1 软件名称

完成人：钟昌宏

学号：20152100127

完成时间：2017/12/7

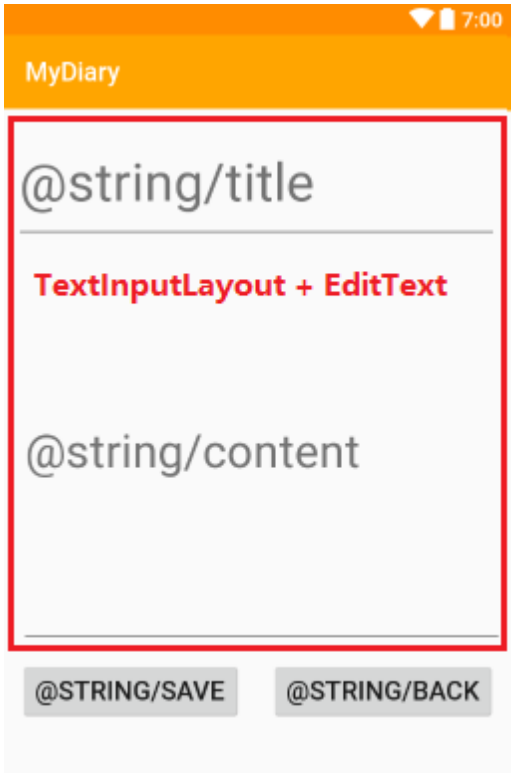
2 软件内容简介

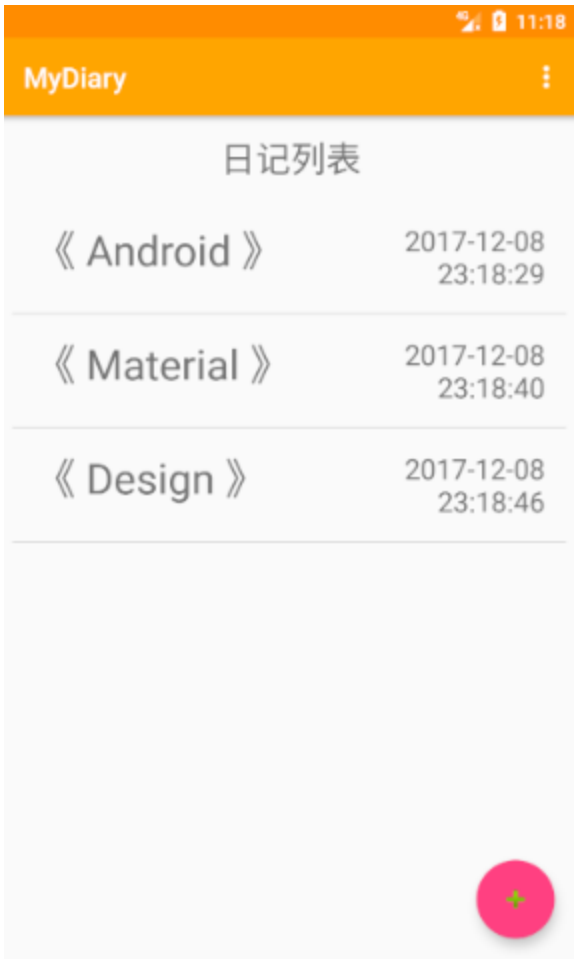
一个简单的“日记本”App，采用 SQLite 数据库进行信息存储，实现日记的浏览、新建、编辑、删除等功能，有友好的交互界面。

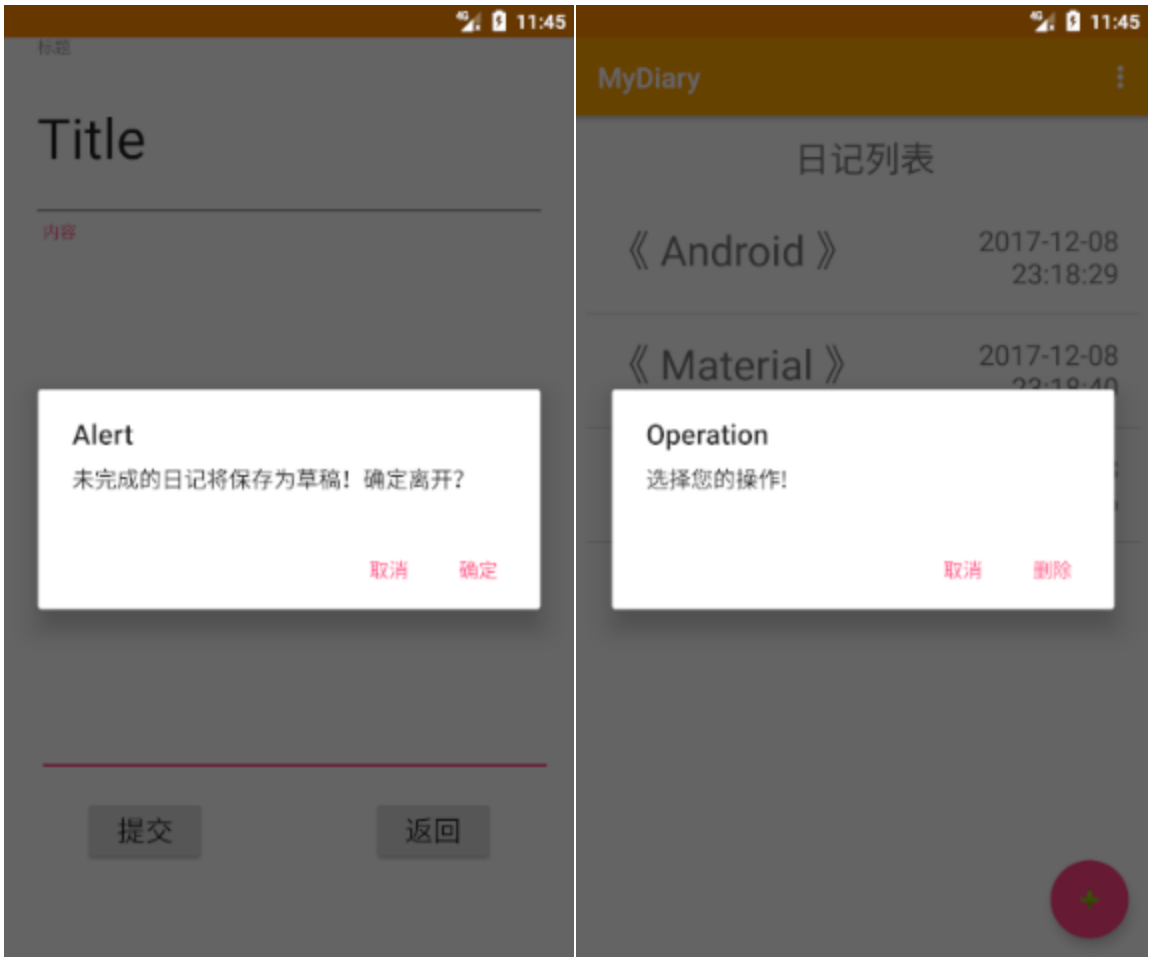
3 界面设计

主界面

编辑日记界面


7:00
MyDiary
@string/title
TextInputLayout + EditText
@string/content
@STRING/SAVE
@STRING/BACK


11:18
MyDiary
日记列表
《 Android 》
2017-12-08
23:18:29
《 Material 》
2017-12-08
23:18:40
《 Design 》
2017-12-08
23:18:46



4 代码设计

4.1 RecyclerView 的设计

新建一个 layout 作为 RecyclerView 的样式。在本 App 中，RecyclerView 的一个 Item 中需要显示日记标题、日期、具体时间，所以在 layout 文件中只需设计三个 TextView 按照上面的布局摆放。这里需要注意

的是，控件放在 FrameLayout 里，可以用 android:gravity 来指定摆放位置。

日记标题：`android:gravity="start"`

日期：`android:gravity="end"`

时间：`android:gravity="end|bottom"`

在 MainActivity 添加如下函数，自定义 Adapter：

```
private class NormalRecyclerViewAdapter extends RecyclerView.Adapter<NormalRecyclerViewAdapter.MyTextViewHolder>
{
    private final LayoutInflater mLayoutInflater = LayoutInflater.from(getApplicationContext());
    private MyOnItemClickListener itemClickListener;
    private MyOnItemLongClickListener itemLongClickListener;
    @Override
    public MyTextViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        return new MyTextViewHolder(mLayoutInflater.inflate(R.layout.recyclerview_style, parent, false));
    }
    public int getItemCount() { return myTitles.size(); }
    public void onBindViewHolder(final MyTextViewHolder holder, int position) {
        /*将接收到的ViewHolder强转成自定义的VIewHolder*/
        //final MyTextViewHolder myViewHolder = (MyTextViewHolder) holder;
        String text1 = myTitles.get(position);
        String temp = myTimes.get(position);
        String text2 = temp.substring(0, temp.indexOf(" "));//yyyy-mm-dd
        String text3 = temp.substring(temp.indexOf(" "), temp.length());//hh-mm-ss
        holder.tv.setText("《 "+ text1+" 》");
        holder.date.setText(text2);
        holder.time.setText(text3);
```

```
class MyTextViewHolder extends RecyclerView.ViewHolder
{
    TextView tv;
    TextView date;
    TextView time;
    public MyTextViewHolder(View view)
    {
        super(view);
        tv = (TextView) view.findViewById(R.id.textViewData);
        date = (TextView) view.findViewById(R.id.textViewDate);
        time = (TextView) view.findViewById(R.id.textViewTime);
    }
}
```

（此类的定义放在 Adapter 里）

最后是 onCreate 函数里：

```
myRecyclerViewList = (RecyclerView) findViewById(R.id.recyclerViewList);
myRecyclerViewList.setAdapter(adapter);
```

4.2 RecyclerView 点击事件

在 RecyclerView 中不像 ListView 一样有点击事件的监听器，因此我们需要自己动手处理点击事件。

首先新建两个类，分别是点击和长按：

```
/**
 * item点击接口
 */
public interface MyOnItemClickListener {
    void OnItemClick(View view, int position);
}
```

```
public interface MyOnItemLongClickListener
{
    void OnItemLongClick(View view, int position);
}
```

接下来在 Adapter 里新建两个私有变量用于保存用户设置的监听及其 set 方法：

```
private MyOnItemClickListener itemClickListener;
private MyOnItemLongClickListener itemLongClickListener;
```

```
public void setOnItemClickListener(MyOnItemClickListener itemClickListener) {
    this.itemClickListener = itemClickListener;
}
public void setOnItemLongClickListener(MyOnItemLongClickListener itemLongClickListener){
    this.itemLongClickListener = itemLongClickListener;
}
```

在 Adapter 里的 onBindViewHolder 方法内，实现回调：

```
/*自定义item的点击事件不为null，设置监听事件*/
if (itemClickListener != null) {
    holder.itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            itemClickListener.OnItemClick(holder.itemView, holder.getLayoutPosition());
        }
    });
}//
if(itemLongClickListener != null){
    holder.itemView.setOnLongClickListener(new View.OnLongClickListener() {
        @Override
        public boolean onLongClick(View v) {
            itemLongClickListener.OnItemLongClick(holder.itemView,
                    holder.getLayoutPosition());
            return true;
        }
    });
}
```

最后在 onCreate 函数中调用设置好的监听器：

```
adapter.setOnItemClickListener(new MyOnItemClickListener() {
    @Override
    public void OnItemClick(View view, int position) {
        //实现点击功能代码
    }
});
```

```
adapter.setOnItemLongClickListener(new MyOnItemLongClickListener() {
    @Override
    public void OnItemLongClick(View view, int position) {
        //实现长按功能代码
    }
});
```

4.3 SQLite 数据库设计

在本 App 中，创建了一个数据库两张表格，分别是 data 表和 drafts 表，data 表里列有 id，title，content，time，分别保存 id，标题、内容、时间；drafts 表列有 id，title，content，分别保存 id，标题、内

容。

```
private class MySqlHelper extends SQLiteOpenHelper{
    public MySqlHelper(Context context) { super(context, "diary.db", null, 2); }
    public void onCreate(SQLiteDatabase db){
        //db.execSQL("delete from data");
        String sql = "CREATE TABLE data(_id INTEGER PRIMARY KEY AUTOINCREMENT," +
                " title VARCHAR(20), content VARCHAR(200), time VARCHAR(20))";
        db.execSQL(sql);

        String draft = "CREATE TABLE drafts(_id INTEGER PRIMARY KEY AUTOINCREMENT," +
                " title CARCHAR(20), content VARCHAR(200))";
        db.execSQL(draft);
    }
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion){}
    public void onOpen(SQLiteDatabase db) { super.onOpen(db); }
}
```

在做本 App 过程中遇到一个问题，一开始在 data 数据库中没有 time 这一列，后来加上了，但是发现无论怎么执行代码，都会抛出异常说找不到 time 这一列。后来查资料知道，需要卸载 App 再安装，才会重新执行新建数据库的函数，修改的代码才能执行，这个数据库在安装 App 的时候就已经建立好了，在运行 App 的时候一般是不能去修改表结构的。

4.4 保存为草稿（返回数据的 Intent）

本 App 的页面跳转很简单，从主界面点击添加日记，进入日记编辑界面，最终无论是提交日记，还是返回，都要从日记编辑界面回到主界面，回到主界面是把写好的日记提交，还是保存为草稿呢？这里就涉及到数据返回

的问题，于是这里需要通过 Intent 对象返回对象给上一级。

```
public void onClick(View view) {
    Intent intent = new Intent(MainActivity.this, EditDiaryPage.class);
    intent.putExtra("title", draftTitle);
    intent.putExtra("content", draftContent);
    startActivityForResult(intent, 1);
}
```

这里用到了 startActivityForResult 函数，第一个参数是 Intent 对象，第二个参数是 requestCode 请求码，理解为向下一个页面请求返回数据，requestCode 可以作为返回的数据进行认证，哪一些数据是给“我”的。

在子 Intent 里添加以下代码：

```
Intent intent = new Intent(ShowDiaryPage.this, MainActivity.class);
intent.putExtra("editedTitle", title);
intent.putExtra("editedContent", content);
intent.putExtra("editedTime", time);
intent.putExtra("index", index);
setResult(1, intent);//should save
finish();
```

需要返回的数据

resultCode

resultCode 是结果码，用于告知用户执行了什么操作得到了什么结果，比如按了确定，按了取消等等。如果代码要规范一点的话，requestCode 和 resultCode 最好都用全局变量来代替，比如 Request_Operation，Result_OK，Rresult_Cancel，这样比较明了。但是本 App 就省略了这一步，因为操作很少。

返回了数据后，主界面又该如何接收这些数据呢？这就需要重写 onActivityResult 函数。

```
protected void onActivityResult(int requestCode, int resultCode, Intent data){
    super.onActivityResult(requestCode, resultCode, data);
    switch(requestCode){
        case 1:{//请求码为1
            if(resultCode == 1){//结果码为1 submit
                //执行代码
            }//if
            else if(resultCode == 2){//结果码为2 save to draft
```

代码中先判断 requestCode，再判断 resultCode。保存草稿的话，返回标题和内容，存储在 drafts 表里，每次启动 App 先读取表内数据，在添加日记的时候选择传入未完成的日记，即可做到保存草稿功能。

4.5 提交日记并刷新 RecyclerView 数据

提交日记返回数据的思路和上面保存草稿的一样，都是通过 Intent 返回数据，关键是存储进数据库并使得 RecyclerView 刷新。存储进数据库的部分就不累述，因为上次的作业就是实现数据库的存储读取，已经较为熟练，下面重点说一下 RecyclerView。

函数	作用
notifyItemChanged(int position)	position 位置的数据改变
notifyItemInserted(int position)	在 position 插入了数据，原本在 position 的数据移到 position +1
notifyItemRemoved(int position)	与上面的函数相反
notifyItemMoved(int from, int to)	From 位置数据移到了 to
notifyItemRangeChanged(int start,int count)	从 start 位置开始，后面 count 个数据改变了

以上是比较常见的更新函数，还有很多参数的变化在此不一一列出，可通过安卓官方文档查询。

4.6 编辑、修改、删除日记

用户在点击 RecyclerView 的 Item 后会进入日记的修改页面，大致的页面和编辑日记的页面一样，只不过变成了保存。这个功能也不是非常难，主要是要思考通过什么来唯一确定数据库中的行，以至于能够更新数据库，我用的是编辑时间。时间是不可能重复的，所以可以作为唯一确定的元素。在编辑日记之前，先获得用户选择修改的日记的保存时间，接着进入修改日记界面。保存之后，通过旧时间找到数据库中的一行，更新标题、内容和时间即可，最终在列表中再刷新一次，即可完成日记的编辑和修改。

```
/*
* Update database
* */
private boolean updateDatabase(String t, String c, String d, String ot)//oldTime
{
    SQLiteDatabase db = helper.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put("title", t);
    values.put("content", c);
    values.put("time", d);
    try {
        db.update("data", values, "time=?", new String[]{ot});
    }catch(Exception e){
        e.printStackTrace();
        return false;
    }
    db.close();
    return true;
}
```

删除日记也是一样的，我做的是长按日记跳出对话框，选择删除或取消，这里就涉及到 AlertDialog 的使用了，本 App 中多次用到了它，在写日记未提交时点击返回时会弹框提醒，在删除时会弹框提醒。

要实现它并不难，只需要添加以下代码：

```
private AlertDialog alert;
```

```
/*
* Set AlertDialog
* */
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setPositiveButton("删除", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        //点击删除的代码
    }
});
builder.setNegativeButton("取消", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {}
});
alert = builder.create();
alert.setTitle("Operation");
alert.setMessage("选择您的操作!");
```

通过 AlertDialog.Builder 来创建，设置好按钮的功能，最后 Builder.create()实例化。要使用的时候执行以下代码：

```
alert.show();
```

4.7 RecyclerView 原生 Bug 的解决

在实现 RecyclerView 的刷新功能的时候遇到一个 Bug 无法解决，一刷新就抛出以下异常：

```
java.lang.IndexOutOfBoundsException: Inconsistency detected. Invalid view holder
adapter positionViewHolder{431a7450 position=1 id=-1, oldPos=-1, pLpos:-1 scrap
[attachedScrap] tmpDetached no parent}
        at
android.support.v7.widget.RecyclerView$Recycler.validateViewHolderForOffsetPosition(Re
cyclerView.java:4251)
```

上网查阅了非常多的资料，了解到这个 RcyclerView 的一个原生 Bug，最简单的解决办法是写一个继承 LinearLayoutManager 的类，在 onLayoutChildren()方法里 try-catch 捕获该异常。

```
public class WrapContentLinearLayoutManager extends LinearLayoutManager {  继承 LinearLayoutManager
    //... constructor
    public WrapContentLinearLayoutManager(Context context) { super(context); }
    public WrapContentLinearLayoutManager(Context context, int orientation, boolean reverseLayout){
        super(context, orientation, reverseLayout);
    }
    @Override
    public void onLayoutChildren(RecyclerView.Recycler recycler, RecyclerView.State state) {
        try {
            super.onLayoutChildren(recycler, state);
        } catch (IndexOutOfBoundsException e) {  捕获异常
            Log.e("probe", "meet a IOOBE in RecyclerView");
        }
    }
}
```

不要忘了在 onCreate 函数里初始化 RecyclerView 的时候设置对应的 LayoutManager：

```
myRecyclerViewList.setLayoutManager(new WrapContentLinearLayoutManager(this,
        LinearLayoutManager.VERTICAL, false));// fix a bug in RecyclerView
```

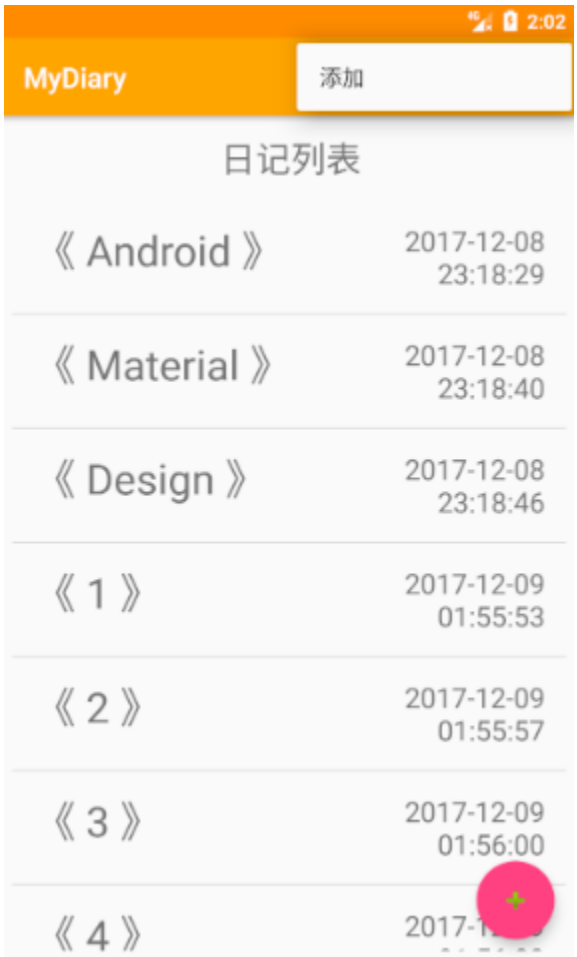
当然还有其他办法去解决这个问题，再次不展开讨论，详见下面的参考地址：

http://www.jianshu.com/p/2eca433869e9

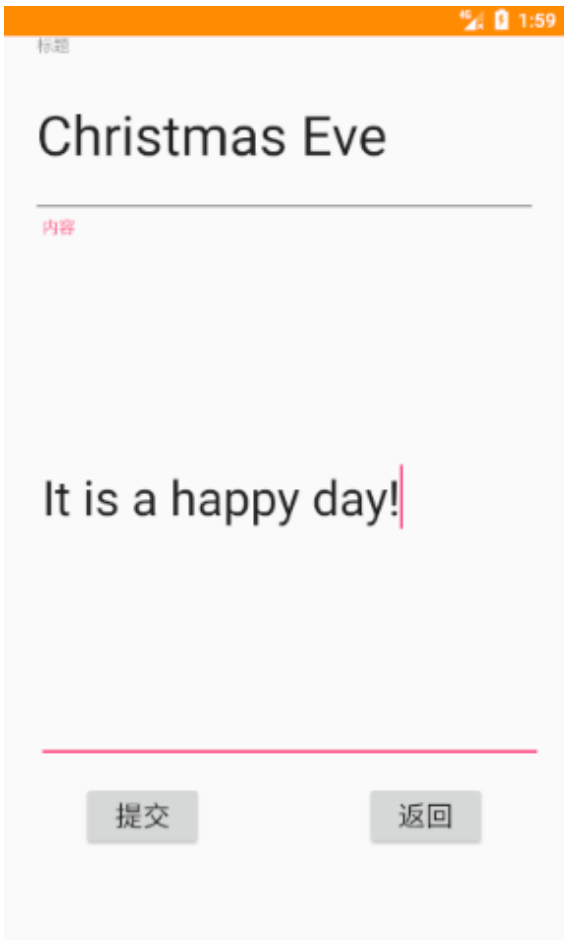
https://stackoverflow.com/questions/31759171/recyclerview-and-java-lang-indexoutofboundsexception-inconsistency-detected-in

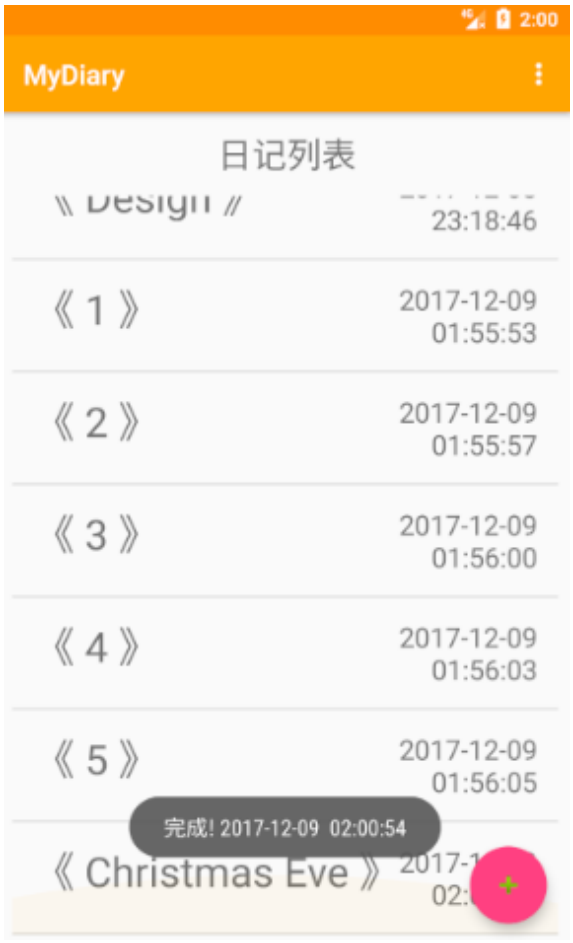
5 软件操作流程

5.1 主界面



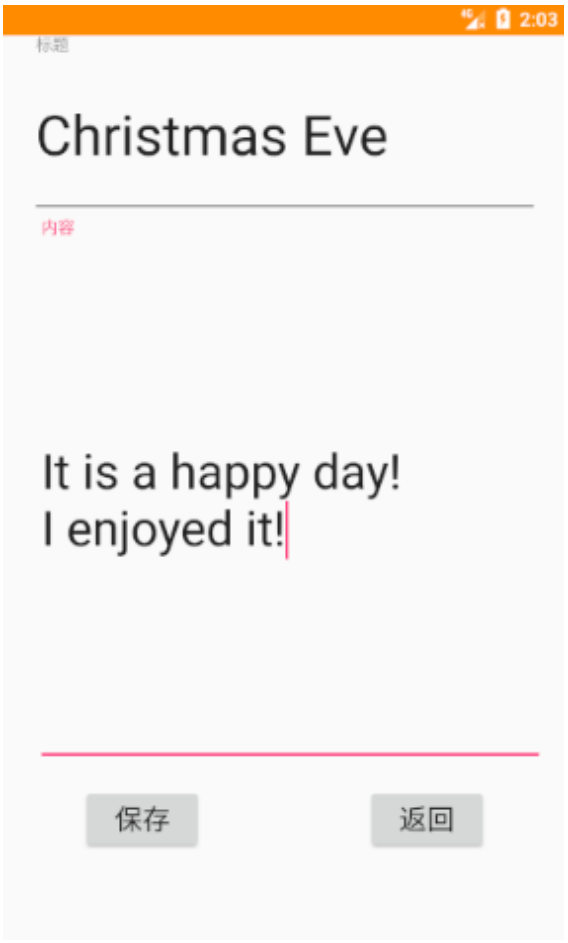
5.2 写日记并保存

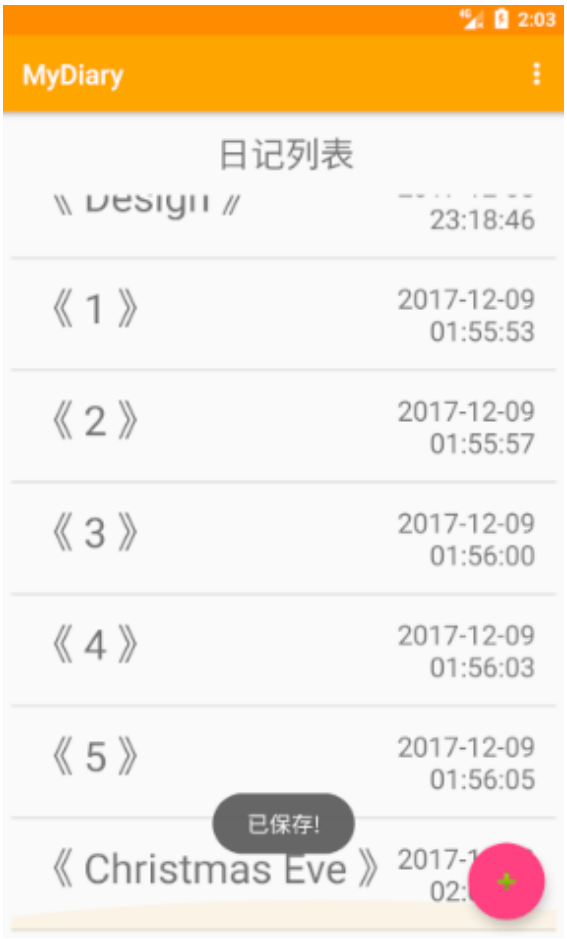

1:59
标题
Christmas Eve
内容
It is a happy day!
提交
返回


2:00
MyDiary
日记列表
23:18:46
《 1 》
2017-12-09
01:55:53
《 2 》
2017-12-09
01:55:57
《 3 》
2017-12-09
01:56:00
《 4 》
2017-12-09
01:56:03
《 5 》
2017-12-09
01:56:05
完成! 2017-12-09 02:00:54
《 Christmas Eve 》

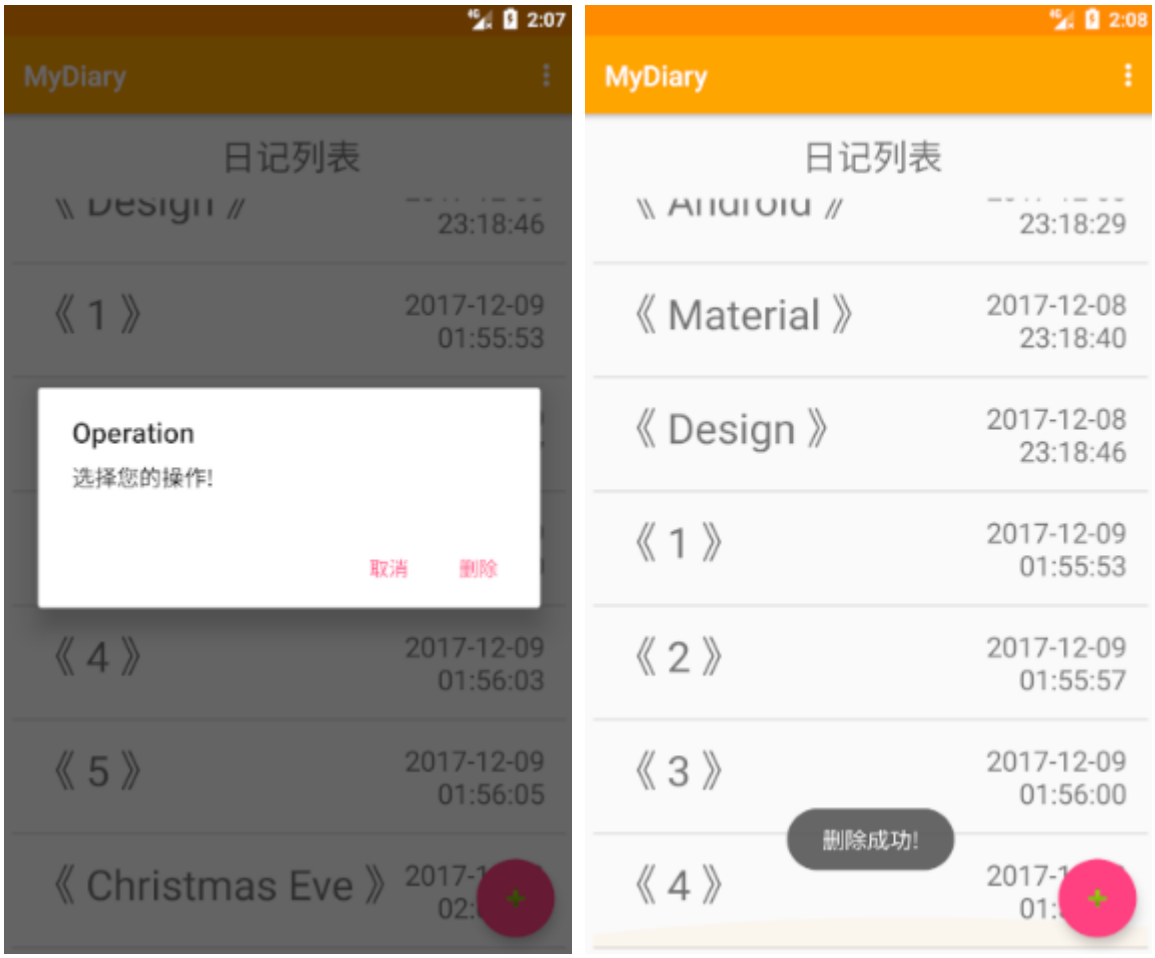
5.3 保存草稿

5.4 修改日记


2:03
标题
Christmas Eve
内容
It is a happy day!
I enjoyed it!
保存
返回


2:03
MyDiary
日记列表
23:18:46
《 1 》
2017-12-09
01:55:53
《 2 》
2017-12-09
01:55:57
《 3 》
2017-12-09
01:56:00
《 4 》
2017-12-09
01:56:03
《 5 》
2017-12-09
01:56:05
已保存!
《 Christmas Eve 》

5.5 删除日记



6 难点和解决方案

6.1 RecyclerView 的数据刷新

数据刷新花了一些时间来搞清楚区别，中途查阅了很多资料，尝试了几种方法。还有很多复杂的用法我没有探究到，希望在以后的开发过程中能弄懂。

6.2 Intent 的数据返回

写数据返回时思路有一点不清晰，逻辑有点混乱，搞不清楚 requestCode 和 resultCode 的却别，看了很久才理解其中的原理。最终发现官方文档才是最好的资料。

6.3 复杂的数据存储和传输

在数据库与类对象之间有很多数据传输，一旦日记有信息更改，就要马上写入数据库，否则可能数据丢失；一旦更新了数据库，类对象里的数据也要相应更新，否则可能导致数据库里的内部数据和外部数据不一致，最终存储错乱；数据更新了，相应地在控件里也要更新。

在打代码的时候，遇到了很多次数据错乱的问题，数据库里的数据和显示出来的数据不一致，此时必须清空数据库里的表，否则会一直错下去。对数据库进行添加、更新、删除操作都必须小心翼翼。

7 不足之处和今后设想

7.1 批量删除

本 App 还没有批量删除的功能。

7.2 上传照片

日记里要是能上传照片或者小贴图的话，就非常好了，是一个比较创新的功能。