**考勤系统**

****

**软件设计报告**

**学 院 未来技术学院**

**专 业 计算机技术**

**组 员 毕凯斌**

**组 员 魏金旭**

**组 员 李璟昕**

**组 员 贾智云**

**组 员 冯昱中**

**年 级 2023级**

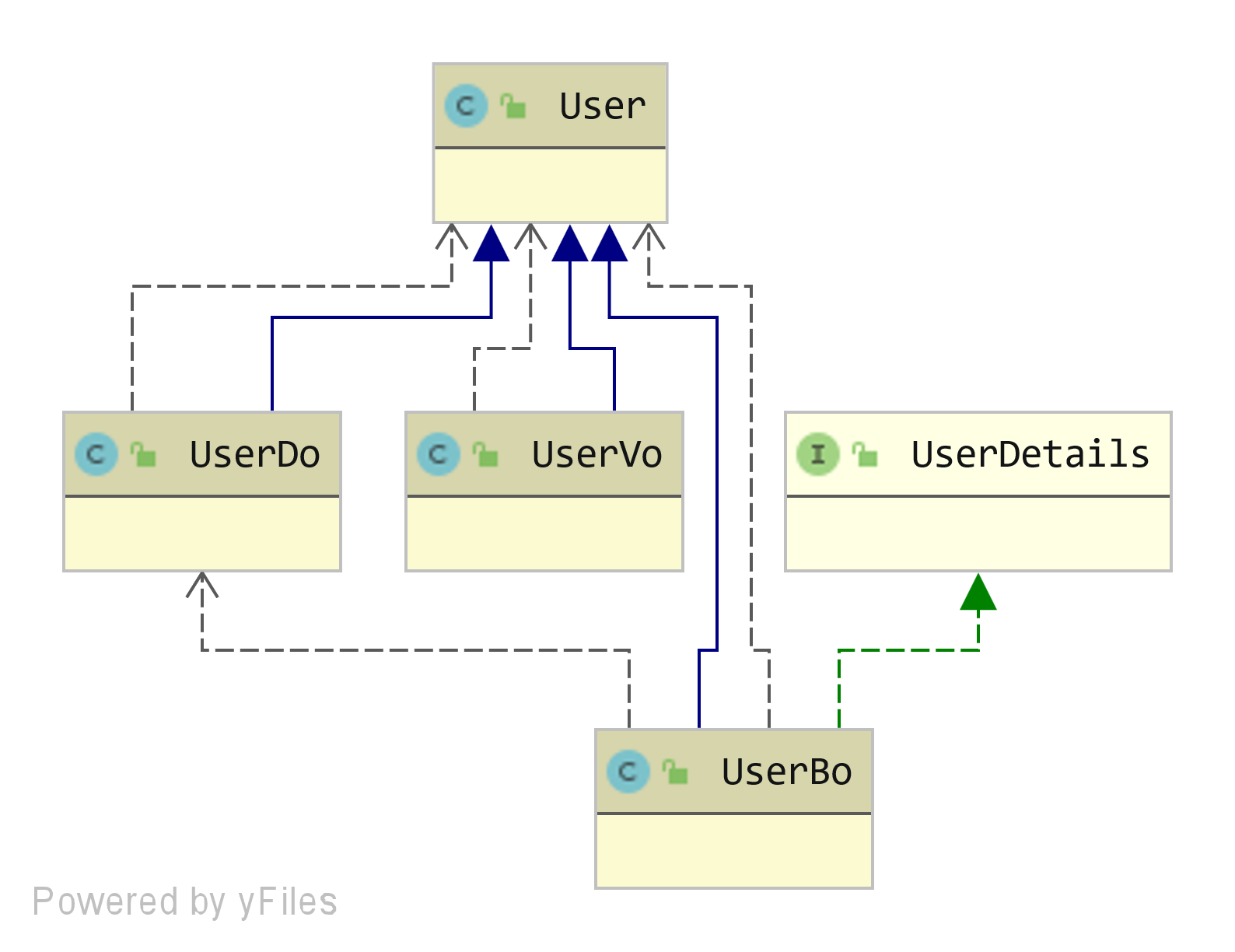
# 服务器

本服务器拟采用java开发，使用Spring框架，使用MySQL数据库。Java版本：1.8，SpringBoot版本：2.4.0。本服务器还使用了一些第三方库，详见pom.xml。本服务器仅支持使用json登录和使用json注销，使用token来验证身份，未正常注销可能影响下一次登录。

## 安全模块

安全模块包括com.example.demo.data.user包和com.example.demo.webconfig.securityconfig包。与登录有关的类位于securityconfig中的loginconfig包，与注销有关的类位于securityconfig中的logoutconfig包，与权限验证有关的类位于securityconfig中的authenticationconfig包。

### user包



User类中定义了用户的id、名字、部门id、部门名字、拥有的权限（员工、部门经理、总经理）、用户类别（普通员工、人事部员工、管理员）。用户的权限和类别都设为整数常量，方便进行权限验证。User类的构造函数的可见性为protected。User类的每个成员变量均提供对应的get、set方法。在设置User的权限、类型、部门名字、用户名字时会检测输入合法性，若输入不合法会抛出异常，同时会使用logger记录。

#### UserVo类

UserVo类继承自User类。

#### UserBo类

UserBo类继承自User类，实现UserDetails接口。UserBo类是实现权限验证的重要部分。可直接将UserDo转换为UserBo。UserBo记录用户拥有的权限集合，增加了新的成员变量：用户密码，维护一个映射关系来记录用户ID和该用户的token对应的key。UserBo类提供登录用户、检测用户是否已登录、注销用户、获得用户的token对应的key等方法。UserBo类提供用户密码的get、set方法，若用户密码不合法，则会抛出异常，同时会使用logger记录。UserBo还记录了每个用户使用过的密文密码，若用户尝试使用已用过的密文密码登录，则会被拒绝。

#### UserDo类

UserDo类继承自User类，增加了新的成员变量：用户密码。UserDo类提供用户密码的get、set密码，此处的set方法不检测输入合法性（默认数据库中存储的是符合要求的密码）。

### securityconfig包

本服务器的安全配置采用全局配置。

#### UserDetailServiceImpl类

UserDetailServiceImpl类实现UserDetailsService接口，重写loadUserByUsername方法。服务器会从数据库中加载用户，若找不到用户则会抛异常，logger会做相应记录。

#### UserGrantedAuthority类

UserGrantedAuthority类实现GrantedAuthority、Comparable<UserGrantedAuthority>接口。UserGrantedAuthority类提供两种构造函数，参数为权限，类型为int或String，若参数非法则会抛出异常，logger会做相应记录。重载了compareTo方法，便于比较权限。若比较的权限类型不同也会抛出异常，logger会做相应记录。

#### AuthenticationCompareAttribute类

AuthenticationCompareAttribute类实现ConfigAttribute接口，参数为String，内容为>，>=，=，<，<=，若参数非法则会抛出异常，logger会做相应记录。此类定义了权限验证时的操作，即如何比较权限。

#### DecisionAttribute类

DecisionAttribute类实现ConfigAttribute接口，参数为String，内容为“AND”，“OR”，“DYNAMIC” ，若参数非法则会抛出异常，logger会做相应记录。此类定义了权限验证时的判断逻辑。“AND”表示执行“与”操作，即每个权限都要满足，“OR”表示执行“或”操作，即只要满足其中一个权限即可，“DYNAMIC”为动态判断，仅用于“审批”（/approval）部分的权限判断。

#### UrlPathFilterInvocationSecurityMetadataSource类

UrlPathFilterInvocationSecurityMetadataSource类实现FilterInvocationSecurityMetadataSource接口。这个类用于读取网址的权限设置（读取resources文件夹中的urlAuthority.settings，详细说明见部署使用文档），并建立网址和权限之间的映射关系，可按网址返回权限集合。

#### UrlAccessDecisionManager类

UrlAccessDecisionManager类实现AccessDecisionManager接口，可根据对应的url来判断当前用户是否拥有充足的权限。判断权限的策略根据DecisionAttribute类和AuthenticationCompareAttribute类的内容来选择。若输入非法会抛出异常，或用户权限不足，logger会进行相应记录。

#### AuthenticationTokenFilter类

AuthenticationTokenFilter类继承自OncePerRequestFilter类。此类会检测用户发送的token是否合法，logger会记录token的检测结果。若token合法，则设该用户为已登录。

#### AuthenticationAccessDeniedHandler类

AuthenticationAccessDeniedHandler实现AccessDeniedHandler接口。当用户权限不足时，Spring框架会调用此类。此类会向前端发送一个json，告知前端权限不足。

#### JsonLoginAuthenticationFilter类

JsonLoginAuthenticationFilter类继承自UsernamePasswordAuthenticationFilter类。此类用于从json串中提取用户名和密码信息，并将该信息传递给AuthenticationManager类。

#### LoginAuthenticationProvider类

LoginAuthenticationProvider类实现AuthenticationProvider接口。此类用于检查用户名和密码是否匹配，以及该用户是否存在。若该用户的密码正确且使用未登录过的密文密码才可登录，否则无法登录，logger会记录每个用户的登录操作。

#### PasswordOverusedException类

PasswordOverusedException类继承自AuthenticationException类。此异常在用户使用登录过的密码时抛出。

#### LoginSuccessAuthenticationHandler类

LoginSuccessAuthenticationHandler类实现AuthenticationSuccessHandler接口。此类在用户登录成功时被调用，会返回给前端一个json，里面记录了登录状态和一个token。同时将登录信息记录到UserBo中。

#### LoginFailureAuthenticationHandler类

LoginFailureAuthenticationHandler类实现AuthenticationFailureHandler接口。此类在用户登录失败时被调用，会返回给前端一个json，里面记录了登录状态。

#### JsonLogoutHandler类

JsonLogoutHandler类继承自SecurityContextLogoutHandler类。此类用于解析注销时前端发送的json。若用户名和密码正确，且该用户处于登录状态，则可注销，否则失败。logger会记录相应结果。

#### MyLogoutSuccessHandler类

MyLogoutSuccessHandler实现LogoutSuccessHandler接口。此类在用户注销成功时被调用，会返回给前端一个json，里面记录了注销状态。

#### NotLoggedInException类

NotLoggedInException类继承自AuthenticationException类。此异常在未登录的用户尝试注销时抛出。

## Web配置

Web配置模块位于com.example.demo.webconfig包中

#### WebSecurityConfig类

WebSecurityConfig类继承自WebSecurityConfigurerAdapter类，里面有服务器的所有安全配置。由于采用token来验证已登录用户，故禁用CSRF。处于方便考虑，对于所有请求都启用CORS。本服务器支持Spring自带的网页登录，也支持使用json登录。本服务器支持使用json注销。对于登录和注销页面不检查登录状态和权限，其余页面均检查。

#### CorsConfig类

CorsConfig类实现WebMvcConfigurer接口，里面有服务器的所有关于CORS的配置。

#### HttpServletRequestReplacedFilter类

HttpServletRequestReplacedFilter类实现Filter接口。此类用于解决HttpServletRequest的getInputStream()方法不能被多次调用的问题。

#### BodyReaderHttpServletRequestWrapper类

BodyReaderHttpServletRequestWrapper类继承自HttpServletRequestWrapper，用于将HttpServletRequest中的InputStream保存至本地，便于重复读取。支持暂存二进制形式的数据。

#### RequestHolder类

RequestHolder类实现Filter接口，用于保存FilterChain中使用过的HttpServletRequest，用于解决@Autowired获取的HttpServletRequest可能为未经BodyReaderHttpServletRequestWrapper处理的HttpServletRequest。

#### DispatcherServlet类

DispatcherServlet类继承自org.springframework.web.servlet.DispatcherServlet，重载了service()方法，用于将Servlet中使用的HttpServletRequest替换为经BodyReaderHttpServletRequestWrapper处理后的HttpServletRequest。

#### UploadConfig类

UploadConfig类注册了Bean “multipartResolver”，解决了替换Servlet后无法上传文件的问题。

## API设计

#### 协议

HTTP

#### 域名

[http://localhost:8081/AttendanceSystem/](http://localhost:8080/AttendanceSystem/)

#### 路径

http://localhost:8081/AttendanceSystem/employee

http://localhost:8081/AttendanceSystem/leave

http://localhost:8081/AttendanceSystem/approval

http://localhost:8081/AttendanceSystem/holidays

http://localhost:8081/AttendanceSystem/attendance

#### HTTP动词

|  |  |  |
| --- | --- | --- |
| **http动词** | **路径** | **解释** |
| GET | /employee | 获得所有员工的员工信息 |
| GET | /employee/id | 获得某一员工的员工信息 |
| POST | /employee | 添加某一员工员工信息 |
| PUT | /employee | 修改某一员工的员工信息 |
| DELETE | /employee | 删除某一员工的员工信息 |
| GET | /holidays | 获取所有员工假期信息 |
| GET | /holidays/id | 获取某一员工假期信息 |
| POST | /holidays | 添加某一员工假期信息 |
| PUT | /holidays | 修改某一员工假期信息 |
| DELETE | /holidays | 删除某一员工假期信息 |
| GET | /leave | 获得所有已提交的请假信息 |
| GET | /leave/id | 获得某一员工的所有请假申请或某一条请假申请(取决于id的填写) |
| POST | /leave | 添加请假申请 |
| PUT | /leave | 修改请假申请 |
| DELETE | /leave | 删除请假申请 |
| GET | /attendance | 获取所有打卡记录 |
| GET | /attendance/id | 获取某一员工所有的打卡记录 |
| POST | /attendance | 导入打卡记录，打卡记录的格式见附录 |
| DELETE | /attendance | 删除某一打卡记录 |
| GET | /approval | 获得所有请假审批 |
| GET | /approval/id | 获得某一审批人的所有审批或某一条请假申请的审批 |
| POST | /approval | 添加某一条请假申请的审批 |
| PUT | /approval | 修改已添加的审批 |
| DELETE | /approval | 删除某一条审批 |
| GET | /employee/report | 导出考勤报表 |

#### MVC架构设计

* + - 1. **ApprovalService类**

本类为Approval API的Service层，实现审批的查询、提交、修改与删除。

前端申请查询审批后，此类调用相关repository层的方法返回前端所查询的数据。

前端申请提交审批后，此类将从前端获取审批结果，从相关repository层获取员工假期信息与该审批对应的请假申请，并判断该审批是否合法。审批为合法时将其存入数据库。

前端申请修改审批后，此类将从前端获取更改后的审批结果，从相关repository层获取更改前的审批结果与该审批对应的请假申请，从而对数据库的Leave表、Holidays表进行相应更改。

前端申请删除审批后，此类将从相关repository层获取所删除的审批的前一条审批，并对数据库的Leave表、Holidays表进行相应更改。

* + - 1. **EmployeeService类**

本类为Employee API的Service层，实现员工信息的查询、添加、修改与删除。

前端申请添加员工信息后，此类将通过前端所提供的员工信息建立对应的员工假期信息实例，并将员工信息与假期信息添加至数据库的Employee表与Leave表

前端申请删除员工信息后，此类将同时删除所有与该Employee相关的信息

* + - 1. **HolidaysService类**

本类为Holidays API的Service层，仅实现员工假期信息

* + - 1. **LeaveService类**

本类为Leave API的Service层，实现员工请假申请的查询、添加、删除与修改。

前端申请查询员工请假申请后，此类将通过前端所提供的员工empID查询员工权限，并返回相应的数据

前端申请添加员工请假申请后，此类将通过前端所提供的请假申请信息并从相应repository层获取员工假期信息，通过检测员工假期是否大于所申请的天数，判断是否驳回此申请，若无异常，则将其存入数据库

前端申请修改员工请假申请后，此类将进行与添加员工申请相同的合法性判断，若无异常，将修改数据库数据。

基本框架如下图



## 数据库设计

#### employee表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 说明 | 数据类型 | 主外键 |
| empId | 员工编号 | bigint | PRIMARY KEY |
| empName | 员工姓名 | varchar |  |
| birthday | 员工生日 | Date |  |
| getJobDay | 员工入职时间 | Date |  |
| gender | 员工性别 | Varchar |  |
| deptId | 员工部门编号 | bigint |  |
| deptName | 员工部门名称 | varchar |  |
| auth | 员工权限 | int |  |

#### approval表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 说明 | 数据类型 | 主外键 |
| id | 自增id | int | PRIMARY KEY |
| empId | 审批人员工编号 | bigint |  |
| appId | 审批的请假申请编号 | bigint | FOREIGN KEY |
| time | 审批时间 | date |  |
| approvalType | 审批状态 | int |  |
| comment | 审批人评语 | varchar |  |
| approval | 审批结果 | tinyint |  |

#### holidays表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 说明 | 数据类型 | 主外键 |
| empId | 员工编号 | bigint | PRIMARY KEY、FOREIGN KEY |
| yearHoliday | 员工剩余年假天数 | int |  |
| privateHolidays | 员工已请事假天数 | int |  |
| sickHolidays | 员工剩余病假天数 | int |  |

#### leave表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 说明 | 数据类型 | 主外键 |
| empId | 员工编号 | bigint | PRIMARY KEY、  FOREIGN KEY |
| appId | 请假申请编号 | bigint | PRIMARY KEY |
| empName | 员工姓名 | varchar |  |
| empDeptId | 员工部门编号 | bigint |  |
| empDeptName | 员工部门名称 | varchar |  |
| reason | 请假原因 | varchar |  |
| leaveType | 请假种别 | int |  |
| state | 请假申请状态 | int |  |
| startTime | 假期开始时间 | date |  |
| endTime | 假期结束时间 | date |  |

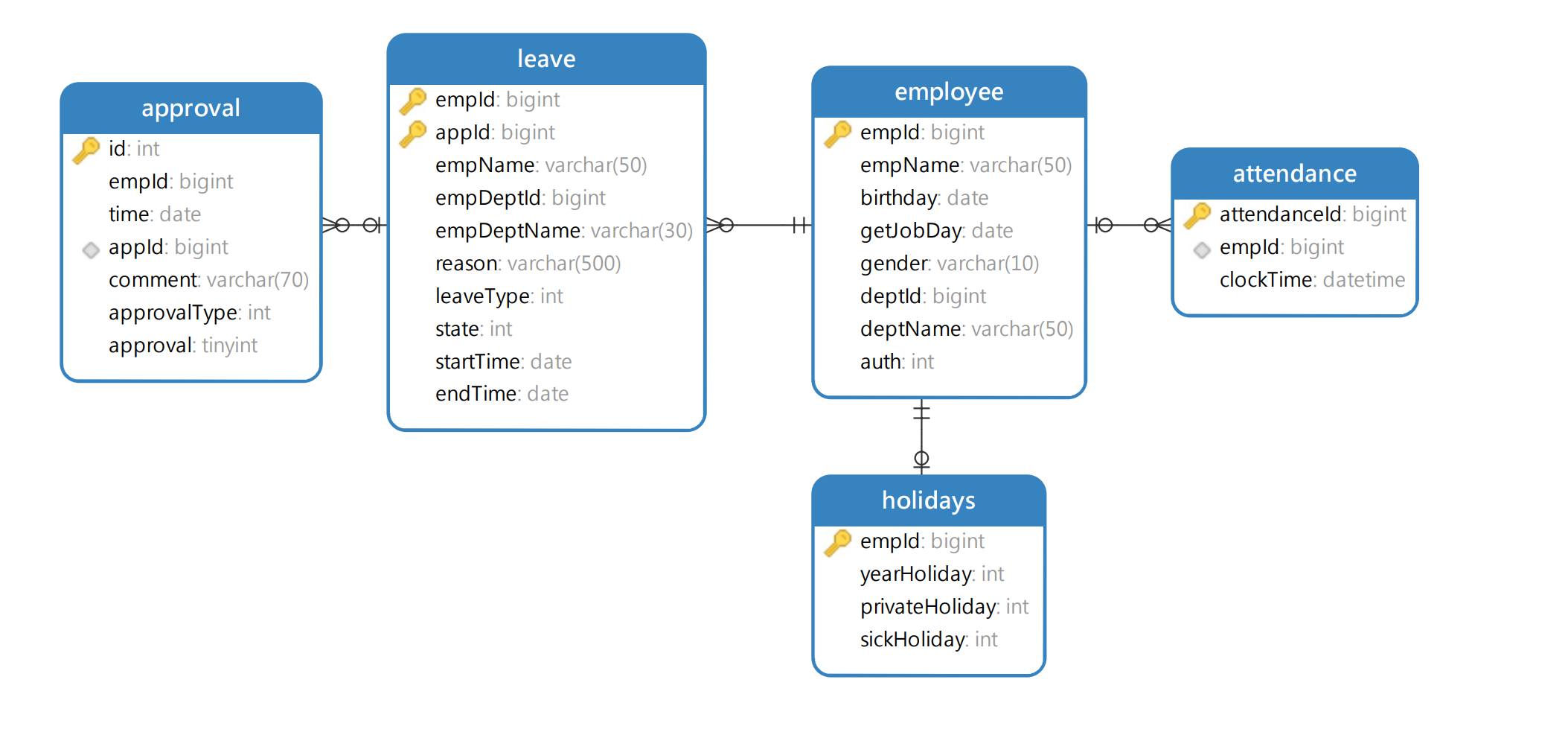
#### attendance表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 说明 | 数据类型 | 主外键 |
| attendanceId | 打卡编号 | bigint | PRIMARY KEY |
| appId | 请假申请编号 | bigint | PRIMARY KEY、FOREIGN KEY |
| clockTime | 打卡时间 | datetime |  |

#### user表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 说明 | 数据类型 | 主外键 |
| id | 用户编号（同员工编号） | bigint | PRIMARY KEY  FOREIGN KEY |
| name | 用户姓名 | varchar |  |
| auth | 员工级别 | int |  |
| userType | 员工类别 | int |  |
| password | 用户密码 | varchar |  |

#### 数据库ER图



#### 相关SQL语句

create table user

(

id BIGINT,

name VARCHAR(30),

auth INTEGER,

userType INTEGER,

password VARCHAR(30),

primary key (id)

foreign key (id) references employee(empId) on update cascade on delete cascade

);

create table employee(

empId bigint primary key ,

empName varchar(50),

birthday date,

getJobDay date,

gender varchar(10),

deptId bigint,

deptName varchar(50),

auth int default(0)

);

create table attendance (

attendanceId bigint primary key ,

empId bigint,

clockTime datetime,

foreign key (empId) references employee(empId)

on update cascade on delete cascade

);

create table `leave`(

empId bigint,

appId bigint,

empName varchar(50),

empDeptId bigint,

empDeptName varchar(30),

reason varchar(500),

leaveType int,

state int default(1),

startTime date,

endTime date,

index (appId),

primary key (empId, appId),

foreign key (empId) references employee(empId)

on update cascade on delete cascade

);

create table approval(

id int AUTO\_INCREMENT PRIMARY KEY,

empId bigint,

time date,

appId bigint,

comment varchar(70),

approvalType int,

approval tinyint,

foreign key (appId) references `leave`(appId)

on update cascade on delete cascade

);

create table holidays(

empId bigint primary key ,

yearHoliday int,

privateHoliday int,

sickHoliday int default(5),

foreign key (empId) references employee(empId)

);

# 附录

## 打卡记录

### 格式

[打卡记录ID],[员工编号],[日期]

一行为一条记录

### 样例

1,1,1999-10-1

2,1,1999-10-2

3,2,1999-10-2