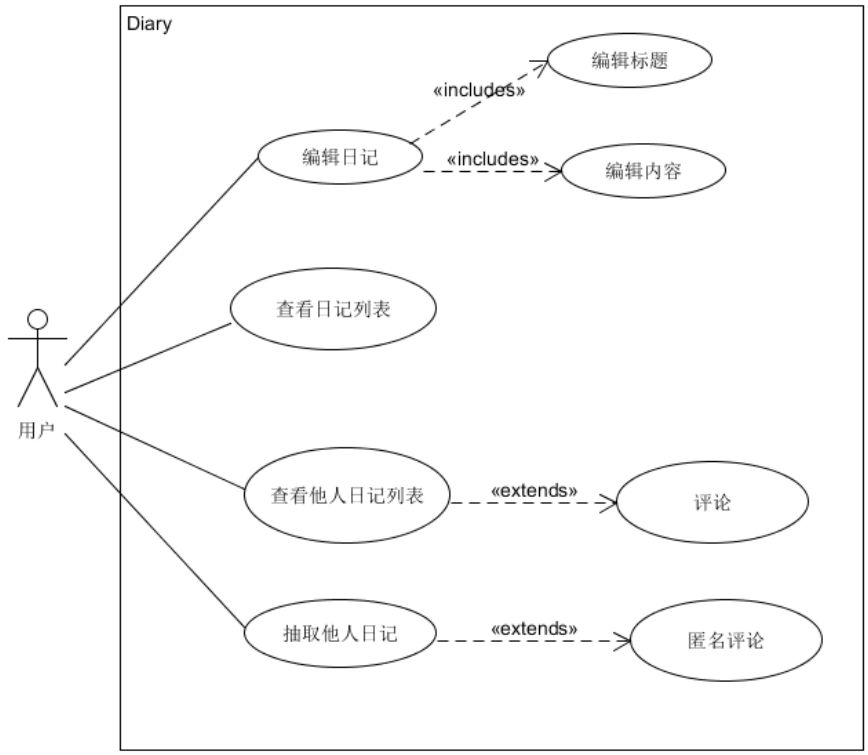
软件设计文档

中山大学

LiekkasNote 小组

1 模型图及其说明

1.1 用例图

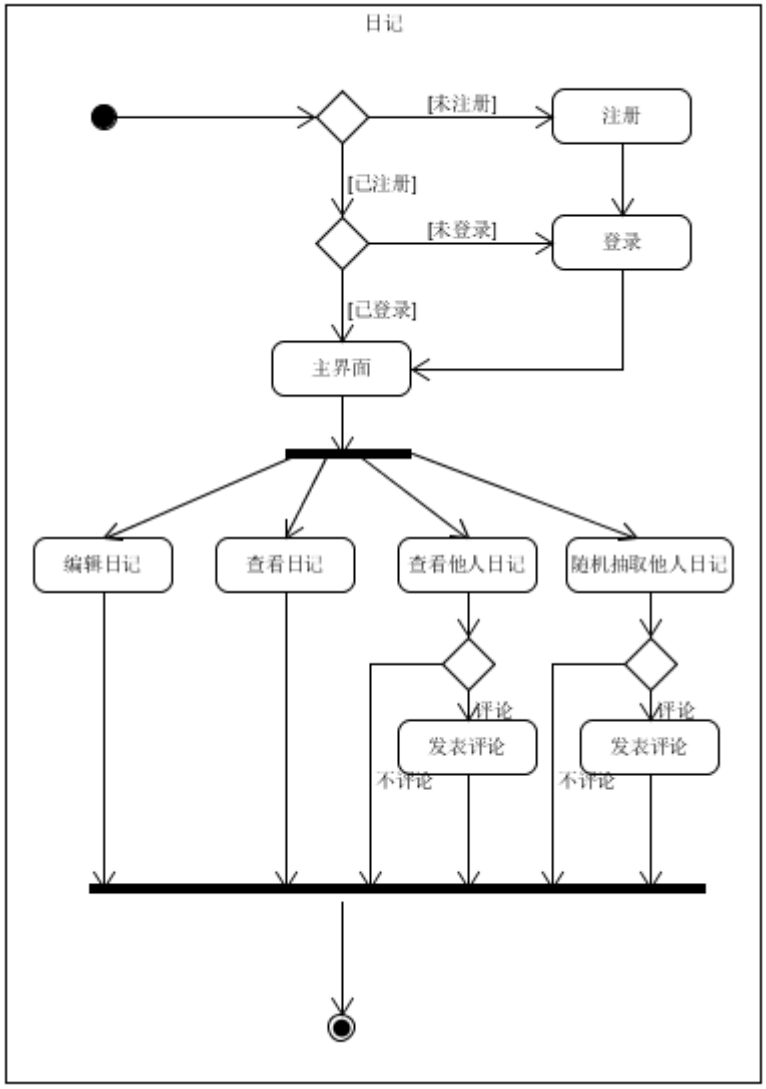


本项目中有四个主用例，包括了编辑日记、查看日记列表、查看他人日记列表、抽取他人日记的四个用例。

编辑日记主要有两个步骤，即编辑标题、编辑内容，然后提交。

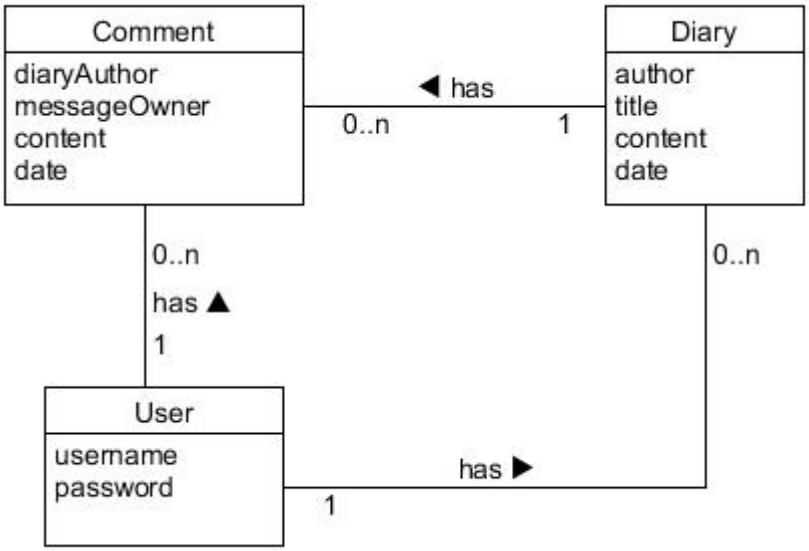
查看日记列表主要就是可以查看自己之前的日记总数，及点击以往日记进行查看。
查看他人日记列表是可以查看自己以往抽取他人日记的列表。
抽取他人日记是的步骤主要是可以随机从服务器抽取他人的日记，并进行评论。

1.2 活动图



活动图根据整个项目的用例，描述了项目的整个活动流程，大致说明了整个项目的逻辑和包括了分支逻辑。活动图能够帮助我们更好地理解整个项目的活动流程。

1.3 领域模型



领域模型描述了各个类初步的属性和相应的联系。虽然领域图并不准确，但是通过绘制领域模型可以帮助我们理清各个类的成员变量，对我们的开发过程也会有很大的帮助。绘制领域模型，可以防止我们造成数据库中的数据重复，同时也会让业务逻辑更加清晰，这对后续数据库的开发和代码的开发会有很大的帮助。

2 技术选型理由

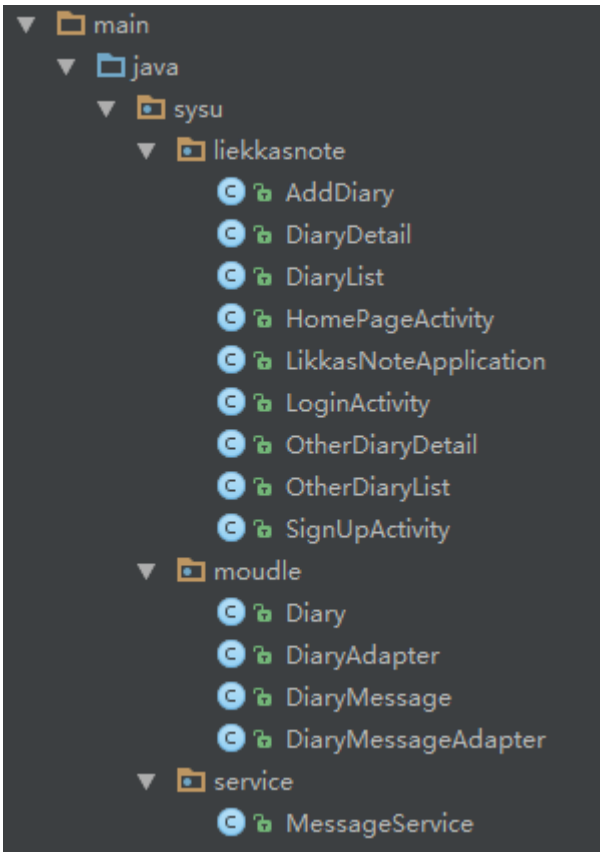
在这个项目中，我们采用了 Android 应用开发技术，理由如下：

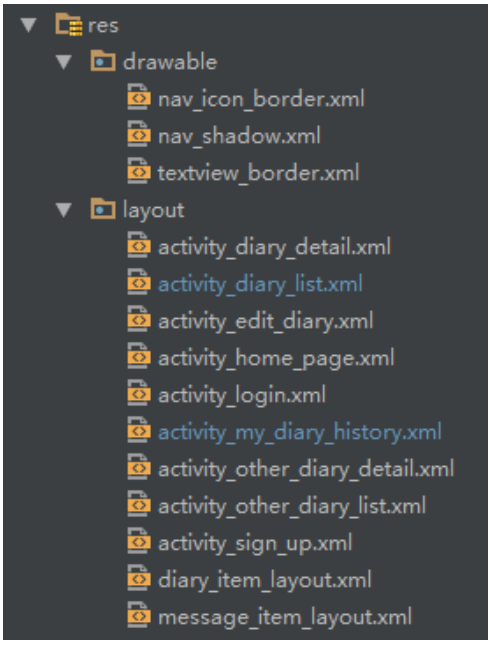
- Android 系统是市场上两大手机系统之一，份额位居全球第一。
- 对比 ios 或者其他应用系统，android 平台拥有他特有的开放性，开源这个特点吸引了诸多的开发者。
- Android 平台也提供给了第三方开发商一个自由的环境，没有各种条条框框的约束，更易于开发
- 小组成员基本都修过《安卓手机应用开发》课程，并且都对安卓开发有广泛的兴趣。

3，设计技术

① 结构化编程

将软件划分成为不同的模块，降低了耦合性，容易维护和扩展。整个软件依次分为 Activity 模块，Module 模块，Service 模块，Layout 模块以及 Resource 模块。





② 面向对象编程

在明确了需求了，我们将关注点放在了领域模型的设计当中，领域模型的设计很好地体现了面向对象编程的思想，程序中的类来自于对现实生活中的概念的抽象。

<1> Diary 类

```
public class Diary implements Serializable {
    public String author;   // 用户
    public String title;    // 日记标题
    public String content;  // 日记内容
    public String objectId; // 唯一标识符 由数据库自动生成
    public Date date; // 日期

    public Diary(String author, String title, String content, String objectId, Date date) {
        this.author = author;
        this.title = title;
        this.content = content;
        this.objectId = objectId;
        this.date = date;
    }

}
```

<2> DiaryMessage 类

```java
public class DiaryMessage implements Serializable {
    public String author;  // 日记作者
    public String diaryId;  // 该日记在数据库中的ID
    public String messageOwner; // 评论的用户的ID
    public String messageContent; // 评论的内容
    public String objectId;   // 该评论在数据库中的ID
    public Date date;  // 日期

    public DiaryMessage(String author, String diaryId, String messageOwner, String messageContent, String objectId, Date date) {
        this.author = author;
        this.diaryId = diaryId;
        this.messageOwner = messageOwner;
        this.messageContent = messageContent;
        this.objectId = objectId;
        this.date = date;
    }
}
```

③ 设计模式

自适应界面布局

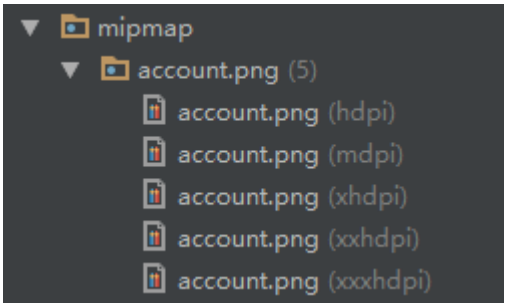
关于 UI 控件

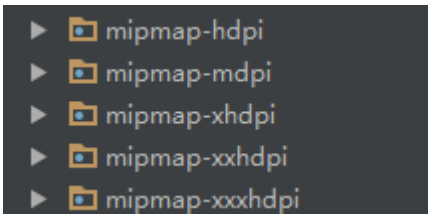
为了实现自适应布局，UI 控件的尺寸必须使用 dp 来定义，而不是 px，这是因为 px 是指像素，而不同设备之间的显示设备的像素往往不一样，如果使用 px 来定义就不能实现自适应的界面，因此必须使用 dp，这是因为 dp 在其意义上不依赖于像素的大小

```xml
<ImageView
    android:layout_width="150dp"
    android:layout_height="100dp"
    android:src="@mipmap/logo" />
```

```xml
<LinearLayout
    android:layout_marginLeft="35dp"
    android:layout_marginRight="35dp"
    android:id="@+id/login_message_password"
    android:layout_below="@+id/login_message_username"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="50dp">
```

除此之外，对于 App 中要使用到的图片资源，也需要备份多份，并且根据其尺寸的不同大小存放在 drawable-ldpi, drawbale-mdpi, drawable-hdpi, drawable-xhdpi, drawable-xxhdpi 这五个不同的文件夹当中





最后，多用 match-parent 以及 warp-content，以及使用相对布局，可以使得 UI 界面达到自适应的效果。

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/app_title"
    android:textSize="24dp"
    android:layout_marginBottom="20dp" />
```

适配器模式

```
public class DiaryAdapter extends BaseAdapter {
    public ArrayList<Diary> diaryArrayList;
    public Context context;


    public DiaryAdapter(Context context) {
        this.context = context;
        diaryArrayList = new ArrayList<Diary>();
    }

    @Override
    public int getCount() { return diaryArrayList.size(); }

    @Override
    public Object getItem(int i) { return diaryArrayList.get(i); }

    @Override
    public long getItemId(int i) { return i; }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        // to be continue
        if (view == null) {
            view = LayoutInflater.from(context).inflate(R.layout.diary_item_layout, null);
        }
        TextView title = (TextView)view.findViewById(R.id.title);
```