**OpenHarmony竞赛训练营**

**智能日程表应用**

|  |  |
| --- | --- |
| 学校名称： | 华中科技大学 |
| 团队名称： | 智8小队 |
|  |  |
| 队长： | 何沁遥 |
| 队员1： | 徐楚雯 |
| 队员2： | 张施琪 |

OpenHarmony竞赛训练营组委会

2024年10月7日

# 摘 要

**背景：**

在数字化时代，随着智能设备的广泛普及和生活节奏的不断加快，人们对于时间管理和日程安排的需求变得前所未有的迫切。

在这样的背景下，我们设计了一款基于OpenHarmony和AI的智能日程管理应用。该应用深度聚焦用户体验，不仅拥有直观、易用的UI界面，还集成了先进的智能日程解析、提醒和冲突检测等功能。它旨在帮助用户以更高的效率规划每一天，确保不错过任何重要的活动或任务。

通过这款应用，用户可以轻松管理自己的时间，享受更加有序和便捷的生活体验，无论是工作安排还是个人生活，都能够轻松掌握。

**目的：**

该项目致力于解决现代生活中日益增长的时间管理挑战，为用户提供一种更加高效、直观的日程安排。在智能设备无处不在、生活节奏日益加快、工作压力不断增长的今天，人们迫切需要一种能够轻松应对复杂日程的工具，以确保时间得到最佳利用。

项目的核心目的在于通过开发一款基于OpenHarmony的智能日程管理应用，帮助用户优化日程管理。通过手动日程添加、智能日程解析等基本功能，用户能够轻松记录和组织生活中的每一个重要时刻。进阶功能的加入，如日程冲突检测、智能提醒，进一步提升了应用的实用性和智能化水平，使用户能够更加精准地掌握时间，避免日程冲突，提高生活和工作效率。

**关键词：**日程管理，智能日程解析

目 录

[摘 要 2](#_Toc179195776)

[1 设计需求分析 4](#_Toc179195777)

[1.1 主要解决问题 4](#_Toc179195778)

[1.2 改进点 4](#_Toc179195779)

[2 特色与创新 5](#_Toc179195780)

[3 设计描述 5](#_Toc179195781)

[3.1总体设计 5](#_Toc179195782)

[3.2实现思路 6](#_Toc179195783)

[3.3系统结构 7](#_Toc179195784)

[3.3.1 模块划分 7](#_Toc179195785)

[3.3.2 系统架构说明 8](#_Toc179195786)

[3.3.3 文件结构 10](#_Toc179195787)

[3.4 模块功能描述 11](#_Toc179195788)

[3.4.1 视图模块功能描述 11](#_Toc179195789)

[3.4.2 控制器模块功能描述 12](#_Toc179195790)

[3.4.3 模型模块功能描述 14](#_Toc179195791)

[3.5业务/实现流程说明 15](#_Toc179195792)

[3.5.1 添加日程处理流程 15](#_Toc179195793)

[3.5.2 删除日程处理流程 16](#_Toc179195794)

[3.6接口描述 17](#_Toc179195795)

[3.6.1 调用接口 17](#_Toc179195796)

[3.6.2 提供接口 18](#_Toc179195797)

[3.7 UI设计 18](#_Toc179195798)

[4 其他 19](#_Toc179195799)

[4.1 成员分工 19](#_Toc179195800)

[4.2困难与思考 20](#_Toc179195801)

[4.3参考 20](#_Toc179195802)

# **1**设计需求分析

## 主要解决问题

本项目解决了用户在日常生活和工作中管理日程时所面临的繁琐手动输入问题。在传统的日程管理工具中，用户通常需要逐一输入每个日程的详细信息，这不仅耗时，而且容易出错。

因此我们归纳和总结了当前用户的主要需求：

1. **简化输入流程**：用户希望能够快速、便捷地添加日程，而不是进行繁琐的手动输入。
2. **提高信息准确性**：用户希望能够减少因手动输入而导致的错误，确保日程信息的准确性。
3. **高效的信息提取**：在信息来源多样化的今天，用户希望能够直接从社交平台或通知中提取关键信息，减少手动操作。
4. **智能化管理**：用户希望拥有智能化的管理工具，能够根据其日程自动提供提醒和冲突检测，以优化时间安排。

针对这些需求，我们的应用通过集成AI智能解析功能，允许用户直接将复制的通知内容粘贴到日程表中，AI系统会自动识别并提取有效信息，如事件的标题、日期和时间，并生成相应的日程块，大幅提升了用户在日程安排上的效率，减轻了他们的负担，让他们能够更专注于重要事务的处理。

## 改进点

本项目在基础的日程管理功能上进行了多项重要改进。

1. **AI智能日程解析：**首先，加入的AI智能日程解析功能显著减少了用户手动输入的需要，提升了使用便利性，用户只需简单粘贴通知消息，系统即可自动生成日程，避免了手动输入过程中可能出现的错误。
2. **直观易用的用户界面**：其次，在用户界面的设计上，我们注重直观性和易用性，用户可以通过清晰的日历视图快速浏览和管理自己的日程。这种设计确保用户能够在最短时间内找到所需信息，做出更快速的决策。
3. **冲突检测功能：**不仅如此，我们加入了冲突检测功能，当用户尝试添加新日程时，系统会自动检查是否与现有日程发生时间冲突，并及时通知用户，帮助用户有效避免日程冲突，提升日程安排的整体效率和合理性。
4. **智能提醒功能：**此外我们还引入了智能提醒功能，应用能够在事件开始前15分钟自动发送提醒通知，帮助用户合理安排时间，避免错过重要日程。
5. **模糊日程处理功能：**当在用户遇到时间信息不明确的任务时，AI智能助手会根据任务的性质智能地为任务分配合理的时间长度，并将其妥善安排进用户的日程表中，确保时间管理的连贯性和有效性。

综上所述，这些改进不仅提升了应用的实用性，也增强了用户体验，使其在日程管理上变得更加高效和智能。

# **2**特色与创新

本项目与现有其他应用相比结合AI进行了许多自己的创新。

1. **AI的自然语言处理技术：**本项目结合了AI的自然语言处理技术，用户在管理日程时可以通过简单的复制和粘贴操作，快速生成日程块，这一功能与传统的日程管理应用相比，极大地降低了信息录入的复杂性，提高了用户的使用效率。除此之外，当在用户输入时间信息不明确的日程信息时，AI智能助手会进行模糊处理，即根据日程性质来估算时间长度，将其妥善安排进用户的日程表中，确保时间管理的连贯性和有效性。
2. **冲突检测：**用户在添加新日程时，系统会自动检查时间冲突，并及时提醒用户，避免了因安排重叠导致的时间浪费和计划混乱。这一功能不仅提高了日程管理的合理性，也帮助用户更有效地安排时间，提升了整体的工作效率。
3. **便捷的下拉输入方式：**应用支持通过下拉通知栏直接进入与AI对话页面。当应用监听到剪贴板内容发生变化时，会及时发送通知栏的通知，用户只需下拉通知栏，点击进入即可将复制的内容粘贴到日程表中。这种设计大大简化了日程输入流程，使得用户能够迅速添加新日程，提升了整体效率。

# **3**设计描述

## **3.1改进点**

根据上述设计需求分析，我们相应地设计了以下多种功能：

1. 基础功能
   1. **手动添加日程功能**：用户可以手动输入日程安排，系统提供简洁的UI界面以方便操作。
   2. **删除/编辑日程功能**：用户通过点击日程块，可以便捷地在弹出的日程信息弹窗中选择删除或编辑操作，实现对该日程块的灵活管理。
2. 进阶功能
   1. **一键删除日程功能：**用户可以选择一键删除当天日程或者日程表中所有日程，不必逐个对日程块进行删除操作。
   2. **AI智能助手功能：**用户输入语句即可与AI进行对话。
   3. **智能日程解析功能：**通过粘贴社交平台（如QQ群、微信群）的通知消息，AI能够自动识别并提取日程的关键信息，如地点、时间、描述等，并自动加入到日程表中。
   4. **模糊日程处理功能：**在用户遇到时间信息不明确的任务时，AI智能助手将自动进行时间估算，确保任务被合理地安排进日程表中。
   5. **日程冲突检测功能：**系统能检测并提醒用户日程安排中的冲突，避免时间上的重叠。
   6. **通知栏快捷入口功能：**当应用检测到剪贴板内容更新时，它会立即触发一个通知栏提醒。用户只需下拉通知栏并轻触通知，即可直接跳转至AI智能助手对话界面，用户可以直接粘贴剪贴板中的内容，与助手进行交流。
   7. **智能日程提醒功能：**根据比对日程时间，通过下拉式通知栏弹窗提醒用户即将到来的日程。

## **3.2**实现思路

对于以上这些主要的功能，我们结合OpenHarmony对应的模块，一一实现了它们的功能：

1. **手动添加日程功能**：使用了@ohos.data.relationalStore模块管理数据，使用了@ohos.data.ValuesBucket模块存储用户输入的日程，并将其存储到数据库中，使用relationalStore.RdbStore 提供的 API 进行插入操作，确保用户可以创建新日程。在成功插入日程信息后，使用@ohos.events.emitter模块发送事件，通知其他组件或页面更新数据。
2. **删除/编辑日程功能、一键删除日程功能**：使用relationalStore.RdbStore 提供的 API 进行删除和更新操作，确保用户可以删除或编辑已有日程。在成功删除或更改日程信息后，使用@ohos.events.emitter模块发送事件，通知其他组件或页面更新数据。
3. **AI智能助手功能**：使用了@ohos.net.http模块来实现与外部API的通信，主要用于发送用户的问题并接收助手的回答。
4. **智能日程解析功能、模糊日程处理功能**：在处理时间信息不确切的情况时，智能助手将运用其内置算法对时间进行智能估算，以确定最佳的日程安排。对于用户提供的明确的日程信息，将直接使用正则表达式来提取智能助手返回的日程信息，包括事件标题、描述、地点、日期、开始时间和结束时间等信息，并结合@ohos.data.relationalStore模块实现日程信息的存储。在成功插入日程信息后，使用@ohos.events.emitter模块发送事件，通知其他组件或页面更新数据。
5. **日程冲突检测功能：**通过构造SQL查询，利用三种条件检查新添加的日程与数据库中已有的日程是否存在时间重叠。在检测到冲突时，通过@ohos.promptAction模块向用户展示提示信息，确保用户能够及时了解问题，并做出相应的调整。
6. **通知栏快捷入口功能：**使用了@ohos.pasteboard模块对用户的剪贴板进行监听，当监听到用户的剪贴板内容变化时进行发布通知。通过@ohos.notificationManager和@ohos.app.ability.wantAgent模块，实现了通过通知栏直接进入应用特定页面的功能。用户在复制通知内容到剪贴板后即可点击通知进入与AI智能助手对话的页面。
7. **智能日程提醒功能：**结合了@ohos.data.relationalStore和@ohos.notificationManager模块，通过定时器setInterval定期检查即将到来的日程，确保在日程开始前15分钟发出提醒。

## 3.3系统结构

### **3.3.1模块划分**

根据应用的结构和功能需求，我们将系统划分为几个主要模块，以便于管理和维护。具体模块划分如下：

1. **视图模块**

·主界面：提供日程表的入口界面，用户可以在此界面查看和管理日程。

·日程添加界面：用户可以在此手动输入日程的标题、日期、时间等信息。

·智能助手界面：集成AI助手功能，帮助用户智能解析日程信息并添加进日程表。

1. **控制器模块**

·日程管理控制器：负责日程块的修改删除等操作，在数据库中更新数据。

·日程显示控制器：负责日程块的显示，从数据库中加载数据。

·日期时间选择控制器：负责在用户手动添加日程时提供日期与时间选择的窗口。

·通知管理控制器：处理与通知相关的逻辑，包括发布提醒和权限请求。

1. **模型模块**

·日期模型：保存对日期的所有处理，例如获取今天的日期、返回指定日期所在周的七天等。

·时间模型：保存对时间的所有处理，例如获取当前的时间、计算两个时间之间的时间间隔等。

·日程模型：定义日程的基本属性（如标题、时间、描述等）。

1. **系统结构**

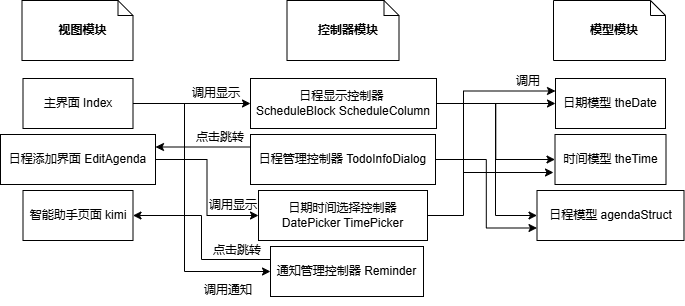


图3-1 系统结构uml图

### **3.3.2系统架构说明**

当用户首次启动应用时，系统会请求通知权限。这一流程由通知管理控制器负责，确保用户能够收到输入通知和日程提醒。

进入**主界面**后，系统会加载当前日期的日程，通过日程显示控制器从数据库获取数据并展示在界面上，涉及的模块如下图：

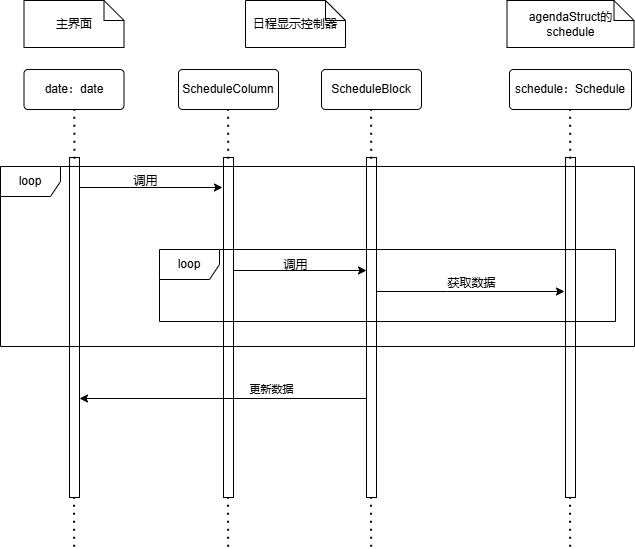


图3-2 主界面uml图

用户在主界面点击添加日程按钮，系统会导航到**日程添加界面。**在该界面，用户可以输入日程信息并提交，同时系统会检查日程是否存在冲突，确保无冲突后信息会被存入数据库并添加到日程表中，涉及的模块如下图：

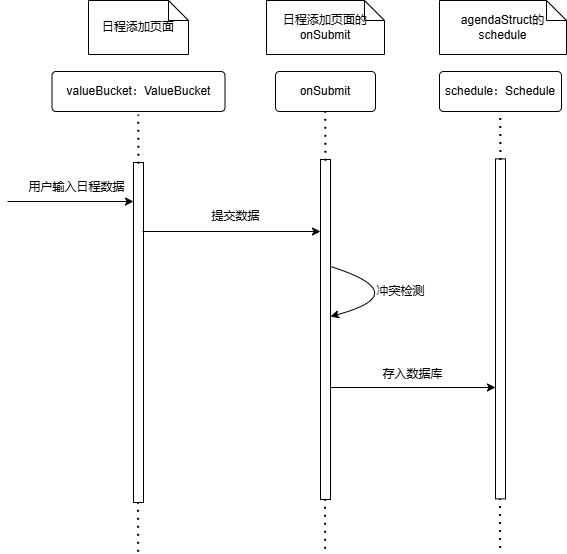


图3-3 日程添加页面uml图

通过通知栏点击或者主页面导航即可进入**智能助手界面**。在该界面，用户可以输入自然语言日程信息，系统会调用AI智能助手，以解析用户输入并将日程添加到日程表中，涉及的模块如下图：

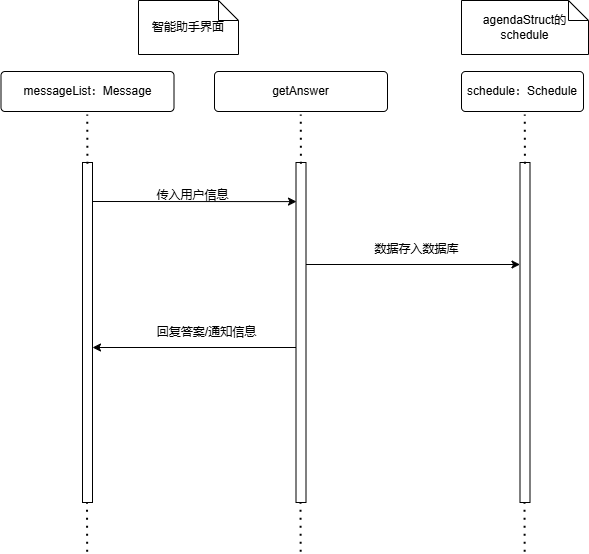


图3-4 智能助手界面uml图

### **3.3.3文件结构**

我们设计了以下文件结构：

|  |
| --- |
| ┣ main ┃ ┣ ets ┃ ┃ ┣ aiability ┃ ┃ ┃ ┗ AIAbility.ets //处理与智能助手相关的功能 ┃ ┃ ┣ DateModule //日期相关的模块 ┃ ┃ ┃ ┣ agendaStruct.ets //日程模型 ┃ ┃ ┃ ┣ DatePicker.ets //日期选择器 ┃ ┃ ┃ ┣ Reminder.ets //通知提醒 ┃ ┃ ┃ ┣ ScheduleBlock.ets //日程块 ┃ ┃ ┃ ┣ ScheduleColumn.ets //日程列 ┃ ┃ ┃ ┣ theDate.ets //日期模型 ┃ ┃ ┃ ┣ theTime.ets //时间模型 ┃ ┃ ┃ ┣ TimePicker.ets //时间选择器 ┃ ┃ ┃ ┗ TodoInfoDialog.ets //日程块信息弹窗 ┃ ┃ ┣ entryability ┃ ┃ ┃ ┗ EntryAbility.ets ┃ ┃ ┣ GlobalContext //全局上下文管理模块 ┃ ┃ ┃ ┗ GlobalContext.ts ┃ ┃ ┗ pages ┃ ┃ ┃ ┣ EditAgenda.ets //日程添加界面 ┃ ┃ ┃ ┣ Index.ets //主界面（日程表界面） ┃ ┃ ┃ ┗ kimi.ets //智能助手界面 ┃ ┣ resources ┃ ┃ ┣ base ┃ ┃ ┃ ┣ element ┃ ┃ ┃ ┣ media ┃ ┃ ┃ ┗ profile ┃ ┃ ┃ ┃ ┗ main\_pages.json ┃ ┃ ┣ en\_US ┃ ┃ ┣ rawfile ┃ ┃ ┗ zh\_CN ┃ ┗ module.json5 //配置文件 |

## 3.4 模块功能描述

### **3.4.1视图模块**

* **主界面 Index.ets**

1. 类型：视图组件
2. 目的：提供日程表的入口界面，允许用户查看、管理日程并进行相关操作。
3. 功能列表：

|  |
| --- |
| Index |
| MyMenu() //显示智能助手和一键删除的菜单 |
| requestNotificationPermission() //请求通知授权 |
| publishNotification() //通知栏显示 |
| aboutToAppear() //页面即将显示时请求授权和更新当前日期 |
| build() //显示UI，导航栏构建添加按钮和菜单按钮，主体部分为日程表 |
| pageTransition() //页面进场和离场动画 |
| DateTitle() //渲染每一天的标题 |
| deleteSchedule() //删除日程 |

* **日程添加界面 EditAgenda.ets**

1. 类型：视图组件
2. 目的：提供用户添加日程界面，允许用户创建或编辑日程，包括设置标题、描述、地点、时间和颜色等信息。
3. 功能列表：

|  |
| --- |
| EditAgenda |
| aboutToAppear() //初始化保存日程信息的结构体 |
| build() //显示UI，提供输入信息的文本框、日期和时间选择弹窗和取消及提交按钮 |
| onSubmit() //提交日程 |
| checkScheduleConflict() //检查日程是否冲突 |

* **智能助手界面 kimi.ets**

1. 类型：视图组件
2. 目的：提供用户与智能助手交互的界面，允许用户输入问题并获取日程安排的智能回复。
3. 功能列表：

|  |
| --- |
| kimi |
| build() // 显示UI，包含消息列表和输入框 |
| getAnswer() // 发送HTTP请求获取智能助手的回复 |
| extractDataFromResponse() // 从智能助手回复中提取日程信息 |
| insertScheduleToDB() // 将提取的日程信息插入数据库 |

### 

### **3.4.2控制器模块**

* **日程管理控制器 ScheduleColumn.ets**

1. 类型：视图组件
2. 目的：展示特定日期的日程安排，允许加载并显示当天的所有日程信息。
3. 功能列表：

|  |
| --- |
| ScheduleColumn |
| aboutToAppear() // 订阅事件以加载日程数据，并调用加载函数 |
| loadSchedulesFromDb() // 从数据库加载指定日期的日程数据，并更新状态 |
| build() // 渲染日程列，循环展示每个日程块 |

* **日程管理控制器 ScheduleBlock.ets**

1. 类型：视图组件
2. 目的：显示单个日程块，提供日程信息并响应用户点击以展示详细信息。
3. 功能列表：

|  |
| --- |
| ScheduleBlock |
| build() // 渲染日程块，展示日程标题，并设置点击事件以打开日程信息弹窗 |
| calculateYPosition() // 计算日程块在页面上的垂直位置 |
| calculateHeight() // 计算日程块的高度，基于日程的持续时间 |

* **日程显示控制器 TodoInfoDialog.ets**

1. 类型：自定义对话框组件
2. 目的：展示日程的详细信息，并提供删除和编辑日程的功能。
3. 功能列表：

|  |
| --- |
| TodoInfoDialog |
| build() // 渲染日程信息对话框，显示日程名称、时间、地点和备注，并提供删除和编辑按钮 |

* **日期选择控制器 DatePicker.ets**

1. 类型：日期选择组件
2. 目的：允许用户选择日期并在界面上显示所选日期，提供自定义对话框用于日期选择。
3. 功能列表：

|  |
| --- |
| DatePicker |
| build() // 渲染日期显示区域，用户点击后打开日期选择对话框 |
| DateSelectDialog() // 自定义对话框，包含日期选择器，允许用户选择日期并更新所选日期 |

* **时间选择控制器 TimePicker.ets**

1. 类型：时间选择组件
2. 目的：允许用户选择时间并在界面上显示所选时间，提供自定义对话框用于时间选择。
3. 功能列表：

|  |
| --- |
| TimePicker |
| build() // 渲染时间显示区域，用户点击后打开时间选择对话框 |
| TimeSelectDialog() // 自定义对话框，包含时间选择器，允许用户选择时间并更新所选时间 |

* **通知管理控制器 Reminder.ets**

1. 类型：通知提醒类
2. 目的：检查并发布日程提醒通知，提醒用户即将开始的日程
3. 功能列表：

|  |
| --- |
| Reminder |
| checkReminders() // 检查指定日期的日程，判断是否需要发布提醒通知。 |
| publishNotification() // 根据日程信息生成并发布通知，提醒用户日程即将开始。 |

### **3.4.3模型模块**

* **日期模型 theDate.ets**

1. 类型：日期处理类
2. 目的：提供对日期的各种操作和格式转换，包括获取今天的日期、计算周内日期、解析日期等功能
3. 功能列表：

|  |
| --- |
| theDate |
| today() // 获取今天的日期，格式为 'yyyy-MM-dd' |
| parseDate() // 将 'yyyy-MM-dd' 字符串转化为 Date 类型 |
| daysInTheSameWeek() // 返回指定日期所在周的七天，格式为 'yyyy-MM-dd' |
| getYear() // 获取指定日期的年份 |
| getMonth() // 获取指定日期的月份（1 月 = 1，12 月 = 12） |
| getDate() // 获取指定日期是该月的第几天 |
| getDay() // 获取指定日期是周几（0 = 周日，1 = 周一，6 = 周六） |
| date2yyyyMMDD() // 将 Date 对象转换为 'yyyy-MM-dd' 字符串 |
| dateWithOffset() // 根据偏移量调整日期，并返回新的日期（格式为 'yyyy-MM-dd'） |
| getWeekday() // 获取指定日期对应的星期几的名称 |
| isToday() // 判断指定日期是否是今天 |

* **时间模型 theTime.ets**

1. 类型：时间处理类
2. 目的：提供对时间的各种操作和格式转换，包括获取当前时间、时间间隔计算、时间偏移等功能。
3. 功能列表：

|  |
| --- |
| theTime |
| date2hhmm() // 将 Date 对象转换为 'hh' 格式的时间字符串 |
| now() // 获取当前时间，返回 'hh' 格式的时间字符串 |
| getHours() // 从时间字符串（'hh' 格式）中提取小时部分 |
| getMinutes() // 从时间字符串（'hh' 格式）中提取分钟部分 |
| getDuration() // 计算两个时间之间的时间间隔（以分钟为单位） |
| hhmm2position() // 根据时间获得位置（以分钟为单位） |
| timeWithOffset() // 时间加上偏移量（以分钟为单位），确保结果在当天范围内 |

* **日程模型 agendaStruct.ets**

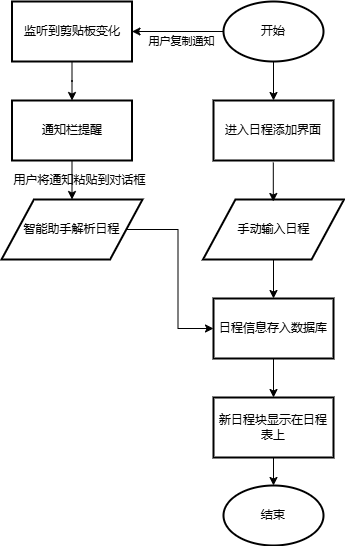
1. 类型：日程数据结构
2. 目的：封装日程信息，包括日程的基本属性及其初始值，并提供将日程对象转换为字符串的功能，以便存储到数据库。
3. 功能列表：无。

## 3.5 业务/实现流程说明

### **3.5.1添加日程处理流程**

用户复制通知后应用监听到剪贴板变化，触发通知栏提醒功能，用户可以下拉通知栏点击进入与智能助手对话的界面并将复制的通知粘贴到对话框中，智能助手使用自然语言处理技术智能解析其中的日程信息并将其整理成直观易读的格式回复给用户，同时将该日程存入数据库，其他组件同步接收数据库变化并显示到日程表中，从而达到增加新日程的目的。

用户也可以选择手动添加日程，点击进入添加日程界面然后依次输入日程的标题、日期、时间等信息，点击提交即可将日程存入数据库，同步将日程块显示在日程表中，相关流程如下图：

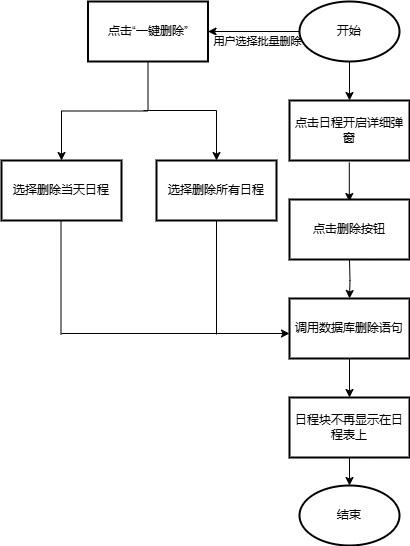


#### 图3-5 添加日程处理流程图

### **3.5.2删除日程处理流程**

当用户想要删除日程时，可以点击想要删除的日程进入弹窗然后单击右上角的删除按钮，即可单独将该日程从数据库中删除，同时也会不再显示在日程表上。

如果用户想要批量删除日程，可以点击主界面右上角的更多按钮，点击“一键删除”，然后便可选择删除当天日程或者删除所有日程，同样可以实现对指定范围内的日程的删除，相关流程如下图：



#### 图3-6 删除日程处理流程图

## 3.6接口描述

### **3.6.1调用接口**

导入了通知管理@ohos.notificationManager，用于发送和管理系统通知，主要使用了isNotificationEnabled()方法用于检查是否启用了通知权限，使用requestEnableNotification()方法用于请求用户启用通知，使用了publish()方法用于发布通知到系统通知栏。

导入了剪贴板库 @ohos.pasteboard，用于监听剪贴板内容的变化。主要使用了getSystemPasteboard()方法用于获取系统剪贴板实例，使用了on('update', listener)方法用于监听剪贴板内容的更新事件，并触发相关操作。

导入了关系型数据库管理库 @ohos.data.relationalStore，用于操作日程表中的数据主要使用了RdbStore.executeSql()方法来执行SQL语句，用于增、删、改、查日程表中的数据。

导入了 @ohos.events.emitter 库，主要使用了 emitter.emit() 方法来通知其他页面日程数据已更新，确保其他页面可以实时反映修改后的日程。

导入了 @ohos.net.http 模块，使用了 http.createHttp()方法创建 httpRequest 实例，调用 request() 方法发送 POST 请求，用于与远程服务进行数据交互，并根据用户输入获取 AI 智能助手的响应内容，使用了 on('headersReceive') 方法监听 HTTP 响应头接收事件，并使用 destroy() 进行请求销毁和资源释放。

导入了 @ohos.promptAction 模块，该模块提供了用于创建提示框和其他交互控件的功能，主要使用的是 CustomDialogController 类和相关方法，来控制自定义对话框的显示与消失，new CustomDialogController({ builder, cancel })：用于创建自定义对话框控制器。

### **3.6.2提供接口**

数据存储接口位于 GlobalContext 模块中，提供对关系型数据库 RdbStore 的访问，主要提供了获取存储实例的接口 GlobalContext.getInstance().getStore()，允许其他模块操作日程数据，包括增删改查。

## 3.7 UI设计

1. **该应用使用ArkTS语言进行开发，主要设计了：**

**·三个页面**：主页面（日程表界面）、日程添加页面和智能助手页面。

**·多个自定义组件**：日期选择器、时间选择器、日程块、日程列、日程信息弹窗。

1. **整个设计中使用的UI控件和布局：**

**UI控件方面：**

·按钮组件：在页面和自定义组件中设计了多种样式的按钮，包括取消按钮、确认按钮和图标按钮，以适应不同的操作需求和视觉层次，同时实现事件的触发或页面的跳转。

·弹窗组件：向用户提供清晰的操作指引。

·图片组件：使用户界面友好、直观。

**布局方面：**

·线性布局：在日程表界面中，使用线性布局来组织日程块，使其按照时间顺序排列。在其他页面和自定义组件中也使用了线性布局。

·弹性布局：主要在日程信息弹窗组件中利用弹性布局来创建灵活的页面结构，确保内容的对齐和分布。

·层叠布局：主要在日程列组件中使用层叠布局来管理层次关系，展示覆盖的内容。

# **4**其他

## 4.1 成员分工

在项目开发中，团队的每个成员都扮演着不可或缺的角色，共同协作，以确保应用开发的高效和成功。

1. **队长何沁遥**

·应用设计：分析应用功能，绘制用户界面，确保界面直观友好。

·任务分配：根据每个队员的专长和项目需求，合理分配任务，确保团队协作的高效性。

·功能实现：主要负责“智能助手”界面的编写，这是应用的创新点和用户交互的核心。同时解决了通知横幅、监听剪贴板和中文输入法的问题，进一步完善应用功能。

·内容完善：管理整个项目的架构，并对队员编写的其他界面内容进行补充和完善，提升应用的整体体验。

·PPT编写：负责应用介绍PPT的编写，确保团队的工作成果得到有效的记录和展示。

1. **队员1徐楚雯**

·功能实现：专注于“编辑日程”界面、一键删除功能和冲突检测功能的开发，确保用户能够轻松地创建、编辑或删除日程。

·弹窗设计：设计并实现日程块的信息弹窗，为用户提供删除、修改日程信息的功能，增强应用的交互性。

·文档编写：负责开发设计文档的编写，确保团队的工作成果得到有效的记录和展示。

1. **队员2张施琪**

·界面开发：负责日程块结构的设计以及“我的日程”主界面的编写，这是用户与应用交互的主要入口。

·通知设计：开发通知栏功能，确保用户能够及时接收日程提醒、通过点击通知栏进入应用粘贴通知信息。

·文档编写：负责开发设计文档的编写，确保团队的工作成果得到有效的记录和展示。

**4.团队协作**

·接力式编程：采用接力棒式的编写任务，当一个队员遇到难题或瓶颈时，其他队员会主动提供支持。

·知识共享：队员之间相互分享查询到的资料和解决方案，共同克服技术难题。

·代码审查：定期进行代码审查，确保代码质量，同时促进团队成员之间的学习和成长。

·创意碰撞：在开发过程中，团队成员不时提出创新想法，不断优化和改进应用功能和应用界面。

## 4.2困难与思考

**技术熟练度：**我们在使用ArkTS语言编写应用时遇到了挑战，在修复不同类型的bug上耗费了较长的时间。为了提高开发效率，我们需要加强ArkTS语言的学习和实践，以便更快地掌握语言特性和调试技巧。

**版本兼容性问题：**最开始项目是基于API9开发的，后来由于功能需求需要升级到API11，在从API 9升级到API 11的过程中遇到了难题，因为缺乏充足的资料来指导升级，所以该过程耗时较长。针对这一问题，我们需要多研究不同API版本之间的差异，并寻找或创建适当的迁移指南。

**团队协作与整合：**由于不同成员负责不同的功能模块，这在集成各个部分时带来了挑战，有时会导致UI界面的设计、效果实现或功能实现上的遗漏。为了解决这一问题，我们需要优化团队的工作流程，增强跨模块的沟通和协作，确保每个部分在整合时都能无缝对接，避免功能和设计的遗漏。

**优化用户体验：**我们实现了剪贴板的实时监听，当剪贴板内容改变时，有一个横幅式通知栏提醒。但频繁的无关提醒在一定程度上造成了资源的浪费，如何提高对剪贴板中日程相关文本的识别准确性，以确保只有相关日程的文本才会触发通知是一个难点。后续我们考虑开发一个文本分析模块，使用关键词匹配和模式识别技术来分析剪贴板中的文本。若识别到用户复制了一个有关日程的消息，则有一个横幅式通知栏提醒，若与日程安排无关则不会打扰用户。

## 4.3参考

**[OpenHarmony 4.1 Release版本的应用开发文档](https://docs.openharmony.cn/pages/v4.1/zh-cn/application-dev/quick-start/start-overview.md)**

**[OpenHarmony 应用开发FullSDK获取与替换](https://blog.csdn.net/Devlin_/article/details/139742551)**

**[@ohos.data.relationalStore（关系型数据库）指南](https://docs.openharmony.cn/pages/v4.1/zh-cn/application-dev/reference/apis-arkdata/js-apis-data-relationalStore.md)**

**[OpenHarmony开发记录-从API9转API10的痛苦之路](https://blog.csdn.net/m0_65179181/article/details/135824828?spm=1001.2101.3001.6661.1&utm_medium=distribute.pc_relevant_t0.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ECtr-1-135824828-blog-140772066.235%5Ev43%5Epc_blog_bottom_relevance_base3&depth_1-utm_source=distribute.pc_relevant_t0.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ECtr-1-135824828-blog-140772066.235%5Ev43%5Epc_blog_bottom_relevance_base3&utm_relevant_index=1)**

**[ArkTS构建布局](https://blog.csdn.net/g984160547/article/details/136735254?ops_request_misc=%257B%2522request%255Fid%2522%253A%25229F710694-8879-41DE-87A6-AB5F04FAAD39%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=9F710694-8879-41DE-87A6-AB5F04FAAD39&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-1-136735254-null-null.142%5ev100%5epc_search_result_base2&utm_term=arkts%20column%E4%B8%8Bimage%E6%80%8E%E4%B9%88%E5%B8%83%E5%B1%80&spm=1018.2226.3001.4187)**