

# 课程设计总报告和个人小结

## 1. 引言

### 1.1 项目背景和现状

在日常的校园生活中，学生们对各种日用品及学习物品的需求量很大，由于学生的经济能力有限，二手交易是一种很好的交易方式。在如今的校园中，二手交易已经成为一种十分普遍的现象。学生们的二手交易通常是通过 QQ、微信等社交软件来传播物品信息从而来寻找买家或者卖家，但这种方式不利于信息的整合，而且效率底下、扩散面不广，另外，这种传播方式的时效较短，过了几天这些物品信息就会被其他信息所覆盖，更加难以找到合适的买卖者。

### 1.2 开发目的和项目简介

为了让学生们的二手交易过程完善化、系统化、便捷化，我们开发了二手交易平台。在此平台上，用户可以发布自己想要出售的二手商品信息，对于想要购买二手商品的人来说，只需要登录平台，在相应的物品信息页面浏览，找到自己想要的物品后，通过发送邮件的方式联系物品持有者，双方沟通完毕后即可进行物品发送，这就是此平台的整个工作流程。此外，为了确保用户发布的物品信息不包含反动、暴力等有危害言论，此平台有专门的管理员端对用户发布的物品信息、邮件信息进行审核并可以接收用户的投诉。

## 2. 系统需求

### 2.1 功能需求

#### 2.1.1 系统角色和及其功能

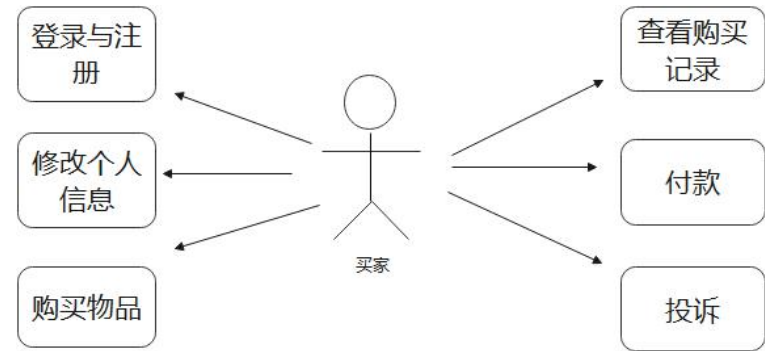
角色视图：



角色说明：

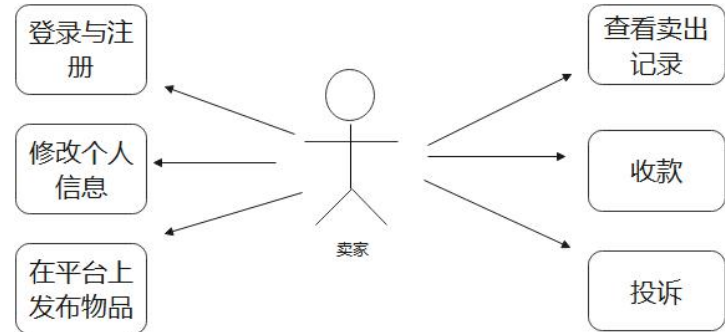
角色名称	说明
买家	广大用户，具有注册和登录、修改个人信息、在平台上购买物品、查看购买记录、付款、投诉等功能。群众
卖家	广大用户，具有注册和登录、修改个人信息、在平台上发布物品、查看卖出记录、收款、投诉等功能。群众
管理员	平台工作人员。具有注册登录、修改个人信息、查看用户提交的物品信息、查看用户所发邮件、处理客户投诉、审核用户交易信息、删除不当交易信息、封禁客户账号等功能。工作人员

(1) 物品持有者参与业务：



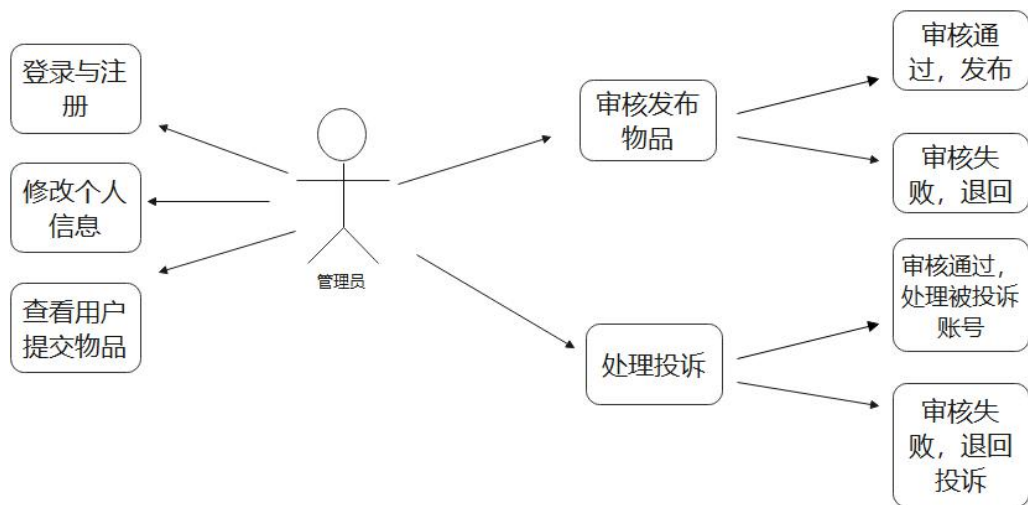
说明：买家登录，注册买家账号，后，可以修改个人信息，可以在平台上购买发布物品，付款，查看自己的购买记录，可以进行投诉，在管理员审核通过后对被投诉账号进行处罚。

(2) 卖家参与业务



说明： 卖家可以在平台上注册，登录卖家账号，修改个人信息，在平台上发布物品，查看自己的卖出记录，收款，投诉等，在经过管理员审核通过后对被投诉账号进行处罚

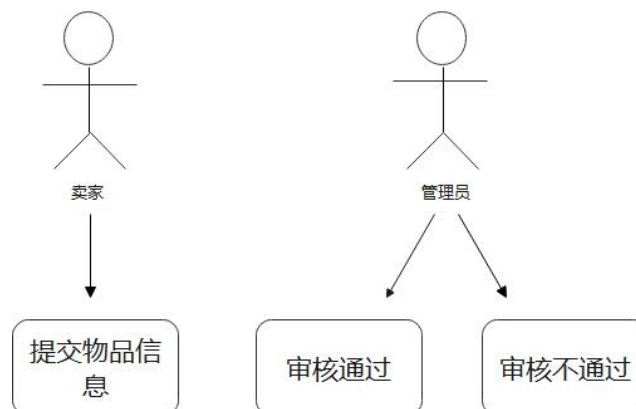
### (3) 管理员参与业务



说明：管理员登录，注册管理员账号，可以修改个人信息，查看卖家提交全部物品，审核物品，审核通过后发布到平台，审核不通过退回，可以处理投诉信息，审核通过后处理被投诉账号，审核不通过则退回投诉。

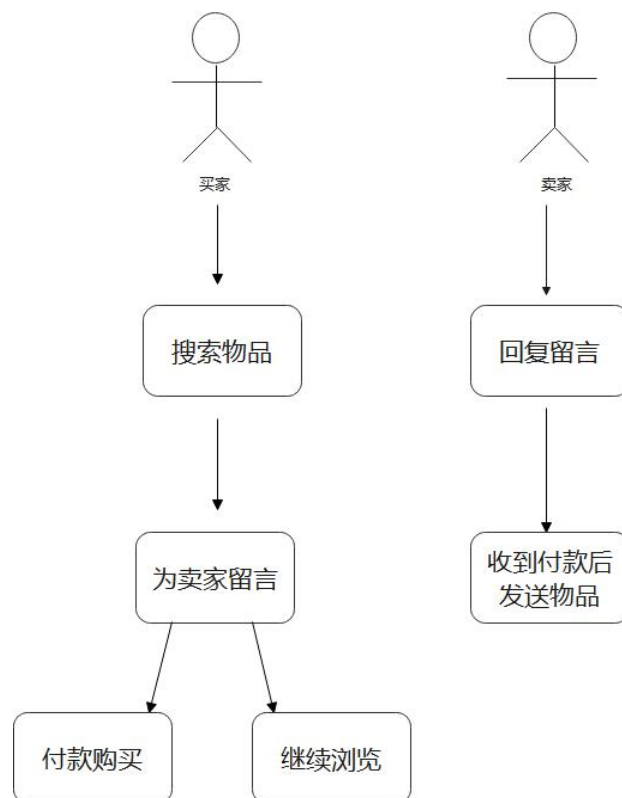
### 2.1.2 需求规定

#### (1) 发布物品信息业务



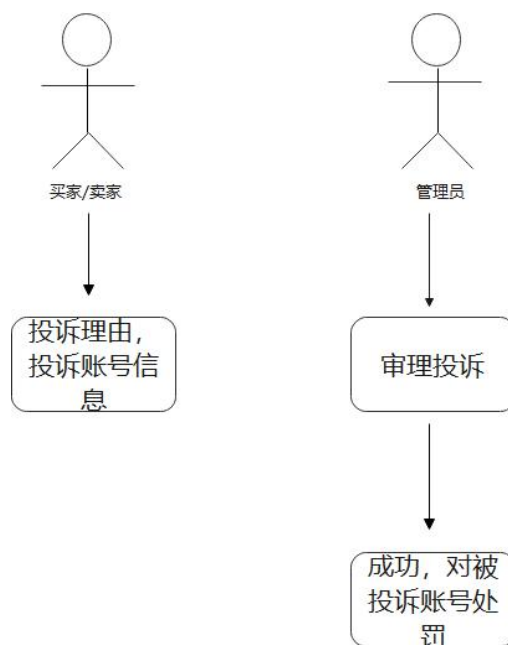
发布物品信息业务说明：物品的发布需要通过卖家在网上输入物品信息，然后提交给管理员，管理员审核通过之后确认发布物品信息，如果审核不通过则不能发布。

#### (2) 购买物品业务



购买物品物业务说明： 买家搜索自己想要物品后，给卖家留言，卖家收到留言后回复，买家同意购买，付款购买，卖家收到付款，发送物品，买家不同意购买，继续浏览。

### (3) 投诉业务



投诉业务说明： 卖家或买家提交投诉理由，管理员接受投诉信息审核，审核

通过对被投诉账号做出处罚。

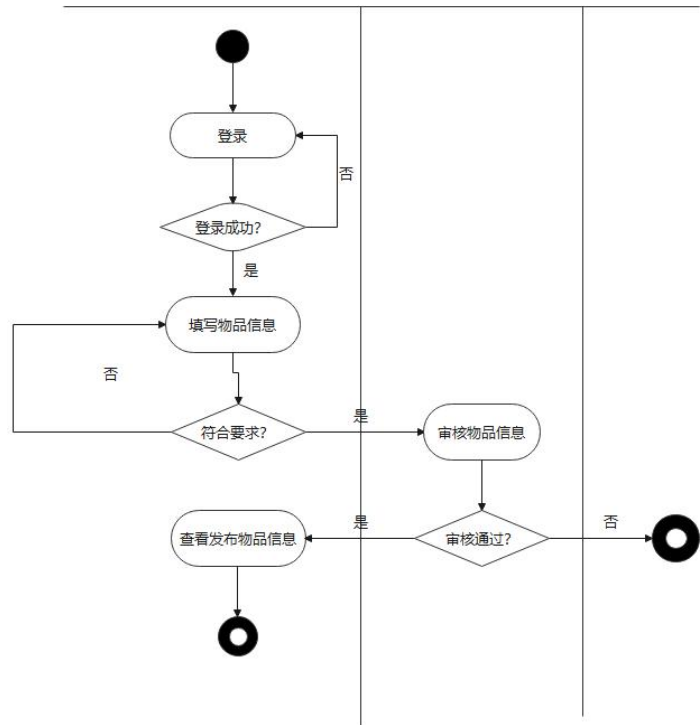
#### 2.1.2.1 发布物品功能点

业务说明：

用例名称	发布物品
实现名称	post item
用例描述	卖家通过此用例向管理员请求发布自己所持有的物品
参与者	卖家 管理员
前置条件	用户登录界面，通过验证。
后置条件	1. 提交物品信息等待审核
主事件流	<ol style="list-style-type: none"><li>1.卖家根据以前注册的账号登录系统,计算机显示系统主页</li><li>2.卖家选择上传物品，计算机显示上传物品界面</li><li>3.卖家填写持有物品的具体信息，并确认提交。</li><li>4.管理员根据管理员账号登录系统，计算机显示审核页面。</li><li>5.管理员对卖家提交的物品信息进行审核,计算机更新审核状态。</li><li>6.卖家查看物品信息，计算机显示发布是否成功。</li><li>7.用例结束</li></ol>
备选事件流	<ol style="list-style-type: none"><li>1. a 登录信息不正确 拒绝登录，计算机执行主流事件 1；</li><li>2. a 持有物品信息不符合要求，提交失败 计算机执行主流事件 3；</li><li>3. a 管理员登录信息不正确 拒绝登录，计算机执行主流事件 4</li><li>4. a 管理员通过审核 计算机执行主流事件 6，显示发布成功</li><li>4. b 管理员不通过审核 计算机执行主流事件 6，显示发布失败</li></ol>
业务规则	根据持有物品情况提交物品信息

涉及的业务实体	物品，物品记录，用户，管理员
非功能性需求	支持各地区的用户

业务场景分析：



业务场景分析说明：卖家首先登录平台，登录失败则重新登录，成功则填写自己期望发布物品的信息，不符合要求则重新填写，符合要求发送给管理员审核，审核成功发布该物品，失败则不发布该物品。

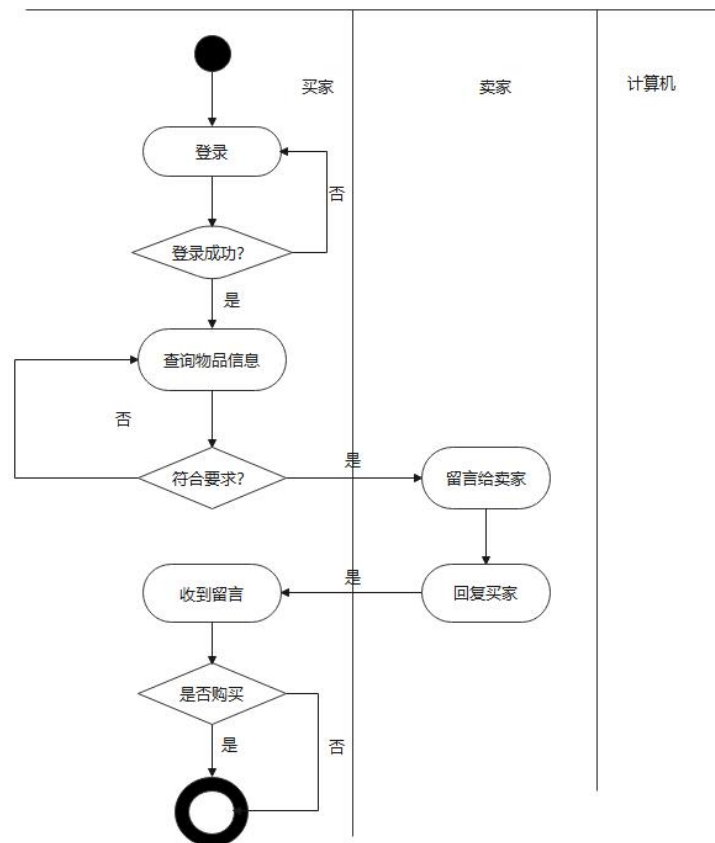
### 2. 1. 2. 2 购买物品功能点

业务说明：

用例名称	购买物品
实现名称	buy item
用例描述	买家通过此用例向卖家购买物品
参与者	卖家，买家
前置条件	用户登录界面，通过验证
后置条件	买家购买发布物品
主事件流	1.买家根据以前注册的账号登录系统，计算机显示系统主页。 2.买家查询物品，计算机显示查询结果。

	<p>3.买家进入留言系统为卖家留言。</p> <p>4.卖家收到留言回复买家。</p> <p>5.买家选则购买或者不购买。</p> <p>6.用例结束。</p>
<b>备选事件流</b>	<p>1. a 买家信息不正确 拒绝登录，计算机执行主流事件 1；</p> <p>2. a 买家查询到希望购买物品 计算机执行主流事件 2；</p> <p>2. b 买家未找到自己希望购买物品 用例结束；</p> <p>3. a 买家留言给卖家 执行主流事件 3；</p> <p>4. a 卖家收到留言回复买家 执行主流事件 4；</p> <p>5. a 买家购买物品 执行主流事件 5；</p> <p>5. b 买家不购买物品 用例结束</p>
<b>业务规则</b>	买家根据自己期望购买到的物品登录平台搜索寻找
<b>涉及的业务实体</b>	物品，卖家，买家
<b>非功能性需求</b>	支持各地区的用户

业务场景分析：



业务场景说明： 买家登录平台，登录失败则重新登录，登录成功查询要买物品，符合要求，通过邮件发收功能留言给卖家，卖家收到留言回复买家，买家收到回复决定是否购买，结束。

### 2.1.2.3 投诉功能点

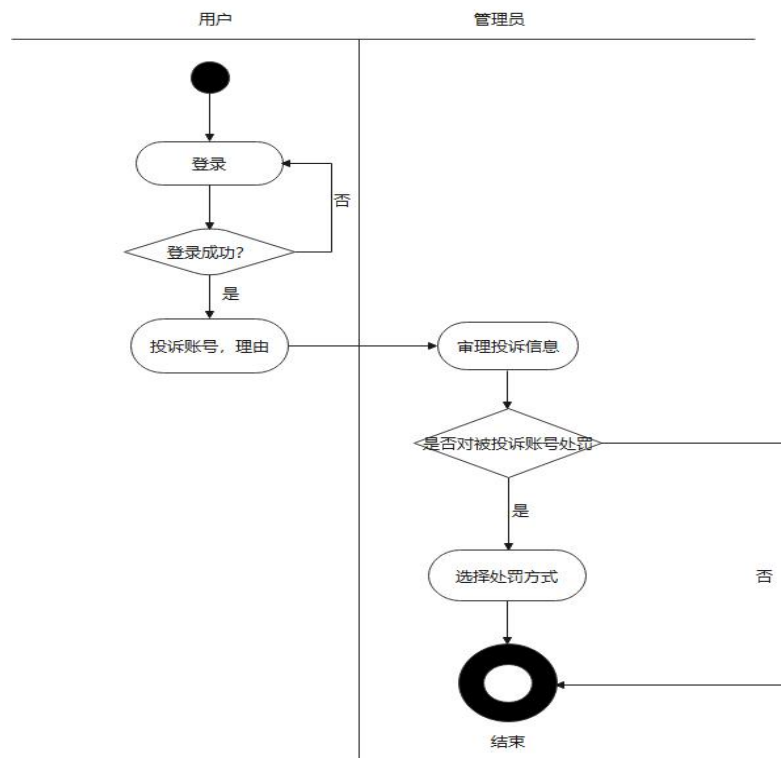
业务说明：

用例名称	投诉功能点
实现名称	Complaints item
用例描述	用户向管理员投诉
参与者	用户，管理员
前置条件	用户登录界面，通过验证
后置条件	用户投诉
主事件流	<ol style="list-style-type: none"> <li>1.用户根据以前注册的账号登录系统，计算机显示系统主页。</li> <li>2.用户向管理员通过邮件发收功能，提交投诉。</li> <li>3.管理员接到投诉并审核。</li> </ol>



	<p>4.管理员成功审理投诉，对被投诉账号进行相关处罚</p> <p>5.用例结束。</p>
<b>备选事件流</b>	<p>1. a 用户信息不正确 拒绝登录，计算机执行主流事件 1；</p> <p>2. a 用户提交投诉账号，理由 计算机执行主流事件 2；</p> <p>3. a 管理员审理投诉 计算机执行主流事件 3；</p> <p>4. a 管理员审理投诉成功 执行主流事件 4；</p> <p>4. b 管理员审理投诉失败 执行主流事件 5；</p>
<b>业务规则</b>	用户向管理员投诉
<b>涉及的业务实体</b>	用户，管理员
<b>非功能性需求</b>	支持各地区的用户

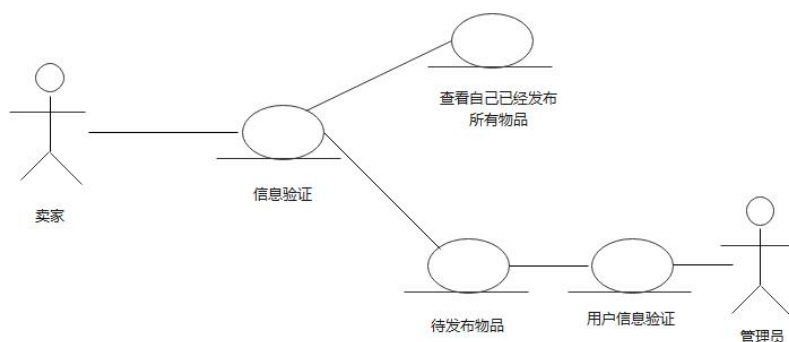
业务场景分析：



业务场景分析说明：用户登录平台，失败则重新登录，成功登录后，通过邮件发送功能递送违规账号描述相关账号的违规信息，管理员通过邮件接收功能查看邮件，管理员审理该投诉信息，审理成功对被投诉账号做出删除交易信息或者封禁账号处罚。

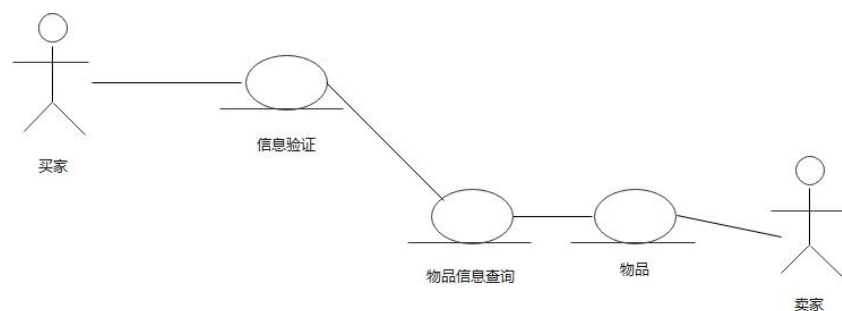
### 2.1.3 业务实体分析

#### 1. 发布物品的业务实体分析



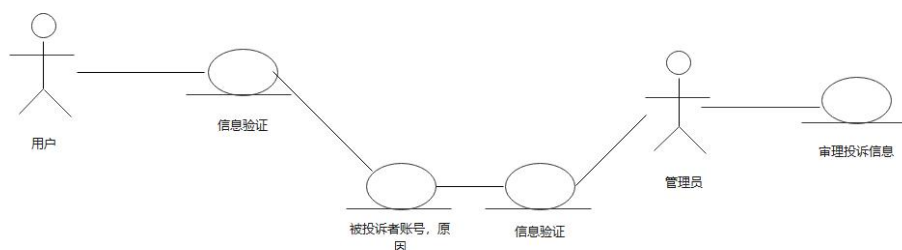
业务实体说明：上图显示的是发布物品的实体过程，卖家通过登录个人的信息进行验证，进入信息发布页面后发布持有物品的信息，管理员登录并审核，对其每次成功发布的物品信息都有一个记录

#### 2. 购买商品业务实体分析



业务实体说明：上图显示的是一个购买商品的实体分析，买家登录平台搜索自己希望得到的物品信息，向卖家购买的过程。

### 3. 投诉的业务实体分析



业务实体说明：上图显示的是用户投诉的业务实体分析，用户登录界面进行身份验证，对他人账号进行投诉，并写明投诉原因，提交给管理员，管理员经过信息验证登录后审理投诉，并进行判断。

## 2.2 非功能性需求

### (1) 性能需求：

系统主要的功能是二手商品交易平台，面向的用户只有南理工内的老师、学生和后勤人员，系统设计又受到组内技术经验的影响，因此该系统构造较小。但其需要实现的操作有，调用多个不同存储内容的数据库，处理多个图片，审核的过程需要人机交互，这些会导致页面响应时间比较长，经过一系列优化后才能投入实际应用。

### (2) 系统可靠性：

受硬件设备和组内人员技术经验的影响，该系统可能会存在一定 bug，对系统的可靠性可能会造成一定的影响。但是我们可以尽量使该平台所实现的基本功能可以正常、可靠的使用。

### (3) 系统可扩展性：

可实现负载均衡，日后若信息量较大，则系统可相应增加服务器实现扩展。

### (4) 系统安全性：

系统使用华为 openGauss 数据库来存储数据，可以保证数据的安全可靠、不丢失。除了管理员以外其他用户只有权限修改自身账号所关联的信息。

3. 系统设计

3.1 基于 MVC 的软件架构设计

3.1.1 架构概览

二手物品交易平台的系统设计工作是整个系统阶段实现的重要内容，其中系统架构的设计尤为重要，涉及到系统的总体架构，围绕用户操作、业务逻辑处理和数据信息存储等重要内容进行设计。本平台的系统软件架构设计主要采用基于 MVC 的五层架构的方式进行实现，详细软件架构设计如图 2.1 所示。



图 2.1 系统软件架构设计图

从图 2.1 可以看出五层体系软件架构设计方面的内容，按照业务类型、功能类别、关联关系等方式，对系统每个逻辑层涉及的相关功能和服务进行模块设计，每一层操作的具体内容如下：

- (1) 界面控制层  
界面层包括界面层和控制层。界面层是指系统移动端用户最终的使用界面，主要负责系统数据的展现，同时接受用户的输入数据并对其进行校验。控制层主要负责封装界面层输入的数据，控制页面的跳转，以及对异常进行处理。界面控制层与业务逻辑层存在依赖关系，层次间的通信主要采用 HTTP 协议进行传输。
- (2) 业务逻辑层

业务逻辑层主要负责为系统提供业务逻辑的接口，实现系统的业务逻辑，对事务进行控制，以及对外提供或调用 Web 服务。该逻辑层主要与技术服务层和数据层存在依赖关系，层次间的通信主要采用 Web 服务和 API 调用的方式。

(3) 技术服务层

技术服务层包含由系统开发平台和第三方产品提供的各类基础模块。该逻辑层主要与数据层存在依赖关系，层次间的通信主要采用 API 调用的方式。

(4) 数据层

数据层主要提供存储数据实体，向业务逻辑层提供访问数据库、文件系统等资源的接口。该逻辑层主要与基础架构服务层存在依赖关系，层次间的通信主要采用 Web 服务和 API 调用的方式。

(5) 基础架构服务层

基础架构服务层主要包含应用服务器、数据库等。

二手商品交易平台主要采用互联网端服务器应用，实现基于 MVC 模式的二手商品发布交易等功能，系统主要包括客户端和管理员端。其中客户端主要实现账号登录、二手商品发布、二手商品购买等功能，后续根基平台的使用情况，对其进行扩展；管理员端主要实现账号登录、商品信息管理、客户信息管理等功能。

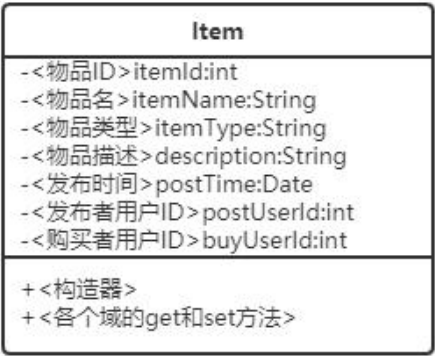
3.1.2 实体类的设计

(1) 用户实体



说明：管理员与客户共用一个实体类，每一个用户个体由用户 ID 唯一标识，而 idAdmin 来区分该用户是管理员还是客户。

(2) 商品实体



说明：用于存储商品的基本信息，每一个商品由物品 ID 唯一标识。其中，物品类型为固定几类的选项(用下拉框实现)，物品描述为用户自由编写。

(3) 邮件实体



说明：该实体用于存储用户与用户之间的邮件信息的存储，每一封邮件由邮件消息 ID 唯一标识。

#### (4) 投诉实体



说明：该实体用于存储用户的投诉信息，管理员可以查看到所有的投诉信息，并给予回复。

#### (5) 页面分页实体



说明：该实体类用于页面分页信息的存储。

### 3.1.3 数据访问 dao 层的设计

#### (1) 用户 dao 层设计

UserDao
<pre> +queryUserById(userId:int):User +queryUserByUsername(userName:String):User +queryUserIsLocked(userId:int):int +queryUserIsAdmin(userId:int):int +addUser(newUser:User):int +updateUserName(userId:int):int +updatePassword(userId:int):int </pre>

说明：以上方法从上至下分别为：1) 以用户 ID 查找用户信息；2) 以用户名查找用户信息；3) 判断用户账户是否被封禁；4) 判断用户账户是否为管理员账户；5) 增加用户信息；6) 更新用户名；7) 更新用户密码

## (2) 物品 dao 层设计

ItemDao
<pre> +addItem(newItem:Item):int +deleteItemById(itemId:int):int +queryItemById(itemId:int):Item +queryItemByName(itemName:String):List&lt;Item&gt; +queryItemByItemType(itemType:String):List&lt;Item&gt; +queryItemByUserId(postUserId:int):List&lt;Item&gt; +queryAllItem():List&lt;Item&gt; +queryItemByPage(page:int):List&lt;Item&gt; </pre>

说明：以上方法从上至下分别为：1) 增加商品信息；2) 删除商品信息；3) 通过商品 ID 查找商品信息；4) 通过商品名查找商品信息；5) 通过商品种类查找商品信息；6) 通过发布者 ID 查找商品信息；7) 查找全部商品信息；8) 分页查看商品信息

## (3) 邮件 dao 层设计

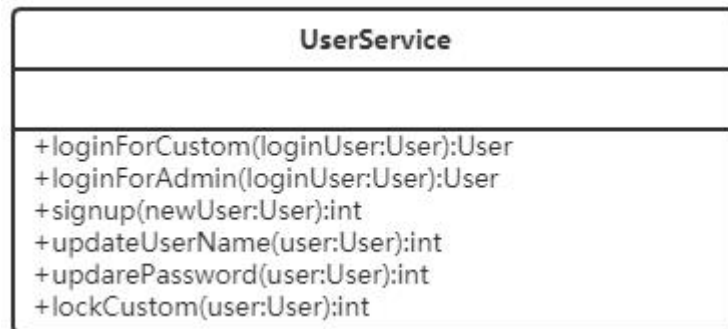
MailDao
<pre> +addMail(newMail:Mail):int +deleteMail(mailId:int):int +updateIsRead(mailId:int):int +queryMailBymailId(mailId:int):Mail +queryAllMail():List&lt;Mail&gt; +querySendMail(sendUserId:String):List&lt;Mail&gt; +querySendMailByPage(page:int,sendUserId:String):List&lt;Mail&gt; +queryReceMail(sendUserId:String):List&lt;Mail&gt; +queryReceMailByPage(page:int,sendUserId:String):List&lt;Mail&gt; +updateSendIdToNULL(mailId:int):int +updateRecIdToNULL(mailId:int):int </pre>

说明：以上方法从上至下分别为：1) 新建邮件并发送；2) 删除邮件；3) 标记邮件为已读；4) 通过邮件 ID 查找邮件；5) 查询全部邮件；6) 查询已发送

的邮件；7) 分页查询已发送的邮件；8) 查询已收到的邮件；9) 分页查询已收到的邮件；10) 将邮件的发送 ID 设置为 NULL(通过设置发送 ID 为 NULL 实现删除自己已发送文件的同时，让接收方依然内查看到该邮件)；11) 将邮件的接受 ID 设置为 NULL(通过设置接收 ID 为 NULL 实现删除自己已接收文件的同时，让发送方依然内查看到该邮件)

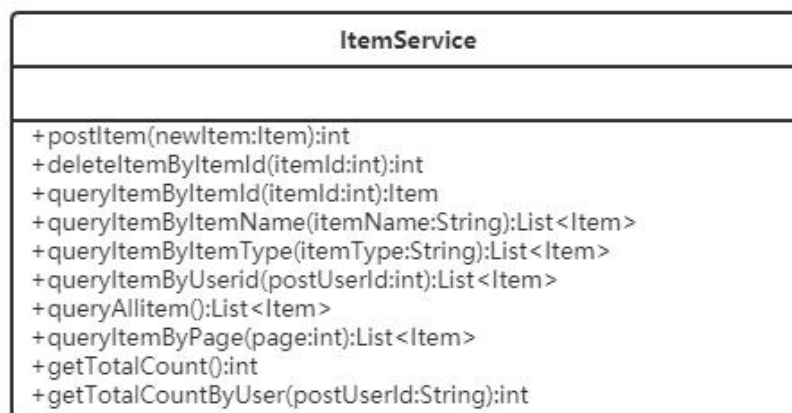
### 3.1.4 业务逻辑 service 层的设计

#### (1) 用户 service 层设计



说明：以上方法从上至下分别为：1) 客户登录；2) 管理员登录；3) 注册新账户；4) 更新用户名；5) 更新用户密码；6) 管理员封禁客户

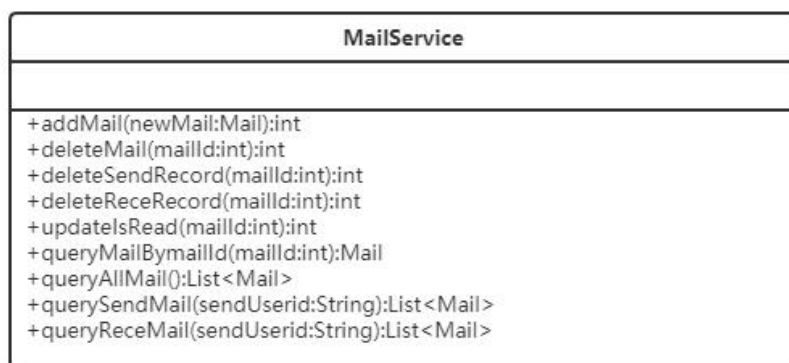
#### (2) 物品 service 层设计



说明：以上方法从上至下分别为：1) 发布新商品；2) 删除商品信息；3) 通过商品 ID 查找商品信息；4) 通过商品名查找商品信息；5) 通过商品种类查找商品信息；6) 通过发布者 ID 查找商品信息；7) 查找全部商品信息；8) 分页查看商品信息；9) 获取商品总数量(用于分页)；10) 获取某一用户发布商品总数量(用于分页)

#### (3) 邮件 service 层设计





说明：以上方法从上至下分别为：1）发送新邮件；2）删除邮件；3）删除发送记录；4）删除接收记录；5）标记邮件已读；6）通过邮件 ID 查找邮件；7）查找全部邮件；8）查找已发送邮件；9）查找已接收邮件

## 3.2 功能模块设计

### 3.2.1 功能模块划分

基于本文前面对二手物品交易平台的需求分析，将系统主要的业务功能模块设计为账户管理、交易管理、通信管理、审核管理，为卖家、买家以及管理员等用户提供良好的服务。卖家通过注册账号并登陆，在平台上发布交易信息；而买家同样通过注册账号并登陆，在平台上寻找合适的物品，通过平台达成交易，平台详细的模块如图 2.1 所示。

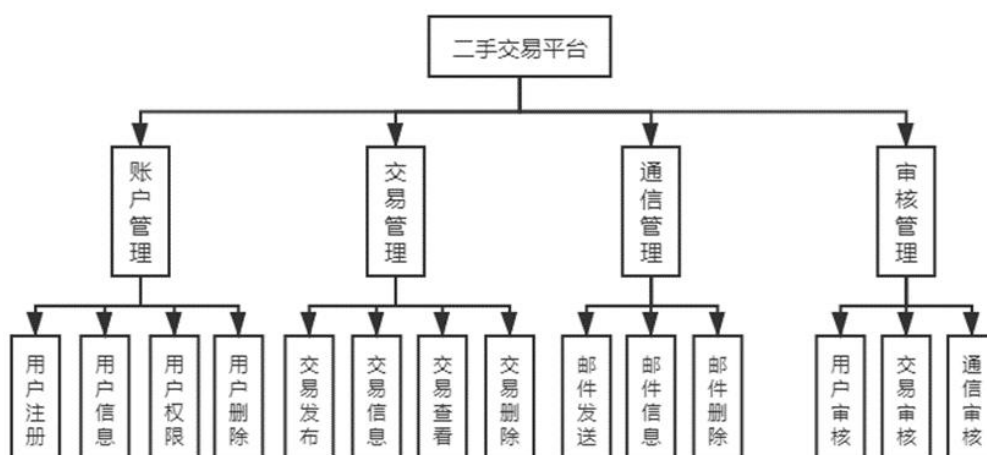


图 2.1 二手物品交易平台详细功能模块划分图

我们知道，对于一个交易平台，最重要的模块便是账户管理模块以及交易管理模块，而通信管理模块可以作为该平台功能的扩展，而审核管理则可以整合到账户管理之中。所以下面我们将对账户管理模块和交易管理模块进行详细的功能点设计，具体的设计如 2.2.2、2.2.3 所示。

### 3.2.2 账户管理模块设计

在该二手交易平台中，账户管理模块主要负责账户注册、账户信息管理、账户权限设定、账号删除等功能。用到的类主要有用户类及管理员类如下：



所需资源如下:

**View:** login.jsp; signup.jsp; home.jsp; selfCenter.jsp; updateUserInfo.jsp

**Entity:** User.java

**Service:** UserService.java

**Dao:** UserDao.java

**Controller:** LoginServlet.java; SignupServlet.java; UpdateUserName.java; UpdatePassword.java

以下为具体功能点设计:

### 3.2.2.1 用户注册功能点设计

为保证平台用户质量,用户注册时需要填写相关个人信息,其中不止包括用户名、密码、电话号码、地址等使用平台所需的必填项,还用有完善个人形象的非必要填写项,例如性别、生日、喜好等。填写中若触发违禁词,将自动屏蔽。填写好所有必填项后,经过管理员审核,即可完成注册。

所填项目	填写要求	是否必填
用户 ID	数字。具有唯一性	Y
用户名	字符	Y
密码	字母+数字	Y
电话号码	合法电话号码	Y
地址		Y
性别	选择项 男、女	N
生日	日期选择	N
喜好		N

业务逻辑:用户先通过 signup.jsp 页面进行账号注册,若检测带账号已经存在,则系统将返回相关信息,并不通过此次注册;若检测账号不存在,则创建账号,将账号信息存入数据库,并返回相关信息,通过此次注册。

系统用户注册及审核的详细序列图如图 2.2 所示。

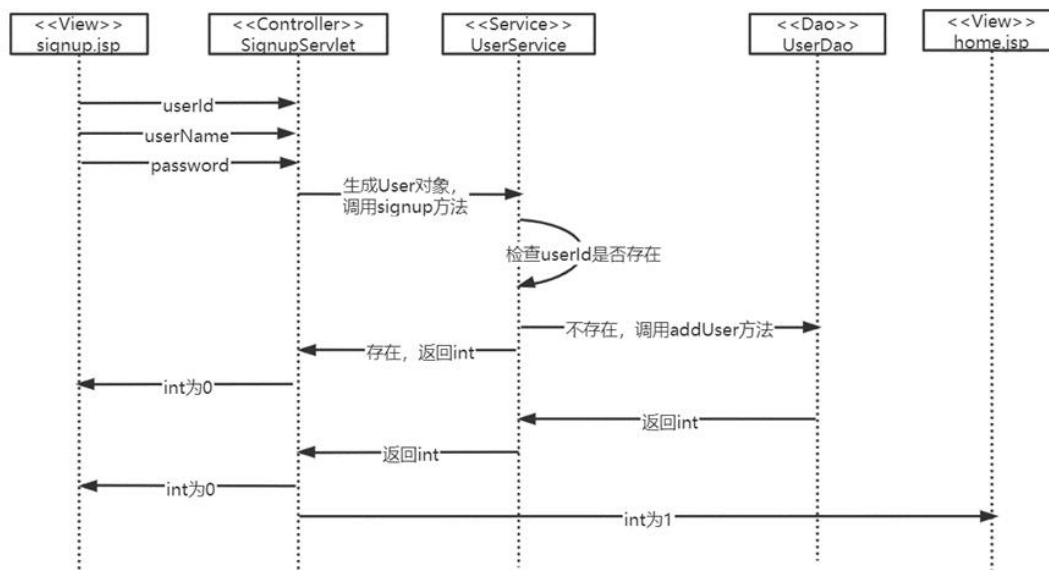


图 2.2 系统用户注册及审核的详细序列图

### 3.2.2.2 用户登录功能点设计

业务逻辑:该功能较为简单。在登录界面中,正确输入用户 ID 与密码即可登录跳转。否则系统将给出相应的错误信息,例如:账户名错误、密码错误、验证码错误等等

用户登录的详细序列图如图 2.3 所示。

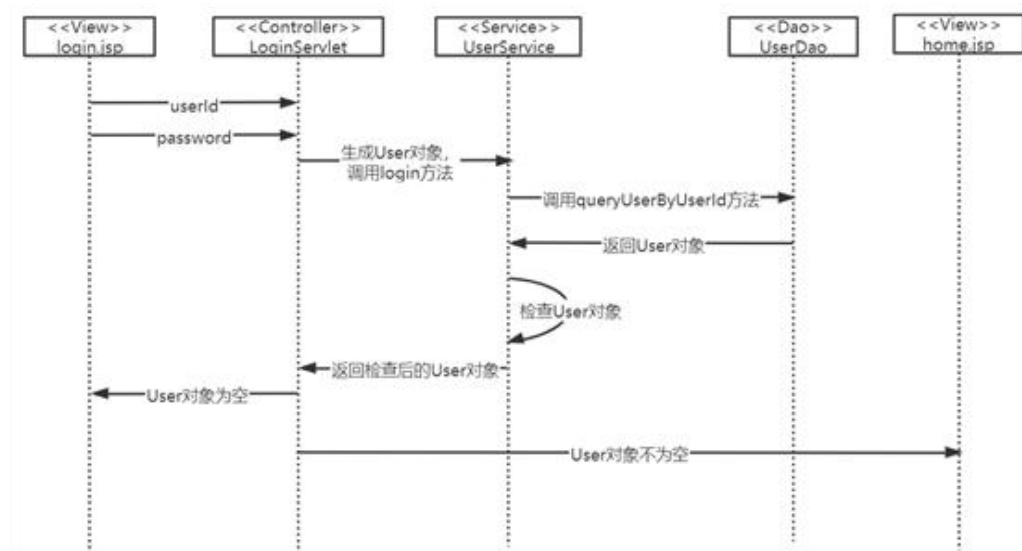


图 2.3 用户登录详细序列图

### 3.2.2.3 用户名修改功能点设计

业务逻辑:由于用户 ID 才是唯一标识某位用户的信息,用户名的作用更多是在用户间交流起到更方便、更贴近日常交流行为的称谓作用,同时为了满足用户不同时期对称谓的不同想法,因此对用户名进行可修改的设计。用户只需在登陆

后，进入个人信息，就可轻松修改。  
用户名修改的详细序列图如图 2.4 所示。

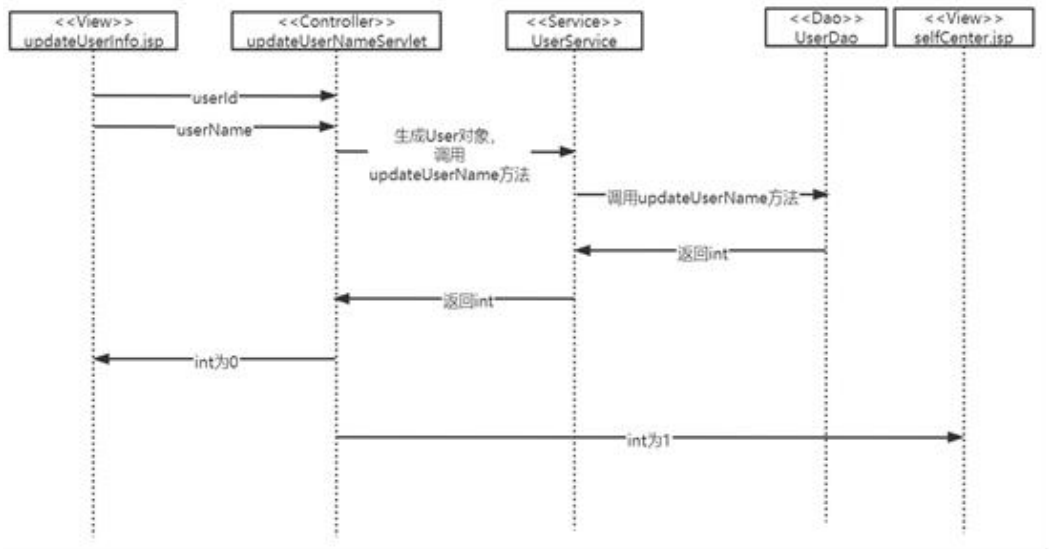
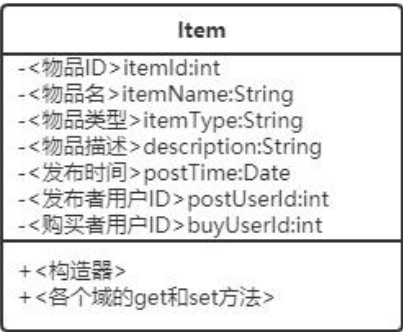


图 2.4 用户名修改的详细序列图

3.2.3 交易管理模块设计

此模块是对在平台上所展示的二手交易品的信息进行管理。主要包括交易品信息的上传、展示、搜索、删除等功能点。所使用的类：



所需资源如下：

**View:** postUsedItem.jsp; showUsedItem.jsp; usedItemDetail.jsp; selfCenter.jsp; home.jsp

**Entity:** UsedItem.java

**Service:** UsedItemService.java

**Dao:** UsedItemDao.java

**Controller:** PostUsedItemServlet.java; GetUsedItemByPageServlet.java; GetUsedItemByIdServlet.java; GetUsedItemByTypeServlet.java; GetUsedItemByNameServlet.java; GetUsedItemByUserIdServlet.java; DeleteUsedItemServlet.java

以下为具体功能点设计：

3.2.3.1 发布二手物品信息功能点设计

业务逻辑：二手物品的发布与用户注册具有相似性。同样需要填写信息，例如物品名称、物品类型、物品状态等。正确填写完毕后，若通过审核，即可正常发布信息；若审核未通过，则会被退回，系统将会返回相关退回信息。

发布二手物品的详细序列图如图 2.5 所示。

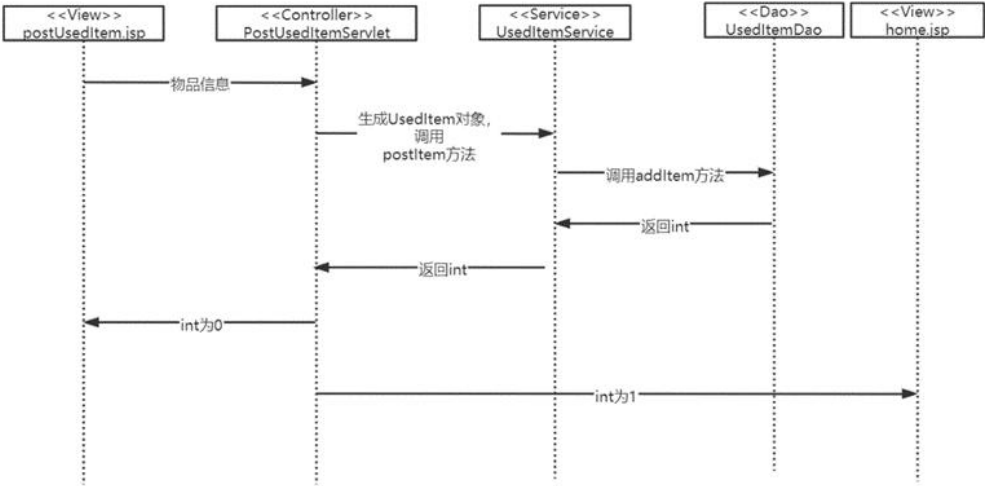


图 2.5 发布二手物品的详细序列图

3.2.3.2 分页显示二手物品信息功能点设计

业务逻辑：平台上所有物品统一在分页上进行显示，方便用户浏览，找到自己所需物品。

分页显示二手物品信息的详细序列图如图 2.6 所示。

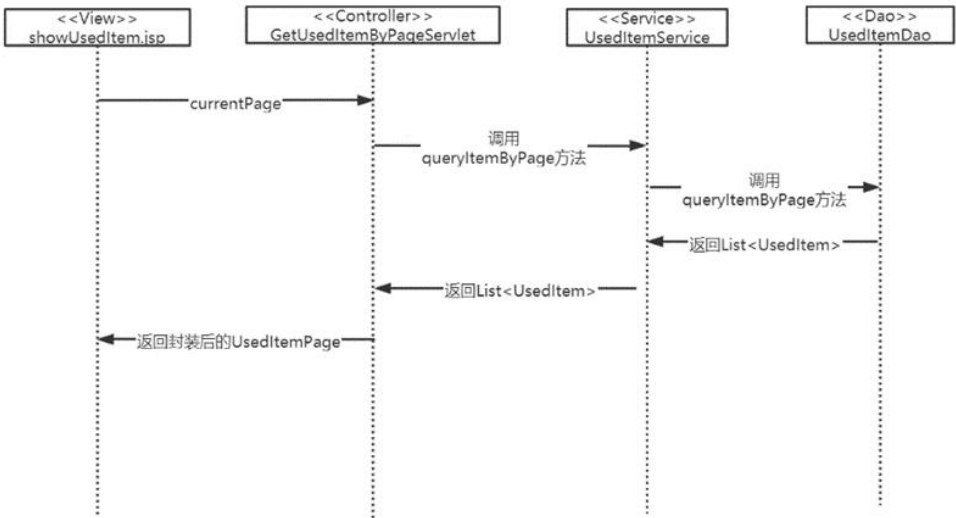


图 2.6 分页显示二手物品信息的详细序列图

3.2.3.3 查看二手物品详细信息功能点设计

业务逻辑：由于本平台分页显示的仅是大概信息，用户对某个物品具有进一

步了解的兴趣之后，可通过点击，看到相关物品的详细信息。  
查看二手物品详细信息的详细序列图如图 2.7 所示。

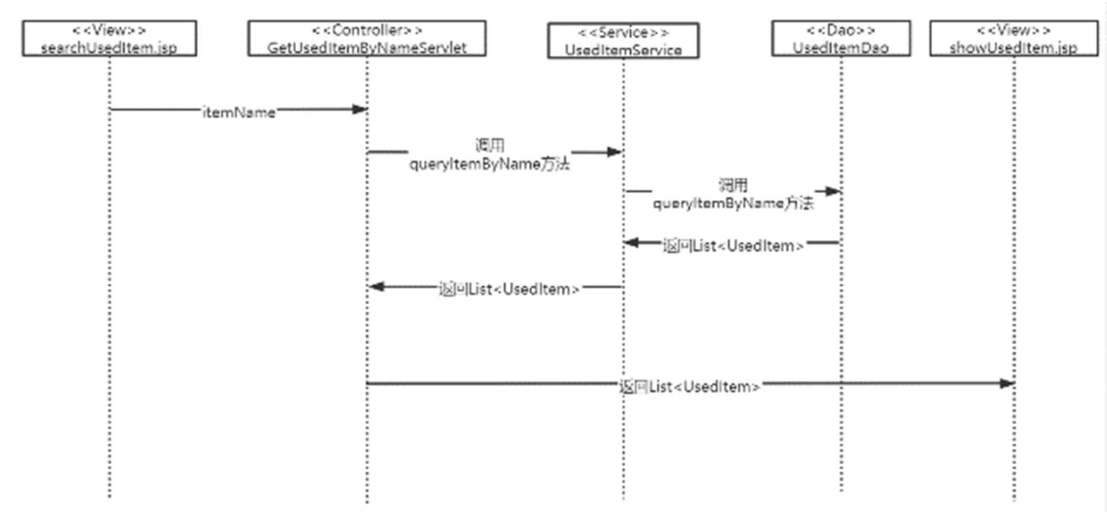


图 2.7 查看二手物品详细信息的详细序列图

3.2.3.4 搜索二手物品功能点设计

业务逻辑：除了在分页上随意浏览获取二手物品信息外，此平台还将提供更加目的性更强的搜索功能。用户可以根据自己的偏好或者需要，使用关键词、类型等不同方式，细类搜索物品。效率更高，达成交易概率更大。

搜索二手物品的详细序列图如图 2.8、图 2.9 所示。

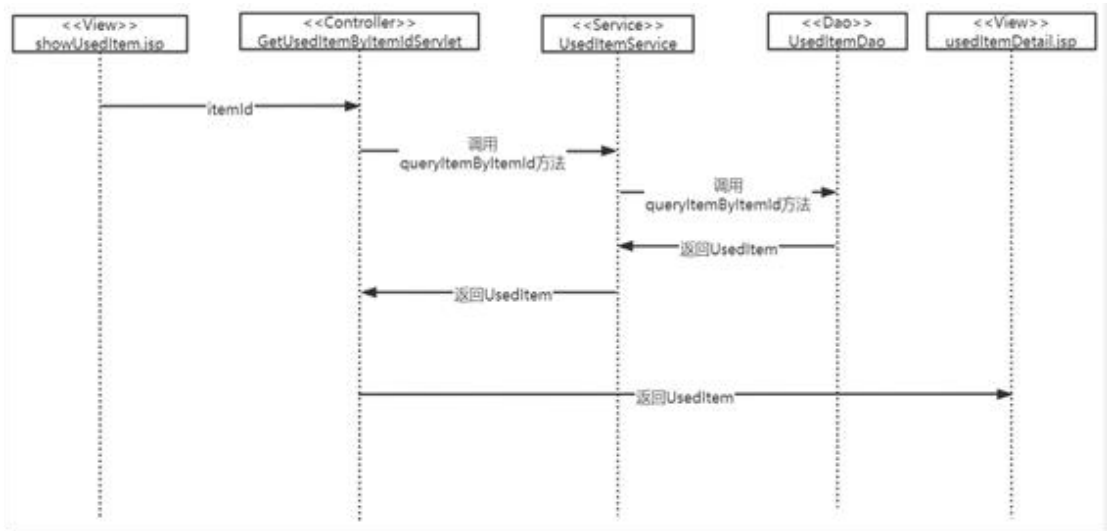


图 2.8 按物品 ID 搜索二手物品的详细序列图

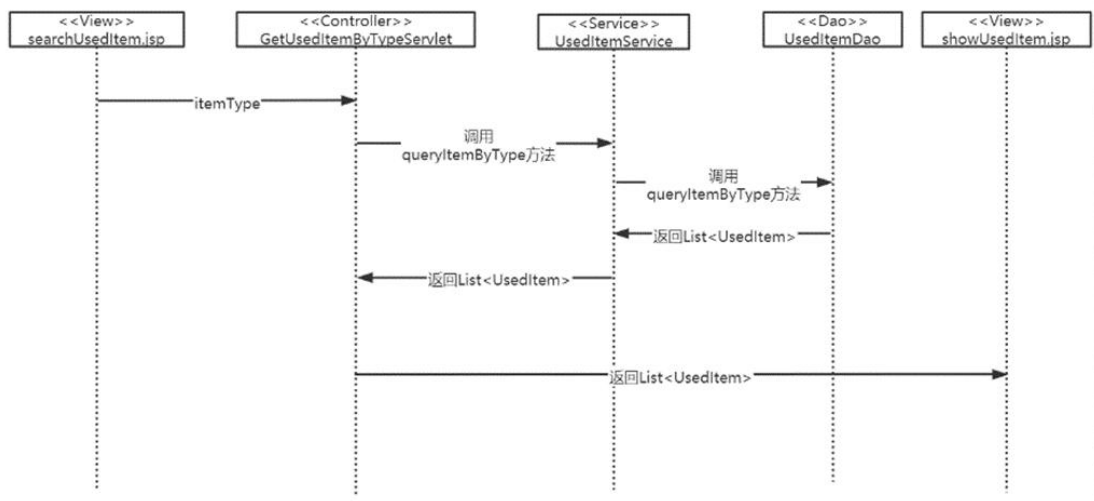


图 2.9 按物品类型搜索二手物品的详细序列图

### 3.2.3.5 删除二手物品功能点设计

业务逻辑：如果用户通过其他方式达成了二手交易或由于其他原因无需再在平台上挂出物品，则可以删除此物品信息。或者管理员在发现发布的物品有违规现象，则可以删除此物品信息

删除二手物品的详细序列图如图 2.10 所示。

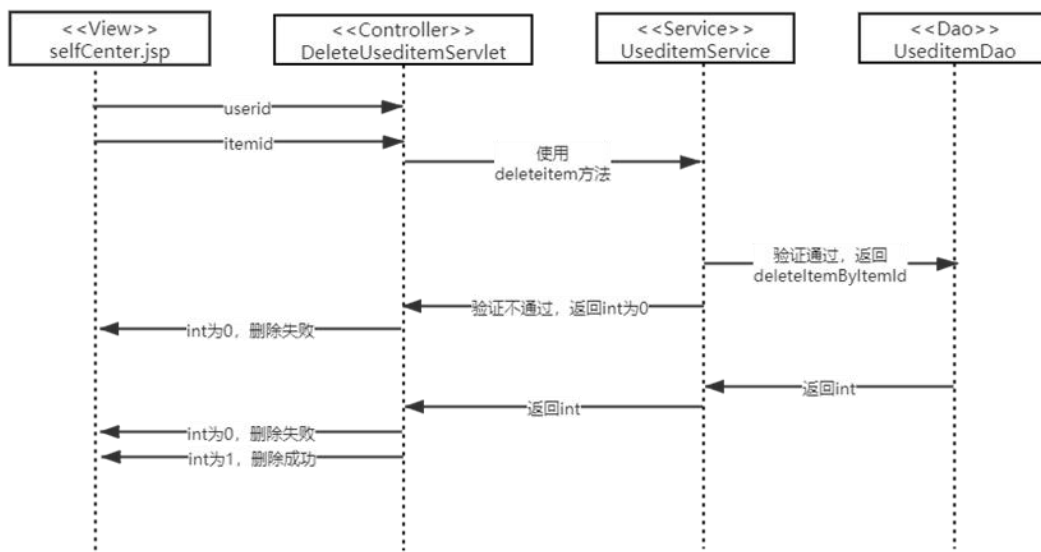


图 2.10 删除二手物品的详细序列图

## 3.3 数据库设计

本小节的工作内容是对二手物品交易平台数据信息存储的数据库进行设计。根据系统业务功能模块设计，构建系统数据信息存储的数据库，当用户点击功能模块进行操作时，系统负责调用业务逻辑程序对应功能模块的数据信息进行处理。



本系统采用 openGauss 作为系统开发数据库。

系统数据库的设计分为逻辑结构设计和详细数据设计两个方面，数据库逻辑设计主要是针对数据库的存储进行数据库实体的提取，数据库详细设计主要是根据数据库实体进行数据库表格的详细设计。

3.3.1 逻辑结构设计

在二手物品交易平台中，数据库的信息存储是整个平台的核心，各个业务功能模块的点击都需要对对应模块的数据库信息进行操作，因此，为了确保数据库数据信息存储的完整性及一致性，首先进行逻辑结构的设计，从而确定系统的数据库实体。本文为平台设计的数据库实体有用户（其中包括客户和管理员）、商品、邮件、投诉等，具体的设计结果通过 E-R 图进行展现，在 E-R 图中给出每一个数据库实体对应的属性信息以及他们之间的联系，本平台的详细 E-R 图如图 2.11 所示。

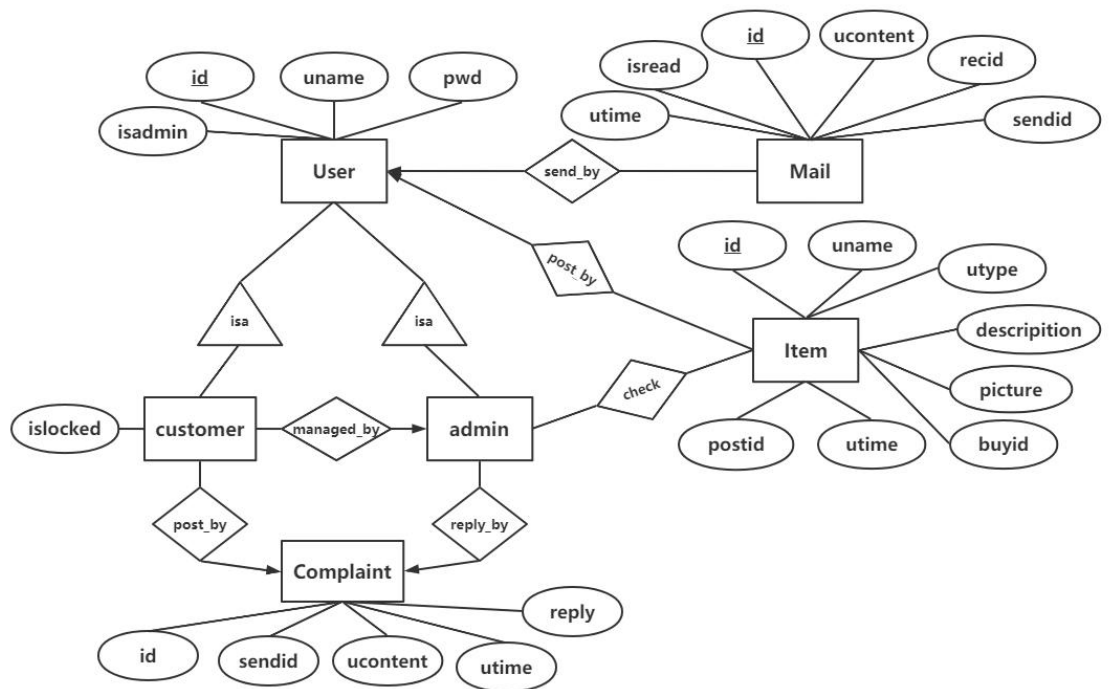


图 2.11 二手物品交易平台 E-R 图

3.3.2 物理结构设计

根据上文对系统数据库的逻辑设计，参照数据库设计理论，面二手物品交易平台主要数据用表如表 2.1 所示。

表 2.1 系统主要数据表清单

序号	数据表	数据表描述
1	User	用户信息表
2	Item	商品信息表
3	Mail	邮件信息表
4	Complaint	投诉信息表

系统关键数据表设计详情如下：

（1）用户信息表，表名为 User，数据设计详情如表 2.2 所示



表 2.2 用户信息数据表

字段名	描述	字段类型	可空	默认值	主键
Id	用户 ID	INT	否	无	是
Uname	用户名	VARCHAR	否	无	否
Pwd	用户密码	VARCHAR	否	无	否
Islocked	账号是否被封	INT	否	无	否
Isadmin	是否为管理员账号	INT	否	无	否

(2) 商品信息表，表名为 Item，数据设计详情如表 2.3 所示。

表 2.3 商品信息数据表

字段名	描述	字段类型	可空	默认值	主键
Id	物品 ID	INT	否	无	是
Uname	物品名称	VARCHAR	否	无	否
Utype	物品类型	VARCHAR	否	无	否
Description	物品描述	VARCHAR	否	无	否
Picture	物品图片（名字）	VARCHAR	否	无	否
Utime	发布时间	DATE	否	无	否
Postid	发布用户 ID	INT	否	无	否
Buyid	购买者用户 ID	INT	否	无	否

(3) 邮件信息表，表名为 Mail，数据设计详情如表 2.4 所示。

表 2.4 邮件信息数据表

字段名	描述	字段类型	可空	默认值	主键
Id	邮件消息 ID	INT	否	无	是
Sendid	发送方 ID	INT	否	无	否
Recid	接收方 ID	INT	否	无	否
Ucontent	邮件内容	VARCHAR	否	无	否
Utime	发送时间	DATE	否	无	否
Isread	是否已读	INT	否	无	否

(4) 投诉信息表，表名为 Complaint，数据设计详情如表 2.4 所示。

表 2.4 邮件信息数据表

字段名	描述	字段类型	可空	默认值	主键
Id	投诉 ID	INT	否	无	是
Sendid	投诉者 ID	INT	否	无	否
Ucontent	投诉内容	VAECHAR	否	无	否
Utime	发送时间	DATE	否	无	否
Reply	管理员回复	VARCHAR	是	无	否

## 4. 项目开发

### 4.1 技术方案

本项目开发过程中，我们主要运用到的技术方案包括 MVC 设计模式、MyBatis

数据访问框架、Bootstrap 和 javascript 前端框架等，这些框架更加有利于我们项目的搭建和各项功能的具体实现。下面我们按照 MVC 模式逐层讲解我们的技术方案和路线。

### 4.1.1 实体类 entity 技术方案

首先我们来阐述本项目的实体类，即 entity。本项目一共有 5 个实体类，如下图所示：

```
▼ edu.njust.entity
  > Complaint.java
  > Item.java
  > Mail.java
  > Page.java
  > User.java
```

首先是最基础的 **User 实体**，其主要用于存储用户的基本信息。由于管理员和普通用户的账户信息基本一致，包括：账号 ID、账号名、账号密码，所以我们在 User 实体中加入域 isAdmin 来判别用户类型，其中 isAdmin=1 为管理员，isAdmin=0 为普通用户。为了管理员更加便利的管理用户账号，我们又在 User 实体中加入了域 isLocked 来判断用户账号状态，其中 isLocked=1 表示账号被封禁，isLocked=0 表示账号正常。值得一提的是，对于管理员而言 isLocked 永远为 0，其不会被封禁。

其次是最为重要的 **Item 实体**，其用于存储商品的基本信息。不同的商品由商品 ID 来唯一识别。所以我们允许存在用户设置重名的商品。该实体除了基本商品信息的存储，我们还加入了域 buyUserId，来存储商品的购买状况，若有人进行了购买，则将其设置为购买人的账号 ID，若没有人购买则设置为-1。

之后是 **Mail 实体**和 **Complaint 实体**，两者将为类似。只不过 Mail 实体中存储有收发双方的 ID 信息，而 Complaint 实体仅包括发布者的 ID 信息。此外 Complaint 实体中还包含域 reply，该域用于存储管理员对投诉的相关回复，如果管理员未回复，则是 null。

最后是 **Page 实体**，该实体仅用于 shopmain.jsp 的分页显示商品功能。其中存储有当前页面 nowpage、最小页面(基本为 1)minpage 以及当前商品数量所能达到的最大页数 maxpage。

### 4.1.2 数据访问 dao 层技术方案

下面我们来介绍本项目的数据库访问 dao 层。该层使用了 MyBatis 框架。要使用 MyBatis 首先要写好它的配置文件 mybatis-config.xml。

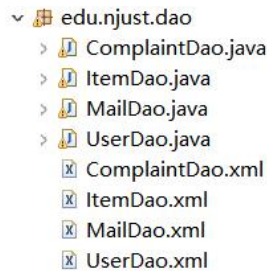
```
<environment id="9191062301-test">
  <transactionManager type="JDBC"/>
  <dataSource type="POOLED">
    <property name="driver" value="org.postgresql.Driver"/>
    <property name="url" value="jdbc:postgresql://192.168.56.102:26000/shop"/>
    <property name="username" value="dboper"/>
    <property name="password" value="dboper@123"/>
  </dataSource>
</environment>
```

上图为 mybatis-config.xml 文件中配置开发环境的部分，可以看到：该项目使用的是 JDBC 数据库连接方式，驱动为 org.postgresql.Driver（openGauss 数据库就是基于 postgresql 数据库开发的）还有数据库连接地址 url、数据库

用户名 username 以及数据库密码 password。这些代码是 MyBatis 连接数据库的基础。

```
<mappers>
  <mapper resource="edu/njust/dao/UserDao.xml"/>
  <mapper resource="edu/njust/dao/ComplaintDao.xml"/>
  <mapper resource="edu/njust/dao/MailDao.xml"/>
  <mapper resource="edu/njust/dao/ItemDao.xml"/>
</mappers>
```

上图为 mybatis-config.xml 文件中配置映射文件的部分，可以看到我们为每一个数据访问 dao 都设置了对应的映射文件。



上图则是 edu.njust.dao 包里有数据库访问的所有文件，可以看到每个数据访问 Dao 都包含一个映射文件 xml 和一个与之相对应的 java 文件。下面我们将以 UserDao 中的 findAlluser 方法来描述数据访问的具体实现：

(1) 由于数据库的字段命名和实体类的域名不同，因此还需要在 xml 映射文件中加入 resultMap 如下图所示：

```
<resultMap id="UserResultMap" type="edu.njust.entity.User">
  <result column="id" property="userId"/>
  <result column="uname" property="userName"/>
  <result column="pwd" property="password"/>
  <result column="isadmin" property="isAdmin"/>
  <result column="islocked" property="isLocked"/>
</resultMap>
```

(2) 在相应的文件 xml 映射文件中书写 sql 语句，如下图所示为 findAlluser 访问对应的 xml 映射：

```
<select id="findAlluser" resultType="edu.njust.entity.User" resultMap="UserResultMap">
  select * from login
</select>
```

首先我们设定为查询功能 select，并设置 id 为方法名 findAlluser（这样不易混淆）。返回类型 resultType 为 User 实体，并运用 resultMap 让数据库字段和相应的实体一一对应。最后书写 sql 语句。

(3) 在 java 文件中我们直接调用 mapper 文件中配置的 sql 语句：

```
public List<User> findAlluser() throws IOException{
    String resource = "mybatis-config.xml";
    InputStream inputStream = Resources.getResourceAsStream(resource);
    SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);
    SqlSession sqlSession = sqlSessionFactory.openSession();
    List<User> list = sqlSession.selectList("UserDao.findAlluser");
    for(int i=0;i<list.size();i++)
    {
        User user=list.get(i);
        System.out.println(user.getId()+" "+user.getName()+" "+user.getPwd()+" "+user.getIsAdm
    }
    return list;
}
```

该方法首先定位核心配置文件 mybatis-config.xml，之后获得 SqlSessionFactory、获得 SqlSession 并调用在 mapper 文件中配置的 sql 语句。

调用 mapper 中的方法：（命名空间.id, 参数）。

#### 4.1.3 业务逻辑 service 层技术方案

下面我们来介绍本项目的业务逻辑层，即 service。本项目一共设置了 4 个 service，每个 service 都与相应的 dao 层一一对应，如下图所示：

```

v edu.njust.service
> ComplaintService.java
> ItemService.java
> MailService.java
> UserService.java

public class UserService {
    public int checklogin(User user1) throws IOException{}
    public int checksignup(String newname) throws IOException{}
    public int findIdByName(String username) throws IOException{}
    public ArrayList findAll() throws IOException{}
    public int setLocked(int islocked,int id) throws IOException{}
    public boolean addnewUser(User newuser) throws IOException{}
}

```

首先是 **UserService**，其与 dao 层的 UserDao 相对应。其一共包含 6 个方法。分别为：checklogin、checksignup、findIdByName、findAll、setLocked、addnewUser。其中 checklogin 用于检测用户输入的账号名是否存在，以及密码十分正确、账号是否被封。若检测无误则可成功登陆（返回 0），否则返回错误信息。checksignup 用于检查用户新创建的用户名是否重名，未重名则可成功创建（返回 0），否则返回错误信息。findIdByName 用于通过用户名查找用户 ID。findAll 用于返回全部用户信息。addnewUser 则用于注册时新增用户。

```

public class ItemService {
    public ArrayList findByPage(int n,int m) throws IOException{}
    public int getCount() throws IOException{}
    public Item findById(int itemid) throws IOException{}
    public int buyItem(int itemid,int buyid) throws IOException{}
    public ArrayList findOrder(int userid) throws IOException{}
    public ArrayList findPost(int userid) throws IOException{}
    public boolean addNewItem(Item newItem) throws IOException{}
    public ArrayList findAll() throws IOException{}
    public boolean DelItem(int itemid) throws IOException{}
}

```

之后是 **ItemService**，其与 dao 层的 ItemDao 相对应。其一共包含 9 个方法。分别为 findByPage、getCount、findById、buyItem、findOrder、findPost、addNewItem、findAll、DelItem。其中 findByPage 用于 shopmain.jsp（商品主界面）来分页显示商品。而 getCount 用于获取数据库中商品的记录数量来辅助商品分页显示。findAll 用于返回全部商品信息。buyItem 用于用户购买商品时，改变商品购买信息。findOrder 和 findPost 分别用于查找用户的订单信息和发布商品信息。addNewItem 用于用户发布新商品。DelItem 用于管理员退回商品。

```

public class MailService {
    public ArrayList findRecMail(int userid) throws IOException{}
    public ArrayList findSendMail(int userid) throws IOException{}
    public int haveread(int mailid) throws IOException{}
    public boolean postMail(Mail mail) throws IOException{}
    public ArrayList findAll() throws IOException{}
    public boolean delMail(int mailid) throws IOException{}
}

```

之后是 **MailService**，其与 dao 层的 MailDao 相对应。其一共包含 6 个方法。分别为：findRecMail、findSendMail、haveread、postMail、findAll、delMail。



其中 findRecMail 和 findSendMail 用于查找用户接收的邮件以及发送的邮件。haveread 用于标记邮件已读。postMail 用于用户发布邮件。findAll 和 delMail 用于管理员查看所有邮件并可删除邮件。

```
public class ComplaintService {
    public ArrayList findAll() throws IOException{}
    public int updateReply(int compid,String reply) throws IOException{}
    public ArrayList findBySendid(int userid) throws IOException{}
    public boolean addComp(Complaint comp) throws IOException{}
}
```

最后是 **ComplaintService**，其与 dao 层的 ComplaintDao 相对应。其一共包含 4 个方法。分别为：findAll、updateReply、findBySendid、addComp。其中 findAll 用于管理员查看所有投诉信息。updateReply 用于管理员回复投诉。findBySendid 用于用户查找自己的投诉记录。addComp 用于用户发布投诉。

#### 4.1.4 控制器 controller 层技术方案

Controller 层（包括 Service 层、Dao 层、entity 层）与 View 层沟通的桥梁。其在 Web 项目中扮演着十分重要的角色。本项目一共使用了 6 个 Servlet。分别为：AdminServlet、ComplaintServlet、LoginServlet、MailServlet、ShopPageServlet、SignupServlet。具体如下图所示：

```
edu.njust.controller
> AdminServlet.java
> ComplaintServlet.java
> LoginServlet.java
> MailServlet.java
> ShopPageServlet.java
> SignupServlet.java
```

其中 LoginServlet 用于登录方面的信息交互与控制。SignupServlet 用于注册方面的信息交互与控制。除了登录注册，我们选择将管理员和用户的相关功能进行分离。由于管理员端的功能相较于客户端较为简单，所以统一使用 AdminServlet 进行信息交互与控制。而客户端则将商品信息交互（ShopPageServlet）、邮件信息交互（MailServlet）、投诉信息交互（ComplaintServlet）分为三个独立的部分，这些可以使程序条理清晰、便于维护。

#### 4.1.5 表示 view 层技术方案

表示层主要是各种 jsp 页面。本项目一共使用了 21 个 jsp 文件。具体如下图所示：

complaintlist.jsp	main.jsp	reply.jsp
details.jsp	mainforadmin.jsp	shopmain.jsp
detailsforadmin.jsp	mainpageforadmin.jsp	signup.jsp
img.jsp	mycomplaint.jsp	signupFailure.jsp
login.jsp	myemail.jsp	signupSuccess.jsp
loginFailure.jsp	postemail.jsp	tradingrecord.jsp
maillist.jsp	postitem.jsp	userlist.jsp

这 21 个 jsp 文件可以分为以下 3 类：

（1）共用：

login.jsp

（登录）

loginFailure.jsp	(登录失败界面)
signup.jsp	(注册)
signupFailure.jsp	(注册失败界面)
signupSuccess.jsp	(注册成功界面)
img.jsp	(用于生成验证码, 不用 ui 设计)

### (2) 客户端:

main.jsp	(登录成功界面)
shopmain.jsp	(商品显示界面)
details.jsp	(查看商品详细)
tradingrecord.jsp	(查看交易记录)
postitem.jsp	(发布邮件)
myemail.jsp	(查看邮件)
postemail.jsp	(发邮件)
mycomplaint.jsp	(投诉界面)

### (3) 管理员端:

mainforadmin.jsp	(登录成功界面)
mainpageforadmin.jsp	(显示商品列表)
detailsforadmina.jsp	(查看商品详细)
userlist.jsp	(显示用户详细)
maillist.jsp	(显示邮件列表)
complaintlist.jsp	(显示投诉列表)
reply.jsp	(回复投诉界面)

其次我们还使用了 Bootstrap、Java Script 等前端框架来制作前端页面。我们导入了 Bootstrap 框架的 css 样式和 js, 如下图所示, 从而更加高效地完成前端界面的设计。

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<script src="js/jquery-3.5.1.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
```

## 4.2 关键模块

本项目一共包含四个主要模块, 分别为: 处理用户登录以及注册账号的**用户信息模块**; 作为主体的**二手交易模块**; 方便不同用户之间通信的**邮件通信模块**; 以及与邮件模块逻辑类似的**投诉发布模块**。

### 4.2.1 用户信息模块

用户进入网站时, 首先看见网站首页 login.jsp, 该网页主要是用于用户填写相关信息来进行系统登录。在该页面用户需要填写账号、密码、验证码、账号类别 (下拉框选择)。下图为 login.jsp 填写用户信息的表单:

```

<table>
<tr> <td style="width:200px;"> 用户名: </td>
<td style="width:250px;"><input type = "text" name = "uname"> </td><td style="width:20px;"></td> </tr><br>
<tr> <td> 密码: </td><td><input type = "password" name = "upwd"></td><td></td> </tr><br>
<tr>
<td>验证码: </td>
<td><input type="text" name="checkcode" id= "checkcode" size="4"> </td>
<td style="text-align:left;"><a href="javascript:reloadCheckImg();"></a> </td>
</tr>
<tr>
<td>登录方式:</td>
<td><select name="isAdmin" id="isAdmin" style="width: 150px;" onchange="add1(this.value);">
<option value="0">客户登录</option>
<option value="1">管理员登录</option>
</select></td>
</tr>
</table>

```

填写完所有信息后点击登录按钮，会跳转到 LoginServlet 来处理登录事务。LoginServlet 获取参数后封装成 User 对象并向下调用 UserService 的 checklogin 函数。若返回的参数为 0，则代表登录成功；否则登录失败并转到 loginFailure.jsp 页面显示错误信息。下图为具体代码：

```

User user=new User();
user.setName(username);
user.setPwd(pwd);
if(isadmin.equals("1"))
    user.setIsAdmin(1);
else if(isadmin.equals("0"))
    user.setIsAdmin(0);
UserService us=new UserService();
int result=us.checklogin(user);//返回登录信息存入result

```

还可以在 login.jsp 页面点击超链接跳转到 signup.jsp 进行注册。正确填写账号、两次密码后，点击注册跳转至 SignupServlet 来处理注册事务。SignupServlet 获取参数后封装成 User 对象并向下调用 UserService 的 checksignup 函数。

若返回的参数为 0，则代表注册成功，SignupServlet 再调用 UserService 的 addnewUser 函数完成新用户信息的添加，同时跳转至成功注册界面 signupSuccess.jsp；否则注册失败并转到 signupFailure.jsp 页面显示错误信息。下图为具体代码：

```

UserService us=new UserService();
int result=us.checksignup(newuser);
if(result==1)
{
    request.setAttribute("error_code", new Integer(2));//用户名重复
    request.getRequestDispatcher("signupFailure.jsp").forward(request,response);
}
else
{
    User user=new User();
    user.setName(newuser);
    user.setPwd(newpwd1);
    boolean rs=us.addnewUser(user);
    if(rs==true)
    {
        response.sendRedirect("signupSuccess.jsp");
    }
    else
    {
        request.setAttribute("error_code", new Integer(3));//未知重复
        request.getRequestDispatcher("signupFailure.jsp").forward(request,response);
    }
}
}

```

除此之外管理员成功登陆后可在 userlist.jsp 页面查看所有用户的信息。对于非管理员账号，管理员可以进行封禁和解封操作。点击封禁或解封按钮，将跳转至 AdminServlet 来处理相关事务。AdminServlet 获取参数后向下调用 UserService 的 setLocked 函数更改账号封禁状态。下图为封禁操作的具体代码：

```
UserService us=new UserService();
int user_id=Integer.parseInt(request.getParameter("user"));
int rs=us.setLocked(1, user_id);
```

#### 4.2.2 二手交易模块

用户在成功登陆后将跳转至 main.jsp，main.jsp 设置在 5 秒后自动转至 ShopPageServlet，ShopPageServlet 向下调用 ItemService 的 findByPage 方法分页获取商品信息。然后将获取的商品信息传递给 shopmain.jsp 进行分页显示。下图为 jsp 页面分页显示商品的主体代码：

```
<%
    ArrayList al=(ArrayList)session.getAttribute("al");
    Iterator iter=al.iterator();
    while(iter.hasNext())
    {
        Item item=(Item)iter.next();
%>
        <dl>
            <dt>
                <i></i>
                <a href="<%=request.getContextPath()%>/ShopPageServlet?op=details&item=<%=item.getId()%>">
                <!-- 
                
                </a>
            </dt>
            <dd>
                <p><a href="<%=request.getContextPath()%>/ShopPageServlet?op=details&item=<%=item.getId()%>">
                <%=item.getName() %></a></p>
                <span><%=item.getType() %></span>
            </dd>
        </dl>
    }
%>
```

在 shopmain.jsp 可以点击商品的图片或者商品的名字跳转至 ShopPageServlet。ShopPageServlet 将通过所点击商品的 ID 信息向下调用 ItemService 的 findById 方法获取单个商品信息。之后将获取的信息传递给 details.jsp，在该页面上用户可以获取商品的详细信息（在 shopmain.jsp 仅显示商品的图片、名字、种类）。下图为具体代码：

```
int item_id=Integer.parseInt(request.getParameter("item"));
ItemService is=new ItemService();
Item item=is.findById(item_id);
session.setAttribute("item_detail",item);
request.getRequestDispatcher("details.jsp").forward(request,response);
```

其次对于非自己发布的和未被购买的商品，用户可以点击购买按钮，之后页面将转至 ShopPageServlet。ShopPageServlet 通过商品的 ID 信息以及用户 ID 信息向下调用 ItemService 的 buyItem 方法，对商品进行购买操作。下图为具体代码：

```
ItemService is=new ItemService();
int item_id=Integer.parseInt(request.getParameter("item"));
int buy_id=Integer.parseInt(session.getAttribute("user_id").toString());
int rs=is.buyItem(item_id, buy_id);
```

用户点击页面上方的“交易记录”超链接，页面将转至 ShopPageServlet。



其将通过用户 ID 信息向下调用 ItemService 的 findPost 和 findOrder 方法获取用户的发布商品信息和订购商品信息。之后将这些信息传给 tradingrecord.jsp 进行显示。下图为具体代码：

```
ItemService is=new ItemService();
int userid=Integer.parseInt(session.getAttribute("user_id").toString());
ArrayList al1=is.findOrder(userid);
ArrayList al2=is.findPost(userid);
session.setAttribute("orderlist", al1);//订单信息
session.setAttribute("postlist", al2);//发布信息
request.getRequestDispatcher("tradingrecord.jsp").forward(request,response);
```

除此之外管理员还可以在 mainpageforadmin.jsp 查看所有商品信息。点击“查看详情”超链接可以转至 detailsforadmin.jsp 页面查看商品具体细节，其逻辑与用户查看商品详细逻辑相同。在 detailsforadmin.jsp 页面管理员还可以对商品进行退回操作（删除）。

#### 4.2.3 邮件通信模块

用户点击页面上方的“我的邮箱”超链接将跳转至 MailServlet。其通过用户 ID 信息向下调用 MailService 的 findRecMail 和 findSendMail 方法，分别获取用户接收的全部邮件以及用户发送的全部邮件。之后将获取到的信息创给 myemail.jsp 页面进行显示。下图为具体代码：

```
MailService ms=new MailService();
int userid=Integer.parseInt(session.getAttribute("user_id").toString());
ArrayList al1=ms.findRecMail(userid);
ArrayList al2=ms.findSendMail(userid);
session.setAttribute("recmail", al1);//接收邮件
session.setAttribute("sendmail", al2);//发送邮件
request.getRequestDispatcher("myemail.jsp").forward(request,response);
```

在 myemail.jsp 页面，用户还可以点击“标记已读”按钮，可以将未读的邮件标记已读。

用户点击页面上方的“发送邮件”超链接将跳转至 postemail.jsp 页面。在该页面正确填写相关邮件信息，包括收件人 ID、详细内容、发送时间（系统默认为当前时间）后，即可正常发送。下图为具体的表单代码：

```
<form name="add" action="MailServlet" onsubmit="return validateForm()" method="get">
  <table border="1">
    <tr><td style="font-size:1.5vw;">收件人ID:<input type="hidden" name="op" value="newmail"> </td>
    <td><input type="text" name="recid"/></td></tr>

    <tr><td style="font-size:1.5vw;">邮件内容:<input type="hidden" name="pg" value="none"></td>
    <td><input type="text" name="content" style="height:100px;width:600px"/></td></tr>
    <tr> <td style="font-size:1.5vw;"> 发送时间: </td>
    <td><input type="text" name="mailtime" value="%<%=currentTime.toLocaleString() %>"></td>
  </tr>
</table>
<input type="submit" value="发布"/>
<input type="reset" value="重置" />
</form>
```

正确填写上述表单后，点击发布按钮，将跳转至 MailServlet。其将用户所填写信息封装在 Mail 实体中并向下调用 MailService 的 postMail 方法进行邮件的发送操作。下图为具体代码：

```

int userid=Integer.parseInt(session.getAttribute("user_id").toString());
MailService ms=new MailService();
Mail mail=new Mail();
mail.setsendId(userid);
mail.setreceId(Integer.parseInt(request.getParameter("recid")));
mail.setcontent(request.getParameter("content"));
java.text.SimpleDateFormat formatter = new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
try
{
    mail.setTime(formatter.parse(request.getParameter("mailtime")));
}
catch (ParseException e)
{
    e.printStackTrace();
}
boolean rs=ms.postMail(mail);

```

除此之外管理员还可以在 maillist.jsp 页面中查看所有的邮件信息。在该页面中管理员还可以进行退回邮件操作（删除）。

#### 4.2.4 投诉发布模块

用户点击页面上方的“我要投诉”超链接将跳转至 ComplaintServlet。其通过用户 ID 信息向调用 ComplaintService 的 findBySendid 方法获取用户的全部投诉信息并将其传递给 mycomplaint.jsp 页面进行显示。下图为具体代码：

```

ComplaintService cp=new ComplaintService();
int userid=Integer.parseInt(session.getAttribute("user_id").toString());
ArrayList al=cp.findBySendid(userid);
session.setAttribute("al", al);
request.getRequestDispatcher("mycomplaint.jsp").forward(request,response);

```

同样是在 mycomplaint.jsp，用户可以通过填写相关表单发布投诉。下图为具体的表单代码：

```

<form name="add" action="ComplaintServlet" onsubmit="return validateForm()" method="get">
    <table border="1">
        <tr><td style="font-size:1.5vw;">投诉内容:<input type="hidden" name="op" value="newcomp"></td>
        <td><input type="text" name="content" style="height:100px;width:600px"/></td></tr>

        <tr> <td style="font-size:1.5vw;"> 发送时间: </td>
        <td><input type="text" name="comptime" value="%<%=currentTime.toLocaleString()%>"></td></tr>
    </table>
    <input type="submit" value="发布"/>
    <input type="reset" value="重置" />
</form>

```

正确填写上述表单之后，点击发布将跳转至 ComplaintServlet。其将用户所填信息封装在 Complaint 实体类中并向调用 ComplaintService 的 addComp 方法将用户的投诉信息存储进数据库中。下图为具体代码：

```

Complaint comp=new Complaint();
String content=request.getParameter("content");
int userid=Integer.parseInt(session.getAttribute("user_id").toString());
java.text.SimpleDateFormat formatter = new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
try
{
    comp.setPosttime(formatter.parse(request.getParameter("comptime")));
}
catch (ParseException e)
{
    e.printStackTrace();
}
comp.setSendid(userid);
comp.setContent(content);
ComplaintService cp=new ComplaintService();
boolean rs=cp.addComp(comp);

```

除此之外管理员还可以在 complaint.jsp 页面中查看所有的投诉信息。其逻辑与用户查看投诉信息类似。其次，管理员还可以对回复为空的（reply==null）投诉进行回复处理。点击回复将跳转至 reply.jsp 页面，管理员在此处填写信息并点击回复后，便会跳转至 AdminServlet 页面，其根据投诉的 ID 信息向调用

ComplaintService 的 updateReply 方法对投诉的回复信息进行修改。下图为具体代码：

```
int comp_id=Integer.parseInt(session.getAttribute("comp").toString());
ComplaintService cs=new ComplaintService();
String content=request.getParameter("content");
int rs=cs.updateReply(comp_id, content);
```

## 4.3 特殊问题以及解决方法

### 问题 1

**问题描述：**在成功建立 openGauss 数据库后，导入 postgresql.jar 包后运行程序，发现程序一直报错，报错信息为“Unsupported major.minor version 52.0”。

**解决方案：**通过查阅资料发现上述报错信息表示 postgresql.jar 驱动需要 1.8 版本以上的 jdk 版本才能使用。而当时我们默认使用的是 jdk1.7。于是我们将 Tomcat 的 jdk 版本更改为本地的 1.8 版本。

**解决结果：**程序不再报错能够成功运行。

### 问题 2

**问题描述：**在 openGauss 数据库中建立表 User 后，发现访问数据库出现异常，似乎无法访问正确的 User 表。

**解决方案：**通过在华为微信群内询问，了解到 openGauss 数据库中每新建一个数据库，该数据库会自带一个名为 user 的函数。由于数据库不区分大小写，所以新建的 user 表与数据库的 user 函数发生了重名冲突。所以我们最终将数据库表名改为 login。

**解决结果：**可以成功访问数据库。

### 问题 3

**问题描述：**由于商品信息中包含图片信息，而图片信息无论是传递还是入库存储都难以实现。我们也查阅了相关资料，有将图片转换为 bit 流存入数据库的方法，但实现过于困难。

**解决方案：**最终我们以存储图片名字，并使用 commons.fileupload 包和 commons.io 包将图片导入指定的文件夹的方式来解决图片存储问题，这样我们就可以通过图片的名字以相对路径来访问图片并在页面上加以显示。

**解决结果：**可以成功在页面上显示图片。

## 5. 个人体会

### 5.1 孙傲歆的个人小结

在本次项目开发中，我主要负责 openGauss 数据库的搭建、服务器中项目的部署、用户端的后端开发以及最终前后端代码的整合。

从个人层面来说，这次课程设计让我学到了许多新的知识。首先是 openGauss 数据库的搭建。我们知道 openGauss 数据库是基于 postgresql 开发的。而在之前的 J2EE 课程中，老师要求我们使用的是 mysql 数据库。虽然不同数据库的基本 sql 语句没有什么差别，但是在数据库搭建上却有着极大的不同。而且

openGauss 数据库必须搭建在 Linux 操作系统中，这对于从来没有接触过 Linux 操作的我也是一大挑战。其次本次课程还要求我们使用 mybatis 框架进行数据访问。由于 mybatis 是我从来没有接触过的东西，也没有什么现成的学习资料可以使用。所以我在 csdn 等论坛上查阅了大量资料，从什么是 mybatis、mybatis 的基本原理及作用，再到 mybatis 相关文件的编写与使用。这样一个循序渐进的学习过程。除了这些新知识，我觉得本次课设给我的最大收获是其提高了我解决实际问题的能力。在上学期的 J2EE 课程中，我们主要是编写一些十分简单的项目，这些小项目题材范围窄而且有十分明确的需求，所以也不容易出错。但是这次项目开发要求我们自己选定课题，自己分析需求，这就使得开发的难度大大的提升了。首先我们需要考虑如何选题才能在众多小组之中脱颖而出，怎样才能让我们的项目脱离那种只是简单查找逻辑的“管理系统”。而在确定选题之后我们还要分析需求、确定项目的具体功能点等等。而在实际编程中，我们还需要“一边推进一边优化”，比如说我们项目的最早版本是没有投诉模块的。我们本来的想法是将投诉模块和邮箱模块合二为一，如果用户想要投诉直接给管理员发邮件即可。但我们意识到在实际软件使用过程中，对于用户而言，其难以获取管理员的账号信息，也就难以给管理员发送邮件。而对于管理员而言，如果使用邮箱进行信息交互，可能会出现同一个管理员收到太多的投诉邮件而来不及处理的情况。所以我们加入了投诉信息模块。这个模块可以让用户更加直接的发布投诉，只需简单填写投诉信息即可。而对于管理员而言，投诉模块支持所有管理员同时进行投诉处理，大大减轻了管理员的工作量。

从团队层面来说，这次课设让我认识到了团队协作的重要性。我作为我们小组的组长，需要协调小组成员的各项工作，并且定时进行小组讨论交流小组成员之间的工作情况以及遇到的问题。虽然我们组的分工任务完成的很快，但在实际编程的过程中，却遇到了各种各样的问题，比如说参数的名称不一致、类的取名方式没规定好，或是页面跳转的时候遇到了问题。这些问题前期不会显现出来，但是到了后期每个人的工作结合的时候就会遇到各种各样因为分工时没规定好而产生的问题，严重拖累项目开发进度。这可能是我们在前期分配任务中没有协商好，以及在编程工作中没有及时进行交流导致的。但也正是通过这些错误，我们认识到了团队的前期协商的重要性。

总而言之，通过这次课设。我学到了许多新的知识，并且对以前学过的 Web 项目开发的相关知识进行了巩固与扩展。同时，我也意识到在团队合作中，想要成为一个优秀的“领头人”并不是一件易事。而且在团队合作中，合理的分工合作以及及时的交流与沟通是十分重要，不能让每个成员都闭门造车、自己干自己的。这样做前期或许看似很顺利，但到后期项目整合时会出现许多问题，严重拖慢进度。

## 5.2 李子园的个人小结

在本次项目的开发中，我主要负责和管理员部分有关的后端开发、和管理员有关前端后端的代码整合以及一些和本地文件上传方面，图片显示方面的有关的技术的研究。

从个人层面来讲，本次课设让我获益良多。首先是对上学期 J2EE 的知识巩固和提高，由于上学期我们的只是停留在对功能要求明确，功能简单的小程序阶段，而本次课设我们团队设想的是做出一个功能完备，十分人性化且实用的二手



平台交易系统。所以本次课设项目对我而言是一次极大的挑战。

本次课设中我依旧使用 mysql 数据库，巩固复习了上个学期中所学到的数据库连接，数据库操作等内容，其次是一些新知识点的掌握和从使用者出发的一些人性化的考虑，本次编写项目的过程中我遇到的第一个困难就是上传文件和显示，在最初的程序中我们的代码只能显示一些在 WebRoot 文件下的 image 文件夹中已经提前存储好的图片文件，对于提前未存储的图片文件我们无法显示，这完全不能满足我们最初设想的用户发布功能以及管理员的查看发布商品功能，于是我在 csdn 上查阅大量资料，发现现在的开发人员大都使用框架做图片上传，不过我选择了最基本的 servlet 方式，了解上传文件的过程发现上传的图片文件最终存储到 tomcat 的服务器上(我使用的是 tomcat)。最终实现图片文件的上传和显示。

其次是人性化设计，在对管理员有关的后端进行设计是我们设身处地，一边推进一边优化，比如管理员的封禁账号功能，起初的设计是封禁等同于注销，但经过我们团队的考虑与和实际情况的结合我们更改为管理员可以解封账号的形式。还有用户和管理员的沟通方面的改进比如将投诉和邮件功能分开等方面都是经过了实际的使用体验的考量和团队的讨论，起初我打算使用留言功能代替投诉和邮件功能，但是经过讨论考量这种方式限制了用户和管理员之间的双向沟通，因为用户并不知道管理员 id, 只能等管理员给用户先留言后，用户才能给管理员留言，并且功能混淆在一起使得管理员真实操作起来十分麻烦。于是我们将留言功能分成邮件和投诉两个功能，减轻了管理员的工作量。

从团队方面来讲，这次课设让我们都意识到了团队合作和为他人着想的重要性，在我们将各自的负责板块基本完成开始整合的时候发现了许多问题，首先的大家使用的参数名称不同使得整合出现了极大的困难，并且特别是用户和管理员交互的部分，发生功能上的冲突，以及无法轻易整合的尴尬，还有注释的使用问题，由于注释不充足，导致后面整合时并不清楚代码含义，严重拖延了项目开发进度，由此我们都学习到了及时沟通交流的重要行，为今后的项目开发积累了许多经验。

总之，本次课设，我受益良多，不仅对以前学习到的有关网页开发的知识进行了巩固与更新，同时第一次体会到了大家一起完成一个项目的不容易，并在其中积累了许多经验，相信在今后的项目开发中这次宝贵的经历会起到十分重要的作用。

### 5.3 郑陈烨的个人小结

在本次项目开发中，我主要负责前端界面的开发布局。在刚开始的开发阶段，基于上学期的 web 课程基础以及对 css、html 的知识了解掌握，对管理员界面进行设计开发时，选择了手打代码进行样式设计。在过程中，逐渐认识到课堂上由于教学要求等原因所了解到的页面设计是最为基础的，同时，在页面较少的情况下，对每个页面逐一从零设计还是可实现的，但在这次项目中，页面较多且大框架相似的情况下，使用合适的框架是更为高效且明智的选择。因此，在各类论坛以及学习网站进行搜索了解后，选择了 bootstrap 进行框架开发。使用框架开发不仅方便设计样式，更提高了网页的美观程度，能够提供用户更好的使用体验。

通过这次项目设计，了解到自身在这方面知识的缺漏，并对此进行学习加强，对网页设计方面的知识有了更多的了解与掌握。也只有不断地实践，才能使自己的能力有所提高与突破。同时，作为团队项目，沟通协作显然是十分重要的。沟

通不足所带来的整合问题就会在后期出现，例如页面显示内容所需的后端数据库内容等。虽然这些问题将项目完成时间延后了一点，但算是对团队合作的一道考验，也在后期的加强沟通中得到了解决，更磨合了我们这个小团队。

最后，在课设中，不仅在知识方面收获颇丰，而且对团队合作有了全新感悟，更感谢两位队友在项目过程中的帮助和贡献。以后也是要不断学习，不仅是在编程技术上，还有合作能力上，以期更好地完成项目。