

2019年软件工程课程设计报告

**课设题目：**  员工考勤系统

|  |  |
| --- | --- |
| 姓名 | 学号 |
| 孙洪武（队长） |  |
| 丁振远 | 41624192 |
| 董志伟 |  |
| 董润时 |  |
| 张铮睿 |  |
| 丁梓硕 |  |

提交时间：2019年6月20日

[一、总体说明 1](#_Toc11761193)

[1.1 开发团队 1](#_Toc11761194)

[1.2 开发环境和工具 1](#_Toc11761195)

[1.2.1 后端 1](#_Toc11761196)

[1.2.2 Web端 2](#_Toc11761197)

[1.2.3 移动端 2](#_Toc11761198)

[二、软件需求 2](#_Toc11761199)

[2.1 需求陈述 2](#_Toc11761200)

[2.2 可行性分析 4](#_Toc11761201)

[2.3 需求分析 5](#_Toc11761202)

[2.3.1 功能性需求 5](#_Toc11761203)

[2.3.2 非功能性需求 10](#_Toc11761204)

[三、系统设计 12](#_Toc11761205)

[3.1 总体设计（或体系结构设计） 12](#_Toc11761206)

[3.1.1 面向对象分析 12](#_Toc11761207)

[3.1.2 整体结构设计 14](#_Toc11761208)

[3.2 详细设计 16](#_Toc11761209)

[3.2.1数据库结构设计 16](#_Toc11761210)

[3.2.2数据接口设计 17](#_Toc11761211)

[3.2.3人机接口设计 17](#_Toc11761212)

[3.2.4数据驱动设计 17](#_Toc11761213)

[四、系统实现 17](#_Toc11761214)

[4.1 后端 17](#_Toc11761215)

[4.1.1 代码结构 17](#_Toc11761216)

[4.1.2 存放地点 20](#_Toc11761217)

[4.1.3 关键代码逻辑 20](#_Toc11761218)

[4.2 Web端 23](#_Toc11761219)

[4.3 移动端 23](#_Toc11761220)

[五、系统测试 23](#_Toc11761221)

[5.1 测试方案 23](#_Toc11761222)

[5.1.1 后端 23](#_Toc11761223)

[5.1.2 Web端 23](#_Toc11761224)

[5.1.3 移动端 23](#_Toc11761225)

[5.2 测试结果 24](#_Toc11761226)

[5.1.1 后端 24](#_Toc11761227)

[5.1.2 Web端 24](#_Toc11761228)

[5.1.3 移动端 24](#_Toc11761229)

[5.3 分析 24](#_Toc11761230)

[六、部署和运行 24](#_Toc11761231)

[参考文献 25](#_Toc11761232)

# 一、总体说明

## 1.1 开发团队

|  |  |
| --- | --- |
| 孙洪武 | 后端设计开发，后端测试，全队整体设计开发指导 |
| 丁振远 | 后端设计开发，后端测试，文档设计整理 |
| 董志伟 | Web前端设计开发，前端测试 |
| 董润时 | Web前端设计开发，前端测试 |
| 张铮睿 | 移动端设计开发，移动端测试，原型设计 |
| 丁梓硕 | 移动端设计开发，移动端测试，文献翻译 |

## 1.2 开发环境和工具

### 1.2.1 后端

|  |  |
| --- | --- |
| 语言 | Go语言 |
| 数据库 | Mysql数据库 |
| 依赖包 | github.com/gin-gonic/gin  github.com/Unknwon/goconfig  github.com/jinzhu/gorm  github.com/go-sql-driver/mysql  github.com/dgrijalva/jwt-go  golang.org/x/crypto/bcrypt  github.com/skip2/go-qrcode  github.com/tencentcloud/tencentcloud-sdk-go  github.com/gin-contrib/cors |
| 协同工具 | GitHub |

### 1.2.2 Web端

### 1.2.3 移动端

# 二、软件需求

## 2.1 需求陈述

基本需求陈述如下：

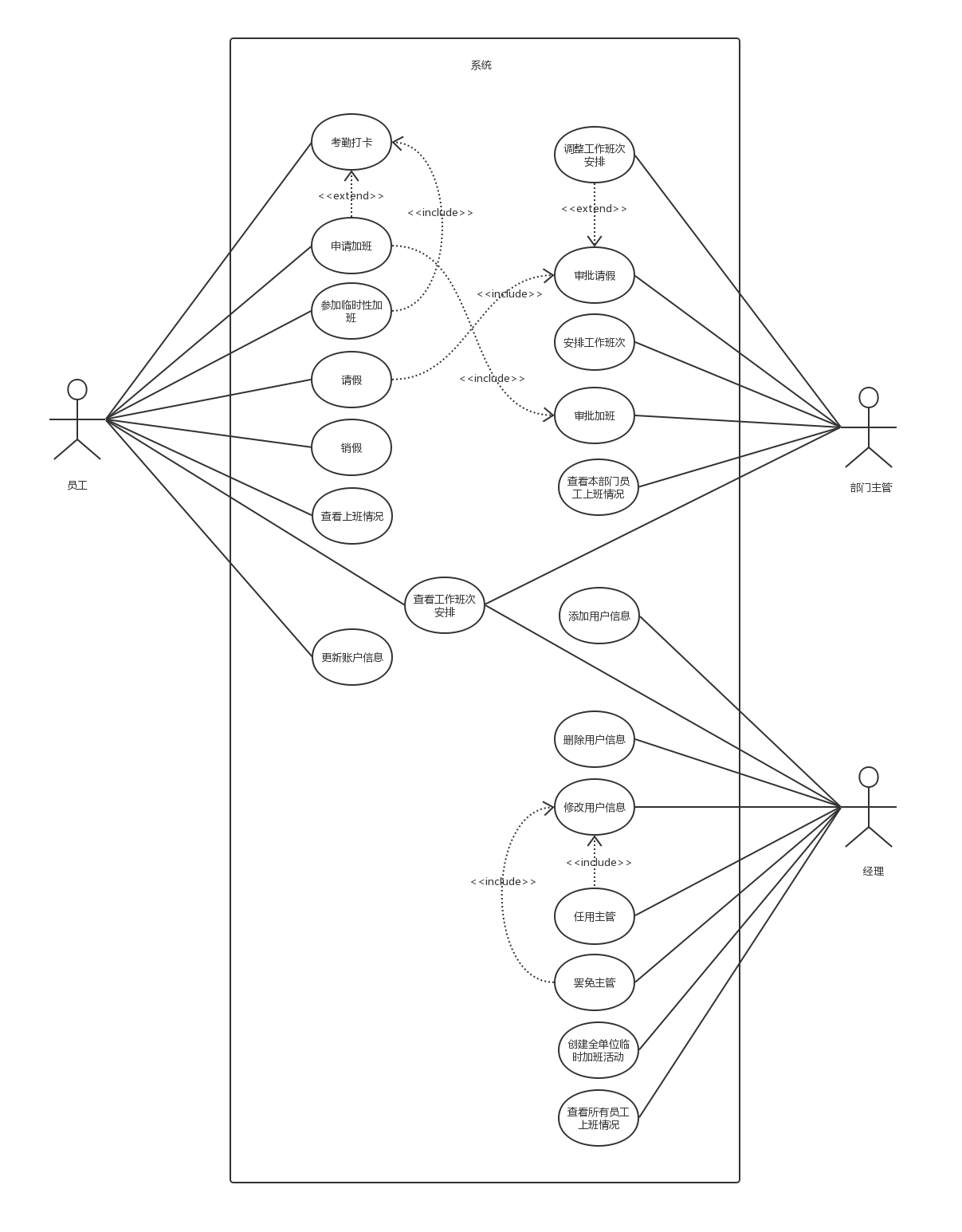
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 说明 | 必要性 | 最低要求 | 最高要求 |
| 1 | 经理能够向系统中添加、删除、管理用户信息，包括工号、所属部门、初始密码、是否为部门主管，一个部门只能有一个主管，经理能够任免主管。 | TRUE | 经理能通过系统依次添加或管理员工信息。 | 经理能够通过文件（如excel）导入和更新员工信息。 |
| 2 | 部门主管能够为本部门的员工安排工作班次（通常以月度为单位，但不限于月度），某员工的一个工作班次是每天的上下班时间。注意，不同员工的工作班次安排可能不同，一名员工每天的工作班次安排也可能不同。 | TRUE |  | 为了简化主管的操作，可以允许主管（1）批量设置部门员工的工作班次安排、（2）复制/粘贴工作班次安排、（3）根据常见的工作时间模式设置员工的工作班次（如：“朝九晚五”、“三班倒”等）、（4）存储和导入以往的工作班次安排。 |
| 3 | 系统能够以月为单位展示（1）某一部门所有员工的工作班次安排、（2）某一员工的工作班次安排。 | TRUE |  |  |
| 4 | 部门主管可以随时通过系统调整工作班次安排。 | TRUE |  | 调整工作班次安排可以是临时调整或永久调整。 |
| 5 | 员工可以通过系统查看自己的工作班次安排。 | TRUE |  | 根据员工的设置，系统能够在员工上班前自动提醒员工的上下班时间。 |
| 6 | 员工可以通过系统进行请假和销假，系统能够自动提醒主管进行审批。如果员工的请假申请被批准，系统能够提醒主管调整该员工在请假期间的工作班次安排。请假申请包括请假的起始日期和终止日期、请假理由和类型（事假/病假）等。 | TRUE |  |  |
| 7 | 员工能够通过系统进行考勤记录，即系统能够记录员工实际上下班的时间（打卡）。 | TRUE | 系统可以通过如二维码的方式为员工进行打卡，请注意系统必须能够设法防止员工的作弊行为。 | 系统能够以人脸识别的方式实现打卡功能。 |
| 8 | 如果员工加班，如下班时间超过计划时间一定阈值，系统提醒员工是否要申报加班。如果员工要申请加班，可以通过系统提交加班理由，等待部门经理审批。 | TRUE |  |  |
| 9 | 经理能够创建全单位的临时性加班活动，独立于部门经理的工作班次安排计划。系统能够记录员工是否参与了临时性加班（需打卡）。 | TRUE | 系统可以通过如二维码的方式为员工进行打卡，请注意系统必须能够设法防止员工的作弊行为。 | 系统能够以人脸识别的方式实现打卡功能。 |
| 10 | 系统能够以月为单位展示每名员工的上班情况，部门主管能够查看本部门的员工情况，经理能够查看所有员工的情况。 | TRUE |  | 系统还能够统计员工的正常工作时间、平时加班时间、节假日加班时间，以方便进行薪酬统计。 |
| 11 | 员工能够通过系统更新自己的账户信息 | TRUE |  |  |

## 2.2 可行性分析

## 2.3 需求分析

### 2.3.1 功能性需求

#### 2.3.2.1 用况图



#### 2.3.2.2 用况说明

|  |
| --- |
| 用况：考勤打卡  参与者：员工 |
| 1. 【员工】：选择用二维码模式或者人脸识别模式 2. 【系统】：判断员工的选择 3. 【系统】：如果员工选用二维码模式，则调用程序生成二维码进行展示 4. 【员工】：扫描二维码，并允许定位 5. 【系统】：判断员工定位信息 6. 【系统】：如果员工定位在附近，则打卡成功，记录时间 7. 【系统】：如果员工定位不在附近，则打卡失败，提示员工 8. 【系统】：如果员工选用人脸识别模式，则打开摄像头对用户拍摄 9. 【员工】：正面面向摄像头 10. 【系统】：根据采集的面部信息判断员工身份 11. 【系统】：识别身份成功，打卡成功，记录时间 12. 【系统】：识别身份失败，打卡失败，提示员工 13. 【系统】：根据下班的打卡时间判断员工是否超过计划时间一定阈值 14. 【系统】：如果超过阈值，提醒员工是否申请加班，调用申请加班子功能 |

|  |
| --- |
| 用况：申请加班  参与者：员工 |
| 1. 【员工】：选择申请加班或者不申请，如果申请需要填写加班理由 2. 【系统】：如果员工申请加班，则将加班信息发送给部门经理审批 3. 【系统】：如果员工不申请加班，则恢复原状 4. 【系统】：如果员工申请加班，而未填写理由正确，提醒用户 |

|  |
| --- |
| 用况：参加临时性加班  参与者：员工 |
| 1. 【员工】：选择参加临时性加班 2. 【系统】：记录员工参与了临时性加班 3. 【员工】：下班打卡 4. 【系统】：获得员工下班打卡时间并记录判断 5. 【系统】：如果员工下班时间达到临时加班规定时长，记录员工参与临时加班成功 6. 【系统】：如果员工下班时间未达到时长，记录员工参与临时加班失败，提醒员工 |
| 用况：请假  参与者：员工 |
| 1. 【员工】：选择请假，填写请假申请并提交 2. 【系统】：将申请发送给员工对应的部门主管，等待对请假的审批 |

|  |
| --- |
| 用况：销假  参与者：员工 |
| 1. 【员工】：选择销假 2. 【系统】：将员工剩余的请假假期状态全部取消，提醒其相应的部门主管销假消息。 |

|  |
| --- |
| 用况：查看上班情况  参与者：员工 |
| 1. 【员工】：选择查看上班情况，选择月份 2. 【系统】：显示员工此月上班情况 |

|  |
| --- |
| 用况：查看工作安排  参与者：员工，部门主管，经理 |
| 1. 【员工】或【部门主管】或【经理】选择查看工作安排，选择部门和月份 2. 【系统】：显示该部门该月份的工作班次安排 |

|  |
| --- |
| 用况：更新账户信息  参与者：员工 |
| 1. 【员工】：选择更新账户信息 2. 【系统】：显示员工账户原来信息 3. 【员工】：修改账户信息，并提交修改后的信息 4. 【系统】：保存修改后的信息替代原信息 |

|  |
| --- |
| 用况：调整工作班次安排  参与者：部门主管 |
| 1. 【部门主管】：选择调整工作班次安排 2. 【系统】：显示调整界面 3. 【部门主管】：选择本部门员工，并重新设定工作班次，提交调整后的工作班次 4. 【系统】：保存调整后的工作班次 |

|  |
| --- |
| 用况：审批请假  参与者：部门主管 |
| 1. 【部门主管】：收到请假提示消息并查看 2. 【系统】：显示请假申请信息，申请人和时间 3. 【部门主管】：选择批准与否 4. 【系统】：如果部门主管批准，则提示该部门主管调整工作班次安排，调用调整工作安排的功能 5. 【系统】：如果部门主管拒绝，则提示员工申请失败 |

|  |
| --- |
| 用况：安排工作班次  参与者：部门主管 |
| 1. 【部门主管】：选择安排工作班次 2. 【系统】：显示选择界面 3. 【部门主管】：选择员工，设定工作班次，提交设定后的表单 4. 【系统】：判断是否有工作班次冲突 5. 【系统】：没有冲突则保存新设定的工作班次 6. 【系统】：有冲突，则提醒部门主管发生冲突，设定失败 |

|  |
| --- |
| 用况：审批加班  参与者：部门主管 |
| 1. 【部门主管】：收到加班请求并查看 2. 【系统】：显示加班申请人信息，上下班日期和时间 3. 【部门主管】：选择是否批准 4. 【系统】：若批准，则加班情况被记录，提醒员工加班成功 5. 【系统】：若否决，则提醒员工加班失败 |

|  |
| --- |
| 用况：查看本部门员工上班情况  参与者：部门主管 |
| 1. 【部门主管】：选择查看本部门员工上班情况，选择员工，选择月份 2. 【系统】：显示选定员工该学分的上班情况 |

|  |
| --- |
| 用况：添加用户信息  参与者：经理 |
| 1. 【经理】：选择添加用户信息，选择依次添加或者文件导入 2. 【系统】：如果依次添加，则显示空的用户信息表单供填写 3. 【经理】：输入用户信息到表单中，选择提交 4. 【系统】：将经理提交的用户信息保存，提示保存成功 5. 【系统】：如果选择文件导入，则提供文件导入的接口 6. 【经理】：输入文件 7. 【系统】：从文件中读用户信息，每个员工一行，进行自动创建存储，完成后提示成功 |

|  |
| --- |
| 用况：删除用户信息  参与者：经理 |
| 1. 【经理】：选择删除某用户的信息 2. 【系统】：提醒经理确定 3. 【经理】：确认删除用户 4. 【系统】：删除用户信息，并提示删除成功 |

|  |
| --- |
| 用况：修改用户信息  参与者：经理 |
| 1. 【经理】：选择修改某用户的信息 2. 【系统】：显示用户的信息表单 3. 【经理】：修改信息后提交 4. 【系统】：将收到的修改后的用户信息保存，并提示经理修改成功 |

|  |
| --- |
| 用况：任用主管  参与者：经理 |
| 1. 【经理】：选择对应的员工，并选择任用为某部门主管 2. 【系统】：修改员工的用户信息，更改职位为部门主管 3. 【系统】：修改完成后提示经理修改完成，提示被任用的用户和所有该部门员工关于新的部门主管任用信息 |

|  |
| --- |
| 用况：罢免主管  参与者：经理 |
| 1. 【经理】：选择对应的部门主管，选择罢免主管 2. 【系统】：修改部门主管的用户信息，更改职位为员工 3. 【系统】：修改完成后提示经理修改完成，提示被罢免的主管和所有该部门员工关于罢免部门主管的消息 |

|  |
| --- |
| 用况：创建全单位临时加班活动  参与者：经理 |
| 1. 【经理】：选择创建全单位临时加班活动 2. 【系统】：提供创建加班活动的表单 3. 【经理】：填写加班活动表单并提交 4. 【系统】：将经理提交的表单信息整理，并发布给全单位所有用户 |

|  |
| --- |
| 用况：查看所有员工上班情况  参与者：经理 |
| 1. 【经理】：选择查看某员工（任意），某一个月份的上班情况 2. 【系统】：显示该员工此月的上班情况 |

### 2.3.2 非功能性需求

#### 2.3.2.1 精度需求

（1）对于系统中的日期时间节点统一精确到秒，可精确记录用户打卡等时间即可。

（2）对于系统中的工时，数据精度精确到个位，向下取整进行记录和计算。

#### 2.3.2.2 时间特性需求

（1）支持多用户（企业级用户规模）并发使用，并且性能不受影响。

（2）前端和小程序端的响应时间在网络通畅的情况下按不应超过2秒。

（3）批量添加用户的响应时间在网络通畅的情况下不应超过10秒。

（4）后端重启过程中加载人脸库（企业级规模人脸库）过程用时控制在10分钟之内。

#### 2.3.2.3 灵活性需求

（1）数据库调整改变时，能够方便地进行数据库迁移。

（2）系统可以在多端使用：Web浏览器端，IOS应用，Android应用，微信小程序端均可使用。

（3）数据通信接口的可扩张性比较好，冗余少。

（4）绝大多功能（检验合法性、逻辑判断、逻辑操作等）均置于后端执行，减小前段对计算资源的需求量。

#### 2.3.2.4 安全性需求

（1）后端需要对所有的输入进行所有情况的合法性检验。

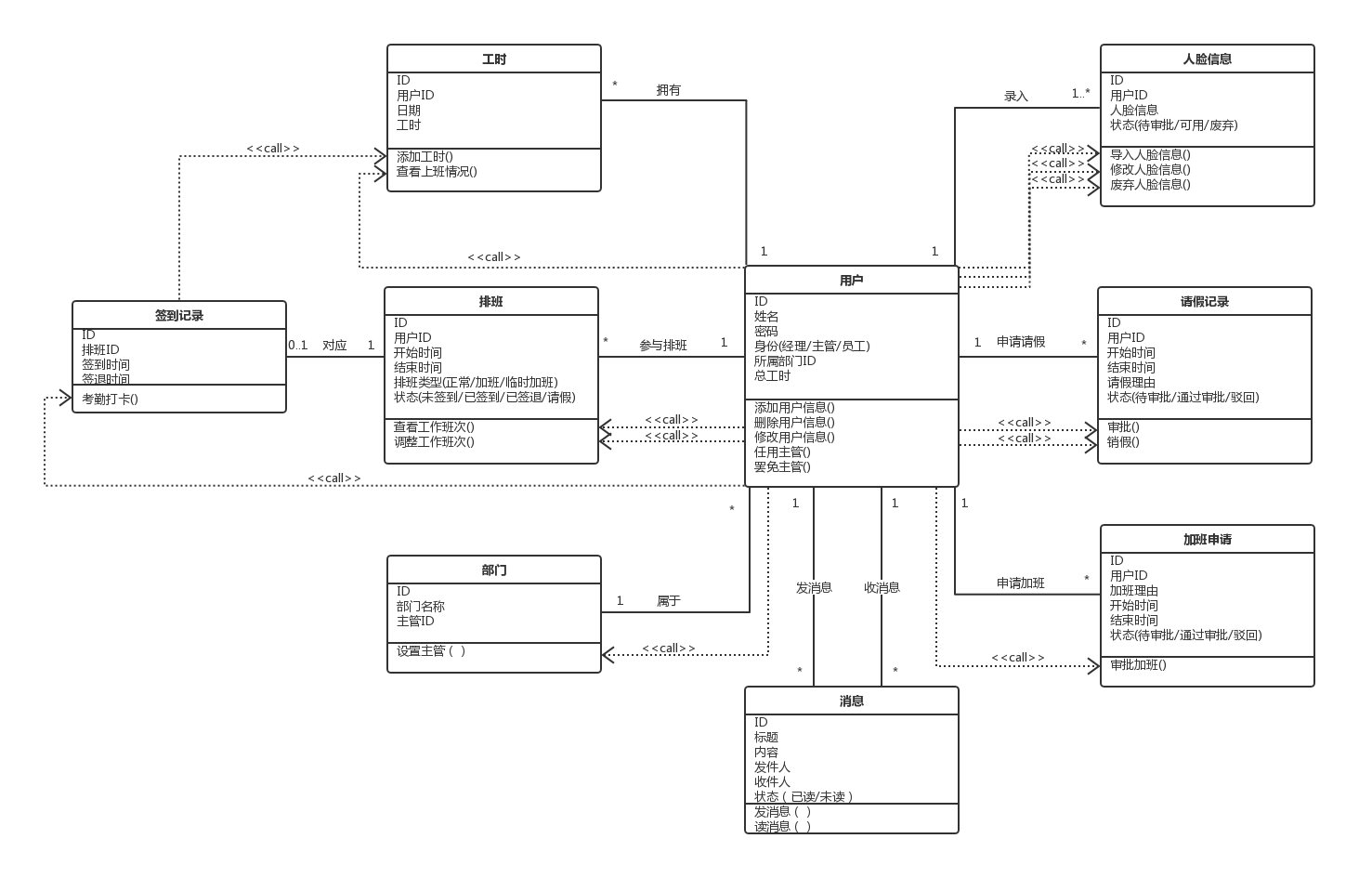
（2）部门便于检验的数据可以在前端也加入合法性检验。

# 三、系统设计

## 3.1 总体设计（或体系结构设计）

### 3.1.1 面向对象分析

#### 分析类图



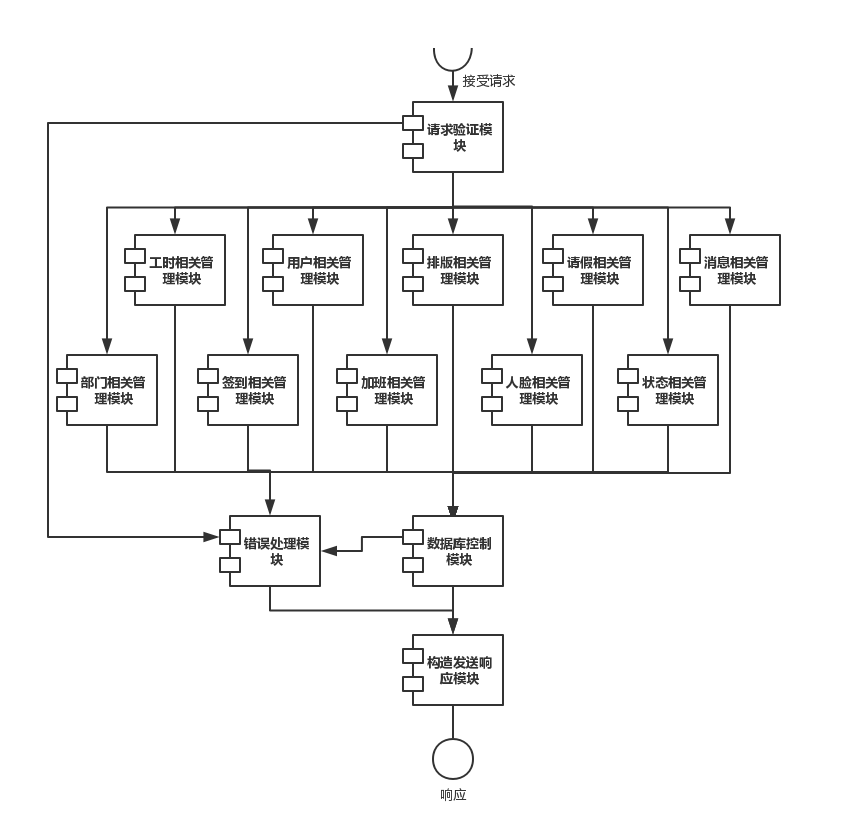
#### 3.1.1.2 类图说明

1. ID为每个对象的唯一标识，在用户处可以对应为职工号
2. 用户通过身份属性进行区分，分成员工，部门主管和经理。通过此字段的不同，确定不同的用户的不同权限，以及后面方法的不同执行方式。
3. 一个员工的一段完整的工作安排为一个排班，比如在小明在5月2号上午8点到下午5点需要工作，则为一个排班。排班的类型可以是正常排班或者加班。如果经理需要展开临时加班可以在这里为所有参与者创建类型为加班的排班实现。
4. 一个排班对应一个签到记录，签到时签两次，进公司一次，离开一次，用户通过调用**考勤打卡（）**方法，填入时间等信息，完成签到。正常和加班类型的排班都对应一个签到记录。
5. 一个用户可以拥有多段工时，每段工时都记录了工时的个数和日期等信息。用户可以通过调用**查看上班情况（）**方法来查看上班情况。此方法对于参与者身份不同，所拥有的权限不同。如果是普通员工，则可以看自己的上班情况，对于部门主管，可以看本部门所有员工的情况，对于经理，则可以查看所有员工的上班情况。类似地，**查看工作班次（）**方法也根据身份划分权限。
6. 类似地，对于用户的添加、删除、修改用户信息以及任用、罢免主管的方法都只有身份为经理才有权利执行。
7. 调整工作班次安排、审批请假，审批加班都需要部门主管才能调用。同样地，人脸信息的录入和修改、废弃也需要特定权限。（经理）
8. 经理的全单位临时加班活动可以通过给全单位所有人参加者都建立一个加班类型的排班来实现。员工可以选择参加临时加班活动，每参加一个临时加班活动，会生成一个加班类型的排班。同样的，临时加班类型的排班也对应一个签到记录，需要参与员工进行打卡。
9. 部门类中记录所有的部门，主要由部门经理进行增删改查和设置主管操作。
10. 消息类中记录系统运行过程中产生的消息，主要是通过请假、销假、加班申请等申请请求，和批准拒绝等审核类通知。消息的状态用已读和未读标记。

### 3.1.2 整体结构设计

#### 3.1.2.1 后端

##### 3.1.2.1.1 设计图



##### 3.1.2.1.2 设计说明

后端主要分为如上图所示的一些模块。

1. 接受请求之后先调用请求验证模块中的方法，对请求如JSON格式、权限、数据格式有效性的进行初步验证，验证通过进入后续逻辑处理模块，验证不通过直接进入错误处理。
2. 逻辑处理部分模块按照功能分为十大模块，分别处理工时相关、部门相关、用户相关、签到相关、加班相关、请假相关、人脸相关、排版相关、消息相关、状态相关的逻辑，包括一些数据处理，逻辑上的验证等，最后将复杂的逻辑简化为针对数据库的增删改查逻辑，然后进入数据库控制模块。（过程中出现任何问题即进入错误处理模块）
3. 数据库处理模块是唯一可以直接对数据库执行写操作的模块，主要进行对数据库简单的增删改查，过程中如果出现错误直接进入到错误处理模块，正常完成则进入到响应模块构造发送正确实现的响应。
4. 错误处理模块主要接收后端处理过程中发生的所有的错误，并且生成对应的错误提示送给响应模块。
5. 响应模块是负责和前端通信的模块之一，如果是错误响应，则发送错误对应状态码和错误提示，如果是完成响应，则发送正确完成的状态码和请求的数据。

#### 3.1.2.2 Web端

##### 3.1.2.2.1 设计图

##### 3.1.2.2.2 设计说明

#### 3.1.2.3 移动端

##### 3.1.2.3.1 设计图

##### 3.1.2.3.2 设计说明

## 3.2 详细设计

### 3.2.1数据库结构设计

### 3.2.2数据接口设计

|  |  |  |  |
| --- | --- | --- | --- |
| 接口 | URL | 附加 | 响应 |
| 错误格式响应 |  |  | {          "status": 4xx/5xx,          "message": "error message"      } |
| 用户认证 | POST `/user/auth` | JSON参数：  - name: 用户姓名  - password: 用户密码 | {          "status": 200,          "token": "{Token}"      } |
| 用户列表 | GET `/user` | Header：  Authorization: Bearer {Token}  URL Query参数：  - page: 分页 | {          "status": 200,          "total": 100,          "current\_page": 1,          "per\_page": 20,          "data": [              {                  "id": 1,                  "name": "test",                  "role": "user/manager/master",                  "department": {                      "id": 1,                      "name": "test"                  },                  "hours": 10              },              ...          ]      } |
| 添加用户 | POST `/user` | Header：  Authorization: Bearer {Token}  JSON参数：  - name: 姓名  - password: 密码  - department: 部门 | {          "status": 201,          "user\_id": 2      } |
| 批量添加用户 | POST `/user/batch` | Header：  Authorization: Bearer {Token}  JSON参数：  - batch: 批量用户信息 | [          {              "status": 201,              "user\_id": 2          },          {              "status": 400,              "message": "why failed"          },          ...      ] |
| 修改用户 | PUT `/user/{user\_id}` | Header：  Authorization: Bearer {Token}  JSON参数：  - name: 姓名  - department: 部门  - role: 身份  - hours: 工时 | {          "status": 200      } |
| 获取指定用户信息 | GET `/user/{user\_id}` | Header：  Authorization: Bearer {Token} | {          "status": 200,          "data": {               "id": 1,               "name": "test",               "role": "user/manager/master",               "department": {                   "id": 1,                   "name": "test"               },               "hours": 10          }      } |
| 删除用户 | DELETE `/user/{user\_id}` | Header：  Authorization: Bearer {Token} | {          "status": 200      } |
| 修改密码 | PUT `/user/{user\_id}  /password` | Header：  Authorization: Bearer {Token}  JSON参数：  - old\_password: 原密码，manager 和 master 权限可选  - new\_password: 新密码 | {          "status": 200      } |
| 部门列表 | GET `/department` | Header：  Authorization: Bearer {Token}  URL Query参数：  - page: 分页 | {          "status": 200,          "total": 100,          "current\_page": 1,          "per\_page": 20,          "data": [              {                  "id": 1,                  "name": "X 部门"              },              ...          ]      } |
| 添加部门 | POST `/department` | Header：  Authorization: Bearer {Token}  JSON参数：  - name: 部门名称 | {          "status": 201,          "department\_id": 2      } |
| 编辑部门 | PUT `/department  /{department\_id}` | Header：  Authorization: Bearer {Token}  JSON参数：  - name: 部门名称 | {          "status": 200      } |
| 获取指定部门信息 | GET `/department  /{department\_id}` | Header：  Authorization: Bearer {Token} | {          "status": 200,          "data": {              "id": 1,              ...          }      } |
| 删除部门 | DELETE `/department  /{department\_id}` | Header：  Authorization: Bearer {Token} | {          "status": 200      } |
| 获取指定用户可用人脸 | GET `/face/user/{user\_id}` | Header：  Authorization: Bearer {Token} | {          "status": 200,          "data": {              "id": 1,              "user": {                  "id": 1,                  "name": "test name",                  ...              },              "info": "xxxxxxx",              "status": "wait/available/discarded"          }      } |
| 更新用户人脸信息 | POST `/face/user/{user\_id}` | Header：  Authorization: Bearer {Token}  JSON参数：  - info: 人脸信息 | {          "status": 201,          "face\_id": 2      } |
| 获取人脸列表 | GET `/face` | Header：  Authorization: Bearer {Token}  URL Query参数：  - user\_id: 获取指定用户人脸信息，可选  - status: 获取制定状态人脸信息，可选  - page: 分页 | {          "status": 200,          "total": 100          "current\_page": 1          "per\_page": 20,          "data": [              {                  "id": 1,                  "user": {                      "id": 1,                      "name": "name",                      ...                  },                  "info": "xxxxxxx",                  "status": "wait/available/discarded"              },              ...          ]      } |
| 审批人脸信息 | PUT `/face/{face\_id}` | Header：  Authorization: Bearer {Token} | {          "status": 200      } |
| 获取工时记录 | GET `/hours` | Header：  Authorization: Bearer {Token}  URL Query参数：  - user\_id: 用户 ID，可选  - start\_at: 开始日期，可选  - end\_at: 结束日期，可选  - page: 分页 | {          "status": 200,          "total": 100,          "current\_page": 1,          "per\_page": 20,          "data": [              {                  "id": 1,                  "user": {                      "id": 1,                      "name": "test",                      ...                  },                  "date": "2019-02-02 11:11:11",                  "hours": 12              },              ...          ]      } |
| 排版列表 | GET `/shift` | Header：  Authorization: Bearer {Token}  URL Query参数：  - user\_id: 用户 ID，可选  - department\_id: 部门 ID，可选  - start\_at: 开始日期，可选  - end\_at: 结束日期，可选  - page: 分页 | {          "status": 200,          "total": 100,          "current\_page": 1,          "per\_page": 20,          "data": [              {                  "id": 1,                  "user": {                      "id": 1,                      "name": "test",                      ...                  },                  "start\_at": "2019-02-02 11:11:11",                  "end\_at": "2019-02-02 11:11:11",                  "type": "normal/allovertime",                  "status": "no/on/off/leave"              },              ...          ]      } |
| 添加排班 | POST `/shift/user/{user\_id}` | Header：  Authorization: Bearer {Token}  JSON参数：  - start\_at: 开始时间  - end\_at: 结束时间  - type: 排班类型 | {          "status": 201,          "shift\_id": 2      } |
| 部门排班 | POST `/shift/department  /{department\_id}` | Header：  Authorization: Bearer {Token}  JSON参数：  - start\_at: 开始时间  - end\_at: 结束时间  - type: 排班类型 | {          "status": 201,          "shift\_ids": [2, 3]      } |
| 全单位排版 | POST '/shift/all' | Header：  Authorization: Bearer {Token}  JSON参数：  - start\_at: 开始时间  - end\_at: 结束时间  - type: 排班类型 | {          "status": 201,          "shift\_ids": [2, 3]      } |
| 删除排版 | DELETE `/shift/{shift\_id}` | Header：  Authorization: Bearer {Token} | {          "status": 200      } |
| 修改排班 | PUT `/shift/{shift\_id}` | Header：  Authorization: Bearer {Token}  JSON参数：  - start\_at: 开始时间  - end\_at: 结束时间  - effect: temp/all  temp | {          "status": 200      } |
|  |  | Header：  Authorization: Bearer {Token} |  |
| 请假列表 | GET `/leave` | Header：  Authorization: Bearer {Token}  JSON参数：  - page: 分页 | {          "status": 200,          "total": 100,          "current\_page": 1,          "per\_page": 20,          "data": [              {                  "id": 1,                  "user": {                      "id": 1,                      "name": "name",                      ...                  },                  "start\_at": "2019-02-02 11:11:11",                  "end\_at": "2019-02-02 11:11:11",                  "remark": "身体原因",                  "status": "wait/pass/reject"              },              ...          ]      } |
| 获取指定用户请假 | GET `/leave/user/{user\_id}` | Header：  Authorization: Bearer {Token}  JSON参数：  - page: 分页 | {          "status": 200,          "page": 1,          "data": [              {                  "id": 1,                  "user": {                      "id": 1,                      "name": "name",                      ...                  },                  "start\_at": "2019-02-02 11:11:11",                  "end\_at": "2019-02-02 11:11:11",                  "remark": "身体原因",                  "status": "wait/pass/reject"              },              ...          ]      } |
| 申请请假 | POST `/leave/user/{user\_id}` | Header：  Authorization: Bearer {Token}  JSON参数：  - start\_at: 开始时间  - end\_at: 结束时间  - remark: 请假理由 | {          "status": 201,          "leave\_id": 2      } |
| 审批请假 | PUT `/leave/{leave\_id}` | Header：  Authorization: Bearer {Token}  JSON参数：  - status: 状态 | {          "status": 200      } |
| 销假 | DELETE `/leave/{leave\_id}` | Header：  Authorization: Bearer {Token} | {          "status": 200      } |
| 加班申请列表 | GET `/overtime` | Header：  Authorization: Bearer {Token}  JSON参数：  - page: 分页 | {          "status": 200,          "total": 100,          "current\_page": 1,          "per\_page": 20,          "data": [              {                  "id": 1,                  "user": {                      "id": 1,                      "name": "test",                      ...                  },                  "start\_at": "2019-02-02 11:11:11",                  "end\_at": "2019-02-02 11:11:11",                  "remark": "任务未完成",                  "status": "wait/pass/reject"              },              ...          ]      } |
| 获取指定用户加班 | GET `/overtime/user/{user\_id}` | Header：  Authorization: Bearer {Token}  JSON参数：  - page: 分页 | {          "status": 200,          "total": 100,          "current\_page": 1,          "per\_page": 20,          "data": [              {                  "id": 1,                  "user": {                      "id": 1,                      "name": "test",                      ...                  },                  "start\_at": "2019-02-02 11:11:11",                  "end\_at": "2019-02-02 11:11:11",                  "remark": "任务未完成",                  "status": "wait/pass/reject"              },              ...          ]      } |
| 申请加班 | POST `/overtime  /user/{user\_id}` | Header：  Authorization: Bearer {Token}  JSON参数：  - remark: 理由 | {          "status": 201,          "overtime\_id": 2      } |
| 审批加班 | PUT `/overtime/{overtime\_id}` | Header：  Authorization: Bearer {Token}  JSON参数：  - status: 状态 | {          "status": 200      } |
| 获取用户当前签到状态 | GET `/sign/user/{user\_id}` | Header：  Authorization: Bearer {Token} | {          "status": 200,          "sign\_id": 2,          "shift": {              "id": 1,              "start\_at": "2019-xxxx",              "end\_at": "2019-xxxx",              ...          }      } |
| 获取二维码 | GET `/sign/qrcode` | Header：  Authorization: Bearer {Token} | {          "status": 200,          "data": {              "qrcode": "data:image/png;...",              "expired\_at": "2019-02-02 11:11:11"          }      } |
| 二维码签到 | POST `/sign/qrcode/{user\_id}` | Header：  Authorization: Bearer {Token}  JSON参数：  - token: 签到 Token | {          "status": 200,          "sign\_id: 3,          "user": {              "id": 1,              "name": "xxx",              ...          }      } |
| 人脸签到 | POST `/sign/face` | Header：  Authorization: Bearer {Token}  JSON参数：  - face: 人脸图片的 base64 编码 | {          "status": 200,          "sign\_id: 3,          "user": {              "id": 1,              "name": "xxx",              ...          }      } |
| 签退 | POST `/sign/off/{sign\_id}` | Header：  Authorization: Bearer {Token} | {          "status": 200,          "overtime": true      } |
| 获取信息列表 | GET `/message` | Header：  Authorization: Bearer {Token}  URL Query参数：  - from\_user\_id: 获取指定发送用户的信息，可选  - to\_user\_id: 获取指定接收用户的信息，可选  - status: 获取指定状态的信息，可选  - page: 分页 | {          "status": 200,          "total": 100,          "current\_page": 1,          "per\_page": 20,          "data": [              {                  "id": 1,                  "from": {                      "id": 1,                      "name": "test",                      ...                  },                  "to": {                      "id": 2,                      "name": "test2",                      ...                  },                  "title": "test title",                  "status": "unread/read"              },              ...          ]      } |
| 获取指定信息 | GET `/message/{message\_id}` | Header：  Authorization: Bearer {Token} | {          "status": 200,          "data": {              "id": 1,              "from": {                  "id": 1,                  "name": "test",                  ...              },              "to": {                  "id": 2,                  "name": "test2",                  ...              },              "title": "test title",              "content": "test content",              "status": "unread/read"          }      } |
| 获取用户相关数据 | GET `/status/user` | Header：  Authorization: Bearer {Token} | {          "status": 200,          "data": {              "users": 100          }      } |
| 获取签到相关数据 | GET `/status/sign` | Header：  Authorization: Bearer {Token} | {          "status": 200,          "data": {              "signed": 10,              "lated": 3,              "leaved": 2          }      } |
| 获取用户的工作时间和加班时间 | GET '/status/hours/{user\_id}' | Header：  Authorization: Bearer {Token}  JSON参数：  - start\_at: 获取开始时间，必选  - end\_at: 获取结束时间，必选 | {           "status": 200,           "shift\_hour": 300,           "overtime\_hour": 100       } |

### 3.2.3人机接口设计

#### 3.2.3.1Web端

#### 3.2.3.2移动端

### 3.2.4数据驱动设计

# 四、系统实现

（介绍代码结构、存放地点、关键代码逻辑等等）

## 后端

### 4.1.1 代码结构

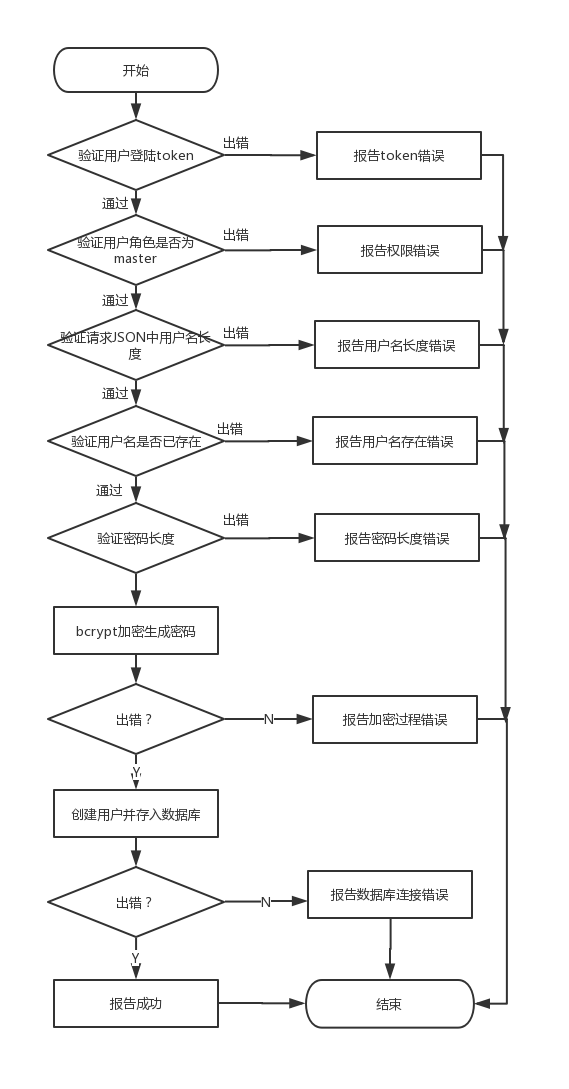
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 结构 | | | | 介绍 |
| server | config | config.go |  | 配置信息 |
| controllers | department.go |  | 部门相关操作 |
| face.go |  | 人脸相关操作 |
| home.go |  | 默认页面操作 |
| hours.go |  | 工时相关操作 |
| leave.go |  | 请假相关操作 |
| message.go |  | 消息相关操作 |
| overtime.go |  | 加班相关操作 |
| shift.go |  | 排版相关操作 |
| sign.go |  | 签到相关操作 |
| status.go |  | 状态相关操作 |
| user.go |  | 用户相关操作 |
| middleware | auth.go |  | 登陆验证 |
| manager.go |  | 用户为主管及以上权限验证 |
| master.go |  | 用户为经理的权限验证 |
| models | common.go |  | 模型公有字段定义 |
| department.go |  | 部门模型定义 |
| face.go |  | 人脸模型定义 |
| hours.go |  | 工时模型定义 |
| leave.go |  | 请假模型定义 |
| message.go |  | 消息模型定义 |
| overtime.go |  | 加班模型定义 |
| qrcode.go |  | 二维码模型定义 |
| shift.go |  | 排版模型定义 |
| sign.go |  | 签到模型定义 |
| user.go |  | 用户模型定义 |
| requests | department.go |  | 部门相关请求JSON验证 |
| face.go |  | 人脸相关请求JSON验证 |
| leave.go |  | 请假相关请求JSON验证 |
| overtime.go |  | 加班相关请求JSON验证 |
| shift.go |  | 排版相关请求JSON验证 |
| sign.go |  | 签到相关请求JSON验证 |
| user.go |  | 用户相关请求JSON验证 |
| response | common.go |  | 公共响应 |
| department.go |  | 部门相关响应构造发送 |
| face.go |  | 人脸相关响应构造发送 |
| hours.go |  | 工时相关响应构造发送 |
| leave.go |  | 请假相关响应构造发送 |
| message.go |  | 消息相关响应构造发送 |
| overtime.go |  | 加班相关响应构造发送 |
| shift.go |  | 排版相关响应构造发送 |
| sign.go |  | 签到相关响应构造发送 |
| status.go |  | 状态相关响应构造发送 |
| user.go |  | 用户相关响应构造发送 |
| router | router.go |  | 路由设置 |
| utils | common | common.go | 共同常用操作 |
| department.go | 部门常用操作 |
| database | db.go | 数据库连接操作 |
| migration.go | 数据库迁移填充操作 |
| message | message.go | 消息常用操作 |
| qcloud | iai.go | 二维码和人脸库的初始化和常用操作 |
| main.go |  |  | 程序入口 |

### 4.1.2 存放地点

### 4.1.3 关键代码逻辑

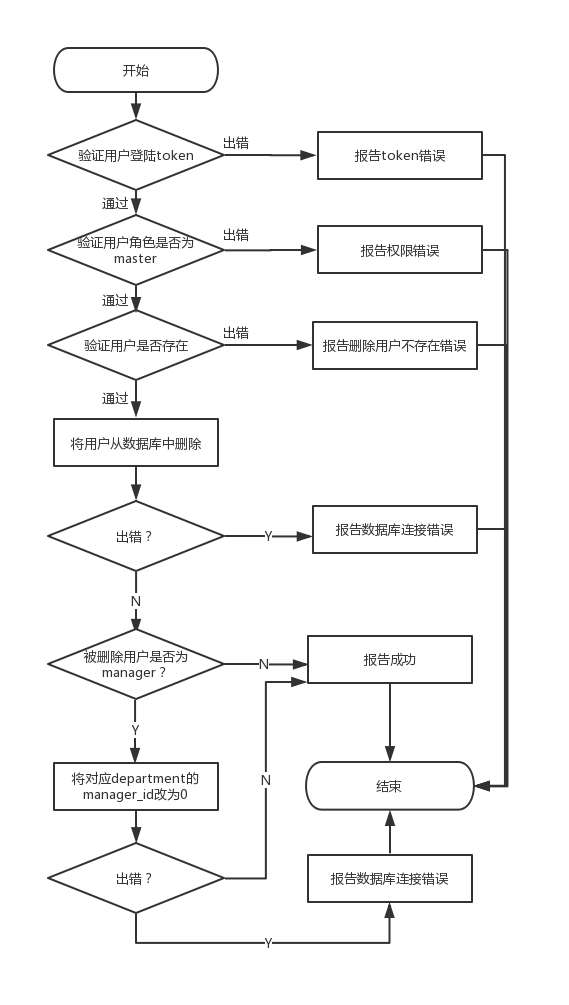
#### 4.1.3.1 添加

以添加用户为例，代码逻辑如下：



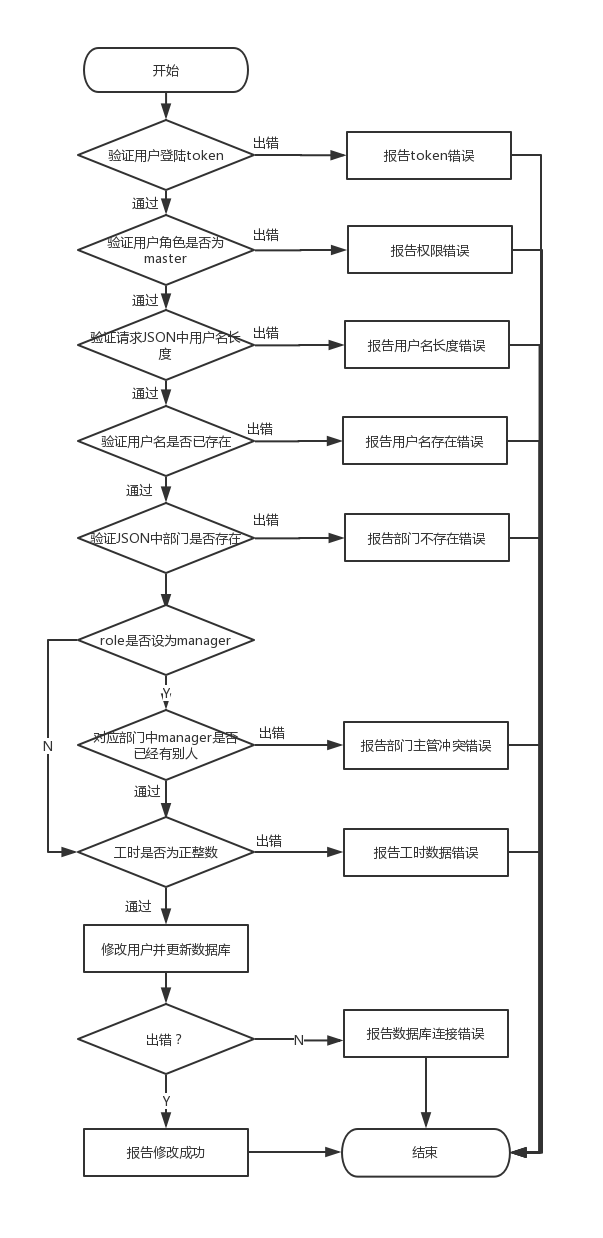
#### 4.1.3.2删除

以删除用户为例，代码逻辑如下：



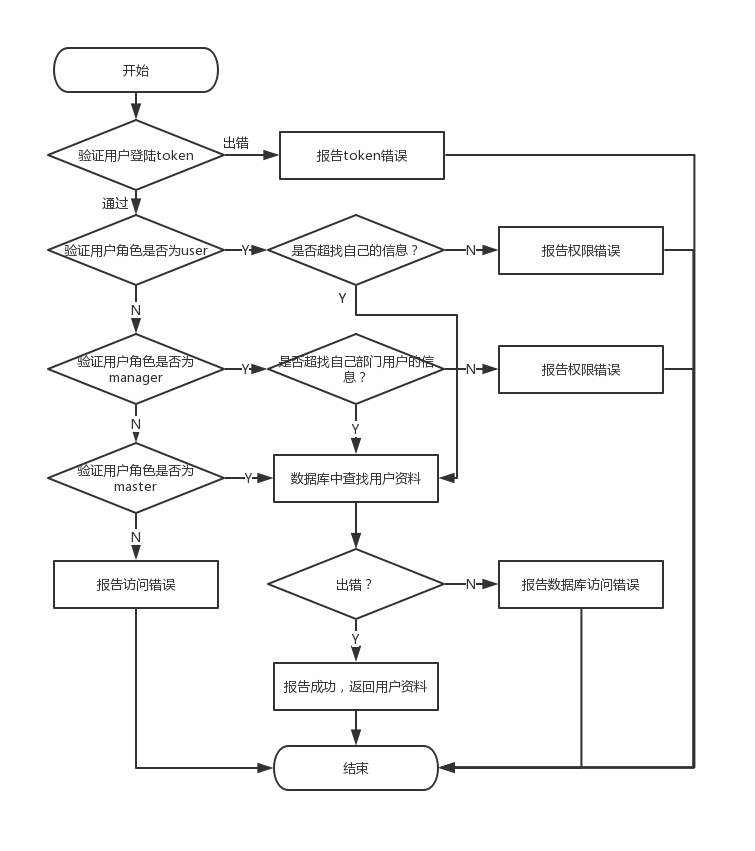
#### 4.1.3.3 修改

以修改用户资料为例，代码逻辑如下：



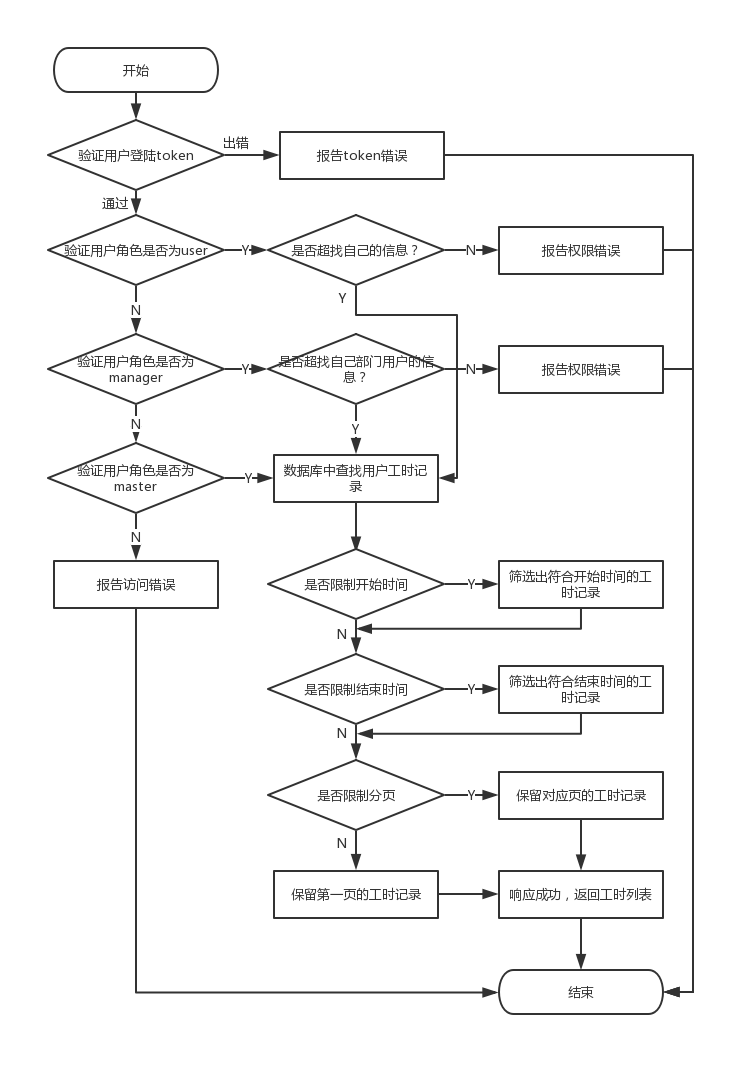
#### 4.1.3.4 查找指定目标

以超找指定用户资料为例，代码逻辑如下：



#### 4.1.3.5 查找列表

以获取工时列表为例，代码逻辑如下：



## Web端

## 4.3 移动端

# 五、系统测试

## 5.1 测试方案

### 5.1.1 后端

### 5.1.2 Web端

### 5.1.3 移动端

## 5.2 测试结果

### 5.1.1 后端

### 5.1.2 Web端

### 5.1.3 移动端

## 5.3 分析

# 六、部署和运行

（介绍系统部署和运行的情况）

## 参考文献

[1] …

[2] …