目录

[摘 要 1](#_Toc39269486)

[Abstract 2](#_Toc39269487)

[第1章 绪论 3](#_Toc39269488)

[1.1研究的背景及意义 3](#_Toc39269489)

[1.2国内外研究现状 4](#_Toc39269490)

[1.3 本文的主要工作 5](#_Toc39269491)

[1.4 论文的组织结构 5](#_Toc39269492)

[第2章 系统相关技术分析 6](#_Toc39269493)

[2.1 移动UI框架 6](#_Toc39269494)

[2.1.1 Flutter（自绘+原生） 6](#_Toc39269495)

[2.1.2 React Native（js+原生渲染） 7](#_Toc39269496)

[2.1.3 HyBrid（H5+原生） 7](#_Toc39269497)

[2.1.4 对比 8](#_Toc39269498)

[2.2 服务器技术选择 8](#_Toc39269499)

[2.2.1 Springboot 8](#_Toc39269500)

[2.2.2 MySQL 8](#_Toc39269501)

[2.3开发及运行环境 9](#_Toc39269502)

[2.3.1 运行环境 9](#_Toc39269503)

[2.3.2 开发工具及配置 9](#_Toc39269504)

[2.4 本章小结 9](#_Toc39269505)

[第3章 需求分析 11](#_Toc39269506)

[3.1需求分析概述 11](#_Toc39269507)

[3.2 可行性分析 11](#_Toc39269508)

[3.2.1 技术可行性分析 11](#_Toc39269509)

[3.2.2 操作可行性分析 12](#_Toc39269510)

[3.2.3 经济可行性分析 12](#_Toc39269511)

[3.2.4 总结 12](#_Toc39269512)

[3.3 功能需求分析 12](#_Toc39269513)

[3.3.1 用户分析 12](#_Toc39269514)

[3.3.2 总体需求分析 13](#_Toc39269515)

[3.3.3 详细功能模块分析 14](#_Toc39269516)

[3.4 系统非功能性需求 17](#_Toc39269517)

[3.4.1系统安全性需求 17](#_Toc39269518)

[3.4.2 系统易用性需求 18](#_Toc39269519)

[3.4.3系统性能需求 18](#_Toc39269520)

[3.5本章小结 18](#_Toc39269521)

[第4章 移动众包手机应用系统设计 19](#_Toc39269522)

[4.1 系统设计原则 19](#_Toc39269523)

[4.2 系统模型结构 19](#_Toc39269524)

[4.3 总体设计 20](#_Toc39269525)

[4.3.1 面向客户端的服务器接口设计 20](#_Toc39269526)

[4.3.2 项目结构设计 22](#_Toc39269527)

[4.3.3 跨组件状态共享 23](#_Toc39269528)

[4.4 数据库设计 23](#_Toc39269529)

[4.4.1 用户信息表 24](#_Toc39269530)

[4.4.2 位置信息表 25](#_Toc39269531)

[4.4.3 登录信息表 26](#_Toc39269532)

[4.4.4 离线任务表 26](#_Toc39269533)

[4.4.5 离线任务接单表 28](#_Toc39269534)

[4.4.6 线上任务表 28](#_Toc39269535)

[4.4.7 线上任务接单表 29](#_Toc39269536)

[4.5 用户功能层设计 30](#_Toc39269537)

[4.5.1 即时通讯模块 30](#_Toc39269538)

[4.5.2 用户管理模块 30](#_Toc39269539)

[4.5.3任务模块 31](#_Toc39269540)

[4.6 本章小结 32](#_Toc39269541)

[第5章 移动众包手机应用系统的实现 32](#_Toc39269542)

[5.1 基础功能块实现 32](#_Toc39269543)

[5.1.1 图片选择组件 32](#_Toc39269544)

[5.1.2 百度定位插件 33](#_Toc39269545)

[5.1.3 QQ登录插件 34](#_Toc39269546)

[5.1.4 自定义Toast组件 34](#_Toc39269547)

[5.2 客户端功能实现 35](#_Toc39269548)

[5.2.1 个人信息模块 35](#_Toc39269549)

[5.2.2 众包发布功能实现 37](#_Toc39269550)

[5.2.3 线上众包审核功能 39](#_Toc39269551)

[5.3 主题相关 39](#_Toc39269552)

[5.4 服务端功能实现 40](#_Toc39269553)

[5.5 本章小结 41](#_Toc39269554)

[第6章 系统测试 42](#_Toc39269555)

[6.1 测试目的与原则 42](#_Toc39269556)

[6.2 系统稳定性测试 42](#_Toc39269557)

[6.3 功能测试 42](#_Toc39269558)

[6.3.1 登录注册测试 42](#_Toc39269559)

[6.3.2 线上任务发布 43](#_Toc39269560)

[6.4 安全性测试 44](#_Toc39269561)

[6.5 本章小结 44](#_Toc39269562)

[第7章 总结与展望 45](#_Toc39269563)

[7.1 总结 45](#_Toc39269564)

[7.2 展望 45](#_Toc39269565)

[致谢 47](#_Toc39269566)

[参考文献 48](#_Toc39269567)

[附录1 英文原文 50](#_Toc39269568)

[附录2 译文 53](#_Toc39269569)

移动众包手机应用系统设计与实现

# 摘 要

在当前，众包的概念逐渐兴起，众包是一个企业或者机构把由自己员工完成的任务以自愿的形式外包给不特定的群众网络的行为。众包有利于充分发掘广大消费者的潜力和力量来参加企业的各种开发活动。而当前的众包发布要么采用网页的形式，运行速度慢，对于高频使用用户低效且难用；要么采用原生开发，开发效率低下，无法适应当前快速发展的互联网时代。

因此本系统决定开发一个众包移动手机应用，使用由谷歌推出的移动UI框架Flutter，它可以在android和ios上快速构建一个高品质的用户界面，后端采用当前主流的springboot进行构建。其用户主要是众包任务的参与者，可以进行任务的发布、执行、完成等。本文以众包移动程序需求出发，分析了相应的需求，设计了各个模块，有利于跟踪目前流行的众包应用模式和移动应用开发技术，较为条理的论述众包的分发模式，在探究flutter开发可行性和积累相应的flutter开发经验的同时，也可应用于小范围内的众包分发场景实际使用。

**关键词：** 众包；Flutter；移动应用；springboot

# Abstract

At present, the concept of crowdsourcing is gradually emerging. Crowdsourcing is the behavior of an enterprise or organization to extend the tasks completed by its employees to the network of unspecified participants in a voluntary manner. Crowdsourcing helps to fully explore the potential and power of consumers to participate in various development activities of enterprises. However, the current crowdsourcing release or adopting the form of a webpage runs slowly, which is inefficient and difficult to convert users. Using native development, the development efficiency is low and cannot adapt to the current rapid development of the Internet era.

Therefore, this system decided to develop a crowdsourced mobile phone application, using a mobile UI framework Flutter, which is mainly promoted by Google, which can quickly build a high-quality user interface on Android and ios, and use the current mainstream springboot to build. It mainly uses users as participants in crowdsourcing tasks, and can publish, execute, and complete tasks. Starting from the needs of crowdsourcing mobile programs, the corresponding needs are analyzed, and the various modules are designed to help track the current popular crowdsourcing application model and mobile application development technology. Multiple organized additional crowdsourcing distribution models are exploring Flutter. While developing usability and accumulating corresponding Flutter development experience, it can also replace the actual use of crowdsourcing layout scenarios in a small area.

**KeyWord**: crowdsourcing; Flutter; mobile application; springboot

# 第1章 绪论

## 1.1研究的背景及意义

众包产生于欧美企业的创新任务，由J.Howe首次提出，认为众包是一个企业或者机构把由自己员工完成的任务以自愿的形式外包给不特定的群众网络的行为。众包在2007年被引入中国并且快速发展，在2016年国务院常务会议上李克强总理强调，发展服务外包有利于推动产业升级和扩大就业，也给众包在国内的发展注入了一针强心剂。众包被认为充分发挥了解决者们多元化的知识背景。

研究众包，首先就需要与外包进行区分，伴随着全球化的发展，外包服务逐渐发展并热门，在当今全球化每一个人都可以作为个体参加全球竞争与合作，似乎外包的极致便是众包，其实二者还是有着很大区别的。外包基本上是一对一（这里的一既可以指个人也可以指团体）的关系，而众包则为一对多（这里的多指分散的个体）的关系；外包强调的是高度专业化，而众包则主张业余化；外包强调的是高度团体化，而众包则强调群众个人。在作用效果上，外包主要是将不具有核心竞争力或难度较低的业务分配出去，而众包却集合劳动力来加强企业的竞争力。当然，对于使用者来说众包的概念并不受用，他并不关心到底是如何完成的，他只关心完成的质量如何。

在当前，较为典型的众包平台网站主要有：阿里众包、百度众包、译言网等等。其中译言网采用的模式值得研究，较有代表性。译言网（<http://auth.yeeyan.org>）是一家提供翻译的网站，它采用的翻译策略是：对于质量不高、时间要求不紧的翻译任务，采用众包协作模式，由业余译者或者有闲暇时间的专业译者免费做翻译，从翻译者方面看，他们的目的是练手或者交流；对于专业性较强、时间要求较紧的任务，还是选择专业译者，由专人把控进度。从译言网的分配模式可以看出，众包这种用高效率新方式、低价格高质量的模式是确实有可能成立的，但是操作上难度较大，原因包括但不限于信任、沟通成本、标准化等等。

根据上面的分析，可以看到，众包还是存在很多是线上的问题的，因此本课题决定具体设计一款移动众包应用程序，这对于跟踪目前流行的众包应用模式，拓展前景具有重要意义。

在实现技术上，目前大前端的概念特别火热：大前端通俗意义上就是所有前端的统称，例如ios、web、android等等，将用户直观能接触到的UI层统一起来，就是大前端。大前端相比原生开发的最大特点是在于一端开发，多端适用，它可以降低多端开发的成本。本毕业设计使用Flutter架构实现也旨在探索大前端在当前环境下的可行性，学习使用当今主流的移动UI框架。

## 1.2国内外研究现状

当今众包模式大为热门，国内外有着许多的探索与具体实现。在国外流行的众包在线合作平台一般发展历史都比较长，例如Upwork，是全球最规范严谨的人力资源平台，它的前身是成立于1999年的欧美人力外包市场Elance和成立于2003年的Odesk,两者握手言和，成为了世界上最优良的、范围最广的、最标准的、综合类人力外包效能平台。在Upwork上，接单有两种身份，个人模式和团队模式，用户可以自行选择。它的基本流程是客户在平台上发布具体的工作要求和需要的人数（也可以直接向指定的自由工作者发送邀请），然后自由工作者根据搜索得到任务，并且提交针对性的proposal，客户审查合格后发送相应的offer。Upwork平台主要经过平台抽成和用户开通包月会员进行盈利，这两种方式也是众包平台的主要盈利方式。又比如GrossOver，比较专注于给予高质量、长期稳定的远程工作岗位，它虽然优势在于提供长期稳定的工作岗位，但是它选拔程序员的方式是通过一种崭新的招聘模式-Global Hiring Tournament，即环球招聘巡回赛，通过类似于黑客马拉松的方式，从众包中选择精英，将众包的作用别样的展现出来。

在国内，流行的众包平台包括也有很多，比如：阿里众包、京东物流（尤在物流）、美团外包、众人帮等等。在国内的众包平台中，主要分为两类，一种是主线下类型，这种类型的平台相对较少，但是其中的佼佼者美团众包大众认知度却很高，这类线下众包主要是负责外卖配送等环节，众包能够最大程度的聚集大众的劳动力，再加上相应平台的智能算法，让整个外卖配送系统有条不紊；线上类型的平台相对较多，这类众包一般为一对多任务，通常是需要很多人完成的任务，用最低的成本得到目标的成果。笔者曾有一时间使用过主推线上众包的众人帮App，在该平台上，发布任务和领取任务的都是相对孤立的个体，任务体量较小，完成难度较低，且为了保障众包完成者的权益，需要预支相应的费用才可以发布众包；但是缺点也很突出，任务类型单一，多为绑卡注册推广类型（以获得被注册平台的推广佣金），由于相对门槛较低，因此任务完成度相对较低，但是由于任务的性质，同质化严重。

在Flutter使用上，国内主流的互联网公司已经开始接收使用甚至推广Flutter，例如阿里的闲鱼团队安卓端已经开始全线Flutter，并且开源了用于数据管理的基于Redux的组装式框架flutter\_redux,又例如字节跳动团队推出了西瓜视频的Flutter内测版本，并且推出相应的课程讲解flutter在其团队的应用。

不过通过他们的具体使用都可以看出，他们目前只是使用Flutter做一些小产品的尝试（闲鱼西瓜虽大，但是对于企业的体量来说，确实很小），只是处于初探阶段。在国外，由于Flutter是谷歌退出，因此国外使用Flutter开发的App更多，比如InKino、Orientation application、Toughest等等，由于这些应用国内都不可以使用，因此这里不多做赘述。

## 1.3 本文的主要工作

本文即侧重于众包的具体实现方式，又侧重于Flutter在移动UI开发中的可行程度，因此主要分为两部分：

第一部分是具体阐明众包的概念在当下的应用场景，在系统的实现时考虑众包的具体实现，跟踪主流的众包模式，探究众包的未来发展前景；

第二部分则是基于Flutter的系统开发的具体设计与实现，首先对当今多个大前端UI框架进行了分析，阐述了选择Flutter的理由。之后介绍了具体的实现过程，其中包括了通过模块图分析的功能需求、通过活动图和类图展示的详细设计、实现过程中遇到的特殊难点等。

## 1.4 论文的组织结构

本文共分为七章，每章的主要内容如下：

第一章是绪论，介绍了本文研究领域的背景，对国内外研究现状进行了简要的介绍，随后描述了本文的主要工作，最后说明了本文的组织结构。

第二章讲解了本文开发系统相关技术，对它们进行了分析，主要对比了当今主流移动UI框架的优劣以及框架的选择，确定技术选型。

第三章为需求分析，从可行性、功能需求、非功能性需求三个方面进行了分析，确定了系统的最终目标。

第四章为系统设计，首先讲解了相应的设计原则，然后从总体设计、数据库设计、用户功能层设计等多个层面进行详细设计分析。

第五章为系统实现，实现层面主要偏向于Flutter移动端，讲解了系统实现过程中的构建的基础组件与插件，再分功能描述了实现细节。

第六章为系统测试，主要采用黑盒测试，从稳定性、功能性、安全性等三个方面进行测试，部分测试详细描述了测试用例或者方法。

第七章为总结与展望，总结性的说明了本系统的实现效果与对众包的探究成果，以及对于未来工作的展望。

# 第2章 系统相关技术分析

## 2.1 移动UI框架

移动开发技术一直在不断进步，原始的原生开发由于要分为android、ios、web三个方向，一直面临着开发成本高的问题。为了解决多端分别开发带来的高成本问题，许许多多的跨平台技术涌现出来，比如H5+原生、js+原生渲染、自绘＋原生等。本节将对上述三种方案的具体框架进行分析，对比其优劣，并阐明选择Flutter的理由。

### 2.1.1 Flutter（自绘+原生）

Flutter是Google发布的一个用于创建跨平台、高性能移动应用的框架[14]，可以快速在ios和android上搭建高质量的原生用户界面。它最初以“sky”的名字亮相于2015年的Dart开发者峰会，并在2018年12月5日正式发布1.0版本。Flutter没有利用原生控件，而是本身利用dart实现了一个自绘引擎，并且使用自身的布局绘制系统。它号称可以书写一端，运行三端（android、ios、web），基本上得到了实现。

然后分析一下Flutter的优缺点。优点上，①由于它自绘界面，不需要调用原生界面，因此运行速度较快；②为了缩小包的大小，不支持热更新，由此不会被苹果封禁（苹果禁止热更新）；③生态丰富，根据github上Flutter项目的数量来看，Flutter的活跃用户正在快速增长，文档资源也逐渐丰富。缺点上，①它需要重新学习dart，有着一定的学习成本；②Flutter还没有成为主流开发框架，大多数中小公司还是处于初探阶段。

### 2.1.2 React Native（js+原生渲染）

React Native（下面简称RN）是Facebook在2015年发布的跨平台移动应用开发框架，是脸书先前开源的js框架React在原生移动应用平台的衍生产物。RN目前支持android和ios两个平台，采用响应式编程，尽最大可能的减少内存的使用。

接着分析一下RN的优缺点。优点上，①RN使用javascript语言，css以及jsx来开发移动应用，对于原来有web端开发基础的人很友好，减少了学习成本；②生态上目前还是主流的跨平台开发框架，被接受度高；③将虚拟DOM映射为原生控件，运行速度较快。缺点上，①将虚拟Dom映射为原生组件有一个过程，会降低运行速度；②近来传出FaceBook官方要重写RN的消息，现在学习RN的意义受到质疑。

### 2.1.3 HyBrid（H5+原生）

这类框架的实现原理就是将需要动态变更的界面内容通过H5来实现，然后通过原生的webview（Android）或者WKWrbView（ios）来加载。这样一来H5部分是可以动态修改的。在这方面还有一些其他的代表，比如微信小程序，不过微信小程序不是采用的原生渲染，但是将来可能会使用原生渲染。

最后来分析一下HyBrid的优缺点。优点上：①学习成本极低，原生web开发后适配三端；②动态化最简单直接。缺点上：①运行速度慢，每次运行都需要先下载网页，然后解析加载，特别是当网络环境差时，速度极慢；②部分原生api无法调用。

### 2.1.4 对比

选择框架时主要从运行速度、开发成本、可动态性、生态等方面分析：在运行速度方面，Flutter>RN>HyBrid，特别是Hybrid在运行速度上存在较大问题，直接舍弃。开发成本上，对于我个人既没有web基础也没有dart基础，Flutter和RN的学习成本相同，生态和可动态性两者也大致相同，再加上Flutter当今迅猛发展的态势，因此笔者决定选用Flutter作为开发的移动UI框架。

## 2.2 服务器技术选择

在本文的系统中，选择了Springboot作为后台开发框架，mysql作为数据库，下面详细的介绍二者。

### 2.2.1 Springboot

Springboot是Prvotal团队提供的用于简化spring新应用的最初搭建和开发过程的框架。Springboot框架有两个非常重要的策略：开箱即用和约定优于配置。开箱即用，是指开发者可以通过在pom.xml文件添加相关依赖，来替代繁琐的XML配置文件。约定优于配置，是值springboot本身搭建配置架构，由开发者填充信息的一种软件设计范式。这两点使开发人员避免了繁琐复杂的配置工作，将大量配置信息自动化。

数据库操作框架选择了JPA（Java Persistence API，java持久层API），它是jak5.0注解或者xml描述对象-关系表的映射关系，并且将运行期的实体对象持久化到数据库中。他相对于其他框架，具有操作方面、标准化、具有高级特性的诸多优点。

### 2.2.2 MySQL

MySQL属于关系型数据库管理系统（RDNMS ，Relational Database Management System），是Oracle旗下产品。MySQL是当今最流行的关系型数据库管理系统之一，它使用c和c++编写，并且使用了多种编译器继续测试，保证了源码的可移植性。与其他数据库相比，它具有以下几点优势：

1、MySQL开源源代码且提供免费版本，减少了小型开发团队和个人开发者的使用成本。开源源代码保证了其安全性，提供免费版本无疑成为初学者及小型开发团队的首选。

2、MySQL使用简单，对用户友好。

3、性能卓越、服务稳定，很少出现异常宕机。其核心线程是完全支持多线程，支持多处理器，通过高度优化的类库实现SQL函数，通常在查询初始化后不会有任何内存分配，没有内存漏洞。

当然，mysql也有着安全系统复杂、不支持热备份等问题，但是对于本文，mysql无疑是最好的选择。

## 2.3开发及运行环境

### 2.3.1 运行环境

前端：安卓手机要求系统android4.1或者更高版本，目前市面上的手机基本都能满足该要求。苹果手机ios版本不做要求。

在运行时，会申请相应的权限，如读写文件（用来保存上传图片，如头像和线上任务图片）、定位权限（百度sdk需要）等，请同意使用，否则部分特性无法使用。

后端：需要web服务器Tomcat，以及mysql8.0及以上版本。

### 2.3.2 开发工具及配置

首先下载安装JAVA sdk1.8，并且在环境变量中进行配置。Flutter开发软件使用的android studio，它对flutter开发提供了支持，详细搭建环节请参考：<https://book.flutterchina.club/chapter1/install_flutter.html>，另外本次开发使用的flutter版本为1.12.13，dart版本为2.7.0。

## 2.4 本章小结

本章对于移动众包手机应用系统涉及到的技术进行了说明，首先在移动UI框架选择上，对比了当今流行的几种跨平台框架的优缺点，并且在个人水平基础上决定选择使用Flutter作为移动UI开发框架。然后介绍了后端开发选择的服务器框架springboot，并且对移动众包手机应用系统使用的数据库mysql进行介绍，上面的这些技术组合为本文系统的实现奠定了基石。最后介绍了相应的环境配置等。

# 第3章 需求分析

## 3.1需求分析概述

需求分析是软件开发生命周期中最重要的一步，它叙述了目标系统的行为、属性或者特性，是开发过程中对系统的要求。移动众包手机应用系统的主要目的是为了综合当前的众包使用场景，覆盖线下线上两种场景，让更多的人可以参与到众包当中。具体上来说，移动众包手机应用系统必须能够满足以下几点基本需求：

1、注册登录，修改维护个人信息，登录接入第三方。

2、用户可以进行线下任务的查看、发布、接收、完成，以及相关的位置的保存和通过百度SDK周边位置poi检索。

3、用户可以进行线上任务的查看、发布、接收、监控、提交、审核，其中任务的各个环节都需要用户友好。

4、发布任务的双方可以进行即时通讯，合理沟通。

5、支持任务绑定的相应金额扣除，充值等。

另外本系统前端作为移动端软件，需具备移动端App主题多样化的功能要求。

## 3.2 可行性分析

### 3.2.1 技术可行性分析

移动端采用Flutter，目前Flutter已经基本成熟，很多企业已经利用Flutter开发出可以实际应用的软件，并且当前Flutter的生态较为丰富，在github上搜索Flutter已经有111865个Flutter相关库（截止到2020年4月22日），而且android studio 已经充分支持Flutter开发，热更新等内容也已经实现。然后一些相应的定位等难点已经有第三方SDK支持，只需要编写相应的插件就可以（在Flutter

中，插件是指dart与原生代码交互的一种方式）。使用的后台框架Springboot目前也已经相当成熟，被应用在很多web系统中。根据上面的分析，我们可以明白，用当前的技术实现移动众包手机应用系统是可行的。

### 3.2.2 操作可行性分析

该系统基于c/s架构，用户只要安装相应的app就可以使用，且界面简单通用，Flutter展现的最终效果与原生实现并没有明显差异，只要有一定智能手机知识的人都能够轻松的使用，不需要人额外培训。相关的功能都遵从当前主流系统的开发要求，符合大众认知。

### 3.2.3 经济可行性分析

尽可能用最少的投入，得到性能、技术、安全等各方面都满足需求的系统，是开发者必须追求的。本系统成本主要在于前期开发和相应服务器的购买或租用，开发使用Flutter，开发一端，三端可用，极大的降低了前端的开发成本，使用的数据库免费且占内存小，也极大地降低了对服务器的要求。通过本套系统，可以通过相对较少的投入，极大的促进众包在现实生活中的应用，提高人们生活的便捷性，充分利用大众的创造力，而且我们可以通过中间抽佣的方式盈利（目前大多数众包平台也都是使用的这种方式），从经济上来说是完全可行的。

### 3.2.4 总结

通过上面的分析，我们可以知道移动众包手机应用系统的开发在经济、技术以及操作上都是可行的，因此可以着手开发移动众包手机应用系统。

## 3.3 功能需求分析

本节将从用户需求分析、总体需求和详细功能模块三个方面对本系统的功能需求进行介绍。

### 3.3.1 用户分析

根据对系统的不同要求，我们可以将用户分为两类：

1、众包发布用户：作为众包任务的发布者，可以利用系统APP发布线上线下任务。

2、众包完成用户：作为众包平台数量最多的用户，可以在平台App上接收、接收任务，并获取到相应的奖励。

对于移动众包手机应用系统，这两类有不同的功能的要求。根据两类用户对不同功能的需求分析可以得到：

1、任务发布用户：

1. 可以保存个人位置信息，用于线下任务的送达地点选择，同时对于不同位置有优先级的选择。
2. 可以发布线下任务，输入相应的线下金额，选择相应的拿取环节。
3. 发布线上任务，按照步骤的输入各类要求的信息，分别包括扫描二维码、输入网址、图文信息、上传截图、上传手机号，这基本包含了线上任务的全部要求。
4. 可以审核线上线下任务，给出审核不通过的理由。
5. 查看自己发布的任务，查看任务相关接收完成信息。
6. 金额充值。
7. 与接单者即时通讯。

2、任务完成用户：

1. 查看、浏览、接收线下任务，根据线下任务信息实现相应的代拿代取任务。
2. 查看、浏览、接收线上任务，查看线上任务的每一个步骤，可以查看相应的提示图和网址，最后上传自己的完成审核信息。
3. 查看自己已经接受的任务，查看历史人物的提交状态。
4. 金额提取
5. 与发布者即时通讯。

另外对于两者都应该具有的功能：注册登录，修改个人信息。

### 3.3.2 总体需求分析

根据移动众包手机应用系统设计与实现提出最初的设想，并且参照其他众包相关App的基本功能，对整个移动众包手机应用系统的功能上进行需求分析。根据软件工程的具体流程，首先例举出本系统需要完成的功能列表，之后逐个分析各个功能中的详细细节和要求，以及最后需要达到的效果的最终目标。根据当今众包线上线下的需要，充分结合模块化的设计想法，可以将本文的移动众包手机应用系统划分为：用户信息相关模块、线上任务模块，线下任务模块，即时通讯模块、主题管理模块等五个部分。

移动众包手机应用系统

用户信息相关模块

线上任务模块

线下任务模块

即时通讯模块

主题管理模块

图3-1 系统总体模块划分

### 3.3.3 详细功能模块分析

1、用户信息相关模块

用户信息相关模块主要是指用户的注册登录以及相应的修改个人信息，同时本系统要接入第三方登录（QQ），因此需要再次详细划分，具体如图3-2所示。

1. 密码登录：用户通过密码注册登录。
2. 验证码登录：用户通过验证码注册登录。
3. QQ登录：用户通过QQ注册登录。
4. 个人信息相关：查看、修改个人信息

用户信息相关模块

个人信息相关

密码登录

验证码登录

Q

Q

登录

图3-2 用户信息相关模块

2、线上任务模块

线上任务主要是负责线上任务的相关界面内容，包括：新增线上任务界面、查看线上任务界面、线上任务提交审核、历史任务查看，如图3-3。

线上任务模块

新增线上任务界面

查看线上任务界面

线上任务提交审核

历史人物查看

图3-3 线上任务相关模块

3、线下任务模块

线下任务相比于线上，提交环节相对简单，不需要额外的审批，但是新增了位置保存要求，故包括：位置信息保存、新增线下任务界面、线下任务查看界面，历史任务查看，如图3-4。

线下任务模块

位置信息保存

新增线下任务界面

线下任务查看界面

历史任务查看

图3-4 线下任务相关模块

3、即时通讯模块

即时通讯模块功能相对统一，主要包括收发消息，无需再次划分。

4、主题管理模块

主题管理模块与本次众包任务无关，而是移动app的固有设置，一个界面优美、可以适配多种主题的app才是一个合格的应用，主要可以细分为：颜色主题更换、黑暗模式适配、多语言适配（目前仅支持中英文）。

主题管理模块

颜色主题更换

黑暗模式适配

多语言适配

图3-5 主题管理相关模块

## 3.4 系统非功能性需求

系统非功能性需求并不涉及到整个系统的功能需求，目的只是为了保证系统可以稳定运行需要满足的要求。本节将会从“系统安全性”、“系统易用性”和“系统性能需求”这三个方面来进行系统非功能性需求的分析。

### 3.4.1系统安全性需求

系统安全性需求是基于保护数据通信安全、数据库数据安全出发的，本系统安全性需求具体包括以下几方面：

1. 密码MD5加密

数据库中的密码采用md5加密，可以防止由于数据库被攻击导致密码泄露。同时有必要对密码进行限制，增加密码安全性。

1. 通过token保存登陆状态，且token通过rsa加密

通过token加密，使得服务端不需要存储相应的登陆记录，且满足移动端需要长期保持登陆状态的需求，不放在session，减少了服务端存储压力。同时，为了防止token被拦截，需要使用混淆+Rsa加密。

### 3.4.2 系统易用性需求

系统的易用性需求主要表现在对界面以及一些操作的要求，操作界面要做到自然流程，逻辑清晰，突出重点，特定要着重显示众包的相关特点；要让用户方便操作，具有良好的交互性。

### 3.4.3系统性能需求

性能需求一般通过性能指标来显示，性能指标一般包括响应时间、吞吐量、并发用户等。

本系统移动端通过Flutter开发，由于Flutter界面绘制并不需要映射为原生，也不需要异地下载相关文件，因此界面较为流畅，基本可以媲美原生。服务端方面，采用的是个人电脑作为数据库，足以满足学生毕设所需要的服务器信息，数据库采用mysql，速度快，相关查询努力减少查询时间，能够满足系统的性能需求。

## 3.5本章小结

本章是对于移动众包手机应用系统的需求分析，首先介绍了本系统要实现的目的，然后从技术可行性、操作可行性、经济可行性三个方面分析并确认了本系统在当前环境下的可行性。之后本系统又从用户、总体等方面进行了功能分析，并且确认了本系统可分为用户信息相关模块、线上任务模块，线下任务模块，即时通讯模块、主题管理模块等五个部分；最后从系统安全性需求，系统易用性需求，系统性能需求三个方面分析了系统非功能性需求，为下一步系统的设计奠定了基础。

# 第4章 移动众包手机应用系统设计

确定好相关的需求，为了从整体上全面了解系统，对系统相关技术框架进行梳理

## 4.1 系统设计原则

移动众包手机应用系统的设计需要满足基本的基本原则，才能够设计出满足需求的系统，详细来说分为下面几个部分：

1. 系统开发流程需要严格满足软件需求功能的规范和要求。
2. 软件编码严格遵守规范，采用驼峰命名法。
3. 软件端的操作界面简单易用原则。
4. 系统安装部署的方便、快捷。
5. 移动众包应用手机系统作为初步探索项目，不需要很多的开发经费，其利用率高，极大的减少了人力成本。
6. 保证用户个人资料的安全性，最大努力不被非法入侵，盗取数据。
7. 支持用户即时通讯，增加交流的便捷性。

## 4.2 系统模型结构

移动众包手机应用系统设计的初衷是众包模式的应用，采用信息化、网络化的方式，集成主流众包应用场景。由上得到，移动中包手机应用系统可谓分为两端：移动端和服务端，移动端主要可分为用户信息管理、线上任务相关、线下任务相关、即时通讯模块，由于即时通讯采用第三方sdk，服务端少了相应的即时通讯模块，最后得出的系统模块图，如下图4-1所示。

图4-1 模型结构

## 4.3 总体设计

### 4.3.1 面向客户端的服务器接口设计

本文所阐述的系统前后端采用http请求交互，这里将相应的接口设计为符合RESTFUL API的格式规范，设计过程分为三部：

1. 将需要请求的数据划分为资源，并且提供定位URI来明确资源地址，比如<http://localhost:8080/user>代表了用户user在本web环境下的唯一位置。
2. 确定资源传输的类型格式，常见的数据格式有XML、json、txt等。
3. 对资源操作进行相关的定义，我们通常将CRID与restful架构相对应，而http的集中请求方式（GET、POST、PUT和DELETE）刚好对应着资源的四种操作：删除、更新、查询以及新增。相应的对应关系如下表4-1所示。

表4-1 Restful API对应关系

|  |  |  |
| --- | --- | --- |
| 请求方式 | 操作一个URI资源 | 操作多个URI资源 |
| GET | 获取某个资源的信息 | 获取某组全部资源的信息 |
| POST | 更换某个资源 | 更换整组的资源 |
| PUT | 修改某个资源 | 修改整组资源 |
| DELETE | 删除某个资源 | 删除某组资源 |

而在我们的移动众包手机应用系统中用到的Restful API由账号注册，账号登陆（其中多种登录方式采用同个API，后台进行区分），添加线上任务，接收线上任务，提交线上任务，审核线上任务，添加线下任务，接收线下任务，提交线下任务等等，这些Api全部使用HTTP方式请求，并且传递JSON格式数据，下表展示了主要API的设计细节。

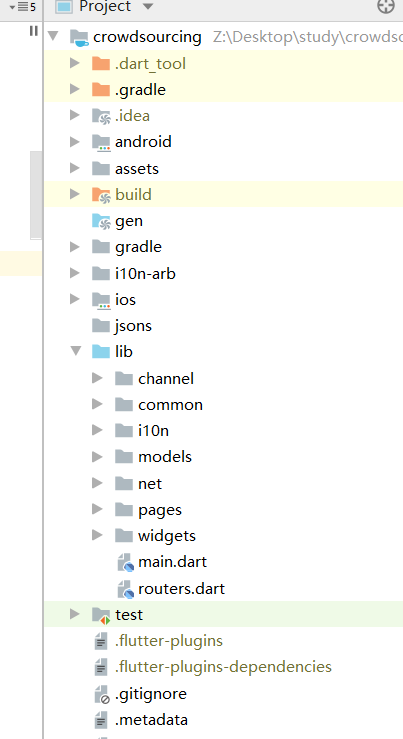
表4-2 Restful API设计

|  |  |  |  |
| --- | --- | --- | --- |
| API  http：//localhost：8080/ | 作用 | 请求方式 | 参数 |
| peoples | 获取所有的注册用户，测试用 | get |  |
| people | 注册 | post | people |
| people | 修改个人资料 | put | people,id |
| people | 获取个人资料 | get | id |
| location | 获取位置信息 | get | id |
| location | 新增位置信息 | post | id，location |
| location | 删除位置信息 | delete | locationid |
| location | 修改个人信息 | put | location，locationid |
| offineOrder | 获取离线任务 | get |  |
| ofineOrder | 新增离线任务 | post | offineOrder |
| offineOrdering | 新增离线任务接单 | post | id,offineOrderId |
| offineOrdering | 提交离线任务 | put | list |
| offineOrdering | 查看离线任务提交详情 | get | offineOrderingId |
| money | 充值金额 | post | money |
| money | 提现金额 | put | money |

### 4.3.2 项目结构设计

客户端采用的是Flutter常用的mvp架构，这点不同于经典的Android原生框架，因为安卓控制层与界面层是分离的，因此可以将两者分层，而在Flutter

中，我们控制视图变更的方式是数据驱动，控制层与两者紧密耦合，难以单独列出，因此我们采用mvp架构，Presenter层把Model和view层完全分离，主要的逻辑程序在Presenter里面实现。本文使用android studio开发客户端，将界面信息放在page目录下，模型文件全部放在model文件，语言相关文件放在i10n-arb文件下，加强了对代码的管理，提高了开发速率，降低了后期维护的难度。

图4-2

### 4.3.3 跨组件状态共享

在Flutter中，状态管理是一个十分重要的话题。通常我们会让组件管理自己的状态，但是当某些状态是要多个组件共享时，这时就需要有他们的父组件来管理相应的状态。但是由于这样管理状态很复杂，因此涌现出了许许多多的管理方式，如Flutter的全局事件总线EventBus，跨组件状态共享Provider。在对状态共享做分析后，发现provider相对简单，eventbus更加灵活，但是本应用没必要使用eventbus，因此决定provider。

Provider的原理来自于Flutter提供的一个组件：IneritedWidget，这个组件可以绑定其与子组件的依赖关系，并且当其数据发生变化时，会自动的更新依赖其的子组件。Provider利用了这种特性，并且充分发挥，使用单一用例来做到状态共享。

## 4.4 数据库设计

本文设计的系统数据主要存储在后端，需要保存用户信息、任务信息等，这些信息都存放在占用内存较小的mysql数据库，如图4-3是本系统的基本E-R图。

本数据库设计过程中，遵守了数据库设计的第一范式、第二范式和第三范式，尽可能的减少冗余。

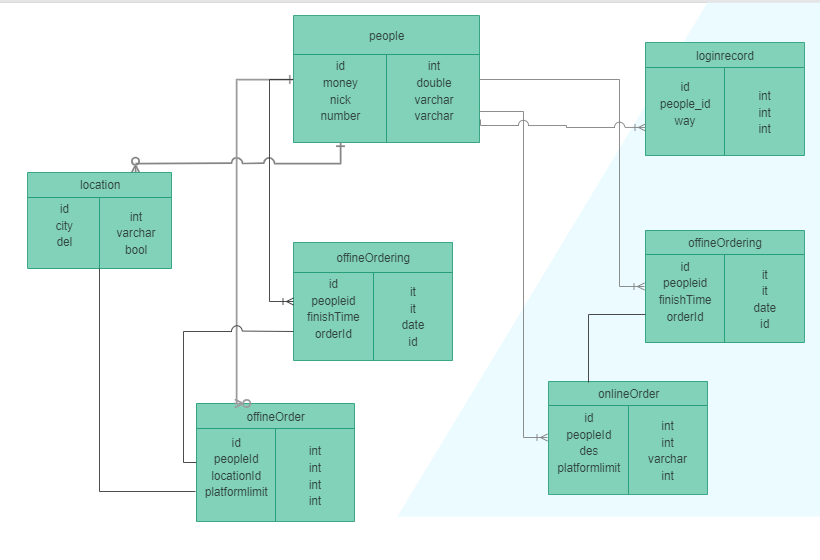


图4-3 数据库ER图

### 4.4.1 用户信息表

用户信息表存储用户相关信息，包括id、手机号、密码、创建日期、头像、余额、昵称、QQ、token以及性别，其中token用于即时通讯时作为融云的身份码，需要我们上传融云我们的昵称、id以及头像。具体信息如下表4-3所示。

表4-3 用户people表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段中文名 | 类型 | 长度 | 说明 |
| id | 编号 | int | 4 | 主键 |
| date | 时间 | datetime | 8 | 新增用户时间 |
| gender | 性别 | varchar | 255 | 直接存储男女性别名词 |
| head | 头像 | varchar | 255 |  |
| money | 钱 | double | 8 | 余额 |
| nick | 昵称 | varchar | 255 |  |
| number | 号码 | varchar | 255 | 手机号 |
| pass\_word | 密码 | varchar | 255 |  |
| qq | QQ | varchar | 255 |  |
| token | 标记 | varchar | 255 | 融云即时通讯返还的身份标识，对应着我们的头像和id |

### 4.4.2 位置信息表

Location表是用于存储位置信息的，对于发表离线任务起辅助作用，类似于淘宝购物的收货地址，主要是考虑到收货地址比较常用，且相关信息必不可少，因此决定单独列举出来。具体信息见下表4-4。

表4-4 位置location表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段中文名 | 类型 | 长度 | 说明 |
| id | 编号 | int | 4 | 主键 |
| created\_time | 时间 | datetime | 8 | 创建位置时间 |
| city | 城市 | varchar | 255 |  |
| del | 删除 | bit | 1 | 代表是否删除该位置 |
| main | 主要的 | bit | 1 | 是否为默认收货地址 |
| others | 其他 | varchar | 255 | 详细收货信息 |
| number | 号码 | varchar | 255 | 收货手机号 |
| people\_id | 用户id | int | 4 | 外键，对应用户表id |
| plot | 区 | varchar | 255 |  |
| province | 省份 | varchar | 255 |  |
| street | 街道 | varchar | 255 |  |

### 4.4.3 登录信息表

在本系统中，我们使用token保存登陆状态，避免了用户反复传递账号密码信息，增加了系统的安全性。Token通过rsa加密算法进行加密，内涵相应的用户id。但是为了做到单点登录，我们必须保存相应的token状态来保存相应的唯一登陆状态，在同一时间下，一个用户只可以有一个未结束token。login\_record表详细字段见下表4-5。

表4-5 标记login\_record表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段中文名 | 类型 | 长度 | 说明 |
| id | 编号 | int | 4 | 主键 |
| created\_time | 创建时间 | datetime | 8 | 创建token  时间 |
| finish\_time | 结束时间 | datetime | 8 | token无效时间 |
| people\_id | 用户id | int | 4 | 外键，对应用户表id |

### 4.4.4 离线任务表

离线任务通常为代买代拿代办任务，因此需要一定的位置信息以及购买方式等，送达位置我们考虑到常用性单独进行了划分，但是中途的购买信息存储存在一定的难度，因此我们决定将其转化为json格式数据段，在app获取数据后再自行解析，降低了后端服务器的存储难度。

表4-6 离线任务offineOrder表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段中文名 | 类型 | 长度 | 说明 |
| id | 编号 | int | 4 | 主键 |
| buy\_messages | 购买信息 | varchar | 255 | 代拿代取信息，json格式 |
| des | 描述 | varchar | 255 | 必填，40字以内 |
| end | 送达地 | int | 4 | 对应送达位置location的id，当为-1时代表不需要送达 |
| finish | 结束 | int | 4 |  |
| limited\_time | 截止时间 | varchar | 255 | 可为无限制 |
| people\_id | 发布者id |  |  |  |
| platformlimit | 平台限制 | int | 4 | 0代表无限制，1代表限制安卓端，2代表限制ios端 |
| price | 价格 | double | 8 | 这里特指赏金，需要区分的是，在buy\_messages也有price，指的是购买物品的价格 |
| title | 标题 | varchar | 255 |  |
| require | 要求 | varchar | 255 |  |

### 4.4.5 离线任务接单表

由于线下任务不需额外提交信息，因此离线任务的接单表相对简单，仅仅记录开始结束时间和双方身份。具体信息如下表4-7。

表4-7 离线任务接单OffineOrdering表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段中文名 | 类型 | 长度 | 说明 |
| id | 编号 | int | 8 | 主键 |
| created\_time | 创建时间 | datetime | 8 |  |
| finish\_time | 结束时间 | datetime | 8 |  |
| people\_id | 用户id | int | 8 | 外键，对应用户表id |
| offineorderId | 离线订单id | int | 8 |  |

### 4.4.6 线上任务表

线上任务与线下任务相差较大，但是我们为了方便灵活，都在中间增加了可自定义的东西，线下任务是代拿代取中间地，而线上任务就是我们的一个个步骤，这种步骤内容可以自定义，因此很难转化为固定的表结构，因此我们决定采用和线上任务相同的方式，将这些信息转化为json，在使用时再进行相应的转换。除此之外，线上任务可以多人完成，因此需要记录相应的数量，使用total表示总名额，remain表示剩余名额，finish代表完成名额，submit表示待审核名额，这样就能清晰的分出各类名额的个数。详细信息见下表4-8。

表4-8 在线任务onlineOrder表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段中文名 | 类型 | 长度 | 说明 |
| id | 编号 | int | 4 | 主键 |
| onlinesteps | 步骤信息 | text |  | json格式 |
| des | 描述 | varchar | 255 | 必填，40字以内 |
| total | 总数 | int | 4 |  |
| finish | 结束 | int | 4 |  |
| limited\_time | 截止时间 | varchar | 255 | 可为无限制 |
| people\_id | 发布者id |  |  |  |
| platformlimit | 平台限制 | int | 4 | 0代表无限制，1代表限制安卓端，2代表限制ios端 |
| price | 价格 | double | 8 |  |
| remain | 剩余名额 | int | 4 |  |
| title | 标题 | varchar | 255 |  |
| require | 要求 | varchar | 255 |  |
| submit | 提交待审核 | int | 4 |  |

### 4.4.7 线上任务接单表

线上任务接单信息需要提交一定的数据，比如图片、电话号码等，为了方便的存储，我们仍然采用前面介绍过的json格式存储。同时线上任务增加了审核的过程，众包发布者可以拒绝通过并给出原因。具体信息见下表4-9。

表4-9 在线任务接单OnlineOrdering表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段中文名 | 类型 | 长度 | 说明 |
| id | 编号 | int | 8 | 主键 |
| created\_time | 创建时间 | datetime | 8 |  |
| finish\_time | 结束时间 | datetime | 8 |  |
| people\_id | 用户id | int | 8 | 外键，对应用户表id |
| onlineorderId | 在线订单id | int | 8 |  |
| resources | 提交资源 | varchar | 255 | json格式 |
| submit\_date | 提交时间 | datetime | 8 |  |

## 4.5 用户功能层设计

### 4.5.1 即时通讯模块

由于众包发布者和接收者后期需要就一些有疑问的地方进行交流，因此有即时通讯这一方面的需求。由于即时通讯需要的后台较为繁琐，因此我们决定使用第三方框架，而这方面具有Flutter包的中融云做的最好，因此我们决定使用融云。

在数据库用户表的描述中我们就了解到融云通过token识别对应的id、用户昵称以及头像，这样在交流时客户端就可以不用下载太多的内容而正确的展示。

由于本文设计系统重点在于众包分发，而非社交，因此本文没有设置账号搜索查找功能，而是在每当众包接单者接受任务时，自动给众包发布者发送一条消息，建立两者至今的会话，努力做到简单高效。

### 4.5.2 用户管理模块

用户管理模块主要涉及用户的个人信息，也就是使用本文系统的第一步，主要包括用户注册登录、信息管理等。接下来将描述这两个部分的设计。

1. 用户注册登录。

注册登录是使用本系统的必须环节，因此要尽可能的简化相关流程。在这里我们忽略了相应的注册环节，第一次登陆时若发现没有用户信息，则代表默认注册加登录。同时为了防止每次登录都重新输入用户名密码带来的网络传输风险，本系统采用token代表登陆状态，通过对用户id混淆后再加密组成token，用户每次访问携带token便可以代表身份。进一步，本文的系统不允许同一账号多点同时登陆，如果多点登录，只有最后访问的用户可以保持访问状态，之前的请求都会返回417状态码。为了做到单点登录，必须对token加以存储，以便保持单一用户未完成的token唯一性。

图4-4是相应的流程图。

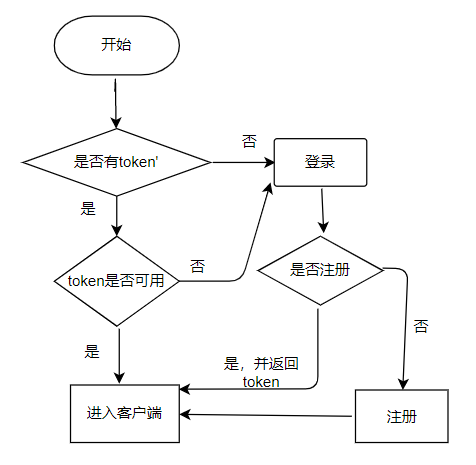


图4-4 用户登录流程图

1. 信息管理

信息管理较为简单，普通属性只需要传递相应的参数，难点主要在于头像的修改需要传输图片，对此本系统采取的设计是将图片上传设立单独的接口，并且返回保存的路径，这样在修改头像时，只需要首先上传图片到图片保存接口，再将返回的路径上传到单独的头像路径就可以。

### 4.5.3任务模块

任务模块包括线上任务与线下任务，两者即很相似，但是又有所区别。两者都具有标题、描述、限制描述、限制时间、限制平台、价格等属性，但是线上需要逐步引导，线下需要与实地交互。下面本文按照需求详细分析两种任务的设计过程。

1. 线上任务多为线上推广任务，而推广步骤多为复杂组合，需要一定的引导，同时还有一些特殊类型任务，与代玩、语音手机、翻译等等，为了方便存储与解析，又尽可能多的适配相应的类型，我们将步骤共分为五类：扫描二维码、图文演示、网址、上传手机号以及上传图片。对于复杂的类型，用户可以将存储相应信息的网址连接放到网址页面。
2. 线下任务多为代拿代买任务，与之相对应的就是一些位置关系，为此我们需要使用定位以及周边搜索功能，相关定位我们可以接入第三方package使用。

## 4.6 本章小结

本章对移动众包手机应用系统进行了具体设计，首先描述了系统设计的原则，然后从模型架构、总体设计、数据库设计等方面来对系统进行了分析，前后端双方交互API遵循Restful的原则，数据库设计详细描述了每个表的字段以及意义，基本将系统定型。最后从即时通讯、用户管理、任务三个模块分析了对用户功能层，也就是移动端的设计。

# 第5章 移动众包手机应用系统的实现

本章将对系统每个模块的具体实现流程进行一个较为详细的说明，其中会给出部分重要功能的具体运行截图界面。

## 5.1 基础功能块实现

在整体功能实现的情况下，适当的铺垫可以减少后期开发的难度，并且保持系统的统一性，便于维护。在Flutter中，基础的铺垫主要是一些自定义组件以及插件，自定义组件可以提供更适合系统的功能（实际上在Flutter中万物皆flutter，这里的自定义组件指那种可复用的基础性组件），而插件用于Flutter与原生系统进行交互。在介绍这些组件与插件的同时，我们会结合一些第三方package，描述其基础功能。

### 5.1.1 图片选择组件

图片的显示上传在本系统中比较常用，因此首先自定义了该组件。对于图片的选择，通常需要调起相机本地相册等服务，因此我们使用了第三方图片选择器image\_gallery\_saver，它可以主动调起图片选择界面，并且自动处理相关权限问题。为了保存选择的图片路径，我们使用了path变量进行存储。在显示时，需要能够显示多种图片，包括网络图片、本地图片、AssetImage等等，我们对这些类别分别进行了判断，当未指定imageProvider时，使用默认的本地库图片。

### 5.1.2 百度定位插件

线下任务必须要使用到相应的定位服务，因此百度定位插件提供相应的定位功能，但是由于百度地图并没有提供Flutter包，因此需要我们自己来实现相应的插件。我们使用的功能主要为定位以及poi搜索，当需要定位功能时，我们遵循以下的逻辑顺序，首先请求判断本地有没有位置（对于单用户，单次登录通常不会进行较大的位置变动，因此可以保留第一次请求的位置下次使用），若有则直接使用，若无则判断是否也有定位权限，权限的判断本文使用了第三方包permission\_handler，若获取权限则请求原生获取定位，原生获取到后会传回相应的位置信息。但是当手机位置信息关闭时，获取到的位置信息为空，也需要进行适当的提示。

具体流程图见下图5-1。

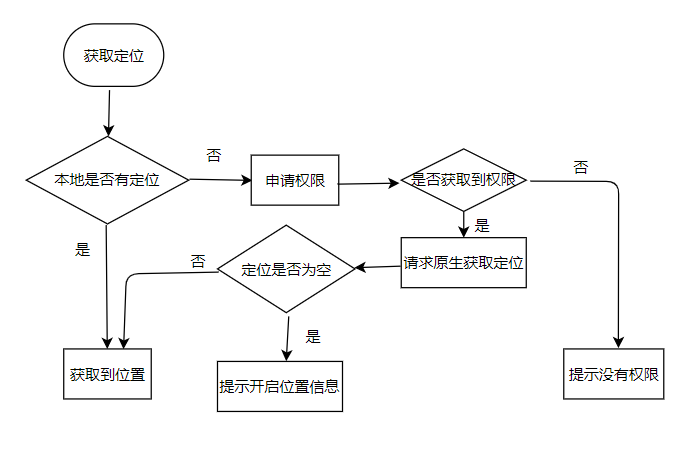


图5-1

### 5.1.3 QQ登录插件

QQ作为当今最流行的主流社交工具，接入QQ登录可以极大的减轻用户的登录负担，同时通过QQ登录可以获取用户的部分QQ资料，这样可以减少用户的信息填写时间。而目前腾讯官方只是发布了QQ登录的原生SDK包，因此也需要实现插件与之进行交互。

使用QQ登录功能首先需要在腾讯开发平台进行申请，填写相应的应用信息就可以使用。QQ登录获取需要分为两步，第一步调起QQ，这一步会返回给我们一个token以及用户个人识别码（非QQ号），第二部是获取用户个人信息，这里需要携带第一步获取的token作为用户识别。

在具体实现上，QQ登录只有在系统内有QQapp的时候才可以使用，因此检测系统是否安装QQ登录系列App（如QQ、TIM、QQ控件），若不存在，则不显示QQ登录按钮，否则显示。然后涉及到获取QQ个人信息时，考虑到本系统服务器信息与QQ用户信息冲突，因此仅仅在第一次登录，即注册时获取用户个人信息。

### 5.1.4 自定义Toast组件

Flutter本身提供了toast插件，它通过使用原生toast来发送通知，但是本系统需要实现一个功能更为丰富的能够使用Flutter原生开发的toast组件。

关于自定义toast组件，考虑到toast组件需要悬浮在所有组件之上，而Flutter对于MaterialApp内有一个overlay浮层，这个浮层在内部所有组件上方，可以用来实现我们的悬浮所有组件顶端的需求。

本系统实现的toast组件可以同时展示多条消息，并且根据消息的展示时间进行管理，为此自定义了ListNotify类，它继承了ChangeNotifier，同时内部可以放置一个存储消息类型的list，这样可以实现对弹窗消息的管理，并且可以在增加删除消息的同时进行overlay浮层的刷新。

## 5.2 客户端功能实现

### 5.2.1 个人信息模块

为了简化流程，本系统取消注册界面，采用首次登录自动注册的策略，设置了三种实现方式，密码登录、验证码登录、QQ登录等，同时登陆界面作为用户第一次打开APP直接看到的界面，其界面的美观自然对于用户的进一步使用有很大作用，因此登录界面努力做到让用户认可接收。

然后具体实现，验证码登录接入了bmob第三方的短信登陆，固定的短信模板（如图5-1）,bmob提供短信发送与验证功能，短信验证码的接入极大的降低了用户使用成本。每次接收验证码之后开启计时，60秒后才可再次获取。但是由于Flutter单线程的运行方式，保证了这种耗时操作下ui不会产生卡顿。

登录界面的具体效果如图5-2，5-3。

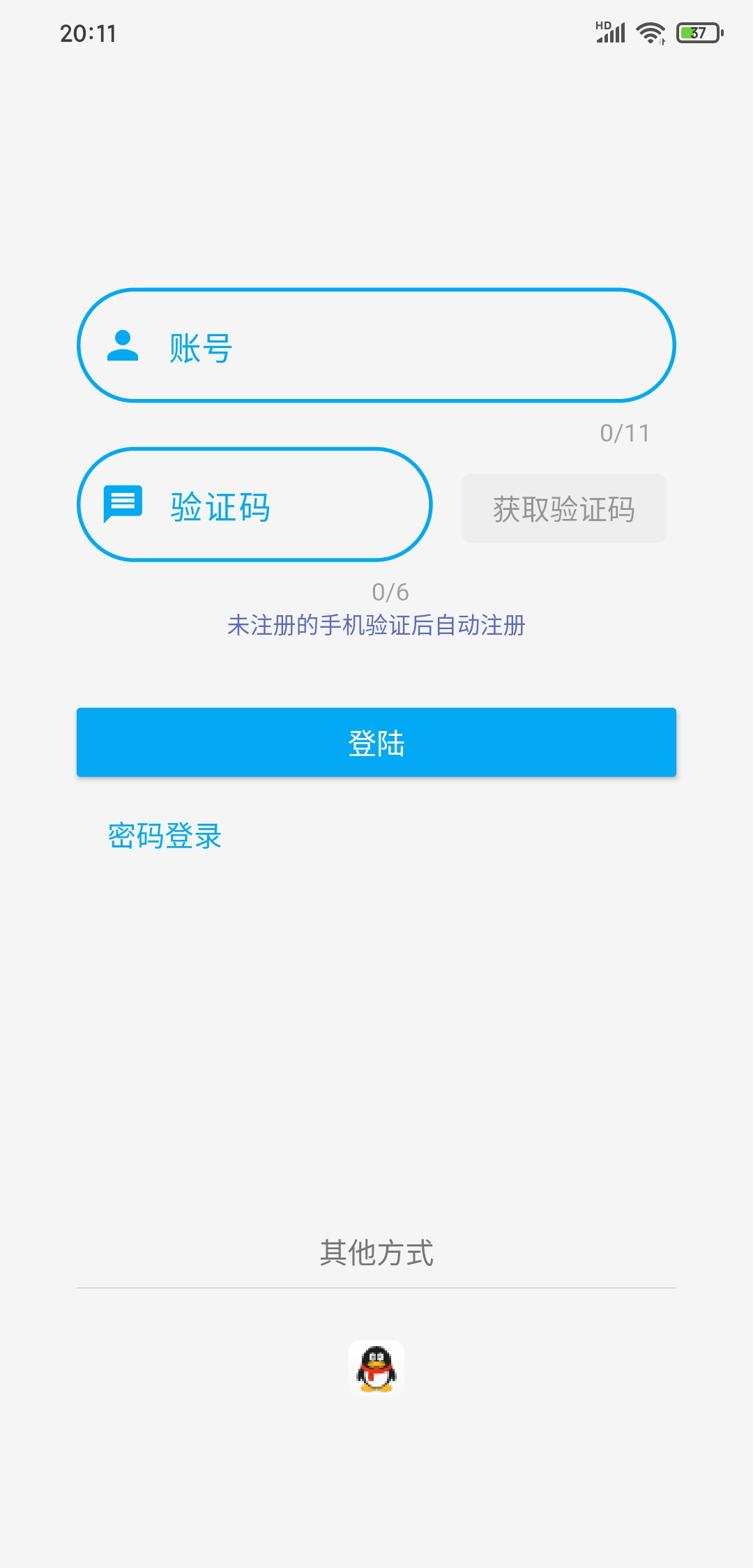
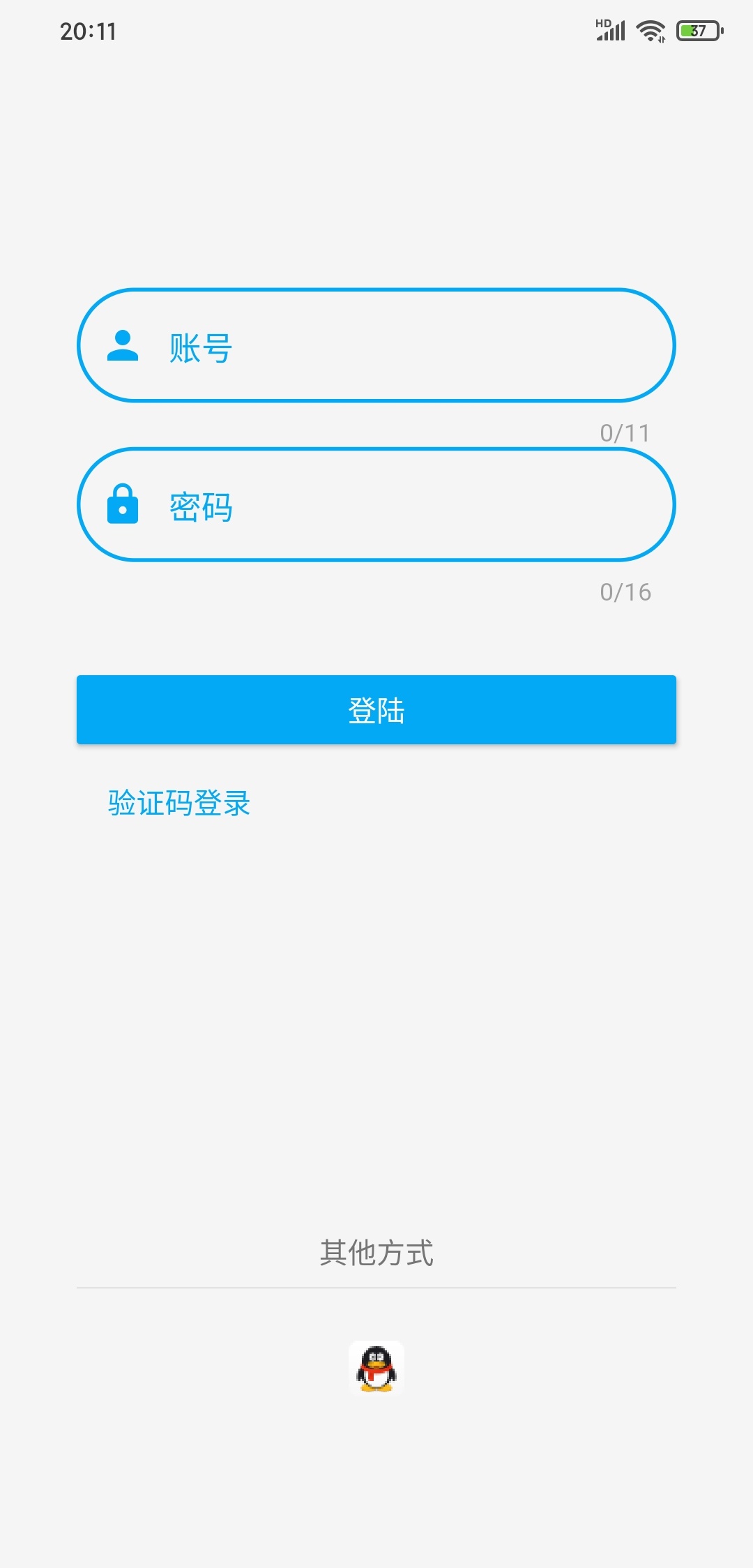


图5-2

验证码登录

图5-3

密码登录

在第一次登陆后，我们存储token到本地，并且在接下来的请求中将token放在消息头，用来保存登陆状态。在请求时会首先判断token可行性，如果不可以，会自动返回417状态码，同时弹窗提示用户身份过期，需要再次登录。相关的token我们采用了RSA加密，极大的保证了用户身份的可信性。并且token会被存在数据库中用来判断同一时间下一个用户只在一个地方登录，这样在最后登录的用户登录时，数据库会查找是否存在之前的未完成的token，如果有则设置结束时间，代表下线。下线的用户访问时会被判断token不可用。客户端请求携带token相关代码如下图5-4。



图5-4

登陆时，为了尽最大努力减少相应的接口，我们统一使用people接口，之后后台解析相关参数，判断具体使用了哪一种登陆方式，进入相关的逻辑处理。同时，与该登录用户相关的众包发布、众包接收信息也会被传输到客户端。这其中包括各种订单的数量，并会在用户信息界面展示。

在个人信息界面，涉及到一些逻辑处理，首先是展示了各类订单的处理数目，但是这个数目是在你的使用过程中不断使用的，因此我们使用了设计阶段展示的Provider组件，可以做到状态共享，不必再进行单独更新。

用户信息更改界面，也是采用统一的接口，不针对每个参数具体设置接口，这样减少了双方对接的复杂性。

个人信息界面见下图5-5，5-6。

图5-2

验证码登录

图5-3

密码登录



### 5.2.2 众包发布功能实现

无论线下众包还是线上众包，众包的发布是本系统的核心功能之一，因此众包发布的起始位置应该放在比较重要的位置，因此我们将发布按钮放在底部菜单的核心，如下图所示：

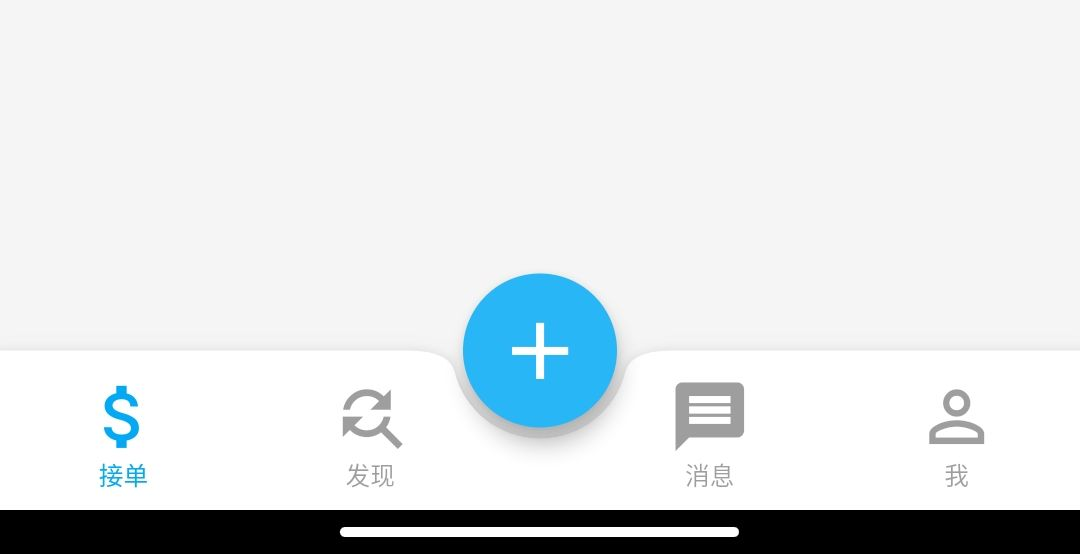


图5-4

实现上面的截图效果我们采用了Flutter提供的BottomAppBar与FloatingActionButton，FloatingActionButton是Flutter提供的Material设计规范的一种悬浮Button，我们可以将其放在BottomAppBar中的Row布局中，实现居中悬浮按钮。在实现具体的弹窗界面时，由于Flutter提供的弹窗无法自定义布局，或者遮盖底部，因此我们利用Flutter顶部的一层层叠布局，将相应的界面绘制在层叠布局顶部。

在线下众包的发布界面，只需要按照相应的提示顺序输入相应的逻辑信息即可，同时会选择默认的存储位置作为送达位置（默认位置选择主位置，若未设置主位置则选择第一个位置，若没有位置则提示新增），当然也可以直接选择不需配送，具体界面如图5-5。当选择中间送达位置时，可以选择就近购买和指定位置，指定位置我们接入了百度的poi搜索，这样可以更为清晰的展示相应的位置。由于中间位置有的需要指定位置，有的只是随机购买，因此电话号码等都是选填的，较为方便可行。

而在线上众包发布时，需要设置填写的信息相对较多，我们将步骤分为了：扫描二维码、图文演示、网址、上传手机号以及上传图片，对于不同的步骤类型，我们都设置了相应的定制界面，并且采用不同的定制界面。

在发布完成后会进入相应的预览界面，预览界面与最终用户看到的界面几乎一致，仅仅在于底部按钮不同。在预览界面，我们尽量将所有内容直接展开展示给用户，方便用户判断众包任务的难易以及与自身完成能力的匹配。同时我们注意保护用户的隐私，线下任务包含送达地，只有当你成功接单后才可以看到详细信息。以下图5-6为线下任务预览界面，图5-7为线上任务预览界面。



图5-6

线下众包

发布预览



图5-7

线下众包

发布预览



图5-5

线下众包

地点选择

### 5.2.3 线上众包审核功能

线上众包需要进行审核过程，因此需要提供给线上众包发布者审核查看界面。首先是需要对各类众包数目的一个总体审核，相应的界面已经在图5-2中显示。当点到具体的分类后，会看到该分类的众包列表，众包列表会展示众包的接单信息，包括已接人数、未接人数、等待审核人数，点击订单，可以查看的具体的订单信息。

审核界面我们考虑到审核的便捷性，因此对于众包信息不做过多的展示，只展示用户提交的信息和相应的标题，审核者可以进行查看，并且可以给出审核结果，如果审核通过直接点击通过，否则需要给出相应的理由。

## 5.3 主题相关

主题设置是本系统移动端的拓展功能，但也是充分利用Flutter特性的一部分。Flutter的主题、语言都可以在初始化MaterialApp时进行设置，并且当相应的主题改变时，会自动重绘整个App，但是不会丢失其他信息，简单的设置让主题设置变的简单易行。本系统移动端的主题设置包括颜色、语言、深色模式三部分，下面详细介绍实现方式。

MaterialApp通过指定theme与darkTheme字段来进行主题的设置，主题是一个叫做ThemeData的类，其可以指定包括主色调、色彩模式、文字形式在内的样式，我们可以让用户自己选择相应的颜色，利用provider跨组件共享的特征，实时刷新MaterialApp，这样相关的所有颜色主题等都会被改变，并且只是主题颜色重绘，并不会丢失现有数据。在主题设置上，我们设置了18色可选，只需要设置相应的颜色为设置的主色点就可以实现，相应界面如下图5-8。

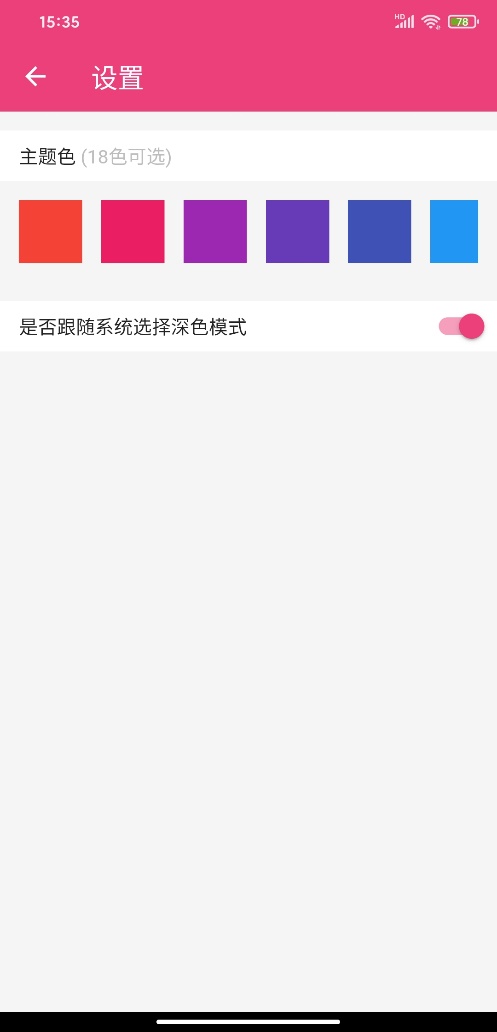


图5-8 多主体设置

在语言的适配上，Flutter也提供了支持，在MaterialApp可以通过supportedLocales设置相应的支持语言，并且设置相应的语言代理。在多语言生成上，我们使用了第三方包Intl，它可以轻松的帮我们实现国际化，并且可以将字符串分离为单独的文本。

## 5.4 服务端功能实现

服务端采用springboot结构， 通过各类注解自动搭建相应环境。在webconfig种配置相应的拦截处理，对token的由于进行判断，在各类controller中处理相应的请求，例如在errorController中对相应的错误请求进行处理，返回相应的错误代码。

个别难点在于token的处理并且放到相应的参数中，需要从请求的头信息获取相应的token参数，然后放到相应的请求参数中，为此，在token的拦截处理中，将token从头部信息中取出，再加入到请求参数中，这样就可以用于下一步的处理。但是这样参数是不可以被controller解析的，为此设置了CurrentUserId注解，只需要增加相应的注解就可以得到想要的参数。

下图为后端UML图。

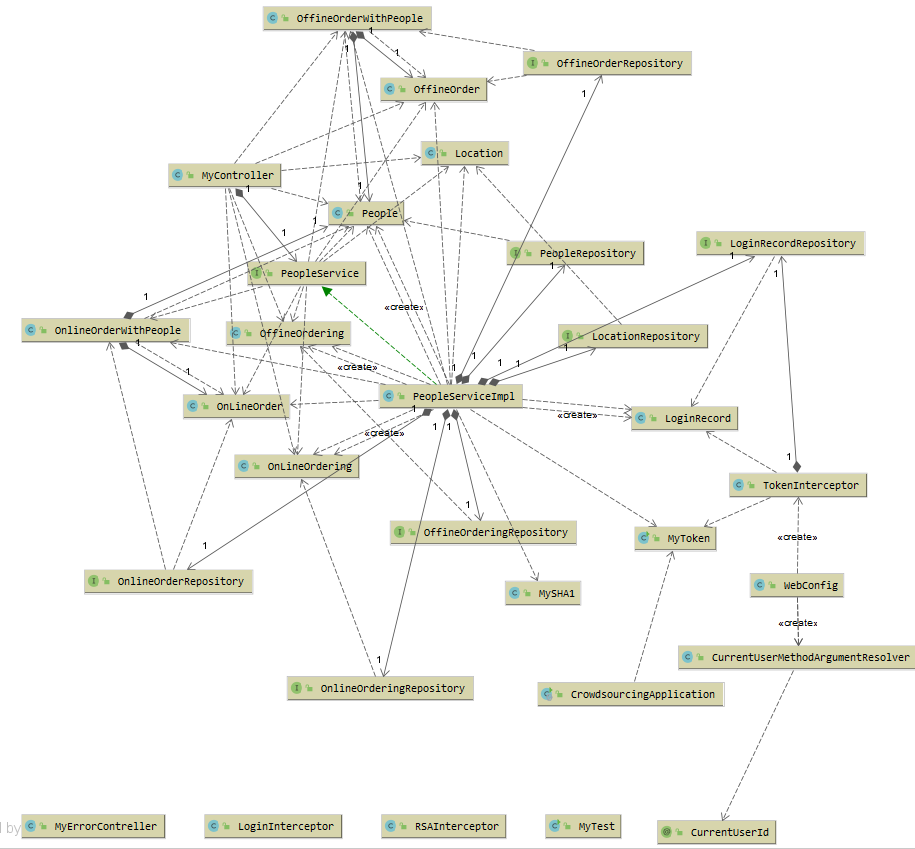


图5-9

## 5.5 本章小结

本章较为详细的讲解了系统前后端的实现过程，讲解了移动端的功能的不同组建方式，基础性组件、插件的完成减少了后期的开发任务，也为移动端的学习有了更深刻的认知。对于后端token的处理可以说作为本次系统实现的难点，自定义注释的使用合理的解决了这个问题，也给springboot注解实现原理有了更深的理解。

# 第6章 系统测试

为了确保本系统可以满足众包分发的用户需求，本章将会对移动端的主要功能进行多方面的测试，包括系统的稳定性测试、功能的完整性测试以及安全性测试。本章的测试描述都是在一部系统为安卓10.0的小米手机以及一台笔记本电脑作为服务器上进行的。

## 6.1 测试目的与原则

软件测试是检验软件质量的重要环节，该测试是有组织性的、计划性的，主要可以根据测试模式分为黑盒测试和白盒测试。软件测试的目的是对比测试的结果和代码的预期结果，分析其中错误的原因，解决其中的问题。

软件测试的原则：（1）软件测试工作应该拒绝由软件程序的开发者进行；（2）设计软件测试方案时，需要包含系统功能触发与其的全部测试结果；（3）测试用例应该留存，作为软件文档的组成部分；（4）程序的检查应该全面，包含该做与不该做两面。

## 6.2 系统稳定性测试

稳定性测试主要是看系统长时间运行时的运行状况，以及系统的容错性。在保持手机电量足够，客户端持续运行的情况下，持续三天每天进行3次功能测试，相应的测试下，客户端始终可以保持对服务端的正常请求，没有出现任何崩溃的情况。在容错性方面，我们进入主界面后关闭网络进行测试，系统可以给出合理的提示，对于本地保存的数据也可以进行正常展示，系统主题切换等与网络无关的方面也没有异常，因此本次系统稳定性测试通过。

## 6.3 功能测试

### 6.3.1 登录注册测试

对注册登录界面的测试，偏向于对于用户账号密码验证码等的多种组合进行测试，为此设计的测试用例如表6-1所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 预测结果 | 测试结果 | 错误原因 |
| 手机号正确，密码错误 | 无法正常登入 | 正确 |  |
| 手机号错误，密码正确 | 无法正常登入 | 正确 |  |
| 手机号正确，验证码错误 | 无法正常登入 | 正确 |  |
| 手机号错误，验证码正确 | 无法正常登入 | 正确 |  |
| QQ拒绝授予登录信息 | 无法正常登入 | 正确 |  |
| 用户名和密码都正确 | 可以正常登入 | 正确 |  |
| QQ授予权限 | 可以正常登入 | 正确 |  |

图表 6-1

### 6.3.2 线上任务发布

线上任务涉及到许多字段的填写，其中涉及到标题、描述、限制、价格、步骤、数目等许多字段，每个字段都有着相应的要求填写，下面对此进行相应的发布要求测试。

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 预测结果 | 测试结果 | 错误原因 |
| 标题为空，其他正确填写 | 无法发布 | 正确 |  |
| 描述为空，其他正确填写 | 无法发布 | 正确 |  |
| 价格为空，其他正确填写 | 无法发布 | 正确 |  |
| 步骤为空，其他正确填写 | 无法发布 | 正确 |  |
| 数目为空，其他正确填写 | 无法发布 | 正确 |  |
| 全部正确填写 | 可以正常发布 | 正确 |  |

图表 6-2

## 6.4 安全性测试

系统的安全测试目的是查找软件系统中存在的安全问题，下面采用几种常用的方式进行安全检测。

1. 静态代码检查

笔者对系统代码进行了代码走读的检查，进行了源代码安全性的测试，检查相应的数据流，保证系统可以安全运行。

1. 用户安全性测试

此项检查主要在于检查用户密码是否加密、是否可以复制，以及用户是否会出现冲突，此项检查通过

## 6.5 本章小结

本章首先对系统测试的目的与原则进行了介绍，然后从稳定性、安全性、功能三方面进行了测试，本系统都通过了上面的测试，符合软件安全可信、满足用户需求的要求。

# 第7章 总结与展望

## 7.1 总结

本文实现的移动众包应用系统基于当前主流移动UI框架Flutter，它在当前众包理论概念的基础上，参考了主流众包平台的经营理念，结合线上众包和线下众包，对于众包发展模式探索和Flutter开发探究都有着积极意义。在完成本系统过程中，我们首先进行了需求分析，确定了系统开发的目标；之后进行系统的设计，确定了相关的框架并在最后进行的实现充分结合Flutter特性。本系统可以实现用户信息状态的管理，用户可以通过移动端进行相应的众包发布发布接收过程，用户界面合理显示相关众包数量，简洁的审核界面简化相应的审核难度。

本系统充分结合Flutter特性，进行相应的主题、语言等的设置，18种主题色供用户选择，语言可以根据系统进行切换，增加了移动端的适用范围。此外，为了避免客户端和服务器之间用户名和密码的频繁传输，本系统使用了基于token的身份认证机制，将用户id通过rsa加密，确保了安全性。

在众包应用模式跟踪探索方面，感受到当前众包分配模式的高灵活性与业余性，它可以充分集合社会大众的创造性与劳动力，以较低的成本来实现需要大众才可以完成的任务。但是业余性、大众性也带来一些缺点，由于众包完成者来自于社会各界，由此带来的完成质量与新人问题就值得思考，因此众包的应用模式还是值得不断探索发展的。

## 7.2 展望

尽管本文设计实现的系统可以实现基本众包任务的发送与接收，基本满足了我们的需求，但是限于本人时间精力有限，在安全性、功能等方面还是有许多待完善之处，现列举如下：

1. 安全性原因主要来自于众包的业余性，用户之间的不信任导致有些众包任务难以继续，为了解决安全性问题，系统可以增加相应的实名认证功能，并且为了制约众包双方，可以要求双方提前支付一定的违约金（不过这又增加了众包的门槛）。
2. 功能上，线下空间可以增加地图功能，更加便捷的帮助用户选择位置，即时通讯方面可以增加消息范围，增加语音图片等类型，进一步降低交流双方的难度。
3. 设计上，由于对Flutter框架部分特性不是很了解，造成了部分类的过度耦合，可以充分利用provider以及Flutter其他特性进行重构，降低耦合性。

# 致谢

伴随着致谢的到来，这篇毕业论文进入了尾声，而我四年的大学时光也终于要结束了。时长三个月的毕业设计，持续不断的学习，系统的完善，论文的反复修改，都不断的锻炼着自己，也给自己的大学生活画上了一个句号。

回想整个论文的完成过程，自己从最初的懵懂到最后的熟练，这个过程离不开老师和同学对我的教导与帮助。首先我要感谢我的毕业论文导师何伟老师，何伟老师在整个论文写作过程给予我很多帮助，一直耐心和蔼的指导我。最初选题时，我还对题目一知半解，对系统的各部分更是毫无思路，何伟老师详细的给我讲解毕设题目，让我清晰了论文方向，在论文完成过程中，也详细分析我遇到的的各种细节方面的问题。

其次我要感谢大学四年的同学、朋友，或伴随或带领我从一个稚嫩的学弟成长为一个可以不断进取帮助他人的学长：搏创俱乐部带领我走入编程这座山，邵长旭、王子悦、刘奕辰三位部长的小课堂让我醍醐灌顶；张宇、于宏越等小伙伴与我一起学习成长；王浩浩、欧鼎魁、马钱虓三位可爱舍友的日常欢乐。大学生活的成长、磨练都离不开大家的陪伴帮助，四年同窗情，此生不忘。

最后我要感谢我自己，四年时间过去，我已不再是那个少年，过去的轻狂逐渐蜕化为沉稳，过去的稚嫩蜕化为成熟，往昔的种种经历，自己的每一次选择，也许有些许遗憾，也许有时候懈怠，但是更感谢过去的自己一直在努力进步的路上。

本次毕业设计遇到了许多的困难，无论是后端解析token到参数中，还是Flutter中状态管理的使用，都让自己苦恼了好久，好在自己一直坚持着，坚持探索，坚持尝试，最终得到了解决。刚刚过去不久的秋招让自己第一次接触社会，自己也跌跌撞撞最后找到了合适的归宿，希望未来的自己能够一直保持一个学生的心态，一直进步，一直拼搏，一直向上。

“凡我在处，便是山大；有你在时，那边是家”，时常哼出的这首歌伴随了我四年，不久我便要离开山大，但是我心中一直有山大，未来，我来了。

# 参考文献

1. Jiaxing Liu. Jammer Localization Approach Based on Crowdsourcing Carrier-to-Noise Density Power Ratio Fusion[C]. 中国卫星导航系统管理办公室学术交流中心.第十一届中国卫星导航年会论文集——S11 抗干扰与反欺骗技术.中国卫星导航系统管理办公室学术交流中心:中科北斗汇(北京)科技有限公司,2020:7.
2. Regan Linda,Hopson Laura R,Branzetti Jeremy,Gisondi Michael A. Transforming didactic conferences with live technology-facilitated crowdsourcing.[J]. Pubmed,2020,54(5).
3. 沈超,曹婷婷,王一萌,阙艳红,李文雅.基于React Native的农业气象自动化观测系统国家级平台移动应用APP设计[J].农业与技术,2019,39(18):132-135+138.
4. Todd R Weiss. Google Flutter Lets Developers Write Code Once for iOS, Android Apps[J]. SQL Server Pro,2018.
5. 陈耀庭,黄和亮.我国生鲜电商“最后一公里”众包配送模式[J].中国流通经济,2017,31(02):10-19.
6. 孙信昕. 众包环境下的任务分配技术研究[D].扬州大学,2016.
7. 庞建刚.众包社区创新的风险管理机制设计[J].中国软科学,2015(02):183-192.
8. 冯剑红,李国良,冯建华.众包技术研究综述[J].计算机学报,2015,38(09):1713-1726.
9. 周齐飞.基于Android平台的Hybrid App开发[J].电脑编程技巧与维护,2014(15):46-48.
10. 赵小龙,张为民.基于Android平台的智能监控风光互补控制器系统的设计与实现[J].阜阳师范学院学报(自然科学版),2014,31(01):59-62.
11. Claburn, Thomas. Google's Flutter Buy: Big Gesture?[J]. ProQuest,2013.
12. 张志强,逄居升,谢晓芹,周永.众包质量控制策略及评估算法研究[J].计算机学报,2013,36(08):1636-1649.
13. 宋小远,薛云志.一种Android平台混合应用运行环境[J]. 计算机系统应用. 2016(09).
14. 杜文.Flutter实战.机械工业出版社[M]，2020.
15. 亢少军.Flutter技术入门与实战[M]. 机械工业出版社,2019..
16. 郭霖.第一行代码第三版[M].人民邮电出版社，2020.
17. 林学森.深入理解Android内核设计思想[M].2017.
18. 郭宏志. Android应用开发详解[M]. 电子工业出版社, 2010.
19. Payet T , Spoto F . Static Analysis of Android Programs[C]// International Conference on Automated Deduction. Springer, Berlin, Heidelberg, 2011.
20. Wazny Kerri. "Crowdsourcing" ten years in: A review.[J]. Journal of global health,2017,7(2).

# 附录1 英文原文

“Crowdsourcing” ten years in: A review

Kerri Wazny

 University of Edinburgh  
ku.ca.de@ynzaw.irrek

**Abstract**

Background

First coined by Howe in 2006, the field of crowdsourcing has grown exponentially. Despite its growth and its transcendence across many fields, the definition of crowdsourcing has still not been agreed upon, and examples are poorly indexed in peer–reviewed literature. Many examples of crowdsourcing have not been scaled–up past the pilot phase. In spite of this, crowdsourcing has great potential, especially in global health where resources are lacking. This narrative review seeks to review both indexed and grey crowdsourcing literature broadly in order to explore the current state of the field.

Methods

This is a review of reviews of crowdsourcing. Semantic searches were conducted using Google Scholar rather than indexed databases due to poor indexing of the topic. 996 articles were retrieved, of which 69 were initially identified as being reviews or theoretically–based. 21 of these were found to be irrelevant and 48 articles were reviewed.

Results

This narrative review focuses on defining crowdsourcing, taxonomies of crowdsourcing, who constitutes the crowd, research that is amenable to crowdsourcing, regulatory and ethical aspects of crowdsourcing and some notable examples of crowdsourcing.

Conclusions

Crowdsourcing has the potential to be hugely promising, especially in global health, due to its ability to collect information rapidly, inexpensively and accurately. Rigorous ethical and regulatory controls are needed to ensure data are collected and analysed appropriately and crowdsourcing should be considered complementary to traditional research methods.

“No one knows everything, everyone knows something [and] all knowledge resides in humanity; digitalisation and communication technologies must become central in this coordination of far flung genius” [1]. Although examples of crowdsourcing and “wisdom of the crowds” have been reported hundreds of years ago [2,3], the term “crowdsourcing” was coined in 2006 by Howe in his Wired magazine article [4]. In the article, Howe defines crowdsourcing as “the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call” [4] and he adds that “crowdsourcing is the mechanism by which talent and knowledge is matched to those in need of it” to the definition in a later article [5]. Since Howe’s article and partially due to the availability of modern technology [6,7], use of crowdsourcing has skyrocketed [8]. Although research in this area has grown exponentially in the last decade, many authors feel that the potential of crowdsourcing is still underutilised and underexploited [5,9–11].

As crowdsourcing requires, depending on the definition, ‘outsourcing’ a task or tasks to a large crowd [12], advances in technology have facilitated the efficiency of this method [2,6,13–15]. Indeed, research that was previously inconceivable due to the scale is now achievable through crowdsourcing [6]. Kamajian states that 35% of smart phone users check their phones prior to getting out of bed and, as of 2013, over 5 billion people worldwide had access to mobile phones [14,16]. Prior to Howe’s Wired article, Louis van Ahn introduced the idea of human computing, where humans are used to solve complex problems that computers are not capable of [17]. While machine learning has made great strides, computers are poor at perception; humans can conceptualise, discriminate and filter, learn, adapt using their background knowledge and apply common sense and experience that machine are unable to do [18]. In addition to humans actively crowdsourcing data, ubiquitous computing, where computers exist through the physical environment, are virtually invisible to the user, and act as passive sensors has great potential for generating large amounts of data [16,18]. Cell phones, for example, can collect photo, video, acoustic, gyroscopic, acccelerometric and proximal information and can also be used to add pairing devices to collect additional information, such as pollution sensors [16]. Crowdsourced spatial analysis from GIS data can be very useful, especially for providing resources in emergency situations, for delivering logistics and for efficient targeting of interventions [19].

As individuals are biased towards the correct answer, Buecheler et al. estimate that if a million individuals were to contribute towards answering a problem via crowdsourcing, there would be a 97.7% likelihood that the crowd would arrive at the correct answer [20]. While pilot studies have not reached sample sizes close to that scale, many have had great success in achieving extremely promising results. For example, crowdsourcing has been demonstrated to produce accurate results across a range of medical diagnostic studies, including malaria, grading images for glaucoma and diabetic retinopathy, skin self–examination for skin cancers, and images for cancer polyps [21–26].

Despite the interest in the area of crowdsourcing exploding in the past decade, many authors do not agree on its definition or on what counts as crowdsourcing, with some academics considering Wikipedia a “classic” example of crowdsourcing, for example, while others insist it is not crowdsourcing [12]. Text and data mining is another example that is on the fringe of crowdsourcing’s definition.

In addition to there being many definitions of crowdsourcing, many authors have offered different taxonomies of crowdsourcing, some focusing on types of crowdsourcing while others focus on its production model. Furthermore, there are debates on who participates in crowdsourcing – whether it is laypersons, amateurs, professionals, experts, or a combination.

Although crowdsourcing has existed for decades, it is agreed upon that technology has facilitated its growth. Platforms such as Amazon Mechanical Turk and Crowdflower enable companies to hire workers to perform crowdsources exercises for extremely low prices. Other crowdsourcing platforms, such as Innocentive or Crowdmed, offer a competitive winner–takes–all model. Sensors in wearable technology have also facilitated the ability to collect mass amounts of information.

Crowdsourcing can increase the accuracy of computer automated tasks, lower costs, increase the scale of research, transcend boundaries and borders, produce novel discoveries and increase the speed of research progression, among other benefits. However, there are concerns with the generalisability of the samples, as the crowd is self–selected, security and data protection issues of sensitive data, and the possibility of malicious workers. Some studies have added quality protection measures to weed out malicious workers, such as adding cut–offs for scores on previous tasks and screening questions. Additional regulation is needed for ethical issues, such as obtaining informed consent and data use policies.

Crowdsourcing has considerable benefits in research, as it has the potential to substantially lower costs while massively increasing the sample size and researchers can receive the data in real–time [7,16,19,27–29]. Because of these qualities, crowdsourcing has potential to improve global health research. Indeed, crowdsourcing is used frequently to set research priorities in global health, most often in maternal, newborn and child health, due to the popularity of the Child Health and Nutrition Research Initiative’s (CHNRI) method of research priority setting which uses collective opinion to identify and score research priorities against a set list of criteria [30]. The CHNRI method is becoming the most frequently used research priority setting method due to its transparent, systematic nature; it was designed to capitalise on the principles of Surowiecki’s “Wisdom of the Crowd,” which will be described in the further in the paper [31]. Furthermore, research in global health faces an even larger burden than research in high–income countries with regards to funding, logistics, poor existing health care systems, health care workers to collect data, equipment, and patient access to health care, especially in rural or conflict areas [21,32–37]. As access to mobile phones in low– and middle–income countries is still increasing, crowdsourcing may provide a complementary route of data collection to traditional sources, capitalising on structures and knowledge already in place in the countries .

# 附录2 译文

“众包”十年：回顾

Kerri Wazny

 University of Edinburgh  
ku.ca.de@ynzaw.irrek

**Abstract**

背景

最早由Howe于2006年创造，众包领域呈指数增长。尽管众包的发展和超越，但众包的定义仍未达成共识，在同行评议的文献中，示例的索引也很少。在试点阶段之后，还没有扩大众包的许多示例。尽管如此，众包具有巨大的潜力，尤其是在缺乏资源的全球卫生领域。这篇叙述性评论旨在广泛地检索索引和灰色众包文献，以探索该领域的当前状况。

方法

这是对众包评论的评论。由于主题索引的质量较差，因此使用Google学术搜索而不是使用索引数据库进行语义搜索。共检索到996条文章，其中69条最初被认为是评论或理论上的文章。发现其中21篇无关紧要，对48篇文章进行了评论。

结果

这篇叙述性评论着重于定义众包，组成众包的众包分类法，适合于众包的研究，众包的法规和道德方面以及一些著名的众包实例。

结论

众包具有快速，廉价和准确地收集信息的能力，因此潜力巨大，尤其是在全球卫生领域。需要严格的道德和法规控制，以确保适当地收集和分析数据，并且应该认为众包是传统研究方法的补充。

“没有人知道一切，每个人都知道某事，并且所有知识都存在于人类之中。数字化和通信技术必须成为遥不可及的天才协调工作的中心” [ 1 ]。尽管“群众的智慧”已经报道数百年前众包的例子，[ 2，3 ]，术语“众包”是在2006年在他创造的豪连线杂志的文章[ 4 ]。在文章中，Howe将众包定义为“公司或机构采取的职能，即由员工执行一次职能，然后以公开电话的形式将其外包给不确定的（通常是大型的）人员网络” [ 4]，他补充说：“众包是一种将人才和知识与需要的人才相匹配的机制”，在随后的文章[ 5 ] 中也有定义。由于Howe的文章和部分由于现代技术[可用性6，7 ]，使用众包的猛增[ 8 ]。虽然这方面的研究在过去的十年中成倍增长，许多学者认为，众包的潜力仍未得到充分利用和开发不足[ 5，9 - 11 ]。

如众包需要，根据不同的定义，“外包”一个或多个任务到一大群人[ 12 ]，在技术进步已经促进了这一方法[效率2，6，13 - 15 ]。实际上，由于规模庞大，以前无法想象的研究现在可以通过众包[ 6 ]来实现。Kamajian指出，智能手机用户的35％，检查自己的手机下床，并为2013年之前，超过5十亿全世界人民不得不进入手机[ 14，16 ]。在Howe 连线之前路易斯·范·安（Louis van Ahn）在文章中介绍了人类计算的思想，其中人类被用来解决计算机无法解决的复杂问题[ 17 ]。尽管机器学习取得了长足的进步，但计算机的感知能力却很差。人类可以利用他们的背景知识来概念化，区分和过滤，学习，适应并运用机器无法做到的常识和经验[ 18 ]。除了人类积极众包数据，无处不在的计算，其中，通过物理环境中存在的计算机，是几乎看不见给用户，并充当无源传感器具有很大的潜力，用于产生大量的数据[的16，18]。例如，手机可以收集照片，视频，声学，陀螺仪，加速度计和近端信息，还可以用于添加配对设备以收集其他信息，例如污染传感器[ 16 ]。来自GIS数据的众包空间分析可能非常有用，特别是在紧急情况下提供资源，提供物流和有效地确定干预措施时[ 19 ]。

当个人偏向正确答案时，Buecheler等人。据估计，如果有一百万个人通过众包解决问题做出贡献，那么人群将有97.7％的可能性得出正确的答案[ 20 ]。尽管试点研究尚未达到接近该规模的样本规模，但许多研究在取得极为可喜的结果方面都取得了巨大的成功。例如，众包已被证明在一系列医疗诊断研究中，包括疟疾的产生准确的结果，分级为癌性息肉[青光眼和糖尿病性视网膜病，皮肤自我检查皮肤癌的图像，和图像21 - 26 ]。

尽管在过去十年中对众包领域发生了兴趣，但许多作者不同意它的定义或所谓的众包，例如，一些学者认为Wikipedia是众包的“经典”例子，而另一些人则坚持认为而不是众包[ 12 ]。文本和数据挖掘是众包定义边缘的另一个例子。

除了对众包的定义有很多外，许多作者还提供了不同的众包分类法，其中一些关注于众包的类型，而另一些关注于其生产模型。此外，关于谁参与众包的辩论也很激烈，无论是非专业人士，业余爱好者，专业人士，专家还是他们的组合。

尽管众包已经存在了几十年，但人们一致认为技术促进了它的发展。诸如Amazon Mechanical Turk和Crowdflower之类的平台使公司可以雇用工人以极低的价格进行众包演习。其他众包平台，例如Innocentive或Crowdmed，提供了一种竞争性的赢家－所有产品模式。可穿戴技术中的传感器还促进了收集大量信息的能力。

众包可以提高计算机自动化任务的准确性，降低成本，增加研究规模，超越国界，产生新发现并提高研究进展速度等诸多好处。但是，由于人群是自选的，因此样本的通用性，敏感数据的安全性和数据保护问题以及恶意工作人员的可能性令人担忧。一些研究增加了质量保护措施，以清除恶意工作人员，例如为先前任务的分数增加分界线并筛选问题。对于道德问题，需要其他法规，例如获得知情同意和数据使用政策。

众包在研究相当大的好处，因为它具有相当低的成本的潜力，而大量增加样本大小和研究人员可以在实时接收数据[ [7](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5735781/#R7)，[16](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5735781/" \l "R16)，[19](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5735781/" \l "R19)，[27](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5735781/" \l "R27) - [29](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5735781/#R29) ]。由于这些特性，众包具有改善全球卫生研究的潜力。确实，由于儿童健康与营养研究计划（CHNRI）的研究优先级确定方法广受欢迎，该方法使用集体意见来确定全球卫生，因此经常使用众包确定研究优先级，尤其是在孕产妇，新生儿和儿童健康中并根据一系列标准对研究重点进行评分[ [30](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5735781/#R30)]。由于其透明，系统的性质，CHNRI方法正成为最常用的研究优先级设置方法。它旨在利用Surowiecki的“人群的智慧”原则，将在论文[ [31](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5735781/#R31) ]中进一步进行描述。此外，在资金，物流，现有卫生保健系统较差，医护人员收集数据，设备以及患者获得医疗保健等方面，全球卫生研究面临比高收入国家更大的负担。或冲突地区[ [21](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5735781/#R21)，[32](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5735781/#R32) - [37](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5735781/#R37)]。随着中低收入国家对手机的使用仍在增加，众包可以利用这些国家已经具备的结构和知识，为传统来源提供补充的数据收集途径。