

**本科生毕业设计[论文]**

**基于OAuth2.0的第三方用户信息系统设计与实现**

专业班级 软件工程1301

姓 名 邢铭哲

学 号 U201317429

指导教师 卢力

2017年 6月 1日

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名： 年 月 日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于 1、保密囗，在 年解密后适用本授权书

2、不保密囗 。

（请在以上相应方框内打“√”）

作者签名： 年 月 日

导师签名： 年 月 日

摘 要

随着移动互联网规模的不断扩大、移动设备的不断普及，网络技术的迅速更新，由用户为主导而生成的内容互联网模式——Web 2.0的发展更是成为互联网的重大变革，推动了人与人之间对于网络互动资源的共享。但随之却产生了一系列问题，比如人们的账号数量不断增加，用户的账号密码很难维护，也会随之出现用户隐私信息泄露问题，并且各个网络应用之间的账号信息以及资源不能进行有效的沟通和共享，从而形成数据资源孤岛，不利于数据价值挖掘和利用。基于OAuth 2.0的用户信息系统便是针对这些问题应运而生的。

该课题主要基于PHP语言和laravel框架，从用户信息系统的需求出发，先进行可行性分析，分别对技术可行性、经济可行性、操作可行性做了较为详细的说明，之后给出了系统的功能性需求和非功能性需求，经过详细设计、OAuth 2.0的工作原理和在工程中的实现介绍、用户信息系统各个功能模块的实现以及测试，最终实现了用户信息系统，并达到了最初的目标。

结合对于最终实现的用户信息系统的单元测试、集成测试、性能测试等结果表明系统不仅功能逻辑正确并完善、使用灵活、维护成本低、扩展性高，并且性能达到了实际生产中的要求，可以投入实际使用中，对于提高用户信息的维护效率和安全性具有重要意义。

**关键词：**OAuth 2.0协议 开放式授权 单点登录 用户中心

Abstract

With the continuous expansion of the size of the mobile Internet, mobile devices continue to spread, the rapid updating of network technology, user-led content generated by the Internet model - Web 2.0 development is to become a major change in the Internet, to promote people Between the sharing of resources for the network. But with the rapid development of the Internet world, followed by a series of problems, such as the increasing number of people account, the user's account password is difficult to maintain, will also appear with the user privacy information disclosure problems, and between the various network applications The account system and resources can not be effectively communicated and shared, so as to form data islands, is not conducive to the value of data mining and utilization. OAuth 2.0 based on the user information system is for this problem came into being.

This topic is mainly based on the PHP language and laravel framework, starting from the demand of user information system, the feasibility of the first analysis, respectively, the technical feasibility, economic feasibility, operational feasibility of a more detailed description, and then given the system Functional requirements and non-functional requirements, after a detailed design, OAuth 2.0 working principle and the realization of the project, the user information system to achieve the various functional modules and testing, and ultimately to achieve the user information system, and reached the initial goal The

Combining the results of unit testing, integration testing, performance test and so on for the final realization of the user information system shows that the system function logic is correct and perfect, flexible use, low maintenance cost, high expansibility and performance to meet the actual production requirements. In use, it is important to improve the maintenance efficiency and safety of user information.

**Key Words：**OAuth 2.0 Protocol Open Authorization Single Sign On User Center

目录

[1 绪论 7](#_Toc484208668)

[1.1 系统的研究背景及意义 7](#_Toc484208669)

[1.2 国内外研究现状 8](#_Toc484208670)

[1.3 本文主要研究工作 10](#_Toc484208671)

[1.4 本论文的组织结构 10](#_Toc484208672)

[2 相关技术介绍 12](#_Toc484208673)

[2.1 PHP语言 12](#_Toc484208676)

[2.2 Laravel框架 13](#_Toc484208677)

[2.3 Apache服务器 14](#_Toc484208678)

[2.4 OAuth 2.0协议 14](#_Toc484208679)

[2.5 三元权限 15](#_Toc484208680)

[2.6 Redis存储 15](#_Toc484208681)

[2.7 RabbitMQ消息队列 15](#_Toc484208682)

[2.8 本章小结 16](#_Toc484208683)

[3 用户信息系统的分析与设计 17](#_Toc484208689)

[3.1 用户信息系统的需求分析 17](#_Toc484208690)

[3.1.1 目标 17](#_Toc484208691)

[3.1.2 可行性分析 18](#_Toc484208692)

[3.1.3 系统的功能需求 19](#_Toc484208693)

[3.1.4 系统的业务需求 20](#_Toc484208694)

[3.1.5 系统的性能需求 21](#_Toc484208695)

[3.1.6 系统的安全需求 22](#_Toc484208696)

[3.1.7 系统的接口需求 22](#_Toc484208697)

[3.2 用户信息系统的总体设计 23](#_Toc484208698)

[3.2.1 系统体系架构 23](#_Toc484208699)

[3.2.2 系统功能模块结构 23](#_Toc484208700)

[3.3 用户信息系统的功能模块设计 24](#_Toc484208701)

[3.3.1 基于OAuth 2.0的身份认证 24](#_Toc484208702)

[3.3.2 组织管理 26](#_Toc484208703)

[3.3.3 项目组管理 28](#_Toc484208704)

[3.3.4 任务管理 30](#_Toc484208705)

[3.3.5 用户管理 32](#_Toc484208706)

[3.3.6 客户端管理 34](#_Toc484208707)

[3.3.7 开放平台管理 35](#_Toc484208708)

[3.4 用户信息系统的数据库设计 37](#_Toc484208709)

[3.4.1 系统E-R建模 38](#_Toc484208710)

[3.4.2 数据库表结构设计 38](#_Toc484208711)

[3.5 用户信息系统的安全设计 42](#_Toc484208712)

[3.5.1 实现HTTPS进行安全传输 42](#_Toc484208713)

[3.5.2 实现对私密数据进行Hash加密 43](#_Toc484208714)

[3.5.3 实现系统的访问控制 43](#_Toc484208715)

[3.5.4 实现系统的双机热备 43](#_Toc484208716)

[3.6 本章小结 43](#_Toc484208717)

[4 用户信息系统的实现与测试 45](#_Toc484208718)

[4.1 系统的软硬件环境 45](#_Toc484208719)

[4.1.1 软件环境 45](#_Toc484208720)

[4.1.2 硬件配置 45](#_Toc484208721)

[4.2 系统功能实现 45](#_Toc484208722)

[4.2.1 登录实现 46](#_Toc484208723)

[4.2.2 基于OAuth 2.0的身份凭证认证实现 47](#_Toc484208724)

[4.2.3 组织管理实现 49](#_Toc484208725)

[4.2.4 项目组管理实现 50](#_Toc484208726)

[4.2.5 任务管理实现 52](#_Toc484208727)

[4.2.6 用户管理实现 54](#_Toc484208728)

[4.2.7 客户端管理实现 55](#_Toc484208729)

[4.2.8 开放平台实现 56](#_Toc484208730)

[4.3 系统测试 57](#_Toc484208731)

[4.3.1 单元测试 57](#_Toc484208732)

[4.3.2 集成测试 58](#_Toc484208733)

[4.3.3 性能测试 59](#_Toc484208734)

[4.4 本章小结 60](#_Toc484208735)

[5 总结与展望 61](#_Toc484208736)

[5.1 全文总结 61](#_Toc484208737)

[5.2 研究展望 62](#_Toc484208738)

[致谢 63](#_Toc484208739)

[参考文献 64](#_Toc484208740)

# 1 绪论

## 系统的研究背景及意义

在传统互联网时代，各个网络应用或者服务之间的资源是封闭的，形成了数据资源孤岛，对于这些海量的数据资源更是得不到有效的共享利用。网络应用与服务之间海量数据的整合与共享已经成为必然趋势，网络信息逐渐从分散的状态进行聚合。而数据共享需要提高系统服务之间的相互协作，对用户的认证和授权也提出了更高的要求。2007年，Facebook最先提出开放平台的概念[1]。开放平台提供开放的API（Open API），开发者和第三方应用可以利用简洁易懂的Open API获取系统的数据。开放平台的产生代表了互联网数据共享的发展，国内各个IT公司逐渐开发了自己的开放平台，如新浪微博开放平台，人人网开放平台，腾讯开放平台，百度开放平台等。但随之而来的问题是如何对这些开放API进行认证和授权，如今，互联网上主要有两种认证授权协议：OpenID和OAuth[2]。本文主要介绍基于OAuth 2.0的用户信息管理系统的设计实现，以下简称用户信息系统。

本课题来源于中国科学院软件研究所操作系统基础版研发支撑平台（ISDE），随着支撑平台的开发日益完善，其子系统的数量和规模也逐渐扩大，支撑平台集成了如Gitlab，JIRA，mantis，QDS，TestLink，LogCat等经过二次开发的开源应用以及针对项目开发周期中开发的特定应用，我们在日常开发工作中建立了项目组、用户、任务的基本概念，我们需要将这三个对象进行对应管理，当数据资源发生变化时需要与各个子系统进行消息推送，使每个系统做到数据一致性，并且每个子系统都有自己的账号，导致账号数量繁多，不易进行管理、维护、统一，我们也需要将各个子系统的用户信息统一收敛到一个第三方的用户信息管理系统，进行统一的管理维护，因此如何将这几个子系统整合到一起，并进行数据资源共享就成为了一个问题。针对以上问题，我们设计开发了基于OAuth 2.0的第三方用户信息管理系统，此系统即为本文进行讨论的用户信息系统。此用户信息系统需要维持用户在全局子系统中的在线状态并保持数据一致性以及对项目组、用户、任务等信息进行有效管理，此外，用户信息系统也要提供开放平台，Open API以供第三方应用、社区等的注册接入。意义是通过实现、完善这些功能，能够提升网络账号的管理能力、提高安全稳定性，并进一步更好地支撑操作系统基础版研发支撑平台（ISDE）各子系统。

## 国内外研究现状

如今，网络应用主要是由B/S架构和C/S架构组成。B/S（Browser/Server）结构也就是浏览器和服务器结构，它是对C/S结构的变化和改进，B/S结构少部分逻辑处理在浏览器端进行处理，大部分逻辑任务在服务器端处理，服务器如IIS、Apache、Nginx等处理网络请求，维护管理整个Web应用[22]。C/S（Client/Server）架构也就是客户端和服务器架构，主要是应用在桌面应用上，服务器是网络的核心，而客户端是网络的基础，客户端依靠服务器获得所需要的网络资源，而服务器为客户端提供网络必须的资源，将任务合理的分配到客户端和服务端，可以降低系统的通讯开销，但是由于移动客户端计算性能的局限性，复杂的逻辑计算逐渐转移到服务器端进行处理，这时客户端就像B/S架构中的浏览器端一样只负责数据的展示以及少部分逻辑事务处理，C/S架构和B/S架构应用之间的界限逐渐缩小，因此我们可以将这种C/S架构应用和B/S架构应用统称为网络应用。随着Web、网络技术的迅速发展以及以社交分享为代表的Web 2.0时代的到来，人们越来越注重人与人之间在互联网上的交互。从较早的博客、论坛、BBS、等，到如今社会性网络服务（Social Network Site）应用、电子商务等日益普及，与此同时，随着移动互联网的速度、稳定性和普及度不断提高，致使一批批的移动互联网应用逐渐兴起，如现在较为流行的照片实时分享应用Instagram，根据统计结果，Instagram在2010年10月13日时的用户数量仅仅不到10万，而到了2010年12月21日用户数就突破了100万。2012年，Instagram的用户数就达到了8000多万，距离该网站的用户数超过5000万仅仅2个多月。这意味着，每月Instagram的用户数增长数超过1000万。Instagram同时公布统计结果，其应用的用户已分享了近40亿张照片[3]。那么在拥有规模如此巨大的网络应用以及网络应用用户的今天，用户信息的保护与管理就成了一个很严峻的问题，很显然，用户账号数量越少，其数据隐私，账号信息泄露的几率就越少，针对这个问题，先后出现了OpenID和OAuth两种协议来进行跨应用授权和资源共享，用户不必记忆繁杂的账号密码，并且可以跨应用分享自己的数据。

2007年，OpenID正式发布，OpenID是一个去中心化的网上身份认证系统[4]。对于以往的需要用户进行身份凭证校验的网站，用户需要记住像用户名和密码这样的传统验证标记来进行登录操作。如果网站支持OpenID，则用户只需要在一个作为OpenID身份提供者（identity provider, IdP）的网站上注册，就可以在支持IdP的网站上直接授权登录[23]。OpenID正在被越来越多的大网站采用，比如作为身份提供者的雅虎和微信、MySpace等。

2009年4月23日，OAuth宣告了1.0协议上的一个的安全漏洞。该漏洞对OAuth的认证流程产生了影响。于是，针对这一漏洞发布了OAuth 1.0a版本来。后来，一位博士生发现了隐蔽重定向漏洞。该漏洞首先由CNET报道，搜狐网、网易、新浪微博、阿里巴巴、腾讯QQ、人人网、亚马逊、微软Live、开心网、Facebook、Google、、GitHub Yahoo、WordPress、eBay、PayPal、LinkedIn等大量知名网站受到漏洞影响并造成损失。违法分子可以伪造钓鱼网站，显示错误的链接误导用户访问并登录钓鱼网站，然后不法分子即可获取认证服务器返回的Access Token，并读取用户的私人信息。该漏洞被网易，人民网，CSDN，搜狐，凤凰网，太平洋电脑网等大量中文网站报道[21]。OAuth2.0是OAuth协议的下一版本，但不向下兼容OAuth 1.0，即完全废止了OAuth1.0。OAuth 2.0关注客户端开发者的简易性。2012年10月，OAuth 2.0协议正式发布为RFC 6749。

在如今的Web领域中，OAuth授权协议在国内外的实施已经成熟并且得到广泛运用。国内外知名Web服务商，IT公司都提供了开放平台，提供了Open API，用户可以十分便捷的接入第三方授权认证，获取第三方数据资源并进行共享，OAuth协议对用户，消费方，资源提供方也都带来了好处：用户不必再记忆数量繁多并且复杂的账号密码，隐私数据的安全性有了保障，并且可以跨应用分享自己的数据，使用户可以更加融入网络社交中；对于消费方，大大减少了对用户账号数据的维护工作量，并且可以通过接入到其他应用与其他应用交互提高用户的兴趣，同时减少繁琐的注册工作使用户的体验更加友好；对于资源提供方，可以加强其在社会化网络世界中的主要地位，增加用户流量。OAuth协议的普及使网络应用从传统的封闭式应用变成资源共享、互联的新型互联网网络格局，极大地促进了社会化网络以及Web2.0时代的发展[24]。

OAuth协议从1.0到如今2.0，中间遇到了很多问题和漏洞，OAuth协议也一直不断完善规范中，总体来说，OAuth协议已经取得了显著地效果和广泛的应用，并且在未来拥有广大的应用前景。

## 本文主要研究工作

本毕业论文题目为《基于OAuth 2.0的第三方用户信息系统的设计与实现》，因此，本文主要围绕着两个重点来讨论：一个是OAuth 2.0网络认证框架的研发；一个是用户中心的设计与实现，包括公司、项目组、用户、任务等几个基本概念的相互之间关联的实现。

其中，OAuth 2.0网络认证框架的研究主要包括OAuth 2.0协议的具体工作流程的研究以及其具体实现，并且实现开放平台，提供Open API，并提供第三方应用注册客户端，接入网络账号并提供不同的权限管理，满足开放社区用户注册和登录需求；用户中心的研发主要包括实现公司，项目组，用户，任务之间关联关系，三元审核，ISDE工作平台内部所有子业务系统账号系统的梳理，当用户中心的数据发生改变时能够实时向第三方应用推送通知，从而保持数据一致性，统一规范设计和研发用户信息数据库，用户信息的分组管理，用户的权限管理，用户状态管理模块用户认证。该项目要求实现高并发、大负载架构设计，最后能够基本稳定地为ISDE各子系统提供账号登录、权限管理、用户在线状态保持等服务。 能够支持并发1000人/秒的在线请求服务。

## 本论文的组织结构

全文共分为五个章节：

第一章为绪论，主要介绍本课题的来源、意义以及研究背景，关于OAuth 2.0、开放平台的国内外研究现状，以及本论文的主要研究工作。

第二章为相关技术介绍，介绍了项目开发中所使用的框架、技术、设计方法，为接下来的设计与实现最好基础。

第三章为需求分析和设计，主要从系统的功能、业务、性能、安全、接口方面做出详细需求分析，然后使用系统体系架构图和系统功能模块结构图介绍了系统的体系架构和组成结构，并针对系统的功能模块设计、数据库设计、安全设计做出介绍。

第四章为系统的实现与测试，介绍了系统具体的功能实现，并且经过单元测试、集成测试、性能测试之后验证系统满足需求。

第五章为总结与展望，主要对本文所做的工作做出总结，并分析出不足之处，并介绍了未来项目需要完善和补充的地方。

# 2 相关技术介绍



## PHP语言

PHP(Hypertext Preprocessor)语言是一门年轻的创建Web站点的服务器端脚本语言，拥有完整的面向对象编程特点，在当前的Web开发领域得到了广泛的运用[5]。根据w3tech.com上2017年统计的数据PHP写的网站已经拥有超过80%的市场份额，统计结果如图2-1所示。

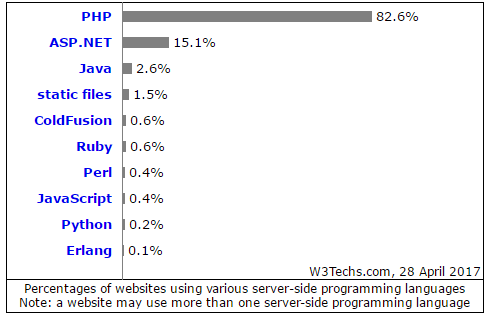


图2-1 W3Tech上对于PHP的统计数据

PHP语言有非常多的优点：对于有类似C语言代码基础的人简单易学，上手较快，其弱类型的变量定义方式使代码简洁优雅，在项目管理，团队组建方面有其特有的优势。PHP拥有庞大的社区，大量的开源项目，合适的类库，功能较为丰富，性能足够好，可以进行热部署，项目部署比较简单，易修改，PHP具有较好的兼容性，，可以再不同的平台上运行（Windows，\*Unix，Mac OS），PHP与目前Web应用所使用的服务器（IIS，Apache，Nginx）所兼容，可以高效的运行于服务器端，并且PHP提供了广泛的数据库支持。

针对PHP的以上优点，本项目采用PHP作为主要开发语言。

## Laravel框架

Laravel是一款基于PHP语言的优雅的Web框架，其借鉴了许多框架以及设计技巧，比如Ruby on Rails、ASP.NET MVC等。它集成了许多有用的功能，这些功能是经常在Web应用中需要实现的，而Laravel集成了这些功能模块，增加了开发的便利，这些模块包括安全验证、路由、数据迁移、Session、缓存、邮件、Eloquent ORM等功能[25]。在日常开发中，身份凭证验证几乎是必不可少的，而laravel框架已经继承了身份验证的模块，只需要Auth命令即可实现包括登录验证、获取登录用户信息、密码加密等功能。laravel提供了非常简洁的路由方式，所有的路由集中在route文件中，查看项目的路由可以非常方便简洁。数据库迁移在团队协作中可以非常方便的保持数据的统一。Eloquent ORM对象关系映射框架简化了数据库操作，并避免了SQL注入等可能存在的数据库漏洞[26]。这些Laravel框架的特性大大提高了程序员的开发效率。对于PHP5.3以上的版本，可以使用Composer作为依赖管理工具，Composer借鉴了Maven、npm的设计理念，Composer安装依赖到项目的Vender目录，而不是全局，因此它仅仅是依赖管理工具。Composer可以通过命令行require安装，也可以在项目根目录下的composer.json文件里声明依赖，然后通过install命令下载或者update命令更新。Composer的出现方便了PHP开发者进行依赖管理，极大地提高了PHP的开发效率。PHP各个框架的市场占有情况如图2-2所示。

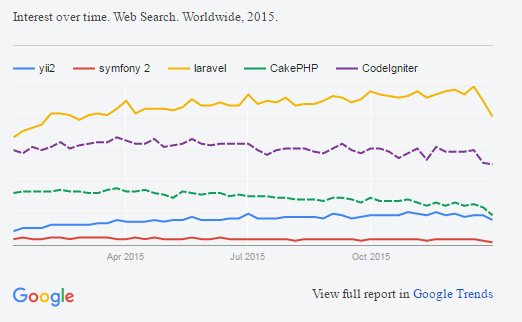


图2-2 PHP框架市场占有情况

## Apache服务器

Apache http server是一款开源的Web服务器，以其安全性、稳定性和跨平台性而被广泛的应用，是当前市场上最流行的服务器端软件之一[13]。它支持多种服务器端语言，比如PHP，Python，Java，Perl等。Apache在当前的Web服务器领域的应用已经十分成熟，并且有着十分庞大的社区资源，不断有开发者为其开发新的特性，新的功能，修复缺陷。Apache的优点是速度快、性能稳定，可以作为代理服务器使用、配置简单（比如对PHP的支持配置）。当下流行服务器市场占有情况如图2-3所示。

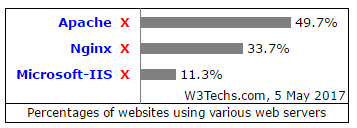


图2-3 流行服务器市场占有情况

## OAuth 2.0协议

OAuth 是一个开放性的网络授权协议，可以让用户通过用户授权的第三方应用访问该用户在提供第三方认证的网站上存储的私密资源（如博客、图片、音乐等），而不用给第三方应用提供用户名和密码[9]。OAuth 2.0协议是国际上关于开放性网络授权的第二代标准，目前，在世界各大互联网公司和运营商中得到了广泛的应用[14]。

下面主要简单介绍OAuth 2.0的应用场景和协议流程。随着社会化网络的发展，网络应用的数量和规模呈爆发性增长，数据量也不断增加，为了避免形成数据资源孤岛，我们需要进行资源共享，在双方信任的基础上，第三方用户可以在服务提供方的认证服务器上经过认证，然后可以在规定的权限内获取资源服务器中的数据。当前，OAuth 2.0主要有四种模式——授权码模式、简化模式、密码模式、客户端模式。其中授权码模式是最完整，也是流程最严密的授权方式。本系统中主要使用授权码模式进行认证。OAuth 2.0的抽象协议流图如图2-4所示。

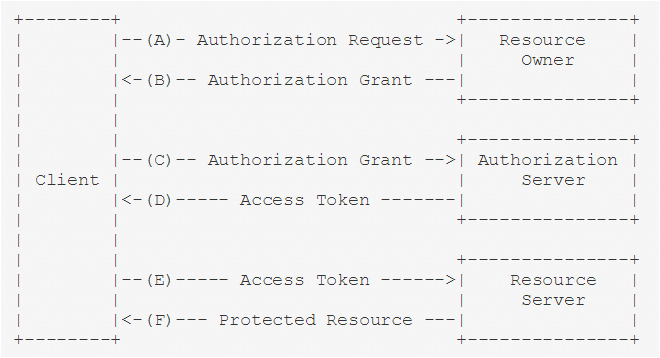


图2-4 OAuth 2.0抽象协议流图

## 三元权限

对于权限的管理实际上是对于不同资源执行不同操作。在权限管理中，我们抽象出Who（主体）、What(资源)、How（操作）三个对象，这样就构成了权限三元组[6]。在用户信息系统中，权限的管理至关重要，对于管理员、项目组负责人、成员等角色的权限分析都是在三元权限的基础上进行的。

## Redis存储

Redis是一个高性能的基于key-value的数据库Redis 本身支持 master-slave（主从）同步,提高了Redis的可靠性，并且可以通过读写分记提高性能[7]。并且设计简单，可以对数据库中数据进行缓存，降低数据库负载，并对Web页面进行缓存，提高响应速度，降低网站负载[8]。

## RabbitMQ消息队列

RabbitMQ是应用于应用之间进行实时消息传递的工具。RabbitMQ中主要有交换机（Exchange）、消息队列（Queue）、通道（Channel）的概念[15]。生产者发出消息，经过交换机储存在消息队列中，消费者从消息队列中依次获取数据。RabbitMQ工作示意图如图2-5所示。

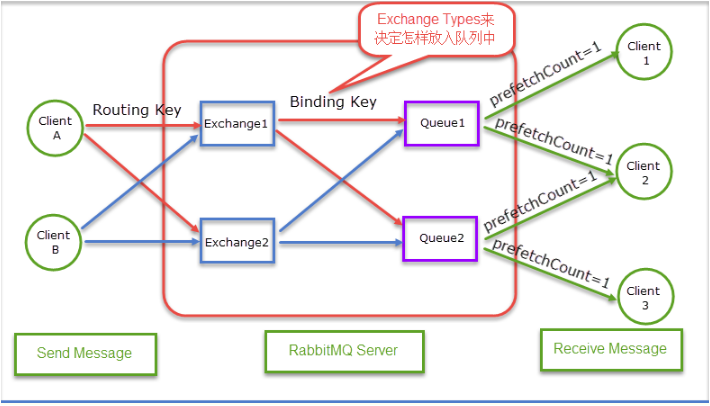


图2-5 RabbitMQ工作示意图

在用户信息系统中，需要把数据变动实时推送到其他系统中，以保持数据一致性。

## 2.8 本章小结

本章主要介绍了系统设计开发中使用到的技术以及设计思想，包括PHP语言、laravel框架、OAuth 2.0协议流程、三元权限、RabbitMQ消息队列等，为接下来的讨论提供了技术基础。



# 用户信息系统的分析与设计

## 用户信息系统的需求分析

### 目标

本课题来自中国科学院软件研究所操作系统基础版研发支撑平台（ISDE）。在描述本用户信息系统的开发目标之前先引入应用集成的概念。随着公司企业的不断发展，公司使用的应用软件会不断增加，这些软件大多是来自世界各地的开源贡献者贡献的开源应用或者系统以及公司内部针对解决特定业务而开发的第三方应用。这些第三方应用大多都提供了开放平台和Open API对外提供服务。通过对这些应用的集成可以节省开发成本，满足新的业务需求。狭义来讲，本用户信息系统是针对当前实现的应用集成平台（Application Integration）来进行开发的，但随着之后应用规模、数量的不断扩大，用户信息系统也会应用于软件所的社区，数据资源管理等方面，本文主要讲述针对于应用集成平台（以下简称AI）的用户信息系统的开发。

本网络账号开发的目的主要有以下几点：

（1）基于OAuth 2.0授权协议的单点登录，支持应用集成平台AI，包括Gitlab、JIRA、mantis、TestLink等系统的登录认证。并且我们要实现针对于用户信息系统和AI平台的工作流系统。工作流系统的作用是“使在多个参与者之间按照某种预定义的规则传递文档、信息或任务的过程自动进行，从而实现某个预期的业务目标，或者促使此目标的实现”[9]。在进行服务集成的时候工作流系统可将业务流程的任务关联到第三方工具的具体服务，通过业务流程的执行满足业务需求。因此我们需要实现用户信息系统与AI应用集成平台之间的预定义的文档、消息、任务等的自动传递。

（2）实现开放平台，开放简洁易用的Open API，向外提供丰富的接口以提供大量的信息，并提供接口测试工具使接入用户更加直接快捷的了解学习Open API和消息队列的调用和数据参数传递。

（3）用户信息系统作为用户中心平台，还需要管理公司内部所有人员的基本信息，群组关系以及任务分配。因此需要处理用户、群组、任务之间的相对关系，并且对三者的数据进行管理。

### 可行性分析

可行性分析的目的是为了对问题进行研究，以最小的代价和最短的时间确定问题是否可行，以及分析在经济、技术、运行组织上可能存在问题。

#### 经济可行性

经济可行性分析是从经济角度出发，估算系统的成本效益，包括系统运行、维护成本，对社会经济的效益。

用户信息系统将各个独立的子系统的账号统一收敛到一起，避免了传统的账号系统存在的账号繁杂、难以记忆维护的情况；并对公司的成员、项目组、任务关系进行统一管理，大大的节约了时间、人力成本，并且规范了业务操作程序，避免出现差错，提高了安全性和可靠性，从而可以提高公司以及项目组的信息化和现代化管理水平。总体来说，系统收益远大于成本。

#### 技术可行性

技术可行性是根据系统的目标来衡量是否具备所需要的技术[10]。

本系统在开发过程中使用Ubuntu 16.04作为操作系统，PhpStrom作为集成开发环境，使用PHP和Laravel作为开发语言和Web框架，使用MySQL作为数据库系统。现有技术完全可以承担开发任务。

#### 操作可行性

本系统是基于B/S架构的系统，所以操作不受操作系统的影响，易于使用，不用进行专门的培训，管理员用户需要对公司内部的组织结构，任务分配有清晰的概念，并能进行有效管理。

### 系统的功能需求

功能需求规定了必须在产品中实现的软件功能，用户利用这些功能来完成任务，满足业务需求[11]。经过与软件所总体部领导以及老师进行讨论，结合他们的工作流程以及功能细节，确定了系统所需具备的基本功能以及对于不同身份的用户应该具有怎样的功能，以及确定功能之间的关系。

进过详细的分析与规划，我们最终确定用户信息系统应具有以下几点功能：

（1）OAuth 2.0授权流程实现以及多系统之间的单点登录。实现OAuth 2.0协议中作为功能最完整、流程最严密的授权码模式（authorization code）在客户端、授权服务器、资源服务器之间的流程，客户端使用access\_token实现单点登录。

（2）开放平台提供Open API，并且允许第三方应用注册接入。提供开放平台，Open API列表，管理员可以添加、删除、修改接口组、接口、测试API并查看消息结果及格式；用户可以查看接口组列表、接口列表、测试API并查看消息结果及格式。提供客户端注册，第三方应用需要提供重定向的URL，并可以选择客户端权限。

（3）用户中心的设计与实现。用户中心管理主要包括对用户、项目组、任务之间的管理，并且当用户、项目组、任务等信息发生变化时，需要对第三方应用即客户端进行信息推送，客户端进行对应数据操作从而保持数据一致性。其中因为用户有管理员和普通用户两种角色，因此用户信息系统用户中心的功能主要围绕着管理员用户和普通用户两种角色分开讨论。管理员用户主要有系统管理、注册新用户、组织管理、任务管理、人员管理、项目组管理、客户端注册管理、开放平台信息管理等功能。上述功能均具有增加、删除、修改、查询等基本操作。普通用户须具有客户端注册、开放平台查看接口列表和接口组列表的基本功能，并且普通用户可以申请管理员，经过管理员审核通过后角色转为管理员用户。

### 系统的业务需求

#### 组织结构介绍

本用户信息系统自上到下分别有公司（此系统中只有一个公司，即软件所总体部）、项目组、成员。用户类型主要分为学生、职工、实习生、责任QA，除此之外还有系统管理员角色。组内成员角色主要分为负责人、QA、普通成员。

#### 角色需求分析

作为用户信息系统，本系统不同角色主要需要实现的业务流程有：管理员管理用户、权限、用户类型、组内角色、审核管理员申请、注册用户、组织管理、任务管理、项目组管理、客户端管理、开放平台管理等业务；项目负责人主要负责项目组人员管理、任务分配等业务；普通成员包括学生、实习生、职工等主要负责解决人物、实际开发。

#### 系统用例图

系统从功能角度出发有管理员、项目负责人和普通成员角色，本用户信息系统的用例图如图3-1所示，展示了系统的各个功能模块模块的具体需求，也更加形象的展示了各个功能模块之间的联系。



图3-1 用户信息系统用例图

### 系统的性能需求

对于本用户信息系统，主要有以下几点性能需求：

（1）数据库的数据加载速度必须远大于新数据的生成速度，不能出现数据库数据加载、更新不及时的现象；

（2）数据库应支持应用系统或者公司、项目组、用户、任务等数据的同步或异步更新；

（3）支持用户并发业务的处理时间<1s；

（4）能够支持并发1000人/秒的在线请求服务；

（5）系统需要满足7×24小时的不间断运行，运行一段时间后，整体执行效率不衰减。

（6）系统网站首页打开速度3s以下，账号登录速度 8s以下。

（7）系统能在高于实际系统运行压力1倍的情况下，稳定的运行24小时。

### 系统的安全需求

本用户信息系统涉及到用户的网络账号以及项目组信息、任务信息、用户信息 私密数据，本系统为Web应用，访问用户为互联网上的所有用户，这样大量的访问增加了危险性，因此，本系统的安全需求非常重要。本系统的安全需求分析主要得出以下几点需求：

（1）对系统敏感数据（如用户密码、数据库连接信息等配置信息）需进行加密处理。

（2）实现系统的保密性，为不同的用户提供不同的访问权限，只有经过授权的用户才能读写用户信息系统的信息，而且必须防止信息的非法、非授权的泄漏，防止对系统信息的越权存取。

（3）服务器需要能进行网络周边防御，以最大限度降低外界攻击的危险，并能及时检测外界的非法攻击和访问。

（4）本系统针对软件所总体部内部员工使用，因此需要进行IP地址访问控制，要求只能从指定的IP、IP范围、DNS来访问本系统。

（5）保证网络传输过程中的安全，避免第三方非法截取或者伪造数据。

### 系统的接口需求

本用户信息系统提供了开放平台，开放Open API供第三方接入并返回对应数据，此系统提供标准的Json数据格式与数据库对接，对第三方接入的应用的数据请求进行规范处理，以便于数据库的数据操作处理。对于一个有好的开放平台来说，从业务角度来说，Open API应该能够对公司组织、项目组、成员进行统一管理，维护公司、项目组、成员的职能和公共秩序，并能进行数据采集和数据挖掘分析；从技术上角度来说，我们需要实现RESTful风格的API，RESTful风格的API具有轻量，充分利用HTTP本身的语义，不需要其他的消息协议，并且具有透明性，面向资源，一目了然[27]，无状态，不用考虑上下文，不考虑当前的状态，极大地降低了复杂度的优点[28]，而且我们需要逐步完善统一Open API的数据标准，这些标准不仅能应用与本系统，也能够用于第三方应用以及本系统未来的扩展，这样可以促进系统内部和系统之间数据的标准化和一致性，有利于数据资源的整合、跨应用的数据资源共享和数据分析利用，有利于系统未来的升级和维护。因此我们对于这些开放平台提供的Open API也提出了以下接口需求：

（1）接口要简洁明了，使第三方可以快捷的接入，并迅速学会使用；

（2）保证接口数据的安全性，第三方必须提供access\_token以及相应的参数信息才能获取数据，并且要避免非法SQL注入或其他攻击来截取数据；

（3）保证接口的性能，要求达到1000人/秒的压力。

## 用户信息系统的总体设计

### 系统体系架构

用户信息系统的总体架构设计如图3-2所示。



图3-2 用户信息系统总体架构设计图

### 系统功能模块结构

用户信息系统的功能模块结构图如图3-3所示。



图3-3 用户信息系统功能模块结构图

## 用户信息系统的功能模块设计

### 基于OAuth 2.0的身份认证

本用户信息系统最基础的功能模块支撑就是基于OAuth 2.0协议的身份凭证认证框架，身份凭证认证框架提供了基础的账号密码认证，包括获取access\_token的OAuth 2.0协议流程，根据access\_token授权获取数据等基本功能。

其功能结构图如图3-4所示。



图3-4 基于OAuth 2.0的认证模块功能结构图

基于OAuth 2.0的身份凭证认证模块的抽象协议流图如图3-5所示。



图3-5 OAuth 2.0抽象协议流图

其中，我们主要是用授权码模式来实现OAuth 2.0的授权流程，其抽象协议流图如图3-6所示。



图3-6 授权码模式的抽象协议流图

每一步的消息传递如下：

（A）当用户访问客户端时，客户端会将用户转向认证服务器的授权登录页面。

（B）在授权登录页面，用户可以登录以及选择是否给客户端授权。

（C）如果用户给客户端授权，认证服务器将用户重定向至客户端事先规定的附有授权码（Authorization Code）的URL（Redirection URL）。

（D）客户端请求附上授权码的重定向URL，向认证服务器申请Access Token。客户端在后台的服务器上完成这一步骤，对用户是不可见的。

（E）认证服务器在后台校验授权码和重定向URI的对应关系，确认一致后，通过回调URL向客户端发送授权令牌（Access Token）和更新令牌（Refresh Token）。

### 组织管理

在用户信息系统的用户中心部分，组织为最上层的结构，目前，只有中国科学院软件研究所总体部一个组织，所有项目组、成员、任务都是在总体部下面所有。其功能结构图如图3-7所示。



图3-7 组织管理模块功能结构图

组织管理功能只有管理员才能进行管理，下面给出创建组织的流程图，如图3-8所示，修改组织信息、删除组织、查询组织流程图类似。



图3-8 创建组织流程图

### 项目组管理

管理员在最高层组织下可以新建项目组，并且可以无穷递归下去，位于树的叶子节点的项目组下是成员，项目组管理的功能模块结构图如图3-9所示。



图3-9 项目组管理模块功能结构图

在项目管理功能模块中分为管理员和非管理员角色，管理员可以在组织这个最高层结构下建立项目组，也可以在项目组中建立子项目组，对于子项目组的管理模块和其上层的项目组管理模块结构功能都是一样的，并且可以修改项目组信息，进行成员管理，在成员管理子功能模块中，可以对子项目组添加成员，修改成员角色，删除成员，也可以删除项目组和查询项目组，对于非管理员角色，只能查询项目组列表，以及项目组信息。下面给出创建项目组的流程图，如图3-10所示，子项目组管理、修改组信息、成员管理、删除项目组、查询项目组、添加成员、修改成员角色、删除成员的流程图类似。



图3-10 新建项目组功能流程图

### 任务管理

在任务管理模块，管理员用户可以创建任务，修改任务信息，删除任务，组织管理可以将任务分配到组织，项目组管理可以将任务分配到指定组织下的项目组，成员管理可以将任务分配到指定项目组下的成员。其功能结构图如图3-11所示。



图3-11 任务管理模块功能结构图

添加任务功能的流程图如图3-12所示，修改任务信息、组织管理、项目组管理、成员管理、任务删除、任务查询的流程图类似。



图3-12 添加任务流程图

### 用户管理

管理员用户可以对组织下的用户进行管理，主要操作有查找用户、添加用户、删除用户。普通用户可以在通讯录里面查询用户。其功能结构图如图3-13所示。



图3-13 用户管理模块功能结构图

注册用户流程图如图3-14所示，删除用户、查询用户的流程图类似。



图3-14 注册用户流程图

### 客户端管理

第三方应用可以注册客户端，获取Client ID和Client Secret，并且注册回调函数以便第三方应用可以接入用户信息系统，并且可以选择第三方应用访问资源服务器所拥有的权限。其功能结构图如图3-15所示。



图3-15 客户端管理模块功能结构图

注册客户端流程图如图3-16所示，删除客户端的流程图类似。



图3-16 注册客户端流程图

### 开放平台管理

用户信息系统提供开放平台，在开放平台里，会提供Open API以供第三方接入者查阅使用，其功能结构图如图3-17所示。



图3-17 开放平台管理模块功能结构图

对于管理员用户，可以添加接口组、添加接口、修改接口、删除接口，普通用户和管理员用户可以测试接口，测试接口请求参数和返回数据格式，添加API的流程图如图3-18所示，添加接口组、修改接口、删除接口、测试接口的流程图类似。



图3-18 添加API流程图

## 用户信息系统的数据库设计

本用户信息系统采用MySQL数据库。数据库是为用户信息系统提供分类组织、存储、管理各种数据的的工具，负责信息的管理、维护，因此，想要设计好一个系统，就必须要建立合理的数据库模型，合理的设计数据库才能使系统稳定工作，保证数据的一致性和完整性，提高系统数据查询的处理速度和性能，并且优化存储效率。本系统的各个功能模块的数据以数据表的形式存储在数据库中，因此，想要设计合理的数据库就要处理好数据表以及数据表之间关系的设计。

### 系统E-R建模

E-R图是用来描述概念模型的工具，通过抽象数据并设计局部视图，集成局部视图从而得到全局视图的方法来描述数据库的逻辑结构[12]。图3-19为系统的E-R图。



图3-19 系统E-R图

### 数据库表结构设计

下面列出主要数据表的字段信息。表3-1是用户表，存储用户基本信息。

表3-1 用户表（user）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 主键 | 为空 | 描述 |
| id | int | 10 | 是 |  | 自增键 |
| group\_id | int | 10 |  |  | 项目组ID |
| name | varchar | 32 |  |  | 姓名 |
| username | varchar | 32 |  |  | 用户名 |
| user\_category | varchar | 32 |  |  | 用户类别 |
| email | varchar | 128 |  |  | 邮件 |
| password | varchar | 64 |  |  | 密码 |
| remember\_token | varchar | 100 |  | 是 | 记住我 |
| sex | varchar | 11 |  | 是 | 性别 |
| address | varchar | 255 |  | 是 | 住址 |
| phone | varchar | 100 |  | 是 | 电话 |
| portrait | varchar | 255 |  | 是 | 照片 |
| created\_at | timestamp | 0 |  |  | 创建时间 |
| updated\_at | timestamp | 0 |  |  | 更新时间 |

表3-2是项目组表，存储项目组基本信息。

表3-2 项目组表（project\_group）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 主键 | 为空 | 描述 |
| id | int | 10 | 是 |  | 自增键 |
| name | varchar | 128 |  |  | 项目组名字 |
| enname | varchar | 32 |  |  | 项目组拼音 |
| englishname | varchar | 100 |  |  | 项目组英文名 |
| company\_id | int | 10 |  |  | 项目组ID |
| created\_at | timestamp | 0 |  |  | 创建时间 |
| updated\_at | timestamp | 0 |  |  | 更新时间 |
| super\_id | int | 10 |  | 是 | 父项目组ID |
| leaf | int | 10 |  |  | 是否为叶子项目组 |

表3-3是用户-项目组关联表。

表3-3 用户-项目组关联表（project\_group\_member）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 主键 | 为空 | 描述 |
| id | int | 10 | 是 |  | 自增键 |
| user\_id | int | 10 |  |  | 用户ID |
| project\_group\_id | int | 10 |  |  | 项目组ID |
| type | int | 1 |  |  | 用户类型 |
| created\_at | timestamp | 0 |  |  | 创建时间 |
| updated\_at | timestamp | 0 |  |  | 更新时间 |

表3-4是任务表，存储任务基本信息。

表3-4 任务表（task）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 主键 | 为空 | 描述 |
| id | int | 11 | 是 |  | 任务ID |
| code | varchar | 128 |  |  | 任务代码 |
| name | varchar | 128 |  |  | 任务名称 |
| enname | varchar | 128 |  |  | 任务英文名 |
| keyword | varchar | 128 |  |  | 关键词 |
| type | int | 10 |  |  | 任务类型 |
| description | varchar | 225 |  |  | 任务描述 |
| created\_at | timestamp | 0 |  |  | 创建时间 |
| updated\_at | timestamp | 0 |  |  | 更新时间 |

表3-5是任务-项目组关联表。

表3-5 任务-项目组关联表（task\_project）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 主键 | 为空 | 描述 |
| id | int | 10 | 是 |  | 自增键 |
| task\_id | int | 10 |  |  | 任务ID |
| project\_id | int | 10 |  |  | 项目组ID |
| created\_at | timestamp | 0 |  |  | 创建时间 |
| updated\_at | timestamp | 0 |  |  | 更新时间 |

表3-6是任务-成员关联表。

表3-6 任务-成员关联表（task\_member）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 主键 | 为空 | 描述 |
| id | int | 10 | 是 |  | 自增键 |
| task\_id | int | 10 |  |  | 任务ID |
| user\_id | int | 10 |  |  | 用户ID |
| task\_role | int | 10 |  |  | 任务角色 |
| created\_at | timestamp | 0 |  |  | 创建时间 |
| updated\_at | timestamp | 0 |  |  | 更新时间 |

表3-7是API信息表，存储API信息。

表3-7 API信息表（api\_info）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 主键 | 为空 | 描述 |
| id | int | 11 | 是 |  | 自增键 |
| group\_id | int | 11 |  |  | API组ID |
| name | varchar | 128 |  |  | API名字 |
| url | varchar | 128 |  |  | API的URL |
| method | varchar | 10 |  |  | 请求方法 |
| params | varchar | 256 |  |  | 请求参数 |
| success\_response | varchar | 512 |  | 是 | 成功响应 |
| fail\_response | varchar | 256 |  | 是 | 失败响应 |
| created\_at | timestamp | 0 |  |  | 创建时间 |
| updated\_at | timestamp | 0 |  |  | 更新时间 |

表3-8是API组表，存储API组的信息。

表3-8 API组表（api\_group）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 主键 | 为空 | 描述 |
| id | int | 11 | 是 |  | 自增键 |
| name | varchar | 128 |  |  | API组名称 |
| description | varchar | 512 |  |  | API组描述 |
| created\_at | timestamp | 0 |  |  | 创建时间 |
| updated\_at | timestamp | 0 |  |  | 更新时间 |

## 用户信息系统的安全设计

### 实现HTTPS进行安全传输

我们用Apache作为本项目的服务器，需要在Apache环境下实现HTTPS。HTTPS全称为超文本传输安全协议，通过提供了正在通信或者相关联网络上的安全身份验证，确保用户和站点之间的数据不能被获取，或者被第三方伪造，从而来保护隐私敏感数据[16]。而在项目中想要使用HTTPS是通过安装SSL证书来实现的。SSL全称为安全套接字层，是一种加密浏览器与服务器之间信息通信的标准安全技术，可以确保服务器和浏览器之间信息传递的安全性[17]。我们需要在受信任的证书颁发机构（CA）处购买证书，然后将证书安装到Apache服务器，即可使用https来访问服务器端内容。

### 实现对私密数据进行Hash加密

在用户信息系统中，我们使用Hash加密算法对账号的密码等隐私数据进行加密处理。Hash加密是将需要加密的文本加密成具有不同长度的，可逆的杂凑字符串[18]。在PHP 5.5之后PHP有一个新特性：Password Hashing API，我们只用password\_hash()就可以对目标字符串进行加密，也可以使用password\_verify()来验证已经经过加密的密文，判断其hash字符串是否一致。

### 实现系统的访问控制

为保证用户信息系统中敏感数据的安全，要限制外部请求的来源。设置硬件防火墙，局域网端口访问限制，IP地址访问限制，来限制请求访问只能来源于受信任的用户，从而确保服务器端隐私数据的安全性。

### 实现系统的双机热备

我们的系统中实现了双机热备。双机热备是通过使用冗余的硬件和双机热备软件来提高系统的可靠性，双机热备软件会检测服务器的状态，当正在使用的服务器发生故障时，双机热备软件会尝试修复主服务器并启动备用服务器，可以保障服务7\*24小时运行[19]。在用户信息系统中主要是用镜像双机热备份方案，镜像双机热备份方案是保证主服务器和备用服务器上的数据一致，主服务器向外提供服务，并且随着业务的进行，会将数据实时备份到备用服务器上，这样，当主服务器发生故障后，可以马上切换到备用服务器提供服务。这样可以提高系统可靠性和数据安全性[20]。

## 本章小结

本章主要对用户信息系统进行了详细的需求分析，包括定义了系统的可行性、功能性需求和非功能性需求，以图形的方式介绍了系统的体系架构设计和功能模块结构，并且详细介绍了系统的功能模块设计、数据库设计和安全设计。

# 用户信息系统的实现与测试

## 系统的软硬件环境

### 软件环境

（1）服务器端操作系统：Ubuntu 14.04 64bit

（2）开发工具：PHPStrom

（3）Web服务器：Apache

（4）DBMS：MySQL

### 硬件配置

（1）CPU：Intel core i7，4核，2.7GHz

（2）内存：32G

（3）存储：1TB

（4）网卡：Intel千兆网卡

## 系统功能实现

前面分别介绍了用户信息系统的需求分析和设计，接下来主要实现该系统。下面对用户信息系统的具体实现以及实现结果做出简单介绍。图4-1为用户信息系统的项目目录结构。

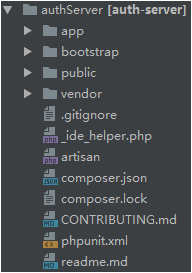


图4-1 AuthServer项目目录结构

### 登录实现

用户信息系统的登录功能是用户首先需要用到的功能，用户使用email和password进行登录验证，其中对password进行laravel框架集成的加密算法加密之后POST到后台进行验证，根据不同的email，用户拥有不同的权限，比如管理员、项目负责人、普通用户等。并且为了提高系统的易用性，我们实现了错误提示，对是否输入email和password以及email格式验证、密码格式验证、密码长度验证进行错误提示，当输入不合法的信息时，会在前端界面显示错误信息，可以减少用户登录失败次数，提升用户体验。并且系统根据session状态判断用户的在线状态，把用户的在线状态存储在active\_user数据表中，以便当用户操作时可以实时判断用户回话是否到期以及是否需要重新登录。登录界面如图4-2所示。

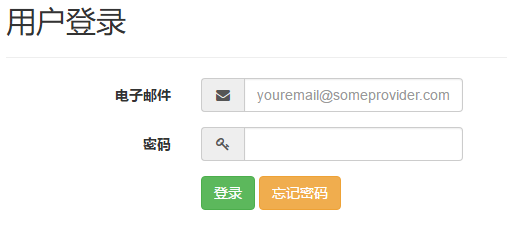


图4-2 用户信息系统登录界面

### 基于OAuth 2.0的身份凭证认证实现

基于OAuth 2.0的身份认证框架是此用户信息系统的基础支撑功能，该功能为整个用户信息系统提供了身份认证和授权功能。这一模块的功能我们主要使用基于League/oauth2-server的Lucadegasperi/oauth2-server-laravel的PHP库来实现OAuth 2.0认证模块。League/oauth2-server提供了OAuth 2.0的基本认证协议流程实现，Lucadegasperi/oauth2-server-laravel是在League/oauth2-server的基础上对其进行功能拓展和封装，使其更加便于接入使用，更加灵活。在本系统中，我们主要实现授权码认证模式和刷新令牌模式。

接下来主要是用Lucadegasperi/oauth2-server-laravel库提供的API实现OAuth 2.0认证流程：

（1）首先在第三方应用发起认证请求，发送GET请求到http://<OAuth-Server-IP>/auth/authorize?client\_id=client1id&client\_secret=client1secret&response\_type=code&redirect\_uri=http://<Client-IP>/callback。

（2）OAuth认证服务器端收到请求，在处理请求之前，先经过过滤器，有check-authorization-params和auth两个过滤器，check-authorization-params是Lucadegasperi/oauth2-server-laravel提供的过滤器，在向controller提交请求之前，会先检查参数的合法性和一一对应关系，比如client\_id，client\_secret，redirect\_uri等参数。auth过滤器可以通过Auth::check()判断用户登录状态，返回值为1则用户未登录，需要重定向到登录页面填写email和password进行身份认证，为0则代表用户已经处于登录状态，可以直接进行认证。在check-authorization-params过滤器中，如果参数的合法性和一一对应性检查通过后，会将这些参数保存在Session中，所以，请求通过过滤器到达controller后，可以从Session::get('authorize-params')中获取参数，获取参数之后可以将信息渲染到确认授权的前端页面。

（3）OAuth认证服务器端向客户端返回确认授权页面，当用户确认授权后，客户端回向OAuth认证服务器端发出POST请求， URL是http://<OAuth-Server-IP>/auth/authorize ?client\_id='.$params["client\_id"].'&redirect\_uri='.$params['redirect\_uri'].'&response\_type=code，POST的参数有client\_id、redirect\_uri、response\_type等。

（4）OAuth认证服务器端收到POST请求后，会将请求先经过过滤器，过滤器分别为check-authorization-params、auth、csrf，check-authorization-params和auth过滤器分别来处理参数验证和登录验证，csrf过滤器是laravel内置的过滤器，用来避免应用受到csrf（跨站请求伪造）攻击，前端对POST一个隐藏字段，该隐藏字段是csrf令牌，csrf token会随着参数一起POST到后台，由csrf过滤器来验证csrf token的合法性来确保安全。在controller中，同样，根据Session::get('authorize-params')获取经过校验的参数，通过laravel内置的认证模块Auth::user()->id获取登录用户的id，然后使用Lucadegasperi/oauth2-server-laravel的API获取获取Auth Code——AuthorizationServer::newAuthorizeRequest('user', $params['user\_id'], $params)，并将Auth Code保存在数据库中，与Client ID、Client Secret、Redirect URL等信息建立一一对应关系。然后，OAuth认证服务器携带Auth Code返回客户端——Redirect::to(AuthorizationServer::makeRedirectWithCode($code, $params))。

（5）客户端后台收到携带Auth Code的请求后，会在后台，用户不可见的情况下向OAuth认证服务器发送POST请求，URL是http://<OAuth-Server-IP>/auth/access\_token，参数为grant\_type为authorization\_code、client\_id、client\_secret、redirect\_uri和code。

（6）OAuth认证服务器端收到客户端的POST请求后，会执行Lucadegasperi/oauth2-server-laravel获取Access Token的API——AuthorizationServer::performAccessTokenFlow()，首先查询数据库验证参数合法性，然后生成Access Token，并保存至数据库，与Client ID、Client Secret、Redirect URL等信息建立一一对应关系，然后向客户端返回Access Token以及其属性信息。

（7）当客户端收到OAuth认证服务器返回的数据后，会将access\_token、token\_type、expires、expires\_in、refresh\_token等信息保存到客户端数据库。至此，客户端可以使用Access Token来获取资源服务器中的数据。

（8）当Access Token的生命周期结束后，如果需要刷新获取新的Access Token，客户端会执行7）的步骤，其中grant\_type 参数值替换成refresh\_token，code参数替换成refresh\_token，值为7）中获取的refresh\_token参数值。

### 组织管理实现

组织是用户信息系统最顶层组织结构，在本系统中，组织为中国科学院软件研究所总体部，所有项目组、成员、任务等业务操作都是在组织下进行的。组织管理包括对组织的增加、删除、修改、查询功能，其中对于组织的添加、删除、修改操作只有系统管理员才能进行。

组织管理功能模块为MVC三层架构，在Model层编写数据对象文件，使用laravel的Eloquent ORM对象虚拟映射框架与数据库进行交互，对于组织对象的增加、删除、修改、查询操作在Model文件——/app/models/Company.php中实现。Controller负责逻辑事务的处理，是连接View层和Model层的桥梁，用户通过View层向Controller发出请求，在Controller层中进行逻辑处理并从与Model层交互获取数据前要先经过过滤器判断用户是否具有管理员权限，然后将处理后的数据返回View层。View层主要使用Bootstrap框架实现，效果如图4-3所示。

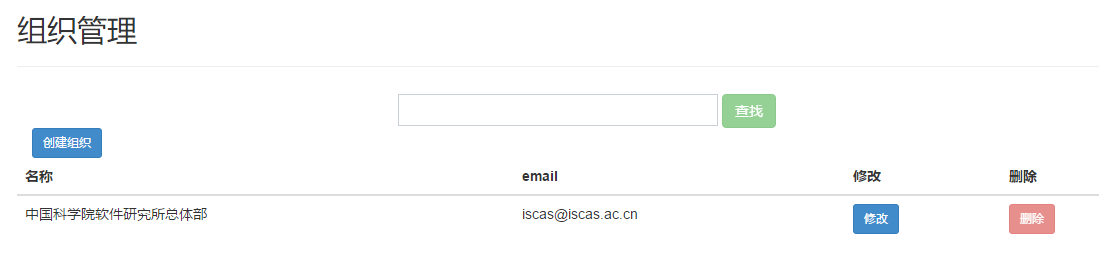


图4-3 组织管理页面

### 项目组管理实现

项目组是组织的下层机构，是一个树形结构。树形项目组结构如图4-4所示。



图4-4 树形项目组结构



图4-5 删除子项目组

这里的树形结构主要使用superid和leaf两个属性来实现，每层的项目组都可以无限的向下建立项目组，其子项目组的superid属性为其本身的id，如B、C、D的superid是A的id，当删除子项目组时，会将子项目组的superid设置为0，如删除F项目组后，F的superid置为0，其子项目组也随之删除，对于位于叶子上的项目组，其leaf值为1，如E、J、G、H、I的leaf值为1，其他项目组的leaf值为0，在叶子上的项目组其子节点可以为用户，其余的子节点是子项目组。项目组的修改、增加、删除、查询操作主要在Controller层和Model层实现，其中，修改、增加、删除只有管理员才能进行。View层视图界面如图4-6、图4-7所示。



图4-6 项目组信息界面

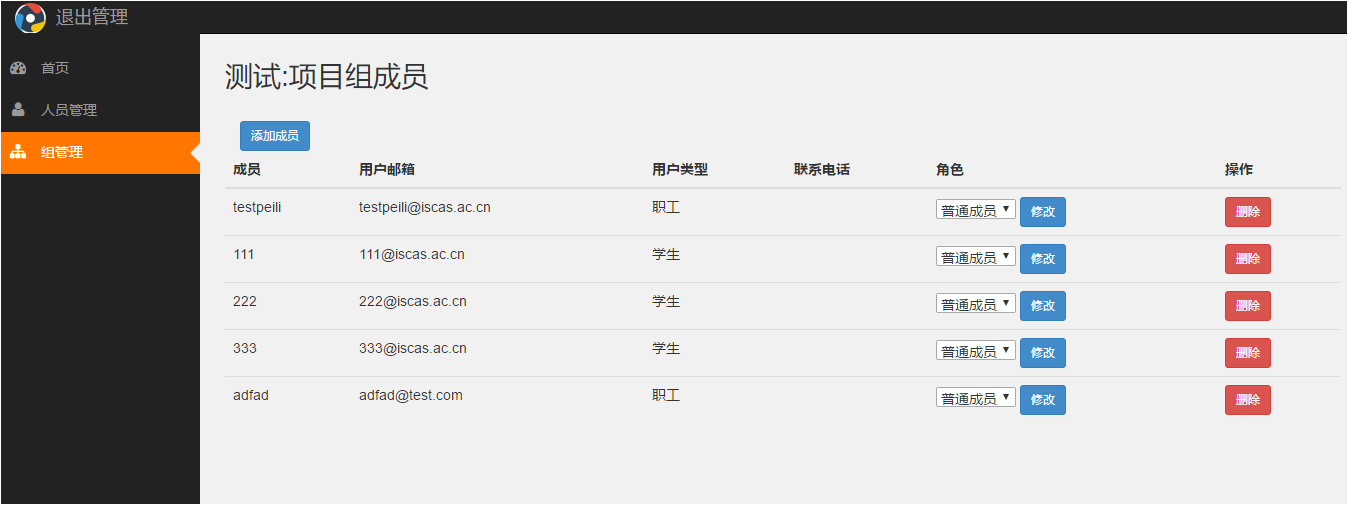


图4-7 成员管理界面

### 任务管理实现

在组织和下面项目组的日常开发工作中，会发布任务，系统管理员具有发布任务、删除任务、修改任务、查看任务、分配任务到组织、分配任务到项目组、分配任务到用户的权限，项目负责人具有发布任务、删除任务、修改任务、查看任务、分配任务到用户的权限，普通用户具有查看任务的权限。任务和项目组之间是多对多关系，任务和用户之间也是多对多关系，因此，在Model层的实现中，需要增加两个二元关系来表示任务和项目组、用户之间的多对多关系，所以，共有/app/models/Task.php、/app/models/TaskProject.php、/app/models/TaskMember.php，分别以task\_id、project\_id、user\_id建立二元关联关系，Controller层在根据请求对Model层进行处理前需要进行过滤器判断是否是系统管理员或者项目负责人，才能继续操作。View层前端视图如图4-8、图4-9所示。

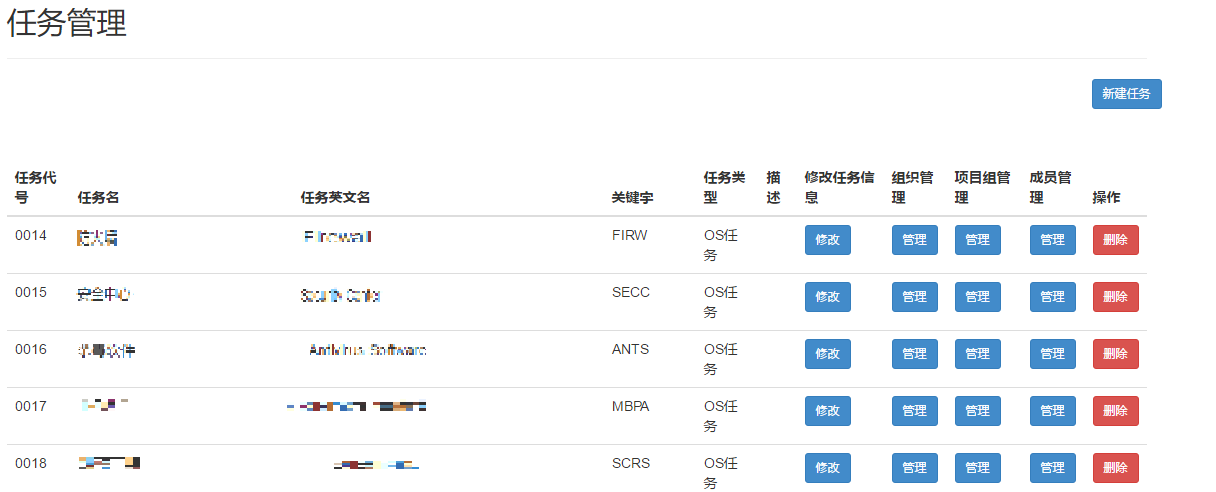


图4-8 任务管理界面



图4-9 新建任务界面

### 用户管理实现

人员管理是对组织下的人员进行查询、修改、删除、注册功能，对于普通用户只能进行查询操作。用户的属性如项目组ID、电话、Email、性别等信息的操作主要在Model层进行，在/app/models/User.php中实现了人员的查询、修改、删除、注册功能，并且与ActiveUser建立一一对应关系，一个用户最多只能有一个在线状态，在Model层实现根据Session判断用户在线状态并更新ActiveUser表的功能。在Controller层进行对应操作处理前需要经过过滤器判断用户是否具备管理员权限。View层前端视图如图4-10、图4-11所示。



图4-10 用户管理主界面



图4-11 新用户注册

### 客户端管理实现

客户端管理模块实现对第三方需要进行接入用户信息系统的应用进行注册。用户可以进行注册客户端、删除客户端，并且每个用户可以注册自己的第三方应用客户端，拥有独立的客户端管理功能。第三方应用注册客户端需要Client ID，选择客户端类型、Redirect URL、选择客户端权限。客户端的查询、注册、删除操作主要在/app/models/Client.php和Controller层中实现。View层前端视图如图4-12、图4-13所示。



图4-12 客户端主界面



图4-13 客户端注册

### 开放平台实现

开放平台主要实现用户信息系统对已注册的第三方应用开放的Open API，以及对这些Open API的管理。对于管理员，可以对这些Open API进行添加接口组、添加接口、修改接口、删除接口、测试接口，对于普通成员只能进行查看接口组、接口具体信息以及测试接口的操作。Model层有/app/models/ApiGroup.php和/app/models/ApiInfo.php分别处理API组合API的增加、删除、修改等操作。Controller层在处理请求前先经过判断是否具有管理员权限的过滤器，并且模拟API的HTTP请求，实现接口测试功能。View层前端视图如图4-14、图4-15所示。



图4-14 开放平台主界面

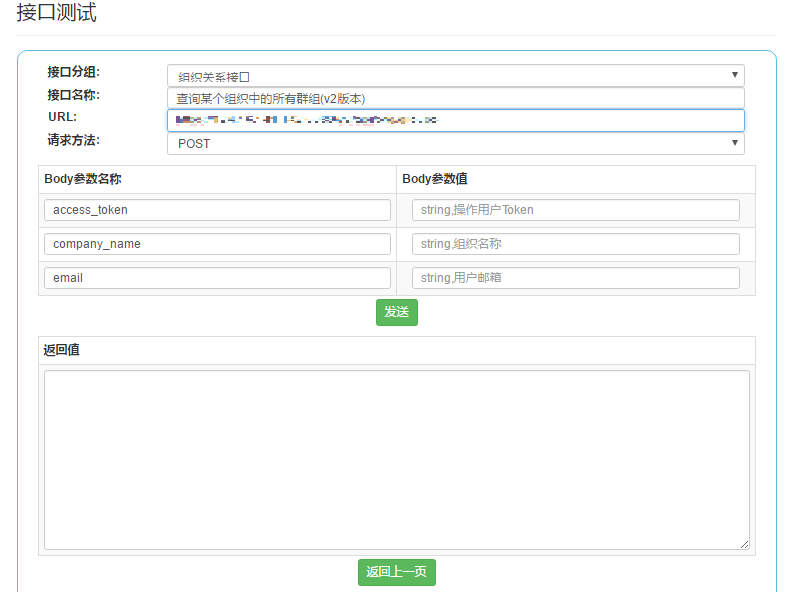


图4-15 接口测试

## 系统测试

### 单元测试

在系统开发中虽然采用诸多手段和方法，并尽可能的考虑多种原因场景和输入输出，但不可避免的可能会出现错误和缺陷，单元测试是针对系统的基本功能模块为单位进行测试的，以检查系统基本组成单位的正确性，其目的在于发现各模块内部可能存在的缺陷[29]。本系统的单元测试针对系统的各个功能模块开展测试，测试包括各个功能模块的子模块以及数据接口。系统的单元测试主要采用白盒测试的方法，辅以黑盒测试的方法，主要针对边界值和模块接口、错误处理进行测试。

下面主要以基于OAuth 2.0的登录模块为例，介绍了此模块的测试用例和测试结果。

表4-1 单元测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 测试流程 | 测试用例（邮箱/密码） | 测试结果 | 结论 |
| 1 | 用户登录 | zzyo1995@qq.com/ | 错误：密码为空 | 通过测试 |
| 2 | 用户登录 | zzyo1995@qq.com/12 | 错误：密码长度小于3 | 通过测试 |
| 3 | 用户登录 | zzyo1995/123456 | 错误：邮箱格式错误 | 通过测试 |
| 4 | 用户登录 | zzyo1995@qq.com/123456 | 进入个人主页 | 通过测试 |
| 5 | 用户登录 | 停止操作20分钟刷新页面 | 回话过期，进入登录页面 | 通过测试 |

### 集成测试

在针对系统的各个功能模块进行单元测试之后，需要将各个功能模块组装成系统，主要需要测试以下问题：组装模块时，模块之间的接口数据是否正确、有效；新集成的模块会不会对之前的功能模块产生消极影响；集成后的系统全局是否存在缺陷[30]。以上都是集成测试需要进行的测试，本系统主要采用自底向上的渐增式集成测试方法，在对各个功能模块进行单元测试的基础上，将这些模块逐渐组装成完整的系统，在不断集成的过程中，一边进行集成一边进行测试。

下表给出集成测试用例和测试结果。

表4-2 集成测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 测试流程 | 测试用例 | 测试结果 | 结论 |
| 1 | OAuth 2.0认证模块 | 客户端输入账号、密码 | 进入系统个人主页，显示各个子功能 | 通过测试 |
| 2 | 用户管理模块 | 管理员点击用户管理 | 进入用户管理页面，显示各个子功能 | 通过测试 |
| 3 | 组织管理模块 | 管理员点击组织管理 | 进入组织管理页面，显示各个子功能 | 通过测试 |
| 4 | 项目组管理模块 | 管理员点击项目组管理 | 进入项目组管理页面，显示各个子功能 | 通过测试 |
| 5 | 任务管理模块 | 管理员/项目负责人点击任务管理 | 进入任务管理页面，显示各个子功能 | 通过测试 |
| 6 | 客户端管理模块 | 点击客户端管理 | 进入客户端管理页面，显示各个子功能 | 通过测试 |
| 7 | 开放平台管理模块 | 管理员点击开放平台管理 | 进入开放平台管理页面，显示各个子功能 | 通过测试 |

### 性能测试

在进过对于用户信息系统的单元测试和集成测试之后，证明系统满足当前业务逻辑需求，并能正常运行。但是，我们需要将用户信息系统正式投入日常使用中，不仅需要业务逻辑的正确，也需要保证性能需求。因此，我们需要对系统进行性能测试。对于Web应用的性能测试主要采用自动化的测试工具进行模拟，对系统进行压力、负载测试[31]。下面主要使用著名的在线性能测试工具——Load Impact来测试系统页面加载速度和2500个并发用户的负载测试。

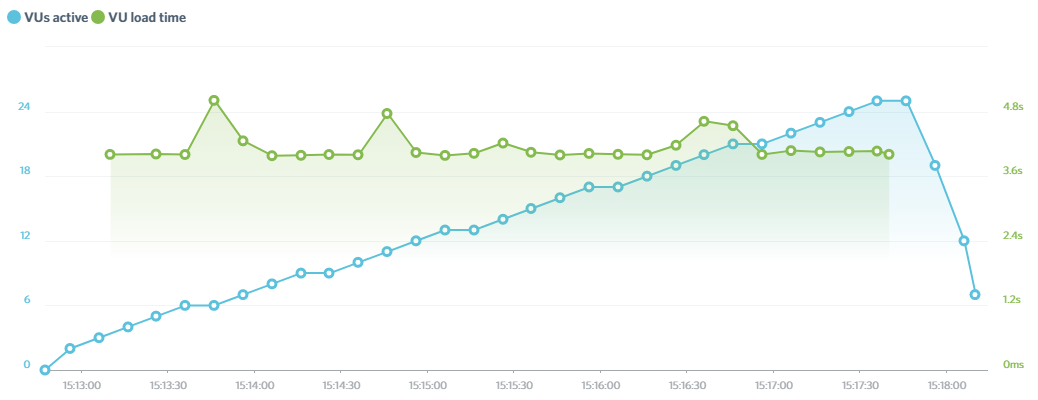


图4-16 性能测试报告

测试结果表明：在2500个并发用户的情况下系统的响应时间在4.8s以内。这一性能表现满足当前的需求。

## 本章小结

本章主要介绍用户信息系统的实现以及测试。实现部分介绍了开发、部署的软硬件环境，并从功能模块出发对实现的关键部分和效果截图进行介绍和展示。测试部分主要围绕着单元测试、集成测试、性能测试对系统的测试展开介绍，并验证系统满足需求，并可投入实际的生产应用中。

# 总结与展望

## 全文总结

用户信息系统主要基于MVC设计模式、PHP语言、Laravel框架、MySQL数据库等工具或技术进行开发，由于之前对于开发这样大型、投入实际使用的Web项目开发经验较少，对许多技术知识的学习较为浅显，因此，在开发过程中遇到了许多困难，在学校内和软件所的各位老师的帮助指导下，经过努力最终完成了用户信息系统的设计与实现。

本文主要介绍了基于OAuth 2.0的第三方用户信息系统的设计与实现。用户信息系统采用了当前较为流行的开发框架、开发技术、开发理念，使编码的效率大大提高，也提高了系统的可用性、易用性、安全性、和性能，提高了此系统使用人员的业务操作效率。本系统主要基于B/S架构进行开发，提高了系统的跨平台性，使系统更加方便移植和扩展。

下面对本文的工作作出总结：

（1）本文首先介绍用户信息系统的研究背景以及应用意义，简述了国内外研究现状，分析了国内外关于OAuth 2.0协议的应用场景和优点、缺陷，然后说明了本文的主要研究开发工作以及组织结构。

（2）介绍了当下比较流行的开发框架和开发技术、理念，如简洁易学、易部署、高性能的Web开发语言——PHP，基于PHP语言应用最广、最流行、开发效率最高的laravel框架，服务器端使用Apache，系统最核心的部分——OAuth 2.0协议，以及三元权限、Redis、RabbitMQ等工具和设计思想，为接下来的开发打好技术基础。

（3）然后对于系统的功能、业务、性能、安全、接口方面进行了详细的需求分析，使用系统体系架构图，系统功能模块图形象生动的介绍了系统实现的架构和系统的功能模块组成，并针对系统的功能模块设计、数据库设计、安全性设计等进行了详细说明。

（4）最后在前面需求分析和详细设计的基础上，对各个功能模块进行了编码实现和整合部署，根据不同的测试目的和测试方法对系统进行了测试，证明其符合系统的需求。

## 研究展望

由于开发时间、个人开发水平经验的不足，用户信息系统还存在许多不完善，对于系统的用户体验也需要提高。随着用户信息系统的使用，用户数量也会不断增加，用户对于功能的需求也不断增加。之后的开发过程中会将用户信息系统集成到社区中，满足各个子系统的单点登录，数据资源共享。本系统也需要许多等待完善的地方，比如，当前项目组、用户、任务三者之间的在日常的开发工作中的结构并不是十分科学，需要继续梳理三者之间的关系，建立科学的项目组-用户-任务模型；用户信息系统以后将对外进行开放，外部的第三方应用可以进行接入，用户的数量也会呈爆发式增加，对于系统的性能、安全性是较大的挑战。因此对于本系统还需要继续完善，这也是下一步需要进行研究开发的工作。

# 致谢

在这几个月的毕业设计开发中，由于本科所学知识范围所限和实际开发经验技术有限，在实际开发过程、论文文档写作过程中遇到了许多问题，所幸，在各位老师的帮助指导和自身不断学习探索下，最终顺利完成毕业设计，并达到了预期的目标。

首先感谢的是软件学院的卢力老师，他在我们整个毕业设计过程中认真负责，给我们提出了许多宝贵的意见和建议，并且非常辛苦的对我们的各个文档进行评阅、修改。在此，谨向卢力老师表示衷心的感谢。非常感谢中科院软件所的王青老师和李守斌老师，他们在项目开发中提出了很多技术指导，对我在开发过程中遇到的问题作出了耐心讲解和帮助。在此，向王青老师和李守斌老师表示万分感谢。最后，向我在本科学习生活和毕业设计中给我关心、支持的各位同学表示感谢！

# 参考文献

1. 时子庆, 刘金兰, 谭晓华. 基于OAuth2.0的认证授权技术[J]. 计算机系统应用, 2012, 21(3):260-264.
2. 黄延炜, 刘嘉勇. 新浪微博数据获取技术研究[J]. 信息安全与通信保密, 2013(6):71-73.
3. 朱宇飞. Web2.0时代的安全框架——OAuth[J]. 科技情报开发与经济, 2012, 22(17):80-81.
4. 卢慧锋, 赵文涛, 孙志峰,等. 社会化网络服务中OAuth2.0的应用研究与实现[J]. 计算机应用, 2014(a01):50-54.
5. 肖维明. 基于PHP+MySQL的网站开发[J]. 物流工程与管理, 2009, 31(6):90-92.
6. 霍素涛. 企业内网电子文件安全管理系统的研究与实现[D]. 华北电力大学(北京) 华北电力大学, 2011.
7. 邱祝文. 基于redis的分布式缓存系统架构研究[J]. 网络安全技术与应用, 2014(10):52-52.
8. 曾超宇, 李金香. Redis在高速缓存系统中的应用[J]. 微型机与应用, 2013, 32(12):11-13.
9. 张型龙, 李松犁, 肖俊超. 面向服务集成的工作流模型及其实现方法[J]. 计算机应用, 2015, 35(7):1993-1998.
10. 上官一雄. 学生公寓管理系统的设计与实现[J]. 西部大开发旬刊, 2012(4):60-60.
11. 陈敬勇. 网络化HRM系统关键技术研究与实现[D]. 大连交通大学, 2010.
12. 杨洁霞. 基于实用型的E-R图教学探讨[J]. 信息与电脑:理论版, 2010(6):178.
13. Mockus A, Fielding R T, Herbsleb J D. Two case studies of open source software development: Apache and Mozilla[M]. ACM, 2002.19-20
14. Chari S, Jutla C S, Roy A. Universally Composable Security Analysis of OAuth v2.0[J]. Iacr Cryptology Eprint Archive, 2011, 2011(2011).
15. Maatkamp M, Delden M V, Lekhac N A. Unidirectional Secure Information Transfer via RabbitMQ[J]. 2016.
16. 魏兴国. HTTP和HTTPS协议安全性分析[J]. 程序员, 2007(7):53-55.
17. Clark J, Oorschot P C V. SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements[J]. IEEE Symposium on Security & Privacy, 2013:511-525.
18. 黄智颖, 冯新喜, 张焕国. 哈希加密方案[J]. 通信技术, 2001(7):87-89.
19. 姜坚华. 双机热备系统的技术研究和具体实现[J]. 微型电脑应用, 2004, 20(3):7-9.
20. 刘晓洁, 黄永佳. 基于Linux的双机热备系统的实现技术[J]. 计算机应用研究, 2007, 24(4):255-257.
21. 张天琪. OAuth协议安全性研究[J]. 信息网络安全, 2013(3):68-70.
22. Xia X X, Xin B Y. Compare B/S mode with C/S mode[J]. Journal of Yanbian University, 2002.
23. Delft B V, Oostdijk M. A Security Analysis of OpenID[C]// IFIP Working Conference on Policies and Research in Identity Management. Springer, Berlin, Heidelberg, 2010:73-84.
24. 刘镝, 张智江, 张尼. 基于国内开放平台的Oauth认证框架研究[J]. 信息通信技术, 2011, 05(6):43-46.
25. He R Y. design and implementation of web based on laravel framework[C]// International Conference on Computer Science and Electronic Technology. 2015.
26. Halpin T. Object-Role Modeling (ORM/NIAM)[J]. 1998.
27. Pautasso C. RESTful Web service composition with BPEL for REST[J]. Data & Knowledge Engineering, 2009, 68(9):851-866.
28. 唐旭华, 邹峥嵘. 基于RESTful Web Services的空间数据共享[J]. 测绘科学, 2010, 35(4):122-124.
29. 张巍, 尹海波, 孙立财. 软件的单元测试方法[J]. 光电技术应用, 2006, 21(2):36-38.
30. 夏耘, 林华. 软件系统的集成测试技术研究[J]. 计算机工程, 2000, 26(10):102-104.
31. 段念. 软件性能测试过程详解与案例剖析[M]. 清华大学出版社, 2012.32~41