TP 5 : Développer une application Android – Gestion de contacts avec SQLite

Objectif :

L'objectif de ce TP est de créer une application Android qui permet à l'utilisateur de gérer une liste de contacts, incluant l'ajout, la modification, la suppression et l'affichage des contacts à l'aide d'une base de données SQLite locale. Vous utiliserez des concepts d'intents pour passer des données entre les activités et stocker les informations dans la base de données.

Détails de l'application :

1. Base de données SQLite :

- Vous devrez créer une base de données SQLite pour stocker les contacts avec les informations suivantes : ID, Nom, et Numéro de téléphone.
- La table devra avoir les colonnes : ID (clé primaire), Nom (chaîne de texte), et Numéro (chaîne de texte).
- Pour cela vous créez une classe java nommée DataBaseContacts qui étend la classe SQLiteOpenHelper et dans laquelle vous créer les méthodes suivantes :
  - ✓ onCreate(SQLiteDatabase db)
  - ✓ addContact (String name, String number)
  - ✓ updateContact (int id, String name, String number)
  - ✓ deleteContact (int id)
  - ✓ getContactById (int id)
  - ✓ getAllContacts ()

2. Écran principal (MainActivity) :

- **Champs de saisie pour les informations de contact :** Utilisez deux EditText pour que l'utilisateur puisse entrer le nom et le numéro du contact.
- **Validation des champs :** Avant de sauvegarder un contact, vérifiez que les deux champs sont remplis. Si l'un des champs est vide, affichez un message d'erreur indiquant : "Veuillez remplir tous les champs".
- **Boutons :**
  - ✓ **Ajouter un contact :** Ce bouton doit sauvegarder les informations dans la base de données lorsque les champs sont remplis.
  - ✓ **Voir la liste des contacts :** Ce bouton ouvre une nouvelle activité (ContactsListActivity) qui affiche tous les contacts enregistrés dans la base de données.

3. ContactsListActivity :

- Affichez tous les contacts sous forme de liste (utilisez ListView ou RecyclerView).
- Chaque élément de la liste doit être cliquable. Lorsqu'un contact est sélectionné, ouvrez une nouvelle activité pour modifier ou supprimer le contact.

4. EditContactActivity :

- Cette activité doit contenir deux EditText préremplis avec les informations du contact sélectionné.
- **Boutons :**
  - ✓ **Modifier :** Ce bouton doit mettre à jour les informations du contact dans la base de données avec les valeurs des EditText.
  - ✓ **Supprimer :** Ce bouton doit supprimer définitivement le contact de la base de données.

Étapes à suivre :

1. Créer un nouveau projet Android :

- Ouvrez Android Studio et créez un nouveau projet.
- Sélectionnez "Empty Views Activity" comme modèle de départ et nommez le projet : ContactsApp.

2. Créer la base de données :

- Créez une classe DatabaseContact pour gérer la base de données SQLite. Elle doit contenir des méthodes pour ajouter, modifier, supprimer et récupérer des contacts.
- Structure de la base de données :
  - ✓ `ID` (clé primaire)
  - ✓ `NAME` (nom du contact)
  - ✓ `TELNUMBER` (numéro du contact)

- Extrait du code de la classe DatabaseContacts (partiellement fourni) que vous pouvez utiliser pour créer votre propre classe :

```
public class DatabaseContacts extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "contacts.db";
    private static final String TABLE_NAME = "contacts";
    private static final String COL2 = "NAME";
    private static final String COL3 = "TELNUMBER";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, 1);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE " + TABLE_NAME + " (ID INTEGER PRIMARY
KEY AUTOINCREMENT, NAME TEXT, TELNUMBER TEXT)");
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }

    public boolean addContact(String name, String telnumber) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(COL2, name);
        contentValues.put(COL3, telnumber);
        long result = db.insert(TABLE_NAME, null, contentValues);
        return result != -1;
    }
    public boolean updateContact(int id, String name, String telnumber)
{
        SQLiteDatabase db = this.getWritableDatabase();
        …
    }    public Integer deleteContact(int id) {
        SQLiteDatabase db = this.getWritableDatabase();
        …
    }
    public Cursor getContactById(int id) {
        SQLiteDatabase db = this.getReadableDatabase();
        …
    }
    public Cursor getAllContacts() {
        …
    }
}
```

3. Conception de l'interface utilisateur (activity_main.xml) :

- Ajoutez deux EditText pour saisir le nom et le numéro de téléphone.
- Ajoutez deux boutons : un pour ajouter un contact et un autre pour afficher la liste des contacts.
- Utiliser les hint pour les editText et personnaliser votre interface avec des **couleurs** de votre choix

4. Conception de l'interface utilisateur pour ContactsListActivity (activity_contacts_list.xml) :

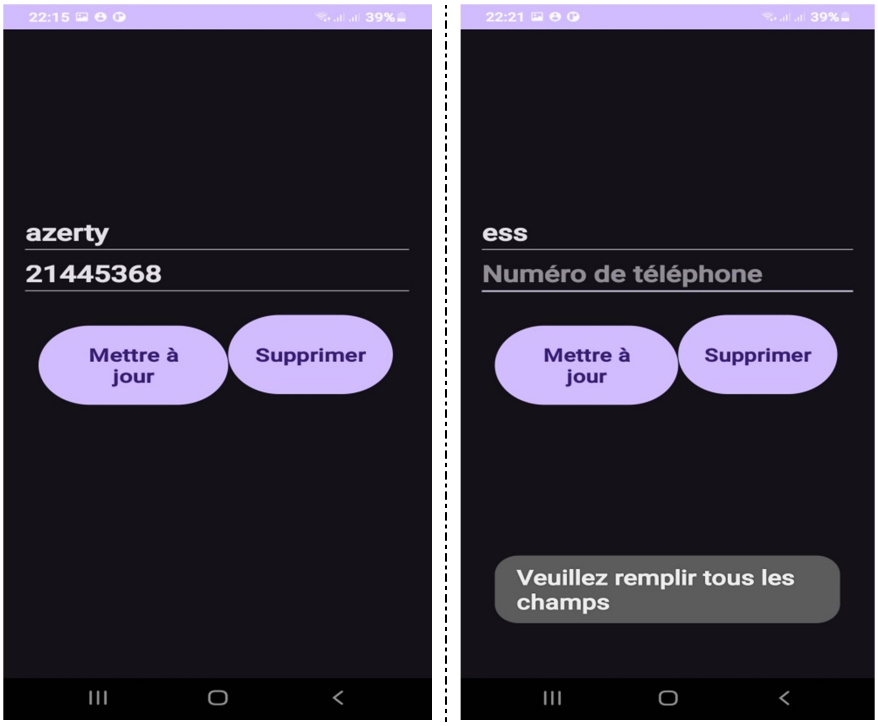
- Utilisez un ListView ou RecyclerView pour afficher les contacts sous forme de liste.
- Chaque contact doit être cliquable. Lorsqu'un contact est sélectionné, ouvrez l'activité EditContactActivity.
- Exemple d'interface :

- Extrait du code de la classe ContactsListActivity:

```
...
listViewContacts.setOnItemClickListener(new AdapterView.OnItemClickListener()
{
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id)
    {
        int contactId = contactIds.get(position);
        Intent intent = new Intent( packageContext: ContactsListActivity.this, EditContactActivity.class);
        ...
    }
});
...
```

5. Conception de l'interface utilisateur pour EditContactActivity (activity_edit_contact.xml) :

- Ajoutez deux EditText pour afficher et modifier les informations du contact sélectionné.
- Ajoutez deux boutons : un pour modifier et enregistrer les nouvelles informations et un autre pour supprimer le contact.
- Exemple d'interface :



- Extrait du code de la classe EditContactActivity :

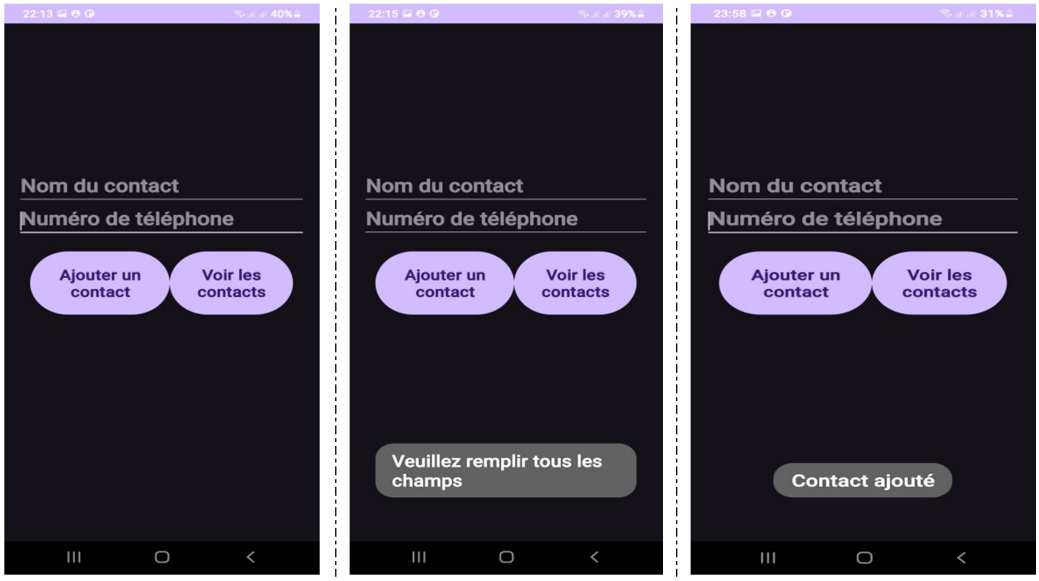
```
...
private void updateContact() {  1 usage
    String name = editTextEditName.getText().toString();
    String number = editTextEditNumber.getText().toString();

    if (!name.isEmpty() && !number.isEmpty()) {
        ...
    } else {
        ...
    }
}

private void deleteContact() {  1 usage
    ...
}
...
```

6. Gestion de la logique Java pour les différentes activités :

- **MainActivity :** Implémentez la logique pour valider les champs, ajouter un contact à la base de données et ouvrir ContactsListActivity.
- **ContactsListActivity :** Affichez tous les contacts dans une liste et ouvrez EditContactActivity lorsque l'utilisateur clique sur un contact.
- **EditContactActivity :** Permet à l'utilisateur de modifier ou de supprimer un contact de la base de données.
- Exemple d'affichage



Extension (optionnel):

- Ajouter une fonctionnalité de recherche pour filtrer les contacts dans ContactsListActivity.
- Ajouter une confirmation avant de supprimer un contact pour éviter les suppressions accidentelles.