1 社保归档

1.1 归档历史列表

（1）controller

```
    //查询某年内社保归档列表"
    @RequestMapping(value = "/historys/{year}/list", method = RequestMethod.GET)
    public Result historiesList(@PathVariable(value = "year") String year)
throws Exception {
        List<Archive> archiveVoList =
archiveService.findArchiveByYear(companyId,year);
        return new Result(ResultCode.SUCCESS, archiveVoList);
    }
```

（2）service

```
    public List<Archive> findArchiveByYear(String companyId, String yearMonth) {
        return archiveDao.findByCompanyIdAndYearsMonthLike(companyId,
yearMonth);
    }
```

1.2 归档详情

（1）controller

```
    @RequestMapping(value = "/historys/{yearMonth}", method = RequestMethod.GET)
    public Result histories(@ApiParam(value = "年月", required = true)
@PathVariable(value = "yearMonth") String yearMonth, @ApiParam(value = "操作类型
1-未归档数据,其他:已归档数据", required = true) @RequestParam(value = "opType")
Integer opType) {
        List<ArchiveDetail> reportVoList = new ArrayList<>();
        if (opType == 1) {
            reportVoList.addAll(archiveService.getReports(yearMonth,companyId));
        } else {
            Archive archive = archiveService.findArchive(companyId, yearMonth);
            if (archive != null) {
                reportVoList =
archiveService.findAllDetailByArchiveId(archive.getId());
            }
        }
        return new Result(ResultCode.SUCCESS, reportVoList);
    }
```

（2）service

```
    public List<ArchiveDetail> findAllDetailByArchiveId(String archiveId) {
        return archiveDetailDao.findByArchiveId(archiveId);
    }
```

（1）controller

```
    @RequestMapping(value = "/historys/data", method = RequestMethod.GET)
    public ArchiveDetail histories(@RequestParam(value = "userId") String
userId, @RequestParam(value = "yearMonth") String yearMonth) {
        return archiveService.findUserArchiveDetail(userId,yearMonth);
    }
```

（2）service

```
    public ArchiveDetail findUserArchiveDetail(String userId, String yearMonth)
{
        return archiveDetailDao.findByUserIdAndYearsMonth(userId,yearMonth);
    }s
```

2 考勤设置

考勤，顾名思义，就是考查出勤，也是就通过某种方式来获得学生、员工或者某些团体、个人在某个特定的场所及特定的时间段内的出勤情况，包括上下班、迟到、早退、病假、婚假、丧假、公休、工作时间、加班情况等。通过对以前阶段，本阶段内出勤情况的研究，进行以后阶段的统筹、安排等。

2.1 需求分析

不同的企业,不同的部门考勤核算是不一样的。所以需要对考勤的出勤时间，请假约定，出勤不足是否扣款，以及加班规则进行设置

- 出勤设置

固定班制是大多数公司较为常见的工作模式，可以根据需要设置上班时间，下班时间。管理员可以根据公司的考勤时间，改变时间设置

- 请假设置

员工在正常工作日可能需要进行请假，调休等操作，管理员可以根据公司的实际情况进行请假控制。如果当请假开关关闭后，员工则不可在员工端提交请假申请

作为一种惩罚制度，扣款可能是大部分公司都会采用的模式，可以设置迟到，早退，旷工的扣款开关，当迟到超过固定的时间，且次数大于某次的时候，将扣除工资

- 加班设置

管理员可以根据不同的部门进行加班设置。

- 法定节假日

2.2 表分析

考勤配置表

```
CREATE TABLE `atte_attendance_config` (
  `id` varchar(40) NOT NULL COMMENT '主键ID',
  `company_id` varchar(40) DEFAULT NULL COMMENT '公司ID',
  `department_id` varchar(40) DEFAULT NULL COMMENT '部门ID',
  `morning_start_time` time DEFAULT NULL COMMENT '上午打卡时间',
  `morning_end_time` time DEFAULT NULL COMMENT '上午打卡时间',
  `afternoon_start_time` time DEFAULT NULL COMMENT '下午打卡时间',
  `afternoon_end_time` time DEFAULT NULL COMMENT '下午打卡时间',
  `create_by` varchar(64) DEFAULT NULL,
  `create_date` datetime DEFAULT NULL,
  `update_by` varchar(64) DEFAULT NULL,
  `update_date` datetime DEFAULT NULL,
  `remarks` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`) USING BTREE,
  UNIQUE KEY `company_id` (`company_id`,`department_id`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC COMMENT='考勤配置表'
```

调休配置表

```
CREATE TABLE `atte_day_off_config` (
  `id` varchar(40) NOT NULL COMMENT '主键ID',
  `company_id` varchar(40) DEFAULT NULL COMMENT '公司ID',
  `department_id` varchar(40) DEFAULT NULL COMMENT '部门ID',
  `latest_effect_date` datetime DEFAULT NULL COMMENT '调休最后有效日期',
  `unit` varchar(20) DEFAULT NULL COMMENT '调休单位(天最小0.5)',
  `create_by` varchar(64) DEFAULT NULL,
  `create_date` datetime DEFAULT NULL,
  `update_by` varchar(64) DEFAULT NULL,
  `update_date` datetime DEFAULT NULL,
  `remarks` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`) USING BTREE,
  UNIQUE KEY `company_id` (`company_id`,`department_id`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC COMMENT='调休配置表'
```

扣款字典表

```
CREATE TABLE `atte_deduction_dict` (
  `id` varchar(40) NOT NULL COMMENT '主键ID',
  `company_id` varchar(40) DEFAULT NULL COMMENT '公司ID',
  `department_id` varchar(40) DEFAULT NULL COMMENT '部门ID',
  `ded_type_code` varchar(20) DEFAULT NULL COMMENT '扣款类型编码',
```

```
  `times_lower_limit` varchar(20) DEFAULT NULL COMMENT '次数下限',
  `times_upper_limit` varchar(20) DEFAULT NULL COMMENT '次数上限',
  `ded_amonut_lower_limit` decimal(20,0) DEFAULT NULL COMMENT '扣款金额下限',
  `ded_amonut_upper_limit` decimal(20,0) DEFAULT NULL COMMENT '扣款金额上限',
  `absence_times_upper_limt` varchar(20) DEFAULT NULL COMMENT '旷工次数上限',
  `absence_days` varchar(20) DEFAULT NULL COMMENT '旷工天数',
  `fine_salary_multiples` varchar(20) DEFAULT NULL COMMENT '罚款工资倍数',
  `is_absenteeism` varchar(20) DEFAULT NULL COMMENT '是否旷工0不旷工1旷工',
  `is_enable` int(20) DEFAULT NULL COMMENT '是否可用 0开启 1 关闭',
  `create_by` varchar(64) DEFAULT NULL,
  `create_date` datetime DEFAULT NULL,
  `update_by` varchar(64) DEFAULT NULL,
  `update_date` datetime DEFAULT NULL,
  `remarks` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`) USING BTREE,
  UNIQUE KEY `company_id` (`company_id`,`department_id`,`ded_type_code`) USING
BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC COMMENT='扣款字典表'
```

扣款类型表

```
CREATE TABLE `atte_deduction_type` (
  `code` varchar(20) DEFAULT NULL COMMENT '扣款类型编码',
  `description` varchar(60) DEFAULT NULL COMMENT '扣款类型编码说明',
  `create_by` varchar(64) DEFAULT NULL,
  `create_date` datetime DEFAULT NULL,
  `update_by` varchar(64) DEFAULT NULL,
  `update_date` datetime DEFAULT NULL,
  `remarks` varchar(255) DEFAULT NULL,
  UNIQUE KEY `code` (`code`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC COMMENT='扣款类型表'
```

加班配置表

```
CREATE TABLE `atte_extra_duty_config` (
  `id` varchar(40) NOT NULL COMMENT '主键ID',
  `company_id` varchar(40) DEFAULT NULL COMMENT '公司ID',
  `department_id` varchar(40) DEFAULT NULL COMMENT '部门ID',
  `work_hours_day` int(20) DEFAULT NULL COMMENT '每日标准工作时长，单位小时',
  `is_clock` int(20) DEFAULT NULL COMMENT '是否打卡0开启1关闭',
  `is_compensationint` varchar(30) DEFAULT NULL COMMENT '是否开启加班补偿0开启1关
闭',
  `create_by` varchar(64) DEFAULT NULL,
  `create_date` datetime DEFAULT NULL,
  `update_by` varchar(64) DEFAULT NULL,
  `update_date` datetime DEFAULT NULL,
  `remarks` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`) USING BTREE,
  UNIQUE KEY `company_id` (`company_id`,`department_id`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC COMMENT='加班配置表'
```

加班规则

```
CREATE TABLE `atte_extra_duty_rule` (
```

```
  `company_id` varchar(40) DEFAULT NULL COMMENT '公司ID',
  `department_id` varchar(40) DEFAULT NULL COMMENT '部门ID',
  `rule` varchar(150) DEFAULT NULL COMMENT '规则内容',
  `rule_start_time` time DEFAULT NULL COMMENT '规则生效每日开始时间',
  `rule_end_time` time DEFAULT NULL COMMENT '规则生效每日结束时间',
  `is_time_off` varchar(10) DEFAULT NULL COMMENT '是否调休0不调休1调休',
  `is_enable` int(10) DEFAULT NULL COMMENT '是否可用 0开启 1 关闭',
  `create_by` varchar(64) DEFAULT NULL,
  `create_date` datetime DEFAULT NULL,
  `update_by` varchar(64) DEFAULT NULL,
  `update_date` datetime DEFAULT NULL,
  `remarks` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC COMMENT='加班规则'
```

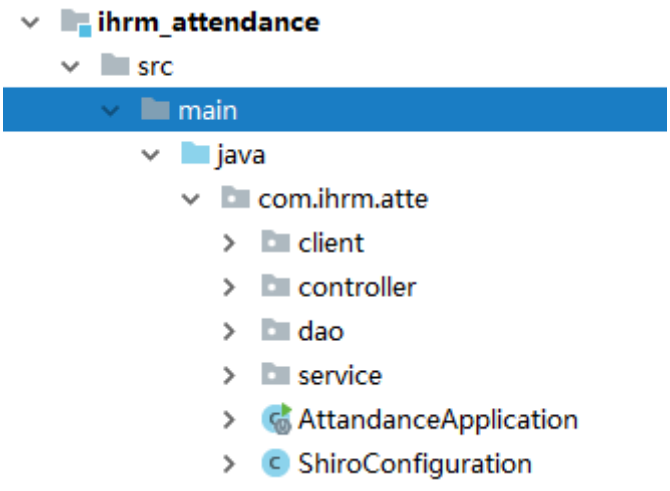
请假配置表

```
CREATE TABLE `atte_leave_config` (
  `id` varchar(40) NOT NULL COMMENT '主键ID',
  `company_id` varchar(40) DEFAULT NULL COMMENT '公司ID',
  `department_id` varchar(40) DEFAULT NULL COMMENT '部门ID',
  `leave_type` varchar(30) DEFAULT NULL COMMENT '请假类型',
  `is_enable` int(10) DEFAULT NULL COMMENT '是否可用 0开启 1 关闭',
  `create_by` varchar(64) DEFAULT NULL,
  `create_date` datetime DEFAULT NULL,
  `update_by` varchar(64) DEFAULT NULL,
  `update_date` datetime DEFAULT NULL,
  `remarks` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`) USING BTREE,
  UNIQUE KEY `company_id` (`company_id`,`department_id`,`leave_type`) USING
BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC COMMENT='请假配置表'
```

2.3 环境搭建

2.3.1 创建考勤模块

创建考勤模块`ihrm_attendance`



使用代码生成工具，生成响应表的实体类，dao接口，service层代码，并导入到项目中

2.3.3 配置

（1）配置启动类

```
@SpringBootApplication(scanBasePackages = "com.ihrm")
@EntityScan(basePackages = {"com.ihrm.domain"})
@EnableEurekaClient
@EnableFeignClients
public class AttandanceApplication {

    public static void main(String[] args) {
        SpringApplication.run(AttandanceApplication.class, args);
    }

    @Bean
    public IdWorker idWorkker() {
        return new IdWorker(1, 1);
    }
}
```

（2）设置网管的地址映射

```
    ihrm-attendance: #考勤
      path: /attendances/** #配置请求URL的请求规则
      serviceId: ihrm-attendance #指定Eureka注册中心中的服务id
      strip-prefix: false
      sentiviteHeaders:
      customSensitiveHeaders: true
    ihrm-atte-archive: #考勤归档
      path: /archive/** #配置请求URL的请求规则
      serviceId: ihrm-attendance #指定Eureka注册中心中的服务id
      strip-prefix: false
      sentiviteHeaders:
      customSensitiveHeaders: true
    ihrm-atte-cfg: #考勤配置
      path: /cfg/** #配置请求URL的请求规则
      serviceId: ihrm-attendance #指定Eureka注册中心中的服务id
      strip-prefix: false
      sentiviteHeaders:
      customSensitiveHeaders: true
    ihrm-atte-report: #考勤报表
      path: /report/** #配置请求URL的请求规则
      serviceId: ihrm-attendance #指定Eureka注册中心中的服务id
      strip-prefix: false
      sentiviteHeaders:
      customSensitiveHeaders: true
```

2.4 代码实现

```
    /**
     * 考勤保存更新
```

```
    public Result atteSaveOrUpdate(HttpServletRequest request , @RequestBody
@Valid AttendanceConfig attendanceConfig){
        //公共
        attendanceConfig.setCompanyId(this.companyId);

 configurationService.attendanceConfigSaveOrUpdate(attendanceConfig,this.userId)
;
        return  new Result(ResultCode.SUCCESS);
    }

    /**
     * 请假保存更新
     */
    @RequestMapping(value = "/leave", method = RequestMethod.PUT)
    public Result leaveSaveOrUpdate(@RequestBody  List<LeaveConfig>
leaveConfigList){
        //公共
        for (LeaveConfig leaveConfig:leaveConfigList) {

            leaveConfig.setCompanyId(this.companyId);

 configurationService.leaveConfigSaveOrUpdate(leaveConfig,this.userId);
        }
        return  new Result(ResultCode.SUCCESS);
    }

    /**
     * 扣款保存更新
     */
    @RequestMapping(value = "/deduction", method = RequestMethod.PUT)
    public Result deductionSaveOrUpdate(HttpServletRequest request ,
@RequestBody /*@Valid*/ List<DeductionDict> deductionDictList){
        for (DeductionDict deductionDict : deductionDictList ) {
            //公共
            deductionDict.setCompanyId(this.companyId);

 configurationService.deductionDictSaveOrUpdate(deductionDict,this.userId);
        }
        return  new Result(ResultCode.SUCCESS);
    }

    /**
     * 加班保存更新
     */
    @RequestMapping(value = "/extDuty", method = RequestMethod.PUT)
    public Result extDutySaveOrUpdate(HttpServletRequest request , @RequestBody
@Valid ExtDutyVO atteExtDutyVO){
        atteExtDutyVO.setCompanyId(this.companyId);
        configurationService.extDutySaveOrUpdate(atteExtDutyVO,this.userId);
        return  new Result(ResultCode.SUCCESS);
    }

    /**
     * 考勤设置信息查询
     */
    @RequestMapping(value = "/atte/item", method = RequestMethod.POST)
```

```
configurationService.getAtteCfgItem(companyId,departmentId);
        return new Result(ResultCode.SUCCESS,atteCfgItem);
    }

    /**
     * 请假设置信息查询
     */
    @RequestMapping(value = "/leave/list", method = RequestMethod.POST)
    public Result leaveCfgItem(String  departmentId) throws CommonException {
        List<LeaveConfig> leaveConfigList =
configurationService.getLeaveCfg(companyId,departmentId);
        return  new Result(ResultCode.SUCCESS,leaveConfigList);
    }

    /**
     * 扣款设置信息查询
     */
    @RequestMapping(value = "/ded/list", method = RequestMethod.POST)
    public Result dedCfgItem(String  departmentId) throws Exception {
        //公共
        List<DeductionDict> deductionDictList =
configurationService.getDedCfgList(companyId,departmentId);
        return new Result(ResultCode.SUCCESS,deductionDictList);
    }

    /**
     * 加班设置信息查询
     */
    @RequestMapping(value = "/extDuty/item", method = RequestMethod.POST)
    public Result extWorkCfgItem(String  departmentId) throws CommonException {
        Map map = configurationService.getExtWorkCfg(companyId,departmentId);
        return new Result(ResultCode.SUCCESS,map);
    }
```

3 考勤管理

考勤模块是为了减轻手工考勤的繁杂工作，做到公平可信，建立严格的出勤制度，从而提高企业的执行效率。考勤统计数据根据设定自动转入薪资模块，按照既定公式进行薪资计算。可以为某些行业特殊的人员生产率报表提供可靠的数字

序号	姓名	工号	部门	手机	8/1	8/2	8/3	8/4	8/5	8/6	8/7	8/8
0	itcast	9002	test1	13800000002	√	旷工	旷工	旷工	旷工	旷工	旷工	旷工
1	zbz	111	测试部	13800000003	旷工	补签	旷工	旷工	旷工	旷工	旷工	旷工
2	ll	1111	测试部	13800000004	旷工	旷工	旷工	旷工	旷工	旷工	旷工	旷工
3	a01	1001	开发部	13400000001	旷工	旷工	旷工	旷工	旷工	旷工	旷工	旷工
4	a02	1002	开发部	13400000002	旷工	旷工	旷工	旷工	旷工	旷工	旷工	旷工
5	test001	2001	开发部	13500000001	旷工	旷工	旷工	旷工	旷工	旷工	旷工	旷工
6	test002	2002	开发部	13500000002	旷工	旷工	旷工	旷工	旷工	旷工	旷工	旷工
7	test003	2003	开发部	13500000003	旷工	旷工	旷工	旷工	旷工	旷工	旷工	旷工

3.1 数据导入

1. 通过将IHRM系统和考勤机联通来实现底层代码的数据互通
2. 将考勤机系统导出为固定格式的Excel文件，再将excel文件导入到IHRM系统

为了系统的适应性更加广泛，我们采用第二种方案。

（1）controller代码

```
    /**
     * 考勤导入
     */
    @RequestMapping(value = "/import", method = RequestMethod.POST)
    public Result importAttendances(@RequestParam(name = "file") MultipartFile
file) throws Exception {
        excelImportService.importAttendanceExcel(file,this.companyId);
        return new Result(ResultCode.SUCCESS);
    }
```

（2）service代码

```
  public void importAttendanceExcel(MultipartFile file,String companyId) throws
Exception{
        //将导入的excel解析为list集合
        List<AtteUploadVo> list = new
ExcelImportUtil(AtteUploadVo.class).readExcel(file.getInputStream(), 1, 0);
        //循环获取每个用户数据
        for (AtteUploadVo atteUploadVo : list) {
            //查询用户
            User user = userDao.findByMobile(atteUploadVo.getMobile());
            //构造考勤对象
            Attendance attendance = new Attendance(atteUploadVo,user);

 attendance.setDay(com.ihrm.common.utils.DateUtil.parseDate2String(atteUploadVo.
getInTime(), "yyyyMMdd"));
            String mMdd =
com.ihrm.common.utils.DateUtil.parseDate2String(atteUploadVo.getInTime(),
"MMdd");
            if(holidays.contains(mMdd)) {
                attendance.setAdtStatu(23); //休息
            }else
if(com.ihrm.common.utils.DateUtil.isWeekend(atteUploadVo.getInTime()) ||
!workingDays.contains(mMdd)){
                attendance.setAdtStatu(23); //休息
            }else{
                //查询考勤设置
                AttendanceConfig config =
attendanceConfigDao.findByCompanyIdAndDepartmentId(companyId,
user.getDepartmentId());
                //如果上班时间大于考勤上班时间 - 迟到

 if(com.ihrm.common.utils.DateUtil.comparingDate(config.getMorningStartTime(),at
teUploadVo.getInTime())) {
                    attendance.setAdtStatu(3);
                }
                //如果下班时间大于考勤下班时间 - 早退
```

```
teUploadVo.getOutTime())) {
                    attendance.setAdtStatu(4);
                }
            }
            Attendance exis = attendanceDao.findByUserIdAndDay(user.getId(),
attendance.getDay());
            if(exis == null) {
                attendance.setId(idWorker.nextId() + "");
                attendanceDao.save(attendance);
            }
        }
    }
```

3.2 考勤列表

(1) controller代码

```
    @RequestMapping(value = "", method = RequestMethod.GET)
    public Result atteStatisMonthly(@RequestBody(required = false)
Map<String,Object> map,int page,int pagesize) throws CommonException {
        if(map == null) {
            map = new HashMap<>();
        }
        map.put("companyId",companyId);
        Map resultMap = new HashMap();
        resultMap.put("data",atteService.getAtteList(map, page, pagesize));
        resultMap.put("monthOfReport",LocalDate.now().getMonthValue());
        resultMap.put("tobeTaskCount",0);
        return new Result(ResultCode.SUCCESS,resultMap);
    }
```

(2) service代码

```
    public PageResult getAtteList(Map<String,Object> map, int page, int size)
throws CommonException {

        Result result = systemFeignClient.findAll(page, size);

        PageResult<User> pr = BeanConvert.convertValue(result.getData(),
PageResult.class);

        List list = pr.getRows();

        List<AtteItemBO> atteItemVoList = new ArrayList<>();

        for (Object obj : list) {
            User user  = BeanConvert.convertValue(obj, User.class);
            AtteItemBO atteItemVO = new AtteItemBO();
            BeanUtils.copyProperties(user,atteItemVO);
            //考勤记录
            List<Attendance> attendanceRecordList = new ArrayList<>();
            //获取本月的天数
            List<DaysMonthlyBO> daysMonthlyBOList =
daysMonthlyDao.getDaysMonthly();
```

```
                String day = dayMonthlyBO.getDay();
                Attendance attendance =
attendanceDao.findByUserIdAndDay(user.getId(),day);
                if(attendance == null) {
                    attendance = new Attendance();
                    attendance.setAdtStatu(2);//考勤状态，没有旷工
                    attendance.setId(user.getId());
                    attendance.setDay(day);
                }
                attendanceRecordList.add(attendance);
            }
            atteItemVO.setAttendanceRecord(attendanceRecordList);
            atteItemVoList.add(atteItemVO);
        }
        return new PageResult(pr.getTotal(),atteItemVoList);
```

3.3 考勤修改

(1) controller代码

```
/**
     * 考勤编辑
     */
@RequestMapping(value = "/{id}", method = RequestMethod.PUT)
public Result atteEdit(@RequestBody Attendance attendance) throws
CommonException {
    attendance.setCompanyId(companyId);
    atteService.editAtte(attendance);
    return new Result(ResultCode.SUCCESS);
}
```

(2) service代码

```
public void editAtte(Attendance attendance) throws CommonException {
    Attendance vo = attendanceDao.findByUserIdAndDay(attendance.getUserId(),
attendance.getDay());
    if(vo == null) {
        attendance.setId(idWorker.nextId()+ "");
    }
    attendanceDao.save(attendance);
}
```

3.4 考勤归档

3.4.1 查看月报表

(1) controller

```
    public Result reports(String atteDate) throws CommonException {
        List<ArchiveMonthlyInfo> reports = archiveService.getReports(atteDate,
companyId);
        return  new Result(ResultCode.SUCCESS,reports);
    }
```

(2) service

```
    public List<ArchiveMonthlyInfo> getReports(String atteDate, String
companyId) {
        List<User> users  = userDao.findByCompanyId(companyId);
        List<ArchiveMonthlyInfo> infos = new ArrayList<>();
        for (User user : users) {
            ArchiveMonthlyInfo info = new ArchiveMonthlyInfo(user);
            Map map = attendanceDao.statisByUser(user.getId(),atteDate+"%");
            info.setStatisData(map);
            infos.add(info);
        }
        return infos;
    }
```

3.4.2 数据归档

(1) controller

```
    @RequestMapping(value = "/archives", method = RequestMethod.GET)
    public Result archives(String atteDate){
        archiveService.saveArchive(atteDate,companyId);
        return  new Result(ResultCode.SUCCESS);
    }
```

(2) service

```
    @Transactional(rollbackFor = Exception.class)
    public void saveArchive(String yearMonth,String companyId) {

        List<User> users  = userDao.findByCompanyId(companyId);

        ArchiveMonthly archiveMonthly = new ArchiveMonthly();
        archiveMonthly.setId(String.valueOf(idWorkker.nextId()));
        archiveMonthly.setCompanyId(companyId);
        archiveMonthly.setArchiveYear(yearMonth.substring(0,4));
        archiveMonthly.setArchiveMonth(yearMonth.substring(5,7));

        for (User user : users) {
            ArchiveMonthlyInfo info = new ArchiveMonthlyInfo(user);
            Map map = attendanceDao.statisByUser(user.getId(),yearMonth+"%");
            info.setAtteArchiveMonthlyId(archiveMonthly.getId());
            info.setStatisData(map);
            info.setId(idWorkker.nextId()+"");
            archiveMonthlyInfoDao.save(info);
        }
```

```
        //全勤认数
        archiveMonthly.setFullAttePeopleNum(users.size());
        //是否归档
        archiveMonthly.setIsArchived(0);

        atteArchiveMonthlyDao.save(archiveMonthly);
    }
```

（3）归档列表与详情

```
    /**
     * 归档列表
     */
    @RequestMapping(value = "/reports/year", method = RequestMethod.GET)
    @ResponseBody
    public Result archiveList(String year){
        List<ArchiveMonthly> atteArchiveMonthlies =
archiveService.findArchiveListByYear(companyId,year);
        return  new Result(ResultCode.SUCCESS,atteArchiveMonthlies);
    }


    /**
     * 归档详情列表
     */
    @RequestMapping(value = "/reports/{month}", method = RequestMethod.GET)
    @ResponseBody
    public Result archiveInfo(String atteArchiveMonthlyId){
        List<ArchiveMonthlyInfo> archiveMonthlyInfo =
archiveService.findArchiveInfoById(atteArchiveMonthlyId);
        return  new Result(ResultCode.SUCCESS,archiveMonthlyInfo);
    }
```

4 薪资管理

薪资管理是指企业制定的合理的工资发放制度及系统，包括不同员工的薪资标准、薪资的明确组成部分、发放薪资的政策、薪资发放办法和原则、对该员工工作评价制度和薪资评价制度等。薪资管理针对不同的企业有不同的模式，薪资管理是企业管理的重要组成部分。

部门：

序号	姓名	手机	工号	聘用形式	部门	入职时间	工资基数	津贴方案	操作
1	itcast	13800000002	9002	正式	test1	2018-11-02	10	通用方案	调薪 查看
2	zbz	13800000003	111	正式	测试部	2018-11-04	8	通用方案	调薪 查看
3	ll	13800000004	1111	正式	测试部	2018-12-02	6	通用方案	调薪 查看
4	a01	13400000001	1001	正式	开发部	2018-01-01		通用方案	定薪 查看
5	a02	13400000002	1002	正式	开发部	2018-01-01		通用方案	定薪 查看

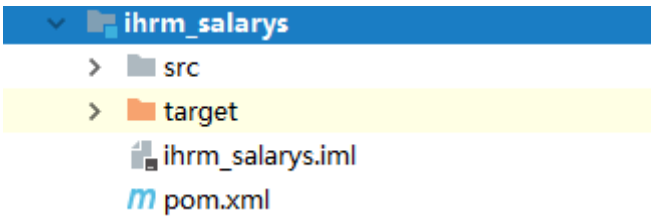
4.1 需求分析

4.2 数据库表

4.3 环境搭建

（1）创建模块

创建考勤模块 `ihrm_salarys`



（2）导入基本代码

导入资料中提供好的基本代码

（3）配置启动类

```
@SpringBootApplication(scanBasePackages = "com.ihrm")
@EntityScan(value = "com.ihrm.domain")
@EnableFeignClients
public class SalarysApplication {
    /**
     * 启动方法
     */
    public static void main(String[] args) {
        SpringApplication.run(SalarysApplication.class, args);
    }

    @Bean
    public IdWorker idWorker() {
        return new IdWorker();
    }
}
```

在网关模块 `ihrm_gate` 添加对新资模块的地址映射

```
    ihrm-salarys: #工资
      path: /salarys/** #配置请求URL的请求规则
      serviceId: ihrm-salarys #指定Eureka注册中心中的服务id
      strip-prefix: false
      sentiviteHeaders:
      customSensitiveHeaders: true
```

4.4 列表展示

4.4.1 企业设置

根据企业的实际情况，可以设置企业计薪方式，津贴的统计方式

```
/**
 * 获取企业计薪及津贴设置
 */
@RequestMapping(value = "/settings", method = RequestMethod.GET)
public Result getSettings() throws Exception {
    Settings settings = settingsService.findById(companyId);
    return new Result(ResultCode.SUCCESS, settings);
}

/**
 * 保存企业计薪及津贴设置
 */
@RequestMapping(value = "/settings", method = RequestMethod.POST)
public Result saveSettings(@RequestBody Settings settings) throws Exception {
    settings.setCompanyId(companyId);
    settingsService.save(settings);
    return new Result(ResultCode.SUCCESS);
}
```

4.4.2 员工薪资列表

（1）controller

```
@RequestMapping(value = "/list", method = RequestMethod.POST)
public Result list(@RequestBody Map map) {
    //1.获取请求参数,page,size
    Integer page = (Integer)map.get("page");
    Integer pageSize = (Integer)map.get("pageSize");
    //2.调用service查询
    PageResult pr = salaryService.findAll(page,pageSize,companyId);
    return new Result(ResultCode.SUCCESS, pr);
}
```

（2）service

```
pageSize));
    return new PageResult(page1.getTotalElements(),page1.getContent());
}
```

（3）dao

```
    @Query(nativeQuery = true,
            value="select bu.id,bu.username,bu.mobile,bu.work_number
workNumber," +
                    "bu.in_service_status inServiceStatus,bu.department_name
departmentName,bu.department_id departmentId,bu.time_of_entry timeOfEntry ," +
                    "bu.form_of_employment formOfEmployment
,sauss.current_basic_salary currentBasicSalary,sauss.current_post_wag
currentPostWage from bs_user bu LEFT JOIN sa_user_salary sauss ON
bu.id=sauss.user_id WHERE bu.company_id = ?1",
            countQuery = "select count(*) from bs_user bu LEFT JOIN
sa_user_salary sauss ON bu.id=sauss.user_id WHERE bu.company_id = ?1"
    )
    Page findPage(String companyId, PageRequest pageRequest);
```