Application To-Do List (Base de données SQLite / Room)

1. Introduction

Après avoir développé une application de calculatrice, l'étape suivante consiste à créer une application capable de **conserver les données de manière permanente**. Pour cela, Android met à disposition une base de données locale appelée **SQLite**. Afin de simplifier son utilisation, Google recommande aujourd'hui l'utilisation de la bibliothèque **Room**, qui constitue une couche d'abstraction moderne au-dessus de SQLite.

L'application **To-Do List** (Liste de tâches) est un excellent exemple permettant de découvrir la gestion des bases de données locales. Elle offre à l'utilisateur la possibilité d'ajouter, consulter, modifier et supprimer des tâches, même après la fermeture de l'application.

> **Objectif pédagogique :** Cette application permet d'apprendre à utiliser une base de données locale avec **Room**, à manipuler les opérations CRUD (Créer, Lire, Modifier, Supprimer) et à organiser le code selon les bonnes pratiques du développement Android.

2. SQLite et Room

SQLite est une base de données relationnelle légère intégrée directement au système Android. Toutes les données sont enregistrées dans un fichier local présent sur l'appareil, sans nécessiter de connexion Internet.

Bien que SQLite soit très performant, son utilisation directe demande l'écriture de nombreuses requêtes SQL. Pour simplifier le développement, Google propose **Room Persistence Library**, une bibliothèque officielle qui automatise une grande partie du travail.

Room présente plusieurs avantages :

- Réduction de la quantité de code à écrire.
- Vérification des requêtes SQL à la compilation.
- Meilleure organisation du projet.
- Maintenance plus simple.
- Compatibilité avec l'architecture MVVM.

apcpedagogie.com

> **Bon à savoir :** Room repose entièrement sur SQLite. Il ne remplace pas SQLite mais fournit une interface moderne, plus sûre et plus facile à utiliser.

3. Architecture de l'application

L'application To-Do List est composée de plusieurs éléments :

- **Entity :** représente la table de la base de données.
- **DAO (Data Access Object) :** contient les opérations SQL.
- **Room Database :** gère la connexion avec SQLite.
- **MainActivity :** affiche les tâches et gère les interactions utilisateur.

L'architecture peut être représentée de la manière suivante :

```
Utilisateur
      |
      ▼
MainActivity
      |
      ▼
DAO (CRUD)
      |
      ▼
Room Database
      |
      ▼
SQLite
```

4. Création de l'entité Task

Chaque tâche est représentée par une classe Java annotée avec `@Entity`.

```
@Entity(tableName = "tasks")
public class Task {

    @PrimaryKey(autoGenerate = true)
    private int id;

    @ColumnInfo(name = "titre")
```

apcpedagogie.com

```
    private String titre;

    public Task(String titre){
        this.titre = titre;
    }

    public int getId(){
        return id;
    }

    public void setId(int id){
        this.id = id;
    }

    public String getTitre(){
        return titre;
    }

    public void setTitre(String titre){
        this.titre = titre;
    }
}
```

Analyse :

- `@Entity` indique que cette classe correspond à une table SQLite.
- `@PrimaryKey` définit la clé primaire.
- `autoGenerate = true` permet à Room de générer automatiquement l'identifiant.
- `@ColumnInfo` définit le nom de la colonne dans la base de données.

5. Création du DAO

Le DAO contient toutes les opérations CRUD.

```
@Dao
public interface TaskDao {

    @Insert
    void insert(Task task);

    @Update
    void update(Task task);
```

apcpedagogie.com

```
    @Delete
    void delete(Task task);

    @Query("SELECT * FROM tasks ORDER BY id DESC")
    List<Task> getAllTasks();
}
```

Les annotations utilisées sont :

- **@Insert**: ajoute une nouvelle tâche.
- **@Update**: modifie une tâche existante.
- **@Delete**: supprime une tâche.
- **@Query**: exécute une requête SQL personnalisée.

6. Création de la base de données

Room nécessite une classe héritant de `RoomDatabase`.

```
@Database(entities = {Task.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {

    public abstract TaskDao taskDao();

}
```

Cette classe représente la base de données locale de l'application.

7. Interface utilisateur

L'interface comporte :

- Un champ `EditText` permettant de saisir une nouvelle tâche.
- Un bouton **Ajouter**.
- Une `RecyclerView` affichant toutes les tâches enregistrées.

```
<EditText
    android:id="@+id/editTask"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Nouvelle tâche"/>

<Button
```

apcpedagogie.com

```
    android:id="@+id/btnAjouter"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Ajouter"/>

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

8. Fonctionnement de l'application

Le déroulement de l'application est le suivant :

1. L'utilisateur saisit une tâche.
2. Il clique sur le bouton **Ajouter**.
3. La tâche est enregistrée dans Room.
4. Room sauvegarde automatiquement les données dans SQLite.
5. La RecyclerView recharge les données et affiche immédiatement la nouvelle tâche.
6. Les tâches restent disponibles même après la fermeture de l'application.

9. Les opérations CRUD

Cette application met en œuvre les quatre opérations fondamentales d'une base de données :

Opération	Description
Create	Ajouter une nouvelle tâche.
Read	Afficher toutes les tâches enregistrées.
Update	Modifier une tâche existante.
Delete	Supprimer une tâche.

> **Important :** Les opérations sur la base de données ne doivent pas être exécutées sur le thread principal afin d'éviter le blocage de l'interface utilisateur. Il est recommandé d'utiliser des threads, **Executors**, **LiveData** ou **ViewModel**.

apcpedagogie.com

10. Avantages de Room

- Code plus propre et plus lisible.
- Moins d'erreurs SQL.
- Meilleure organisation du projet.
- Maintenance facilitée.
- Excellentes performances.
- Intégration avec LiveData, ViewModel et MVVM.
- Base de données entièrement locale, sans connexion Internet.

11. Résultat obtenu

À la fin de cette application, l'utilisateur peut :

- Ajouter de nouvelles tâches.
- Afficher toutes les tâches enregistrées.
- Modifier une tâche existante.
- Supprimer une tâche terminée.
- Conserver automatiquement les données après la fermeture de l'application.

> **Conclusion :** L'application **To-Do List** constitue une étape essentielle dans l'apprentissage du développement Android. Elle introduit les concepts fondamentaux de la persistance des données, de l'architecture Room, des opérations CRUD et de la gestion d'une base de données locale, compétences indispensables pour développer des applications professionnelles.

Rédigé et structuré par : **Med Khalil Kribi**

Source : https://apcpedagogie.com/application-to-do-list-android-sqlite · téléchargé le 07/11/2026 · © APC Pédagogie · apcpedagogie.com