

人力资源管理系统项目文档

作 者： 邱杰

2022 年 4 月

摘 要

在如今这个人才需求量大的时代，各方企业为了永葆企业的活力与生机，在不断开拓进取的同时，又广泛纳用人才，为企业的长久发展奠定了基础。于是，各个企业与部门机构，都不可避免地会接触到人力资源管理的问题。

Hrm 是一款人力资源管理系统，其主要功能模块有员工个人信息修改、请假、员工的薪资管理、考勤管理、社保管理。其中考勤管理实现了员工考勤状态的修改与员工考勤月报表的导出，以及通过员工考勤记录的导入来实现员工考勤状态的判断。社保管理，主要实现了员工社保的计算以及明细的修改。薪资管理，实现了员工工资的调整，以及员工月工资报表的导出。

本项目采用了前后端分离的技术，前端是基于 Vue+ElementUI+Axios 开发的，后端则是基于 Spring Boot+MyBatis Plus+Jwt+Mysql。本项目实现了权限菜单管理，通过员工的权限动态渲染菜单，并动态生成路由。通过 Jwt token 来判断当前登录的员工以及员工的登录状态。

关键词：人力资源管理系统，Spring Boot，Vue，权限管理

Abstract

In today's era of large demand for talents, in order to maintain the vitality and vigor of the enterprise, enterprises of all parties continue to forge ahead while recruiting talents extensively, laying the foundation for the long-term development of the enterprise. Therefore, various enterprises and departments will inevitably come into contact with the issue of human resource management.

Hrm is a human resource management system. Its main functional modules include employee personal information modification, leave, employee salary management, attendance management, and social security management. Among them, attendance management realizes the modification of staff attendance status and the export of staff attendance monthly report, and the judgment of staff attendance status through the import of staff attendance records. Social security management, mainly realizes the calculation of employee social security and the modification of details. Salary management, realizes the adjustment of employee salary, and the export of employee monthly salary report.

This project adopts the technology of separating front and back ends. The front end is developed based on Vue+ElementUI+Axios, and the back end is based on Spring Boot+MyBatis Plus+Jwt+Mysql. This project implements permission menu management, dynamically renders menus through employee permissions, and dynamically generates routes. The currently logged in employee and the employee's login status are judged by the Jwt token.

Key Words: hrm system, Spring Boot, Vue, permission management

目 录

第 1 章 绪论.....	1
1.1 选题背景.....	1
1.2 选题目的.....	1
1.3 选题意义.....	1
第 2 章 关键技术介绍.....	2
2.1 前端技术.....	2
2.1.1 Vue.....	2
2.1.2 Axios.....	2
2.2 后端技术.....	2
2.2.1 Spring Boot.....	2
2.2.2 MyBatis Plus.....	2
2.2.3 JWT.....	3
2.2.4 Swagger.....	3
第 3 章 系统设计.....	4
3.1 功能结构设计.....	4
3.2 前端设计.....	4
3.2.1 前端接口封装.....	4
3.2.2 组件封装.....	5
3.2.3 动态路由.....	5
3.3 后端设计.....	5
3.3.1 全局异常处理.....	5
3.3.2 数据传输对象.....	5
3.4 数据库设计.....	5
3.4.1 系统管理模块.....	5
3.4.2 权限管理模块.....	7
3.4.3 考勤管理模块.....	8
3.4.4 薪资管理模块.....	10
3.5 系统类结构设计.....	13
3.5.1 员工模块.....	13

3.5.2 角色模块.....	13
3.5.3 菜单模块.....	14
3.5.4 文件模块.....	15
3.5.5 考勤模块.....	16
3.5.6 薪资模块.....	17
3.5.7 社保模块.....	18
第4章 系统实现.....	19
4.1 登录.....	19
4.2 个人信息编辑.....	21
4.3 修改密码.....	22
4.4 首页图表展示.....	23
4.5 标签栏页面跳转.....	25
4.6 多条件分页查询.....	26
4.7 角色分配.....	27
4.8 菜单分配.....	29
4.9 员工请假.....	30
4.10 考勤数据导入.....	33
4.11 考勤月报表导出.....	35
4.12 工资调整.....	36
4.13 月工资报表导出.....	38

第 1 章 绪论

1.1 选题背景

人力资源管理是企业运营中必不可少的一环，它关系到企业的前途与发展。尤其对于中小微企业来说，对企业的发展有着举足轻重的作用。随着近年来，政府对创业项目的大力扶持，我国创业型企业蓬勃发展。据统计，2019 年，我国创业企业数量已达 1810 万余家，占全国企业数的 97%，截止 2020 年，我国创业企业数量达到了 2030 万，同比增长 10%。虽然我国创业企业的基数在不断增大，但是能够长久存活的企业却少之又少。

在创业初期，随着企业初具规模，大多数创业者开始将主要精力集中在市场调研和开发产品上，而忽略了团队的内部管理。据调查，中国企业的平均寿命是 7.02 年，但 70% 的企业存活不超过 5 年，究其原因有很多，其中最重要的一点就是，人力资源管理未能有效推动企业的向前发展。

1.2 选题目的

在传统人事信息管理的模式下，各岗位的人事信息往往是独立，且需要单独分配人员进行管理，提高了维护信息的成本。由于数据互不相通，所以在进行人事调动的时候往往做了重复的工作。

通过开发一款人力资源管理系统，大大减少企业人事管理的劳动力成本，运用数据对人力资源进行精准调控和分配。

1.3 选题意义

随着计算机技术的不断进步和现代经济的不断发展，传统的管理技术已经不能满足企业的需要，计算机人力资源管理系统越来越受到企业的重视。

人力资源管理是企业生存发展的关键，它可以改善和加强企业的管理，企业要想进行一项生产活动和做出一些远景规划，就应该重视人力资源的分配与规划。通过一系列的考核、激励、打卡制度，对员工的工作积极性和业绩进行考察。使得创业者能够对企业的发展状况有更加细致的了解，并及时做出合理的调整，充分发挥员工的潜力。

无论是企业还是机构中，人事调动都是一个永远都离不开的话题，为一个岗位找到一名匹配的员工对公司的发展是极其有意义的。

第 2 章 关键技术介绍

2.1 前端技术

2.1.1 Vue

Vue 是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。

2.1.2 Axios

Axios 是一个基于 promise 的 HTTP 库，可以用在浏览器和 node.js 中，用于前端向后端发起请求，它拥有全局的请求和响应的拦截，可以非常方便的处理请求异常的问题。

2.2 后端技术

2.2.1 Spring Boot

Spring Boot 是 Spring 项目下的子项目，旨在快速开发应用，相比于 Spring，Spring Boot 避免了繁重的 xml 配置，它还采用了约定优于配置的软件设计范式，并提供了大量开箱即用的依赖模块，并且通过少量的配置，就能快速的搭建项目。

2.2.2 MyBatis Plus

MyBatis 是一款优秀的持久层框架，通过 XML 文件或注解配置来完成实体类与数据库之间的映射，舍弃了传统的 preparedStatement 设置参数操作数据库和使用 resultSet 获取结果集的过程。

MyBatis Plus 是由苞米豆团队开发的一款 MyBatis 增强工具，为简化数据库操作，提高开发效率为生。在 MyBatis 的基础上提供了常用的 crud 方法，甚至不需要配置 Mapper.xml 文件都能对数据库进行基础的操作。除此之外，MyBatis Plus 还提供了自动分页、代码生成的功能，通过配置相应的模板，就能一键生成绝大部分的后端代码，真正做到了简化开发。

2.2.3 JWT

JWT 全称 JSON Web Token，是目前比较流行跨域验证方案。相比于 session，session 生成的用户数据都会保存在服务器端，服务器只给用户的返回一个 sessionId，下次访问这个网站时，通过 cookie 将 sessionId 传递给服务器，从而得到相关的用户信息。毫无疑问，在这种情况下，服务器的内存会被大大的消耗，会带来一些性能开销。若服务器突然宕机，保存在服务器的用户数据就会消失，用户再次访问服务器就会被认为是第一次登录。

而 JWT 是保存在浏览器本地的，当用户第一次访问服务器，并且登录成功了，服务器会根据用户的唯一标识信息（比如 id），生成一个加密的 token，并返回用户信息。只要用户每次访问服务器的时候，在请求头中携带上 token，后端的拦截器获取 token，验证签证信息通过之后，就允许访问。

2.2.4 Swagger

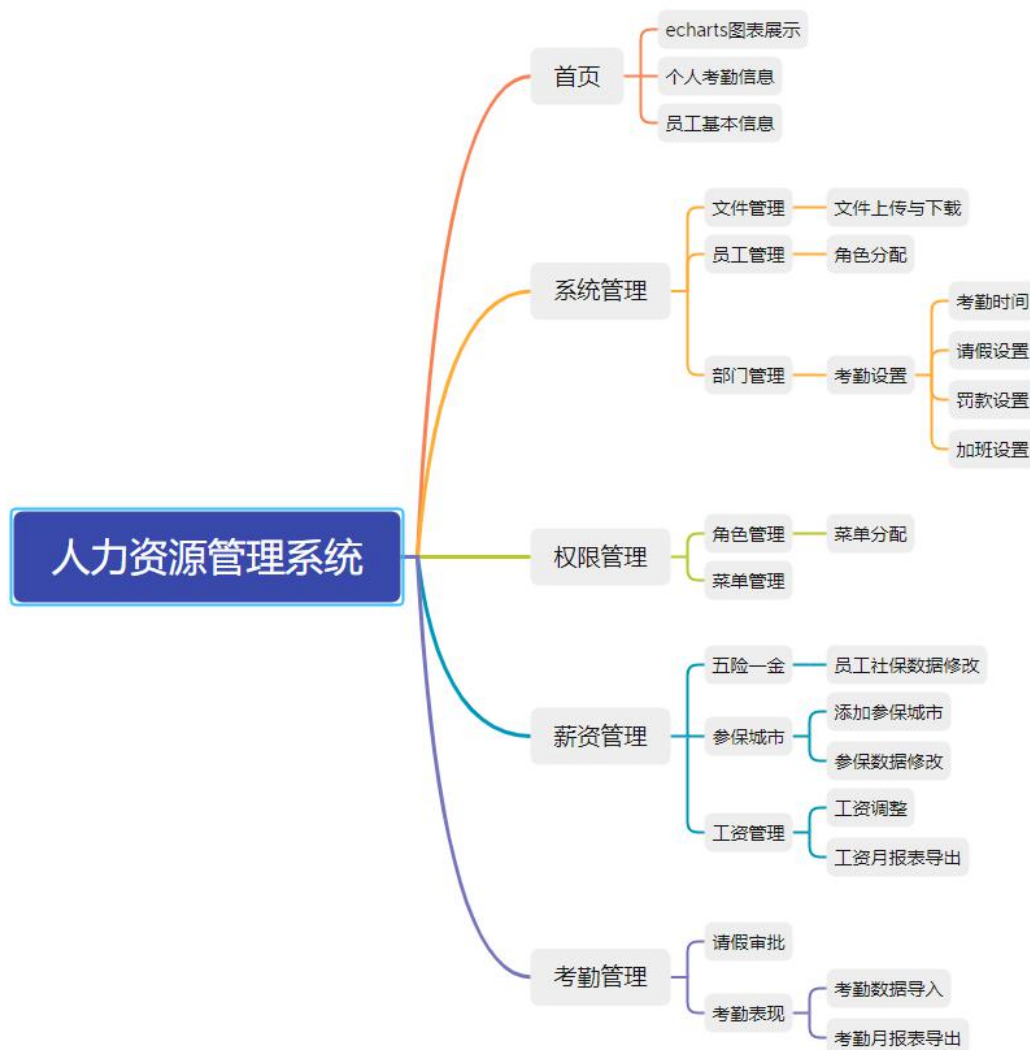
Swagger 是一款用于生成 Api 接口文档的工具，通过简单的注解配置，就可以将后端人员编写的接口，以文档的方式呈现。同时它还拥有简单的在线测试功能，相当于一个小型的 postman。

通过 Swagger 文档，极大地减少了前后端人员的交流成本，将前后端正真的连接起来。

第3章 系统设计

3.1 功能结构设计

本系统主要分四个模块，分别是系统管理和权限管理、薪资管理、考勤管理，系统管理主要用于日常事务管理，权限管理，用于控制员工的访问权限，薪资管理主要是对员工的五险一金以及社保数据的修改和添加，考勤管理主要是对员工的日常打卡进行记录和统计。



3.2 前端设计

3.2.1 前端接口封装

本项目对 Axios 进行了全局的封装，对前端请求和后端响应进行了统一的拦截，并

进行相应的处理。前端调用的 Api 都封装在 `src/api` 模块下，进行统一的管理。

3.2.2 组件封装

为了解决代码复用的问题，通过结合 Element UI。本项目对 form 表单和 table 数据表进行了进一步的组件封装。

3.2.3 动态路由

本项目采用了基于后端权限菜单的来实现动态路由，为了保证菜单数据的全局共享，菜单数据使用 `vuex` 来保存。当员工访路由时，通过全局路由守卫进行拦截，并向后端请求该员工的菜单数据。

3.3 后端设计

3.3.1 全局异常处理

为了高效地处理异常，本项目对异常进行了全局的统一处理，使用 `@ControllerAdvice` 声明一个全局异常处理器，并通过继承 `RuntimeException` 实现一个异常类，在需要异常处理的地方抛出自定义的异常。

3.3.2 数据传输对象

本项目中，后端响应给前端的数据都统一封装在 `ResponseDTO` 中，然后前端通过解析得到 `ResponseDTO` 的 json 对象。通过定义全局的业务状态码枚举类。前端通过后端响应数据的状态码来判断业务处理是否异常。

3.4 数据库设计

根据《阿里开发手册》，本项目的每张表都包含 `id`、`create_time`、`update_time` 三个字段。项目统一采用逻辑删除，每张表都包括 `is_deleted` 字段。根据实际业务需求，项目分为系统管理、权限管理、薪资管理、考勤管理四大模块。表前缀 `sys_`(系统管理)、`per_`(权限管理)、`att_`(考勤管理)、`sal_`(工资管理)、`soc_`(五险一金)。

3.4.1 系统管理模块

系统管理模块主要涉及 3 张表，负责对员工、部门、以及被上传文件的数据信息进

行保存。

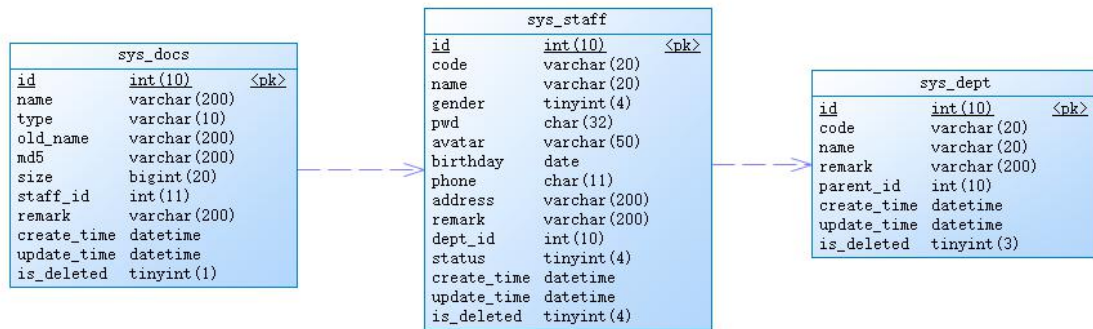


图 4.2 系统管理模块物理模型

员工表：

员工表包含了用户的基本信息，如电话、生日、地址等。

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	员工id
code	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>		员工编码
name	varchar	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		员工姓名
gender	tinyint	4	0	<input type="checkbox"/>	<input type="checkbox"/>		性别，0男，1女，默认0
pwd	char	32	0	<input type="checkbox"/>	<input type="checkbox"/>		员工密码
avatar	varchar	50	0	<input type="checkbox"/>	<input type="checkbox"/>		员工头像，设置默认头像
birthday	date	0	0	<input type="checkbox"/>	<input type="checkbox"/>		员工生日
phone	char	11	0	<input type="checkbox"/>	<input type="checkbox"/>		员工电话
address	varchar	200	0	<input type="checkbox"/>	<input type="checkbox"/>		地址
remark	varchar	200	0	<input type="checkbox"/>	<input type="checkbox"/>		员工备注
dept_id	int	10	0	<input type="checkbox"/>	<input type="checkbox"/>		部门id
status	tinyint	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		员工状态，0异常，1正常
create_time	datetime	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		创建时间
update_time	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		更新时间
is_deleted	tinyint	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		逻辑删除，0未删除，1已删除

图 4.3 员工表

部门表：

用于存放部门数据，本项目中部门只分为两级，一级部门，二级部门。parent_id 代表父级部门 id，若 parent_id 为 0 代表一级部门。

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	部门id
code	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>		部门编码
name	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>		部门名称
remark	varchar	200	0	<input type="checkbox"/>	<input type="checkbox"/>		部门备注
parent_id	int	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		父级部门id，0根部门
create_time	datetime	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		创建时间
update_time	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		更新时间
is_deleted	tinyint	3	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		逻辑删除，0未删除，1删除

图 4.4 部门表

文件表：

用于存放本地上传的文件信息，其中 md5 代表文件的标识，若将要上传的文件已经

存在于服务器上，则不用再上传。

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	
name	varchar	200	0	<input type="checkbox"/>	<input type="checkbox"/>		文件名称
type	varchar	10	0	<input type="checkbox"/>	<input type="checkbox"/>		文件类型
old_name	varchar	200	0	<input type="checkbox"/>	<input type="checkbox"/>		文件的原名称
md5	varchar	200	0	<input type="checkbox"/>	<input type="checkbox"/>		文件md5信息
size	bigint	20	0	<input type="checkbox"/>	<input type="checkbox"/>		文件大小KB
staff_id	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>		文件上传者id
create_time	datetime	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		创建时间
update_time	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		修改时间
is_deleted	tinyint	1	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		0未删除，1已删除，默认为0
remark	varchar	200	0	<input type="checkbox"/>	<input type="checkbox"/>		文件备注

图 4.5 文件表

3.4.2 权限管理模块

权限管理模块主要涉及 5 张表，主要对菜单数据、角色数据进行保存。

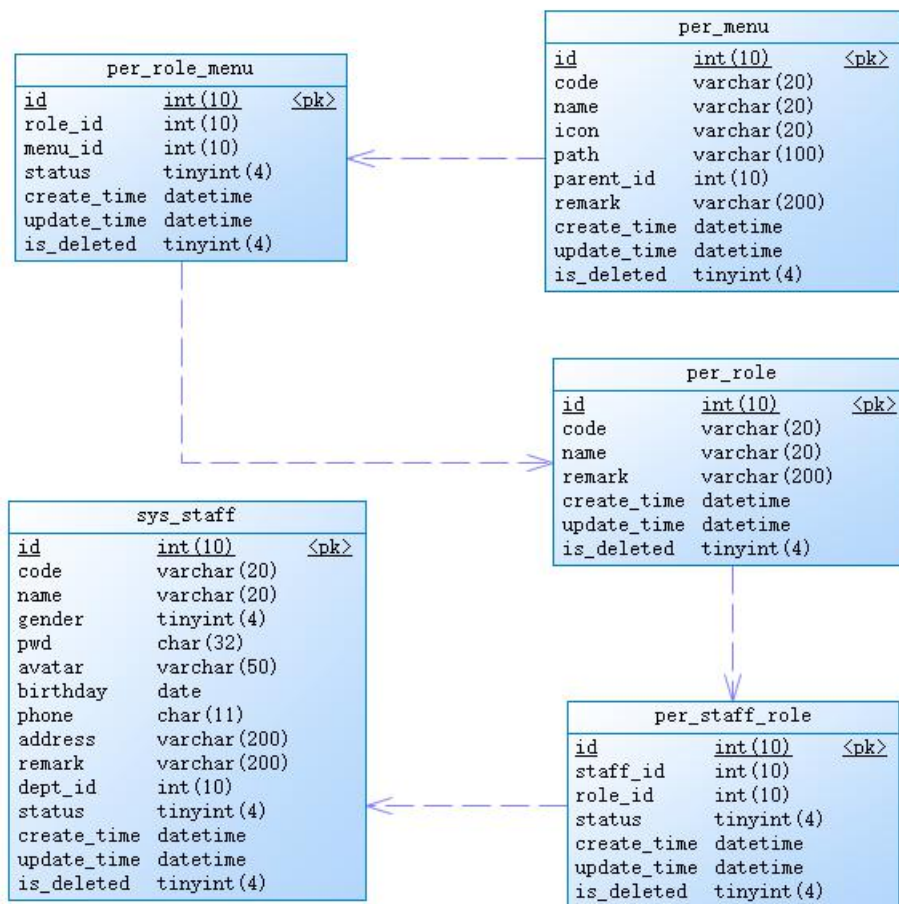


图 4.6 权限管理模块物理模型

角色表：

用于存放角色数据。

名	类型	长度	小数点	不是 null	虚拟	键	注释
► id	int	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	角色id
code	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>		角色编码
name	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>		角色名称
remark	varchar	200	0	<input type="checkbox"/>	<input type="checkbox"/>		角色备注
create_time	datetime	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
update_time	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
is_deleted	tinyint	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		

图 4.7 角色表

菜单表：

用于存放菜单数据，本项目中菜单只分为两级，一级菜单，二级菜单。其中 path 代表路由，parent_id 代表父级菜单的 id。如果为 0，则代表是一级菜单。

名	类型	长度	小数点	不是 null	虚拟	键	注释
► id	int	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	菜单id
code	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>		菜单编码
name	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>		菜单名称
icon	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>		
path	varchar	100	0	<input type="checkbox"/>	<input type="checkbox"/>		菜单路径
parent_id	int	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		父菜单id, 0代表根菜单, 默认0
remark	varchar	200	0	<input type="checkbox"/>	<input type="checkbox"/>		备注
create_time	datetime	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
update_time	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
is_deleted	tinyint	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		

图 4.8 菜单表

3.4.3 考勤管理模块

员工考勤模块主要涉及 9 张表，主要对员工的考勤数据，以及各部门的考勤规则、加班、请假、工作时间等规则信息的保存。

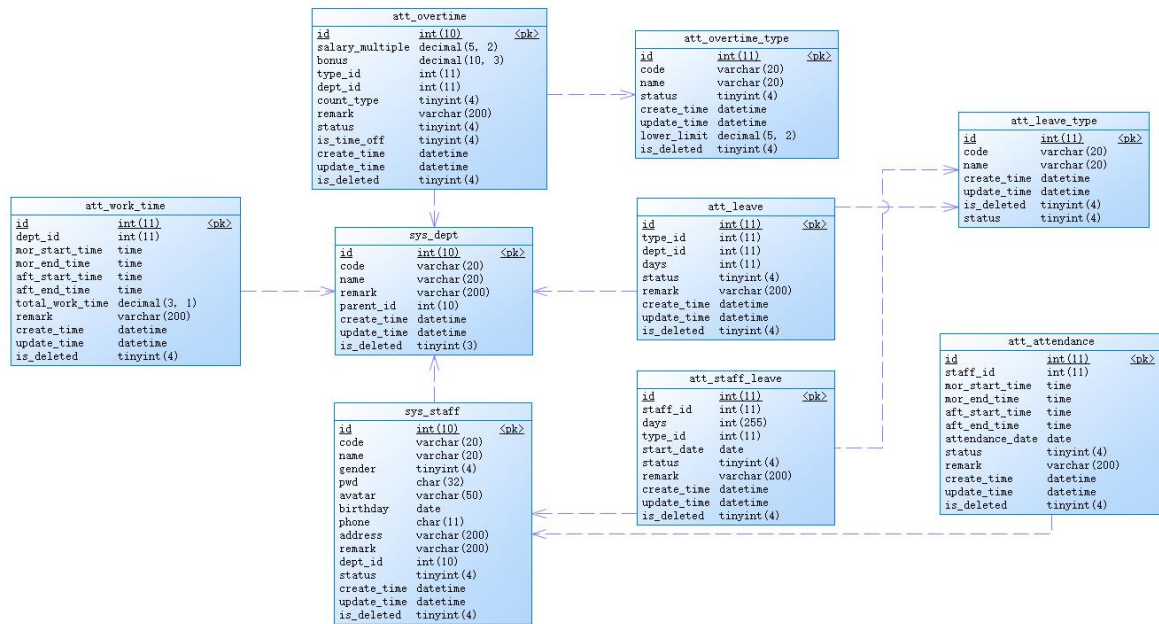


图 4.9 考勤管理模块物理模型

考勤表：

用于保存员工每天的上下班打卡时间，并记录当前的员工考勤状态。

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	11	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	
staff_id	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>		员工id
mor_start_time	time	0	0	<input type="checkbox"/>	<input type="checkbox"/>		上午上班时间
mor_end_time	time	0	0	<input type="checkbox"/>	<input type="checkbox"/>		上午下班时间
aft_start_time	time	0	0	<input type="checkbox"/>	<input type="checkbox"/>		下午上班时间
aft_end_time	time	0	0	<input type="checkbox"/>	<input type="checkbox"/>		下午下班时间
attendance_date	date	0	0	<input type="checkbox"/>	<input type="checkbox"/>		考勤日期
status	tinyint	4	0	<input type="checkbox"/>	<input type="checkbox"/>		0正常, 1迟到, 2早退, 3旷工, 4休假
remark	varchar	200	0	<input type="checkbox"/>	<input type="checkbox"/>		
create_time	datetime	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
update_time	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
is_deleted	tinyint	4	0	<input type="checkbox"/>	<input type="checkbox"/>		

图 4.10 员工考勤表

部门上班时间表：

主要保存每个部门所规定的上下班时间，用于对员工工作日当天是否迟到和旷工等状态进行判定。

人力资源管理系统

名	类型	长度	小数点	不是 null	虚拟	键	注释
▶ id	int	11	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	
dept_id	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>		
mor_start_time	time	0	0	<input type="checkbox"/>	<input type="checkbox"/>		上午上班时间
mor_end_time	time	0	0	<input type="checkbox"/>	<input type="checkbox"/>		上午下班时间
aft_start_time	time	0	0	<input type="checkbox"/>	<input type="checkbox"/>		下午上班时间
aft_end_time	time	0	0	<input type="checkbox"/>	<input type="checkbox"/>		下午下班时间
total_work_time	decimal	3	1	<input type="checkbox"/>	<input type="checkbox"/>		员工总工作时长
remark	varchar	200	0	<input type="checkbox"/>	<input type="checkbox"/>		
create_time	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
update_time	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
is_deleted	tinyint	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		

图 4.11 部门上班时间表

员工请假表：

主要保存员工的一些请假申请的记录，以及请假申请的审核状态。

名	类型	长度	小数点	不是 null	虚拟	键	注释
▶ id	int	11	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	
staff_id	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>		员工id
days	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>		请假的天数
type_id	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>		请假类型id
start_date	date	0	0	<input type="checkbox"/>	<input type="checkbox"/>		请假的开始日期
status	tinyint	4	0	<input type="checkbox"/>	<input type="checkbox"/>		0未审核，1审核通过，2驳回，3撤销
remark	varchar	200	0	<input type="checkbox"/>	<input type="checkbox"/>		
create_time	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
update_time	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
is_deleted	tinyint	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		

图 4.12 员工请假表

3.4.4 薪资管理模块

薪资管理模块主要涉及 7 张表，用于保存参保城市社保信息、员工每个的工资明细、以及考勤扣款情况。

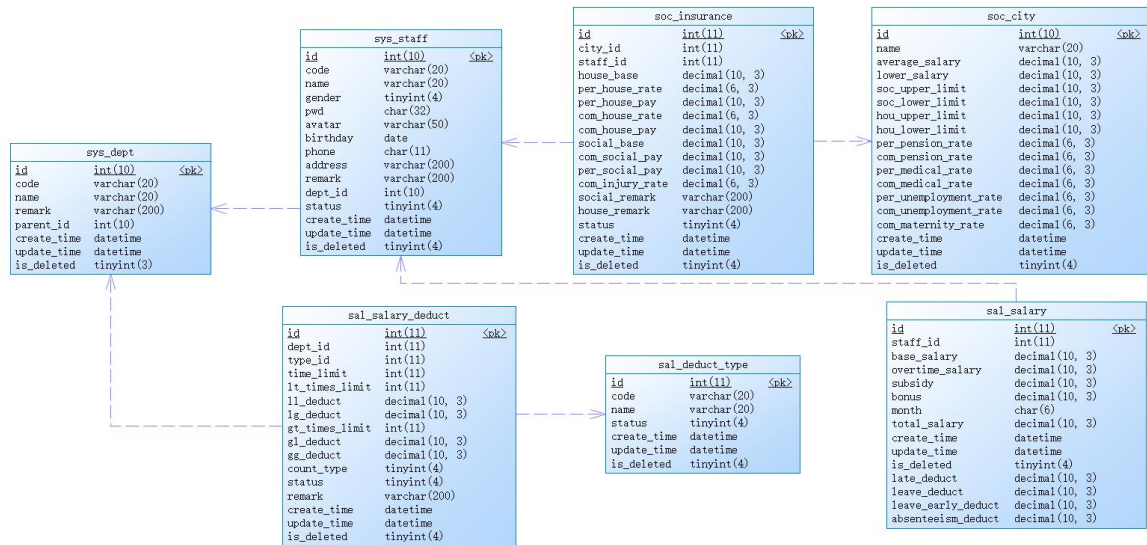


图 4.13 薪资管理模块物理模型

工资表：

工资表主要保存每个员工每个月的工资明细，以及考勤的罚款情况。

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	11	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	
staff_id	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>		员工id
base_salary	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		基础工资
overtime_salary	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		加班费
subsidy	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		生活补贴
bonus	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		奖金
total_salary	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		总工资
late_deduct	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		早退扣款
leave_deduct	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		休假扣款
leave_early_deduct	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		早退扣款
absenteeism_deduct	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		旷工扣款
month	char	6	0	<input type="checkbox"/>	<input type="checkbox"/>		月份
remark	varchar	200	0	<input type="checkbox"/>	<input type="checkbox"/>		
create_time	datetime	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
update_time	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
is_deleted	tinyint	4	0	<input type="checkbox"/>	<input type="checkbox"/>		

图 4.14 工资表

城市社保表：

社保城市表详细记录了当地的社保的各个项目以及公积金的缴费比例。

人力资源管理系统

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	
name	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>		参保城市
average_salary	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		职工上年度平均月工资
lower_salary	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		职工上年度最低月工资
soc_upper_limit	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		职工社保缴纳基数上限
soc_lower_limit	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		职工社保缴纳基数下限
hou_upper_limit	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		公积金缴纳基数上限
hou_lower_limit	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		公积金缴纳基数下限
per_pension_rate	decimal	6	3	<input type="checkbox"/>	<input type="checkbox"/>		个人养老保险缴费比例
com_pension_rate	decimal	6	3	<input type="checkbox"/>	<input type="checkbox"/>		企业养老保险缴费比例
per_medical_rate	decimal	6	3	<input type="checkbox"/>	<input type="checkbox"/>		个人医疗保险缴费比例
com_medical_rate	decimal	6	3	<input type="checkbox"/>	<input type="checkbox"/>		企业医疗保险缴费比例
per_unemployment_rate	decimal	6	3	<input type="checkbox"/>	<input type="checkbox"/>		个人失业保险缴费比例
com_unemployment_rate	decimal	6	3	<input type="checkbox"/>	<input type="checkbox"/>		企业失业保险缴费比例
com_maternity_rate	decimal	6	3	<input type="checkbox"/>	<input type="checkbox"/>		企业生育保险缴费比例
create_time	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
update_time	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
is_deleted	tinyint	4	0	<input type="checkbox"/>	<input type="checkbox"/>		

图 4.15 城市社保表

员工社保表：

员工社保表保存了员工社保以及公积金的缴纳费用和明细，也包括了企业为员工缴纳部分的明细和金额。

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	11	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	
city_id	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>		城市id
staff_id	int	11	0	<input type="checkbox"/>	<input type="checkbox"/>		员工id
house_base	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		公积金基数
per_house_rate	decimal	6	3	<input type="checkbox"/>	<input type="checkbox"/>		公积金个人缴纳比例
per_house_pay	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		公积金个人缴纳费用
com_house_rate	decimal	6	3	<input type="checkbox"/>	<input type="checkbox"/>		公积金企业缴纳比例
com_house_pay	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		公积金企业缴纳费用
social_base	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		社保基数
com_social_pay	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		社保企业缴纳费用
per_social_pay	decimal	10	3	<input type="checkbox"/>	<input type="checkbox"/>		社保个人缴纳费用
com_injury_rate	decimal	6	3	<input type="checkbox"/>	<input type="checkbox"/>		工伤保险企业缴纳比例
social_remark	varchar	200	0	<input type="checkbox"/>	<input type="checkbox"/>		社保备注
house_remark	varchar	200	0	<input type="checkbox"/>	<input type="checkbox"/>		公积金备注
status	tinyint	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		0未支付，1已支付，2支付失败
create_time	datetime	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
update_time	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		
is_deleted	tinyint	4	0	<input type="checkbox"/>	<input type="checkbox"/>		

图 4.16 员工社保表

3.5 系统类结构设计

为了统一处理后端给前端的响应，本项目中后端传递给前端的数据都封装在 ResponseDTO 中。

3.5.1 员工模块

StaffService 主要负责对员工进行操作，而 StaffRoleService 用于处理员工与角色的关联业务，如获取员工角色，为员工设置角色。

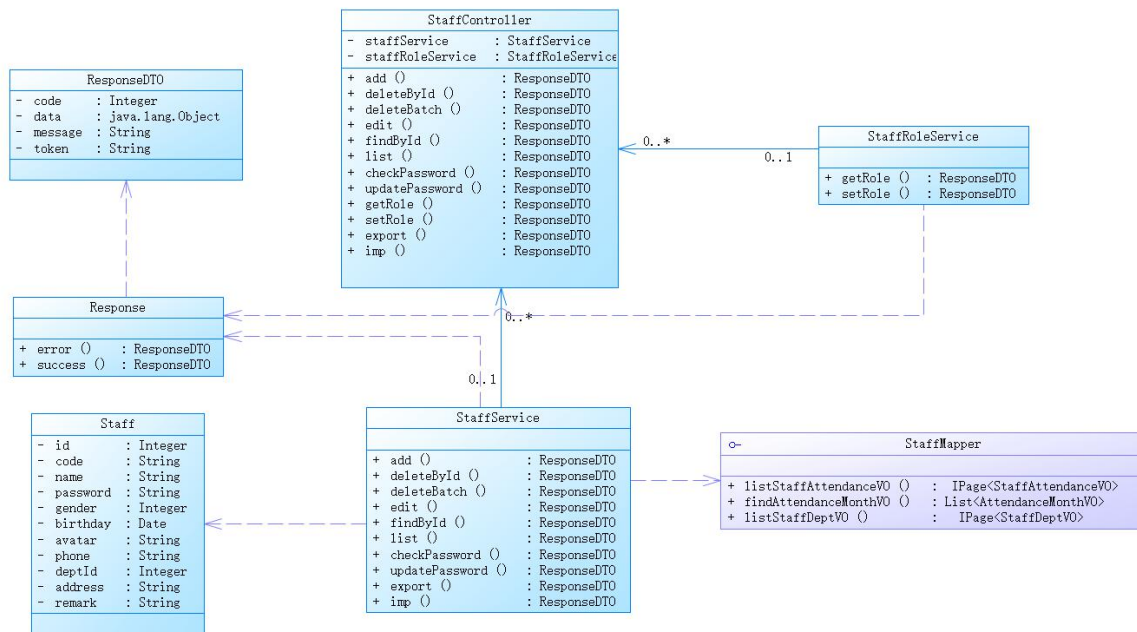


图 4.17 员工模块类图

3.5.2 角色模块

RoleService 主要负责对角色进行操作，而 RoleMenuService 用于处理角色与菜单的关联业务，如获取角色所分配的菜单，为角色设置菜单。

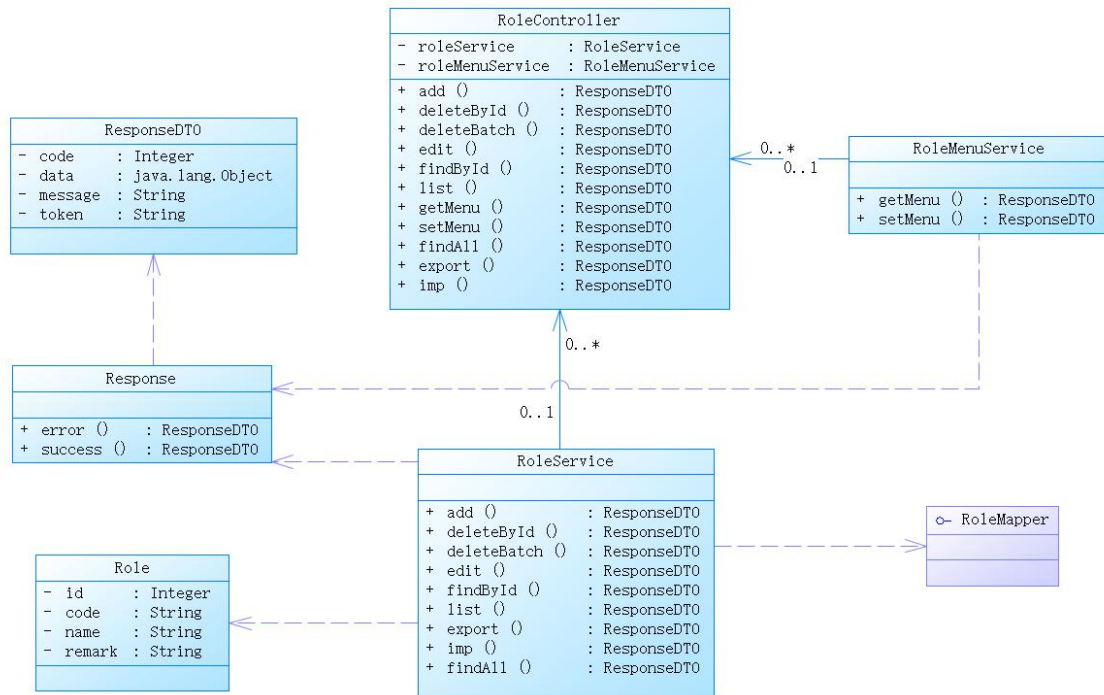


图 4.18 角色模块类图

3.5.3 菜单模块

MenuService 用于处理菜单业务，Menu 实体类中的 parentId 标识菜单的父级菜单，children 代表当前菜单的子菜单。

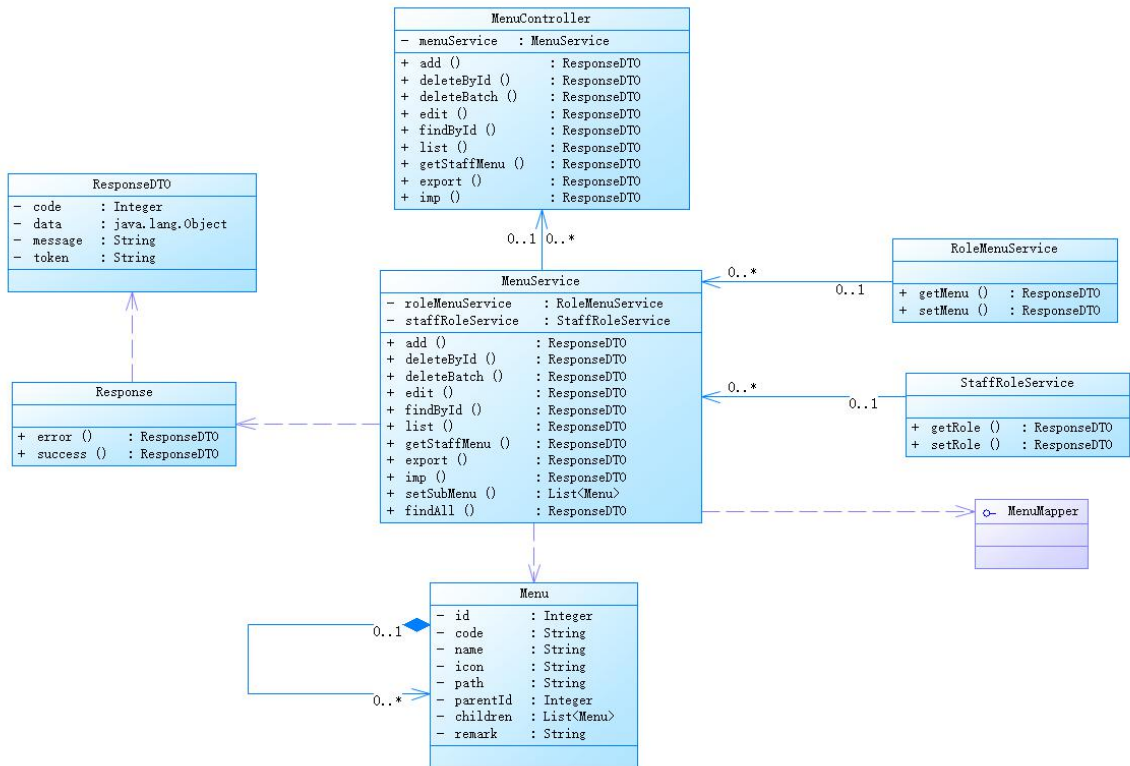


图 4.19 菜单模块类图

3.5.4 文件模块

DocsService 用于处理文件业务，其中 upload、download 分别为文件的上传与下载，imp、download 分别为数据的导入与导出。

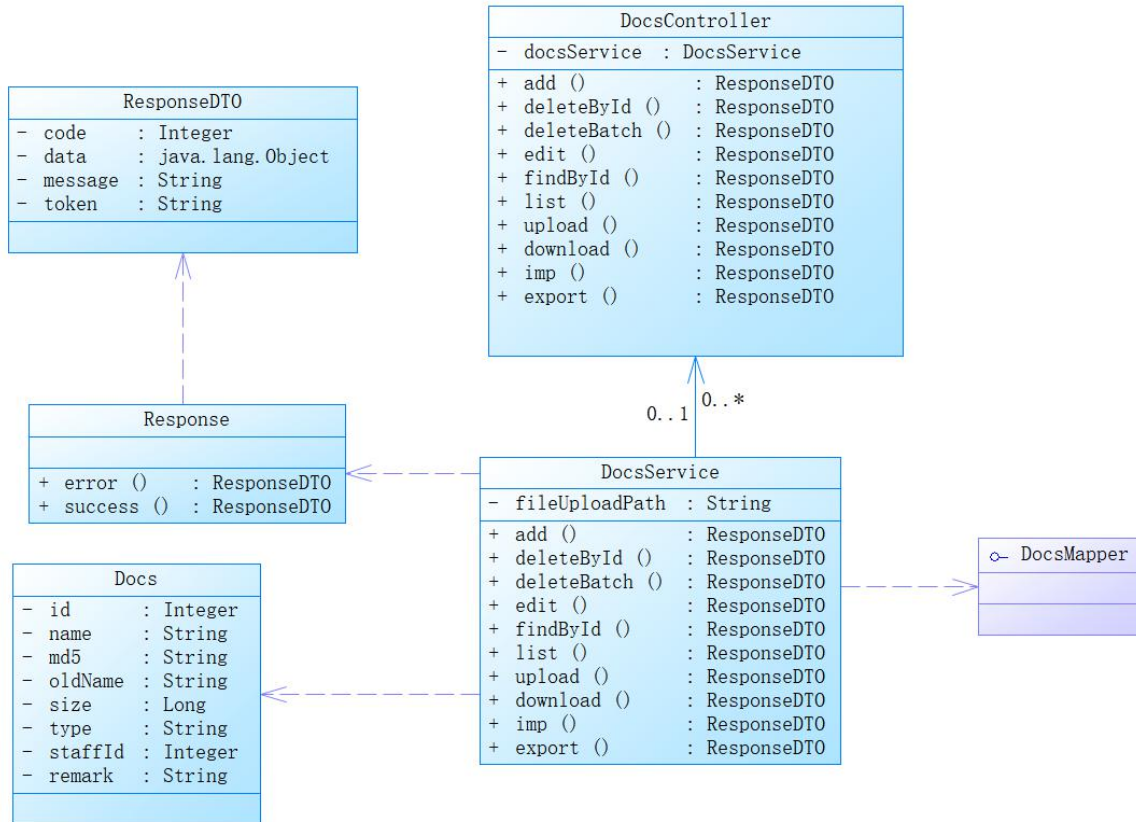


图 4.20 文件模块类图

3.5.5 考勤模块

考勤模块主要完成员工考勤状态的统计，以及月考勤报表的生成。通过导入员工的打卡时间表，完成对员工考勤状态的一个考量。

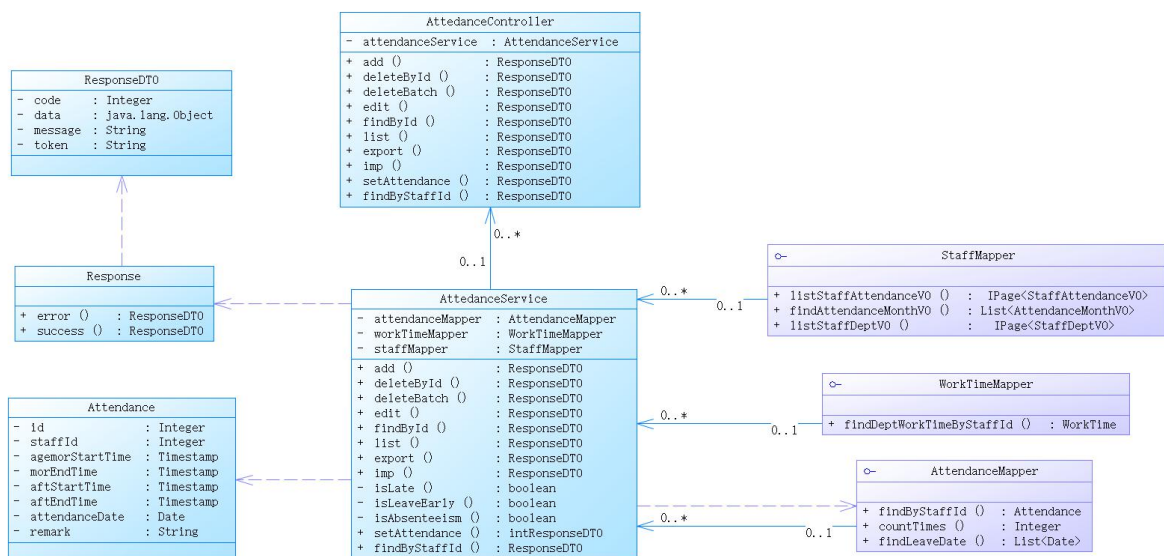


图 4.21 考勤模块类图

3.5.6 薪资模块

薪资模块主要完成员工工资的调整，以及员工工资月报表的导出。

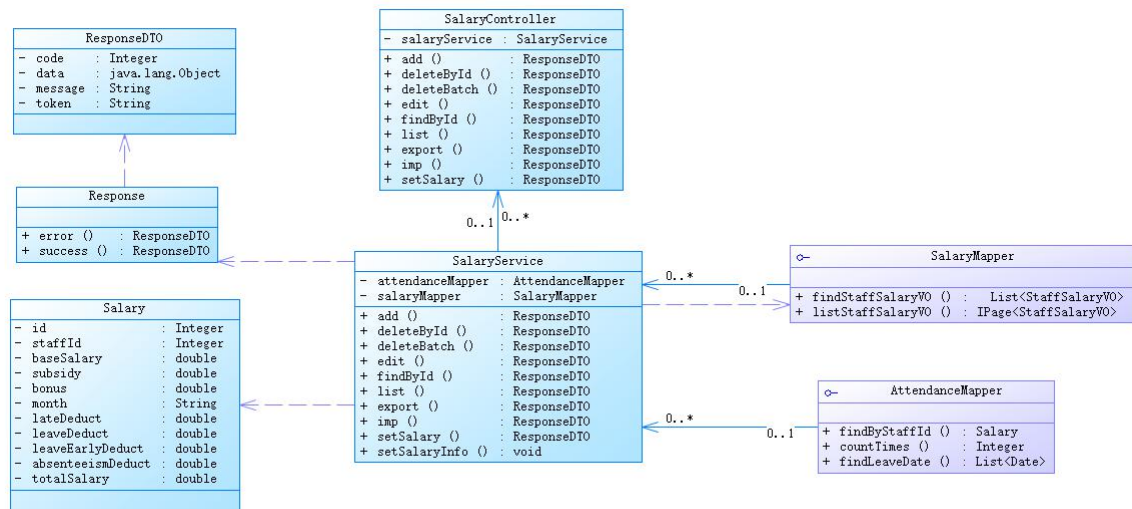


图 4.22 薪资模块类图

3.5.7 社保模块

社保模块主要是对参保地社保以及公积金的缴费税率的修改以及调整，还有个人社保信息的调整。

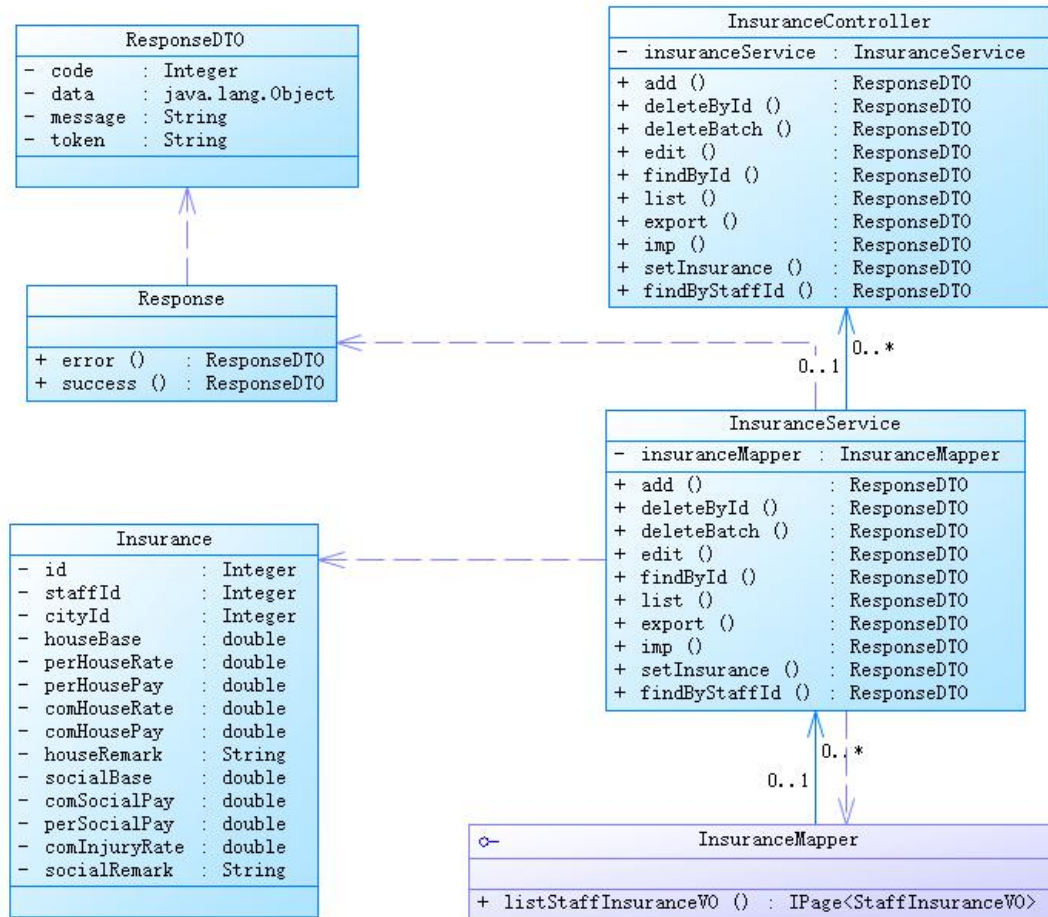


图 4.23 社保模块类图

第 4 章 系统实现

4.1 登录

此模块完成了员工的登录功能，员工通过工号和密码进行登录。若员工状态异常则无法登录。

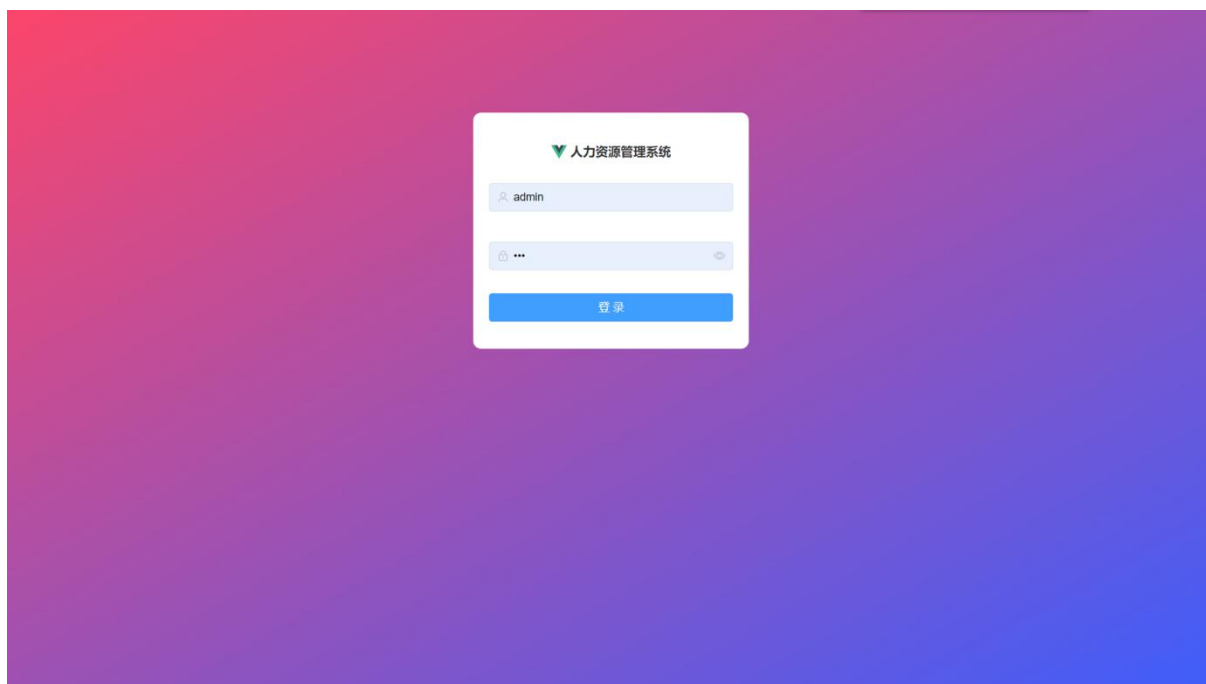


图 5.1 登录页面

登录接口：

```
@PostMapping("/login")
public ResponseDTO login(@RequestBody Staff staff)
```

接口说明：

当员工填写好登录账号和密码之后，前端会提交登录表单。后端接收到数据之后，会进行数据库查询，并将最终的查询结果以状态码的形式封装在 `ResponseDTO` 之中，并返回到前端。

核心代码:

```
public ResponseDTO login(Staff staff) {  
    StaffDeptVO staffDeptVO = this.staffMapper.findStaffInfo(staff.getCode(),  
MD5Util.MD55(staff.getPassword()));  
    if (staffDeptVO != null) {  
        if (staffDeptVO.getStatus() == 1) {  
            String token = JWTUtil.generateToken(staffDeptVO);  
            staffDeptVO.setPassword("");  
            return Response.success(staffDeptVO, token);  
        }  
        return Response.error(BusinessStatusEnum.STAFF_STATUS_ERROR);  
    }  
    return Response.error("用户名或密码错误！");  
}
```

登录流程:

当后端通过前端传递的账号和密码进行查询时，如果查询出多个员工，则会抛出异常，这里使用了自定义的异常，方便对全局异常进行控制和判断。当登录账号和密码都正常之后，后端向前端传递员工的信息和 token。前端接收之后，将员工信息和 token 存储在 localStorage 中。

4.2 个人信息编辑

此模块实现了员工个人信息的查看与修改，员工可以进行个人头像的修改。

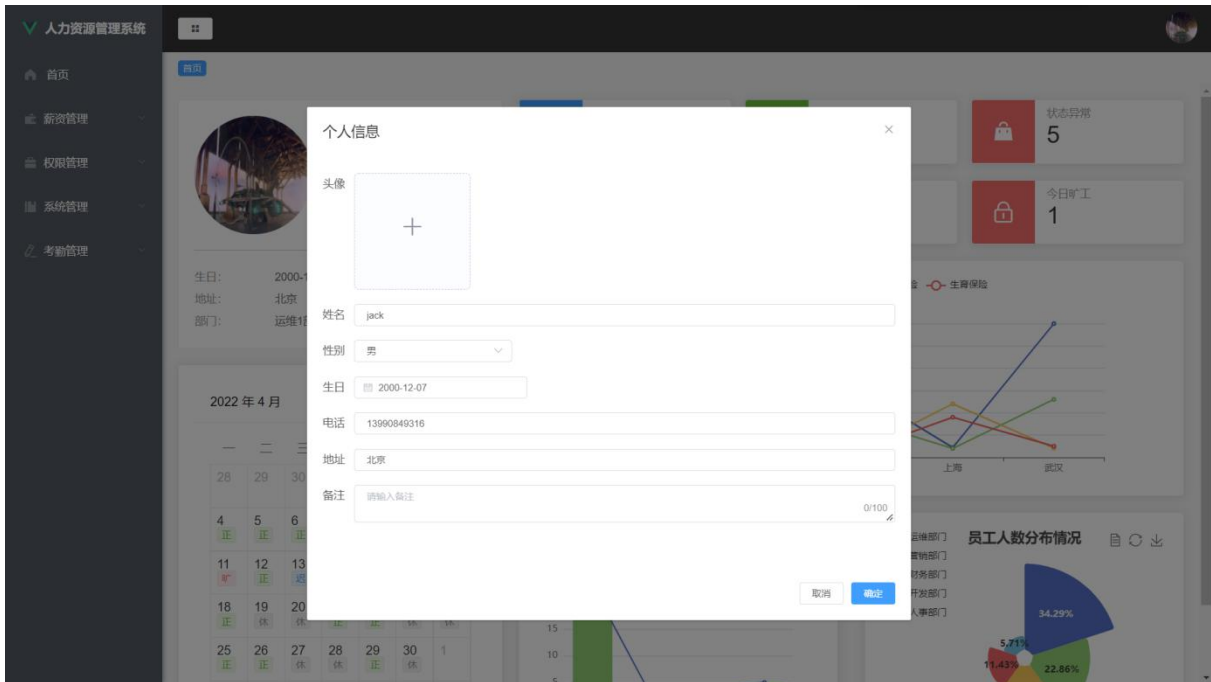


图 5.2 个人信息编辑页面

编辑接口：

```
@PostMapping
public ResponseDTO edit(@RequestBody Staff staff)
```

接口说明：

员工填写的信息会通过 el-form 组件的 data 属性，将数据绑定到一个 json 对象中，并通过 put 提交，最后后端接收数据，并完成相应员工信息的更新。

核心代码：

```
public ResponseDTO edit(Staff staff) {
    if (updateById(staff)) {
        return Response.success();
    }
    return Response.error();
}
```

流程：

员工填写的信息会通过 el-form 组件的 data 属性，将数据绑定到一个 json 对象中，并通过 put 提交，最后后端接收数据，并完成相应员工信息的更新。

4.3 修改密码

此模块完成了的员工个人密码的修改，若员工修改的密码与上一次密码项目，则提示修改失败。

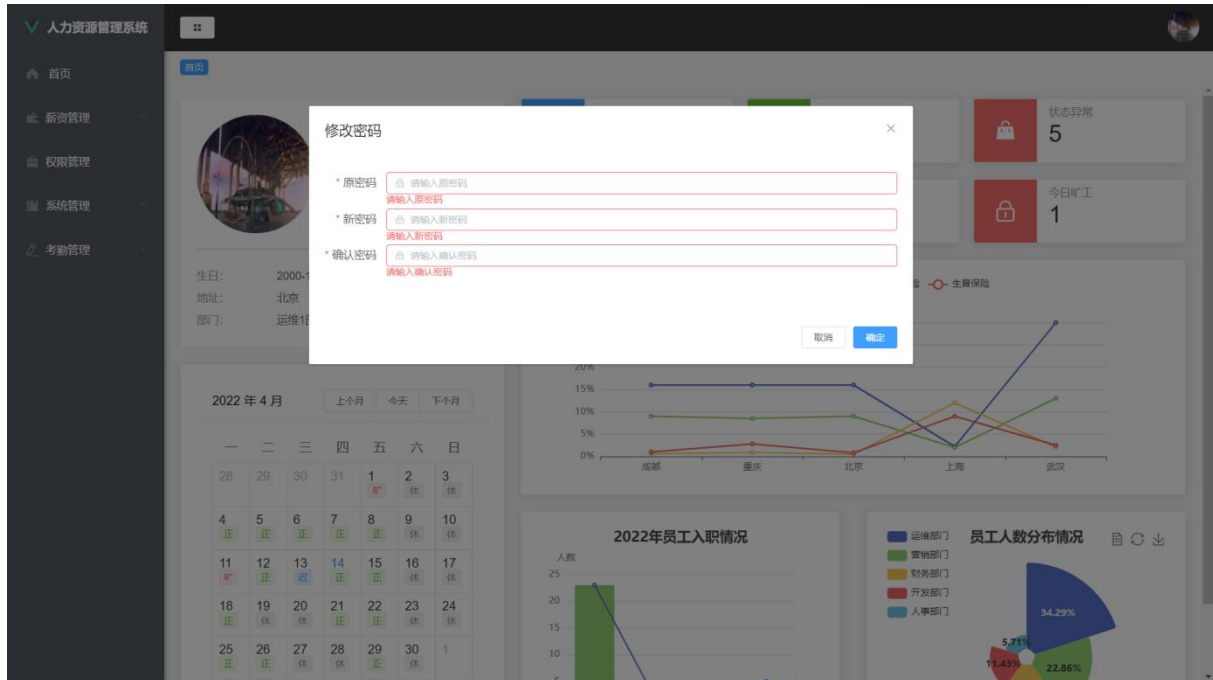


图 5.3 密码修改

密码检查接口：

```
@GetMapping("/check/{pwd}/{id}")
public ResponseDTO checkPassword(@PathVariable String pwd,@PathVariable Integer id)
```

接口说明：

通过前端传递的密码和 id，来判断员工填写的密码是否正确。

密码修改接口：

```
@PutMapping("/pwd")
public ResponseDTO updatePassword(@RequestBody Staff staff)
```

接口说明：

为了保证密码的安全性，使用 put 提交。

核心代码：

```
public ResponseDTO checkPassword(String pwd, Integer id) {
    Staff staff = getById(id);
    if(staff != null) {
        if (StringUtil.isNotBlank(pwd)) {
            if (MD5Util.MD55(pwd).equals(staff.getPassword())) {
                return Response.success();
            }
            throw new ServiceException(500,"密码错误！");
        }
        throw new ServiceException(500,"密码不能为空！");
    }
    throw new ServiceException(500,"此员工不存在！");
}

public ResponseDTO updatePassword(Staff staff) {
    staff.setPassword(MD5Util.MD55(staff.getPassword()));
    if(updateById(staff)){
        return Response.success();
    }
    return Response.error();
}
```

流程：

当员工打开修改密码的对话框，员工需要先正确填写原来的密码，然后填入将要修改的密码。当你修改的密码与原来的密码相同时，将会提示不能使用原来的密码，并且修改密码失败。密码修改成功之后，会自动退出登录，需要重新登录。

4.4 首页图表展示

首页主要展示了当前员工的一些基本信息，以及个人在当月的考勤情况。另外显示了系统的一些基本数据。

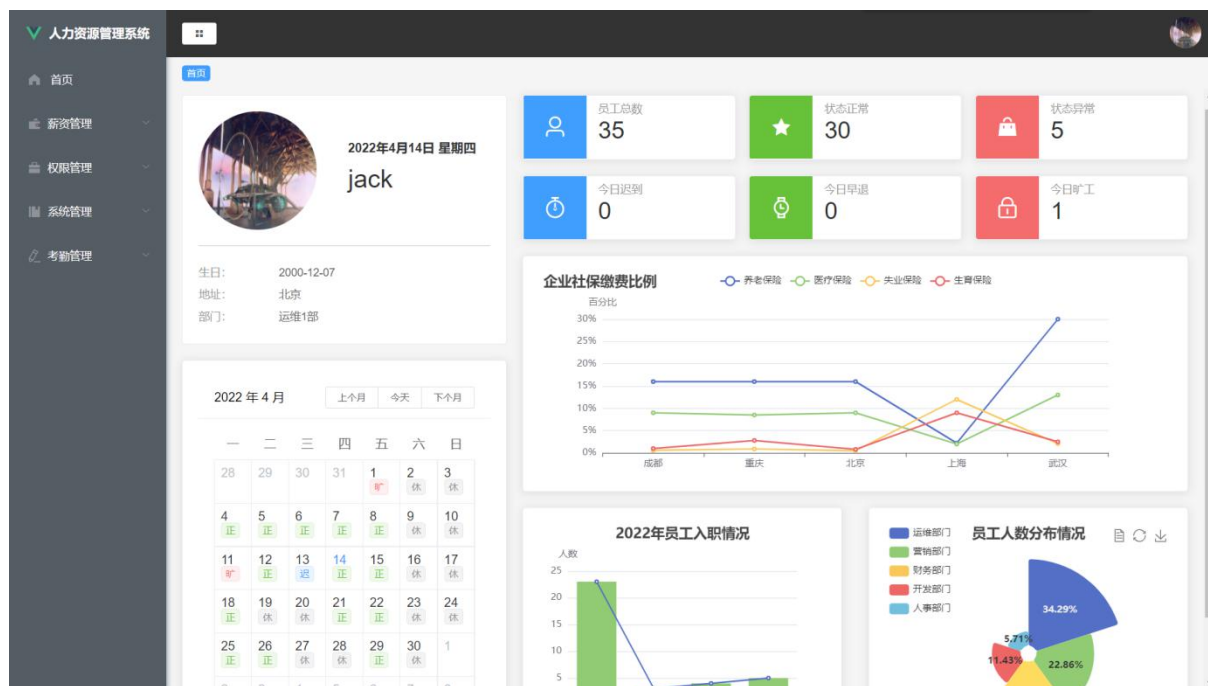


图 5.4 首页

图表数据接口：

```
@GetMapping("/staff")
public ResponseDTO getStaffData()
```

接口说明：

根据员工表的 create_time 字段，获取最近一年内的新增员工数。

统计数据接口：

```
@GetMapping("/count")
public ResponseDTO getCountData()
```

接口说明：

统计员工总数以及状态正常的员工的数目。

核心代码：

```
getRegisterData().then(response => {
  if (response.code = 200) {
    const quarters = ['一季度', '二季度', '三季度', '四季度']
    this.commonOption.xAxis.data = quarters      this.commonOption.series.forEach(item =>
    {
      item.data = response.data      })

    const registerChart = echarts.init(this.$refs.register);
    registerChart.setOption(this.commonOption);

    const commonChart = echarts.init(this.$refs.common);
    commonChart.setOption(this.commonOption);

    this.pieOption.series.forEach(item => {
      item.data = quarters.map((q, i) => ({
        name: q,
        value: response.data[i]
      }))
    })

    const pieChart = echarts.init(this.$refs.pie);
    pieChart.setOption(this.pieOption);
  } else {
    this.$message.error("获取数据失败！")
  }
})
```

流程：

当前端获取后端提供的数据之后，对数据项进行配置，然后将初始化好的图表挂载到 dom 元素节点上。

4.5 标签栏页面跳转

通过点击菜单标签，可以完成页面的跳转，以及标签的删除。



图 5.5 标签栏

核心代码：

```
ADD_TAG(state, menu) {
  if (menu.code !== 'home') {
    const result = state.tagList.findIndex(
      item => item.code === menu.code
    )
    if (result === -1) {
      state.tagList.push(menu)
    }
  }
  localStorage.removeItem("tagList")
  localStorage.setItem("tagList", JSON.stringify(state.tagList))
},
CLOSE_TAG(state, menu) {
  state.tagList = state.tagList.filter(item => item.code !== menu.code)
  localStorage.removeItem("tagList")
  localStorage.setItem("tagList", JSON.stringify(state.tagList))
}
```

流程：

当每点击一次侧边栏的菜单项，就将菜单数据存储在 vuex 中。而且为了保证页面刷新之后，标签栏不会消失，又将菜单数据保存在 localStorage 中。标签栏由 vuex 中存储的菜单数据动态生成。

4.6 多条件分页查询

选择不同条件，进行多条件分页查询

图 5.6 多条件分页查询

分页接口：

```
@PostMapping("/page")
public ResponseDTO list(@RequestParam(defaultValue = "1") Integer current,
  @RequestParam(defaultValue = "10") Integer size, @RequestBody Staff staff)
```

接口说明：

current 代表第几页，size 每次页面所展示的数据的个数，staff 包含了多条件查询的条件。

核心代码：

```
public ResponseDTO list(Integer current, Integer size, Staff staff) {
    IPage<Staff> pageConfig = new Page<>(current, size);
    QueryWrapper<Staff> wrapper = new QueryWrapper<>();
    if (staff.getName() != "" && staff.getName() != null) {
        wrapper.like("name", staff.getName());
    }
    if (staff.getBirthDay() != null) {
        wrapper.ge("birthday", staff.getBirthDay());
    }
    if (staff.getDeptId() != null) {
        wrapper.eq("dept_id", staff.getDeptId());
    }
    if (staff.getStatus() != null) {
        wrapper.eq("status", staff.getStatus());
    }
    IPage<Staff> page = page(pageConfig, wrapper);
    Map map = new HashMap();
    map.put("pages", page.getPages());
    map.put("total", page.getTotal());
    map.put("list", page.getRecords());
    return Response.success(map);
}
```

流程：

后端根据前端提交的查询数据，然后使用 MyBatis Plus 提供的分页方法进行分页，然后将满足条件的数据返回给前端。

4.7 角色分配

此模块主要实现了为员工分配角色，一个员工可以分配多个角色。

图 5.7 角色分配

角色接口:

```
@GetMapping("/all")
public ResponseDTO findAll()
```

接口说明:

获取所有角色。

员工角色接口:

```
@GetMapping("/role/{staffId}")
public ResponseDTO getRole(@PathVariable Integer staffId)
```

接口说明:

获取目前员工已拥有的角色。

设置员工角色接口:

```
@PostMapping("/role/{staffId}")
public ResponseDTO setRole(@PathVariable Integer staffId, @RequestBody List<Integer>
roleIds)
```

接口说明:

员工可以选择多个角色，根据员工的 id 和选择的角色 id，为员工设置角色。

核心代码:

```
public ResponseDTO setRole(Integer staffId, List<Integer> roleIds) {
    QueryWrapper<StaffRole> wrapper = new QueryWrapper<>();
    wrapper.eq("staff_id",staffId);
    List<StaffRole> list = list(wrapper);
    for (StaffRole staffRole : list) {
        if (roleIds.contains(staffRole.getRoleId())){
            staffRole.setStatus(1);
        }else{
            staffRole.setStatus(0);
        }
        updateById(staffRole);
    }
    for (Integer roleId : roleIds) {
        StaffRole staffRole = new StaffRole();
        staffRole.setStaffId(staffId);
        staffRole.setRoleId(roleId);
        staffRole.setStatus(1);
        QueryWrapper<StaffRole> queryWrapper = new QueryWrapper<>();
        queryWrapper.eq("staff_id",staffId).eq("role_id",roleId);
        if(!saveOrUpdate(staffRole,queryWrapper)){
            throw new ServiceException(500,"添加角色失败！");
        }
    }
    return Response.success();
}
```

流程：

当员工点击分配菜单按钮时，前端会向后端请求所有的角色数据，紧接着再查询当前员工所拥有的角色，并将对应的角色框勾选上。当员工选择好了角色并提交之后，后端会先禁用不需要的角色，然后再添加或更新。

4.8 菜单分配

此模块实现了为角色分配菜单，一个角色可以选择多个菜单。

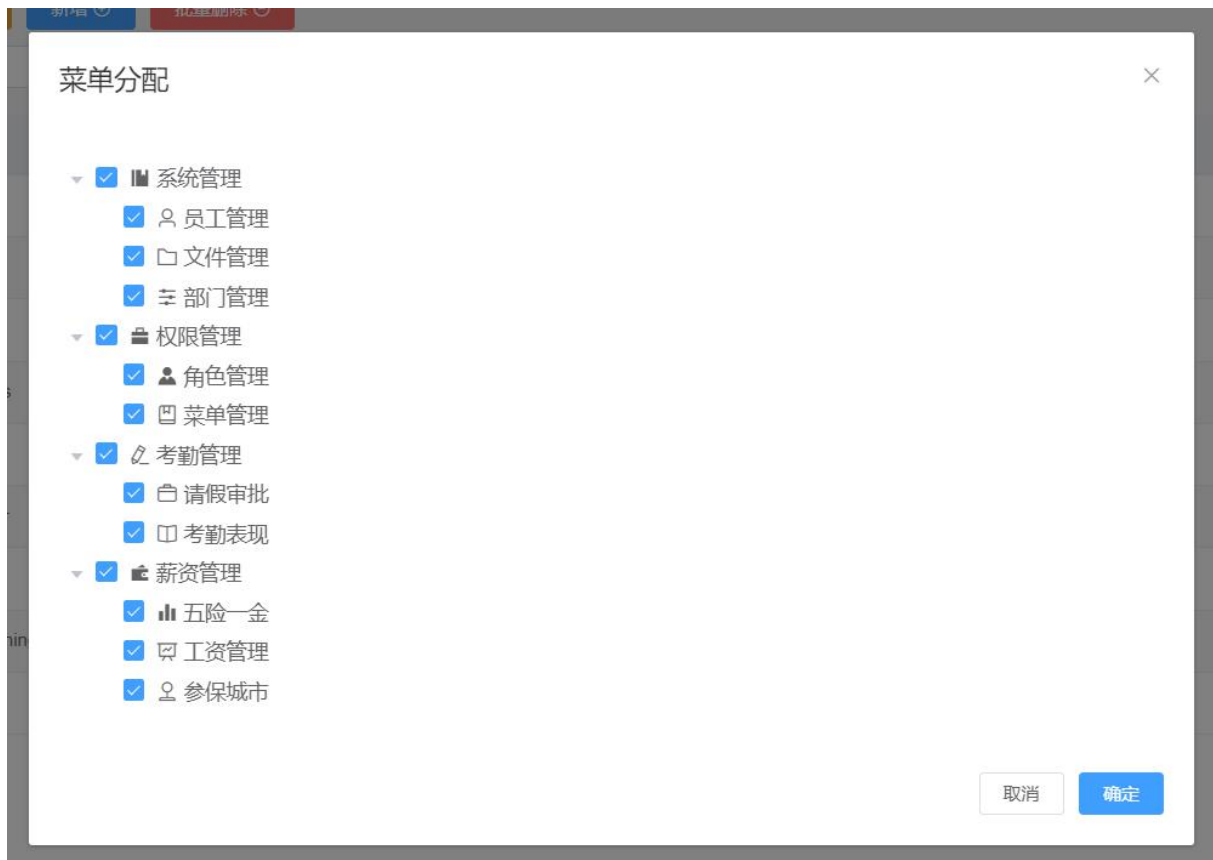


图 5.8 分配菜单

菜单接口：

```
@GetMapping("/all")
public ResponseDTO findAll()
```

接口说明：

获取所有的菜单数据。

角色菜单接口：

```
@GetMapping("/menu/{roleId}")
public ResponseDTO getMenu(@PathVariable Integer roleId)
```

接口说明：

获取角色的菜单数据。

设置角色菜单接口：

```
@PostMapping("/menu/{roleId}")
public ResponseDTO setMenu(@PathVariable Integer roleId, @RequestBody List<Integer> menuIds)
```

接口说明：

根据角色 id，和选择的菜单 id，为角色设置菜单。

核心代码：

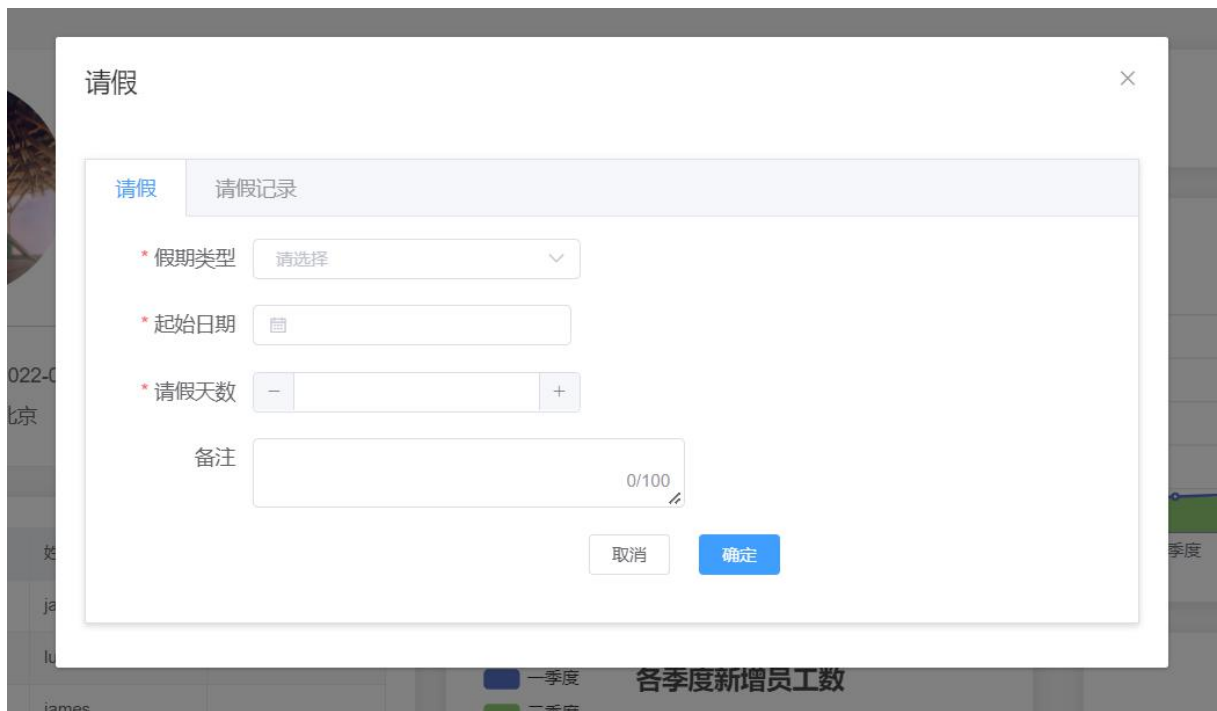
```
public ResponseDTO setMenu(Integer roleId, List<Integer> menuIds) {
    QueryWrapper<RoleMenu> wrapper = new QueryWrapper<>();
    wrapper.eq("role_id", roleId);
    List<RoleMenu> list = list(wrapper);
    for (RoleMenu roleMenu : list) {
        if (menuIds.contains(roleMenu.getMenuId())) {
            roleMenu.setStatus(1);
        } else {
            roleMenu.setStatus(0);
        }
        updateById(roleMenu);
    }
    for (Integer menuId : menuIds) {
        RoleMenu roleMenu = new RoleMenu();
        roleMenu.setRoleId(roleId);
        roleMenu.setMenuId(menuId);
        roleMenu.setStatus(1);
        QueryWrapper<RoleMenu> queryWrapper = new QueryWrapper();
        queryWrapper.eq("role_id", roleId).eq("menu_id", menuId);
        if (!saveOrUpdate(roleMenu, queryWrapper)) {
            throw new ServiceException(500, "角色添加菜单失败！");
        }
    }
    return Response.success();
}
```

流程：

当点击分配菜单按钮，前端会向后端请求所有的菜单数据，这里只返回了父级菜单，因为子菜单都作为父级菜单的 `children` 属性被携带。当菜单被渲染好之后，紧接着获取当前角色的所有拥有的菜单，并将对应项勾选。提交之后，后端先将不需要的菜单禁用，然后再重新更新或设置菜单。

4.9 员工请假

当员工填写请假的基本信息，点击确定，完成了请假申请的提交。



请假

请假记录

* 假期类型 请选择

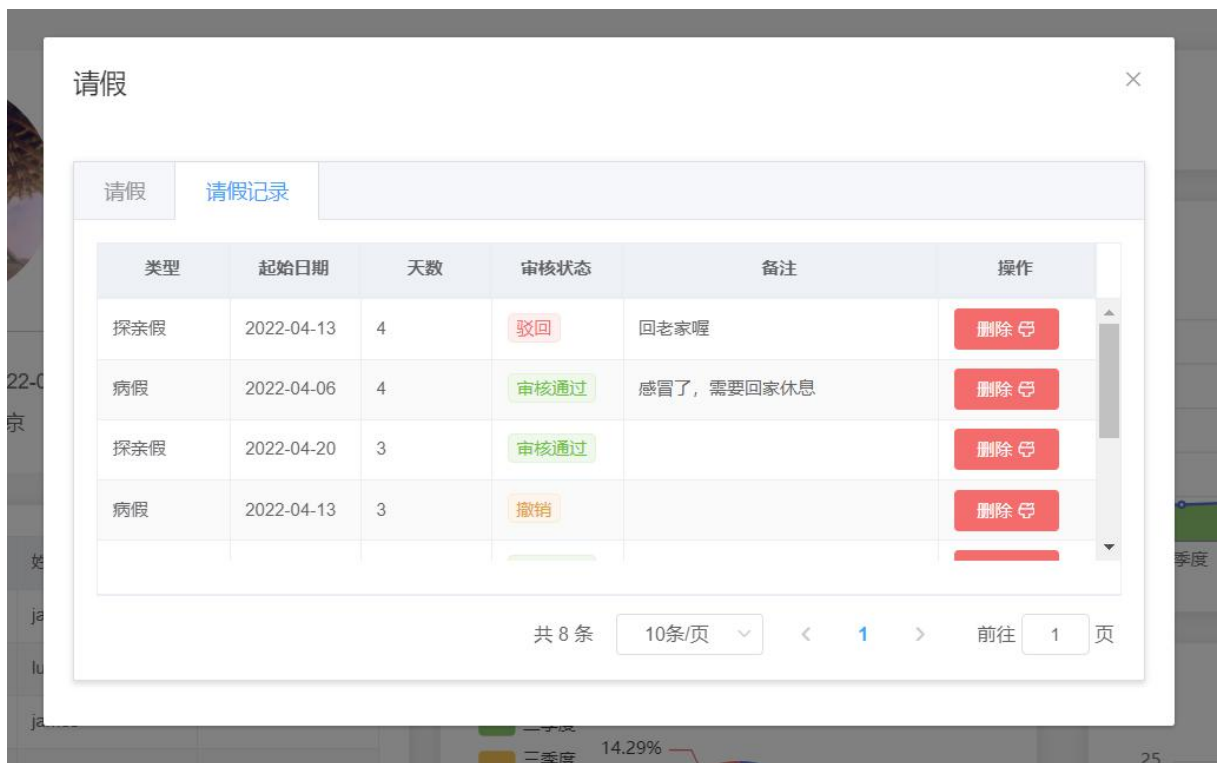
* 起始日期

* 请假天数 - +

备注 0/100

取消 确定

图 5.9 员工请假



请假

请假记录

类型	起始日期	天数	审核状态	备注	操作
探亲假	2022-04-13	4	驳回	回老家喔	删除
病假	2022-04-06	4	审核通过	感冒了, 需要回家休息	删除
探亲假	2022-04-20	3	审核通过		删除
病假	2022-04-13	3	撤销		删除

共 8 条 10条/页 < 1 > 前往 1 页

图 5.10 请假记录

请假接口:

```
@PostMapping
public ResponseDTO add(@RequestBody StaffLeave staffLeave)
```

接口说明:

新增一个请假申请

请假记录接口：

```
@GetMapping("/staff")
public ResponseDTO findByStaffId(@RequestParam(defaultValue = "1") Integer current,
    @RequestParam(defaultValue = "10") Integer size, Integer id)
```

接口说明：

获取员工的请假记录

请假内容更新接口：

```
@PutMapping
public ResponseDTO edit(@RequestBody StaffLeave staffLeave)
```

接口说明：

如果请假申请审批通过，那么就将节假日的考勤状态设置为休假状态

核心代码：

```
public ResponseDTO edit(StaffLeave staffLeave) {
    if (staffLeave.getStatus() == AuditStatusEnum.APPROVE) {
        for (int i = 0; i < staffLeave.getDays(); i++) {
            Date attendanceDate = DateUtil.offsetDay(staffLeave.getStartDate(),
i).toSqlDate();
            if (!DateUtil.isWeekend(attendanceDate)) {
                Attendance attendance = new
Attendance().setAttendanceDate(attendanceDate).setStaffId(staffLeave.getStaffId()).setStatus
(AttendanceStatusEnum.LEAVE);
                QueryWrapper<Attendance> queryWrapper = new QueryWrapper<>();
                queryWrapper.eq("staff_id",
attendance.getStaffId()).eq("attendance_date", attendance.getAttendanceDate());
                if (!this.attendanceService.saveOrUpdate(attendance, queryWrapper)) {
                    return Response.error();
                }
            }
        }
    }
    if (updateById(staffLeave)) {
        return Response.success();
    }
    return Response.error();
}
```

流程：

当员工提交一个请假申请之后，申请处于未审核状态，如果员工拥有未被审核的请假申请时，该员工是不能再次发起请假申请，另外员工也可以将未被审核的申请进行撤销。当申请被管理员审批通过了之后，系统就会自动地将员工休假期间的考勤状态设置为休假。

4.10 考勤数据导入

通过导入考勤数据，完成员工考勤状态的记录。

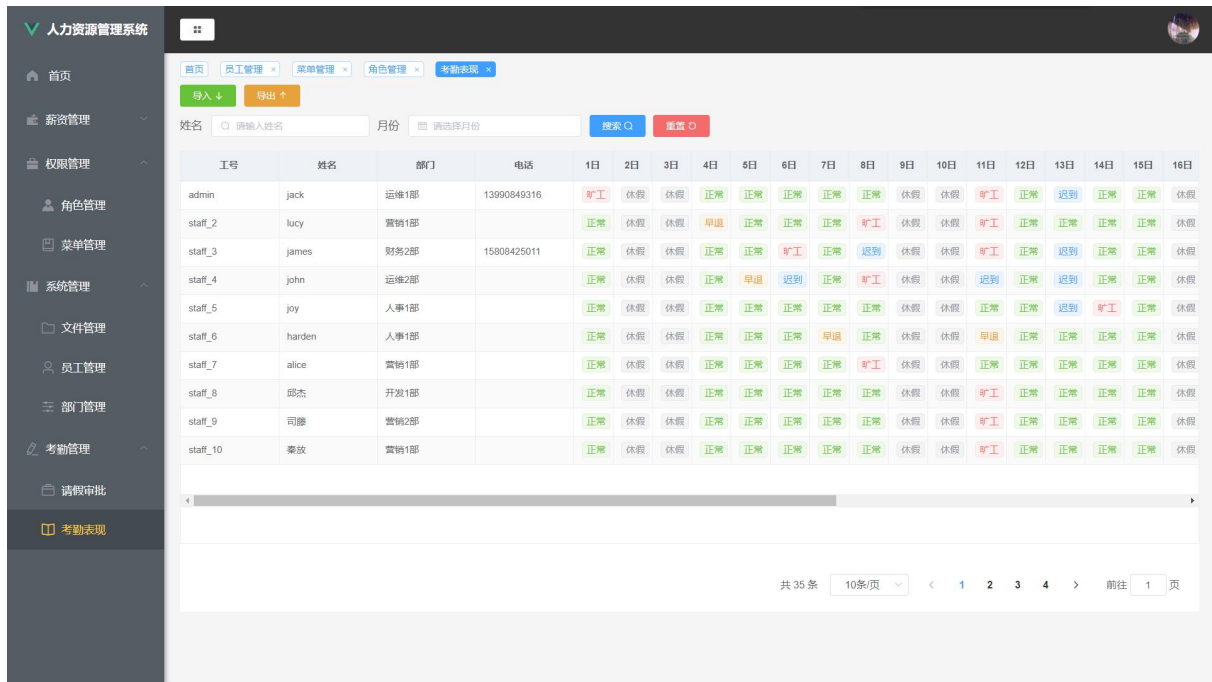


图 5.11 员工考勤页面

A	B	C	D	E	F
员工id	上午上班时间	上午下班时间	下午上班时间	下午下班时间	考勤日期
1	2022/3/23 6:00	2022/3/23 12:00	2022/3/23 13:00	2022/3/23 18:00	2022/10/16
1	2022/3/23 7:30	2022/3/23 11:30	2022/3/23 13:30	2022/3/23 18:00	2022/10/17
1	2022/3/23 6:50	2022/3/23 12:00	2022/3/23 13:00	2022/3/23 17:00	2022/10/18
1	2022/3/23 9:00	2022/3/23 10:30	2022/3/23 13:01	2022/3/23 16:00	2022/10/19
2	2022/3/23 6:45	2022/3/23 11:40	2022/3/23 13:10	2022/3/23 16:30	2022/10/16
2	2022/3/23 7:00	2022/3/23 12:35	2022/3/23 13:12	2022/3/23 19:20	2022/10/17
2	2022/3/23 8:40	2022/3/23 13:00	2022/3/23 14:09	2022/3/23 18:10	2022/10/18
2	2022/3/23 8:17	2022/3/23 12:23	2022/3/23 13:01	2022/3/23 16:49	2022/10/19
3	2022/3/23 7:34	2022/3/23 12:10	2022/3/23 15:11	2022/3/23 21:00	2022/10/16
3	2022/3/23 6:12	2022/3/23 11:00	2022/3/23 13:01	2022/3/23 16:00	2022/10/17
3	2022/3/23 7:00	2022/3/23 11:30	2022/3/23 13:30	2022/3/23 18:00	2022/10/18
3	2022/3/23 7:20	2022/3/23 12:00	2022/3/23 14:01	2022/3/23 17:00	2022/10/19
4	2022/3/23 8:00	2022/3/23 11:00	2022/3/23 13:01	2022/3/23 16:00	2022/10/16
4	2022/3/23 9:30	2022/3/23 10:40	2022/3/23 13:10	2022/3/23 16:30	2022/10/17
4	2022/3/23 8:30	2022/3/23 12:35	2022/3/23 13:12	2022/3/23 19:20	2022/10/18
4	2022/3/23 6:40	2022/3/23 13:00	2022/3/23 14:09	2022/3/23 18:10	2022/10/19

图 5.12 考勤数据导入模板

数据导入接口：

```
@PostMapping("/import")
public ResponseDTO imp(MultipartFile file)
```

接口说明：

此处只需要导入考勤数据表，系统读取数据，来对员工的考勤状态进行判断。

核心代码：

```

@Transactional(rollbackFor = Exception.class)
public ResponseDTO imp(MultipartFile file) throws IOException {
    InputStream inputStream = file.getInputStream();
    List<Attendance> list = HutoolExcelUtil.readExcel(inputStream, 1, Attendance.class);
    for (Attendance attendance : list) {
        // 判断是否是周末，如果是周末就不用记录考勤情况，如果不是周末就判断员工是否请假
        if (attendance.getStaffId() == null || attendance.getAttendanceDate() == null ||
            DateUtil.isWeekend(attendance.getAttendanceDate()) ||
            isLeave(attendance)) {
            continue;
        } else {
            WorkTime workTime =
this.workTimeMapper.findDeptWorkTimeByStaffId(attendance.getStaffId());
            if (isAbsenteeism(attendance, workTime)) {
                attendance.setStatus(AttendanceStatusEnum.ABSENTEEISM);
            } else if (isLate(attendance, workTime)) {
                attendance.setStatus(AttendanceStatusEnum.LATE);
            } else if (isLeaveEarly(attendance, workTime)) {
                attendance.setStatus(AttendanceStatusEnum.LEAVE_EARLY);
            } else {
                attendance.setStatus(AttendanceStatusEnum.NORMAL);
            }
            QueryWrapper<Attendance> queryWrapper = new QueryWrapper<>();
            queryWrapper.eq("staff_id", attendance.getStaffId()).eq("attendance_date",
attendance.getAttendanceDate());
            if (!saveOrUpdate(attendance, queryWrapper)) {
                throw new
ServiceException(BusinessStatusEnum.DATA_IMPORT_ERROR);
            }
        }
    }
    return Response.success();
}

```

流程：

当系统将考勤数据读取了之后，会根据员工的 id 获取员工所在部门的上班考勤时间，然后将员工的打卡时间与部门规定的上班時間进行比对，若员工的四个打卡时间缺少一个就认定为旷工，如果员工既迟到又早退也视为旷工。判定出员工相应的考勤状态之后，就将考勤状态更新到数据库。

4.11 考勤月报表导出

通过汇总当月员工的考勤状况得到当月的员工考勤报表。

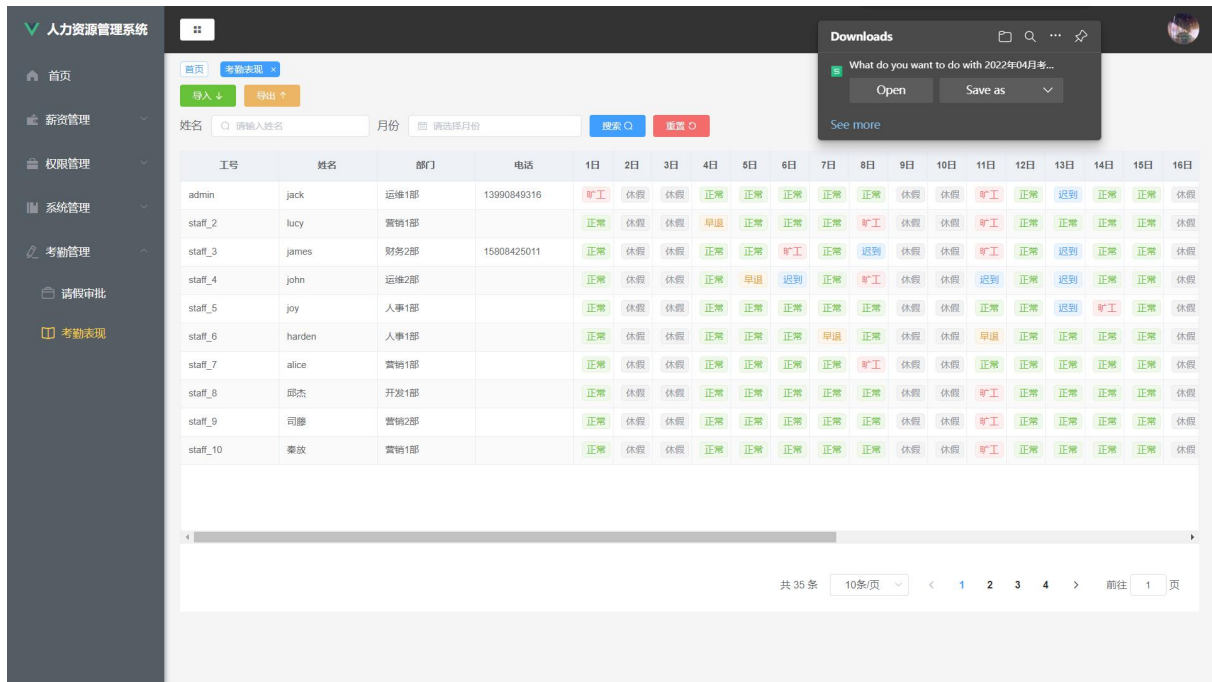


图 5.13 月考勤报表导出

2022年04月考勤报表									
员工工号	员工姓名	电话	地址	部门	迟到次数	早退次数	旷工次数	休假天数	
admin	jack	13990849315	北京	运维1部	0	0	1	8	
staff_2	lucy			营销1部	0	1	0	9	
staff_3	james	15808425010		财务2部	0	0	0	0	
staff_4	john			运维2部	2	1	0	0	
staff_5	joy			运维2部	1	0	1	0	
staff_6	harden			运维1部	0	1	0	0	
staff_7	alice			营销1部	0	0	0	0	
staff_8	邱杰		四川南充	开发1部	0	0	0	0	
staff_9	司藤			营销2部	0	0	0	0	
staff_10	秦放			营销1部	0	0	0	0	
staff_11	心悦			运维2部	0	0	0	0	
staff_12	邱圣		四川南充	开发1部	0	0	0	0	
staff_13	梧桐		北京	运维2部	0	0	0	0	
staff_14	福瑞		苍城山	财务3部	0	0	0	0	
staff_15	瓦房		苍城山	财务1部	0	0	0	0	
staff_16	白金		金陵	运维2部	0	0	0	0	
staff_17	世真		上海	运维2部	0	0	0	0	
staff_18	白英	13990849315	中海	财务1部	0	0	0	0	
staff_19	西竹	13990849315	禹航	营销1部	0	0	0	0	
staff_20	跳跳	13990849315	北京	运维2部	0	0	0	0	
staff_21	伊人	13990849315	杭州	财务2部	0	0	0	0	
staff_22	秋冬	13990849315	成都	财务3部	0	0	0	0	
staff_23	林拜			运维2部	0	0	0	0	
staff_24	老板			运维1部	0	0	0	0	
staff_25	秋雅	13990849315	武汉	营销1部	0	0	0	0	
staff_26	秋水	13990849315	武汉	开发2部	0	0	0	0	
staff_27	秋天		武汉	财务2部	0	0	0	0	
staff_28	邱卡	123	北京	运维2部	0	0	0	0	

图 5.14 月考勤报表

数据导出接口：

```
@GetMapping("/export/{month}")
public ResponseDTO export(HttpServletResponse response, @PathVariable String month)
```

接口说明：

根据月份，导出相应月份的考勤数据。

核心代码：


```

public ResponseDTO export(HttpServletResponse response, String month) throws
IOException {
    List<AttendanceMonthVO> list = this.staffMapper.findAttendanceMonthVO();
    for (AttendanceMonthVO attendanceMonthVO : list) {

attendanceMonthVO.setLateTimes(this.attendanceMapper.countTimes(attendanceMonthVO.
getStaffId(),AttendanceStatusEnum.LATE.getCode(), month));

attendanceMonthVO.setLeaveEarlyTimes(this.attendanceMapper.countTimes(attendanceMon
thVO.getStaffId(),AttendanceStatusEnum.LEAVE_EARLY.getCode(), month));

attendanceMonthVO.setAbsenteeismTimes(this.attendanceMapper.countTimes(attendanceMo
nthVO.getStaffId(),AttendanceStatusEnum.ABSENTEEISM.getCode(), month));
        List<Date> leaveDateList =
this.attendanceMapper.findLeaveDate(attendanceMonthVO.getStaffId(),
            AttendanceStatusEnum.LEAVE.getCode(), month);
        int count = 0;
        for (Date date : leaveDateList) {
            if (!DateUtil.isWeekend(date)) {
                count++;
            }
        }
        attendanceMonthVO.setLeaveDays(count);
    }
    String yearMonth = month.substring(0, 4) + "年" + month.substring(4) + "月";
    HutoolExcelUtil.writeExcel(response, list, yearMonth + "考勤报表",
AttendanceMonthVO.class);
    return Response.success();
}

```

流程：

首先获取所有员工的基本数据信息，然后根据员工 id 和月份查询员工在此月份迟到、早退、旷工的次数和休假的天数，此处休假的天数不包含周末。

4.12 工资调整

通过修改基本工资、以及生活补贴、奖金完成员工工资的基本调整。



图 5.15 工资调整界面

工资保存接口：

```
@PostMapping("/set")
public ResponseDTO setSalary(@RequestBody Salary salary)
```

接口说明：

当员工的基本工资信息填写完毕之后，将员工的工资数据保存到数据库。

核心代码：

```
public ResponseDTO setSalary(Salary salary) {
    QueryWrapper<Salary> query = new QueryWrapper<>();
    query.eq("month", salary.getMonth()).eq("staff_id", salary.getStaffId());
    if (saveOrUpdate(salary, query)) {
        return Response.success();
    }
    return Response.error();
}
```

流程：

当员工工资信息被提交之后，首先根据月份和员工 id 查询是否有当前员工的工资信息，如果没有就添加，有则更新。

4.13 月工资报表导出

通过汇总当月员工的考勤状况、以及社保、基本工资，得到员工当月的工资报表。

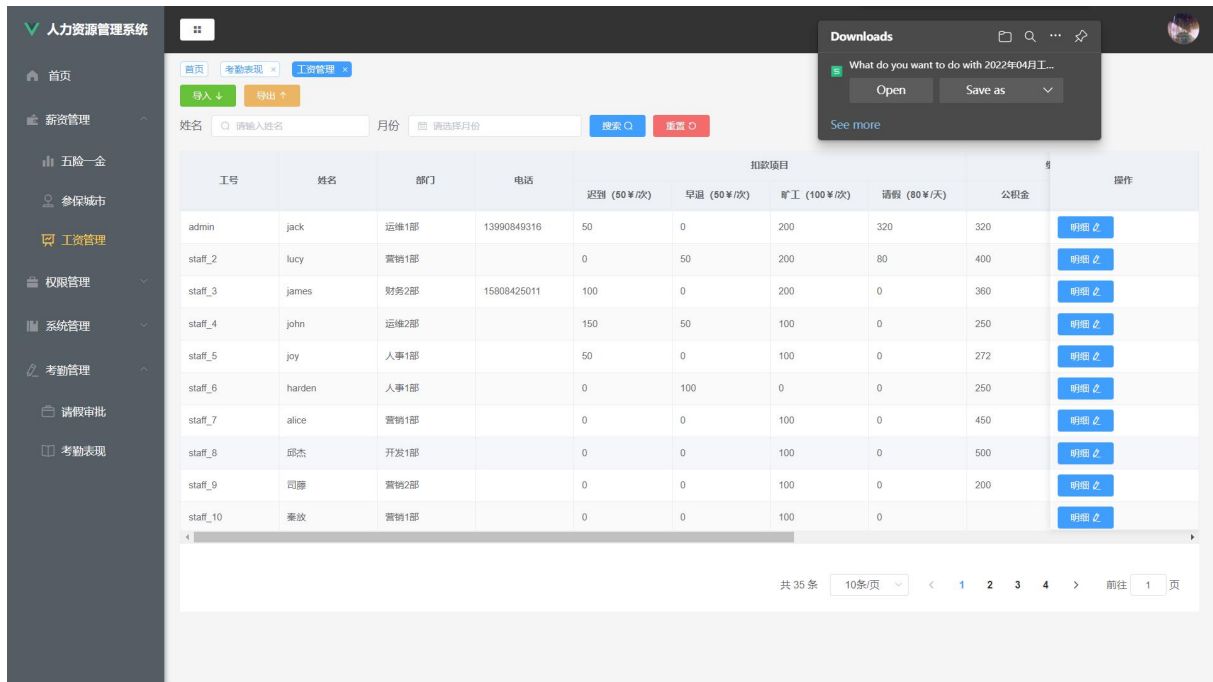


图 5.16 月工资报表导出

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	2022年04月工资报表															
2	员工工号	员工姓名	电话	地址	部门	迟到扣款	病假扣款	早退扣款	旷工扣款	公积金缴纳费用	社保缴纳费用	基本工资	生活补贴	奖金	最终工资	
3	admin	jack	13990849315	北京	运维1部	0.00	640.00	0.00	100.00	320.00	945.00	7000.00	400.00	200.00	5595.00	
4	staff_2	lucy			营销1部	0.00	720.00	50.00	0.00	400.00	840.00	6000.00	500.00	200.00	5460.00	
5	staff_3	james	15808425010		财务2部	0.00	0.00	0.00	0.00	360.00	465.00	6000.00	0.00	0.00	5175.00	
6	staff_4	john			运维2部	100.00	0.00	50.00	0.00	250.00	840.00	6000.00	500.00	300.00	5710.00	
7	staff_5	joy			运维2部	50.00	0.00	0.00	100.00	272.00	1360.00	7000.00	0.00	0.00	5368.00	
8	staff_6	hardden			运维1部	0.00	0.00	50.00	0.00	250.00	1360.00	8000.00	0.00	0.00	6340.00	
9	staff_7	alice			营销1部	0.00	0.00	0.00	0.00	450.00	840.00	6000.00	1000.00	0.00	5710.00	
10	staff_8	邱杰		四川南充	开发1部	0.00	0.00	0.00	0.00	500.00	3400.00	6000.00	0.00	0.00	2100.00	
11	staff_9	司藤			营销2部	0.00	0.00	0.00	0.00	200.00	525.00	6000.00	0.00	0.00	5275.00	
12	staff_10	秦放			营销1部	0.00	0.00	0.00	0.00							
13	staff_11	心悅			运维2部	0.00	0.00	0.00	0.00							
14	staff_12	邱圣		四川南充	开发1部	0.00	0.00	0.00	0.00							
15	staff_13	梧桐		北京	运维2部	0.00	0.00	0.00	0.00							
16	staff_14	福瑞		苍城山	财务3部	0.00	0.00	0.00	0.00							
17	staff_15	瓦房		苍城山	财务1部	0.00	0.00	0.00	0.00							
18	staff_16	白金		金陵	运维2部	0.00	0.00	0.00	0.00							
19	staff_17	世真		上海	运维2部	0.00	0.00	0.00	0.00							
20	staff_18	白英	13990849315	中海	财务1部	0.00	0.00	0.00	0.00							
21	staff_19	西竹	13990849315	禹航	营销1部	0.00	0.00	0.00	0.00							
22	staff_20	跳跳	13990849315	北京	运维2部	0.00	0.00	0.00	0.00							
23	staff_21	伊人	13990849315	杭州	财务2部	0.00	0.00	0.00	0.00							
24	staff_22	秋冬	13990849315	成都	财务3部	0.00	0.00	0.00	0.00							
25	staff_23	林拜			运维2部	0.00	0.00	0.00	0.00							
26	boss	老板			运维1部	0.00	0.00	0.00	0.00							
27	staff_25	秋雅	13990849315	武汉	营销1部	0.00	0.00	0.00	0.00							
28	staff_26	秋水	13990849315	武汉	开发2部	0.00	0.00	0.00	0.00							
29	staff_27	秋天	13990849315	武汉	财务2部	0.00	0.00	0.00	0.00							
30	staff_28	邱飞	123	北京	运维2部	0.00	0.00	0.00	0.00							

图 5.17 月工资报表

报表导出接口：

```
@GetMapping("/export/{month}")
public ResponseDTO export(HttpServletResponse response, @PathVariable String month)
```

接口说明：

如果不选择月份，默认导出当前月份的工资报表。

核心代码：

```

private void setSalaryInfo(String month, List<StaffSalaryVO> list) {
    for (StaffSalaryVO staffSalaryVO : list) {
        BigDecimal lateDeduct =
BigDecimal.valueOf(this.attendanceMapper.countTimes(staffSalaryVO.getStaffId(),
AttendanceStatusEnum.LATE.getCode(), month) * 50);
        staffSalaryVO.setLateDeduct(lateDeduct);
        BigDecimal leaveEarlyDeduct =
BigDecimal.valueOf(this.attendanceMapper.countTimes(staffSalaryVO.getStaffId(),
AttendanceStatusEnum.LEAVE_EARLY.getCode(), month) * 50);
        staffSalaryVO.setLeaveEarlyDeduct(leaveEarlyDeduct);
        BigDecimal absenteeismDeduct =
BigDecimal.valueOf(this.attendanceMapper.countTimes(staffSalaryVO.getStaffId(),
AttendanceStatusEnum.ABSENTEEISM.getCode(), month) * 100);
        staffSalaryVO.setAbsenteeismDeduct(absenteeismDeduct);
        List<Date> leaveDateList =
this.attendanceMapper.findLeaveDate(staffSalaryVO.getStaffId(),
AttendanceStatusEnum.LEAVE.getCode(), month);
        int count = 0;
        for (Date date : leaveDateList) {
            if (!DateUtil.isWeekend(date)) {
                count++;
            }
        }
        BigDecimal leaveDeduct = (BigDecimal.valueOf(count * 80));
        staffSalaryVO.setLeaveDeduct(leaveDeduct);
        QueryWrapper<Salary> queryWrapper = new QueryWrapper<>();
        queryWrapper.eq("staff_id", staffSalaryVO.getStaffId()).eq("month", month);
        Salary one = getOne(queryWrapper);
        if (one != null) {
            staffSalaryVO.setBaseSalary(one.getBaseSalary()).setSubsidy(one.getSubsidy())
                .setBonus(one.getBonus()).setRemark(one.getRemark())
                .setTotalSalary(one.getBaseSalary()
                    .add(one.getBonus())
                    .add(one.getSubsidy())
                    .subtract(lateDeduct)
                    .subtract(leaveEarlyDeduct)
                    .subtract(absenteeismDeduct)
                    .subtract(leaveDeduct)
                    .subtract(staffSalaryVO.getSocialPay())
                    .subtract(staffSalaryVO.getHousePay()));
        }
    }
}

```

流程：

当进行数据导出时，系统首先统计员工当前月的迟到、早退、旷工的次数和休假的天数，再根据罚款规则得到相应的扣款金额。