Hananchi Thamer

# Compte Rendu de la Vidéo : Méthodes de Sauvegarde et Utilisation d'une Base de Données Locale

---

**Contexte**

Apres avoir vu une méthode de sauvegarde à l'aide de sérialisation dans un fichier binaire. Cette méthode permet d'enregistrer et de récupérer des objets de manière légère, mais ne permet pas de travailler directement sur le fichier binaire ; il est nécessaire de reconvertir les données en objets.

---

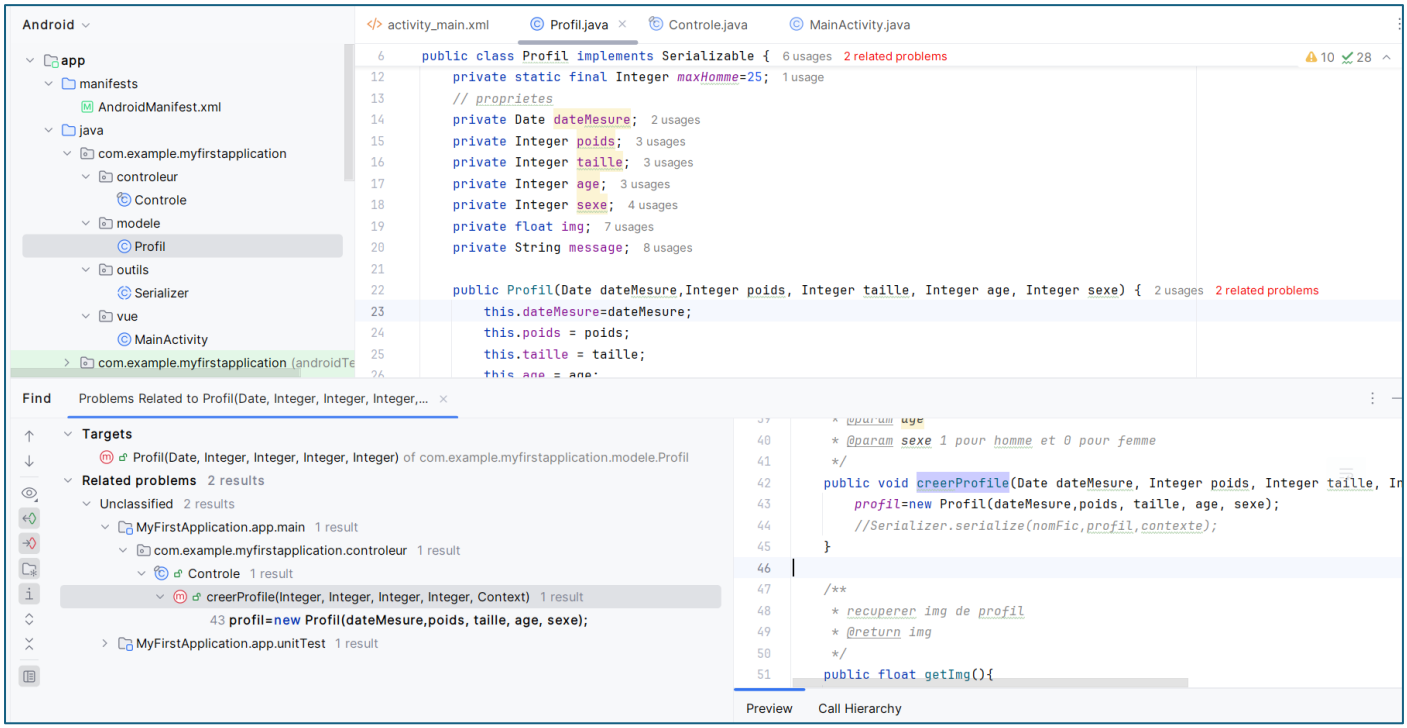
**Objectif de la Séance**

Aujourd'hui, nous explorons l'enregistrement dans une base de données SQLite, une base de données légère accessible directement sur le téléphone. Cela permet de travailler en local et d'utiliser des requêtes SQL.

---

**Étapes de Configuration**

1. **Désactivation des Méthodes de Sérialisation**
    - Nous désactivons les fonctionnalités de sérialisation dans MainActivity, notamment dans la méthode init(), en commentant les lignes associées :
        - recupProfil : récupère le profil enregistré
        - recupSerial : enregistrement sérialisé
        - serialIsr : initialise la sérialisation
    - *Capture d'écran de MainActivity après la désactivation.*
2. **Vérification de la Désactivation**
    - Nous lançons l'application pour confirmer qu'aucune donnée sérialisée précédente n'est récupérée. À l'exécution, aucun profil ne s'affiche, ce qui confirme que la désactivation a bien fonctionné.
    - *Capture d'écran montrant l'absence de données affichées.*
3. **Ajout d'une Propriété Unique : dateMesure**
    - Pour permettre l'enregistrement de plusieurs profils, nous ajoutons une propriété dateMesure de type Date, permettant une identification unique par horodatage. Cela permet de mémoriser la date et l'heure précises de chaque entrée.
    - Nous ajoutons également :
        - Le getter pour dateMesure
        - La propriété dateMesure dans le constructeur




**Mise à Jour des Créations de Profil**

- Dans le code de contrôle de création d'un profil, nous ajoutons le paramètre dateMesure pour chaque nouvel enregistrement, en utilisant l'horodatage actuel (new Date()).

## 4. Création de la Classe MySQLiteOpenHelper

- **Objectif** : Créer une classe MySQLiteOpenHelper pour gérer l'accès à la base de données en étendant la classe SQLiteOpenHelper d'Android.
- **Étapes** :
  - Nous ajoutons une nouvelle classe MySQLiteOpenHelper dans notre répertoire Outils.
  - Nous faisons hériter cette classe de SQLiteOpenHelper, ce qui nous permet d'utiliser les méthodes d'accès à la base de données prédéfinies d'Android.
  - Nous implémentons les deux méthodes abstraites onCreate() et onUpgrade() de SQLiteOpenHelper.

Hananchi Thamer

- 

```
8
9   public class MySQLiteOpenHelper extends SQLiteOpenHelper {  no usages
10      /**
11       * constructeur
12       * @param context
13       * @param name
14       * @param factory
15       * @param version
16       */
17      public MySQLiteOpenHelper(@Nullable Context context, @Nullable String name, @Nul
18          super(context, name, factory, version);
19      }
20
21      /**
22       * @param sqLiteDatabase
23       */
24      @Override
25      public void onCreate(SQLiteDatabase sqLiteDatabase) {
26
27      }
28
29      /**
30       * @param sqLiteDatabase
31       * @param i
32       * @param i1
33       */
34      @Override  no usages
35      public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
36
37      }
38  }
39
```

---

5. **Méthodes de la Classe MySQLiteOpenHelper**

- **OnCreate** :
  - Cette méthode est exécutée lors de la création de la base de données.

- On y ajoute une requête SQL pour créer la table profil avec les champs dateMesure, poids, taille, et sexe, chacun ayant un type de données précis.

```
public class MySQLiteOpenHelper extends SQLiteOpenHelper {  no usages

    //proprietés
    private String creation="create table profil ("  no usages
            +"datemesure TEXT PRIMARY KEY,"
            +"poids INTEGER NOT NULL,"
            +"taille INTEGER NOT NULL,"
            +"age INTEGER NOT NULL,"
            +"sexe INTEGER NOT NULL);";
```

- 

- *Remarque :* Le champ dateMesure est enregistré en tant que type TEXT car SQLite ne supporte pas directement le type Date.

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {
    sqLiteDatabase.execSQL(creation);

}
```

- **OnUpgrade** :
  - Cette méthode est déclenchée lors des modifications de version de la base de données.

---

6. **Classe AccesLocal pour Gérer l'Accès à la Base de Données**

- **Objectif** : Créer une classe AccesLocal qui servira d'interface pour manipuler les données dans la base.
- **Étapes** :
  - Nous créons une nouvelle classe AccesLocal.

- Dans cette classe, nous définissons le nom de la base de données (bd_coach.db), la version (1), et un objet MySQLiteOpenHelper pour gérer les connexions.

```
public class AccesLocal {  no usages

    // proprietes
    private String nomBase="bdMyFirstApp.sqlite";  1 usage
    private Integer versionBase=1;  1 usage
    private MySQLiteOpenHelper accesBD;  3 usages
    private SQLiteDatabase bd;  4 usages

    /**
     * constructeur
     * @param context
     */
    public AccesLocal(Context context) {  no usages
        accesBD=new MySQLiteOpenHelper(context,nomBase, factory: null,versionBase);
    }
```

---

7. **Méthodes de AccesLocal**

- **Ajout de Profils** :
    - La classe inclura une méthode pour ajouter des profils dans la base de données, en utilisant l'objet SQLiteDatabase.

- **Récupération du Dernier Profil** :
    - Nous créons une méthode pour récupérer le dernier profil enregistré, ce qui illustre le principe de lecture dans SQLite. Nous verrons plus tard comment afficher tous les profils.

8. **Constructeur de la Classe AccesLocal**

- **Objectif** : Créer un constructeur pour la classe AccesLocal qui permet d'instancier MySQLiteOpenHelper afin de gérer les connexions à la base de données.
- **Étapes** :

- Déclaration et instanciation de l'objet MySQLiteOpenHelper (accès_bd) dans le constructeur de AccesLocal.
- Ce constructeur nécessite un contexte, transmis en paramètre, qui est utilisé lors de l'appel au constructeur de MySQLiteOpenHelper.
- Le constructeur prend également le nom de la base de données (nom_base) et la version (version_base), avec null pour le paramètre factory.

```
/**
 * constructeur
 * @param context
 */
public AccesLocal(Context context) {  no usages
    accesBD=new MySQLiteOpenHelper(context,nomBase, factory: null,versionBase);
}
```

---

9. **Ajout d'un Profil dans la Base de Données**

- **Méthode ajout()** :
  - Cette méthode permet d'insérer un profil dans la table profil.
  - Elle reçoit en paramètre un objet Profil contenant les données à insérer (date, poids, taille, âge, sexe).
  - **Étapes** :
    - Création de la requête SQL avec INSERT INTO pour ajouter les champs dateMesure, poids, taille, âge, sexe.
    - **Remarque** : Les valeurs texte (comme dateMesure) sont entourées de guillemets grâce à un caractère d'échappement (\) pour être interprétées correctement dans la requête SQL.
    - Utilisation de bd.execSQL(requete) pour exécuter la requête, où bd est un objet SQLiteDatabase instancié avec accès_bd.getWritableDatabase() pour permettre l'écriture.

---

**10. Gestion des Types dans la Requête SQL**

- **Gestion des Types pour dateMesure** :
    - Puisque dateMesure est de type String, des guillemets sont ajoutés autour de cette valeur dans la requête SQL.
    - **Caractère d'échappement** : Utilisation de \" pour inclure les guillemets dans une chaîne:

- ```
  String req="insert into profil (datemesure,poids,taille,age,sexe) values ";
  req+="(\""+profil.getDateMesure()+"\","+profil.getPoids()+","+profil.getTaille()+
  ","+profil.getAge()+","+profil.getSexe()+")";
  ```

**Récupération du Dernier Profil de la Base de Données**

**1. Méthode récupDernier() dans la Classe AccesLocal**

- **Objectif** : Créer une méthode récupDernier() qui permet de récupérer le dernier profil inséré dans la base de données. Cette méthode retourne un objet Profil.
- **Étapes** :
    - Création d'une connexion en lecture seule à la base de données via getReadableDatabase().
    - Déclaration d'un objet Profil initialisé à null, qui contiendra le dernier profil si des données existent dans la base.

```
/**
 * ajout de profil dans database
 * @param profil
 */
public void ajout(Profil profil) {  no usages
    bd=accesBD.getWritableDatabase();
    String req="insert into profil (datemesure,poids,taille,age,sexe) values ";
    req+="(\""+profil.getDateMesure()+"\","+profil.getPoids()+","+profil.getTaille()+","+profil.getAge()+","+profil.getSexe()+")";
    bd.execSQL(req);
}
```

    - **Requête SQL** : Exécution d'une requête SELECT * FROM profil pour récupérer tous les profils.
```

Hananchi Thamer

- Utilisation d’un objet Cursor pour naviguer dans les lignes de résultat et positionner directement le curseur sur la dernière ligne avec moveToLast().
- **Vérification** : Vérification si le curseur contient des éléments avec !cursor.isAfterLast(), ce qui permet de s’assurer qu’il y a bien des données avant de continuer.

2. Récupération des Données et Création de l’Objet Profil

- **Extraction des données** :
  - La date, le poids, la taille, l’âge et le sexe sont extraits colonne par colonne à l’aide des méthodes cursor.getString() et cursor.getInt(), avec les index de colonnes correspondant.
  - **Note** : La date n’étant pas encore manipulée, une valeur par défaut est affectée (comme new Date()), en attendant une conversion correcte du format.
- **Création de l’Objet Profil** :
  - Une fois les données récupérées, un nouvel objet Profil est créé et initialisé avec les informations extraites.

**3. Fermeture et Retour de la Méthode**

- **Fermeture** :
  - Fermeture du curseur pour libérer les ressources.
- **Retour de l’Objet Profil** :
  - L'objet Profil est retourné, permettant d’accéder au dernier profil inséré dans la base de données.

```java
/**
 * recuperation de dernier profol de base de données
 * @return
 */
public Profil recupDernier(){  no usages
    bd=accesBD.getReadableDatabase();
    Profil profil=null;
    String req="select * from profil";
    Cursor curseur =bd.rawQuery(req, selectionArgs: null);
    curseur.moveToLast();
    if (!curseur.isAfterLast()){
        Date date= new Date();
        Integer poids= curseur.getInt( i: 1);
        Integer taille= curseur.getInt( i: 2);
        Integer age= curseur.getInt( i: 3);
        Integer sexe= curseur.getInt( i: 4);
        profil=new Profil(date,poids,taille,age,sexe);
    }
    curseur.close();
    return profil;
}
```

---

**Intégration dans la Classe Controle**

**1. Déclaration de l'Accès à la Base de Données**

- **Ajout d'un Accès Statique** :

Hananchi Thamer

- Dans la classe Controle, ajout d'une propriété statique AccesLocal pour faciliter l'accès à la base de données.

```
public final class Controle {  8 usages

    private static Controle instance=null;  3 usages
    private static Profil profil;  13 usages
    private static String nomFic="saveprofil";  1 usage
    private static AccesLocal accesLocal;  2 usages
```

- **Initialisation** :
  - Initialisation de AccesLocal avec une instance de la classe, en passant le context nécessaire.

**2. Récupération et Sauvegarde du Profil**

- **Récupération du Dernier Profil** :
  - Récupération du dernier profil via recupDernier() et affectation à un objet Profil.

```
/**
 * creation d el'instance
 * @return instance
 */
public static final Controle getInstance(Context contexte){
    if (Controle.instance==null) {
        Controle.instance=new Controle();
        accesLocal=new AccesLocal(contexte);
        profil=accesLocal.recupDernier();
        //recupSerialize(contexte);
    }
    return Controle.instance;
}
```

- **Enregistrement d'un Nouveau Profil** :

Hananchi Thamer

- Utilisation de la méthode ajout() pour insérer un nouveau profil dans la base.

```java
/**
 * creation de profil
 * @param poids
 * @param taille est en centimetre
 * @param age
 * @param sexe 1 pour homme et 0 pour femme
 */
public void creerProfile( Integer poids, Integer taille, Integer age, Integer sexe , Context contexte){
    profil=new Profil(new Date(),poids, taille, age, sexe);
        accesLocal.ajout(profil);
```

**Décommenter la Récupération du Profil dans l'Activité Principale**

- Dans votre activité principale (MainActivity), décommentez l'appel à la méthode récupProfil() pour que l'application puisse charger le dernier profil sauvegardé au démarrage.
- **Objectif** : Permettre à MainActivity de demander au contrôleur (Controle) de récupérer le dernier profil enregistré sans se soucier de la méthode de stockage utilisée (sérialisation ou base de données SQLite).

la puissance du **design MVC** :
dans la classe `MainActivity`, on se moque complètement de la manière dont les informations sont récupérées, car c'est le **contrôleur** qui s'en occupe.
Dans `MainActivity`, on a simplement commenté ou décommenté une ligne, car `MainActivity` ne fait que demander les informations au contrôleur.
C'est donc au contrôleur de décider s'il doit récupérer les données via la **sérialisation**, une base de données **SQLite** locale, ou même une base de données **distante**. Le **MainActivity** n'a pas à se soucier de cette logique.

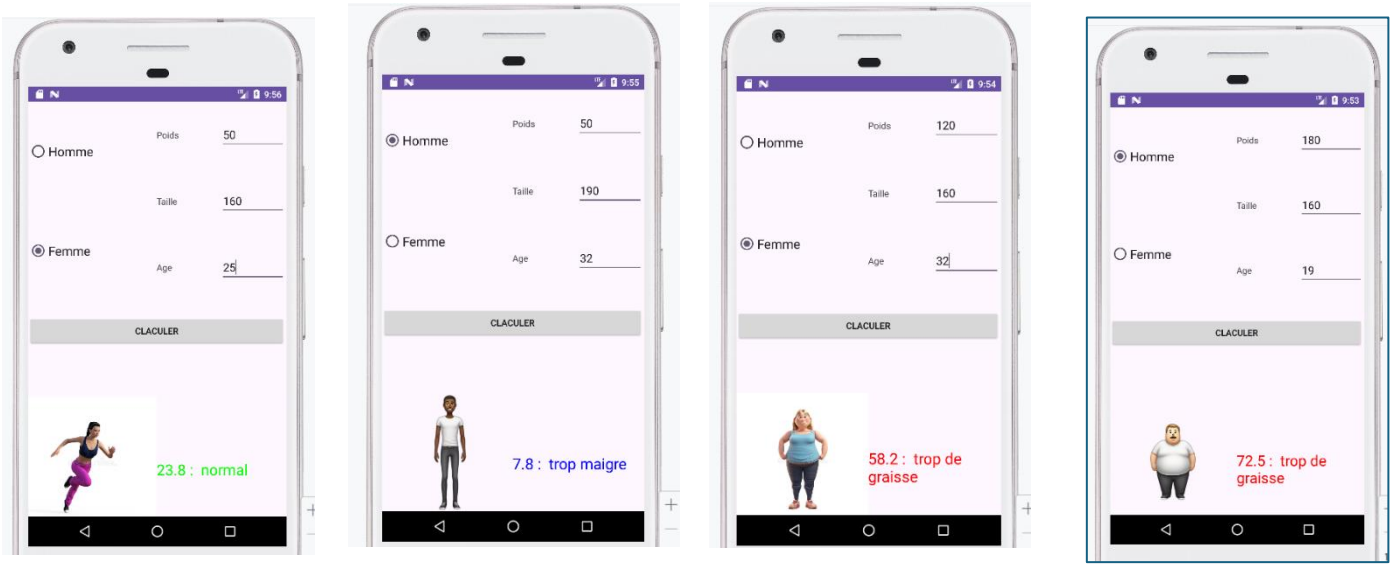
```java
private void init(){  1 usage
    PoidsText = (EditText) findViewById(R.id.PoidsText) ;
    TailleText = (EditText) findViewById(R.id.TailleText);
    AgeText = (EditText) findViewById(R.id.AgeText) ;
    radioHomme = (RadioButton) findViewById(R.id.radioHomme) ;
    bodyShape = (ImageView) findViewById(R.id.bodyShape) ;
    observation = (TextView) findViewById(R.id.observation) ;
    this.control= Controle.getInstance( contexte: this);
    ecouteCalcul();
    recupprofil();
}
```

Hananchi Thamer

**2. Exécuter l'Application et Tester les Données**

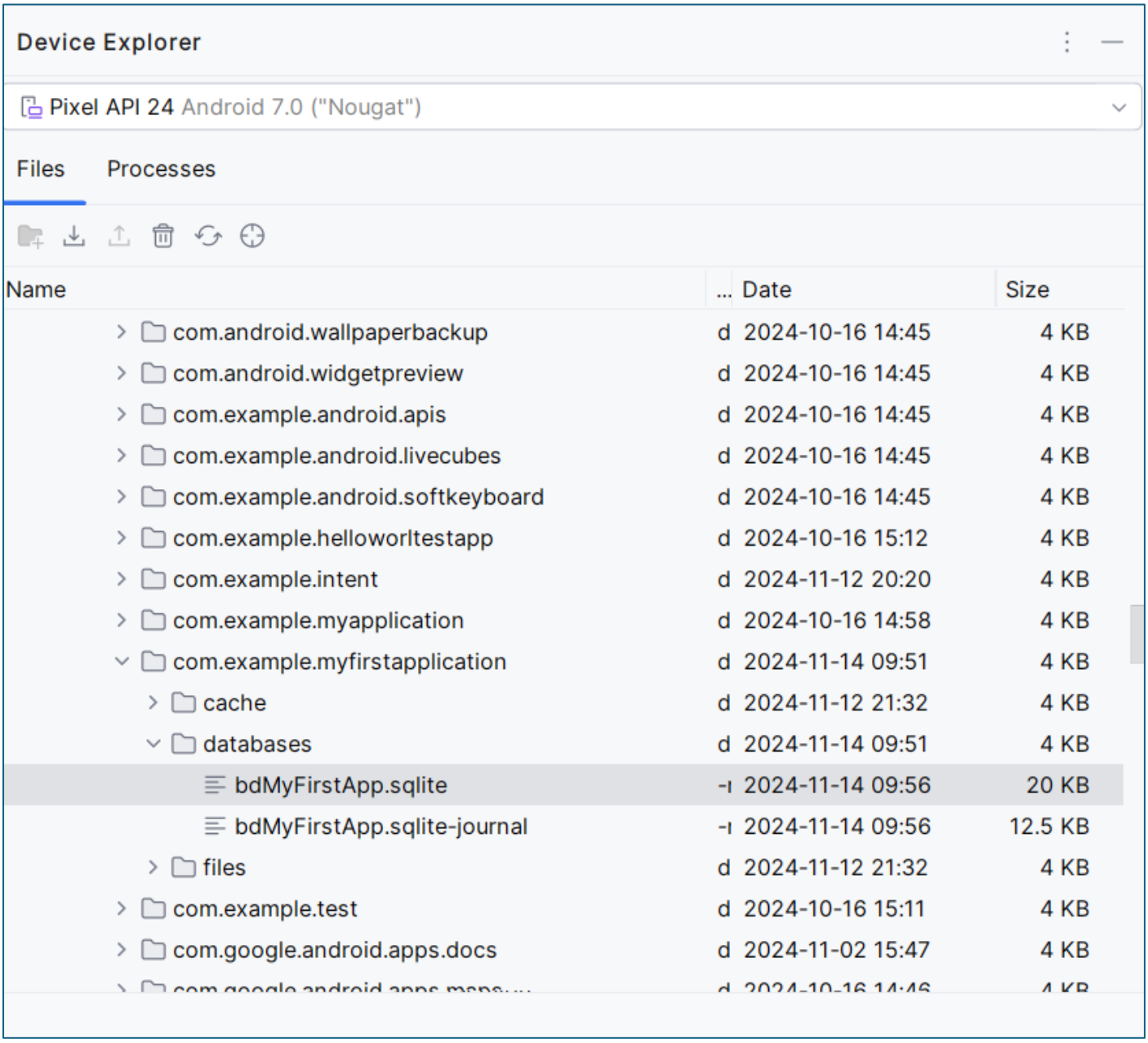
- Lancez l'application et entrez des données pour le profil (par exemple, poids, taille, âge), puis appuyez sur le bouton "Calculer" pour voir le résultat.
- Fermez et relancez l'application pour vérifier que le dernier profil saisi est bien affiché.

- **Test de Scénarios** :
    - Entrez des valeurs différentes et relancez l'application pour vérifier qu'elle récupère bien les dernières valeurs enregistrées.

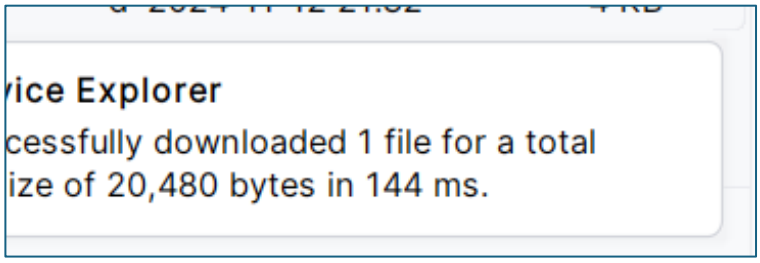


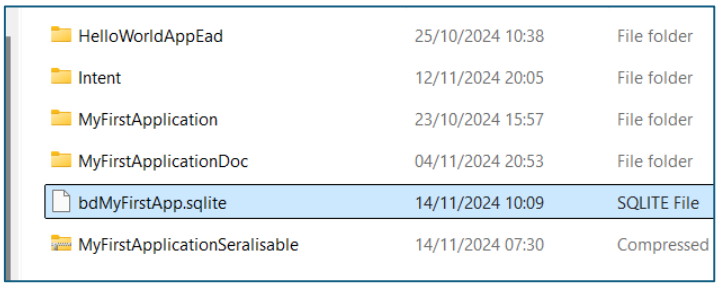

**3. Vérification de la Base de Données SQLite avec Android Device Monitor**

- Ouvrez **Android Device Monitor** pour accéder à la base de données SQLite.
- Naviguez dans les dossiers :
    - Accédez à data > data > votre package d'application (com.votreapp.coach par exemple).
    - Ouvrez le dossier databases pour voir le fichier bdCoach.sqlite.




- **Exportation de la Base de Données** :
  - Utilisez l'option "Pull a file from the device" pour télécharger le fichier SQLite sur votre ordinateur pour inspection.

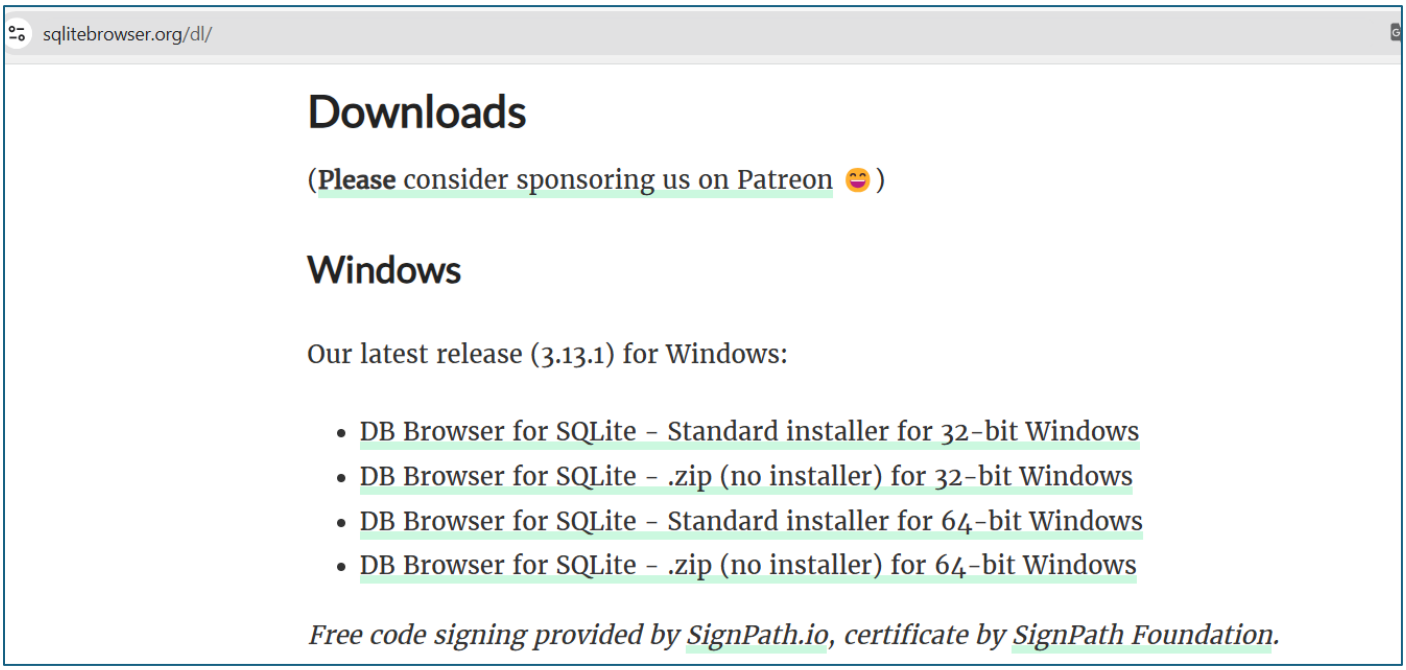


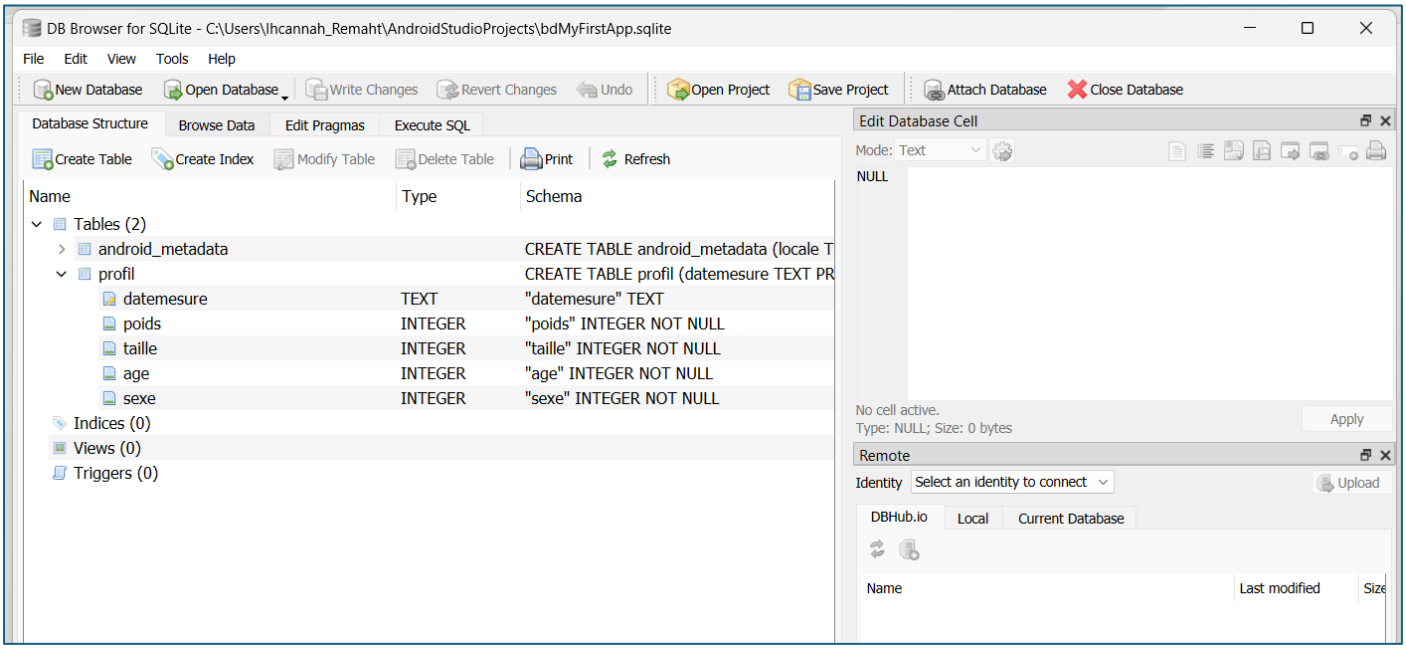

**4. Inspection de la Base de Données avec SQLite Browser**

- **Télécharger SQLite Browser** : Téléchargez et installez le logiciel à partir de sqlitebrowser.org.

Hananchi Thamer




- Ouvrez la base de données (bdCoach.sqlite) dans SQLite Browser.




- **Vérification des Tables et des Données** :
  - Allez dans l'onglet **"Browse Data"** et sélectionnez la table profil.
  - Vérifiez que les données saisies dans l'application apparaissent bien, y compris le dernier profil enregistré.

Hananchi Thamer

DB Browser for SQLite - C:\Users\Ihcannah_Remaht\AndroidStudioProjects\bdf

File Edit View Tools Help

New Database | Open Database | Write Changes | Revert Changes

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: profil

| | datemesure | poids | taille | age | sexe |
|---|---|---|---|---|---|
| | Filter | Filter | Filter | Fil... | Filter |
| 1 | Thu Nov 14 09:51:43 GMT+01:00 2024 | 80 | 190 | 30 | 0 |
| 2 | Thu Nov 14 09:51:54 GMT+01:00 2024 | 80 | 190 | 30 | 0 |
| 3 | Thu Nov 14 09:52:11 GMT+01:00 2024 | 80 | 160 | 19 | 1 |
| 4 | Thu Nov 14 09:52:31 GMT+01:00 2024 | 80 | 160 | 19 | 1 |
| 5 | Thu Nov 14 09:52:54 GMT+01:00 2024 | 180 | 160 | 19 | 1 |
| 6 | Thu Nov 14 09:53:04 GMT+01:00 2024 | 180 | 160 | 19 | 1 |
| 7 | Thu Nov 14 09:54:34 GMT+01:00 2024 | 120 | 160 | 32 | 0 |
| 8 | Thu Nov 14 09:55:25 GMT+01:00 2024 | 50 | 190 | 32 | 1 |
| 9 | Thu Nov 14 09:56:14 GMT+01:00 2024 | 50 | 160 | 25 | 0 |

Hananchi Thamer