企业资产管理系统

软件设计说明

版本：1.0

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

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：“1.0”

文档编号 “ SS-NWPU-EAM-SDI-1.0(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。

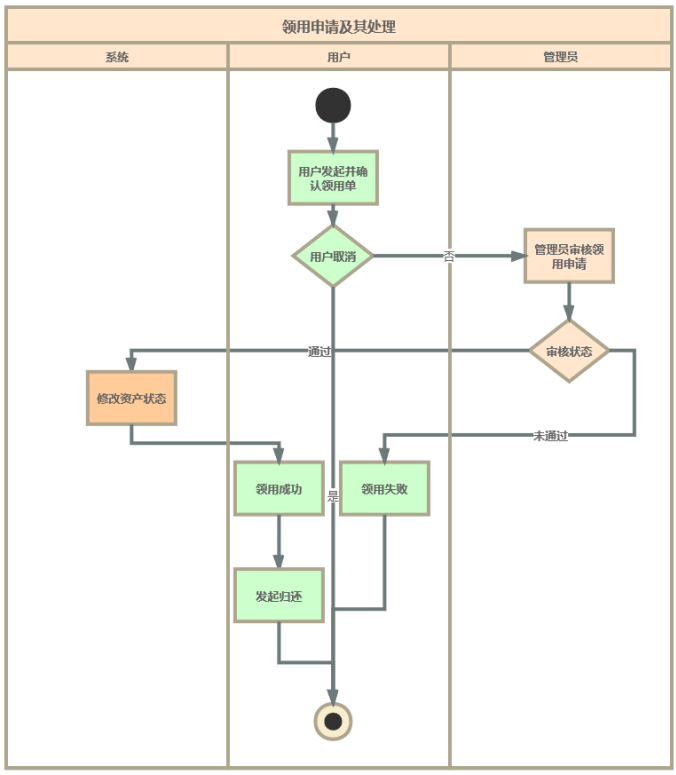
本系统的三大主要功能模块与 23 个功能点如下所示：

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

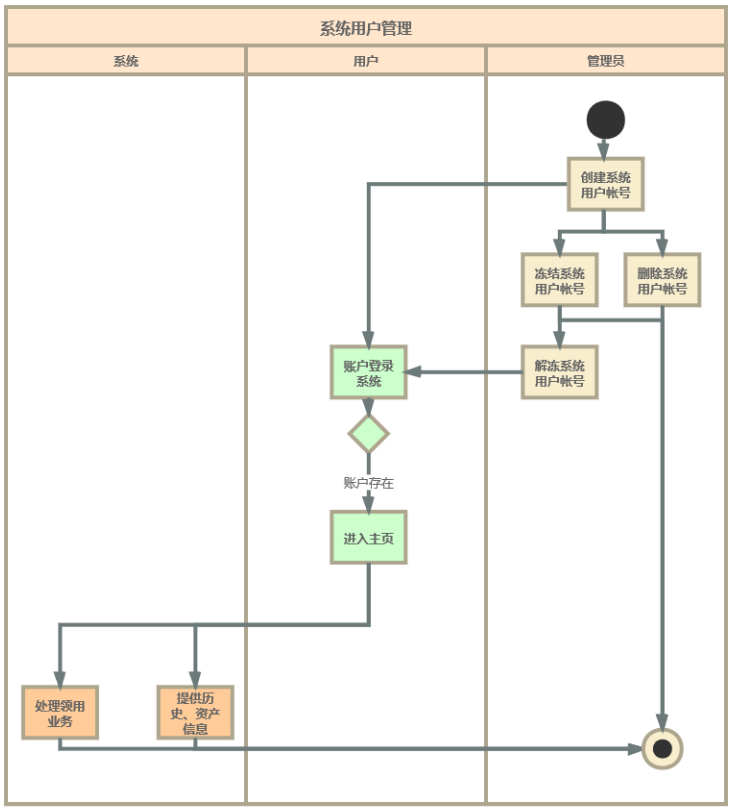
			情况
		Property_5	管理员可以通过结束申请并结束领用，改变资产状态
	普通用户管理	UserManage_1	管理员可以增加新的系统用户
		UserManage_2	管理员可以删除系统用户
		UserManage_3	管理员可以修改系统用户的基本信息
		UserManage_4	管理员可以查看系统用户列表
		UserManage_5	管理员可以修改系统用户的基本信息
		UserManage_6	管理员可以锁定特定用户的账号（锁定后仅可以查看历史记录不可以发起请求）
普通用户模块	资产领用	Apply_1	用户可以浏览可领用的资产列表
		Apply_2	用户可以发起资产领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

本系统的主要业务有资产领用与普通用户管理两部分，流程图分别如下所示：

资产领用功能：本系统核心在于完成企业资产领用的全程管理：



用户管理：该系统普通用户的创建及管理流程如图：

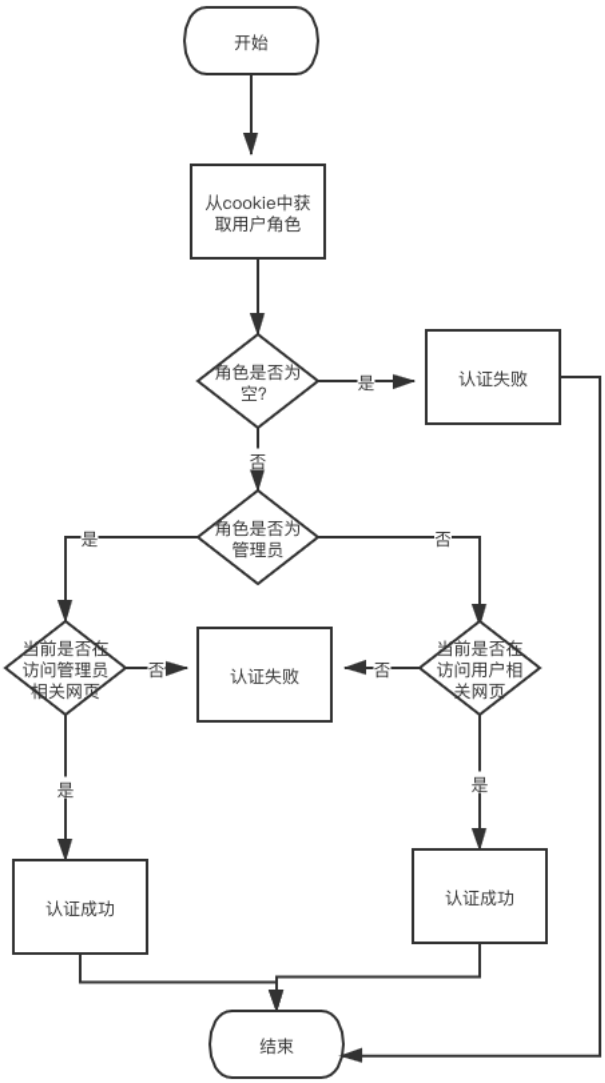


3 功能模块设计

3.1 登录权限管理

3.1.1 视图权限管理

模块分析图



功能说明

用户不能越级访问管理员功能且未登录则不能访问页面。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视图和功能。其中用户不能越级访问管理员功能。

(1) 设计图

本功能的处理过程如图所示：

(1) 功能描述

根据当前登陆角色及当前访问页面判断是否正常跳转，若当前没有登录角色或登录角色与访问页面不符则不允许访问页面。

(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) 函数说明

本程序包含的函数如下：

1) preHandle（HttpServletRequest request，HttpServletResponse response，

Object handler）函数：

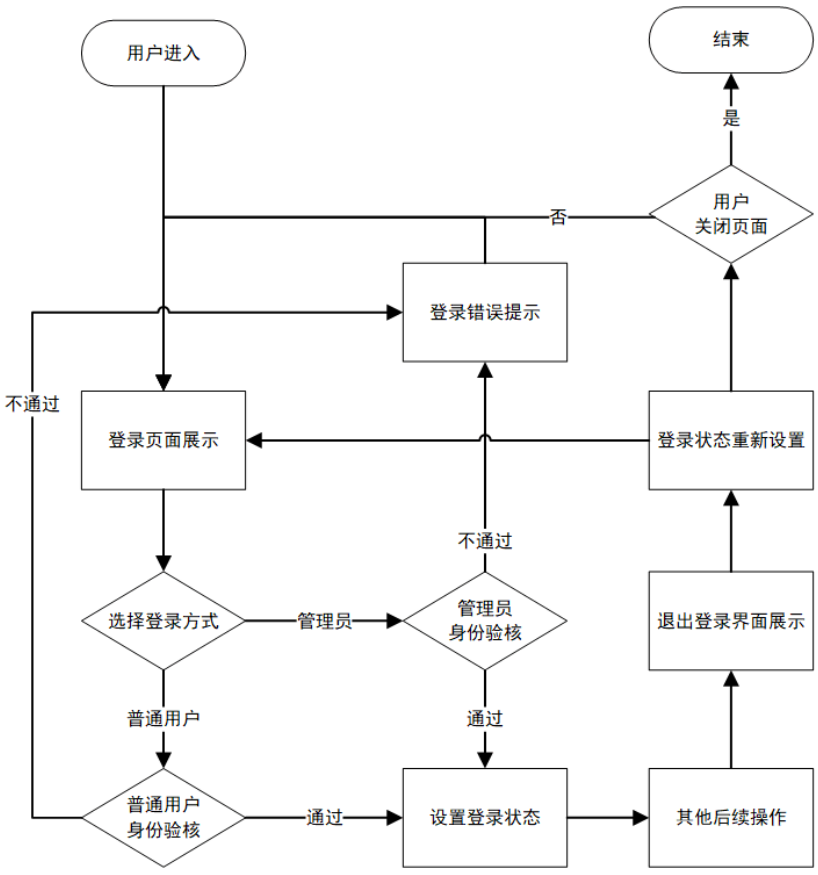
模块的入口，包括 http 请求与相应参数。

2）postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView)函数：登陆拦截器后处理。

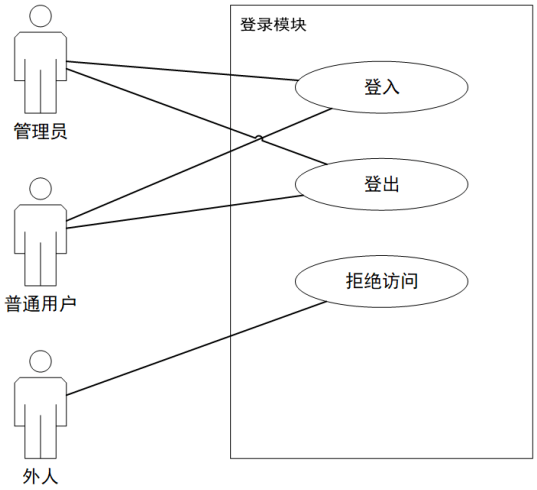
3）afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex)：登陆拦截器完成操作。

3.1.2 登录管理模块

模块流程图



模块用例图



- **子功能 1：用户登录模块**

(1) 功能描述

此模块实现了用户登录所必须的前端和后端，包括页面展示，身份确认，出错提示，自动跳转等功能。本模块要求用户输入用户名和密码并选择登录方式，如果能够通过身份验核则跳转到对应的界面，如果失败则给出提示并返回最初的登录页面。

(2) 输入数据

本模块接受用户输入的用户名和密码以及用户选择的登录方式。

(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

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

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。

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。

服务器调用用户查找验证接口，判断用户身份是否有效。

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。

如果用户身份无效，服务器发送 302 响应。

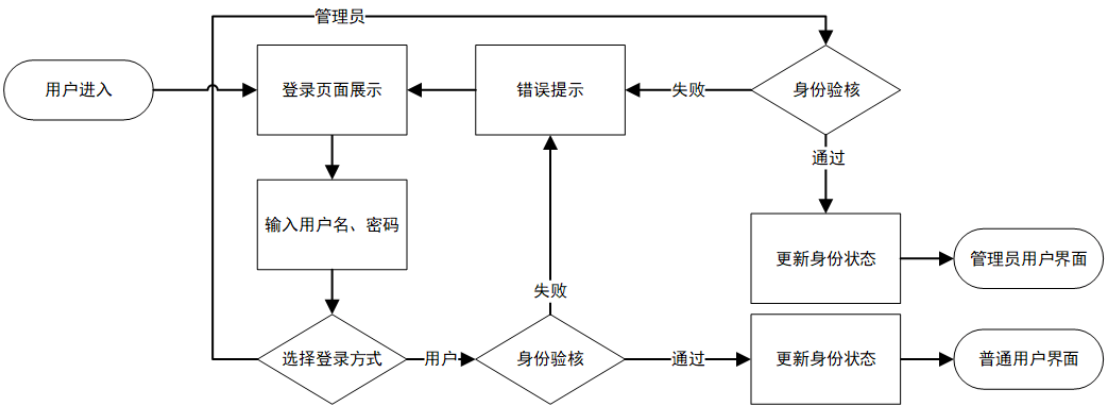
页面收到响应后进行相应跳转。

(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) 设计图



(7) 源程序文件说明

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 是浏览器端的登录页面，用于提供用户登录的 GUI，并获取用户输入的用户名和密码。

(8) 函数说明

1) AuthController.java

JsonResponse userLogin(@RequestParam(value="info") String info, HttpServletRequest request, HttpServletResponse response)：用于普通用户登录。

JsonResponse managerLogin(@RequestParam String info, HttpServletRequest request, HttpServletResponse response)：用于管理员登录

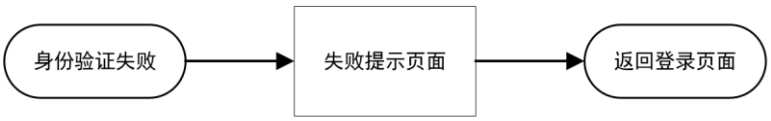
2) login.js

userLogin：用于普通用户登录。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: 用于管理员用户登录。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。

- **子功能 2：用户登录失败提示功能**

(1) 流程图



(2) 功能描述

该模块实现了用户输入用户名和密码错误时的跳转提示，包括动态效果和自动跳转。

(3) 输入数据

本模块输入由用户登录失败触发。

(4) 输出数据

本模块输出为失败提示页面。

(5) 业务算法和流程

用户触发登录失败。

使用 jsp 实现动态动画并展示登录失败的文字提示。

跳转回登录页面。

(6) 源程序文件说明

AuthController.java 文件，是服务端控制程序。

loginError.jsp 是浏览器端的错误展示页面，用于提供用户登录的 GUI，并获取用户输入的

用户名和密码。

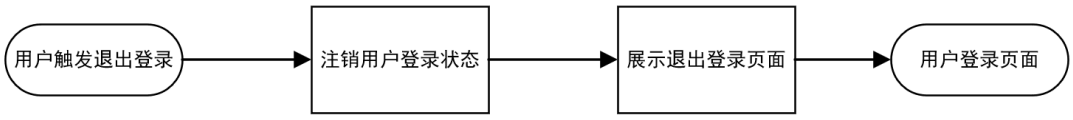
(7) 函数说明

AuthController.java

String loginErrorPage()：用于出错时返回登录失败页面。返回的是 loginError.jsp。

- **子功能 3：用户退出登录功能**

(1) 设计图



(2) 功能描述

该模块实现了用户登录状态的注销和用户退出登录页面的展示。

(3) 输入数据

该模块输入为用户触发退出登录。

(4) 输出数据

该模块输出为用户退出登录页面

(5) 业务算法和流程

用户触发退出登录。

服务器注销用户的登录状态，并清除用户的 cookie。

展示退出登录的页面。

跳转回登录页面。

(6) 源程序文件说明

AuthController.java 文件，是服务端控制程序。

loginError.jsp 是浏览器端的错误展示页面，用于提供用户登录的 GUI，并获取用户输入的用户名和密码。

(7) 函数说明

AuthController.java

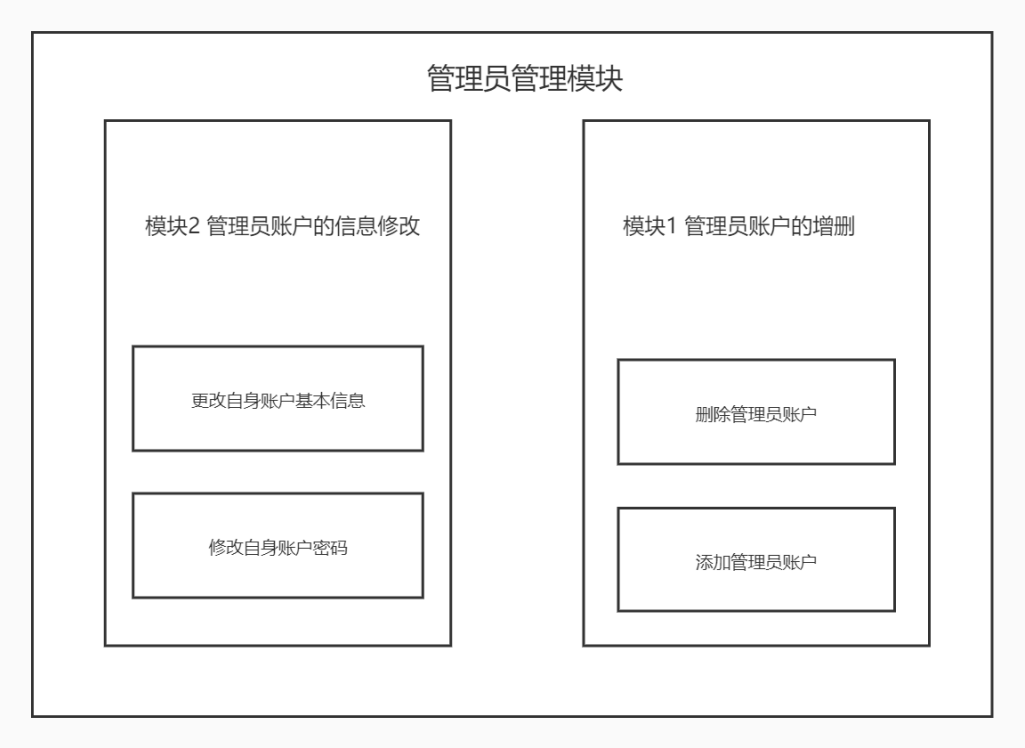
String userLogout(HttpSession session)：用于普通用户退出登录

String managerLogout(HttpSession session)：用于管理员用户退出登录

3.2 系统管理模块

3.2.1 管理员账户管理

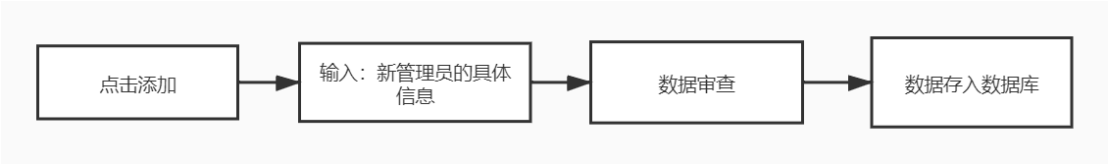
管理员账户管理模块分析图：



- **子功能 1：添加管理员功能**

(1) 设计图

本功能的处理过程如图所示：



(2) 功能描述

本模块通过触发管理员的请求，获取新管理员的具体账户，密码，邮箱，并对数据进行相应的审查，通过审查后将数据存入数据库并更新，生成新的管理员。将通过管理员的响应进行包装，将包装的响应转化为具体页面的响应。

(3) 输入数据

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；

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。

密码是长度为 5-20 的字符串，用于核实用户账户信息。

确认密码为长度 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。

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，通过此可以联系管理员。

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。

(4) 输出数据

管理员增添模块输出一个前台包装管理员的 JsonManager 的包装数据，将包装数据转换为具体的管理员类，并存入数据库中。并且将请求的响应包装，返回至前端转化为响应的页面。

(5) 业务算法和流程

从程序模块根据输入的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管理员类，并存入数据库中，拒绝不符合标准的请求。将请求的响应包装，返回至前端转化为响应的页面。

(6) 数据设计

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

将请求返回的网页封装为 JsonResponse；

private int status; 页面状态

private String msg; 页面信息

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*

(6) 源程序文件说明

本程序位于根目录下的 ManagerController.java 文件，是管理员控制模块主程序。

(7) 函数说明

本程序包含的函数如下：

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

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

2） 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，添加成功后将返回网页请求的 JsonResponse 的包装类。

3）public Manager parse()：

将 JsonManager 包装类转化为 Manager 类的函数

本程序包含的函数类如下：

1）Manager 类：封装管理员信息的类。，

其中包含的函数如下：

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

2）JsonManager 类：前端传递的封装管理员信息的类：

public Manager parse() 转换为 Manager

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

3）JsonResponse 类：将请求返回的网页的封装类：

将请求返回的网页封装为 JsonResponse

public int getStatus() 获取网页状态

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

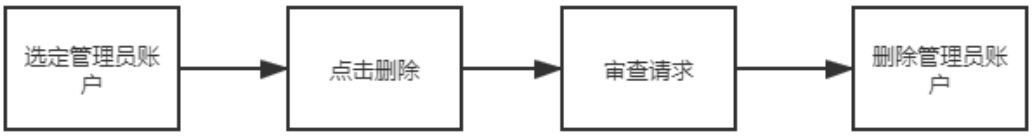
public String getMsg()获取网页信息

public void setMsg(String msg)设置网页信息

- **子功能 2：删除管理员功能**

(1) 设计图

本模块的处理过程如图所示：



(2) 功能描述

本模块通过超级管理员的请求，审查删除管理员的请求，删除管理员。

(3) 输入数据

输入：管理员用户名，被删除管理员用户名。通过识别管理员唯一标准 id，确认用户的管理员身份。并且通过被删除管理员用户名，确定被删除管理员的身份。

(4) 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。

(5) 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。

(6) 数据设计

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

管理员列表 ids 为 List<Integer>属性

Http 请求 request 为 HttpServletRequest 属性

将请求返回的网页封装为 JsonResponse；

private int status; 页面状态

private String msg; 页面信息

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

(7) 源程序文件说明

本程序位于根目录下的 ManagerController.java 文件，是管理员控制模块主程序。

(8) 函数说明

本程序包含的函数如下：

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

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

2) 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 ids 中相同 id 的管理员，判断是否删除成功，，返回网页请求的 JsonResponse 的包装类。

3）public Manager parse()：

将 JsonManager 包装类转化为 Manager 类的函数

4）public boolean remove()

删除列表中的元素

5）public Integer getmId()

获取管理员 id

本程序包含的函数类如下：

1）Manager 类：封装管理员信息的类。,

其中包含的函数如下：

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

2）JsonManager 类：前端传递的封装管理员信息的类：

public Manager parse() 转换为 Manager

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

3）JsonResponse 类：将请求返回的网页的封装类：

将请求返回的网页封装为 JsonResponse

public int getStatus() 获取网页状态

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

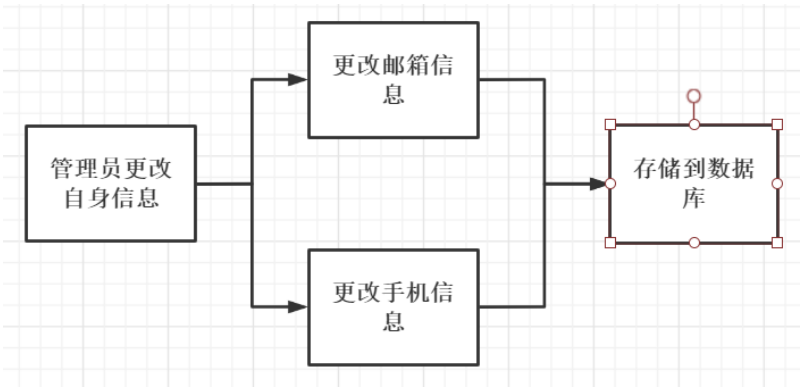
public String getMsg()获取网页信息

public void setMsg(String msg)设置网页信息

- **子功能 3：管理员账户的基本信息修改**

(1) 设计图

本模块的处理过程如图所示：



(2) 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。在经过判定后，输入的有效邮箱信息会取代了原有的邮箱信息，同理，手机信息经过验证码的验证后，会对原有的手机号进行取缔，绑定了新输入的手机信息。上述更新后的文本信息将会存储到数据库进行保存。

(3) 输入数据

管理员账户信息的修改主要分为两部分：一部分是邮箱信息，另一部分是手机信息。

两部分信息均为一串文字，邮箱信息可以是英文数字与“@”的集合，例如“123456@example.com”；其代表了更改后的邮箱信息，加强了有一定的安全性。

而手机信息则是管理员可使用的手机号，由 11 位数字组合而成，亦具有一定的提高安全性的能力。

(4) 输出数据

数据经超级管理员输入后存储到后台的数据库中，数据库可以对后台的数据进行清洗或者存储的操作，包括删除、修改、添加邮箱或者手机的信息。

(5) 业务算法和流程

从程序模块根据输入的信息对相应的模块进行更新迭代，例如从邮箱信息程序模块输入“123456@example.com”，即将其存储于数据库中，对原有的邮箱信息删除后添加新输入的信息。最终在后台的数据库中进行储存。

(6) 数据设计

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

将请求返回的网页封装为 JsonResponse；

private int status; 页面状态

private String msg; 页面信息

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*

(7) 源程序文件说明

本程序位于根目录下的 ManagerController.java 文件，是管理员控制模块主程序。

(8) 函数说明

本程序包含的函数如下：

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

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

2) 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，添加成功后将返回网页请求的 JsonResponse 的包装类。

3）public Manager parse()：

将 JsonManager 包装类转化为 Manager 类的函数

本程序包含的函数类如下：

1）Manager 类：封装管理员信息的类。，

其中包含的函数如下：

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

2）JsonManager 类：前端传递的封装管理员信息的类：

public Manager parse() 转换为 Manager

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

3）JsonResponse 类：将请求返回的网页的封装类：

将请求返回的网页封装为 JsonResponse

public int getStatus() 获取网页状态

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

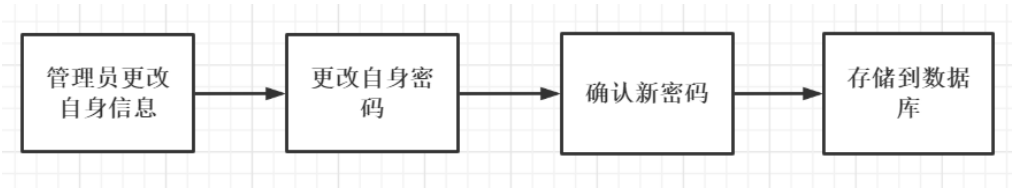
public String getMsg()获取网页信息

public void setMsg(String msg)设置网页信息

- **子功能 4：管理员账户的密码修改**

(1) 设计图

本模块的处理过程如图所示：



(2) 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。

(3) 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。

(4) 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。

(5) 业务算法和流程

从密码文本信息模块输入的信息经过两次比对核验后存储于数据库之中，对原有的密码数据进行删除，最终在后台的数据库中进行存储，下次登录时将新密码与输入密码进行比对，两者相同则管理员账户可成功登录。

(6) 数据设计

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

管理员列表 ids 为 List<Integer>属性

Http 请求 request 为 HttpServletRequest 属性

将请求返回的网页封装为 JsonResponse；

private int status; 页面状态

private String msg; 页面信息

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*

(7) 源程序文件说明

本程序位于根目录下的 ManagerController.java 文件，是管理员控制模块主程序。

(8) 函数说明

本程序包含的函数如下：

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

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

2 ） 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 ids 中相同 id 的管理员，判断是否删除成功，，返回网页请求的 JsonResponse 的包装类。

3）public Manager parse()：

将 JsonManager 包装类转化为 Manager 类的函数

4）public boolean remove()

删除列表中的元素

5）public Integer getmId()

获取管理员 id

本程序包含的函数类如下：

1）Manager 类：封装管理员信息的类。，

其中包含的函数如下：

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

2）JsonManager 类：前端传递的封装管理员信息的类：

public Manager parse() 转换为 Manager

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

3）JsonResponse 类：将请求返回的网页的封装类：

将请求返回的网页封装为 JsonResponse

public int getStatus() 获取网页状态

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

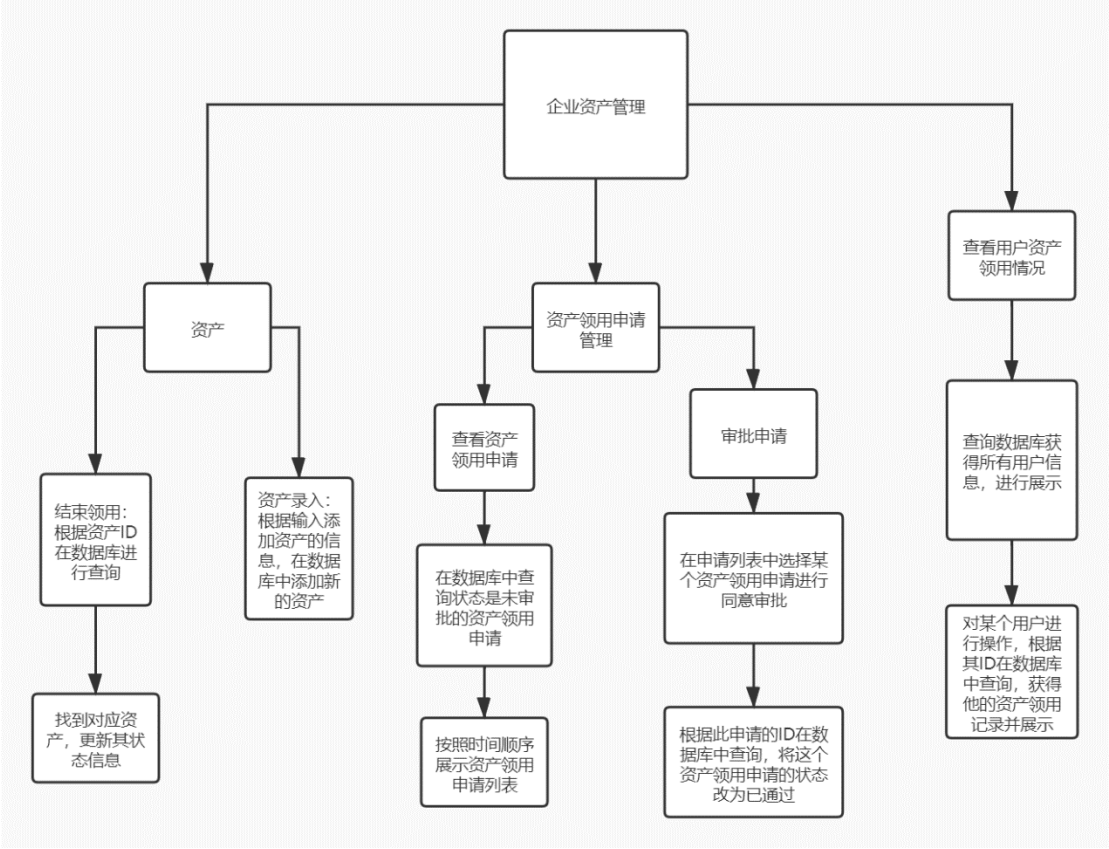
public String getMsg()获取网页信息

public void setMsg(String msg)设置网页信息

3.2.2 企业资产管理

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。

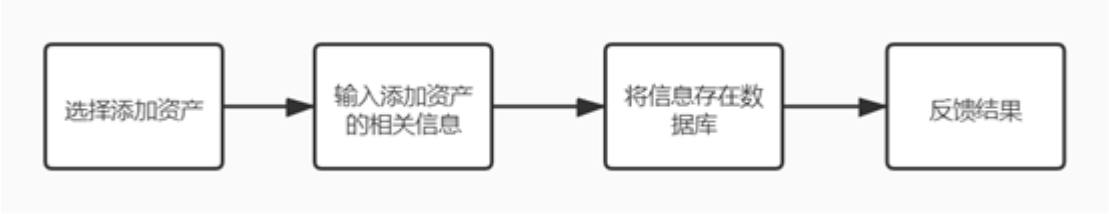
模块分析图



- **子功能 1：资产录入功能**

(1) 设计图

本模块的处理过程如图所示：



(2) 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。

(3) 输入数据

管理员选择添加资产之后，填写资产的相关信息，相关信息有名称、品牌、型号、规格、数量，这些信息作为输入。

(4) 输出数据

在填写资产的名称、品牌、型号、规格、数量等相关信息之后，如果输入信息符合规定，就将资产信息添加到数据库。

(5) 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。

(6) 数据设计

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。

(7) 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。

(8) 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。

- **子功能 2：使用审批功能**

(1) 设计图

本模块的处理过程如图所示：

(2) 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，然后对申请进行操作，可以选择同意某个资产领用申请。

(3) 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。

(4) 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。

(5) 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。

(6) 数据设计

private Integer aId：申请单的 ID。

private User user：使用的用户。

private Property property：领用的资产。

private Manager manager：审批的管理员。

private Date beginTime：开始时间。

private Date reviewTime：审批时间。

private Date endTime：结束时间。

private Integer aStatus：申请单状态。

private Integer operation：管理员的操作。

(7) 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。

(8) 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.findById：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。

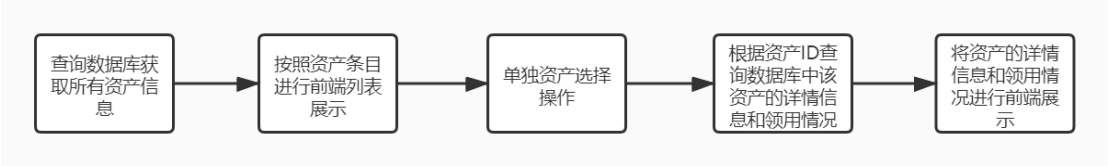
⑩ Application.setManager：设置审批某个资产领用申请的管理员。

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

- **子功能 3：资产查看功能**

(1) 设计图

本模块的处理过程如图所示：



(2) 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。

(3) 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。

(4) 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。

(5) 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

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

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

另外还有 id，name 等一系列资产相关的属性参数和 curPage，sumPage 等有关分页的数据。

(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。

- **子功能 4：用户查看功能**

(1) 设计图

此模块的功能结构如下所示：

(2) 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。

(3) 输入数据

数据库中所有用户信息，包括用户的用户名、邮箱、账户状态等用户详情信息及其资产领用历史等相关数据。

(4) 输出数据

所有用户条目的相关信息的列表分页展示以及单独用户的详情信息和资产领用历史展示。

(5) 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。

(6) 数据设计

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

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

另外还有 id，name 等一系列用户相关的属性参数和 curPage，sumPage 等有关分页的数据。

(7) 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。

(8) 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。

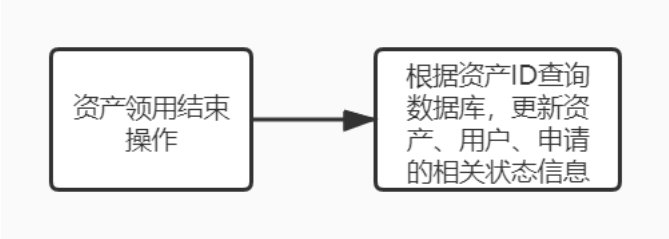
⑧ ManagerfController.getUserDetail：获取用户管理下的用户详情的视图，调用 UserManageService 的 findBypId 和 ApplicationManageService 的 getNumOfPageAndUser、findByPageAndUser 方法得到对应用户的属性信息和使用历史信息以及分页的相关参数，传递给前端 userDetail 页面。

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。

- **子功能 5：资产领用结束功能**

(1) 设计图

此模块的功能结构如下所示：



(2) 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。

(3) 输入数据

需要进行结束领用操作的资产 ID。

(4) 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。

(5) 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

无。

(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。

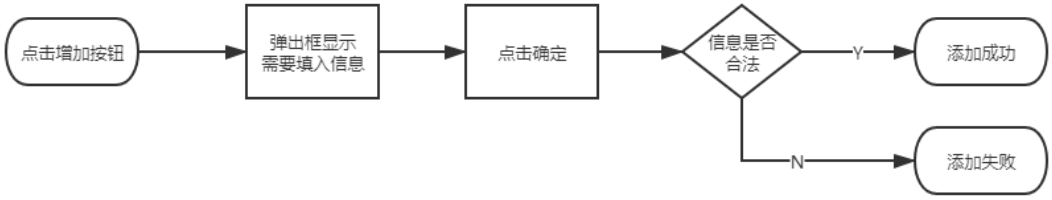
3.2.3 普通用户管理

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。

- **子功能 1 管理员增加新的系统用户**

(1) 设计图

本模块的处理过程如图所示：



(2) 功能描述

管理员可以创建新的用户账户。通过点击“增加”按钮，弹出框显示输入信息，管理员输入用户名、密码和邮箱的信息后，点击确定，当输入信息无误时，即可创建新用户。

(3) 输入数据

输入用户名、密码、邮箱。

(4) 输出数据

创建出新的用户。

(5) 业务算法和流程

根据输入的用户信息进行包装，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用户类，并存入数据库中，拒绝不

符合标准的请求。将请求的响应包装，返回至前端转化为响应的页面。

(6) 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。

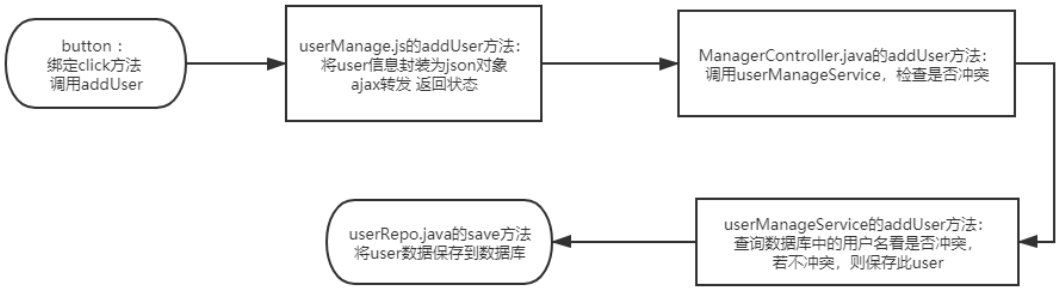
Dao 层：接口 UserRepo.java，继承 JpaRepository 类，实现对数据库的操作。

(7) 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

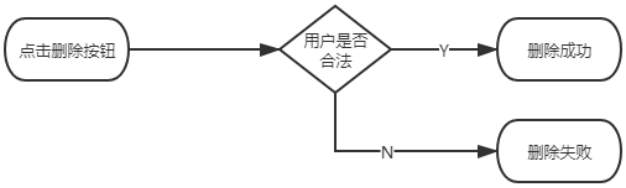
(8) 详细流程图



- **子功能 2 管理员删除系统用户**

(1) 设计图

本模块的处理过程如图所示：



(2) 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。

(3) 触发-响应

触发：选择用户并点击删除；

响应：提示删除成功。

(4) 业务算法和流程

从程序模块根据输入的用户信息进行包装，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，判断此用户是否存在，若存在则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。

(5) 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。

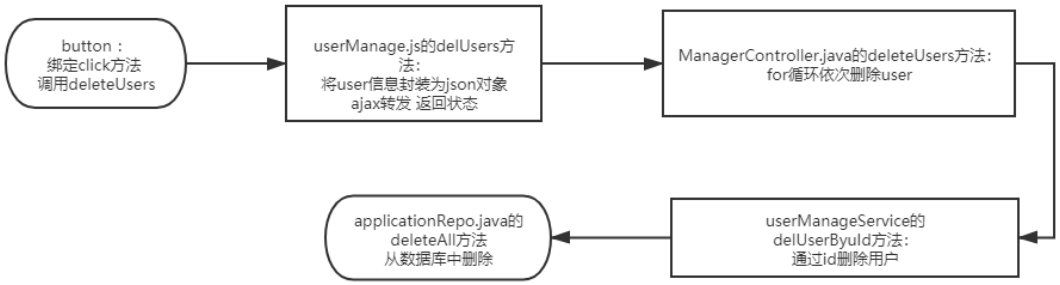
Dao 层：接口 UserRepo.java，继承 JpaRepository 类，实现对数据库的操作。

(6) 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

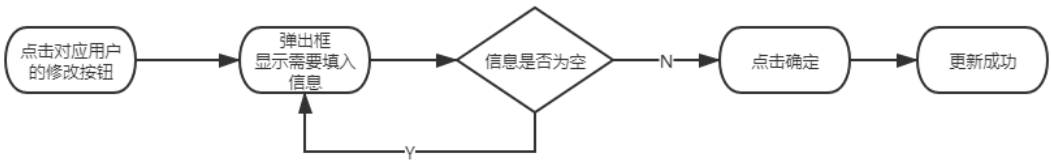
(7) 详细流程图



- **子功能 3 管理员修改系统用户的基本信息**

(1) 设计图

本模块的处理过程如图所示：



(2) 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。

(3) 输入数据

输入需要修改的密码、邮箱。

(4) 输出数据

无。

(5) 业务算法和流程

根据输入的信息对相应的数据进行更新，根据前端输入的信息，判断信息是否为空，当信息非空时在数据库中对原有的信息删除，然后添加新输入的信息。

(6) 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。

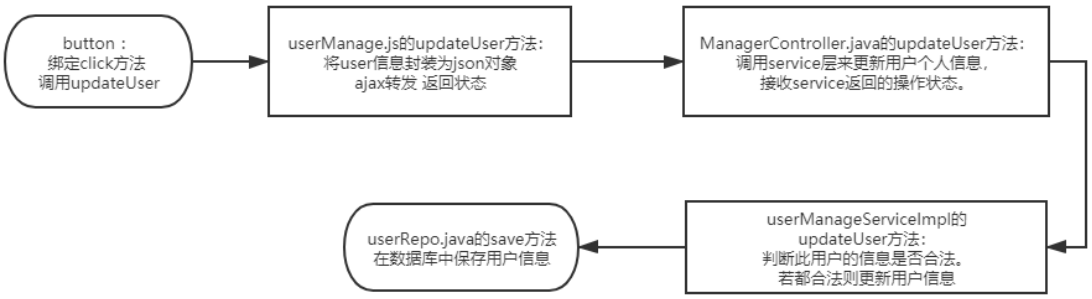
Dao 层：接口 UserRepo.java，继承 JpaRepository 类，实现对数据库的操作。

(7) 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

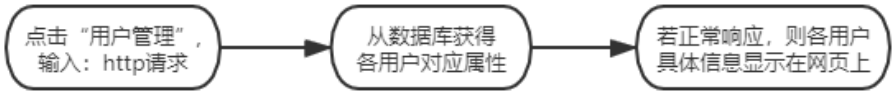
(8) 详细流程图



- **子功能 4 管理员可以查看用户的信息以及领用情况**

(1) 设计图

本功能的处理过程如图所示：



(2) 功能描述

管理员登陆后将自动跳转至“用户管理”界面，可以清晰看到所有用户的详细信息：ID、用户名、邮箱、账户激活状态等。

(3) 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。

(4) 输入数据

无。

(5) 输出数据

用户的详细信息：ID、用户名、邮箱、账户激活状态输出显示在页面列表上。

(6) 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

(7) 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。

(8) 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，页面大小 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。

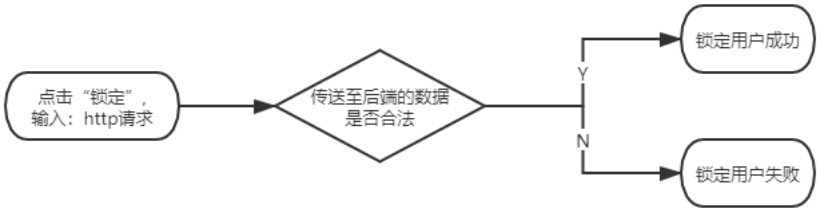
private String getStatus(User user)

获取用户账户状态，返回到页面上。

- **子功能 5 管理员可以锁定特定用户的账号**

(1) 设计图

本功能的处理过程如图所示：



(2) 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。

(3) 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。

(4) 输入数据

点击“锁定”按钮后传入后端的 http 请求。

(5) 输出数据

若正常响应，则成功锁定指定用户，页面上显示“锁定成功”提示。

(6) 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

(7) 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。

(8) 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。

3.3 普通用户模块

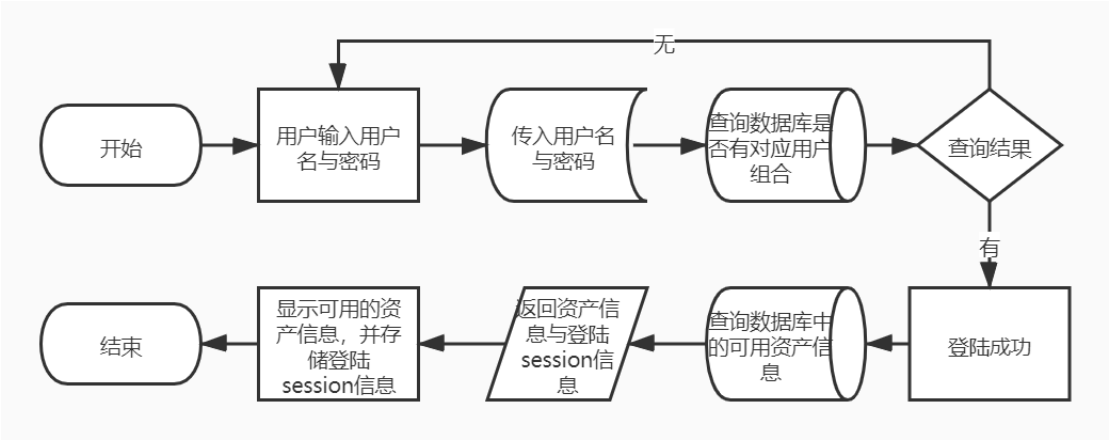
3.3.1 普通用户资产领用

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。

- **子功能 1：登录&浏览可用资产**

(1) 设计图

本模块的处理过程如图所示：



(2) 功能描述

用户进入软件，输入用户名和密码登录系统。登陆成功后，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。登录失败则返回错误信息，并提示重新登录。

(3) 输入数据

登录&浏览可用资产模块需要用户的用户名和密码两个数据。

系统根据用户名与密码的组合查询数据库，当用户名与密码的组合存在时，用户登录成功，否则登录失败。

(4) 输出数据

登录&浏览可用资产模块在用户登录成功后将登录信息存储于 session 中，并查询数据库输出所有可以领用的资产及详细信息。

(5) 业务算法和流程

用户进入软件，输入用户名和密码登录系统。系统根据用户名和密码查询数据库，并返回登录结果，若登陆成功则将登录信息存储于 session 中。登陆成功后，系统查询数据库中的资产数据并返回信息，在用户登录后的默认页面显示所有可领用的资产及详细信息。

(6) 数据设计

用户名：由用户给出的一串字符串，加密后传入。

密码：由用户给出的一串字符串，加密后传入。

用户类：以用户的用户名，密码等信息封装的类，用于查询数据库。

资产类：查询资产信息时，从数据库读出信息以一个资产类的实体的形式封装，并在返回时将内部信息写入页面。

登录 session 信息：存储于 session 中的登录信息，分别为用户名与密码

(7) 源程序文件说明

AuthController.java：登录的统一控制类

UserController.java：普通用户相关功能的控制类

HomeController.java：主页控制类（主页与 404）

User.java：用户类，用作信息封装

Property.java：资产类，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

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。

AuthUtil.java：用于处理用户输入的用户名与密码

index.jsp：主页 jsp 页面

userViewProperties.jsp：用户查看资产 jsp 页面

(8) 函数说明

1) AuthController.java：

public String loginErrorPage()：返回登录错误页面。

public String userLogout(HttpSession session)：用户登出，清除登录的 session 信息。

public JsonResponse userLogin(@RequestParam(value="info") String info,

HttpServletRequest request, HttpServletResponse response)：用户登录，根据传入加密的用户名和密码，解密并验证登录信息，成功登录则封装用户类，设置 session 并返回加密登录信息的 cookies，失败则返回登录错误页面。

2) UserController.java：

public String viewProperties(@PathVariable Integer pageNo, @PathVariable 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

3) HomeController.java：

public String home()：返回主页

public String notFound()：返回 404

4) User.java/ Property.java：

get/setxxx：变量获取/设置函数，不重复赘述

5) UserPropertyService.java：

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

6) UserPropertyServiceImpl.java：

UserPropertyService.java 的具体实现，不在赘述

7) UPListTag.java：

public void doTag()：SimpleTag 具体行为实现，拆封 Properties 的信息并写入 jsp

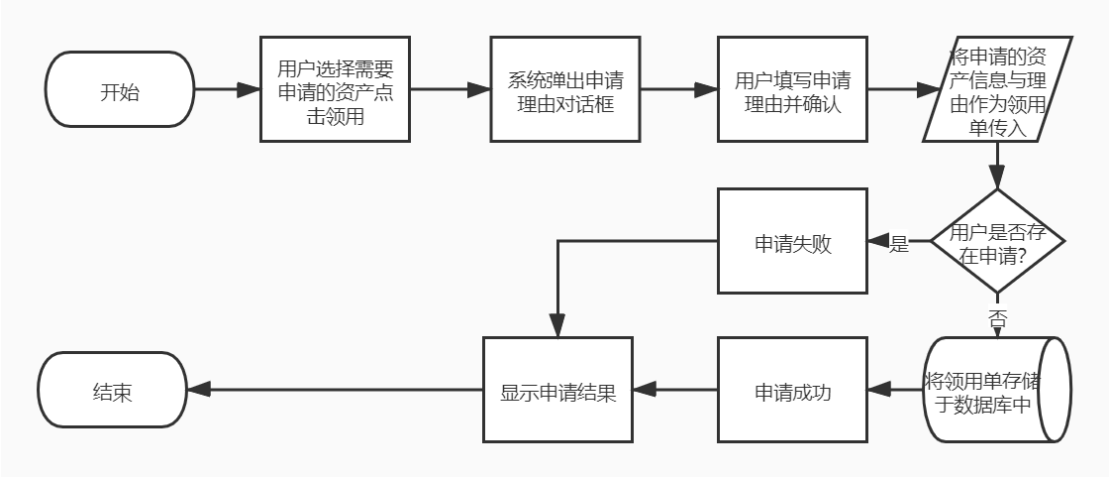
8) AuthUtil.java：

public static String[] getUserInfo(String userData)：解析传入的用户名与密码字符串并重写为方便的字符串数组的形式。

- **子功能 2：资产领用申请功能**

(1) 设计图

本模块的处理过程如图所示：



(2) 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。

(3) 输入数据

资产领用申请模块需要所申请的资产和申请理由两个输入数据。

(4) 输出数据

资产领用申请模块返回（输出）申请结果（是否成功）。

(5) 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与理由生成领用单并保存至数据库，并返回申请的结果（是否成功）。

(6) 数据设计

用户类：用户信息的封装类。

资产类：资产信息的封装类。

申请理由：用户输入的申请理由，以二进制编码的形式被存储传入。

领用单类：将申请的资产与申请理由封装在一起，传入数据库保存。

(7) 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

userViewProperties.jsp：用户查看与申请资产的 jsp 页面

(8) 函数说明

不再赘述出现过的函数

1) UserController.java：

public JsonResponse applyProperty(@PathVariable 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。

2) UserApplicationService.java：

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

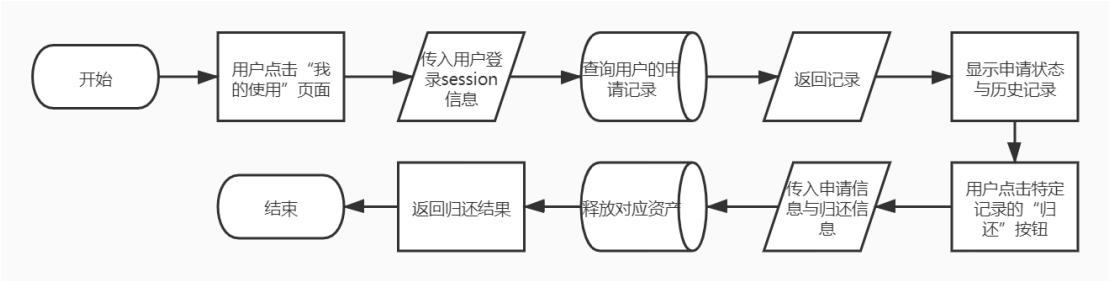
3) UserApplicationServiceImpl.java：

UserApplicationService.java 的具体实现，不再赘述

- **子功能 3：申请状态、历史记录与归还功能**

(1) 设计图

本模块的处理过程如图所示：



(2) 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。

(3) 输入数据

资产领用申请模块需要用户的登录信息（保存于当前登录用户的 session 中）。

(4) 输出数据

资产领用申请模块返回（输出）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，以及用户归还资产的结果信息。

(5) 业务算法和流程

用户点击进入“我的使用”页面，系统根据存储在 session 中的用户登录信息向数据库查询用户的记录，之后向用户返回查询结果，在用户界面上显示领用单的审批状态与历史记录。用户点击特定记录的“归还”按钮尝试归还资产，系统传入对应资产信息，并在数据库

中将对应资产设置为可领用，最后返回归还结果。

(6) 数据设计

用户类：用户信息的封装类。

资产类：资产信息的封装类。

领用单类：将申请的资产与申请理由封装在一起，传入数据库保存。

(7) 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

(8) 函数说明

1) UserController.java：

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：

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：

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

4 数据流图

本系统顶层数据流图如图所示，系统的信息来源实体主要是系统的普通用户和管理员，管理员向系统提供了有关于用户、资产、管理员的诸多信息，用户向系统提供了领用申请的信息，同时管理员从系统中获取领用相关信息，用户则可以获取资产、领用记录等信息。

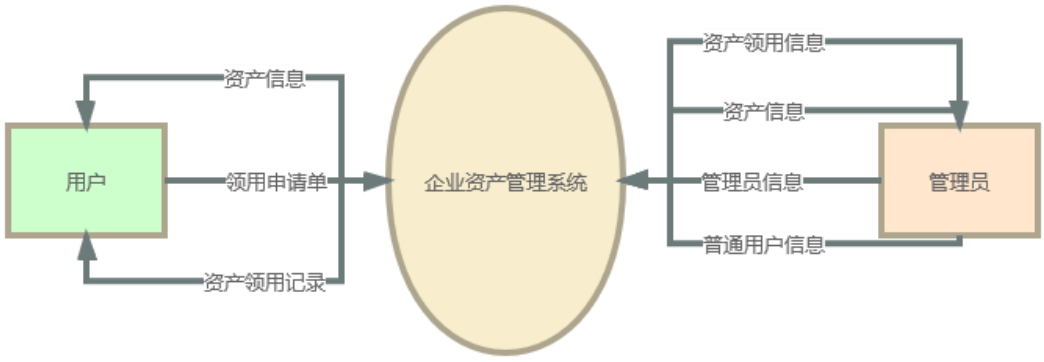


图 4.1 顶层数据流图

5 数据库设计

5.1 实体类设计

5.1.1 实体及其属性

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: appId(PK), uId(FK), pId(FK), mId(FK), appTime, viewTime, viewState, isUsing, termTime

5.1.2 实体类图

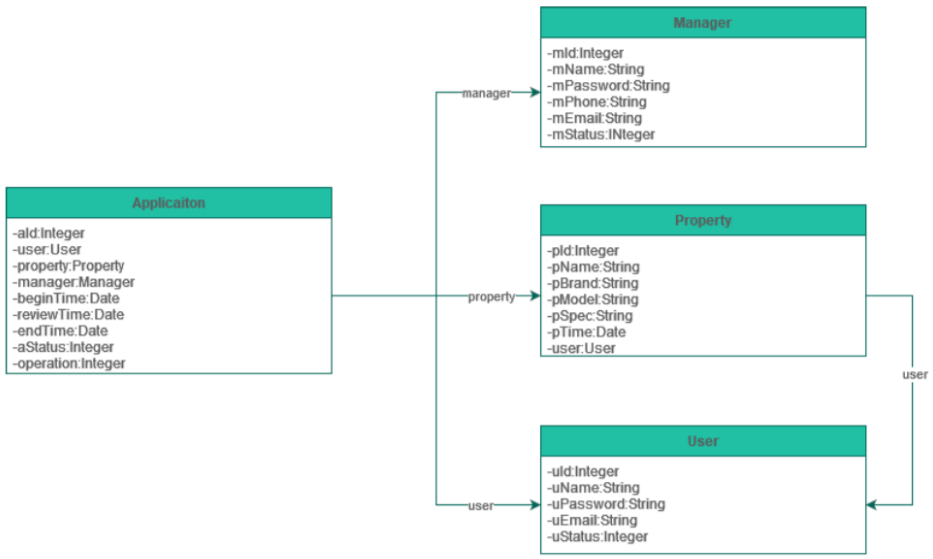


图 5.1 实体类图

5.2 实体间关系

5.2.1 实体关系

View(Manager, Application):

管理员和领用单之间存在一对多的审查关系，一个管理员可以审核多个领用 单，一个领用单由一名管理员审核。

Apply(User, Application):

用户和领用单之间存在一对多的申请关系，一个用户可以发起多项领用申请， 一个领用单对应某个用户的一个申请。

Include(Property, Application):

企业资产与领用单存在一对多的参与关系，一个资产可能被多次领用产生多 条 领 用记录，一条领用记录对应一个资产。

5.2.2 实体关系图

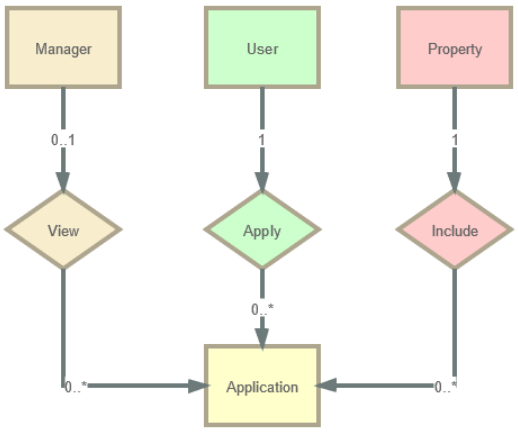


图 5.2 实体关系图

5.3 数据表属性及其数据字典：

表名：管理员 Manager:

属性名	mId	mName	mPassword	mPhone	mEmail	mState
属性含义	管理员 ID	管理员用户名	管理员密码	管理员手机号	管理员邮箱	管理员状态
属性约束	PK	Unique&Notnull	Notnull	Notnull		Notnull(default=true)
数据类型	Long	String	String	String	String	boolean
其他		4~16	4~16	11 位手机号码	合法邮箱	true，false

表 5.1

用户 User:

属性名	uId	uName	uPassword	uEmail	uState
属性含义	用户 ID	用户名	用户密码	用户邮箱	用户状态
属性约束	PK	Unique&Notnul	Notnull		Notnull(default=0)
数据类型	Long	String	String	String	Int
其他		4~16	4~16	合法邮箱	0，1，2

表 5.2

资产 Property:

属性名	pId	pName	pBrand	pModel	pSpec	pTime
属性含义	资产 ID	设备名称	设备品牌	设备型号	设备规格	设备采购日期
属性约束	PK	Notnull	Notnull	Notnull	Notnull	Notnull
数据类型	Long	String	String	String	String	Date
其他		4~16	4~16	4~16	4~16	

表 5.3

领用单 Application:

属性名	appId	uId	pId	mId	appTime	viewTime
属性含义	领用单号	领用人编号	领用设备编号	审核人编号	申请时间	审核时间
属性约束	PK	FK	FK	FK	Notnull	
数据类型	Long	Long	Long	Long	Date	Date
属性名	viewState	isUsing	termTime			
属性含义	审核状态	使用状态	归还时间			
属性约束	Notnull(default=0)	Notnull(delfault=true)				
数据类型	int	boolean	Date			
其他	0，1，2	true，false				

表 5.4

6 转换关系

6.1 状态属性解读

(1) 管理员状态 mState：该状态用于表征该管理员身份是否有效，涉及功能点为管理员登录、创建、删除，涉及用例为 UC1.1、UC3、UC4。该字段为 boolean 类型，true 表示管理身份有效，false 表示管理身份无效。

(2) 用户状态 uState：该状态用于表征用户身份所处状态，涉及用例 UC1，UC1~17。该字段为 int 类型，0 表示该用户身份有效，1 表示该用户被锁定，2 表示该用户被删除，三者可能的状态转化为:

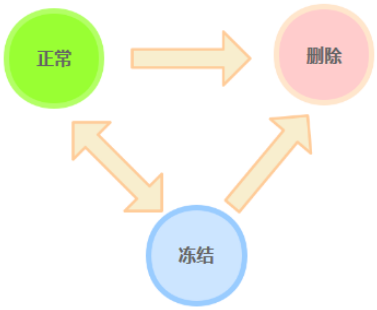


图 6.1 用户状态转换图

6.2 系统页面跳转图

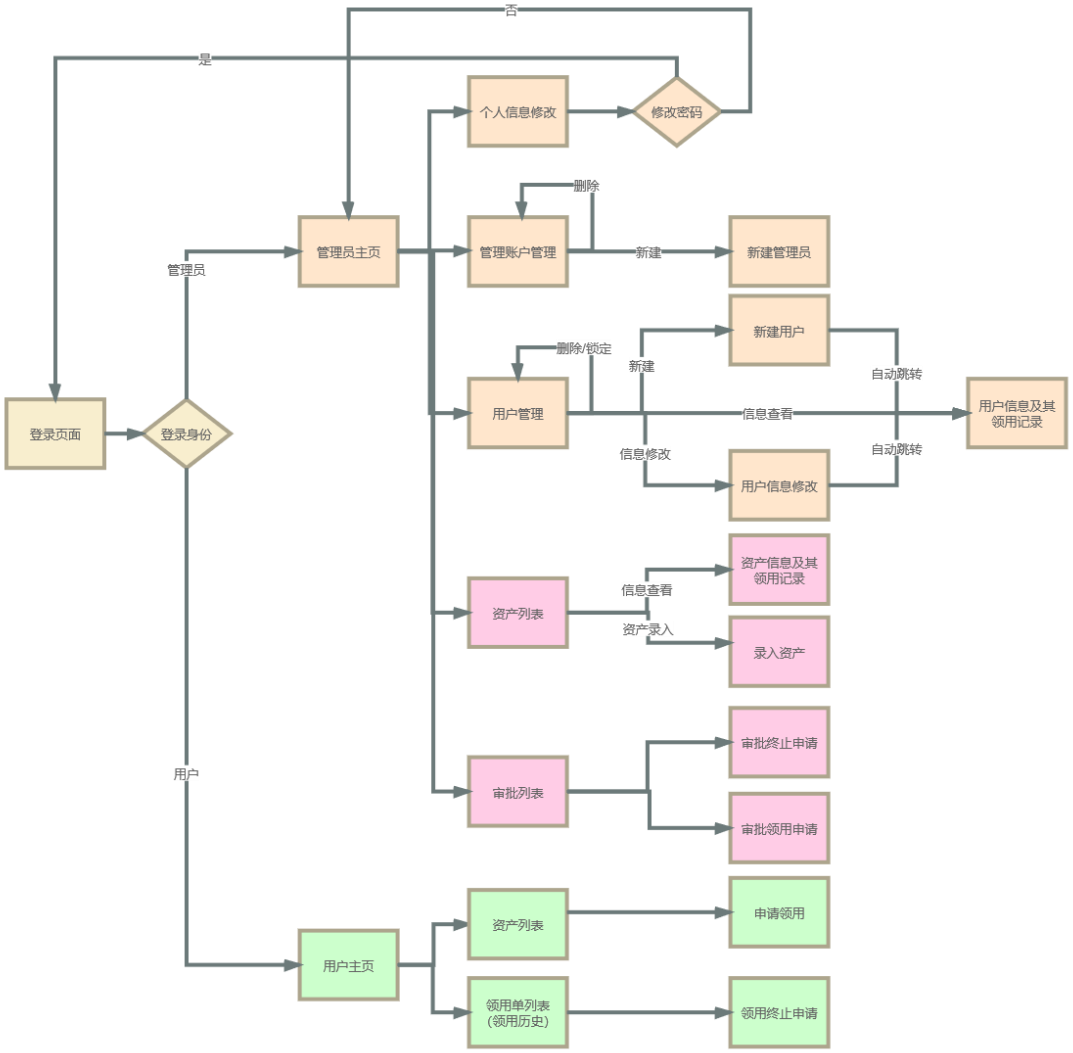


图 6.2 系统页面跳转图

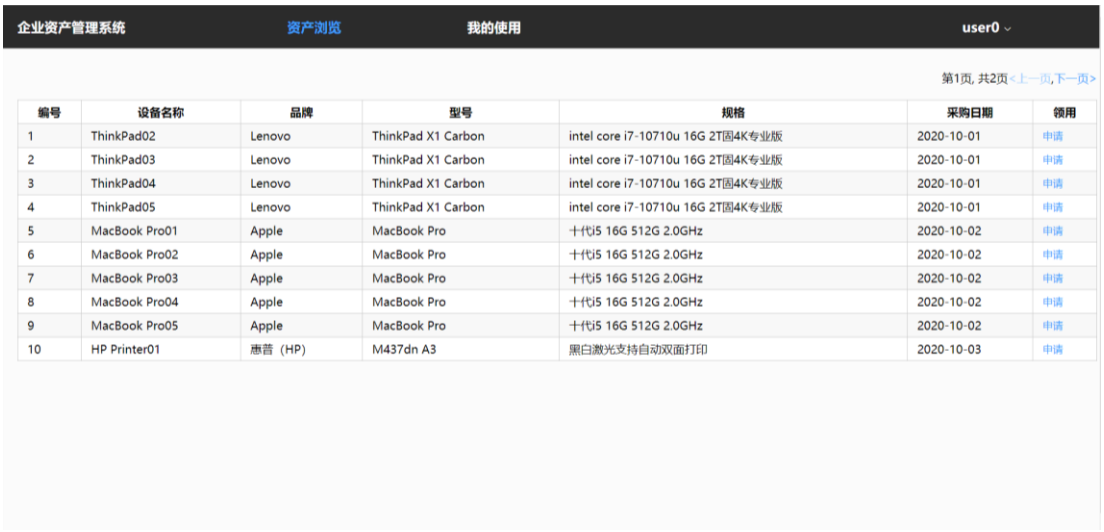
7 界面设计

7.1 登录界面

图 7.1 登录界面

以黑色为背景，使得白字显目而庄重，整个登录系统大气简约，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。

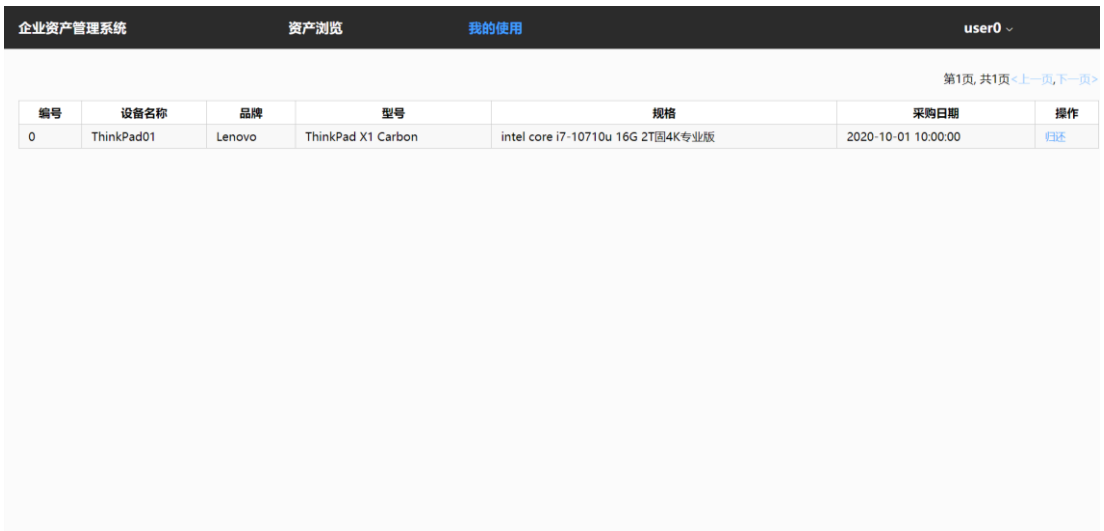
7.2 普通用户界面



编号	设备名称	品牌	型号	规格	采购日期	领用
1	ThinkPad02	Lenovo	ThinkPad X1 Carbon	intel core i7-10710u 16G 2T固4K专业版	2020-10-01	申请
2	ThinkPad03	Lenovo	ThinkPad X1 Carbon	intel core i7-10710u 16G 2T固4K专业版	2020-10-01	申请
3	ThinkPad04	Lenovo	ThinkPad X1 Carbon	intel core i7-10710u 16G 2T固4K专业版	2020-10-01	申请
4	ThinkPad05	Lenovo	ThinkPad X1 Carbon	intel core i7-10710u 16G 2T固4K专业版	2020-10-01	申请
5	MacBook Pro01	Apple	MacBook Pro	十代i5 16G 512G 2.0GHz	2020-10-02	申请
6	MacBook Pro02	Apple	MacBook Pro	十代i5 16G 512G 2.0GHz	2020-10-02	申请
7	MacBook Pro03	Apple	MacBook Pro	十代i5 16G 512G 2.0GHz	2020-10-02	申请
8	MacBook Pro04	Apple	MacBook Pro	十代i5 16G 512G 2.0GHz	2020-10-02	申请
9	MacBook Pro05	Apple	MacBook Pro	十代i5 16G 512G 2.0GHz	2020-10-02	申请
10	HP Printer01	惠普（HP）	M437dn A3	黑白激光支持自动双面打印	2020-10-03	申请

图 7.2 普通用户界面 1

普通用户登录后进入资产浏览界面，依旧是采用黑白结合的简朴干练的 UI 设计，简洁明了地把编号、设备名称、品牌、型号、规格、采购日期、领用展示给了普通员工用户，方便其进行申请领用，使得软件系统上手简单，易于操作。



编号	设备名称	品牌	型号	规格	采购日期	操作
0	ThinkPad01	Lenovo	ThinkPad X1 Carbon	intel core i7-10710u 16G 2T固4K专业版	2020-10-01 10:00:00	归还

图 7.3 普通用户界面 2

点击上方“我的使用”按钮，即可查看已经借用的企业资产。如果借用完毕，员工可以很方便地归还企业资产。右上角“user0”点击后可以选择退出登录，方便清晰。

7.3 管理员界面

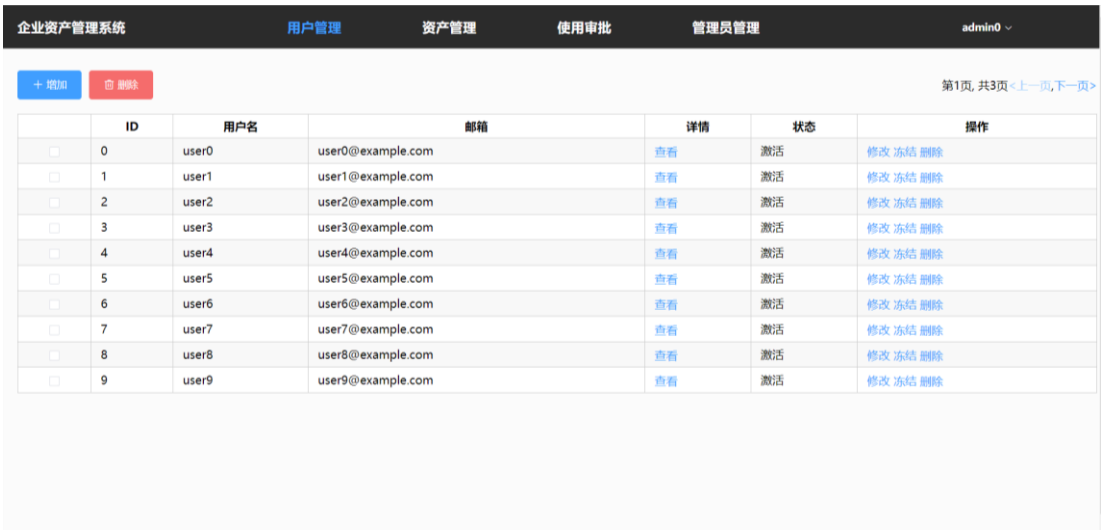


图 7.4 管理员界面 1

管理员界面与普通用户界面类似，不同之处在于在左上角添加了颜色鲜明的“添加”，“删除”按钮，蓝色的“添加”标识醒目舒适，红色的“删除”标识对管理员进行操作时起到了提醒的作用，提醒管理员删除用户的操作要慎重。

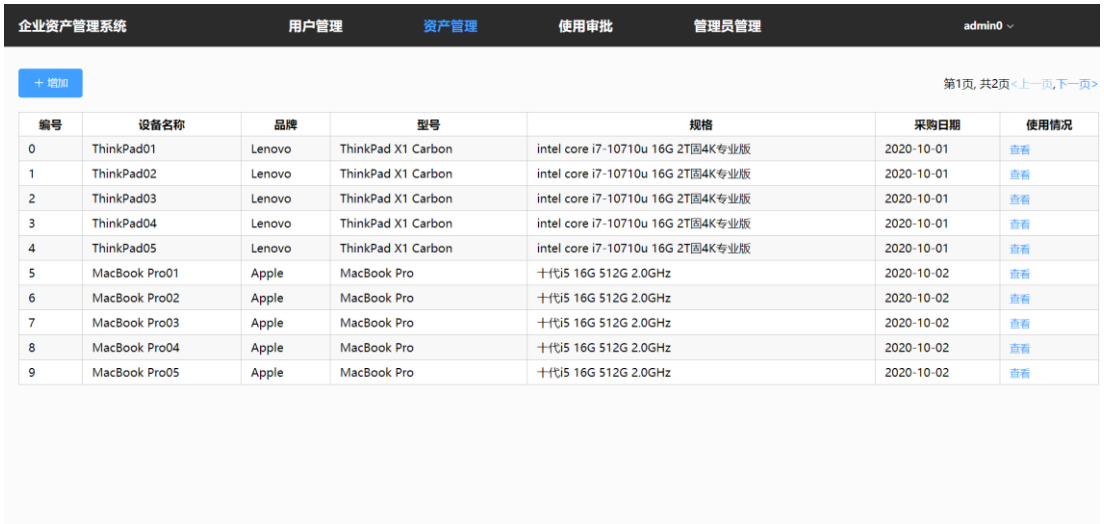


图 7.5 管理员界面 2

管理员的资产管理界面与普通用户的资产浏览界面类似，不同之处在于左上角的“添加”按钮，以及每一件资产后面的“使用情况”栏不同。用户对于资产是借用与归还，而管理员界面则是对资产的使用情况进行管理，清楚无歧义，方便管理员管理。

图 7.6 管理员界面 3

管理员界面中的使用审批界面，对于普通用户的申请领用进行批准或者驳回，对管理者的使用很友好，清楚简单。“上一页”，“下一页”可以查找更多的资产使用情况。

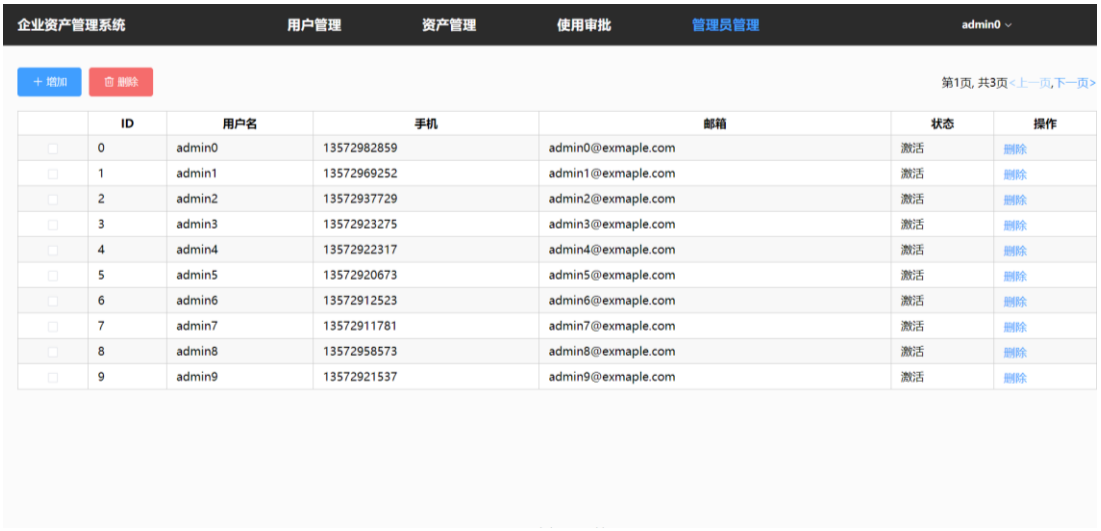


图 7.7 管理员界面 4

如图所示是管理员中的超级管理员的界面，类似于管理员对于普通用户账号的增删权限，超级管理员可以对管理员账号进行管理，激活后的账号可以对其删除。

图 7.8 管理员界面 5

管理员界面的右上角“admin0”按钮与普通用户的不同，它多出来一项修改信息的功能，在功能中每一行的表述都很清晰明了，管理员可以对于自身的密码进行修改，二次确认后即修改成功，也可以对手机号和邮箱地址进行改动，系统的理解和使用无障碍。

7.4 退出登录界面

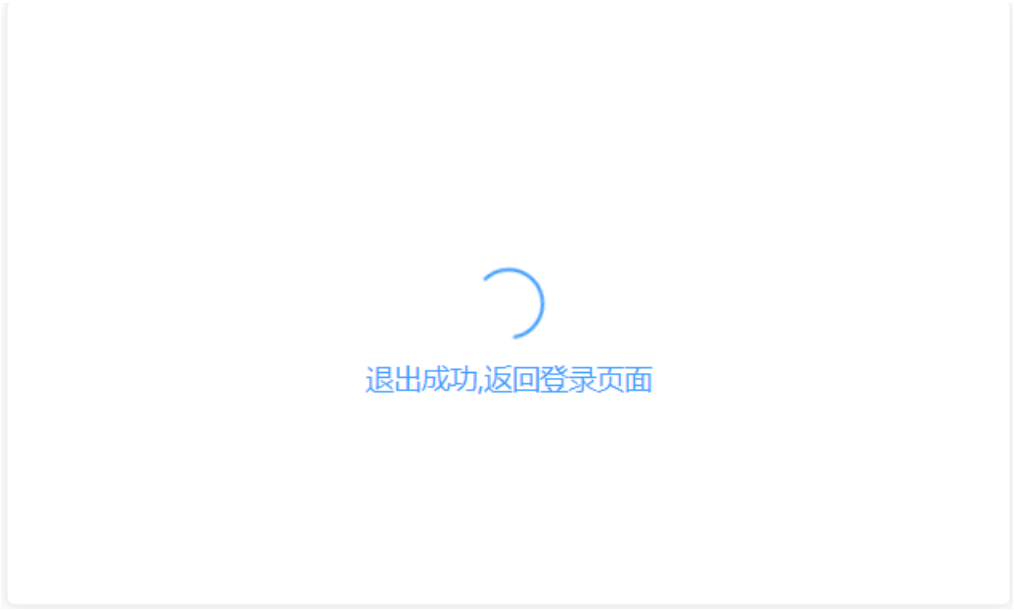


图 7.9 退出登录界面

依旧是一贯的简洁风格，蓝色文字看起来轻松愉快。

8 接口设计

参见《EAM 接口设计文档 V2.0》