

**毕 业 设 计**

**题 目 基于Spring Boot 的高可靠人事管理**

**系统设计与实现**

**英文题名 Design and implementation of highly reliable personnel management system based on Spring Boot**

**学生姓名**： **揭其樾** **申请学位门类**： **工学**

**学 号**： **201820800117**

**专 业**： **计算机科学与技术**

**学 院**：  **信息工程学院**

**指导教师：** **职称：** **xxxx**

**二〇二一年五月二十日**

**作 者 声 明**

本人以信誉郑重声明：所呈交的学位毕业设计（论文），是本人在指导教师指导下由本人独立撰写完成的，没有剽窃、抄袭、造假等违反道德、学术规范和其他侵权行为。文中引用他人的文献、数据、图件、资料均已明确标注出，不包含他人成果及为获得东华理工大学或其他教育机构的学位或证书而使用过的材料。对本设计（论文）的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本毕业设计（论文）引起的法律结果完全由本人承担。

本毕业设计（论文）成果归东华理工大学所有。

特此声明。

毕业设计（论文）作者（签字）：

签字日期： 年 月 日

本人声明：该学位论文是本人指导学生完成的研究成果，已经审阅过论文的全部内容，并能够保证题目、关键词、摘要部分中英文内容的一致性和准确性。

学位论文指导教师签名：

年 月 日

**基于Spring Boot 的高可靠人事管理**

**系统设计与实现**

揭其樾

Design and implementation of highly reliable personnel management system based on Spring Boot

二 〇 二 一 五 月 二 十 日

摘 要

人事管理是每一个企业都有的一项工作。怎样高效的管理人员流动，查看部门信息，修改工资套账等就显得尤为重要。一个好的系统能够简化这些问题。能够让企业更加高效的运转，降低人力物力成本。并且这个系统要做到易于上手，学习难度低，并且能记录所有管理员的操作，防止误操作。并且做好员工信息的保存，防止泄露。并且在以后公司规模扩大的时候，这个系统也能易于拓展，提供稳定的服务。

基于这个目标，我设计了这套系统。这套系统采用了Spring Boot + Vue.js 作为主要技术栈。配合MySQL数据库作为存储。以及Redis 作为缓存。加上Spring Security作为权限管理。并且引入消息中间件Kafka解耦邮件发送模块。在设计上采用了MVC模式，并且采用了前后端分离的模式让整个系统更加的解耦。

该系统主要包括以下六个模块：公共模块，员工资料，人事管理，工资管理，系统管理，统计管理。

关键字：人事管理系统； Spring Boot； 前后端分离； Vue.js； Kafka消息队列； MySQL数据库

# **ABSTRACT**

Personnel management is a job that every enterprise has. How to efficiently manage the flow of personnel, view department information, and modify salary accounting is particularly important. A good system can simplify these problems. It can make enterprises operate more efficiently and reduce labor and material costs. And this system should be easy to use, with low learning difficulty, and can record the operations of all administrators to prevent misoperation. And do a good job in the preservation of employee information to prevent leakage. And when the company expands in the future, this system can also be easily expanded to provide stable services.

Based on this goal, I designed this system. This system uses Spring Boot + Vue.js as the main technology stack. Cooperate with MySQL database as storage. and Redis as a cache. Plus Spring Security as permissions management. And introduce the message middleware Kafka to decouple the mail sending module. In the design, the MVC pattern is adopted, and the front-end and back-end separation pattern is adopted to make the whole system more decoupled.

The system mainly includes the following six modules: public module, employee data, personnel management, salary management, system management, statistics management.

**Key words:**  Personnel management system； Spring Boot； Front and back end separation； Vue.js； Kafka； MySQL Database

目 录

[**ABSTRACT** II](#_Toc101774767)

[第1章 绪 论 1](#_Toc101774768)

[1.1 研究背景 1](#_Toc101774769)

[1.2目的和意义 1](#_Toc101774770)

[1.3国内外研究现状 2](#_Toc101774771)

[1.4研究方法 2](#_Toc101774772)

[1.5 本文的组织结构 2](#_Toc101774773)

[第2章 实现技术 3](#_Toc101774774)

[2.1 Spring 3](#_Toc101774775)

[2.2 Spring Boot 3](#_Toc101774776)

[2.3 MySQL 3](#_Toc101774777)

[2.4 Spring Security 3](#_Toc101774778)

[2.5 Kafka 4](#_Toc101774779)

[2.6 Redis 4](#_Toc101774780)

[2.7 系统开发环境 5](#_Toc101774781)

[第3章 需求及可行性分析 6](#_Toc101774782)

[3.1 项目可行性分析 6](#_Toc101774783)

[3.2 项目的组成 7](#_Toc101774784)

[3.3 项目功能需求分析 7](#_Toc101774785)

[第4章 总体设计 36](#_Toc101774786)

[4.1 项目整体架构 36](#_Toc101774787)

[4.2 项目整体设计 36](#_Toc101774788)

[4.3 数据库设计 36](#_Toc101774789)

[第五章 系统详细设计 43](#_Toc101774790)

[5.1 项目结构 43](#_Toc101774791)

[5.2 通用模块以及全局异常处理 44](#_Toc101774792)

[5.1 员工模块实现 47](#_Toc101774793)

[5.2部门模块实现 49](#_Toc101774794)

[5.3 权限管理模块实现 52](#_Toc101774795)

[5.4 操作日志模块实现 53](#_Toc101774796)

[5.5 邮件模块实现 56](#_Toc101774797)

[第六章 系统测试 57](#_Toc101774798)

[6.1 系统测试 57](#_Toc101774799)

[结 论 59](#_Toc101774800)

[致 谢 61](#_Toc101774801)

[参考文献 61](#_Toc101774802)

## 第1章 绪 论

### 1.1 研究背景

本项目目的是实现机关和事业单位的人事管理信息系统。机关和事业单位的人事管理在政府工作中占有极其重要的地位，如何加强各个部门之间的协调和提高工作效率是人事管理的当务之急。通过人事管理系统人事组织部门能做到以人为中心，各部门之间使用共有的资源，即时通信，现时提高工作效率，简化繁琐的手工统计、信息汇总和工资业务大量人工工作，让人事组织和工资管理工作在人事组织相关的各部门之间活起来。

现在数据库等技术日渐成熟，对于数据的完整性做到了很好的保证。即使在一些极端情况下，比如说断电等都能很好的保证数据不丢失。并且内存数据库也得到了很多发展，这使得系统的性能可以得到很好的提升。并且现在信息安全也越来越受到重视。目前也有一些系统数据泄露的事件。

本项目就是在这样背景下提出的。本项目在技术方面采用了较为先进的 J2SE 信息技术。

### 1.2目的和意义

#### 1.2.1 课题目的

在以上的背景下，我们的目的是基于Java语言及其生态圈以及其他技术，比如Spring全家桶，数据库，缓存等来设计架构出一个可拓展，高性能的时时人事管理系统。对于员工的信息修改能够快速的被系统管理员查看到。

设计出界面美观，易于使用，能够应对未来一段时间内的用户增长所带来的系统压力。并且有很高的安全性。防止使用该系统的用户的信息被泄露。

#### 1.2.2 课题意义

该项目可以提高工作效率，让用户从繁杂的人事管理中解脱出来，专注于其它事情。对于用户来说，可以很好的做一些自定义的个性化操作。在系统上进行二次开发也是非常方便的。可以方便的添加一些功能模块。

对于该系统接入成本低。并且都是一些开箱即用的功能。还有一些方便的一键导入数据等功能。这使得企业等接入特别方便。

该项目也能学习国内外优秀的人事管理系统的设计思路，架构理念。在总结中能够得出一套可复用的软件架构设计方案。

### 1.3国内外研究现状

目前国内的公司如雨后春笋般涌现。特别是互联网公司，体量发展的很快。对现行系统的详细调查研究是了解系统需求和进行系统分析与设计的重要 基础工作，因此要对一般企业的人事信息管理部门的工作进行全面、细致而充分的调查研究。系统调研采取的主要方法是到单位的人事部门，通过询问、观察等活动去了解单位人事管理的主要流程工作。人事部门的主要工作内容包括:新员工加入时人事档案的建立，员工调动、辞职退休，还有员工信息的查询以及工资管理等方面。人事管理工作主要流程就是，首先把新员工建立档案，员工调动、 辞职、退休，员工信息查询、工资管理等.第二步是把上述工作中完成的工作记录存档。第三步是把各项工作记录分项做成统计报表向上级部门汇报。

并且现行系统结合了很多自动化的工具，比如自动报表生成，员工薪资变化折线图，一键导出员工资料等等。对于数据库等也有一键恢复等功能。信息管理向着自动化推进。

我们研究了国内外具有代表的欢雀管理系统。该系统具有远程招聘，人才数据库，员工花名册，电子合同，考勤管理，薪酬管理，绩效考核等功能。经过多年的发展，它的提供的服务已经趋于成熟。它的核心功能还是对信息的统计聚合。并且主要服务于公司。

根据调研，我们得出以下结论：

1. 目前大多数人事管理系统不能做到共用性。没有将某些服务抽取出来。对于不同的行业需要重新设计。
2. 系统大多采用现阶段流行技术。并且都考虑到了系统的拓展性以及可用性。
3. 系统都是有着操作难度低，上手快的特点。

### 1.4研究方法

通过查阅所用到的技术的相关文献，结合国内外开源项目对该技术的使用情况以及该技术的流行度。以及通过对每一种技术的历史调研，分析技术未来发展趋势，判断出最适合本项目的技术。进而得出项目的可行性以及架构设计。通过对不同数据量的系统运行速度进行压测，得到接口的响应速度变化趋势。判断在当前架构的前提下并在未来可能发展的体量下，系统是否能正常的运行。

### 1.5 本文的组织结构

本文系统全面的梳理和阐述了一个人事管理系统的设计与实现。对于本文每一部分的内容介绍如下：

第一章：绪论部分。该部分先从调查国内外的管理系统的实现开始，得到现阶段该项目的背景。以及确定本项目的研究目的和意义。

第二章：实现技术。该章节主要介绍了本项目中所使用到的各个技术，简单阐述了该项技术的优劣，以及为什么该项技术能够被使用在本项目中。

第三章：需求及可行性分析。本章节先从三个方面入手，分析项目的可行性。并最终得出项目可行的结论。最后再把本项目所有的需求列出来。以便后续按照需求来实现。

第四章：总体设计。本章节通过画项目的整体架构图来给出项目的全貌。以及从更高的角度来说明每一个模块的组成和架构，以及各个模块之间是怎么相互配合的。而且本章还会给出数据库表的设计，以及数据表的E-R图。

第五章：系统详细设计。本章节就会对上面提到过的每一个需求的具体实现做一个详细的说明。

第六章：系统测试。对于系统能够承受的qps进行压测。来熟悉系统的最大负载。并且对于接口进行一些集成测试，对于用户的非法输入进行测试。

## 第2章 实现技术

### 2.1 Spring

Spring是 Java生态圈一个重要的框架。它最重要的方面是Bean容器和AOP切面编程，事务管理。

IOC容器使得对象不再被开发人员手动创建，而是通过配置文件的形式，让框架管理对象的创建。并且能创建多种类型的对象。单例，或者是原型对象。这使得对象之间的关系不再耦合。并且能够将开发人员从复杂的对象创建中解救出来，使开发人员能够专注于业务逻辑的开发。

### 2.2 Spring Boot

Spring Boot是依托于Spring框架而诞生的一个框架。它简化了配置。实现了一些自动化的配置。并且拥有一些嵌入式的web容器。它无需使用XML文件的配置形式，而是转而使用Java代码的配置形式。这使得开发人员更加容易的看出配置意图。

### 2.3 MySQL

MySQL作为一个关系型数据库，它能够支持多种存储引擎。并且性能很好，而且免费。它使用C/C++编写，这保证它的源代码的可移植性。在本项目中会使用InnoDB存储引擎。它能够支持事务，MVCC等特性。这使得数据在存储和查询直接得到了一个很好的平衡。并且能够通过Undo Log，Bin log等保证数据的安全性。该数据库被很多OLTP系统所使用。

### 2.4 Spring Security

Spring Security作为Spring全家桶中的一员，能够很好的和Spring框架进行整合。它提供了认证和授权两大类功能。认证主要是判断用户身份的真实性，而授权是判断系统能够给予这个用户哪些权限。并且Spring Security提供了很多自定义的过滤器，这使得能够很方便的实现自己的认证功能。并且它支持多种身份验证模型，这些模型可以是第三方提供的，也可以是框架本身提供的。

### 2.5 Kafka

Kafka作为一个分布式的消息队列，能够有很好的拓展性。它主要可以在系统或应用程序之间构建可靠的用于传输实时数据的管道，消息队列功能。以及构建实时的流数据处理程序来变换或处理数据流，数据处理功能。我们可以通过Kafka让两个系统异步的通讯。并且它能够和大数据平台的一些时时处理框架进行整合。

Kafka整体架构如下：

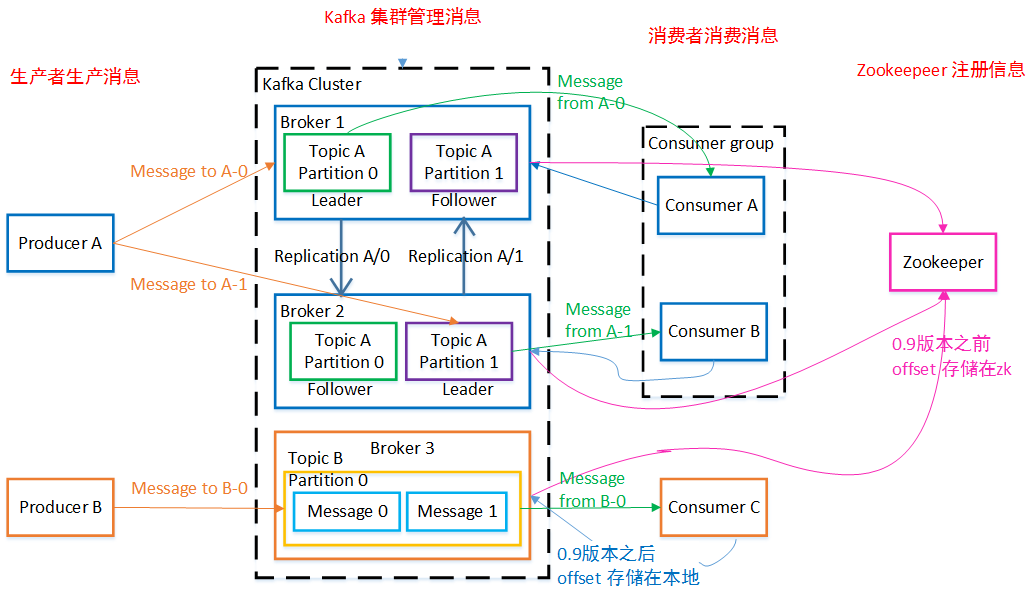


图2-1 Kafka的整体架构

### 2.6 Redis

Redis作为一个分布式的内存数据库，一般可以用作系统的缓存。这使得热点数据可以存放在内存中。所以数据的存取非常的快。并且Redis支持主从架构，可以支持横向拓展。对于数据量巨大的时候也可以做分片处理。并且它支持多种数据结构，并且自身能够通过RDB和AOF这两种机制来支持数据持久化到磁盘上面，虽然它是一个内存数据库但是在掉电的情况下，也能够最大限度的保证数据的安全性。并且有哨兵机制，能够自动进行故障转移。

并且Redis作为单线程的缓存，能够保证线程的安全性。并且基于Reactor模式使得并发度得到了很大的提高。

### 2.7 系统开发环境

#### 2.7.1 Java

该项目采用JDK1.8作为开发语言。Java作为一种表述力很强的编译型语言，以及非常活跃的生态圈非常适合用于工程项目中。

Java1.8引入了很多新的语言特性，这使得它是一个非常重要的版本。比如说：Optional 类已经加入了JDK1.8中，可以用来提前判断指针是否为空。并且引入了新的时间处理API，Date Time API − 加强对日期与时间的处理。还有一些函数式编程的接口，也引入了Lambda表达式。

#### 2.7.2 Intellij IDEA + WebStorm

采用JetBrains公司的IDEA和WebStorm进行开发。

#### 2.7.3 Windows 11

该项目在Windows 11系统下进行开发运行。未来可以部署在各个版本的Linux系统上。

#### 2.7.4 Tomcat

该项目使用Tomcat作为Web容器。

## 第3章 需求及可行性分析

### 3.1 项目可行性分析

#### 3.1.1 技术可行性

从目前流行的基于数据库开发的人事管理系统软件来看，用基于 J2SE 的架构开发 B/S 结构，在数据库方面采用 MySQL，无疑是这个应用中最合适的一种解决方案。Spring生态中，更新频率非常高，遇到的大部分问题在社区都有解决方案。所以不存在遇到问题找不到解决方案的情况。本系统对计算机的硬件环境有一定的要求，对计算机的操作系统、内存、主频、外设等都有最低要求，如果低于这个要求将影响到本系统的正常运行。

#### 3.1.2 经济可行性

机关和事业单位的人事信息管理目前还完全以传统的人工管理方式进行管 理，不仅耗时多，效率低下并且极易出现错误。由于认为失误而早成用形和无形的经济损失事件层出不穷，我们也无法估算出所造成损失的总额。而利用计算机来实现人事管理的以成为适应当今人事管理的方式。开发一套能满足人事信息管理的软件是十分必要的，实现人事管理的自动化，在减少由于认为失误而早成损失的同时，也可以使人事信息管理部门减少许多费用支出，如实现自动化管理后可以精简人员，减少工资支出等。由上述三方面的分析可以看出，本系统的开发时机已经成熟，从多种角度考虑开发此系统都是可行的，并且也是十分必要的。并且本系统可以选择分布式部署，也可以是单机系统，非常的灵活。

#### 3.1.3 运营可行性

由于本系统只是面向机关和事业单位内人事信息管理人员和在职人员开发的信息管理系统，尽管这些人员可能没有使用过类似的系统，但Windows的友好界面和本系统良好的安全性设置，可以使人事信息管理员在专业人员的指导帮助下迅速掌握系统的操作方法。同时，用户也可以根据我们编写的详细的“用户操作说明书”，来学习正确的操作本系统。即使由于用户的误操作使得数据库系统的数据丢失，我们也可以提前做好数据库的备份。在发生问题时能够及时的将数据进行恢复。并且对于本系统中所涉及的技术，如数据库，缓存以及消息中间件等都可以用分布式的方式进行部署，这也调高了系统的可运营性。

### 3.2 项目的组成

项目主要包括以下功能模块，包括管理员模块，权限模块，邮件模块，部门模块，员工模块，工资套账模块，系统管理模块，菜单模块。

### 3.3 项目功能需求分析

员工模块：员工录入，员工删除，搜索员工，导出员工 Excel 列表以及导入。

权限模块：采用 Spring Security 做权限控制。实现了 RBAC 模型。以接口为级别做了访问权限控制。并且可以动态修改管理员所拥有的权限。

菜单模块：根据管理员所拥有的权限来动态的展示其所能操作的菜单。

部门模块：部门能够添加多级部门，并且在前端显示的时候也能够以部门树的形式来进行展现。

邮件模块：在员工入职的时候，发送邮件。采用了 Kafka 来接收发送消息，并且结合了 Redis 保证了消息的可靠性，以及不被重复消费。

系统管理模块：通过 AOP 的方式记录管理员的操作记录，并且通过引入线程池来异步的将记录落盘到 MySQL 上。 同时，还需要利用 AOP 来做了接口防刷的功能。

管理员模块：超级管理员能够给予和调整其它管理员所拥有的权限。并且这个权限能够实时生效。

前端模块：整个前端采用Vue.js框架。结合ElementUI组件库进行搭建。并且使用Axios作为Http库进行网络请求的发送。

## 第4章 总体设计

### 4.1 项目整体架构

本项目整体架构图如下：



图4-1 项目架构图

### 4.2 项目整体设计

本项目采用了RESTful风格接口设计。采用前后端分离的方式进行开发。前端通过HTTP请求的方式与后端进行通讯。后端采用单体架构的方式。数据持久化方面选择了支持事务的MySQL数据库。并且使用了Redis作为缓存。

### 4.3 数据库设计

#### 4.3.1 数据表设计

hr表：该表存放着管理员的各项信息。Userface是管理员的头像。该头像填写的是一个网址，通过第三方的存储来存放头像。

表4-1 表 hr

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **小数点** | **注释** |
| id | int | 0 | 0 | hrID |
| name | varchar | 32 | 0 | 姓名 |
| phone | char | 11 | 0 | 手机号码 |
| telephone | varchar | 16 | 0 | 住宅电话 |
| address | varchar | 64 | 0 | 联系地址 |
| enabled | tinyint | 1 | 0 |  |
| username | varchar | 255 | 0 | 用户名 |
| password | varchar | 255 | 0 | 密码 |
| userface | varchar | 255 | 0 |  |
| remark | varchar | 255 | 0 |  |

Adjustsalary表：该表记录了员工的调薪记录。

表4-2 表 Adjustsalary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **小数点** | **注释** |
| id | int | 0 | 0 |  |
| eid | int | 0 | 0 |  |
| asDate | date | 0 | 0 | 调薪日期 |
| beforeSalary | int | 0 | 0 | 调前薪资 |
| afterSalary | int | 0 | 0 | 调后薪资 |
| reason | varchar | 255 | 0 | 调薪原因 |
| remark | varchar | 255 | 0 | 备注 |

Appraise表：该表记录了员工的考核记录。

表4.3 表 Appraise

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **小数点** | **注释** |
| id | int | 0 | 0 |  |
| eid | int | 0 | 0 |  |
| appDate | date | 0 | 0 | 考评日期 |
| appResult | varchar | 32 | 0 | 考评结果 |
| appContent | varchar | 255 | 0 | 考评内容 |
| remark | varchar | 255 | 0 | 备注 |

Department表：该表存放了部门的各项信息。

表4.4 表 Appraise

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **小数点** | **注释** |
| id | int | 0 | 0 | 部门id |
| name | varchar | 32 | 0 | 部门名称 |
| parentId | int | 0 | 0 | 父部门 |
| depPath | varchar | 255 | 0 | 部门路径 |
| enabled | tinyint | 1 | 0 | 是否启用 |
| isParent | tinyint | 1 | 0 | 是否是父部门 |

Employee表：员工表，存放了员工的一些信息

表4.5 表 Employee

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **小数点** | **注释** |
| id | int | 0 | 0 | 员工编号 |
| name | varchar | 10 | 0 | 员工姓名 |
| gender | char | 4 | 0 | 性别 |
| birthday | date | 0 | 0 | 出生日期 |
| idCard | char | 18 | 0 | 身份证号 |
| wedlock | enum | 0 | 0 | 婚姻状况 |
| nationId | int | 0 | 0 | 民族 |
| nativePlace | varchar | 20 | 0 | 籍贯 |
| politicId | int | 0 | 0 | 政治面貌 |
| email | varchar | 20 | 0 | 邮箱 |
| phone | varchar | 11 | 0 | 电话号码 |
| address | varchar | 64 | 0 | 联系地址 |
| departmentId | int | 0 | 0 | 所属部门 |
| jobLevelId | int | 0 | 0 | 职称ID |
| posId | int | 0 | 0 | 职位ID |
| engageForm | varchar | 8 | 0 | 聘用形式 |
| tiptopDegree | enum | 0 | 0 | 最高学历 |
| specialty | varchar | 32 | 0 | 所属专业 |
| school | varchar | 32 | 0 | 毕业院校 |
| beginDate | date | 0 | 0 | 入职日期 |
| workState | enum | 0 | 0 | 在职状态 |
| workID | char | 8 | 0 | 工号 |
| contractTerm | double | 0 | 0 | 合同期限 |
| conversionTime | date | 0 | 0 | 转正日期 |
| notWorkDate | date | 0 | 0 | 离职日期 |
| beginContract | date | 0 | 0 | 合同起始日期 |
| endContract | date | 0 | 0 | 合同终止日期 |
| workAge | int | 0 | 0 | 工龄 |

Employeetrain表：员工培训表。

表4.6 表 Employeetrain

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **小数点** | **注释** |
| id | int | 0 | 0 |  |
| eid | int | 0 | 0 | 员工编号 |
| trainDate | date | 0 | 0 | 培训日期 |
| trainContent | varchar | 255 | 0 | 培训内容 |
| remark | varchar | 255 | 0 | 备注 |

Empsalary表：员工薪资关联表。该表并没有直接记录员工的薪资。因为员工的薪资有不同等级的。所以通过一个关联表来将员工的id和员工的薪资等级进行关联。

表4.7 表 Empsalary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **小数点** | **注释** |
| id | int | 0 | 0 |  |
| eid | int | 0 | 0 | 员工id |
| sid | int | 0 | 0 | 薪资id |

Hr\_role表：该表是员工角色关联表。用来表示员工具有哪些角色。

表4.8 表 Hr\_role

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **小数点** | **注释** |
| id | int | 0 | 0 |  |
| hrid | int | 0 | 0 | hr的id |
| rid | int | 0 | 0 | 角色id |

Joblevel表：该表存放员工的职级。

表4.9 表 Joblevel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **小数点** | **注释** |
| id | int | 0 | 0 |  |
| name | varchar | 32 | 0 | 职称名称 |
| titleLevel | enum | 0 | 0 |  |
| createDate | timestamp | 0 | 0 |  |
| enabled | tinyint | 1 | 0 |  |

Mail\_send\_log表：该表用来记录邮件发送的失败状态，用于后续在发送失败的时候进行重试。在多次重试后任然失败的，会进行人工补发。

表4.10 表 Mail\_send\_log

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **注释** |
| msgId | varchar | 255 |  |
| empId | int | 0 |  |
| status | int | 0 | 0发送中，1发送成功，2发送失败 |
| routeKey | varchar | 255 |  |
| exchange | varchar | 255 |  |
| count | int | 0 | 重试次数 |
| tryTime | date | 0 | 第一次重试时间 |
| createTime | date | 0 |  |
| updateTime | date | 0 |  |

Menu表：该表将接口作为粒度，记录了该接口的一些信息。

表4.11 表 Menu

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **注释** |
| id | int | 0 |  |
| url | varchar | 64 | 该接口的url |
| path | varchar | 64 | 请求路径 |
| component | varchar | 64 | Vue中的组件名称 |
| name | varchar | 64 | 接口用途 |
| iconCls | varchar | 64 | 接口图标 |
| keepAlive | tinyint | 1 |  |
| requireAuth | tinyint | 1 |  |
| parentId | int | 0 |  |
| enabled | tinyint | 1 |  |

Menu\_role表：将menu表中的那些接口所需要的角色和角色表进行关联。

表4.12 表 Menu\_role

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **注释** |
| id | int | 0 |  |
| mid | int | 0 | 通过将mid和角色id关联。知道某一个角色有多少个mid。通过这个mid能够知道，用户可以操作的mean。menu表中的id和这个字段是关联的。 |
| rid | int | 0 | 角色id |

Msgcontent表：保存所有的通知信息。

表4.13 表 Msgcontent

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **注释** |
| id | int | 0 |  |
| title | varchar | 64 | 通知标题 |
| message | varchar | 255 | 通知信息 |
| createDate | timestamp | 0 | 创建时间 |

Nation表：该表存放了各个名族的名称以及id。

表4.14 表 Nation

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **注释** |
| id | int | 0 |  |
| name | varchar | 32 | 名族名称 |

Politicsstatus表：该表存放了所有的政治面貌。

表4.15 表 Politicsstatus

、

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **注释** |
| id | int | 0 |  |
| name | varchar | 32 |  |

Position表：职位表。

表4.16 表 Position

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **注释** |
| id | int | 0 |  |
| name | varchar | 32 | 职位名称 |
| createDate | timestamp | 0 |  |
| enabled | tinyint | 1 |  |

Role表：角色表。

表4.17 表 Role

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **注释** |
| id | int | 0 |  |
| name | varchar | 64 | 角色名称 |
| nameZh | varchar | 64 |  |

Salary表：薪资等级表。

表4.18 表 Salary

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **注释** |
| id | int | 0 |  |
| basicSalary | int | 0 | 基本工资 |
| bonus | int | 0 | 奖金 |
| lunchSalary | int | 0 | 午餐补助 |
| trafficSalary | int | 0 | 交通补助 |
| allSalary | int | 0 | 应发工资 |
| pensionBase | int | 0 | 养老金基数 |
| pensionPer | float | 0 | 养老金比率 |
| createDate | timestamp | 0 | 启用时间 |
| medicalBase | int | 0 | 医疗基数 |
| medicalPer | float | 0 | 医疗保险比率 |
| accumulationFundBase | int | 0 | 公积金基数 |
| accumulationFundPer | float | 0 | 公积金比率 |
| name | varchar | 32 |  |

Sys\_oper\_log表：管理员操作日志表。

表4.19 表 Sys\_oper\_log

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名称** | **字段类型** | **字段长度** | **注释** |
| oper\_id | bigint | 0 | 日志主键 |
| title | varchar | 50 | 模块标题 |
| business\_type | int | 0 | 业务类型（0其它 1新增 2修改 3删除） |
| method | varchar | 100 | 方法名称 |
| request\_method | varchar | 10 | 请求方式 |
| operator\_type | int | 0 | 操作类别（0其它 1后台用户 2手机端用户） |
| oper\_name | varchar | 50 | 操作人员 |
| dept\_name | varchar | 50 | 部门名称 |
| oper\_url | varchar | 255 | 请求URL |
| oper\_ip | varchar | 50 | 主机地址 |
| oper\_location | varchar | 255 | 操作地点 |
| oper\_param | varchar | 2000 | 请求参数 |
| json\_result | varchar | 2000 | 返回参数 |
| status | int | 0 | 操作状态（0正常 1异常） |
| error\_msg | varchar | 2000 | 错误消息 |
| oper\_time | datetime | 0 | 操作时间 |

#### 4.3.2 E-R图

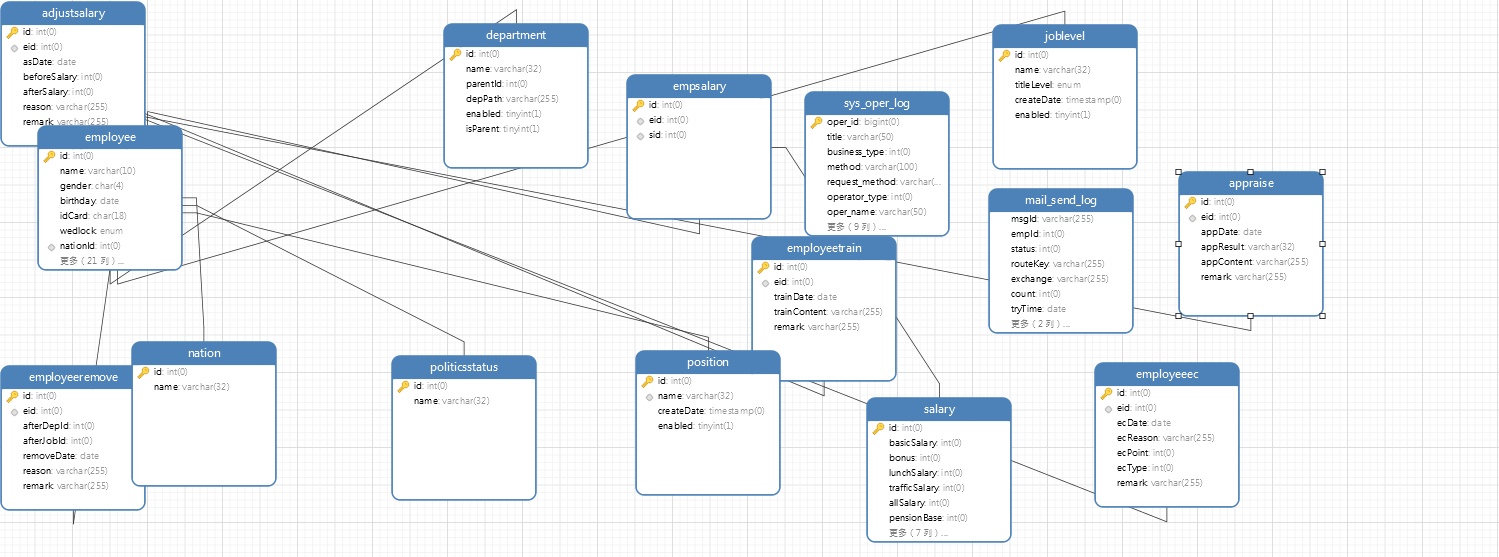
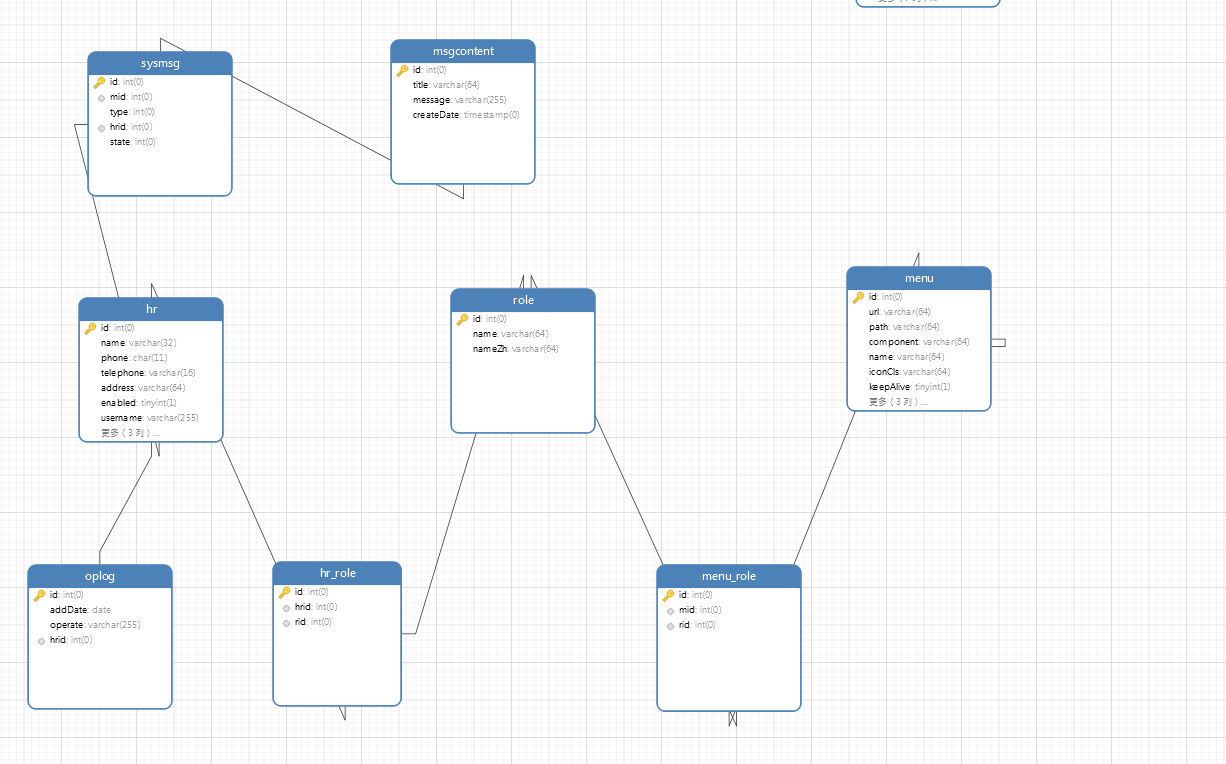


图4-2 E-R图

## 第五章 系统详细设计

### 5.1 项目结构

本项目采用Maven作为包管理工具。整体的项目结构如下图所示：

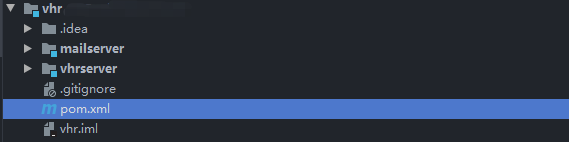


图5-1 项目结构

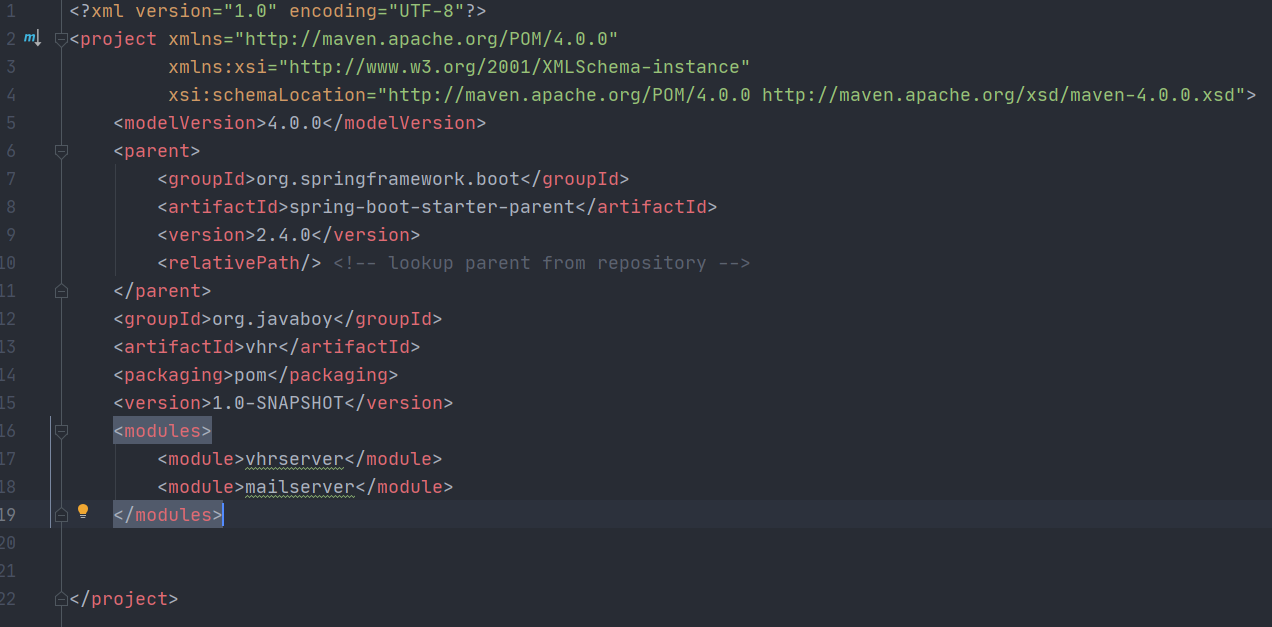
整体上可以分为mailserver模块和vhrserver模块。他们都是vhr模块下面的子模块。在vhr模块下的pom.xml文件中可以看到，它标明了vhr模块下面有哪些子模块。

图5-2 pom文件

在vhrserver模块下的包结构如下图所示：

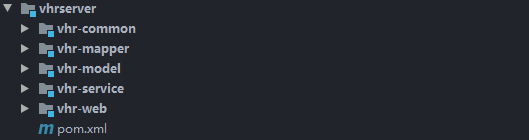


图5-3 模块结构

现在对每一个模块做简要说明：

Vhr-common：主要是放置一些被其它模块引入的共用的工具类。

Vhr-mapper：主要是放置Mybatis框架相关的接口文件以及相应的xml文件。

Vhr-model：放置项目中所有的实体类。

Vhr-service：作为MVC架构的service层。

Vhr-web：定义被前端请求的接口，以及AOP切面的一些定义。

### 5.2 通用模块以及全局异常处理

在系统中有一些共用的模块需要被其它的模块所依赖。比如一些工具类，以及实体类等。本项目中的工具类如下：

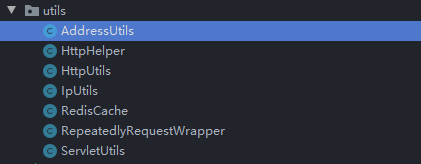


图5-4 异常处理

HttpHelper类：该类对使用Java语言发送HTTP请求进行了封装。并且对请求结果进行了读取。可以发送GET，POST，SSLPOST等类型的请求。

AddressUtils类：ip地址查询类。

RedisCache类：对Redis的一些操作进行了封装。包括设置key/value，设置过期时间等。

对于要返回给前端的结构，项目中定义了一个通用的类来封装接口操作结果，并由序列化框架将该类进行序列化。

RespPageBean类：该类是用在后端有很多数据要返回的时候，需要通过分页来返回数据。



图5-5 数据对象Bean

在该类中total字段表示符合条件的数据有多少项。

Data字段是该页的数据。

RestBean类：该类定义了如下属性：

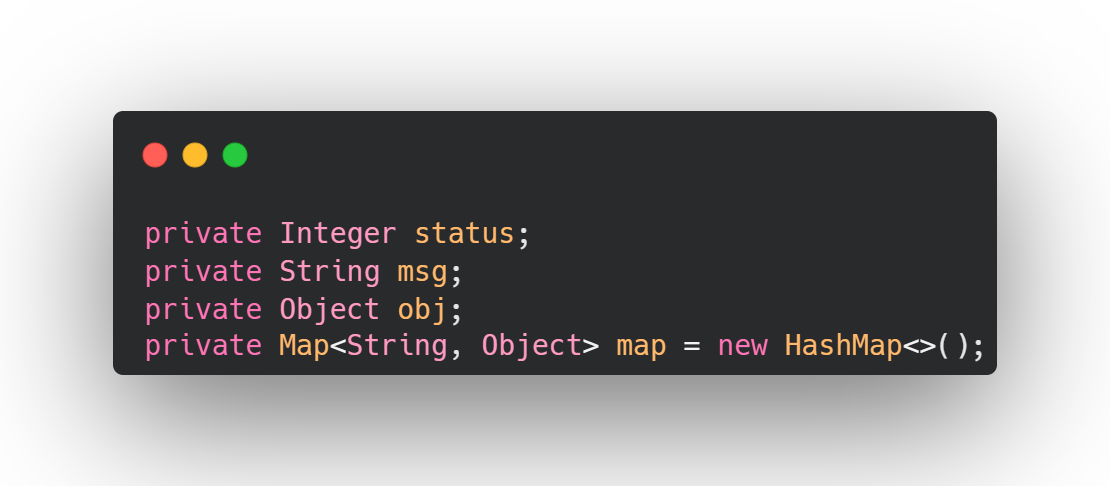


图5-6 返回结果Bean

在status字段中保存本次操作的状态码。这些状态码可以与HTTP状态码对应起来也可以自定义一些状态码，只要和前端约定好某种状态码的特定含义就行。此处的status字段并不表示HTTP状态码，正常来说HTTP状态码在请求成功的时候会是200。此处的状态码是项目中自己定义的状态码。在前端也有相应的判断，对于不同的返回的status给用户不同的提示信息。代码如下：

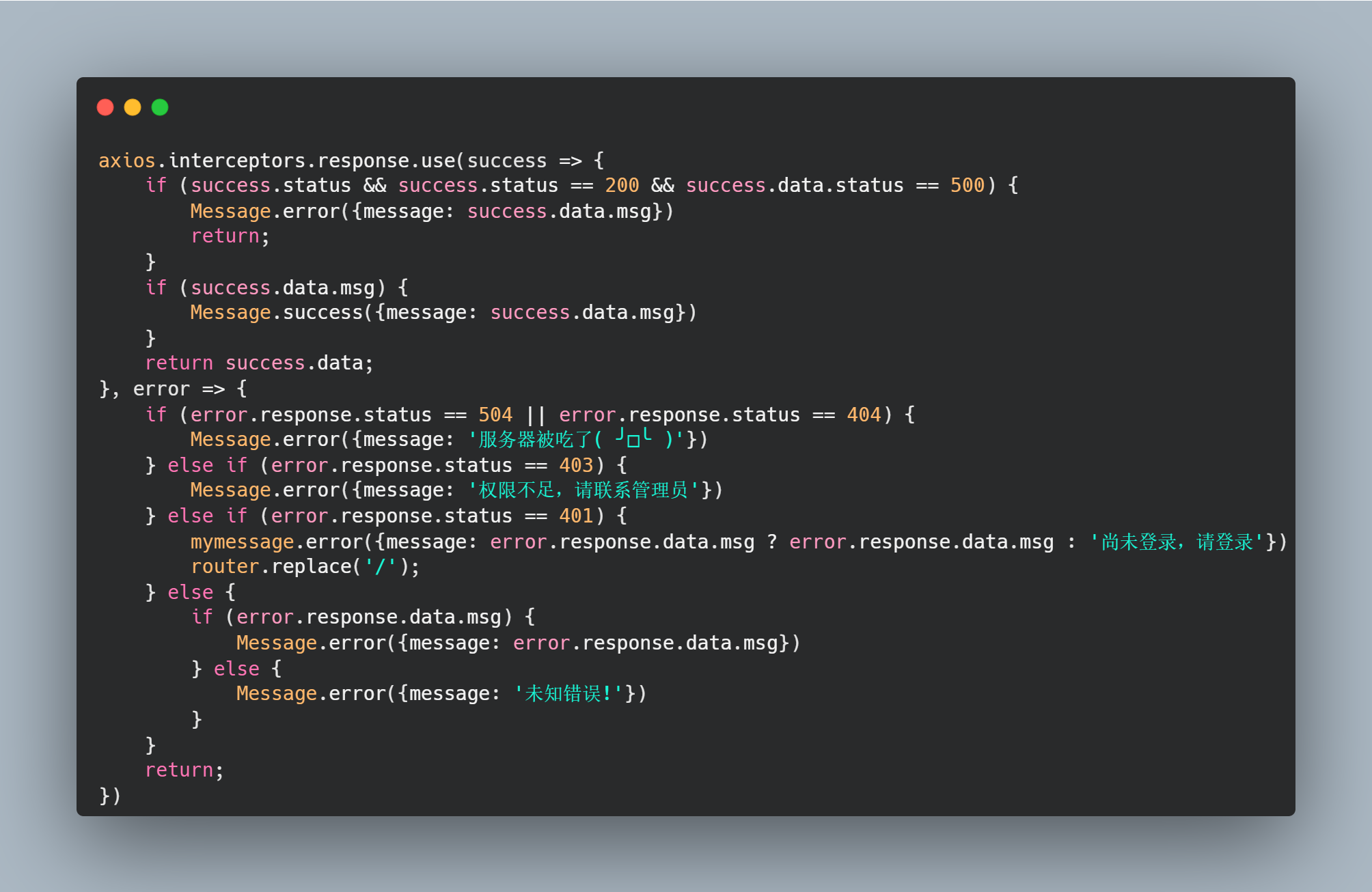


图5-7 前端拦截器

在msg字段中放置提示信息。在obj字段中方式要给前端传回的数据。对于key/value类型的数据可以放在map字段中。

本项目中的全局异常处理类如下：

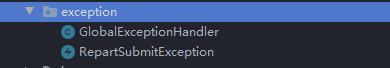


图5-8 全局异常处理

在项目开发中，需要抛出各种各样的异常。对于每一个异常可以向上抛出，也可以在抛出的时候捕获。但是如果在业务代码中耦合着各种各样的异常处理代码，那么就会降低程序的可阅读性。所以，一个全局的异常处理就显得非常重要了。

SpringBoot框架里面，将@ControllerAdvice标注在类上然后配合使用@ExceptionHandler就能够配置开启全局异常处理。@ExceptionHandle的作用是捕捉特定的异常。比如说捕获了特定的异常之后就可以给前端返回特定的信息来提示用户。对于用户输入错误的类型可以返回给用户错误提示。对于服务器内部的错误可以给用户提示服务暂时不可用。并且对于一些重要的服务，还可以做一些告警功能来提示开发人员项目出错了。

在本项目中捕获了两种类型的异常SQLException和RepartSubmitException。该部分的代码如下：



图5-9 异常处理

在发生异常时通过将RestBean的status字段设置为500再回传给前端。

### 5.1 员工模块实现

#### 5.1.1 员工添加

后台部分：

对于员工添加模块，如下图所示：



图5-10 员工添加

首先需要填入员工的各项信息，并且在添加的时候在前端先进行一遍参数校验。比如说邮箱。前端会检测鼠标光标移出的动作，当光标移出之后将会捕获该输入框内的内容，并进行正则表达式的判断。如果符合邮箱的格式就不会提示用户，否则提示用户邮箱格式不正确。由于我们在数据库中还需要存放一个合同期限，这个期限是由前端提交数据之后进行计算的。并且在Mybatis中开启了主键回填的机制。所以能够在添加成功之后再后端代码中得到该员工的主键id。这个id最终要填写到发往消息队列的消息中。由于在员工添加成功之后会给员工发送一封欢迎邮件，我的实现方式是在引入了消息队列。员工添加成功，就往消息队列里面发送消息，然后在另外一共Spring Boot的工程中对这个消息队列进行监听。并消费消息队列中的消息。由于这个是员工添加模块，所以具体的邮件发送以及一些幂等性的保证放到后面的邮件模块讲解。

在添加成功一个员工之后Mybatis会返回一个被影响的行数。通过这个行数就能够判断出员工是否添加成功。如果被影响的行数小于1就会返回添加失败的RestBean。

前端部分：

在添加员工按钮上面绑定一个点击事件。在单击了添加员工之后首先初始化一个emp对象。这个emp对象里面就对应这后端的employee对象。在这个对象中的一些字段是有初始值的。像职位，名族，职称这些基本不变的字段就给了一个初始值。并且会向后端请求获取最大的员工编号保存到刚刚创建的emp对象中。因为员工工号是由一个自增字段来决定的，而不是员工自己选择的。然后在后续将emp对象发送到后端去保存员工。

对于民族，职称，等不变的数据，请求之后加载到浏览器内存中。initData函数会在初次加载该页面的时候调用。在此函数中去请求后端的不变数据，然后存入前端浏览器的sessionStorage中。

#### 5.1.2 员工删除

删除员工是通过在前端将员工id放在url中，请求后端删除员工的接口，后端从url中提取出员工id，再去数据库中进行员工的删除操作。由于员工还有薪资套账，所以在删除员工的时候需要同时去两个表中删除数据。所以这是一个事务操作。所以需要在方法上标注Spring提供的@Transactional()注解。

#### 5.1.3 员工搜索

对于搜索的实现有两种方式，一种是高级搜索，一种是普通搜索。

高级搜索能够同时指定多个条件，而普通搜索只能通过员工姓名进行搜索。具体如下图：

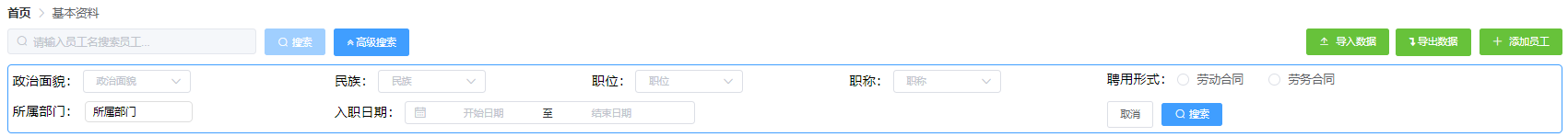


图5-11 员工搜索框

后端实现：

高级搜索和普通搜索都是使用的是同一个接口，只不过参数有所不同。如果是高级搜索会将条件中的信息填入employee对象中，然后后端通过Mybatis的条件if语句判断来拼接SQL语句，再去数据库中查询员工。在默认情况下会对数据进行分页返回。因为有可能根据条件搜索出来很多员工。

前端实现：

在前端为了实现管理员每打一个字就能时时的进行显示搜索结果，于是在搜索框中添加了监听器。只要搜索框中的文字发生变化，就立马发送查询请求到后端。代码如下所示：



图5-12 前端搜索框

#### 5.1.4 员工导入和导出

本项目支持通过xls格式的数据的导入和导出。在点击导出数据按钮之后，会发送导出数据的请求到后端。由于导出数据默认是将所有数据导出，所有后端会先在数据库中将所有的数据查出。然后通过字节流的形式返回给前端。该功能的实现是通过apache的poi第三方库进行实现的。导入数据也是先通过前端将xls格式的文件上传到后端，后端进行反序列化之后得到employee对象的数组。然后通过批量添加的方式存放到数据库中。

#### 5.1.5 分页的实现

员工部分大多数情况下都面临着数据量大的问题。所以会要进行分页操作。通过MySQL的limit关键字做的分页。Limit的第一个参数作为分页的起始位置，第二个参数表示从起始位置开始要返回多少数据。

### 5.2部门模块实现

由于员工和部门之间有一对一的关系，在MySQL中这两张表实现了外键的关系。Employee表中的departmentid字段是department表中的主键。

在一个顶级部门下面可能有多个子部门，而每一个子部门有可能有很多个子部门，于是这是一个树形的结构。想要高效的操作这些部门，就得设计一种合理的数据结构来存放。由于该项目使用的是关系型数据库，所以考虑怎样用行来将部门的关系存放。

首先，每一个部门都有一个唯一的id，这个id是部门表的主键。可以将最顶级的部门看做是一棵树的根节点。每一个子部门是这棵树的子节点。然后通过一个从根节点到本部门的id的最短连线作为一个字符串，存放到部门表中的deppath字段中。这样就能很快的得出一个部门的父部门，以及父部门的父部门是什么。而且通过一个isParent字段来标识这个节点是否含有子节点。这样就能快速判断该部门下面是否有子部门。并且通过enabled字段来标识当前部门是否启用。在后续如果要临时禁用一个部门的话就不用将数据表中的数据进行删除，可以进行软删除，也就是将enabled字段置位0。下面说下这个结构怎样被构建出来的。

#### 5.2.1 部门添加：

在前端可以选择在哪个部门下面添加部门，被添加的部门可以加到该部门下面：



图5-13 部门添加

在点击添加部门之后，前端会向后端发送要在哪个部门下面添加，也就是带上parentId。前端的请求荷载如下：

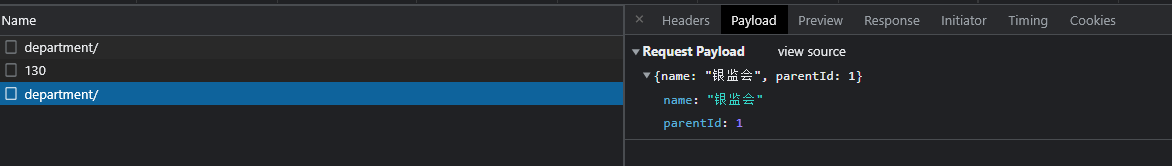


图5-14 前端请求

在后端接收到添加请求之后，会调用一个MySQL中的存储过程。

存储过程的优点主要包括以下几点：

第一点，性能提高。这是相对于不适用存储过程来说的，因为存储过程在创建的时候就编译好了，而后每次调用都不会再次编译，这相对于传统的SQL语句中每次调用都需要编译的情况来说，性能提高了何止一点两点。

第二点，重用性强。存储过程使用名字即可使用，也就是传说中的“一次编写，随便调用”。这样不仅提高了重用性，还减少了出错的几率，也会加快开发速度，可以说是一件非常好的事情。

第三点，减少网络流量。这一点对于小数据量的时候一般体现不出来，那么当数据量较大的时候，我们会发现由于使用存储过程比使用SQL语句会使用更少的字节数，因此它会降低传输的数据量。

第四点，安全性提高。由于存储过程也可以使用权限控制，而且参数化的存储过程可以防止SQL注入攻击，也在一定程度上保证了安全性。

第五点，灵活性增强。由于存储过程可以使用流程控制语句来编写，导致它有着很强的灵活性，可以根据实际情况来执行不同的SQL语句，而不是只能单纯的简单的执行命令。而且该存储过程还可以修改其逻辑而其他部分不用改变，也就是说，我们的表的结构改变了，我们只需要修改相应的存储过程即可，我们的Java程序不需要改变。

缺点：  
第一点，工作量加大。这里并不是说我们把程序该做的事让mysql去做不好，而是mysql本身并没有很像样的IDE来开发我们的存储过程，我们很多时候还是需要手写，这样就会比较麻烦，而且存储过程的调试也是一个问题，没有很像样的调试工具。

第二点，优势不明显。运行速度上，对于大多数的语句缓存来说，编译sql的时间开销并不是很大，但是执行存储过程还需要检查权限等一些其他开销，所以，对于很简单的sql，存储过程并没有很大优势。

所以，对于部门的添加来说并不是一条SQL语句就能完成的，存储过程在这里能发挥它的长处。于是选择了存储过程。

该存储过程如下：

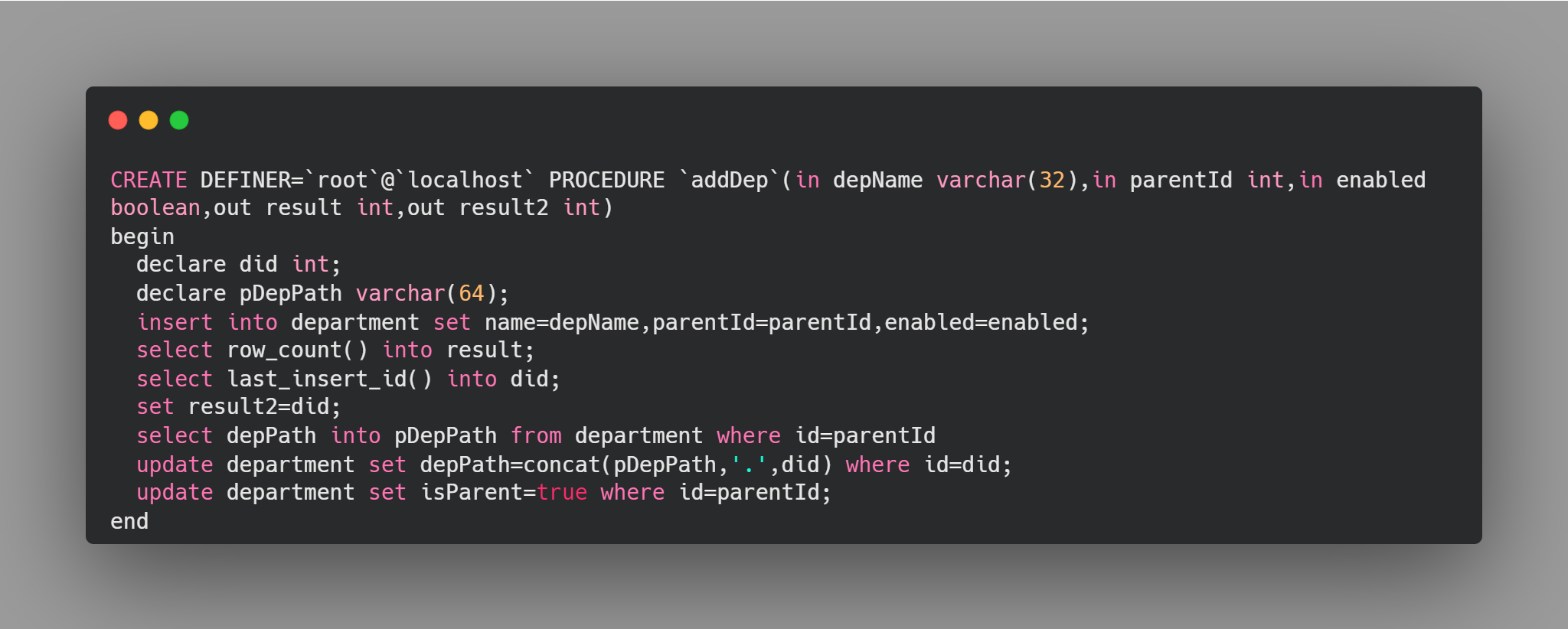


图5-15 存储过程

首先定义两个变量，did作为将要插入的部门的id，pDepPath变量作为父部门的depPath列的值。由于parentId是前端传过来的参数，所以先通过parentId查找到该id所对应的部门。进而赋值刚开始创建的两个变量，通过MySQL中的concat函数对路径结果进行拼接。得到新插入的部门的depPath。最后再将结果更新到表中，并返回最新插入的部门的id。

#### 5.2.2 部门删除：

在删除部门的时候将部门id传入后端。后端也是调用一个存储过程进行删除。删除部门的时候需要考虑两个点：

第一：该部门下是否有子部门。

第二：该部门或者其子部门下是否有员工。

只有当这两者都不存在的时候，才能进行删除操作。同时，如果违反了这两者中的某一项的话就需要给予前端清晰的提示，告诉管理员是哪种错误导致部门不能删除。所以在存储过程中如果发现该部门下有子部门，那么就会返回result为-2,。如果该部门或者其子部门下有员工则将result赋值为-1。然后在后端中进行判断，并返回不同的结果给前端。并且在删除部门之后还需要判断被删除部门的父部门是否还有子部门，如果没有子部门的话，需要将它的isParent字段设置为0。

该过程如下：

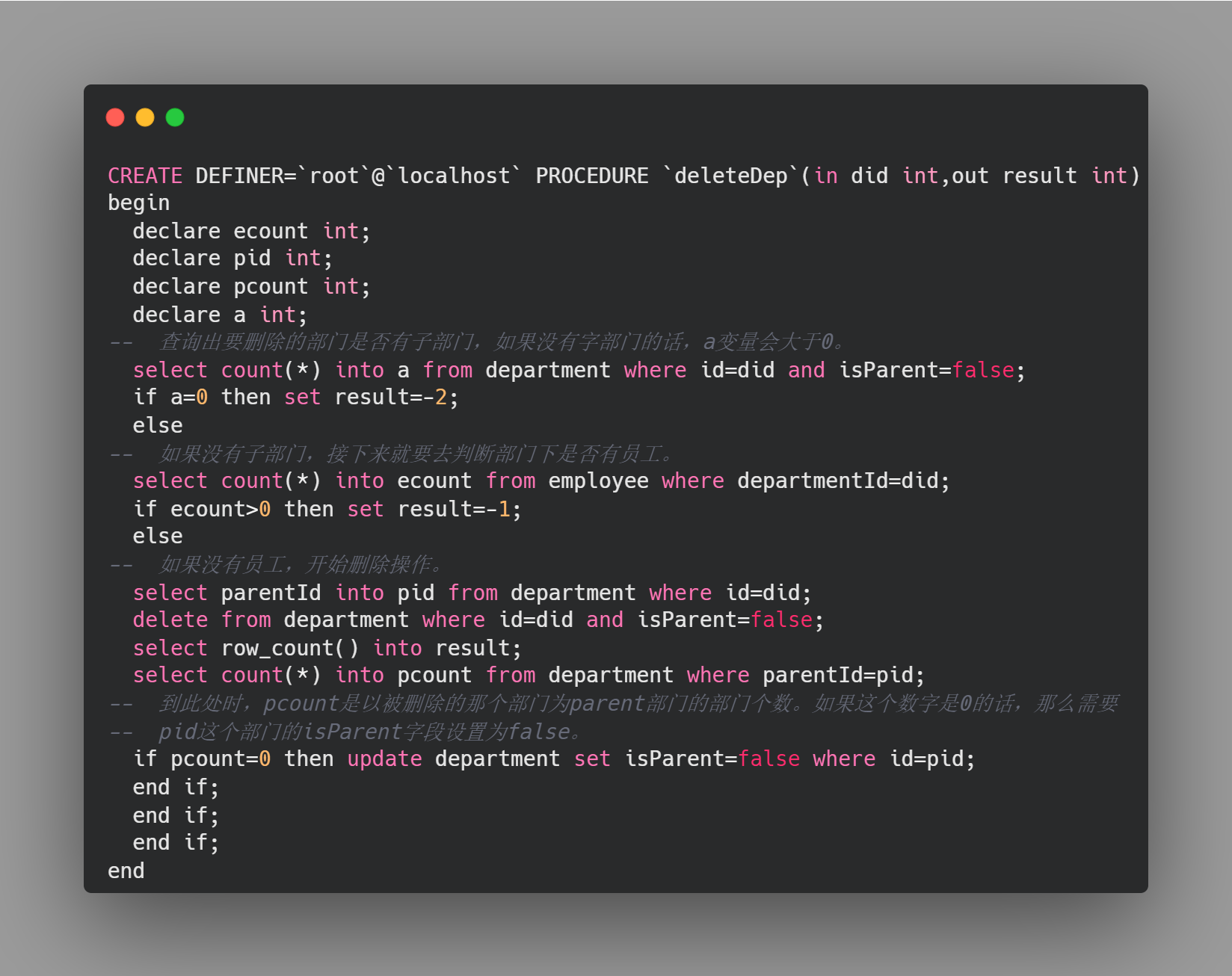


图5-16 存储部门

### 5.3 权限管理模块实现

首先说下本项目用到的权限管理模型是RBAC模型。RBAC（Role-based access control）是一种以角色为基础的访问控制（Role-based access control，RBAC），它是一种较新且广为使用的权限控制机制，这种机制不是直接给用户赋予权限，而是将权限赋予角色。

RBAC 权限模型将用户按角色进行归类，通过用户的角色来确定用户对某项资源是否具备操作权限。RBAC 简化了用户与权限的管理，它将用户与角色关联、角色与权限关联、权限与资源关联，这种模式使得用户的授权管理变得非常简单和易于维护。

在本项目中使用了五张表来实现这个权限管理模型。分别是User表，代表了管理员。Role角色表。Menu资源表。User\_role用户角色关联表。通过这张表能够根据用户id查询出该用户所拥有的角色。Role\_menu表角色资源关联表。这张表表示操作资源所需要的角色。本项目中将接口抽象为一种资源。

首先，我们在数据库中有资源menu表。这个表中有url地址这一栏。那么，这一栏是用户在前端发送的接口地址。就是说，接口地址的设计一定要遵从这个ant风格的地址。那么，我们在后端接收到一个前端请求的接口地址之后，我们将这个接口地址与数据库中menu表中的url这一栏进行比较，看看符合哪个ant风格的接口地址。从而我们能够得到menu的id。我们还有一个menu\_role表。刚刚我们已经获得了menu的id，那么我们通过这个menu\_role表就可以得到这个menu所需要的role（角色）。得到所需要的角色之后，我们就去拿到已经登录的用户的角色，看看这个已经登录的用户的角色是否在需要的角色里面。如果在的话，那么表明，这个用户有这个角色，也就是有这个权限能够访问这个接口。从而可以放行。

核心代码如下：



图5-17 权限模块

这里每一次对于接口的操作都会进行权限的判断，所以如果每一次都需要去数据库中进行查询的话效率会非常的查，所以这里引入了Redis作为Cache。也就是将所有的Menu资源所需要的角色都查询出来。存到内存中。

### 5.4 操作日志模块实现

该模块的主要作用就是记录管理员做过哪些操作。本模块在管理系统中也起着非常重要的作用。当管理员进行误操作的时候能够找到是哪个管理员进行操作的。该功能记录如下：

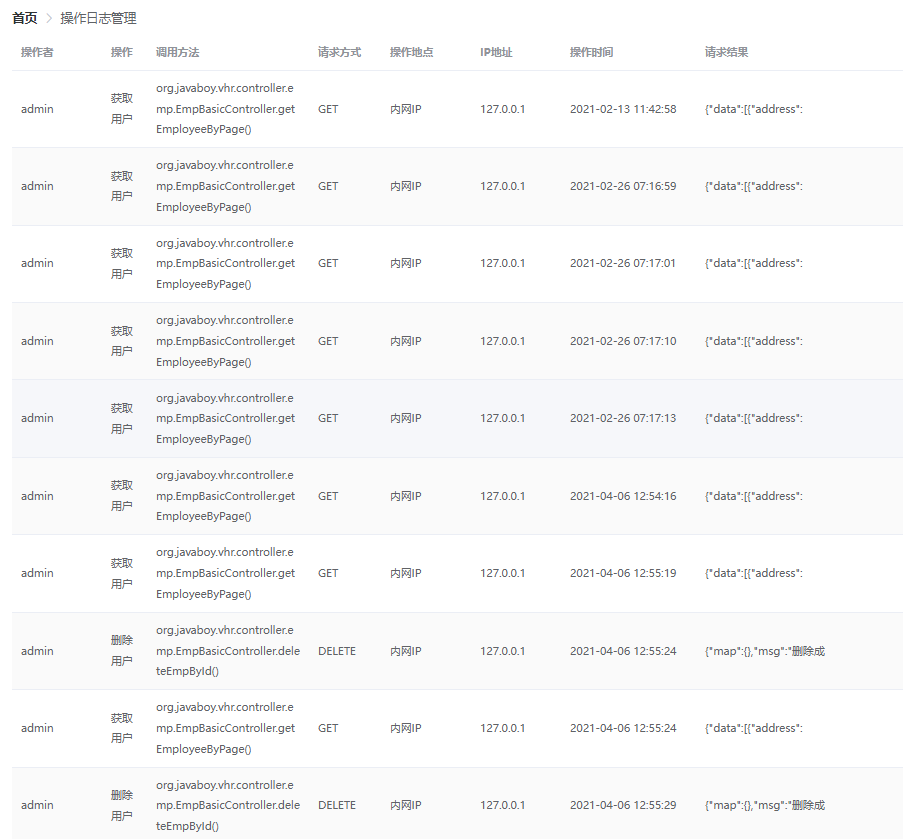


图5-18 日志

后端实现：

由于记录日志只会在某些关键方法中才会记录。所以这里的实现方式是通过自定义注解，并且通过AOP切面的方式来实现的。这种实现方式能够特别灵活的对要进行切面的方法进行配置。当某个时候不需要进行日志记录的话，只需要很小的代价就能够将日志记录的功能进行移除。具体实现如下；

首先自定义注解。

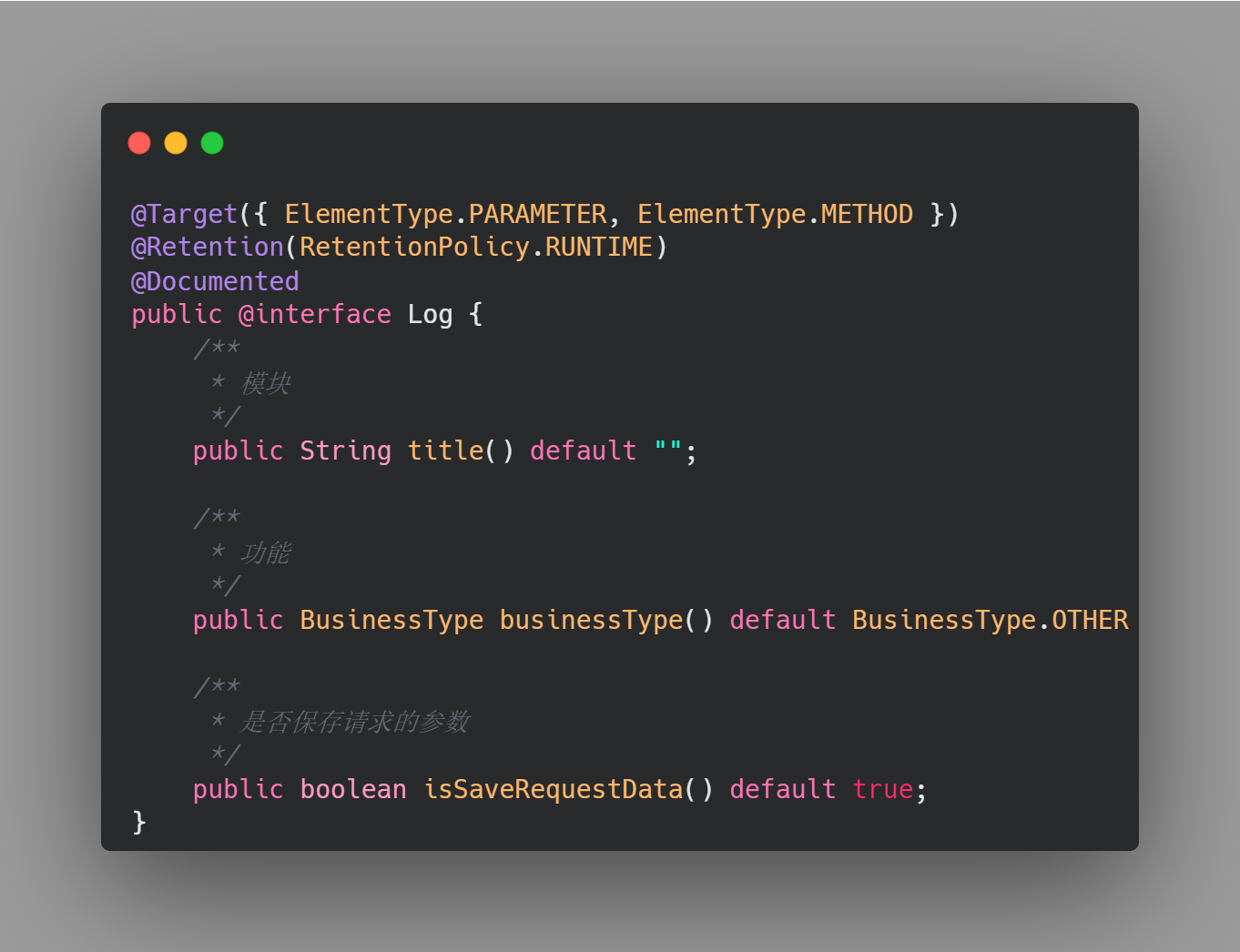


图5-19 日志类

该注解能够标注在方法上面。对于要进行日志记录的方法，就将这个注解标注在上面，并且声明好这个方法的功能。和HTTP请求的类型。如下所示：



图5-20 日志注解

对于删除用户这个敏感操作进行了日志记录。于是在deleteEmpById方法上面标注了@log注解。于是在运行时就能够通过反射来获取到这个注解的信息。并且通过Aop来进行切点表达式的配置，使得Spring容器能够在初始化的时候注入代理类。

在切点表达式中配置了只要是标注了@log注解的一律在运行时做一个切面。那么在运行时就能够在方法被调用的前后进行一些自定义的操作。比如说记录方法运行时间，或者是获取到操作员ip地址等。在获取到这些信息之后需要将这些信息进行落库。落库的流程由于不能影响主流程的进度，因此我采用了提交任务到线程池来异步的进行记录日志的操作。这样的话即使在记录日志的时候抛出了异常，或者是由于数据库在写磁盘等等原因让线程出现卡顿，也不会影响到主流程的响应速度。

对于线程池的参数设置，我将核心线程数设置为了5，最大线程数设置为了15。并且设置了一个有界的队列，长度为15。并且将拒绝策略设置为了CallerRunsPolicy。这样做的目的是为了不让应用启动太多的线程，因为在Java中，一个线程是对应这一个内核线程的。这就意味着上下文切换以及恢复现场需要CPU消耗。

### 5.5 邮件模块实现

在员工成功入职之后，会发送一封欢迎邮件。这个的流程是先前端发送添加员工的请求之后后端在成功将员工的信息插入数据库之后，就会往消息队列中投递员工入职的消息，然后在另外一个Spring Boot的工程中订阅该消息队列topic中的消息。每当收到一封消息就按照投递到队列中消息标注的邮箱发送一封邮件。当然这会涉及到消息的幂等以及消息丢失如何恢复以及重复消费等问题。这在下文中会讲述解决方案。

另外一种实现方式是将发送邮件的动作交给线程池来执行，但是这种方法会让系统中邮件发送服务和员工添加模块耦合。所以选用了消息队列来进行解耦。

针对上面提到的问题，在本项目中有以下的解决方案。

1. 消息丢失如何避免？

本项目中将acks的值设置为all。acks是Kafka中很重要的一个参数。由于Kafka是一个主从架构，有Leader节点也有follower节点，当acks参数设置为1的时候只要Leader节点接收到了消息就会给客户端返回消息成功写入到结果。如果此时Leader宕机，那么刚刚写入的那一条消息就丢失了。所以将acks的值设置为all，表示只有当所有的Follower都接收到消息才回认为这条消息是成功写入的。为了保证 leader 副本能有follower 副本能同步消息，我们一般会为topic设置replication.factor>= 3。这样就可以保证每个分区(partition) 至少有 3 个副本。虽然造成了数据冗余，但是带来了数据的安全性。还有一点就是unclean.leader.election.enable = false 。这个参数表示不会将没跟上Leader的副本参与到选主当中。这样的话只要选出了主节点，那么就一定是有最新消息的，不然就不能选择出主节点。

另外一个方法就是业务重试。如果消息没有成功到达kafka，此时就会触发消息发送失败回调。在这个回调中，首先创建一张表，用来记录发送到中间件上的消息，像下面这样：

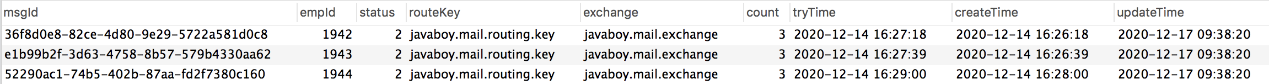


图5-21 存储字段

将消息发送到Kafka之后，就提取一些该消息的重要信息字段存入数据库。不需要将所有的字段都存入数据库，只要能够在后续出错的时候能够恢复发送就行。比较重要的有以下三个字段：

* status：表示消息的状态，有三个取值，0，1，2 分别表示消息发送中、消息发送成功以及消息发送失败。
* tryTime：表示消息的第一次重试时间（消息发出去之后，在 tryTime 这个时间点还未显示发送成功，此时就可以开始重试了）。
* count：表示消息重试次数。

在消息发送的时候，我们就往该表中保存一条消息发送记录，并设置状态 status 为 0，tryTime 为 1 分钟之后。在 confirm 回调方法中，如果收到消息发送成功的回调，就将该条消息的 status 设置为1（在消息发送时为消息设置 msgId，在消息发送成功回调时，通过 msgId 来唯一锁定该条消息）。另外开启一个定时任务，定时任务每隔 10s 就去数据库中捞一次消息，专门去捞那些 status 为 0 并且已经过了 tryTime 时间记录，把这些消息拎出来后，首先判断其重试次数是否已超过 3 次，如果超过 3 次，则修改该条消息的 status 为 2，表示这条消息发送失败，并且不再重试。对于重试次数没有超过 3 次的记录，则重新去发送消息，并且为其 count 的值+1。

当然这种思路有两个弊端：

一、这种解决办法引入了新的组件也就是MySQL。因为本身发送邮件的功能是不需要数据库进行参与的，这就增大了系统出错的可能性。

二、按照上面的思路，可能会出现同一条消息重复发送的情况，但是在消息消费时，解决好幂等性问题就行了。

1. 怎样解决消息重复消费问题？（幂等性问题）

幂等性本身是数学上的概念，即使公式：f(x)=f(f(x)) 能够成立的数学性质。在开发领域，则表示对于同一个系统，使用相同的条件，一次请求和多次请求对系统资源的影响是一致的。

在分布式系统中幂等性尤为重要，因为分布式系统中，我们经常会用到接口调用失败进而进行重试这个功能，这样就带来了对一个接口可能会使用相同的条件进行重复调用，在这样的条件下，保证接口的幂等性就尤为重要了。Token 则是目前使用比较广的一种方式，核心思想就是每个操作都有一个唯一凭证 token，一旦执行成功，对于重复的请求，总是返回同一个结果。大致的思路是这样，首先将 Kafka的消息自动确认机制改为手动确认，然后每当有一条消息消费成功了，就把该消息的唯一 ID 记录在 Redis 上，然后每次收到消息时，都先去 Redis 上查看是否有该消息的 ID，如果有，表示该消息已经消费过了，不再处理，否则再去处理。并且将这个唯一ID设置一个过期时间，那么只要在短时间内收到了两个同样的消息就不同重复消费了。

## 第六章 系统测试

### 6.1 系统测试

为了准确了解现行系统的接口能够响应多大的流量，需要对接口进行压力测试。压力测试是非常必要的。在系统上线之前，如果不知道接口能够承受的压力，那么在流量高峰期就特别容易造成宕机，进而引发损失。所以接下来使用superbenchmarker进行系统性能压测。

对于查询员工接口的测试：

PS C:\WINDOWS\system32> sb -u http://localhost:8080/#/emp/basic -c 20 -N 60

Starting at

[Press C to stop the test]

263639 (RPS: 4132)

---------------Finished!----------------

Finished at 2022/4/25 10:19:38 (took 00:01:03.9368223)

Status 200: 263639

RPS: 4312.4 (requests/second)

Max: 2123ms

Min: 0ms

Avg: 3.8ms

50% below 3ms

60% below 4ms

70% below 4ms

80% below 4ms

90% below 5ms

95% below 5ms

98% below 7ms

99% below 8ms

99.9% below 13ms

性能波动折线图如下：

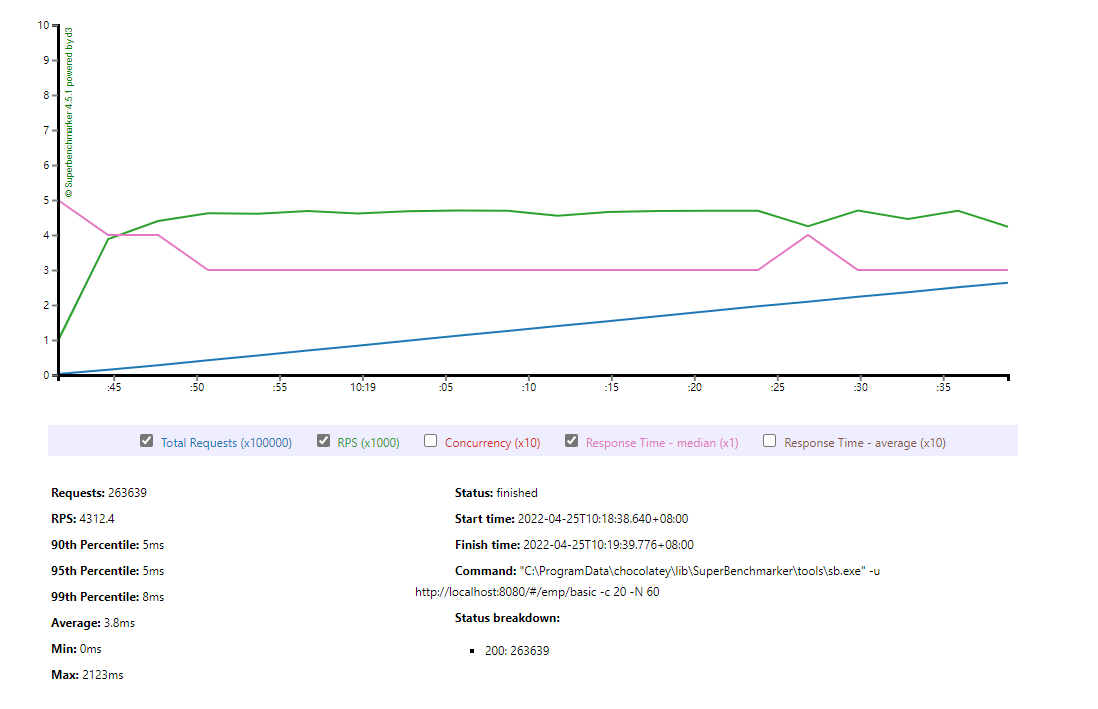


图6-1 性能测试

可以看出RPS为4312.4。即一秒钟之内能够处理的请求数量为4312.4。并且折线图没有明显的毛刺。波动基本维持在正常范围内。并且99.9%的请求都能够在13ms内被处理。

对搜索员工的接口进行测试：

C:\WINDOWS\system32>sb -u http://localhost:8080/employee/basic/?page=1"&"size=10"&"name=c -c 20 -N 60

Starting at

[Press C to stop the test]

11613 (RPS: 263.5) ...

Exiting.... please wait! (it might throw a few more requests)

11633 (RPS: 263.4)

---------------Finished!----------------

Finished at 2022/4/25 10:15:29 (took 00:00:44.2565283)

Status 401: 11633

RPS: 283.1 (requests/second)

Max: 2196ms

Min: 9ms

Avg: 67.5ms

50% below 61ms

60% below 65ms

70% below 69ms

80% below 75ms

90% below 84ms

95% below 89ms

98% below 95ms

99% below 102ms

99.9% below 2183ms

性能波动折线图如下：



图6-2 性能曲线

可以看到百分之90的请求能够在84ms内返回，0.9%的接口被有明显的响应时间放大的行为。99%的请求能够在102ms以下返回。

从上面的测试结果能够得出，本系统具有良好的系统稳定性，良好的性能。

## 结 论

本次项目课题的研究得出了以下结论：人事管理系统都在朝着易于操作，安全性高，页面精美的方向发展。人事管理系统对于企业的发展起着重要的作用。它节省人力物力成本，让管理者在最短的时间内就可以掌握非常准确的信息。可以预测人事管理系统的前景是非常好的，随着计算机的发展，每一个单位都需要有信息管理软件。本项目未来要研究的方向是怎样应对超大规模的数据和流量下系统的稳定性保证。这可以借鉴现有的一些分布式系统的设计与架构。

## 致 谢

这个毕业论文到这里也就已经结束了。在这里我衷心的感谢对我在这个项目课题研究以及论文撰写中给予支持的指导老师陶亮亮以及我的辅导员虞芬。他们在我的毕业论文以及课题研究中都给予了我非常大的帮助。另外，特别要感谢那些无私奉献的博客作者，他们的很多博客都给了我很大的启发，让我能够顺利的完成这个毕业设计。

## 参考文献

[1]牛兴霞,张萌.基于大数据理念的高校人事管理系统设计[J].工业技术与职业教育,2021,19(03):121-124.

[2]杨天普.人事管理系统的设计与实现[J].电子技术与软件工程,2020(16):42-43.

[3]鲁肖肖.高校人事管理系统的探究与设计[J].信息与电脑(理论版),2020,32(14):89-91.

[4]晁艺泽.人事管理系统设计[J].广西质量监督导报,2020(06):23-24.

[5]唐阳. 人事信息系统的设计与开发[D].南昌大学,2020.

[6]孟祥雪. 基于SSM的外聘人员人事管理系统设计与实现[D].电子科技大学,2020.

[7]赵国文,夏平平.基于JavaEE的人事管理系统的设计与实现[J].现代工业经济和信息化,2019,9(11):88-89.

[8]文玲华.基于SOA架构的高校人事管理系统的设计与实现[J].数码世界,2019(12):221-222.

[9]张晓天. 基于SSH框架的企业信息管理系统的设计与实现[D].大连海事大学,2019.

[10]安然.人事管理系统需求分析数据库设计[J].现代盐化工,2019,46(05):93-94