

腾讯客户端菁英班

产品报告

学 校：	华中科技大学
姓 名：	许佳俊
运行环境：	Android Studio
产品名称：	记帐小助手

2024 年 12 月 24 日

目录

1 产品功能介绍	2
一、 核心功能	2
2 程序概要设计	2
一、 整体系统架构	2
3 实现原理與效果展示	4
1、应用启动与数据库初始化	5
2、用户界面与交互	7
3、数据录入与展示	7
4、数据查询与更新	10
5、事件处理	13
6、数据适配与展示	15
7、应用配置与资源管理	16
8、应用退出与资源释放	16
4 待开发功能	17
1、数据分析与图表可视化	17
2、预警提醒	17
3、数据备份与恢复	17
4、多账户管理	17
5、跨平台同步	17
6、智能建议	17

1 产品功能介绍

一、核心功能

“记帐小助手”允许用户快速记录每日的支出与收入，包括金额、类别（如餐饮、交通、储蓄等）以及备注信息，便于用户随时随地掌握个人或家庭的财务状况，同时还会记录每笔收支的具体日期，支持自动记录当前日期或手动选择日期，让用户能够按时间线查询和分析历史收支数据。该应用采用数据库管理系统，确保所有收支记录安全、高效地存储于云端或本地数据库中，实现数据的持久化保存与同步更新。“记帐小助手”还提供每日的收支总额计算，让用户一目了然地了解各时间段的财务状况，包括总支出、总收入及净收入（收入-支出）。

2 程序概要设计

一、整体系统架构

1. 数据库层 (Database Layer)

DBManager: 负责管理数据库操作，包括初始化数据库、增删改查等。

DBOpenHelper: 继承自 `SQLiteOpenHelper`，负责创建和升级数据库。

AccountBean: 数据类，表示一条记账记录。

TypeBean: 数据类，表示一个类型记录。

2. 适配器层 (Adapter Layer)

AccountAdapter: 为记账列表提供数据适配，将 `AccountBean` 列表数据绑定到视图。

TypeBaseAdapter: 为类型列表提供数据适配，将 `TypeBean` 列表数据绑定到视图。

3. 视图层 (View Layer)

包含 XML 布局文件，定义了用户界面的布局结构。

activity_edit_or_delete_item: 编辑或删除记账项的布局。

main_layout: 主界面布局，包含今日收支情况的 ListView。

activity_record: 添加新记账项的布局，包含输入金额、类型选择等。

4. 控制层 (Controller Layer)

EditOrDeleteItem: 处理编辑或删除记账项的 Activity。

MainActivity: 主界面 Activity，处理用户查看今日收支情况和添加新记账项的逻辑。

RecordActivity: 添加新记账项的 Activity，处理用户输入和保存新记账项的逻辑。

5. 应用层 (Application Layer)

UniteApp: 应用的全局 Application 类，负责初始化数据库。

6. 交互视图

用户通过视图层的界面与应用交互。

控制层接收用户输入，调用适配器层的方法来更新视图。

控制层也调用数据库层的方法来持久化数据。

7. 数据流

用户操作（如添加、删除记录）通过控制层传递给数据库层。

数据库层通过 DBManager 和 DBOpenHelper 执行实际的数据库操作。

数据查询结果通过数据库层返回给控制层，控制层再通过适配器层更新视图。

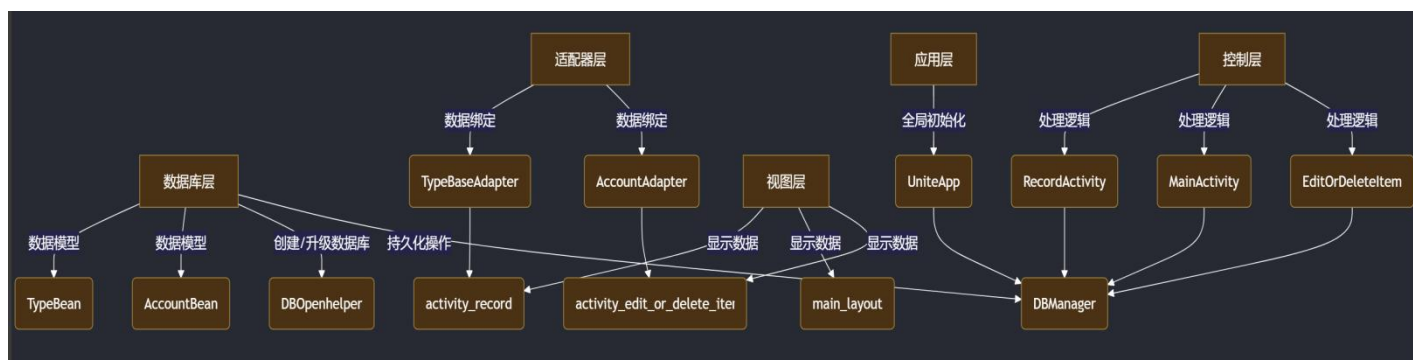


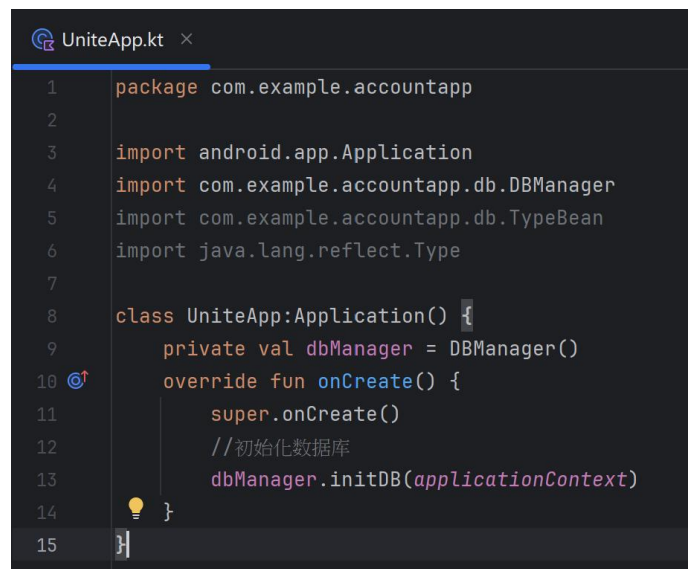
图 2.1 软件架构图

3 实现原理与效果展示

该记账应用的实现原理基于经典的客户端-服务器架构，其中客户端是用户直接交互的移动应用，而服务器则是本地数据库。以下是该应用实现原理的简要概述：

1. 应用启动与数据库初始化

当应用启动时，`UniteApp`（继承自`Application`）的`onCreate`方法会被调用，它负责初始化全局的数据库实例`DBManager`。

The image shows a code editor window titled 'UniteApp.kt'. The code is in Kotlin and defines a class 'UniteApp' that inherits from 'Application'. It includes imports for 'android.app.Application', 'com.example.accountapp.db.DBManager', 'com.example.accountapp.db.TypeBean', and 'java.lang.reflect.Type'. The 'onCreate' method is overridden, calling 'super.onCreate()' and then 'dbManager.initDB(applicationContext)' with a comment '//初始化数据库' (Initialize database).

```
1 package com.example.accountapp
2
3 import android.app.Application
4 import com.example.accountapp.db.DBManager
5 import com.example.accountapp.db.TypeBean
6 import java.lang.reflect.Type
7
8 class UniteApp:Application() {
9     private val dbManager = DBManager()
10    override fun onCreate() {
11        super.onCreate()
12        //初始化数据库
13        dbManager.initDB(applicationContext)
14    }
15 }
```

图 3-1 UniteApp 中的 onCreate()方法调用

`DBManager`通过`DBOpenHelper`内部类来创建或打开数据库文件，以及在数据库结构变化时进行版本管理。

```

10  *负责管理数据库的类
11  * 主要对于表当中的内容进行操作：增删改查
12  */
13  class DBManager {
14      private lateinit var db: SQLiteDatabase
15      val dbName = "AccountApp.db"
16      val dbVersion = 1
17      fun initDB(context: Context) {
18          val helper = DBHelper(context, dbName, dbVersion) //得到帮助类对象
19          db = helper.writableDatabase //得到数据库对象
20      }
21      /*
22      * 获取数据库当中的数据，写入内存集合
23      * 传入kind 表示输入或输出
24      */
25      fun getTypeList(kind: Int): List<TypeBean> {
26          val list = mutableListOf<TypeBean>() //创建一个可变的列表来存储结果
27          val sql = "select * from typetb where kind = $kind"
28          val cursor = db.rawQuery(sql, selectionArgs: null)
29
30          val typeNameIndex = cursor.getColumnIndex("typename")
31          val imageIdIndex = cursor.getColumnIndex("imageId")
32          val simageIdIndex = cursor.getColumnIndex("simageId")

```

图 3-2 `DBOpenHelper` 内部类的调用

2. 用户界面与交互

`MainActivity` 是应用的主界面，显示今日的收支情况，分别对今日的支出和收入记录进行求和并展示在页面上方。它使用 `ListView` 来展示数据，其中每一项数据都由 `AccountAdapter` 提供。

```

31  class MainActivity : Activity() {
32
33      override fun onCreate(savedInstanceState: Bundle?) {
34          super.onCreate(savedInstanceState)
35          setContentView(R.layout.main_layout)
36
37          initTime()
38          todayLv = findViewById(R.id.main_lv)
39          setLVLongClickListener()
40
41          //添加 ListView 布局
42          addLVHeaderView()
43          mData = mutableListOf()
44          //设置适配器：加载每一行数据到列表当中
45          adapter = AccountAdapter(context, this, mData)
46          todayLv.adapter = adapter
47
48          val btnedit: Button = findViewById(R.id.main_btn_edit)
49          val clicklistener = View.OnClickListener { view ->
50              handleClick(view)
51          }
52          btnedit.setOnClickListener(clicklistener)
53
54      }
55
56      private fun addLVHeaderView() {
57          //将布局转换成View对象
58          headerView = getLayoutInflater().inflate(R.layout.item_mainlv_top, root: null);
59          todayLv.addHeaderView(headerView);
60          //保持主界面可滚动性

```

图 3-3 `ListView` 和 `AccountAdapter` 调用以展示数据



图 3-4 记录小助手首页

用户可以通过点击按钮进入`RecordActivity`来添加新的记账项，或者在`MainActivity`中长按列表项来编辑或删除已有的记账项，这些功能将在接下来的功能介绍中逐一展示。

```

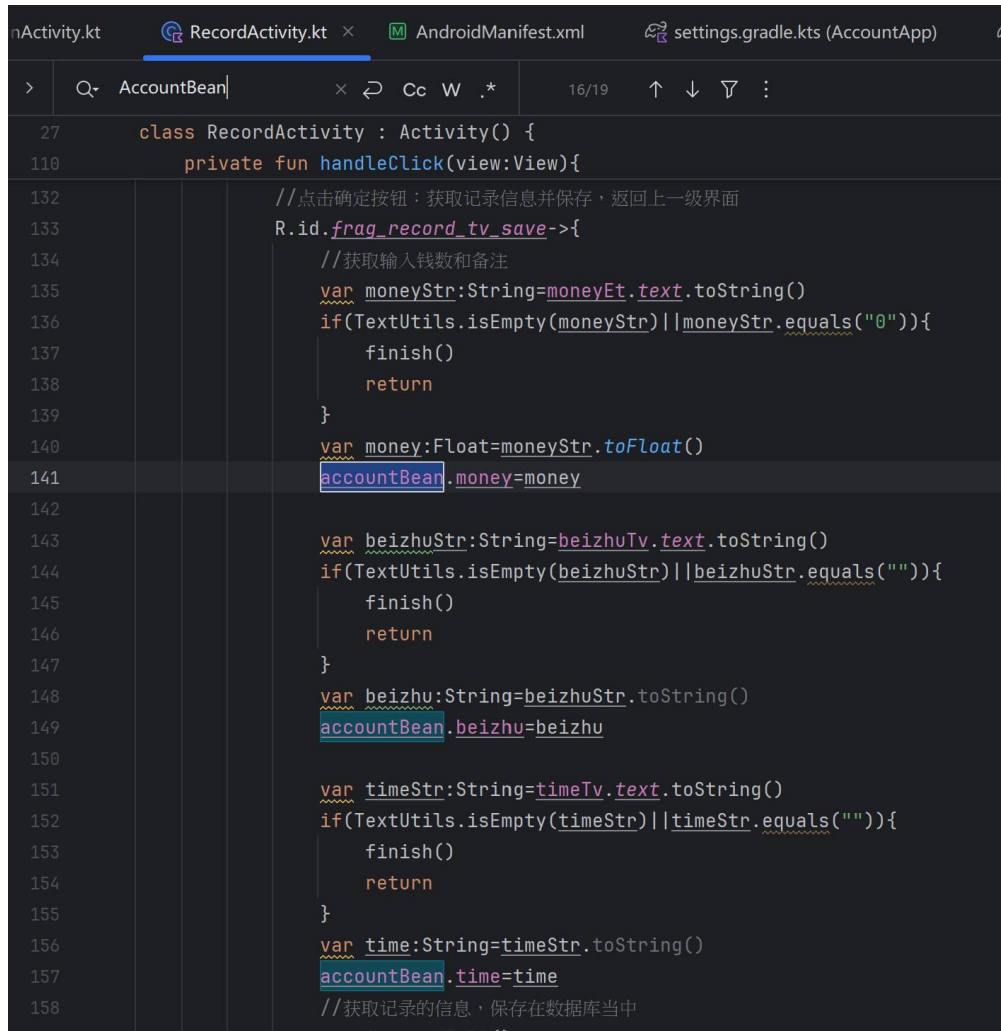
MainActivity.kt x AndroidManifest.xml settings.gradle.kts (AccountApp) build.gradle
> Q- ListView 4/5 ↑ ↓ 🔍 :
31 class MainActivity : Activity() {
66     private fun addLVHeaderView() {
78     }
79
80     /*
81     *设置ListView的长按事件
82     */
83     private fun setLVLongClickListener() {
84         todayLv.setOnItemClickListener() { parent, view, position, id ->
85             if (position == 0) {
86                 false
87             } else {
88                 val pos = position - 1
89                 val clickBean = mDataas[pos] // 获取正在被点击的这条信息
90                 // 弹出提示用户选择删除还是修改的对话框
91                 GotoEditOrDelete(clickBean)
92                 false
93             }
94         }
95     }
}

```

图 3-5 长按列表项实现

3. 数据录入与展示

在`RecordActivity`中，用户输入金额、选择类型、添加备注，并选择是收入还是支出。这些信息被封装进`AccountBean`对象中。



```
27 class RecordActivity : Activity() {
110     private fun handleClick(view: View) {
132         // 点击确定按钮：获取记录信息并保存，返回上一级界面
133         R.id.frag_record_tv_save->{
134             // 获取输入钱数和备注
135             var moneyStr: String = moneyEt.text.toString()
136             if (TextUtils.isEmpty(moneyStr) || moneyStr.equals("0")) {
137                 finish()
138                 return
139             }
140             var money: Float = moneyStr.toFloat()
141             accountBean.money = money
142
143             var beizhuStr: String = beizhuTv.text.toString()
144             if (TextUtils.isEmpty(beizhuStr) || beizhuStr.equals("")) {
145                 finish()
146                 return
147             }
148             var beizhu: String = beizhuStr.toString()
149             accountBean.beizhu = beizhu
150
151             var timeStr: String = timeTv.text.toString()
152             if (TextUtils.isEmpty(timeStr) || timeStr.equals("")) {
153                 finish()
154                 return
155             }
156             var time: String = timeStr.toString()
157             accountBean.time = time
158             // 获取记录的信息，保存在数据库当中
```

图 3-6 封装进`AccountBean`对象实现

在首页中点击右下角的”记一笔”按键，即可跳转到记帐录入界面。



图 3-3 点击右下角”记一笔”按键



图 3-4 记帐录入界面 1

第一步可选择所要录入的资金类型（收入 / 支出，餐饮、交通、购物、服饰、日用品等等）。值得一提的是系统调用了 Google 键盘方便用户输入信息，用户可输入相关金额，按照需要添加备注，以及更改收支日期（默认为手机系统中当前时间）。

例如此时设定记帐类型为”支出”的”交通”，金额为”5”，时间为默认，效果如图 3-5 所示。



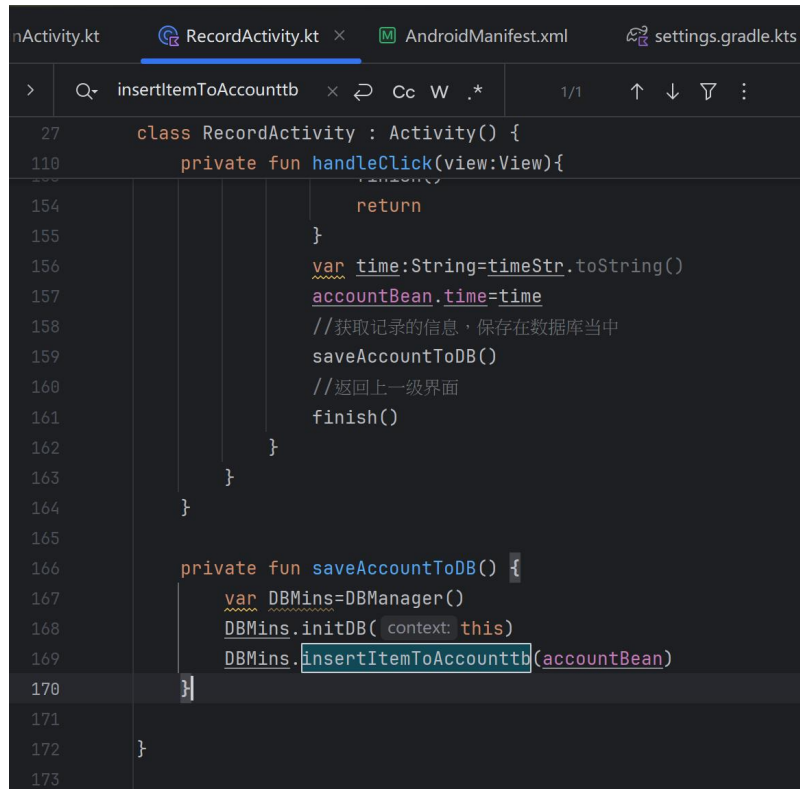
图 3-5 记帐录入界面 2



图 3-6 添加新支出记录后的首页展示

随后点击保存便可成功保存该条支出记录，系统自动退回至首页，方便用户查看到刚刚记录下来的记帐记录。同时位于顶部的收入和支出总额会自动作动态更新，以为用户提供最新的统计数据。

用户提交新的记账项后，`AccountBean` 对象通过 `DBManager` 的 `insertItemToAccounttb` 方法被插入到本地数据库中。



```
RecordActivity.kt
class RecordActivity : Activity() {
    private fun handleClick(view: View) {
        return
    }
    var time: String = timeStr.toString()
    accountBean.time = time
    // 获取记录的信息，保存在数据库当中
    saveAccountToDB()
    // 返回上一级界面
    finish()
}

private fun saveAccountToDB() {
    var DBMins = DBManager()
    DBMins.initDB(context = this)
    DBMins.insertItemToAccounttb(accountBean)
}
```

图 3-7 ‘insertItemToAccounttb’方法调用

4. 数据查询与更新

‘DBManager’提供了一系列方法来查询、更新和删除数据库中的记录。例如 ‘getAcntListFromAcnttb’ 用于获取特定日期的记账列表， ‘changeItemFromAccounttcById’ 用于更新记录， ‘deleteItemFromAccounttbById’ 用于删除记录。

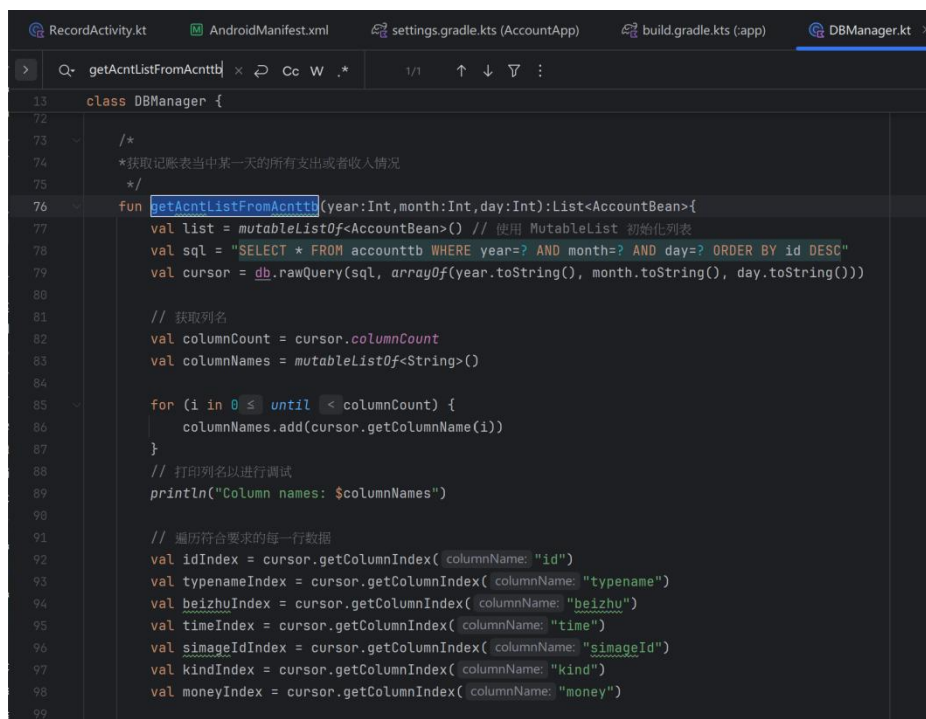


图 3-8 `getAcntListFromAcntttb`调用

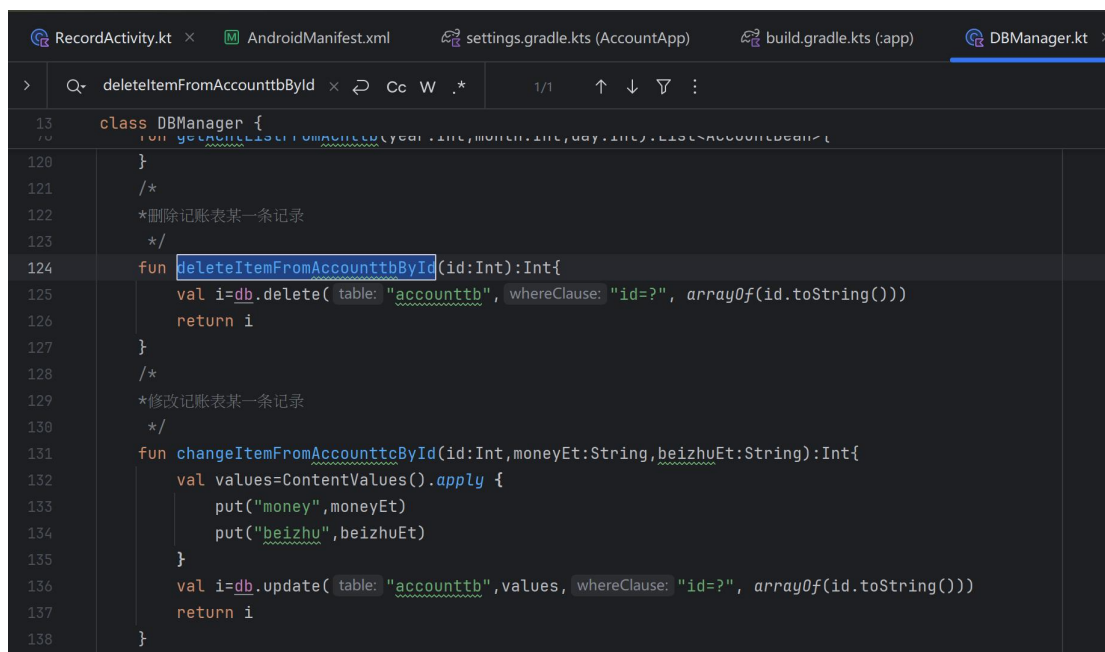


图 3-9 `changeItemFromAccounttcById`和`deleteItemFromAccounttbById`调用

点击其中一条记录，会弹出修改记录画面，在该画面中可以对该条记录进行类型、金额和备注上的修改，或者进行删除该条记录的操作。假如需要删除该条记录，系统会再次询问是否要删除该记录以作确认，避免用户误删除的情况。



图 3-9 点击其中一条记录

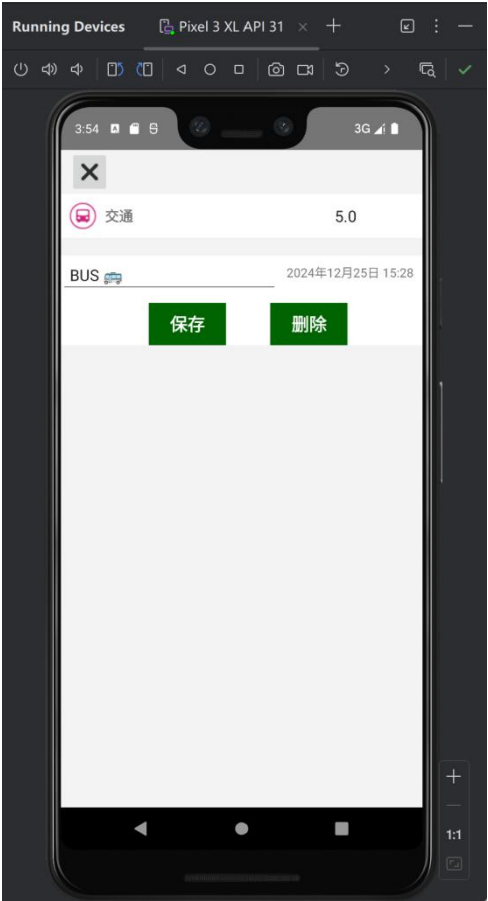


图 3-10 对该条记录进行修改



图 3-11 删除前确认



图 3-12 删除成功后的主页面

5. 事件处理

应用中的各个`Activity`负责处理用户事件，如按钮点击和列表项的长按。这些事件通过`View.OnClickListener`和`ListView.OnItemClickListener`来监听和响应。

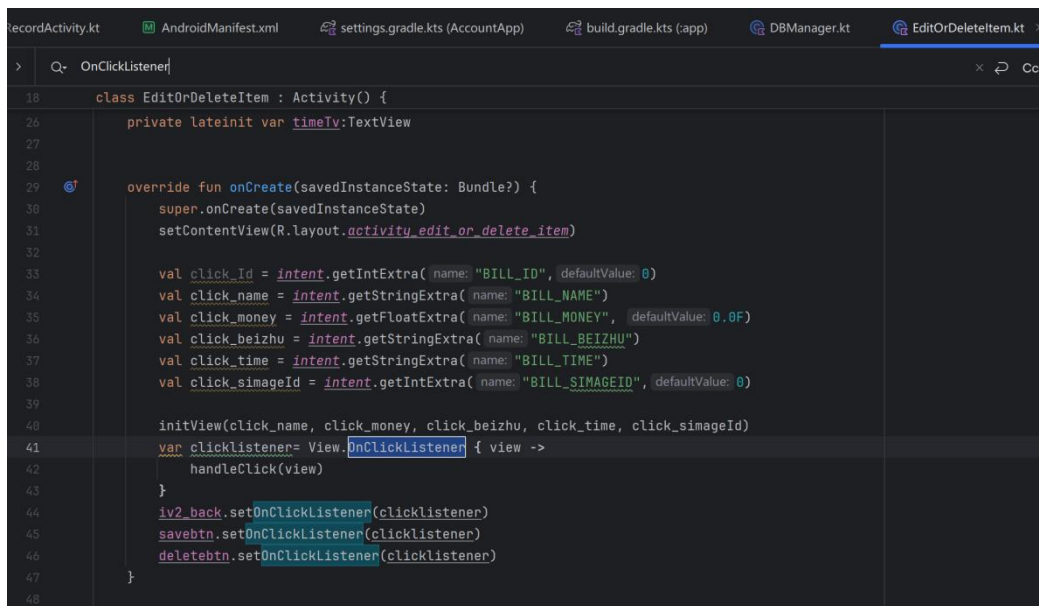


图 3-13 View.OnClickListener 调用

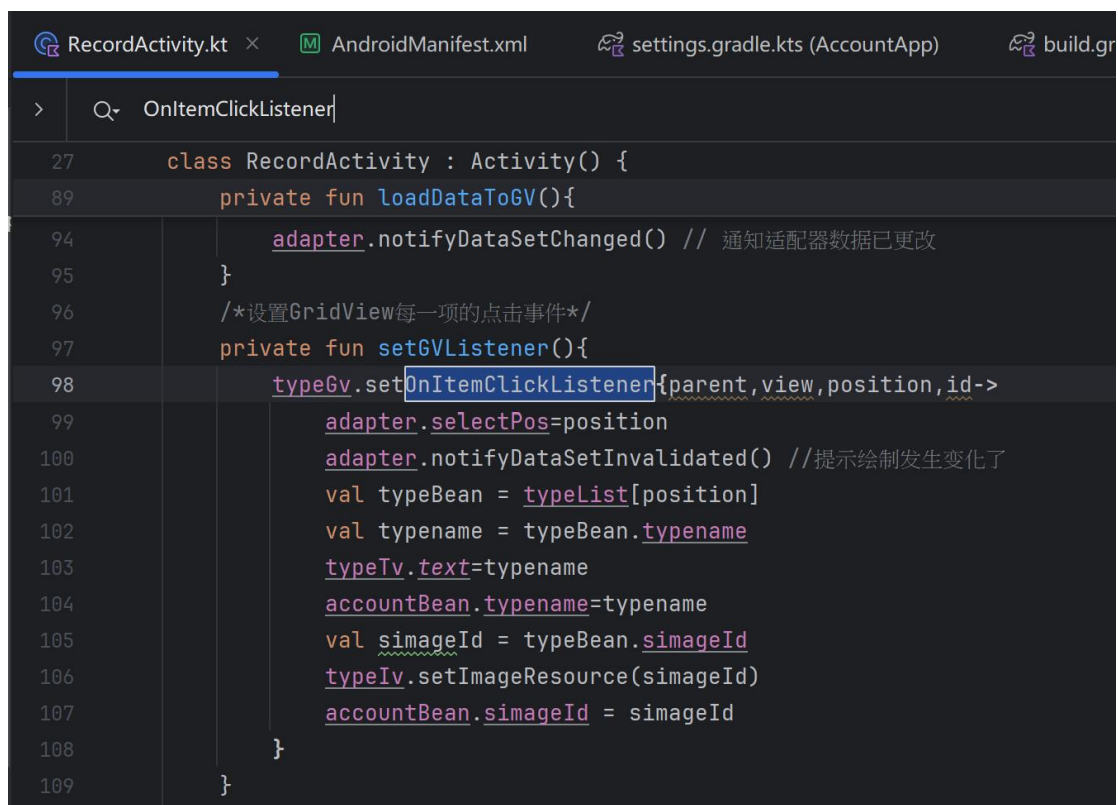


图 3-14 ListView.OnItemClickListener 调用

6. 数据适配与展示

`AccountAdapter`和`TypeBaseAdapter`作为数据适配器，负责将数据库查询结果转换为用户界面可以展示的格式，并在数据变化时更新列表视图。

```
AccountAdapter.kt x TypeBaseAdapter.kt MainActivity.kt RecordActivity.kt AndroidManifest.xml settings.gradle.kts (Acco
16 <> class AccountAdapter(private val context: Context, private val mDataas: List<AccountBean>): BaseAdapter() {
17     private val inflater: LayoutInflater = LayoutInflater.from(context)
18
19     val calendar: java.util.Calendar = java.util.Calendar.getInstance()
20     private var year: Int = calendar.get(Calendar.YEAR)
21     private var month: Int = calendar.get(Calendar.MONTH) + 1
22     private var day: Int = calendar.get(Calendar.DAY_OF_MONTH)
23
24     override fun getCount(): Int {
25         return mDataas.size
26     }
27
28     override fun getItem(position: Int): Any {
29         return mDataas[position]
30     }
31
32     override fun getItemId(position: Int): Long {
33         return position.toLong()
34     }
35
36     override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {
37         var holder: ViewHolder
38         val view: View = convertView ?: inflater.inflate(R.layout.item_mainlv, parent, attachToRoot: false).apply {
39             holder = ViewHolder(this)
40             tag = holder // 将 ViewHolder 存储在视图的标签中
41         }.let { it } // 如果 convertView 不为 null，这里的 let 不会执行
42
43         holder = convertView?.tag as? ViewHolder ?: ViewHolder(view) // 从 tag 中获取 ViewHolder
44
45         val bean: AccountBean = mDataas[position]
46         holder.typeIv.setImageResource(bean.simageId)
```

图 3-15 AccountAdapter 局部代码

```
AccountAdapter.kt x TypeBaseAdapter.kt MainActivity.kt RecordActivity.kt AndroidManifest.xml settings.gradle.kts (AccountAp
13 <> class TypeBaseAdapter(private val context: Context, private val mDataas: List<TypeBean>): BaseAdapter() {
14     var selectPos = 0 // 选中位置
15     override fun getCount(): Int {
16         return mDataas.size
17     }
18
19     override fun getItem(position: Int): Any {
20         return mDataas[position]
21     }
22
23     override fun getItemId(position: Int): Long {
24         return position.toLong()
25     }
26
27     override fun getView(position: Int, convertView: View?, parent: ViewGroup?): View {
28         val view = convertView ?: LayoutInflater.from(context).inflate(R.layout.item_record_gv, parent, attachToRoot: false)
29
30         // 查找布局中的控件
31         val iv: ImageView = view.findViewById(R.id.item_record_iv)
32         val tv: TextView = view.findViewById(R.id.item_record_tv)
33
34         // 获取指定位置的数据源
35         val typeBean = mDataas[position]
36         tv.text = typeBean.typeName
37
38         // 判断当前位置是否为选中位置，如果是选中位置，就设置为带颜色的图片，否则为灰色图片
39         iv.setImageResource(if (selectPos == position) typeBean.simageId else typeBean.imageId)
40
41         return view
42     }
43 }
```

图 3-16 TypeBaseAdapter 局部代码

7. 应用配置与资源管理

应用的配置和资源（如字符串、图片等）通过`res`目录下的资源文件进行管理。

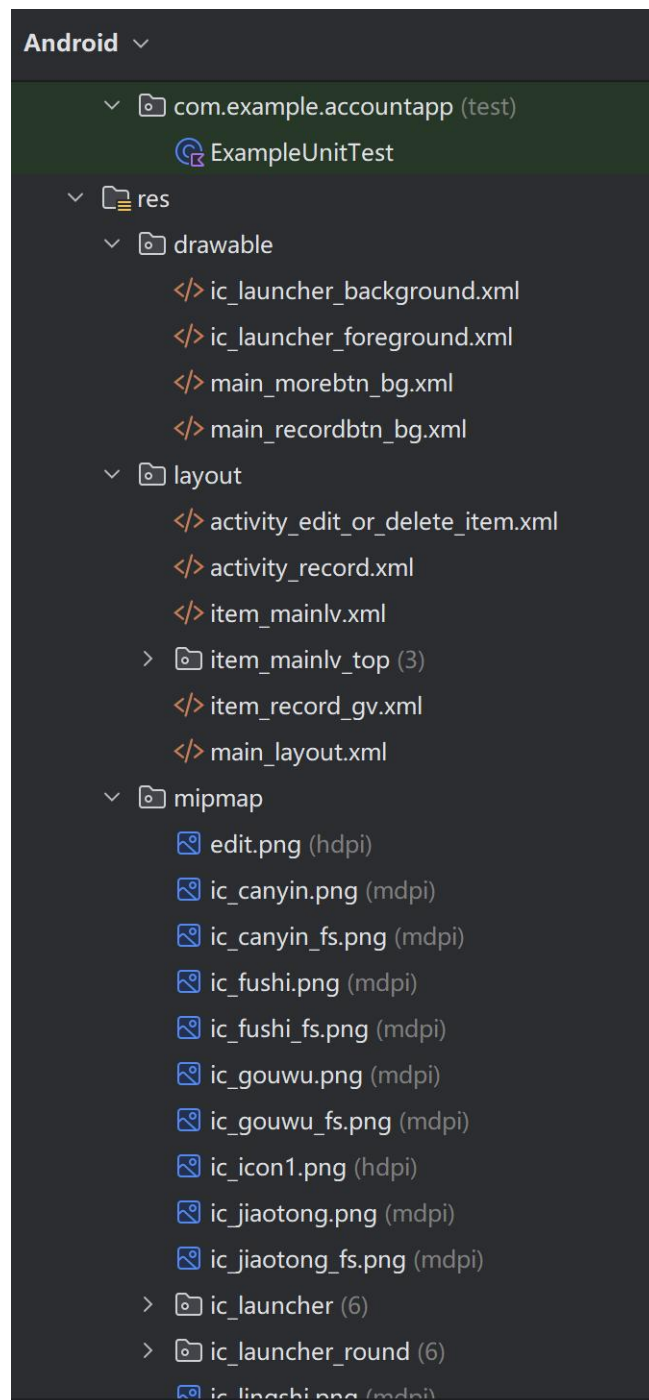


图 3-17 `res`目录下管理资源文件

8. 应用退出与资源释放

当用户退出应用时，系统会调用各个组件的`onDestroy`方法，此时应用会释放资源，包括关闭数据库连接等。

4 待开发功能

该应用程序后续的优化重点将瞄准以下数个方向：

1. 数据分析与图表可视化

让应用集成多种图表类型（如折线图、饼图、柱状图等），根据用户选择的时间段和分类，直观展示收支分布、趋势变化及消费结构，帮助用户发现消费习惯，做出更合理的财务规划。该方案在工程已有初步构建，但因本人时间精力有限未能顺利研发完成。

2. 预警提醒

允许用户设置预算上限，当接近或超出预算时自动发送提醒，帮助用户有效控制开支。

3. 数据备份与恢复

提供数据备份功能，确保用户数据在设备更换或意外情况下不丢失，并支持数据恢复操作。

4. 多账户管理

支持在同一应用内创建多个账户，适用于家庭成员各自管理或企业多部门财务记录。

5. 跨平台同步

支持手机、平板及计算机等多种终端设备的数据同步，确保用户在任何设备上都能查看最新的收支记录。

6. 智能建议：

基于用户的收支数据，提供个性化的节省建议和投资理财方案，助力用户实

现财富增值。

github 倉庫: <https://github.com/miracle-201/AccountApp>