Compte rendu :

TP4 Développement Mobile : Création d'une application de calcul d'Indice de Masse Grasse (IMG)

Enseignant : Maddouri Faouzi

Réalisé par :

Guesmi Ikram

Farhaoui Eya

Année universitaire : 2024/2025

Etablissement: ISET Rades

Objectif

`Partie 1:` enregistrer un profil utilisateur (composé des données : poids, taille, âge, sexe) par sérialisation dans un fichier binaire. Ce fichier sera ensuite récupéré lors de l'exécution suivante de l'application .

`Partie 2:`Enregistrement de plusieurs profils dans une base de données locale SQLite.

1 Partie 1: Création de la Serializer

creation d'un package outils dans ce package on a crée une classe `Serializer`. Elle permet de sauvegarder un objet `Profile` dans un fichier binaire et de le récupérer lors de l'ouverture de l'application. au niveau de cette classe on a crée les méthodes `serialize()` et `deserialize()` qui assurent l'écriture et la lecture des objets depuis un fichier.

Code: Serializer Class

```java
package com.example.myapplication.outils;

import android.content.Context;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.StreamCorruptedException;

public abstract class Serializer {

    public static void serialize(String filename, Object object, Context context){
        try{
            FileOutputStream file = context.openFileOutput(filename, Context.MODE_PRIVATE);
            ObjectOutputStream oos;
            try {
                oos = new ObjectOutputStream(file);
                oos.writeObject(object);
                oos.flush();
                oos.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }

    public static Object deserialize(String filename, Context context){
        try{
            FileInputStream file = context.openFileInput(filename);
            ObjectInputStream ois;
            try {
                ois = new ObjectInputStream(file);
                try {
                    Object object = ois.readObject();
                    ois.close();
                    return object;
                } catch (ClassNotFoundException e) {
                    e.printStackTrace();
                }
            } catch (StreamCorruptedException e) {
                e.printStackTrace();
            } catch(IOException e) {
                e.printStackTrace();
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
```

```
51          }
52          return null;
53      }
54  }
```

2 Sauvegarde du profil après calcul

Après que l'utilisateur ait calculé son IMG, les informations du profil sont sauvegardées dans un fichier binaire afin qu'elles puissent être récupérées lors de la prochaine exécution de l'application.

3 Conclusion

Grâce à la sérialisation des données du profil utilisateur, l'application permet à l'utilisateur de sauvegarder ses informations personnelles et de les retrouver à chaque ouverture de l'application.

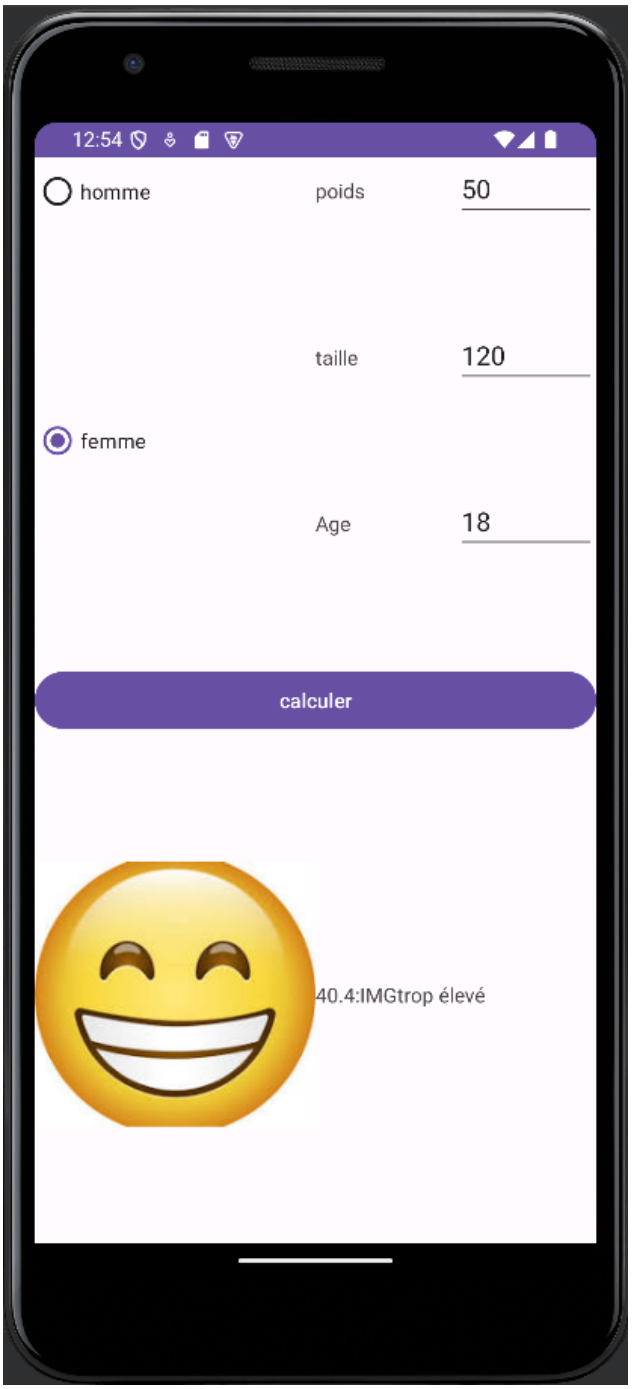

Figure 1: Exemple de sauvegarde du profil utilisateur.

4 Partie 2: Base de données locale SQLite

4.1 La classe `MySQLiteOpenHelper`

Dans ce projet, nous utilisons une base de données SQLite, ce qui nécessite de gérer la création et la mise à jour de la base de données. Pour ce faire, nous avons créé la classe `MySQLiteOpenHelper` qui hérite de `SQLiteOpenHelper`. Elle permet à l'application de créer la base, de gérer les versions et de fournir un accès facile aux données (ajout, récupération, etc.).

```
public class MySQLiteOpenHelper extends SQLiteOpenHelper {
    1 usage
    private String creation="create table profil("
            +"datemesure Text PRIMARY Key,"
            +"poids INTEGER NOT NULL,"
            +"taille INTEGER NOT NULL,"
            +"age INTEGER NOT NULL,"
            +"sexe INTEGER NOT NULL);";


    1 usage
    public MySQLiteOpenHelper(@Nullable Context context, @Nullable String name, @Nu
        super(context, name, factory, version);
    }
/*ne s execute que si il y a un changement dans la base de donnée ne se fait qu une
    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        sqLiteDatabase.execSQL(creation);

    }
/* si il ya une changement de version */
    no usages
    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

    }
```

Figure 2: La classe MySQLiteOpenHelper

4.2 La méthode `recupProfil()` de la classe MainActivity

La méthode `recupProfil()` de la classe `MainActivity` permet de récupérer et d'afficher les informations de profil sauvegardées précédemment dans l'application. Elle est particulièrement utile pour restaurer l'état du profil lorsque l'utilisateur redémarre l'application, permettant ainsi de ne pas avoir à ressaisir les données.

```
1 usage
private void recupProfil() {
    if (controle.getPoids() != null) {
        txtPoids.setText(controle.getPoids().toString());
        txtTaille.setText(controle.getTaille().toString());
        txtAge.setText(controle.getAge().toString());
        rdFemme.setChecked(true);
        if (controle.getSexe() == 1) {
            rdHomme.setChecked(true);
        }
        ((Button) findViewById(R.id.btnCalc)).performClick();
    }
}
```

Figure 3: `recupProfil()` de la classe MainActivity

4.3 La méthode `recupDernier()` de la classe AccessLocal

La méthode `recupDernier()` appartient à la classe `AccessLocal`, qui gère l'accès aux données stockées dans la base SQLite. Elle est utilisée pour récupérer le dernier profil enregistré dans la base de données. Cette méthode permet d'afficher le dernier profil lorsque l'utilisateur redémarre l'application ou lorsque l'on souhaite restaurer les dernières données saisies.

```
1 usage
public Profil recupDernier(){
        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;
}
```

Figure 4: `recupDernier()` de la classe AccessLocal

4.4 Comportement de l'application lors de l'arrêt et du redémarrage

Lors de l'arrêt de l'application, l'objet Profil est sauvegardé grâce à Serializer.serialize() lorsque la méthode creerProfil() est appelée.

Lors du redémarrage de l'application,Controle.getInstance(this) est appelée dans la méthode onCreate(), et recuperSerialize() charge l'objet Profil sauvegardé. Si la méthode recupProfil() est appelée après l'initialisation, elle remplit les champs de l'interface avec les données sauvegardées et déclenche un calcul automatique pour afficher le résultat.

Cela permet d'améliorer la continuité des données entre les sessions, offrant ainsi une meilleure expérience utilisateur.

4.5 Étapes pour visualiser la base de données

- Accédez à **VIEW**, choisissez **Tool windows**, puis allez dans **Device Explorer**.
- allez au dossier **databases**, et telechargez le fichier **bdCoach.sqlLte**
- Créez un nouveau dossier pour enregistrer le fichier `.sqlite` téléchargé.

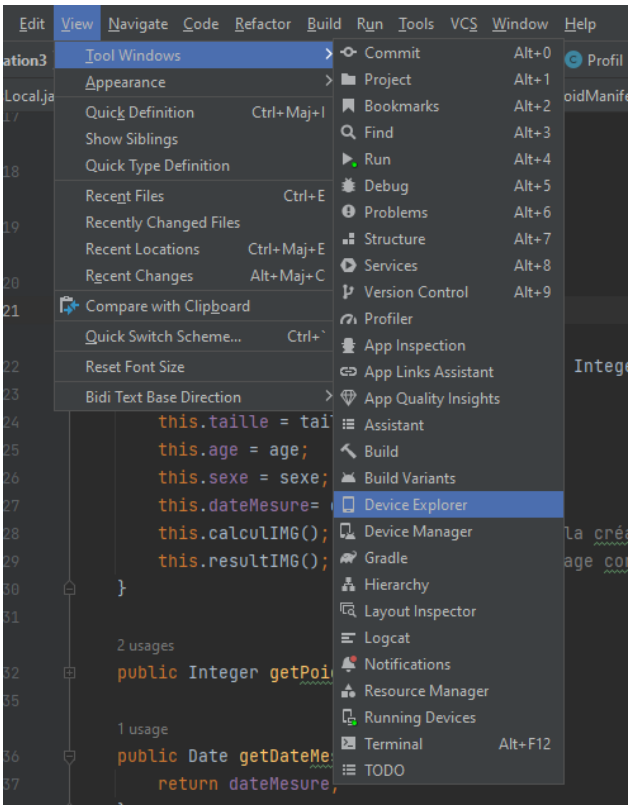

Figure 5: La première étape de la visualisation de la base de données.

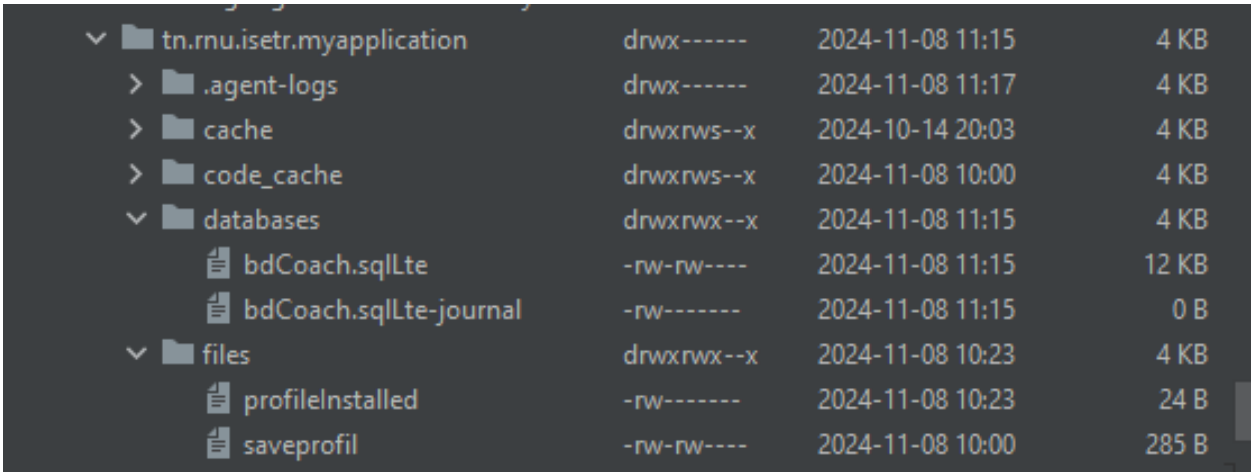

Figure 6: La deuxième étape de la visualisation de la base de données.

- Téléchargez **DB Browser for SQLite**, allez à **Nouvelle base de données** et ajoutez le fichier `.sqlite` que vous avez téléchargé au préalable.

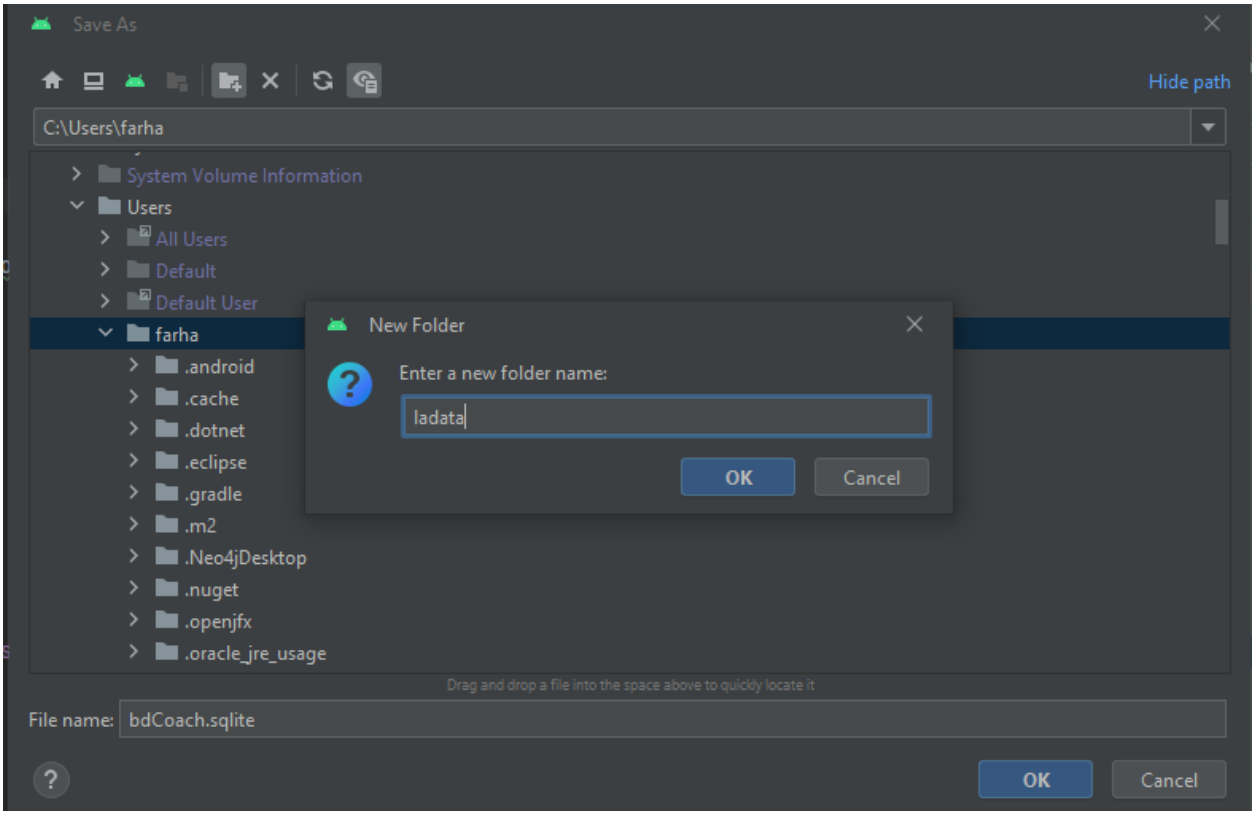

Figure 7: La troisième étape de la visualisation de la base de données.

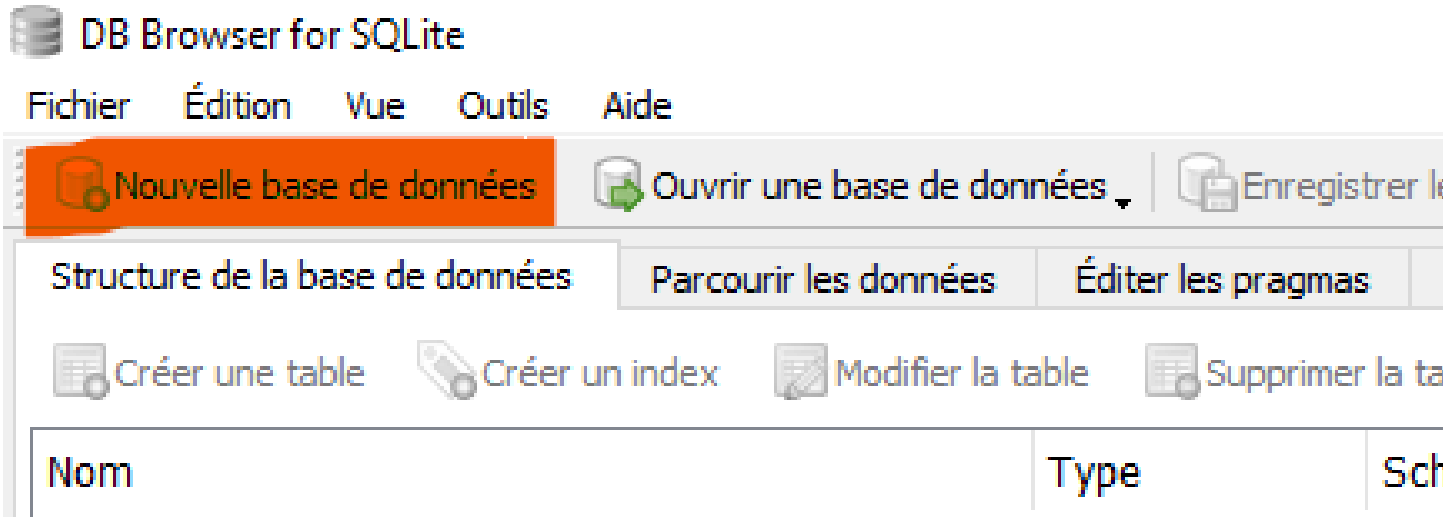

Figure 8: DB Browser for SQLite.