

数据库课程设计报告

—小型网上书店及管理系统



姓名：汪超

班级：大数据 18

学号：201800301328

目录

一、系统开发平台.....	4
1.1 题目：小型网上书店.....	4
1.2 开发语言：java.....	4
1.3 开发工具.....	4
1.3.1 后端开发.....	4
1.3.2 前端开发.....	4
1.4 数据库 Microsoft SQL Server.....	4
1.5 操作系统：Windows 10.....	5
二、数据库规划.....	5
2.1 任务陈述.....	5
2.2 任务目标.....	5
2.2.1 前端用户.....	5
2.2.2 后台管理员.....	5
三、系统定义.....	6
3.1 系统边界.....	6
3.2 用户视图.....	6
3.2.1 书店会员视图.....	6
3.2.2 管理员视图.....	6
四、需求分析.....	7
4.1 用户需求说明.....	7
4.1.1 数据需求.....	7
4.1.2 事务需求.....	7
4.1.3 用例图.....	8
4.2 系统需求.....	9
4.2.1 网络和共享需求.....	9
4.2.2 性能：.....	9
4.2.3 安全性：.....	9
4.2.4 法律问题：.....	10
4.2.5 运行需求.....	10
4.2.6 其它需求.....	10
五、数据库逻辑设计.....	11
5.1 ER 图.....	11
5.2 数据字典(主要).....	12
5.2.1 书籍实体表.....	12
5.2.2 书店用户.....	13
5.2.3 地址.....	13
5.2.4 订单联系表.....	13
5.2.5 管理员用户.....	14
5.2.6 管理员角色.....	14
5.3 关系表.....	15
六、数据库物理设计.....	15
6.1 索引.....	15

6.2 安全机制.....	16
七、应用程序设计.....	16
7.1 功能模块.....	16
7.2 界面设计.....	17
7.2.1 书店首页.....	17
7.2.2 图书购买页.....	18
7.2.3 个人信息页.....	18
7.2.4 登录注册页.....	20
7.2.5 后端运行情况页面.....	21
7.2.7 系统管理员页面.....	22
7.2.8 图书管理列表.....	23
7.2.9 其它页面.....	24
7.3 事务设计.....	26
7.3.1 登录注册.....	26
7.3.2 图书查询.....	27
7.3.3 购买图书/订单状态更新	28
八、系统的运行测试——购买图书.....	34
8.1 下订单支付.....	34
8.2 发货收货.....	35
8.3 退款	37
九、总结.....	38
9.1 系统优点.....	38
9.2 系统不足.....	38
9.3 心得	38

一、系统开发平台

1.1 题目：小型网上书店

随着时代的发展，信息技术、Internet 技术、数据库技术的不断发展完善，网络进程的加快，传统的购物方式也越来越不能满足人们快节奏的生活需求，使得企业的 IT 部门已经认识到 Internet 的优势，电子商务就是在这样一个背景下发展起来的。伴随着电子商务的不断成熟，电子商务的功能也越来越强大，注册用户可以在网上搜索购买到自己想要的各种商品，初步让人们体会到足不出户便可随心所欲购物的快感。而书籍作为人类精神的食粮更是需要对其特殊化，可以结合电子商务开发出网上书店。

1.2 开发语言：java

用 Java 做设计流程清晰、结构合理，有良好的可扩充性和耦合性。

1.3 开发工具

1.3.1 后端开发

IDEA + SpringBoot

1.3.2 前端开发

IDEA + VUE + Element-UI + Echarts

1.4 数据库 Microsoft SQL Server

SQL Server Enterprise Edition 则是一套完整的数据库和分析产品，具有高度可扩展性和可靠性，对市场的快速反应能力强，可以快速构建各种业务方案，且具有高度的安全性保障。

1.5 操作系统 : Windows 10

二、数据库规划

2.1 任务陈述

网上书店管理系统数据库用以收集、存储书籍信息、人员(用户、系统管理员)信息、购买信息, 售后信息, 图书收藏信息等等, 记录存储各个环节信息的变更, 以便管理、查询、显示、输出, 一个完善的书店管理系统能够更好的专注于对客户的服务, 具有查找方便、可靠性高、存储量大、更新快、寿命长、成本低等优点。

本网上书店管理系统包括书店会员管理, 书籍管理、订单管理、系统用户管理四个模块, 一共设置了4个角色身份, 分别是Administrator角色, LibraryManager角色, BookManager角色, UserManager角色。其中管理员可以拥有全部的权限, 而其它的角色对应着不同的权限, 如同它们的名称所示。

2.2 任务目标

2.2.1 前端用户

- 1.用户或者游客可以浏览查询(分类)书籍的相关信息
- 2.用户可以通过图书详情页来将图书加入购物车,提交图书购买订单以及收藏图书(包括预售图书)
- 3.用户可以申请成为会员
- 4.用户可以修改自己的个人资料(包括收货地址)
5. 用户可以查看并修改自己的个人订单(退款)
- 6.用户可以管理自己的购物车以及自己的收藏图书

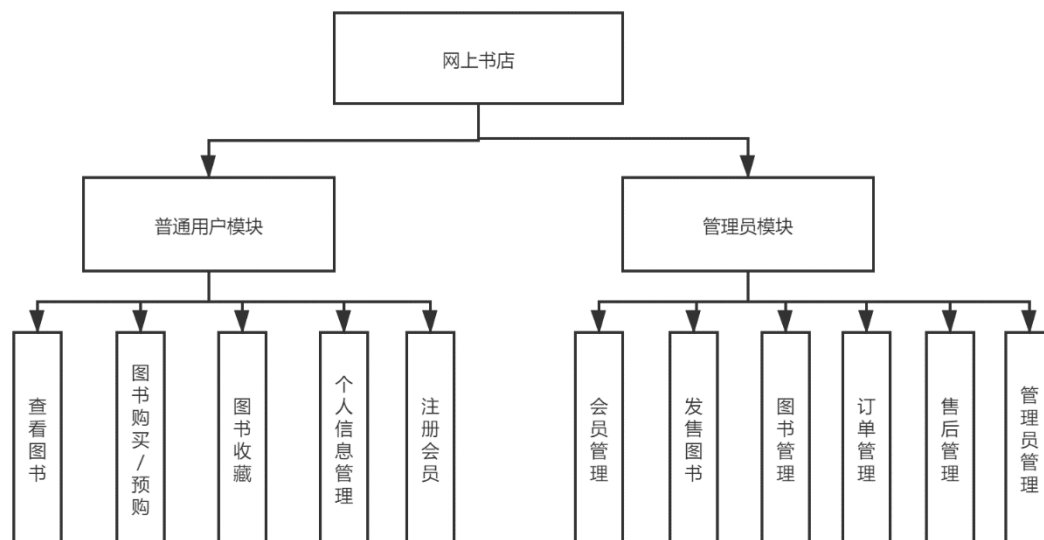
2.2.2 后台管理员

- 1.后台管理员(以下简称员工)可以查询并管理(查改)书店的书籍
- 2.员工可以上架新的图书并且上架已经进行预售的图书
- 3.员工可以对书店的用户进行管理(资料修改)
- 4.员工可以对所有的订单进行管理(包括退款申请的处理)

三、系统定义

3.1 系统边界

系统边界描述数据库系统和企业信息系统的其他部分的接口，是信息系统内部构成元素与外部有联系实体之间的信息关系的描述与分割。它并不需要在它们之间划一条物理边界，而只需要弄清它们之间信息输入与输出的分割。



3.2 用户视图

3.2.1 书店会员视图

- 1) 修改个人信息。
- 2) 按照图书名称或者作者名称在某一分类下模糊查询书籍
- 3) 购买图书
- 4) 个人信息查询与修改
- 5) 售后服务(发货退货)

3.2.2 管理员视图

- 1) 查询并修改相关书籍信息(查询条件与书店会员相匹配)
- 2) 添加，上架，下架，预售书籍
- 3) 书店会员信息管理
- 4) 系统管理员权限管理

- 5) 用户订单管理
- 6) 售后服务处理
- 7) 统计查询

四、需求分析

4.1 用户需求说明

4.1.1 数据需求

- 1、书店会员基本信息记录：
用户名、密码、昵称、邮箱、手机号
- 2、管理员基本信息记录：
用户名、密码、昵称、角色、邮箱、手机号
- 3、书籍基本信息记录：
书籍号、书籍名称、作者、出版社、分类、价格、摘要、出版日期
- 4、图书购买管理：
选购的图书号, 用户号, 购买的数量、支付的状态
- 5、地址信息管理：
地址号、用户号、收件人姓名、地址、收件人电话
- 6、收藏信息管理：
收藏号、用户号、书籍号

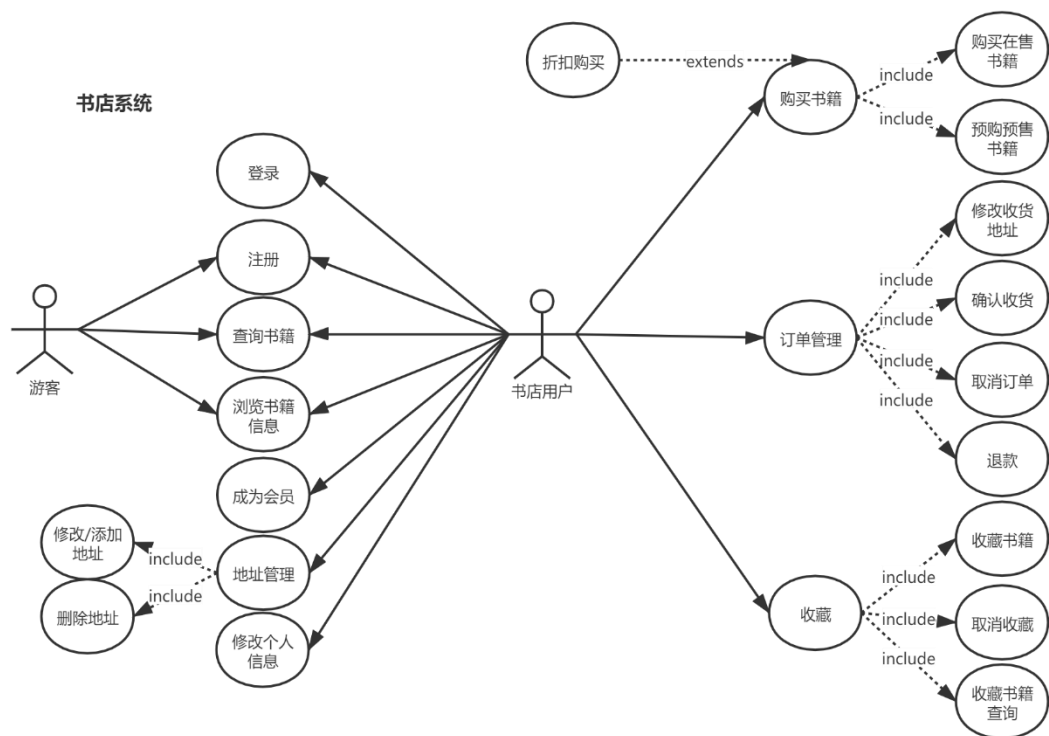
4.1.2 事务需求

- 1) 数据录入
 - a 新用户注册时录入自己的详细情况;
 - b 录入图书的各项信息;
 - c 用户/管理员登录后录入自己的请求信息;
- 2) 数据更新/删除
 - d 更新/删除用户的个人信息;
 - e 更新/删除某些图书的基本信息;
 - f 更新/删除某些图书的购买信息;
 - g 更新/删除某些用户的请求信息;
- 3) 数据查询
 - h 查询用户的个人信息;

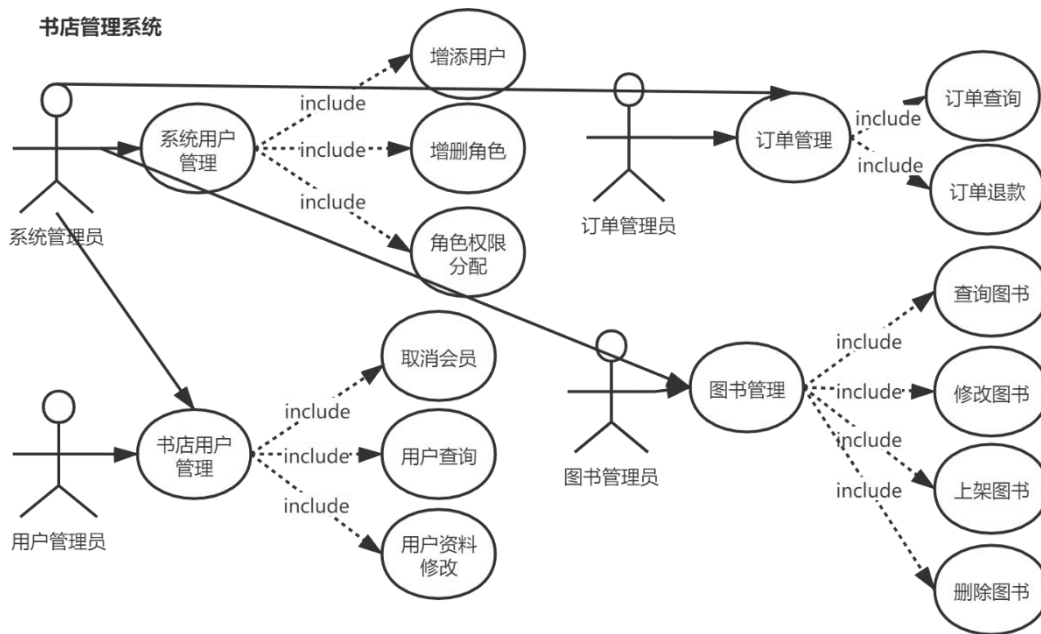
- i 查询图书的基本信息;
- j 查询图书的收藏信息;
- k 查询用户的订单信息;
- l 查询用户的请求信息;

4.1.3 用例图

1)书店会员用例图



2)管理员用例图



4.2 系统需求

4.2.1 网络和共享需求

1. 所有用户必须安全的和总部数据库网络互连；
2. 必须能够支持网上书店至少5名用户同时访问；

4.2.2 性能：

1. 单个记录查询时间少于3 秒
2. 多个记录查询时间少于6 秒
3. 更新/保存记录时间少于2 秒

4.2.3 安全性：

1. 必须有口令保护
2. 每个管理员分配特定的用户视图所应有的访问权限

4.2.4 法律问题：

对管理员和读者信息管理，遵守法律

4.2.5 运行需求

4.2.5.1 用户界面

使用浏览器界面结构，采用导航栏界面方式，尽力带给操作用户便利，对用户友好；对鼠标和键盘单独支持。

4.2.5.2 硬件接口

本软件需要能够互联网的支撑，用户的硬件平台应该能够与互联网连接。

4.2.5.3 软件接口

运行于Windows10操作系统之上，或者其他系统。

4.2.5.4 故障处理

正常使用时不应出错，若运行时遇到不可恢复的系统错误，也必须保证数据库完好无损。

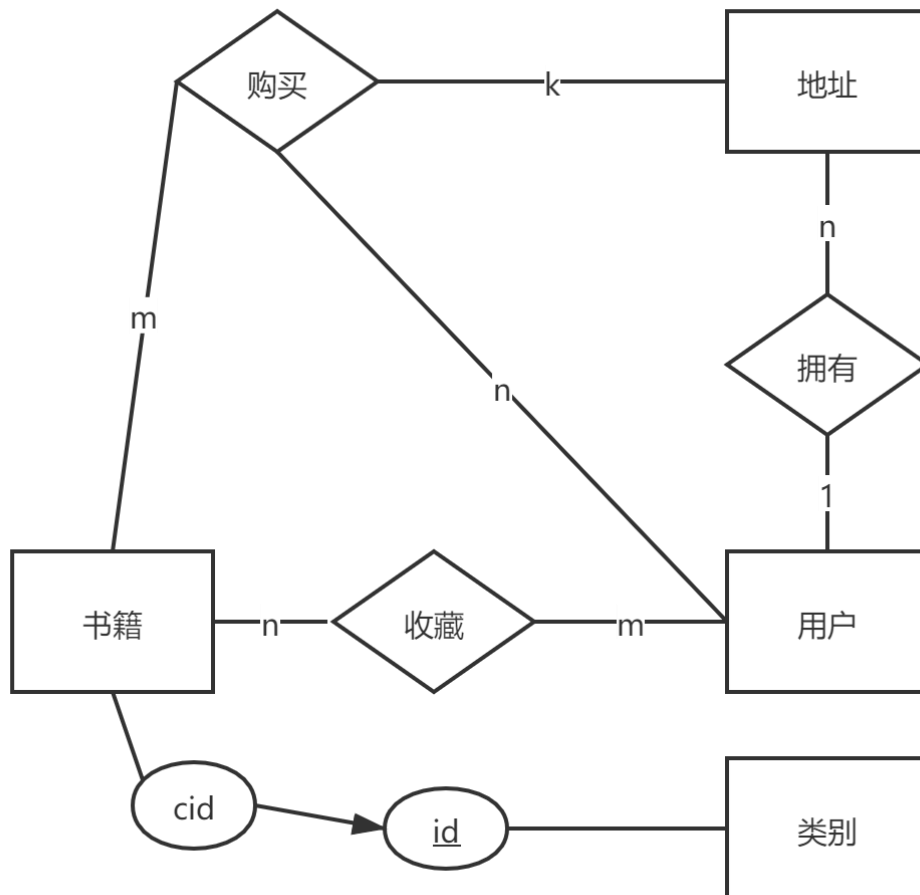
4.2.6 其它需求

1. 系统的功能实现情况：用户可在本系统下实现各种用户要求的功能
2. 系统的安全性：对于系统的重要数据都有密码保护，具有一定的安全性
3. 系统的容错性：用户输错数据都有提示信息，具有较好的容错性能。
4. 系统的封闭性：用户的封闭性较好，用户基本上在提示信息下输数据。

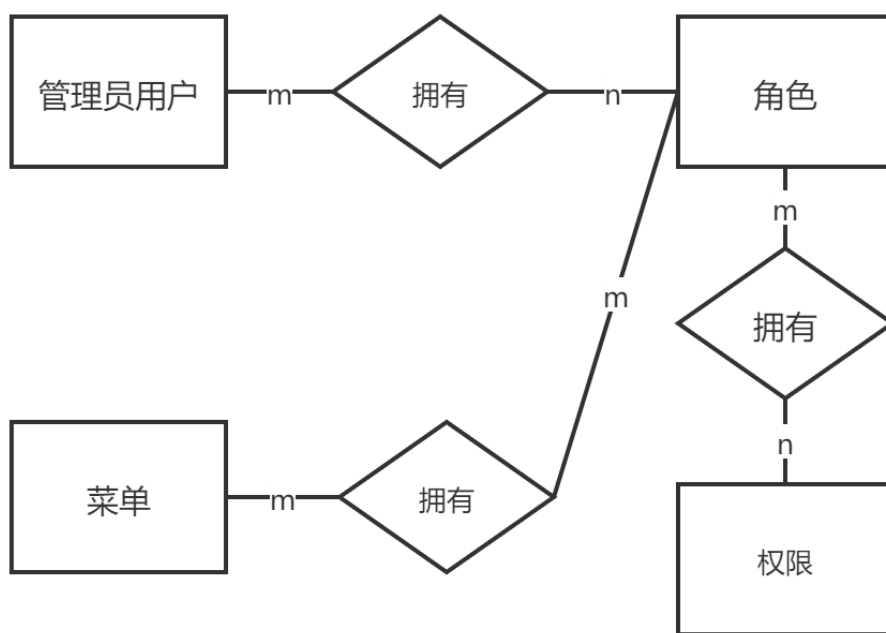
五、数据库逻辑设计

5.1 ER 图

考虑到下面会给出表结构，故 er 图省略属性，首先是书店 er 图：



然后是管理系统 er 图：



5.2 数据字典(主要)

5.2.1 书籍实体表

字段名	数据类型	默认值	是否允许为空
abs	varchar(1000)	NULL	是
author	varchar(255)	空字符串	是
cid	int(11)	NULL	是
cover	varchar(255)	空字符串	是
date	varchar(20)	空字符串	是
id	int(11)	NULL	否
press	varchar(255)	空字符串	是
price	double(10,2) unsigned	50.00	是
sale	int(1)	NULL	是
title	varchar(255)	空字符串	否
unshelve	tinyint(1)	0	是

5.2.2 书店用户

字段名	数据类型	默认值	是否允许为空
email	varchar(255)	NULL	是
enabled	tinyint(1)	NULL	是
id	int(11)	NULL	否
name	varchar(255)	NULL	是
password	varchar(255)	NULL	是
phone	varchar(255)	NULL	是
username	char(255)	NULL	否
vip	int(1)	NULL	是

5.2.3 地址

字段名	数据类型	默认值	是否允许为空
addr	varchar(255)	NULL	否
hostname	varchar(255)	NULL	否
id	int(11) unsig	NULL	否
phone	varchar(255)	NULL	否
uid	int(11)	NULL	是

5.2.4 订单联系表

字段名	数据类型	默认值	是否允许为空	字段说明
aid	int(11)	NULL	否	地址id
bid	int(11)	NULL	否	图书id
delivered	tinyint(1)	NULL	是	是否发货
finish	tinyint(1)	NULL	是	订单结束标志
id	int(11)	NULL	否	订单id
number	int(11)	NULL	否	购买数量
paid_all	tinyint(1)	NULL	是	预购书籍尾款
price	decimal(10,2)	NULL	否	总金额
refund	tinyint(1)	NULL	否	退款状态
successful	tinyint(1)	NULL	否	订单支付状态
uid	int(11)	NULL	否	用户id

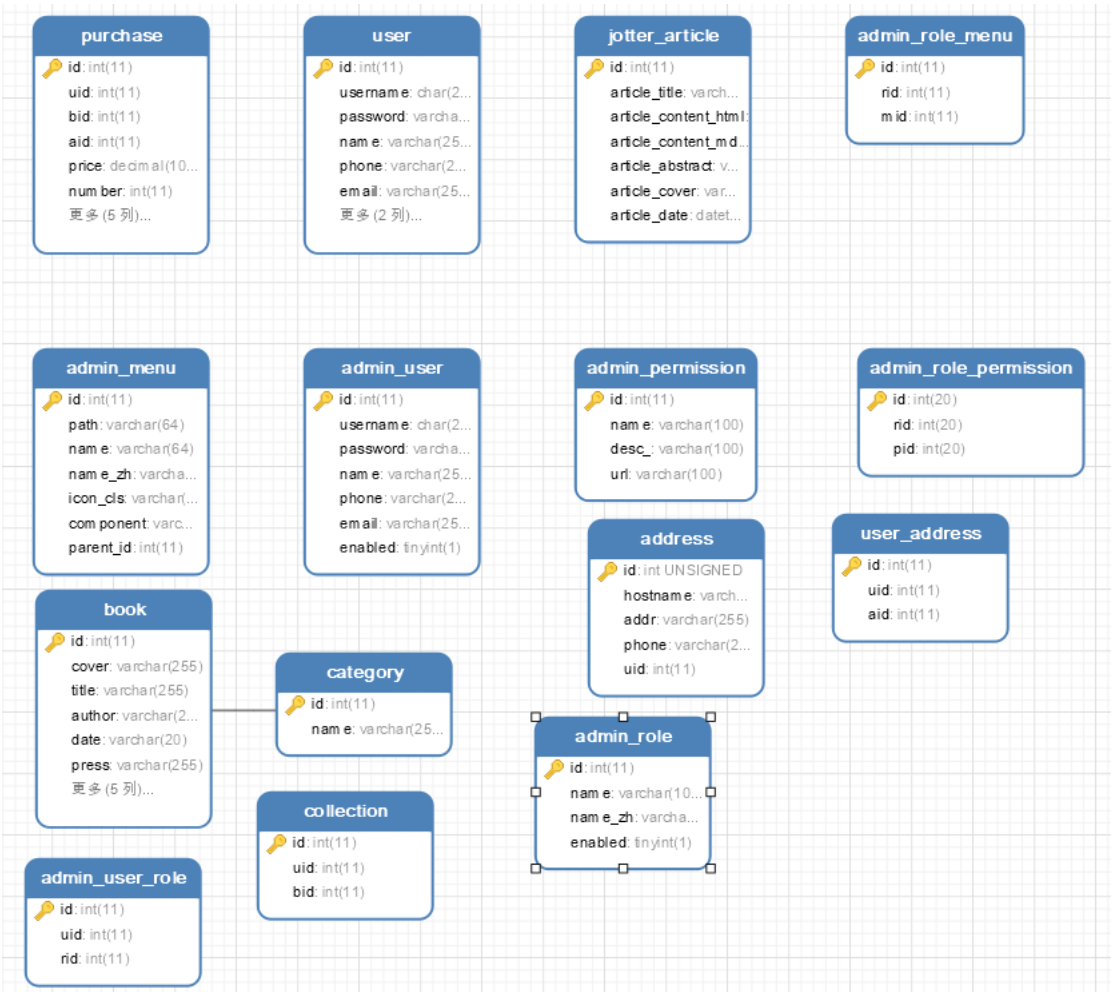
5.2.5 管理员用户

字段名	数据类型	默认值	是否允许为空
email	varchar(255)	NULL	是
enabled	tinyint(1)	NULL	是
id	int(11)	NULL	否
name	varchar(255)	NULL	是
password	varchar(255)	NULL	是
phone	varchar(255)	NULL	是
username	char(255)	NULL	否

5.2.6 管理员角色

字段名	数据类型	默认值	是否允许为空
enabled	tinyint(1)	NULL	是
id	int(11)	NULL	否
name	varchar(100)	NULL	是
name_zh	varchar(100)	NULL	是

5.3 关系表



总的表结构如图所示

六、数据库物理设计

6.1 索引

- 1) 因为添加索引会改善系统性能，所以，根据不同的情况，我引入了不同类型的索引。举例说明如下。
- 2) 在列出书店中所有书时，选择排序的列为book_which_lib，非主键，此索引为聚簇索引。
- 3) 在列出书店中所有会员时，选择排序的列为user_id，主键，此索引为主索引。
- 4) 但在一些表中，如user_address表，没有建索引，因为通过课程学习我们了解

到，不必为小表建立索引。

5)列出图书馆中所有图书，此操作进行较为频繁，因而为其建立二级索引。先按

类别进行分类，再按字典序依book_date 进行排序。

6)在表book 中，因book_title 、book_author为搜索条件，建立索引。

6.2 安全机制

在本书店的信息管理系统中，我建立了两种类型的安全机制，系统安全和数据安全。

1)在系统安全的建设方面，不允许以游客身份访问本系统，所有用户都必须注册并登陆，登录时会验证用户名和密码。只有两者匹配时，才可访问本系统,并且本系统设置了前端拦截器，即不允许非法跳转。

2)在数据安全方面， 数据库对象的访问和使用有严格的控制，其中的某些表只有具有特定权限才可以访问。

七、应用程序设计

7.1 功能模块

该部分在上文中的功能实现以及用例图中都有体现，故不再赘述。

7.2 界面设计

7.2.1 书店首页



书店首页的左边是侧边栏，内容中心是图书详情，鼠标滑过还有缩略图先是，上方有搜索栏，注册会员按钮和登出信息。

7.2.2 图书购买页

图书详情



书名 奢侈与逸乐

作者 马克辛·伯格

出版日期 2019-3

出版社 中国工人出版社

分类 文化

图书摘要

本书探讨了十八世纪英国新式、时尚的消费品的发明、制造和购买。

会员价 ¥33.60 6折优惠

请选择收货地址

-

1

+

提交订单

☆ 收藏

购买页里可以看到与图书相关的基本信息，然后可以选择自己的收货地址以及购买数量，下方的收藏按钮还可以对图书进行收藏。

7.2.3 个人信息页

7.2.3.1 个人订单

图书名	作者	收货地址	单价	购买数量	总金额	订单状态	
与病对话	胡冰霜	wcfad sdu 123	1.53	3	4.59	未收货	查看详情
冒牌人生	陈思安	wcfad sdu 123	1.5	3	4.5	未发货	查看详情
密室中的旅行	[美] 保罗·奥斯特	wcfad sdu 123	73.8	3	221.4	未发货	查看详情

订单表格中显示了用户的购买状态，图书的相关信息等等。

7.2.3.2 收货地址

收货人姓名	收货人地址	联系方式	+ 添加地址	
wcfad	sdu	123	Edit	Delete
ss	山东大学	13956861245	Edit	Delete

收货地址

* 收件人姓名

* 地址

* 联系方式

取消

确定

首页是收货地址的表格，而添加或者编辑地址会弹出一个地址相关的表单。

7.2.3.3 个人信息页

账号admin

昵称

阿超

邮箱

857078179@qq.com

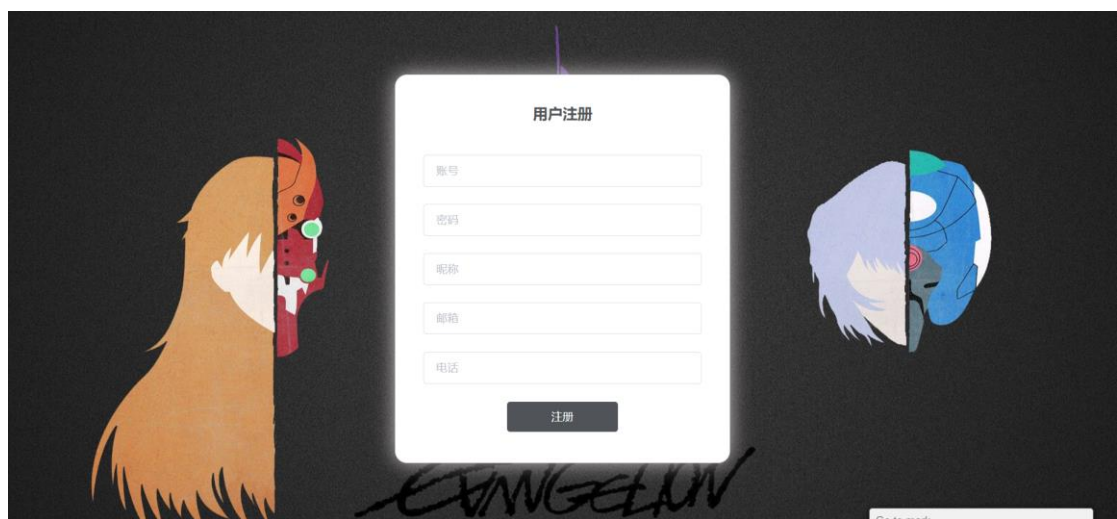
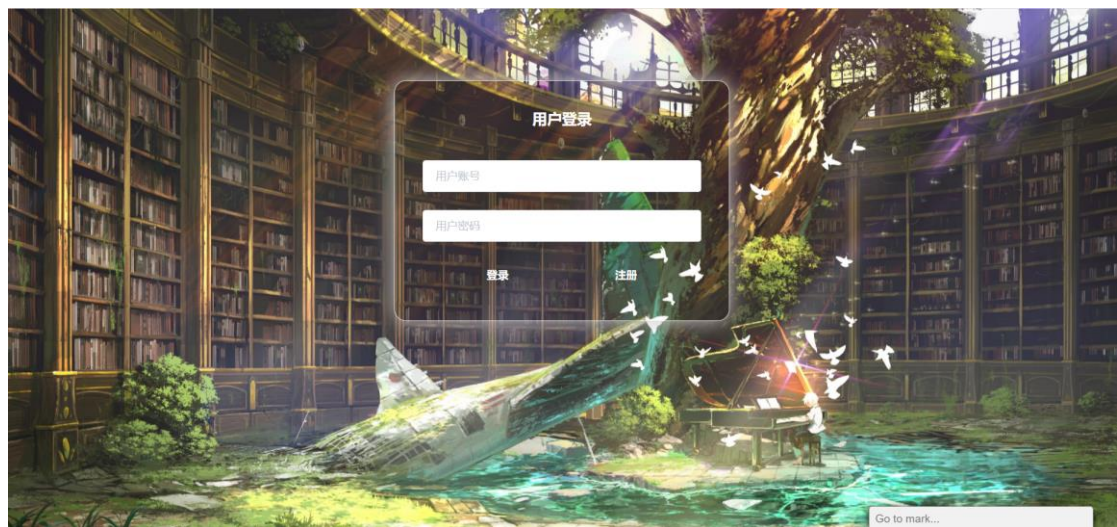
电话

16523569875

编辑个人信息

个人信息页包含了个人的相关信息。

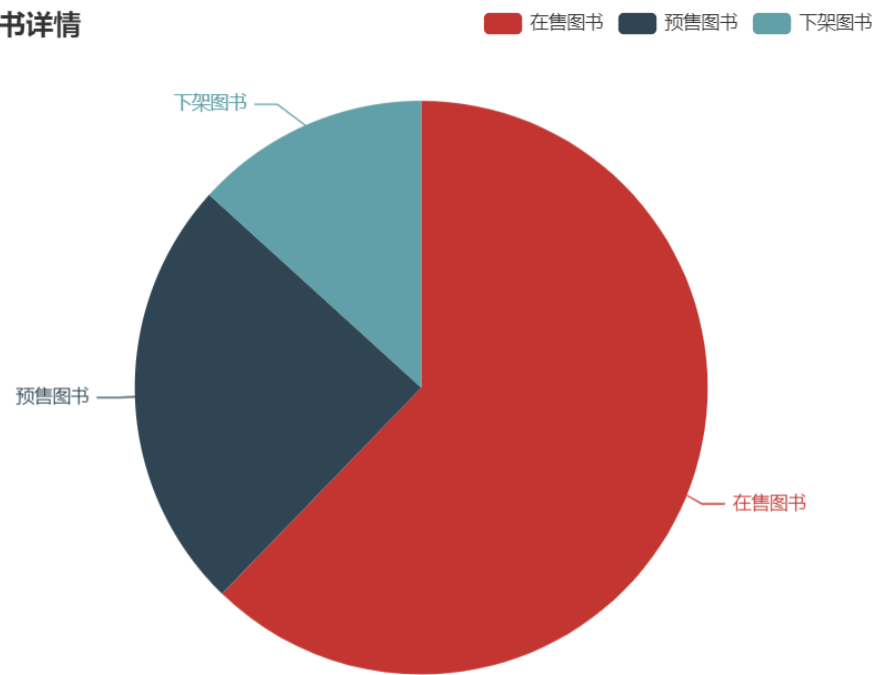
7.2.4 登录注册页



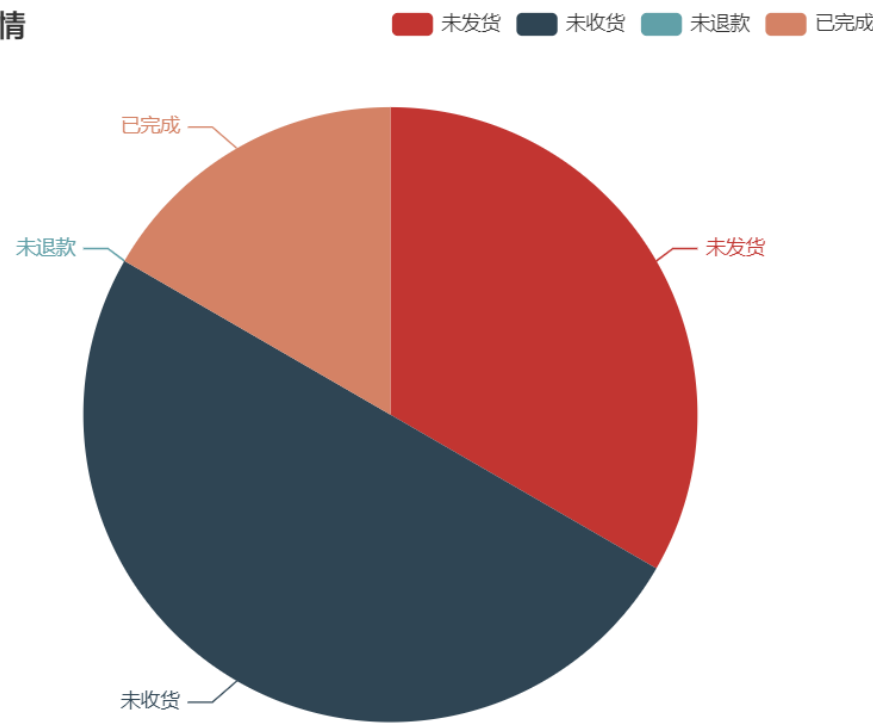
登录注册页中包含了登录注册需要录入的信息，精心设计了页面。

7.2.5 后端运行情况页面

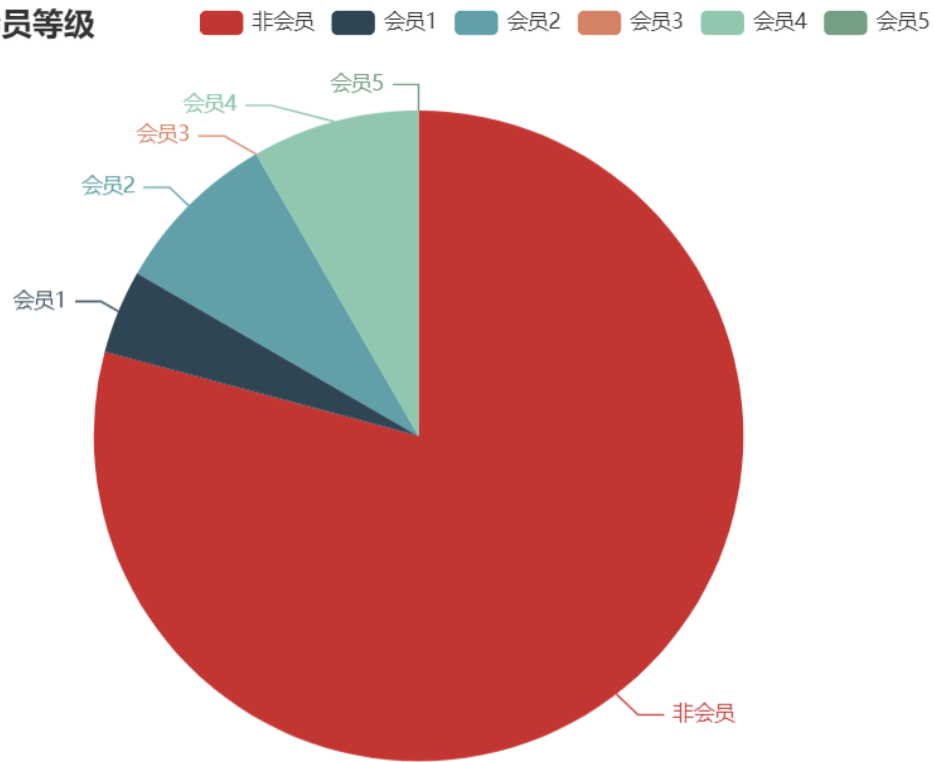
图书详情



订单详情



用户会员等级



用 echarts 做了三个饼状图，分别反映了图书详情，用户详情以及用户会员等级。

7.2.7 系统管理员页面

添加用户

搜索

用户总数目: 19

<input type="checkbox"/>	id	用户名	真实姓名	手机号	邮箱	操作
<input type="checkbox"/>	1	admin	管理员	15662672289	857078179@qq.com	编辑 移除
<input type="checkbox"/>	2	test	测试	123123	123@123.com	编辑 移除
<input type="checkbox"/>	125	order	订单	15662672289	5461@qq.c	编辑 移除
<input type="checkbox"/>	126	test	测试账号	15662672289	2345@w.com	编辑 移除
<input type="checkbox"/>	127	test2	测试账号	123	324@af.c	编辑 移除
<input type="checkbox"/>	128	test3	测试账号	15662672289	fadsf@afa.com	编辑 移除
<input type="checkbox"/>	129	test4	测试账号	15662672289	324@rq.com	编辑 移除
<input type="checkbox"/>	131	test5	测试账号	15662672289	324@rq.com	编辑 移除

该页面是系统管理员的列表，包含了相关的信息以及编辑删除操作，其中编辑页面如下：

用户名

order

昵称

订单

手机号

15662672289

邮箱

5461@qq.c

密码

123

角色分配

☐ 系统管理员

☐ 图书管理员

☐ 访客

☐ 用户管理员

☒ 订单管理员

☐ af

☐ rqew

☐ 2r

☐ qr

☐ 测试角色

编辑页面则包含了用户的相关信息修改以及角色相关的内容，不同角色具有不同的权限。

7.2.8 图书管理列表

管理中心 > 图书管理 > 图书列表

通过作者或者图书名搜索...

请选择分类

搜索

图书总数目: 98

<input type="checkbox"/>	Id	封面	标题	作者	出版日期	出版社	分类	价格	上架	在售	
<input type="checkbox"/>	1		目在人间	余秀华	2019-2-1	湖南文艺出版社	经管	22	<input checked="" type="checkbox"/>	<input type="checkbox"/>	编辑
<input type="checkbox"/>	2		三体	刘慈欣	2008-1	重庆出版社	流行	22	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	编辑
<input type="checkbox"/>	32		叙事的虚构性	海登·怀特	2019-3	南京大学出版社	文化	12	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	编辑
<input type="checkbox"/>	35		圣母	秋吉理香子	2019-3	新星出版社	文学	23	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	编辑
<input type="checkbox"/>	37		奢侈与逸乐	马克辛·伯格	2019-3	中国工人出版社	文化	56	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	编辑
<input type="checkbox"/>	38		忧伤动物	[德]莫妮卡·马龙	2019-4	漓江出版社	文学	45	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	编辑

图书管理列表里面包含了图书的基本信息，上架和调整在售状态操作做以及编辑图书的操作其中编辑图书的表单如下：

修改图书

书名

奢侈与逸乐

作者

马克辛·伯格

出版日期

2019-3

出版社

中国工人出版社

价格

56

封面

https://i.loli.net/2019/04/10/5cada8986e13a.jpg

点击上传

只能上传jpg/png文件，且不超过500kb

简介

本书探讨了十八世纪英国新式、时尚的消费品的发明、制造和购买。

分类







文化

7.2.9 其它页面

其它还有订单管理，用户管理，收藏管理页面，但是但是和上述页面基本相似，就不再过多赘述，下面放图：

<input type="checkbox"/>	id ⬆	用户名	密码	昵称	电话	邮箱	会员等级		
<input type="checkbox"/>	1	admin	123	阿超	16523569875	857078179@qq.com	4	编辑	删除
<input type="checkbox"/>	2	test	123	阿超	12312312312	123@123.com	1	编辑	删除
<input type="checkbox"/>	3	editor	123	编辑	15662672289	54325@qq.com	2	编辑	删除
<input type="checkbox"/>	113	huiye	123	辉夜	15662672289	5566@qq.com	0	编辑	删除
<input type="checkbox"/>	114	antidote	123456	曜	15662672289	8564@qq.com	4	编辑	删除
<input type="checkbox"/>	134	test1	123	测试账号	15662672289	324@qq.com	2	编辑	删除
<input type="checkbox"/>	135	test2	123	测试账号	15662672289	324@qq.com	0	编辑	删除
<input type="checkbox"/>	136	test3	123	测试账号	15662672289	324@qq.com	0	编辑	删除
<input type="checkbox"/>	137	test4	123	测试账号	15662672289	324@qq.com	0	编辑	删除

管理中心 > 用户管理 > 收藏列表

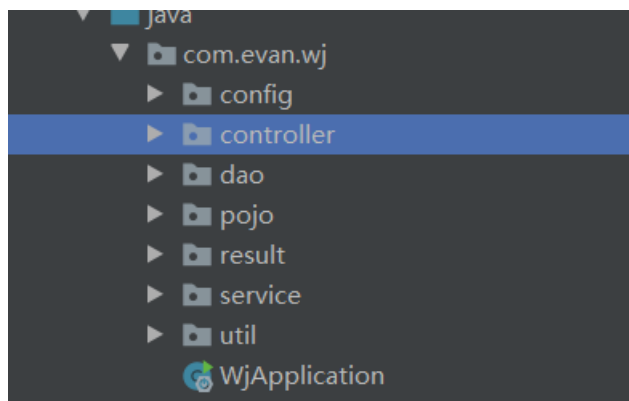
<input type="checkbox"/>	id ⬆	用户id	用户名	封面	标题	
<input type="checkbox"/>	11	113	huiye		C++ Primer 中文版 (第5版)	删除收藏...
<input type="checkbox"/>	12	113	huiye		水浒传 群星闪耀时	删除收藏...
<input type="checkbox"/>	15	1	admin		叙事的虚构性	删除收藏...
<input type="checkbox"/>	20	1	admin		且在人间	删除收藏...
<input type="checkbox"/>	21	1	admin		密室中的旅行	删除收藏...
<input type="checkbox"/>	22	1	admin		上帝笑了99次	删除收藏...

<input type="checkbox"/>	id ⬆	图书id	用户id	图书名	作者	收货地址	单价	购买数量	总金额	订单
<input type="checkbox"/>	106	63	1	与病对话	胡冰霜	wcfad sdu 123	1.53	3	4.59	未收
<input type="checkbox"/>	107	63	114	与病对话	胡冰霜	老爸 1fadsfas 13856258479	1.53	3	4.59	已退
<input type="checkbox"/>	108	60	1	冒牌人生	陈思安	wcfad sdu 123	1.5	3	4.5	未发
<input type="checkbox"/>	109	55	1	密室中的旅行	[美] 保罗·奥斯特	wcfad sdu 123	73.8	3	221.4	未发

id	图书id	用户id	图书名	作者	收货地址	单价	购买数量	总金额	操作
108	60	1	冒牌人生	陈思安	wcfad sdu 123	1.5	3	4.5	发货
109	55	1	密室中的旅行	[美] 保罗·奥斯 特	wcfad sdu 123	73.8	3	221.4	发货

7.3 事务设计

因为后端采用的是 SpringBoot 连接数据库，使用到了 JPA 来进行高级语言的查询，故很多查询的方法已经被封装好了，只要自定义接口即可，其中项目结构如下：



下面给出事务处理的相关代码

7.3.1 登录注册

用户和管理员的登录注册

@CrossOrigin

@PostMapping(value = "api/login")

@ResponseBody

```
public User login(@RequestBody User requestUser) {
    String username = requestUser.getUsername();
    username = HtmlUtils.htmlEscape(username);
    User user = userService.get(username, requestUser.getPassword());
    return user;
}
```

@CrossOrigin

@PostMapping(value = "api/adminLogin")

@ResponseBody

```

public AdminUser adminLogin(@RequestBody AdminUser requestUser) {
    String username = requestUser.getUsername();
    username = HtmlUtils.htmlEscape(username);
    AdminUser user = adminUserService.get(username, requestUser.getPassword());
    return user;
}

@CrossOrigin
@PostMapping(value = "api/userRegister")
@ResponseBody
public int userRegister(@RequestBody User user) {
    return userService.register(user);
}

@CrossOrigin
@PostMapping(value = "api/adminRegister")
@ResponseBody
public void adminRegister(@RequestBody AdminUser user) {
    adminUserService.register(user);
}

```

7.3.2 图书查询

```

@CrossOrigin
@GetMapping("/api/books")
public List<Book> list(@RequestParam("index") int index) throws Exception {
    return bookService.list(index);
}

@CrossOrigin
@PostMapping("/api/books")
public Book addOrUpdate(@RequestBody Book book) throws Exception {
    bookService.addOrUpdate(book);
    return book;
}

@CrossOrigin
@PostMapping("/api/delBook")
public void delete(@RequestBody Book book) throws Exception {
    bookService.deleteById(book.getId());
}

@CrossOrigin
@GetMapping("/api/categories/{cid}/books")

```

```

public List<Book> listByCategory(@PathVariable("cid") int cid, @RequestParam("index") int index)
throws Exception {
    if (0 != cid) {
        return bookService.listByCategory(cid, index);
    } else {
        return list(index);
    }
}

@CrossOrigin
@GetMapping("/api/search")
public List<Book> searchResult(@RequestParam("keywords") String keywords,
                               @RequestParam("index") int index,
                               @RequestParam("cid") int cid) {
    // 关键词为空时查询出所有书籍
    if ("".equals(keywords)) {
        return bookService.list(index);
    } else {
        return bookService.Search(keywords, index, cid);
    }
}

```

7.3.3 购买图书/订单状态更新

```

@CrossOrigin
@PostMapping("/api/submitOrder")
public void addOrder(@RequestBody Order order) throws Exception {
    orderService.addOrUpdate(order);
}

```

```

@CrossOrigin
@GetMapping("/api/getOrder")
public List<Order> getOrder(@RequestParam("uid") int uid) {
    return orderService.getOrderById(uid);
}

```

```

@CrossOrigin
@PostMapping("/api/delOrder")
public void delOrder(@RequestParam("id") int oid) {
    orderService.delOrder(oid);
}

```

```

@CrossOrigin

```

```
@PostMapping("/api/updateOrder")
public void updateByAid(@RequestBody Order order) {
    orderService.updateByAid(order.getId(), order.getAid());
}
```

```
@CrossOrigin
@PostMapping("/api/updatePayment")
public void updatePayment(@RequestParam("id") int id) {
    orderService.updatePayment(id);
}
```

```
@CrossOrigin
@PostMapping("/api/updateRefund")
public void updateRefund(@RequestParam("id") int id) {
    orderService.updateRefund(id);
}
```

```
@CrossOrigin
@PostMapping("/api/finish")
public void updateFinish(@RequestParam("id") int id) {
    orderService.updateFinish(id);
}
```

```
@CrossOrigin
@PostMapping("/api/paidAll")
public void updatePaidAll(@RequestParam("id") int id) {
    orderService.updatePaidAll(id);
}
```

```
@CrossOrigin
@GetMapping("/api/getAllOrders")
public List<Order> getAllOrders() {
    return orderService.list();
}
```

```
@CrossOrigin
@GetMapping("/api/getAllDelivers")
public List<Order> getAllDelivers() {
    return orderService.deliverList();
}
```

```
@CrossOrigin
@GetMapping("/api/getAllRefunds")
public List<Order> getAllRefund() {
    return orderService.refundList();
}
```

```

}
@CrossOrigin
@PostMapping("/api/deliverOrder")
public void deliverOrder(@RequestParam("id") int id) {
    orderService.deliverOrder(id);
}
@CrossOrigin
@PostMapping("/api/refundOrder")
public void refundOrder(@RequestParam("id") int id) {
    orderService.updateFinish(id); //这里操作是一样的
}

```

7.3.4 地址以及信息修改

```

@CrossOrigin
@GetMapping("/api/address")
public List<Address> listByCategory(@RequestParam("uid") int uid) throws Exception {
    return addressService.findAllByUid(uid);
}

```

```

@CrossOrigin
@PostMapping("/api/modifyAddress")
public void addOrUpdate(@RequestBody Address address) throws Exception {
    addressService.addOrUpdate(address);
}

```

```

@CrossOrigin
@PostMapping("/api/delAddress")
public void delAddress(@RequestBody Address address) throws Exception {
    addressService.delAddress(address.getId());
}

```

```

@CrossOrigin
@GetMapping("/api/getAddress")
public Address getAddress(@RequestParam("id") int id) throws Exception {
    return addressService.findAddressById(id);
}

```

```

@CrossOrigin
@PostMapping("/api/becomeVip")
public void becomeVip(@RequestParam("id") int id,@RequestParam("degree") int vip) throws
Exception {
    userService.becomeVip(id,vip);
}

```

```

@CrossOrigin
@GetMapping("/api/userList")
public List<User> userList() {
    return userService.list();
}

```

```

@CrossOrigin
@PostMapping("/api/modifyUserInfo")
public void becomeVip(@RequestBody User user) throws Exception {
    userService.updateUser(user);
}

```

7.3.4 后台的查询管理

```

@CrossOrigin
@GetMapping("/api/menu")
public List<AdminMenu> menu(@RequestParam("uid") int uid) {
    return adminMenuService.getMenusByUid(uid);
}

```

```

@CrossOrigin
@GetMapping("/api/admin/user")
public List<AdminUser> allAdminUsers() {
    return adminUserService.list();
}

```

```

@CrossOrigin
@GetMapping("/api/admin/role")
public List<AdminRole> allRoles() {
    return adminRoleService.findAll();
}

```

//修改管理员用户

```

@CrossOrigin
@PostMapping("/api/admin/modifyUser")
public void updateUser(@RequestBody AdminUser adminUser) {
    adminUserService.updateUser(adminUser);
}

```

```

@CrossOrigin
@GetMapping("/api/admin/role/perm")
public List<AdminPermission> allPerms() {
    return adminPermissionService.list();
}

```

```

@CrossOrigin

```

```
@GetMapping("/api/admin/role/menu")
public List<AdminMenu> listAllMenus() {
    return adminMenuService.getMenusByRid(1);
}
```

```
@CrossOrigin
@PostMapping("/api/admin/modifyRole")
public void updateRole(@RequestBody AdminRole adminRole) {
    adminRoleService.updateRole(adminRole);
}
```

```
@CrossOrigin
@PostMapping("/api/admin/addRole")
public void addRole(@RequestBody AdminRole adminRole) {
    adminRoleService.addRole(adminRole);
}
```

```
@CrossOrigin
@PostMapping("/api/admin/role/del")
public void delRole(@RequestParam("id") int id) {
    adminRoleService.delRole(id);
}
```

```
@CrossOrigin
@PostMapping("/api/admin/modifyMenu")
public void updateRoleMenu(@RequestParam int rid, @RequestBody LinkedHashMap menusIds)
{
    adminRoleMenuService.updateRoleMenu(rid, menusIds);
}
```

```
@CrossOrigin
@PostMapping("/api/admin/deleteUser")
public void deleteUser(@RequestParam("id") int id) {
    adminUserService.delUser(id);
}
```

```
@CrossOrigin
@GetMapping("/api/admin/user/search")
public List<User> searchUser(@RequestParam("keywords") String keywords) {
    return userService.search(keywords);
}
```

```
@CrossOrigin
@GetMapping("/api/admin/book/search")
```



```
public List<Book> searchBook(@RequestParam("keywords") String keywords,
    @RequestParam("cid") int cid) {
    return bookService.adminSearch(keywords, cid);
}
```

```
@CrossOrigin
@GetMapping("/api/admin/system/search")
public List<AdminUser> searchAdminUser(@RequestParam("keywords") String keywords) {
    return adminUserService.search(keywords);
}
```

```
@CrossOrigin
@GetMapping("/api/admin/order/search")
public List<Order> searchOrder(@RequestParam("keywords") int id) {
    return orderService.search(id);
}
```

```
@CrossOrigin
@PostMapping("api/admin/book/unshelve")
public void unshelveBook(@RequestParam("id") int id) {
    bookService.unshelveBook(id);
}
```

```
@CrossOrigin
@PostMapping("api/admin/book/shelve")
public void shelveBook(@RequestParam("id") int id) {
    bookService.shelveBook(id);
}
```

```
@CrossOrigin
@PostMapping("api/admin/book/preSell")
public void preSellBook(@RequestParam("id") int id) {
    bookService.preSellBook(id);
}
```

```
@CrossOrigin
@PostMapping("/api/admin/book/onSale")
public void bookOnSale(@RequestParam("id") int id) {
    bookService.bookOnSale(id);
}
```

```
@CrossOrigin
@GetMapping("/api/admin/getData")
```

```
@ResponseBody
public Statistic getData(){
    return statisticService.getList();
}
```

八、系统的运行测试——购买图书

考虑到信息修改，登录注册以及查询功能很简单，所以就不演示了，在界面设计中可以看出是怎么回事，下面主要就来演示一下图书的购买流程。

8.1 下订单支付

图书详情



书名 奢侈与逸乐

作者 马克辛·伯格

出版日期 2019-3

出版社 中国工人出版社

分类 文化

图书摘要

本书探讨了十八世纪英国新式、时尚的消费品的发明、制造和购买。

会员价 ¥33.60 6折优惠

收货地址 ss 山东大学 13956861245

- 1 +

购买数量

提交订单

☆ 收藏

商品单价

收货地址

订单支付成功!



首先在图书详情页选择图书购买的数量以及收货地址，然后提交订单支付成功以后，会在自己的订单列表里面看到一个相关的订单，其中的状态就是未发货，我们只需要等待后台管理员看见并且发货即可。

8.2 发货收货

首先后台员工进行发货

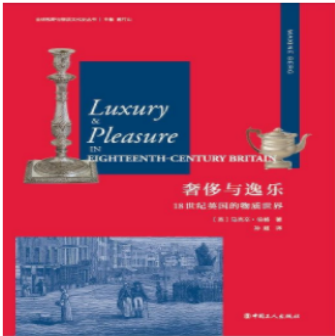


刷新订单列表可以看见订单的状态已经发生了变化



接着在查看详情页里进行收货处理

图书详情



书名 奢侈与逸乐

作者 马克辛·伯格

出版日期 2019-3

出版社 中国工人出版社

分类 文化

图书摘要

本书探讨了十八世纪英国新式、时尚的消费品的发明、制造和购买。

价格: ¥33.6 × 1

收货地址: ss 山东大学 13956861245

需支付总金额: 33.60

收货

申请退款

点击收货即可，订单状态再次发生改变

✓ 已确认收货

逸乐

马克辛·伯格

ss 山东大学 13956861245

33.6

1

33.6

已收货

查看详情

8.3 退款

图书详情



书名 奢侈与逸乐

作者 马克辛·伯格

出版日期 2019-3

出版社 中国工人出版社

分类 文化

图书摘要

本书探讨了十八世纪英国新式、时尚的消费品的发明、制造和购买。

价格: ¥33.6 × 1

收货地址: ss 山东大学 13956861245

需支付总金额: 33.60

已收货

申请退款

点击申请退款得到

✓ 申请退款中!

奢侈与逸乐	马克辛·伯格	ss 山东大学 13956861245	33.6	1	33.6	退款中	查看详情
-------	--------	---------------------	------	---	------	-----	------

此时后台员工可以对退款申请进行同意

图书id	用户id	图书名	作者	收货地址	单价	购买数量	总金额	操作
37	1	奢侈与逸乐	马克辛·伯格	ss 山东大学 13956861245	33.6	1	33.6	退款

点击退款后订单的状态再次发生改变，同时该订单也没有了后续的操作，订单就此结束了。

✔ 退款成功!

奢侈与逸乐

马克辛·伯格

ss 山东大学 13956861245

33.6

1

33.6

已退款

查看详情

九、总结

9.1 系统优点

- 1、界面采用了vue+element-ui的设计，界面美观，容易上手，操作明晰。
- 2、数据库设计内容具体详细，条理清晰，关系明确，能够遵循数据库设计的准则来描述信息关系，可以稳定地为系统提供服务。
- 3、信息提示系统细致完善，对于用户可能发生的错误操作，给予错误信息提示。

9.2 系统不足

- 1.前端代码写的有的太麻烦，不易于修改。
- 2.数据库的设计有些冗余，有的地方可以再简介一些。

9.3 心得

这次课程设计感触颇深，首先是深刻感受到程序的实际应用性，这学期的课程设计的题目都是贴近实际生活的问题，我们就能够很清楚的明白自己写的程序要解决什么样的实际问题，应该解决什么样的实际问题，觉得自己的程序更有实用价值。我设计的这个小型网上书店系统是一个非常简单的模型，实际生活中应用的

网上书店功能十分的强大，我的程序的可以实现中最基本也是最重要的业务，如购买图书，发货，收货，退款等等，其实还有很多比如推荐功能，可以考虑自己学过的机器学习的只是进行简单的嵌入，努力，不断的改正错误，改进自己的程序，有些自己实在是无法解决的问题，则会与其他同学讨论，或上网查询、搜寻资料……在不断的改进过程中，深刻的认识到自己程序的漏洞和不健全性，而且一些vue的语法和函数自己不是很了解，以致有很多错误改了很长时间才能调试正确。通过这次设计，不但让我进一步加深了对知识的巩固，而且很好的锻炼了我的独立思考能力，以及分析问题解决问题的能力。今后凡事都不要着急，要冷静的分析思考，越是急越是无法解决，只有沉着冷静深入思考才能真正的解决问题。只要自己努力凡事都能解决。而且深刻感受到了知识的重要性。平时如果不多积累知识，在编写自己的程序时会手忙脚乱无所适从，

相反如果平时自己扎实学习，知识掌握的都很牢固，到写程序的时候会得心应手，很熟练很快的写出程序，虽然有些毛病，但是如果仔细检查，也会在很短的时间内解决。由此我更加认识到课堂知识的重要性，以及自己多积累，多查资料的重要性。最重要的一点经验就是老师提供了设计提纲，跟着老师的设计思路慢慢完善很重要，总是先建表，再找关系，这是很不正确的，以后设计也要按照老师给的这个思路，要先进行目标明确，任务分析，然后是系统边界，再是需求分析，之后是ER 图，然后才是数据库设计，最后是程序的编写与完善。总之做这个课设需要循序渐进，不能急于求成，否则会手忙脚乱，不知所措。