

RAPPORT DE TP : APPLICATION

LifeBalance

Développé par : **Soukaina
AOUDANE**

1. INTRODUCTION

Titre : Développement d'une application Android avec SQLite

Projet : LifeBalance - Suivi personnel et organisation quotidienne

Durée : TP 8 - Persistance des données avec SQLite

Contexte : Application mobile de gestion de tâches avec stockage local

2. OBJECTIFS PÉDAGOGIQUES

- Comprendre le rôle d'une base de données locale SQLite
- Créer et configurer une base SQLite dans Android
- Implémenter une couche d'accès aux données (DAO)
- Manipuler les données (CRUD) dans un contexte réel
- Développer une application mobile structurée

3. ARCHITECTURE TECHNIQUE

3.1. Structure du projet

```
com.example.lifebalance/
├── activities/      # Écrans de l'application
├── database/        # Gestion de la base de données
├── models/          # Modèles de données
├── adapters/        # Adaptateurs RecyclerView
└── utils/           # Utilitaires
```

3.2. Technologies utilisées

- **Langage :** Java
- **Base de données :** SQLite (embarquée)
- **UI :** XML layouts, RecyclerView, Material Design
- **Architecture :** MVC avec DAO pattern

4. FONCTIONNALITÉS IMPLÉMENTÉES

4.1. Fonctionnalités principales

Création de tâches avec titre, description, date et statut

Affichage des tâches dans une liste dynamique (RecyclerView)

Modification des tâches existantes

Suppression des tâches avec confirmation

Statuts : À faire / En cours / Terminé

Persistance des données après fermeture de l'app

4.2. Écrans développés

1. **Écran principal** - Navigation vers les fonctionnalités
2. **Ajout de tâche** - Formulaire avec validation
3. **Liste des tâches** - Affichage avec actions
4. **Modification** - Édition des données existantes

5. BASE DE DONNÉES SQLITE

5.1. Structure des tables

```
CREATE TABLE Tache (  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    titre TEXT NOT NULL,  
    description TEXT,  
    date TEXT,  
    statut TEXT  
)
```

5.2. Opérations CRUD

- **Create** : insertTache() - Insertion d'une nouvelle tâche
- **Read** : getAllTaches() - Récupération de toutes les tâches
- **Update** : updateTache() - Modification d'une tâche
- **Delete** : deleteTache() - Suppression d'une tâche

6. COMPOSANTS DÉVELOPPÉS

6.1. Classes Java

- **DatabaseHelper** : Gestion de la création/upgrade de la BD
- **TacheDAO** : Couche d'accès aux données (CRUD operations)
- **Tache** : Modèle de données avec getters/setters
- **TacheAdapter** : Adaptateur pour RecyclerView avec actions
- **MainActivity** : Écran principal de navigation
- **AddTacheActivity** : Formulaire d'ajout
- **ListeTachesActivity** : Affichage de la liste
- **EditTacheActivity** : Modification des tâches

6.2. Layouts XML

- **activity_main.xml** : Interface d'accueil
- **activity_add_tache.xml** : Formulaire d'ajout
- **activity_liste_taches.xml** : Liste des tâches
- **activity_edit_tache.xml** : Formulaire de modification
- **item_tache.xml** : Item de la liste RecyclerView

7. DIFFICULTÉS RENCONTRÉES ET SOLUTIONS

7.1. Problèmes techniques

1. **Crash au lancement** : NullPointerException sur les vues
 - a. **Solution** : Vérification des IDs dans les fichiers XML
2. **Base de données non créée** : Erreurs SQL
 - a. **Solution** : Implémentation correcte de SQLiteOpenHelper
3. **RecyclerView vide** : Données non chargées
 - a. **Solution** : Appel correct du DAO dans onCreate()

7.2. Bonnes pratiques implémentées

- Séparation des responsabilités (DAO pattern)
- Validation des entrées utilisateur
- Messages d'erreur explicites
- Architecture modulaire et maintenable

- Couleurs personnalisées et gradients
- Cartes avec ombres et coins arrondis
- États vides stylisés
- Feedback visuel (Toast messages)
- Confirmation avant suppression
- Navigation fluide entre les écrans
- Validation des formulaires

9. TESTS ET VALIDATION

9.1. Tests fonctionnels

- Ajout d'une tâche avec données valides
- Affichage correct de la liste
- Modification d'une tâche existante
- Suppression avec confirmation
- Persistance après redémarrage

9.2. Tests d'erreur

- Validation des champs obligatoires
- Gestion des listes vides
- Robustesse face aux entrées invalides

10. COMPÉTENCES ACQUISES

10.1. Techniques

- Gestion de SQLite dans Android
- Implémentation du pattern DAO
- Gestion du cycle de vie des Activities
- Manipulation des bases de données relationnelles

10.2. Méthodologiques

- Développement structuré d'application mobile
- Debugging et résolution de problèmes
- Documentation du code
- Tests et validation

11. CONCLUSIONS ET PERSPECTIVES

11.1. Bilan

L'application **LifeBalance** fonctionne correctement et répond aux objectifs pédagogiques. Toutes les fonctionnalités CRUD sont implémentées avec une interface utilisateur intuitive.

11.2. Points forts

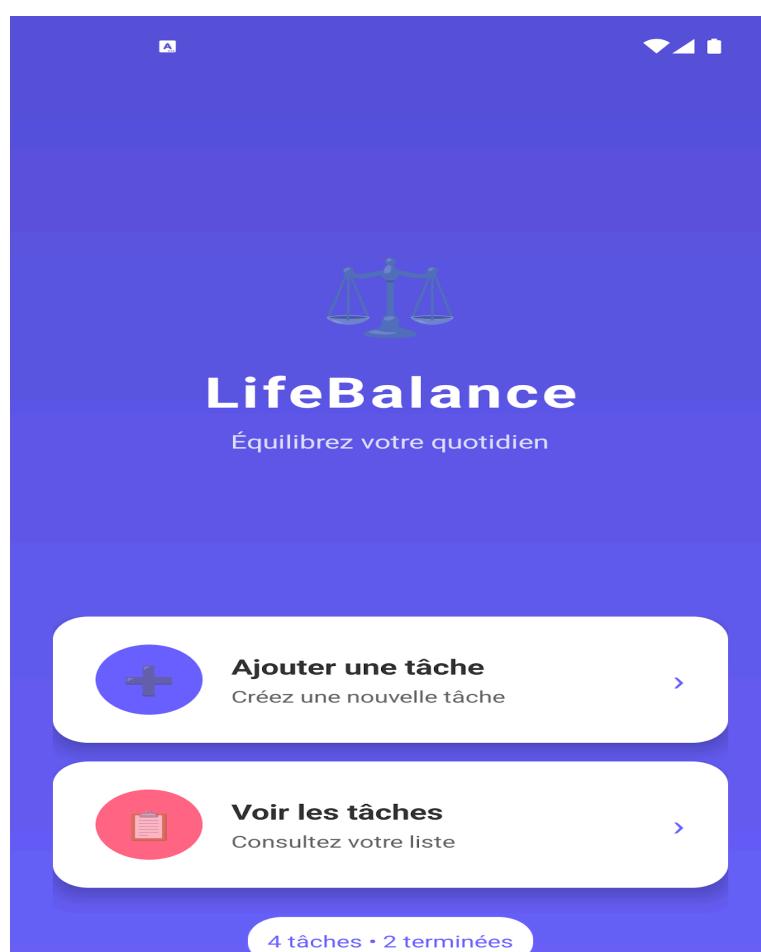
- Architecture propre et maintenable
- Code bien structuré et commenté
- Interface utilisateur moderne

RÉALISÉ PAR : Soukaina AOUDANE

DATE : 03 février 2026

12. Visualisation des interfaces:

1/Interface principale



1/Interface d'ajout des tâches

Nouvelle Tâche

Titre *

Description

Date

Statut

AJOUTER LA TÂCHE

AFFICHER LA LIST DES TACHES

1/Interface liste des tâches



Liste des Tâches

RETOUR

En cours

10/3/2026

Backend

Developper le backend avec node.js

MODIFIER

SUPPRIMER

Terminé

15/11/2025

Maquettes Figma

Trouver les maquettes figma sur les quelles je dios travailler

MODIFIER

SUPPRIMER

Terminé

1/11/2025

Conception

Creer les diagrammer UML

MODIFIER

SUPPRIMER

À faire

10/04/2026

Frontend

Developper Le frontend avec React

MODIFIER

SUPPRIMER

+ AJOUTER UNE TÂCHE

