

一. 初始应用的功能

1. 新建笔记和编辑笔记

(1) 在主界面点击下方的黄色按钮，新建笔记并进入编辑界面



实现步骤：

a 布局文件： fab_layout.xml 中定义了主界面的 FloatingActionButton，点击可进入新建笔记界面。

```
17 <com.google.android.material.floatingactionbutton.FloatingActionButton
18     android:id="@+id/fab_new_note"
19     android:layout_width="wrap_content"
20     android:layout_height="wrap_content"
21     android:layout_gravity="bottom|end"
22     android:layout_margin="24dp"
23     android:src="@drawable/ic_add_white_48dp"
24     app:backgroundTint="#FF9800"
25     app:tint="#FFFFFF"
26     app:elevation="8dp" />
```

b 点击事件：在 NotesList.java 的 onOptionsItemSelected() 方法中，处理 R.id.menu_add 事件，启动 NoteEditor 的 ACTION_INSERT。

c 跳转逻辑：通过 Intent.ACTION_INSERT 启动 NoteEditor，并传递数据 URI。

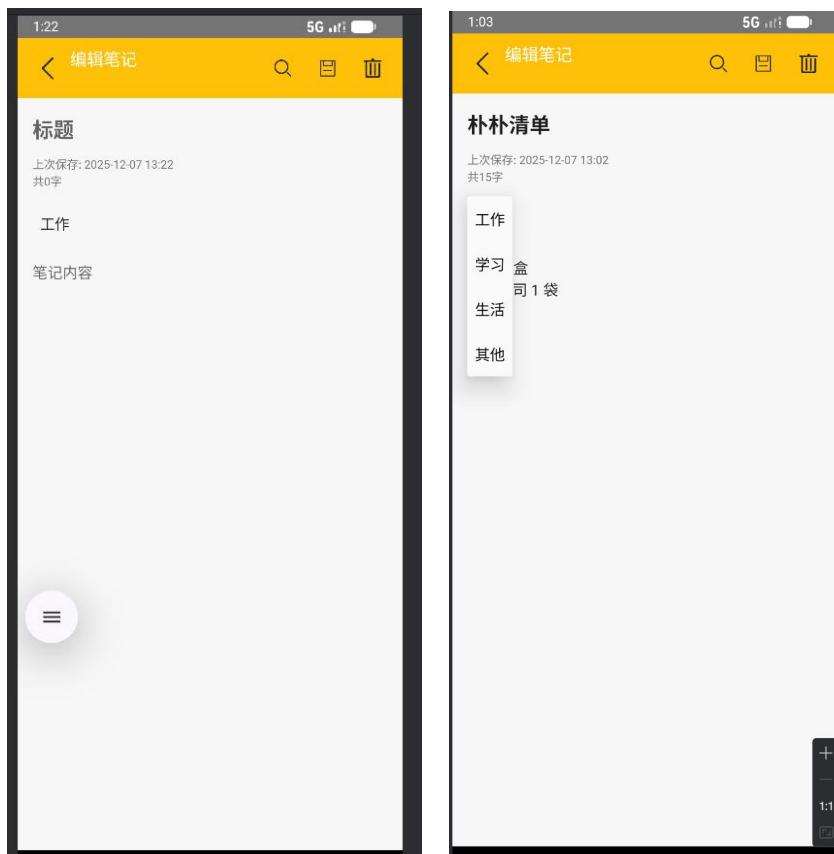
```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.menu_add) {

            Intent insertIntent = new Intent(Intent.ACTION_INSERT, getIntent().getData());
            insertIntent.setClassName( packageContext: this, NoteEditor.class.getName());
            startActivity(insertIntent);
            return true;

        } else if (item.getItemId() == R.id.menu_paste) {

            Intent pasteIntent = new Intent(Intent.ACTION_PASTE, getIntent().getData());
            pasteIntent.setClassName( packageContext: this, NoteEditor.class.getName());
            startActivity(pasteIntent);
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

这是新建笔记页面，输入标题和内容，可以通过下拉框选择分类，



(2) 进入笔记编辑界面后，可进行笔记编辑

可以通过下拉框进行分类设置，也可以自己新建分类

保存后可以显示字数

在该条笔记里面也可以进行搜索

添加内容后，点击右上角保存，可以发现时间和字数都更新了



实现步骤：

a 布局文件：note_editor.xml 定义了编辑界面的 UI，包括标题、分类、时间、字数统计和编辑区域。

```
<TextView  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:text="笔记编辑"  
    android:textColor="#FFFFFF"  
    android:textSize="18sp"  
    android:textStyle="bold"  
    android:gravity="center" />  
  
<ImageButton  
    android:id="@+id	btn_save"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@android:drawable/ic_menu_save"  
    android:background="?android:attr/selectableItemBackground"  
    android:contentDescription="保存"  
    tools:ignore="HardcodedText,ContentDescription" />  
  
<ImageButton  
    android:id="@+id	btn_search"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@android:drawable/ic_menu_search"  
    android:background="?android:attr/selectableItemBackground"  
    android:contentDescription="搜索"  
    tools:ignore="HardcodedText,ContentDescription" />  
  
<ImageButton  
    android:id="@+id	btn_delete"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@android:drawable/ic_menu_delete"  
    android:background="?android:attr/selectableItemBackground"  
    android:contentDescription="删除"  
    tools:ignore="HardcodedText,ContentDescription" />  
  
<!-- 标题行 -->  
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:orientation="horizontal"  
    android:layout_marginBottom="8dp">  
  
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="标题:"  
    android:textSize="16sp"  
    android:textStyle="bold" />  
  
<EditText  
    android:id="@+id/title"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:layout_marginStart="8dp"  
    android:layout_marginLeft="8dp"  
    android:textSize="16sp"  
    android:hint="输入标题"  
    android:maxLines="1"  
    android:inputType="textCapSentences"  
    android:importantForAutofill="yes"  
    tools:ignore="LabelFor" />  
</LinearLayout>
```

```

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginBottom="8dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="分类:"
        android:textSize="16sp"
        android:textStyle="bold" />

    <Spinner
        android:id="@+id/spinner_category"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp" />

    <Button
        android:id="@+id/btn_new_category"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="新建分类" />
</LinearLayout>

<!-- 时间戳和字数统计行 -->
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:gravity="center_vertical">

    <TextView
        android:id="@+id/tv_time"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="上次保存:"
        android:textSize="14sp"
        android:textColor="#666666" />

    <TextView
        android:id="@+id/tv_word_count"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="共0字"
        android:textSize="14sp"
        android:textColor="#666666"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp" />
</LinearLayout>
</LinearLayout>

<!-- 搜索框（默认隐藏） -->
<LinearLayout
    android:id="@+id/search_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="8dp"
    android:background="#FFE0B2"
    android:visibility="gone">

    <EditText
        android:id="@+id/search_box"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:hint="搜索笔记内容"
        android:inputType="text"
        android:importantForAutofill="yes" />

    <Button
        android:id="@+id/btn_search_content"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="搜索" />

    <Button
        android:id="@+id/btn_close_search"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="关闭" />
</LinearLayout>

```

b 编辑逻辑: NoteEditor.java 的 onCreate() 中根据 ACTION_INSERT 处理编辑或新建逻辑

```
public class NoteEditor extends Activity {
    protected void onCreate(Bundle savedInstanceState) {
        // 设置按钮点击事件
        setupButtons();

        // 初始化分类
        initCategories();

        final Intent intent = getIntent();
        final String action = intent.getAction();

        if (Intent.ACTION_EDIT.equals(action)) {
            mState = STATE_EDIT;
            mUri = intent.getData();
        } else if (Intent.ACTION_INSERT.equals(action) || Intent.ACTION_PASTE.equals(action)) {
            mState = STATE_INSERT;
            mUri = getContentResolver().insert(intent.getData(), values: null);

            if (mUri == null) {
                Log.e(TAG, msg: "Failed to insert new note");
                finish();
                return;
            }
            setResult(RESULT_OK, (new Intent()).setAction(mUri.toString()));
        } else {
            Log.e(TAG, msg: "Unknown action, exiting");
            finish();
            return;
        }
    }
}
```

c 分类选择：通过 Spinner (R.id.spinner_category) 和 ArrayAdapter 实现分类选择与新建。

```
private void initCategories() {
    mCategories.add("工作");
    mCategories.add("学习");
    mCategories.add("生活");
    mCategories.add("其他");

    ArrayAdapter<String> adapter = new ArrayAdapter<>( context: this,
        android.R.layout.simple_spinner_item,
        new ArrayList<>(mCategories));
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    mCategorySpinner.setAdapter(adapter);

    mCategorySpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        no usages
        ↗
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            mCurrentCategory = (String) parent.getItemAtPosition(position);
        }

        no usages
        ↗
        @Override
        public void onNothingSelected(AdapterView<?> parent) {
            mCurrentCategory = "其他";
        }
    });
}
```

3. 笔记列表

在进行笔记的新建和编辑后，在主界面中呈现笔记列表。在初始应用中背景颜色为白色，字体颜色为黑色，且笔记列表中的每个笔记显示标题和一部分内容，标题前面是分类情况，每个笔记都是一个单独的卡片，时间在右下角

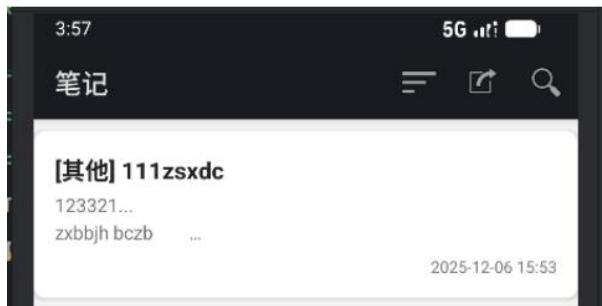


二. 拓展基本功能

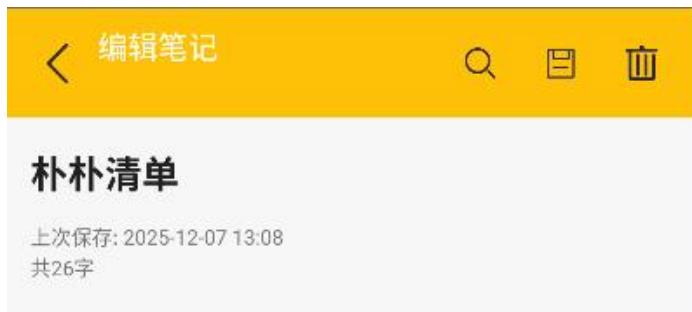
(一) . 笔记条目增加时间戳显示

1. 功能要求

每个新建笔记都会保存新建时间并显示；在修改笔记后更新为修改时间
右下角是时间，在主界面以“MM-dd HH:mm”格式显示



在编辑笔记页面保存后也会显示上次保存的时间



2. 实现思路和技术实现

(1) 初始应用的笔记列表 item 只有一个标题，需要再添加一个 TextView 用来显示时间，布局使用 LinearLayout

Todo: 这里截图有关时间戳实现的代码部分

a) 数据库表结构（时间戳字段）

在文件: NotePadProvider.java 中 DatabaseHelper.onCreate() 方法中的 SQL 建表语句

```
192
193 @Override
194 public void onCreate(SQLiteDatabase db) {
195     db.execSQL("CREATE TABLE " + NotePad.Notes.TABLE_NAME + "("
196             + NotePad.Notes._ID + " INTEGER PRIMARY KEY,"
197             + NotePad.Notes.COLUMN_NAME_TITLE + " TEXT,"
198             + NotePad.Notes.COLUMN_NAME_NOTE + " TEXT,"
199             + NotePad.Notes.COLUMN_NAME_CREATE_DATE + " INTEGER," // 创建时间字段
200             + NotePad.Notes.COLUMN_NAME_MODIFICATION_DATE + " INTEGER" // 修改时间字段
201             + ")");
202 }
```

b) 时间戳自动更新逻辑

在文件: NotePadProvider.java 中 insert() 和 update() 方法中的时间戳处理

在 insert() 中设置创建时间和修改时间为当前时间，在 update() 中更新修改时间为当前时间：

```

517
518     // Gets the current system time in milliseconds
519     Long now = Long.valueOf(System.currentTimeMillis());
520
521     // If the values map doesn't contain the creation date, sets the value to the current time.
522     if (values.containsKey(NotePad.Notes.COLUMN_NAME_CREATE_DATE) == false) {
523         values.put(NotePad.Notes.COLUMN_NAME_CREATE_DATE, now);
524     }
525
526     // If the values map doesn't contain the modification date, sets the value to the current
527     // time.
528     if (values.containsKey(NotePad.Notes.COLUMN_NAME_MODIFICATION_DATE) == false) {
529         values.put(NotePad.Notes.COLUMN_NAME_MODIFICATION_DATE, now);
530     }
531
532     // If the values map doesn't contain a title, sets the value to the default title.
533     if (values.containsKey(NotePad.Notes.COLUMN_NAME_TITLE) == false) {
534         Resources r = Resources.getSystem();
535         values.put(NotePad.Notes.COLUMN_NAME_TITLE, r.getString(android.R.string.untitled));
536     }

```

c) 时间戳自动插入逻辑

在文件: NotePadProvider.java 的 insert() 方法中的时间戳处理

```

518     // Gets the current system time in milliseconds
519     Long now = Long.valueOf(System.currentTimeMillis());
520
521     // If the values map doesn't contain the creation date, sets the value to the current time.
522     if (values.containsKey(NotePad.Notes.COLUMN_NAME_CREATE_DATE) == false) {
523         values.put(NotePad.Notes.COLUMN_NAME_CREATE_DATE, now); // 创建时间
524     }
525
526     // If the values map doesn't contain the modification date, sets the value to the current
527     // time.
528     if (values.containsKey(NotePad.Notes.COLUMN_NAME_MODIFICATION_DATE) == false) {
529         values.put(NotePad.Notes.COLUMN_NAME_MODIFICATION_DATE, now); // 修改时间
530     }

```

d) 时间戳显示格式化

在 NotesList.java 的 updateListAdapter() 中从 Cursor 获取时间戳并格式化

```

private void updateListAdapter(Cursor cursor) {
    String[] dataColumns = { NotePad.Notes.COLUMN_NAME_TITLE };
    int[] viewIDs = { android.R.id.text1 };

    SimpleCursorAdapter adapter = new SimpleCursorAdapter(
        context: this,
        R.layout.noteslist_item,
        cursor,
        dataColumns,
        viewIDs
    );
}

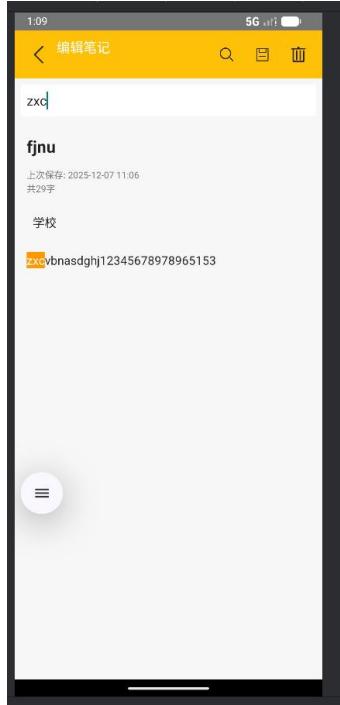
```

二). 笔记查询功能 (按标题查询)

1. 功能要求: 点击搜索按钮展开搜索框

实时搜索标题和内容

搜索结果高亮显示



2. 实现思路和技术实现

Todo: 这里截图有关搜索功能实现的代码部分

a) 搜索菜单项配置

在文件: list_options_menu.xml 中添加 SearchView 菜单项

```
24      <!-- 搜索 -->
25      <item
26          android:id="@+id/menu_search"
27          android:title="搜索"
28          android:icon="@android:drawable/ic_menu_search"
29          app:actionViewClass="android.widget.SearchView"
30          app:showAsAction="collapseActionView|always" />
31  </menu>
```

b) 搜索字符串资源

在文件: strings.xml 中添加 search_hint 字符串

```
<string name="menu_paste">粘贴</string>
<string name="menu_search">搜索</string>
<string name="menu_set_category">设置分类</string>
<string name="menu_filter_category">分类筛选</string>
<!-- 其他 -->
```

c) 搜索功能核心实现

在文件: NotesList.java 中, 配置 SearchView 监听器, 并且根据输入构建查询条件, 执行 managedQuery 并更新适配器

```
private void setupSearchView(SearchView searchView) {
    if (searchView == null) return;

    // 设置搜索提示
    searchView.setQueryHint(getString(R.string.search_hint));
    // 默认展开搜索框
    searchView.setIconifiedByDefault(false);
    // 获取SearchView中的搜索编辑框并配置输入属性
    int searchPlateId = searchView.getContext().getResources()
        .getIdentifier("name: \"android:id/search_src_text\"", "defType: null", "defPackage: null");
    if (searchPlateId != 0) {
        View searchPlate = searchView.findViewById(searchPlateId);
        if (searchPlate instanceof android.widget.EditText) {
            android.widget.EditText searchEditText = (android.widget.EditText) searchPlate;
            // 移除任何可能限制输入的过滤器
            searchEditText.setFilters(new android.text.InputFilter[0]);
        }
    }
}
```

设置搜索文本监听器：

```
// 设置搜索文本监听器
searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    no usages
    ↴
    @Override
    public boolean onQueryTextSubmit(String query) {
        // 用户点击键盘搜索按钮时执行
        performSearch(query);
        searchView.clearFocus();
        return true;
    }
    no usages
    ↴
    @Override
    public boolean onQueryTextChange(String newText) {
        // 文本变化时执行，带有防抖处理
        handleSearchTextChange(newText);
        return true;
    }
});
// 监听搜索关闭事件
searchView.setOnCloseListener(new SearchView.OnCloseListener() {
    no usages
    ↴
    @Override
    public boolean onClose() {
        clearSearch();
        return false; // 返回false让系统处理默认行为
    }
});
// 监听搜索展开事件
searchView.setOnSearchClickListener(new View.OnClickListener() {
    ↴
    @Override
    public void onClick(View v) {
        mIsSearchMode = true;
    }
});
```

三. 拓展附加功能

(一) .UI 美化

1. 功能要求

记事本以黄色为主题，每条笔记的卡片 可以按分类给颜色，比如效果：工作=蓝色，生活=绿色，学习=黄色等

主界面的每条笔记是单独的一个圆角卡片，布局不会紧凑，看起来更简洁，如图



2. 实现思路和技术实现

(1) 主题颜色定义

在文件: colors.xml 中，定义主色调 colorPrimary、colorPrimaryDark 等

```

10
11      <!-- 记事本应用自定义颜色 -->
12      <color name="colorPrimary">#FF9800</color>
13      <color name="colorPrimaryDark">#F57C00</color>
14      <color name="colorAccent">#FF9800</color>
15      <color name="background">#FFFFFF</color>
16      <color name="card_background">#FFFFFF</color>
17      <color name="text_primary">#212121</color>
18      <color name="text_secondary">#757575</color>
19      <color name="divider">#E0E0E0</color>
20      <color name="category_tag">#FF9800</color>
21      <!-- fab_color --> #FF9800
22  
```

(2) 样式定义

在 styles.xml 中定义 AppTheme、CardStyle、CategoryTag 样式

```

<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
    <item name="android:windowBackground">@color/background</item>
</style>

<style name="CardStyle">
    <item name="android:background">@drawable/bg_item_card</item>
    <item name="android:padding">16dp</item>
    <item name="android:layout_margin">8dp</item>
</style>

<style name="CategoryTag">
    <item name="android:background">@drawable/bg_category_tag</item>
    <item name="android:textColor">@color/white</item>
    <item name="android:textSize">12sp</item>

```

(3) 卡片布局

在文件: noteslist_item.xml 中使用 CardView 实现圆角阴影效果

```

<ListView
    android:id="@+id/list"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:divider="@null"
    android:dividerHeight="0dp"
    android:background="#FAFAFA" />

```

(4) 分类颜色标记

实现: 在适配器中根据分类设置 category TextView 背景色

(二). 笔记分类:

- 在笔记编辑界面, 点击分类的下拉框, 可以选择系统已有的生活, 学习, 工作等分类, 也可以自己新建分类,



保存后在主界面的每一条笔记的标题前面可以看到分类情况



2. 实现步骤

(1) 数据库添加分类字段

在文件: NotePadProvider.java 的 DatabaseHelper.onCreate() 方法中添加 category TEXT
DEFAULT '其他'

```
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE " + NotePad.Notes.TABLE_NAME + " (" +
                + NotePad.Notes._ID + " INTEGER PRIMARY KEY, " +
                + NotePad.Notes.COLUMN_NAME_TITLE + " TEXT, " +
                + NotePad.Notes.COLUMN_NAME_NOTE + " TEXT, " +
                + NotePad.Notes.COLUMN_NAME_CREATE_DATE + " INTEGER, " +
                + NotePad.Notes.COLUMN_NAME_MODIFICATION_DATE + " INTEGER, " +
                + "category TEXT DEFAULT '其他'" // 添加分类字段
                + ")");
    }
}
```

(2) 编辑界面分类选择器

在文件: note_editor.xml 中实现 Spinner 控件 @+id/spinner_category

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="分类:"
    android:textSize="16sp"
    android:textStyle="bold" />

<Spinner
    android:id="@+id/spinner_category"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp" />

<Button
    android:id="@+id/btn_new_category"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/category_new" />
</LinearLayout>
```

(3) 分类初始化与新建

在文件: NoteEditor.java 中创建 initCategories()、showNewCategoryDialog()方法
可以实现: 初始化默认分类集合, 设置 Spinner 适配器, 弹出对话框新建分类

```
private void initCategories() {
    mCategories.add("工作");
    mCategories.add("学习");
    mCategories.add("生活");
    mCategories.add("其他");

    ArrayAdapter<String> adapter = new ArrayAdapter<>( context: this,
        android.R.layout.simple_spinner_item,
        new ArrayList<>(mCategories));
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    mCategorySpinner.setAdapter(adapter);

    mCategorySpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        no usages
        ↴
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            mCurrentCategory = (String) parent.getItemAtPosition(position);
        }

        no usages
        ↴
        @Override
        public void onNothingSelected(AdapterView<?> parent) {
            mCurrentCategory = "其他";
        }
    });
}
```

新建分类：

```
private void showNewCategoryDialog() {
    EditText input = new EditText( context: this);
    new AlertDialog.Builder( context: this)
        .setTitle("新建分类")
        .setView(input)
        .setPositiveButton( text: "确定", ( DialogInterface dialog, int which) -> {
            String newCategory = input.getText().toString().trim();
            if (!newCategory.isEmpty() && !mCategories.contains(newCategory)) {
                mCategories.add(newCategory);
                initCategories(); // 重新初始化分类列表
                mCategorySpinner.setSelection((ArrayAdapter<String>)mCategorySpinner.getAdapter()
                    .getPosition(newCategory));
            }
        })
        .setNegativeButton( text: "取消", listener: null)
        .show();
}
```

(4) 保存分类到数据库

在文件： NoteEditor.java 中新建方法 saveNote()

最后可以将 mCurrentCategory 保存到数据库

```
private void saveNote() {
    String title = mTitleText.getText().toString();
    String content = mText.getText().toString();

    if (title.isEmpty() && content.isEmpty()) {
        Toast.makeText(context, text: "笔记内容为空", Toast.LENGTH_SHORT).show();
        return;
    }

    ContentValues values = new ContentValues();
    values.put(NotePad.Notes.COLUMN_NAME_TITLE, title);
    values.put(NotePad.Notes.COLUMN_NAME_NOTE, content);
    values.put(NotePad.Notes.COLUMN_NAME_MODIFICATION_DATE, System.currentTimeMillis());

    // |分类保存
    values.put(NotePad.Notes.COLUMN_NAME_CATEGORY, mCurrentCategory);

    getContentResolver().update(mUri, values, where: null, selectionArgs: null);
    updateTime();
}
```