**校园通行码预约管理系统**

**技术报告**

**2025年6月**

目录

[第一章 系统概述 5](#_Toc201776623)

[1.1 系统背景与目标 5](#_Toc201776624)

[1.2 主要工作内容 5](#_Toc201776625)

[第二章 系统框架 6](#_Toc201776626)

[2.1 网络架构 6](#_Toc201776627)

[2.2 系统架构 7](#_Toc201776628)

[2.3 系统功能 8](#_Toc201776629)

[第三章 系统需求分析 8](#_Toc201776630)

[3.1 用户端功能模块 8](#_Toc201776631)

[3.1.1 预约申请功能 8](#_Toc201776632)

[3.1.2 预约查询功能 9](#_Toc201776633)

[3.1.3 通行码展示功能 9](#_Toc201776634)

[3.2 管理端功能模块 10](#_Toc201776635)

[3.2.1 管理员登录功能 10](#_Toc201776636)

[3.2.2 预约管理功能 10](#_Toc201776637)

[3.2.3 管理员管理功能 11](#_Toc201776638)

[3.2.4 部门管理功能 11](#_Toc201776639)

[3.2.5 统计报表功能 12](#_Toc201776640)

[3.2.6 审计日志功能 12](#_Toc201776641)

[3.2.7 用户管理功能 13](#_Toc201776642)

[3.3 系统安全功能模块 13](#_Toc201776643)

[3.3.1 数据加密功能 13](#_Toc201776644)

[3.3.2 权限控制功能 14](#_Toc201776645)

[第四章 系统详细设计 14](#_Toc201776646)

[4.1 Beans 14](#_Toc201776647)

[4.1.1 Department 14](#_Toc201776648)

[4.1.2 Admin 17](#_Toc201776649)

[4.1.3 Appointment 26](#_Toc201776650)

[4.1.4 AccompanyingPerson 39](#_Toc201776651)

[4.1.5 AuditLog 43](#_Toc201776652)

[4.2 Dao 49](#_Toc201776653)

[4.2.1 AdminDao 49](#_Toc201776654)

[4.2.2 AppointmentDao 54](#_Toc201776655)

[4.2.3 AuditLogDao 64](#_Toc201776656)

[4.2.4 DepartmentDao 68](#_Toc201776657)

[4.2.5 BaseDao 72](#_Toc201776658)

[4.3 Controller 75](#_Toc201776659)

[4.3.1 AdminAppointmentsServlet 75](#_Toc201776660)

[4.3.2 AdminAuditLogsServlet 76](#_Toc201776661)

[4.3.3 AdminDepartmentsServlet 78](#_Toc201776662)

[4.3.4 AdminLoginServlet 80](#_Toc201776663)

[4.3.5 AdminLogoutServlet 82](#_Toc201776664)

[4.3.6 AdminManageServlet 84](#_Toc201776665)

[4.3.7 AdminReportsServlet 89](#_Toc201776666)

[4.3.8 AdminUsersServlet 91](#_Toc201776667)

[4.3.9 AppointmentApplyServlet 95](#_Toc201776668)

[4.3.10 AppointmentQueryServlet 98](#_Toc201776669)

[4.3.11 DepartmentOperationServlet 99](#_Toc201776670)

[4.3.12 PassCodeViewServlet 103](#_Toc201776671)

[4.3.13 QRCodeServlet 106](#_Toc201776672)

[4.4 Util 108](#_Toc201776673)

[4.4.1 CryptoUtil 108](#_Toc201776674)

[第五章 数据库设计 112](#_Toc201776675)

[5.1 数据库表概念结构设计 112](#_Toc201776676)

[5.1.1 部门（Department） 112](#_Toc201776677)

[5.1.2 管理员（Admin） 113](#_Toc201776678)

[5.1.3 预约记录（Appointment） 114](#_Toc201776679)

[5.1.4 随行人员（accompanying\_person） 115](#_Toc201776680)

[5.1.5 审计日志表（Audit\_Log） 115](#_Toc201776681)

[5.2 数据库表概念关系模型 115](#_Toc201776682)

[5.2.1 ER图 115](#_Toc201776683)

[第六章 系统实现功能 117](#_Toc201776684)

[6.1 用户端功能模块 117](#_Toc201776685)

[6.1.1 首页功能模块 118](#_Toc201776686)

[6.1.2 预约申请功能模块 118](#_Toc201776687)

[6.1.3 通行码功能模块 118](#_Toc201776688)

[6.1.4 二维码功能模块 118](#_Toc201776689)

[6.2 管理端功能模块 119](#_Toc201776690)

[6.2.1 管理员登录功能模块 120](#_Toc201776691)

[6.2.2 管理员控制台功能模块 120](#_Toc201776692)

[6.2.3 预约管理功能模块 120](#_Toc201776693)

[6.2.4 部门管理功能模块 120](#_Toc201776694)

[6.2.5 管理员管理功能模块 120](#_Toc201776695)

[6.2.6 统计报告功能模块 120](#_Toc201776696)

[6.2.7 操作审计功能模块 120](#_Toc201776697)

[6.3 系统安全功能模块 121](#_Toc201776698)

[6.3.1 国密算法加密模块 121](#_Toc201776699)

[6.3.2 密码安全策略模块 121](#_Toc201776700)

[6.3.3 账户安全控制模块 122](#_Toc201776701)

[6.3.4 安全审计日志模块 122](#_Toc201776702)

[6.3.5 访问控制权限模块 122](#_Toc201776703)

[第七章 程序清单 122](#_Toc201776704)

# 系统概述

## 系统背景与目标

随着疫情防控政策的调整，各大高校逐步开放校园，允许校外人员入校访问。为规范校园通行管理，本系统旨在设计并实现一套基于 MVC 设计模式的校园通行码预约管理系统，满足公务预约和社会公众预约需求，实现通行码的自动化审核与管理，提升校园安全管理效率。

## 主要工作内容

1. 基于 Servlet、JSP、JavaBean、JDBC 等技术，结合 DAO 模式实现系统架构设计。
2. 开发手机端（预约申请、通行码查看、历史记录查询）和后台管理端（管理员管理、部门管理、预约审核与操作审计）功能。
3. 集成国密算法 SM3、SM2/SM4 实现数据加密与脱敏，满足等保三级安全要求。
4. 设计二维码生成模块，实现通行码的有效性状态显示。并对于有效与无效通行码给予相关显示。

# 系统框架

## 网络架构

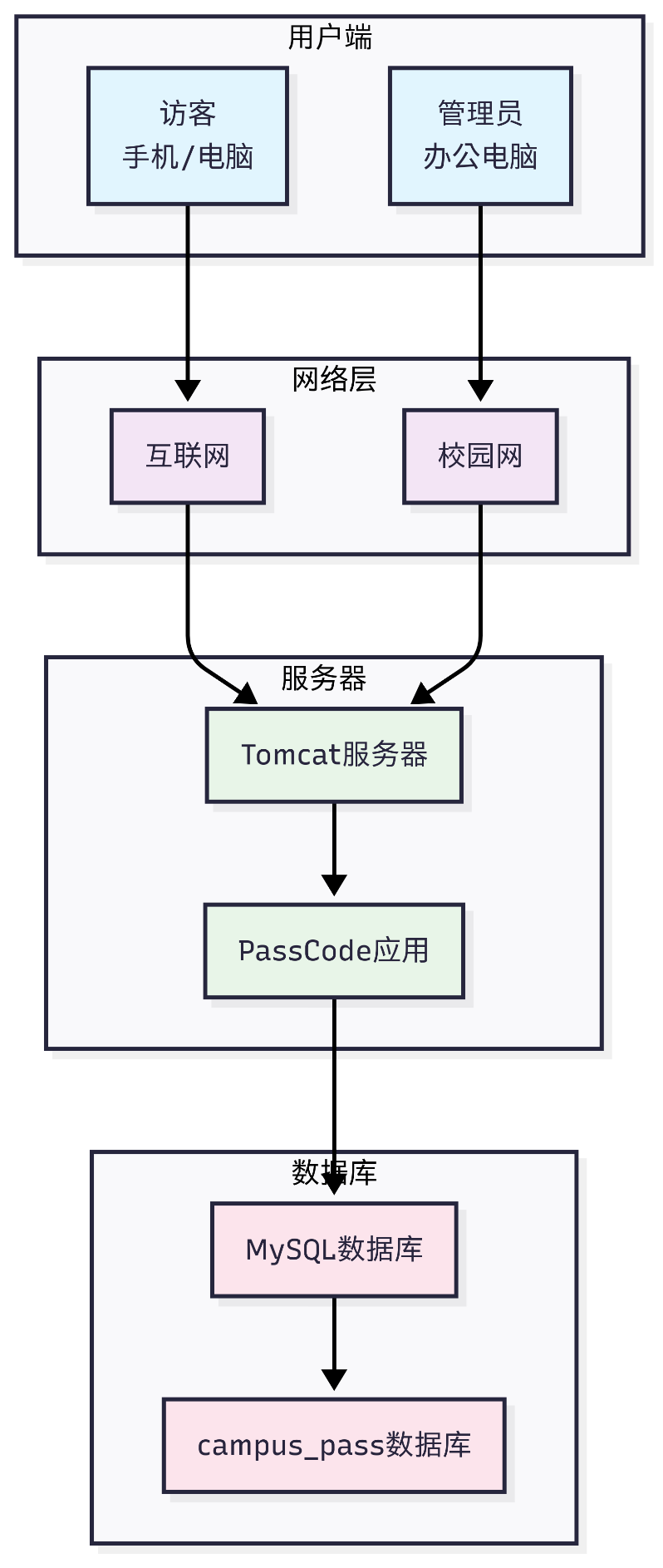


图 1 网络架构

## 系统架构

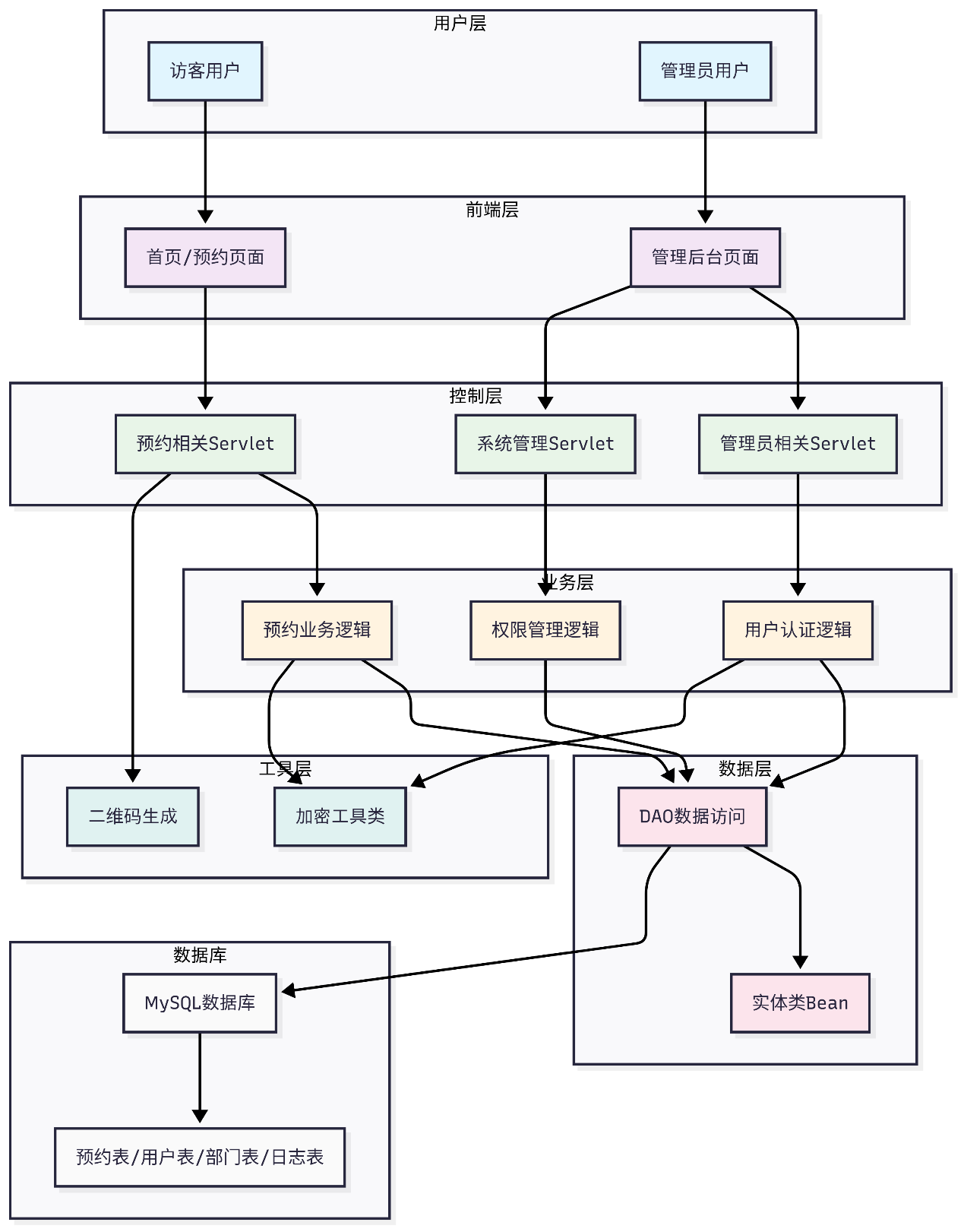


图 2 系统架构

## 系统功能

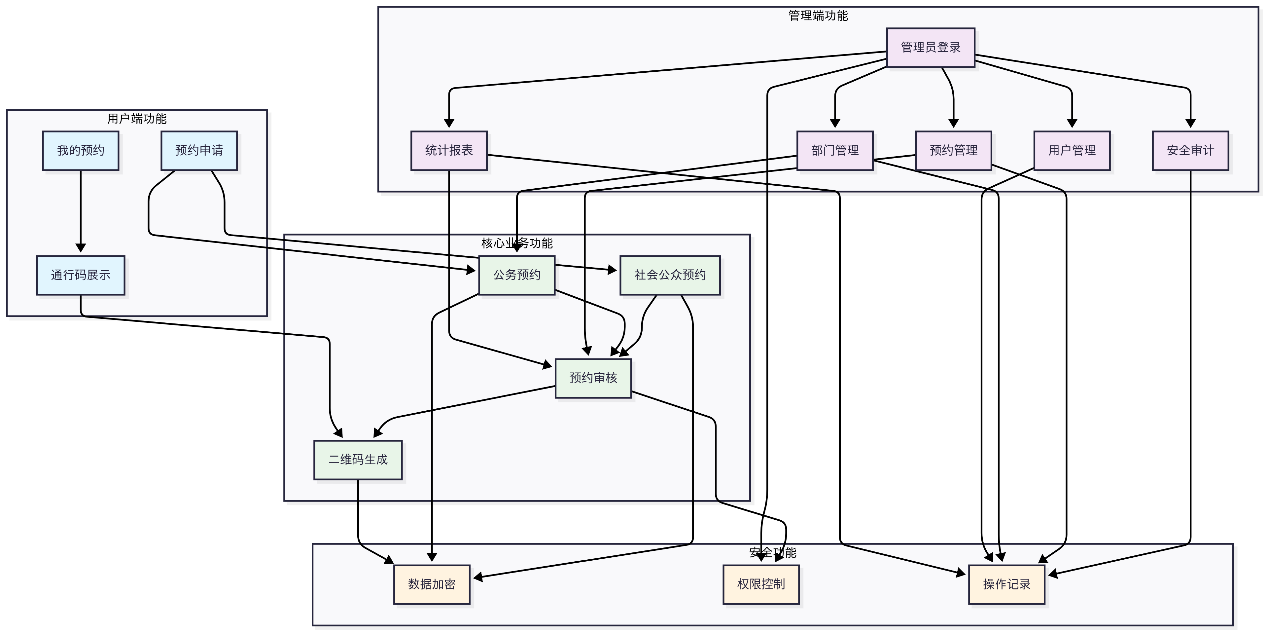


图 3 系统功能

# 系统需求分析

## 用户端功能模块

### 预约申请功能

#### 功能描述

用户填写预约申请表单，包括访客基本信息、预约信息、预约类型等，提交后系统自动审核或等待部门审核，成功后生成通行码。

#### 业务处理流程及要求

（1）用户必须填写所有必填字段：访客姓名、手机号、身份证号、校区、进校时间、预约类型；

（2）手机号必须符合中国大陆手机号格式（1开头的11位数字）；

（3）身份证号必须符合18位身份证号格式；

（4）进校时间不能是过去的日期；

（5）公务预约需要选择访问部门和填写事由；

（6）系统对敏感信息（手机号、身份证号）进行SM4加密存储；

（7）预约提交后记录审计日志。

#### 输入信息

访客姓名、手机号、身份证号、所在单位、校区、进校时间、交通方式、车牌号、预约类型（社会公众/公务）、公务访问部门、公务联系人、公务事由、随行人员信息。

#### 输出信息

若输入错误则显示错误提示信息，若提交成功则跳转到通行码展示页面，显示预约ID、访客姓名、进校时间等信息。

### 预约查询功能

#### 功能描述

用户通过访客姓名、身份证号、手机号查询历史预约记录，查看预约状态和详细信息。

#### 业务处理流程及要求

（1）用户必须提供访客姓名、身份证号、手机号中的至少两项信息；

（2）系统对输入的身份证号和手机号进行SM4加密后与数据库中的加密数据比对；

（3）查询结果按申请时间倒序排列；

（4）显示预约的完整信息，包括状态、审核结果等。

#### 输入信息

访客姓名、身份证号、手机号。

#### 输出信息

显示符合条件的预约记录列表，包括预约ID、访客姓名、预约类型、进校时间、校区、状态、申请时间等信息。

### 通行码展示功能

#### 功能描述

#### 系统根据预约信息生成动态二维码通行码，供访客向门卫出示验证身份。

#### 业务处理流程及要求

（1）系统根据预约ID生成唯一的二维码；

（2）二维码包含预约的基本信息（ID、姓名、时间等）；

（3）根据预约状态显示不同的通行码状态（有效/无效）；

（4）支持二维码刷新和重新生成。

#### 输入信息

预约ID。

#### 输出信息

显示访客姓名、进校时间、校区、预约ID，以及对应的二维码通行码。

## 管理端功能模块

### 管理员登录功能

#### 功能描述

管理员输入登录名和密码进行身份验证，验证成功后进入管理后台。

#### 业务处理流程及要求

（1）登录名和密码不能为空；

（2）密码必须符合复杂度要求（8位以上，包含数字、大小写字母、特殊字符）；

（3）系统检查账户是否被锁定（连续5次失败登录锁定30分钟）；

（4）密码使用SM3哈希算法验证；

（5）登录成功后记录审计日志，失败时增加失败次数；

（6）会话超时时间为30分钟。

#### 输入信息

#### 登录名、密码。

#### 输出信息

若登录失败则显示错误提示，若登录成功则跳转到管理后台首页。

### 预约管理功能

#### 功能描述

管理员查看、审核和管理访客预约申请，包括通过、拒绝预约操作。

#### 业务处理流程及要求

（1）系统管理员和学校管理员可查看所有预约；

（2） 部门管理员只能查看本部门的公务预约，根据权限可能查看社会公众预约；

（3）审核操作需要记录审核人、审核时间；

（4）所有操作记录审计日志。

#### 输入信息

筛选条件（申请日期、预约日期、校区、访客单位、访客姓名、身份证号、公务部门、公务联系人、状态）、审核操作（通过/拒绝）。

#### 输出信息

显示预约列表，包括ID、访客姓名、预约类型、进校时间、校区、状态、申请时间等信息，以及审核操作按钮。

### 管理员管理功能

#### 功能描述

系统管理员和学校管理员可以添加、编辑、删除管理员账户，设置管理员权限。

#### 业务处理流程及要求

（1）只有系统管理员和学校管理员有权限管理其他管理员；

（2）添加管理员时必须设置登录名、密码、姓名、部门、角色等基本信息；

（3）密码必须符合复杂度要求；

（4）支持设置管理员权限（能否管理社会公众预约、能否统计社会公众预约）；

（5）所有操作记录审计日志。

#### 输入信息

管理员基本信息（登录名、密码、姓名、部门、手机号、角色）、权限设置。

#### 输出信息

显示管理员列表，包括ID、登录名、姓名、部门、角色、权限等信息，以及添加、编辑、删除操作按钮。

### 部门管理功能

#### 功能描述

管理员可以添加、编辑、删除部门信息，管理学校各部门和机构。

#### 业务处理流程及要求

（1）部门编号必须唯一；

（2）部门名称不能为空；

（3）部门类型必须选择（行政部门、直属部门、学院）；

（4）删除部门前检查是否有关联的预约记录；

（5）所有操作记录审计日志。

#### 输入信息

部门编号、部门名称、部门类型。

#### 输出信息

显示部门列表，包括ID、部门编号、部门名称、部门类型等信息，以及添加、编辑、删除操作按钮。

### 统计报表功能

#### 功能描述

管理员查看系统使用统计和报告，包括按申请月份、进校月份、校区、部门的统计信息。

#### 业务处理流程及要求

（1）系统管理员和学校管理员可查看所有统计数据；

（2）部门管理员根据权限查看本部门相关统计；

（3）统计数据包括预约数量、预约类型分布、校区分布等；

#### 输入信息

无。

#### 输出信息

显示各种统计图表和数据，包括按申请月份统计、按进校月份统计、按校区统计、按部门统计等。

### 审计日志功能

#### 功能描述

管理员查看系统操作日志，包括登录记录、操作记录等，用于安全审计。

#### 业务处理流程及要求

（1）记录所有重要操作（登录、预约申请、审核、管理等）；

（2）日志包含操作人、操作时间、操作类型、操作详情、IP地址等信息；

（3）使用SM3算法保护日志完整性；

#### 输入信息

无。

#### 输出信息

显示审计日志列表，包括操作时间、操作人、操作类型、操作详情、IP地址等信息。

### 用户管理功能

#### 功能描述

管理员管理普通用户信息，查看用户预约记录和状态。

#### 业务处理流程及要求

（1）支持按多种条件查询用户预约记录；

（2）显示用户的预约历史和状态；

（3）支持重置用户密码等操作；

（4）所有操作记录审计日志。

#### 输入信息

无。

#### 输出信息

显示审计日志列表，包括操作时间、操作人、操作类型、操作详情、IP地址等信息。

## 系统安全功能模块

### 数据加密功能

#### 功能描述

系统对敏感信息进行加密存储和传输，保障数据安全。

#### 业务处理流程及要求

（1）密码使用SM3哈希算法存储；

（2）身份证号、手机号使用SM4加密算法存储；

（3）审计日志使用HMAC-SM3算法保护完整性；

（4）页面显示时对敏感信息进行脱敏处理。

#### 输入信息

原始敏感数据。

#### 输出信息

加密后的数据或脱敏显示的数据。

### 权限控制功能

#### 功能描述

系统根据管理员角色和权限控制功能访问，确保安全性。

#### 业务处理流程及要求

（1）四种管理员角色：系统管理员、学校管理员、部门管理员、审计管理员；（2）不同角色有不同的功能权限；

（3）部门管理员默认情况下只能管理本部门相关数据；

（4）所有功能访问都需要验证权限。

#### 输入信息

管理员角色、权限设置。

#### 输出信息

根据权限显示相应的功能菜单和操作选项。

# 系统详细设计

## Beans

### Department

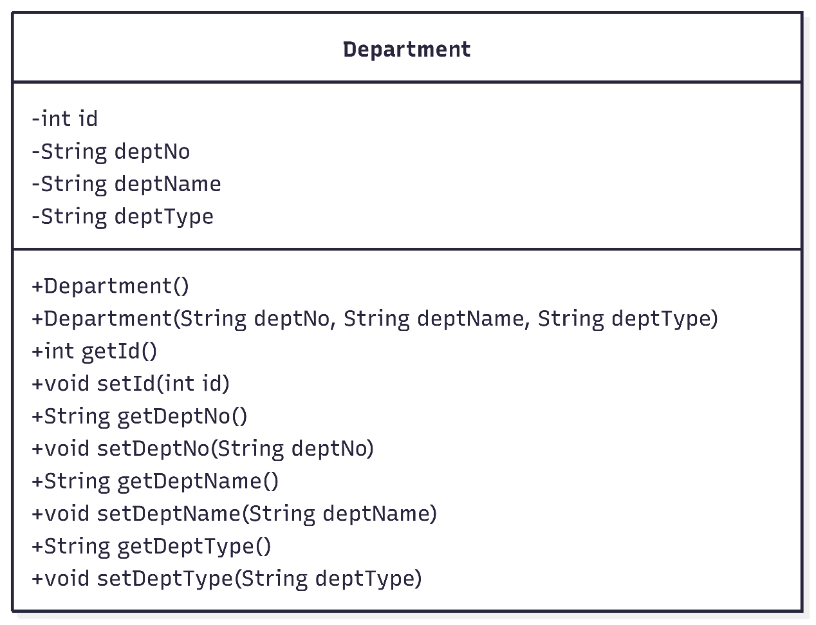


图 4 Department

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | Department |
| 继承或实现 | 无 |
| 描述 | 部门类 |
| 类别 | 实体类 |
| 包 | com.zjut.passcode.bean |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| id | int | 部门ID |
| deptNo | String | 部门编号 |
| deptName | String | 部门名称 |
| deptType | String | 部门类型 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| getId | public | 获取部门ID |
| setId | public | 设置部门ID |
| getDeptNo | public | 获取部门编号 |
| setDeptNo | public | 设置部门编号 |
| getDeptName | public | 获取部门名称 |
| setDeptName | public | 设置部门名称 |
| getDeptType | public | 获取部门类型 |
| setDeptType | public | 设置部门类型 |

#### 方法详细说明

##### getId 方法说明

|  |  |
| --- | --- |
| 方法名称 | getId |
| 描述 | 获取部门ID |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 id 的值。 |
| 输出参数 | int |

##### setId 方法说明

|  |  |
| --- | --- |
| 方法名称 | setId |
| 描述 | 设置部门ID |
| 输入参数 | int id |
| 步骤 | 将传入参数 id 赋值给成员变量 id。 |
| 输出参数 | void |

##### getDeptNo 方法说明

|  |  |
| --- | --- |
| 方法名称 | getDeptNo |
| 描述 | 获取部门编号 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 deptNo 的值。 |
| 输出参数 | String |

##### setDeptNo 方法说明

|  |  |
| --- | --- |
| 方法名称 | setDeptNo |
| 描述 | 设置部门编号 |
| 输入参数 | String deptNo |
| 步骤 | 将传入参数 deptNo 赋值给成员变量 deptNo。 |
| 输出参数 | void |

##### getDeptName 方法说明

|  |  |
| --- | --- |
| 方法名称 | getDeptName |
| 描述 | 获取部门名称 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 deptName 的值。 |
| 输出参数 | String |

##### setDeptName 方法说明

|  |  |
| --- | --- |
| 方法名称 | setDeptName |
| 描述 | 设置部门名称 |
| 输入参数 | String deptName |
| 步骤 | 将传入参数 deptName 赋值给成员变量 deptName。 |
| 输出参数 | void |

#### getDeptType 方法说明

|  |  |
| --- | --- |
| 方法名称 | getDeptType |
| 描述 | 获取部门类型 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 deptType 的值。 |
| 输出参数 | String |

##### setDeptType 方法说明

|  |  |
| --- | --- |
| 方法名称 | setDeptType |
| 描述 | 设置部门类型 |
| 输入参数 | String deptType |
| 步骤 | 将传入参数 deptType 赋值给成员变量 deptType。 |
| 输出参数 | void |

### Admin



图 5 Admin

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | Admin |
| 继承或实现 | 无 |
| 描述 | 管理员类 |
| 类别 | 实体类 |
| 包 | com.zjut.passcode.bean |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| id | int | 管理员ID |
| loginName | String | 登录名 |
| passwordHash | String | 密码哈希 |
| fullName | String | 全名 |
| deptId | int | 部门ID |
| phone | String | 电话 |
| role | String | 角色 |
| passwordLastChanged | Timestamp | 密码最后修改时间 |
| failedLoginAttempts | int | 登录失败次数 |
| lockoutUntil | Timestamp | 锁定截止时间 |
| deptName | String | 部门名称（用于显示） |
| canManagePublicAppointment | boolean | 是否可以管理公共预约 |
| canReportPublicAppointment | boolean | 是否可以报告公共预约 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| getId | public | 获取管理员ID |
| setId | public | 设置管理员ID |
| getLoginName | public | 获取登录名 |
| setLoginName | public | 设置登录名 |
| getPasswordHash | public | 获取密码哈希 |
| setPasswordHash | public | 设置密码哈希 |
| getFullName | public | 获取全名 |
| setFullName | public | 设置全名 |
| getDeptId | public | 获取部门ID |
| setDeptId | public | 设置部门ID |
| getPhone | public | 获取电话 |
| setPhone | public | 设置电话 |
| getRole | public | 获取角色 |
| setRole | public | 设置角色 |
| getPasswordLastChanged | public | 获取密码最后修改时间 |
| setPasswordLastChanged | public | 设置密码最后修改时间 |
| getFailedLoginAttempts | public | 获取登录失败次数 |
| setFailedLoginAttempts | public | 设置登录失败次数 |
| getLockoutUntil | public | 获取锁定截止时间 |
| setLockoutUntil | public | 设置锁定截止时间 |
| getDeptName | public | 获取部门名称 |
| setDeptName | public | 设置部门名称 |
| isCanManagePublicAppointment | public | 获取是否可以管理公共预约 |
| setCanManagePublicAppointment | public | 设置是否可以管理公共预约 |
| isCanReportPublicAppointment | public | 获取是否可以报告公共预约 |
| setCanReportPublicAppointment | public | 设置是否可以报告公共预约 |

#### 方法详细说明

##### getId 方法说明

|  |  |
| --- | --- |
| 方法名称 | getId |
| 描述 | 获取管理员ID |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 id 的值。 |
| 输出参数 | int |

##### setId 方法说明

|  |  |
| --- | --- |
| 方法名称 | setId |
| 描述 | 设置管理员ID |
| 输入参数 | int id |
| 步骤 | 将传入参数 id 赋值给成员变量 id。 |
| 输出参数 | void |

##### getLoginName 方法说明

|  |  |
| --- | --- |
| 方法名称 | getLoginName |
| 描述 | 获取登录名 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 loginName 的值。 |
| 输出参数 | String |

##### setLoginName 方法说明

|  |  |
| --- | --- |
| 方法名称 | setLoginName |
| 描述 | 设置登录名 |
| 输入参数 | String loginName |
| 步骤 | 将传入参数 loginName 赋值给成员变量 loginName。 |
| 输出参数 | void |

##### getPasswordHash 方法说明

|  |  |
| --- | --- |
| 方法名称 | getPasswordHash |
| 描述 | 获取密码哈希 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 passwordHash 的值。 |
| 输出参数 | String |

##### setPasswordHash 方法说明

|  |  |
| --- | --- |
| 方法名称 | setPasswordHash |
| 描述 | 设置密码哈希 |
| 输入参数 | String passwordHash |
| 步骤 | 将传入参数 passwordHash 赋值给成员变量 passwordHash。 |
| 输出参数 | void |

##### getFullName 方法说明

|  |  |
| --- | --- |
| 方法名称 | getFullName |
| 描述 | 获取全名 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 fullName 的值。 |
| 输出参数 | String |

##### setFullName 方法说明

|  |  |
| --- | --- |
| 方法名称 | setFullName |
| 描述 | 设置全名 |
| 输入参数 | String fullName |
| 步骤 | 将传入参数 fullName 赋值给成员变量 fullName。 |
| 输出参数 | void |

##### getDeptId 方法说明

|  |  |
| --- | --- |
| 方法名称 | getDeptId |
| 描述 | 获取部门ID |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 deptId 的值。 |
| 输出参数 | int |

##### setDeptId 方法说明

|  |  |
| --- | --- |
| 方法名称 | setDeptId |
| 描述 | 设置部门ID |
| 输入参数 | int deptId |
| 步骤 | 将传入参数 deptId 赋值给成员变量 deptId。 |
| 输出参数 | void |

##### getPhone 方法说明

|  |  |
| --- | --- |
| 方法名称 | getPhone |
| 描述 | 获取电话 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 phone 的值。 |
| 输出参数 | String |

##### setPhone 方法说明

|  |  |
| --- | --- |
| 方法名称 | setPhone |
| 描述 | 设置电话 |
| 输入参数 | String phone |
| 步骤 | 将传入参数 phone 赋值给成员变量 phone。 |
| 输出参数 | void |

##### getRole 方法说明

|  |  |
| --- | --- |
| 方法名称 | getRole |
| 描述 | 获取角色 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 role 的值。 |
| 输出参数 | String |

##### setRole 方法说明

|  |  |
| --- | --- |
| 方法名称 | setRole |
| 描述 | 设置角色 |
| 输入参数 | String role |
| 步骤 | 将传入参数 role 赋值给成员变量 role。 |
| 输出参数 | void |

##### getPasswordLastChanged 方法说明

|  |  |
| --- | --- |
| 方法名称 | getPasswordLastChanged |
| 描述 | 获取密码最后修改时间 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 passwordLastChanged 的值。 |
| 输出参数 | Timestamp |

##### setPasswordLastChanged 方法说明

|  |  |
| --- | --- |
| 方法名称 | setPasswordLastChanged |
| 描述 | 设置密码最后修改时间 |
| 输入参数 | Timestamp passwordLastChanged |
| 步骤 | 将传入参数 passwordLastChanged 赋值给成员变量 passwordLastChanged。 |
| 输出参数 | void |

##### getFailedLoginAttempts 方法说明

|  |  |
| --- | --- |
| 方法名称 | getFailedLoginAttempts |
| 描述 | 获取登录失败次数 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 failedLoginAttempts 的值。 |
| 输出参数 | int |

##### setFailedLoginAttempts 方法说明

|  |  |
| --- | --- |
| 方法名称 | setFailedLoginAttempts |
| 描述 | 设置登录失败次数 |
| 输入参数 | int failedLoginAttempts |
| 步骤 | 将传入参数 failedLoginAttempts 赋值给成员变量 failedLoginAttempts。 |
| 输出参数 | void |

##### getLockoutUntil 方法说明

|  |  |
| --- | --- |
| 方法名称 | getLockoutUntil |
| 描述 | 获取锁定截止时间 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 lockoutUntil 的值。 |
| 输出参数 | Timestamp |

##### setLockoutUntil 方法说明

|  |  |
| --- | --- |
| 方法名称 | setLockoutUntil |
| 描述 | 设置锁定截止时间 |
| 输入参数 | Timestamp lockoutUntil |
| 步骤 | 将传入参数 lockoutUntil 赋值给成员变量 lockoutUntil。 |
| 输出参数 | void |

##### getDeptName 方法说明

|  |  |
| --- | --- |
| 方法名称 | getDeptName |
| 描述 | 获取部门名称 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 deptName 的值。 |
| 输出参数 | String |

##### setDeptName 方法说明

|  |  |
| --- | --- |
| 方法名称 | setDeptName |
| 描述 | 设置部门名称 |
| 输入参数 | String deptName |
| 步骤 | 将传入参数 deptName 赋值给成员变量 deptName。 |
| 输出参数 | void |

##### isCanManagePublicAppointment 方法说明

|  |  |
| --- | --- |
| 方法名称 | isCanManagePublicAppointment |
| 描述 | 获取是否可以管理公共预约 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 canManagePublicAppointment 的值。 |
| 输出参数 | boolean |

##### setCanManagePublicAppointment 方法说明

|  |  |
| --- | --- |
| 方法名称 | setCanManagePublicAppointment |
| 描述 | 设置是否可以管理公共预约 |
| 输入参数 | boolean canManagePublicAppointment |
| 步骤 | 将传入参数 canManagePublicAppointment 赋值给成员变量 canManagePublicAppointment。 |
| 输出参数 | void |

##### isCanReportPublicAppointment 方法说明

|  |  |
| --- | --- |
| 方法名称 | isCanReportPublicAppointment |
| 描述 | 获取是否可以报告公共预约 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 canReportPublicAppointment 的值。 |
| 输出参数 | boolean |

##### setCanReportPublicAppointment 方法说明

|  |  |
| --- | --- |
| 方法名称 | setCanReportPublicAppointment |
| 描述 | 设置是否可以报告公共预约 |
| 输入参数 | boolean canReportPublicAppointment |
| 步骤 | 将传入参数 canReportPublicAppointment 赋值给成员变量 canReportPublicAppointment。 |
| 输出参数 | void |

### Appointment

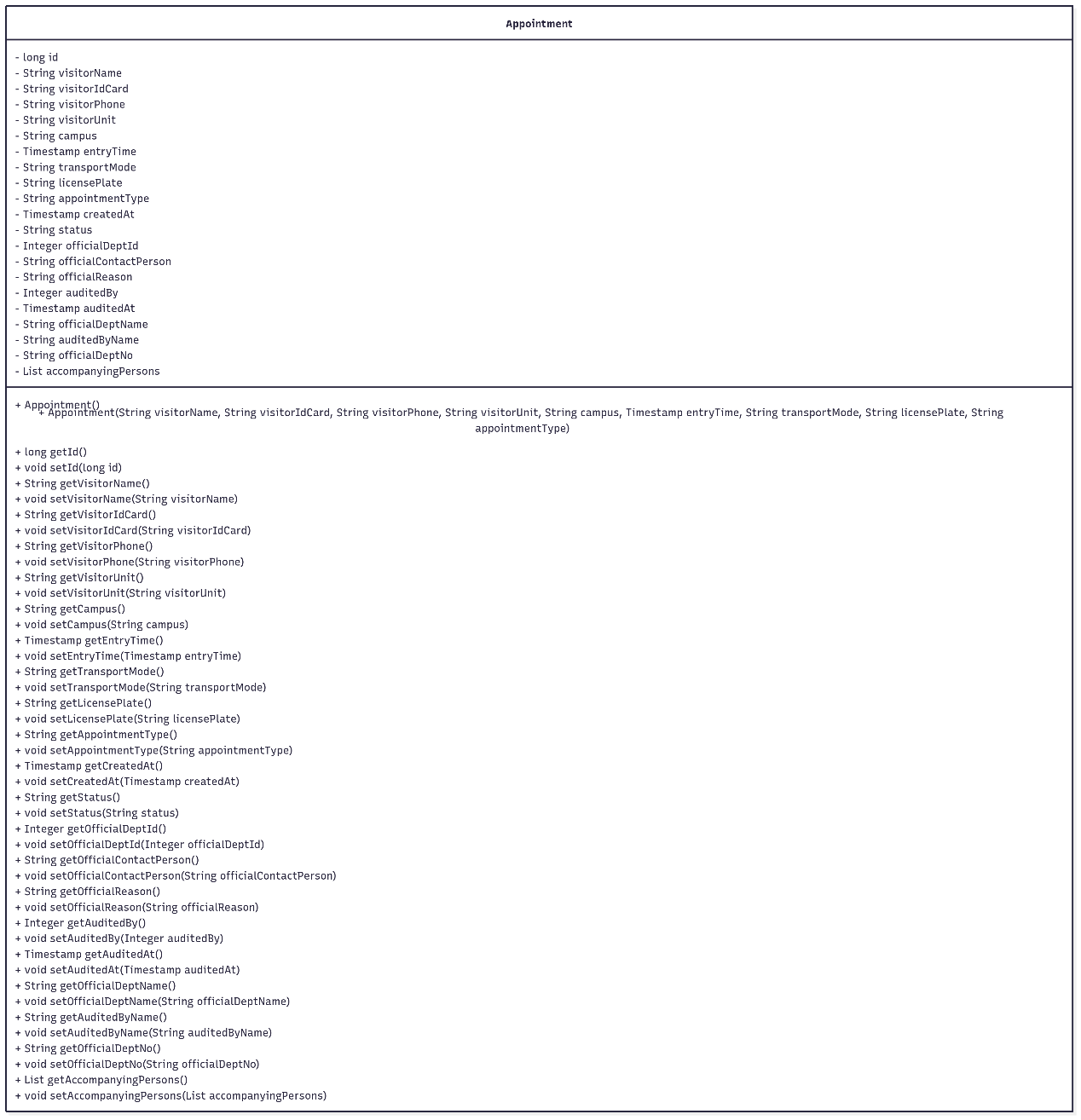


图 6 Appointment

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | Appointment |
| 继承或实现 | 无 |
| 描述 | 预约实体类，封装预约相关信息 |
| 类别 | 实体类 |
| 包 | com.zjut.passcode.bean |
| 与其他类的调用关系 | 是 |

#### 属性表

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| id | long | 预约ID |
| visitorName | String | 访客姓名 |
| visitorIdCard | String | 访客身份证号 |
| visitorPhone | String | 访客手机号 |
| visitorUnit | String | 访客单位 |
| campus | String | 校区 |
| entryTime | Timestamp | 进校时间 |
| transportMode | String | 交通方式 |
| licensePlate | String | 车牌号 |
| appointmentType | String | 预约类型 |
| createdAt | Timestamp | 创建时间 |
| status | String | 预约状态 |
| officialDeptId | Integer | 公务访问部门ID |
| officialContactPerson | String | 公务联系人 |
| officialReason | String | 公务事由 |
| auditedBy | Integer | 审核人ID |
| auditedAt | Timestamp | 审核时间 |
| officialDeptName | String | 公务访问部门名称 |
| auditedByName | String | 审核人姓名 |
| officialDeptNo | String | 公务访问部门编号 |
| accompanyingPersons | List<AccompanyingPerson> | 随行人员列表 |

#### 主要方法表

名称 属性 描述

|  |  |  |
| --- | --- | --- |
| getId | public | 获取预约ID |
| setId | public | 设置预约ID |
| getVisitorName | public | 获取访客姓名 |
| setVisitorName | public | 设置访客姓名 |
| getVisitorIdCard | public | 获取访客身份证号 |
| setVisitorIdCard | public | 设置访客身份证号 |
| getVisitorPhone | public | 获取访客手机号 |
| setVisitorPhone | public | 设置访客手机号 |
| getVisitorUnit | public | 获取访客单位 |
| setVisitorUnit | public | 设置访客单位 |
| getCampus | public | 获取校区 |
| setCampus | public | 设置校区 |
| getEntryTime | public | 获取进校时间 |
| setEntryTime | public | 设置进校时间 |
| getTransportMode | public | 获取交通方式 |
| setTransportMode | public | 设置交通方式 |
| getLicensePlate | public | 获取车牌号 |
| setLicensePlate | public | 设置车牌号 |
| getAppointmentType | public | 获取预约类型 |
| setAppointmentType | public | 设置预约类型 |
| getCreatedAt | public | 获取创建时间 |
| setCreatedAt | public | 设置创建时间 |
| getStatus | public | 获取预约状态 |
| setStatus | public | 设置预约状态 |
| getOfficialDeptId | public | 获取公务访问部门ID |
| setOfficialDeptId | public | 设置公务访问部门ID |
| getOfficialContactPerson | public | 获取公务联系人 |
| setOfficialContactPerson | public | 设置公务联系人 |
| getOfficialReason | public | 获取公务事由 |
| setOfficialReason | public | 设置公务事由 |
| getAuditedBy | public | 获取审核人ID |
| setAuditedBy | public | 设置审核人ID |
| getAuditedAt | public | 获取审核时间 |
| setAuditedAt | public | 设置审核时间 |
| getOfficialDeptName | public | 获取公务访问部门名称 |
| setOfficialDeptName | public | 设置公务访问部门名称 |
| getAuditedByName | public | 获取审核人姓名 |
| setAuditedByName | public | 设置审核人姓名 |
| getOfficialDeptNo | public | 获取公务访问部门编号 |
| setOfficialDeptNo | public | 设置公务访问部门编号 |
| getAccompanyingPersons | public | 获取随行人员列表 |
| setAccompanyingPersons | public | 设置随行人员列表 |

#### 方法详细说明

##### getId 方法说明

|  |  |
| --- | --- |
| 方法名称 | getId |
| 描述 | 获取预约ID |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 id 的值。 |
| 输出参数 | long |

##### setId 方法说明

|  |  |
| --- | --- |
| 方法名称 | setId |
| 描述 | 设置预约ID |
| 输入参数 | long id |
| 步骤 | 将传入参数 id 赋值给成员变量 id。 |
| 输出参数 | void |

##### getVisitorName 方法说明

|  |  |
| --- | --- |
| 方法名称 | getVisitorName |
| 描述 | 获取访客姓名 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 visitorName 的值。 |
| 输出参数 | String |

##### setVisitorName 方法说明

|  |  |
| --- | --- |
| 方法名称 | setVisitorName |
| 描述 | 设置访客姓名 |
| 输入参数 | String visitorName |
| 步骤 | 将传入参数 visitorName 赋值给成员变量 visitorName。 |
| 输出参数 | void |

##### getVisitorIdCard 方法说明

|  |  |
| --- | --- |
| 方法名称 | getVisitorIdCard |
| 描述 | 获取访客身份证号 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 visitorIdCard 的值。 |
| 输出参数 | String |

##### setVisitorIdCard 方法说明

|  |  |
| --- | --- |
| 方法名称 | setVisitorIdCard |
| 描述 | 设置访客身份证号 |
| 输入参数 | String visitorIdCard |
| 步骤 | 将传入参数 visitorIdCard 赋值给成员变量 visitorIdCard。 |
| 输出参数 | void |

##### getVisitorPhone 方法说明

|  |  |
| --- | --- |
| 方法名称 | getVisitorPhone |
| 描述 | 获取访客手机号 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 visitorPhone 的值。 |
| 输出参数 | String |

##### setVisitorUnit 方法说明

|  |  |
| --- | --- |
| 方法名称 | setVisitorUnit |
| 描述 | 设置访客单位 |
| 输入参数 | String visitorUnit |
| 步骤 | 将传入参数 visitorUnit 赋值给成员变量 visitorUnit。 |
| 输出参数 | void |

##### getCampus 方法说明

|  |  |
| --- | --- |
| 方法名称 | getCampus |
| 描述 | 获取校区 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 campus 的值。 |
| 输出参数 | String |

##### setCampus 方法说明

|  |  |
| --- | --- |
| 方法名称 | setCampus |
| 描述 | 设置校区 |
| 输入参数 | String campus |
| 步骤 | 将传入参数 campus 赋值给成员变量 campus。 |
| 输出参数 | void |

##### getEntryTime 方法说明

|  |  |
| --- | --- |
| 方法名称 | getEntryTime |
| 描述 | 获取进校时间 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 entryTime 的值。 |
| 输出参数 | Timestamp |

##### setEntryTime 方法说明

|  |  |
| --- | --- |
| 方法名称 | setEntryTime |
| 描述 | 设置进校时间 |
| 输入参数 | Timestamp entryTime |
| 步骤 | 将传入参数 entryTime 赋值给成员变量 entryTime。 |
| 输出参数 | void |

##### getTransportMode 方法说明

|  |  |
| --- | --- |
| 方法名称 | getTransportMode |
| 描述 | 获取交通方式 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 transportMode 的值。 |
| 输出参数 | String |

##### setTransportMode 方法说明

|  |  |
| --- | --- |
| 方法名称 | setTransportMode |
| 描述 | 设置交通方式 |
| 输入参数 | String transportMode |
| 步骤 | 将传入参数 transportMode 赋值给成员变量 transportMode。 |
| 输出参数 | void |

##### getLicensePlate 方法说明

|  |  |
| --- | --- |
| 方法名称 | getLicensePlate |
| 描述 | 获取车牌号 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 licensePlate 的值。 |
| 输出参数 | String |

##### setLicensePlate 方法说明

|  |  |
| --- | --- |
| 方法名称 | setLicensePlate |
| 描述 | 设置车牌号 |
| 输入参数 | String licensePlate |
| 步骤 | 将传入参数 licensePlate 赋值给成员变量 licensePlate。 |
| 输出参数 | void |

##### getAppointmentType 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAppointmentType |
| 描述 | 获取预约类型 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 appointmentType 的值。 |
| 输出参数 | String |

##### setAppointmentType 方法说明

|  |  |
| --- | --- |
| 方法名称 | setAppointmentType |
| 描述 | 设置预约类型 |
| 输入参数 | String appointmentType |
| 步骤 | 将传入参数 appointmentType 赋值给成员变量 appointmentType。 |
| 输出参数 | void |

##### getCreatedAt 方法说明

|  |  |
| --- | --- |
| 方法名称 | getCreatedAt |
| 描述 | 获取创建时间 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 createdAt 的值。 |
| 输出参数 | Timestamp |

##### setCreatedAt 方法说明

|  |  |
| --- | --- |
| 方法名称 | setCreatedAt |
| 描述 | 设置创建时间 |
| 输入参数 | Timestamp createdAt |
| 步骤 | 将传入参数 createdAt 赋值给成员变量 createdAt。 |
| 输出参数 | void |

##### getStatus 方法说明

|  |  |
| --- | --- |
| 方法名称 | getStatus |
| 描述 | 获取预约状态 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 status 的值。 |
| 输出参数 | String |

##### setStatus 方法说明

|  |  |
| --- | --- |
| 方法名称 | setStatus |
| 描述 | 设置预约状态 |
| 输入参数 | String status |
| 步骤 | 将传入参数 status 赋值给成员变量 status。 |
| 输出参数 | void |

##### getOfficialDeptId 方法说明

|  |  |
| --- | --- |
| 方法名称 | getOfficialDeptId |
| 描述 | 获取公务访问部门ID |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 officialDeptId 的值。 |
| 输出参数 | Integer |

##### setOfficialDeptId 方法说明

|  |  |
| --- | --- |
| 方法名称 | setOfficialDeptId |
| 描述 | 设置公务访问部门ID |
| 输入参数 | Integer officialDeptId |
| 步骤 | 将传入参数 officialDeptId 赋值给成员变量 officialDeptId。 |
| 输出参数 | void |

##### getOfficialContactPerson 方法说明

|  |  |
| --- | --- |
| 方法名称 | getOfficialContactPerson |
| 描述 | 获取公务联系人 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 officialContactPerson 的值。 |
| 输出参数 | String |

##### setOfficialContactPerson 方法说明

|  |  |
| --- | --- |
| 方法名称 | setOfficialContactPerson |
| 描述 | 设置公务联系人 |
| 输入参数 | String officialContactPerson |
| 步骤 | 将传入参数 officialContactPerson 赋值给成员变量 officialContactPerson。 |
| 输出参数 | void |

##### getOfficialReason 方法说明

|  |  |
| --- | --- |
| 方法名称 | getOfficialReason |
| 描述 | 获取公务事由 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 officialReason 的值。 |
| 输出参数 | String |

##### setOfficialReason 方法说明

|  |  |
| --- | --- |
| 方法名称 | setOfficialReason |
| 描述 | 设置公务事由 |
| 输入参数 | String officialReason |
| 步骤 | 将传入参数 officialReason 赋值给成员变量 officialReason。 |
| 输出参数 | void |

##### getAuditedBy 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAuditedBy |
| 描述 | 获取审核人ID |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 auditedBy 的值。 |
| 输出参数 | Integer |

##### setAuditedBy 方法说明

|  |  |
| --- | --- |
| 方法名称 | setAuditedBy |
| 描述 | 设置审核人ID |
| 输入参数 | Integer auditedBy |
| 步骤 | 将传入参数 auditedBy 赋值给成员变量 auditedBy。 |
| 输出参数 | void |

##### getAuditedAt 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAuditedAt |
| 描述 | 获取审核时间 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 auditedAt 的值。 |
| 输出参数 | Timestamp |

##### setAuditedAt 方法说明

|  |  |
| --- | --- |
| 方法名称 | setAuditedAt |
| 描述 | 设置审核时间 |
| 输入参数 | Timestamp auditedAt |
| 步骤 | 将传入参数 auditedAt 赋值给成员变量 auditedAt。 |
| 输出参数 | void |

##### getOfficialDeptName 方法说明

|  |  |
| --- | --- |
| 方法名称 | getOfficialDeptName |
| 描述 | 获取公务访问部门名称 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 officialDeptName 的值。 |
| 输出参数 | String |

##### setOfficialDeptName 方法说明

|  |  |
| --- | --- |
| 方法名称 | setOfficialDeptName |
| 描述 | 设置公务访问部门名称 |
| 输入参数 | String officialDeptName |
| 步骤 | 将传入参数 officialDeptName 赋值给成员变量 officialDeptName。 |
| 输出参数 | void |

##### getAuditedByName 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAuditedByName |
| 描述 | 获取审核人姓名 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 auditedByName 的值。 |
| 输出参数 | String |

##### setAuditedByName 方法说明

|  |  |
| --- | --- |
| 方法名称 | setAuditedByName |
| 描述 | 设置审核人姓名 |
| 输入参数 | String auditedByName |
| 步骤 | 将传入参数 auditedByName 赋值给成员变量 auditedByName。 |
| 输出参数 | void |

##### getOfficialDeptNo 方法说明

|  |  |
| --- | --- |
| 方法名称 | getOfficialDeptNo |
| 描述 | 获取公务访问部门编号 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 officialDeptNo 的值。 |
| 输出参数 | String |

##### setOfficialDeptNo 方法说明

|  |  |
| --- | --- |
| 方法名称 | setOfficialDeptNo |
| 描述 | 设置公务访问部门编号 |
| 输入参数 | String officialDeptNo |
| 步骤 | 将传入参数 officialDeptNo 赋值给成员变量 officialDeptNo。 |
| 输出参数 | void |

##### getAccompanyingPersons 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAccompanyingPersons |
| 描述 | 获取随行人员列表 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 accompanyingPersons 的值。 |
| 输出参数 | List<AccompanyingPerson> |

##### setAccompanyingPersons 方法说明

|  |  |
| --- | --- |
| 方法名称 | setAccompanyingPersons |
| 描述 | 设置随行人员列表 |
| 输入参数 | List<AccompanyingPerson> accompanyingPersons |
| 步骤 | 将传入参数 accompanyingPersons 赋值给成员变量 accompanyingPersons。 |
| 输出参数 | void |

### AccompanyingPerson

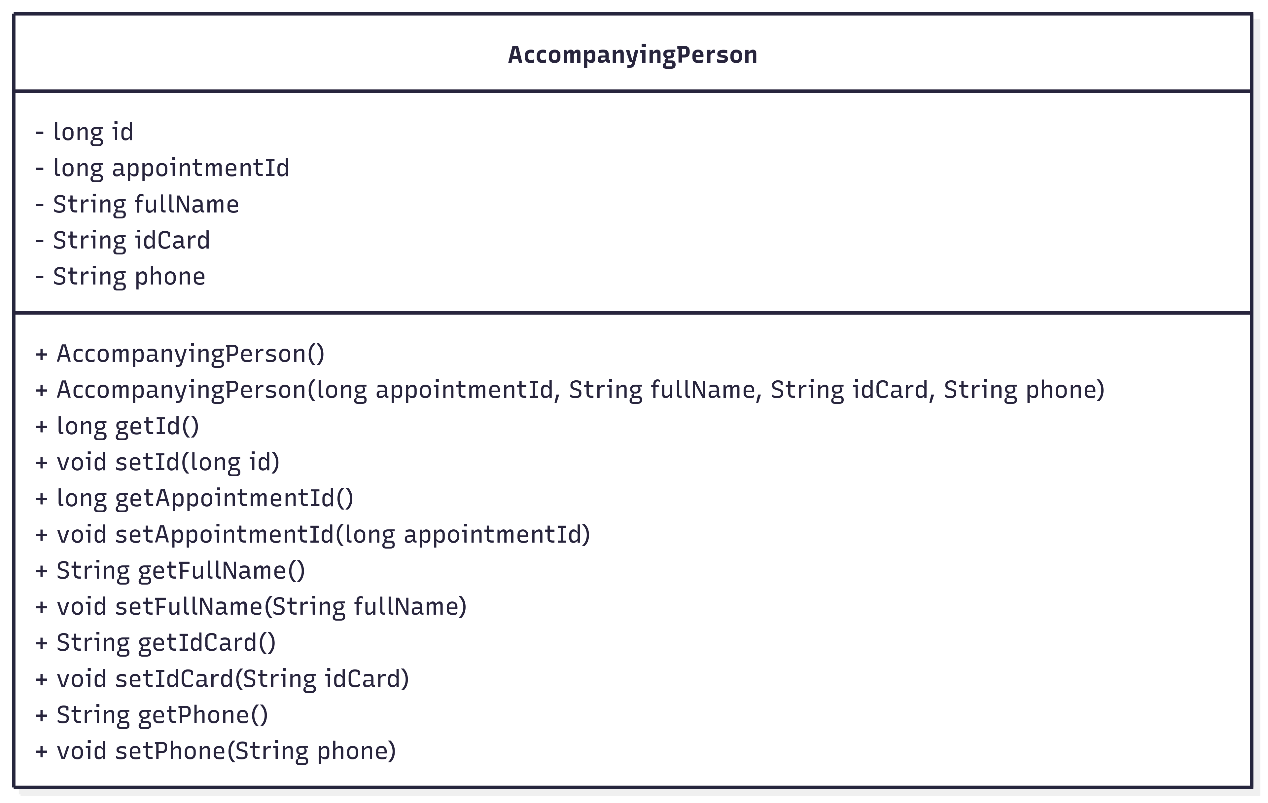


图 7 AccompanyingPerson

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | AccompanyingPerson |
| 继承或实现 | 无 |
| 描述 | 随行人员类 |
| 类别 | 实体类 |
| 包 | com.zjut.passcode.bean |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| id | long | 随行人员ID |
| appointmentId | long | 预约ID |
| fullName | String | 姓名 |
| idCard | String | 身份证号 |
| phone | String | 手机号 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| getId | public | 获取随行人员ID |
| setId | public | 设置随行人员ID |
| getAppointmentId | public | 获取预约ID |
| setAppointmentId | public | 设置预约ID |
| getFullName | public | 获取姓名 |
| setFullName | public | 设置姓名 |
| getIdCard | public | 获取身份证号 |
| setIdCard | public | 设置身份证号 |
| getPhone | public | 获取手机号 |
| setPhone | public | 设置手机号 |

#### 方法详细说明

##### getId 方法说明

|  |  |
| --- | --- |
| 方法名称 | getId |
| 描述 | 获取随行人员ID |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 id 的值。 |
| 输出参数 | long |

##### setId 方法说明

|  |  |
| --- | --- |
| 方法名称 | setId |
| 描述 | 设置随行人员ID |
| 输入参数 | long id |
| 步骤 | 将传入参数 id 赋值给成员变量 id。 |
| 输出参数 | void |

##### getAppointmentId 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAppointmentId |
| 描述 | 获取预约ID |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 appointmentId 的值。 |
| 输出参数 | long |

##### setAppointmentId 方法说明

|  |  |
| --- | --- |
| 方法名称 | setAppointmentId |
| 描述 | 设置预约ID |
| 输入参数 | long appointmentId |
| 步骤 | 将传入参数 appointmentId 赋值给成员变量 appointmentId。 |
| 输出参数 | void |

##### getFullName 方法说明

|  |  |
| --- | --- |
| 方法名称 | getFullName |
| 描述 | 获取姓名 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 fullName 的值。 |
| 输出参数 | String |

##### setFullName 方法说明

|  |  |
| --- | --- |
| 方法名称 | setFullName |
| 描述 | 设置姓名 |
| 输入参数 | String fullName |
| 步骤 | 将传入参数 fullName 赋值给成员变量 fullName。 |
| 输出参数 | void |

##### getIdCard 方法说明

|  |  |
| --- | --- |
| 方法名称 | getIdCard |
| 描述 | 获取身份证号 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 idCard 的值。 |
| 输出参数 | String |

##### setIdCard 方法说明

|  |  |
| --- | --- |
| 方法名称 | setIdCard |
| 描述 | 设置身份证号 |
| 输入参数 | String idCard |
| 步骤 | 将传入参数 idCard 赋值给成员变量 idCard。 |
| 输出参数 | void |

##### getPhone 方法说明

|  |  |
| --- | --- |
| 方法名称 | getPhone |
| 描述 | 获取手机号 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 phone 的值。 |
| 输出参数 | String |

##### setPhone 方法说明

|  |  |
| --- | --- |
| 方法名称 | setPhone |
| 描述 | 设置手机号 |
| 输入参数 | String phone |
| 步骤 | 将传入参数 phone 赋值给成员变量 phone。 |
| 输出参数 | void |

### AuditLog



图 8 AuditLog

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | AuditLog |
| 继承或实现 | 无 |
| 描述 | 审计日志类 |
| 类别 | 实体类 |
| 包 | com.zjut.passcode.bean |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| id | long | 审计日志ID |
| adminId | int | 管理员ID |
| adminName | String | 管理员姓名 |
| action | String | 操作类型 |
| details | String | 操作详情 |
| ipAddress | String | IP地址 |
| createdAt | Timestamp | 创建时间 |
| hmacSignature | String | HMAC签名 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| getId | public | 获取审计日志ID |
| setId | public | 设置审计日志ID |
| getAdminId | public | 获取管理员ID |
| setAdminId | public | 设置管理员ID |
| getAdminName | public | 获取管理员姓名 |
| setAdminName | public | 设置管理员姓名 |
| getAction | public | 获取操作类型 |
| setAction | public | 设置操作类型 |
| getDetails | public | 获取操作详情 |
| setDetails | public | 设置操作详情 |
| getIpAddress | public | 获取IP地址 |
| setIpAddress | public | 设置IP地址 |
| getCreatedAt | public | 获取创建时间 |
| setCreatedAt | public | 设置创建时间 |
| getHmacSignature | public | 获取HMAC签名 |
| setHmacSignature | public | 设置HMAC签名 |

#### 方法详细说明

##### getId 方法说明

|  |  |
| --- | --- |
| 方法名称 | getId |
| 描述 | 获取审计日志ID |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 id 的值。 |
| 输出参数 | long |

##### setId 方法说明

|  |  |
| --- | --- |
| 方法名称 | setId |
| 描述 | 设置审计日志ID |
| 输入参数 | long id |
| 步骤 | 将传入参数 id 赋值给成员变量 id。 |
| 输出参数 | void |

##### getAdminId 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAdminId |
| 描述 | 获取管理员ID |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 adminId 的值。 |
| 输出参数 | int |

##### setAdminId 方法说明

|  |  |
| --- | --- |
| 方法名称 | setAdminId |
| 描述 | 设置管理员ID |
| 输入参数 | int adminId |
| 步骤 | 将传入参数 adminId 赋值给成员变量 adminId。 |
| 输出参数 | void |

##### getAdminName 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAdminName |
| 描述 | 获取管理员姓名 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 adminName 的值。 |
| 输出参数 | String |

##### setAdminName 方法说明

|  |  |
| --- | --- |
| 方法名称 | setAdminName |
| 描述 | 设置管理员姓名 |
| 输入参数 | String adminName |
| 步骤 | 将传入参数 adminName 赋值给成员变量 adminName。 |
| 输出参数 | void |

##### getAction 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAction |
| 描述 | 获取操作类型 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 action 的值。 |
| 输出参数 | String |

##### setAction 方法说明

|  |  |
| --- | --- |
| 方法名称 | setAction |
| 描述 | 设置操作类型 |
| 输入参数 | String action |
| 步骤 | 将传入参数 action 赋值给成员变量 action。 |
| 输出参数 | void |

##### getDetails 方法说明

|  |  |
| --- | --- |
| 方法名称 | getDetails |
| 描述 | 获取操作详情 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 details 的值。 |
| 输出参数 | String |

##### setDetails 方法说明

|  |  |
| --- | --- |
| 方法名称 | setDetails |
| 描述 | 设置操作详情 |
| 输入参数 | String details |
| 步骤 | 将传入参数 details 赋值给成员变量 details。 |
| 输出参数 | void |

##### getIpAddress 方法说明

|  |  |
| --- | --- |
| 方法名称 | getIpAddress |
| 描述 | 获取IP地址 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 ipAddress 的值。 |
| 输出参数 | String |

##### setIpAddress 方法说明

|  |  |
| --- | --- |
| 方法名称 | setIpAddress |
| 描述 | 设置IP地址 |
| 输入参数 | String ipAddress |
| 步骤 | 将传入参数 ipAddress 赋值给成员变量 ipAddress。 |
| 输出参数 | void |

##### getCreatedAt 方法说明

|  |  |
| --- | --- |
| 方法名称 | getCreatedAt |
| 描述 | 获取创建时间 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 createdAt 的值。 |
| 输出参数 | Timestamp |

##### setCreatedAt 方法说明

|  |  |
| --- | --- |
| 方法名称 | setCreatedAt |
| 描述 | 设置创建时间 |
| 输入参数 | Timestamp createdAt |
| 步骤 | 将传入参数 createdAt 赋值给成员变量 createdAt。 |
| 输出参数 | void |

##### getHmacSignature 方法说明

|  |  |
| --- | --- |
| 方法名称 | getHmacSignature |
| 描述 | 获取HMAC签名 |
| 输入参数 | 无 |
| 步骤 | 直接返回成员变量 hmacSignature 的值。 |
| 输出参数 | String |

##### setHmacSignature 方法说明

|  |  |
| --- | --- |
| 方法名称 | setHmacSignature |
| 描述 | 设置HMAC签名 |
| 输入参数 | String hmacSignature |
| 步骤 | 将传入参数 hmacSignature 赋值给成员变量 hmacSignature。 |
| 输出参数 | void |

## Dao

### AdminDao

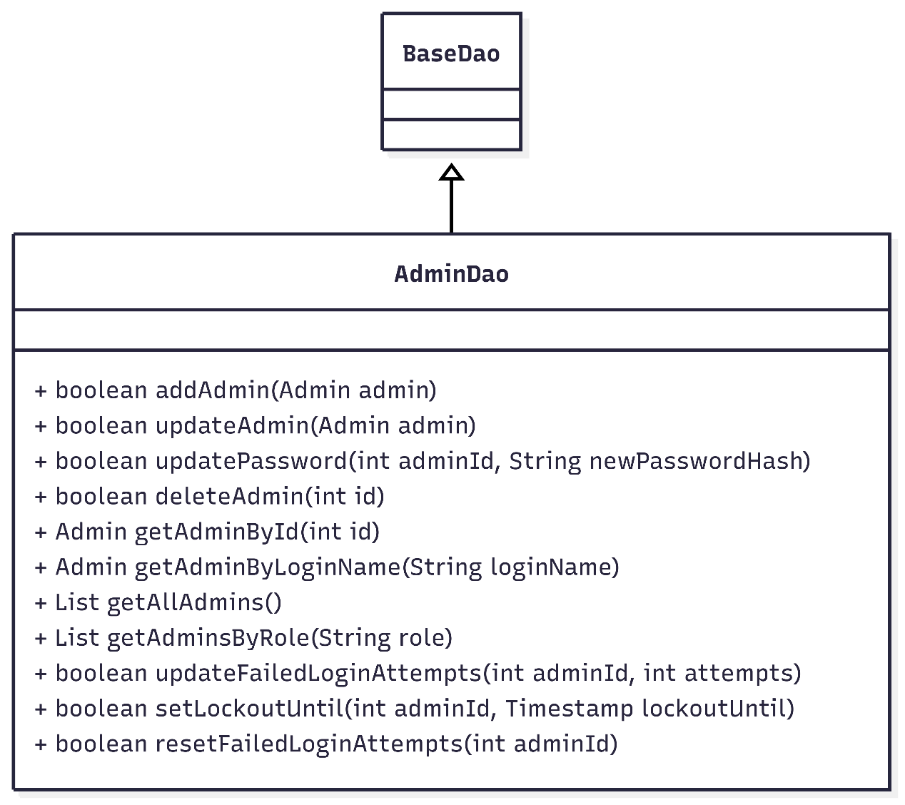


图 9 AdminDao

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | AdminDao |
| 继承或实现 | 继承 BaseDao |
| 描述 | 管理员数据访问对象类 |
| 类别 | DAO类 |
| 包 | com.zjut.passcode.dao |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| 无 | 无 | 无 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| addAdmin | public | 添加管理员 |
| updateAdmin | public | 更新管理员信息 |
| updatePassword | public | 更新管理员密码 |
| deleteAdmin | public | 删除管理员 |
| getAdminById | public | 根据ID获取管理员 |
| getAdminByLoginName | public | 根据登录名获取管理员 |
| getAllAdmins | public | 获取所有管理员 |
| getAdminsByRole | public | 根据角色获取管理员列表 |
| updateFailedLoginAttempts | public | 更新登录失败次数 |
| setLockoutUntil | public | 设置锁定截止时间 |
| resetFailedLoginAttempts | public | 重置登录失败次数 |

#### 方法详细说明

##### addAdmin 方法说明

|  |  |
| --- | --- |
| 方法名称 | addAdmin |
| 描述 | 添加管理员 |
| 输入参数 | Admin admin |
| 步骤 | 1. 准备SQL插入语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行SQL语句；  5. 返回执行结果。 |
| 输出参数 | boolean |

##### updateAdmin 方法说明

|  |  |
| --- | --- |
| 方法名称 | updateAdmin |
| 描述 | 更新管理员信息 |
| 输入参数 | Admin admin |
| 步骤 | 1. 准备SQL更新语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行SQL语句；  5. 返回执行结果。 |
| 输出参数 | boolean |

##### updatePassword 方法说明

|  |  |
| --- | --- |
| 方法名称 | updatePassword |
| 描述 | 更新管理员密码 |
| 输入参数 | int adminId, String newPasswordHash |
| 步骤 | 1. 准备SQL更新密码语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行SQL语句；  5. 返回执行结果。 |
| 输出参数 | boolean |

##### deleteAdmin 方法说明

|  |  |
| --- | --- |
| 方法名称 | deleteAdmin |
| 描述 | 删除管理员 |
| 输入参数 | int id |
| 步骤 | 1. 准备SQL删除语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行SQL语句；  5. 返回执行结果。 |
| 输出参数 | boolean |

##### getAdminById 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAdminById |
| 描述 | 根据ID获取管理员 |
| 输入参数 | int id |
| 步骤 | 1. 准备SQL查询语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行查询；  5. 遍历结果集并构建Admin对象；  6. 返回Admin对象。 |
| 输出参数 | Admin |

##### getAdminByLoginName 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAdminByLoginName |
| 描述 | 根据登录名获取管理员 |
| 输入参数 | String loginName |
| 步骤 | 1. 准备SQL查询语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行查询；  5. 遍历结果集并构建Admin对象；  6. 返回Admin对象。 |
| 输出参数 | Admin |

##### getAllAdmins 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAllAdmins |
| 描述 | 获取所有管理员 |
| 输入参数 | 无 |
| 步骤 | 1. 准备SQL查询语句；  2. 获取数据库连接；  3. 创建Statement对象；  4. 执行查询；  5. 遍历结果集并构建Admin对象列表；  6. 返回Admin列表。 |
| 输出参数 | List<Admin> |

##### getAdminsByRole 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAdminsByRole |
| 描述 | 根据角色获取管理员列表 |
| 输入参数 | String role |
| 步骤 | 1. 准备SQL查询语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行查询；  5. 遍历结果集并构建Admin对象列表；  6. 返回Admin列表。 |
| 输出参数 | List<Admin> |

##### updateFailedLoginAttempts 方法说明

|  |  |
| --- | --- |
| 方法名称 | updateFailedLoginAttempts |
| 描述 | 更新登录失败次数 |
| 输入参数 | int adminId, int attempts |
| 步骤 | 1. 准备SQL更新语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行SQL语句；  5. 返回执行结果。 |
| 输出参数 | boolean |

##### setLockoutUntil 方法说明

|  |  |
| --- | --- |
| 方法名称 | setLockoutUntil |
| 描述 | 设置锁定截止时间 |
| 输入参数 | int adminId, Timestamp lockoutUntil |
| 步骤 | 1. 准备SQL更新语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行SQL语句；  5. 返回执行结果。 |
| 输出参数 | boolean |

##### resetFailedLoginAttempts 方法说明

|  |  |
| --- | --- |
| 方法名称 | resetFailedLoginAttempts |
| 描述 | 重置登录失败次数 |
| 输入参数 | int adminId |
| 步骤 | 1. 准备SQL更新语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行SQL语句；  5. 返回执行结果。 |
| 输出参数 | boolean |

### AppointmentDao

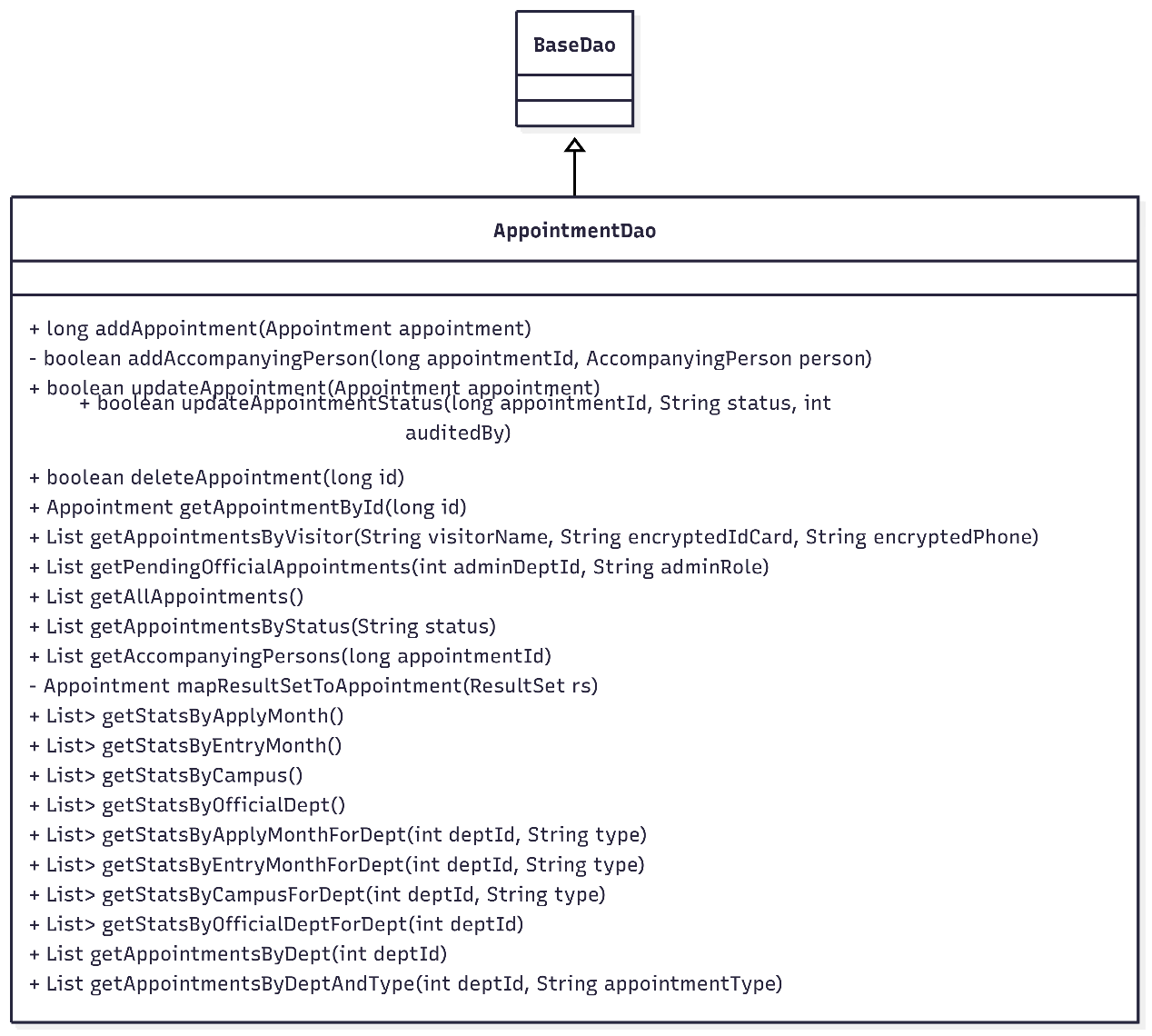


图 10 AppointmentDao

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | AppointmentDao |
| 继承或实现 | 继承 BaseDao |
| 描述 | 预约相关数据库操作的DAO类 |
| 类别 | DAO类 |
| 包 | com.zjut.passcode.dao |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| 无 | 无 | 无 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| addAppointment | public | 新增预约记录 |
| addAccompanyingPerson | private | 新增随行人员记录 |
| updateAppointment | public | 更新预约信息 |
| updateAppointmentStatus | public | 更新预约状态 |
| deleteAppointment | public | 删除预约 |
| getAppointmentById | public | 根据ID获取预约 |
| getAppointmentsByVisitor | public | 根据访客信息获取预约列表 |
| getPendingOfficialAppointments | public | 获取待审核的公务预约 |
| getAllAppointments | public | 获取所有预约 |
| getAppointmentsByStatus | public | 根据状态获取预约列表 |
| getAccompanyingPersons | public | 获取随行人员列表 |
| mapResultSetToAppointment | private | 将结果集映射为预约对象 |
| getStatsByApplyMonth | public | 按申请月度统计 |
| getStatsByEntryMonth | public | 按预约月度统计 |
| getStatsByCampus | public | 按校区统计 |
| getStatsByOfficialDept | public | 按公务访问部门统计 |
| getStatsByApplyMonthForDept | public | 按申请月度统计（指定部门） |
| getStatsByEntryMonthForDept | public | 按预约月度统计（指定部门） |
| getStatsByCampusForDept | public | 按校区统计（指定部门） |
| getStatsByOfficialDeptForDept | public | 按公务访问部门统计（指定部门） |
| getAppointmentsByDept | public | 根据部门ID获取预约列表 |
| getAppointmentsByDeptAndType | public | 根据部门ID和类型获取预约列表 |

#### 方法详细说明

##### addAppointment 方法说明

|  |  |
| --- | --- |
| 方法名称 | addAppointment |
| 描述 | 新增预约记录 |
| 输入参数 | Appointment appointment |
| 步骤 | 1. 准备SQL插入语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行SQL语句；  5. 获取生成的主键；  6. 添加随行人员记录；  7. 返回预约ID。 |
| 输出参数 | long |

##### addAccompanyingPerson 方法说明

|  |  |
| --- | --- |
| 方法名称 | addAccompanyingPerson |
| 描述 | 新增随行人员记录 |
| 输入参数 | long appointmentId, AccompanyingPerson person |
| 步骤 | 1. 准备SQL插入语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行SQL语句；  5. 返回执行结果。 |
| 输出参数 | boolean |

##### updateAppointment 方法说明

|  |  |
| --- | --- |
| 方法名称 | updateAppointment |
| 描述 | 更新预约信息 |
| 输入参数 | Appointment appointment |
| 步骤 | 1. 准备SQL更新语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行SQL语句；  5. 返回执行结果。 |
| 输出参数 | boolean |

##### updateAppointmentStatus 方法说明

|  |  |
| --- | --- |
| 方法名称 | updateAppointmentStatus |
| 描述 | 更新预约状态 |
| 输入参数 | long appointmentId, String status, int auditedBy |
| 步骤 | 1. 准备SQL更新语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行SQL语句；  5. 返回执行结果。 |
| 输出参数 | boolean |

##### deleteAppointment 方法说明

|  |  |
| --- | --- |
| 方法名称 | deleteAppointment |
| 描述 | 删除预约 |
| 输入参数 | long id |
| 步骤 | 1. 准备SQL删除语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行SQL语句；  5. 返回执行结果。 |
| 输出参数 | boolean |

##### getAppointmentById 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAppointmentById |
| 描述 | 根据ID获取预约 |
| 输入参数 | long id |
| 步骤 | 1. 准备SQL查询语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行查询；  5. 遍历结果集并构建Appointment对象；  6. 返回Appointment对象。 |
| 输出参数 | Appointment |

##### getAppointmentsByVisitor 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAppointmentsByVisitor |
| 描述 | 根据访客信息获取预约列表 |
| 输入参数 | String visitorName, String encryptedIdCard, String encryptedPhone |
| 步骤 | 1. 准备SQL查询语句；  2. 获取数据库连接；  3. 设置PreparedStatement参数；  4. 执行查询；  5. 遍历结果集并构建Appointment对象列表；  6. 返回Appointment列表。 |
| 输出参数 | List<Appointment> |

##### getPendingOfficialAppointments 方法说明

|  |  |
| --- | --- |
| 方法名称 | getPendingOfficialAppointments |
| 描述 | 获取待审核的公务预约 |
| 输入参数 | int adminDeptId, String adminRole |
| 步骤 | 1. 构建SQL查询语句；2. 根据管理员角色添加条件3. 获取数据库连接；4. 设置PreparedStatement参数；5. 执行查询；6. 遍历结果集并构建Appointment对象列表；7. 返回Appointment列表。 |
| 输出参数 | List<Appointment> |

##### getAllAppointments 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAllAppointments |
| 描述 | 获取所有预约 |
| 输入参数 | 无 |
| 步骤 | 1. 准备SQL查询语句；2. 获取数据库连接；3. 执行查询；4. 遍历结果集并构建Appointment对象列表；5. 返回Appointment列表。 |
| 输出参数 | List<Appointment> |

##### getAppointmentsByStatus 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAppointmentsByStatus |
| 描述 | 根据状态获取预约列表 |
| 输入参数 | String status |
| 步骤 | 1. 准备SQL查询语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行查询；5. 遍历结果集并构建Appointment对象列表；6. 返回Appointment列表。 |
| 输出参数 | List<Appointment> |

##### getAccompanyingPersons 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAccompanyingPersons |
| 描述 | 获取随行人员列表 |
| 输入参数 | long appointmentId |
| 步骤 | 1. 准备SQL查询语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行查询；5. 遍历结果集并构建AccompanyingPerson对象列表；6. 返回AccompanyingPerson列表。 |
| 输出参数 | List<AccompanyingPerson> |

##### mapResultSetToAppointment 方法说明

|  |  |
| --- | --- |
| 方法名称 | mapResultSetToAppointment |
| 描述 | 将结果集映射为预约对象 |
| 输入参数 | ResultSet rs |
| 步骤 | 1. 创建Appointment对象；2. 从结果集中读取各字段值；3. 设置Appointment对象的属性；4. 返回Appointment对象。 |
| 输出参数 | Appointment |

##### getStatsByApplyMonth 方法说明

|  |  |
| --- | --- |
| 方法名称 | getStatsByApplyMonth |
| 描述 | 按申请月度统计预约次数和人次 |
| 输入参数 | 无 |
| 步骤 | 1. 准备SQL统计语句；2. 获取数据库连接；3. 执行查询；4. 遍历结果集并构建统计结果；5. 返回统计结果列表。 |
| 输出参数 | List<Map<String, Object>> |

##### getStatsByEntryMonth 方法说明

|  |  |
| --- | --- |
| 方法名称 | getStatsByEntryMonth |
| 描述 | 按预约月度统计预约次数和人次 |
| 输入参数 | 无 |
| 步骤 | 1. 准备SQL统计语句；2. 获取数据库连接；3. 执行查询；4. 遍历结果集并构建统计结果；5. 返回统计结果列表。 |
| 输出参数 | List<Map<String, Object>> |

##### getStatsByCampus 方法说明

|  |  |
| --- | --- |
| 方法名称 | getStatsByCampus |
| 描述 | 按校区统计预约次数和人次 |
| 输入参数 | 无 |
| 步骤 | 1. 准备SQL统计语句；2. 获取数据库连接；3. 执行查询；4. 遍历结果集并构建统计结果；5. 返回统计结果列表。 |
| 输出参数 | List<Map<String, Object>> |

##### getStatsByOfficialDept 方法说明

|  |  |
| --- | --- |
| 方法名称 | getStatsByOfficialDept |
| 描述 | 按公务访问部门统计预约次数和人次 |
| 输入参数 | 无 |
| 步骤 | 1. 准备SQL统计语句；2. 获取数据库连接；3. 执行查询；4. 遍历结果集并构建统计结果；5. 返回统计结果列表。 |
| 输出参数 | List<Map<String, Object>> |

##### getStatsByApplyMonthForDept 方法说明

|  |  |
| --- | --- |
| 方法名称 | getStatsByApplyMonthForDept |
| 描述 | 按申请月度统计（指定部门和类型） |
| 输入参数 | int deptId, String type |
| 步骤 | 1. 准备SQL统计语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行查询；5. 遍历结果集并构建统计结果；6. 返回统计结果列表。 |
| 输出参数 | List<Map<String, Object>> |

##### getStatsByEntryMonthForDept 方法说明

|  |  |
| --- | --- |
| 方法名称 | getStatsByEntryMonthForDept |
| 描述 | 按预约月度统计（指定部门和类型） |
| 输入参数 | int deptId, String type |
| 步骤 | 1. 准备SQL统计语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行查询；5. 遍历结果集并构建统计结果；6. 返回统计结果列表。 |
| 输出参数 | List<Map<String, Object>> |

##### getStatsByCampusForDept 方法说明

|  |  |
| --- | --- |
| 方法名称 | getStatsByCampusForDept |
| 描述 | 按校区统计（指定部门和类型） |
| 输入参数 | int deptId, String type |
| 步骤 | 1. 准备SQL统计语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行查询；5. 遍历结果集并构建统计结果；6. 返回统计结果列表。 |
| 输出参数 | List<Map<String, Object>> |

##### getStatsByOfficialDeptForDept 方法说明

|  |  |
| --- | --- |
| 方法名称 | getStatsByOfficialDeptForDept |
| 描述 | 按公务访问部门统计（指定部门） |
| 输入参数 | int deptId |
| 步骤 | 1. 准备SQL统计语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行查询；5. 遍历结果集并构建统计结果；6. 返回统计结果列表。 |
| 输出参数 | List<Map<String, Object>> |

##### getAppointmentsByDept 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAppointmentsByDept |
| 描述 | 根据部门ID获取该部门的所有预约 |
| 输入参数 | int deptId |
| 步骤 | 1. 准备SQL查询语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行查询；5. 遍历结果集并构建Appointment对象列表；6. 返回Appointment列表。 |
| 输出参数 | List<Appointment> |

##### getAppointmentsByDeptAndType 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAppointmentsByDeptAndType |
| 描述 | 根据部门ID和预约类型获取预约列表 |
| 输入参数 | int deptId, String appointmentType |
| 步骤 | 1. 准备SQL查询语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行查询；5. 遍历结果集并构建Appointment对象列表；6. 返回Appointment列表。 |
| 输出参数 | List<Appointment> |

### AuditLogDao

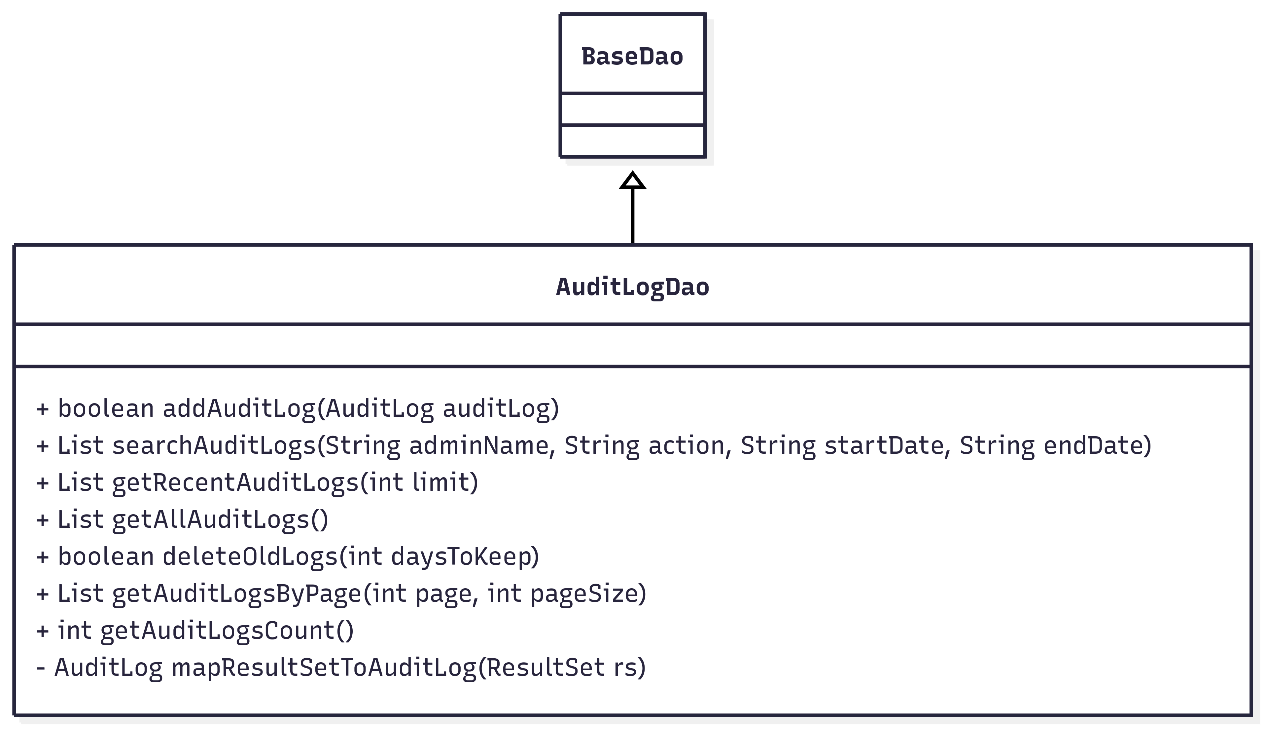


图 11 AuditLogDao

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | AuditLogDao |
| 继承或实现 | 继承 BaseDao |
| 描述 | 审计日志数据访问对象类 |
| 类别 | DAO类 |
| 包 | com.zjut.passcode.dao |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| 无 | 无 | 无 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| addAuditLog | public | 添加审计日志 |
| searchAuditLogs | public | 搜索审计日志 |
| getRecentAuditLogs | public | 获取最近的审计日志 |
| getAllAuditLogs | public | 获取所有审计日志 |
| deleteOldLogs | public | 删除旧日志 |
| getAuditLogsByPage | public | 分页获取审计日志 |
| getAuditLogsCount | public | 获取审计日志总数 |
| mapResultSetToAuditLog | private | 将结果集映射为审计日志对象 |

#### 方法详细说明

##### addAuditLog 方法说明

|  |  |
| --- | --- |
| 方法名称 | addAuditLog |
| 描述 | 添加审计日志 |
| 输入参数 | AuditLog auditLog |
| 步骤 | 1. 准备SQL插入语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行SQL语句；5. 返回执行结果。 |
| 输出参数 | boolean |

##### searchAuditLogs 方法说明

|  |  |
| --- | --- |
| 方法名称 | searchAuditLogs |
| 描述 | 搜索审计日志 |
| 输入参数 | String adminName, String action, String startDate, String endDate |
| 步骤 | 1. 构建动态SQL查询语句；2. 添加查询条件；3. 获取数据库连接；4. 设置PreparedStatement参数；5. 执行查询；6. 遍历结果集并构建AuditLog对象列表；7. 返回AuditLog列表。 |
| 输出参数 | List<AuditLog> |

##### getRecentAuditLogs 方法说明

|  |  |
| --- | --- |
| 方法名称 | getRecentAuditLogs |
| 描述 | 获取最近的审计日志 |
| 输入参数 | int limit |
| 步骤 | 1. 准备SQL查询语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行查询；5. 遍历结果集并构建AuditLog对象列表；6. 返回AuditLog列表。 |
| 输出参数 | List<AuditLog> |

##### getAllAuditLogs 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAllAuditLogs |
| 描述 | 获取所有审计日志 |
| 输入参数 | 无 |
| 步骤 | 1. 准备SQL查询语句；2. 获取数据库连接；3. 执行查询；4. 遍历结果集并构建AuditLog对象列表；5. 返回AuditLog列表。 |
| 输出参数 | List<AuditLog> |

##### deleteOldLogs 方法说明

|  |  |
| --- | --- |
| 方法名称 | deleteOldLogs |
| 描述 | 删除旧日志 |
| 输入参数 | int daysToKeep |
| 步骤 | 1. 准备SQL删除语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行SQL语句；5. 返回执行结果。 |
| 输出参数 | boolean |

##### getAuditLogsByPage 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAuditLogsByPage |
| 描述 | 分页获取审计日志 |
| 输入参数 | int page, int pageSize |
| 步骤 | 1. 准备SQL分页查询语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行查询；5. 遍历结果集并构建AuditLog对象列表；6. 返回AuditLog列表。 |
| 输出参数 | List<AuditLog> |

##### getAuditLogsCount 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAuditLogsCount |
| 描述 | 获取审计日志总数 |
| 输入参数 | 无 |
| 步骤 | 1. 准备SQL统计语句；2. 获取数据库连接；3. 执行查询；4. 获取结果集中的计数值；5. 返回计数值。 |
| 输出参数 | int |

##### mapResultSetToAuditLog 方法说明

|  |  |
| --- | --- |
| 方法名称 | mapResultSetToAuditLog |
| 描述 | 将结果集映射为审计日志对象 |
| 输入参数 | ResultSet rs |
| 步骤 | 1. 创建AuditLog对象；2. 从结果集中读取各字段值；3. 设置AuditLog对象的属性；4. 返回AuditLog对象。 |
| 输出参数 | AuditLog |

### DepartmentDao

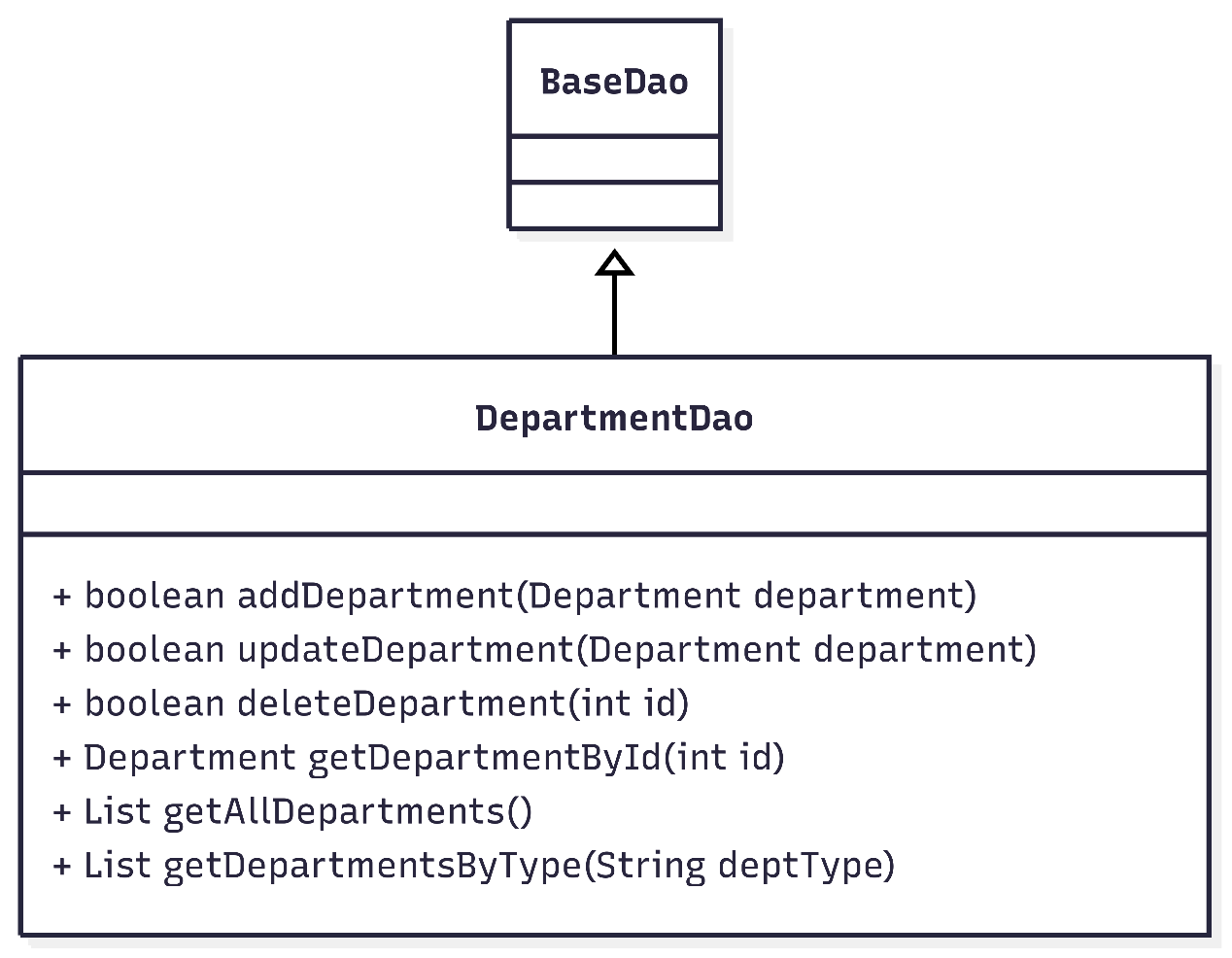


图 12 DepartmentDao

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | DepartmentDao |
| 继承或实现 | 继承 BaseDao |
| 描述 | 部门数据访问对象类 |
| 类别 | DAO类 |
| 包 | com.zjut.passcode.dao |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| 无 | 无 | 无 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| addDepartment | public | 添加部门 |
| updateDepartment | public | 更新部门信息 |
| deleteDepartment | public | 删除部门 |
| getDepartmentById | public | 根据ID获取部门 |
| getAllDepartments | public | 获取所有部门 |
| getDepartmentsByType | public | 根据类型获取部门列表 |

#### 方法详细说明

##### addDepartment 方法说明

|  |  |
| --- | --- |
| 方法名称 | addDepartment |
| 描述 | 添加部门 |
| 输入参数 | Department department |
| 步骤 | 1. 准备SQL插入语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行SQL语句；5. 返回执行结果。 |
| 输出参数 | boolean |

##### updateDepartment 方法说明

|  |  |
| --- | --- |
| 方法名称 | updateDepartment |
| 描述 | 更新部门信息 |
| 输入参数 | Department department |
| 步骤 | 1. 准备SQL更新语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 打印调试信息；5. 执行SQL语句；6. 打印执行结果；7. 返回执行结果。 |
| 输出参数 | boolean |

##### deleteDepartment 方法说明

|  |  |
| --- | --- |
| 方法名称 | deleteDepartment |
| 描述 | 删除部门 |
| 输入参数 | int id |
| 步骤 | 1. 准备SQL删除语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行SQL语句；5. 返回执行结果。 |
| 输出参数 | boolean |

##### getDepartmentById 方法说明

|  |  |
| --- | --- |
| 方法名称 | getDepartmentById |
| 描述 | 根据ID获取部门 |
| 输入参数 | int id |
| 步骤 | 1. 准备SQL查询语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行查询；5. 遍历结果集并构建Department对象；6. 返回Department对象。 |
| 输出参数 | Department |

##### getAllDepartments 方法说明

|  |  |
| --- | --- |
| 方法名称 | getAllDepartments |
| 描述 | 获取所有部门 |
| 输入参数 | 无 |
| 步骤 | 1. 准备SQL查询语句；2. 获取数据库连接；3. 创建Statement对象；4. 执行查询；5. 遍历结果集并构建Department对象列表；6. 返回Department列表。 |
| 输出参数 | List<Department> |

##### getDepartmentsByType 方法说明

|  |  |
| --- | --- |
| 方法名称 | getDepartmentsByType |
| 描述 | 根据类型获取部门列表 |
| 输入参数 | String deptType |
| 步骤 | 1. 准备SQL查询语句；2. 获取数据库连接；3. 设置PreparedStatement参数；4. 执行查询；5. 遍历结果集并构建Department对象列表；6. 返回Department列表。 |
| 输出参数 | List<Department> |

### BaseDao

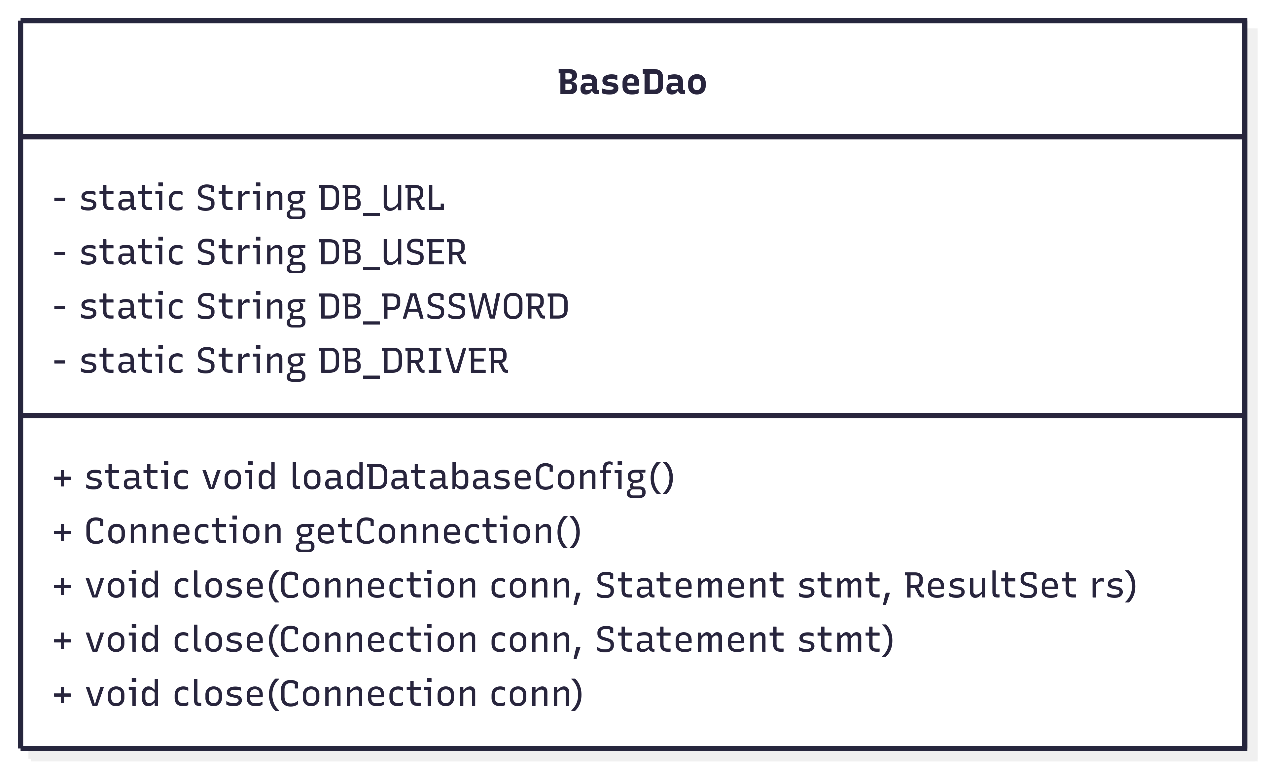


图 13 BaseDao

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | BaseDao |
| 继承或实现 | 无 |
| 描述 | 基础数据访问对象类 |
| 类别 | DAO基类 |
| 包 | com.zjut.passcode.dao |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| DB\_URL | static String | 数据库连接URL |
| DB\_USER | static String | 数据库用户名 |
| DB\_PASSWORD | static String | 数据库密码 |
| DB\_DRIVER | static String | 数据库驱动类 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| loadDatabaseConfig | private static | 加载数据库配置文件 |
| getConnection | public | 获取数据库连接 |
| close(Connection, Statement, ResultSet) | public | 关闭数据库资源（三个参数） |
| close(Connection, Statement) | public | 关闭数据库资源（两个参数） |
| close(Connection) | public | 关闭数据库连接 |

#### 方法详细说明

##### loadDatabaseConfig 方法说明

|  |  |
| --- | --- |
| 方法名称 | loadDatabaseConfig |
| 描述 | 加载数据库配置文件 |
| 输入参数 | 无 |
| 步骤 | 1. 创建Properties对象；2. 尝试从classpath加载配置文件；3. 如果失败，尝试从当前工作目录加载；4. 如果失败，尝试从上级目录加载；5. 加载配置项到静态变量；6. 验证必要配置项；7. 加载数据库驱动类。 |
| 输出参数 | void |

##### getConnection 方法说明

|  |  |
| --- | --- |
| 方法名称 | getConnection |
| 描述 | 获取数据库连接 |
| 输入参数 | 无 |
| 步骤 | 1. 使用DriverManager创建数据库连接；2. 返回Connection对象。 |
| 输出参数 | Connection |

##### close(Connection, Statement, ResultSet) 方法说明

|  |  |
| --- | --- |
| 方法名称 | close(Connection, Statement, ResultSet) |
| 描述 | 关闭数据库资源（三个参数） |
| 输入参数 | Connection conn, Statement stmt, ResultSet rs |
| 步骤 | 1. 关闭ResultSet；2. 关闭Statement；3. 关闭Connection。 |
| 输出参数 | void |

##### close(Connection, Statement) 方法说明

|  |  |
| --- | --- |
| 方法名称 | close(Connection, Statement) |
| 描述 | 关闭数据库资源（两个参数） |
| 输入参数 | Connection conn, Statement stmt |
| 步骤 | 1. 调用三个参数的close方法，传入null作为ResultSet参数。 |
| 输出参数 | void |

##### close(Connection) 方法说明

|  |  |
| --- | --- |
| 方法名称 | close(Connection) |
| 描述 | 关闭数据库连接 |
| 输入参数 | Connection conn |
| 步骤 | 1. 调用三个参数的close方法，传入null作为Statement和ResultSet参数。 |
| 输出参数 | void |

## Controller

### AdminAppointmentsServlet

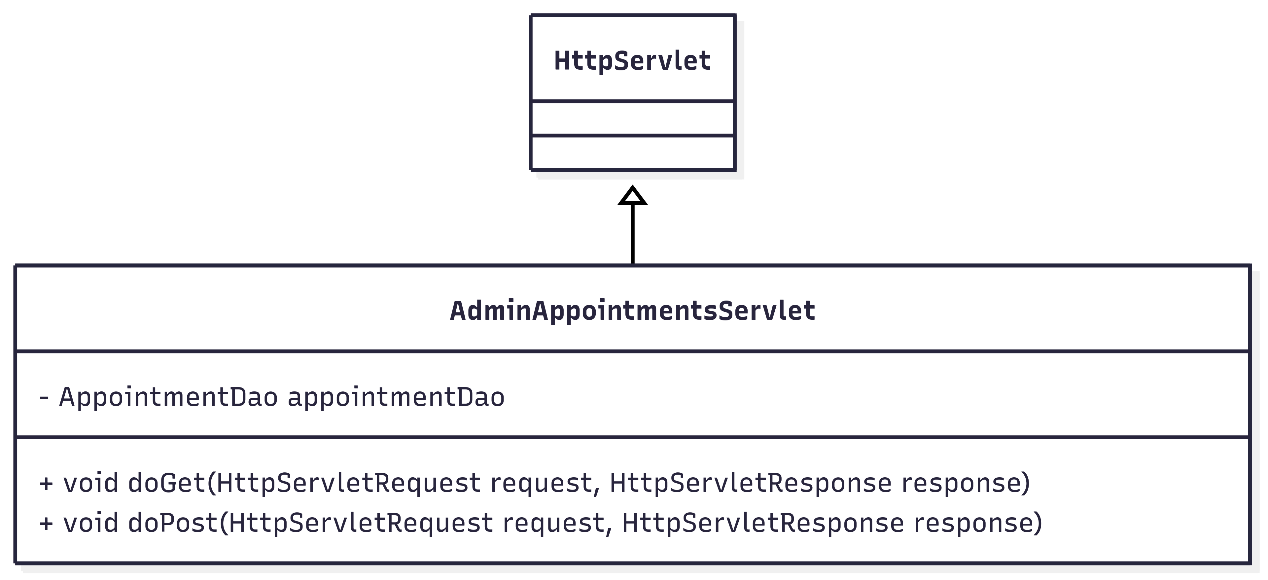


图 14 AdminAppointmentsServlet

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | AdminAppointmentsServlet |
| 继承或实现 | 继承 HttpServlet |
| 描述 | 管理员预约管理控制器 |
| 类别 | Controller类 |
| 包 | com.zjut.passcode.controller |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| appointmentDao | AppointmentDao | 预约数据访问对象 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| doGet | public | 处理GET请求，显示预约列表 |
| doPost | public | 处理POST请求，审核预约 |

#### 方法详细说明

##### doGet 方法说明

|  |  |
| --- | --- |
| 方法名称 | doGet |
| 描述 | 处理GET请求，显示预约列表 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 检查管理员登录状态；2. 根据管理员角色获取相应预约；3. 部门管理员根据权限获取预约；4. 学校管理员和系统管理员获取所有预约；5. 设置请求属性；6. 转发到预约管理页面。 |
| 输出参数 | void |

##### doPost 方法说明

|  |  |
| --- | --- |
| 方法名称 | doPost |
| 描述 | 处理POST请求，审核预约 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 检查管理员登录状态；2. 获取操作类型和预约ID；3. 获取预约信息进行权限检查；4. 部门管理员根据权限审核预约；5. 执行审核操作；6. 记录审计日志；7. 设置操作结果消息；8. 调用doGet方法刷新页面。 |
| 输出参数 | void |

### AdminAuditLogsServlet

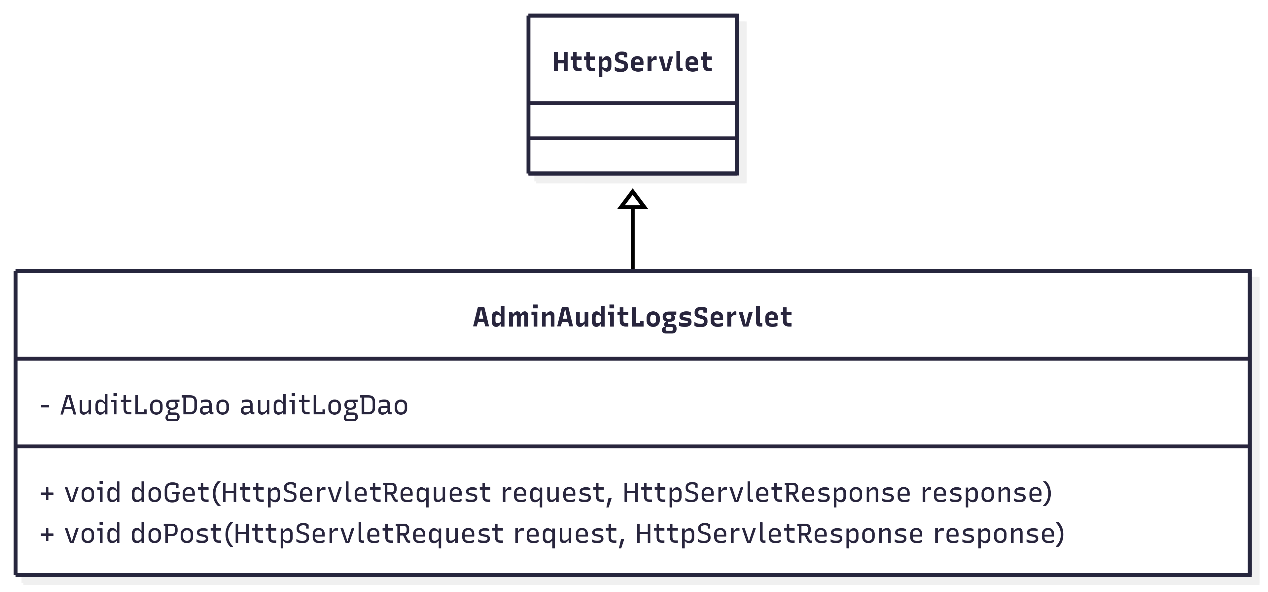


图 15 AdminAuditLogsServlet

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | AdminAuditLogsServlet |
| 继承或实现 | 继承 HttpServlet |
| 描述 | 管理员审计日志控制器 |
| 类别 | Controller |
| 包 | com.zjut.passcode.controller |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| auditLogDao | AuditLogDao | 审计日志数据访问对象 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| doGet | public | 处理GET请求，显示审计日志列表 |
| doPost | public | 处理POST请求，转发到doGet |

#### 方法详细说明

##### doGet 方法说明

|  |  |
| --- | --- |
| 方法名称 | doGet |
| 描述 | 处理GET请求，显示审计日志列表 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 检查管理员登录状态；2. 获取分页参数；3. 计算总页数；4. 分页查询审计日志；5. 设置请求属性；6. 转发到审计日志页面。 |
| 输出参数 | void |

##### doPost 方法说明

|  |  |
| --- | --- |
| 方法名称 | doPost |
| 描述 | 处理POST请求，转发到doGet |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 直接调用doGet方法处理请求。 |
| 输出参数 | void |

### AdminDepartmentsServlet

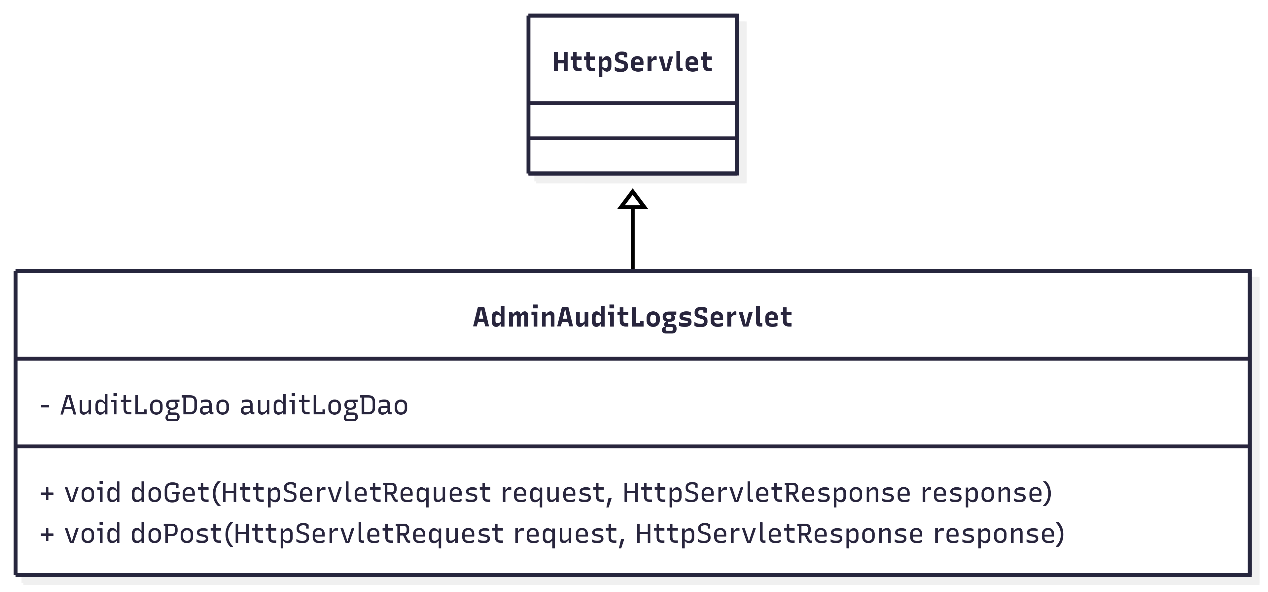


图 16 AdminDepartmentsServlet

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | AdminDepartmentsServlet |
| 继承或实现 | 继承 HttpServlet |
| 描述 | 管理员部门管理控制器 |
| 类别 | Controller |
| 包 | com.zjut.passcode.controller |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| departmentDao | DepartmentDao | 部门数据访问对象 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| doGet | public | 处理GET请求，显示部门列表 |
| doPost | public | 处理POST请求，转发到doGet |

#### 方法详细说明

##### doGet 方法说明

|  |  |
| --- | --- |
| 方法名称 | doGet |
| 描述 | 处理GET请求，显示部门列表 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 设置字符编码；2. 检查管理员登录状态；3. 获取所有部门信息；4. 设置请求属性；5. 检查操作消息；6. 转发到部门管理页面。 |
| 输出参数 | void |

##### doPost 方法说明

|  |  |
| --- | --- |
| 方法名称 | doPost |
| 描述 | 处理POST请求，转发到doGet |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 直接调用doGet方法处理请求。 |
| 输出参数 | void |

### AdminLoginServlet

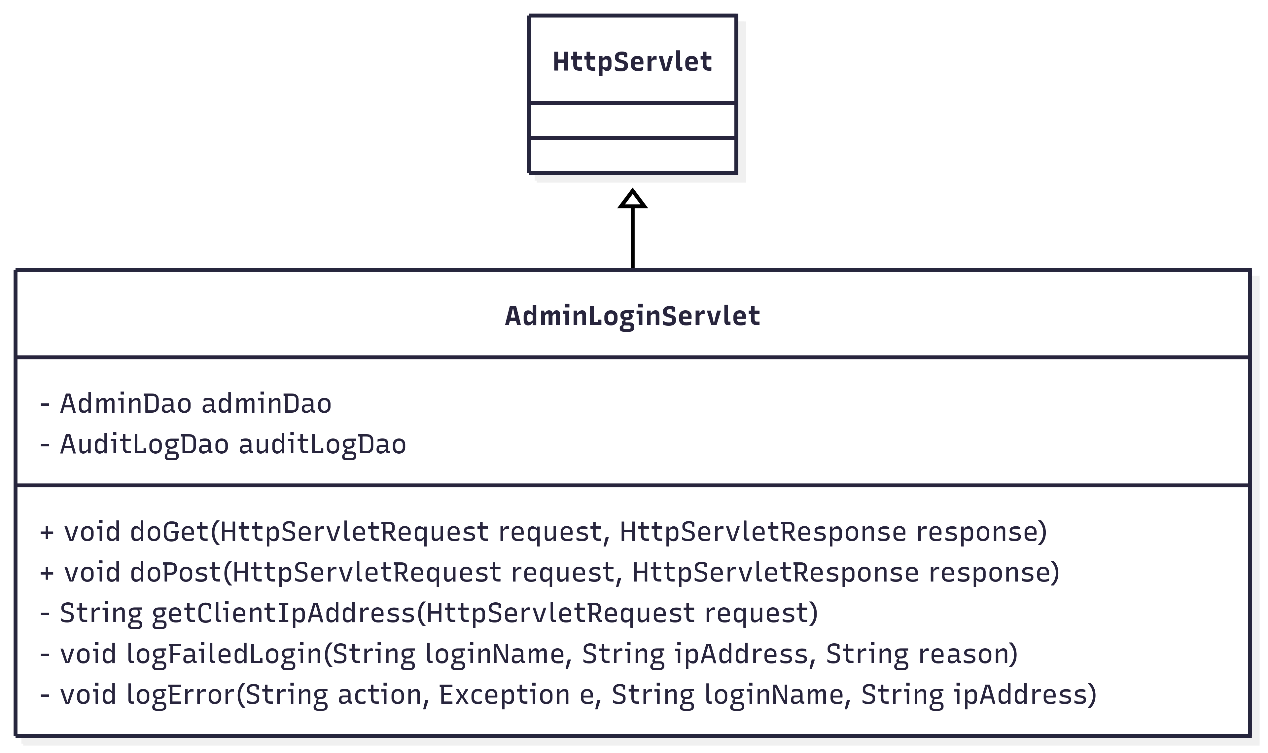


图 17 AdminLoginServlet

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | AdminLoginServlet |
| 继承或实现 | 继承 HttpServlet |
| 描述 | 管理员登录Servlet，处理管理员登录相关请求 |
| 类别 | Controller |
| 包 | com.zjut.passcode.controller |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| adminDao | AdminDao | 管理员数据访问对象 |
| auditLogDao | AuditLogDao | 审计日志数据访问对象 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| doGet | public | 处理GET请求，跳转到登录页面 |
| doPost | public | 处理POST请求，执行登录逻辑 |
| getClientIpAddress | private | 获取客户端IP地址 |
| logFailedLogin | private | 记录登录失败日志 |
| logError | private | 记录系统错误日志 |

#### 方法详细说明

##### doGet 方法说明

|  |  |
| --- | --- |
| 方法名称 | doGet |
| 描述 | 处理GET请求，跳转到登录页面 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 直接转发到管理员登录页面。 |
| 输出参数 | void |

##### doPost 方法说明

|  |  |
| --- | --- |
| 方法名称 | doPost |
| 描述 | 处理POST请求，执行登录逻辑 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 设置字符编码；2. 获取登录参数和客户端IP；3. 验证输入参数；4. 获取管理员信息；5. 检查账户锁定状态；6. 验证密码；7. 处理登录失败；8. 检查密码过期；9. 重置失败次数；10. 记录登录日志；11. 设置会话信息；12. 根据角色重定向。 |
| 输出参数 | void |

##### getClientIpAddress 方法说明

|  |  |
| --- | --- |
| 方法名称 | getClientIpAddress |
| 描述 | 获取客户端IP地址 |
| 输入参数 | HttpServletRequest request |
| 步骤 | 返回RemoteAddr。 |
| 输出参数 | String |

##### logFailedLogin 方法说明

|  |  |
| --- | --- |
| 方法名称 | logFailedLogin |
| 描述 | 记录登录失败日志 |
| 输入参数 | String loginName, String ipAddress, String reason |
| 步骤 | 1. 创建审计日志对象；2. 设置HMAC签名；3. 添加到审计日志；4. 输出日志信息。 |
| 输出参数 | void |

##### logError 方法说明

|  |  |
| --- | --- |
| 方法名称 | logError |
| 描述 | 记录系统错误日志 |
| 输入参数 | String action, Exception e, String loginName, String ipAddress |
| 步骤 | 1. 构建错误详情；2. 创建审计日志对象；3. 设置HMAC签名；4. 添加到审计日志；5. 输出错误信息到控制台。 |
| 输出参数 | void |

### AdminLogoutServlet

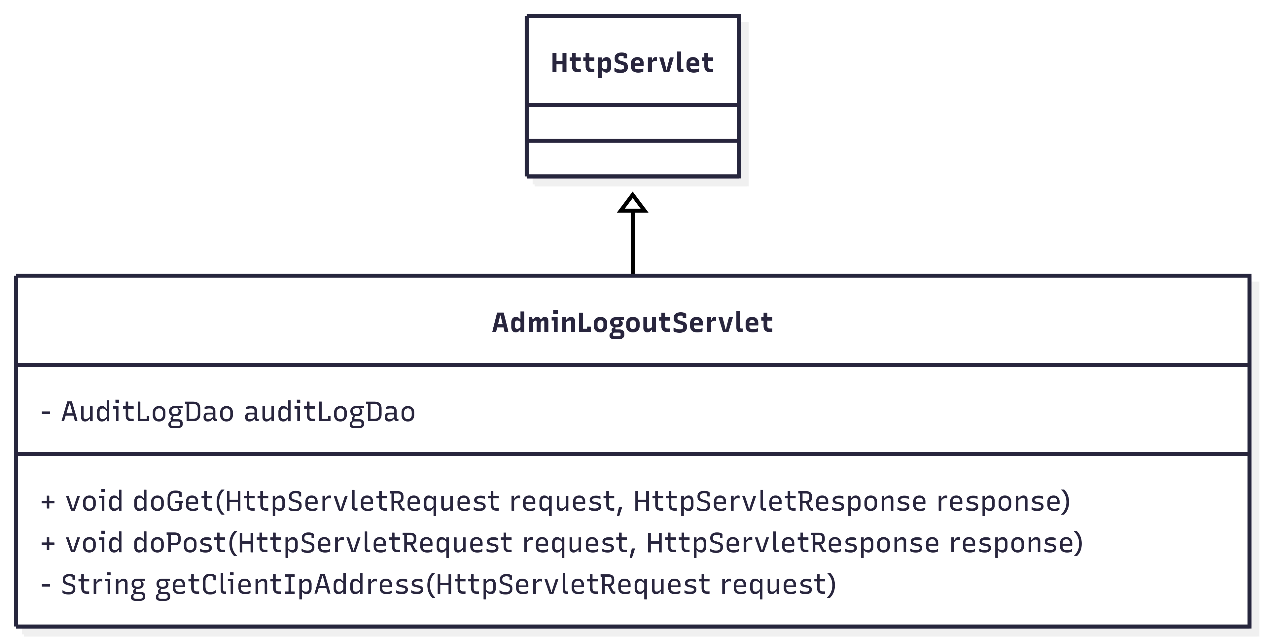


图 18 AdminLogoutServlet

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | AdminLogoutServlet |
| 继承或实现 | 继承 HttpServlet |
| 描述 | 管理员退出登录控制器 |
| 类别 | Controller |
| 包 | com.zjut.passcode.controller |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| auditLogDao | AuditLogDao | 审计日志数据访问对象 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| doGet | public | 处理GET请求，转发到doPost |
| doPost | public | 处理POST请求，执行退出登录逻辑 |
| getClientIpAddress | private | 获取客户端IP地址 |

#### 方法详细说明

##### doGet 方法说明

|  |  |
| --- | --- |
| 方法名称 | doGet |
| 描述 | 处理GET请求，转发到doPost |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 直接调用doPost方法处理请求。 |
| 输出参数 | void |

##### doPost 方法说明

|  |  |
| --- | --- |
| 方法名称 | doPost |
| 描述 | 处理POST请求，执行退出登录逻辑 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取当前会话；2. 获取管理员信息和客户端IP；3. 记录退出登录审计日志；4. 使会话失效；5. 重定向到登录页面。 |
| 输出参数 | void |

##### getClientIpAddress 方法说明

|  |  |
| --- | --- |
| 方法名称 | getClientIpAddress |
| 描述 | 获取客户端IP地址 |
| 输入参数 | HttpServletRequest request |
| 步骤 | 返回RemoteAddr。 |
| 输出参数 | String |

### AdminManageServlet

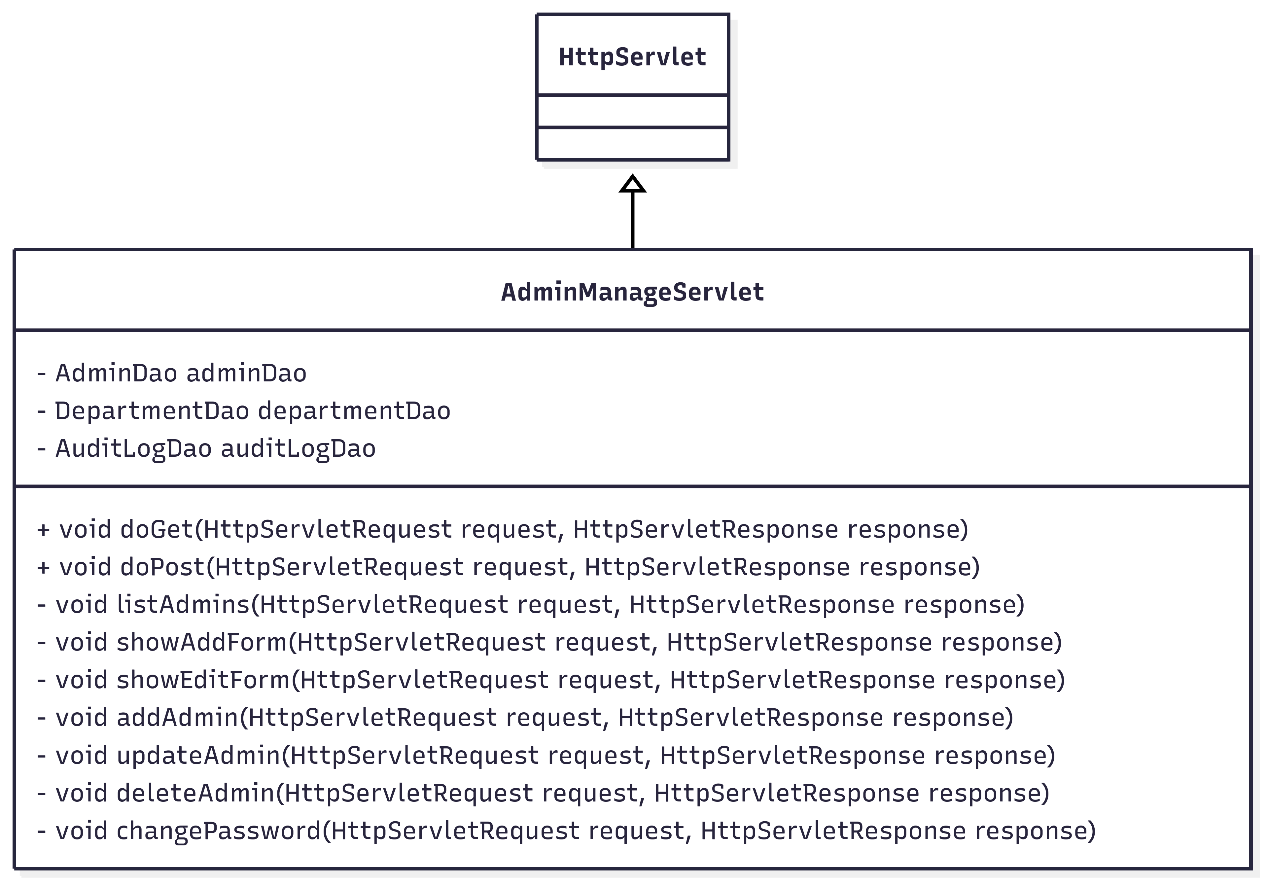


图 19 AdminManageServlet

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | AdminManageServlet |
| 继承或实现 | 继承 HttpServlet |
| 描述 | 管理员管理控制器 |
| 类别 | Controller |
| 包 | com.zjut.passcode.controller |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| adminDao | AdminDao | 管理员数据访问对象 |
| departmentDao | DepartmentDao | 部门数据访问对象 |
| auditLogDao | AuditLogDao | 审计日志数据访问对象 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| doGet | public | 处理GET请求，根据action参数执行相应操作 |
| doPost | public | 处理POST请求，根据action参数执行相应操作 |
| listAdmins | private | 显示管理员列表 |
| showAddForm | private | 显示添加管理员表单 |
| showEditForm | private | 显示编辑管理员表单 |
| addAdmin | private | 添加管理员 |
| updateAdmin | private | 更新管理员信息 |
| deleteAdmin | private | 删除管理员 |
| changePassword | private | 修改管理员密码 |

#### 方法详细说明

##### doGet 方法说明

|  |  |
| --- | --- |
| 方法名称 | doGet |
| 描述 | 处理GET请求，根据action参数执行相应操作 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 检查管理员登录状态；2. 验证管理员权限；3. 根据action参数调用相应方法；4. 执行list、add、edit或delete操作。 |
| 输出参数 | void |

##### doPost 方法说明

|  |  |
| --- | --- |
| 方法名称 | doPost |
| 描述 | 处理POST请求，根据action参数执行相应操作 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 检查管理员登录状态；2. 验证管理员权限；3. 根据action参数调用相应方法；4. 执行add、update或changePassword操作。 |
| 输出参数 | void |

##### listAdmins 方法说明

|  |  |
| --- | --- |
| 方法名称 | listAdmins |
| 描述 | 显示管理员列表 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取所有管理员信息；2. 设置请求属性；3. 转发到管理员管理页面。 |
| 输出参数 | void |

##### showAddForm 方法说明

|  |  |
| --- | --- |
| 方法名称 | showAddForm |
| 描述 | 显示添加管理员表单 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取所有部门信息；2. 设置请求属性；3. 转发到添加管理员页面。 |
| 输出参数 | void |

##### showEditForm 方法说明

|  |  |
| --- | --- |
| 方法名称 | showEditForm |
| 描述 | 显示编辑管理员表单 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取管理员ID参数；2. 根据ID获取管理员信息；3. 获取所有部门信息；4. 设置请求属性；5. 转发到编辑管理员页面。 |
| 输出参数 | void |

##### addAdmin 方法说明

|  |  |
| --- | --- |
| 方法名称 | addAdmin |
| 描述 | 添加管理员 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取表单参数；2. 验证输入参数；3. 验证密码复杂度；4. 检查登录名是否已存在；5. 加密密码和手机号；6. 创建管理员对象；7. 添加到数据库；8. 记录审计日志；9. 设置操作结果消息。 |
| 输出参数 | void |

##### updateAdmin 方法说明

|  |  |
| --- | --- |
| 方法名称 | updateAdmin |
| 描述 | 更新管理员信息 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取表单参数；2. 验证管理员ID；3. 获取管理员信息；4. 更新管理员信息；5. 加密手机号；6. 更新数据库；7. 记录审计日志；8. 设置操作结果消息。 |
| 输出参数 | void |

##### deleteAdmin 方法说明

|  |  |
| --- | --- |
| 方法名称 | deleteAdmin |
| 描述 | 删除管理员 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取管理员ID参数；2. 验证管理员ID；3. 获取管理员信息；4. 删除管理员；5. 记录审计日志；6. 设置操作结果消息。 |
| 输出参数 | void |

##### changePassword 方法说明

|  |  |
| --- | --- |
| 方法名称 | changePassword |
| 描述 | 修改管理员密码 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取管理员ID和新密码；2. 验证参数；3. 验证密码复杂度；4. 加密新密码；5. 更新数据库；6. 记录审计日志；7. 设置操作结果消息。 |
| 输出参数 | void |

### AdminReportsServlet

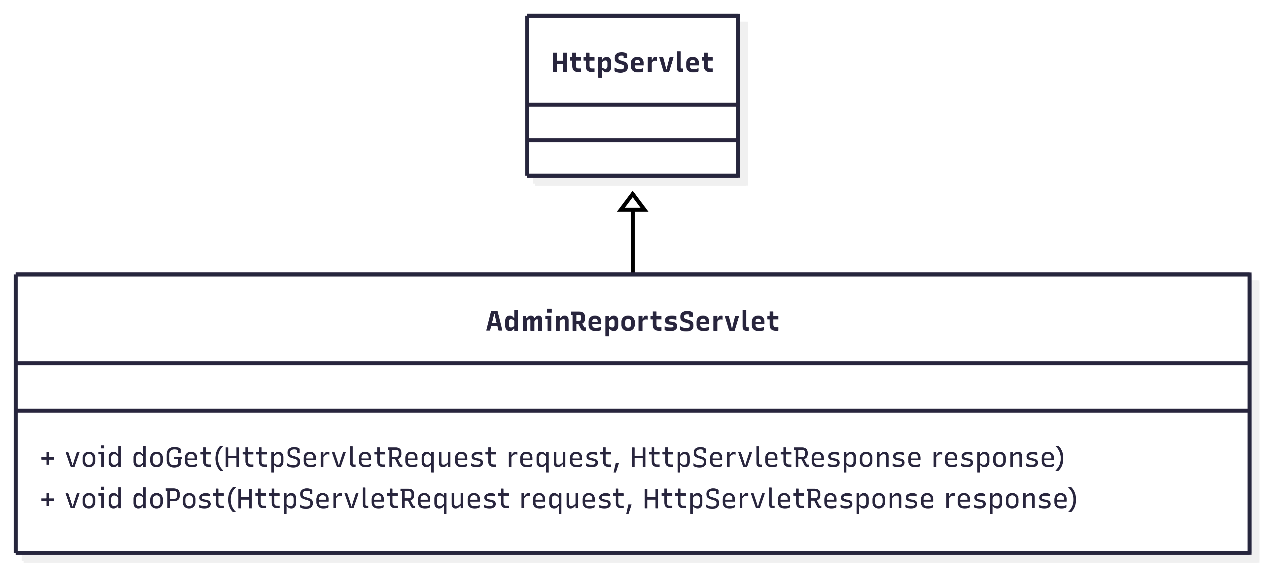


图 20 AdminReportsServlet

#### 类说明

|  |  |  |
| --- | --- | --- |
| 属性 | 内容 |  |
| 名称 | AdminReportsServlet |  |
| 继承或实现 | 继承 HttpServlet |  |
| 描述 | 管理员统计报告控制器 |  |
| 类别 | Controller |  |
| 包 | com.zjut.passcode.controller |  |
| 与其他类的调用关系 | 是 |  |

#### 属性表

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| 无 | 无 | 无 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| doGet | public | 处理GET请求，显示统计报告 |
| doPost | public | 处理POST请求，转发到doGet |

#### 方法详细说明

##### doGet 方法说明

|  |  |
| --- | --- |
| 方法名称 | doGet |
| 描述 | 处理GET请求，显示统计报告 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 检查管理员登录状态；2. 获取管理员信息和权限；3. 根据管理员角色获取相应统计数据；4. 学校/系统管理员获取所有统计数据；5. 部门管理员根据权限获取本部门统计数据；6. 设置请求属性；7. 转发到统计报告页面。 |
| 输出参数 | void |

##### doPost 方法说明

|  |  |
| --- | --- |
| 方法名称 | doPost |
| 描述 | 处理POST请求，转发到doGet |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 直接调用doGet方法处理请求。 |
| 输出参数 | void |

### AdminUsersServlet

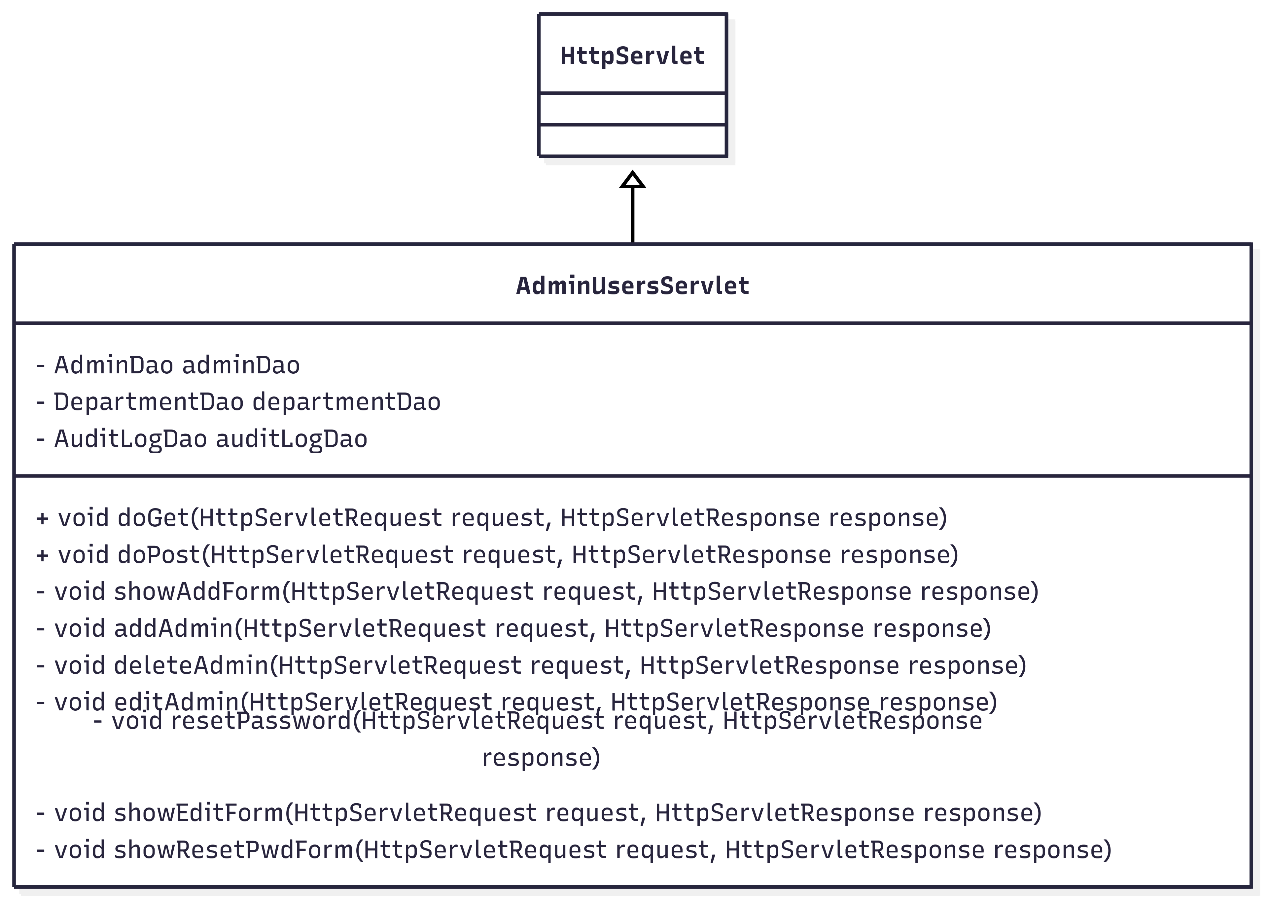


图 21 AdminUsersServlet

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | AdminUsersServlet |
| 继承或实现 | 继承 HttpServlet |
| 描述 | 管理员用户管理控制器 |
| 类别 | Controller |
| 包 | com.zjut.passcode.controller |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| adminDao | AdminDao | 管理员数据访问对象 |
| departmentDao | DepartmentDao | 部门数据访问对象 |
| auditLogDao | AuditLogDao | 审计日志数据访问对象 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| doGet | public | 处理GET请求，显示用户管理页面 |
| doPost | public | 处理POST请求，执行用户管理操作 |
| showAddForm | private | 显示添加管理员表单 |
| addAdmin | private | 添加新管理员 |
| deleteAdmin | private | 删除管理员 |
| editAdmin | private | 编辑管理员信息 |
| resetPassword | private | 重置管理员密码 |
| showEditForm | private | 显示编辑管理员表单 |
| showResetPwdForm | private | 显示重置密码表单 |

#### 方法详细说明

##### doGet 方法说明

|  |  |
| --- | --- |
| 方法名称 | doGet |
| 描述 | 处理GET请求，显示用户管理页面 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 检查管理员登录状态；2. 验证管理员权限；3. 根据action参数执行相应操作；4. 获取管理员列表；5. 转发到用户管理页面。 |
| 输出参数 | void |

##### doPost 方法说明

|  |  |
| --- | --- |
| 方法名称 | doPost |
| 描述 | 处理POST请求，执行用户管理操作 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 检查管理员登录状态；2. 验证管理员权限；3. 根据action参数执行相应操作；4. 调用doGet方法刷新页面。 |
| 输出参数 | void |

##### showAddForm 方法说明

|  |  |
| --- | --- |
| 方法名称 | showAddForm |
| 描述 | 显示添加管理员表单 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取所有部门列表；2. 设置departments属性；3. 转发到用户管理页面。 |
| 输出参数 | void |

##### addAdmin 方法说明

|  |  |
| --- | --- |
| 方法名称 | addAdmin |
| 描述 | 添加新管理员 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取表单参数；2. 验证权限和输入数据；3. 检查密码复杂度；4. 检查登录名是否已存在；5. 加密密码和手机号；6. 创建新管理员对象；7. 保存到数据库；8. 记录审计日志。 |
| 输出参数 | void |

##### deleteAdmin 方法说明

|  |  |
| --- | --- |
| 方法名称 | deleteAdmin |
| 描述 | 删除管理员 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取要删除的用户ID；2. 检查是否删除自己；3. 获取用户信息；4. 验证权限；5. 执行删除操作；6. 设置结果消息。 |
| 输出参数 | void |

##### editAdmin 方法说明

|  |  |
| --- | --- |
| 方法名称 | editAdmin |
| 描述 | 编辑管理员信息 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取表单参数；2. 获取要编辑的用户信息；3. 验证权限；4. 验证输入数据；5. 更新用户信息；6. 保存到数据库；7. 设置结果消息。 |
| 输出参数 | void |

##### resetPassword 方法说明

|  |  |
| --- | --- |
| 方法名称 | resetPassword |
| 描述 | 重置管理员密码 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取用户ID和新密码；2. 验证密码不为空；3. 验证两次密码一致；4. 验证密码复杂度；5. 加密新密码；6. 更新数据库；7. 记录审计日志。 |
| 输出参数 | void |

##### showEditForm 方法说明

|  |  |
| --- | --- |
| 方法名称 | showEditForm |
| 描述 | 显示编辑管理员表单 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取用户ID；2. 获取用户信息；3. 获取所有部门列表；4. 设置属性；5. 转发到编辑页面。 |
| 输出参数 | void |

##### showResetPwdForm 方法说明

|  |  |
| --- | --- |
| 方法名称 | showResetPwdForm |
| 描述 | 显示重置密码表单 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取用户ID；2. 获取用户信息；3. 验证用户存在；4. 设置resetUser属性；5. 转发到重置密码页面。 |
| 输出参数 | void |

### AppointmentApplyServlet

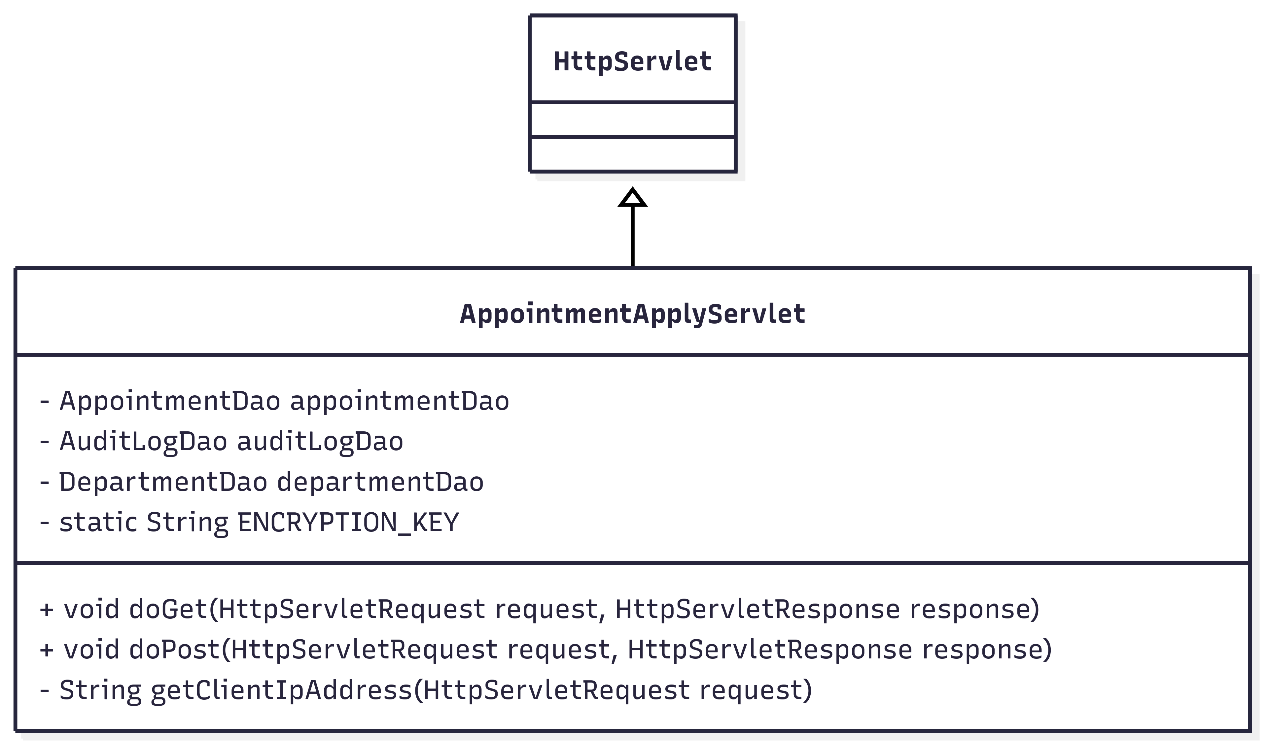


图 22 AppointmentApplyServlet

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | AppointmentApplyServlet |
| 继承或实现 | 继承 HttpServlet |
| 描述 | 预约申请控制器 |
| 类别 | Controller |
| 包 | com.zjut.passcode.controller |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| appointmentDao | AppointmentDao | 预约数据访问对象 |
| auditLogDao | AuditLogDao | 审计日志数据访问对象 |
| departmentDao | DepartmentDao | 部门数据访问对象 |
| ENCRYPTION\_KEY | static String | 加密密钥常量 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| doGet | public | 处理GET请求，显示预约申请表单 |
| doPost | public | 处理POST请求，提交预约申请 |
| getClientIpAddress | private | 获取客户端IP地址 |

#### 方法详细说明

##### doGet 方法说明

|  |  |
| --- | --- |
| 方法名称 | doGet |
| 描述 | 处理GET请求，显示预约申请表单 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 转发到预约申请表单页面。 |
| 输出参数 | void |

##### doPost 方法说明

|  |  |
| --- | --- |
| 方法名称 | doPost |
| 描述 | 处理POST请求，提交预约申请 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取表单数据；2. 验证必填字段；3. 验证手机号格式；4. 验证身份证号格式；5. 解析进校时间；6. 验证访问日期；7. 创建预约对象；8. 加密敏感信息；9. 保存预约到数据库；10. 记录审计日志；11. 跳转到通行码页面。 |
| 输出参数 | void |

##### getClientIpAddress 方法说明

|  |  |
| --- | --- |
| 方法名称 | getClientIpAddress |
| 描述 | 获取客户端IP地址 |
| 输入参数 | HttpServletRequest request |
| 步骤 | 返回getRemoteAddr()。 |
| 输出参数 | String |

### AppointmentQueryServlet

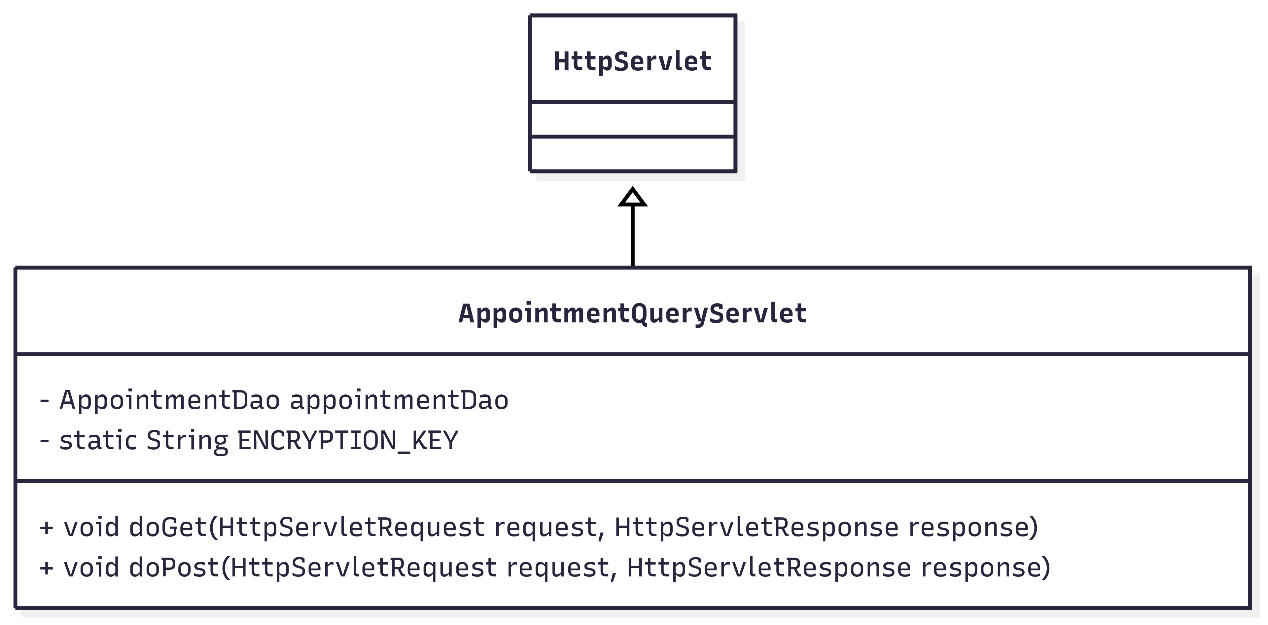


图 23 AppointmentQueryServlet

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | AppointmentQueryServlet |
| 继承或实现 | 继承 HttpServlet |
| 描述 | 预约查询控制器 |
| 类别 | Controller |
| 包 | com.zjut.passcode.controller |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| appointmentDao | AppointmentDao | 预约数据访问对象 |
| ENCRYPTION\_KEY | static String | 加密密钥常量 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| doGet | public | 处理GET请求，显示预约查询页面 |
| doPost | public | 处理POST请求，执行预约查询 |

#### 方法详细说明

##### doGet 方法说明

|  |  |
| --- | --- |
| 方法名称 | doGet |
| 描述 | 处理GET请求，显示预约查询页面 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 转发到我的预约页面。 |
| 输出参数 | void |

##### doPost 方法说明

|  |  |
| --- | --- |
| 方法名称 | doPost |
| 描述 | 处理POST请求，执行预约查询 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取查询参数（访客姓名、手机号、身份证号）；2. 验证至少提供一个查询条件；3. 加密手机号和身份证号；4. 调用DAO查询预约记录；5. 解密敏感信息用于显示；6. 设置查询结果到request属性；7. 转发到结果页面。 |
| 输出参数 | void |

### DepartmentOperationServlet

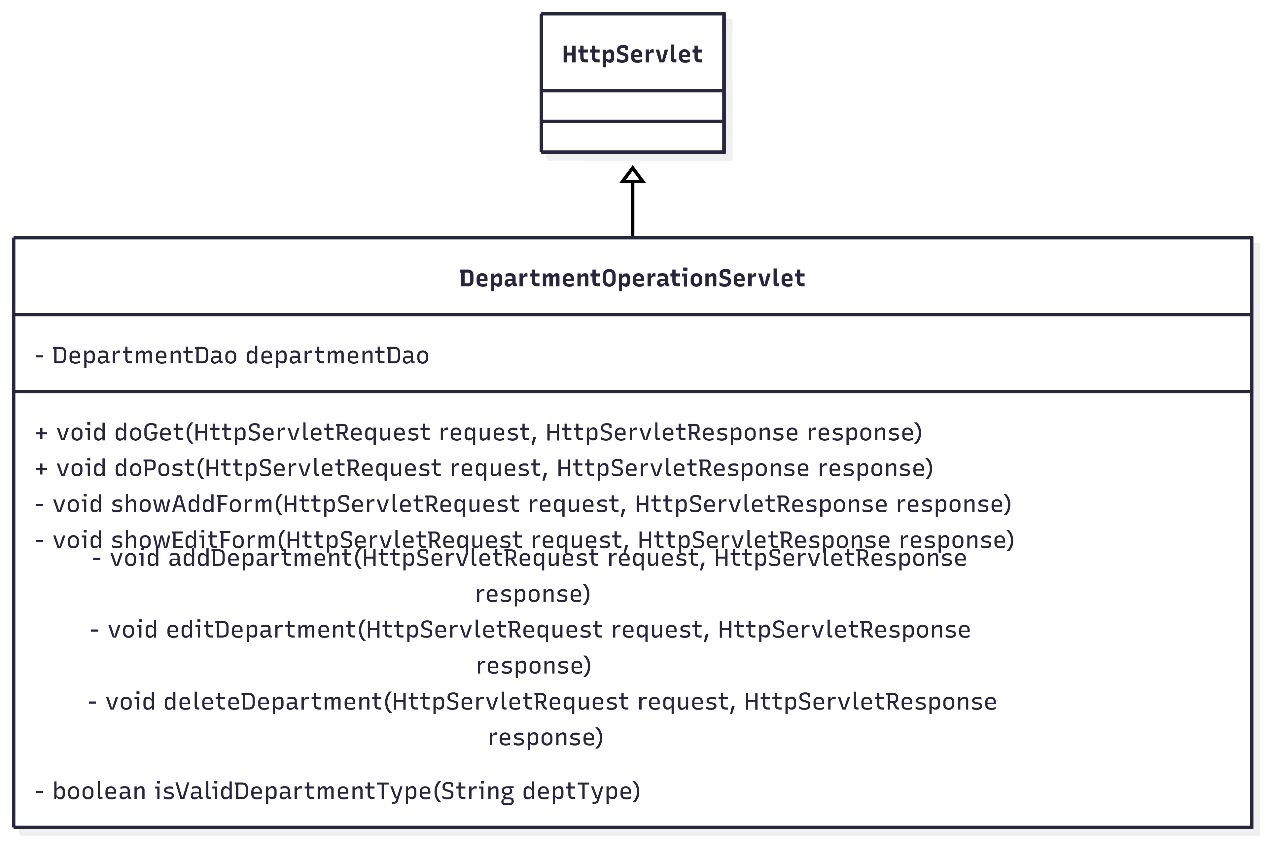


图 24 DepartmentOperationServlet

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | DepartmentOperationServlet |
| 继承或实现 | 继承 HttpServlet |
| 描述 | 部门操作控制器 |
| 类别 | Controller |
| 包 | com.zjut.passcode.controller |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| departmentDao | DepartmentDao | 部门数据访问对象 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| doGet | public | 处理GET请求，根据路径执行相应操作 |
| doPost | public | 处理POST请求，根据路径执行相应操作 |
| showAddForm | private | 显示添加部门表单 |
| showEditForm | private | 显示编辑部门表单 |
| addDepartment | private | 添加新部门 |
| editDepartment | private | 编辑部门信息 |
| deleteDepartment | private | 删除部门 |
| isValidDepartmentType | private | 验证部门类型是否有效 |

#### 方法详细说明

##### doGet 方法说明

|  |  |
| --- | --- |
| 方法名称 | doGet |
| 描述 | 处理GET请求，根据路径执行相应操作 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 检查管理员登录状态；2. 解析路径信息；3. 根据操作类型调用相应方法；4. 处理添加、编辑、删除操作。 |
| 输出参数 | void |

##### doPost 方法说明

|  |  |
| --- | --- |
| 方法名称 | doPost |
| 描述 | 处理POST请求，根据路径执行相应操作 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 检查管理员登录状态；2. 解析路径信息；3. 根据操作类型调用相应方法；4. 处理添加、编辑操作。 |
| 输出参数 | void |

##### showAddForm 方法说明

|  |  |
| --- | --- |
| 方法名称 | showAddForm |
| 描述 | 显示添加部门表单 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 转发到添加部门页面。 |
| 输出参数 | void |

##### showEditForm 方法说明

|  |  |
| --- | --- |
| 方法名称 | showEditForm |
| 描述 | 显示编辑部门表单 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取部门ID参数；2. 验证ID不为空；3. 根据ID获取部门信息；4. 验证部门存在；5. 设置部门属性；6. 转发到编辑页面。 |
| 输出参数 | void |

##### addDepartment 方法说明

|  |  |
| --- | --- |
| 方法名称 | addDepartment |
| 描述 | 添加新部门 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取表单参数；2. 验证必填字段；3. 去除空白字符；4. 验证部门类型；5. 创建部门对象；6. 保存到数据库；7. 处理成功或失败结果。 |
| 输出参数 | void |

##### editDepartment 方法说明

|  |  |
| --- | --- |
| 方法名称 | editDepartment |
| 描述 | 编辑部门信息 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取表单参数；2. 验证必填字段；3. 解析部门ID；4. 去除空白字符；5. 验证部门类型；6. 获取现有部门信息；7. 更新部门对象；8. 保存到数据库；9. 处理结果。 |
| 输出参数 | void |

##### deleteDepartment 方法说明

|  |  |
| --- | --- |
| 方法名称 | deleteDepartment |
| 描述 | 删除部门 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取部门ID参数；2. 验证ID不为空；3. 解析部门ID；4. 检查部门是否存在；5. 执行删除操作；6. 处理成功或失败结果。 |
| 输出参数 | void |

##### isValidDepartmentType 方法说明

|  |  |
| --- | --- |
| 方法名称 | isValidDepartmentType |
| 描述 | 验证部门类型是否有效 |
| 输入参数 | String deptType |
| 步骤 | 检查部门类型是否为"行政部门"、"直属部门"或"学院"。 |
| 输出参数 | boolean |

### PassCodeViewServlet

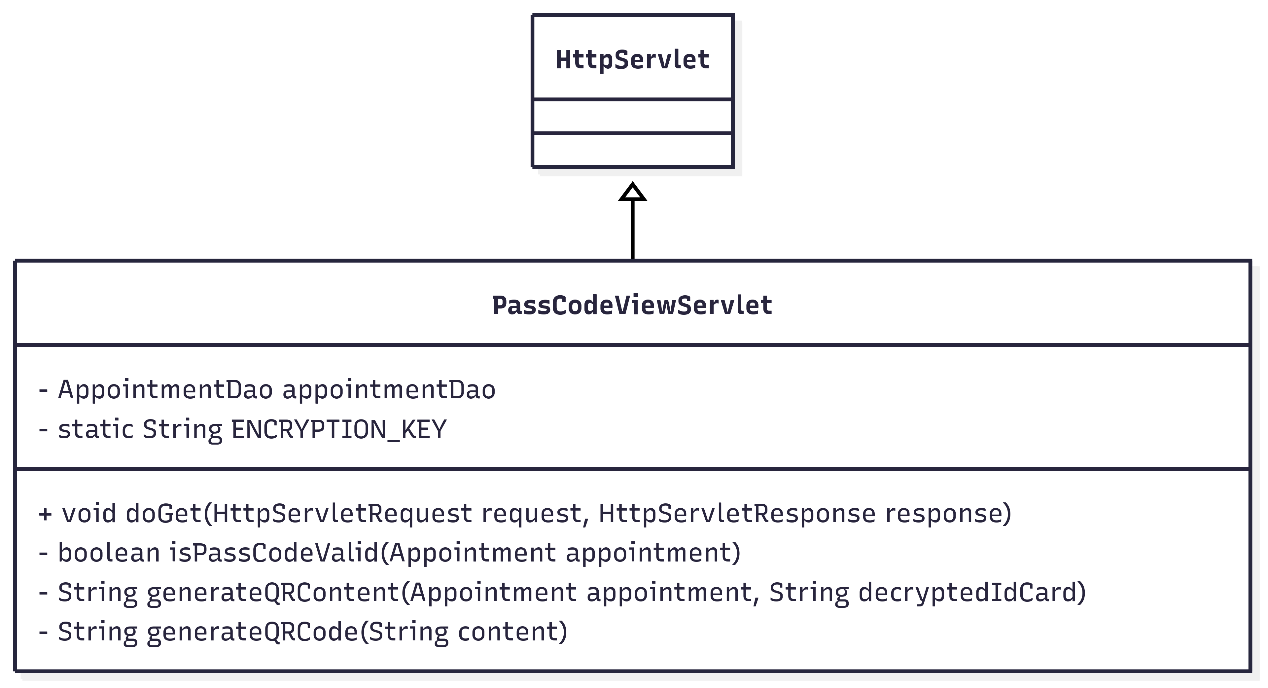


图 25 PassCodeViewServlet

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | PassCodeViewServlet |
| 继承或实现 | 继承 HttpServlet |
| 描述 | 通行码查看控制器 |
| 类别 | Controller |
| 包 | com.zjut.passcode.controller |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| appointmentDao | AppointmentDao | 预约数据访问对象 |
| ENCRYPTION\_KEY | static String | 加密密钥常量 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| doGet | public | 处理GET请求，显示通行码页面 |
| isPassCodeValid | private | 检查通行码是否有效 |
| generateQRContent | private | 生成二维码内容 |
| generateQRCode | private | 生成二维码图片 |

#### 方法详细说明

##### doGet 方法说明

|  |  |
| --- | --- |
| 方法名称 | doGet |
| 描述 | 处理GET请求，显示通行码页面 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取预约ID参数；2. 验证ID不为空；3. 根据ID获取预约信息；4. 解密敏感信息；5. 检查通行码有效性；6. 生成二维码内容；7. 生成二维码图片；8. 设置请求属性；9. 转发到通行码页面。 |
| 输出参数 | void |

##### isPassCodeValid 方法说明

|  |  |
| --- | --- |
| 方法名称 | isPassCodeValid |
| 描述 | 检查通行码是否有效 |
| 输入参数 | Appointment appointment |
| 步骤 | 1. 检查预约状态是否为APPROVED；2. 获取当前时间和进校时间；3. 计算有效时间范围（进校当天0点到23:59:59）；4. 判断当前时间是否在有效范围内。 |
| 输出参数 | boolean |

##### generateQRContent 方法说明

|  |  |
| --- | --- |
| 方法名称 | generateQRContent |
| 描述 | 生成二维码内容 |
| 输入参数 | Appointment appointment, String decryptedIdCard |
| 步骤 | 1. 创建StringBuilder对象；2. 添加姓名（脱敏处理）；3. 添加身份证号（脱敏处理）；4. 添加校区信息；5. 添加进校时间；6. 添加预约类型；7. 添加状态信息；8. 添加生成时间；9. 返回完整内容字符串。 |
| 输出参数 | String |

##### generateQRCode 方法说明

|  |  |
| --- | --- |
| 方法名称 | generateQRCode |
| 描述 | 生成二维码图片 |
| 输入参数 | String content |
| 步骤 | 1. 创建QRCodeWriter对象；2. 编码内容生成BitMatrix；3. 创建BufferedImage对象；4. 遍历像素设置黑白颜色；5. 将图片写入ByteArrayOutputStream；6. 转换为Base64编码；7. 返回完整的data URL。 |
| 输出参数 | String |

### QRCodeServlet

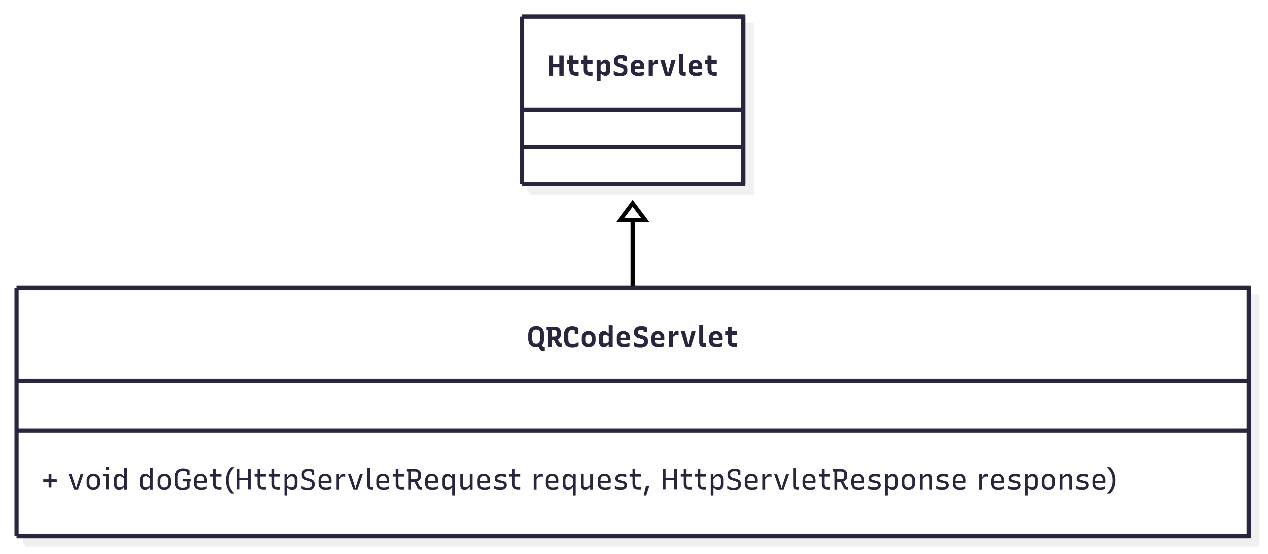


图 26 QRCodeServlet

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | QRCodeServlet |
| 继承或实现 | 继承 HttpServlet |
| 描述 | 二维码生成控制器 |
| 类别 | Controller |
| 包 | com.zjut.passcode.controller |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| 无属性 | - | 该类没有定义任何属性 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| doGet | public | 处理GET请求，生成二维码图片 |

#### 方法详细说明

##### doGet 方法说明

|  |  |
| --- | --- |
| 方法名称 | doGet |
| 描述 | 处理GET请求，生成二维码图片 |
| 输入参数 | HttpServletRequest request, HttpServletResponse response |
| 步骤 | 1. 获取content参数；2. 验证内容不为空；3. 创建QRCodeWriter对象；4. 编码内容生成BitMatrix；5. 创建BufferedImage对象；6. 遍历像素设置黑白颜色；7. 设置响应内容类型为image/png；8. 将图片写入响应输出流。 |
| 输出参数 | void |

## Util

### CryptoUtil

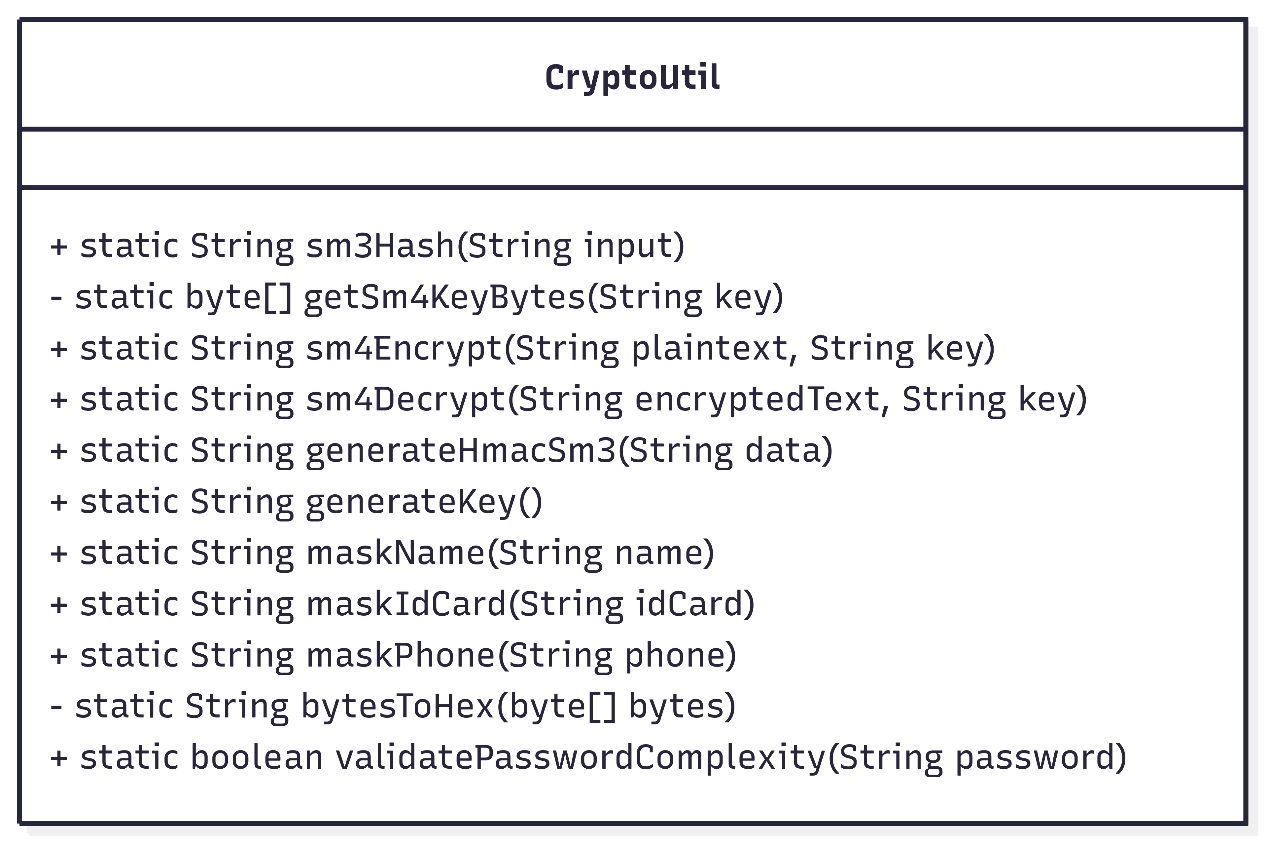


图 27 CryptoUtil

#### 类说明

|  |  |
| --- | --- |
| 属性 | 内容 |
| 名称 | CryptoUtil |
| 继承或实现 | 无 |
| 描述 | 加密工具类，提供SM3/SM4加密、脱敏处理等功能 |
| 类别 | Util |
| 包 | com.zjut.passcode.util |
| 与其他类的调用关系 | 是 |

#### 属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| 无实例属性 | - | 工具类，所有方法都是静态方法 |

#### 主要方法

|  |  |  |
| --- | --- | --- |
| 名称 | 属性 | 描述 |
| sm3Hash | public static | SM3哈希运算 |
| getSm4KeyBytes | private static | 获取SM4密钥字节数组 |
| sm4Encrypt | public static | SM4加密 |
| sm4Decrypt | public static | SM4解密 |
| generateHmacSm3 | public static | 生成HMAC-SM3签名 |
| generateKey | public static | 生成随机密钥 |
| maskName | public static | 姓名脱敏处理 |
| maskIdCard | public static | 身份证号脱敏处理 |
| maskPhone | public static | 手机号脱敏处理 |
| bytesToHex | private static | 字节数组转十六进制 |
| validatePasswordComplexity | public static | 密码复杂度校验 |

#### 方法详细说明

##### sm3Hash 方法说明

|  |  |
| --- | --- |
| 方法名称 | sm3Hash |
| 描述 | 对输入字符串进行SM3哈希运算 |
| 输入参数 | String input |
| 步骤 | 1. 创建SM3Digest对象；2. 将输入字符串转换为UTF-8字节数组；3. 更新摘要数据；4. 生成哈希值；5. 将哈希值转换为十六进制字符串；6. 返回哈希结果。 |
| 输出参数 | String |

##### getSm4KeyBytes 方法说明

|  |  |
| --- | --- |
| 方法名称 | getSm4KeyBytes |
| 描述 | 获取SM4加密算法所需的16字节密钥 |
| 输入参数 | String key |
| 步骤 | 1. 将密钥字符串转换为UTF-8字节数组；2. 检查密钥长度是否为16字节；3. 如果长度不正确则抛出异常；4. 返回密钥字节数组。 |
| 输出参数 | byte[] |

##### sm4Encrypt 方法说明

|  |  |
| --- | --- |
| 方法名称 | sm4Encrypt |
| 描述 | 使用SM4算法对明文进行加密 |
| 输入参数 | String plaintext, String key |
| 步骤 | 1. 获取16字节密钥；2. 将明文转换为UTF-8字节数组；3. 创建SM4加密器并初始化；4. 执行加密操作；5. 将加密结果转换为Base64字符串；6. 返回加密后的字符串。 |
| 输出参数 | String |

##### sm4Decrypt 方法说明

|  |  |
| --- | --- |
| 方法名称 | sm4Decrypt |
| 描述 | 使用SM4算法对密文进行解密 |
| 输入参数 | String encryptedText, String key |
| 步骤 | 1. 获取16字节密钥；2. 将Base64密文解码为字节数组；3. 创建SM4解密器并初始化；4. 执行解密操作；5. 将解密结果转换为UTF-8字符串；6. 返回解密后的明文。 |
| 输出参数 | String |

##### generateHmacSm3 方法说明

|  |  |
| --- | --- |
| 方法名称 | generateHmacSm3 |
| 描述 | 生成HMAC-SM3签名 |
| 输入参数 | String data |
| 步骤 | 1. 定义固定密钥；2. 将密钥与数据组合；3. 对组合数据进行SM3哈希运算；4. 返回哈希结果作为签名。 |
| 输出参数 | String |

##### generateKey 方法说明

|  |  |
| --- | --- |
| 方法名称 | generateKey |
| 描述 | 生成随机128位密钥 |
| 输入参数 | 无 |
| 步骤 | 1. 创建AES密钥生成器；2. 初始化128位密钥长度；3. 生成密钥；4. 将密钥编码为Base64字符串；5. 返回Base64编码的密钥。 |
| 输出参数 | String |

##### maskName 方法说明

|  |  |
| --- | --- |
| 方法名称 | maskName |
| 描述 | 对姓名进行脱敏处理 |
| 输入参数 | String name |
| 步骤 | 1. 检查姓名是否为空或长度小于等于1；2. 如果长度为2，则中间用替换；3. 如果长度大于2，则保留首尾字符，中间用替换；4. 返回脱敏后的姓名。 |
| 输出参数 | String |

##### maskIdCard 方法说明

|  |  |
| --- | --- |
| 方法名称 | maskIdCard |
| 描述 | 对身份证号进行脱敏处理 |
| 输入参数 | String idCard |
| 步骤 | 1. 检查身份证号是否为空或长度小于8；2. 保留前6位和后4位；3. 中间8位用替换；4. 返回脱敏后的身份证号。 |
| 输出参数 | String |

maskPhone

|  |  |
| --- | --- |
| 方法名称 | maskPhone |
| 描述 | 对手机号进行脱敏处理 |
| 输入参数 | String phone |
| 步骤 | 1. 检查手机号是否为空或长度小于7；2. 保留前3位和后4位；3. 中间4位用替换；4. 返回脱敏后的手机号。 |
| 输出参数v | String |

##### bytesToHex 方法说明

|  |  |
| --- | --- |
| 方法名称 | bytesToHex |
| 描述 | 字节数组转为十六进制字符串 |
| 输入参数 | byte[] bytes |
| 步骤 | 1. 创建StringBuilder对象；2. 遍历字节数组；3. 将每个字节转换为两位十六进制字符串；4. 拼接所有十六进制字符串；5. 返回最终的十六进制字符串。 |
| 输出参数 | String |

##### validatePasswordComplexity 方法说明

|  |  |
| --- | --- |
| 方法名称 | validatePasswordComplexity |
| 描述 | 校验密码复杂度，要求包含数字、大小写字母和特殊字符，且长度不少于8位 |
| 输入参数 | String password |
| 步骤 | 1. 检查密码是否为空或长度小于8；2. 遍历密码字符；3. 检查是否包含数字、小写字母、大写字母、特殊字符；4. 如果所有条件都满足则返回true，否则返回false。 |
| 输出参数 | boolean |

# 数据库设计

## 数据库表概念结构设计

### 部门（Department）

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 意义 | 类型 | 值域/约束 |
| id | 部门主键ID | INT | 自增，主键 |
| dept\_no | 部门编号 | VARCHAR(50) | 唯一，非空 |
| dept\_name | 部门名称 | VARCHAR(100) | 非空 |
| dept\_type | 部门类型 | ENUM | '行政部门'/'直属部门'/'学院'，非空 |

### 管理员（Admin）

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 意义 | 类型 | 值域/约束 |
| id | 管理员主键ID | INT | 自增，主键 |
| login\_name | 登录名 | VARCHAR(50) | 唯一，非空 |
| password\_hash | 密码（SM3加密） | VARCHAR(255) | 非空 |
| full\_name | 姓名 | VARCHAR(50) | 非空 |
| dept\_id | 所在部门ID | INT | 外键，可空 |
| dept\_no | 所在部门编号 | VARCHAR(50) | 可空 |
| dept\_name | 所在部门名称 | VARCHAR(100) | 可空 |
| phone | 联系电话（加密） | VARCHAR(255) | 非空 |
| role | 角色 | ENUM | 'SYSTEM\_ADMIN'等，非空 |
| password\_last\_changed | 密码最后修改时间 | TIMESTAMP | 默认当前时间 |
| failed\_login\_attempts | 连续登录失败次数 | INT | 默认0 |
| lockout\_until | 账户锁定截止时间 | TIMESTAMP | 可空 |
| can\_manage\_public\_appointment | 能否管理社会公众预约 | TINYINT(1) | 默认0 |
| can\_report\_public\_appointment | 能否统计社会公众预约 | TINYINT(1) | 默认0 |

### 预约记录（Appointment）

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 意义 | 类型 | 值域/约束 |
| id | 预约主键ID | BIGINT | 自增，主键 |
| visitor\_name | 预约人姓名 | VARCHAR(50) | 非空 |
| visitor\_id\_card | 预约人身份证号 | VARCHAR(255) | 非空，加密 |
| visitor\_phone | 预约人手机号 | VARCHAR(255) | 非空，加密 |
| visitor\_unit | 所在单位 | VARCHAR(200) | 可空 |
| campus | 预约校区 | VARCHAR(50) | 非空 |
| entry\_time | 预约进校时间 | TIMESTAMP | 非空 |
| transport\_mode | 交通方式 | VARCHAR(50) | 可空 |
| license\_plate | 车牌号 | VARCHAR(20) | 可空 |
| appointment\_type | 预约类型 | ENUM | 'PUBLIC'/'OFFICIAL' |
| created\_at | 申请时间 | TIMESTAMP | 默认当前时间 |
| status | 审核状态 | ENUM | 'PENDING'/'APPROVED'/'REJECTED' |
| official\_dept\_id | 公务访问部门ID | INT | 外键，可空 |
| official\_dept\_no | 公务访问部门编号 | VARCHAR(50) | 可空 |
| official\_dept\_name | 公务访问部门名称 | VARCHAR(100) | 可空 |
| official\_contact\_person | 公务访问接待人 | VARCHAR(50) | 可空 |
| official\_reason | 来访事由 | TEXT | 可空 |
| audited\_by | 审核人ID | INT | 外键，可空 |
| audited\_at | 审核时间 | TIMESTAMP | 可空 |

### 随行人员（accompanying\_person）

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 意义 | 类型 | 值域/约束 |
| id | 主键ID | BIGINT | 自增，主键 |
| appointment\_id | 关联的预约ID | BIGINT | 外键，非空 |
| full\_name | 姓名 | VARCHAR(50) | 非空 |
| id\_card | 身份证号（加密） | VARCHAR(255) | 非空 |
| phone | 手机号（加密） | VARCHAR(255) | 非空 |

### 审计日志表（Audit\_Log）

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 意义 | 类型 | 值域/约束 |
| id | 日志主键ID | BIGINT | 自增，主键 |
| admin\_id | 操作管理员ID | INT | 外键，非空 |
| admin\_name | 操作管理员姓名 | VARCHAR(50) | 可空 |
| action | 操作类型 | VARCHAR(255) | 非空 |
| details | 操作详情 | TEXT | 可空 |
| ip\_address | 操作者IP地址 | VARCHAR(50) | 可空 |
| created\_at | 操作时间 | TIMESTAMP | 默认当前时间 |
| hmac\_signature | 日志完整性校验值 | VARCHAR(255) | 可空 |

## 数据库表概念关系模型

### ER图

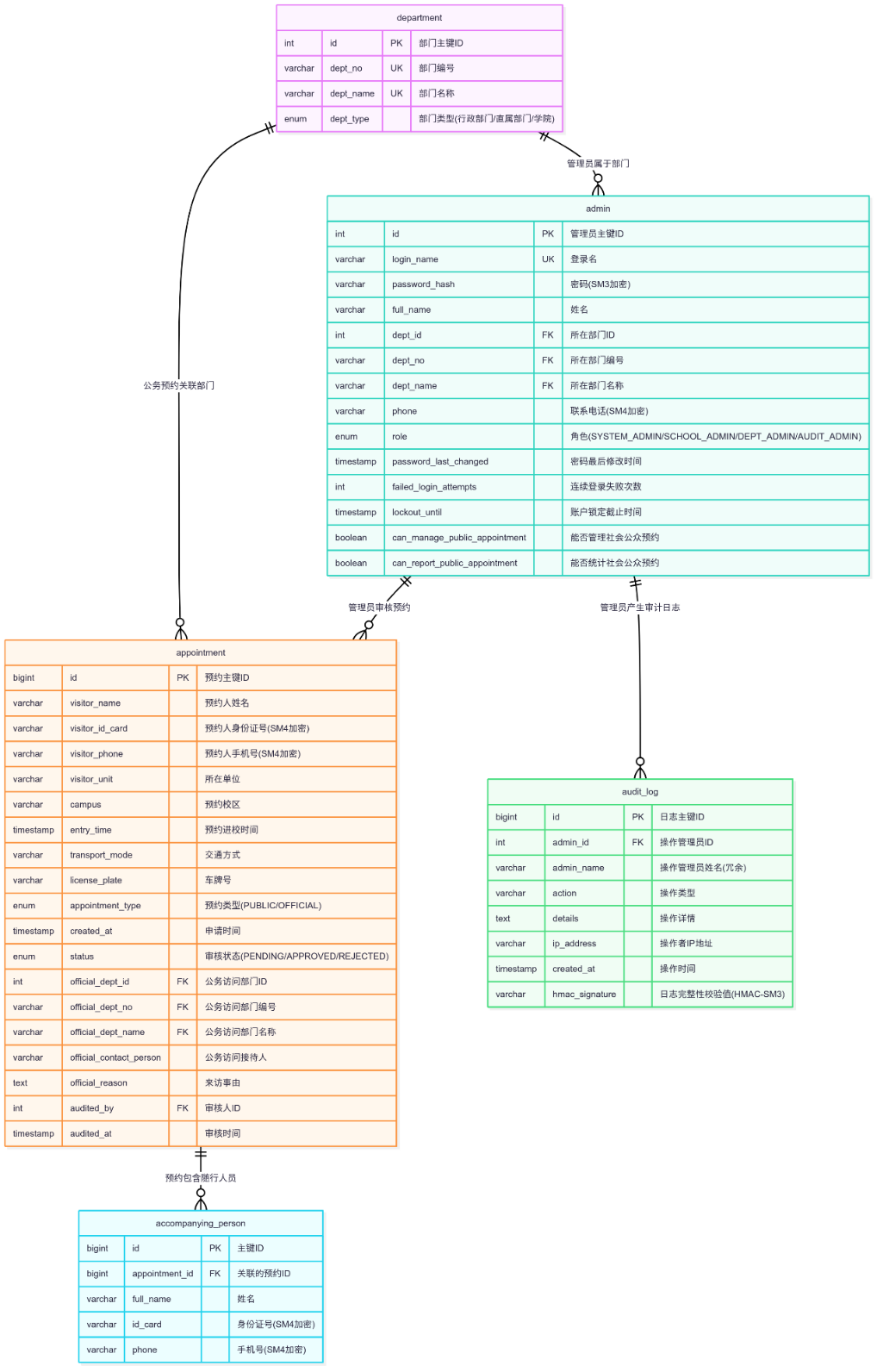


图 28 ER图

# 系统实现功能

## 用户端功能模块

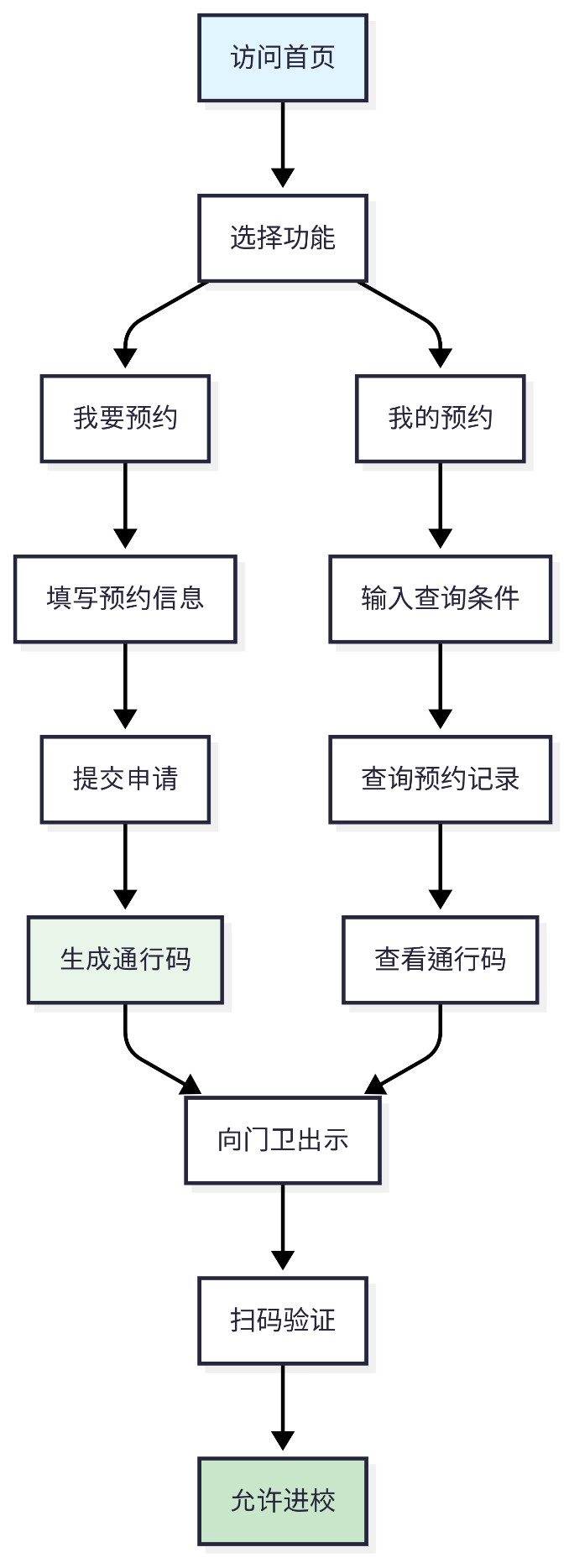


图 29 用户端功能模块

### 首页功能模块

* 校园通行码系统标题和简介
* 预约通道二维码展示（支持扫码进入预约系统）
* 主要功能入口按钮

### 预约申请功能模块

* 基本信息填写
* 预约信息填写
* 预约类型选择
* 公务预约特殊字段
* 随行人员管理

### 通行码功能模块

* 通行码生成
* 通行码状态显示
* 通行码信息展示
* 通行码有效期管理

### 二维码功能模块

* 二维码生成
* 二维码内容编码表单验证功能

## 管理端功能模块

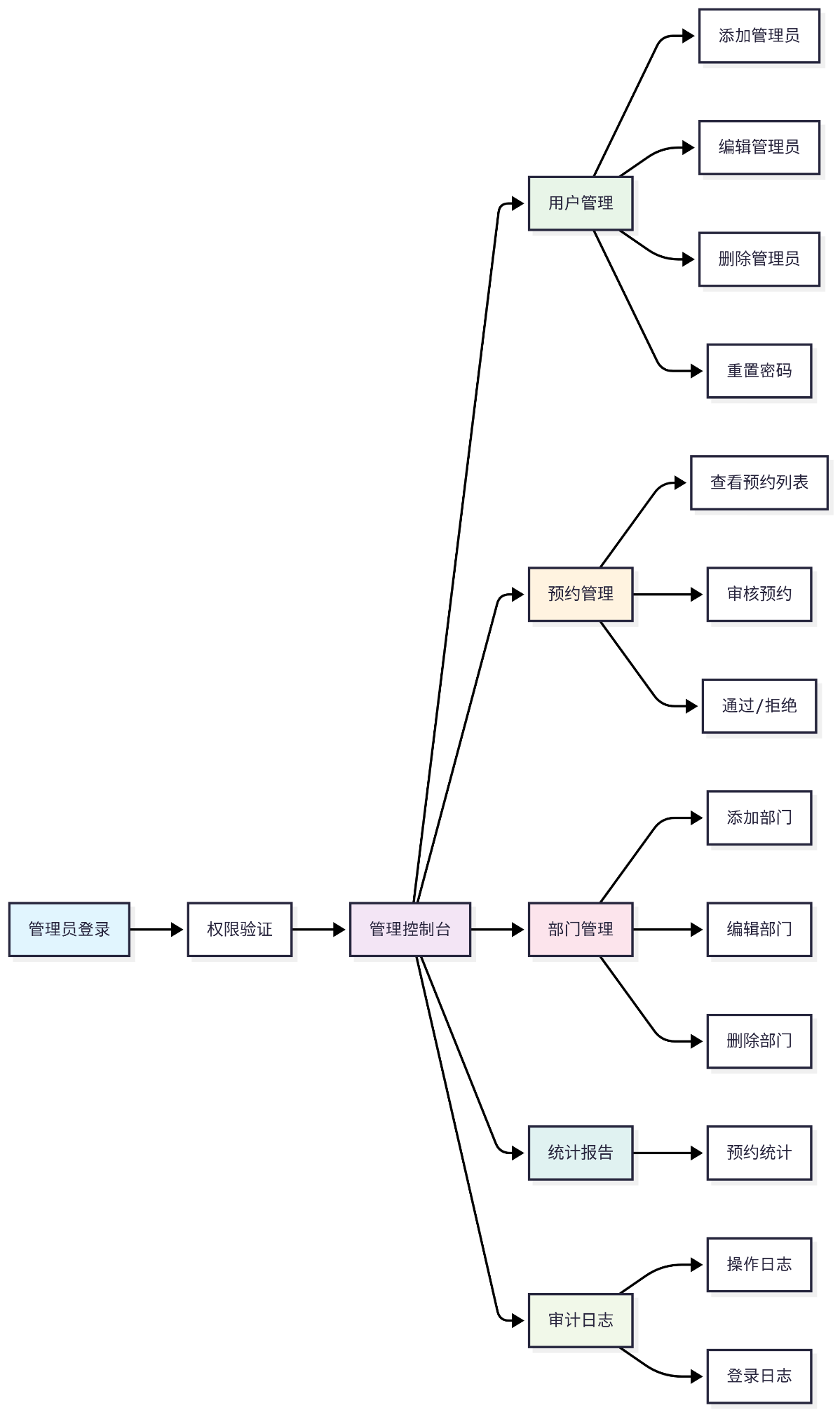


图 30 管理端功能模块

### 管理员登录功能模块

* 登陆验证
* 会话管理

### 管理员控制台功能模块

* 系统概览
* 功能导航

### 预约管理功能模块

* 预约查询
* 预约审核
* 预约详情查看

### 部门管理功能模块

* 部门信息管理
* 部门类型管理

### 管理员管理功能模块

* 管理员账户管理
* 权限管理
* 密码管理

### 统计报告功能模块

* 预约统计

### 操作审计功能模块

* 审计日志查询
* 日志内容

## 系统安全功能模块

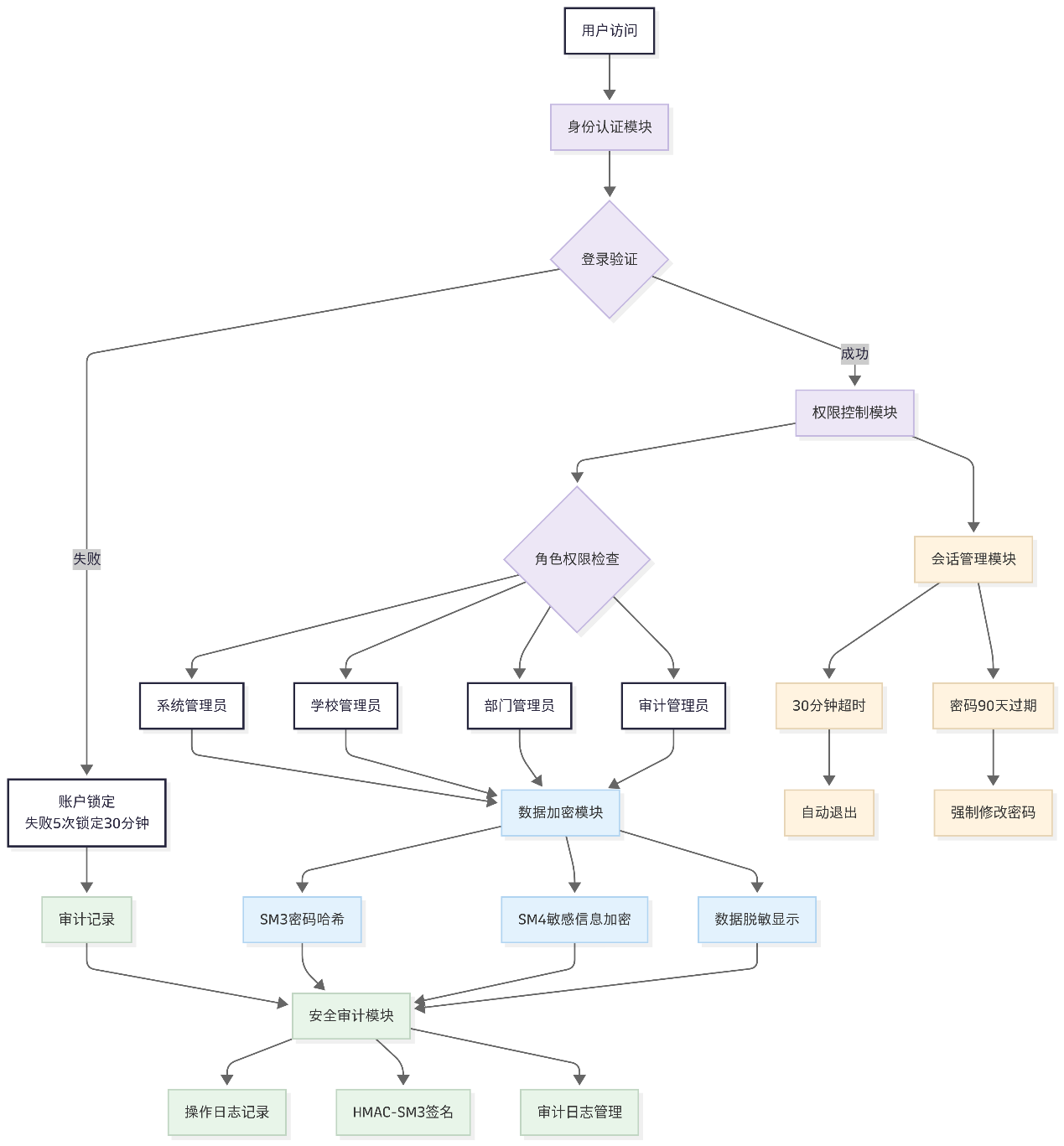


图 31 系统安全功能模块

### 国密算法加密模块

* SM3哈希算法
* SM4对称加密算法
* HMAC-SM3完整性保护

### 密码安全策略模块

* 密码复杂度要求
* 密码过期管理
* 密码存储安全

### 账户安全控制模块

* 登录失败锁定
* 会话安全管理
* IP地址记录

### 安全审计日志模块

* 操作日志记录
* 日志完整性保护
* 日志管理功能

### 访问控制权限模块

* 基于角色的访问控制
* 功能权限细分
* 数据权限控制

# 程序清单

#### src/main/java/com/zjut/passcode/bean/AccompanyingPerson.java

package com.zjut.passcode.bean;  
  
public class AccompanyingPerson {  
 private long id;  
 private long appointmentId;  
 private String fullName;  
 private String idCard;  
 private String phone;  
   
 public AccompanyingPerson() {}  
   
 public AccompanyingPerson(long appointmentId, String fullName, String idCard, String phone) {  
 this.appointmentId = appointmentId;  
 this.fullName = fullName;  
 this.idCard = idCard;  
 this.phone = phone;  
 }  
   
 public long getId() {  
 return id;  
 }  
   
 public void setId(long id) {  
 this.id = id;  
 }  
   
 public long getAppointmentId() {  
 return appointmentId;  
 }  
   
 public void setAppointmentId(long appointmentId) {  
 this.appointmentId = appointmentId;  
 }  
   
 public String getFullName() {  
 return fullName;  
 }  
   
 public void setFullName(String fullName) {  
 this.fullName = fullName;  
 }  
   
 public String getIdCard() {  
 return idCard;  
 }  
   
 public void setIdCard(String idCard) {  
 this.idCard = idCard;  
 }  
   
 public String getPhone() {  
 return phone;  
 }  
   
 public void setPhone(String phone) {  
 this.phone = phone;  
 }  
}

#### src/main/java/com/zjut/passcode/bean/Admin.java

package com.zjut.passcode.bean;  
  
import java.sql.Timestamp;  
  
public class Admin {  
 private int id;  
 private String loginName;  
 private String passwordHash;  
 private String fullName;  
 private int deptId;  
 private String phone;  
 private String role;  
 private Timestamp passwordLastChanged;  
 private int failedLoginAttempts;  
 private Timestamp lockoutUntil;  
 private String deptName; // 用于显示  
 private boolean canManagePublicAppointment;  
 private boolean canReportPublicAppointment;  
   
 public Admin() {}  
   
 public Admin(String loginName, String passwordHash, String fullName, int deptId, String phone, String role) {  
 this.loginName = loginName;  
 this.passwordHash = passwordHash;  
 this.fullName = fullName;  
 this.deptId = deptId;  
 this.phone = phone;  
 this.role = role;  
 }  
   
 public int getId() {  
 return id;  
 }  
   
 public void setId(int id) {  
 this.id = id;  
 }  
   
 public String getLoginName() {  
 return loginName;  
 }  
   
 public void setLoginName(String loginName) {  
 this.loginName = loginName;  
 }  
   
 public String getPasswordHash() {  
 return passwordHash;  
 }  
   
 public void setPasswordHash(String passwordHash) {  
 this.passwordHash = passwordHash;  
 }  
   
 public String getFullName() {  
 return fullName;  
 }  
   
 public void setFullName(String fullName) {  
 this.fullName = fullName;  
 }  
   
 public int getDeptId() {  
 return deptId;  
 }  
   
 public void setDeptId(int deptId) {  
 this.deptId = deptId;  
 }  
   
 public String getPhone() {  
 return phone;  
 }  
   
 public void setPhone(String phone) {  
 this.phone = phone;  
 }  
   
 public String getRole() {  
 return role;  
 }  
   
 public void setRole(String role) {  
 this.role = role;  
 }  
   
 public Timestamp getPasswordLastChanged() {  
 return passwordLastChanged;  
 }  
   
 public void setPasswordLastChanged(Timestamp passwordLastChanged) {  
 this.passwordLastChanged = passwordLastChanged;  
 }  
   
 public int getFailedLoginAttempts() {  
 return failedLoginAttempts;  
 }  
   
 public void setFailedLoginAttempts(int failedLoginAttempts) {  
 this.failedLoginAttempts = failedLoginAttempts;  
 }  
   
 public Timestamp getLockoutUntil() {  
 return lockoutUntil;  
 }  
   
 public void setLockoutUntil(Timestamp lockoutUntil) {  
 this.lockoutUntil = lockoutUntil;  
 }  
   
 public String getDeptName() {  
 return deptName;  
 }  
   
 public void setDeptName(String deptName) {  
 this.deptName = deptName;  
 }  
   
 public boolean isCanManagePublicAppointment() {  
 return canManagePublicAppointment;  
 }  
   
 public void setCanManagePublicAppointment(boolean canManagePublicAppointment) {  
 this.canManagePublicAppointment = canManagePublicAppointment;  
 }  
   
 public boolean isCanReportPublicAppointment() {  
 return canReportPublicAppointment;  
 }  
   
 public void setCanReportPublicAppointment(boolean canReportPublicAppointment) {  
 this.canReportPublicAppointment = canReportPublicAppointment;  
 }  
}

#### src/main/java/com/zjut/passcode/bean/Appointment.java

package com.zjut.passcode.bean;  
  
import java.sql.Timestamp;  
import java.util.List;  
  
*/\*\*  
 \* 预约实体类，封装预约相关信息。  
 \*/*public class Appointment {  
 */\*\* 预约ID \*/* private long id;  
 */\*\* 访客姓名 \*/* private String visitorName;  
 */\*\* 访客身份证号 \*/* private String visitorIdCard;  
 */\*\* 访客手机号 \*/* private String visitorPhone;  
 */\*\* 访客单位 \*/* private String visitorUnit;  
 */\*\* 校区 \*/* private String campus;  
 */\*\* 进校时间 \*/* private Timestamp entryTime;  
 */\*\* 交通方式 \*/* private String transportMode;  
 */\*\* 车牌号 \*/* private String licensePlate;  
 */\*\* 预约类型 \*/* private String appointmentType;  
 */\*\* 创建时间 \*/* private Timestamp createdAt;  
 */\*\* 预约状态 \*/* private String status;  
 */\*\* 公务访问部门ID \*/* private Integer officialDeptId;  
 */\*\* 公务联系人 \*/* private String officialContactPerson;  
 */\*\* 公务事由 \*/* private String officialReason;  
 */\*\* 审核人ID \*/* private Integer auditedBy;  
 */\*\* 审核时间 \*/* private Timestamp auditedAt;  
 */\*\* 公务访问部门名称 \*/* private String officialDeptName;  
 */\*\* 审核人姓名 \*/* private String auditedByName;  
 */\*\* 公务访问部门编号 \*/* private String officialDeptNo;  
 */\*\* 随行人员列表 \*/* private List<AccompanyingPerson> accompanyingPersons;  
   
 */\*\*  
 \* 无参构造方法。  
 \*/* public Appointment() {}  
   
 */\*\*  
 \* 带参数构造方法。  
 \* @param visitorName 访客姓名  
 \* @param visitorIdCard 访客身份证号  
 \* @param visitorPhone 访客手机号  
 \* @param visitorUnit 访客单位  
 \* @param campus 校区  
 \* @param entryTime 进校时间  
 \* @param transportMode 交通方式  
 \* @param licensePlate 车牌号  
 \* @param appointmentType 预约类型  
 \*/* public Appointment(String visitorName, String visitorIdCard, String visitorPhone,   
 String visitorUnit, String campus, Timestamp entryTime,   
 String transportMode, String licensePlate, String appointmentType) {  
 this.visitorName = visitorName;  
 this.visitorIdCard = visitorIdCard;  
 this.visitorPhone = visitorPhone;  
 this.visitorUnit = visitorUnit;  
 this.campus = campus;  
 this.entryTime = entryTime;  
 this.transportMode = transportMode;  
 this.licensePlate = licensePlate;  
 this.appointmentType = appointmentType;  
 this.status = "PENDING";  
 }  
   
 public long getId() {  
 return id;  
 }  
   
 public void setId(long id) {  
 this.id = id;  
 }  
   
 public String getVisitorName() {  
 return visitorName;  
 }  
   
 public void setVisitorName(String visitorName) {  
 this.visitorName = visitorName;  
 }  
   
 public String getVisitorIdCard() {  
 return visitorIdCard;  
 }  
   
 public void setVisitorIdCard(String visitorIdCard) {  
 this.visitorIdCard = visitorIdCard;  
 }  
   
 public String getVisitorPhone() {  
 return visitorPhone;  
 }  
   
 public void setVisitorPhone(String visitorPhone) {  
 this.visitorPhone = visitorPhone;  
 }  
   
 public String getVisitorUnit() {  
 return visitorUnit;  
 }  
   
 public void setVisitorUnit(String visitorUnit) {  
 this.visitorUnit = visitorUnit;  
 }  
   
 public String getCampus() {  
 return campus;  
 }  
   
 public void setCampus(String campus) {  
 this.campus = campus;  
 }  
   
 public Timestamp getEntryTime() {  
 return entryTime;  
 }  
   
 public void setEntryTime(Timestamp entryTime) {  
 this.entryTime = entryTime;  
 }  
   
 public String getTransportMode() {  
 return transportMode;  
 }  
   
 public void setTransportMode(String transportMode) {  
 this.transportMode = transportMode;  
 }  
   
 public String getLicensePlate() {  
 return licensePlate;  
 }  
   
 public void setLicensePlate(String licensePlate) {  
 this.licensePlate = licensePlate;  
 }  
   
 public String getAppointmentType() {  
 return appointmentType;  
 }  
   
 public void setAppointmentType(String appointmentType) {  
 this.appointmentType = appointmentType;  
 }  
   
 public Timestamp getCreatedAt() {  
 return createdAt;  
 }  
   
 public void setCreatedAt(Timestamp createdAt) {  
 this.createdAt = createdAt;  
 }  
   
 public String getStatus() {  
 return status;  
 }  
   
 public void setStatus(String status) {  
 this.status = status;  
 }  
   
 public Integer getOfficialDeptId() {  
 return officialDeptId;  
 }  
   
 public void setOfficialDeptId(Integer officialDeptId) {  
 this.officialDeptId = officialDeptId;  
 }  
   
 public String getOfficialContactPerson() {  
 return officialContactPerson;  
 }  
   
 public void setOfficialContactPerson(String officialContactPerson) {  
 this.officialContactPerson = officialContactPerson;  
 }  
   
 public String getOfficialReason() {  
 return officialReason;  
 }  
   
 public void setOfficialReason(String officialReason) {  
 this.officialReason = officialReason;  
 }  
   
 public Integer getAuditedBy() {  
 return auditedBy;  
 }  
   
 public void setAuditedBy(Integer auditedBy) {  
 this.auditedBy = auditedBy;  
 }  
   
 public Timestamp getAuditedAt() {  
 return auditedAt;  
 }  
   
 public void setAuditedAt(Timestamp auditedAt) {  
 this.auditedAt = auditedAt;  
 }  
   
 public String getOfficialDeptName() {  
 return officialDeptName;  
 }  
   
 public void setOfficialDeptName(String officialDeptName) {  
 this.officialDeptName = officialDeptName;  
 }  
   
 public String getAuditedByName() {  
 return auditedByName;  
 }  
   
 public void setAuditedByName(String auditedByName) {  
 this.auditedByName = auditedByName;  
 }  
   
 public String getOfficialDeptNo() {  
 return officialDeptNo;  
 }  
   
 public void setOfficialDeptNo(String officialDeptNo) {  
 this.officialDeptNo = officialDeptNo;  
 }  
   
 public List<AccompanyingPerson> getAccompanyingPersons() {  
 return accompanyingPersons;  
 }  
   
 public void setAccompanyingPersons(List<AccompanyingPerson> accompanyingPersons) {  
 this.accompanyingPersons = accompanyingPersons;  
 }  
}

#### com.zjut.passcode.bean.AuditLog

package com.zjut.passcode.bean;  
  
import java.sql.Timestamp;  
  
public class AuditLog {  
 private long id;  
 private int adminId;  
 private String adminName;  
 private String action;  
 private String details;  
 private String ipAddress;  
 private Timestamp createdAt;  
 private String hmacSignature;  
   
 public AuditLog() {}  
   
 public AuditLog(int adminId, String adminName, String action, String details, String ipAddress) {  
 this.adminId = adminId;  
 this.adminName = adminName;  
 this.action = action;  
 this.details = details;  
 this.ipAddress = ipAddress;  
 }  
   
 public long getId() {  
 return id;  
 }  
   
 public void setId(long id) {  
 this.id = id;  
 }  
   
 public int getAdminId() {  
 return adminId;  
 }  
   
 public void setAdminId(int adminId) {  
 this.adminId = adminId;  
 }  
   
 public String getAdminName() {  
 return adminName;  
 }  
   
 public void setAdminName(String adminName) {  
 this.adminName = adminName;  
 }  
   
 public String getAction() {  
 return action;  
 }  
   
 public void setAction(String action) {  
 this.action = action;  
 }  
   
 public String getDetails() {  
 return details;  
 }  
   
 public void setDetails(String details) {  
 this.details = details;  
 }  
   
 public String getIpAddress() {  
 return ipAddress;  
 }  
   
 public void setIpAddress(String ipAddress) {  
 this.ipAddress = ipAddress;  
 }  
   
 public Timestamp getCreatedAt() {  
 return createdAt;  
 }  
   
 public void setCreatedAt(Timestamp createdAt) {  
 this.createdAt = createdAt;  
 }  
   
 public String getHmacSignature() {  
 return hmacSignature;  
 }  
   
 public void setHmacSignature(String hmacSignature) {  
 this.hmacSignature = hmacSignature;  
 }  
}

#### src/main/java/com/zjut/passcode/bean/Department.java

package com.zjut.passcode.bean;  
  
import java.sql.Timestamp;  
  
public class AuditLog {  
 private long id;  
 private int adminId;  
 private String adminName;  
 private String action;  
 private String details;  
 private String ipAddress;  
 private Timestamp createdAt;  
 private String hmacSignature;  
   
 public AuditLog() {}  
   
 public AuditLog(int adminId, String adminName, String action, String details, String ipAddress) {  
 this.adminId = adminId;  
 this.adminName = adminName;  
 this.action = action;  
 this.details = details;  
 this.ipAddress = ipAddress;  
 }  
   
 public long getId() {  
 return id;  
 }  
   
 public void setId(long id) {  
 this.id = id;  
 }  
   
 public int getAdminId() {  
 return adminId;  
 }  
   
 public void setAdminId(int adminId) {  
 this.adminId = adminId;  
 }  
   
 public String getAdminName() {  
 return adminName;  
 }  
   
 public void setAdminName(String adminName) {  
 this.adminName = adminName;  
 }  
   
 public String getAction() {  
 return action;  
 }  
   
 public void setAction(String action) {  
 this.action = action;  
 }  
   
 public String getDetails() {  
 return details;  
 }  
   
 public void setDetails(String details) {  
 this.details = details;  
 }  
   
 public String getIpAddress() {  
 return ipAddress;  
 }  
   
 public void setIpAddress(String ipAddress) {  
 this.ipAddress = ipAddress;  
 }  
   
 public Timestamp getCreatedAt() {  
 return createdAt;  
 }  
   
 public void setCreatedAt(Timestamp createdAt) {  
 this.createdAt = createdAt;  
 }  
   
 public String getHmacSignature() {  
 return hmacSignature;  
 }  
   
 public void setHmacSignature(String hmacSignature) {  
 this.hmacSignature = hmacSignature;  
 }  
}

#### src/main/java/com/zjut/passcode/controller/AdminAppointmentsServlet.java

package com.zjut.passcode.controller;  
  
import java.io.IOException;  
import java.util.List;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import javax.servlet.http.HttpSession;  
  
import com.zjut.passcode.bean.Admin;  
import com.zjut.passcode.bean.Appointment;  
import com.zjut.passcode.dao.AppointmentDao;  
  
@WebServlet("/admin/appointments")  
public class AdminAppointmentsServlet extends HttpServlet {  
 private AppointmentDao appointmentDao = new AppointmentDao();  
   
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
   
 // Check if admin is logged in  
 HttpSession session = request.getSession(false);  
 if (session == null || session.getAttribute("admin") == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
   
 Admin admin = (Admin) session.getAttribute("admin");  
 System.*out*.println("INFO: Admin appointments page accessed by: " + admin.getLoginName());  
   
 try {  
 List<Appointment> appointments;  
   
 // 根据管理员角色获取相应的预约  
 if ("DEPT\_ADMIN".equals(admin.getRole())) {  
 // 部门管理员根据权限获取预约  
 if (admin.isCanManagePublicAppointment()) {  
 // 可以管理社会公众预约，获取本部门所有预约（公务预约 + 社会公众预约）  
 appointments = appointmentDao.getAppointmentsByDept(admin.getDeptId());  
 System.out.println("INFO: Dept admin with public permission retrieved " + appointments.size() + " appointments for dept " + admin.getDeptId());  
 } else {  
 // 只能管理公务预约，获取本部门公务预约  
 appointments = appointmentDao.getAppointmentsByDeptAndType(admin.getDeptId(), "OFFICIAL");  
 System.out.println("INFO: Dept admin without public permission retrieved " + appointments.size() + " official appointments for dept " + admin.getDeptId());  
 }  
 } else if ("SCHOOL\_ADMIN".equals(admin.getRole()) || "SYSTEM\_ADMIN".equals(admin.getRole())) {  
 // 学校管理员和系统管理员可以查看所有预约  
 appointments = appointmentDao.getAllAppointments();  
 System.out.println("INFO: School/System admin retrieved " + appointments.size() + " appointments");  
 } else {  
 // 其他角色无权访问  
 response.sendError(HttpServletResponse.SC\_FORBIDDEN, "无权访问预约管理");  
 return;  
 }  
   
 // Set attributes for JSP  
 request.setAttribute("appointments", appointments);  
 request.setAttribute("admin", admin);  
   
 // Forward to appointments management page  
 request.getRequestDispatcher("/admin/appointments.jsp").forward(request, response);  
   
 } catch (Exception e) {  
 System.out.println("ERROR: Failed to retrieve appointments");  
 System.out.println("ERROR Details: " + e.getMessage());  
 e.printStackTrace();  
   
 request.setAttribute("error", "获取预约信息失败");  
 request.setAttribute("errorDetails", "错误详情: " + e.getMessage());  
 request.getRequestDispatcher("/error.jsp").forward(request, response);  
 }  
 }  
   
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 // Check if admin is logged in  
 HttpSession session = request.getSession(false);  
 if (session == null || session.getAttribute("admin") == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
 Admin admin = (Admin) session.getAttribute("admin");  
  
 String action = request.getParameter("action");  
 String appointmentIdStr = request.getParameter("appointmentId");  
 if (action != null && appointmentIdStr != null) {  
 try {  
 long appointmentId = Long.parseLong(appointmentIdStr);  
   
 // 获取预约信息进行权限检查  
 Appointment appointment = appointmentDao.getAppointmentById(appointmentId);  
 if (appointment == null) {  
 request.setAttribute("error", "预约不存在");  
 doGet(request, response);  
 return;  
 }  
   
 // 权限检查：部门管理员根据权限审核预约  
 if ("DEPT\_ADMIN".equals(admin.getRole())) {  
 if ("OFFICIAL".equals(appointment.getAppointmentType())) {  
 // 公务预约：检查是否是本部门的预约  
 if (appointment.getOfficialDeptId() == null ||   
 appointment.getOfficialDeptId() != admin.getDeptId()) {  
 request.setAttribute("error", "您只能审核本部门的公务预约");  
 doGet(request, response);  
 return;  
 }  
 } else if ("PUBLIC".equals(appointment.getAppointmentType())) {  
 // 社会公众预约：检查是否有管理权限  
 if (!admin.isCanManagePublicAppointment()) {  
 request.setAttribute("error", "您没有权限审核社会公众预约");  
 doGet(request, response);  
 return;  
 }  
 }  
 }  
   
 boolean result = false;  
 if ("approve".equals(action)) {  
 result = appointmentDao.updateAppointmentStatus(appointmentId, "APPROVED", admin.getId());  
 } else if ("reject".equals(action)) {  
 result = appointmentDao.updateAppointmentStatus(appointmentId, "REJECTED", admin.getId());  
 }  
 if (result) {  
 // 记录审计日志  
 // 这里假设有AuditLogDao和AuditLog类  
 com.zjut.passcode.dao.AuditLogDao auditLogDao = new com.zjut.passcode.dao.AuditLogDao();  
 com.zjut.passcode.bean.AuditLog log = new com.zjut.passcode.bean.AuditLog(  
 admin.getId(), admin.getFullName(),  
 ("approve".equals(action) ? "通过预约" : "拒绝预约"),  
 ("approve".equals(action) ? "通过预约ID: " : "拒绝预约ID: ") + appointmentId,  
 request.getRemoteAddr()  
 );  
 log.setHmacSignature(com.zjut.passcode.util.CryptoUtil.generateHmacSm3(log.toString()));  
 auditLogDao.addAuditLog(log);  
 request.setAttribute("message", ("approve".equals(action) ? "预约已通过" : "预约已拒绝"));  
 } else {  
 request.setAttribute("error", "操作失败，请重试");  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 request.setAttribute("error", "系统错误，请重试");  
 }  
 }  
 doGet(request, response);  
 }  
}

#### src/main/java/com/zjut/passcode/controller/AdminAuditLogsServlet.java

package com.zjut.passcode.controller;  
  
import java.io.IOException;  
import java.util.List;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import javax.servlet.http.HttpSession;  
  
import com.zjut.passcode.bean.Admin;  
import com.zjut.passcode.bean.AuditLog;  
import com.zjut.passcode.dao.AuditLogDao;  
  
@WebServlet("/admin/audit\_logs")  
public class AdminAuditLogsServlet extends HttpServlet {  
 private AuditLogDao auditLogDao = new AuditLogDao();  
   
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
   
 // Check if admin is logged in  
 HttpSession session = request.getSession(false);  
 if (session == null || session.getAttribute("admin") == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
   
 Admin admin = (Admin) session.getAttribute("admin");  
 System.*out*.println("INFO: Admin audit logs page accessed by: " + admin.getLoginName());  
   
 try {  
 // 分页参数  
 int page = 1;  
 int pageSize = 20;  
 String pageParam = request.getParameter("page");  
 if (pageParam != null) {  
 try { page = Integer.*parseInt*(pageParam); } catch (NumberFormatException ignored) {}  
 }  
 String pageSizeParam = request.getParameter("pageSize");  
 if (pageSizeParam != null) {  
 try { pageSize = Integer.*parseInt*(pageSizeParam); } catch (NumberFormatException ignored) {}  
 }  
 int totalLogs = auditLogDao.getAuditLogsCount();  
 int totalPages = (int)Math.*ceil*((double)totalLogs / pageSize);  
 if (page < 1) page = 1;  
 if (page > totalPages && totalPages > 0) page = totalPages;  
 // 分页查询  
 List<AuditLog> auditLogs = auditLogDao.getAuditLogsByPage(page, pageSize);  
 // Set attributes for JSP  
 request.setAttribute("auditLogs", auditLogs);  
 request.setAttribute("admin", admin);  
 request.setAttribute("page", page);  
 request.setAttribute("pageSize", pageSize);  
 request.setAttribute("totalLogs", totalLogs);  
 request.setAttribute("totalPages", totalPages);  
 // Forward to audit logs page  
 request.getRequestDispatcher("/admin/audit\_logs.jsp").forward(request, response);  
   
 } catch (Exception e) {  
 System.*out*.println("ERROR: Failed to retrieve audit logs");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 e.printStackTrace();  
   
 request.setAttribute("error", "获取审计日志失败");  
 request.setAttribute("errorDetails", "错误详情: " + e.getMessage());  
 request.getRequestDispatcher("/error.jsp").forward(request, response);  
 }  
 }  
   
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 doGet(request, response);  
 }  
}

#### src/main/java/com/zjut/passcode/controller/AdminDepartmentsServlet.java

package com.zjut.passcode.controller;  
  
import java.io.IOException;  
import java.util.List;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import javax.servlet.http.HttpSession;  
  
import com.zjut.passcode.bean.Admin;  
import com.zjut.passcode.bean.Department;  
import com.zjut.passcode.dao.DepartmentDao;  
  
@WebServlet("/admin/departments")  
public class AdminDepartmentsServlet extends HttpServlet {  
 private DepartmentDao departmentDao = new DepartmentDao();  
   
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
   
 // Set character encoding to handle Chinese characters  
 request.setCharacterEncoding("UTF-8");  
 response.setCharacterEncoding("UTF-8");  
   
 // Check if admin is logged in  
 HttpSession session = request.getSession(false);  
 if (session == null || session.getAttribute("admin") == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
   
 Admin admin = (Admin) session.getAttribute("admin");  
 System.*out*.println("INFO: Admin departments page accessed by: " + admin.getLoginName());  
   
 try {  
 // Get all departments  
 List<Department> departments = departmentDao.getAllDepartments();  
 System.*out*.println("INFO: Retrieved " + departments.size() + " departments");  
   
 // Set attributes for JSP  
 request.setAttribute("departments", departments);  
 request.setAttribute("admin", admin);  
   
 // Check for messages from other operations  
 String successMsg = request.getParameter("success");  
 String errorMsg = request.getParameter("error");  
   
 if (successMsg != null && !successMsg.trim().isEmpty()) {  
 request.setAttribute("success", successMsg);  
 }  
   
 if (errorMsg != null && !errorMsg.trim().isEmpty()) {  
 request.setAttribute("error", errorMsg);  
 }  
   
 // Forward to departments management page  
 request.getRequestDispatcher("/admin/departments.jsp").forward(request, response);  
   
 } catch (Exception e) {  
 System.*out*.println("ERROR: Failed to retrieve departments");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 e.printStackTrace();  
   
 request.setAttribute("error", "获取部门信息失败");  
 request.setAttribute("errorDetails", "错误详情: " + e.getMessage());  
 request.getRequestDispatcher("/error.jsp").forward(request, response);  
 }  
 }  
   
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 doGet(request, response);  
 }  
}

#### src/main/java/com/zjut/passcode/controller/AdminLoginServlet.java

package com.zjut.passcode.controller;  
  
import java.io.IOException;  
import java.sql.Timestamp;  
import java.time.LocalDateTime;  
import java.time.temporal.ChronoUnit;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import javax.servlet.http.HttpSession;  
  
import com.zjut.passcode.bean.Admin;  
import com.zjut.passcode.bean.AuditLog;  
import com.zjut.passcode.dao.AdminDao;  
import com.zjut.passcode.dao.AuditLogDao;  
import com.zjut.passcode.util.CryptoUtil;  
  
*/\*\*  
 \* 管理员登录Servlet，处理管理员登录相关请求。  
 \*/*@WebServlet("/admin/login")  
public class AdminLoginServlet extends HttpServlet {  
 */\*\* 管理员数据访问对象 \*/* private AdminDao adminDao = new AdminDao();  
 */\*\* 审计日志数据访问对象 \*/* private AuditLogDao auditLogDao = new AuditLogDao();  
   
 */\*\*  
 \* 处理GET请求，跳转到登录页面。  
 \*/* @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 request.getRequestDispatcher("/admin\_login.jsp").forward(request, response);  
 }  
   
 */\*\*  
 \* 处理POST请求，执行登录逻辑。  
 \*/* @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 request.setCharacterEncoding("UTF-8");  
 response.setContentType("text/html;charset=UTF-8");  
   
 String loginName = request.getParameter("loginName");  
 String password = request.getParameter("password");  
 String ipAddress = getClientIpAddress(request);  
   
 System.*out*.println("=== Login Attempt ===");  
 System.*out*.println("Username: " + loginName);  
 System.*out*.println("IP Address: " + ipAddress);  
 System.*out*.println("Timestamp: " + LocalDateTime.*now*());  
   
 try {  
 // Validate input  
 if (loginName == null || loginName.trim().isEmpty() ||   
 password == null || password.trim().isEmpty()) {  
 System.*out*.println("ERROR: Input validation failed - empty username or password");  
 request.setAttribute("error", "用户名和密码不能为空");  
 request.setAttribute("errorDetails", "请检查输入的用户名和密码是否完整");  
 request.setAttribute("errorType", "INPUT\_VALIDATION");  
 request.getRequestDispatcher("/admin\_login.jsp").forward(request, response);  
 return;  
 }  
   
 // Get admin information  
 Admin admin = null;  
 try {  
 System.*out*.println("INFO: Attempting to retrieve admin from database...");  
 admin = adminDao.getAdminByLoginName(loginName);  
 if (admin != null) {  
 System.*out*.println("INFO: Admin found - ID: " + admin.getId() + ", Name: " + admin.getFullName());  
 } else {  
 System.*out*.println("WARNING: Admin not found for username: " + loginName);  
 }  
 } catch (Exception e) {  
 // Database connection error  
 System.*out*.println("ERROR: Database connection failed");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 e.printStackTrace();  
 logError("Database connection failed", e, loginName, ipAddress);  
 request.setAttribute("error", "系统暂时无法连接数据库，请稍后重试");  
 request.setAttribute("errorDetails", "错误详情: " + e.getMessage());  
 request.setAttribute("errorType", "DATABASE\_CONNECTION");  
 request.getRequestDispatcher("/error.jsp").forward(request, response);  
 return;  
 }  
   
 if (admin == null) {  
 System.*out*.println("ERROR: User not found - " + loginName);  
 logFailedLogin(loginName, ipAddress, "User not found");  
 request.setAttribute("error", "用户名或密码错误");  
 request.setAttribute("errorDetails", "用户名 '" + loginName + "' 在系统中不存在");  
 request.setAttribute("errorType", "USER\_NOT\_FOUND");  
 request.getRequestDispatcher("/admin\_login.jsp").forward(request, response);  
 return;  
 }  
   
 // Check if account is locked  
 if (admin.getLockoutUntil() != null &&   
 admin.getLockoutUntil().after(Timestamp.*valueOf*(LocalDateTime.*now*()))) {  
 System.*out*.println("ERROR: Account locked for user: " + loginName);  
 System.*out*.println("ERROR: Lockout until: " + admin.getLockoutUntil());  
 logFailedLogin(loginName, ipAddress, "Account locked");  
 request.setAttribute("error", "账户已被锁定，请稍后再试");  
 request.setAttribute("errorDetails", "账户锁定截止时间: " + admin.getLockoutUntil());  
 request.setAttribute("errorType", "ACCOUNT\_LOCKED");  
 request.getRequestDispatcher("/admin\_login.jsp").forward(request, response);  
 return;  
 }  
   
 // Verify password  
 String hashedPassword = null;  
 try {  
 System.*out*.println("INFO: Hashing password...");  
 hashedPassword = CryptoUtil.*sm3Hash*(password);  
 if (hashedPassword == null) {  
 throw new Exception("Password hashing failed");  
 }  
 System.*out*.println("INFO: Password hashed successfully");  
 } catch (Exception e) {  
 System.*out*.println("ERROR: Password encryption failed");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 e.printStackTrace();  
 logError("Password encryption failed", e, loginName, ipAddress);  
 request.setAttribute("error", "密码处理失败，请稍后重试");  
 request.setAttribute("errorDetails", "密码加密过程出现错误: " + e.getMessage());  
 request.setAttribute("errorType", "PASSWORD\_ENCRYPTION");  
 request.getRequestDispatcher("/error.jsp").forward(request, response);  
 return;  
 }  
   
 System.*out*.println("INFO: Input password hash: " + hashedPassword);  
 System.*out*.println("INFO: Database password hash: " + admin.getPasswordHash());  
 System.*out*.println("INFO: Password match: " + hashedPassword.equals(admin.getPasswordHash()));  
   
 if (!hashedPassword.equals(admin.getPasswordHash())) {  
 // Increase failed attempts  
 int failedAttempts = admin.getFailedLoginAttempts() + 1;  
 System.*out*.println("ERROR: Password mismatch for user: " + loginName);  
 System.*out*.println("ERROR: Failed attempts: " + failedAttempts);  
   
 try {  
 adminDao.updateFailedLoginAttempts(admin.getId(), failedAttempts);  
 System.*out*.println("INFO: Updated failed login attempts in database");  
 } catch (Exception e) {  
 System.*out*.println("ERROR: Failed to update login attempts");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 logError("Failed to update login attempts", e, loginName, ipAddress);  
 }  
   
 // Lock account if failed 5 times  
 if (failedAttempts >= 5) {  
 try {  
 Timestamp lockoutUntil = Timestamp.*valueOf*(LocalDateTime.*now*().plus(30, ChronoUnit.*MINUTES*));  
 adminDao.setLockoutUntil(admin.getId(), lockoutUntil);  
 System.*out*.println("WARNING: Account locked for 30 minutes");  
 System.*out*.println("WARNING: Lockout until: " + lockoutUntil);  
 logFailedLogin(loginName, ipAddress, "Password error, account locked");  
 request.setAttribute("error", "密码错误次数过多，账户已被锁定30分钟");  
 request.setAttribute("errorDetails", "连续失败次数: " + failedAttempts + "，锁定截止时间: " + lockoutUntil);  
 request.setAttribute("errorType", "ACCOUNT\_LOCKED");  
 } catch (Exception e) {  
 System.*out*.println("ERROR: Failed to set account lockout");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 logError("Failed to set account lockout", e, loginName, ipAddress);  
 request.setAttribute("error", "密码错误次数过多，但锁定账户时出现错误");  
 request.setAttribute("errorDetails", "错误详情: " + e.getMessage());  
 request.setAttribute("errorType", "LOCKOUT\_ERROR");  
 }  
 } else {  
 logFailedLogin(loginName, ipAddress, "Password error");  
 request.setAttribute("error", "用户名或密码错误");  
 request.setAttribute("errorDetails", "连续失败次数: " + failedAttempts + "，剩余尝试次数: " + (5 - failedAttempts));  
 request.setAttribute("errorType", "PASSWORD\_MISMATCH");  
 }  
 request.getRequestDispatcher("/admin\_login.jsp").forward(request, response);  
 return;  
 }  
   
 // Check if password is expired (90 days)  
 if (admin.getPasswordLastChanged() != null) {  
 long daysSinceChange = ChronoUnit.*DAYS*.between(  
 admin.getPasswordLastChanged().toLocalDateTime(),   
 LocalDateTime.*now*()  
 );  
 System.*out*.println("INFO: Days since password change: " + daysSinceChange);  
 if (daysSinceChange > 90) {  
 System.*out*.println("WARNING: Password expired for user: " + loginName);  
 request.setAttribute("warning", "密码已过期，请及时修改密码");  
 request.setAttribute("warningDetails", "密码已使用 " + daysSinceChange + " 天，建议修改密码");  
 }  
 }  
   
 // Login successful, reset failed attempts  
 try {  
 adminDao.resetFailedLoginAttempts(admin.getId());  
 System.*out*.println("INFO: Reset failed login attempts");  
 } catch (Exception e) {  
 System.*out*.println("ERROR: Failed to reset login attempts");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 logError("Failed to reset login attempts", e, loginName, ipAddress);  
 // Don't affect successful login, just log error  
 }  
  
 System.*out*.println("SUCCESS: Login successful for user: " + loginName);  
  
 // Record login log  
 try {  
 AuditLog log = new AuditLog(admin.getId(), admin.getFullName(), "登录",   
 "管理员登录成功", ipAddress);  
 log.setHmacSignature(CryptoUtil.*generateHmacSm3*(log.toString()));  
 auditLogDao.addAuditLog(log);  
 System.*out*.println("INFO: Login audit log recorded");  
 } catch (Exception e) {  
 System.*out*.println("ERROR: Failed to record login audit log");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 logError("Failed to record login audit log", e, loginName, ipAddress);  
 // Don't affect successful login, just log error  
 }  
   
 // Set session  
 HttpSession session = request.getSession();  
 session.setAttribute("admin", admin);  
 session.setAttribute("adminId", admin.getId());  
 session.setAttribute("adminName", admin.getFullName());  
 session.setAttribute("adminRole", admin.getRole());  
 session.setAttribute("adminDeptId", admin.getDeptId());  
 session.setAttribute("loginTime", LocalDateTime.*now*().format(java.time.format.DateTimeFormatter.*ofPattern*("yyyy-MM-dd HH:mm:ss")));  
 session.setMaxInactiveInterval(30 \* 60); // 30 minutes timeout  
 System.*out*.println("INFO: Session created for user: " + loginName);  
 System.*out*.println("INFO: Session timeout: 30 minutes");  
   
 // Redirect based on role  
 if ("AUDIT\_ADMIN".equals(admin.getRole())) {  
 System.*out*.println("INFO: Redirecting to audit admin page");  
 response.sendRedirect(request.getContextPath() + "/admin/audit\_logs");  
 } else {  
 System.*out*.println("INFO: Redirecting to admin index page");  
 response.sendRedirect(request.getContextPath() + "/admin/index.jsp");  
 }  
   
 } catch (Exception e) {  
 // Catch all unexpected exceptions  
 System.*out*.println("ERROR: Unexpected error during login process");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 e.printStackTrace();  
 logError("Unexpected error during login process", e, loginName, ipAddress);  
 request.setAttribute("error", "系统发生未知错误，请稍后重试");  
 request.setAttribute("errorDetails", "错误详情: " + e.getMessage());  
 request.setAttribute("errorType", "UNKNOWN\_ERROR");  
 request.getRequestDispatcher("/error.jsp").forward(request, response);  
 }  
 }  
   
 */\*\*  
 \* 获取客户端IP地址。  
 \* @param request HTTP请求对象  
 \* @return 客户端IP地址  
 \*/* private String getClientIpAddress(HttpServletRequest request) {  
 return request.getRemoteAddr();  
 }  
   
 */\*\*  
 \* 记录登录失败日志。  
 \* @param loginName 登录名  
 \* @param ipAddress 客户端IP  
 \* @param reason 失败原因  
 \*/* private void logFailedLogin(String loginName, String ipAddress, String reason) {  
 try {  
 AuditLog log = new AuditLog(0, loginName, "登录失败", reason, ipAddress);  
 log.setHmacSignature(CryptoUtil.*generateHmacSm3*(log.toString()));  
 auditLogDao.addAuditLog(log);  
 System.*out*.println("INFO: Failed login logged to audit trail");  
 } catch (Exception e) {  
 System.*out*.println("ERROR: Failed to log failed login attempt");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 e.printStackTrace();  
 }  
 }  
   
 */\*\*  
 \* 记录系统错误日志。  
 \* @param action 操作描述  
 \* @param e 异常对象  
 \* @param loginName 登录名  
 \* @param ipAddress 客户端IP  
 \*/* private void logError(String action, Exception e, String loginName, String ipAddress) {  
 try {  
 String details = action + " - Exception: " + e.getClass().getSimpleName() + ": " + e.getMessage();  
 AuditLog log = new AuditLog(0, loginName, "System Error", details, ipAddress);  
 log.setHmacSignature(CryptoUtil.*generateHmacSm3*(log.toString()));  
 auditLogDao.addAuditLog(log);  
 System.*out*.println("INFO: Error logged to audit trail");  
 } catch (Exception ex) {  
 System.*out*.println("ERROR: Failed to log error to audit trail");  
 System.*out*.println("ERROR Details: " + ex.getMessage());  
 ex.printStackTrace();  
 }  
 // Also output to console  
 System.*out*.println("ERROR: Login error - " + action + ": " + e.getMessage());  
 e.printStackTrace();  
 }  
}

#### src/main/java/com/zjut/passcode/controller/AdminLogoutServlet.java

package com.zjut.passcode.controller;  
  
import java.io.IOException;  
import java.time.LocalDateTime;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import javax.servlet.http.HttpSession;  
  
import com.zjut.passcode.bean.Admin;  
import com.zjut.passcode.bean.AuditLog;  
import com.zjut.passcode.dao.AuditLogDao;  
import com.zjut.passcode.util.CryptoUtil;  
  
@WebServlet("/admin/logout")  
public class AdminLogoutServlet extends HttpServlet {  
 private AuditLogDao auditLogDao = new AuditLogDao();  
   
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 doPost(request, response);  
 }  
   
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 HttpSession session = request.getSession(false);  
   
 if (session != null) {  
 Admin admin = (Admin) session.getAttribute("admin");  
 String ipAddress = getClientIpAddress(request);  
   
 if (admin != null) {  
 System.*out*.println("INFO: Admin logout - User: " + admin.getLoginName());  
 System.*out*.println("INFO: Logout time: " + LocalDateTime.*now*());  
   
 // Record logout audit log  
 try {  
 AuditLog log = new AuditLog(admin.getId(), admin.getFullName(), "退出登录",   
 "管理员退出登录", ipAddress);  
 log.setHmacSignature(CryptoUtil.*generateHmacSm3*(log.toString()));  
 auditLogDao.addAuditLog(log);  
 System.*out*.println("INFO: Logout audit log recorded");  
 } catch (Exception e) {  
 System.*out*.println("ERROR: Failed to record logout audit log");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 }  
 }  
   
 // Invalidate session  
 session.invalidate();  
 System.*out*.println("INFO: Session invalidated");  
 }  
   
 // Redirect to login page  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 }  
   
 private String getClientIpAddress(HttpServletRequest request) {  
 String xForwardedFor = request.getHeader("X-Forwarded-For");  
 if (xForwardedFor != null && !xForwardedFor.isEmpty() && !"unknown".equalsIgnoreCase(xForwardedFor)) {  
 return xForwardedFor.split(",")[0];  
 }  
 String xRealIp = request.getHeader("X-Real-IP");  
 if (xRealIp != null && !xRealIp.isEmpty() && !"unknown".equalsIgnoreCase(xRealIp)) {  
 return xRealIp;  
 }  
 return request.getRemoteAddr();  
 }  
}

#### src/main/java/com/zjut/passcode/controller/AdminManageServlet.java

package com.zjut.passcode.controller;  
  
import java.io.IOException;  
import java.util.List;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import javax.servlet.http.HttpSession;  
  
import com.zjut.passcode.bean.Admin;  
import com.zjut.passcode.bean.AuditLog;  
import com.zjut.passcode.bean.Department;  
import com.zjut.passcode.dao.AdminDao;  
import com.zjut.passcode.dao.AuditLogDao;  
import com.zjut.passcode.dao.DepartmentDao;  
import com.zjut.passcode.util.CryptoUtil;  
  
@WebServlet("/admin/manage")  
public class AdminManageServlet extends HttpServlet {  
 private AdminDao adminDao = new AdminDao();  
 private DepartmentDao departmentDao = new DepartmentDao();  
 private AuditLogDao auditLogDao = new AuditLogDao();  
   
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 HttpSession session = request.getSession(false);  
 if (session == null || session.getAttribute("admin") == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
   
 String adminRole = (String) session.getAttribute("adminRole");  
 if (!"SYSTEM\_ADMIN".equals(adminRole) && !"SCHOOL\_ADMIN".equals(adminRole)) {  
 request.setAttribute("error", "权限不足");  
 request.getRequestDispatcher("/error.jsp").forward(request, response);  
 return;  
 }  
   
 String action = request.getParameter("action");  
 if ("list".equals(action) || action == null) {  
 listAdmins(request, response);  
 } else if ("add".equals(action)) {  
 showAddForm(request, response);  
 } else if ("edit".equals(action)) {  
 showEditForm(request, response);  
 } else if ("delete".equals(action)) {  
 deleteAdmin(request, response);  
 }  
 }  
   
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 HttpSession session = request.getSession(false);  
 if (session == null || session.getAttribute("admin") == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
   
 String adminRole = (String) session.getAttribute("adminRole");  
 if (!"SYSTEM\_ADMIN".equals(adminRole) && !"SCHOOL\_ADMIN".equals(adminRole)) {  
 request.setAttribute("error", "权限不足");  
 request.getRequestDispatcher("/error.jsp").forward(request, response);  
 return;  
 }  
   
 String action = request.getParameter("action");  
 if ("add".equals(action)) {  
 addAdmin(request, response);  
 } else if ("update".equals(action)) {  
 updateAdmin(request, response);  
 } else if ("changePassword".equals(action)) {  
 changePassword(request, response);  
 }  
 }  
   
 private void listAdmins(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 List<Admin> admins = adminDao.getAllAdmins();  
 request.setAttribute("admins", admins);  
 request.getRequestDispatcher("/admin/admin\_manage.jsp").forward(request, response);  
 }  
   
 private void showAddForm(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 List<Department> departments = departmentDao.getAllDepartments();  
 request.setAttribute("departments", departments);  
 request.getRequestDispatcher("/admin/admin\_add.jsp").forward(request, response);  
 }  
   
 private void showEditForm(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 String idStr = request.getParameter("id");  
 if (idStr != null && !idStr.trim().isEmpty()) {  
 try {  
 int id = Integer.*parseInt*(idStr);  
 Admin admin = adminDao.getAdminById(id);  
 if (admin != null) {  
 List<Department> departments = departmentDao.getAllDepartments();  
 request.setAttribute("admin", admin);  
 request.setAttribute("departments", departments);  
 request.getRequestDispatcher("/admin/admin\_edit.jsp").forward(request, response);  
 return;  
 }  
 } catch (NumberFormatException e) {  
 // 忽略  
 }  
 }  
 request.setAttribute("error", "管理员不存在");  
 listAdmins(request, response);  
 }  
   
 private void addAdmin(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 request.setCharacterEncoding("UTF-8");  
   
 String loginName = request.getParameter("loginName");  
 String password = request.getParameter("password");  
 String fullName = request.getParameter("fullName");  
 String deptIdStr = request.getParameter("deptId");  
 String phone = request.getParameter("phone");  
 String role = request.getParameter("role");  
   
 // 验证输入  
 if (loginName == null || loginName.trim().isEmpty() ||  
 password == null || password.trim().isEmpty() ||  
 fullName == null || fullName.trim().isEmpty() ||  
 phone == null || phone.trim().isEmpty() ||  
 role == null || role.trim().isEmpty()) {  
 request.setAttribute("error", "请填写所有必填字段");  
 showAddForm(request, response);  
 return;  
 }  
   
 // 验证密码复杂度  
 if (!CryptoUtil.*validatePasswordComplexity*(password)) {  
 request.setAttribute("error", "密码必须包含8位以上，包含数字、大小写字母、特殊字符");  
 showAddForm(request, response);  
 return;  
 }  
   
 // 检查登录名是否已存在  
 Admin existingAdmin = adminDao.getAdminByLoginName(loginName);  
 if (existingAdmin != null) {  
 request.setAttribute("error", "登录名已存在");  
 showAddForm(request, response);  
 return;  
 }  
   
 try {  
 // 加密密码和手机号  
 String hashedPassword = CryptoUtil.*sm3Hash*(password);  
 String encryptedPhone = CryptoUtil.*sm4Encrypt*(phone, "campus\_pass\_key\_16");  
   
 Admin admin = new Admin();  
 admin.setLoginName(loginName);  
 admin.setPasswordHash(hashedPassword);  
 admin.setFullName(fullName);  
 admin.setDeptId(deptIdStr != null && !deptIdStr.trim().isEmpty() ? Integer.*parseInt*(deptIdStr) : 0);  
 admin.setPhone(encryptedPhone);  
 admin.setRole(role);  
   
 if (adminDao.addAdmin(admin)) {  
 // 记录审计日志  
 HttpSession session = request.getSession();  
 int currentAdminId = (Integer) session.getAttribute("adminId");  
 String currentAdminName = (String) session.getAttribute("adminName");  
 AuditLog log = new AuditLog(currentAdminId, currentAdminName, "添加管理员",   
 "添加管理员: " + loginName, request.getRemoteAddr());  
 log.setHmacSignature(CryptoUtil.*generateHmacSm3*(log.toString()));  
 auditLogDao.addAuditLog(log);  
   
 request.setAttribute("message", "管理员添加成功");  
 } else {  
 request.setAttribute("error", "管理员添加失败");  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 request.setAttribute("error", "系统错误，请重试");  
 }  
   
 listAdmins(request, response);  
 }  
   
 private void updateAdmin(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 request.setCharacterEncoding("UTF-8");  
   
 String idStr = request.getParameter("id");  
 String loginName = request.getParameter("loginName");  
 String fullName = request.getParameter("fullName");  
 String deptIdStr = request.getParameter("deptId");  
 String phone = request.getParameter("phone");  
 String role = request.getParameter("role");  
   
 if (idStr == null || idStr.trim().isEmpty()) {  
 request.setAttribute("error", "管理员ID不能为空");  
 listAdmins(request, response);  
 return;  
 }  
   
 try {  
 int id = Integer.*parseInt*(idStr);  
 Admin admin = adminDao.getAdminById(id);  
 if (admin == null) {  
 request.setAttribute("error", "管理员不存在");  
 listAdmins(request, response);  
 return;  
 }  
   
 // 更新信息  
 admin.setLoginName(loginName);  
 admin.setFullName(fullName);  
 admin.setDeptId(deptIdStr != null && !deptIdStr.trim().isEmpty() ? Integer.*parseInt*(deptIdStr) : 0);  
 admin.setPhone(CryptoUtil.*sm4Encrypt*(phone, "campus\_pass\_key\_"));  
 admin.setRole(role);  
   
 if (adminDao.updateAdmin(admin)) {  
 // 记录审计日志  
 HttpSession session = request.getSession();  
 int currentAdminId = (Integer) session.getAttribute("adminId");  
 String currentAdminName = (String) session.getAttribute("adminName");  
 AuditLog log = new AuditLog(currentAdminId, currentAdminName, "更新管理员",   
 "更新管理员: " + loginName, request.getRemoteAddr());  
 log.setHmacSignature(CryptoUtil.*generateHmacSm3*(log.toString()));  
 auditLogDao.addAuditLog(log);  
   
 request.setAttribute("message", "管理员更新成功");  
 } else {  
 request.setAttribute("error", "管理员更新失败");  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 request.setAttribute("error", "系统错误，请重试");  
 }  
   
 listAdmins(request, response);  
 }  
   
 private void deleteAdmin(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 String idStr = request.getParameter("id");  
 if (idStr == null || idStr.trim().isEmpty()) {  
 request.setAttribute("error", "管理员ID不能为空");  
 listAdmins(request, response);  
 return;  
 }  
   
 try {  
 int id = Integer.*parseInt*(idStr);  
 Admin admin = adminDao.getAdminById(id);  
 if (admin == null) {  
 request.setAttribute("error", "管理员不存在");  
 listAdmins(request, response);  
 return;  
 }  
   
 if (adminDao.deleteAdmin(id)) {  
 // 记录审计日志  
 HttpSession session = request.getSession();  
 int currentAdminId = (Integer) session.getAttribute("adminId");  
 String currentAdminName = (String) session.getAttribute("adminName");  
 AuditLog log = new AuditLog(currentAdminId, currentAdminName, "删除管理员",   
 "删除管理员: " + admin.getLoginName(), request.getRemoteAddr());  
 log.setHmacSignature(CryptoUtil.*generateHmacSm3*(log.toString()));  
 auditLogDao.addAuditLog(log);  
   
 request.setAttribute("message", "管理员删除成功");  
 } else {  
 request.setAttribute("error", "管理员删除失败");  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 request.setAttribute("error", "系统错误，请重试");  
 }  
   
 listAdmins(request, response);  
 }  
   
 private void changePassword(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 request.setCharacterEncoding("UTF-8");  
   
 String idStr = request.getParameter("id");  
 String newPassword = request.getParameter("newPassword");  
   
 if (idStr == null || idStr.trim().isEmpty() ||   
 newPassword == null || newPassword.trim().isEmpty()) {  
 request.setAttribute("error", "参数错误");  
 listAdmins(request, response);  
 return;  
 }  
   
 // 验证密码复杂度  
 if (!CryptoUtil.*validatePasswordComplexity*(newPassword)) {  
 request.setAttribute("error", "密码必须包含8位以上，包含数字、大小写字母、特殊字符");  
 listAdmins(request, response);  
 return;  
 }  
   
 try {  
 int id = Integer.*parseInt*(idStr);  
 String hashedPassword = CryptoUtil.*sm3Hash*(newPassword);  
   
 if (adminDao.updatePassword(id, hashedPassword)) {  
 // 记录审计日志  
 HttpSession session = request.getSession();  
 int currentAdminId = (Integer) session.getAttribute("adminId");  
 String currentAdminName = (String) session.getAttribute("adminName");  
 AuditLog log = new AuditLog(currentAdminId, currentAdminName, "修改密码",   
 "修改管理员密码，ID: " + id, request.getRemoteAddr());  
 log.setHmacSignature(CryptoUtil.*generateHmacSm3*(log.toString()));  
 auditLogDao.addAuditLog(log);  
   
 request.setAttribute("message", "密码修改成功");  
 } else {  
 request.setAttribute("error", "密码修改失败");  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 request.setAttribute("error", "系统错误，请重试");  
 }  
   
 listAdmins(request, response);  
 }  
}

#### src/main/java/com/zjut/passcode/controller/AdminReportsServlet.java

package com.zjut.passcode.controller;  
  
import java.io.IOException;  
import java.util.List;  
import java.util.Map;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import javax.servlet.http.HttpSession;  
  
import com.zjut.passcode.bean.Admin;  
import com.zjut.passcode.dao.AppointmentDao;  
  
@WebServlet("/admin/reports")  
public class AdminReportsServlet extends HttpServlet {  
   
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
   
 // Check if admin is logged in  
 HttpSession session = request.getSession(false);  
 if (session == null || session.getAttribute("admin") == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
   
 Admin admin = (Admin) session.getAttribute("admin");  
 String role = admin.getRole();  
 boolean canReportPublic = admin.isCanReportPublicAppointment();  
 System.*out*.println("INFO: Admin reports page accessed by: " + admin.getLoginName());  
   
 // Set attributes for JSP  
 request.setAttribute("admin", admin);  
   
 // 统计数据  
 AppointmentDao appointmentDao = new AppointmentDao();  
 List<Map<String, Object>> applyMonthStats = null;  
 List<Map<String, Object>> entryMonthStats = null;  
 List<Map<String, Object>> campusStats = null;  
 List<Map<String, Object>> deptStats = null;  
 if ("SYSTEM\_ADMIN".equals(role) || "SCHOOL\_ADMIN".equals(role)) {  
 // 学校/系统管理员可查所有  
 applyMonthStats = appointmentDao.getStatsByApplyMonth();  
 entryMonthStats = appointmentDao.getStatsByEntryMonth();  
 campusStats = appointmentDao.getStatsByCampus();  
 deptStats = appointmentDao.getStatsByOfficialDept();  
 } else if ("DEPT\_ADMIN".equals(role)) {  
 // 部门管理员根据权限统计预约  
 if (canReportPublic) {  
 // 可以统计社会公众预约，统计本部门所有预约  
 applyMonthStats = appointmentDao.getStatsByApplyMonthForDept(admin.getDeptId(), "PUBLIC");  
 List<Map<String, Object>> officialApply = appointmentDao.getStatsByApplyMonthForDept(admin.getDeptId(), "OFFICIAL");  
 applyMonthStats.addAll(officialApply);  
 entryMonthStats = appointmentDao.getStatsByEntryMonthForDept(admin.getDeptId(), "PUBLIC");  
 List<Map<String, Object>> officialEntry = appointmentDao.getStatsByEntryMonthForDept(admin.getDeptId(), "OFFICIAL");  
 entryMonthStats.addAll(officialEntry);  
 campusStats = appointmentDao.getStatsByCampusForDept(admin.getDeptId(), "PUBLIC");  
 List<Map<String, Object>> officialCampus = appointmentDao.getStatsByCampusForDept(admin.getDeptId(), "OFFICIAL");  
 campusStats.addAll(officialCampus);  
 deptStats = appointmentDao.getStatsByOfficialDeptForDept(admin.getDeptId());  
 } else {  
 // 只能统计公务预约  
 applyMonthStats = appointmentDao.getStatsByApplyMonthForDept(admin.getDeptId(), "OFFICIAL");  
 entryMonthStats = appointmentDao.getStatsByEntryMonthForDept(admin.getDeptId(), "OFFICIAL");  
 campusStats = appointmentDao.getStatsByCampusForDept(admin.getDeptId(), "OFFICIAL");  
 deptStats = appointmentDao.getStatsByOfficialDeptForDept(admin.getDeptId());  
 }  
 } else {  
 // 其它角色无权访问  
 response.sendError(HttpServletResponse.*SC\_FORBIDDEN*, "无权访问统计报告");  
 return;  
 }  
 request.setAttribute("applyMonthStats", applyMonthStats);  
 request.setAttribute("entryMonthStats", entryMonthStats);  
 request.setAttribute("campusStats", campusStats);  
 request.setAttribute("deptStats", deptStats);  
   
 // Forward to reports page  
 request.getRequestDispatcher("/admin/reports.jsp").forward(request, response);  
 }  
   
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 doGet(request, response);  
 }  
}

#### src/main/java/com/zjut/passcode/controller/AdminUsersServlet.java

package com.zjut.passcode.controller;  
  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import javax.servlet.http.HttpSession;  
  
import com.zjut.passcode.bean.Admin;  
import com.zjut.passcode.bean.AuditLog;  
import com.zjut.passcode.bean.Department;  
import com.zjut.passcode.dao.AdminDao;  
import com.zjut.passcode.dao.AuditLogDao;  
import com.zjut.passcode.dao.DepartmentDao;  
import com.zjut.passcode.util.CryptoUtil;  
  
@WebServlet("/admin/users")  
public class AdminUsersServlet extends HttpServlet {  
 private AdminDao adminDao = new AdminDao();  
 private DepartmentDao departmentDao = new DepartmentDao();  
 private AuditLogDao auditLogDao = new AuditLogDao();  
   
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
   
 // Check if admin is logged in  
 HttpSession session = request.getSession(false);  
 if (session == null || session.getAttribute("admin") == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
   
 Admin admin = (Admin) session.getAttribute("admin");  
 String adminRole = admin.getRole();  
 if (!"SYSTEM\_ADMIN".equals(adminRole) && !"SCHOOL\_ADMIN".equals(adminRole)) {  
 request.setAttribute("error", "权限不足");  
 request.getRequestDispatcher("/error.jsp").forward(request, response);  
 return;  
 }  
   
 String action = request.getParameter("action");  
 List<Department> departments = departmentDao.getAllDepartments();  
 request.setAttribute("departments", departments);  
 if ("add".equals(action)) {  
 showAddForm(request, response);  
 return;  
 } else if ("editUser".equals(action)) {  
 showEditForm(request, response);  
 return;  
 } else if ("resetPwd".equals(action)) {  
 showResetPwdForm(request, response);  
 return;  
 }  
   
 System.*out*.println("INFO: Admin users page accessed by: " + admin.getLoginName());  
   
 try {  
 // Get admins based on current admin's role  
 List<Admin> admins;  
 if ("SYSTEM\_ADMIN".equals(adminRole)) {  
 // 系统管理员可以查看所有管理员  
 admins = adminDao.getAllAdmins();  
 } else if ("SCHOOL\_ADMIN".equals(adminRole)) {  
 // 学校管理员只能查看部门管理员  
 admins = adminDao.getAdminsByRole("DEPT\_ADMIN");  
 } else {  
 admins = new ArrayList<>();  
 }  
   
 System.*out*.println("INFO: Retrieved " + admins.size() + " admin users");  
   
 // Set attributes for JSP  
 request.setAttribute("admins", admins);  
 request.setAttribute("currentAdmin", admin);  
   
 // Forward to users management page  
 request.getRequestDispatcher("/admin/users.jsp").forward(request, response);  
   
 } catch (Exception e) {  
 System.*out*.println("ERROR: Failed to retrieve admin users");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 e.printStackTrace();  
   
 request.setAttribute("error", "获取用户信息失败");  
 request.setAttribute("errorDetails", "错误详情: " + e.getMessage());  
 request.getRequestDispatcher("/error.jsp").forward(request, response);  
 }  
 }  
   
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 HttpSession session = request.getSession(false);  
 if (session == null || session.getAttribute("admin") == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
 Admin admin = (Admin) session.getAttribute("admin");  
 String adminRole = admin.getRole();  
 if (!"SYSTEM\_ADMIN".equals(adminRole) && !"SCHOOL\_ADMIN".equals(adminRole)) {  
 request.setAttribute("error", "权限不足");  
 request.getRequestDispatcher("/error.jsp").forward(request, response);  
 return;  
 }  
 String action = request.getParameter("action");  
 if ("add".equals(action)) {  
 addAdmin(request, response);  
 return;  
 } else if ("edit".equals(action)) {  
 editAdmin(request, response);  
 return;  
 } else if ("delete".equals(action)) {  
 deleteAdmin(request, response);  
 return;  
 } else if ("resetPwd".equals(action)) {  
 resetPassword(request, response);  
 return;  
 }  
 doGet(request, response);  
 }  
  
 private void showAddForm(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
 List<Department> departments = departmentDao.getAllDepartments();  
 request.setAttribute("departments", departments);  
 request.getRequestDispatcher("/admin/users.jsp?action=showAddForm").forward(request, response);  
 }  
  
 private void addAdmin(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
 request.setCharacterEncoding("UTF-8");  
 Admin admin = (Admin) request.getSession().getAttribute("admin");  
 String adminRole = admin.getRole();  
 String loginName = request.getParameter("loginName");  
 String password = request.getParameter("password");  
 String fullName = request.getParameter("fullName");  
 String deptIdStr = request.getParameter("deptId");  
 String phone = request.getParameter("phone");  
 String role = request.getParameter("role");  
 boolean canManagePublicAppointment = "1".equals(request.getParameter("canManagePublicAppointment"));  
 boolean canReportPublicAppointment = "1".equals(request.getParameter("canReportPublicAppointment"));  
   
 // 权限检查：学校管理员只能添加部门管理员  
 if ("SCHOOL\_ADMIN".equals(adminRole) && !"DEPT\_ADMIN".equals(role)) {  
 request.setAttribute("error", "学校管理员只能添加部门管理员");  
 showAddForm(request, response);  
 return;  
 }  
   
 if (loginName == null || loginName.trim().isEmpty() ||  
 password == null || password.trim().isEmpty() ||  
 fullName == null || fullName.trim().isEmpty() ||  
 phone == null || phone.trim().isEmpty() ||  
 role == null || role.trim().isEmpty()) {  
 request.setAttribute("error", "请填写所有必填字段");  
 showAddForm(request, response);  
 return;  
 }  
 if (!CryptoUtil.*validatePasswordComplexity*(password)) {  
 request.setAttribute("error", "密码必须包含8位以上，包含数字、大小写字母、特殊字符");  
 showAddForm(request, response);  
 return;  
 }  
 Admin existingAdmin = adminDao.getAdminByLoginName(loginName);  
 if (existingAdmin != null) {  
 request.setAttribute("error", "登录名已存在");  
 showAddForm(request, response);  
 return;  
 }  
 try {  
 String hashedPassword = CryptoUtil.*sm3Hash*(password);  
 String encryptedPhone = CryptoUtil.*sm4Encrypt*(phone, "campus\_pass\_key\_");  
 Admin newAdmin = new Admin();  
 newAdmin.setLoginName(loginName);  
 newAdmin.setPasswordHash(hashedPassword);  
 newAdmin.setFullName(fullName);  
 newAdmin.setDeptId(deptIdStr != null && !deptIdStr.trim().isEmpty() ? Integer.*parseInt*(deptIdStr) : 0);  
 newAdmin.setPhone(encryptedPhone);  
 newAdmin.setRole(role);  
 newAdmin.setCanManagePublicAppointment(canManagePublicAppointment);  
 newAdmin.setCanReportPublicAppointment(canReportPublicAppointment);  
 if (adminDao.addAdmin(newAdmin)) {  
 // 记录审计日志  
 int currentAdminId = admin.getId();  
 String currentAdminName = admin.getFullName();  
 AuditLog log = new AuditLog(currentAdminId, currentAdminName, "添加管理员", "添加管理员: " + loginName, request.getRemoteAddr());  
 log.setHmacSignature(CryptoUtil.*generateHmacSm3*(log.toString()));  
 auditLogDao.addAuditLog(log);  
 request.setAttribute("message", "管理员添加成功");  
 } else {  
 request.setAttribute("error", "管理员添加失败");  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 request.setAttribute("error", "系统错误，请重试");  
 }  
 doGet(request, response);  
 }  
  
 private void deleteAdmin(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
 int userId = Integer.*parseInt*(request.getParameter("userId"));  
 Admin currentAdmin = (Admin) request.getSession().getAttribute("admin");  
 String adminRole = currentAdmin.getRole();  
   
 if (userId == currentAdmin.getId()) {  
 request.setAttribute("error", "不能删除自己");  
 doGet(request, response);  
 return;  
 }  
   
 // 获取要删除的用户信息  
 Admin userToDelete = adminDao.getAdminById(userId);  
 if (userToDelete == null) {  
 request.setAttribute("error", "用户不存在");  
 doGet(request, response);  
 return;  
 }  
   
 // 权限检查：学校管理员只能删除部门管理员  
 if ("SCHOOL\_ADMIN".equals(adminRole) && !"DEPT\_ADMIN".equals(userToDelete.getRole())) {  
 request.setAttribute("error", "学校管理员只能删除部门管理员");  
 doGet(request, response);  
 return;  
 }  
   
 if (adminDao.deleteAdmin(userId)) {  
 request.setAttribute("message", "管理员删除成功");  
 } else {  
 request.setAttribute("error", "管理员删除失败");  
 }  
 doGet(request, response);  
 }  
  
 private void editAdmin(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
 int userId = Integer.*parseInt*(request.getParameter("userId"));  
 Admin currentAdmin = (Admin) request.getSession().getAttribute("admin");  
 String adminRole = currentAdmin.getRole();  
 String loginName = request.getParameter("loginName");  
 String fullName = request.getParameter("fullName");  
 String deptIdStr = request.getParameter("deptId");  
 String phone = request.getParameter("phone");  
 String role = request.getParameter("role");  
 boolean canManagePublicAppointment = "1".equals(request.getParameter("canManagePublicAppointment"));  
 boolean canReportPublicAppointment = "1".equals(request.getParameter("canReportPublicAppointment"));  
   
 // 获取要编辑的用户信息  
 Admin user = adminDao.getAdminById(userId);  
 if (user == null) {  
 request.setAttribute("error", "用户不存在");  
 doGet(request, response);  
 return;  
 }  
   
 // 权限检查：学校管理员只能编辑部门管理员，且不能改变角色  
 if ("SCHOOL\_ADMIN".equals(adminRole)) {  
 if (!"DEPT\_ADMIN".equals(user.getRole())) {  
 request.setAttribute("error", "学校管理员只能编辑部门管理员");  
 doGet(request, response);  
 return;  
 }  
 if (!"DEPT\_ADMIN".equals(role)) {  
 request.setAttribute("error", "学校管理员不能将部门管理员改为其他角色");  
 doGet(request, response);  
 return;  
 }  
 }  
   
 if (loginName == null || loginName.trim().isEmpty() ||  
 fullName == null || fullName.trim().isEmpty() ||  
 phone == null || phone.trim().isEmpty() ||  
 role == null || role.trim().isEmpty()) {  
 request.setAttribute("error", "请填写所有必填字段");  
 request.getRequestDispatcher("/admin/edit\_user.jsp?userId=" + userId).forward(request, response);  
 return;  
 }  
   
 user.setLoginName(loginName);  
 user.setFullName(fullName);  
 user.setDeptId(deptIdStr != null && !deptIdStr.trim().isEmpty() ? Integer.*parseInt*(deptIdStr) : 0);  
 user.setPhone(phone);  
 user.setRole(role);  
 user.setCanManagePublicAppointment(canManagePublicAppointment);  
 user.setCanReportPublicAppointment(canReportPublicAppointment);  
 if (adminDao.updateAdmin(user)) {  
 request.setAttribute("message", "管理员信息修改成功");  
 } else {  
 request.setAttribute("error", "管理员信息修改失败");  
 }  
 doGet(request, response);  
 }  
  
 private void resetPassword(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
 int userId = Integer.*parseInt*(request.getParameter("userId"));  
 String newPassword = request.getParameter("newPassword");  
 String confirmPassword = request.getParameter("confirmPassword");  
   
 // 基本验证：密码不能为空  
 if (newPassword == null || newPassword.trim().isEmpty()) {  
 request.setAttribute("error", "新密码不能为空");  
 showResetPwdForm(request, response);  
 return;  
 }  
   
 // 基本验证：确认密码不能为空  
 if (confirmPassword == null || confirmPassword.trim().isEmpty()) {  
 request.setAttribute("error", "确认密码不能为空");  
 showResetPwdForm(request, response);  
 return;  
 }  
   
 // 基本验证：两次密码必须一致（前端已验证，这里作为双重保险）  
 if (!newPassword.equals(confirmPassword)) {  
 request.setAttribute("error", "两次输入的密码不一致");  
 showResetPwdForm(request, response);  
 return;  
 }  
   
 // 验证密码复杂度  
 if (!CryptoUtil.*validatePasswordComplexity*(newPassword)) {  
 request.setAttribute("error", "密码必须包含8位以上，包含数字、大小写字母、特殊字符");  
 showResetPwdForm(request, response);  
 return;  
 }  
   
 String hashedPassword = CryptoUtil.*sm3Hash*(newPassword);  
 if (adminDao.updatePassword(userId, hashedPassword)) {  
 // 记录审计日志  
 Admin currentAdmin = (Admin) request.getSession().getAttribute("admin");  
 AuditLog log = new AuditLog(currentAdmin.getId(), currentAdmin.getFullName(), "重置密码",   
 "重置管理员密码，用户ID: " + userId, request.getRemoteAddr());  
 log.setHmacSignature(CryptoUtil.*generateHmacSm3*(log.toString()));  
 auditLogDao.addAuditLog(log);  
   
 request.setAttribute("message", "密码重置成功");  
 } else {  
 request.setAttribute("error", "密码重置失败");  
 }  
 doGet(request, response);  
 }  
  
 private void showEditForm(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
 int userId = Integer.*parseInt*(request.getParameter("userId"));  
 Admin user = adminDao.getAdminById(userId);  
 List<Department> departments = departmentDao.getAllDepartments();  
 request.setAttribute("editUser", user);  
 request.setAttribute("departments", departments);  
 request.getRequestDispatcher("/admin/edit\_user.jsp").forward(request, response);  
 }  
  
 private void showResetPwdForm(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
 int userId = Integer.*parseInt*(request.getParameter("userId"));  
 Admin user = adminDao.getAdminById(userId);  
 if (user == null) {  
 request.setAttribute("error", "用户不存在");  
 doGet(request, response);  
 return;  
 }  
 request.setAttribute("resetUser", user);  
 request.getRequestDispatcher("/admin/reset\_password.jsp").forward(request, response);  
 }  
}

#### src/main/java/com/zjut/passcode/controller/AppointmentApplyServlet.java

package com.zjut.passcode.controller;  
  
import java.io.IOException;  
import java.sql.Timestamp;  
import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
import com.zjut.passcode.bean.Appointment;  
import com.zjut.passcode.bean.AuditLog;  
import com.zjut.passcode.bean.Department;  
import com.zjut.passcode.dao.AppointmentDao;  
import com.zjut.passcode.dao.AuditLogDao;  
import com.zjut.passcode.dao.DepartmentDao;  
import com.zjut.passcode.util.CryptoUtil;  
  
@WebServlet("/appointment/apply")  
public class AppointmentApplyServlet extends HttpServlet {  
 private AppointmentDao appointmentDao = new AppointmentDao();  
 private AuditLogDao auditLogDao = new AuditLogDao();  
 private DepartmentDao departmentDao = new DepartmentDao();  
 private static final String *ENCRYPTION\_KEY* = "campus\_pass\_key\_";  
   
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 request.getRequestDispatcher("/appointment\_form.jsp").forward(request, response);  
 }  
   
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 request.setCharacterEncoding("UTF-8");  
 response.setContentType("text/html;charset=UTF-8");  
   
 // 获取表单数据  
 String visitorName = request.getParameter("visitorName");  
 String visitorPhone = request.getParameter("visitorPhone");  
 String visitorIdCard = request.getParameter("visitorIdCard");  
 String visitorUnit = request.getParameter("visitorUnit");  
 String campus = request.getParameter("campus");  
 String entryTimeStr = request.getParameter("entryTime");  
 String transportMode = request.getParameter("transportMode");  
 String licensePlate = request.getParameter("licensePlate");  
 String appointmentType = request.getParameter("appointmentType");  
 String ipAddress = getClientIpAddress(request);  
 // 新增公务预约相关字段  
 String officialDeptIdStr = request.getParameter("officialDeptId");  
 String officialContactPerson = request.getParameter("officialContactPerson");  
 String officialReason = request.getParameter("officialReason");  
   
 // 验证必填字段  
 if (visitorName == null || visitorName.trim().isEmpty() ||  
 visitorPhone == null || visitorPhone.trim().isEmpty() ||  
 visitorIdCard == null || visitorIdCard.trim().isEmpty() ||  
 campus == null || campus.trim().isEmpty() ||  
 entryTimeStr == null || entryTimeStr.trim().isEmpty() ||  
 appointmentType == null || appointmentType.trim().isEmpty()) {  
   
 request.setAttribute("error", "请填写所有必填字段");  
 request.getRequestDispatcher("/appointment\_form.jsp").forward(request, response);  
 return;  
 }  
   
 // 验证手机号格式  
 if (!visitorPhone.matches("^1[3-9]\\d{9}$")) {  
 request.setAttribute("error", "请输入正确的手机号码");  
 request.getRequestDispatcher("/appointment\_form.jsp").forward(request, response);  
 return;  
 }  
   
 // 验证身份证号格式  
 if (!visitorIdCard.matches("^[1-9]\\d{5}(18|19|20)\\d{2}((0[1-9])|(1[0-2]))(([0-2][1-9])|10|20|30|31)\\d{3}[0-9Xx]$")) {  
 request.setAttribute("error", "请输入正确的身份证号码");  
 request.getRequestDispatcher("/appointment\_form.jsp").forward(request, response);  
 return;  
 }  
   
 // 解析进校时间  
 LocalDateTime entryTime;  
 try {  
 entryTime = LocalDateTime.*parse*(entryTimeStr, DateTimeFormatter.*ofPattern*("yyyy-MM-dd'T'HH:mm"));  
 } catch (Exception e) {  
 request.setAttribute("error", "进校时间格式错误");  
 request.getRequestDispatcher("/appointment\_form.jsp").forward(request, response);  
 return;  
 }  
   
 // 验证访问日期（不能是过去的日期）  
 if (entryTime.isBefore(LocalDateTime.*now*())) {  
 request.setAttribute("error", "进校时间不能是过去的日期");  
 request.getRequestDispatcher("/appointment\_form.jsp").forward(request, response);  
 return;  
 }  
   
 try {  
 // 创建预约对象  
 Appointment appointment = new Appointment();  
 appointment.setVisitorName(visitorName.trim());  
 appointment.setVisitorPhone(CryptoUtil.*sm4Encrypt*(visitorPhone.trim(), *ENCRYPTION\_KEY*));  
 appointment.setVisitorIdCard(CryptoUtil.*sm4Encrypt*(visitorIdCard.trim(), *ENCRYPTION\_KEY*));  
 appointment.setVisitorUnit(visitorUnit != null ? visitorUnit.trim() : "");  
 appointment.setCampus(campus.trim());  
 appointment.setEntryTime(Timestamp.*valueOf*(entryTime));  
 appointment.setTransportMode(transportMode != null ? transportMode : "");  
 appointment.setLicensePlate(licensePlate != null ? licensePlate.trim() : "");  
 appointment.setAppointmentType(appointmentType);  
 appointment.setStatus("PENDING");  
 appointment.setCreatedAt(Timestamp.*valueOf*(LocalDateTime.*now*()));  
 // 公务预约字段赋值  
 if ("OFFICIAL".equals(appointmentType)) {  
 if (officialDeptIdStr != null && !officialDeptIdStr.trim().isEmpty()) {  
 int deptId = Integer.*parseInt*(officialDeptIdStr);  
 appointment.setOfficialDeptId(deptId);  
 Department dept = departmentDao.getDepartmentById(deptId);  
 if (dept != null) {  
 appointment.setOfficialDeptNo(dept.getDeptNo());  
 appointment.setOfficialDeptName(dept.getDeptName());  
 }  
 }  
 appointment.setOfficialContactPerson(officialContactPerson != null ? officialContactPerson.trim() : "");  
 appointment.setOfficialReason(officialReason != null ? officialReason.trim() : "");  
 }  
   
 // 保存预约  
 long appointmentId = appointmentDao.addAppointment(appointment);  
 if (appointmentId > 0) {  
 // 记录审计日志  
 AuditLog log = new AuditLog(1, visitorName, "预约申请",  
 "提交预约申请，预约ID: " + appointmentId, ipAddress);  
 log.setHmacSignature(CryptoUtil.*generateHmacSm3*(log.toString()));  
 auditLogDao.addAuditLog(log);  
   
 // 设置成功信息并跳转  
 request.setAttribute("appointmentId", appointmentId);  
 request.setAttribute("visitorName", visitorName);  
 request.setAttribute("entryTime", entryTimeStr);  
 request.getRequestDispatcher("/pass\_code.jsp").forward(request, response);  
 } else {  
 request.setAttribute("error", "预约提交失败，请重试");  
 request.getRequestDispatcher("/appointment\_form.jsp").forward(request, response);  
 }  
   
 } catch (Exception e) {  
 e.printStackTrace();  
 request.setAttribute("error", "系统错误，请稍后重试");  
 request.getRequestDispatcher("/appointment\_form.jsp").forward(request, response);  
 }  
 }  
   
 private String getClientIpAddress(HttpServletRequest request) {  
 String xForwardedFor = request.getHeader("X-Forwarded-For");  
 if (xForwardedFor != null && !xForwardedFor.isEmpty() && !"unknown".equalsIgnoreCase(xForwardedFor)) {  
 return xForwardedFor.split(",")[0];  
 }  
 String xRealIp = request.getHeader("X-Real-IP");  
 if (xRealIp != null && !xRealIp.isEmpty() && !"unknown".equalsIgnoreCase(xRealIp)) {  
 return xRealIp;  
 }  
 return request.getRemoteAddr();  
 }  
}

#### src/main/java/com/zjut/passcode/controller/AppointmentQueryServlet.java

package com.zjut.passcode.controller;  
  
import java.io.IOException;  
import java.util.List;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
import com.zjut.passcode.bean.Appointment;  
import com.zjut.passcode.dao.AppointmentDao;  
import com.zjut.passcode.util.CryptoUtil;  
  
@WebServlet("/appointment/query")  
public class AppointmentQueryServlet extends HttpServlet {  
 private AppointmentDao appointmentDao = new AppointmentDao();  
 private static final String *ENCRYPTION\_KEY* = "campus\_pass\_key\_";  
   
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 request.getRequestDispatcher("/my\_appointments.jsp").forward(request, response);  
 }  
   
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 request.setCharacterEncoding("UTF-8");  
 response.setContentType("text/html;charset=UTF-8");  
   
 String visitorName = request.getParameter("visitorName");  
 String visitorPhone = request.getParameter("visitorPhone");  
 String visitorIdCard = request.getParameter("visitorIdCard");  
   
 // 验证至少提供一个查询条件  
 if ((visitorName == null || visitorName.trim().isEmpty()) &&  
 (visitorPhone == null || visitorPhone.trim().isEmpty()) &&  
 (visitorIdCard == null || visitorIdCard.trim().isEmpty())) {  
 request.setAttribute("error", "请至少提供一个查询条件");  
 request.getRequestDispatcher("/my\_appointments.jsp").forward(request, response);  
 return;  
 }  
   
 try {  
 // 加密查询条件  
 String encryptedPhone = null;  
 String encryptedIdCard = null;  
   
 if (visitorPhone != null && !visitorPhone.trim().isEmpty()) {  
 encryptedPhone = CryptoUtil.*sm4Encrypt*(visitorPhone.trim(), *ENCRYPTION\_KEY*);  
 }  
 if (visitorIdCard != null && !visitorIdCard.trim().isEmpty()) {  
 encryptedIdCard = CryptoUtil.*sm4Encrypt*(visitorIdCard.trim(), *ENCRYPTION\_KEY*);  
 }  
   
 // 查询预约记录 - 使用现有的方法  
 List<Appointment> appointments = appointmentDao.getAppointmentsByVisitor(  
 visitorName != null ? visitorName.trim() : "",  
 encryptedIdCard != null ? encryptedIdCard : "",  
 encryptedPhone != null ? encryptedPhone : ""  
 );  
   
 // 解密敏感信息用于显示  
 for (Appointment appointment : appointments) {  
 if (appointment.getVisitorPhone() != null) {  
 appointment.setVisitorPhone(CryptoUtil.*sm4Decrypt*(appointment.getVisitorPhone(), *ENCRYPTION\_KEY*));  
 }  
 if (appointment.getVisitorIdCard() != null) {  
 appointment.setVisitorIdCard(CryptoUtil.*sm4Decrypt*(appointment.getVisitorIdCard(), *ENCRYPTION\_KEY*));  
 }  
 }  
   
 request.setAttribute("appointments", appointments);  
 request.setAttribute("queryName", visitorName);  
 request.setAttribute("queryPhone", visitorPhone);  
 request.setAttribute("queryIdCard", visitorIdCard);  
   
 if (appointments.isEmpty()) {  
 request.setAttribute("message", "未找到相关预约记录");  
 }  
   
 request.getRequestDispatcher("/my\_appointments.jsp").forward(request, response);  
   
 } catch (Exception e) {  
 e.printStackTrace();  
 request.setAttribute("error", "查询失败，请重试");  
 request.getRequestDispatcher("/my\_appointments.jsp").forward(request, response);  
 }  
 }  
}

#### src/main/java/com/zjut/passcode/controller/DepartmentOperationServlet.java

package com.zjut.passcode.controller;  
  
import java.io.IOException;  
import java.net.URLEncoder;  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import javax.servlet.http.HttpSession;  
  
import com.zjut.passcode.bean.Admin;  
import com.zjut.passcode.bean.Department;  
import com.zjut.passcode.dao.DepartmentDao;  
  
@WebServlet("/admin/department/\*")  
public class DepartmentOperationServlet extends HttpServlet {  
 private DepartmentDao departmentDao = new DepartmentDao();  
   
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
   
 // Set character encoding to handle Chinese characters  
 request.setCharacterEncoding("UTF-8");  
 response.setCharacterEncoding("UTF-8");  
   
 // Check if admin is logged in  
 HttpSession session = request.getSession(false);  
 if (session == null || session.getAttribute("admin") == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
   
 String pathInfo = request.getPathInfo();  
 if (pathInfo == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 return;  
 }  
   
 String[] pathParts = pathInfo.split("/");  
 if (pathParts.length < 2) {  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 return;  
 }  
   
 String operation = pathParts[1];  
   
 switch (operation) {  
 case "add":  
 showAddForm(request, response);  
 break;  
 case "edit":  
 showEditForm(request, response);  
 break;  
 case "delete":  
 deleteDepartment(request, response);  
 break;  
 default:  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 break;  
 }  
 }  
   
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
   
 // Set character encoding to handle Chinese characters  
 request.setCharacterEncoding("UTF-8");  
 response.setCharacterEncoding("UTF-8");  
   
 // Check if admin is logged in  
 HttpSession session = request.getSession(false);  
 if (session == null || session.getAttribute("admin") == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
   
 String pathInfo = request.getPathInfo();  
 if (pathInfo == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 return;  
 }  
   
 String[] pathParts = pathInfo.split("/");  
 if (pathParts.length < 2) {  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 return;  
 }  
   
 String operation = pathParts[1];  
   
 switch (operation) {  
 case "add":  
 addDepartment(request, response);  
 break;  
 case "edit":  
 editDepartment(request, response);  
 break;  
 default:  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 break;  
 }  
 }  
   
 private void showAddForm(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 request.getRequestDispatcher("/admin/add\_department.jsp").forward(request, response);  
 }  
   
 private void showEditForm(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 String idParam = request.getParameter("id");  
 if (idParam == null || idParam.trim().isEmpty()) {  
 request.setAttribute("error", "部门ID不能为空");  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 return;  
 }  
   
 try {  
 int id = Integer.*parseInt*(idParam);  
 Department department = departmentDao.getDepartmentById(id);  
   
 if (department == null) {  
 request.setAttribute("error", "部门不存在");  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 return;  
 }  
   
 request.setAttribute("department", department);  
 request.getRequestDispatcher("/admin/edit\_department.jsp").forward(request, response);  
   
 } catch (NumberFormatException e) {  
 request.setAttribute("error", "无效的部门ID");  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 }  
 }  
   
 private void addDepartment(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
   
 // Get form parameters  
 String deptNo = request.getParameter("deptNo");  
 String deptName = request.getParameter("deptName");  
 String deptType = request.getParameter("deptType");  
   
 // Debug: Print received parameters  
 System.*out*.println("DEBUG: Received parameters:");  
 System.*out*.println(" deptNo: " + deptNo);  
 System.*out*.println(" deptName: " + deptName);  
 System.*out*.println(" deptType: " + deptType);  
   
 // Validate input  
 if (deptNo == null || deptNo.trim().isEmpty() ||  
 deptName == null || deptName.trim().isEmpty() ||  
 deptType == null || deptType.trim().isEmpty()) {  
 request.setAttribute("error", "所有字段都是必填的");  
 request.setAttribute("deptNo", deptNo);  
 request.setAttribute("deptName", deptName);  
 request.setAttribute("deptType", deptType);  
 request.getRequestDispatcher("/admin/add\_department.jsp").forward(request, response);  
 return;  
 }  
   
 // Trim whitespace  
 deptNo = deptNo.trim();  
 deptName = deptName.trim();  
 deptType = deptType.trim();  
   
 // Validate department type  
 if (!isValidDepartmentType(deptType)) {  
 request.setAttribute("error", "无效的部门类型");  
 request.setAttribute("deptNo", deptNo);  
 request.setAttribute("deptName", deptName);  
 request.setAttribute("deptType", deptType);  
 request.getRequestDispatcher("/admin/add\_department.jsp").forward(request, response);  
 return;  
 }  
   
 try {  
 // Create department object  
 Department department = new Department(deptNo, deptName, deptType);  
   
 // Add to database  
 boolean success = departmentDao.addDepartment(department);  
   
 if (success) {  
 response.sendRedirect(request.getContextPath() + "/admin/departments?success=" + URLEncoder.*encode*("部门添加成功", "UTF-8"));  
 } else {  
 request.setAttribute("error", "部门添加失败，请检查部门编号是否已存在");  
 request.setAttribute("deptNo", deptNo);  
 request.setAttribute("deptName", deptName);  
 request.setAttribute("deptType", deptType);  
 request.getRequestDispatcher("/admin/add\_department.jsp").forward(request, response);  
 }  
   
 } catch (Exception e) {  
 System.*out*.println("ERROR: Failed to add department");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 e.printStackTrace();  
   
 request.setAttribute("error", "添加部门时发生错误: " + e.getMessage());  
 request.setAttribute("deptNo", deptNo);  
 request.setAttribute("deptName", deptName);  
 request.setAttribute("deptType", deptType);  
 request.getRequestDispatcher("/admin/add\_department.jsp").forward(request, response);  
 }  
 }  
   
 private void editDepartment(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
   
 // Get form parameters  
 String idParam = request.getParameter("id");  
 String deptNo = request.getParameter("deptNo");  
 String deptName = request.getParameter("deptName");  
 String deptType = request.getParameter("deptType");  
   
 // Debug: Print received parameters  
 System.*out*.println("DEBUG: Edit department parameters:");  
 System.*out*.println(" id: " + idParam);  
 System.*out*.println(" deptNo: " + deptNo);  
 System.*out*.println(" deptName: " + deptName);  
 System.*out*.println(" deptType: " + deptType);  
   
 // Validate input  
 if (idParam == null || idParam.trim().isEmpty() ||  
 deptNo == null || deptNo.trim().isEmpty() ||  
 deptName == null || deptName.trim().isEmpty() ||  
 deptType == null || deptType.trim().isEmpty()) {  
 request.setAttribute("error", "所有字段都是必填的");  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 return;  
 }  
   
 try {  
 int id = Integer.*parseInt*(idParam);  
   
 // Trim whitespace  
 deptNo = deptNo.trim();  
 deptName = deptName.trim();  
 deptType = deptType.trim();  
   
 // Validate department type  
 if (!isValidDepartmentType(deptType)) {  
 request.setAttribute("error", "无效的部门类型");  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 return;  
 }  
   
 // Get existing department  
 Department existingDept = departmentDao.getDepartmentById(id);  
 if (existingDept == null) {  
 request.setAttribute("error", "部门不存在");  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 return;  
 }  
   
 // Update department object  
 existingDept.setDeptNo(deptNo);  
 existingDept.setDeptName(deptName);  
 existingDept.setDeptType(deptType);  
   
 // Update in database  
 boolean success = departmentDao.updateDepartment(existingDept);  
   
 if (success) {  
 response.sendRedirect(request.getContextPath() + "/admin/departments?success=" + URLEncoder.*encode*("部门更新成功", "UTF-8"));  
 } else {  
 response.sendRedirect(request.getContextPath() + "/admin/departments?error=" + URLEncoder.*encode*("部门更新失败", "UTF-8"));  
 }  
   
 } catch (NumberFormatException e) {  
 request.setAttribute("error", "无效的部门ID");  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 } catch (Exception e) {  
 System.*out*.println("ERROR: Failed to update department");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 e.printStackTrace();  
   
 request.setAttribute("error", "更新部门时发生错误: " + e.getMessage());  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 }  
 }  
   
 private void deleteDepartment(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
   
 String idParam = request.getParameter("id");  
 if (idParam == null || idParam.trim().isEmpty()) {  
 request.setAttribute("error", "部门ID不能为空");  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 return;  
 }  
   
 try {  
 int id = Integer.*parseInt*(idParam);  
   
 // Check if department exists  
 Department department = departmentDao.getDepartmentById(id);  
 if (department == null) {  
 request.setAttribute("error", "部门不存在");  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 return;  
 }  
   
 // Delete from database  
 boolean success = departmentDao.deleteDepartment(id);  
   
 if (success) {  
 response.sendRedirect(request.getContextPath() + "/admin/departments?success=" + URLEncoder.*encode*("部门删除成功", "UTF-8"));  
 } else {  
 response.sendRedirect(request.getContextPath() + "/admin/departments?error=" + URLEncoder.*encode*("部门删除失败", "UTF-8"));  
 }  
   
 } catch (NumberFormatException e) {  
 request.setAttribute("error", "无效的部门ID");  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 } catch (Exception e) {  
 System.*out*.println("ERROR: Failed to delete department");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 e.printStackTrace();  
   
 request.setAttribute("error", "删除部门时发生错误: " + e.getMessage());  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 }  
 }  
   
 private boolean isValidDepartmentType(String deptType) {  
 return "行政部门".equals(deptType) ||   
 "直属部门".equals(deptType) ||   
 "学院".equals(deptType);  
 }  
}

#### src/main/java/com/zjut/passcode/controller/PassCodeViewServlet.java

package com.zjut.passcode.controller;  
  
import java.awt.image.BufferedImage;  
import java.io.ByteArrayOutputStream;  
import java.io.IOException;  
import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
  
import javax.imageio.ImageIO;  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
import com.google.zxing.BarcodeFormat;  
import com.google.zxing.common.BitMatrix;  
import com.google.zxing.qrcode.QRCodeWriter;  
import com.zjut.passcode.bean.Appointment;  
import com.zjut.passcode.dao.AppointmentDao;  
import com.zjut.passcode.util.CryptoUtil;  
  
@WebServlet("/passcode/view")  
public class PassCodeViewServlet extends HttpServlet {  
 private AppointmentDao appointmentDao = new AppointmentDao();  
 private static final String *ENCRYPTION\_KEY* = "campus\_pass\_key\_";  
   
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response)   
 throws ServletException, IOException {  
 request.setCharacterEncoding("UTF-8");  
 response.setContentType("text/html;charset=UTF-8");  
   
 String appointmentIdStr = request.getParameter("id");  
 if (appointmentIdStr == null || appointmentIdStr.trim().isEmpty()) {  
 response.sendError(HttpServletResponse.*SC\_BAD\_REQUEST*, "预约ID不能为空");  
 return;  
 }  
   
 try {  
 long appointmentId = Long.*parseLong*(appointmentIdStr);  
 Appointment appointment = appointmentDao.getAppointmentById(appointmentId);  
   
 if (appointment == null) {  
 response.sendError(HttpServletResponse.*SC\_NOT\_FOUND*, "预约不存在");  
 return;  
 }  
   
 // 解密敏感信息用于显示  
 if (appointment.getVisitorPhone() != null) {  
 appointment.setVisitorPhone(CryptoUtil.*sm4Decrypt*(appointment.getVisitorPhone(), *ENCRYPTION\_KEY*));  
 }  
 if (appointment.getVisitorIdCard() != null) {  
 appointment.setVisitorIdCard(CryptoUtil.*sm4Decrypt*(appointment.getVisitorIdCard(), *ENCRYPTION\_KEY*));  
 }  
   
 // 检查通行码是否有效  
 boolean isValid = isPassCodeValid(appointment);  
   
 // 生成二维码内容  
 String qrContent = generateQRContent(appointment, appointment.getVisitorIdCard());  
   
 // 生成二维码图片  
 String qrCodeBase64 = generateQRCode(qrContent);  
   
 // 设置请求属性  
 request.setAttribute("appointment", appointment);  
 request.setAttribute("isValid", isValid);  
 request.setAttribute("qrCodeBase64", qrCodeBase64);  
 request.setAttribute("qrContent", qrContent);  
 request.setAttribute("maskedVisitorName", CryptoUtil.*maskName*(appointment.getVisitorName()));  
 request.setAttribute("maskedIdCard", CryptoUtil.*maskIdCard*(appointment.getVisitorIdCard()));  
 request.setAttribute("decryptedIdCard", appointment.getVisitorIdCard());  
 request.setAttribute("decryptedPhone", appointment.getVisitorPhone());  
   
 // 转发到通行码页面  
 request.getRequestDispatcher("/pass\_code.jsp").forward(request, response);  
   
 } catch (NumberFormatException e) {  
 response.sendError(HttpServletResponse.*SC\_BAD\_REQUEST*, "无效的预约ID");  
 } catch (Exception e) {  
 e.printStackTrace();  
 response.sendError(HttpServletResponse.*SC\_INTERNAL\_SERVER\_ERROR*, "系统错误");  
 }  
 }  
   
 private boolean isPassCodeValid(Appointment appointment) {  
 if (!"APPROVED".equals(appointment.getStatus())) {  
 return false;  
 }  
   
 LocalDateTime now = LocalDateTime.*now*();  
 LocalDateTime entryTime = appointment.getEntryTime().toLocalDateTime();  
   
 // 通行码在进校当天0点到23:59:59都有效  
 LocalDateTime validStart = entryTime.toLocalDate().atStartOfDay();  
 LocalDateTime validEnd = entryTime.toLocalDate().atTime(23, 59, 59);  
   
 return !now.isBefore(validStart) && !now.isAfter(validEnd);  
 }  
   
 private String generateQRContent(Appointment appointment, String decryptedIdCard) {  
 StringBuilder content = new StringBuilder();  
 content.append("姓名:").append(CryptoUtil.*maskName*(appointment.getVisitorName())).append("\n");  
 content.append("身份证:").append(CryptoUtil.*maskIdCard*(decryptedIdCard)).append("\n");  
 content.append("校区:").append(appointment.getCampus()).append("\n");  
 content.append("进校时间:").append(appointment.getEntryTime().toLocalDateTime().format(DateTimeFormatter.*ofPattern*("yyyy-MM-dd HH:mm"))).append("\n");  
 content.append("预约类型:").append("OFFICIAL".equals(appointment.getAppointmentType()) ? "公务预约" : "社会公众预约").append("\n");  
 content.append("状态:").append(appointment.getStatus()).append("\n");  
 content.append("生成时间:").append(LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*("yyyy-MM-dd HH:mm:ss")));  
   
 return content.toString();  
 }  
   
 private String generateQRCode(String content) {  
 try {  
 QRCodeWriter qrCodeWriter = new QRCodeWriter();  
 BitMatrix bitMatrix = qrCodeWriter.encode(content, BarcodeFormat.*QR\_CODE*, 200, 200);  
 BufferedImage image = new BufferedImage(200, 200, BufferedImage.*TYPE\_INT\_RGB*);  
 for (int x = 0; x < 200; x++) {  
 for (int y = 0; y < 200; y++) {  
 image.setRGB(x, y, bitMatrix.get(x, y) ? 0xFF000000 : 0xFFFFFFFF);  
 }  
 }  
 ByteArrayOutputStream baos = new ByteArrayOutputStream();  
 ImageIO.*write*(image, "png", baos);  
 String base64 = java.util.Base64.*getEncoder*().encodeToString(baos.toByteArray());  
 return "data:image/png;base64," + base64;  
 } catch (Exception e) {  
 e.printStackTrace();  
 return "";  
 }  
 }  
}

#### src/main/java/com/zjut/passcode/controller/QRCodeServlet.java

package com.zjut.passcode.controller;  
  
import java.awt.image.BufferedImage;  
import java.io.IOException;  
  
import javax.imageio.ImageIO;  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
import com.google.zxing.BarcodeFormat;  
import com.google.zxing.WriterException;  
import com.google.zxing.common.BitMatrix;  
import com.google.zxing.qrcode.QRCodeWriter;  
  
@WebServlet("/qrcode")  
public class QRCodeServlet extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 String content = request.getParameter("content");  
 if (content == null || content.isEmpty()) {  
 response.sendError(HttpServletResponse.*SC\_BAD\_REQUEST*, "内容不能为空");  
 return;  
 }  
 try {  
 QRCodeWriter qrCodeWriter = new QRCodeWriter();  
 BitMatrix bitMatrix = qrCodeWriter.encode(content, BarcodeFormat.*QR\_CODE*, 200, 200);  
 BufferedImage image = new BufferedImage(200, 200, BufferedImage.*TYPE\_INT\_RGB*);  
 for (int x = 0; x < 200; x++) {  
 for (int y = 0; y < 200; y++) {  
 image.setRGB(x, y, bitMatrix.get(x, y) ? 0xFF000000 : 0xFFFFFFFF);  
 }  
 }  
 response.setContentType("image/png");  
 ImageIO.*write*(image, "png", response.getOutputStream());  
 } catch (WriterException e) {  
 response.sendError(HttpServletResponse.*SC\_INTERNAL\_SERVER\_ERROR*, "二维码生成失败");  
 }  
 }  
}

#### src/main/java/com/zjut/passcode/dao/AdminDao.java

package com.zjut.passcode.dao;  
  
import java.sql.Connection;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.sql.Timestamp;  
import java.util.ArrayList;  
import java.util.List;  
  
import com.zjut.passcode.bean.Admin;  
  
public class AdminDao extends BaseDao {  
 public boolean addAdmin(Admin admin) {  
 String sql = "INSERT INTO admin (login\_name, password\_hash, full\_name, dept\_id, phone, role, can\_manage\_public\_appointment, can\_report\_public\_appointment) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setString(1, admin.getLoginName());  
 pstmt.setString(2, admin.getPasswordHash());  
 pstmt.setString(3, admin.getFullName());  
 pstmt.setInt(4, admin.getDeptId());  
 pstmt.setString(5, admin.getPhone());  
 pstmt.setString(6, admin.getRole());  
 pstmt.setBoolean(7, admin.isCanManagePublicAppointment());  
 pstmt.setBoolean(8, admin.isCanReportPublicAppointment());  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 } finally {  
 close(conn, pstmt);  
 }  
 }  
   
 public boolean updateAdmin(Admin admin) {  
 String sql = "UPDATE admin SET login\_name=?, full\_name=?, dept\_id=?, phone=?, role=?, can\_manage\_public\_appointment=?, can\_report\_public\_appointment=? WHERE id=?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setString(1, admin.getLoginName());  
 pstmt.setString(2, admin.getFullName());  
 pstmt.setInt(3, admin.getDeptId());  
 pstmt.setString(4, admin.getPhone());  
 pstmt.setString(5, admin.getRole());  
 pstmt.setBoolean(6, admin.isCanManagePublicAppointment());  
 pstmt.setBoolean(7, admin.isCanReportPublicAppointment());  
 pstmt.setInt(8, admin.getId());  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 } finally {  
 close(conn, pstmt);  
 }  
 }  
   
 public boolean updatePassword(int adminId, String newPasswordHash) {  
 String sql = "UPDATE admin SET password\_hash=?, password\_last\_changed=CURRENT\_TIMESTAMP WHERE id=?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setString(1, newPasswordHash);  
 pstmt.setInt(2, adminId);  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 } finally {  
 close(conn, pstmt);  
 }  
 }  
   
 public boolean deleteAdmin(int id) {  
 String sql = "DELETE FROM admin WHERE id=?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setInt(1, id);  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 } finally {  
 close(conn, pstmt);  
 }  
 }  
   
 public Admin getAdminById(int id) {  
 String sql = "SELECT a.\*, d.dept\_name FROM admin a LEFT JOIN department d ON a.dept\_id = d.id WHERE a.id=?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setInt(1, id);  
 rs = pstmt.executeQuery();  
 if (rs.next()) {  
 Admin admin = new Admin();  
 admin.setId(rs.getInt("id"));  
 admin.setLoginName(rs.getString("login\_name"));  
 admin.setPasswordHash(rs.getString("password\_hash"));  
 admin.setFullName(rs.getString("full\_name"));  
 admin.setDeptId(rs.getInt("dept\_id"));  
 admin.setPhone(rs.getString("phone"));  
 admin.setRole(rs.getString("role"));  
 admin.setPasswordLastChanged(rs.getTimestamp("password\_last\_changed"));  
 admin.setFailedLoginAttempts(rs.getInt("failed\_login\_attempts"));  
 admin.setLockoutUntil(rs.getTimestamp("lockout\_until"));  
 admin.setDeptName(rs.getString("dept\_name"));  
 admin.setCanManagePublicAppointment(rs.getBoolean("can\_manage\_public\_appointment"));  
 admin.setCanReportPublicAppointment(rs.getBoolean("can\_report\_public\_appointment"));  
 return admin;  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 return null;  
 }  
   
 public Admin getAdminByLoginName(String loginName) {  
 String sql = "SELECT a.\*, d.dept\_name FROM admin a LEFT JOIN department d ON a.dept\_id = d.id WHERE a.login\_name=?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setString(1, loginName);  
 rs = pstmt.executeQuery();  
 if (rs.next()) {  
 Admin admin = new Admin();  
 admin.setId(rs.getInt("id"));  
 admin.setLoginName(rs.getString("login\_name"));  
 admin.setPasswordHash(rs.getString("password\_hash"));  
 admin.setFullName(rs.getString("full\_name"));  
 admin.setDeptId(rs.getInt("dept\_id"));  
 admin.setPhone(rs.getString("phone"));  
 admin.setRole(rs.getString("role"));  
 admin.setPasswordLastChanged(rs.getTimestamp("password\_last\_changed"));  
 admin.setFailedLoginAttempts(rs.getInt("failed\_login\_attempts"));  
 admin.setLockoutUntil(rs.getTimestamp("lockout\_until"));  
 admin.setDeptName(rs.getString("dept\_name"));  
 admin.setCanManagePublicAppointment(rs.getBoolean("can\_manage\_public\_appointment"));  
 admin.setCanReportPublicAppointment(rs.getBoolean("can\_report\_public\_appointment"));  
 return admin;  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 return null;  
 }  
   
 public List<Admin> getAllAdmins() {  
 List<Admin> admins = new ArrayList<>();  
 String sql = "SELECT a.\*, d.dept\_name FROM admin a LEFT JOIN department d ON a.dept\_id = d.id ORDER BY a.id";  
 Connection conn = null;  
 Statement stmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 stmt = conn.createStatement();  
 rs = stmt.executeQuery(sql);  
 while (rs.next()) {  
 Admin admin = new Admin();  
 admin.setId(rs.getInt("id"));  
 admin.setLoginName(rs.getString("login\_name"));  
 admin.setPasswordHash(rs.getString("password\_hash"));  
 admin.setFullName(rs.getString("full\_name"));  
 admin.setDeptId(rs.getInt("dept\_id"));  
 admin.setPhone(rs.getString("phone"));  
 admin.setRole(rs.getString("role"));  
 admin.setPasswordLastChanged(rs.getTimestamp("password\_last\_changed"));  
 admin.setFailedLoginAttempts(rs.getInt("failed\_login\_attempts"));  
 admin.setLockoutUntil(rs.getTimestamp("lockout\_until"));  
 admin.setDeptName(rs.getString("dept\_name"));  
 admin.setCanManagePublicAppointment(rs.getBoolean("can\_manage\_public\_appointment"));  
 admin.setCanReportPublicAppointment(rs.getBoolean("can\_report\_public\_appointment"));  
 admins.add(admin);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, stmt, rs);  
 }  
 return admins;  
 }  
   
 public List<Admin> getAdminsByRole(String role) {  
 List<Admin> admins = new ArrayList<>();  
 String sql = "SELECT a.\*, d.dept\_name FROM admin a LEFT JOIN department d ON a.dept\_id = d.id WHERE a.role=? ORDER BY a.id";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setString(1, role);  
 rs = pstmt.executeQuery();  
 while (rs.next()) {  
 Admin admin = new Admin();  
 admin.setId(rs.getInt("id"));  
 admin.setLoginName(rs.getString("login\_name"));  
 admin.setPasswordHash(rs.getString("password\_hash"));  
 admin.setFullName(rs.getString("full\_name"));  
 admin.setDeptId(rs.getInt("dept\_id"));  
 admin.setPhone(rs.getString("phone"));  
 admin.setRole(rs.getString("role"));  
 admin.setPasswordLastChanged(rs.getTimestamp("password\_last\_changed"));  
 admin.setFailedLoginAttempts(rs.getInt("failed\_login\_attempts"));  
 admin.setLockoutUntil(rs.getTimestamp("lockout\_until"));  
 admin.setDeptName(rs.getString("dept\_name"));  
 admin.setCanManagePublicAppointment(rs.getBoolean("can\_manage\_public\_appointment"));  
 admin.setCanReportPublicAppointment(rs.getBoolean("can\_report\_public\_appointment"));  
 admins.add(admin);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 return admins;  
 }  
   
 public boolean updateFailedLoginAttempts(int adminId, int attempts) {  
 String sql = "UPDATE admin SET failed\_login\_attempts=? WHERE id=?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setInt(1, attempts);  
 pstmt.setInt(2, adminId);  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 } finally {  
 close(conn, pstmt);  
 }  
 }  
   
 public boolean setLockoutUntil(int adminId, Timestamp lockoutUntil) {  
 String sql = "UPDATE admin SET lockout\_until=? WHERE id=?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setTimestamp(1, lockoutUntil);  
 pstmt.setInt(2, adminId);  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 } finally {  
 close(conn, pstmt);  
 }  
 }  
   
 public boolean resetFailedLoginAttempts(int adminId) {  
 String sql = "UPDATE admin SET failed\_login\_attempts=0, lockout\_until=NULL WHERE id=?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setInt(1, adminId);  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 } finally {  
 close(conn, pstmt);  
 }  
 }  
}

#### src/main/java/com/zjut/passcode/dao/AppointmentDao.java

package com.zjut.passcode.dao;  
  
import java.sql.Connection;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
import com.zjut.passcode.bean.AccompanyingPerson;  
import com.zjut.passcode.bean.Appointment;  
  
*/\*\*  
 \* 预约相关数据库操作的DAO类。  
 \*/*public class AppointmentDao extends BaseDao {  
 */\*\*  
 \* 新增预约记录。  
 \* @param appointment 预约对象  
 \* @return 新增预约的ID，失败返回-1  
 \*/* public long addAppointment(Appointment appointment) {  
 String sql = "INSERT INTO appointment (visitor\_name, visitor\_id\_card, visitor\_phone, visitor\_unit, campus, entry\_time, transport\_mode, license\_plate, appointment\_type, status, official\_dept\_id, official\_dept\_no, official\_dept\_name, official\_contact\_person, official\_reason) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql, Statement.*RETURN\_GENERATED\_KEYS*);  
 pstmt.setString(1, appointment.getVisitorName());  
 pstmt.setString(2, appointment.getVisitorIdCard());  
 pstmt.setString(3, appointment.getVisitorPhone());  
 pstmt.setString(4, appointment.getVisitorUnit());  
 pstmt.setString(5, appointment.getCampus());  
 pstmt.setTimestamp(6, appointment.getEntryTime());  
 pstmt.setString(7, appointment.getTransportMode());  
 pstmt.setString(8, appointment.getLicensePlate());  
 pstmt.setString(9, appointment.getAppointmentType());  
 pstmt.setString(10, appointment.getStatus());  
 if (appointment.getOfficialDeptId() != null) {  
 pstmt.setInt(11, appointment.getOfficialDeptId());  
 } else {  
 pstmt.setNull(11, java.sql.Types.*INTEGER*);  
 }  
 pstmt.setString(12, appointment.getOfficialDeptNo());  
 pstmt.setString(13, appointment.getOfficialDeptName());  
 pstmt.setString(14, appointment.getOfficialContactPerson());  
 pstmt.setString(15, appointment.getOfficialReason());  
 int affectedRows = pstmt.executeUpdate();  
 if (affectedRows > 0) {  
 rs = pstmt.getGeneratedKeys();  
 if (rs.next()) {  
 long appointmentId = rs.getLong(1);  
 if (appointment.getAccompanyingPersons() != null) {  
 for (AccompanyingPerson person : appointment.getAccompanyingPersons()) {  
 addAccompanyingPerson(appointmentId, person);  
 }  
 }  
 return appointmentId;  
 }  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 return -1;  
 }  
  
 */\*\*  
 \* 新增随行人员记录。  
 \* @param appointmentId 预约ID  
 \* @param person 随行人员对象  
 \* @return 是否添加成功  
 \*/* private boolean addAccompanyingPerson(long appointmentId, AccompanyingPerson person) {  
 String sql = "INSERT INTO accompanying\_person (appointment\_id, full\_name, id\_card, phone) VALUES (?, ?, ?, ?)";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setLong(1, appointmentId);  
 pstmt.setString(2, person.getFullName());  
 pstmt.setString(3, person.getIdCard());  
 pstmt.setString(4, person.getPhone());  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, pstmt);  
 }  
 return false;  
 }  
  
 */\*\*  
 \* 更新预约信息。  
 \* @param appointment 预约对象  
 \* @return 是否更新成功  
 \*/* public boolean updateAppointment(Appointment appointment) {  
 String sql = "UPDATE appointment SET visitor\_name=?, visitor\_id\_card=?, visitor\_phone=?, visitor\_unit=?, campus=?, entry\_time=?, transport\_mode=?, license\_plate=?, appointment\_type=?, status=?, official\_dept\_id=?, official\_contact\_person=?, official\_reason=? WHERE id=?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setString(1, appointment.getVisitorName());  
 pstmt.setString(2, appointment.getVisitorIdCard());  
 pstmt.setString(3, appointment.getVisitorPhone());  
 pstmt.setString(4, appointment.getVisitorUnit());  
 pstmt.setString(5, appointment.getCampus());  
 pstmt.setTimestamp(6, appointment.getEntryTime());  
 pstmt.setString(7, appointment.getTransportMode());  
 pstmt.setString(8, appointment.getLicensePlate());  
 pstmt.setString(9, appointment.getAppointmentType());  
 pstmt.setString(10, appointment.getStatus());  
 pstmt.setObject(11, appointment.getOfficialDeptId());  
 pstmt.setString(12, appointment.getOfficialContactPerson());  
 pstmt.setString(13, appointment.getOfficialReason());  
 pstmt.setLong(14, appointment.getId());  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 } finally {  
 close(conn, pstmt);  
 }  
 }  
  
 */\*\*  
 \* 更新预约状态。  
 \* @param appointmentId 预约ID  
 \* @param status 新状态  
 \* @param auditedBy 审核人ID  
 \* @return 是否更新成功  
 \*/* public boolean updateAppointmentStatus(long appointmentId, String status, int auditedBy) {  
 String sql = "UPDATE appointment SET status = ?, audited\_by = ?, audited\_at = CURRENT\_TIMESTAMP WHERE id = ?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setString(1, status);  
 pstmt.setInt(2, auditedBy);  
 pstmt.setLong(3, appointmentId);  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, pstmt);  
 }  
 return false;  
 }  
  
 */\*\*  
 \* 删除预约。  
 \* @param id 预约ID  
 \* @return 是否删除成功  
 \*/* public boolean deleteAppointment(long id) {  
 String sql = "DELETE FROM appointment WHERE id=?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setLong(1, id);  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 } finally {  
 close(conn, pstmt);  
 }  
 }  
  
 */\*\*  
 \* 根据ID获取预约。  
 \* @param id 预约ID  
 \* @return 预约对象，未找到返回null  
 \*/* public Appointment getAppointmentById(long id) {  
 String sql = "SELECT a.\*, d.dept\_name as official\_dept\_name, adm.full\_name as audited\_by\_name " +  
 "FROM appointment a " +  
 "LEFT JOIN department d ON a.official\_dept\_id = d.id " +  
 "LEFT JOIN admin adm ON a.audited\_by = adm.id " +  
 "WHERE a.id = ?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setLong(1, id);  
 rs = pstmt.executeQuery();  
 if (rs.next()) {  
 return mapResultSetToAppointment(rs);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 return null;  
 }  
  
 */\*\*  
 \* 根据访客信息获取预约列表。  
 \* @param visitorName 访客姓名  
 \* @param encryptedIdCard 加密身份证号  
 \* @param encryptedPhone 加密手机号  
 \* @return 预约列表  
 \*/* public List<Appointment> getAppointmentsByVisitor(String visitorName, String encryptedIdCard, String encryptedPhone) {  
 String sql = "SELECT a.\*, d.dept\_name as official\_dept\_name, adm.full\_name as audited\_by\_name " +  
 "FROM appointment a " +  
 "LEFT JOIN department d ON a.official\_dept\_id = d.id " +  
 "LEFT JOIN admin adm ON a.audited\_by = adm.id " +  
 "WHERE a.visitor\_name = ? AND a.visitor\_id\_card = ? AND a.visitor\_phone = ? " +  
 "ORDER BY a.created\_at DESC";  
 List<Appointment> appointments = new ArrayList<>();  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setString(1, visitorName);  
 pstmt.setString(2, encryptedIdCard);  
 pstmt.setString(3, encryptedPhone);  
 rs = pstmt.executeQuery();  
 while (rs.next()) {  
 appointments.add(mapResultSetToAppointment(rs));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 return appointments;  
 }  
   
 public List<Appointment> getPendingOfficialAppointments(int adminDeptId, String adminRole) {  
 List<Appointment> appointments = new ArrayList<>();  
 StringBuilder sqlBuilder = new StringBuilder();  
 sqlBuilder.append("SELECT a.\*, d.dept\_name as official\_dept\_name, adm.full\_name as audited\_by\_name ");  
 sqlBuilder.append("FROM appointment a LEFT JOIN department d ON a.official\_dept\_id = d.id ");  
 sqlBuilder.append("LEFT JOIN admin adm ON a.audited\_by = adm.id ");  
 sqlBuilder.append("WHERE a.appointment\_type = 'OFFICIAL' AND a.status = 'PENDING' ");  
   
 if ("DEPT\_ADMIN".equals(adminRole)) {  
 sqlBuilder.append("AND a.official\_dept\_id = ? ");  
 }  
   
 sqlBuilder.append("ORDER BY a.created\_at ASC");  
   
 try (Connection conn = getConnection();  
 PreparedStatement pstmt = conn.prepareStatement(sqlBuilder.toString())) {  
 if ("DEPT\_ADMIN".equals(adminRole)) {  
 pstmt.setInt(1, adminDeptId);  
 }  
 ResultSet rs = pstmt.executeQuery();  
 while (rs.next()) {  
 appointments.add(mapResultSetToAppointment(rs));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return appointments;  
 }  
   
 public List<Appointment> getAllAppointments() {  
 String sql = "SELECT a.\*, d.dept\_name as official\_dept\_name, adm.full\_name as audited\_by\_name " +  
 "FROM appointment a " +  
 "LEFT JOIN department d ON a.official\_dept\_id = d.id " +  
 "LEFT JOIN admin adm ON a.audited\_by = adm.id " +  
 "ORDER BY a.created\_at DESC";  
   
 List<Appointment> appointments = new ArrayList<>();  
 try (Connection conn = getConnection();  
 PreparedStatement pstmt = conn.prepareStatement(sql);  
 ResultSet rs = pstmt.executeQuery()) {  
   
 while (rs.next()) {  
 appointments.add(mapResultSetToAppointment(rs));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return appointments;  
 }  
   
 public List<Appointment> getAppointmentsByStatus(String status) {  
 String sql = "SELECT a.\*, d.dept\_name as official\_dept\_name, adm.full\_name as audited\_by\_name " +  
 "FROM appointment a " +  
 "LEFT JOIN department d ON a.official\_dept\_id = d.id " +  
 "LEFT JOIN admin adm ON a.audited\_by = adm.id " +  
 "WHERE a.status = ? " +  
 "ORDER BY a.created\_at DESC";  
   
 List<Appointment> appointments = new ArrayList<>();  
 try (Connection conn = getConnection();  
 PreparedStatement pstmt = conn.prepareStatement(sql)) {  
   
 pstmt.setString(1, status);  
 ResultSet rs = pstmt.executeQuery();  
   
 while (rs.next()) {  
 appointments.add(mapResultSetToAppointment(rs));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return appointments;  
 }  
   
 public List<AccompanyingPerson> getAccompanyingPersons(long appointmentId) {  
 String sql = "SELECT \* FROM accompanying\_person WHERE appointment\_id = ?";  
 List<AccompanyingPerson> persons = new ArrayList<>();  
   
 try (Connection conn = getConnection();  
 PreparedStatement pstmt = conn.prepareStatement(sql)) {  
   
 pstmt.setLong(1, appointmentId);  
 ResultSet rs = pstmt.executeQuery();  
   
 while (rs.next()) {  
 AccompanyingPerson person = new AccompanyingPerson();  
 person.setId(rs.getLong("id"));  
 person.setAppointmentId(rs.getLong("appointment\_id"));  
 person.setFullName(rs.getString("full\_name"));  
 person.setIdCard(rs.getString("id\_card"));  
 person.setPhone(rs.getString("phone"));  
 persons.add(person);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return persons;  
 }  
   
 private Appointment mapResultSetToAppointment(ResultSet rs) throws SQLException {  
 Appointment appointment = new Appointment();  
 appointment.setId(rs.getLong("id"));  
 appointment.setVisitorName(rs.getString("visitor\_name"));  
 appointment.setVisitorIdCard(rs.getString("visitor\_id\_card"));  
 appointment.setVisitorPhone(rs.getString("visitor\_phone"));  
 appointment.setVisitorUnit(rs.getString("visitor\_unit"));  
 appointment.setCampus(rs.getString("campus"));  
 appointment.setEntryTime(rs.getTimestamp("entry\_time"));  
 appointment.setTransportMode(rs.getString("transport\_mode"));  
 appointment.setLicensePlate(rs.getString("license\_plate"));  
 appointment.setAppointmentType(rs.getString("appointment\_type"));  
 appointment.setCreatedAt(rs.getTimestamp("created\_at"));  
 appointment.setStatus(rs.getString("status"));  
   
 int officialDeptId = rs.getInt("official\_dept\_id");  
 if (!rs.wasNull()) {  
 appointment.setOfficialDeptId(officialDeptId);  
 }  
   
 appointment.setOfficialDeptNo(rs.getString("official\_dept\_no"));  
 appointment.setOfficialDeptName(rs.getString("official\_dept\_name"));  
 appointment.setOfficialContactPerson(rs.getString("official\_contact\_person"));  
 appointment.setOfficialReason(rs.getString("official\_reason"));  
   
 int auditedBy = rs.getInt("audited\_by");  
 if (!rs.wasNull()) {  
 appointment.setAuditedBy(auditedBy);  
 }  
   
 appointment.setAuditedAt(rs.getTimestamp("audited\_at"));  
 appointment.setAuditedByName(rs.getString("audited\_by\_name"));  
   
 return appointment;  
 }  
  
 */\*\*  
 \* 按申请月度统计预约次数和人次  
 \*/* public List<Map<String, Object>> getStatsByApplyMonth() {  
 String sql = "SELECT DATE\_FORMAT(created\_at, '%Y-%m') AS month, COUNT(\*) AS count, " +  
 "SUM(1 + (SELECT COUNT(\*) FROM accompanying\_person ap WHERE ap.appointment\_id = a.id)) AS person\_count " +  
 "FROM appointment a GROUP BY month ORDER BY month DESC";  
 List<Map<String, Object>> result = new ArrayList<>();  
 try (Connection conn = getConnection();  
 PreparedStatement pstmt = conn.prepareStatement(sql);  
 ResultSet rs = pstmt.executeQuery()) {  
 while (rs.next()) {  
 Map<String, Object> map = new HashMap<>();  
 map.put("month", rs.getString("month"));  
 map.put("count", rs.getInt("count"));  
 map.put("personCount", rs.getInt("person\_count"));  
 result.add(map);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
  
 */\*\*  
 \* 按预约月度统计预约次数和人次  
 \*/* public List<Map<String, Object>> getStatsByEntryMonth() {  
 String sql = "SELECT DATE\_FORMAT(entry\_time, '%Y-%m') AS month, COUNT(\*) AS count, " +  
 "SUM(1 + (SELECT COUNT(\*) FROM accompanying\_person ap WHERE ap.appointment\_id = a.id)) AS person\_count " +  
 "FROM appointment a GROUP BY month ORDER BY month DESC";  
 List<Map<String, Object>> result = new ArrayList<>();  
 try (Connection conn = getConnection();  
 PreparedStatement pstmt = conn.prepareStatement(sql);  
 ResultSet rs = pstmt.executeQuery()) {  
 while (rs.next()) {  
 Map<String, Object> map = new HashMap<>();  
 map.put("month", rs.getString("month"));  
 map.put("count", rs.getInt("count"));  
 map.put("personCount", rs.getInt("person\_count"));  
 result.add(map);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
  
 */\*\*  
 \* 按校区统计预约次数和人次  
 \*/* public List<Map<String, Object>> getStatsByCampus() {  
 String sql = "SELECT campus, COUNT(\*) AS count, " +  
 "SUM(1 + (SELECT COUNT(\*) FROM accompanying\_person ap WHERE ap.appointment\_id = a.id)) AS person\_count " +  
 "FROM appointment a GROUP BY campus ORDER BY count DESC";  
 List<Map<String, Object>> result = new ArrayList<>();  
 try (Connection conn = getConnection();  
 PreparedStatement pstmt = conn.prepareStatement(sql);  
 ResultSet rs = pstmt.executeQuery()) {  
 while (rs.next()) {  
 Map<String, Object> map = new HashMap<>();  
 map.put("campus", rs.getString("campus"));  
 map.put("count", rs.getInt("count"));  
 map.put("personCount", rs.getInt("person\_count"));  
 result.add(map);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
  
 */\*\*  
 \* 按公务访问部门统计预约次数和人次  
 \*/* public List<Map<String, Object>> getStatsByOfficialDept() {  
 String sql = "SELECT d.dept\_name, COUNT(\*) AS count, " +  
 "SUM(1 + (SELECT COUNT(\*) FROM accompanying\_person ap WHERE ap.appointment\_id = a.id)) AS person\_count " +  
 "FROM appointment a LEFT JOIN department d ON a.official\_dept\_id = d.id " +  
 "WHERE a.appointment\_type = 'OFFICIAL' GROUP BY d.dept\_name ORDER BY count DESC";  
 List<Map<String, Object>> result = new ArrayList<>();  
 try (Connection conn = getConnection();  
 PreparedStatement pstmt = conn.prepareStatement(sql);  
 ResultSet rs = pstmt.executeQuery()) {  
 while (rs.next()) {  
 Map<String, Object> map = new HashMap<>();  
 map.put("deptName", rs.getString("dept\_name"));  
 map.put("count", rs.getInt("count"));  
 map.put("personCount", rs.getInt("person\_count"));  
 result.add(map);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
  
 // 按申请月度统计（指定部门和类型）  
 public List<Map<String, Object>> getStatsByApplyMonthForDept(int deptId, String type) {  
 String sql = "SELECT DATE\_FORMAT(created\_at, '%Y-%m') AS month, COUNT(\*) AS count, " +  
 "SUM(1 + (SELECT COUNT(\*) FROM accompanying\_person ap WHERE ap.appointment\_id = a.id)) AS person\_count " +  
 "FROM appointment a WHERE a.official\_dept\_id = ? AND a.appointment\_type = ? GROUP BY month ORDER BY month DESC";  
 List<Map<String, Object>> result = new ArrayList<>();  
 try (Connection conn = getConnection();  
 PreparedStatement pstmt = conn.prepareStatement(sql)) {  
 pstmt.setInt(1, deptId);  
 pstmt.setString(2, type);  
 ResultSet rs = pstmt.executeQuery();  
 while (rs.next()) {  
 Map<String, Object> map = new HashMap<>();  
 map.put("month", rs.getString("month"));  
 map.put("count", rs.getInt("count"));  
 map.put("personCount", rs.getInt("person\_count"));  
 result.add(map);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
 // 按预约月度统计（指定部门和类型）  
 public List<Map<String, Object>> getStatsByEntryMonthForDept(int deptId, String type) {  
 String sql = "SELECT DATE\_FORMAT(entry\_time, '%Y-%m') AS month, COUNT(\*) AS count, " +  
 "SUM(1 + (SELECT COUNT(\*) FROM accompanying\_person ap WHERE ap.appointment\_id = a.id)) AS person\_count " +  
 "FROM appointment a WHERE a.official\_dept\_id = ? AND a.appointment\_type = ? GROUP BY month ORDER BY month DESC";  
 List<Map<String, Object>> result = new ArrayList<>();  
 try (Connection conn = getConnection();  
 PreparedStatement pstmt = conn.prepareStatement(sql)) {  
 pstmt.setInt(1, deptId);  
 pstmt.setString(2, type);  
 ResultSet rs = pstmt.executeQuery();  
 while (rs.next()) {  
 Map<String, Object> map = new HashMap<>();  
 map.put("month", rs.getString("month"));  
 map.put("count", rs.getInt("count"));  
 map.put("personCount", rs.getInt("person\_count"));  
 result.add(map);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
 // 按校区统计（指定部门和类型）  
 public List<Map<String, Object>> getStatsByCampusForDept(int deptId, String type) {  
 String sql = "SELECT campus, COUNT(\*) AS count, " +  
 "SUM(1 + (SELECT COUNT(\*) FROM accompanying\_person ap WHERE ap.appointment\_id = a.id)) AS person\_count " +  
 "FROM appointment a WHERE a.official\_dept\_id = ? AND a.appointment\_type = ? GROUP BY campus ORDER BY count DESC";  
 List<Map<String, Object>> result = new ArrayList<>();  
 try (Connection conn = getConnection();  
 PreparedStatement pstmt = conn.prepareStatement(sql)) {  
 pstmt.setInt(1, deptId);  
 pstmt.setString(2, type);  
 ResultSet rs = pstmt.executeQuery();  
 while (rs.next()) {  
 Map<String, Object> map = new HashMap<>();  
 map.put("campus", rs.getString("campus"));  
 map.put("count", rs.getInt("count"));  
 map.put("personCount", rs.getInt("person\_count"));  
 result.add(map);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
 // 按公务访问部门统计（指定部门）  
 public List<Map<String, Object>> getStatsByOfficialDeptForDept(int deptId) {  
 String sql = "SELECT d.dept\_name, COUNT(\*) AS count, " +  
 "SUM(1 + (SELECT COUNT(\*) FROM accompanying\_person ap WHERE ap.appointment\_id = a.id)) AS person\_count " +  
 "FROM appointment a LEFT JOIN department d ON a.official\_dept\_id = d.id " +  
 "WHERE a.official\_dept\_id = ? AND a.appointment\_type = 'OFFICIAL' GROUP BY d.dept\_name ORDER BY count DESC";  
 List<Map<String, Object>> result = new ArrayList<>();  
 try (Connection conn = getConnection();  
 PreparedStatement pstmt = conn.prepareStatement(sql)) {  
 pstmt.setInt(1, deptId);  
 ResultSet rs = pstmt.executeQuery();  
 while (rs.next()) {  
 Map<String, Object> map = new HashMap<>();  
 map.put("deptName", rs.getString("dept\_name"));  
 map.put("count", rs.getInt("count"));  
 map.put("personCount", rs.getInt("person\_count"));  
 result.add(map);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return result;  
 }  
  
 */\*\*  
 \* 根据部门ID获取该部门的所有预约（包括公务预约和社会公众预约）  
 \* @param deptId 部门ID  
 \* @return 预约列表  
 \*/* public List<Appointment> getAppointmentsByDept(int deptId) {  
 String sql = "SELECT a.\*, d.dept\_name as official\_dept\_name, adm.full\_name as audited\_by\_name " +  
 "FROM appointment a " +  
 "LEFT JOIN department d ON a.official\_dept\_id = d.id " +  
 "LEFT JOIN admin adm ON a.audited\_by = adm.id " +  
 "WHERE a.official\_dept\_id = ? OR a.appointment\_type = 'PUBLIC' " +  
 "ORDER BY a.created\_at DESC";  
   
 List<Appointment> appointments = new ArrayList<>();  
 try (Connection conn = getConnection();  
 PreparedStatement pstmt = conn.prepareStatement(sql)) {  
   
 pstmt.setInt(1, deptId);  
 ResultSet rs = pstmt.executeQuery();  
   
 while (rs.next()) {  
 appointments.add(mapResultSetToAppointment(rs));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return appointments;  
 }  
  
 */\*\*  
 \* 根据部门ID和预约类型获取预约列表  
 \* @param deptId 部门ID  
 \* @param appointmentType 预约类型  
 \* @return 预约列表  
 \*/* public List<Appointment> getAppointmentsByDeptAndType(int deptId, String appointmentType) {  
 String sql = "SELECT a.\*, d.dept\_name as official\_dept\_name, adm.full\_name as audited\_by\_name " +  
 "FROM appointment a " +  
 "LEFT JOIN department d ON a.official\_dept\_id = d.id " +  
 "LEFT JOIN admin adm ON a.audited\_by = adm.id " +  
 "WHERE a.official\_dept\_id = ? AND a.appointment\_type = ? " +  
 "ORDER BY a.created\_at DESC";  
   
 List<Appointment> appointments = new ArrayList<>();  
 try (Connection conn = getConnection();  
 PreparedStatement pstmt = conn.prepareStatement(sql)) {  
   
 pstmt.setInt(1, deptId);  
 pstmt.setString(2, appointmentType);  
 ResultSet rs = pstmt.executeQuery();  
   
 while (rs.next()) {  
 appointments.add(mapResultSetToAppointment(rs));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return appointments;  
 }  
}

#### src/main/java/com/zjut/passcode/dao/AuditLogDao.java

package com.zjut.passcode.dao;  
  
import java.sql.Connection;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.util.ArrayList;  
import java.util.List;  
  
import com.zjut.passcode.bean.AuditLog;  
  
public class AuditLogDao extends BaseDao {  
 public boolean addAuditLog(AuditLog auditLog) {  
 String sql = "INSERT INTO audit\_log (admin\_id, admin\_name, action, details, ip\_address, hmac\_signature) VALUES (?, ?, ?, ?, ?, ?)";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setInt(1, auditLog.getAdminId());  
 pstmt.setString(2, auditLog.getAdminName());  
 pstmt.setString(3, auditLog.getAction());  
 pstmt.setString(4, auditLog.getDetails());  
 pstmt.setString(5, auditLog.getIpAddress());  
 pstmt.setString(6, auditLog.getHmacSignature());  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 } finally {  
 close(conn, pstmt);  
 }  
 }  
   
 public List<AuditLog> searchAuditLogs(String adminName, String action, String startDate, String endDate) {  
 List<AuditLog> logs = new ArrayList<>();  
 StringBuilder sqlBuilder = new StringBuilder();  
 sqlBuilder.append("SELECT \* FROM audit\_log WHERE 1=1 ");  
   
 List<Object> params = new ArrayList<>();  
   
 if (adminName != null && !adminName.trim().isEmpty()) {  
 sqlBuilder.append("AND admin\_name LIKE ? ");  
 params.add("%" + adminName + "%");  
 }  
 if (action != null && !action.trim().isEmpty()) {  
 sqlBuilder.append("AND action LIKE ? ");  
 params.add("%" + action + "%");  
 }  
 if (startDate != null && !startDate.trim().isEmpty()) {  
 sqlBuilder.append("AND DATE(created\_at) >= ? ");  
 params.add(startDate);  
 }  
 if (endDate != null && !endDate.trim().isEmpty()) {  
 sqlBuilder.append("AND DATE(created\_at) <= ? ");  
 params.add(endDate);  
 }  
   
 sqlBuilder.append("ORDER BY created\_at DESC");  
   
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sqlBuilder.toString());  
 for (int i = 0; i < params.size(); i++) {  
 pstmt.setObject(i + 1, params.get(i));  
 }  
 rs = pstmt.executeQuery();  
 while (rs.next()) {  
 logs.add(mapResultSetToAuditLog(rs));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 return logs;  
 }  
   
 public List<AuditLog> getRecentAuditLogs(int limit) {  
 List<AuditLog> logs = new ArrayList<>();  
 String sql = "SELECT \* FROM audit\_log ORDER BY created\_at DESC LIMIT ?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setInt(1, limit);  
 rs = pstmt.executeQuery();  
 while (rs.next()) {  
 logs.add(mapResultSetToAuditLog(rs));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 return logs;  
 }  
   
 public List<AuditLog> getAllAuditLogs() {  
 List<AuditLog> logs = new ArrayList<>();  
 String sql = "SELECT \* FROM audit\_log ORDER BY created\_at DESC";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 rs = pstmt.executeQuery();  
 while (rs.next()) {  
 logs.add(mapResultSetToAuditLog(rs));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 return logs;  
 }  
   
 public boolean deleteOldLogs(int daysToKeep) {  
 String sql = "DELETE FROM audit\_log WHERE created\_at < DATE\_SUB(NOW(), INTERVAL ? DAY)";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setInt(1, daysToKeep);  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 } finally {  
 close(conn, pstmt);  
 }  
 }  
   
 public List<AuditLog> getAuditLogsByPage(int page, int pageSize) {  
 List<AuditLog> logs = new ArrayList<>();  
 String sql = "SELECT \* FROM audit\_log ORDER BY created\_at DESC LIMIT ? OFFSET ?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setInt(1, pageSize);  
 pstmt.setInt(2, (page - 1) \* pageSize);  
 rs = pstmt.executeQuery();  
 while (rs.next()) {  
 logs.add(mapResultSetToAuditLog(rs));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 return logs;  
 }  
  
 public int getAuditLogsCount() {  
 String sql = "SELECT COUNT(\*) FROM audit\_log";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 rs = pstmt.executeQuery();  
 if (rs.next()) {  
 return rs.getInt(1);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 return 0;  
 }  
   
 private AuditLog mapResultSetToAuditLog(ResultSet rs) throws SQLException {  
 AuditLog log = new AuditLog();  
 log.setId(rs.getLong("id"));  
 log.setAdminId(rs.getInt("admin\_id"));  
 log.setAdminName(rs.getString("admin\_name"));  
 log.setAction(rs.getString("action"));  
 log.setDetails(rs.getString("details"));  
 log.setIpAddress(rs.getString("ip\_address"));  
 log.setCreatedAt(rs.getTimestamp("created\_at"));  
 log.setHmacSignature(rs.getString("hmac\_signature"));  
 return log;  
 }  
}

#### src/main/java/com/zjut/passcode/dao/BaseDao.java

package com.zjut.passcode.dao;  
  
import java.io.IOException;  
import java.io.InputStream;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.util.Properties;  
  
public class BaseDao {  
 private static String *DB\_URL*;  
 private static String *DB\_USER*;  
 private static String *DB\_PASSWORD*;  
 private static String *DB\_DRIVER*;  
  
 static {  
 *loadDatabaseConfig*();  
 try {  
 Class.*forName*(*DB\_DRIVER*);  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 */\*\*  
 \* 加载数据库配置文件  
 \*/* private static void loadDatabaseConfig() {  
 Properties props = new Properties();  
 InputStream input = null;  
   
 try {  
 // 方法1: 尝试从classpath加载（WEB-INF/classes目录）  
 input = BaseDao.class.getClassLoader().getResourceAsStream("db.properties");  
 if (input != null) {  
 System.*out*.println("成功从classpath加载配置文件");  
 }  
   
 // 方法2: 如果classpath没有，尝试从当前工作目录加载  
 if (input == null) {  
 try {  
 input = new java.io.FileInputStream("db.properties");  
 System.*out*.println("成功从当前工作目录加载配置文件");  
 } catch (IOException e) {  
 // 继续尝试其他方法  
 }  
 }  
   
 // 方法3: 尝试从上级目录加载  
 if (input == null) {  
 try {  
 input = new java.io.FileInputStream("../db.properties");  
 System.*out*.println("成功从上级目录加载配置文件");  
 } catch (IOException e) {  
 // 继续尝试其他方法  
 }  
 }  
   
 if (input == null) {  
 throw new RuntimeException("无法找到数据库配置文件 db.properties");  
 }  
   
 props.load(input);  
   
 *DB\_URL* = props.getProperty("db.url");  
 *DB\_USER* = props.getProperty("db.username");  
 *DB\_PASSWORD* = props.getProperty("db.password");  
 *DB\_DRIVER* = props.getProperty("db.driver");  
   
 // 验证必要的配置项  
 if (*DB\_URL* == null || *DB\_USER* == null || *DB\_PASSWORD* == null || *DB\_DRIVER* == null) {  
 throw new RuntimeException("数据库配置文件缺少必要的配置项");  
 }  
   
 System.*out*.println("数据库配置加载成功");  
   
 } catch (IOException e) {  
 e.printStackTrace();  
 throw new RuntimeException("无法加载数据库配置文件: " + e.getMessage());  
 } finally {  
 if (input != null) {  
 try {  
 input.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
  
 public Connection getConnection() throws SQLException {  
 return DriverManager.*getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 }  
  
 public void close(Connection conn, Statement stmt, ResultSet rs) {  
 try { if (rs != null) rs.close(); } catch (Exception e) { e.printStackTrace(); }  
 try { if (stmt != null) stmt.close(); } catch (Exception e) { e.printStackTrace(); }  
 try { if (conn != null) conn.close(); } catch (Exception e) { e.printStackTrace(); }  
 }  
  
 public void close(Connection conn, Statement stmt) {  
 close(conn, stmt, null);  
 }  
  
 public void close(Connection conn) {  
 close(conn, null, null);  
 }  
}

#### src/main/java/com/zjut/passcode/dao/DepartmentDao.java

package com.zjut.passcode.dao;  
  
import java.sql.Connection;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.util.ArrayList;  
import java.util.List;  
  
import com.zjut.passcode.bean.Department;  
  
public class DepartmentDao extends BaseDao {  
 public boolean addDepartment(Department department) {  
 String sql = "INSERT INTO department (dept\_no, dept\_name, dept\_type) VALUES (?, ?, ?)";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setString(1, department.getDeptNo());  
 pstmt.setString(2, department.getDeptName());  
 pstmt.setString(3, department.getDeptType());  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 } finally {  
 close(conn, pstmt);  
 }  
 }  
   
 public boolean updateDepartment(Department department) {  
 String sql = "UPDATE department SET dept\_no=?, dept\_name=?, dept\_type=? WHERE id=?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setString(1, department.getDeptNo());  
 pstmt.setString(2, department.getDeptName());  
 pstmt.setString(3, department.getDeptType());  
 pstmt.setInt(4, department.getId());  
   
 // Debug: Print update parameters  
 System.*out*.println("DEBUG: Updating department:");  
 System.*out*.println(" ID: " + department.getId());  
 System.*out*.println(" deptNo: " + department.getDeptNo());  
 System.*out*.println(" deptName: " + department.getDeptName());  
 System.*out*.println(" deptType: " + department.getDeptType());  
   
 int result = pstmt.executeUpdate();  
 System.*out*.println("DEBUG: Update result: " + result + " rows affected");  
 return result > 0;  
 } catch (SQLException e) {  
 System.*out*.println("ERROR: Failed to update department in database");  
 e.printStackTrace();  
 return false;  
 } finally {  
 close(conn, pstmt);  
 }  
 }  
   
 public boolean deleteDepartment(int id) {  
 String sql = "DELETE FROM department WHERE id=?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setInt(1, id);  
 return pstmt.executeUpdate() > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 } finally {  
 close(conn, pstmt);  
 }  
 }  
   
 public Department getDepartmentById(int id) {  
 String sql = "SELECT \* FROM department WHERE id=?";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setInt(1, id);  
 rs = pstmt.executeQuery();  
 if (rs.next()) {  
 Department dept = new Department();  
 dept.setId(rs.getInt("id"));  
 dept.setDeptNo(rs.getString("dept\_no"));  
 dept.setDeptName(rs.getString("dept\_name"));  
 dept.setDeptType(rs.getString("dept\_type"));  
 return dept;  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 return null;  
 }  
   
 public List<Department> getAllDepartments() {  
 List<Department> departments = new ArrayList<>();  
 String sql = "SELECT \* FROM department ORDER BY dept\_no";  
 Connection conn = null;  
 Statement stmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 stmt = conn.createStatement();  
 rs = stmt.executeQuery(sql);  
 while (rs.next()) {  
 Department dept = new Department();  
 dept.setId(rs.getInt("id"));  
 dept.setDeptNo(rs.getString("dept\_no"));  
 dept.setDeptName(rs.getString("dept\_name"));  
 dept.setDeptType(rs.getString("dept\_type"));  
 departments.add(dept);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, stmt, rs);  
 }  
 return departments;  
 }  
   
 public List<Department> getDepartmentsByType(String deptType) {  
 List<Department> departments = new ArrayList<>();  
 String sql = "SELECT \* FROM department WHERE dept\_type=? ORDER BY dept\_no";  
 Connection conn = null;  
 PreparedStatement pstmt = null;  
 ResultSet rs = null;  
 try {  
 conn = getConnection();  
 pstmt = conn.prepareStatement(sql);  
 pstmt.setString(1, deptType);  
 rs = pstmt.executeQuery();  
 while (rs.next()) {  
 Department dept = new Department();  
 dept.setId(rs.getInt("id"));  
 dept.setDeptNo(rs.getString("dept\_no"));  
 dept.setDeptName(rs.getString("dept\_name"));  
 dept.setDeptType(rs.getString("dept\_type"));  
 departments.add(dept);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 close(conn, pstmt, rs);  
 }  
 return departments;  
 }  
}

#### src/main/java/com/zjut/passcode/util/CryptoUtil.java

package com.zjut.passcode.util;  
  
import java.util.Base64;  
  
import javax.crypto.KeyGenerator;  
import javax.crypto.SecretKey;  
  
import org.bouncycastle.crypto.digests.SM3Digest;  
import org.bouncycastle.crypto.engines.SM4Engine;  
import org.bouncycastle.crypto.paddings.PKCS7Padding;  
import org.bouncycastle.crypto.paddings.PaddedBufferedBlockCipher;  
import org.bouncycastle.crypto.params.KeyParameter;  
  
public class CryptoUtil {  
   
 */\*\*  
 \* 对输入字符串进行SM3哈希运算。  
 \* @param input 输入字符串  
 \* @return 哈希值的十六进制字符串  
 \*/* public static String sm3Hash(String input) {  
 try {  
 System.*out*.println("INFO: Hashing input: " + input);  
 SM3Digest digest = new SM3Digest();  
 byte[] inputBytes = input.getBytes("UTF-8");  
 digest.update(inputBytes, 0, inputBytes.length);  
 byte[] hash = new byte[digest.getDigestSize()];  
 digest.doFinal(hash, 0);  
 String hashResult = *bytesToHex*(hash);  
 System.*out*.println("INFO: Hash result: " + hashResult);  
 return hashResult;  
 } catch (Exception e) {  
 System.*out*.println("ERROR: Password hashing failed");  
 System.*out*.println("ERROR Details: " + e.getMessage());  
 e.printStackTrace();  
 return null;  
 }  
 }  
   
 */\*\*  
 \* 获取SM4加密算法所需的16字节密钥。  
 \* @param key 密钥字符串  
 \* @return 16字节密钥数组  
 \* @throws Exception 如果密钥长度不为16字节则抛出异常  
 \*/* private static byte[] getSm4KeyBytes(String key) throws Exception {  
 byte[] keyBytes = key.getBytes("UTF-8");  
 if (keyBytes.length != 16) {  
 throw new IllegalArgumentException("SM4 key must be 16 bytes (128 bits), but got " + keyBytes.length + " bytes");  
 }  
 return keyBytes;  
 }  
   
 */\*\*  
 \* 使用SM4算法对明文进行加密（ECB/PKCS7Padding）。  
 \* @param plaintext 明文  
 \* @param key 16字节密钥  
 \* @return 加密后的Base64字符串  
 \*/* public static String sm4Encrypt(String plaintext, String key) {  
 try {  
 byte[] keyBytes = *getSm4KeyBytes*(key);  
 byte[] input = plaintext.getBytes("UTF-8");  
 PaddedBufferedBlockCipher cipher = new PaddedBufferedBlockCipher(new SM4Engine(), new PKCS7Padding());  
 cipher.init(true, new KeyParameter(keyBytes));  
 byte[] output = new byte[cipher.getOutputSize(input.length)];  
 int len = cipher.processBytes(input, 0, input.length, output, 0);  
 len += cipher.doFinal(output, len);  
 byte[] encrypted = new byte[len];  
 System.*arraycopy*(output, 0, encrypted, 0, len);  
 return Base64.*getEncoder*().encodeToString(encrypted);  
 } catch (Exception e) {  
 e.printStackTrace();  
 return null;  
 }  
 }  
   
 */\*\*  
 \* 使用SM4算法对密文进行解密（ECB/PKCS7Padding）。  
 \* @param encryptedText Base64编码的密文  
 \* @param key 16字节密钥  
 \* @return 解密后的明文字符串  
 \*/* public static String sm4Decrypt(String encryptedText, String key) {  
 try {  
 byte[] keyBytes = *getSm4KeyBytes*(key);  
 byte[] input = Base64.*getDecoder*().decode(encryptedText);  
 PaddedBufferedBlockCipher cipher = new PaddedBufferedBlockCipher(new SM4Engine(), new PKCS7Padding());  
 cipher.init(false, new KeyParameter(keyBytes));  
 byte[] output = new byte[cipher.getOutputSize(input.length)];  
 int len = cipher.processBytes(input, 0, input.length, output, 0);  
 len += cipher.doFinal(output, len);  
 byte[] decrypted = new byte[len];  
 System.*arraycopy*(output, 0, decrypted, 0, len);  
 return new String(decrypted, "UTF-8");  
 } catch (Exception e) {  
 e.printStackTrace();  
 return null;  
 }  
 }  
   
 */\*\*  
 \* 生成HMAC-SM3签名。  
 \* @param data 待签名数据  
 \* @return HMAC-SM3签名字符串  
 \*/* public static String generateHmacSm3(String data) {  
 try {  
 String key = "campus\_pass\_secret\_key\_2024";  
 String combined = key + data;  
 return *sm3Hash*(combined);  
 } catch (Exception e) {  
 e.printStackTrace();  
 return null;  
 }  
 }  
   
 */\*\*  
 \* 生成随机128位密钥（Base64编码）。  
 \* @return Base64编码的密钥字符串  
 \*/* public static String generateKey() {  
 try {  
 KeyGenerator keyGen = KeyGenerator.*getInstance*("AES");  
 keyGen.init(128);  
 SecretKey secretKey = keyGen.generateKey();  
 return Base64.*getEncoder*().encodeToString(secretKey.getEncoded());  
 } catch (Exception e) {  
 e.printStackTrace();  
 return null;  
 }  
 }  
   
 */\*\*  
 \* 对姓名进行脱敏处理。  
 \* @param name 姓名  
 \* @return 脱敏后的姓名  
 \*/* public static String maskName(String name) {  
 if (name == null || name.length() <= 1) {  
 return name;  
 }  
 if (name.length() == 2) {  
 return name.charAt(0) + "\*" + name.charAt(1);  
 } else {  
 StringBuilder masked = new StringBuilder();  
 masked.append(name.charAt(0));  
 for (int i = 1; i < name.length() - 1; i++) {  
 masked.append("\*");  
 }  
 masked.append(name.charAt(name.length() - 1));  
 return masked.toString();  
 }  
 }  
   
 */\*\*  
 \* 对身份证号进行脱敏处理。  
 \* @param idCard 身份证号  
 \* @return 脱敏后的身份证号  
 \*/* public static String maskIdCard(String idCard) {  
 if (idCard == null || idCard.length() < 8) {  
 return idCard;  
 }  
 return idCard.substring(0, 6) + "\*\*\*\*\*\*\*\*" + idCard.substring(idCard.length() - 4);  
 }  
   
 */\*\*  
 \* 对手机号进行脱敏处理。  
 \* @param phone 手机号  
 \* @return 脱敏后的手机号  
 \*/* public static String maskPhone(String phone) {  
 if (phone == null || phone.length() < 7) {  
 return phone;  
 }  
 return phone.substring(0, 3) + "\*\*\*\*" + phone.substring(phone.length() - 4);  
 }  
   
 */\*\*  
 \* 字节数组转为十六进制字符串。  
 \* @param bytes 字节数组  
 \* @return 十六进制字符串  
 \*/* private static String bytesToHex(byte[] bytes) {  
 StringBuilder result = new StringBuilder();  
 for (byte b : bytes) {  
 result.append(String.*format*("%02x", b));  
 }  
 return result.toString();  
 }  
   
 */\*\*  
 \* 校验密码复杂度，要求包含数字、大小写字母和特殊字符，且长度不少于8位。  
 \* @param password 密码  
 \* @return 若符合复杂度要求返回true，否则返回false  
 \*/* public static boolean validatePasswordComplexity(String password) {  
 if (password == null || password.length() < 8) {  
 return false;  
 }  
   
 boolean hasDigit = false;  
 boolean hasLower = false;  
 boolean hasUpper = false;  
 boolean hasSpecial = false;  
   
 for (char c : password.toCharArray()) {  
 if (Character.*isDigit*(c)) {  
 hasDigit = true;  
 } else if (Character.*isLowerCase*(c)) {  
 hasLower = true;  
 } else if (Character.*isUpperCase*(c)) {  
 hasUpper = true;  
 } else {  
 hasSpecial = true;  
 }  
 }  
   
 return hasDigit && hasLower && hasUpper && hasSpecial;  
 }  
}

#### src/main/webapp/admin/add\_department.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **page** import="com.zjut.passcode.bean.Admin" %>  
<%  
 // Check if admin is logged in  
 Admin admin = (Admin) session.getAttribute("admin");  
 if (admin == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
%>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>添加部门 - 校园通行码系统</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
   
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 }  
   
 .header {  
 background: white;  
 box-shadow: 0 2px 10px rgba(0,0,0,0.1);  
 padding: 20px;  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 }  
   
 .logo {  
 font-size: 1.5em;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .user-info {  
 display: flex;  
 align-items: center;  
 gap: 20px;  
 }  
   
 .user-name {  
 color: #333;  
 font-weight: 500;  
 }  
   
 .back-btn {  
 background: #6c757d;  
 color: white;  
 padding: 8px 16px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 text-decoration: none;  
 font-size: 0.9em;  
 }  
   
 .back-btn:hover {  
 background: #5a6268;  
 }  
   
 .container {  
 max-width: 800px;  
 margin: 40px auto;  
 padding: 0 20px;  
 }  
   
 .page-title {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 margin-bottom: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 text-align: center;  
 }  
   
 .title {  
 font-size: 2em;  
 color: #333;  
 margin-bottom: 10px;  
 }  
   
 .subtitle {  
 color: #666;  
 font-size: 1.1em;  
 }  
   
 .form-content {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 }  
   
 .form-group {  
 margin-bottom: 20px;  
 }  
   
 .form-label {  
 display: block;  
 margin-bottom: 8px;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .form-input {  
 width: 100%;  
 padding: 12px;  
 border: 2px solid #e9ecef;  
 border-radius: 8px;  
 font-size: 1em;  
 transition: border-color 0.3s;  
 }  
   
 .form-input:focus {  
 outline: none;  
 border-color: #667eea;  
 }  
   
 .form-select {  
 width: 100%;  
 padding: 12px;  
 border: 2px solid #e9ecef;  
 border-radius: 8px;  
 font-size: 1em;  
 background: white;  
 cursor: pointer;  
 }  
   
 .form-select:focus {  
 outline: none;  
 border-color: #667eea;  
 }  
   
 .btn-group {  
 display: flex;  
 gap: 15px;  
 margin-top: 30px;  
 }  
   
 .btn {  
 padding: 12px 24px;  
 border: none;  
 border-radius: 8px;  
 cursor: pointer;  
 font-size: 1em;  
 text-decoration: none;  
 display: inline-block;  
 text-align: center;  
 transition: background-color 0.3s;  
 }  
   
 .btn-primary {  
 background: #28a745;  
 color: white;  
 }  
   
 .btn-primary:hover {  
 background: #218838;  
 }  
   
 .btn-secondary {  
 background: #6c757d;  
 color: white;  
 }  
   
 .btn-secondary:hover {  
 background: #5a6268;  
 }  
   
 .error-message {  
 background: #f8d7da;  
 color: #721c24;  
 padding: 12px;  
 border-radius: 8px;  
 margin-bottom: 20px;  
 border: 1px solid #f5c6cb;  
 }  
   
 .success-message {  
 background: #d4edda;  
 color: #155724;  
 padding: 12px;  
 border-radius: 8px;  
 margin-bottom: 20px;  
 border: 1px solid #c3e6cb;  
 }  
   
 @media (max-width: 768px) {  
 .header {  
 flex-direction: column;  
 gap: 15px;  
 }  
   
 .user-info {  
 flex-direction: column;  
 gap: 10px;  
 }  
   
 .title {  
 font-size: 1.5em;  
 }  
   
 .btn-group {  
 flex-direction: column;  
 }  
 }  
 </style>  
</head>  
<body>  
 <div class="header">  
 <div class="logo">🏢 部门管理系统</div>  
 <div class="user-info">  
 <div class="user-name">  
 管理员：<%= admin.getFullName() %>  
 </div>  
 <a href="${pageContext.request.contextPath}/admin/departments" class="back-btn">🔙 返回部门管理</a>  
 </div>  
 </div>  
   
 <div class="container">  
 <div class="page-title">  
 <div class="title">添加新部门</div>  
 <div class="subtitle">创建新的部门信息</div>  
 </div>  
   
 <div class="form-content">  
 <% if (request.getAttribute("error") != null) { %>  
 <div class="error-message">  
 <%= request.getAttribute("error") %>  
 </div>  
 <% } %>  
   
 <% if (request.getAttribute("success") != null) { %>  
 <div class="success-message">  
 <%= request.getAttribute("success") %>  
 </div>  
 <% } %>  
   
 <form action="${pageContext.request.contextPath}/admin/department/add" method="post">  
 <div class="form-group">  
 <label for="deptNo" class="form-label">部门编号 \*</label>  
 <input type="text" id="deptNo" name="deptNo" class="form-input"   
 value="${param.deptNo}" required   
 placeholder="请输入部门编号，如：D001">  
 </div>  
   
 <div class="form-group">  
 <label for="deptName" class="form-label">部门名称 \*</label>  
 <input type="text" id="deptName" name="deptName" class="form-input"   
 value="${param.deptName}" required   
 placeholder="请输入部门名称">  
 </div>  
   
 <div class="form-group">  
 <label for="deptType" class="form-label">部门类型 \*</label>  
 <select id="deptType" name="deptType" class="form-select" required>  
 <option value="">请选择部门类型</option>  
 <option value="行政部门" ${param.deptType == '行政部门' ? 'selected' : ''}>行政部门</option>  
 <option value="直属部门" ${param.deptType == '直属部门' ? 'selected' : ''}>直属部门</option>  
 <option value="学院" ${param.deptType == '学院' ? 'selected' : ''}>学院</option>  
 </select>  
 </div>  
   
 <div class="btn-group">  
 <button type="submit" class="btn btn-primary">✅ 添加部门</button>  
 <a href="${pageContext.request.contextPath}/admin/departments" class="btn btn-secondary">❌ 取消</a>  
 </div>  
 </form>  
 </div>  
 </div>  
</body>  
</html>

#### src/main/webapp/admin/admin\_manage.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **page** import="com.zjut.passcode.bean.Admin" %>  
<%@ **page** import="com.zjut.passcode.bean.Department" %>  
<%@ **page** import="java.util.List" %>  
<%  
 Admin currentAdmin = (Admin) session.getAttribute("admin");  
 if (currentAdmin == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
 List<Admin> admins = (List<Admin>) request.getAttribute("admins");  
 List<Department> departments = (List<Department>) request.getAttribute("departments");  
 String action = request.getParameter("action");  
 Admin editAdmin = (Admin) request.getAttribute("editAdmin");  
%>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>管理员管理 - 校园通行码系统</title>  
 <style>  
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #e0eafc 0%, #cfdef3 100%);  
 min-height: 100vh;  
 margin: 0;  
 }  
 .header {  
 background: #fff;  
 box-shadow: 0 2px 10px rgba(0,0,0,0.07);  
 padding: 24px 40px;  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 }  
 .logo {  
 font-size: 1.7em;  
 font-weight: bold;  
 color: #3a3a3a;  
 }  
 .user-info {  
 color: #555;  
 font-size: 1em;  
 }  
 .container {  
 max-width: 1100px;  
 margin: 40px auto;  
 background: #fff;  
 border-radius: 18px;  
 box-shadow: 0 8px 32px rgba(0,0,0,0.08);  
 padding: 40px 32px 32px 32px;  
 }  
 .page-title {  
 font-size: 2.1em;  
 color: #333;  
 margin-bottom: 10px;  
 text-align: center;  
 }  
 .subtitle {  
 color: #888;  
 font-size: 1.1em;  
 text-align: center;  
 margin-bottom: 30px;  
 }  
 .add-btn {  
 background: linear-gradient(90deg, #667eea 0%, #764ba2 100%);  
 color: #fff;  
 border: none;  
 border-radius: 6px;  
 padding: 10px 28px;  
 font-size: 1em;  
 cursor: pointer;  
 margin-bottom: 24px;  
 float: right;  
 }  
 .add-btn:hover {  
 background: linear-gradient(90deg, #5a67d8 0%, #6b47b6 100%);  
 }  
 .admin-table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 10px;  
 }  
 .admin-table th, .admin-table td {  
 padding: 13px 10px;  
 border-bottom: 1px solid #e9ecef;  
 text-align: left;  
 }  
 .admin-table th {  
 background: #f6f8fa;  
 color: #333;  
 font-weight: bold;  
 }  
 .admin-table tr:hover {  
 background: #f3f6fa;  
 }  
 .role-badge {  
 padding: 4px 10px;  
 border-radius: 12px;  
 font-size: 0.9em;  
 font-weight: bold;  
 color: #fff;  
 }  
 .role-system-admin { background: #dc3545; }  
 .role-school-admin { background: #fd7e14; }  
 .role-dept-admin { background: #ffc107; color: #212529; }  
 .role-audit-admin { background: #17a2b8; }  
 .action-btn {  
 padding: 6px 14px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 font-size: 0.95em;  
 margin: 2px;  
 text-decoration: none;  
 display: inline-block;  
 }  
 .btn-edit { background: #17a2b8; color: #fff; }  
 .btn-delete { background: #dc3545; color: #fff; }  
 .btn-reset { background: #6c757d; color: #fff; }  
 .form-modal {  
 position: fixed;  
 top: 0; left: 0; right: 0; bottom: 0;  
 background: rgba(0,0,0,0.18);  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 z-index: 1000;  
 }  
 .form-box {  
 background: #fff;  
 border-radius: 12px;  
 box-shadow: 0 8px 32px rgba(0,0,0,0.13);  
 padding: 32px 36px 24px 36px;  
 min-width: 350px;  
 max-width: 95vw;  
 }  
 .form-box h3 {  
 margin-top: 0;  
 margin-bottom: 18px;  
 color: #333;  
 }  
 .form-row {  
 margin-bottom: 18px;  
 }  
 .form-row label {  
 display: inline-block;  
 width: 90px;  
 color: #555;  
 }  
 .form-row input, .form-row select {  
 padding: 7px 10px;  
 border: 1px solid #ccc;  
 border-radius: 5px;  
 width: 220px;  
 font-size: 1em;  
 }  
 .form-actions {  
 text-align: right;  
 }  
 .form-actions button, .form-actions a {  
 margin-left: 10px;  
 }  
 .error-msg { color: #dc3545; margin-bottom: 10px; }  
 .success-msg { color: #28a745; margin-bottom: 10px; }  
 @media (max-width: 700px) {  
 .container { padding: 10px; }  
 .form-box { padding: 18px 8px; }  
 }  
 </style>  
</head>  
<body>  
 <div class="header">  
 <div class="logo">👤 管理员管理</div>  
 <div class="user-info">当前登录：<%= currentAdmin.getFullName() %></div>  
 </div>  
 <div class="container">  
 <div class="page-title">管理员管理</div>  
 <div class="subtitle">管理系统所有管理员账户，可添加、编辑、删除和重置密码</div>  
 <button class="add-btn" onclick="showAddModal()">➕ 添加新管理员</button>  
 <table class="admin-table">  
 <thead>  
 <tr>  
 <th>ID</th>  
 <th>登录名</th>  
 <th>姓名</th>  
 <th>角色</th>  
 <th>部门</th>  
 <th>联系电话</th>  
 <th>操作</th>  
 </tr>  
 </thead>  
 <tbody>  
 <% if (admins != null && !admins.isEmpty()) {  
 for (Admin user : admins) { %>  
 <tr>  
 <td><%= user.getId() %></td>  
 <td><%= user.getLoginName() %></td>  
 <td><%= user.getFullName() %></td>  
 <td>  
 <span class="role-badge role-<%= user.getRole().toLowerCase().replace("\_", "-") %>">  
 <%= getRoleDisplayName(user.getRole()) %>  
 </span>  
 </td>  
 <td><%= user.getDeptName() != null ? user.getDeptName() : "未分配" %></td>  
 <td><%= user.getPhone() != null ? user.getPhone() : "未设置" %></td>  
 <td>  
 <button class="action-btn btn-edit" onclick="showEditModal(<%= user.getId() %>)">编辑</button>  
 <% if (user.getId() != currentAdmin.getId()) { %>  
 <button class="action-btn btn-delete" onclick="confirmDelete(<%= user.getId() %>)">删除</button>  
 <% } %>  
 <button class="action-btn btn-reset" onclick="resetPwd(<%= user.getId() %>)">重置密码</button>  
 </td>  
 </tr>  
 <% }  
 } else { %>  
 <tr><td colspan="7" style="text-align:center;color:#888;">暂无管理员记录</td></tr>  
 <% } %>  
 </tbody>  
 </table>  
 </div>  
 <!-- 添加管理员弹窗 -->  
 <div class="form-modal" id="addModal" style="display:none;">  
 <div class="form-box">  
 <h3>添加新管理员</h3>  
 <form method="post" action="<%=request.getContextPath()%>/admin/users">  
 <input type="hidden" name="action" value="add" />  
 <div class="form-row">  
 <label>登录名：</label>  
 <input type="text" name="loginName" required />  
 </div>  
 <div class="form-row">  
 <label>密码：</label>  
 <input type="password" name="password" required />  
 </div>  
 <div class="form-row">  
 <label>姓名：</label>  
 <input type="text" name="fullName" required />  
 </div>  
 <div class="form-row">  
 <label>部门：</label>  
 <select name="deptId">  
 <option value="">请选择部门</option>  
 <% if (departments != null) for (Department dept : departments) { %>  
 <option value="<%=dept.getId()%>"><%=dept.getDeptName()%></option>  
 <% } %>  
 </select>  
 </div>  
 <div class="form-row">  
 <label>联系电话：</label>  
 <input type="text" name="phone" required />  
 </div>  
 <div class="form-row">  
 <label>角色：</label>  
 <select name="role" required>  
 <% if ("SYSTEM\_ADMIN".equals(currentAdmin.getRole())) { %>  
 <option value="SYSTEM\_ADMIN">系统管理员</option>  
 <option value="SCHOOL\_ADMIN">学校管理员</option>  
 <option value="DEPT\_ADMIN">部门管理员</option>  
 <option value="AUDIT\_ADMIN">审计管理员</option>  
 <% } else if ("SCHOOL\_ADMIN".equals(currentAdmin.getRole())) { %>  
 <option value="DEPT\_ADMIN">部门管理员</option>  
 <% } %>  
 </select>  
 </div>  
 <% if (request.getAttribute("error") != null) { %>  
 <div class="error-msg">错误：<%=request.getAttribute("error")%></div>  
 <% } %>  
 <% if (request.getAttribute("message") != null) { %>  
 <div class="success-msg">提示：<%=request.getAttribute("message")%></div>  
 <% } %>  
 <div class="form-actions">  
 <button type="submit" class="add-btn" style="float:none;">提交</button>  
 <a href="#" onclick="closeAddModal();return false;" class="back-btn">取消</a>  
 </div>  
 </form>  
 </div>  
 </div>  
 <!-- 编辑管理员弹窗 -->  
 <div class="form-modal" id="editModal" style="display:none;">  
 <div class="form-box">  
 <h3>编辑管理员</h3>  
 <form method="post" action="<%=request.getContextPath()%>/admin/users">  
 <input type="hidden" name="action" value="update" />  
 <input type="hidden" name="id" id="editId" />  
 <div class="form-row">  
 <label>登录名：</label>  
 <input type="text" name="loginName" id="editLoginName" required />  
 </div>  
 <div class="form-row">  
 <label>姓名：</label>  
 <input type="text" name="fullName" id="editFullName" required />  
 </div>  
 <div class="form-row">  
 <label>部门：</label>  
 <select name="deptId" id="editDeptId">  
 <option value="">请选择部门</option>  
 <% if (departments != null) for (Department dept : departments) { %>  
 <option value="<%=dept.getId()%>"><%=dept.getDeptName()%></option>  
 <% } %>  
 </select>  
 </div>  
 <div class="form-row">  
 <label>联系电话：</label>  
 <input type="text" name="phone" id="editPhone" required />  
 </div>  
 <div class="form-row">  
 <label>角色：</label>  
 <select name="role" id="editRole" required>  
 <% if ("SYSTEM\_ADMIN".equals(currentAdmin.getRole())) { %>  
 <option value="SYSTEM\_ADMIN">系统管理员</option>  
 <option value="SCHOOL\_ADMIN">学校管理员</option>  
 <option value="DEPT\_ADMIN">部门管理员</option>  
 <option value="AUDIT\_ADMIN">审计管理员</option>  
 <% } else if ("SCHOOL\_ADMIN".equals(currentAdmin.getRole())) { %>  
 <option value="DEPT\_ADMIN">部门管理员</option>  
 <% } %>  
 </select>  
 </div>  
 <div class="form-actions">  
 <button type="submit" class="add-btn" style="float:none;">保存</button>  
 <a href="#" onclick="closeEditModal();return false;" class="back-btn">取消</a>  
 </div>  
 </form>  
 </div>  
 </div>  
 <script>  
 function showAddModal() {  
 *document*.getElementById('addModal').style.display = 'flex';  
 }  
 function closeAddModal() {  
 *document*.getElementById('addModal').style.display = 'none';  
 }  
 function showEditModal(id) {  
 // 这里可用AJAX请求管理员信息，或用JSP渲染时输出JS对象  
 var admins = <%=  
 admins != null ? new com.google.gson.Gson().toJson(admins) : "[]"  
 %>;  
 var adminList = typeof admins === 'string' ? *JSON*.parse(admins) : admins;  
 var admin = adminList.find(function(a){return a.id==id});  
 if (!admin) return;  
 *document*.getElementById('editId').value = admin.id;  
 *document*.getElementById('editLoginName').value = admin.loginName;  
 *document*.getElementById('editFullName').value = admin.fullName;  
 *document*.getElementById('editDeptId').value = admin.deptId;  
 *document*.getElementById('editPhone').value = admin.phone;  
 *document*.getElementById('editRole').value = admin.role;  
 *document*.getElementById('editModal').style.display = 'flex';  
 }  
 function closeEditModal() {  
 *document*.getElementById('editModal').style.display = 'none';  
 }  
 function confirmDelete(id) {  
 if (confirm('确定要删除该管理员吗？')) {  
 *window*.*location*.href = '<%=request.getContextPath()%>/admin/users?action=delete&id=' + id;  
 }  
 }  
 function resetPwd(id) {  
 if (confirm('确定要重置该管理员的密码吗？')) {  
 *window*.*location*.href = '<%=request.getContextPath()%>/admin/users?action=resetPassword&id=' + id;  
 }  
 }  
 </script>  
 <%!  
 private String getRoleDisplayName(String role) {  
 switch (role) {  
 case "SYSTEM\_ADMIN": return "系统管理员";  
 case "SCHOOL\_ADMIN": return "学校管理员";  
 case "DEPT\_ADMIN": return "部门管理员";  
 case "AUDIT\_ADMIN": return "审计管理员";  
 default: return role;  
 }  
 }  
 %>  
</body>  
</html>

#### src/main/webapp/admin/appointments.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **page** import="com.zjut.passcode.bean.Admin" %>  
<%@ **page** import="com.zjut.passcode.bean.Appointment" %>  
<%@ **page** import="java.util.List" %>  
<%  
 // Check if admin is logged in  
 Admin admin = (Admin) request.getSession().getAttribute("admin");  
 if (admin == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
   
 List<Appointment> appointments = (List<Appointment>) request.getAttribute("appointments");  
%>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>预约管理 - 校园通行码系统</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
   
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 }  
   
 .header {  
 background: white;  
 box-shadow: 0 2px 10px rgba(0,0,0,0.1);  
 padding: 20px;  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 }  
   
 .logo {  
 font-size: 1.5em;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .user-info {  
 display: flex;  
 align-items: center;  
 gap: 20px;  
 }  
   
 .user-name {  
 color: #333;  
 font-weight: 500;  
 }  
   
 .back-btn {  
 background: #6c757d;  
 color: white;  
 padding: 8px 16px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 text-decoration: none;  
 font-size: 0.9em;  
 }  
   
 .back-btn:hover {  
 background: #5a6268;  
 }  
   
 .container {  
 max-width: 1200px;  
 margin: 40px auto;  
 padding: 0 20px;  
 }  
   
 .page-title {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 margin-bottom: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 text-align: center;  
 }  
   
 .title {  
 font-size: 2em;  
 color: #333;  
 margin-bottom: 10px;  
 }  
   
 .subtitle {  
 color: #666;  
 font-size: 1.1em;  
 }  
   
 .appointments-content {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 }  
   
 .appointments-table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 20px;  
 }  
   
 .appointments-table th,  
 .appointments-table td {  
 padding: 12px;  
 text-align: left;  
 border-bottom: 1px solid #e9ecef;  
 }  
   
 .appointments-table th {  
 background: #f8f9fa;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .appointments-table tr:hover {  
 background: #f8f9fa;  
 }  
   
 .status-badge {  
 display: inline-block;  
 padding: 4px 8px;  
 border-radius: 12px;  
 font-size: 0.8em;  
 font-weight: bold;  
 }  
   
 .status-pending {  
 background: #fff3cd;  
 color: #856404;  
 }  
   
 .status-approved {  
 background: #d4edda;  
 color: #155724;  
 }  
   
 .status-rejected {  
 background: #f8d7da;  
 color: #721c24;  
 }  
   
 .action-btn {  
 padding: 6px 12px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 font-size: 0.8em;  
 margin: 2px;  
 text-decoration: none;  
 display: inline-block;  
 }  
   
 .btn-approve {  
 background: #28a745;  
 color: white;  
 }  
   
 .btn-reject {  
 background: #dc3545;  
 color: white;  
 }  
   
 .btn-view {  
 background: #17a2b8;  
 color: white;  
 }  
   
 .empty-state {  
 text-align: center;  
 padding: 40px;  
 color: #666;  
 }  
   
 .empty-state h3 {  
 margin-bottom: 10px;  
 color: #333;  
 }  
   
 @media (max-width: 768px) {  
 .header {  
 flex-direction: column;  
 gap: 15px;  
 }  
   
 .user-info {  
 flex-direction: column;  
 gap: 10px;  
 }  
   
 .title {  
 font-size: 1.5em;  
 }  
   
 .appointments-table {  
 font-size: 0.9em;  
 }  
   
 .appointments-table th,  
 .appointments-table td {  
 padding: 8px 4px;  
 }  
 }  
 </style>  
</head>  
<body>  
 <div class="header">  
 <div class="logo">📋 预约管理系统</div>  
 <div class="user-info">  
 <div class="user-name">  
 管理员：<%= admin.getFullName() %>  
 </div>  
 <a href="${pageContext.request.contextPath}/admin/index.jsp" class="back-btn">🔙 返回控制台</a>  
 </div>  
 </div>  
   
 <div class="container">  
 <div class="page-title">  
 <div class="title">预约管理</div>  
 <div class="subtitle">查看、审核和管理访客预约申请</div>  
 </div>  
   
 <div class="appointments-content">  
 <% if (appointments != null && !appointments.isEmpty()) { %>  
 <table class="appointments-table">  
 <thead>  
 <tr>  
 <th>ID</th>  
 <th>访客姓名</th>  
 <th>预约类型</th>  
 <th>进校时间</th>  
 <th>校区</th>  
 <th>状态</th>  
 <th>申请时间</th>  
 <th>操作</th>  
 </tr>  
 </thead>  
 <tbody>  
 <% for (Appointment appointment : appointments) { %>  
 <tr>  
 <td><%= appointment.getId() %></td>  
 <td><%= appointment.getVisitorName() %></td>  
 <td><%= getAppointmentTypeDisplay(appointment.getAppointmentType()) %></td>  
 <td><%= appointment.getEntryTime() %></td>  
 <td><%= appointment.getCampus() %></td>  
 <td>  
 <span class="status-badge status-<%= appointment.getStatus().toLowerCase() %>">  
 <%= getStatusDisplay(appointment.getStatus()) %>  
 </span>  
 </td>  
 <td><%= appointment.getCreatedAt() %></td>  
 <td>  
 <a href="<%=request.getContextPath()%>/passcode/view?id=<%= appointment.getId() %>" class="action-btn btn-view" target="\_blank">查看</a>  
 <% if ("PENDING".equals(appointment.getStatus())) { %>  
 <form method="post" action="<%=request.getContextPath()%>/admin/appointments" style="display:inline;" onsubmit="return confirmApprove();">  
 <input type="hidden" name="action" value="approve" />  
 <input type="hidden" name="appointmentId" value="<%= appointment.getId() %>" />  
 <button type="submit" class="action-btn btn-approve">通过</button>  
 </form>  
 <form method="post" action="<%=request.getContextPath()%>/admin/appointments" style="display:inline;" onsubmit="return confirmReject();">  
 <input type="hidden" name="action" value="reject" />  
 <input type="hidden" name="appointmentId" value="<%= appointment.getId() %>" />  
 <button type="submit" class="action-btn btn-reject">拒绝</button>  
 </form>  
 <% } %>  
 </td>  
 </tr>  
 <% } %>  
 </tbody>  
 </table>  
 <% } else { %>  
 <div class="empty-state">  
 <h3>暂无预约记录</h3>  
 <p>当前系统中没有预约申请记录</p>  
 </div>  
 <% } %>  
 </div>  
 </div>  
   
 <%!  
 private String getAppointmentTypeDisplay(String type) {  
 switch (type) {  
 case "PUBLIC":  
 return "社会公众预约";  
 case "OFFICIAL":  
 return "公务访问";  
 default:  
 return type;  
 }  
 }  
   
 private String getStatusDisplay(String status) {  
 switch (status) {  
 case "PENDING":  
 return "待审核";  
 case "APPROVED":  
 return "已通过";  
 case "REJECTED":  
 return "已拒绝";  
 default:  
 return status;  
 }  
 }  
 %>  
 <script>  
 function confirmApprove() {  
 return confirm('确定要通过该预约吗？');  
 }  
 function confirmReject() {  
 return confirm('确定要拒绝该预约吗？');  
 }  
 </script>  
</body>  
</html>

#### src/main/webapp/admin/audit\_log\_manage.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **page** import="com.zjut.passcode.bean.Admin" %>  
<%@ **page** import="java.util.List" %>  
<%  
 // Check if admin is logged in  
 Admin admin = (Admin) session.getAttribute("admin");  
 if (admin == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
   
 // Check if user has audit admin role  
 if (!"AUDIT\_ADMIN".equals(admin.getRole())) {  
 response.sendRedirect(request.getContextPath() + "/admin/index.jsp");  
 return;  
 }  
%>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>审计日志管理 - 校园通行码系统</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
   
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 }  
   
 .header {  
 background: white;  
 box-shadow: 0 2px 10px rgba(0,0,0,0.1);  
 padding: 20px;  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 }  
   
 .logo {  
 font-size: 1.5em;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .user-info {  
 display: flex;  
 align-items: center;  
 gap: 20px;  
 }  
   
 .user-name {  
 color: #333;  
 font-weight: 500;  
 }  
   
 .back-btn {  
 background: #6c757d;  
 color: white;  
 padding: 8px 16px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 text-decoration: none;  
 font-size: 0.9em;  
 }  
   
 .back-btn:hover {  
 background: #5a6268;  
 }  
   
 .container {  
 max-width: 1200px;  
 margin: 40px auto;  
 padding: 0 20px;  
 }  
   
 .page-title {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 margin-bottom: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 text-align: center;  
 }  
   
 .title {  
 font-size: 2em;  
 color: #333;  
 margin-bottom: 10px;  
 }  
   
 .subtitle {  
 color: #666;  
 font-size: 1.1em;  
 }  
   
 .audit-content {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 }  
   
 .audit-section {  
 margin-bottom: 30px;  
 }  
   
 .section-title {  
 font-size: 1.5em;  
 color: #333;  
 margin-bottom: 20px;  
 border-bottom: 2px solid #667eea;  
 padding-bottom: 10px;  
 }  
   
 .audit-grid {  
 display: grid;  
 grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));  
 gap: 20px;  
 }  
   
 .audit-card {  
 background: #f8f9fa;  
 border-radius: 10px;  
 padding: 20px;  
 border-left: 4px solid #667eea;  
 }  
   
 .audit-card h4 {  
 color: #333;  
 margin-bottom: 10px;  
 }  
   
 .audit-card p {  
 color: #666;  
 margin-bottom: 5px;  
 }  
   
 .status-badge {  
 display: inline-block;  
 padding: 4px 8px;  
 border-radius: 12px;  
 font-size: 0.8em;  
 font-weight: bold;  
 }  
   
 .status-success {  
 background: #d4edda;  
 color: #155724;  
 }  
   
 .status-warning {  
 background: #fff3cd;  
 color: #856404;  
 }  
   
 .status-danger {  
 background: #f8d7da;  
 color: #721c24;  
 }  
   
 .status-info {  
 background: #d1ecf1;  
 color: #0c5460;  
 }  
   
 .stats-overview {  
 display: grid;  
 grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));  
 gap: 20px;  
 margin-bottom: 30px;  
 }  
   
 .stat-item {  
 background: #f8f9fa;  
 border-radius: 10px;  
 padding: 20px;  
 text-align: center;  
 border: 1px solid #e9ecef;  
 }  
   
 .stat-number {  
 font-size: 2em;  
 font-weight: bold;  
 color: #667eea;  
 margin-bottom: 5px;  
 }  
   
 .stat-label {  
 color: #666;  
 font-size: 0.9em;  
 }  
   
 @media (max-width: 768px) {  
 .header {  
 flex-direction: column;  
 gap: 15px;  
 }  
   
 .user-info {  
 flex-direction: column;  
 gap: 10px;  
 }  
   
 .title {  
 font-size: 1.5em;  
 }  
   
 .audit-grid {  
 grid-template-columns: 1fr;  
 }  
   
 .stats-overview {  
 grid-template-columns: repeat(2, 1fr);  
 }  
 }  
 </style>  
</head>  
<body>  
 <div class="header">  
 <div class="logo">📊 审计日志管理系统</div>  
 <div class="user-info">  
 <div class="user-name">  
 审计管理员：<%= admin.getFullName() %>  
 </div>  
 <a href="${pageContext.request.contextPath}/admin/index.jsp" class="back-btn">🔙 返回控制台</a>  
 </div>  
 </div>  
   
 <div class="container">  
 <div class="page-title">  
 <div class="title">审计日志管理</div>  
 <div class="subtitle">监控系统安全，查看操作日志</div>  
 </div>  
   
 <div class="stats-overview">  
 <div class="stat-item">  
 <div class="stat-number">0</div>  
 <div class="stat-label">今日登录</div>  
 </div>  
 <div class="stat-item">  
 <div class="stat-number">0</div>  
 <div class="stat-label">登录失败</div>  
 </div>  
 <div class="stat-item">  
 <div class="stat-number">0</div>  
 <div class="stat-label">系统错误</div>  
 </div>  
 <div class="stat-item">  
 <div class="stat-number">0</div>  
 <div class="stat-label">安全事件</div>  
 </div>  
 </div>  
   
 <div class="audit-content">  
 <div class="audit-section">  
 <h3 class="section-title">🔍 最近审计事件</h3>  
 <%-- 分页表格 --%>  
 <% List auditLogs = (List)request.getAttribute("auditLogs"); %>  
 <% Integer page = (Integer)request.getAttribute("page"); %>  
 <% Integer totalPages = (Integer)request.getAttribute("totalPages"); %>  
 <table class="logs-table">  
 <thead>  
 <tr>  
 <th>ID</th>  
 <th>管理员</th>  
 <th>操作</th>  
 <th>详情</th>  
 <th>IP地址</th>  
 <th>时间</th>  
 </tr>  
 </thead>  
 <tbody>  
 <% if (auditLogs != null && !auditLogs.isEmpty()) {  
 for (Object obj : auditLogs) {  
 com.zjut.passcode.bean.AuditLog log = (com.zjut.passcode.bean.AuditLog)obj; %>  
 <tr>  
 <td><%= log.getId() %></td>  
 <td><%= log.getAdminName() %></td>  
 <td><%= log.getAction() %></td>  
 <td><%= log.getDetails() %></td>  
 <td><%= log.getIpAddress() %></td>  
 <td><%= log.getCreatedAt() %></td>  
 </tr>  
 <% }  
 } else { %>  
 <tr><td colspan="6" style="text-align:center;">暂无审计日志</td></tr>  
 <% } %>  
 </tbody>  
 </table>  
 <%-- 分页控件 --%>  
 <div style="margin:20px 0;text-align:center;">  
 <% if (totalPages != null && totalPages > 1) { %>  
 <% for (int i = 1; i <= totalPages; i++) { %>  
 <% if (i == page) { %>  
 <span style="margin:0 5px;font-weight:bold;"><%=i%></span>  
 <% } else { %>  
 <a href="?page=<%=i%>" style="margin:0 5px;">[<%=i%>]</a>  
 <% } %>  
 <% } %>  
 <% } %>  
 </div>  
 </div>  
 <div class="audit-section">  
 <h3 class="section-title">⚠️ 安全警告</h3>  
 <div class="audit-grid">  
 <div class="audit-card">  
 <h4>密码过期提醒</h4>  
 <p><strong>用户：</strong>admin</p>  
 <p><strong>过期时间：</strong>2024-04-15</p>  
 <p><strong>建议：</strong>请及时修改密码</p>  
 <span class="status-badge status-warning">需要关注</span>  
 </div>  
 <div class="audit-card">  
 <h4>登录尝试</h4>  
 <p><strong>用户：</strong>unknown\_user</p>  
 <p><strong>时间：</strong>2024-01-15 09:55:00</p>  
 <p><strong>结果：</strong>用户不存在</p>  
 <span class="status-badge status-danger">登录失败</span>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
   
 <script>  
 // Auto refresh every 30 seconds  
 setTimeout(() => {  
 *location*.reload();  
 }, 30000);  
   
 // Update current time  
 function updateTime() {  
 const now = new *Date*();  
 *console*.log('Current time: ' + now.toLocaleString('zh-CN'));  
 }  
   
 updateTime();  
 setInterval(updateTime, 60000); // Update every minute  
 </script>  
</body>  
</html>

#### src/main/webapp/admin/audit\_logs.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **page** import="com.zjut.passcode.bean.Admin" %>  
<%@ **page** import="com.zjut.passcode.bean.AuditLog" %>  
<%@ **page** import="java.util.List" %>  
<%  
 // Check if admin is logged in  
 Admin admin = (Admin) session.getAttribute("admin");  
 if (admin == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
   
 List<AuditLog> auditLogs = (List<AuditLog>) request.getAttribute("auditLogs");  
%>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>审计日志 - 校园通行码系统</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
   
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 }  
   
 .header {  
 background: white;  
 box-shadow: 0 2px 10px rgba(0,0,0,0.1);  
 padding: 20px;  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 }  
   
 .logo {  
 font-size: 1.5em;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .user-info {  
 display: flex;  
 align-items: center;  
 gap: 20px;  
 }  
   
 .user-name {  
 color: #333;  
 font-weight: 500;  
 }  
   
 .back-btn {  
 background: #6c757d;  
 color: white;  
 padding: 8px 16px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 text-decoration: none;  
 font-size: 0.9em;  
 }  
   
 .back-btn:hover {  
 background: #5a6268;  
 }  
   
 .container {  
 max-width: 1200px;  
 margin: 40px auto;  
 padding: 0 20px;  
 }  
   
 .page-title {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 margin-bottom: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 text-align: center;  
 }  
   
 .title {  
 font-size: 2em;  
 color: #333;  
 margin-bottom: 10px;  
 }  
   
 .subtitle {  
 color: #666;  
 font-size: 1.1em;  
 }  
   
 .logs-content {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 }  
   
 .logs-table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 20px;  
 }  
   
 .logs-table th,  
 .logs-table td {  
 padding: 12px;  
 text-align: left;  
 border-bottom: 1px solid #e9ecef;  
 }  
   
 .logs-table th {  
 background: #f8f9fa;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .logs-table tr:hover {  
 background: #f8f9fa;  
 }  
   
 .action-badge {  
 display: inline-block;  
 padding: 4px 8px;  
 border-radius: 12px;  
 font-size: 0.8em;  
 font-weight: bold;  
 }  
   
 .action-login {  
 background: #d4edda;  
 color: #155724;  
 }  
   
 .action-logout {  
 background: #f8d7da;  
 color: #721c24;  
 }  
   
 .action-system-error {  
 background: #fff3cd;  
 color: #856404;  
 }  
   
 .empty-state {  
 text-align: center;  
 padding: 40px;  
 color: #666;  
 }  
   
 .empty-state h3 {  
 margin-bottom: 10px;  
 color: #333;  
 }  
   
 @media (max-width: 768px) {  
 .header {  
 flex-direction: column;  
 gap: 15px;  
 }  
   
 .user-info {  
 flex-direction: column;  
 gap: 10px;  
 }  
   
 .title {  
 font-size: 1.5em;  
 }  
   
 .logs-table {  
 font-size: 0.9em;  
 }  
   
 .logs-table th,  
 .logs-table td {  
 padding: 8px 4px;  
 }  
 }  
 </style>  
</head>  
<body>  
 <div class="header">  
 <div class="logo">📊 审计日志系统</div>  
 <div class="user-info">  
 <div class="user-name">  
 管理员：<%= admin.getFullName() %>  
 </div>  
 <a href="${pageContext.request.contextPath}/admin/index.jsp" class="back-btn">🔙 返回控制台</a>  
 </div>  
 </div>  
   
 <div class="container">  
 <div class="page-title">  
 <div class="title">审计日志</div>  
 <div class="subtitle">查看系统操作和登录日志</div>  
 </div>  
   
 <div class="logs-content">  
 <% if (auditLogs != null && !auditLogs.isEmpty()) { %>  
 <table class="logs-table">  
 <thead>  
 <tr>  
 <th>ID</th>  
 <th>管理员</th>  
 <th>操作</th>  
 <th>详情</th>  
 <th>IP地址</th>  
 <th>时间</th>  
 </tr>  
 </thead>  
 <tbody>  
 <% for (AuditLog log : auditLogs) { %>  
 <tr>  
 <td><%= log.getId() %></td>  
 <td><%= log.getAdminName() %></td>  
 <td>  
 <span class="action-badge action-<%= getActionClass(log.getAction()) %>">  
 <%= log.getAction() %>  
 </span>  
 </td>  
 <td><%= log.getDetails() %></td>  
 <td><%= log.getIpAddress() %></td>  
 <td><%= log.getCreatedAt() %></td>  
 </tr>  
 <% } %>  
 </tbody>  
 </table>  
 <% } else { %>  
 <div class="empty-state">  
 <h3>暂无审计日志</h3>  
 <p>当前系统中没有审计日志记录</p>  
 </div>  
 <% } %>  
 </div>  
 </div>  
   
 <%!  
 private String getActionClass(String action) {  
 if (action.contains("登录")) {  
 return "login";  
 } else if (action.contains("退出")) {  
 return "logout";  
 } else if (action.contains("错误") || action.contains("Error")) {  
 return "system-error";  
 } else {  
 return "login";  
 }  
 }  
 %>  
</body>  
</html>

#### src/main/webapp/admin/departments.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **page** import="com.zjut.passcode.bean.Admin" %>  
<%@ **page** import="com.zjut.passcode.bean.Department" %>  
<%@ **page** import="java.util.List" %>  
<%  
 // Check if admin is logged in  
 Admin admin = (Admin) session.getAttribute("admin");  
 if (admin == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
   
 List<Department> departments = (List<Department>) request.getAttribute("departments");  
%>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>部门管理 - 校园通行码系统</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
   
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 }  
   
 .header {  
 background: white;  
 box-shadow: 0 2px 10px rgba(0,0,0,0.1);  
 padding: 20px;  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 }  
   
 .logo {  
 font-size: 1.5em;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .user-info {  
 display: flex;  
 align-items: center;  
 gap: 20px;  
 }  
   
 .user-name {  
 color: #333;  
 font-weight: 500;  
 }  
   
 .back-btn {  
 background: #6c757d;  
 color: white;  
 padding: 8px 16px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 text-decoration: none;  
 font-size: 0.9em;  
 }  
   
 .back-btn:hover {  
 background: #5a6268;  
 }  
   
 .container {  
 max-width: 1200px;  
 margin: 40px auto;  
 padding: 0 20px;  
 }  
   
 .page-title {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 margin-bottom: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 text-align: center;  
 }  
   
 .title {  
 font-size: 2em;  
 color: #333;  
 margin-bottom: 10px;  
 }  
   
 .subtitle {  
 color: #666;  
 font-size: 1.1em;  
 }  
   
 .departments-content {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 }  
   
 .add-dept-btn {  
 background: #28a745;  
 color: white;  
 padding: 10px 20px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 font-size: 1em;  
 margin-bottom: 20px;  
 text-decoration: none;  
 display: inline-block;  
 }  
   
 .add-dept-btn:hover {  
 background: #218838;  
 }  
   
 .departments-table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 20px;  
 }  
   
 .departments-table th,  
 .departments-table td {  
 padding: 12px;  
 text-align: left;  
 border-bottom: 1px solid #e9ecef;  
 }  
   
 .departments-table th {  
 background: #f8f9fa;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .departments-table tr:hover {  
 background: #f8f9fa;  
 }  
   
 .type-badge {  
 display: inline-block;  
 padding: 4px 8px;  
 border-radius: 12px;  
 font-size: 0.8em;  
 font-weight: bold;  
 }  
   
 .type-administrative {  
 background: #dc3545;  
 color: white;  
 }  
   
 .type-direct {  
 background: #fd7e14;  
 color: white;  
 }  
   
 .type-college {  
 background: #28a745;  
 color: white;  
 }  
   
 .action-btn {  
 padding: 6px 12px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 font-size: 0.8em;  
 margin: 2px;  
 text-decoration: none;  
 display: inline-block;  
 }  
   
 .btn-edit {  
 background: #17a2b8;  
 color: white;  
 }  
   
 .btn-edit:hover {  
 background: #138496;  
 }  
   
 .btn-delete {  
 background: #dc3545;  
 color: white;  
 }  
   
 .btn-delete:hover {  
 background: #c82333;  
 }  
   
 .empty-state {  
 text-align: center;  
 padding: 40px;  
 color: #666;  
 }  
   
 .empty-state h3 {  
 margin-bottom: 10px;  
 color: #333;  
 }  
   
 .message {  
 padding: 12px;  
 border-radius: 8px;  
 margin-bottom: 20px;  
 border: 1px solid;  
 }  
   
 .success-message {  
 background: #d4edda;  
 color: #155724;  
 border-color: #c3e6cb;  
 }  
   
 .error-message {  
 background: #f8d7da;  
 color: #721c24;  
 border-color: #f5c6cb;  
 }  
   
 .delete-confirm {  
 background: #fff3cd;  
 color: #856404;  
 border: 1px solid #ffeaa7;  
 padding: 10px;  
 border-radius: 5px;  
 margin-bottom: 20px;  
 }  
   
 @media (max-width: 768px) {  
 .header {  
 flex-direction: column;  
 gap: 15px;  
 }  
   
 .user-info {  
 flex-direction: column;  
 gap: 10px;  
 }  
   
 .title {  
 font-size: 1.5em;  
 }  
   
 .departments-table {  
 font-size: 0.9em;  
 }  
   
 .departments-table th,  
 .departments-table td {  
 padding: 8px 4px;  
 }  
 }  
 </style>  
 <script>  
 function confirmDelete(element) {  
 var deptId = element.getAttribute('data-id');  
 var deptName = element.getAttribute('data-name');  
 if (confirm('确定要删除部门 "' + deptName + '" 吗？此操作不可撤销。')) {  
 *window*.*location*.href = '${pageContext.request.contextPath}/admin/department/delete?id=' + deptId;  
 }  
 }  
 </script>  
</head>  
<body>  
 <div class="header">  
 <div class="logo">🏢 部门管理系统</div>  
 <div class="user-info">  
 <div class="user-name">  
 管理员：<%= admin.getFullName() %>  
 </div>  
 <a href="${pageContext.request.contextPath}/admin/index.jsp" class="back-btn">🔙 返回控制台</a>  
 </div>  
 </div>  
   
 <div class="container">  
 <div class="page-title">  
 <div class="title">部门管理</div>  
 <div class="subtitle">管理学校部门和机构信息</div>  
 </div>  
   
 <div class="departments-content">  
 <% if (request.getAttribute("success") != null) { %>  
 <div class="message success-message">  
 ✅ <%= request.getAttribute("success") %>  
 </div>  
 <% } %>  
   
 <% if (request.getAttribute("error") != null) { %>  
 <div class="message error-message">  
 ❌ <%= request.getAttribute("error") %>  
 </div>  
 <% } %>  
   
 <a href="${pageContext.request.contextPath}/admin/department/add" class="add-dept-btn">➕ 添加新部门</a>  
   
 <% if (departments != null && !departments.isEmpty()) { %>  
 <table class="departments-table">  
 <thead>  
 <tr>  
 <th>ID</th>  
 <th>部门编号</th>  
 <th>部门名称</th>  
 <th>部门类型</th>  
 <th>操作</th>  
 </tr>  
 </thead>  
 <tbody>  
 <% for (Department dept : departments) { %>  
 <tr>  
 <td><%= dept.getId() %></td>  
 <td><%= dept.getDeptNo() %></td>  
 <td><%= dept.getDeptName() %></td>  
 <td>  
 <span class="type-badge type-<%= getTypeClass(dept.getDeptType()) %>">  
 <%= dept.getDeptType() %>  
 </span>  
 </td>  
 <td>  
 <a href="${pageContext.request.contextPath}/admin/department/edit?id=<%= dept.getId() %>"   
 class="action-btn btn-edit">✏️ 编辑</a>  
 <a href="javascript:void(0)"   
 data-id="<%= dept.getId() %>"  
 data-name="<%= dept.getDeptName() %>"  
 onclick="confirmDelete(this)"   
 class="action-btn btn-delete">🗑️ 删除</a>  
 </td>  
 </tr>  
 <% } %>  
 </tbody>  
 </table>  
 <% } else { %>  
 <div class="empty-state">  
 <h3>暂无部门记录</h3>  
 <p>当前系统中没有部门记录，点击上方"添加新部门"按钮开始创建</p>  
 </div>  
 <% } %>  
 </div>  
 </div>  
   
 <%!  
 private String getTypeClass(String deptType) {  
 switch (deptType) {  
 case "行政部门":  
 return "administrative";  
 case "直属部门":  
 return "direct";  
 case "学院":  
 return "college";  
 default:  
 return "administrative";  
 }  
 }  
 %>  
</body>  
</html>

#### src/main/webapp/admin/edit\_department.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **page** import="com.zjut.passcode.bean.Admin" %>  
<%@ **page** import="com.zjut.passcode.bean.Department" %>  
<%  
 // Check if admin is logged in  
 Admin admin = (Admin) session.getAttribute("admin");  
 if (admin == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
   
 Department department = (Department) request.getAttribute("department");  
 if (department == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/departments");  
 return;  
 }  
%>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>编辑部门 - 校园通行码系统</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
   
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 }  
   
 .header {  
 background: white;  
 box-shadow: 0 2px 10px rgba(0,0,0,0.1);  
 padding: 20px;  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 }  
   
 .logo {  
 font-size: 1.5em;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .user-info {  
 display: flex;  
 align-items: center;  
 gap: 20px;  
 }  
   
 .user-name {  
 color: #333;  
 font-weight: 500;  
 }  
   
 .back-btn {  
 background: #6c757d;  
 color: white;  
 padding: 8px 16px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 text-decoration: none;  
 font-size: 0.9em;  
 }  
   
 .back-btn:hover {  
 background: #5a6268;  
 }  
   
 .container {  
 max-width: 800px;  
 margin: 40px auto;  
 padding: 0 20px;  
 }  
   
 .page-title {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 margin-bottom: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 text-align: center;  
 }  
   
 .title {  
 font-size: 2em;  
 color: #333;  
 margin-bottom: 10px;  
 }  
   
 .subtitle {  
 color: #666;  
 font-size: 1.1em;  
 }  
   
 .form-content {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 }  
   
 .form-group {  
 margin-bottom: 20px;  
 }  
   
 .form-label {  
 display: block;  
 margin-bottom: 8px;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .form-input {  
 width: 100%;  
 padding: 12px;  
 border: 2px solid #e9ecef;  
 border-radius: 8px;  
 font-size: 1em;  
 transition: border-color 0.3s;  
 }  
   
 .form-input:focus {  
 outline: none;  
 border-color: #667eea;  
 }  
   
 .form-select {  
 width: 100%;  
 padding: 12px;  
 border: 2px solid #e9ecef;  
 border-radius: 8px;  
 font-size: 1em;  
 background: white;  
 cursor: pointer;  
 }  
   
 .form-select:focus {  
 outline: none;  
 border-color: #667eea;  
 }  
   
 .btn-group {  
 display: flex;  
 gap: 15px;  
 margin-top: 30px;  
 }  
   
 .btn {  
 padding: 12px 24px;  
 border: none;  
 border-radius: 8px;  
 cursor: pointer;  
 font-size: 1em;  
 text-decoration: none;  
 display: inline-block;  
 text-align: center;  
 transition: background-color 0.3s;  
 }  
   
 .btn-primary {  
 background: #17a2b8;  
 color: white;  
 }  
   
 .btn-primary:hover {  
 background: #138496;  
 }  
   
 .btn-secondary {  
 background: #6c757d;  
 color: white;  
 }  
   
 .btn-secondary:hover {  
 background: #5a6268;  
 }  
   
 .error-message {  
 background: #f8d7da;  
 color: #721c24;  
 padding: 12px;  
 border-radius: 8px;  
 margin-bottom: 20px;  
 border: 1px solid #f5c6cb;  
 }  
   
 .success-message {  
 background: #d4edda;  
 color: #155724;  
 padding: 12px;  
 border-radius: 8px;  
 margin-bottom: 20px;  
 border: 1px solid #c3e6cb;  
 }  
   
 @media (max-width: 768px) {  
 .header {  
 flex-direction: column;  
 gap: 15px;  
 }  
   
 .user-info {  
 flex-direction: column;  
 gap: 10px;  
 }  
   
 .title {  
 font-size: 1.5em;  
 }  
   
 .btn-group {  
 flex-direction: column;  
 }  
 }  
 </style>  
</head>  
<body>  
 <div class="header">  
 <div class="logo">🏢 部门管理系统</div>  
 <div class="user-info">  
 <div class="user-name">  
 管理员：<%= admin.getFullName() %>  
 </div>  
 <a href="${pageContext.request.contextPath}/admin/departments" class="back-btn">🔙 返回部门管理</a>  
 </div>  
 </div>  
   
 <div class="container">  
 <div class="page-title">  
 <div class="title">编辑部门</div>  
 <div class="subtitle">修改部门信息</div>  
 </div>  
   
 <div class="form-content">  
 <% if (request.getAttribute("error") != null) { %>  
 <div class="error-message">  
 <%= request.getAttribute("error") %>  
 </div>  
 <% } %>  
   
 <% if (request.getAttribute("success") != null) { %>  
 <div class="success-message">  
 <%= request.getAttribute("success") %>  
 </div>  
 <% } %>  
   
 <form action="${pageContext.request.contextPath}/admin/department/edit" method="post">  
 <input type="hidden" name="id" value="<%= department.getId() %>">  
   
 <div class="form-group">  
 <label for="deptNo" class="form-label">部门编号 \*</label>  
 <input type="text" id="deptNo" name="deptNo" class="form-input"   
 value="<%= department.getDeptNo() %>" required   
 placeholder="请输入部门编号，如：D001">  
 </div>  
   
 <div class="form-group">  
 <label for="deptName" class="form-label">部门名称 \*</label>  
 <input type="text" id="deptName" name="deptName" class="form-input"   
 value="<%= department.getDeptName() %>" required   
 placeholder="请输入部门名称">  
 </div>  
   
 <div class="form-group">  
 <label for="deptType" class="form-label">部门类型 \*</label>  
 <select id="deptType" name="deptType" class="form-select" required>  
 <option value="">请选择部门类型</option>  
 <option value="行政部门" <%= "行政部门".equals(department.getDeptType()) ? "selected" : "" %>>行政部门</option>  
 <option value="直属部门" <%= "直属部门".equals(department.getDeptType()) ? "selected" : "" %>>直属部门</option>  
 <option value="学院" <%= "学院".equals(department.getDeptType()) ? "selected" : "" %>>学院</option>  
 </select>  
 </div>  
   
 <div class="btn-group">  
 <button type="submit" class="btn btn-primary">✅ 保存修改</button>  
 <a href="${pageContext.request.contextPath}/admin/departments" class="btn btn-secondary">❌ 取消</a>  
 </div>  
 </form>  
 </div>  
 </div>  
</body>  
</html>

#### src/main/webapp/admin/edit\_user.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **page** import="com.zjut.passcode.bean.Admin" %>  
<%@ **page** import="java.util.List" %>  
<%  
 Admin user = (Admin) request.getAttribute("editUser");  
 Admin currentAdmin = (Admin) session.getAttribute("admin");  
 List departments = (List)request.getAttribute("departments");  
 boolean isRoleDisabled = "SCHOOL\_ADMIN".equals(currentAdmin.getRole()) && "DEPT\_ADMIN".equals(user.getRole());  
%>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>编辑管理员 - 校园通行码系统</title>  
 <style>  
 body { font-family: 'Microsoft YaHei', Arial, sans-serif; background: linear-gradient(135deg, #667eea 0%, #764ba2 100%); min-height: 100vh; }  
 .container { max-width: 600px; margin: 40px auto; background: #fff; border-radius: 15px; box-shadow: 0 10px 30px rgba(0,0,0,0.1); padding: 40px; }  
 h2 { text-align: center; margin-bottom: 30px; }  
 label { display: block; margin-bottom: 8px; color: #333; }  
 input, select { width: 100%; padding: 10px; margin-bottom: 20px; border-radius: 5px; border: 1px solid #ccc; }  
 .btn { background: #28a745; color: #fff; border: none; padding: 10px 30px; border-radius: 5px; cursor: pointer; font-size: 1em; }  
 .btn-cancel { background: #6c757d; margin-left: 10px; }  
 .error { color: #dc3545; margin-bottom: 10px; }  
 .message { color: #28a745; margin-bottom: 10px; }  
 .disabled { background-color: #f8f9fa; color: #6c757d; cursor: not-allowed; }  
 </style>  
</head>  
<body>  
 <div class="container">  
 <h2>编辑管理员</h2>  
 <form method="post" action="<%=request.getContextPath()%>/admin/users">  
 <input type="hidden" name="action" value="edit" />  
 <input type="hidden" name="userId" value="<%= user.getId() %>" />  
 <div>  
 <label>登录名：</label>  
 <input type="text" name="loginName" value="<%= user.getLoginName() %>" required />  
 </div>  
 <div>  
 <label>姓名：</label>  
 <input type="text" name="fullName" value="<%= user.getFullName() %>" required />  
 </div>  
 <div>  
 <label>部门：</label>  
 <select name="deptId">  
 <option value="">请选择部门</option>  
 <% if (departments != null) { for (Object obj : departments) { com.zjut.passcode.bean.Department dept = (com.zjut.passcode.bean.Department)obj; %>  
 <option value="<%=dept.getId()%>" <%=dept.getId()==user.getDeptId()?"selected":""%>><%=dept.getDeptName()%></option>  
 <% }} %>  
 </select>  
 </div>  
 <div>  
 <label>联系电话：</label>  
 <input type="text" name="phone" value="<%= user.getPhone() %>" required />  
 </div>  
 <div>  
 <label>角色：</label>  
 <% if (isRoleDisabled) { %>  
 <select name="role" required disabled>  
 <% } else { %>  
 <select name="role" required>  
 <% } %>  
 <% if ("SYSTEM\_ADMIN".equals(currentAdmin.getRole())) { %>  
 <option value="SYSTEM\_ADMIN" <%= "SYSTEM\_ADMIN".equals(user.getRole())?"selected":""%>>系统管理员</option>  
 <option value="SCHOOL\_ADMIN" <%= "SCHOOL\_ADMIN".equals(user.getRole())?"selected":""%>>学校管理员</option>  
 <option value="DEPT\_ADMIN" <%= "DEPT\_ADMIN".equals(user.getRole())?"selected":""%>>部门管理员</option>  
 <option value="AUDIT\_ADMIN" <%= "AUDIT\_ADMIN".equals(user.getRole())?"selected":""%>>审计管理员</option>  
 <% } else if ("SCHOOL\_ADMIN".equals(currentAdmin.getRole())) { %>  
 <option value="DEPT\_ADMIN" <%= "DEPT\_ADMIN".equals(user.getRole())?"selected":""%>>部门管理员</option>  
 <% } %>  
 </select>  
 <% if (isRoleDisabled) { %>  
 <input type="hidden" name="role" value="DEPT\_ADMIN" />  
 <small style="color: #6c757d;">学校管理员不能改变部门管理员的角色</small>  
 <% } %>  
 </div>  
 <div id="deptAdminPerms" style="display:<%= "DEPT\_ADMIN".equals(user.getRole()) ? "block" : "none" %>;margin-bottom:20px;">  
 <label style="font-weight:bold;">部门管理员权限：</label>  
 <div style="margin:8px 0 0 10px;">  
 <input type="checkbox" name="canManagePublicAppointment" value="1" <%= user.isCanManagePublicAppointment() ? "checked" : "" %> />  
 <span>可管理社会公众预约（默认只能管理公务预约）</span>  
 </div>  
 <div style="margin:8px 0 0 10px;">  
 <input type="checkbox" name="canReportPublicAppointment" value="1" <%= user.isCanReportPublicAppointment() ? "checked" : "" %> />  
 <span>可统计社会公众预约（默认只能统计公务预约）</span>  
 </div>  
 </div>  
 <script>  
 // 动态显示/隐藏部门管理员权限  
 *document*.addEventListener('DOMContentLoaded', function() {  
 var roleSelect = *document*.querySelector('select[name="role"]');  
 var permsDiv = *document*.getElementById('deptAdminPerms');  
 if (roleSelect && !roleSelect.disabled) {  
 roleSelect.addEventListener('change', function() {  
 if (this.value === 'DEPT\_ADMIN') {  
 permsDiv.style.display = 'block';  
 } else {  
 permsDiv.style.display = 'none';  
 }  
 });  
 }  
 });  
 </script>  
 <% if (request.getAttribute("error") != null) { %>  
 <div class="error">错误：<%=request.getAttribute("error")%></div>  
 <% } %>  
 <% if (request.getAttribute("message") != null) { %>  
 <div class="message">提示：<%=request.getAttribute("message")%></div>  
 <% } %>  
 <button type="submit" class="btn">保存</button>  
 <a href="<%=request.getContextPath()%>/admin/users" class="btn btn-cancel">取消</a>  
 </form>  
 </div>  
</body>  
</html>

#### src/main/webapp/admin/index.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **page** import="com.zjut.passcode.bean.Admin" %>  
<%  
 // Check if admin is logged in  
 Admin admin = (Admin) session.getAttribute("admin");  
 if (admin == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
%>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>管理员控制台 - 校园通行码系统</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
   
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 }  
   
 .header {  
 background: white;  
 box-shadow: 0 2px 10px rgba(0,0,0,0.1);  
 padding: 20px;  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 }  
   
 .logo {  
 font-size: 1.5em;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .user-info {  
 display: flex;  
 align-items: center;  
 gap: 20px;  
 }  
   
 .user-name {  
 color: #333;  
 font-weight: 500;  
 }  
   
 .logout-btn {  
 background: #dc3545;  
 color: white;  
 padding: 8px 16px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 text-decoration: none;  
 font-size: 0.9em;  
 }  
   
 .logout-btn:hover {  
 background: #c82333;  
 }  
   
 .container {  
 max-width: 1200px;  
 margin: 40px auto;  
 padding: 0 20px;  
 }  
   
 .welcome-card {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 margin-bottom: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 text-align: center;  
 }  
   
 .welcome-title {  
 font-size: 2em;  
 color: #333;  
 margin-bottom: 10px;  
 }  
   
 .welcome-subtitle {  
 color: #666;  
 font-size: 1.1em;  
 }  
   
 .stats-grid {  
 display: grid;  
 grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));  
 gap: 20px;  
 margin-bottom: 30px;  
 }  
   
 .stat-card {  
 background: white;  
 border-radius: 15px;  
 padding: 25px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 text-align: center;  
 transition: transform 0.3s ease;  
 }  
   
 .stat-card:hover {  
 transform: translateY(-5px);  
 }  
   
 .stat-icon {  
 font-size: 2.5em;  
 margin-bottom: 15px;  
 }  
   
 .stat-number {  
 font-size: 2em;  
 font-weight: bold;  
 color: #667eea;  
 margin-bottom: 10px;  
 }  
   
 .stat-label {  
 color: #666;  
 font-size: 1em;  
 }  
   
 .actions-grid {  
 display: grid;  
 grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));  
 gap: 20px;  
 }  
   
 .action-card {  
 background: white;  
 border-radius: 15px;  
 padding: 25px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 text-align: center;  
 transition: all 0.3s ease;  
 cursor: pointer;  
 text-decoration: none;  
 color: inherit;  
 }  
   
 .action-card:hover {  
 transform: translateY(-5px);  
 box-shadow: 0 15px 40px rgba(0,0,0,0.15);  
 }  
   
 .action-icon {  
 font-size: 3em;  
 margin-bottom: 15px;  
 }  
   
 .action-title {  
 font-size: 1.3em;  
 font-weight: bold;  
 color: #333;  
 margin-bottom: 10px;  
 }  
   
 .action-description {  
 color: #666;  
 font-size: 0.95em;  
 line-height: 1.5;  
 }  
   
 .role-badge {  
 background: #667eea;  
 color: white;  
 padding: 5px 12px;  
 border-radius: 15px;  
 font-size: 0.8em;  
 margin-left: 10px;  
 }  
   
 @media (max-width: 768px) {  
 .header {  
 flex-direction: column;  
 gap: 15px;  
 }  
   
 .user-info {  
 flex-direction: column;  
 gap: 10px;  
 }  
   
 .welcome-title {  
 font-size: 1.5em;  
 }  
   
 .stats-grid {  
 grid-template-columns: 1fr;  
 }  
   
 .actions-grid {  
 grid-template-columns: 1fr;  
 }  
 }  
 </style>  
</head>  
<body>  
 <%-- 管理员控制台首页，显示欢迎信息和快捷操作 --%>  
 <%-- 检查管理员是否已登录，未登录则重定向 --%>  
 <div class="header">  
 <div class="logo">🏫 校园通行码管理系统</div>  
 <div class="user-info">  
 <div class="user-name">  
 欢迎，<%= admin.getFullName() %>  
 <span class="role-badge"><%= getRoleDisplayName(admin.getRole()) %></span>  
 </div>  
 <a href="${pageContext.request.contextPath}/admin/logout" class="logout-btn">🚪 退出登录</a>  
 </div>  
 </div>  
   
 <div class="container">  
 <div class="welcome-card">  
 <div class="welcome-title">欢迎使用管理员控制台</div>  
 <div class="welcome-subtitle">  
 当前时间：<span id="current-time"></span> |   
 登录时间：<%= session.getAttribute("loginTime") != null ? session.getAttribute("loginTime") : "未知" %>  
 </div>  
 </div>  
   
 <div class="stats-grid">  
 </div>  
   
 <div class="actions-grid">  
 <% if ("SYSTEM\_ADMIN".equals(admin.getRole())) { %>  
 <a href="${pageContext.request.contextPath}/admin/users" class="action-card">  
 <div class="action-icon">👥</div>  
 <div class="action-title">用户管理</div>  
 <div class="action-description">管理系统用户和管理员账户</div>  
 </a>  
 <% } else if ("AUDIT\_ADMIN".equals(admin.getRole())) { %>  
 <a href="${pageContext.request.contextPath}/admin/audit\_logs" class="action-card">  
 <div class="action-icon">📊</div>  
 <div class="action-title">审计日志</div>  
 <div class="action-description">查看系统操作和登录日志</div>  
 </a>  
 <% } else if ("SCHOOL\_ADMIN".equals(admin.getRole())) { %>  
 <a href="${pageContext.request.contextPath}/admin/users" class="action-card">  
 <div class="action-icon">👥</div>  
 <div class="action-title">用户管理</div>  
 <div class="action-description">仅管理部门管理员账户</div>  
 </a>  
 <a href="${pageContext.request.contextPath}/admin/appointments" class="action-card">  
 <div class="action-icon">📋</div>  
 <div class="action-title">预约管理</div>  
 <div class="action-description">审核社会公众预约和公务预约</div>  
 </a>  
 <a href="${pageContext.request.contextPath}/admin/departments" class="action-card">  
 <div class="action-icon">🏢</div>  
 <div class="action-title">部门管理</div>  
 <div class="action-description">管理所有学校部门和机构信息</div>  
 </a>  
 <a href="${pageContext.request.contextPath}/admin/reports" class="action-card">  
 <div class="action-icon">📈</div>  
 <div class="action-title">统计报告</div>  
 <div class="action-description">查看系统使用统计和报告</div>  
 </a>  
 <% } else if ("DEPT\_ADMIN".equals(admin.getRole())) { %>  
 <a href="${pageContext.request.contextPath}/admin/appointments" class="action-card">  
 <div class="action-icon">📋</div>  
 <div class="action-title">预约管理</div>  
 <div class="action-description">  
 <% if (admin.isCanManagePublicAppointment()) { %>  
 管理本部门公务预约和社会公众预约  
 <% } else { %>  
 仅能管理本部门公务预约  
 <% } %>  
 </div>  
 </a>  
 <a href="${pageContext.request.contextPath}/admin/reports" class="action-card">  
 <div class="action-icon">📈</div>  
 <div class="action-title">统计报告</div>  
 <div class="action-description">  
 <% if (admin.isCanReportPublicAppointment()) { %>  
 查看本部门公务预约和社会公众预约统计  
 <% } else { %>  
 查看本部门公务预约统计  
 <% } %>  
 </div>  
 </a>  
 <% } else { %>  
 <a href="${pageContext.request.contextPath}/admin/appointments" class="action-card">  
 <div class="action-icon">📋</div>  
 <div class="action-title">预约管理</div>  
 <div class="action-description">查看、审核和管理访客预约申请</div>  
 </a>  
 <a href="${pageContext.request.contextPath}/admin/users" class="action-card">  
 <div class="action-icon">👥</div>  
 <div class="action-title">用户管理</div>  
 <div class="action-description">管理系统用户和管理员账户</div>  
 </a>  
 <a href="${pageContext.request.contextPath}/admin/departments" class="action-card">  
 <div class="action-icon">🏢</div>  
 <div class="action-title">部门管理</div>  
 <div class="action-description">管理学校部门和机构信息</div>  
 </a>  
 <a href="${pageContext.request.contextPath}/admin/audit\_logs" class="action-card">  
 <div class="action-icon">📊</div>  
 <div class="action-title">审计日志</div>  
 <div class="action-description">查看系统操作和登录日志</div>  
 </a>  
 <a href="${pageContext.request.contextPath}/admin/reports" class="action-card">  
 <div class="action-icon">📈</div>  
 <div class="action-title">统计报告</div>  
 <div class="action-description">查看系统使用统计和报告</div>  
 </a>  
 <% } %>  
 </div>  
 </div>  
   
 <script>  
 // 实时更新时间显示  
 function updateTime() {  
 const now = new *Date*();  
 const timeString = now.toLocaleString('zh-CN');  
 *document*.getElementById('current-time').textContent = timeString;  
 }  
   
 // Update time every second  
 updateTime();  
 setInterval(updateTime, 1000);  
   
 // Auto logout after 30 minutes of inactivity  
 let *inactivityTimer*;  
 function resetInactivityTimer() {  
 clearTimeout(*inactivityTimer*);  
 *inactivityTimer* = setTimeout(() => {  
 alert('登录超时，即将退出系统');  
 *window*.*location*.href = '${pageContext.request.contextPath}/admin/login';  
 }, 30 \* 60 \* 1000); // 30 minutes  
 }  
   
 // Reset timer on user activity  
 *document*.addEventListener('mousemove', resetInactivityTimer);  
 *document*.addEventListener('keypress', resetInactivityTimer);  
 *document*.addEventListener('click', resetInactivityTimer);  
   
 // Start timer  
 resetInactivityTimer();  
 </script>  
   
 <%!  
 private String getRoleDisplayName(String role) {  
 switch (role) {  
 case "SYSTEM\_ADMIN":  
 return "系统管理员";  
 case "SCHOOL\_ADMIN":  
 return "学校管理员";  
 case "DEPT\_ADMIN":  
 return "部门管理员";  
 case "AUDIT\_ADMIN":  
 return "审计管理员";  
 default:  
 return "管理员";  
 }  
 }  
 %>  
</body>  
</html>

#### src/main/webapp/admin/reports.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **page** import="com.zjut.passcode.bean.Admin" %>  
<%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<%  
 // Check if admin is logged in  
 Admin admin = (Admin) session.getAttribute("admin");  
 if (admin == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
%>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>统计报告 - 校园通行码系统</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
   
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 }  
   
 .header {  
 background: white;  
 box-shadow: 0 2px 10px rgba(0,0,0,0.1);  
 padding: 20px;  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 }  
   
 .logo {  
 font-size: 1.5em;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .user-info {  
 display: flex;  
 align-items: center;  
 gap: 20px;  
 }  
   
 .user-name {  
 color: #333;  
 font-weight: 500;  
 }  
   
 .back-btn {  
 background: #6c757d;  
 color: white;  
 padding: 8px 16px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 text-decoration: none;  
 font-size: 0.9em;  
 }  
   
 .back-btn:hover {  
 background: #5a6268;  
 }  
   
 .container {  
 max-width: 1200px;  
 margin: 40px auto;  
 padding: 0 20px;  
 }  
   
 .page-title {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 margin-bottom: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 text-align: center;  
 }  
   
 .title {  
 font-size: 2em;  
 color: #333;  
 margin-bottom: 10px;  
 }  
   
 .subtitle {  
 color: #666;  
 font-size: 1.1em;  
 }  
   
 .reports-content {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 }  
   
 .report-section {  
 margin-bottom: 30px;  
 padding: 20px;  
 border: 1px solid #e9ecef;  
 border-radius: 10px;  
 }  
   
 .section-title {  
 font-size: 1.3em;  
 color: #333;  
 margin-bottom: 15px;  
 font-weight: bold;  
 }  
   
 .report-table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 15px;  
 }  
   
 .report-table th,  
 .report-table td {  
 padding: 12px;  
 text-align: left;  
 border-bottom: 1px solid #e9ecef;  
 }  
   
 .report-table th {  
 background: #f8f9fa;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .report-table tr:hover {  
 background: #f8f9fa;  
 }  
   
 .export-btn {  
 background: #28a745;  
 color: white;  
 padding: 10px 20px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 font-size: 1em;  
 margin-top: 20px;  
 text-decoration: none;  
 display: inline-block;  
 }  
   
 .export-btn:hover {  
 background: #218838;  
 }  
   
 .empty-state {  
 text-align: center;  
 padding: 40px;  
 color: #666;  
 }  
   
 .empty-state h3 {  
 margin-bottom: 10px;  
 color: #333;  
 }  
   
 @media (max-width: 768px) {  
 .header {  
 flex-direction: column;  
 gap: 15px;  
 }  
   
 .user-info {  
 flex-direction: column;  
 gap: 10px;  
 }  
   
 .title {  
 font-size: 1.5em;  
 }  
   
 .report-table {  
 font-size: 0.9em;  
 }  
   
 .report-table th,  
 .report-table td {  
 padding: 8px 4px;  
 }  
 }  
 </style>  
</head>  
<body>  
 <div class="header">  
 <div class="logo">📈 统计报告系统</div>  
 <div class="user-info">  
 <div class="user-name">  
 管理员：<%= admin.getFullName() %>  
 </div>  
 <a href="${pageContext.request.contextPath}/admin/index.jsp" class="back-btn">🔙 返回控制台</a>  
 </div>  
 </div>  
   
 <div class="container">  
 <div class="page-title">  
 <div class="title">统计报告</div>  
 <div class="subtitle">查看系统使用统计和报告</div>  
 </div>  
   
 <div class="reports-content">  
 <!-- 申请月度统计 -->  
 <div class="report-section">  
 <div class="section-title">按申请月度统计</div>  
 <table class="report-table">  
 <thead>  
 <tr>  
 <th>申请月份</th>  
 <th>预约次数</th>  
 <th>预约人次</th>  
 </tr>  
 </thead>  
 <tbody>  
 <**c:forEach** var="row" items="${applyMonthStats}">  
 <tr>  
 <td>${row.month}</td>  
 <td>${row.count}</td>  
 <td>${row.personCount}</td>  
 </tr>  
 </**c:forEach**>  
 <**c:if** test="${empty applyMonthStats}">  
 <tr><td colspan="3" style="text-align:center;color:#666;">暂无数据</td></tr>  
 </**c:if**>  
 </tbody>  
 </table>  
 </div>  
 <!-- 预约月度统计 -->  
 <div class="report-section">  
 <div class="section-title">按预约月度统计</div>  
 <table class="report-table">  
 <thead>  
 <tr>  
 <th>预约月份</th>  
 <th>预约次数</th>  
 <th>预约人次</th>  
 </tr>  
 </thead>  
 <tbody>  
 <**c:forEach** var="row" items="${entryMonthStats}">  
 <tr>  
 <td>${row.month}</td>  
 <td>${row.count}</td>  
 <td>${row.personCount}</td>  
 </tr>  
 </**c:forEach**>  
 <**c:if** test="${empty entryMonthStats}">  
 <tr><td colspan="3" style="text-align:center;color:#666;">暂无数据</td></tr>  
 </**c:if**>  
 </tbody>  
 </table>  
 </div>  
 <!-- 校区统计 -->  
 <div class="report-section">  
 <div class="section-title">按校区统计</div>  
 <table class="report-table">  
 <thead>  
 <tr>  
 <th>校区</th>  
 <th>预约次数</th>  
 <th>预约人次</th>  
 </tr>  
 </thead>  
 <tbody>  
 <**c:forEach** var="row" items="${campusStats}">  
 <tr>  
 <td>${row.campus}</td>  
 <td>${row.count}</td>  
 <td>${row.personCount}</td>  
 </tr>  
 </**c:forEach**>  
 <**c:if** test="${empty campusStats}">  
 <tr><td colspan="3" style="text-align:center;color:#666;">暂无数据</td></tr>  
 </**c:if**>  
 </tbody>  
 </table>  
 </div>  
 <!-- 公务访问部门统计 -->  
 <div class="report-section">  
 <div class="section-title">按公务访问部门统计</div>  
 <table class="report-table">  
 <thead>  
 <tr>  
 <th>部门</th>  
 <th>预约次数</th>  
 <th>预约人次</th>  
 </tr>  
 </thead>  
 <tbody>  
 <**c:forEach** var="row" items="${deptStats}">  
 <tr>  
 <td>${row.deptName}</td>  
 <td>${row.count}</td>  
 <td>${row.personCount}</td>  
 </tr>  
 </**c:forEach**>  
 <**c:if** test="${empty deptStats}">  
 <tr><td colspan="3" style="text-align:center;color:#666;">暂无数据</td></tr>  
 </**c:if**>  
 </tbody>  
 </table>  
 </div>  
 </div>  
 </div>  
</body>  
</html>

#### src/main/webapp/admin/reset\_password.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **page** import="com.zjut.passcode.bean.Admin" %>  
<%  
 Admin user = (Admin) request.getAttribute("resetUser");  
%>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>重置管理员密码 - 校园通行码系统</title>  
 <style>  
 body { font-family: 'Microsoft YaHei', Arial, sans-serif; background: linear-gradient(135deg, #667eea 0%, #764ba2 100%); min-height: 100vh; }  
 .container { max-width: 500px; margin: 40px auto; background: #fff; border-radius: 15px; box-shadow: 0 10px 30px rgba(0,0,0,0.1); padding: 40px; }  
 h2 { text-align: center; margin-bottom: 30px; }  
 label { display: block; margin-bottom: 8px; color: #333; }  
 input { width: 100%; padding: 10px; margin-bottom: 20px; border-radius: 5px; border: 1px solid #ccc; }  
 .btn { background: #17a2b8; color: #fff; border: none; padding: 10px 30px; border-radius: 5px; cursor: pointer; font-size: 1em; }  
 .btn:disabled { background: #6c757d; cursor: not-allowed; }  
 .btn-cancel { background: #6c757d; margin-left: 10px; }  
 .error { color: #dc3545; margin-bottom: 10px; }  
 .message { color: #28a745; margin-bottom: 10px; }  
 .password-hint { color:#888; font-size:0.9em; margin-bottom: 10px; }  
 .password-status { font-size: 0.9em; margin-bottom: 10px; }  
 .password-match { color: #28a745; }  
 .password-mismatch { color: #dc3545; }  
 </style>  
</head>  
<body>  
 <div class="container">  
 <h2>重置管理员密码</h2>  
 <form method="post" action="<%=request.getContextPath()%>/admin/users" id="resetPasswordForm">  
 <input type="hidden" name="action" value="resetPwd" />  
 <input type="hidden" name="userId" value="<%= user.getId() %>" />  
 <div>  
 <label>管理员姓名：</label>  
 <input type="text" value="<%= user.getFullName() %>" disabled />  
 </div>  
 <div>  
 <label>新密码：</label>  
 <input type="password" name="newPassword" id="newPassword" required />  
 <div class="password-hint">密码需8位以上，含数字、大小写字母、特殊字符</div>  
 </div>  
 <div>  
 <label>确认密码：</label>  
 <input type="password" name="confirmPassword" id="confirmPassword" required />  
 <div class="password-hint">请再次输入新密码</div>  
 <div class="password-status" id="passwordStatus"></div>  
 </div>  
 <% if (request.getAttribute("error") != null) { %>  
 <div class="error">错误：<%=request.getAttribute("error")%></div>  
 <% } %>  
 <% if (request.getAttribute("message") != null) { %>  
 <div class="message">提示：<%=request.getAttribute("message")%></div>  
 <% } %>  
 <button type="submit" class="btn" id="submitBtn" disabled>重置密码</button>  
 <a href="<%=request.getContextPath()%>/admin/users" class="btn btn-cancel">取消</a>  
 </form>  
 </div>  
  
 <script>  
 const *newPassword* = *document*.getElementById('newPassword');  
 const *confirmPassword* = *document*.getElementById('confirmPassword');  
 const *passwordStatus* = *document*.getElementById('passwordStatus');  
 const *submitBtn* = *document*.getElementById('submitBtn');  
  
 // 检查密码复杂度  
 function validatePassword(password) {  
 const hasNumber = /\d/.test(password);  
 const hasLower = /[a-z]/.test(password);  
 const hasUpper = /[A-Z]/.test(password);  
 const hasSpecial = /[!@#$%^&\*(),.?":{}|<>]/.test(password);  
 const isLongEnough = password.length >= 8;  
   
 return hasNumber && hasLower && hasUpper && hasSpecial && isLongEnough;  
 }  
  
 // 实时检查密码匹配和复杂度  
 function checkPasswords() {  
 const newPwd = *newPassword*.value;  
 const confirmPwd = *confirmPassword*.value;  
   
 // 如果两个密码框都为空，不显示状态  
 if (!newPwd && !confirmPwd) {  
 *passwordStatus*.textContent = '';  
 *passwordStatus*.className = 'password-status';  
 *submitBtn*.disabled = true;  
 return;  
 }  
   
 // 检查密码复杂度  
 if (newPwd && !validatePassword(newPwd)) {  
 *passwordStatus*.textContent = '密码复杂度不足';  
 *passwordStatus*.className = 'password-status password-mismatch';  
 *submitBtn*.disabled = true;  
 return;  
 }  
   
 // 检查密码匹配  
 if (newPwd && confirmPwd) {  
 if (newPwd === confirmPwd) {  
 *passwordStatus*.textContent = '✓ 密码匹配且符合要求';  
 *passwordStatus*.className = 'password-status password-match';  
 *submitBtn*.disabled = false;  
 } else {  
 *passwordStatus*.textContent = '✗ 两次输入的密码不一致';  
 *passwordStatus*.className = 'password-status password-mismatch';  
 *submitBtn*.disabled = true;  
 }  
 } else {  
 *passwordStatus*.textContent = '请完成密码输入';  
 *passwordStatus*.className = 'password-status password-mismatch';  
 *submitBtn*.disabled = true;  
 }  
 }  
  
 // 添加事件监听器  
 *newPassword*.addEventListener('input', checkPasswords);  
 *confirmPassword*.addEventListener('input', checkPasswords);  
  
 // 表单提交验证（双重保险）  
 *document*.getElementById('resetPasswordForm').addEventListener('submit', function(e) {  
 const newPwd = *newPassword*.value;  
 const confirmPwd = *confirmPassword*.value;  
   
 if (newPwd !== confirmPwd) {  
 e.preventDefault();  
 alert('两次输入的密码不一致，请重新输入');  
 return false;  
 }  
   
 if (!validatePassword(newPwd)) {  
 e.preventDefault();  
 alert('密码必须包含8位以上，包含数字、大小写字母、特殊字符');  
 return false;  
 }  
 });  
 </script>  
</body>  
</html>

#### src/main/webapp/admin/users.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **page** import="com.zjut.passcode.bean.Admin" %>  
<%@ **page** import="java.util.List" %>  
<%  
 // Check if admin is logged in  
 Admin admin = (Admin) session.getAttribute("admin");  
 if (admin == null) {  
 response.sendRedirect(request.getContextPath() + "/admin/login");  
 return;  
 }  
   
 List<Admin> admins = (List<Admin>) request.getAttribute("admins");  
 Admin currentAdmin = (Admin) request.getAttribute("currentAdmin");  
   
 String action = request.getParameter("action");  
%>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>管理员管理 - 校园通行码系统</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
   
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 }  
   
 .header {  
 background: white;  
 box-shadow: 0 2px 10px rgba(0,0,0,0.1);  
 padding: 20px;  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 }  
   
 .logo {  
 font-size: 1.5em;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .user-info {  
 display: flex;  
 align-items: center;  
 gap: 20px;  
 }  
   
 .user-name {  
 color: #333;  
 font-weight: 500;  
 }  
   
 .back-btn {  
 background: #6c757d;  
 color: white;  
 padding: 8px 16px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 text-decoration: none;  
 font-size: 0.9em;  
 }  
   
 .back-btn:hover {  
 background: #5a6268;  
 }  
   
 .container {  
 max-width: 1200px;  
 margin: 40px auto;  
 padding: 0 20px;  
 }  
   
 .page-title {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 margin-bottom: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 text-align: center;  
 }  
   
 .title {  
 font-size: 2em;  
 color: #333;  
 margin-bottom: 10px;  
 }  
   
 .subtitle {  
 color: #666;  
 font-size: 1.1em;  
 }  
   
 .users-content {  
 background: white;  
 border-radius: 15px;  
 padding: 30px;  
 box-shadow: 0 10px 30px rgba(0,0,0,0.1);  
 }  
   
 .add-user-btn {  
 background: #28a745;  
 color: white;  
 padding: 10px 20px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 font-size: 1em;  
 margin-bottom: 20px;  
 text-decoration: none;  
 display: inline-block;  
 }  
   
 .add-user-btn:hover {  
 background: #218838;  
 }  
   
 .users-table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 20px;  
 }  
   
 .users-table th,  
 .users-table td {  
 padding: 12px;  
 text-align: left;  
 border-bottom: 1px solid #e9ecef;  
 }  
   
 .users-table th {  
 background: #f8f9fa;  
 font-weight: bold;  
 color: #333;  
 }  
   
 .users-table tr:hover {  
 background: #f8f9fa;  
 }  
   
 .role-badge {  
 display: inline-block;  
 padding: 4px 8px;  
 border-radius: 12px;  
 font-size: 0.8em;  
 font-weight: bold;  
 }  
   
 .role-system-admin {  
 background: #dc3545;  
 color: white;  
 }  
   
 .role-school-admin {  
 background: #fd7e14;  
 color: white;  
 }  
   
 .role-dept-admin {  
 background: #ffc107;  
 color: #212529;  
 }  
   
 .role-audit-admin {  
 background: #17a2b8;  
 color: white;  
 }  
   
 .action-btn {  
 padding: 6px 12px;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
 font-size: 0.8em;  
 margin: 2px;  
 text-decoration: none;  
 display: inline-block;  
 }  
   
 .btn-edit {  
 background: #17a2b8;  
 color: white;  
 }  
   
 .btn-delete {  
 background: #dc3545;  
 color: white;  
 }  
   
 .btn-reset-pwd {  
 background: #6c757d;  
 color: white;  
 }  
   
 .empty-state {  
 text-align: center;  
 padding: 40px;  
 color: #666;  
 }  
   
 .empty-state h3 {  
 margin-bottom: 10px;  
 color: #333;  
 }  
   
 @media (max-width: 768px) {  
 .header {  
 flex-direction: column;  
 gap: 15px;  
 }  
   
 .user-info {  
 flex-direction: column;  
 gap: 10px;  
 }  
   
 .title {  
 font-size: 1.5em;  
 }  
   
 .users-table {  
 font-size: 0.9em;  
 }  
   
 .users-table th,  
 .users-table td {  
 padding: 8px 4px;  
 }  
 }  
 </style>  
</head>  
<body>  
 <div class="header">  
 <div class="logo">👤 管理员管理系统</div>  
 <div class="user-info">  
 <div class="user-name">  
 管理员：<%= currentAdmin.getFullName() %>  
 </div>  
 <a href="${pageContext.request.contextPath}/admin/index.jsp" class="back-btn">🔙 返回控制台</a>  
 </div>  
 </div>  
   
 <div class="container">  
 <div class="page-title">  
 <div class="title">管理员管理</div>  
 <div class="subtitle">管理学校和部门管理员信息</div>  
 </div>  
   
 <div class="users-content">  
 <% if ("showAddForm".equals(action)) { %>  
 <h2 style="margin-bottom:20px;">添加新管理员</h2>  
 <form method="post" action="<%=request.getContextPath()%>/admin/users">  
 <input type="hidden" name="action" value="add" />  
 <div style="margin-bottom:15px;">  
 <label>登录名：</label>  
 <input type="text" name="loginName" required style="padding:6px;width:220px;" />  
 </div>  
 <div style="margin-bottom:15px;">  
 <label>密码：</label>  
 <input type="password" name="password" required style="padding:6px;width:220px;" />  
 <span style="color:#888;font-size:0.9em;">密码需8位以上，含数字、大小写字母、特殊字符</span>  
 </div>  
 <div style="margin-bottom:15px;">  
 <label>姓名：</label>  
 <input type="text" name="fullName" required style="padding:6px;width:220px;" />  
 </div>  
 <div style="margin-bottom:15px;">  
 <label>部门：</label>  
 <select name="deptId" style="padding:6px;width:230px;">  
 <option value="">请选择部门</option>  
 <%   
 List departments = (List)request.getAttribute("departments");  
 if (departments != null) {  
 for (Object obj : departments) {  
 com.zjut.passcode.bean.Department dept = (com.zjut.passcode.bean.Department)obj;  
 %>  
 <option value="<%=dept.getId()%>"><%=dept.getDeptName()%></option>  
 <% }} %>  
 </select>  
 </div>  
 <div style="margin-bottom:15px;">  
 <label>联系电话：</label>  
 <input type="text" name="phone" required style="padding:6px;width:220px;" />  
 </div>  
 <div style="margin-bottom:15px;">  
 <label>角色：</label>  
 <select name="role" id="addRoleSelect" required style="padding:6px;width:230px;">  
 <% if ("SYSTEM\_ADMIN".equals(currentAdmin.getRole())) { %>  
 <option value="SYSTEM\_ADMIN">系统管理员</option>  
 <option value="SCHOOL\_ADMIN">学校管理员</option>  
 <option value="DEPT\_ADMIN">部门管理员</option>  
 <option value="AUDIT\_ADMIN">审计管理员</option>  
 <% } else if ("SCHOOL\_ADMIN".equals(currentAdmin.getRole())) { %>  
 <option value="DEPT\_ADMIN">部门管理员</option>  
 <% } %>  
 </select>  
 </div>  
 <div id="addDeptAdminPerms" style="display:none;margin-bottom:15px;">  
 <label style="font-weight:bold;">部门管理员权限：</label>  
 <div style="margin:8px 0 0 10px;">  
 <input type="checkbox" name="canManagePublicAppointment" value="1" />  
 <span>可管理社会公众预约（默认只能管理公务预约）</span>  
 </div>  
 <div style="margin:8px 0 0 10px;">  
 <input type="checkbox" name="canReportPublicAppointment" value="1" />  
 <span>可统计社会公众预约（默认只能统计公务预约）</span>  
 </div>  
 </div>  
 <script>  
 // 动态显示/隐藏部门管理员权限  
 *document*.addEventListener('DOMContentLoaded', function() {  
 var roleSelect = *document*.getElementById('addRoleSelect');  
 var permsDiv = *document*.getElementById('addDeptAdminPerms');  
 roleSelect.addEventListener('change', function() {  
 if (this.value === 'DEPT\_ADMIN') {  
 permsDiv.style.display = 'block';  
 } else {  
 permsDiv.style.display = 'none';  
 }  
 });  
 });  
 </script>  
 <div style="margin-bottom:20px;">  
 <button type="submit" class="add-user-btn">提交</button>  
 <a href="<%=request.getContextPath()%>/admin/users" class="back-btn" style="margin-left:10px;">取消</a>  
 </div>  
 <% if (request.getAttribute("error") != null) { %>  
 <div style="color:#dc3545;margin-bottom:10px;">错误：<%=request.getAttribute("error")%></div>  
 <% } %>  
 <% if (request.getAttribute("message") != null) { %>  
 <div style="color:#28a745;margin-bottom:10px;">提示：<%=request.getAttribute("message")%></div>  
 <% } %>  
 </form>  
 <% } else { %>  
 <a href="${pageContext.request.contextPath}/admin/users?action=showAddForm" class="add-user-btn">➕ 添加新管理员</a>  
   
 <% if (admins != null && !admins.isEmpty()) { %>  
 <table class="users-table">  
 <thead>  
 <tr>  
 <th>ID</th>  
 <th>登录名</th>  
 <th>姓名</th>  
 <th>角色</th>  
 <th>部门</th>  
 <th>联系电话</th>  
 <th>最后登录</th>  
 <th>状态</th>  
 <th>操作</th>  
 </tr>  
 </thead>  
 <tbody>  
 <% for (Admin user : admins) { %>  
 <tr>  
 <td><%= user.getId() %></td>  
 <td><%= user.getLoginName() %></td>  
 <td><%= user.getFullName() %></td>  
 <td>  
 <span class="role-badge role-<%= user.getRole().toLowerCase().replace("\_", "-") %>">  
 <%= getRoleDisplayName(user.getRole()) %>  
 </span>  
 </td>  
 <td><%= user.getDeptName() != null ? user.getDeptName() : "未分配" %></td>  
 <td><%= user.getPhone() != null ? user.getPhone() : "未设置" %></td>  
 <td><%= user.getPasswordLastChanged() != null ? user.getPasswordLastChanged() : "从未登录" %></td>  
 <td>  
 <% if (user.getLockoutUntil() != null) { %>  
 <span style="color: #dc3545;">已锁定</span>  
 <% } else { %>  
 <span style="color: #28a745;">正常</span>  
 <% } %>  
 </td>  
 <td>  
 <form method="get" action="<%=request.getContextPath()%>/admin/users" style="display:inline;">  
 <input type="hidden" name="action" value="editUser" />  
 <input type="hidden" name="userId" value="<%= user.getId() %>" />  
 <button type="submit" class="action-btn btn-edit">编辑</button>  
 </form>  
 <form method="get" action="<%=request.getContextPath()%>/admin/users" style="display:inline;">  
 <input type="hidden" name="action" value="resetPwd" />  
 <input type="hidden" name="userId" value="<%= user.getId() %>" />  
 <button type="submit" class="action-btn btn-reset-pwd">重置密码</button>  
 </form>  
 <% if (user.getId() != currentAdmin.getId()) { %>  
 <form method="post" action="<%=request.getContextPath()%>/admin/users" style="display:inline;" onsubmit="return confirmDelete();">  
 <input type="hidden" name="action" value="delete" />  
 <input type="hidden" name="userId" value="<%= user.getId() %>" />  
 <button type="submit" class="action-btn btn-delete">删除</button>  
 </form>  
 <% } %>  
 </td>  
 </tr>  
 <% } %>  
 </tbody>  
 </table>  
 <% } else { %>  
 <div class="empty-state">  
 <h3>暂无用户记录</h3>  
 <p>当前系统中没有用户记录</p>  
 </div>  
 <% } %>  
 <% } %>  
 </div>  
 </div>  
   
 <%!  
 private String getRoleDisplayName(String role) {  
 switch (role) {  
 case "SYSTEM\_ADMIN":  
 return "系统管理员";  
 case "SCHOOL\_ADMIN":  
 return "学校管理员";  
 case "DEPT\_ADMIN":  
 return "部门管理员";  
 case "AUDIT\_ADMIN":  
 return "审计管理员";  
 default:  
 return role;  
 }  
 }  
 %>  
 <script>  
 function confirmDelete() {  
 return confirm('确定要删除该管理员吗？此操作不可恢复！');  
 }  
 </script>  
</body>  
</html>

#### src/main/webapp/admin\_login.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>管理员登录 - 校园通行码系统</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
   
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 padding: 20px;  
 }  
   
 .login-container {  
 background: white;  
 border-radius: 20px;  
 box-shadow: 0 20px 40px rgba(0,0,0,0.1);  
 padding: 40px;  
 max-width: 400px;  
 width: 100%;  
 }  
   
 .login-header {  
 text-align: center;  
 margin-bottom: 30px;  
 }  
   
 .login-icon {  
 font-size: 3em;  
 margin-bottom: 15px;  
 }  
   
 .login-title {  
 font-size: 1.8em;  
 color: #333;  
 margin-bottom: 10px;  
 font-weight: bold;  
 }  
   
 .login-subtitle {  
 color: #666;  
 font-size: 1em;  
 }  
   
 .form-group {  
 margin-bottom: 20px;  
 }  
   
 .form-group label {  
 display: block;  
 margin-bottom: 8px;  
 color: #333;  
 font-weight: 500;  
 }  
   
 .form-group input {  
 width: 100%;  
 padding: 15px;  
 border: 2px solid #e1e5e9;  
 border-radius: 10px;  
 font-size: 1em;  
 transition: border-color 0.3s ease;  
 }  
   
 .form-group input:focus {  
 outline: none;  
 border-color: #667eea;  
 }  
   
 .password-container {  
 position: relative;  
 }  
   
 .password-toggle {  
 position: absolute;  
 right: 15px;  
 top: 50%;  
 transform: translateY(-50%);  
 background: none;  
 border: none;  
 cursor: pointer;  
 font-size: 1.2em;  
 color: #666;  
 }  
   
 .password-toggle:hover {  
 color: #333;  
 }  
   
 .btn-login {  
 width: 100%;  
 padding: 15px;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 color: white;  
 border: none;  
 border-radius: 10px;  
 font-size: 1.1em;  
 font-weight: 500;  
 cursor: pointer;  
 transition: all 0.3s ease;  
 margin-bottom: 20px;  
 }  
   
 .btn-login:hover {  
 transform: translateY(-2px);  
 box-shadow: 0 10px 20px rgba(102, 126, 234, 0.3);  
 }  
   
 .btn-login:disabled {  
 opacity: 0.6;  
 cursor: not-allowed;  
 transform: none;  
 }  
   
 .back-link {  
 text-align: center;  
 }  
   
 .back-link a {  
 color: #667eea;  
 text-decoration: none;  
 font-size: 0.9em;  
 }  
   
 .back-link a:hover {  
 text-decoration: underline;  
 }  
   
 .message {  
 padding: 15px;  
 border-radius: 10px;  
 margin-bottom: 20px;  
 text-align: center;  
 font-weight: 500;  
 }  
   
 .message-error {  
 background: #f8d7da;  
 color: #721c24;  
 border: 1px solid #f5c6cb;  
 }  
   
 .message-warning {  
 background: #fff3cd;  
 color: #856404;  
 border: 1px solid #ffeaa7;  
 }  
   
 .security-info {  
 background: #e7f3ff;  
 color: #0c5460;  
 padding: 15px;  
 border-radius: 10px;  
 margin-top: 20px;  
 font-size: 0.9em;  
 border: 1px solid #bee5eb;  
 }  
   
 .security-info h4 {  
 margin-bottom: 10px;  
 font-size: 1em;  
 }  
   
 .security-info ul {  
 list-style: none;  
 padding-left: 0;  
 }  
   
 .security-info li {  
 margin-bottom: 5px;  
 padding-left: 20px;  
 position: relative;  
 }  
   
 .security-info li:before {  
 content: "🔒";  
 position: absolute;  
 left: 0;  
 }  
   
 @media (max-width: 480px) {  
 .login-container {  
 padding: 30px 20px;  
 }  
   
 .login-title {  
 font-size: 1.5em;  
 }  
   
 .login-icon {  
 font-size: 2.5em;  
 }  
 }  
 </style>  
</head>  
<body>  
 <div class="login-container">  
 <div class="login-header">  
 <div class="login-icon">🔐</div>  
 <div class="login-title">管理员登录</div>  
 <div class="login-subtitle">校园通行码管理系统</div>  
 </div>  
   
 <!-- 错误消息 -->  
 <% if (request.getAttribute("error") != null) { %>  
 <div class="message message-error">  
 ❌ <%= request.getAttribute("error") %>  
 <% if (request.getAttribute("errorDetails") != null) { %>  
 <div style="margin-top: 10px; font-size: 0.9em; font-weight: normal;">  
 📋 <%= request.getAttribute("errorDetails") %>  
 </div>  
 <% } %>  
 <% if (request.getAttribute("errorType") != null) { %>  
 <div style="margin-top: 5px; font-size: 0.85em; font-weight: normal; color: #721c24;">  
 🔍 错误类型: <%= getErrorTypeDescription((String)request.getAttribute("errorType")) %>  
 </div>  
 <% } %>  
 </div>  
 <% } %>  
   
 <!-- 警告消息 -->  
 <% if (request.getAttribute("warning") != null) { %>  
 <div class="message message-warning">  
 ⚠️ <%= request.getAttribute("warning") %>  
 <% if (request.getAttribute("warningDetails") != null) { %>  
 <div style="margin-top: 10px; font-size: 0.9em; font-weight: normal;">  
 📋 <%= request.getAttribute("warningDetails") %>  
 </div>  
 <% } %>  
 </div>  
 <% } %>  
   
 <form action="${pageContext.request.contextPath}/admin/login" method="post">  
 <div class="form-group">  
 <label for="loginName">用户名</label>  
 <input type="text" id="loginName" name="loginName" required   
 placeholder="请输入登录用户名">  
 </div>  
   
 <div class="form-group">  
 <label for="password">密码</label>  
 <div class="password-container">  
 <input type="password" id="password" name="password" required   
 placeholder="请输入密码">  
 <button type="button" class="password-toggle" onclick="togglePassword()">  
 👁️  
 </button>  
 </div>  
 </div>  
   
 <button type="submit" class="btn-login" id="loginBtn">  
 🔑 登录  
 </button>  
 </form>  
   
 <div class="back-link">  
 <a href="${pageContext.request.contextPath}/">← 返回首页</a>  
 </div>  
   
 <div class="security-info">  
 <h4>🔒 安全提示</h4>  
 <ul>  
 <li>密码长度至少8位，包含数字、大小写字母、特殊字符</li>  
 <li>连续登录失败5次将锁定账户30分钟</li>  
 <li>密码90天过期，请及时修改</li>  
 <li>登录超时30分钟自动退出</li>  
 </ul>  
 </div>  
 </div>  
   
 <script>  
 function togglePassword() {  
 const passwordInput = *document*.getElementById('password');  
 const toggleBtn = *document*.querySelector('.password-toggle');  
   
 if (passwordInput.type === 'password') {  
 passwordInput.type = 'text';  
 toggleBtn.textContent = '🙈';  
 } else {  
 passwordInput.type = 'password';  
 toggleBtn.textContent = '👁️';  
 }  
 }  
   
 // 表单提交时禁用按钮防止重复提交  
 *document*.querySelector('form').addEventListener('submit', function() {  
 const loginBtn = *document*.getElementById('loginBtn');  
 loginBtn.disabled = true;  
 loginBtn.textContent = '⏳ 登录中...';  
 });  
   
 // 自动聚焦到用户名输入框  
 *window*.onload = function() {  
 *document*.getElementById('loginName').focus();  
 };  
 </script>  
   
 <%!  
 private String getErrorTypeDescription(String errorType) {  
 switch (errorType) {  
 case "INPUT\_VALIDATION":  
 return "输入验证错误";  
 case "USER\_NOT\_FOUND":  
 return "用户不存在";  
 case "PASSWORD\_MISMATCH":  
 return "密码不匹配";  
 case "ACCOUNT\_LOCKED":  
 return "账户被锁定";  
 case "DATABASE\_CONNECTION":  
 return "数据库连接错误";  
 case "PASSWORD\_ENCRYPTION":  
 return "密码加密错误";  
 case "LOCKOUT\_ERROR":  
 return "账户锁定错误";  
 case "UNKNOWN\_ERROR":  
 return "未知错误";  
 default:  
 return "系统错误";  
 }  
 }  
 %>  
</body>  
</html>

#### src/main/webapp/appointment\_form.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>进校预约申请 - 校园通行码系统</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
   
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 padding: 20px;  
 }  
   
 .container {  
 max-width: 800px;  
 margin: 0 auto;  
 background: white;  
 border-radius: 15px;  
 box-shadow: 0 20px 40px rgba(0,0,0,0.1);  
 overflow: hidden;  
 }  
   
 .header {  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 color: white;  
 padding: 30px;  
 text-align: center;  
 }  
   
 .header h1 {  
 font-size: 2em;  
 margin-bottom: 10px;  
 }  
   
 .header p {  
 opacity: 0.9;  
 font-size: 1.1em;  
 }  
   
 .form-container {  
 padding: 40px;  
 }  
   
 .form-section {  
 margin-bottom: 30px;  
 }  
   
 .section-title {  
 font-size: 1.3em;  
 color: #333;  
 margin-bottom: 20px;  
 padding-bottom: 10px;  
 border-bottom: 2px solid #667eea;  
 }  
   
 .form-row {  
 display: flex;  
 gap: 20px;  
 margin-bottom: 20px;  
 }  
   
 .form-group {  
 flex: 1;  
 }  
   
 .form-group.full-width {  
 flex: 1;  
 }  
   
 .form-group label {  
 display: block;  
 margin-bottom: 8px;  
 color: #333;  
 font-weight: 500;  
 }  
   
 .form-group input,  
 .form-group select,  
 .form-group textarea {  
 width: 100%;  
 padding: 12px;  
 border: 2px solid #e1e5e9;  
 border-radius: 8px;  
 font-size: 1em;  
 transition: border-color 0.3s ease;  
 }  
   
 .form-group input:focus,  
 .form-group select:focus,  
 .form-group textarea:focus {  
 outline: none;  
 border-color: #667eea;  
 }  
   
 .form-group textarea {  
 resize: vertical;  
 min-height: 100px;  
 }  
   
 .required {  
 color: #e74c3c;  
 }  
   
 .accompanying-section {  
 background: #f8f9fa;  
 border-radius: 10px;  
 padding: 20px;  
 margin-top: 20px;  
 }  
   
 .accompanying-item {  
 border: 1px solid #dee2e6;  
 border-radius: 8px;  
 padding: 15px;  
 margin-bottom: 15px;  
 background: white;  
 }  
   
 .accompanying-header {  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 margin-bottom: 15px;  
 }  
   
 .remove-accompanying {  
 background: #dc3545;  
 color: white;  
 border: none;  
 padding: 5px 10px;  
 border-radius: 5px;  
 cursor: pointer;  
 font-size: 0.9em;  
 }  
   
 .add-accompanying {  
 background: #28a745;  
 color: white;  
 border: none;  
 padding: 10px 20px;  
 border-radius: 8px;  
 cursor: pointer;  
 font-size: 1em;  
 margin-top: 10px;  
 }  
   
 .btn-group {  
 display: flex;  
 gap: 15px;  
 justify-content: center;  
 margin-top: 30px;  
 }  
   
 .btn {  
 padding: 15px 30px;  
 border: none;  
 border-radius: 8px;  
 font-size: 1.1em;  
 cursor: pointer;  
 text-decoration: none;  
 display: inline-block;  
 transition: all 0.3s ease;  
 font-weight: 500;  
 }  
   
 .btn-primary {  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 color: white;  
 }  
   
 .btn-primary:hover {  
 transform: translateY(-2px);  
 box-shadow: 0 10px 20px rgba(102, 126, 234, 0.3);  
 }  
   
 .btn-secondary {  
 background: #6c757d;  
 color: white;  
 }  
   
 .btn-secondary:hover {  
 background: #5a6268;  
 transform: translateY(-2px);  
 }  
   
 .error {  
 background: #f8d7da;  
 color: #721c24;  
 padding: 15px;  
 border-radius: 8px;  
 margin-bottom: 20px;  
 border: 1px solid #f5c6cb;  
 }  
   
 .official-fields {  
 display: none;  
 }  
   
 .official-fields.show {  
 display: block;  
 }  
   
 @media (max-width: 768px) {  
 .form-row {  
 flex-direction: column;  
 gap: 15px;  
 }  
   
 .form-container {  
 padding: 20px;  
 }  
   
 .header {  
 padding: 20px;  
 }  
   
 .header h1 {  
 font-size: 1.5em;  
 }  
 }  
 </style>  
</head>  
<body>  
 <%-- 预约申请页面，用户填写进校预约信息 --%>  
 <div class="container">  
 <div class="header">  
 <h1>📝 进校预约申请</h1>  
 <p>请如实填写以下信息，带\*为必填项</p>  
 </div>  
   
 <div class="form-container">  
 <% if (request.getAttribute("error") != null) { %>  
 <div class="error">  
 <%= request.getAttribute("error") %>  
 </div>  
 <% } %>  
   
 <form action="${pageContext.request.contextPath}/appointment/apply" method="post">  
 <!-- 基本信息 -->  
 <div class="form-section">  
 <div class="section-title">📋 基本信息</div>  
 <div class="form-row">  
 <div class="form-group">  
 <label>预约人姓名 <span class="required">\*</span></label>  
 <input type="text" name="visitorName" required>  
 </div>  
 <div class="form-group">  
 <label>身份证号 <span class="required">\*</span></label>  
 <input type="text" name="visitorIdCard" required pattern="[0-9Xx]{18}">  
 </div>  
 </div>  
 <div class="form-row">  
 <div class="form-group">  
 <label>手机号 <span class="required">\*</span></label>  
 <input type="tel" name="visitorPhone" required pattern="[0-9]{11}">  
 </div>  
 <div class="form-group">  
 <label>所在单位</label>  
 <input type="text" name="visitorUnit">  
 </div>  
 </div>  
 </div>  
   
 <!-- 预约信息 -->  
 <div class="form-section">  
 <div class="section-title">📅 预约信息</div>  
 <div class="form-row">  
 <div class="form-group">  
 <label>预约校区 <span class="required">\*</span></label>  
 <select name="campus" required>  
 <option value="">请选择校区</option>  
 <option value="朝晖校区">朝晖校区</option>  
 <option value="屏峰校区">屏峰校区</option>  
 <option value="莫干山校区">莫干山校区</option>  
 </select>  
 </div>  
 <div class="form-group">  
 <label>预约进校时间 <span class="required">\*</span></label>  
 <input type="datetime-local" name="entryTime" required>  
 </div>  
 </div>  
 <div class="form-row">  
 <div class="form-group">  
 <label>交通方式</label>  
 <select name="transportMode">  
 <option value="">请选择交通方式</option>  
 <option value="步行">步行</option>  
 <option value="自行车">自行车</option>  
 <option value="电动车">电动车</option>  
 <option value="汽车">汽车</option>  
 <option value="公共交通">公共交通</option>  
 </select>  
 </div>  
 <div class="form-group">  
 <label>车牌号</label>  
 <input type="text" name="licensePlate" placeholder="如开车进校请填写">  
 </div>  
 </div>  
 </div>  
   
 <!-- 预约类型 -->  
 <div class="form-section">  
 <div class="section-title">🏢 预约类型</div>  
 <div class="form-row">  
 <div class="form-group full-width">  
 <label>预约类型 <span class="required">\*</span></label>  
 <select name="appointmentType" id="appointmentType" required onchange="toggleOfficialFields()">  
 <option value="">请选择预约类型</option>  
 <option value="PUBLIC">社会公众预约</option>  
 <option value="OFFICIAL">公务预约</option>  
 </select>  
 </div>  
 </div>  
   
 <!-- 公务预约特有字段 -->  
 <div class="official-fields" id="officialFields">  
 <div class="form-row">  
 <div class="form-group">  
 <label>公务访问部门 <span class="required">\*</span></label>  
 <select name="officialDeptId" required>  
 <option value="">请选择部门</option>  
 <option value="1">计算机学院</option>  
 <option value="2">机械工程学院</option>  
 <option value="3">化学工程学院</option>  
 <option value="4">信息工程学院</option>  
 <option value="5">经济学院</option>  
 <option value="6">管理学院</option>  
 <option value="7">理学院</option>  
 <option value="8">外国语学院</option>  
 <option value="9">人文学院</option>  
 <option value="10">艺术学院</option>  
 </select>  
 </div>  
 <div class="form-group">  
 <label>公务访问接待人 <span class="required">\*</span></label>  
 <input type="text" name="officialContactPerson" placeholder="接待老师姓名">  
 </div>  
 </div>  
 <div class="form-row">  
 <div class="form-group full-width">  
 <label>来访事由 <span class="required">\*</span></label>  
 <textarea name="officialReason" placeholder="请详细说明来访事由"></textarea>  
 </div>  
 </div>  
 </div>  
 </div>  
   
 <!-- 随行人员 -->  
 <div class="form-section">  
 <div class="section-title">👥 随行人员（可选）</div>  
 <div class="accompanying-section">  
 <div id="accompanyingList">  
 <!-- 随行人员列表将在这里动态生成 -->  
 </div>  
 <button type="button" class="add-accompanying" onclick="addAccompanying()">  
 ➕ 添加随行人员  
 </button>  
 </div>  
 </div>  
   
 <div class="btn-group">  
 <button type="submit" class="btn btn-primary">📤 提交预约</button>  
 <a href="${pageContext.request.contextPath}/" class="btn btn-secondary">🔙 返回首页</a>  
 </div>  
 </form>  
 </div>  
 </div>  
   
 <script>  
 let *accompanyingCount* = 0;  
   
 function toggleOfficialFields() {  
 const appointmentType = *document*.getElementById('appointmentType').value;  
 const officialFields = *document*.getElementById('officialFields');  
   
 if (appointmentType === 'OFFICIAL') {  
 officialFields.classList.add('show');  
 // 设置必填字段  
 officialFields.querySelectorAll('input, select, textarea').forEach(field => {  
 field.required = true;  
 });  
 } else {  
 officialFields.classList.remove('show');  
 // 移除必填字段  
 officialFields.querySelectorAll('input, select, textarea').forEach(field => {  
 field.required = false;  
 });  
 }  
 }  
   
 function addAccompanying() {  
 *accompanyingCount*++;  
 const accompanyingList = *document*.getElementById('accompanyingList');  
   
 const accompanyingItem = *document*.createElement('div');  
 accompanyingItem.className = 'accompanying-item';  
 accompanyingItem.innerHTML = `  
 <div class="accompanying-header">  
 <h4>随行人员 ${accompanyingCount}</h4>  
 <button type="button" class="remove-accompanying" onclick="removeAccompanying(this)">删除</button>  
 </div>  
 <div class="form-row">  
 <div class="form-group">  
 <label>姓名</label>  
 <input type="text" name="accompanyingName" required>  
 </div>  
 <div class="form-group">  
 <label>身份证号</label>  
 <input type="text" name="accompanyingIdCard" required pattern="[0-9Xx]{18}">  
 </div>  
 </div>  
 <div class="form-row">  
 <div class="form-group">  
 <label>手机号</label>  
 <input type="tel" name="accompanyingPhone" required pattern="[0-9]{11}">  
 </div>  
 </div>  
 `;  
   
 accompanyingList.appendChild(accompanyingItem);  
 }  
   
 function removeAccompanying(button) {  
 button.closest('.accompanying-item').remove();  
 }  
   
 // 设置默认进校时间为明天上午9点  
 *window*.onload = function() {  
 const tomorrow = new *Date*();  
 tomorrow.setDate(tomorrow.getDate() + 1);  
 tomorrow.setHours(9, 0, 0, 0);  
   
 const dateTimeLocal = tomorrow.toISOString().slice(0, 16);  
 *document*.querySelector('input[name="entryTime"]').value = dateTimeLocal;  
 };  
 </script>  
</body>  
</html>

#### src/main/webapp/error.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **page** import="java.time.LocalDateTime" %>  
<%@ **page** import="java.time.format.DateTimeFormatter" %>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>错误 - 校园通行码系统</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
   
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 padding: 20px;  
 }  
   
 .error-container {  
 background: white;  
 border-radius: 20px;  
 box-shadow: 0 20px 40px rgba(0,0,0,0.1);  
 padding: 40px;  
 text-align: center;  
 max-width: 600px;  
 width: 100%;  
 }  
   
 .error-icon {  
 font-size: 4em;  
 margin-bottom: 20px;  
 color: #e74c3c;  
 }  
   
 .error-title {  
 font-size: 1.8em;  
 color: #333;  
 margin-bottom: 15px;  
 font-weight: bold;  
 }  
   
 .error-message {  
 color: #666;  
 margin-bottom: 20px;  
 font-size: 1.1em;  
 line-height: 1.6;  
 }  
   
 .error-details {  
 background: #f8f9fa;  
 border: 1px solid #e9ecef;  
 border-radius: 10px;  
 padding: 20px;  
 margin: 20px 0;  
 text-align: left;  
 }  
   
 .error-details h4 {  
 color: #495057;  
 margin-bottom: 10px;  
 font-size: 1.1em;  
 }  
   
 .error-details p {  
 color: #6c757d;  
 margin: 5px 0;  
 font-size: 0.95em;  
 }  
   
 .error-type {  
 display: inline-block;  
 background: #dc3545;  
 color: white;  
 padding: 5px 12px;  
 border-radius: 15px;  
 font-size: 0.85em;  
 margin-bottom: 15px;  
 }  
   
 .timestamp {  
 color: #6c757d;  
 font-size: 0.9em;  
 margin-bottom: 20px;  
 }  
   
 .btn-group {  
 display: flex;  
 gap: 15px;  
 justify-content: center;  
 flex-wrap: wrap;  
 }  
   
 .btn {  
 padding: 12px 25px;  
 border: none;  
 border-radius: 8px;  
 font-size: 1em;  
 cursor: pointer;  
 text-decoration: none;  
 display: inline-block;  
 transition: all 0.3s ease;  
 font-weight: 500;  
 }  
   
 .btn-primary {  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 color: white;  
 }  
   
 .btn-primary:hover {  
 transform: translateY(-2px);  
 box-shadow: 0 10px 20px rgba(102, 126, 234, 0.3);  
 }  
   
 .btn-secondary {  
 background: #6c757d;  
 color: white;  
 }  
   
 .btn-secondary:hover {  
 background: #5a6268;  
 transform: translateY(-2px);  
 }  
   
 .btn-danger {  
 background: #dc3545;  
 color: white;  
 }  
   
 .btn-danger:hover {  
 background: #c82333;  
 transform: translateY(-2px);  
 }  
   
 .error-code {  
 font-family: 'Courier New', monospace;  
 background: #f1f3f4;  
 padding: 10px;  
 border-radius: 5px;  
 margin: 10px 0;  
 font-size: 0.9em;  
 color: #333;  
 }  
   
 @media (max-width: 480px) {  
 .error-container {  
 padding: 30px 20px;  
 }  
   
 .error-title {  
 font-size: 1.5em;  
 }  
   
 .btn-group {  
 flex-direction: column;  
 }  
 }  
 </style>  
</head>  
<body>  
 <div class="error-container">  
 <div class="error-icon">⚠️</div>  
 <div class="error-title">出错了！</div>  
   
 <%   
 String errorMessage = (String) request.getAttribute("error");  
 String errorDetails = (String) request.getAttribute("errorDetails");  
 String errorType = (String) request.getAttribute("errorType");  
 String currentTime = LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*("yyyy-MM-dd HH:mm:ss"));  
 %>  
   
 <div class="error-message">  
 <% if (errorMessage != null) { %>  
 <%= errorMessage %>  
 <% } else { %>  
 系统发生未知错误，请稍后重试。  
 <% } %>  
 </div>  
   
 <% if (errorType != null) { %>  
 <div class="error-type">  
 错误类型: <%= getErrorTypeDescription(errorType) %>  
 </div>  
 <% } %>  
   
 <div class="timestamp">  
 发生时间: <%= currentTime %>  
 </div>  
   
 <% if (errorDetails != null && !errorDetails.trim().isEmpty()) { %>  
 <div class="error-details">  
 <h4>🔍 错误详情</h4>  
 <p><%= errorDetails %></p>  
   
 <% if (errorType != null) { %>  
 <h4>💡 解决建议</h4>  
 <p><%= getErrorSolution(errorType) %></p>  
 <% } %>  
 </div>  
 <% } %>  
   
 <div class="btn-group">  
 <a href="${pageContext.request.contextPath}/" class="btn btn-primary">  
 🏠 返回首页  
 </a>  
 <a href="javascript:*history*.back()" class="btn btn-secondary">  
 🔙 返回上页  
 </a>  
 <% if ("DATABASE\_CONNECTION".equals(errorType) || "UNKNOWN\_ERROR".equals(errorType)) { %>  
 <a href="javascript:*location*.reload()" class="btn btn-danger">  
 🔄 重新加载  
 </a>  
 <% } %>  
 </div>  
 </div>  
   
 <%!  
 private String getErrorTypeDescription(String errorType) {  
 switch (errorType) {  
 case "INPUT\_VALIDATION":  
 return "输入验证错误";  
 case "USER\_NOT\_FOUND":  
 return "用户不存在";  
 case "PASSWORD\_MISMATCH":  
 return "密码不匹配";  
 case "ACCOUNT\_LOCKED":  
 return "账户被锁定";  
 case "DATABASE\_CONNECTION":  
 return "数据库连接错误";  
 case "PASSWORD\_ENCRYPTION":  
 return "密码加密错误";  
 case "LOCKOUT\_ERROR":  
 return "账户锁定错误";  
 case "UNKNOWN\_ERROR":  
 return "未知错误";  
 default:  
 return "系统错误";  
 }  
 }  
   
 private String getErrorSolution(String errorType) {  
 switch (errorType) {  
 case "INPUT\_VALIDATION":  
 return "请检查输入的用户名和密码是否完整，确保没有多余的空格。";  
 case "USER\_NOT\_FOUND":  
 return "请确认用户名是否正确，或联系系统管理员创建账户。";  
 case "PASSWORD\_MISMATCH":  
 return "请确认密码是否正确，注意大小写。如果忘记密码，请联系系统管理员重置。";  
 case "ACCOUNT\_LOCKED":  
 return "账户因多次登录失败被锁定，请等待锁定时间结束后再试，或联系系统管理员解锁。";  
 case "DATABASE\_CONNECTION":  
 return "系统暂时无法连接数据库，请稍后重试。如果问题持续存在，请联系技术支持。";  
 case "PASSWORD\_ENCRYPTION":  
 return "密码处理过程中出现技术问题，请稍后重试。如果问题持续存在，请联系技术支持。";  
 case "LOCKOUT\_ERROR":  
 return "账户锁定过程中出现错误，请联系系统管理员处理。";  
 case "UNKNOWN\_ERROR":  
 return "系统发生未知错误，请稍后重试。如果问题持续存在，请联系技术支持并提供错误详情。";  
 default:  
 return "请稍后重试，如果问题持续存在，请联系技术支持。";  
 }  
 }  
 %>  
</body>  
</html>

#### src/main/webapp/index.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>校园通行码预约管理系统</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
   
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 }  
   
 .container {  
 background: white;  
 border-radius: 20px;  
 box-shadow: 0 20px 40px rgba(0,0,0,0.1);  
 padding: 40px;  
 text-align: center;  
 max-width: 500px;  
 width: 90%;  
 }  
   
 .logo {  
 font-size: 2.5em;  
 color: #333;  
 margin-bottom: 20px;  
 font-weight: bold;  
 }  
   
 .subtitle {  
 color: #666;  
 margin-bottom: 40px;  
 font-size: 1.1em;  
 }  
   
 .qr-section {  
 background: #f8f9fa;  
 border-radius: 15px;  
 padding: 30px;  
 margin-bottom: 30px;  
 }  
   
 .qr-title {  
 font-size: 1.3em;  
 color: #333;  
 margin-bottom: 20px;  
 }  
   
 .qr-code {  
 width: 200px;  
 height: 200px;  
 background: #ddd;  
 margin: 0 auto 20px;  
 border-radius: 10px;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 font-size: 0.9em;  
 color: #666;  
 }  
   
 .btn-group {  
 display: flex;  
 flex-direction: column;  
 gap: 15px;  
 }  
   
 .btn {  
 padding: 15px 30px;  
 border: none;  
 border-radius: 10px;  
 font-size: 1.1em;  
 cursor: pointer;  
 text-decoration: none;  
 display: inline-block;  
 transition: all 0.3s ease;  
 font-weight: 500;  
 }  
   
 .btn-primary {  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 color: white;  
 }  
   
 .btn-primary:hover {  
 transform: translateY(-2px);  
 box-shadow: 0 10px 20px rgba(102, 126, 234, 0.3);  
 }  
   
 .btn-secondary {  
 background: #6c757d;  
 color: white;  
 }  
   
 .btn-secondary:hover {  
 background: #5a6268;  
 transform: translateY(-2px);  
 }  
   
 .admin-link {  
 margin-top: 20px;  
 padding-top: 20px;  
 border-top: 1px solid #eee;  
 }  
   
 .admin-link a {  
 color: #667eea;  
 text-decoration: none;  
 font-size: 0.9em;  
 }  
   
 .admin-link a:hover {  
 text-decoration: underline;  
 }  
   
 @media (max-width: 480px) {  
 .container {  
 padding: 20px;  
 }  
   
 .logo {  
 font-size: 2em;  
 }  
   
 .btn {  
 padding: 12px 20px;  
 font-size: 1em;  
 }  
 }  
 </style>  
</head>  
<body>  
 <div class="container">  
 <div class="logo">🏫 校园通行码</div>  
 <div class="subtitle">预约管理系统</div>  
   
 <div class="qr-section">  
 <div class="qr-title">📱 扫码预约</div>  
 <div class="qr-code">  
 <img src="${pageContext.request.contextPath}/qrcode?content=${pageContext.request.contextPath}/appointment/apply" alt="预约通道二维码" width="200" height="200"/>  
 </div>  
 <p style="color: #666; font-size: 0.9em;">  
 扫描二维码进入预约系统<br>  
 或点击下方按钮直接访问  
 </p>  
 </div>  
   
 <div class="btn-group">  
 <a href="${pageContext.request.contextPath}/appointment/apply" class="btn btn-primary">  
 📝 我要预约  
 </a>  
 <a href="${pageContext.request.contextPath}/appointment/query" class="btn btn-secondary">  
 📋 我的预约  
 </a>  
 </div>  
   
 <div class="admin-link">  
 <a href="${pageContext.request.contextPath}/admin/login">管理员登录</a>  
 </div>  
 </div>  
</body>  
</html>

#### src/main/webapp/login\_test.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>登录测试 - 校园通行码系统</title>  
 <style>  
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 padding: 20px;  
 }  
   
 .test-container {  
 background: white;  
 border-radius: 20px;  
 box-shadow: 0 20px 40px rgba(0,0,0,0.1);  
 padding: 40px;  
 max-width: 600px;  
 width: 100%;  
 text-align: center;  
 }  
   
 .test-title {  
 font-size: 2em;  
 color: #333;  
 margin-bottom: 20px;  
 }  
   
 .test-info {  
 color: #666;  
 margin-bottom: 30px;  
 line-height: 1.6;  
 }  
   
 .test-accounts {  
 background: #f8f9fa;  
 border-radius: 10px;  
 padding: 20px;  
 margin-bottom: 30px;  
 text-align: left;  
 }  
   
 .test-accounts h3 {  
 color: #333;  
 margin-bottom: 15px;  
 }  
   
 .account-item {  
 margin-bottom: 10px;  
 padding: 10px;  
 background: white;  
 border-radius: 5px;  
 border-left: 4px solid #667eea;  
 }  
   
 .account-item strong {  
 color: #333;  
 }  
   
 .btn-group {  
 display: flex;  
 gap: 15px;  
 justify-content: center;  
 flex-wrap: wrap;  
 }  
   
 .btn {  
 padding: 12px 25px;  
 border: none;  
 border-radius: 8px;  
 font-size: 1em;  
 cursor: pointer;  
 text-decoration: none;  
 display: inline-block;  
 transition: all 0.3s ease;  
 font-weight: 500;  
 }  
   
 .btn-primary {  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 color: white;  
 }  
   
 .btn-primary:hover {  
 transform: translateY(-2px);  
 box-shadow: 0 10px 20px rgba(102, 126, 234, 0.3);  
 }  
   
 .btn-secondary {  
 background: #6c757d;  
 color: white;  
 }  
   
 .btn-secondary:hover {  
 background: #5a6268;  
 transform: translateY(-2px);  
 }  
   
 .status-info {  
 background: #e7f3ff;  
 color: #0c5460;  
 padding: 15px;  
 border-radius: 10px;  
 margin-top: 20px;  
 border: 1px solid #bee5eb;  
 }  
 </style>  
</head>  
<body>  
 <div class="test-container">  
 <div class="test-title">🔐 登录功能测试</div>  
 <div class="test-info">  
 这个页面用于测试管理员登录功能。请使用以下测试账号进行登录测试。  
 </div>  
   
 <div class="test-accounts">  
 <h3>📋 测试账号信息</h3>  
 <div class="account-item">  
 <strong>系统管理员：</strong>admin / Admin@123  
 </div>  
 <div class="account-item">  
 <strong>学校管理员：</strong>school\_admin / Admin@123  
 </div>  
 <div class="account-item">  
 <strong>部门管理员：</strong>dept\_admin / Admin@123  
 </div>  
 <div class="account-item">  
 <strong>审计管理员：</strong>audit\_admin / Admin@123  
 </div>  
 </div>  
   
 <div class="btn-group">  
 <a href="${pageContext.request.contextPath}/admin/login" class="btn btn-primary">  
 🔑 开始登录测试  
 </a>  
 <a href="${pageContext.request.contextPath}/test\_error.jsp" class="btn btn-secondary">  
 ⚠️ 错误处理测试  
 </a>  
 </div>  
   
 <div class="status-info">  
 <h4>💡 测试说明</h4>  
 <ul style="text-align: left; margin: 10px 0;">  
 <li>所有账号的密码都是：Admin@123</li>  
 <li>系统管理员和学校管理员会跳转到管理员控制台</li>  
 <li>审计管理员会跳转到审计日志管理页面</li>  
 <li>可以尝试输入错误密码测试错误处理功能</li>  
 <li>连续失败5次会锁定账户30分钟</li>  
 </ul>  
 </div>  
 </div>  
</body>  
</html>

#### src/main/webapp/my\_appointments.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<%@ **taglib** prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>我的预约 - 校园通行码系统</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
   
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 padding: 20px;  
 }  
   
 .container {  
 max-width: 900px;  
 margin: 0 auto;  
 background: white;  
 border-radius: 15px;  
 box-shadow: 0 20px 40px rgba(0,0,0,0.1);  
 overflow: hidden;  
 }  
   
 .header {  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 color: white;  
 padding: 30px;  
 text-align: center;  
 }  
   
 .header h1 {  
 font-size: 2em;  
 margin-bottom: 10px;  
 }  
   
 .header p {  
 opacity: 0.9;  
 font-size: 1.1em;  
 }  
   
 .content {  
 padding: 40px;  
 }  
   
 .search-form {  
 background: #f8f9fa;  
 border-radius: 10px;  
 padding: 25px;  
 margin-bottom: 30px;  
 }  
   
 .form-title {  
 font-size: 1.3em;  
 color: #333;  
 margin-bottom: 20px;  
 font-weight: bold;  
 }  
   
 .form-row {  
 display: flex;  
 gap: 20px;  
 margin-bottom: 20px;  
 }  
   
 .form-group {  
 flex: 1;  
 }  
   
 .form-group label {  
 display: block;  
 margin-bottom: 8px;  
 color: #333;  
 font-weight: 500;  
 }  
   
 .form-group input {  
 width: 100%;  
 padding: 12px;  
 border: 2px solid #e1e5e9;  
 border-radius: 8px;  
 font-size: 1em;  
 transition: border-color 0.3s ease;  
 }  
   
 .form-group input:focus {  
 outline: none;  
 border-color: #667eea;  
 }  
   
 .btn-group {  
 display: flex;  
 gap: 15px;  
 justify-content: center;  
 }  
   
 .btn {  
 padding: 12px 25px;  
 border: none;  
 border-radius: 8px;  
 font-size: 1em;  
 cursor: pointer;  
 text-decoration: none;  
 display: inline-block;  
 transition: all 0.3s ease;  
 font-weight: 500;  
 }  
   
 .btn-primary {  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 color: white;  
 }  
   
 .btn-primary:hover {  
 transform: translateY(-2px);  
 box-shadow: 0 10px 20px rgba(102, 126, 234, 0.3);  
 }  
   
 .btn-secondary {  
 background: #6c757d;  
 color: white;  
 }  
   
 .btn-secondary:hover {  
 background: #5a6268;  
 transform: translateY(-2px);  
 }  
   
 .btn-small {  
 padding: 8px 15px;  
 font-size: 0.9em;  
 }  
   
 .results-section {  
 margin-top: 30px;  
 }  
   
 .results-title {  
 font-size: 1.3em;  
 color: #333;  
 margin-bottom: 20px;  
 font-weight: bold;  
 }  
   
 .appointment-card {  
 border: 1px solid #dee2e6;  
 border-radius: 10px;  
 padding: 20px;  
 margin-bottom: 20px;  
 background: white;  
 box-shadow: 0 2px 10px rgba(0,0,0,0.05);  
 transition: transform 0.3s ease;  
 }  
   
 .appointment-card:hover {  
 transform: translateY(-2px);  
 box-shadow: 0 5px 15px rgba(0,0,0,0.1);  
 }  
   
 .card-header {  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 margin-bottom: 15px;  
 padding-bottom: 10px;  
 border-bottom: 1px solid #f1f3f4;  
 }  
   
 .card-title {  
 font-size: 1.2em;  
 color: #333;  
 font-weight: bold;  
 }  
   
 .status-badge {  
 padding: 5px 12px;  
 border-radius: 15px;  
 font-size: 0.9em;  
 font-weight: bold;  
 }  
   
 .status-pending {  
 background: #fff3cd;  
 color: #856404;  
 }  
   
 .status-approved {  
 background: #d4edda;  
 color: #155724;  
 }  
   
 .status-rejected {  
 background: #f8d7da;  
 color: #721c24;  
 }  
   
 .card-content {  
 display: grid;  
 grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));  
 gap: 15px;  
 margin-bottom: 15px;  
 }  
   
 .info-item {  
 display: flex;  
 flex-direction: column;  
 }  
   
 .info-label {  
 color: #666;  
 font-size: 0.9em;  
 margin-bottom: 5px;  
 }  
   
 .info-value {  
 color: #333;  
 font-weight: 500;  
 }  
   
 .card-actions {  
 display: flex;  
 gap: 10px;  
 justify-content: flex-end;  
 }  
   
 .message {  
 padding: 15px;  
 border-radius: 8px;  
 margin-bottom: 20px;  
 text-align: center;  
 }  
   
 .message-info {  
 background: #d1ecf1;  
 color: #0c5460;  
 border: 1px solid #bee5eb;  
 }  
   
 .message-error {  
 background: #f8d7da;  
 color: #721c24;  
 border: 1px solid #f5c6cb;  
 }  
   
 .no-results {  
 text-align: center;  
 padding: 40px;  
 color: #666;  
 }  
   
 .no-results-icon {  
 font-size: 3em;  
 margin-bottom: 20px;  
 opacity: 0.5;  
 }  
   
 @media (max-width: 768px) {  
 .form-row {  
 flex-direction: column;  
 gap: 15px;  
 }  
   
 .content {  
 padding: 20px;  
 }  
   
 .header {  
 padding: 20px;  
 }  
   
 .header h1 {  
 font-size: 1.5em;  
 }  
   
 .card-content {  
 grid-template-columns: 1fr;  
 }  
   
 .card-actions {  
 flex-direction: column;  
 }  
 }  
 </style>  
</head>  
<body>  
 <div class="container">  
 <div class="header">  
 <h1>📋 我的预约</h1>  
 <p>查询您的预约记录和通行码</p>  
 </div>  
   
 <div class="content">  
 <!-- 查询表单 -->  
 <div class="search-form">  
 <div class="form-title">🔍 查询预约</div>  
 <form action="${pageContext.request.contextPath}/appointment/query" method="post">  
 <div class="form-row">  
 <div class="form-group">  
 <label>姓名 <span style="color: #e74c3c;">\*</span></label>  
 <input type="text" name="visitorName" value="${visitorName}" required>  
 </div>  
 <div class="form-group">  
 <label>身份证号 <span style="color: #e74c3c;">\*</span></label>  
 <input type="text" name="visitorIdCard" value="${visitorIdCard}" required pattern="[0-9Xx]{18}">  
 </div>  
 </div>  
 <div class="form-row">  
 <div class="form-group">  
 <label>手机号 <span style="color: #e74c3c;">\*</span></label>  
 <input type="tel" name="visitorPhone" value="${visitorPhone}" required pattern="[0-9]{11}">  
 </div>  
 </div>  
 <div class="btn-group">  
 <button type="submit" class="btn btn-primary">🔍 查询预约</button>  
 <a href="${pageContext.request.contextPath}/" class="btn btn-secondary">🏠 返回首页</a>  
 </div>  
 </form>  
 </div>  
   
 <!-- 消息提示 -->  
 <**c:if** test="${not empty error}">  
 <div class="message message-error">  
 ❌ ${error}  
 </div>  
 </**c:if**>  
   
 <**c:if** test="${not empty message}">  
 <div class="message message-info">  
 ℹ️ ${message}  
 </div>  
 </**c:if**>  
   
 <!-- 查询结果 -->  
 <**c:if** test="${not empty appointments}">  
 <div class="results-section">  
 <div class="results-title">📊 查询结果 (${appointments.size()} 条记录)</div>  
   
 <**c:forEach** var="appointment" items="${appointments}">  
 <div class="appointment-card">  
 <div class="card-header">  
 <div class="card-title">  
 ${appointment.visitorName} - ${appointment.campus}  
 </div>  
 <div class="status-badge status-${appointment.status.toLowerCase()}">  
 <**c:choose**>  
 <**c:when** test="${appointment.status == 'PENDING'}">⏳ 待审核</**c:when**>  
 <**c:when** test="${appointment.status == 'APPROVED'}">✅ 已通过</**c:when**>  
 <**c:when** test="${appointment.status == 'REJECTED'}">❌ 已拒绝</**c:when**>  
 </**c:choose**>  
 </div>  
 </div>  
   
 <div class="card-content">  
 <div class="info-item">  
 <span class="info-label">身份证号</span>  
 <span class="info-value">${appointment.visitorIdCard}</span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">手机号</span>  
 <span class="info-value">${appointment.visitorPhone}</span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">所在单位</span>  
 <span class="info-value">${appointment.visitorUnit}</span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">进校时间</span>  
 <span class="info-value">  
 <**fmt:formatDate** value="${appointment.entryTime}" pattern="yyyy-MM-dd HH:mm"/>  
 </span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">交通方式</span>  
 <span class="info-value">${appointment.transportMode}</span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">预约类型</span>  
 <span class="info-value">  
 ${appointment.appointmentType == 'OFFICIAL' ? '公务预约' : '社会公众预约'}  
 </span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">申请时间</span>  
 <span class="info-value">  
 <**fmt:formatDate** value="${appointment.createdAt}" pattern="yyyy-MM-dd HH:mm:ss"/>  
 </span>  
 </div>  
 <**c:if** test="${appointment.appointmentType == 'OFFICIAL'}">  
 <div class="info-item">  
 <span class="info-label">访问部门</span>  
 <span class="info-value">${appointment.officialDeptName}</span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">接待人</span>  
 <span class="info-value">${appointment.officialContactPerson}</span>  
 </div>  
 </**c:if**>  
 </div>  
   
 <div class="card-actions">  
 <a href="${pageContext.request.contextPath}/passcode/view?id=${appointment.id}"   
 class="btn btn-primary btn-small">  
 👁️ 查看通行码  
 </a>  
 <**c:if** test="${appointment.status == 'PENDING'}">  
 <span class="btn btn-secondary btn-small" style="cursor: default;">  
 ⏳ 等待审核  
 </span>  
 </**c:if**>  
 </div>  
 </div>  
 </**c:forEach**>  
 </div>  
 </**c:if**>  
   
 <!-- 无结果提示 -->  
 <**c:if** test="${empty appointments && not empty visitorName}">  
 <div class="no-results">  
 <div class="no-results-icon">📭</div>  
 <h3>未找到预约记录</h3>  
 <p>请检查您输入的信息是否正确，或尝试重新申请预约。</p>  
 <a href="${pageContext.request.contextPath}/appointment/apply" class="btn btn-primary" style="margin-top: 20px;">  
 📝 申请新预约  
 </a>  
 </div>  
 </**c:if**>  
 </div>  
 </div>  
</body>  
</html>

#### src/main/webapp/pass\_code.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<%@ **taglib** prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>通行码 - 校园通行码系统</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
   
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 padding: 20px;  
 }  
   
 .container {  
 max-width: 400px;  
 margin: 0 auto;  
 background: white;  
 border-radius: 20px;  
 box-shadow: 0 20px 40px rgba(0,0,0,0.1);  
 overflow: hidden;  
 }  
   
 .header {  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 color: white;  
 padding: 30px;  
 text-align: center;  
 }  
   
 .header h1 {  
 font-size: 1.8em;  
 margin-bottom: 10px;  
 }  
   
 .header p {  
 opacity: 0.9;  
 font-size: 1em;  
 }  
   
 .passcode-container {  
 padding: 30px;  
 text-align: center;  
 }  
   
 .status-indicator {  
 display: inline-block;  
 padding: 8px 20px;  
 border-radius: 20px;  
 font-weight: bold;  
 margin-bottom: 20px;  
 font-size: 1.1em;  
 }  
   
 .status-valid {  
 background: #d4edda;  
 color: #155724;  
 border: 2px solid #c3e6cb;  
 }  
   
 .status-invalid {  
 background: #f8d7da;  
 color: #721c24;  
 border: 2px solid #f5c6cb;  
 }  
   
 .qr-code-container {  
 margin: 20px 0;  
 padding: 20px;  
 background: #f8f9fa;  
 border-radius: 15px;  
 }  
   
 .qr-code {  
 width: 200px;  
 height: 200px;  
 margin: 0 auto 15px;  
 border: 2px solid #dee2e6;  
 border-radius: 10px;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 background: white;  
 }  
   
 .qr-code img {  
 max-width: 100%;  
 max-height: 100%;  
 }  
   
 .qr-placeholder {  
 color: #6c757d;  
 font-size: 0.9em;  
 text-align: center;  
 }  
   
 .info-section {  
 margin: 20px 0;  
 text-align: left;  
 }  
   
 .info-title {  
 font-size: 1.2em;  
 color: #333;  
 margin-bottom: 15px;  
 font-weight: bold;  
 border-bottom: 2px solid #667eea;  
 padding-bottom: 5px;  
 }  
   
 .info-item {  
 display: flex;  
 justify-content: space-between;  
 margin-bottom: 10px;  
 padding: 8px 0;  
 border-bottom: 1px solid #f1f3f4;  
 }  
   
 .info-label {  
 color: #666;  
 font-weight: 500;  
 }  
   
 .info-value {  
 color: #333;  
 font-weight: bold;  
 }  
   
 .btn-group {  
 display: flex;  
 gap: 10px;  
 justify-content: center;  
 margin-top: 30px;  
 }  
   
 .btn {  
 padding: 12px 25px;  
 border: none;  
 border-radius: 8px;  
 font-size: 1em;  
 cursor: pointer;  
 text-decoration: none;  
 display: inline-block;  
 transition: all 0.3s ease;  
 font-weight: 500;  
 }  
   
 .btn-primary {  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 color: white;  
 }  
   
 .btn-primary:hover {  
 transform: translateY(-2px);  
 box-shadow: 0 10px 20px rgba(102, 126, 234, 0.3);  
 }  
   
 .btn-secondary {  
 background: #6c757d;  
 color: white;  
 }  
   
 .btn-secondary:hover {  
 background: #5a6268;  
 transform: translateY(-2px);  
 }  
   
 .warning {  
 background: #fff3cd;  
 color: #856404;  
 padding: 15px;  
 border-radius: 8px;  
 margin: 20px 0;  
 border: 1px solid #ffeaa7;  
 }  
   
 .note {  
 background: #e7f3ff;  
 color: #0c5460;  
 padding: 15px;  
 border-radius: 8px;  
 margin: 20px 0;  
 border: 1px solid #bee5eb;  
 font-size: 0.9em;  
 }  
   
 @media (max-width: 480px) {  
 .container {  
 margin: 10px;  
 }  
   
 .header {  
 padding: 20px;  
 }  
   
 .header h1 {  
 font-size: 1.5em;  
 }  
   
 .passcode-container {  
 padding: 20px;  
 }  
   
 .qr-code {  
 width: 180px;  
 height: 180px;  
 }  
 }  
 </style>  
</head>  
<body>  
 <div class="container">  
 <div class="header">  
 <h1>🎫 校园通行码</h1>  
 <p>请向门卫出示此通行码</p>  
 </div>  
   
 <div class="passcode-container">  
 <!-- 状态指示器 -->  
 <div class="status-indicator ${isValid ? 'status-valid' : 'status-invalid'}">  
 ${isValid ? '✅ 有效通行码' : '❌ 无效通行码'}  
 </div>  
   
 <!-- 二维码 -->  
 <div class="qr-code-container">  
 <div class="qr-code">  
 <**c:if** test="${not empty qrCodeBase64}">  
 <img src="${qrCodeBase64}" alt="通行码二维码">  
 </**c:if**>  
 <**c:if** test="${empty qrCodeBase64}">  
 <div class="qr-placeholder">  
 📱 二维码<br>  
 200×200px  
 </div>  
 </**c:if**>  
 </div>  
 <p style="color: #666; font-size: 0.9em;">  
 ${isValid ? '扫描二维码验证通行信息' : '通行码已过期或无效'}  
 </p>  
 </div>  
   
 <!-- 预约信息 -->  
 <div class="info-section">  
 <div class="info-title">📋 预约信息</div>  
 <div class="info-item">  
 <span class="info-label">预约人：</span>  
 <span class="info-value">${maskedVisitorName}</span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">身份证：</span>  
 <span class="info-value">${maskedIdCard}</span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">手机号：</span>  
 <span class="info-value">${decryptedPhone}</span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">所在单位：</span>  
 <span class="info-value">${appointment.visitorUnit}</span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">预约校区：</span>  
 <span class="info-value">${appointment.campus}</span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">进校时间：</span>  
 <span class="info-value">  
 <**fmt:formatDate** value="${appointment.entryTime}" pattern="yyyy-MM-dd HH:mm"/>  
 </span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">交通方式：</span>  
 <span class="info-value">${appointment.transportMode}</span>  
 </div>  
 <**c:if** test="${not empty appointment.licensePlate}">  
 <div class="info-item">  
 <span class="info-label">车牌号：</span>  
 <span class="info-value">${appointment.licensePlate}</span>  
 </div>  
 </**c:if**>  
 <div class="info-item">  
 <span class="info-label">预约类型：</span>  
 <span class="info-value">  
 ${appointment.appointmentType == 'OFFICIAL' ? '公务预约' : '社会公众预约'}  
 </span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">申请时间：</span>  
 <span class="info-value">  
 <**fmt:formatDate** value="${appointment.createdAt}" pattern="yyyy-MM-dd HH:mm:ss"/>  
 </span>  
 </div>  
 </div>  
   
 <!-- 公务预约特有信息 -->  
 <**c:if** test="${appointment.appointmentType == 'OFFICIAL'}">  
 <div class="info-section">  
 <div class="info-title">🏢 公务信息</div>  
 <div class="info-item">  
 <span class="info-label">访问部门：</span>  
 <span class="info-value">${appointment.officialDeptName}</span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">接待人：</span>  
 <span class="info-value">${appointment.officialContactPerson}</span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">来访事由：</span>  
 <span class="info-value">${appointment.officialReason}</span>  
 </div>  
 <**c:if** test="${not empty appointment.auditedByName}">  
 <div class="info-item">  
 <span class="info-label">审核人：</span>  
 <span class="info-value">${appointment.auditedByName}</span>  
 </div>  
 <div class="info-item">  
 <span class="info-label">审核时间：</span>  
 <span class="info-value">  
 <**fmt:formatDate** value="${appointment.auditedAt}" pattern="yyyy-MM-dd HH:mm:ss"/>  
 </span>  
 </div>  
 </**c:if**>  
 </div>  
 </**c:if**>  
   
 <!-- 状态说明 -->  
 <**c:if** test="${!isValid}">  
 <div class="warning">  
 ⚠️ 通行码已过期或无效，请重新申请预约  
 </div>  
 </**c:if**>  
   
 <div class="note">  
 💡 通行码在进校当天（0:00-23:59）内有效<br>  
 🔒 个人信息已进行脱敏处理，保护隐私安全  
 </div>  
   
 <div class="btn-group">  
 <a href="${pageContext.request.contextPath}/appointment/apply" class="btn btn-primary">  
 📝 重新预约  
 </a>  
 <a href="${pageContext.request.contextPath}/" class="btn btn-secondary">  
 🏠 返回首页  
 </a>  
 </div>  
 </div>  
 </div>  
</body>  
</html>

#### src/main/webapp/test\_error.jsp

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>错误处理测试 - 校园通行码系统</title>  
 <style>  
 body {  
 font-family: 'Microsoft YaHei', Arial, sans-serif;  
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  
 min-height: 100vh;  
 padding: 20px;  
 }  
 .container {  
 max-width: 800px;  
 margin: 0 auto;  
 background: white;  
 border-radius: 20px;  
 padding: 30px;  
 box-shadow: 0 20px 40px rgba(0,0,0,0.1);  
 }  
 .test-section {  
 margin-bottom: 30px;  
 padding: 20px;  
 border: 1px solid #e9ecef;  
 border-radius: 10px;  
 }  
 .test-section h3 {  
 color: #333;  
 margin-bottom: 15px;  
 }  
 .btn {  
 display: inline-block;  
 padding: 10px 20px;  
 margin: 5px;  
 background: #667eea;  
 color: white;  
 text-decoration: none;  
 border-radius: 5px;  
 transition: all 0.3s ease;  
 }  
 .btn:hover {  
 background: #5a6fd8;  
 transform: translateY(-2px);  
 }  
 .btn-danger {  
 background: #dc3545;  
 }  
 .btn-danger:hover {  
 background: #c82333;  
 }  
 </style>  
</head>  
<body>  
 <div class="container">  
 <h1>🔧 错误处理测试页面</h1>  
 <p>这个页面用于测试各种错误情况的处理。</p>  
   
 <div class="test-section">  
 <h3>📋 登录错误测试</h3>  
 <p>测试不同类型的登录错误：</p>  
 <a href="test\_error.jsp?error=INPUT\_VALIDATION" class="btn">输入验证错误</a>  
 <a href="test\_error.jsp?error=USER\_NOT\_FOUND" class="btn">用户不存在</a>  
 <a href="test\_error.jsp?error=PASSWORD\_MISMATCH" class="btn">密码不匹配</a>  
 <a href="test\_error.jsp?error=ACCOUNT\_LOCKED" class="btn">账户被锁定</a>  
 </div>  
   
 <div class="test-section">  
 <h3>🔧 系统错误测试</h3>  
 <p>测试系统级别的错误：</p>  
 <a href="test\_error.jsp?error=DATABASE\_CONNECTION" class="btn btn-danger">数据库连接错误</a>  
 <a href="test\_error.jsp?error=PASSWORD\_ENCRYPTION" class="btn btn-danger">密码加密错误</a>  
 <a href="test\_error.jsp?error=UNKNOWN\_ERROR" class="btn btn-danger">未知错误</a>  
 </div>  
   
 <div class="test-section">  
 <h3>🔗 页面跳转测试</h3>  
 <p>测试页面跳转功能：</p>  
 <a href="${pageContext.request.contextPath}/" class="btn">返回首页</a>  
 <a href="${pageContext.request.contextPath}/admin/login" class="btn">管理员登录</a>  
 </div>  
   
 <%   
 String errorType = request.getParameter("error");  
 if (errorType != null) {  
 request.setAttribute("error", getErrorMessage(errorType));  
 request.setAttribute("errorDetails", getErrorDetails(errorType));  
 request.setAttribute("errorType", errorType);  
 request.getRequestDispatcher("/error.jsp").forward(request, response);  
 }  
 %>  
 </div>  
   
 <%!  
 private String getErrorMessage(String errorType) {  
 switch (errorType) {  
 case "INPUT\_VALIDATION":  
 return "用户名和密码不能为空";  
 case "USER\_NOT\_FOUND":  
 return "用户名或密码错误";  
 case "PASSWORD\_MISMATCH":  
 return "用户名或密码错误";  
 case "ACCOUNT\_LOCKED":  
 return "账户已被锁定，请稍后再试";  
 case "DATABASE\_CONNECTION":  
 return "系统暂时无法连接数据库，请稍后重试";  
 case "PASSWORD\_ENCRYPTION":  
 return "密码处理失败，请稍后重试";  
 case "UNKNOWN\_ERROR":  
 return "系统发生未知错误，请稍后重试";  
 default:  
 return "系统发生错误";  
 }  
 }  
   
 private String getErrorDetails(String errorType) {  
 switch (errorType) {  
 case "INPUT\_VALIDATION":  
 return "请检查输入的用户名和密码是否完整";  
 case "USER\_NOT\_FOUND":  
 return "用户名 'testuser' 在系统中不存在";  
 case "PASSWORD\_MISMATCH":  
 return "连续失败次数: 2，剩余尝试次数: 3";  
 case "ACCOUNT\_LOCKED":  
 return "账户锁定截止时间: 2024-01-15 14:30:00";  
 case "DATABASE\_CONNECTION":  
 return "错误详情: Connection refused: connect";  
 case "PASSWORD\_ENCRYPTION":  
 return "密码加密过程出现错误: NoSuchAlgorithmException";  
 case "UNKNOWN\_ERROR":  
 return "错误详情: NullPointerException in AdminLoginServlet";  
 default:  
 return "未知错误详情";  
 }  
 }  
 %>  
</body>  
</html>