|  |  |
| --- | --- |
| 学院标志(改) 副本 | 惠州学院  HUIZHOU UNIVERSITY |

毕 业 论 文（设 计）

**中文题目：** ­­­­­­­­**基于SpringBoot的“效率空间”个人网站**

**设计与实现**

**英文题目：** "Efficiency Space" Personal Website Design

and Implementation Based on SpringBoot

**姓 名 温志浩**

**学 号 1614080902145**

**专业班级 16软件工程1班**

**指导教师 蒋辉教授**

**提交日期 2020年xx月xx日**

教务处制

**学术诚信声明**

本人所呈交的毕业论文，是在指导教师的指导下独立完成。研究工作所取得的成果、数据、图片资料均真实可靠。除文中已注明引用的内容外，不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究做出重要贡献的个人和集体，均已在文中以明确的方式标明。本人完全意识到本声明的法律结果由本人承担。

本人签名： 日期：

**基于SpringBoot的“效率空间”个人网站**

**设计与实现**

作者姓名：温志浩 专业班级：16软件工程1班 指导老师：蒋辉教授

**摘要**

随着网络及移动端平台的飞速发展，市面上出现的软件种类日益增长，面对数目繁多的软件之海，人们开始追求功能齐全的一站式服务。

本论文阐述的“效率空间”网站，采用Java后端SpringBoot框架以及前端的Vue.js框架进行前后端分离式开发，该网站具备以下功能模块：用户功能模块、日记功能模块、备忘录功能模块、项目管理功能模块、后台管理功能模块。网站部署于阿里云服务器中，通过Nginx服务器转发请求进行访问。

网站提供人们最常用的服务，有写日记、添加备忘录、简单的项目进度管理等各种实用功能，使用户省去下载和挑选软件客户端时间的同时，也能享受功能齐全的一站式服务。

**关键词：**SpringBoot Vue.js 网站 Nginx 前后端分离

"Efficiency Space" personal website design and implementation based on SpringBoot

Author: Wen Zhihao Specialty: software engineering Tutor: professor Jiang Hui

**Abstract**

With the rapid development of network and mobile platform, there are more and more kinds of software on the market. Facing a large number of software, people begin to pursue one-stop service with complete functions.

The "Efficiency Space" website described in this paper adopts the Java back-end SpringBoot framework and the front-end Vue.js framework to develop the front-end and back-end separately. The website has the following functional modules: user function module, diary function module, memo function module, project management function module and back-end management function module. The website is deployed in Alibaba cloud server and accessed by forwarding requests through Nginx server.

The website provides people with the most commonly used services, including diary writing, adding memos, simple project progress management and other practical functions, so that users can save the time to download and select software clients, but also enjoy a full-featured one-stop service.

**Keywords:** SpringBoot Vue.js Website Nginx Front and Rear End Separation

目录

[1 绪论 1](#_Toc6129)

[1.1 选题的背景和意义 1](#_Toc22687)

[1.2 课题研究的主要内容 1](#_Toc6853)

[2 关键技术介绍 1](#_Toc22064)

[2.1 SpringBoot框架概述 1](#_Toc13063)

[2.2 Vue.js框架概述 2](#_Toc28972)

[2.3 前后端分离开发模式概述 2](#_Toc29427)

[2.4 开发环境 3](#_Toc19054)

[3 网站需求分析 4](#_Toc2443)

[3.1 网站技术需求分析 4](#_Toc23057)

[3.2 网站功能需求分析 4](#_Toc19785)

[3.3 网站性能需求分析 6](#_Toc23693)

[3.4 用例模型 6](#_Toc4159)

[4 网站概要设计 7](#_Toc251)

[4.1 网站功能设计 7](#_Toc30218)

[4.1.1 用户模块功能设计 8](#_Toc28782)

[4.1.2 日记模块功能设计 8](#_Toc9470)

[4.1.3 备忘录模块功能设计 8](#_Toc19146)

[4.1.4 项目管理模块功能设计 9](#_Toc17120)

[4.1.5 后台管理模块功能设计 9](#_Toc16990)

[4.2 接口设计 9](#_Toc24600)

[4.2.1 接口安全性 9](#_Toc30712)

[4.2.2 接口隔离性 10](#_Toc19485)

[4.2.3 接口可读性 10](#_Toc28563)

[4.3 数据库设计 11](#_Toc3053)

[4.1.1 数据库分析 11](#_Toc12124)

[4.1.2 数据库概念设计 11](#_Toc30086)

[4.1.3 数据库表 13](#_Toc982)

[5 网站详细设计与实现 14](#_Toc8178)

[5.1 前端项目搭建 15](#_Toc31269)

[5.2 后端项目搭建 17](#_Toc5348)

[5.3 网站的实现 18](#_Toc27091)

[5.3.1 用户模块 18](#_Toc22514)

[5.3.2 日记模块 26](#_Toc29059)

[5.3.3 备忘录模块 31](#_Toc14578)

[5.3.4 项目管理模块 34](#_Toc9839)

[5.3.5 后台管理模块 37](#_Toc21005)

[5.4 网站的部署 39](#_Toc29860)

[6 结论与展望 41](#_Toc22393)

[参考文献 41](#_Toc24155)

[致谢 42](#_Toc19968)

1 绪论

1.1 选题的背景和意义

随着互联网的飞速发展，市面上的软件越来越多。其中，日常工具类的软件已经愈发成熟，但大部分软件只提供一种类型的服务。随着C/S模式的没落，B/S模式的崛起，慢慢的人们更加愿意通过浏览器快速的访问一个网站来获取软件服务，比如开发者需要转换一些数据格式，往往只需要打开网页，找到对应的网站便可以完成数据格式转换的需求，此类网站通常还包括其他有用的功能，可谓是非常方便。

然而就目前的现状来看，工具类网站提供的服务大都是为开发者之类的人员准备的，类似于写日记、备忘录之类的日常类工具却鲜有网站进行整合开发。但是随着互联网的继续发展，网民的比例继续上升，普通的用户对于这样的一站式服务工具必将有所求，因此开发一个包含一些日常工具的个人网站，方便人们的生活，一旦获取用户的信任，不断扩展网站，壮大用户群，那网站的发展前景就相当可观了。

1.2 课题研究的主要内容

完成一个个人网站，包括用户模块、日记模块、备忘录模块、项目管理模块、后台管理模块。不同的用户通过邮箱注册自己的独立账户，根据独立账户进行登录。登录成功的用户进入首页，首页导航栏包括日记模块、备忘录模块、项目管理模块，用户可以选择相应的模块使用相应的功能。

2 关键技术介绍

2.1 SpringBoot框架概述

SpringBoot框架是是由Pivotal团队提供的全新框架，其设计目的是简化Spring应用的初始搭建以及开发过程，将开发人员从繁重的XML配置中解放出来，从而有更多的时间专注于业务而不是配置。其具有以下特点：

1. 创建独立的Spring应用程序；
2. 直接内嵌Tomcat、Jetty或Undertow服务器，无需部署WAR文件；
3. 提供完善的“start”依赖项，以简化构建配置；
4. 尽可能自动配置Spring和第三方库；
5. 提供可用于生产的功能，例如指标，运行状况检查和外部化配置。

基于上述优点，SpringBoot在推出后迅速走红，成为快速应用开发领域的领导者，是快速建立后端应用的不二之选。

2.2 Vue.js框架概述

Vue.js是一套构建用户界面的渐进式框架。它是以数据驱动和组件化的思想构建的，采用了自底向上增量开发的设计。通过提供简洁易懂的api，使开发者能快速上手并使用Vue.js构建前端页面。

Vue.js的数据传递遵循MVVM模式，本质上是MVC模式的改进版。MVVM模式的交互示意图如下，其通过ViewModel实现对View和Model层面的双向监听，从而实现数据的双向绑定，只要View和Model层面有任何一方的变化，另外一方马上就会进行相应的更新。

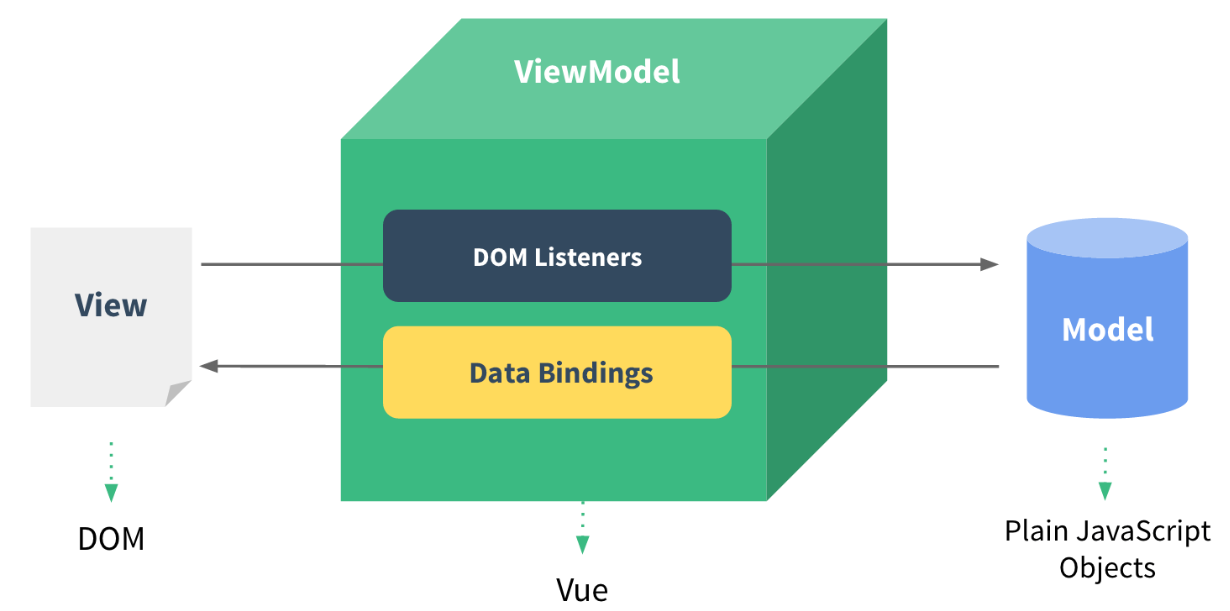


图2-1 MVVM模式交互示意图

2.3 前后端分离开发模式概述

在早期的Java Web项目中，传统的开发模式是前端代码（html、css、js）写在JSP中，甚至有些JSP还会嵌入Java代码。当用户访问网站时，页面数据是由Servlet容器将JSP编译成Servlet，然后将JSP中的html、css、js代码输出到浏览器，这个过程步骤繁琐，效率低下，页面的响应速度慢，而且前后端代码混合编写，后期维护起来十分困难。

为了弥补传统开发模式的不足，前后端分离开发的概念被提出。采用该种模式时，前端只负责渲染html页面，用户所看到的效果全由前端页面控制，前端所需要的数据，通过Ajax访问后端api获得，后端只需要专注于自己的业务代码即可，示意图如下。在这种情况下，前后端的代码分别开发，前后端开发人员通过api文档对接接口，职责分离，分工明确。

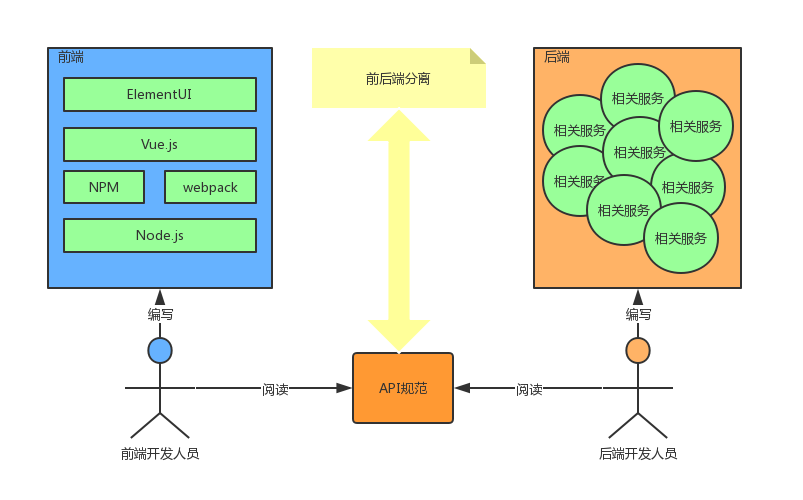


图2-2 前后端分离开发模式

2.4 开发环境

开发一个稳定的网站，需要我们搭建一套完善的开发环境，以下将介绍笔者开发时使用的环境：

（1）本地环境

操作系统：Windows 10专业版

服务器：Nginx 1.10.2

JDK：1.8

开发工具：IntelliJ IDEA

数据库：MySQL 5.7

MySQL图形化管理软件：Navicat Premium 11.1.13

浏览器：Google Chrome 80

（2）服务器环境

操作系统：CentOS 7.6 64位

服务器：Nginx 1.10.2

JDK：1.8

数据库：MySQL 5.7

3 网站需求分析

需求分析作为软件开发的第一步，在此需要充分分析网站的所有功能，前后端的联动情况，服务的运转流程，任何一个环节都不能疏忽。充分的分析是建设一个好网站的前提，能否满足用户的功能需求？日后如何扩展系统？都是需要先在这一步考虑到的，所以本章节就是要介绍“效率空间”网站是怎么分析设计的。

3.1 网站技术需求分析

随着时代的发展，各种新奇的互联网技术层出不穷，而且技术的迭代也非常的迅速，可能在短短的两三年间，一种新的技术就可以发展壮大，并获得大批开发者的追捧，社区活跃，人员交流频繁。为了使开发出来的网站是健壮的，也为了应对网站日益增长的用户需求，对于开发者来说，想要建设一个好网站，就得综合考虑，要采用何种方式去实现功能，使之开发过程更加省时省力，扩展系统更加方便快捷，用户更加舒适自然。

因此在综合分析之后，笔者选用了时下最流行的前后端框架，使之开发过程更加轻松，这也符合当下的软件开发思想，即先选成熟稳定的技术进行开发，而不是新生或偏门的技术，许多的公司开发实践已经证明了该想法的正确性。毕竟我们个人开发者在选择了成熟的技术开发时会比较有底气，遇到开发难题可以在网上或者社区寻找解决方案，而不是像一些冷门技术，出了问题，找不到解决方法，导致软件开发因此而夭折。

3.2 网站功能需求分析

“效率空间”网站是为提高人们效率而建，在日后将持续更新，提供新的工具供用户使用，在网站建设初期，选取日记、备忘录、项目管理等功能进行实现，实现效果要符合以下条件：

（1）界面整洁，交互简单。

因为网站的初衷是提高效率，因此在界面交互上应该更加简单，而不该为了炫技而增加一大堆无关的特效和操作，当然，这也不意味着界面可以随便糊弄完事，用户大部分肯定还是“看脸的”，过于难看的系统是留不住用户的。所以笔者在系统开发时选用了较为流行的UI组件库ElementUI作为界面开发组件，并且对页面的排版有一定的规范，使之看起来简而不陋。

（2）其次，用户可以迅速找到自己想要的功能。

考虑到用户第一次进站什么都不了解的情况下，能快速了解本站提供的功能，这里笔者使用顶部导航栏进行功能导航，导航栏显示对应功能的名称，使用户看上去一目了然。

（3）可以快速搜索到自己想要的日记。

随着用户保存的日记增多，日记的搜索功能也是必不可少的，该网站在顶部导航栏处显示一个搜索框，以供用户时刻都可以查询想要的日记内容，这里采用模糊匹配的方式，用户输入关键字，网站显示标题或内容包含关键字的日记，能让用户快速查找到自己想要查看的日记。

（4）方便的日记编辑器，图片上传功能。

编辑器可以说是日记中最重要的功能，直接决定了用户写日记的操作体验，笔者在这采用了当前比较火热的markdown编辑器，可以很方便的进行页面的排版，并且拥有实时渲染的功能，让用户有所见即所得的体验。另外为了用户插入图片方便，笔者另外申请了七牛云服务器专门作为图片上传服务器，实现用户的图片上传功能。

（5）多样化的备忘录。

考虑到用户的备忘需求多种多样，“效率空间”网站的备忘录也采用多样化的实现方式，分为：提醒、生日、纪念日三种类别，提醒类的备忘录可以根据设定的时间实时提醒，生日和纪念日则可以设置提前时间提前发送提醒邮箱。另外，三种不同的备忘录也有不同的邮件模板。

（6）简单方便的项目管理功能。

考虑到完整的大型项目管理是十分复杂的，该网站实现的是适用于个人的小型项目管理功能，用户可以很轻松的创建和结束项目，项目内部的管理则采用小卡片的形式，提供一种可拖拽的前端交互技术，实现高效方便的项目进度变化。

（7）便捷的管理功能。

最后，在完成用户功能的前提下，有一个好的管理后台对网站管理员来说也是至关重要的，比如网站所拥有的用户数，用户管理，用户分析等功能，能让管理员更加清楚的了解网站的运行情况。

3.3 网站性能需求分析

本网站是基于B/S模式的，用户只需要有浏览器进行页面展示即可，大部分业务处理都是由后端服务器进行处理，由于网站业务并没有涉及非常复杂的计算，服务器的CPU是完全可以满足的。

其次在网站访问量大的时候，产生的瞬时流量可能会比较大，由于笔者目前只有一个云服务器，可能会出现一些网络卡顿的情况，但由于使用了Nginx反向代理服务器，可以很方便的增加服务器进行分流扩展，以满足并发量大的情况。

综合考虑本站的实际情况来看，网站的性能需求对服务器要求并不高，是可以接受的。在超出服务器处理能力的情况下，仅需要增加运行后端程序的服务器即可。

3.4 用例模型

通过前面的分析可以确定本网站的使用者有以下两种：普通用户、网站管理员。下面介绍这两种角色的权限区别。

1. 普通用户

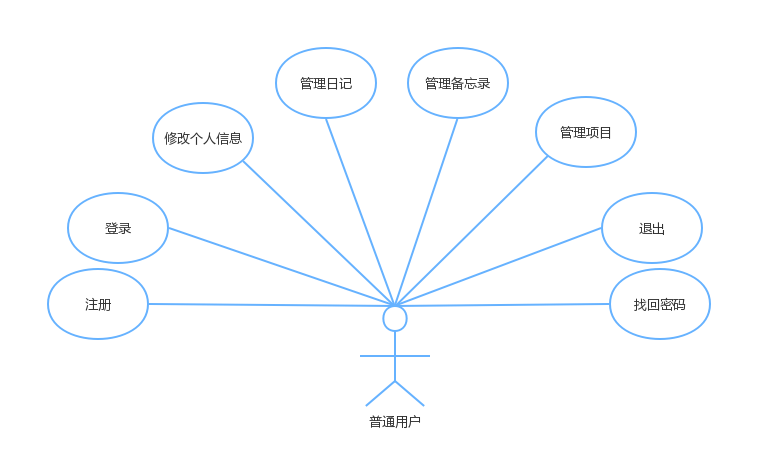


图3-1 普通用户主要功能

（2）网站管理员

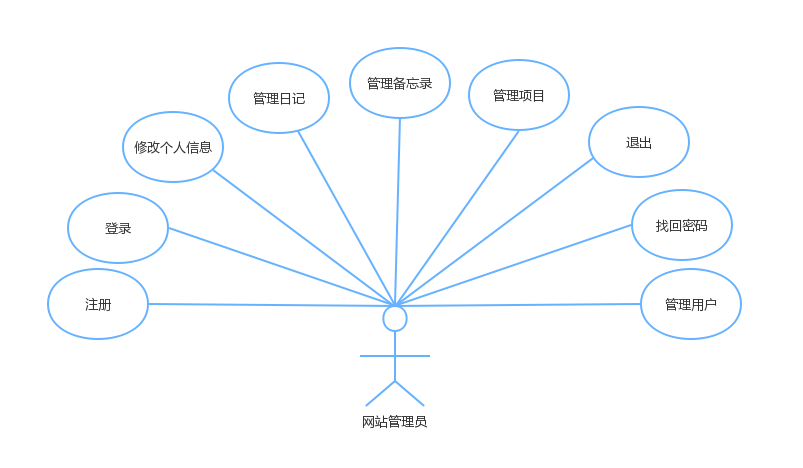


图3-2 网站管理员主要功能

4 网站概要设计

4.1 网站功能设计

根据网站功能需求分析与用例模型，网站总体实现的功能包含五个模块：用户模块、日记模块、备忘录模块、项目管理模块、后台管理模块，网站的总体功能结构图如图4-1所示。

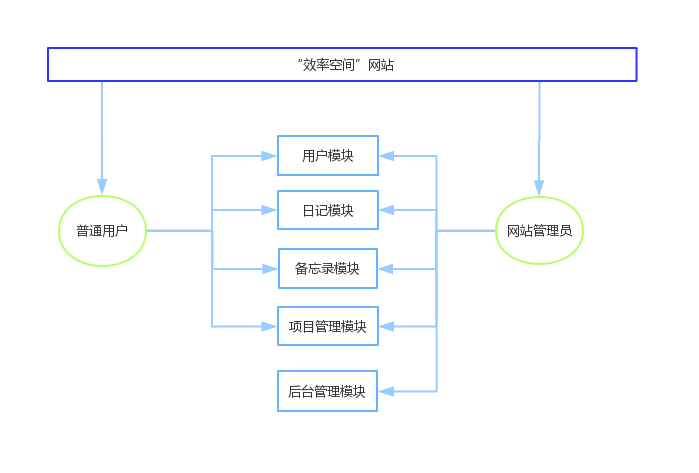


图4-1 网站总体功能结构图

以下笔者将根据各个功能模块进行具体分析。

4.1.1 用户模块功能设计

用户模块是网站的基础模块，其应该包含用户的登录、注册、找回密码、查看信息、修改信息、退出登录等功能，结构如图4-2所示。

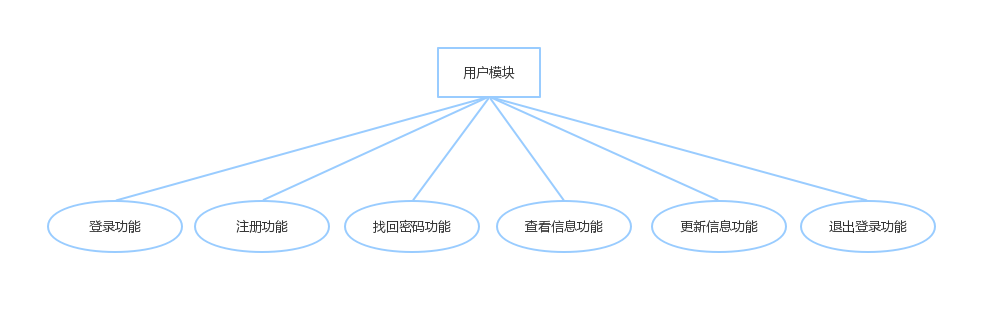


图4-2 用户模块功能结构

4.1.2 日记模块功能设计

日记模块为用户提供写日记服务，其应该包含日记的新增、删除、修改、搜索、查看和归档，用户写日记的过程难免会需要插入图片，因此还需要上传图片的功能，最后结构如图4-3所示。

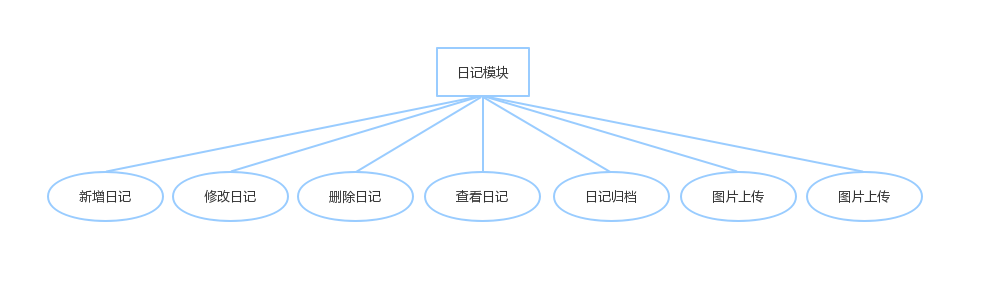


图4-3 日记模块功能结构

4.1.3 备忘录模块功能设计

备忘录模块为用户提供记备忘录服务，其包含三种类型：提醒、生日、纪念日，每种类型由对应的新增、删除功能，结构如图4-4所示。

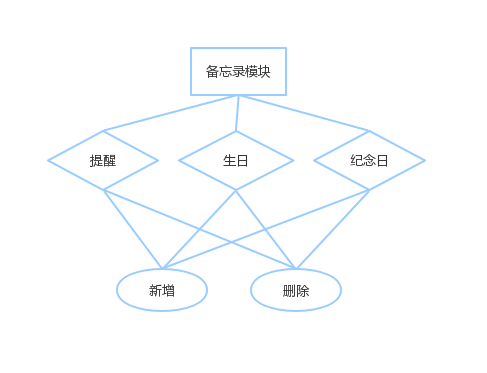


图4-4 备忘录模块功能结构

4.1.4 项目管理模块功能设计

项目管理模块为用户提供项目管理服务，其包含项目的新增、删除、修改、标记完成等功能，结构如图4-5所示。

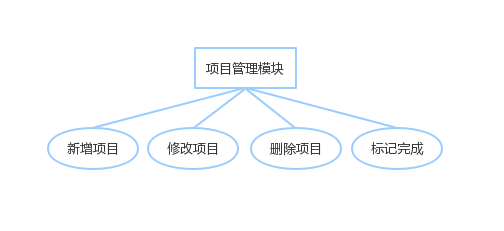


图4-5 项目管理模块功能结构

4.1.5 后台管理模块功能设计

后台管理模块为管理员提供用户管理及用户分析服务，其中用户管理包含用户信息的查看和删除功能，用户分析包含用户男女比例分析功能，结构如图4-6所示。

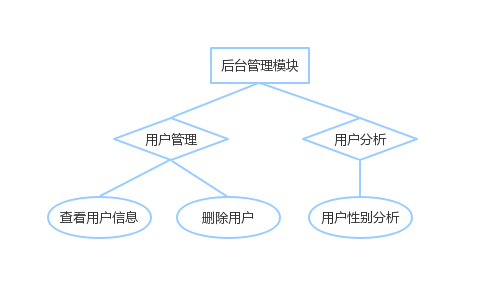


图4-6 后台管理模块功能结构

4.2 接口设计

接口作为连接前后端数据的桥梁，起着举足轻重的作用。接口定义设计的好坏很大程度上直接决定了一个系统的稳定性和扩展性，一个好的接口设计也能使前后端分离更加彻底，使开发人员仅需关注规范的接口文档，前后端的实现互不干扰。以下笔者将就接口的安全性、隔离性和可读性三个方面介绍网站开发时的接口设计过程。

4.2.1 接口安全性

接口的设计是绕不开安全二字的，任何系统脱离了安全，想必都不会有用户愿意买单。因此为了达到尽可能的安全，我们需要尽可能的增加攻击网站的难度，一般常见的手段有过期验证、签名验证、避免SQL注入、限流等。

本网站的接口带有权限认证，对应的管理员接口需要有管理员的权限才会对其开放。用户在登录后服务器会返回token给用户，token标识了用户的登录状态，若浏览器没有发送的请求头中没有夹带token，那么网站会跳转到登录页面要求用户登录，此外在用户进行数据库的操作时，服务器会使用token验证用户身份，避免横向越权。在找回密码的接口中，为了避免用户重复调用邮箱验证码接口，做了多少分钟后才能再次调用的设定。

4.2.2 接口隔离性

接口的隔离性涉及到接口的代码设计，需要遵循“高内聚，低耦合”的代码设计原则。

低耦合，也就是一个系统的不同模块之间，尽可能的减少互相依赖，使各个模块相互独立，即一个模块完成任务不需要依赖其他模块。

高内聚，也就是一个模块内的所有元素彼此结合的紧密程度越高越好。宏观体现的结果就是一个模块内的代码相关性都很强，且整体只负责一项任务，也就是设计原则中的单一责任原则。

为了好的接口隔离性，需要后端开发者在业务层进行良好的编码设计，尽可能使各个业务模块之间互不依赖，能独立完成自身负责的业务。

4.2.3 接口可读性

本系统采用的是前后端开发模式，前后端的开发都依赖于接口，因此接口设计必需结构清晰、符合规范、易于理解、便于扩展，使开发人员一眼便能看懂接口。为此笔者采用了RESTful风格的接口设计，RESTful架构目前已经成为Web API的标准，许多公司使用的都是这种风格的接口，其主要特点如下：

（1）每一个URI代表一种资源；

（2）客户端和服务器之间，传递这种资源的某种表现层；

（3）客户端通过四个HTTP动词，对服务器端资源进行操作，实现"表现层状态转化"。

此外，除了接口请求要遵循RESTful规范，响应的数据也要进行规范化的处理，下面是本系统统一的响应格式及响应状态说明：

响应格式：

{

“code” : 0,

“msg” : ””,

“data” : {}

}

响应状态说明：

|  |  |
| --- | --- |
| 响应码 | 说明 |
| 200 | 操作成功 |
| 201 | 操作失败 |
| 403 | 需要登录 |

4.3 数据库设计

4.1.1 数据库分析

本系统主要是供个人用户使用，主要操作以查询为主，因此即使在高并发的情况下，对数据库的压力也不会太大，所以综合考虑，笔者选择了目前使用最为广泛的开源数据库MySQL作为系统的基础数据库。

本网站采用的是MySQL-5.7版本的数据库，采用utf8mb4编码，分为5个表，分别为用户信息表（myspace\_user）、日记信息表（myspace\_diary）、备忘录信息表（myspace\_memo）、项目信息表（myspace\_project）和项目详情表（myspace\_detail），数据库设计具体如下：

4.1.2 数据库概念设计

本系统涉及多个实体，具体有用户信息实体、日记信息实体、备忘录信息实体、项目信息实体、项目详情实体等。各个实体的E-R图如下所示：

1. 用户信息实体E-R图（图4-7）。

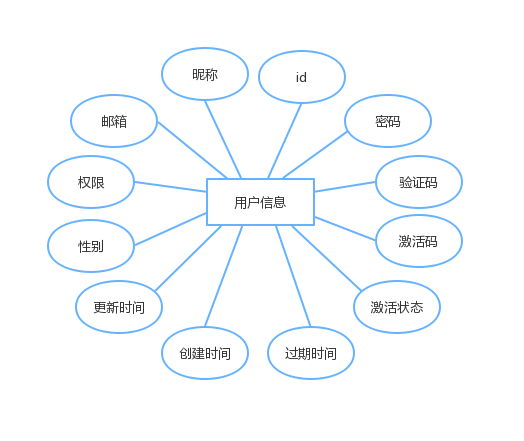


图4-7 用户信息实体E-R图

1. 日记信息实体E-R图（图4-8）。

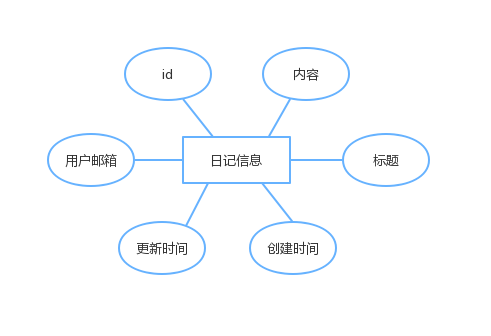


图4-8 日记信息实体E-R图

1. 备忘录信息实体E-R图（图4-9）。

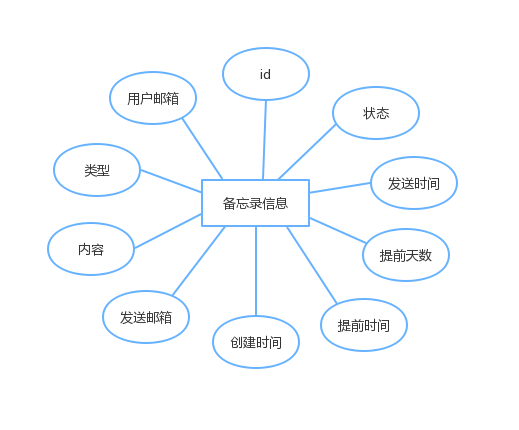


图4-9 备忘录信息实体E-R图

1. 项目信息实体E-R图（图4-10）。

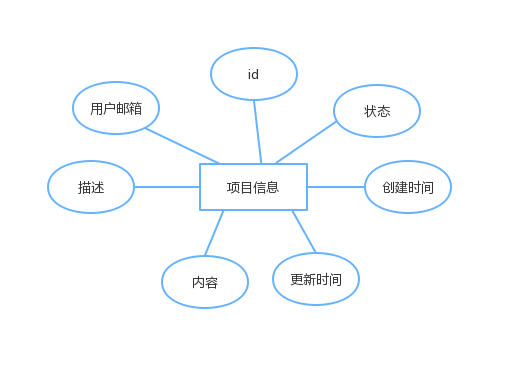


图4-10 项目信息实体E-R图

1. 项目详情实体E-R图（图4-11）。

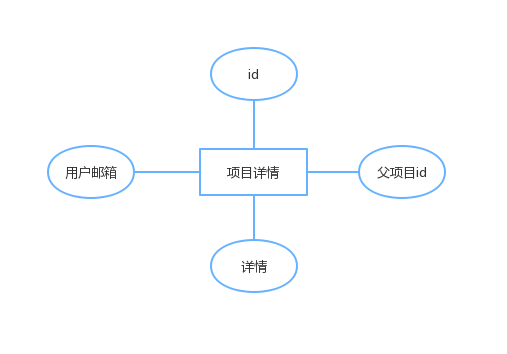


图4-11 项目详情实体E-R图

4.1.3 数据库表

结合实际情况分析后，设计出以下数据库表：

表4-1 myspace\_user用户信息表结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 允许空值 | 主外键关系 | 描述 |
| id | varchar | 50 | 否 | 主键 | 用户id |
| email | varchar | 30 | 否 |  | 用户邮箱 |
| password | varchar | 50 | 否 |  | 用户密码 |
| nickname | varchar | 40 | 否 |  | 用户昵称 |
| sex | int | 1 | 否 |  | 用户性别 |
| role | int | 1 | 否 |  | 用户角色 |
| active\_state | int | 1 | 否 |  | 激活状态 |
| active\_code | varchar | 80 | 是 |  | 激活码（注册） |
| verify\_code | varchar | 8 | 是 |  | 验证码（找回密码） |
| exp\_time | bigint | 13 | 是 |  | 失效时间 |
| create\_time | bigint | 13 | 否 |  | 进站时间 |
| update\_time | bigint | 13 | 否 |  | 最后登录时间 |

表4-2 myspace\_diary日记信息表结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 允许空值 | 主外键关系 | 描述 |
| id | int | 11 | 否 | 主键 | 日记id |
| user\_email | varchar | 30 | 否 |  | 日记所属用户邮箱 |
| diary\_title | varchar | 50 | 否 |  | 日记标题 |
| diary\_body | text |  | 是 |  | 日记内容 |
| create\_time | bigint | 13 | 否 |  | 创建时间 |
| update\_time | bigint | 13 | 否 |  | 更新时间 |

表4-3 myspace\_memo备忘录信息表结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 允许空值 | 主外键关系 | 描述 |
| id | int | 11 | 否 | 主键 | 备忘录id |
| user\_email | varchar | 30 | 否 |  | 备忘录所属用户邮箱 |
| memo\_type | int | 1 | 否 |  | 备忘录类型 |
| memo\_email | varchar | 50 | 否 |  | 接收提醒的邮箱 |
| pre\_time | int | 1 | 是 |  | 提前天数 |
| pre\_date | bigint | 13 | 是 |  | 提前发送时间 |
| memo\_date | bigint | 13 | 否 |  | 发送时间 |
| memo\_content | varchar | 255 | 否 |  | 备忘录内容 |
| memo\_state | int | 1 | 否 |  | 备忘录状态 |
| create\_time | bigint | 13 | 否 |  | 创建时间 |

表4-4 myspace\_project项目信息表结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 允许空值 | 主外键关系 | 描述 |
| id | Int | 11 | 否 | 主键 | 用户id |
| uesr\_email | varchar | 30 | 否 |  | 用户邮箱 |
| project\_body | varchar | 255 | 否 |  | 用户密码 |
| project\_desc | varchar | 255 | 是 |  | 用户昵称 |
| project\_state | int | 1 | 否 |  | 用户性别 |
| create\_time | bigint | 13 | 否 |  | 进站时间 |
| update\_time | bigint | 13 | 否 |  | 最后登录时间 |

表4-5 myspace\_detail项目详情表结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 允许空值 | 主外键关系 | 描述 |
| id | varchar | 11 | 否 | 主键 | 用户id |
| parent\_id | varchar | 11 | 否 |  | 父项目id |
| detail | text |  | 否 |  | 项目详情 |
| user\_email | varchar | 30 | 否 |  | 所属用户邮箱 |

5 网站详细设计与实现

5.1 前端项目搭建

前端采用的是Vue.js框架开发，搭建框架前需要先安装好Node.js环境（npm包管理器）、vue-cli脚手架构建工具（构建项目结构）、cnpm npm的淘宝镜像（加快下载速度）。安装好环境后，在cmd命令界面输入vue init webpack命令开始构建项目，效果如图5-1所示。

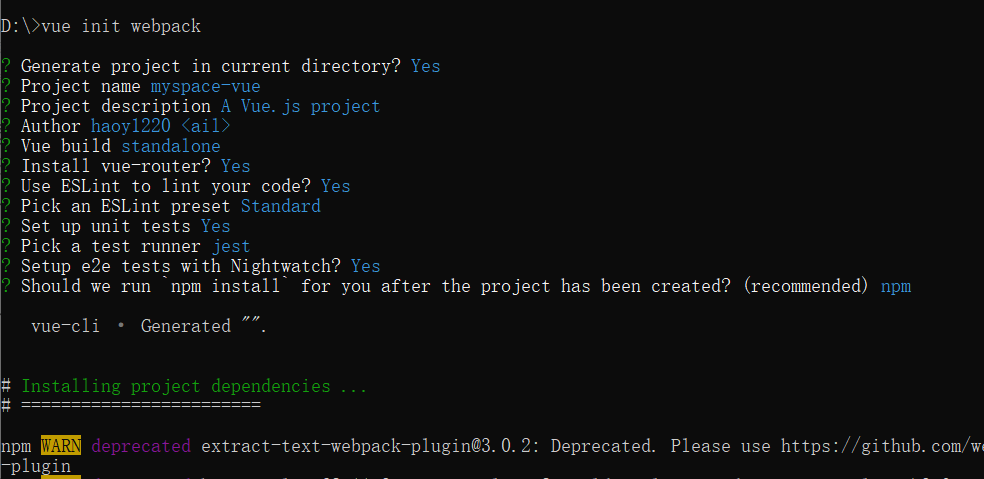


图5-1 vue项目构建过程

项目构建完成时会自动生成项目文件夹，文件夹结构如图5-2所示。

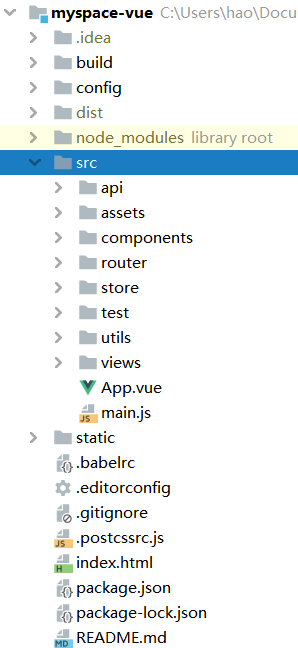


图5-2 vue项目结构

其中笔者就重要文件夹src讲解项目编码结构，其中包含的内容及作用如下：

Api文件夹：由笔者自行创建，存放后端请求接口的一系列方法；

Assets文件夹：用于存放资源文件，比如视图引用的相关图片；

Views文件夹：由笔者自行创建，存放主要的视图组件；

Components文件夹：存放除主要视图外的其他一切视图组件；

Router文件夹：存放控制路由跳转的配置文件；

Store文件夹：用于存储全局变量，记录从token中获取到的用户信息；

Utils文件夹：存放开发工具的文件夹，页面时间格式的转换、文件保存的方式、发送后端接口请求等都依赖于该部分的工具；

Test文件夹：存放开发过程中的测试页面；

App.vue：根组件，是其他一切组件的起源；

Main.js：项目的入口文件，所有页面都会加载该文件。

至此，前端部分的项目搭建完毕，具体的编码实现部分将在实现阶段详细解说。

5.2 后端项目搭建

后端采用的是SpringBoot框架，事先安装好JDK环境，利用IDEA编辑器的新建向导功能，可以很快的构建SpringBoot项目，构建完毕的项目结构如图5-3所示。

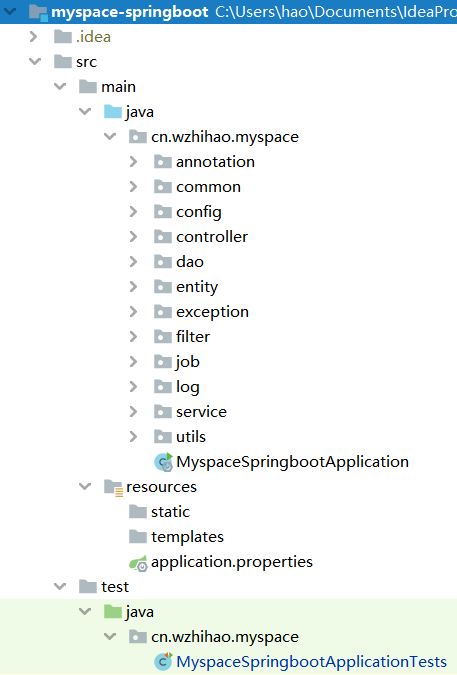


图5-3 后端项目结构

下面笔者将就一些重要的文件夹讲解其作用：

①java文件夹：存放后端服务的主要程序源代码。

Annotation包：存放自定义的注解；

Common包：存放通用的类，比如统一的接口响应格式类、全局静态变 量等；

Config包：存放自定义配置类，笔者在其中配置了druid数据源以及全 局拦截器；

Controller包：存放MVC模式的控制器类；

Service包：存放MVC模式的服务类接口及其实现；

Dao包：存放MVC模式的数据库交互类；

Entity包：存放实体类；

Filter包：存放过滤器，全部请求将会被全局拦截器拦截到此处进行token 验证；

Job包：存放定时任务类；

Utils包：则是存放各种工具类的包。

②resources文件夹：

Static文件夹：存放css、js等资源文件；

Template文件夹：存放html页面类型的资源文件；

Application.properties文件：SpringBoot的核心配置文件。

③test文件夹：存放开发过程中的一些单元测试类。

至此，后端项目搭建完成，接下来将在实现章节介绍重要编码。

5.3 网站的实现

本节介绍网站各个功能模块的实现结果以及涉及到的部分重要编码。

5.3.1 用户模块

用户模块包含用户的登录、注册、找回密码、查看信息、修改信息、退出登录等功能，应包含登录页面、注册页面、激活页面、找回密码页面、个人信息页面。下面将分别对这些页面进行讲解。

①登录页面

登录页面是用户使用网站的入口，用户输入邮箱和密码即可登录，实现效果图如图5-4。其中前端完成页面展示、数据格式校验工作，后端负责连接服务器验证邮箱和密码的正确性。



图5-4 登录页面

登录接口代码如下：

User user = new User();

user.setEmail(email);

user.setPassword(MD5Util.MD5EncodeUtf8(password));

user = userMapper.selectOne(user);

if (user == null) {

return Result.Error(Const.StatusCode.ERROR, "邮箱不存在或密码错误");

} else if (user.getActiveState() == Const.Active.NO) {

return Result.Error(Const.StatusCode.ERROR, "邮箱尚未激活，请前往邮箱激活");

} else {

user.setUpdateTime(Calendar.getInstance().getTimeInMillis());

String token = JwtTokenUtil.createToken(user);

userMapper.updateByPrimaryKeySelective(user);

user.setPassword(StringUtils.EMPTY);

Map<String, Object> dataMap = new HashMap<>();

dataMap.put("token", token);

dataMap.put("user", user);

return Result.SuccessByData("登录成功", dataMap);

}

另外，保存到数据库的密码为了安全起见，笔者使用MD5加密的方式存储密文密码到数据库，MD5加密工具类代码将放在附录部分展出，后续的各种重要工具类也将如此处理，此部分只展示核心业务逻辑代码。

②注册页面

注册页面提供给不能登录的用户注册邮箱使用，效果图如图5-5。其中前端完成页面展示、邮箱和昵称格式规范校验、密码一致校验等工作，用户提交信息后由后端将数据写入数据库，并发送激活邮件，如图5-5。



图5-5 注册界面



图5-6 注册激活邮箱

注册接口代码如下：

String id = UUID.randomUUID().toString();

user.setId(id);

user.setPassword(MD5Util.MD5EncodeUtf8(user.getPassword()));

user.setRole(Const.Role.ROLE\_CUSTOMER);

user.setActiveState(Const.Active.NO);

long time = Calendar.getInstance().getTimeInMillis();

user.setCreateTime(time);

user.setUpdateTime(time);

String activeCode = MD5Util.MD5EncodeUtf8(user.getEmail() + user.getPassword() + user.getNickname() + user.getId() + time);

user.setActiveCode(activeCode);

user.setExpTime(time + Const.ACTIVE\_EXP\_TIME);

int res = userMapper.insertSelective(user);

if (res == 0) {

return Result.Error(Const.StatusCode.ERROR, "注册失败");

} else {

SimpleMailMessage simpleMailMessage = EmailUtil.sendActiveEmail(user.getId(), user.getActiveCode(), user.getEmail(), Const.ADMIN\_EMAIL);

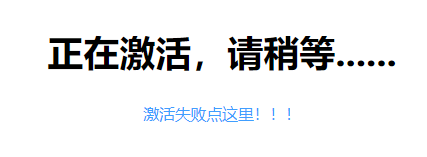
javaMailSender.send(simpleMailMessage);

return Result.Success("注册成功,请前往邮箱激活");

}

③激活页面

用户点击激活邮箱链接便会跳转到激活页面，成功和失败分别有两种显示，如图5-7所示。



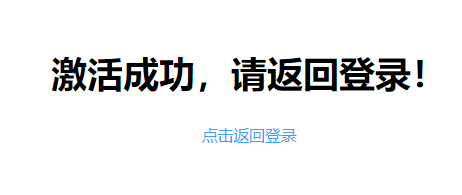


图5-7 激活页面

激活接口代码如下：

User user = new User();

user.setId(id);

user.setActiveCode(activeCode);

user = userMapper.selectOne(user);

if (user == null || user.getActiveState() == Const.Active.YES) {

return Result.Error(Const.StatusCode.ERROR, "邮箱已激活或激活码无效");

} else {

long currTime = Calendar.getInstance().getTimeInMillis();

if (currTime > user.getExpTime()) {

userMapper.deleteByPrimaryKey(user);

return Result.Error(Const.StatusCode.ERROR, "激活码已过期，请重新注册！！！");

} else {

user.setActiveCode("");

user.setExpTime(0L);

user.setActiveState(Const.Active.YES);

user.setUpdateTime(currTime);

userMapper.updateByPrimaryKey(user);

return Result.Success("激活成功，请前往登录！！！");

}

}

④找回密码页面

当用户忘记密码，就需要进入找回密码界面重置密码，效果如图5-8。用户输入邮箱，点击获取邮箱验证码即可从邮箱找到验证码，如图5-9，最终提交信息由后端更新数据库。



图5-8 找回密码页面



图5-9 邮箱验证码

找回密码接口代码如下：

User user = new User();

user.setEmail(email);

user.setVerifyCode(verifyCode);

user = userMapper.selectOne(user);

if (user == null) {

return Result.Error(Const.StatusCode.ERROR, "邮箱或验证码错误");

} else {

long currTime = Calendar.getInstance().getTimeInMillis();

if (currTime > user.getExpTime()) {

user.setExpTime(0L);

user.setVerifyCode("");

userMapper.updateByPrimaryKeySelective(user);

return Result.Error(Const.StatusCode.ERROR, "验证码已过期");

} else {

user.setPassword(MD5Util.MD5EncodeUtf8(newPassword));

user.setVerifyCode("");

user.setExpTime(0L);

userMapper.updateByPrimaryKeySelective(user);

return Result.Success("密码重置成功，请重新登录！");

}

}

验证码接口代码如下：

Result<String> result = checkValid(email);

if (result.getCode() == Const.StatusCode.ERROR) {

User user = new User();

user.setEmail(email);

user = userMapper.selectOne(user);

String verifyCode = String.valueOf(new Random().nextInt(899999) + 100000);

long time = Calendar.getInstance().getTimeInMillis();

user.setExpTime(time + Const.VERIFY\_EXP\_TIME);

user.setUpdateTime(time);

user.setVerifyCode(verifyCode);

userMapper.updateByPrimaryKeySelective(user);

SimpleMailMessage simpleMailMessage = EmailUtil.sendVerifyEmail(user.getVerifyCode(), user.getEmail(), Const.ADMIN\_EMAIL);

javaMailSender.send(simpleMailMessage);

return Result.Success("验证码已发送至邮箱，请前往邮箱获取！");

} else {

return Result.Error(Const.StatusCode.ERROR, "邮箱不存在或未激活");

}

⑤个人信息页面

用户登录后会进入首页（图5-10），首页由导航栏及一页日历组成，点开界面右上角的菜单，可以调出个人信息页面，如图5-11所示，用户可以修改信息并提交，由后端服务器更新数据库。



图5-10 首页



图5-11 个人信息页面

获取个人信息接口代码如下：

User user = new User();

user.setEmail(email);

user = userMapper.selectOne(user);

if (user == null) {

return Result.Error(Const.StatusCode.ERROR, "找不到当前用户信息");

}

user.setPassword(StringUtils.EMPTY);

return Result.SuccessByData("成功返回", user);

更新个人信息接口代码如下：

User currUser = userMapper.selectOneByEmail(email);

currUser.setNickname(nickname);

int res = userMapper.updateByPrimaryKeySelective(currUser);

if (res == 0) {

return Result.Error(Const.StatusCode.ERROR, "更新失败");

} else {

currUser.setPassword(StringUtils.EMPTY);

return Result.SuccessByData("更新个人信息成功", currUser);

}

5.3.2 日记模块

日记模块包含日记的新增、删除、修改、搜索、查看和归档、图片上传等功能，应包含日记列表页面、日记编辑页面、日记详情页面、日记搜索页面。

①日记列表页面

日记列表页面用于展示用户所有的日记记录、归档信息，对日记的新增、修改、删除、搜索操作都可以在该页面完成，最终效果如图5-12。

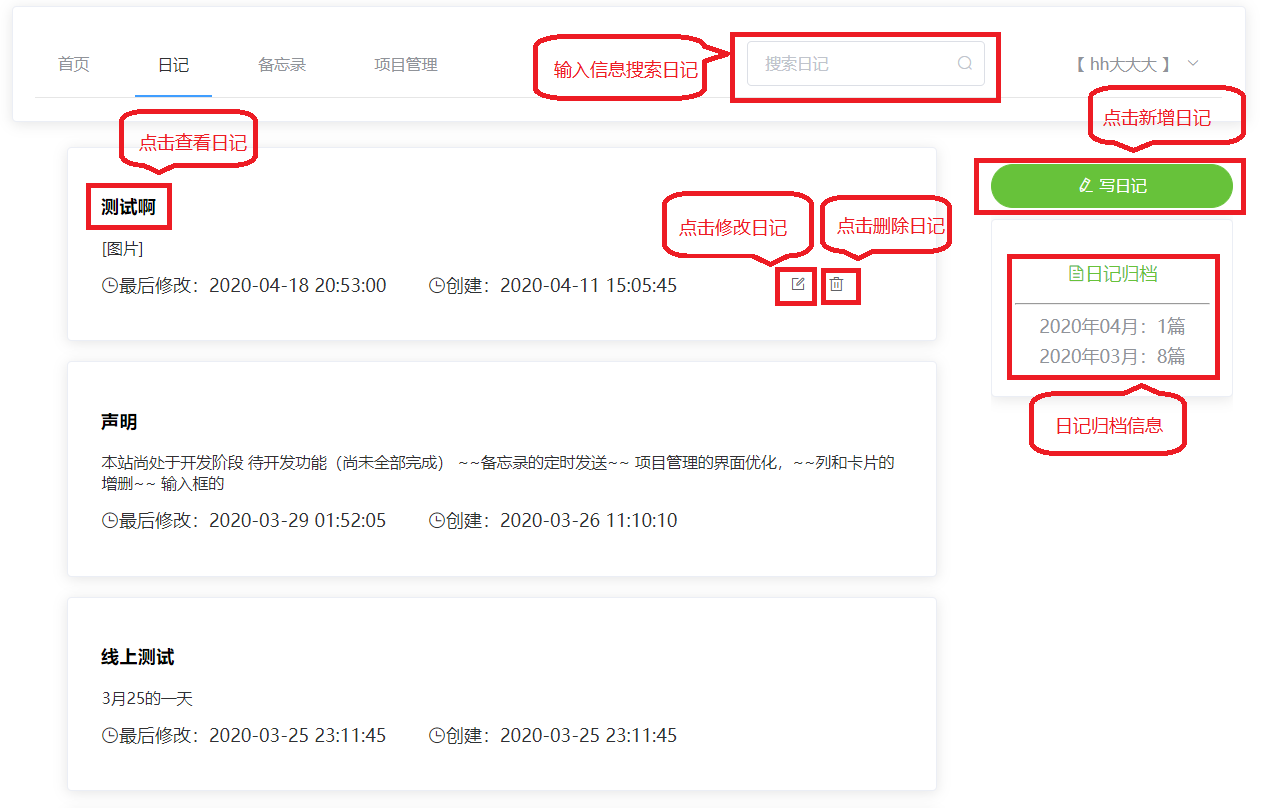


图5-12 日记列表页面

获取日记列表接口如下：

PageHelper.startPage(pageNum, pageSize);

List<Diary> diaryList = diaryMapper.selectByEmail(JwtTokenUtil.getEmailFromRequest(request));

for (Diary diary : diaryList) {

String body = diary.getDiaryBody();

if (body.length() > Const.MAX\_BODY\_CHAR\_COUNT) {

diary.setDiaryBody(body.substring(0, Const.MAX\_BODY\_CHAR\_COUNT));

}

}

PageInfo<Diary> pageInfo = new PageInfo<>(diaryList);

return Result.SuccessByData("获取分页列表成功", pageInfo);

获取归档信息接口代码如下：

List<Map<String, Integer>> resMap = diaryMapper.

selectArchiveByEmail(JwtTokenUtil.getEmailFromRequest(request));

return Result.SuccessByData("查询归档数据成功", resMap);

②日记编辑页面

用户新增或修改日记都可以调出该界面，不同之处在于修改时该界面会自动填充日记原有的信息，另外在用户编辑日记的时候可以上传图片（见图5-13），后端服务器会自动返回图片所在的地址，最终界面效果如图5-14所示。

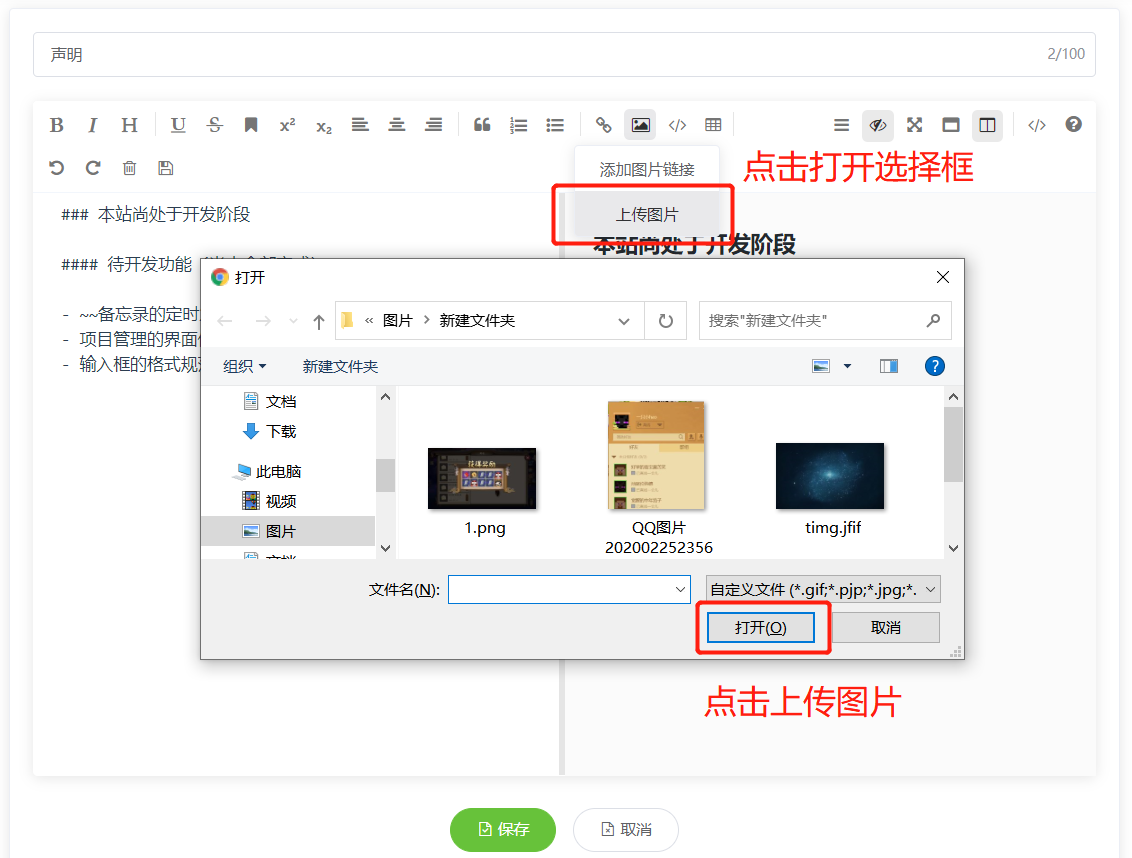


图5-13 图片上传

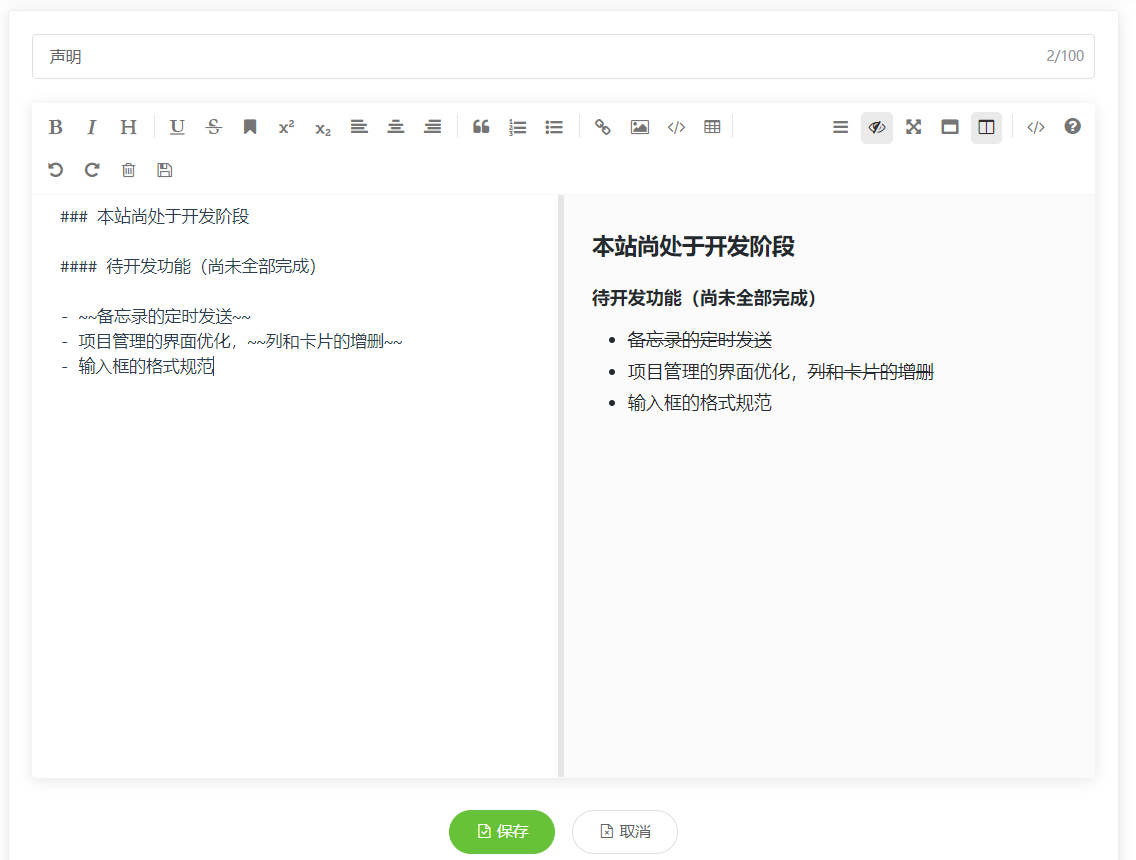


图5-14 日记编辑页面

新增日记接口代码如下：

Diary diary = new Diary();

diary.setUserEmail(JwtTokenUtil.getEmailFromRequest(request));

diary.setDiaryTitle(diaryTitle);

diary.setDiaryBody(diaryBody);

diary.setCreateTime(Calendar.getInstance().getTimeInMillis());

diary.setUpdateTime(Calendar.getInstance().getTimeInMillis());

int res = diaryMapper.insertSelective(diary);

if (res > 0) {

return Result.Success("保存日记成功");

} else {

return Result.Error(Const.StatusCode.ERROR, "数据库插入失败");

}

更新日记接口如下：

Diary diary = new Diary();

diary.setId(id);

diary.setUserEmail(JwtTokenUtil.getEmailFromRequest(request));

diary.setDiaryTitle(diaryTitle);

diary.setDiaryBody(diaryBody);

diary.setUpdateTime(Calendar.getInstance().getTimeInMillis());

int res = diaryMapper.updateByEmailAndId(diary);

if (res > 0) {

return Result.Success("修改成功");

} else {

return Result.Error(Const.StatusCode.ERROR, "修改失败");

}

图片上传接口如下：

String fileName = file.getOriginalFilename();

assert fileName != null;

String fileExtName = fileName.substring(fileName.lastIndexOf(".") + 1);

String fileSaveName = UUID.randomUUID().toString() + "." + fileExtName;

String savePath = QiNiuUploadUtil.uploadImg(file,fileSaveName);

return Result.SuccessByData("图片上传成功", savePath);

③日记详情页面

用户在日记列表页面点击日记标题即可查看日记详情，页面左上角提供返回按钮，页面显示的是markdown渲染后的内容，效果如图5-15所示。



图5-15 日记详情页面

获取日记详情的接口代码如下：

Diary diary = new Diary();

diary.setId(id);

diary.setUserEmail(JwtTokenUtil.getEmailFromRequest(request));

diary = diaryMapper.selectOne(diary);

if (diary != null) {

return Result.SuccessByData("获取日记成功", diary);

} else {

return Result.Error(Const.StatusCode.ERROR, "权限不足或数据不存在");

}

④日记搜索页面

用户在导航栏的搜索框输入信息并提交，后端服务器将会在数据库中模糊查找，并返回相关的日记列表，效果如图5-16所示。

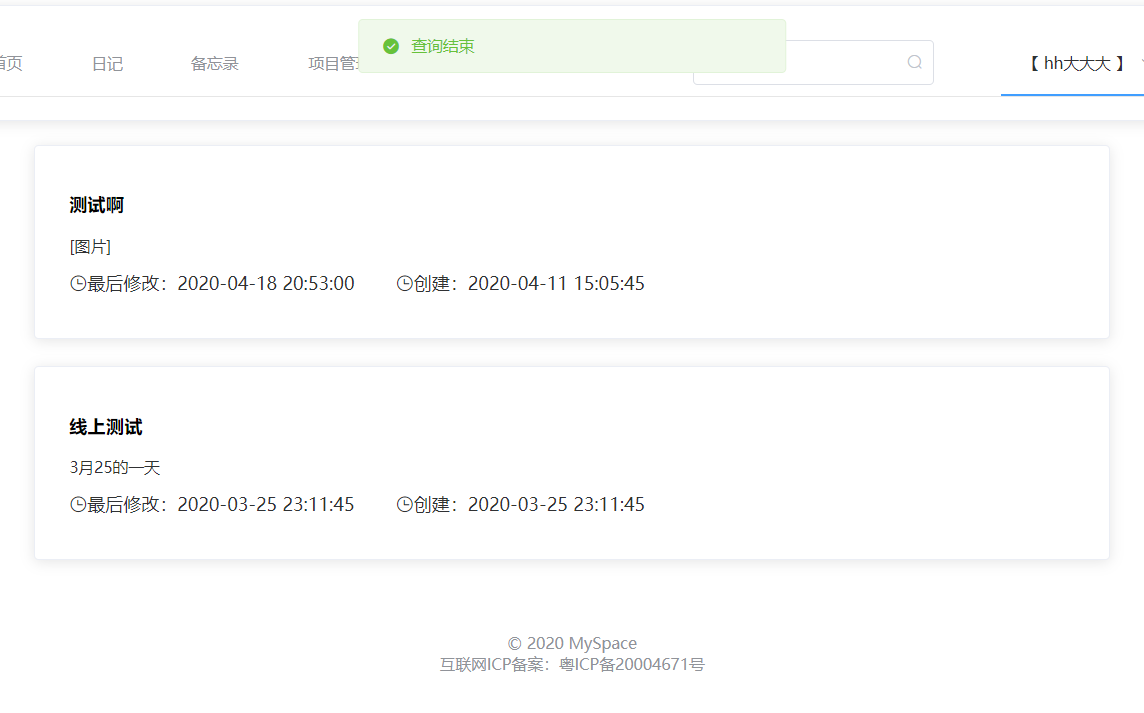


图5-16 日记搜索页面

日记搜索接口代码如下：

PageHelper.startPage(pageNum, pageSize);

List<Diary> diaryList =

diaryMapper.selectByEmailAndText(JwtTokenUtil.getEmailFromRequest(request), text);

for (Diary diary : diaryList) {

String body = diary.getDiaryBody();

if (body.length() > Const.MAX\_BODY\_CHAR\_COUNT) {

diary.setDiaryBody(body.substring(0, Const.MAX\_BODY\_CHAR\_COUNT));

}

}

PageInfo<Diary> pageInfo = new PageInfo<>(diaryList);

if (pageInfo.getTotal() != 0) {

return Result.SuccessByData("搜索成功", pageInfo);

} else {

return Result.Error(Const.StatusCode.ERROR, "没搜索到内容");

}

5.3.3 备忘录模块

备忘录模块包含三种类型：提醒、生日、纪念日，每种类型有对应的新增、删除功能，三种类型对应三个页面，因为三个页面大同小异，在此只演示提醒页面（见图5-17），用户添加备忘录后（见图5-18），后端服务器会生成任务队列，并在所定时间内发送邮件到设置邮箱中，效果如图5-19所示。

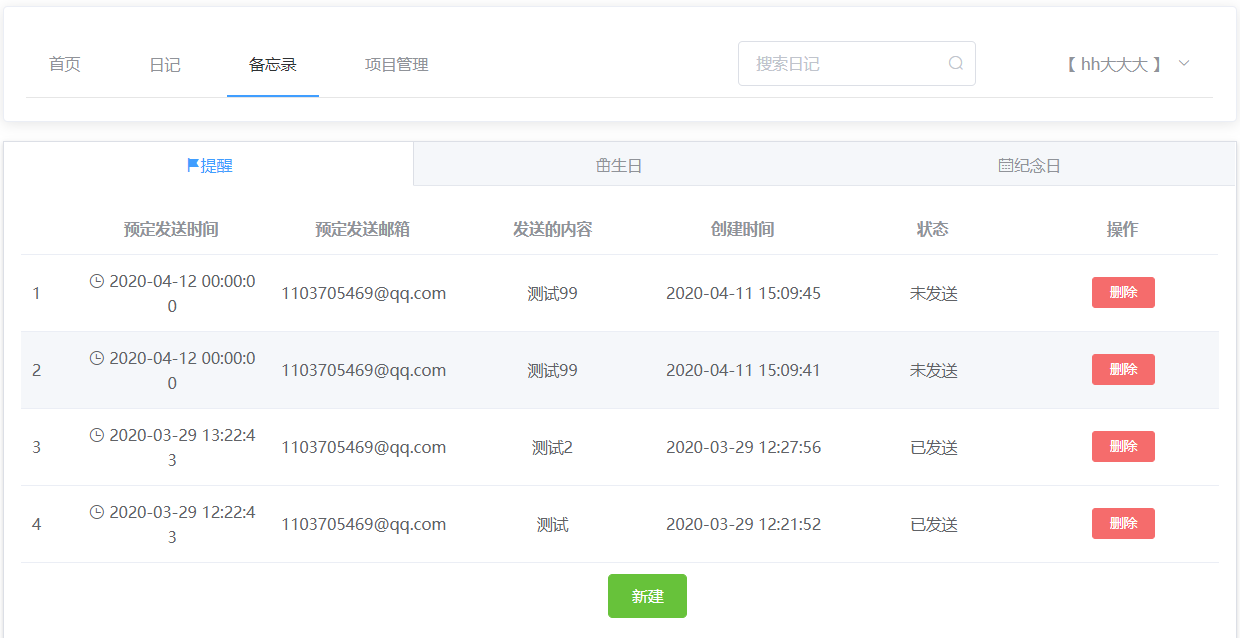


图5-17 提醒页面

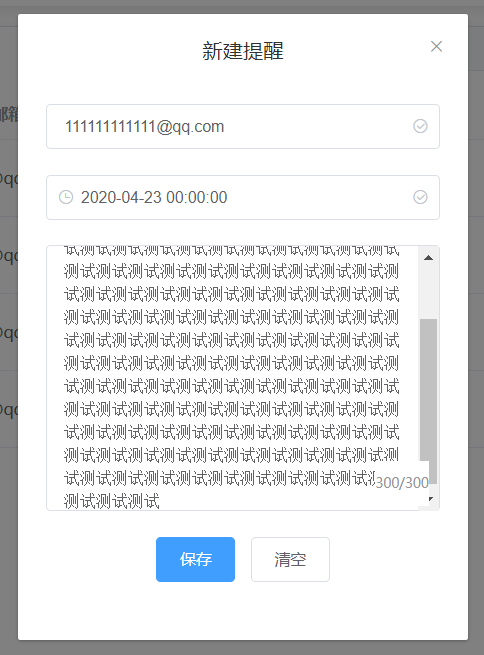


图5-18 新增提醒

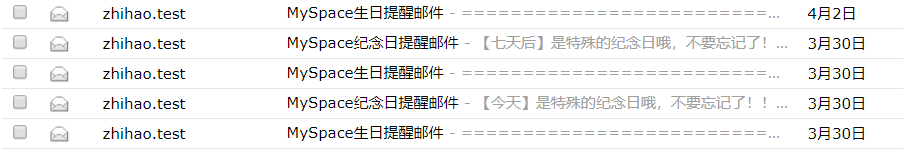


图5-19 提醒邮件

获取备忘录列表接口代码如下：

PageHelper.startPage(pageNum, pageSize);

List<Memo> memoList = memoMapper.selectByTypeAndEmail(type, JwtTokenUtil.getEmailFromRequest(request));

logger.info(memoList.toString());

PageInfo<Memo> pageInfo = new PageInfo<>(memoList);

return Result.SuccessByData("获取成功", pageInfo);

添加备忘录接口代码如下：

memo.setUserEmail(JwtTokenUtil.getEmailFromRequest(request));

memo.setMemoState(Const.Send.NO);

memo.setCreateTime(Calendar.getInstance().getTimeInMillis());

if (memo.getPreTime() != 0) {

memo.setPreDate(DateUtil.computePreDate(memo.getMemoDate(), memo.getPreTime()));

}

int res = memoMapper.insert(memo);

if (memo.getMemoType() == 0) {

ScheduledUtil.addJob(scheduler, RemindJob.class, memo);

}

if (res > 0) {

return Result.Success("保存成功");

} else {

return Result.Error(Const.StatusCode.ERROR, "权限不足或数据库出错了");

}

删除备忘录接口代码如下：

Memo memo = memoMapper.selectByPrimaryKey(id);

if (memo.getUserEmail()

.equals(JwtTokenUtil.getEmailFromRequest(request))) {

int res = memoMapper.deleteByPrimaryKey(id);

if (memo.getMemoType() == 0) {

ScheduledUtil.deleteJob(scheduler, memo.getId() + "", memo.getId() + "");

}

if (res == 1) {

return Result.Success("删除成功");

} else {

return Result.Error(Const.StatusCode.ERROR, "数据出错");

}

} else {

return Result.Error(Const.StatusCode.ERROR, "没有权限");

}

5.3.4 项目管理模块

项目管理模块包含项目的新增、删除、修改、标记完成等功能，应包含项目列表页面、项目详情页面。

①项目列表页面

项目列表页面用于展示用户所拥有的项目，项目分为正在进行和已完成两种，新增、修改、删除项目，以及编辑项目详情等操作都可以在此页面完成，页面效果如图5-20所示。

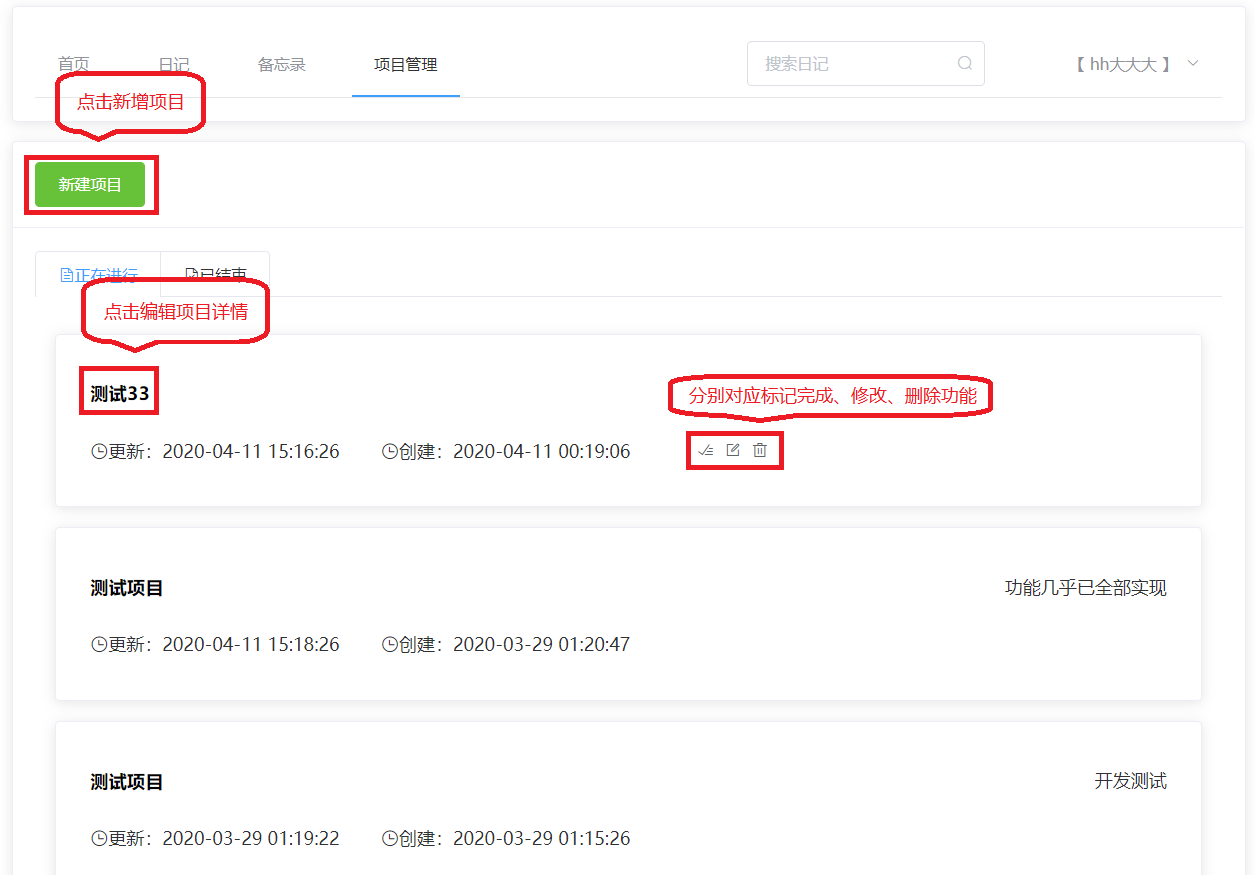


图5-20 项目列表页面

获取项目列表接口代码如下：

PageHelper.startPage(pageNum, pageSize);

ArrayList<Project> projectList = projectMapper.selectByStateAndEmail(projectState, JwtTokenUtil.getEmailFromRequest(request));

PageInfo<Project> pageInfo = new PageInfo<>(projectList);

return Result.SuccessByData("成功获取项目列表", pageInfo);

新增项目接口代码如下：

Project project = new Project();

project.setUserEmail(JwtTokenUtil.getEmailFromRequest(request));

project.setProjectState(Const.Project.UNFINISHED);

project.setProjectBody(projectBody);

project.setProjectDesc(projectDesc);

project.setCreateTime(Calendar.getInstance().getTimeInMillis());

project.setUpdateTime(Calendar.getInstance().getTimeInMillis());

int res = projectMapper.insertSelective(project);

if (res > 0) {

return Result.Success("新建项目成功");

} else {

return Result.Error(Const.StatusCode.ERROR, "插入数据库失败");

}

更新项目接口代码如下：

Project project = projectMapper.selectByPrimaryKey(id);

project.setProjectBody(projectBody);

project.setProjectDesc(projectDesc);

project.setUpdateTime(Calendar.getInstance().getTimeInMillis());

projectMapper.updateByPrimaryKeySelective(project);

return Result.Success("修改成功");

删除项目接口代码如下：

projectMapper.deleteByIdAndEmail(id, JwtTokenUtil.getEmailFromRequest(request));

detailMapper.deleteByParentId(id, JwtTokenUtil.getEmailFromRequest(request));

return Result.Success("项目删除成功");

标记项目完成接口代码如下：

Project project = new Project();

project.setId(id);

project.setUserEmail(JwtTokenUtil.getEmailFromRequest(request));

project=projectMapper.selectOne(project);

if (project==null){

return Result.Error(Const.StatusCode.ERROR,"数据不存在或权限不足");

}else {

project.setProjectState(Const.Project.FINISHED);

project.setUpdateTime(Calendar.getInstance().getTimeInMillis());

projectMapper.updateByPrimaryKeySelective(project);

return Result.Success("项目结束成功");

}

②项目详情页面

项目详情页面展示了一个项目的细节，其分为行和卡片，项目可以包含多行，行可以包含多张卡片，卡片可以在多个行之间拖拽以改变归属行，从而实现方便的项目管理功能，最终效果如图5-21所示。

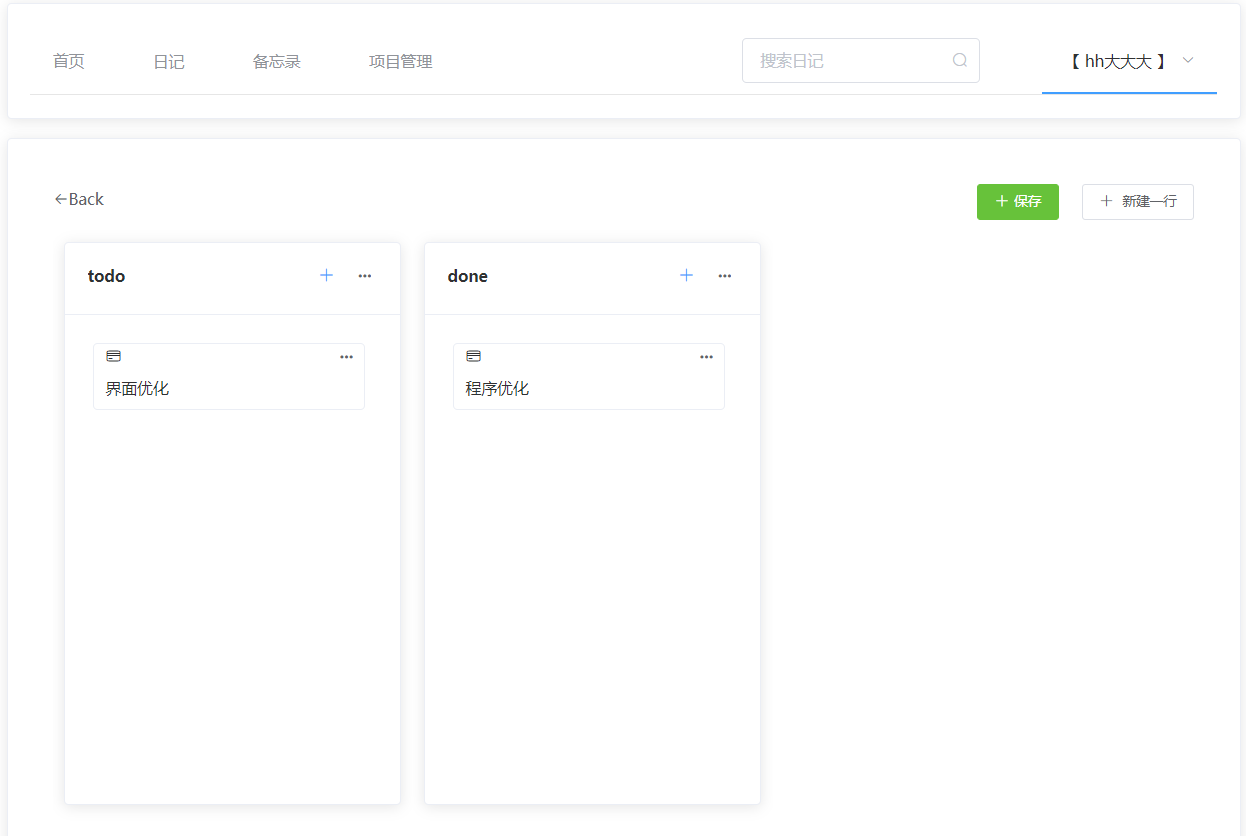


图5-21 项目详情页面

项目详情更新接口代码如下：

Detail curDetail = new Detail();

curDetail.setParentId(id);

curDetail.setUserEmail(JwtTokenUtil.getEmailFromRequest(request));

Detail oldDetail = detailMapper.selectOne(curDetail);

if (oldDetail == null) {

curDetail.setDetail(detail);

detailMapper.insertSelective(curDetail);

} else {

curDetail.setId(oldDetail.getId());

curDetail.setDetail(detail);

detailMapper.updateByPrimaryKey(curDetail);

}

Project project = new Project();

project.setId(id);

project.setUpdateTime(Calendar.getInstance().getTimeInMillis());

projectMapper.updateByPrimaryKeySelective(project);

return Result.Success("保存成功");

5.3.5 后台管理模块

后台管理模块包含用户信息的查看、删除和用户分析功能，应包含用户管理页面、用户画像页面。

①用户管理页面

后台管理入口只有管理员权限才会显示，管理员可以由此查看网站所有用户的信息，并对违规的用户进行删号处理，最终效果如图5-22所示。



图5-22 用户管理页面

获取用户列表接口代码如下：

PageHelper.startPage(pageNum, pageSize);

List<User> users = userMapper.selectAll();

PageInfo<User> pageInfo = new PageInfo<>(users);

return Result.SuccessByData("获取用户列表成功", pageInfo);

删除用户接口代码如下：

userMapper.deleteByPrimaryKey(id);

return Result.Success("删除用户成功");

②用户画像页面

用户画像页面用于方便管理员了解网站用户分布，为了使用方便，以对用户收集的信息越少越好为原则，故该页面当前只提供男女比例画像，使用圆形比例图展示，效果如图5-23所示。



图5-23 用户画像页面

获取男女比例接口代码如下：

double percentage = userMapper.selectBoy(Const.Sex.BOY);

int per = (int) (percentage \* 100);

return Result.SuccessByData("查询用户分布成功",per);

5.4 网站的部署

完成编码后，需要将项目部署到云服务器，以方便外网的用户访问。这里我选择的是腾讯云服务器，服务器配置为：1核 2GB 1Mbps。选择好服务器后，需要将我们的前后端项目构建或打包、然后使用Nginx服务器转发实现部署。

1. 构建前端静态文件

首先进入前端项目所在根目录，调出cmd界面，输入npm run build命令进行构建，运行效果如图5-24所示。

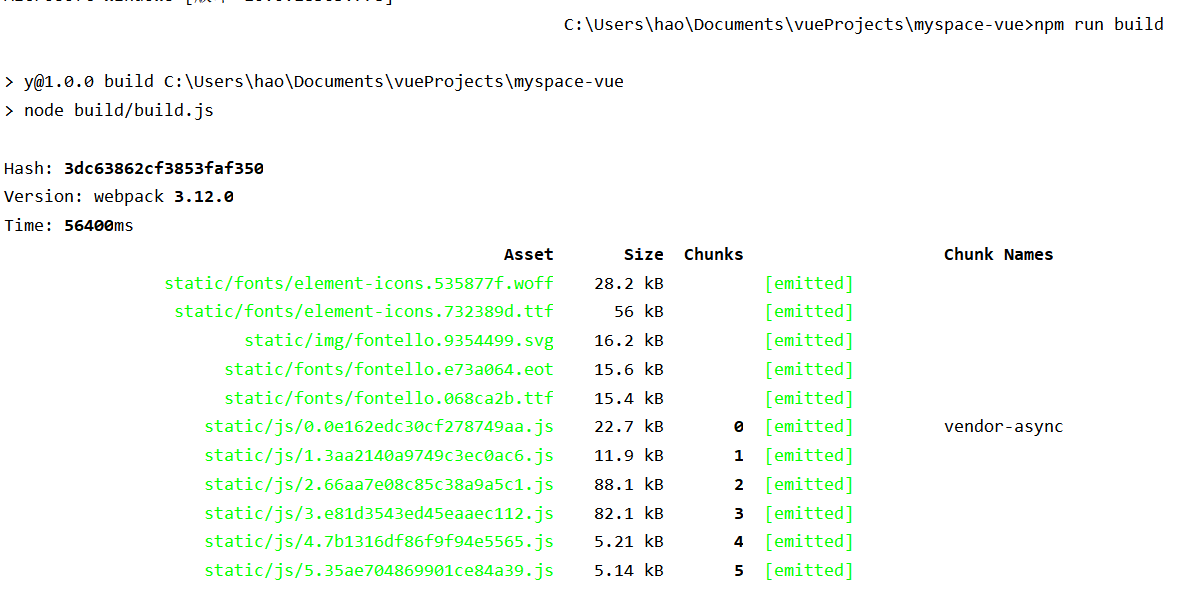


图5-24 前端静态文件构建

构建完成后会生成dist文件夹，里面有index.html文件和static文件夹，利用上传工具xftp上传到云服务器。

1. 打包SpringBoot项目

接着在IDEA编辑器中调出maven插件工具，点击package进行打包，运行效果如下图5-25所示。

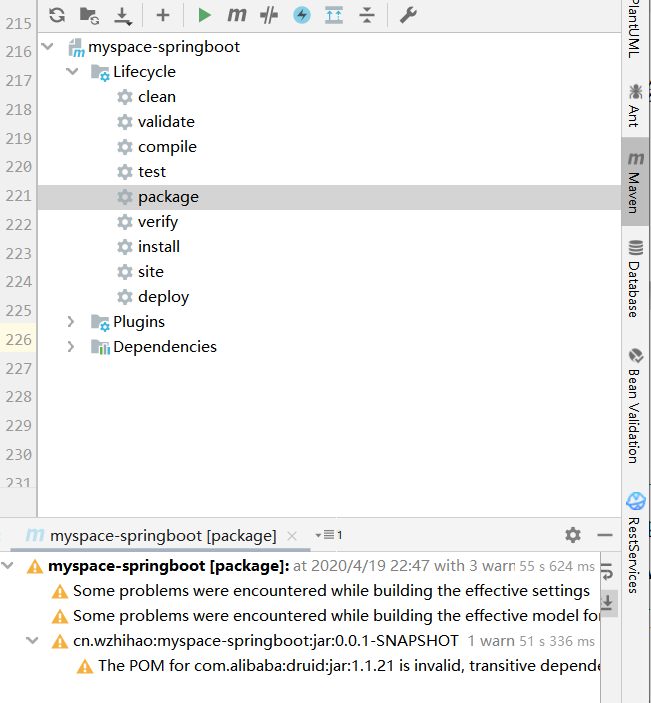


图5-25 SpringBoot项目打包

打包成功会生成target命令，从根目录可以找到对应的.jar包，将改包上传到服务器，使用nohup java -jar xxxxx.jar &命令运行后端项目。

1. 配置Nginx转发服务

最后在云服务器安装好Nginx服务器，进入conf文件夹配置.conf文件，详细配置如图5-26和图5-27所示。

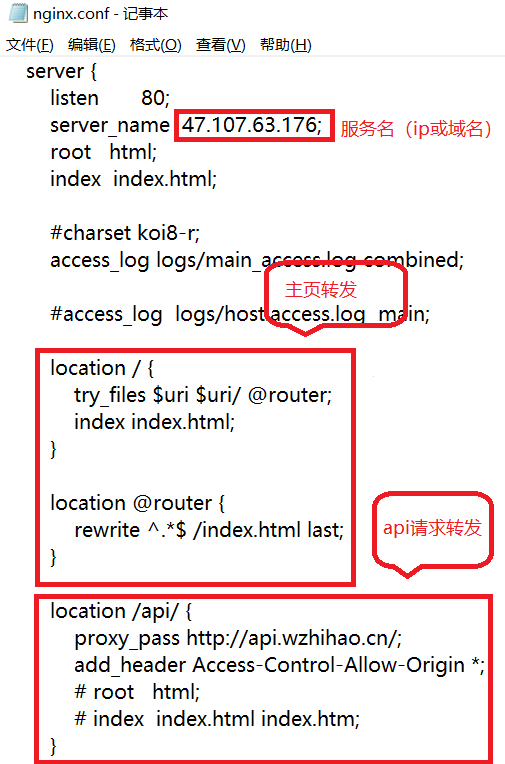


图5-26 前端请求转发



图5-27 后端api请求转发

至此，在浏览器网址栏输入云服务器ip成功访问到首页，说明项目部署成功。

6 结论与展望

6.1 结论

本毕业设计主要介绍了基于SpringBoot的个人网站的设计与实现，从课题确定后，笔者先详细的分析了网站的功能需求，构想出大致的界面设计效果，之后笔者通过不断的查阅文献资料，确定了技术选型，针对选定技术进行深入研究，完成了功能模块的详细设计、各层接口的规范与约定、数据库表的相关设计等工作，最后根据设计完成了各部分的编码工作并成功部署项目上线。

通过这次项目开发，使我进步飞快，让我的编程能力有了极大的突破，在开始实现项目之前，笔者实际上有许多不熟悉的地方，对涉及的技术有些也只是一知半解，无论是前端页面的编写，还是后端的框架使用，都是花费了大量时间查阅资料，不断实践才慢慢掌握的。另外在完整的开发流程之下，笔者也明白了软件开发的一些要领，从而总结出一个“保守又开放”原则，即软件都是从无到有、从小到大发展而来的，因此设计实现的时候不能空谈，而是最好能先进行简单的原型实现，分析实现的可行性再做决定，从而做到技术上的“保守”，而在设计的过程中不仅要着眼当下，还要展望未来，对接口、模块的设计要充分的考虑到日后的扩展可能性，从而达到设计上的“开放”。对于这次项目能顺利完成，笔者还是很高兴的，我得到了成长，也获得了丰富的开发经验，相信这段经历能为我以后的软件开发生涯提供巨大的帮助。

6.2 展望

目前该项目已经部署上线，相信推广之后用户数量会越来越多，而基于高扩展性的接口设计，日后网站所能提供的功能也将越来越丰富。当然，一个好的产品也需要能直面自己的缺点，本系统框架也是不尽完美的，有一些需要改进的地方，这也是展望之处：

1. 减少不必要的冗余代码；
2. 进一步提升页面的加载速度；
3. 卡片拖拽的效果可以更加顺畅和舒适。

参考文献

1. Hatma Suryotrisongko,Dedy Puji Jayanto,Aris Tjahyanto. Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot[J]. Procedia Computer Science,2017,124.
2. Mirko Perkusich,Lenardo Chaves e Silva,Alexandre Costa,Felipe Ramos,Renata Saraiva,Arthur Freire,Ednaldo Dilorenzo,Emanuel Dantas,Danilo Santos,Kyller Gorgônio,Hyggo Almeida,Angelo Perkusich. Intelligent software engineering in the context of agile software development: A systematic literature review[J]. Information and Software Technology,2020,119.
3. 刘桃丽,曾志超.MVC架构下网站的设计与实现[J/OL].计算机技术与发展,2020(02):1-6[2019-12-12].
4. 莫秋晶,黄志远,王爱华,黄华,陶歆.基于Spring Boot的信息化服务管理系统的设计与实现[J].电子技术与软件工程,2018(21):50-51.
5. 贺奕橙,张丽霞,谢宇彤,于钟皓.大学校园服务网站设计与开发[J].信息与电脑(理论版),2019,31(18):78-79.
6. 孙一笑,张玉军,孙宇成,吕卉元.基于WebAPI前后端完全分离的软件开发模式[J].信息与电脑(理论版),2019(06):96-97.
7. 江雁.浅谈Spring Boot框架下如何快速进行后台开发[J].海峡科技与产业,2019(02):131-132.
8. 刘云浩,李沛.基于Spring Boot的后台服务器开发[J].中国科技信息,2018(17):61-62.
9. 张峰.应用SpringBoot改变web应用开发模式[J].科技创新与应用,2017(23):193-194.
10. 梁文楷.基于Elasticsearch全文检索系统的实现[J].电脑编程技巧与维护,2019(06):116-119.

致 谢

在论文完成之际，我要感谢我的指导教师蒋辉教授，感谢他对我孜孜不倦的教诲、耐心和关怀，在这疫情爆发的关键时刻，他依旧坚守在教学一线，通过线上指导为我们的毕业论文答疑解惑，他严谨的教学态度和过硬的理论知识深深的感染着我，每每在我项目遇到困难想要放弃时给予我动力，让我时刻保持着一颗积极向上的心，对每件事保持激情，从而把事做好，做完美，在完成论文的过程中，也是老师给出了诸多建议和帮助，我的论文才能如此顺利的完成，在此，向蒋辉老师致以最崇高的谢意！

其次，感谢大学四年来所有教导过我的老师们，感谢你们的耐心教导，感谢你们的奉献，感谢大学四年一起学习的同学朋友，感谢你们的鼓励和帮助。

最后，感谢爸爸妈妈一直以来的支持和陪伴，祝爸爸妈妈身体健康，祝蒋辉老师万事如意，祝老师们开开心心，祝同学朋友们毕业快乐，前程似锦！

附录（含课题审

批表、任务书、开题报告、中期检查表、指导记录、成绩评定表）。