北京邮电大学

网络空间安全学院

小米便签 2.0 新增功能软件设计报告

课程名称：软件工程技术基础

授课教师：芦效峰

成员一：江增臻 2020211938 2020211802班

成员二：谢思远 2020211867 2020211803班

成员三：林于翔 2020211919 2020211806班

2022 年 **6** 月 **16** 日

目录

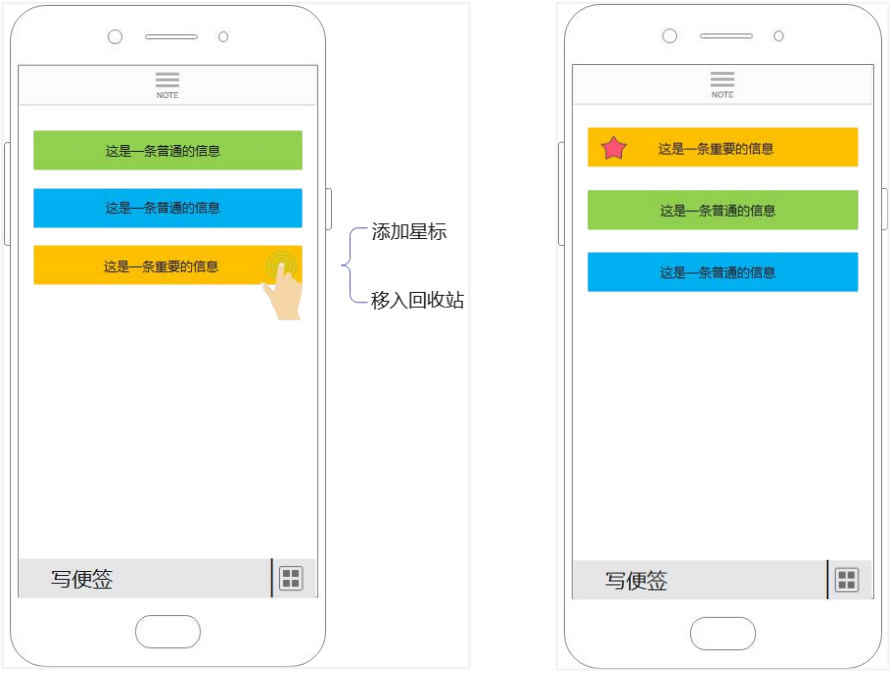
一、新增功能说明

（1）回收站功能

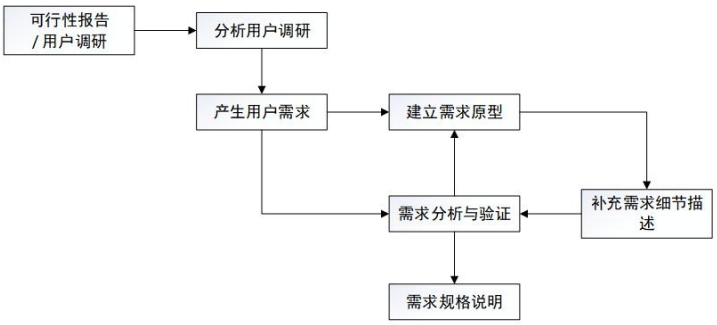
功能描述：用户删除便签后，便签从主页面消失，缓存到回收站中，一段时间后，系统到期彻底删除便签。回收站中的便签可以被恢复，也可以被用户手动直接彻底删除。样式设计图如下：

（2）标签置顶功能

功能描述：小米便签默认的排序方式是时间排序。增添该功能后，用户可以给某些便签打上“星标”，再切换为“星标排序”，打上星标的标签就会被置顶。样式设计图如下：



二、需求分析



2.1 设计约束

1. **预期的软硬件环境**
   预期软件环境：Android 6.0 及其后续更新版本系统
   预期硬件环境：安卓手机
2. **预期的使用环境**
   用户需要长久记录事项时
3. **预期的使用行业**
   便签软件具有普适性，适合任何行业
4. **预期的使用用户**
   使用安卓手机的个人用户

2.2 市场分析、需求调研和功能定位

主要方法：面谈法、实地调研、同类产品对比

（1）回收站功能

1.市场分析：进行市场调研和同类产品比对后，我们发现市场上大多数手机自带的记事本、备忘录、便签等相同线路的产品都有回收站功能，比如华为备忘录，类似具有回收站功能的还有手机相册，被删除的照片被放进“最近删除”。

2.需求调研：用户删除的便签不会立刻从硬盘上移除，而是在回收站中保存一段时间，如果这段时间用户没有恢复删除的数据，到期系统才会自动彻底清除数据。这样的功能给用户留下了反悔的空间。

3.功能定位：回收站功能定位外围功能，将该功能纳入考虑的目的时与同类产品抵消

功能上的竞争差距。

（2）星标置顶功能

1. 市场分析：通过现场观察的需求调研，我们发现，很多备忘录、便签类的产品仅仅支持按照便签创建时间或者编辑时间排序。

2. 需求调研：与一些用户面谈时我们发现，很多经常使用便签的人表示仅仅这种排序不太方便，比如很早以前创建的便签要滑动很久才能找到，而有时候这些便签可能非常重要，或者常用。

3. 功能定位：置顶功能定位杀手功能。经过我们调研，很多同类产品没有类似重要度排序、置顶等功能，大多数是按照创建或者编辑时间排序，该新增功能

2.3 功能需求

功能需求包括三点基本的要求：
完整性、准确性、一致性

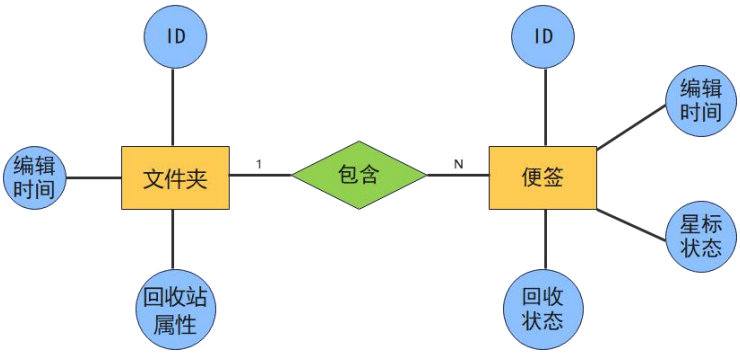
针对我们增添的两个功能：回收站功能和置顶功能对他们的功能需求三要素作出如下的设计：

功能需求的要求

优化功能	完整性	准确性	一致性
回收站	从用户删除便签，到软件到期自动清理回收站中便签，以及用户恢复便签，整套服务应当功能完整，流程清晰	用户的指令必须得到软件的准确处理；到期自动清理需要每个回收站中的便签必须获得准确的计时功能支持；日志要按时和准确地更新	后端数据库中的数据一致性不能被破坏；删除前和恢复后的便签要保持一致
置顶功能	置顶、取消置顶；这些提供给用户的服务要完整，可以采用，也可以取消，流程清晰，处理有序	在用户进行有关的操作时，软件准确分析和按照流程处理用户的指令，呈现给用户的效果、数据库中数据的修改、日志的更新修改准确且唯一，无歧义	数据库中的数据在与置顶有关的操作时保持一致性

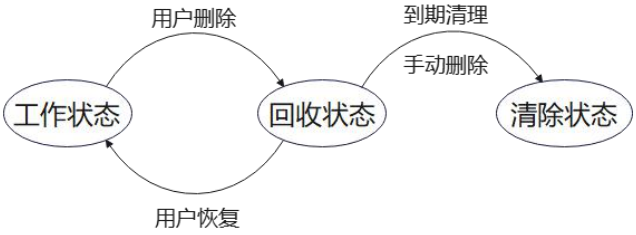
2.4 需求建模

数据模型：实体关系图

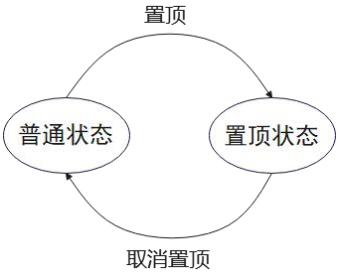


行为模型：状态准换图

（1）回收站功能

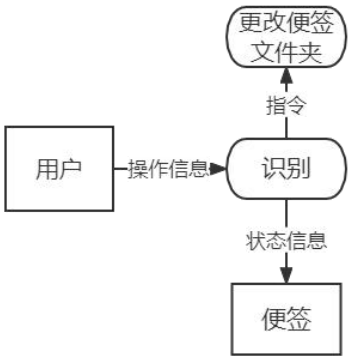


（2）星标排序功能

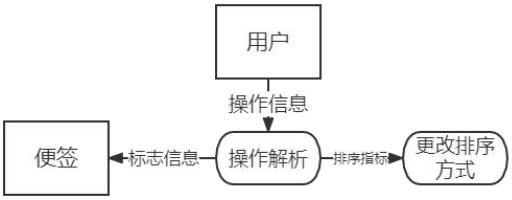


数据流图：

（1）回收站



（2）星标置顶



2.4 其他非功能需求设计

三、数据设计

数据设计指将实体－关系图中描述的对象和关系，以及数据词典中描述的详细数据内容转化为数据结构的定义。

在小米便签 1.0 的基础上开发新功能，因此必须在原有数据结构的基础上进行数据设计。

小米便签中的每个“便签”和“文件夹”的信息都组织在数据库中 Notes 表上，type 字段标志该数据项对应的实体的类型。type=0 的项是便签，type=1 是文件夹，每个个体都有唯一的 ID 标识。实体关系图中元素的数据设计如下：

回收站/文件夹

属性	映射字段
ID	_id()（已有字段）
编辑时间	modified_data（已有字段）
回收站属性	_id=-4（已有字段）
创建一个全局文件夹，即type=1,赋予唯一 _id,将该文件夹作为回收站即可	

便签

属性	映射字段
ID	_id()（已有字段）
编辑时间	modified_data（已有字段）
星标状态	is_star（新插入字段）
回收状态	is_delete（新插入字段）
回收状态	delete_date（新插入字段）

数据流/过程图中模块的数据设计如下：

用户操作信息映射为 java 中的事件 Event;对操作信息的操作解析处理映射为 java 中的监听器（EventListener）对事件的监听和响应，可以映射成 EventListener 中的某一具体方法，再调用小米便签中原有的方法进行操作/或重写方法；

元素	数据设计	输入	效果
操作信息	Event对象		
识别/操作解析	EventListener对象		解析Event调用对应方法
更改便签文件夹	method()方法	_id集合	修改便签is_delete值
更改便签星标标志	method_2()方法	_id集合	修改便签is_star值
更改排序方式	method_3()方法 sortway新插入Notes表的字段	sortway作为method_3()参数	sortway选择排序方式

更新后的 Notes 表数据字典：

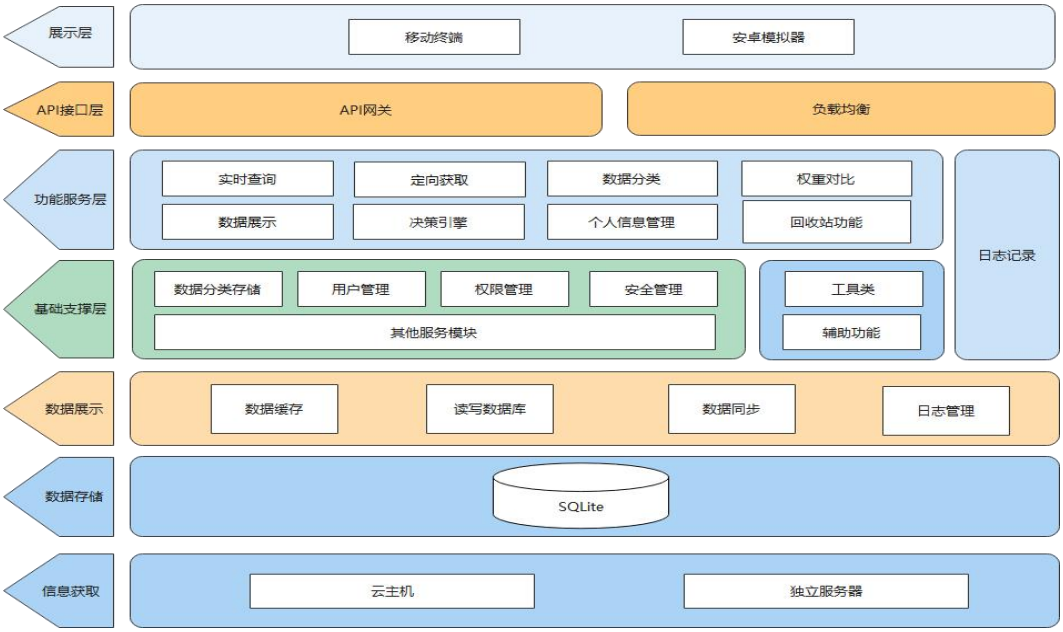
表 1　Notes 表数据字典

字段名	类型	默认值	是否主键	描述
_id	INTEGER		1	便签 id
parent_id	INTEGER	0	0	父文件夹 id
alert_date	INTEGER	0	0	提醒时间
bg_color_id	INTEGER	0	0	背景颜色 id
created_date	INTEGER	(strftime('%s','now') * 1000)	0	创建时间
has_attachment	INTEGER	0	0	
modified_date	INTEGER	(strftime('%s','now') * 1000)	0	修改时间
delete_date	INTEGER	0	0	删除时间
is_delete	INTEGER	0	0	便签是否删除
notes_count	INTEGER	0	0	当前文件夹下便签数量
snippet	TEXT	Null	0	
type	INTEGER	0	0	
widget_id	INTEGER	0	0	桌面组件的 id
widget_type	INTEGER	-1	0	桌面组件类型
sync_id	INTEGER	0	0	同步进程的 id
local_modified	INTEGER	0	0	是否重新编辑过
origin_parent_id	INTEGER	0	0	原父文件夹 id
gtask_id	TEXT	Null	0	同步任务 id
version	INTEGER	0	0	版本
is_star	INTEGER	0	0	是否添加星标，值为 0 或 1

四、软件总体设计

4.1 软件整体模块分析

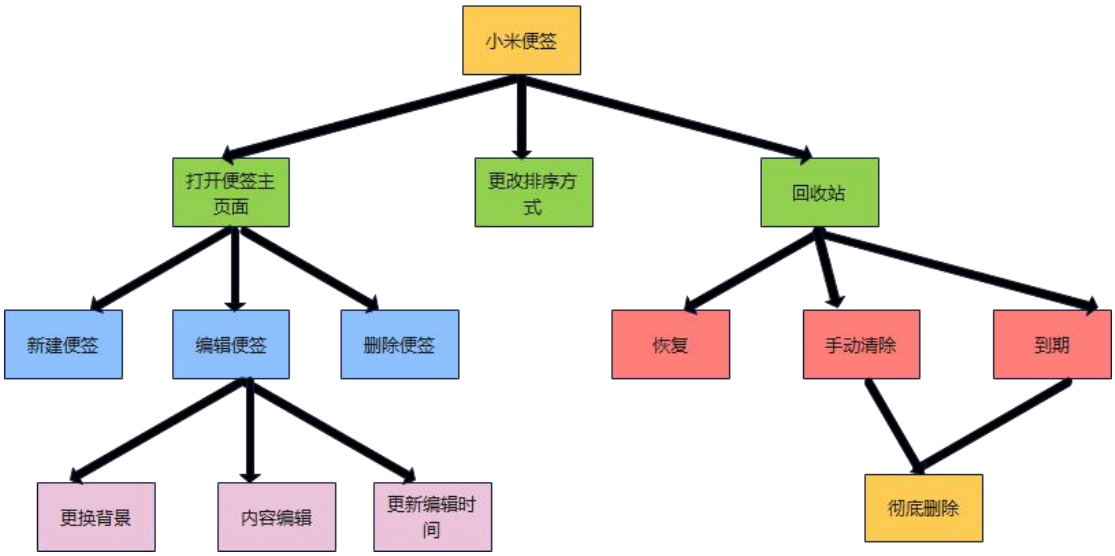
我们对小米便签源码进行分析，同时结合新增的功能，给出小米便签整体的层次——模块划分：



模块划分图

4.2 功能分解和软件模块设计

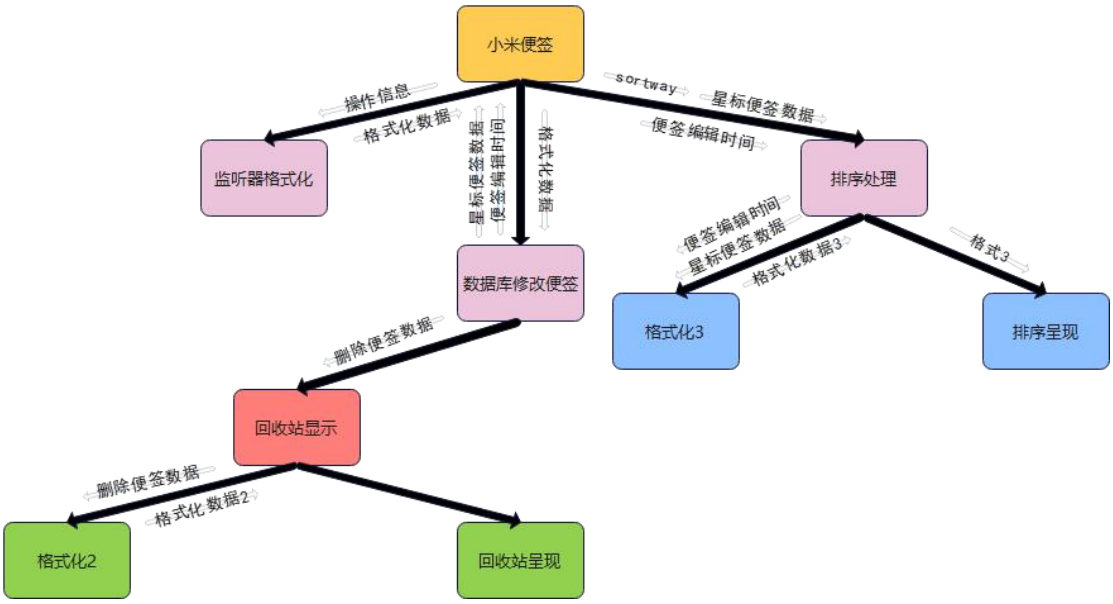
结合原有功能和新增的两个功能，设计系统结构图设计如下：（我们主要针对新增功能进行分析）



小米便签主页面后，可以选择新建便签、编辑便签和删除便签三个模块。
编辑便签调用更换背景、内容编辑、系统自动更新便签编辑时间三个下属模块；
回收站模块下提供恢复便签、手动清除便签和到期自动清除便签三个功能模块，其中手动清除和到期自动清除，都调用“彻底删除”这一功能模块。

4.3 变换流映射

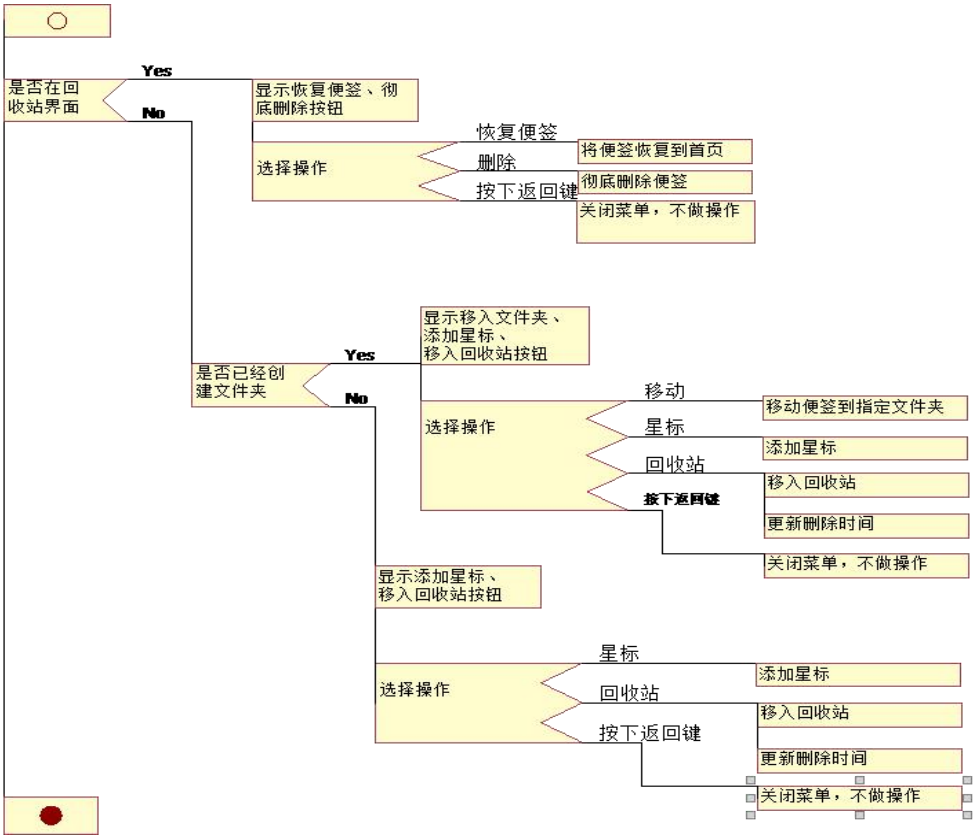
模块划分和系统设计后，形成具有预定功能的模块组成结构，接下来做进一步的总体设计，表示出模块间的控制关系，并给出模块之间的接口——数据交换模式。 如下给出变换流映射：（主要针对新增功能进行分析，结合部分原有模块）



五、详细设计

5.1 过程设计

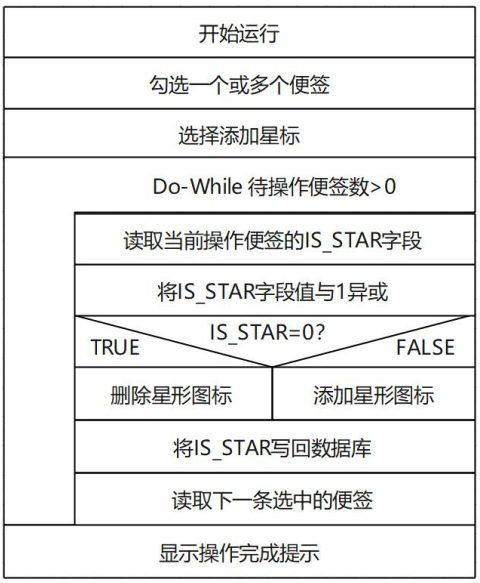
（1）回收站：问题分析 PAD 图



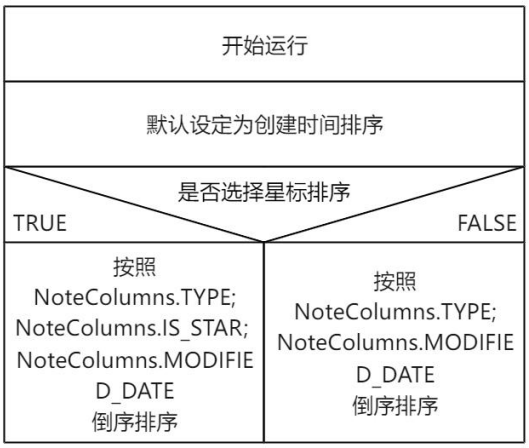
（2）星标置顶：NS 图

在数据设计中，我们对 Notes 表设计了 IS_STAR 字段，用于判断是否添加星标。在星标排序功能中，将根据用户选择的排序方式，按照不同字段进行排序。

添加星标算法 NS 图



选择排序算法 NS 图



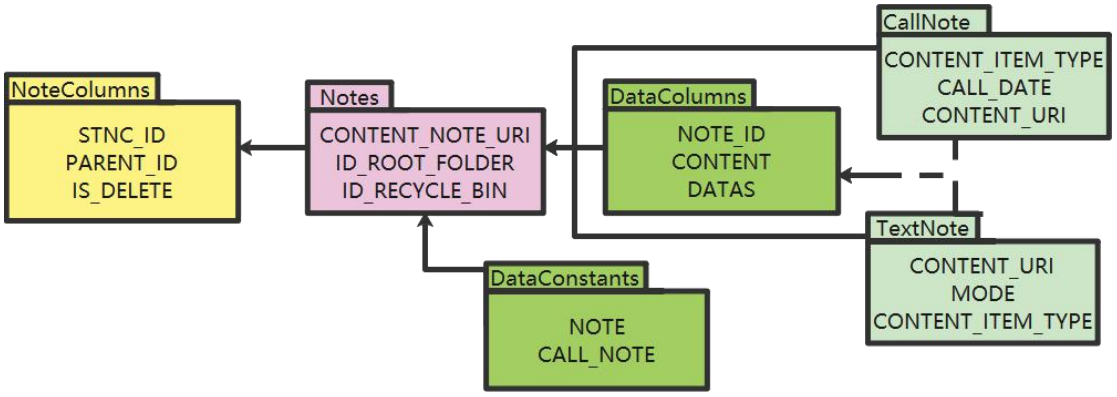
5.2 模源码分析和准备工作

UML 详细设计：

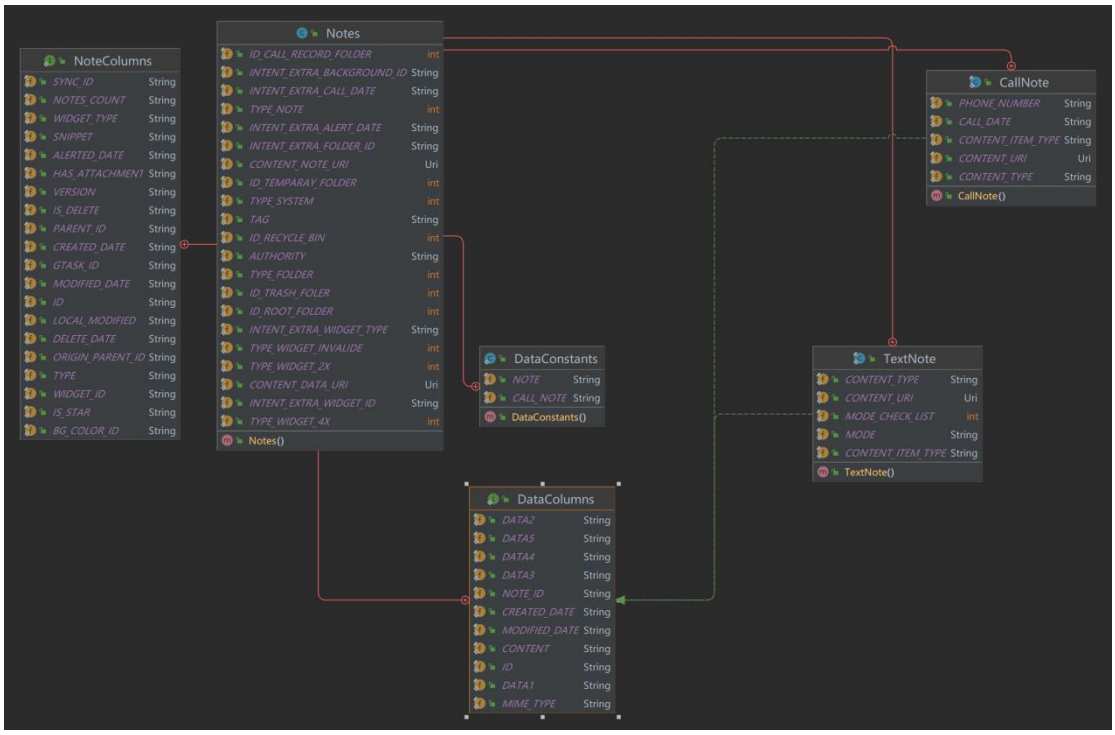
在我们的需求中，Note 数据是极其重要的。Note 表记录了便签的各种信息。在原有基础上，我们对数据库中的 Note 表进行了更新：添加了 IS_STAR、DELETE_DATE 字段，并且设计了触发器 NOTE_CHANGE_DELETE_DATE_ON_UPDATE_TRIGGER，以便在删除时自动更新 DELETE_DATE 字段。

我们参考小米便签原有架构，给出详细的设计分析：

Notes 类是软件的元数据，内部设计了 NoteColumns 接口，在软件初次安装创建数据库时，NotesDatabaseHelper 将会根据 NoteColumns 接口创建相应的表。Notes 类的 UML 类图如下：



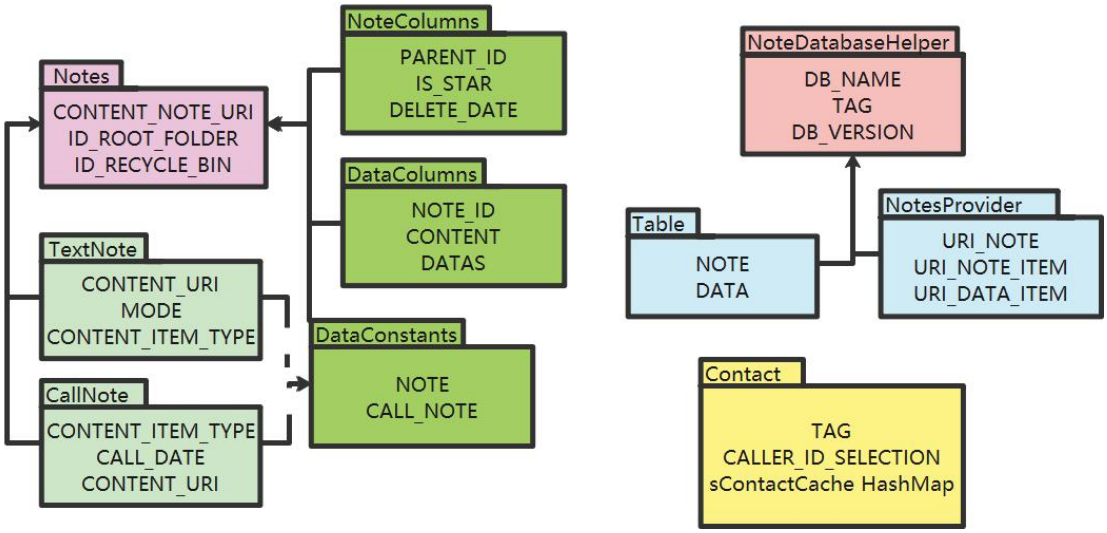
上面是简化的模块调用图，详细原图如下：



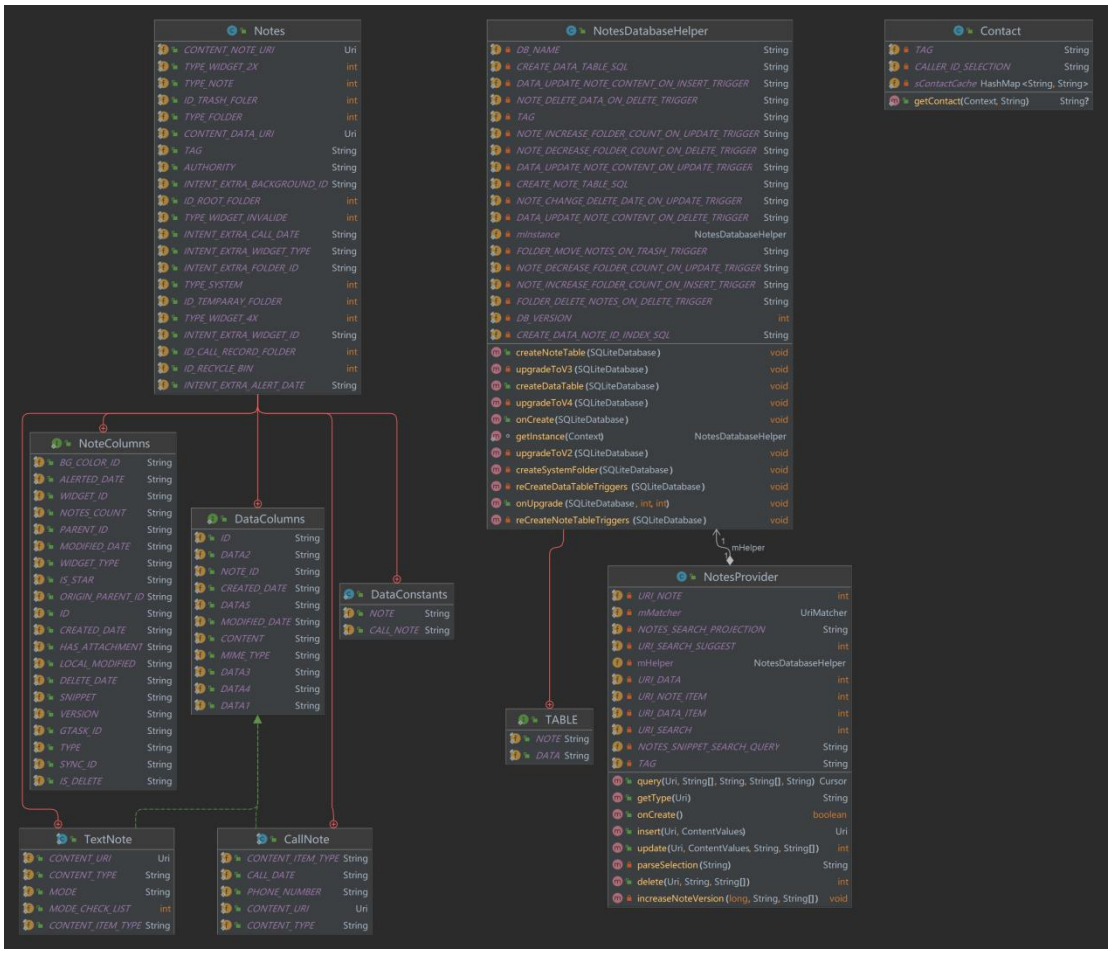
整体数据部分的 UML 如下：

Provider 是 Android 四大组件之一，是一种数据共享机制(封装数据的接口规范)。是一个数据库的代理，实现跨应用间的数据共享，类似于 window 服务本身不负责启动和关闭，多个应用可同时访问统一 provider。在本软件中，NotesProvider 是 ContentProvider 的子类，负责向 Activity 提供统一的接口以便增删改查数据。SQLite 是 Android 内置的一个小型、关系型、属于文本型的数据库。

NotesDatabaseHelper 继承自 SQLiteOpenHelper。Android 中，通过 SQLiteOpenHelper 类来实现对 SQLite 数据库的操作。Android 提供了对 SQLite 数据库的完全支持，应用程序中的任何类都可以通过名称来访问任何的数据库，但是应用程序之外的就不能访问。



上面是简化的模块调用图，详细原图如下：



5.3 数据结构和算法设计

接口：

在项目中，NotesProvider 类负责向 Activity 提供统一的接口以便增删改查数据，是 ContentProvider 的子类。ContentResolver 类对所有的 ContentProvider 进行统一管理。我们的 DataUtils 类封装了操作便签时的常用操作，通过操作 ContentResolver 实现。

现介绍**回收站**所运用的方法及具体使用过程：

1. 移入回收站：

类&方法名	作用	参数	类型	说明	返回值	返回值类型
DataUtils.batchMoveToFolder	批量移动便签到指定文件夹	resolver	ContentResolver	提供 ContentResolver 对象	True：成功移动 False：移动失败	boolean
		ids	HashSet<Long>	选中便签的 id 集合		
		folderId	long	目标文件夹		

移动到回收站时，folderId 指定为-4。

2. 恢复便签：

恢复便签时，folderId 指定为-1。

类&方法名	作用	参数	类型	说明	返回值	返回值类型
DataUtils.batchMoveToFolder	批量移动便签到指定文件夹	resolver	ContentResolver	提供 ContentResolver 对象	True：成功移动 False：移动失败	boolean
		ids	HashSet<Long>	选中便签的 id 集合		
		folderId	long	目标文件夹		

3. 彻底删除：

类&方法名	作用	参数	类型	说明	返回值	返回值类型
DataUtils.batchDeleteNotes	批量移删除便签	resolver	ContentResolver	提供 ContentResolver 对象	True：删除移动 False：删除失败	boolean
		ids	HashSet<Long>	选中便签的 id 集合		

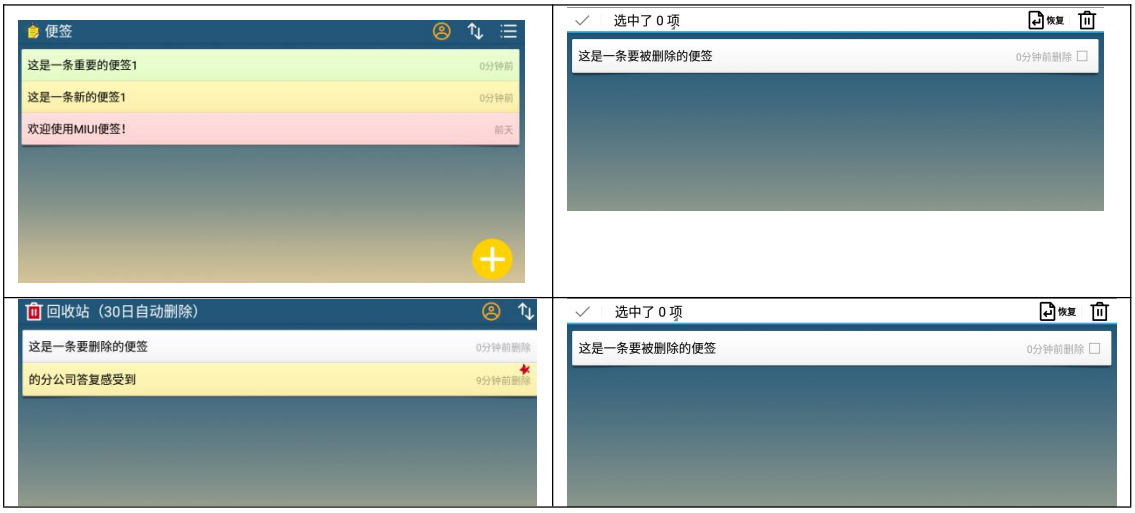
添加星标所运用的方法及具体使用过程：

类&方法名	作用	参数	类型	说明	返回值	返回值类型
DataUtils.batchAddStar	批量添加星标	resolver	ContentResolver	提供 ContentResolver 对象	True：添加移动 False：添加失败	boolean
		ids	HashSet<Long>	选中便签的 id 集合		

设置星标排序所运用的方法及具体使用过程：

类&方法名	作用	参数	类型	说明	返回值	返回值类型
NotesListActivity.startAsyncNotesListQuery	排序	sortway	Int	排序方式	无	无

5.4 设计测试用例



六、安全设计

6.1 安全问题估计和需要考虑的安全设计原则

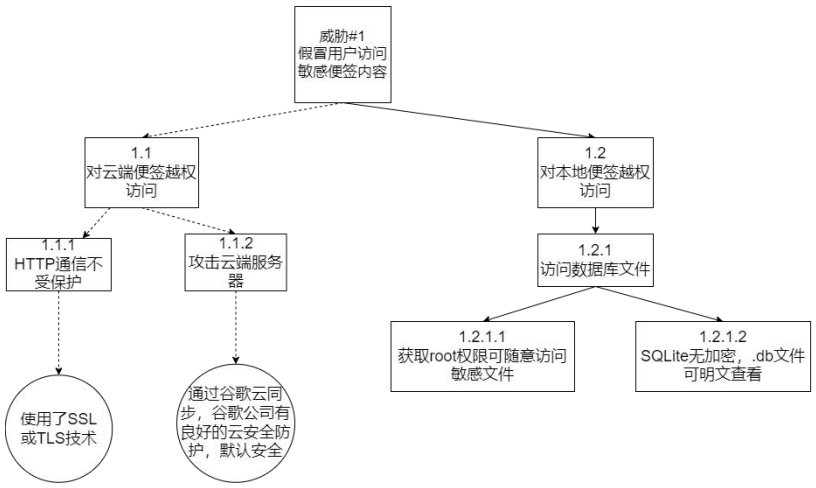
我们在小米便签增添的功能（回收站功能、星标排序功能），主要可能遇到的安全问题有：

1. 数据一致性问题：设计后端数据修改，必须保证数据一致性。
2. 权限和修改（数据保护）：回收站和星标功能的实现需要对后端标签类的属性进行增添、修改，因此需要界定修改、访问的条件，对数据进行保护。
3. 异常处理：要对可能的异常设计异常处理模块，防止系统崩溃或漏洞被攻击者利用。

主要涉及到的安全设计原则有：安全的错误退出、最小授权原则、保护机制的简单性、最小共享机制等

6.2 威胁建模：安全设计威胁树

为了保障软件的安全，对软件进行安全建模是很有必要的。建立威胁树，可以系统地分析所开发的应用程序中的威胁和漏洞。威胁树描述了攻击者破坏各组件所经历的决策过程。我们分析了越权访问敏感信息的威胁树



6.3 风险排查和威胁消减

（1）数据库查询问题：数据保护

回收站、重要度排序均需要与数据库进行交互，在代码层面，我们通过游标（cursor）与 SQLite 数据库交互，在每次交互完成后应该关闭游标，即使用 cursor.close()，保护数据。

（2）风险排查：已经存在资源使用未关闭问题修复

在原项目 net.micode.notes/tool/BackupUtils.java 中存在如下函数：

```
1.  private PrintStream getExportToTextPrintStream() {
2.      File file = generateFileMountedOnSDcard(mContext, R.string.file_path,
3.              R.string.file_name_txt_format);
4.      if (file == null) {
5.          Log.e(TAG, "create file to exported failed");
6.          return null;
7.      }
8.      mFileName = file.getName();
9.      mFileDirectory = mContext.getString(R.string.file_path);
10.     PrintStream ps = null;
11.     try {
12.         FileOutputStream fos = new FileOutputStream(file);
13.         ps = new PrintStream(fos);
14.     } catch (FileNotFoundException e) {
15.         e.printStackTrace();
16.         return null;
17.     } catch (NullPointerException e) {
18.         e.printStackTrace();
19.         return null;
```

```
20.      }
21.      return ps;
22.          }
```

注意到 try 语句，资源创建应当使用“try-with-resource”模式，用完资源应当自动关闭防止发生内存泄露问题，所以需要重构整个 try-catch 模块。重构后如下：

```
1.  FileOutputStream fos = null;
2.  try {
3.          fos = new FileOutputStream(file);
4.          ps = new PrintStream(fos);
5.      } catch (FileNotFoundException e) {
6.          e.printStackTrace();
7.          return null;
8.      } catch (NullPointerException e) {
9.          e.printStackTrace();
10.         return null;
11.     } finally {
12.         if (fos != null) {
13.             try {
14.          fos.close();
15.             } catch (IOException e) {
16.          e.printStackTrace();
17.             }
18.         }
19.     }
```