



**本科毕业设计（论文）**

**基于node.js的移动端购物商城的设计与实现**

**学 院** **应用数学学院**

**专 业 信息与计算科学**

**（信息安全方向)**

**年级班别 2014级（2）班**

**学 号 3114008254**

**学生姓名 方彦凯**

**指导教师**  **蔡前凤**

**2018 年6月**

**基于**

**n**

**o**

**d**

**e**

**.**

**j**

**s**

**的移动端购物商城的设计与实现**

**方彦凯**

**应用数学学院**

**摘 要**

1999年，35岁的阿里巴巴集团创始人马云正式辞去公职，创办阿里巴巴网站。在现如今这个物欲横流的时代，信息量爆炸性增长的时代，马云的淘宝成为了从用户到用户的一个购物平台。现如今人们开店可以不用店面，逛街可以不用上街，打开智能手机，触摸几下屏幕就可以形成一笔交易。

移动端购物商城是针对大众开发的一款购物软件。截止至2017年11月，中国手机用户突破14亿，人均使用流量2.25G。易携带的智能手机相比于体积较大的电脑，还是手机更方便于我们的网上购物，因此该购物商城是针对于移动端而设计。

本商城使用HTML+jQuery+css开发，采用目前相对比较主流的MVC Web应用程序的模式。使用MongoDB作为数据库。界面采用Bootstrap前端框架，使界面看起来简洁大方而又不失美观。后台采用的是Node.js中的Express框架对数据库的数据进行操作。最后采用jQuery处理用户交互的部分。包含的用户模块有用户注册模块（用来注册新的用户，而且只能注册普通用户，同时采用DES加密技术对输入的密码进行加密处理），用户登录模块（用户分为管理员和普通用户），添加商品的详细数据模块（管理员进行商品的添加，不能重复添加同一件商品），查询并修改商品的详细数据模块（管理员查询需要修改的商品数据，然后进行修改），查询并删除商品的详细数据模块（管理员查询需要删除的商品数据进行删除），物流管理模块（管理员对用户已下单的商品的物流状态进行更改），修改个人的详细信息模块（普通用户对自己的个人信息进行修改），查询有意向的商品模块（普通用户查询自己有意向购买的商品），查看有意向的商品详情模块（普通用户可以查看商品的详情），购买商品模块（普通用户可以对商品进行购买操作）等模块。

**关键词：**移动端，购物商城，加密，Node.js

**Abstract**

In 1999, Jack Ma, the 35-year-old founder of Alibaba group, officially quit his job and started Alibaba.com. In this materialistic age,with the growth of information Ma's Taobao has become a shopping platform from users to users. Nowadays, people can open stores without storefront, go shopping without going to the streets, open smart phones, and touch screens to form a transaction.

Mobile terminal shopping is a shopping software for public development. Until November 2017, China’s mobile phone users exceeded 1.4 billion, and the per capita use of traffic was 2.25G.. Compared with larger computers or mobile phones, mobile phones are more convenient for our online shopping. Therefore, shopping mall is designed for mobile terminals.

This mall uses HTML+jQuery+ CSS development to adopt the current relatively mainstream MVC Web application model. Use MongoDB as the database. The interface adopts the Bootstrap front-end frame to make the interface look simple and easy and beautiful. In the background, the Express frame in node.js is used to operate the data of the database. Finally, jQuery handles the user interaction part. The included user module has a user registration module (used to register a new user, and can only register ordinary users, at the same time, using DES encryption technology to encrypt the input password), a user login module (users are divided into administrators and ordinary users), and a detailed data module for adding goods (manager's goods Add, can not repeat the same product), query and modify the product detailed data module (administrator query the need to modify the product data, and then modify), query and delete the product detailed data module (administrator need to delete the product data To delete), the logistics management module (the administrator changes the logistics status of the goods the user has placed), modify the personal detailed information module (ordinary users modify their own personal information), and inquire about the intent of the goods module (general The user inquires about the goods he intends to purchase), looks at the modules of the intentional merchandise details (the general user can view the details of the merchandise), purchase the merchandise module (the ordinary user can perform the merchandise purchase operation) and other modules.

**Key words:** Mobile terminal, Shopping mall, Encryption, Node.js.

目录

[1 绪论 1](#_Toc514702221)

[1.1开发背景 1](#_Toc514702222)

[1.2 可行性分析 1](#_Toc514702223)

[1.2.1 操作可行性 1](#_Toc514702224)

[1.2.2 技术可行性 1](#_Toc514702225)

[1.2.3经济上的可行性 2](#_Toc514702226)

[1.3 目标 2](#_Toc514702227)

[2 系统开发相关技术的简介 3](#_Toc514702228)

[2.1 jQuery 3](#_Toc514702229)

[2.2 MongoDB数据库 3](#_Toc514702230)

[2.3 Node.js 3](#_Toc514702231)

[2.4 Express 4](#_Toc514702232)

[3 系统分析 5](#_Toc514702233)

[3.1 系统需求分析 5](#_Toc514702234)

[3.1.1 确认用户类型 5](#_Toc514702235)

[3.1.2 模块功能分析 5](#_Toc514702236)

[3.1.3 业务逻辑分析 7](#_Toc514702237)

[3.1.4 系统总体分析 9](#_Toc514702238)

[3.2 性能需求分析 9](#_Toc514702239)

[4 系统设计 10](#_Toc514702240)

[4.1 总体设计 10](#_Toc514702241)

[4.2 详细设计 10](#_Toc514702242)

[4.2.1 登录模块 10](#_Toc514702243)

[4.2.2 注册模块 11](#_Toc514702244)

[4.2.3 消费者购买模块 11](#_Toc514702245)

[4.2.4 消费者更改信息模块 11](#_Toc514702246)

[4.2.5 消费者查询信息模块 11](#_Toc514702247)

[4.2.6 系统管理员模块 12](#_Toc514702248)

[4.3 数据库设计 12](#_Toc514702249)

[4.3.1 实体设计 12](#_Toc514702250)

[4.3.2 数据库总体关系模型设计 13](#_Toc514702251)

[4.3.3 数据库表结构设计 13](#_Toc514702252)

[5 系统实现 16](#_Toc514702253)

[5.1 项目目录一览 16](#_Toc514702254)

[5.2 app.js入口文件 17](#_Toc514702255)

[5.2.1 模块的导入 17](#_Toc514702256)

[5.2.2 模块的划分 19](#_Toc514702257)

[5.3 首页功能的实现 20](#_Toc514702258)

[5.4 登录注册功能的实现 22](#_Toc514702259)

[5.4.1 注册功能的实现 22](#_Toc514702260)

[5.4.2 登录功能的实现 24](#_Toc514702261)

[5.4.3 退出登录功能的实现 26](#_Toc514702262)

[5.5 个人信息查看和设置功能的实现 26](#_Toc514702263)

[5.5.1 图片上传功能的实现 27](#_Toc514702264)

[5.5.2 个人信息设置功能的实现 29](#_Toc514702265)

[5.6 商品查找功能的实现 30](#_Toc514702266)

[5.6.1 查找功能的实现 30](#_Toc514702267)

[5.7 添加至购物车功能的实现 33](#_Toc514702268)

[5.7.1 添加至购物车功能的实现 33](#_Toc514702269)

[5.7.2 更改购物车信息功能的实现 34](#_Toc514702270)

[5.8后台管理员功能的实现 35](#_Toc514702271)

[5.8.1 判断登录身份功能的实现 35](#_Toc514702272)

[5.8.2 分页功能的实现 36](#_Toc514702273)

[6 商城系统测试 38](#_Toc514702274)

[6.1 性能测试 38](#_Toc514702275)

[6.1.1 Cache-Control方法 40](#_Toc514702276)

[6.1.2 ETag方法 41](#_Toc514702277)

[6.2 系统测试 41](#_Toc514702278)

[6.3 安全测试 44](#_Toc514702279)

[结论 46](#_Toc514702280)

[参考文献 47](#_Toc514702281)

[致谢 48](#_Toc514702282)

**1 绪论**

**1.1开发背景**

2015年3月5日上午十二届全国人大三次会议上，李克强总理在政府工作报告中首次提出“互联网+”行动计划。将互联网与物质交换行为结合起来，就产生了电子商务。消费可以拉动内需，虽然这个社会物欲横流，但是快节奏的生活方式一直在压缩着人们休闲娱乐的时间，电子商务的产生，可以让人们足不出户就可以开店，购买自己所需要的商品。开发这样的一个商城，对于前期想开店却苦于没有投资资金租赁店面的人，闲时时间零散的人来说，何尝不是佳音。

**1.2 可行性分析**

1.2.1 操作可行性

在今年刚刚结束的两会中，可以总结出两条趋势，互联网经济模式是最前沿的趋势，实体店的生存压力急剧增大。传统的实体店需要店面，需要更多的人工，而移动端购物系统的开发，将会给所有的卖家和买家提供极大的便利，大大降低做生意的成本，随时随地可以做生意，下订单。移动端购物平台的需求日益迫切。

1.2.2 技术可行性

就拿1997年的ES1.0的出现来说，到现在已经是到达ES6版本了，进入21世纪，互联网普遍进入每个人的生活，对网络编程的要求是越来越高，各色各样的前端框架Angular、VUE发展迅速，无时无刻不是在进步的。由于大量开源框架出现，编写Web程序技术门槛变得更低。技术难点不再是出现在实现功能上，而是网站的负载和高并发上。管理系统主要体现其实用性和安全性，结合所给需求思考分析，在开源框架等技术下功能上实现上变得更为简单。

1.2.3经济上的可行性

对于移动商城的初期投资，在为了方便大家的同时，也是为了获得更高的经济效益。Node.js可以搭建服务器，MongoDB可以作为数据库，jQuery可以进行数据交互，这些框架都是开源的，免费的。与此同时，域名申请几十块钱就可以解决，域名空间也就百来块钱，与上线之后的经济效益相比，简直不值一提。因此，移动商城在经济上是可行的。

**1.3 目标**

客户端的语言使用HTML/CSS/jQuery，服务器端的语言使用Node.js。而数据库也将随之采用基于JavaScript的MongoDB数据库。

开发人员在编写代码的过程中要严格按照代码规范，对前端的HTML界面代码要注重语义化，对项目进行层次分明的模块化管理，进行项目的过程中要时刻牢记前端优化标准，提升用户浏览界面速度。

**2 系统开发相关技术的简介**

**2.1 jQuery**

jQuery是一个快速、简洁的JavaScript框架，是继Prototype之后又一个优秀的JavaScript代码库（或JavaScript框架）。jQuery设计的宗旨是“write Less，Do More”，即倡导写更少的代码，做更多的事情。它封装JavaScript常用的功能代码，提供一种简便的JavaScript设计模式，优化HTML文档操作、事件处理、动画设计和Ajax交互。

jQuery的核心特性可以总结为：具有独特的链式语法和短小清晰的多功能接口；具有高效灵活的css选择器，并且可对CSS选择器进行扩展；拥有便捷的插件扩展机制和丰富的插件。jQuery兼容各种主流浏览器，如IE 6.0+、FF 1.5+、Safari 2.0+、Opera 9.0+等[1]。

**2.2 MongoDB数据库**

MongoDB是一种面向文档的NoSQL数据库。 具体来说，每个数据都具有更新前后的字段，并且管理事务的状态。 然后，在提交之前，查询前一个数据; 提交后，后面的数据被查询。 通过这种方法，我们显示复数数据可以更新为具有指定隔离级别的事务[2]。

MongoDB是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。他支持的数据结构非常松散，是类似json的bson格式，因此可以存储比较复杂的数据类型。MongoDB最大的特点是他支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引[3]。

**2.3 Node.js**

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型，使其轻量又高效。Node.js 的包管理器 npm，是全球最大的开源库生态系统[4]。Node.js提供客户端 - 服务器开发集成，帮助web应用程序重用代码，并且是开发快速，可扩展网络应用程序的完美工具[5]。

**2.4 Express**

Express 是一个基于 Node.js 平台的极简、灵活的 web 应用开发框架，它丰富的 HTTP 快捷方法和任意排列组合的 Connect 中间件，让你创建健壮、友好的 API 变得既快速又简单。Express 不对 Node.js 已有的特性进行二次抽象，只是在它之上扩展了 Web 应用所需的基本功能[6]。

**3 系统分析**

**3.1 系统需求分析**

3.1.1 确认用户类型

首先确认使用对象的类型，确定系统的使用对象类型,是需求分析阶段的任务之一,也是分析和划分系统功能的重要根据。不同的身份对系统来说有不同的操作权限。此系统用户分类为：

普通消费者：可对商品进行搜索，查看，购买，评价，收藏；

商品信息管理员：可对商品信息进行录入，查找，删除，处理订单。

3.1.2 模块功能分析

我们一般使用用例图进行模块功能分析，用例图表明系统做什么，与谁交互。用例是系统提供的功能，参与者是系统与谁交互,参与者可以是人、系统或其他实体。一个系统可以创建一个或多个用例图。活动图可以在分析系统业务时用来淙业务流，也可以在收集系统需求的时候显示一个用例中的事件流。活动图显示了系统中某个业务或者某个用例中，要经历哪些活动，这些活动按什么顺序发生。

登录模块：登录是在登录框输入用户的账号和密码，通过调用/api/user/login接口，去数据库查找是否已经注册该账号，如果存在该账号，会根据不同的身份跳转至不同的界面。同时也对为登录直接输入跨身份界面地址的行为进行监测和限制，防止出现普通消费者也可以进入管理员管理信息界面这种不符合逻辑的情况的发生。

注册模块：注册则是在注册界面输入要注册的用户的信息，包括昵称，手机号，密码，确认密码。后续的信息（真实姓名，性别，地址等信息在注册之后再进行设置）。管理员身份的用户是由后台直接提供，而普通用户是直接在注册界面进行注册的。

普通消费者模块：消费者登录后能查看个人信息以及设置直接的个人信息，还可以在主商品页面浏览商品，在搜索栏搜索商品，如果有中意的商品，消费者可以对其进行商品收藏，商铺收藏，或者是添加到购物车，然后去购物车进行最后的下单决定，可以删除掉购物车一些不想买的商品，下单之后，当物流状态变为已签收，则可以对商品进行评价。

图 3.1 普通消费者功能UML图

管理员操作模块：管理员登录直接进入管理模块，对商品信息进行录入，查找，更改，和删除，同时也可以处理用户的订单情况，注销用户账户。

图 3.2 管理员功能UML图

3.1.3 业务逻辑分析

普通消费者购物流程，消费者登录账号，然后页面跳转到个人信息界面，首先修改个人信息（头像，真实姓名，地址等），然后跳转进主页面浏览商品，没有查询到主页面推荐的商品可以通过查询功能进行查找，假设终于找到中意的商品，则点击商品，进入详情页，可以收藏该商品，也可以关注店铺，同时，更可以直接下单，接下来订单的处理是管理员的工作了，直到订单的状态为已签收，则消费者可以对该商铺进行评分。

开始

消费者登录

设置个人信息

浏览商品

是否有中意商品

是

否

购买商品

收藏商品

关注店铺

查找商品

查看订单

评价

图 3.3 消费者购物流程图

管理员使用流程，首先通过后台给定的管理员身份，通过登录进入后台管理系统，首先先创建一个商品，如果这个商品的信息在保存之后发现该商品已经不再售卖了，或者是信息录入错误了，可以查询这个商品，然后对商品进行删除，更改，在消费者下单之后，管理员可以看到是那个消费者下单的，并控制订单状态，最后，管理员还可以注销普通用户的账号。

开始

管理员登录

处理订单

录入商品

是否需要更改或删除

是

否

查找商品

删除商品

更改商品信息

注销用户

图 3.4 管理员管理信息流程图

3.1.4 系统总体分析

通过以上模块及流程分析，可得总体设计如图3.5。

登陆

注册

未注册

已注册

用户

管理员

查看商品

选择商品

进入购物车

下单

查看修改个人信息

查看订单信息

查看收藏的商家或商品

添加商品

查找商品

删除商品

处理订单

注销用户

图 3.5 系统总体流程

**3.2 性能需求分析**

实时性：消费者的注册，信息更改，头像上传都是存入数据库的操作，需实时响应是否成功。消费者和管理员的查询、修改操作，都需要系统响应时间足够迅速，能满足用户需求，以免产生不好的体验，违背前端优化的宗旨。

可扩展性：本系统采用模块化程序设计的方法，结构清晰，通俗易懂，既便于程序添加、修改功能，又便于未参与开发的技术人员维护。

可操作性：操作界面简单明了，流程明确，易于操作，对输入格式和数据类型有严格限制，采用错误提醒机制，提示用户输入正确数据。

**4 系统设计**

**4.1 总体设计**

本系统采用的是典型的MVC分层结构，分别为视图层、控制层、业务逻辑层、数据层，如图4.1所示。

业务逻辑层

视图层

控制层

数据层

HTML

jQuery

Node.js

MongoDB

图 4.1 系统结构图

1. 视图层中，系统主要采用了HTML和CSS技术，HTML搭建DOM树，CSS渲染HTML界面，将数据层传递过来的数据反馈给前台页面。
2. 业务逻辑层，主要采用jQuery，来实现具体系统的业务逻辑处理。
3. 控制层主要是为了控制协调系统中各模块的执行流程，在控制层中，由Node.js中的路由模块的app.use中间件来控制系统流程转向。
4. 数据层存放系统中的数据，本系统采用MongoDB数据库。

**4.2 详细设计**

4.2.1 登录模块

1. 登录成功需正确输入用户名、密码。
2. 登录者的身份分为两种，一类是普通消费者，一类是管理员，由后台直接提供。
3. 根据不同登录身份登录成功后将用户信息保存至服务器，并且能跳转到相应的页面。
4. 对登录提交表单输入作必要的限制。使用正则表达式限制输入格式内容，防止此项XSS攻击，并且当表单为空的时候，会提示表单为空。

4.2.2 注册模块

需要以正确的格式并且全部填满才能注册，注册的用户身份只能是普通消费者。

4.2.3 消费者购买模块

1. 在首页浏览商品。
2. 在查找栏查找商品，可根据店铺名称，商品名称，商品类别进行搜索。
3. 点击查看商品详情，商品详情包括商品图，商品名称，店铺名称，商品描述，商品价格，商品库存，商品评分。
4. 收藏商品或者关注店铺。
5. 点击商品加入购物车。
6. 去购物车下单，下单之后商品状态变为已下单。

4.2.4 消费者更改信息模块

1. 消费者登录之后，点击头像进入设置个人信息模式，填完基本信息。

4.2.5 消费者查询信息模块

1. 消费者可以通过点击头像查看个人信息。
2. 消费者查看收藏的商品和关注的店铺。
3. 消费者查看已支付和未支付的商品，已支付的商品有物流信息。
4. 当物流信息显示已签收的时候，消费者可以进行商品评价。

4.2.6 系统管理员模块

1. 管理员查找、修改、添加、删除商品信息。
2. 管理员更改消费者的物流信息。
3. 管理员注销普通用户账号。

**4.3 数据库设计**

数据库就是存放数据的仓库，即数据按照一定的规律存放在计算机中，是信息系统的核心和基础，是信息系统开发和建设的重要组成部分，把信息系统中大量的数据按一定的模型组织起来，提供存储、维护、检索数据的功能，使信息系统可以方便、及时、准确地从数据库中获得所需的信息。数据库设计是信息资源管理最有效的手段，数据库设计就是根据具体的应用环境，构造合理数据库模式，建立数据库逻辑结构，有效存储数据，满足用户实际需求和系统应用处理要求。在动态网站设计中数据库的重要性不言而喻。合理地设计数据库结构可以提高系统运行的效率。如果设计不当，查询起来就非常吃力，程序的性能也会受到影响[7]。

4.3.1 实体设计

根据系统设计和业务逻辑分析，本系统定义的数据库包含以下五个表：

用户信息表：用于保存用户的昵称，头像信息，密码，手机号码，真实姓名，性别，收货地址，是否是管理员等信息。

商品表：用于保存店铺名称，店铺ID，商品名称，商品ID，商品类别，价格，描述，库存，图片地址，商品评分等信息。

消费者购物车信息表：用于保存消费者的用户名和放入购物车的商品对应的店铺名称列表字符串，商品名称列表字符串，放入购物车商品数量列表字符串，放入购物车的商品是否支付的列表字符串和物流状态列表字符串等。

消费者收藏商品信息表：用于保存消费者的用户名，收藏的商品对应的店铺名称列表字符串和收藏的商品名称列表字符串。

消费者关注店铺信息表：用于保存消费者的用户名，关注的店铺名称列表字符串。

4.3.2 数据库总体关系模型设计

结合实体设计以及模块设计，可得本系统的实体关系ER图，如图4.2所示。

1

消费者

查询

商品

n

1

管理员

查询

商品

n

改/删

1

1

购买

1

1

修改

个人信息

修改

物流信息

1

1

1

1

删

1

n

图 4.2 系统实体关系ER图

4.3.3 数据库表结构设计

从实体设计分析，数据库中的主要表结构设计如下：

**表4.1 用户信息表t\_** **user**

|  |  |  |  |
| --- | --- | --- | --- |
| 主键 | 字段名 | 类型 | 说明 |
| key | \_id | String | 用户的身份id，数据库分配 |
|  | Username | String | 用户名称 |
|  | password | String | 登录密码 |
|  | sex | String | 性别：男，女，未知（默认） |
|  | Realname | String | 真实姓名 |
|  | Mobilephone | Number | 联系电话 |
|  | Address | String | 地址 |
|  | picSrc | String | 用户头像存放地址 |
|  | IsAdmin | Boolean | 是否为管理员 |

**表4.2 商品表t\_good**

|  |  |  |  |
| --- | --- | --- | --- |
| 主键 | 字段名 | 类型 | 说明 |
| key | \_id | String | 商品表的id（数据库自动分配） |
|  | storeName | String | 店铺名 |
|  | storeNum | String | 店铺序列号 |
|  | goodName | String | 商品名称 |
|  | goodID | String | 商品序列号 |
|  | goodSort | String | 商品类别 |
|  | price | Number | 商品价格 |
|  | description | String | 商品描述 |
|  | repertory | Number | 库存 |
|  | goodpicSrc | String | 图片地址 |
|  | score | Number | 商品评分 |

**表4.3 消费者购物车信息表t\_cart**

|  |  |  |  |
| --- | --- | --- | --- |
| 主键 | 字段名 | 类型 | 说明 |
| key | \_id | String | 消费者购物车列表id |
|  | userName | String | 用户名称 |
|  | storeNameList | String | 放入购物车的商品对应的店铺名称列表字符串 |
|  | goodList | String | 商品名称列表字符串 |
|  | goodNumList | String | 放入购物车商品数量列表字符串 |
|  | isBuyList | String | 放入购物车的商品是否支付的列表字符串 |
|  | logisticsList | String | 物流状态列表字符串 |

**表4.4 消费者收藏商品信息表t\_savegoodlist**

|  |  |  |  |
| --- | --- | --- | --- |
| 主键 | 字段名 | 类型 | 说明 |
| key | \_id | String | 购物信息列表id |
|  | username | String | 用户名称 |
|  | storeNameList | String | 收藏的商品对应的店铺名称列表字符串 |
|  | goodList | String | 收藏的商品名称列表字符串 |

**表4.5 消费者关注店铺信息表t\_savestorelist**

|  |  |  |  |
| --- | --- | --- | --- |
| 主键 | 字段名 | 类型 | 说明 |
| key | \_id | String | 物流信息列表id |
|  | userName | String | 用户名称 |
|  | storeNameList | String | 关注的店铺名称列表字符串 |

**5 系统实现**

**5.1 项目目录一览**

****

图5.1 项目目录一览(1)



图5.2 项目目录一览(2)

**5.2 app.js入口文件**

app.js文件是应用程序的启动入口文件，只要用户发送了一个请求，那么首先进入的就是app.js文件

5.2.1 模块的导入

首先必须加载项目所需要的express框架模块，swig的html模板模块，MongoDB数据库模块，还有验证登录所需要的cookies模块。

//加载express模块

var express=require('express');

//加载模板处理模块

var swig=require('swig');

//加载数据库模块

var mongoose=require('mongoose');

//加载body-parser用来处理post提交过来的数组

var bodyParser=require('body-parser');

//加载cookie模块

var Cookies=require('cookies');

//创建app应用=>NodeJS Http.createServer();

var app=express();

var Register=require('./models/registerModel');

由于在后续的项目编写当中，在html代码编写的时候，会经常出现图片，css样式表等静态文件的导入，如果每次都要使用静态文件的绝对地址的话，会大大印象开发效率，也经常会出错，所以我使用的是node.js提供的设置静态文件托管功能，使得在导入静态文件的时候，只需要相对路径就行了，前面的地址会自动补全。

//当用户访问的url以/public开始，那么直接访问对应\_\_dirname+'/public'下的文件

app.use('/public',express.static(\_\_dirname+'/public'));

接下来是配置应用模板，定义当前应用所使用的模板引擎，第一个参数表示模板引擎名称，同时而是模板文件的后缀。第二个参数表示用于解析处理模板内容的方法。

app.engine('html',swig.renderFile);

接下来是设置模板文件存放的目录，第一个参数必须是views，第二个参数是目录

app.set('views','./views');

注册所使用的模板引擎，第一个参数必须是view engine，第二个参数和之前app.engine这个方法中定义的模板引擎的名称（第一个参数）是一致的

app.set('view engine', 'html');

在开发过程中，需要取消模板缓存，不然每次调试都会对浏览器加载的资源进行缓存，导致修改后的代码无法显示运行效果。

swig.setDefaults({cache:false});

bodyparser设置,添加了这个方法之后自动在api.js里面的post方法中的req添加body属性

app.use(bodyParser.urlencoded({extended:true}));

5.2.2 模块的划分

在这个项目里面，我是根据不同的功能划分模块的，以主页面功能，用户信息及设置功能，查找功能，购物车功能，登陆注册功能，管理员的功能划分模块。

app.use('/',require('./routers/home'));

app.use('/user',require('./routers/user'));

app.use('/search',require('./routers/search'));

app.use('/cart',require('./routers/cart'));

app.use('/api',require('./routers/api'));

app.use('/manager',require('./routers/manager'));

在app.use()方法里面，第一个参数是给路由设置地址前缀，第二个参数是导入分路由文件，分路由文件定义了视图地址的后半部分。

在app.js里面，最后一部分代码是连接数据库，这样才能对数据库进行操作

mongoose.connect('mongodb://localhost:27017/blog',function(err){

if(err){

console.log('数据库连接失败!');

}else{

console.log('数据库连接成功!');

app.listen('4200');

}

});

一旦数据库连接成功，就开启一个4200端口，并且对这个端口进行监听。

但是在这之前，必须先开启数据库，然后在当前项目目录下输入命令行：

mongod --dbpath F:\MMPZ\db

后面的地址是数据库生成文件夹的地址。

**5.3 首页功能的实现**



图5.3 首页

系统的首页是显示每个不同类别（服装、日用品、化妆品、食品、家电、电子产品、医用品、书籍、文具）推荐的商品 ，侧边栏是一个导航栏，使用HTML里面<a>标签的锚点方法实现相应当前页面不同位置的跳转。



图5.4 商品详情界面（1）

点击商品即会弹出相应的详情弹框，这是通过bootstrap的弹框插件实现的，通过给每个商品a盒子模型设置相应参数[8]

<a class="box" data-toggle="modal" data-target="#myBuyModal">

data-toggle=”modal”表示这是一个控制弹框modal弹出的控制装置盒子，data-targeet属性设置的则是对应的目标弹框的id。

在这个视图里面，是多个a控制装置控制同一个弹框，那么弹框里面的数据是如何实现动态变化的呢？即每次点击不同商品如何在同一个弹框显示不同的内容？

$("#tuijiant1").attr("src",DailyNecessities[0].goodpicSrc);

$("#tuijiand1").text(DailyNecessities[0].description);

$("#tuijianp1").text(DailyNecessities[0].price);

$("#tuijian1data").text([DailyNecessities[0].goodpicSrc,DailyNecessities[0].goodName,DailyNecessities[0].storeName,DailyNecessities[0].price,DailyNecessities[0].description,DailyNecessities[0].repertory,DailyNecessities[0].score]);

这是这个过程实现的一部分js脚本代码，前面四句是给a控制装置动态插入图片地址，商品描述，商品价格信息，然后再创建一个display为none的span标签，里面的插入一个包含弹框需要的信息的数组，在点击弹框的时候，通过jquery框架的$(this)，实现对点击的元素的定位

var arr=$(this).find("span").eq(4).text().split(",");

$("#picture").attr('src',arr[0]);

$("#goodN").text(arr[1]);

$("#detail-price").text(arr[3]);

$("#desc").text(arr[4]);

$("#sName").text(arr[2]);

$("#reper").text(arr[5]);

$("#score").text('评分:'+arr[6]);

然后通过find()，eq()等过滤方法找出之前display为none的span元素[9]，获取到它的文本节点内容之后通过split()方法将数组字符串分成数组，将每个数组元素插入到弹框对应的位置里面，即可实现这个过程。

**5.4 登录注册功能的实现**

5.4.1 注册功能的实现

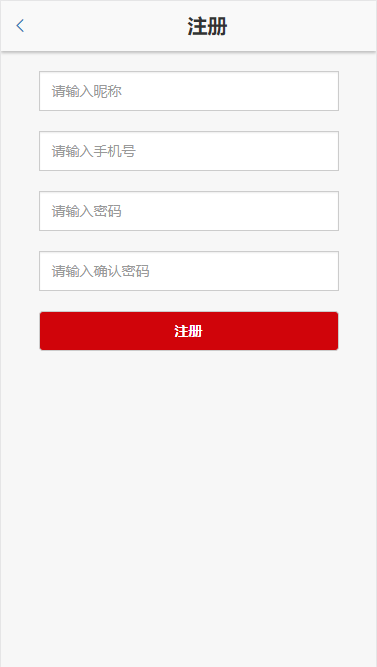


图5.5 注册界面

在注册视图输入昵称，手机号，密码和确认密码信息，确认密码是不存入数库的，它的功能是确保用户的密码输入是自己先输入的字符序列。填完信息点击注册按钮即调用接口。

用户注册的逻辑是1.所有的控件不能为空；2.判断两次输入的密码是否一致；3.判断用户名是否已经注册过了。在这个商城系统里面，所有的用户昵称是第一无二的。

//昵称是否已经被注册  
Register.findOne({  
 username:username  
}).then(**function**(RegisterInfo){  
 **if**(RegisterInfo){  
 //表示数据库中有该记录  
 responseData.code=7;  
 responseData.message="该昵称已被注册";  
 res.json(responseData);  
 **return**;  
 }  
 //保存用户注册的信息到数据库中,不需要直接操作数据库，通过操作对象来操作数据库  
 **var** register=**new** Register({  
 username:username,  
 mobilephone:mobilephone,  
 password:password  
 });  
 **return** register.save();  
}).then(**function** ( newRegisterInfo ) {  
 responseData.message="注册成功";  
 res.json(responseData);  
 **return**;  
});

通过MongoDB查询数据库的方法findOne查找registerDataBase数据库里面是否已经有userName，如果有则返回“该昵称已被注册”的提示，若未被注册则注册成功。then()是一个监听方法，监听操作数据库之后返回的数据。

5.4.2 登录功能的实现

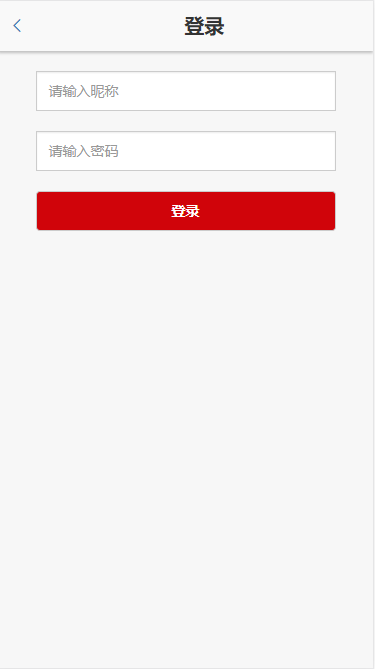


图5.6 登录界面

登录功能其实说白了很容易理解，就是在registerDataBase数据库里面通过userName和password查找相应的数据，如果存在以此条数据，则登录，如果不存在则返回“用户名或密码错误”的消息。但是只是这样判断的话，只能在这个登录视图实现登录状态，如何在其他视图也实现登录状态的判断呢？总不能每个视图的load事件触发的时候都调用一下登录接口吧！

在登录接口/api/user/login中，有这么一段代码。  
req.cookies.set('userinfo', JSON.stringify({  
 \_id: userInfo.\_id,  
 username: userInfo.username,  
 isAdmin: userInfo.isAdmin  
}));

这段代码的功能是，登录成功之后除了返回登录的信息给客户端之前，还要先发送一个cookie给客户端，然后保存在客户端中，在下后续的视图跳转中，就可以通过判断客户端是否存在userinfo这个cookie就可以判断是否处于登录状态了[10]。



图5.7 浏览器cookie图

每次实现视图跳转，都会执行入口文件app.js里面的一个中间件

app.use(**function**(req,res,next){  
 req.cookies=**new** Cookies(req,res);  
 //解析登录用户的cookie信息  
 req.userInfo={};  
 **if**(req.cookies.get('userinfo')){  
 **try**{  
 req.userInfo=JSON.parse(req.cookies.get('userinfo'));  
 Register.findById(req.userInfo.\_id).then(**function**(userInfo){  
 //在接收到请求之后，给cookies对象再添加一个isAdmin属性  
 req.userInfo.isAdmin=Boolean(userInfo.isAdmin);  
 next();  
 });  
 }**catch**(e){  
 next();  
 }  
 }**else**{  
 next();  
 }  
});

在执行到这个中间件的时候，做一下判断，如果存在userinfo的cookie则再给这个cookie对象添加isAdmin属性，表示是管理员身份还是普通用户的身份。

5.4.3 退出登录功能的实现

根据我们前一小节描述的通过浏览器是否存在cookie来判断登录状态，我们退出登录的时候只需要把userinfo的cookie清除掉，设置为null即可。

router.get('/user/logout',**function**(req,res,next){  
 req.cookies.set('userinfo',**null**);  
 res.json(responseData);  
});

**5.5 个人信息查看和设置功能的实现**

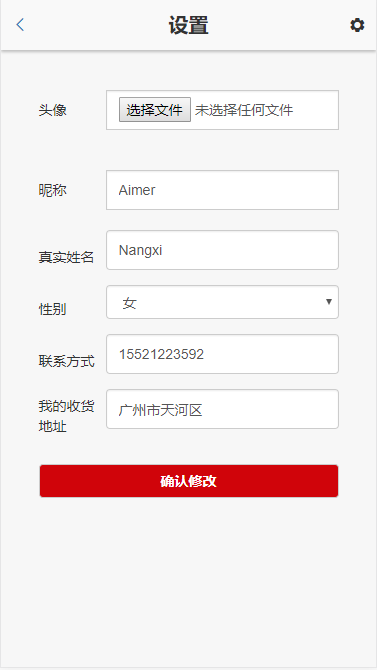


图5.8 个人信息设置界面

5.5.1 图片上传功能的实现

router.post('/set/pic',**function**(req,res,next){  
 **var** form = **new** formidable.IncomingForm(); //创建上传表单  
 form.encoding = 'utf-8'; //设置编辑  
 form.uploadDir = 'public' + AVATAR\_UPLOAD\_FOLDER; //设置上传目录  
 form.keepExtensions = **true**; //保留后缀  
 form.maxFieldsSize = 2 \* 1024 \* 1024; //文件大小  
  
 form.parse(req, **function** (err, fields, files) {  
 **if** (err) {  
 **return** res.json({  
 "code": 500,  
 "message": "内部服务器错误"  
 })  
 }  
 // 限制文件大小 单位默认字节 这里限制大小为2m  
 **if** (files.fulAvatar.size > form.maxFieldsSize) {  
 fs.unlink(files.fulAvatar.path)  
 **return** res.json({  
 "code": 401,  
 "message": "图片应小于2M"  
 })  
 }  
 **var** extName = ''; //后缀名  
 **switch** (files.fulAvatar.type) {  
 **case** 'image/pjpeg':  
 extName = 'jpg';  
 **break**;  
 **case** 'image/jpeg':  
 extName = 'jpg';  
 **break**;  
 **case** 'image/png':  
 extName = 'png';  
 **break**;  
 **case** 'image/x-png':  
 extName = 'png';  
 **break**;  
 }  
 **if** (extName.length == 0) {  
 **return** res.json({  
 "code": 404,  
 "message": "只支持png和jpg格式图片"  
 })  
 }  
 //使用第三方模块silly-datetime  
 **var** t = sd.format(**new** Date(), 'YYYYMMDDHHmmss');  
 //生成随机数  
 **var** ran = parseInt(Math.random() \* 8999 + 10000);  
 // 生成新图片名称  
 **var** avatarName = t + '\_' + ran + '.' + extName;  
 // 新图片路径  
 **var** newPath = form.uploadDir + avatarName;  
 // 更改名字和路径  
 fs.rename(files.fulAvatar.path, newPath, **function** (err) {  
 **if** (err) {  
 **return** res.json({  
 "code": 401,  
 "message": "图片上传失败"  
 })  
 }  
 **return** res.json({  
 "code": 200,  
 "message": "上传成功",  
 "result": AVATAR\_UPLOAD\_FOLDER + avatarName  
 })  
 })  
 });  
});

以下是上传图片需要引入的模块和定义的变量

var formidable = require('formidable');

var sd = require("silly-datetime");

var fs = require('fs');

var AVATAR\_UPLOAD\_FOLDER = '/avatar/'; // 上传图片存放路径，注意在本项目public文件夹下面新建avatar文件夹

这串代码其实最主要的是fs模块，通过它实现对图片的复制，通过formidable模块实现图片的转码，上传目录，后缀和大小的设置，最后是将上传的头像文件存放在public/avatar/文件夹里面。

5.5.2 个人信息设置功能的实现

我们在registerDataBase数据库中，定义的数据库模型是图片地址、真实姓名、地址、是否管理员、用户名、电话、密码，而在注册的时候我们只是注册了用户名、手机号、密码三个，其他的键都是先采用默认的字段，在设置个人信息的时候，采用更新数据库的方法而不是创建数据库的思路去解决。

Register.update({  
 \_id:req.userInfo.\_id  
},{  
 picSrc:picSrc,  
 username:username,  
 realname:realname,  
 sex:sex,  
 mobilephone:mobilephone,  
 address:address  
}).then(**function**(result){  
 res.json({  
 code:200,  
 message: "success"  
 });  
});

**5.6 商品查找功能的实现**

5.6.1 查找功能的实现



图5.9 商品查找界面

商品搜索是根据商品的店铺名，或者商品名称，或者商品类别来查找的，点击查找按钮之后实现视图跳转，跳转至查找结果视图，但是由于视图刷新会导致数据被全部初始化，之前在文本框中输入的文本被清空，没有记忆下来，这里采用了一种客户端存储的方法。

sessionStorage Web存储方法sessionStroage可以通过setItem()方法实现数据存储，其有效期和存储数据的脚本所在的最顶层的窗口或者是浏览器标签页是一样的。一旦窗口或者标签页被永久关闭了，那么所有通过sessionStorage存储的数据也都被删除，所以叫做会话级别的数据存储。[11]

sessionStorage.setItem("keyvalue",val);

设置一个keyvalue的键，val是获取到的文本框里面的字符串，先保存起来，在跳转视图的时候就可以使用了。

那么有个问题，为什么不使用localStroage呢？localStorage也是一个Web客户端存储方法，但是它的存储有效期是永久性的，不手动删除是永久存在的，正是因为这样，基于《雅虎军规》的前端优化策略，为了不给浏览器增加负担，所以还是采用sessionStroage，关闭窗口就删除了。

Goods.find({  
 $or:[{storeName:keyword},{goodName:keyword},{goodSort:keyword}]  
})

这是商品查询的接口，调用find()方法，$or连接符实现或查询。



图5.10 商品查找结果界面

在此视图刷新的时候，通过sessionStroage的getItem()方法获取前一页保存的keyvalue键值，然后进行搜索，这里不再详讲，弹框的实现方法也跟5.3章节的实现是一样的。

**5.7 添加至购物车功能的实现**

5.7.1 添加至购物车功能的实现



图5.11 商品详情界面（2）

选择需要的数量，然后点击添加至购物车。如果添加成功，则后台接口会返回”已添加”，然后通过text()方法渲染在红色按钮上面。 购物车的添加功能其实就是数据库的数据添加，但是这里比较特别的是，在设置carts数据库模型的时候，我为了使用户与商品形成一对多的关系，对商店名称、商品名称、商品数量、是否购买、物流状态采用列表字符串的存储方法。每条数据的用户名都是不重复的，也就是一个用户对应一条数据，在一条数据里面，我们来举个例子：



图5.12 carts数据库截图

上面的这条数据是一个用户名为”laojin”的用户添加商品至购物车的时候产生的，在浏览器获取到这条数据的时候，对后面的几个键值用split(“,”)进行处理，先处理成数组，在storeNameList数组中，第一个元素为苹果iphone旗舰店，这对应着goodList数组中的第一个元素iphone6，以此类推对应3，true和已签收。

5.7.2 更改购物车信息功能的实现

更改购物车包含三个小模块，一个是更改数据，一个是删除前面提到的数组中的每个对应元素，最后一个是清空购物车。



图5.13 购物车界面

首先我们先来说一下清空购物车，这个是最简单的，直接删除已登录的用户在carts数据库存储的数据就行了。  
router.post('/delete',**function**(req,res,next){  
 Carts.remove({  
 userName:req.body.userName  
 }).then(**function**(){  
 responseData.code=200;  
 responseData.message="删除成功";  
 res.json(responseData);  
 })  
});

更改数据和在购物车中删除商品其实都属于数据库的更改。

5.7.1节提到的数组字符串这种方法比较麻烦的是在更改用户的购物车状态的时候，如果用户要修改之前添加到购物车的iphone6的购买数量，那么得依次循环数组storeNameList和goodList，找到对应的商品的数组元素下标，然后更改其他数组对应下标的元素信息，最后保存到数据库的时候通过join(“,”)将数组转化为字符串。如果是删除用户已经添加在购物车的商品的话，则移除对应下标的元素，然后在更新的时候用join(“,”) 将数组转化为字符串。因为一个用户在carts数据库有且仅有一条数据存在，所以在更改数据库信息的时候，必须向遍历一下数据库，如果数据库存在该用户的数据，则isExit为true，否则为false。

**5.8后台管理员功能的实现**

5.8.1 判断登录身份功能的实现

如何在登录的时候判断身份呢？这个就与之前登录的时候，后台给浏览器发送的名为userinfo的cookie有关了。

在数据库中，每注册一个用户，都会给用户数据添加一个isAdmin属性，默认为false，而之前我们经常说的手动给一个管理员的账号，是在数据库中手动修改admin账号的isAdmin属性为true。数据库中isAdmin的默认值设定代码如下。

isAdmin:{

type:Boolean,

default:false

}

因为在每次的页面跳转时，都会经过app.js文件里面的app.use中间件，所以在接收到请求之后，必须给cookies对象再添加一个isAdmin属性

req.userInfo.isAdmin=Boolean(userInfo.isAdmin);

接下来我们用另一个身份（也就是管理员身份），进行登录，登录进去之后，我们根据上面的系统分析可以知道，首先要在“添加数据”栏目里面，添加商品的数据，商品的所属店铺和商品名称是唯一的，这句话的意思是说，在数据库里面，不可能存在两组数据使得店铺名称和商品名称同时相同。这样就保证了不同的店铺可以卖相同的产品，同一个店铺里面不可能再注册相同的产品。

因为商品添加的功能接口实现思路跟之前的添加差不多，这里就不再一一讲述。

5.8.2 分页功能的实现

后台管理部分有个重要需要讲的点，就是分页的功能。



图5.14 管理员查询结果界面

这个查询结果的分页功能是以每五条数据为一页。MongoDB提供了limit()方法和skip()方法来实现这个分页功能。其中，limit(data)方法中data是显示每页的数据条数，也就是5，而skip(data)中的参数是指跳过的数据条数，数据库中第一条数据为基准。我们来查找一下规律：

第一页显示5条数据，也就是数据库中的第1-5条数据。

1-5 skip：0

6-10 skip：5

11-15 skip：10

以此类推我们看出规律为：跳过的数据条数skip=(当前页数-1)\*limit

//计算总页数

pages=Math.ceil(result.length/limit);

//取值不超过pages

page=Math.min(page,pages);

//取值不小于pages

page=Math.max(page,1);

var skip=(page-1)\*limit;

Goods.find({

$or:[{storeName:keyword},{goodName:keyword},{goodSort:keyword}]

}).limit(limit).skip(skip).then(function(goodInfo){

if(goodInfo){

responseData.message="查询成功";

responseData.goodInfo=goodInfo;

res.json(responseData);

}else{

responseData.code=404;

responseData.message="未查询到数据";

res.json(responseData);

}

return;

});

管理员在后台的查询也是根据已注册的商品的店铺名称或商品名称或商品类别来查询的。

至于后台管理的其他功能例如用户注销，商品信息更新，物流状态更新，都与前面的一些功能重复，这里就不再讲述。

**6 商城系统测试**

我们在浏览网站时，在浏览的过程中，有时会出现错误代码提示，或许是连接不上，或者是找不到所要打开的网页，这就是一种网站不稳定的情况，出现这种情况的可能性非常多，比如网站的内测没有做好，又或者网站访问人数过多，又或者网站源代码出现错误，这些都会引起我们在浏览网页时，出现各种各样的问题。通常在建设完成一个网站之后，作为设计开发者，都会对网站进行内测，对于网站系统内测就是指，将网站的所有模块全部进行不断的测试，重复对一个模块不间断的测试，不停的寻找在搭建过程中可能遗漏的地方，只有经过不间断的测试，才能保证网站系统没有大的漏洞。网站搭建完成之后，并不会只是进行内测，还会对网站进行整体的测试，而对于整体测试的要求就要高许多，要测试网站在访问量非常大的情况下的稳定性，还要测试网站对于各种浏览器的兼容性，要知道现在的浏览器各种各样，而每个用户的使用习惯是不一样的，因此要保证网站对于浏览器的兼容性一定要好。但是我的这个项目是基于iphone6来开发的，浏览器的兼容问题可以不用考虑。有时候在网站经过内测以及整体的测试之后，为了保证网站运营的正常，也还会进行一段时间的试运行，网站进入试运行之后，对于真实用户的反应就多了许多，这样也就更加能够保障网站在运营过程中不会出现更多的问题。

可以说网站测试对于网站的稳定性非常的重要，只有保障网站的稳定性，才能吸引更多的用户前来浏览，让更多的用户愿意浏览。

**6.1 性能测试**

检测一个网站的性能，其实从一个方面看就足够了——页面的请求时间，请求时间如果过于长的话，会导致用户体验下降，自然也就意味着性能下降了，在下面这张图中，我截取的是加载资源最多的界面（主界面）的请求过程图。

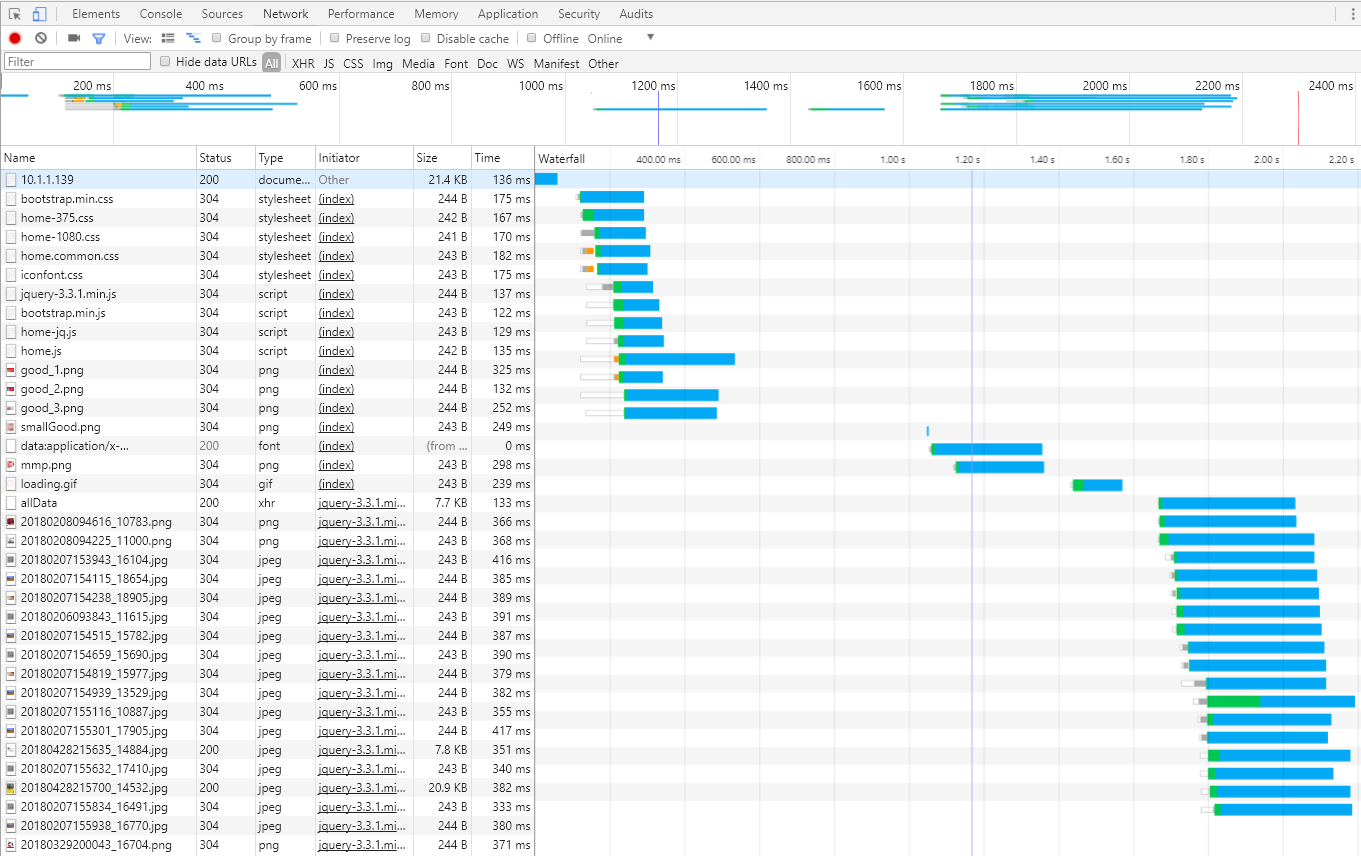


图 6.1 性能测试图

为了测试静态资源是否从缓存加载，我特地修改了两样商品的图片，当主页重新刷新时，更改的两张图片就可以看到效果了。从这张图可以看出，总共的请求时间是2.39s，主页所需的css样式表，js脚本等静态资源请求状态码都为304，表示服务器的资源没有发生改变，这些资源都是直接从浏览器缓存获取的，同理，没有更改过的图片返回的状态码也是304，而更改过的图片返回的状态码为200，也就是从服务器重新请求了。

缓存已被证明是一种有效的减少缓存泄漏能量的技术。然而，目前缓存控制策略通常遭受性能损失，并且不适应不同的程序行为[12]。本商城系统对于是否使用本地缓存采用的是Cache-Control方法和ETag方法，使用这两种方法之后，当进行第二次请求没资源加载的时间就缩短为490ms了，大大提升了访问速度。



图 6.2 第一次请求图

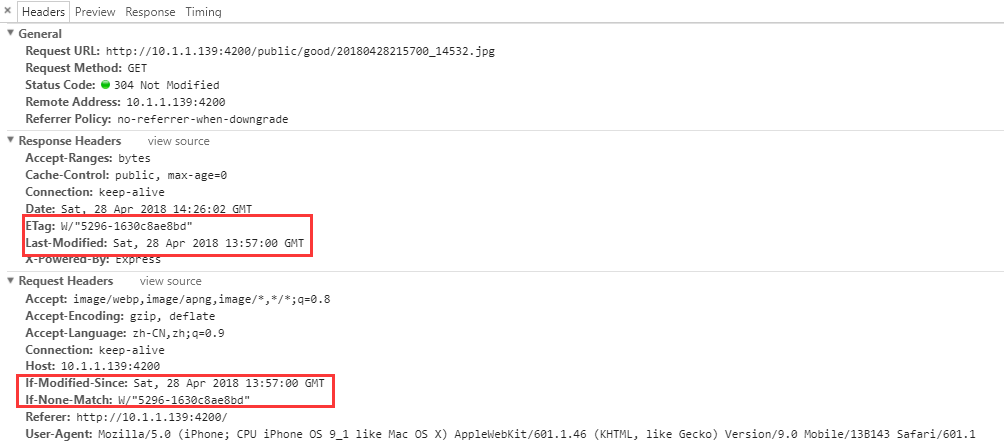


图 6.3 第二次请求图

6.1.1 Cache-Control方法

在浏览器第一次请求某一个URL时，服务器端的返回状态会是200，内容是你请求的资源，同时有一个Last-Modified的属性标记此文件在服务期端最后被修改的时间，我们从上面那张图可以看出，该资源在服务器最后修改的时间是Sat, 28 Apr 2018 13:57:00 GMT。客户端第二次请求此URL时，根据 HTTP 协议的规定，浏览器会向服务器传送 If-Modified-Since 报头，询问该时间之后文件是否有被修改过，上图的If-Modified-Since是Sat, 28 Apr 2018 13:57:00 GMT，该时间与Last-Modified进行比较，发现在If-Modified-Since之后，服务器的数据没有修改过了，自然而然返回的状态码为304。

6.1.2 ETag方法

ETag，全称为Entity Tag，意思是实体标签，从名字上看，是对于某种实体的一个标识。它属于HTTP协议的一部分，也就是所有的Web服务器都应该（也确实能）支持这个特性。它的作用是用一个特殊的字符串来标识某个资源的“版本”，客户端来请求的时候，可以比较，如果ETag一致，那么标识该资源没有修改过，客户端可以使用自己缓存的版本。在上面两张图中，第一次请求静态资源的是，返回的ETag为W/”5296-1630c8ae8bd”，并且状态码为200，接下来，第二次请求这个资源的话，浏览器会尝试在请求头部中包含ETag这个信息，以便服务器可以比较，但是此时请求头中存放的是If-None-Match，同样的我们发现它的值也是W/”5296-1630c8ae8bd”，也就是说浏览器本地缓存的静态资源跟服务器的内容是一样的，当然响应之后返回的状态码就是304[13]。

**6.2 系统测试**

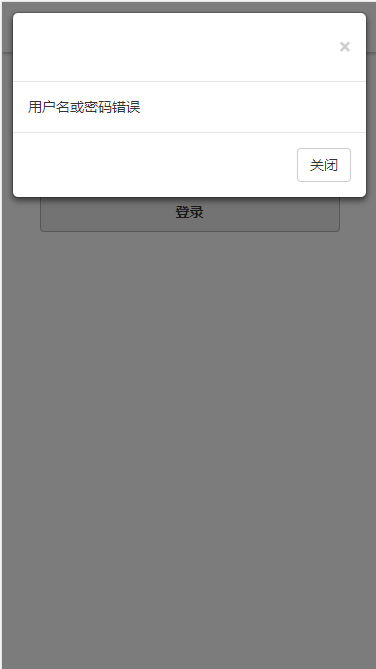


图 6.4 登录测试图

之前已经注册过一个账户名为Aimer密码为123的账号，当输入错误密码时，后台接口会因为查询不到这条用户数据而返回用户名或密码错误。

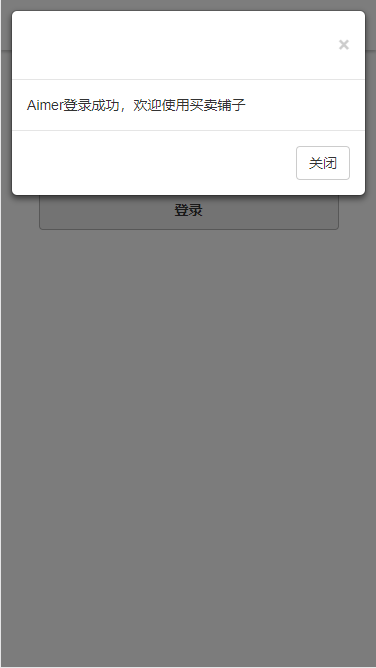


图 6.5 登录成功测试图

当登录成功时，后台返回登录的用户名。



图 6.6 注册测试图

当我们对已经注册过的用户名再次进行注册时，接口会返回该用户名已被注册的信息。



图 6.7 新增商品测试图

在管理后台进行新增商品操作的时候，必须将所有信息都填写完毕才不会提示错误，如果有一项没有填写完全，比如图片没有上传，那么接口会返回“图片还未上传，请上传后重试”的字样。



图 6.8 查询结果测试

查询功能的测试大同小异，这里我用商品查询来举例，当我们输入的商品信息在数据库中查询不到的时候，接口会返回“查询不到数据”的字样。

**6.3 安全测试**

管理员的身份是可以进行普通用户的所有操作的，普通用户的身份是无法进行管理员的操作的，但是如果有些普通用户，在登陆了自己的账户之后，通过手动的方法更改地址栏的地址为后台管理的地址，那么身份的操作权限就乱套了，这是很危险的。我们需要做一下身份跳转限定。



图 6.9 安全测试

这里我用限定普通用户跳转至后台查询视图来做例子  
router.get('/search',**function**(req,res,next){  
 **if**(req.userInfo.isAdmin){  
 res.render('manager-search',{  
 userInfo:req.userInfo  
 });  
 }**else**{  
 res.send("对不起，您不是管理员，无法进入后台界面！");  
 }  
});

如果发现身份是普通用户身份，那么不能跳转至后台管理的视图，同时返回“对不起，您不是管理员，无法进入后台界面！”字样。

重复流程测试。至此，测试完毕。

**结论**

本系统从架构到编码实现，都符合企业开发规范。在实习时候后台开发使用的是C#，但是这种技术已经慢慢被程序界所淘汰，于是趁着毕业论文的机会学习更多知识。所使用的Node.js、MongoDB、Express以及web本地存储功能都是现学现用。靠着自己开发经验和已形成的编程思想，虽然期间磕磕碰碰遇到不少麻烦，如Node.js对所有接口的实现，web本地存储使用，但通过在网上查找资料总能很快解决问题。在编码过程中，受益良多，明白了怎样的程序才是易懂的、高效的、严谨的。良好的封装思想能让程序更大程度地减少耦合度，良好的代码结构能更好地进行后续开发维护。

在思考完成业务流程的同时，我在设计该系统的时候也是严格参考《雅虎军规》的34条规定进行开发，这是一个好的产品值得思考的问题。本系统许多地方都使用页面无刷新技术Ajax实现页面交互和Bootstrap美化页面。在其他优秀的网站上，如processon、慕课网，我总结了一下它们的特点：响应快，操作设计人性化，界面流畅好看，导航分类清晰，它们在细节上做的更是近乎完美。它们围绕者业务核心在完善，而不是添加一堆花哨而无用的功能，商城系统也应如此。我们要明白我们主要是做什么的，而不应该花太多时间在产品功能扩展上。

**参考文献**

[1]朱育发．jQuery与jQuery Mobile开发完全技术宝典：中国铁道出版社，2014.10：1

[2] Procedia Computer Science[J] Volume 96, 2016, Pages 801-810

[3]MongoDB中文文档的[EB/OL] http://www.mongodb.org.cn/tutorial/, 2014 - 2018

[4]Node.js v8.9.3 中文文档[EB/OL] http://nodejs.cn/api/, 2017.10.03

[5] Ioannis K. Chaniotis,Kyriakos-Ioannis D. Kyriakou,Nikolaos D. Tselikas. Is Node.js a viable option for building modern web applications? A performance evaluation study[J]. Computing,2015,97(10).

[6]Express中文文档的[EB/OL] http://www.expressjs.com.cn/ 2015

[7]苗雪兰，刘瑞新，宋歌等. 数据库系统原理及应用教程(第3版)[M].北京：机械工业出版社，2007：9-16，44-53

[8]Bootstrap中文文档的[EB/OL] http://www.bootcss.com/, 2012,5

[9]单东林、张晓菲、魏然.《锋利的jQuery（第2版）》[M].北京：人民邮电出版社，2012.7 ,32-50

[10] (美)泽卡斯（Zakas，N.C.）著；李松峰，曹力译.JavaScript高级程序设计（第3版）[M]北京：人民邮电出版社，2012.3,629-636

[11](美)弗兰纳根(Flanagan,D)著；淘宝前端团队译 JavaScript权威指南(第6版) [M].北京：机械工业出版社，2013.3，581-585

[12] Sun Yu, Wei Zhang. Adaptive Drowsy Cache Control for Java Applications[P]. Embedded and Ubiquitous Computing, 2008. EUC '08. IEEE/IFIP International Conference on,2008.

[13] camel.blog. Yahoo网站性能优化的34条军规的[EB/OL]

https://blog.csdn.net/camel20/article/details/7283893, 2012.02.22

**致谢**

首先要感谢指导老师蔡前凤副教授愿意给我提供这次宝贵的机会，一开始选题的时候，很多老师都不愿意带更多的学生了，是您给我提供了这个让我成长的机会，特别是在我撰写论文的时候给予了很多论文格式上的指导和帮助，真的特别用心！谢谢您！

大学四年眨眼即逝，很多人都还迷迷糊糊不知道自己大学四年做了些什么，学到了些什么。我大一的时候确实很迷茫，跟着素未谋面的师兄师姐开始了长达一个月的数学建模的探索，但是我发现这并不是适合我的一个研究课题，在偶然的一次机会，我接触到了前端，炫酷的界面效果，有趣的逻辑问题一步步将我吸引，从此无法自拔。很庆幸我春招找到的工作是我的兴趣爱好，也很感激读书郎教育科技有限公司给我提供了机会和平台，让我在剩余的大学生活能够静下心来好好撰写这份毕业论文。