毕业设计（论文）



|  |  |
| --- | --- |
| 题 目： | 基于Java EE的跨平台电子商城系统的设计与实现 |
| 学 生： | 陈杰 |
| 指导老师： | 陈丽晖 |
| 院 别： | 软件学院 |
| 专 业： | 信息管理与信息系统 |
| 班 级： | 1502 |
| 学 号： | 3158111226 |

2019年1月

福建工程学院本科毕业设计（论文）作者承诺保证书

本人郑重承诺：本篇毕业设计（论文）的内容真实、可靠。如果存在弄虚作假、抄袭的情况，本人愿承担全部责任。

学生签名：陈杰

2019 年 1 月 13 日

福建工程学院本科毕业设计（论文）指导教师承诺保证书

本人郑重承诺：我已按有关规定对本篇毕业设计(论文)的选题与内容进行了指导和审核，且提交的毕业设计（论文）终稿与上传至“大学生论文管理系统”检测的电子文档相吻合，未发现弄虚作假、抄袭的现象，本人愿承担指导教师的相关责任。

指导教师签名：

2019 年 1 月 日

**基于Java EE的跨平台电子商城系统的设计与实现**

## 摘要：近几年随着手机的普遍和兴起，在线购物已成为当今互联网模式下的一个很大的购物模式，彻底改变了曾今人们的消费理念。在线购物与传统购物相比较优点有很多，例如：商品种类齐全；无需任何经营和管理费用，因此网上购物在价格上与实体店铺内的商品也会相对便宜；实体店购物每当要寻找自己想买的东西时总是问东问西，难以找到自己想要买的东西，在线购物提供的搜索狗能能够快速准确的定位到商品；传统购物交易使用现金或者银行卡，跨银行和大笔现金交易变得极不便捷，而在线购物结合现代的支付技术，很大的提高了资金流动促进消费。本课题基于JaveEE技术的电子商城系统的设计和实现，结合现代主流的互联网技术，打造出一款重体验，设计美观且操作贴合人日常使用的产品，提高卖家的销售量和利润，降低管理和库存等成本。系统主要分为以下模块：商品浏览与搜索，分类模块，购物车模块，订单模块，支付模块，评论模块，管理员模块，后台商户数据分析模块等，前端采用目前主流框架Vue，React和一些其他插件库，后端使用Java EE搭建业务，配合Mysql存储数据，实现一整套从web浏览器端到移动APP端以及数据分析服务一体化电子商城系统体系。

**关键词**：JavaEE;跨平台；Native App；React；React-Native

Based on Java EE cross-platform electronic mall system design and implementation

**Abstract**:In recent years as cell phones and rise generally, online shopping has become the Internet mode of a big shopping mode, completely changed once people's consumption concept. Online shopping advantages compared with traditional shopping has a lot of, such as: product variety is complete; Without any operation and management cost, so the online shopping is the price and the real stores within the commodity will be relatively cheap; Entity shop shopping every time to find what they want to buy always ask east asked west, it is difficult to find what they want to buy, online shopping provide search dogs can fast accurate localization to the goods; Traditional shopping transaction to use cash or bank card, bank and cash transaction becomes extremely convenient, and online shopping combining modern payment technology, greatly improved the capital flow to promote consumption. This topic based on JaveEE electronic mall system design and implementation of the technology, combined with modern mainstream of Internet technology, to create an experience, beautiful design and joint operation of daily use products, improve the seller's sales and profits, reduce the management and inventory cost.Mainly divided into the following modules: browse and search goods, classification module, a shopping cart module, order module, payment module, comments module, administrator module, the background merchant data analysis module, etc, this system at the front desk USES the current mainstream framework Vue, React and some other plug-in library, the back-end using Java EE building business, cooperate with Mysql to store data, implement a set of from the web browser side to mobile APP and service integrated e-commerce system and data analysis system.

**Keywords:**JavaEE;cross-platform;Native App;React;React-Native

目录

**[1概述](#_Toc20191_WPSOffice_Level1)** **[2](#_Toc20191_WPSOffice_Level1)**

**[1.1课题背景和意义](#_Toc4766_WPSOffice_Level1)** **[2](#_Toc4766_WPSOffice_Level1)**

**[1.2本课题的研究内容](#_Toc21549_WPSOffice_Level1)** **[2](#_Toc21549_WPSOffice_Level1)**

**[1.3跨平台技术和电子商务现状](#_Toc5003_WPSOffice_Level1)** **[2](#_Toc5003_WPSOffice_Level1)**

**[2可行性分析](#_Toc24322_WPSOffice_Level1)** **[3](#_Toc24322_WPSOffice_Level1)**

**[2.1技术可行性](#_Toc24278_WPSOffice_Level1)** **[3](#_Toc24278_WPSOffice_Level1)**

**[2.1操作可行性](#_Toc31715_WPSOffice_Level1)** **[3](#_Toc31715_WPSOffice_Level1)**

**[3需求分析](#_Toc17803_WPSOffice_Level1)** **[4](#_Toc17803_WPSOffice_Level1)**

**[3.1用例图](#_Toc25302_WPSOffice_Level1)** **[4](#_Toc25302_WPSOffice_Level1)**

[3.1.1商城系统用例图](#_Toc4766_WPSOffice_Level2) [4](#_Toc4766_WPSOffice_Level2)

[3.1.2用例描述表](#_Toc21549_WPSOffice_Level2) [6](#_Toc21549_WPSOffice_Level2)

**[3.2总体业务流程分析](#_Toc17953_WPSOffice_Level1)** **[9](#_Toc17953_WPSOffice_Level1)**

**[3.3系统各模块分析](#_Toc11502_WPSOffice_Level1)** **[10](#_Toc11502_WPSOffice_Level1)**

[3.3.1商品展示](#_Toc5003_WPSOffice_Level2) [10](#_Toc5003_WPSOffice_Level2)

[3.3.2购物车功能](#_Toc24322_WPSOffice_Level2) [12](#_Toc24322_WPSOffice_Level2)

[3.3.5评论功能](#_Toc24278_WPSOffice_Level2) [15](#_Toc24278_WPSOffice_Level2)

[3.3.3改密功能](#_Toc31715_WPSOffice_Level2) [16](#_Toc31715_WPSOffice_Level2)

[3.3.4个人中心功能](#_Toc17803_WPSOffice_Level2) [17](#_Toc17803_WPSOffice_Level2)

**[3.4非功能性需求](#_Toc16362_WPSOffice_Level1)** **[19](#_Toc16362_WPSOffice_Level1)**

[3.2.1界面设计](#_Toc25302_WPSOffice_Level2) [19](#_Toc25302_WPSOffice_Level2)

[3.2.3可扩展性](#_Toc17953_WPSOffice_Level2) [19](#_Toc17953_WPSOffice_Level2)

[3.2.4安全性](#_Toc11502_WPSOffice_Level2) [20](#_Toc11502_WPSOffice_Level2)

**[4系统设计](#_Toc1153_WPSOffice_Level1)** **[21](#_Toc1153_WPSOffice_Level1)**

**[4.1开发系统和运行环境](#_Toc775_WPSOffice_Level1)** **[21](#_Toc775_WPSOffice_Level1)**

**[4.2数据库设计](#_Toc5060_WPSOffice_Level1)** **[21](#_Toc5060_WPSOffice_Level1)**

[4.2.1数据库概念结构](#_Toc16362_WPSOffice_Level2) [21](#_Toc16362_WPSOffice_Level2)

[4.2.2 E-R图](#_Toc1153_WPSOffice_Level2) [25](#_Toc1153_WPSOffice_Level2)

[4.2.2数据库逻辑设计](#_Toc775_WPSOffice_Level2) [25](#_Toc775_WPSOffice_Level2)

**[4.2常用工具类](#_Toc10510_WPSOffice_Level1)** **[31](#_Toc10510_WPSOffice_Level1)**

[4.2.1静态成员变量存储](#_Toc5060_WPSOffice_Level2) [31](#_Toc5060_WPSOffice_Level2)

[4.2.2Token缓存](#_Toc10510_WPSOffice_Level2) [31](#_Toc10510_WPSOffice_Level2)

[4.2.3FTP服务](#_Toc3426_WPSOffice_Level2) [31](#_Toc3426_WPSOffice_Level2)

[4.2.4加密工具](#_Toc17342_WPSOffice_Level2) [31](#_Toc17342_WPSOffice_Level2)

[4.2.5Json序列化](#_Toc18135_WPSOffice_Level2) [31](#_Toc18135_WPSOffice_Level2)

**[5系统实现](#_Toc3426_WPSOffice_Level1)** **[31](#_Toc3426_WPSOffice_Level1)**

**[5.1 核心工具](#_Toc17342_WPSOffice_Level1)** **[31](#_Toc17342_WPSOffice_Level1)**

[5.1.1 接口规范设计](#_Toc6773_WPSOffice_Level2) [31](#_Toc6773_WPSOffice_Level2)

[5.1.2token缓存](#_Toc23185_WPSOffice_Level2) [32](#_Toc23185_WPSOffice_Level2)

[5.1.3FTPUtil](#_Toc13386_WPSOffice_Level2) [33](#_Toc13386_WPSOffice_Level2)

[5.1.4React&Component](#_Toc14372_WPSOffice_Level2) [34](#_Toc14372_WPSOffice_Level2)

[5.1.5支付宝付款](#_Toc7157_WPSOffice_Level2) [35](#_Toc7157_WPSOffice_Level2)

[5.1.6Virtual DOM](#_Toc31509_WPSOffice_Level2) [35](#_Toc31509_WPSOffice_Level2)

[5.1.7React-Native原理](#_Toc29672_WPSOffice_Level2) [37](#_Toc29672_WPSOffice_Level2)

[5.1.8 webpack](#_Toc25864_WPSOffice_Level2) [38](#_Toc25864_WPSOffice_Level2)

**[6总结](#_Toc18135_WPSOffice_Level1)** **[41](#_Toc18135_WPSOffice_Level1)**

**[致谢](#_Toc6773_WPSOffice_Level1)** **[43](#_Toc6773_WPSOffice_Level1)**

**[参考文献](#_Toc23185_WPSOffice_Level1)** **[44](#_Toc23185_WPSOffice_Level1)**

# １概述

１.1课题背景和意义

随着当今互联网的快速发展，推进我国经济社会创新发展，传统的购物模式已经不能满足大多数人的需求，在线购物，在线支付等已经成为大众的选择，传统企业也面临着很大的挑战，如何正确成功的 转型变得尤为重要，才能为企业创造一个有良好盈利的基于在线商城的销售渠道。该商城系统为用户提供免费查看，选择其所需产品的功能服务，以及对产品其他附带信息的深入了解（产品评论，产品详情图片等），更好的促进了企业的市场竞争力和在当今互联网模式下的快速发展。

在网站建设初期，应充分了解该系统的所需要的所有功能，以便在后期实现时能更好的处理业务逻辑以及功能。通过对电子商城系统的实现来理解和体会在线购物的一体化流程。

１.２本课题的研究内容

## 本课题主要研究基于在线购物的一整个购物流程和对应的跨平台技术的实现。一个成熟的电子商城应该拥有如下功能：商品浏览，商品详情，商品评论，购物车，订单管理，登录注册，支付，管理员数据分析等功能。其次就是跨平台，使用JavaScript的跨平台特性，将代码渲染为Android端原生的UI，实现一种技术在多端下运行。

１.３跨平台技术和电子商务现状

目前跨平台技术体系在中国已经成熟，随着中国互联网的发展和庞大的用户体系，新技术如雨后春笋般萌发，也经受住了一些大企在实际业务中的考验，包括支付，跨平台，前端组件化等优秀的解决方案和众多成功案例,例如:国内已经有众多企业选择使用跨平台技术开发，国内知名的淘宝，优酷，饿了吗，美团等，他们的APP均内置了跨平台技术。

电子商务目前在我国的现状也是非常可观的，到目前为止还一直处于上升中，中国排名第一的互联网企业阿里巴巴主要盈利也是来自淘宝，中国电商的销售额在去年也是占世界第一，达到了9750亿美元，中国的电子商务存在巨大的潜力和上升的空间，前景十分看好，基于电子商务系统下的产品也是很多，淘宝，天猫，网易严选，唯品会，京东等，一切销售型的企业都应该在电商深耕，电商也在不断丰富自身的产品，力求提供一个更加完善的服务。

# 2可行性分析

2.1技术可行性

当今互联网已经有众多的公司使用跨平台方式开发移动APP，类似目前主流的微博，饿了吗，美团，手淘等，他们的APP内都内置了跨平台技术，可像浏览器加载html页面一样从服务器加载JS脚本渲染为原生页面。商城包括两个平台，一个是基于浏览器平台的后台管理和用户商城系统，一个是基于原生APP的用户商城APP，PC门户网站使用浏览器搭建，安卓原生APP使用React-Native渲染为原生native页面。自己在前端方面也研究了很多时间，因此，该系统在技术上是可行的。

后端JAVA更是无可厚非了，java EE一整套的开发体系，长达二十多年的技术沉淀，既简化了开发难度缩短了开发周期，经典的 MVC分层模式，使代码的可维护，可读性都有极大的提高。

2.1操作可行性

通过参考国内淘宝电商的交互和界面，尽可能符合人们使用习惯以及交互并简化操作流程，并增加系统使用人群，并添加测试用户，咨询他们的意见和反馈并做出一定修改。因此，在操作上是可行的 。

# 3需求分析

3.1用例图

3.1.1商城系统用例图

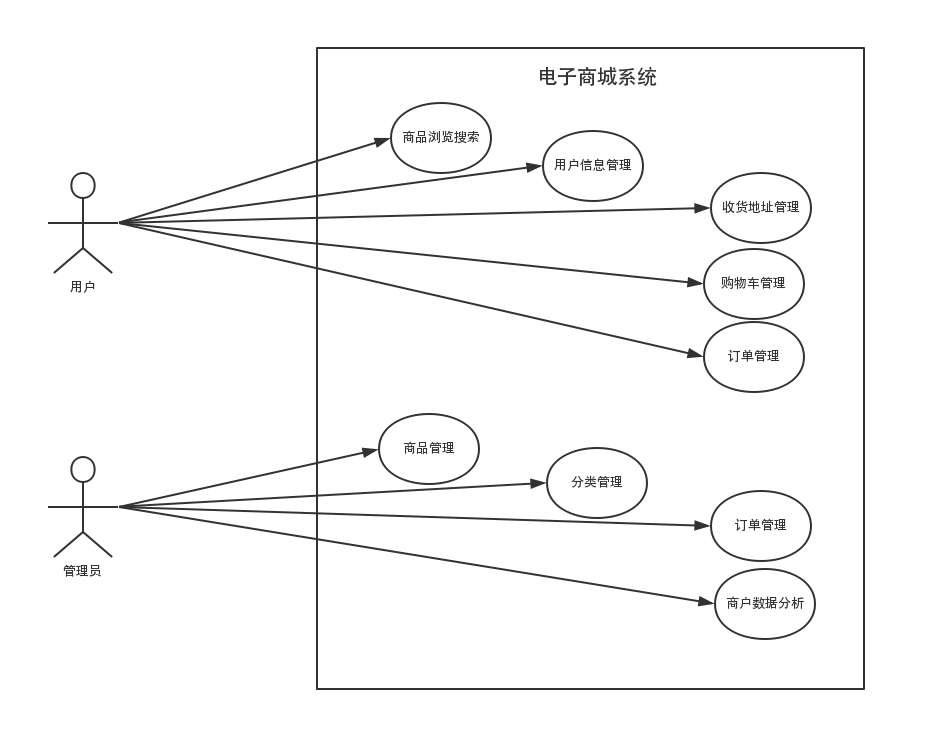
图3- 1商城系统用例图

图3- 1描述了该系统的主要的功能，该系统分为两个角色，管理员和用户，管理员主要对商城进行管理和数据分析，例如查看商城最近的订单，最近成交的商品，各分类下的数据分析等等。用户则是进入商城主页，根据各种查询条件和排序条件以及页码信息进行查找浏览，找到想要购买的商品后可直接加入购物车也可以进入商品详情页面，查看更多商品的信息，例如该商品其他用户购买后的使用评论，存货数量，相关的商品图片等信息，在加入购物车之前检查用户是否登录，没有登录时提醒用户需要登录后才能操作，登录成功后选择对应的商品和数量加入到购物车，用户根据实际情况选择购物车内的商品进行付款，实时计算总价后选择或者新增一个对应的收货地址后确认订单并跳转到付款页面，使用支付宝扫一扫付款，付款成功后后台订单对该笔订单发货，用户收到后确认收货，确认收货后即可对订单内的商品进行评论，该笔订单关闭。

表 3‑1商城功能介绍表

|  |  |
| --- | --- |
| 功能 | 介绍 |
| 商品浏览与搜索 | 用户浏览商品列表，模糊搜索，分类搜索等功能 |
| 用户信息管理 | 用户注册登录，改密 |
| 购物车管理 | 加入购物车，动态计算总价，购物车增删改查以及结算 |
| 地址管理 | 收货地址的增加删除和修改 |
| 订单管理 | 结算后生成订单，未付款，付款，发货，签收等状态的修改，对接蚂蚁金服，实现实时交易。 |
| 商品添加和上下架 | 添加商品信息，上传商品图片，商品上下架 |
| 评论功能 | 用户订单已收货后对订单内商品进行评论 |
| 分类管理 | 增加，修改，删除分类等 |

3.1.2用例描述表详情

用例图是由需求到实现的第一步，目标系统需要哪些看不见的功能或者非常显而易见的功能，有哪些参与用户，画出用例图还不能具体的体现出具体的需求，接下来还要实现多功能的进一步分析，哪些用例之间他们互相存在关系，具体功能细节需要注意什么等等。单单用例图往往会显得太简单了，不足以描述某个用例的详细信息，使用用例描述表来描述这种情况非常有用，也可以做进一步分析，如下，使用用例表常用属性来描述各个用例的情况。

1)对商品搜索与浏览用例进行详情描述，如下表3-1

表3- 1商品搜索与浏览用例描述表

|  |
| --- |
| 用例名称：商品搜索与浏览 |
| 用例参与者：用户 |
| 前置条件：用户进入系统 |
| 后置条件：用户找到自己需要购买的东西 |
| 用例描述：根据参数和分类搜索商品 |
| 补充说明:页码信息默认一页八条数据，第一页。分类默认全局查看 |

2)对用户信息管理用例进行详情描述，如下表3-2

表3- 2用户信息管理用例描述表

|  |
| --- |
| 用例名称：用户信息管理 |
| 用例参与者：用户 |
| 前置条件：用户已登陆且密保问题与答案不为空。 |
| 用例描述：根据密保问题找回密码或根据旧密码找回。 |
| 后置条件：旧密码或者密保问题验证成功即可修改密码，否则无法修改。 |
| 补充说明: |

3)对收货地址管理用例进行详情描述，如下表3-3

表3- 3收货地址管理用例描述表

|  |
| --- |
| 用例名称：收货地址管理 |
| 用例参与者：用户 |
| 前置条件：用户已登陆 |
| 用例描述：用户对自己收货地址的增加和修改，在结算时选择对应的收货地址。 |
| 后置条件：修改地址信息或者新增地址信息成功，否则无法做任何操作。 |
| 补充说明:收货地址应包括收货人，电话，详细地址和邮编 |

4)对购物车管理用例进行详情描述，如下表3-4

表3- 4购物车管理用例描述表

|  |
| --- |
| 用例名称：购物车管理 |
| 用例参与者：用户 |
| 前置条件：用户已登陆 |
| 用例描述：用户对自己购物车内的商品进行数量的修改，删除，以及选择加入到待付款。 |
| 后置条件：成功修改对应的商品的数量，选择是否加入到待付款商品，否则无法查看和操作。 |
| 补充说明：商品浏览出加入购物车数量默认为一，商品主页加入购物车用户可选择对应的数量，加入商品的数量不能超过商品的库存。 |

5)对订单管理用例进行详情描述，如下表3-5

表3- 5订单管理用例描述表

|  |
| --- |
| 用例名称：订单管理 |
| 用例参与者：用户 |
| 前置条件：用户已登陆，且对购物车已结算，生成对应订单 |
| 用例描述：用户对订单付款，签收，订单内商品进行评论 |
| 后置条件：对订单的对应操作成功，否则无法查看到订单信息 |
| 补充说明：订单状态有未付款，已付款，已发货，已签收，关闭这五种状态 |

6)对商品管理用例进行详情描述，如下表3-6

表3- 6订单管理用例描述表

|  |
| --- |
| 用例名称：商品管理 |
| 用例参与者：管理员 |
| 前置条件：管理员进入后台并登录 |
| 用例描述：管理员对商品进行上下架，修改商品信息，上传商品。 |
| 后置条件：操作商品成功，否则无法进入系统 |
| 补充说明：已下架的商品无法被前台用户所看到。 |

7)对分类管理用例进行详情描述，如下表3-7

表3- 7分类管理用例描述表

|  |
| --- |
| 用例名称：分类管理 |
| 用例参与者：管理员 |
| 前置条件：管理员进入后台并登录 |
| 用例描述：对选中分类新增子分类和修改分类信息 |
| 后置条件：操作分类成功，否则无法进入系统 |
| 补充说明：分类的子分类是指该分类下的分类，考虑到商品的复杂性，分类是可多层级的。 |

8)对订单管理用例进行详情描述，如下表3-8

表3- 8订单管理用例描述表

|  |
| --- |
| 用例名称：订单管理 |
| 用例参与者：管理员 |
| 前置条件：管理员进入后台并登录 |
| 用例描述：根据订单用户所选择的地址对该笔订单的商品进行发货，修改订单状态 |
| 后置条件：订单发货成功，否则无法进入系统 |
| 补充说明：管理员根据地址内的收货人姓名，地址，邮编进行第三方物流发货 |

9)对商户数据分析进行详情描述，如下表3-9

表3- 9商户数据分析描述表

|  |
| --- |
| 用例名称：商户数据分析 |
| 用例参与者：管理员 |
| 前置条件：管理员进入后台并登录 |
| 用例描述：从用户，分类，订单，成交金额等方面多元化分析商城数据。 |
| 后置条件：管理员通过各图表信息获取该商城的运营情况。 |
| 补充说明：按照时间的角度分析该月份对应的的营收以及其他情况。 |

3.2活动图

3.2.1商城系统活动图

活动图是UML中对系统动态方面建模的主要形式，它强调的是活动到活动之间的控制流，用来对业务过程，工作流程建模，在本质上，他是一种流程图，但他注重表现一个活动到另一个活动的控制流，是内部处理驱动的流程。

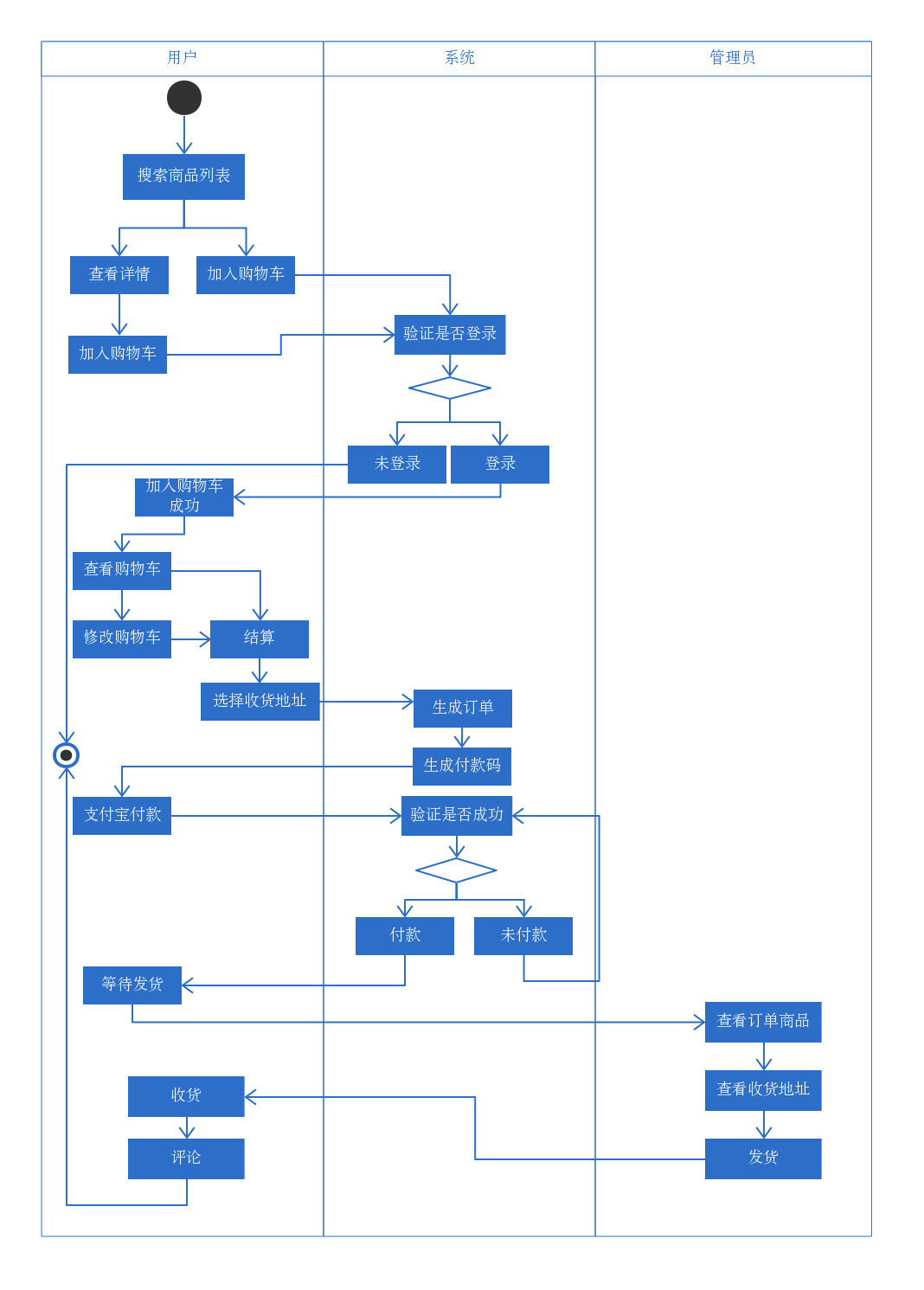
本商城系统的活动图如图3- 2所示：

图3- 2商城系统活动图

3.2.2商品展示活动图

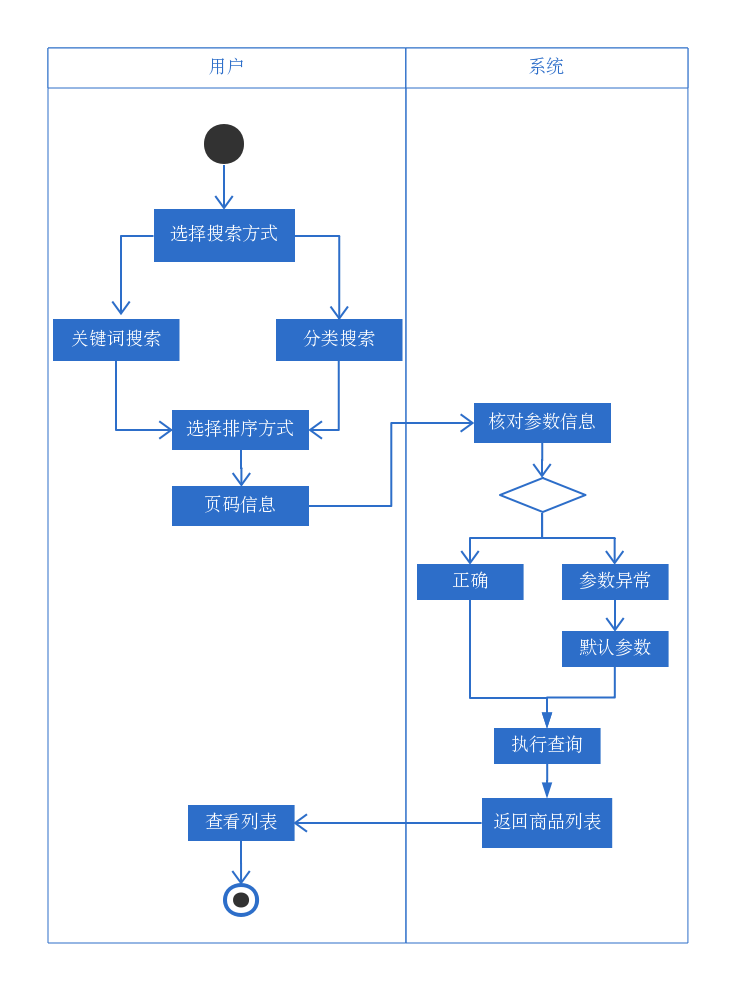
1)商品列表展示

图3- 3 商品搜索活动图

如上图3-3所示，商品信息选购是整个系统最重要的模块，也是最基本的功能，模块包括商品分类展示，按照日期、存货、价格对数据进行排序，更方便直观的让用户找到自己想找的东西。商品搜索功能，根据商品名称进行模糊查询。

1. 商品详情展示

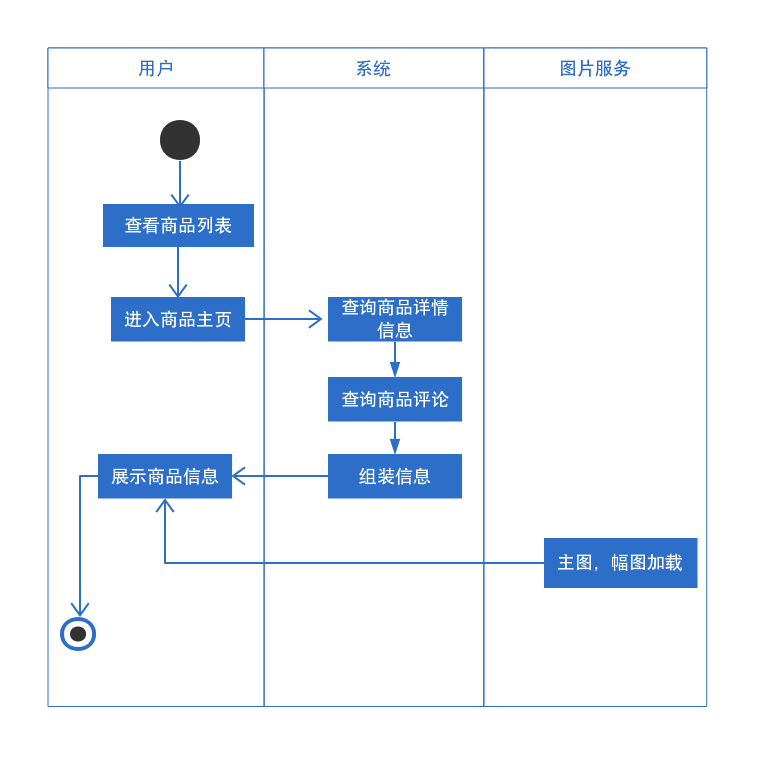
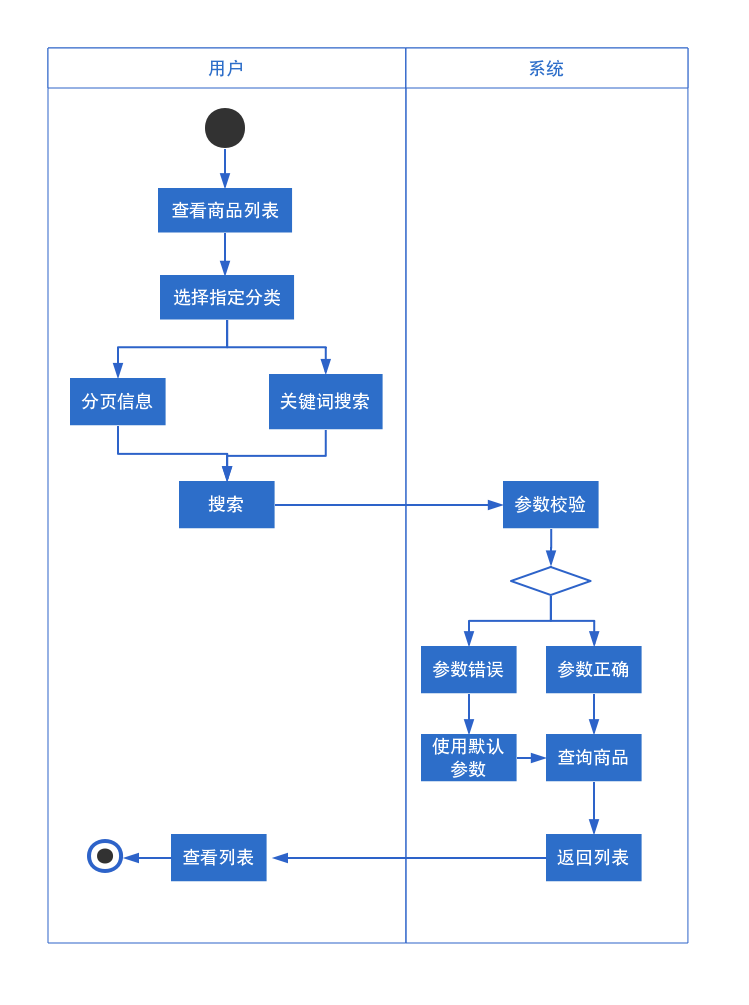


图3- 4 商品详情展示活动图

如上图3-4所示，商品详情主要展示当前商品的基本信息，每个商品分为一个主图和多个附图，其他一些基本信息和富文本介绍，以及购买过该商品的其他用户的评价等，当用户在商品列表看到自己喜欢的东西后就可以进入商品主页进行查看这些信息。

1. 分类展示

图3- 5分类展示活动图

用户可以根据具体的分类查询该制定分类或者子分类或者下级分类下的商品，考虑到商品的复杂性，分类往往有多个层级，例如：手机可能按品牌分，各大平牌下还可以再按照价格或者适用人群细分。

3.2.2购物车活动图

1）加入购物车

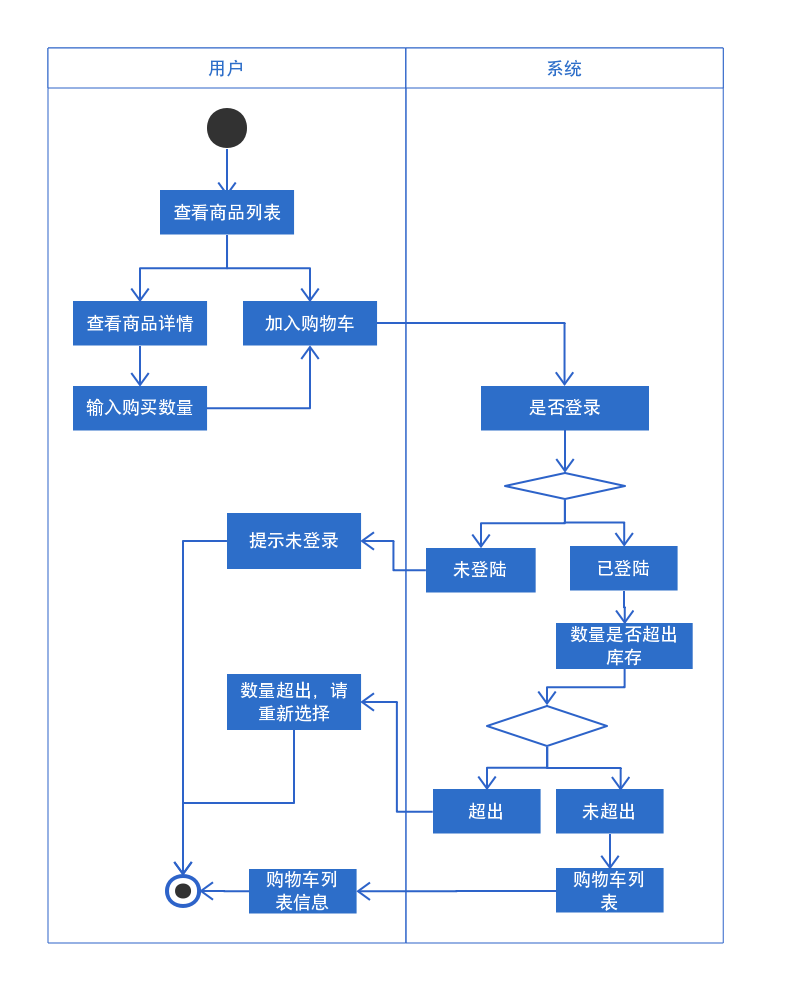


图3- 6 加入购物车活动图

如上图3-6所示，用户可以对要买的商品以及对应的商品数量选择加入购物车中，并在购物车页面对商品数量进行操作，动态展示出操作后的对应的商品金额和总价，对商品进行全选和反选功能，最后对选择的商品进行删除或者结算。

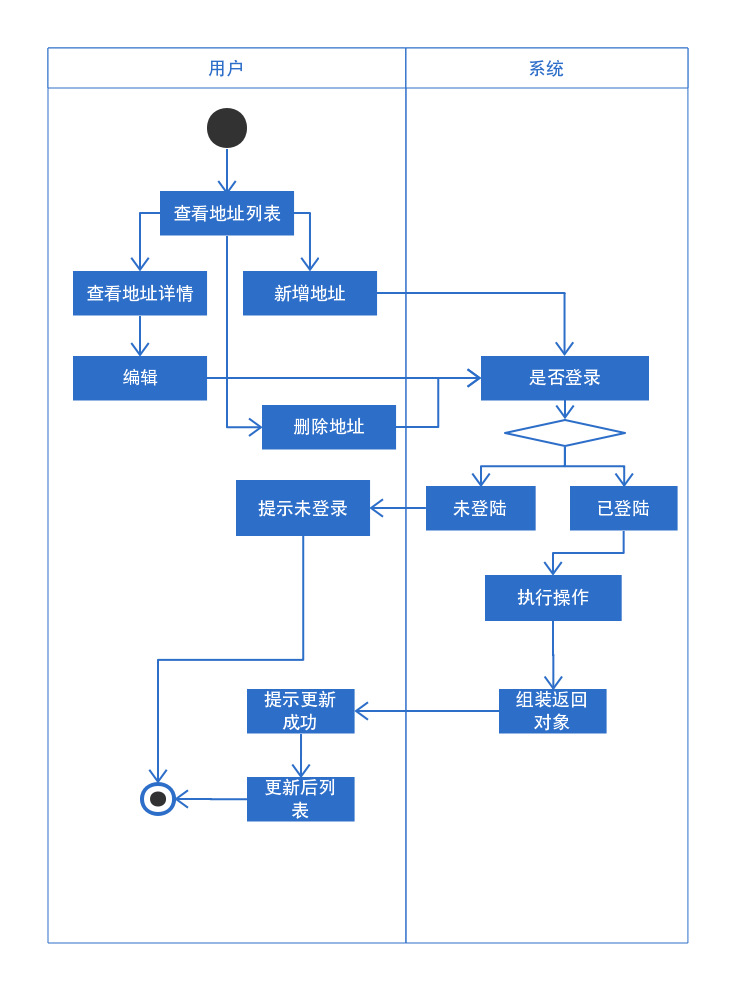
1. 地址管理功能

图3- 7地址管理活动图

如上图3-7所示，地址模块用于存储用户的地址信息，即用户的收货地址，购物车结算前用户需要选择或者新增收货地址并确定后才能进行下一步付款操作。

3）购物车结算

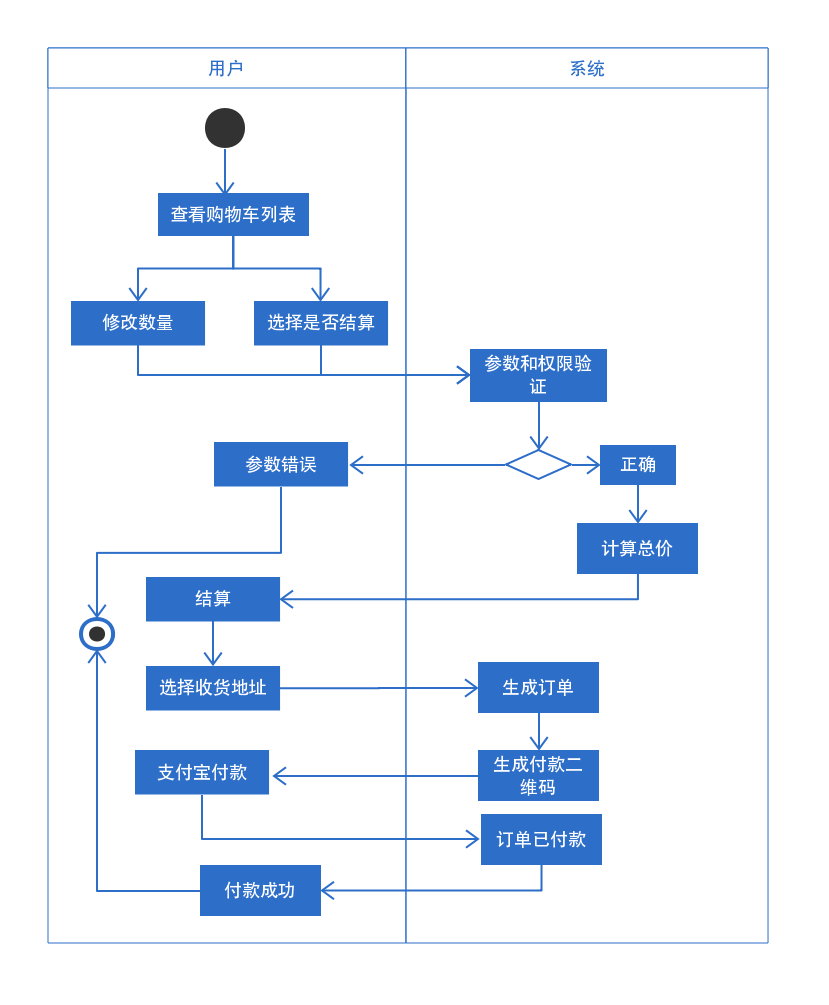


图3- 8购物车结算活动图

如上图3- 8所示，用户选择对应的结算商品，可对多个商品进行结算，选择收货地址后生成对应的订单信息，跳转到订单页面进行付款。

3.2.3商户数据分析活动图

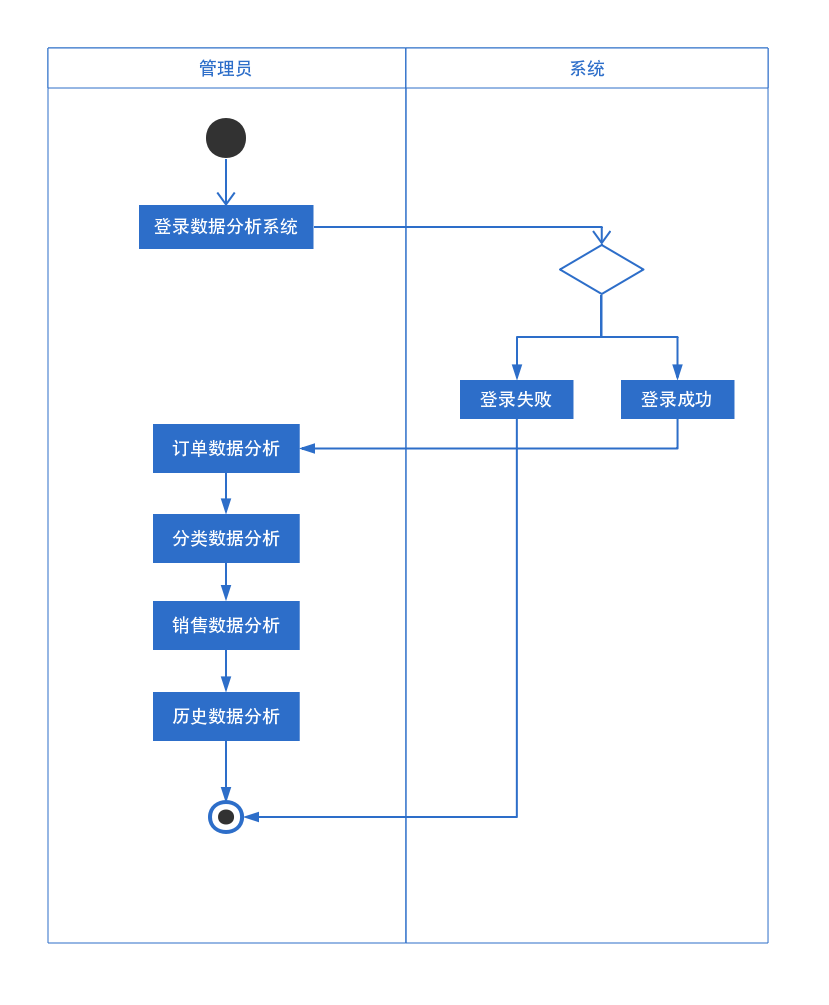


图3- 9商户数据分析活动图

如上图3-9所示，商户可通过后台数据分析系统进行查看该系统的数据，例如年度销售额，每月销售量，最新成交订单以及各分类下商品占比等数据。

3.3时序图

3.3.1用户注册时序图

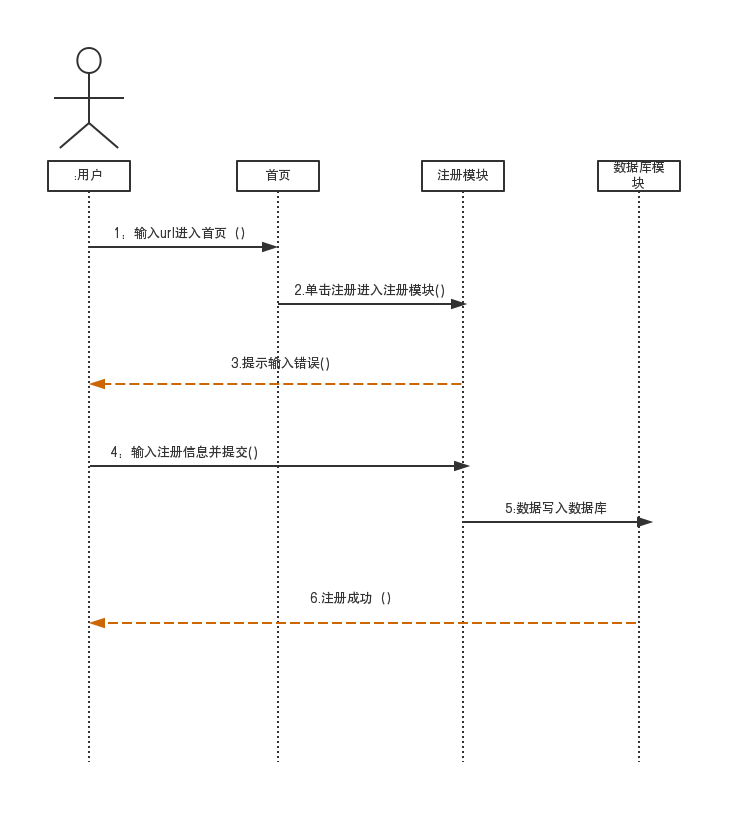


图3- 10用户注册时序图

3.3.2搜索商品时序图

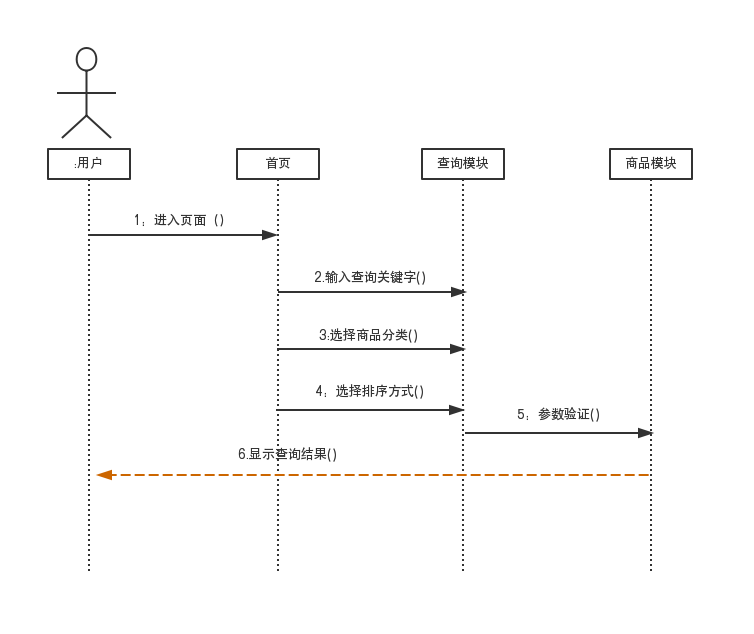


图3- 11用户查询时序图

3.3.3管理员添加商品时序图

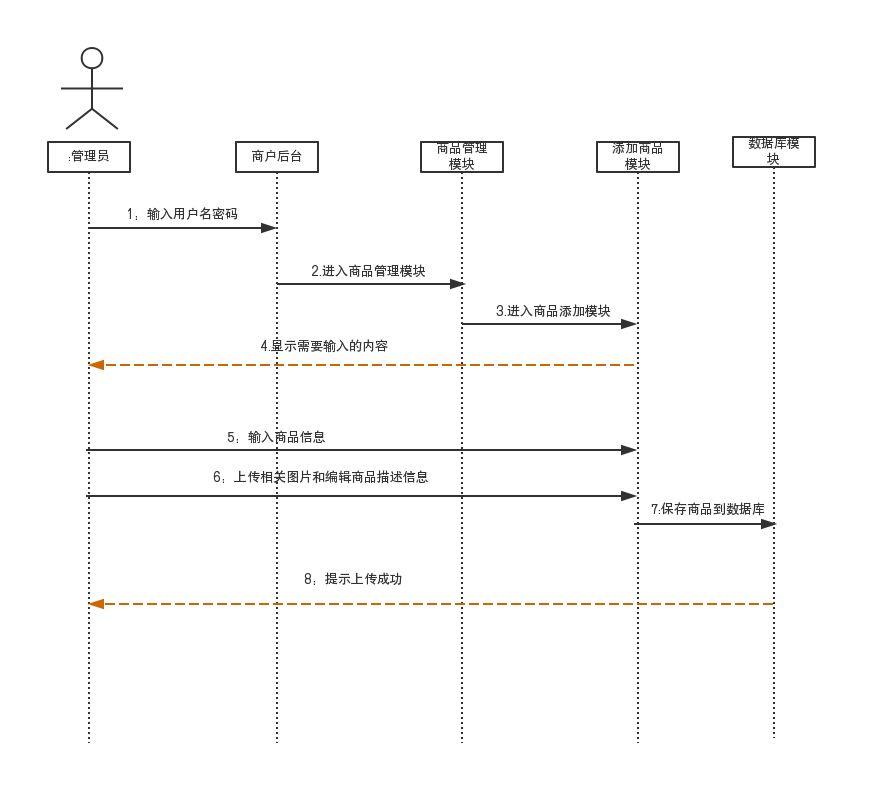


图3- 12管理员新增商品时序图

3.3.4管理员添加分类时序图

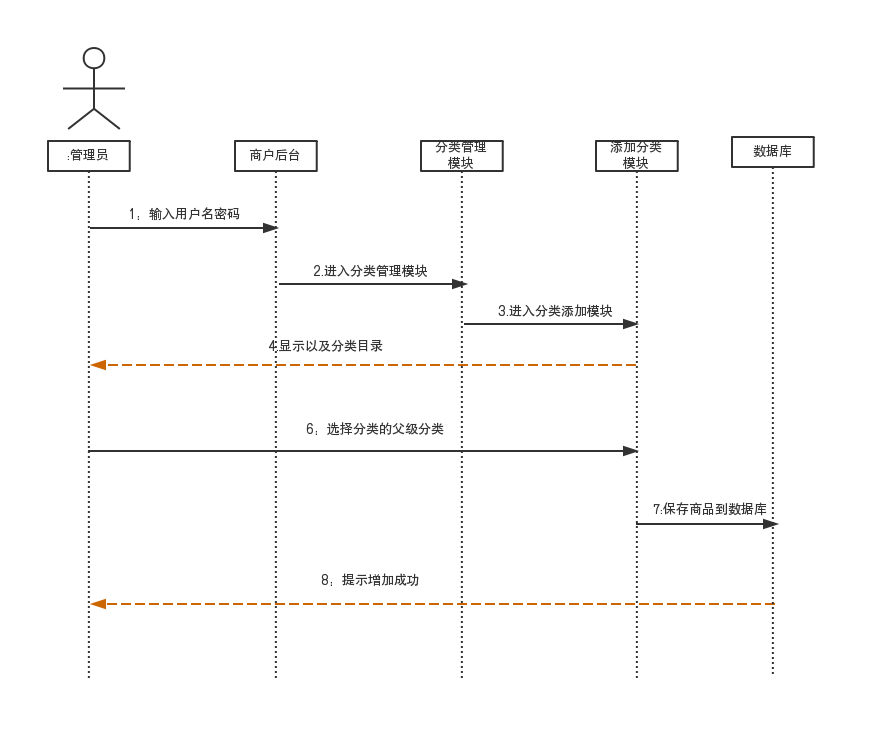


图3- 13管理员新增分类时序图

3.4非功能性需求

3.2.1界面设计和性能

开发出所有当前主流系统的一整套体系产品，包括Android，IOS，PC浏览器端一套产品，并实现UI下的高交互和流畅的用户体验，以及页面的美观性和各不同分辨率的适配性

3.2.3可扩展性

系统后期维护和升级时应当给后台的接口预留更多的配置，以适应系统后期复杂多变的业务环境，以及需求的扩展和业务的变化，对前后端代码都要尽可能的模块化，添加新功能或者修改以往业务时不会影响到其他模块的正常运行。

3.2.4安全性

本系统有多个用户，用户信息的安全性，交易双方的安全性，数据的安全性，系统都应该尽可能的防患于未然，在安全上下保障客户和商家，避免利益丢失。

4系统设计

4.1开发系统和运行环境

系统开发环境:

语言：Java,JavaScript

后端服务器:TomCat9.0,egg.js

数据库：Mysql 5.6

前端框架:React,Vue,React-Native

后台框架:MyBatis+SPring+SpringMVC

按照传统的MVC的设计思想开发

4.2数据库设计

4.2.1数据库概念结构

在需求分析的阶段所总结出的需求，将其抽象为信息的结构，然后结合数据库的表关系，数据字段，数据类型等数据库知识来实现这些具体需求。

概念模型的主要特点有：

1. 能很好的反应模块与模块之间，事物与事物之间的相关的关系，并满足用户对数据的处理要求，是系统真实的模型。
2. 方便修改，当系统环境和系统的需求修改时，可快速的对概念模型进行修改和补充。
3. 易于于其他各种数据模型进行相互转换。
4. 方便理解，无论用户是否熟悉计算机操作，都可以与其交换意见，使用户积极参与到数据库设计中，是系统成功的主要因素。

用户分为普通用户与管理员用户，普通用户只能进行购买浏览和评论等操作，只有权限为管理员的用户才能对系统进行管理。

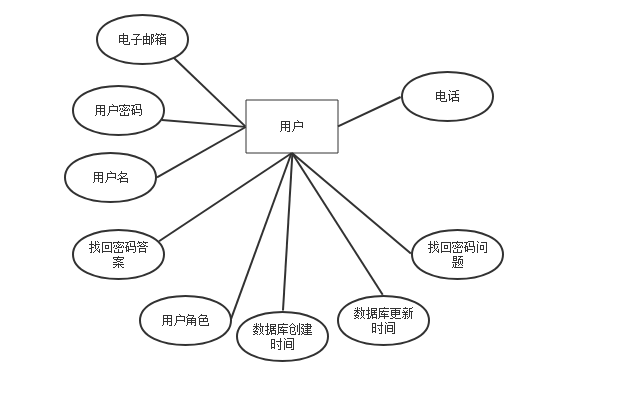


图4-1用户实体图

商品的信息是否能完整的反映出一个商品的所有信息，上传的商品信息越是详细，用户对商品的了解程度就越高，就能更大的程度上激发用户的购买欲望，价格，存货，商品介绍，图片以及其他信息都能吸引客户的眼球。

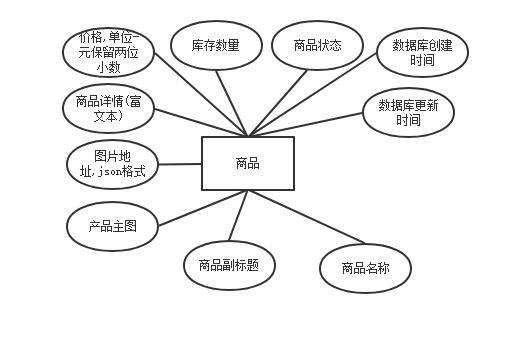


图4-2商品实体图

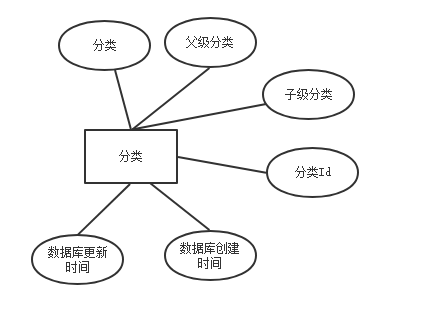
商品的数量很多时，搜索功能已经不能满足用户的需求，并且用户很多时候都是想要买某一分类下的商品。

图4-3分类实体图

订单是对用户购买商品的记录，为了方便购买后的操作以及服务，从购买到付款到管理员发货，然后签收，商品评论的一体化服务，保证了消费者利益最大化，通常一个完整的订单要从订单信息中知道用户的收货地址信息，用户购买的商品信息，订单的价格以及一些相对的付款时间，发货时间等。

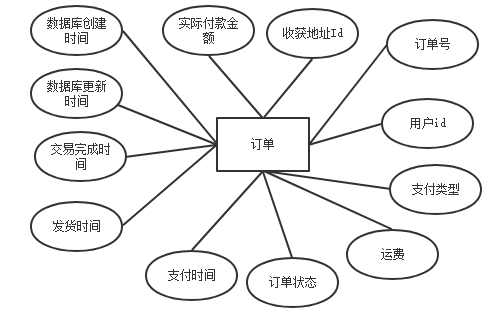


图4-4订单实体图

购物车是用户购买商品的前提，只有加入了自己的购物车才能对购物车进行付款。

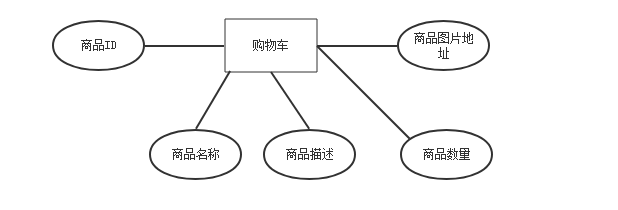


图4-5购物车实体图

4.2.2 E-R图

E-R图提供了表示实体，属性以及联系的方法：

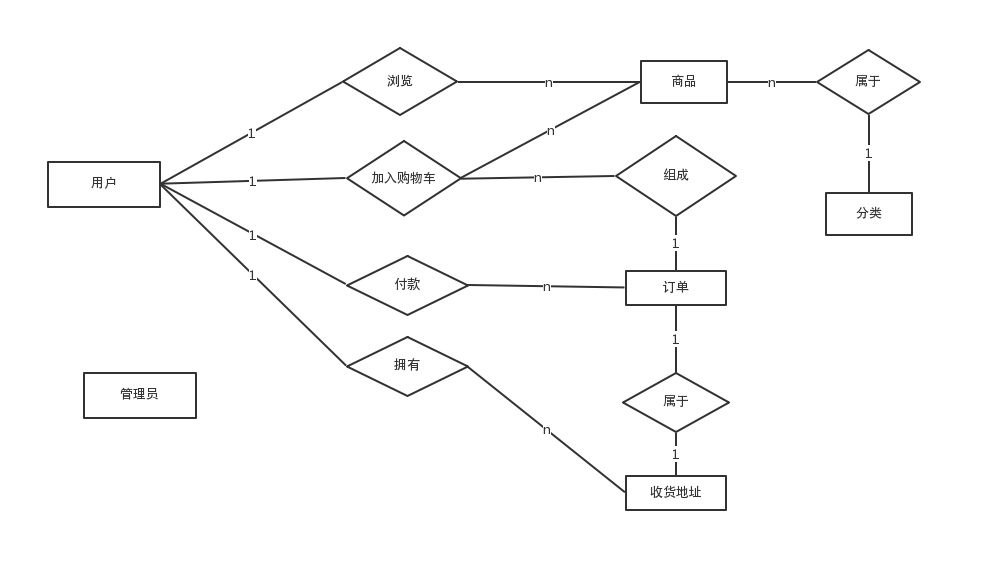


图4-6商城E-R图

如图4-6所示，E-R图说明了该商城每个实体和属性的关系，一个用户可以浏览多个商品，多个商品属于一个分类，一个用户可以加入多个商品到购物车，购物车组成一个订单，用户对多个订单付款，一个订单有一个收货地址，一个用户有多个收货地址。

4.2.2数据库逻辑设计

从需求到设计，从设计到实现，从数据库到数据表，再从数据表到数据，每个部分缺一不可。由他们共同组成一个完整的数据库。

结合系统的业务逻辑以及该系统的需求设计出以下表结构：

表4- 1用户表（mall\_user）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 否 | 主键自增长 |
| 2 | username | VARCHAR | 50 | 否 | 否 | 用户名 |
| 3 | password | VARCHAR | 50 | 否 | 否 | Md5加密后的密码 |
| 4 | m\_email | VARCHAR | 50 | 否 | 否 | 邮箱 |

续表4- 1用户表（mall\_user）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5 | m\_phone | VARCHAR | 20 | 否 | 否 | 电话 |
| 6 | question | VARCHAR | 100 | 否 | 否 | 找回密码问题 |
| 7 | answer | VARCHAR | 100 | 否 | 否 | 找回密码答案 |
| 8 | role | int | 1 | 否 | 否 | 角色权限 |
| 9 | Create-time | DATETIME |  | 否 | 否 | 创建时间 |
| 10 | Update-time | DATETIME |  | 否 | 否 | 修改时间 |

表4- 2商品表（mall\_product）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 否 | 主键自增长 |
| 2 | category\_id | Int | 11 | 否 | 否 | 分类表外键 |
| 3 | name | VARCHAR | 100 | 否 | 否 | 商品名称 |
| 4 | subtitle | VARCHAR | 200 | 否 | 否 | 商品副标题 |
| 5 | main\_image | VARCHAR | 100 | 否 | 否 | 产品主图 |
| 6 | sub\_images | VARCHAR | 500 | 否 | 否 | 图片副图地址 |
| 7 | detail | LONGTEXT |  | 否 | 否 | 富文本详情 |
| 8 | price | decimal | (20,2) | 否 | 否 | 价格,单位-元 |
| 9 | stock | int | 11 | 否 | 否 | 库存数量 |
| 10 | status | int | 1 | 否 | 否 | 商品状态 |
| 11 | create\_time | DATETIME |  | 否 | 否 | 创建时间 |
| 12 | update\_time | DATETIME |  | 否 | 否 | 更新时间 |

表 4‑3分类表（mall\_category）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 否 | 主键自增长 |
| 2 | parent\_id | int | 11 | 否 | 否 | 父类id |
| 3 | name | VARCHAR | 20 | 否 | 否 | 类别名称 |

续表 4‑3分类表（mall\_category）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | status | INT | 1 | 否 | 否 | 类别状态1 |
| 5 | create\_time | DATETIME |  | 否 | 否 | 创建时间 |
| 6 | update\_time | DATETIME |  | 否 | 否 | 更新时间 |

表 4‑4评论表（mall\_comment）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 否 | 主键自增长 |
| 2 | productId | int | 11 | 否 | 否 | 商品外键 |
| 3 | userId | int | 11 | 否 | 否 | 用户外键 |
| 4 | username | VARCHAR | 100 | 否 | 否 | 评论的用户名 |
| 5 | cStarts | int | 1 | 否 | 否 | 星级 |
| 6 | cContent | VARCHAR | 500 | 否 | 否 | 内容 |
| 7 | cTime | DATETIME |  | 否 | 否 | 评论时间 |

表 4‑5购物车表（mall\_cart）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 否 | 主键自增长 |
| 2 | user\_id | int | 11 | 否 | 否 | 用户ID外键 |
| 3 | product\_id | int | 11 | 否 | 否 | 商品id外键 |
| 4 | quantity | int | 11 | 否 | 否 | 数量 |
| 5 | checked | int | 1 | 否 | 否 | 是否选择 |
| 6 | create\_time | DATETIME |  | 否 | 否 | 创建时间 |
| 7 | update\_time | DATETIME |  | 否 | 否 | 更新时间 |

表 4‑6订单表（mall\_order）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 否 | 主键自增长 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | order\_no | BIGINT | 20 | 否 | 否 | 订单号 |
| 3 | user\_id | int | 11 | 否 | 否 | 用户id |
| 4 | shipping\_id | int | 11 | 否 | 否 | 地址ID |
| 5 | payment | decimal | (20,2) | 否 | 否 | 实际付款金额 |
| 6 | payment\_type | int | 1 | 否 | 否 | 支付类型 |
| 7 | postage | int | 10 | 否 | 否 | 运费, |
| 8 | status | int | 1 | 否 | 否 | 订单状态 |
| 9 | payment\_time | DATETIME |  | 否 | 是 | 支付时间 |
| 10 | send\_time | DATETIME |  | 否 | 是 | 发货时间 |
| 11 | end\_time | DATETIME |  | 否 | 是 | 交易完成时间 |
| 12 | close\_time | DATETIME |  | 否 | 是 | 交易关闭时间 |
| 13 | create\_time | DATETIME |  | 否 | 否 | 创建时间 |
| 14 | update\_time | DATETIME |  | 否 | 否 | 更新时间 |

表 4‑7订单快照表（mall\_order\_item）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 否 | 主键自增长 |
| 2 | user\_id | int | 11 | 否 | 否 | 用户ID外键 |
| 3 | order\_no | BIGINT | 20 | 否 | 否 | 订单号 |
| 4 | product\_id | int | 11 | 否 | 否 | 商品ID |
| 5 | product\_name | VARCHAR | 100 | 否 | 否 | 商品名称 |
| 6 | product\_image | VARCHAR | 500 | 否 | 否 | 商品图片地址 |
| 7 | current\_unit\_price | decimal | (20,2) | 否 | 否 | 购买时价格 |
| 8 | quantity | int | 10 | 否 | 否 | 商品数量 |
| 9 | total\_price | decimal | (20,2) | 否 | 否 | 商品总价 |
| 10 | create\_time | DATETIME |  | 否 | 否 | 创建时间 |

续表 4‑7订单快照表（mall\_order\_item）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 11 | update\_time | DATETIME |  | 否 | 否 | 更新时间 |

表 4‑8支付信息表（mall\_pay\_info）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 否 | 主键自增长 |
| 2 | user\_id | Int | 11 | 否 | 否 | 用户id |
| 3 | order\_no | BIGINT | 20 | 否 | 否 | 订单号 |
| 4 | pay\_platform | Int | 10 | 否 | 否 | 支付平台 |
| 5 | platform\_number | VARCHAR | 200 | 否 | 否 | 支付宝流水号 |
| 6 | platform\_status | VARCHAR | 20 | 否 | 否 | 支付宝支付状态 |
| 7 | create\_time | DATETIME |  | 否 | 否 | 创建时间 |
| 8 | update\_time | DATETIME |  | 否 | 否 | 更新时间 |

表 4‑9收货地址表（mall\_shipping）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **数据类型** | **长度** | **主键** | **允许空** | **说明** |
| 1 | id | int | 11 | 是 | 否 | 主键自增长 |
| 2 | user\_id | int | 11 | 否 | 否 | 用户id |
| 3 | receiver\_name | VARCHAR | 50 | 否 | 否 | 收货姓名 |
| 4 | receiver\_phone | VARCHAR | 20 | 否 | 否 | 收货固定电话 |
| 5 | receiver\_city | VARCHAR | 100 | 否 | 否 | 省份 |
| 6 | receiver\_address | VARCHAR | 200 | 否 | 否 | 详细地址 |
| 7 | receiver\_zip | VARCHAR | 6 | 否 | 否 | 邮编 |
| 9 | create\_time | DATETIME |  | 否 | 否 | 创建时间 |
| 10 | update\_time | DATETIME |  | 否 | 是 | 更新时间 |

4.2常用工具类

4.2.1静态成员变量存储

Const:存储静态变量信息，防止人工输入错误导致系统出错，例如：角色是否管理员（IsAdmin），支付方式(PaymentType)，订单状态(OrderStatus)等。

4.2.2Token缓存

CacheToken:缓存Token，用户根据密保问题找回答案验证成功后，返回给用户一个有效期为两个小时的token，修改密码时根据token的有效性验证是否修改密码。

4.2.3FTP服务

FTPUtils：封装ftp上传图片的功能，本系统的图片存储使用第三方ftp图片服务器存储，后端服务不直接存储任何图片，只提供API接口服务。

4.2.4加密工具

MD5Util：为了保护用户信息的安全性，避免数据库被黑客攻击后拿到密码，对用户密码进行md5加盐加密，提高系统的安全性。

4.2.5Json序列化

ServerResponse:返回序列化的Json数据类，更节省网络流量，使用ServerResponse.createSuccess()和ServerResponse.createByErrorCodeMessage()方法定义返回成功态和失败态，返回对应的状态码和提示语。

# 5系统实现

本系统后台逻辑业务主要使用JAVAEE开发，使用MAVEN管理版本依赖；

后台数据分析使用Nodejs下阿里巴巴开源后台框架egg开发，由于数据分析语句涉及大量表连接和频繁的数据库查询，基于node语言特性的非阻塞异步IO来处理多条数据库查询时的等待和延时，很大程度上提高了查询的性能和并发量；

5.1 核心工具代码

5.1.1 接口规范

整个系统采用前后端分离的方式，即前端使用Ajax进行request请求处理，使用当今流行的Axios框架处理异步，后台返回json数据格式，系统初期，应统一后端返回数据格式，每个返回的json中都应该含有status属性，成功为0异常为一，并把数据放在data中，例如，前台操作成功的结果应该为

Response={

Status:0,

Data:[....]

}

操作失败应返回｛status:1,msg:”失败原因”｝规定好后方便前台处理,

并使用node代理后台端口处理浏览器跨域问题，该商城系统主要架构图如下:

5.1.2token缓存

用户忘记密码时使用密保问题找回密码时，密保验证成功后，如何防止其他用户利用此API进行恶意修改密码，保障系统的安全性，如何判断该用户是否已经验证密保问题，比较好的解决方案就是生成token缓存来确定用户身份，用户验证成功后生成给用户一个有限时间的键值对，当用户使用密保修改问题时必须带上token，后台验证成功后才能修改密码。

import com.google.common.cache.CacheBuilder;

import com.google.common.cache.CacheLoader;

import com.google.common.cache.LoadingCache;

public class CjTokenCache {

  private static Logger loggerCj=LoggerFactory.getLogger(TokenCache.class);

        //创建10000可存放键值对的cache，初始化1000个，有效期12小时

        private  static LoadingCache<String,String> localCacheCj=CacheBuilder.newBuilder().initialCapacity(1000).

                maximumSize(10000).expireAfterAccess(12,TimeUnit.HOURS)

                .build(**new** LoadingCacheCj<String, String>(){

                    @Override

                    public String loadCj(String s){

**return**  "null";

                    }

                });

    public  static void setKeyCj(String key,String value){

localCacheCj.put(key,value);

    }

    public static String getKeyCj(String key){

        String value=**null**;

**try**{

            value=localCacheCj.get(key);

**if**("null".equals(value)){

**return**  **null**;

        }

**return**  value;

        }**catch** (Exception error){

            loggerCj.error("LocalCacheCj get Error",error);

        }

**return**  **null**;

    }

}

5.1.3FTPUtil

封装了连接ftp服务以及用于上传图片文件到第三方FTP服务用于存储对应的图片信息。

public class FtpUtilMy {

public static boolean uploadFileCj (List<File> fileList) throws IOException {

    FtpUtil ftpUtil=**new** FtpUtil(“localhost”,21,”CJ”,”123456”);

    boolean result=ftpUtil.uploadFile("img",fileList);

**return** result;

}

private  boolean uploadFile(String remotePath,List<File> fileList) throws IOException {

    boolean upLoaded=**true**;

    FileInputStream fileInputStreamCj=**null**;

**if**(connectServer(**this**.ip,**this**.port,**this**.user,**this**.pwd)){

**try** {

            ftpClientCj.changeWorkingDirectory(remotePath);

            ftpClientCj.setBufferSize(1024);

            ftpClientCj.setControlEncoding("UTF-8");

            ftpClientCj.setFileType(FTPClient.BINARY\_FILE\_TYPE);

            ftpClientCj.enterLocalPassiveMode();

**for** (File fileItem:fileList) {

                fileInputStream=**new** FileInputStream(fileItem);

                ftpClientCj.storeFile(fileItem.getName(),fileInputStream);

            }

        } **catch** (IOException e) {

            logger.error("文件上传异常!",e);

            upLoaded=**false**;

        }finally {

            fileInputStream.close();

            ftpClientCj.disconnect();

        }

    }

**return** upLoaded;

}//上传文件，使用ip，端口，用户名以及密码连接ftp服务

private  boolean connectToServer(String Cjip,int Cjport,String CjuserName,String,PwdCj){

    ftpClientCj=**new** FTPClient();

    boolean isSuccess=**false**;

**try** {

        ftpClient.connect(Cjip);

        isSuccess= ftpClient.login(CjuserName,PwdCj);

    } **catch** (IOException errmsg) {

        logger.error("FTP服务器连接失败",errmsg);

    }

**return**  isSuccess;

}

}

5.1.4React&Component

React是一个用户界面的UI构建框架(库)，它专注于MVC中的View层的渲染，所以React经常还伴随React-Redux或者Mobx来搭配使用，他们用来管理数据状态，控制React什么时候该重新渲染视图了。一个React组件的编写基本代码：

Import React from “react”

Export **default** Class MyComponent Extends React.Component{

Constructor(props){

//构造器，接收props，即父组件传递过来的参数

Super(props);

This.state={

Data:”数据”

}

//state中存放render()方法返回的jsx中需要被渲染或者有状态的数据

}

ComponentWillDidMount(){

Cosnole.log(“我被初始化了”)

//生命周期函数，对应还有组件销毁，组件实例化等

}

Render(){//渲染函数，返回的jsx内容会被编译成VirtualDom,最后浏览器渲染成真实DOM对象

Return(

<div>**this** is jsx{{**this**.state.Data}}</div>

)

}

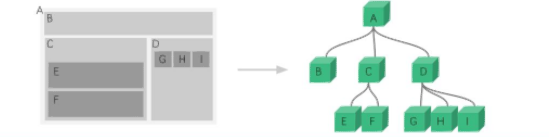
}

Export default 为Javascript模块化语法，在其他js脚本中可以使用import XXX from “组件相对路径” 来导入该组件

Class 为es6的声明类的语法，目前浏览器普遍支持的是ES5的语法，所以还要使用babel来对代码进行转换为ES5

MyComponent 组件名（即类名）

Extends React.Component继承自React.Component类



上图用代码表示为:

<A>

<B/>

<C><E/><F/></C>

<D><G/><H/><I/></D>

</A>

一个完整的应用就是组件与组件之间props传值，每个组件处理自己的逻辑，从而达到组件复用，多个组件拼接成一个应用或者页面。

5.1.5支付宝付款

整个支付过程就是用户挑战到付款界面，根据用户所选择的订单后台查询到该笔订单对应的商品数量级价格，向蚂蚁金服发送付款申请(即支付宝生成该笔订单的交易信息，通知支付宝该笔订单已生成，用户正准备付款)，使用AliPayAPI生成二维码文件上传到ftp服务器，并返回图片地址到前端页面，等待用户用支付宝扫一扫进行付款后，支付宝收到付款通知后向本系统服务器发送该笔订单已成功付款，成功接收到支付宝的通知后修改该订单的付款状态。

@RequestMapping(value = "pay.do",method = RequestMethod.POST)

    @ResponseBody

    public ServerResponse pay(HttpSession session, Long orderNo, HttpServletRequest httpServletRequest){

        User user= (User) session.getAttribute(Const.CURRENT\_USER);

**if**(user==**null**){

**return** ServerResponse.createByErrorCodeMessage(ResponseCode.NEED\_LOGIN.getCode(),ResponseCode.NEED\_LOGIN.getDesc());

        }

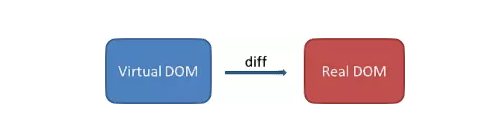
        String path=httpServletRequest.getSession().getServletContext().getRealPath("upload");

**return**  iOrderService.pay(user.getId(),orderNo,path);

    }

5.1.6Virtual DOM

浏览器端的DOM对象的操作是非常昂贵且耗费性能的，大量的DOM操作应用会变得非常卡顿，每次DOM改变浏览器都会重新绘制DOM，从而引出了Virtual DOM机制，其实大部分的DOM变化都是可以预见的，Virtual DOM与浏览器DOM形成一个映射，Virtual DOM改变时最终结果映射到真实DOM，用户触发数据变化时，Virtual DOM进行diff算法查找前后的Virtual DOM结构的变化并生成补丁，根据补丁渲染到浏览器的DOM中，例如：当用户改变组件内的state的值的时候，将张三改成了李四，Virtual DOM就会对比改变前和改变后的，并生成一个diff对象，告诉真实DOM又哪里需要变化，你要改变节点了。



浏览器的节点

<div name=“chenjie”class=”wrap”><p>我是真实节点</p></div>

使用Virtual DOM来表示：

\_dom={

uid:'asdasdasa4a5as74d5a4',

Type:div,

Attributes:[{attributeKey:’name’,attributeValue:’chenjie’},{attributeKey:’class’，attributeValue:’wrap’}],

Textnode:**null**,

Children:{

uid:'asdasdasa4a5as54564',

Type:p,

Attributes:[],

Textnode:’我是真实节点’，

Children:**null**

}

}

当用户改变节点内的内容时，例如把我是真实节点改了我是虚拟节点，此时生成一个diff对象，用来记录两次的差异性。

Diff={

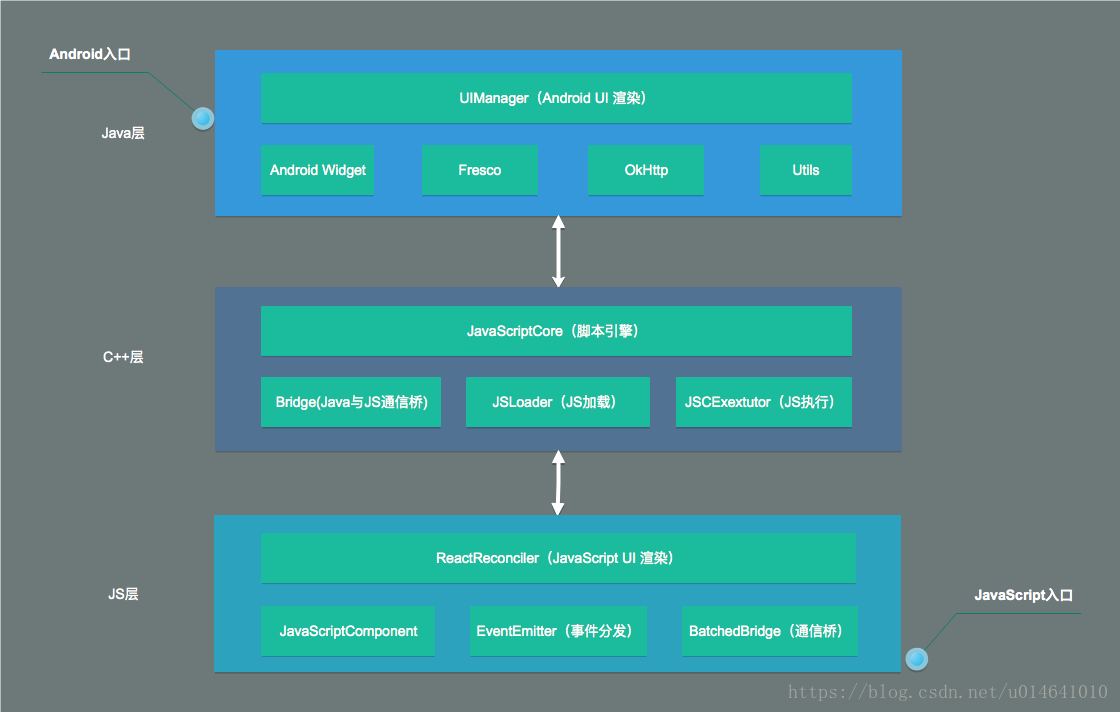
    changUId:asdasdasa4a5as54564’,

Value:”我是虚拟节点”

}

之后把改变的部分映射到浏览器DOM中。

5.1.7React-Native原理



（1）Java层：主要提供一个Android的渲染引擎(UIManager)将JavaScript映射为一些组件的功能，例如OKHttp:服务器获取数据，Fresco:Android图片加载等。

（2）C++层:主要封装了一个JavaScriptCore，用来解析js代码，开发者就可以在脱离浏览器引擎的情况下使用ES6，ES7的新语法特性，如：箭头函数，const，let等API，Bridge桥接了Java与js，从而实现js调用Java的函数，java调用js。基于C++的JSLoader主要用于加载一些静态文件和JavaScriptCore，JSCExextutor主要用与执行js文件。

（3）JS层：加载React框架，并执行对应代码，将Virtual DOM映射到java层，

Virtual DOM改变后生成diff通知java的UIManage绘制对应的视图。JavascriptComonent用于构建jsx构建的React组件。

（4）跨平台最主要的特性就是依赖Virtual DOM ，它将DOM脱离了浏览器，并搬到了内存中进行操作，因此可以在任何操作系统平台下根据Virtual DOM构建一套属于该平台规则的UI体系。

5.1.8 webpack

Webpack是一个web应用程序静态打包工具，他会将项目的所有依赖和模块化代码解析打包为一个或者多个bundle文件，基于nodeJS，所有的组件化模块化代码项目都需要经过webpack等其他打包工具打包为浏览器可运行的静态文件。Webpack还提供了许多的常用功能和工具，利用websocket通信实现代码热更新，解析sass，less代码，配合babel打包ES6和ES7等高级ECMASript定义的语法并将他们转换为ES5，即浏览器端可执行的代码，还可以处理css的方言，各种浏览器内核的一些方言-webkit- -moz-的写法并自动加上对应的前缀，在webpack的配置文件配置一个文件入口，webpack便会寻找与该文件有依赖的所有文件进行对应的处理，下面是前台门户的webpack开发配置文件。

'use strict'

const path = require('path')

const myutils = require('./utils')

const myconfig = require('../config')

const myvueLoaderConfig = require('./vue-loader.conf')

**function** myresolve (dir) {

**return** mypath.join(\_\_dirname, '..', dir)

}

module.exports = {

  context: path.resolve(\_\_dirname, '../'),

  entry: {

    app: './src/main.js'

  },

  output: {

    path: myconfig.build.assetsRoot,

    filename: '[name].js',

    publicPath: process.env.NODE\_ENV === 'production'

      ? myconfig.build.assetsPublicPath

      : myconfig.dev.assetsPublicPath

  },

  resolve: {

    extensions: ['.js', '.vue', '.json'],

    alias: {

      'vue$': 'vue/dist/vue.esm.js',

      '@': resolve('src'),

    }

  },

  module: {

    rules: [

      {

        test: /\.vue$/,

        loader: 'vue-loader',

        options: vueLoaderConfig

      },

      {

        test: /\.js$/,

        loader: 'babel-loader',

        include: [resolve('src'), resolve('test'), resolve('node\_modules/webpack-dev-server/client')]

      },

      {

        test: /\.(png|jpe?g|gif|svg)(\?.\*)?$/,

        loader: 'url-loader',

        options: {

          limit: 10000,

          name: utils.assetsPath('img/[name].[hash:7].[ext]')

        }

      },

      {

        test: /\.(mp4|webm|ogg|mp3|wav|flac|aac)(\?.\*)?$/,

        loader: 'url-loader',

        options: {

          limit: 10000,

          name: myutils.assetsPath('media/[name].[hash:7].[ext]')

        }

      },

      {

        test: /\.(woff2?|eot|ttf|otf)(\?.\*)?$/,

        loader: 'url-loader',

        options: {

          limit: 10000,

          name: myutils.assetsPath('fonts/[name].[hash:7].[ext]')

        }

      }

    ]

  }

  }

}

5.2 核心代码

5.2.1 Axios发起Ajax请求

使用Axios发起Ajax请求，处理异步操作。下列是一个获取指定分类下的子分类:

import axios from 'axios'

axios.post("/manage/category/getChildrenCategory.do?parentId="+this.parentId).then((res) => {

if(res.data.status==1){

this.$message.warning("该分类已经到最后啦，请到分类管理新增分类");

}else{

this.tableData=res.data.data;

}

}).catch((err) => {

});

5.2.2 React-Native跨平台样式

let style=StyleSheet.create({

    subwrap:{width:width\*0.9,height:150,borderBottomColor:'gray',

    borderBottomWidth:1,borderStyle:'solid',borderRadius:15,

    flexDirection:'row',alignItems:"center",justifyContent:"center"},

actradis :{width:20,height:20,borderRadius:10,backgroundColor:'red',borderStyle:'solid',borderWidth:1,borderColor:'white'},

norradis:{width:20,height:20,borderRadius:10,backgroundColor:'white',borderStyle:'solid',borderWidth:1,borderColor:'gray'}

}

# 6总结

整个商城系统从前端到后台，每个后端接口的搭建，每个页面的设计，从浏览器到手机APP，从前台门户系统到管理员后台系统再到订单管理系统，共计页面有三十多个，API接口有四十余个，三个系统共写代码四万多行，用时共计半年左右，大一下自己就励志做一名前端工程师，自学了两年多的前端，对前端有浓厚的兴趣和热爱，这期间也曾今想过放弃，特别是遇到bug的时候，周围也没有人带，都是自己一路凭借着热爱熬下去，课题加跨平台的时候自己本身也是有点慌的，深怕自己完成不了此次毕设，由于跨平台是最近一两年才兴起的技术，FaceBook2015年才开源，只有部分的北上广的互联网企业才会用到此类的新技术，而且新的技术的文档与知识很难获取，目前最新版本还是0.59，本身还是处于一个起草阶段，由于打包后的APP被渲染为原生安卓界面，所以其中JS与安卓的通信的性能消耗，UI渲染时的样式差异性都会遇到一定的问题，途中要查询许多文档API，遇到问题时国内的搜索引擎几乎找不到解决方案，只有上github或者Stack Overflow等国外的网站找解决方案。React也不好学，写多了就感觉他不是一门框架，自由度极高，正是因为其极高的自由度，所以很多功能不能像是其他前端框架一样使用框架提供的API来处理问题，要你自己写基于React语法的代码来处理问题，其中遇到了许多问题，像是：

1. 长列表的性能优化，商品列表数量几百上千时，APP开始显得卡顿。
2. 在重绘一个几乎没有什么变化的页面时，JS 帧率严重降低。
3. 前端基于React-Native封装的fetch异步获取后台数据的API与Java后台spring MVC数据绑定无法传递参数。
4. React的状态管理Mobx使用时，不能很好的体会状态的意思，action触发状态改变，最后全局更新数据状态。
5. 支付宝付款时，用户已付款系统总是接收不到支付成功的官方回调，接收了总是提示404错误。

遇到的问题好多好多，长列表性能优化时，发现React-Native自带的组件flatList可以高效的渲染长列表，只要按照文档内配置好参数即可。重绘页面时的卡顿现象使用React的API(shouldComponentUpdate)函数来指明在什么样的确切条件下你希望这个组件得到重绘。前后端传值问题，为了尽量减少后端代码的修改，把传参全放在了url后,这样后台才能使用数据绑定接收到。关于Mobx状态，MVVM的框架都会有一个全局状态管理的概念，由于业务逻辑开始复杂起来，父子组件普通的props传值开始跨多层，而基于React或者Vue的单向数据流的思想，是不允许跨多层组件传值的，所以要一层一层的往下传输，极大的增加了代码量和性能问题，所以应该要有一个东西来管理一些需要全局跨组件变化的数据，例如：用户信息，购物车数量，用户是否登录，在一个页面变动了其他页面也会自动更新状态。关于支付宝的回调，蚂蚁金服开发者中心配置处有一个回调地址，必须要在回调地址中输入后台系统的公网地址才能成功的接收到回调信息。

通过此次毕设，自己的技术能力和业务能力有了很大的提升，解决问题的能力也变高了，遇到技术问题总能自己解决，单位的项目也能自己独立负责，也经常帮别的部门解决bug，所以公司其他部门同职位的实习生有一两个也是我在带了，总结下来大概就是遇到问题应该不骄不躁，努力找到解决方案，实在不行就换个角度看待问题，不要总在一棵树上吊死。

# 致谢

首先，我要感谢带我的指导老师陈丽晖老师，感谢她的认真负责，热心的帮我们解决了很多问题，在整个论文方面，由于经验的匮乏，在设计初期和系统实现中难免会有许多问题考虑的不周到，从发现问题到解决问题，陈老师也给了我们很多良好的建议，耐心的帮我们讲解。突然发现已经过去了一学期了，上学期边实习边写毕设和论文，不知不觉毕设就写完了，后面用自己所学的也加了一些模块，似乎上学期一整个学期都充满着忙碌和紧张。

# 参考文献

[1]刘国钧，陈绍业，王凤翥.图书馆目录[M].北京：高等教育出版社.

[2]辛希孟，信息技术与信息服务国际研讨会论文集：A集[C].北京：中国社会科学出版社，1994.

[3]何龄修.读顾城《南明史》[J].中国史研究，1998.

[4]金显贺，王昌长，王忠东，等.一种用于在线检测局部放电的数字滤波技术[J].清华大学学报（自然科学版）.

[5]李晓玲 电子商城的设计与实现

[6] 蔡振凡 基于Java EE的跨境电商平台的设计与实现.

[7] 解晓园 基于J2EE技术的微商城系统中前台模块的设计与实现。

[8]陈小龙　基于JavaEE技术的网上商城管理系统的设计与实现

[9]何奔 基于WebComponents的跨终端商城系统的设计与实现

[10]杨文军 天龙时代网上电子商城的设计与实现

[11]Jack Yang The Machine Learning And The Neural Network

[12]Eason The Relationship and Distinction Between Big Data and Data Mining