

MiniTodoApp - 作业提交文档

项目名称: MiniTodoApp

项目类型: Android Mini App - 待办事项管理应用

开发语言: Kotlin

架构模式: MVVM + Room数据库



一、产品功能介绍

1.1 核心功能点

MiniTodoApp是一款专注于待办事项管理的Android应用，以下是目标完成的功能以及完成情况：

✅ 功能1：待办事项管理

- 添加待办:** 用户输入标题，点击添加按钮创建新的待办项
- 完成切换:** 点击复选框标记待办项为已完成/未完成，显示删除线样式
- 删除待办:** 点击删除按钮移除待办项
- 列表展示:** 待办项以列表形式展示，支持无卡顿滚动

实现方式:

- RecyclerView + ListAdapter: 高效列表显示
- DiffUtil: 计算差异，自动更新
- TodoAdapter: 优化ViewHolder重用和监听器清理

✅ 功能2：提醒功能

- 设置提醒时间:** 为待办项设置具体的提醒时间（精确到分钟）
- 系统闹钟:** 到达提醒时间时自动触发系统闹钟
- 系统通知:** 提醒时显示系统通知，点击返回应用

实现方式:

- ReminderDialogFragment: 日期/时间选择对话框
- AlarmManager: 系统级精确闹钟
- BroadcastReceiver: 接收闹钟广播
- NotificationManager: 显示提醒通知

完整流程:

```
用户点击提醒按钮
↓
ReminderDialogFragment (日期/时间选择)
↓
TodoViewModel.updateTodoReminder()
↓
```

```
TodoDao.update() → Room数据库
↓
AlarmUtils.setAlarm() (注册系统闹钟, 提前10分钟)
↓
[到达闹钟时间]
↓
AlarmReceiver.onReceive() (接收系统广播)
↓
NotificationUtils.showReminderNotification() (显示通知)
↓
用户看到通知, 点击打开应用
```

✅ 功能3: 智能排序

- **完成状态优先级:** 未完成优先显示, 已完成排在后面
- **时间排序:** 同一状态内按提醒时间升序排列 (早提醒优先)
- **创建时间排序:** 无提醒时间的按创建时间降序排列 (新的优先)

排序规则:

第一优先级: `isDone` (`false > true`)

└─ 未完成优先于已完成

第二优先级: `remindTime` (时间升序)

└─ 有提醒时间的按时间排序

第三优先级: `createdAt` (时间降序)

└─ 无提醒时间的按创建时间排序

✅ 功能4: 数据统计

- **完成数统计:** 显示已完成的待办项数量
- **未完成数统计:** 显示未完成的待办项数量
- **总数统计:** 显示待办项总数

实现方式:

- ViewModel中的StateFlow实时计算统计数据
- 列表更新时自动重新计算

❌ 功能5: 桌面小部件(Widget)

❌ 功能6: 分类



二、程序概要设计

2.1 应用架构



2.2 数据表设计

待办表 (todo_table)

字段名	类型	说明
id	INTEGER PRIMARY KEY	待办项唯一标识, 自动递增
title	TEXT NOT NULL	待办项标题
isDone	BOOLEAN DEFAULT 0	完成状态 (0=未完成, 1=已完成)
createdAt	LONG	创建时间戳 (毫秒), 默认当前时间
remindTime	TEXT	提醒时间 (格式: yyyy-MM-dd HH:mm, 空值表示无提醒)

索引优化:

- 在isDone上创建索引以加快状态查询

2.3 关键类职责

TodoViewModel (业务逻辑层)

```
// 数据流
val todos: StateFlow<List<TodoEntity>>           // 排序后的待办列表
val completedCount: StateFlow<Int>               // 已完成数
val uncompletedCount: StateFlow<Int>             // 未完成数
val totalCount: StateFlow<Int>                   // 总数
val shouldScrollToTop: StateFlow<Boolean>        // 滚动顶部标志

// 主要方法
fun addTodo(title: String)                      // 添加待办
fun toggleTodo(todo: TodoEntity)                // 切换完成状态
fun updateTodo(todo: TodoEntity)                // 更新待办
fun deleteTodo(todo: TodoEntity)                // 删除待办
fun updateTodoReminder(remindTime: String)       // 设置提醒时间
```

TodoAdapter (列表展示)

- ListAdapter:** 基于DiffUtil的高效列表适配器
- ViewHolder重用:** 清除旧监听器，避免事件混乱
- 性能优化:** 无卡顿滚动

AlarmUtils (系统闹钟)

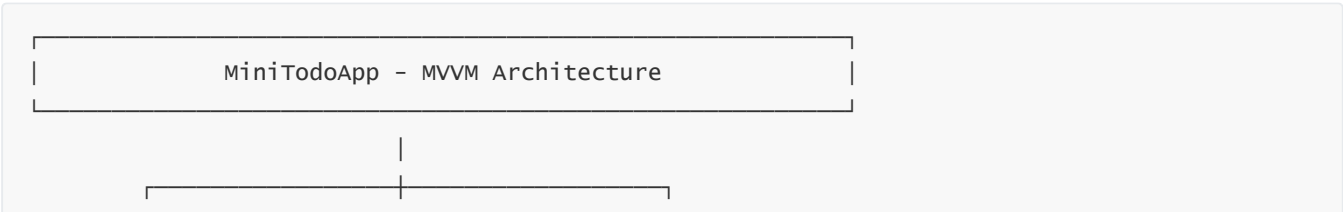
```
fun setAlarm(context, todoId, remindTime)        // 设置闹钟（提前10分钟）
fun removeAlarm(context, todoId)                 // 取消闹钟
fun hasReachedTime(remindTime): Boolean           // 检查时间是否已到达
```

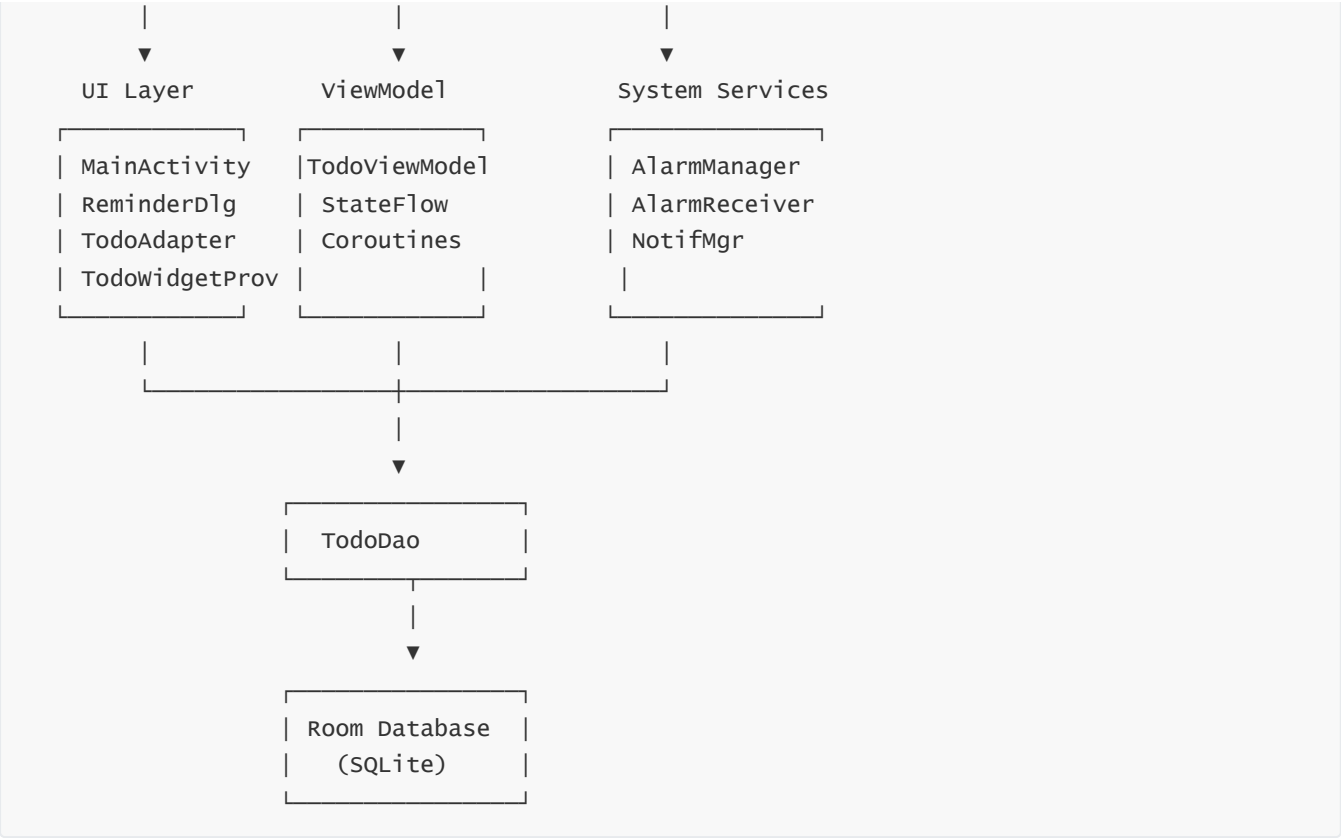
NotificationUtils (系统通知)

```
fun createNotificationChannel(context)            // 创建通知渠道 (Android 8.0+)
fun showReminderNotification(...)                 // 显示待办提醒通知
fun areNotificationsEnabled(context): Boolean     // 检查通知权限 (Android 13+)
```

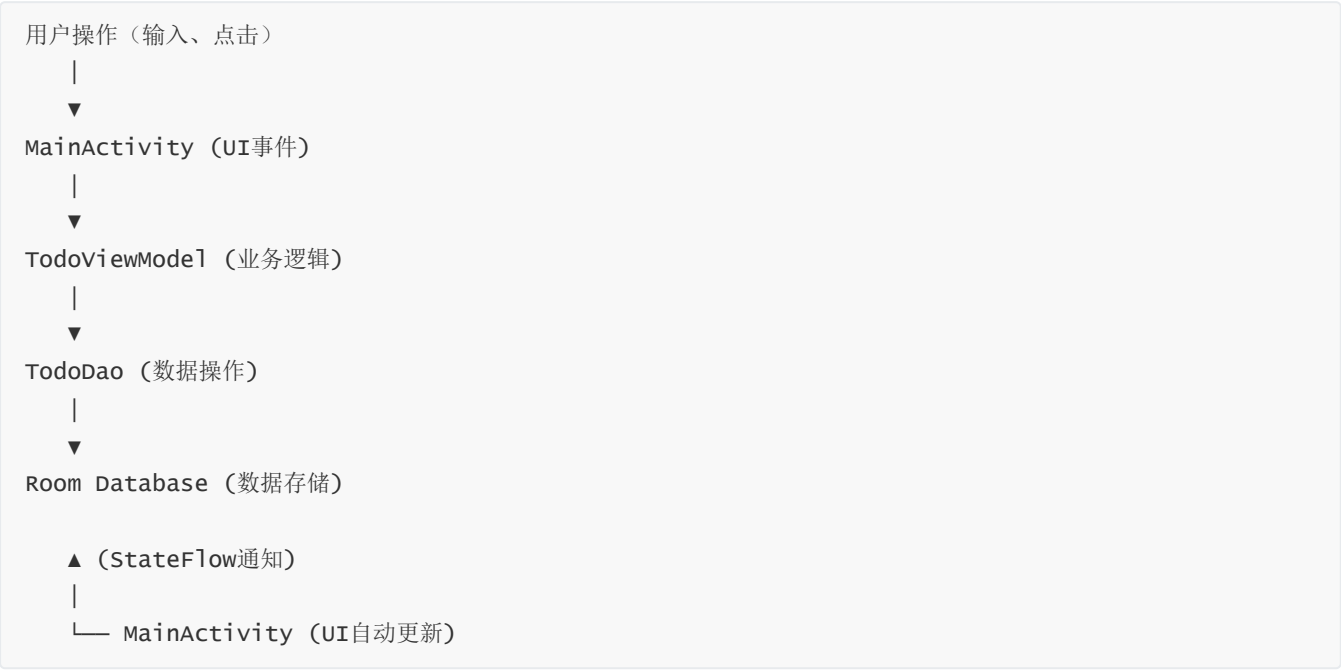
 三、软件架构图

3.1 整体架构

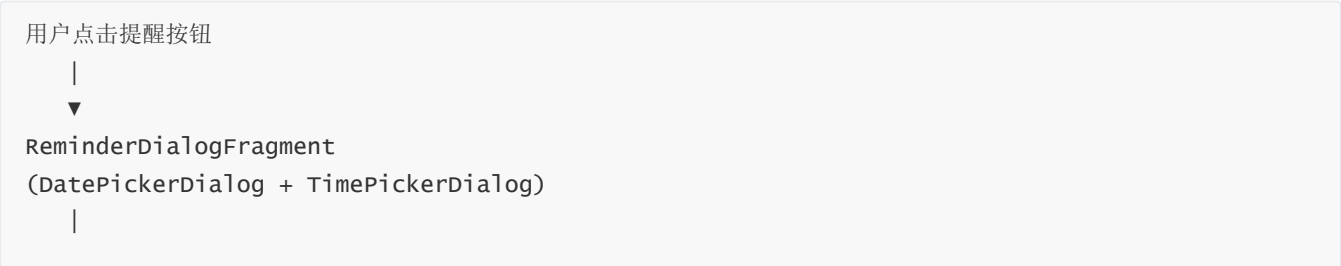


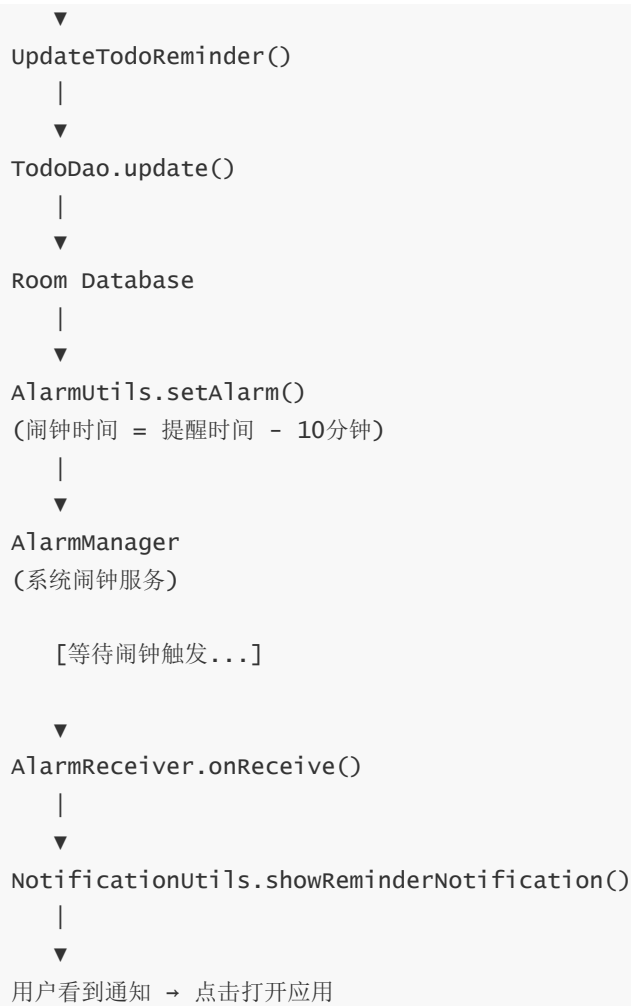


3.2 数据流向



3.3 提醒功能流程





五、开发技术栈总结

编程语言与架构

- Kotlin (现代Android推荐语言)
- MVVM架构 (分离UI和业务逻辑)
- Coroutines + Flow (异步编程)
- StateFlow (响应式状态管理)

数据持久化

- Room ORM (SQLite封装)
- DAO模式 (数据访问接口)
- 数据库版本v3 (含迁移脚本)

UI框架

- RecyclerView + ListAdapter (高效列表)
- DiffUtil (差异计算)
- DialogFragment (对话框)

- AppWidgetProvider (小部件)

系统集成

- AlarmManager (精确闹钟)
- NotificationManager (通知)
- BroadcastReceiver (广播)
- Android 6.0-15.0兼容性处理

性能优化

- 高效列表滚动
- 内存优化 (监听器清理)
- 数据库索引 (查询优化)
- 后台协程 (UI不卡顿)