**原型技术方案设计与实现**

**目录**

[1项目简介 2](#_Toc45484253)

[1.1项目背景 2](#_Toc45484254)

[1.2项目介绍 3](#_Toc45484255)

[2需求分析 3](#_Toc45484256)

[2.1功能性需求 3](#_Toc45484257)

[1首页布局 3](#_Toc45484258)

[2用户登录 4](#_Toc45484259)

[3用户注册 5](#_Toc45484260)

[4图书详情页 6](#_Toc45484261)

[5我的购物车页面 7](#_Toc45484262)

[6订单支付功能 9](#_Toc45484263)

[7订单列表功能 9](#_Toc45484264)

[8用户与权限管理 10](#_Toc45484265)

[9云店铺管理 12](#_Toc45484266)

[10角色权限管理 14](#_Toc45484267)

[11我的店铺 16](#_Toc45484268)

[12个人中心管理 18](#_Toc45484269)

[13个性化 19](#_Toc45484270)

[2.2非功能性需求 20](#_Toc45484271)

[性能需求 20](#_Toc45484272)

[可用性 20](#_Toc45484273)

[可靠性 20](#_Toc45484274)

[易用性 20](#_Toc45484275)

[安全性 20](#_Toc45484276)

[3系统概要设计 21](#_Toc45484277)

[3.1总体架构设计 21](#_Toc45484278)

[3.2功能模块设计 23](#_Toc45484279)

[3.2.1店铺模块 23](#_Toc45484280)

[3.2.2购物车模块 24](#_Toc45484281)

[3.2.3订单模块 24](#_Toc45484282)

[3.2.4主页相关 25](#_Toc45484283)

[3.2.5账户相关 25](#_Toc45484284)

[3.2.6书籍相关 25](#_Toc45484285)

[3.2.7大数据模块 26](#_Toc45484286)

[3.2.8图片相关 26](#_Toc45484287)

[3.3数据库设计 27](#_Toc45484288)

[3.3.1逻辑结构 27](#_Toc45484289)

[3.3.2物理结构 27](#_Toc45484290)

[4系统开发实现 32](#_Toc45484291)

[4.1方案总览 32](#_Toc45484292)

[4.1.1前台要求 32](#_Toc45484293)

[4.1.2后台要求 32](#_Toc45484294)

[4.2前端 34](#_Toc45484295)

[3.2.1用户相关 34](#_Toc45484296)

[3.2.2首页 37](#_Toc45484297)

[3.2.3视图类 39](#_Toc45484298)

[3.2.4记录相关 41](#_Toc45484299)

[3.2.5任务相关 43](#_Toc45484300)

[4.3后端 45](#_Toc45484301)

[4.3.1总览 45](#_Toc45484302)

[4.3.2Controller层 46](#_Toc45484303)

[4.3.3Service层 46](#_Toc45484304)

[4.3.4Dao层 47](#_Toc45484305)

[4.4大数据模块 48](#_Toc45484306)

[4.4.1日志处理与分析 48](#_Toc45484307)

[4.4.2用户聚类 50](#_Toc45484308)

[4.4.3用户个性化推荐 51](#_Toc45484309)

[5系统测试 53](#_Toc45484310)

[5.1用户相关 53](#_Toc45484311)

[5.2店铺&书籍相关 54](#_Toc45484312)

[5.3购物车相关 55](#_Toc45484313)

[5.4订单相关 55](#_Toc45484314)

[5.5主页相关 55](#_Toc45484315)

[5.5其他 56](#_Toc45484316)

[5.6性能测试 56](#_Toc45484317)

# 1项目简介

## 1.1项目背景

​随着互联网行业的井喷式发展，获取信息的方式越来越多，人们从主动获取信息逐渐变成了被动接受信息，信息量也在以几何倍数式爆发增长。同时，垃圾信息越来越多，导致用户获取有价值信息的成本大大增加。由于信息的爆炸式增长，对信息获取的有效性，针对性的需求也就自然出现了，推荐系统应运而生。

以电商为例，当我们打开电商平台，经常会看到“猜你喜欢”，有时候平台所推荐的内容会让觉得 “这不就是我最近浏览过的商品吗？”，但有些时候我们也会惊讶于“这不就是我最近想买的东西吗？它是怎么知道的”。经常有人提问，觉得这像“读心术”。

其实，这是一项大数据技术，是数据挖掘领域的推荐算法。常用的推荐算法有协同过滤和关联规则，当然有些时候也可以将推荐问题转换为预测问题或聚类问题。

俗话说，物以类聚，人以群分。我们的目标就是找到“相似”的人或物品。这里的“相似”并不是指狭义上相似的同一个类别的人或物品，而是指广义上存在某种关联性的两个人或物品。

我们的图书智能推荐系统（有书商城）便是以推荐算法为依托，分析用户历史数据，对用户作智能化个性化书目的推荐。

## 1.2项目介绍

总的来说，这是一个具有个性化推荐功能的Web电商系统。

对用户提供Web前端可视化操作界面，用户可以在我们的有书商城内进行购书等操作；后台负责处理订单等功能；大数据分析模块负责网站的日志分析和用户个性化推荐分析。

# 2需求分析

## 2.1功能性需求

### 1首页布局

#### 功能描述

新书上架

按数据库中图书添加时间和用户偏好进行新书推荐。

图书推荐

根据网站推荐、用户历史搜索、在页面停留时间、曾经购买过的店铺等行为数据为用户进行针对性推荐。

图书top

按照搜索热度或购买数量进行排序，推荐排序前列的图书。

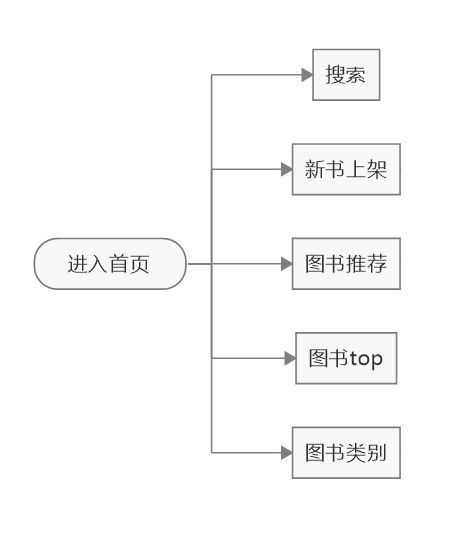
搜索

用户可以输入关键字进行相应的图书或店铺搜索。

按类别查看图书

点击类别后跳转到相应类别的图书页面。

#### 流程图



#### 用例描述

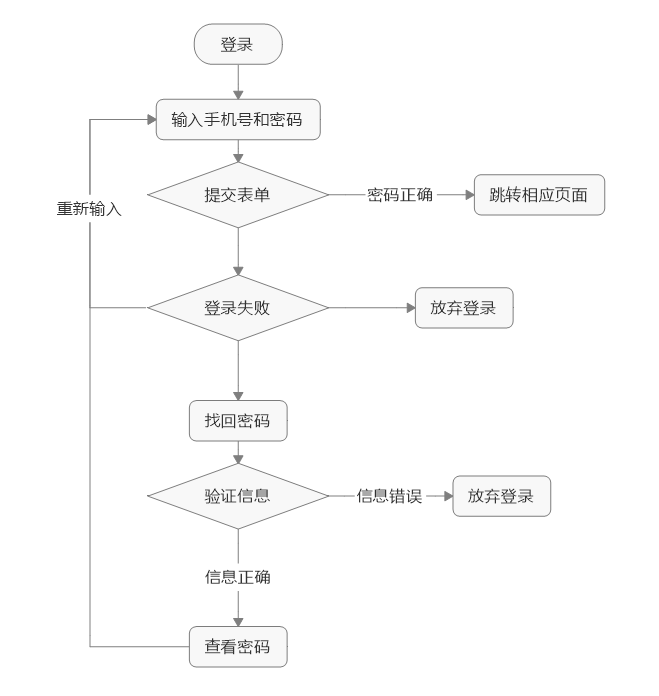
|  |  |
| --- | --- |
| 用例名称 | 浏览首页 |
| 用例id | UC- browse |
| 角色 | 用户、游客 |
| 用例说明 | 用户或游客浏览网站首页 |
| 前置条件 | 打开页面 |
| 基本事件流 | 见流程图 |
| 异常事件流 | 网站加载失败 |

### 2用户登录

#### 功能描述

在拥有账号后，通过注册时使用的手机号和密码进行登录。用户登录后回到首页，商家登录后跳转到商家管理页面。管理员登录后跳转到后台管理界面。忘记密码时可以通过手机号进行密码找回。

#### 流程图



#### 用例描述

|  |  |
| --- | --- |
| 用例名称 | 用户登录 |
| 用例id | UC- login |
| 角色 | 游客 |
| 用例说明 | 游客进行登录 |
| 前置条件 | 游客已是网站用户并持有用户名密码 |
| 基本事件流 | 见流程图 |
| 异常事件流 | 服务器出错，提示用户重新尝试登录 |
| 后置条件 |  |

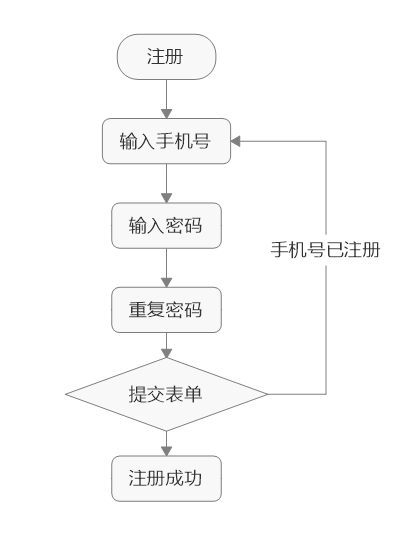
### 3用户注册

#### 功能描述

用户可通过手机号进行注册，在非注册情况下，用户可以进行网页的浏览，注册后方可登录本系统进行图书购买，且会享受更精准的图书推荐。

商家可通过手机号注册店铺账号，拥有店铺账号后可以在平台开店。

#### 流程图



#### 用例描述

|  |  |
| --- | --- |
| 用例名称 | 游客注册 |
| 用例id | UC- register |
| 角色 | 游客 |
| 用例说明 | 游客进行注册 |
| 前置条件 | 游客已打开网站 |
| 基本事件流 | 见流程图 |
| 异常事件流 | 服务器出错，提示用户重新尝试注册 |
| 后置条件 |  |

### 4图书详情页

#### 功能描述

点击某本图书后跳转到图书详情页面，展示该书的名称、作者、出版社、大小、原价、现价、图书图片、运费等信息，并展示图书的详细信息、购书人的评价等，用户可以将图书加入购物车或者分享、收藏图书。

页面侧面根据本图书进行其他图书推荐，图书所在类别的排行榜或者看了本书的人浏览过的其他图书。

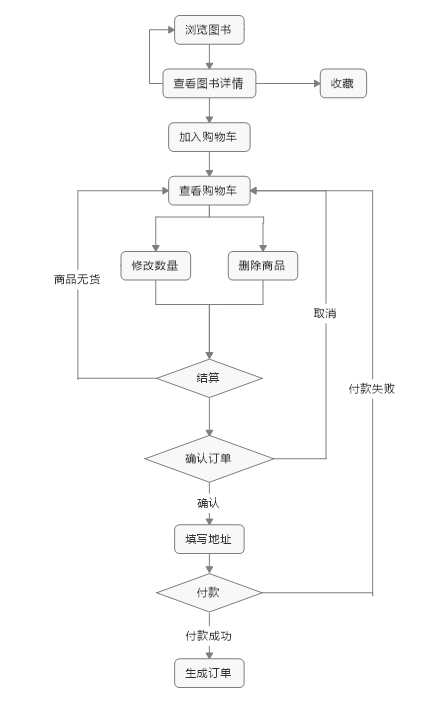
### 5我的购物车页面

#### 功能描述

进入购物车页面后展示已加购的图书列表，按照图书所在的店铺和加购时间进行划分，加购但缺货的图书显示缺货标记。显示选择框、图书名称、图书原价、图书现价、数量、总价和删除按钮。页面底部展示已选择商品数量、总价、优惠金额等信息，并有结算按钮，点击后弹出订单确认页面，叠加邮费，并要求用户填写或选择地址。确认后跳转到支付页面。

页面侧面根据加购的图书的店铺和类型进行其他图书推荐。

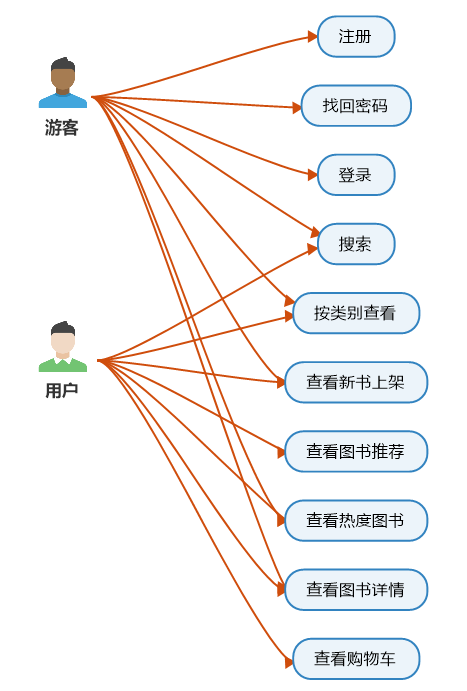
#### 流程图



#### 用例描述

|  |  |
| --- | --- |
| 用例名称 | 用户购物 |
| 用例id | UC-buy |
| 角色 | 用户 |
| 用例说明 | 用户查看图书详情、添加购物车、进行购买 |
| 前置条件 | 用户已登录 |
| 基本事件流 | 见流程图 |
| 异常事件流 | 网络出现故障，提示用户重新尝试 |
| 后置条件 |  |

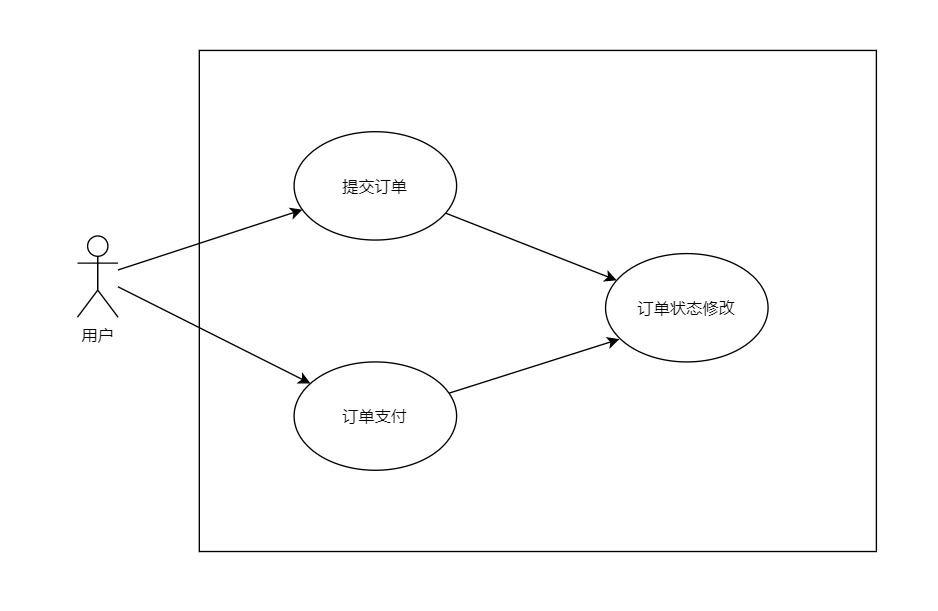
**以上部分用例图如下：**



### 6订单支付功能

在用户的购物车界面可以进行订单的提交，在确认了订单的相关信息之后，系统会跳转到第三方支付平台进行支付，如微信、支付宝等。完成订单之后会修改订单相应的状态。

#### 用例图



#### 用例描述

**提交订单**：用户对订单相关信息进行确认后可以对订单进行提交。

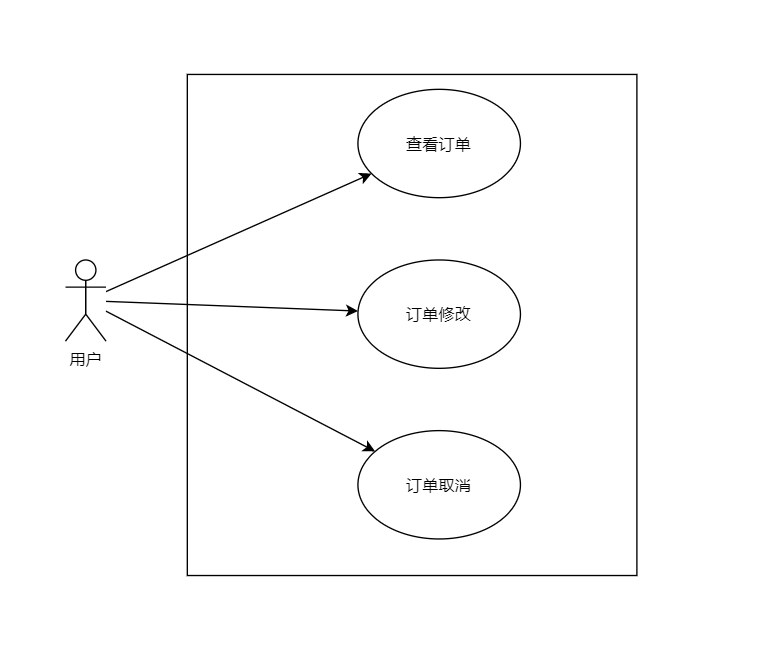
**订单支付**：对于已经提交且未支付的订单，用户可以跳转到第三方支付平台（微信、支付宝等）进行订单支付。

**订单状态修改**：用户在进行提交订单和订单支付的操作之后系统都会对订单的状态进行修改。

### 7订单列表功能

在我的订单页面，系统会列出当前用户的所有订单，在这里用户可以对订单进行查看、修改、支付、删除等操作。

#### 用例图



#### 用例描述

**查看订单**：用户可以对自己的任意订单进行查看。

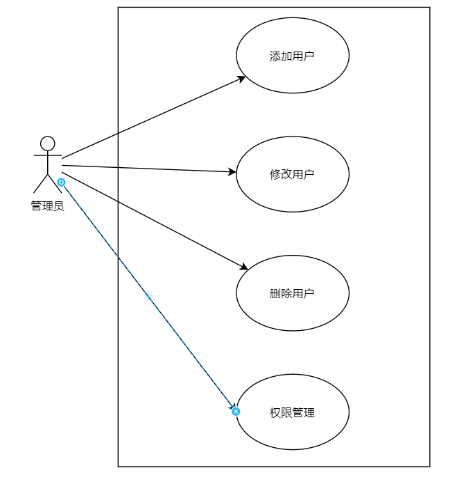
**订单修改**：用户可以对订单进行修改和字符等操作。

**订单取消**：用户可以取消自己的订单。

### 8用户与权限管理

系统管理员可以对用户进行管理，包括对增加用户、删除用户、修改用户信息等。

#### 用例图



#### 用例描述

**添加用户**：管理员可以添加新用户。

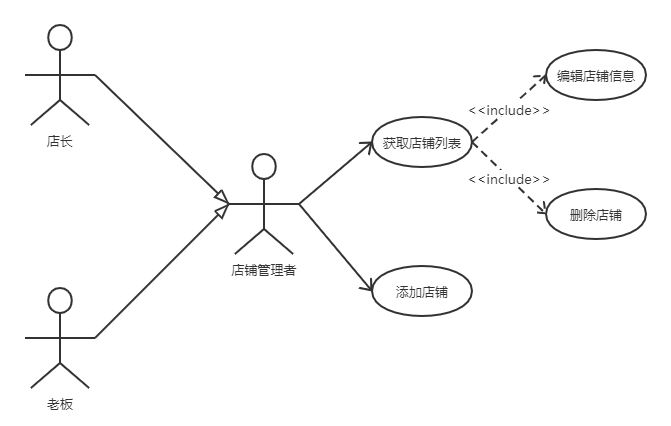
**修改用户**：管理员可以对现有用户进行修改和管理。

**删除用户**：管理员可以删除现有用户。

**权限管理**：管理员可以管理现有用户的权限。

### 9云店铺管理

**用例图：**



#### 云店铺管理 - 云店铺列表及添加

功能描述：角色（老板或店长）可以获取或者添加一间新的云店铺

|  |  |  |  |
| --- | --- | --- | --- |
| ID | UC\_0001 | 名称 | 云店铺列表及添加 |
| 参与者 | 店长或老板 | | |
| 优先级 | 高 | | |
| 触发条件 | 用户需要查看或添加店铺 | | |
| 前置条件 | 用户已登录 | | |
| 后置条件 | 无 | | |
| 正常流程 | 1. 用户正常查看店铺信息并添加店铺 2. 系统检验用户添加店铺是否符合相关规定 3. 系统检验用户所属店铺是否达到其上限 4. 系统授权添加店铺 5. 用户添加店铺成功 | | |
| 扩展流程 | 无 | | |

#### 云店铺管理 - 云店铺编辑与删除

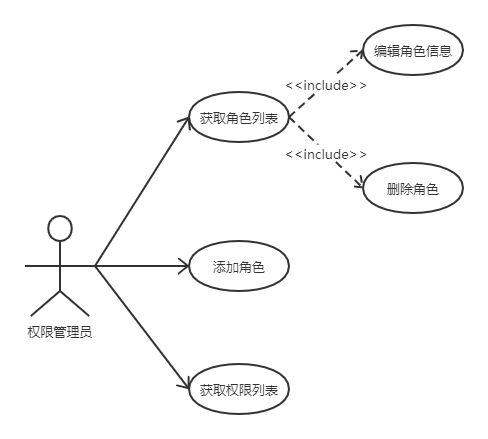
功能描述：角色（老板或店长）编辑一个云店铺的详细信息以及商品信息，同时可以关闭自己不再需要的商铺

|  |  |  |  |
| --- | --- | --- | --- |
| ID | UC\_0002 | 名称 | 云店铺编辑 |
| 参与者 | 店长或老板 | | |
| 优先级 | 中 | | |
| 触发条件 | 用户需要修改店铺信息 | | |
| 前置条件 | 用户已登录，并获取店铺列表 | | |
| 后置条件 | 无 | | |
| 正常流程 | 1. 用户查看某间店铺信息并决定修改店铺信息 2. 系统确认用户修改店铺属于该用户 3. 系统确认用户修改店铺信息符合相关规定 4. 系统授权确认编辑 5. 用户成功编辑店铺信息 | | |
| 扩展流程 | 无 | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | UC\_0003 | 名称 | 云店铺删除 |
| 参与者 | 店长或老板 | | |
| 优先级 | 高 | | |
| 触发条件 | 用户需要删除店铺 | | |
| 前置条件 | 用户已登录，并获取店铺列表 | | |
| 后置条件 | 无 | | |
| 正常流程 | 1. 用户查看某间店铺信息并决定修改店铺信息 2. 系统确认用户删除店铺属于该用户 3. 系统授权用户删除店铺 4. 用户删除店铺成功 | | |
| 扩展流程 | 无 | | |

### 10角色权限管理

**用例图：**



#### 角色权限管理 - 角色列表及编辑与删除

功能描述：权限管理员可以编辑所有用户的信息以及删除注销的用户

|  |  |  |  |
| --- | --- | --- | --- |
| ID | UC\_0003 | 名称 | 角色列表编辑 |
| 参与者 | 权限管理员 | | |
| 优先级 | 高 | | |
| 触发条件 | 用户需要修改角色信息 | | |
| 前置条件 | 用户已登录，并获取角色列表 | | |
| 后置条件 | 无 | | |
| 正常流程 | 1. 管理员查看用户列表并对某个用户信息进行编辑 2. 系统确定用户是否存在 3. 系统授权管理员修改信息 4. 管理员修改信息成功 | | |
| ID | UC\_0004 | 名称 | 角色列表删除 |
| 参与者 | 权限管理员 | | |
| 优先级 | 高 | | |
| 触发条件 | 用户需要注销用户 | | |
| 前置条件 | 用户已登录，并获取角色列表 | | |
| 后置条件 | 无 | | |
| 正常流程 | 1. 管理员查看用户列表并对某个角色进行注销 2. 系统确认用户是否存在 3. 系统授权管理员注销用户 4. 管理员注销用户成功 | | |
| 扩展流程 | 无 | | |

#### 角色权限管理 - 角色添加

功能描述：权限管理员可以自行添加角色，并赋予权限

|  |  |  |  |
| --- | --- | --- | --- |
| ID | UC\_0005 | 名称 | 角色添加 |
| 参与者 | 权限管理员 | | |
| 优先级 | 高 | | |
| 触发条件 | 用户需要修改店铺信息 | | |
| 前置条件 | 用户已登录 | | |
| 后置条件 | 无 | | |
| 正常流程 | 1. 管理员点击角色添加按钮 2. 管理员填写角色信息并提交 3. 系统确认角色信息正确 4. 系统确认角色信息在数据库中无碰撞 5. 系统授权管理员添加 6. 管理员添加用户成功 | | |
| 扩展流程 | 无 | | |

#### 角色权限管理 - 权限列表

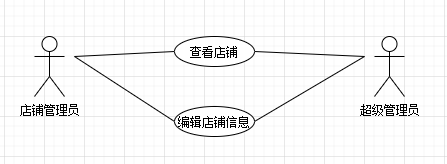
功能描述：权限管理员可以获取各种角色的权限信息

|  |  |  |  |
| --- | --- | --- | --- |
| ID | UC\_0006 | 名称 | 获取权限列表 |
| 参与者 | 权限管理员 | | |
| 优先级 | 高 | | |
| 触发条件 | 管理员查看权限信息 | | |
| 前置条件 | 用户已登录 | | |
| 后置条件 | 无 | | |
| 正常流程 | 1. 管理员查看权限列表 2. 系统检验管理员身份 3. 系统授权管理员查看 4. 管理员查看信息成功 | | |
| 扩展流程 | 无 | | |

### 11我的店铺

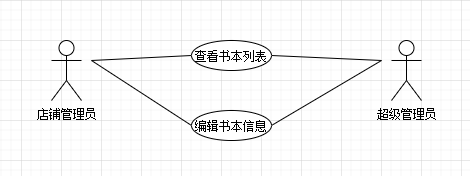
#### 我的店铺-店铺信息查看/编辑

功能描述：有用权限的用户（超级管理员 或 对应店铺的管理员）可以查看并编辑店铺信息。



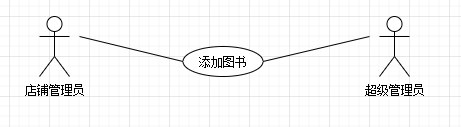
#### 我的店铺-图书列表/编辑

功能描述：有用权限的用户（超级管理员 或 对应店铺的管理员）可以查看店铺内图书列表，编辑对应图书的信息。



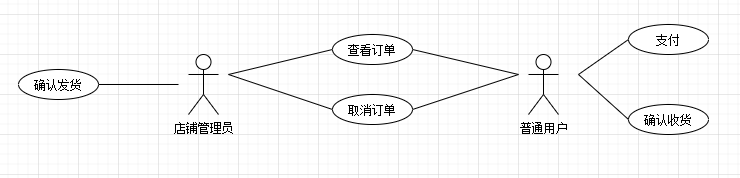
#### 我的店铺-图书添加

功能描述：有用权限的用户（超级管理员 或 对应店铺的管理员）可以对店铺添加新的图书。

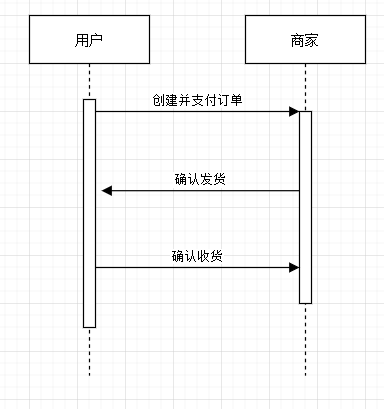


#### 我的店铺-订单管理

功能描述：用户由购买的物品列表生成订单，用户可以查看/支付/取消/确认收货订单，商家可以查看/确认发货/取消订单。



业务流程图：

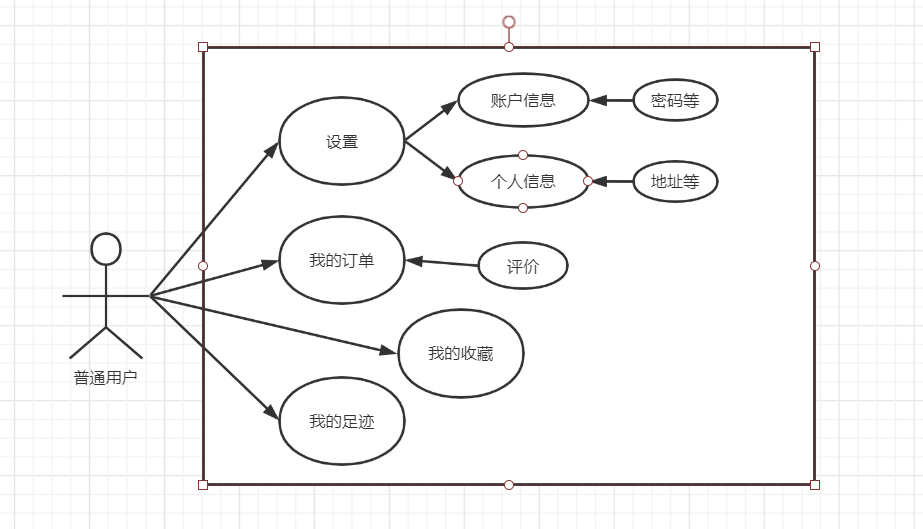


### 12个人中心管理

#### 功能描述

以普通用户的身份登录的时候，用户可以对自己的信息进行修改，其中包括但不限于账号信息、个人信息、收货地址、密码等，并且用户从此界面可以查看我的订单，查看我的关注店铺、收藏等。

#### 用例图



#### 用例描述

设置：用户可以设置自己的账户信息和个人信息，其中特别注意用户密码的修改和私人信息的修改的安全性问题。

我的订单：用户可以查看自己的订单，并对已完成的订单进行评价。

我的收藏&我的足迹：用户可以对店铺或商品进行收藏，

### 13个性化

#### 用户聚类

根据用户历史购买数据，如用户订单数、用户订单平均金额、用户访问次数等对用户进行聚类，此模块可为个人推荐模块提供支持，如同类别的用户进行相似推荐等。

#### 图书推荐

*综合推荐：*在首页将近期销量领先、内容新颖受欢迎或新上架的书目推荐给用户。

*个人推荐：*对新用户（没有历史行为数据或数据很少），采用聚类分析推荐，此外，还会根据个人信息（如职业、地理位置等）寻找用户间相似性，也可根据少量的历史信息采用商品间相似性进行推荐。对老用户：协同过滤算法。

#### 图书词云

商城内所有图书的词云图提供给用户，图书受欢迎程度一目了然。

#### Top20

按图书分类向用户展示销量Top20的图书。

#### 数据分析

数据是一种无形的价值。该模块对网站数据定期进行分析，如每日新增用户数，每日点击量统计，今日最多点击图书分类，网站日志分析（转化率等），分析数据报表对网站运营推广和店铺销售将有极大的作用，相关信息管理员登陆后可以查看。

#### 评价系统

店铺提供的信息真实性的保障，用户可以对已完成的订单进行评价，其他用户可以参考。

## 2.2非功能性需求

### 性能需求

➢ 支持多用户同时操作

➢ 处理的文件数和记录数：1 万

➢ 数据库大小限制：不超过硬盘容量

➢ 单条记录大小限制：不超过数据库大小限制

➢ 存取单条记录时间：不超过 1 秒

➢ 按照索引获取记录时间：不超过 1 秒

### 可用性

➢ 用户操作页面加载失败时，提醒用户重试或重启进行加载

➢ 当客户端由于未知原因死机时，可以通过重启软件来排除故障

➢ 服务器宕机也可以用重启方式恢复。

### 可靠性

➢ 在现有软件和硬件条件和没有出现错误操作情况下，所有模块可以稳定无

故障运行

➢ 可以屏蔽用户错误操作

### 易用性

➢ 用户在简单的操作引导下可以快速上手使用软件

➢ 对用户的操作进行即时的反馈

➢ 执行不可逆操作时对用户进行提示

➢ 支持夜间模式、markdown 语法等用户使用习惯

### 安全性

➢ 用户在记录事件和进行社区分享时，设置为“隐私”的内容对其他用户不可见

➢ 用户多次登录密码错误时对用户进行安全验证，防止偶然或恶意访问

➢ 对数据库进行备份，预防数据丢失

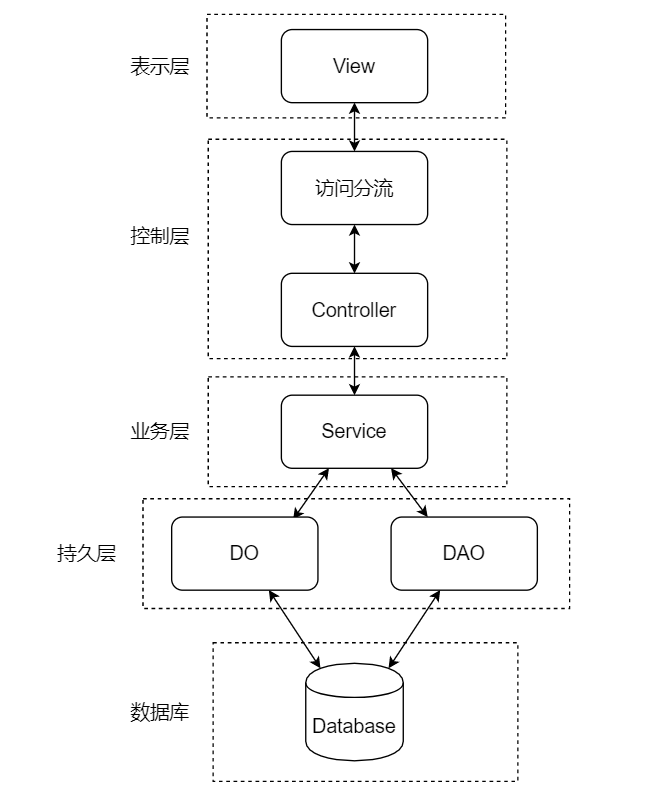
➢ 完善数据库日志管理

# 3系统概要设计

## 3.1总体架构设计

系统根据不同功能划分成不同模块，各模块在进一步划分，各个模块可以独立并行开发，为不同的子模块之间降低耦合性，各模块之间通过接口进行关联调用。衡量模块独立性的标准是耦合性和内聚性。耦合性，也称块间联系，用来度量软件系统结构中各模块之间相互联系的紧密程度。模块之间联系越紧密，耦合性就越强，模块的独立性就越差。内聚性，又称为块内联系，用来衡量模块的功能强度，也就是一个模块内部的各个元素之间彼此结合的紧密程度。如果一个模块内各元素联系得越紧密，则它的内聚性就越高。为软件系统进行模块划分时，要尽量做到高内聚和低耦合，从而提高模块的独立性。

系统总体架构图如下：

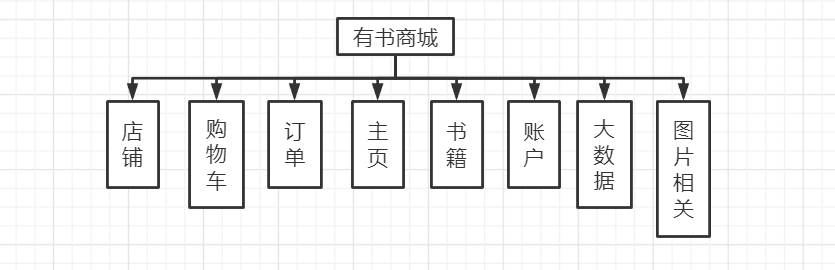


架构说明如下：

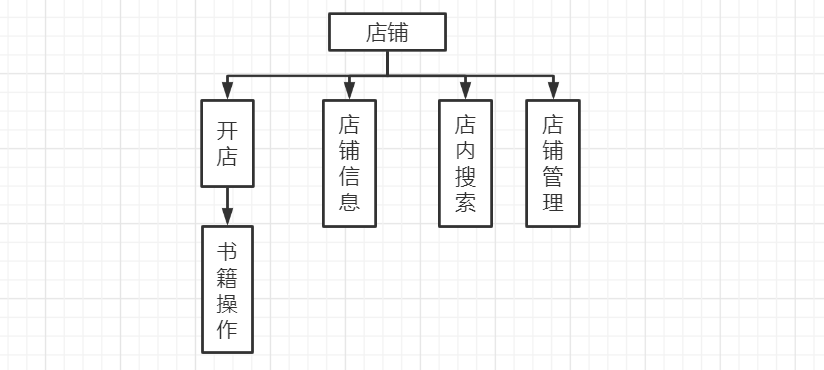
1. 表示层：网页端的表示层主要为使用 Vue.js 制作的界面，用户可以通过浏览器进行访问，访问时表示层向控制层传送参数，将信息传递给控制层。业务逻辑处理后，处理结果将返回给表示层进行展示，实现交互。
2. 控制层：控制层接受来自表示层的信息和请求，将请求分发到业务层不同的模块进行处理，处理后返回结果。同时控制层也承担着对表示层访问请求的分流工作。
3. 业务层：业务层在控制层和持久层中间，调用持久层，又被控制层调用，起一个承上启下的作用，非常关键。系统的结构耦合度较弱，层与层之间的依赖是向下的，上层设计的改变对底层的调用不会有影响。
4. 持久层：持久层采用 Mybatis 框架。Mybatis 提供强大的动态 SQL 功能，将 JDBC 接口封装起来，并通过 mapper 配置文件与数据库相连接，利用 Java 的注解和映射功能进行数据库实例化。Mybatis 可以通过 SqlSessionFactory 对象生成 SqlSession 实例，执行 mapper 中的 SQL 语句。SqlSessionFactory 对象可以通过 XML 配置文件生成或通过 Mybatis 的 Resource 对象构建。
5. 大数据模块也只与Database进行交互，读取相关数据经过处理后写回数据库相应表。

## 3.2功能模块设计

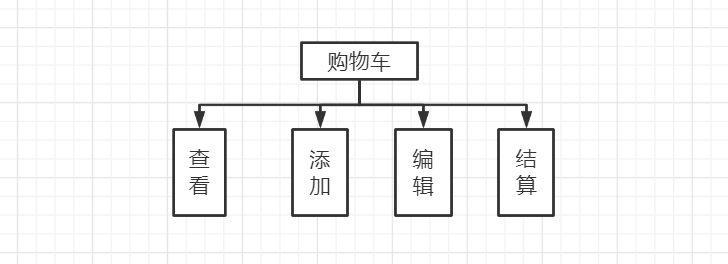
​ 根据需求分析，将系统前台分为八个模块：店铺功能、购物车模块、订单模块、主页相关、书籍相关模块、账户相关、大数据分析相关模块以及图片上传相关模块。功能模块图下图所示。



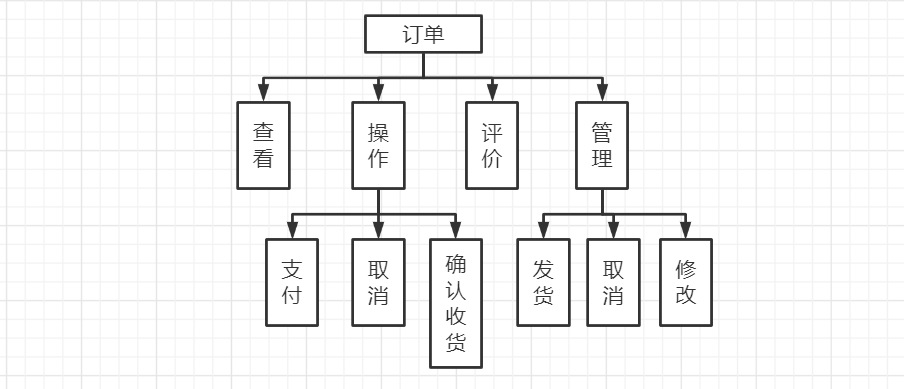
### 3.2.1店铺模块



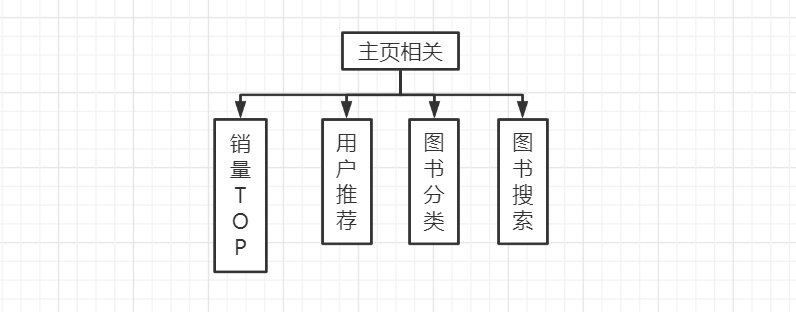
### 3.2.2购物车模块



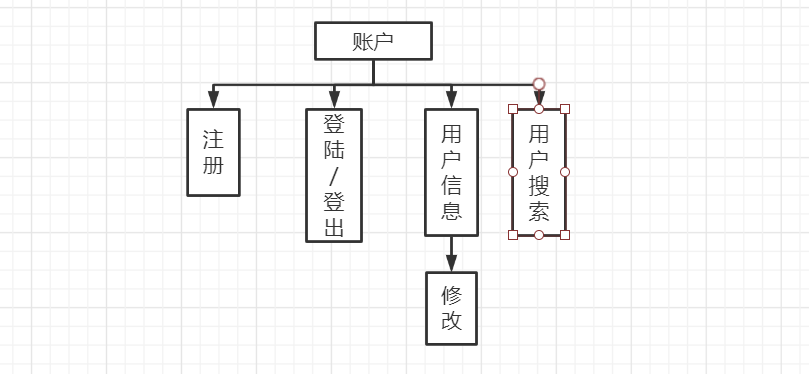
### 3.2.3订单模块



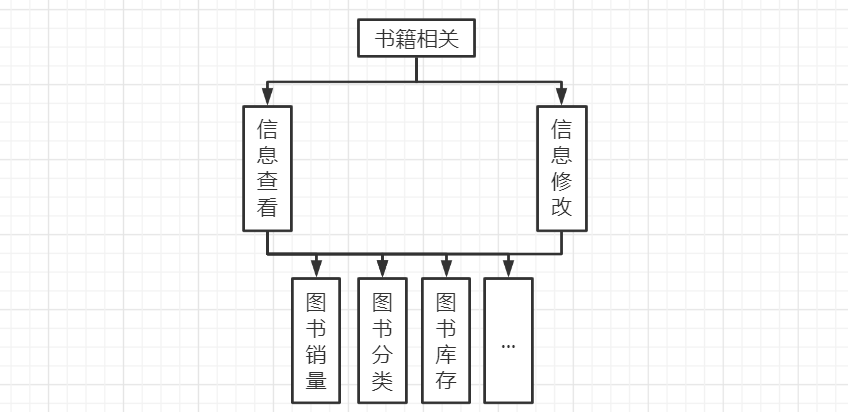
### 3.2.4主页相关



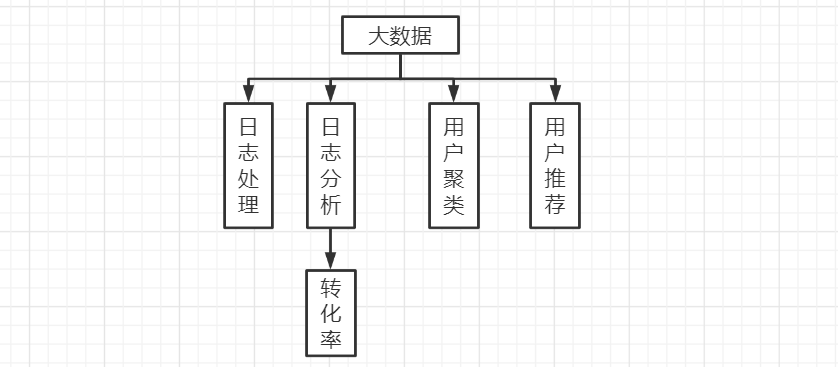
### 3.2.5账户相关



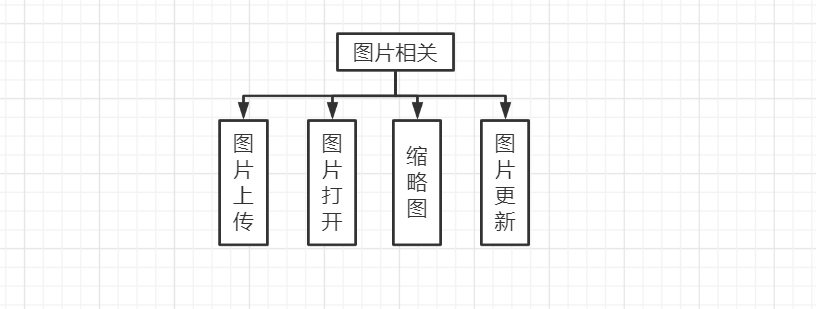
### 3.2.6书籍相关



### 3.2.7大数据模块



### 3.2.8图片相关



## 3.3数据库设计

### 3.3.1逻辑结构

​ 根据系统需求和功能模块设计以及数据库概念模型 E-R 图的设计，共设计了 14 张数据基本表。

​ 后续还会对系统进行优化，开发更多新模块，因此在表的设计中需要保证未来可拓展性。 各模块的关系模型如下：

​ (1) 注册用户表（用户ID，用户账号，密码，盐值）

​ (2) 在线用户表（用户ID，对应token）

​ (3) 用户信息表（用户ID，用户账户，昵称，注册时间，头像URL，VIP费用系数，用户封印标志，账户余额，角色类型）

​ (4) 用户地址表（用户ID，地址ID，标题，内容，姓名，手机号，是否选中）

(5) 书本信息表（书本ID，书名，作者，商店ID，内容，图片URL，销量，库存，价格）

(6) 书本类别表（类别ID，类别名）

(7) 书本评价表（评价ID，书本ID，用户ID，评分，评价内容）

(8) 书本类别标签表（书本ID，类别ID）

(9) 商店信息表（商店ID，商店名，商店老板，商店描述，商店码，商店头像URL，商店评分）

(10) 商店管理员表（用户ID，商店ID）

(11) 购物车信息表（用户ID，商店ID，书本ID，数量）

(12) 订单信息表（评价ID，用户ID，商店ID，种类，订单金额，寄送地址）

(13) 用户偏好表（用户ID，书本ID，偏好值）

(14) 推荐表（用户ID，推荐指数排序，书本ID）

​ 以上数据表中，各表主键均已在图中用金色钥匙🔑符号标出，由于外键的约束会导致复杂的 删除问题，降低效率，所以在数据表的设计中没有设置外键。

### 3.3.2物理结构

​ 各类表具体信息见下。

**注册用户表 user\_reg**

| **列名** | **类型** | **描述** |
| --- | --- | --- |
| uid | INT(11) | 用户ID |
| urn | VARCHAR(255) | 用户账号 |
| pwd | VARCHAR(255) | 用户密码 |
| salt | VARCHAR(255) | 盐值 |

**在线用户表 user\_login**

| **列名** | **类型** | **描述** |
| --- | --- | --- |
| uid | INT(11) | 用户ID |
| token | VARCHAR(255) | 和uid一起做身份验证 |

**用户信息表 user\_info**

| **列名** | **类型** | **描述** |
| --- | --- | --- |
| uid | INT(11) | 用户ID |
| urn | VARCHAR(255) | 用户账号 |
| nickname | VARCHAR(255) | 用户昵称 |
| regtime | BIGINT(20) | 用户注册时间 |
| head | VARCHAR(255) | 用户头像url地址 |
| viprate | INT(11) | Vip优惠率 |
| baned | INT(11) | 是否封禁 |
| money | INT(11) | 用户余额 |
| role | INT(11) | 用户角色 |

**用户地址表 user\_address**

| **列名** | **类型** | **描述** |
| --- | --- | --- |
| uid | INT(11) | 任务ID |
| plid | INT(11) | 地址ID |
| title | text | 标题 |
| content | text | 内容 |
| name | text | 姓名 |
| phone | text | 手机号 |
| selected | INT(11) | 是否选中使用 |

**书本信息表 book\_info**

| **列名** | **类型** | **描述** |
| --- | --- | --- |
| bid | INT(11) | 书本ID |
| name | VARCHAR(255) | 书名 |
| author | VARCHAR(255) | 作者 |
| sid | INT(11) | 商店ID |
| content | VARCHAR(255) | 内容描述 |
| pic | VARCHAR(255) | 图片url地址 |
| sales | VARCHAR(255) | 销量 |
| remain | VARCHAR(255) | 库存 |
| price | INT(11) | 价格 |

**书本类别表 book\_type**

| **列名** | **类型** | **描述** |
| --- | --- | --- |
| tid | INT(11) | 类别ID |
| name | VARCHAR(255) | 类别名称 |

**书本评价表 book\_comment**

| **列名** | **类型** | **描述** |
| --- | --- | --- |
| cid | INT(11) | 评价ID |
| bid | INT(11) | 书本ID |
| uid | INT(11) | 用户ID |
| mark | VARCHAR(255) | 用户评分 |
| content | VARCHAR(255) | 评价内容 |

**书本标签表 book\_tag**

| **列名** | **类型** | **描述** |
| --- | --- | --- |
| bid | INT(11) | 书本ID |
| tid | INT(11) | 类别ID |

**商店信息表 store\_info**

| **列名** | **类型** | **描述** |
| --- | --- | --- |
| sid | INT(11) | 商店ID |
| name | VARCHAR(255) | 商店名称 |
| boss | INT(11) | 商店老板 |
| content | VARCHAR(255) | 商店描述 |
| code | INT(11) | 商店码 |
| head | VARCHAR(255) | 商店头像url地址 |
| mark | double | 商店评分 |

**商店管理员表 store\_manage**

| **列名** | **类型** | **描述** |
| --- | --- | --- |
| uid | INT(11) | 用户ID |
| sid | INT(11) | 商店ID |

**购物车信息表 car\_info**

| **列名** | **类型** | **描述** |
| --- | --- | --- |
| uid | INT(11) | 用户ID |
| sid | INT(11) | 商店ID |
| bid | INT(11) | 书本ID |
| cnt | INT(11) | 购买数量 |

**订单信息表 order\_info**

| **列名** | **类型** | **描述** |
| --- | --- | --- |
| cid | INT(11) | 评价ID |
| uid | INT(11) | 用户ID |
| sid | INT(11) | 商店ID |
| type | INT(11) | 书目种类 |
| money | INT(11) | 订单金额 |
| address | text | 订单寄送地址 |

**用户偏好表 bx\_book\_ratings**

| **列名** | **类型** | **描述** |
| --- | --- | --- |
| userid | INT(11) | 用户ID |
| bookid | INT(11) | 书本ID |
| score | INT(11) | 偏好值[0-10] |

**推荐表 bx\_user\_view**

| **列名** | **类型** | **描述** |
| --- | --- | --- |
| uid | INT(11) | 用户ID |
| place | INT(11) | 推荐指数排序 |
| bid | INT(11) | 书本ID |

# 4系统开发实现

## 4.1方案总览

### 4.1.1前台要求

前台主要要求集中在界面相关，具体要求见UI（界面）设计部分。

### 4.1.2后台要求

后台要求主要集中在接口设计相关，一下展示相关接口定义，详细接口设计请见API设计文档。

**用户**

- /user/login 用户登录

- /user/logout 用户登出

- /user/register 用户注册

- /user/phone 发送手机验证码[不写先]

- /user/getinfo 取本用户基本信息

- /user/setinfo 修改本用户基本信息

**用户管理**

- /user/manage/list 取用户列表

- /user/manage/search 搜索用户

- /user/manage/getinfo 取某用户信息

- /user/manage/setinfo 修改某用户信息

- /user/manage/adduser 创建新用户

**店铺**

- /shop/getinfo 取某商店信息

- /shop/booklist 取某商店书本列表

- /shop/getbookinfo 取书信息

- /shop/search 店铺搜索

- /shop/tags 获取标签文字

**店铺管理**

- /shop/manage/list 取店铺列表

- /shop/manage/getinfo 取某用户信息

- /shop/manage/setinfo 修改某用户信息

- /shop/manage/addbook 添加书

- /shop/manage/delbook 删除书

- /shop/manage/setbookinfo 设置书信息

- /shop/manage/getuser 手机取用户

- /shop/manage/new 开店

**购物车**

- /car/getlist 查看我的购物车

- /car/addbook 添加书

- /car/setlist 编辑列表

- /car/submit 结算一部分

**订单**

- /order/getlist 查看本人所有订单

- /order/getinfo 单个订单信息

- /order/operate 支付/取消/确认收货

- /order/comment 评价

- /order/create 单书直接创建订单

**订单管理**

- /order/manage/getlist 取商铺订单

- /order/manage/operate 确认发货/取消

- /order/manage/setinfo 修改价格/修改产品列表

**图片上传**

- /pic/upload 上传图片

**主页**

- /home/top 销量前n书

- /home/person 给本用户分析推荐的书

- /home/types 图书类别

- /home/books 指定类别的图书列表

- /home/serach 图书搜索

**书籍**

- /book/info 取单书信息

## 4.2前端

### 3.2.1用户相关

#### 3.2.1.1注册页

**界面**

#### 3.2.1.2登录页

**界面**

**业务逻辑流程图**

#### 3.2.1.3个人设置页

**界面**

**业务逻辑流程图**

### 3.2.2首页

**界面**

### 3.2.3视图类

#### 3.2.3.1记录时间线视图

#### 3.2.3.2任务时间线视图

#### 3.2.3.3周视图

#### 3.2.3.4月视图

### 3.2.4记录相关

#### 3.2.4.1记录列表页

#### 3.2.4.2添加记录页

#### 3.2.4.3修改记录页

**界面**

### 3.2.5任务相关

#### 3.2.5.1任务列表页

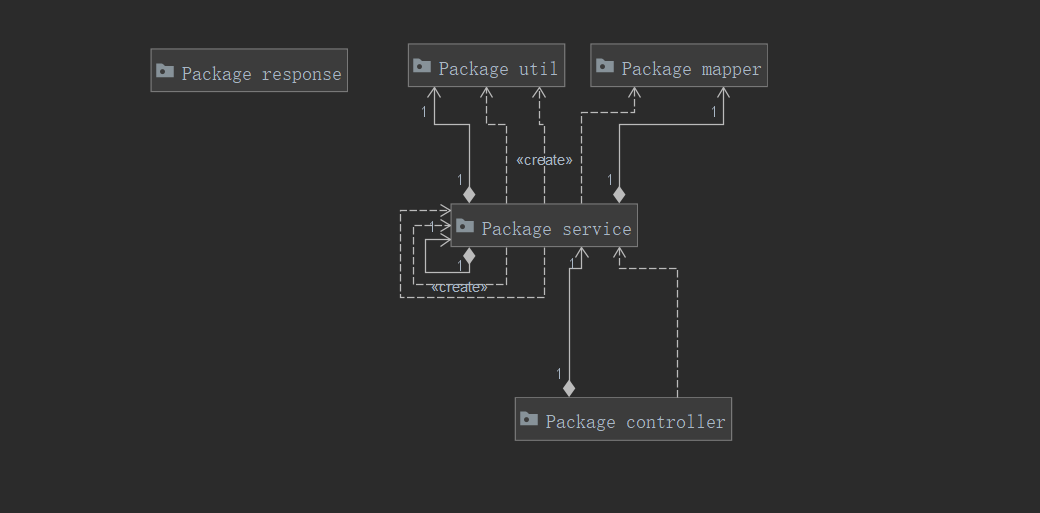
#### 3.2.5.2添加任务页

#### 3.2.5.3修改任务页

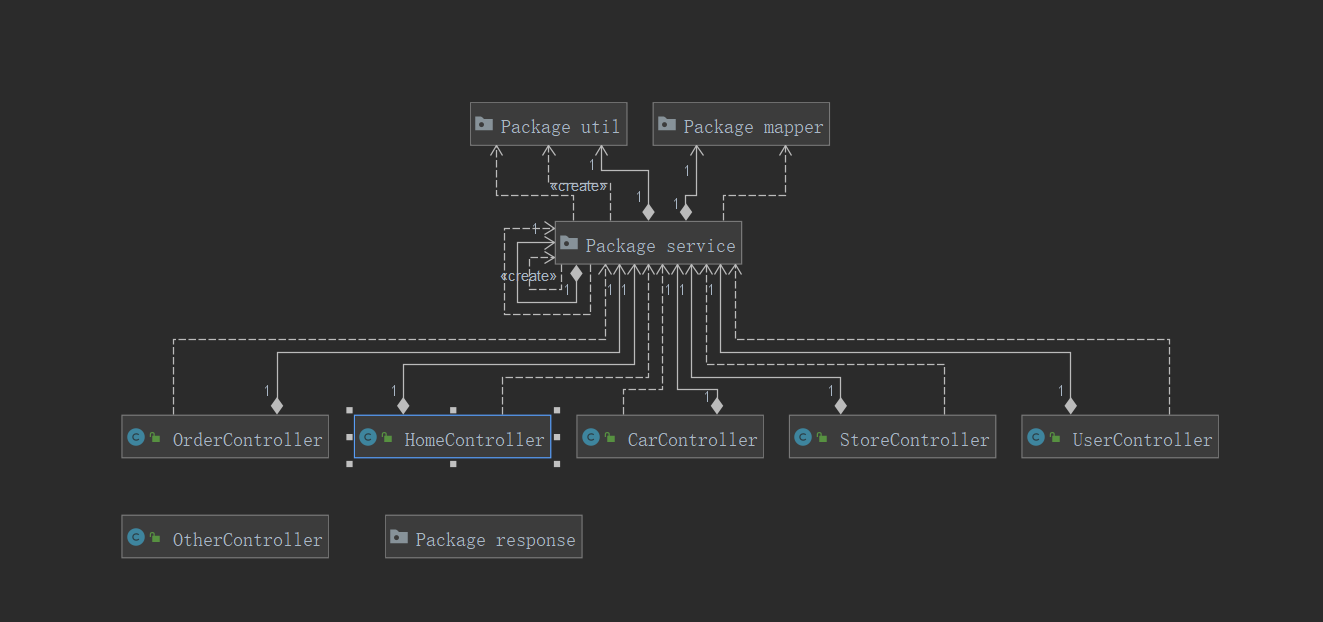
## 4.3后端

### 4.3.1总览

​ 后端采用该Spring Boot框架，MySQL作为数据库存储。包之间的依赖关系见下图。Controller层负责接收请求，Service层对接受的请求进行逻辑处理，并调用Dao层（mapper）进行数据相关的操作。



### 4.3.2Controller层



​ UserController：负责接收来自前端的用户相关的请求。

​ StoreController：负责接收来自前端的商店相关的请求。

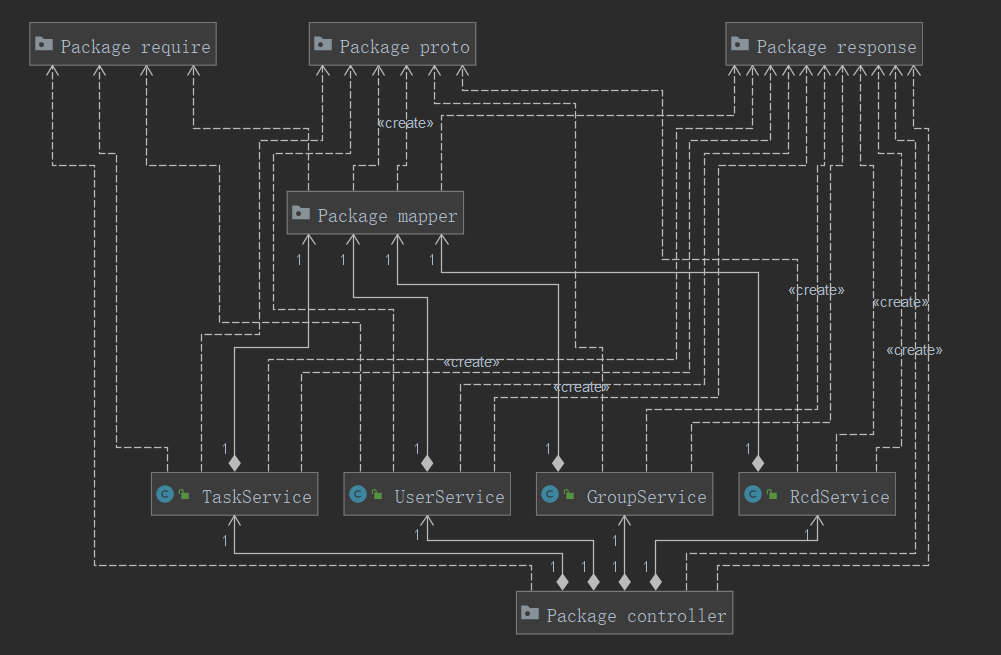
​ CarController：负责接收来自前端的购物车相关的请求。

​ HomeController：负责接收来自前端的主页相关的请求。

OrderController：负责接收来自前端的订单相关的请求。

​ *相关的请求连接请见3.1.2后台要求*

### 4.3.3Service层



​ UserService：负责对来自前端的用户相关的请求进行逻辑处理。

​ StoreService：负责对来自前端的商店相关的请求进行逻辑处理。

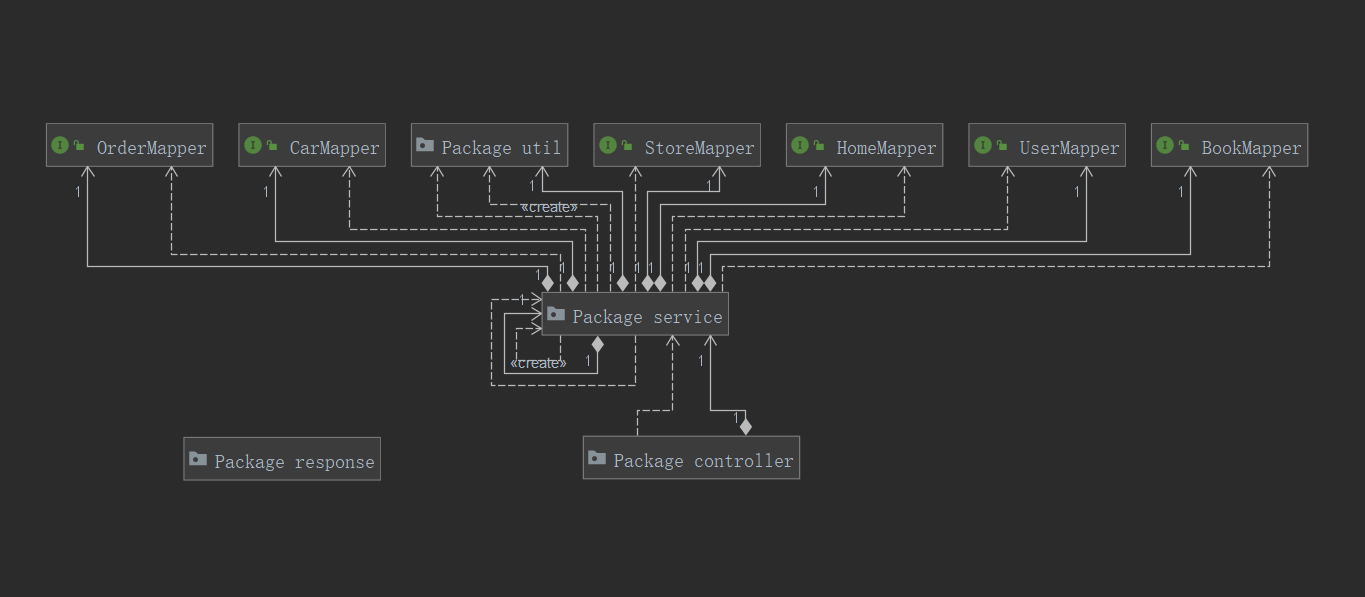
​ CarService：负责对来自前端的购物车相关的请求进行逻辑处理。

​ HomeService：负责对来自前端的主页相关的请求进行逻辑处理。

OrderService：负责对来自前端的订单相关的请求进行逻辑处理。

BookService：负责对来自前端的书籍相关的请求进行逻辑处理。

### 4.3.4Dao层



UserMapper：负责对来自前端的用户相关的请求进行数据库操作。

​ StoreMapper：负责对来自前端的商店相关的请求进行数据库操作。

​ CarMapper：负责对来自前端的购物车相关的请求进行数据库操作。

​ HomeMapper：负责对来自前端的主页相关的请求进行数据库操作。

OrderMapper：负责对来自前端的订单相关的请求进行数据库操作。

BookMapper：负责对来自前端的书籍相关的请求进行数据库操作。

## 4.4大数据模块

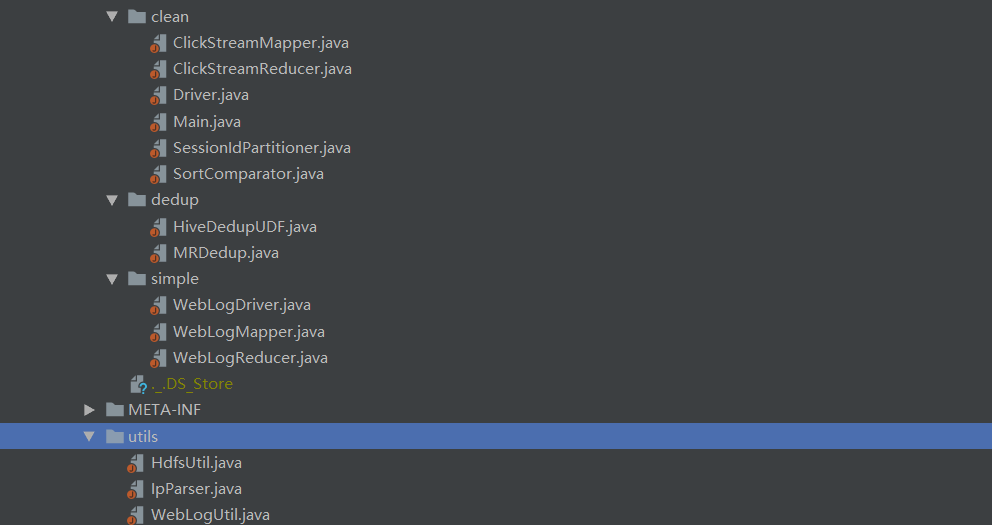
### 4.4.1日志处理与分析

**日志处理标准化**

对网站的日志点击流信息进行解析处理得到如下数据表。

|  |  |
| --- | --- |
| 字段 | 说明 |
| ipaddress | 可以直接从点击流日志中获取 |
| uniqueid | 从点击流日志的Cookie信息中获取 |
| url | 用户访问的链接 |
| sessionid | 由uniqueid和sessionTimes组成生成 |
| sessiontimes | 从点击流日志的Cookie中获取 |
| areaaddress | 通过解析IP地址获取 |
| localaddress | 通过解析IP地址获取 |
| browsertype | 可以直接从点击流日志中获取 |
| operationsys | 可以直接从点击流日志中获取 |
| referurl | 可以直接从点击流日志中获取 |
| receivetime | 可以直接从点击流日志中获取 |
| userid | 从点击流日志的Cookie中获取 |
| csvp | 通过对receiveTime进行排序获取 |

该模块类构成见下图：



 IP地址解析器IpParser类，通过读取ip.dat文件，解析IP的地理位置。

 Mapper类，获取除了csvp字段外的所有字段。

 Partitioner类，控制Shuffle过程，实现按照SessionId分发的规则。

 Comparator类，控制排序过程，实现按照SessionId排序，当SessionId一致时按照receiveTime排序的规则。

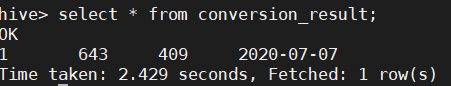
 Reducer类，在经过Shuffle和Sort后，数据已经按照要求排好顺序，这个顺序就是我们需要的csvp，现在只需按照这个顺序对csvp赋值即可。

 Driver类，为Hadoop分发Job，并配置以上类参数。

**转化率分析**



以上为购书转化率分析的三层url漏斗表达式。通过对Hadoop下发MapReduce任务实现转化结果统计，得到如下结果。



转化率约为409/643 = 63.6%。

### 4.4.2用户聚类

**k-means算法**

k-means算法是一种被广泛使用的直接聚类算法。

在k-means算法中，每个聚簇用一个点来代表，这些聚簇用集合C来表示，从中选取k个代表聚簇，被称为聚簇中心。聚类算法通常用相似度的概念，使用某种距离公式对点集进行分组，在k-means算法中默认的相似度标准为欧氏距离。K-means实质是要最小化一个非负的代价函数，即最小化目标是每个点和离它最近的聚簇中心之间的欧氏距离的平方和，这也是k-means的目标函数。

**Canopy算法**

Canopy算法最大的特点是不需要事先指定k值，即聚簇的个数，该算法将聚类过程分为两个过程：第一个过程，Canopy聚类选择简单、计算代价较低的方法计算对象相似性，将相似的对象放在一个子集中，这个子集叫做canopy。第二个过程，根据第一步生成的canopy，再进行一次聚类，这次选取的聚类算法在计算相似性方面会较第一步复杂、精确。

我们很容易想到，Canopy算法的第二步可以用k-menas来完成，而第一步则负责得出k值和初始的聚簇中心。

**Apache Mahout机器学习库**

Apache Mahout 是 Apache Software Foundation (ASF) 旗下的一个开源项目，提供一些可扩展的机器学习领域经典算法的实现，旨在帮助开发人员更加方便快捷地创建智能应用程序，并且，在 Mahout 的最近版本中还加入了对 Apache Hadoop 的支持，使这些算法可以更高效的运行在云计算环境中。Mahout 中提供了常用的多种聚类算法：Canopy算法、k-means算法、模糊k-means算法等。

本项目分四步实现用户聚类：

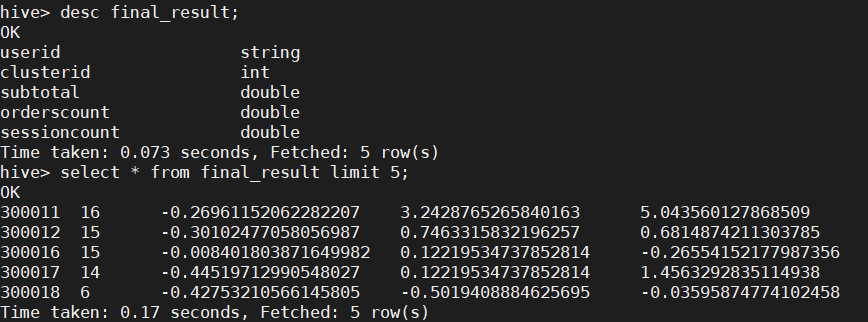
（1） 将输入文件序列化

（2） 生成canopy

（3） 利用生成的canopy执行k-means算法聚类

（4） 将聚类结果输出并解析存入数据库

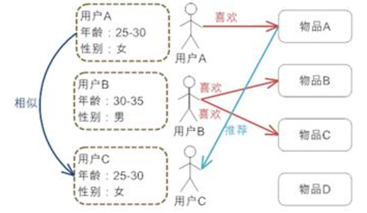
最终结果分为17个cluster，部分示例如下。



### 4.4.3用户个性化推荐

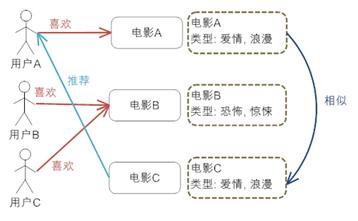
**基于人口统计学的推荐**

基于人口统计学的推荐机制（Demographic-based Recommendation）是一种最易于实现的推荐方法，它只是简单的根据系统用户的基本信息发现用户的相关程度，然后将相似用户喜爱的其他物品推荐给当前用户。



**基于内容的推荐**

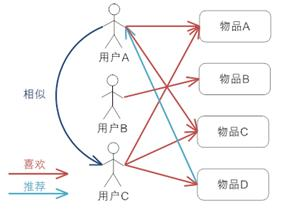
基于内容的推荐是在推荐引擎出现之初应用最为广泛的推荐机制，它的核心思想是根据推荐物品或内容的元数据，发现物品或者内容的相关性，然后基于用户以往的喜好记录，推荐给用户相似的物品。



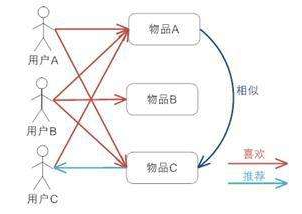
**基于协同过滤的推荐**

根据用户对物品或者信息的偏好，发现物品或者内容本身的相关性，或者是发现用户的相关性，然后再基于这些关联性进行推荐。基于协同过滤的推荐可以分为三个子类：基于用户的推荐（User-based Recommendation），基于项目的推荐（Item-based Recommendation）和基于模型的推荐（Model-based Recommendation）

***基于用户的推荐（根据用户历史偏好推断）***



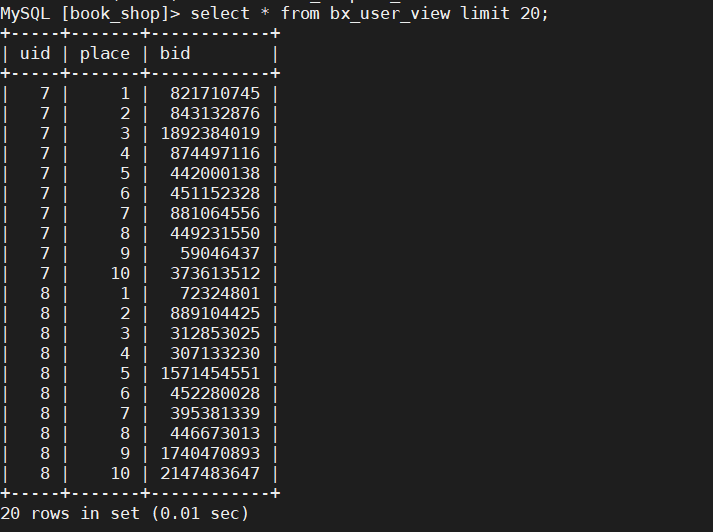
***基于项目的推荐（根据物品本身属性特征信息推断）***



***基于模型的推荐***

基于模型的协同过滤推荐就是基于样本的用户喜好信息，训练一个推荐模型，然后根据实时的用户喜好的信息进行预测，计算推荐。

本系统使用Mahout分布式程序实现基于物品的协同过滤，运行于Hadoop分布式环境。调用RecommenderJob实现商品间相似性分析，从而为用户推荐图书。以下为结果示例。



# 5系统测试

​ 在完成了系统功能开发之后，为了在产品上线前消除潜在的bug，我们需要根据系统测试理论对系统的相关功能和性能指标进行测试。系统测试是将已经集成好的软件系统，作为计算机系统的一个元素，与计算机硬件、某些支持软件、数据和人员等其他系统元素结合在一起，在实际运行环境下，对计算机系统进行一系列的集成测试和确认测试。

​ 因为时间与成本原因，本项目只融合了黑盒与白盒测试技术，对系统进行了简单必要的测试。预期结果为通过的测试用例均已通过。以下列出了部分功能测试样例（以预期结果不通过为主，即可能会导致系统出错的样例）。性能测试均已通过。

## 5.1用户相关

注册

| **测试用例** | **预期结果** | **结果** |
| --- | --- | --- |
| 无效手机号 | 注册失败 | 通过 |
| 错误验证码 | 注册失败 | 通过 |

登陆

| **测试用例** | **预期结果** | **结果** |
| --- | --- | --- |
| 正确账号，错误密码 | 登陆失败 | 通过 |
| 错误账号，正确密码 | 登陆失败 | 通过 |
| 错误账号，错误密码 | 登陆失败 | 通过 |

登出

| **测试用例** | **预期结果** | **结果** |
| --- | --- | --- |
| 错误uid(不在线) | 登出失败 | 通过 |
| 错误token | 登出失败 | 通过 |

## 5.2店铺&书籍相关

开店

| **测试用例** | **预期结果** | **结果** |
| --- | --- | --- |
| 过长标题 | 操作失败 | 通过 |
| 乱码店铺描述内容 | 操作失败 | 通过 |
| 过大店铺图片 | 操作失败 | 通过 |

店铺管理

| **测试用例** | **预期结果** | **结果** |
| --- | --- | --- |
| 非管理员操作 | 操作失败 | 通过 |
| 违规尝试修改评分 | 修改失败 | 通过 |

书籍信息

| **测试用例** | **预期结果** | **结果** |
| --- | --- | --- |
| 内容乱码 | 上架失败 | 通过 |
| 无库存上架 | 上架失败 | 通过 |
| 无效图片url | 上架失败 | 通过 |

## 5.3购物车相关

内容

| **测试用例** | **预期结果** | **结果** |
| --- | --- | --- |
| 超过数量限制 | 添加失败 | 通过 |
| 编辑内容 | 成功修改 | 通过 |
| 删除选项 | 成功删除 | 通过 |
| 商品失效 | 购买等操作失败 | 通过 |

## 5.4订单相关

内容

| **测试用例** | **预期结果** | **结果** |
| --- | --- | --- |
| 无地址 | 创建失败 | 通过 |
| 余额不足 | 支付失败 | 通过 |
| 支付返回 | 支付失败，可重新支付 | 通过 |

## 5.5主页相关

内容

| **测试用例** | **预期结果** | **结果** |
| --- | --- | --- |
| 分类按钮点击跳转 | 跳转正常 | 通过 |
| 推荐界面 | 有效推荐 | 通过 |

## 5.5其他

搜索

| **测试用例** | **预期结果** | **结果** |
| --- | --- | --- |
| 过长搜索关键字 | 无法搜索 | 通过 |
| 无效关键字 | 无法搜索 | 通过 |

同步

| **测试用例** | **预期结果** | **结果** |
| --- | --- | --- |
| 过低版本 | 同步失败 | 通过 |
| 断网同步 | 同步失败 | 通过 |
| 不同设备用户同步 | 同步成功 | 通过 |

可用性+易用性

| **测试用例** | **预期结果** | **结果** |
| --- | --- | --- |
| 不同时间段阶梯可用测试 | 维护时间段外可用 | 通过 |
| 页面点击 | 点击响应迅速正常 | 通过 |
| 页面跳转 | 跳转正常不卡顿 | 通过 |
| 页面滑动 | 顺畅不卡顿 | 通过 |
| 布局美观性 | 美观 | 通过 |

## 5.6性能测试

| **测试描述** | **期望标准** | **测试结果** |
| --- | --- | --- |
| 用户同时操作 | 支持 | 通过 |
| 处理的文件数和记录数 | 10000 | 未知 |
| 数据库大小限制 | 不超过硬盘容量 | 通过 |
| 单条记录大小限制 | 不超过数据库大小限制 | 通过 |
| 存取单条记录时间限制 | 1s | 通过 |
| 按照索引获取记录时间限制 | 1s | 通过 |