ReadMe

[完成任务 1](#_Toc32053)

[效果展示 2](#_Toc17977)

[时间戳+UI 2](#_Toc22598)

[搜索功能 2](#_Toc28480)

[导出笔记 2](#_Toc17824)

[改变字体颜色 3](#_Toc10375)

[具体实现 3](#_Toc26089)

[一、时间戳显示 3](#_Toc29422)

[1.noteslist\_item.xml 3](#_Toc22032)

[2.NotesList.java 4](#_Toc16649)

[二、摘要内容展示 6](#_Toc27184)

[1.noteslist\_item.xml中 6](#_Toc31482)

[2.NoteLists.java 8](#_Toc1162)

[3.效果展示 9](#_Toc20991)

[三、搜索功能 10](#_Toc4454)

[1.list\_options\_menu.xml 10](#_Toc26963)

[2.NotesList.java 10](#_Toc14036)

[3.效果展示 13](#_Toc18276)

[四、UI美化 14](#_Toc32682)

[1.color.xml 14](#_Toc22889)

[2. noteslist\_item.xml 14](#_Toc17196)

[五、导出笔记 15](#_Toc25809)

[1.editor\_options\_menu.xml 15](#_Toc22341)

[2.NoteEditor.java 15](#_Toc23860)

[3.效果展示 18](#_Toc18745)

[六、 改变字体颜色 19](#_Toc2719)

[1. editor\_options\_menu.xml 19](#_Toc31034)

[2. NotePadProvider.java 19](#_Toc2935)

[3. NoteEditor.java 20](#_Toc5954)

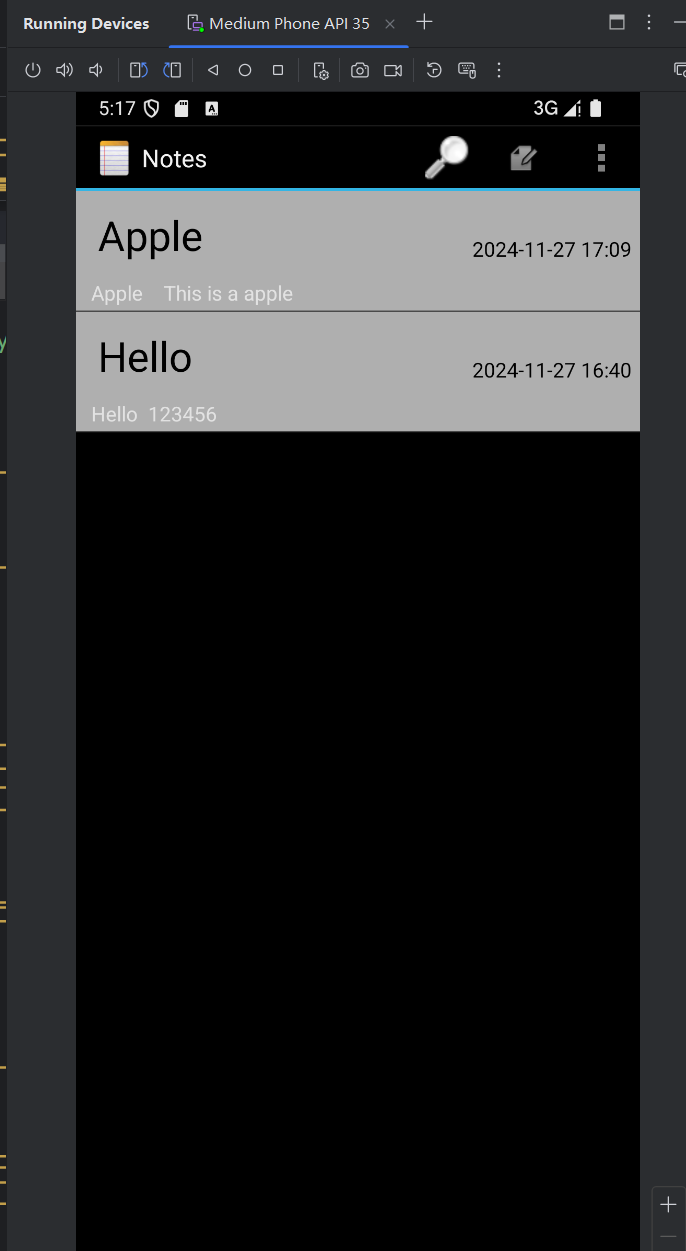
[4. 效果展示 23](#_Toc22092)

# 完成任务

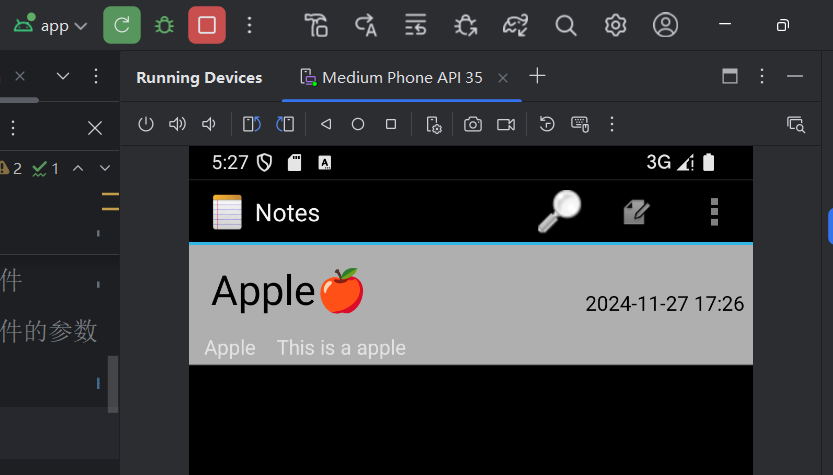
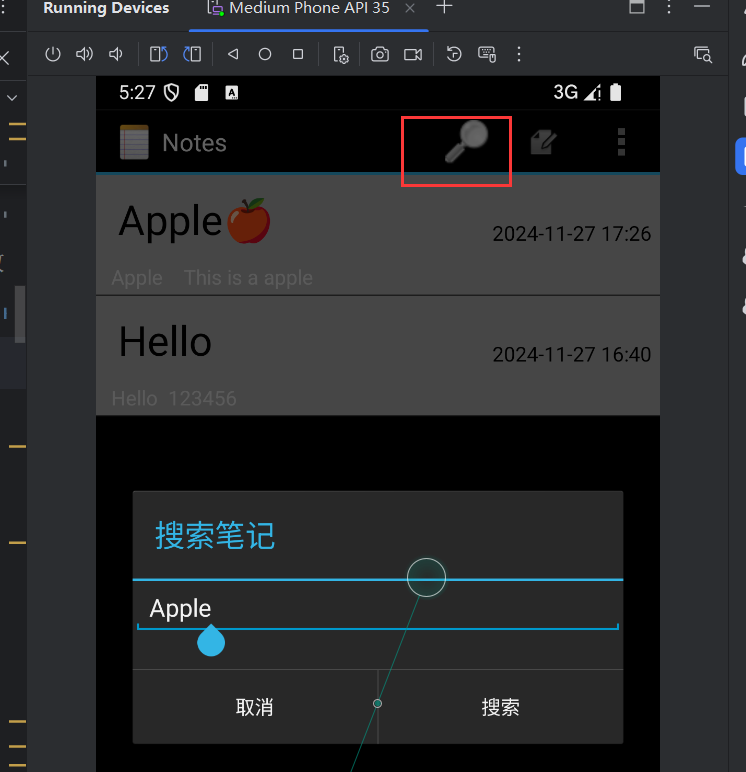
|  |  |  |
| --- | --- | --- |
| 分类 | 实现功能 | 备注 |
| 基本功能 | 显示笔记条目的时间戳 | 增加了文本内容的摘要显示  搜索为标题的模糊搜索 |
| 笔记内容的搜索功能 |
| 拓展功能 | UI美化 |
| 笔记文本导出 |
| 字体颜色改变 |

# 效果展示

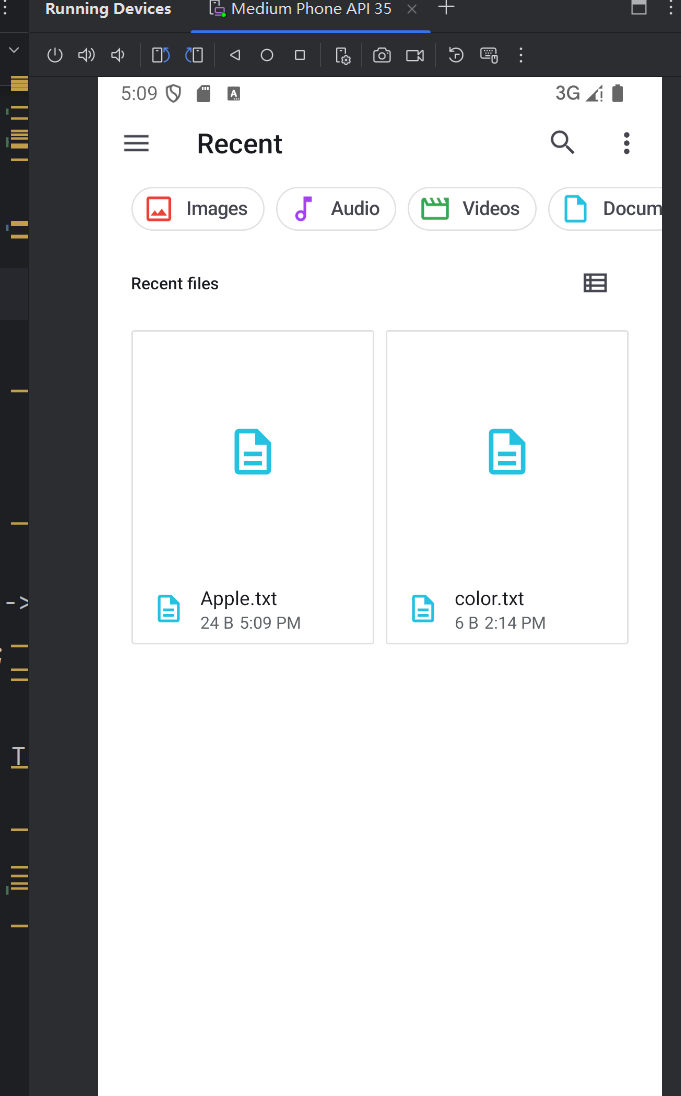
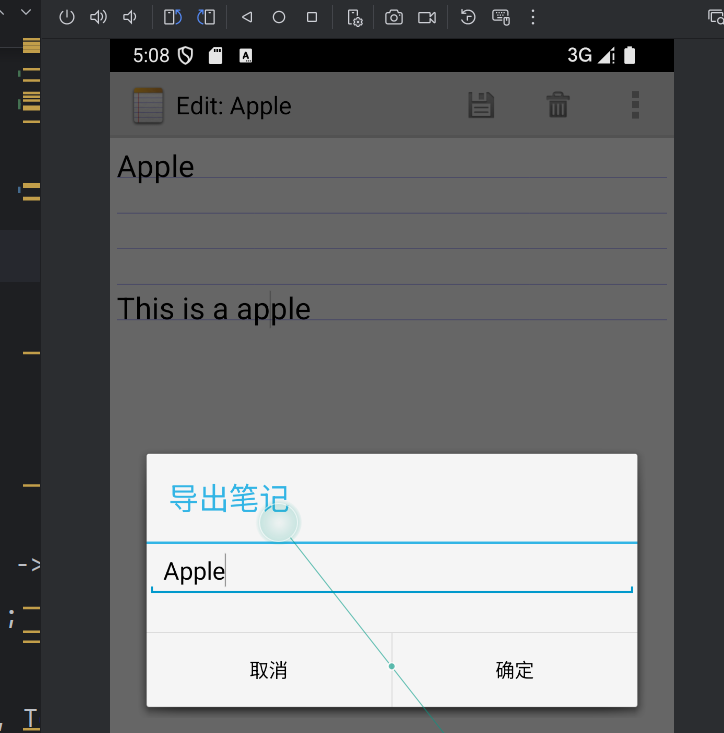
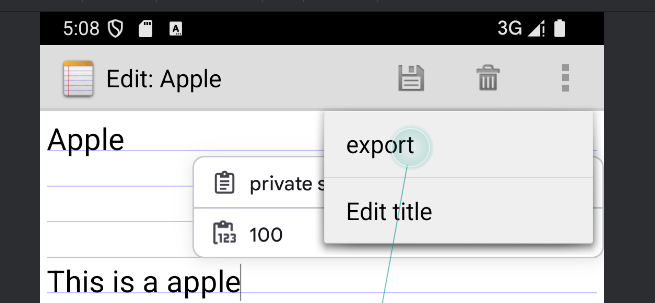
## 时间戳+UI



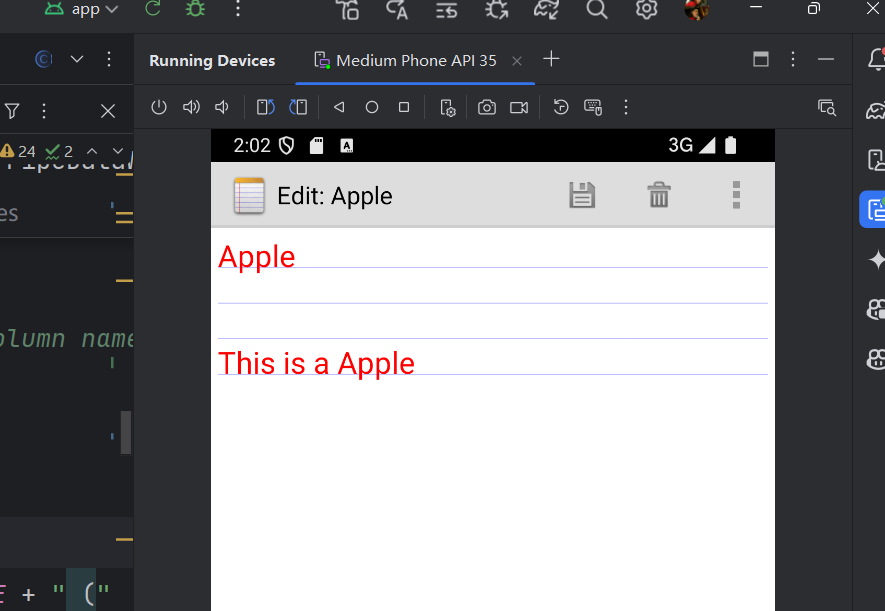
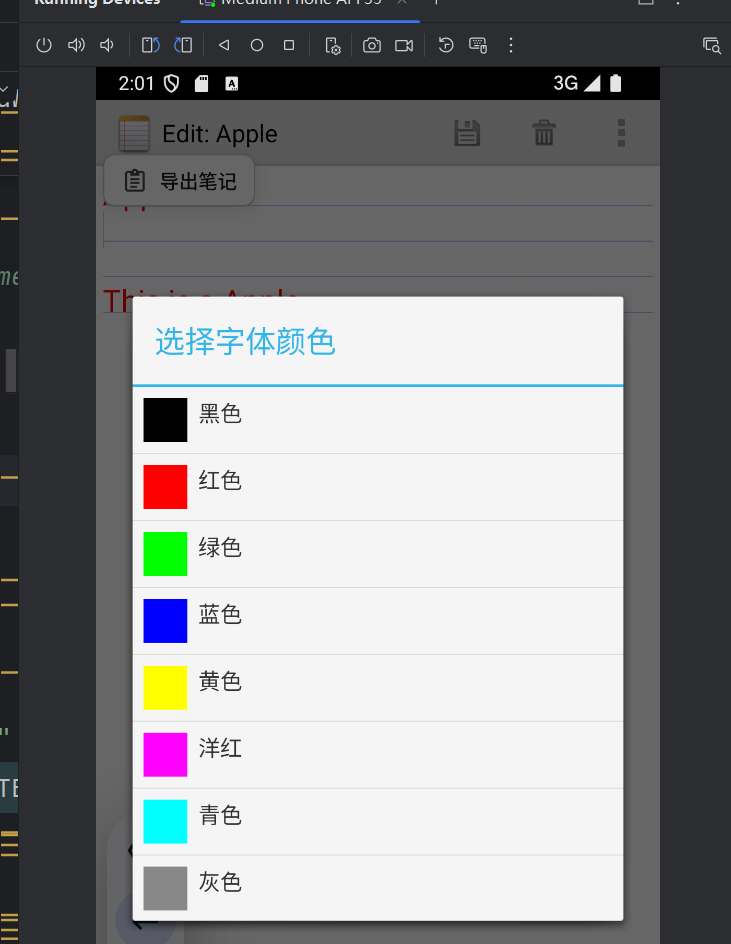
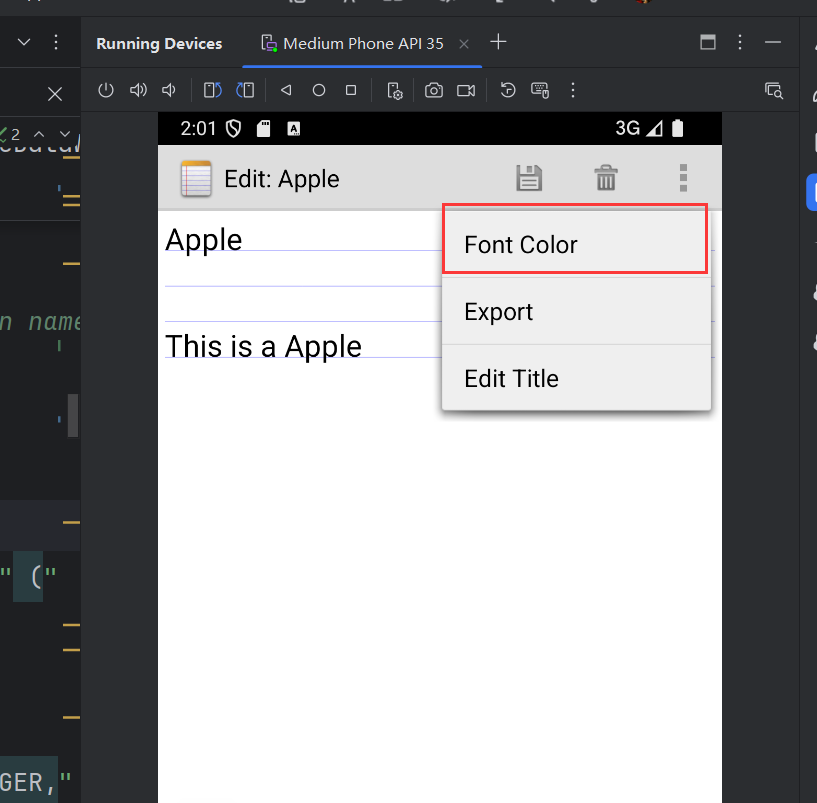
## 搜索功能



## 导出笔记



## 改变字体颜色

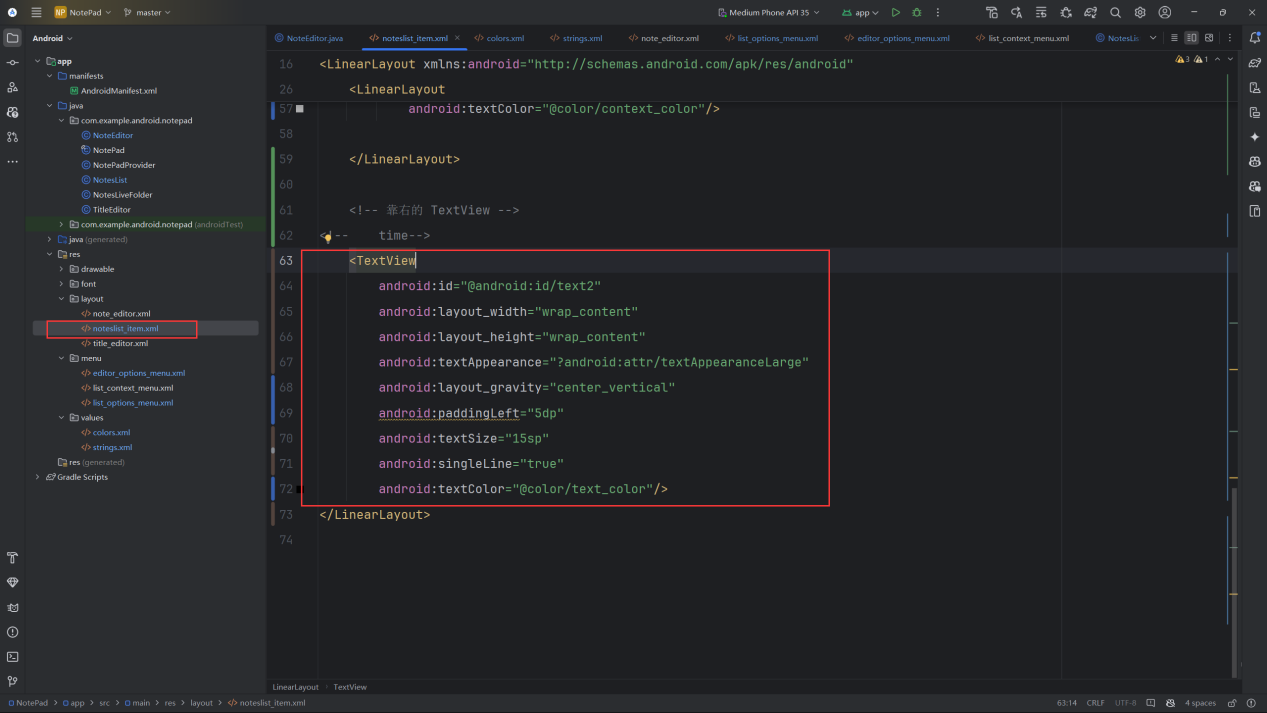


# 具体实现

## 一、时间戳显示

## 1.noteslist\_item.xml

添加timetamp控件布局



<TextView

android:id="@android:id/text2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceLarge"

android:layout\_gravity="center\_vertical"

android:paddingLeft="5dp"

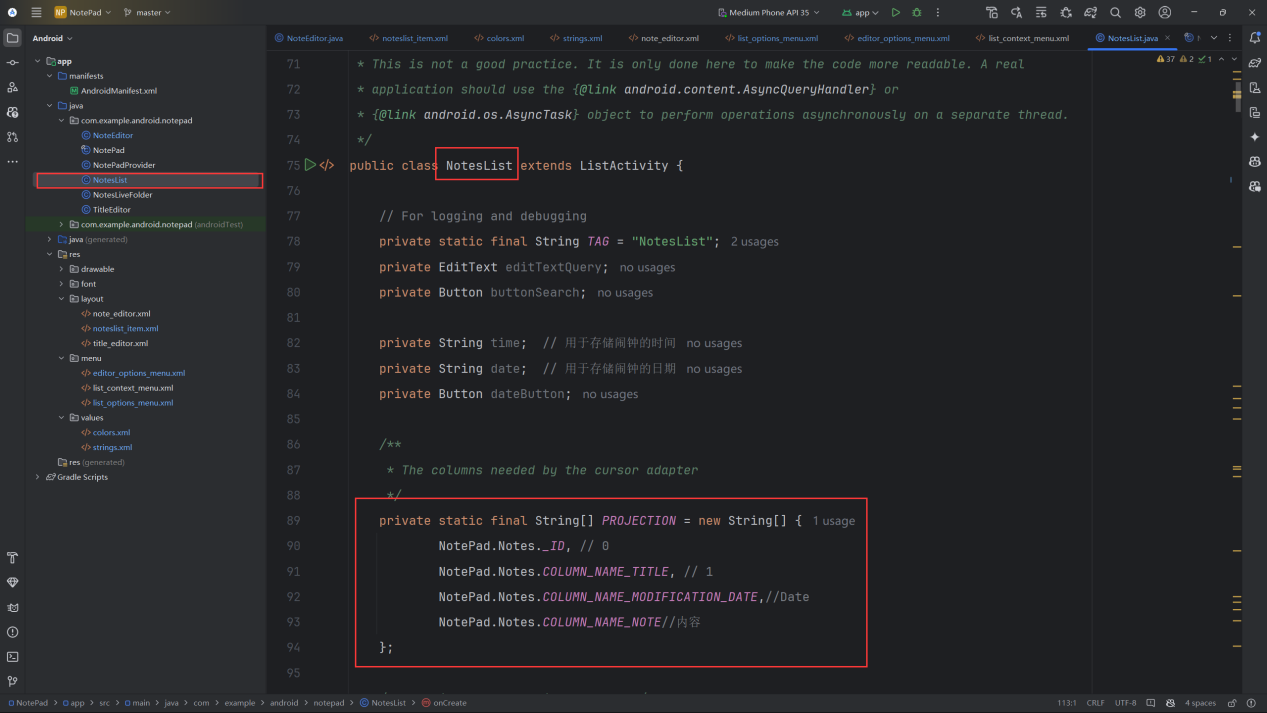
android:textSize="15sp"

android:singleLine="true"

android:textColor="@color/text\_color"/>

### 2.NotesList.java

增加PROJECTION



private static final String[] PROJECTION = new String[] {

NotePad.Notes.\_ID, // 0

NotePad.Notes.COLUMN\_NAME\_TITLE, // 1

NotePad.Notes.COLUMN\_NAME\_MODIFICATION\_DATE//Date

};

显示

protected void onCreate(Bundle savedInstanceState)方法中：

// 创建映射列和视图ID

String[] dataColumns = { NotePad.Notes.COLUMN\_NAME\_TITLE, NotePad.Notes.COLUMN\_NAME\_MODIFICATION\_DATE };

int[] viewIDs = { android.R.id.text1, android.R.id.text2 };

// 创建SimpleCursorAdapter

SimpleCursorAdapter adapter = new SimpleCursorAdapter(

this, // 上下文

R.layout.noteslist\_item, // 列表项布局

cursor, // 游标

dataColumns, // 数据列

viewIDs // 显示视图ID

);

// 设置ViewBinder来自定义时间显示

adapter.setViewBinder(new SimpleCursorAdapter.ViewBinder() {

@Override

public boolean setViewValue(View view, Cursor cursor, int columnIndex) {

if (columnIndex == cursor.getColumnIndex(NotePad.Notes.COLUMN\_NAME\_MODIFICATION\_DATE)) {

long timestamp = cursor.getLong(columnIndex);

String formattedDate = formatDate(timestamp);

TextView textView = (TextView) view;

textView.setText(formattedDate);

return true;

}

return false; // 对其他列不做处理

}

});

// 设置ListView的适配器

setListAdapter(adapter);

}

格式化时间戳为日期字符串



private String formatDate(long timestamp) {

// 创建一个日期格式化器

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm", Locale.getDefault());

// 获取当前时区的时区对象

TimeZone timeZone = TimeZone.getDefault();

// 设置 SimpleDateFormat 使用当前时区

dateFormat.setTimeZone(timeZone);

// 将时间戳转换为 Date 对象，并格式化为字符串

return dateFormat.format(new Date(timestamp));

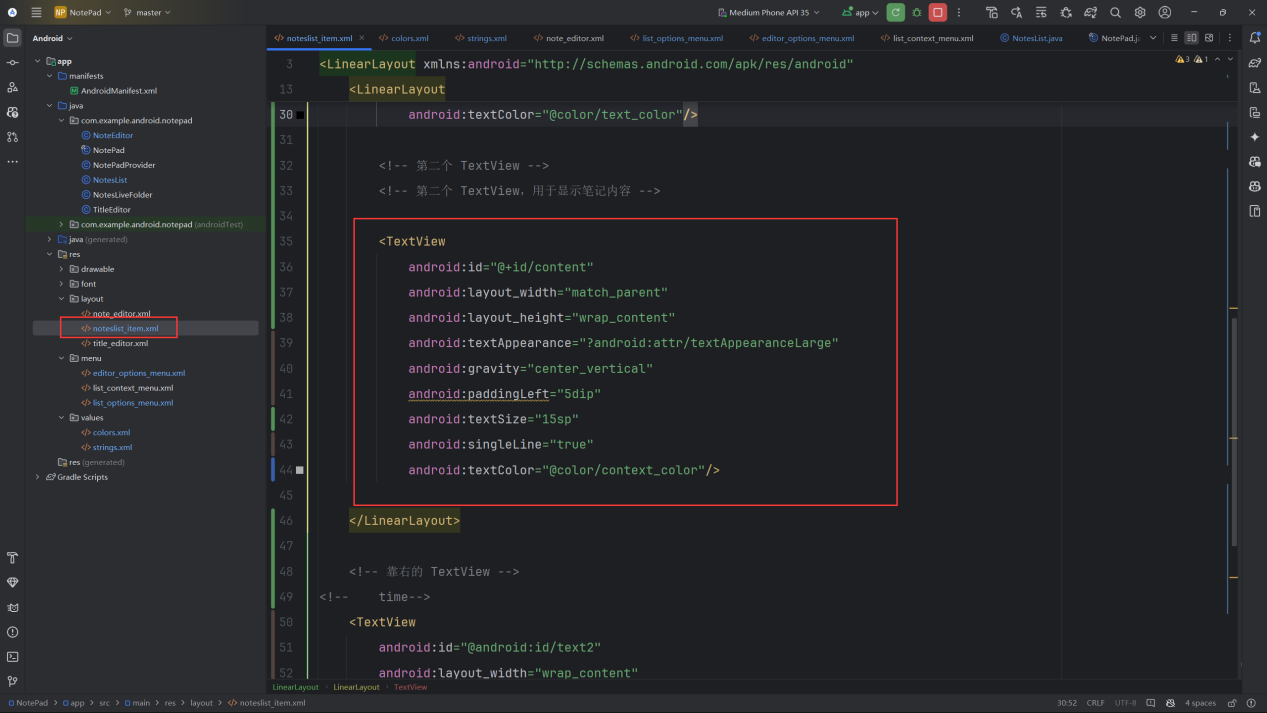
}

## 二、摘要内容展示

### 1.noteslist\_item.xml中

线性布局中，外部容器是水平布局

内部嵌套一个标题和内容的垂直布局



<TextView

android:id="@android:id/text1"

android:layout\_width="match\_parent"

android:layout\_height="?android:attr/listPreferredItemHeight"

android:textAppearance="?android:attr/textAppearanceLarge"

android:gravity="center\_vertical"

android:paddingLeft="10dip"

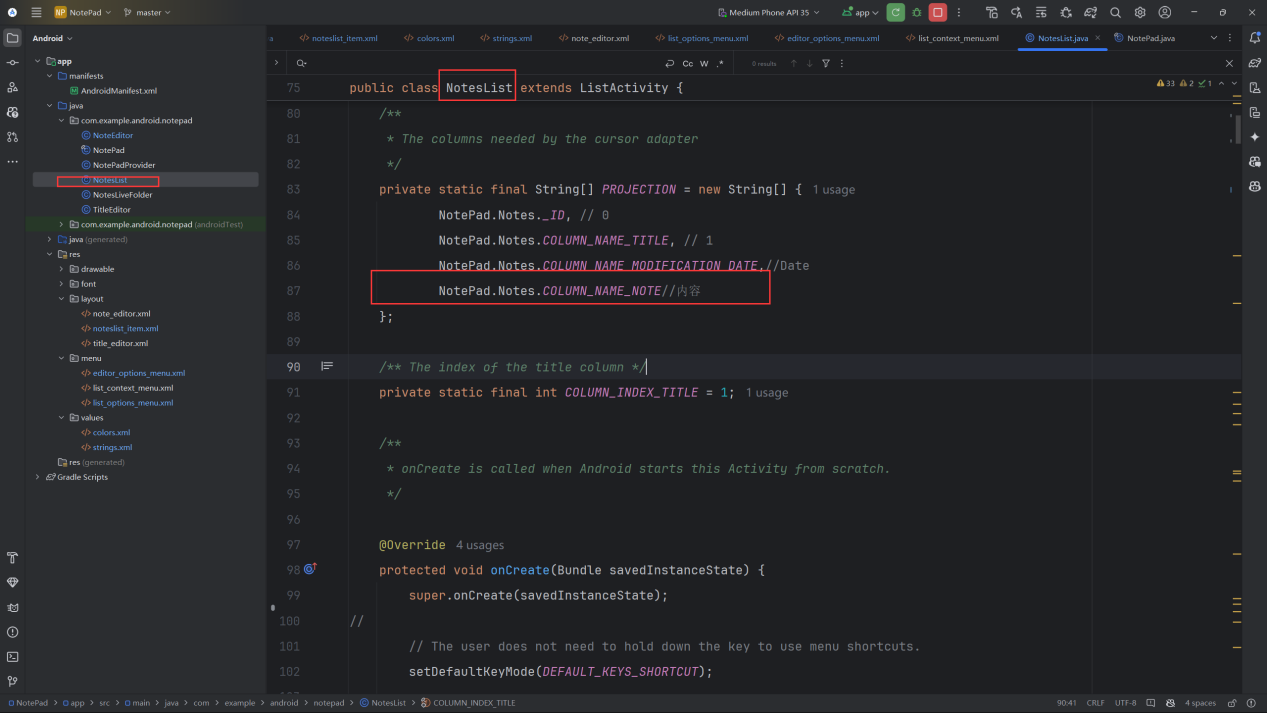
android:textSize="30sp"

android:singleLine="true"

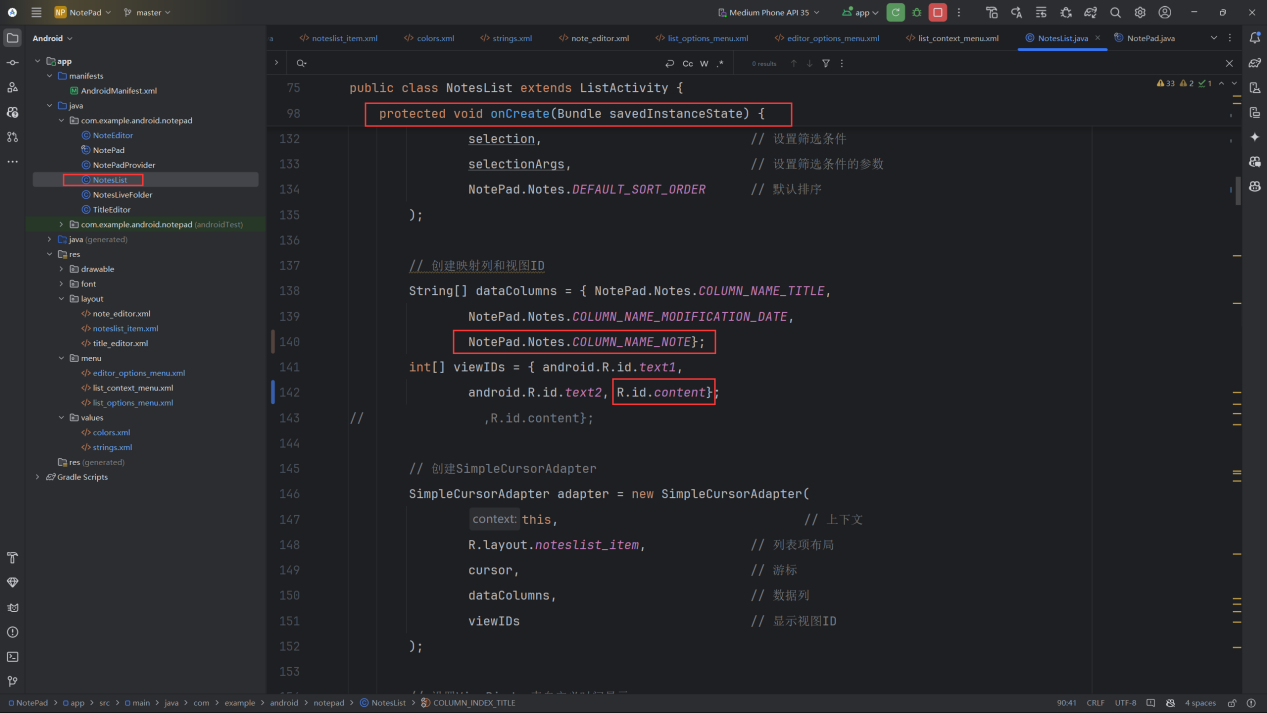
android:textColor="@color/text\_color"/>

### 2.NoteLists.java

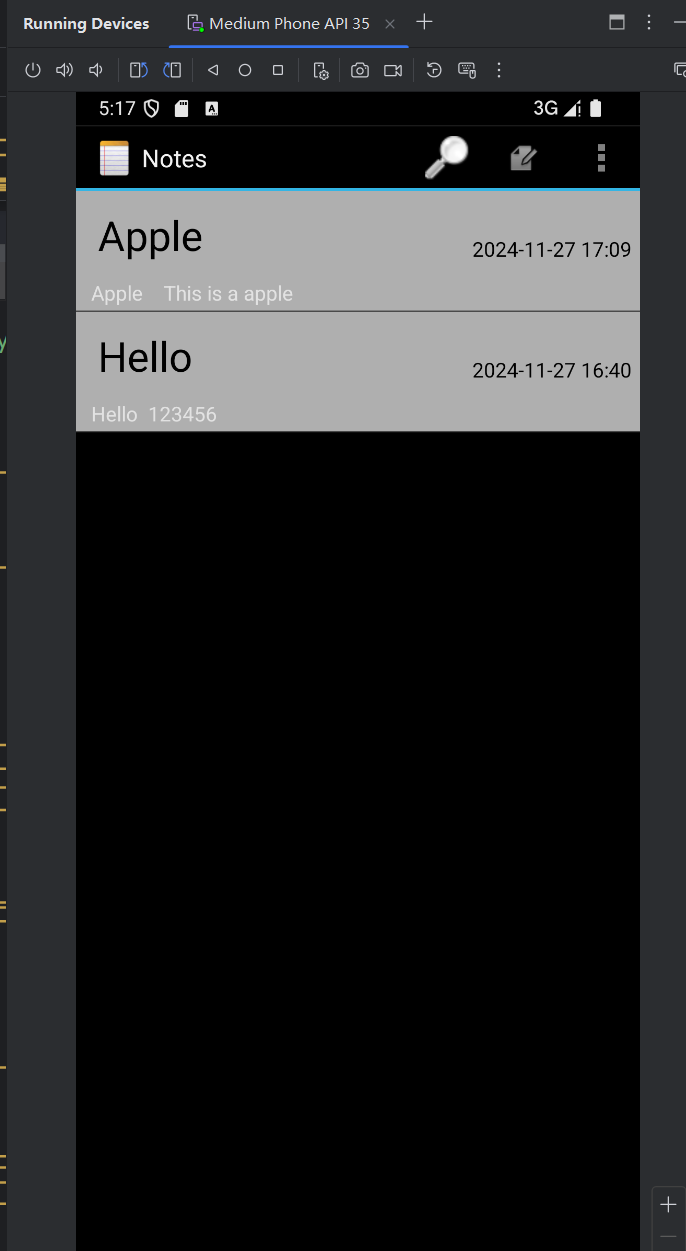
增加内容列



创建映射列和视图ID



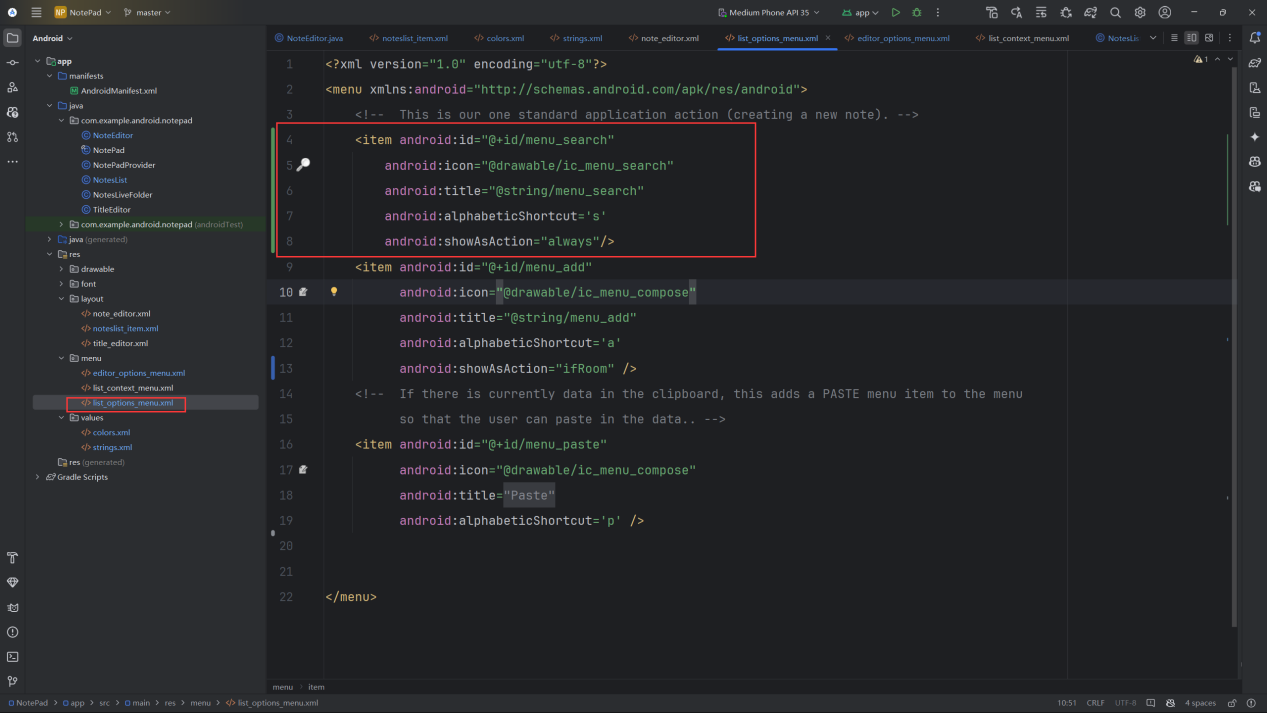
## 3.效果展示



## 三、搜索功能

### 1.list\_options\_menu.xml

增加搜索的控件和图片



<item android:id="@+id/menu\_search"

android:icon="@drawable/ic\_menu\_search"

android:title="@string/menu\_search"

android:alphabeticShortcut='s'

android:showAsAction="ifRoom"/>

### 2.NotesList.java

显示输入操作

public boolean onOptionsItemSelected(MenuItem item) {

//搜索

case R.id.menu\_search:

// 启动一个搜索界面或弹出输入框

// 在这里假设你使用一个 EditText 来获取搜索关键词

// 例如，你可以使用一个对话框来输入查询内容：

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle("搜索笔记");

// 设置输入框

final EditText input = new EditText(this);

builder.setView(input);

builder.setPositiveButton("搜索", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

String searchQuery = input.getText().toString().trim();

if (!searchQuery.isEmpty()) {

// 创建一个Intent传递搜索关键词

Intent intent = new Intent(NotesList.this, NotesList.class);

intent.putExtra("searchQuery", searchQuery); // 将搜索内容传递给NotesList

startActivity(intent);

} else {

Toast.makeText(NotesList.this, "请输入搜索内容", Toast.LENGTH\_SHORT).show();

}

}

});

builder.setNegativeButton("取消", null);

// 显示对话框

builder.show();

return true;

}

执行搜索操作

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// The user does not need to hold down the key to use menu shortcuts.

setDefaultKeyMode(DEFAULT\_KEYS\_SHORTCUT);

// 获取传递过来的搜索条件

String searchQuery = getIntent().getStringExtra("searchQuery");

// 获取Intent

Intent intent = getIntent();

// 如果没有传递URI，设置默认URI

if (intent.getData() == null) {

intent.setData(NotePad.Notes.CONTENT\_URI);

}

// 设置ListView的上下文菜单监听

getListView().setOnCreateContextMenuListener(this);

// 定义查询条件

String selection = null;

String[] selectionArgs = null;

// 如果有搜索条件，设置查询条件

if (searchQuery != null && !searchQuery.isEmpty()) {

selection = NotePad.Notes.COLUMN\_NAME\_TITLE + " LIKE ?"; // 根据标题过滤

selectionArgs = new String[]{"%" + searchQuery + "%"}; // 搜索关键字

}

// 执行查询，传入过滤条件

Cursor cursor = managedQuery(

getIntent().getData(), // 使用默认内容URI

PROJECTION, // 返回的列

selection, // 设置筛选条件

selectionArgs, // 设置筛选条件的参数

NotePad.Notes.DEFAULT\_SORT\_ORDER // 默认排序

);

// 创建映射列和视图ID

String[] dataColumns = { NotePad.Notes.COLUMN\_NAME\_TITLE, NotePad.Notes.COLUMN\_NAME\_MODIFICATION\_DATE };

int[] viewIDs = { android.R.id.text1, android.R.id.text2 };

// 创建SimpleCursorAdapter

SimpleCursorAdapter adapter = new SimpleCursorAdapter(

this, // 上下文

R.layout.noteslist\_item, // 列表项布局

cursor, // 游标

dataColumns, // 数据列

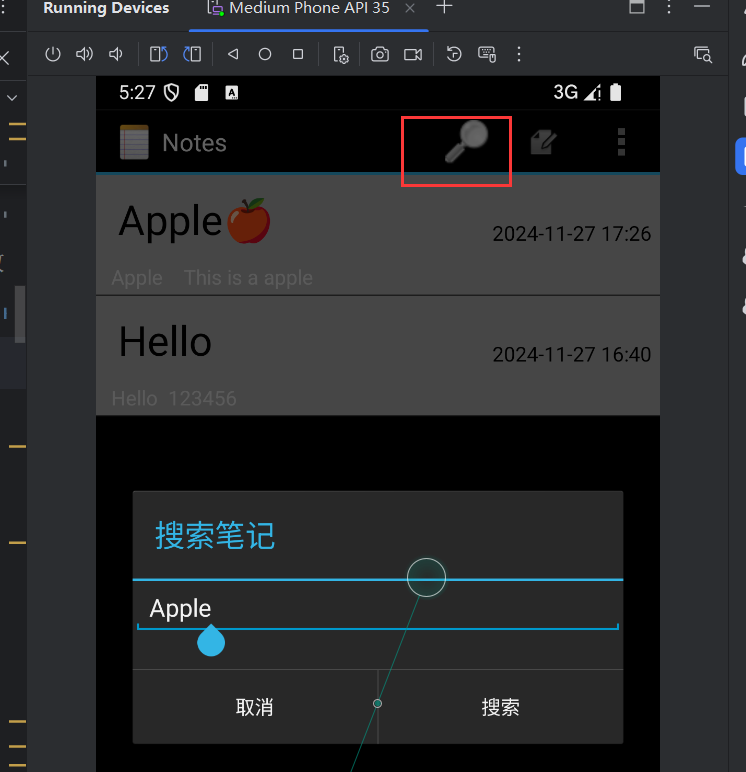
viewIDs // 显示视图ID

);

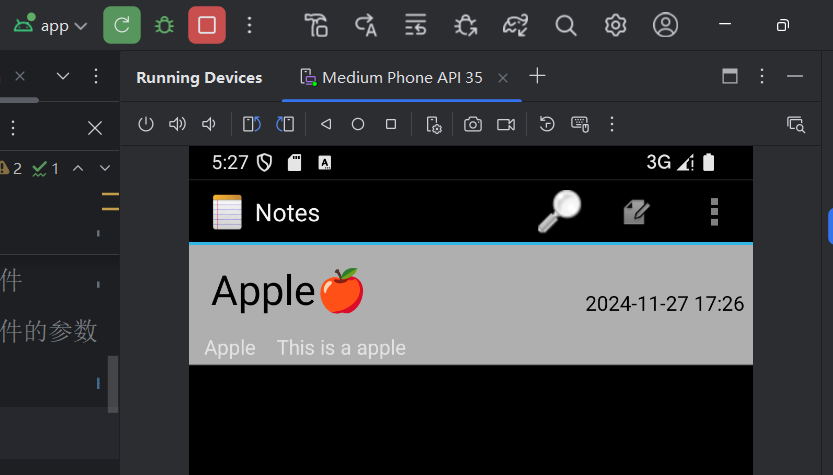
setListAdapter(adapter);

}

## 3.效果展示



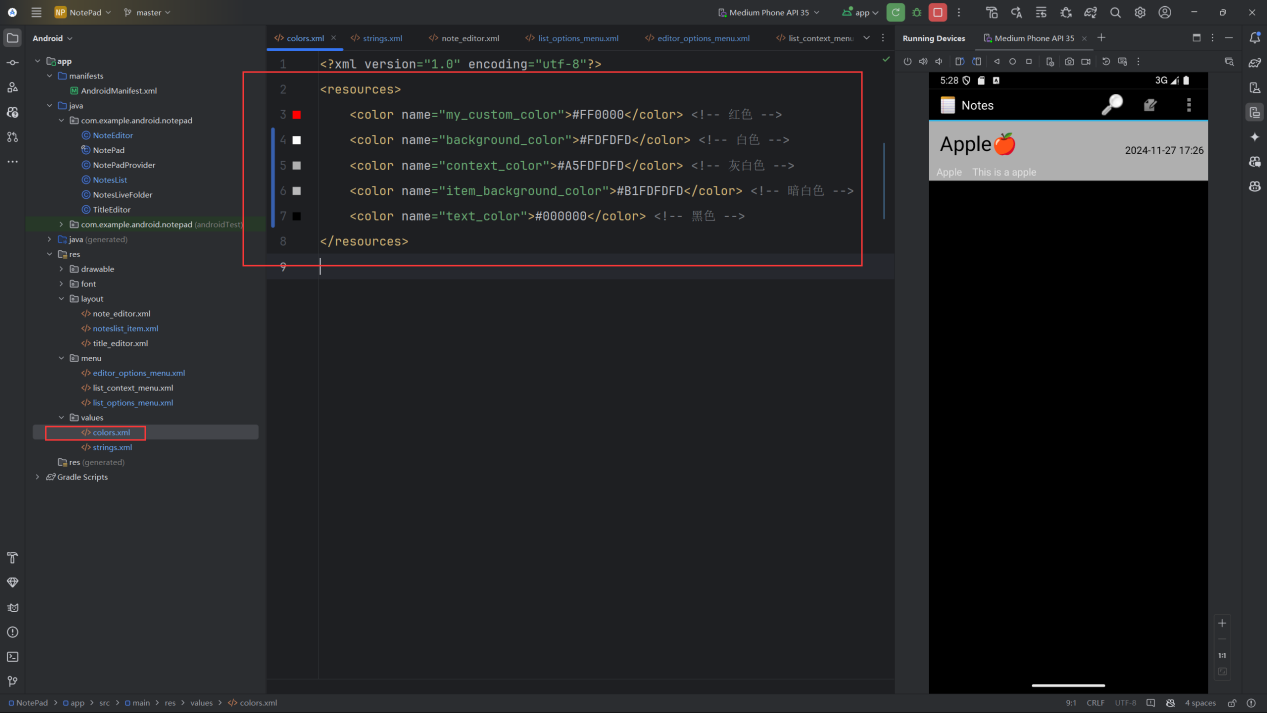
模糊搜索结果



## 四、UI美化

### 1.color.xml

自定义背景资源文件



<resources>

<color name="my\_custom\_color">#FF0000</color> <!-- 红色 -->

<color name="background\_color">#FDFDFD</color> <!-- 白色 -->

<color name="context\_color">#A5FDFDFD</color> <!-- 灰白色 -->

<color name="item\_background\_color">#B1FDFDFD</color> <!-- 暗白色 -->

<color name="text\_color">#000000</color> <!-- 黑色 -->

</resources>

## noteslist\_item.xml

更改Item条目的颜色

android:background="@color/item\_background\_color"

设置item字体显示

<!--title-->

android:textColor="@color/text\_color"

<!--content-->

android:textColor="@color/content\_color"

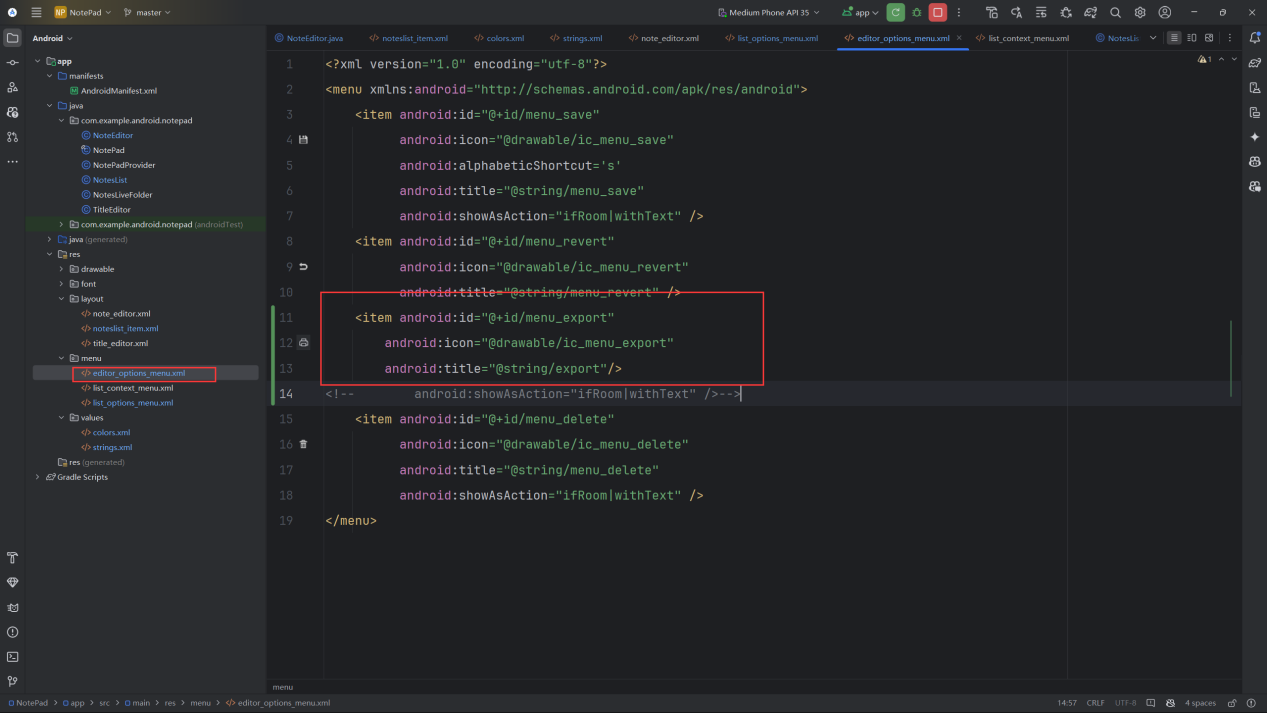
<!-- timetamp-->

android:textColor="@color/text\_color"

四、导出文件

## 五、导出笔记

### 1.editor\_options\_menu.xml

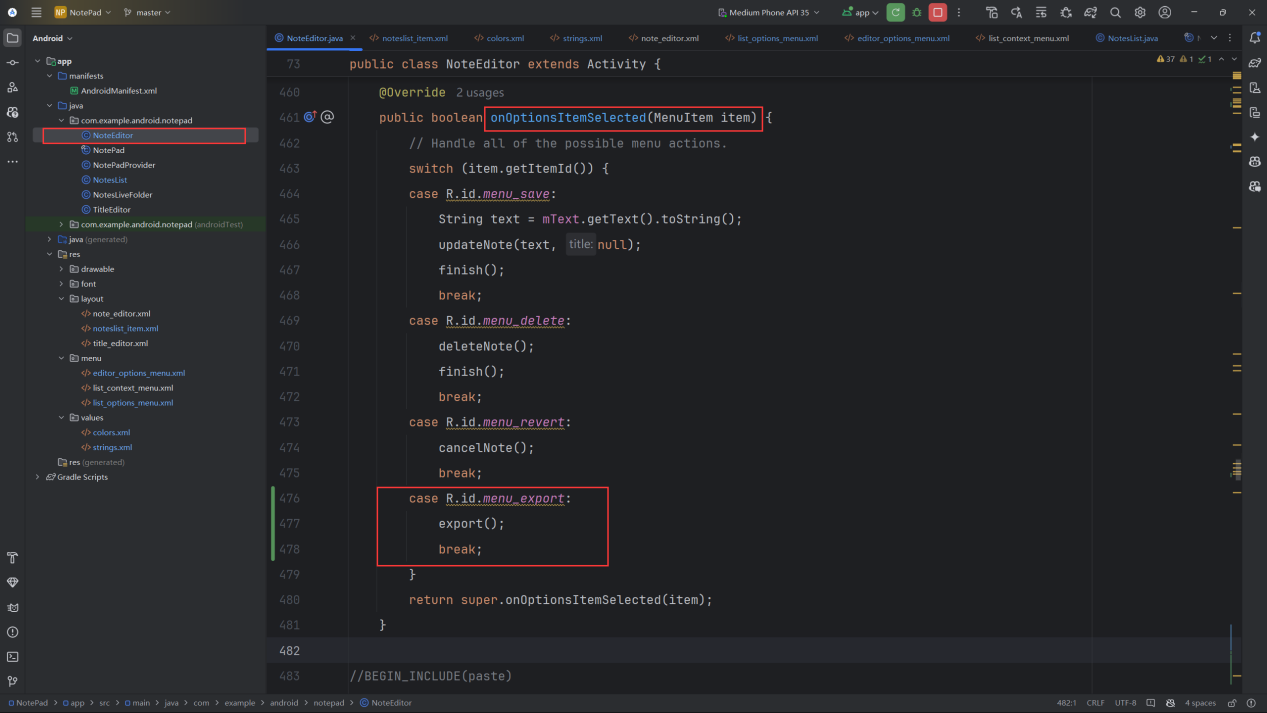


<item android:id="@+id/menu\_export"

android:icon="@drawable/ic\_menu\_export"

android:title="@string/export"/>

### 2.NoteEditor.java



3.在本类中增加方法

private static final int REQUEST\_CODE\_EXPORT = 100; // 100 是一个随意选定的

private void export(){

// 创建一个输入框

EditText input = new EditText(this);

input.setHint("请输入文件名");

// 弹出对话框

AlertDialog dialog = new AlertDialog.Builder(this)

.setTitle("导出笔记")

.setView(input)

.setPositiveButton("确定", (dialogInterface, which) -> {

String fileName = input.getText().toString().trim();

if (fileName.isEmpty()) {

Toast.makeText(this, "文件名不能为空", Toast.LENGTH\_SHORT).show();

} else {

// 启动文件选择器

openFilePicker(fileName);

}

})

.setNegativeButton("取消", null)

.create();

dialog.show();

dialog.getButton(AlertDialog.BUTTON\_POSITIVE).setTextColor(Color.BLACK);

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (requestCode == REQUEST\_CODE\_EXPORT && resultCode == RESULT\_OK) {

Uri fileUri = data.getData();

if (fileUri != null) {

saveNoteToFile(fileUri);

} else {

Toast.makeText(this, "文件创建失败", Toast.LENGTH\_SHORT).show();

}

}

}

private void saveNoteToFile(Uri fileUri) {

try {

// 获取笔记内容

String noteContent = mText.getText().toString();

// 打开输出流并写入数据

try (OutputStream outputStream = getContentResolver().openOutputStream(fileUri)) {

if (outputStream != null) {

outputStream.write(noteContent.getBytes());

outputStream.flush();

Toast.makeText(this, "笔记导出成功", Toast.LENGTH\_SHORT).show();

} else {

Toast.makeText(this, "无法打开文件", Toast.LENGTH\_SHORT).show();

}

}

} catch (Exception e) {

e.printStackTrace();

Toast.makeText(this, "导出失败：" + e.getMessage(), Toast.LENGTH\_SHORT).show();

}

}

private void openFilePicker(String fileName) {

Intent intent = new Intent(Intent.ACTION\_CREATE\_DOCUMENT);

intent.addCategory(Intent.CATEGORY\_OPENABLE);

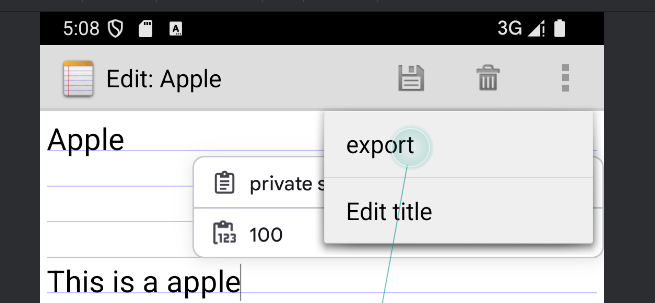
intent.setType("text/plain");

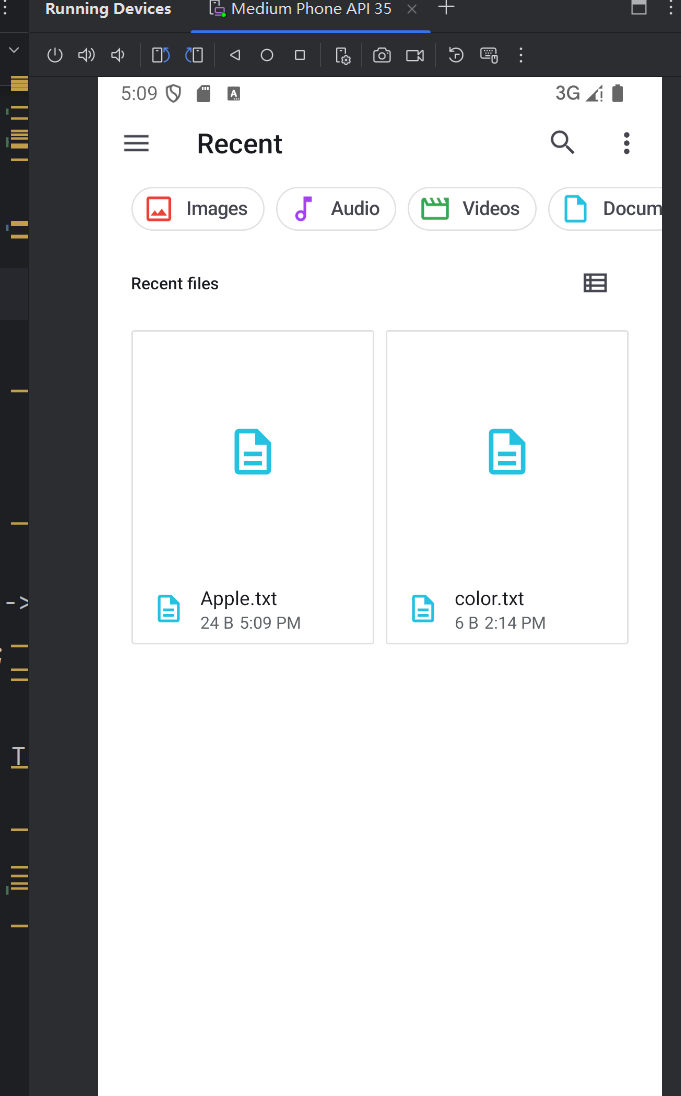
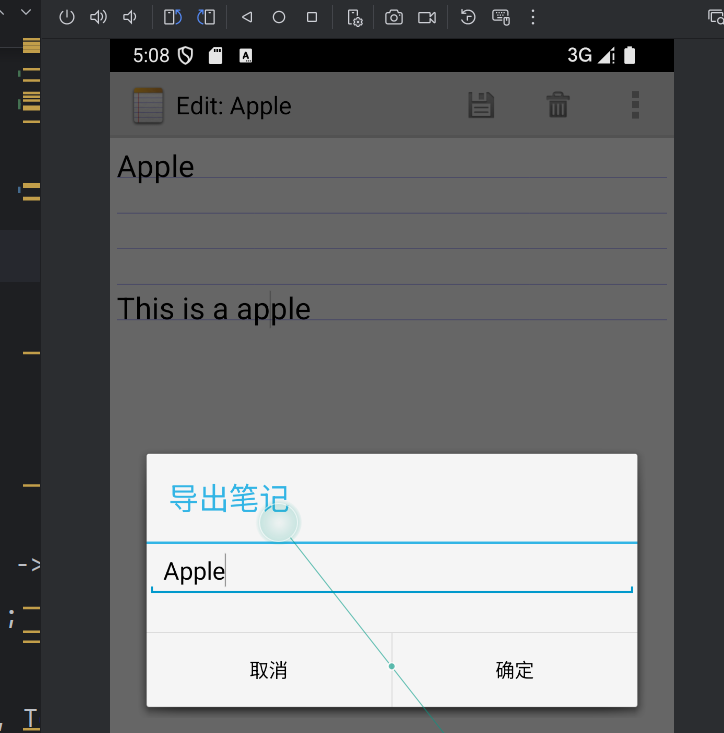
intent.putExtra(Intent.EXTRA\_TITLE, fileName + ".txt"); // 用户输入的文件名

startActivityForResult(intent, REQUEST\_CODE\_EXPORT);

}

## 3.效果展示

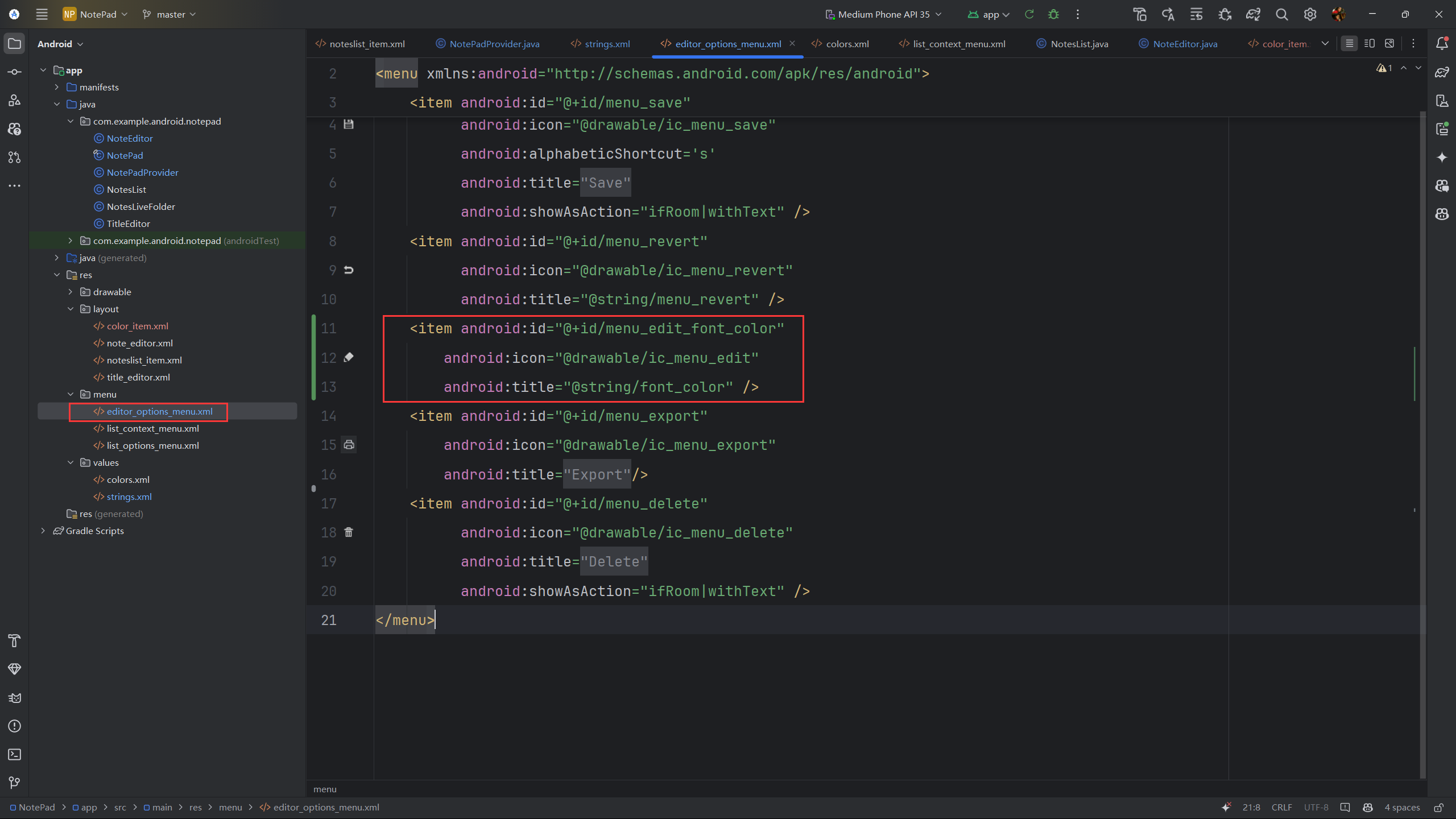




# 改变字体颜色

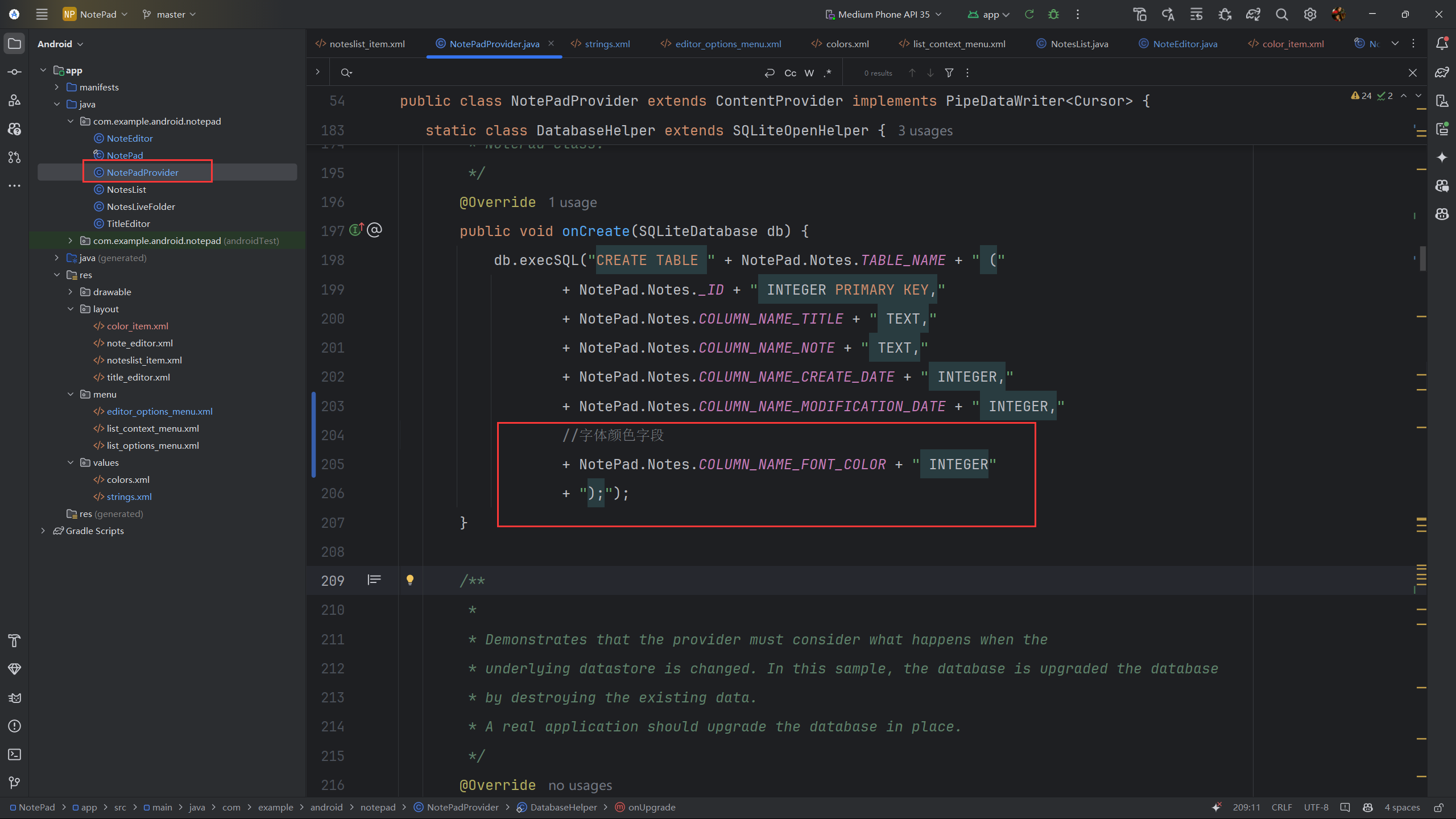
## editor\_options\_menu.xml

增加控件

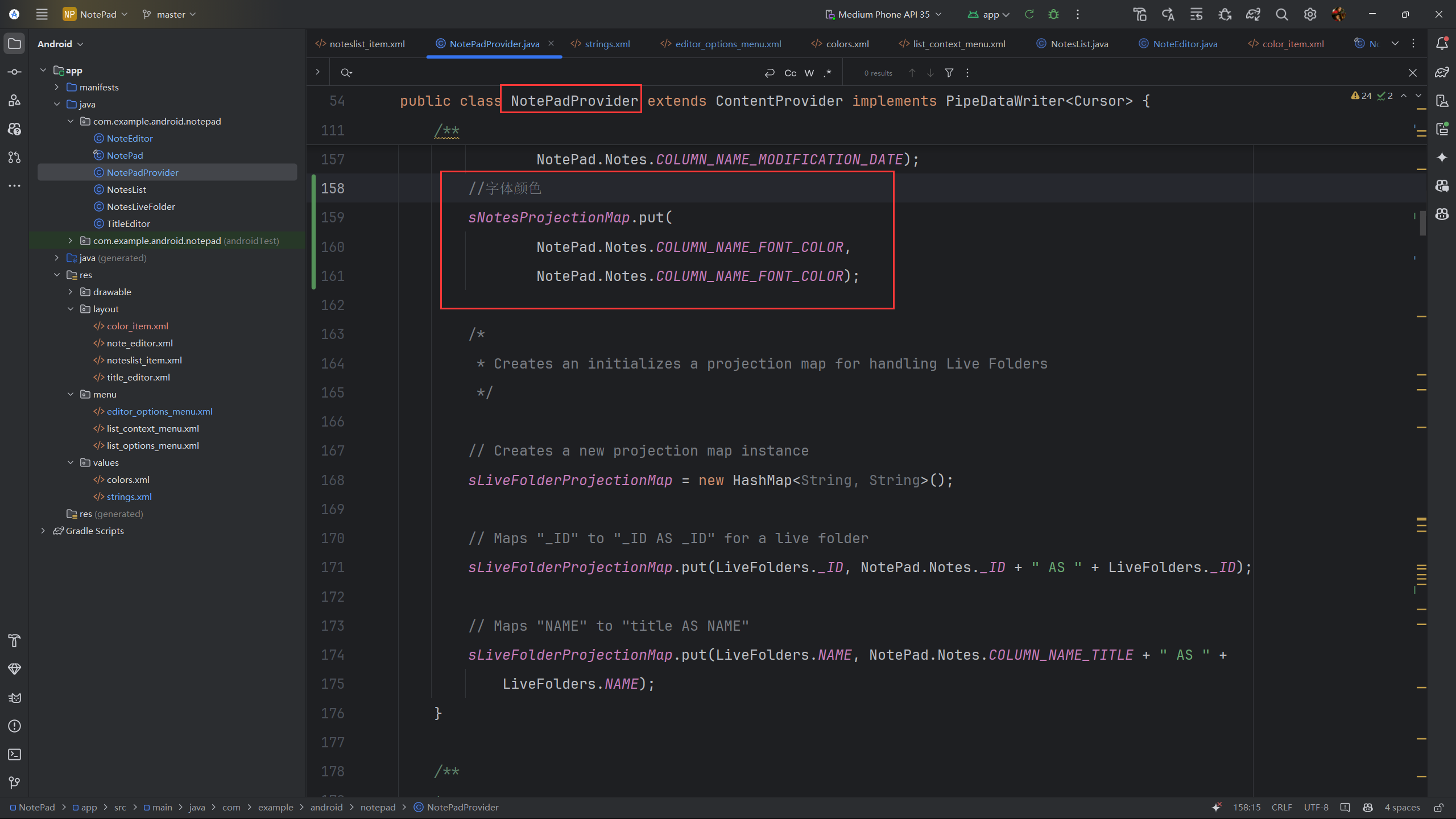


## NotePadProvider.java

在原有的数据库表中增加颜色字段



增加字体颜色的映射



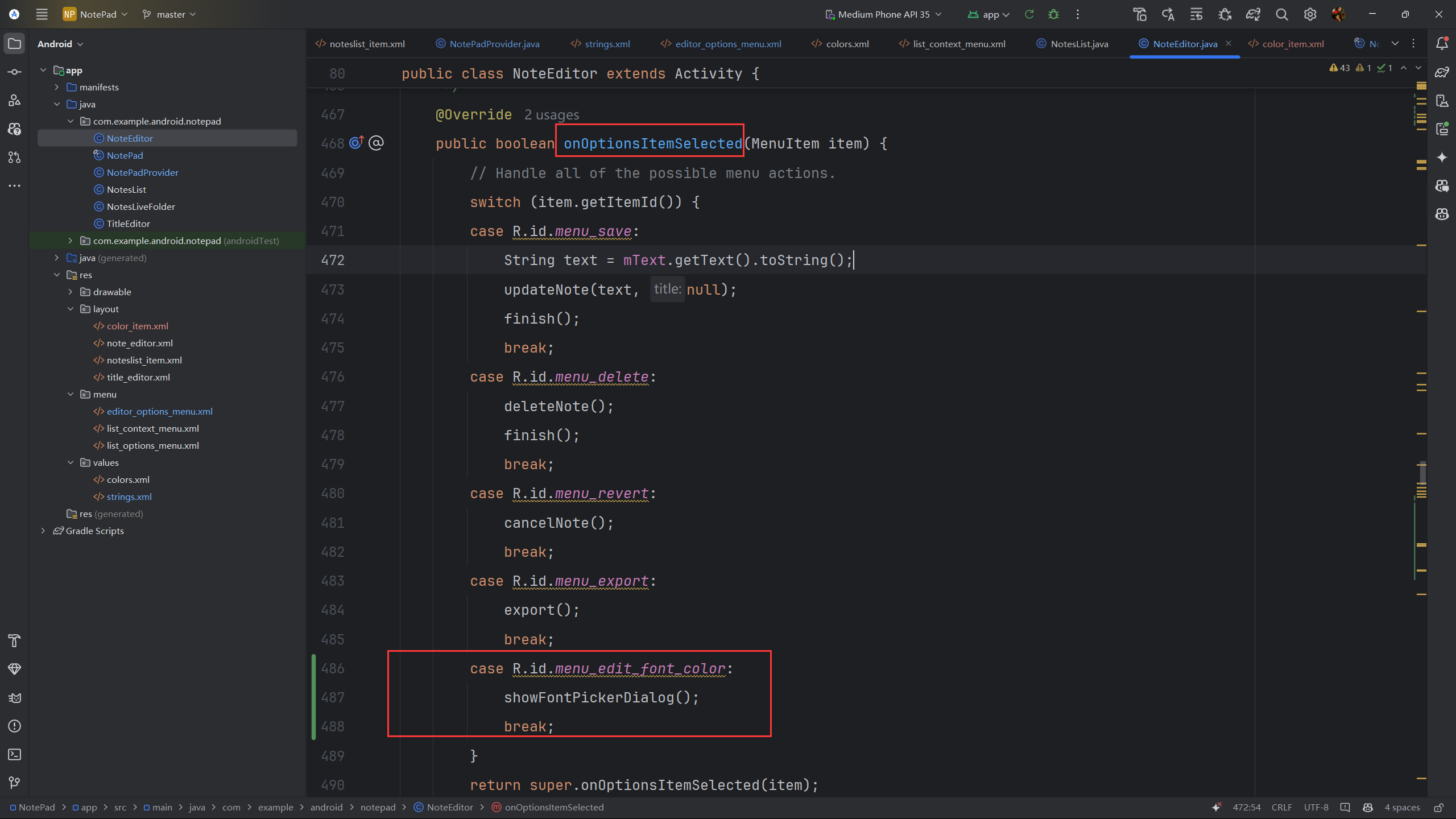
public Uri insert(Uri uri, ContentValues initialValues)中

增加字体默认颜色

//字体默认  
if (!values.containsKey(NotePad.Notes.COLUMN\_NAME\_FONT\_COLOR)) {  
 values.put(NotePad.Notes.COLUMN\_NAME\_FONT\_COLOR, "#000000");  
}

## NoteEditor.java

增加switch选项



实现相应的方法

//字体颜色  
 private void showFontPickerDialog() {  
 // 创建颜色列表  
 List<ColorItem> colorItems = Arrays.asList(  
 new ColorItem(Color.BLACK, "黑色"),  
 new ColorItem(Color.RED, "红色"),  
 new ColorItem(Color.GREEN, "绿色"),  
 new ColorItem(Color.BLUE, "蓝色"),  
 new ColorItem(Color.YELLOW, "黄色"),  
 new ColorItem(Color.MAGENTA, "洋红"),  
 new ColorItem(Color.CYAN, "青色"),  
 new ColorItem(Color.GRAY, "灰色")  
 );  
  
 // 创建适配器  
 ColorAdapter adapter = new ColorAdapter(this, colorItems);  
  
 // 创建对话框  
 new AlertDialog.Builder(this)  
 .setTitle("选择字体颜色")  
 .setAdapter(adapter, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 // 获取用户选择的颜色  
 int selectedColor = colorItems.get(which).getColor();  
 // 更新 EditText 的字体颜色  
 mText.setTextColor(selectedColor);  
 // 将选择的颜色同步到数据库  
 updateFontColor(selectedColor);  
 }  
 })  
 .show();  
 }  
 private void updateFontColor(int color) {  
 // 创建 ContentValues 对象以存储要更新的值  
 ContentValues values = new ContentValues();  
 values.put(NotePad.Notes.COLUMN\_NAME\_FONT\_COLOR, String.format("#%06X", (0xFFFFFF & color))); // 将颜色转换为字符串  
  
// 更新数据库中对应的记录  
 getContentResolver().update(  
 mUri, // 要更新的 URI  
 values, // 要更新的值  
 null, // 不需要选择条件  
 null // 不需要选择参数  
 );  
 }  
 private class ColorAdapter extends ArrayAdapter<ColorItem> {  
 private Context context;  
 private List<ColorItem> colorItems;  
  
 public ColorAdapter(Context context, List<ColorItem> colorItems) {  
 super(context, 0, colorItems);  
 this.context = context;  
 this.colorItems = colorItems;  
 }  
  
 @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
 if (convertView == null) {  
 convertView = LayoutInflater.from(context).inflate(R.layout.color\_item, parent, false);  
 }  
  
 ColorItem colorItem = colorItems.get(position);  
  
 View colorPreview = convertView.findViewById(R.id.color\_preview);  
 TextView colorName = (TextView) convertView.findViewById(R.id.color\_name);  
  
 colorPreview.setBackgroundColor(colorItem.getColor());  
 colorName.setText(colorItem.getName());  
  
 return convertView;  
 }  
 }  
  
 private static class ColorItem {  
 private int color;  
 private String name;  
  
 public ColorItem(int color, String name) {  
 this.color = color;  
 this.name = name;  
 }  
  
 public int getColor() {  
 return color;  
 }  
  
 public String getName() {  
 return name;  
 }  
 }

## 效果展示

