**考勤系统设计文档**



专 业： 2018级软件工程

组长姓名： 库尔班·阿布都克力木

组长学号： 3018218059

组员姓名： 李朋恩

组员学号： 3018216263

组员姓名： 平云

组员学号： 3018216269

组员姓名： 麦尼完尔·阿卜杜热依木

组员学号： 3018218060

授课教师： 李罡

2020年5 月

目录

[一、系统框架 4](#_Toc42634884)

[1、系统总体设计结构： 4](#_Toc42634885)

[2、系统开发环境： 4](#_Toc42634886)

[3、系统总体功能 4](#_Toc42634887)

[4、系统框架图 5](#_Toc42634888)

[二、controller层与前端界面 6](#_Toc42634889)

[1、登陆模块 6](#_Toc42634890)

[2、员工打卡模块 6](#_Toc42634891)

[4、部门管理模块 10](#_Toc42634892)

[5、出差管理模块 11](#_Toc42634893)

[6、请假管理模块 12](#_Toc42634894)

[7、数据统计模块 13](#_Toc42634895)

[8、系统设置模块 14](#_Toc42634896)

[三、service层与dao层及接口设计 15](#_Toc42634897)

[1、总UML图 15](#_Toc42634898)

[2、系统身份认证及登陆模块 15](#_Toc42634899)

[3、员工管理模块 16](#_Toc42634900)

[4、部门管理模块 17](#_Toc42634901)

[5、考勤管理模块 18](#_Toc42634902)

[6、请假管理模块 19](#_Toc42634903)

[7、出差管理模块 21](#_Toc42634904)

[8、数据统计模块 23](#_Toc42634905)

[9、系统日志模块 24](#_Toc42634906)

[四、数据库的设计与实现 24](#_Toc42634907)

[1、数据库config的设计 24](#_Toc42634908)

[2、数据库department的设计 25](#_Toc42634909)

[3、数据库holiday的设计 26](#_Toc42634910)

[4、数据库role的设计 27](#_Toc42634911)

[5、数据库schedule的设计 28](#_Toc42634912)

[6、数据库user的设计 30](#_Toc42634913)

[7、数据库work\_check的设计 31](#_Toc42634914)

[五、小组分工 32](#_Toc42634915)

# 一、系统框架

## 1、系统总体设计结构：

服务器：apache tomcat，

数据库：mysql，

后端： spring-boot+mybatis，

前端：bootstrap+vue.js

## 2、系统开发环境：

MacOs操作系统

IntelliJ IDEA 2020.1.1版本

Mysql数据库

端口号：localhost8081

## 3、系统总体功能

1. 提供员工登陆功能，在登陆后可以进行打卡，请假申请，外出申请等功能。
2. 提供考勤信息，请假申请信息，外出申请信息查询的功能，并且信息的查看可以共享。

3）员工提交请假申请后提交给领导进行审批，为部门领导提供请假申请审核功能，并可以进行批准或不批准。

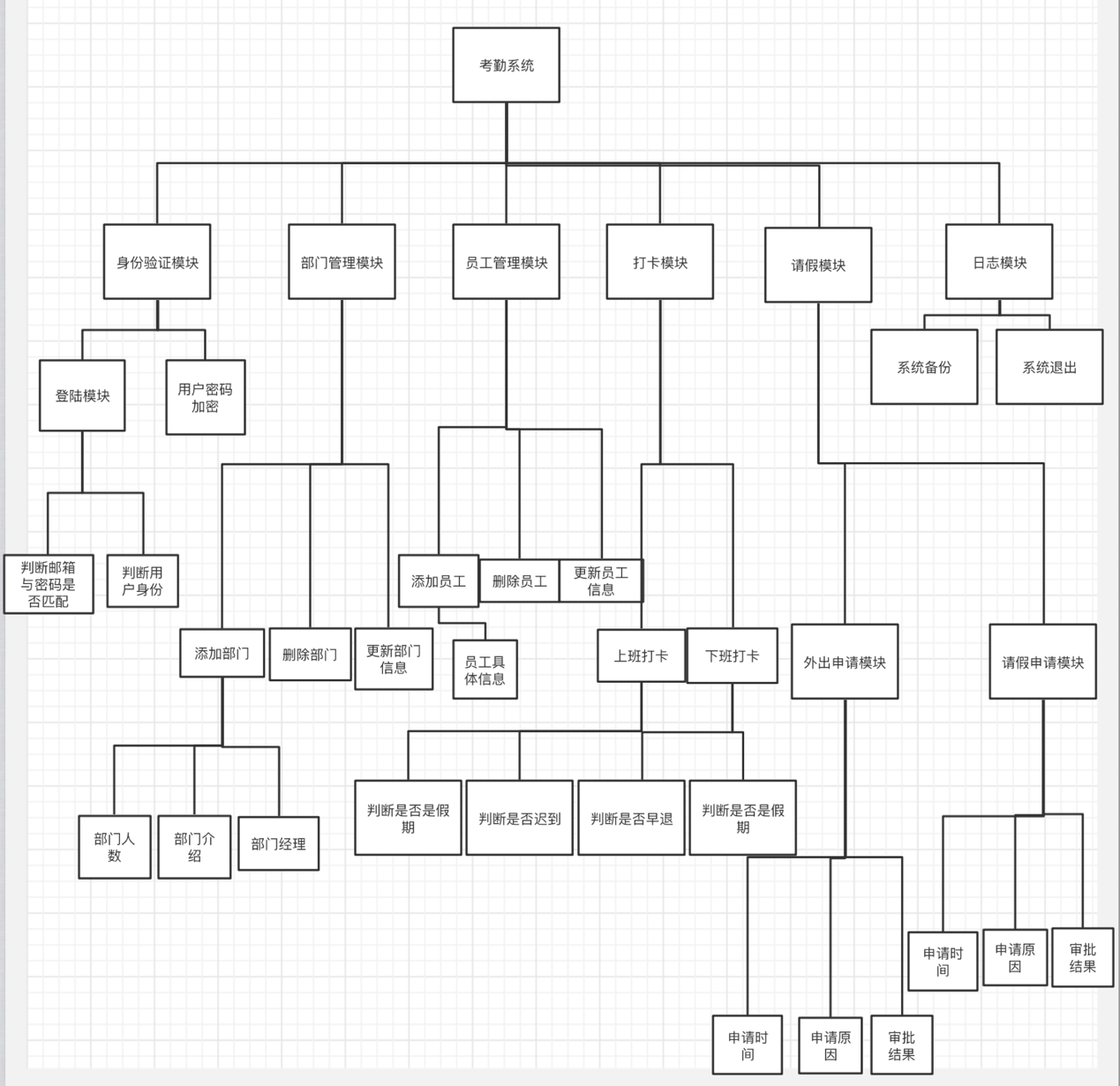
3）提供领导审批功能，使部门经理可以对本部门员工的请假申请和外出申请进行审批。

4）为部门领导提供员工管理功能，可以对员工进行添加，删除，修改员工信息。

5）提供部门管理功能，总经理可以进行部门的添加，删除，修改。

6）系统记录日志，方便管理员进行系统的管理，管理员可以查看和修改系统日志，对系统进行备份和初始化。

## 4、系统框架图



# 二、controller层与前端界面

## 1、登陆模块

功能描述：用户输入自己的邮箱和密码，系统首先验证用户输入的用户名与密码是否匹配，其次判断用户的身份为管理员、普通员工、部门经理或总经理。实现路径：使用本地端口号8081，登录地https://localhost:8081/login进入登录页面。

输入流:员工邮箱地址，员工密码

逻辑实现:系统验证所输入的员工邮箱地址和员工密码与后台数据库记录是否正确，系统通过用户名判断用户角色，对其进行权限的限定。

输出:若所输入的员工邮箱地址和员工密码正确，则允许员工进入首页。若所输入的员工邮箱地址和员工密码不正确，则输出错误信息，并且重新返回登陆界面。

登陆界面



## 2、员工打卡模块

功能描述：

1）员工上班前和下班时点击“打卡”按钮，系统自动记录下员工的邮箱号和上班下班时间。

2）系统管理员可对假期进行设置，方便财务部对于员工是否在工作日进行打卡进行有效的判断。

3）系统管理员可以对于上班时间进行设置，便于系统准确的判断员工是否迟到早退。

实现路径：使用本地端口号8081，通过<http://localhost:8081/work>进入打卡页面。

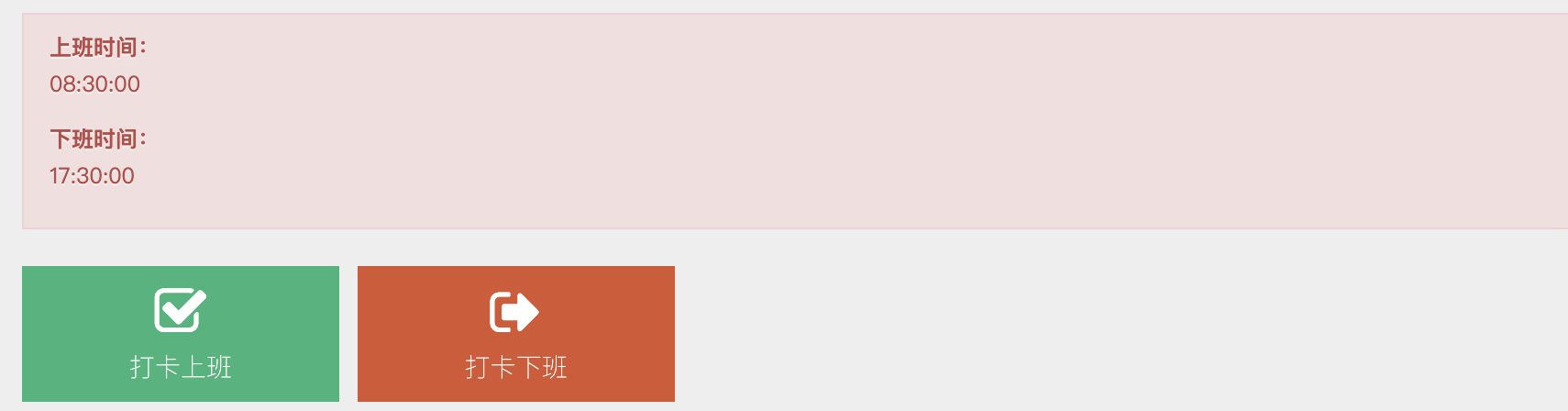
操作方式 ：

1. 员工点击“打卡上班”和“打卡下班”按钮进行打卡
2. 管理员点击假期设置即可进行假期的设定
3. 管理员点击上班时间设置即可进行上下班时间的规定

逻辑实现：调用service层，若第一次打卡时间超过设定上班时间，则系统将记为迟到；若第二次打卡时间早于设定下班时间，则系统将记为早退。当天可以重复打卡但只有第一次和最后一次有效系统时间、员工邮箱号保存到数据库，提示打卡成功。通过POST实现请求。

输出 ：提示打卡成功或者今天已经打过卡。

点击上班打卡，下班打卡，系统即进行记录



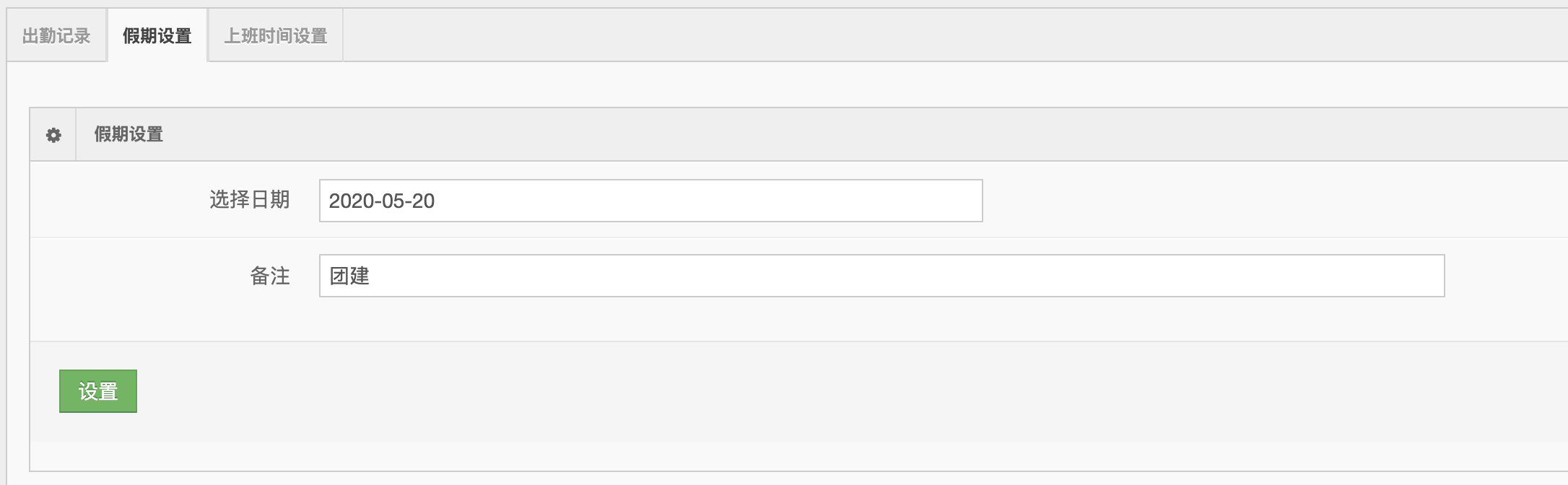
打卡超过1次，系统进行提示



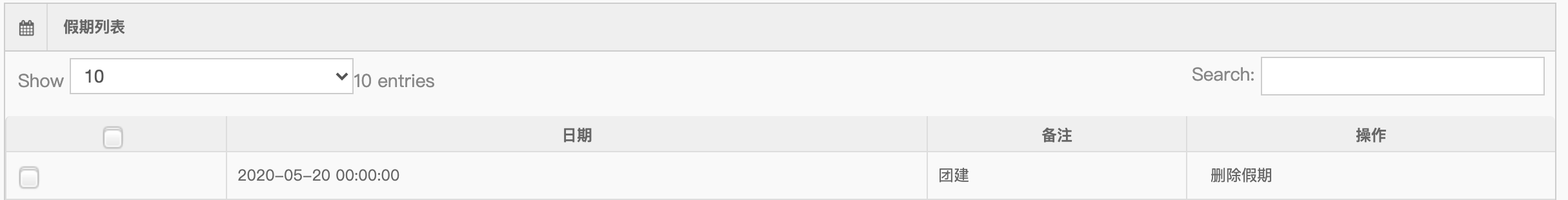
查看出勤记录



进行假期设置



假期设置查看



进行上班时间设置

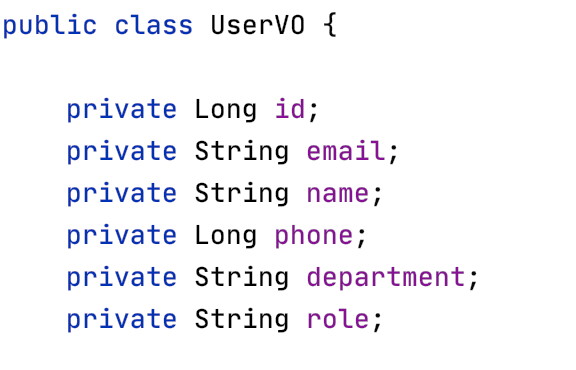
3、员工管理模块

功能描述：

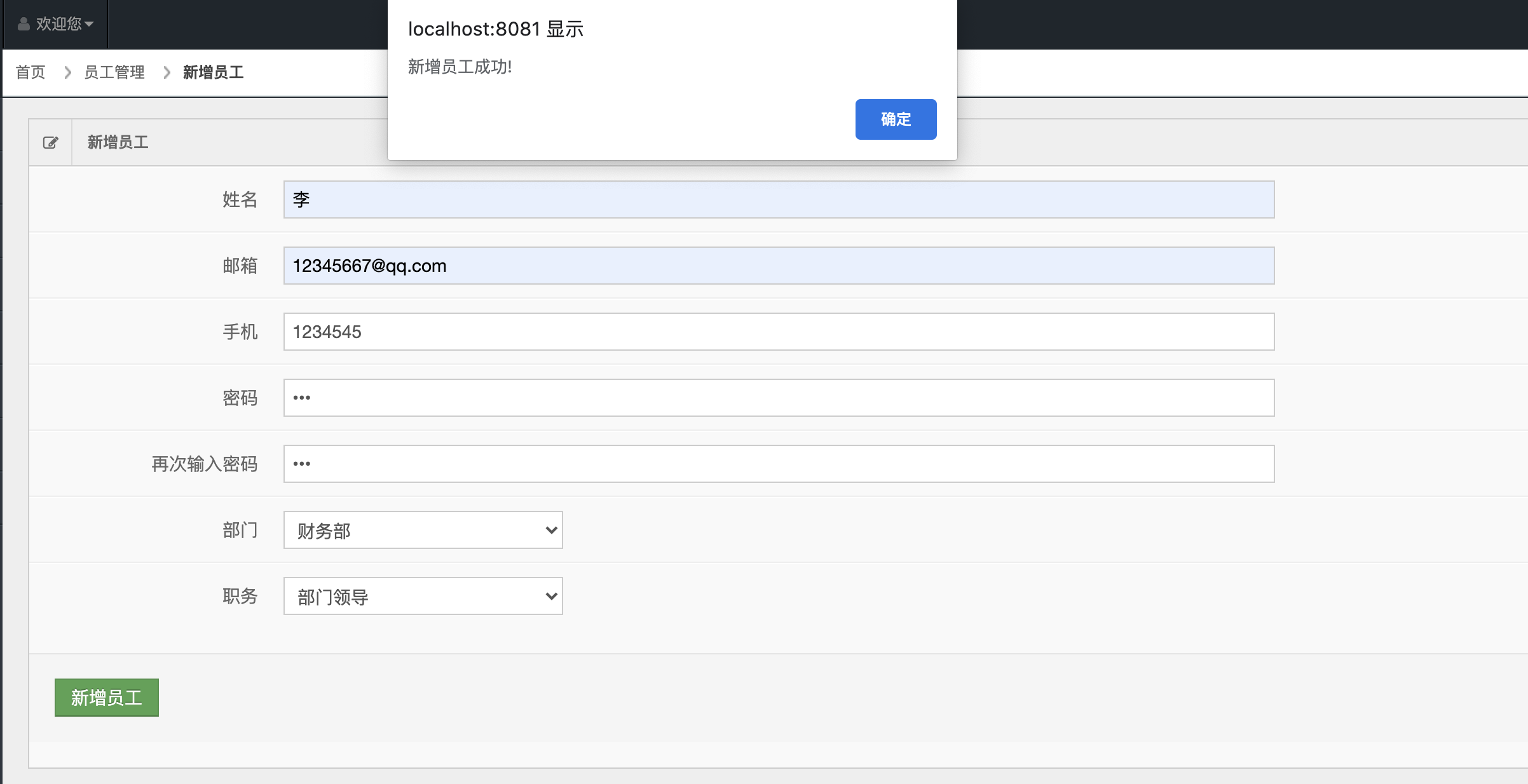
1. 添加员工，需要输入员工姓名，邮箱，手机号，选择员工所属部门，设置员工职务，设置员工登陆密码。且邮箱号不能重复，姓名、员工权限不能为空。
2. 修改员工信息，点击修改信息即可修改
3. 支持搜索有关信息
4. 查看员工信息，员工信息共享，可以查看员工姓名，所属部门，级别，邮箱和手机号，方便企业员工自己联系交流。支持对于员工的搜索。

实现路径：使用本地端口号8081，通过<http://localhost:8081/employee>进入员工管理页面。

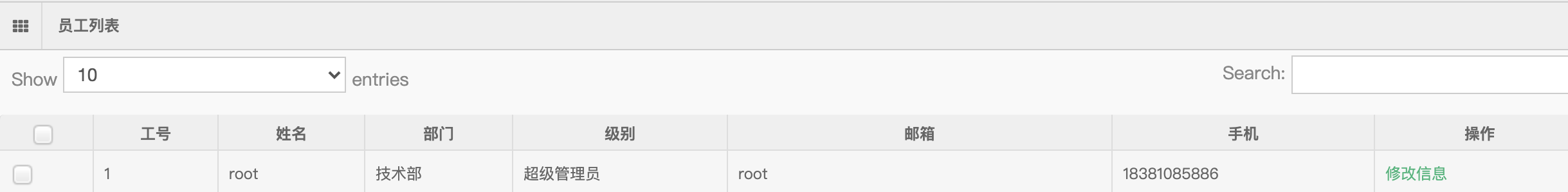
逻辑实现：调用service层的用户信息，角色信息，部门信息，获得员工信息并且对于添加员工的权限进行分配。通过POST实现添加员工请求，通过PUT实现更新员工信息请求，通过DELETE实现删除员工请求。下图为员工的vo设计。



添加员工，输入员工信息



修改查看员工信息



搜索员工信息



## 4、部门管理模块

功能描述：

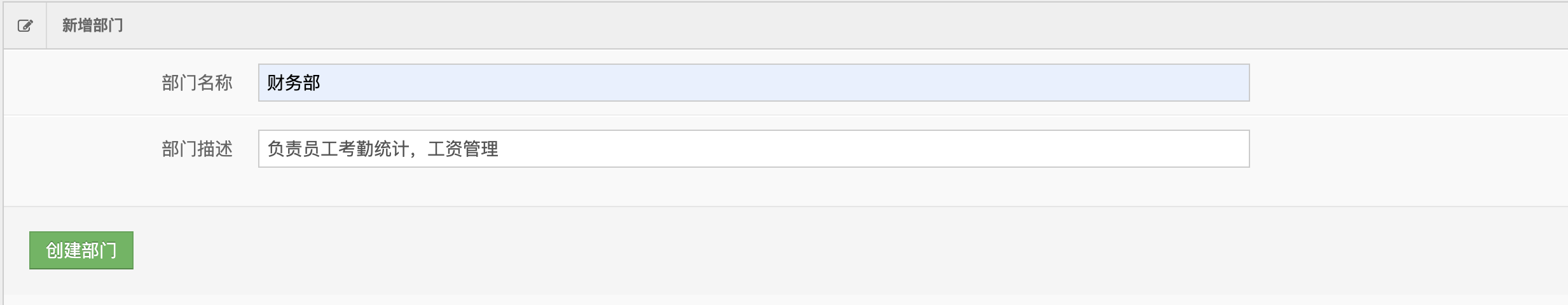
1. 添加部门，需要输入部门名称，部门描述。
2. 删除部门，点击删除即可进行删除
3. 支持搜索部门信息

4）查看部门信息，实现部门信息共享，可以查看部门名称，部门描述，部门人数，部门领导

实现路径：使用本地端口号8081，通过<http://localhost:8081/department>进入部门管理页面。

逻辑实现：调用service层的用户信息，部门信息，获得用户信息，统计各部门员工人数并且获得各部门领导姓名。通过POST实现添加部门请求，通过DELETE实现删除部门请求

添加部门





部门查看



部门搜索



部门删除



## 5、出差管理模块

功能描述：

1. 员工可以进行出差申请，选择开始日期和结束日期，写明出差原因
2. 部门领导和总经理可以对于员工的出差申请进行审批或者拒绝

3) 员工可以查看出差记录，出差记录信息进行共享，可以查看出差原因和出差时间及状态：批准，不批准。

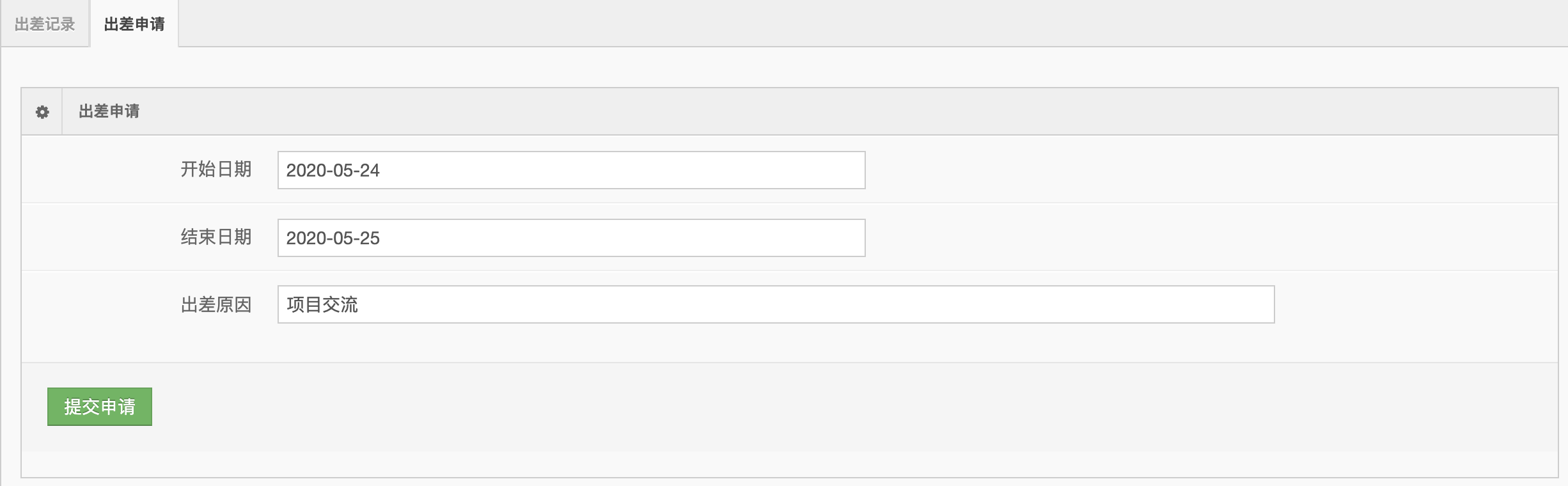
实现路径：使用本地端口号8081，通过<http://localhost:8081/buzz>进入出差管理页面。

逻辑实现：通过调用service层的schedule模块实现出差申请功能，前端通过POST实现提交申请，通过DELETE实现申请的删除，通过PUT实现更新请求。通过返回json格式实现前端与controller层的信息交互。调用role来判断登陆角色的权限，实现部门经理对请假的审批。

出差记录查看



出差申请



## 6、请假管理模块

功能描述：

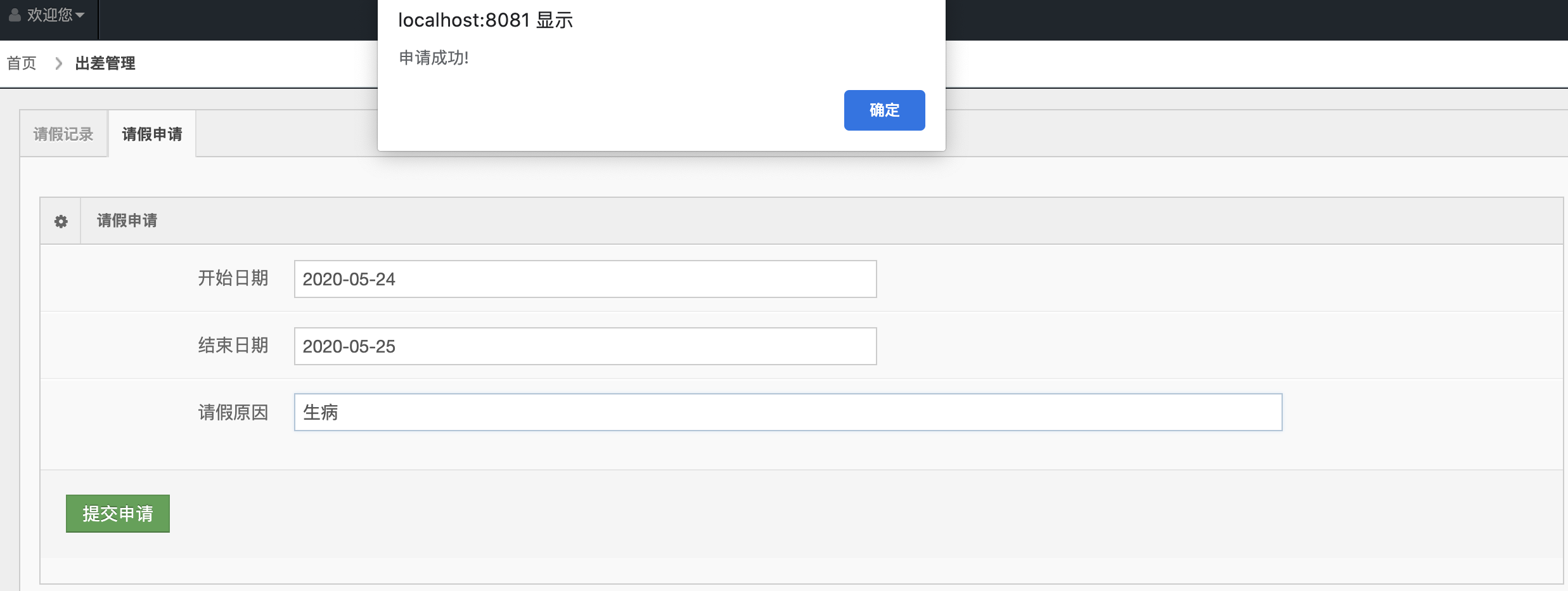
1. 员工可以进行请假申请，选择开始日期和结束日期，写明请假原因
2. 部门领导和总经理可以对于员工的请假申请进行审批或者拒绝

3) 员工可以查看请假记录，请假记录信息进行共享，可以查看请假原因和请假时间及状态：批准，不批准。

实现路径：使用本地端口号8081，通过<http://localhost:8081/leave>进入出差管理页面。

逻辑实现：通过调用service层的schedule模块实现请假申请功能，前端通过POST进行提交申请，通过DELETE进行申请的删除，通过PUT更新请求。通过返回json格式实现前端与controller层的信息交互。调用role来判断登陆角色的权限，实现部门经理对请假的审批。

请假申请



请假记录查看



## 7、数据统计模块

功能描述：实现出勤记录，请假记录，出差记录的信息共享

实现路径：使用本地端口号8081，通过<http://localhost:8081/stat>进入出差管理页面。

逻辑实现：通过调用service层的workservice，userservice，scheduleservice实现信息的查看。







## 8、系统设置模块

功能描述：管理员可以进行系统日志查看，修改和删除，可以实现系统备份和初始化

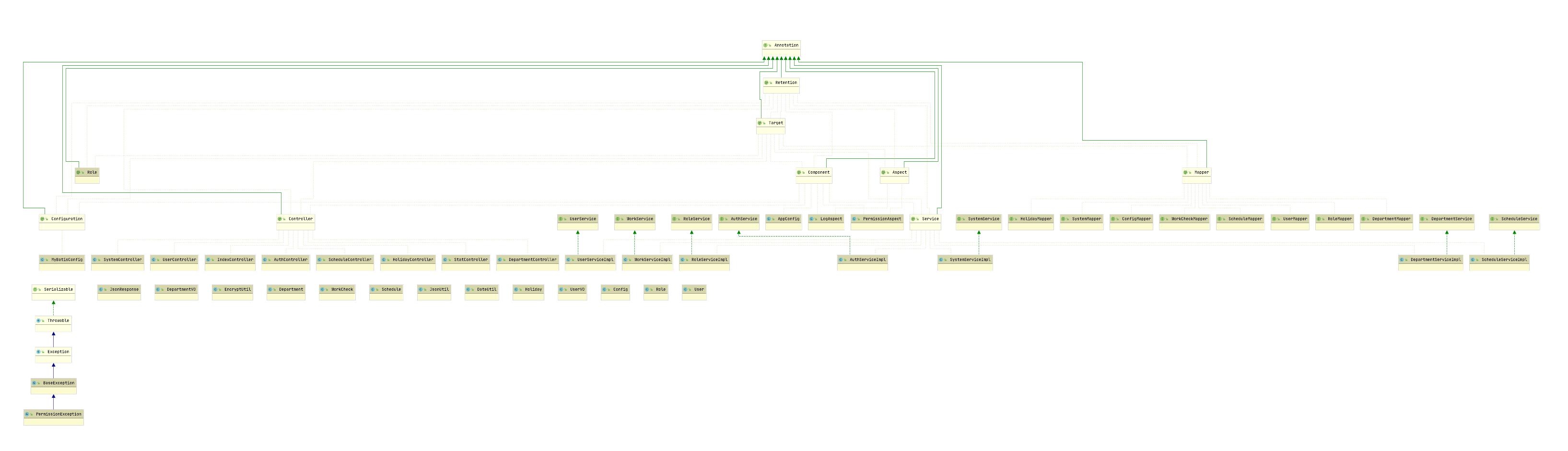
实现路径：使用本地端口号8081，通过<http://localhost:8081/system>进入出差管理页面。

逻辑实现：通过POST请求实现

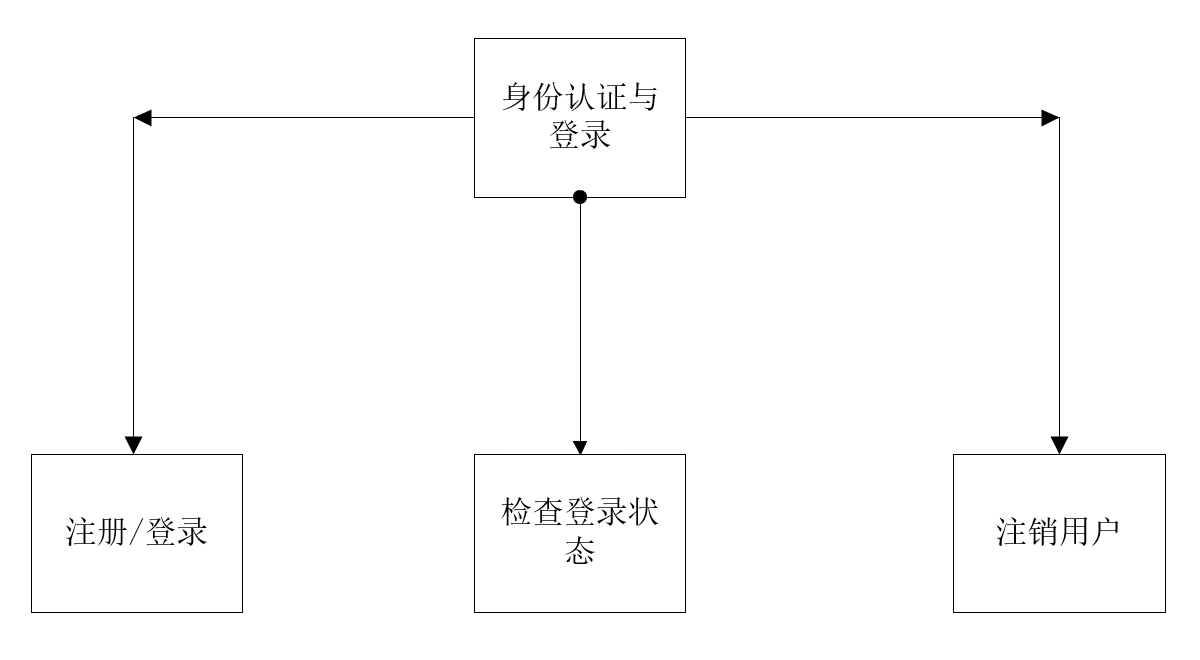


# 三、service层与dao层及接口设计

## 1、总UML图

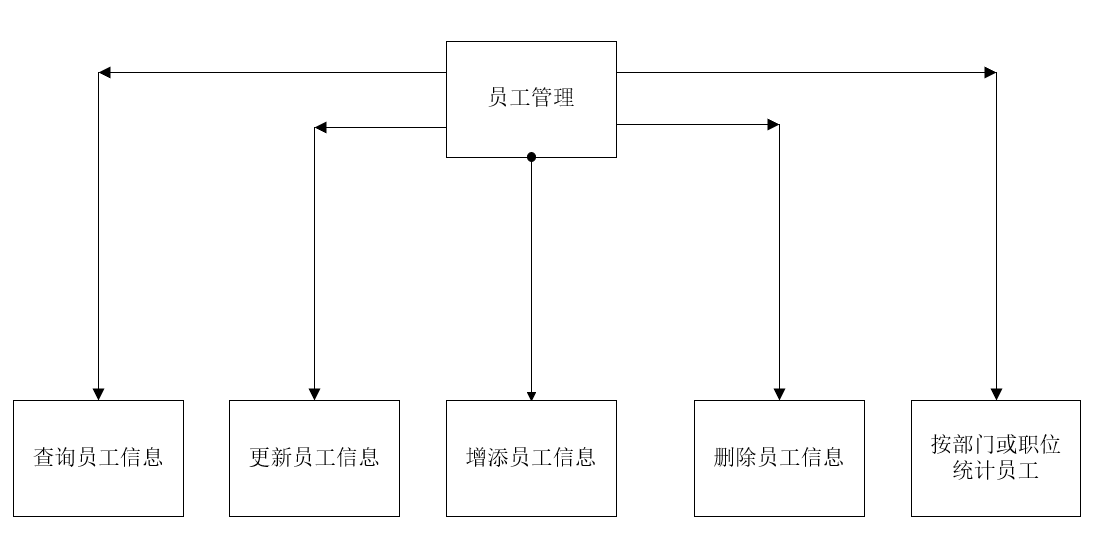


## 2、系统身份认证及登陆模块



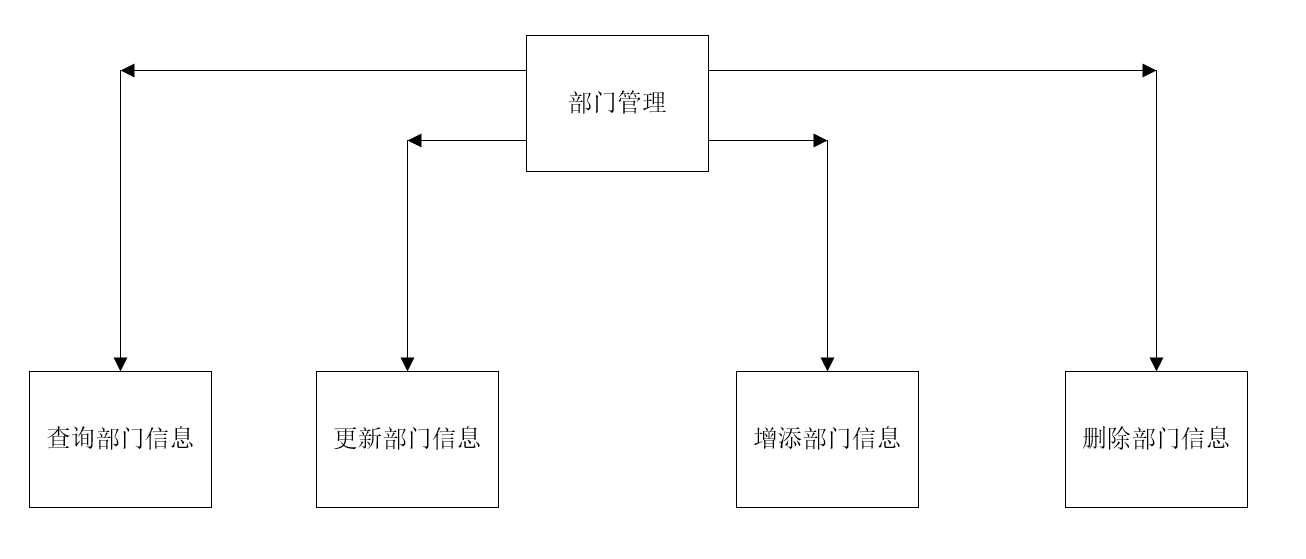
身份认证和登录模块涉及公司员工的包括id、email、password、roleId等属性。User类定义了相关员工属性并且给与了员工各种属性的获取、设置方法。身份认证和登录模块涉及公司员工的注册/登录、注销用户、检查登录状态。这些操作方法在serive层中的接口AuthService中定义，然后由AuthServiceImpl类实现。在实现过程中需要调用dao层的UserMapper中的方法，。其中，公司员工的注册/登录为boolean属性的 login(String username, String password, HttpServletResponse response) throws Exception函数，检查登录状态为User checkAuth(HttpServletRequest request) throws Exception，注销用户为void logout(HttpServletRequest request,HttpServletResponse response)。

## 3、员工管理模块



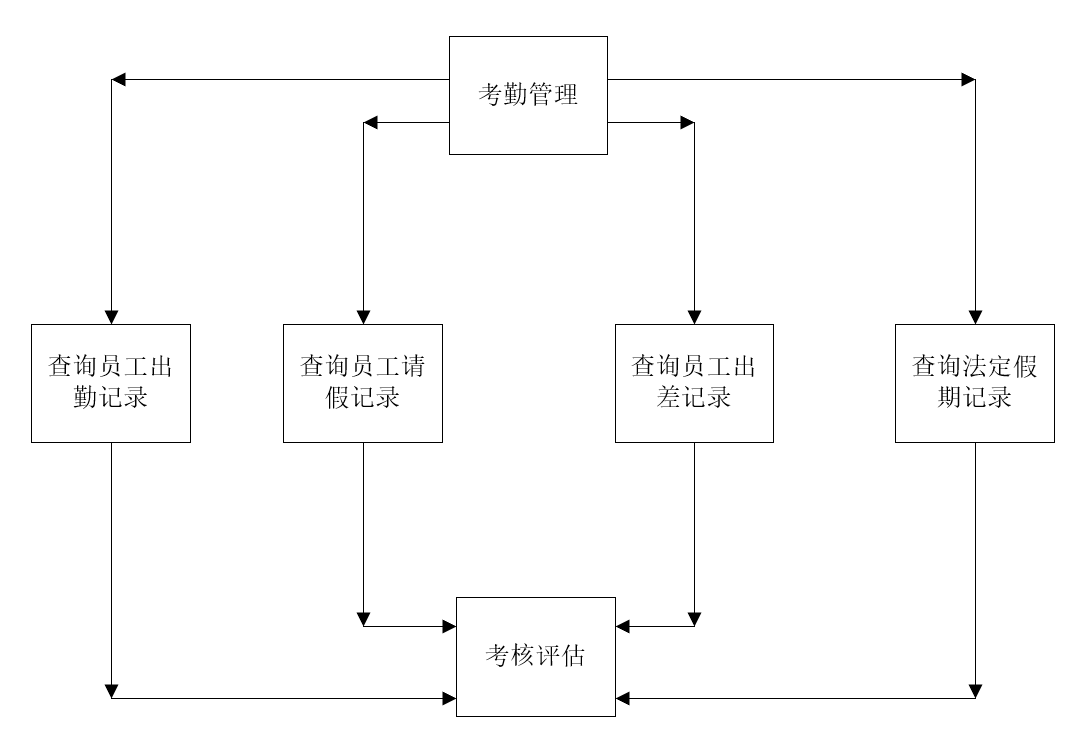
员工包括id、email、name、phone等属性。User类定义了员工的属性并且给与了员工各种属性的获取、设置方法。员工管理涉及员工的查询、增加、更新、删除、员工所属部门等。这些操作方法在serive层中的接口UserService中定义，然后由UserServiceImpl类实现。在实现过程中需要调用dao层的UserMapper中的方法，而UserMapper中的方法又涉及到User类中对部门属性的定义。其中，员工的查询和删除需要员工类中的id，而员工的增加与更新直接以员工对象为参数。其它则需要User类中与员工对应的角色ID（roleId）或部门ID（departmentId）。

## 4、部门管理模块



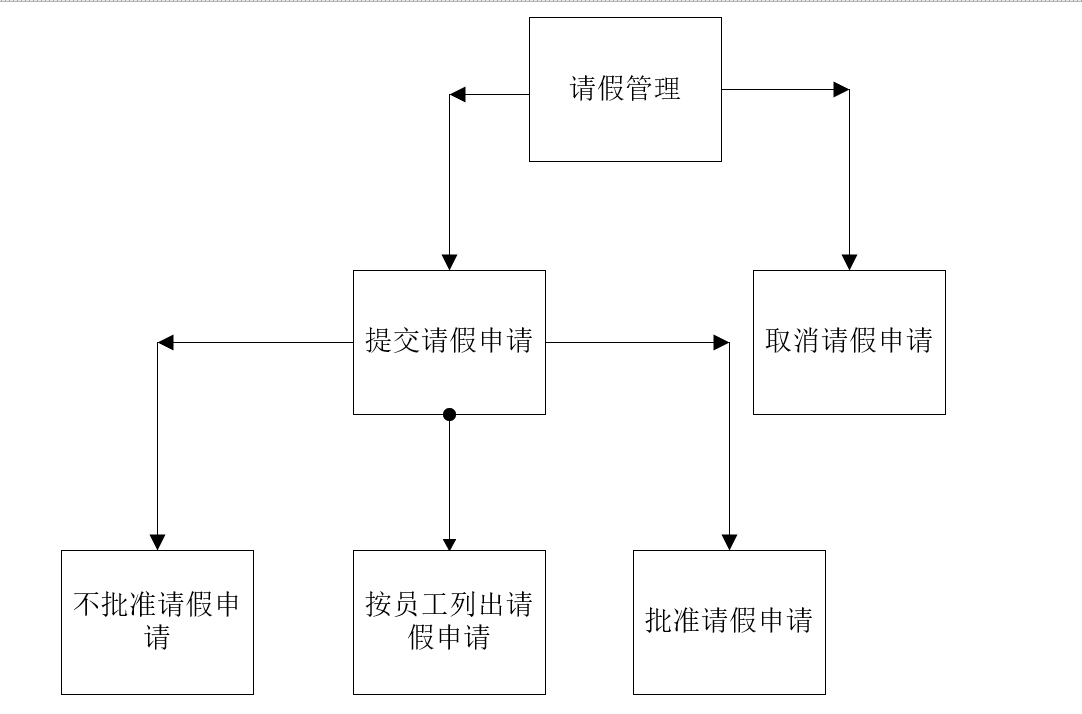
从部门类Department类中我们可以看出，部门（department）有id,name等多种属性信息，Department类也给与了这些属性信息的获取或更新方法。部门管理涉及部门的查询、增加、更新、删除等。这些操作方法在serive层中的接口DepartmentSerive中定义，然后由DepartmentServiceImpl类实现。在实现过程中需要调用dao层的DepartmentMapper中的方法，而DepartmentMapper中的方法又涉及到Department类中对部门属性的定义。其中，部门的查询和删除需要部门的id，而部门的增加与更新直接以部门对象为参数。

## 5、考勤管理模块

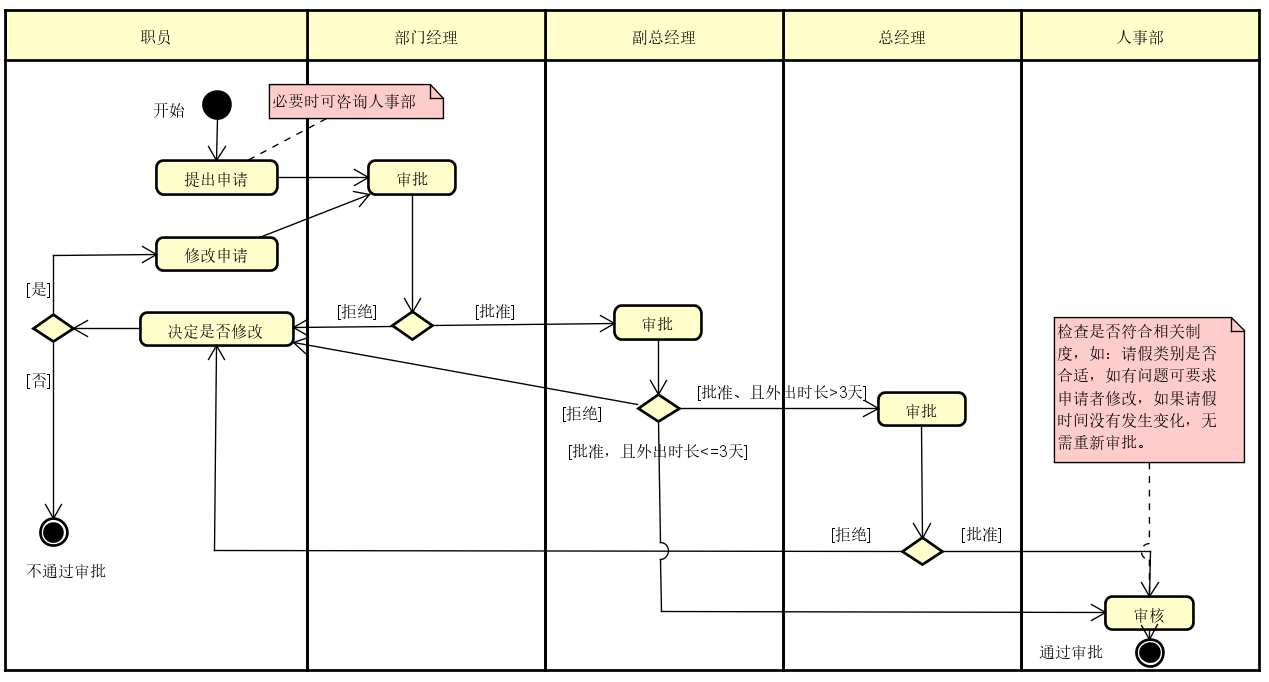


考勤管理包括本次考勤id、startCheck、endCheck、userId、type等属性，这些属性都包含在WorkCheck类中，其中Integer类型的type以0为假期，1为工作日，2为出差，3为请假来区分员工状态。除此之外WorkCheck类还定义并实现了与考勤管理相关的属性的获取或设置方法。考勤管理涉及到员工的工作时间、出差时间、请假时间、法定假期时间。考勤管理需查询员工所有出勤记录、请假记录、出差记录、假期记录然后考核评估。考勤管理相关操作在在serive层中的接口WorkService中定义，涉及boolean属性的 insertWorkCheck(WorkCheck workCheck)、boolean属性的updateEndCheck(long userId)、List<WorkCheck>属性的listWorkChecksByUserId(long userId) 和 listWorkChecks() 等函数方法，然后由WorkServiceImpl类实现。在实现过程中需要调用dao层的WorkCheckMapper中的方法，而WorkCheckMapper中的方法又涉及到WorkCheck类中对考勤管理属性的定义。

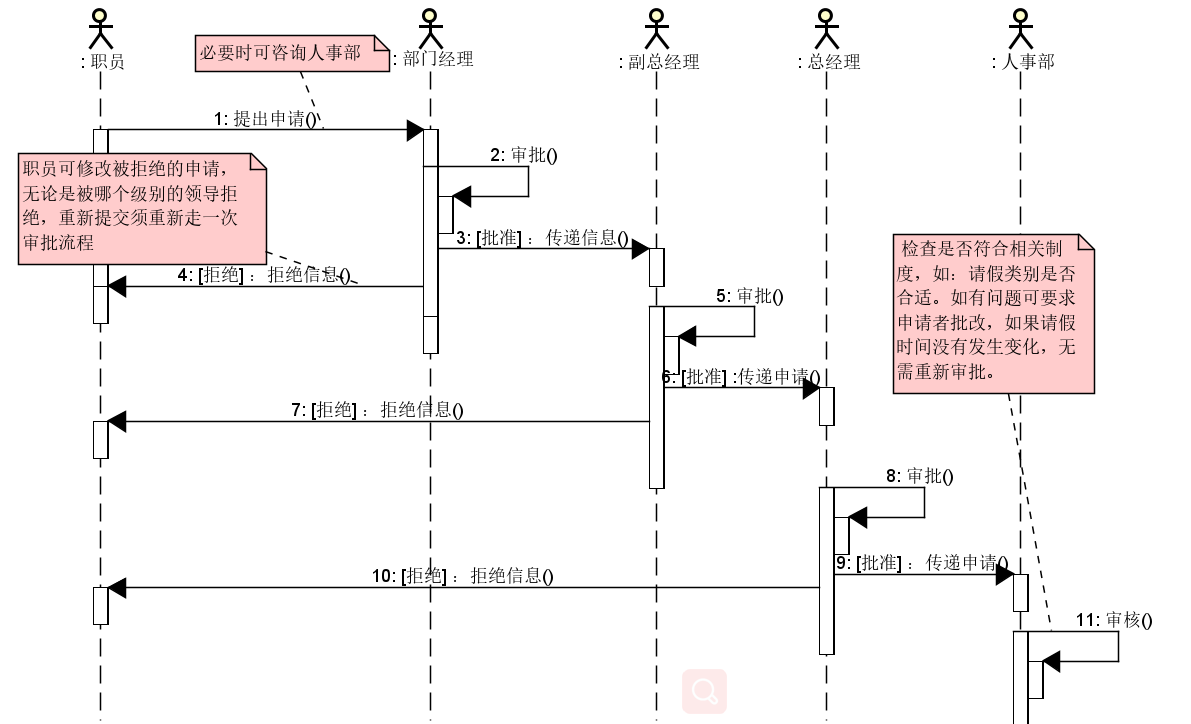
## 6、请假管理模块



活动图：

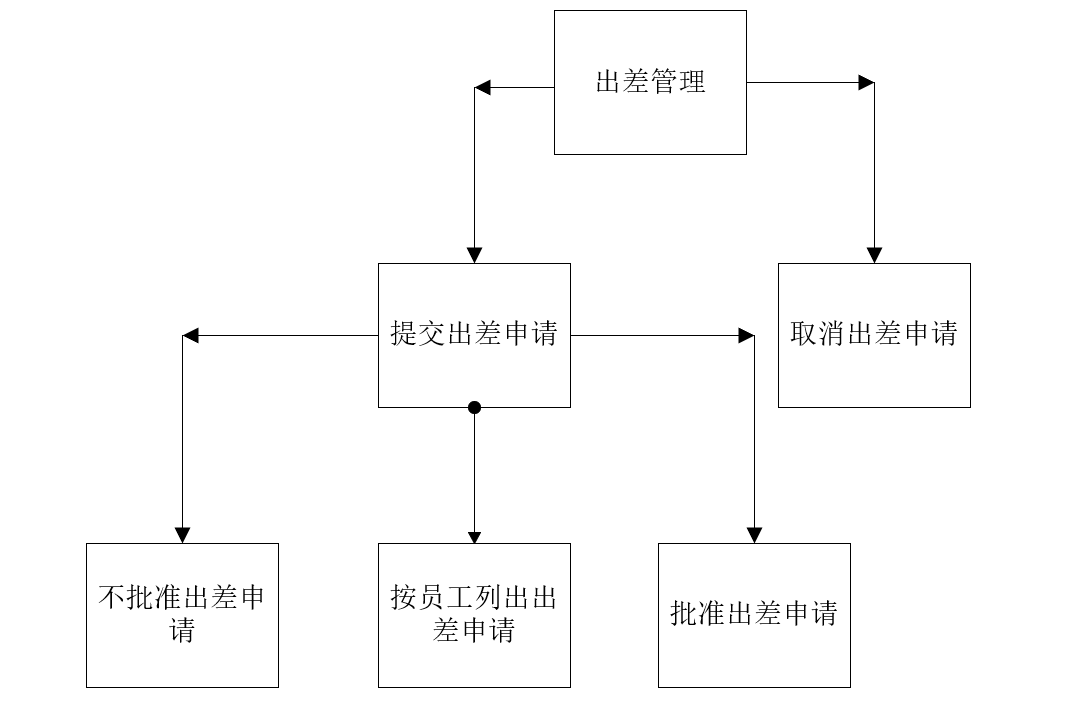


顺序图：

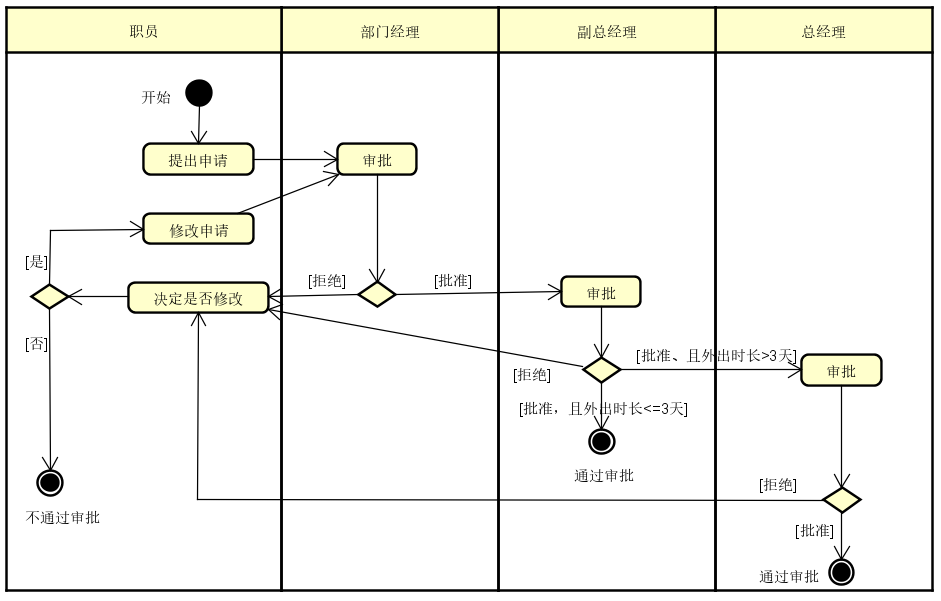


员工请假包括id、reason、start、end等属性。Schedule类除了上述属性还定义了type属性来判断本次活动是请假还是出差，status属性来表示提交申请、不批准申请、批准申请。并且给与各种属性的获取、设置方法。员工请假管理涉及员工的提交请假申请、取消请假申请、批准请假申请、不批准请假申请、按员工列出请假申请（本请假申请属于哪个员工）等。这些操作方法在serive层中的接口ScheduleService中定义，然后由ScheduleServiceImpl类实现。在实现过程中需要调用dao层的ScheduleMapper中的方法，而ScheduleMapper中的方法又涉及到Schedule类中对员工请假属性的定义。其中，员工请假管理的取消请假申请、批准请假申请、不批准请假申请需要Schedule类中的id。而提交请假申请以Schedule对象为参数。按员工列出请假申请（本请假申请属于哪个员工）以Schedule类中的userId为参数。

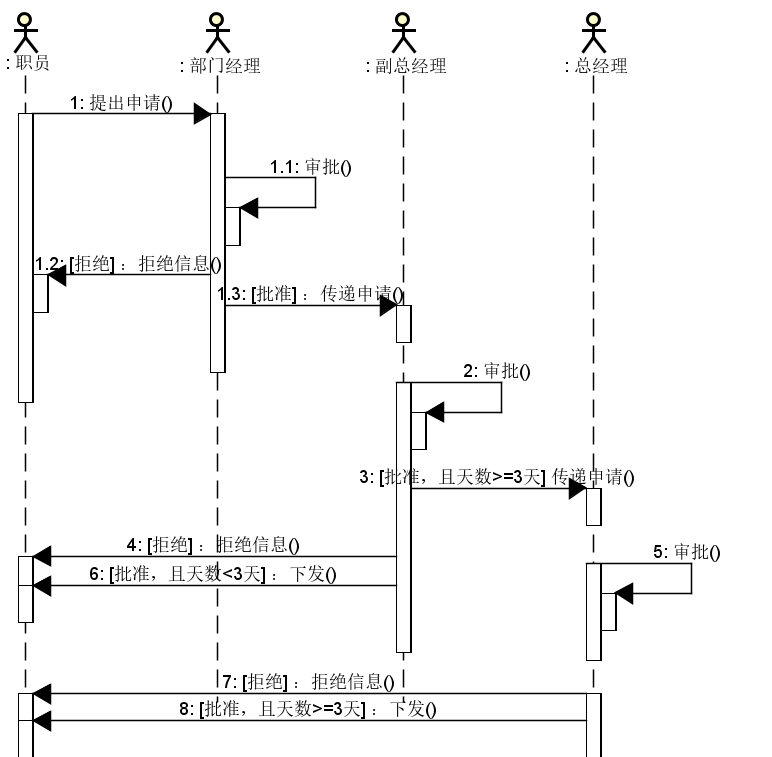
## 7、出差管理模块



活动图：

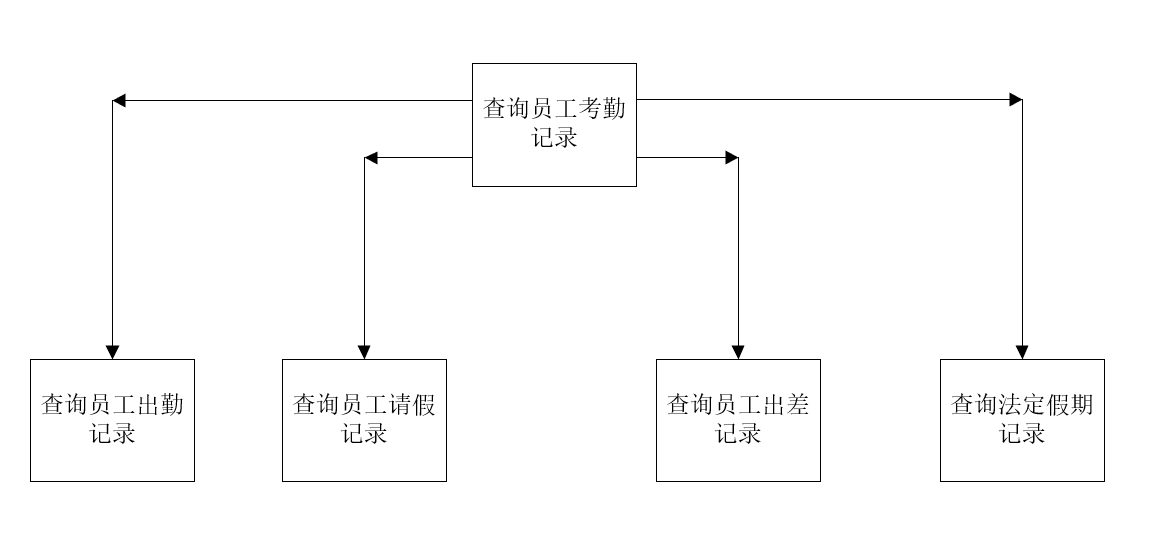


顺序图：



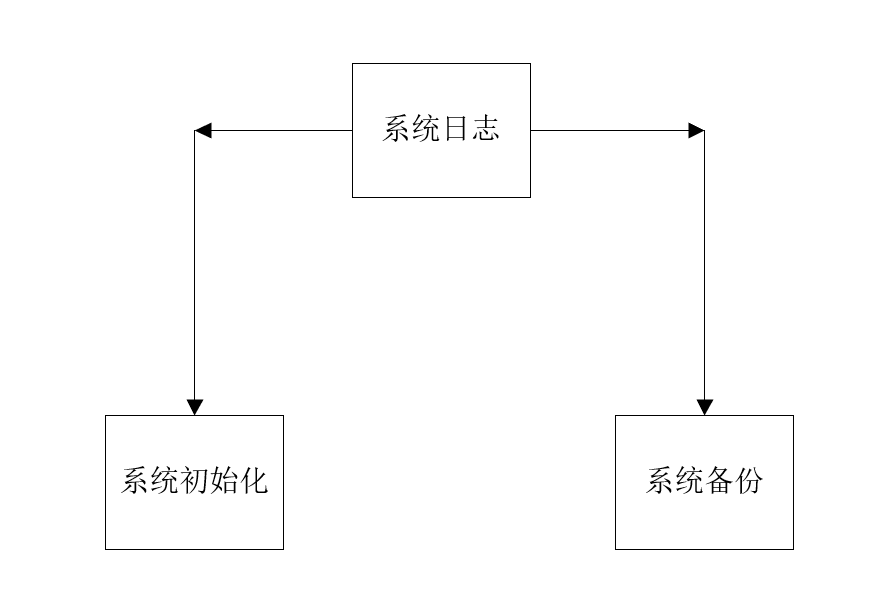
员工出差包括id、reason、start、end等属性。Schedule类除了上述属性还定义了type属性来判断本次活动是请假还是出差，status属性来表示提交申请、不批准申请、批准申请。并且给与各种属性的获取、设置方法。员工出差管理涉及员工的提交出差申请、取消出差申请、批准出差申请、不批准出差申请、按员工列出出差申请（本出差申请属于哪个员工）等。这些操作方法在serive层中的接口ScheduleService中定义，然后由ScheduleServiceImpl类实现。在实现过程中需要调用dao层的ScheduleMapper中的方法，而ScheduleMapper中的方法又涉及到Schedule类中对员工出差属性的定义。其中，员工出差管理的取消出差申请、批准出差申请、不批准出差申请需要Schedule类中的id。而提交出差申请以Schedule对象为参数。按员工列出出差申请（本出差申请属于哪个员工）以Schedule类中的userId为参数。

## 8、数据统计模块



查询员工的系统数据涉及到员工的工作时间、出差时间、请假时间、法定假期时间。查询员工所有出勤记录、请假记录、出差记录、假期记录然后考核评估相关操作在在serive层中的接口WorkService中定义，然后由WorkServiceImpl类实现。在实现过程中需要调用dao层的WorkCheckMapper、scheduleMapper、holidayMapper、configMapper。涉及到WorkCheck类、Config类、Schedule类、Holiday类中各种相关属性。

## 9、系统日志模块

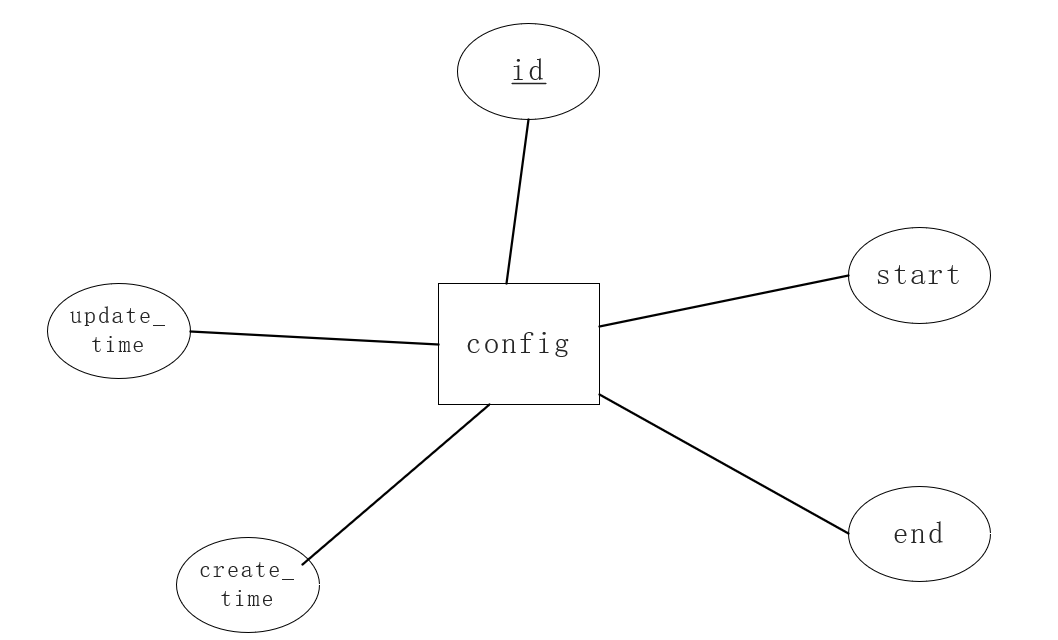


该模块包含的系统初始化，备份在serive层中的接口SystemService中定义，void initialization() 表示系统初始化,void backup() 表示系统备份，然后由SystemServiceImpl类实现。在实现过程中需要调用dao层的SystemMapper中的方法：void属性的 initialization()，deleteDepartment()，deleteHoliday()，deleteSchedule()，deleteUser()，deleteWorkCheck()，backup()。

# 四、数据库的设计与实现

## 1、数据库config的设计

1.1E-R图



1.2数据项说明

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 类型 | 字段约束 | 字段类型 |
| id | int(11) | Not null | pk |
| start | time | Not null |  |
| end | time | DEFAULT NULL |  |
| create\_time | datetime | Not null |  |
| update\_time | datetime | Not null |  |

1.3建表SQL语句

CREATE TABLE `config` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`start` time NOT NULL,

`end` time DEFAULT NULL,

`create\_time` datetime NOT NULL,

`update\_time` datetime NOT NULL,

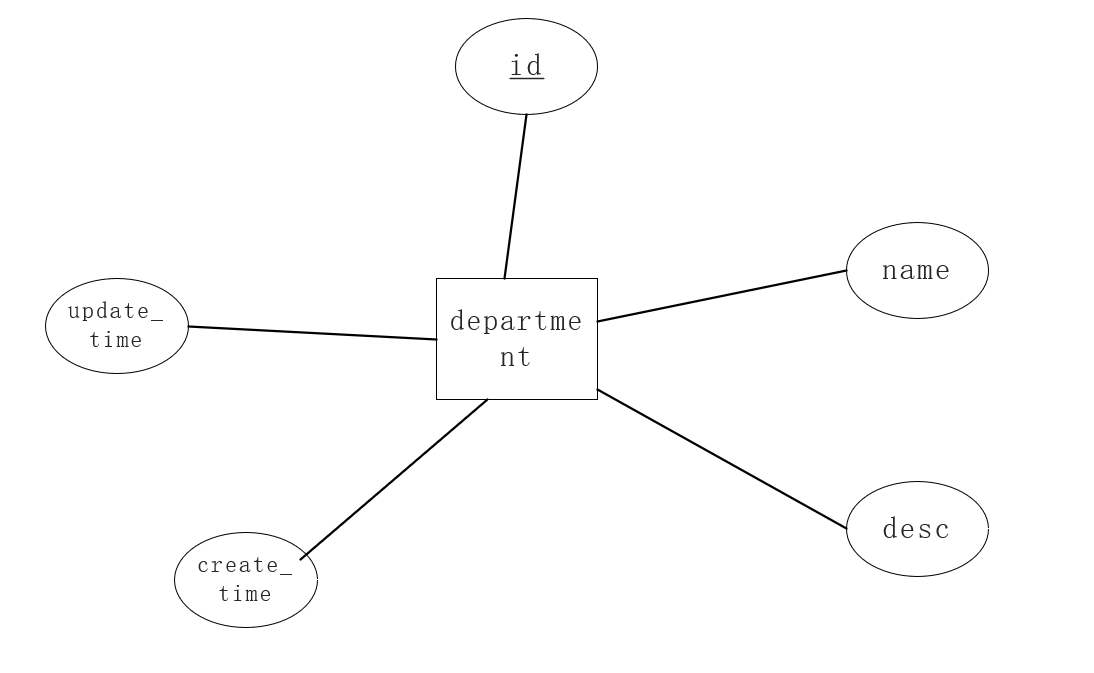
PRIMARY KEY (`id`),

UNIQUE KEY `config\_id\_uindex` (`id`)

ENGINE=InnoDB AUTO\_INCREMENT=8 DEFAULT CHARSET=utf8;

## 2、数据库department的设计

2.1E-R图



2.2数据项说明

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 类型 | 字段约束 | 字段类型 |
| id | int(11) | Not null | pk |
| name | varchar(50) | Not null |  |
| desc | varchar(200) | Not null |  |
| create\_time | datetime | Not null |  |
| update\_time | datetime | Not null |  |

2.3建表SQL语句

CREATE TABLE `department` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(50) NOT NULL,

`desc` varchar(200) NOT NULL,

`create\_time` datetime NOT NULL,

`update\_time` datetime NOT NULL,

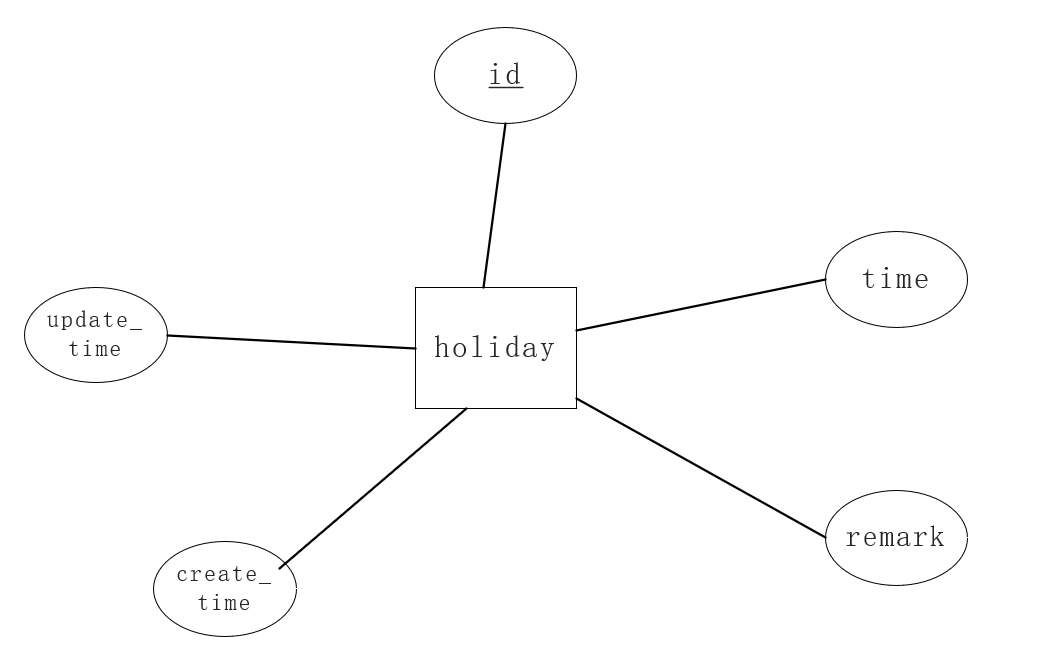
PRIMARY KEY (`id`),

UNIQUE KEY `department\_id\_uindex` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=9 DEFAULT CHARSET=utf8;

## 3、数据库holiday的设计

3.1E-R图



3.2数据项说明

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 类型 | 字段约束 | 字段类型 |
| id | int(11) | Not null | pk |
| time | datetime | Not null |  |
| remark | varchar(50) | Not null |  |
| create\_time | datetime | Not null |  |
| update\_time | datetime | Not null |  |

3.3建表SQL语句

CREATE TABLE `holiday` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`time` datetime NOT NULL,

`remark` varchar(50) NOT NULL,

`create\_time` datetime NOT NULL,

`update\_time` datetime NOT NULL,

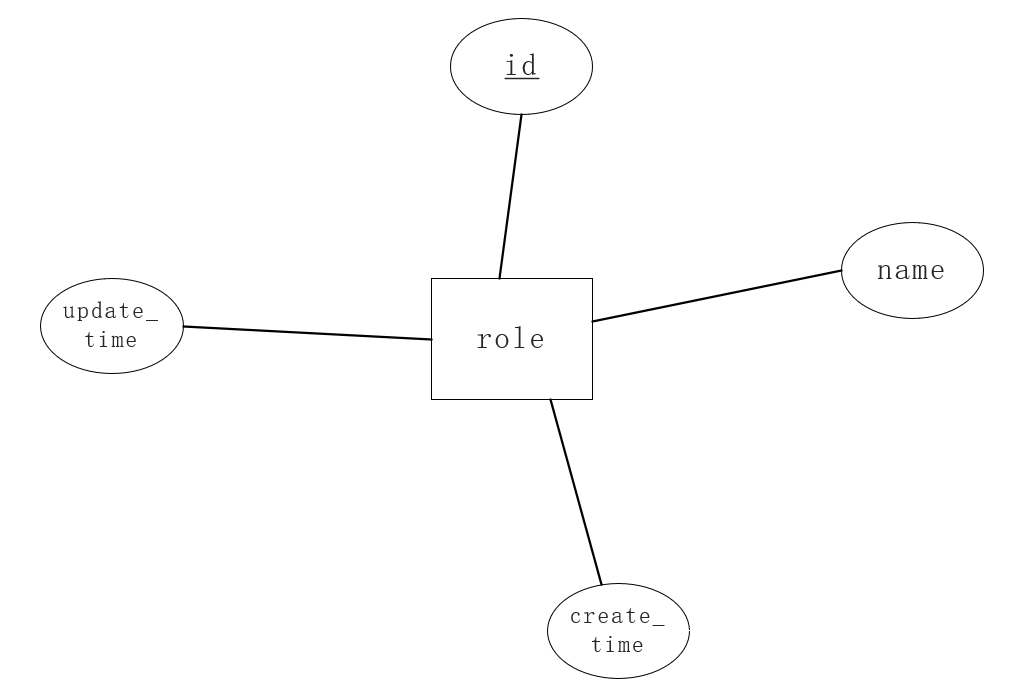
PRIMARY KEY (`id`),

UNIQUE KEY `holiday\_id\_uindex` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8;

## 4、数据库role的设计

4.1E-R图



4.2数据项说明

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 类型 | 字段约束 | 字段类型 |
| id | int(11) | Not null | pk |
| name | varchar(50) | Not null |  |
| create\_time | datetime | Not null |  |
| update\_time | datetime | Not null |  |

4.3建表SQL语句

CREATE TABLE `role` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(50) NOT NULL,

`create\_time` datetime NOT NULL,

`update\_time` datetime NOT NULL,

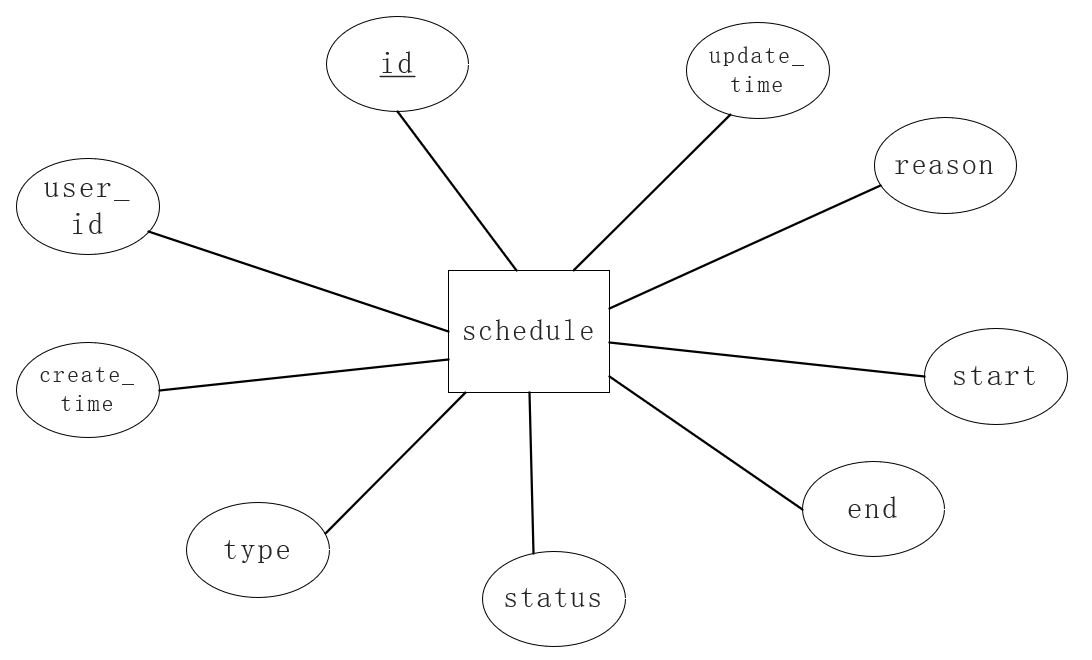
PRIMARY KEY (`id`),

UNIQUE KEY `role\_id\_uindex` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8;

## 5、数据库schedule的设计

5.1E-R图



5.2数据项说明

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 类型 | 字段约束 | 字段类型 |
| id | int(11) | Not null | pk |
| reason | varchar(200) | Not null |  |
| start | datetime | Not null |  |
| end | datetime | Not null |  |
| status | int(11) | Not null |  |
| type | int(11) | Not null |  |
| create\_time | datetime | Not null |  |
| update\_time | datetime | Not null |  |
| user\_id | int(11) | Not null |  |

5.3建表SQL语句

CREATE TABLE `schedule` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`reason` varchar(200) NOT NULL,

`start` datetime NOT NULL,

`end` datetime NOT NULL,

`status` int(11) NOT NULL,

`type` int(11) NOT NULL,

`create\_time` datetime NOT NULL,

`update\_time` datetime NOT NULL,

`user\_id` int(11) NOT NULL,

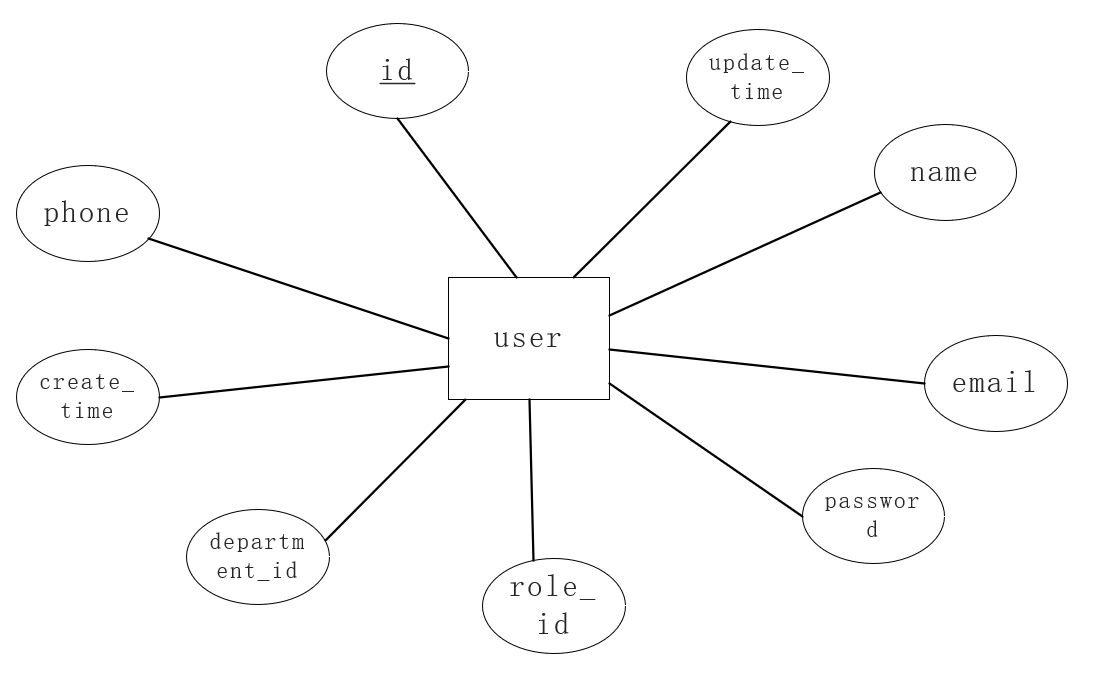
PRIMARY KEY (`id`),

UNIQUE KEY `schedule\_id\_uindex` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=utf8;

## 6、数据库user的设计

6.1E-R图



6.2数据项说明

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 类型 | 字段约束 | 字段类型 |
| id | int(11) | Not null | pk |
| name | varchar(50) | Not null |  |
| email | varchar(255) | Not null |  |
| password | varchar(255) | Not null |  |
| role\_id | int(11) | Not null |  |
| department\_id | int(11) | Not null |  |
| update\_time | datetime | Not null |  |
| create\_time | datetime | Not null |  |
| phone | bigint(20) | Not null |  |

6.3建表SQL语句

CREATE TABLE `user` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(50) NOT NULL,

`email` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

`role\_id` int(11) NOT NULL,

`department\_id` int(11) NOT NULL,

`update\_time` datetime NOT NULL,

`create\_time` datetime NOT NULL,

`phone` bigint(20) NOT NULL,

PRIMARY KEY (`id`),

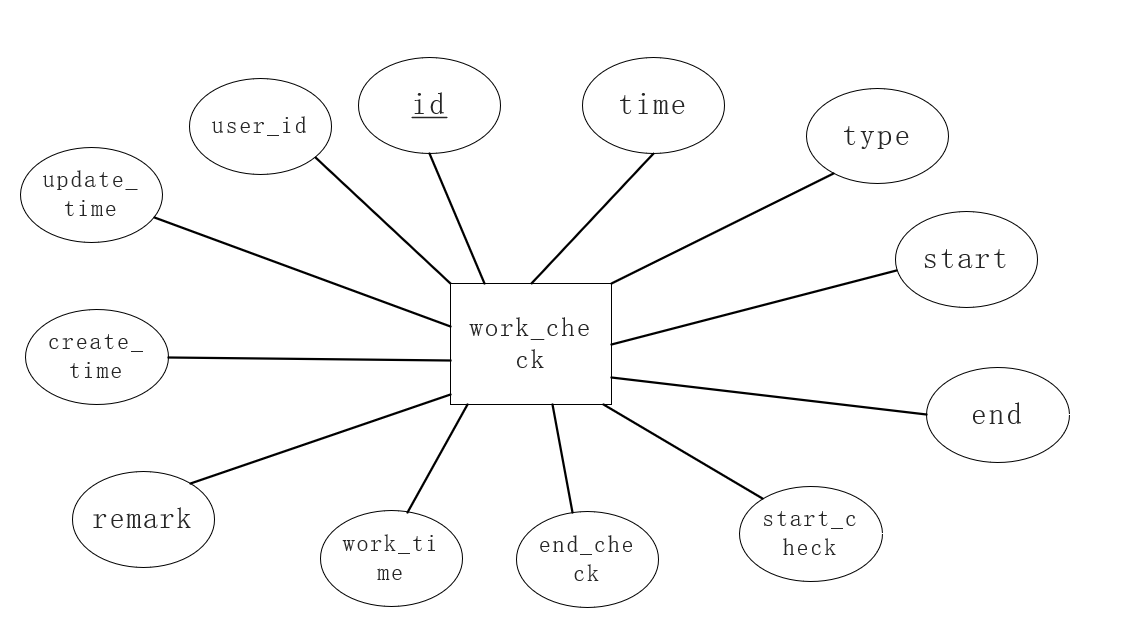
UNIQUE KEY `user\_id\_uindex` (`id`),

UNIQUE KEY `user\_email\_uindex` (`email`)

) ENGINE=InnoDB AUTO\_INCREMENT=10 DEFAULT CHARSET=utf8;

## 7、数据库work\_check的设计

7.1E-R图



7.2数据项说明

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 类型 | 字段约束 | 字段类型 |
| id | int(11) | Not null | pk |
| time | date | Not null |  |
| type | int(11) | Not null |  |
| start | time | Not null |  |
| end | time | DEFAULT NULL |  |
| start\_check | time | DEFAULT NULL |  |
| end\_check | time | DEFAULT NULL |  |
| work\_time | decimal(10,0) | DEFAULT NULL |  |
| remark | varchar(50) | DEFAULT NULL |  |
| create\_time | datetime | Not null |  |
| update\_time | datetime | Not null |  |
| user\_id | int(11) | Not null |  |

7.3建表SQL语句

CREATE TABLE `work\_check` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`time` date NOT NULL,

`type` int(11) NOT NULL,

`start` time NOT NULL,

`end` time DEFAULT NULL,

`start\_check` time DEFAULT NULL,

`end\_check` time DEFAULT NULL,

`work\_time` decimal(10,0) DEFAULT NULL,

`remark` varchar(50) DEFAULT NULL,

`create\_time` datetime NOT NULL,

`update\_time` datetime NOT NULL,

`user\_id` int(11) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `work\_check\_id\_uindex` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=13 DEFAULT CHARSET=utf8;

Dao层接口，供Service层调用以实现业务逻辑

ConfigMapper 实现对数据表Config的操作

DepartmentMapper 实现对数据表Department的操作

HolidayMapper 实现对数据表Holiday的操作

RoleMapper 实现对数据表Role的操作

ScheduleMapper 实现对数据表Schedule的操作

SystemMapper 实现对系统管理的操作

UserMapper 实现对数据表User的操作

WorkCheckMapper 实现对数据表WorkCheck的操作

# 五、小组分工

**平云：**

1. 前端代码

包括mapper文件夹下的configMapper.xml,departmentMapper.xml,holidayMapper.xml,roleMapper.xml, userMapper.xml,workCheckMapper.xml

Static目录下的css代码，.js代码部分的设计

Templates目录下的department.html,department\_add.html,employee,html,employee\_add.html,employ\_undate.html ,leave.html ,

2）后端代码

包括service层的设计AuthServiceImpl.java,DepartmentServiceImpl.java,RoleServiceImpl.java,ScheduleServiceImpl.java,SystemServiceImpl.java,UserServiceImpl.java,WorkServiceImpl.java

Service层的接口

AuthService.java，DepartmentService.java，RoleService.java，ScheduleService.java，SystemServic.java，UserService.java ，WorkService.java

对于controller层的设计的审核

1. 数据库的设计

Department，Schedule的设计

1. 文档编写

总体架构的设计及架构图

Controller层的设计，包括员工管理模块，部门管理模块，日志管理模块的编写

数据库的Department，Schedule的文字描述，表格设计和E-R图的设计

Service层的总UML图的设计

Controller层文档审核，总文档汇总审核

**李朋恩：**

1. 前端代码

包括mapper文件夹下的scheduleMapper.xml,systemMapper.xml

Templates目录下的stat.html,system.html，work.html

2）后端代码

包括dao层的接口设计CongigMapper.java,DepartmentMapper.java,HolidayMapper.java,RoleMapper.java,ScheduleMapper.java,SystemMapper.java,UserMapper.java,WorkCheckMapper.java

Controller层的设计DepartmentController.java，HolidayController.java，ScheduleController.java，UserController.java

对于日期的设定,即DateUtil.java

对于pojo文件下的Bo，Vo的完善和校对

3）数据库的设计

Work\_check，user，Holiday的设计

1. 文档编写

数据库的Work\_check，user，Holiday的文字描述，表格设计和E-R图的设计

Service层员工管理模块，部门管理模块，外出申请模块，请假申请模块的编写

Service层文档设计编写审核

**库尔班：**

1. 前端代码：

包括mapper文件夹下的business\_trip.html

Templates目录下的login.html

1. 后端代码：

包括controller层的AuthController.java，StatController.java，SystemController.java，IndexController.java

包括日志的添加，即LogAspect.java

对于用户密码加密的完善，即EncryUtil.java

1. 数据库设计

Role的设计

1. 文档编写

Service层的登陆模块，考勤管理模块，日志管理模块，编写

数据库的Role的文字描述，表格设计和E-R图的设计

**麦尼外尔：**

1）前端代码：

包括mapper文件夹下的index.html

Static目录下的系统所需图片的查找

1. 后端代码：

包括Vo,Bo的设计，即pojo目录下的Congig.java，Department.java，DepartmentVo.java，Holiday.java，JsonResponse.java，Role.java，Schedule.java，User.java，UserVo.java，WorkCheck.java

对于json格式的统一，即JsonUtil.java

1. 数据库设计

Config设计

1. 文档编写

Controller层包括登陆模块，考勤管理模块，外出申请模块，请假申请模块的编写

数据库的Config的文字描述，表格设计和E-R图的设计