

**软件工程(I)**

**小米便签开源代码**

**维护需求与设计方案**



**组长学号姓名： 210340170 薛皓天**

**成员学号姓名： 210340169 薛傲翔**

**210340139 陈政杨**

**210340157 刘薇**

## 二〇二三年四月

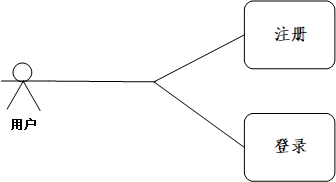
# 1.软件维护需求描述

## 1.1 开源软件分析

所有的便签都在统一的界面进行增加，查找和删除，通过一个界面都可以看到其他人写的全部便签，缺少了私密的概念。因此我们需要一个登陆界面，去进行控制使用便签的用户可见的内容。

## 1.2 开源软件维护需求

1.用例图



2.用例描述

用例名：登陆界面

业务目的：登录小米便签，注册账号

基本交互动作：

1、输入用户名和密码后点击login登录进入

2、点击注册输入用户名、密码、邮箱、姓和名后注册成功

3、进入属于自己的小米便签

4、点击返回直接退出软件

扩展交互动作：

1a、(1)当用户登录时用户名和密码其中有一个为空

1a1、提示”ERROR”并让用户重新输入

(2)当没有找到用户时

1a2、提示需要注册

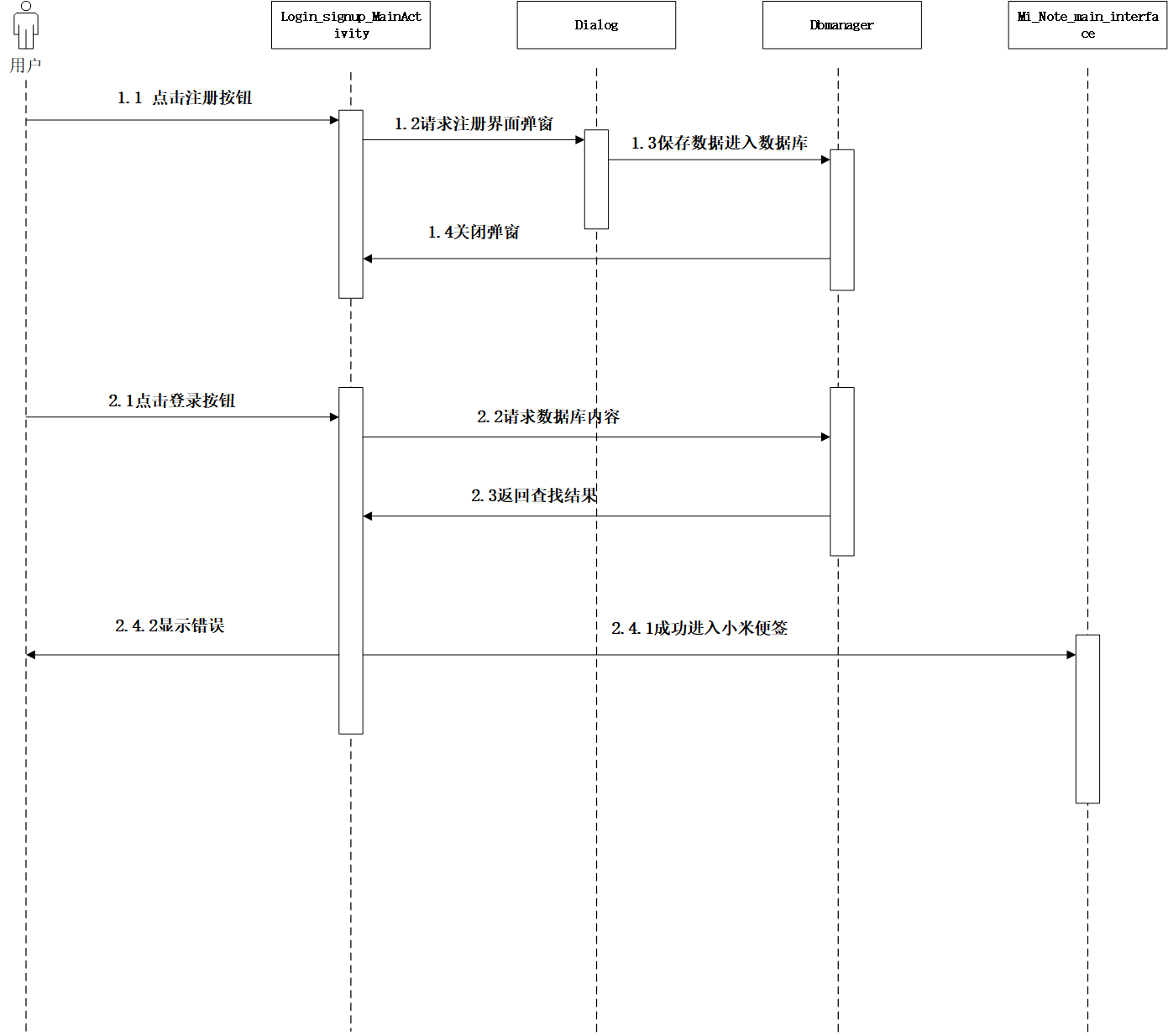
2a、当用户输入信息中有一项或多项为空时

2a1、提示”ERROR”即让其重新输入

后置条件：当点击小米便签后，立即进入注册界面，退出即退出小米便签。

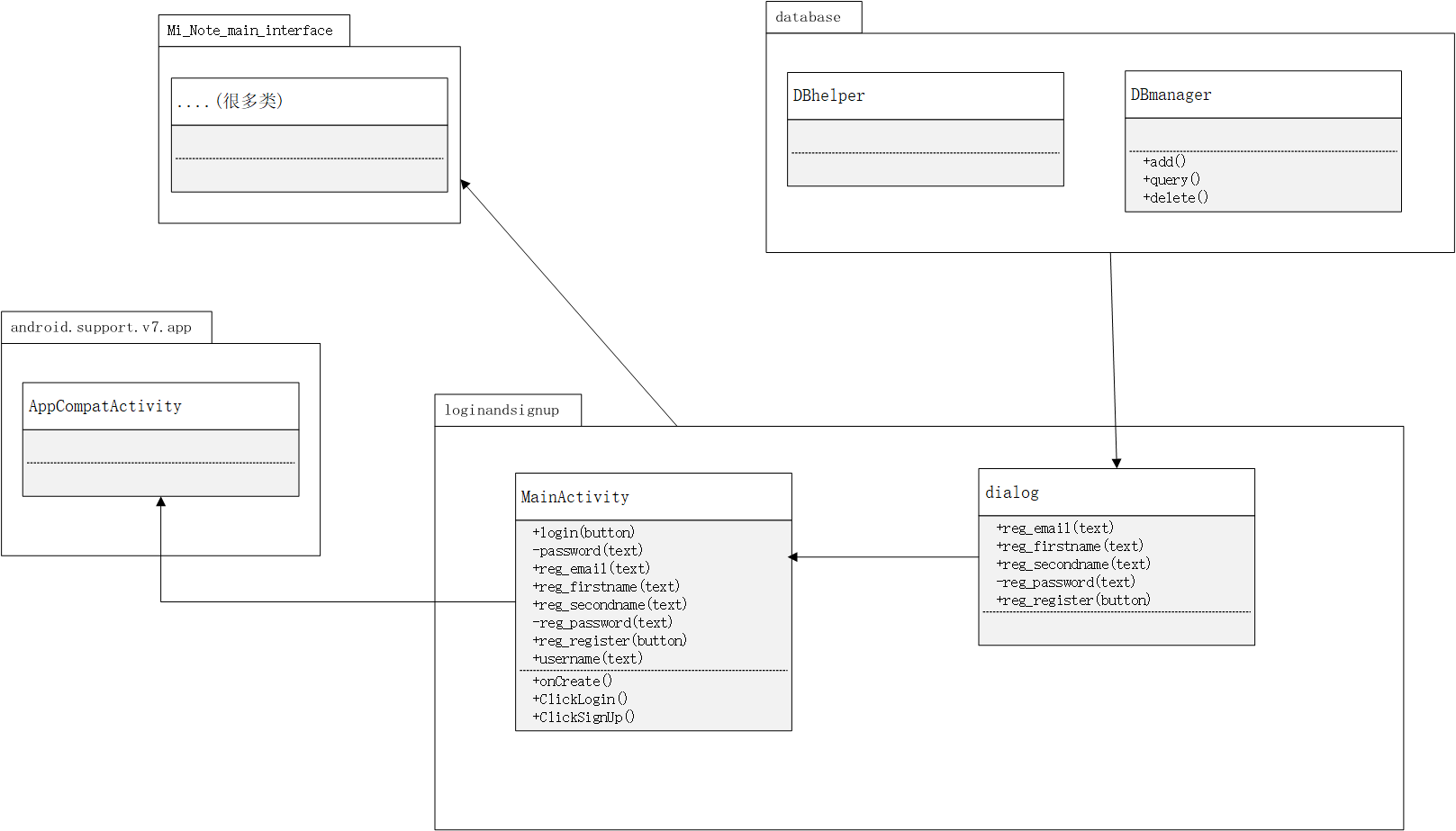
# 2.软件维护需求分析

## 2.1 用例顺序图



其中：小人表示用例的驱动者，当然就是用户；主界面应该是login\_signup\_MainActivity表示每一次打开页面首先看到的页面就是登录界面。在顺序图中1.2、1.3、1.4是通过dialog弹窗和DBmanager数据库将数据进行输入和保存的操作，2.2、2.3是通过DBmanager数据库进行查询操作，系统应该为小米便签的开源软件。

## 2.2 分析类图



1. Loginandsignup包：

（1）继承android.support.v7.app中的AppCompatActivity（该类的作用是添加一个应用 Activity 类，此类可作为使用支持库操作栏实现的 Activity 的基础类）

（2）该包包含两个重要的类：一个是MainActivity的主界面类，另一个是dialog类（弹窗获取用户的各种信息）

2、database包：

（1）DBhelper类：帮助创建数据库（其中包括使用的sqlite数据库的版本和create语句）

（2）DBmanager类：提供增，查的功能（其中删除和更改操作并不需要）

3、Mi\_Note包：

存放的是小米便签开源软件的所有内容

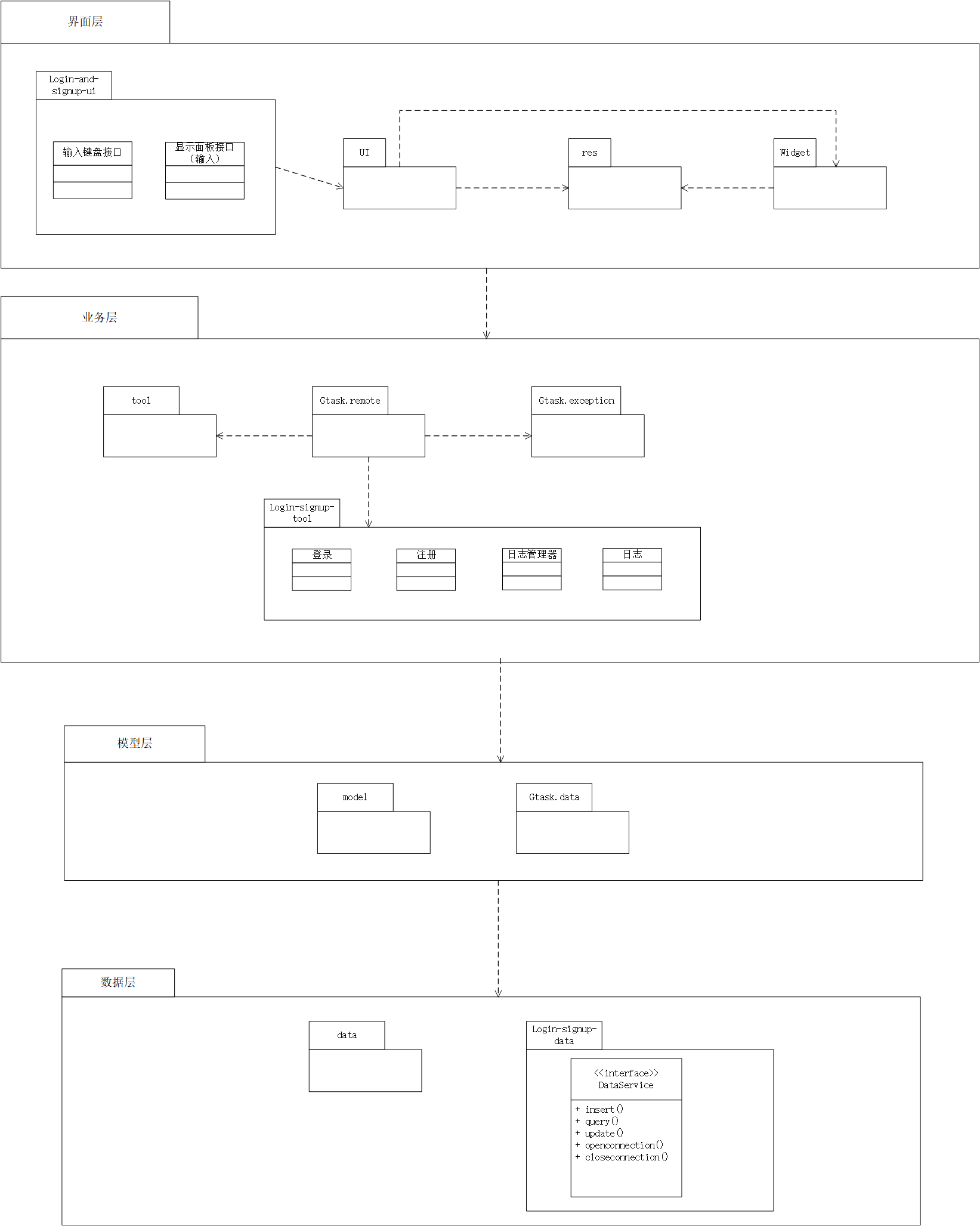
# 3.维护设计和实现方案

## 3.1 开源软件体系结构的设计及调整

原小米便签逻辑视图：



加入新功能后的逻辑视图：



分析：加入新功能后的逻辑视图与原图的区别在于，界面层、业务层、数据层。

1. 界面层：应该在小米便签的UI包中加入login\_and\_signup\_ui界面（该界面就是登录和注册界面）
2. 业务层：该新功能需要用到的方法，应该在login\_signup\_tool中体现（该代码实现在了login\_and\_signup\_ui中通过方法实现功能）

3、数据层：该层添加了login\_signup\_data包，用途就是存储、查询和检查需要使用的用户信息。

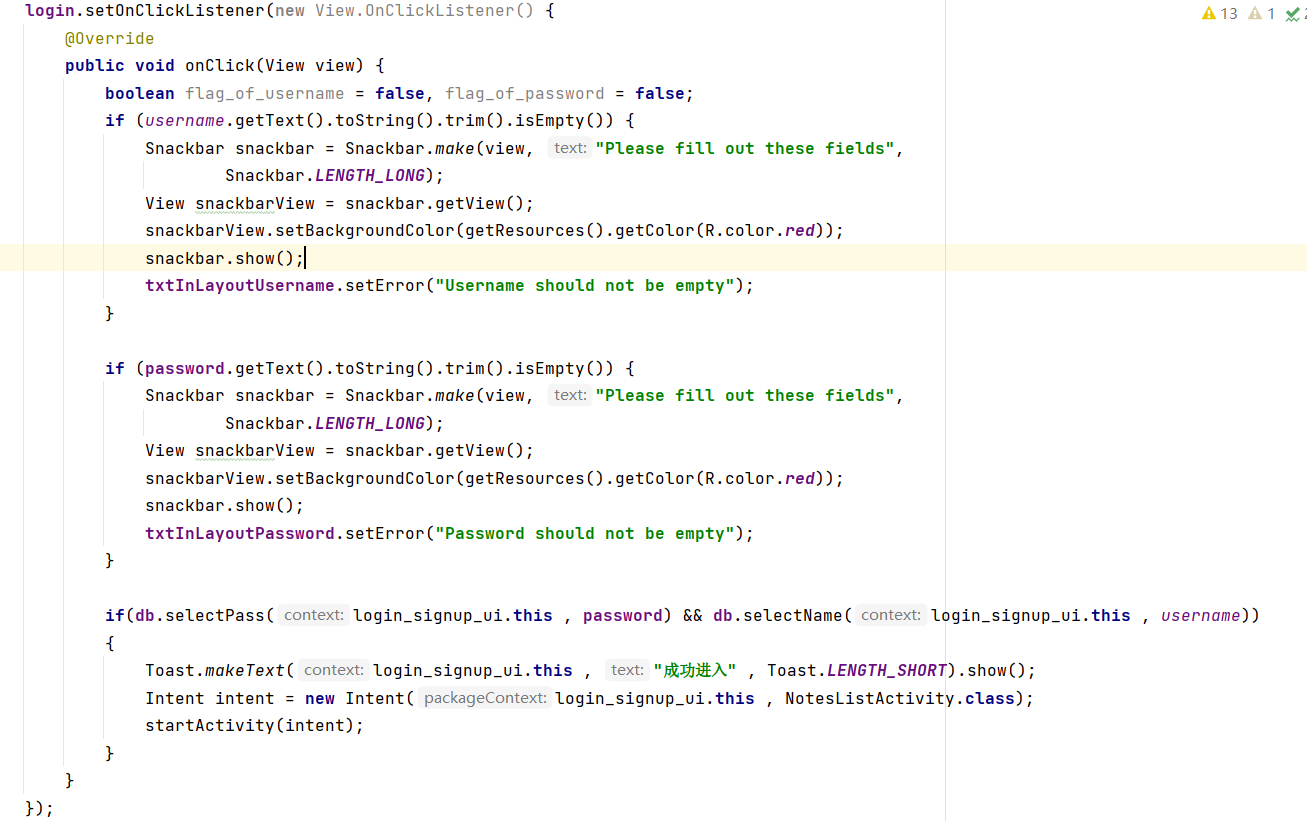
## 3.2 界面设计

在小米便签中的layout文件夹添加两个xml文件分别表示登录的主界面和注册的弹窗界面，如下两张图





界面的跳转login\_signup\_ui中，其中ClickLogin方法中设置按下按钮的监听setOnClickListener，检查输入的内容是否正确。



最后将注册界面设置成主界面即可，更改AndroidManifest的xml文件



## 3.3 详细设计

通过上述的分析，需要实现的类有三个，分别是DBHelper、DBManager和Login\_signup\_ui，可以复用的类是NotesProvider类

1. DBHelper：

分析：使用Create语句进行数据库的创建，创建User表存储用户的信息。

1. package net.micode.notes.data;
2. import android.content.Context;
3. import android.database.sqlite.SQLiteDatabase;
4. import android.database.sqlite.SQLiteOpenHelper;
5. *// 通过SQLiteOpenHelper的子类生成，所以我们需要建一个类来继承SQLiteOpenHelper。*
6. public class DBhelper extends SQLiteOpenHelper
7. {
8. *// 数据库名称*
9. public static final String db\_name = "Users.db";
10. private Context mcontext;
11. *// SQL创建条目*
12. private static final String SQL\_CREATE\_ENTRIES = "create table db\_of\_user(" +
13. "\_user\_name TEXT," +
14. "\_password TEXT," +
15. "\_firstName TEXT," +
16. "\_LastName TEXT," +
17. "\_email TEXT" + ")";
18. *// 由于这个类是一个抽象类，我们需要实现他的构造方法和抽象方法*
19. public DBhelper(Context context) {
20. super(context, db\_name, null, 1);
21. }
22. @Override
23. public void onCreate(SQLiteDatabase db)
24. {
25. db.execSQL(SQL\_CREATE\_ENTRIES);
26. }
27. @Override
28. public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
29. {
30. onUpgrade(db, oldVersion, newVersion);
31. }
32. }
33. DBManager

分析：其中拥有三个方法，以下是具体算法分析

1. Add方法：插入用户信息，创建ContentValues对象使用该对象处理数据
2. SelectPass方法：查询用户输入的密码，其中包括了数据库的query方法（没有定义具体方法实现）
3. SelectName方法：查询用户输入的用户名，其中包括了数据库的query方法

具体算法实现可以阅读代码

1. package net.micode.notes.data;
2. import android.content.ContentValues;
3. import android.content.Context;
4. import android.database.Cursor;
5. import android.database.sqlite.SQLiteDatabase;
6. import android.widget.EditText;
7. public class DBmanager
8. {
9. *// 操作表的对象，进行增删改查*
10. private final SQLiteDatabase db;
11. public DBmanager(Context con) {
12. DBhelper dbHelper = new DBhelper(con);
13. db = dbHelper.getWritableDatabase();
14. }
15. *//    "create table user (" +*
16. *//    "user\_name EditText," +*
17. *//    "password EditText," +*
18. *//    "firstName EditText," +*
19. *//    "LastName EditText," +*
20. *//    "email EditText" + ")"*
21. *// 增加元素*
22. public void add(EditText reg\_username , EditText reg\_password , EditText reg\_firstName ,
23. EditText reg\_lastName , EditText Email)
24. {
25. ContentValues cv = new ContentValues();
26. cv.put("\_user\_name" , reg\_username.getText().toString());
27. cv.put("\_password" , reg\_password.getText().toString());
28. cv.put("\_firstName" , reg\_firstName.getText().toString());
29. cv.put("\_LastName" , reg\_lastName.getText().toString());
30. cv.put("\_email" , Email.getText().toString());
31. db.insert("db\_of\_user" , null , cv);
32. }
33. public boolean selectPass(Context context , EditText reg\_password)
34. {
35. *// Cursor cursor = db.query(*
36. *//    FeedEntry.TABLE\_NAME,   // 表名*
37. *//    projection,             // 要返回的列数组(传递null获取全部)*
38. *//    selection,              // WHERE的列*
39. *//    selectionArgs,          // WHERE子句的值*
40. *//    null,                   // 不要分组a*
41. *//    null,                   // 不要按行组筛选*
42. *//    sortOrder               // 排序顺序*
43. *//    );*
44. Cursor cur = db.query("db\_of\_user" , null , null , null , null , null ,null);
45. int pos = cur.getPosition();
46. boolean flag = false;
47. while(cur.moveToNext())
48. {
49. String pass = cur.getString(1);
50. if(pass.equals(reg\_password.getText().toString().trim()))
51. {
52. flag = true;
53. break;
54. }
55. }
56. cur.close();
57. return flag;
58. }
59. public boolean selectName(Context context , EditText reg\_username)
60. {
61. Cursor cur = db.query("db\_of\_user" , null , null , null , null , null ,null);
62. int pos = cur.getPosition();
63. boolean flag = false;
64. while (cur.moveToNext())
65. {
66. String name = cur.getString(0);
67. System.out.println(name);
68. if (name.equals(reg\_username.getText().toString().trim()))
69. {
70. flag = true;
71. break;
72. }
73. }
74. cur.close();
75. return flag;
76. }
77. }
78. login\_signup\_ui：

分析：其中有三个比较重要的方法，以下是具体算法分析

1. OnCreate：进行最基本的数据库的创建，各种属性的定义，并且调用ClickLogin和注册按钮的监听
2. ClickLogin：当点击登录按钮后，首先需要检查输入参数是否符合要求，然后正确输入后进行用户名和密码的判断，最后定义Intent类进行页面的跳转
3. ClickSignUp：当点击注册按钮后，弹出弹窗，用户输入完数据后，点击返回按钮，系统进行检查，输入正确后，存入数据库

具体算法实现可以阅读代码

1. package net.micode.notes.ui;
2. *// 第一步 存储个人信息*
3. *// 第二步 每一次登录便签首先需要进行密码的输入*
4. import android.app.Activity;
5. import android.content.DialogInterface;
6. import android.content.Intent;
7. import android.database.sqlite.SQLiteDatabase;
8. import android.support.design.widget.Snackbar;
9. import android.support.design.widget.TextInputLayout;
10. import android.support.v7.app.AlertDialog;
11. import android.support.v7.app.AppCompatActivity;
12. import android.os.Bundle;
13. import android.view.LayoutInflater;
14. import android.view.View;
15. import android.widget.Button;
16. import android.widget.CheckBox;
17. import android.widget.EditText;
18. import android.widget.Toast;
19. *// 数据库*
20. import net.micode.notes.data.DBhelper;
21. import net.micode.notes.data.DBmanager;
22. *// R库*
23. import net.micode.notes.R;
24. *// 主界面库*
25. import net.micode.notes.ui.NotesListActivity;
26. import net.micode.notes.ui.\_user;
27. public class login\_signup\_ui extends AppCompatActivity {
28. static EditText username;
29. EditText password;
30. EditText reg\_username;
31. EditText reg\_password;
32. EditText reg\_firstName;
33. EditText reg\_lastName;
34. EditText reg\_email;
35. Button login, signUp, reg\_register;
36. TextInputLayout txtInLayoutUsername, txtInLayoutPassword, txtInLayoutRegPassword;
37. CheckBox rememberMe;
38. \_user us = new \_user();
39. DBmanager db;
40. @Override
41. protected void onCreate(Bundle savedInstanceState) {
42. super.onCreate(savedInstanceState);
43. setContentView(R.layout.activity\_main);
44. *// 创建数据库*
45. DBhelper helper = new DBhelper(login\_signup\_ui.this);
46. db = new DBmanager(login\_signup\_ui.this);
47. username = findViewById(R.id.username);
48. password = findViewById(R.id.password);
49. login = findViewById(R.id.login);
50. signUp = findViewById(R.id.signUp);
51. txtInLayoutUsername = findViewById(R.id.txtInLayoutUsername);
52. txtInLayoutPassword = findViewById(R.id.txtInLayoutPassword);
53. rememberMe = findViewById(R.id.rememberMe);
54. ClickLogin();
55. *// 用于显示注册页面的注册按钮*
56. signUp.setOnClickListener(new View.OnClickListener() {
57. @Override
58. public void onClick(View view) {
59. *// Toast.makeText(MainActivity.this, getBaseContext().toString(), Toast.LENGTH\_SHORT).show();*
60. ClickSignUp();
61. }
62. });
63. }
64. *// 这是执行检查注册操作的方法登录*
65. private void ClickLogin() {
66. login.setOnClickListener(new View.OnClickListener() {
67. @Override
68. public void onClick(View view) {
69. boolean flag\_of\_username = false, flag\_of\_password = false;
70. if (username.getText().toString().trim().isEmpty()) {
71. Snackbar snackbar = Snackbar.make(view, "Please fill out these fields",
72. Snackbar.LENGTH\_LONG);
73. View snackbarView = snackbar.getView();
74. snackbarView.setBackgroundColor(getResources().getColor(R.color.red));
75. snackbar.show();
76. txtInLayoutUsername.setError("Username should not be empty");
77. }
78. if (password.getText().toString().trim().isEmpty()) {
79. Snackbar snackbar = Snackbar.make(view, "Please fill out these fields",
80. Snackbar.LENGTH\_LONG);
81. View snackbarView = snackbar.getView();
82. snackbarView.setBackgroundColor(getResources().getColor(R.color.red));
83. snackbar.show();
84. txtInLayoutPassword.setError("Password should not be empty");
85. }
86. if(db.selectPass(login\_signup\_ui.this , password) && db.selectName(login\_signup\_ui.this , username))
87. {
88. Toast.makeText(login\_signup\_ui.this , "成功进入" , Toast.LENGTH\_SHORT).show();
89. Intent intent = new Intent(login\_signup\_ui.this , NotesListActivity.class);
90. startActivity(intent);
91. }
92. }
93. });
94. }
95. public static String get\_user\_name()
96. {
97. return username.getText().toString().trim();
98. }
99. *// 用于打开注册页面的方法以及用于注册的其他处理或检查*
100. private void ClickSignUp() {
101. *// 先定义 AlertDialog.Builder 父类 builder*
102. *// 使用父类构造 dialog 对话框*
103. AlertDialog.Builder builder = new AlertDialog.Builder(login\_signup\_ui.this);
104. AlertDialog dialog = builder.create();
105. LayoutInflater inflater = getLayoutInflater();
106. View dialogView = inflater.inflate(R.layout.register, null);
107. dialog.setView(dialogView);
108. reg\_username = dialogView.findViewById(R.id.reg\_username);
109. reg\_password = dialogView.findViewById(R.id.reg\_password);
110. reg\_firstName = dialogView.findViewById(R.id.reg\_firstName);
111. reg\_lastName = dialogView.findViewById(R.id.reg\_lastName);
112. reg\_email = dialogView.findViewById(R.id.reg\_email);
113. reg\_register = dialogView.findViewById(R.id.reg\_register);
114. txtInLayoutRegPassword = dialogView.findViewById(R.id.txtInLayoutRegPassword);
115. reg\_register.setOnClickListener(new View.OnClickListener() {
116. @Override
117. public void onClick(View view) {
118. boolean flag = true;
119. *// 获取注册名字*
120. if (reg\_username.getText().toString().trim().isEmpty()) {
121. reg\_username.setError("Please fill out this field");
122. flag = false;
123. } else {
124. *//Here you can write the codes for checking username*
125. us.setUsername(reg\_username);
126. }
127. *// 获得注册密码*
128. if (reg\_password.getText().toString().trim().isEmpty()) {
129. *// 设置密码可见性启用开关*
130. txtInLayoutRegPassword.setPasswordVisibilityToggleEnabled(false);
131. reg\_password.setError("Please fill out this field");
132. flag = false;
133. } else {
134. txtInLayoutRegPassword.setPasswordVisibilityToggleEnabled(true);
135. us.setPassword(reg\_password);
136. }
137. *// 获得名字的姓*
138. if (reg\_firstName.getText().toString().trim().isEmpty()) {
139. reg\_firstName.setError("Please fill out this field");
140. flag = false;
141. } else {
142. *//Here you can write the codes for checking firstname*
143. us.setFirstName(reg\_firstName);
144. }
145. *// 获得名字的名*
146. if (reg\_lastName.getText().toString().trim().isEmpty()) {
147. reg\_lastName.setError("Please fill out this field");
148. flag = false;
149. } else {
150. us.setLastName(reg\_lastName);
151. }
152. *// 获得邮箱*
153. if (reg\_email.getText().toString().trim().isEmpty()) {
154. reg\_email.setError("Please fill out this field");
155. flag = false;
156. } else {
157. us.setEmail(reg\_email);
158. }
159. *// 退出接口*
160. if (flag) {
161. db.add(reg\_username , reg\_password , reg\_firstName , reg\_lastName , reg\_email);
162. dialog.cancel();
163. }
164. *//Toast.makeText(MainActivity.this , "被点击" , Toast.LENGTH\_SHORT).show();*
165. *// moveTaskToBack(true);*
166. *// finish();*
167. }
168. });
169. dialog.show();
170. }
171. }