



**本科毕业设计（论文）**

基于web的大型商场导购系统的设计与实现

学 号：15

学生姓名：牛 明 鑫

指导教师：贺 薪 宇

所在学院：计算机科学技术学院

所学专业：软件工程专业

2018 年 6 月

# 摘 要

随着经济的发展，大型商场越来越多，商场中出现了各种各样的导购系统。本文论述了基于web的大型商场导购系统的设计与实现，主要从概述，方案论证，需求分析，设计论述等几个方面进行逐步论述。基于web的大型商场导购系统，主要是用来帮助用户在商场中有更好的体验。本系统的主要设计思路：首先用户到达该商场时，可以直接访问该系统，可以访问该商场中的各类商品信息，可以查看品牌商铺的信息，可以查看最新打折商品信息，可以进行精确的查找某个商品的信息。然后管理员可以进行登录，登录后可以进行管理各类商品信息和已经入驻该商场的商铺也可以进行发布各种打折促销信息和各类商场中的活动。该系统主要用到的技术，前端主要通过Vue搭配element-ui进行单页面开发，后端主要通过node结合express4.x进行开发，然后把数据存在mysql中。该项目遵循了MVC的开发思想，争取让系统成本低、易升级、可扩充，高质量。

**关键词**：商场导购系统；node；vue；mysql；element-ui；express4.x

# Abstract

This paper discusses the design and implementation of shopping guide system in large-scale shopping malls based on web. It mainly discusses in several aspects such as overview, program demonstration, requirement analysis, and design discussion. With the development of economy, more and more large-scale shopping malls, shopping malls have appeared in a variety of shopping guide system. For example: shopping malls in Beijing, Shanghai, Hong Kong and other places have already installed a shopping guide system. The web-based mall shopping guide system is mainly used to help users have a better experience in shopping malls. The main design idea of ​​the system: First, when the user arrives at the mall, he can directly access the system, can access various commodity information in the mall, can view the information of the merchant, can check the latest commodity information, and can accurately search for a certain Product information. The administrator can then log in, log in to manage merchandise information of various types, and merchants who have settled in the mall can also publish various discounted promotions and activities in various shopping malls. The system mainly uses the technology, the front end mainly through Vue and element-ui for single-page development, the back-end mainly through the node with express4.x development, and then the data exists in mysql. The project follows the development concept of MVC and strives to make the system cost-effective, easy to upgrade, scalable, and high-quality.

**Key words:** Shopping guide system; node；vue；mysql；element-ui；express4.x

**目 录**

[摘 要 I](#_Toc516465820)

[Abstract II](#_Toc516465821)

[第1章 概述 1](#_Toc516465822)

[1.1 课题的背景、意义 1](#_Toc516465823)

[1.1.1 课题的背景 1](#_Toc516465824)

[1.1.2 课题的意义 1](#_Toc516465825)

[1.2 国内外发展状况 1](#_Toc516465826)

[第2章 方案论证 3](#_Toc516465827)

[2.1 方案提出 3](#_Toc516465828)

[2.1.1 方案一 3](#_Toc516465829)

[2.1.2 方案二 3](#_Toc516465830)

[2.1.3 方案三 4](#_Toc516465831)

[2.2 可行性分析 4](#_Toc516465832)

[2.2.1 经济可行性 4](#_Toc516465833)

[2.2.2 技术可行性 4](#_Toc516465834)

[2.2.3 操作可行性 5](#_Toc516465835)

[2.2.4 法律可行性 5](#_Toc516465836)

[2.3 方案确定 5](#_Toc516465837)

[2.4 相关技术介绍 6](#_Toc516465838)

[2.4.1 主要语言 6](#_Toc516465839)

[2.4.2 主要数据库 6](#_Toc516465840)

[2.4.3 主要开发工具 6](#_Toc516465841)

[第3章 需求分析 7](#_Toc516465842)

[3.1 功能需求分析 7](#_Toc516465843)

[3.2 性能需求分析 7](#_Toc516465844)

[3.3 系统数据流图和数据库需求分析 8](#_Toc516465845)

[3.3.1 系统数据流图 8](#_Toc516465846)

[3.3.2 数据库需求分析 8](#_Toc516465847)

[3.4 数据字典 9](#_Toc516465848)

[3.5 系统E-R图 9](#_Toc516465849)

[3.5 逻辑模型 13](#_Toc516465850)

[3.6 系统用例图 13](#_Toc516465851)

[第4章 过程论述 14](#_Toc516465852)

[4.1 概要设计 14](#_Toc516465853)

[4.1.1 总体功能设计 14](#_Toc516465854)

[4.2 数据库详细设计与实现 15](#_Toc516465855)

[4.2.1 创建数据库 15](#_Toc516465856)

[4.2.2 创建表和字段 16](#_Toc516465857)

[4.3 数据库连接 18](#_Toc516465858)

[4.4 详细设计 19](#_Toc516465859)

[4.4.1 系统登录 19](#_Toc516465860)

[4.4.2 主界面设计 20](#_Toc516465861)

[4.4.3 人员管理模块 21](#_Toc516465862)

[4.4.4 商品信息管理/查询模块 22](#_Toc516465863)

[4.4.5 商场管理模块 23](#_Toc516465864)

[4.4.6 商场活动推介模块 25](#_Toc516465865)

[第5章 软件测试与结果分析 27](#_Toc516465866)

[5.1 测试的目的和原则 27](#_Toc516465867)

[5.2 测试计划 27](#_Toc516465868)

[5.3 测试过程 28](#_Toc516465869)

[5.3.1 单元测试 28](#_Toc516465870)

[5.2.2 集成测试 29](#_Toc516465871)

[5.2.3 系统测试 30](#_Toc516465872)

[5.4 测试结论 30](#_Toc516465873)

[结 论 31](#_Toc516465874)

[致 谢 32](#_Toc516465875)

[参考文献 33](#_Toc516465876)

[附 录 34](#_Toc516465877)

# 第1章 概述

## 1.1 课题的背景、意义

### 1.1.1 课题的背景

随着互联网产业的不断发展，随着人们生活水平的提高，大型商场面临着复杂切多变的行业形态，大型商场需要做出有效且快速的转变，才能顺应时代发展的潮流。商场既保留了传统销售模式，也需要和现代科技结合起来，为消费者提高更加灵的导购方式，以此来提高消费者的购物体验。当消费者浏览商场时，需要快速的寻找到自己需要的商品和品牌。也需要根据理想的价格，进行快速的定位到商品的位置。 一些较低折扣的商品，能够快速的被消费者所看到，在商品管理方面可以提供较大的便利。因此需要一个商场导购系统。

大型商场中会看到宣传个人品牌和商品占据了比较大的面积和空间，在宣传时，必然会造成资源的浪费、不环保，每次品牌店换位置，就需要更新很多东西，会造成不必要的成本开支等[1]。利用现代科技，开发导购系统可以避免这些浪费。

### 1.1.2 课题的意义

进入了大学，经过了对软件专业知识的学习，以及经过了一些项目的锻炼，希望通过这次毕设来进行检验自己这么久的学习成果。同时，也希望给用户和商场管理人员在大型商场中购物和管理时提供便利。

如今，node可以同时开发前端和后端，并且node社区已经逐步的完善，使用node开发的人员逐步增多，作为一个前端开发者是必须要掌握这一门技术的。通过本次毕设可以对我大学四年所学知识的一个综合检测，通过毕业实习进行实战项目，从不同的方面深入了解。将实习中所学到的产品设计理念，以及在开发中所运用到的编程技术，结合在大学专业课学习中学到的专业知识。进行本次导购系统的开发。

本课题主要的实践意义，主要是针对越来越大的商场，人们在大型中购物时可以更好的增加用户体验。在商场中开发一个导购系统就可以为我们的生活提供很大的便利。作为一个即将进入社会的程序员，可以更好的学习互联网技术。可以更多接触新兴的技术。伴随科技时代的发展，我个人的技术能力也将有更大的提高。

从现在看来，我开发的这个商场导购系统。可以为用户提供一些基本便利，商店管理人员在管理商场以及商品的时候更加方便。

## 1.2 国内外发展状况

商场作为购物的中心，在国内外都有大型企业涉足，国外有商场巨头沃尔玛，国内有万达，银泰等。国外的商场起步比国内的早，而且大多数国外的商场都设有导购平台和自助导购平台，例如国外家具商场益家在自己的商场内设有自助导购查询平台，客户可以在这里查询到想要找的商品或者类似种类商品的位置。

相比于国外的商场导购系统，国内的商场导购系统虽然起步比较晚，但是国内商场的发展速度快，与之相关的商场配套设施的发展也齐头并进。国内先后出现许多优秀的商场导购系统，但是商场导购系统都是针对固定企业的固定领域，缺少通用的大型商场导购系统。

# 第2章 方案论证

## 2.1 方案提出

### 2.1.1 方案一

方案一技术栈：前端vue+后端node+数据库mysql

选出毕设题目之后，我就在准备个人所要用到的技术。并且了解到node.js是最近新兴的一门技术。并且可以通过npm包进行管理，使得node的生态圈越来越大并且很完善。Node通过js进行运行，解决了弱类型语言不能开发后端的弊端。Node主要是在一些实时数据，并且在单线程非阻塞IO有很大的优势。node可以在不新增额外线程的情况下，依然可以对任务进行并发处理[2]。Express是以node为前提开发的基于web的框架。类似于对java进行封装得到的spring boot，通过express 可以创建强健、友好的API变得快速且简单[3]。

Vue是最近兴起的一个比较流行的前端框架，易学习，并且社区完善，并且不同于传统的html，完全可以不操作dom，就可以实现很多效果。

MySql是一个较小的数据库，在使用中可以很好的结合vue并且。很容易操作。

通过完成vue后，然后进行node开发服务器端是非常容易的，并且node和vue可以完全独立开来。然后node也可以很好的连接mysql。

### 2.1.2 方案二

方案二技术栈：前端vue+后端java+数据库MongoDB

相对于方案一，方案二前端依然是用vue。后端是利用了在学校学习的java。Java作为现在使用最多的语言之一，各种各样的信息都比较完整。Java是一门简单的，跨平台，面向对象，并且移植性非常好的语言[5]。让它作为本项目后端的开发也是一个很好的选择。

MongoDB是在实习期间听得最多的一门语言。主要是由于MongoDB是基于分布式存储的数据库，它是介于关系数据库和非关系数据库之间的产品，是非关系数据库中功能最丰富的，最像关系数据库的[6]。

MongoDB具有以下功能特性：

面向集合存储，易存储对象类型的数据；

模式自由；

支持动态查询；

支持复制和故障恢复；

使用高效的二进制数据存储，包括大型对象；

自动处理碎片，以支持云计算层次的扩展性；

文件存储格式为BSON；

可以通过网络访问。

此方案就需要配置好java然后连接完成数据库，在进行vue的安装与开发。

### 2.1.3 方案三

方案三技术栈：前端vue+后端ASP.NET+数据库SQL Server

方案三是主要你使用了微软所开发的语言以及数据库。ASP是作为微软开发的一门后端语言，和java相比它并非是开源的，因此使用的也比较少。ASP可以很好的保证安全性，并且可以防止asp代码被盗，通过asp开发的程序在进行测试和修改的时候会很容易，无需编译就可以以解释执行[7]。

SQL Sever也是微软开发的一门数据库，当时在使用该数据库的时候，发现诸多问题，使得我在开发的时候，体验较差。但是SQL Sever以及被广泛应用在电子商务、银行等相关行业，SQL Server作为数据库可以提供，安全的，强健的，灵活的数据[8]。

而且，因为它方便操作地特性和它的友好的操作页面，非常受广大用户的喜欢。

这俩语言虽然不是开源的但都是微软下的产品，并且可以实现无缝结合起来。来让个人项目达到最优。

## 2.2 可行性分析

可行性分析是通过对项目的主要内容和配套条件，如市场需求，资源供应等，从技术、经济和工程等方面进行调查研究和分析比较，并且对项目建成以后可能取得的财务、经济效益以及社会环境进行预测[9]。在该项目中，我们要先想想自己在开发过程，所会遇到的问题，并且分析那些方法是可行的。虽然会遇到很多问题，但我们不可能全部解决，我们应该尽量的分析清楚，然后尽可能的避免这些问题。主要通过以下几个方面进行分析。

### 2.2.1 经济可行性

就本系统来说，开发所需要的资金少，系统开发完成后，可以让用户，更容易在商场中找到自己需要的商品，也可以减少商家利用场地打广告的费用，商场成为现代科技的载体，让科技成为廉价的劳动力，让管理员解放出来，不仅可以节省大量时间，还可以极大的提高用户的体验。通过我对本次开发的预算，开发该商场导购系统，可以最大化的经济效益。本次开发主要是个人的花费，最主要的开发该系统可以提高工作效率。并且之上的几种方案，都可以免费使用。

因此从经济上来看，实现web导购系统是完全可行的。

### 2.2.2 技术可行性

在开源的新时代，开发一个基于web的商场导购系统。我们可以有许多工具进行选择，例如我们经常使用的webstorm，HBuilder等都是一些比较理想的前端开发工具，在后台开发工具上，有IDEA，VSCODE等。在数据库选择中有mysql，sqlserver等。

如今前端开发技术已经比较成熟，html，css，js已经广为普及，成为程序员一项比不可少的技能。Html是一个比较火的入门语言，它是所有框架的基础，之后学会了这门语言才可以进行后续的学习，在此之后我们可以学习一系列的框架：angular，vue，react等。Html具有有简易性，可扩展性，平台无关性，通用性[10]。

Css主要是对html中内容进行格式化，他就想当于一个美工，让你写出的页面更有样式，更有体验感。还可以配合各种脚本语言动态地对网页各元素进行格式化[11]。

Js主要是html中行为，他可以让html页面更加有动态感。只有这样，才能更好结合node进行渲染数据。Js是一种动态类型、弱类型、基于原型的语言[12]。可以给html网页增加动态功能前端因此也有许多流行的框构例如vue.js,angular.js,react.js等。该项目就是基于vue.js。vue2.0是一个相对稳定的版本，源码简介，在不操作dom的基础上，通过双向数据绑定，把注意力主要集中在解决问题上。在后端使用了node,express。相对来说，很多大公司以及在使用node了。Node有自己的社区，并且已经很完善，可以很放心的使用。

因此从技术上来看，实现web导购系统是完全可行的。

### 2.2.3 操作可行性

导购系统从操作可行性分析，主要来看，是否方便操作，本系统在开发完成后，在其他电脑上运行是否可以，达到用户的需求。即便出现了问题，也要快速的用现代的技术把他解决，让用户可以尽快的体验到该系统。

除此之外，管理者应该具备一定的计算机知识，只有这样才可以更快的上手。本系统可以在windows，ubuntu，mac下的任意浏览器使用，界面友好，操作便捷。

因此从操作上来看，实现web导购系统是完全可行的。

### 2.2.4 法律可行性

本系统的开发和设计过程完全是个人独立完成出来的，虽然在此过程中遇到了不少的问题，但都通过查阅书籍以及寻求老师和同学的帮助解决了问题，所使用的软件是正版授权的。并且所使用的开发技术都可以在相应的官网找到，不存在侵权的问题。

从以上来看，web导购系统在法律上是可行的。

## 2.3 方案确定

根据以上几种方案的对比分析，结合我所掌握的知识，以及我在平时学习工作中的对它们所掌握的程度，最终选择了用vscode作为编译器，node作为服务器端开发语言，以express作为后端开发框架，以MySql作为数据库，以vue.js来开发前端，即方案一。

## 2.4 相关技术介绍

### 2.4.1 主要语言

在本系统开发中主要使用了，vue和node。Vue.js是一个构建数据驱动的 web 界面的渐进式框架。Vue.js 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。它不仅易于上手，还便于与第三方库或既有项目整合。Vue是简单小巧的核心，渐进式技术栈，足以应付任何规模的应用。

Node.js 是一种建立在Google Chrome's V8 引擎上的non-blocking(非阻塞),event-driven(基于事件的)I/O平台Node.js平台使用的开发语言是JavaScript，平台提供了操作系统底层的API，方便做服务器端编程。

### 2.4.2 主要数据库

本系统的主要使用MySql开发，MySql是一个小型的数据库，其体积小、速度快、成本低，尤其是开放源码这一特点，在很多公司都有使用[4]。Mysql有以下优势：

速度快；

连接性和安全性；

可移植性；

它支持SQL语言；

容易使用；

成本优势。

### 2.4.3 主要开发工具

开发本本系统主要使用VSCode，VSCode 是一个可以运行于Mac、windows和linux之上的，现代化跨平台编译器。该编译器，可以满足我使用命令运行程序，以及自动化检查代码，让代码以一种高亮的方式显示。方便查看和修改。

# 第3章 需求分析

## 3.1 功能需求分析

通过可行性分析，确定该系统的使用用户，以及各个模块的功能和意义之后，本系统主要是在商场中方便用户进行购物。主要包括：商场人员信息管理模块、商品信息管理模块、分类商品销售位置查询模块、商品打折促销模块、商场活动推介模块、商场介绍,品牌商铺信息。以下是我的分析。

1．商场人员信息管理模块：主要给商场中管理人员，提供雇员的详细信息。

2．商品信息管理模块：商场中的管理员可以对自己店里的商品进行管理，主要是增删改查商品。并且可以及时的添加一系列商品最新的消息，商品打折促销信息等。

3．分类商品销售位置查询模块：这一块主要是给每个商场中商铺按售卖的商品类型进行分类，然后把商品的详细位置保存下来，消费者可以根据自己想要的商品查询到该商品的位置。

4. 商品打折促销：这类商品主要是雇员发布商品的消息。商品促销打折等。然后用户可以及时的看到该类消息。

5．商场活动推介模块：主要是商铺与顾客的一场互动，类似于现实中的商铺对商品的介绍，宣传来达到吸引用户的目的。

6. 商场介绍：主要是介绍这个商场中的详细信息，以及各个楼层的详细信息，以及该商场的主要经营的业务，那个方面是强项等。

7. 品牌商铺信息：主要是介绍这个商铺详细信息，以及各个主要的活动等。

## 3.2 性能需求分析

本系统界面友好，反应速度快，可以很好的满足用户的体验，操作便利，方便让管理人员很快的上手。在设计上可以切实的符合用户心里的需求。也可以把各项数据保存在安全可靠的数据库中。不仅可以给消费者节省购物时间，还可以提高管理的质量。

## 3.3 系统数据流图和数据库需求分析

### 3.3.1 系统数据流图

1．顶层数据流图如图3-1所示。



图3-1 系统顶级数据流图

### 3.3.2 数据库需求分析

管理员可以通过web商场导购系统，对商场、商品等信息进行管理。例如：修改查询商品信息。完成这个的前提是，数据库可以满足这类的要求。通过各种分析和查阅资料。收集各类数据以及结构，组成数据字典，为数据库的完善做准备。

通过系统的各个部分的功能分析，总结出如下需求信息：

1．用户可以分为用户（消费者）和管理员（店员）

2．一个用户可以同时查看多个商品和商场的消息，一个商品和商场可以被多个用户查看。

3．一个用户可以查看各个楼层的信息，一个楼层可以被所有的用户查看。

4．一个用户可以查看多个打折促销产品，一个打折促销产品可以被多个用户查看。

5．一个用户可以查看多个商铺活动推介，一个商铺活动推介可以被多个用户查看。

通过以上系统功能分析和需求总结设计如下所示的数据项和数据结构。

1．管理员信息表（adminsql），包括数据项： 用户名，密码，性别，邮箱等。

2．商品信息表（goodssql），包括数据项：商品的名称，商品的简介，商品的价格，商品的图片，位置，发布人，发布时间。

3．商场信息表（marksql），包括数据项：商场简介，商场的位置，商场中的各个楼层介绍。

4．打折促销表（discountsql），包括数据项：发布时间，，折扣价，位置等。

5．品牌商铺信息表（merchantsql），包括数据项：商铺位置，商铺名称等。

## 3.4 数据字典

通过对数据字典的设计，可以得到如下。

1．名字： 管理员表

描述：该系统所委派的管理员

定义：管理员信息=管理员编号+管理员昵称+管理员密码+管理员头像+管理员性别+管理员邮箱。

2．名字： 商品信息表

描述：管理员所添加的商品信息

定义：商品=商品名称+价格+商品描述+商品销量+上架时间+发布商铺+商品位置。

3．名字： 商场信息表

描述：主要介绍该商场的信息

定义：商场信息=商场的名称+商场的主营业务+各个楼层的信息+商场位置。

4．名字： 打折促销信息表

描述：打折促销的商品信息

定义：打折促销信息=打折促销商品名+发布时间+发布楼层+发布人员+发布的商铺+促销价格+商品位置。

5．名字： 品牌商铺信息表

描述：商铺的信息

定义：商铺信息=商铺的位置+商铺的主营业务+商铺名称+活动+视频介绍+推广。

## 3.5 系统E-R图

E-R图也是实体关系图，主要是通过图形把该系统中各个模块的功能进行详细的列举出来。通过矩形表示实体型，矩形框内写出实体名，用椭圆形状表示实体的属性，并用无向边将其连接起来，通过菱形表示实体之间的联系，用无向边分别与有关实体型连接起来，同时在无向边旁标上联系的类型（1:1,1：n,1：m）[13]。

1.系统E-R图，如图3-2所示

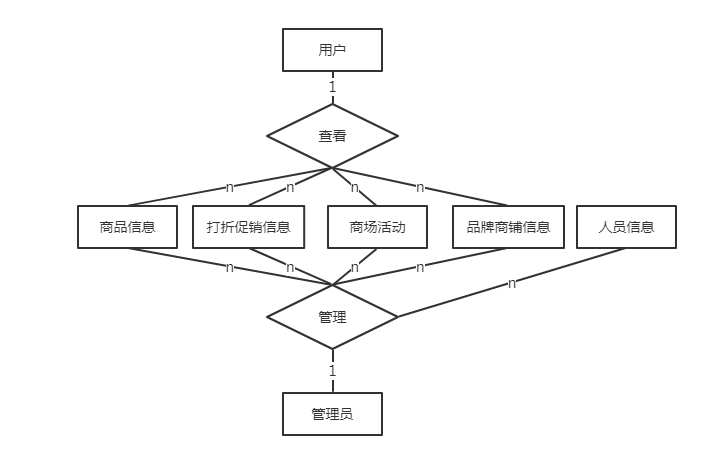


图3-2 系统E-R图

2．管理员的实体属性图，如图3-3所示

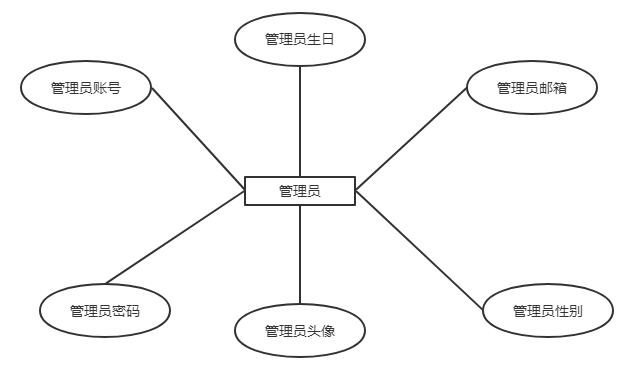
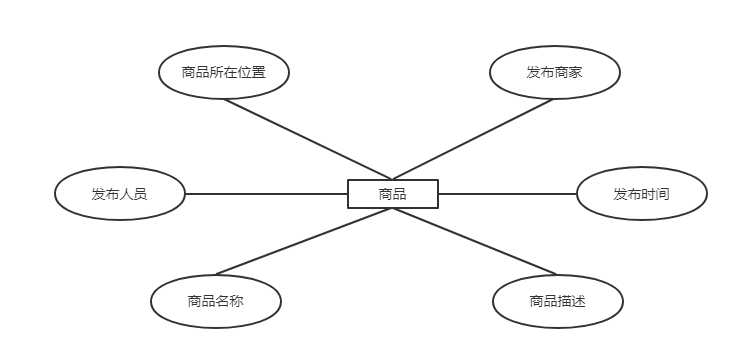


图3-3 管理员实体图

3．商品实体属性图，如图3-4所示



3-4 商品实体图

4．商场实体属性图，如图3-5所示

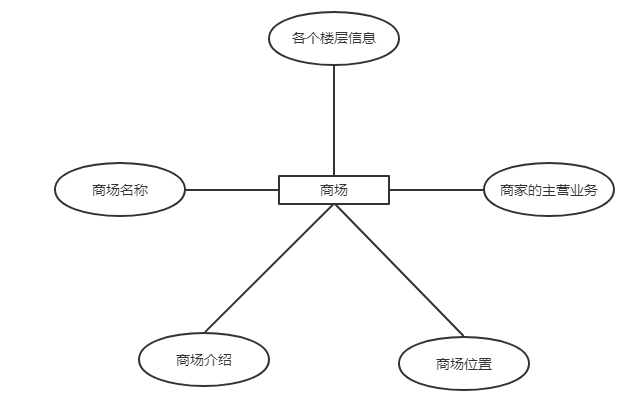


图3-5 商场实体图

5．品牌商铺实体属性图，如图3-6所示

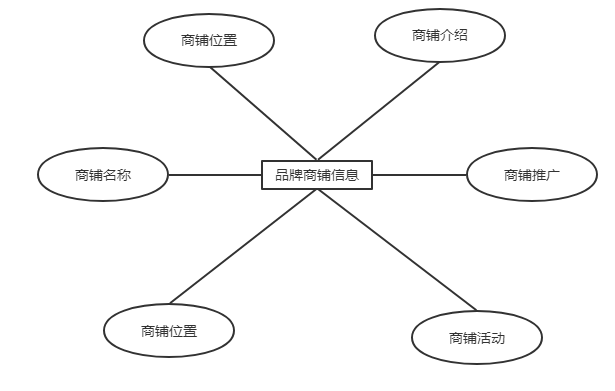


图3-6 品牌商铺实体图

6．打折促销实体属性图，如图3-7所示

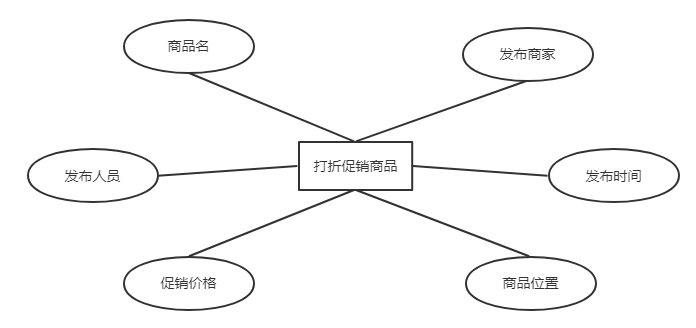


图3-7 打折促销实体图

## 3.5 逻辑模型

E-R图转化为关系模型实际上就是要将实体型、实体的属性和实体型之间的联系转换为关系模式。根据一个实体型转化为一个关系模式，实体的属性就是关系的属性，实体的码就是关系的码的转化原则，关系模式管理员、商铺、打折促销品、商场、商品，商品位置，不存在非主属性对主属性的部分函数依赖，也不存在传递函数依赖，已经达到了3NF。可以将上图的全局E-R图转化为如下关系模型（实体的码用下划线标出）。

1.管理员（管理员id，名字，密码，性别，邮箱）

2.商品（商品id，商品的名称，商品的简介，商品的价格，商品的图片，位置，发布人，发布时间）

3.商场（商场id，商场简介，商场的位置，商场中的各个楼层介绍等）

4.打折促销（打折促销id，发布时间，，折扣价，位置等）

5.品牌商铺（商铺id，商铺位置，商铺名称,商铺活动等）

## 3.6 系统用例图

用例图是由用、角色、系统边界组成的。用例图是表示用户，可以通过该系统可以进行的所有操作[13]。可以帮助你分析系统有那些功能，以及让你清楚的知道系统的内外部（角色）的交互。

本系统的用例图，如图3-8所示

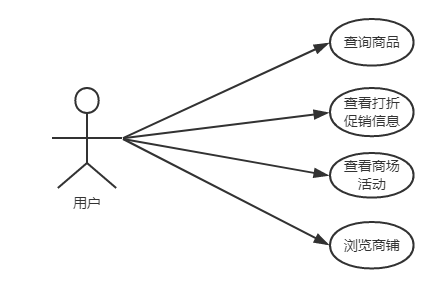


图3-8 系统用例图

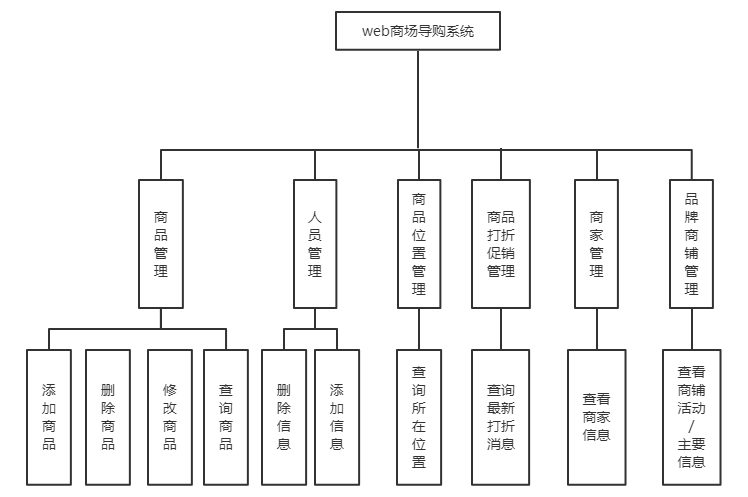
# 第4章 过程论述

## 4.1 概要设计

### 4.1.1 总体功能设计

本商场导购系统主要目的为用户和商铺提供方便、快捷的服务，方便管理人员对雇员、商品、商品打折促销、商场活动推广、商品查询等进行管理。本系统具体的功能如下。

整个web商场导购系统主要围绕跟商场相关的各个方面信息以及使用的权限不同总共有以下相对独立的模块。主要从商场，商品，以及人员进行管理。如图4-1所示。

图4-1 web商场导购系统功能结构图

## 4.2 数据库详细设计与实现

本系统采用了被广泛使用的数据库管理系统MySql，数据库名称为“market”。数据库“market包含以下”5个表：商品表，管理员表，打折促销表，商场表，品牌商铺表。

### 4.2.1 创建数据库

本系统通过node进行开发，使用navicat来进行数据库管理，它可以为用户提供一个数据库的界面，在打开Navicat之后创建一个新数据库“market”，如图4-2所示。

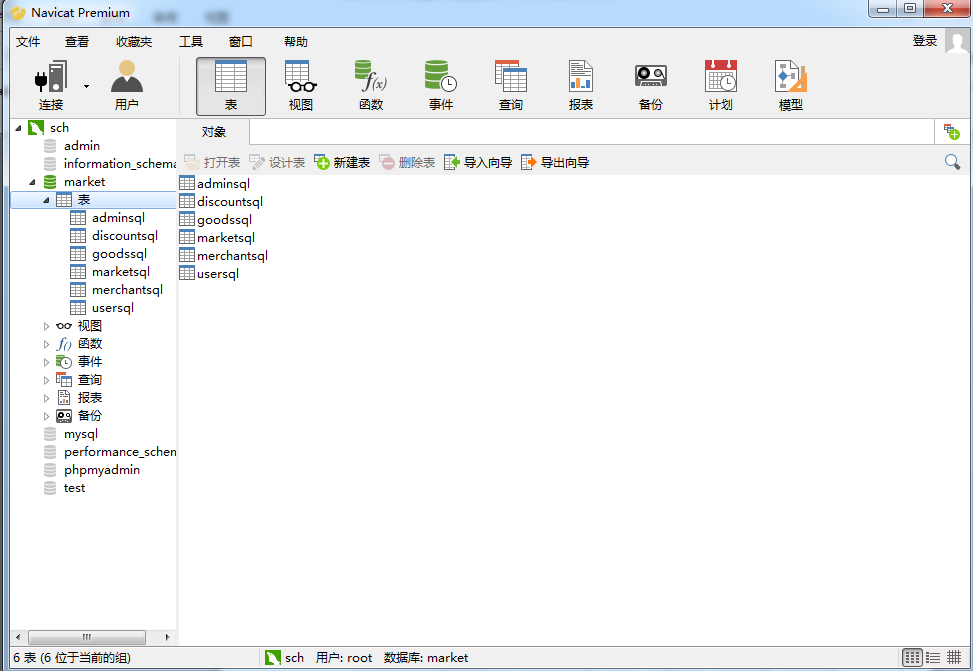


图4-2 Navicat界面图

### 4.2.2 创建表和字段

根据第3章对数据库的概念模型以及逻辑模型设计的理解，结合分析实体属性图和E-R图可以规划出web商场导购系统的具体数据库表，详细设计如下所示：

1．管理员信息表(adminsql)

管理员信息表用于存储管理员的信息。主要包括管理员编号，管理员账号，管理员密码，管理员性别，管理员邮箱，管理员生日。其中管理员编号作为主键。该数据库设计表如4-1所示。

表4-1 管理员信息表(adminsql)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **编号** | **字段名** | **数据类型** | **长度** | **是否为空** | **是否为主键** | **字段说明** |
| 1 | aid | 自动编号 | 4 | 否 | 是 | 管理员编号 |
| 2 | aname | 字符 | 30 | 否 | 否 | 管理员账号 |
| 3 | apwd | 字符 | 30 | 否 | 否 | 管理员密码 |
| 4 | aimg | 字符 | 100 | 否 | 否 | 管理员头像 |
| 5 | asex | 整型 | 1 | 否 | 否 | 管理员性别 |
| 6 | aemail | 字符 | 50 | 否 | 否 | 管理员邮箱 |

3．商品信息表(goodssql)

商品信息表主要存放发布的商品信息。主要包括商品编号，商品名称，商品描述，商品销量，发布时间，发布人员，商品位置。其中商品编号，作为主键。该数据库设计表如4-2所示。

表4-2 商品信息表(goodssql)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **编号** | **字段名** | **数据类型** | **长度** | **是否为空** | **是否为主键** | **字段说明** |
| 1 | gid | 自动编号 | 4 | 否 | 是 | 商品编号 |
| 2 | gname | 文本 |  | 否 | 否 | 商品名称 |
| 3 | gdescribe | 文本 |  | 否 | 否 | 商品描述 |
| 4 | gdate | 日期 |  | 否 | 否 | 发布时间 |
| 5 | gcomsign | 字符 | 255 | 否 | 否 | 发布人员 |
| 6 | glocation | 字符 | 255 | 否 | 否 | 商品位置 |

4．商场信息表(marketsql)

商场信息表主要存在，该导航系统已经进入的商场。主要包括商场编号，商场名称，商场介绍，楼层信息，主营业务。其中商场编号，作为主键。该数据库设计表如4-3示。

表4-3 商场信息表(marketsql)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **编号** | **字段名** | **数据类型** | **长度** | **是否为空** | **是否为主键** | **字段说明** |
| 1 | gid | 自动编号 | 4 | 否 | 是 | 商场编号 |
| 2 | gname | 字符 | 32 | 否 | 否 | 商场名称 |
| 3 | gintroduce | 文本 |  | 否 | 否 | 商场介绍 |
| 4 | ginfo | 文本 |  | 否 | 否 | 楼层信息 |
| 5 | gimportant | 字符 | 255 | 否 | 否 | 主营业务 |
| 6 | glocation | 字符 | 255 | 否 | 否 | 商场位置 |

5．打折促销信息表(discountsql)

打折促销信息表主要存放关于打折促销商品的各项信息。主要包括促销编号，促销名称，发布人员，发布日期，促销价格，发布人员，商品位置。其中促销编号，作为主键。并且，促销的商品，是已经发布的商品。该数据库设计表如4-4所示。

表4-4 打折促销信息表(discountsql)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **编号** | **字段名** | **数据类型** | **长度** | **是否为空** | **是否为主键** | **字段说明** |
| 1 | did | 自动编号 | 4 | 否 | 是 | 促销编号 |
| 2 | dname | 字符 | 32 | 否 | 否 | 促销名称 |
| 3 | duser | 字符 | 32 | 否 | 否 | 发布人员 |
| 4 | ddate | 日期 |  | 否 | 否 | 发布日期 |
| 5 | dprice | 整型 | 8 | 否 | 否 | 促销价格 |
| 6 | dcomsign | 字符 | 255 | 否 | 否 | 发布人员 |
| 7 | dlocation | 字符 | 255 | 否 | 否 | 商品位置 |

6．品牌商铺信息表(merchantsql)

商铺信息表主要存放平台管理员发布的关于商品的各项信息。主要包括商铺名称，商铺位置，商铺主营业务，商铺活动，商铺推广，商铺介绍。该数据库设计表如4-5所示。

表4-5 商铺信息表(merchantsql)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **编号** | **字段名** | **数据类型** | **长度** | **是否为空** | **是否为主键** | **字段说明** |
| 1 | mid | 自动编号 | 4 | 否 | 是 | 商铺编号 |
| 2 | mname | 字符 | 32 | 否 | 否 | 商铺名字 |
| 3 | mvideo | 字符 | 11 | 否 | 否 | 商铺介绍 |
| 4 | mactive | 字符 | 255 | 否 | 否 | 商铺活动 |
| 5 | mactive1 | 字符 | 255 | 否 | 否 | 商铺推广 |
| 6 | mimportant | 字符 | 255 | 否 | 否 | 主营业务 |
| 7 | mlocation | 字符 | 255 | 否 | 否 | 商铺位置 |

## 4.3 数据库连接

Node中连接数据库，我们只需要使用公共的Db类进行操作，在连接数据库的时候。我们需要安装MySql的npm包，这个是官方提供的并且及其稳定的包。连接如下所示：

module.exports = {

mysql: {

host: 'localhost',

user: 'root',

password: '',

database: 'market',

port: '3306'

} }

var conn = mysql.createConnection(models.mysql);

conn.connect();

var models = require('../db/db');

var express = require('express');

var router = express.Router();

var mysql = require('mysql');

var $sql = require('../db/sqlMap');

var conn = mysql.createConnection(models.mysql);

conn.connect();

## 4.4 详细设计

### 4.4.1 系统登录

登录系统分为管理员和超级管理员（admin）管理员只能由admin分配，系统在登录的的时候通过输入的用户名进行判断管理员和admin。如图4-3所示。



图4-3 系统登录界面图

管理员可以通过系统登录界面，进入系统后台，让后可以对各类商品，商铺，以及商场信息进行管理。当点击取消之后，即是用户进行登录，用户可以访问该商场中一系列最新的打折消息，并且可以查看商场中的活动，可以进行访问该商品所在的位置。

代码如下

// 可以完成登录

router.post('/login', function (req, res) {

let params = req.body;

conn.query('select aname from adminsql where aname="' + params.username + '" and apwd="' + params.password+'"', function (err, rows) {

if (err || rows.length == 0) {

console.log(err)

res.send({code: 0})

}else {

res.send({code: 1})

console.log(rows)}})})

### 4.4.2 主界面设计

主界面主要是该系统的主页，主页包括该商场中的平面图，以及商铺的活动推广，和查询打折促销的消息。如图4-4所示

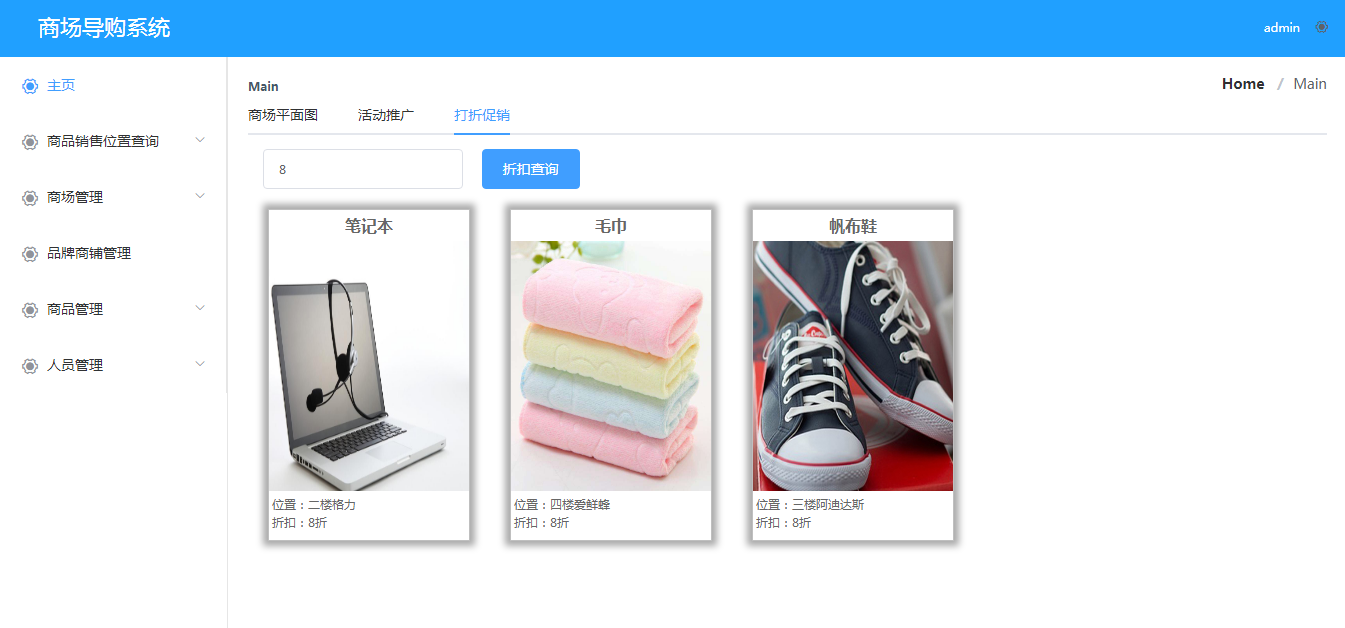


图4-4 系统主界面效果

用户和管理员都可以访问该页面，该页面主要是商铺获取商铺发布的最新促销活动以及，通过一个小视频来介绍，你所访问的商铺的信息。并且可以在打折促销活动中查询打折的产品，然后用户点击该商品就可以弹出该商品所在的位置。

代码如下

queryDiscount() {

let \_this = this;

let url = '/api/query\_discount';

this.$http.post(url, {

discount: \_this.qdiscount,

},{}).then(function(data) {

let info = data.data.data;

if(info.length > 0 ) {

\_this.tableData = info;

} else {

this.$message.warning('该项数据为空')

\_this.tableData = info;

} })

},

### 4.4.3 人员管理模块

人员管理模块，主要是对管理员进行管理的模块。如图4-5所示。

该模块主要是对管理员的信息进行修改，不过仅仅可以删除管理员的信息，当然管理员可以在个人页面修改个人信息。超级管理员可以进行添加该商场中的管理员即雇员。如图4-5所示。

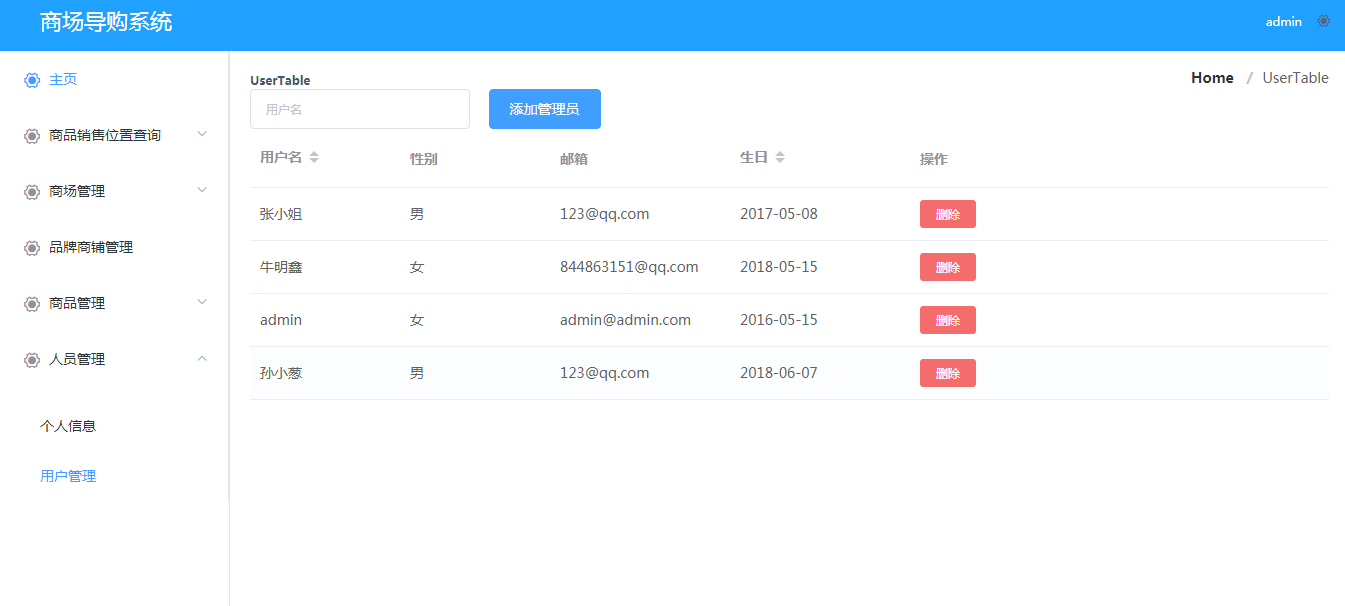


图4-5 人员信息管理界面

该模块主要是admin进行操作，他可以对各个商铺中的管理员进行添加和删除，添加管理员拥有默认的密码。当管理员登录时可以在个人信息页面进行修改个人信息，当然没有对其他人员操作的权限。

代码如下

addAdmin() {

let \_this = this;

console.log(this.uname)

var url = '/api/add\_user';

this.$http.post(url, {

username: \_this.uname,

sex: \_this.sex,

birth: \_this.birth,

pwd: this.pwd

},{}).then(function(data) {

console.log(data.body.code);

if(data.body.code == 0) {

this.$message.error('用户已存在');

} else {

this.$message({

message: '添加成功',

type: 'success'

});

\_this.queryUser();

}})},

### 4.4.4 商品信息管理/查询模块

管理员登录系统时可以可以对商品信息进行管理的查询。如图4-6所示。



图4-6 商品信息管理/查询界面

该模块主要是管理员登录后，查看到已经存在的商品，当某个商品已经过时，管理员可以对其删除，管理员也可以对该类商品信息，进行修改。并且管理员也可以在添加商品页面添加所需要的商品。

代码如下

// 查询商品信息

router.get('/sel\_goods', function (req, res) {

let params = req.body;

conn.query('select \* from goodssql', function (err, rows) {

console.log(rows)

if (err || rows.length == 0) {

console.log(err)

res.send({data: 0})

}else {

res.send({data: rows})

console.log(rows)

}

})

})

// 修改商品信息

router.post('/upd\_goods',function(req, res) {

let params = req.body;

conn.query('update goodssql set gname = "'+params.name + '" ,gdescribe = "' + params.describe + '",gprice = "' + params.price + '" ,glocation = "' + params.location + '"where gid = "' + params.gid + '"', function(err, result) {

!err ? res.send({code: 1}) : res.send({code: 0})

})

})

### 4.4.5 商场管理模块

用户和管理员都可以访问商场管理页面，商场管理页面主要是已经入驻的大型商场。如图4-7所示。



图4-7 商场管理页面

该模块主要是已经入驻该系统的商场，通过访问该商场主页就可以访问该商场中各个楼层的信息。

代码如下

// 查询商场

router.get('/query\_market', function(req, res) {

let params = req.body;

conn.query('select \* from marketsql', function(err, rows) {

if (err || rows.length == 0) {

console.log(err)

res.send({data: 0})

}else {

res.send({data: rows})

}

})

})

当让问该商场主页是就可以看到各个楼层的信息，也可以看到商场主页最新的商品信息。如图4-8所示。

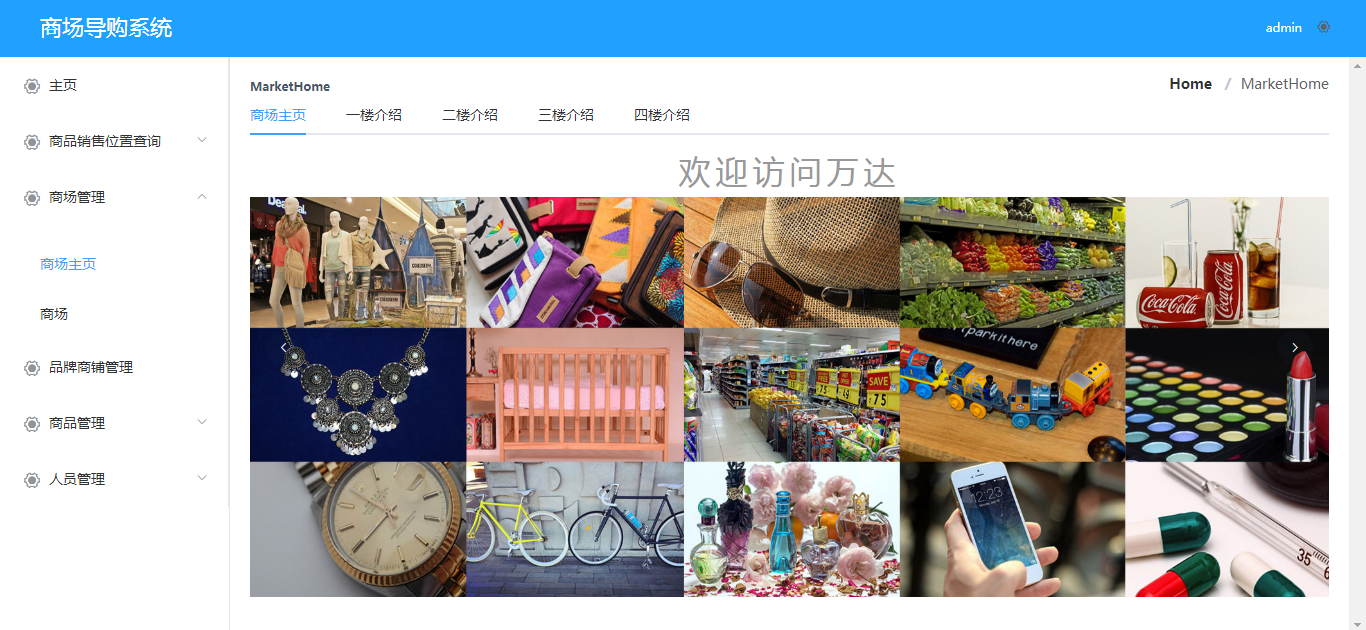


图4-8 楼层介绍

该模块主要用户介绍这个商场，以及各个商场所包含的各个楼层的信息。并且用户可以查看到，该商品所在的位置。

代码如下

// 查询商场

router.get('/query\_market', function(req, res) {

let params = req.body;

conn.query('select \* from marketsql', function(err, rows) {

if (err || rows.length == 0) {

console.log(err)

res.send({data: 0})

}else {

res.send({data: rows})

}

})

})

### 4.4.6 商场活动推介模块

商场活动推介模块主要通过一个简短的视频，用来介绍这个某个商铺的详细信息，然后会有最新的促销活动展示出来。如图4-9所示。

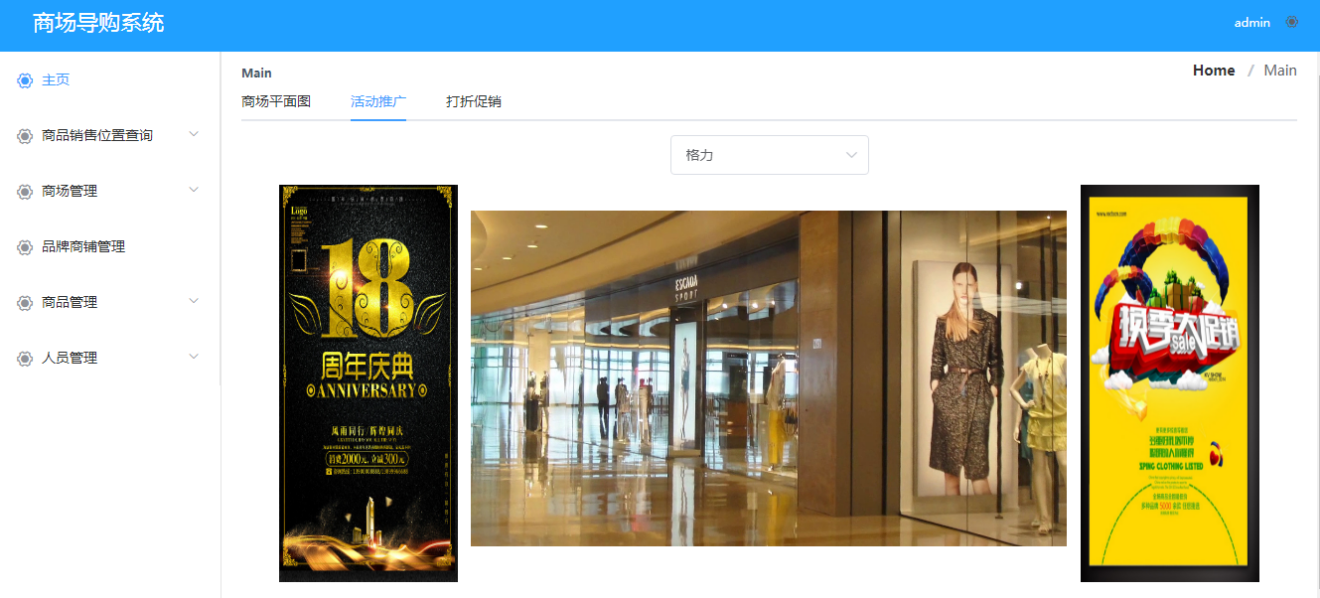


图4-9 商场活动推介页面

该模块主要是已经入驻的商铺，来进行发布一些促销活动和介绍该商铺，主要是用来达到吸引用户的目的。

代码如下

getActive() {

console.log(this.value)

let \_this = this;

let url = '/api/query\_avtive';

this.$http.post(url, {

cname: \_this.value,

},{}).then(function(data) {

console.log(data)

let info = data.data[0];

this.tableData1 = info;

\_this.active = info.cactive;

\_this.active1 = info.cactive1;

\_this.video = info.cvideo;

})

}}

# 第5章 软件测试与结果分析

软件测试是软件发布后不可或缺的一个环节，一个好的系统是必须可以经受的住的测试的。只有经过测试的系统才可以放心的让用户进行使用。软件测试必须要进行全面的考虑，把系统中所有的路径流向都要进行测试到。在现有的条件下，发现程序中出现的错误，是可以衡量软件质量的重要途径，并且判断该软件是否满足该需求分析要求的重要过程[14]。

## 5.1 测试的目的和原则

软件测试的目的是为了让软件在为投入使用之前及时的发现其中的错误，（当然发现错误不是唯一目的）。我们可以从用户和软件开发着角度来看软件测试。从用户角度看，看待该软件是否出现太多错误，来确定是否要使用该软件。从开发人员来看，软件测试是希望自己开发的产品出现较少的错误，可以极大的满足用户的需求，并且可以被用户所接受使用。通过测试还可以发现软件设计的不足，并且把这些不足进行改正，就可以让自己的软件更加完善。

本系统主要使用白盒测试，白盒测试又称结构测试和透明盒测试，逻辑驱动测试或基于代码的测试。白盒测试法的覆盖标准有逻辑覆盖、循环覆盖和基本路径测试。其中逻辑覆盖包括语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖和路径覆盖[15]。

## 5.2 测试计划

测试计划是软件测试需要经过的一个步骤，软件测试计划表如5-1所示。

表5-1 系统测试计划表

|  |  |
| --- | --- |
| **目标** | 系统可以按照需求完成各个功能 |
| **内容** | 系统的登录、商品管理模块、商场管理模块、销售位置查询模块、打折促销管理模块、品牌商铺管理模块 |
| **技术** | 白盒测试与黑盒测试，本系统主要应用白盒测试 |
| **前提** | 系统安装运行 |
| **完成标准** | 功能基本实现 |
| **特殊事项** | 权限分配 |

## 5.3 测试过程

因为本系统主要是使用的白盒测试，并且单元测试，集成测试，系统测试是白盒测试所包含的内容[15]。本次测试，主要是对该系统的基本需求进行测试。以下是测试的详细过程。

### 5.3.1 单元测试

单元测试是对系统中一个小的模块进行测试，查看该模块的功能是否实现。通常是在某个特定条件下，进行测试该功能是否可行。通过单元测试，每一个路径都进行执行一次。如图5-1所示。

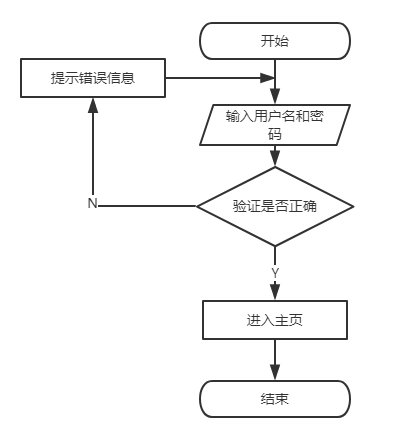


图5-1 登录模块流程图

对应的流程图如图5-2所示：

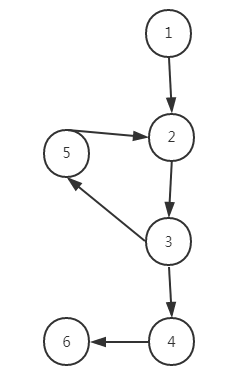


图5-2 登录模块流图

根据流图确定环形复杂性度量V(G)

V(G)=E-N+2p=6-6+2=2

确定基本路径可以确定几条独立的路径：

路径1: 1-2-3-4-5

路径2：1-2-3-4-5-3…

让以上的每一条路径都执行一次，

1．路径1的测试用例

输入正确账号和密码。

期望结果：登陆成功，进入系统管理首页。

2．路径2的测试用例

输入正确账号和错误密码。

期望结果：提示登录失败

### 5.2.2 集成测试

集成测试也称为组装测试或者联合测试，是在单元测试的基础上，将所有模块按照设计要求组装成子系统或者系统，进行测试，在把每个要测试的模块组合到一起时，随之可能出现的错误也许都是与每个单元模块之间的接口连接有着一定的关系，集成测试保证了信息在各个模块之间的畅通[15]。在此基础上，即可进行接下来的系统测试。

### 5.2.3 系统测试

启动电脑windows7+操作系统，开启xampp启动mysql来保证可以连接到数据库，进入项目运行vue（npm run dev），运行server（node app.js），在浏览器中输入地址localhost:8081/#/login（localhost）为网站根目录，login为网站的登录页，8081为端口），访问本系统进行各项功能的测试。

在后端代码测试完整之后，然后进入前端项目，通过npm run dev启动。然后，前后端进行结合，对整体的一系列功能进行测试。测试结果如表5-1所示。

表5-1 整体测试表

|  |  |  |  |
| --- | --- | --- | --- |
| **测试用例** | **操作** | **预期结果** | **实际结果** |
| 登录 | 填写完整/正确信息 | 可以进行提交并且实现 | 功能可以实现 |
| 商场管理 | 访问最新的活动/商场介绍 | 可以看到最新的消息以及商场的完整介绍 | 功能可以实现 |
| 商品 | 查询商品的信息/查看商品位置 | 可以看到商品以及位置 | 功能可以实现 |
| 打折促销 | 可以访问商品/位置 | 可以看到最新的打折促销商品 | 功能可以实现 |
| 品牌商铺 | 访问商铺信息以及位置 | 可以得到完整消息以及位置 | 功能可以实现 |
| 管理员 | 管理管理员信息 | 可以删除/添加信息 | 功能可以实现 |

## 5.4 测试结论

通过单元测试的结果每个模块在逻辑上是可行的，通过集成测试可以说基本上没有接口的错误，软件基本可用。在测试过程中，对遇到的错误进行修改，完善，直到该错误不在对本系统有任何影响，即完全改正遇到的错误。本系统从选题到测试结束已经完成了毕业任务书的要求。从我个人来开，基本上达到了我预期的目标，从用户角度来看，也可以满足用户的基本要求，相对来说该系统在之后的维护也比较方便。因此此次开发也是比较成功的。但是作为一个优秀的程序员，在以后的开发过程中，要不断学习，不断开发，不断测试。

# 结 论

Web商场导购系统，一方面是为了满足消费者在大型商场中购物的方便，可以及时的让消费者找到自己心仪的商品，另一方面，可以让商场管理人员，可以很方便的对商品进行管理，对打折促销消息可以及时的传递给用户，让用户可以第一时间，寻找商品。该系统在商场中应用场景十分广泛并且可以很大发挥其实用和实际的价值，

在商品推广方面节约了，人力物力，相对于传统的方式，该系统不仅保护了环境，还打破了传统的习惯，有利于推动科技的进步。此外，该系统使用起来比较简单，并且版本很稳定，系统界面友好。该系统在VSCode下进行开发，吧数据存储在mysql中，具有安全性，符合设计的基本要求。

总体来说，通过本次毕业设计，对我个人软件设计上来说有了很大的提高。本次毕设从选题到设计都经过了多次的思考。在设计之初我在网上和图书馆查阅了很多资料，并且询问一些消费者和大型商场中的人员，最终确定该系统所需要存在的几大模块，然后在把该系统的各个模块进行细化，并且完善到需求分析和概要设计上，把这些工作做好之后，然后再进行前端页面的设计，页面完成后，就要进行代码开发。所有的代码，都是通过之前的积累，以及平常的学习，进行敲上去的。虽然在开发过程中遇见了很多问题，但这些问题并不能阻挡我前进的步伐，每次遇见问题，我都会询问同学，咨询老师，网上查阅资料。每一次的解决问题，都让我更清楚的认识自己的优点和缺点，并且取长补短，让自己有更大的提高。此次毕设，让我更清楚的认识到，团队精神和沟通学习能力是必不可少的一项技能。

虽然本次毕业时间比较短暂，但我却学到了很多东西，但是我还能感觉到个人还有许许多多有待于提高和进步的地方，商场导购系统，还需要后期不断的完善。从软件的开发到发布，测试是不可缺少的一个环节，但是系统不可能做到十分完美，所以需要我们不断的学习，并且利用新的知识来完善该系统。但是完善和维护该系统的同时，要考虑到维护该系统的成本以及商业价值。由于该系统是本人自己设计和开发的，出于个人学习的考虑，会对该系统不断的完善。

商场导购系统是一个很好的开发项目，具有很高的社会实用价值，虽然这个系统是一个毕业设计，但也存在很多不足之处。因此不断的学习，才能让自己提高。

# 致 谢

通过web商场导购系统的开发，让我学习到了很多知识，并且让我更加清楚的认识了软件开发过程的思想。在此我诚挚的感觉大学无私奉献的老师，以及给予我帮助的同学。因为有了他们，我才能走的更高，看的更远。

商场导购系统是在贺薪宇老师指导下完成的。从商场导购系统的选择到确定再到最后实现，贺薪宇老师一直都是我坚强的后盾。在学习和生活中给予了我无私的帮助以及悉心的指导，从一个细小的问题，再到比较严重的问题，贺薪宇老师都非常耐心的一遍一遍的指导我毕设和论文，才最终有了近乎完美的毕设和论文。贺薪宇老师从大一就是我的老师。并且她是一位非常年轻优秀的老师，她知识渊博，严谨负责，认真对待每一件事。正是因为这种优秀的品质深深的影响了我。让我以更长远的眼光看待问题。除此之外，我同学，也在毕业设计和论文上给予了我很大的帮助，由于实习的方向不一样。在测试的时候，要请教实习过测试的同学，该如何进行完美的测试。在调试代码的时候，同学也会帮助你。正因为有了他们的帮助，才让我的程序，能够完善。在此，要深深的感谢我的同学。通过本次论文，让我学习了很多知识，也增加了很多经验。

最后，再次深深的感谢我的同学和老师。

# 参考文献

1. 姚禕炜. 《购物广场互动导购》解决方案[J].科技风，2013，16期.

[2] 朴灵. 深入浅出Node.js[M]. 北京：人民邮电出版社，2013-12-01，1-150.

[3] 未知. Express 4.x Node.js的Web框架[Z]. 论文库.

[4] 宁海元，周振兴，彭立勋，翟卫祥，刘辉. 高性能MySQL[M]. 北京：电子工 业出版社，2013-5-1，1-150

[5] 明日科技. Java从入门到精通[M]. 北京：清华大学出版社，2016-10-1，2-50.

[6] 程显峰. MongoDB权威指南[M]. 北京：人民邮电出版社，2011-5-1，1-50.

[7] 李晓明. C#与. NET 4高级程序设计[M]. 北京：人民邮电出版社,2011，5-200.

[8] 戴志诚. SQL Server 2005 数据库系统开发与实例[M]. 北京：电子工业出版社，2011，3-60.

[9] 田阳，王成东，崔嵩. 项目可行性研究[M].哈尔滨工业大学出版社，2015-06，4-80.

[10] 李杰，柳靖，刘淼. HTML5高级程序设计[M].人民邮电出版社，2011-1-1，3-50.

[11] 林小志.CSS那些事儿[M].电子工业出版社[M]，2009-10，6-59.

[12] 杨涛，王建桥，杨晓云.JavaScript DOM编程艺术[M]，2006-12，6-65.

[13] 冀振燕. UML系统分析与设计教程[M]. 北京：人民邮电出版社，2014-8，20-150.

[14] 王玉玺. 软件测试──提高软件可靠性的关键手段航空计算技术[M]. 北京：电子工业出版社，2011，1-70.

[15] 佟伟光，郭菲菲. 软件测试[M]. 北京：人民邮电出版社，2015-1，37-199.

[16] 方木云，刘辉. 高级软件工程[M]. 北京：清华大学出版社，2011-6.

[17] Shelley Powers. Learning Node[M]. O'Reilly Media,2012-7.

[18] Addy Osmani. Learning JavaScript Design Patterns[M]. O'Reilly Media,2012-8-22.

# 附 录

### 软件使用说明

1.安装该系统所需要运行的环境

(1)安装node，vue开发环境。

(2)安装XAMPP（用于启动MySql数据库）。

(3)安装Nginx（用户作为资源服务器）。

2.打开代码文件，可以在本机调试或者上传到服务器

(1)进入前端页面开发目录通过npm run dev运行前端页面。

(2)进入后端开发目录通过 node app.js 运行后端

3.打开浏览器输入（http://localhost:8081/#/）进入访问该系统

### 主要源代码

var models = require('../db/db');

var express = require('express');

var router = express.Router();

var mysql = require('mysql');

var $sql = require('../db/sqlMap');

var conn = mysql.createConnection(models.mysql);

conn.connect();

// 可以完成登录

router.post('/login', function (req, res) {

let params = req.body;

conn.query('select aname from adminsql where aname="' + params.username + '" and apwd="' + params.password+'"', function (err, rows) {

if (err || rows.length == 0) {

console.log(err)

res.send({code: 0})

}else {

res.send({code: 1})

console.log(rows)

}

})

})

// 通过用户名查找用户

router.post('/query\_uname', function(req, res) {

let params = req.body;

conn.query('select \* from adminsql where aname = "' +params.aname +'"', function(err, result) {

if (err ) {

res.send({data: 0})

}else {

res.json(result)

console.log(result)

}

})

})

// 修改个人信息

router.post('/upd\_admin',function(req, res) {

let params = req.body;

conn.query('update adminsql set aname = "'+params.name + '" ,apwd = "' + params.pwd + '",asex = "' + params.sex + '" ,aemail = "' + params.email + '" ,abirth = "' + params.birth +'"where aid = "' + params.aid + '"', function(err, result) {

!err ? res.send({code: 1}) : res.send({code: 0})

})

})

// 查询用户

router.get('/query\_admin', function(req, res) {

let params = req.body;

conn.query('select \* from adminsql', function (err, rows) {

console.log(rows)

if (err || rows.length == 0) {

console.log(err)

res.send({data: 0})

}else {

res.send({data: rows})

console.log(rows)

}

})

})

// 删除用户

router.post('/del\_admin',function(req,res) {

let params = req.body;

conn.query('delete from adminsql where aid="' + params.aid + '"', function(err, rows) {

if(err) {

console.log(err)

res.send({code: 0})

} else {

res.send({code: 1});

}

})

})

// 可以完成添加用户

router.post('/add\_user', function(req, res) {

let params = req.body;

conn.query('select aname from adminsql where aname="' + params.username + '"', function(err, rows) {

if(err || rows.length == 0) {

console.log(err);

conn.query('insert into adminsql set ?', {aname: params.username,asex:params.sex, abirth:params.birth,apwd:params.pwd}, function(err, rows) {

if(err) {

console.log(err);

res.send({code: 0, ms: '服务器出错'})

} else {

res.send({code: 1})

}

})

} else {

res.send({code: 0, ms: '账户已存在'})

}

})

})

// 可以完成删除

router.post('/delete\_user', function(req, res) {

let params = req.body;

conn.query('select username from usersql where username="' + params.username + '"', function(err, rows) {

if(err || rows.length == 0) {

res.send({code: 0, ms: '账户不存在'})

console.log(err);

} else {

conn.query('delete from usersql where username="' + params.username + '"', function(err, rows) {

if(err) {

console.log(err)

res.send({code: 0})

} else {

res.send({code: 1});

}

})

}

})

})

// 可以完成修改（写法有点复杂）

router.post('/update\_user', function(req, res) {

let params = req.body;

conn.query('update usersql set username = "'+params.username + '" ,password = "' + params.password + '" where username = "' + params.username + '"', function(err, result) {

!err ? res.send({code: 1}) : res.send({code: 0})

})

})

// 查询商品信息

router.get('/sel\_goods', function (req, res) {

let params = req.body;

conn.query('select \* from goodssql', function (err, rows) {

console.log(rows)

if (err || rows.length == 0) {

console.log(err)

res.send({data: 0})

}else {

res.send({data: rows})

console.log(rows)

}

})

})

// 删除商品信息

router.post('/del\_goods',function(req,res) {

let params = req.body;

conn.query('delete from goodssql where gid="' + params.gid + '"', function(err, rows) {

if(err) {

console.log(err)

res.send({code: 0})

} else {

res.send({code: 1});

}

})

})

// 修改商品信息

router.post('/upd\_goods',function(req, res) {

let params = req.body;

conn.query('update goodssql set gname = "'+params.name + '" ,gdescribe = "' + params.describe + '",gprice = "' + params.price + '" ,glocation = "' + params.location + '"where gid = "' + params.gid + '"', function(err, result) {

!err ? res.send({code: 1}) : res.send({code: 0})

})

})

// 按条件查询商品

router.post('/query\_goods', function(req, res) {

let params = req.body;

conn.query('select \* from goodssql where gname like "%' + params.name + '%"', function (err, rows) {

if (err || rows.length == 0) {

console.log(err)

res.send({data: 0})

} else {

res.send({data: rows})

console.log(rows)

}

})

})

router.post('/query\_disgoods', function(req, res) {

let params = req.body;

conn.query('select \* from goodssql where gname like "%' + params.name + '%" and gdiscount >= "' + params.discount + '"', function (err, rows) {

if (err || rows.length == 0) {

console.log(err)

res.send({data: 0})

} else {

res.send({data: rows})

console.log(rows)

}

})

})

// 添加商品

router.post('/add\_goods', function(req,res) {

let params = req.body;

conn.query('insert into goodssql set ?', {gname: params.name,gdiscount:params.discount,gprice:params.price,gsale:params.sale,gdate:params.date,glocation:params.location,gperson:params.location}, function(err, rows) {

if(err) {

console.log(err);

res.send({code: 0, ms: '服务器出错'})

} else {

res.send({code: 1})

}

})

})

// 查询商场

router.get('/query\_market', function(req, res) {

let params = req.body;

conn.query('select \* from marketsql', function(err, rows) {

if (err || rows.length == 0) {

console.log(err)

res.send({data: 0})

}else {

res.send({data: rows})

}

})

})

// 按楼层查询商品

router.post('/query\_floor', function(req, res) {

let params = req.body;

conn.query('select \* from goodssql where glocation like "%' + params.location + '%"', function (err, rows) {

if (err || rows.length == 0) {

console.log(err)

res.send({data: 0})

} else {

res.send({data: rows})

console.log(rows)

}

})

})

// 按条件查询商品

router.post('/query\_locg', function(req, res) {

let params = req.body;

conn.query('select \* from goodssql where gname like "%' + params.name + '%" and glocation like "%' + params.location + '%"', function (err, rows) {

if (err || rows.length == 0) {

console.log(err)

res.send({data: 0})

} else {

res.send({data: rows})

console.log(rows)

}

})

})

// 查询商品信息

router.get('/query\_allactive', function (req, res) {

let params = req.body;

conn.query('select \* from merchantsql', function (err, rows) {

console.log(rows)

if (err || rows.length == 0) {

console.log(err)

res.send({data: 0})

}else {

res.send({data: rows})

console.log(rows)

}

})

})

// 查看推广活动

router.post('/query\_avtive', function(req,res) {

let params = req.body;

conn.query('select \* from merchantsql where cname = "' + params.cname + '"', function (err, result) {

if(err) {

console.log(err);

res.send({code: 0, ms: '服务器出错'})

} else {

res.json(result)

}

})

})

// 查询商家

router.get('/sel\_merchant', function (req, res) {

let params = req.body;

conn.query('select \* from merchantsql', function (err, rows) {

console.log(rows)

if (err || rows.length == 0) {

console.log(err)

res.send({data: 0})

}else {

res.send({data: rows})

console.log(rows)

}

})

})

// 删除商家信息

router.post('/del\_merchant',function(req,res) {

let params = req.body;

conn.query('delete from merchantsql where cid="' + params.cid + '"', function(err, rows) {

if(err) {

console.log(err)

res.send({code: 0})

} else {

res.send({code: 1});

}

})

})

// 添加商家

router.post('/add\_merchant', function(req,res) {

let params = req.body;

conn.query('insert into merchantsql set ?', {cname: params.name, clocation:params.location, cimportant:params.important, cactive:params.active, cvideo:params.video, cactive1:params.active1}, function(err, rows) {

if(err) {

console.log(err);

res.send({code: 0, ms: '服务器出错'})

} else {

res.send({code: 1})

}

})

})

// 查询打折商品

router.post('/query\_discount', function(req, res) {

let params = req.body;

conn.query('select \* from goodssql where gdiscount = "' + params.discount + '"', function (err, rows) {

if (err || rows.length == 0) {

console.log(err)

res.send({data: 0})

} else {

res.send({data: rows})

console.log(rows)

}

})

})

// 修改品牌商铺信息

router.post('/upd\_merchant',function(req, res) {

let params = req.body;

conn.query('update merchantsql set cname = "'+params.name + '" ,cactive = "' + params.active + '",cvideo = "' + params.video + '" ,cactive1 = "' + params.active1 + '"where cid = "' + params.cid + '"', function(err, result) {

!err ? res.send({code: 1}) : res.send({code: 0})

})

})

// 查询品牌商铺信息

router.post('/query\_merchant', function(req, res) {

let params = req.body;

conn.query('select \* from merchantsql where cname like "%' + params.name + '%"', function (err, rows) {

if (err || rows.length == 0) {

console.log(err)

res.send({data: 0})

} else {

res.send({data: rows})

console.log(rows)

}

})

})

router.get('\*', (req, res) => {

res.send('404');

})

module.exports = router;