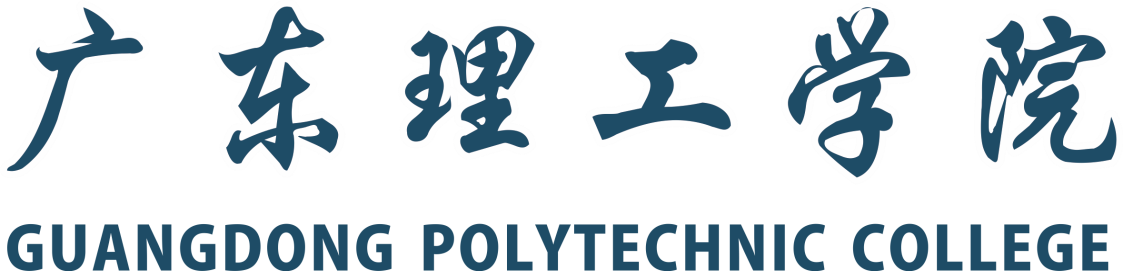
|  |  |
| --- | --- |
| **学号** | 1812402301044 |
| 1812402301010 |
| 1812402301003 |

****

**软件工程及应用**

**课程设计**

|  |  |
| --- | --- |
| **题 目** | **考勤管理系统** |
| **学院（系）** | **信息技术学院** |
| **专 业** | **软件工程** |
| **班 级** | **2018级1班** |
| **姓 名** | **叶锦亮** |
|  | **陈丰明** |
|  | **陈勇强** |
| **指导教师** | **李代平** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **2020** | **年** | **12** | **月** | **31** | **日** |

目录

[第1章 需求分析 1](#_Toc60687866)

[1.1 引言 1](#_Toc60687867)

[1.1.1项目背景 1](#_Toc60687868)

[1.1.2研究意义 1](#_Toc60687869)

[1.1.3 背景 1](#_Toc60687870)

[1.2软件总体概述 1](#_Toc60687871)

[1.3可行性分析 2](#_Toc60687872)

[1.3.1技术可行性 2](#_Toc60687873)

[1.4用户需求分析 3](#_Toc60687874)

[1.4.1功能分析 3](#_Toc60687875)

[1.4.2用例分析 5](#_Toc60687876)

[1.4系统设计目标分析 7](#_Toc60687877)

[2章 概要设计 8](#_Toc60687878)

[2.1引言 8](#_Toc60687879)

[2.2 总体结构设计 8](#_Toc60687880)

[2.2.1系统的流程分析 8](#_Toc60687881)

[2.2 数据库设计 10](#_Toc60687882)

[2.2.1数据流程 10](#_Toc60687883)

[2.2.2数据库需求 10](#_Toc60687884)

[2.2.3类的设计 10](#_Toc60687885)

[2.2.4 E-R模型 12](#_Toc60687886)

[第3章 详细设计 12](#_Toc60687887)

[3.1 引言 12](#_Toc60687888)

[3.2程序系统结构 12](#_Toc60687889)

[3.2.1持久化类的设计 13](#_Toc60687890)

[3.2.2数据库操作类设计 14](#_Toc60687891)

[3.2.3控制器类设计 15](#_Toc60687892)

[3.2.4数据字典 16](#_Toc60687893)

[第4章 软件实现 18](#_Toc60687894)

[4.1 界面 18](#_Toc60687895)

[4.2模块（或子程序）1（标识符）源码 20](#_Toc60687896)

[4.2.1员工打卡模块 20](#_Toc60687897)

[4.2.2管理员登录模块 22](#_Toc60687898)

[4.2.3修改密码模块 23](#_Toc60687899)

[4.2.4工作安排模块 25](#_Toc60687900)

[4.2.5月需工作时间设置模块 27](#_Toc60687901)

[4.2.4每日工作时间报表模块 29](#_Toc60687902)

[4.2.5每日工作时长报表模块 33](#_Toc60687903)

[4.2.6每月工作报表模块 35](#_Toc60687904)

[4.3测试用例 38](#_Toc60687905)

[第5章 系统分析与改进 39](#_Toc60687906)

[5.1测试结果与分析 39](#_Toc60687907)

[5.1.1员工打卡模块测试 39](#_Toc60687908)

[5.1.2管理员登录模块测试 40](#_Toc60687909)

[5.1.3工作安排模块测试 41](#_Toc60687910)

[5.1.4工作时长安排模块测试 42](#_Toc60687911)

[5.1.5统计报表模块测试 43](#_Toc60687912)

[5.1.6密码修改模块测试 44](#_Toc60687913)

[5.2可改进的部分 45](#_Toc60687914)

[第6章 心得体会 46](#_Toc60687915)

[参考文献 46](#_Toc60687916)

[附录 47](#_Toc60687917)

摘 要

随着时代的发展，信息化越来越普遍。这次设计主要面向企业员工的考勤管理，以此建立信息系统节省人力资源。本系统使用B/S架构实现，对员工考勤的管理对实际的考勤流程进行抽象、优化并加以实现。

考勤系统记录员工的实际上下班时间，根据实际设置的正常考勤时间对员工的考勤进行统计分析并的出相应报表。除此之外系统还需要面向管理人员的管理界面以供其完成可能需要的查找和修改操作。

本次课程设计将按照软件工程这门科目相应的设计方法和流程来完成设计，对项目进行可行性研究、需求分析、概要设计和详细设计等具体的设计步骤。

【关键词】信息化、考勤管理、B/S架构

**Abstract**

With the development of the times, information technology is more and more common. This design is mainly for the attendance management of enterprise employees, in order to establish an information system to save human resources. This system uses B / S architecture to realize the management of employee attendance, and abstracts, optimizes and realizes the actual attendance process.

The attendance system records the actual off-duty time of employees, makes statistical analysis on the attendance of employees according to the actual normal attendance time, and produces corresponding reports. In addition, the system also needs a management interface for managers to complete the search and modification operations.

This course design will complete the design according to the corresponding design method and process of software engineering, and carry out the feasibility study, demand analysis, outline design, detailed design and other specific design steps for the project.

【Key Words】informatization,attendance management,B / S architecture

# 第1章 需求分析

1.1 引言

主要分为系统可行性分析、用户需求分析;系统分析要求必须了解系统所实现的功能,这个设计的好坏直接影响的后面的开发。用户需求分析主要是了解用户的有哪些需求,本系统主要是用户的购餐需求以及管理的需求。只有了解了需求分析才能进行系统的开发,否则就会事倍功半,达不到我们预期的效果,因此需求分析是至关重要的。

### 1.1.1项目背景

时代在发展，科技在不停的进步，人民的生活水平也在不停的提高。那么在现在这个数字的时代,在计算机普及后的今天,中国进入了互联网信息时代,生活方式和工作节奏都随着信息传递速度的加快而变得急促。从信件传递发展成 Email,从传统交流到IP网络,一切原本属于现实生活中的行为方式慢慢地向网络转移。传统的购物方式也正在向网络方面发展,出现了“网购”这种越来越受欢迎的新型购物方式。同样的其他行业也不可能避免的不断向信息化靠拢。

### 1.1.2研究意义

中国许多公司企业的考勤管理都还是人工管理的，以人工记录然后再以计算机表格文档的形式进行存储，并没有一套完备的信息化的考勤管理体制。以往的人力考勤管理，耗费考勤管理人员的人力资源而且由于是人工管理有事由于人为失误原因可能出现考勤错漏的情况，对于这样的情况非常有必要建立一套完备的信息化考勤管理系统解决以往耗费人物力、准确性不强的难题。

### 1.1.3 背景

项目名称：考勤管理系统。

人员：叶锦亮、陈丰明、陈勇强。

1.2软件总体概述

软件设计是把需求转化为软件系统的最重要的环节，系统设计的优劣在根本上决定了软件系统的质量。

在此，主要阐述软件系统设计的5个核心内容：体系结构设计、用户界面设计、数据库设计、模块设计、数据结构和算法设计。旨在帮助开发人员搞清楚“设计什么”以及“如何设计”。

一般把设计过程划分为两个阶段：概要设计阶段和详细设计阶段，如下所示：

概要设计阶段的重点是体系结构设计。

详细设计阶段的重点是用户界面设计、数据库设计、模块设计、数据结构与算法设计等。

1.3可行性分析

可行性,是指在当前情况下,设计开发这个系统是否有必要,是否具备必要的条件,它不仅包括必要性,还包括合理性。系统的可行性分析,是针对要设计的系统,研究其设计是否存在外部的必要条件,包括技术上、资金上、经济上、人力资源以及组织管理上的必要条件,同时要分析在目前条件下是否有必要设计这样一个系统。

### 1.3.1技术可行性

1.硬件、软件资源

硬件资源的要求不高，但是开发计算机要求内存要有4GB或以上才能满足系统开发的需要。

系统开发使用到的软件有InteliJ IDEA 2019.2、亿图图示。数据库使用内嵌式的H2database 1.4.2。

2.技术支持

1. JDBC

Java数据库连接，（Java Database Connectivity，简称JDBC）是Java语言中用来规范客户端程序如何来访问数据库的应用程序接口，提供了诸如查询和更新数据库中数据的方法。JDBC也是Sun Microsystems的商标。我们通常说的JDBC是面向关系型数据库的。

1. Spring MVC

Spring MVC属于SpringFrameWork的后续产品，已经融合在Spring Web Flow里面。Spring 框架提供了构建 Web 应用程序的全功能 MVC 模块。使用 Spring 可插入的 MVC 架构，从而在使用Spring进行WEB开发时，可以选择使用Spring的Spring MVC框架或集成其他MVC开发框架，如Struts1(现在一般不用)，Struts 2(一般老项目使用)等等。

1. Spring Boot

SpringBoot是由Pivotal团队在2013年开始研发、2014年4月发布第一个版本的全新开源的轻量级框架。它基于Spring4.0设计，不仅继承了Spring框架原有的优秀特性，而且还通过简化配置来进一步简化了Spring应用的整个搭建和开发过程。另外SpringBoot通过集成大量的框架使得依赖包的版本冲突，以及引用的不稳定性等问题得到了很好的解决。

1. H2database

H2数据库是一个开源的关系型数据库。H2是一个嵌入式数据库引擎，采用java语言编写，不受平台的限制，同时支持网络版和嵌入式版本，有比较好的兼容性，支持相当标准的sql标准，支持集群。提供JDBC、ODBC访问接口，提供了非常友好的基于web的数据库管理界面。

由于本系统是使用Java语言进行开发的，使用以上技术完全能够胜任本系统的开发。

1.4用户需求分析

### 1.4.1功能分析

用户以往记录考勤的流程是由人力操作的方式确认员工的上下班时间，以纸质表格的形式实时记录，最后再归档到电子档表格中。人工手段进行记录统计不可避免的可能出现细节上的疏漏，并且过于浪费人力资源。所以使用软件管理的考勤管理信息系统。公司员工在考勤打卡页面打上班卡或下班卡，每日或每月对考勤数据进行统计分析。并的出相应的考勤数据报表。管理人员使用账户密码登录后台管理系统对员工信息以及考勤信息进行管理。其中企业员工的工作制度又分为弹性工作制和固定工作制，其中弹性工作制主要计算每个月的工作时长，固定工作制主要计算上下班迟到早退记录和上班天数。

功能包括如下几部分：

1. 员工打卡

实现员工上下班离开公司的打卡登记。

1. 修改密码

实现系统管理员对管理员账户的密码修改。

1. 每月工作天数统计

实现对固定工作制员工的工作天数的统计。

1. 每日工作时间统计

实现对弹性工作制员工当天的工作时间统计，并统计当月工时。

1. 每日迟到早退次数统计

实现对每天员工的迟到早退次数统计。

1. 每月工时统计

实现对弹性工作制员工的每月的工时统计。

1. 工作时间安排

系统管理员给每一位员工设置每个月的工作安排，对于固定工作制的员工设置上下班时间和早退迟到条件。对于弹性工作制员工设置其每个月需要满足的工作时长。

1. 考勤管理

给系统管理员提供对员工考勤的增删改查操作功能。

通过实地调查对考勤的了解，以及对项目的需求分析基本就可以了解考情系统的基本功能和操作流程。通过这些信息可以画出功能结构图，以助于更形象的了解本系统实现的基本功能。

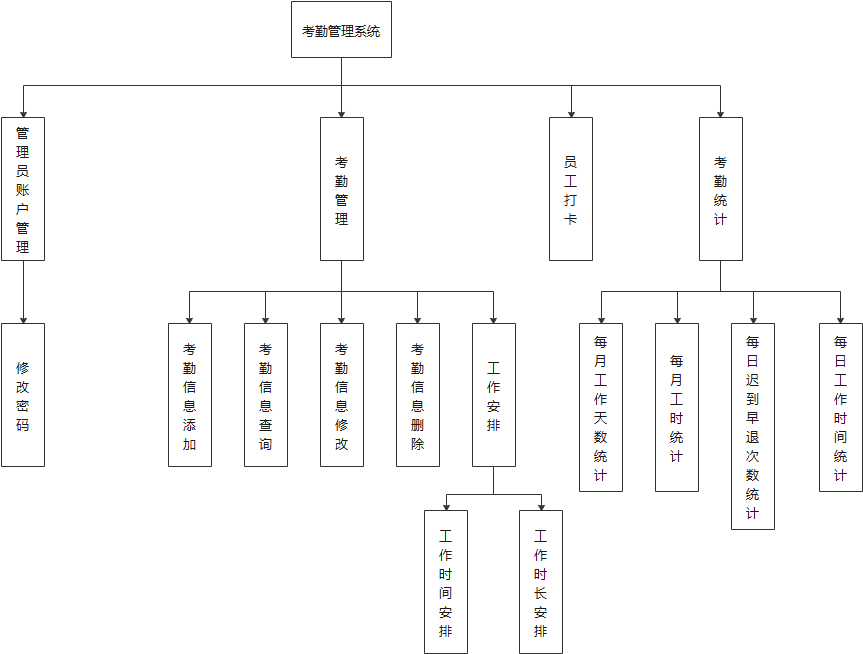


图1-1功能结构图

### 1.4.2用例分析

用例图是被称为参与者的外部用户所能观察到的系统功能的模型图,呈现了一些参与者和一些用例,以及它们之间的关系,主要用于对系统、子系统或类的功能行为进行建模,用例图展示了用例之间以及同用例参与者之间是怎样相互联系的。本系统主要的角色(Actor)包括数据库、管理员、游客、会员;

系统具体用例如下图所示:

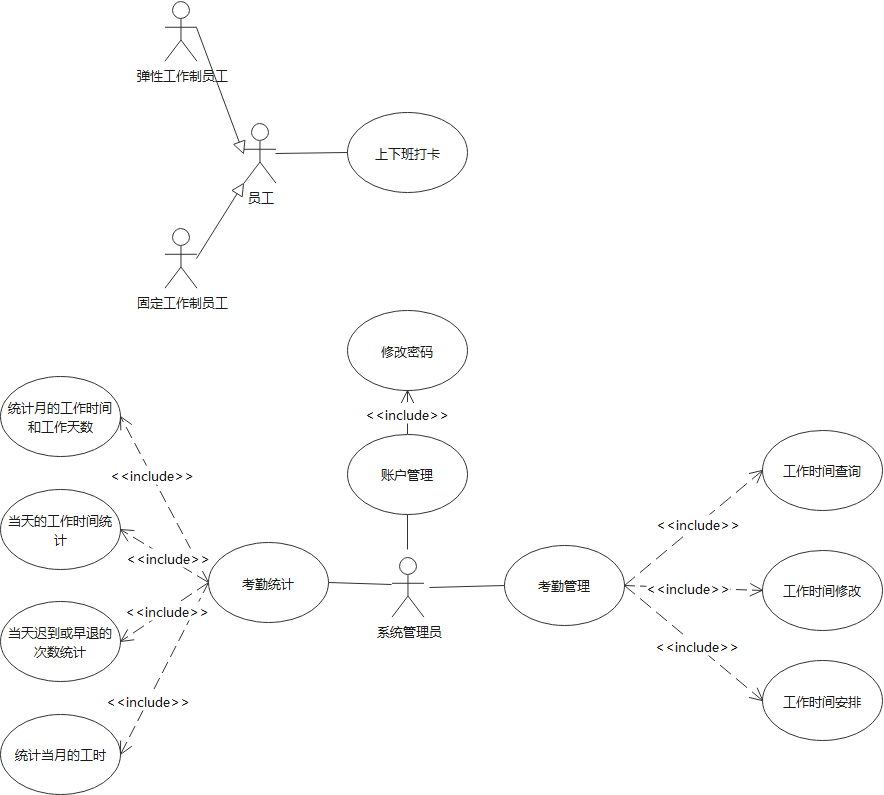


图1-2用例图

下面介绍软件中的主要用例：

表1-1上下班打卡用例描述

|  |  |
| --- | --- |
| 用例名称： | 上下班打卡 |
| 参与者： | 员工 |
| 前置条件： | 员工进入到打卡页面 |
| 用例功能： | 记录员工上下班离开公司时间 |
| 事件流： | 员工输入工号与打卡类型后保存信息到数据库 |
| 异常事件流： | 工号不存在 |
| 后置条件： | 系统无改变 |

表1-2修改密码用例描述

|  |  |
| --- | --- |
| 用例名称： | 修改密码 |
| 参与者： | 系统管理员 |
| 前置条件： | 系统管理员登录到后台管理界面 |
| 用例功能： | 供系统管理员修改个人账号密码 |
| 事件流： | 系统管理员成功修改密码后返回到登录界面 |
| 异常事件流： | 输入的原密码不正确或新密码不符合格式 |
| 后置条件： | 系统无改变 |

表1-3考勤统计用例描述

|  |  |
| --- | --- |
| 用例名称： | 考勤统计 |
| 参与者： | 系统管理员 |
| 前置条件： | 系统管理员登录到后台管理界面 |
| 用例功能： | 供系统管理员查询员工考勤的报表信息 |
| 事件流： | 输入查询条件后返回报表信息 |
| 异常事件流： | 输入查询条件有误 |
| 后置条件： | 系统无改变 |

表1-4考勤管理用例描述

|  |  |
| --- | --- |
| 用例名称： | 考勤管理 |
| 参与者： | 系统管理员 |
| 前置条件： | 系统管理员登录到后台管理界面 |
| 用例功能： | 供系统管理员对考勤信息查询和改动以及安排员工工作 |
| 事件流： | 输入查询条件或修改信息 |
| 异常事件流： | 输入错误有误或不符合格式 |
| 后置条件： | 系统无改变 |

1.4系统设计目标分析

开发的软件是供企业员工进行信息化考勤管理的系统，以企业员工以及考勤管理人员为服务对象。提供友善的操作界面，实现员工便捷打卡，考勤管理人员高效管理为目标。该系统能够实现员工打卡、报表生成、考勤信息修改等功能。初步预想是使用SpringBoot整合Spring+Spring MVC两大框架及H2datebase数据库这几项技术来完成考勤管理系统。

系统主要提供的功能如下：

1. 员工通过输入工号和打卡类型就能便捷的完成考勤。
2. 系统管理员能够通过系统安排员工的上班/放假日期或安排每月必须满足的工作时长。
3. 系统管理员能够通过系统对员工的考勤信息进行增删改查操作。
4. 系统能够提供多种类型的考勤报表。

# 2章 概要设计

2.1引言

系统的概要设计是在系统分析的基础上由抽象到具体的过程。主要包括系统设计、数据库的需求分析以及的整体设计三大块。系统设计主要目标是将系统分析阶段所提出的反映了信息需求的系统逻辑方案转换成可以实施的基于计算机运行环境的具体方案,进行系统的概要设计,为下一阶段系统详细设计提供必要的技术资料。

2.2 总体结构设计

### 2.2.1系统的流程分析

对于系统的业务流程的走向进行系统的分析,从宏观的角度,时时以一名用户的视角去设计项目的业务流程,使项目更加的人性化,自然化,更加符合用户对于订餐的一些基本要求,在分析过程中,始终坚持以用户为核心,以设计用户满意的为目标,坚持用户的体验满意度至上的原则。

在系统使用的过程中用户最先看到的就是系统的主界面，系统主要面向两种类型的用户，所以主界面提供索引功能以供两种类型用户选择其需要的界面。

首先，索引指向的界面之一是员工上下班打卡界面，其满足的是记录员工上下班的需求。在这个界面里主要有显示当前时间的控件、员工输入工号以及选择打卡类型的表单这两项组成。员工在表单中输入了工号和打卡类型选择提交以后，这两项数据传输到系统，系统首先对这两项输入进行校验，如果校验发现输入的工号不存在则返回错误到上线打卡界面显示并让员工重新输入，否则打卡记录将保存到系统数据库。

再一个指向的界面是供系统管理员使用的后台管理界面。在用户点击索引后转到后台管理界面前系统先要判断用户的登录状态，如果用户当前未登录则跳转到管理员登录界面进行登录操作。登录操作需要得到管理员账号密码这两项输入，根据这两项信息输入校验的结果决定转至重新输入状态或已登录状态；用户完成登录以后可以进行修改密码、考勤统计和考勤管理这三项操作。其中考勤统计和考勤管理的流程都为输入查询条件或修改内容后得到报表或操作结果。修改密码的流程则类同登录。

以下是业务流程活动图：

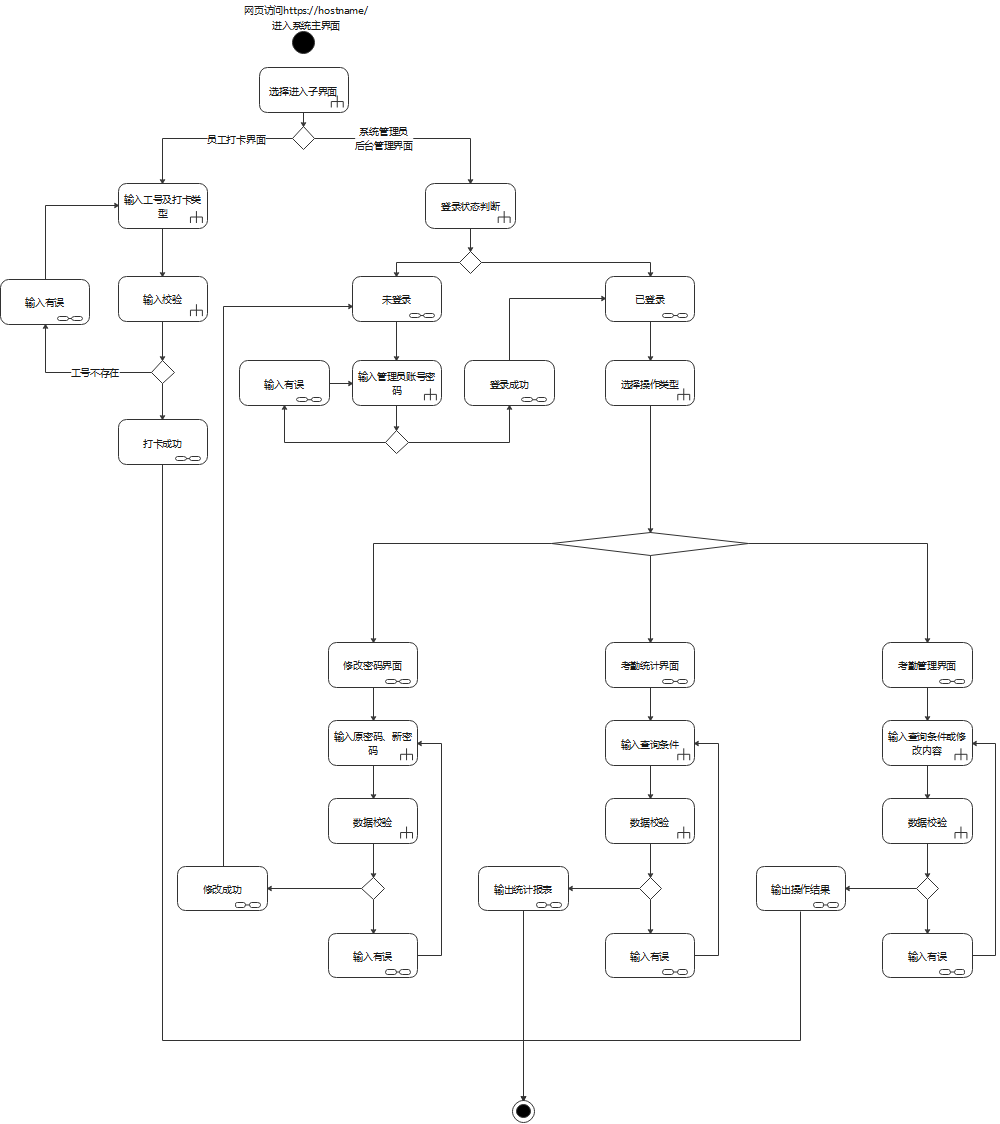


图3-1业务流程活动图

2.2 数据库设计

### 2.2.1数据流程

本系统涉及的数据较多,因此对于系统的设计首先要了解系统中数据的流向,只有清楚数据之间的关系,才能正确的设计系统的功能。

在本系统中运行过程中将会产生非常多的数据，比如用户打卡产生的打卡记录、管理员的账号密码、各种考勤报表等等。都是需要持久化的数据，需要存储在数据库中。

### 2.2.2数据库需求

可以知道在本系统中流动的数据基本都是要持久化的数据，而持久化就要使用到数据库来完成，但是我们系统面向的用户相对需要相对轻量化的应用程序，所以本系统选择的数据库为H2database，H2数据库非常快，开源，支持JDBC API嵌入式和服务器模式，是一种内存数据库，控制台是基于浏览器的控制台应用程序且占用空间小，jar只有2MB大小，刚好满足我们系统设计的需求。

### 2.2.3类的设计

对系统进行类的设计得到以下结果：

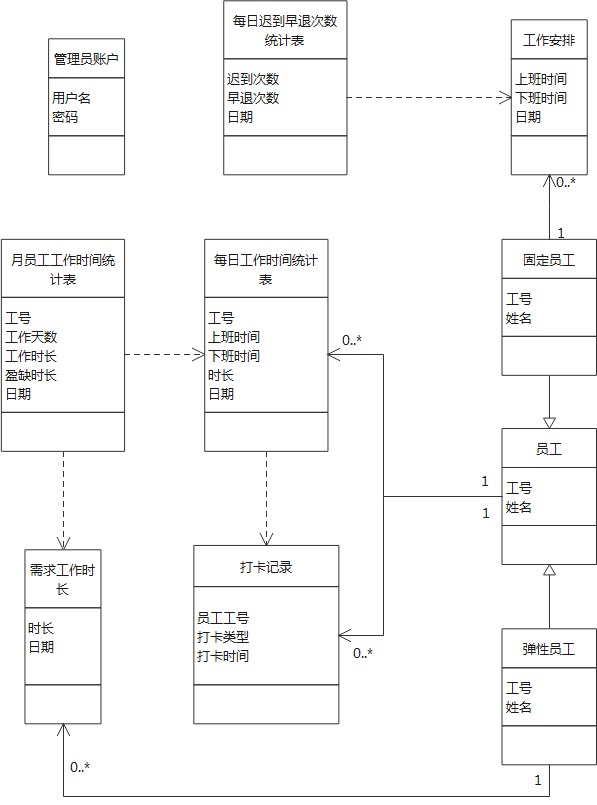


图2-2类图

数据库表的描述：

表2-1数据库表描述

|  |  |  |
| --- | --- | --- |
| 编号 | 表名称 | 描述 |
| 1 | Emplyee | 员工表 |
| 2 | Manager | 系统管理员表 |
| 3 | ClockRecord | 打卡记录表 |
| 4 | WorkArrange | 工作安排表 |
| 5 | WorkLengthRequired | 需求工作时长表 |
| 6 | WorkStatistic\_Day | 每日工作统计表 |
| 7 | WorkStatistic\_Day | 每日工时统计表 |
| 8 | WorkStatistic\_Month | 每月工作统计表 |

### 2.2.4 E-R模型

E-R模型是一种叫做实体联系模型的简称,它是以图形的方式表现的,又被称作实体联系图,ER模型是人们描述数据及其联系的概念数据模型,是数据库应用系统设计人员和普通非计算机专业用户进行数据建模和沟通与交流的有力工具,使用起来直观易懂,简单易行,ER模型是一种用图形表示数据及其联系的方法,所使用的图形构件(元件)包括矩形、菱形、椭圆形和连接线。

主要的ER模型图如下:

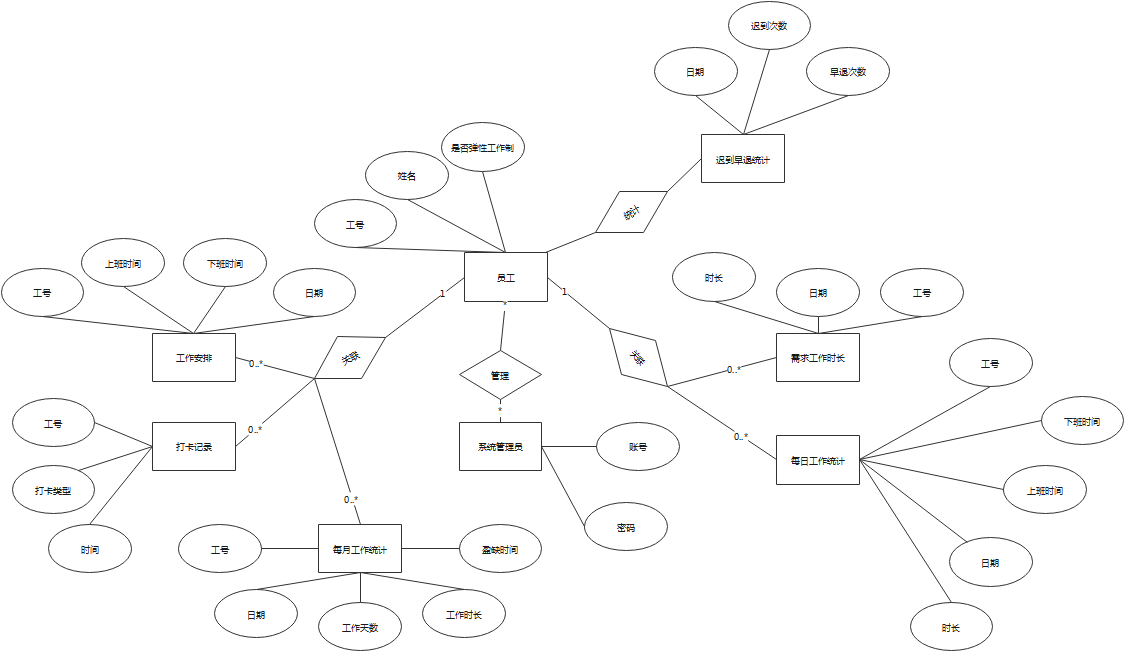


图2-3 E-R模型图

# 第3章 详细设计

3.1 引言

根据前面的分析和用户的需求，本考勤管理系统主要有员工打卡和后台管理两部分组成，员工打卡主要是记录员工上线时间，后台管理提供考勤管理操作。

3.2程序系统结构

### 3.2.1持久化类的设计

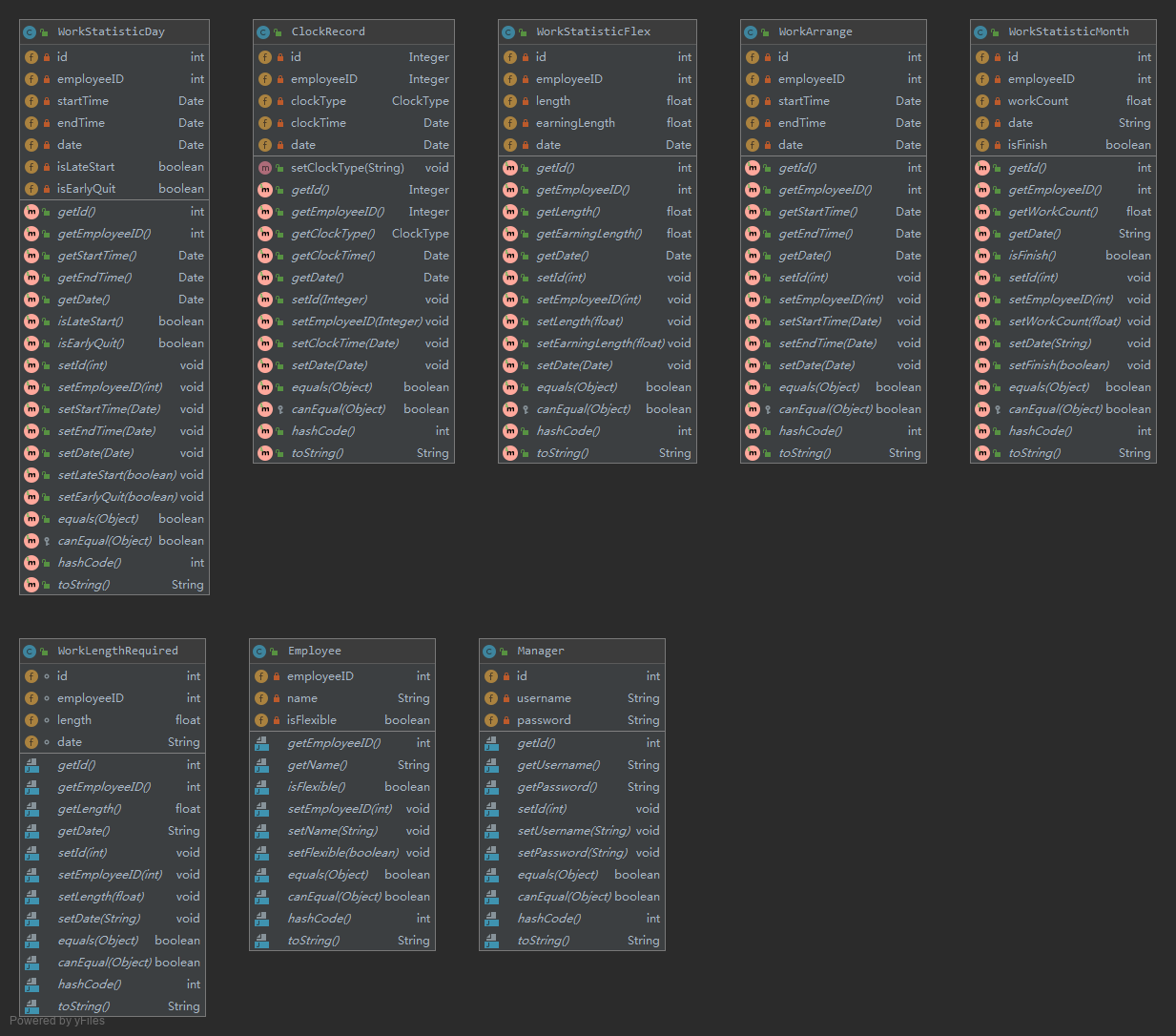


图3-1持久化类类图

持久化类描述：

表3-1持久化类描述表

|  |  |  |
| --- | --- | --- |
| 编号 | 类名 | 描述 |
| 1 | ClockRecord | 打卡记录类 |
| 2 | Employee | 员工类 |
| 3 | Manager | 系统管理员账户类 |
| 4 | WorkArrange | 工作安排类 |
| 5 | WorkLengthRequired | 月需工作时长类 |
| 6 | WorkStatisticDay | 每日工作时间报表类 |
| 7 | WorkStatisticFlex | 每日工作时长报表类 |
| 8 | WorkStatisticMonth | 每月工作时间报表类 |

### 3.2.2数据库操作类设计

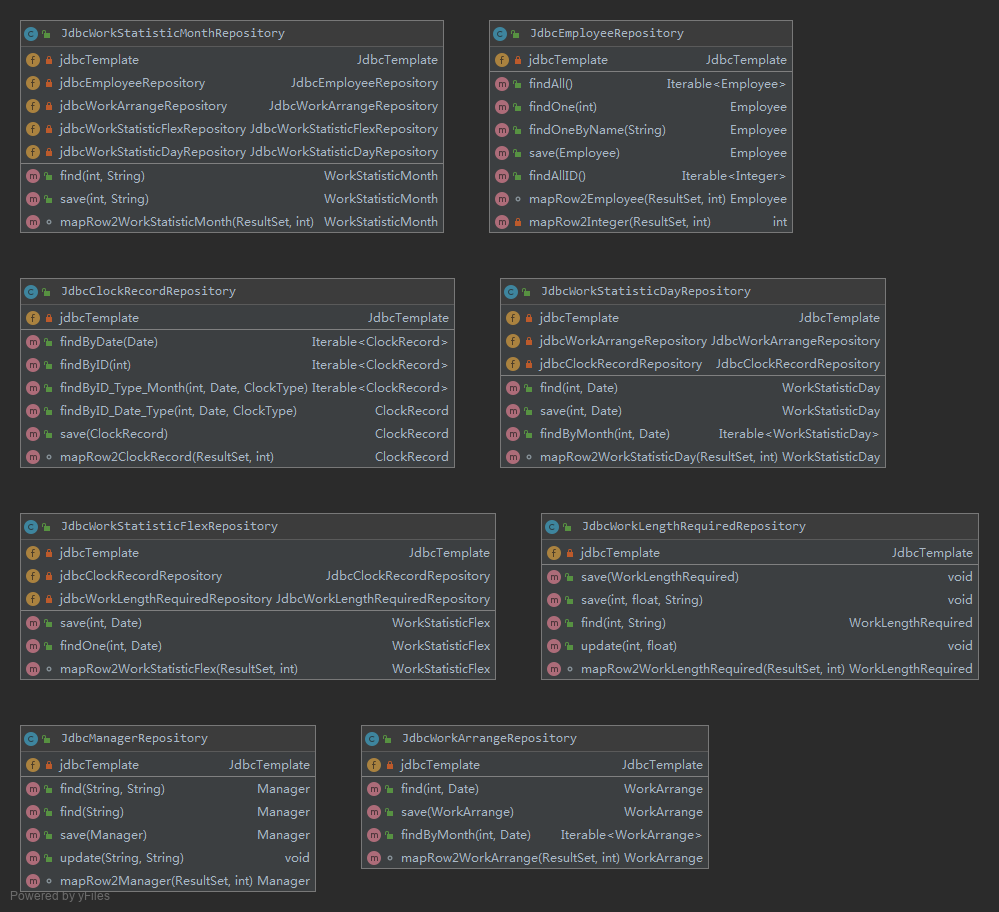


图3-2数据库操作类类图

数据库操作类描述：

表3-2数据库操作类描述表

|  |  |  |
| --- | --- | --- |
| 编号 | 类名 | 描述 |
| 1 | JdbcClockRecordRepository | 打卡记录Dao操作实现类 |
| 2 | JdbcEmployeeRepository | 员工Dao操作实现类 |
| 3 | JdbcManagerRepository | 管理员Dao操作实现类 |
| 4 | JdbcWorkArrangeRepository | 工作安排Dao操作实现类 |
| 5 | JdbcWorkLengthRequiredRepository | 月需工作时长Dao操作实现类 |
| 6 | JdbcWorkStatisticDayRepository | 每日工作时间报表Dao操作实现类 |
| 7 | JdbcWorkStatisticFlexRepository | 每日工作时长报表Dao操作实现类 |
| 8 | JdbcWorkStatisticMonthRepository | 每月工作时间报表Dao操作实现类 |

### 3.2.3控制器类设计

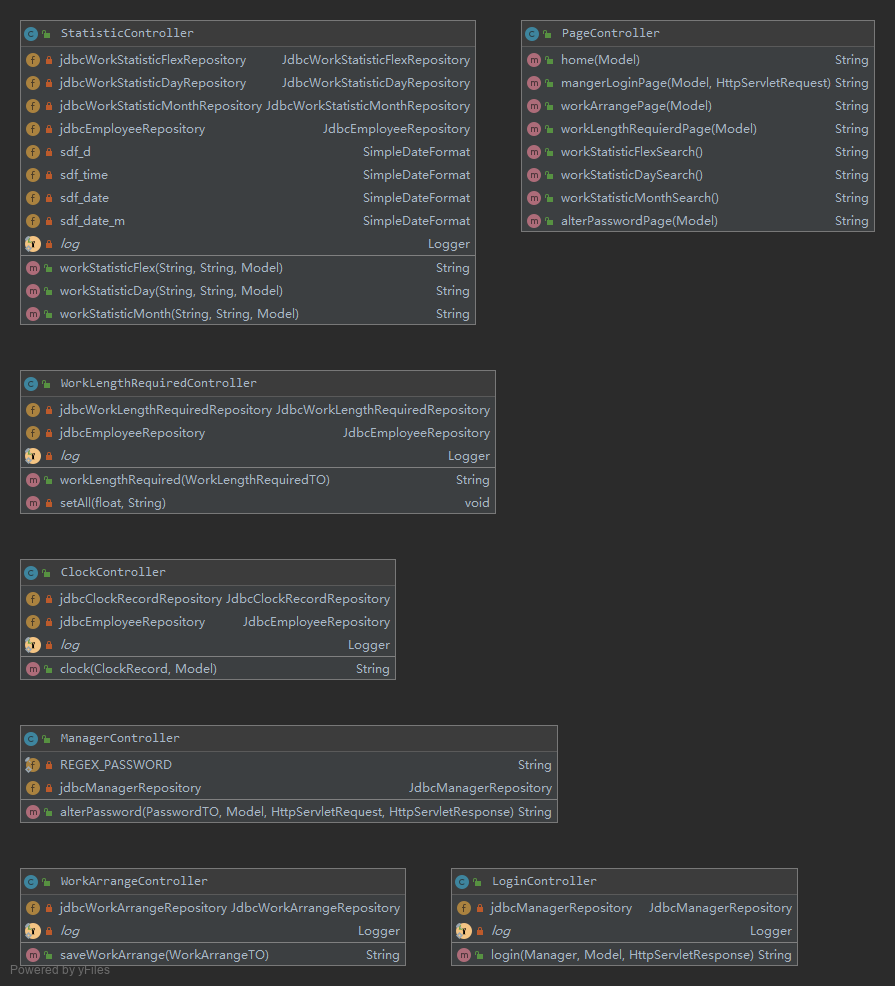


图3-3控制器类类图

控制器类描述：

表3-3 控制器类描述表

|  |  |  |
| --- | --- | --- |
| 编号 | 类名 | 描述 |
| 1 | ClockRecordControler | 打卡操作控制器类，接受打卡请求控制打卡操作流程。 |
| 2 | LoginController | 登录操作控制器类，主要处理管理员登录后台管理页面请求。 |
| 3 | ManagerControler | 管理员账户控制器类，主要处理管理员账户修改相关的请求。 |
| 4 | PageController | 页面控制器类，主要处理客户端发来的页面请求，返回用户需要的界面。 |
| 5 | StatisticController | 报表控制器类，处理查询报表相关的请求。 |
| 6 | WorkArrangeController | 工作安排控制器类，处理管理员安排员工工作时间的请求。 |
| 7 | WorkLengthRequiredContrller | 需工作时长控制器类，完成设置弹性工作制员工每月所需工作时长操作。 |

### 3.2.4数据字典

员工表：

表3-4员工表描述

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| name | VARCHAR(20) | 否 | 否 |
| isFlexible | BOOL | 否 | 否 |

打卡记录表：

表3-5打卡记录表描述

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| employeeID | INT | 否 | 否 |
| clockType | VARCHAR(3) | 否 | 否 |
| clockTime | dateTime | 否 | 否 |
| date | DATE | 否 | 否 |

管理员账户表：

表3-6管理员账户表描述

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| username | VARCHAR(20) | 否 | 否 |
| password | VARCHAR(20) | 否 | 否 |

工作安排表：

表3-7工作安排表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| employeeID | INT | 否 | 否 |
| startTime | TIME | 否 | 否 |
| endTime | TIME | 否 | 否 |
| date | DATE | 否 | 否 |

需工作时长表：

表3-8需工作时长表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| employeeID | INT | 否 | 否 |
| length | FLOAT | 否 | 否 |
| date | DATE | 否 | 否 |

每日工作时长报表:

表3-9每日工作时长报表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| employeeID | INT | 否 | 否 |
| length | FLAOT | 否 | 否 |
| date | DATE | 否 | 否 |
| earningLength | FLAOT | 否 | 否 |

每日工作时间表：

表3-10每日工作时间表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| employeeID | INT | 否 | 否 |
| startTime | TIME | 否 | 否 |
| endTime | TIME | 否 | 否 |
| date | DATE | 否 | 否 |
| isLateStart | BOOL | 否 | 否 |
| isEarlyQuit | BOOL | 否 | 否 |

每月工作时间表：

表3-11每月工作时间表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| employeeID | INT | 否 | 否 |
| workCount | FLOAT | 否 | 否 |
| isFinish | BOOL | 否 | 否 |

# 第4章 软件实现

4.1 界面

本系统使用B/S架构开发，服务端主要在java环境下运行，下图是在cmd命令行运行的界面。余下的为面向用户的界面。

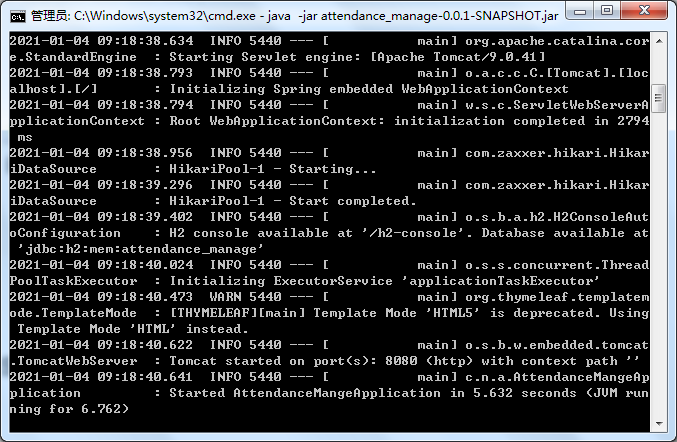


图4-1服务端运行界面

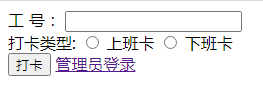


图4-2主界面

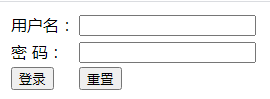


图4-3管理员登录界面



图4-4后台管理索引界面

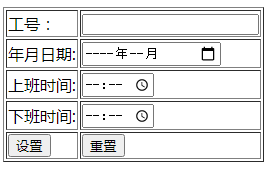


图4-5工作安排界面

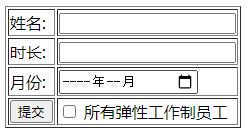


图4-6需工作时长设置界面

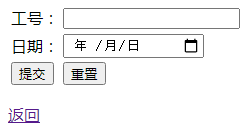


图4-7每日报表及月报表查询界面



图4-8修改密码界面

4.2模块（或子程序）1（标识符）源码

### 4.2.1员工打卡模块

表4-1模块涉及文件表

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 名称 | 类型 | 描述 |
| 1 | ClockRecord.java | Java源代码文件 | 打卡记录类 |
| 2 | JdbcClockRecordRepository.java | Java源代码文件 | 打卡记录Dao类 |
| 3 | PageController.java | Java源代码文件 | 页面控制器类 |
| 4 | Home.html | Html文件 | 打卡页面 |
| 5 | Message.html | Html文件 | 打卡结果页面 |

主要代码描述:

//ClockRecord Dao实现：  
public ClockRecord save(ClockRecord clockRecord) {  
 clockRecord.setClockTime(new Date());  
 jdbcTemplate.update(  
 "insert into clockRecord(employeeID,clockType,clockTime,date) values(?,?,?,?)",  
 clockRecord.getEmployeeID(),  
 clockRecord.getClockType().name(),  
 new Timestamp(clockRecord.getClockTime().getTime()),  
 new java.sql.Date(clockRecord.getClockTime().getTime()));  
 return clockRecord;  
}

//控制器实现：  
@PostMapping  
public String clock(ClockRecord clockRecord, Model model){  
 clockRecord = jdbcClockRecordRepository.save(clockRecord);  
 model.addAttribute("time",new Date(clockRecord.getClockTime().getTime()));  
 model.addAttribute("name",jdbcEmployeeRepository.findOne(clockRecord.getEmployeeID()).getName());  
 model.addAttribute("clockType",clockRecord.getClockType().name().equals("IN")?"上班卡":"下班卡");  
 return "message";  
}

### 4.2.2管理员登录模块

表4-1模块涉及文件表

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 名称 | 类型 | 描述 |
| 1 | Manager.java | Java源代码文件 | 管理员实体类 |
| 2 | JdbcManagerRepository.java | Java源代码文件 | Dao类 |
| 3 | LoginController.java | Java源代码文件 | 登录控制器类 |
| 4 | managerLoginPage.html | Html文件 | 登录页面 |
| 5 | Message.html | Html文件 | 打卡结果页面 |

主要代码描述：

//Dao实现：

public Manager find(String username) {  
 try{  
 return jdbcTemplate.queryForObject("select \* from Manager where username=?",this::mapRow2Manager,username);  
 }catch (Exception e){  
 return null;  
 }  
}

//控制器实现：

@PostMapping("/login")

public String login(Manager manager, Model model, HttpServletResponse response){

if(jdbcManagerRepository.find(manager.getUsername())==null){

model.addAttribute("msg\_username","用户名不存在");

return "managerLoginPage";

}else if(jdbcManagerRepository.find(manager.getUsername(),manager.getPassword())==null){

model.addAttribute("msg\_password","密码错误");

return "managerLoginPage";

}

Cookie cookie = new Cookie("u\_token",manager.getUsername());

cookie.setMaxAge(3600);

response.addCookie(cookie);

return "manageIndex";

}

### 4.2.3修改密码模块

//Dao实现：  
public void update(String username, String newPassword) {  
 jdbcTemplate.update("update Manager set password=? where username=?",newPassword,username);  
}

//控制器实现：

@PostMapping("manage/alterPassword")  
public String alterPassword(PasswordTO passwordTO, Model model, HttpServletRequest httpServletRequest,HttpServletResponse httpServletResponse){  
 if(!passwordTO.getNewPassword().equals(passwordTO.getCheckPassword())){  
 model.addAttribute("err\_msg","两次输入不一致");  
 model.addAttribute("redirect\_url","3;url=/manage/alterPassword");  
 return "err";  
 }  
 if(!Pattern.*matches*(*REGEX\_PASSWORD*,passwordTO.getNewPassword())){  
 model.addAttribute("err\_msg","密码格式应为6-20个字符，至少1个字母，1个数字和1个特殊字符");  
 model.addAttribute("redirect\_url","3;url=/manage/alterPassword");  
 return "err";  
 }  
 Cookie[] cookies = httpServletRequest.getCookies();  
 String username = null;  
 for (Cookie cookie : cookies)  
 if (cookie.getName().equals("u\_token")){  
 username = cookie.getValue();  
 break;  
 }  
  
 if(username==null){  
 model.addAttribute("err\_msg","用户未登录");  
 model.addAttribute("redirect\_url","3;url=/manage");  
 return "err";  
 }  
  
 if(jdbcManagerRepository.find(username,passwordTO.getOldPassword())==null){  
 model.addAttribute("err\_msg","原密码错误");  
 model.addAttribute("redirect\_url","3;url=/manage/alterPassword");  
 return "err";  
 }  
  
 jdbcManagerRepository.update(username,passwordTO.getNewPassword());  
 Cookie cookie = new Cookie("u\_token",username);  
 cookie.setMaxAge(0);  
 httpServletResponse.addCookie(cookie);  
 return "redirect:/";  
}

### 4.2.4工作安排模块

//Dao实现

@Override  
public WorkArrange find(int employeeID,Date date) {  
 try {  
 return jdbcTemplate.queryForObject(  
 "select \* from WorkArrange where date=? and employeeID=?",  
 this::mapRow2WorkArrange,  
 new java.sql.Date(date.getTime()),employeeID  
 );  
 }catch(Exception e){  
 return null;  
 }  
}  
  
@Override  
public WorkArrange save(WorkArrange workArrange) {  
 if(find(workArrange.getEmployeeID(),workArrange.getDate())!=null){  
 jdbcTemplate.update(  
 "update WorkArrange set startTime=?,endTime=? where date=? and employeeID = ?",  
 new Time(workArrange.getStartTime().getTime()),  
 new Time(workArrange.getEndTime().getTime()),  
 new Date(workArrange.getDate().getTime()),  
 workArrange.getEmployeeID()  
 );  
 }else {  
 jdbcTemplate.update(  
 "insert into WorkArrange(startTime,endTime,date,employeeID) values (?,?,?,?)",  
 new Time(workArrange.getStartTime().getTime()),  
 new Time(workArrange.getEndTime().getTime()),  
 new java.sql.Date(workArrange.getDate().getTime()),  
 workArrange.getEmployeeID()  
 );  
 }  
 return workArrange;  
}

//控制器实现

@PostMapping("/saveWorkArrange")  
public String saveWorkArrange(WorkArrangeTO workArrangeTO){  
 *log*.info(workArrangeTO.toString());  
 for(Date date:workArrangeTO.getDates()){  
 jdbcWorkArrangeRepository.save(  
 new WorkArrange(-1,  
 workArrangeTO.getEmployeeID(),  
 workArrangeTO.getStartTime(),  
 workArrangeTO.getEndTime(),  
 date));  
 }  
 return "redirect:/manage/workArrange";  
}

### 4.2.5月需工作时间设置模块

//Dao实现

@Override  
public void save(WorkLengthRequired workLengthRequired) {  
 WorkLengthRequired obj = find(workLengthRequired.getEmployeeID(), workLengthRequired.getDate());  
 if(obj!=null && obj.getLength()!=workLengthRequired.getLength()){  
 jdbcTemplate.update("update WorkLengthRequired set length=? where id=?",obj.getId());  
 }else{  
 jdbcTemplate.update(  
 "insert into WorkLengthRequired(employeeID,length,date) values(?,?,?)",  
 workLengthRequired.getEmployeeID(),  
 workLengthRequired.getLength(),  
 workLengthRequired.getDate()  
 );  
 }  
}  
  
@Override  
public void save(int id, float length, String date) {  
 jdbcTemplate.update("insert into WorkLengthRequired(employeeID,length,date) values(?,?,?)",  
 id,length,date);  
}  
  
  
@Override  
public WorkLengthRequired find(int employeeID,String date) {  
 try{  
 return jdbcTemplate.queryForObject(  
 "select \* from WorkLengthRequired where employeeID=? and date=?",  
 this::mapRow2WorkLengthRequired,  
 employeeID,date  
 );  
 }catch (Exception e){  
 return null;  
 }  
}  
  
@Override  
public void update(int id, float length) {  
 jdbcTemplate.update("update WorkLengthRequired set length=? where id=?",length,id);  
}  
  
WorkLengthRequired mapRow2WorkLengthRequired(ResultSet resultSet,int rowNum) throws SQLException {  
 return new WorkLengthRequired(  
 resultSet.getInt("id"),  
 resultSet.getInt("employeeID"),  
 resultSet.getFloat("length"),  
 resultSet.getString("date")  
 );

//控制器实现

@PostMapping("manage/workLengthRequired")  
public String workLengthRequired(WorkLengthRequiredTO workLengthRequiredTO){  
 if(workLengthRequiredTO.isSetAll()){  
 setAll(workLengthRequiredTO.getLength(),workLengthRequiredTO.getDate());  
 }else{  
 jdbcWorkLengthRequiredRepository.save(  
 jdbcEmployeeRepository.findOneByName(workLengthRequiredTO.getName()).getEmployeeID(),  
 workLengthRequiredTO.getLength(),  
 workLengthRequiredTO.getDate()  
 );  
 }  
 return "workLengthRequired";  
}  
  
private void setAll(float length, String date){  
 Iterable<Integer> ids = jdbcEmployeeRepository.findAllID();  
 for(int id:ids){  
 WorkLengthRequired obj = jdbcWorkLengthRequiredRepository.find(id,date);  
 if(obj!=null && obj.getLength()!=length)  
 jdbcWorkLengthRequiredRepository.update(obj.getId(),length);  
 else  
 jdbcWorkLengthRequiredRepository.save(id,length,date);  
 }  
}

### 4.2.4每日工作时间报表模块

//Dao实现

@Override  
public WorkStatisticDay find(int employeeID, Date date) {  
 WorkStatisticDay workStatisticDay;  
 try {  
 save(employeeID, date);  
 workStatisticDay = jdbcTemplate.queryForObject(  
 "select \* from WorkStatistic\_Day where employeeID=? and date=?",  
 this::mapRow2WorkStatisticDay,  
 employeeID, new java.sql.Date(date.getTime()));  
 return workStatisticDay;  
 } catch (Exception e2) {  
 return null;  
 }  
  
}  
  
@Override  
public WorkStatisticDay save(int employeeID, Date date) {  
 WorkArrange workArrange = jdbcWorkArrangeRepository.find(employeeID, date);  
 if (workArrange == null)  
 return null;  
 ClockRecord start = jdbcClockRecordRepository.findByID\_Date\_Type(employeeID, date, ClockType.*IN*);  
 ClockRecord end = jdbcClockRecordRepository.findByID\_Date\_Type(employeeID, date, ClockType.*OUT*);  
 if (start == null || end == null)  
 return null;  
 WorkStatisticDay workStatisticDay = new WorkStatisticDay(-1,  
 employeeID,  
 start.getClockTime(),  
 end.getClockTime(),  
 date,  
 start.getClockTime().getHours() > workArrange.getStartTime().getHours() || (start.getClockTime().getMinutes() > workArrange.getStartTime().getMinutes()),  
 end.getClockTime().getHours() < workArrange.getEndTime().getHours() || (end.getClockTime().getMinutes() < workArrange.getEndTime().getMinutes()));  
 jdbcTemplate.update(  
 "insert into WorkStatistic\_Day(employeeID,startTime,endTime,date,isLateStart,isEarlyQuit) values(?,?,?,?,?,?)",  
 employeeID,  
 workStatisticDay.getStartTime(),  
 workStatisticDay.getEndTime(),  
 new java.sql.Date(date.getTime()),  
 workStatisticDay.isLateStart(),  
 workStatisticDay.isEarlyQuit());  
 return workStatisticDay;  
}  
  
@Override  
public Iterable<WorkStatisticDay> findByMonth(int employeeID, Date date) {  
 Date end = new java.sql.Date(date.getTime());  
 Date start = new java.sql.Date(date.getTime());  
 start.setDate(1);  
 date.setDate(1);  
 for(int i=1;i<=end.getDate();date.setDate(++i))  
 save(employeeID, date);  
 return jdbcTemplate.query(  
 "select \* from (select \* from WorkStatistic\_Day where employeeID=?) where date between ? and ? ",  
 this::mapRow2WorkStatisticDay,  
 employeeID,start, end);  
}

//控制器实现：

@GetMapping("workStatisticDay")  
public String workStatisticDay(  
 @PathParam("employeeID")String employeeID,  
 @PathParam("date")String date,  
 Model model) throws ParseException {  
 WorkStatisticDay workStatisticDay = jdbcWorkStatisticDayRepository.find(Integer.*parseInt*(employeeID),sdf\_d.parse(date));  
 if(workStatisticDay!=null){  
 model.addAttribute("wsd",workStatisticDay);  
 model.addAttribute("sta\_time",sdf\_time.format(workStatisticDay.getStartTime()));  
 model.addAttribute(("end\_time"),sdf\_time.format(workStatisticDay.getEndTime()));  
 model.addAttribute("msg\_date",sdf\_date.format(workStatisticDay.getDate()));  
 return "msg\_wsDay";  
 }else{  
 model.addAttribute("err\_msg","结果不存在");  
 model.addAttribute("redirect\_url","3;url=workStatisticDaySearch");  
 return "err";  
 }  
}

### 4.2.5每日工作时长报表模块

//Dao实现

@Override  
public WorkStatisticFlex save(int employeeID, Date date) {  
 Iterable<ClockRecord> start = jdbcClockRecordRepository.findByID\_Type\_Month(employeeID, date, ClockType.*IN*);  
 Iterable<ClockRecord> end = jdbcClockRecordRepository.findByID\_Type\_Month(employeeID, date, ClockType.*OUT*);  
 float length = 0;  
 for (ClockRecord record : start) {  
 for (ClockRecord record1 : end) {  
 if (record.getDate().getDay() == record1.getDate().getDay()) {  
 length += (record1.getClockTime().getTime() - record.getClockTime().getTime()) / 3600000;  
 break;  
 }  
 }  
 }  
  
 WorkStatisticFlex workStatisticFlex = new WorkStatisticFlex();  
 workStatisticFlex.setEmployeeID(employeeID);  
 workStatisticFlex.setLength(length);  
 StringBuffer sb = new StringBuffer();  
 sb.append((1900 + date.getYear()) + "-");  
 sb.append((1 + date.getMonth()) < 10 ? "0" + (1 + date.getMonth()) : 1 + date.getMonth() + "");  
 jdbcTemplate.update(  
 "insert into WorkStatistic\_Flex(employeeID,length,earningLength,date) values(?,?,?,?)",  
 employeeID,  
 length,  
 length - jdbcWorkLengthRequiredRepository.find(employeeID, sb.toString()).getLength(),  
 new java.sql.Date(date.getTime()));  
 return workStatisticFlex;  
}  
  
@Override  
public WorkStatisticFlex findOne(int employeeID, Date date) {  
 WorkStatisticFlex workStatisticFlex;  
 try {  
 save(employeeID, date);  
 workStatisticFlex = jdbcTemplate.queryForObject(  
 "select \* from WorkStatistic\_Flex where employeeID=? and date =?",  
 this::mapRow2WorkStatisticFlex,  
 employeeID, new java.sql.Date(date.getTime()));  
 return workStatisticFlex;  
 } catch (Exception e2) {  
 return null;  
 }  
  
}

//控制器实现：

@GetMapping("workStatisticFlex")  
public String workStatisticFlex(  
 @PathParam("employeeID")String employeeID,  
 @PathParam("date")String date,  
 Model model) throws ParseException {  
 WorkStatisticFlex workStatisticFlex = jdbcWorkStatisticFlexRepository.findOne(Integer.*parseInt*(employeeID),sdf\_d.parse(date));  
 if(workStatisticFlex!=null){  
 model.addAttribute("wsf",workStatisticFlex);  
 model.addAttribute("wsf\_date",sdf\_date.format(workStatisticFlex.getDate()));  
 return "msg\_wsFlex";  
 }else{  
 model.addAttribute("err\_msg","结果不存在");  
 model.addAttribute("redirect\_url","3;url=workStatisticFlexSearch");  
 return "err";  
 }  
}

### 4.2.6每月工作报表模块

//Dao实现

@Override  
public WorkStatisticMonth find(int employeeID, String date) {  
 WorkStatisticMonth workStatisticMonth;  
 try {  
 save(employeeID, date);  
 workStatisticMonth = jdbcTemplate.queryForObject(  
 "select \* from WorkStatistic\_Month where employeeID=? and date=?",  
 this::mapRow2WorkStatisticMonth,  
 employeeID, date);  
 return workStatisticMonth;  
 } catch (Exception e1) {  
 return null;  
 }  
}  
  
@Override  
public WorkStatisticMonth save(int employeeID, String date) {  
 Employee employee = jdbcEmployeeRepository.findOne(employeeID);  
 int year = Integer.*parseInt*(date.split("-")[0]) - 1900;  
 int month = Integer.*parseInt*(date.split("-")[1]) - 1;  
 if (employee == null)  
 return null;  
 Date date1 = new Date();  
 date1.setYear(year);  
 date1.setMonth(month);  
 date1.setDate(1);  
 Calendar calendar = Calendar.*getInstance*();  
 calendar.setTime(date1);  
 date1.setDate(calendar.getActualMaximum(Calendar.*DAY\_OF\_MONTH*));  
 if (employee.isFlexible()) {  
 WorkStatisticFlex workStatisticFlex = jdbcWorkStatisticFlexRepository.findOne(employeeID, date1);  
 if (workStatisticFlex != null)  
 jdbcTemplate.update(  
 "insert into WorkStatistic\_Month(employeeID,workCount,date,isFinish) values(?,?,?,?)",  
 employeeID,  
 workStatisticFlex.getLength(),  
 date,  
 workStatisticFlex.getEarningLength() >= 0);  
 } else {  
 int requiredDays = 0;  
 int actualDays = 0;  
 Iterable<WorkArrange> arranges = jdbcWorkArrangeRepository.findByMonth(employeeID, date1);  
 Iterable<WorkStatisticDay> statisticDays = jdbcWorkStatisticDayRepository.findByMonth(employeeID, date1);  
 for (WorkArrange ignored : arranges)  
 requiredDays++;  
 for (WorkStatisticDay ignored : statisticDays)  
 actualDays++;  
 jdbcTemplate.update(  
 "insert into WorkStatistic\_Month(employeeID,workCount,date,isFinish) values(?,?,?,?)",  
 employeeID,  
 actualDays,  
 date,  
 actualDays >= requiredDays);  
 }  
  
 return null;  
}

//控制器实现：

@GetMapping("workStatisticMonth")  
public String workStatisticMonth(  
 @PathParam("employeeID")String employeeID,  
 @PathParam("date")String date,  
 Model model) throws ParseException {  
 WorkStatisticMonth workStatisticMonth = jdbcWorkStatisticMonthRepository.find(Integer.*parseInt*(employeeID),date);  
 if(workStatisticMonth!=null){  
 model.addAttribute("wsm\_id",workStatisticMonth.getEmployeeID());  
 model.addAttribute("wsm\_length",jdbcEmployeeRepository.findOne(workStatisticMonth.getEmployeeID()).isFlexible()?  
 workStatisticMonth.getWorkCount()+"小时":workStatisticMonth.getWorkCount()+"天");  
 model.addAttribute("wsm\_date",workStatisticMonth.getDate());  
 return "msg\_wsMonth";  
 }else{  
 model.addAttribute("err\_msg","结果不存在");  
 model.addAttribute("redirect\_url","3;url=workStatisticMonthSearch");  
 return "err";  
 }  
}

4.3测试用例

在数据库中插入以下数据作为测试用例：

--添加员工及管理员信息  
insert into Employee(name,isFlexible) values ( 'Hari',TRUE );  
insert into Employee(name,isFlexible) values ( 'Mari',FALSE);  
insert into Manager(username,password) values('admin','');  
  
--弹性工作制工作时长设置  
insert into WorkLengthRequired(employeeID,length,date) VALUES(1,50,'2021-01');  
  
--固定工作制工作安排设置  
insert into WorkArrange(employeeID,startTime,endTime,date) values(2,'07:00:01','20:00:01','2021-01-01');  
insert into WorkArrange(employeeID,startTime,endTime,date) values(2,'07:00:01','20:00:01','2021-01-02');  
insert into WorkArrange(employeeID,startTime,endTime,date) values(2,'07:00:01','20:00:01','2021-01-03');  
  
--打卡记录  
INSERT INTO CLOCKRECORD(EMPLOYEEID,CLOCKTYPE,CLOCKTIME,DATE) VALUES(1,'IN','2021-01-03 07:00','2021-01-03');  
INSERT INTO CLOCKRECORD(EMPLOYEEID,CLOCKTYPE,CLOCKTIME,DATE) VALUES(1,'OUT','2021-01-03 19:50','2021-01-03');  
INSERT INTO CLOCKRECORD(EMPLOYEEID,CLOCKTYPE,CLOCKTIME,DATE) VALUES(1,'IN','2021-01-02 07:10','2021-01-02');  
INSERT INTO CLOCKRECORD(EMPLOYEEID,CLOCKTYPE,CLOCKTIME,DATE) VALUES(1,'OUT','2021-01-02 20:20','2021-01-02');  
INSERT INTO CLOCKRECORD(EMPLOYEEID,CLOCKTYPE,CLOCKTIME,DATE) VALUES(2,'IN','2021-01-03 07:00','2021-01-03');  
INSERT INTO CLOCKRECORD(EMPLOYEEID,CLOCKTYPE,CLOCKTIME,DATE) VALUES(2,'OUT','2021-01-03 19:50','2021-01-03');  
INSERT INTO CLOCKRECORD(EMPLOYEEID,CLOCKTYPE,CLOCKTIME,DATE) VALUES(2,'IN','2021-01-02 07:10','2021-01-02');  
INSERT INTO CLOCKRECORD(EMPLOYEEID,CLOCKTYPE,CLOCKTIME,DATE) VALUES(2,'OUT','2021-01-02 20:20','2021-01-02');

# 第5章 系统分析与改进

5.1测试结果与分析

### 5.1.1员工打卡模块测试

测试输入：

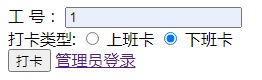
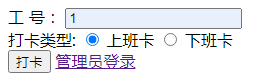


图5-1员工打卡模块测试输入

测试输出：

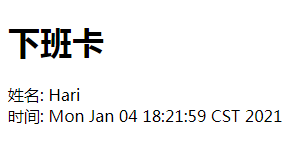
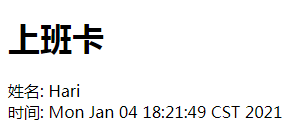


图5-2员工打卡模块测试输出

在数据库管理后台查看操作结果：

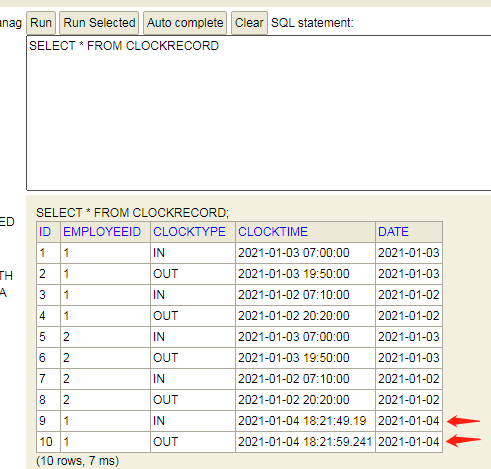


图5-3员工打卡模块数据库记录

### 5.1.2管理员登录模块测试

测试输入及输出:

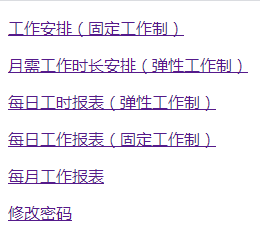
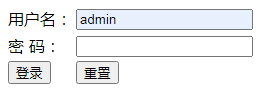


图5-4管理员登录模块输入及输出

查看进行登录操作后Cookie添加情况：

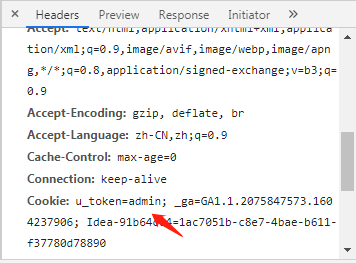


图5-5Cookie添加情况

根据登录后跳转的页面以及对Cookie的查看可以确认功能正常运行。

### 5.1.3工作安排模块测试

工作安排的输入如下：



图5-6工作安排输入

查看工作安排数据库记录：

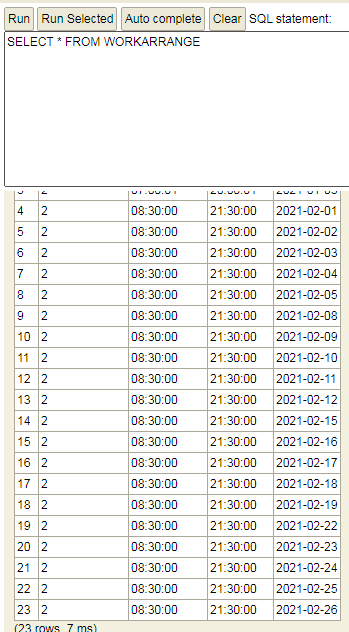


图5-7工作安排数据库记录

### 5.1.4工作时长安排模块测试

测试输入：

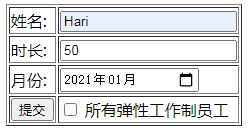


图5-8工作时长安排模块测试输入

操作后数据库表记录：

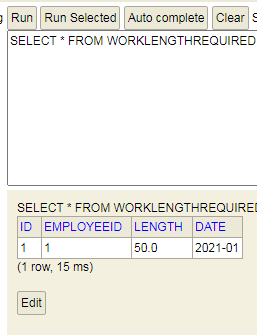


图5-9工作时长安排测试数据库记录

### 5.1.5统计报表模块测试

每日报表部分输入：

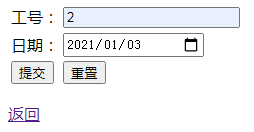
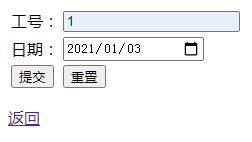


图5-10每日报表部分输入

输出：

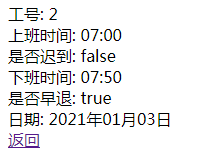
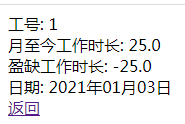


图5-11每日报表部分输出

月报表部分输入：

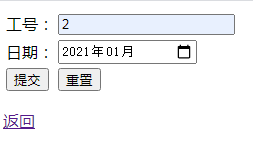


图5-12月报表部分输入

月报表部分输出：

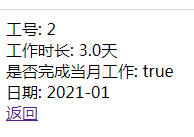
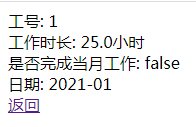


图5-13月报表部分输入

### 5.1.6密码修改模块测试

测试输入：

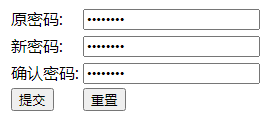


图5-14密码修改模块测试输入

操作后表记录：

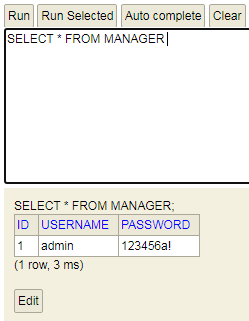


图5-15操作后表记录

5.2可改进的部分

此次项目还有因为工期时间的要求原因，所以面向用户的界面都过于简陋，只有一个简单的框架。所以在用户界面方面还有非常大的改进空间。

另外，从之前的代码逻辑可以看到，报表模块的生成逻辑是当用户发起查询要求时即生成报表，由于开发人员的技术水平原因而没有达到定时自动生成报表的要求。这一方面是非常大的不足。

# 第6章 心得体会

通过此次课程设计,使我更加扎实的掌握了有关软件工程方面的知识,在设计过程中虽然遇到了一些问题,但经过一次又一次的思考,一遍又一遍的检查终于找出了原因所在,也暴露出了前期我在这方面的知识欠缺和经验不足。实践出真知,通过亲自动手设计实现,使我们掌握的知识不再是纸上谈兵。

过而能改,善莫大焉在课程设计过程中,我们不断发现错误,不断改正,不断领悟。最终的检测调试环节,本身就是在践行过而能改,善莫大焉的知行观。这次课程设计终于顺利完成了,在设计中遇到了很多问题,最后在老师的指导下,终于游逆而解。在今后社会的发展和学习实践过程中,一定要不懈努力,不能遇到问题就想到要退缩,一定要不厌其烦的发现问题所在,然后一进行解决,只有这样,才能成功的做成想做的事,才能在今后的道路上劈荆斩棘,而不是知难而退,那样永远不可能收获成功,收获喜悦,也永远不可能得到社会及他人对你的认可!

软件工程课程设计诚然是一门专业课,给我很多专业知识以及专业技能上的提升,同时又是一门讲道课,一门辩思课,给了我许多道,给了我很多思,给了我莫大的空间。同时,设计让我感触很深。使我对抽象的理论有了具体的认识。通过这次课程设计,我掌握了软件工程结构化设计和面向对象设计的知识，以及项目中可行性分析、需求分析、概要设计、详细设计的内容。我认为,在这学期的实验中,不仅培养了独立思考、动手操作的能力,在各种其它能力上也都有了提高。更重要的是,在实验课上,我们学会了很多学习的方法。而这是日后最实用的,真的是受益匪浅。要面对社会的挑战,只有不断的学习、实践,再学习、再实践。这对于我们的将来也有很大的帮助。以后,不管有多苦,我想我们都能变苦为乐,找寻有趣的事情,发现其中珍贵的事情。

# 参考文献

**[1] 李代平，杨成义 . 软件工程实践与课程设计[M] . –北京：清华大学出版社，2017**

**[2] 李代平，杨成义 . 软件工程（第4版）[M]，-北京：清华大学出版社，2017**

**[3] [美]克雷格·沃斯 . Spring实战（第5版）[M]. 人民邮电出版社，2020-02**

**[4]诸华，霍秋艳 . 软件设计师教程[M] –5版. –北京：清华大学出版社，2018**

**[5]陈志泊主编；崔晓晖，路贺俊，阮豫红.数据库原理及应用教程：微课版 -4版 –北京：人民邮电出版社，2017.11**

**致 谢**

时光如梭，转眼就到了学期末，我的课程设计也到了尾声。从论文的选题、论文大纲的编写和论文正文的撰写编排都离不开我的老师、同学和朋友的帮助。

在这里最重要的是要感谢我们的李代平老师。在我们课程设计选题的时候就给予了细心的指导，使我有了明确的目标和设计方向。在后来课程设计修改中，李老师对于每一个标点、每一个错别字都研究的为我之处，并在写作中提出了许多宝贵的意见和对我的不懈鼓励和支持，使我能够不断地学习提高。感谢我的同学,在论文中对我提出的意见和帮助,还有在其他生活中学习中给予我支持和鼓励的朋友们,在这里,对于老师,同学,朋友再一次表示我最真诚的感谢

# 附录

表1-1上下班打卡用例描述

|  |  |
| --- | --- |
| 用例名称： | 上下班打卡 |
| 参与者： | 员工 |
| 前置条件： | 员工进入到打卡页面 |
| 用例功能： | 记录员工上下班离开公司时间 |
| 事件流： | 员工输入工号与打卡类型后保存信息到数据库 |
| 异常事件流： | 工号不存在 |
| 后置条件： | 系统无改变 |

表1-2修改密码用例描述

|  |  |
| --- | --- |
| 用例名称： | 修改密码 |
| 参与者： | 系统管理员 |
| 前置条件： | 系统管理员登录到后台管理界面 |
| 用例功能： | 供系统管理员修改个人账号密码 |
| 事件流： | 系统管理员成功修改密码后返回到登录界面 |
| 异常事件流： | 输入的原密码不正确或新密码不符合格式 |
| 后置条件： | 系统无改变 |

表1-3考勤统计用例描述

|  |  |
| --- | --- |
| 用例名称： | 考勤统计 |
| 参与者： | 系统管理员 |
| 前置条件： | 系统管理员登录到后台管理界面 |
| 用例功能： | 供系统管理员查询员工考勤的报表信息 |
| 事件流： | 输入查询条件后返回报表信息 |
| 异常事件流： | 输入查询条件有误 |
| 后置条件： | 系统无改变 |

表1-4考勤管理用例描述

|  |  |
| --- | --- |
| 用例名称： | 考勤管理 |
| 参与者： | 系统管理员 |
| 前置条件： | 系统管理员登录到后台管理界面 |
| 用例功能： | 供系统管理员对考勤信息查询和改动以及安排员工工作 |
| 事件流： | 输入查询条件或修改信息 |
| 异常事件流： | 输入错误有误或不符合格式 |
| 后置条件： | 系统无改变 |

表2-1数据库表描述

|  |  |  |
| --- | --- | --- |
| 编号 | 表名称 | 描述 |
| 1 | Emplyee | 员工表 |
| 2 | Manager | 系统管理员表 |
| 3 | ClockRecord | 打卡记录表 |
| 4 | WorkArrange | 工作安排表 |
| 5 | WorkLengthRequired | 需求工作时长表 |
| 6 | WorkStatistic\_Day | 每日工作统计表 |
| 7 | WorkStatistic\_Day | 每日工时统计表 |
| 8 | WorkStatistic\_Month | 每月工作统计表 |

表3-1持久化类描述表

|  |  |  |
| --- | --- | --- |
| 编号 | 类名 | 描述 |
| 1 | ClockRecord | 打卡记录类 |
| 2 | Employee | 员工类 |
| 3 | Manager | 系统管理员账户类 |
| 4 | WorkArrange | 工作安排类 |
| 5 | WorkLengthRequired | 月需工作时长类 |
| 6 | WorkStatisticDay | 每日工作时间报表类 |
| 7 | WorkStatisticFlex | 每日工作时长报表类 |
| 8 | WorkStatisticMonth | 每月工作时间报表类 |

表3-2数据库操作类描述表

|  |  |  |
| --- | --- | --- |
| 编号 | 类名 | 描述 |
| 1 | JdbcClockRecordRepository | 打卡记录Dao操作实现类 |
| 2 | JdbcEmployeeRepository | 员工Dao操作实现类 |
| 3 | JdbcManagerRepository | 管理员Dao操作实现类 |
| 4 | JdbcWorkArrangeRepository | 工作安排Dao操作实现类 |
| 5 | JdbcWorkLengthRequiredRepository | 月需工作时长Dao操作实现类 |
| 6 | JdbcWorkStatisticDayRepository | 每日工作时间报表Dao操作实现类 |
| 7 | JdbcWorkStatisticFlexRepository | 每日工作时长报表Dao操作实现类 |
| 8 | JdbcWorkStatisticMonthRepository | 每月工作时间报表Dao操作实现类 |

表3-3 控制器类描述表

|  |  |  |
| --- | --- | --- |
| 编号 | 类名 | 描述 |
| 1 | ClockRecordControler | 打卡操作控制器类，接受打卡请求控制打卡操作流程。 |
| 2 | LoginController | 登录操作控制器类，主要处理管理员登录后台管理页面请求。 |
| 3 | ManagerControler | 管理员账户控制器类，主要处理管理员账户修改相关的请求。 |
| 4 | PageController | 页面控制器类，主要处理客户端发来的页面请求，返回用户需要的界面。 |
| 5 | StatisticController | 报表控制器类，处理查询报表相关的请求。 |
| 6 | WorkArrangeController | 工作安排控制器类，处理管理员安排员工工作时间的请求。 |
| 7 | WorkLengthRequiredContrller | 需工作时长控制器类，完成设置弹性工作制员工每月所需工作时长操作。 |

表3-4员工表描述

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| name | VARCHAR(20) | 否 | 否 |
| isFlexible | BOOL | 否 | 否 |

表3-5打卡记录表描述

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| employeeID | INT | 否 | 否 |
| clockType | VARCHAR(3) | 否 | 否 |
| clockTime | dateTime | 否 | 否 |
| date | DATE | 否 | 否 |

表3-6管理员账户表描述

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| username | VARCHAR(20) | 否 | 否 |
| password | VARCHAR(20) | 否 | 否 |

表3-7工作安排表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| employeeID | INT | 否 | 否 |
| startTime | TIME | 否 | 否 |
| endTime | TIME | 否 | 否 |
| date | DATE | 否 | 否 |

表3-8需工作时长表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| employeeID | INT | 否 | 否 |
| length | FLOAT | 否 | 否 |
| date | DATE | 否 | 否 |

表3-9每日工作时长报表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| employeeID | INT | 否 | 否 |
| length | FLAOT | 否 | 否 |
| date | DATE | 否 | 否 |
| earningLength | FLAOT | 否 | 否 |

表3-10每日工作时间表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| employeeID | INT | 否 | 否 |
| startTime | TIME | 否 | 否 |
| endTime | TIME | 否 | 否 |
| date | DATE | 否 | 否 |
| isLateStart | BOOL | 否 | 否 |
| isEarlyQuit | BOOL | 否 | 否 |

表3-11每月工作时间表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 主键 | 是否允许空值 |
| id | INT | 是 | 否 |
| employeeID | INT | 否 | 否 |
| workCount | FLOAT | 否 | 否 |
| isFinish | BOOL | 否 | 否 |