TP 7 – Persistance des données : SQLite

L'objectif de ce TP est de créer et manipuler une base de données interne sur le SGBD SQLite.

1.1. SQLite

— Dans ce TP on se propose de voir comment interagir avec une base de données SQLite embarquée sur le système android.
— Le SQLite est un SGBDR intégré, efficace en terme de mémoire que le moteur d'exécution d'Android peut l'inclure dans son intégralité.
— Le package « `android.database.sqlite` » contient les classes de gestion de base de données SQLite qu'une application utilise pour gérer sa propre base de données privée.

— La classe « `SQLiteDatabase` » : Contient des méthodes pour gérer une base de données : création, ouverture, fermeture, insertion, mise à jour, suppression et exécute une base de données SQLite

— La méthode « `SQLiteDatabase.openOrCreateDatabase()` » permet d'ouvrir une base de données existante ou en créer une dans la zone de données d'application :

```
import android.database.sqlite.SQLiteDatabase;
SQLiteDatabase db;
db=openOrCreateDatabase("DBName", SQLiteDatabase.CREATE_IF_NECESSARY,
null);
```

— La méthode « `execSQL()` » pour la création des tables : Créer une chaîne statique contenant l'instruction SQLite CREATE, et utiliser la méthode « `execSQL()` » pour l'exécuter.

```
String requete = CREATE TABLE IF NOT EXISTS auteur(id INTEGER PRIMARY
KEY,nom VARCHAR, prenom VARCHAR);
db.execSQL(requete);
```

— La méthode « `insert()` » pour insérer de nouvels enregistrements dans une table :

```
import android.content.ContentValues;
ContentValues values = new ContentValues();
values.put("nom", "Hugo");
values.put("prenom", "Victor");
db.insert("auteur","", values);
```

— La méthode « `update()` » pour mettre à jour un enregistrement d'une table :

```
import android.content.ContentValues;
ContentValues values = new ContentValues();
values.put("nom", "Nouveau Nom");
values.put("prenom", "Nouveau Prénom");
db.update("auteur",values,"id = 1",null);
```

— La méthode « `delete()` » pour supprimer un enregistrement d'une table :

```
db.delete("auteur","id = 1",null);
```

— La méthode « `rawquery()` » de la classe SQLiteDatabase qui exécutent des requêtes sur la BD et renvoie les résultats dans un objet Cursor :

```
Cursor c=db.rawQuery("SELECT * FROM auteur", null);
```

— La classe Cursor gère le curseur résultat d'une sélection :

— récupération des valeurs de champ par des méthodes getT(int columnIndex) où T est le type à récupérer : `getString`, `getInt...`

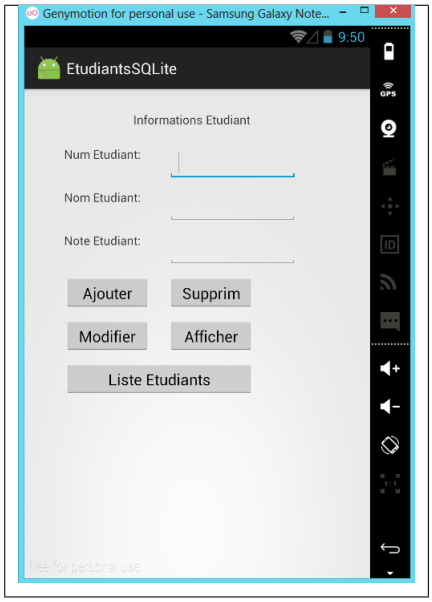
— déplacements avec les méthodes de type move(...) : `moveToFirst()`, `move(int pos)`, `moveToNext()...`

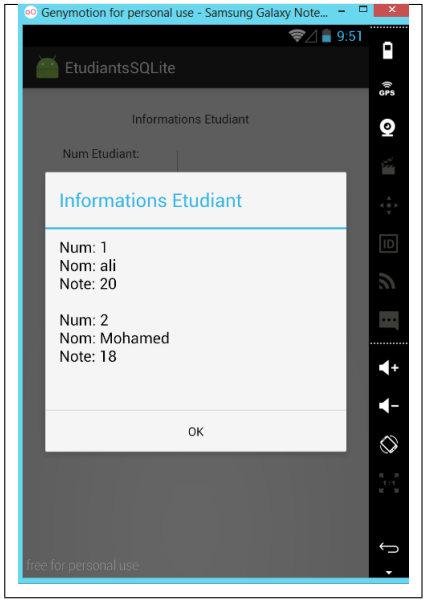
Voyons dans ce TP comment créer et interroger une Base de Données SQLite .

1.2. Travail à faire

On se propose dans cet exercice de créer une activité android permettant de sauvegarder dans une table, des informations sur les étudiants : leurs numéros, noms et notes.

Cette activité doit permettre à son utilisateur de gérer ces informations :





(1) Créer un nouveau projet Android, et réaliser l'interface graphique de cette activité.
Utiliser le fichier « strings.xml » pour y référer les noms des boutons ainsi que les labels.

(2) En suivant les indications mentionnées dans la 1ére partie du TP :

(a) Déclarer un attribut correspondant à la Base de données qu'on souhaite créer.

(b) Créer une nouvelle base de données, qu'on va nommer « `BDEtudiants` ».

(c) Créer dans cette base de données, une nouvelle table « `Etudiant(NumE, NomE, NoteE)` »

(3) Les actions des boutons :

(a) En ajoutant un nouveau étudiant à la table « `Etudiant` », le bouton « Ajouter » doit d'abord vérifier que tous les champs sont remplis sinon une AlertDialog s'affiche à l'utilisateur lui demandant de remplir tous les champs. Après l'opération d'insertion, un message s'affiche à l'utilisateur lui indiquant que l'ajout s'est effectué avec succés.

(b) Le bouton « Supprimer » permet de supprimer de la table « Etudiant » l'étudiant dont le numéro a été saisi par l'utilisateur.
Utiliser un curseur afin de récupérer tous les enregistrements dans la table ayant comme numéro celui que l'utilisateur a saisi (unique puisque c'est une clé primaire),

(i) Si ce curseur contient des éléments, on supprime l'étudiant dont le numéro a été saisi par l'utilisateur (Penser à utiliser la méthode booléenne du curseur contenant le résulat de la requête select : « `moveToFirst` »

(ii) Sinon, afficher un message d'erreur à l'utilisateur.

(c) Le bouton « Modifier » permet d'appeler la méthode « update() » afin d'appliquer less modifications apportées par un utilisateur sur un enregistrement donné.

(d) Le bouton « Afficher » remplit les zones de textes Nom et Note par les valeurs correspondantes à l'étudiant dont le huméro a été saisi par l'utilisateur.

(e) Le bouton « Liste des étudiants » utilise un curseur qui contiendra la liste de tous les étudiants enregistrés dans la Base.

(i) Si ce curseur est vide (i.e la méthode `getCount()` qui retourne le nombre d'élements dans un curseur, retourne 0), dans ce cas un message indiquant qu'Aucun enregistrement n'a été trouvé, sera affiché à l'utilisateur.

(ii) Sinon, on affiche la liste des étudiants en utilisant une boucle répétitive qui parcourt le curseur :

```
while(cursor_name.moveToNext()) {
 //Votre code ici
}
```