“旺旺”理财小助手

使

用

说

明

梁润勇

2021年12月10日

目录

[一、 图标和启动 - 1 -](#_Toc17930)

[二、 登录注册 - 3 -](#_Toc9243)

[三、 管理主页面 - 5 -](#_Toc28294)

[四、 新增支出 - 7 -](#_Toc24302)

[五、 支出管理 - 10 -](#_Toc14479)

[六、 新增收入 - 12 -](#_Toc20501)

[七、 收入管理 - 14 -](#_Toc19218)

[八、 标签管理与编辑 - 15 -](#_Toc11603)

[九、 密码修改与侧边栏 - 19 -](#_Toc24908)

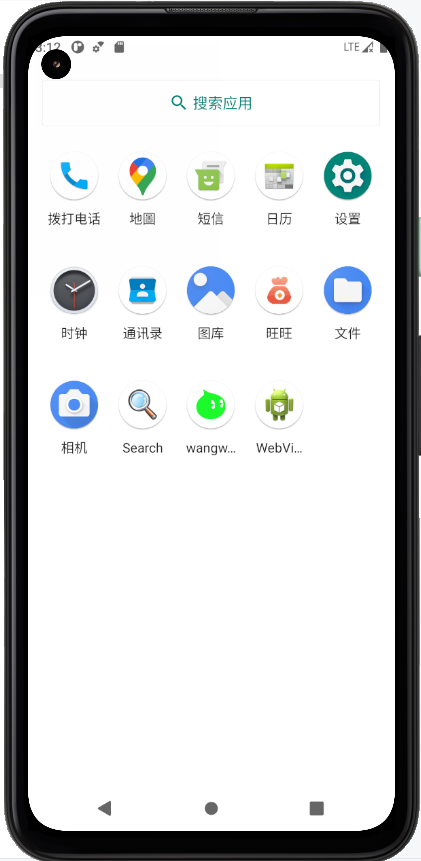
[十、 数据下载 - 20 -](#_Toc16816)

[十一、 程序退出 - 21 -](#_Toc3890)

[十二、 软件技术说明 - 21 -](#_Toc26850)

1. **图标和启动**

**（一）图标**



应用图标

图1 应用图标

**说明:**图标采用浅红色“旺旺”，寓意红火的收入。另外应用名称支持国际化，在设置中设置语言模式即可换成对应的国际化（目前支持中英文）。

**（二）启动页：**

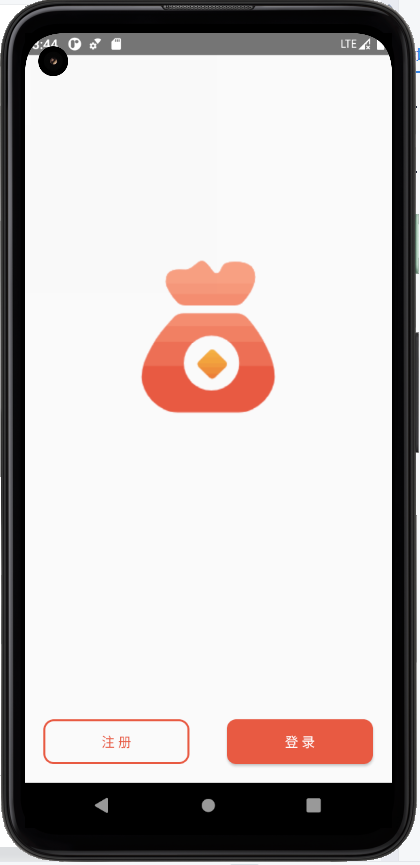
 

图2 启动页

**说明：**点击启动页面，出现启动页面，向左翻动页面，抵达登录、注册页面

1. **登录注册**
2. **注册**



图3 注册页面

说明：用户点击“注册”按钮，到达注册页面。输入基本信息->注册成功(以注册账号:1,密码:0举例使用)。

1. **登录**

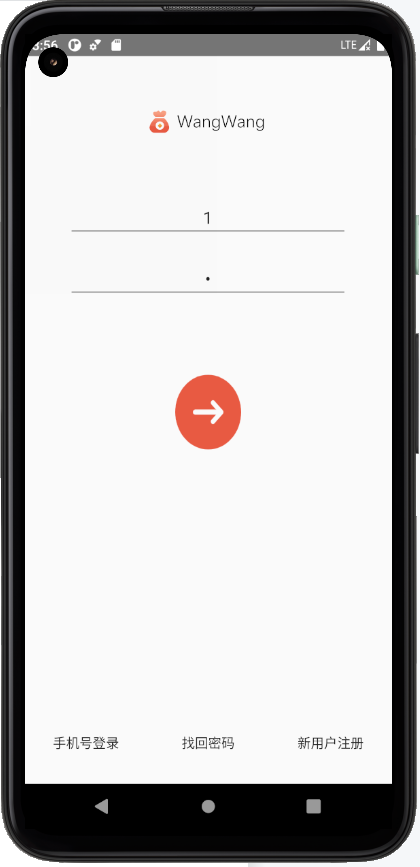


图4 登录页面

**说明:**当第一次登录时需要手动输入账号密码,之后软件自动提供记住密码服务.使用密码登录增强软件安全性

1. **管理主页面**



图5 管理主页面

**说明:**在管理主页面提供各功能模块的入口，用户点击各项，进入不同的模块进行操作。另外软件提供侧边栏，根据不同用户提供不同需求，同时侧边框还根据性别确定不同的头像以及快速修改密码的入口。



图6 侧边栏页面

**说明：**点击左边出现侧边栏，显示修改密码和对应性别的头像，后期将提供头像自定义上传功能。

1. **新增支出**
2. **新增支出页面**



图7 新增支出页面

1. **时间选择对话框**



图8 时间选择页面

**说明：**用户每次仅仅能点击时间进行选择，不能自定义时间，这样避免因不规范导致的问题。

**(三)下拉类型选择**



图9 下拉类型

**说明：**点击出现对话框，选择合适的支付类型，后期将提供自定义增加支付类型功能。

1. **支出管理**



月、周查询

支出总额

图10 支出记录

**说明：**在该页面，可以查看所有的支出记录，同时显示支出总额。用户还可以需要查任意月份支出、按月和周查询一周的支出。此时支出金额也会自动发生变化，计算一月、一周的支出。

另外点击可以选择删除，在删除中显示用户曾经的备注。



图11 支出记录删除

1. **新增收入**



图12 新增收入页面

**说明：**点击新增收入，进入页面



图13 新增收入页面

1. **收入管理**

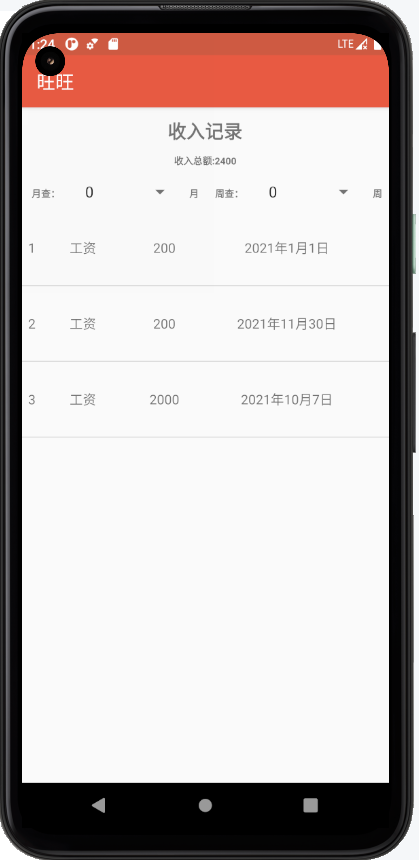
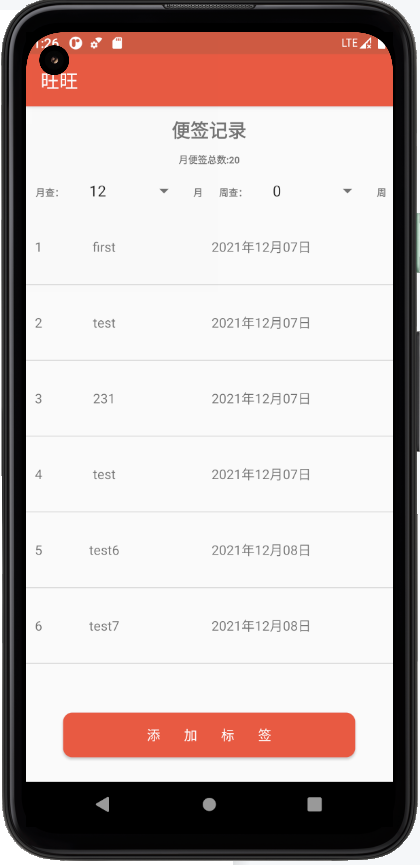


图14 收入记录管理页面

1. **标签管理与编辑**
2. **便签展示页面**



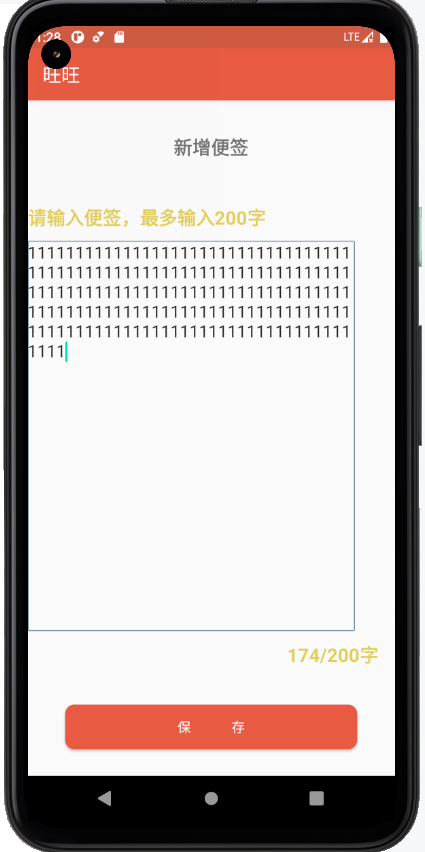
添加标签

月、周查询

标签计数

图15 便签展示页面

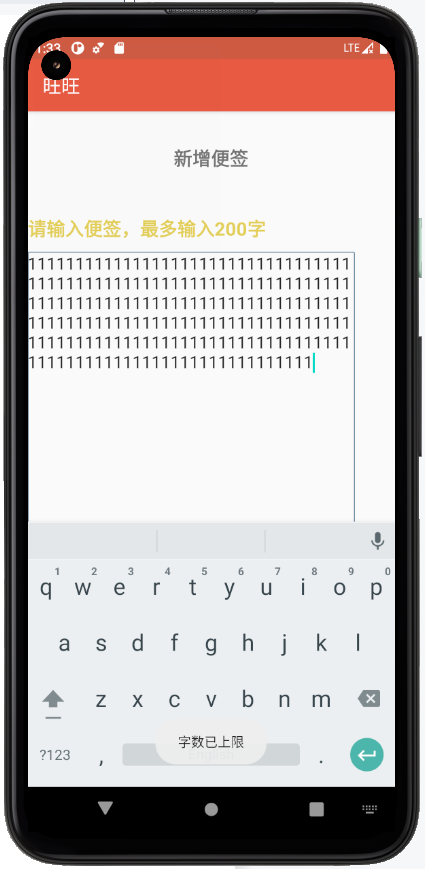
1. **添加标签页面**



计算字数

图16 新增标签展示

**说明：**可以根据用户的输入自动计算字数，当字数达到上限时自动弹出提示，并禁止继续编辑。



字数提示

图17 字数超限提示

1. **标签编辑与删除页面**



修改标签名按钮

修改标签内容按钮

修改提交按钮

删除标签按钮

图18 标签编辑与删改页面

1. **密码修改与侧边栏**

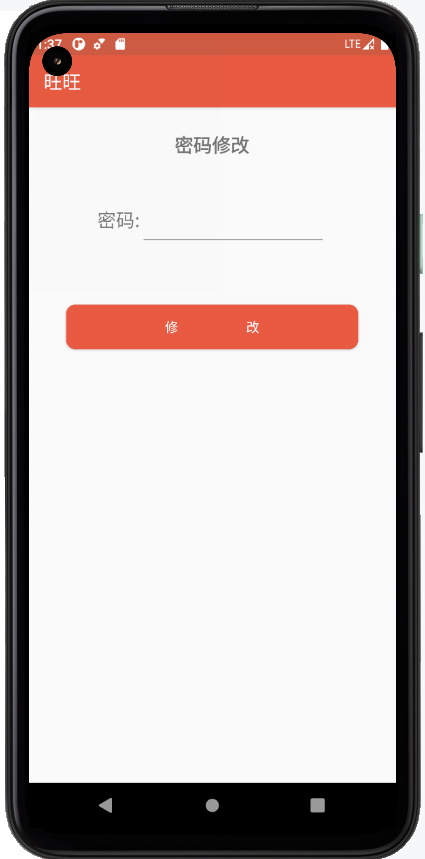


图19 系统设置页面

**说明：**点击系统设置或侧边栏修改密码，进入系统设置页面修改密码。修改密码后将自动弹出到登录页面进行重新登录。

1. **数据下载**



图20 数据管理页面

**说明：**点击数据管理，进入下载页面，可以选择任意文件进行下载，下载支持txt下载和excel两种格式，用户点击任意一种即可下载。后期软件更新后提供数据可视化分析，自动生成图标等。



图21 下载说明图

1. **程序退出**

点击“退出”，app退至登录页面

1. **软件技术说明**

**总体设计**

**(一)背景知识**

软件主要是涉及Android UI和sqlite数据库知识，另外还设计activity的页面跳转以及启动模式设置。

**1.Android UI方面主要考虑以下四个方面的知识**

1).ListView和Adapter适配器的使用。

2).多线程处理下Handler消息机制动态处理UI。

3).res库中drawable样式制作、图片加载；Values国际化设置；anim动图效果；

4).fragment的学习以及滑动切换页面的设置。

**2.Sqlite数据库方面主要考虑以下几个方面**

1).持久化工具Room 持久性库：和web后端一样，我网上查询并学习了RoomDatabase持久化工具。做到熟练使用RoomDatabase进行sqlite数据库开发。同时逐步学习并了解了谷歌推荐的Jetpack component。

2).另外在开发中采用多线程机制，加快数据请求处理。

**Activity页面跳转与启动模式修改**

1).在页面跳转做“返回功能”时，出现back弹回原先页面的bug。此时需将返回到的页面设计为singleTask。

2).同时为了响应被返回页面的消息，返回到的页面可以在onRestart()中设置方法做改动。

**(二)基本原理**

Handler消息机制

Adapter

Room持久化工具

**(三)模块介绍**

个人理财小助手主要为用户提供 5 个主要模块，包括：登录模块、收入管理模块、支出管理模块、便签管理模块、系统设置模块。对日常的开支进行随时记录，存储到本机的 SQLite 数据库。

使用 SQLite 数据库实现增加、删除、修改；

使用 ListView 进行显示；

增加私密性，验证成功才能进行收支管理；

能提供简单的收支统计（按月、按周）功能；

提供数据管理，能对数据进行导出（如将收入/支出明细导出为 txt 或Excel 文档）。

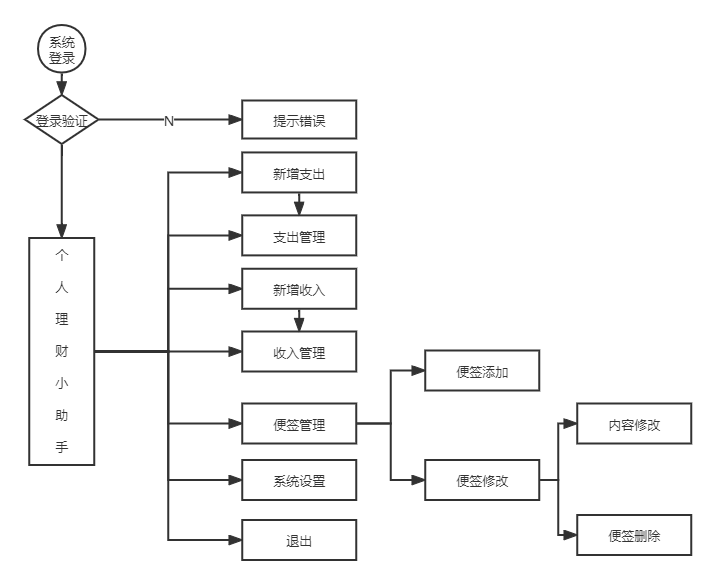


图1 系统结构图

**(四)设计步骤**

1)Room数据库环境配置、安装与学习。

2)数据库设计，包含表的键、约束、索引等设计。

3)个人登录页面的设计和密码保存设计。

4)新增支出页面：时间对话框设计、下拉框spinner学习与设计、时间的处理与插入。

5)支出管理页面：按月、周查询，listView的动态展示，数据增删改查。

6)新增收入页面：与新增支出页面大致相同。

7)收入管理页面：与支出管理页面大致相同。

8)便签管理页面：便签的展示，添加便签选项、查看、修改、删除便签事件。

9)系统设置：密码的修改。

10)侧边框的设计，在侧边框根据用户性别调整头像，同时也提供密码的修改。

11)退出：activity进程的杀死

**详细设计**

**(一)滑动切换页面设计**

使用ViewPager组件和FragAdapter适配器。创建四个fragment，其三个页面每个页面仅一个组件ImageView，每个ImageView一个海报。最后一个fragment制作选择登录页面、注册页面。

List<Fragment> fragments = new LinkedList<Fragment>();

fragments.add(new Poster1());

fragments.add(new Poster2());

fragments.add(new Poster3());

fragments.add(new main());

FragAdapter fragAdapter=new FragAdapter(getSupportFragmentManager(),fragments);

ViewPager viewPager=(ViewPager)findViewById(R.id.login\_viewpager);

viewPager.setAdapter(fragAdapter);

1. **数据库设计**

**表1 用户表**

| account | password | sex |
| --- | --- | --- |
| 1 | 0 | 男 |
| 2 | 1 | 女 |

**表2 支出表**

| id | account | payMoney | payDate | month | week | payType | payPlace | notes |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 200 | 2021年1月1日 | 1 | 1 | 吃饭 | Beijing | Test01 |
| 1 | 1 | 50 | 2021年1月1日 | 1 | 1 | 吃饭 | Beijing | Test02 |

**表3 收入表**

| id | account | IncomeMoney | incomeDate | month | week | incomeType | payPeople | notes |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 150 | 2021年1月1日 | 1 | 1 | 工资 | Beijing | Test01 |
| 1 | 1 | 100 | 2021年1月1日 | 1 | 1 | 基金 | Beijing | Test02 |

**表4 标签表**

| id | account | name | notesDate | month | week | content |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | test01 | 2021年12月07日 | 12 | 2 | 第一个标签 |
| 1 | 1 | Test02 | 2021年12月08日 | 12 | 2 | 字数测试 |

1. **Room数据库设计**

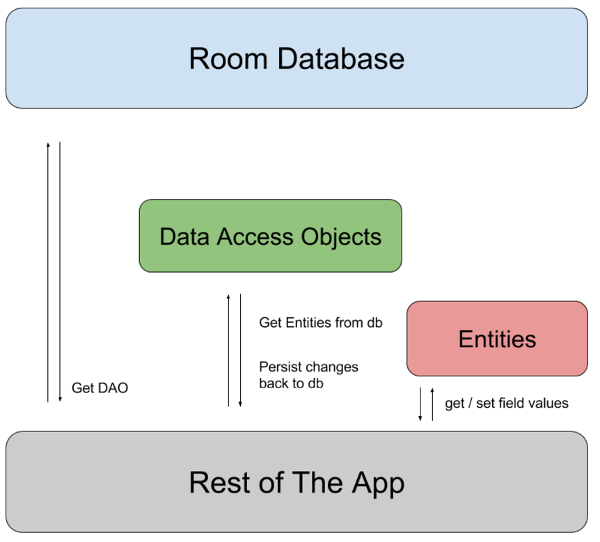


图2 ROOM架构图

**1.Room 包含 3 个主要组件：**

**[数据库](https://developer.android.google.cn/reference/androidx/room/Database?hl=zh-cn)：**包含数据库持有者，并作为应用已保留的持久关系型数据的底层连接的主要接入点。

使用 [@Database](https://developer.android.google.cn/reference/androidx/room/Database?hl=zh-cn) 注释的类应满足以下条件：

是扩展 [RoomDatabase](https://developer.android.google.cn/reference/androidx/room/RoomDatabase?hl=zh-cn) 的抽象类。

在注释中添加与数据库关联的实体列表。

包含具有 0 个参数且返回使用 [@Dao](https://developer.android.google.cn/reference/androidx/room/Dao?hl=zh-cn) 注释的类的抽象方法。

在运行时，可以通过调用Room.inMemoryDatabaseBuilder()获取 [Database](https://developer.android.google.cn/reference/androidx/room/Database?hl=zh-cn) 的实例。

**[Entity](https://developer.android.google.cn/training/data-storage/room/defining-data?hl=zh-cn)：**表示数据库中的表。

**[DAO](https://developer.android.google.cn/training/data-storage/room/accessing-data?hl=zh-cn)：**包含用于访问数据库的方法。

**2.Room数据库依赖导入：**

**在官网中复制导入依赖代码，使用gradle导入依赖**

//导入Room数据库

def room\_version = "2.3.0"

implementation "androidx.room:room-runtime:$room\_version"

annotationProcessor "androidx.room:room-compiler:$room\_version"

// optional - RxJava2 support for Room

implementation "androidx.room:room-rxjava2:$room\_version"

// optional - Guava support for Room, including Optional and ListenableFuture

implementation "androidx.room:room-guava:$room\_version"

**3.Entity设计:**

package com.example.wangwang.entity;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.room.Entity;

import androidx.room.PrimaryKey;

@Entity

public class User {

@PrimaryKey

@NonNull

private String account;

private String password;

private String sex;

public User(String account, String password, String sex) {

this.account = account;

this.password = password;

this.sex = sex;

}

public String getAccount() {

return account;

}

public void setAccount(String account) {

this.account = account;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getSex() {

return sex;

}

public void setSex(String sex) {

this.sex = sex;

}

@Override

public String toString() {

return "User{" +

"account='" + account + '\'' +

", password='" + password + '\'' +

", sex='" + sex + '\'' +

'}';

}

}

注意点：要保留某个字段，Room 必须拥有该字段的访问权限。您可以将某个字段设为公开字段，也可以为其提供 getter 和 setter。如果使用 getter 和 setter 方法，请注意，这些方法需遵循 Room 中的 JavaBeans 规范。

**4.Dao层设计**

package com.example.wangwang.dao;

import androidx.room.Dao;

import androidx.room.Delete;

import androidx.room.Insert;

import androidx.room.Query;

import androidx.room.Update;

import com.example.wangwang.entity.User;

import java.util.List;

@Dao

public interface UserDao {

@Insert

public Long insertUser(User user);

@Update

public void updateUser(User user);

@Delete

public void deleteUser(User user);

@Query("SELECT \* FROM user")

public List<User> loadAllUsers();

@Query("SELECT \* FROM user WHERE account= :account")

public List<User> findAllUserByAccount(String account);

@Query("SELECT \* FROM user WHERE account= :account and password=:password")

public List<User> loginUser(String account, String password);

}

方法参数与sql语句参数绑定方法：在编译时处理此查询时，Room 会将 :account 绑定参数与account 方法参数进行匹配。Room 通过参数名称进行匹配。如果有不匹配的情况，则应用编译时会出现错误。

**5.数据库层设计**

@Database(entities = {User.class, Expenditure.class, Income.class, Notes.class}, version = 1,exportSchema=false)

public abstract class FinancialAssistant extends RoomDatabase {

public abstract UserDao userDao();

public abstract ExpenditureDao ExpenditureDao();

public abstract IncomeDao incomeDao();

public abstract NotesDao NotesDao();

}

分别指定了UserDao、ExpenditureDao、IncomeDao、NotesDao的获取方法。

RoomDatabase实例创建方式：

AppDatabase

db=Room.databaseBuilder(getApplicationContext(),AppDatabase.class,"database-name").build();

1. **登录验证页面设计**

输入账号密码、用户点击提交，此时查找指定账号密码的账号，若结果为空则账号密码不正确。(注意使用线程设计)

new Thread(new Runnable() {

@Override

public void run() {

FinancialAssistant db = Room.*databaseBuilder*(getApplicationContext(), FinancialAssistant.class, "financial\_assistant").build();

UserDao userDao = db.userDao();

List user = userDao.loginUser(account, password);

Message message =new Message();

if (user.size() == 0) {

message.what=1;

handler.sendMessage(message);

}else{*//登录验证成功*

message.what=0;

handler.sendMessage(message);

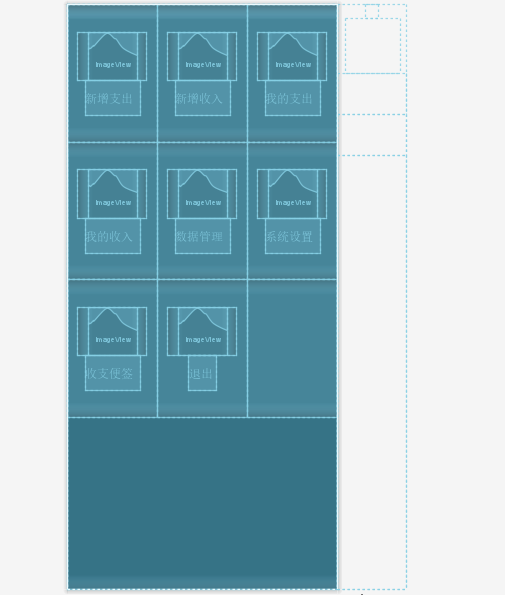
}

}

}).start();

1. **主页面设计**

采用DrawerLayout布局作为根布局，来实现侧边栏，里面含有两个LinearLayout。主页面的LinearLayout里面使用TableLayout布局，将各功能模块做成表格布局。另一个LinearLayout则实现侧边栏功能。



**图3 功能页面布局结构图**

1. **新增支出设计**

该部分主要包括时间对话框的使用以及对时间的处理、存储。

首先设置EditView组件，使得其点击即触发弹窗，而避免先聚焦在弹窗（需点击两次）

editText = findViewById(R.id.*pay\_money\_time*);

editText.setFocusable(false);

为editText设置监听器，点击触发后执行时间对话框操作。

@Override

public void onClick(View view) {

if (view.getId() == R.id.*pay\_money\_time*) {

Calendar calendar = Calendar.*getInstance*();

DatePickerDialog datePickerDialog = new DatePickerDialog(this, this, calendar.get(calendar.*YEAR*), calendar.get(calendar.*MONTH*), calendar.get(calendar.*DAY\_OF\_MONTH*));

datePickerDialog.show();

}

选定时间后点击确定，DatePickerDialog.OnDateSetListener执行触发事件onDateSet：

@Override

public void onDateSet(DatePicker datePicker, int year, int month, int day) {

String desc = String.*format*("%d年%d月%d日", year, month + 1, day);

editText.setText(desc);

}

这样便规定了输入时间的统一格式为XX年XX月XX日

1. **支出管理设计**

在该模块主要实现按月、周查询，利用ListView组件动态展示数据，同时支持数据的增删改查。

**1.Adapter自定义**

Adapter代码需要自定义，使用Holder机制以及convertView变量对adapter优化：

Adapter代码：

package com.example.wangwang.adapter;

import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.BaseAdapter;

import android.widget.TextView;

import com.example.wangwang.R;

import com.example.wangwang.entity.Expenditure;

import java.util.LinkedList;

import java.util.List;

public class MyPayAdapter extends BaseAdapter {

private Context context;

private List<Expenditure> list = new LinkedList<Expenditure>();

public void addData(Expenditure expenditure) {

list.add(expenditure);

notifyDataSetChanged();

}

public void changeData(List list){

this.list=list;

notifyDataSetChanged();

}

public MyPayAdapter(Context context, List<Expenditure> list) {

this.context = context;

this.list = list;

}

@Override

public int getCount() {

return list.size();

}

@Override

public Object getItem(int i) {

return list.get(i);

}

@Override

public long getItemId(int i) {

return i;

}

@Override

public View getView(int i, View view, ViewGroup viewGroup) {

ViewHolder viewHolder;

if (view == null) {

view = LayoutInflater.from(context.getApplicationContext()).inflate(R.layout.my\_pay\_list\_item, viewGroup, false);

viewHolder = new ViewHolder();

viewHolder.textViewNum = view.findViewById(R.id.my\_pay\_list\_number);

viewHolder.textViewType = view.findViewById(R.id.my\_pay\_list\_type);

viewHolder.textViewMoney = view.findViewById(R.id.my\_pay\_list\_money);

viewHolder.textViewTime = view.findViewById(R.id.my\_pay\_list\_time);

viewHolder.textViewId = view .findViewById(R.id.my\_pay\_list\_id);

view.setTag(viewHolder);

} else {

viewHolder = (ViewHolder) view.getTag();

}

Expenditure expenditure = list.get(i);

int num = i+1;

String type = expenditure.getPayType();

int money = expenditure.getPayMoney();

String time = expenditure.getPayDate();

int id = expenditure.getId();

viewHolder.textViewNum.setText(""+num);

viewHolder.textViewType.setText(type);

viewHolder.textViewMoney.setText(""+money);

viewHolder.textViewTime.setText(time);

viewHolder.textViewId.setText(""+id);

return view;

}

class ViewHolder {

TextView textViewNum;

TextView textViewType;

TextView textViewMoney;

TextView textViewTime;

TextView textViewId;

}

}

通过设置成员变量ListView以及提供changeData(List list)方法来实现数据的动态修改。

**2.Spinner组件**

Spinner提供选项功能来支持按月、按周查询。由于是移动端使用，为了增强用户体验，spinner设置为弹窗而非下拉更为适合。android:spinnerMode="dialog"

同时spinner事件为ItemSelected。代码如下:

spinnerMonth.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

@Override

public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {

String month = months[i];

Message msg = new Message();

msg.what = 0;

msg.obj = month;

handler.sendMessage(msg);

}

@Override

public void onNothingSelected(AdapterView<?> adapterView) {

}

});

spinnerWeek.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

@Override

public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {

int month\_index = spinnerMonth.getSelectedItemPosition();

if (month\_index != 0) {//等于0全部列出，在月中全列就可以了

String week = weeks[i];

String month = months[month\_index];

HashMap hashMap = new HashMap();

hashMap.put("week", week);

hashMap.put("month", month);

Message msg = new Message();

msg.what = 1;//按周查询实际上是周月同时查询

msg.obj = hashMap;

handler.sendMessage(msg);

}

}

两者通过Item选择触发消息机制，进行增删改查，修改UI（ListView）

HashMap hashMap = (HashMap) msg.obj;

int week = Integer.parseInt((String) hashMap.get("week"));

int month = Integer.parseInt((String) hashMap.get("month"));

if (week!=0) {//不等于0按周查

Thread thread = new Thread(new Runnable() {

@Override

public void run() {

FinancialAssistant db = Room.databaseBuilder(getApplicationContext(), FinancialAssistant.class, "financial\_assistant").build();

ExpenditureDao expenditureDao = db.ExpenditureDao();

list = expenditureDao.selectExpenditureByMonthAndWeek(account,month, week);

for (Expenditure expenditure : list) {

count = count + expenditure.getPayMoney();

}

}

});

thread.start();

try {

thread.join();

countView.setText("周支出总额:" + count);

myPayAdapter.changeData(list);

count = 0;

} catch (InterruptedException e) {

e.printStackTrace();

}

}else if(week==0){//周为0，则全月查询

Thread thread = new Thread(new Runnable() {

@Override

public void run() {

FinancialAssistant db = Room.databaseBuilder(getApplicationContext(), FinancialAssistant.class, "financial\_assistant").build();

ExpenditureDao expenditureDao = db.ExpenditureDao();

list = expenditureDao.selectExpenditureByMonth(account,month);

for (Expenditure expenditure : list) {

count = count + expenditure.getPayMoney();

}

}

});

thread.start();

try {

thread.join();

countView.setText("月支出总额:"+count);

myPayAdapter.changeData(list);

count = 0;

} catch (Exception e) {

e.printStackTrace();

}

}

逻辑是当使用按月查询时，如果选择0，则查询所有支出记录；如果非0则支持本月交易记录全部查询。当月份非0时，如果周查询选择0，则本月全部查询，如非0则按照指定周查询。

**3.支出记录删除**

由于支出记录名字可以重复、备注等属性都可以重复，且相同年月日也可以存储多个标签，因此我使用自增来实现主键的存储，同时在每一条Item中设置一个隐藏主键TextView存储该主键。当点击Item时获取该条的隐藏组件，获取自增Id进行删除操作。

public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {

TextView textView = view.findViewById(R.id.my\_pay\_list\_id);

int id = Integer.parseInt(textView.getText().toString());//隐藏Tetxview内容获取

new Thread(new Runnable() {

@Override

public void run() {

FinancialAssistant db = Room.databaseBuilder(getApplicationContext(), FinancialAssistant.class, "financial\_assistant").build();

ExpenditureDao expenditureDao = db.ExpenditureDao();

List<Expenditure> list = expenditureDao.selectExpenditureById(id);

Expenditure expenditure = list.get(0);

Message message = new Message();

message.what = 2;

message.obj = expenditure;

handler.sendMessage(message);

}

}).start();

}

1. **便签管理设计**

主要解决便签的增加修改、删除。以及Intent跳转下activity的启动模式。

**1.标签的展示：**

与支出管理、收入管理一样，进入页面后展示所有标签及其名字；同时也支持按月、周查询数据。

**2.标签的创建**：

点击“创建”按钮自动跳转到创建页面；创建页面主要是设置EditView的字数限制，同时设置动态显示输入字数。当字数达到规定限制时，显示Toast，提示字数上限。

EditView字数改变事件为TextChanged事件，其监听器主要有改变前、改变、改变后的设置，改变时动态计算其输入长度，同时使用TextView组件动态显示总字数：

editText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

}

@Override

public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {

textView.setText(String.valueOf(charSequence.length()));

}

@Override

public void afterTextChanged(Editable s) {

content = editText.getText().toString();

textView.setText(String.valueOf(s.length()));

if (s.length() == 200) {

Toast.makeText(AddNotesActivity.this, "字数已上限", Toast.LENGTH\_SHORT).show();

}

}

});

3.标签的删除、修改

用户在标签展示页面选择任意标签点击，获取隐藏TextView内容，按照自增Id（与支出页面删除同一原理），获取该条标签记录；同时跳转到标签编辑页面。在标签编辑页面将获取的记录展示出来，并提供用户修改。用户在修改页面可以选择修改也可以选择删除操作。

4.activity的启动模式设置，进入编辑页面，在编辑页面，其可以更改标签名字还可以更改标签内容。

我们不得不考虑当进入修改页面或者添加页面后，点击保存或者添加按钮，执行记录修改和添加后使用Intent跳回至展示页面。此时若点击back按钮不会返回至主页面，而是跳转到了保存页面、修改页面，这是不符合逻辑的！原因在于activity采用了栈保存，默认采用的启动方式是standard模式；当点击跳转时是直接采取了创建新的展示页面activity实例，点击back后弹出栈，原先的保存页面activity被再次置于栈顶。因此我们改变展示activity的启动模式，将其设置为SingleTask，即选择跳转回展示页面时，先查看展示页面是否处于栈中，若处于栈中则直接弹出所有在其上的activity来返回至原先的activity中。当然为了在修改后动态刷新展示页面，这个实例没有被再次创建，因此我们需要重写展示页面的OnRestart方法即可：

@Override

protected void onRestart() {

super.onRestart();

spinnerMonth.setSelection(0);

}

这样便实现了编辑后动态刷新以及activity正确启动。

1. **系统设置与密码设置**

在系统设置中主要实现密码修改功能，为了方面以及多样性，我还选择增加了侧边栏修改密码的功能。点击系统设置或者侧边栏密码修改即可实现跳转密码修改页面。密码修改主要是获取账号、查询用户，提交新密码替换即可。

侧边栏的设计主要是使用了DrawerLayout布局即可。

1. **程序退出**

**杀死当前Activity即可返回至登录页面：**

else if (view.getId() == R.id.exit) {

android.os.Process.killProcess(android.os.Process.myPid());

System.exit(0);

}