# 基本的

图片列表有多个，基本都是使用RecyclerView的分组标题的形式显示，图片加载框架glide

总体一般都是是用的是MVP结构，Activity和Fragment负责页面，presenter负责页面逻辑。

异步操作用的rxjava，只用了其中的简单代码，

## 初始化执行流程

进入APP，首先是LaunchActivity界面，这里目前主要执行了权限申请和开屏公告。

然后会进入ChoosePicActivity，这里首先调用APP的初始化代码，会执行：

读取APP基本配置，初始化bmob，设置异常捕捉器，启动后台线程，然后获取屏幕宽度，发送通知等操作。

然后初始化ChoosePicActivity的页面，获取intent数据等。然后初始化几个Fragment。首先进入本地图片的Fragment。下一步见后面

# 图片选择界面

页面结构 ViewPager+fagment。

其中的Fragment都继承了BaseFragment，BaseFragment目前主要用于检测Fragment第一次可见，第一次可见之后执行对应Fragment的初始化。

其中有onActivityResult中，通过intent获取苏三说，对各种返回情况的处理比较多。

目前3个部分，本地图片，贴图-选择表情等贴到图片上，模板-用于添加文字等的模板，如熊猫人等

## 执行流程

首先，本地图片LocalPresenter的startLoad回去加载本地的图片，通过数据库、媒体扫描图片，将扫描的数据放到usuManager、picInfoScanner等对象中。然后刷新页面，刷新文件列表信息。

后面此Fragment再次可见时也会调用异步扫描，然后刷新。

## 本地图片

在localPicture包下面，

页面上采用右侧抽屉布局，点击之后滑出文件列表

页面上显示的的本地的文件，本地文件首页分了3个部分，最近使用的图片，最近图片，和常用喜爱的图片。

点击文件夹某一项进入文件夹，显示文件夹的图片

**数据层：**

通过usuManager管理3种图片，最近和喜爱的数据存到数据库中。

注意各个地方图片的增删对常用图片列表，文件列表的数据同步操作。

## 贴图：

在tietuChoose包下面

贴图界面上面是分类列表，分类列表下面同样的用RecyclerView等进行展示。

**数据层：**

采用了缓存的方式，利用rxjava，下载图片都会先判断服务器图片更新时间是否大于本地下载时间，有更新则下载，没用用缓存。

先下载贴图的分类，然后下载具体分类下的贴图列表。

下载操作放在TietuDataManager中完成，将数据返回给UI层。

存贴图数据的对象是tietu\_material，然后里面的url交给glide判断缓存以及加载。

## 模板：

目前还没写，应该是利用贴图的同一个Fragment，传入不同参数