

《软件设计与体系结构》

课程设计报告

( 2024 - 2025 学年 第一学期)

**题目：智能假期考勤一体化管理系统**

姓 名： 刘硕

学 号： 202209080329

班 级： 2022级3班

专 业： 软件工程

系 （院）： 信息科学与工程学院

指导老师： 陈新疆

2024年 12月3日

目录

[1.需求分析 1](#_Toc185631395)

[1.1需求目标分析和可行性分析[1] 1](#_Toc185631396)

[1.2 功能分析 2](#_Toc185631397)

[1.3USE-CASE图 5](#_Toc185631406)

[1.4用例描述； 5](#_Toc185631407)

[1.5功能清单 6](#_Toc185631408)

[2.系统设计 7](#_Toc185631414)

[2.1.技术选型： 7](#_Toc185631415)

[2.2总体框架设计 7](#_Toc185631416)

[2.3UML设计 8](#_Toc185631417)

[2.4原型界面设计 11](#_Toc185631418)

[2.5算法分析与设计 12](#_Toc185631419)

[2.6数据库设计 13](#_Toc185631420)

[3.系统实现 18](#_Toc185631421)

[3.1模块之间的接口的具体实现 18](#_Toc185631422)

[3.2主要的业务逻辑和流程的具体实现。（部分给出流程图）[3][5] 23](#_Toc185631423)

[3.3. 界面的具体实现 26](#_Toc185631424)

[4.系统测试 33](#_Toc185631425)

[4.1测试环境 33](#_Toc185631426)

[4.2关键函数功能描述[2] 33](#_Toc185631427)

[4.3测试用例与测试结果 34](#_Toc185631428)

[5.总结 36](#_Toc185631429)

[5.1设计 36](#_Toc185631430)

[5.2收获 36](#_Toc185631431)

[5.3不足 36](#_Toc185631432)

[5.4改进方向 36](#_Toc185631433)

[6.参考文献 38](#_Toc185631434)

[7.致谢 39](#_Toc185631435)

1.需求分析

1.1需求目标分析和可行性分析[1]

目的：该系统的目的是为公司提供一个有效的工具，用于管理员工的考勤记录、假期申请、员工状态和签到签出等。这可以帮助雇主提高管理效率，避免人工记录错误，提高透明度，并加强与员工的沟通。

背景：在企业管理中，考勤和假期管理一直是一个复杂且容易出错的过程。传统的人工记录和管理方式不仅容易出错，还可能造成员工与管理层之间的矛盾。随着企业规模的扩大和管理需求的复杂化，传统方式无法满足现代企业的需求。因此，自动化的考勤与假期管理系统成为了必然的选择。

目标：

* 提高管理效率：通过自动化的考勤和假期管理，减少人工干预，提高工作效率。
* 减少人为错误：系统化的记录方式可以避免人工记录时出现的错误。
* 提高透明度：员工可以随时查询自己的假期、考勤等信息，增强信任感。
* 优化资源管理：通过数据分析，帮助公司更好地进行员工资源调配。

可行性分析：

**技术可行性**：从技术层面看，使用 Spring 框架、MyBatis 以及数据库技术，项目的开发是完全可行的。相关的开发工具和框架（如 Spring Boot、MyBatis）都有广泛的应用案例，开发人员可以利用现有技术实现大部分功能。

**经济可行性**：从经济角度看，系统开发的投入主要集中在初期的开发阶段，但一旦系统投入使用，它能显著提高企业管理效率，节省人工成本，长远来看具备较高的投资回报率。

**法律和道德风险**：从数据保护的角度，系统需要确保员工个人信息的安全，包括考勤记录、假期数据等。应遵循 GDPR（欧盟数据保护法规）等相关法律要求，保障员工的隐私和数据安全。

存在的问题：

**手工管理易出错**：当前的考勤和假期管理往往依赖人工记录，容易产生误差，导致员工对结果不满。

**信息不透明**：员工无法实时了解自己的考勤和假期状况，可能产生不必要的纠纷。

**数据难以统计与分析**：如果没有一个系统来记录和管理，管理者可能需要花费大量时间进行人工统计，无法有效进行数据分析。

解决这些问题的价值：

通过实现一个自动化系统，可以提高数据准确性，减少人为错误，并提供员工自助查询功能，提升工作透明度。系统化的数据记录和统计功能可以帮助管理者更高效地调配资源和进行数据分析。通过自动化审批流程，可以减少时间消耗，提高处理效率。

技术和经济可行性：

技术上可行：使用 Spring、MyBatis 等成熟的技术栈完全可以支持该项目的开发。

**经济上可行**：初期的开发投资主要用于开发人员、测试人员等人力资源，系统投入后可以显著提升效率，减少人工管理成本，经济回报良好。

1.2 功能分析

功能分析是需求分析阶段的重要环节，它详细描述了系统应实现的具体功能、操作流程及用户需求。对于本考勤和假期管理系统来说，功能分析涵盖了员工签到、假期管理、员工信息管理、报表统计和通知提醒等方面。下面是该系统的主要功能分析：

### 1.2.1 员工签到/签出功能

该功能是系统的核心功能之一，主要涉及员工考勤的记录和管理。通过该功能，员工可以在系统中进行签到和签出，确保考勤数据的准确性和实时性。具体分析如下：

**签到功能：**员工登录后，点击签到按钮，系统会自动记录当前时间并将该记录保存到数据库中。如果员工已经签到或遇到网络问题，系统会进行相应的提示，避免重复签到和数据丢失。

**签出功能：**员工完成工作后，点击签出按钮，系统记录签出时间，并计算工时。签出功能同样需要考虑异常情况，如网络故障等，确保操作的顺利完成。

**查看考勤记录：**员工可以随时查询自己的签到/签出记录，透明化管理，减少误解和纠纷。

### 1.2.2 假期管理功能

假期管理功能主要包括假期申请、审批和查询。通过该功能，员工可以轻松申请假期，管理员可以审批假期，确保公司假期管理的高效性和准确性。

**假期申请：**员工可以登录系统，选择假期类型（如年假、病假、事假等），并填写假期的起止时间。提交申请后，系统将自动发送申请给管理员进行审批。

假期审批：管理员可以查看员工的假期申请，并根据公司的政策进行审批。审批通过后，系统会通知员工假期申请已通过，反之则通知申请被拒绝。

假期余额查询：员工可以查看自己的假期余额，系统根据员工的工作时间和已用假期自动计算假期余额。假期余额不足时，系统会给予提醒，避免员工因假期不足而提交无效申请。

### 1.2.3 员工信息管理功能

该功能是管理员管理员工个人信息的基础功能，涉及员工的基本信息、考勤记录、假期记录等的维护和更新。

员工信息管理：管理员可以在系统中添加、删除或更新员工的信息（如姓名、工号、入职时间、部门等）。这些信息将用于考勤和假期管理。

数据维护：管理员可以查看和编辑员工的考勤记录与假期历史，确保系统数据的完整性和准确性。

### 1.2.4 报表与统计功能

报表与统计功能帮助管理员对员工的考勤、假期等数据进行综合分析，支持公司资源的优化调配。

**员工考勤报表：**管理员可以查看所有员工的考勤数据，按日期、部门等维度进行汇总与分析，生成考勤报表。这些报表将为员工绩效考核、薪资结算等提供依据。

**假期统计报表：**管理员可以查看所有员工的假期使用情况，分析假期分配与使用的规律，便于合理安排公司资源，减少假期的浪费。

**导出功能：**管理员可以将考勤报表和假期统计报表导出为 Excel 或 PDF 格式，方便存档或共享给相关部门。

### 1.2.5 通知与提醒功能

通知与提醒功能有助于提高系统的自动化和员工的参与度，确保员工和管理员的沟通及时有效。

**员工假期提醒：**当员工的假期余额低于设定的阈值时，系统会自动发送提醒通知，提示员工申请假期的相关事宜，避免因为假期余额不足而造成不必要的纠纷。

**签到提醒：**如果员工在工作时间内未进行签到或签出，系统会自动提醒员工进行考勤操作。

**审批通知：**当管理员审批完员工的假期申请后，系统将自动通知员工审批结果（通过或拒绝），确保员工及时了解申请进展。

### 1.2.6 权限控制功能

为了确保系统的安全性和信息的私密性，权限控制功能将定义不同角色的访问权限。具体分析如下：

**员工角色权限：**员工只能查看和管理自己的考勤记录、假期余额及个人信息，提交假期申请，但不能查看其他员工的数据。

**管理员角色权限：**管理员可以查看和管理所有员工的考勤记录、假期记录、审批假期申请、生成报表等。管理员角色有最高权限，可以对系统中的所有数据进行维护和管理。

### 1.2.7 系统日志与审计功能

该功能能够记录系统中的关键操作日志，确保系统操作的可追溯性，并为管理员提供审计支持。

**日志记录：**系统将记录每一次员工签到、签出、假期申请、审批等操作，并存储相关日志，确保数据的完整性和安全性。

**审计功能：**管理员可以查看系统的操作日志，追踪异常操作或安全事件，保障系统的稳定性和数据安全。

### 总结

本考勤与假期管理系统的功能分析涵盖了员工考勤、假期管理、信息维护、数据统计与报表生成、通知提醒、权限控制等多个方面，所有功能都旨在提高管理效率、减少人为错误、提升透明度并加强员工与管理层的沟通。通过自动化和系统化的管理方式，企业能够更高效地进行资源调配、数据分析和决策支持，确保员工与公司之间的良好合作与信任。

1.3USE-CASE图



图1-1 用例图

1.4用例描述；

用例 1：员工签到

1. 员工在系统中登录。
2. 员工点击签到按钮。
3. 系统记录员工的签到时间。
4. 员工成功签到，并显示签到成功信息。

**异常事件流：**

1. 如果员工已经签到，系统会提示“您已签到”。
2. 如果网络问题导致无法签到，系统会提示“网络异常，请稍后重试”。

用例 2：员工申请假期

1. 员工在系统中登录。
2. 员工点击“申请假期”。
3. 填写假期类型和开始结束时间。
4. 提交申请，系统发送假期申请给管理员。
5. 管理员审批通过或拒绝，员工收到通知。

异常事件流：

1. 如果员工假期余额不足，系统提示“假期余额不足”。
2. 如果管理员拒绝假期申请，系统提示“假期申请被拒绝”。

1.5功能清单

### 员工签到/签出功能：

1. 签到/签出操作。
2. 查看自己的考勤记录。

### 假期管理功能：

1. 提交假期申请。
2. 假期余额查询。
3. 假期申请审批（管理员角色）。

### 员工信息管理功能：

1. 管理员可以添加、删除、更新员工信息。

### 报表与统计功能：

1. 员工可以查看自己的考勤记录。
2. 管理员查看全员的考勤和假期数据。

### 通知与提醒功能：

假期余额低于一定阈值时自动提醒员工。

2.系统设计

2.1.技术选型：

**后端框架：**Spring Boot + Spring MVC + MyBatis。

Spring Boot 提供了快速的开发环境，简化了配置和部署流程，使得开发人员可以快速搭建和运行应用。它还提供了广泛的第三方库支持，方便集成各种服务，对于简单的系统来说更加方便。

Spring MVC 用于处理请求和响应，适合实现清晰的分层架构，使得代码更加模块化和易于更新维护，当需要增加或删除新功能时不会影响其他模块的运行。

MyBatis 能够灵活地与数据库交互，，支持编写复杂的SQL语句，并且能够轻松地映射数据库结果到Java对象，提高了开发效率和灵活性。

**数据库：**MySQL。MySQL 是一个开源数据库，降低软件成本。具备高效的数据存储和查询能力，适合开发以打卡考勤功能为主的中小型管理系统。同时，对于需要快速开发和部署的项目，MySQL提供了足够的性能和灵活性

**前端技术：**Vue.js+Element UI。Vue.js 提供了简单易用的数据绑定和组件化开发，便于实现动态用户界面。Element UI 是一款成熟的前端组件库，支持美观大方的 UI 设计。Element UI的组件库减少了前端开发的工作量，因为它提供了预先设计好的组件，可以直接使用或根据需要进行定制。

2.2总体框架设计

**分层设计：**

表示层（前端）：Vue.js 与 RESTful API 通信，提供用户界面。

业务逻辑层（后端服务）：Spring Boot 实现业务逻辑，包括假期管理、考勤管理等模块。

数据访问层：使用 MyBatis 与数据库交互。

数据库层：MySQL 存储用户、考勤和假期等数据。

**模块划分：**

员工管理模块：用户注册、登录、信息更新等功能。

假期管理模块：假期申请、审批、余额查询。

考勤管理模块：签到签退记录管理。

报表模块：员工个人和全员的统计与图表展示。

2.3UML设计

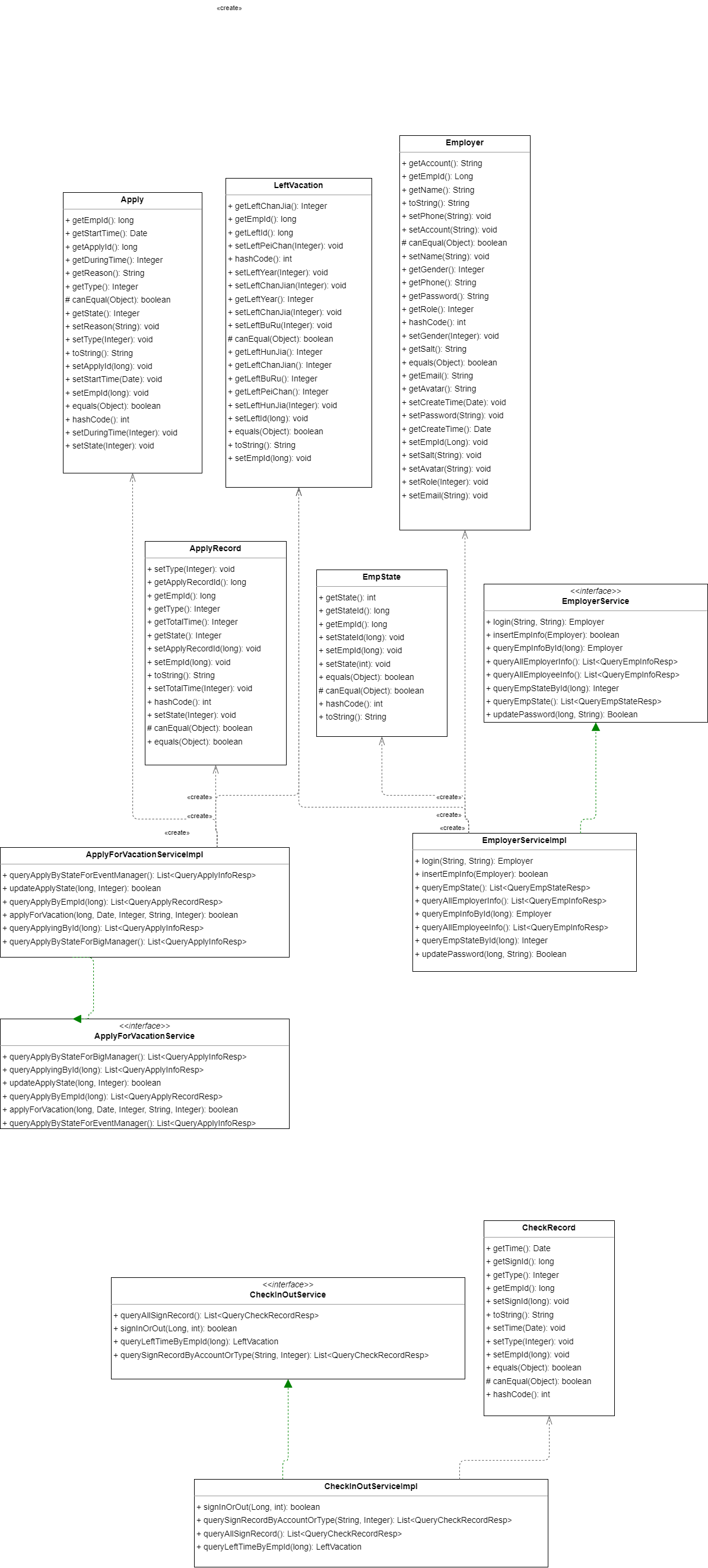


图2-1 类图

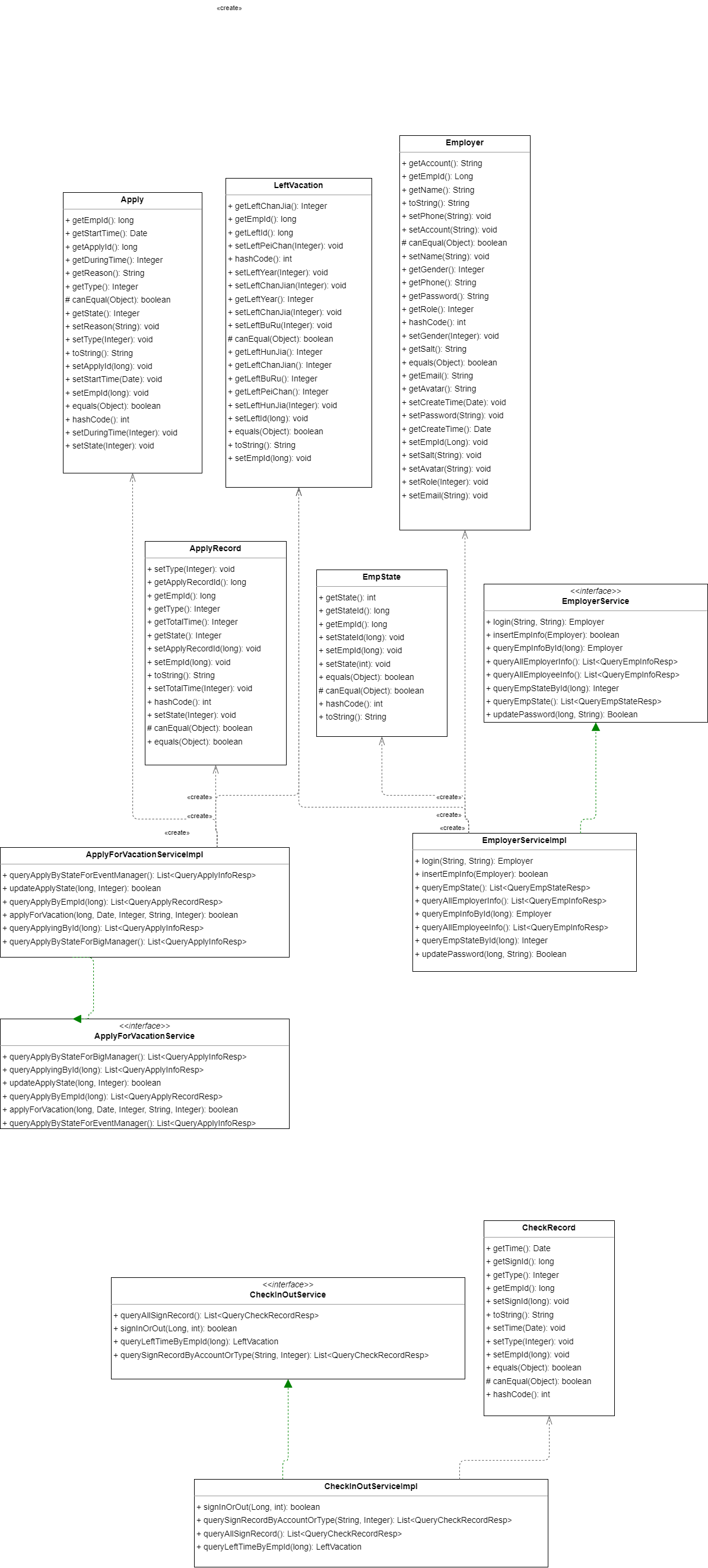


图2-2 类图



图2-3 员工假期申请成功顺序图



图2-4 员工考勤用例图



图2-5 员工请假申请状态图

2.4原型界面设计

原型设计工具：AxureRP

功能预览：



图2-6 用户登录页

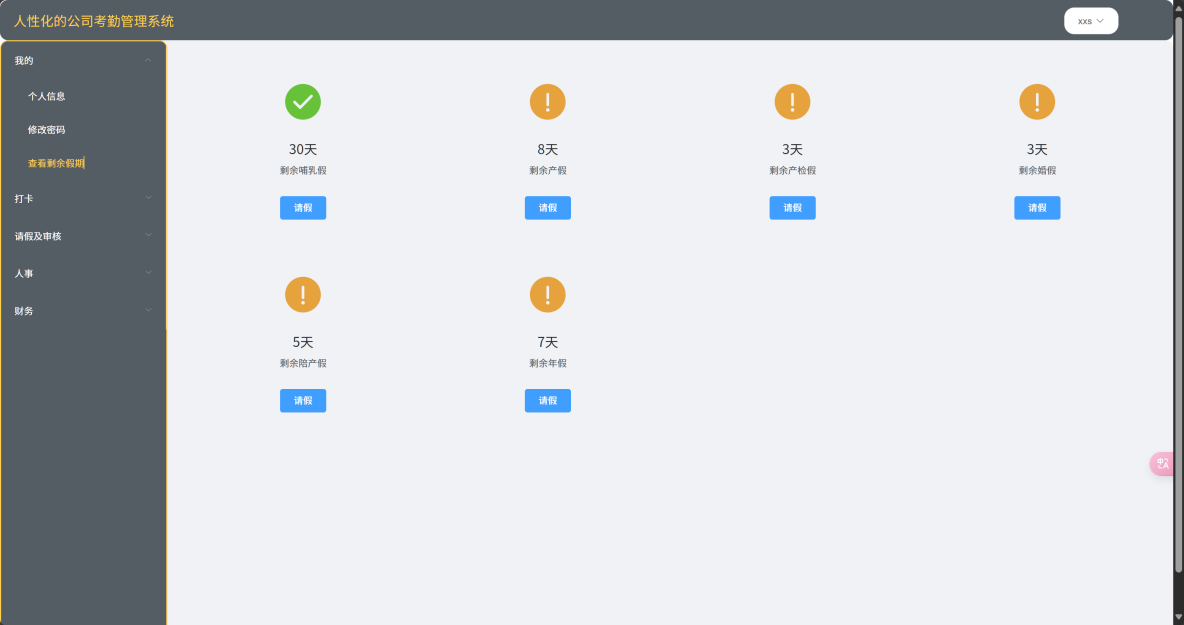


图2-7 用户打卡界面

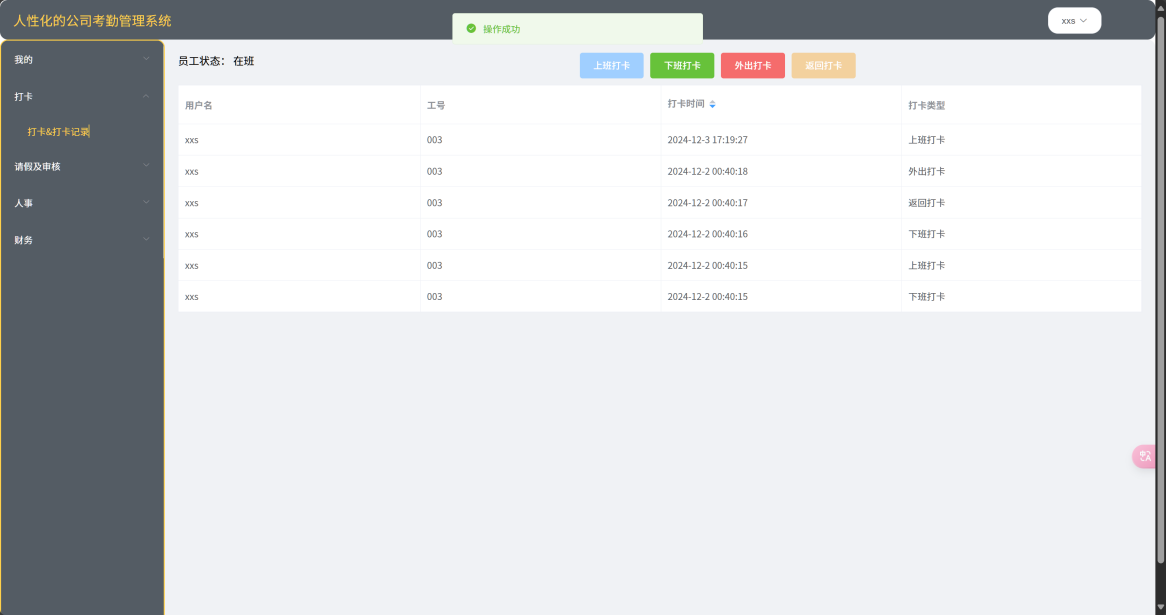


图2-8 假期申请页

图形用户界面, 文本, 应用程序

描述已自动生成

图2-9 管理员审批页。

2.5算法分析与设计

**考勤计算**：

基于签到签退记录，计算工作时长、迟到/早退等。

算法：时间差计算。

**假期余额管理**：

根据员工已使用的假期，动态更新剩余天数。

算法：动态递减算法。

**统计报表生成**：

数据聚合与分类统计。

算法：SQL 分组与聚合函数（如 GROUP BY）。

2.6数据库设计

**数据存储：**

使用 MySQL 数据库，已设计的表结构完全满足系统需求。

表设计：

**Apply(请假申请表)：**  
CREATE TABLE `apply` (

`apply\_id` bigint NOT NULL AUTO\_INCREMENT,

`start\_time` datetime NULL DEFAULT NULL,

`during\_time` int NULL DEFAULT NULL,

`reason` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL,

`type` int NULL DEFAULT NULL,

`state` int NULL DEFAULT NULL,

`emp\_id` bigint NOT NULL,

PRIMARY KEY (`apply\_id`, `emp\_id`) USING BTREE,

INDEX `emp\_id`(`emp\_id` ASC) USING BTREE,

CONSTRAINT `apply\_ibfk\_1` FOREIGN KEY (`emp\_id`) REFERENCES `employer` (`emp\_id`) ON DELETE RESTRICT ON UPDATE RESTRICT

) ENGINE = InnoDB AUTO\_INCREMENT = 34 CHARACTER SET = utf8 COLLATE = utf8\_general\_ci ROW\_FORMAT = Dynamic;

**Apply\_record(请假记录表):**

CREATE TABLE `apply\_record` (

`apply\_record\_id` bigint NOT NULL AUTO\_INCREMENT,

`emp\_id` bigint NULL DEFAULT NULL,

`type` int NULL DEFAULT NULL,

`total\_time` int NULL DEFAULT NULL,

`state` int NULL DEFAULT NULL,

PRIMARY KEY (`apply\_record\_id`) USING BTREE,

INDEX `emp\_id`(`emp\_id` ASC) USING BTREE,

CONSTRAINT `apply\_record\_ibfk\_1` FOREIGN KEY (`emp\_id`) REFERENCES `employer` (`emp\_id`) ON DELETE RESTRICT ON UPDATE RESTRICT

) ENGINE = InnoDB AUTO\_INCREMENT = 23 CHARACTER SET = utf8 COLLATE = utf8\_general\_ci ROW\_FORMAT = Dynamic;

Check\_record(打卡记录表)

CREATE TABLE `check\_record` (

`sign\_id` bigint NOT NULL AUTO\_INCREMENT,

`time` datetime NULL DEFAULT NULL,

`type` int NULL DEFAULT NULL COMMENT '0：上班打卡\n1：下班打卡\n2：外出打卡\n3：返回打卡',

`emp\_id` bigint NULL DEFAULT NULL,

PRIMARY KEY (`sign\_id`) USING BTREE,

INDEX `emp\_id`(`emp\_id` ASC) USING BTREE,

CONSTRAINT `check\_record\_ibfk\_1` FOREIGN KEY (`emp\_id`) REFERENCES `employer` (`emp\_id`) ON DELETE RESTRICT ON UPDATE RESTRICT

) ENGINE = InnoDB AUTO\_INCREMENT = 253 CHARACTER SET = utf8 COLLATE = utf8\_general\_ci ROW\_FORMAT = Dynamic;:

Emp\_state(员工状态表):

CREATE TABLE `emp\_state` (

`state\_id` bigint NOT NULL AUTO\_INCREMENT,

`emp\_id` bigint NULL DEFAULT NULL,

`state` int NULL DEFAULT NULL COMMENT '0：未在班\n1：在班',

PRIMARY KEY (`state\_id`) USING BTREE,

INDEX `emp\_id`(`emp\_id` ASC) USING BTREE,

CONSTRAINT `emp\_state\_ibfk\_1` FOREIGN KEY (`emp\_id`) REFERENCES `employer` (`emp\_id`) ON DELETE RESTRICT ON UPDATE RESTRICT

) ENGINE = InnoDB AUTO\_INCREMENT = 14 CHARACTER SET = utf8 COLLATE = utf8\_general\_ci ROW\_FORMAT = Dynamic;

Employer(员工信息表):

CREATE TABLE `employer` (

`emp\_id` bigint NOT NULL AUTO\_INCREMENT,

`account` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL,

`name` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL,

`gender` int NULL DEFAULT NULL COMMENT '0：女\n1：男',

`phone` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL,

`password` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL,

`salt` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL,

`email` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL,

`create\_time` datetime NULL DEFAULT NULL,

`role` int NULL DEFAULT NULL,

`avatar` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL,

PRIMARY KEY (`emp\_id`) USING BTREE,

INDEX `emp\_id`(`emp\_id` ASC) USING BTREE

) ENGINE = InnoDB AUTO\_INCREMENT = 16 CHARACTER SET = utf8 COLLATE = utf8\_general\_ci ROW\_FORMAT = Dynamic;

Left\_vacation(剩余假期表):

CREATE TABLE `left\_vacation` (

`left\_id` bigint NOT NULL AUTO\_INCREMENT,

`emp\_id` bigint NULL DEFAULT NULL,

`left\_year` int NULL DEFAULT NULL,

`left\_hun\_jia` int NULL DEFAULT NULL,

`left\_chan\_jian` int NULL DEFAULT NULL,

`left\_chan\_jia` int NULL DEFAULT NULL,

`left\_bu\_ru` int NULL DEFAULT NULL,

`left\_pei\_chan` int NULL DEFAULT NULL,

`total` int NULL DEFAULT NULL,

PRIMARY KEY (`left\_id`) USING BTREE,

INDEX `emp\_id`(`emp\_id` ASC) USING BTREE,

CONSTRAINT `left\_vacation\_ibfk\_1` FOREIGN KEY (`emp\_id`) REFERENCES `employer` (`emp\_id`) ON DELETE RESTRICT ON UPDATE RESTRICT

) ENGINE = InnoDB AUTO\_INCREMENT = 14 CHARACTER SET = utf8 COLLATE = utf8\_general\_ci ROW\_FORMAT = Dynamic;

共有6个表：apply(请假申请表）,apply\_record（请假记录）,check\_record（打卡记录）,emp\_state（员工状态）,employer（员工信息表）,left\_vacation（假期剩余）

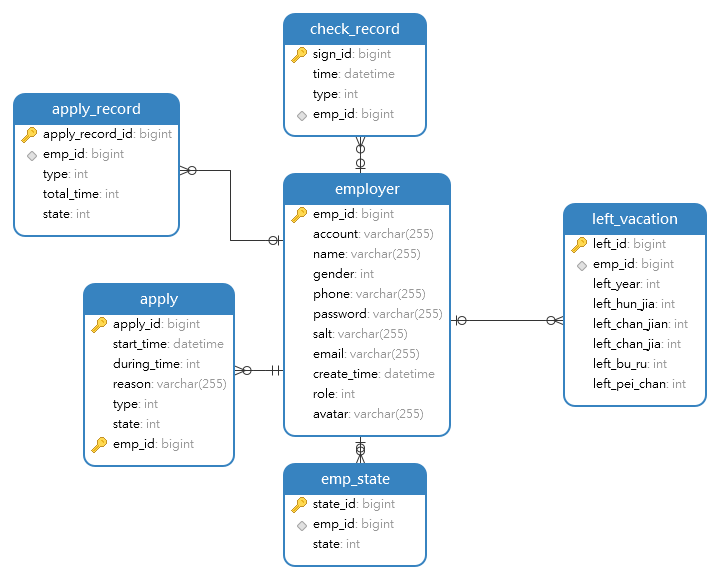


图2-10 数据库物理模型图

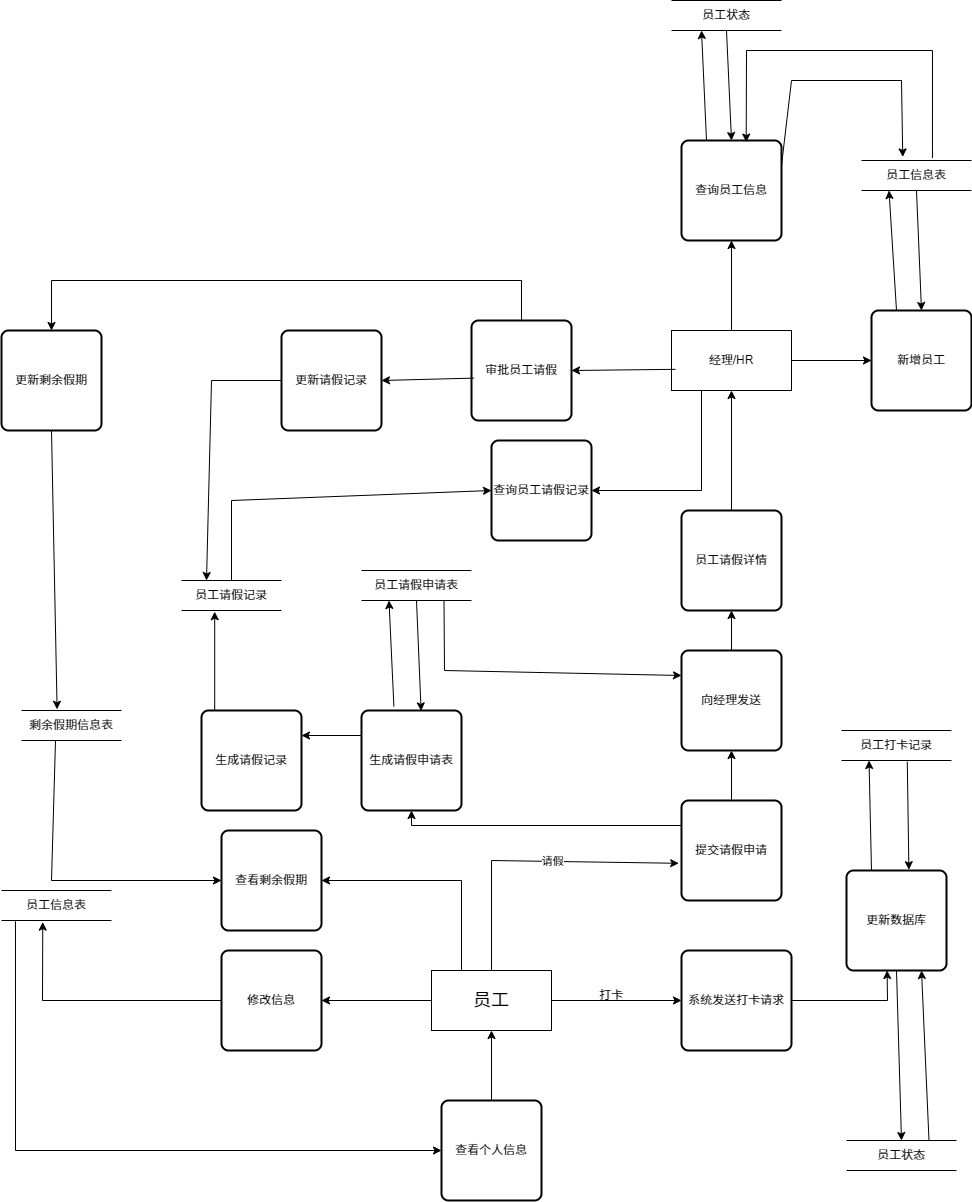


图2-11 软件数据流图

3.系统实现

3.1模块之间的接口的具体实现

本系统采用 RESTful API 风格设计接口，各模块通过 HTTP 请求进行交互。以下是主要接口的定义与实现：

**员工模块[3][4]**

用户登录接口

URL: POST /login

请求体:

{

"account": "string",

"password": "string"

}

返回: 登录结果，包括用户的基本信息。

{

"status": "success",

"message": "登录成功",

"data": {

"empId": "long",

"account": "string",

"name": "string",

"role": "string"

}

}

员工信息查询接口

URL: GET /queryEmpInfoById/{empId}

参数: empId（路径参数，员工编号）

返回: 员工的详细信息。

{

"empId": "long",

"account": "string",

"name": "string",

"role": "string",

"state": "integer"

}

**考勤模块**

打卡接口

URL: POST /checkInOrOut/{empId}/{signType}

参数:

empId（路径参数，员工编号）

signType（路径参数，打卡类型，1为签到，2为签退）

返回: 是否打卡成功的结果。

{

"status": "success",

"message": "打卡成功"

}

查询所有打卡记录接口

URL: GET /queryAllRecords

返回: 所有打卡记录的列表。

[

{

"empId": "long",

"signType": "integer",

"signTime": "string"

}

]

根据员工ID查询剩余假期时间接口

URL: GET /getLeftTimeById/{empId}

返回: 员工的剩余假期时间。

{

"empId": "long",

"leftVacationTime": "integer"

}

**假期管理模块**

假期申请接口

URL: POST /applyVacation/{empId}/{startTime}/{duringTime}/{reason}/{type}

参数:

empId（路径参数，员工编号）

startTime（路径参数，请假开始时间）

duringTime（路径参数，请假时长）

reason（路径参数，请假理由）

type（路径参数，请假类型）

返回: 是否申请成功。

{

"status": "success",

"message": "申请成功"

}

假期审批接口

URL: POST /checkApplyInfo/{applyId}/{state}

参数:

applyId（路径参数，申请ID）

state（路径参数，审批状态，0为拒绝，1为批准）

返回: 审批结果。

{

"status": "success",

"message": "审批成功"

}

查询个人假期记录接口

URL: GET /getApplyRecordById/{empId}

参数:

empId（路径参数，员工编号）

返回: 员工的假期申请记录。

[

{

"applyId": "long",

"startTime": "string",

"duringTime": "integer",

"reason": "string",

"type": "string",

"state": "integer"

}

]

查询部门经理审批列表接口

URL: GET /getApplyInfoForEvent

返回: 部门经理审批的假期申请列表（只显示小于3天的假期）。

[

{

"applyId": "long",

"empId": "long",

"startTime": "string",

"duringTime": "integer",

"reason": "string",

"type": "string",

"state": "integer"

}

]

查询总经理审批列表接口

URL: GET /getApplyInfoForBig

返回: 总经理审批的假期申请列表（只显示大于等于3天的假期）。

[

{

"applyId": "long",

"empId": "long",

"startTime": "string",

"duringTime": "integer",

"reason": "string",

"type": "string",

"state": "integer"

}

]

查询正在申请中的假期记录接口

URL: GET /queryApplyingById/{empId}

参数: empId（路径参数，员工编号）

返回: 正在申请中的假期记录。

[

{

"applyId": "long",

"startTime": "string",

"duringTime": "integer",

"reason": "string",

"type": "string",

"state": "integer"

}

]

**员工管理模块**

新增员工接口

URL: POST /insertEmpInfo

请求体:

{

"empId": "long",

"account": "string",

"name": "string",

"password": "string",

"role": "string"

}

返回: 新增员工的结果。

{

"status": "success",

"message": "新增员工成功"

}

修改员工密码接口

URL: POST /updatePassword/{empId}/{password}

参数:

empId（路径参数，员工编号）

password（路径参数，新密码）

返回: 密码更新结果。

{

"status": "success",

"message": "密码更新成功"

}

3.2主要的业务逻辑和流程的具体实现。（部分给出流程图）[3][5]

（1）请假申请（applyVacation）

业务流程：如图3-1



图3-1

员工根据提供的 empId、startTime、duringTime、reason 和 type 发起请假申请。

系统根据传入的参数调用 ApplyForVacationService 的 applyForVacation 方法处理请假申请，保存相关信息到数据库。

返回布尔值，表示是否申请成功。

(2) 请假审批（checkApplyInfo）

业务流程：如图3-2



图3-2

领导根据提供的 applyId 和 state 审批员工的请假申请。

系统通过 applyForVacationService.updateApplyState() 更新审批状态，改变请假记录的状态。

返回布尔值，表示是否成功更新状态。

(3)查询员工请假记录（getApplyRecordById）

业务流程：如图3-3



图3-3

员工根据 empId 查询自己的请假记录。

系统调用 applyForVacationService.queryApplyByEmpId() 返回员工的所有请假记录。

(4)查询审批表信息（queryApplyByStateForEventManager 和 queryApplyByStateForBigManager）

业务流程：

部门经理和高层经理分别查看自己的审批表。

部门经理只查看小于3天的请假记录，而总经理及副总经理查看大于等于3天的请假记录。

系统调用相应的服务方法来查询符合条件的请假记录。

(5)查询正在申请的请假记录（queryApplyingById）

业务流程：如图3-4



图3-4

员工根据 empId 查询正在申请的请假记录。

系统调用 applyForVacationService.queryApplyingById() 返回正在申请的记录。

(6)员工打卡（checkInOrOut）

业务流程：如图3-5



图3-5



员工通过传入 empId 和 signType（打卡类型）进行打卡。

系统通过 checkInOutService.signInOrOut() 方法记录打卡信息。

(7) 查询打卡记录（queryAllRecords 和 queryRecordsByActOrType）

查询全部打卡记录或根据员工工号及打卡类型查询特定打卡记录。

系统通过 checkInOutService.querySignRecordByAccountOrType() 返回符合条件的打卡记录。

(8)查询剩余假期（getLeftTimeById）

业务流程：

员工根据 empId 查询剩余的假期时间。

系统通过 checkInOutService.queryLeftTimeByEmpId() 返回员工剩余的假期时间。

(9)员工信息管理（insertEmpInfo, updateEmpInfo）

新增或修改员工信息，处理员工密码加密，确保信息安全。

对于新增员工，通过 PasswordUtil 生成盐并加密密码，之后将员工信息保存至数据库。

对于修改员工信息，更新数据库中的员工数据。

(10)登录（login）

员工通过输入 account 和 password 登录系统。

系统通过 employerService.login() 验证用户名和密码是否匹配，并返回相应的登录信息。

3.3. 界面的具体实现

前端界面中普通员工主要实现以下功能：

**登录界面**：员工输入用户名和密码登录，登录成功后，返回用户信息。

图形用户界面, 网站

描述已自动生成

图3-6 登录界面

**请假申请界面**：员工通过界面（图3-7）查看剩余假期信息，并可以填写请假信息（3-8），点击提交按钮后，将数据传递给后端接口进行申请。

图形用户界面

描述已自动生成

图3-7 查看剩余假期

图形用户界面, 应用程序

描述已自动生成

图3-8 请假界面

**打卡界面**：员工选择打卡类型（上班打卡或下班打卡），并提交打卡请求。

图形用户界面, 网站

描述已自动生成

图3-9 打卡界面

**员工信息管理界面**：可以查看、修改个人信息，输入密码，并确保密码经过加密处理。

图形用户界面, 应用程序

描述已自动生成

图3-10 修改密码

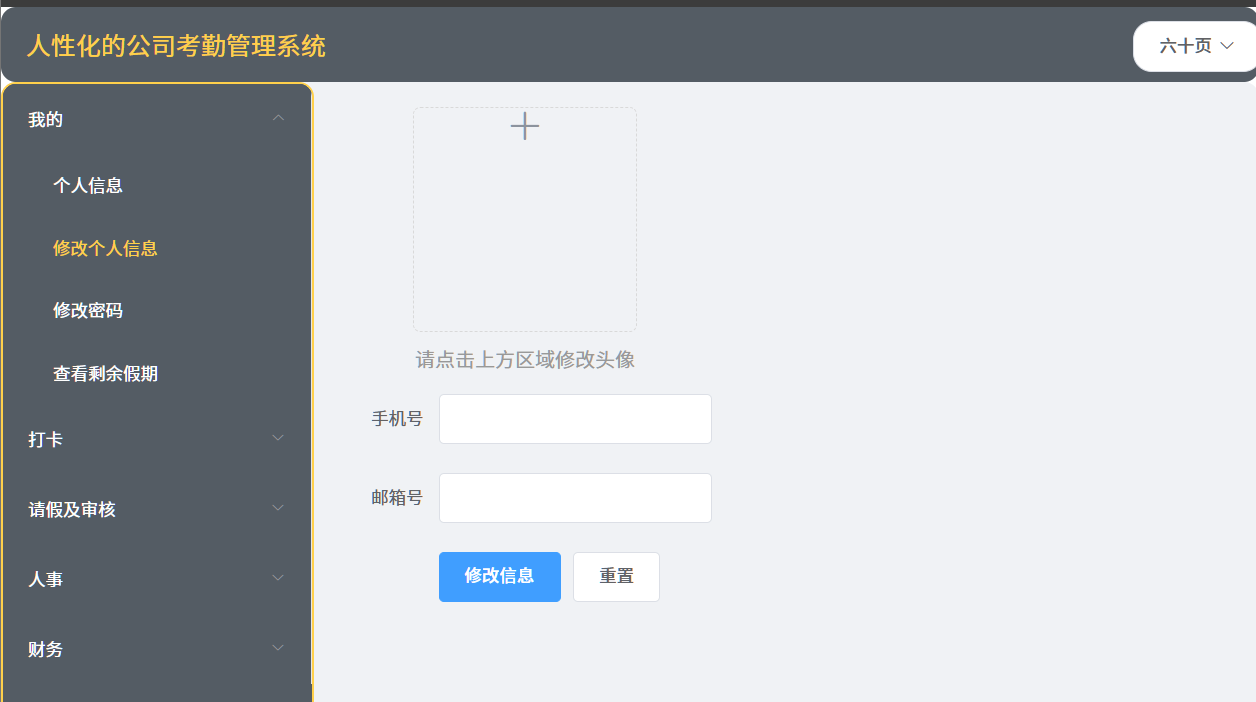


图3-11 修改个人信息

图形用户界面, 应用程序

描述已自动生成

图 3-12 显示个人信息

前端界面中管理员主要实现以下功能：

**请假审批界面**：管理员根据不同的审批需求（小于3天或大于等于3天），通过按钮选择是否批准，系统展示待审批的请假记录。



图3-13 审批界面

**添加新员工**：管理员可以添加新的员工

图形用户界面, 应用程序

描述已自动生成

图3-14

**查看所有员工状态**：管理员可以看到所有员工当前的状态

图形用户界面, 应用程序, 表格, Teams

描述已自动生成

图3-15

**查询员工打卡记录**：管理员可以根据员工工号和打卡类型查询打卡的时间

图形用户界面, 应用程序, Teams

描述已自动生成

图3-16

**查看请假记录表**：管理可以查看指定工号的所有请假记录

表格

低可信度描述已自动生成

图3-17

4.系统测试

4.1测试环境

使用语言：HTML,CSS,Vue2,JavaScript,Java1.8,MySQL8

编辑器：VS Code,IDEA

代码管理平台：github,地址[sureboom/egg](https://github.com/sureboom/egg)

4.2关键函数功能描述[2]

（1）用户相关函数

public ResponseEntity<Map<String,Object>login(Map<String, String> loginRequest)

实现用户登录功能，验证账号和密码，返回用户信息及状态。

public ResponseEntity<String> insertEmpInfo(Employer employer)

新增员工信息功能，包括对密码的加密处理和数据存储。

public boolean updatePassword(long empId, String password)

实现用户密码更新功能，更新后的密码会存储为加密格式。

public Employer queryEmpInfoById(long empId)

查询指定员工的详细信息。

public ResponseEntity<String> logout()

实现用户登出功能，清理用户的登录状态。

（2）考勤相关函数

public boolean checkInOrOut(long empId, int signType)

实现员工签到或签退功能，记录操作时间并防止重复操作。

public List<QueryCheckRecordResp>querySignRecordByAccountOrType(String account, Integer type)

根据工号和考勤类型查询员工的考勤记录。

public LeftVacation queryLeftTimeByEmpId(long empId)

查询指定员工的剩余假期时间。

（3）假期相关函数

public boolean applyForVacation(long empId, Date startTime, Integer duringTime, String reason, Integer type)

员工提交假期申请，并检查剩余假期时间是否足够。

public boolean updateApplyState(long applyId, Integer state)

更新假期申请的审批状态（如通过、拒绝）。

public List<QueryApplyRecordResp> queryApplyByEmpId(long empId)

查询员工的所有假期申请记录，包括申请时间和审批状态。

/\*（4）报表相关函数

public ResponseEntity<ByteArrayResource> generateAttendanceReport(String dateRange)

生成考勤报表，支持下载为 Excel 或 PDF 格式。

public ResponseEntity<ByteArrayResource> generateVacationReport(String dateRange)

生成假期统计报表，供管理员查看和分析。

public List<SummaryReport> generateSummaryReport()

汇总全体员工的考勤和假期数据，用于管理层的决策支持。\*/

4.3测试用例与测试结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试编号 | 测试内容 | 输入数据 | 预期结果 | 实际结果 | 备注 |
| TC1 | 请假申请功能测试 | empId=1, startTime=2024-12-20 09:00:00, duringTime=2, reason="休假", type=1 | 返回true表示申请成功 | true |  |
| TC2 | 请假审批功能测试 | applyId=1, state=1 (批准) | 返回true表示状态更新成功 | true |  |
| TC3 | 查询员工请假记录功能测试 | empid=1 | 返回员工的所有请假记录 |  |  |
| TC4 | 查询小于3天的请假记录功能测试 | (无输入参数) | 返回小于3天的请假记录列表 |  |  |
| TC5 | 查询大于等于3天的请假记录功能测试 | (无输入参数) | 返回大于3天的请假记录列表 |  |  |
| TC6 | 员工打卡功能测试 | empId=1, signType=1 (上班打卡) | 返回true表示打卡成功 | true |  |
| TC7 | 查询打卡记录功能测试 | account="001", type=1 | 返回对应的打卡记录 | 252 2024-12-05 15:28:45 |  |
| TC8 | 查询剩余假期功能测试 | empId=1 | 返回剩余假期时间（如15天） | 68 |  |
| TC9 | 员工信息更新功能测试 | empId=1, newName="张三", newPassword="newPass" | 返回true表示更新成功 | 返回true表示登录成功 |  |
| TC10 | 登录功能测试 | account="2021229033", password="password123" | 返回true表示登录成功 | 返回true表示登录成功 |  |

5.总结

5.1设计

本系统基本实现了最初的设计目标，各模块功能齐全，达到了预期的业务需求。主要功能如用户管理、考勤管理、假期管理等均成功开发并通过了全面测试。系统整体运行稳定，所需资源较少，界面简洁友好，响应迅速，使用加密密码服务，保护了用户的隐私，满足了用户的核心需求。但是因为未使用云服务器，所有未实现头像信息等修改与上传；生成报表功能因时间关系未能实现

5.2收获

通过本次系统开发，我在以下方面有了显著的收获：

**技术提升**：熟练掌握了 Spring Boot、Vue.js 等技术栈，并对 RESTful API 的设计和实现有了更深刻的理解。

**问题解决**：在开发过程中，解决了诸如数据库设计优化、接口性能提升等多个技术难题，积累了实战经验。

**用户体验**：通过用户调研和反馈迭代设计，不断优化了系统界面和操作流程，提升了用户满意度。

5.3不足

虽然系统功能基本完善，但仍存在一些不足之处：

**用户权限管理细化不足**：当前权限管理主要针对管理员和普通员工，未对更复杂的权限需求（如分级审批）做充分考虑。

**数据分析功能简单**：报表功能仅实现了基础的数据统计，未深入挖掘员工考勤和假期的行为模式，缺乏数据驱动的智能化分析。

**界面优化空间**：部分界面的响应式设计还需改进，以适配更多设备和分辨率。

**性能优化**：系统在高并发情况下性能表现良好，但对于极端情况下的压力测试仍需进一步优化。

5.4改进方向

如果有更多时间和资源，系统可在以下方面进一步提升：

**权限管理扩展**：引入基于角色的权限控制（RBAC），支持更细粒度的用户权限划分。

**智能化分析**：通过引入机器学习算法，对考勤和假期数据进行深度分析，提供智能化决策支持。

**移动端支持**：开发移动端应用，增强员工的便捷性和体验感。

**性能优化**：采用缓存技术和异步处理机制，进一步优化接口性能，提升系统的高并发处理能力。

**功能扩展**：增加实时提醒功能，如考勤异常提醒等，进一步提升系统实用性。

6.参考文献

[1]毛新军.软件工程[M].北京:高等教育出版社,2020:103.

[2] [Spring Boot - 快速入门 | Spring Boot 教程](https://www.w3ccoo.com/spring_boot/spring_boot_quick_start.html)

[3] Spring Boot 2.X https://www.didispace.com/spring-boot-2/1-2-quick-start.html

[4] [Vue 实例 — Vue.js](https://v2.cn.vuejs.org/v2/guide/instance.html)

[5] [指南 | Element](https://element.eleme.cn/" \l "/zh-CN/guide/design)

7.致谢

在本系统的开发与实施过程中，得到了许多人的帮助与支持，在此我们深表感谢。

首先，感谢指导老师***陈新疆***，在项目设计、开发和测试的每个阶段，给予了我们悉心的指导和宝贵的建议，让我们能够克服难点，顺利完成任务。

同时，感谢使用和测试本系统的用户，您们的反馈和建议是我们不断优化系统的重要依据。

最后，我们还要感谢提供支持的所有资源，包括相关技术文档、开发工具和平台，以及各类网络资源，为我们的开发提供了有力保障。

本系统的顺利完成，离不开每一位支持与帮助过我们的人的付出与努力，再次表示衷心的感谢！