****

**本 科 毕 业 论 文（设计）**

|  |  |
| --- | --- |
| 课题名称 | 软件外包承接系统前端设计与实现 |
| 学 院 | 计算机科学与网络工程学院 |
| 专 业 | 网络工程 |
| 班级名称 | 网工162班 |
| 学生姓名 | 潘凯帆 |
| 学 号 | 1606100233 |
| 指导教师 | 金政哲 |
| 完成日期 | 2020年5月10日 |

教 务 处 制

软件外包承接系统前端设计与实现

摘要 互联网技术快速发展的当下，软件项目外包的需求随之出现，许多公司都在互联网里寻求开发人员来开发本公司的产品，然而互联网中的开发人员鱼龙混杂，难以分辨外包的开发人员能否满足软件项目的开发需求。针对这个问题，研究设计并实现面向熟人的软件外包承接系统，提供一个可靠的平台让人们在这发布项目、承接项目。本文所研究的系统为Web应用程序，前端基于Angular前端框架和Ant-UI组件库，利用HTML5的优点，为用户提供良好的使用体验。使用gitee进行项目的代码和版本管理，采用前后端分离模式进行项目开发。

本系统重点实现了项目的发布与承接、个人能力资质展示、个人评价等功能。

关键词 Angular；Web前端；软件外包承接系统；HTML5；

**ABSTRACT** With the rapid development of Internet technology, the demand for software project outsourcing has emerged. Many companies seek developers on the Internet to develop their products. However, the developers on the Internet are mixed, and it is difficult to tell whether the outsourced developers can meet the software The development needs of the project. In response to this problem, research and design and implement a software outsourcing undertaking system for acquaintances, providing a reliable platform for people to publish projects and undertake projects here. The system studied in this article is a web application. The front end is based on the Angular front-end framework and Ant-UI component library. It takes advantage of HTML5 to provide users with a good experience. Use gitee for project code and version management, and use front and back end separation mode for project development.

       This system focuses on the functions of project release and undertaking, personal ability qualification display, and personal evaluation.

**KEY WORDS** Angular；Web Front-end；Software outsourcing undertaking system

；HTML5；

目录

[1.前 言](#_Toc904276850_WPSOffice_Level1) [4](#_Toc904276850_WPSOffice_Level1)

[2.开发工具以及技术说明](#_Toc439248131_WPSOffice_Level1) [5](#_Toc439248131_WPSOffice_Level1)

[2.1 Angular](#_Toc904276850_WPSOffice_Level2) [5](#_Toc904276850_WPSOffice_Level2)

[2.2 Visual Studio Code](#_Toc439248131_WPSOffice_Level2) [5](#_Toc439248131_WPSOffice_Level2)

[2.3 Nginx](#_Toc1542042978_WPSOffice_Level2) [5](#_Toc1542042978_WPSOffice_Level2)

[2.4 HTML5](#_Toc1283679250_WPSOffice_Level2) [5](#_Toc1283679250_WPSOffice_Level2)

[2.5 Ant Design of Angular](#_Toc1176436988_WPSOffice_Level2) [5](#_Toc1176436988_WPSOffice_Level2)

[2.6 TypeScript](#_Toc494519387_WPSOffice_Level2) [5](#_Toc494519387_WPSOffice_Level2)

[2.7 Vegeta](#_Toc625623419_WPSOffice_Level2) [6](#_Toc625623419_WPSOffice_Level2)

[2.8 Docker](#_Toc772867421_WPSOffice_Level2) [6](#_Toc772867421_WPSOffice_Level2)

[3 需求分析](#_Toc1542042978_WPSOffice_Level1) [6](#_Toc1542042978_WPSOffice_Level1)

[3.1 系统设计目标](#_Toc1601647691_WPSOffice_Level2) [6](#_Toc1601647691_WPSOffice_Level2)

[3.2 系统目标用户分析](#_Toc185227492_WPSOffice_Level2) [6](#_Toc185227492_WPSOffice_Level2)

[3.3 可行性分析](#_Toc1414653541_WPSOffice_Level2) [6](#_Toc1414653541_WPSOffice_Level2)

[3.4 功能性需求分析](#_Toc1290607650_WPSOffice_Level2) [7](#_Toc1290607650_WPSOffice_Level2)

[4系统总体设计](#_Toc1283679250_WPSOffice_Level1) [7](#_Toc1283679250_WPSOffice_Level1)

[4.1系统概要设计](#_Toc1176436988_WPSOffice_Level1) [7](#_Toc1176436988_WPSOffice_Level1)

[4.2 系统业务功能设计](#_Toc494519387_WPSOffice_Level1) [8](#_Toc494519387_WPSOffice_Level1)

[4.2.1 登录](#_Toc1657938850_WPSOffice_Level2) [8](#_Toc1657938850_WPSOffice_Level2)

[4.2.2 注册](#_Toc1377932125_WPSOffice_Level2) [8](#_Toc1377932125_WPSOffice_Level2)

[4.2.3 找回密码](#_Toc441575627_WPSOffice_Level2) [8](#_Toc441575627_WPSOffice_Level2)

[4.2.4 导航栏](#_Toc2005562604_WPSOffice_Level2) [8](#_Toc2005562604_WPSOffice_Level2)

[4.2.5 项目列表](#_Toc587362116_WPSOffice_Level2) [8](#_Toc587362116_WPSOffice_Level2)

[4.2.6 项目详情](#_Toc1960242000_WPSOffice_Level2) [9](#_Toc1960242000_WPSOffice_Level2)

[4.2.7 发布项目](#_Toc1240665373_WPSOffice_Level2) [9](#_Toc1240665373_WPSOffice_Level2)

[4.2.8 邀请码](#_Toc1944195288_WPSOffice_Level2) [9](#_Toc1944195288_WPSOffice_Level2)

[4.2.9 评价](#_Toc2126516311_WPSOffice_Level2) [9](#_Toc2126516311_WPSOffice_Level2)

[4.2.10 邀请他人评价](#_Toc1936785603_WPSOffice_Level2) [9](#_Toc1936785603_WPSOffice_Level2)

[4.2.11 个人能力资质](#_Toc2145992042_WPSOffice_Level2) [9](#_Toc2145992042_WPSOffice_Level2)

[4.2.12 我的项目](#_Toc700398529_WPSOffice_Level2) [10](#_Toc700398529_WPSOffice_Level2)

[4.2.13 用户权限管理](#_Toc1240207696_WPSOffice_Level2) [10](#_Toc1240207696_WPSOffice_Level2)

[4.3 系统业务功能流程图](#_Toc625623419_WPSOffice_Level1) [10](#_Toc625623419_WPSOffice_Level1)

[5系统的具体实现和效果展示](#_Toc772867421_WPSOffice_Level1) [11](#_Toc772867421_WPSOffice_Level1)

[5.1 系统具体实现的简要说明](#_Toc694468890_WPSOffice_Level2) [11](#_Toc694468890_WPSOffice_Level2)

[5.2 系统业务功能的实现](#_Toc365012785_WPSOffice_Level2) [11](#_Toc365012785_WPSOffice_Level2)

[5.2.1 登录](#_Toc494519387_WPSOffice_Level3) [11](#_Toc494519387_WPSOffice_Level3)

[5.2.2 注册](#_Toc625623419_WPSOffice_Level3) [13](#_Toc625623419_WPSOffice_Level3)

[5.2.3 项目列表](#_Toc772867421_WPSOffice_Level3) [14](#_Toc772867421_WPSOffice_Level3)

[5.2.4 项目详情](#_Toc1601647691_WPSOffice_Level3) [16](#_Toc1601647691_WPSOffice_Level3)

[5.2.5 发布项目](#_Toc185227492_WPSOffice_Level3) [18](#_Toc185227492_WPSOffice_Level3)

[5.2.6 评价](#_Toc1414653541_WPSOffice_Level3) [22](#_Toc1414653541_WPSOffice_Level3)

[5.2.7 个人能力资质](#_Toc1290607650_WPSOffice_Level3) [23](#_Toc1290607650_WPSOffice_Level3)

[5.2.8 邀请码](#_Toc1657938850_WPSOffice_Level3) [30](#_Toc1657938850_WPSOffice_Level3)

[5.2.9 用户管理](#_Toc1377932125_WPSOffice_Level3) [31](#_Toc1377932125_WPSOffice_Level3)

[5.3 非业务功能的实现](#_Toc1556581663_WPSOffice_Level2) [33](#_Toc1556581663_WPSOffice_Level2)

[5.3.1 拦截器](#_Toc441575627_WPSOffice_Level3) [33](#_Toc441575627_WPSOffice_Level3)

[5.3.2 路由守卫](#_Toc2005562604_WPSOffice_Level3) [37](#_Toc2005562604_WPSOffice_Level3)

[5.3.3 Http请求封装](#_Toc587362116_WPSOffice_Level3) [39](#_Toc587362116_WPSOffice_Level3)

[5.4 前端应用部署](#_Toc822222287_WPSOffice_Level2) [44](#_Toc822222287_WPSOffice_Level2)

[5.5 API接口压力测试结果](#_Toc32709164_WPSOffice_Level2) [46](#_Toc32709164_WPSOffice_Level2)

[5.5.1 登录API](#_Toc1960242000_WPSOffice_Level3) [46](#_Toc1960242000_WPSOffice_Level3)

[5.5.2 获取项目列表API](#_Toc1240665373_WPSOffice_Level3) [46](#_Toc1240665373_WPSOffice_Level3)

[5.5.3 搜索用户API](#_Toc1944195288_WPSOffice_Level3) [46](#_Toc1944195288_WPSOffice_Level3)

[5.5.4 发布项目API](#_Toc2126516311_WPSOffice_Level3) [47](#_Toc2126516311_WPSOffice_Level3)

[5.5.5 获取用户个人能力资质API](#_Toc1936785603_WPSOffice_Level3) [47](#_Toc1936785603_WPSOffice_Level3)

[5.5.6 获取竞争者列表API](#_Toc2145992042_WPSOffice_Level3) [47](#_Toc2145992042_WPSOffice_Level3)

[5.5.7 修改个人信息API](#_Toc700398529_WPSOffice_Level3) [48](#_Toc700398529_WPSOffice_Level3)

[6结论](#_Toc1601647691_WPSOffice_Level1) [48](#_Toc1601647691_WPSOffice_Level1)

[参考文献](#_Toc185227492_WPSOffice_Level1) [49](#_Toc185227492_WPSOffice_Level1)

[致谢](#_Toc1414653541_WPSOffice_Level1) [49](#_Toc1414653541_WPSOffice_Level1)

软件外包承接系统前端设计与实现

1.前 言

本文所研究的软件外包承接系统，一大特点就是面向熟人的软件外包平台，想成为这个平台的用户必须要得到平台用户的邀请码才能注册成为新用户，并且要完善个人能力资质信息才允许承接外包项目。若想在平台发布外包项目需求，要向管理员申请发布权限，管理员查看申请者的个人能力资质信息认为其有条件发布外包项目需求后，方可将发布权限给予申请者。由于在当前的互联网中存在各种各样的诈骗，所以在互联网里寻找合适外包开发人员或者承接软件外包项目都要格外注意，担心自己会上当受骗，因此一个可靠的软件外包平台是必要的。而且软件外包承接系统为Web应用程序，用户只需要浏览器即可访问使用，无需再安装其他软件，项目采用了前后端分离的合作开发方式，采用前后端分离的模式原因在于传统的互联网Web软件开发中，程序员需要同时负责前后端的开发，很容易出现前后端代码高度耦合，使软件开发的效率降低也会增加代码的冗余。而在前后端分离的开发模式中，程序员就可以专注于自己方向的开发，即后端程序员主要负责业务逻辑、数据库操作等，并向前端程序员提供API接口，而前端则负责将API接口上获得的数据渲染到HTML页面上。在这种模式中，前后端可以有效的避免高度耦合，职责分明，分工明确，后端程序员只需要把每一个视图视为一个接口,前端程序员通过对API接口的访问来与数据库进行交互。通过gitee进行代码托管，并且在gitee上建立wiki，提供完善的技术文档，开发人员根据技术文档来设计前后端交互数据的格式，提高项目的开发效率，并且利用git的分布式代码管理特性，相对独立不影响协同开发。项目页面是采用HTML5来实现，原因在于从现有技术体系来看，HTML5是WEB前端开发的关键技术，该技术日益成熟，进一步增强WEB功能，产生深远影响。从目前技术发展来看，HTML5已经被应用在微软浏览器（IE）、谷歌浏览器（Chrome）等相对成熟的系统。

2.开发工具以及技术说明

2.1 Angular

Angular是使用HTML和TypeScript构建单页客户端应用程序的平台和框架，基本的构建块是NgModules，它为组件提供编译上下文。NgModules将相关代码收集到功能集中；Angular应用由一组NgModule定义。应用程序始终至少具有启用引导的根模块，并且通常具有更多功能模块。Angular是基于MVC的设计模式，在MVC模式中，模型和视图是分离的，通常视图里面不会有任何逻辑实现，而模型也不依赖于视图。同一个模型可能会有很多种不同的展示方式，同一个视图里可能也会有多种不同的模型，它们仅仅是通过Controlle联系在一起。在这种模式下，系统的耦合度大大降低，从而增强了系统的可维护性。

2.2 Visual Studio Code

Visual Studio Code是一个由微软开发，同时支持Windows 、 Linux和macOS等操作系统且开放源代码的代码编辑器，它支持测试，并内置了Git 版本控制功能，同时也具有开发环境功能，例如代码补全、代码片段和代码重构等。

2.3 Nginx

Nginx是一款轻量级的Web服务器、反向代理服务器，由于它的内存占用少，启动极快，高并发能力强，在互联网项目中广泛应用。反向代理（Reverse Proxy）方式是指以代理服务器来接受Internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给Internet上请求连接的客户端，此时代理服务器对外就表现为一个服务器。

2.4 HTML5

HTML5是HTML最新修订的版本，用于编写前端界面，新增了许多新的特性，例如：用于媒体回放的video和audio、更好的支持本地离线存储等。

2.5 Ant Design of Angular

Ant Design是一个开箱即用的高质量 Angular 组件，与 Angular 保持同步升级，提炼自企业级中后台产品的交互语言和视觉风格，支持 OnPush 模式，性能卓越。

2.6 TypeScript

TypeScript 是 JavaScript 的超集，具有静态类型特性，旨在简化大型 JavaScript 应用程序的开发，也被称为 **JavaScript that scales**（**可拓展的 JavaScript**）。TypeScript的有点在于能在编译时检查不同部分代码的正确性。在编译时检查出错误，便于开发者发现错误的位置和具体问题，类型本身就是一种文档信息，方便日后开发者本人或者其他开发者查询。

2.7 Vegeta

Vegeta是一种多功能的HTTP负载测试工具，其构建目的是为了以恒定的请求速率钻取HTTP服务。可以在github下载编译好的程序直接使用，也可以在golang开发中作为库函数使用。

2.8 Docker

Docker属于一种Linux容器的封装，是一个开源的应用容器引擎，是基于golang编写实现的。可以把开发好的应用以及依赖库打包成一个镜像，然后使用Docker运行这个镜像就会生成一个虚拟容器，应用在虚拟容器运行就好像在真实的物理主机运行一样，并且不用担心运行环境问题。

3 需求分析

3.1 系统设计目标

本系统的目标在于实现外包软件项目发布与承接的信息化管理，提供一个页面让平台用户可以展示自己的擅长的专业领域，通过系统邮件的方式通知用户项目的有关情况。

3.2 系统目标用户分析

经过分析，定义系统中的用户角色如下：

系统管理员，负责管理本系统内用户的权限，例如发布项目、邀请新用户等权限。

项目发布者，拥有发布项目的权限，可从参与竞标的用户中选择合适的人选成为此项目的承接者。

普通用户，可以在项目列表中查看项目需求以及其他相关信息，若对某项目感兴趣可以参与项目竞标，竞标成功则成为项目的承接者，与项目发布在进一步沟通。

3.3 可行性分析

（1）技术可行性分析：本系统采用前后端分离的模式进行开发，即前后端开发各分工一人，前端开发采用web开发三大框架之一Angular进行开发，后端采用golang进行开发，前后端采用http请求来进行数据交互，交互数据格式采用JOSN，JSON数据格式的优势在于易读性以及方便编写，使用关系型数据库Mysql进行数据存储，使用redis作为缓存。

以上所使用的技术在当前都是比较热门、常见的，对于掌握计算机开发技术的学生来说，是可以实现此系统的。

（2）经济可行性：目前使用到的技术都是在github开源的，因此在软件、硬件成本上免费的，而开发此系统是作为毕业设计不用于商业用途，所以从经济上说是可行的。

3.4 功能性需求分析

（1）用户登录注册：必须通过账号登录才允许进入本系统，使用系统业务功能

（2）项目：用户在本系统发布的项目需求信息，或者在此承接感兴趣的项目

（3）个人能力资质：主要用于展示用户的专业领域

（4）评价：通过他人的评价来进一步判断用户的专业能力

（5）用户权限管理：系统管理员才能使用的功能，管理平台用户可以使用哪些 系统业务功能

4系统总体设计

4.1系统概要设计

本系统主要分为三层，分别为前端展示层，后端业务逻辑层，数据存储层。

前端展示层：Web页面，通过页面展示项目信息列表、用户个人能力资质等，用户在页面操作后，前端通过HTTP请求访问后端提供的API接口，得到后端反馈JSON数据后解析请求结果，并把最终结果呈现给用户。

后端业务逻辑层：向前端提供HTTP协议的API接口，在服务器中监听HTTP请求，根据前端发送的HTTP请求进行相应的业务逻辑处理以及对数据层进行操作。

数据层：在本系统中，通过文件系统存储用户上传的头像图片，关系数据库Mysql存储项目详细信息，用户个人能力资质等系统数据，内存数据库Redis 存储session用于用户账号的辨别。

4.2 系统业务功能设计

4.2.1 登录

用户登录账号，登录方式有两种，第一种是输入账号/邮箱和密码进行登录，若用户输入一个不存在的账号、输入错误的密码或者没有注册的邮箱，以浅谈窗的形式提示用户账号或密码错误。第二种为输入用户的个人邮箱（该邮箱必须为注册账号时输入的邮箱），系统会自动发送登录验证码到用户邮箱，输入得到的验证码，若验证码输入错误或者输入一个未注册的邮箱，弹出验证码错误或邮箱未注册的错误提示。成功登录则自动跳转到项目页面。此外本页面还包括两个页面跳转按钮，一个跳转至找回密码页面，另一个跳转至注册页面。

4.2.2 注册

用于注册新账号，需要填写的信息有：登录账号，用户名，密码（输入时提示密码必须最少6位），二次输入密码，个人邮箱，验证码（点击发送验证码则会在邮箱里得到验证码），邀请码。点击注册会验证用户输入的验证码、邀请码和邮箱格式是否正确，邮箱是否已被使用，账号是否唯一，如果填写有误，提示用户输入信息存在错误。注册成功后有两个页面跳转按钮：完善个人信息，直接进入平台。

4.2.3 找回密码

用户忘记账号密码时可以通过以下操作重置密码，输入账号所绑定的邮箱和新密码，通过点击发送验证码将验证码发送至用户邮箱，输入系统发送的验证码，验证码正确后提示重置密码成功，跳转至登录页面。

4.2.4 导航栏

导航栏用于跳转至不同的页面，根据用户的权限不同，导航栏的跳转页面内容有所不同，通用的跳转按钮有：项目，评价，邀请码，个人能力资质，我的项目，退出。拥有发布项目权限的用户，导航栏上会多一个发布项目按钮，管理员则在拥有发布项目权限的用户的基础上多一个管理按钮。

4.2.5 项目列表

显示所有的项目信息简介，包括项目名称、项目预算、项目发布预期、项目需求、项目周期、竞标人数、当前项目状态（项目状态有以下几个：等待承接、等待签订合同、签订合同、项目结束），此页面还提供项目筛选功能，可选择的筛选条件有：项目周期大小、项目预算、项目发布时间，用户还可以自行选择一个页面里显示多少条项目信息，默认一页显示10条项目信息。

4.2.6 项目详情

若用户对某个项目感兴趣，可以点击进入该项目的详情页面，项目详情页面显示的信息有该项目的名称、预算、周期、发布日期、需求描述、竞争者列表。若一个项目里有多个竞争者，则以分页形式来显示各个竞争者的信息简介，可以点击竞争者的头像跳转至竞争者的用户信息页面进一步查看竞争者的详细信息，若想承接此项目，点击我要承接即可参与项目承接的竞标，此时项目发布者会收到邮件通知有新的竞争者想承接项目。

4.2.7 发布项目

用户发布新项目需求，用户需要有发布项目权限才能使用此功能，需要填写项目名称、周期、预算，通过富文本编辑器编写项目需求，发布成功后会自动跳转至我的项目页面。

4.2.8 邀请码

每个用户都有自己专属的邀请码，邀请码可以刷新获得新的邀请码，而且旧的邀请码会失效，无法再使用。

4.2.9 评价

用户的评价页面，包括两部分，第一部分为邀请的评价，可查看自己多他人的评价，若有他人邀请用户本人评价，可在最前面看见受到了他人的评价邀请，可直接评价或者查看他人的详细个人信息后再作评价。第二部分为对我的评价，可查看他人对自己的评价。

4.2.10 邀请他人评价

此页面需要搜索用户名来邀请自己想要邀请的用户来对本人进行评价，搜索功能支持模糊搜索，邀请用户可以邀请至少一个平台用户来对本人进行评价，以及该页面有二级导航栏，可以通过二级导航栏返回上次跳转前的页面。

4.2.11 个人能力资质

此页面可以查看用户的详细信息，若查看的是用户本人的信息，可以在此页面对本人的信息进行修改编辑。用户的个人能力资质以四个模块来显示。

第一个模块为个人基础信息包括：头像，用户名、个人邮箱、个人微信、电话号码、性别、现住址，点击编辑可进行修改，若电话号码和邮箱的格式错误会提示，此部分的所以信息都是必填项，如果有一部分未填写或者信息格式有误则无法保存修改，点击修改头像可以上传本地图片到服务器，该图片将会是新的用户头像。

第二个模块为评价列表，可以查看他人的评价，点击邀请评价按钮则跳转至邀请他人评价页面。

第三个模块为擅长领域，显示用户的擅长专业领域，可在此修改擅长领域，修改时以标签形式显示系统提供的默认专业领域，若系统提供的专业领域没有合适的，用户可自行添加。

第四个模块为证明材料，显示的是那个证明用户专业能力的材料信息，例如证书，学历等。点击编辑可以对材料进行修改，以富文本的形式来编辑证明材料，证明材料可以为图片、文字两种形式，以此增加用户本身的专业能力和可靠性。

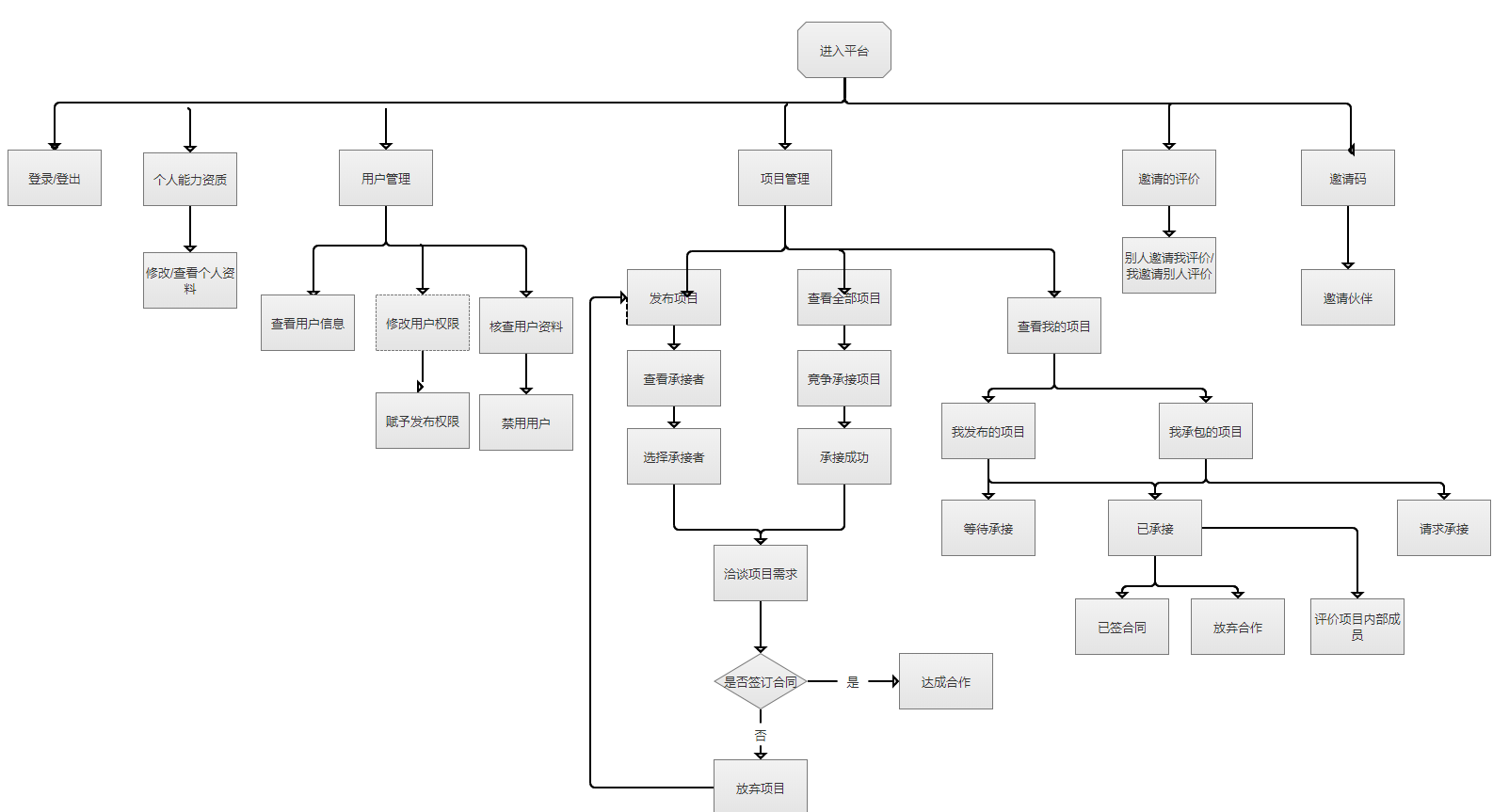
4.2.12 我的项目

我的项目页面分为两个部分，第一个部分为我创建的项目，可以查看目前项目的状态，有多少人竞争，也可以点击跳转至项目详情页面，此时项目的竞争者中有合适的可以选择该名竞争者为此项目的承接人，项目状态会改为等待签订合同，被选为承接者的用户会收到系统邮件通知，项目发布者与承接者洽谈成功后，项目状态会改为签订合同，承接者交付项目后项目状态会改为项目结束。第二部分为我承包的项目，可查看自己参与竞标的项目状态。该页面有二级导航栏，可以通过此导航栏选择页面显示不同状态的项目，导航栏会显示每个项目状态的统计数量。

4.2.13 用户权限管理

只有系统管理员才能使用此功能，主要用于给平台用户赋予或收回系统业务功能使用权限，包括：发布项目、承接项目、邀请他人评价或评价他人、邀请新用户。若某平台用户存在严重违规行为，系统管理员可禁用此用户账号，而此账号则不能登录进入平台，也可重新解禁账号

4.3 系统业务功能流程图



5系统的具体实现和效果展示

5.1 系统具体实现的简要说明

根据上述的需求分析和业务功能设计，本章主要介绍系统功能的具体实现，以及前端效果展示。

5.2 系统业务功能的实现

5.2.1 登录

登录页面如图5-1所示：

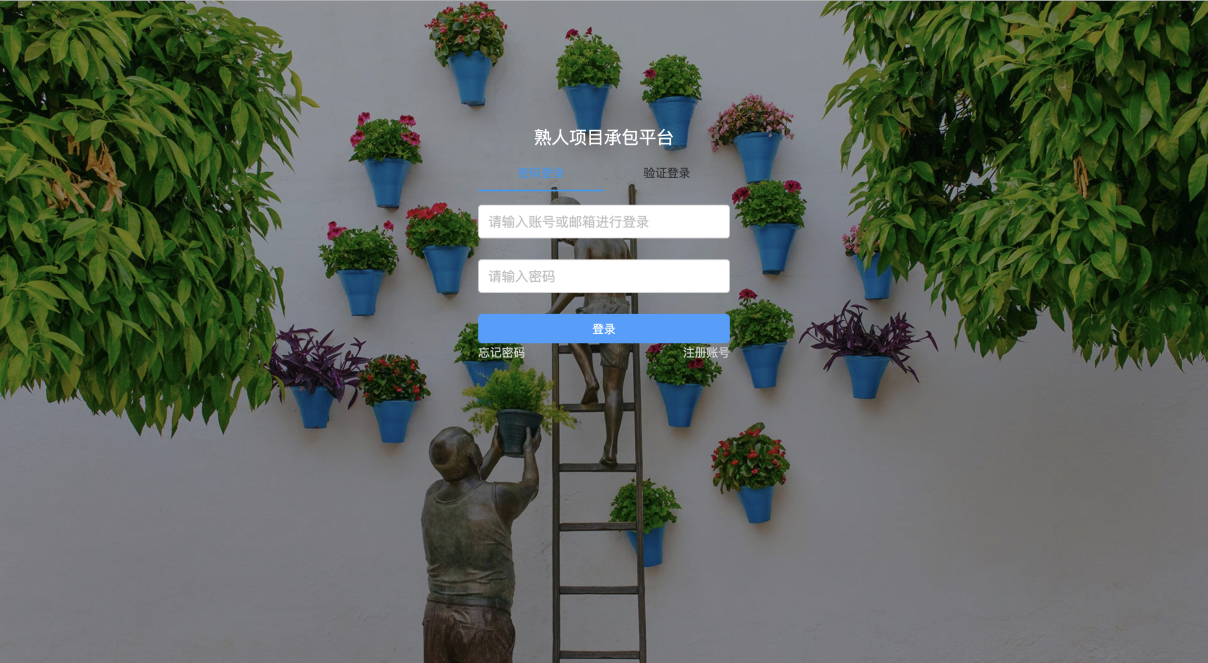


图5-1 登录页面

登录页面的背景为Bing每日图片，微软的Bing搜索引擎首页每天都会提供了一些有趣的图片，而这些图片很多都是有故事含义的，因此很适合作为本系统登录页面的背景图片。

以下代码实现了将Bing每日图片作为登录页面的背景图

**.login\_body**{

background-image:url("https://api.dujin.org/bing/1920.php");

height:**100%**;

background-position: center;

background-size: cover;

}

根据业务功能的设计，用户登录有两种方式，一种为输入账号/邮箱和密码，具体实现为通过HTTP的GET请求方式，以URL的形式将账号/邮箱和密码传给后端。另一种为通过验证码登录，即输入邮箱后点击发送验证码，向后端请求邮件验证码发送，在邮箱中得到系统发送的验证码后，输入验证码即可登录，同样也是通过HTTP的GET请求方式将数据发送给后端。

后端返回成功示例如下：

{

"code": 0,

"data":{

"user\_name":"羽泉",//用户名

"permission":31,//用户权限

"level":4//用户身份等级

}

}

5.2.2 注册

注册页面如图5-2所示：

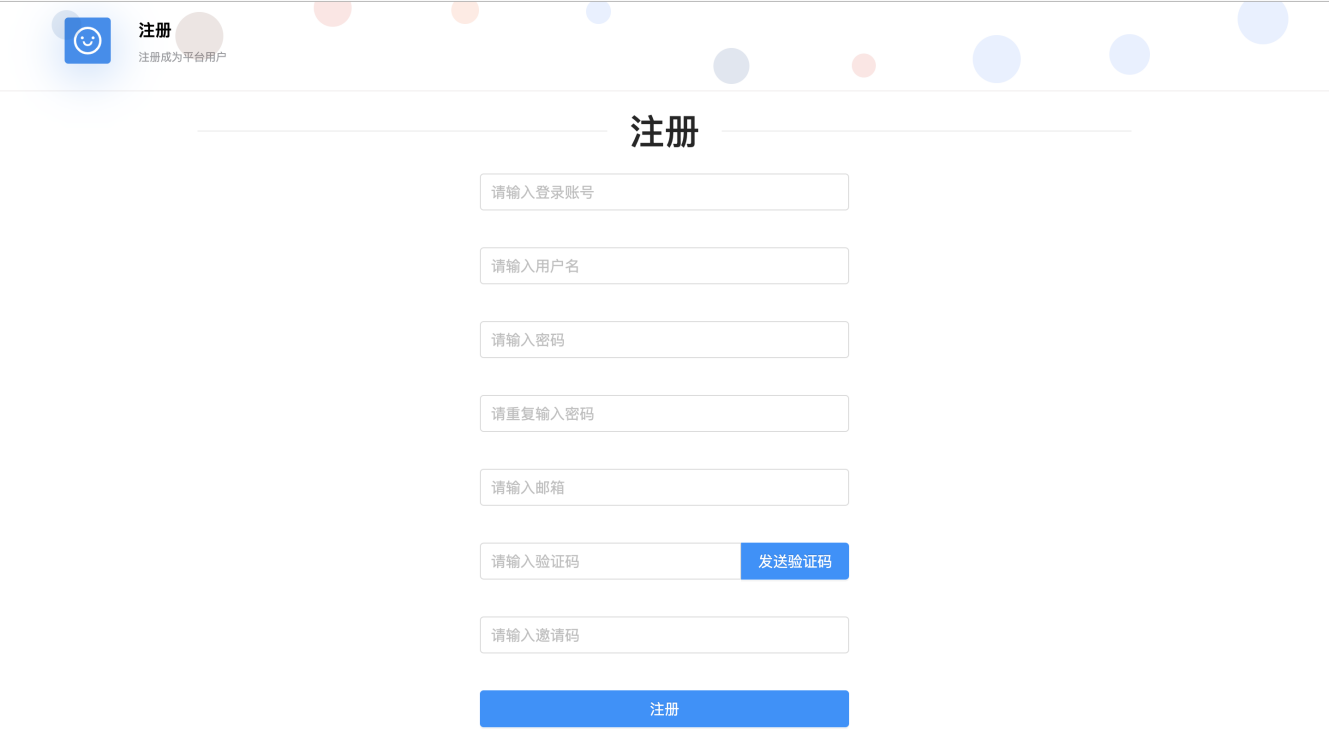


图5-2 注册页面

每一项信息都为必填项，如果其中一项信息没有提示，则无法提交数据到后端，数据提交前，会验证邮箱格式是否正确，两次密码的输入是否一致，通过HTTP的GET请求方式将数据发送至后端，发送的数据示例如下：

http://xxx/api/mail\_verification\_register?mail=MAIL&user\_name=USERNAME&password=PASSWORD&account=account&code=CODE&invitation\_code=INVITATIONCODE

如果邮箱或者登录账号已被使用，则会得到后端反馈的请求失败，示例如下：

{

code： 301, msg:该账号已存在

}

成功则得到以下数据：

{

code： 0

}

5.2.3 项目列表

项目列表页面如图5-3所示：



图5-3 项目列表页面

以HTTP的POST请求方式向后端请求项目数据，请求参数示例：

{

"cycle\_range\_start":0,//周期范围起始参数

"cycle\_range\_end":90,//周期范围末尾参数

"money\_range\_start":0,//金额范围起始参数

"money\_range\_end":5000000,//金额范围末尾参数

"date\_range\_start":1473083213,//发布日期范围起始参数

"date\_range\_end":1536155213,//发布日期末尾起始参数

"current\_page":1,//当前页数

"show\_nums":2//每页显示项目数量

}

通过修改上述请求参数里的数值来达到项目条件筛选以及分页显示的效果，而且采取分页来获取项目信息，可有效提高HTTP请求速率，不会因为项目数量的增加而导致请求速率下降。当数据量非常大时全表查询的速率是非常慢的，采用分页显示的方法可以有效避免后端操作数据库时数据库通过全表查询来获得数据。

后端返回成功数据示例：

{

"code": 200,

"data": [

{

"project\_name": "欢乐7酱",

"project\_id": "5",

"Publisher\_id": 2,

"project\_publish\_time": "2017-07-07",

"project\_circle": "77",

"project\_detail": "有趣的项目开发",

"project\_money": "777777",

"project\_status": 1,

"competitors\_nums": 1

},

{

"project\_name": "足彩app",

"project\_id": "4",

"Publisher\_id": 1,

"project\_publish\_time": "2018-08-05",

"project\_circle": "60",

"project\_detail": "好玩的app",

"project\_money": "120000",

"project\_status": 1,

"competitors\_nums": 2

}

]

}

5.2.4 项目详情

项目详情页面如图5-4所示：



图5-4 项目详情页面

此页面是同时多个HTTP的GET请求方式来分别获取项目详情信息，用户相对于该项目的身份、该项目竞争者列表。其中竞争者列表中的头像是可以点击的，点击后会跳转至该用户的个人能力资质页面，代码逻辑实现为将选择的用户UID传给后端对应的API接口，后端根据UID得到该名用户的个人能力资质数据返回给前端。

请求竞争者列表返回数据示例如下：

{

"code":200,

"data":[

"total":50,

"bidding\_list":[

{

"user\_id":1,

"head\_img":"2k123kslcvk34.img",

"name": " 陈宇",

"expertise\_field": ["Android","iOS","java"],

"address":"广东省 广州市 番禺区",

"finish\_project\_count":12,//完成项目数

"identity":1

},

{

"user\_id":2,

"head\_img":"2k123kslcvk34.img",

"name": " 陈宇",

"expertise\_field": ["Android","iOS","java"],

"address":"广东省 广州市 番禺区",

"finish\_project\_count":12,//完成项目数

"identity":0

},

]

]

}

若是项目发布者查看自己发布的详情项目，则如图5-5所示：



图5-5 项目详情页面（项目发布者视角）

5.2.5 发布项目

发布项目页面如图5-6所示

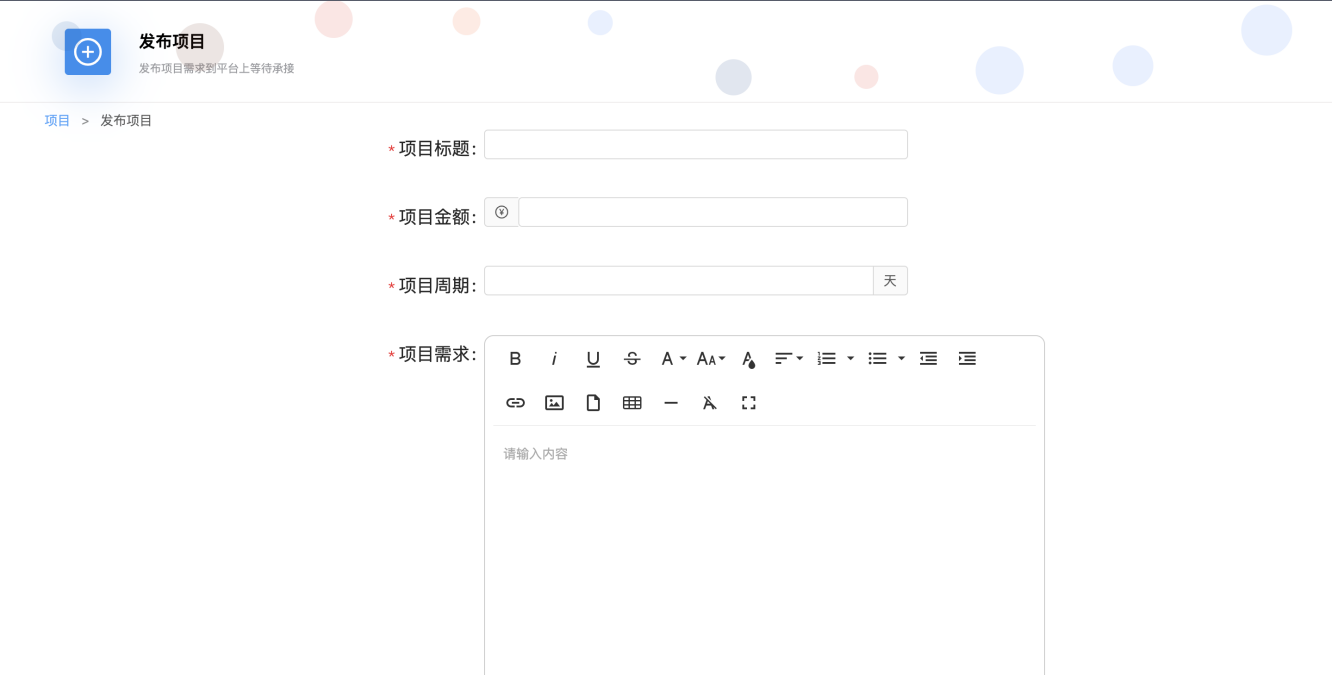


图5-6 发布项目页面

每一项信息都设定为必填项，图5-7为输入项目相关信息输入为空时的效果：



图5-7 输入为空示例

项目标题设定为长度不能超过20，项目金额与项目周期也做了输入限制，只允许输入数字以及小数点。而项目需求采用了Froala编辑器，基于Web的WYSIWYG富文本编辑器，用户可以图文描述项目需求，令项目需求更加清晰明了。

以下是实现限制输入的代码：

<nz-form-label class="input-label" [nzSm]="8" [nzXs]="24" nzRequired nzFor="title">项目标题</nz-form-label>

<nz-form-control [nzSm]="8" [nzXs]="24" >

<input type="text" nz-input formControlName="title" id="title" ng-minl>

<nz-form-explain \*ngIf="validateForm.get('title').dirty && validateForm.get('title').errors">

<ng-container \*ngIf="validateForm.get('title').hasError('required')">

标题不能为空

</ng-container>

<ng-container \*ngIf="validateForm.get('title').hasError('maxlength')">

标题字数不能超过20个字

</ng-container>

</nz-form-explain>

</nz-form-control>

<nz-form-label class="input-label" [nzSm]="8" [nzXs]="24" nzFor="money" nzRequired>项目金额</nz-form-label>

<nz-form-control [nzSm]="8" [nzXs]="24" >

<nz-input-group nzAddOnBeforeIcon="anticon anticon-pay-circle-o">

<nz-input-number formControlName="money" id="'money'" [nzMin]="1" [nzMax]="999999999999" style="width:100%"></nz-input-number>

</nz-input-group>

<nz-form-explain \*ngIf="validateForm.get('money').dirty && validateForm.get('money').errors">项目金额不能为空</nz-form-explain>

</nz-form-control>

<nz-form-label class="input-label" [nzSm]="8" [nzXs]="24" nzFor="cycle" nzRequired>项目周期</nz-form-label>

<nz-form-control [nzSm]="8" [nzXs]="24" >

<nz-input-group nzAddOnAfter="天">

<nz-input-number nz-input id="cycle" formControlName="cycle" placeholder="cycle" [nzMin]="1" [nzMax]="999999999999" style="width:100%"></nz-input-number>

</nz-input-group>

<nz-form-explain \*ngIf="validateForm.get('cycle').dirty && validateForm.get('cycle').errors">项目周期不能空</nz-form-explain>

</nz-form-control>

完善好项目信息后，点击发布项目，前端会以HTTP的POST请求方式将新增项目信息发送至后端，以下为请求参数示例：

{

"project\_name":"足球彩讯APP",//项目名称

"project\_circle":30,//项目周期

"project\_money":11000.00,//项目金额

"project\_detail":"<div>足球彩讯APP的项目需求</div>"//项目需求

}

后端接收到请求参数后会返回此次请求结果：

请求成功返回

{

"code":200

}

请求失败返回

{

"code":201,

"msg":"参数错误"

}

5.2.6 评价

评价页面如图5-8所示：



图5-8评价页面

用户点击导航栏的评价按钮即可跳转至此页面，通过HTTP的GET请求方式请求后端相对应的API接口，GET请求URL示例如下：

http://xxx/api/get\_evaluation\_to\_me?evaluate\_page=PAGEINDEX

后端响应此请求后，以JOSN数据格式返回前端请求的数据，返回成功示例如下：

{

"code":0,

"data":{

"evaluate\_total":2,//评价总数

"evaluate\_list":[

{

"estimator":"张三", //评论者

"estimator\_head\_img":"head\_image.jpg",//头像

"evaluate\_content":"这个人很牛",//评论内容

"evaluate\_time":"2020-02-02",//评论时间

"is\_evaluate":true,//是否评价

},

{

"estimator":"张三",

"estimator\_head\_img ":"head\_image.jpg",

"evaluate\_content":"其实这个人很菜",

"evaluate\_time":"2020-02-03",

"is\_evaluate":true,

}]

}

}

返回失败示例如下：

{

"code":201,

"msg":"参数错误"

}

5.2.7 个人能力资质

个人能力资质页面如图5-9所示

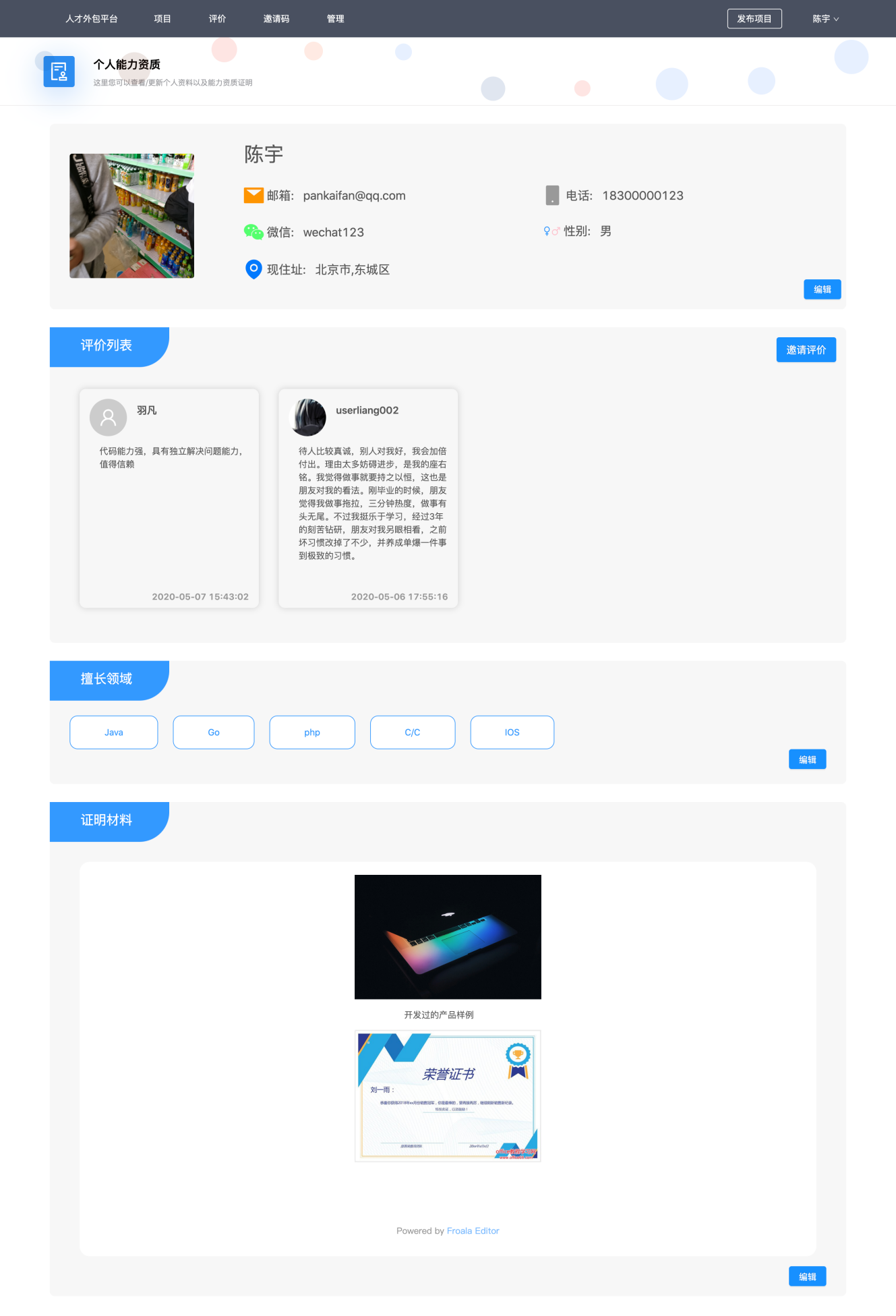


图5-9

此页面由个人基础信息、评价列表、擅长领域、证明材料这四部分组成，展示的数据是使用HTTP的GET方式向后端对应的API接口请求得到的。每一部分的修改数据都是相对独立的，不会影响到其他部分的数据修改，其中修改个人基础信息是通过HTTP的POST方式将指定参数发送至后端，而现住址的选择方式是省市区三级联动，省市区三级联动选择框使用Ant Design of Angular中的cascader级联选择实现的，代码示例如下：

**import** { China\_City } **from** '../../../const/const\_city';

**export** **class** QualificationComponent **implements** OnInit {

ngOnInit() {

**this**.title.setTitle('个人能力资质');

**this**.titleBar.**set**(**this**.titleinfo);

**this**.initUserIdentity()

**this**.initInformationData()

**this**.initFrom()

}

initFrom(){

**this**.FromUserName = **this**.fb.group({

userName:[**null**,[Validators.required]]

})

**this**.FromPhone = **this**.fb.group({

phone:[**null**,[Validators.required,**this**.phontValidator]]

})

**this**.FromWeChat = **this**.fb.group({

wechat:[**null**,[Validators.required]]

})

**this**.FromSex = **this**.fb.group({

sex:[**null**,[Validators.required]]

})

**this**.FromAddress = **this**.fb.group({

address:[**null**,Validators.required]

})

**this**.FromField = **this**.fb.group({

field\_defined:[**null**]

})

**this**.nzOptions = China\_City;

}

个人基础信息请求参数示例：

{

"username": " 陈宇",//用户名

"sex":"男",//性别

"wechat": "weixin\_223",//微信

"phone":"13800289212",//手机号码

"address":"广东省,广州市,番禺区",//现住址

"head\_img":"image.jpg"//头像

}

修改擅长领域是使用HTTP的GET请求将选中的领域标签数据发送至后端，服务端将接收到的领域标签数据更新至数据库中。

修改擅长领域URL示例：

http://xxx/api/update\_expertise\_field?fields="Java,Go"

证明材料的编辑同样采用了Froala编辑器，使用了HTTP的POST请求将编辑好的数据发送至服务端，如果上传数据中有图片，图片会存放在服务器的文件系统中，服务器接收到图片会自动生成图片的URL并把图片的URL存入数据库中。由于Froala编辑器在多处功能中有使用到，因此封装成一个公用组件，有需要的页面可直接引用，以下为Froala编辑器公共组件的代码逻辑：

import { Component, OnInit ,Input,Output,EventEmitter} from '@angular/core';

import { UPLOAD\_FILE } from '../../const/const\_project'

@Component({

selector: 'froala-editor',

templateUrl: './froala-editor.component.html',

styleUrls: ['./froala-editor.component.css']

})

export class FroalaEditorComponent implements OnInit {

froalaText:string;

@Output() froala = new EventEmitter<string>();

@Input() initialValues :string

public options:Object;

toolbarbuttons=['bold','italic', 'underline', 'strikeThrough',

'fontFamily', 'fontSize', 'textColor',

'align', 'formatOL', 'formatUL', 'outdent', 'indent','|','insertLink', 'insertImage', 'embedly', 'insertFile', 'insertTable',

'insertHR', 'clearFormatting',

'spellChecker', 'fullscreen'

];

constructor() {

this.froalaText = "";

}

ngOnInit() {

if(this.initialValues){

this.froalaText = this.initialValues

}

var that = this

this.options = {

language:"zh\_cn",

quickInsertEnabled:false,

placeholderText:"请输入内容",

charCounterCount:true,

charCounterMax:400,

toolbarButtons:this.toolbarbuttons,

toolbarButtonsSM:this.toolbarbuttons,

toolbarButtonsXS:this.toolbarbuttons,

codeMirror:false,

codeMirrorOptions:{

tabSize:4

},

//配置文件上传

fileUploadParam:'apcpfile',

fileUploadURL: UPLOAD\_FILE,

fileUploadMethod:'POST',

//配置图片上传

imageUploadParam:'apcpfile',

imageUploadMethod:'POST',

imageUploadURL: UPLOAD\_FILE,

//size

height:300,

// width:800,

events:{

'keyup':function(e,editor){

that.froala.emit(that.froalaText);

},

'blur':function(e,editor,touchendEvent){

that.froala.emit(that.froalaText);

},

'file.error':function(e,editor,error,response){

if (error.code == 1) {

console.log("Bad link")

}

// No link in upload response.

else if (error.code == 2) {

console.log(" No link in upload response.")

}

// Error during file upload.

else if (error.code == 3) {

console.log("Error during file upload.")

}

// Parsing response failed.

else if (error.code == 4) {

console.log("Parsing response failed.")

}

// File too text-large.

else if (error.code == 5) {

console.log("File too text-large.")

}

// Invalid file type.

else if (error.code == 6) {

console.log("Invalid file type")

}

// File can be uploaded only to same domain in IE 8 and IE 9.

else if (error.code == 7) {

console.log(" File can be uploaded only to same domain in IE 8 and IE 9.")

}

// Response contains the original server response to the request if available.

}

}

};

}

}

5.2.8 邀请码

邀请码页面如图5-10所示：



图5-10 邀请码页面

通过HTTP的GET请求从后端获得邀请码的数据，点击刷新邀请码同样是通HTTP的GET请求获得新的邀请码，而后端会将新生成的邀请码替代旧邀请码并存入数据库中，防止旧邀请码仍有效。

以下为获取/刷新邀请码后端返回的示例：

{

"code":200,

"data":{

"invitation\_code":"INVITATIONCODE"

}

}

5.2.9 用户管理

用户权限页面如图5-11所示：

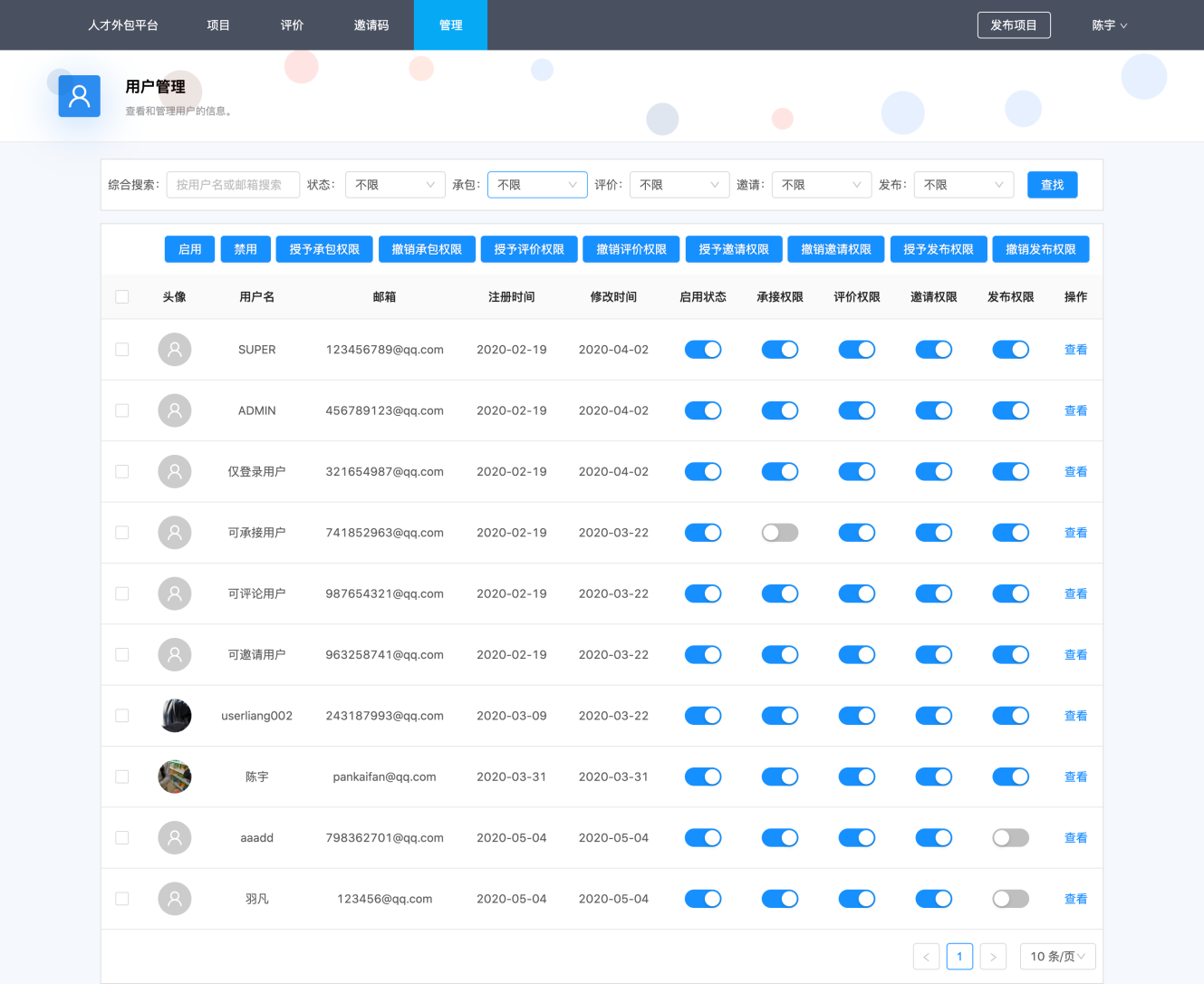


图5-11 用户管理页面

通过HTTP的POST请求更改用户权限，支持批量操作，选定复选框点击上部分的按钮即可同时给多个用户赋予使用权限，请求的参数包括用户ID和权限名称，示例如下：

{

"list":[

{

"user\_id":1,

"permission":31

},

{

"user\_id":2,

"permission":31

}

]

}

此页面实现了用户筛选功能，筛选条件包括用户名/邮箱以及各种使用权限。通过HTTP的POST请求将筛选条件发送至后端，后端通过数据库查询得到符合筛选条件的数据。请求参数示例如下：

{

keyword: "陈"，//关键字

num: 10, //每页显示数目

page: 1, //当前页数

select\_comment: "all", //评论权限

select\_invite: "all", //邀请新用户权限

select\_state: "all", //账号状态（启用、禁用）

select\_undertake\_project: "fale", // 承接项目权限

select\_release\_project: "true"发布项目权限

}

5.3 非业务功能的实现

5.3.1 拦截器

当用户没有登录或者cookie过期，拦截器会限制用户无法进入系统平台，直接输入其他页面的URL拦截器会重定向至登录页面，并提示用户重新登录。如果在跳转页面期间得到后端错误返回，拦截器会阻止当前页面跳转并提示后端返回的操作信息。以下为实现拦截器的逻辑代码：

**import** {Injectable} **from** '@angular/core';

**import** {HttpErrorResponse, HttpEvent, HttpHandler, HttpRequest, HttpResponse} **from** '@angular/common/http';

**import** {Observable, throwError} **from** 'rxjs';

**import** {catchError, tap, switchMap} **from** 'rxjs/operators';

**import** {Router} **from** '@angular/router';

**import** {ToastService} **from** '../../share/toast/toast.service';

**import** {ModalService} **from** '../../share/modal/modal.service';

**import** {environment} **from** '../../../environments/environment';

**import** {MockLoginService} **from** '../../share/mock-login/mock-login.service';

**import** { FB\_CODE } **from** '../../const/const\_err\_code';

**import** { BEFORE\_LOGIN } **from** '../../const/const\_routes';

**import** { URL\_WECHAT\_LOGIN } **from** '../../const/const\_api';

**import** { EMPTY } **from** 'rxjs';

**import** { IsloginService } **from** '../islogin.service';

@Injectable({

providedIn: 'root'

})

**export** **class** HttpInterceptorService {

**constructor**(

**private** rout: Router,

**private** toast: ToastService,

**private** route: Router,

**private** modal: ModalService,

**private** mockLoginService: MockLoginService,

**private** islogin:IsloginService

) {

}

intercept(req: HttpRequest<any>, next: HttpHandler):

Observable<HttpEvent<any>> {

// check param

**if** (!req) {

**this**.modal.error('need req param.');

}

**if** (!next) {

**this**.modal.error('need next param.');

}

**const** authReq = req.clone();

**return** next.handle(authReq).pipe(

tap(event => {

**if** (event **instanceof** HttpResponse) {

**if** (event.status == 200) {

// check response

**if** (!event.body) {

**this**.modal.error('The server return nothing.');

**return**;

}

**if** (!event.body.code) {

**if**(event.body.code!=0){

**this**.modal.error('The response struct returned has not code attribute.');

}

**return**;

}

/\* 如果后端返回错误:

\* 1,如果是cookie为空,cookie过期,sid无效,则跳转至登陆页面

\* 2,其他错误弹出模态框打印错误码及相关信息

\*\*/

**switch** (event.body.code) {

**case** FB\_CODE.GLOBAL\_SUCCESS.code:

// do nothing

**break**;

**case** FB\_CODE.COOKIE\_NULL.code:

**case** FB\_CODE.COOKIE\_EMPTY.code:

**case** FB\_CODE.ERROR\_INVALID\_SID.code:

**case** FB\_CODE.SESSION\_EXPIRED.code:

**this**.toast.error('请重新登陆!')

**this**.islogin.SetIsLogin(**false**)

**this**.route.navigate([BEFORE\_LOGIN.login.url]);

// this.toLogin(this.route);

**return**;

default:

// 有一些错误并不需要模态框弹出，所以 这里同样需要根据需要去添加

// this.modal.apiError(event.body.code.toString(), event.body.msg, event.url);

**return**;

}

}

}

// return event;

})

, catchError(err => {

**if** (err **instanceof** HttpErrorResponse) {

**return** throwError(err);

}

**return** throwError(err);

})

);

}

toLogin(route: any) {

// check param

**if** (!route) {

**this**.modal.error('need the route param.');

}

**if** (environment.BUILD\_TYPE == 'dev') {

**if** (**this**.route.url != '/mock\_login') {

**this**.mockLoginService.redirectURL = **this**.route.url;

}

route.navigate([BEFORE\_LOGIN.login.url]);

**return**;

}

**if** (environment.BUILD\_TYPE == 'deploy') {

**let** url = URL\_WECHAT\_LOGIN + '?redirectURL=' + route.url;

window.open(url, '\_self');

**return**;

}

}

}

5.3.2 路由守卫

通过路由守卫，使用户从一个页面导航至另一个页面，或者输入URL浏览器会导航至相应的页面。每个 [Route](https://angular.cn/api/router/Route) 都会把一个 URL 的 path 映射到一个组件，输入本系统不存在的URL时会重定向至404页面，如图5-12所示：



图5-12

路由守卫实现的代码如下：

**import** {NgModule} **from** '@angular/core';

**import** {RouterModule, Routes} **from** '@angular/router';

**import** {Page404Component} **from** './share/page-404/page-404.component';

**import** {MockLoginComponent} **from** './share/mock-login/mock-login.component';

**import** {LoginGuard} **from** './guard/login.guard';

**import** { LoginComponent } **from** './before\_login/login/login.component';

**import** { RegisterComponent } **from** './before\_login/register/register.component';

**import** { ForgerPasswordComponent } **from** './before\_login/forger-password/forger-password.component';

**import** { CompleteInfoComponent } **from** './before\_login/complete-info/complete-info.component';

**import** { HaveLoginGuard } **from** './guard/havelogin.guard';

**const** routes: Routes = [

{

path: '',

redirectTo: 'pc',

pathMatch: 'full',

},

{

path: 'pc',

loadChildren: './pc/pc.module#PcModule',

canActivate:[LoginGuard],

},

{

path:'login',

component:LoginComponent,

canActivate:[HaveLoginGuard],

},

{

path:'register',

component:RegisterComponent

},

{

path:'forget\_password',

component:ForgerPasswordComponent

},

{

path:'complete\_info',

component:CompleteInfoComponent,

canActivate:[LoginGuard],

},

{

path: 'super',

loadChildren: './super/super.module#SuperModule'

},

{

path: 'mock\_login',

component:MockLoginComponent,

},

{

path: '\*\*',

component: Page404Component

}

];

@NgModule({

imports: [RouterModule.forRoot(routes, {useHash: **true**})],

exports: [RouterModule]

})

**export** **class** AppRoutingModule { }

5.3.3 Http请求封装

由于前端代码实现当中，会多次使用到http的各种请求方式，因此封装了一个专门处理http请求的公用组件，减少代码的冗杂度以及错误信息统一处理。以下为实现此部分的逻辑代码：

**import** {Injectable} **from** '@angular/core';

**import** {HttpClient, HttpErrorResponse, HttpRequest} **from** '@angular/common/http';

**import** {Observable, throwError} **from** 'rxjs';

**import** {catchError} **from** 'rxjs/operators';

**import** {FbData} **from** '../../const/const\_project';

**import** {ModalService} **from** '../../share/modal/modal.service';

@Injectable({

providedIn: 'root'

})

**export** **class** PubHttpService {

**constructor**(

**private** httpclient: HttpClient,

**private** modal: ModalService,

) {

}

/\*\*

\* GetDataByHttpGet

\* @param url apiurl

\* @returns {Observable<FbData>}

\*/

GetDataByHttpGet(url: string, \_next:(data)=>**void**) {

**this**.httpclient.**get**<FbData>(url, {withCredentials: **true**})

.pipe(

catchError(**this**.handleErr(**this**.modal)),

)

.subscribe(

// check response.data

(response) => {

**if**(!response.data){

**this**.modal.error('The response struct returned has not data attribute.');

**return**;

}

\_next(response.data);

}

);

}

/\*\*

\* GetDataByPost

\* @param \_url apiurl

\* @param \_body postdata

\* @returns {Observable<FbData>}

\*/

GetDataByPost(\_url: string, \_body: object,\_next:(data)=>**void**) {

**this**.httpclient.post<FbData>(

\_url,

JSON.stringify(\_body),

{withCredentials: **true**})

.pipe(catchError(**this**.handleErr(**this**.modal)))

.subscribe(

(response) => {

// check response.data

**if**(!response.data){

**this**.modal.error('The response struct returned has not data attribute.');

**return**;

}

\_next(response.data);

}

);

}

/\*\*

\* get

\* @param url apiurl

\* @param httpparams url parameter

\* @returns {Observable<FbData>}

\*/

**get**(url: string,httpparams?:any) {

**return** **this**.httpclient.**get**<FbData>(url, {withCredentials: **true**,params:httpparams})

.pipe(

catchError(**this**.handleErr(**this**.modal)),

);

}

/\*\*

\* post

\* @param url apiurl

\* @param body JSON

\* @returns {Observable<FbData>}

\*/

post(url: string, body: any) {

**return** **this**.httpclient.post<FbData>(url, body, {

reportProgress: **true**, withCredentials: **true**,

}

).pipe(

catchError(**this**.handleErr(**this**.modal))

);

}

/\*\*

\* postjson

\* @param url apiurl

\* @param body postdata

\* @returns {Observable<FbData>}

\*/

postjson(url: string, body: object) {

**return** **this**.httpclient.post<FbData>(url, JSON.stringify(body), {withCredentials: **true**}).pipe(catchError(**this**.handleErr(**this**.modal)));

}

//上传Ex文件

UploadEx(url: string, formdata: any) {

**const** req = **new** HttpRequest('POST', url, formdata, {

reportProgress: **true**

});

**return** **this**.httpclient.request(req);

}

getWithOption(url: string, option: any) {

**return** **this**.**get**(url, option).pipe(catchError(**this**.handleErr(**this**.modal)));

}

**private** handleErr(modal: ModalService) {

**return** **function** handleErr(error: HttpErrorResponse) {

**if** (error.status == 0) {

modal.error('服务无响应,是否断网了?');

}

**if** (error.status == 400) {

modal.error('请求错误-400:' + error.message);

}

**if** (error.status == 404) {

modal.error('请求地址不存在-404!');

}

**if** (error.status == 500) {

modal.error('服务器错误-500!');

}

// return an observable with a user-facing error message

**return** throwError(error);

};

}

}

5.4 前端应用部署

本系统使用了Docker容器化技术，可使本系统易在各种系统服务器上部署。本系统的前端应用通过DaoCloud进行自动构建容器，监控前端代码在gitee的master分支是否有最新代码提交，若有最新代码提交则自动执行本地的DockerFile构建出最新版本的前端应用容器并发布至线上服务器。前端应用主要是通过Nginx来处理静态资源的，提供给外部用户访问页面资源，此外还需要对页面中的API请求进行转发，用户在界面进行操作触发请求后端API后，请求会先到达Nginx，Nginx再将流量转发到真正的服务端应用中。

以下为构建Docker容器的代码示例：

#ish2b/apcpfrontend\_build 为专属构建环境镜像

**FROM** ish2b/apcpfrontend\_build as buildenv

**COPY** ./ /APCP\_FRONTEND

**WORKDIR** /APCP\_FRONTEND

**ARG** build\_type=apcp\_build\_test

**RUN** npm install -g @angular/cli@9.0.5 && \

npm install --save-dev @angular-devkit/build-angular && \

ng version && \

npm run ${build\_type}

#运行环境镜像

**FROM** ish2b/apcpfrontend\_nginx

**RUN** mkdir /dist && mkdir /dist/apcp && mkdir /usr/local/nginx

#复制配置文件

**COPY** ./nginx-dep.conf /etc/nginx/conf.d/

**COPY** ./nginx.conf /etc/nginx/

#复制dist文件

**COPY** --from=buildenv /APCP\_FRONTEND/dist/apcp /dist/apcp

# 运行nginx

**RUN** rm /etc/nginx/conf.d/default.conf && \

mkdir /run/nginx && \

nginx -c /etc/nginx/nginx.conf && \

nginx -s stop

**EXPOSE** 80

**EXPOSE** 6619

**CMD** ["nginx", "-g", "daemon off;"]

5.5 API接口压力测试结果

由于本系统为Web应用程序，因此存在多人同时使用同一个业务功能的情况，进行压力测试可得知本系统性能情况。压力测试工具采用了golang编写的Vegeta。系统性能如何主要通过压力测试得到的QPS来判断，QPS是对一个服务器在规定时间内能处理多少请求的衡量标准。

5.5.1 登录API

设置了每秒10000次的http-get请求，成功发送http请求7860次，服务器成功响应714次。即本接口可在每秒处理714次http请求。具体结果如图5-13所示：



图5-13

5.5.2 获取项目列表API

设置了每秒1000次的http-post请求，成功发送http请求1000次，服务器成功响应864次。即本接口可在每秒处理864次http请求。具体结果如图5-14所示：

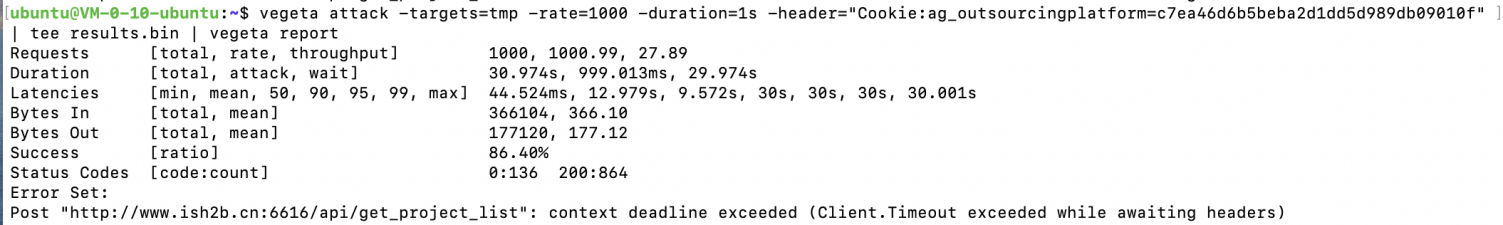


图 5-14

5.5.3 搜索用户API

设置了每秒1000次的http-post请求，成功发送http请求1000次，服务器成功响应413次。即本接口可在每秒处理413次http请求。具体结果如图5-15所示：

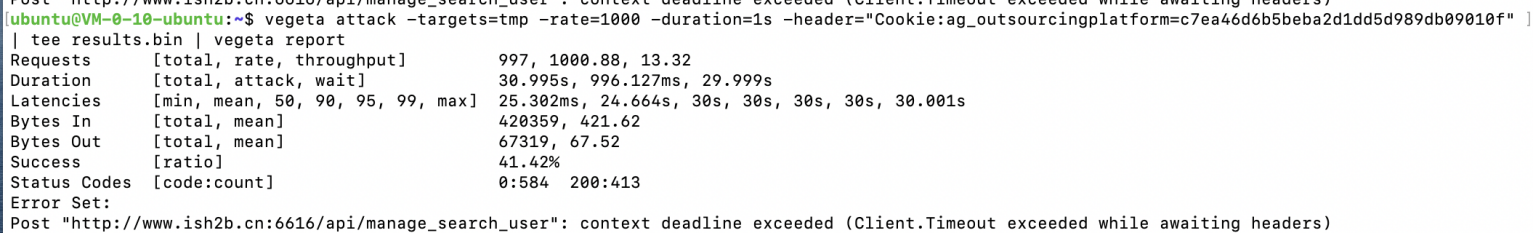


图 5-15

5.5.4 发布项目API

设置了每秒1000次的http-post请求，成功发送http请求1000次，服务器成功响应855次。即本接口可在每秒处理855次http请求。具体结果如图5-16所示：

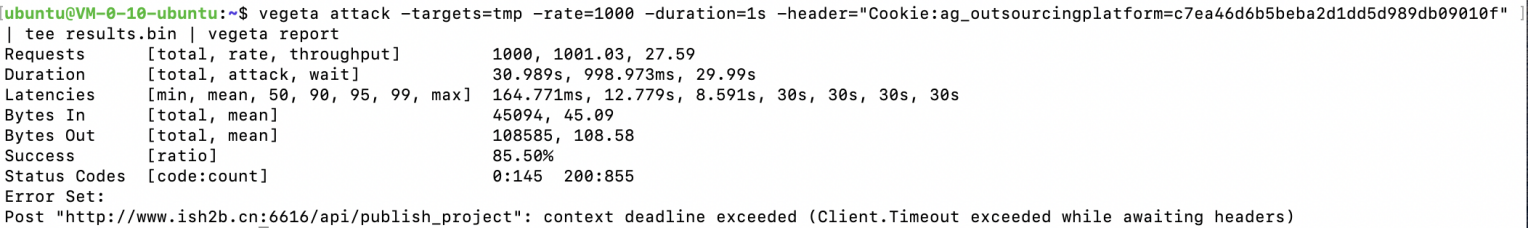


图 5-16

5.5.5 获取用户个人能力资质API

设置了每秒1000次的http-get请求，成功发送http请求1000次，服务器成功响应758次。即本接口可在每秒处理758次http请求。具体结果如图5-17所示：

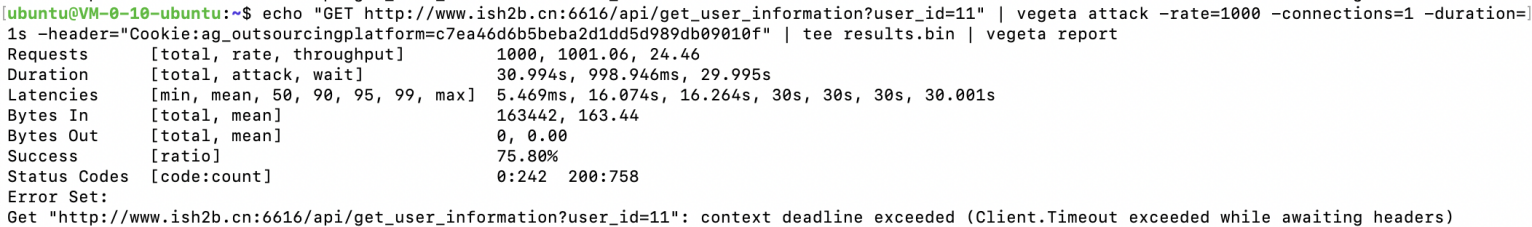


图 5-17

5.5.6 获取竞争者列表API

设置了每秒1000次的http-get请求，成功发送http请求1000次，服务器成功响应828次。即本接口可在每秒处理828次http请求。具体结果如图5-19所示：

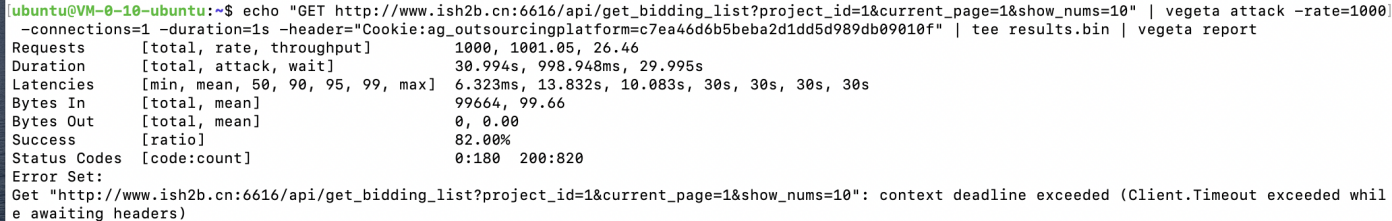


图5-19

5.5.7 修改个人信息API

设置了每秒1000次的http-post请求，成功发送http请求1000次，服务器成功响应951次。即本接口可在每秒处理951次http请求。具体结果如图5-20所示：

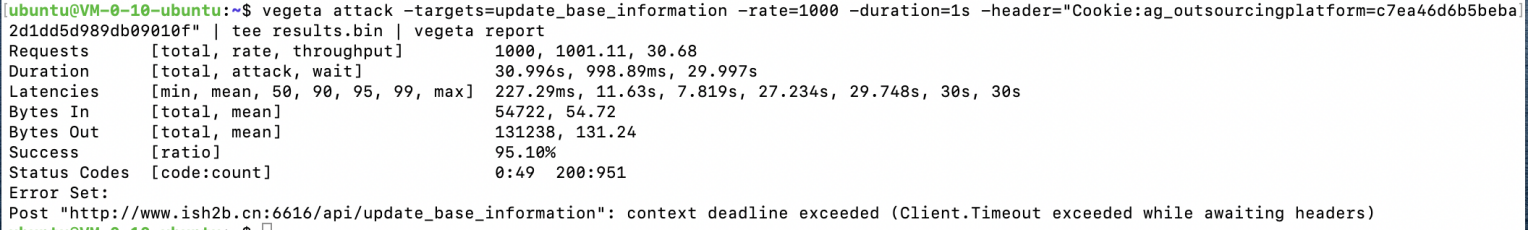


图 5-20

6结论

经过几个月的毕业设计实践，终于实现了本系统的最初版本并且在线上部署，在此过程中，了解了一个项目是如何从零开始，一个项目初期需要做什么准备。切身体会了软件工程中的几个关键流程，从需求分析，功能设计到编码实现，最后进行系统性能测试得到一个线上版本。本系统是团队合作的方式实现的，所以在实现过程中知道了团队开发是怎么样的，期间对功能设计、数据库设计存在不同的想法，经过多次讨论后统一了想法，认为当前的设计是每个人都能接受的并且可行的，还了解到了然后对系统编写单元测试，以及如何进行压力测试。

在毕业设计期间，还经历了春秋招，而在毕业设计里学习到的知识对春秋招有很大的帮助，从一开始只知道业务逻辑代码编写，到深入了解HTTP的不同方式请求、跨平台开发、不同框架的使用。毕业设计采用的开发流程与当前企业的开发流程，让我能够快速上手实习期间的工作。

此次的毕业设计期间，我遇到过各种各样的问题，包括技术上的，团队合作上的问题，我在解决问题中学到了很多新的知识，最重要的是提升了自己自学与独立思考解决方法的能力，通过毕业设计了解到自己大学四年学到了什么，有什么地方需改改进，还有如何对待一个问题的出现。

参考文献

[1] Sau Sheong Chang.Go Web Programming[M].Manning,2015:325.  
[2] 王建,罗政,张希,张梦琪,张科,马文成.Web项目前后端分离的设计与实现[J].软件工程,2020,23(04):22-24.

[3] 焦新伟. HTML5在WEB前端开发中的应用研究[J]. 网络安全技术与应用,2020(04):73-75.  
[4] Josiah L. Carlson.Redis实战[M].人民邮电出版社,2015:300.  
[5] 谢希仁.计算机网络（第六版)[M].北京:电子工业出版社,2013.  
[6] 柯采. 基于J2EE和MVC模式的Web应用研究[J]. 软件,2020,41(03):165-167.  
[7] 黄明和等编著.数据结构[M].江西教育出版社,1998:283.  
[8] 肖文娟,王加胜. 基于Vue和Spring Boot的校园记录管理Web App的设计与实现[J]. 计算机应用与软件,2020,37(04):25-30+88.

[9] Jonathan Baier.Getting Started with Kubernetes[M].Packt Publishing,2016:137.  
[10] Hans-Jurgen Schonig.Troubleshooting PostgreSQL[M].Packt Publishing - ebooks Account,2015:157.  
[11] Salahaldin Juba,Achim Vannahme,Andrey Volkov.Learning PostgreSQL[M].Packt Publishing - ebooks Account,2015:464.  
[12]Chitij Chauhan.PostgreSQL Cookbook[M].Packt Publishing,2015:286.  
[13]Deepak Vohra.Kubernetes Microservices with Docker[M].Apress,2016:432.  
[14]李子骅.Redis入门指南(第2版)[M].人民邮电出版社,2015:213.  
[15]Vinoo Das.Learning Redis[M].Packt Publishing - ebooks Account,2015:412.  
[16]上野宣.图解HTTP[M].人民邮电出版社,2014:250.

致谢

时光飞逝，转眼间大学的四年学生时光已经接近尾声，这也代表着即将离开校园，离开这座象牙塔步入社会。虽然有许许多多的不舍，但是又对新的生活新的机会充满着期待。通过大学四年的学习，我找到了自己想要奋斗的目标，并且得到了很多人的帮助，有很多人是我应该感谢的。

感谢我的导师金政哲老师，我们都很亲切的称他为老金，他也是是我们团队的导师，他一直充当着亦师亦友的角色。他会放开手让我们在软件开发实践中学习，当我们在项目开发中遇到问题、遇到困难，会给我们提供思路，这不仅锻炼了我们独立思考解决问题的能力，也避免了因缺乏经验而产生的低级错误，让我们少走弯路，把更多的精力放在学习新知识上。

感谢大学四年一直陪伴我的宿友，我们一起在宿舍里学习，一起在宿舍里讨论不同方向的专业知识，让我的视野逐渐开阔，让我可以从多个不同的角度思考问题，也让我有了一个充实美好的大学时光。

感谢CST的小伙伴们，在实验室中大家合作实现了多个软件项目，在协作中学到了团队合作的方法，以及共同营造了一个学习的氛围，在其中每个人都在不断进步，进而促使自己始终向上，以跟上大家的步伐。

感谢我的父母，是他们为我创造了这么好的学习条件，不怕辛苦的把我让大成人，让我接受高等教育。

就要离开校园了，心中有万分不舍，但我对新的生活新的机会充满期待，我相信我不会辜负父母的期待，不会让母校失望，不会让老师失望。最后希望母校会更加辉煌。