**00000000**

**10701**

**学号**

**代号**

**2015年5月**

**提交论文日期**

**工程领域**

**作者姓名**

**base on Node**

**The design and implementation of POS trading system**

**ine**

**line**

**基于Node的Pos机交易系统的设计与实现**

**题（中、英文）目**

`

**软件工程**

**杨涛**

**曾浩 高工**

**李青山 教授**

**学校指导教师姓名职称**

**企业指导教师姓名职称**

**编号**

**UDC**

**公开**

**TP000**



**密级**

**分类号**

摘 要

POS机是电子商务交易平台集成框架的关键组成部分，为电子商务工作提供电子交易功能。随着移动互联网和云计算等相关技术的发展，未来的POS机将不再局限于POS收单终端，而将成为云商店，移动支付，网络支付，线下收单等结合的综合型POS机交易系统。目前已有的POS机在功能性和扩展性以及通用性等方面还不能满足需要。

本文通过分析现有POS机现状以及未来云POS交易系统的发展方向，设计并实现了一个高性能的基于Node的POS机交易系统。在分析系统功能需求的基础上，通过用例和逻辑等视图分析系统的设计架构，采用Express为系统框架，Ejs为视图模板引擎，MongoDB为后台数据库。然后进行用户，商品，购物车，收货信息，订单等功能模块的设计与实现。最后，部署了一个POS机交易系统应用环境对系统进行测试。测试结果表明，该POS机交易系统的可行性和有效性达到了设计要求。

关键字：交易系统 **Node Express Embedded JavaScript MongoDB**

**Abstract**

Establish a common e-commerce trading platform is an important way to accelerate the development and integration of e-commerce. POS trading system is a key part of the integrated frame of the e-commerce trading platform, providing electronic trading function for e-commerce, and improve the operating performance of e-commerce trading system. The existing POS machine trading trading system can not meet the needs of the platform in general and scalability.

This paper is on the basis of analyzing the functional requirement of the system, design and implement a high performance POS machine trading system based on Node. By analyzing the architecture of the system with use case and logic, the system use the Express as the system frame, which make a better scalability for the system. And the system take the Ejs as the view template engine, which make the system clear, maintain good HTML code structure. And the system uses MongoDB as the background database, which make the system has good data access performance. Finally, the application environment of a POS transaction system is deployed. The experimental results show that the feasibility and effectiveness of the POS system based on Node can meet the design requirements.

**Keyword: Trading System Node Express Embedded JavaScript MongoDB**

目 录

[第一章 绪论 1](#_Toc420613770)

[1.1 选题背景及意义 1](#_Toc420613771)

[1.2 国内外现状分析 2](#_Toc420613772)

[1.2.1 国外发展现状 2](#_Toc420613773)

[1.2.2 国内发展现状 2](#_Toc420613774)

[1.3 论文工作内容 3](#_Toc420613775)

[1.4 论文组织结构 3](#_Toc420613776)

[1.5 本章小节 4](#_Toc420613777)

[第二章 相关技术概述 5](#_Toc420613778)

[2.1 Node.js技术 5](#_Toc420613779)

[2.2 Express技术 6](#_Toc420613780)

[2.3 Ejs技术 6](#_Toc420613781)

[2.4 MongoDB技术 7](#_Toc420613782)

[2.5 本章小结 8](#_Toc420613783)

[第三章 基于Node的POS机交易系统需求分析 9](#_Toc420613784)

[3.1 基于Node的POS机交易系统总体分析 9](#_Toc420613785)

[3.2 基于Node的POS机交易系统业务陈述 9](#_Toc420613786)

[3.3 基于Node的POS机交易系统需求建模 11](#_Toc420613787)

[3.4 基于Node的POS机交易系统数据建模 14](#_Toc420613788)

[3.5 本章小结 15](#_Toc420613789)

[第四章 基于Node的POS机交易系统设计与实现 17](#_Toc420613790)

[4.1 基于Node的POS机交易系统应用架构 17](#_Toc420613791)

[4.2 基于Node的POS机交易系统功能设计 17](#_Toc420613792)

[4.3 基于Node的POS机交易系统数据库设计 18](#_Toc420613793)

[4.4 基于Node的POS机交易系统功能实现 20](#_Toc420613794)

[4.4.1资源层 20](#_Toc420613795)

[4.4.2表示层 23](#_Toc420613796)

[4.4.3序列图分析 32](#_Toc420613797)

[4.5 本章小结 34](#_Toc420613798)

[第五章 基于Node的POS机交易系统测试及分析 35](#_Toc420613799)

[5.1 系统运行环境 35](#_Toc420613800)

[5.2 测试用例及过程 35](#_Toc420613801)

[5.3 测试结果分析 37](#_Toc420613802)

[5.4 本章小结 41](#_Toc420613803)

[第六章 结束语 43](#_Toc420613804)

[6.1 论文工作总结 43](#_Toc420613805)

[6.2 后续工作展望 43](#_Toc420613806)

[致 谢 44](#_Toc420613807)

[参考文献 45](#_Toc420613808)

第一章 绪论

1.1 选题背景及意义

POS(point of Sales)交易系统最初定义为即时点销售管理系统或前台系统。该系统有收款机，条形码扫描仪，数据显示器，票据打印机，电子秤，磁场阅读器，封闭式钱箱，信息记录与传送设备等硬件设备和相应的软件构成，是通过收银员在营业场所内的收银机上的实际操作，将商品卖给顾客，实现即时销售管理的过程。

POS系统通常有前台和后台，二者相关联，网络将企业前台电子收银机与后台计算机连接起来，收银员将销售终端采集到的交易时刻，商品编号和数量等信息数据通过特定的方式传送给后台，并用后台的计算机处理完整的商品销售数据，将结果存储到服务器数据库中，企业管理者依此掌握当时、当日、当月等各种商品销售信息，为决策提供依据。POS系统还可以处理如前台商品退货，优惠销售以及盘点等非直接销售业务内容，，从而构成一整套以销售业务为主，其他业务处理为辅的前台系统。

然而，随着互联网IT技术的发展以及电子商务的兴起与壮大，POS机交易系统已经从传统的实体商场、商店的简单收款机，逐渐发展成为电子商务系统交易平台的重要组成部分。

随着移动互联网技术的发展以及POS硬件的提升，出现了基于智能手机和平板电脑的移动POS。以打补丁方式逐步升级的老一代POS正在被新一代开放，全功能的POS平台所替代。而随着零售商与消费者各自需求的提升，现在的POS机已经不仅仅是商家为消费者提供服务的销售点或者服务点，它已经拓展延伸到消费者手中的搜索点，社交点以及购物点。基于互联网技术的电子商务为商家与消费者开辟了一个现实世界意外的虚拟零售世界，移动技术则讲着两个世界整合在一起，为商家与消费者建立了一个1+1>2的多维零售世界。

进来随着云计算等科技的发展，POS机也往云POS方向发展。移动互联网技术与云计算技术的结合使得未来的POS机将不再局限于只是POS收单终端，而将是云商店、移动支付、网络支付、线下收单等功能结合的综合性POS机交易系统。

而Node.js与MongoDB对于构建云应用有着语言与性能的优势。

实现基于Node的以MongoDB为后台数据库的POS机交易系统，将会为电子商务，移动支付以及云POS交易系统带来良好的稳定性，更加便捷良好的可拓展性，以及更加快速处理庞大复杂的交易数据的性能，为社会带来巨大收益。

本文在分析现有POS机以及云POS机的发展方向的基础上，设计并实现了以Node.js为平台，Express为系统框架，EJS为系统框架视图模板引擎，MongoDB为后台数据库的基于Node的POS机交易系统。

1.2 国内外现状分析

### 1.2.1 国外发展现状

POS机最先发展于国外，第一代机采用总线方式联网，具有简单收款机功能，客观上不需要强大的计算机信息管理。随着功能需求的增加，出现了第二，第三代POS。前面的一二三代POS机，由大厂商把控，售价高昂，一般比较大型的商场才负担得起。随着互联网技术的发展以及硬件技术的提升，出现了第四代POS机。

而随着移动技术的发展，出现了移动POS机。欧美许多大型零售企业2013年已经开始大量部署移动POS，这些移动POS包括基于智能手机，平板电脑以及各种移动装置的具有传统POS机功能的POS、店员助理、导购机以及各种门店与消费者互动的接触点。大型零售商虽然大量部署移动POS，但目前还是采用固定POS与移动POS混合使用的系统。而小型零售点与餐饮店等则趋向于逐步全部采用移动POS。

而在移动POS软件市场，零售企业成了市场的主宰者，自行开发的，包括委托第三家开发采用自家品牌的专用POS移动程序成了这个市场的主流。

与此同时，BYOD式CPOS受到商家与消费者青睐开始进入实用阶段。  
　　在零售业IT应用中，最近出现了一种称为自带设备（BYOD）的使用移动设备的工作模式。简单得说，自带设备方式就是将用户自己的手机或平板电脑装上企业免费提供的移动软件，变成一台具有某些特殊功能的移动终端。前面提到过的个人购物助手、Walmart的Scan&Go，都是属于这一类。如果用户自己的移动设备装上某家公司提供的POS软件，那么这个设备就可以在该公司作为POS机使用。这类按BYOD方式工作的POS称为CPOS（Customer POS），即客户POS。  
　　这种新生的CPOS对零售商与顾客都有一定的吸引力。零售商可以省一笔购买移动POS的费用，顾客可用自己手上的设备自助完成某些原来需要店员协助的POS操作，获得更惬意的购物体验。然而，在具体实现中也会遇到种种困难，虽然业内人士对这类CPOS普遍看好，同时也可以听到异议的声音。对于任何一种新兴技术，这种现象是十分正常的。

### 1.2.2 国内发展现状

与国外相比，国内POS机起步较晚，早期的高端POS机几乎都是进口于外国厂商， 价格昂贵，因此只有大型商场才负担得起，而小型卖场都使用国产的具有简单记账功能的收款机。上世纪九十年代末，随着PC硬件的迅速发展，出现了组装POS机。而随着国产POS品牌机的出现，国产POS机在价格方面的优势以及在性能于技术上与进口POS机相差不大，因此国产POS机已经逐渐占据国内中低端的市场，并向高端发起了攻势。

近年来随着阿里巴巴，京东等电子商务网站的发展壮大，移动互联网技术的蓬勃发展，以及POS硬件的提升，出现了基于智能手机与平板电脑的移动POS。移动POS的发展动向正在影响电子商务的交易方式并正在改变零售业和餐饮业的面貌，现在的POS机已经不仅仅是商家为消费者提供服务的销售点或者服务点，它已经拓展延伸到消费者手中的搜索点，社交点，已经购物点。移动POS的发展使得商品交易的主控从商家转移到消费者手中，让消费者更加主动，更有发言权。基于互联网技术的电子商务为商家与消费者开辟了一个现实世界以外的虚拟零售世界。移动技术则将这两个世界整合在一起,为商家与消费者建立了一个1+1﹥2的多维零售世界。

进来随着云计算等科技的发展，POS机交易系统也往云POS方向发展。移动互联网技术与云计算技术的结合，将使得POS机不再局限于只是一个POS收单终端，而是网络商店、移动支付、网络支付、线下收单等功能结合的综合性POS机交易系统。未来的云POS机将可供客户根据不同行业需求，整合自身进销存管理、客户ERP管理等应用程序和资源，为B端客户提供集进销存管理、资金归集为一体的差异化解决方案。

2011~2012年是云POS从概念到使用的发展时期。这段时期内不少公司都在市场推出云POS产品，一些小型零售商与餐饮店也开始采用云POS，来扩充或者接替传统POS。虽然云POS还处于刚刚发展的初期，云POS产品水平与应用水平都还不高，但随着技术条件的实现，以及日益完善的发展生态环境的建立，未来云POS将会展现出良好的发展前景！

1.3 论文工作内容

本文主要研究基于Node的POS机交易系统的设计与实现，在整个过程中主要完成以下工作：

1. 基于Node的POS机交易系统的整体设计。本系统采用Nodejs为开发平台，express为系统框架，Ejs为视图模版引擎，MongoDB为后台数据库。Node.js与express结合使得本系统可通过加载模块的方式增加功能，因此本系统具有良好的可拓展性；而ejs使系统具备明确、维护性良好的HTML代码结构；MongoDB则使得系统具有良好的数据存储能力。
2. 设计并实现系统的各个功能模块，包括用户、商品、购物车、收货信息、订单等模块，这些功能模块使得POS机交易系统不再局限于只是一个POS收单终端。
3. 部署一个POS机交易系统应用环境测试该系统，验证基于Node的POS机交易系统的有效性。

1.4 论文组织结构

论文共分为六章，各章主要内容如下：

第一章：绪论。提出论文选题背景及意义、国内外研究现状分析、论文主要的工作内容和组织结构。

第二章：相关技术概述。列举并简要描述了项目开发过程中涉及到的相关技术，其中包括Node.js技术、Express框架技术、EJS视图模板引擎技术、MongoDB数据库技术等。

第三章：基于Node的POS机交易系统的业务需求分析。在介绍系统业务数据和过程的基础上，完成系统应用需求建模、数据建模和过程建模。

第四章：基于Node的POS机交易系统的设计与实现。设计系统的应用架构，按照功能分解结构细化系统功能，设计软件组件，设计系统数据库。按照组件设计结构进行开发，并完成系统集成。

第五章：基于Node的POS机交易系统测试及分析。阐述系统运行环境，采用测试用例介绍系统测试过程，针对测试结果进行分析给出结论。

第六章：结束语。总结了本文的主要工作，指出工作的不足及进一步的改进方向。

1.5 本章小节

本章提出了论文的选题背景以及意义、国内外研究现状分析以及论文主要的工作内容和论文的组织结构。

第二章 相关技术概述

2.1 Node.js技术

Node.js于2009年由开发者Ryan Danl在github上发布初始版本的node.js包，后来获得云计算服务商Joyent资助，2011年在微软支持下发布了windows版本，目前已经支持Linux，Mac，windows三大平台。

Node.js是一个基于Chrome JavaScript运行时建立的平台，用于方便地搭建响应速度快，易于拓展的网络应用。Node.js使用事件驱动，非阻塞I/O模型而得以轻量和高效，非常适合在分布式设备上运行的数据密集型的实时应用。

Node.js有以下优点：（1）RESTful API；（2）单线程：Node.js可以在不新增额外线程的情况下，依然可以对任务进行并行处理 —— Node.js是单线程的。它通过事件轮询（event loop）来实现并行操作，对此，我们应该要充分利用这一点 —— 尽可能的避免阻塞操作，取而代之，多使用非阻塞操作；（3）非阻塞I/O；（4）V8虚拟机；（5）事件驱动。

Node是一个JavaScript运行环境（runtime）。实际上它是对Google V8 引擎进行了封装。Node对一些特殊用例进行了优化，提供了替代的API，使得V8在非浏览器环境下运行得更好。

V8引擎本身使用了一些最新的编译技术。这使得JavaScript这类脚本语言编写出来的代码运行速度获得了极大提升却节省了开发成本。对性能的苛求是Node的一个关键因素。JavaScript是一个事件驱动语言，Node使用了这个优点，编写出可拓展性高的服务器。Node采用了一个成为“事件循环（event loop）”的架构，使得编写可拓展性高的服务器变得既容易又安全。提高服务器性能的技巧是多种多样的。Node选择了一种既能提高性能，又能减低开发复杂度的架构。这是一个非常重要的特性。并发编程通常是很复杂而且地雷。Node绕过了这些，但仍然提供很好的性能。

Node采用一系列“非阻塞”库来支持事件循环的方式。本质上就是为文件系统、数据库之类的资源提供接口。向文件系统发送一个请求时，无需等待硬盘（[寻址](http://baike.baidu.com/view/1303626.htm" \t "_blank)并检索文件）的结果，当硬盘准备好的时候非阻塞接口会通知Node。该模型以可扩展的方式简化了对慢资源的访问， 直观，易懂，高效。尤其是对于熟悉[onmouseover](http://baike.baidu.com/view/3587213.htm)、onclick等[DOM](http://baike.baidu.com/subview/14806/8904138.htm" \t "_blank)事件的用户，更有一种似曾相识的感觉。

虽然让JavaScript运行于服务器端不是Node的独特之处，但却是其一强大功能。[浏览器](http://baike.baidu.com/view/7718.htm" \t "_blank)环境限制了我们选择编程语言的自由。任何服务器与日益复杂的[浏览器](http://baike.baidu.com/view/7718.htm)客户端应用程序间共享代码的愿望只能通过JavaScript来实现。虽然还存在其他一些支持JavaScript在服务器端 运行的平台，但因为上述特性，Node发展迅猛，成为事实上的平台。

Node.js使用Module模块去划分不同的功能，以简化应用的开发。Modules模块有点像C语言中的类库。每一个Node.js的类库都包含了十分丰富的各类函数，比如http模块就包含了和http功能相关的很多函数，可以帮助开发者很容易地对比如http，tcp/udp等进行操作，还可以很容易的创建http和tcp/udp的服务器。

要在程序中使用模块是十分方便的，只需要如下：

在这里，引入了http类库，并且对http类库的引用存放在http变量中了。这个时候，node.js会在我们应用中搜索是否存在node\_modules的目录，并且搜索这个目录中是否存在http的模块。如果node.js找不到这个目录，则会到全局模块缓存中去寻找，用户可以通过相对或者绝对路径，指定模块的位置，比如：

var myModule = require('./myModule.js');

模块中包含了很多功能代码片断，在模块中的代码大部分都是私有的，意思是在模块中定义的函数方法和变量，都只能在同一个模块中被调用。当然，可以将某些方法和变量暴露到模块外，这个时候可以使用exports对象去实现。

在Node启动的很短时间内，社区就已经贡献了大量的扩展库（模块）。其中很多是连接数据库或是其他软件的驱动，但还有很多是凭他们的实力制作出来的非常有用的软件。

虽然Node项目从2009年才开始发展，还非常年轻，但很少看到对一个项目如此狂热的社区。不管是新手，还是专家，大家都围绕着项目，使用并贡献自己的能力，致力于打造一个探索、支持、分享、听取建议的乐土。

Node.js发展迅猛，目前在不断完善中，版本已经发布至0.12.3.

Node.js集成开发环境：具备书写JavaScript的IDE均可，普通的记事本也可以进行开发。

2.2 Express技术

Express是一个简洁而灵活的Node.js Web应用框架，提供一系列强大特性帮助开发者创建各种Web和移动设备应用。它是目前最稳定，使用最广泛，而且是Node.js官方推荐的唯一一个Web开发框架。

Express有丰富的HTTP快捷方法和任意排列组合的connect中间件，让开发者创建健壮，友好的API变得既快速又简单。

Express并不对Node.js已有的特性进行二次抽象，而是在其基础上拓展Web应用所需的基本功能。

2.3 Ejs技术

Ejs是Express框架的一种模板引擎，它将页面模板和要显示的数据集合起来从而生成HTML。在MVC架构中，模板引擎包含在服务器端。控制器得到用户的请求后，从模型获取数据，调用模板引擎。模板引擎以数据和页面模板为输入，生成HTML页面，然后返回给控制器，由控制器交回客户端。

Ejs只有三种标签：

（1）<% code %>：其中的code是被执行的JavaScript代码；

（2）<% =code %>：显示替换过的HTML特殊字符的内容；

（3）<% -code %>：显示原始HTML的内容。

<%= code %>和<%- code %>的区别在于，当变量code为普通字符串时，两者没有区别；而当code比如为<h1>hello</h1>这种字符串时，<%= code %>会原样输出<h1>hello</h1>，而<%- code %>则会显示H1大的hello字符串。

下面是ejs的官方示例：

|  |
| --- |
| **The Data:**  Supplies: [‘mop’, ’broom’, ‘duster’ ]  **The Template:**  <ul>  <% for(var i=0; i<supplies.length; i++) {%>  <li><%= supplies[i] %></li>  <% } %>  </ul>  **The Result:**  <ul>  <li>mop</li>  <li>broom</li>  <li>duster</li>  </ul> |

我们可以用上述三种标签实现页面模板系统能实现的任何内容。

2.4 MongoDB技术

MongoDB 是一个基于分布式文件存储的 NoSQL（非关系型数据库）的一种，由 C++ 语言编写，旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。MongoDB 支持的数据结构非常松散，是类似 json 的 bjson 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。

MongoDB 没有关系型数据库中行和表的概念，不过有类似的文档和集合的概念。文档是 MongoDB 最基本的单位，所谓文档就是一个关联数组式的对象，每个文档都会以唯一的 \_id 标识，文档的属性为 key/value 的键值对形式，一个值可以是一个数，字符串，日期，数组，甚至可以嵌套另一个文档，因此可以存储比较复杂的数据类型。集合是许多文档的总和，一个数据库可以有多个集合，一个集合可以有多个文档。

MongoDB轻巧，灵活，最大的优势在于所有的数据持久操作都无需开发人员手动编写SQL语句，直接调用方法就可以轻松的实现CRUD操作。

2.5 本章小结

本章简单概述了项目的相关技术，其中包括 Node.js技术，Express技术，Ejs技术，MongoDB技术。

第三章 基于Node的POS机交易系统需求分析

3.1 基于Node的POS机交易系统总体分析

本部分主要对系统进行总体分析，确定系统的参与者和系统的边界。

1. 系统的参与者：注册用户，为注册用户。

注册用户任务：登录系统，搜索商品，浏览商品，提交订单，查询订单

未注册用户任务：浏览商品，搜索商品，注册请求

2. 系统的边界采用**上**下文数据流图（Context data flow diagram）来表示，本系统的上下文数据流图如图3.1：



图3.1系统上下文数据流图

3.2 基于Node的POS机交易系统业务陈述

本部分主要描述待系统的主要业务过程及分析可能存在的问题。采用**活动图**（Activity Diagram）+文字形式进行描述。图3.2是系统的活动图：



图3.2系统总体活动图

说明：用户进行注册，登录，然后也可以浏览商品，搜索商品，点击商品之后进入商品页面，在商品页面有“加入购物车”按钮，登录的用户点击之后商品加入购物车，在导航条点击购物车链接之后进入用户购物车页面，在购物车页面，用户可以对在购物车的商品进行数量增减或者删除商品，购物车下方的收货人地址，用户点击“添加收货地址”后，填写收货人地址，收货姓名，收货人电话，然后点击“提交”按钮后，生成新的收获人信息，用户选收货人信息后，点击“提交订单”按钮，系统生成订单，用户在导航条点击“我的订单”按钮后，进入用户订单页面，页面按时间由近到远列表显示用户的订单信息，登录的用户随时都可以退出，未注册的用户随时都可以注册，未登录的用户随时都可以登录，未登录的用户可以浏览商品，搜索商品，但不能进入购物车页面和订单页面。

3.3 基于Node的POS机交易系统需求建模

本部分对业务过程的内容进行分析，抽取出业务功能、质量属性和约束。业务功能采用**用例图**（Use case Diagram）+文字形式进行描述，质量属性和约束采用文字进行描述。

系统总体用例图如图3.3：

|  |
| --- |
| 图3.3系统总体用例图 |

以下是部分用例详细描述：

表3.1 用户注册用例详细描述

|  |  |
| --- | --- |
| 用例ID | 101 |
| 用例名称 | 用户注册 |
| 用例简介 | 用户进行注册。 |
| 参与者 | 用户 |
| 前置条件 | 用户未注册 |
| 基本事件流 | 1. 用户点击注册 2. 系统返回注册页面 3. 用户填写注册相关信息 4. 点击提交 5. 用户注册成功 |
| 其他事件流 | 用户注册后生成用户对象实例传到数据库 |
| 异常事件流 | 相同用户名的用户已经存在 |
| 后置条件 | 用户注册成功后系统转到主页 |
| 扩展点 | 暂无。 |

表3.2 用户登录用例详细描述

|  |  |
| --- | --- |
| 用例ID | 102 |
| 用例名称 | 用户登录 |
| 用例简介 | 已经注册但未登录的用户进行登录 |
| 参与者 | 用户 |
| 前置条件 | 用户已经注册但是未登录 |
| 基本事件流 | 1. 用户点击登录链接  2. 系统返回登录页面  3. 用户填写用户名和密码  4. 用户点击登录  5. 系统转到主页 |
| 其他事件流 |  |
| 异常事件流 | 1． 库存量不足，则重新选择其他商品  2． 若没有用户搜索的商品，则显示没有相关商品 |
| 后置条件 | 购物车更新 |
| 扩展点 | 暂无。 |

表3.3 用户添加商品用例详细描述

|  |  |
| --- | --- |
| 用例ID | 103 |
| 用例名称 | 添加商品到购物车 |
| 用例简介 | 用户往购物车中添加库存量大于0的商品，商品将在购物车中呈现。 |
| 参与者 | 用户 |
| 前置条件 | 商品数量不为0 |
| 基本事件流 | 1. 用户登录  2. 搜索商品  3. 浏览商品  4. 选择商品   1. 添加商品到购物车 |
| 其他事件流 | 在购物车中可更改商品数量，移除商品等 |
| 异常事件流 | 1． 库存量不足，则重新选择其他商品  2． 若没有用户搜索的商品，则显示没有相关商品 |
| 后置条件 | 购物车更新 |
| 扩展点 | 暂无。 |

表3.4 用户在购物车中删除商品用例详细描述

|  |  |
| --- | --- |
| 用例ID | 104 |
| 用例名称 | 用户在购物车中删除商品 |
| 用例简介 | 用户在购物车中对已经存在的商品进行删除 |
| 参与者 | 用户 |
| 前置条件 | 购物车中存在商品 |
| 基本事件流 | 1. 用户选择商品  2. 用户点击“删除”按钮  3. 被删除的商品在购物车中消失 |
| 其他事件流 | 被删除的商品的显示状态更新 |
| 异常事件流 | 被删除的商品刷新页面后再次在购物车中出现 |
| 后置条件 | 被删除的商品从购物车中消失 |
| 扩展点 | 暂无。 |

表3.5 用户添加收货信息用例详细描述

|  |  |
| --- | --- |
| 用例ID | 105 |
| 用例名称 | 添加选择收货信息 |
| 用例简介 | 若还没有收货信息，则添加；若已经有，则选择收货信息 |
| 参与者 | 用户 |
| 前置条件 | 用户已经登录 |
| 基本事件流 | 1. 点击添加地址  2. 填写收货人地址，姓名，电话等  3. 提交地址  4. 生成新收货信息 |
| 其他事件流 | 若存在多个收货地址，则只能选择一个 |
| 异常事件流 | 若没有选择地址，则不能提交订单 |
| 后置条件 | 收货人信息栏更新 |
| 扩展点 | 暂无。 |

表3.6 用户提交订单用例详细描述

|  |  |
| --- | --- |
| 用例ID | 106 |
| 用例名称 | 用户提交订单 |
| 用例简介 | 购物车中已确定要购买的商品，并选择了收货信息 |
| 参与者 | 用户 |
| 前置条件 | 购物车商品不为空，已经选择了收货信息 |
| 基本事件流 | 1. 确定商品  2. 选择收货信息  3. 点击“提交订单”按钮  4. 购物车商品清空 |
| 其他事件流 | 数据库生成订单对象实例并显示在订单页面 |
| 异常事件流 | 购物车为空 |
| 后置条件 | 订单页面更新，购物车页面更新 |
| 扩展点 | 暂无。 |

表3.7 用户查询订单用例详细描述

|  |  |
| --- | --- |
| 用例ID | 107 |
| 用例名称 | 用户查询订单 |
| 用例简介 | 用户查询已经生成的订单 |
| 参与者 | 用户 |
| 前置条件 | 用户已经登录，订单已经生成 |
| 基本事件流 | 1. 用户登录  2. 进入订单列表页面  3. 查询订单（按订单生成的时间倒序列出） |
| 其他事件流 | 用户登录，订单生成 |
| 异常事件流 | 1. 新用户，没有历史订单  2. 没有要查询的订单 |
| 后置条件 | 显示订单信息列表 |
| 扩展点 | 暂无。 |

3.4 基于Node的POS机交易系统数据建模

对业务过程进一步细化，按照数据建模过程分析系统数据对象集合。

系统数据数据对象集合如下：

1. 用户（.用户ID，用户名，用户密码，用户邮箱）；
2. 商品（商品ID，商品名称，商品价格，商品图片，商品商家）；
3. 购物车（购物车ID，商品对象，用户对象）；
4. 收货信息（收货信息ID，收货人地址，收货人姓名，收货人电话）；
5. 订单（订单ID，用户对象，购物车对象，收货信息对象）。

由于系统采用MongoDB数据库存储数据，而MongoDB为非关系型数据库，文档是其最基本单位，文档的属性为key/value的键值对形式，一个值可以是一个数，字符串，数组甚至是嵌套的另一个文档，而多个文档总和成一个集合，一个数据库可以有多个集合，一个集合可以有多个文档，因此本部分没有实体联系图，但有类似文档类型的数据对象集合。

3.5 本章小结

本章概述了系统总体分析，给出了上下文数据流图；概述了系统业务分析，给出了系统活动图并进行了描述；概述了系统需求建模，给出了系统总体用例图以及各个用例详细描述；概述了系统数据建模，描述了系统数据对象集合。

第四章 基于Node的POS机交易系统设计与实现

4.1 基于Node的POS机交易系统应用架构

系统应用架构设计用来描述系统的骨架和轮廓。本系统为集中式系统，因此可以采用层次模型图描述。

用户注册、登录，添加收货人信息，删除收货人信息，提交订单，添加商品到购物车，在购物车中增减，删除商品等操作向MongoDB数据库发送数据；

而用户登录后，以及用户搜索商品，查看购物车，选择收货人信息，查看订单等操作需要从MongoDB中请求数据。

系统层次模型图如图4.1：

|  |
| --- |
| 图4.1 基于Node的POS机交易系统的层次模型图 |

* 1. 基于Node的POS机交易系统功能设计

基于Node的POS机交易系统包含：用户模块、商品模块、购物车模块、收货信息模块、订单模块，其中：

（1）用户模块包含：用户注册，用户登录，用户退出；

（2）商品模块包含：浏览商品，搜索商品，商品单页；

（3）购物车模块包含：商品加入购物车，增减商品，删除商品；

（4）收货信息模块包含：添加收货信息，删除收货信息，选择收货信息；

（5）订单模块包含：提交订单，查看订单。

用户模块接口为user，商品模块接口为commodity，购物车模块接口为cart，收货信息模块接口为address，订单模块接口为order。

系统功能模块分解图如下：

|  |
| --- |
| 图4.2 基于Node的POS机交易系统功能模块分解图 |

4.3 基于Node的POS机交易系统数据库设计

数据库设计是在概念数据建模的基础上根据特定数据库系统构建系统逻辑数据模型。本系统采用逻辑数据模型+文字进行描述。

系统包含五个数据对象集合：用户，商品，购物车，收货信息，订单。

每个数据对象集合的数据库设计如下：

表4.1 用户表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Comment** | **Default** | **Data Type** | **P** | **F** | **Null** |
| user.id | 用户ID |  | INTEGER(30) | √ |  |  |
| name | 用户名 |  | VARCHAR(16) |  |  |  |
| password | 用户密码 |  | VARCHAR (6) |  |  |  |
| Address | 地址 |  | VARCHAR (50) |  |  |  |
| Email | 邮箱 |  | VARCHAR (20) |  |  | √ |

表4.2 商品表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Comment** | **Default** | **Data Type** | **P** | **F** | **Null** |
| commodity.id | 商品ID |  | INTEGER(30) | √ |  |  |
| cname | 商品名称 |  | VARCHAR(16) |  |  |  |
| cimage | 商品图片 |  |  |  |  |  |
| cprice | 商品价格 |  | INTEGER |  |  |  |
| name | 商家名称 |  | INTEGER |  | √ |  |

表4.3收货信息表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Comment** | **Default** | **Data Type** | **P** | **F** | **Null** |
| address.id | 信息ID |  | INTEGER(30) | √ |  |  |
| Ip | 收货人地址 |  | VARCHAR(100) |  |  |  |
| who | 收货人姓名 |  | VARCHAR (6) |  |  |  |
| phone | 收货人电话 |  | VARCHAR (20) |  |  |  |

表4.4 购物车信息表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Comment** | **Default** | **Data Type** | **P** | **F** | **Null** |
| cart.id | 购物车ID |  | VARCHAR(30) | √ |  |  |
| cname | 商品名称 |  | VARCHAR(16) |  |  |  |
| cprice | 商品价格 |  | FLOAT (6) |  |  |  |
| camount | 商品数量 |  | INTEGER (20) |  |  |  |
| user | 用户对象 |  | User |  |  | √ |

表4.5 订单表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Comment** | **Default** | **Data Type** | **P** | **F** | **Null** |
| order.id | 订单号 |  | INTEGER(30) | √ |  |  |
| user | 用户对象 |  | User |  | √ |  |
| address | 收货信息对象 |  | Address |  |  | √ |
| cart | 购物车对象 |  | Cart |  |  |  |

根据数据库设计系统逻辑数据模型图如4.3：

|  |
| --- |
| 图4.3 系统逻辑数据模型图 |

4.4 基于Node的POS机交易系统功能实现

### 4.4.1资源层

#### 4..4.1.1资源曾概述

资源层组件是完成数据库处理的组件，例如数据源的管理、执行数据库的查询、CRUD。

#### 4.4.1.2资源层类图

资源层类图如图4.4.

|  |
| --- |
| 图4.4 系统总体类图 |

#### 4.4.1.3类详细设计

由系统包含用户类、商品类、购物车类、收货信息类、订单类共5个类，以下是类详细设计：

表4.6 用户类

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 类名称 | user | | 所属包名称 | |  | | |
| 类变量定义(先Public->protected->private) | | | | | | | |
| 名称 | | 类型 | | | | | 描述 |
| name | | string | | | | | 用户名 |
| user.id | | string | | | | | 用户编号 |
| Password | | String | | | | | 用户密码 |
| Email | | string | | | | | 用户注册邮箱 |
| 方法(先Public->protected->private) | | | | | | | |
| 名称 | | 参数 | | 返回值 | | 简单描述 | |
| User.prototype.save | | user | | user | | 保存用户信息 | |
| User.get | | user | | user | | 返回用户信息 | |

表4.7 商品类

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 类名称 | commodity | | | 所属包名称 |  | | |
| 类变量定义(先Public->protected->private) | | | | | | | |
| 名称 | | 类型 | | | | | 描述 |
| commodity.id | | string | | | | | 商品编号 |
| cname | | string | | | | | 商品名称 |
| cimage | |  | | | | | 商品图片 |
| cprice | | String | | | | | 商品价格 |
| name | | string | | | | | 商品商家信息 |
| 方法(先Public->protected->private) | | | | | | | |
| 名称 | | 参数 | 返回值 | | | 简单描述 | |
| Commodity.prototype.save | | commodity | commodity | | | 保存商品信息 | |
| Commodity.getAll | | name | commodity | | | 返回所有商品信息 | |
| Commodity.getOne | | name | commodity | | | 返回一个商品的信息 | |
| Commodity.search | | commodity | commodity | | | 搜索商品 | |

表4.8 购物车类

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 类名称 | cart | | | 所属包名称 |  | | |
| 类变量定义(先Public->protected->private) | | | | | | | |
| 名称 | | 类型 | | | | | 描述 |
| cart.id | | string | | | | | 购物车编号 |
| commodity | | commodity | | | | | 商品对象 |
| camount | | integer | | | | | 商品数量 |
| state | | String | | | | | 商品购物车状态 |
| user | | user | | | | | 用户 |
| name | | String | | | | | 商品商家信息 |
| 方法(先Public->protected->private) | | | | | | | |
| 名称 | | 参数 | 返回值 | | | 简单描述 | |
| Cart.prototype.save | | cart | cart | | | 添加商品到购物车后，生成一个购物车对象 | |
| Cart.get | | user，cart | cart，commodity | | | 返回用户对应的购物车的所有商品 | |
| Cart.del | | user，cart | cart | | | 删除购物车中的商品 | |
| Cart.getnum | | user，cart | cart,commodity.camount | | | 返回购物车中商品数量 | |

表4.9 收货信息类

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 类名称 | address | | 所属包名称 | |  | | |
| 类变量定义(先Public->protected->private) | | | | | | | |
| 名称 | | 类型 | | | | | 描述 |
| add.id | | string | | | | | 收货信息编号 |
| ip | | string | | | | | 收货地址 |
| who | | String | | | | | 收货人姓名 |
| phone | | string | | | | | 收货人电话 |
| 方法(先Public->protected->private) | | | | | | | |
| 名称 | | 参数 | | 返回值 | | 简单描述 | |
| Address.prototype.save | | address | | address | | 保存收货人信息 | |
| Address.get | | user | | address | | 获取收货人信息 | |
| Address.del | | address | | null | | 删除收货人信息 | |

表4.10 订单类

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 类名称 | order | | 所属包名称 | |  | | |
| 类变量定义(先Public->protected->private) | | | | | | | |
| 名称 | | 类型 | | | | | 描述 |
| order.id | | string | | | | | 订单标号 |
| user | | user | | | | | 用户对象 |
| cart | | carts | | | | | 购物车对象 |
| address | | address | | | | | 收货信息对象 |
| 方法(先Public->protected->private) | | | | | | | |
| 名称 | | 参数 | | 返回值 | | 简单描述 | |
| Order.save | | order | | order | | 保存订单信息 | |
| Order.get | | order | | order | | 返回订单信息 | |

### 4.4.2表示层

#### 4.4.2.1表示层概述

表示层负责：管理与用户的请求，做出相应的响应；提供一个controller，调用业务逻辑和其他上层接口；处理异常；简单验证。

#### 4.4.2.2表示层类图

表示层类图如图4.4。

#### 4.4.2.3表示层用例实现

表示层包括用户注册、用户登录、添加商品到购物车、在购物车删除商品、添加收货信息、删除收货信息、订单共7个用例的实现，每个用例包括用例名称，用例概述，用例视图设计以及控制器。

用例一

1. 用例名称：用户注册
2. 用例概述：未注册用户进行注册
3. 视图设计：

表4.11 用户注册页面

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ejs页面说明** | | | | | |
| **页面名称** | | reg.ejs | | | |
| **功能简述** | | 用户点击注册链接，填写注册信息后提交注册 | | | |
| **进入的URL** | | /shop/views/reg.ejs | | **进入方式** | request |
| **数据来源定义** | | | | | |
| **编号** | **参数名称** | | **来源** | **说明** | |
| 1 | name | | 用户输入 | 注册名 | |
| 2 | password | | 用户输入 | 注册密码 | |
| 3 | email | | 用户输入 | 注册邮箱 | |
| **页面外观** | | | | | |
|  | | | | | |

1. 控制器（express）：

表4.12 用户注册路由

|  |  |  |  |
| --- | --- | --- | --- |
| **express路由说明** | | | |
| **路由名称** | shop/routes/index | | |
| **功能简述** | 用户添加商品到购物车  执行的操作：   1. 新用户点击注册链接； 2. 系统返回注册ejs页面; 3. 用户填写注册信息点击注册； 4. 数据库从session中得到数据并生成user对象，然后通过connect-mongo保持到数据库，将回话保持在session中，然后系统返回首页ejs。 | | |
| **输入定义** | | | |
| **#** | **名称** | **类型** | **说明** |
| 1 | name | string | 新用户的用户名 |
| 2 | password | string | 新用户的密码 |
| 3 | email | string | 新用户的邮箱 |
| **处理流程** | | | |
| 获得新用户的信息，通过connect-mongo传给mongodb,数据库，数据库生成新用户对象，保存用户会话在session中，然后返回主页面ejs，数据库信息更新 | | | |
| **输出定义** | | | |
| **#** | **名称** | **类型** | **说明** |
| 1 | user | user | 将新生成的用户对象保存在users集合数据库中 |
| 2 | Commodity | commodity | 将商品对象信息从commoditys集合数据库中获取，放到request中返回给ejs页面显示 |

用例二

1. 用例名称：用户登录
2. 用例概述：已经注册并退出的用户进行登录
3. 视图设计

表4.13 用户登录页面

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ejs页面说明** | | | | | |
| **页面名称** | | login.ejs | | | |
| **功能简述** | | 用户点击登录链接，系统返回登录页面 | | | |
| **进入的URL** | | /shop/views/login.ejs | | **进入方式** | request |
| **数据来源定义** | | | | | |
| **编号** | **参数名称** | | **来源** | **说明** | |
| 1 | name | | 用户输入 | 输入用户名 | |
| 2 | password | | 用户输入 | 输入用户密码 | |
| **页面外观** | | | | | |
|  | | | | | |

1. 控制器

表4.14 用户登录路由

|  |  |  |  |
| --- | --- | --- | --- |
| **Express路由说明** | | | |
| **路由名称** | shop/routes/index | | |
| **功能简述** | 用户注册  执行的操作：   1. 用户点击登录链接； 2. 系统返回登录页面； 3. 用户填写登录信息并点击登录； 4. 系统返回主页面ejs。 | | |
| **输入定义** | | | |
| **#** | **名称** | **类型** | **说明** |
| 1 | name | string | 用户名 |
| 2 | password | string | 密码 |
| **处理流程** | | | |
| 获取用户输入的用户信息，将信息通过connect-mongo传给mongodb,数据库获取用户session，返回主页面ejs | | | |
| **输出定义** | | | |
| **#** | **名称** | **类型** | **说明** |
| 1 | Commodity | Commodity | 将商品信息从commoditys集合数据库中获取，通过request中，传递给首页ejs显示 |

用例三

1. 用例名称：用户添加商品到购物车
2. 用例概述：登录用户进入商品单页后添加商品到购物车
3. 视图设计

表4.15 添加商品页面

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ejs页面说明** | | | | | |
| **页面名称** | | goods.ejs | | | |
| **功能简述** | | 用户进入商品单页，显示上面mongodb传递过来的商品信息 | | | |
| **进入的URL** | | /shop/views/goods.ejs | | **进入方式** | request |
| **数据来源定义** | | | | | |
| **编号** | **参数名称** | | **来源** | **说明** | |
| 1 | Commodity | | commoditys集合 | 从商品对象集合中获取商品信息 | |
| **页面外观** | | | | | |
|  | | | | | |

1. 控制器

表4.16 添加商品路由

|  |  |  |  |
| --- | --- | --- | --- |
| **Express路由说明** | | | |
| **路由名称** | shop/routes/index | | |
| **功能简述** | 用户添加商品到购物车  执行的操作：   1. 从session中获得商品id； 2. 调用commodity.getOne()方法返回商品对象信息，传给ejs页面显示； 3. 用户点击“加入购物车”按钮后，通过connect-mongo对数据库进行操作，数据库生成cart对象，将cart对象传递给cart.ejs页面显示。 | | |
| **输入定义** | | | |
| **#** | **名称** | **类型** | **说明** |
| 1 | commodity.id | string | 商品的ID号，从用户点击“加入购物车”得到 |
| **处理流程** | | | |
| 取出该商品的信息，传递给后面views中的ejs，商品加入购物车，数据通过connect-mongo传给mongodb,数据库生成新的cart对象，数据库信息更新，购物车信息更新 | | | |
| **输出定义** | | | |
| **#** | **名称** | **类型** | **说明** |
| 1 | commodity | commodity | 将商品信息从commoditys集合数据库中获取，通过Request传递给下面的ejs页面显示 |
| 2 | cart | cart | 将商品加入购物车，数据通过connect-mongo传给mongodb,数据库生成新的cart对象保存到carts集合，数据库信息更新，购物车信息更新 |

用例四

1. 用例名称：购物车中删除商品
2. 用例概述：用户在购物车删除已经存在的商品
3. 视图设计

表4.17 购物车页面

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ejs页面说明** | | | | | |
| **页面名称** | | cart.ejs | | | |
| **功能简述** | | 用户删除商品，  显示剩余商品信息 | | | |
| **进入的URL** | | shop/views/cart.ejs | | **进入方式** | request |
| **数据来源定义** | | | | | |
| **编号** | **参数名称** | | **来源** | **说明** | |
| 1 | commodity | | carts集合 | 列表显示购物车对象集合中获取对应用户的购物车对象信息 | |
| 2 | user | | users集合 | 从用户对象集合获取用户对象信息 | |
| **页面外观** | | | | | |
|  | | | | | |

1. 控制器

表4.18 删除商品路由

|  |  |  |  |
| --- | --- | --- | --- |
| **Express路由说明** | | | |
| **路由名称** | shop/routes/index | | |
| **功能简述** | 用户在购物车中删除商品  执行的操作：   1. 用户进入购物页面，选择要删除的商品； 2. 点击“删除”按钮； 3. Session将操作传递给mongodb，数据库更新购物车中该商品显示状态； 4. 购物车页面更新 | | |
| **输入定义** | | | |
| **#** | **名称** | **类型** | **说明** |
| 1 | cart | cart | 购物车对象， |
| **处理流程** | | | |
| 用户取得购物车信息，选择要删除的商品并点击删除按钮后，数据传递到mongodb，更新mongodb数据，传递给后面的购物车ejs | | | |
| **输出定义** | | | |
| **#** | **名称** | **类型** | **说明** |
| 1 | cart | cart | 将用户更改后的购物车信息从数据库中取到Cart数据对象中，并将该Cart对象存放到Request中，传递给下面的ejs页面显示 |

用例五

1. 用例名称：用户添加收货信息
2. 用例概述：用户添加新的收货信息
3. 视图设计

表4.19 收货信息页面

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ejs页面说明** | | | | | |
| **页面名称** | | cart.ejs | | | |
| **功能简述** | | 用户添加收货信息，点击提交后将新生成的收货信息显示在收货信息列表栏 | | | |
| **进入的URL** | | shop/views/cart.ejs | | **进入方式** | respond |
| **数据来源定义** | | | | | |
| **编号** | **参数名称** | | **来源** | **说明** | |
| 1 | address | | 用户输入 | 获取用户添加的收货信息。 | |
| **页面外观** | | | | | |
|  | | | | | |

1. 控制器

表4.20 添加收货信息路由

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Express路由说明** | | | | |
| **路由名称** | | shop/routes/index | | |
| **功能简述** | | 用户添加新的收货信息  执行的操作：   1. 用户在收货信息文本框填写收货地址，收货人姓名，收货人电话； 2. 用户点击提交； 3. Session会话将数据通过connect-mongo传递给数据库，数据库生成新的收货信息对象，并将收货信息对象返回到ejs页面显示 | | |
| **输入定义** | | | | |
| **#** | **名称** | | **类型** | **说明** |
| 1 | ip | | string | 收货地址 |
| 2 | who | | string | 收货人姓名 |
| 3 | phone | | string | 收货人电话 |
| **处理流程** | | | | |
| 用户取得购物车信息，更改选中的商品的数量，数据传递到mongodb，更新mongodb数据，传递给后面的购物车ejs | | | | |
| **输出定义** | | | | |
| **#** | **名称** | | **类型** | **说明** |
| 1 | address | | address | 将用户新添加的收货信息从数据库中取到address数据对象中，并将该address对象存放到Request中，传递给ejs页面显示 |

用例六

1. 用例名称：用户删除收货信息
2. 用例概述：用户对已有的收货信息进行删除
3. 视图设计

表4.21 删除收货信息页面

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ejs页面说明** | | | | | |
| **页面名称** | | cart.ejs | | | |
| **功能简述** | | 用户在收货信息列表栏选择要删除的收货信息进行删除，然后显示删除后剩余的收货信息 | | | |
| **进入的URL** | | shop/views/cart.ejs | | **进入方式** | respond |
| **数据来源定义** | | | | | |
| **编号** | **参数名称** | | **来源** | **说明** | |
| 1 | address | | addresss集合 | 从收货信息集合获取对应用户的收货信息。 | |
| **页面外观** | | | | | |
|  | | | | | |

1. 控制器

表4.22 用户删除已有收货信息路由

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Express路由说明** | | | | |
| **路由名称** | | shop/routes/index | | |
| **功能简述** | | 用户删除已有收货信息  执行的操作： | | |
| **输入定义** | | | | |
| **#** | **名称** | | **类型** | **说明** |
| 1 | address | | address | 收货信息对象 |
| **处理流程** | | | | |
| 用户取得购物车信息，更改选中的商品的数量，数据传递到mongodb，更新mongodb数据，传递给后面的购物车ejs | | | | |
| **输出定义** | | | | |
| **#** | **名称** | | **类型** | **说明** |
| 1 | address | | address | 将用户新添加的收货信息从数据库中取到address数据对象中，并将该address对象存放到Request中，传递给ejs页面显示 |

用例七

1. 用例名称：用户提交查看订单
2. 用例概述：用户在确定要购买商品并选择收货信息后提交订单并查看订单
3. 视图设计

表4.23 订单页面

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ejs页面说明** | | | | | |
| **页面名称** | | order.ejs | | | |
| **功能简述** | | 用户在购物车ejs点击“提交订单”按钮后，新生成的订单在订单ejs页面显示 | | | |
| **进入的URL** | | shop/views/order | | **进入方式** | Respond |
| **数据来源定义** | | | | | |
| **编号** | **参数名称** | | **来源** | **说明** | |
| 1 | cart | | carts集合 | 构成订单的购物车信息，从购物车对象集合获取 | |
| 2 | Address | | address集合 | 构成订单的收货信息，从收货信息对象集合获得 | |
| **页面外观** | | | | | |
|  | | | | | |

1. 控制器

表4.24 用户提交查看订单路由

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Express路由说明** | | | | | |
| **路由名称** | shop/routes/index | | | | |
| **功能简述** | 用户提交查看订单  执行的操作如下：   1. 用户在购物车页面点击“生成订单”按钮； 2. 从session中获取购物车对象信息和收货信息对象，通过connect-mongo将数据传到数据库； 3. 数据库生成新的订单对象实例，更新购物车数据库集合信息和订单数据库集合信息之后，将实例传递到订单ejs页面显示； 4. 用户点击“我的订单”链接，系统转到订单页面，页面显示用户 | | | | |
| **输入定义** | | | | | |
| **#** | | | **名称** | **类型** | **说明** |
| 1 | | | cart | cart | 从购物车对象集合中获取对应用户的购物车对象 |
| 2 | | | address | address | 从收货信息对象集合中获取对应用户选择的收货信息对象 |
| **处理流程** | | | | | |
| 从cart得到构成订单的购物车信息以及收货信息，通过connect-mongo传递到数据库之后生成新的订单对象，更新数据库信息之后，新订单对象传递给订单ejs页面，订单页面显示订单信息 | | | | | |
| **输出定义** | | | | | |
| **#** | | **名称** | | **类型** | **说明** |
| 1 | | Order | | Order | 从订单对象集合中获取对应用户的订单对象，通过Request，传递给下面的ejs页面显示 |

### 4.4.3序列图分析

（1）用户模块功能实现

序列图如图4.5，用户点击系统导航条的注册或者登录链接后，系统返回注册或者登录的ejs页面，用户填写注册或者登录用的用户个人信息并点击“提交”按钮，用户提交后，ejs页面将数据发送到MongoDB数据库，如果用户是注册，则数据库查询有没有该用户，如果有，则注册失败，提示用户重新注册；如果没有，则数据库生成一个新的用户对象实例，并保存到数据库，然后返回登录后的ejs主页，此时用户注册成功；如果用户是登录，则数据库查询有没有给用户，如果有，则返回登录后的ejs主页，如果数据库没有该用户，则系统提示没有该用户，并转到注册ejs页面。

|  |
| --- |
| 图4.5 用户注册、登录序列图 |

（2）商品模块功能实现

序列图如图4.6，用户登录后，系统返回主页，主页列表显示商品。用户在搜索框输入要搜索的商品的关键字并点击“搜索”按钮后，搜索框ejs页面将关键字发送到MongoDB数据库进行查询，如果数据库有与关键字相关的商品，则返回商品，系统转到搜索结果页面；如果没有，则返回空白页。用户点击某个商品后，进入商品单页ejs页面，显示单个商品的信息，商品下方有“加入购物车”按钮，用户点击“加入购物车”按钮后，ejs将请求传送到MongoDB数据库，数据库生成并保存新的购物车对象实例，然后购物车对象实例在购物车页面按时间由近到远显示，此时用户成功将商品添加到购物车。

|  |
| --- |
| 图4.6 搜索、浏览商品，添加商品到购物车序列图 |

（3）系统购物车模块功能实现

序列图如图4.6，登录用户点击系统导航条的“购物车”链接后，购物车ejs页面按时间由近到远依次显示用户已经添加到购物车而且未提交到订单的商品的数据，用户可以对商品进行数量的增减或者删除商品，当增减或者删除商品后，购物车ejs页面将数据传送到数据库，数据库更新页面购物车对象信息，并将结果返回到购物车ejs页面。

（4）系统收货信息模块功能实现

序列图如图4.6，当登陆的用户在购物车ejs页面下方点击“新添地址”按钮后，系统弹出三个文本框，提示用户分别填写收货人地址，收货人姓名，收货人电话，当填写完毕后，点击“提交”按钮，ejs页面将数据发送到MongoDB数据库，数据库生成并保存一个新的收货人信息对象实例，然后对象实例显示在收货人信息栏。当用户点击已有收货人信息的“删除”按钮后，ejs页面将操作请求发送到MongoDB数据库，数据库更新收货人信息对象数据并返回数据给ejs页面。

（5）系统订单模块功能实现

序列图如图4.6，当登录用户在购物车ejs页面已经确定购物车商品的信息，并且已经选择收货人信息后，点击页面“提交订单”按钮，ejs页面将数据发送到MongoDB，MongoDB生成并保存一个新的订单对象实例，订单对象实例在订单ejs页面按时间由近到远列表显示，购物车ejs页面将已经提交订单的商品信息清除，MongoDB数据库信息更新。用户点击系统导航条“我的订单”链接后，系统转到订单ejs页面，页面按时间由近到远列表显示已经生成的订单的信息；如果用户还没有提交过订单，则页面提示“订单为空”。

4.5 本章小结

本章第一部分进行系统应用架构，用层次模型图进行描述；第二部分进行系统功能设计，用功能模块分解图进行描述；第三部分进行系统数据库设计，根据系统数据实体设计5个数据库，并用逻辑数据模型图进行描述；第五部分进行系统功能实现，用类图以及序列图进行描述。

第五章 基于Node的POS机交易系统测试及分析

5.1 系统运行环境

系统测试平台需要真实或者模拟的运行环境支持。本系统在Linux操作系统Ubuntu14.04LTS版本中进行测试，Node.js版本为V0.12.2，express版本为4.12.2，MongoDB版本为1.4.15，浏览器为Chrome，启动程序之后，访问浏览器localhost：3000端口。

5.2 测试用例及过程

本部分分步骤采用文字对系统测试用例以及过程进行描述。

根据系统需求分析、软件测试计划以及测试用例对系统进行测试。为了测试用户注册、登录以及退出功能，本文设计了测试用例1、2；为了测试用户浏览商品、搜索商品、添加商品到购物车以及在购物车中删除商品，本文设计了测试用例3、4；为了测试用户添加、删除以及选择收货信息功能，本文设计了测试用例5、6；为了测试系统提交订单以及查看订单功能，本文设计了测试用例7、8。

测试用例以及过程如下：

|  |
| --- |
| 1. 用户注册测试：   用例编号：TC\_101  测试目的：未注册用户是否可以注册，已注册用户是否可以再次注册。  测试场景：在注册页面，填写注册相关信息，进行注册。  所需数据：注册用户名，密码，邮箱。  测试步骤：①进入系统注册页面；  ②填写注册的用户名，密码，邮箱等相关信息；  ③用户点击“注册”按钮。  预期结果：①如果用户名尚未存在，则注册成功；  ②如果用户已经存在，则注册不成功，提示“用户名已经存在”，系统转到注册页面。 |

|  |
| --- |
| 1. 用户登录测试：   用例编号：TC\_102  测试目的：未登录用户是否可以登录，已登录用户是否可以再次登录。  测试场景：在登录页面，填写登录相关信息，进行登录。  所需数据：注册用户名，密码。  测试步骤：①进入系统登录页面；  ②填写登录用户名，密码等相关信息；  ③用户点击“登录”按钮。  预期结果：①如果用户名尚未存在，则登录失败；  ②如果用户已经注册，但还没登录，则登录成功；  ③如果用户已经注册并且已经登录，显示用户已经登录。 |

|  |
| --- |
| 1. 用户添加商品到购物车测试：   用例编号：TC\_201  测试目的：用户是否可以将商品加入购物车。  测试场景：在商品单页面，将商品添加到购物车。  所需数据：数据库中已经存在的商品。  测试步骤：①进入系统商品单页面；  ②点击“加入购物车”按钮；  ③查看购物车。  预期结果：①如果用户尚未登录，则转到登录页面；  ②如果用户已经登录，则购物车中显示添加的商品。 |

|  |
| --- |
| 1. 在购物车删除商品测试：   用例编号：TC\_202  测试目的：登录的用户是否删除购物车中已有商品。  测试场景：在购物车中选择已有商品进行删除。  所需数据：购物车中已有商品。  测试步骤：①进入系统购物车页面；  ②选择购物车中的商品；  ③对商品进行删除。  预期结果：①点击“删除”按钮后，刷新页面，已删除的商品没有出现。 |

|  |
| --- |
| 1. 用户增加收货信息测试：   用例编号：TC\_301  测试目的：用户是否可以增加收货信息。  测试场景：在购物车页面进行增加收货信息。  所需数据：收货人地址，收货人姓名，收货人电话。  测试步骤：①点击“添加收货信息“按钮；  ②系统弹出文本框后名，填写地址，姓名，电话；  ③用户点击“提交信息”按钮；  ④用户查看是否出现新增加的收货信息。  预期结果：①页面出现新增加的收货信息。 |

|  |
| --- |
| 1. 用户删除收货信息测试：   用例编号：TC\_302  测试目的：用户实时可以删除已经有的收货信息。  测试场景：在收货信息栏列表选择要删除的信息进行删除。  所需数据：已经有的收货信息。  测试步骤：①进入系统收货信息页面；  ②选择要删除的收货信息，点击“删除”按钮；  ③用户刷新页面，查看已经删除的信息是否出现；  预期结果：①被删除的信息没有在出现。 |

|  |
| --- |
| 1. 提交查看订单测试：   用例编号：TC\_401  测试目的：用户是否可提交订单。  测试场景：用户已经确定要购买的商品并选择了收货信息。  所需数据：用户要购买的商品，收货信息。  测试步骤：①用户在购物车确定要购买的商品的数量并选择收货信息  ②点击“提交订单”按钮；  ③刷新购物车页面；  ④用户查看订单页面。  预期结果：①如果购物为空，则不能提交订单；  ②提交订单后，购物车信息为空；  ③订单页面出现提交的订单信息。 |

5.3 测试结果分析

按测试用例对测试结果进行分析：

1. 用户注册：

未注册的用户在注册页面填写注册信息然后点击注册，如果系统中还没有相同用户名的用户，则用户注册成功；否则注册失败，用户重新注册；已经注册并登录的用户点击注册链接时，系统没有转到注册页面。注册页面如图5.1：

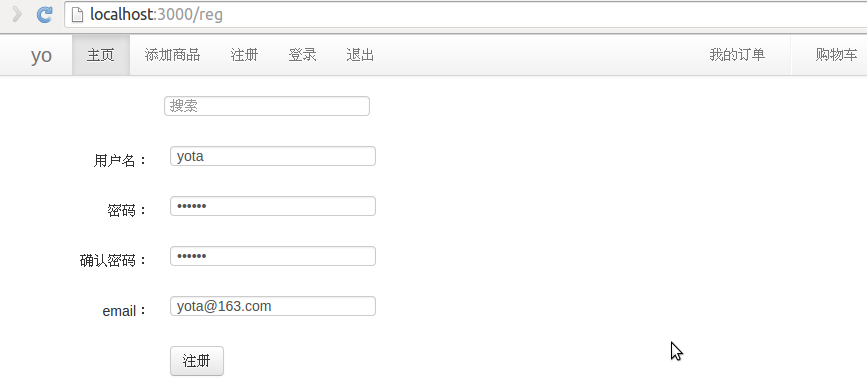


图5.1用户注册页面

结论：用户注册功能成功。

1. 用户登录：

未注册的用户进行登录，登录不成功，页面转到注册页面；已经注册的用户进行登录，填写不匹配的用户名与密码时登录不成功；填写匹配的用户名与密码时，登录成功；登录后用户点击登录链接时系统没有转到登录页面，登录用户点击退出链接时，用户退出。登录页面如图5.2：

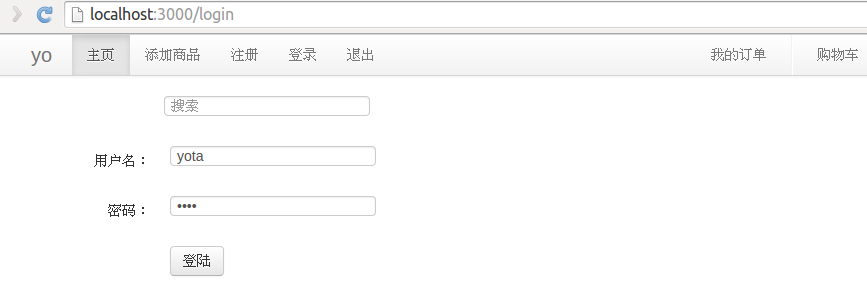


图5.2 用户登录页面

结论：用户登录退出功能成功。

1. 用户添加商品到购物车：

用户登录到系统主页时，可以浏览商品，也可以搜索商品，点击某个商品时，进入商品单页，商品页面如图5.3：

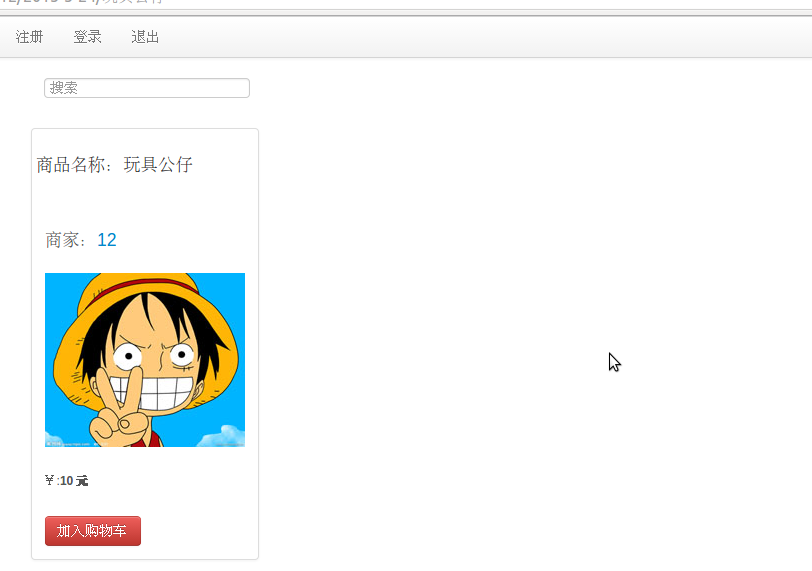


图5.3 用户添加商品

在商品单页点击“加入购物车”按钮，然后到用户的购物车页面查看，商品已经添加到购物车。

结论：用户添加商品到购物车成功。

1. 用户删除商品：

用户在购物车中选择已经添加到购物车的商品，点击“删除”按钮，在购物中消失。购物车商品页面如图5.4：



图5.4 购物车商品页面

结论：用户删除商品成功。

1. 用户添加删除收货信息：

用户在收货信息相关文本框填写收货人地址，收货人姓名，收货人电话，然后点击“添加地址”按钮，页面显示新添加的收货信息，收货信息页面如图5.5：



图5.5 用户添加删除收货信息

选择其中某个收货信息，然后点击“删除”按钮，选择的收货信息从页面消失。

结论：用户添加删除收货信息成功。

1. 用户提交查看订单：

用户确定要购买的商品并选择收货信息后，点击“生成订单”按钮，页面显示“订单已经生成”，到订单页面查看，订单页面显示新生成的订单，订单页面如图5.6。



图5.6 用户查看订单

结论：用户提交、查看订单成功。

本文对系统所有功能模块进行大于50次的测试，其中除了在购物车删商品这一个功能模块中页面刷新会偶尔出现迟缓现状之外，其他功能模块全部流畅稳定运行，部分运行结果截图如上所示。经过分析得知，购物车删除商品功能模块页面刷新会偶尔出现迟缓现象是因为浏览器打开多个页面导致图片加载速度变慢的原因，而这在可控范围内是可接受的。

分析以上测试结果可得结论：系统实现需求分析要求的全部功能性需求与非功能性需求，并且保存良好的稳定和良好可拓展性。

5.4 本章小结

本章部署系统测试环境，根据系统需求分析、软件测试计划以及测试用例对系统进行测试，给出部分测试结果，并对测试结果进行了分析。

第六章 结束语

6.1 论文工作总结

本文至此已经基本完成基于Node的POS机交易系统的设计与实现的项目以及论文撰写。

本文首先概述了绪论，描述了选题背景以及国内外现状等；

然后介绍了相关的技术，其中包括：Node.js技术、Express技术、EJS技术、MongoDB技术；

之后进行系统业务需求分析，描述了系统业务陈述，系统需求分析，数据建模以及过程建模；

然后开始系统设计与实现，包括进行系统应用架构，系统功能设计，系统数据库设计，以及系统功能的实现；

之后部署系统运行环境，根据系统需求分析、软件测试计划和测试用例对软件进行了测试并分析测试结果。

6.2 后续工作展望

项目页面不够美观，虽然采用bootstrap前端框架，但后面还需要进行进一步的页面美化设计以及改进。

针对项目在购物车中删除商品刷新页面后再次出现的问题，是因为在路由中设置了每次刷新购物车页面都对商品状态值置1，因此商品会再次出现的，后面将修改过来。

致 谢

项目实施以及论文工作至此已经基本完成。谢谢在此过程中家人的支持与鼓励，谢谢项目指导老师曾老师的支持以及指导，谢谢论文指导老师李教授的支持与指导，谢谢在实验室中一起完成项目的同学们的相互支持与鼓励。

参考文献

【普通图书】

[1] 郭家宝. Node.js开发指南[M]. 第一版. 北京:人民邮电出版社,2007

[2] 赵坤, 寸志, 雷宗民, 吴中骅. Node.js实战[M]. 第一版. 北京:电子工业出版社2014

【电子文献】

[3] Node.js官方网站. Node.js官方API. 2015. https://nodejs.org/api/

[4] Express官方网站. Express官方API. 2015. <http://expressjs.com/4x/api.html>

[5] EJS官方网站. EJS官方API. 2015. <http://www.embeddedjs.com>

[6] MongoDB官方网站. MongoDB官方API. 2015 http://docs.mongodb.org/manual/