# 技术报告

## 第一章 系统概述

随着企业薪酬管理的复杂性不断增加，有效管理员工工资成为了企业管理中的重要环节。某公司每个月发放的薪水包括基本工资、岗位津贴、午餐补贴、加班工资、全勤工资等项目，扣除社保、公积金、个人所得税及迟到请假等费用后，才是到手的实发工资。为优化公司薪酬管理，特设计开发一套人员工资管理系统。

系统主要功能如下：

1. 对部门进行查询和管理，包括建立部门，删除部门。
2. 对任意进行进行查询和管理，包括新增人员，删除人员等。
3. 对工资进行查询和管理，包括历史工资查询，工资导入等。
4. 基于SpringBoot框架和React框架，使用ant design组件库，开发实现了部门、人员和工资管理系统。每个功能模块都包括基本的增加、修改、删除和查询。

使用本课程设计实现的公司员工工资管理系统，可以确保工资计算准确、及时，并符合国家税法的相关规定，提高企业薪酬管理的效率和准确性。

## 第二章 系统框架

### 2．1 网络架构

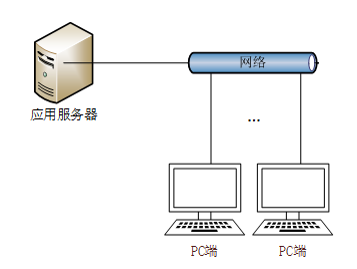


图 1 网络架构图

### 2．2 系统架构

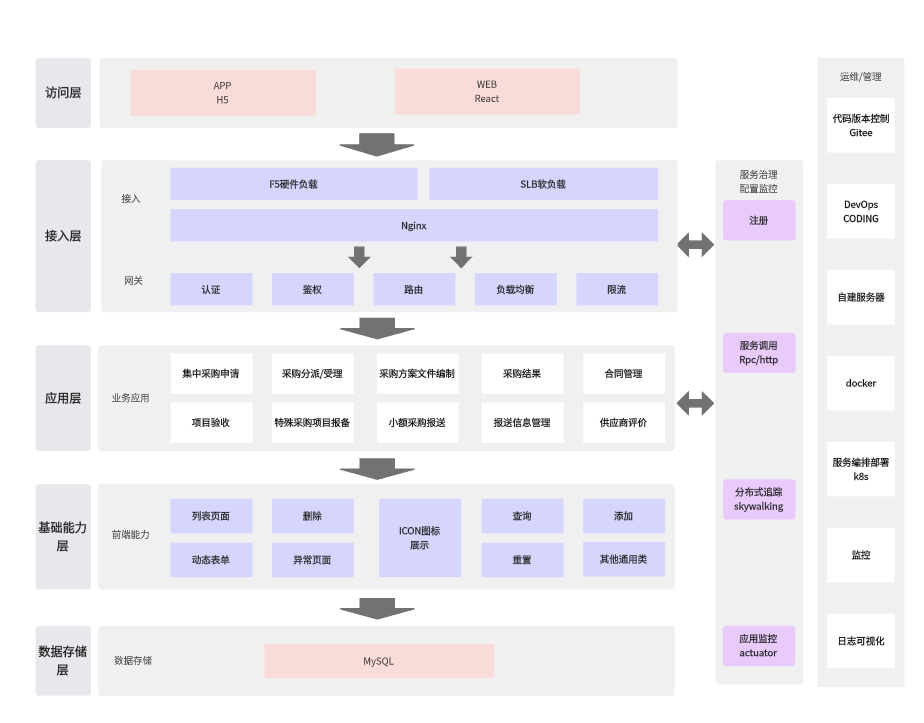


图 2 系统架构图

### 2．3 系统功能

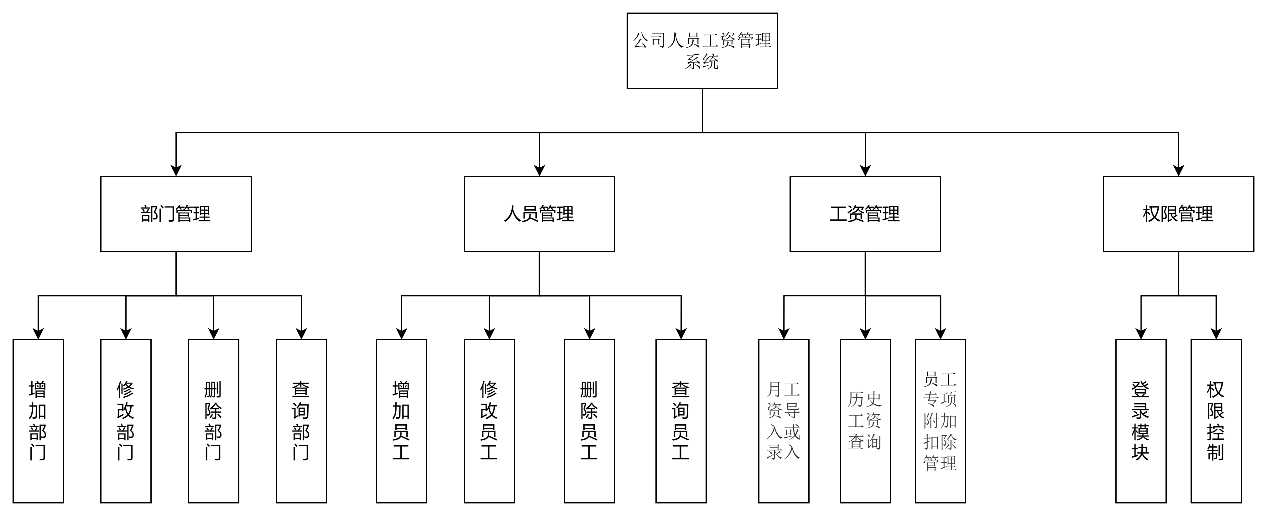


图 3 系统功能流程图

## 第三章 系统需求分析

### 3．1 登录模块

#### 3．1．1 登录功能

**1、功能描述**

输入用户名和密码后，按登录按钮，若输入错误则弹出警告框，若输入正确则进入欢迎页面。

**2、业务处理流程及要求**

（1）密码不能小于8位；

（2）若输入错误则弹出警告框；

（3）若输入正确则进入欢迎页面。

**3、输入信息** 用户名和密码。

**4、输出信息**

若输入错误则弹出警告框“您输入的用户名和密码有误”， 若输入正确则进入欢迎页面。

#### 3．1．2 注册功能

**1、功能描述**

输入用户名、密码和确认密码后，按注册按钮，若输入错误则弹出警告框，若输入正确显示注册成功。

**2、业务处理流程及要求**

（1）密码不能小于8位；

（2）若输入错误则弹出警告框；

（3）若输入正确则显示注册成功。

**3、输入信息** 用户名、密码和确认密码。

**4、输出信息**

若输入错误则弹出警告框“您输入的密码和确认密码不一致”， 若输入正确则显示注册成功。

由于部门管理、人员管理、工作管理功能较为相似所以这里以部门管理为例。

### 3．2 部门管理模块

#### 3．2．1 部门查询功能

**1、**功能描述

在筛选框中相应字段填入要匹配的信息，留空字段表示任意。点击查询后下方记录框中为查询到的信息。

**2、**业务处理流程及要求

点击查询按钮后，前端把收集到的信息转为json对象，通过post方法传回后端，后端通过Controller层完成相应查询在把结果传回前端

**3、**输入信息  
部门的相应字段具体值

**4、**输出信息  
部门查询结果

#### 3．2．2 部门增加功能

**1、**功能描述

点击增加按钮，在出来的表单中把相应字段的内容填进去，最后点确认。

**2、**业务处理流程及要求

点击增加按钮后，前端调出React额外表单，用户填好信息后通过Post方法传回后端，后端进行相应增加操作。

**3、**输入信息  
部门的相应字段具体值

**4、**输出信息  
无，后台数据库信息增加

#### 3．2．3 部门修改功能

**1、**功能描述

点击修改按钮，在出来的表单中把相应字段的内容填进去，最后点确认。

**2、**业务处理流程及要求

点击增加按钮后，前端调出React额外表单，用户填好信息后通过Post方法传回后端，后端进行相应修改操作。

**3、**输入信息  
部门的相应字段具体值

**4、**输出信息  
当前查询结果中相应记录的结果会更新

#### 3．2．4 部门删除功能

**1、**功能描述

点击查询结果具体记录的删除按钮，该记录会被删除。

**2、**业务处理流程及要求

点击删除按钮后，前端j将该记录在数据表中的主键值返回给后端，后端进行相应删除操作。

**3、**输入信息  
选中相应记录

**4、**输出信息  
在查询结果中该记录小消失。

## 第四章 系统详细设计

### 4．1 登录模块

登录模块的uml类图如下。

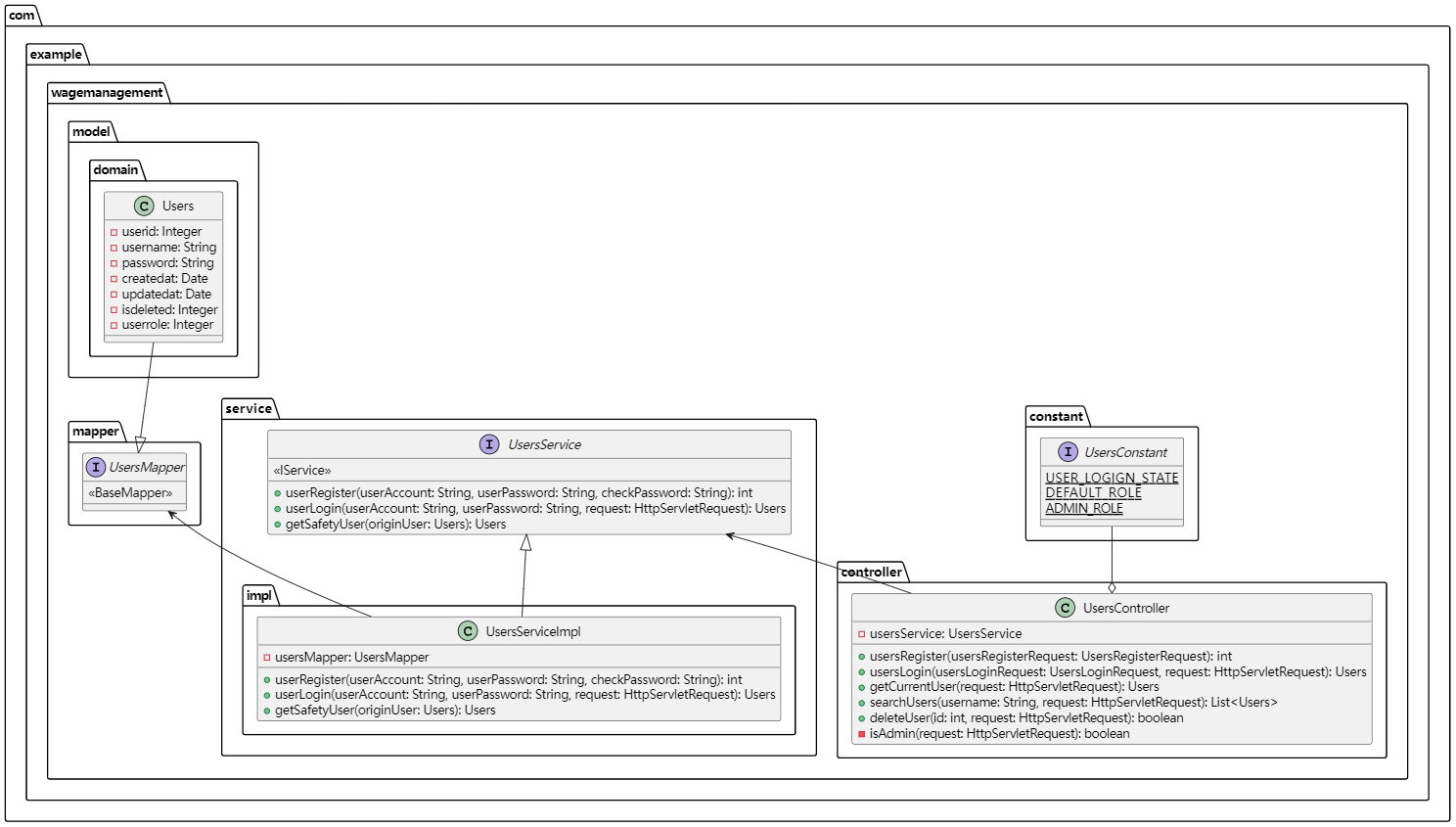


图 4 登录模块的uml类图

表 1 React页面：index.tsx

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 描述 | 用户登录界面 | | | |
| 所在工程目录 | src/pages/User/Login/index.tsx | | | |
| 加载的文件 | 文件名 | | 描述 | |
| src/components/Footer/index.tsx | | 页脚 | |
| 主要的typescript函数 | 名称 | 属性 | | 描述 |
| handleSubmit | 异步函数 | | 登录 |

表 2 函数1的说明

|  |  |
| --- | --- |
| 函数名称 | handleSubmit |
| 描述 | 处理用户的登录请求 |
| 输入参数 | values：这是一个对象，其类型为 API.LoginParams。这个对象包含了用户的登录信息，如用户名和密码。 |
| 步骤 | 1. 尝试调用 login 函数，这个函数接收一个对象作为参数，这个对象的属性来自于 values，并且包含一个 type 属性。 2. 如果 login 函数调用成功，那么显示一个成功的提示信息 '登录成功！'，然后调用 fetchUserInfo 函数获取用户的信息，并将页面重定向到之前的页面或者首页。 3. 如果 login 函数调用失败，那么设置用户的登录状态为失败。 4. 如果在调用 login 函数过程中发生错误，那么显示一个错误提示信息 '登录失败，请重试！'。 |
| 输出参数 | return true或false |
| 注释 | 这个函数可能被用在登录表单的提交事件中，当用户填写完表单并提交时，调用这个函数来处理用户的登录请求。 |

表 3 model类

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | Users | | |
| 继承或实现 | Java.io.Serializable接口 | | |
| 描述 | 用户的model | | |
| 类别 | 实体类 | | |
| 包 | com.example.wagemanagement.model.domain | | |
| 与其它类的调用关系 | 无 | | |
| 属性 | 名称 | 类型 | 描述 |
| userid | Integer | 用户编号 |
| username | String | 用户名 |
| password | String | 加密后的密码 |
| createdat | Date | 创建时间 |
| updatedat | Date | 更新时间 |
| isdeleted | Integer | 逻辑删除标志 |
| userrole | Integer | 用户角色，0-普通用户，1-用户管理员 |

表 4 Controller类

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | UsersController | | |
| 继承或实现 | 无 | | |
| 描述 | 部门的controller | | |
| 类别 | 接口 | | |
| 包 | com.example.wagemanagement.controller | | |
| 与其它类的调用关系 | 会调用 usersService | | |
| 方法 | 名称 | 类型 | 描述 |
| usersRegister | public int | 注册 |
| usersLogin | public Users | 登录 |
| getCurrentUser | public Users | 获取当前用户登录态 |
| deleteUser | boolean | 删除用户 |
| isAdmin | boolean | 判断是否为管理员 |

表 5 方法1的说明

|  |  |
| --- | --- |
| 方法名称 | usersRegister |
| 描述 | 这个方法用于处理用户注册的请求。它接收一个 UsersRegisterRequest 对象作为输入，这个对象包含了用户的账号、密码和确认密码。如果这个对象为 null，或者其中任何一个字段为空，这个方法就会返回 -1。否则，它会调用 usersService.userRegister 方法来注册用户，并返回注册结果。 |
| 输入参数 | 这个方法有一个输入参数 usersRegisterRequest，它是 UsersRegisterRequest 类型。这个对象应该包含以下字段：   * userAccount：用户的账号，它是一个字符串。 * userPassword：用户的密码，它是一个字符串。 * checkPassword：用户的确认密码，它是一个字符串。 |
| 步骤 | 1. 检查 usersRegisterRequest 是否为 null。如果是，返回 -1。 2. 从 usersRegisterRequest 中获取 userAccount、userPassword 和 checkPassword。 3. 检查这三个字段是否有任何一个为空。如果是，返回 -1。 4. 调用 usersService.userRegister 方法，传入 userAccount、userPassword 和 checkPassword，并返回结果。 |
| 输出参数 | 这个方法返回一个整数。如果 usersRegisterRequest 为 null 或者其中任何一个字段为空，它会返回 -1。否则，它会返回 usersService.userRegister 方法的返回值。 |

表 6 方法2的说明

|  |  |
| --- | --- |
| 方法名称 | usersLogin |
| 描述 | 这个方法用于处理用户登录的请求。它接收一个 UsersLoginRequest 对象作为输入，这个对象包含了用户的账号和密码。如果这个对象为 null，或者其中任何一个字段为空，这个方法就会返回 null。否则，它会调用 usersService.userLogin 方法来验证用户的登录信息，并返回登录结果。 |
| 输入参数 | 这个方法有两个输入参数：   * usersLoginRequest：它是 UsersLoginRequest 类型。这个对象应该包含以下字段： * userAccount：用户的账号，它是一个字符串。 * userPassword：用户的密码，它是一个字符串。 * request：它是 HttpServletRequest 类型。这个对象包含了 HTTP 请求的所有信息。 |
| 步骤 | 1. 检查 usersLoginRequest 是否为 null。如果是，返回 null。 2. 从 usersLoginRequest 中获取 userAccount 和 userPassword。 3. 检查这两个字段是否有任何一个为空。如果是，返回 null。 4. 调用 usersService.userLogin 方法，传入 userAccount、userPassword 和 request，并返回结果。 |
| 输出参数 | 这个方法返回一个 Users 对象。如果 usersLoginRequest 为 null 或者其中任何一个字段为空，它会返回 null。否则，它会返回 usersService.userLogin 方法的返回值，这个值是一个 Users 对象，它包含了用户的所有信息。 |

表 7 方法3的说明

|  |  |
| --- | --- |
| 方法名称 | getCurrentUser |
| 描述 | 这个方法用于获取当前登录的用户。它从 HTTP 请求的会话中获取用户对象，然后使用这个对象的 userid 字段来从 usersService 中获取用户的详细信息。最后，它调用 usersService.getSafetyUser 方法来获取用户的安全信息，并返回这个信息。 |
| 输入参数 | 这个方法有一个输入参数 request，它是 HttpServletRequest 类型。这个对象包含了 HTTP 请求的所有信息。 |
| 步骤 | 1. 从 request 的会话中获取 USER\_LOGIGN\_STATE 属性，然后把这个属性转换为 Users 对象。 2. 检查 currentUser 是否为 null。如果是，返回 null。 3. 从 currentUser 中获取 userid。 4. 调用 usersService.getById 方法，传入 userid，并获取用户的详细信息。 5. 调用 usersService.getSafetyUser 方法，传入用户的详细信息，然后获取用户的安全信息。 6. 返回用户的安全信息。 |
| 输出参数 | 这个方法返回一个 Users 对象。如果 currentUser 为 null，它会返回 null。否则，它会返回 usersService.getSafetyUser 方法的返回值，这个值是一个 Users 对象，它包含了用户的安全信息。 |

表 8 方法4的说明

|  |  |
| --- | --- |
| 方法名称 | deleteUser |
| 描述 | 这个方法用于处理删除用户的请求。首先，它会检查当前的用户是否是管理员，如果不是，这个方法就会返回 false。然后，它会检查输入的 id 是否大于 0，如果不是，这个方法也会返回 false。最后，如果以上两个条件都满足，它会调用 usersService.removeById 方法来删除指定 id 的用户，并返回删除结果。 |
| 输入参数 | 这个方法有两个输入参数：   1. id：它是 int 类型。这个参数表示要删除的用户的 id。 2. request：它是 HttpServletRequest 类型。这个对象包含了 HTTP 请求的所有信息。 |
| 步骤 | 1. 调用 isAdmin 方法，传入 request，并检查返回结果。如果返回 false，这个方法就会返回 false。 2. 检查 id 是否大于 0。如果不是，这个方法就会返回 false。 3. 调用 usersService.removeById 方法，传入 id，并返回结果。 |
| 输出参数 | 这个方法返回一个布尔值。如果当前的用户不是管理员，或者 id 不大于 0，它会返回 false。否则，它会返回 usersService.removeById 方法的返回值，这个值表示删除操作的结果。 |

表 9 方法5的说明

|  |  |
| --- | --- |
| 方法名称 | isAdmin |
| 描述 | 这个方法用于判断当前的用户是否是管理员。它从 HTTP 请求的会话中获取用户对象，然后检查这个对象的 userrole 字段是否等于 ADMIN\_ROLE。如果等于，这个方法就会返回 true，否则，它就会返回 false。 |
| 输入参数 | 这个方法有一个输入参数 request，它是 HttpServletRequest 类型。这个对象包含了 HTTP 请求的所有信息。 |
| 步骤 | 1. 从 request 的会话中获取 USER\_LOGIGN\_STATE 属性，然后把这个属性转换为 Users 对象。 2. 检查 users 是否为 null，并且 users 的 userrole 字段是否等于 ADMIN\_ROLE。如果都满足，返回 true，否则，返回 false。 |
| 输出参数 | 这个方法返回一个布尔值。如果 users 为 null，或者 users 的 userrole 字段不等于 ADMIN\_ROLE，它会返回 false。否则，它会返回 true。 |

### 4．2 部门管理模块

由于部门管理、人员管理、工作管理功能较为相似所以这里以部门管理为例，其uml类图如下。

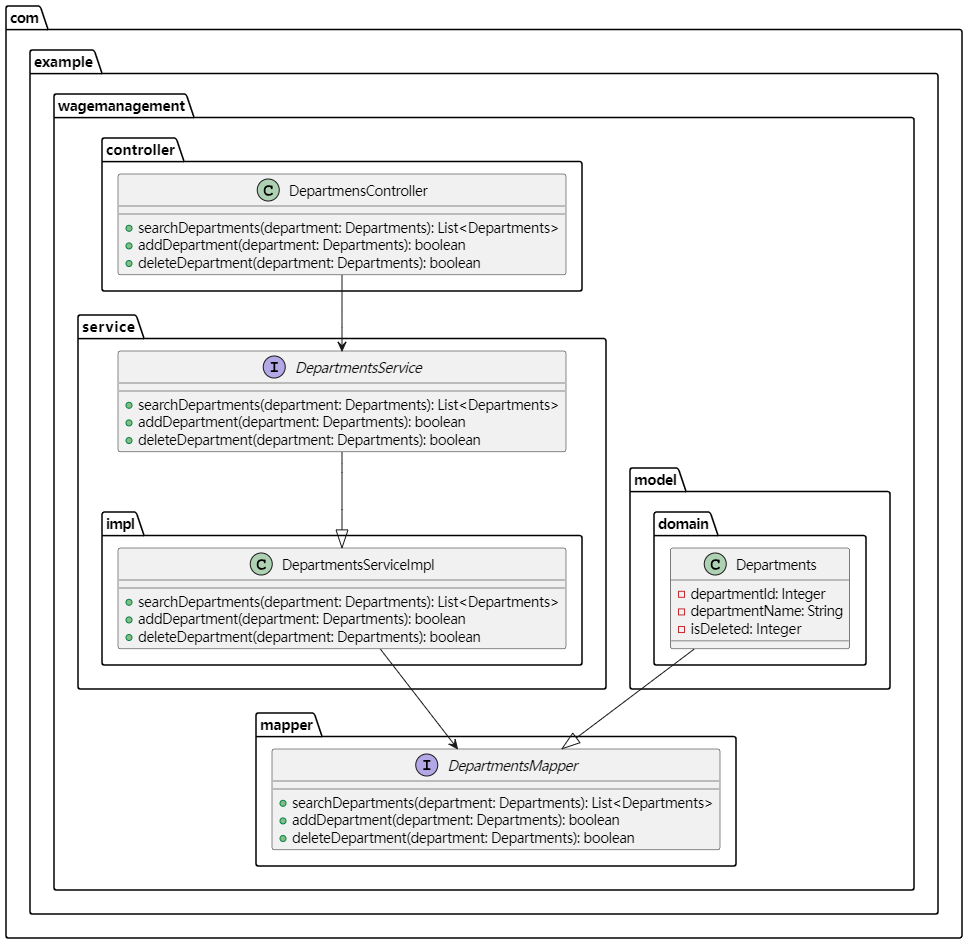


图 5 部门管理模块的uml类图

表 10 React页面：index.tsx

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 描述 | 部门管理界面 | | | |
| 所在工程目录 | src/pages/TableList/index.tsx | | | |
| 加载的文件 | 文件名 | | 描述 | |
| Icons.tsx | | 图标文件 | |
| UpdateForm.tsx | | 表格配置文件 | |
| 主要的typescript函数 | 名称 | 属性 | | 描述 |
| handleAdd | 异步函数 | | 新建部门 |
| handleRemove | 异步函数 | | 删除部门 |

表 11 函数1的说明

|  |  |
| --- | --- |
| 函数名称 | handleAdd |
| 描述 | 添加一个新的部门 |
| 输入参数 | fields：这是一个对象，其类型为 API.CurrentDepartment。这个对象包含了添加新部门所需要的所有信息。 |
| 步骤 | 1. 显示一个加载信息 '正在添加'。 2. 尝试调用 addDepartment 函数，这个函数接收一个对象作为参数，这个对象的属性来自于 fields。 3. 如果 addDepartment 函数调用成功，那么隐藏加载信息，并显示一个成功的提示信息 'Added successfully'。 4. 如果在调用 addDepartment 函数过程中发生错误，那么隐藏加载信息，并显示一个错误提示信息 'Adding failed, please try again!' |
| 输出参数 | return true或false |
| 注释 | 用在表单的提交事件中，当用户填写完表单并提交时，调用这个函数来添加新的部门。 |

表 12 函数2的说明

|  |  |
| --- | --- |
| 函数名称 | handleRemove |
| 描述 | 删除一个新的部门 |
| 输入参数 | departmentName：这是一个字符串，代表要删除的部门的名称。 |
| 步骤 | 1. 显示一个加载信息 '正在删除'，并将返回的隐藏函数赋值给 hide。 2. 检查 departmentName 是否存在。如果不存在（即 departmentName 是空或者 undefined），则函数直接返回 true。 3. 尝试调用 removeDepartment 函数，这个函数接收一个对象作为参数，这个对象的 name 属性值为 departmentName。 4. 如果 removeDepartment 函数调用成功，那么隐藏加载信息，并显示一个成功的提示信息 'Deleted successfully and will refresh soon'。 5. 如果在调用 removeDepartment 函数过程中发生错误，那么隐藏加载信息，并显示一个错误提示信息 'Delete failed, please try again'。 |
| 输出参数 | return true或false |
| 注释 | 用在删除部门的操作中，当用户点击删除按钮并确认删除时，调用这个函数来删除指定的部门。 |

表 13 model类

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | Departments | | |
| 继承或实现 | Java.io.Serializable接口 | | |
| 描述 | 部门的model | | |
| 类别 | 实体类 | | |
| 包 | com.example.wagemanagement.model.domain | | |
| 与其它类的调用关系 | 无 | | |
| 属性 | 名称 | 类型 | 描述 |
| departmentId | Integer | 部门编号 |
| departmentName | String | 部门名 |
| isDeleted | String | 是否被删除 |

表 14 Controller类

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | DepartmentsController | | |
| 继承或实现 | 无 | | |
| 描述 | 部门的controller | | |
| 类别 | 接口 | | |
| 包 | com.example.wagemanagement.controller | | |
| 与其它类的调用关系 | 会调用 employeesService | | |
| 方法 | 名称 | 类型 | 描述 |
| searchDepartments | public List<Departments> | 查找部门 |
| addDepartment | public boolean | 增加或修改部门 |
| deleteDepartment | public boolean | 删除 |

表 15 方法1的说明

|  |  |
| --- | --- |
| 方法名称 | searchDepartments |
| 描述 | 查找部门 |
| 输入参数 | 前端返回的 Json对象转换为的Departments Model |
| 步骤 | 读取该对象的departmentId和departmentName，利用查询构造器 QueryWrapper<>()从department表读取所有和该对象相同的对象，存储在departmentlist中 |
| 输出参数 | List<Departments> |

表 16 方法2的说明

|  |  |
| --- | --- |
| 方法名称 | addDepartment |
| 描述 | 增加或修改部门 |
| 输入参数 | 前端返回的 Json对象转换为的Departments Model |
| 步骤 | 先利用查询构造器判断数据库中是否存在相同的departmentId记录，若有，则删除。最后在增加该对象转换成的记录 |
| 输出参数 | boolean |

表 17 方法3的说明

|  |  |
| --- | --- |
| 方法名称 | deleteDepartment |
| 描述 | 删除部门 |
| 输入参数 | 前端返回的 Json对象转换为的Departments Model |
| 步骤 | 先利用查询构造器把员工表中所有在该部门的员工都删除了，在在部门表中把当前部门删除了。 |
| 输出参数 | boolean |

## 第五章 数据库设计

### 5．1 数据库表概念结构设计

表 18 实体1：用户(users)

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 意义 | 类型 | 值域 |
| userid | 用户编号 | INT | 长度<=4 |
| username | 用户名 | VARCHAR | 长度<=20 |
| password | 用户密码 | VARCHAR | 长度<=50 |
| createdat | 创建日期 | TIMESTAMP | '1970-01-01 00:00:01' UTC到 '2038-01-19 03:14:07' UTC |
| updatedat | 更新日期 | TIMESTAMP | '1970-01-01 00:00:01' UTC 到 '2038-01-19 03:14:07' UTC |
| isdeleted | 逻辑删除 | TINYINT | 长度<=2 |
| userrole | 用户权限 | INT | 长度<=4 |

表 19 实体2：部门（departments）

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 意义 | 类型 | 值域 |
| departmentId | 部门编号 | INT | 长度<=4 |
| departmentName | 部门名称 | VARCHAR | 长度<=255 |
| isDeleted | 逻辑删除 | TINYINT | 长度<=2 |

表 20 实体3：员工（employees）

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 意义 | 类型 | 值域 |
| employeeId | 员工编号 | INT | 长度<=4 |
| name | 员工姓名 | VARCHAR | 长度<=255 |
| departmentId | 部门编号 | INT | 长度<=4 |
| isDeleted | 逻辑删除 | TINYINT | 长度<=2 |

表 21 实体4：工资（salaries）

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 意义 | 类型 | 值域 |
| salaryId | 工资编号 | INT | 长度<=4 |
| employeeId | 员工编号 | INT | 长度<=4 |
| year | 年 | INT | 长度<=4 |
| month | 月 | INT | 长度<=4 |
| workDays | 工作天数 | INT | 长度<=4 |
| actualWorkDays | 实际工作天数 | INT | 长度<=4 |
| basicSalary | 基本工资 | DECIMAL | 长度<=10 |
| positionAllowance | 岗位津贴 | DECIMAL | 长度<=10 |
| lunchAllowance | 午餐补贴 | DECIMAL | 长度<=10 |
| overtimeSalary | 加班工资 | DECIMAL | 长度<=10 |
| fullAttendanceBonus | 全勤工资 | DECIMAL | 长度<=10 |
| socialInsurance | 社保 | DECIMAL | 长度<=10 |
| housingFund | 公积金 | DECIMAL | 长度<=10 |
| tax | 个人所得税 | DECIMAL | 长度<=10 |
| deductions | 额外勾扣除 | DECIMAL | 长度<=10 |
| netSalary | 实际工资 | DECIMAL | 长度<=10 |
| isDeleted | 逻辑删除 | TINYINT | 长度<=2 |

### 5．2 数据库表概念关系模型

画出E-R图，列出表与表之间的关系图

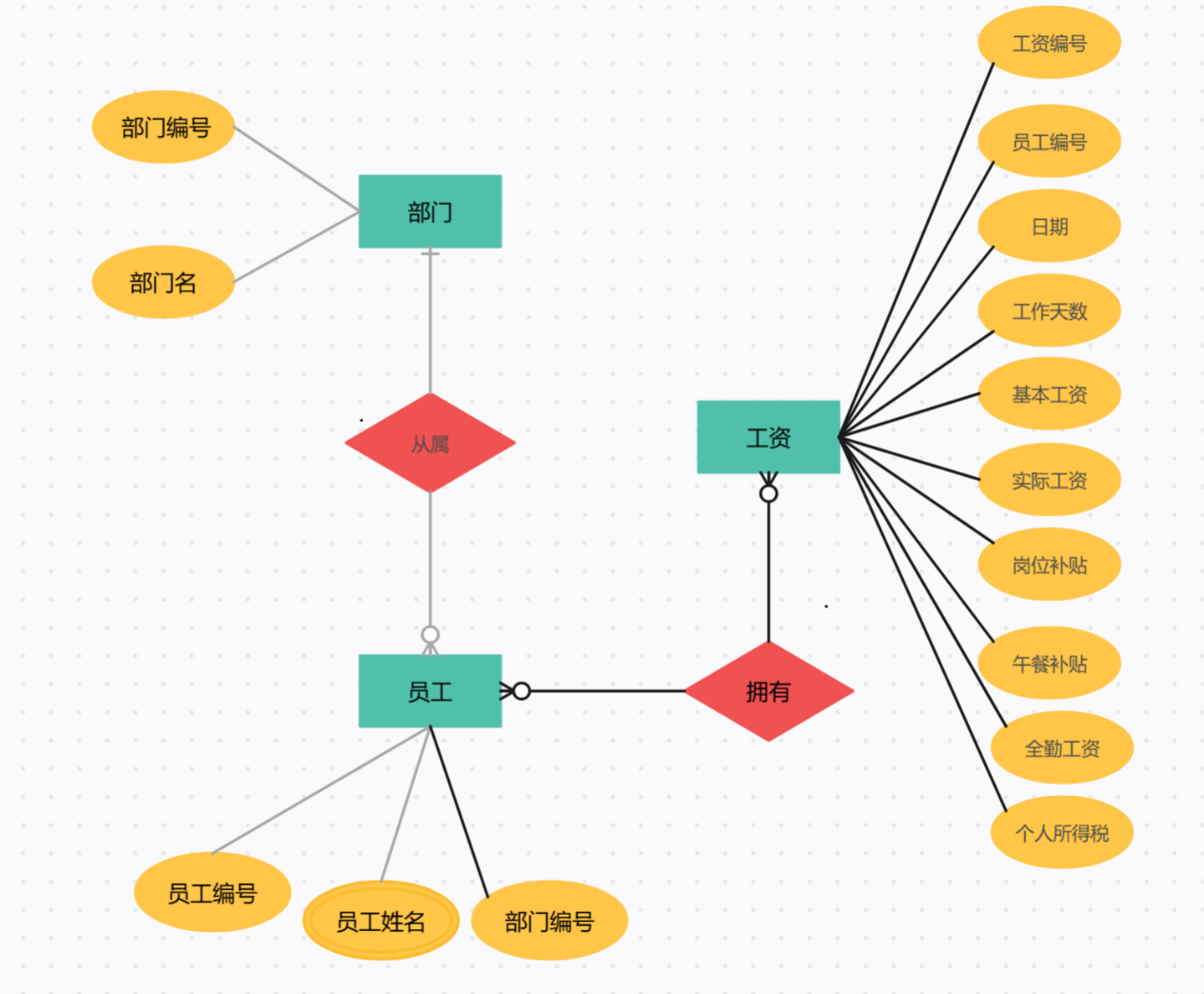


图 6 E-R图

## 第六章 系统实现功能

### 6．1 登录模块：登录、注册

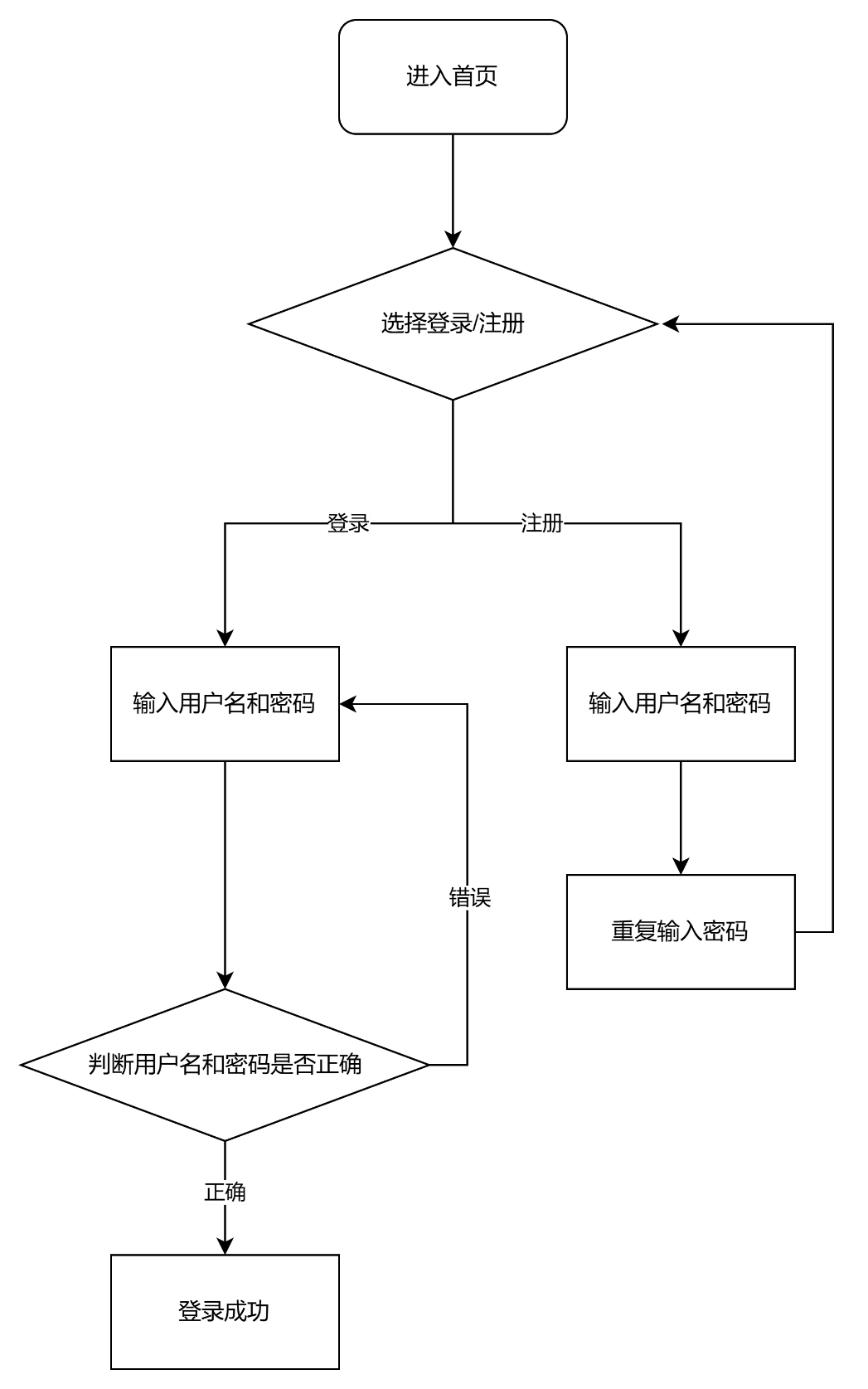


图 7 登录模块流程图

具体功能演示详见系统说明书。

### 6．2 部门管理

由于部门管理、人员管理、工作管理功能较为相似所以这里以部门管理为例。

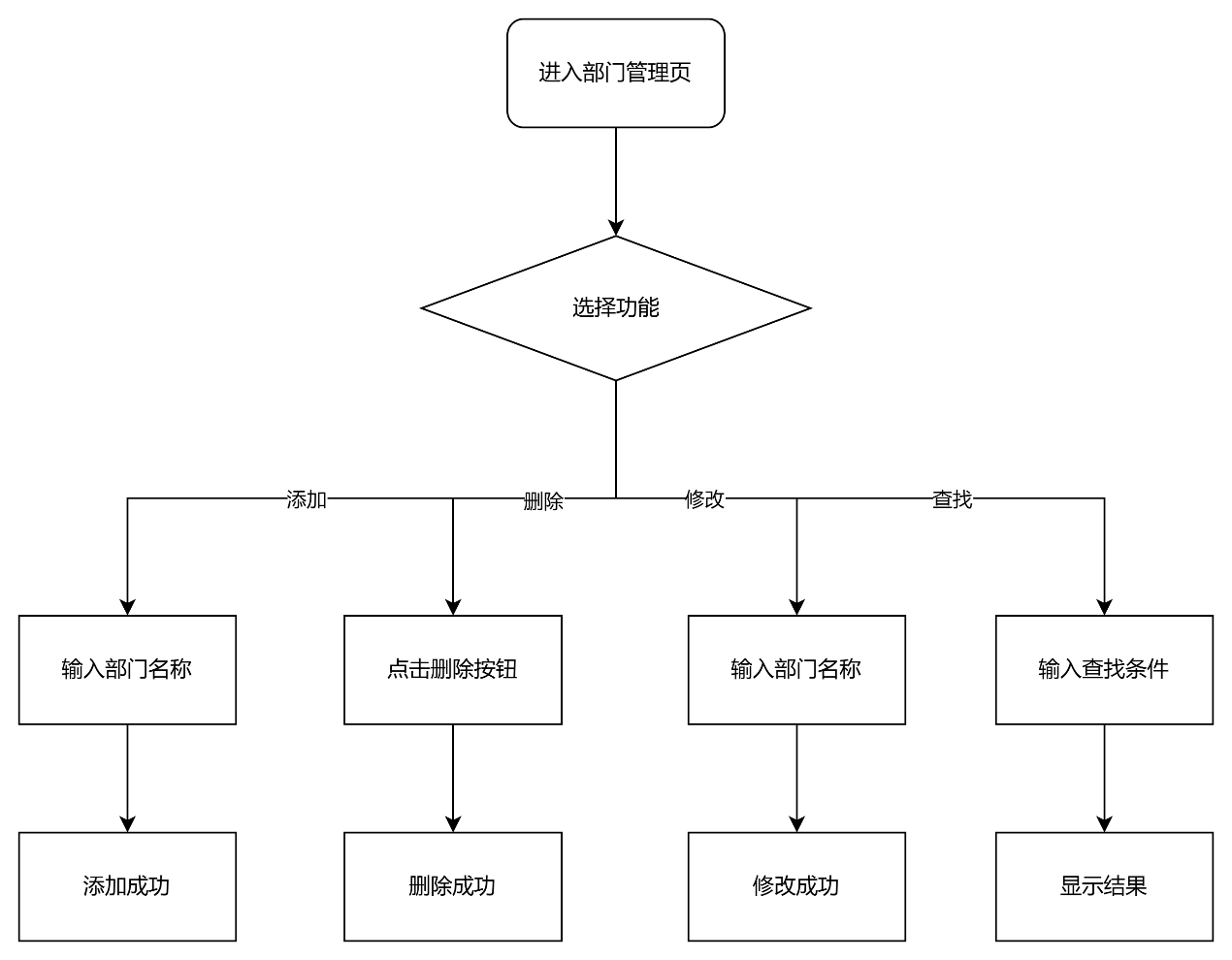


图 8 部门管理流程图

具体功能演示详见系统说明书。

## 第七章 程序清单

Departments.java: 部门的模型

Employees.java 雇员的模型

Salaries.java 工资的模型

User.java: 用户的模型

DepartmensController.java 部门的控制层

EmployeeController.java 员工的控制层

SalariesController.java 工资的控制层

UserController.java 用户的控制层

DepartmentsMapper.java 部门的DAO层

EmployeeMapper.java 员工的DAO层

SalariesMapper.java 工资的DAO层

UserMapper.java 用户的DAO层

DepartmentService.java 部门服务接口

DepartmentServiceImpl.java 部门服务实现

EmployeeService.java 员工服务接口

EmployeeServiceImpl.java 员工服务实现

SalariesService.java 工资服务接口

SalariesServiceImpl.java 工资服务实现

UserService.java 用户服务接口

UserServiceImpl.java 用户服务实现