基于Android的日程管理系统的设计与实现

摘要

在科技创新之手的推动下，社会发展的车轮滚滚前行，人们生活质量得到不断提高的同时，每天需要处理的事务也与日俱增，如何按时、高效地完成生活中接踵而来的各项任务已经称为摆在我们面前的一个巨大挑战。

针对生活节奏不断加快的大环境下用户日益增涨的日程管理需求，本文借助Android平台设计和实现了一款功能丰富的日程管理系统，旨在帮助用户在当下快节奏的社会环境中提高学习和工作的效率，增加生活幸福感。

“日程管理”这一行为的核心在于记录，但如果一个日程管理软件仅提供记事本功能是远远无法满足当下用户需求的。记录、查看、提醒这三大功能是用户对日程管理的基本诉求，因此一个合格的日程管理软件只有集记事本、日历、闹钟三者为一体才能满足人们日常生活中绝大多数的使用场景。以此为蓝本，本文所设计和实现的日程管理系统提供了日程收集箱、TODO日历、到点提醒等功能以满足用户的日常使用。而在此基础之上，本系统还提供了定时推送、数据反馈、专注计时等功能以帮助用户合理地规划、利用时间。

关键词：日程管理；Android；效率；TODO

**DESIGN AND IMPLEMENTATION OF SCHEDULE MANAGERMENT SYSTEM BASED ON ANDROID**

**ABSTRACT**

Driven by the hand of technological innovation, the wheel of social development is rolling forward. While the quality of people&#39;s life is constantly improving, the daily affairs that need to be dealt with are also increasing day by day. How to complete the tasks that follow in life on time and efficiently has been called a huge challenge before us.

Aiming at the increasing schedule management needs of users in the environment of the accelerating pace of life, this paper designs and implements a feature-rich schedule management system with the help of the Android platform, aiming to help users improve learning and learning in the current fast-paced social environment. Work efficiency and increase happiness in life.

The core of schedule management is recording, but if a schedule management software only provides a notepad function, it is far from meeting the needs of current users. The three functions of recording, viewing, and reminding are the basic demands of users for schedule management. Therefore, a qualified schedule management software can only meet the vast majority of usage scenarios in people&#39;s daily life only by integrating notepad, calendar, and alarm clock. Based on this, the schedule management system designed and implemented in this paper provides functions such as task collection box, TODO calendar, and arrival reminder to meet users&#39; daily use. On this basis, the system also provides functions such as regular push, data feedback, and focused timing to help users plan and utilize time reasonably.

**Key words:** Schedule management; Android; Efficiency; TODO

# 绪论

## 课题背景及意义

近年来，随着移动互联网技术的高速发展，智能手机的购买和使用率也逐年增长。xxx研究报告的最新数据显示，2021年全球智能手机的出货量为x部，同比增长xx。应用在智能手机的操作系统也百花齐放，包括Android、iOS、Windows Phone、BlackBerry OS、Symbian等等。凭借着高度开源的特性，Android系统的市场占有率遥遥领先于iOS等其他系统，多年来一直稳居智能手机操作系统市场的头把交椅。据xx最新数据显示，截至2021年年底，Android的市场份额已达到xxx，与之相比iOS占xx，其他操作系统仅占xx。

与此同时，随着社会的不断进步，人们的生活质量越来越高，生活节奏也越来越快。随着生活节奏的不断加快，人们每天需要处理的事务也更加繁杂，经常会面临当前的事情还没处理完，新的任务就已经来到的情况。面对接踵而来的任务，我们常常忙得焦头烂额也处理不完，其中一部分原因是任务太多以至于我们不知道应该先处理哪个才好，另一部分原因则是生活中往往会有太多的外来因素干扰我们，使我们无法专注于当前事项，导致学习、工作效率低下。因此即使社会和科技的发展使得人们的生活质量得到了很大的飞跃，生活幸福感却不见得有太多的提高。

由此观之，在当前快节奏的社会环境下，人们迫切地需要一些辅助手段或者工具来帮助自己统筹安排生活中的大小事项，合理规划时间、提高工作和学习的效率。因此，日程管理系统软件随之应运而生。日程管理系统诞生的目的是帮助用户更好地管理时间，在有限的时间内高质量地完成更多的事。换而言之，即帮助用户提高生活效率，增加生活幸福感。

## 国内外研究现状

随着移动互联网技术的发展和智能手机的普及，目前国内外手机应用市场上的日程管理软件如雨后春笋般冒出，包括365日历、滴答清单、WunderList、高效ToDo、Any.Do、GTasks和CalenGo等产品。这些产品前赴后继地推出，填补了市面上日程管理类软件的空白，也便利了人们的日常生活。

虽然目前市面上存在大量的日程管理系统，但是他们的质量却是良莠不齐的。其中一部分日程管理软件严格来说仅仅是一个增加了记事本功能的日历软件，只具有简单的日程管理功能，它们功能简单，使用场景单一，无法满足用户日常生活中的大多数需求。这类软件只能说是日历软件的扩展，远远谈不上日程管理系统，过于单一的功能也得不到用户的青睐，导致它们的下载量也很少。而另一部分比较受用户欢迎的日程管理软件功能显然更为丰富，它们不仅提供了最基础的日历和记事本功能，还提供了日程提醒甚至社交等功能，同时也具有良好的UI设计和交互设计，备受用户喜爱。但这类软件往往都有一个功能的弊端：高级功能需要收费甚至软件的下载都需要收费，普通用户仅能使用最基础的日程管理功能，这就导致软件虽然提供了丰富的功能但是大部分用户却无法使用的状况。同时市面上大部分的日程管理软件都没有提供用户数据的反馈功能，用户无法得知某段时间内自己的任务完成状况和工作效率，而这些数据往往是大部分用户都想要看到的。

## 论文构成

本文针对基于Android的日程管理系统的设计与实现展开研究，共分为七个章节，大体结构如下：

1. 绪论。该章节主要交代了本课题的研究背景及意义、国内外研究现状，并对论文的内容与结构进行了详细的说明。
2. 开发工具与关键技术介绍。该章节对设计和实现所研究系统的开发工具Android Studio，以及相关核心技术进行了介绍。
3. 系统需求分析。该章节对所研究系统进行了需求分析，阐述了系统的功能模块，并通过流程图介绍和展示了相关业务用例，确定了所研究系统的可行性。
4. 系统设计。该章节描述了系统的总体设计，划分了具体的功能模块并描述了相关逻辑。
5. 系统实现。该章节描述了系统各模块的具体功能及实现方式。
6. 系统测试。该章节主要针对所研究系统进行功能测试和性能测试，根据测试结果来分析系统的实用价值。
7. 总结与展望。该章节对全文的核心内容进行概括，总结了系统的实现程度，自己的收获与不足，并对整个过程进行反思。

# 开发工具与关键技术介绍

## 开发工具介绍

### Android Studio

Android Studio是Google公司推出的以IntelliJ IDEA为基础构建的Android应用集成开发环境，除了IntelliJ强大的代码编辑器和开发者工具之外，Android还提供了更多可提高Android应用构建效率的功能，包括：

1. 基于Gradle的灵活构建系统；
2. 快速且功能丰富的模拟器；
3. Apply Changes功能：可将代码和资源更改推送到正在运行的应用，而无需重启应用；
4. 通用代码模板以及GitHub集成；
5. 大量的测试以及捕获性能、可用性、版本兼容性的工具和框架
6. C++和NDK的支持
7. 强大的布局编辑器，可以通过拖拽UI控件布局并进行效果预览
8. 支持ProGuard和应用签名

图片包含 应用程序

描述已自动生成

图 2‑1 Android Studio

### Sourcetree

Sourcetree是Atlassian公司提供的一款可以在Windows和Mac上使用的免费的Git GUI。Soucetree提供了一个精美简洁的操作界面帮助开发者管理Git仓库，大大地简化了开发者和Git仓库之间的交互方式，使得开发者可以更专注于编码工作。



图 2‑2 Sourcetree

## 关键技术介绍

### Android

Android是一种基于Linux内核的自由且开源的操作系统，主要应用于智能手机、平板电脑、智能手表等移动设备。Android系统最初由Andy Rubin开发，2005年8月被Google公司收购注资。从2007年11月开始，Android由Google公司及开放手持设备联盟（Open Handset Alliance）共同领导及开发。

据Statcounter数据显示，截至2021年4月全球移动操作系统中，Google公司Android系统的市场占有率高达72.2%，Apple的iOS系统市场占有率为26.99%，其余操作系统的市场占有率之和低于1%。

图标

描述已自动生成

图 2‑3 Android系统

Google官方将Android平台分为五层架构，自上而下分别是：

1. 应用层(System Apps)：该层负责与用户交互，所有应用程序都属于该层；
2. 应用框架层(Java API Framework)：该层为Android开发人员提供了编写应用程序时所需的API，这些API都是由Java语言编写的；
3. 系统运行库层(Native & ART)：该层分为两部分：C/C++程序库和Android运行时库，为Android系统提供了主要的特性支持；
4. 硬件抽象层(HAL)：该层为位于操作系统和硬件电路之间的接口层，目的在于向更上层提供抽象化的硬件控制接口；
5. Linux内核层(Linux Kernel)：Android系统运行于Linux内核之上，其核心服务都是基于Linux内核的。



图 2‑4 Android平台架构

### RecyclerView

RecyclerView是Android 5.0版本中作为support-v7库中的一个组件出现，作为一个帮助开发者灵活地展示列表形式或者网格形式的数据（如文本或图片）的容器，它高效替代了Google最初推出的ListView等列表组件。

当列表滑动的时候，实际上只有少量邻近的视图会显示在屏幕上。当视图滑出屏幕时，RecyclerView 会复用它并且填充新的数据。由于它是通过回收已有的结构而不是持续创建新的列表项，所以它可以有效提高应用的时间效率和空间效率。

图片包含 游戏机, 建筑, 画, 门

描述已自动生成

图 2‑5 RecyclerView数据填充原理

RecyclerView兼顾了灵活性和个性化具有以下特点：

* RecyclerView实现了**视图代码解耦**，使用ViewHolder 模式，具备强大的Item**复用机制。**这样做可以提高性能，因为它无需频繁调用findViewById()方法即可访问表项的视图；
* RecyclerView使用[LayoutManager](https://link.zhihu.com/?target=https%3A//developer.android.google.cn/reference/androidx/recyclerview/widget/RecyclerView.LayoutManager" \t "_blank)，它支持纵向滑动的列表和横向滑动的列表，以及交错布局的列表和网格布局的列表；
* RecyclerView 提供默认的表项动画以及自定义动画的入口。

### Service

Service作为Android四大组件之一，是Android中实现程序后台运行的解决方案，它非常适合于去执行哪些不需要和用户交互并且还要求长期的任务。Service的运行不依赖于任何用户界面，即使应用程序被切换到了后台，或者用户打开了另一个应用程序，Service仍然能保持正常运行。不过需要注意的是，Service并不是运行在一个独立的进程当中，而是依赖于创建服务时所在的应用程序进程。当某个应用程序进程被杀掉时，所依赖于该进程的Service也会停止运行。

通过IPC(Inter-Process Communication)机制，Android的组件可以绑定到一个Service并与之交互，例如Service可以处理网络操作、播放音乐、操作文件I/O或者与Content Provider交互，并且这些活动都是在后台进行的。

### SQLite

SQLite是一款由D.RichardHipp在2000年发布的开源的轻量级嵌入式关系数据库，遵循ACID，运算速度快、占用资源少，非常适合在移动设备上使用。Android将SQLite嵌入到系统中，使得本地持久化功能有了质的飞跃。除了SQLiteOpenHelper等Android官方提供的帮助类，还存在LitePal、GreenDao等第三方数据库操作框架帮助用户使用SQLite实现数据的持久化。



图 2‑6 SQLite

# 系统需求分析

## 系统功能需求

该系统作为一个基于Android平台的日程管理系统，为用户提供的是日常管理方面的服务，包括以下功能：添加日程、删除日程、修改日程、查看日程、日程提醒、数据反馈、番茄计时等功能。其中，以日程的增删改查为系统的核心功能。

### 添加日程

用户在主界面可以根据日历所选择的日期，通过底部的输入框添加当前日期下待完成的日程。在此过程中用户仅需输入日程标题即可，如果用户需要在指定时间点（某小时）完成日程，同样可以通过底部输入框中的控件完成待做日程精确时间的设置。

### 删除日程

如果用户需要删除某个日程，可以在日程列表界面长按该日程项，通过弹出的删除对话框实现该需求。同时用户也可以在用户详情界面通过删除按钮实现具体日程的删除。

### 修改日程

当日程信息（包括标题、内容、时间）发生改变时，用户可以通过点击日程列表页中待修改日程项进入到日程详情页，然后在日程详情页对日程信息进行修改，修改后的日程数据自动保存到本地数据库中。

### 查看日程

本系统为用户提供了多种不同的视图以查看日程信息。用户可以在主页面通过切换日期来查看不同日期下的待做事件列表；用户也可以在收集箱页面查看全部未完成和已完成的日程；用户还可以通过点击日程列表中具体的日程项进入日程详情页面，查看该日程的完成情况、时间、标题、内容等详细信息。

### 日程提醒

本系统可以根据用户在日程项中设置的具体时间（分钟），在指定时间到达时系统会通过通知栏对用户进行提醒；同时系统也会在每天八点在通知栏中将当天待做事项全部推送给用户。用户可以在添加日程时就设置提醒时间，也可以在添加日程时不设置具体时间，后续通过日程详情页进行设置。

### 数据反馈

系统会根据后台收集到的用户数据，将用户本周以及本月的日程完成情况、专注时长等数据以图表的形式在特定的界面反馈给用户，以供用户参考。

### 番茄计时

系统根据番茄学习法提供了番茄钟以帮助用户专注于当前事项，提高日程完成的效率。同时后台也会记录相关数据，并以图表的形式实时反馈给用户。

## 业务用例分析

根据对功能需求的分析可以得出系统的业务用例如下图：

图示

描述已自动生成

图 3‑1 系统业务用例图

# 系统设计

## 业务实现方案设计

### 添加日程

图形用户界面

描述已自动生成

图 4‑1 添加日程业务序列图

点击MainActivity底部的输入框后会弹出定制的Dialog以供用户输入待添加的日程信息。输入完毕后，点击确定按钮，系统会先检查用户的输入是否合法，如果输入合法则会将日程数据保存到SQLite中，然后刷新视图，将用户新添加的日程在正确的位置展示。

### 删除日程

图示

描述已自动生成

图 4‑2 删除日程业务序列图

通过长按目标Item项可以实现日程的删除。当系统监听到列表中某个日程Item被长按时，会弹出删除Dialog，用户点击确定后，系统会先删除Adapetr中缓存的目标日程数据，然后再删除数据库中的目标日程数据。之后系统会根据返回结果刷新视图，展示最新的数据。

### 修改日程

图形用户界面, 图示

中度可信度描述已自动生成

图 4‑3 修改日程业务序列图

用户如果需要修改现有日程的信息，可以通过点击日程列表中目标日程Item的方式进入日程详情页面，该页面的职责是展示和修改日程的具体信息，包括标题、内容、时间、完成状态。在日程详情页修改日程信息后，会同步更新到SQLite数据库中。

### 查看日程

图形用户界面

描述已自动生成

系统启动时，主动从数据库中读取用户的日程数据，然后通过RecyclerView控件将日程数据已列表的形式展示。在初始化RecyclerView时，需要通过Adapter完成数据和ItemView的适配，然后才能正常地展示列表数据。

### 日程提醒

图示

低可信度描述已自动生成

图 4‑4 日程提醒业务序列图

用户可以通过设置日程的具体时间，让系统在指定的时间点以通知的形式提醒自己完成任务。该功能主要通过Android四大组件之一的Service实现：用户可以在添加/修改日程时设置日程的具体时间，一旦系统监听到某个日程被设置了具体的完成时间，则会在后台的Service中通过AlarmManager设置一个目标时间点的闹钟，在指定时间唤醒手机系统发出通知提醒用户完成日程。因为Servcie的特殊性质，即使应用程序不在前台运行用户仍然能得到日程提醒。

### 数据反馈

图形用户界面

描述已自动生成

图 4‑5 数据反馈业务序列图

在数据统计页面，系统会将用户最近一周、最近一月的日程完成情况、专注时长等数据以折线图、饼状图等形式反馈给用户。该功能借助于第三方框架HelloCharts实现：ChartsActivity启动时，系统会从SQLite数据库中读取相关数据，然后对数据进行加工处理以适配HelloCharts提供的接口，最后通过HelloCharts将相关数据以图表的形式展示给用户。

### 番茄计时

图片包含 图示

描述已自动生成

图 4‑6 番茄计时业务序列图

在番茄计时页面中，用户可以通过上下滑动手指来设置计时时长，然后点击开始计时即可。番茄计时的目的是根据番茄学习法帮助用户提高对当前事项的专注度，以提高工作效率。该功能的核心是通过自定义View的方式实现的，通过自定义View将视图绘制和Android提供的计时器相结合，从而达到边计时边展示进度的目的。

## 数据库设计

根据基于Android的日程管理系统的需求分析，我们设计并实现了Schedule、ConcentrationData两种表用以承载系统功能。

### 日程信息表

日程信息表（Schedule）是系统的核心表，用于记录用户的日程信息，包括日程的标题、描述、完成状态、计划完成时间、日期等数据项。

**表 4‑1 日程信息表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 说明 | 类型 | 是否为空 | 是否为主键 |
| id | 唯一标识符 | int | 否 | 是 |
| title | 日常标题 | text | 否 | 否 |
| des | 日程描述 | text | 否 | 否 |
| is\_finish | 完成状态 | int | 否 | 否 |
| time | 日程目标完成时间 | int | 否 | 否 |
| year | 年 | int | 否 | 否 |
| month | 月 | int | 否 | 否 |
| day | 日 | int | 否 | 否 |

### 专注时长表

专注时长表（ConcentrationData）用于记录用户的专注数据，包括专注时长、日期等数据，其中专注时长是当前所记录日期的总专注时长。通过该表记录用户的专注数据，用于实现数据反馈功能。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 说明 | 类型 | 是否为空 | 是否为主键 |
| id | 唯一标识符 | int | 否 | 是 |
| duration | 专注时长 | int | 否 | 否 |
| year | 年 | int | 否 | 否 |
| month | 月 | int | 否 | 否 |
| day | 日 | int | 否 | 否 |

# 系统实现

## 主界面实现

系统主界面分为三部分实现：顶部导航栏、中间主体FrameLayout、底部输入框控件。顶部导航栏采用Toolbar实现，左侧是一个菜单按钮，用于弹出DrawLayout的菜单，用户可以通过此菜单切换界面使用不同的功能，中间位置是两个TextView，用以展示用户当前选中日期数据；中间主体使用FrameLayout承载ScheduleFrag和EventSetFrag的Fragment，其中ScheduleFrag为主界面的展示控件，用以展示日历视角下的日程列表，通过点击日历上不同的日期，可以实现日期的切换，系统会随之展示不同日期下的TODO List（即日程列表），日程列表的每个日程项和数据库中的日程信息数据一一对应，向用户展示了日程的标题、时间、完成情况，用户还可以通过点击单个日程项查看或修改日程的详细信息、长按单个日程项实现目标日程的删除；底部的输入框帮助用户添加新的日程，该控件由ImageButton + EditTextView + ImageButton的组合方式实现。

图形用户界面, 应用程序

描述已自动生成 图形用户界面, 应用程序

描述已自动生成

图 5‑1 系统主界面 图 5‑2 系统菜单栏

图形用户界面, 应用程序

描述已自动生成文本

描述已自动生成

图 5‑3 主界面布局树1 图 5‑4 主界面布局树2

主界面通过initView()方法用于初始化页面中的各个控件：

private void initView() {

mDrawer = findViewById(R.id.dlMain);

btnMenu = findViewById(R.id.btnMenu);

……

btnMenu.setOnClickListener(this);

tvMainTitle.setOnClickListener(this);

……

= getResources().getStringArray(R.array.calendar\_month); //获取月份字符串数组

llTitleDate.setVisibility(View.VISIBLE); //初始化标题栏的日期数据

}

日历控件中日期被选择后通过setCurrentSelectDate()方法进行回调：

private void setCurrentSelectDate(int year, int month, int day) {

mCurrentSelectYear = year;

mCurrentSelectMonth = month;

mCurrentSelectDay = day;

}

通过gotoScheduleFragment()和gotoEventSetFragment()方法实现视图的切换：

private void gotoScheduleFragment() {

FragmentTransaction ft = getSupportFragmentManager().beginTransaction();

ft.setTransition(FragmentTransaction.TRANSIT\_NONE);

if (mScheduleFragment == null) {

mScheduleFragment = ScheduleFragment.getInstance();

ft.add(R.id.flMainContainer, mScheduleFragment);

}

if (mEventSetFragment != null) {

ft.hide(mEventSetFragment);

}

ft.show(mScheduleFragment);

ft.commit();

llTitleDate.setVisibility(View.VISIBLE);

tvMainTitle.setVisibility(View.GONE);

mDrawer.closeDrawer(Gravity.START);

}

## 日程管理实现

### 日程展示

日程管理模块包含日程的增删改查四大功能，具体实现依赖于ScheduleFragment、EventSetFragment两大碎片（碎片是Android的一个视图组件），这两大碎片的功能相似，区别在于UI布局：ScheduleFragment是以日历视图来展示用户的日程数据，可以通过点击日历上的日期来切换展示不同日期下的日程安排；EventSetFragment则是日程收集箱，不区分日期，以日程的完成情况为划分依据，将用户所有的日程数据以列表的形式展示。

图形用户界面, 应用程序

描述已自动生成 图形用户界面, 应用程序

描述已自动生成

图 5‑5 ScheduleFrag实现效果 图 5‑6 EventSetFrag实现效果

无论是ScheduleFragment还是EventSetFragment，在展示日程数据时都是依赖于Android的RecyclerView控件去实现的。

RecyclerView的初始化代码如下：

private void initScheduleList() {

rvScheduleList = slSchedule.getSchedulerRecyclerView();

LinearLayoutManager manager = new LinearLayoutManager(mActivity);

manager.setOrientation(LinearLayoutManager.VERTICAL);

rvScheduleList.setLayoutManager(manager);

DefaultItemAnimator itemAnimator = new DefaultItemAnimator();

itemAnimator.setSupportsChangeAnimations(false);

rvScheduleList.setItemAnimator(itemAnimator);

mScheduleAdapter = new ScheduleAdapter(mActivity, this, scheduleList, rvScheduleList);

rvScheduleList.setAdapter(mScheduleAdapter);

}

### 日程添加

主页底部的输入栏用于帮助用户添加新的日程。用户在中间输入框填写日程的标题后，然后点击右侧添加按钮即可。如果用户在添加日程的同时想设置日程的日期时间，在点击右侧添加按钮之前先通过点击左侧按钮弹出的对话框设置日期和时间即可。

图形用户界面, 应用程序

描述已自动生成 电脑萤幕

中度可信度描述已自动生成

图 5‑7 添加日程 图 5‑8 设置日程时间

设置日程的日期和时间时所适使用的对话框并没有使用Android提供的原生控件，而是使用xml文件自定义实现，其布局树如下图所示：

文本

描述已自动生成

图 5‑9 日期选择Dialog布局树

添加日程的核心代码如下：

public void insertSchedule() {

String content = etInputContent.getText().toString();

if (TextUtils.isEmpty(content)) {

ToastUtils.showShortToast(mActivity, R.string.schedule\_input\_content\_is\_no\_null);

} else {

closeSoftInput();

etInputContent.getText().clear();

Calendar calendar = Calendar.getInstance();

calendar.set(mCurrentSelectYear, mCurrentSelectMonth, mCurrentSelectDay);

Schedule schedule = new Schedule();

schedule.setTitle(content);

schedule.setDesc("");

schedule.setYear(mCurrentSelectYear);

schedule.setMonth(mCurrentSelectMonth);

schedule.setDay(mCurrentSelectDay);

schedule.setFinish(false);

schedule.setTime(mTime);

mTime = 0;

if(mScheduleAdapter != null) {

mScheduleAdapter.insertItem(schedule);

}

}

}

### 日程删除

当系统监听到ScheduleFrag或者EventSetFrag所展示的日程列表中的某个日程项被长按时，会自动弹出一个删除对话框，帮助用户实现对某个目标日程的删除。

图形用户界面, 应用程序, Teams

描述已自动生成

图 5‑10 日程的删除

长按Item所弹出的删除对话框也没有使用Android提供的Dialog，而是通过xml定制实现，其布局树如下：

图形用户界面, 文本

描述已自动生成

图 5‑11 删除对话框的布局树

删除功能核心代码如下：

public void removeItem(Schedule schedule) {

final int position = mSchedules.indexOf(schedule);

if(position != -1) {

while(!mRv.isComputingLayout() && mRv.getScrollState() == RecyclerView.SCROLL\_STATE\_IDLE) {

mCheckState.delete(position);

mSchedules.remove(position);

ScheduleDao.getInstance().deleteSchedule(schedule.getId());

notifyItemRemoved(position);

notifyDataSetChanged();

mFragment.resetVisibilityOfNoTaskView();

break;

}

}

}

### 日程修改

如果用户需要对某个日程数据进行修改，可以通过单击日程列表中的目标日程项跳转到日程详情页进行修改操作。日程详情页会展示日程的标题、内容、计划完成时间、完成状态等数据，用户在此页面修改信息后数据会同步更新到数据库中。

图片包含 背景图案

描述已自动生成

图 5‑12 日程详情页

该页面由标题栏和内容栏组成，标题栏左侧包含返回按钮，内容栏中通过CheckBox展示日程的完成状态，通过TextView展示日程计划完成时间，通过EditTextView展示日程的标题和内容，其布局树如下：

文本

描述已自动生成

图 5‑13 日程详情页布局树

日程修改的核心代码如下：

etTitle.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {

}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {

}

@Override

public void afterTextChanged(Editable s) {

//Log.d(TAG, "afterTextChanged: Title=" + etTitle.getText().toString());

mSchedule.setTitle(etTitle.getText().toString());

ScheduleDao.getInstance().updateSchedule(mSchedule);

}

});

## 日程提醒实现

为帮助用户更好地完成日程任务，系统还提供了两种日程提醒方式以及时告知用户待做事项。第一种方式是“到点提醒”：用户为日程设置了计划完成时间后，系统会根据用户所设置的时间在后台创建一个闹钟任务，当到达指定时间后，闹钟任务会调用Android系统组件在通知栏发起一个日程通知，提醒用户及时换成任务，用户还可以通过点击通知栏中的通知跳转到日程详情页中查看日程的详细信息；第二种提醒方式是“定时推送”：系统每天八点会将当天用户全部日程任务罗列在通知栏中，提醒用户今天有哪些任务需要完成，通过点击通知栏中的提醒通知，用户可以跳转到系统主页查看当前日期下的日程任务。

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成 图形用户界面

描述已自动生成

图 5‑14 到点提醒效果图 图 5‑15 定时推送效果图

日程管理模块的核心实现在于Service组件的使用，Service组件是Android四大组件之一，它运行在系统的后台，即使应用程序从前台退出，Service也仍然能正常运行。我们在Service中通过AlarmManager为每个日程都设置了定时任务，到达指定时间后，AlarmManger设置的闹钟被执行，紧接着会通知Service调用NotificationManager发起系统通知提醒用户日程信息。

日程提醒模块的核心代码如下：

public class AlarmBinder extends Binder {

public void addAlarm(long id, long time) {

Log.d(TAG, "addAlarm: ");

Intent intent = new Intent();

intent.setAction(MainActivity.ACTION\_ALARM\_NOTIFICATION);

intent.putExtra(ScheduleDetailActivity.SCHEDULE\_ID, id);

PendingIntent pi = PendingIntent.getBroadcast(AlarmService.this, (int) id, intent, 0);

am.set(AlarmManager.RTC\_WAKEUP, time, pi);

}

public void cancelAlarm(long id) {

Log.d(TAG, "cancelAlarm: ");

Intent intent = new Intent();

intent.setAction(MainActivity.ACTION\_ALARM\_NOTIFICATION);

intent.putExtra(ScheduleDetailActivity.SCHEDULE\_ID, id);

PendingIntent pi = PendingIntent.getBroadcast(AlarmService.this, (int) id, intent, 0);

am.cancel(pi);

}

}

## 番茄钟实现

为帮助用户提高工作效率，系统根据番茄学习法为用户提供了番茄钟功能。在开始某项日程之前，用户可以根据个人情况设置一个倒计时，在倒计时结束之前专注于当前任务，倒计时结束之后休息一会儿再继续任务。系统提供的番茄钟的操作也很简单，通过在屏幕内上下滑动手指可以来实现番茄计时时长的调整，时长设置完毕之后点击开始按钮即可开始计时。番茄计时的数据会以日期为划分依据，同步跟新到数据库中，用户可以在数据反馈页面查看自己的专注情况。

图示

描述已自动生成

图 5‑16 番茄钟实现效果

番茄钟的实现主要通过自定义View的形式实现，核心在于TomatoView的实现。番茄计时页面的布局树如下：

图形用户界面, 文本, 应用程序

描述已自动生成

图 5‑17 番茄钟页面布局树

番茄钟的主体由外圈的一个随着时间不断消失的黑色圆环以及内圈一个数值不断减小的时间文本组成，无论是视图动画还是滑动调整时长功能都是通过自定义View的方式实现的，其核心代码如下：

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

mRectF.set(centerX - radius, centerY - radius, centerX + radius, centerY + radius);

mPaint.setAntiAlias(true); //为Paint添加抗锯齿属性

//黑圆

canvas.save();

mPaint.setColor(Color.BLACK);

mPaint.setStyle(Paint.Style.STROKE);

mPaint.setStrokeWidth(dpToPixel(5));

canvas.drawCircle(centerX, centerY, radius, mPaint);

canvas.restore();

//灰圆

canvas.save();

mPaint.setColor(mColor);

canvas.drawArc(mRectF, START\_ANGLE, 360 \* sweepVelocity, false, mPaint);

canvas.restore();

//时间

canvas.save();

timePaint.setColor(Color.BLACK);

timePaint.setStyle(Paint.Style.FILL);

timePaint.setTextSize(dpToPixel(40));

canvas.drawText(textTime, centerX - timePaint.measureText(textTime) / 2,

centerY - (timePaint.ascent() + timePaint.descent()) / 2, timePaint);

canvas.restore();

}

## 数据反馈实现

为帮助用户更好地查看和调整自身状态，系统将收集到的数据从SQLite数据库中取出，并借助第三方框架HelloCharts将取出的数据以图表的形式在应用程序界面中向用户展示，所展示的数据包括本周的日程完成情况、专注时长数据，以及本月的日程完成情况、专注时长数据。其中系统借助TextView将各项具体数据以本文形式展示，将日程完成情况以饼状图的形式展示，将用户专注时长以折线图的形式展示，并通过按钮切换本周、本月的数据。

图形用户界面, 应用程序

描述已自动生成

图 5‑18 数据反馈页实现效果

数据反馈页布局树如下：

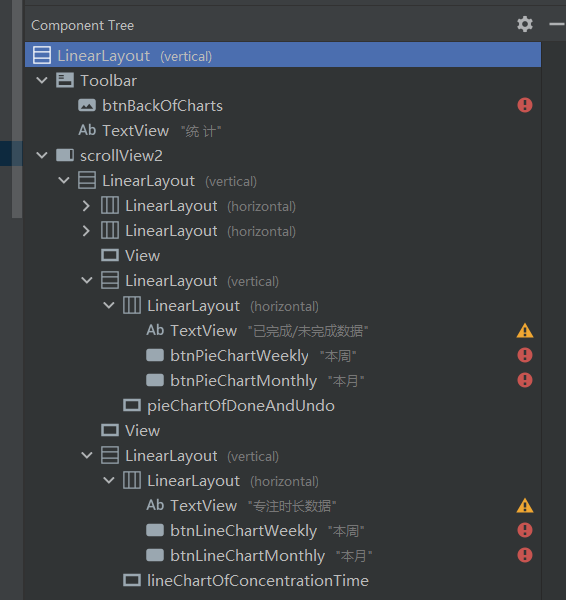


图 5‑19 数据反馈页布局树

其核心代码如下：

private void initView() {

btnBack = findViewById(R.id.btnBackOfCharts);

……

pieChart = findViewById(R.id.pieChartOfDoneAndUndo);

btnShowWeekDataOfPieChart = findViewById(R.id.btnPieChartWeekly);

btnShowMonthDataOfPieChart = findViewById(R.id.btnPieChartMonthly);

lineChart = findViewById(R.id.lineChartOfConcentrationTime);

btnShowWeekDataOfLineChart = findViewById(R.id.btnLineChartWeekly);

btnShowMonthDataOfLineChart = findViewById(R.id.btnLineChartMonthly);

btnBack.setOnClickListener(this);

btnShowWeekDataOfPieChart.setOnClickListener(this);

btnShowMonthDataOfPieChart.setOnClickListener(this);

btnShowWeekDataOfLineChart.setOnClickListener(this);

btnShowMonthDataOfLineChart.setOnClickListener(this);

}

# 系统测试

# 总结与展望

参考文献

致谢