《移动互联网跨平台综合开发》报告

报 告 题 目　二手交易平台开发实践

专　　业　软件工程

小组成员

学号：201592038　姓名：阙建明

学号：　　姓名：

学号：　　姓名：

学号：　　姓名：

学号：　　姓名：

学号：　　姓名：

任 课 教 师　朱明

完 成 日 期　2018 年 5 月 16 日

大连理工大学软件学院

课程设计参数信息

代码量	设计图的个数	数据库表的个数

课程评价结果

1 级评价指标	2 级评价指标	评价方式	评价分值	评价结果
4. 研究：能够基于科学原理并采用科学方法对复杂工程问题进行研究，包括设计实验、分析与解释数据、并通过信息综合得到合理有效的结论。	（初级）③ 通过信息综合得到合理有效的结论的能力，包括从实验中呈现资料、解释资料或结果，并总结结论及给予建议，形成报告	报告、答辩	20	
5. 使用现代工具：能够针对复杂工程问题，开发、选择与使用恰当的技术、资源、现代工程工具和信息技术工具，包括对复杂工程问题的预测与模拟，并能够理解其局限性。	（高级）③ 对复杂工程问题进行预测与模拟，并能够理解其局限性	报告、答辩	50	
9. 个人和团队：能够在多学科背景下的团队中承担个体、团队成员以及负责人的角色。	（中级）② 在设计与开发解决方案过程中明确角色，承担责任	报告、答辩	30	

目　　录

1 背景与设计目标

1.1 背景

随着经济的飞速发展，人们的物质生活水平与过去相比也有了极大的提升。在以往的艰苦岁月里，衣物都是“新三年，旧三年，缝缝补补又三年”，贫穷所迫，大多数人对物品都是极其所用，即便坏了也是不舍得扔掉。而现今身处网络大时代的人们不仅物质生活水平提高了，人们的消费观念也有所改变。网络购物算是改变了大部分人的购物方式，足不出户就能买到大至大型家电，小到生活用品。

当然，消费的便捷和价格的便宜，也就带来了很多盲目消费。诸如双 11 等网络促销活动，让很多人就是为了买东西而买东西，经常会购置一大批现在用不到，或者一段时间之后可能用到，或者有需要但是需求量不大的东西，这就导致大量东西的闲置。同时，由于消费观念的改变以及物质生活的提高，很多东西基本上用不了很久就会换新了，这也让那些被淘汰但可能仍然可以使用的东西也被闲置。这种情况在校园里就更明显了，经常毕业季就会有大量的东西需要处理，书籍、电器、运动器材等等，这就带来了二手商品出售的需求。

在平时生活当中我们也会遇到这种情况，需要用到一种商品，但不必是全新的，只要功能完好，满足短期的使用需求即可。比如考研资料，一门选修课程的课本，宿舍里使用的洗衣机等等，这就带来了二手商品购置的需求。

1.2 设计目标

针对校园内二手商品交易的需求，本项目欲设计一款可以发布图文商品信息以及发布者联系信息，同时可以较好的形式展示给需要购买二手商品的用户，而且不涉及第三方资金保管，让用户直接接触的简易二手商品交易平台。

2 需求分析

2.1 用户需求

首先用户群体应该分为两种，一种是出售二手商品的，一种是购买二手商品的。

其中出售二手商品的用户应该具备基本的添加需要出售的商品的基本信息（名字，描述，价格等），用于买家联系的联系方式（手机号，QQ，微信等），用于展示商品信息的若干张图片信息。同时对于已发布的商品需要具备一定的管理能力。删除商品，查看自己发布的商品列表等。

而购买二手商品的用户应该具备查看当前都有哪些商品在出售，可以查看某个具体在售商品的详细信息（包括图文信息，卖家的联系方式等等）。

2.2 功能需求

- ✧ 登录、注册
- ✧ 访问相册以及选中多个图片资源
- ✧ 预览要上传的图片，查看商品详情时查看商品的图片
- ✧ 图片压缩
- ✧ 上传商品信息（包括图片信息）
- ✧ 获取并展示在售的商品
- ✧ 删除商品
- ✧ 查看用户自己发布的商品

2.3 业务需求

1. 服务器端需求

首先需要一个部署在云端的服务器，管理用户的信息，保存商品的信息。以及提供访问这些信息的接口。支持文件上传。

2. 客户端需求

- ✧ 客户端首先要提供网络请求的能力，以便于服务器端进行通信。
- ✧ 其次由于用户有上传图文信息的需求，客户端就必须提供可以访问相册，并选择多张相册的能力。

- ✧ 而且考虑到服务器的压力，以及大多数用户手机拍摄的图片多大 10M，客户端就需要具备在上传图片之前压缩图片的功能。
- ✧ 由于后期商品数据量巨大，不可能一次获取全部信息，所以客户端需要提供分页获取数据并展示的功能（下拉刷新，上拉加载更多）。
- ✧ 最后客户端需要以比较友好的方式展示商品的信息（采用瀑布流布局的方式展示获取到的在售商品信息）。
- ✧ 考虑到用户的体验性，需要较为流畅的网络图片加载与缓存策略（采用 Google 官方推荐的 Glide 图片加载库来实现）

3. 其他需求

- ✧ 性能需求： 网络请求应该进行异步处理，同时在获取到数据之后在 UI 线程对视图进行更新（Android 编程本就要求这样操作）。对于其他耗时操作，比如获取用户的相册信息等等，也应该获取，并且在获取过程中应该给用户明显的提示，且不能阻塞 UI 线程。用户操作流畅，无明显的卡顿、闪退现象。针对运行时可能出现的诸如空指针异常等错误应当适当处理。而且当前网络配置无法访问到服务器时，应当给用户相应的提示
- ✧ 交互需求：界面应该简单易用，且符合大多数人的操作习惯，对于主要的功能应该打开即得。对于需要输入大量信息的交互场景，也应该提供默认值以减轻用户的使用负担。

3 详细设计与实现

3.1 服务端设计及实现

考虑到项目整体的结构并不复杂，而且有简单易用的商业化后台解决方案，本项目中的后台采用的是 Bmob 的后台接入方案，其对常用的对数据的增删改查需求，用户信息管理需求等提供的便捷的解决方案。客户端在接入的时候只要在客户端定义好模型，然后调用 SDK 中的接口就可以在 Bmob 后台生成相应的数据结构

3.1.1 数据模型设计

1. 用户信息

Bmob 提供了对用户管理的支持，基本的 username，phone，email 等字段都已经封装在其提供的 BmobUser 当中，下面客户端定义的两个字段是对 Bmob 提供的 User 实体的信息进行的扩展

```
/**
 * 用户信息Model
 *
 * @param descption 用户个签
 * @param avatar 用户头像
 */
data class UserInfo(val descption: String = "",
                    val avatar: String = "") : BmobUser() {
}
```

图 1 客户端对 BmobUser 的扩展

下面就是 Bmob 后台针对上面的模型定义生成的数据表

objectId String	username String	password String	mobilePhoneNumberVer	mobilePhoneNumber String	descption String
e2eca50fa7	18340857280	******			
fdbc7fabba	qwe	******			
8b5c53cf06	qjm256	******			
f0c9c2ea4e	Rein	******			
6cd0fc8372	qjm255	******			
c7975703c9	qjm254	******			
a0a197ae36	qjm253	******			

图 2 Bmob 后台生成的 User 信息的表（一）

emailVerified Boolean	email String	authData authData	createdAt Date	updatedAt Date
			2018-05-13 11:02:26	2018-05-13 11:02:26
			2018-04-16 17:35:33	2018-04-16 17:35:33
			2018-04-14 18:30:40	2018-04-14 18:30:40
			2018-04-14 15:48:13	2018-04-14 15:48:13
			2018-04-14 15:43:13	2018-04-14 15:43:13
			2018-04-14 14:49:23	2018-04-14 14:49:23
			2018-04-14 14:46:29	2018-04-14 14:46:29

图 3　Bmob 后台生成的 User 信息的表（二）

2. 商品信息

```
/**
 * 商品信息Model
 *
 * @param name              商品的名字
 * @param description       对商品的描述
 * @param pictures          一个文件数组，包含了该商品的图片
 * @param originPrice       原价
 * @param remainNum         库存（扩展字段，未使用）
 * @param discount          折扣（扩展字段，未使用）
 * @param qqNumber          卖家的QQ号
 * @param weChatCount       卖家的微信号
 * @param owner             卖家的信息（与UserInfo是一对一的关联）
 * @see UserInfo
 */
class Commodity(val name: String = "",
                val description: String = "",
                val pictures: Array<BmobFile>? = null,
                val originPrice: Float = 1f,
                val remainNum: Int = 1,              //库存
                val discount: Float = 1f,
                val qqNumber: String = "",
                val phoneNumber: String = "",
                val weChatCount: String = "",
                val owner: UserInfo = UserInfo()
                ) : BmobObject()
```

图 4 客户端定义的商品类的模型

下面定义的商品信息当中，商品与商品的图片是一对多的关系，商品与商品主人是一对一的关系，客户端只需要像上面这样定义模型之间的关系，Bmob 后台就会根据模型间的依赖关系创建对应的数据表，并添加正确的外键关联。（其中，Bmob 对文件是单独处理的，它把每个上传的文件看做一个素材，所以上面一个商品关联多个图片素材，Bmob 在定义该字段的时候就会用一个固定格式的字符串数组来替代，BmobSDK 能正确解析该字段，并在查询的时候返回对应的 BmobFile 对象）

objectId String	weChatCount String	discount Number	email String	originPrice Number	owner Pointer<_User>	phoneNumber String	pictures Array
ddde7e9ea2	无	1		20	e2eca50fa7	无	[{"__type":"File","filename":
bbdc715e7b	无	1		0	e2eca50fa7	无	[{"__type":"File","filename":
576844f13a	无	1		0	e2eca50fa7	无	[{"__type":"File","filename":
30581466be	无	1		20	e2eca50fa7	无	[{"__type":"File","filename":
b17276fd73	无	1		0	e2eca50fa7	无	[{"__type":"File","filename":
2c414a069e	无	1		0	e2eca50fa7	无	[{"__type":"File","filename":
ceb26a4d39	无	1		0	e2eca50fa7	无	[{"__type":"File","filename":
44a0e0fafe	无	1		0	e2eca50fa7	无	[{"__type":"File","filename":
0049b39486	无	1		0	a0a197ae36	无	[{"group":"","url":"http://bmo
517b3fd6ae	无	1		0	a0a197ae36	无	[{"__type":"File","filename":
105cf7c47b	无	1		20	a0a197ae36	18340857280	[{"group":"","url":"http://bmo
856ed90f32	无	1		1000000	a0a197ae36	18340857269	[{"__type":"File","filename":
62742db644	无	1		10000000	a0a197ae36	18340852366	[{"__type":"File","filename":

图 5　Bmob 后台生成的 Commodity 信息的表（一）

qqNumber String	remainNum Number	title String	description String	createdAt Date	updatedAt Date	ACL ACL
无	0	接口	接口	2018-05-16 10:54:43	2018-05-16 10:54:43	
无	0	磁笔	`(*∩_∩*)'	2018-05-16 10:53:03	2018-05-16 10:53:03	
无	0	滑雪板	(´ε｀)	2018-05-16 10:52:21	2018-05-16 10:52:21	
无	1	俯卧撑抓手+腹肌滚轮	ლ(╹◡╹ლ)	2018-05-16 10:51:41	2018-05-16 10:51:41	
无	0	水下实验室	这个不卖	2018-05-16 10:49:21	2018-05-16 10:49:21	
无	0	软工习题	ლ(╹◡╹ლ)偷偷告诉你，	2018-05-16 10:48:50	2018-05-16 10:48:50	
无	0	哦哦哦哦哦哦	咯哦哦	2018-05-16 10:11:07	2018-05-16 10:11:07	
无	0	一只妹妹	(,,･ω･,,)	2018-05-16 10:05:43	2018-05-16 10:05:43	
无	0	X___X	(。＿＿。)	2018-05-14 20:57:43	2018-05-14 20:57:43	
无	0	一只弟弟	一只弟弟，两只，三只....	2018-05-14 16:48:49	2018-05-14 16:48:49	
1250422644	1	夏峰计网笔记	绝对真实	2018-05-14 15:29:18	2018-05-14 15:29:18	
123456789	1	海边别墅	ლ(╹◡╹ლ)	2018-05-14 15:25:11	2018-05-14 15:25:11	
125407369		海边别墅	(ฅωฅ)hiahiahia	2018-05-14 14:35:53	2018-05-14 14:35:53	

图 6　Bmob 后台生成的 Commodity 信息的表（一）

3.1.2 其它设计

对于数据库设计，用户信息管理，提供给客户端接口等需求 Bmob 提供的结局方案已经解决了这个问题。客户端只需要调用 Bmob SDK 提供的接口对数据进行操作即可。不许要额外的实现。

3.2 客户端设计及实现

3.2.1 用户管理设计

客户端对用户管理的需求无非是向服务端注册用户，提供用户的登录功以及从服务端获取用户基本信息的功能。同时要提供用户信息的本地持久化，保持登录状态等等。这些需求基本上在 BmobSDK 中的用户管理方案中已经进行了实现。

✧ 对于界面设计，需要提供登录注册的界面，以及展示用户信息的界面。

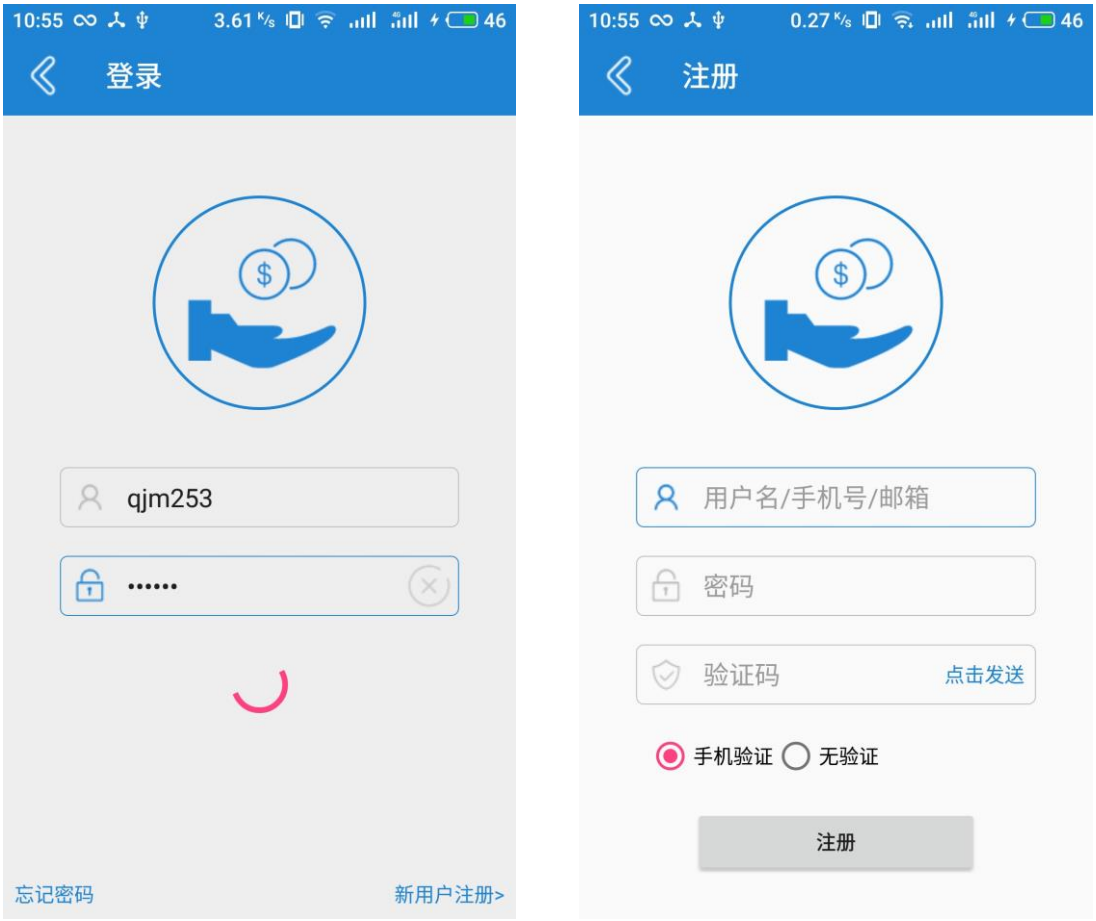

图 7 登录注册页面设计

✧ 同时对于用户状态的保持（已经登录的用户在一段时间内不需要再次登录，客户端可以保持登录的状态），只需要调用 BmobUser.getCurrentUser() 函数，如果返回的用户对象是 null，则表示没有登录，如果不是 null，则表示已经登录

3.2.2 网络框架设计

由于本项目后台采用的是Bmob的解决方案，所以所有的网络请求都通过调用Bmob SDK 中提供的接口来实现，其会自动在异步线程中进行网络请求，并且回调会在在 UI 线程中执行。同时对于错误的请求，也会提供相应的错误信息。

```
override fun login(username: String, password: String) {
    BmobUser.loginByAccount(username, password, object : LogInListener<UserInfo>() {
        override fun done(p0: UserInfo?, p1: BmobException?) {
            p1?.let { it: BmobException
                mPresenter.loginFail()
                ^let it.message?.let { mPresenter.toast(it) }
            } ?: p0?.let { mPresenter.loginSuccess() }
            ?: mPresenter.loginFail()
        }
    })
}
```

图 8 通过 BmobSDK 提供的接口进行网络请求示例

3.2.3 访问相册选择图片模块设计及实现

对于访问相册的能力，Android 系统通过 ContentProvider 对其它应用程序提供了访问相册的能力，读取相册信息可以通过这个系统的接口获取（代码易得，格式简单，这边就没有贴出获取相册资源的代码）。

对于相册信息的展示，这里采用网格布局的方式进行展示，每行显示三张图片，具体实现时采用的是 RecyclerView，并设置其布局管理器为 GridLayoutManager 来实现该效果。同时对于不同文件夹下的图片分开显示，提供一个自定义的继承自 PopupWindow 的弹窗来实现。

处于对代码复用性的考虑，通过访问相册选择若干张图片，并返回这些选中图片的 Path 的需求是很常见的。这里并没有将这一部分的功能直接写进项目里面，而是单独创建了一个 Library，将这部分的功能封装成一个库，对外只暴露跳转进相册选择界面的接口，以及在选择完成后返回选中图片 Path 的功能。同时在启动的时候也支持一些配置（诸如是单选模式还是多选模式，是否保存上一次选择的状态等等）

下面展示的是图片选择（多选），以及选择相册的界面

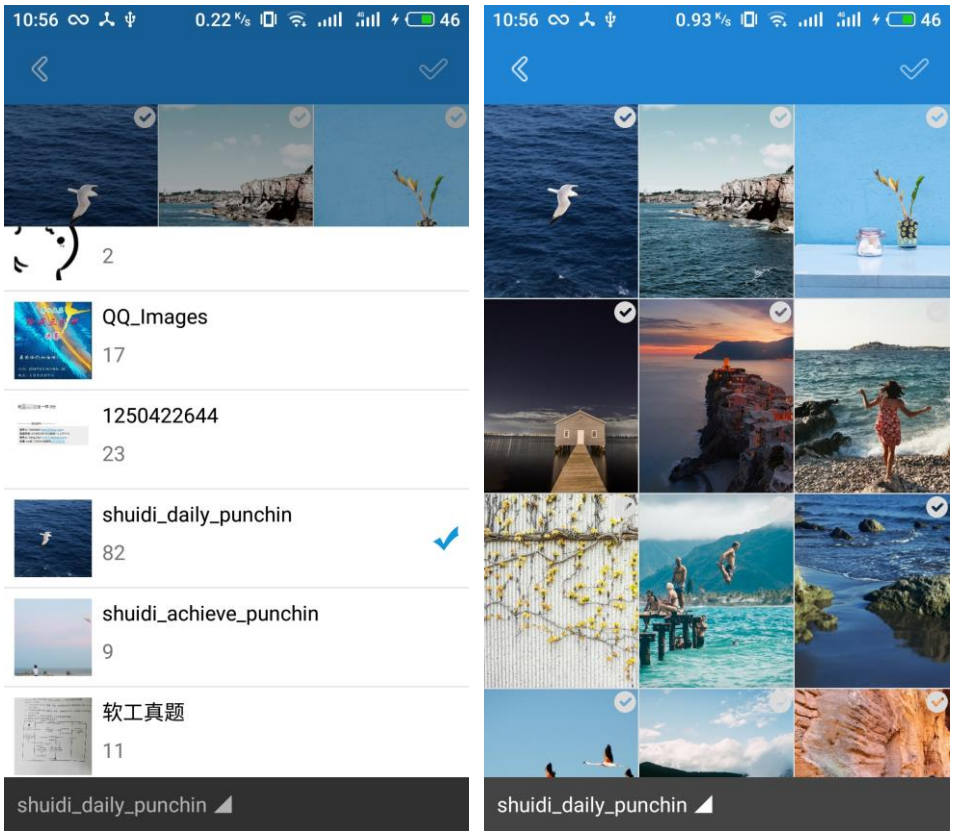

图 9 图片多选以及切换文件夹界面

下面展示的是该图片选择库 Github 文档中介绍的该库的使用方式

- **启动相册**

```
EasyDCIM.with(this) // 需要一个Activity对象
          .setMode(EasyDCIM.MODE_SELECT_SINGLE)    //设置模式：多选/单选
          .setSaveState(false)                     //设置是否保存上一次的选择状态
          .setEasyBarParam(EasyBarParams(titleRes = R.string.title, barBgColor = R.color.colorAccent,
                  rightRes = R.drawable.cat))      //设置相册选择器中Bar的样式
          .jumpForResult(0)                        //传入一个requestCode
```

- **回调中接收结果**

```
// 不管是单选还是多选，相册选择器返回的都是一个字符串数组，每个元素代表一张被选中图片的path
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    when(requestCode){
      0 -> {
            data?.getStringArrayExtra(DCIMActivity.RESULT_PATHS)?.let {
                result.text = Arrays.toString(it)
            }
        }
    }
}
```

图 10 DCIMDemo 库的使用方式

上面这个封装的图片选择库已经通过 JitPack 发布，使用的时候只需要通过 Gradle 包管理器导入即可。该库的 github 地址：https://github.com/SunnyQjm/DCIMDemo

3.2.4 上传商品界面设计及实现

上传商品信息，需要包括商品的名字，介绍，图片等等，其中图片的添加可以在界面中嵌入一个采用网格布局的 RecyclerView，选择的时候可以调用上文中介绍的 DCIMDemo 库来获取本机的图片资源，同时通过 Glide 图片库来将选择结果显示到上传界面上。下面展示的是添加商品界面，其中点击加号会跳转到图片选择界面，选中要上传的图片之后就会在列表中展示。点击图片右上角的删除按钮可以删除某一个选中的图片，点击任意一张图片就会进入图片预览界面，里面可以对图片进行放大缩小，以及左右滚动查看其它图片。（其中该图片预览空间是下文中会提到的另一个本项目成员封装的库）。

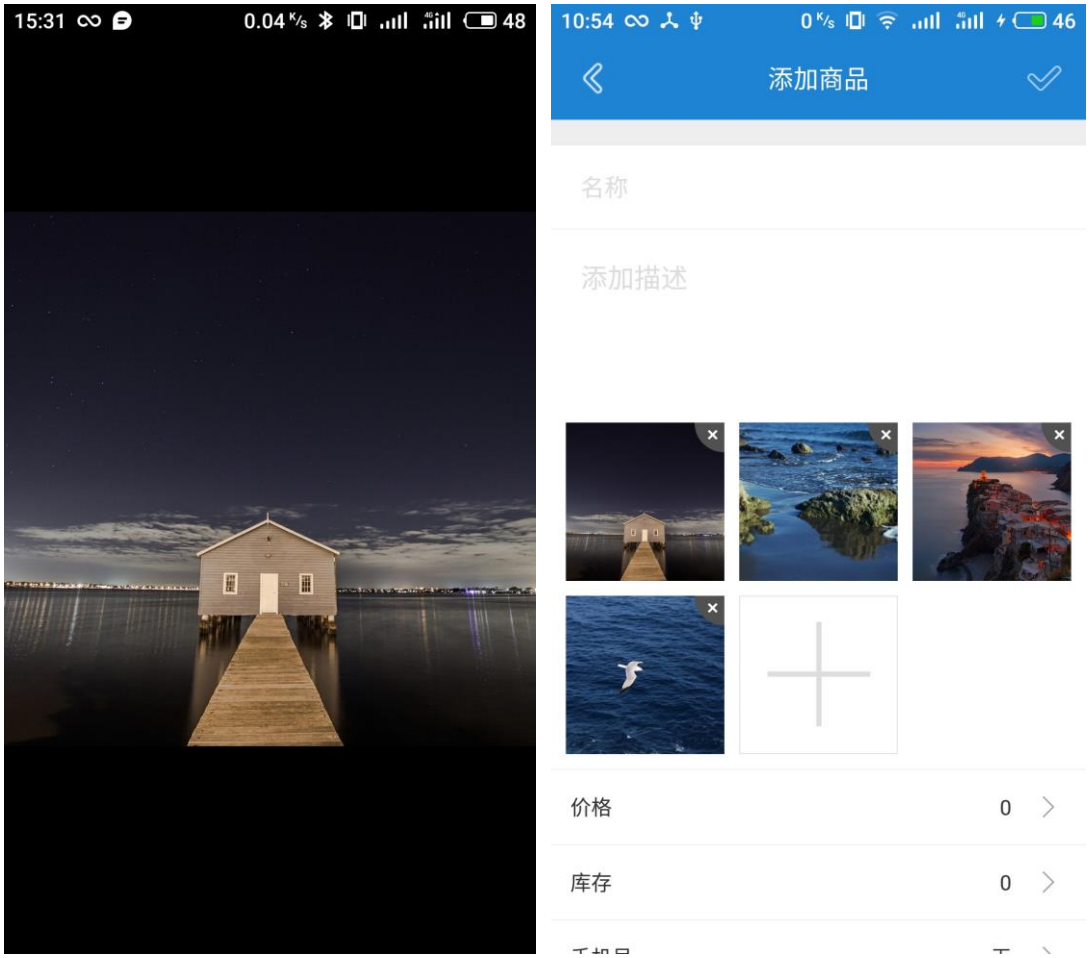

图 11 添加商品界面以及图片预览界面

3.2.5 图片预览控件实现

图片的预览功能也是很常见的功能，所以为了代码复用，本项目里面也是将图片预览的功能单独抽离出来写成了一个控件库。要实现的功能有：可以显示多张图片，通过作用滚动查看若干张图片；提供图片放大缩小查看的功能；双击放大或者恢复成原大小，图片适应屏幕大小（根据图片的比例自动选择是适应屏幕宽度还是屏幕高度）。

实现的使用使用 ViewPage 来实现左右切换查看图片的功能。对于图片的放大缩小等功能，则通过一个自定的 ImageView 来实现（复写其 onGlobalLayout 方法实现初始布局，复写 onTouch 方法以及设置手势监听实现缩放以及边界检测等）

下面展示的是该图片预览库 Github 文档中介绍的该库的使用方式

- Step 3 : 在布局中使用

```
<com.sunny.scrollphotoview.ScrollPhotoView
    android:id="@+id/spv"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</com.sunny.scrollphotoview.ScrollPhotoView>
```

- Step 4 : 在代码中设置
  - 设置显示的图片集合（传入一个string数组，可以是url，也可以是图片路径，都在下面的回调中自行处理）

```
spv.setUrls(arrayOf("https://timgsa.baidu.com/timg?image&quality=80&size=b9999_10000&sec=1514351983845&di=3e
```

  - 设置回调加载（必须设置，建议用Glide来加载，可加载网络图片，本地图片，gif等）

```
spv.imgLoader = {
        url, view ->
        Glide.with(this)
                .load(url)
                .into(view)
    }
```

  - 设置点击回调，支持单击和双击

```
spv.onScrollPhotoViewClickListener = object : ScrollPhotoView.OnScrollPhotoViewClickListener{
        override fun onDoubleTap(e: MotionEvent?) {
        }
        override fun onClick(e: MotionEvent?) {
        }
    }
```

图 12 ScrollPhotoView 库的使用方式

上面这个封装的图片选择库已经通过 JitPack 发布，使用的时候只需要通过 Gradle 包管理器导入即可。该库的 github 地址：https://github.com/SunnyQjm/ScrollPhotoView

3.2.6 交易中心显示商品列表实现

首先是需要获取到商品的信息，而且由于商品的信息可能很多，所以需要分页获取。所幸的是 BmobSDK 中提供了分页获取查询信息的实现方案，只要调用查询对象的 skip() 方法，传入需要跳过的条目数量，就能过实现分页的需求。

要展示商品的信息，需要一个比较合理的方式，在一屏中放入适当数量的包含商品信息的 Item。本项目采用的是三列瀑布流的方式。由于每个商品的图片信息尺寸可能不一样，所以用瀑布流效果可以很好的展示图片的全貌，同时整体的视觉效果也会比较好。

具体实现瀑布流采用的是 RecyclerView + Glide 的实现方案，其中设置 RecyclerView 的布局管理器对象为 StaggeredGridLayoutManager，已实现瀑布流布局效果，而 Glide 则用于异步加载网络图片，以及对图片进行缓存，同时还附带动画效果。

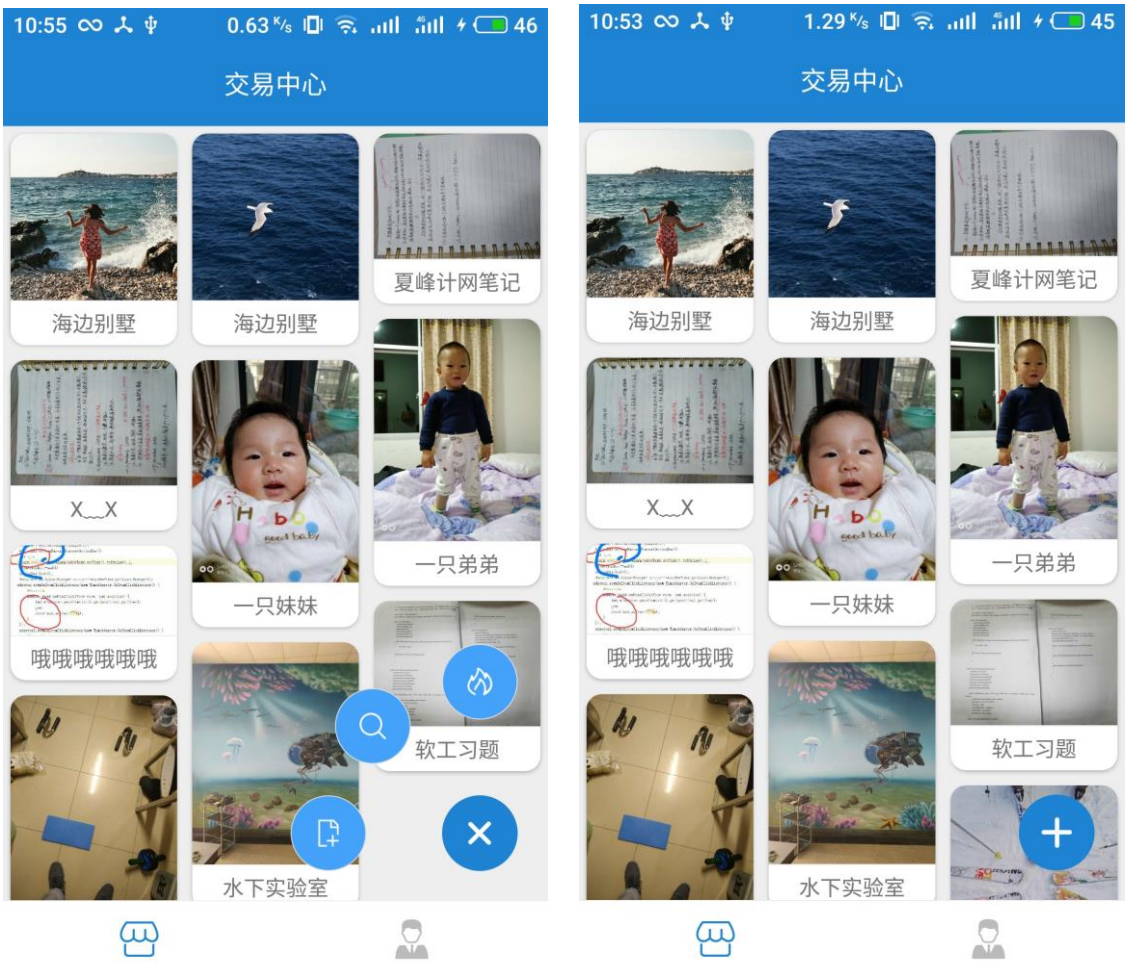

图 13 交易中心瀑布流显示商品信息界面

图 13 展示的是交易中心瀑布流显示商品信息列表的效果图，同时还有一个弧形展开菜单栏效果（该菜单栏也是项目成员之前的项目中提炼而成的一个开源库）。

3.2.7 商品详情页设计与实现

商品详情页主要展示商品的详细信息（名字，介绍，价格，物主的联系方式，以及图片介绍），同时对于商品的发布者还应该显示删除编辑等功能。

对于图片的显示，本项目中采用的是一个轮播栏的方式，如果商品有多张图片，则会轮流播放（轮播功能的实现是用 Github 上成熟的 banner 库来实现的），其它信息则以文本的方式显示，其中联系方式的文本设置可以点击或者复制（手机号点击之后展示可以打电话发短信等）。

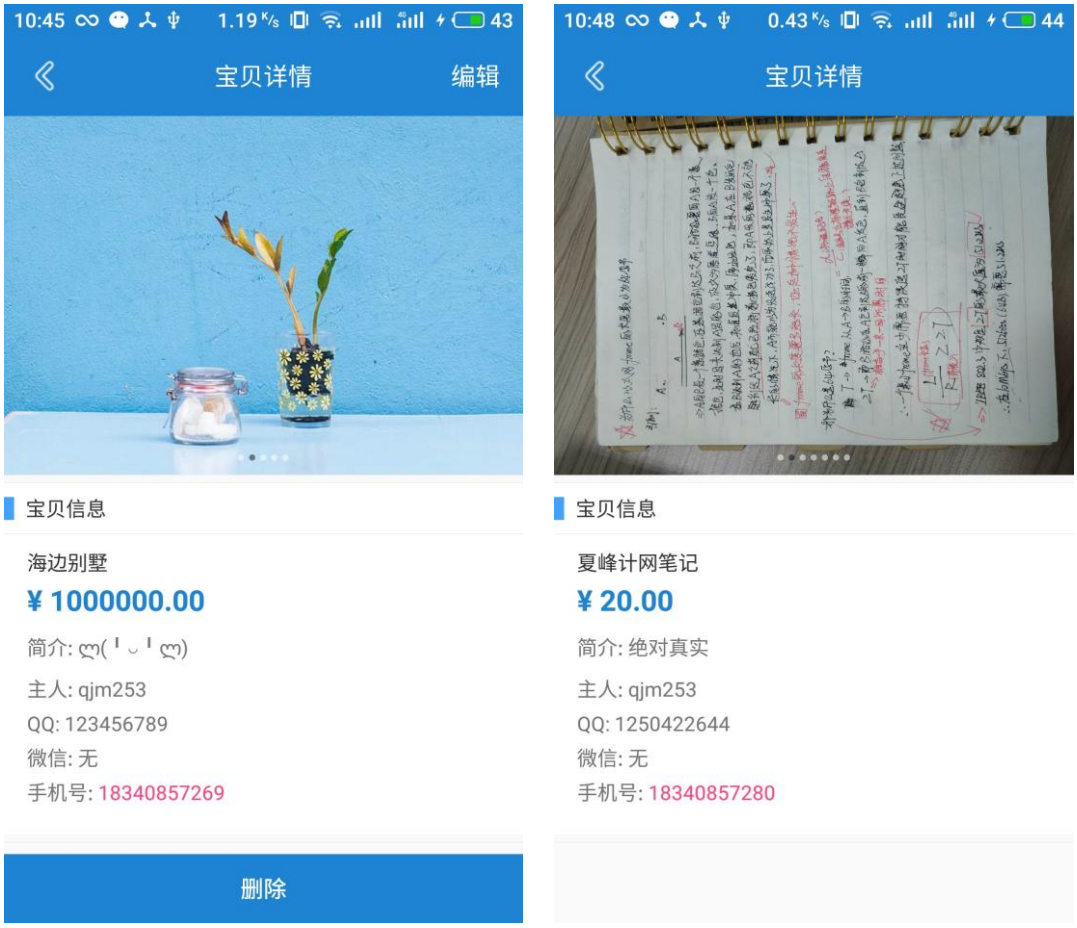

图 14 商品详情页（左边为发布者打开时的界面）

图 14 展示的是商品详情界面，其中左边是商品发布者打开时的界面，可以执行删除该商品的，而右边则是买家打开商品的界面。其中最上面是轮播介绍商品的图片，点击可以进入图片预览界面（图片预览也是用的上文中介绍的 ScrollPhotoView 这个控件实现的）。而对于联系方式可以选择复制，手机号码可以点击拨打电话或者发送短信。

3.2.8 用户管理已发布的商品

对于管理已发布的商品列表，本项目的实现是简单的获取到本用户所发布的商品的列表，然后将其展示，点击其中的某一项可以进入该商品的详情页，查看详情或者执行删除操作。

图 15 用户管理已发布商品界面

3.2.9 客户端所采用的架构等其他细节介绍

客户端整体采用的 MVP 架构，目的是实现 model 层和 view 层的分隔，使得代码易于维护。并且项目中提炼出了 MVPBaseActivity 和 MVPBaseFragment，使得所有继承自这两个抽象类的 Activity 或者 Fragment 可以很简单的实现 MVP 模式，而且便于统一管理。

其中 MVPBaseFragment 中实现了对 Fragment 懒加载的统一实现，所有继承自该类的 Fragment 都具有懒加载的特性，可以减少不必要的请求，给用户更流畅的体验。

对于下拉刷新和上拉加载更多，项目中通过一个自定义的继承自 ViewGroup 的 RefreshLayout 来实现，同时对外提供刷新头和加载脚布局接口，可以很方便的切换不同的刷新头和加载脚布局的不同效果。同时项目中提炼出了两个基类，分别为 BaseRecyclerViewActivity 和 BaseRecyclerViewFragment，使得所有继承自这两个基类的函数只需要简单复写 loadData 函数，提供加载数据的实现。就能够拥有下拉刷新和上拉加载更多的实现。这样在多个地方使用的时候就可以大大减少重复的代码。

4 依赖类库

4.1 第三方库

- Glide：https://github.com/bumptech/glide
- BRVAH：https://github.com/CymChad/BaseRecyclerViewAdapterHelper
- Luban：https://github.com/Curzibn/Luban
- banner：https://github.com/youth5201314/banner

4.2 自己封装的库

- ScrollPhotoView：https://github.com/SunnyQjm/ScrollPhotoView
- DCIMDemo：https://github.com/SunnyQjm/DCIMDemo
- EasyBar：https://github.com/SunnyQjm/EasyBar
- ArcMenu：https://github.com/SunnyQjm/ArcMenu