

中國民航大學

软件工程(I)

小米便签开源代码

质量分析报告



组长学号姓名: 210340146 郭德宁

成员学号姓名: 210340001 陈鹤林

210340143 高源

210340142 段逸群

二〇二三年六月

小米便签开源软件是一款较为流行的备忘录软件，其由 Java 语言编写，基于 Android 操作系统进行开发运行和维护。小米便签开源软件规模适中，通过初步阅读分析其代码质量较高，故对其进行进一步的质量分析和审查。

一、代码质量分析方法

对开源软件代码的质量进行分析旨在理解高质量的软件系统在代码层面和设计层面应具有哪些方面的特征，掌握编写高质量程序代码的方法和技巧，同时发现开源软件代码中尚存的质量问题。对开源软件代码的质量分析可以采用以下二种方法：人工分析和自动分析。

1、人工

利用人工泛读，添加注释，对代码进行体系结构分析、用例图绘制以及不同类型间的关系图描述等方法熟悉整体代码并且在软件实现、测试、维护过程中发现的代码中尚存的质量问题，并给予修改

2、自动

本报告主要使用 sonarQube 工具进行针对小米便签开源软件进行质量分析，辅以少量的人工分析。

二、SonarQube 简介

Sonarqube 软件是一种静态代码检查工具，采用 B/S 架构，帮助检查代码缺陷，改善代码质量，提高开发速度，通过插件形式，可以支持 Java、C、C++JavaScript 等多种编程语言代码质量管理与检测。Sonar 客户端采用 IDE 插件、Sonar-Scanner 插件、ant 插件和 Maven 插件方式，并且通过不同分析机制对项目代码进行分析扫描，并把分析扫描后的结果上传到 Sonar 的数据库，通过 Sonarweb 界面对分析结果进行管理

三、 SonarQube 使用配置说明

(一) 首先前置条件

1. 下载和安装 Java JDK:

SonarQube 需要在 Java 的基础上运行。因此，如果您没有安装 Java JDK，则需要首先下载和安装它。请注意，SonarQube 最低要求 JDK 11 以上版本。

2. 配置数据库:

在开始使用之前，您需要配置一个数据库来存储 SonarQube 数据。常见的选项是 MySQL 或 PostgreSQL。

(二) 安装步骤

1. 下载 SonarQubeServer:

从 SonarQube 官方网站 (<https://www.sonarqube.org/downloads/>) 下载适用于您的操作系统版本，并解压缩文件到您想要安装它的目录中。

2. 配置数据库:

在开始使用之前，您需要配置一个数据库来存储 SonarQube 数据。常见的选项是 MySQL 或 PostgreSQL。可以参考官方文档以了解如何配置数据库。

以 PostgreSQL 数据库为例:

(1) 安装 PostgreSQL

首先，您需要在计算机上安装 PostgreSQL 数据库。您可以从官方网站 下载并安装最新版本的 PostgreSQL。

(2) 创建数据库

接下来，在 PostgreSQL 中创建一个新的数据库用于 SonarQube。您可以使用命令行工具（例如 psql）或 pgAdmin 等图形界面管理工具来完成此操作。以下是使用命令行工具创建名为“sonarqube”的数据库的示例命令:

```
....
```

```
createdb sonarqube
```

```
....
```

(3) 创建用户

接下来，为 SonarQube 创建一个专用用户，并将其分配给新创建的数据库。以下是使用命令行工具创建名为“sonarqube”用户并将其授权访问“sonarqube”数据库的示例命令：

```
....
```

```
createuser sonarqube
```

```
psql sonarqube
```

```
grant all privileges on database sonarqube to sonarqube;
```

```
....
```

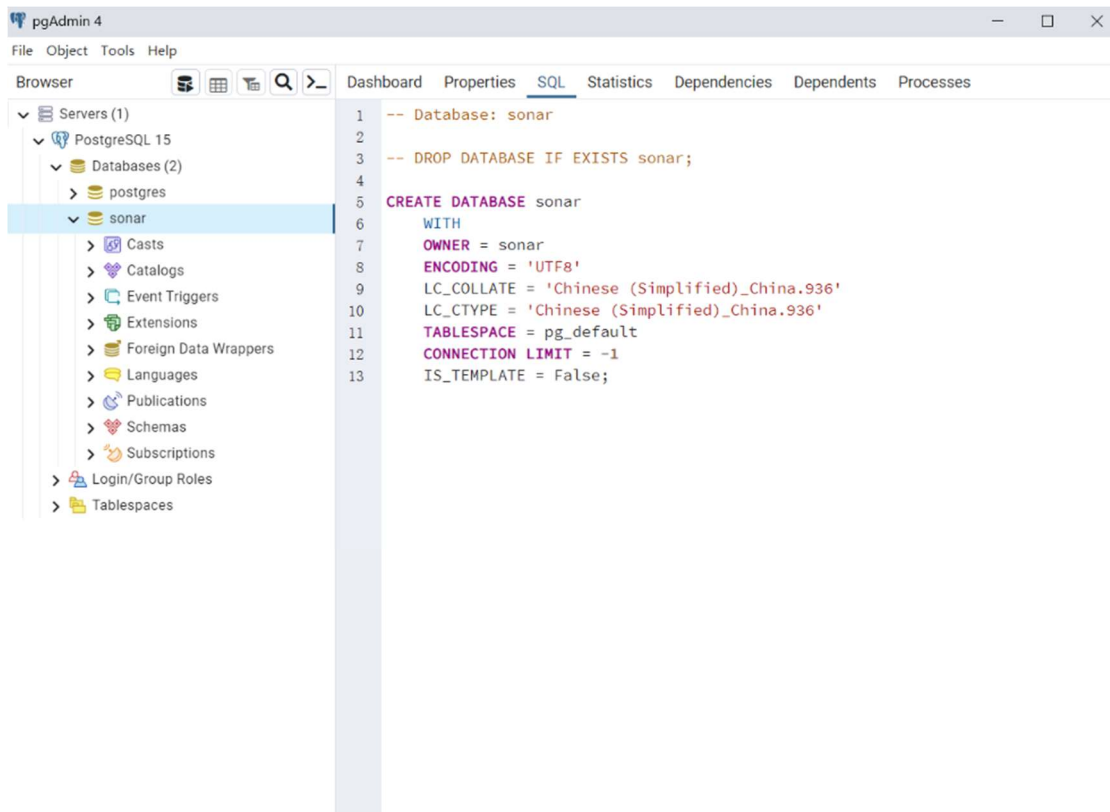


图 1. 用户创建

修改 SonarQube 配置文件

找到 SonarQube 配置文件，在其中找到以下属性并进行修改：

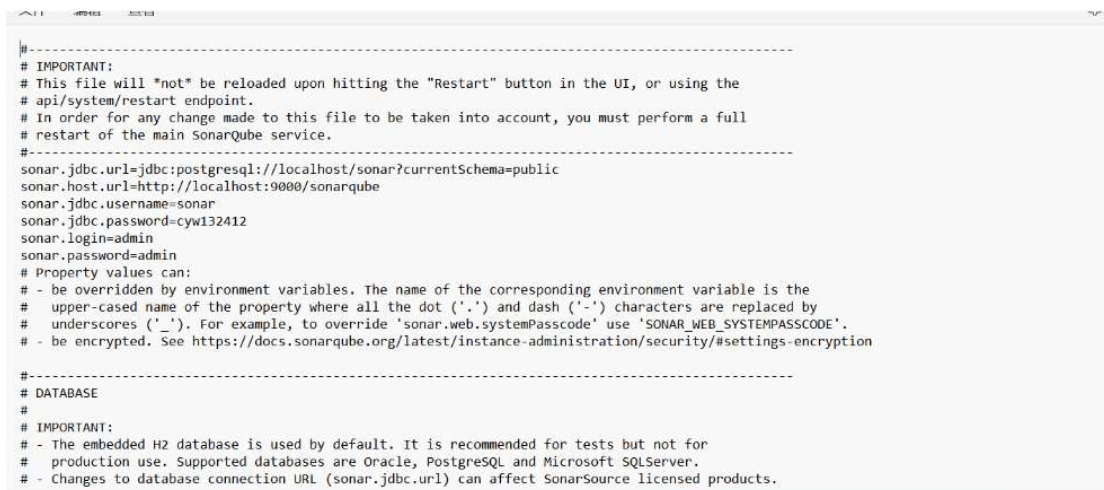
```
# 数据库连接 url
sonar.jdbc.url=jdbc:postgresql://localhost/sonarqube

# 数据库用户名
sonar.jdbc.username=sonarqube

# 数据库密码
sonar.jdbc.password=your_password
...

```

注意：将“your_password”替换为实际的密码。



```
#-----
# IMPORTANT:
# This file will "not" be reloaded upon hitting the "Restart" button in the UI, or using the
# api/system/restart endpoint.
# In order for any change made to this file to be taken into account, you must perform a full
# restart of the main SonarQube service.
#-----
sonar.jdbc.url=jdbc:postgresql://localhost/sonar?currentSchema=public
sonar.host.url=http://localhost:9000/sonarqube
sonar.jdbc.username=sonar
sonar.jdbc.password=cyw132412
sonar.login=admin
sonar.password=admin
# Property values can:
# - be overridden by environment variables. The name of the corresponding environment variable is the
#   upper-cased name of the property where all the dot('.') and dash('-') characters are replaced by
#   underscores('_'). For example, to override 'sonar.web.systemPasscode' use 'SONAR_WEB_SYSTEMPASSCODE'.
# - be encrypted. See https://docs.sonarqube.org/latest/instance-administration/security/#settings-encryption
#-----
# DATABASE
#
# IMPORTANT:
# - The embedded H2 database is used by default. It is recommended for tests but not for
#   production use. Supported databases are Oracle, PostgreSQL and Microsoft SQLServer.
# - Changes to database connection URL (sonar.jdbc.url) can affect SonarSource licensed products.

```

图 2.建立数据库连接

3. 重启 SonarQube:

完成上述步骤后，重启 SonarQube 并验证连接是否成功。

(三) 启动

1. 启动 SonarQube Server:

进入解压后的 bin 目录，并以管理员身份启动 sonar.sh (Linux / macOS) 或 sonar.bat (Windows) 脚本。在启动过程中，控制台将显示程序输出并将等待 SonarQube Server 启动。请注意，这可能需要几分钟的时间。

启动成功实例:

```
C:\WINDOWS\system32\cmd. x + v
.\Java\bin\java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djava.io.tmpdir=D:\sonarqube-9.9.0.65466\temp -XX:-OmitStackTraceInFastThrow --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED --add-exports=java.base/jdk.internal.ref=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.management/sun.management=ALL-UNNAMED --add-opens=jdk.management/com.sun.management.internal=ALL-UNNAMED -Dcom.redhat.fips=false -Xmx512m -Xms128m -XX:+HeapDumpOnOutOfMemoryError -Dhttp.nonProxyHosts=localhost|127.*|[:1] -cp ./Lib/sonar-application-9.9.0.65466.jar;D:\sonarqube-9.9.0.65466\lib\jdbc\postgresql\postgresql-42.5.1.jar org.sonar.server.app.WebServer D:\sonarqube-9.9.0.65466\temp\sq-process9635270062635296325properties
WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System:setSecurityManager has been called by org.sonar.process.PluginSecurityManager (file:/D:/sonarqube-9.9.0.65466/lib/sonar-application-9.9.0.65466.jar)
WARNING: Please consider reporting this to the maintainers of org.sonar.process.PluginSecurityManager
WARNING: System:setSecurityManager will be removed in a future release
2023.03.28 21:59:36 INFO app[[o.s.a.SchedulerImpl] Process[web] is up
2023.03.28 21:59:36 INFO app[[o.s.a.ProcessLauncherImpl] Launch process[COMPUTE_ENGINE] from [D:\sonarqube-9.9.0.65466]; D:\Java\bin\java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djava.io.tmpdir=D:\sonarqube-9.9.0.65466\temp -XX:-OmitStackTraceInFastThrow --add-opens=java.base/java.util=ALL-UNNAMED --add-exports=java.base/jdk.internal.ref=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.management/sun.management=ALL-UNNAMED --add-opens=jdk.management/com.sun.management.internal=ALL-UNNAMED -Dcom.redhat.fips=false -Xmx512m -Xms128m -XX:+HeapDumpOnOutOfMemoryError -Dhttp.nonProxyHosts=localhost|127.*|[:1] -cp ./Lib/sonar-application-9.9.0.65466.jar;D:\sonarqube-9.9.0.65466\lib\jdbc\postgresql\postgresql-42.5.1.jar org.sonar.ce.app.CeServer D:\sonarqube-9.9.0.65466\temp\sq-process1425930388788318713properties
WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System:setSecurityManager has been called by org.sonar.process.PluginSecurityManager (file:/D:/sonarqube-9.9.0.65466/lib/sonar-application-9.9.0.65466.jar)
WARNING: Please consider reporting this to the maintainers of org.sonar.process.PluginSecurityManager
WARNING: System:setSecurityManager will be removed in a future release
2023.03.28 21:59:40 INFO app[[o.s.a.SchedulerImpl] Process[ce] is up
2023.03.28 21:59:40 INFO app[[o.s.a.SchedulerImpl] SonarQube is operational
```

图 3.成功启动实例

2. 路径配置

如图，文件夹目录添加至路径

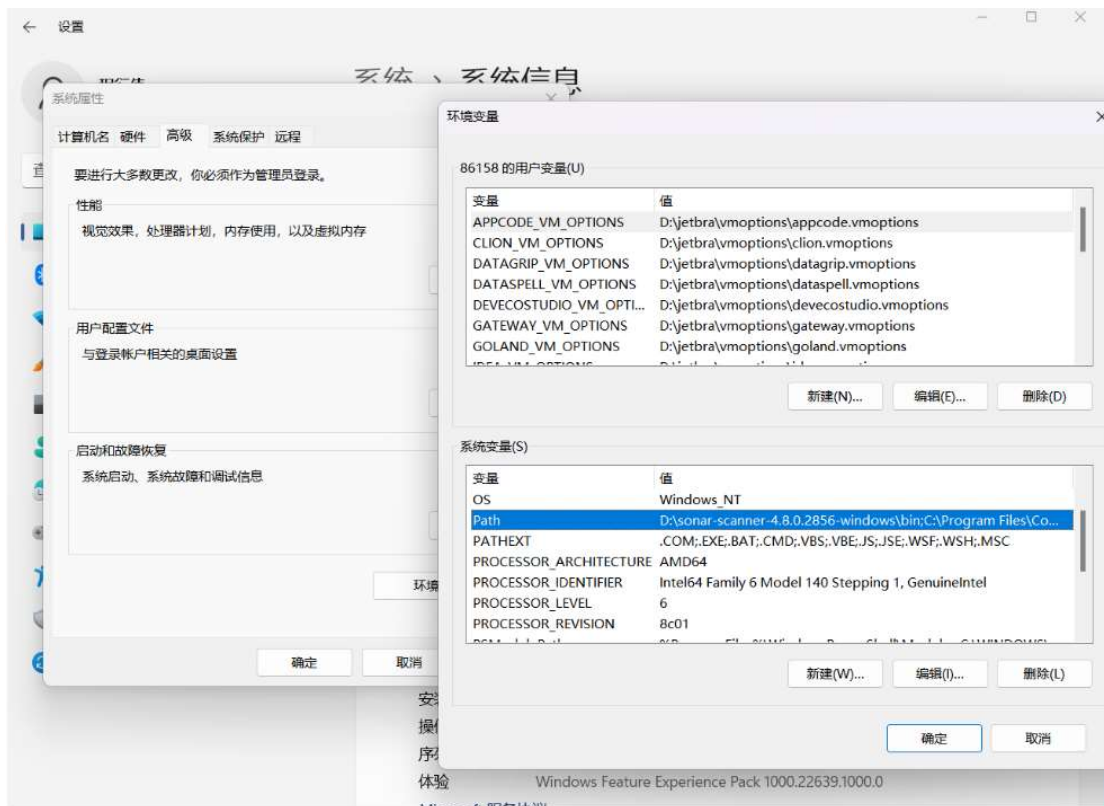


图 4. sonarqube 路径配置

四、通过人工方式分析小米便签的质量情况

通过对“小米便签”开源软件代码的阅读和分析，可以发现“小米便签”程序代码蕴含着诸多值得学习的编程方法和技巧、编写高质量代码的规范和经验等。如“modle”中方法“note”的部分程序代码，可以发现该代码具有以下优点：

- 第 355 行：

```
355  @ private boolean inRangeOfView(View view, MotionEvent ev) {
356      int []location = new int[2];
357      view.getLocationOnScreen(location);
358      int x = location[0];
359      int y = location[1];
360      if (ev.getX() < x
361          || ev.getX() > (x + view.getWidth())
362          || ev.getY() < y
363          || ev.getY() > (y + view.getHeight())) {
364          return false; //如果触控的位置超出了给定的范围, 返回false
365      }
366      return true;
367  }
```

图 5.触控范围控制判断语句

这里巧妙地使用了一个 if 判断语句来控制触控位置范围，因为判断条件较多，这里大部分时候会想到使用 switch-case 来进行判断，但这里使用 if 语句在保证正确率的同时减少多余代码段，保证了灵活性和便于维护性

- 第 431 行：

```

431 @ public void onClick(View v) {
432     int id = v.getId();
433     if (id == R.id.btn_set_bg_color) {
434         mNoteBgColorSelector.setVisibility(View.VISIBLE);
435         findViewById(sBgSelectorSelectionMap.get(mWorkingNote.getBgColorId())).setVisibility(
436             View.VISIBLE);
437     } else if (sBgSelectorBtnsMap.containsKey(id)) {
438         findViewById(sBgSelectorSelectionMap.get(mWorkingNote.getBgColorId())).setVisibility(
439             View.GONE);
440         mWorkingNote.setBgColorId(sBgSelectorBtnsMap.get(id));
441         mNoteBgColorSelector.setVisibility(View.GONE);
442     } else if (sFontSizeBtnsMap.containsKey(id)) {
443         findViewById(sFontSelectorSelectionMap.get(mFontSizeId)).setVisibility(View.GONE);
444         mFontSizeId = sFontSizeBtnsMap.get(id);
445         mSharedPreferences.edit().putInt(PREFERENCE_FONT_SIZE, mFontSizeId).commit();
446         findViewById(sFontSelectorSelectionMap.get(mFontSizeId)).setVisibility(View.VISIBLE);
447         if (mWorkingNote.getCheckListMode() == TextNote.MODE_CHECK_LIST) {
448             getWorkingText();
449             switchToListMode(mWorkingNote.getContent());
450         } else {
451             mNoteEditor.setTextAppearance(context: this,
452                 TextAppearanceResources.getTexAppearanceResource(mFontSizeId));
453         }
454         mFontSizeSelector.setVisibility(View.GONE);
455     }
456 }

```

图 6.用户储存设置

代码结构清晰，采用了 if-else if 语句，易于理解和维护。使用了常量和映射表，增加了代码的可读性和可维护性。使用了 View 的 setVisibility 方法，实现了控件的显示和隐藏，提高了用户体验。

使用了 SharedPreferences 来存储用户设置，保证了用户下次打开应用时能够看到之前的设置。

- 第 113 行:


```

private View mHeaderViewPanel;
//私有化一个界面操作mHeaderViewPanel, 对表头的操作
8 usages
private View mNoteBgColorSelector;
//私有化一个界面操作mNoteBgColorSelector, 对背景颜色的操
8 usages
private View mFontSizeSelector;
//私有化一个界面操作mFontSizeSelector, 对标签字体的操作
11 usages
private EditText mNoteEditor;
//声明编辑控件, 对文本操作
3 usages
private View mNoteEditorPanel;
//私有化一个界面操作mNoteEditorPanel, 文本编辑的控制板
//private WorkingNote mWorkingNote;
59 usages
public WorkingNote mWorkingNote;
//对模板WorkingNote的初始化
3 usages
private SharedPreferences mSharedPrefs;
//私有化SharedPreferences的数据存储方式
//它的本质是基于XML文件存储key-value键值对数据
11 usages
private int mFontSizeId;
//用于操作字体的大小

```

图 7.控制界面功能标签的定义

该段代码分别私有化表头、背景颜色、标签字体等功能标签。尽管内容繁杂，但保证了后续代码的稳定性并且减少了出错率

- 第 487 行：

```
487 public boolean onPrepareOptionsMenu(Menu menu) {
488     if (isFinishing()) {
489         return true;
490     }
491     clearSettingState();
492     menu.clear();
493     if (mWorkingNote.getFolderId() == Notes.ID_CALL_RECORD_FOLDER) {
494         getMenuInflater().inflate(R.menu.call_note_edit, menu);
495     } else {
496         getMenuInflater().inflate(R.menu.note_edit, menu);
497     }
498     if (mWorkingNote.getCheckListMode() == TextNote.MODE_CHECK_LIST) {
499         menu.findItem(R.id.menu_list_mode).setTitle("Leave check list");
500     } else {
501         menu.findItem(R.id.menu_list_mode).setTitle("Enter check list");
502     }
503     if (mWorkingNote.hasClockAlert()) {
504         menu.findItem(R.id.menu_alert).setVisible(false);
505     } else {
506         menu.findItem(R.id.menu_delete_remind).setVisible(false);
507     }
508     return true;
509 }
```

图 8.菜单项设置

做了 `isFinishing()` 的判断，避免在 Activity 已经结束时还执行操作，增加了代码的健壮性，同时 `clearSettingState()` 清除了之前可能存在的设置状态，避免了状态混乱；`menu.clear()` 清除了之前可能存在的菜单项，避免了菜单项重复。根据 `mWorkingNote` 的状态选择不同的菜单项，提高了代码的可读性和用户体验。

但是从质量的视角，该开源软件代码尚存在一些美中不足之处，具体表现为：

- 第 697 行：

```
697 private void switchToListMode(String text) {
698     mEditTextList.removeAllViews();
699     String[] items = text.split(regex: "\\n");
700     int index = 0;
701     for (String item : items) {
702         if (!TextUtils.isEmpty(item)) {
703             mEditTextList.addView(getListItem(item, index));
704             index++;
705         }
706     }
707     mEditTextList.addView(getListItem(item: "", index));
708     mEditTextList.getChildAt(index).findViewById(R.id.et_edit_text).requestFocus();
709
710     mNoteEditor.setVisibility(View.GONE);
711     mEditTextList.setVisibility(View.VISIBLE);
712 }
```

图 9.质量问题段代码分析

没有对输入的文本进行校验，如果输入的文本包含特殊字符或者非法字符会

导致程序常；

没有对添加的子 View 进行重复性校验，如果重复添加会导致界面错乱。没有对子 View 的数量进行限制，如果添加的子 View 数量过多会导致界面卡顿或者崩溃。没有对布局进行优化，可能会导致界面显示异常或者出现性能问题。

五、使用 SonarQube 进行自动分析

一般地，根据严重程度的差异性，可以将程序代码中的缺陷和问题分为以下几个不同的等级。

- Blocker: 严重程度最高，极有可能造成系统和应用程序崩溃和功能丧失，比如死循环问题。
- Critical: 严重程度较高，可能存在影响程序运行的错误或者安全缺陷。
- Major: 严重程度一般，如存在部分次要功能没有完全实现。
- Minor: 严重程度较低，在一定程度上给用户带来不便。
- Info: 严重程度最低，大多为代码质量分析软件提出的一些改进和建议。

在 Android Studio 中使用 SonarQube 对所加载程序进行分析。通过分析，SonarQube 工具反馈有关“小米便签”开源软件代码的质量报告如下：

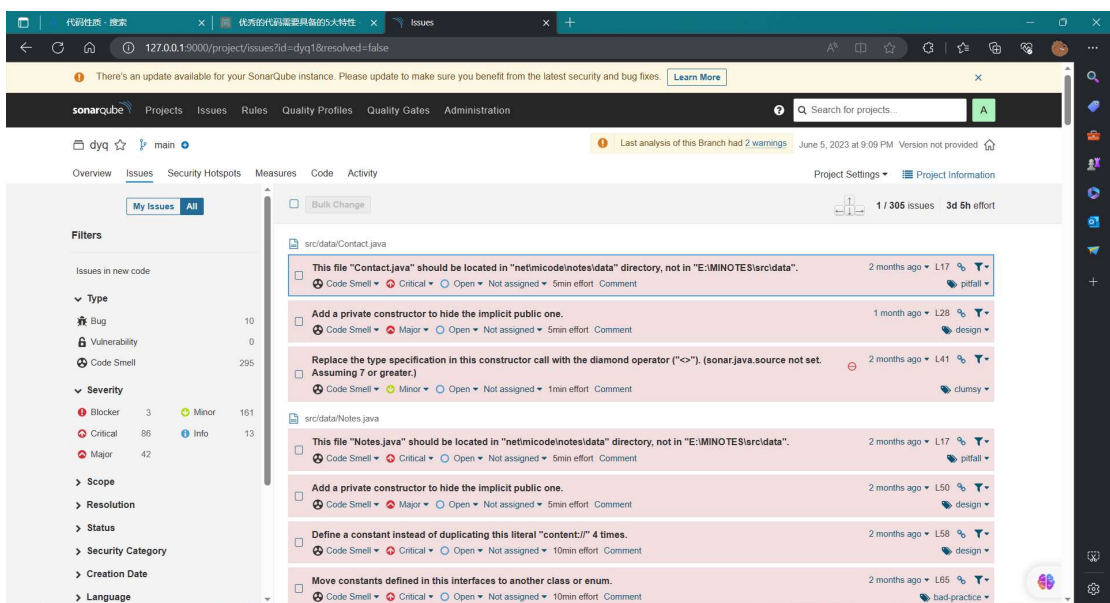


图 10. SonarQube 分析结果

在小米便签软件的源代码中，共审查出 17 个文件中的 131 个问题，其中，Blocker 严重程度问题共有 3 个；Critical 严重程度问题共有 86 个；Major 严重程度问题共有 42 个。

现对代码存在的质量问题进行汇总统计，对严重程度为 Blocker、Critical 和 Major 的问题进行逐一的分析：

1. 严重程度：Blocker

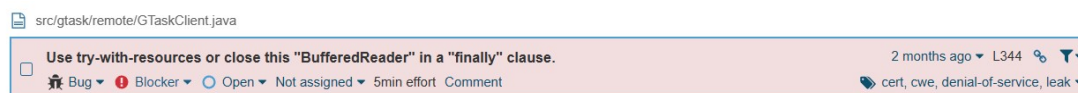


图 11.自动分析结果-Blocker 类型

文件目录：src/gtask/remote/GTaskClient.java 位置：309-324 行

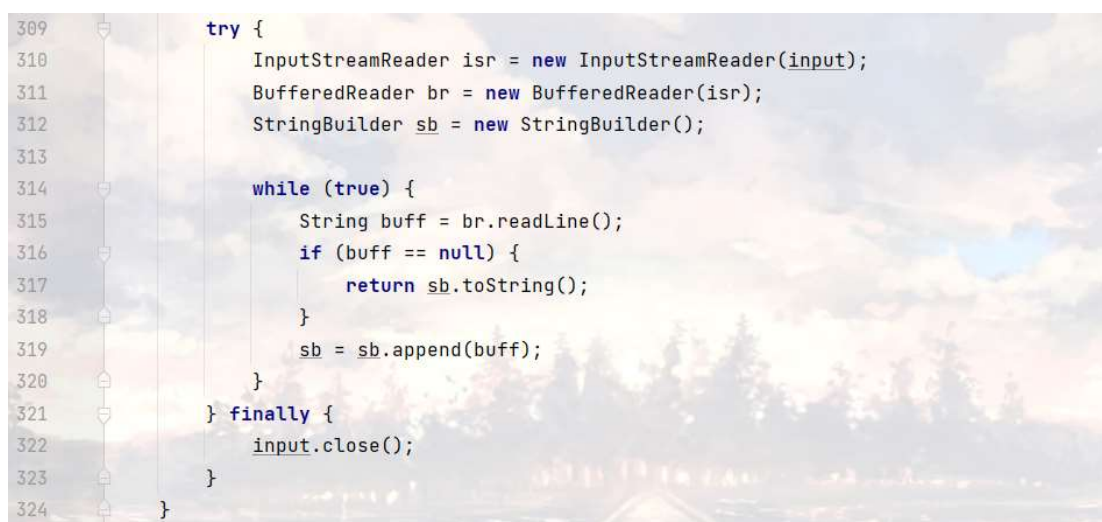


图 12.Blocker 程度代码

具体问题：未能正确关闭资源将导致资源泄漏，这可能会首先使应用程序，然后可能是应用程序陷入困境。

实现 Closeable 接口或其超级接口 AutoCloseable 的连接、流、文件和其他类在使用后需要关闭。此外，必须在 finally 块中进行该关闭调用，否则异常可能会阻止调用。最好，当类实现 AutoCloseable 时，应使用“试用资源”模式创建资源，并将自动关闭。

2. 严重程度：Critical

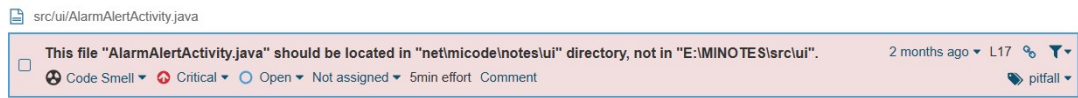


图 13.自动分析结果-Critical 类型

目录: src/ui/AlarmAlertActivity.java 位置: 17 行

```
package net.micode.notes.ui;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
```

图 14.Critical 程度代码

具体问题: 放置在名称中带有包名而不是具有等效文件夹结构的文件夹中的源代码将编译,但在运行时会导致问题。例如,将编译放置在 org/foo.bar 中的包声明为 org.foo.bar 的代码,但类加载器将始终根据包结构将类搜索到文件夹中,因此期望源位于 org/foo/bar 文件夹中。因此,foo.bar 不是源的正确文件夹名称。

3. 严重程度: Major



图 15.自动分析结果-Major 类型

目录: src/model/WorkingNote.java 位置: 190-199 行

```
190 public synchronized boolean saveNote() {
191     if (isWorthSaving()) {
192         if (!existInDatabase()) {
193             if ((mNoteId = Note.getNewNoteId(mContext, mFolderId)) == 0) {
194                 Log.e(TAG, msg: "Create new note fail with id:" + mNoteId);
195                 return false;
196             }
197         }
198     }
199     mNote.syncNote(mContext, mNoteId);
```

图 16. Major 程度代码

具体问题：合并可折叠的 if 语句可提高代码的可读性。建议修改方法如下：

```
if (file != null) {  
    if (file.isFile() || file.isDirectory()) {  
        /* ... */  
    }  
}
```

图 17. 修改前

改为：

```
if (file != null && isFileOrDirectory(file)) {  
    /* ... */  
}  
  
private static boolean isFileOrDirectory(File file) {  
    return file.isFile() || file.isDirectory();  
}
```

图 18. 修改后