# CustomerApp开发文档

[1. 项目背景&运行环境 2](#_Toc30089)

[2. 功能划分&描述 3](#_Toc14051)

[3. 架构描述 4](#_Toc10469)

[3.1 整体架构MVP 5](#_Toc18143)

[3.2 网络请求 9](#_Toc3681)

[3.3 设计风格 10](#_Toc6976)

[3.4 注解工具 10](#_Toc13155)

[3.5 异步架构 10](#_Toc13290)

[3.6 图片加载 11](#_Toc2033)

[3.7 LBS服务 11](#_Toc14776)

[3.8 支付方式 11](#_Toc20157)

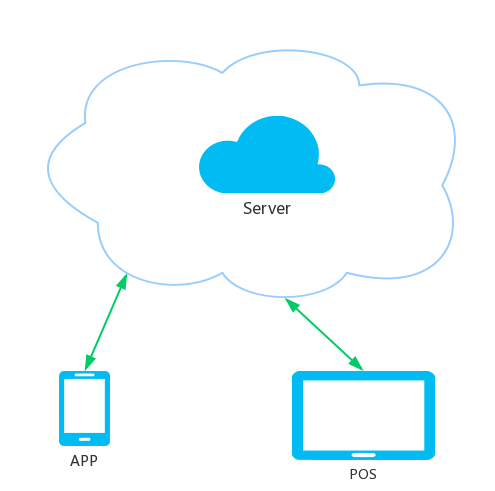
[3.9 异常报告 12](#_Toc13516)

[3.10其他使用 12](#_Toc26293)

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 修改日期 | 修改人员 | 备注 |
| V1.0 | 2017-01-11 | Jack | 初步建立 |
|  |  |  |  |
|  |  |  |  |

## 1. 项目背景&运行环境

Customer App是以POS为核心而开发的一款自助点餐的应用，中间由Server作为连接从而保持与POS数据的一致性。



标准屏幕: 720 X 1280

Java SDK: 1.8

最低API : 16 (Android4.1.1)

支持类型: 只支持手机

默认语言: 英语

其他语言: 简体华语，马来语，泰米尔语

英文名称: W&O Customer

中文名称: W&O 自助点餐

## 2. 功能划分&描述

1. Google LBS 查询附件餐厅

使用WIFI/GPS进行定位获取附件餐厅，还可以针对餐厅类型，食物风格进行筛选

1. 根据条件筛选/搜索餐厅

进入搜索界面选择搜索词进行搜索

1. 下单

进入下单页面选择菜市和点餐类型完成点餐初始化

1. 支付

选择Masterpass进行支付完成订单

1. 订单查看/删除/退款/取消

在订单列表页面进行操作

1. 个人设定(地址/个人信息等)

登陆成功后对地址和个人信息进行操作

详细的流程参照 [墨刀](https://modao.cc/app/f26c342dbb1a1d02f3d974c3f82a02571e1e7e93/workflow#flows)

## 3. 架构描述

CustomerApp采用MVP作为主要结构，使用当前主流开发框架，当然由于个人的能力和时间有限不能充分发挥其能力，希望日后能重构代码，将框架发挥到极致。

整体架构: MVP

网络请求: okhttp3.3.1 + retrofit2.1.0

设计风格: Material Design

注解工具: ButterKnife5.1.1

异步架构: RxJava1.1.6 + RxAndroid1.2.1 + RxBus

图片加载: Glide3.6.1

LBS服务: Google LBS

支付方式: MCPayment

异常报告: ARCA

其他使用: zxing + Android5.0权限 + Proguard

参考开源项目:

[PrettyGirls](https://github.com/PleaseCallMeCoder/PrettyGirls) 一款MVP+Retrofit+RxJava+MaterialDesign和gank.io的MeiZhi客户端

[AndroidFire](https://github.com/jaydenxiao2016/AndroidFire) 基于 Material Design + MVP + RxJava + Retrofit + Glide一款新闻阅读 App框架

[EasyRecyclerView](https://github.com/Jude95/EasyRecyclerView) 一款将Recyclerview封装的很好的开发库

[MaterialDesign](https://github.com/navasmdc/MaterialDesignLibrary) 包含Material Design风格的Button, Switch, Widgets

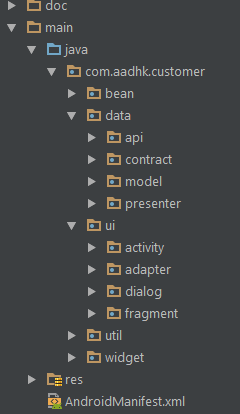
[MaterialEditText](https://github.com/rengwuxian/MaterialEditText) Material Design风格的编辑控件

[FlycoTabLayout](https://github.com/H07000223/FlycoTabLayout) 安卓TabLayout库

[EasyPermissions](https://github.com/googlesamples/easypermissions) 针对Android5.0后权限处理库

## 3.1 整体架构MVP

项目结构图1.1

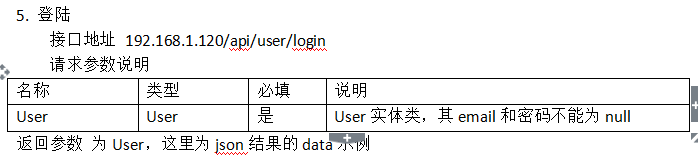


图(1.1)

* bean-------------------------------实体类
* data-------------------------------数据相关
* api--------------------------请求接口
* contract-------------------MVP接口约束
* model---------------------Model数据处理层
* presenter-----------------Presenter层
* ui----------------------------------UI相关
* activity---------------------Activity相关
* adapter--------------------Adapter相关
* dialog----------------------Dialog相关
* fragment------------------Fragment相关
* util---------------------------------工具相关
* widget----------------------------自定义View相关

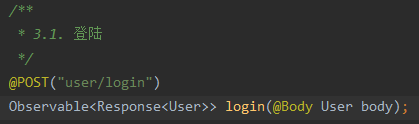
这里以登陆为例，阐明开发流程和思路:

Step 1. 定义服务器接口，接口名称，请求参数，返回参数。



这里json结果就不给出了，[接口文档编写参考](https://www.juhe.cn/docs/api/id/103)

Step 2. 在com.aadhk.customer.data.api.ApiService写好该接口。



这里使用Retrofit2框架，和Okhttp结合起来，设定连接/读/写超时，拦截器，GSON转化器，基础URL等设定。例如URL为 http://47.91.143.157:8080/api

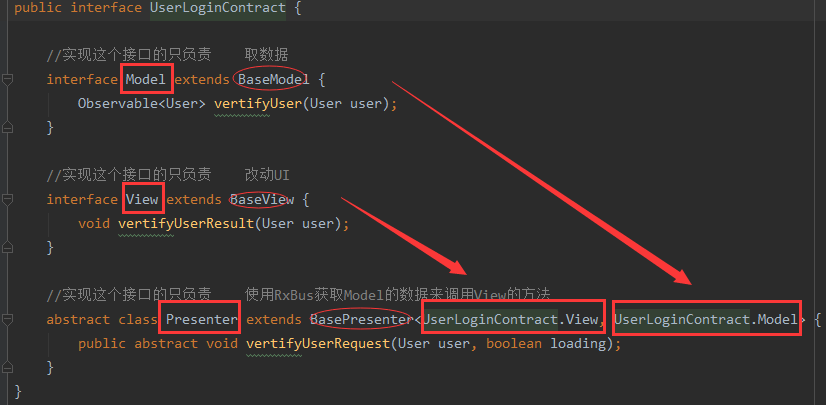
就可以通过定义这个接口来表明

请求路径: http://47.91.143.157:8080/api/ use/login

请求数据: User

返回数据: Response<User> [这里做了封装]

Step 3. 在com.aadhk.customer.data.contract新建UserLoginContract类。

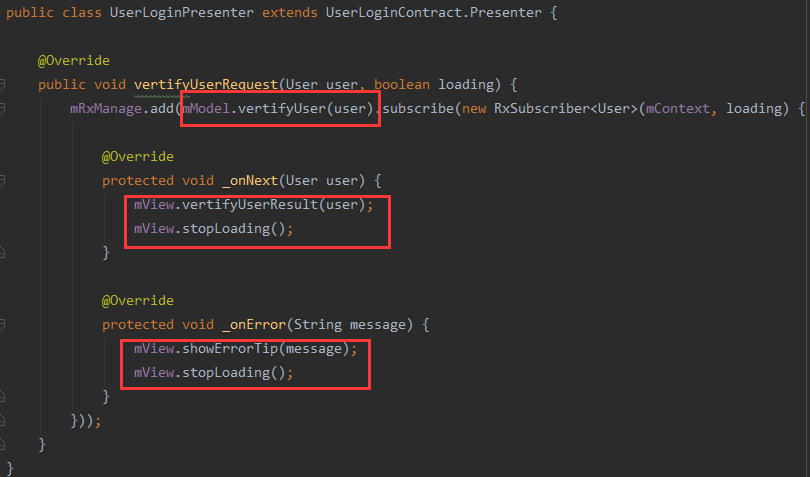


Contract的作用就是约束MVP三层接口的规范，设计思路首先从Presenter开始，写好请求名称和参数，然后写Model的方法，最后根据结果不同写View的接口。

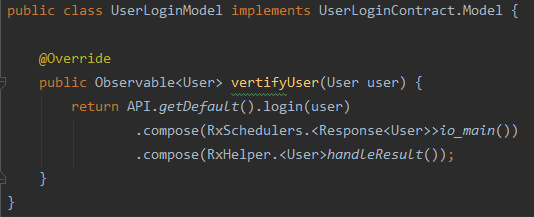
比如登陆，那么点击发出登陆请求，首先传递给Presenter层，Presenter调用Model层的网络数据请求，Model调用后返回结果，可能成功也可能不成功，这个时候再来写View层的数据返回接口。

这样做的好处就是View层没有具体的数据请求代码，Model层没有View层的代码，Presenter完全基于接口进行处理。

Presenter层

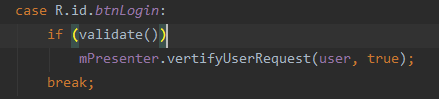


Model层

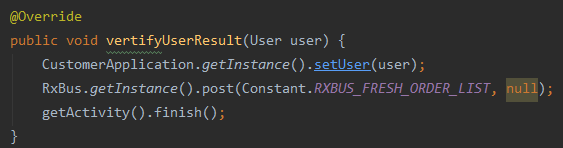


View层

调用



数据成功回传



当然这里做了一些Rx高级封装，不过MVP大致就是这样，完全基于纯接口编程。

RxJava和RxAndroid的使用经典教程

<http://gank.io/post/560e15be2dca930e00da1083>

<http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2015/0430/2815.html>

这里讲一下RxBus的原理

RxBus是一种模式，为事件总线，采用观察者模式，有点和回调相似，但是比起回调高了一个等级，在全局任意地方可以使用，需要注意的是要及时取消订阅，防止内存泄漏。

例子: 订单列表页用来显示用户的订单信息，而用户在登陆，注销，下单时需要对这个页面进行刷新，而订单列表页的Activity启动模式为singleTask，只有一个实例。

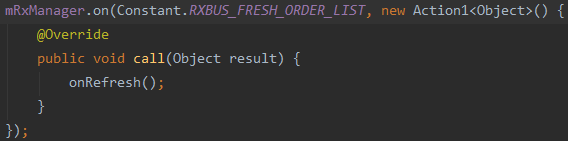
这里就使用了RxBus来进行订阅和传递消息。

详情需要自行去学习，这里参考是前面Github上的AndroidFire

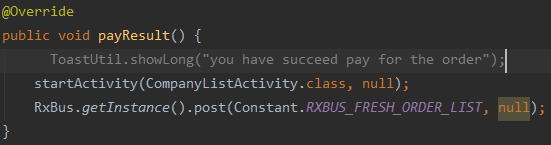
目前为止我也不能完全掌握，源代码中的代码需要好好研究。

下面给出图片的例子

在订单列表页面中



而在登陆，注销，下单后调用即可完成刷新



但是一定不要出现内存泄漏，在生命周期结束时记得取消订阅。

## 3.2 网络请求

这里就不做过多的阐述了，网上一大堆，这里仅仅给出资料或者参考CustomerApp

<https://square.github.io/retrofit/> Retrofit官网

<https://github.com/square/retrofit>

<https://github.com/swanson/retrofit-demo>

## 3.3 设计风格

Material Design

这里主要参考几个库

[MaterialDesign](https://github.com/navasmdc/MaterialDesignLibrary) 包含Material Design风格的Button, Switch, Widgets

[MaterialEditText](https://github.com/rengwuxian/MaterialEditText) Material Design风格的编辑控件

[FlycoTabLayout](https://github.com/H07000223/FlycoTabLayout) 安卓TabLayout库

还有几个控件的使用

Toolbar

CardView

Icon

<http://www.iconfont.cn/plus/home/index?spm=a313x.7781069.0.0.VxTKIo> 阿里巴巴矢量图库

<https://material.uplabs.com/> Logo生成器

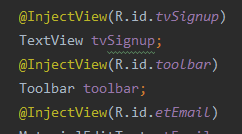
<https://fonts.google.com/?category=Serif> 字体

<https://www.materialpalette.com/> 官方参考

<http://wiki.jikexueyuan.com/project/material-design/> Material Design 中文版

## 3.4 注解工具

[Butterknife](http://jakewharton.github.io/butterknife/) 安装插件后不需要手动写findViewById(R.id.tvName)，直接生成下面的代码



## 3.5 异步架构

RxJava 和 RxAndroid

这里说明的是RxJava可以被当作纯Java多线程框架，而RxAndroid是为了支持Android而辅助RxJava的一款多线程框架，非常强大，能熟练使用代码将非常清晰，配合Retrofit发挥出最大的威力。

经典入门教程 <http://gank.io/post/560e15be2dca930e00da1083>

由于时间的关系，这种架构我只是研究了一般用法，在后续的工作中讲持续补充。

[2017年1月11日]

## 3.6 图片加载

图片加载框架选择了Glide，因为和其他框架比起来目前是最好的一个, Glide

是Google的开源项目，在Google I/O大会上被推荐使用，遵循BSD, MIT, Apache2.0协议，具有获取、解码和展示视频剧照、图片、动画等功能，它还有灵活的API, 能够将Glide应用在几乎任何网络协议栈里。创建Glide的主要目的有两个，一个是实现平滑的图片列表滚动效果，另一个是支持远程图片的获取、大小调整和展示。

在实际项目中需要做的是配置好缓存，然后封装工具类使用。

在CustomerApp中体现在

com.aadhk.customer.util.CustomCachingGlideModule

com.aadhk.library.utils.ImageLoaderUtils

## 3.7 LBS服务

因为GWF的存在导致本次Google LBS的开发非常不顺利，Google 位置信息提供了非常详细的API和文档，这里大致流程如下，进入开发者中心，申请一个KEY和打开哪些服务，建立项目得到API\_KEY集成到项目中, 免费账号日定位次数不能超过1000。

Google Map 分为V1 和V2版，在V1版本中可以通过wifi和GPS进行定位，而V2完全依靠GoogleService进行位置获取，因为V2可以做到以最少的资源和最合理的方式拿到最精确的定位，目前备受推崇，API分水岭为Android5.0

但是GWF的存在就不能通过GoogleSercvice获取到经纬度，所以这里就优先使用V2，失败了再使用V1，如果还失败那么使用Google提供的地图进行位置选取。

详情见

<https://developers.google.com/maps/documentation/android-api/signup>

<https://developers.google.com/places/android-api/>

[Android-location-tracker](https://github.com/quentin7b/android-location-tracker) V1下封装好的位置获取

com.aadhk.customer.ui.activity.CompanyListActivity

com.aadhk.customer.util.FetchAddressIntentService

## 3.8 支付方式

Masterpass，中文名称万事达，全球第二大信用卡国际组织，其原生态API太过复杂，通过新加坡一家金融科技公司MC Payment进行支付，集成详情[MCP MasterPass API Integration Guide v1.04.pdf]，其中遇到很多坑，首先是账号信息有误，其次是许多隐晦的东西没有说明导致开发时间异常的长久，这里推荐一款非常棒的发送请求的Chrom插件Advanced REST client。

## 3.9 异常报告

ACRA 全名是 Application Crash Reports for Android。是用来跟踪Android应用崩溃。

初始化后，ACRA会自动捕捉到崩溃，有try和catch的地方，要加上ACRA.getErrorReporter().handleException(e);

## 3.10其他使用

二维码扫描: <https://github.com/dm77/barcodescanner>

Android5.0权限: <https://github.com/googlesamples/easypermissions>

Proguard: <https://www.guardsquare.com/en/proguard/manual/examples#androidapplication>

<http://blog.csdn.net/banketree/article/details/41928175>

参考

饿了么

POS