****

**编译原理与技术大作业开发报告**



**题目：C--编译器的实现**

**学 院 智能与计算学部**

**专 业 软件工程专业**

**姓 名 周翔**

**学 号 3019244136**

**指导教师 韩亚洪**

目 录

[第一章 绪论 5](#_Toc27423)

[1.1 背景 5](#_Toc20339)

[1.2 定义 5](#_Toc5833)

[1.3 假定和约束 5](#_Toc11205)

[1.4 目标 6](#_Toc5130)

[第二章 需求分析 7](#_Toc27900)

[2.1 业务概念分析 7](#_Toc22169)

[2.1.1 概述 7](#_Toc32597)

[2.1.2 业务概念总览 7](#_Toc19349)

[2.2 业务流程分析 7](#_Toc28138)

[2.2.1 概述 7](#_Toc8799)

[2.2.2 流程总览 7](#_Toc24726)

[第三章 结构设计 9](#_Toc18777)

[3.1 基本设计概念 9](#_Toc16966)

[3.1.1 技术总览 9](#_Toc9613)

[3.1.2 软件功能架构总览 9](#_Toc6784)

[3.2 前端设计 9](#_Toc1154)

[3.2.1 项目依赖统一管理 9](#_Toc15455)

[3.2.2 样式全局管理 10](#_Toc461)

[3.2.3 页面组件分模块管理 11](#_Toc15973)

[3.2.4 Axios封装请求 12](#_Toc9763)

[3.2.5 请求拦截器的使用 12](#_Toc2503)

[3.3 后端架构设计 14](#_Toc29682)

[3.3.1 三层架构及接口设计原则 14](#_Toc1308)

[3.4 相互关系 17](#_Toc30157)

[3.4.1 ER图 17](#_Toc15775)

[3.4.2 DAO层 19](#_Toc17710)

[3.4.3 Service层 20](#_Toc19295)

[3.4.4 Controller层 21](#_Toc29791)

[第四章 系统测试 22](#_Toc20500)

[4.1 测试对象 22](#_Toc23685)

[4.1.1 软件基本信息 22](#_Toc11634)

[4.1.2 测试接口概述 22](#_Toc6431)

[4.2 测试方法与内容 23](#_Toc14122)

[4.2.1 Postman接口测试 23](#_Toc1153)

[4.2.2 JUnit接口测试 24](#_Toc1541)

[第五章 系统部署 26](#_Toc25866)

[5.1 前端部署 26](#_Toc31929)

[5.1.1 修改资源路径 26](#_Toc14632)

[5.1.2 关闭跨域代理 26](#_Toc21596)

[5.1.3 打包 26](#_Toc16489)

[5.2 后端部署 26](#_Toc30132)

[5.2.1 Idea连接远程服务器 26](#_Toc4651)

[5.2.2 Application.yml文件的配置 27](#_Toc19885)

[5.1.3 项目打包 27](#_Toc22181)

[5.1.4 Xftp部署项目并运行 27](#_Toc32433)

[5.1.5 配置接口域名绑定以及第三方资源路径代理 28](#_Toc7520)

[第六章 项目总结 29](#_Toc7626)

[参考文献 30](#_Toc25692)

[致 谢 31](#_Toc24998)

# 绪论

## 1.1 背景

在学习了编译原理这门课程后，为加深对编译知识的理解，我们协作完成了C--编译器的开发。

# 需求分析

## 2.1 业务功能说明

### 2.1.1 概述

本系统的目标是完成代码编译的词法分析和语法分析部分demo。

### 2.1.2 业务功能的规格说明

总体业务概念如图2-1所示。

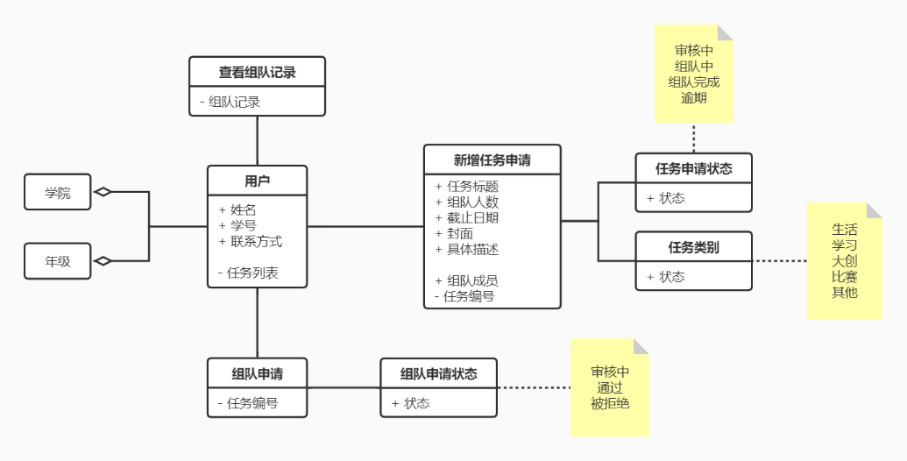


图2-1 总体业务概念类图

## 2.2 业务流程分析

### 2.2.1 概述

新增任务申请和组队申请都需要审核，且在不同阶段处于不同状态。

### 2.2.2 流程总览

流程中的角色有组长，管理员，用户，主要流程有新增任务申请和组队申请。所有业务流程的顺序图如图2-2所示。

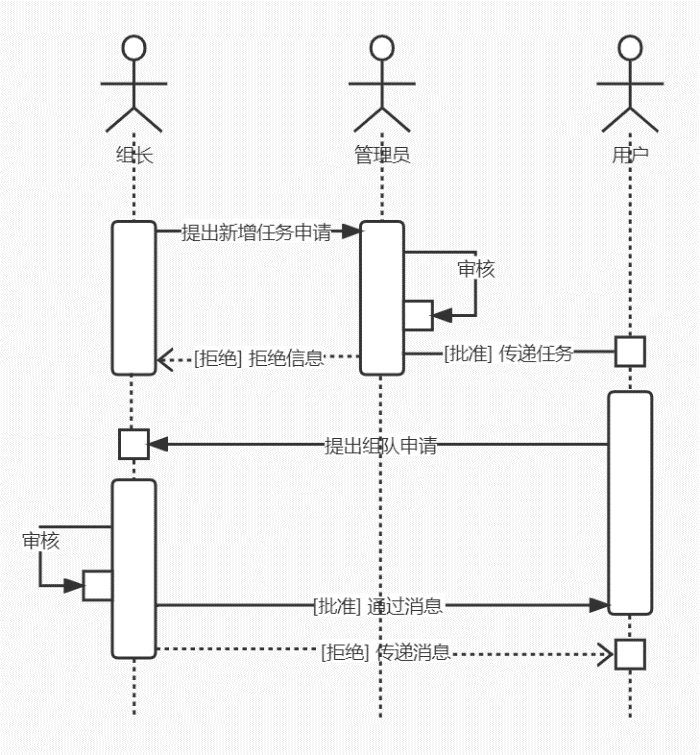


图2-1 业务流程顺序图

# 结构设计

## 3.1 基本设计概念

### 3.1.1 技术总览

表3‑1 开发所用的主要技术总览

|  |  |
| --- | --- |
| **前端** | vue.js, elementui, animate.css, fontawesome, googlefonts, iconfont, quileditir, boxicons, axios, sessionStorage和localStorage router路由 |
| **后端** | springboot, springDataJpa, maven, junit |

### 3.1.2 软件功能架构总览

本软件使用 Vue（前端）和SpringBoot（服务端）开发基于前后端分离架构的“‘北洋有约’校园互拼平台”。根据需求文档，我们将软件分为六个功能模块，分别围绕日志记录、系统消息、个人信息、团队信息、标签板块和任务招募展开。前五个模块包含的函数主要实现信息数据的增删改查，任务招募模块的函数功能则主要实现招募任务的生命流程。基于功能模块的划分，整个后端基于三层架构设计实现，每个模块拥有自己的DAO层、Service层和Controller层，结合实体层Entity和数据库完成数据的处理和传递。

## 3.2 前端设计

### 3.2.1 项目依赖统一管理

公用组件库与elementui, animate.css, aioxs等技术统一在main.js中进行加载，对于googlefonts, fontawesome, echarts等在index.html中使用cdn链接加载减少最终打包的项目大小。



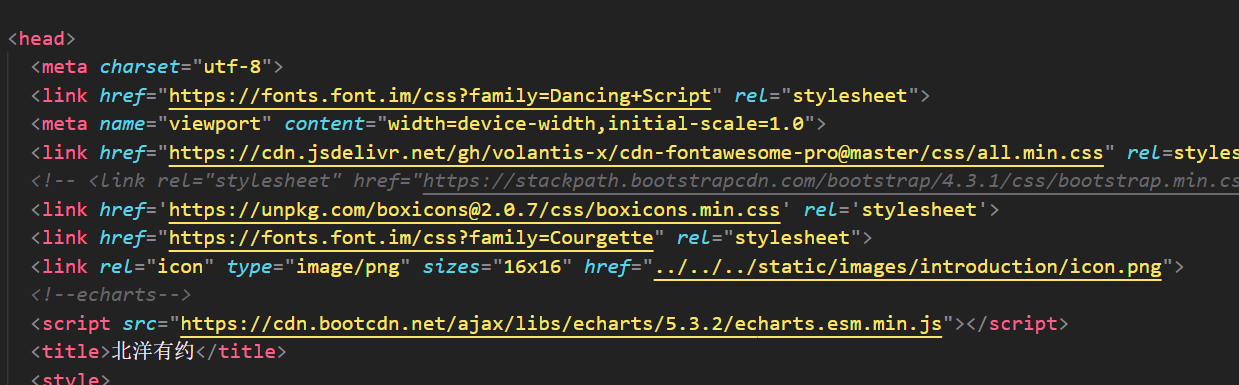


图3‑1 前端组件和技术依赖导入

(a) 前端组件导入；(b) 技术依赖导入

### 3.2.2 样式全局管理

为了方便修改全局的主题配色，在index.html中使用root设置了全局的主题配色。当需要修改全局主题配色时，只需要在这里进行修改即可完成全局的操作，增强了后期可维护性。

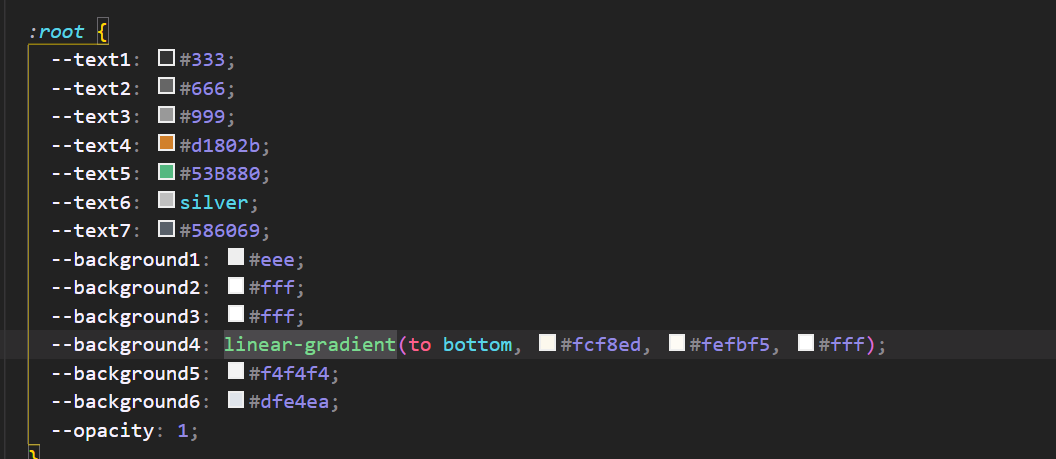


图3-2 全局样式管理

每一个组件都有对应的局部样式文件，写在了各自vue文件中的style标签下，其中标签的scoped属性会保证其样式只在对应的组件局部生效，不会造成全局的样式污染。

全局样式的添加、组件库样式覆盖等类似需求，统一在src/styles下的index.scss中完成，scss优先级更高，可以覆盖默认样式。同时为了防止全局污染，每一个样式选择器都添加了对应页面的名称。

### 3.2.3 页面组件分模块管理

为了增强文件结构的可读性及功能模块的可扩展性和可维护性，每一个页面都独立地对应一个views页面下的文件夹，在每一个这样的文件夹中，index.vue是该模块页面的入口加载文件，其他的是该模块页面index.vue使用的私有组件。全局的共用组件封装在src/components下方。

考虑到接口可能在后期需要进行更改，并且伴随着业务的增加，后期的接口数量也会显著增加，我们按模块分装了每一个页面使用的接口调用方法，apis下的每一个文件夹都对应一个页面的所有接口请求函数。每一个页面使用的第三方静态资源如图标、图片素材等，也分模块存储到了static下，保证各文件的存储内容到开发后期也是易于辨识的，便于在不会影响其他模块正常工作的前提下对某个模块进行需改。

### 3.2.4 Axios封装请求

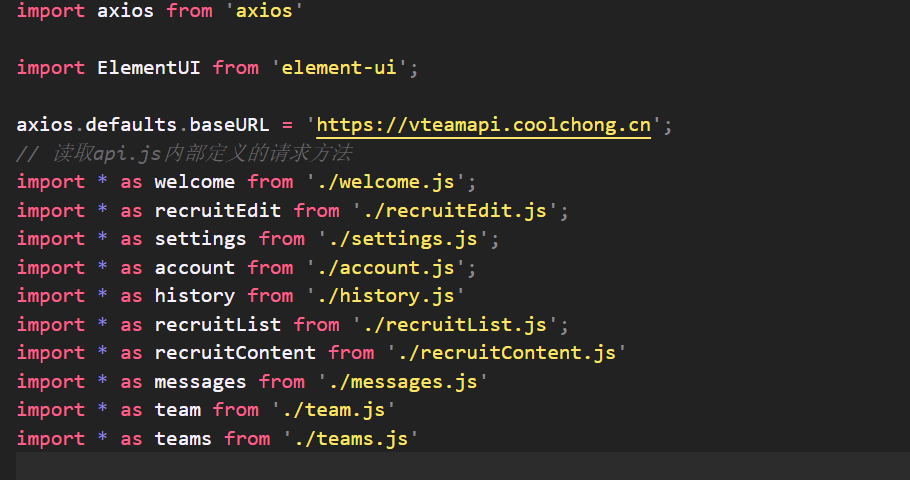


图3-3 Axios封装请求

一般情况下，如果将请求函数的具体实现写入vue组件，后期维护接口时就不得不在众多组件的script js中查找修改。再加上每一个vue组件中的请求函数都会写出具体的url路径，如果有1000个请求接口函数，就会有1000个url和具体的请求函数实现方法随机分布在不同的vue组件中，造成后期维护成本指数型增长。因此为了提高后期的代码可扩展性，我们对axios请求函数进行统一的封装管理。

我们建立了index.js用于提取各模块所有的请求函数，封装成一个对外开放的对象apis。这个apis对象包含了整个项目所有请求函数的具体实现，并通过main.js加载到全局，当页面调用请求函数时，就无需再写具体的实现逻辑了，只需要调用apis已经提前封装好并对外开放的请求函数。

还需要在index.js中使用axios.baseUTL设置统一的接口路径，所有封装的请求函数都会默认加载这个路径。如果后台ip发生更改，前台开发人员也只需要对index.js中的baseURL进行一次修改，即可完成整个项目上百个请求函数路径的更改。

### 3.2.5 请求拦截器的使用

这里我们使用了请求拦截器以及响应拦截器，其中请求拦截器主要用于实现token的自动加载。由于我们对除了登录注册以外的所有接口都使用了token认证，因此前台需要在这些请求发送前，在请求头中加入token值，而这个token值是存储到了本地的sessionStorage中进行持久化存储。使用请求拦截器会在每一个请求发送之前首先检测本地是否存储有token值，如果检测到了会自动在请求头中加入token值，从而实现token的自动检验装载。



图3‑4 请求拦截器

响应拦截器主要用于报错。由于后台可能会出现请求处理的错误并返还500错误码，所以每一个请求函数都需要添加catch来进行错误码的前台报错。但是这使代码变得冗余，我们可以使用响应拦截器对请求响应进行拦截，然后检验返还的状态码。如果状态码为200，则正常请求并返还响应结果；如果返还的请求响应状态码不正常，响应拦截器就会自动catch这些响应报错，从而实现了代码的简洁性。



图3‑5 响应拦截器

## 3.3 后端架构设计

### 3.3.1 三层架构及接口设计原则

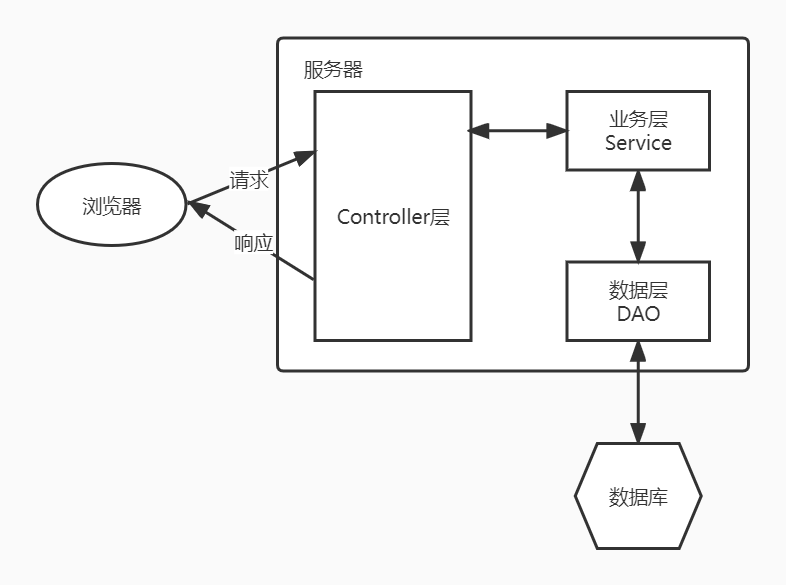


图3‑6 三层架构示意图

从逻辑上划分，基于Spring框架，我们将软件后端分为Repository层、DAO层、Entity层、Service层和Controller层。Controller层调用Service层的方法，Service层调用DAO层中的方法，其中调用的参数使用Entity层进行传递。

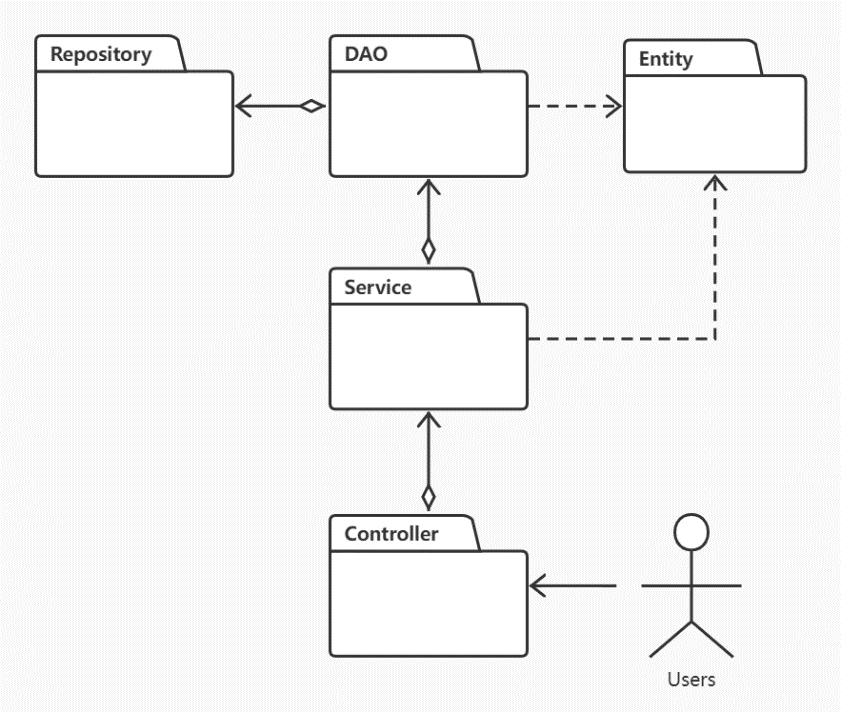


图3‑7 包图

Repository层直接与数据库对接，继承数据库的操作函数。

Entity层是实体层，也就是所谓的model，是数据库在项目中的类，该文件包含实体类的属性和对应属性的set、get方法。在本软件项目中，Entity层主要包括的数据结构有：日志记录、系统消息、招募中的任务信息、标签板块名称、用户个人展示信息、组队成功的团队信息和用户的系统信息。

DAO层会导入Repository类，对数据库实际使用到的方法进行封装以供Service层调用。比如在日志记录的功能模块中，DAO层在insertHistory()方法中调用Repository层的save()函数，对应数据库中的“增”。

Service层主要负责业务模块的逻辑应用设计。Service层会调用DAO层的接口，接收DAO层返回的数据，完成项目的基本功能设计。封装Service层的业务逻辑有利于业务逻辑的独立性和重复利用性。

Controller层负责具体的业务模块流程的控制，包括前后端交互，接受前端请求，调用Service层，接收service层返回的数据，最后返回具体的页面和数据到客户端。即，将前端的请求和数据类型进行汇总整理交给Service层进行业务处理，并将Service层处理的结果返回给前端。

例如在系统消息功能模块的工作流程中，用户进入主页，页面需要显示任务招募信息列表，调用Controller层的函数getScreenRecruitInfo()并发送http请求，Controller层调用Service层的getScreenRecruitInfo()函数并等待返回结果。Service层调用DAO层函数getValidRecruitInfo()，并注入Entity层的RecruitInfo类用于传递DAO层获得数据。DAO层将查询得到的原始数据存储在Entity层的RecruitInfo类中后，Service层根据需求调用RecruitInfo类的get方法，将获得的属性数据进行处理整合并统一格式，再返还给Controller层。Controller将Service层处理好的招募列表信息呈现给用户。

在实现三层架构的过程中，我们使用了“先设计放接口的类，再创建实现的类”的思维方式，通过对接口而不是对类编程达到代码重用的目的，尽可能地保证了接口的功能单一和高可复用性。三层架构的代码结构清晰，每层之间耦合度低，可维护性高，可扩展性高，利于开发任务同步进行，容易适应需求变化。

下面展示功能模块的部分结构类图：

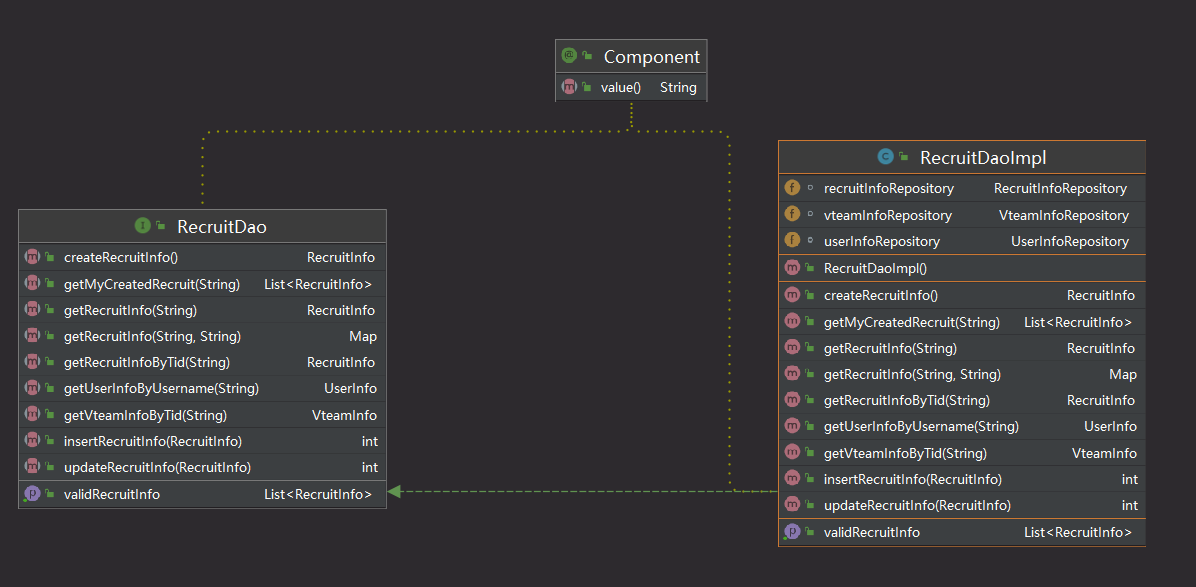


图3‑8 DAO层任务招募模块函数结构

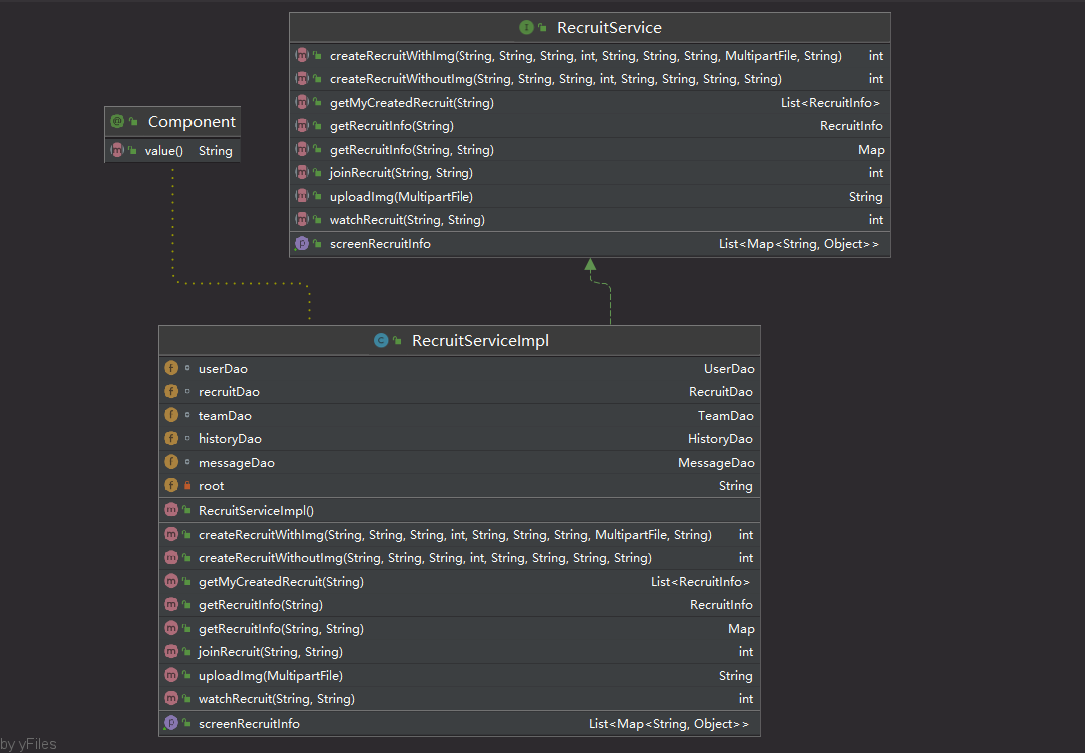


图3‑9 Service层任务招募模块函数结构

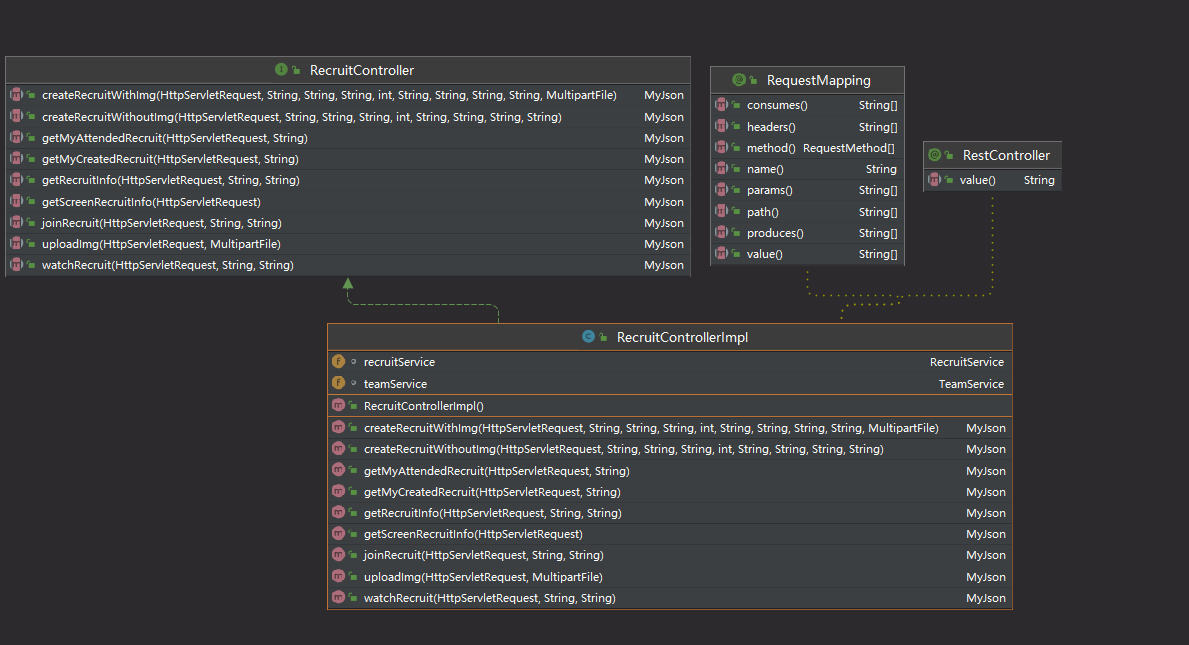


图3‑10 Controller层任务招募模块函数结构

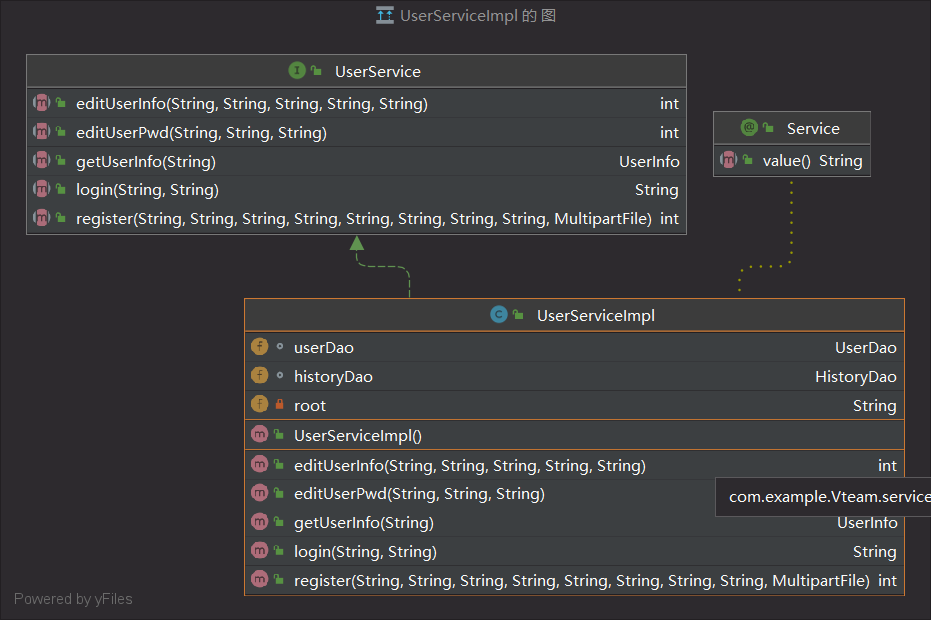


图3‑11 Service层用户信息模块函数结构

## 3.4 相互关系

### 3.4.1 ER图

本项目的数据库ER图如下：

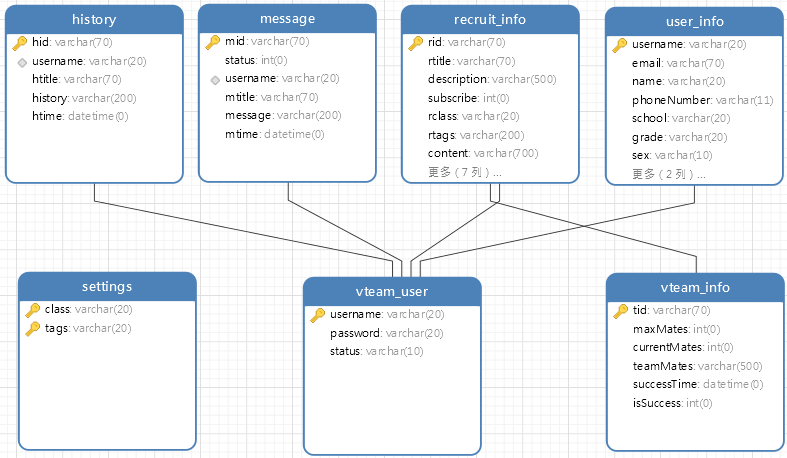


图3‑12 数据库表之间的联系

### 3.4.2 DAO层

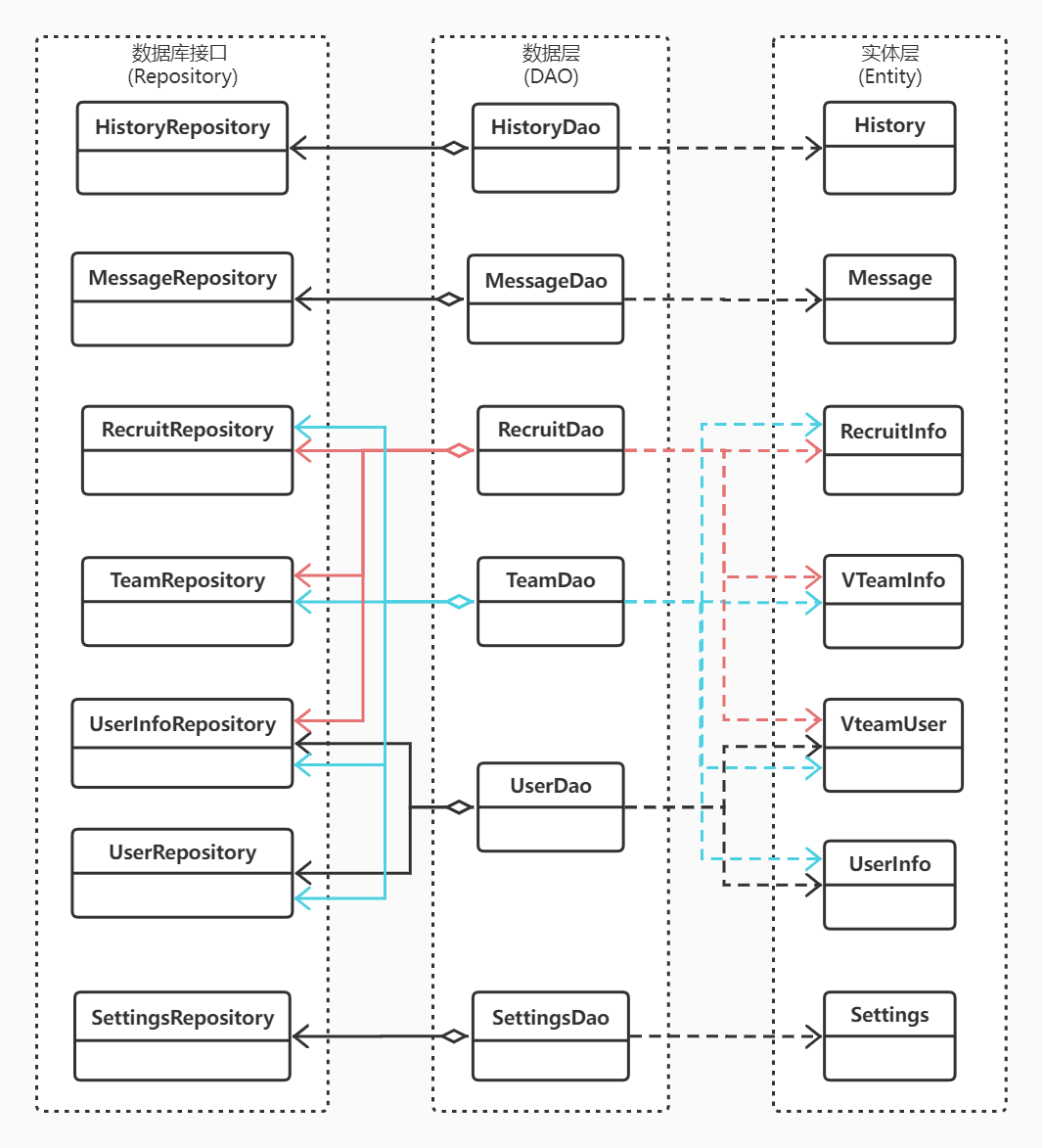


图3‑13 类图1: DAO Repository Entity

### 3.4.3 Service层

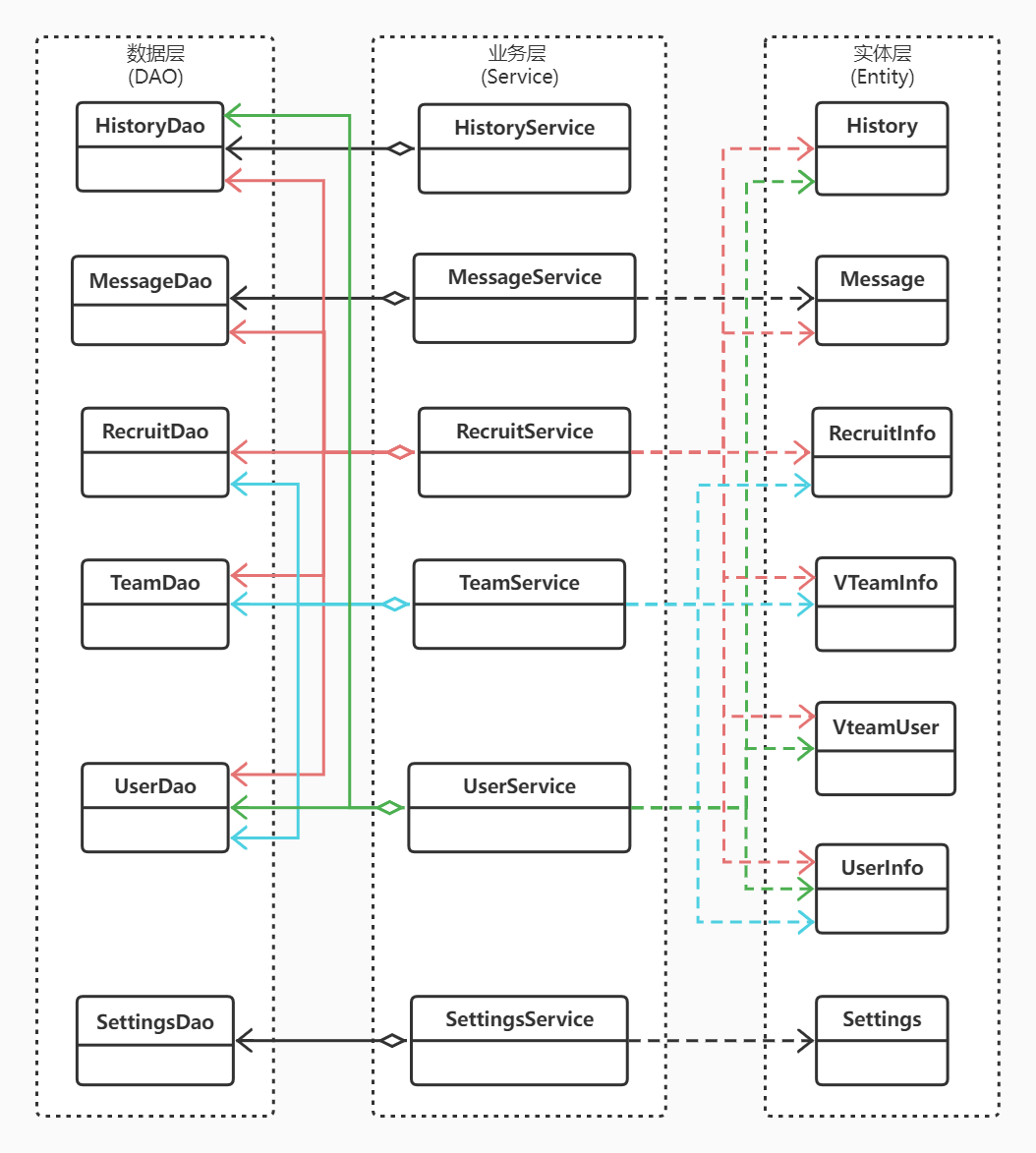


图 3-14 类图2: DAO Service Entity

### 3.4.4 Controller层

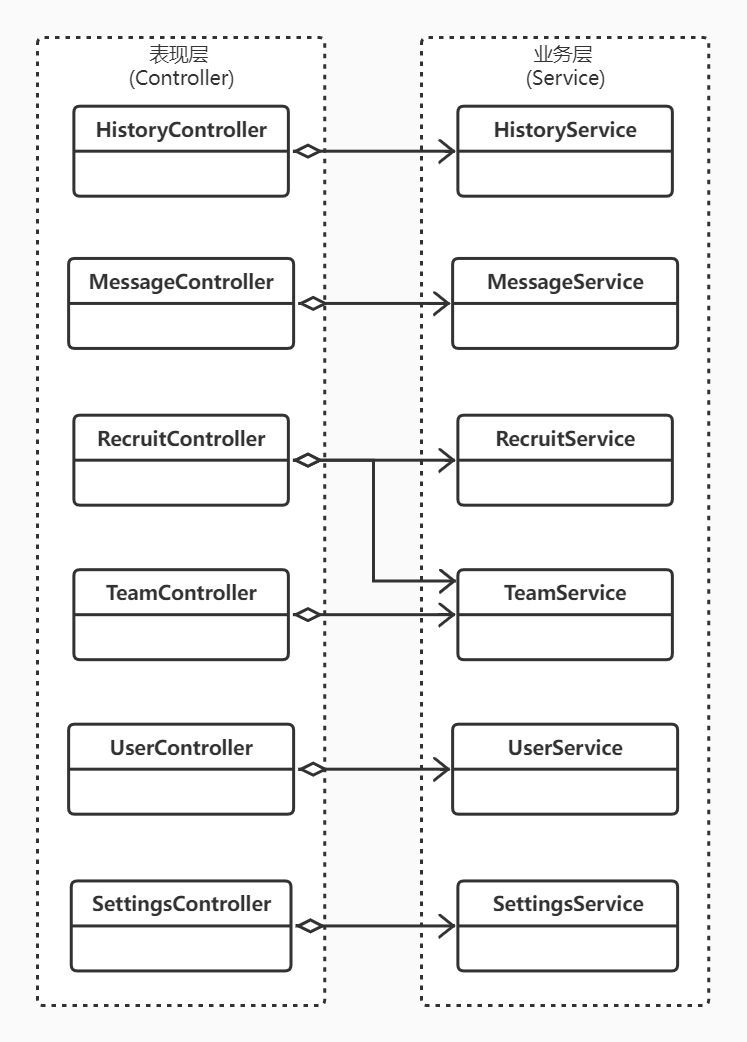


图3-15 类图3: Controller Service

# 系统测试

## 4.1 测试对象

### 4.1.1 软件基本信息

表4‑1 软件开发基本信息

|  |  |  |  |
| --- | --- | --- | --- |
| **被测软件名称** | **开发语言** | **开发环境** | **运行环境** |
| “北洋有约”校内招募组队平台 | Java | Intellij idea | centOS 8 |

### 4.1.2 测试接口概述

测试接口如表4-2所示。

表4‑2 测试接口说明及与Service层模块的对应关系

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **标识符** | **功能名称** | **用途与说明** | **对应模块**  **(Service层)** |
| 1 | COT\_GXCX | 个人信息查询 | 通过用户id进行查询，可以返回用户详细信息，包括邮箱地址、姓名、电话号码、性别、年级、注册时间等。 | User  Service |
| 2 | COT\_GXZC | 个人信息注册 | 通过填写用户基本信息，可以将用户详细信息上传至数据库，包括邮箱地址、姓名、电话号码、性别、年级、注册时间等。 |
| 3 | COT\_GXXG | 个人信息修改 | 通过修改用户基本信息，可以修改数据库中的用户详细信息，包括邮箱地址、姓名、电话号码、性别、年级、注册时间等。 |
| 4 | COT\_MMXG | 密码修改 | 通过修改用户密码，可以修改数据库中的密码信息。 |
| 5 | COT\_TCCX | 团队成员查询 | 通过团队id进行查询，可以返回团队所有成员的详细信息。 | Recruit  Service |
| 6 | COT\_YTCX | 用户团队查询 | 通过用户id进行查询，可以返回该用户参与的所有团队信息，包括团队名称、团队成员人数、团队创建时间、团队所属板块等。 |

表4‑2 测试接口说明及与Service层模块的对应关系(续)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **标识符** | **功能名称** | **用途与说明** | **对应模块**  **(Service层)** |
| 7 | COT\_BBZL | 标签板块总览 | 通过用户id进行查询，可以返回该用户参与的所有团队信息，包括团队名称、团队成员人数、团队创建时间、团队所属板块等。 |  |
| 8 | COT\_TZCJ | 团队招募创建 | 通过填写团队基本信息，可以将团队详细信息上传至数据库，包括成员id、团队名称、任务描述、成员人数、结束时间、板块标签等。 |
| 9 | COT\_TZJR | 团队申请加入 | 通过添加团队加入申请，可以成为该团队的成员之一。 |
| 10 | COT\_GTCX | 个人团队查询 | 通过用户id进行查询，可以返回该用户参加的所有团队信息，包括团队名称、团队成员人数、团队创建时间、团队所属板块等。 | Team  Service |
| 11 | COT\_TXCX | 团队具体招募信息查询 | 通过用户id和招募id进行查询，可以返回该团队的具体招募信息，包括招募名称、招募创建人、招募创建时间、招募所属板块等。 |
| 12 | COT\_ZZCX | 主页招募信息列表查询 | 通过postman查询，可以返回主页的基本招募信息列表，包括招募名称、招募人数需求、已申请人数、招募所属板块等。 |
| 13 | COT\_ZRGX | 招募热度值更新 | 通过用户id和招募id进行查询，可以更新主页的招募热度值。 |
| 14 | COT\_GMCX | 个人消息列表查询 | 通过用户id，可以返回个人收到的所有消息。 | Message Service |
| 15 | COT\_GMGX | 个人消息状态更新 | 通过消息id或用户id，可以更新消息状态为已读。 |
| 16 | COT\_GRCX | 个人日志查询 | 通过用户id进行查询，可以返回用户详细信息，包括邮箱地址、姓名、电话号码、性别、年级、注册时间等。 | Histroy  Service |

## 4.2 测试方法与内容

### 4.2.1 Postman接口测试

Postman接口测试如表4-3所示。

表4‑3 COT\_GXCX\_1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **用例名称** | 个人信息查询接口测试 | | | **接口URL** | /user/api/getUserInfo | |
| **用例综述** | 通过用户id进行查询，可以返回用户详细信息，包括邮箱地址、姓名、电话号码、性别、年级、注册时间等。 | | | | | |
| **测试步骤** | | | | | | |
| **序号** | **输入及操作** | | **期望结果与评估标准** | | **实测结果** | **通过与否** |
| 1 | 输入并发送username (String)。 | | 通过用户id，返回查询结果包括正确的对应详细用户信息。 | | 返回结果信息完整且正确。 | 通过 |
| **执行结果** | | 通过 | | **测试时间** | 2022/5/10 | |
| **结果截图** | |  | | | | |

### 4.2.2 JUnit接口测试

考虑到测试时会在终端打印大量的信息，为了方便查看输出的内容，在Utils/Myfunction.java中封装一个打印彩色信息的函数colorfulPrint，然后在test/java下存储各模块的测试代码，每个test文件都针对相应的Service模块中的函数进行了测试，如图4-1是对MessageService进行测试的截图。

可以看到所有的测试样例都通过的了测试。例子中的测试方式有两种，一种是针对返还数据类型为int的标志变量使用assertTrue()函数来检验输出的标志变量是否与预期相同，如果相同则通过检验；另一种是针对返还数据信息的函数，这里使用了之前封装好的colorfulPrint()函数打印出具体的数据内容，如终端中的绿色字样。考虑到在java中实体类对象、列表对象、map对象打印时都只会打印其内存地址，因此我们需要重定义其toString()方法，修改为打印它内部的成员变量。如图4-2是对实体类Message的toString()方法的重写。

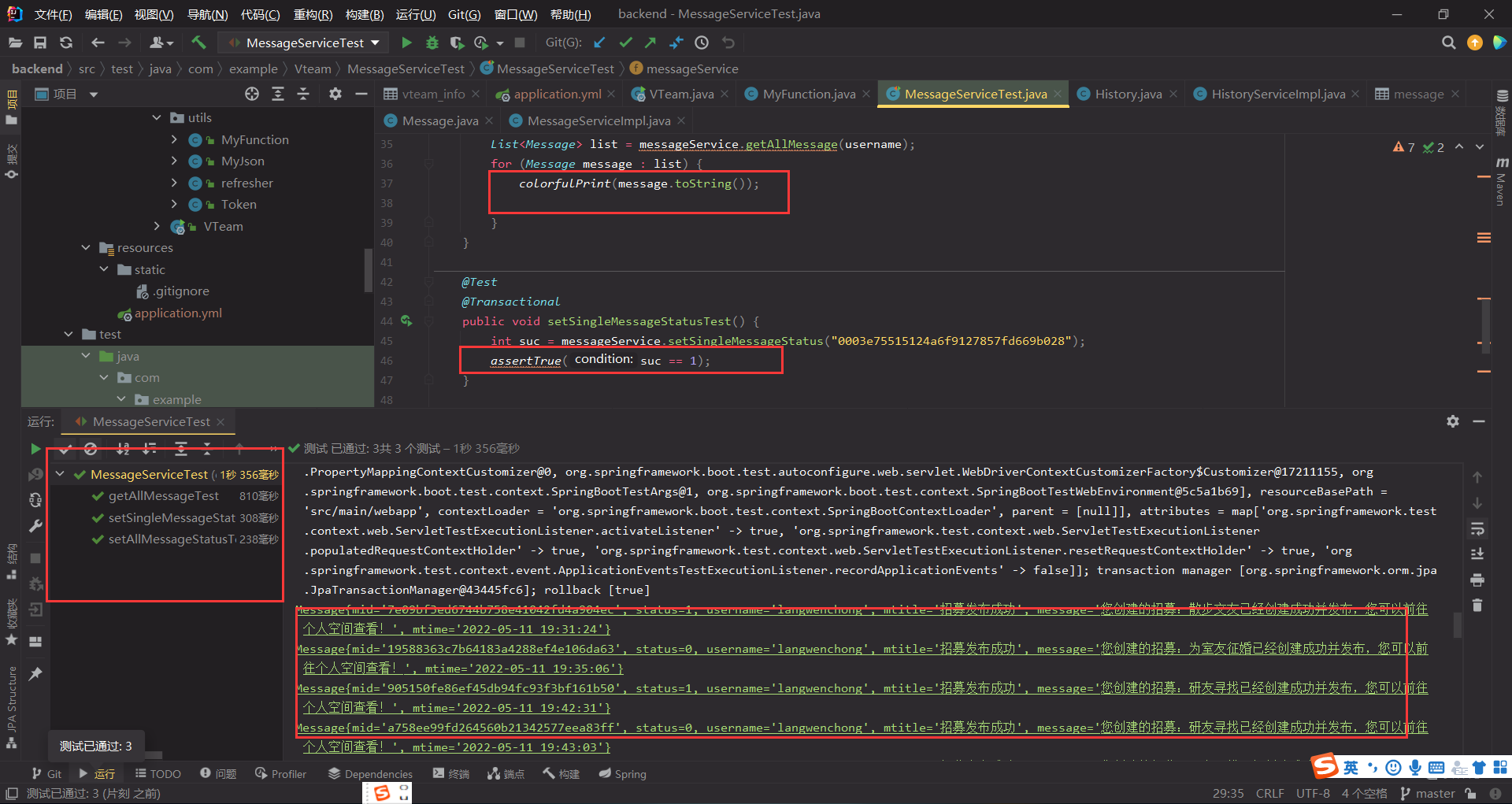


图4-1 MessageServer测试截图

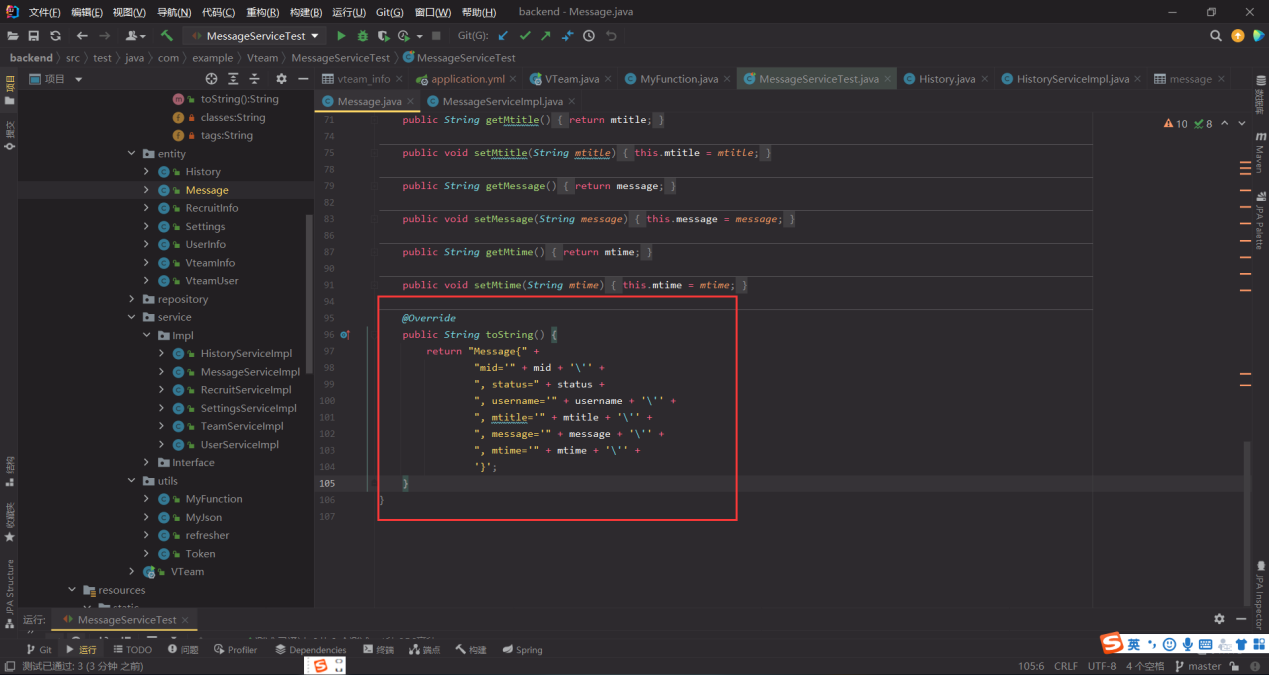


图4-2 Message的toString()方法的重写

# 系统部署

## 5.1 前端部署

### 5.1.1 修改资源路径

打开 config/index.js文件，然后在build环境下加入 assetsPublicPath:'./。

为了保证静态的资源（包括图片，elementui组件的icon图标等）正常加载显示，需要打开 build/utils.js文件，然后在 fallback下加入 publicPath:'../../'。

### 5.1.2 关闭跨域代理

在开发时，我们为了解决前后端跨域问题需要修改 proxyTable。但是部署到服务器后项目统一放置在一个文件件内，无需配置跨域代理。因此将 config/index.js文件的 proxyTable处的 '/api'后面的代码注释掉。

### 5.1.3 打包

新建终端，输入 npm run biuld打包项目，项目会生成一个index.html文件和一个用来存储静态依赖的static文件夹，统一放置在dist文件夹下。

至此vue项目的打包就完成了。打开index.html查看能否正常显示页面，如果可以正常显示即说明打包成功。若使用了elementui组件库，还需要检查icon图标是否也能够正常显示。

## 5.2 后端部署

本项目使用的数据库并不运行在本地，而是在远程服务器的docker容器里运行。因此无需本地维护数据库，只需要通过填写账户以及远程服务器端口号即可进行连接。

### 5.2.1 Idea连接远程服务器

在本地通过idea连接远程服务器，打开idea（不能是付费版），点击右侧数据库选择连接数据源，然后点击mysql数据源输入密码信息即可以连接到数据库。连接成功后就可以在右侧查看到所有数据库表内数据。

### 5.2.2 Application.yml文件的配置

为了能够本地调试和服务端运行配置，需要在resources下新建一个application.yml文件来配置一些选项，具体的配置见图5-1：

|  |
| --- |
| name: langwenchong  root: /home/Vteam/images/ #部署远程服务器时素材存放处  #root: src/main/resources/static/images/ #本地调试时素材存放处  server: #运行端口    port: 8088    servlet:      session:        timeout: 77m  # address: localhost #如果本地启动内网ip请填写ipconfig下对应的ip  spring: # 使用的是mysql驱动进行数据库连接    datasource:      driver-class-name: com.mysql.cj.jdbc.Driver      url: jdbc:mysql://1.117.169.85:3310/Vteam      username: root      password: 123456    jpa: # 使用jpa进行数据持久化存储      hibernate:        naming:          physical-strategy: org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl      open-in-view: true    servlet: # 定义最大传输文件容量      multipart:        max-file-size: 5MB        max-request-size: 5MB |

图5-1 application.yml的配置

### 5.1.3 项目打包

点击右侧的maven，然后在生命周期中选择install，打包完成。

### 5.1.4 Xftp部署项目并运行

打开xftp将jar包上传到/home/Vteam路径下，进入Vteam文件夹输入如下运行jar文件来启动后台服务：

nohup java -jar Vteam.jar >VteamLog.txt 2>&1 &

如果正常启动可以在日志输出文件VteamLog.txt中看到启动时的输出。

### 5.1.5 配置接口域名绑定以及第三方资源路径代理

虽然开启了后台的服务，但是此时如果前台要请求接口，需要输入ip+value的路径形式进行请求。为了方便，我们为这个服务绑定一个二级域名即vteamapi.coolchong.cn。还是参照上面去申请https格式的二级域名，然后打开nginx.conf文件配置nginx。然后再配置一个vteamresources.coolchong.cn二级域名来代理指向存储图片的路径。配置基本完成，接下来输入如下指令重启nginx：

Systemctl restart nginx.service

至此前后台项目的打包部署就完成了，可以通过https://vteam.coolchong.cn访问项目界面，然后界面会通过https://vteamapi.coolchong.cn调用后台服务，最后会使用https://vteamresources.coolchong.cn+图片相对路径，来获取存储在后台的静态资源。

# 项目总结

“北洋有约”校园组队系统通过建立一个智能化的校园组队平台，成功使校园组队智能化，方便了老师对队伍的管理，同时也让同学们能根据自己的兴趣爱好更自由地选择想要加入的队伍，获得了阶段性的成功。

# 参考文献

[1]秦不凡.大数据时代下软件工程技术的应用[J].农家参谋,2020(10):214.

[2]刘文霞,韩辉.多重控制信息失效下配电网电压波动性的评估[J].电力系统自动化,2020,44(09):69-80.

[3]王文龙,王淑超,徐光福.一种直流配电网保护控制软件平台的实现方法[J].供用电,2020,37(05):24-30.

[4]刘伟.ArcGIS软件在驷马山滁河四级站干渠工程地类量算中的应用[J].广西水利水电,2020(02):65-68.

[5]兰刚,甘涛,管庆,傅翀.软件工程专业综合设计课程过程管理的改革和探索[J].教育教学论坛,2020(18):196-199.

[6]董招生,咸云飞.大型桥梁结构智能健康监测系统集成技术分析[J].科技创新与应用,2020(12):146-147.

[7]陈修龙,姜帅,贾永皓,余神合.基于工程软件的“机械设计基础”课程设计实践[J].实验科学与技术,2020,18(02):53-5

# 致 谢

时光的流逝也许是客观的，然而流逝的快慢却纯是一种主观的感受。当自己终于可以从考研、找工作、毕业论文的压力下解脱出来，长长地吁出一口气时，我忽然间才意识到，原来四年已经过去，到了该告别的时候了。一念至此，竟有些恍惚，所谓白驹过隙、百代过客云云，想来便是这般惆怅了。

可是怅然之后，总要说些什么。大学四年，生活其实很简单，只是一些读书、写字和考试的周而复始。如果把这种单调的生活看作一场场循环的演出，提供那么我只是一个安静的演员。这篇毕业论文也称不上什么精彩的台词，只不过是这种循环演出即将告一段落时的谢幕词。但是无论多么蹩脚的演员，无论台下有多少观众，即使是只说给自己听，在他谢幕时也总要感激一些人，是这些人帮助他走上舞台，成功或者不那么成功地“演出”。

我在这里首先要感谢的是我的学位论文指导老师：张怡教授和李罡教授。这篇毕业论文从开题、资料查找、修改到最后定稿，如果没有他们的心血，尚不知以何等糟糕的面目出现。我很自豪有这样位老师，他们值得我感激和尊敬。

感谢和我共度四年美好大学生活的19级智能与计算学部的全体同学。感谢所有授课老师，你们使我终身受益。感谢所有关心、鼓励、支持我的家人。