企业资产管理系统

软件设计说明

版本：3.0

编写：EAM 开发团队全体成员

校对：成成

审核：范炜祺

西北工业大学 - EAM 开发团队

2021 年 6 月

目 录

1 引言 4

1.1 文档标识 4

1.2 文档概述 4

1.3 项目概述 4

1.4 参考资料 4

1.5 文档标准 5

2 功能概述 5

3 数据流图 7

4 状态转换图 7

5 底层数据库设计 8

5.1 实体类设计 8

5.2 实体间关系 9

5.3 数据表属性及其数据字典 11

6 系统架构 12

7 功能设计 14

7.1 登录权限管理模块.....14

7.2 系统管理模块.....19

7.3 普通用户模块.....47

8 界面设计.....58

8.1 登录界面.....58

8.2 普通用户界面.....59

8.3 管理员界面.....59

1 引言

1.1 文档标识

中文名称：《软件设计说明》。

英文名称：“Software Design Introduction(SDI)”。

文档版本：“3.0”

文档编号 “ SS-NWPU-EAM-SDI-3.0(E)”

1.2 文档概述

本文档依据《国标 GB/T 8567-2006 计算机软件文档编制规范》制定，属于技术文档，仅限于 EAM 开发团队 等相关人员阅读。本文档描述开发者对软件各方面的设计。

1.3 项目概述

企业资产管理（EAM）系统是一个以 SpringMVC 框架为基础搭建的 JavaWeb 项目，便于企业管理员对企业重要设备（价值 5000 人民币以上）进行统筹分配，给予企业内员工申请调用特定设备的权限。

本软件系统用于企业日常的资产管理流程，完成资产登记、使用、审批、归还的一系列完整业务流程，旨在通过建立 Web 系统，自动化企业资产管理过程。

本系统有两类用户群体，根据角色权限分为：系统管理员和普通用户。

1.4 参考资料

[1] 《Java Web 框架开发技术(Spring+SpringMVC+MyBatis)》, 史胜辉, 王春明 编

著，清华大学出版社；

[2] 《Java Web 应用开发基础教程》，郭庆 等 编著，清华大学出版社。

1.5 文档标准

《GB/T 8567-2006 计算机软件文档编制规范》，国家标准

2 功能概述

本系统需要有两大用户类——管理员与用户，需要完成的主要功能模块有三大部分——登录权限模块、管理员系统管理模块和普通用户模块（管理员中仅有一位超级管理员，其账户由系统后台直接录入，id 为 0，拥有管理其他管理员账户的权限）。系统要求用户通过登录账户后申请需要的公司资产，管理员对所有账户进行创建和管理，了解当前的资产的信息和使用情况，并对用户资产的申请进行审批。

本系统的三大主要功能模块与 23 个功能点如表 1 所示：

功能模块	子功能模块	功能编号	功能描述
登录权限管理	权限管理	Authority_1	用户不能越级访问管理员功能
		Authority_2	未登录状态下不允许使用系统功能
	登录管理	Login_1	用户和管理员使用账户密码登录系统
		Login_2	登录失败系统打印错误提示信息
		Login_3	用户和管理员可以退出登录
系统管理模块	管理账户管理	Manager_1	管理员可以增加新的管理员账户
		Manager_2	管理员可以删除自己和初始管理员以外的其他管理员账户
		Manager_3	管理员可以更改自身账户的基本信息

		Manager_4	管理员可以修改自身账户密码
	企业资产管理	Property_1	管理员可以对企业资产进行录入
		Property_2	管理员可以按照申请时间顺序查看并审批资产领用申请并批准领用
		Property_3	管理员可以查看资产列表及单独资产的领用情况(领用历史)
		Property_4	管理员可以查看用户的信息以及领用情况
		Property_5	管理员可以通过结束申请并结束领用，改变资产状态
	普通用户管理	UserManage_1	管理员可以增加新的系统用户
		UserManage_2	管理员可以删除系统用户
		UserManage_3	管理员可以修改系统用户的基本信息
		UserManage_4	管理员可以查看系统用户列表
		UserManage_5	管理员可以修改系统用户的基本信息
		UserManage_6	管理员可以锁定特定用户的账号 (锁定后仅可以查看历史记录不可以发起请求)
普通用户模块	资产领用	Apply_1	用户可以浏览可领用的资产列表
		Apply_2	用户可以发起资产领用申请并确认领用单
		Apply_3	用户可以查看领用单审批状态和领用单历史记录
		Apply_4	用户可以查看具体领用资产的详细信息

3 数据流图

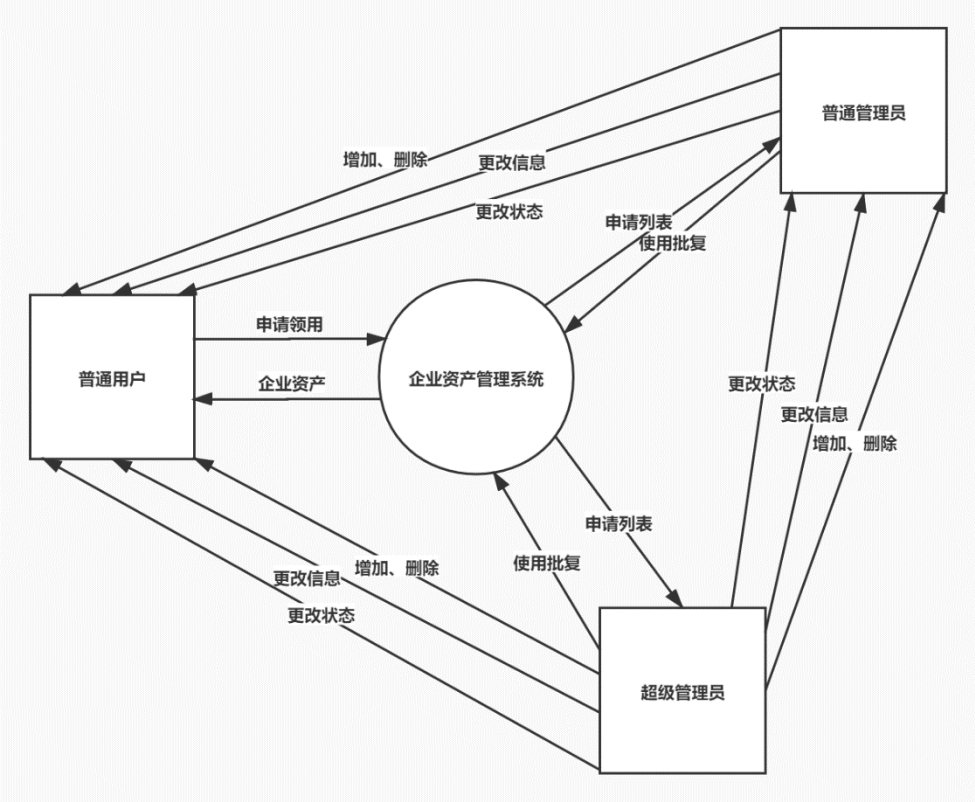

图 1 数据流图

4 状态转换图

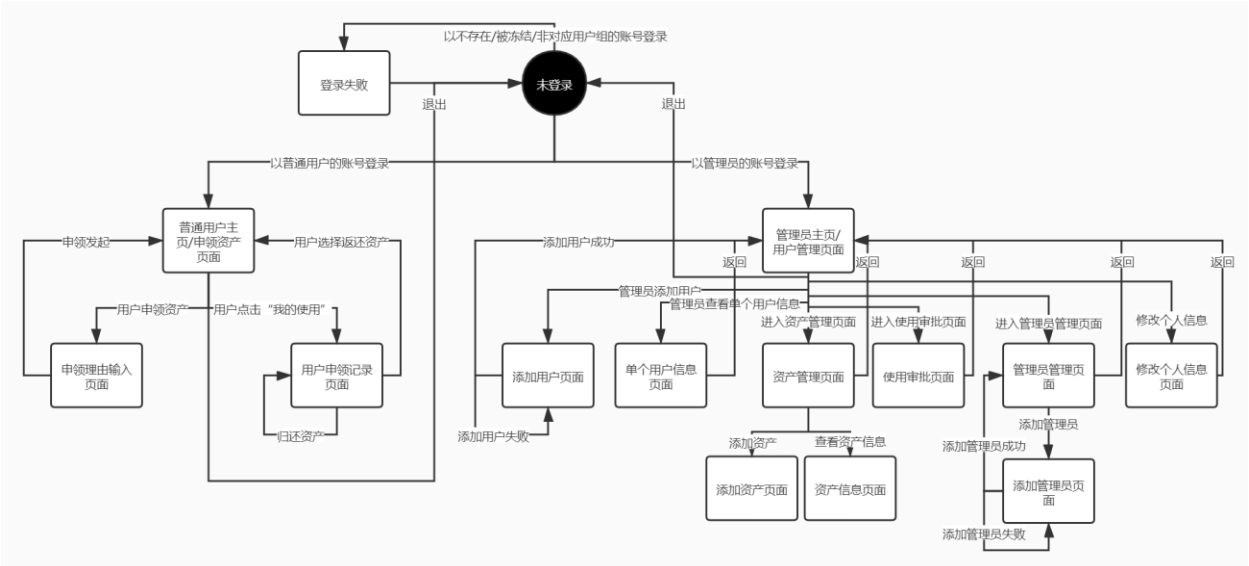

图 2 状态转换图

5 底层数据库设计

5.1 实体类设计

5.1.1 实体类及其属性

Manager: mId(PK), mName(AK), mPassword, mPhone, mEmail, mState;

User: uId(PK), uName(AK), uPassword, uEmail, uState;

Property: pId(PK), pName, pBrand, pModel, pSpec, pTime;

Application: aId(PK), beginTime, reviewTime, endTime, aStatus, operation

5.1.2 实体类图

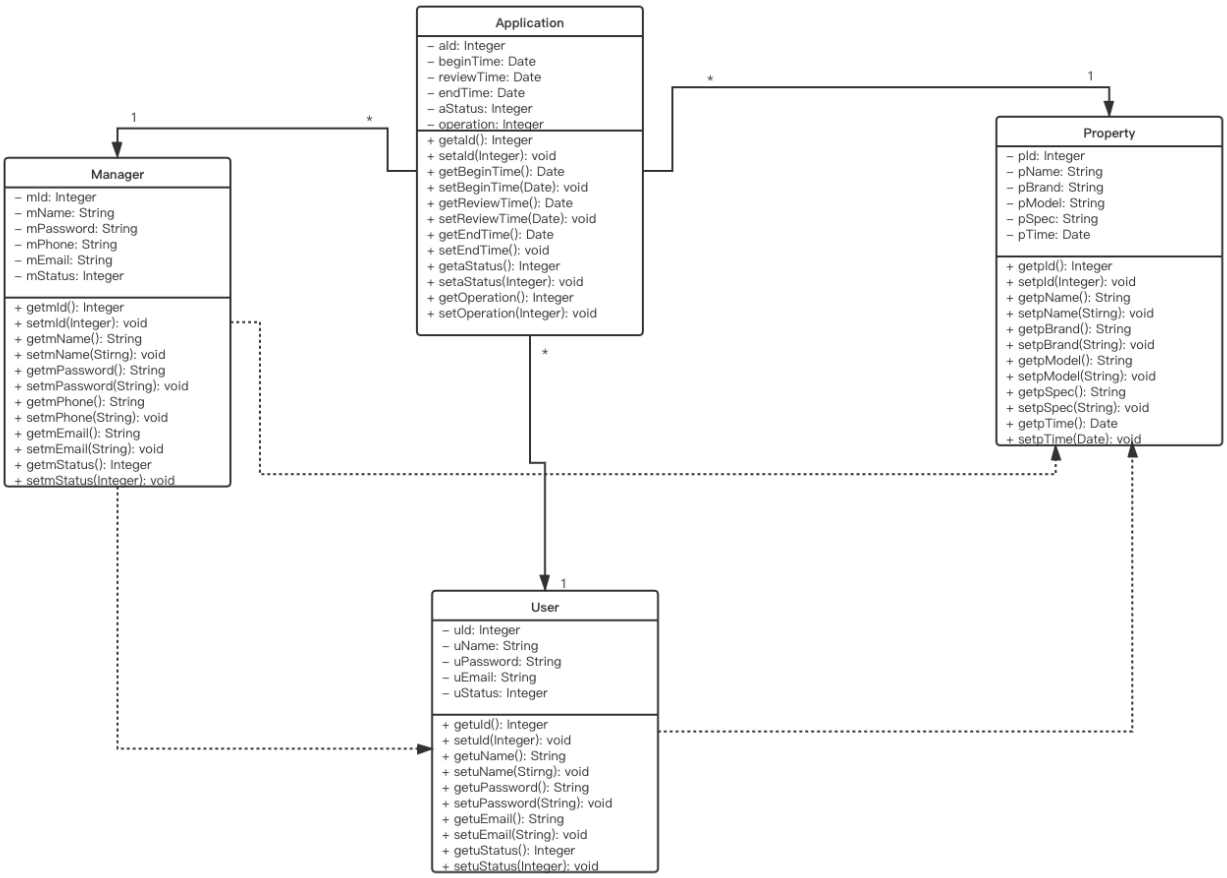

图 3 实体类图

5.2 实体间关系

5.2.1 实体关系图

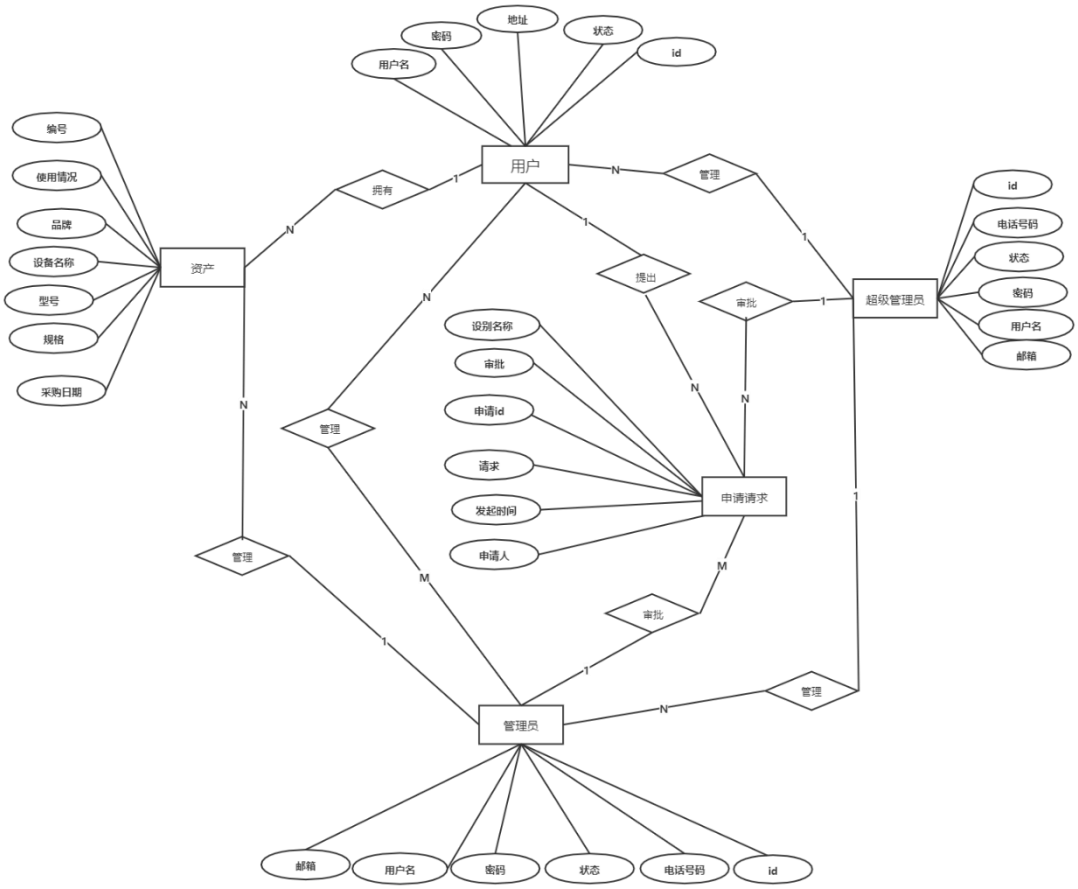

图 4 实体关系图

5.2.2 实体关系

管理员---资产管理：

管理员和资产存在一对多的管理关系，一个管理员可以管理多个资产，可以对资产进行添加和修改；

超级管理员---资产管理：

超级管理员和资产存在一对多的管理关系，一个超级管理员可以管理多个资产，可以对资产

进行添加和信息修改；

管理员---用户管理：

管理员和用户存在多对多的管理关系，管理员可以修改用户的个人资料，对用户进行冻结，删除等操作。单个用户可以被多个管理员管理；

超级管理员---管理员管理：

超级管理员和管理员存在一对多的管理关系，超级管理员可以修改管理员的个人资料，对管理员进行删除等操作。所有管理员只能被一个超级管理员管理；

超级管理员---用户管理：

超级管理员和用户存在一对多的管理关系，超级管理员可以修改用户的个人资料，对用户进行冻结，删除等操作；

用户---资产拥有：

用户和资产存在一对多的关系，一个用户可以拥有多个资产，单个资产只能被一个用户拥有；用户通过申请请求来获取资产，使资产状态成为被拥有；

用户---申请请求提出：

用户和申请请求存在一对多的关系，一个用户可以发起多个请求，通过发起请求用户来获取资产；

管理员---申请请求审批：

管理员和申请请求存在一对多的关系，一个管理员可以审批多个申请请求，一个申请请求只能被一个管理员审批；管理员可以根据申请理由通过或者拒绝申请请求，以此达到对资产进行管理；

超级管理员---申请请求审批：

超级管理员和申请请求存在一对多的关系，一个超级管理员可以审批多个申请请求，一个申

请请求只能被一个超级管理员审批。超级管理员可以通过或者拒绝申请请求，以此达到对资产进行管理。

5.3 数据表属性及其数据字典

表名：管理员 Manager:

属性英文名	属性中文名	数据类型	备注
mId	管理员 ID	Long	
mName	管理员用户名	String	4~16
mPassword	管理员密码	String	4~16
mPhone	管理员手机号	String	11 位手机号码
mEmail	管理员邮箱	String	合法邮箱
mState	管理员状态	boolean	true，false

注：mId：PK

mState：default=true

用户 User:

属性英文名	属性中文名	数据类型	备注
uId	用户 ID	Long	
uName	用户名	String	4~16
uPassword	用户密码	String	4~16
uEmail	用户邮箱	String	合法邮箱
uState	用户状态	Int	0，1，2

注：uId：PK

uState：default=0

资产 Property:

属性英文名	属性中文名	数据类型	备注
pId	资产 ID	Long	
pName	设备名称	String	4~16

pBrand	设备品牌	String	4~16
pModel	设备型号	String	4~16
pSpec	设备规格	String	4~16
pTime	设备采购日期	Date	

注：pId：PK

领用单 Application:

属性英文名	属性中文名	数据类型	备注
aId	领用单号	Long	
beginTime	申请时间	Date	
reviewTime	审核时间	Date	
endTime	归还时间	Date	
aState	审核状态	int	0，1，2
operation	使用状态	int	0，1

注：aId：PK

reviewState：default=0

6 系统架构

系统采用 J2EE 框架 Spring+SpringMVC+Mybatis 组合，其中利用 Spring 的 IOC 和 AOP 实现项目中对 JavaBean 的生命周期的管理、事务控制和日志管理，利用 SpringMVC 进行请求转发，使用 Mybatis 作为持久层操作的框架，三者各司其职、相互配合。同时在目前较为流行的 MVC 架构基础上进行了进一步的扩展，将原有的三层模型继续细分成如图所示七层结构，更加细致的分层使得代码结构更加清楚，层次更加分明，易于扩展和管理，从架构上保证了系统的灵活性、扩展性。

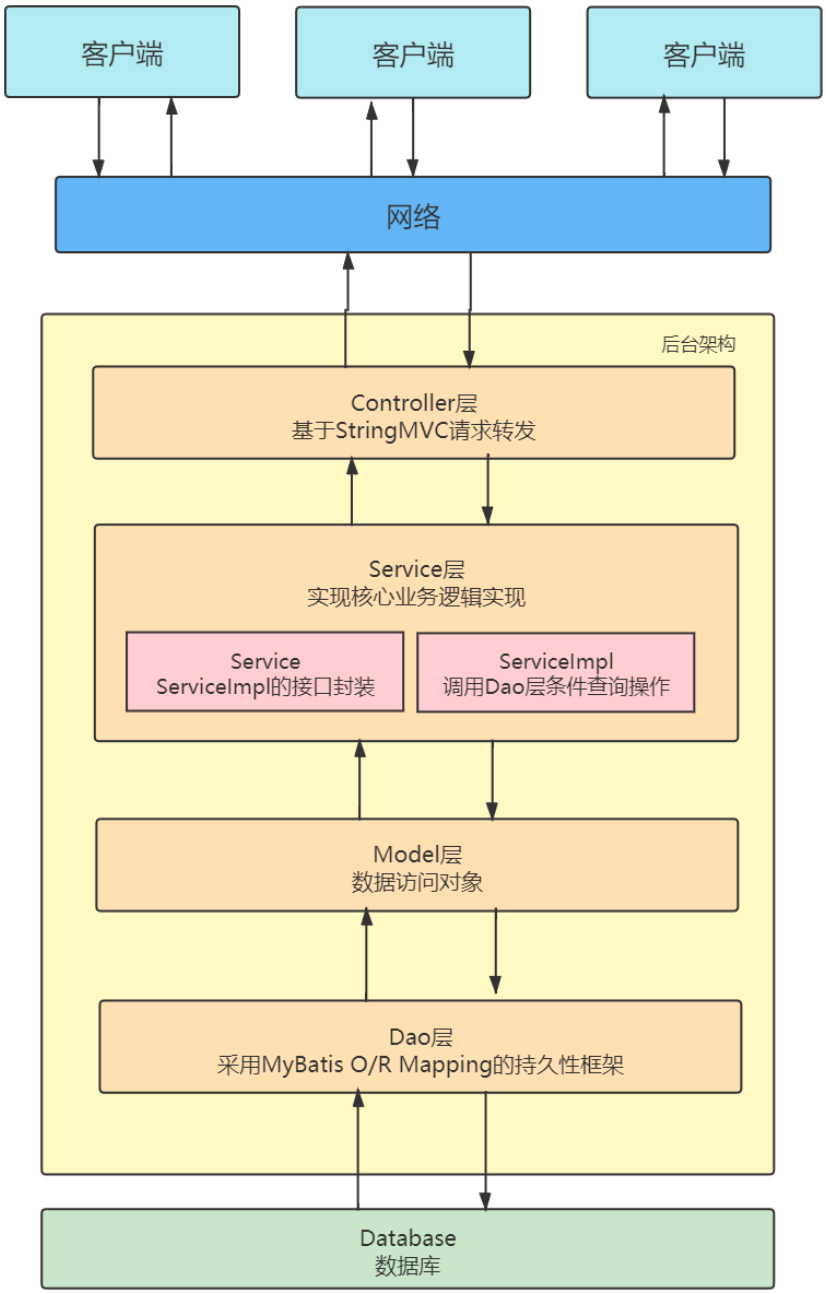

图 5 系统架构图

Controller 层：主要负责接受来自前端的请求，并根据请求内容，调用下层 Service 层相关服务，在下层 Service 层返回后将其返回结果返回给前端请求，完成前后端的数据交互。

Service 层：对前端请求内容的具体实现，通过调用下层 Dao 层获得访问数据库的能力，根据具体的请求内容，通过整合下层 Dao 层接口，实现相关的业务逻辑，为上层 Controller

层提供服务。这一层是整个项目后端的核心部分，所有的业务逻辑代码都在这一层实现。该层包含 Service 和 ServiceImpl 两种定义类型，前者为后者的函数接口封装类，后者包含如下程序文件

Model 层：主要包含项目中通过面向对象的方式建模后所定义的实体类 Entity，在面向对象的建模中对应于现实中的某一实体，同时也与数据库表一一对应。

Dao 层：Dao 层主要负责数据的查询、存储、删除、更新操作，与数据库直接打交道，上层 Service 通过 Dao 层获得访问数据库的能力。Dao 层主要通过 Mybatis 这一持久层框架实现，开发者定义相关数据库操作接口，利用 Mybatis 自带的标签语言可使查询操作更加的灵活，Service 层调用 Dao 层接口，Mybatis 框架会自动维护相关的查询回话，自动完成参数绑定，执行查询操作，在获取到查询结果后，Mybatis 通过对象关系模型自动完成查询结果集到对象的绑定并返回给上层 Service 层，实现数据库查询的底层屏蔽。

7 功能设计

7.1 登录权限管理模块

模块流程图：

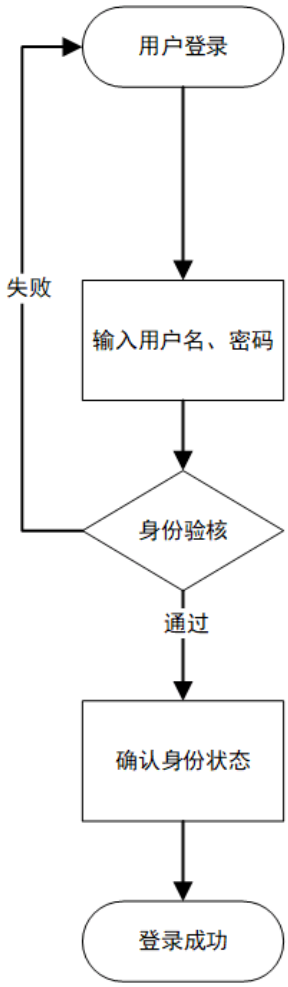

图 6 登录模块流程图

7.1.1 登录管理子模块

- **功能 1：用户登录**

(1) 功能说明

此模块实现了用户登录涉及的身份确认，出错提示，自动跳转等功能，要求用户选择登录身份，输入用户名和密码，如果能够通过身份验核则成功跳转，否则提示“登录失败”。

(2) 输入数据

用户输入的用户名和密码以及用户选择的登录身份（用户或管理员）。

(3) 输出数据

本模块输出为跳转用户登录后应该显示的界面，并更新用户登录状态。

(4) 业务算法和流程

1) 显示用户登录页面，该页面使用 jsp 进行渲染，读取用户输入的用户名和密码以及用户选择的登录方式；

2) 将用户名、密码组合后经过 3 次 base64 加密后按照对应登录方式发送到服务器进行鉴权；

3) 服务器收到请求后对 base64 进行解密并提取其中的用户名和密码；

4) 服务器调用用户查找验证接口，判断用户身份是否有效；

5) 如果用户身份有效服务器生成用户对应的 token，并添加 cookie 发送响应；

6) 如果用户身份无效，服务器发送 302 响应。

(5) 数据设计

用户名、密码：string，网页端采用 base64 三次编码后发回服务器端；

Token：cookie。

(6) 程序设计说明

- AuthController.java 文件，是服务端控制程序；
- login.js 是浏览器端的 js 文件，用于控制浏览器端的登录行为，向服务器发送请求；
- index.jsp 是浏览器端的页面渲染程序，用于获取用户输入的用户名和密码。

(7) 具体函数说明

1) AuthController.java 中相关函数如下所示：

- JsonResponse userLogin(@RequestParam(value="info") String info, HttpServletRequest request, HttpServletResponse response)：普通用户登录；
- JsonResponse managerLogin(@RequestParam String info, HttpServletRequest request, HttpServletResponse response)：管理员登录；

2) login.js 中相关函数如下所示：

- userLogin：普通用户登录。将用户名和密码拼合后经过三次 base64 加密，然后发送给服务器；
- managerLogin: 管理员登录。将用户名和密码拼合后经过三次 base64 加密，然后发送给服务器；
- String loginErrorPage()：用于出错时返回登录失败提示。

- **功能 2：用户退出登录**

(1) 刺激-响应

用户触发退出登录功能，若成功响应，则服务器注销用户的登录状态，并清除用户的 cookie，并提示“退出成功”。

(2) 程序设计说明

AuthController.java 文件，是服务端控制程序。

(3) 具体函数说明

AuthController.java 中相关函数如下所示：

- String userLogout(HttpSession session)：用于普通用户退出登录；
- String managerLogout(HttpSession session)：用于管理员用户退出登录。

7.1.2 视图权限管理子模块

(1) 功能说明

用户不能越级访问管理员功能，且登录前不能访问用户功能。用户和管理员登录后会创建相应 cookie，获得 cookie 对应角色后允许访问角色相对应的功能；

(2) 输入数据

HttpServletRequest 及 HttpServletResponse；

(3) 输出数据

返回布尔值 True 或 False，表明是否正常跳转；

(4) 业务算法和流程

从 http 请求中获得 cookie，并获得对应的角色，根据当前角色及当前访问页面判断是否正常跳转；

(5) 具体函数说明

相关函数如下：

- preHandle（HttpServletRequest request，HttpServletResponse response，Object handler）函数：

  模块的入口，包括 http 请求与相应参数；

- postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView)函数：

  登陆拦截器后处理；

- afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex)：

  登陆拦截器完成操作。

7.2 系统管理模块

7.2.1 管理员账户管理子模块

- **功能 1：添加管理员（仅超级管理员可操作）**

(1) 功能描述

管理员的请求触发后，获取所添加的新管理员的具体账户信息，数据通过审查后存入数据库并更新，生成新的管理员，提示“添加成功”。

(2) 输入数据

1) 添加管理员模块需要五项输入数据：用户名，密码，确认密码，手机，邮箱；

2) 用户名是管理员的唯一标识，为一串字符串，长度为 1-100，用于识别管理员；

3) 密码是长度为 5-20 的字符串，用于核实用户账户信息；

4) 确认密码同为长度 5-20 的字符串,在与密码不同的情况下禁止添加管理员的申请，保证了密码的准确性；

5) 手机号为符合中国手机号标准的 11 位数字组成，是管理员基础信息的一部分，可以通过电话联系管理员；

6) 邮箱为符合邮箱标准的字符串组成，是管理员基础信息的一部分，通过此可以联系和唯一识别管理员。

(3) 输出数据

向后台输出一个封装管理员信息的 JsonManager 的数据，在后端将数据解析转换为具体的管理员类，并存入数据库中。

(4) 数据设计

1) 将添加的管理员信息封装为 JsonManager，通过前端传往后端：

private String info: 封装结果；

List<Integer> ids：管理员列表；

HttpServletRequest HttpRequest：前端用户请求

2) 将请求返回的信息封装为 JsonResponse：

private int status: 页面状态；

private String msg: 页面信息;

3) 将封装后的 JsonManager 转换为 Manager 类,是管理员信息的封装类:

private Integer mId: 用户 id;

private String mName: 用户姓名;

private String mPassword: 密码;

private String mPhone: 手机号;

private String mEmail: 邮箱;

private Integer mStatus: 状态;

(5) 程序文件说明

ManagerController.java 文件，是管理员控制模块主程序。

(6) 具体函数说明

1) ManagerController.java 包含的函数如下:

· public String managerManage(@PathVariable Integer pageNo, @PathVariable Integer pageSz, HttpServletRequest request):

模块的入口，输入页数、尺寸、Http 请求，生成一个 String 字符串与相应的页面;

· public JsonResponse addManager(JsonManager jsonManager):

模块的功能实现函数，通过将 JsonManager 的包装类进行转换后，判断是否添加成功，添加成功后将返回 JsonResponse 的包装类;

- public Manager parse()：

  将 JsonManager 包装类转化为 Manager 类的函数。

2) Manager 类：封装管理员信息的类，包含具体方法如下：

   public Integer getmId()：获取管理员 id；

   public void setmId(Integer mId)：设置管理员 id；

   public String getmName()：获取管理员名称；

   public void setmName(String mName)：设置管理员名称；

   public String getmPassword()：获取管理员密码；

   public void setmPassword(String mPassword)：设置管理员密码；

   public String getmPhone()：获取管理员手机号；

   public String getmEmail()：获取管理员邮箱；

   public void setmEmail(String mEmail)：设置管理员邮箱；

   public Integer getMStatus()：获取管理员状态；

   public void setMStatus(Integer status)：设置管理员状态；

   public String toString()：获取所有信息；

3) JsonManager 类：前端传递的封装管理员信息的类，包含具体方法如下：

   public Manager parse()：将数据转换为 Manager 类；

   public Map<String, String> toMap()：对信息进行加密；

   String getInfo()：获取封装结果；

   public void setInfo(String info)：设置封装结果；

4) JsonResponse 类：请求返回的数据封装的类，包含具体方法如下：

public int getStatus()：获取网页状态；

public void setStatus(int status)：设置网页状态；

public String getMsg()：获取网页信息；

public void setMsg(String msg)：设置网页信息

- **功能 2：删除管理员（仅超级管理员可操作）**

(1) 输入数据

向后台被删除管理员信息，通过识别管理员唯一标准 id，确认用户的管理员身份。

(2) 输出数据

将具体的管理员数据删除后，管理员删除模块输出一个前台请求的 JsonResponse 包装数据，将包装数据转换为具体的页面响应。

(3) 业务算法和流程

从程序模块根据输入的超级管理员信息进行包装，对包装类进行审查，对数据库中的数据进行匹配，拒绝不符合标准的数据的请求。符合标准的请求转换为具体的删除操作，将管理员数据从数据库中删除。将删除结果进行包装后返回，将返回结果转换为具体的页面响应。

(4) 数据设计

与功能 1 设计一致。

(5) 程序文件说明

ManagerController.java 文件，是管理员控制模块主程序。

(6) 具体函数说明

1) ManagerController.java 包含的函数如下：

- public JsonResponse delManager(@RequestBody List<Integer> ids, HttpServletRequest request)：

  模块的功能实现函数，通过解析 HttpServletRequest 的请求获取管理员 id，删除对应 id 的管理员，判断是否删除成功，返回 JsonResponse 的包装类；
- public Manager parse()：将 JsonManager 包装类转化为 Manager 类的函数；
- public boolean remove()：删除列表中的元素；
- public Integer getmId()：获取管理员 id；

2) Manager 类：封装管理员信息的类，包含的相关函数与功能 1 一致；

3) JsonManager 类：前端传递的封装管理员信息的类，包含的相关函数与功能 1 一致；

4) JsonResponse 类：将请求返回的网页的封装类，包含的相关函数与功能 1 一致。

- **功能 3：管理员账户的基本信息修改**

(1) 功能描述

本模块依照管理员输入的个人信息对原信息进行更新，若输入信息合法，则取代原账户信息，并保存到数据库。

(2) 输入数据

- 邮箱信息：合法邮箱地址，英文数字与“@”的集合，例如 123456@example.com；
- 手机信息：合法中国手机号码，11 位数字组合而成。

(3) 输出数据

信息正常更新后，提示“修改成功”。

(4) 数据设计

与功能 1、2 设计一致。

(5) 程序文件说明

ManagerController.java 文件，是管理员控制模块主程序。

(6) 具体函数说明

1) ManagerController.java 包含的函数如下：

- public String managerManage(@PathVariable Integer pageNo, @PathVariable Integer pageSz, HttpServletRequest request)：

  模块的入口，输入页数、尺寸、Http 请求，生成一个 String 字符串与相应的页面；
- public JsonResponse updateSelf(JsonManager manager, HttpServletRequest request)：

  模块的功能实现函数，通过将 JsonManager 的包装类进行转换后，将 manager 的信息进行修改，修改成功后将返回网页请求的 JsonResponse 的包装类。
- public Manager parse()：

  将 JsonManager 包装类转化为 Manager 类的函数；

2) Manager 类：封装管理员信息的类，包含的相关函数与功能 1、2 一致；

3) JsonManager 类：前端传递的封装管理员信息的类，包含的相关函数与功能 1、2 一致；

4) JsonResponse 类：将请求返回的网页的封装类，包含的相关函数与功能 1、2 一致。

- **功能 4：管理员账户的密码修改**

(1) 功能描述

本模块依照超级管理员输入的文本信息对原本的文本信息进行更新。第一次输入新密码后，系统会要求管理员再次输入新密码，如若两次密码经过确认两者相同，则会对原有的密码进行取缔，绑定了新输入的密码信息。上述更新后的密码将会存储到数据库进行保存。

(2) 输入数据

本子模块的功能主要是对自身密码进行修改。

新密码的文本信息内容不限，符合一般可识别字符均可，数字、英文、符号等均予以识别，在两次输入数据相同时，及确定了新的密码，确定了新的输入数据。

(3) 输出数据

新密码在经过两次确定之后会存储到后台的数据库中进行保存，下一次登录时和后台数据库中的文本数据进行比对，若两者相同则管理员账户可成功登录。

(4) 数据设计

与功能 1、2、3 设计一致。

(5) 源程序文件说明

ManagerController.java 文件，是管理员控制模块主程序。

(6) 具体函数说明

1) ManagerController.java 包含的函数如下：

· public String managerManage(@PathVariable Integer pageNo, @PathVariable Integer pageSz, HttpServletRequest request)：

模块的入口，输入页数、尺寸、Http 请求，生成一个 String 字符串与相应的页面；

· public JsonResponse updateSelf(JsonManager manager, HttpServletRequest request)：

模块的功能实现函数，通过将 JsonManager 的包装类进行转换后，将 manager 的信息进行修改，修改成功后将返回网页请求的 JsonResponse 的包装类。

· public Manager parse()：

将 JsonManager 包装类转化为 Manager 类的函数

· public boolean remove()：删除列表中的元素

· public Integer getmId()：获取管理员 id

2) Manager 类：封装管理员信息的类，包含的相关函数与功能 1、2 一致；

3) JsonManager 类：前端传递的封装管理员信息的类，包含的相关函数与功能 1、2 一致；

4) JsonResponse 类：将请求返回的网页的封装类，包含的相关函数与功能 1、2 一致。

7.2.2 企业资产管理子模块

本模块主要涉及资产信息和状态的管理工作，包括资产的增添、使用申请的审批和结束、资产信息和状态的查看和更新以及对用户信息和领用历史的查看，主要分为资产录入模块、使用审批模块、资产查看模块、用户管理模块和资产领用结束模块等子模块。

- **功能 1：资产录入**

(1) 功能描述

本模块实现的是资产录入功能。此项功能由管理员进行操作。管理员输入要添加的资产的相关信息，包括名称、品牌、型号、规格、数量，输入信息之后进行资产添加。

(2) 输入数据

资产相关信息：名称、品牌、型号、规格、数量等。

(3) 输出数据

若输入信息合法，则正常响应，提示“录入成功”。

(4) 业务算法和流程

由管理员输入要添加资产的信息，获取输入的信息之后，在数据库中添加新的资产；新添加的资产的各项属性值是对应的各个输入信息。

(5) 数据设计

1) 将添加的资产信息封装为 JsonProperty，通过前端传往后端：

private String info: 封装结果；

List<Integer> ids：资产列表；

HttpServletRequest HttpRequest：前端用户请求；

2) 将前端操作返回的信息封装为 JsonResponse：

private int status: 页面状态；

private String msg: 页面信息；

3) 将封装后的 JsonProperty 转换为 Property 类,是资产信息的封装类：

private Integer pId：资产 ID；

private String pName：资产名称；

private String pBrand：资产品牌；

private String pModel：资产型号；

private String pSpec：资产规格；

private Date pTime：资产采购日期；

private User user：资产当前的使用用户；

(6) 源程序文件说明

ManagerController.java：管理员管理的 Controller；

JsonProperty.java：前台传来的 Property 数据的封装类；

PropertyManageService.java：资产管理的接口；

PropertyManageServiceImpl.java：资产管理的 Service 层接口实现；

Property.java：资产的类；

(7) 函数说明

1) ManagerController.addProperty：调用下列函数 2、3、4、5、6、7、8，返回资产添加的结果；

2) JsonProperty.toMap：获得输入的资产信息，返回的是资产的名称、品牌、型号、规格、数量；

3) PropertyManageService.addProperty：添加资产，返回添加的结果；如果添加成功，返回值为 true，否则为 false；

4) Property.setpName：设置添加资产的名称；

5) Property.setpBrand：设置添加资产的品牌；

6) Property.setpModel：设置添加资产的型号；

7) Property.setpSpec：设置添加资产的规格；

8) Property.setpTime：设置资产添加的日期。

- **功能 2：使用审批**

(1) 功能描述

本模块实现的是资产申请审批功能。此项功能由管理员进行操作。管理员可以查看资产申请列表，然后根据申请理由对申请进行操作，选择同意或者驳回某个资产领用申请。

(2) 输入数据

进入使用审批界面，查看资产领用申请列表，资产领用申请是按照时间顺序排列；同时，对资产领用申请进行操作，同意某个资产领用申请。

(3) 输出数据

在管理员同意某个资产领用申请之后，该项资产领用申请的状态会改为已通过，在使用审批界面不会显示已经通过的资产领用申请。

(4) 业务算法和流程

首先根据资产领用申请的状态在数据库中进行查询，然后再展示状态是未审批的资产领用申请。管理员选择同意某个资产领用申请之后，在数据库中查找此项资产领用申请，然后将这个资产领用申请的状态改为已通过。

(5) 数据设计

1) applications：List<Application>对象，个体为组成一页的 application 对象数组；

2) 将前端操作返回的信息封装为 JsonResponse：

private int status: 页面状态；

private String msg: 页面信息。

(6) 源程序文件说明

ManagerController.java：管理员管理的 Controller。

ApplicationManageService.java：申请单管理接口。

ApplicationManageServiceImpl.java：申请单管理的 Service 层接口实现。

ApplicationRepo.java：访问 Application 的 JPA 接口。

Application.java：资产领用申请的类。

(7) 函数说明

1) ManagerController.getApplicationManage：调用下列的函数 2、4、5，获取资产领用申请信息，分页展示申请列表；

2) ApplicationManageService.findByPage：分页查询，调用函数 3，返回资产领用申请列表；

3) ApplicationRepo.findAllUnChecked：查询所有未审批的资产领用申请；

4) ApplicationManageService.getNumOfPageByPageAndProperty：通过页面大小和资产来获取页面的数量；

5) ApplicationManageService.findByPageAndProperty：通过资产进行分页查询；

6) ManagerController.reviewApplication：调用下列函数 7、8、9、10、11，完成审资产领用申请申的审批，改变某个申请的状态；

7) ApplicationManageService.updateApplication：更新申请的状态，如果管理员拒绝申请，返回 true，管理员同意则返回 false；

8) ApplicationRepo.findByaId：通过某个申请 ID 来查询资产领用申请；

9) Application.setReviewTime：设置某个资产领用申请审批的时间；

10) Application.setManager：设置审批某个资产领用申请的管理员；

11) Application.setaStatus：设置某个资产领用申请的状态。

- **功能 3：资产查看**

(1) 功能描述

本模块通过获取数据库中所有资产信息，以列表形式进行展现所有资产条目，并可进一步查看单独资产的编号、名称、品牌等有关资产详情和资产的领用历史。

(2) 输入数据

数据库中所有资产信息，包括资产的编号、设备名称、品牌、型号、规格、采购日期、领用情况等相关数据。

(3) 输出数据

所有资产条目的相关信息的列表分页展示以及单独资产的详情信息和领用情况展示。

(4) 业务算法和流程

通过查询数据库获取所有资产的相关信息，传递到前端进行列表分页展示；通过选择特定资产，获取对应 ID 进行数据库查询并前端展示，实现单独资产的详情信息和领用情况的查看操作。

(6) 数据设计

1) properties：List<Property>对象，个体为组成一页的 property 对象数组；

2) applications：List<Application>，个体为针对单独资产组成一页的 application 对象数组；

3) id，name 等一系列资产相关的属性参数，封装为 Property 对象；

4) curPage，sumPage 等有关分页的数据设计。

(7) 源程序文件说明

PrepertyRepo.java：访问 Property 的 JPA 接口；

PropertyManageService.java：资产管理的 Service 接口；

PropertyManageServiceImpl.java：资产管理的 Service 接口的实现类；

ApplicationManageService.java：申请单管理的 Service 接口；

ApplicationManageServiceImpl.java：申请单管理的 Service 接口的实现类；

ManagerfController.java：管理员管理的 Controller；

propertyManage.jsp：资产管理的 jsp 页面；

propertyDetail.jsp：单独资产详情的 jsp 页面；

PMListTag.java：PropertyManage 中列表的 SimpleTag 处理器类；

PDListTag.java：PropertyDetail 中列表的 SimpleTag 处理类。

(8) 函数说明

1) PropertyManageService.getNumOfPage：获取所有资产的页面数量，10 个为一页；

2) PropertyManageService.findByPage：分页查询，按页返回对应的 10 个资产 property 对象；

3) ManagerfController.propertyManage：返回资产详情的视图，调用 PropertyManageService 的 getNumOfPage 和 findByPage 方法得到所有资产的列表对象 properties 以及分页的相关参数，传递给前端 propertyManage 页面；

4) PMListTag.doTag：PropertyManage 中列表的 SimpleTag 的 doTag 方法覆写，

获取参数所有资产 properties，实现每个资产 property 的属性展示；

5) ApplicationManageService.getNumOfPageByPageAndProperty：通过页面大小和资产来获取页面的数量；

6) ApplicationManageService.findByPageAndProperty：通过资产进行分页查询；

7) PropertyManageService.findBypId：通过 id 查找资产；

8) ManagerfController.getPropertyDetail：获取资产详情，调用 PropertyManageService 的 findBypId 和 ApplicationManageService 的 getNumOfPageByPageAndProperty、findByPageAndProperty 方法得到对应资产的详情信息和使用历史信息以及分页的相关参数，传递给前端 propertyDetail 页面；

9) PDListTag.doTag：PropertyDetail 中列表的 SimpleTag 的 doTag 方法覆写，获取参数所有申请 applications，实现单独资产使用历史（领用历史）的信息展示。

- **功能 4：用户查看**

(1) 功能描述

本模块通过获取数据库中所有用户信息，以列表形式进行展现所有用户条目，并可进一步查看单独用户的用户名、邮箱、账户状态等用户详情信息及其资产领用历史。

(2) 业务算法和流程

通过查询数据库获取所有用户的相关信息，传递到前端进行列表分页展示；通过选择特定用户，获取对应 ID 进行数据库查询并前端展示，实现单独用户的详情信息和资产领用历史的查看操作。

(3) 数据设计

1) users：List<User>列表对象，个体为组成一页的 user 对象数组；

2) applications：List<Application>，个体为针对单独用户组成一页的 application 对象数组；

3) id，name 等一系列用户相关的属性参数，封装为 User 对象；

4) curPage，sumPage 等有关分页的数据设计。

(4) 源程序文件说明

UserRepo.java：访问 User 的 JPA 接口；

UserManageService.java：用户管理的 Service 接口；

UserManageServiceImpl.java：用户管理的 Service 接口的实现类；

ApplicationManageService.java：申请单管理的 Service 接口；

ApplicationManageServiceImpl.java：申请单管理的 Service 接口的实现类；

ManagerfController.java：管理员管理的 Controller；

userManage.jsp：用户管理的 jsp 页面；

userDetail.jsp：单独用户详情的 jsp 页面；

UMListTag.java：UserManage 中列表的 SimpleTag 处理器类；

UDListTag.java：UserDetail 中列表的 SimpleTag 处理类。

(5) 函数说明

1) UserManageService.getNumOfPage：获取所有用户的页面数量，10 个为一页；

2) UserManageService.findByPage：分页查询，按页返回对应的 10 个用户 user 对象；

3) ManagerfController.userManage：返回用户管理对应分页下的信息，调用 UserManageService 的 getNumOfPage 和 findByPage 方法得到所有用户的列表对象 users 以及分页的相关参数，传递给前端 userManage 页面；

4) UMListTag.doTag：UserManage 中列表的 SimpleTag 的 doTag 方法覆写，获取参数所有用户 users，实现每个用户 user 的属性展示；

5) UserManageService.findByuId：通过 id 查找用户；

6) ApplicationManageService.getNumOfPageAndUser：通过页面大小和用户来获取页面的数量；

7) ApplicationManageService.findByPageAndUser：通过用户进行分页查询；

8) ManagerfController.getUserDetail：获取用户管理下的用户详情的视图，调用 UserManageService 的 findBypId 和 ApplicationManageService 的 getNumOfPageAndUser、findByPageAndUser 方法得到对应用户的属性信息和使用历史信息以及分页的相关参数，传递给前端 userDetail 页面；

9) UDListTag.doTag：UserDetail 中列表的 SimpleTag 的 doTag 方法覆写，获取参数所有申请 applications，实现单独用户使用历史（领用历史）的信息展示。

- **功能 5：结束资产领用**

(1) 功能描述

本模块针对处于使用状态的资产，进行结束申请、结束领用的信息更新，并更新资产状

态。

(2) 输入数据

需要进行结束领用操作的资产 ID。

(3) 输出数据

数据库和前端有关该资产的状态信息、申请条目信息和相关用户状态信息的更新。

(4) 业务算法和流程

通过选择特定资产，获取对应 ID 进行数据库查询，获得对应的申请条目，继而获得对应的申请用户，最后进行数据库该资产的状态信息、申请条目信息和相关用户状态信息的数据更新。

(6) 数据设计

将前端操作返回的信息封装为 JsonResponse：

private int status: 页面状态；

private String msg: 页面信息。

(7) 源程序文件说明

PrepertyRepo.java：访问 Property 的 JPA 接口；

PropertyManageService.java：资产管理的 Service 接口；

PropertyManageServiceImpl.java：资产管理的 Service 接口的实现类；

ApplicationManageService.java：申请单管理的 Service 接口；

ApplicationManageServiceImpl.java：申请单管理的 Service 接口的实现类；

ManagerfController.java：管理员管理的 Controller；

propertyDetail.jsp：单独资产详情的 jsp 页面。

(8) 函数说明

1) PropertyManageService.findBypId：通过 id 查找资产；

2) PropertyManageService.endApplicationBypId：通过资产 id 查询对应资产，结束资产被领用的状态；

3) ApplicationManageService.findByproperty_pId_on：通过资产 id 查找现行的通过申请；

4) ApplicationManageService.endApplicationByaId：通过申请 id 查找对应申请，将其状态设置为结束；

5) ManagerfController.delApplication：结束领用，通过前端 propertyDetail 传回的对应资产的结束领用触发事件和对应资产 id，调用 PropertyManageService 的 findBypId 和 endApplicationBypId 方法，改变对应资产的被领用状态；调用 ApplicationManageService 的 findByproperty_pId_on 和 endApplicationByaId 方法，结束对应的申请。

7.2.3 普通用户管理子模块

本模块将实现管理员对所有用户的管理及用户的资产领用信息的管理。管理员可以创建新的用户，删除已有用户。

- **功能 5：管理员锁定特定用户的账号**

(1) 功能描述

管理员可以锁定用户的账号，账号锁定后用户仍可正常登录，但登录后仅可以查看申请设备的历史记录，不可以发起申请请求。

(2) 业务算法和流程

锁定账户的 http 请求传到后端之后，返回布尔值 True 或 False，表明是否正常跳转，若正常则返回原页面并锁定指定用户。

(3) 数据设计

返回到页面的数据是 User 类中的属性，是用户信息的封装类：

private Integer uId：用户 id；

private String uName：用户名；

private String uEmail：用户邮箱；

private Integer uStatus：用户账户状态；

(4) 源程序文件说明

- 前端程序位于根目录下的 useManage.jsp 文件，是用户管理页面的前端设计程序；
- UMListTag.java 文件：UserManage 中列表的 SimpleTag 处理器，设计“锁定按钮”，调用 useManage.js 中相关函数，将返回的 request 属性渲染到页面上；
- useManage.js：useManage.jsp 中的 script 部分，包含所用从该前端页面转发数据给后端的函数；

- ManagerController.java 文件：后端程序，是管理员控制模块主程序，调用 UserManageService 中的函数完成用户状态更新；
- UserManageService.java 文件：后端程序，封装 UserManageServiceImpl 的接口；
- UserManageServiceImpl.java 文件：后端程序，包含更新用户状态的实体函数，调用 UserRepo 中的函数完成更新；
- UserRepo.java：后端程序，包含数据库更新函数，运用数据库 update 语句对 User 的存储数据进行更新；
- 用户类是 User.java 文件，是用户信息的封装类。

(5) 函数说明

1) User.java 其中包含的相关函数如下：
   - public Integer getUStatus()：获取用户状态；
   - public Integer setUStatus(Integer uStatus)：设置用户状态；
2) UserManageServiceImpl.java 包含的相关函数如下：
   - public boolean updateUser(Integer id, Integer status)：

     该函数接口封装在 UserManageService.java 中，判断 controller 传来的用户 id 和状态值，排除 user 为空（该用户不存在）等不合法情况，调用 repo 中的 updateStatus 函数；
3) ManagerController 包含的相关函数如下：
   - public JsonResponse updateUser(@PathVariable Integer id, @PathVariable Integer status, HttpServletRequest request)：

     接收前端的请求信息，获取用户 ID 和更新状态（参数），更新用户的状态，

更新失败返回 403，更新成功返回 302。

4) useManage.js 包含的相关函数如下：

· updateUserStatus: function(id, status)：

接收 jsp 传来的注定用户 id 和 status，运用 ajax 转发给 Controller。

5) UMListTag.java 包含的相关函数如下：

· private String generateOp(User user)：

将包括“锁定”在内的状态改变相关的按钮渲染到页面上。

- **功能 1：管理员增加新的系统用户**

(1) 功能描述

管理员可以创建新的用户账户，输入用户的用户名、密码和邮箱的信息，当输入信息合法，即可正常响应，创建新用户。

(2) 输入数据

输入用户名、密码、邮箱。

(3) 输出数据

创建出新的用户，提示“创建成功”。

(4) 源程序文件说明

· Controller 层：ManagerController.java 负责接收来自前端的请求；

· Service 层：UserManageServie.java 是 service 层的接口，UserManageServiceImpl 继承 UserManageServie 接口，完成对前端内容的具体实现；

· Dao 层：接口 UserRepo.java，继承 JpaRepository 类，实现对数据库的操作。

(5) 函数说明

本程序包含的函数如下：

所在文件	函数名称	函数参数	作用
userManage.js	addUser	json	封装为 json 对象，ajax 转发， 并返回状态
ManagerController.java	addUser	jsonUser	接收前端的 json，调用 service 层的 addUser 方法，返回 JsonResponse
UserManageServiceImpl.java	addUser	user	判断用户名是否冲突，若不冲突则添加用户
userRepo.java	save	无	保存用户到数据库

- **功能 2：管理员删除系统用户**

(1) 功能描述

管理员可以删除已有的系统用户，在选定 (可多选) 需要删除的用户后，点击删除按钮，即可成功删除用户。

(2) 触发-响应

选择用户并删除，若正常响应，则提示“删除成功”。

(3) 源程序文件说明

· Controller 层：ManagerController.java 负责接收来自前端的请求。

· Service 层：UserManageServie.java 是 service 层的接口，UserManageServiceImpl 继承 UserManageServie 接口，完成对前端内容的具体实现。

· Dao 层：接口 UserRepo.java，继承 JpaRepository 类，实现对数据库的操作。

(4) 函数说明

本程序包含的函数如下：

所在文件	函数名称	函数参数	作用
userManage.js	deleteUsers	json	封装为 json 对象，ajax 转发，并返回状态
ManagerController.java	delUsers	Ids，request	接收前端传来的 id 列表，for 循环依次删除 user
UserManageServiceImpl.java	delUserByuId	uId	通过 id 删除用户
applicationRepo.java	deleteAll	applications	数据库中删除用户

- **功能 3：管理员修改系统用户的基本信息**

(1) 功能描述

管理员可以修改用户的账户信息。点击对应用户的修改按钮，通过弹出框来修改对应账户信息，点击确定提交修改。

(2) 输入数据

输入需要修改的密码、邮箱。

(3) 源程序文件说明

· Controller 层：ManagerController.java 负责接收来自前端的请求。

· Service 层：UserManageServie.java 是 service 层的接口，UserManageServiceImpl 继承 UserManageServie 接口，完成对前端内容的具体实现。

· Dao 层：接口 UserRepo.java，继承 JpaRepository 类，实现对数据库的操作。

(4) 函数说明

本程序包含的函数如下：

所在文件	函数名称	函数参数	作用
userManage.js	updateUser	json	用户信息封装为 json，ajax 异步请求，若返回状态为 200，则提示更新成功，若返回状态为 403 则提示更新失败
ManagerController.java	updateUser	Integer id, JsonUser jsonUser	调用 service 层来更新用户个人信息，接收 service 返回的操作状态。若成功则设置异步请求的状态为 200，若失败则设置状态为 403
UserManageServiceImpl .java	updateUser	Integer id, User parse	判断此用户的 id 是否存在，信息是否合法。若都合法则更新用户信息
userRepo.java	updateUser	User user	数据库中删除用户

- **功能 4：管理员查看用户的信息以及领用情况**

(1) 功能描述

管理员可查看所有用户的详细信息：ID、用户名、邮箱、账户激活状态等。

(2) 业务算法和流程

前端 http 请求传到后端之后，返回布尔值 True 或 False，表明是否正常跳转，若正常则获取数据库数据并返回到页面上。

(3) 数据设计

返回到页面的数据是 User 类中的属性，是用户信息的封装类：

private Integer uId：用户 id；

private String uName：用户名；

private String uEmail：用户邮箱；

private Integer uStatus：用户账户状态；

(4) 源程序文件说明

- 前端程序位于根目录下的 useManage.jsp 文件，是用户管理页面的前端设计程序；
- UMListTag.java 文件，是 UserManage 中列表的 SimpleTag 处理器，将返回的 request 属性渲染到页面上；
- 后端程序是 ManagerController.java 文件，是管理员控制模块主程序，绑定 request 属性，用于 jsp 渲染；
- 用户类是 User.java 文件，是用户信息的封装类。

(5) 函数说明

1) User.java：封装用户信息的类，其中包含的相关函数如下：

public Integer getmId()：获取用户 id；

public String getuName()：获取用户名称；

public String getuEmail()：获取用户邮箱；

public Integer getUStatus()：获取用户状态；

2) ManagerController 包含的相关函数如下：

- public String userManage(@PathVariable Integer pageNo, @PathVariable Integer pageSz, HttpServletRequest request)：

  模块的入口，参数为页号 pageNo、尺寸 pageSz 和 Http 请求，运用 request.setAttribute 绑定 request 属性，用于 jsp 渲染。

- public String getUserDetail(@PathVariable Integer id, @PathVariable Integer pageNo, @PathVariable Integer pageSz, HttpServletRequest request)：

  参数同上，获取用户管理下的用户详情的 request 属性

3) UMListTag.java 包含的相关函数如下：

- public void doTag() throws JspException, IOException:

  接收 ManagerController 的 request 属性，获取用户信息，渲染显示在用户详情页面上;

- private String getStatus(User user): 获取用户账户状态，返回到页面上。

7.3 普通用户模块

7.3.1 普通用户资产领用

资产领用模块是普通用户在使用本软件时会接触到的功能模块。用户可以经由本模块登录个人的账户，查看可以领用的资产，发起资产领用申请并填写申请理由，查看领用单审批状态、历史记录，以及领用资产的详细信息。

- **功能 1：浏览可用资产**

(1) 功能描述

用户登陆后，在用户登录后的默认页面显示所有可领用的资产及资产的详细信息。

(2) 输入数据

模块不需要用户输入数据，但会自动获取用户之前的登录信息：存储于 session 和

cookie 中的一串 token 字符串，用于标识当前登录的用户。

(3) 输出数据

模块在查询数据库输出所有可以领用的资产及其信息，返回一个含有多个 Property 类（封装资产信息）的数组，并通过 jsp 页面给用户。

(4) 业务算法和流程

系统查询数据库中的资产信息并返回，将属性与 request 绑定后，将数据传入 jsp 与 Tag 类处理并最终显示。

(5) 数据设计

1）存储于 session 中的用户登录信息 Token，为一个 String 类实体；

2）查询传回的 Property 数组 List<Property>，其中 Property 是资产信息的封装类：

private Integer pId：资产 ID，唯一标识符；

private String pName：资产名；

private String pBrand：资产商标；

private String pModel：资产型号；

private String pSpec：资产规格；

private Date pTime：资产采购日期；

private User user：资产当前用户；

3）Property 类中含有的 User 类，封装用户信息：

private Integer uId：用户 ID，唯一标识符；

private String uName：用户名；

private String uPassword：用户密码；

private String uEmail：用户邮箱；

private Integer uStatus：用户状态；

(6) 源程序文件说明

1) UserController.java：

普通用户相关功能的控制类；

2) User.java/ Property.java：

信息封装类；

3) UserPropertyService.java 与 UserPropertyServiceImpl.java：

普通用户查看资产的功能接口与具体实现；

4) UPListTag.java：

SimpleTag 处理类，用于将数据库读出的资产信息写入 jsp 页面；

5) userViewProperties.jsp：

用户查看资产 jsp 页面。

(7) 函数说明

1) UserController.java：普通用户相关功能的控制类，包含的具体相关函数如下：

- public String viewProperties(Integer pageNo,Integer pageSz, HttpServletRequest request)：

  查看可用资产，根据当前页码分页查询资产信息，并写入 jsp 页面返回；

- private User getUser(HttpServletRequest request)：

  辅助工具，用于获取当前登录的用户；

- private int[] getPageInfo(int num, int pageNo, int pageSz)：

  对分页查询进行限定。

2) User.java/ Property.java：信息封装类，包含的具体相关函数如下：

- get/setXXX：变量获取/设置函数，不重复赘述。

3) UserPropertyService.java 与 UserPropertyServiceImpl.java：普通用户查看资产的功能接口与具体实现，包含的具体相关函数如下：

- int getNumOfPage(Integer pageSz)：

  根据一页的条目数，获取资产信息的页面数量；

- List<Property> findByPage(int pageNo, int pageSz)：

  根据页号与条目数分页查询资产信息；

- Property findBypId(Integer pId)：根据 pId 查找资产；

- int getNumOfUnused(Integer pageSz)：

  根据条目数（页面大小）查找未使用的资产；

- List<Property> findUnusedByPage(int pageNo, int pageSz)：

  根据页号与条目数分页查询未使用的资产信息。

4) UPListTag.java：UserProperty 中列表的 SimpleTag 处理类，用于将数据库读出的资产信息写入 jsp 页面，包含的具体相关函数如下：

· public void doTag()：

SimpleTag 具体行为实现，拆封 Properties 的信息并写入 jsp。

- **功能 2：资产领用申请**

(1) 功能描述

用户在可领用的资产页面选择需要领用的资产并填写申请理由，之后提交申请。

(2) 输入数据

1) 模块需要两项输入数据：所申请的资产 ID 和申请理由；

2) 申请的资产 ID 是 Interger 整型，用于确认所申请的资产；

3) 申请理由是一串 UTF-8 编码字符串，用于记录申请理由。

(3) 输出数据

模块向后台输出一个封装申请信息的 JsonApplication 类实体，并在后端将其转换为 Application 类存入数据库。

(4) 业务算法和流程

用户挑选所需要申请的资产，点击申请按钮，系统弹出申请理由对话框，用户填写申请理由后确认。系统根据申请的资产 ID 与理由生成申请并传至后台，之后保存至数据库，最

后返回申请的结果。

(5) 数据设计

1）Property，资产信息的封装类：

private Integer pId：资产 ID，唯一标识符；

private String pName：资产名；

private String pBrand：资产商标；

private String pModel：资产型号；

private String pSpec：资产规格；

private Date pTime：资产采购日期；

private User user：资产当前用户；

2）User，用户信息的封装类：

private Integer uId：用户 ID，唯一标识符；

private String uName：用户名；

private String uPassword：用户密码；

private String uEmail：用户邮箱；

private Integer uStatus：用户状态；

3）Application，申请的资产 ID 和申请理由的封装类：

private Integer aId：申请 ID，唯一标识符；

private Integer pId：申请的资产的 ID；

private String aReason：申请理由；

4）JsonApplication 类，用于将申请信息从前端传往后端：

private String info：封装结果；

(6) 源程序文件说明

1) UserController.java：普通用户相关功能的控制类；

2) User.java：用户类，用作信息封装；

3) Property.java：资产类，用作信息封装；

4) Application.java：申请类，用作信息封装；

5) JsonApplication.java：传递申请信息；

6) UserPropertyService.java 与 UserPropertyServiceImpl.java：

普通用户查看资产的功能接口与具体实现；

7) UserApplicationService.java 与 UserApplicationServiceImpl.java：

普通用户与资产交互的相关功能接口与具体实现；

8) userViewProperties.jsp：用户查看与申请资产的 jsp。

(7) 函数说明

1) UserController.java，包含的具体相关函数如下：

- public JsonResponse applyProperty(Integer id, HttpServletRequest request)：

  资产申请的功能实现，根据资产的状态以及登录用户是否已经在申请资产判断申请结果，成功则封装申请信息保存。最后返回申请结果；

- public Application parse()：

  将 Json Application 包装类转化为 Application 类的函数；

2) UserApplicationService.java，包含的具体相关函数如下：

- boolean existsByUserAndProperty(User curUser, Property property)：根据用户和资产判断用户是否已经存在一个申请；
- public boolean addApplication(Integer operation, User curUser, Property property)：尝试添加申请存入数据库，并返回尝试结果信息。

- **功能 3：申请状态、历史记录与归还**

(1) 功能描述

用户可以查看申请资产领用单的审批状态和自己发出的领用单的历史记录，并将已领用的资产归还。

(2) 输入数据

1) 浏览历史纪录时，模块不需要用户输入数据，但会自动获取用户之前的登录信息：存储于 session 和 cookie 中的一串 token 字符串，用于标识当前登录的用户；
2) 在归还资产时，模块还需要一项输入数据：申请的 ID；
3) 申请 ID 是 Interger 整型，用于确认所归还的资产对应的申请。

(3) 输出数据

1) 浏览历史纪录时，模块查询数据库输出用户的所有申请记录，返回一个含有多个 Application 类（封装申请信息）的数组，并通过 jsp 页面显示给用户；
2) 归还资产时，模块向后台传入申请的 ID 与归还的指令信息。

(4) 业务算法和流程

系统查询数据库中对应用户 ID 的申请信息并返回，将属性与 request 绑定后，将数据

传入 jsp 页面与 Tag 类处理并最终显示。

用户挑选所需要归还的资产申请，点击归还按钮。系统将申请 ID 与归还指令传至后台，之后保存至数据库，释放资产，最后返回归还的结果。

(5) 数据设计

1）Property，资产信息的封装类：

private Integer pId：资产 ID，唯一标识符；

private String pName：资产名；

private String pBrand：资产商标；

private String pModel：资产型号；

private String pSpec：资产规格；

private Date pTime：资产采购日期；

private User user：资产当前用户；

2）User，用户信息的封装类：

private Integer uId：用户 ID，唯一标识符；

private String uName：用户名；

private String uPassword：用户密码；

private String uEmail：用户邮箱；

private Integer uStatus：用户状态；

3）Application，申请的资产 ID 和申请理由的封装类：

private Integer aId：申请 ID，唯一标识符；

private Integer pId：申请的资产的 ID；

private String aReason：申请理由；

(6) 源程序文件说明

UserController.java：普通用户相关功能的控制类；

User.java：用户类，用作信息封装；

Property.java：资产类，用作信息封装；

Application.java：申请单类，用作信息封装；

UserPropertyService.java：普通用户查看资产的功能接口；

UserPropertyServiceImpl.java：UserPropertyService 接口的具体实现；

UserApplicationService.java：普通用户与资产交互的相关功能接口；

UserApplicationServiceImpl.java：UserApplicationService 接口的具体实现。

(7) 函数说明

1) UserController.java 包含的具体相关函数如下：

- public String viewUsed(@PathVariable Integer pageNo, @PathVariable Integer pageSz, HttpServletRequest request)：

  查看申请资产的信息与历史记录的功能实现，根据每页条目数分页显示；

- public JsonResponse returnProperty(@PathVariable Integer id, HttpServletRequest request)：

  归还资产的功能实现，根据传入的资产 id 与 request 中的用户信息尝试归还资产，并返回结果。

2) userPropertyService.java 包含的具体相关函数如下：

- int getNumOfPageByUser(int pageSz, User user)：

  根据用户信息查询申请资产记录的页数。

3) userApplicationService.java 包含的具体相关函数如下：

- List<Application> findByPageAndUser(Integer pageNo, Integer pageSz, String colName, User user)：

  根据页码、页条目数与用户名查询申请记录；

- boolean returnProperty(int aId, int uId)：尝试返还资产。

8 界面设计

8.1 登录界面

图 7.1 登录界面

按照界面提示输入用户名与密码，根据用户的身份选择用户登录或者管理员登录。

8.2 普通用户界面

图 7.2 普通用户主页

普通主页上方为导航栏，下方为详细信息展示，将设备信息展示给了普通员工用户，方便其进行申请领用。

8.3 管理员界面

图 7.3 管理员主页

管理员界面的布局与普通用户界面类似，上方为导航栏，下方为详细信息内容。导航栏有用户管理、资产管理和使用审批三个部分。