Initiation au développement Android
TP05 - Biblio

Objectifs :

- **Mise en place d'une application de gestion de livres**
- **Persistance des données assurée par une base de données embarquée SQLite**

Autres mécanismes de persistance des données pour les applications Android :

- *Fichiers bruts* (`FileOutputStream`)
- *Préférences* (`SharedPreferences`) : informations enregistrées dans un fichier sous la forme de paires clé/valeur

Base de données SQLite

Android intègre nativement un moteur de base de données, nommé **SQLite**

Quelques caractéristiques SQLite :

- Moteur de base de données relationnel open source qui supporte les fonctionnalités standards (syntaxe SQL, transactions,…)
- SGBD très compact qui nécessite peu de mémoire lors de l'exécution (env. 250 ko) et par conséquent très efficace pour les applications embarquées
- SQLite ne nécessite pas de serveur pour fonctionner, ce qui signifie que son exécution se fait dans le même processus que celui de l'application

Etape 1 - Prise en main du projet « Biblio »

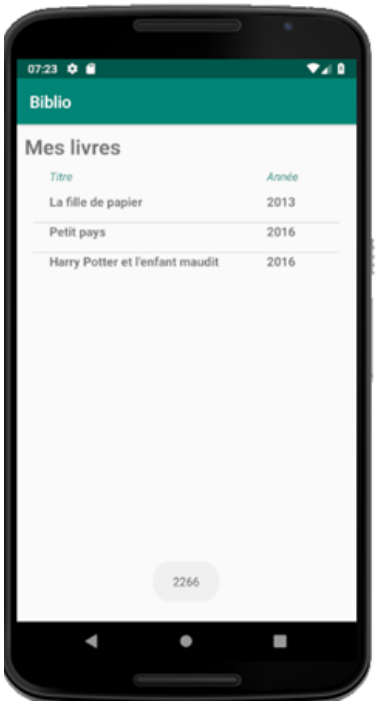
- Récupérer puis importer le projet `Biblio` disponible dans le dossier `05TP_DevMobile_SQLite`
- Tester l'application `Biblio`

Comportement

L'application affiche la liste des livres stockés dans la base de données embarquée.

Pour chaque livre, sont affichés le titre et l'année de sa parution (l'ISBN du livre est rendu invisible)

Un click sur un livre affiche dans un *Toast* l'ISBN de celui-ci (dans l'exemple, l'utilisateur a diqué sur le livre « La fille de papier »)



Classe `Livre.java`

```
public class Livre {
    // attributs privés
    private String isbn;
    private String titre;
    private Integer annee;
    private String auteur;
    private Integer nbPages;

    // constructeurs
    public Livre() {}

    public Livre(String isbn, String titre, Integer annee, String auteur, Integer nbPages) {
        this.isbn = isbn;
        this.titre = titre;
        this.annee = annee;
        this.auteur = auteur;
        this.nbPages = nbPages;
    }

    // getteurs - setteurs
    public String getIsbn() {return isbn;}
    public void setIsbn(String isbn) { this.isbn = isbn; }
    public String getTitre() {return titre;}
    public void setTitre(String titre) {this.titre = titre;}
    public Integer getAnnee() {return annee;}
    public void setAnnee(Integer annee) {this.annee = annee;}
    public String getAuteur() {return auteur;}
    public void setAuteur(String auteur) {this.auteur = auteur;}
    public Integer getNbPages() {return nbPages;}
    public void setNbPages(Integer nbPages) {this.nbPages = nbPages;}
}
```

Classe Métier : la classe `Livre.java` contient les attributs privés, le constructeur et les accesseurs (getteurs/setteurs)

Base de données embarquée SQLite

☑ Classe `BiblioHelper` : fichier `BiblioHelper.java`

Cette classe gère la création du schéma (structure) de la base de données embarquée ainsi que la création d'un jeu de données.

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;


public class BiblioHelper extends SQLiteOpenHelper {


    public BiblioHelper(Context context)
    {
        super(context, "baseBiblio.db", null, 1);
    }


    @Override
    public void onCreate(SQLiteDatabase db) {
        // création des tables de la base embarquée
        // création de la table LIVRE
        db.execSQL("CREATE TABLE Livre ("
                + "isbnLivre TEXT NOT NULL PRIMARY KEY,"
                + "titreLivre TEXT NOT NULL,"
                + "anneeLivre INTEGER NOT NULL,"
                + "auteurLivre TEXT NOT NULL,"
                + "pagesLivre INTEGER);");

        // création d'un jeu d'essai
        Livre lesLivres[]= {
                new Livre("2266","La fille de papier",2013,"Guillaume Musso",608),
                new Livre("7333", "Petit pays", 2016, "Gaël Faye", 224),
                new Livre("4208","Harry Potter et l''enfant maudit",2016,"J.K Rowling",352)};

        for (Livre liv : lesLivres) {
            db.execSQL("INSERT INTO Livre " +
                    " VALUES ('"+liv.getIsbn()+"','"+liv.getTitre()+"','"+liv.getAnnee()+"','"
                                                +liv.getAuteur()+"','"+liv.getNbPages() +"');");
        }

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS Livre;");
        onCreate(db);
    }

}
```

SQLiteOpenHelper : Classe (abstraite) utilitaire qui permet de gérer la création et la mise à jour des bases de données
La classe `BiblioHelper` étend la classe `SQLiteOpenHelper` et définit le constructeur ainsi que les méthodes `onCreate` et `onUpdate`

Paramètres du constructeur
- Nom du fichier qui contient la base
- N° de version à incrémenter en cas de modification du schéma de la base

onCreate : Contient les instructions qui permettent de créer la structure de chaque table de la base SQLite (ici notre base contient une seule table LIVRE) et éventuellement d'insérer un jeu de données initial.
Cette méthode est appelée si la BDD n'est pas encore créée.

Nombre de types réduit : *INTEGER, REAL, TEXT, BLOB (objet binaire - image-)*

Contraintes : *PRIMARY KEY, NOT NULL, CHECK, DEFAULT, REFERENCES*
(rem : la prise en charge des Foreign Keyne se fait pas par défaut)

onUpgrade : Android gère un numéro de version pour la base. Cette méthode est appelée si la version de la base de données est augmentée dans le code de l'application (cas d'une mise à jour du schéma de la base par exemple).

☑ Classe BiblioDAO : fichier BiblioDAO.java

Cette classe permet de gérer l'accès aux données de la base de données embarquée : requêtes de sélection, d'insertion, de modification et de suppression

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.Cursor;

public class BiblioDAO {

    private SQLiteDatabase maBase;
    private BiblioHelper maBiblioHelper;

    public BiblioDAO(Context context){
        maBiblioHelper = new BiblioHelper(context);
        maBase = maBiblioHelper.getWritableDatabase();
    }

    public Cursor selectionnerTousLesLivres() {
        Cursor curseurContact = maBase.rawQuery("SELECT isbnLivre, titreLivre, anneeLivre from Livre" , new String[] {});
        return curseurContact;
    }

}
```

SQLiteDataBase : Classe qui représente une instance de la BD et utilisée pour réaliser les opérations sur la base (lecture, insertion, modification, suppression - CRUD)

getWritableDatabase : Permet de récupérer un identifiant qui sera utilisé pour accéder à la base de données en Lecture et/ou Ecriture

Cette méthode déclenche le `onCreate` ou le `onUpgrade` de la classe `BiblioHelper`

Cursor : objet qui permet de pointer vers le résultat retourné par une requête de sélection

`getCount()` retourne le nombre de lignes,
`moveToFirst()` retourne la 1ère ligne (ou `null`),
`moveToNext()` indique s'il y a une autre ligne ou non,...

Activité MainActivity

☑ Layout de l'activité : fichier activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="@string/titre"
        android:textSize="25sp"
        android:textStyle="bold" />

    <ListView
        android:id="@+id/lv_livres"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp">
    </ListView>

</LinearLayout>
```

Fichier strings.xml

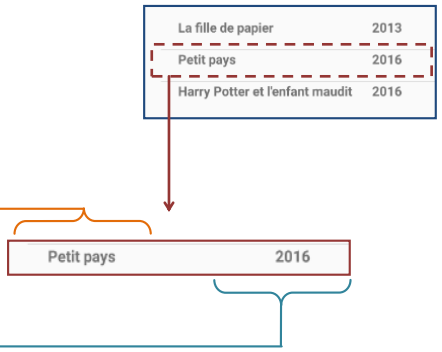
```
<resources>
    <string name="app_name">Biblio</string>
    <string name="titre">Mes livres</string>
</resources>
```

☑ Layout d'une ligne de la *ListView* : fichier layout_ligne_lvlivre.xml

Création d'un fichier de ressource *Layout* permettant de définir le style des lignes de la *ListView*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
    <TextView
        android:id="@+id/col_isbnLivre"
        android:layout_width="20dp"
        android:layout_height="wrap_content"
        android:visibility="invisible"/>
    <TextView
        android:id="@+id/col_titreLivre"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="3"
        android:textSize="15sp"
        android:textStyle="bold"/>
    <TextView
        android:id="@+id/col_anneeLivre"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textSize="15sp"
        android:textStyle="bold"/>
</LinearLayout>
```

Cette zone, prévue pour contenir l'isbn du livre, est masquée



layout_weight : La propriété poids permet d'agir sur la taille des éléments sur un principe de proportionnalité
Attention : il faut placer la dimension (ici la largeur) à 0

☑ Layout de l'entête de la *ListView* : fichier `layout_header_lvlivre.xml`

Création d'un fichier de ressource *Layout* permettant de définir le style de l'entête de la *ListView*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
    <TextView
        android:layout_width="20dp"
        android:layout_height="wrap_content"
        android:visibility="invisible"
        android:layout_marginBottom="10dp"/>
    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="3"
        android:text="Titre"
        android:textColor="@color/colorPrimary"
        android:textStyle="italic"
        android:layout_marginBottom="10dp"/>
    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Année"
        android:textColor="@color/colorPrimary"
        android:textStyle="italic"
        android:layout_marginBottom="10dp"/>
</LinearLayout>
```

☑ Contrôleur de l'activité : fichier `MainActivity.java`

```
public class MainActivity extends AppCompatActivity {
    private ListView lstLivres = null;
    private BiblioDAO maBDD;
    private ArrayList<HashMap<String,String>> lesLivres;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        lstLivres = (ListView)findViewById(R.id.lv_livres);
        // construction entête ListView
        enteteLstLivres();
        // chargement de la ListView
        initLstLivres();

        // écouteur d'évènement sur la listView
        lstLivres.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> a, View v, int position, long id) {
                //on récupère la HashMap contenant les infos de l'item sélectionné
                HashMap<String, String> item = (HashMap<String, String>) lstLivres.getItemAtPosition(position);
                //on récupère l'isbn sélectionné et on l'affiche
                String isbnSelect = item.get("ISBN");
                Toast msg = Toast.makeText(MainActivity.this, isbnSelect, Toast.LENGTH_SHORT);
                msg.show();
            }
        });
    }
    @Override
    protected void onResume () {
        super.onResume();
        initLstLivres();
    }

    // méthode qui construit l'entête de la liste des livres
    public void enteteLstLivres() {
        // ajout d'un header à la listView
        LayoutInflater layoutInflater = getLayoutInflater();
        View header = layoutInflater.inflate(R.layout.layout_header_lvlivre,null,false);
        lstLivres.addHeaderView(header,null,false);
    }

    // méthode qui charge la liste des livres
    public void initLstLivres() {
        // récupération des livres enregistrés dans la base et initialisation d'un tableau associatif
        maBDD = new BiblioDAO(this);
        Cursor curseurTous = maBDD.selectionnerTousLesLivres();
        lesLivres = new ArrayList<HashMap<String, String>>();
        for(curseurTous.moveToFirst(); !curseurTous.isAfterLast(); curseurTous.moveToNext()) {
            String isbn= curseurTous.getString(curseurTous.getColumnIndex("isbnLivre"));
            String titre = curseurTous.getString(curseurTous.getColumnIndex("titreLivre"));
            String annee = curseurTous.getString(curseurTous.getColumnIndex("anneeLivre"));
            HashMap<String,String> unLivre = new HashMap<String, String>();
            unLivre.put("ISBN", isbn);
            unLivre.put("TITRE", titre);
            unLivre.put("ANNEE", annee);
            lesLivres.add(unLivre);
        }
        curseurTous.close();


        // remplissage de la listView en utilisant un adapter personnalisé
        lstLivres.setAdapter(new SimpleAdapter( this,
                lesLivres,
                R.layout.layout_ligne_lvlivre,
                new String[] {"ISBN", "TITRE", "ANNEE"},
                new int[] {R.id.col_isbnLivre, R.id.col_titreLivre, R.id.col_anneeLivre}));
    }
}
```

OnItemClickListener : *listener* qui permet de déclencher un traitement lorsque l'utilisateur clique sur un item de la *ListView*

`position` : index de l'élément sélectionné

`getItemAtPosition()` : retourne l'item à l'index spécifié

onResume : dans le cycle de vie d'une activité, la méthode `onResume()` est appelée lorsque l'activité passe au premier plan

LayoutInflater : objet qui permet de construire un objet `View` (fragment) à partir d'un fichier de définition XML
La vue ainsi construite est ajoutée comme `header` à la `listView`

SimpleAdapter : utilisé pour combiner les données à afficher avec le style des items de *ListView*
Paramètres :
- *contexte* : activité courante
- *données à afficher* : liste de *HashMap* (couple clé/valeur) obligatoirement
- *layout* : référence au *layout (style)* personnalisé
- correspondance donnée / vue du layout

TRAVAIL A FAIRE - Pour vérifier que vous avez compris …

Mettre en place les évolutions suivantes :

1. **Dans la liste des livres, on souhaite pouvoir visualiser le nombre de pages du livre**
2. **On souhaite stocker dans la base de données embarquée le nom de l'éditeur du livre**

Etape 2 - Mise en place de l'activité « GererLivre »

☑ Layout de l'activité : fichier *activity_gerer_livre.xml*

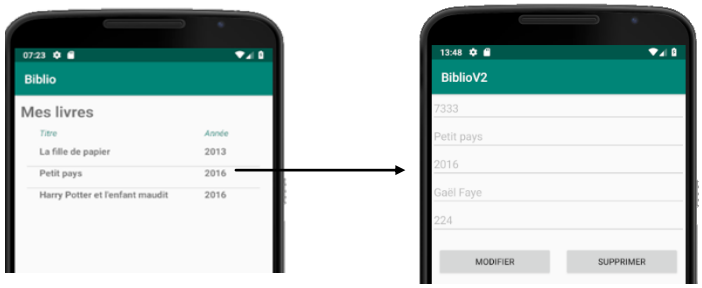
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"  >
    <EditText
        android:id="@+id/isbn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/textIsbn"/>
    <EditText
        android:id="@+id/titre"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/textTitre"/>
    <EditText
        android:id="@+id/annee"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/textAnnee"
        android:inputType="number"/>
    <EditText
        android:id="@+id/auteur"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/textAuteur"/>
    <EditText
        android:id="@+id/nbPages"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/textNbPages"
        android:inputType="number"/>
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:orientation="horizontal" >
        <Button
            android:id="@+id/modifier"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:layout_gravity="center"
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp"
            android:text="@string/textModif"/>
        <Button
            android:id="@+id/supprimer"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:layout_gravity="center"
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp"
            android:text="@string/textSuppr" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:orientation="horizontal">
        <Button
            android:id="@+id/valider"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:layout_gravity="center"
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp"
            android:text="@string/textValid" />
        <Button
            android:id="@+id/annuler"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:layout_gravity="center"
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp"
            android:text="@string/textAnnul" />
    </LinearLayout>
</LinearLayout>
```

☑ Chaînes de caractères : fichier *strings.xml*

```
<resources>
    <string name="app_name">Biblio</string>
    <string name="titre">Mes livres</string>
    <string name="textIsbn">ISBN</string>
    <string name="textTitre">Titre</string>
    <string name="textAnnee">Année parution</string>
    <string name="textAuteur">Auteur</string>
    <string name="textNbPages">Nombre pages</string>
    <string name="textModif">Modifier</string>
    <string name="textSuppr">Supprimer</string>
    <string name="textValid">Valider</string>
    <string name="textAnnul">Annuler</string>
</resources>
```

☑ Comportement attendu

1. Lorsque l'utilisateur sélectionne un livre, l'activité `GererLivre` se lance affichant les informations sur le livre. Le visuel de l'activité `GererLivre` doit ressembler à ceci :



Les informations affichées sont en lecture seule

Seuls les boutons MODIFIER et SUPPRIMER sont visibles

Quelques éléments pour l'affichage des informations du livre :

✓ Ajout d'une méthode `selectionnerUnLivre()` dans le fichier `BiblioDAO.java`:

```
public Livre selectionnerUnLivre(String unISBN) {
    // création d'un curseur stockant le résultat de la requête
    Cursor curseurLivre = maBase.rawQuery("SELECT * FROM Livre WHERE isbnLivre = ?", new String[]{unISBN});
    // lecture du curseur
    curseurLivre.moveToFirst();
    Livre unLivre = new Livre();
    unLivre.setIsbn(curseurLivre.getString(0));
    unLivre.setTitre(curseurLivre.getString(1));
    unLivre.setAnnee(curseurLivre.getInt(2));
    unLivre.setAuteur(curseurLivre.getString(3));
    unLivre.setNbPages(curseurLivre.getInt(4));
    // fermeture du curseur
    curseurLivre.close();
    return unLivre;
}
```

✓ Apparence des vues de l'activité

```
editISBN.setEnabled(false); // rend inaccessible la vue
btnValider.setVisibility(View.INVISIBLE); // rend invisible la vue
```

2. Un click sur le bouton `Supprimer` permet de supprimer le livre après confirmation de l'utilisateur ; on revient ensuite sur l'activité `MainActivity` (on constate que la liste des livres a été actualisée)

Quelques éléments pour la suppression du livre :

✓ Ajout d'une méthode `supprimerLivre()` dans le fichier le fichier `BiblioDAO.java`:

```
public void supprimerLivre(String unISBN) {
    maBase.delete("Livre", "isbnLivre = ?", new String[] {unISBN});
}
```

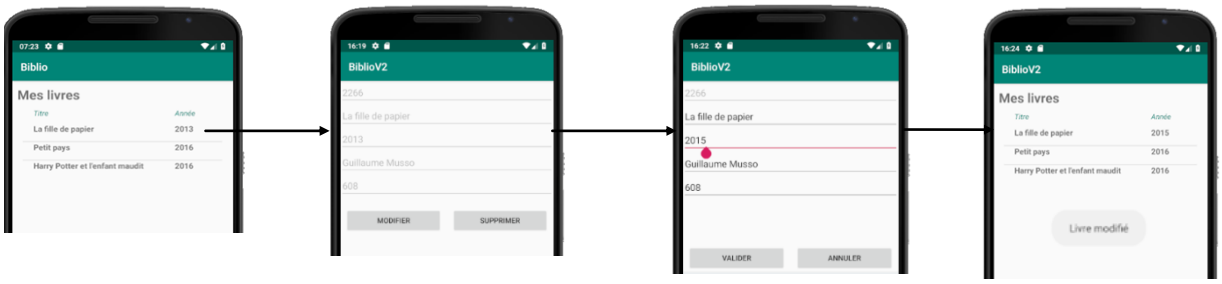
whereClause : Correspond à la clause Where en SQL, permet de spécifier une ou plusieurs conditions

✓ Gestion de la boîte de dialogue

```
AlertDialog.Builder boite = new AlertDialog.Builder(GererLivre.this);
boite.setTitle("Confirmation"); boite.setMessage("Voulez-vous supprimer ce livre ?");
boite.setPositiveButton("OUI", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) { // traitement_OUI }
});
boite.setNegativeButton("NON", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) { // traitement_NON }
});
boite.show();
```

Utilisation de la méthode `dismiss()` de la classe `DialogInterface` pour faire disparaître la boîte de dialogue

3. Un click sur le bouton `Modifier` permet à l'utilisateur de modifier les informations sur le livre :
   - les différentes zones sont éditables, le curseur est placé sur la première zone
   - les boutons `Valider` et `Annuler` apparaissent (ils étaient invisibles), les boutons `Modifier` et `Supprimer` sont masqués
   - si l'utilisateur clique sur `Annuler`, on revient à l'état précédent
   - si l'utilisateur clique sur `Valider`, les modifications sont prises en compte dans la base de données ; un message en informe l'utilisateur
   - on revient ensuite sur l'activité `MainActivity` (la liste des livres a été actualisée)



Quelques éléments pour la modification d'un livre :

✓ Ajout d'une méthode `modifierLivre()` dans le fichier le fichier `BiblioDAO.java`:

```
public void modifierLivre(Livre unLivre) {
    //création d'un ContentValues
    ContentValues v = new ContentValues();
    // ajout des propriétés au ContentValues
    v.put("isbnLivre", unLivre.getIsbn());
    v.put("titreLivre", unLivre.getTitre());
    v.put("anneeLivre", unLivre.getAnnee());
    v.put("auteurLivre", unLivre.getAuteur());
    v.put("pagesLivre", unLivre.getNbPages());
    // modifie le livre dans la table
    maBase.update("Livre", v, "isbnLivre = ?", new String[]{unLivre.getIsbn()});
}
```

ContentValues : objet qui représente une ligne d'une table sous la forme d'un ensemble de paires clé/valeur ; La clé correspond au nom d'une colonne de la table et valeur, la valeur à attribuer à cette colonne

Etape 3 - Mise en place du bouton *Ajouter* sur l'activité « MainActivity »

☑ Comportement attendu

L'activité `MainActivity` contient un bouton `Ajouter`. Un click sur ce bouton lance l'activité `GererLivre` :

- toutes les zones sont vides, le curseur est placé sur la première zone
- les boutons `Valider` et `Annuler` apparaissent (ils étaient invisibles), les boutons `Modifier` et `Supprimer` sont masqués
- si l'utilisateur clique sur `Annuler`, on revient à l'état précédent
- si l'utilisateur clique sur `Valider`, le nouveau livre est ajouté dans la base de données ; un message en informe l'utilisateur
- on revient ensuite sur l'activité `MainActivity` (la liste des contacts a été actualisée)

Quelques éléments pour l'ajout d'un livre :

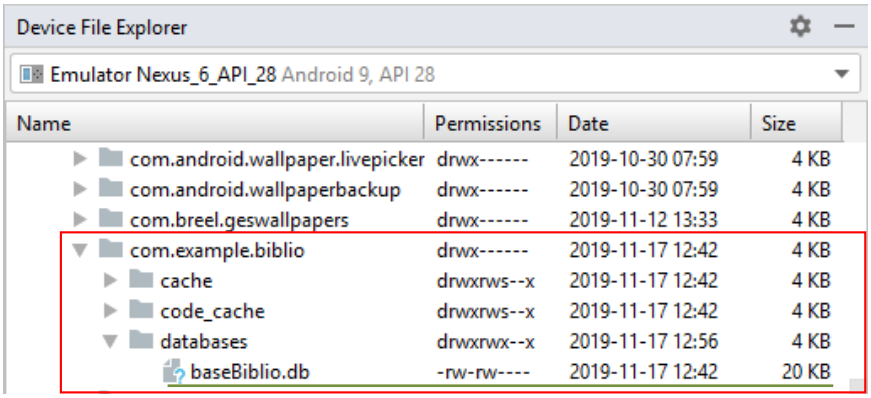
✓ Ajout d'une méthode `ajouterLivre()` dans le fichier le fichier `BiblioDAO.java`:

```
public void ajouterLivre(Livre unLivre) {
    //création d'un ContentValues
    ContentValues v = new ContentValues();
    // ajout des propriétés au ContentValues
    v.put("isbnLivre", unLivre.getIsbn());
    v.put("titreLivre", unLivre.getTitre());
    v.put("anneeLivre", unLivre.getAnnee());
    v.put("auteurLivre", unLivre.getAuteur());
    v.put("pagesLivre", unLivre.getNbPages());
    // ajout du livre dans la table
    maBase.insert("Livre", null, v);
}
```

DB Browser For SQLite : Outil d'administration de bases de données SQLite

☑ Extraire le fichier `baseBiblio.db` de l'émulateur

- Le fichier de base de données se trouve dans le dossier *data/data/com.example.biblio/databases* du système de fichier de l'émulateur
- Utilisation de l'onglet `Device File Explorer` (onglet en bas à droite dans Android Studio) pour accéder à l'explorateur de fichiers de l'émulateur



Faire un click-droit pour extraire et enregistrer le fichier *baseBiblio.db* dans le dossier de votre choix

☑ Visualiser le fichier `baseBiblio.db` (extrait et copié sur l'ordinateur) sous DB Browser For SQLite (Menu *File / Open Database)*