小米便签应用代码泛读、标注和维护报告

黄泽楷　秦薪淇

2025 年 1 月 14 日

摘要

MiCode便签是由MIUI团队开发的开源Android应用项目，采用Java语言实现。项目具有完整的代码结构，涵盖用户界面、数据存储、系统服务等Android应用的核心模块，总计约13000行代码，包含6个功能包和39个类，基于MIT开源协议。尽管应用功能相对简单，但其模块设计清晰、结构合理，是学习和研究Android开发的优秀参考。

本报告围绕MiCode便签项目展开，对其进行了全面的代码泛读、标注和功能维护，并取得了以下详细成果：

1. **代码泛读：**分析了项目的整体架构和主要功能模块，明确了MVC分层设计（界面层、业务层、模型层、数据层）的实现思路。

   梳理了类与类之间的交互关系，聚焦核心功能（如便签创建、删除、移动）的实现路径，为后续功能扩展奠定了基础。

2. **代码标注：**对6个包中39个类的源码以及res包中部分xml文件完成了总计3924左右行详细注释，包括类的功能说明、关键方法逻辑解析、关键语句和语句块功能说明以及重要变量定义，增强了代码的可读性和可维护性。

3. **功能扩展与维护：**在项目的基础上新增了16项功能，涵盖文本处理、界面优化、隐私保护、语音交互、翻译、大模型使用等领域，显著提升了应用的实用性与用户体验。

   新增代码量约7000行，优化了代码质量并通过质量检测工具验证，确保结构清晰、逻辑一致。

通过本次实践，我们不仅掌握了针对大型项目的代码分析和优化方法，还在功能设计、模块实现及质量控制方面积累了宝贵经验。同时，团队在协作开发和任务分工中得到了磨合与提升。这些成果为我们后续的软件工程实践提供了丰富的借鉴和坚实的基础。

目录

1 小米便签的代码泛读

1.1 功能描述

根据对开源软件的使用以及对代码的阅读和理解，该软件的整体功能描述如下，其软件需求的用例模型如图1所示。

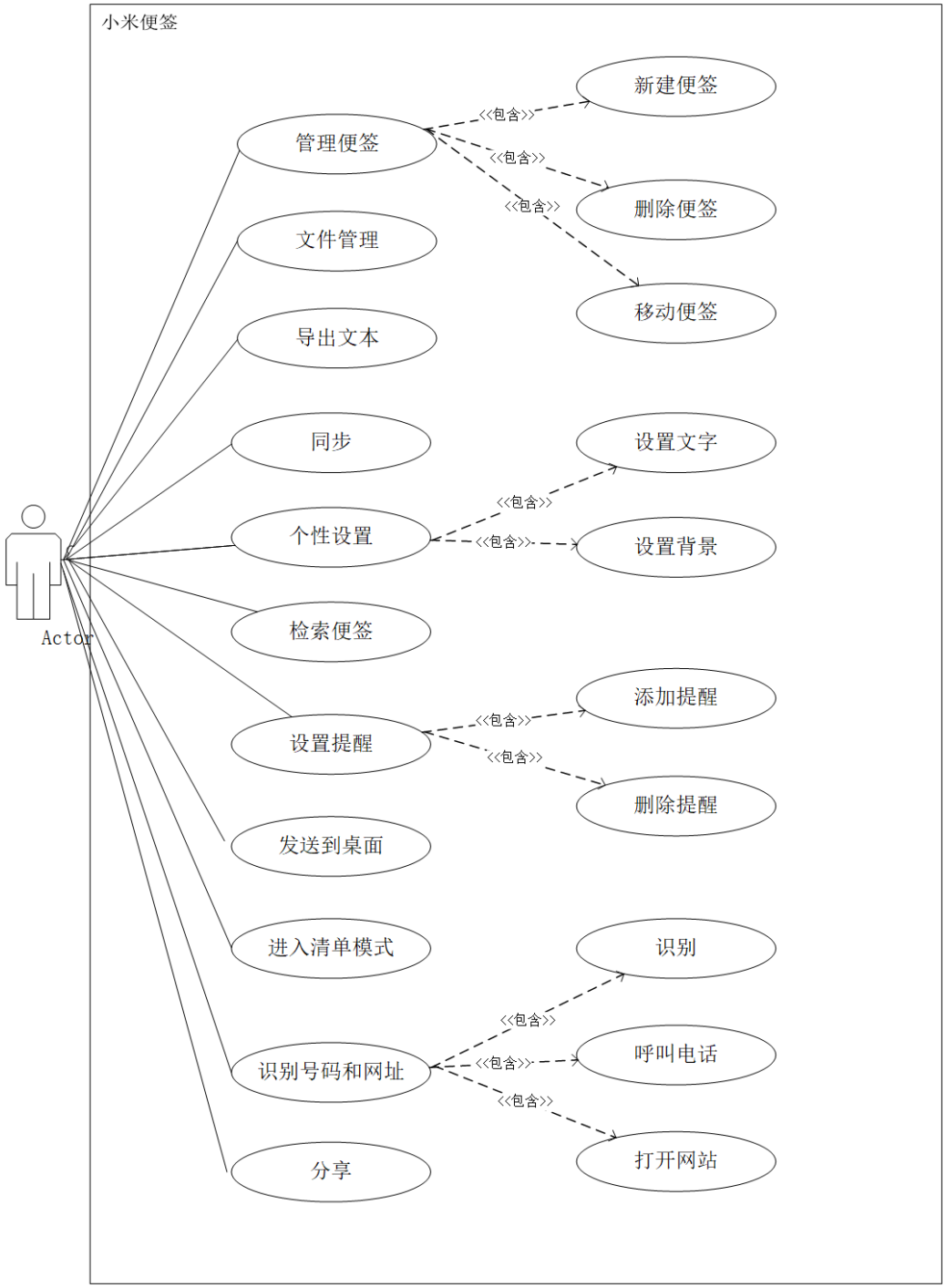


图 1: 小米便签的软件需求用例图

下面根据用例图详细阐述小米便签的功能:

- 功能1：新建/删除/移动便签

  1. 新建便签：通过小米便签软件的主界面下方的”写便签”快捷键和文件夹、便签视图下的选项”新建便签”可以在当前目录创建一个便签并打开进入文本编辑。其中主界面下的快速创建方式可以迅速创建一个待编辑的便签，用于满足临时迅速记录的需求。
  2. 删除便签：通过长按便签并选择删除选项可以删除已创建的便签，用于对便签集进行有效的管理。
  3. 移动便签：通过长按便签并选择移动选项可以将已创建的便签移动到某个文件夹中，用于在快速创建便签后对便签集进行有效的管理。

- 功能2：文件管理
  在当前目录下新建、查看、删除一个文件夹用于分类管理便签。用户在主界面的选项中可以选择“新建文件夹”创建一个文件夹，根据create参数值设置对话框标题（创建或是修改文件夹），设置对话框的正确定按钮和取消按钮。

- 功能3：导出文本
  在主界面的选项中可以选择”导出文件”，在Android手机提供SD卡支持的情况下，将小米便签中的便签内容逐个转化为.txt的文本文档。

- 功能4：同步
  在主界面的选项中可以选择”同步”，与Google Task中的备忘录事项，将本地的事项上传到服务器，或将Google服务器上的表单下载到本地。

- 功能5：检索便签
  在主界面的选项中可以选择 ”搜索”，通过关键词查找到包含该关键词的便签，显示在界面上。

- 功能6：修改字体大小
  在便签编辑的界面，在选项中选择”字体大小”，可以将当前便签的所有字体进行放大和缩小，其中包括4中字体大小：小、正常、大、超大。

- 功能7：进入清单模式
  在便签编辑的页面，可以选择进入清单模式的选项。选择后，在便签的每一行（段内部的自动换行除外）行首出现一个勾选框，用于当前便签下标记某事项的完成情况。如果该事项已完成，则用户在勾选框中轻触，此时勾选框中将出现一个对勾，框后的陈述文字被添加了中央删除线。（外侧文件夹并不能显示事项的完成状态，可优化）

- 功能8：发送到桌面
  在Android操作系统的桌面创建小米便签的小部件后，在编写便签完成后，使用选项”发送至桌面”，便可在便签小部件上显示当前便签的内容。

- 功能9：添加/删除提醒

1. 添加提醒：在便签编辑界面可以选择”添加提醒”选项，然后弹出一个对话框用于选择提醒的时间（包括月、日、星期、时、分），之后会在便签上显示一个闹钟的图标，标志提醒时间，到了提醒时间时，操作系统便会弹出一个对话框显示便签的内容并响铃，闹钟图表标志变为”已过期”。
2. 删除提醒：在便签编辑界面可以选择”删除提醒”选项。

- 功能10：分享
  在便签编辑页面上可以选择”分享”选项，之后可以将便签内容分享给GTask、QQ、微信等应用程序，其过程以纯文本格式进行。

- 功能11：识别便签文本

  1. 识别邮箱：小米便签可识别文本中邮箱，点击可复制相应的邮箱
  2. 识别网站：小米便签可识别文本中网站，点击可直接打开网站
  3. 识别电话：小米便签可识别文本中电话，可直接复制或者呼叫

1.2 小米便签的软件架构以及各个包和类的作用

1.2.1 小米便签的软件架构

小米便签整体上是一个层次性的体系结构，由界面层、业务层、模型层和数据层 4 层构成，每层包含若干个程序包。相邻层次的程序包之间存在交互。小米便签的体系结构图见图2。

- 界面层：根据表格（在前置动作部分）中出的主要功能，我们可以很轻松地将ui和 widget填入其中，而res包含了所有的xml文件和相对应图片，属于MVC架构里面 View部分，所以也应该填入界面部分。

- 业务层：根据MVC架构的逻辑，个人判断它应该担任是C（Control控制器）的角色：处理用户输入的信息。负责从视图读取数据，控制用户输入，并向模型发送数据，是应用程序中处理用户交互的部分。负责管理与用户交互交互控制。

- 模型层：该层负责对小米便签的单个便签项进行建模，提供了便签项的基本操作功能和对应业务, 并与数据层进行交互，以支持便签的创建、访问和修改。该层主要通过 model 程序包中的 Note 类、WorkingNote 类等加以实现。

- 数据层：该层负责组织和存储小米便签的相关数据,提供数据访问、数据合法性检验、数据访问缺失异常处理等功能和服务。该层主要通过 data 和 gtask.data 程序包加以实现。

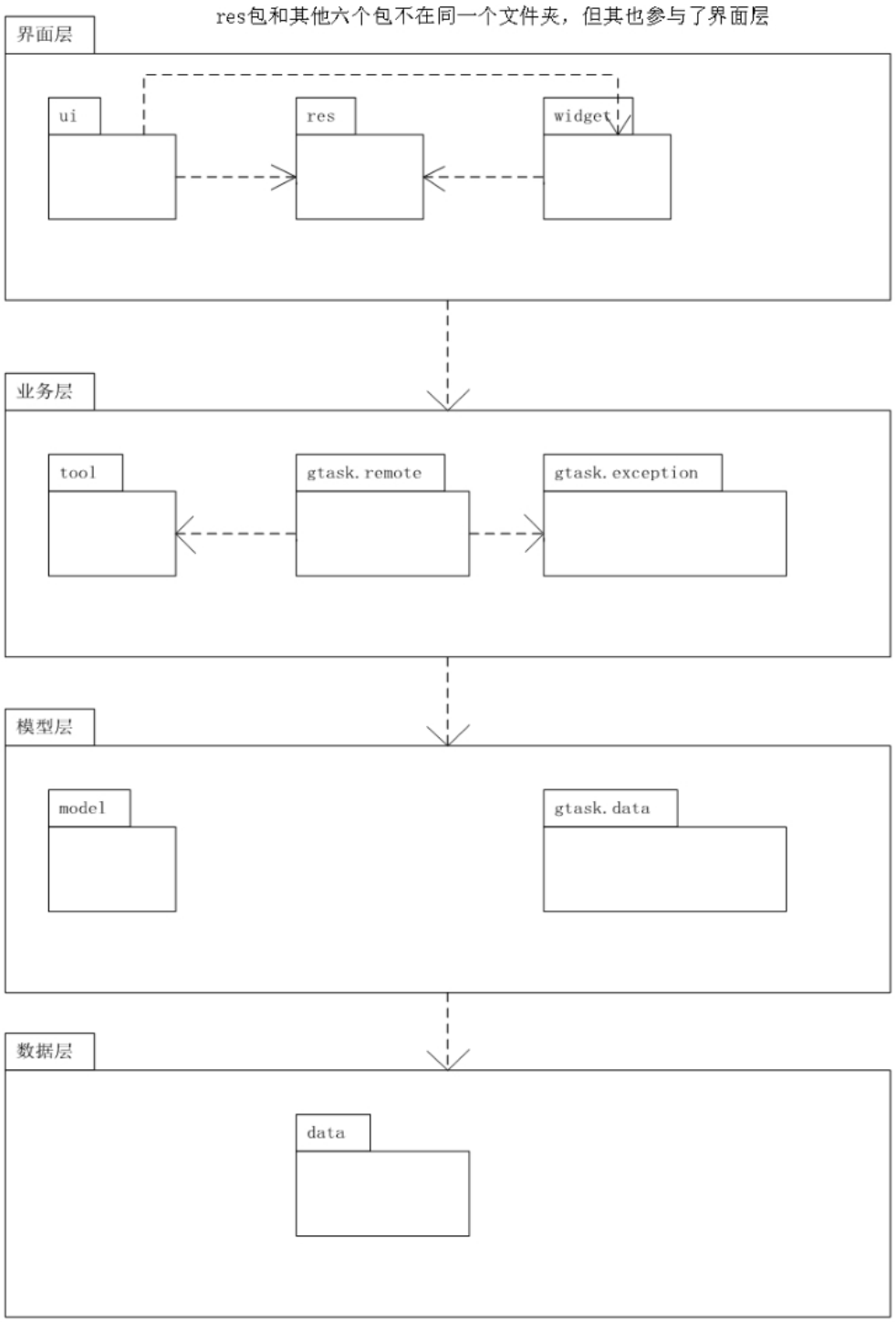


图 2: 小米便签的软件架构图

1.2.2 各个包内的关系图

该开源软件的类组织成7个子系统(6个源代码包，1个资源文件包)，这些子系统之间的关系如图2所示。

虚线表示包与包之间存在依赖关系。

Res包用于存放应用程序的资源文件。res是"resources"的缩写，这个包主要包含了应用

程序使用的各种资源，例如图像、布局文件、字符串、颜色等。这些资源文件在应用程序的开发和运行过程中起到重要的作用，可以提供界面展示、用户交互、美化等功能。通过使用这些资源文件，开发人员可以更方便地进行界面设计和开发。

Data包用于存储和处理备忘录应用程序的数据。通过将数据处理和管理相关的类放置在Data包中，可以使代码结构更加清晰，方便开发人员对数据进行操作和管理。

Model包用于存放数据模型相关的类，它主要负责定义应用程序中所需的数据结构、业务逻辑和数据操作的接口。通过将数据模型相关的类放置在model包中，可以使代码结构更加清晰，方便开发人员对数据进行管理和操作。

Widget包用于存放与应用程序桌面小部件（Widget）相关的类和方法。桌面小部件是一种能够在Android设备的主屏幕上显示内容的组件，它们可以提供快捷操作、实时信息展示等功能，为用户带来便利。

Tool包用于存放与工具类和辅助功能相关的类和方法。这个包通常用于处理一些通用的功能模块和工具方法，以提供开发人员更便捷的开发和实现。

Ui包用于存放与用户界面相关的类、布局和资源文件，比如闹钟提醒。通过将与用户界面相关的类和资源文件放置在ui包中，可以使代码结构更加清晰，方便开发人员对界面进行管理和操作。

Gtask包用于存放与数据更新，同步，检测异常相关的类和方法。这个包通常用于实现应用程序中数据更新，网络同步功能。

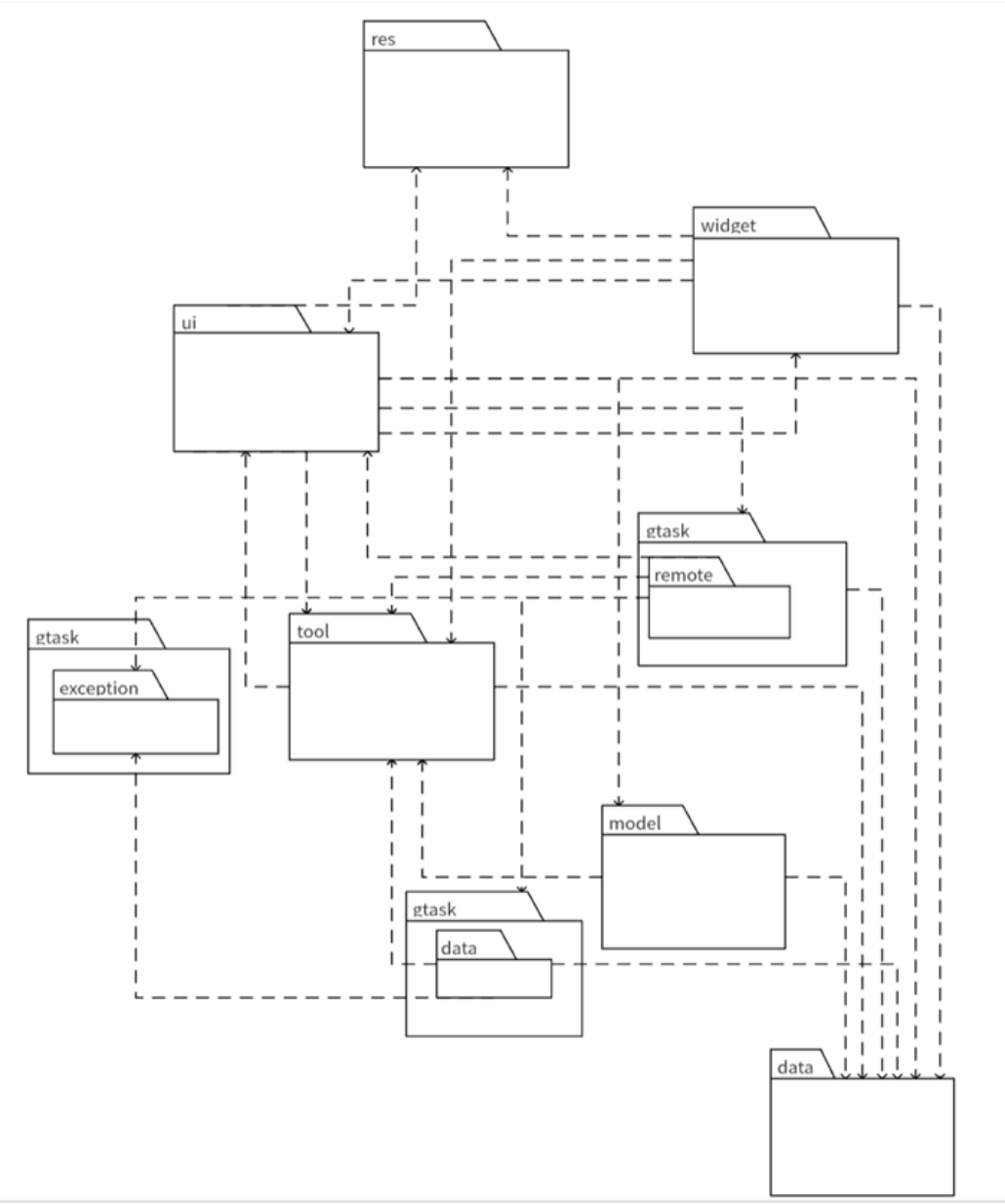


图 3: 小米便签的包间关系图

下面我将对每个类进行单独分析，以便更好地理解小米便签的代码结构。

- data包：
  如图4所示，Data包当中：
  Contact类 独 立， Notes类 和NotesDatabaseHelper类 关 联， NotesDatabaseHelper类和NotesProvider类是聚合关系(后者为整体，前者为部分)。

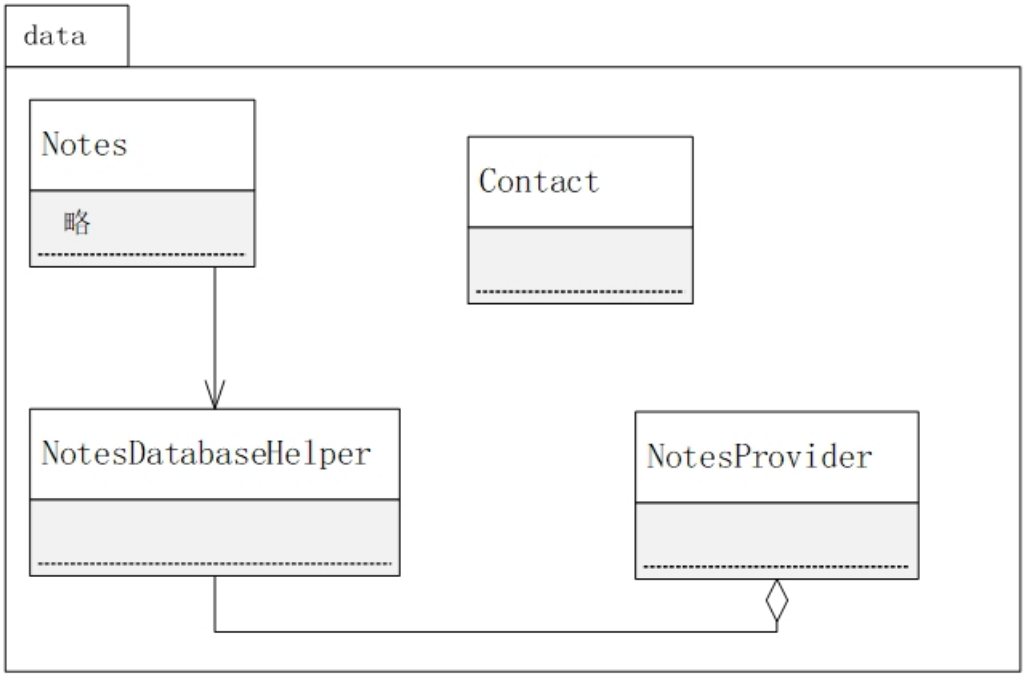


图 4: data子系统中的类图

- model包：
  如图5所示，Model包当中：
  Note类和WorkingNote类是聚合关系(Note为部分，WorkingNote为整体)。

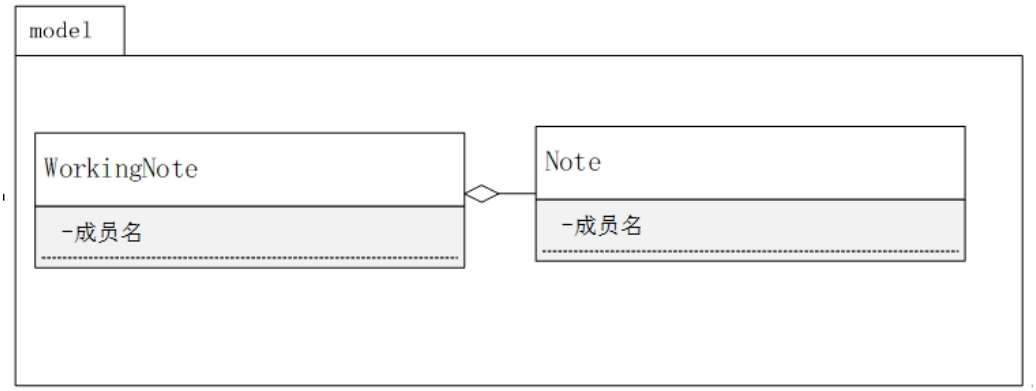


图 5: model子系统中的类图

- widget包：
  如图6所示，Widget包当中：

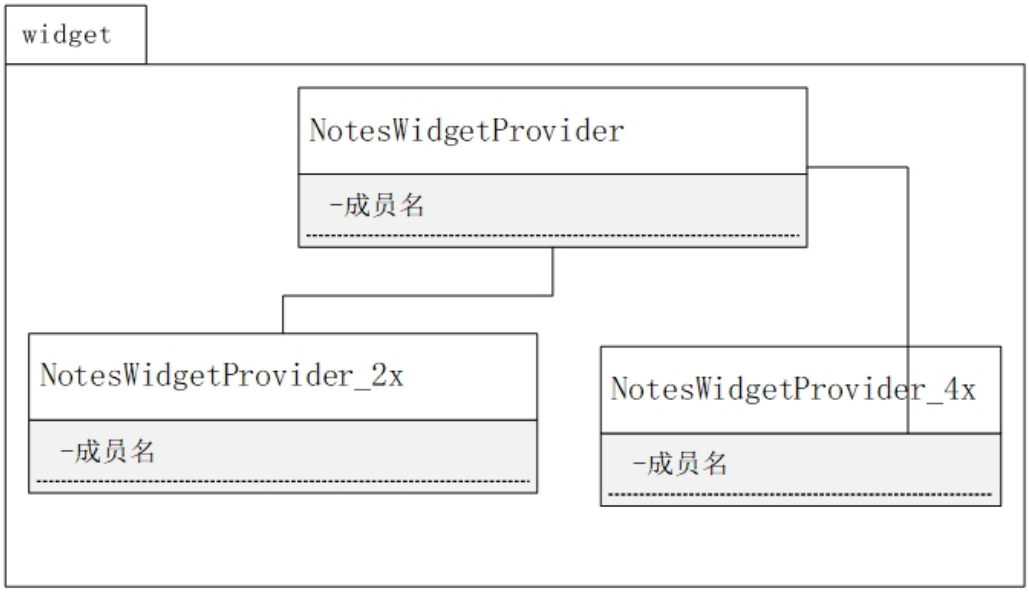


图 6: widget子系统中的类图

NoteWidgetProvider_4x类和NoteWidgetProvider类是泛化关系(前者是对后者的泛化或继承)。 NoteWidgetProvider_2x类和NoteWidgetProvider类是泛化关系(前者是对后者的泛化或继承)。

- tool包：

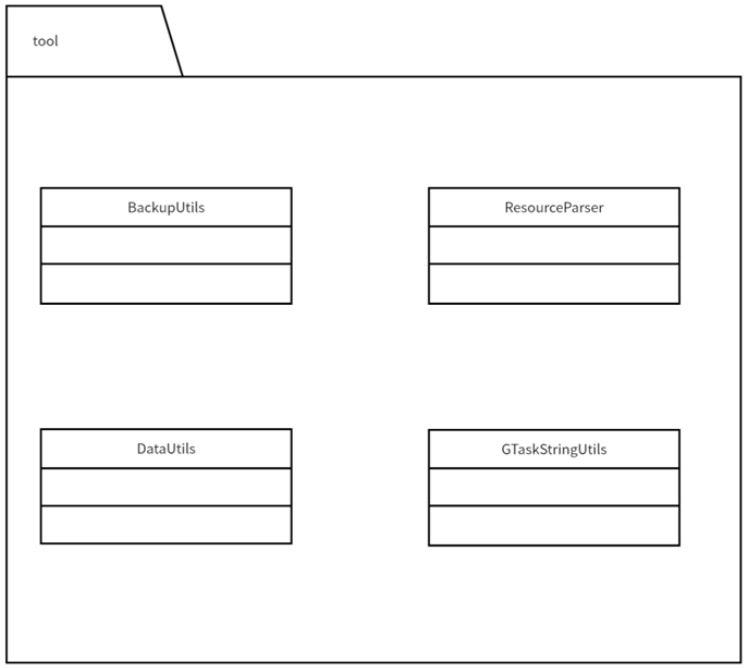


图 7: tool子系统中的类图

如图7所示，Tool包中：四个类独立并列，彼此之间没有明显的相关关系。

- ui包：

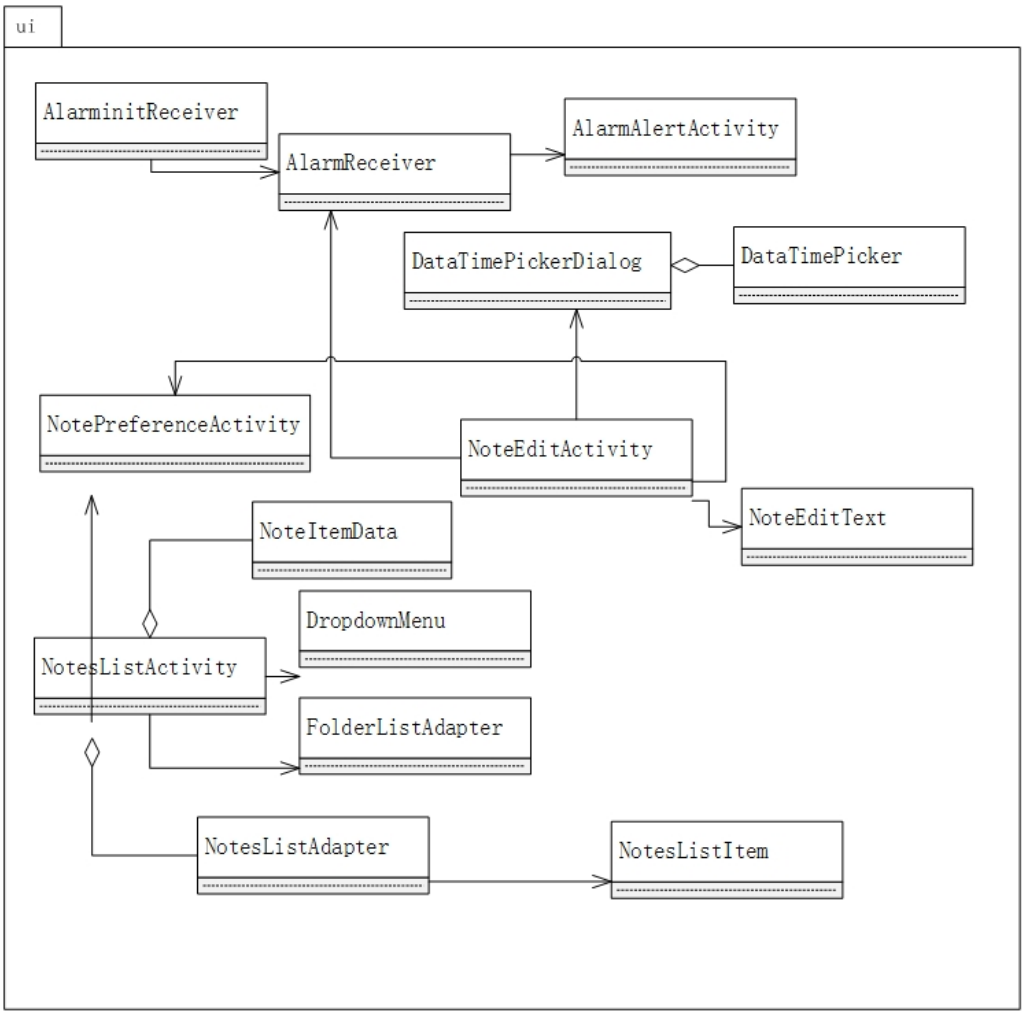


图 8: ui子系统中的类图

如图8所示，Ui包中：

1. DateTimePicker类和DateTimePickerDialog类是聚合关系(前者为部分，后者为整体)； NoteItemData类和NotesListActivity类是聚合关系(前者为部分，后者为整体)；NoteListAdapter类和NotesListActivity类是聚合关系(前者为部分，后者为整体)。
2. AlarmInitReceiver类关联AlarmReceiver类；
3. AlarmReceiver类关联AlarmAlertActivity类；
4. NoteEditActivity类关联AlarmReceiver类，DateTimePickerDialog类NotesPreferenceActivity类和NoteEditText类；
5. NoteEditActivity类和NotesListActivity类双向相互关联。
6. NoteItemData类关联NoteEditActivity类；
7. NotesListActivity类关联DropdownMenu类和FoldersListAdapter类；

8. NoteListAdapter类关联NoteItemData类和NotesListItem类。
9. NotesListItem类关联NoteItemData类。

- Gtask包：

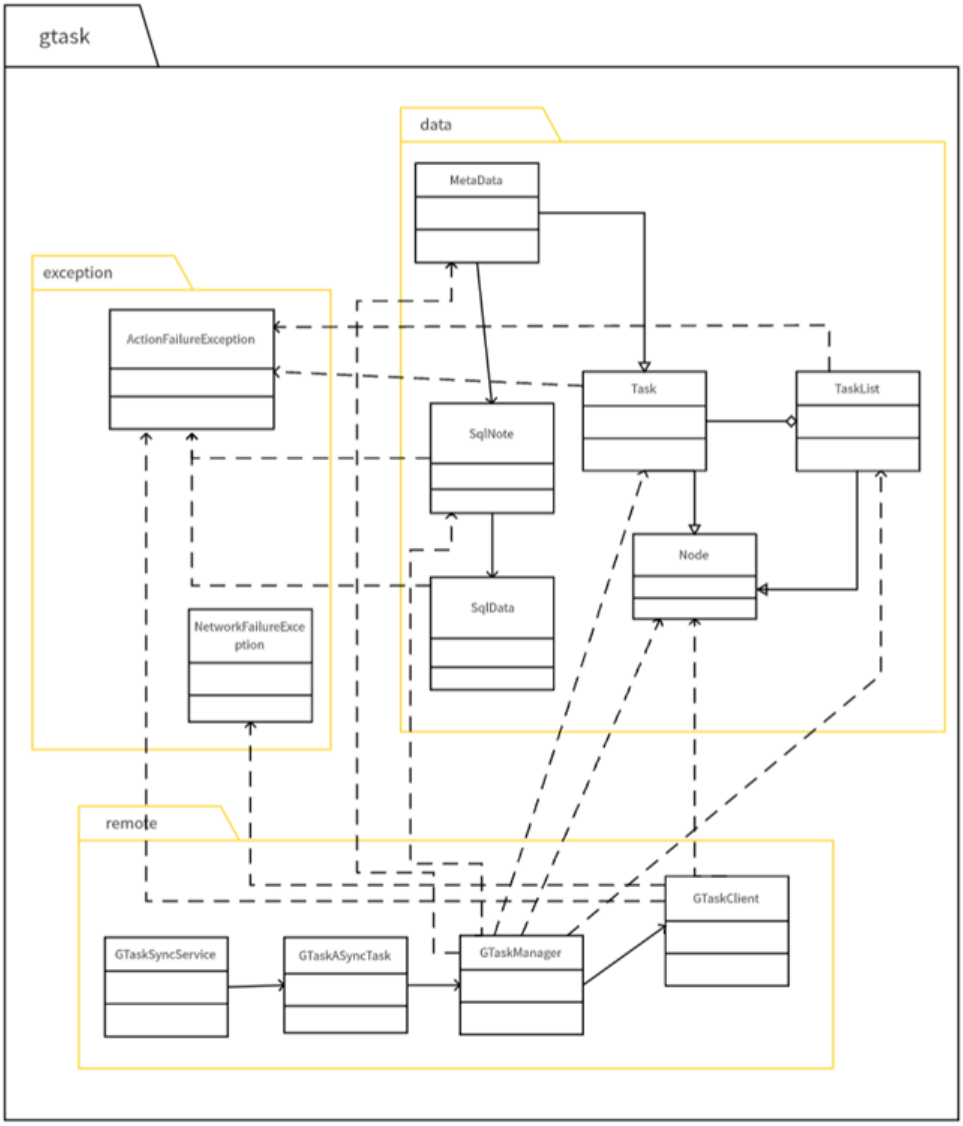


图 9: Gtask子系统中的类图

如图9所示，Gtask包中：

1. MetaData类是对Task类的泛化或继承； Task类是对Node类的泛化或继承；TaskList类是对Node类的泛化或继承；
2. Task类和TaskList类是聚合关系(前者为部分，后者为整体)；
3. MetaData类关联SqlNote类； SqlNote类关联SqlData类； GTaskSyncService类关联GTaskASyncTask类；GTaskASyncTask类关联GTaskManager类；GTaskManager类

关联GTaskClient类；

4. SqlNote类依赖ActionFailureException类；
5. SqlData类依赖ActionFailureException类；
6. TaskList类依赖ActionFailureException类；
7. Task类依赖ActionFailureException类；
8. GTaskManager类依赖MetaData类，SqlNote类，Task类，Node类和TaskList类；
9. GTaskClient类 依 赖ActionFailureException类， NetworkFailureException类和Node类；

1.3 软件功能与类间的对应关系

软件功能与类的实现对应关系见表1

表 1: 软件功能与类的实现对应关系

序号	功能名称	实现模块	主要实现方法
1	新建便签	NoteEditActivity	createNewNote() startActivityForResult() onOptionsItemSelected()
2	删除便签	NoteEditActivity	deleteCurrentNote() setNegativeButton() onOptionsItemSelected()
3	移动便签	DataUtils	moveNoteFolder() batchMoveToFolder()
4	新建文件夹	NoteListActivity	showCreateOrModifyFolderDialog() mContentResolver.update() mContentResolver.insert()
5	查看文件夹	NoteListActivity	openFolder() startAsyncNotesListQuery()
6	删除文件夹	NoteListActivity	deleteFolder() DataUtils.batchDeleteNotes() DataUtils.batchMoveToFolder
7	修改文件夹名称	NoteListActivity	showCreateOrModifyFolderDialog()
8	导出文本	NoteListActivity	exporNoteToText()
9	同步	GTaskSyncService	startSync()

下页继续

续表 1：软件功能与类的实现对应关系

序号	功能名称	实现模块	实现方法
10	搜索	NoteListActivity	onSearchRequest() startSearch()
11	修改字体大小	NoteEditActivity	onOptionsItemSelected()
12	修改便签背景颜色	WorkingNote	setBgColorId()
13	进入清单模式	WorkingNote	setCheckListMode()
14	发送到桌面	NoteEditActivity	sendToDestop()
15	添加提醒	NoteEditActivity	setAlertData()
16	删除提醒	WorkingNote	setAlertData()
17	识别电话号码	Contact	getContact()
18	分享	NoteEditActivity	getWorkingText(),sendTo()
19	保存便签	WorkingNote	saveNote()

这部分的工作主要是通过增加断点进行代码调试工作才确认类对应的功能。主要涉及到的功能界面有两个，对应的类为NotesListActivity和NoteEditActivity。

像某些功能对应两种实现方式的需要特别注意。例如：新建便签和删除便签，在主界面和便签菜单栏都可以实现。因此调试的方法才更有必要，只有通过断点调试才能确定其中的正确的对应关系。

另外还有一些功能，在AS虚拟机上并不能实现。例如：导出文本功能，该功能需要手机装备SD卡；又例如同步功能和分享功能，该功能需要进行联网并备有Google等平台的账号，用来进一步的操作。

有一些功能无法进行断点调试。例如：识别电话号码、邮箱功能，在便签内输入类似电话、邮箱的文字后，会在文字产生下划线，并附有超链接，可以进行点击操作，但在调试过程中并没有出现中断现象。

2 小米便签的代码标注

小组对6个包中共39个类的代码进行了标注，标注的代码行数共有3924行左右。具体的，这些代码注释的分部如表2所示。

表 2: 代码标注分布情况

序号	包名称	类名称	标注的代码行数
1	data	Contact	48行。在该类中分布在类的总体介绍(2,3)；引入库文件的说明(8,9)；类的静态变量说明(23,32-36)；方法中关键语句(39,44,49)
2	data	Notes	172行。在该类中分布在静态变量说明(23到49为int类型,之后为final string类型)
3	data	NotesDatabaseHelper	104行。在该类中分布在静态成员变量(18,21,25...)和对各种具体格式(SQL语句和uri接口)中
4	data	NotesProvider	47行。对各种变量、数据的sql语句进行了标注
5	gtask.data	MetaData	50行。在该类中分布在对元数据的定义、获取、传输、同步等方法中
6	gtask.data	Node	54行。对节点同步动作的常量、JSON动作、设置节点的相关内容方法进行了标注
7	gtask.data	SqlData	108行。对数据库管理类sqldata进行了说明，包括Cursor操作,属性说明,标签定义,以及重要方法sqlData构造函数的标注
8	gtask.data	SqlNote	127行。对静态变量进行了标注，对重要方法如获取、更新、记录变迁属性的功能进行了标注
9	gtask.data	Task	127行。对任务类task的成员变量tag等进行了标注，对JSON等方法进行了标注
10	gtask.data	TaskList	139行。任务列表类，继承自Node类。用于管理一组任务(Task)对象。对task任务列表的静态常量如标签、索引列表等静态常量做了标注。定义了类的构造函数，更新列表的重要方法的标注
11	gtask.exception	ActionFailureException	38行。该异常类是运行时异常的子类，用于表示操作失败的异常情况。它可以包含一个错误消息和导致异常的Throwable对象。解释了抛出异常的具体信息
12	gtask.exception	NetworkFailureException	34行。主要对网络操作失败异常类作用进行了注释，对构造函数进行了注释
13	gtask.remote	GTaskASyncTask	65行。对静态常量如同步通知栏ID等进行了标注，对重要方法如构造函数、发布、显示更新进度的方法等进行了标注

下页继续

续表 2：代码标注分布情况

序号	包名称	类名称	标注的代码行数
14	gtask.remote	GTaskClient	172行。 接下来的注释说明了GTaskClient类的作用， 它用于与Google任务服务进行远程交互，并提供了登录、获取任务列表、添加任务等操作的方法。注释还解释了类的各种属性，如日志标签、基础URL、请求URL、单例模式实例、HTTP客户端、 版本号、 登录状态和操作ID等。 对于GTaskClient的私有构造方法，注释说明了其用途是初始化各种属性。提供了getGTaskClient方法来获取GTaskClient的单例实例，并确保只创建一个实例
15	gtask.remote	GTaskManager	221行。说明了GTaskManager类的用途， 包括日志标签、 任务状态常量(如成功、 网络错误、 内部错误、同步进行中和同步已取消)。 注释描述了类的各种属性， 例如单例实例、关联的Activity对象、上下文对象、内容解析器、同步状态标记、任务列表和任务的HashMap、 元数据的HashMap等
16	gtask.remote	GTaskSyncService	62行。注释描述了类中定义的同步操作类型常量， 加后台同步、取消同步和无效操作。提供了服务广播的名称、标志以及广播的同步进度消息
17	model	Note	56行。描述了为数据库中添加新笔记生成一个新的笔记ID，创建一个新的笔记等方法

下页继续

续表 2：代码标注分布情况

序号	包名称	类名称	标注的代码行数
18	model	WorkingNote	230行。描述了类中的各种属性，如笔记对象、笔记的唯一标识符、笔记的内容、笔记的模式、提醒日期的时间戳、笔记最后修改日期的时间戳、笔记背景颜色的资源ID、小部件的ID和小部件的类型等。描述了构造函数、设置提醒日期、标记笔记为已删除、设置笔记背景颜色ID、等方法
19	tool	BackupUtils	120行。主要对备份工具类和内部类TextExport等进行了说明
20	tool	DataUtils	134行。描述了批量删除笔记的方法。描述了将笔记移动到指定文件夹的方法。描述了批量将笔记移动到指定文件夹的方法
21	tool	GtaskStringUtils	34行。描述了批量删除笔记的方法、将笔记移动到指定文件夹的方法、批量将笔记移动到指定文件夹的方法。获取删除系统文件夹外的所有用户文件夹数量的方法。描述了检查指定类型的笔记在数据库中是否可见的方法等等方法
22	tool	ResourceParser	72行。描述了定义笔记背景颜色的常量，获取默认笔记背景id、笔记背景资源头、提供获取不同背景资源的方法
23	ui	AlarmAlertActivity	150行。提醒界面
24	ui	AlarmInitReceiver	51行。后台消息接收器
25	ui	AlarmReceiver	39行。间接提醒接收器
26	ui	DateTimePicker	130行。设置提醒时间的部件
27	ui	DateTimePickerDialog	32行。设置提醒时间的对话提示界面
28	ui	DropdownMenu	26行。下拉菜单界面
29	ui	FoldersListAdapter	32行。文件夹列表链接器(链接数据库)
30	ui	NoteEditActivity	485行。便签编辑活动
31	ui	NoteEditText	120行。便签的文本编辑界面

下页继续

续表 2：代码标注分布情况

序号	包名称	类名称	标注的代码行数
32	ui	NoteItemData	39行。便签项数据
33	ui	NotesListActivity	311 主界面，实现处理文件夹列表的活动
34	ui	NotesListAdapter	20 便签列表链接器(链接数据库)
35	ui	NotesListItem	89 便签列表项
36	ui	NotesPreferenceActivity	187 便签同步的设置界面
37	widget	NoteWidgetProvider	20 桌面挂件
38	widget	NoteWidgetProvider_2x	12行。 NoteWidgetProvider,负 责 处理2x大小的小部件更新和其他操作
39	widget	NoteWidgetProvider_4x	12，4倍大小挂件
合计	6个包	39个类	大约3924行左右代码标注

3 小米便签的代码维护

3.1 维护的内容

对小米便签进行了如表3所示的维护，包括增加新的功能等等。

表 3: 维护内容列表

序号	维护类别	名称	描述
1	新增功能	设置界面背景	修改小米便签初始背景，并可以进行切换操作。
2	新增功能	欢迎界面	打开小米便签时，会有短暂的欢迎界面，2s 后界面自动消失，此时进入小米便签。
3	新增功能	登录密码	登录界面，在进入主界面时需要输入密码，设置了一个开放的密码锁保护隐私。可以进行密码的增添、删除和替换操作。
4	新增功能	翻译功能	在学习生活中，我们会经常使用英汉互译功能，将英汉互译功能添加到小米便签里面，可以方便我们在特定条件下的使用。
5	新增功能	插入图片	现在主流便签都有插入图片功能，这方便用户记录图片信息。
6	新增功能	统计字符个数	在主流编辑软件里面，可以统计带空格的字符数，这方便用户知道自己写作的篇幅。
7	新增功能	富文本功能	在主流的便签类软件都有富文本功能，可以方便用户对文本中的内容进行加粗、斜体、删除线以及高亮等。

下页继续

续表 3：维护内容列表

序号	维护类别	名称	描述
8	新增功能	朗读功能	在学习生活中，我们会经常使用英汉互译功能，将英汉互译功能添加到小米便签里面，可以方便我们在特定条件下的使用。
9	新增功能	私密模式	在便签的初始界面会显示便签的第一行内容，这样一定程度上容易泄露个人的隐私，进入私密模式后可以设定第一行内容，这样可以起到一定的保密作用。
10	新增功能	笔记编辑内搜索	在笔记编辑界面，可以进行搜索操作，如果没有搜索到内容会显示未找到相关内容，如果搜索到内容会高亮所有搜索到的内容。
11	新增功能	模板便签	每个用户都有特定的笔记编辑习惯，提前输入好模板，可以便捷用户记录便签。
12	新增功能	语音听写	在学习生活中，语音输入可以方便用户快速输入内容。
13	新增功能	语音合成	基于讯飞星火AI的语音合成功能，支持多语种、多声色和智能调控音调、节奏等丰富功能。
14	新增功能	对话式大模型	如今市面上主流的软件都内置了大模型，可以方便用户基于便签文本的上下文进行对话，获取帮助。
15	新增功能	撤回功能	用户在对便签进行编辑时，往往可能会误触删除一部分文本，撤回功能可以撤销误删除的操作。
16	新增功能	获取地理位置	一些软件需要获取地理位置权限才可以正常运行，为小米便签植入获取地理位置的功能，可以帮助用户及时定位自己的位置。

3.2 开源软件维护后所产生的设计

维护后的小米便签主要更改了ui包和增加了第三方API包，ui包和第三方API包中增加和更改类的类图见图10,维护后产生的体系架构图见图11。

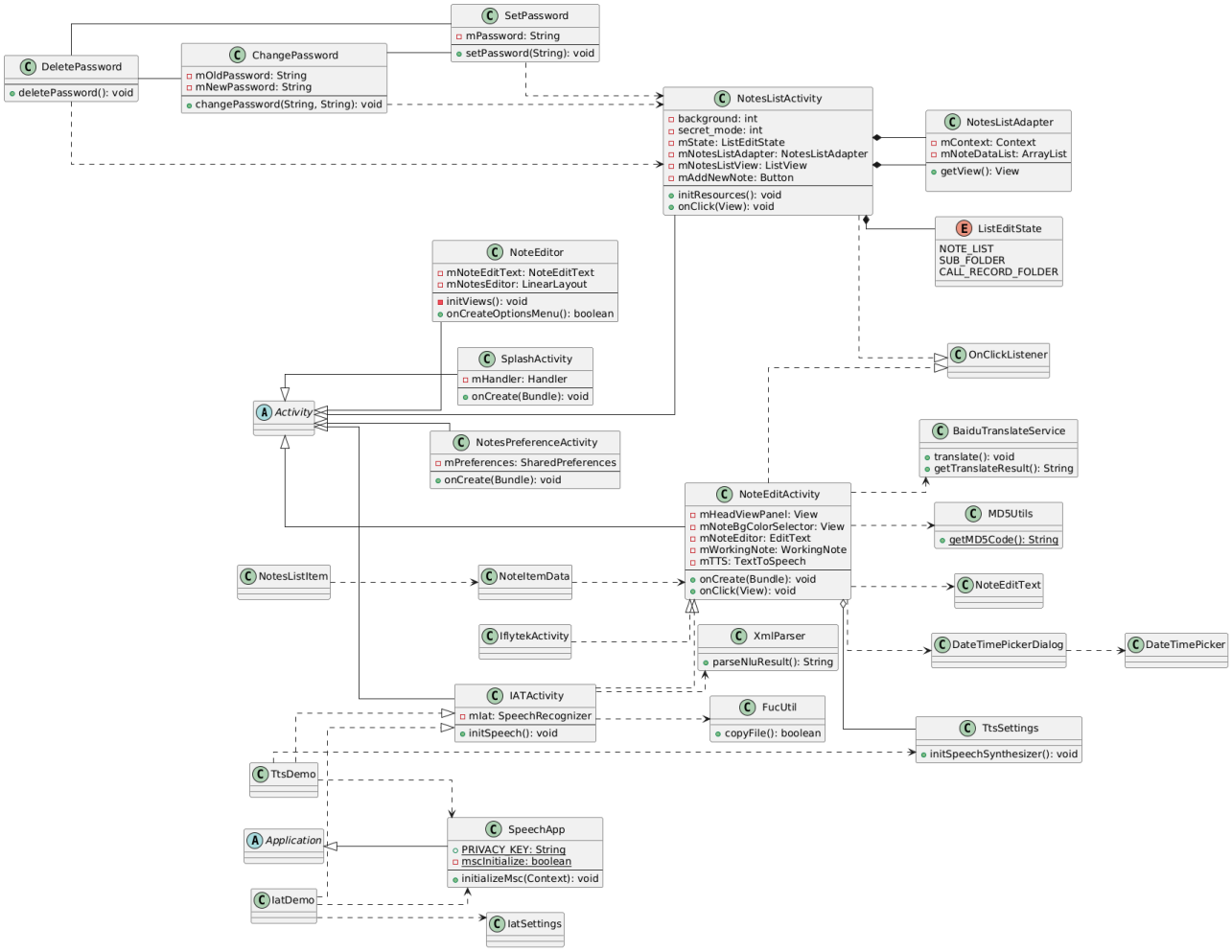


图 10: 维护后的小米便签的类图

3.2.1 类解释

此类图展示了项目中的主要类及其相互关系。项目采用了模块化设计，每个类负责特定的功能，同时通过接口和继承实现代码的复用和扩展性。以下是对主要类的详细解释：

1. Activity 类的继承层次结构

- **Activity**: 这是Android框架的核心组件，多个自定义Activity类继承自 `Activity`。
- **SplashActivity**: 实现了应用启动界面，持有一个 `Handler` 对象，用于处理异步任务和延时操作。
- **NotesListActivity**: 显示笔记列表的主界面，核心字段包括：

- background: 用于管理背景显示。
- secret_mode: 一个整数，用于标识是否处于加密模式。
- mState 和 mNotesListAdapter: 分别用于记录当前界面的状态和管理笔记列表的适配器。

它通过方法 initResources() 初始化资源，并处理按钮点击事件 (onClick(View) 方法)。

- **NotesPreferenceActivity**: 用于管理应用的设置选项，持有 SharedPreferences 对象以存储用户设置。
- **NoteEditActivity**: 提供对笔记内容的编辑功能，字段包括：
  - mHeadViewPanel 和 mNoteBgColorSelector: 分别用于界面头部视图和背景颜色选择器。
  - mNoteEditor 和 mWorkingNote: 分别表示编辑框和当前正在编辑的笔记。

它还支持语音合成功能（通过字段 mTTS: TextToSpeech 实现），并重写 onCreate(Bundle) 和 onClick(View) 方法处理界面初始化和点击事件。

2. 笔记列表相关类

- **NotesListItem** 和 **NoteItemData**: 这两个类分别用于表示笔记列表中的单个条目及其具体数据。通过对象之间的组合关系，NotesListItem 可以获取并管理 NoteItemData。
- **NotesListAdapter**: 用于适配笔记列表到用户界面。其主要字段包括 mContext（上下文对象）和 mNoteDataList（笔记列表数据数组）。它通过 getView() 方法为每个笔记条目生成对应的视图。

3. 密码管理功能类

- **SetPassword**: 负责设置密码，其字段包括 mPassword，并提供 setPassword(String) 方法设置新密码。
- **ChangePassword**: 允许用户更改密码，核心字段包括 mOldPassword 和 mNewPassword，方法 changePassword(String, String) 用于实现密码的更新。
- **DeletePassword**: 提供删除密码的功能，通过方法 deletePassword() 实现。

4. 第三方服务和工具类

- **BaiduTranslateService**: 封装百度翻译API的调用，提供方法 `translate()` 和 `getTranslateResult()` 实现翻译功能。
- **MD5Utils**: 提供生成MD5哈希值的工具方法 `getMD5Code()`，可能用于对密码或文件进行加密校验。
- **XmlParser**: 处理 XML 数据解析的工具类，方法 `parseXmlResult()` 用于解析返回的字符串数据。
- **FucUtil**: 提供通用文件管理功能，包括 `copyFile()` 方法。

5. 语音识别和语音合成相关类

- **IATActivity** 和 **IflytekActivity**: 处理语音输入功能，通过字段 `SpeechRecognizer` 初始化语音识别功能 (`initSpeech()` 方法)。
- **TtsDemo** 和 **TtsSettings**: 提供语音合成功能，其核心方法 `initSpeechSynthesizer()` 用于初始化语音合成引擎。
- **SpeechApp**: 用作全局应用类，管理与语音相关的初始化参数，如 `PRIVACY_KEY` 和 `msInitialize`。

6. 通用对话框类

- **DateTimePickerDialog**: 提供日期和时间选择功能，通常用于笔记的创建或修改时间管理。

7. 类之间的关系

- **继承关系**: 多个Activity类继承自Android框架的 `Activity` 类（如 `NotesListActivity`、`NoteEditActivity`）。
- **组合关系**: 类之间通过组合关系实现模块化，例如 `NotesListAdapter` 组合了 `NoteItemData` 以生成笔记列表视图。
- **关联关系**: `NoteEditActivity` 通过字段和方法调用与 `BaiduTranslateService`、`MD5Utils` 等类进行功能交互。
- **接口实现**: 部分类（如 `NotesListActivity` 和 `NoteEditActivity`）通过实现接口（如 `OnClickListener`）来处理用户交互事件。

总结 类图中的模块设计体现了单一职责原则（SRP），通过继承和组合的方式实现了代码复用与模块化。各个类的功能划分清晰，既包括核心的笔记管理功能（如 `NotesListActivity`、`NoteEditActivity`），又包含工具类和第三方API集成模块（如 `BaiduTranslateService` 和 `XmlParser`）。整体架构设计合理，具有良好的可扩展性和可维护性。

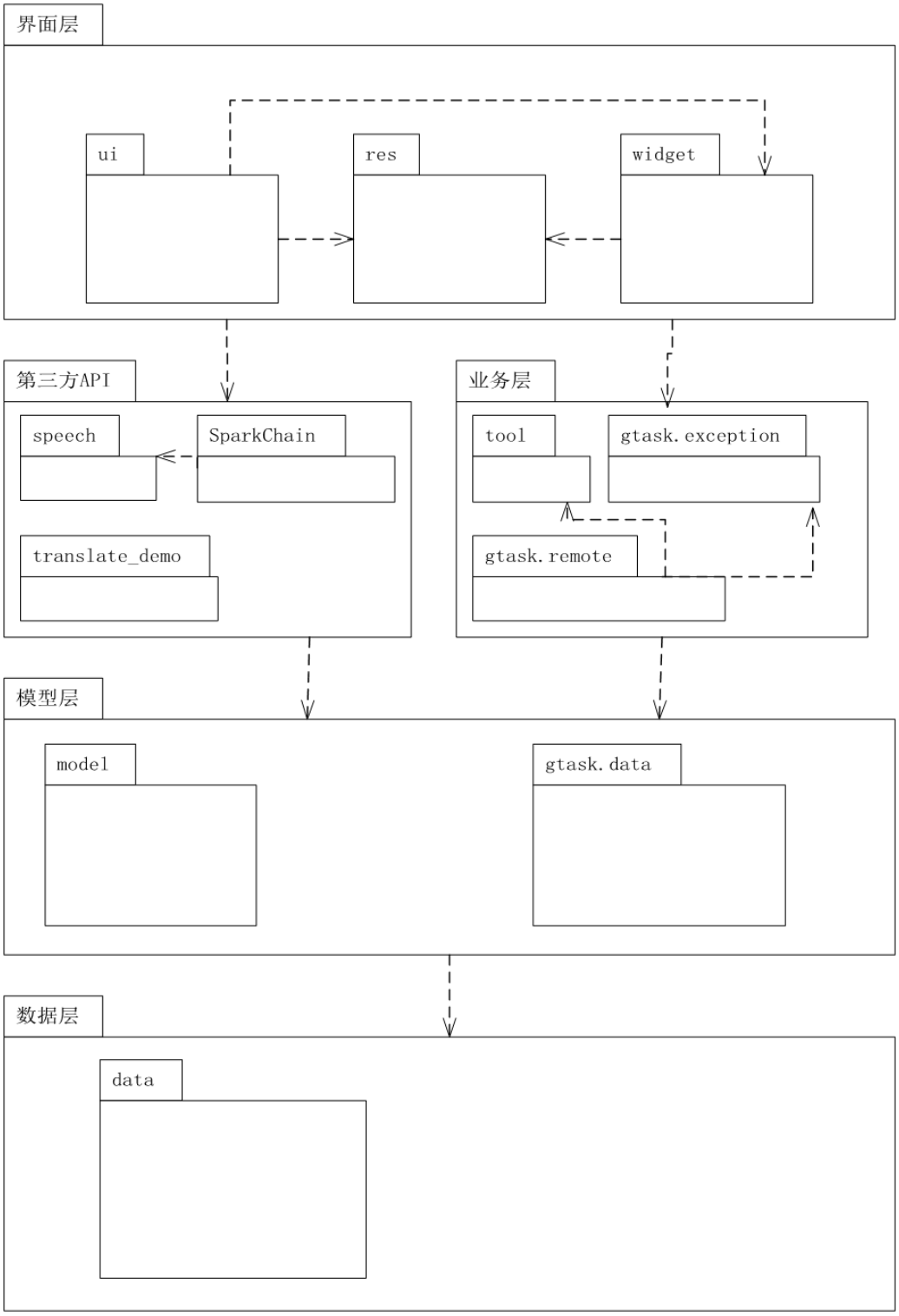


图 11: 维护后的小米便签的架构图

3.2.2 架构解释

该项目采用分层架构设计，各层职责明确，模块化程度高。这种架构提高了代码的可读性、可维护性和可扩展性。以下是各层的详细解释：

1. 界面层（UI Layer） 界面层负责用户界面的显示和交互，主要包含以下模块：

- **ui**: 实现具体的界面功能，如 `Activity`、`Fragment`，用于响应用户操作。
- **res**: 包含资源文件（如布局文件、图片、字符串资源等），通过 `R` 类引用。
- **widget**: 提供自定义控件或组件，用于增强 UI 的可复用性或满足特定的 UI 需求。

连接：界面层通过调用业务层获取数据并渲染到 UI 中，`widget` 模块可能直接与业务层交互以更新自定义组件的行为。

2. 第三方 API 层（Third-party API Layer） 该层主要负责与外部第三方服务或库的交互，包含以下模块：

- **speech**: 封装与语音识别或语音合成相关的第三方 API。
- **translate_demo**: 调用第三方翻译服务（如 Google Translate 或其他翻译 API）。
- **SparkChain**: 处理区块链相关操作的模块，可能用于数据记录、身份验证或交易管理。

连接：第三方 API 层通过封装对外部 API 的调用，为业务层提供统一接口，避免业务逻辑直接依赖第三方库。

3. 业务层（Business Layer） 业务层负责实现核心的应用逻辑和功能，主要包含以下模块：

- **tool**: 提供通用工具类和辅助功能，例如数据格式转换、日志工具等。
- **gtask.remote**: 处理远程通信，例如通过网络请求与后端服务器交互（可能使用 `Retrofit` 或 `OkHttp`）。
- **gtask.exception**: 用于定义和处理全局异常，确保应用中错误管理的一致性。

连接：业务层从第三方 API 层获取数据，处理后传递给界面层或模型层。同时，`exception` 模块与 `remote` 模块紧密结合，统一处理网络或业务逻辑中的异常。

4. 模型层（Model Layer） 模型层定义和操作核心数据结构，主要包含以下模块：

- **model**: 定义应用中的核心数据模型，例如用于表示用户、商品或订单的 Java 或 Kotlin 类。
- **gtask.data**: 管理本地数据或缓存，例如 `SQLite` 数据库、`SharedPreferences` 或 `Room` 框架。

连接：模型层为业务层和界面层提供统一接口，通过封装数据操作实现与其他层的解耦。

5. 数据层（Data Layer） 数据层负责存储和管理应用程序的数据，包含以下模块：

- **data**: 封装了数据的持久化操作，例如数据库管理（如 SQL、`Room ORM`）或文件系统管理。

连接：数据层通过模型层提供数据接口，将数据存储到本地或从本地读取，并向业务层提供支持。

总体架构特性总结

- **分层解耦：**每一层职责明确，界面层专注于用户交互，业务层专注于逻辑处理，数据层专注于数据管理。
- **模块化设计：**各层进一步划分为多个模块（如 `tool`、`gtask.remote`），增强了代码的复用性和维护性。
- **良好的扩展性：**通过第三方 API 层封装，便于未来替换或增加新的第三方服务（如增加新的翻译 API）。
- **异常管理：**在业务层中通过 `gtask.exception` 模块统一处理异常，提升了应用的稳定性。
- **清晰的数据流向：**从数据层到界面层，数据流向清晰明确，各层通过特定接口交互，避免直接依赖。

适用场景和优势

- 适用于中大型 Android 项目，尤其是需要与多个第三方服务交互的场景（如语音识别、多语言翻译等）。
- 便于团队协作：不同开发者可以分别专注于某一层的开发，减少冲突和耦合。
- 易于测试：每一层都可以独立进行单元测试，例如测试业务逻辑或第三方 API 的封装。

3.2.3 界面设计

小米便签应用的界面设计从提供的两张截图来看，简洁美观，同时功能强大，下面针对截图中的界面设计进行详细分析。

主界面

- **功能菜单设计**：主界面的功能菜单通过右上角的三点图标展开，包含多个功能选项，例如“新建文件夹”、“导出文本”、“同步”、“设置”等。这种菜单设计在布局上便于快速访问，同时保持界面整洁。菜单中功能选项排列紧凑，功能分布清晰，例如数据管理类功能（如导出文本、同步）和个性化设置（如切换图片背景）被明确区分。
- **便签内容列表**：主界面中部为便签的内容列表，每个便签以黄色背景块区分，突出文字内容，同时便于用户快速预览和区分各个便签。列表设计采用块状卡片布局，用户能够直观地浏览每条便签的内容摘要，便签标题也帮助用户快速聚焦。
- **背景设计**：背景为星空主题，搭配柔和的配色，营造出宁静的使用氛围，与便签功能的“记录灵感”定位相符。同时，下方的“写便签”按钮以皮革质感样式设计，增强界面的现代与复古结合的美感。

内容详情界面

- **功能扩展菜单**：右上角的功能菜单通过三点图标展开，提供针对便签内容的更多功能选项，包括“添加备注”、“删除”、“字体大小调整”、“进入清单模式”、“分享”等。这些功能操作集中在内容详情界面，便于用户针对当前便签进行管理或深度编辑。功能菜单的布局整齐清晰，同时与主界面功能菜单保持一致，增强了界面的统一性。
- **内容显示区域**：便签内容显示在黄色背景区域中，文字清晰易读。上方还提供了“朗读”、“翻译”功能按钮，扩展了便签的应用场景，例如用户可通过语音或翻译功能处理多语言文本。内容显示的黄色背景设计柔和，既减少视觉疲劳，又能帮助用户专注于记录的内容。
- **底部快捷功能按钮**：界面底部设计有“星火”和“听写”按钮，这些功能可能涉及语音识别或与外部服务的集成，为用户提供便捷入口。按钮布局简洁，保持了整体界面的干净与直观。

界面整体布局分析

- 两个界面均采用了顶部菜单、中部内容、底部功能的三段式布局，保证了使用逻辑的一致性，同时最大化利用屏幕空间。
- 背景设计温馨，功能菜单和操作按钮布局简洁明了，为用户提供了流畅的操作体验。

- 便签内容区域设计醒目，以黄色背景突出文字，适合内容阅读和记录场景。

图 12: 维护后的小米便签的界面设计

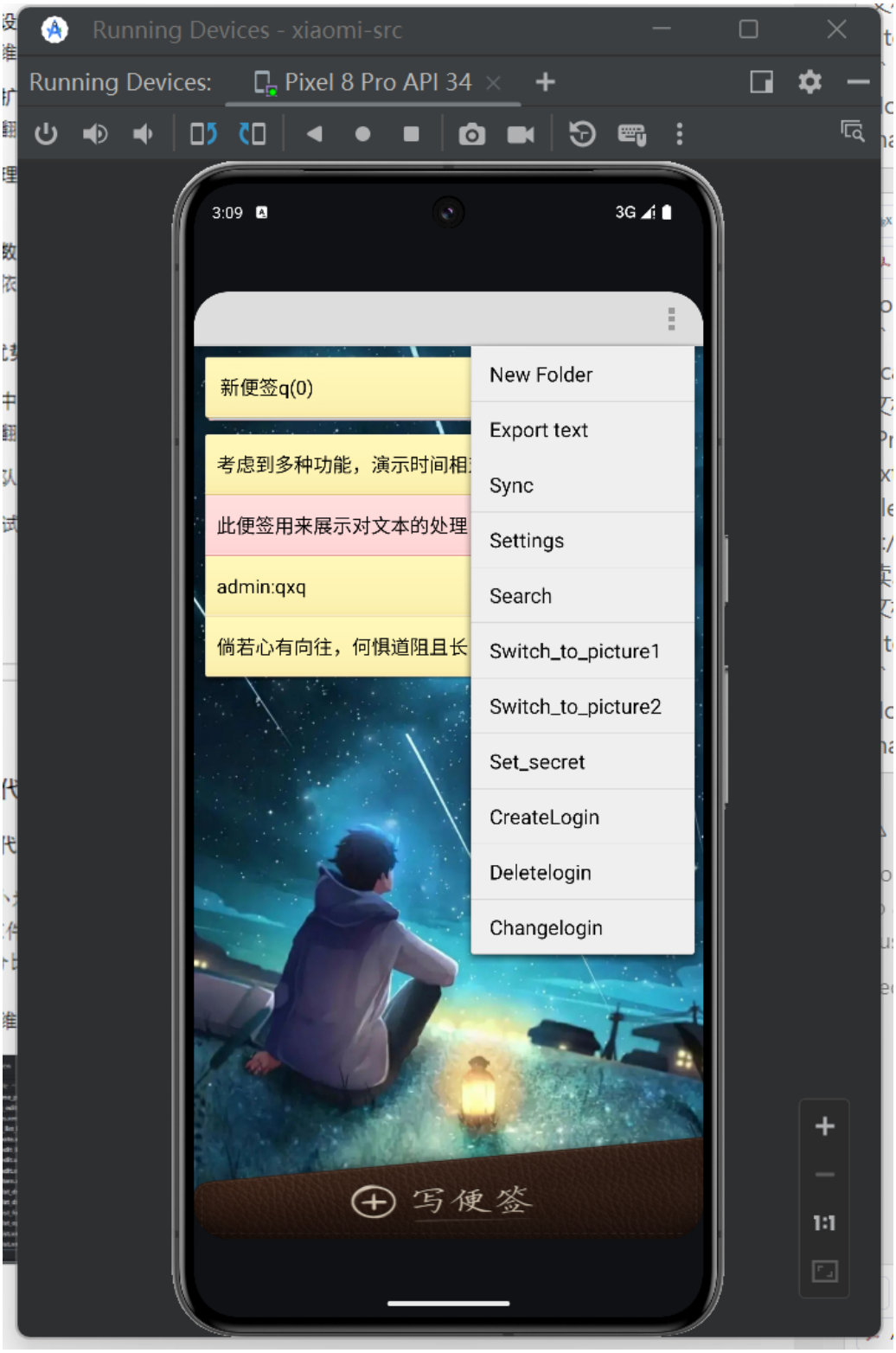


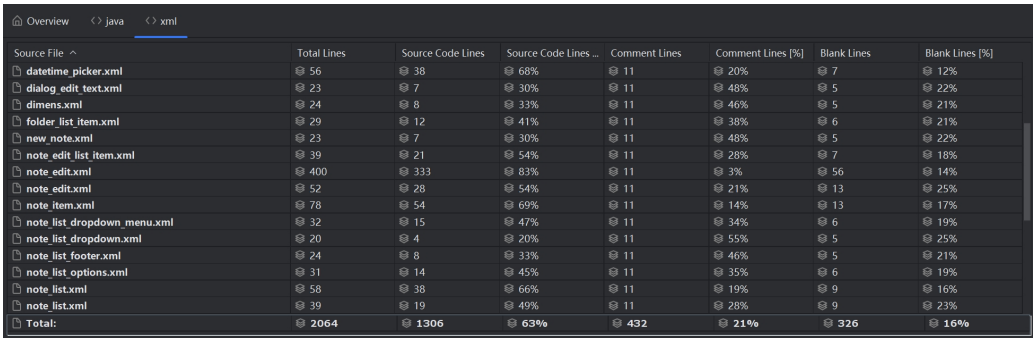
图 13: 维护后的小米便签的界面设计

3.3 维护代码数量以及质量情况

3.3.1 维护代码数量

在完成小米便签的维护工作之后，我们使用一个插件统计小米便签项目的代码数量情况，从java文件和xml文件两个方面进行统计，分别从总行数、源代码数量、注释行数、注释行数的百分比、空行数、空行数的百分比进行统计。

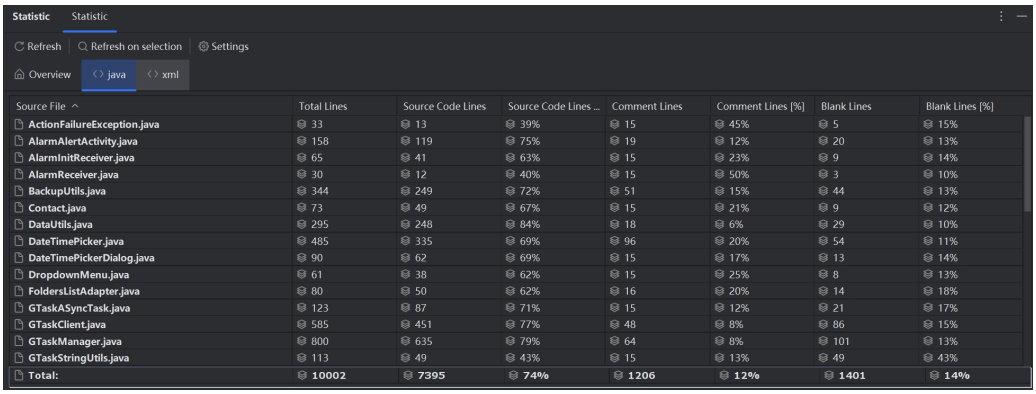
1. 首先是维护之前的数据，见图14、图15。



Overview | java | xml

Source File	Total Lines	Source Code Lines	Source Code Lines ...	Comment Lines	Comment Lines [%]	Blank Lines	Blank Lines [%]
datetime_picker.xml	56	38	68%	11	20%	7	12%
dialog_edit_text.xml	23	7	30%	11	48%	5	22%
dimens.xml	24	8	33%	11	46%	5	21%
folder_list_item.xml	29	12	41%	11	38%	6	21%
new_note.xml	23	7	30%	11	48%	5	22%
note_edit_list_item.xml	39	21	54%	11	28%	7	18%
note_edit.xml	400	333	83%	11	3%	56	14%
note_edit.xml	52	28	54%	11	21%	13	25%
note_item.xml	78	54	69%	11	14%	13	17%
note_list_dropdown_menu.xml	32	15	47%	11	34%	6	19%
note_list_dropdown.xml	20	4	20%	11	55%	5	25%
note_list_footer.xml	24	8	33%	11	46%	5	21%
note_list_options.xml	31	14	45%	11	35%	6	19%
note_list.xml	58	38	66%	11	19%	9	16%
note_list.xml	39	19	49%	11	28%	9	23%
Total:	2064	1306	63%	432	21%	326	16%

图 14: 维护之前xml文件数量



Statistic | Statistic

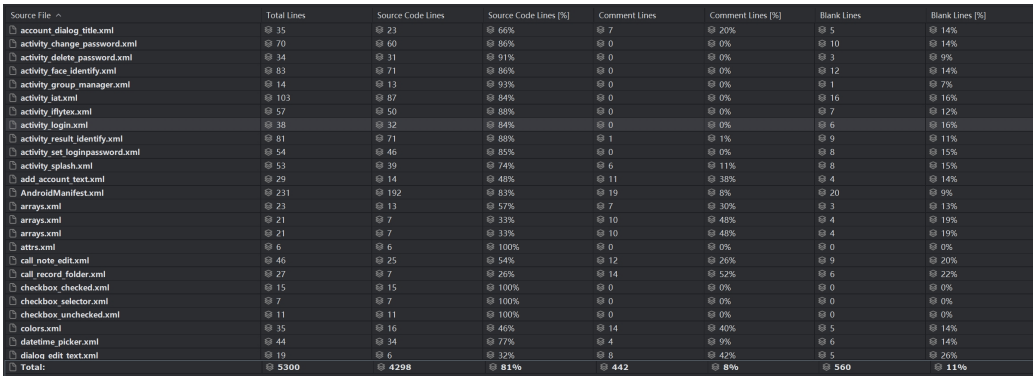
Refresh | Refresh on selection | Settings

Overview | java | xml

Source File	Total Lines	Source Code Lines	Source Code Lines ...	Comment Lines	Comment Lines [%]	Blank Lines	Blank Lines [%]
ActionFailureException.java	33	13	39%	15	45%	5	15%
AlarmAlertActivity.java	158	119	75%	19	12%	20	13%
AlarmInitReceiver.java	65	41	63%	15	23%	9	14%
AlarmReceiver.java	30	12	40%	15	50%	3	10%
BackupUtils.java	344	249	72%	51	15%	44	13%
Contact.java	73	49	67%	15	21%	9	12%
DataUtils.java	295	248	84%	18	6%	29	10%
DateTimePicker.java	485	335	69%	96	20%	54	11%
DateTimePickerDialog.java	90	62	69%	15	17%	13	14%
DropdownMenu.java	61	38	62%	15	25%	8	13%
FoldersListAdapter.java	80	50	62%	16	20%	14	18%
GTaskASyncTask.java	123	87	71%	15	12%	21	17%
GTaskClient.java	585	451	77%	48	8%	86	15%
GTaskManager.java	800	635	79%	64	8%	101	13%
GTaskStringUtils.java	113	49	43%	15	13%	49	43%
Total:	10002	7395	74%	1206	12%	1401	14%

图 15: 维护之前java文件数量

2. 其次是维护之后的数据，见图16、图17。



Source File	Total Lines	Source Code Lines	Source Code Lines [%]	Comment Lines	Comment Lines [%]	Blank Lines	Blank Lines [%]
account_dialog_title.xml	35	23	66%	7	20%	5	14%
activity_change_password.xml	70	60	86%	0	0%	10	14%
activity_delete_password.xml	34	31	91%	0	0%	3	9%
activity_face_identify.xml	83	71	86%	0	0%	12	14%
activity_group_manager.xml	14	13	93%	0	0%	1	7%
activity_iat.xml	103	87	84%	0	0%	16	16%
activity_iflytex.xml	57	50	88%	0	0%	7	12%
activity_login.xml	38	32	84%	0	0%	6	16%
activity_result_identify.xml	81	71	88%	1	1%	9	11%
activity_set_loginpassword.xml	54	46	85%	0	0%	8	15%
activity_splash.xml	53	39	74%	6	11%	8	15%
add_account_text.xml	29	14	48%	11	38%	4	14%
AndroidManifest.xml	231	192	83%	19	8%	20	9%
arrays.xml	23	13	57%	7	30%	3	13%
arrays.xml	21	7	33%	10	48%	4	19%
arrays.xml	21	7	33%	10	48%	4	19%
attrs.xml	6	6	100%	0	0%	0	0%
call_note_edit.xml	46	25	54%	12	26%	9	20%
call_record_folder.xml	27	7	26%	14	52%	6	22%
checkbox_checked.xml	15	15	100%	0	0%	0	0%
checkbox_selector.xml	7	7	100%	0	0%	0	0%
checkbox_unchecked.xml	11	11	100%	0	0%	0	0%
colors.xml	35	16	46%	14	40%	5	14%
datetime_picker.xml	44	34	77%	4	9%	6	14%
dialog_edit_text.xml	19	6	32%	8	42%	5	26%
Total:	5300	4298	81%	442	8%	560	11%

图 16: 维护之后xml文件数量

Overview | java | properties | txt | xml

Source File	Total Lines	Source Code Lines	Source Code Lines [%]	Comment Lines	Comment Lines [%]	Blank Lines	Blank Lines [%]
ActionFailureException.java	47	14	30%	26	55%	7	15%
AlarmAlertActivity.java	196	119	61%	59	30%	18	9%
AlarmInitReceiver.java	71	41	58%	21	30%	9	13%
AlarmReceiver.java	30	12	40%	16	53%	2	7%
BackupUtils.java	440	249	57%	142	32%	49	11%
BaiduTranslateService.java	24	11	46%	12	50%	1	4%
ChangePassword.java	68	63	93%	0	0%	5	7%
Contact.java	80	49	61%	21	26%	10	12%
DataUtils.java	396	248	63%	118	30%	30	8%
DateTimePicker.java	583	335	57%	190	33%	58	10%
DateTimePickerDialog.java	121	62	51%	47	39%	12	10%
DeletePassword.java	63	56	89%	0	0%	7	11%
DropdownMenu.java	73	38	52%	28	38%	7	10%
FaceIdentifyActivity.java	467	325	70%	81	17%	61	13%
FaceRect.java	23	14	61%	4	17%	5	22%
Total:	18665	11872	64%	4703	25%	2090	11%

图 17: 维护之后java文件数量

3. 最后将java文件和xml文件的数据绘制成图，见图18、图19、图20、图21。

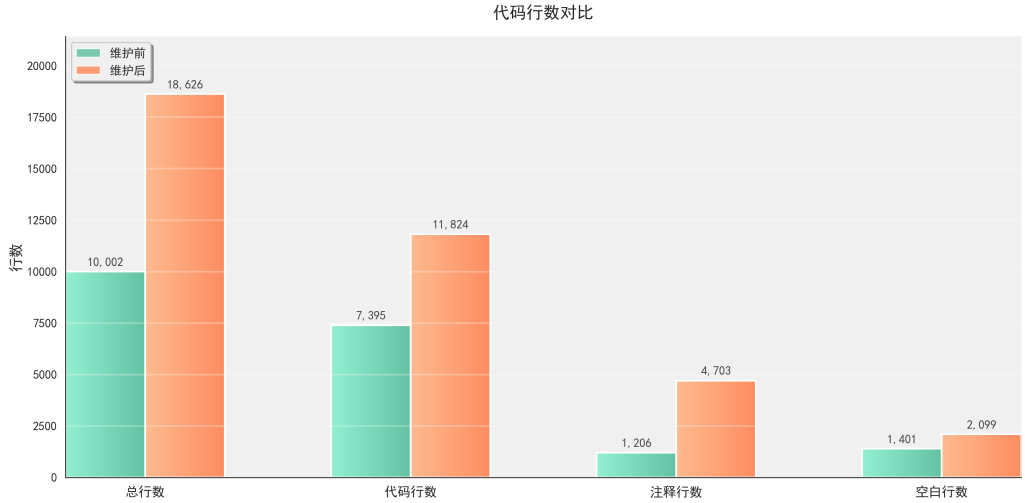


图 18: 维护前后java文件数量对比——柱状图

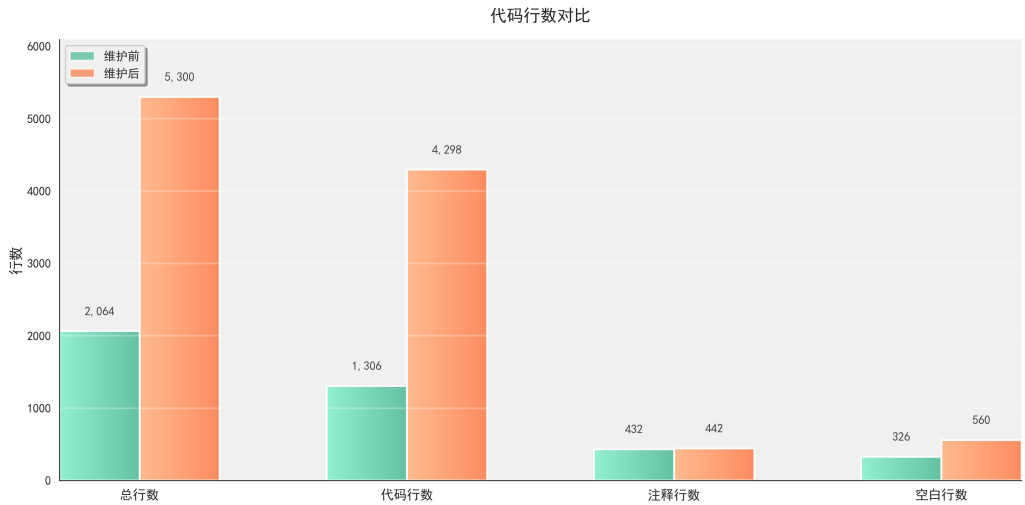


图 19: 维护前后xml文件数量对比——柱状图

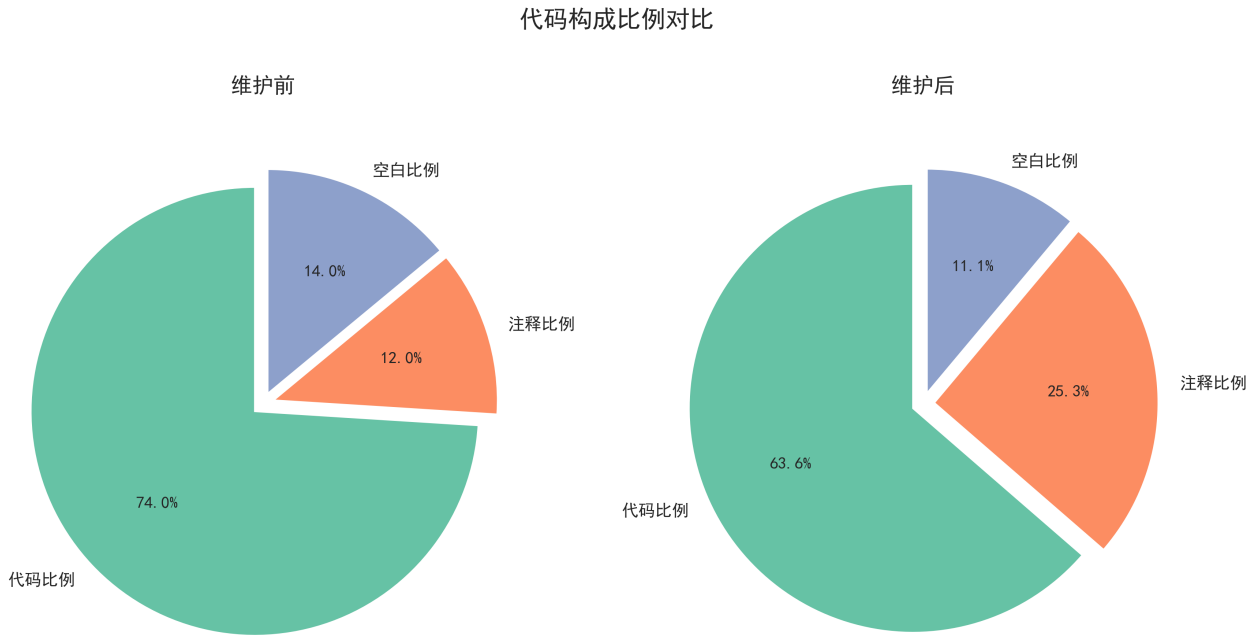


图 20: 维护前后java文件数量对比——饼状图

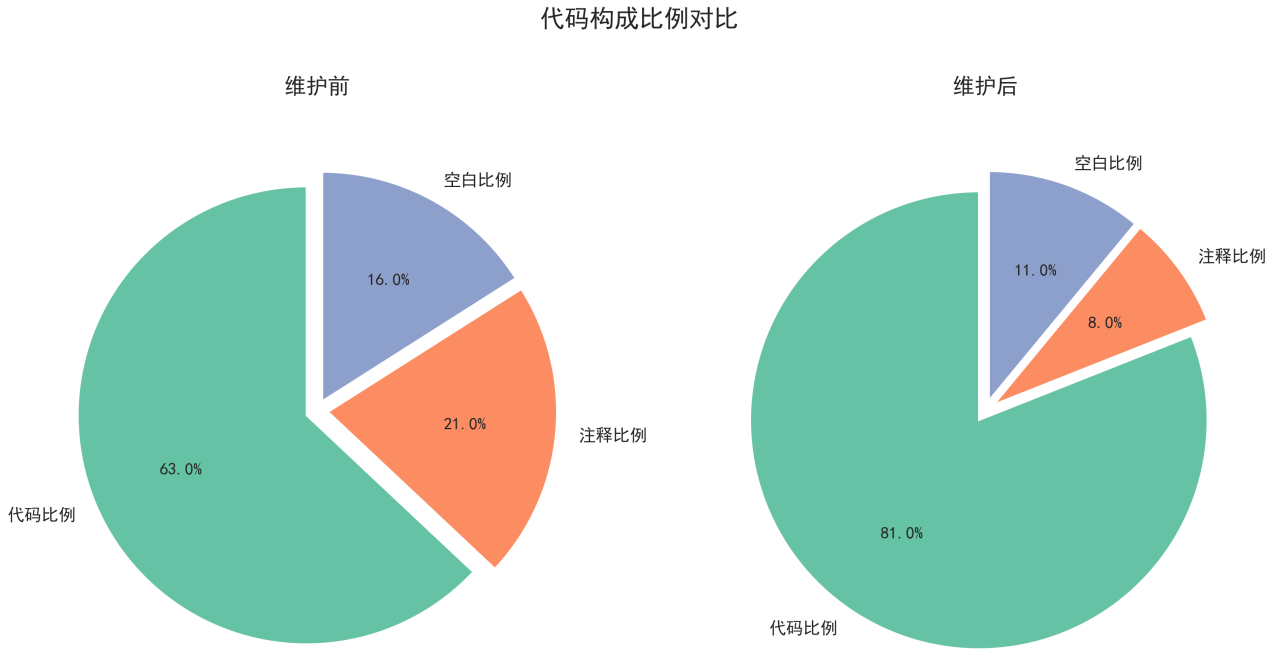


图 21: 维护前后xml文件数量对比——饼状图

4. 最后将项目代码数量汇总，我们维护增加的代码数量接近7000行，注释的代码行数有3500行左右，见图22、图23。

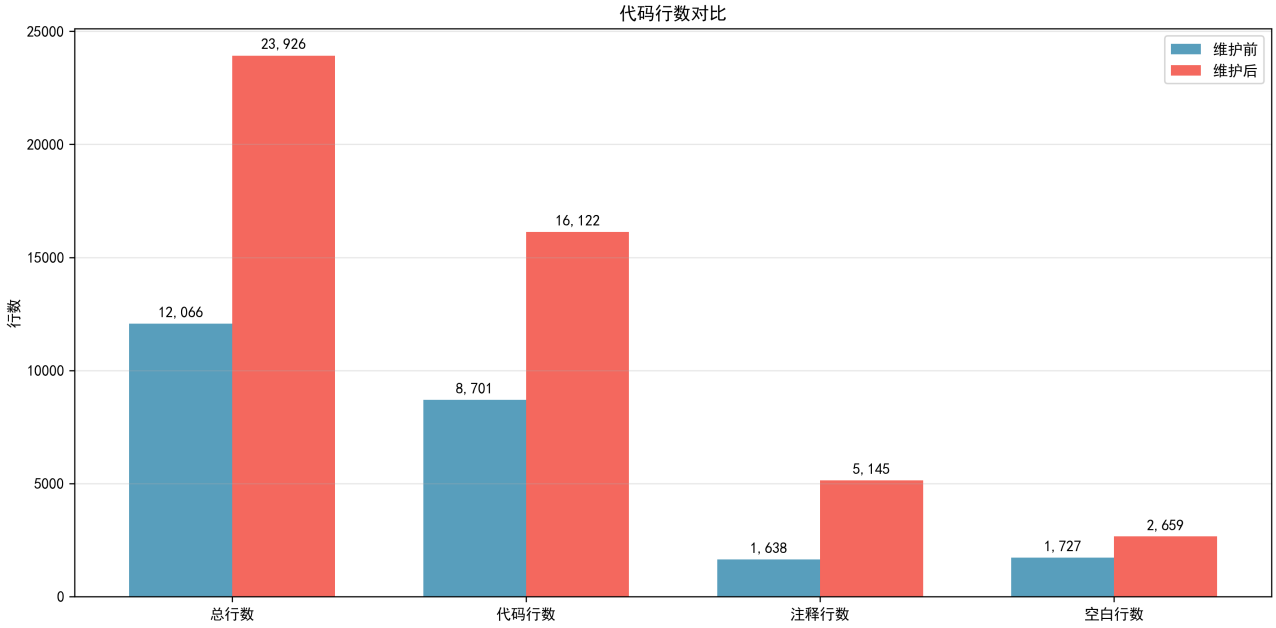


图 22: 维护前后代码数量对比——柱状图

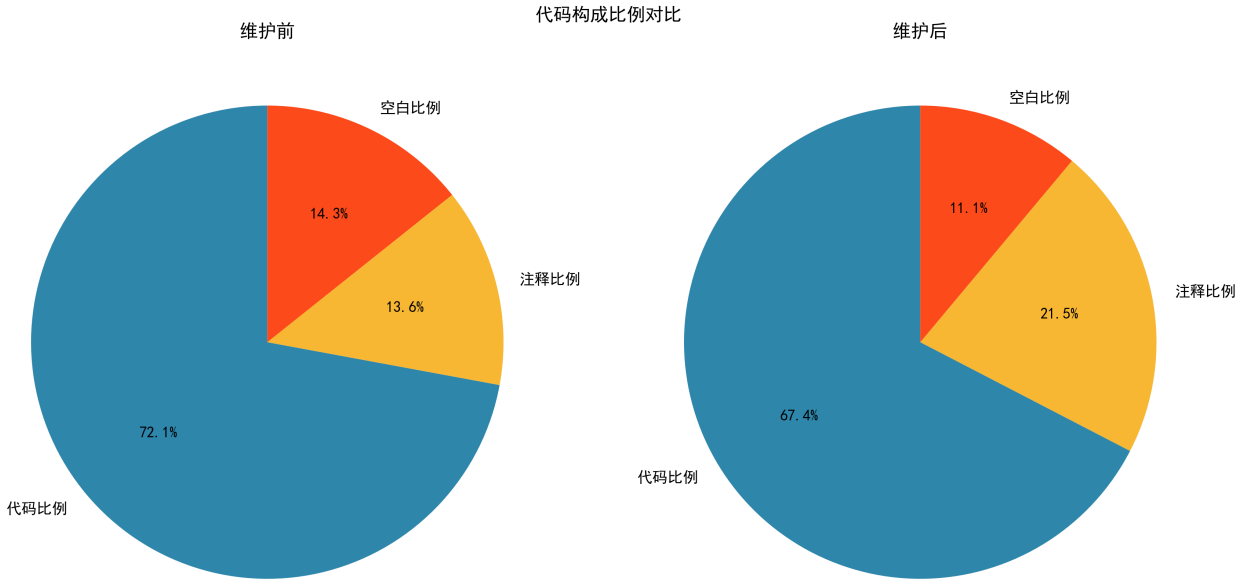


图 23: 维护前后代码数量对比——饼状图

3.3.2 各功能代码规模表

在完成代码维护之后，我们小组对维护的代码进行了统计，展示了各功能代码规模，统计结果见表4，如下所示：

表 4: 各功能代码规模表

序号	维护类别	名称	受影响的代码行数
1	新增功能	设置界面背景	42
2	新增功能	欢迎界面	115
3	新增功能	登陆密码	224
4	新增功能	翻译功能	255
5	新增功能	插入图片	242
6	新增功能	统计字符个数	169
7	新增功能	富文本功能	224
8	新增功能	朗读功能	114
9	新增功能	私密模式	186
10	新增功能	笔记编辑框内搜索	185
11	新增功能	模板便签	102
12	新增功能	语音听写	781
13	新增功能	语音合成	797
14	新增功能	对话式大模型	672
15	新增功能	撤回功能	116
16	新增功能	获取地理位置	147

3.3.3 维护后的质量分析

在代码维护之后，我们小组再次对代码进行了质量分析，使用CodeArts中的代码审查功能对代码的质量进行分析。分析结果见图24、图25。

图 24: 代码审查结果

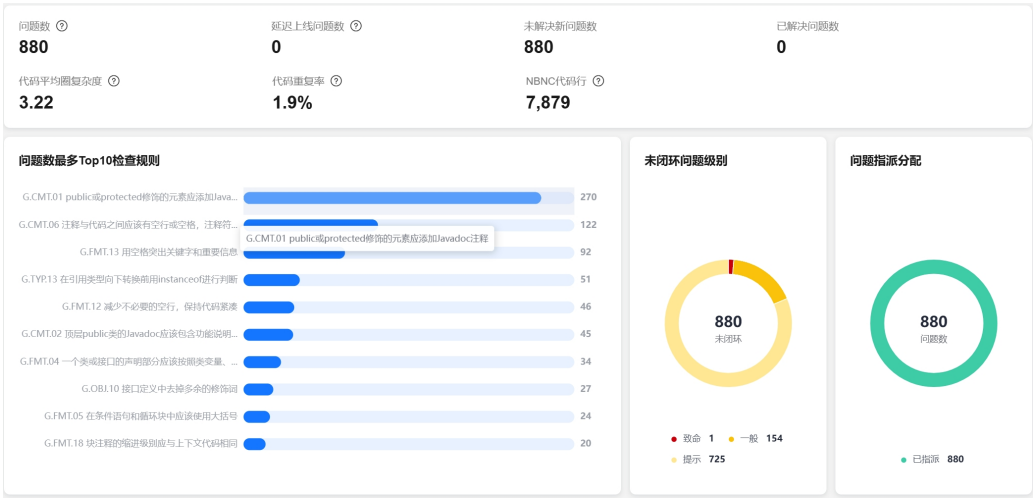


图 25: 代码审查结果

- 总体情况

  1. 未解决问题数：880
  2. 已解决问题数：0
  3. 代码行数：7,879
  4. 代码重复率：1.9%
  5. 代码平均复杂度：3.22

- 问题分析

  1. 问题数量： 880个未解决的问题是一个相对较高的数字，表明代码中存在较多的潜在问题，可能影响代码的可维护性和可读性。
  2. 问题类型：

     G.CMT.01（270个）：涉及public或protected修饰符的元素未添加JavaDoc注释。这表明代码文档化不足，可能导致后续维护和理解困难。

     G.CMT.06（122个）：注释与代码之间存在不一致性，可能导致误解。
  3. 代码复杂度：平均复杂度为3.22，虽然不算高，但仍需关注复杂度较高的部分，可能需要重构以提高可读性和可维护性。
  4. 代码重复率： 1.9%的重复率相对较低，表明代码的重复使用情况良好，但仍需定期检查以避免潜在的重复代码问题。

- 详细的分析结果汇总见表5，如下所示：

表 5: 代码质量分析详细结果

规则编号	违规数量	规则说明
G.CMT.02	45	顶层public类的Javadoc应该包含功能说明和创建日期/版本信息
G.FMT.05	24	在条件语句和循环块中应该使用大括号
G.FMT.18	20	快注释的缩进级别应与上下文代码相同
G.EXP.04	7	表达式的比较，应该遵循左侧倾向于变化、右侧倾向于不变的原则–使用equals方法进行字符串比较
G.FMT.12	46	减少不必要的空行，保持代码紧凑
G.OBJ.10	27	接口定义中去掉多余的修饰词
G.FMT.04	34	一个类或接口的声明部分应该按照类变量、静态初始化块、实例变量、构造器、方法的顺序出现，且用空行分隔
G.CMT.03	3	方法的Javadoc中应该包含功能说明，根据实际需要按顺序使用@param、@return、@throws标签对参数、返回值、异常进行注释
G.EXP.04	4	表达式的比较，应该遵循左侧倾向于变化、右侧倾向于不变的原则–表达式比较左变右不变
G.OBJ.08	1	正确实现单例模式
G.FMT.20	3	数字字面量应该设置合适的后缀，long类型应该使用L作为后缀
G.DCL.03	5	禁止C风格的数组声明
G.FMT.06	11	对于非空块结构，左大括号应该放在行尾，右大括号应该另起一行
G.OBJ.07	13	子类覆写父类方法或实现接口时必须加上@Override注解
G.ERR.03	6	不要直接捕获可通过预检查进行处理的RuntimeException，如NullPointerException、IndexOutOfBoundsException等
G.FMT.08	6	使用空格进行缩进，每次缩进4个空格
G.CTL.01	14	不要在控制性条件表达式中执行赋值操作或执行复杂的条件判断
G.OTH.01	1	安全场景下必须使用密码学意义上的安全随机数
G.NAM.04	5	方法名应采用小驼峰命名
G.CMT.06	122	注释与代码之间应该有空行或空格，注释符与注释内容之间应该有空格
G.CMT.01	270	public或protected修饰的元素应添加Javadoc注释
G.TYP.13	51	在引用类型向下转换前用instanceof进行判断
G.OTH.03	15	不用的代码段包括import，直接删除，不要注释掉–不用的代码段，直接删除，不要注释掉
G.FMT.10	8	行宽不超过120个字符符
G.NAM.03	2	类、枚举和接口名应采用大驼峰命名
G.FMT.07	1	应该避免空块，必须使用空块时，应采用统一的大括号换行风格
G.CMT.03	2	方法的Javadoc中应该包含功能说明，根据实际需要按顺序使用@param、@return、@throws标签对参数、返回值、异常进行注释–功能描述和标签之间要有一个空行

下页继续

续表 5：代码质量分析详细结果

规则编号	违规数量	规则说明
G.MET.01	15	方法要简短–方法的代码块深度不应超过4层
G.OTH.03	9	不用的代码段包括import，直接删除，不要注释掉–不用的import语句，直接删除，不要注释掉
G.MET.01	13	方法要简短–方法行数不应超过50行
G.MET.01	1	方法要简短–方法的参数不应超过5个
G.NAM.05	1	常量名采用全大写单词，单词间以下划线分隔
G.CON.12	3	避免不加控制地创建新线程，应该使用线程池来管控资源
G.FMT.13	92	用空格突出关键字和重要信息
总计	880	

针对如上代码质量分析，我们小组对应地进行了如下改进：

1. **代码注释规范化：** 针对G.CMT.01、G.CMT.03和G.CMT.06问题，我们对所有public和protected方法添加了规范的Javadoc注释，包含完整的功能说明、参数说明和返回值说明。同时规范了注释格式，确保注释与代码之间保持适当的空行和空格。
2. **代码格式标准化：**解决G.FMT.06、G.FMT.08和G.FMT.10等问题，统一采用4空格缩进，控制行宽在120字符以内，并规范了大括号的放置位置。通过IDE的代码格式化工具，保证了整体代码风格的一致性。
3. **异常处理优化：**针对G.ERR.03问题，重构了异常处理逻辑。对可能出现NullPointerException等运行时异常的代码段加入了try-catch或者分支语句作为预检查机制，提高了代码的健壮性。
4. **方法结构优化：**解决G.MET.01系列问题，对部分超过50行的方法进行了重构和拆分，降低了代码块的嵌套深度，并优化了参数数量，提升了代码的可维护性。
5. **命名规范统一：**针对G.NAM.03、G.NAM.04和G.NAM.05问题，统一了类名使用大驼峰命名法，方法名使用小驼峰命名法，常量使用全大写下划线分隔的命名规范；对部分常量进行了统一声明。
6. **代码冗余清理：**解决G.OTH.03问题，清理了所有注释掉的代码段和未使用的import语句，提高了代码的整洁度和可读性。

接下来本小组对维护后的代码进行了代码风格展示。

3.3.4 维护代码风格展示

1. 注释规范性与完整性

- 注释完整性：新增代码中添加了大量注释(约3900行)，占总代码量的50%左右，有助于代码的可读性和可维护性。
- 命名规范：类注释、语句与语句块注释符合规范。
- 代码格式：遵从最基本的缩进、空行、空格、换行等格式规范。

具体的注释规范性展示见图26。

```
124  * @ProjectName:
125  * @Package:        net.micode.notes.ui
126  * @ClassName:      NoteEditActivity
127  * @Description:    编辑小米便签应用中的便签内容。它向用户提供了一个界面，允许用户输入、编辑和保存文本信息。
128  * 具体来说，用户可以在该界面中添加或编辑标题、标签、内容、图片等信息，并将这些信息保存到小米便签数据库中。
129  * 还提供了一些常用的编辑功能，如颜色设置、插入图片等，使得用户可以更加方便地对便签内容进行编辑和美化。
130  * @Author:         Xingi Qin
131  * @CreateDate:     2024-12-20 8:44
132  * @UpdateUser:     更新者：Liren_Qiu
133  * @UpdateDate:     2024-01-03 8:44
134  * @UpdateRemark:   更新说明：
135  * @Version:        1.0
136  */
137
138 public class NoteEditActivity extends Activity implements OnClickListener,
139         NoteSettingChangedListener, OnTextViewChangeListener {
140     /**
141      * 顾名思义，本类用于对便签的编辑
142      * 头部视图的ViewHolder类，用于存储头部视图中的UI组件引用。
143      * 继承的都是系统内部与监听相关的类
144      */
145
146     private class HeadViewHolder {  2 usages
147         public TextView tvModified; // 显示修改日期的文本视图  2 usages
148         public ImageView ivAlertIcon; // 提示图标图像视图  3 usages
149         public TextView tvAlertDate; // 显示提醒日期的文本视图  5 usages
150         public ImageView ibSetBgColor; // 设置背景颜色的图像视图  2 usages
151         private TextView tvNum;//  在头部视图holder里声明文本视图用于统计字符，编辑器edittext直接借用下面的mNo
152     }
153
```

图 26: 注释规范性与完整性

2. 代码质量指标

- 代码复杂度：平均复杂度为3.22，处于合理范围内，表明代码结构相对简单清晰。
- 代码重复率：1.9%的重复率较低，说明代码复用情况良好。
- 异常处理：增加了预检查机制。

具体的代码质量指标展示见图27，利用了较多的分支和try-catch语句预检查。

```
264     public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResult
278         && grantResults[1] == PackageManager.PERMISSION_GRANTED) {
279     Toast.makeText( context: this,  text: "申请权限", Toast.LENGTH_LONG).show();
280     try {
281         List<String> providers = locationManager.getProviders( enabledOnly: true);
282         if (providers.contains(LocationManager.NETWORK_PROVIDER)) {//如果是Network
283             locationProvider = LocationManager.NETWORK_PROVIDER;
284
285         } else if (providers.contains(LocationManager.GPS_PROVIDER)) {//如果是GPS
286             locationProvider = LocationManager.GPS_PROVIDER;
287         }
288         Location location = locationManager.getLastKnownLocation(locationProvider);
289         if (location != null) {//如果不能正常从LocationProvider中获取位置，则复用上次位置
290             Toast.makeText( context: this,  text: location.getLongitude() + " " +
291                     location.getLatitude() + " ", Toast.LENGTH_SHORT).show();
292             Log.v( tag: "TAG",  msg: "获取上次的位置-经纬度：" + location.getLongitude() + "    " + loca
293         } else {
294             // 监视地理位置变化，第二个和第三个参数分别为更新的最短时间minTime和最短距离minDistace
295             locationManager.requestLocationUpdates(locationProvider,  minTimeMs: 0,  minDistanceM: 0
296         }
297
298     } catch (SecurityException e) {
299         e.printStackTrace();
300     }
301 } else {
302     Toast.makeText( context: this,  text: "缺少权限", Toast.LENGTH_LONG).show();
303     finish();
304 }
305
```

图 27: 代码质量指标

3. **架构设计**

- 模块化：新增功能按照既有架构进行模块划分，保持了良好的代码组织结构。
- 接口设计：通过接口封装第三方服务调用，提高了代码的可维护性和可扩展性。
- 依赖管理：合理使用了外部依赖，但需要注意第三方SDK的版本管理和兼容性问题。

具体的架构设计展示见图28，独立使用第三方SDK的信息传递文件以及配置文件单独成包。

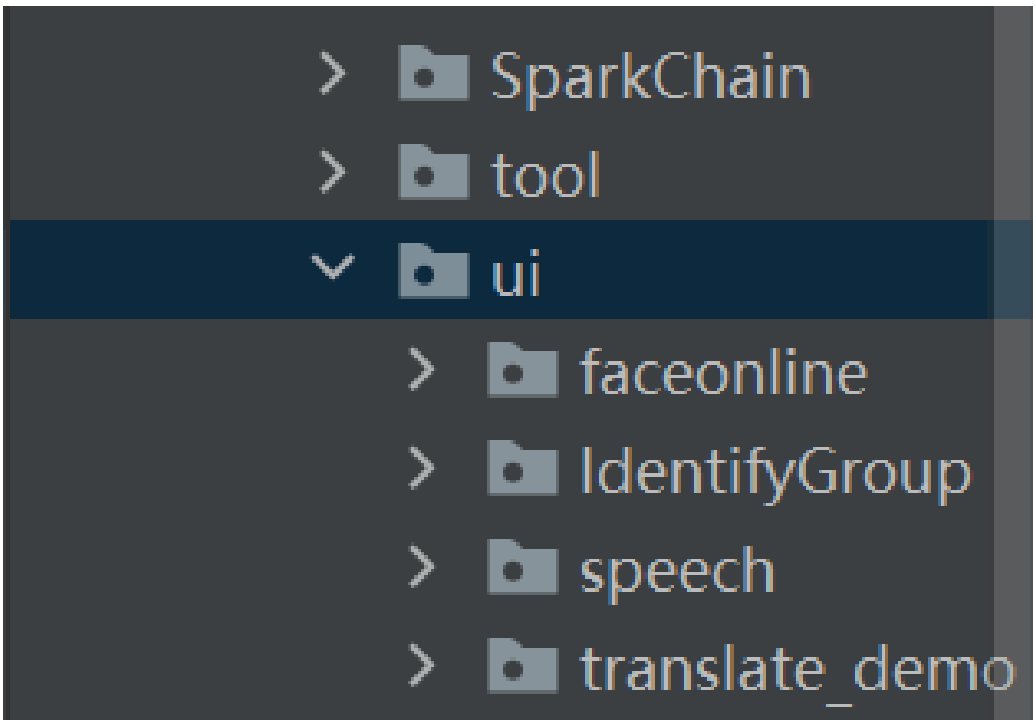


图 28: 架构设计

4. **性能考虑**

- 资源管理：对于线程创建等资源密集型操作，部分使用线程池进行管理。
- 内存使用：需要注意图片处理、语音识别等功能的内存占用情况。
- 响应速度：异步处理网络请求和耗时操作，保证UI响应的流畅性。

具体的性能考虑展示见图29，使用了多线程进行异步处理并使用线程池管理，保证响应的流畅性。

```
457     }
458
459     private float dp2px(Context context, float dipValue) {//dp转px  2 usages
460         if (context == null) {
461             return 0;
462         }
463         final float scale = context.getResources().getDisplayMetrics().density;//获取当前屏幕的密度
464         return (float) (dipValue * scale + 0.5f);
465     }
466
467     public static void showToast(final Activity context, final String content){//在子线程中调用  no usag
468
469         context.runOnUiThread(new Runnable() {//在主线程中调用
470             @Override
471             public void run() {//
472                 int random = (int) (Math.random()*(1-0)+0);//随机数，用于在屏幕上随机位置显示
473                 Toast.makeText(context,content,random).show();//显示Toast
474             }
475         });
476
477     }
478
479     public void toend(){//滚动到底部  3 usages
480         int scrollAmount = chatText.getLayout().getLineTop(chatText.getLineCount()) - chatText.getHe
481         if (scrollAmount > 0) {//如果高度差大于0，则滚动
482             chatText.scrollTo( x: 0,  y: scrollAmount+10);//滚动到指定位置
483         }
484     }
```

图 29: 性能考虑

5. 安全性

- 数据安全：实现了密码保护和私密模式等功能，增强了应用的安全性。
- 权限管理：合理申请和使用地理位置等敏感权限。
- 加密措施：使用MD5等加密算法保护用户数据。

具体的代码安全性展示见图30，使用了MD5等加密算法保护用户数据。

```java

import java.security.MessageDigest;

/**
 * 加密解密工具类(对字符串加密) MD5加密
 */
public class MD5Utils {  3 usages

    /**
     * MD5加密算法使用 对字符串加密
     *
     * @param info 参数为需要加密的String
     * @return 返回加密后的String
     */
    public static String getMD5Code(String info) {  2 usages
        try {
            MessageDigest md5 = MessageDigest.getInstance( algorithm: "MD5");
            md5.update(info.getBytes( charsetName: "utf-8"));//设置编码格式
            byte[] encryption = md5.digest();//加密所需的字节数组
            StringBuffer stringBuffer = new StringBuffer();
            for (int i = 0; i < encryption.length; i++) {
                if (Integer.toHexString( i: 0xff & encryption[i]).length() == 1) {//如果为1位，前面补0
                    stringBuffer.append("0").append(Integer.toHexString( i: 0xff & encryption[i]));
                } else {//如果为2位，直接转换
                    stringBuffer.append(Integer.toHexString( i: 0xff & encryption[i]));
                }
            }
            return stringBuffer.toString();
        } catch (Exception e) {
            return "MD5加密异常";
        }
```

图 30: 安全性

3.4 维护后的软件原型以及功能展示

3.4.1 设置界面背景

修改了小米便签的初始背景，更具有美观性，让用户有良好的使用体验。同时，可以对背景进行切换操作，总共设计了2款不同背景。（见图31、图32、图33）

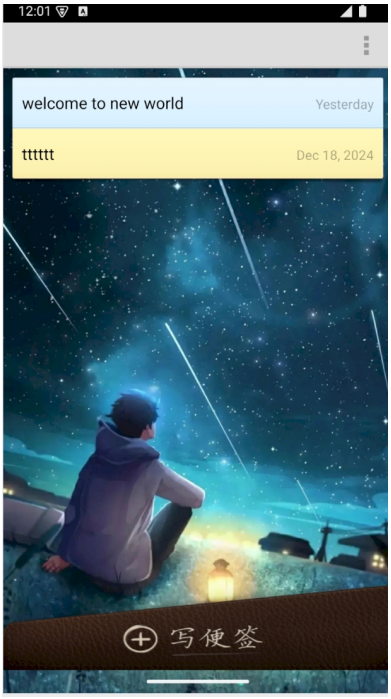


图 31: 设置背景1

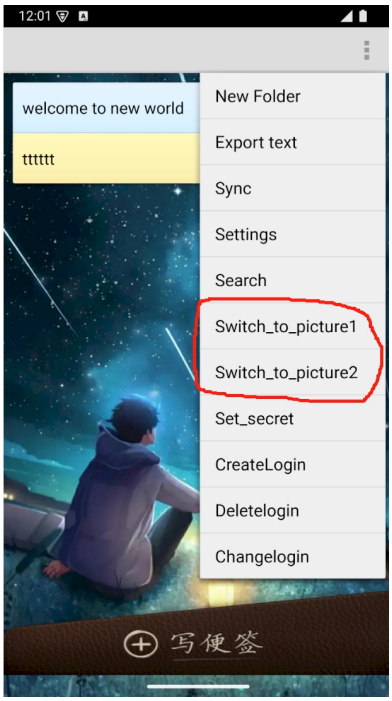


图 32: 设置背景按钮

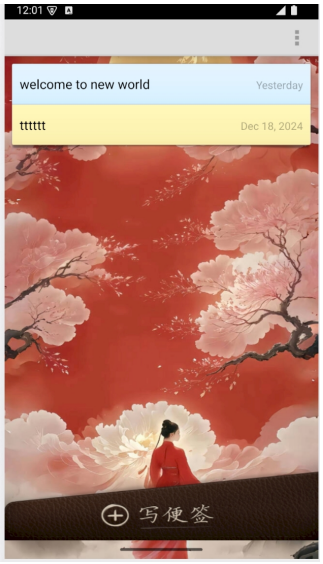


图 33: 设置背景2

3.4.2 欢迎界面

实现进入便签前两秒钟的欢迎界面，让用户有良好的使用体验。（见图34）

图 34: 欢迎界面

3.4.3 登录密码

在主界面菜单中，有新建登陆密码按钮、删除登陆密码按钮和修改登录密码按钮。点击新建登陆密码会出来新建登录密码界面，里面有输入密码和确认密码，如果输入密码和确认密码不一致则要求用户重新输入。（见图35）

这一功能的时序图如图40所示。

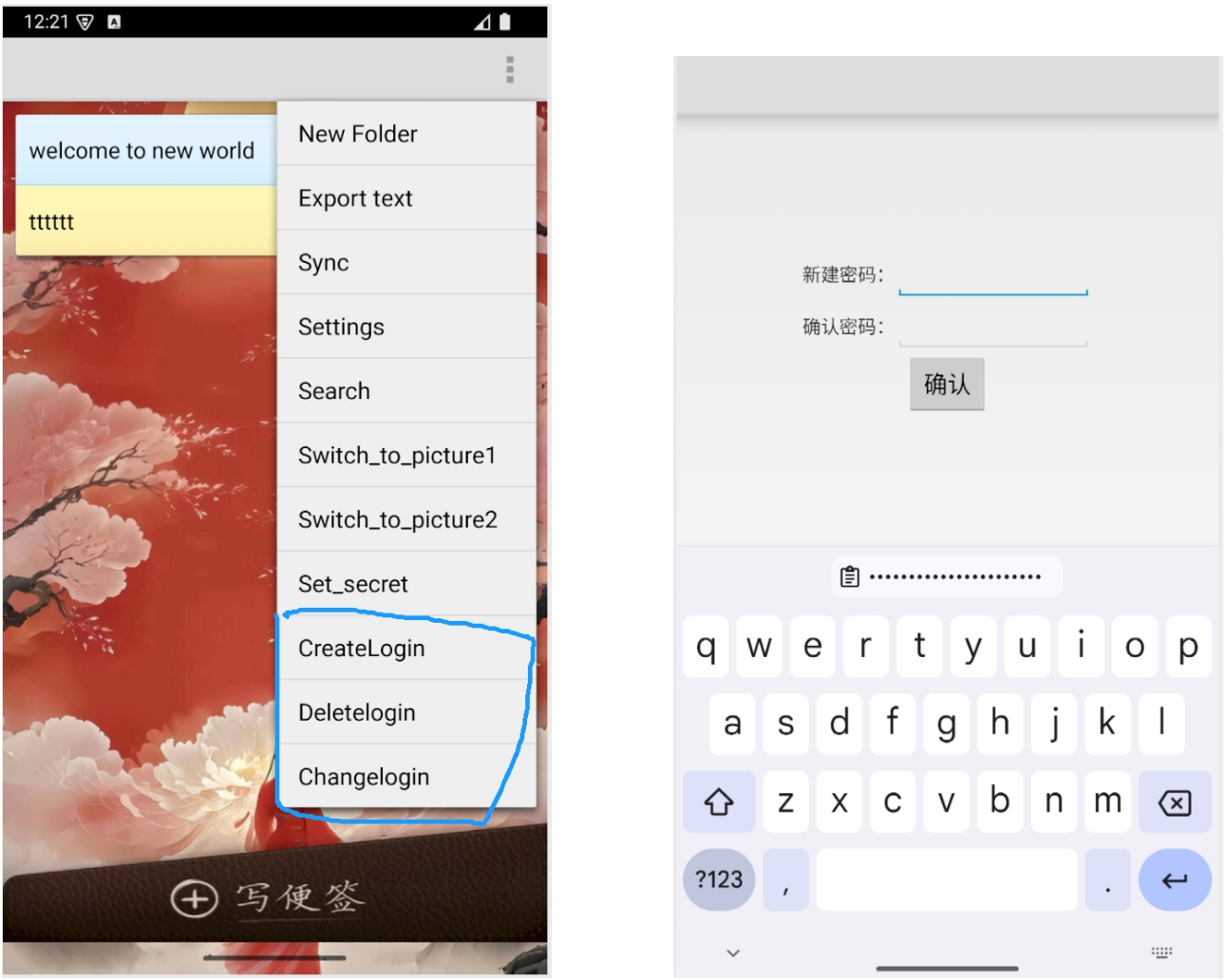


图 35: 新增设置密码的操作

我们新建密码为“111”，再次进行登录操作可以发现，必须要输入密码才能进入应用。输入密码之后可以成功进入。（见图36）

图 36: 新增密码后的登录界面，要求输入密码

如图37所示，我们重新设置密码的时候，需要我们输入现有的密码，然后才能设置新的密码，输入新的密码“123”，确认密码，密码进行更换。

图 37: 重新设置密码

再次登录的时候，原本的密码“111”已经不再适用，输入现有密码“123”才能够成功进入。（见图38）

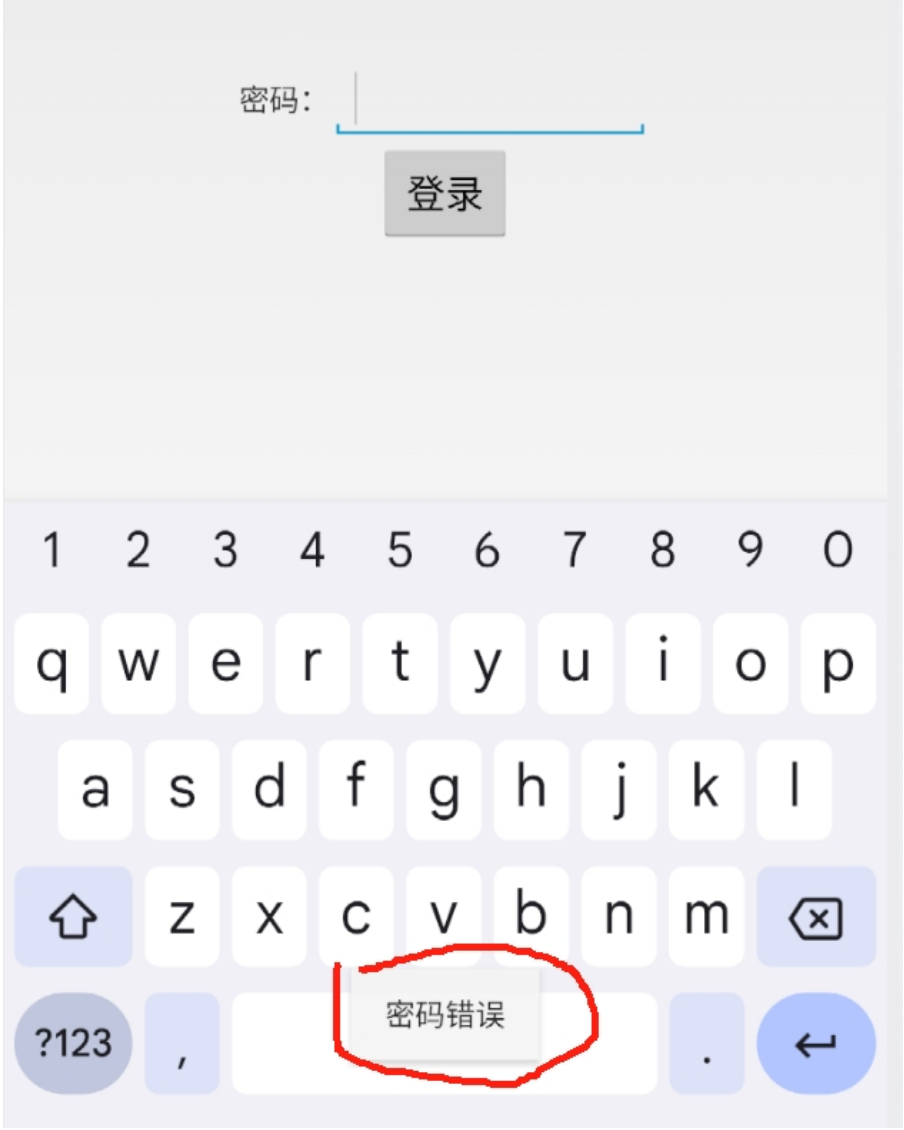
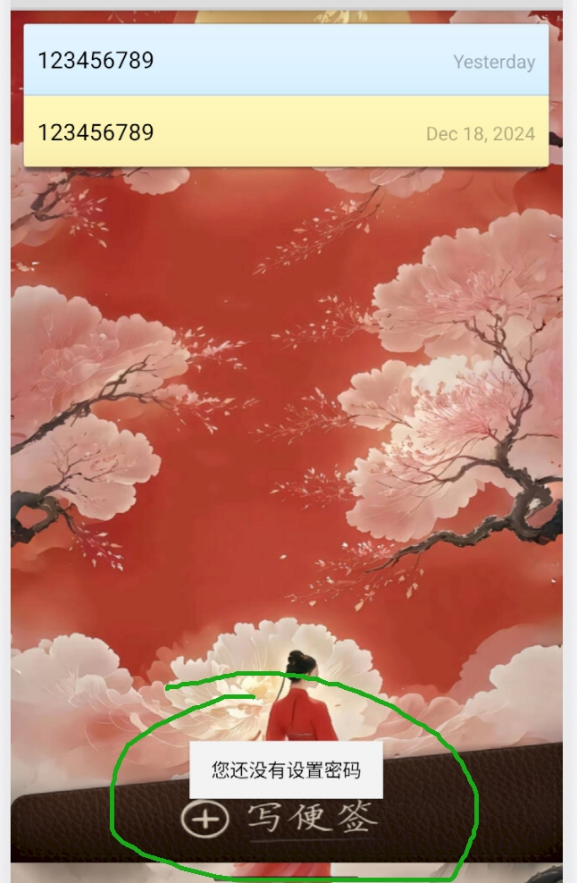


图 38: 使用原有的密码登录失败

同样的，要想删除设置的密码，也要先输入原有的密码。（如图39）删除之后，再次点击删除密码的按钮会提醒“没有设置密码”。

(a) 删除密码



(b) 删除密码之后显示没有设置密码

图 39: 删除设置的密码

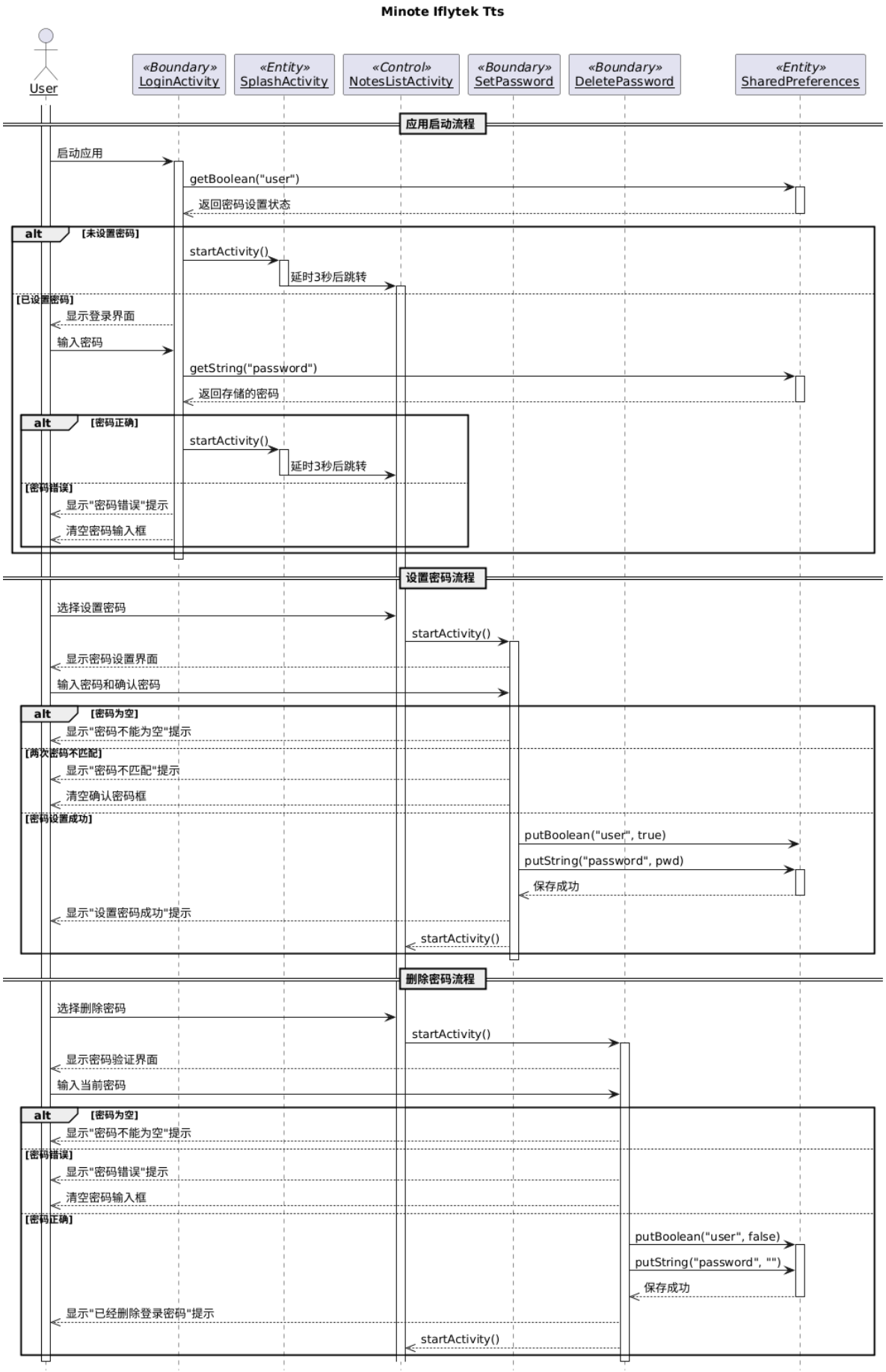


图 40: 登录密码的时序图

3.4.4 翻译功能

在便签编辑界面的上方，有个翻译按钮，点开之后有三个按钮可以选择：中文翻译为英文、英文翻译为中文、还原，其中还原可以将文本还原为翻译之前的内容。这一功能的时序图如图44所示。

如图41所示，可以将便签内的内容由英文转化为中文。例：“Welcome to the new world.”可以翻译为“欢迎来到新世界。”

图 41: 翻译功能中的将英文翻译为中文

与此同时，我们也可以将便签内的内容由中文转化为英文。（见图42）例：“欢迎来到新世界。”可以翻译为“Welcome to the new world.”

图 42: 翻译功能中将中文转化为英文

另外，我们在翻译功能中还实现了，将翻译的内容进行还原，可以将翻译之后的内容再还原回去。（见图43）

图 43: 翻译功能中的还原操作

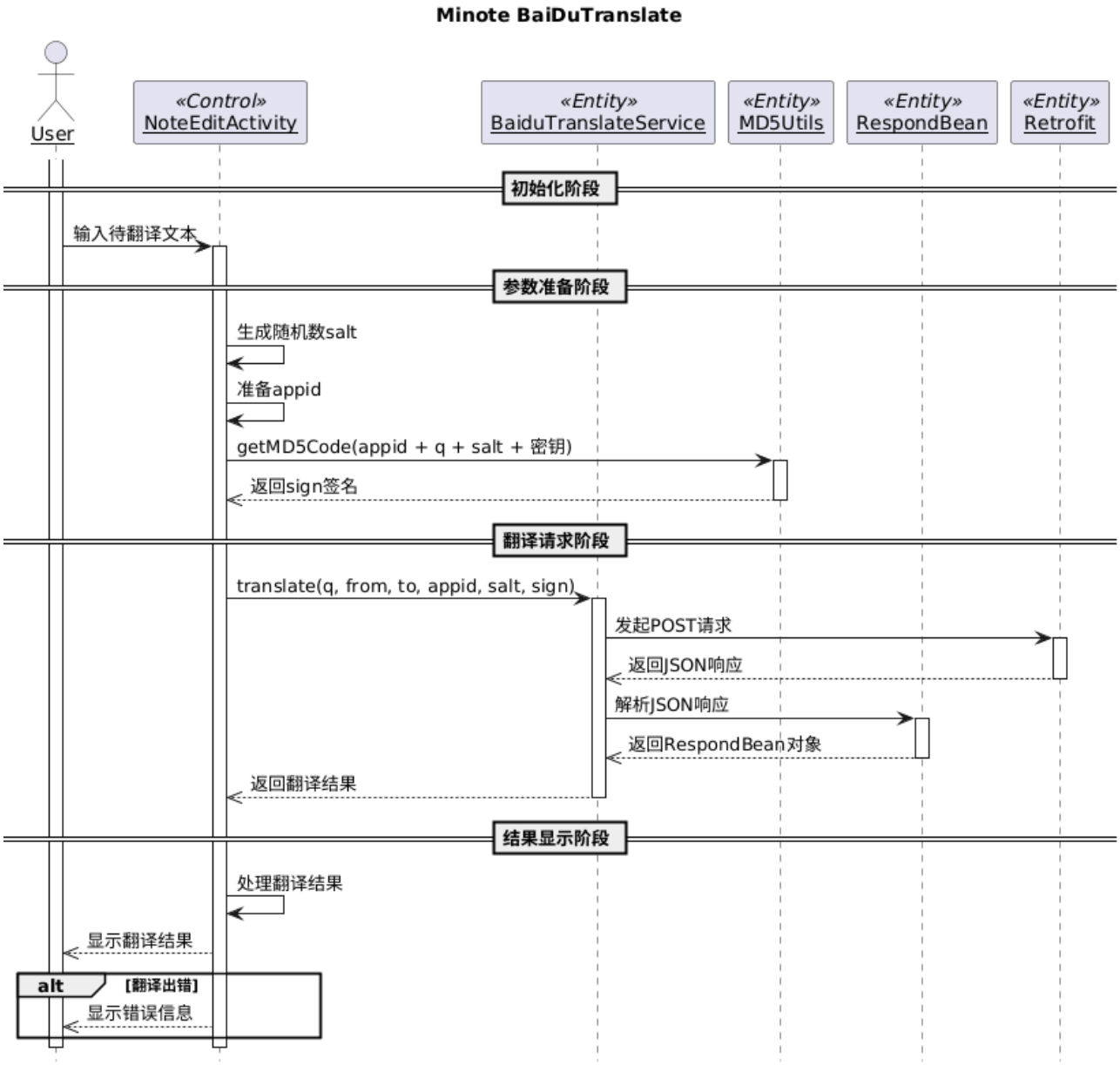


图 44: 翻译功能的时序图

3.4.5 插入图片

在便签编辑界面增加插入图片按钮，点击后跳转到相册，供用户选择插入图片，用户选好图片后将图片的路径插入到文本中，在将文本中的路径显示成图片。（见图45）

图 45: 将本地相册中的图片导入便签编辑界面中

时序图如下图（图46）所示

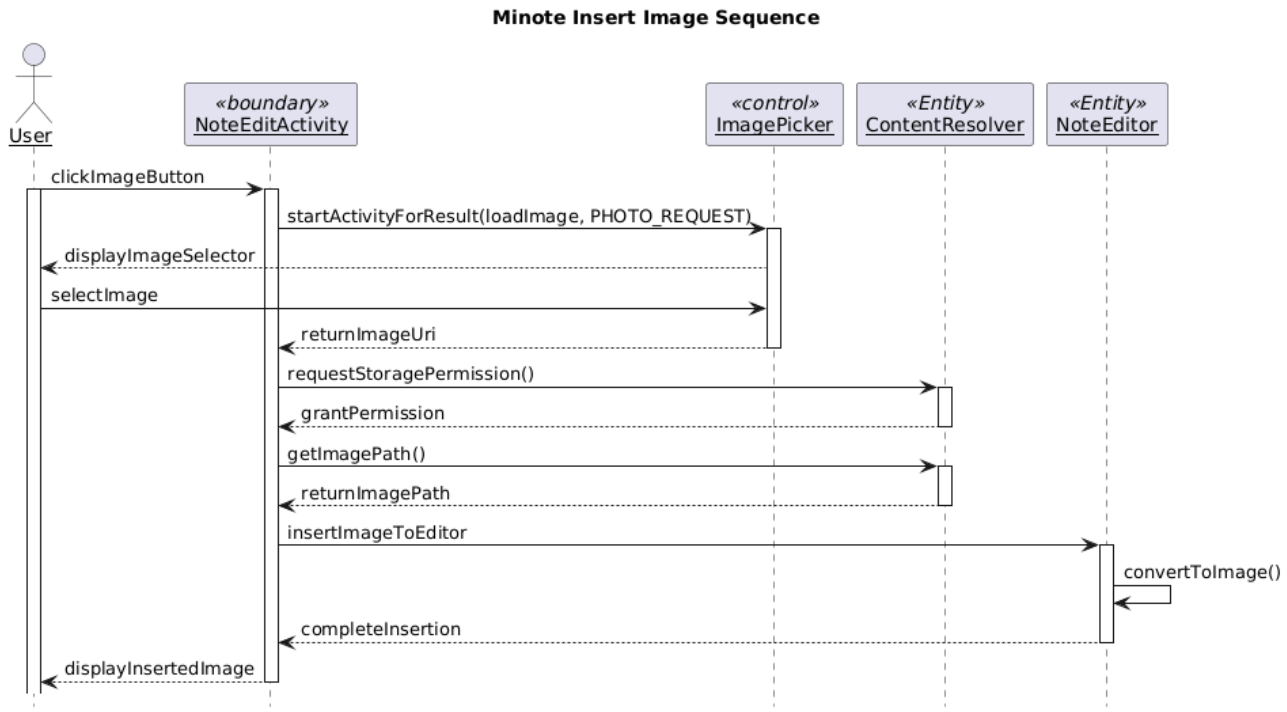


图 46: 插入图片的时序图

3.4.6 统计字符个数

在便签编辑页面的左上角菜单里面，添加了统计总字符数的按钮，可以实时反映便签中的字符个数。如图47所示。

图 47: 统计字符个数

时序图如下图所示

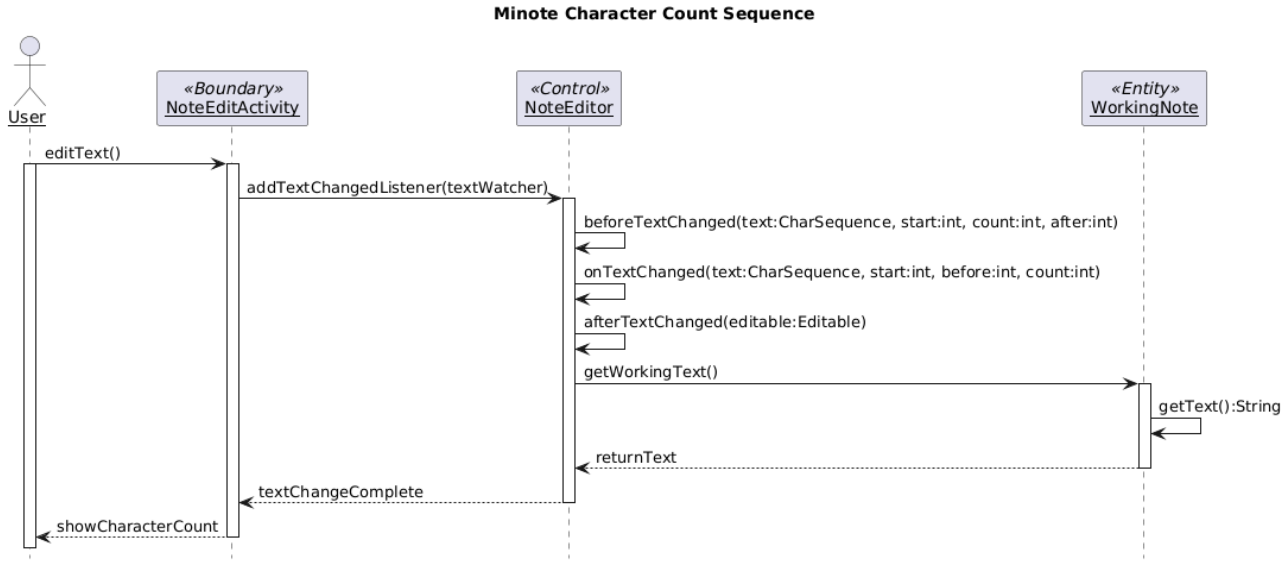


图 48: 统计字符个数的时序图

3.4.7 富文本功能

在便签编辑界面，当选中文本后，选中文本下面会显示加粗、斜体、删除线以及高亮按钮，可以对文本进行加粗、斜体、删除线以及高亮等操作。(见图49)

其中加粗功能见图50，斜体功能见图51，删除线功能见图52，高亮功能见图53。

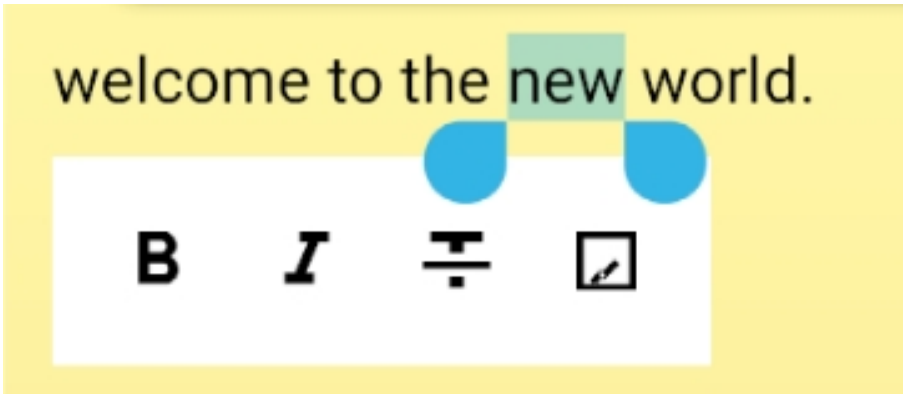


图 49: 富文本功能

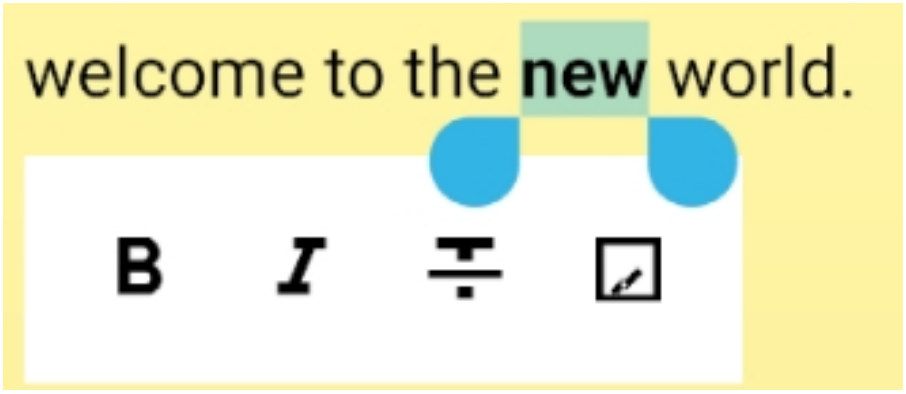


图 50: 加粗功能

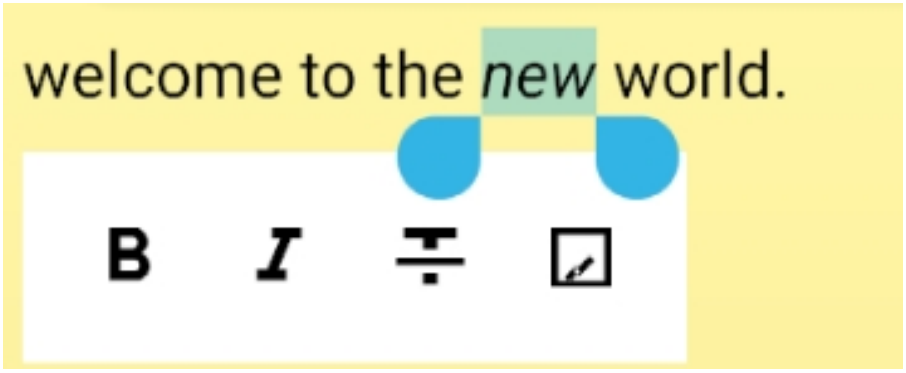


图 51: 斜体功能

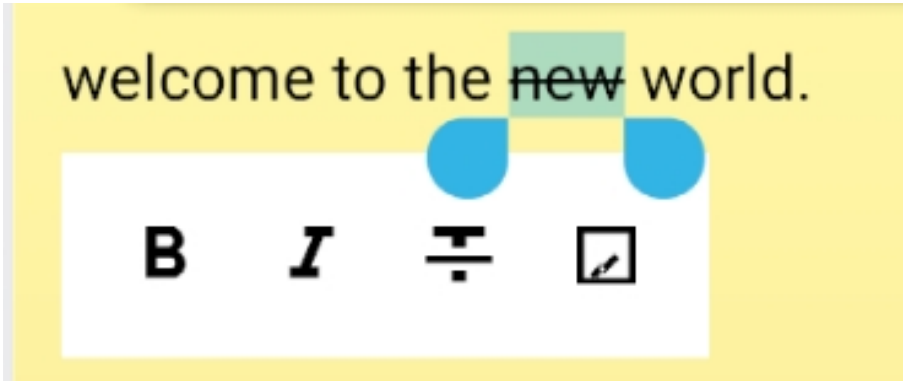


图 52: 删除线功能

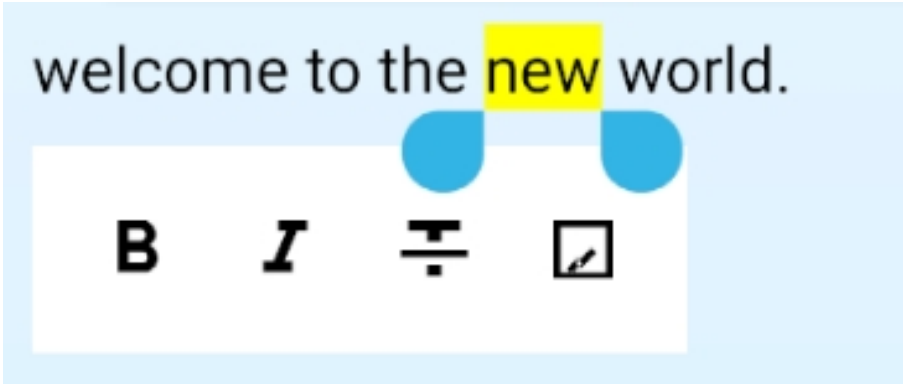


图 53: 高亮功能

时序图如下图所示

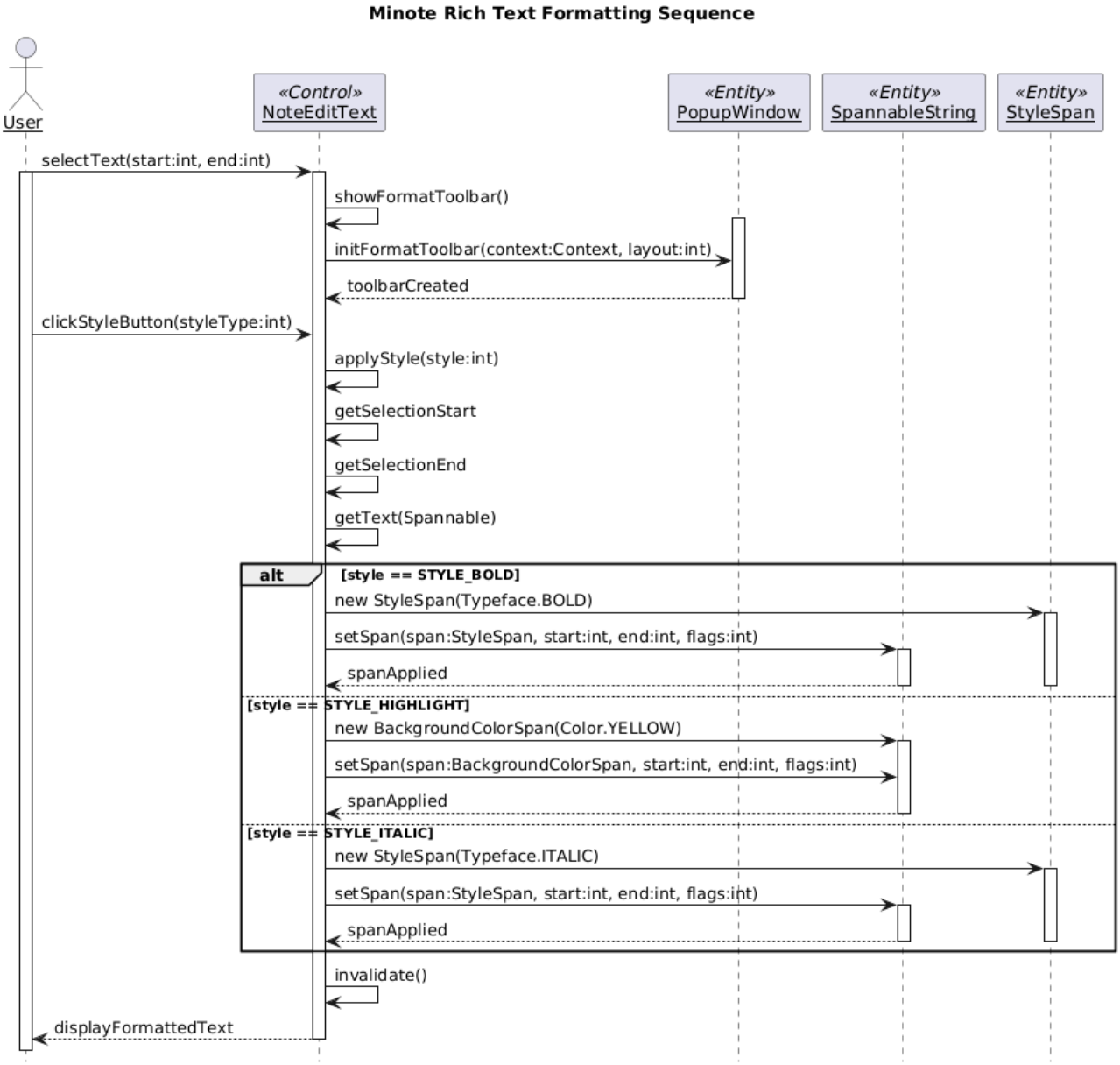


图 54: 富文本功能的时序图

3.4.8 朗读功能

在修改便签界面上侧有朗读按钮，点击朗读按钮就可以朗读便签文本的内容（除去了空格、换行符、图片路径），功能界面见图55。

图 55: 朗读功能

时序图如下图所示

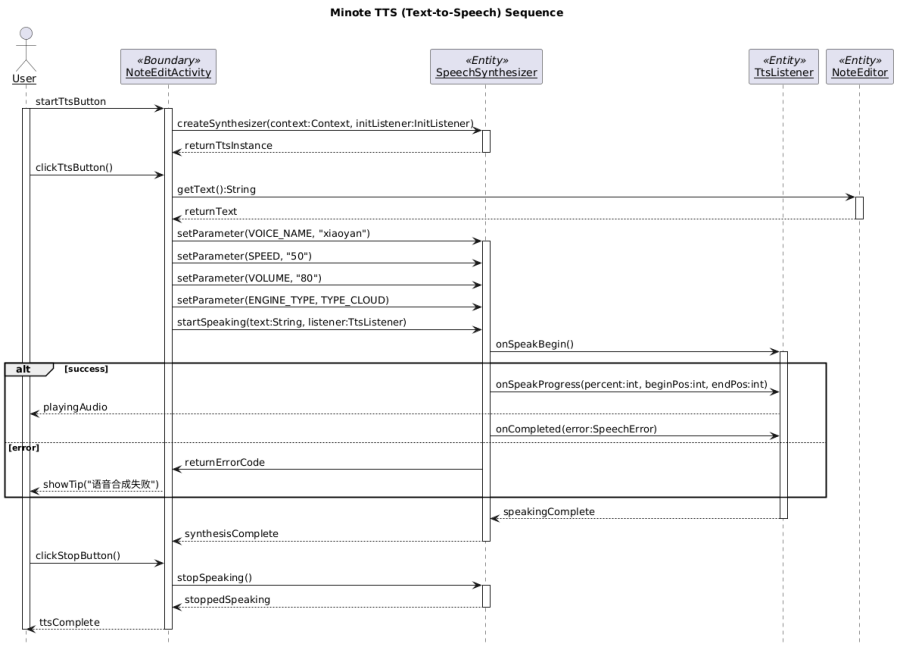
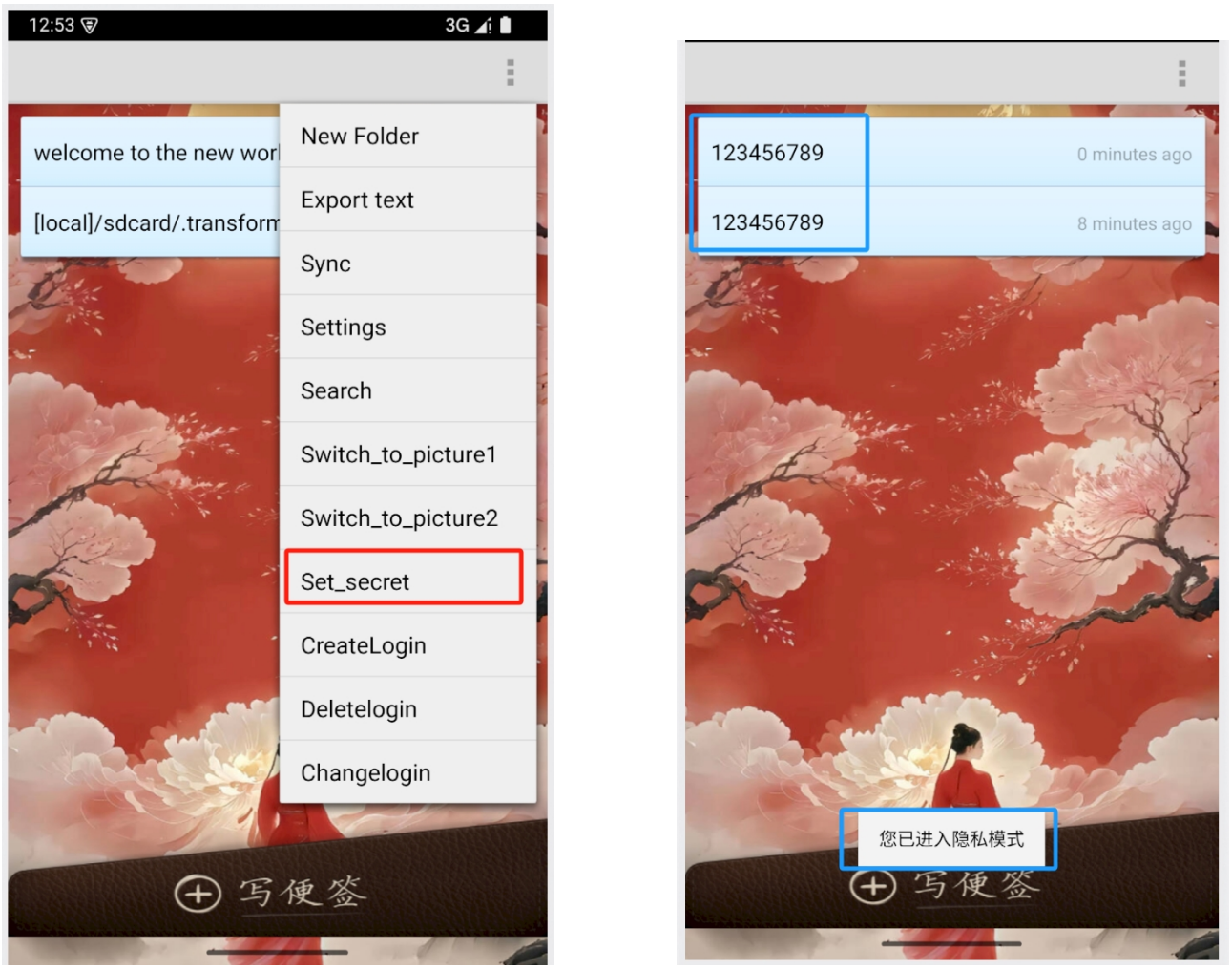


图 56: 朗读功能的时序图

3.4.9 私密模式

在写便签前的界面，右上角的菜单中，添加两个按钮：私密模式和退出私密模式。点击私密模式后，所有便签的在首页的显示都是“123456789”，（见图57）防止其他人通过首页了解便签内部信息。点击退出私密模式即可退出私密模式。



(a) 私密模式按钮　　(b) 进入私密模式

图 57: 私密模式

当我们退出私密模式时，首行内容重新显示出来。如图58所示。

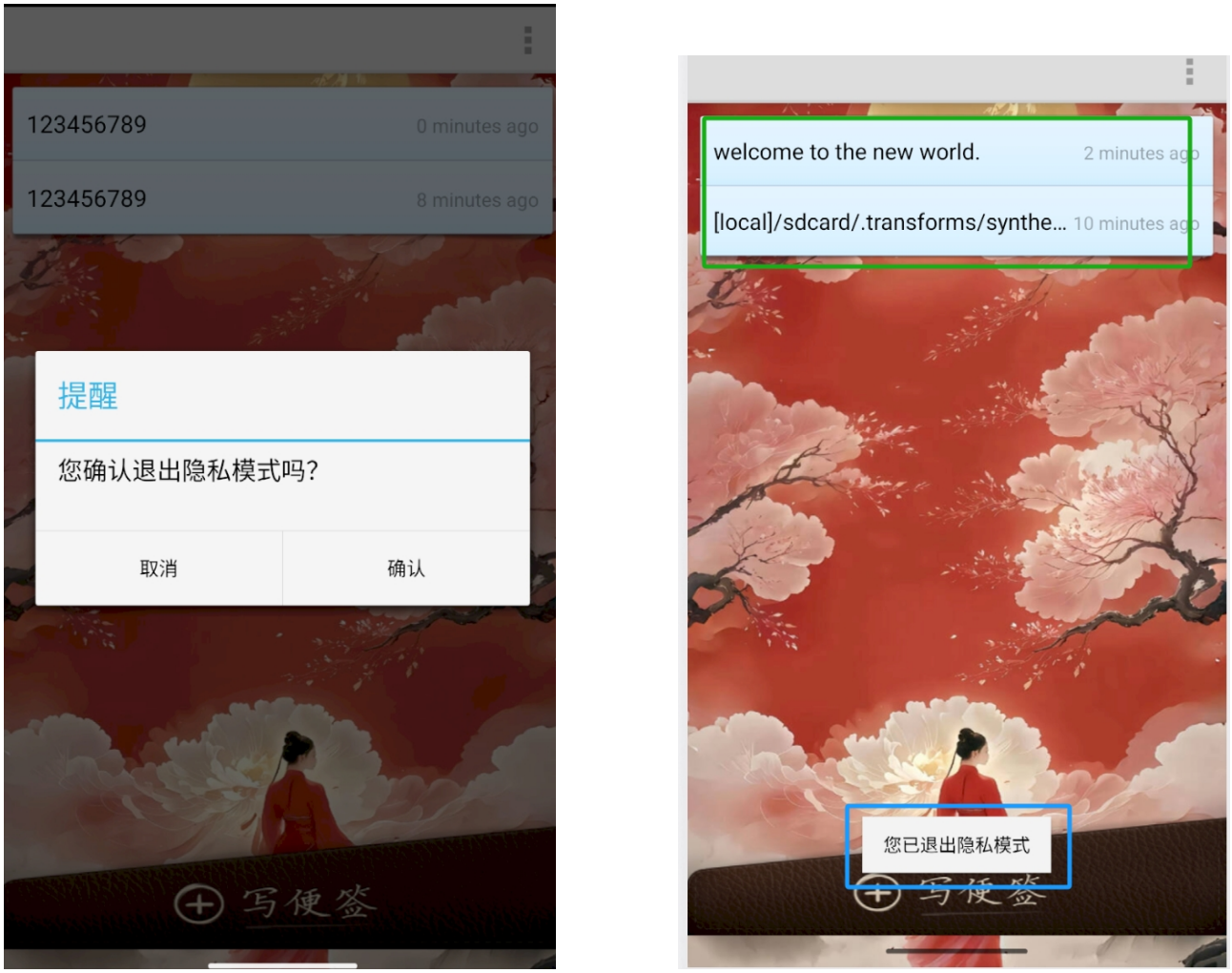


图 58: 退出私密模式

时序图如下图所示

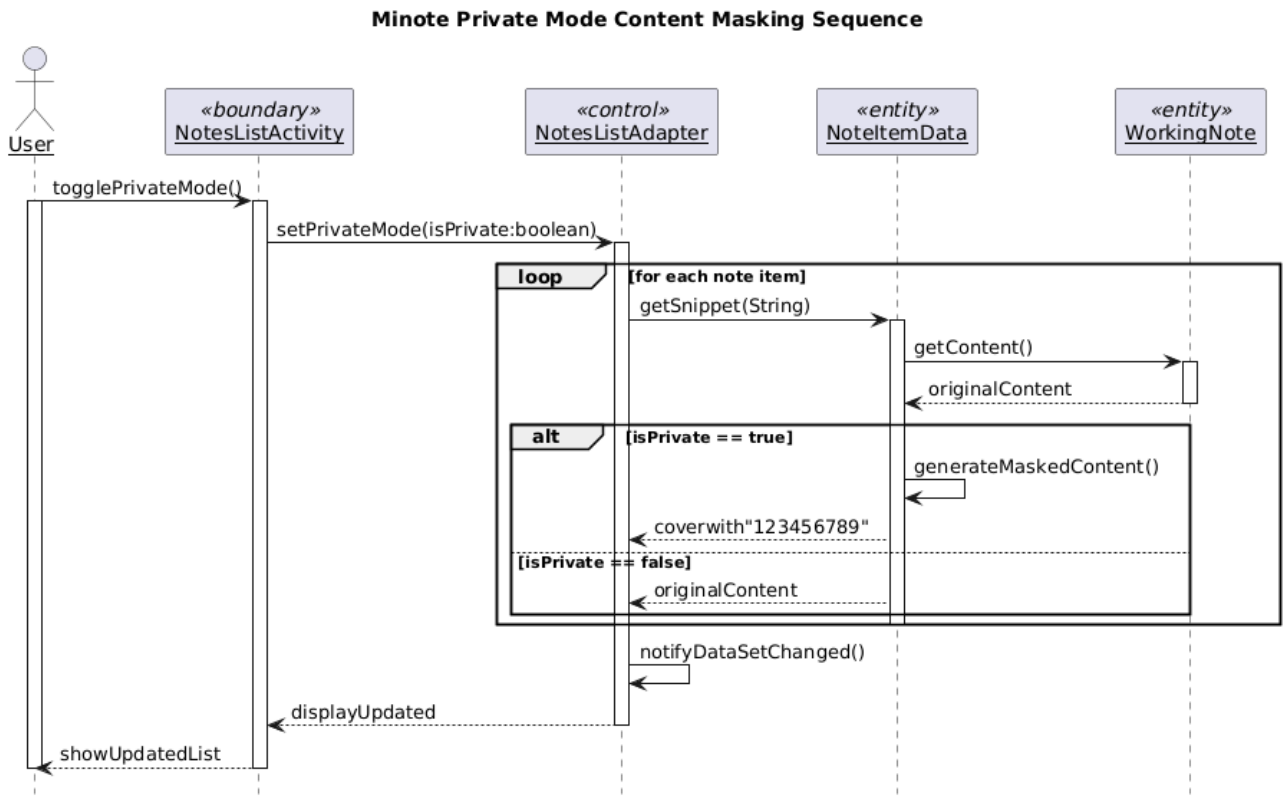
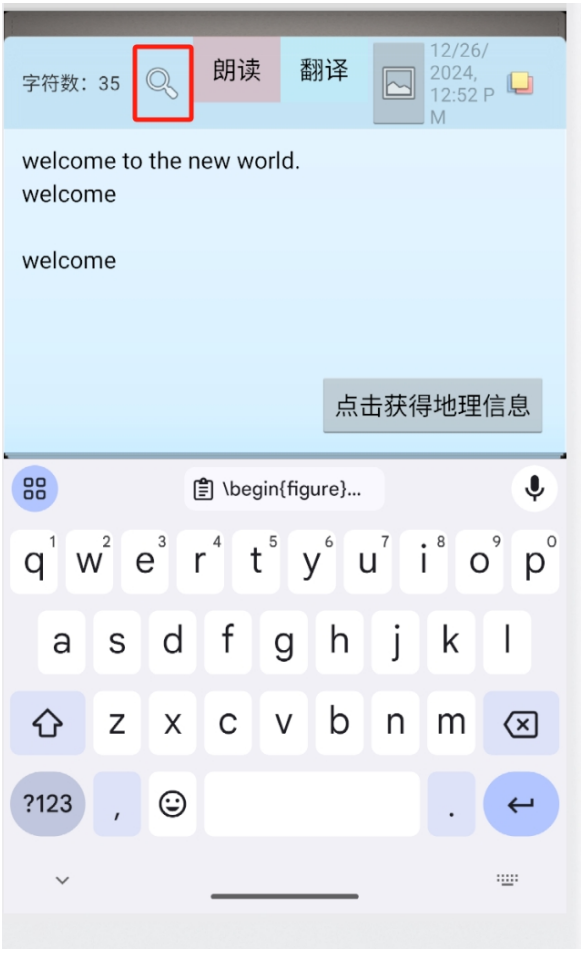


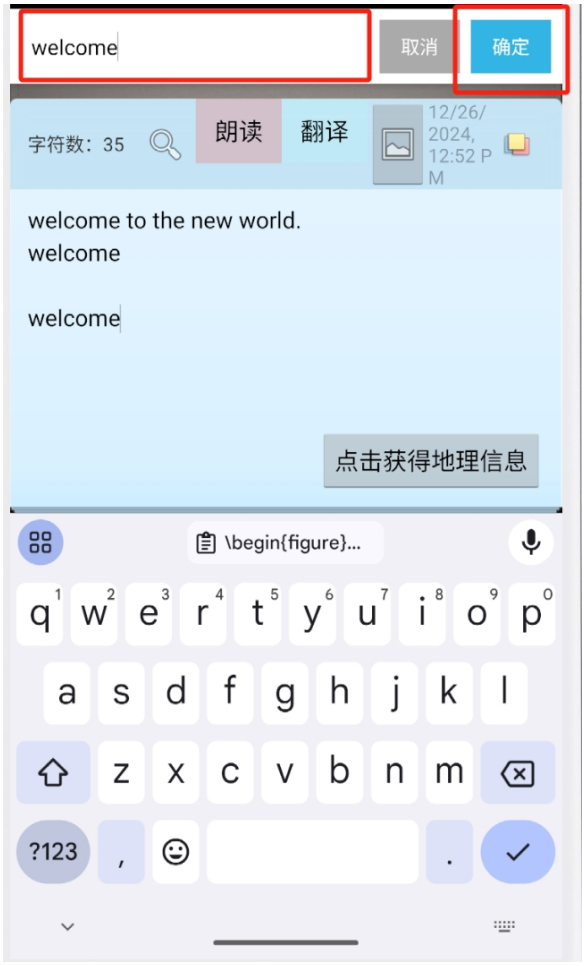
图 59: 私密模式的时序图

3.4.10 笔记编辑内搜索

- 在编辑笔记界面，点击搜索按钮，可以对笔记内容进行搜索。(见图60(a))
- 点击搜索按钮后，会弹出搜索框，输入要搜索的内容，点击确定按钮，可以搜索到所有包含该内容的内容。(见图60(b))
- 当搜索到内容时，会高亮所有搜索到的内容。(见图61(a))
- 如果搜索不到内容，会显示“未找到相关内容”。(见图61(b))
- 点击取消按钮，可以取消搜索，高亮的文本会恢复到原来的颜色。

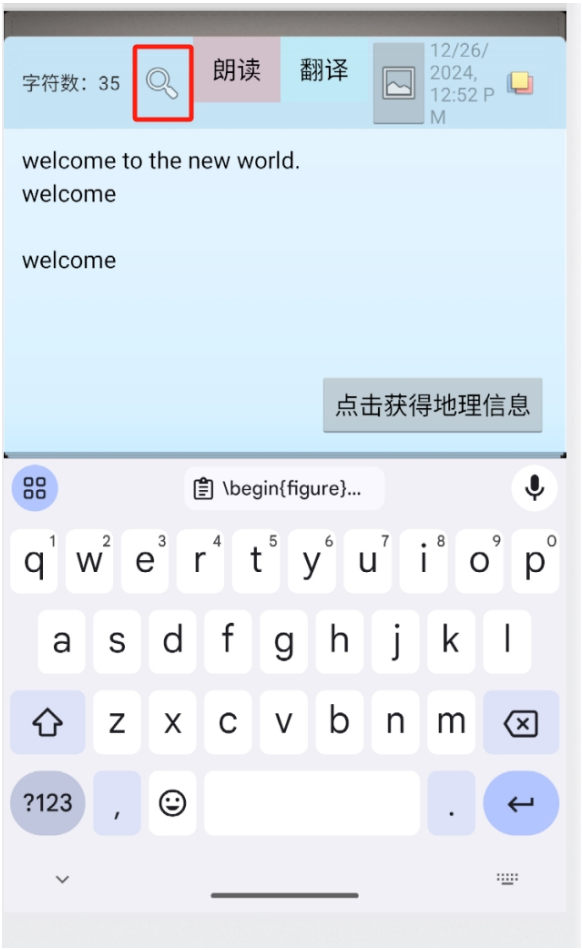


(a) 搜索按钮

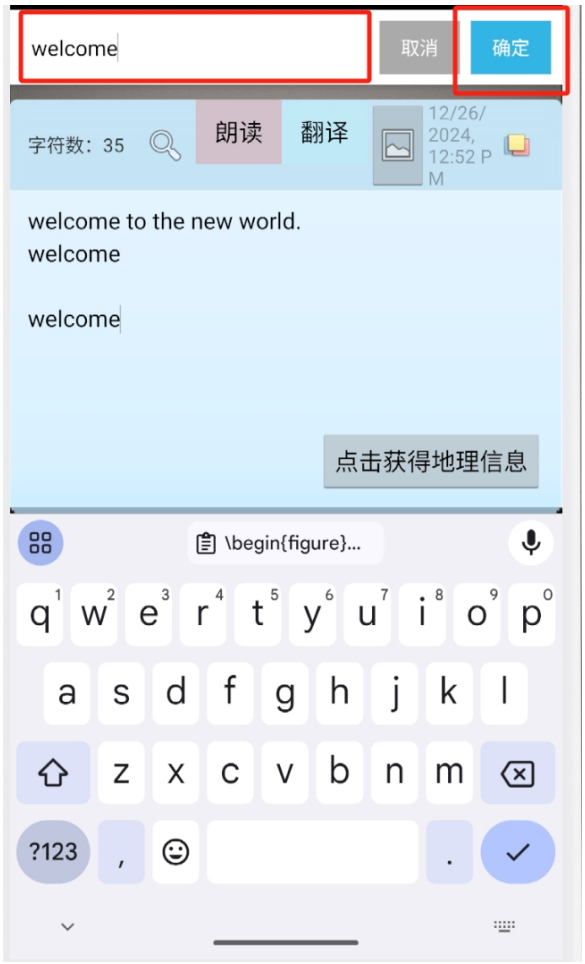


(b) 点击搜索按钮，弹出搜索框

图 60: 笔记内搜索



(a) 搜索到内容，高亮显示



(b) 搜索不到内容，显示“未找到相关内容”

图 61: 笔记内搜索

时序图如下图所示

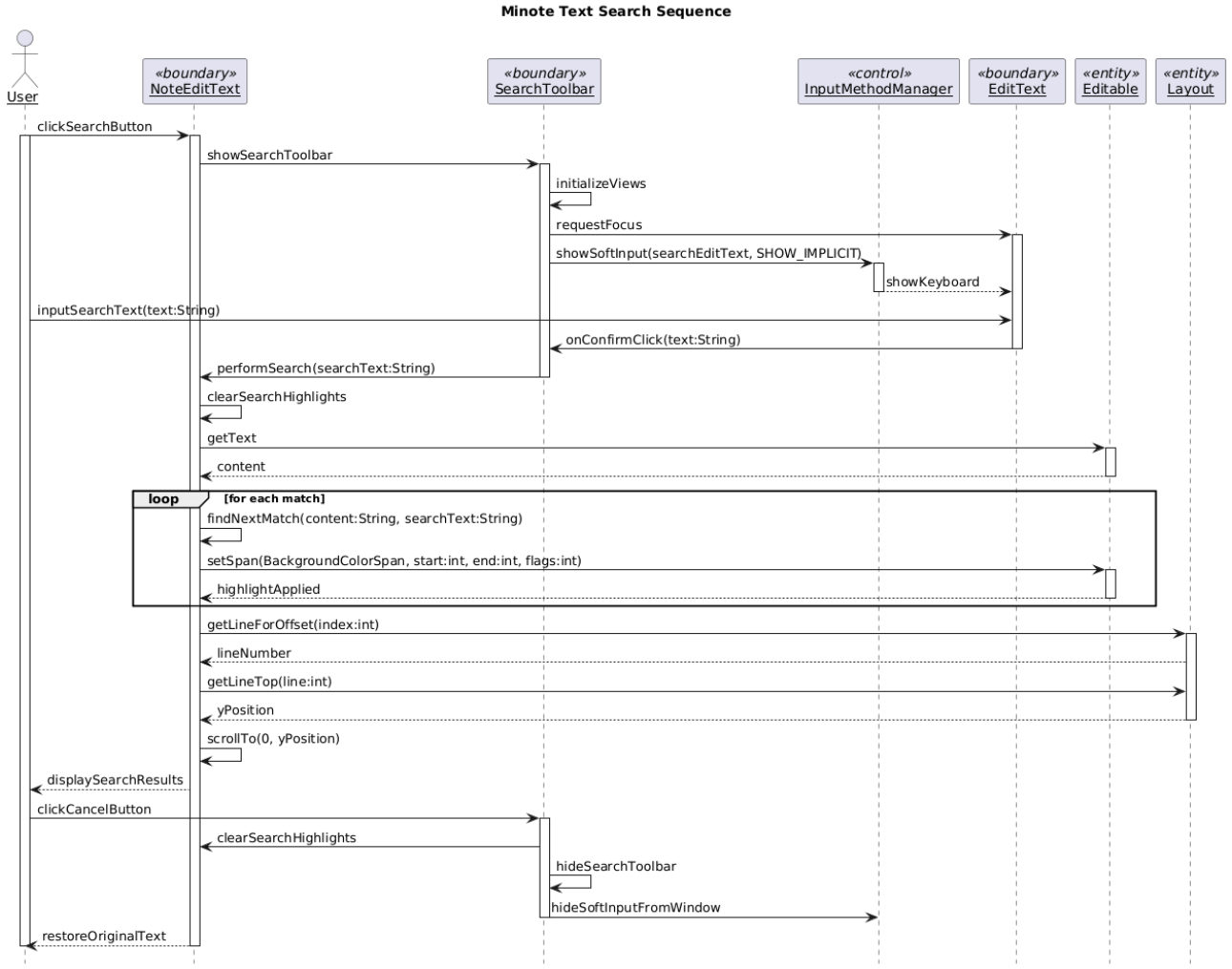


图 62: 笔记内搜索的时序图

3.4.11 模板便签

在便签编辑界面，点击模板按钮，可以对便签内容进行模板化。（见图63（a）、63（b））

图 63: 模板便签(a)

图 64: 模板便签(b)


Running Devices - xiaomi-src
Running Devices:
Pixel 8 Pro API 34
3:09
3G
字符数：22
朗读
翻译
1/8/2025,
8:13 AM
admin:qxq
passwd:123456
点击获得地理信息
星火
听写
1:1

图 65: 模版1

图 66: 模版2

模板便签功能允许用户快速选择预设的便签格式，以提高便签创建的效率。用户可以在编辑界面中点击模板按钮，系统将展示可用的模板列表。选择一个模板后，便签内容将自动填充为该模板的格式，用户只需进行必要的修改即可。此功能特别适合需要频繁使用相似格式的用户，能够有效节省时间并保持便签的一致性。

时序图如下图所示

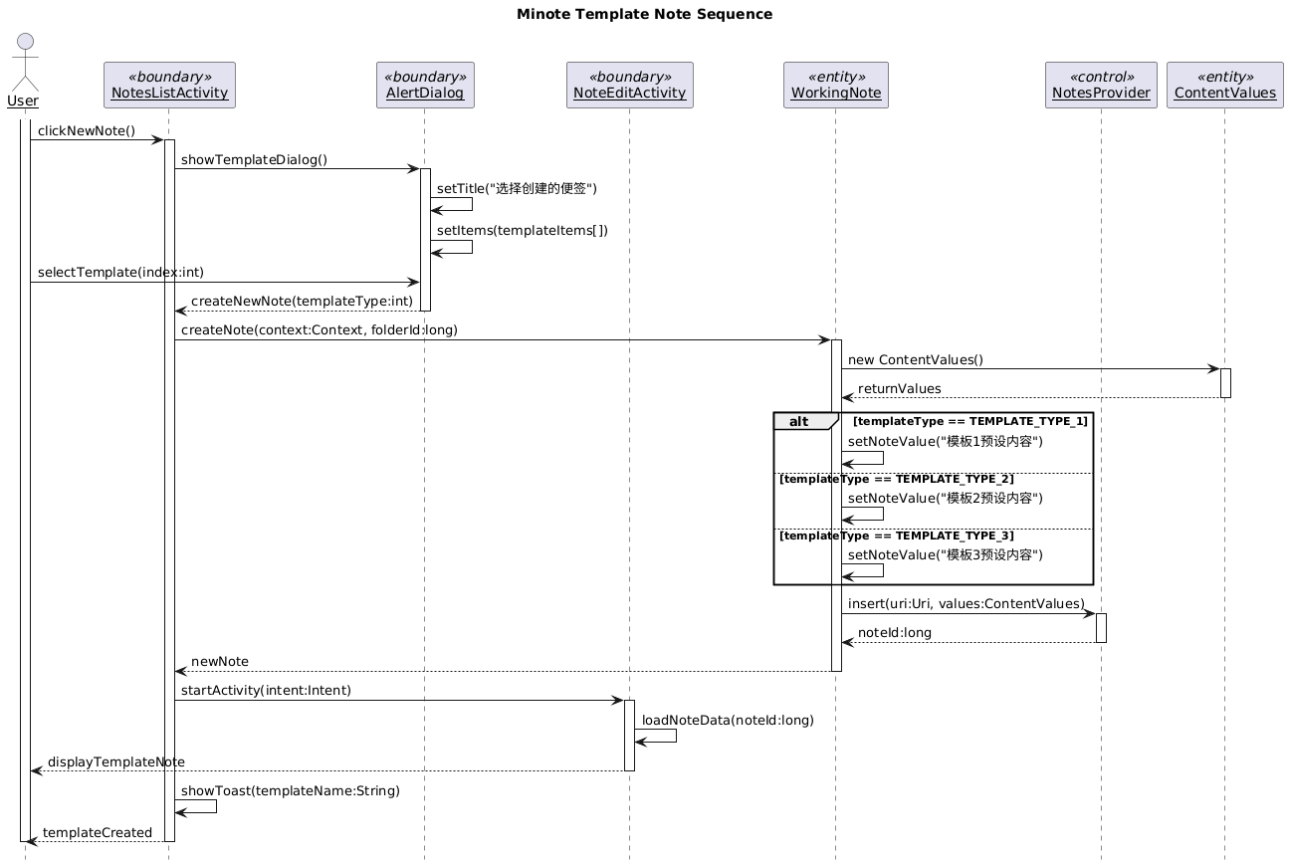


图 67: 模板便签的时序图

3.4.12 语音听写

在便签编辑界面，点击语音听写按钮，可以对便签内容进行语音听写。（见图68（a）、68（b））

时序图如图70所示。

图 68: 语音听写(a)

图 69: 语音听写(b)

通过语音听写功能,用户可以直接通过语音输入内容,无需手动输入文字。具体使用步骤如下:

1. 在便签编辑界面点击语音听写按钮,进入语音听写模式
2. 系统会调用手机麦克风,用户可以开始说话
3. 语音识别引擎会实时将用户的语音转换为文字
4. 转换后的文字会自动插入到当前光标位置
5. 用户可以随时暂停或结束语音输入

该功能支持中文普通话识别,可以较为准确地识别日常用语和专业术语。语音识别过程在本地完成,保护用户隐私。同时支持实时预览识别结果,方便用户及时纠错。这大大提高了便签记录的效率,特别适合需要快速记录想法或无法使用手写输入的场景。

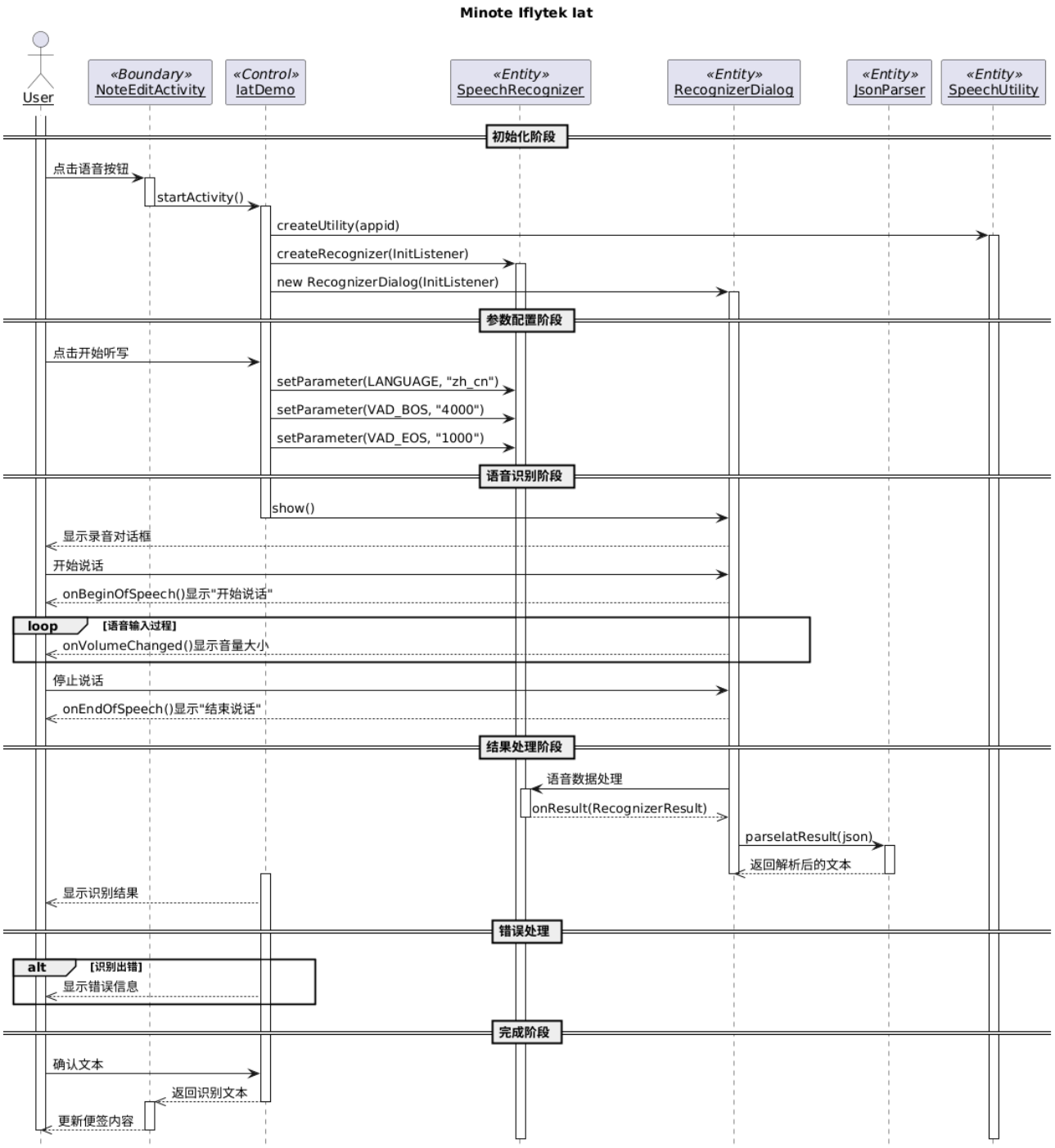


图 70: 语音听写的时序图

下图所示为语音听写的部分代码展示，主要展示了核心类mRecongnizerDialogListner的创立。

```
302     private void printResult(RecognizerResult results) {  2 usages
317             resultBuffer.append(mIatResults.get(key));
318         }
319         mResultText.setText(resultBuffer.toString());
320         mResultText.setSelection(mResultText.length());
321     }
322
323     /**
324      * 听写UI监听器
325      */
326     private RecognizerDialogListener mRecognizerDialogListener = new RecognizerDialogListener() {  1 usage
327         // 返回结果
328         public void onResult(RecognizerResult results, boolean isLast) { printResult(results); }
331
332         // 识别回调错误
333         public void onError(SpeechError error) { showTip(error.getPlainDescription( b: true)); }
336
337     };
338
339
340     private void showTip(final String str) {  17 usages
341         if (mToast != null) {
342             mToast.cancel();
343         }
344         mToast = Toast.makeText(getApplicationContext(), str, Toast.LENGTH_SHORT);
345         mToast.show();
346     }
347
348     /**
```

图 71: 语音听写的代码展示

配合流程图，可以看到核心类mRecongnizerDialogListner的创立，以及其内部方法的调用的局部，详情可见代码IatDemo。

3.4.13 语音合成

在便签编辑界面，点击语音合成按钮，可以对便签内容进行语音合成。（见图72 ）点击语音合成按钮后，会弹出语音合成界面，输入要合成的内容，点击确定按钮，可以合成语音。（见图73、74）

时序图如图69所示。

图 72: 语音合成

该语音合成功能，还支持语音类型和音频类型，见图67、68。

图 73: 语音类型

图 74: 音频类型

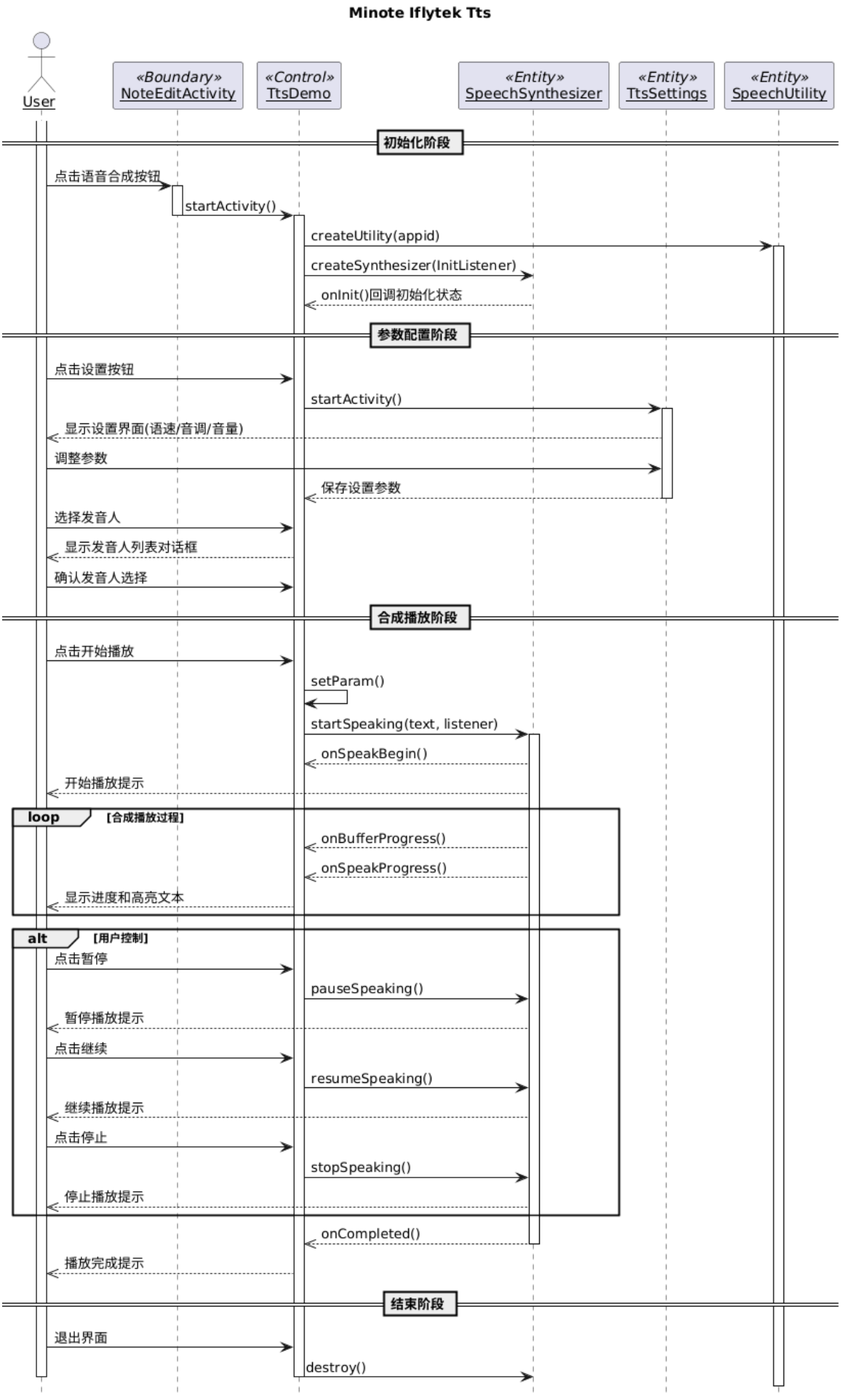


图 75: 语音合成的时序图

下图所示为语音合成的部分代码展示，主要展示了各控件的具体含义与功能实现。另外也注明了控件对应方法的参数与注意事项，详情可见代码TtsDemo。

```
109     public void onClick(View view) {
117         case R.id.image_tts_set:
118             if (SpeechConstant.TYPE_CLOUD.equals(mEngineType)) {
119                 Intent intent = new Intent( packageContext: TtsDemo.this, TtsSettings.class);
120                 startActivity(intent);
121             } else {
122                 showTip( str: "请前往xfyun.cn下载离线合成体验");
123             }
124             break;
125         // 开始合成
126         // 收到onCompleted 回调时，合成结束、生成合成音频
127         // 合成的音频格式：只支持pcm格式
128         case R.id.tts_play:
129             pcmFile = new File(getExternalCacheDir().getAbsolutePath(), child: "tts_pcmFile.pcm");
130             pcmFile.delete();
131             texts = ((EditText) findViewById(R.id.tts_text)).getText().toString();
132             // 设置参数
133             setParam();
134             // 合成并播放
135             int code = mTts.startSpeaking(texts, mTtsListener);
136 //          /**
137 //           * 只保存音频不进行播放接口,调用此接口请注释startSpeaking接口
138 //           * text:要合成的文本，uri:需要保存的音频全路径，listener:回调接口
139 //           */
140 //              String path = getExternalFilesDir("msc").getAbsolutePath() + "/tts.pcm";
141 //              //  synthesizeToUri 只保存音频不进行播放
142 //              int code = mTts.synthesizeToUri(texts, path, mTtsListener);
143
144             if (code != ErrorCode.SUCCESS) {
145                 showTip( str: "语音合成失败,错误码: " + code + ",请点击网址https://www.xfyun.cn/document/error-code查询解决方案");
146             }
147             break;
148         // 取消合成
149         case R.id.tts_cancel:
150             mTts.stopSpeaking();
151             break;
152         // 暂停播放
153         case R.id.tts_pause:
```

图 76: 语音合成的代码展示

3.4.14 对话式大模型

在便签编辑界面和接入了对话式大模型，可以对便签内容进行对话式大模型交互。（见图77）

对话式大模型的流程图见图78，时序图见图79。

图 77: 对话式大模型

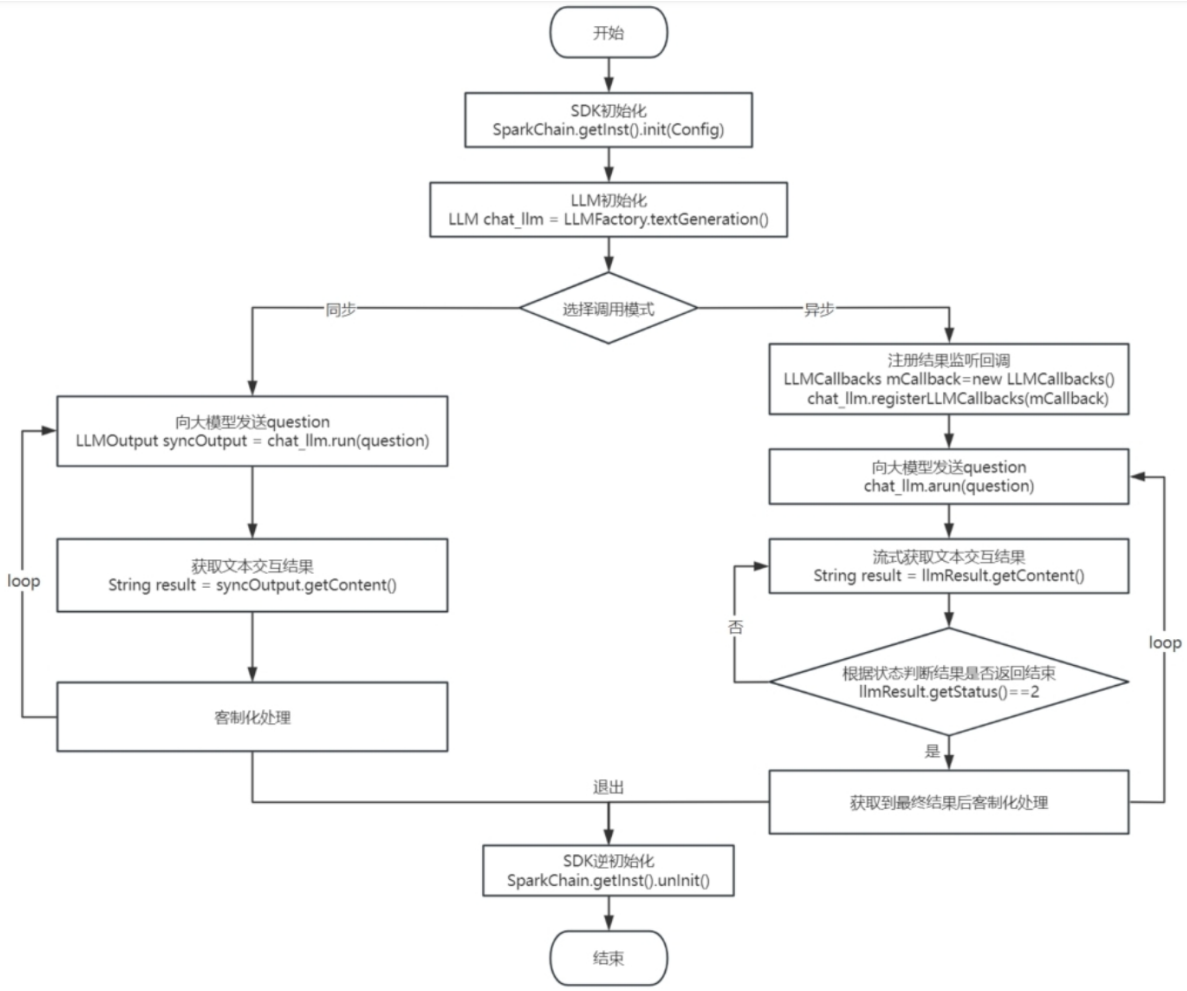


图 78: 对话式大模型的流程图

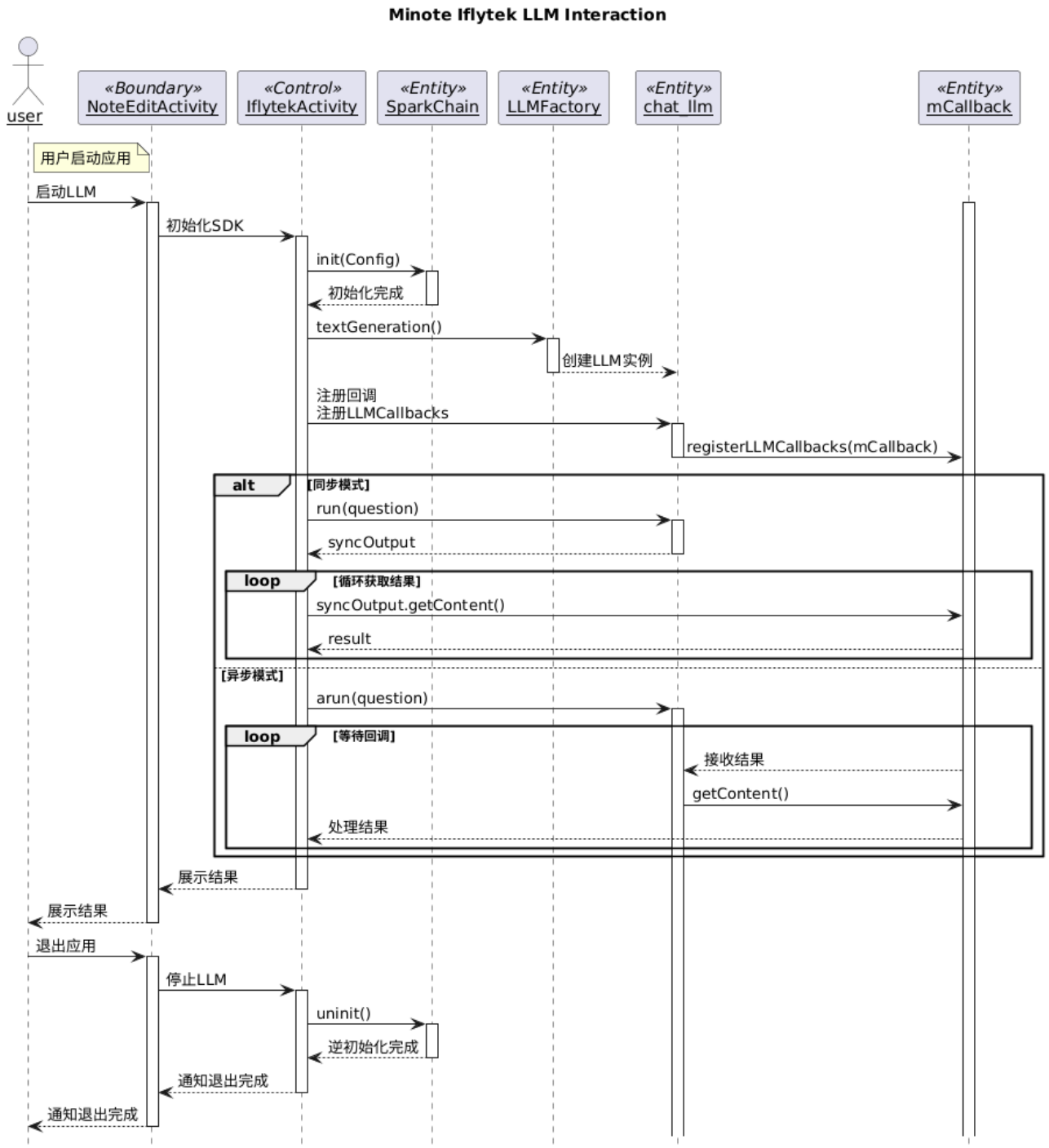


图 79: 对话式大模型的时序图

下图所示为对话式大模型的部分代码展示，主要展示了大模型交互类的配置信息，以及调试所用到的JSON格式。

```
212     @Override
213     protected void onDestroy() {
214         super.onDestroy();
215         SparkChain.getInst().unInit();//qxq:逆初始化
216     }
217     /***************
218      * 配置文本交互LLM，注册结果监听回调
219      * ******************/
220     private void setLLMConfig(){  1 usage
221         Log.d(TAG, msg: "setLLMConfig");
222         LLMConfig llmConfig = LLMConfig.builder()
223                 .domain( s: "generalv3");
224         llmConfig.showRefLabel( b: true);//返回信源信息，4.0Utral版本支持，其他版本传递无效。
225 //        Memory window_memory = Memory.windowMemory(5);
226         llm = LLMFactory.textGeneration(llmConfig);
227 //        String rawJson = "{\n" +
228 //                "        \"header\": {\n" +
229 //                "            \"app_id\": 30dfb58a,\n" +
230 //                "            \"uid\": \"1234\",            \n" +
231 //                "            # \"patch_id\": []    #接入微调模型，对应服务发布后的resourceid          \n" +
232 //                "        },\n" +
233 //                "        \"parameter\": {\n" +
234 //                "            \"chat\": {\n" +
235 //                "                \"domain\": 4.0Ultra,\n" +
236 //                "                \"temperature\": 0.5,\n" +
237 //                "                \"max_tokens\": 4096,\n" +
238 //                "                \"auditing\": \"default\",\n" +
239 //                "            }\n" +
240 //                "        },\n" +
241 //                "        \"payload\": {\n" +
242 //                "            \"message\": {\n" +
243 //                "                \"text\": [{\"role\": \"user\", \"content\": 给我写一篇100字的作文}]\n" +
244 //                "            }\n" +
245 //                "        }";
246 //        int ret = llm.arunWithJson(rawJson,token);
247         llm.registerLLMCallbacks(llmCallbacks);//注册回调函数
248 
249     }
```

图 80: 对话式大模型的代码展示

这里对对话式大模型以及语音合成与转写功能进行一部分特殊说明，由于使用了第三方SDK并且需要与第三方服务器（Iflytek Server等）连接，因此虚拟机必须需要连接网络（这一点百度翻译也是一样的）。其次，讯飞星火大模型和语音识别功能还依赖官方提供的特定MSC和aar文件，这些库依赖需要在build.gradle中额外声明（包括dependencies依赖项声明和依赖树结构），对应的MSC和AAR文件已经上传至项目中的源码文件夹下的app子包的libs包中，值得一提的是，该MSA和AAR文件都只支持arm64-v8a或armeabi-v7a处理器，部分虚拟机可能不具备调试的前提；最后，讯飞的SDK服务有一定期限，到期后需要用户重新购买服务，并在本项目中替换appid、密钥等信息。 MSC的部分文件如表6所示：

表 6

序号	文件名	说明
1	接口InitListener	初始化回调接口
2	接口SpeechListener	通用回调接口
3	类SpeechRecognizer	语音识别类
4	类SpeechSynthesizer	语音识别类
5	类SpeechUtility	SDK初始化类

3.4.15 撤回功能

在进行便签编辑时，有时候会不小心错误的操作，这时候就需要撤回功能来恢复之前的状态。（见图81）点击菜单栏中的撤回按钮，可以恢复到上一个状态。

图 81: 撤回功能

具体时序图如图82所示。

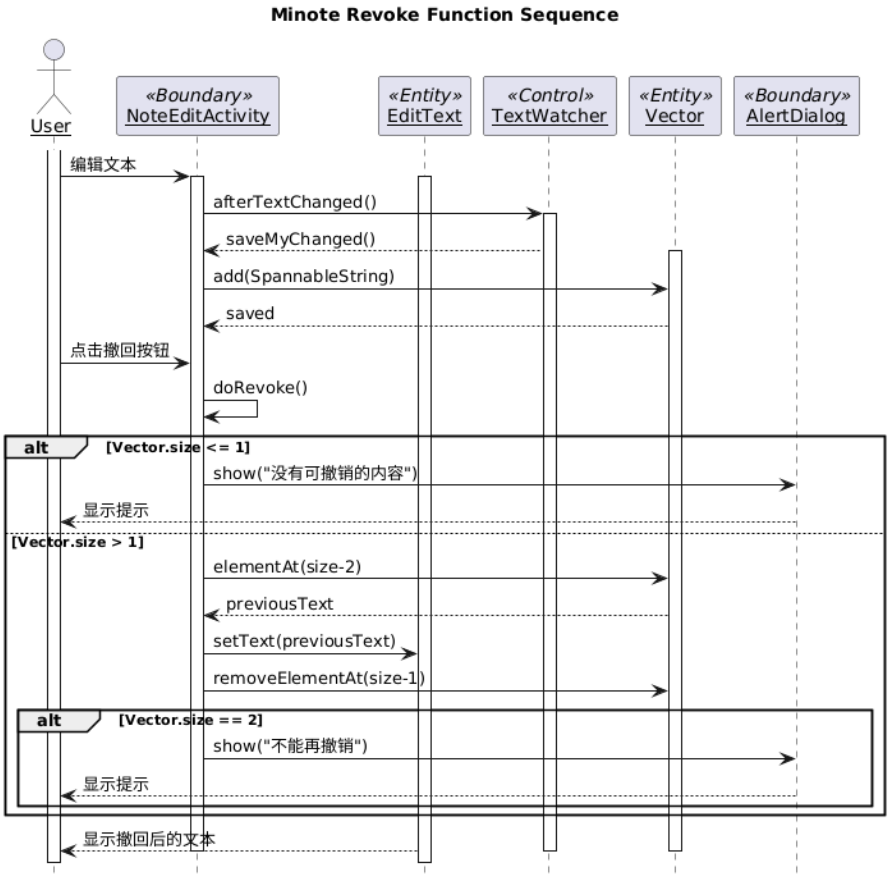


图 82: 撤回功能的时序图

撤回功能的实现比较简单，点击事件主要依靠一个监听器对存储曾经编辑的便签内容（存储在一个栈中）进行监听，然后通过一个按钮的点击事件，将曾经编辑的便签内容恢复到当前编辑界面中。

3.4.16 获取地理位置

在编辑编辑的下方，有一个获取地理位置按钮，点击该按钮，弹出功能界面，有两个选项获取地理位置和清楚地理位置。

点击获取地理位置按钮，可以获取当前的地理位置。（见图83）其地理位置为虚拟机设定的地理位置，可以在Android Studio中的Device Manager中添加设定自己IP所在位置，也可以沿用虚拟机本身的设置。本图中返回的西经122°，北纬37°经查阅为北美洲加利福尼亚州地区的西南沿岸，与中国时差约为10小时左右，时差和虚拟机-真机时差符合的比较好，可见功能的正确性。

点击清除地理位置按钮，可以清除当前的地理位置。（见图84）

图 83: 获取地理位置

图 84: 清除地理位置

具体时序图如图85所示。

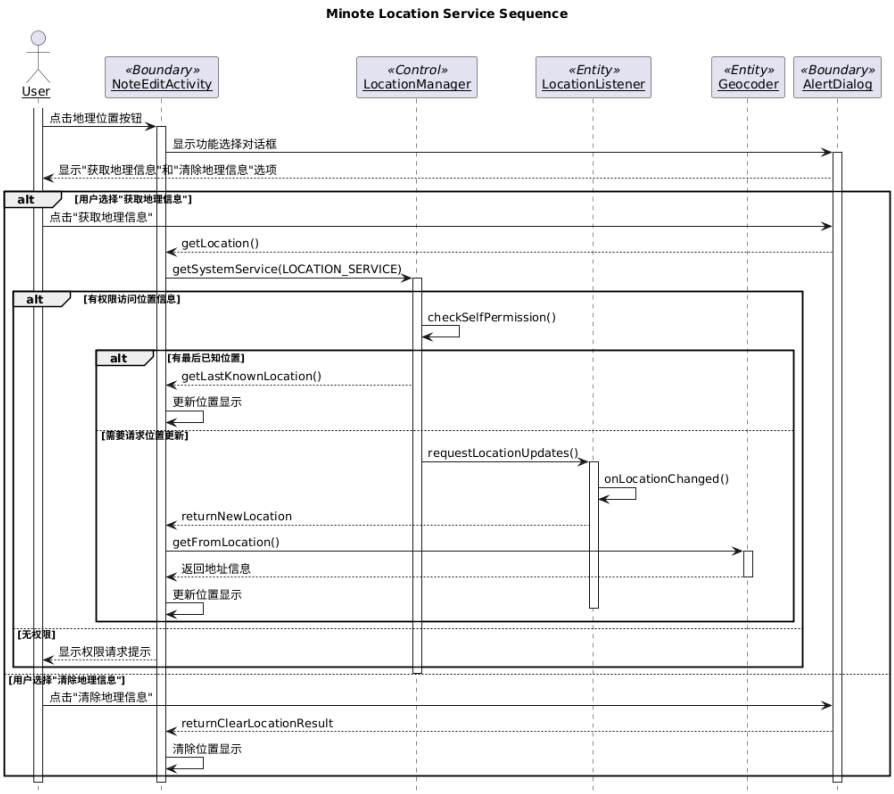


图 85: 获取地理位置的时序图

4 实践收获和体会

4.1 收获

1. **对代码分析方法的深入掌握**
   从面对大规模代码的初步略读到精读，再到实际的维护，实现了完整的开发流程实践。通过逐步分析代码的用例、各个包和类之间的逻辑关系，熟悉了复杂软件架构的设计理念。在代码标注过程中，提升了对代码细节的把控能力，包括对重要变量、方法功能的精确说明和逻辑梳理。

2. **Java语言和软件工程能力的提升**
   学会了Java语言的使用规范，能够撰写符合高质量标准的代码。熟练掌握了对Java代码的深度阅读与优化方法，能够独立完成对代码功能的扩展与维护。

3. **代码质量分析能力的提升**
   熟悉了多种代码质量分析方法，包括手工分析和工具辅助分析（如Sonarqube）。特别是通过工具发现和修复代码中的潜在问题，从而提高了代码的可维护性与稳定性。

4. **团队协作能力的提高**
   通过将大规模工作细分为具体任务，合理分工并协调团队成员的进度，确保了项目的顺利完成。在项目推进过程中，学会了如何高效沟通与协作，特别是在问题集中处理和解决上形成了有效的团队策略。

5. **用户体验设计的优化意识**
   在维护工作中新增了大量以用户需求为导向的功能，如翻译功能、语音听写、模板便签等，这些实践增强了对用户体验设计的理解与感知。

4.2 体会

1. **应对复杂代码架构的挑战**
   面对大规模、复杂的代码体系，初期分析时会因为包和类之间的逻辑关系复杂而产生理解偏差，需要反复调试与验证才能准确把握整体架构。在实现新功能时，需要权衡新增功能对现有系统的影响，并在开发中保持代码风格与原有代码一致，确保项目的整体性。

2. **团队分工和协作的重要性**
   项目初期团队分工不够明确，导致了部分工作的重复与进度缓慢。通过明确分工和定期检查进度，有效解决了这些问题，确保了项目按计划完成。

3. **代码维护与优化的思维转变**
   在实践中逐渐认识到，代码维护不仅仅是功能新增，还包括对原有代码的优化、注释的

补充以及对可读性和扩展性的提升。尤其是通过代码质量工具的分析，进一步理解了高质量代码的标准与细节，并体会到提升代码质量对于后续维护的重要性。

4. **对软件工程整体流程的深刻理解**
   从代码分析、功能设计到质量检验，再到最终的产品交付，整个过程强化了对软件工程开发流程的认知。体会到软件开发不仅是技术实现，还包括规划、沟通与持续优化。

4.3 参考文献与网址

4.3.1 讯飞智能API介绍

讯飞开放平台文档中心 https://www.xfyun.cn/doc/mscapi/Android/androidinit.html

4.3.2 百度翻译API介绍

百度翻译开放平台文档中心 https://fanyi-api.baidu.com/doc/11

4.3.3 UML建模

UML建模工具：https://plantuml.com/zh/

4.3.4 CodeArts质量分析

CodeArts质量分析工具：https://www.huaweicloud.com/product/codearts/quality.html