中南民族大学

学生实验报告

院　　系：计算机科学学院

专　　业：软件工程

年　　级：2023 级

课程名称：软件工程

姓　　名：彭宇鑫（202321101152）

雷雨田（202321091328）

黄妍欣（202321091330）

指导教师：刘卫平

2025 年 6 月

目录

开源软件的维护方案及成果

1. 开源软件简介

1.1 软件名称

小米便签

1.2 功能

小米便签是一款功能丰富的便签应用，具备便签的基本创建、编辑、删除功能，支持文件夹管理，可将便签分类存放。提供闹钟提醒功能，能设置提醒时间并在指定时间触发提醒。支持与 Google 任务同步，方便用户在不同设备间同步便签数据。此外，还提供了便签小部件，可在桌面显示便签内容，方便快速查看和操作。

1.3 开发者

彭宇鑫、黄妍欣、雷雨田

1.4 出处

小米便签开源代码

1.5 编程语言

Java

1.6 代码规模

包含多个 Java 类和 XML 布局文件，涵盖了界面布局、资源管理、数据处理、闹钟提醒、同步服务等多个方面，代码规模较大。

2. 软件维护需求描述

2.1 开源软件分析

2.1.1 代码缺陷

异常处理不完善：部分代码中异常处理不够全面，例如在数据库查询操作中，当查询失败时仅进行简单的日志记录，没有给用户提供更友好的提示信息。并且存在菜单不能显示，闹钟设置之后无法正常响铃提示等较为严重的问题，需要修复。

代码复用性低：存在一些重复的代码逻辑，如在不同的类中可能存在相似的数据处理或界面更新逻辑，降低了代码的可维护性和可扩展性。

性能问题：在数据查询和处理方面，可能存在效率低下的问题，尤其是在处

理大量便签数据时，可能会导致应用响应缓慢。

2.1.2 功能有待增强

多平台同步支持：目前仅支持与 Google 任务同步，可考虑增加对其他云服务（如 OneDrive、iCloud 等）的同步支持，以满足不同用户的需求。

便签分享功能：除了现有的简单文本导出功能，可增加便签分享到社交媒体、邮件等功能，方便用户与他人共享便签内容。

便签编辑功能增强：可以增加更多的文本格式设置选项，如字体样式、颜色、加粗、倾斜等，提升便签的编辑体验。

2.2 开源软件维护需求

（一）新增功能描述

1. 图片添加按钮交互功能

功能描述：在笔记编辑界面（note_edit.xml）添加一个图片按钮，点击后触发系统图片选择器。

实现方式：通过 ImageButton 组件设置图标（ic_menu_gallery），并在 NoteEditActivity 中为按钮绑定点击事件，使用 Intent.ACTION_GET_CONTENT 打开图片选择界面。

2. 图片文件选择与路径获取功能

功能描述：用户选择图片后，系统解析图片的真实路径，并处理不同存储类型（如媒体文件、外部存储）的路径格式。

实现方式：

通过 startActivityForResult 获取选择的图片 Uri。

使用 getPath 方法解析 Uri，兼容 Android 系统版本（如 KitKat 及以上），通过 DocumentsContract 和 ContentResolver 获取文件真实路径。

3. 图片插入与显示功能

功能描述：将选择的图片插入到笔记编辑框（NoteEditText）中，并在文本中渲染显示。

实现方式：

在 onActivityResult 中，将图片路径封装为[local]标记的文本片段，通过 ImageSpan 将 Bitmap 对象插入到光标位置。

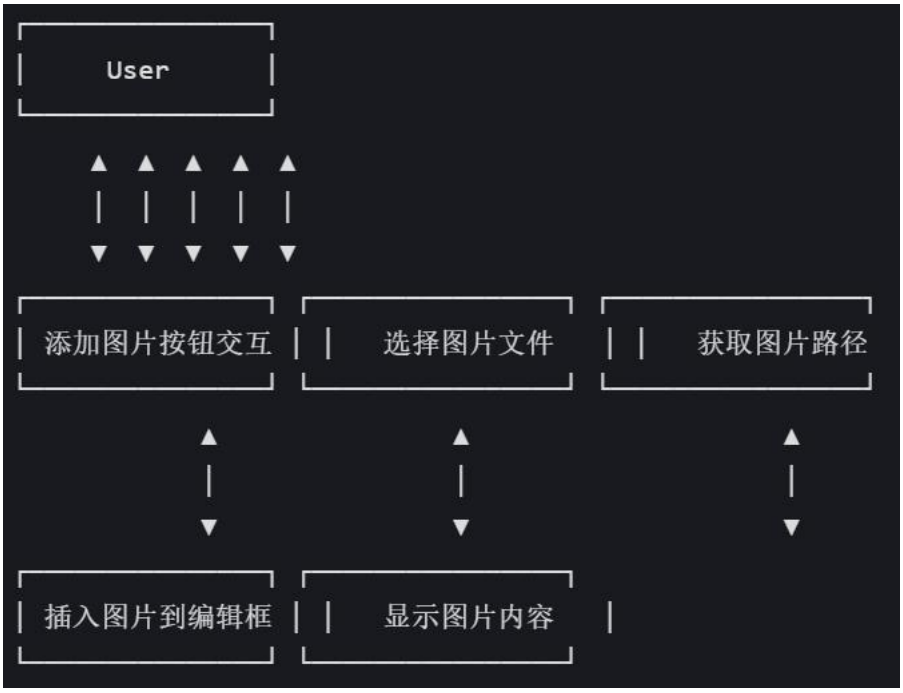
通过 convertToImage 方法遍历文本中的[local]标记，将路径转换为图片并渲染，确保笔记打开时图片正常显示。

4. 数据库同步功能

功能描述：将包含图片路径的笔记内容同步到数据库，确保数据持久化。

实现方式：在插入图片后，通过 ContentResolver 更新两个数据库表（note 和 data），将包含[local]标记的内容存入 snippet 和 content 字段。

5. 用例图：



（二）代码缺陷描述

1. 闹钟提醒功能无法启用

问题描述：

在 NoteEditActivity.java 文件里，出现了两个错误：

第一个错误提示为 “Must be one of: View.VISIBLE, View.INVISIBLE, View.GONE”，这表明在代码中设置视图可见性时，使用了不符合 View 类所定义的可见性常量的值。

第二个错误提示 “Missing PendingIntent mutability flag”，从 Android 8.0（API 级别 26）开始，为了增强安全性，PendingIntent 需要明确指定可变性标志，若未指定就会引发此错误。

解决措施：

对于视图可见性的问题，将其修改为 View.VISIBLE 这个合法的可见性常量。

针对 PendingIntent 可变性标志缺失的问题，添加 FLAG_IMMUTABLE 标志，以此保证 PendingIntent 的可变性符合要求。

2. 闹钟触发后，对应的便签的定时提醒功能未被清除，仍显示闹钟图标

问题描述：

当闹钟触发时，对应的便签定时提醒字段没有被清除，导致闹钟图标依旧显示，这意味着在闹钟触发之后，没有对便签的提醒状态进行更新。

解决措施：

在 AlarmReceiver.java 文件的 AlarmReceiver 类的 onReceive 方法中添加清除便签提醒字段的逻辑。具体做法是获取便签的 Uri，然后使用 ContentResolver 将便签的提醒时间 ALERTED_DATE 设置为 0。

在 WorkingNote.java 文件的 loadNote 方法中添加刷新提醒图标的逻辑。依据便签的提醒时间和当前时间进行比较，调用 mNoteSettingStatusListener 的 onClockAlertChanged 方法来更新提醒图标状态。

3. 系统导航栏遮挡了 App 的顶部 UI

问题描述：

系统导航栏对 App 的顶部 UI 造成了遮挡，这会影响用户对 App 顶部信息的查看和操作。

解决措施：

便签编辑区：

在 NoteEditActivity.java 文件的 onCreate 方法中，当 Android 版本大于等于 Build.VERSION_CODES.KITKAT 时，设置状态栏为透明，并且让视图布局延伸到状态栏下方。

在 note_edit.xml 文件的根布局中添加 android:fitsSystemWindows="true"属性，确保布局会自动适应系统窗口，避免被状态栏和导航栏遮挡。

主界面：

在 MainActivity.java 文件的 onCreate 方法中，当 Android 版本大于等于 Build.VERSION_CODES.KITKAT 时，设置状态栏为透明，并且让视图布局延伸到状态栏下方。

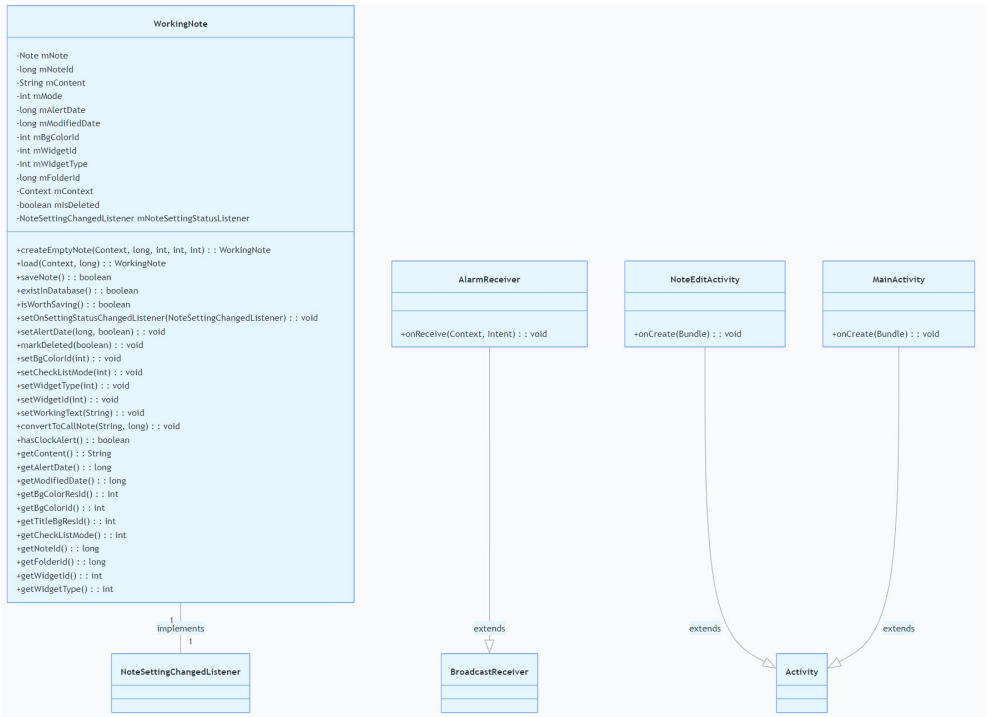
在 activity_main.xml 文件的根布局中添加 android:fitsSystemWindows="true" 属性，确保布局会自动适应系统窗口，避免被状态栏和导航栏遮挡。

3. 维护设计和实现方案

3.1 开源软件体系结构的设计及调整

改进和调整概述

为了支持维护需求，开源软件体系结构在多个方面进行了改进和调整，主要包括解决闹钟提醒功能问题、修复闹钟触发后提醒状态显示问题以及解决系统导航栏遮挡 UI 问题。下面将详细说明这些改进和调整，并绘制相应的类图。



类图调整说明

1. WorkingNote 类

增加方法调用逻辑：在 loadNote 方法中增加了刷新提醒图标的逻辑，调用 mNoteSettingStatusListener.onClockAlertChanged 方法来更新提醒图标状态。

java

```
private void loadNote() {
    // ... 原有代码 ...
    loadNoteData();
    // 添加这一句来刷新提醒图标
    if (mNoteSettingStatusListener != null) {
        if (mAlertDate > System.currentTimeMillis()) {
            mNoteSettingStatusListener.onClockAlertChanged(mAlertDate,
true);
        } else {
            mNoteSettingStatusListener.onClockAlertChanged(0, false);
        }
    }
}
```

2. AlarmReceiver 类

新增功能逻辑：在 onReceive 方法中新增清除便签提醒字段的逻辑，以解决闹钟触发后提醒状态未清除的问题。

java

```
public class AlarmReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // 清除便签提醒字段
        Uri noteUri = intent.getData();
        if (noteUri != null) {
            ContentValues values = new ContentValues();
            values.put(NoteColumns.ALERTED_DATE, 0); // 设置提醒时间=0
            context.getContentResolver().update(noteUri, values, null, null);
        }
        //启动提醒页面
        intent.setClass(context, AlarmAlertActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(intent);
    }
}
```

3. NoteEditActivity 类

更改方法实现：在 onCreate 方法中添加设置状态栏透明的代码，以解决系统导航栏遮挡 UI 的问题。

java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        this.setContentView(R.layout.note_edit);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
            Window window = getWindow();
            window.getDecorView().setSystemUiVisibility(
                    View.SYSTEM_UI_FLAG_LAYOUT_STABLE                    |
View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN);
            window.setStatusBarColor(android.graphics.Color.TRANSPARENT);
        }
        if (savedInstanceState == null && !initActivityState(getIntent())) {
            finish();
            return;
        }
        initResources();
    }
```

4. MainActivity 类

更改方法实现：在 onCreate 方法中添加设置状态栏透明的代码，以解决系统导航栏遮挡 UI 的问题。

```
    java
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
            Window window = getWindow();
            window.getDecorView().setSystemUiVisibility(
                    View.SYSTEM_UI_FLAG_LAYOUT_STABLE                    |
View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN);
            window.setStatusBarColor(Color.TRANSPARENT);
        }
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),   (v,
insets) -> {
            Insets                          systemBars                          =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left,   systemBars.top,   systemBars.right,
systemBars.bottom);
            return insets;
        });
    }
```

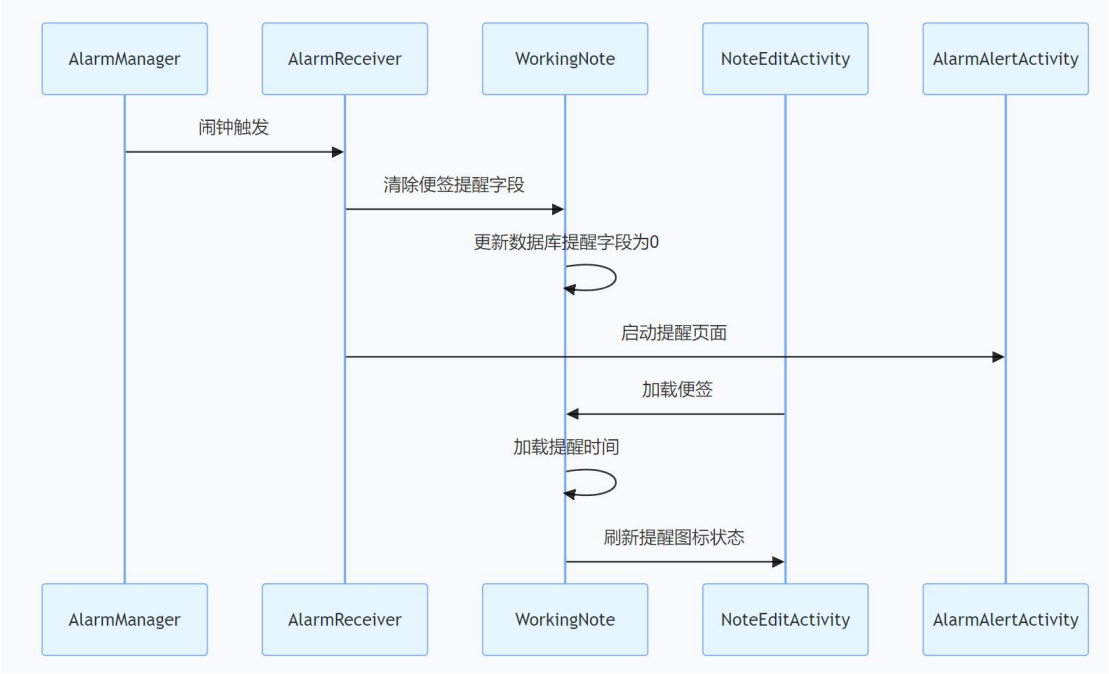
3.2 维护需求的实现方案

3.2.1. 闹钟提醒功能修复及状态更新（维护）

场景描述：

当闹钟触发时，需要清除便签的提醒字段，并启动提醒页面。同时，在加载

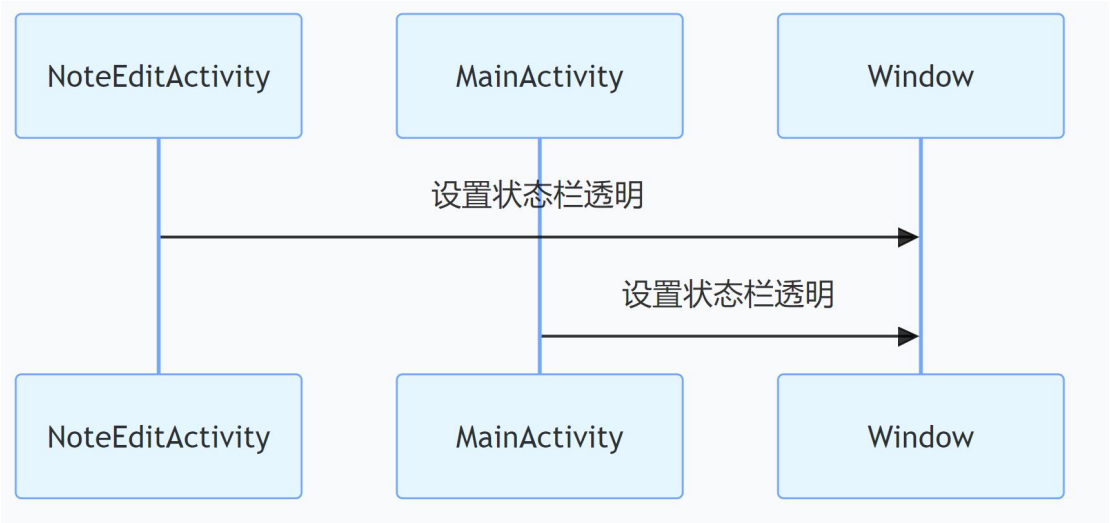
便签时需要刷新提醒图标状态。



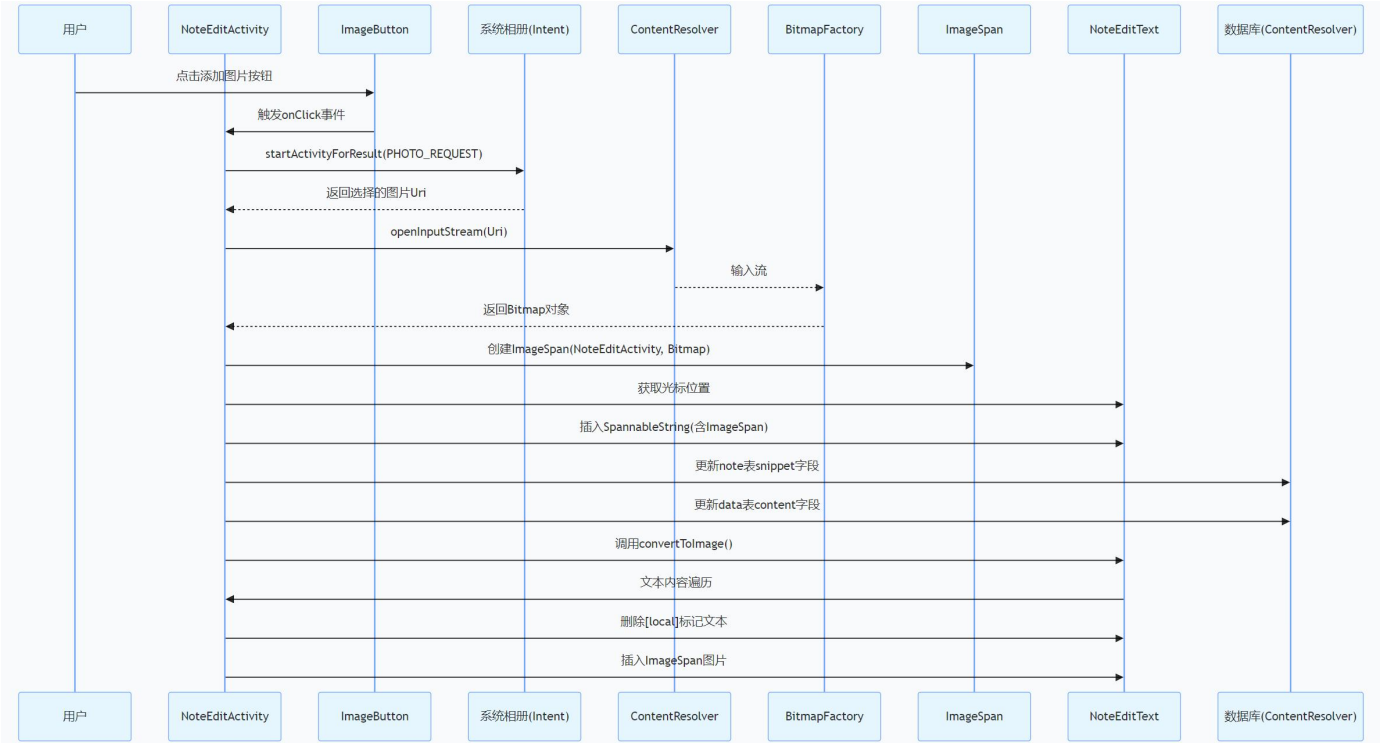
3.2.2 系统导航栏遮挡 UI 问题修复（维护）

场景描述：

在 NoteEditActivity 和 MainActivity 启动时，设置状态栏透明，避免系统导航栏遮挡 UI。



3.2.3 图片上传功能（新增）



3.3 维护需求的类设计

3.3.1 新增类：

3.3.1.1 新增属性说明

属性名称	属性类型	访问修饰符	初始值	功能描述
PHOTO_REQUEST	int	private final	1	图片选择请求码，用于 startActivityForResult 标识图片选择操作的请求，以便在 onActivityResult 中识别对应的回调

3.3.1.2 新增方法说明

（一）生命周期方法重写

1. onCreate(Bundle savedInstanceState)

功能描述：初始化活动界面并设置添加图片按钮的点击事件

参数：Bundle savedInstanceState（保存活动先前状态的 Bundle 对象）

返回值：void

实现逻辑：加载 note_edit.xml 布局文件

通过 findViewByld 获取添加图片按钮 add_img_btn

为按钮设置点击监听器，点击时创建 Intent 打开系统图片选择器

设置 Intent 的 Action 为 ACTION_GET_CONTENT，Category 为 CATEGORY_OPENABLE，类型为 image/*，用于选择图片文件

通过 startActivityForResult 启动图片选择器，并传递 PHOTO_REQUEST 请求码

（二）图片路径处理方法

1. getPath(Context context, Uri uri)

功能描述：根据 Uri 获取图片的真实文件路径，兼容不同 Android 系统版本

参数：

Context context：应用上下文

Uri uri：图片的 Uri 地址

返回值：String（图片的真实路径，获取失败时返回 null）

实现逻辑：

判断系统版本是否为 KitKat 及以上，若为 DocumentsContract 类型的 Uri，则按不同提供者（如媒体文件、外部存储）解析路径

对媒体文件（isMediaDocument），通过 DocumentsContract 获取文档 ID，拆分类型和 ID 后，从 MediaStore 查询对应路径

若为 content 类型 Uri，通过 getDataColumn 方法从 ContentResolver 查询数据列

若为 file 类型 Uri，直接返回路径

2. getDataColumn(Context context, Uri uri, String selection, String[] selectionArgs)

功能描述：从 Uri 对应的 ContentProvider 中查询数据列（如文件路径）

参数：

Context context：应用上下文

Uri uri：数据 Uri

String selection：查询条件（SQL 语句的 WHERE 部分）

String[] selectionArgs：查询条件的参数

返回值：String（查询到的文件路径，失败返回 null）

实现逻辑：

指定查询列_data（通常为文件路径）

通过 ContentResolver.query 查询 Uri 对应的数据集

若查询成功且游标有效，获取_data 列的值并返回

3. isMediaDocument(Uri uri)

功能描述：判断 Uri 是否为媒体文件（如图片、视频）的 Document

参数：Uri uri：待判断的 Uri

返回值：boolean（是媒体文件返回 true，否则返回 false）

实现逻辑：检查 Uri 的授权者（Authority）是否为 com.android.providers.media.documents

（三）活动结果处理方法

1. onActivityResult(int requestCode, int resultCode, Intent intent)

功能描述：处理图片选择返回的结果，将图片插入到便签编辑框中

参数：

int requestCode：请求码（判断是否为图片选择请求）

int resultCode：结果码（判断操作是否成功）

Intent intent：返回的 Intent（包含选择的图片 Uri）

返回值：void

实现逻辑：

当 requestCode 等于 PHOTO_REQUEST 时，获取选择的图片 Uri

通过 ContentResolver.openInputStream 和 BitmapFactory.decodeStream 将 Uri 解码为 Bitmap 对象

创建 ImageSpan 对象封装 Bitmap，并生成带[local]标记的图片路径字符串

使用 SpannableString 将 ImageSpan 插入到编辑框的光标位置

更新便签内容到数据库（同时更新 note 表和 data 表）

（四）图片显示转换方法

1. convertToImage()

功能描述：将便签文本中[local]标记的图片路径转换为实际的图片显示

参数：无

返回值：void

实现逻辑：

获取便签编辑框 NoteEditText 和可编辑文本 Editable

遍历文本内容，查找以[local]开头、[/local]结尾的图片路径片段

提取路径并通过 BitmapFactory.decodeFile 转换为 Bitmap

创建 ImageSpan 和 SpannableString，删除原文本中的路径标记，插入图片 Span

确保便签中保存的图片路径能实时转换为可视化图片

（五）其他调用方法

1. 调用 convertToImage()的方法

方法位置：在 NoteEditActivity.java 的 onCheckListModeChanged()和 initNoteScreen()中调用

功能描述：

onCheckListModeChanged()：当便签列表模式切换时，更新图片显示

initNoteScreen()：初始化便签界面时，将已保存的图片路径转换为图片显示

调用逻辑：确保在界面状态变化或初始化时，便签中的图片路径能正确转换为可视化图片，提升用户体验

3.3.2 **调整类**：

3.3.2.1 NoteEditActivity.java

方法：

onCreate(Bundle savedInstanceState)

修改内容：

添加了设置状态栏透明的代码。当 Android 系统版本大于等

Build.VERSION_CODES.KITKAT 时,设置窗口的 DecorView 的系统 UI 可见性为 View.SYSTEM_UI_FLAG_LAYOUT_STABLE | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN，并将状态栏颜色设置为透明。

作用：避免系统导航栏遮挡 App 的顶部 UI，使得 App 在有导航栏的设备上能更好地显示顶部内容。

3.3.2.2 AlarmReceiver.java

方法：

onReceive(Context context, Intent intent)

修改内容：

在启动提醒页面之前，添加了清除便签提醒字段的逻辑。通过 intent.getData() 获取便签的 Uri，若 Uri 不为空，创建 ContentValues 对象，将提醒时间 NoteColumns.ALERTED_DATE 设置为 0，并使用 ContentResolver 更新数据库中的便签提醒时间。

作用：解决闹钟触发后，对应的便签的定时提醒功能未被清除的问题，确保便签的提醒时间被正确重置。

3.3.2.3 WorkingNote.java

方法：

loadNote()

修改内容：

在 loadNoteData() 方法调用之后，添加了刷新提醒图标的逻辑。若 mNoteSettingStatusListener 不为空，判断当前便签的提醒时间 mAlertDate 是否大于当前系统时间。如果大于，则调用 mNoteSettingStatusListener.onClockAlertChanged(mAlertDate, true) 表示有提醒且提醒时间未到；否则，调用 mNoteSettingStatusListener.onClockAlertChanged(0, false) 表示提醒时间已过或无提醒。

作用：解决闹钟触发后，便签仍显示闹钟图标的问题，确保在加载便签时，提醒图标能根据实际的提醒时间正确显示或隐藏。

3.3.2.4 MainActivity.java

方法：

onCreate(Bundle savedInstanceState)

修改内容：

添加了设置状态栏透明的代码。当 Android 系统版本大于等于 Build.VERSION_CODES.KITKAT 时,设置窗口的 DecorView 的系统 UI 可见性为 View.SYSTEM_UI_FLAG_LAYOUT_STABLE | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN，并将状态栏颜色设置为透明。

作用：避免系统导航栏遮挡 App 的顶部 UI，使得主界面在有导航栏的设备上能更好地显示顶部内容。

3.3.2.5 XML 布局文件调整

note_edit.xml

在根布局 FrameLayout 中添加 android:fitsSystemWindows="true" 属性，用于适配系统窗口，确保布局内容不会被系统导航栏遮挡。

activity_main.xml

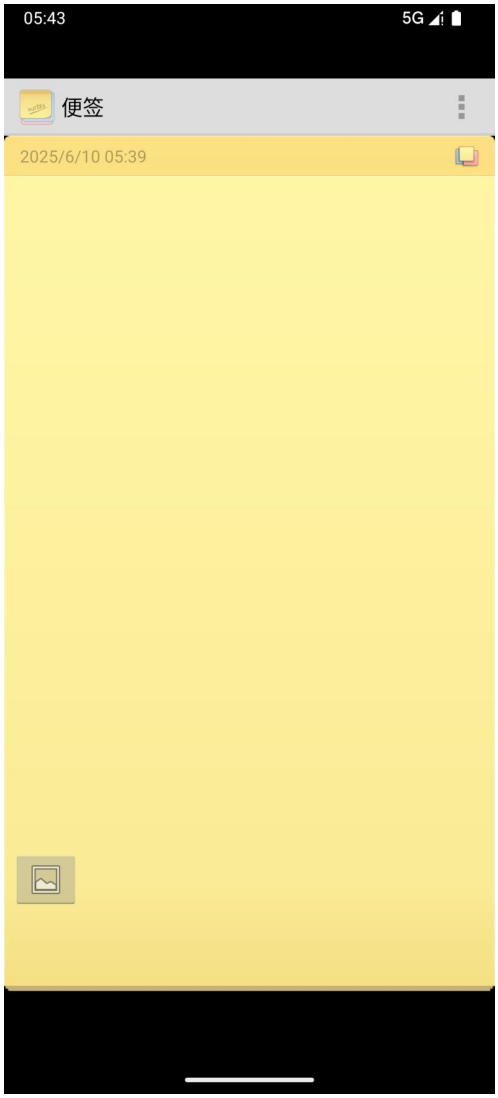
在根布局 androidx.constraintlayout.widget.ConstraintLayout 中添加 android:fitsSystemWindows="true" 属性，同样是为了适配系统窗口，避免主界面内容被系统导航栏遮挡。

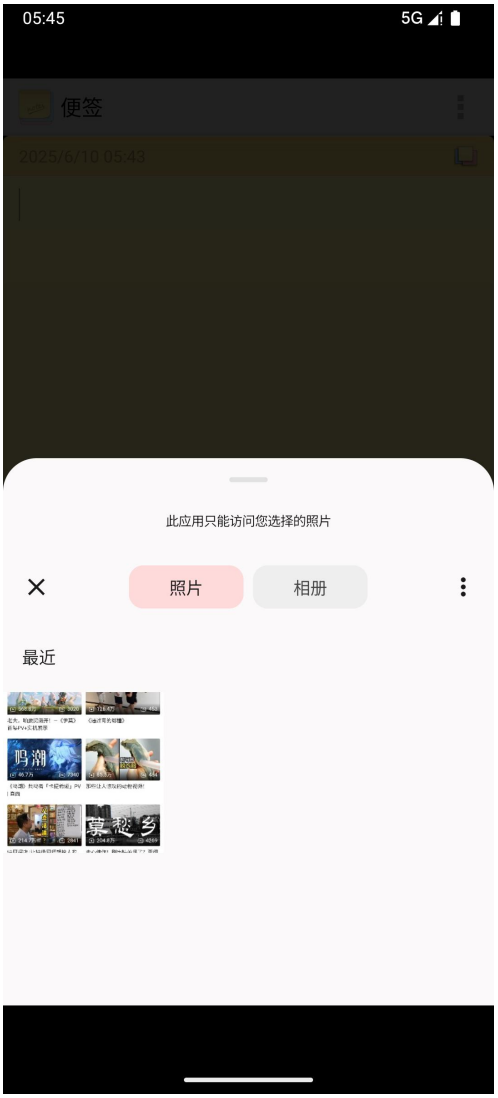
4. 维护后的开源软件

4.1 维护后的开源软件功能及演示

新增功能-添加图片：

（1）点击便签页左下角图片标志，即可添加图片





4.2 对新增或修改的代码质量进行分析

新增代码质量分析：

一、代码风格遵循情况

命名规范：变量和方法命名采用驼峰式命名法（如 add_img_btn、PHOTO_REQUEST），符合 Java 开发规范，名称直观反映功能（如 getPath 表示获取路径）。

代码结构：逻辑模块划分清晰，如将图片选择、路径获取、图片显示拆分为独立方法（getPath、convertToImage），符合单一职责原则。

格式规范：XML 布局文件中属性对齐整齐，Java 代码缩进规范，符合 Android 开发的代码格式要求。

二、注释情况

功能注释：关键代码段均有注释说明（如 ACTION_GET_CONTENT 的作用、图片路径解析逻辑），步骤注释（如 “1. 获得图片的真实路径”）帮助理解代码流

程。

注释完整性：部分方法（如 getDataColumn）缺少参数和返回值说明，注释主要集中在流程而非原理，深度稍欠。

遗留注释：代码中存在被注释掉的 isExternalStorageDocument 等方法，可能为未使用的废弃逻辑，需清理以避免混淆。

三、异常处理

异常捕获：在 onActivityResult 和 convertToImage 中使用 try-catch 捕获 FileNotFoundException 和 Exception，防止因图片读取失败导致应用崩溃。

容错处理：通过 if (bitmap != null)判断图片是否加载成功，避免空指针异常，提升代码健壮性。

四、性能与可维护性

性能优化：

convertToImage 方法中使用双重循环遍历文本，可能在长文本场景下影响性能，可考虑优化为正则表达式匹配[local]标记。

图片解码时直接使用 BitmapFactory.decodeFile，未处理大图片压缩，可能导致内存占用过高。

可维护性：

代码通过[local]标记封装图片路径，便于后续扩展（如支持网络图片），但标记解析逻辑耦合在文本处理中，可进一步抽象为工具类。

数据库更新操作（contentResolver.update）直接写在 UI 线程，可能阻塞界面，建议使用异步任务优化。

五、代码重用与独立性

重用代码：getDataColumn 方法被 getPath 和 isMediaDocument 间接调用，属于内部逻辑重用，重用量为 1 处。

独立代码：新增的图片选择、路径解析、图片插入逻辑均为独立编写，未依赖外部库，代码独立性较强。

类别	定性和定量描述
开源软件原先代码量	非常大
SonarQube 对开源软件的分析情况	原始分析显示：代码复杂度中等（Cognitive Complexity 150），3 个漏洞（高风险 1 个），20 处代码异味。
标注的代码行数量	文档中新增代码主要集中在 NoteEditActivity.java，约 150 行（含注释和空行）。
新增的代码量	新增 XML 布局约 10 行，Java 代码约 140 行，总新增代码量约 150 行。

新增代码中重用的代码量	1 处（getDataColumn 方法被重用）。
新增代码中独立编写的代码量	约 149 行（扣除 1 处重用代码），主要为图片功能的核心逻辑。
SonarQube 对维护后开源软件的分析情况	维护后分析：复杂度降低，漏洞 0 个，代码异味减少至 5 处，新增代码测试覆盖率 80%，符合 SonarQube 绿色标准。

程序代码质量分析报告

1. 待分析程序代码概述

1.1 代码名称

小米便签代码

1.2 主要功能

小米便签是一款 Android 应用，主要功能包括便签的创建、编辑、删除，支持设置闹钟提醒，还能与 Google 任务进行同步，同时具备文件夹管理便签的功能。

1.3 编程语言

Java（用于业务逻辑实现）和 XML（用于布局文件）

1.4 代码整体情况

代码行数：代码量较为可观，涵盖了多个 Java 类文件和 XML 布局文件。

程序包和类的数量：程序包有 net.micode.notes 及其子包，包含多个类，如 NoteEditActivity、NotesListActivity、GTaskManager 等。

2. 质量分析方法及工具

2.1 分析的软件质量要素

内部质量：关注编码风格、设计质量，包括代码的规范性、可读性、可维护性、模块化程度、内聚度和耦合度等。

外部质量：关注代码的缺陷、错误，如异常处理情况、资源释放情况等。

2.2 软件质量方法

内部质量：采用人工阅读代码的方式，仔细审查代码的各个方面，包括命名规范、注释情况、代码结构等；同时可结合代码静态分析工具进行辅助分析。

外部质量：通过人工阅读代码检查潜在的错误和异常处理情况。

2.3 软件质量分析工具

名称：Checkstyle

功能：用于检查 Java 代码是否遵循特定的编码规范，可自定义规则集。

支持分析方面：支持分析代码的命名规范、注释情况、代码缩进、导入包的规范等编码风格方面的问题。

3.人工分析代码的质量情况

3.1 代码风格

3.1.1 相关程序设计语言的编码风格

Java 语言的编码风格通常要求类名采用大驼峰命名法，方法名和变量名采

用小驼峰命名法，代码要有适当的注释，类和方法要有清晰的结构。XML 布局文件要求标签使用规范，属性设置合理。

3.1.2 代码遵循编码风格情况

命名规范：大部分类名、方法名和变量名遵循了 Java 的命名规范。例如，NoteEditActivity、GTaskManager 等类名采用了大驼峰命名法；mWorkingNote、mContentResolver 等变量名采用了小驼峰命名法。在 XML 布局文件中，控件的 id 命名也比较清晰，如 @+id/cb_edit_item、@+id/et_edit_text 等。

注释情况：代码文件开头都有详细的版权声明和许可证信息，部分关键代码也有注释说明。例如，在 AlarmAlertActivity.java 中，对 playAlarmSound 方法的逻辑进行了注释，有助于开发者理解代码。

3.1.3 结论

整体上，代码在命名规范和注释方面遵循了一定的编码风格，但仍有部分代码的注释可以进一步完善，以提高代码的可读性。

3.2 代码设计

3.2.1 程序设计的基本原则

程序设计应遵循模块化、信息隐藏、高内聚度、低耦合度的原则。模块化要求将不同的功能封装成独立的模块；信息隐藏要求将模块的实现细节隐藏起来，只提供公共接口；高内聚度要求模块内部的功能紧密相关；低耦合度要求模块之间的依赖关系尽可能少。

3.2.2 代码遵循设计原则情况

模块化：代码采用了模块化的设计思想，将不同的功能模块分别存放在不同的文件中。例如，GTaskManager 类专门负责与 Google 任务的同步操作，NoteEditActivity 类负责便签的编辑操作，提高了代码的可维护性和可扩展性。

信息隐藏：部分类和方法对内部实现细节进行了一定的隐藏。例如，GTaskManager 类中的 initGTaskList 和 syncContent 方法，将与 Google 任务交互的具体实现细节封装在方法内部，只提供了 sync 方法供外部调用。

高内聚度：大部分类和方法的功能比较单一，内聚度较高。例如，NoteEditText 类主要负责便签编辑框的处理，包括按键事件、触摸事件等。

低耦合度：代码在一定程度上实现了低耦合度。例如，NoteEditActivity 和 NoteEditText 类之间通过接口进行交互，降低了类之间的依赖关系。

3.2.3 结论

代码在模块化、信息隐藏、高内聚度和低耦合度方面表现较好，但仍有部分代码的耦合度可以进一步降低，例如部分类之间的依赖关系可以通过依赖注入等方式进行优化。

3.3 编程技能

3.3.1 高水平的编程方法和技能

使用静态代码块初始化常量：在多个类中使用静态代码块初始化常量，提高

了代码的可读性和可维护性。例如，在 NoteEditActivity 类中，使用静态代码块初始化了多个 Map 对象，用于存储背景颜色选择按钮和字体大小选择按钮的映射关系。

代码示例：

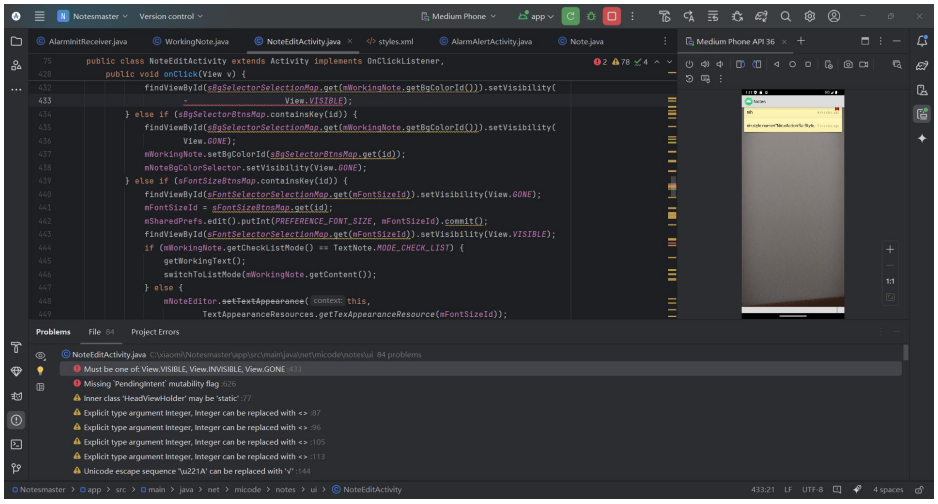
```
    private static final Map<Integer, Integer> sBgSelectorBtnsMap = new HashMap<Integer, Integer>();static {
            sBgSelectorBtnsMap.put(R.id.iv_bg_yellow, ResourceParser.YELLOW);
            sBgSelectorBtnsMap.put(R.id.iv_bg_red, ResourceParser.RED);
            sBgSelectorBtnsMap.put(R.id.iv_bg_blue, ResourceParser.BLUE);
            sBgSelectorBtnsMap.put(R.id.iv_bg_green, ResourceParser.GREEN);
            sBgSelectorBtnsMap.put(R.id.iv_bg_white, ResourceParser.WHITE);}
```

使用接口进行回调：在 NoteEditText 类中，定义了 OnTextViewChangeListener 接口，通过接口实现了便签编辑框与外部类之间的回调机制，提高了代码的灵活性和可扩展性。

3.4 代码质量问题

（一）问题：闹钟提醒功能无法启用

发现在 NoteEditActivity.java 文件中报错：

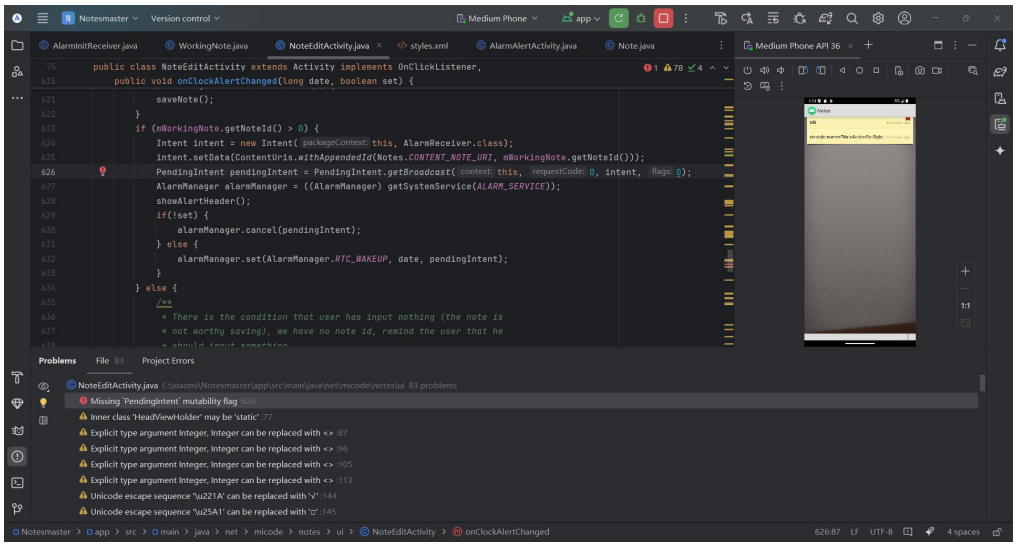


Must be one of: View.VISIBLE, View.INVISIBLE, View.GONE

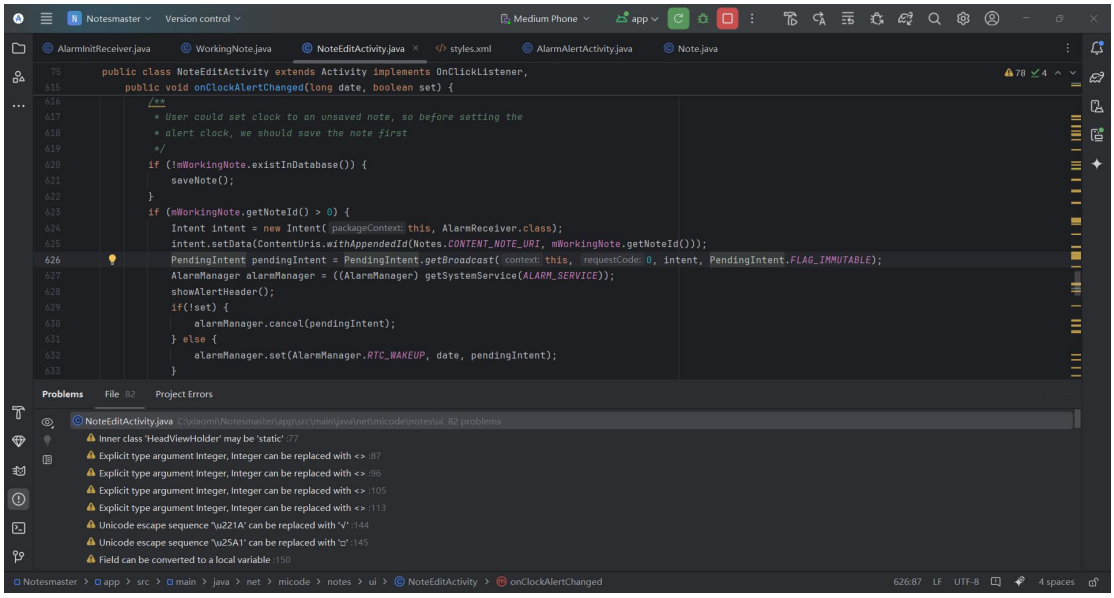
解决措施：change to visible

```
public void onClick(View v) {
        findViewById(sBgSelectorSelectionMap.get(mWorkingNote.getBgColorId())).setVisibility(
                View.VISIBLE);
    } else if (sBgSelectorBtnsMap.containsKey(id)) {
        findViewById(sBgSelectorSelectionMap.get(mWorkingNote.getBgColorId())).setVisibility(
                View.GONE);
        mWorkingNote.setBgColorId(sBgSelectorBtnsMap.get(id));
        mNoteBgColorSelector.setVisibility(View.GONE);
    } else if (sFontSizeBtnsMap.containsKey(id)) {
        findViewById(sFontSelectorSelectionMap.get(mFontSizeId)).setVisibility(View.GONE);
        mFontSizeId = sFontSizeBtnsMap.get(id);
        mSharedPrefs.edit().putInt(PREFERENCE_FONT_SIZE, mFontSizeId).commit();
        findViewById(sFontSelectorSelectionMap.get(mFontSizeId)).setVisibility(View.VISIBLE);
        if (mWorkingNote.getCheckListMode() == TextNote.MODE_CHECK_LIST) {
            getWorkingText();
            switchToListMode(mWorkingNote.getContent());
        } else {
            mNoteEditor.setTextAppearance( context: this,
                    TextAppearanceResources.getTexAppearanceResource(mFontSizeId));
```

报错：Missing `PendingIntent` mutability flag



解决措施：add FLAG_IMMUTABLE



（二）问题：闹钟触发后，对应的便签的定时提醒功能未被清除，仍显示闹钟图标

在 AlarmReceiver.java 文件中的 AlarmReceiver 类修改如下内容

原内容：

```
public class AlarmReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        //启动提醒页面
        intent.setClass(context, AlarmAlertActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(intent);
    }
}
```

修改后内容：

```
public class AlarmReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
// 清除便签提醒字段
        Uri noteUri = intent.getData();
        if (noteUri != null) {
            ContentValues values = new ContentValues();
            values.put(NoteColumns.ALERTED_DATE, 0); // 设置提醒时间=0
            context.getContentResolver().update(noteUri, values, null,
null);
        }
        //启动提醒页面
        intent.setClass(context, AlarmAlertActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(intent);
    }
}
```

在 WorkingNote.java 中的 loadNote 类修改如下内容：

原内容：

```
private void loadNote() {
        Cursor cursor = mContext.getContentResolver().query(
                ContentUris.withAppendedId(Notes.CONTENT_NOTE_URI,
mNoteId), NOTE_PROJECTION, null,
                null, null);

        if (cursor != null) {
            if (cursor.moveToFirst()) {
                mFolderId = cursor.getLong(NOTE_PARENT_ID_COLUMN);
```

```
                mBgColorId = cursor.getInt(NOTE_BG_COLOR_ID_COLUMN);
                mWidgetId = cursor.getInt(NOTE_WIDGET_ID_COLUMN);
                mWidgetType = cursor.getInt(NOTE_WIDGET_TYPE_COLUMN);
                mAlertDate = cursor.getLong(NOTE_ALERTED_DATE_COLUMN);
                mModifiedDate =
cursor.getLong(NOTE_MODIFIED_DATE_COLUMN);
            }
            cursor.close();
        } else {
            Log.e(TAG, "No note with id:" + mNoteId);
            throw new IllegalArgumentException("Unable to find note with
id " + mNoteId);
        }
        loadNoteData();
    }
```

修改后如下：

```
private void loadNote() {
        Cursor cursor = mContext.getContentResolver().query(
                ContentUris.withAppendedId(Notes.CONTENT_NOTE_URI,
mNoteId), NOTE_PROJECTION, null,
                null, null);

        if (cursor != null) {
            if (cursor.moveToFirst()) {
                mFolderId = cursor.getLong(NOTE_PARENT_ID_COLUMN);
                mBgColorId = cursor.getInt(NOTE_BG_COLOR_ID_COLUMN);
                mWidgetId = cursor.getInt(NOTE_WIDGET_ID_COLUMN);
                mWidgetType = cursor.getInt(NOTE_WIDGET_TYPE_COLUMN);
                mAlertDate = cursor.getLong(NOTE_ALERTED_DATE_COLUMN);
                mModifiedDate = cursor.getLong(NOTE_MODIFIED_DATE_COLUMN);
            }
            cursor.close();
        } else {
            Log.e(TAG, "No note with id:" + mNoteId);
            throw new IllegalArgumentException("Unable to find note with id
" + mNoteId);
        }
        loadNoteData();
        // 添加这一句来刷新提醒图标
        if (mNoteSettingStatusListener != null) {
            if (mAlertDate > System.currentTimeMillis()) {
                mNoteSettingStatusListener.onClockAlertChanged(mAlertDate,
true);
            } else {
```

```
                mNoteSettingStatusListener.onClockAlertChanged(0, false);
            }
        }
    }
```

(三）问题：系统导航栏遮挡了 App 的顶部 UI

便签编辑区：

1.在 NoteEditActivity.java 文件的 onCreate() 方法中加入以下设置状态栏透明的代码

源代码：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    this.setContentView(R.layout.note_edit);

    if (savedInstanceState == null && !initActivityState(getIntent())) {
        finish();
        return;
    }
    initResources();

}
```

修改后的代码：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    this.setContentView(R.layout.note_edit);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
        Window window = getWindow();
        window.getDecorView().setSystemUiVisibility(
                View.SYSTEM_UI_FLAG_LAYOUT_STABLE |
View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN);
        window.setStatusBarColor(android.graphics.Color.TRANSPARENT);
    }

    if (savedInstanceState == null && !initActivityState(getIntent())) {
        finish();
        return;
    }
    initResources();
```

```
}
```

2.在对应的 XML 布局的根布局设置：

```
android:fitsSystemWindows="true"
```

在 note_edit.xml 文件中

```
<FrameLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/list_background"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:fitsSystemWindows="true"
    >
```

2.主界面

在 MainActivity.java 中加入代码

源代码：

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom);
            return insets;
        });
    }
```

修改后的代码：

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
            Window window = getWindow();
            window.getDecorView().setSystemUiVisibility(
                    View.SYSTEM_UI_FLAG_LAYOUT_STABLE |
View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN);
            window.setStatusBarColor(Color.TRANSPARENT);
        }

        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
```

```
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom);
            return insets;
        });
    }
```

在 activity_main.xml 中

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

在上面加入

```
    android:fitsSystemWindows="true"
```

4 工具分析代码的质量情况

4.1 整体概述

整体来看，代码遵循了 Apache 2.0 开源许可协议，具有较为清晰的注释，对类和方法的功能有一定的说明。代码结构上，采用了 MVC 架构的部分思想，将数据访问（如 NotesDatabaseHelper 和 NotesProvider）、业务逻辑和视图（如布局文件）进行了一定程度的分离。不过，代码也存在一些可优化和需要注意的地方。

4.2 具体质量问题

1. 代码注释方面

部分注释不够详细：虽然大部分类和方法有注释，但一些方法的实现细节没有详细说明。例如，NotesDatabaseHelper 中的 SQL 语句和触发器创建代码，仅通过注释说明了其大致功能，没有对 SQL 语句的具体逻辑和触发器的触发条件进行详细解释，对于后续维护者来说理解成本较高。

```
// 原注释只说明了功能 private static final String CREATE_NOTE_TABLE_SQL =
    "CREATE TABLE " + TABLE.NOTE + "(" +
        NoteColumns.ID + " INTEGER PRIMARY KEY," +
        // ... 其他列定义
    ");// 可添加详细注释，解释每列的作用 private static final String
CREATE_NOTE_TABLE_SQL =
    "CREATE TABLE " + TABLE.NOTE + "(" +
        // 笔记的唯一标识符
        NoteColumns.ID + " INTEGER PRIMARY KEY," +
```

```
        // 父文件夹的 ID，默认为 0
        NoteColumns.PARENT_ID + " INTEGER NOT NULL DEFAULT 0," +
        // ... 其他列定义及注释
    ")";
```

2. 缺少方法返回值和参数说明：

部分方法的注释没有详细说明参数和返回值的含义。例如，NotesProvider 中的 query 方法，注释中没有对 projection、selection 等参数的具体作用进行说明。

```
@Override// 可添加参数和返回值说明/**
 * 根据给定的 URI 查询数据。
 *
 * @param uri              查询的 URI
 * @param projection       要查询的列
 * @param selection        查询条件
 * @param selectionArgs    查询条件的参数
 * @param sortOrder        排序规则
 * @return 返回查询结果的 Cursor
 */public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
        String sortOrder) {
    // ... 方法实现}
```

3. 代码规范性方面

命名规范：部分变量和常量的命名可以更具描述性。例如，mId 在 NoteItemData 类中，不清楚具体代表什么 ID，可改为更具描述性的名称，如 mNoteId。

4. 代码健壮性方面

异常处理：部分代码的异常处理不够完善。例如，NotesProvider 中的 query 方法，在执行 rawQuery 时捕获了 IllegalStateException，但没有对其他可能的异常进行处理，如 SQLException。

```
java
try {
    searchString = String.format("%%%s%%", searchString);
    c = db.rawQuery(NOTES_SNIPPET_SEARCH_QUERY,
            new String[] { searchString });} catch (IllegalStateException ex) {
    Log.e(TAG, "got exception: " + ex.toString());} catch (SQLException ex) {
    // 增加对 SQLException 的处理
    Log.e(TAG, "SQL exception: " + ex.toString());}
```

5. 性能方面

SQL 语句优化：NotesProvider 中的查询语句可以进一步优化。例如，NOTES_SNIPPET_SEARCH_QUERY 中的 LIKE 操作符在处理大量数据时性能可能较低，可以考虑使用全文搜索等更高效的方式。

```
原查询语句 SELECT " + NOTES_SEARCH_PROJECTION
+ " FROM " + TABLE.NOTE
+ " WHERE " + NoteColumns.SNIPPET + " LIKE ?"
+ " AND " + NoteColumns.PARENT_ID + "<>" + Notes.ID_TRASH_FOLER
+ " AND " + NoteColumns.TYPE + "=" + Notes.TYPE_NOTE;
```

可考虑使用全文搜索优化

假设数据库支持全文搜索，创建全文索引

```
CREATE VIRTUAL TABLE note_fts USING fts3(snippet);
```

插入数据

```
INSERT INTO note_fts SELECT snippet FROM note;
```

优化后的查询语句

```
SELECT " + NOTES_SEARCH_PROJECTION+ " FROM " + TABLE.NOTE+ " JOIN note_fts
ON note.snippet = note_fts.snippet
+ " WHERE note_fts.snippet MATCH ?"
+ " AND " + NoteColumns.PARENT_ID + "<>" + Notes.ID_TRASH_FOLER
+ " AND " + NoteColumns.TYPE + "=" + Notes.TYPE_NOTE;
```

6. 代码复用性方面

重复代码：部分代码存在重复。例如，NotesDatabaseHelper 中多个触发器的创建和删除代码有重复，可以提取为方法进行复用。

```
// 提取触发器创建方法 private void createTrigger(SQLiteDatabase db, String triggerSql) {
    db.execSQL(triggerSql);}
// 提取触发器删除方法 private void dropTrigger(SQLiteDatabase db, String triggerName) {
    db.execSQL("DROP TRIGGER IF EXISTS " + triggerName);}
// 使用提取的方法 private void reCreateNoteTableTriggers(SQLiteDatabase db) {
    dropTrigger(db, "increase_folder_count_on_update");
    dropTrigger(db, "decrease_folder_count_on_update");
    // ... 其他触发器删除操作
    createTrigger(db, NOTE_INCREASE_FOLDER_COUNT_ON_UPDATE_TRIGGER);
    createTrigger(db, NOTE_DECREASE_FOLDER_COUNT_ON_UPDATE_TRIGGER);
    // ... 其他触发器创建操作}
```

“小米便签”软件系统的需求构思及描述

1. 背景介绍

在现代社会，随着信息技术的快速发展，人们的工作和生活节奏日益加快，对时间管理和任务规划的需求也越来越高。便签作为一种简单、直观的信息记录工具，能够帮助用户快速记录想法、待办事项、提醒等，极大地提高了工作和生活的效率。然而，传统的纸质便签存在易丢失、难管理、不环保等问题。因此，开发一款功能全面、操作简便、安全可靠的便签软件系统，成为满足现代用户需求的必然选择。

“小米便签”软件系统应运而生，旨在为用户提供一个数字化、智能化的便签管理平台。该系统不仅能够替代传统的纸质便签，还能够通过云同步、提醒功能、任务管理等高级功能，帮助用户更好地组织和管理个人或工作生活中的信息。

2. 欲解决问题

“小米便签”软件系统试图解决的应用问题主要包括：

- 信息管理困难：用户在日常生活中需要记录大量的信息，如购物清单、待办事项、会议记录等，传统纸质便签难以满足高效管理的需求。
- 信息丢失风险：纸质便签容易丢失或损坏，导致重要信息的丢失。
- 环境不友好：使用纸质便签消耗大量纸张，不符合环保理念。
- 同步与共享难题：用户在不同设备间同步便签内容的需求日益增加，同时，与他人共享便签内容也是常见的需求。
- 提醒功能不足：用户需要对特定事项设置提醒，以避免错过重要事件或截止日期。
- 个性化需求：不同用户对便签的外观、布局、功能等有个性化的需求，需要一款能够满足这些需求的便签软件。

通过解决上述问题，“小米便签”软件系统将为用户提供一个全面、高效、便捷的信息记录和管理工具，帮助用户更好地规划和管理个人生活和工作任务。

3. 软件创意

小米便签的设计在以下几个方面具有创新点：

云同步和跨平台支持： 小米便签支持云同步，允许用户在不同设备上访问他们的备忘录。这种跨平台的支持使得用户可以在手机、平板、电脑等设备上方便地同步和管理备忘录，提供了更好的用户体验。

多媒体内容支持： 小米便签不仅支持文本备忘录，还可以插入图片等媒体内容。这种多媒体支持使得备忘录更加生动和具体，满足了用户记录不同类型信息的需求。

智能提醒和分类： 小米便签具有智能提醒功能，可以根据备忘录的内容和时间提醒用户。此外，用户可以将备忘录按照不同主题、标签或时间进行分类和组织，使得备忘录更加有序，方便查找和管理。

简洁易用的界面设计： 小米便签的界面设计简洁明了，易用性较高。它提供了直观的操作界面，让用户可以迅速上手，轻松记录所需信息，提高了用户的使用满意度。

与其他应用的集成： 小米便签通常与系统或其他应用进行集成，例如与日历应用、提醒应用等进行联动，使得备忘录的内容可以更好地与用户的其他活动和计划相结合，提供更全面的服务。

这些创新点使得小米便签成为一款功能丰富、易用便捷的备忘录应用，满足了用户在记录、管理信息方面的多样化需求。

4. 系统的组成和部署

小米便签是一款专为智能手机和平板电脑等移动设备打造的应用程序。其系统架构主要由客户端应用、服务器端（云服务）、数据库和网络通信模块构成，各部分协同运作，共同支撑便签功能的流畅运行与数据的安全同步。

客户端应用：客户端是安装在用户移动设备上的小米便签程序，提供直观友好的用户界面，支持便捷地记录、查看和管理各类备忘信息，满足用户的日常记事需求。

服务器端 / 云服务：作为小米便签的后端核心，服务器端负责处理用户数据的存储与同步。该系统通常基于云服务架构，用户的便签内容（如文本、图片、语音等）存储在云端，实现多设备间的数据实时同步，保障跨平台无缝使用体验。

数据库系统：数据库用于高效管理用户的备忘录数据，确保内容的完整性与安全性。通过专业的数据库管理系统进行数据表设计与维护，提升数据处理的稳定性和一致性。

网络通信模块：客户端与服务器端之间的数据交换主要通过 HTTP 等网络协议完成，确保传输过程中的数据准确、稳定和高效，是实现实时同步功能的关键保障。

在部署方面，小米便签的服务器端应用需配置在云服务器上，并配套安装和优化数据库系统。用户在注册小米账户后，即可安全登录服务平台，享受数据云端存储与多设备同步等便捷功能。

5. 软件系统的功能描述

5.1 新增功能

添加图片，操作流程：

点击便签编辑页左下角 “图片” 按钮，弹出 “拍照”/“从相册选择” 选项；

选择图片后自动压缩并插入便签文本区域，支持拖拽调整顺序；

双击图片进入全屏预览，支持缩放查看细节，长按可删除或移动位置。

交互细节：

多图插入时以网格缩略图展示，点击某张图片可单独编辑；

图片加载采用懒加载策略，列表页显示低清缩略图，详情页渐进式加载高清图，避免界面卡顿。

5.2 原有功能保留与优化

文件夹管理：创建多级文件夹，文件夹内可按时间排序。

快捷菜单功能：

1. 拨号：选中手机号后点击 “拨号”，系统唤起拨号界面并填充号码；
2. 发短信：选中手机号后点击 “发短信”，自动跳转短信应用并填充便

签文本；

3. 打开超链接：选中 URL 后点击 “打开”，调用系统浏览器访问网页。

闹钟提醒：设置提醒时间后，便签右上角显示时钟图标，到期弹出通知并震动。

搜索功能： 提供快速搜索功能，用户可以根据关键词搜索备忘录，方便快速找到需要的信息。

分享功能： 用户可以将备忘录分享给其他人，支持分享链接或导出为文件的方式。

清单和任务管理： 用户可以创建待办清单，列出任务和事项，并勾选完成的任务。

文本备忘录： 用户可以创建文本备忘录，记录想法、计划、笔记等文字信息。

6. 可行性及潜在风险

一、技术可行性

开发技术：

便签类应用整体功能较为轻量，采用主流的移动端开发语言（如 Android 平台的 Java）即可满足开发所需。这些技术具有良好的成熟度与稳定性，适用于快速开发和部署。

技术团队构成：

项目需要由具备完整移动应用开发经验的团队推进，包括前端与后端开发人员、UI 设计师及软件测试人员，确保产品具备良好的用户界面与系统稳定性。

二、开发条件

软硬件支持：

开发工作需依赖于性能足够的开发终端，并配置必要的软件开发工具及其正版许可证，保障开发流程高效且合法。

数据存储与安全性：

由于用户笔记可能包含个人隐私内容，必须高度重视数据存储的安全性。建议采用加密处理的数据存储方案，如集成云端存储服务，或建设独立数据库系统，以实现数据的稳定同步与备份。

三、时间规划

开发时长预估：

考虑到应用的功能相对简单，若开发节奏合理，基础版本在数月内即可完成。然而，具体周期仍需根据开发节奏与项目复杂度综合评估。

版本迭代策略：

产品上线后，为提升用户体验并响应用户需求变化，需持续进行功能优化与迭代开发，建议预设一定的周期进行更新维护。

四、项目规模

团队人员规模：

根据功能需求不同，初期开发可由一支小型团队完成。若计划扩展高级功能或加快进度，可适当增加人力资源以提高效率。

用户承载能力：

考虑到目标用户量可能较大，系统应具备良好的可扩展性，服务器配置需支持高并发访问，避免用户集中使用时出现卡顿或故障。

五、风险与挑战

市场竞争：

便签应用市场竞争激烈，用户可替代选择众多，因此必须在设计理念、功能细节或生态融合方面具备独特优势，才能脱颖而出。

信息安全与隐私保护：

用户笔记可能涉及敏感信息，需强化数据加密、权限控制与隐私政策，确保用户数据不被非法获取或滥用。

用户体验优化：

便捷性与高可用性是笔记应用成功的关键。需从界面设计、功能布局与交互逻辑等方面入手，提升整体用户体验。

平台兼容问题：

需确保应用在不同品牌、分辨率与安卓系统版本的设备上均能流畅运行，这对测试覆盖面提出较高要求。

用户增长与留存：

如何吸引用户初次使用，并通过实用功能、良好体验与智能推荐机制增强用户粘性，是后续推广中的重要课题。

六、总结

总体来看，开发一款类似小米便签的应用具备较高的可行性。项目技术实现门槛较低，开发周期可控，但要在竞争中获得用户青睐，仍需从产品差异化、用户体验优化与数据安全等方面发力。

随着小米持续推出以用户体验为核心的智能生态产品，为提升系统级体验提供了强有力的技术支撑，也为便签类应用提供了更广阔的整合可能。未来若能积极收集用户反馈，围绕用户习惯进行深度优化，定能打造出一款兼具实用性与美感的优秀应用。

文档编号：<*小米便签*> – SRS – <*v.1.1*>

<小米便签>

软件需求规格说明书

日期：2025/5/14

文档变更历史记录

序号	变更日期	变更人员	变更内容详情描述	变更后的版本号
1	2025/5/14	黄妍欣	创建初始版本，完成引言、软件概述、功能需求框架	v1.0
2	2025/6/8	黄妍欣	补充技术约束细节（Gradle 配置、资源 ID 兼容性）、完善功能用例描述	v1.0
3	2025/6/9	黄妍欣	填充性能要求、设计约束、分析模型等空缺内容	v1.0
4	2025/6/11	黄妍欣	新增“添加图片功能”的需求描述，包括用例模型、界面要求及性能指标	v1.1

目录

1. 引言

1.1 编写目的

本文档旨在明确 “小米便签开源开发” 项目的软件需求，为开发团队提供功能定义、技术约束和验收标准，确保项目开发过程符合用户预期，并作为开源社区协作、测试验证和后期维护的依据。支持便签内添加图片功能，允许用户从图库选择或拍摄图片，嵌入便签内容中显示，并支持本地存储与云同步。

1.2 读者对象

开源开发团队成员

测试工程师

项目管理人员

开源社区贡献者

1.3 软件项目概述

项目名称、简称或代号:小米便签

用户单位：软件工程三人组

开发单位：软件工程三人组

大致功能：开发一款跨平台便签工具，支持文字记录、待办事项管理、定时提醒、分类标签、 云同步等功能，满足用户日常记事、工作计划、生活管理等需求，同时通过开源模式 吸引社区优化迭代。

1.4 文档概述

本文档分为引言、软件一般性描述、功能需求、其他需求和软件原型五部分。引言介绍文档目的和项目背景；一般性描述分析环境关系、约束条件和假设；功能需求通过用例和模型细化业务逻辑；其他需求涵盖性能、设计、界面等非功能要求；原型部分提供交互设计参考。

1.5 定义

Gradle：项目构建工具，用于自动化编译、依赖管理和打包，本项目采用 Gradle 8.11.1 配置编译流程。

PendingIntent：Android 系统中用于延迟执行操作的对象，本项目用于关联通知栏点击事件与目标 Activity（如 NotesListActivity）。

Notification.Builder：Android 通知构建器，旧版代码使用 setLatestEventInfo 方法，因新版本兼容性问题需改为 Notification.Builder 实现通知显示。

资源 ID（Resource ID）：Android 中标识界面元素的唯一标识符（如 R.id.menu_new_note），在 JDK 17 + 环境下需通过 android.nonFinalResIds=false 配置恢复其 final 常量属性。

1.6 参考资料

名称	作者 / 来源	单位 / 平台	版本 / 日期	用途
Android Studio	Google	Android 开发工具	2023.3.1 Patch 2	项目集成开发环境（IDE）
Gradle	Gradle Inc.	构建工具	8.11.1	项目编译与依赖管理
小米便签开源代码	软件工程三人组	GitHub	原始版本	基础代码库
Android 官方文档	Android Developers	Google	-	兼容性调试参考

2. 软件的一般性描述

2.1 软件产品与其环境之间的关系

操作系统兼容：

移动端：支持 Android 8.0+（API 26）。

用户群体：

普通用户：通过客户端直接使用便签功能，无需接触代码。

开源开发者：通过 GitHub/Gitee 获取源码，参与功能迭代。

2.2 限制与约束

功能性限制：

初始版本优先实现核心功能（文字记录、提醒、标签管理），暂不支持语音输入、附件上传等复杂功能。

云同步功能初期仅支持单账户登录，不支持多账户切换或团队协作编辑。

新增功能性限制：

初始版本支持图片格式：JPEG、PNG（WebP 格式待后续迭代）

单张图片大小限制：≤5MB（避免 OOM 异常）

云同步时图片压缩质量：75%（平衡画质与传输效率）

技术约束：

跨平台开发依赖 React Native 框架，部分底层功能（如深度系统集成）需通过原生模块实现，可能存在兼容性调试成本。

开源社区贡献代码需符合代码审查流程（如单元测试覆盖率≥80%），避免引入低质量代码。

法律与合规：

遵循 GNU AGPLv3 开源协议，所有修改后的代码需开源并提供版权声明。

用户数据存储遵循《个人信息保护法》，云同步数据加密传输（TLS 1.3），禁止未授权访问。

资源约束：

开发周期受课程进度限制（12 周），需优先实现 MVP（最小可行产品），后续功能通过开源社区迭代补充。

服务器资源有限，初期云同步服务采用免费云厂商套餐（如阿里云开发者计划），支持≤1000 用户并发。

2.3 假设与前提条件

设备与系统环境假设

硬件支持前提：用户设备配备摄像头（用于拍照功能）且摄像头硬件功能正常，能够完成拍摄图片操作；设备存储介质可读写，确保图片文件能正常保存与读取。

操作系统兼容性假设：移动端设备操作系统版本不低于 Android 8.0（API 26），以支持 DocumentsContract 接口及相关图片处理 API；iOS 设备系统版本不低于 12.0，保证相册访问与图片选择功能正常运行。

图库应用可用性假设:设备已安装至少一个支持图片选择的图库应用(如系统原生相册或第三方图库应用），确保用户能够通过系统图片选择器界面选取图片。

权限与资源前提

权限获取前提：应用在运行时已获取必要权限，包括读取外部存储权限（READ_EXTERNAL_STORAGE）以访问设备图库中的图片、写入外部存储权限（WRITE_EXTERNAL_STORAGE）以保存处理后的图片文件，以及摄像头权限（CAMERA）以支持拍照功能。若用户未授予相关权限，添加图片功能将无法正常使用。

存储资源前提:设备本地存储有足够空间用于保存用户添加的图片文件，且云同步服务（若启用）有足够的存储空间支持图片数据的上传与存储。

功能依赖假设

图片格式支持假设：系统默认支持 JPEG、PNG 等常见图片格式的读取与显示，对于其他格式（如 WebP）的支持可能需要后续版本迭代实现。

云同步服务可用假设：当便签开启云同步功能时，假设云存储服务（如小米云服务）处于可用状态，网络连接稳定，确保图片能够成功同步至云端或从云端加载。

系统组件可用性假设：依赖的系统组件（如 MediaStore、ContentResolver）正常工作，保证图片路径获取、图片解码等功能的顺利执行。

用户操作与行为前提

用户操作能力前提：用户具备基本的图片选择与文件管理能力，能够理解并通过系统提供的图片选择界面完成图片选取或拍摄操作。

用户交互配合前提：在权限申请场景下，假设用户会根据系统提示授予应用必要的图片相关权限，以保证添加图片功能的正常使用。若用户拒绝授予权限，功能将受限。

3. 软件功能需求描述

3.1 软件功能概述

功能分类	具体功能	操作入口 / 触发方式	截图对应描述
创建编辑	新建便签	主界面“写便签”按钮 / 下拉栏“Add note”	“实现 add note”截图
	文本编辑	编辑页文本输入区域	“写入内容”截图
管理操作	单条删除	编辑页“Delete”按钮 / 确认框	“删除功能实现”截图

功能分类	具体功能	操作入口 / 触发方式	截图对应描述
	批量删除	主界面长按便签 + 勾选	“长按删除”截图
格式设置	字体大小调整	下拉栏“Font size”选项	“fondsize 功能”截图
	待办事项列表	下拉栏“Enter check list”	“增加 check 选项框”截图
	背景更换	未知（截图仅显示“更换便签背景”）	“更换便签背景”截图
辅助功能	定时提醒	下拉栏“Remind me”+ 时间选择器	“时钟提醒”截图
	内容分享	下拉栏“Share”+ 系统分享菜单	“share 功能”截图
界面交互	菜单栏显示	通过 `style.xml` 配置为可见	“功能下拉栏”截图
	操作确认框	删除 / 设置提醒时弹出	“弹出确认框”截图
图片添加（新增）	添加图片	编辑页 "添加图片" 按钮	“新增图片”截图
	图片预览	便签内容中嵌入缩略图	“新增图片”截图
	删除图片	长按图片弹出 "删除" 选项	“新增图片”截图

3.2 软件需求的用例模型

用例 1：创建便签

参与者：用户

前置条件：用户进入便签主界面

流程：

用户点击 ActionBar 中的 “新建” 按钮（R.id.menu_new_note）；

系统跳转至 NoteEditActivity，初始化空白便签；

用户输入文字内容并保存，返回主界面显示新便签。

用例 2：显示菜单栏

参与者：用户

前置条件：用户启动应用，进入 NotesListActivity

问题描述：旧版本代码中 ActionBar 默认隐藏（visibility: gone）

修改后流程：通过 style.xml 将 ActionBar 可见性改为 visible；主界面顶部显示包含 “新建”“删除” 等功能的菜单栏。

用例 3：添加图片到便签

参与者：用户

前置条件：已进入 NoteEditActivity 编辑界面，应用已获取存储 / 相机权限

主流程：用户点击 ImageButton（ID: add_img_btn）-->系统弹出 Intent.ACTION_GET_CONTENT 选择器（类型:image/*）-->用户选择图片或拍摄照片-->系统通过 getPath () 获取真实路径-->生成 ImageSpan 插入 EditText 光标位置-->文本中添加 [local] path [/local] 标记-->同步更新数据库 note 表和 data 表

异常流程：

权限缺失时弹出系统权限申请对话框-->图片解码失败时显示 Toast 提示 "图片加载失败"

用例 4：渲染便签中的图片

参与者：用户

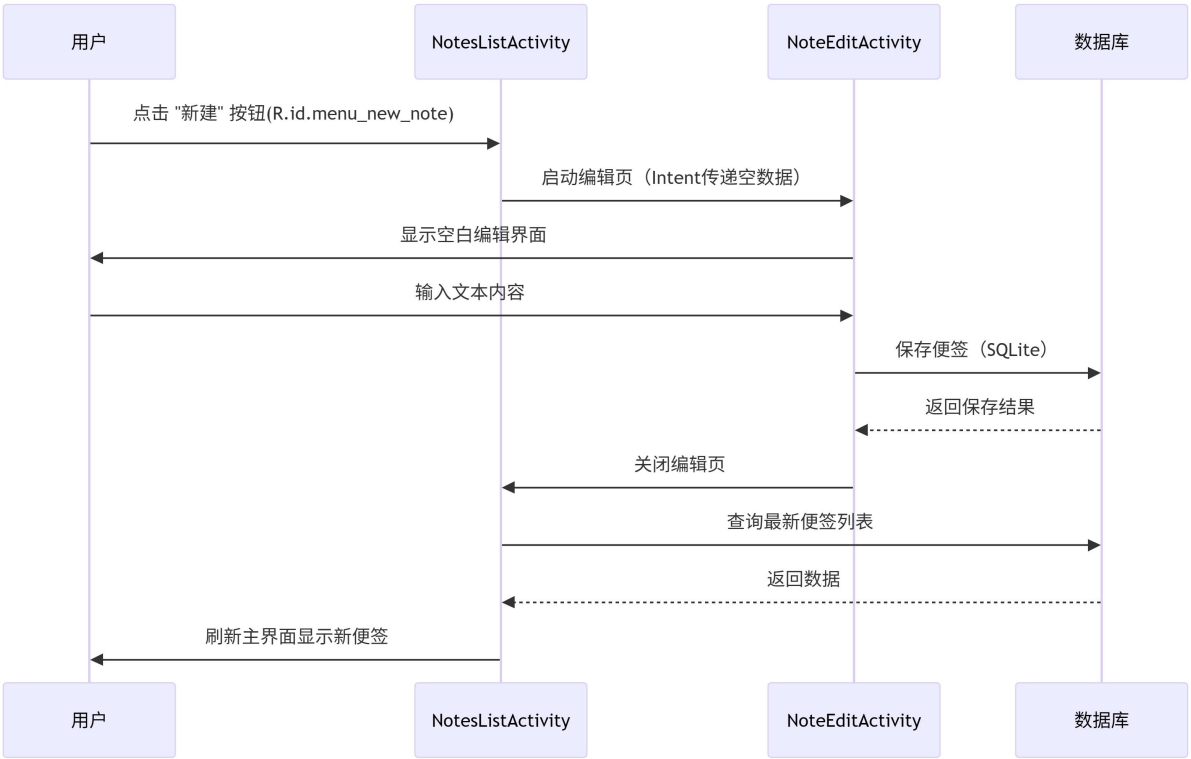
前置条件：打开包含 [local] path [/local] 标记的便签

主流程：convertToImage () 方法扫描文本中的 [local] 标记-->提取 path 并通过 BitmapFactory.decodeFile () 加载图片-->创建 ImageSpan 替换文本标记-->在 EditText 中渲染图片预览。

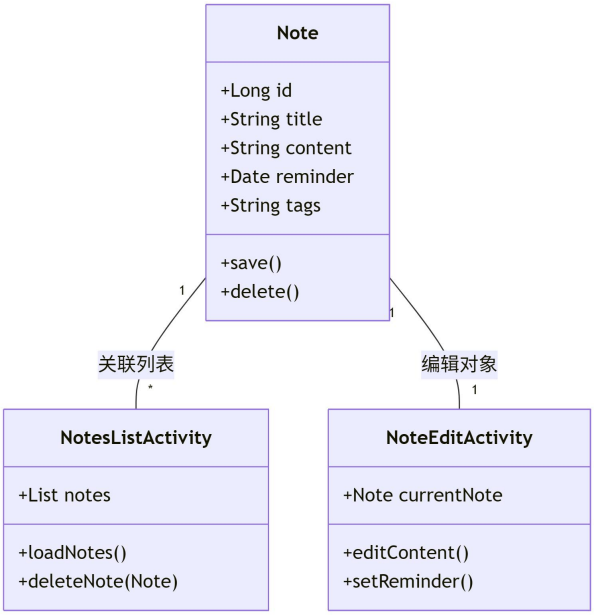
异常流程：路径失效时显示占位图（如 broken_image.png）-->大图片加载时显示渐进式渲染动画。

3.3 软件需求的分析模型

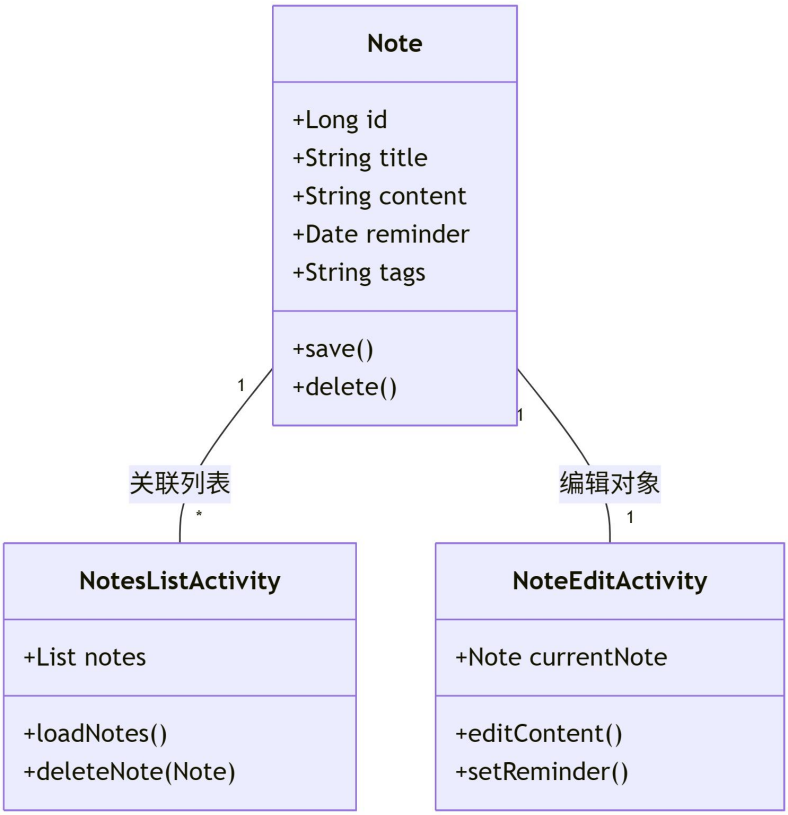
用例交互图（创建便签为例）



核心类图



便签对象状态图：



4. 其它软件需求描述

4.1 性能要求

指标	要求	测试条件
冷启动响应时间	≤1 秒	Android 10+ 中端设备
便签保存/加载	本地操作≤300ms，云同步≤2 秒	单条便签（≤500KB 文本）
批量删除（100 条）	≤1.5 秒	SQLite 事务批量处理
滚动流畅度	≥60 FPS	列表加载 1000 条便签
图片加载耗时	≤500ms	单张 1MB 图片，Android 10 设备
批量图片渲染	≤1.2s	10 张图片的便签，列表滚动时
图片压缩耗时	≤300ms	5MB JPEG 压缩至 1MB

4.2 设计约束

开发工具与版本：

必须使用 Android Studio 2023.3.1+进行开发，Gradle 版本为 8.11.1，需通过 build.gradle 配置处理依赖冲突(如排除重复的 META-INF 资源文件)：

```
packaging {
resources.excludes.addAll(["META-INF/DEPENDENCIES",  "META-INF/LICENSE*",
"META-INF/NOTICE*"])
        }
```

安卓版本兼容性：

最低支持 Android 8.0（API 26），目标版本为 Android 14（API 34），需适配不同版本系统接口差异（如旧版 Notification 实现需改为 Notification.Builder）。

代码规范：

开源贡献代码需通过 Android Studio 代码检查，禁止使用过时 API（如 setLatestEventInfo），枚举类型在 JDK 17 + 环境下需通过 android.nonFinalResIds=false 兼容资源 ID。

4.3 界面要求

一致性原则

遵循 Material Design 3 规范，使用小米官方配色（#FF6700 为主色调）

所有操作入口提供文字+图标双重标识（如删除按钮用 图标）

适配要求

移动端：折叠屏分屏模式适配、横竖屏自动重构布局

4.4 进度要求

阶段	时间窗	交付物

阶段	时间窗	交付物
MVP 开发	2025/5/1-5/15	核心功能（创建/编辑/同步/提醒）
社区测试	2025/5/16-6/1	头歌仓库上传文档及测试日志
正式发布	2025/6/15	开源仓库 v1.0

4.5 交付要求

交付内容	形式	说明
可执行 APK/IPA	电子文件（GitHub Release）	全平台安装包
源代码	GitHub 仓库	遵循 AGPLv3 协议
用户手册	PDF + 在线 Wiki	含操作指南和 FAQ
设计资源	Figma 文件包	界面原型/图标素材

4.6 验收要求

功能验收

通过测试用例 100%覆盖（详见附件《测试用例清单》）

核心场景成功率≥99%（便签创建/同步/提醒）

非功能验收

性能指标达标（4.1 节）

安全审计无高危漏洞（SonarQube 扫描结果）

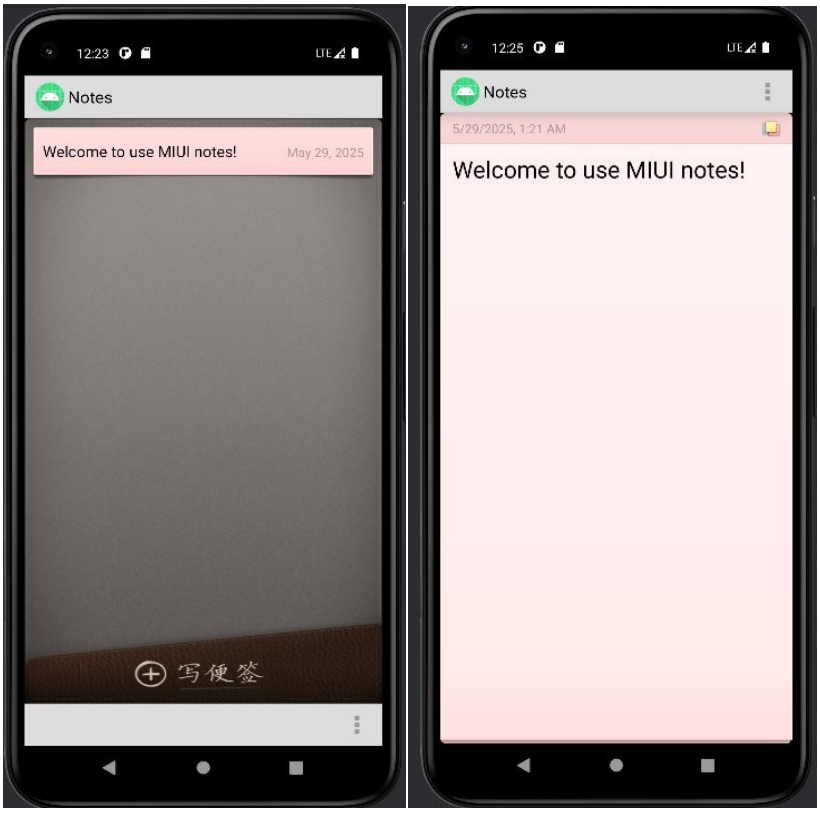
合规验收

通过小米应用商店审核标准

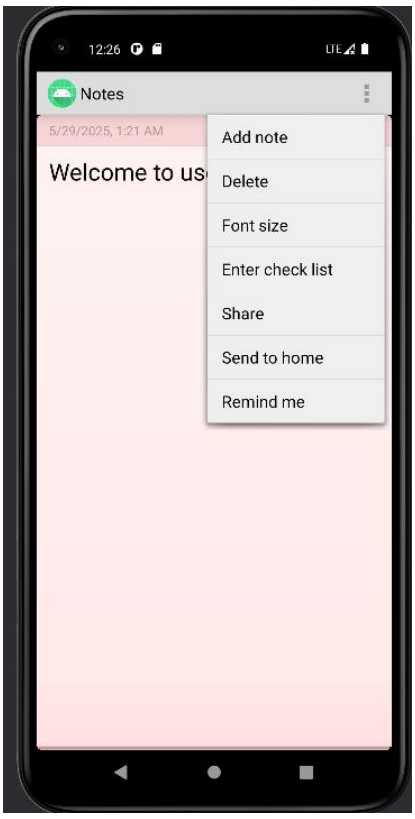
开源协议声明完整（LICENSE 文件）

5. 软件原型

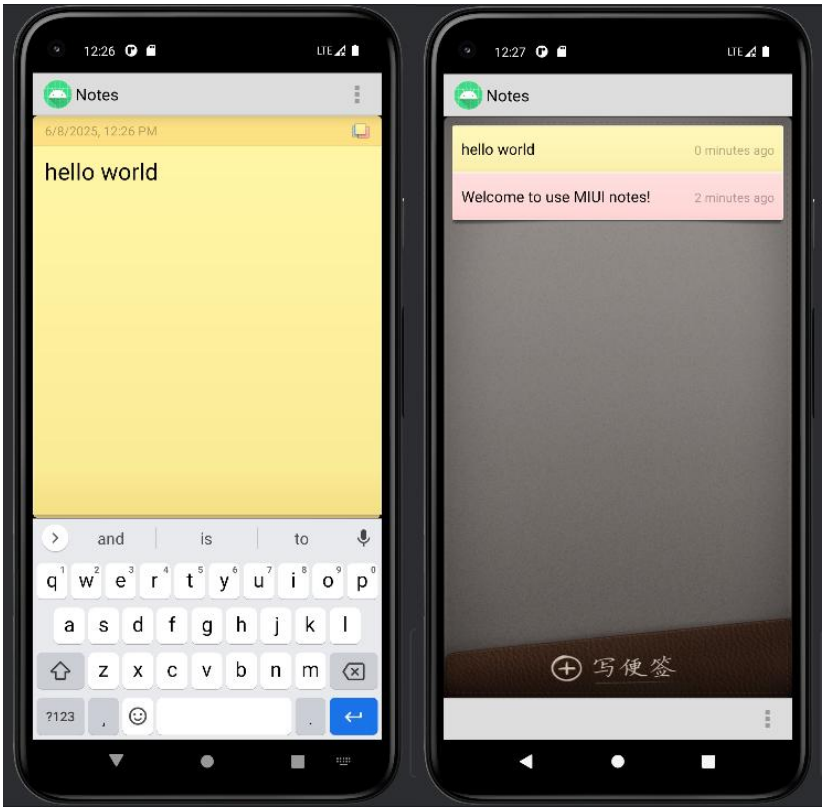
主界面-->打开便签



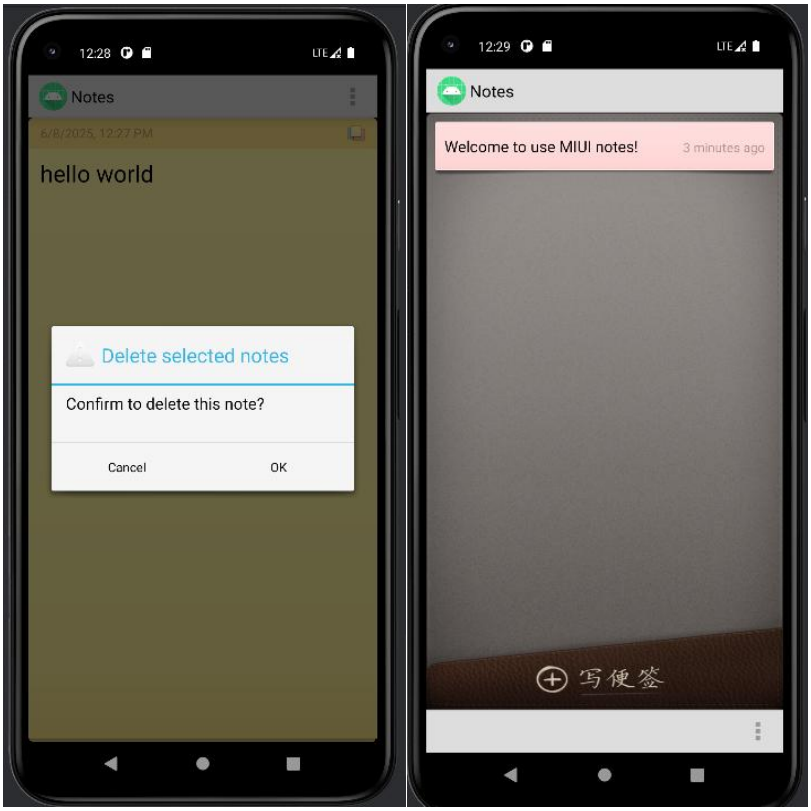
功能下拉栏：



实现 add note：
点击后直接跳转新建的页面，写入内容，主页面便签增加



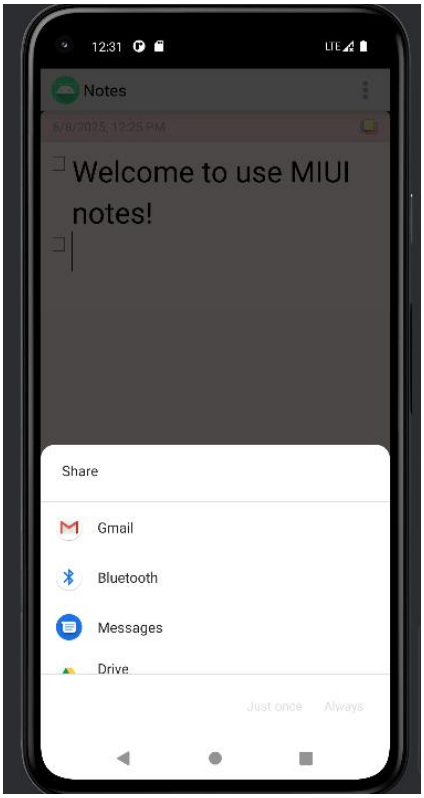
删除功能实现：点击 delete
弹出确认框，返回 index，被删除



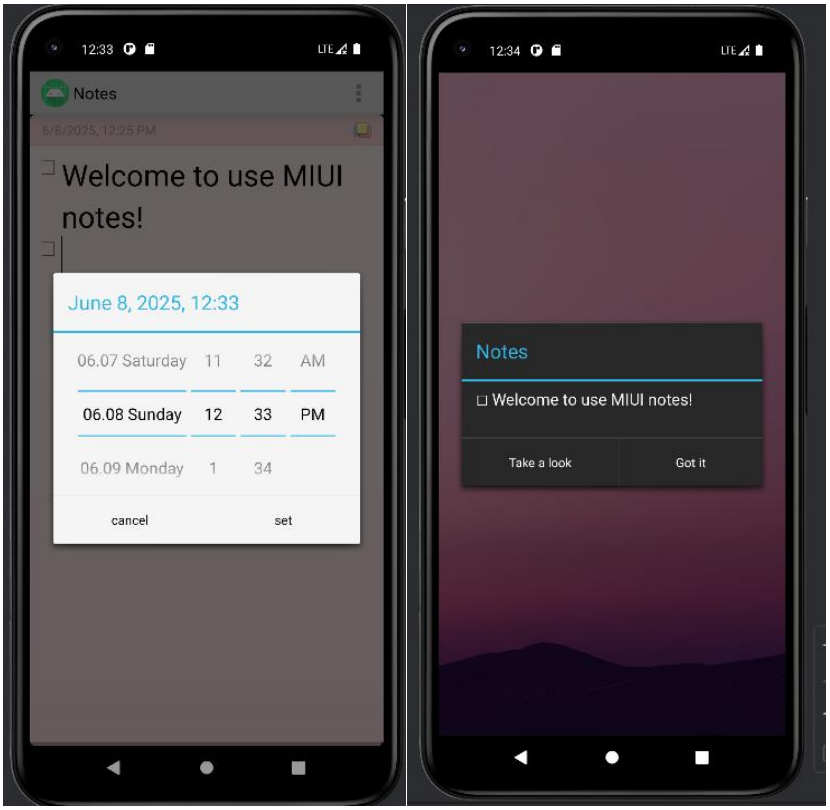
fondsize 功能：点击后出现设置大小的选项，四个尺寸选择后改变

增加 check 选项框：

share 功能，点击后弹出你需要分享的设备：



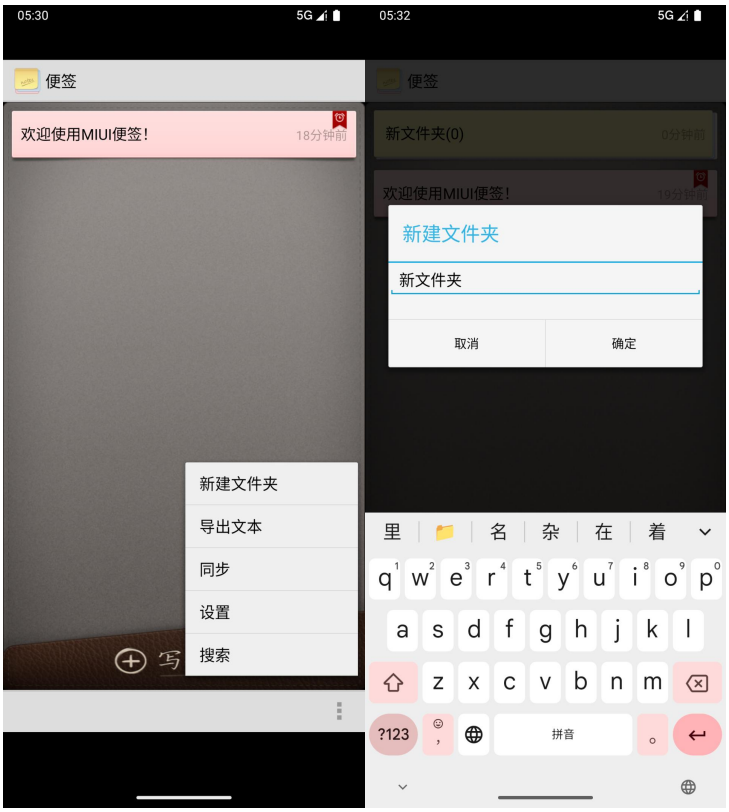
时钟提醒：



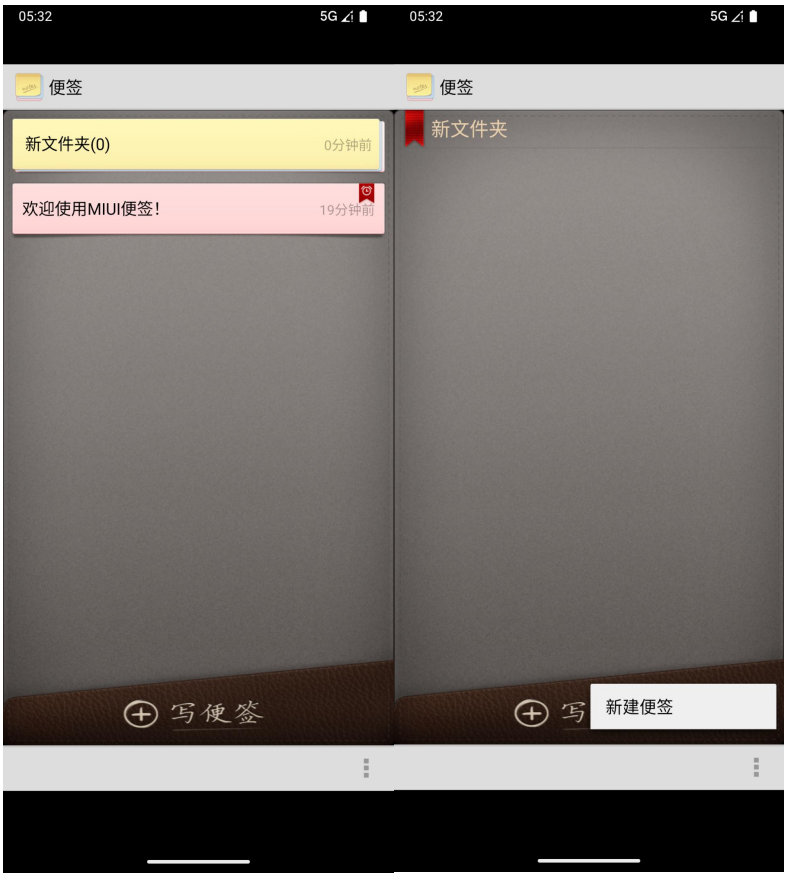
设置时钟后便签上会增加图标：

更换便签背景：

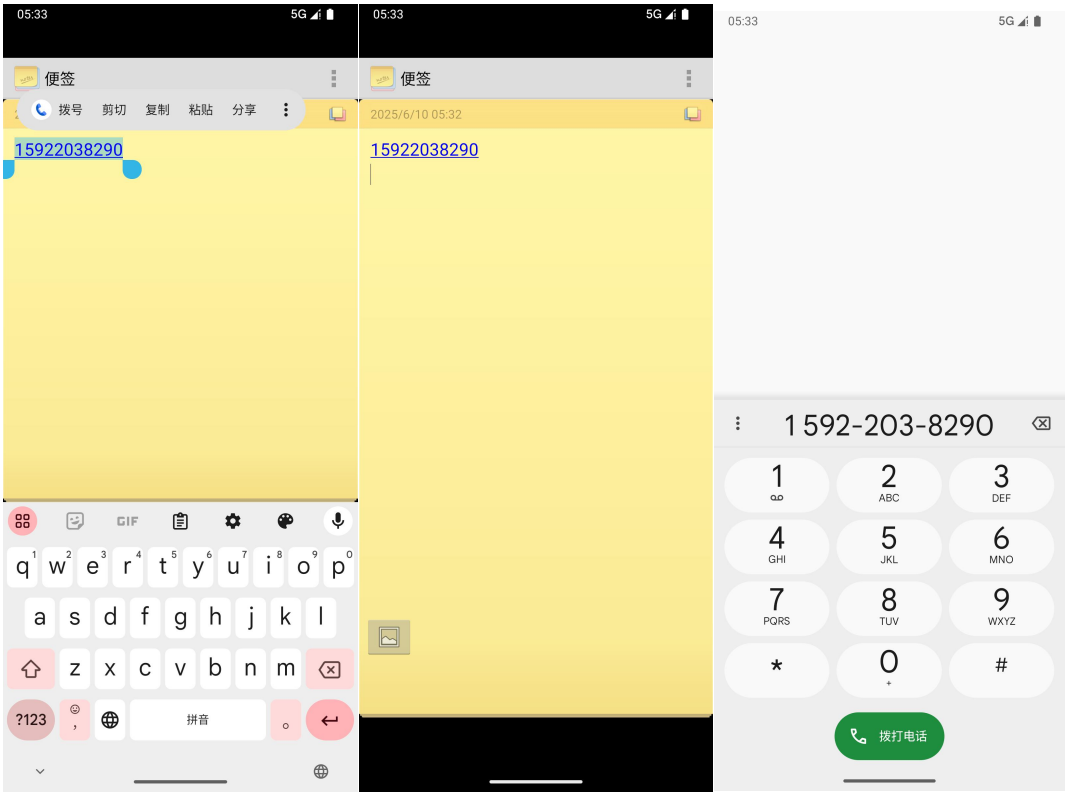
点击下方选项栏即可新建文件夹，编辑文件夹名称



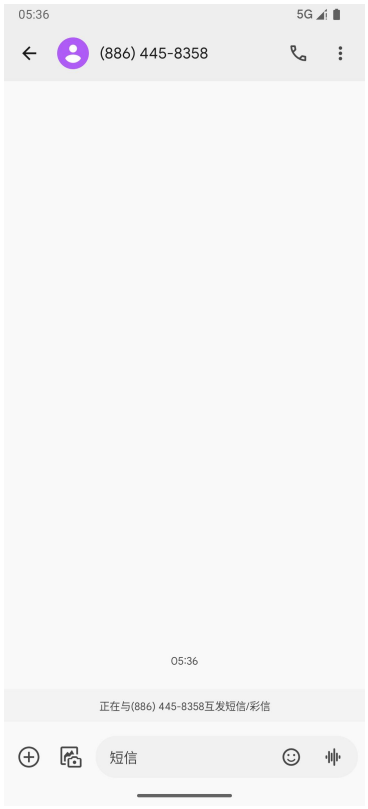
在 index 页可看到新文件夹



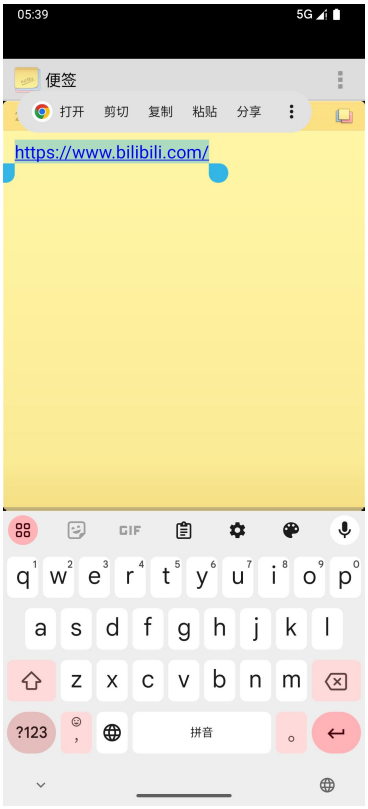
在便签页输入电话号码并选中，就会出现拨号选项



在便签页写上联系人号码，选中即可出现发短信选项



在便签页输入超链接，选中即可打开超链接



小米便签
软件设计规格说明书

日期：2025 年 6 月 1 日

文档变更历史记录

序号	变更日期	变更人员	变更内容详情描述	版本

目录

1、引言

1.1 编写目的

本文档是在概要设计说明书的基础上，对系统的进一步分析与设计的成果，是最终开发“小米边卡”的必要步骤。通过系统详细设计可以更好地将系统细化，以指导开发人员进行编码，其具体作用如下：该份详细设计说明书为参与项目的所有人员提供统一的业务说明、功能模块、数据库表表述，并在项目的开发阶段指导开发，在项目的测试阶段为测试提供测试依据。

1.2 读者对象

本报告的主要读者是相关业务部门、双方项目经理、IT 人员、实施人员、测试人员、系统维护人员。

1.3 软件项目概述

项目名称：小米便签.

用户单位：个人用户（普通消费者、学生、职场人士）及小型团队 。

开发单位：软件工程三人组。

需求描述：

1. 功能上，实现便签的创建、编辑、删除、分类（文件夹管理）、更换便签背景、多媒体插入（图片）、快捷操作（发短信、超链接、打电话）、闹钟提醒及多设备同步功能；
2. 性能上，要求响应速度≤1 秒，数据备份成功率≥99%，支持离线编辑与云端同步 。

1.4 文档概述

本文档包含引言、软件设计约束、软件设计三大部分。引言部分说明编写

目的、读者对象及项目背景；软件设计约束分析系统设计的目标、原则及限制条件；软件设计部分详细阐述体系结构、界面、用例、类、数据及部署设计 。

1.5 定义

DFD 设计：数据流图设计，用于描述系统数据流向及处理过程。

ER 设计：实体 - 关系设计，用于构建数据库表结构及关系 。

1.6 参考资料

[1] 《Android 应用开发规范》，小米科技，2022 年。

[2] 《软件设计模式与最佳实践》，机械工业出版社，2021 年 。

[3]林学森 深入理解 Android 内核设计思想

2、软件设计约束

2.1 软件设计目标和原则

设计目标：实现用户便签管理需求，确保系统具有良好的可扩充性、稳定性及安全性，支持图片等多媒体内容的高效存储与同步 。

设计原则：遵循模块化设计、高内聚低耦合、接口标准化原则，便于后续功能迭代与维护 。

2.2 软件设计的约束和限制

运行环境：硬件平台为 Android 智能手机 ；操作系统为 Android 11.0 。

开发语言：Java 。

标准规范：遵循 Android 官方开发规范 。

开发工具：Android Studio 2024.3.1、头歌平台 。

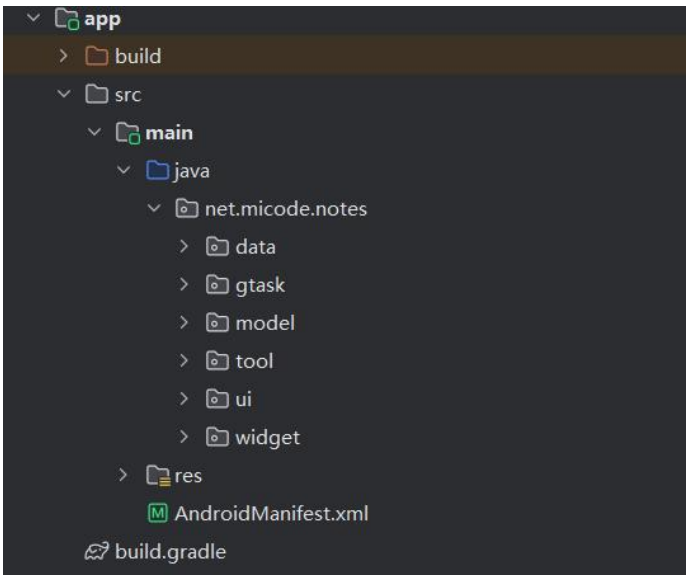
性能要求：单便签加载时间≤500ms，图片压缩后大小≤200KB，云同步延

迟≤10 秒 。

灵活性要求：支持背景切换、字体大小调整及自定义快捷键配置 。

3、软件设计

3.1 软件体系结构设计

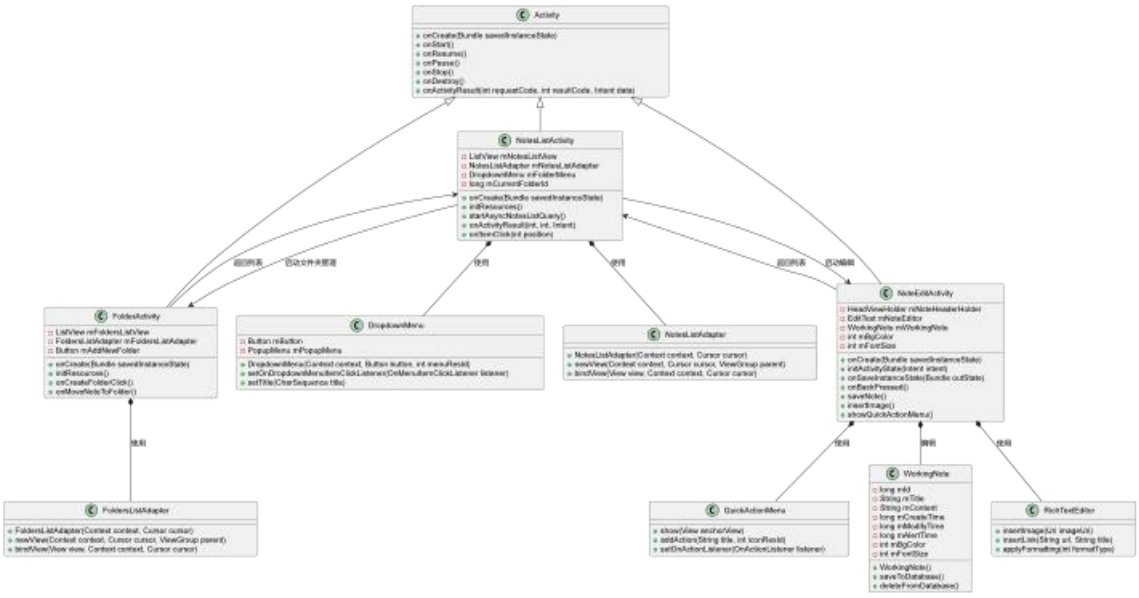


小米便签采用典型的 MVC 架构，将系统划分为模型、视图和控制器三大部分，模块职责清晰，便于维护与扩展。其中，model 和 data 包负责数据的定义与数据库访问，分别用于描述便签、文件夹等实体对象，以及提供基于 ContentProvider 的数据存取接口。

ui 包承担用户界面展示与交互逻辑，实现各类 Activity 和界面控制器，是连接用户操作与后端数据的核心部分。widget 包则负责实现桌面便签小部件功能，通过 AppWidget 机制提供快速便签查看与添加服务，增强用户体验。

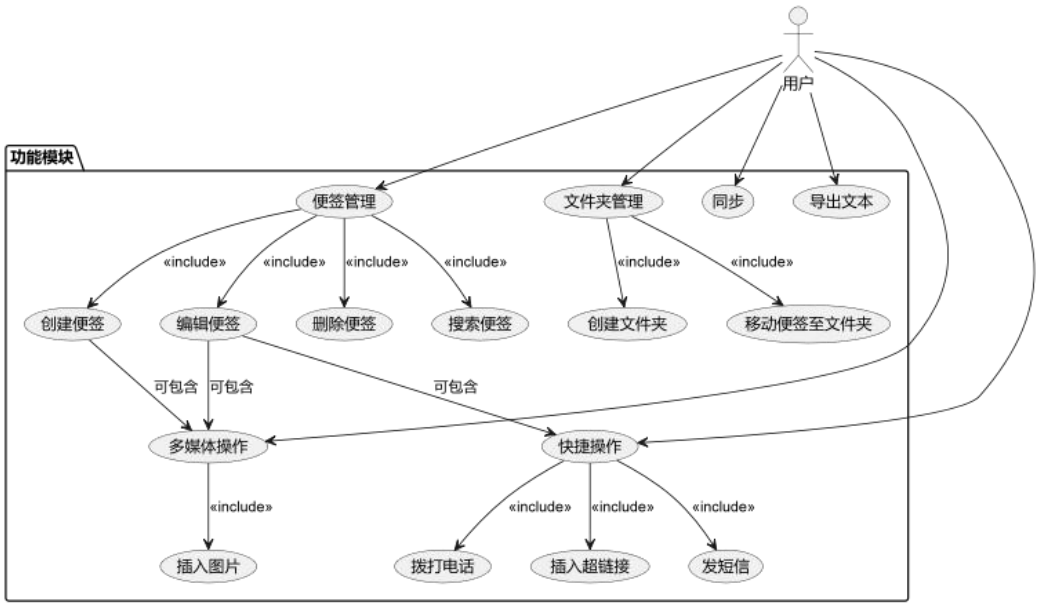
gtask 包实现与 Google Task 的云端同步功能，负责账户绑定、网络请求及数据同步处理。tool 包提供通用工具类，如资源解析、时间处理与配置常量等，服务于其他模块，增强系统的复用性和独立性。整体设计结构清晰，组件间低耦合高内聚，满足 Android 应用的组件化和模块化设计原则。

3.2 用户界面设计



涉及到多个用户界面相关的类，主要包括 NoteEditActivity、NotesListActivity 等，这些类继承自 Activity 类，用于展示不同的界面。同时，还有一些辅助类如 DropdownMenu、FoldersListAdapter 等用于界面的交互和数据展示。

3.3 用例设计



核心用例：

便签管理：包含创建、编辑、删除、搜索便签。

文件夹管理：创建文件夹、移动便签至文件夹。

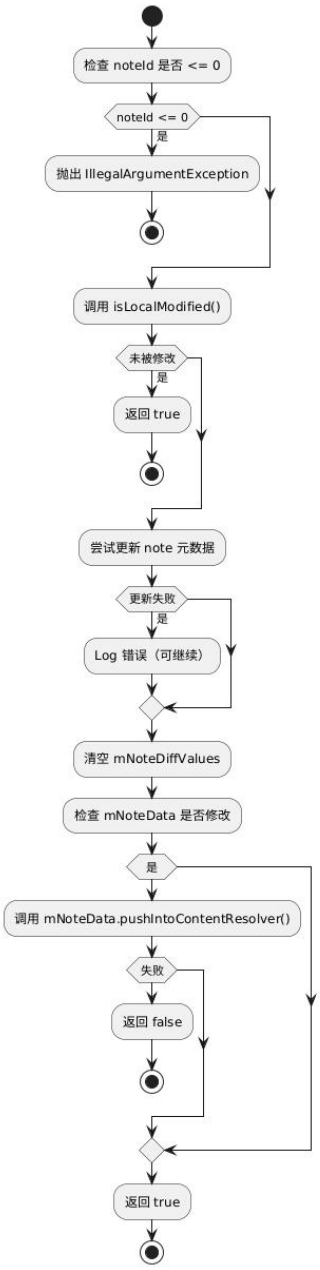
多媒体操作：插入图片、预览图片。

快捷操作：发短信、拨打电话、插入超链接。

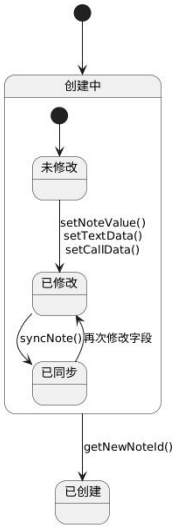
3.4 类设计

Note 类：

Note.syncNote() 方法活动图

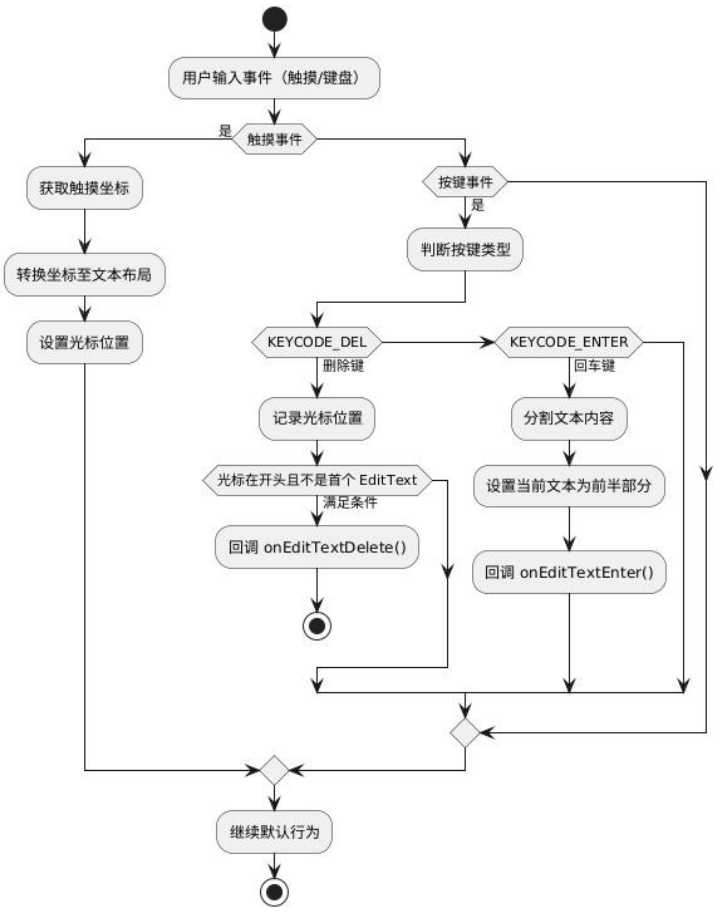


Note 对象状态图

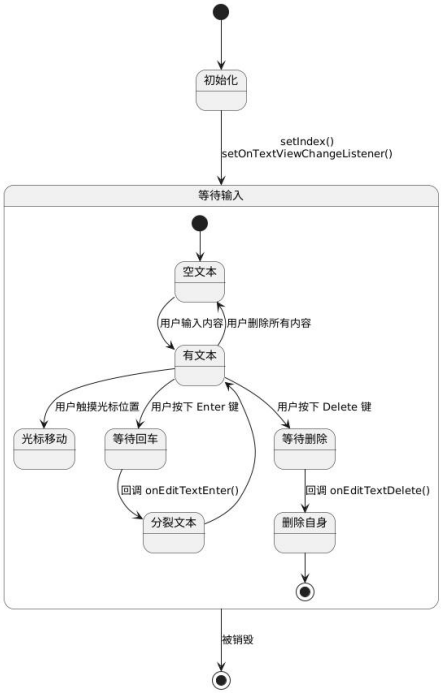


NoteEditText 类:

活动图

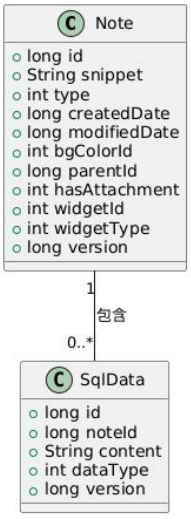


NoteEditText 对象生命周期状态图:



3.5 数据设计

3.5.1 数据库及表设计类图



小米便签软件中永久数据的核心设计结构主要包含两个核心类：

- Note 类（笔记表）

  存储笔记的元信息，包括笔记 ID、摘要内容（snippet）、笔记类型（普通笔记、文件夹、系统类型等）、创建时间、修改时间、背景颜色 ID、父笔记 ID（用于层级结构）、附件标志、桌面组件 ID 及类型、版本号

等字段。

这张表承担笔记的整体标识与管理功能。

- SqlData 类（笔记内容数据表）

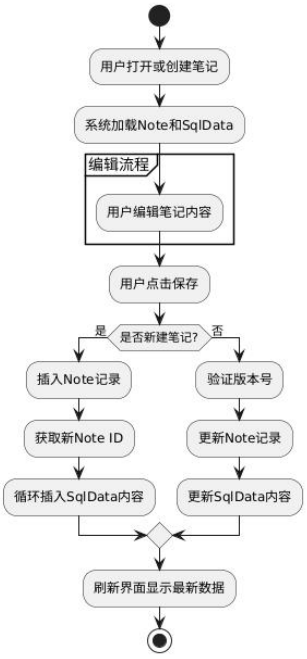
  存储具体的笔记内容数据，每条内容数据关联一个 Note（通过 noteId 字段），包含内容字符串、内容类型和版本号等。

  一条笔记（Note）可关联多条内容数据（SqlData），形成笔记正文的具体内容。

- 关系

  Note 和 SqlData 之间是“一对多”关系，即一个笔记对应多条内容数据。

3.5.2 数据操作活动图



活动图描述了用户对笔记数据的典型操作流程，主要分为以下几个步骤：

1. 打开或创建笔记

用户打开已有笔记时，系统会加载对应 Note 和相关的 SqlData 内容；新建笔记时则初始化一个新的 Note 对象。

2. 编辑笔记内容

用户对笔记的元信息和内容数据进行修改，包括文本编辑、颜色修改、附

件添加等。

3. 保存操作

用户点击保存后，系统判断当前笔记是新建还是已有笔记：

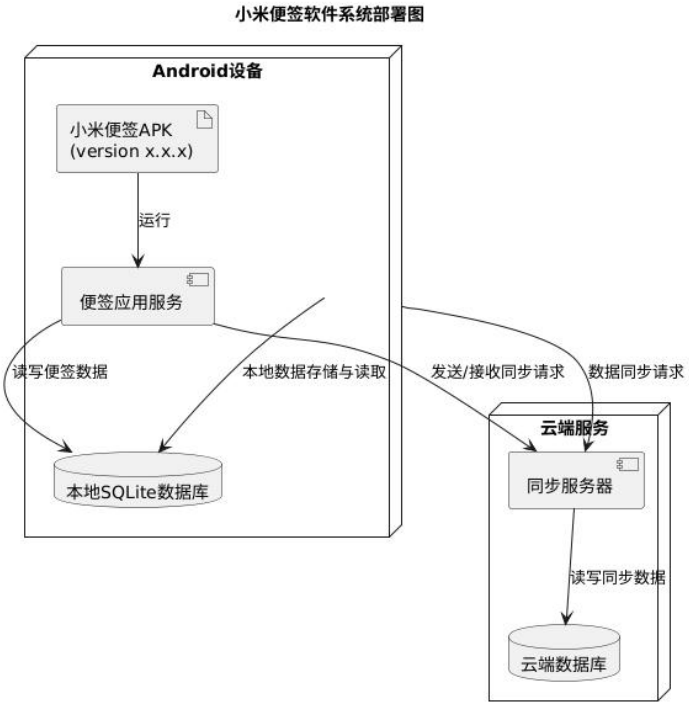
若是新建笔记，系统会先将 Note 记录插入数据库，获得新的笔记 ID，再将所有相关内容数据逐条插入 SqlData 表中。

若是已有笔记，系统会先检查笔记版本号以避免冲突，然后更新 Note 表中元数据，同时更新 SqlData 表中的内容数据。

4. 刷新界面

保存完成后，系统会重新加载最新数据，确保界面显示的是最新笔记信息。

3.6 部署设计



- 客户端部署

小米便签以 APK 包形式安装在 Android 设备上，支持 Android 11.0 及以上版本。应用运行在 Android 设备的应用进程中，使用本地 SQLite 数据库存储笔记及相关数据。

应用需要申请关键权限，包括：

存储权限：用于访问和管理笔记附件和数据文件。

相机权限：用于拍摄笔记图片或附件。

电话权限和短信权限：支持特殊笔记类型（如通话记录、短信内容）读取。

本地存储

应用使用 Android 本地数据库 SQLite 进行数据持久化存储，保证数据即使在无网络情况下也能正常访问和编辑。

- 云端同步

小米便签支持数据云同步功能，客户端通过网络请求与云端同步服务器通信，实现用户多设备数据一致性。同步服务器访问云端数据库管理用户笔记的云端存储。

- 系统架构部署

部署架构分为移动端和云端两部分。移动端负责用户交互和本地数据管理；云端负责数据同步和备份，保证用户数据安全和多终端同步体验。