小米便签报告

环境搭建

配置项目环境

- JDK 19
- Pixel 5 API34

解决初始运行项目时的报错

- 解决Gradle网络问题：

  Connect Time Out 修改DNS。
- 解决switch case 报错 需要常量表达式

  把switch改成if语句。
- Android Studio启动AVD报错：The emulator process for AVD Pixel_5_API_30 has terminated.

  原因是安装时使用自定义安装后，修改了默认安装目录。只需要修改.android文件夹位置，并修改环境变量。
- 解决菜单不显示的问题

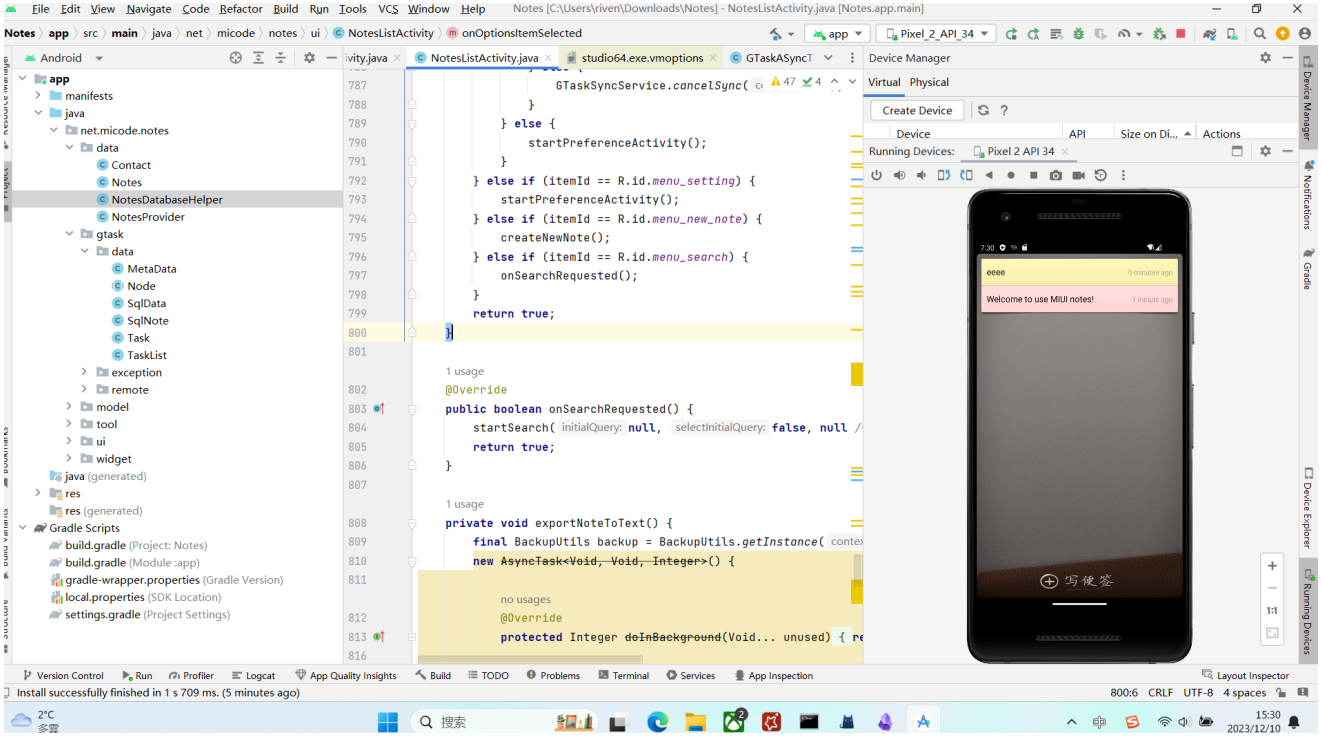
  有一个大坑是菜单默认是隐藏的，需要修改res/values/styles.xml中的两行代码才能开启

```
 <style name="NoteActionBarStyle"
parent="@android:style/Widget.Holo.Light.ActionBar.Solid">
        <item name="android:displayOptions" />
        <item name="android:visibility">gone</item>
    </style>
```

修改为：

```
 <style name="NoteActionBarStyle"
parent="@android:style/Widget.Holo.Light.ActionBar.Solid">
<!--        <item name="android:displayOptions" />-->
        <item name="android:visibility">visible</item>
    </style>
```

成功运行：


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Notes [C:\Users\riven\Downloads\Notes] - NotesListActivity.java [Notes.app.main]
Notes app src main java net micode notes ui NotesListActivity onOptionsItemSelected
Android
app
manifests
java
net.micode.notes
data
Contact
Notes
NotesDatabaseHelper
NotesProvider
gtask
data
MetaData
Node
SqlData
SqlNote
Task
TaskList
exception
remote
model
tool
ui
widget
java (generated)
res
res (generated)
Gradle Scripts
build.gradle (Project: Notes)
build.gradle (Module :app)
gradle-wrapper.properties (Gradle Version)
local.properties (SDK Location)
settings.gradle (Project Settings)
NotesListActivity.java
studio64.exe.vmoptions
GTaskASyncT
GTaskSyncService.cancelSync(
startPreferenceActivity();
} else if (itemId == R.id.menu_setting) {
startPreferenceActivity();
} else if (itemId == R.id.menu_new_note) {
createNewNote();
} else if (itemId == R.id.menu_search) {
onSearchRequested();
return true;
1 usage
@Override
public boolean onSearchRequested() {
startSearch(initialQuery: null, selectInitialQuery: false, null /
return true;
1 usage
private void exportNoteToText() {
final BackupUtils backup = BackupUtils.getInstance(
new AsyncTask<Void, Void, Integer>() {
no usages
@Override
protected Integer doInBackground(Void... unused) {
Device Manager
Virtual Physical
Create Device
Device API Size on Di... Actions
Running Devices: Pixel 2 API 34
eeee
Welcome to use MIUI notes!
写便签
Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection Layout Inspector
Install successfully finished in 1 s 709 ms. (5 minutes ago)
800:6 CRLF UTF-8 4 spaces
2°C 多雲
搜索
15:30 2023/12/10

源码功能整理

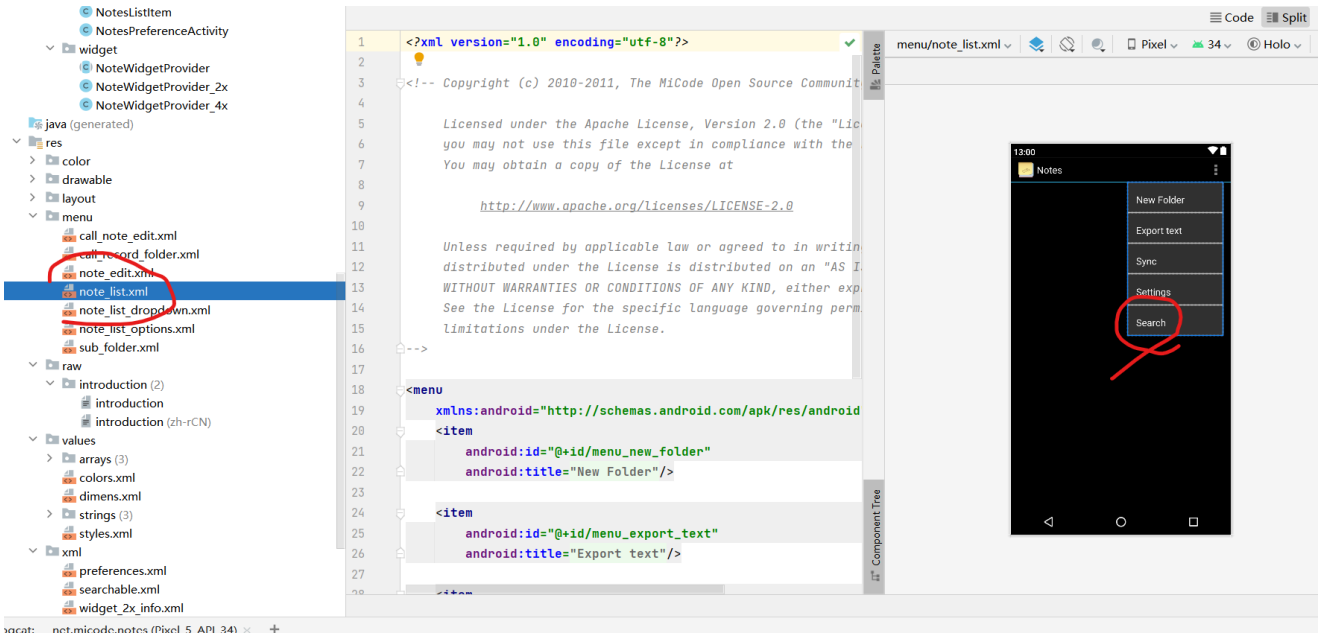
包	子包	类	主要作用
data		contact	联系人数据库
		Notes	便签数据库，用于记录便签相关属性和数据
		NotesDatabaseHelper	数据库帮助类，用于辅助创建、处理数据库的条目
		NotesProvider	便签信息提供类
Gtask	data	MetaData	关于同步任务的元数据
		Node	同步任务的管理结点，用「•设置、保存同步动作的信息
		SqlData	数据麻中基本数据，方法包括读取数据、狭取数据库中数据、提交数据到数据库
		SqINodc	数据库中便签数据，方法包括读取便签内容、从数据库中获取便签数据、设置便签内容、提交便签到数据库
		Task	同步任务，将创建、史新和同步动作包装成JSON对象，用本地和远程的JSON对结点内容进行设 a获取同步信息，进行本地和远程的同步
		TaskList	同步任务列表,将Task组织成同步任务列表进 行管理
	exceptio n	ActionF a ilureExc cption	动作失败异常
		NetworkFailureExceptio n	网络失败异常
	remote	GTaskASyncTask	GTask异步任务，方法包括任务同步和取消，显 示同步任务的进程、通知和结果
		GTaskClient	GTask客户端，提供登录Google账户,创建任务 和任务列表，添加和删除结点，提交、重置更新 更新，获取任务列表等功能
		GTaskManager	GTask管理者，提供同步本地和远端的任务，初 始化任务列表，同步内容、文件夹，添加、更新 本地和远端结点，刷新本地同步任务ID等功能
		GTaskSync Service	GTask同步服务,用丁提供同步服务（开始、取 消同步），发送广播
model		Note	单个便签项
		WorkiiigNote	当前活动便签项
tool		BackupUtils	备份工具类，用于数据备份读取、显示
		Data Utils	便签数据处理工具类，封装如查找、移动、删除数据等操作
		GTaskStriiigUtils	同步中使用的字符出工具类，为jsonObject提
			供 string 对象
		ResourceParser	界面元素的解析工具类，利用 R. java 这个类获取资源供程序调用

ui		AlarmAlertActivity	闹铃提醒界面
		AlarmInitReceiver	闹铃提醒启动消息接收器
		AlarmReceiver	闹铃提醒接收器
		DateTimePicker	设置提醒时间的部件
		DateTimePickerDialog	设置提醒时间的对话框界面
		DropdownMenu	下拉菜单界面
		FoldersListAdapter	文件夹列表链接器（链接数据库）
		NoteEditActivity	便签编辑活动
		NoteEditText	便签的文本编辑界面
		NoteItemData	便签项数据
		NotesListActivity	主界面，实现处理文件夹列表的活动
		NotesListAdapter	便签列表链接器（链接数据库）
		NotesListItem	便签列表项
		NotesPreferenceActivity	便签同步的设置界面
widget		NoteWidgetProvider	桌面挂件
		NoteWidgetProvider_2x	2 倍大小的桌面挂件
		NoteWidgetProvider_4x	4 倍大小的桌面挂件

功能1 搜索功能

源码中的菜单栏有搜索功能"search"，点击后可以弹出搜索栏但是没有反应，这里功能没有做完。

查看note_list.xml文件可以看到绑定的search 按钮对应的代码。



这里我们做了一个修改，把处理android.intent.action.SEARCH的Activity从NoteEditActivity改成了NoteListActivity,并且重写和新增了函数：

```
@Override
protected void onNewIntent(Intent intent) {
```

```
    super.onNewIntent(intent);
    //System.out.println("这里进入NoteListActivity的onNewIntent");
    if (Intent.ACTION_SEARCH.equals(intent.getAction())) {
        String query = intent.getStringExtra(SearchManager.QUERY);
        performSearchQuery(query);
    }
}

private void performSearchQuery(String query) {
    //System.out.println("这里进入NoteListActivity的performSearchQuery");
    String selection = NoteColumns.SNIPPET + " LIKE ?";
    String[] selectionArgs = new String[] { "%" + query + "%" };

    mBackgroundQueryHandler.startQuery(FOLDER_NOTE_LIST_QUERY_TOKEN, null,
            Notes.CONTENT_NOTE_URI, NoteItemData.PROJECTION, selection,
selectionArgs,
            NoteColumns.TYPE + " DESC," + NoteColumns.MODIFIED_DATE + " DESC");
}
```

其中onNewIntent在调用onSearchRequested方法创建搜索动作的Intent后会被触发，并通过intent.getStringExtra()方法获取用户输入的搜索关键字，并调用第二个新创建的方法performSearchQuery()做具体的查询，获取搜索后的便签列表。

功能2 图片添加功能

主要思路：

添加一个插入图片按钮，从相册中选取图片插入到便签中，在保存时保存为[local] 图片路径[\local]的形式，当打开一个便签时，解析其中[local] 图片路径 [\local]的字符串，并将其替换为图片路径指向的图片。

添加图片按钮的 xml 代码(FilePath: ***MiNotes\app\src\main\res\layout\note_edit.xml***)

使得便签编辑页面的底部有一个添加图片按钮，这里我们设置在便签编辑页面的左下角。

```
<ImageButton
        android:id="@+id/add_img_btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="7dp"
        android:layout_marginTop="600dp"
        android:layout_marginBottom="7dp"
        android:src="@android:drawable/ic_menu_gallery" />
```

主要修改NoteEditActivity类，

新增成员变量：

1. `private final int PHOTO_REQUEST = 1;` - 用于标识图片请求的请求码。
2. `private boolean contain_picture = false;` - 指示便签是否包含图片。

新增/改动方法：

1. `onCreate(Bundle savedInstanceState)` - 初始化活动，包括设置图片按钮的监听器。
2. `convertToImage()` - 将文本中的图片路径转换为图像。
3. `replaceTextWithImage(Editable editable, int start, int end, Bitmap bitmap)` -在编辑文本中替换特定文本为图片。
4. `onActivityResult(int requestCode, int resultCode, Intent intent)` - 仅在插入图片时，处理从图片选择器返回的结果。
5. `insertImageIntoEditText(Bitmap bitmap, String uriString)` - 将图片插入到编辑框中。

核心代码：

```
//路径字符串格式 转换为 图片image格式
private void convertToImage() {
    NoteEditText noteEditText = (NoteEditText) findViewById(R.id.note_edit_view);
    //读取出编辑页面的文本
    Editable editable = noteEditText.getText();
    //匹配出标记图片的路径
    Pattern pattern = Pattern.compile("\\[local\\](.*?)\\[/local\\]");
    Matcher matcher = pattern.matcher(editable);

    //加载出每个路径对应的图片
    while (matcher.find()) {

        String uriString = matcher.group(1);
        Uri imageUri = Uri.parse(uriString);
        try {
            //加载图片的同时标记这个便签是包含图片的，方便后续处理
            contain_picture = true;
            Bitmap bitmap =
BitmapFactory.decodeStream(getContentResolver().openInputStream(imageUri));
            //将标记的字符串替换为图片显示出来
            replaceTextWithImage(editable, matcher.start(), matcher.end(),
bitmap);
        } catch (FileNotFoundException e) {
            Log.d(TAG, "convertToImage: File not found for URI", e);
        }
    }
}
```

功能3 加密

主要思路：

添加加密按钮，按下后弹出弹窗给用户输入密码，之后使用密码生成密钥，并通过AES加密便签的文本内容，同时记录通过SHA-256计算得出的密码的哈希值，之后每当用户打开加密过的便签时，弹出弹窗要求输入密码，将用户输入的密码通过SHA-256计算出哈希值，与数据库中读出的哈希值对比可判断密码正确与否，如果正确就使用密码生成密钥并解密密文，显示明文在页面上。

数据库新增字段：

1. is_encrypted： 判断是否加密过。
2. passwordHash： 密码计算出的哈希值，不直接保存密码而保存哈希值，即不影响验证密码，又能在泄露的情况下保证密文不被解密。

新增成员变量：

1. `private static final byte[] SALT`：用于密码加密。
2. `private static final byte[] INIT_VECTOR`：初始化向量（Initialization Vector），用于密码加密中的AES算法。
3. `private String currentPassword`：用于存储当前便签加密时使用的密码。即当用户进入一个加密过的便签作出改动并退出时，便签的密文应该重新计算，所以我们可以在一开始输入密码进入便签时便记录这个密码，在退出便签时通过这个密码加密计算。

新增/改动方法：

1. `private void onEncryptClick(View view)`：加密按钮的点击事件处理方法。当用户点击加密按钮时调用，显示一个输入密码的对话框。
2. `private void encryptNoteContent(String password)`：加密便签内容的方法。使用用户输入的密码加密当前便签的内容。
3. `private byte[] generateKey(String password)`：根据提供的密码生成加密密钥。使用PBKDF2算法和SALT生成。
4. `private void decryptNoteContent(String password)`：解密便签内容的方法。使用用户输入的密码尝试解密当前便签的内容。
5. `private void promptForDecryption()`：弹出解密对话框的方法。如果便签已加密，要求用户输入密码以解密。
6. `private boolean verifyPassword(String password)`：验证解密密码的方法。检查用户输入的密码是否正确。
7. `private boolean checkNoteEncrypted(long noteId)`：检查便签是否加密的方法。根据便签ID检查便签是否已加密。
8. `private void updateUIPostDecryption()`：解密后更新UI的方法。解密便签内容后更新便签编辑界面。

核心代码：

```
private void encryptNoteContent(String password) {
    try {
        String noteContent = mNoteEditor.getText().toString();

        if (noteContent == null || noteContent.isEmpty()) {
            Toast.makeText(this, "便签内容为空，无法加密",
Toast.LENGTH_SHORT).show();
            return;
        }
        //生成密钥
        byte[] key = generateKey(password);
        //生成密文
        byte[] encryptedContent = EncryptionHelper.encrypt(key, INIT_VECTOR,
noteContent.getBytes());
        //将密文转为base64编码，以方便作为字符串格式存在数据库中
        String encryptedContentBase64 = Base64.encodeToString(encryptedContent,
Base64.DEFAULT);

        mWorkingNote.setWorkingText(encryptedContentBase64);
        //计算密码的哈希值
        MessageDigest digest = MessageDigest.getInstance("SHA-256");
        byte[] hash = digest.digest(password.getBytes(StandardCharsets.UTF_8));
        String passwordHash = Base64.encodeToString(hash, Base64.DEFAULT);
        mWorkingNote.setEncryption(true, passwordHash);
        //记录当前的密码，方便修改便签并退出后重新进行加密
        currentPassword = password;

    } catch (Exception e) {
        e.printStackTrace();
        Toast.makeText(this, "加密失败: " + e.getMessage(),
Toast.LENGTH_SHORT).show();
    }
}
```