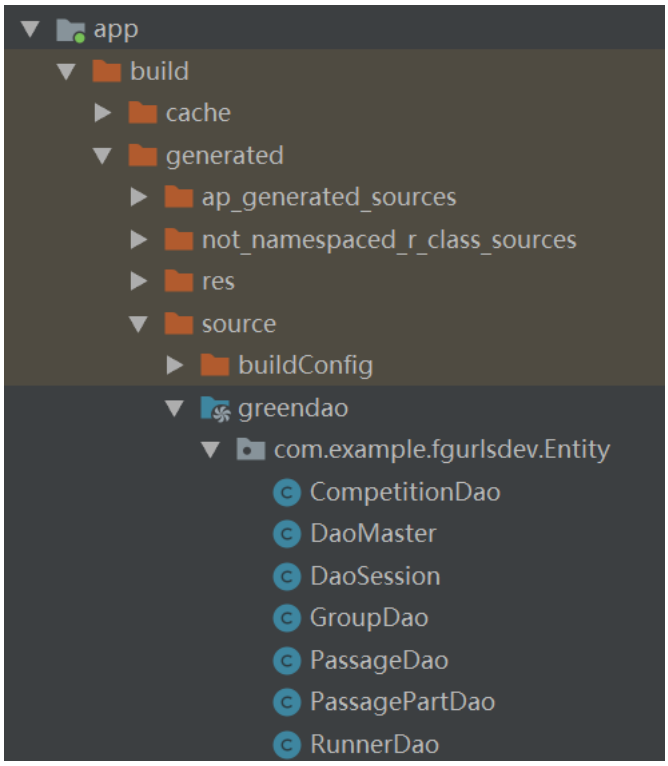
➢ Build.gradle sous projet/gradle

```
buildscript {
    repositories {
        google()
        jcenter()
        mavenCentral()  // Add repository
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.5.0'
        classpath 'org.greenrobot:greendao-gradle-plugin:3.2.2' // Add plugin
    }
}
```

➢ Création des modèles avec annotation

```java
@Entity
public class Runner {
    @Id(autoincrement = true)
    private Long runnerId;
    @Property(nameInDb = "runnerName")
    @Unique
    private String nom;
    @Property(nameInDb = "hasGroup")
    private boolean hasGroup;
```

Ensuite, b*uild/make projet, runnerDao, DaoSession et DaoMaster sont générés automatiquement sous app/generated/source/greendao*



- Récupérer runnerDao

Private DaoUtilsCollection() {

DaoManager daoManager = DaoManager.getInstance();

RunnerDao runnerDao = daoManager.getDaoSession().getRunnerDao();

RunnerDaoUtils = new CommonDaoUtils<>(Runner.class, runnerDao);

…..}

DaoUtilsCollection daoUtilsCollection = DaoUtilsCollection.getInstance();

RunnerDaoUtils = daoUtilsCollection.getRunnerDaoUtils();

- L'exemple avec ajouter un nouveau coureur

currentRunner = new Runner(null, msg, false);

runnerDaoUtils.insertEntity(currentRunner);

Les requêtes SQL native sont aussi supportées sous GreenDao

```
// Use native sql for counting
public Long getSumByNativeSql(String column, String tableName, String conditionLine) {
    String sqlQuery = "SELECT SUM(" + column + ") FROM " + tableName + " WHERE " + conditionLine;
    Cursor cursor = null;
    long count = 0;
    try {
        cursor = daoSession.getDatabase().rawQuery(sqlQuery, selectionArgs: null);
        if (cursor == null || !cursor.moveToFirst()) { return ((long)-1); }
        count = cursor.getLong( columnIndex: 0);
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (cursor != null) cursor.close();
    }
    return count;
}
```

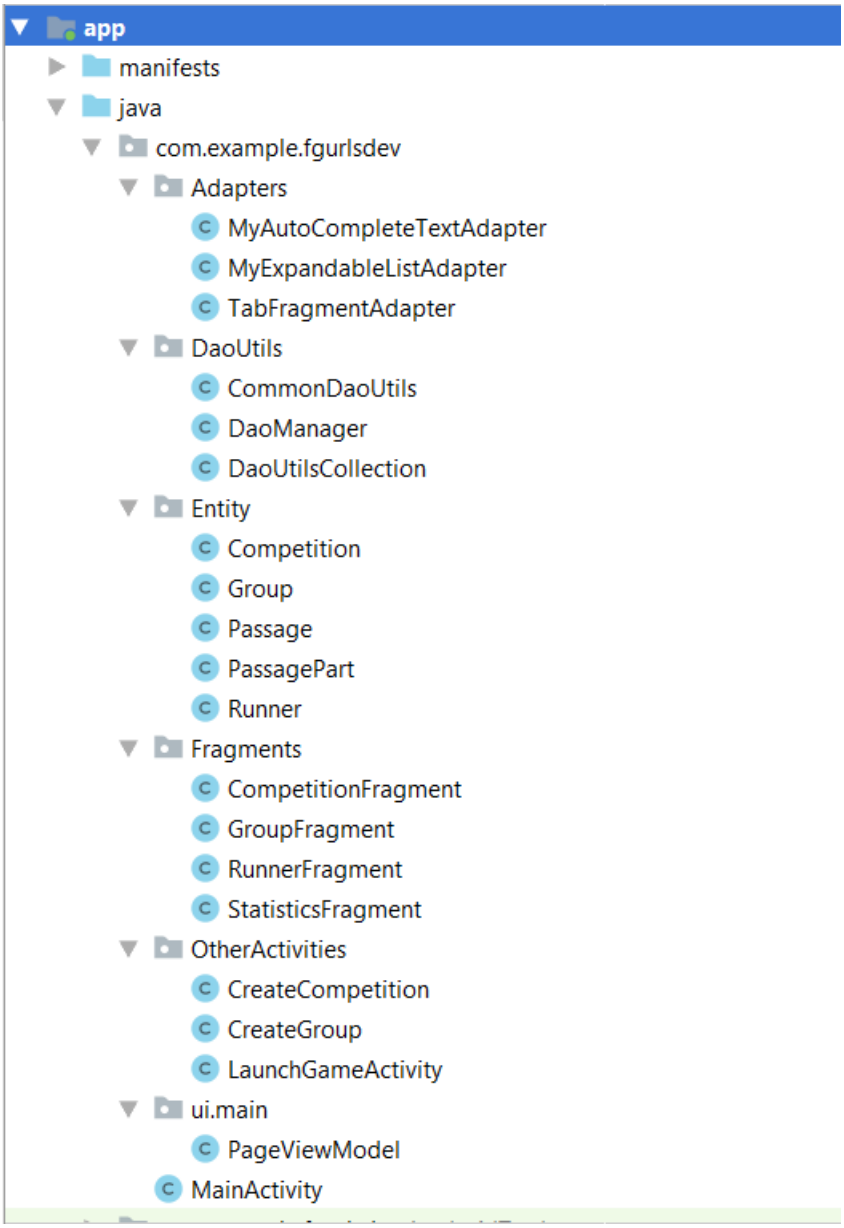
c. Outils de modélisation : MySQL Workbench

MySQL Workbench

MySQL Workbench est un outil visuel unifié pour les architectes de bases de données, les développeurs et les administrateurs de base de données. MySQL Workbench fournit la modélisation des données, le développement SQL et des outils d'administration complets pour la configuration du serveur, l'administration des utilisateurs, la sauvegarde et bien plus encore. MySQL Workbench est disponible sur Windows, Linux et Mac OS X.

IV. REALISATIONS

1. Arborescence du projet



Cette image représente l'arborescence de notre application

- Le premier répertoire **Adapters** contient les différents adaptateurs que nous avons inclut dans notre application pour des fins utiles
- Le répertoire **DaoUtils** est celui qui a été créé avec GreenDao. Dans ce répertoire, on retrouve les différentes méthodes qui indexent la base de données.

- Le répertoire **Entity** représente les classes modèles de notre application et les objets correspondent aux différentes tables de la base de données
- Le répertoire **Fragments** : elle représente les vues et chaque fragment correspond à un onglet.
- Le répertoire **OtherActivities** : c'est toutes les autres activités que nous avons (CreateCompetition) en dehors du mainActivity

2. Réalisations

Dans cette partie, il est question de montrer les détails de la réalisation des interfaces et leurs différentes fonctionnalités.

Cette application consiste, dans le cadre de l'organisation de stage annuels, à enregistrer les personnes désireuses participer à ces différents stages et de sauvegarder les prestations personnelles et en groupes effectuées. L'application ainsi développée se présente comme suit :

❖ La page d'accueil

A l'ouverture de l'application, l'utilisateur accède à la page d'accueil qui présente la liste des courses en présentant leur nom, la date de commencement de la course et leur état enregistré dans la base de données. Cette activité contient plusieurs fragments qui redirigent vers d'autres activités.

1- Redirige vers l'activité de saisie de course

2- Redirige vers la page Accueil (Page actuelle)

3- Redirige vers la page de Gestion de Participants

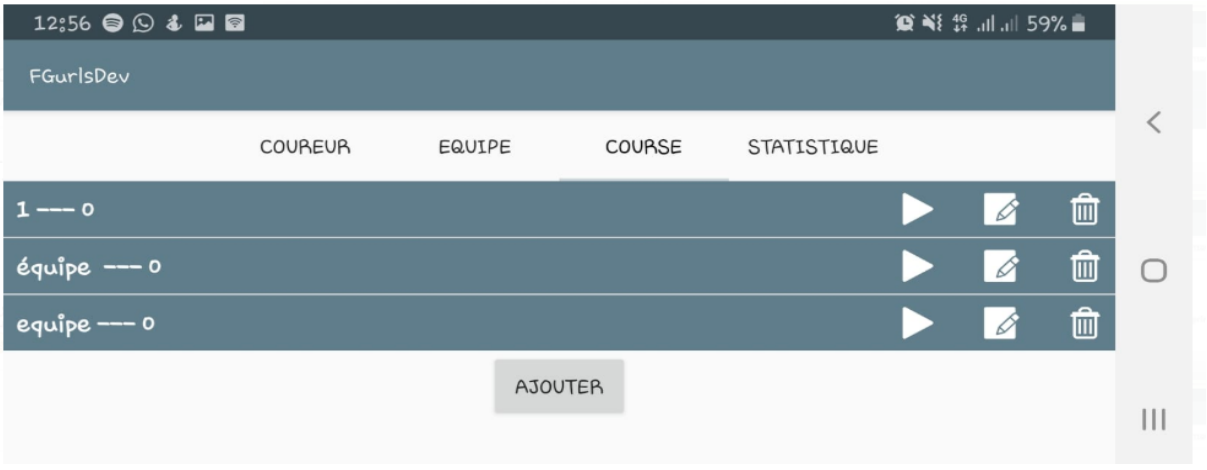
4- Redirige vers la gestion d'équipes

La page d'accueil se présente comme suit :

Page d'accueil

❖ Ajout d'une course

Sur cette activité nous ajoutons les courses qui sont définie dans notre application par leur nom. La date de la course est automatiquement sauvegardée à la création de la course.



Page d'ajout d'une course

❖ Liste des équipes et ajout d'équipe

Cette activité présente la liste de toutes les équipes en présentant leur nom, en donnant la possibilité d'accéder à l'activité :

1- D'ajout de participants

2- De modification de l'équipe

3- De suppression de l'équipe

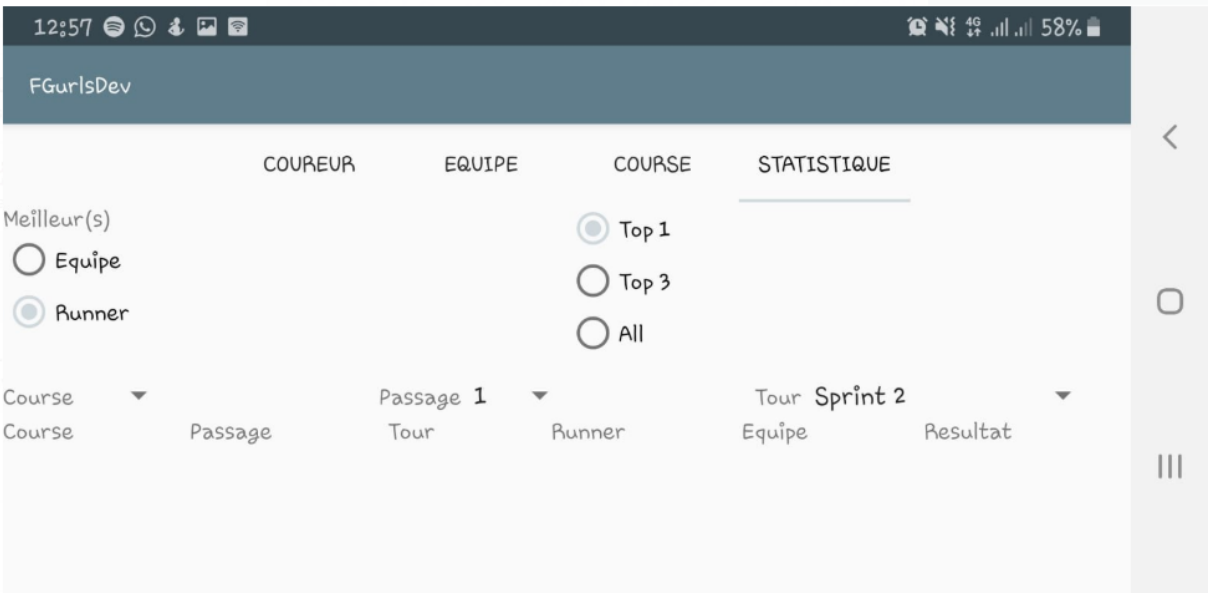
4- D'ajout d'une nouvelle équipe qui permet de renseigner le nom de l'équipe et de l'enregistrer

❖ Statistiques

Nous avons prévu aussi pour la fin de notre projet faire un suivi de tout ce qui a été effectué, via des statistiques, en questionnant la base de données par rapport aux critères précis désirés par l'utilisateur. Aussi pour un suivi des performances des participants et pour la prise des décisions telles que :

- Les joueurs les plus faibles,
- La meilleure équipe,

- L’évolution des joueurs dans le temps



3. Les éventuelles Améliorations

Ajouter un switch dans les pages de création pour choisir la forme d’ajouter un coureur dans une équipe ou pour ajouter une équipe dans une course :

- En cochant les cas
- En saisissant à la main

Utilisation de ToMany clé étranger (GreenDao).

Importer un fichier excel pour insérer les données.

Drag and drop pour changer la position du coureur.

CONCLUSION

Parvenue au terme de notre travail, nous dirons que ce projet a été très instructif pour notre groupe. Nous avons premièrement manipulé le kernel linux puis implémenté un produit et ajouter un composant Android. Nous avons compris le fonctionnement du kernel, et sa compilation. Nous avons vu au travers de cette partie qu'il est assez simple d'ajouter des composants à notre device Android à partir des librairies.

L'application JNI nous a permis de comprendre que nous pouvions utiliser un autre langage que Java pour développer une application Android. La documentation nous a pris un peu de temps, ce pendant nous avons réussi à nous adapter aux normes du langage JNI et à interagir entre les deux langages.

Pour finir, l'application F1Levier, nous a permis d'améliorer plusieurs compétences. Tout d'abord, le sujet étant un peu difficile à comprendre nous avons optimisé notre capacité d'analyse afin de répondre aux besoins du client, mais aussi de lui donner la possibilité d'avoir accès à ses informations sans avoir à faire plusieurs clics. De plus ; le travail en équipe nous avons appris à mettre en commun nos connaissances, pour permettre à chacune de nous d'être satisfaite à la fin du projet. Mieux encore, notre système de travail a permis que chacune ait une idée globale du projet et ce malgré les différents problèmes rencontrés. Ce projet dans sa globalité nous a permis de mieux comprendre le système Android, passant de la configuration de son noyau au développement des applications qui y tournent.