**大连理工大学本科毕业设计（论文）**

**基于Vue实现的外卖商家子系统设计与实现**

**Design and implementation of takeout business subsystem based on Vue.js**

学 院（系）： 软件学院

专 业： 软件工程

学 生 姓 名： 杨天杰

学 号： 201393027

指 导 教 师： 张政凯

评 阅 教 师： 于成

完 成 日 期： 2017年6月7日

大连理工大学

Dalian University of Technology

# 摘 要

当前社会互联网发达，随着高新技术的不停更替，带给我们的不仅仅是娱乐方式的多样化，更多的是生活的便利化。交流从书信转变为了线上即时通信；打车出行从路边招手变成了网约车；在家里不想做饭时，也不用再跑出去找餐厅了，只需要动动手指，美味就能送上门。

这篇文章将实现一个外卖系统的商家子系统，过去的电话订餐除非吃过否则并不能了解到某指定商家的菜品是否理想。现在只需要商家把想要上架的商品拍好照片，写好描述，然后通过此系统上架，用户就能通过此外卖app访问到商家页面，查看到商家的商品照片及描述等等，然后再实现在线下单。

此系统用主要以前端的MVVM框架Vue.js，以及node.js的各种相关包还有mongodb等技术进行项目的设计与构建。该子系统主要分为两大模块：移动端的用户系统，以及管理员用于数据管理的PC端系统。移动客户端主要用来渲染商家信息，以用于用户的便利浏览以及点餐。数据管理系统用于管理人员管理商家信息，查看不同商家信息，修改商家信息，上新产品，添加优惠活动以及新增商家等。

本系统主要为外卖app的商家子系统，对应点击商家列表后进入的商家商品列表，评价等，且不具备下单功能，满足用户浏览各个商家的具体商品等信息，以及管理员管理商家及商品信息。

**关键词：**外卖；Vue；Node.js；Mongodb；Mongoose；在线点餐

**Design and implementation of takeout business subsystem based on Vue.js**

# Abstract

The current social Internet developed, with the high-tech non-stop, to bring us not only the diversification of entertainment, more is the convenience of life. Communication from the letter into a real-time communication; taxi journey from the roadside waving into a network car; do not want to cook at home, do not have to run out to find a restaurant, and only need to move your fingers, delicious will be able to send the door.

This article will implement a takeaway system of business subsystems, past phone reservations, unless you eat it, and can not understand a designated business dishes is ideal. Now only need to businesses want to shelves the goods take a good photo, write a description, and then through this system shelves, users can use this take-away app visit the business page, view the merchant's merchandise photos and description, etc., and then achieve Online orders.

This system uses the main front MVVM framework Vue.js, as well as node.js various related packages and mongodb and other technology to design and build the project. The subsystem is divided into two modules: the mobile user system, and the administrator for data management PC-side system. Mobile client is mainly used to render business information for users to facilitate browsing and ordering. The data management system is used by managers to manage business information, view different business information, modify business information, add new products, add offers, and add new sellers.

Finally, the system is mainly for the takeout app business subsystem, corresponding to click on the list of businesses to enter the merchandise list, evaluation, etc., and do not have the order function to meet the user to browse the specific merchandise and other information, and the administrator manage business and goods information.

**Key Words：**takeout; Vue; Node.js; Mongodb; Mongoose;Online order

目 录

[摘 要 I](#_Toc484090487)

[Abstract II](#_Toc484090488)

[引 言 1](#_Toc484090489)

[1 背景及技术介绍 3](#_Toc484090490)

[1.1 项目背景及意义 3](#_Toc484090491)

[1.1.1 项目背景 3](#_Toc484090492)

[1.1.2 项目意义 3](#_Toc484090493)

[1.2 相关技术 4](#_Toc484090494)

[1.2.1 Vue.js框架 4](#_Toc484090495)

[1.2.2 node.js以及express 4](#_Toc484090496)

[1.2.3 Mongodb数据库 5](#_Toc484090497)

[1.2.4 Webpack 5](#_Toc484090498)

[1.3 本章小结 6](#_Toc484090499)

[2 系统可行性与需求分析 7](#_Toc484090500)

[2.1 可行性分析 7](#_Toc484090501)

[2.1.1 经济可行性 7](#_Toc484090502)

[2.1.2 开发环境可行性 7](#_Toc484090503)

[2.1.3 管理可行性 7](#_Toc484090504)

[2.1.4 技术可行性 7](#_Toc484090505)

[2.2 需求分析 7](#_Toc484090506)

[2.2.1 功能性需求分析 8](#_Toc484090507)

[2.2.2 非功能性需求 16](#_Toc484090508)

[3 系统设计 17](#_Toc484090509)

[3.1 系统整体架构设计 17](#_Toc484090510)

[3.2 系统功能模块设计 18](#_Toc484090511)

[3.2.1 用户点单模块 18](#_Toc484090512)

[3.2.2 外卖商家管理模块 19](#_Toc484090513)

[3.3 数据库设计 20](#_Toc484090514)

[3.3.1 数据库需求分析及E-R图设计 20](#_Toc484090515)

[3.3.2 数据库表设计 23](#_Toc484090516)

[3.4 前端界面设计 24](#_Toc484090517)

[3.4.1 用户点单模块前端设计 24](#_Toc484090518)

[3.4.2 外卖商家管理模块 25](#_Toc484090519)

[4 系统实现 27](#_Toc484090520)

[4.1 系统实现说明 27](#_Toc484090521)

[4.2 用户点单模块 27](#_Toc484090522)

[4.2.1 分组菜单联动功能 27](#_Toc484090523)

[4.2.2 收藏商家功能 30](#_Toc484090524)

[4.2.3 评论筛选功能 32](#_Toc484090525)

[4.2.4 添加至购物车小球飞过效果 34](#_Toc484090526)

[4.3 外卖商家管理模块 38](#_Toc484090527)

[4.3.1 商家搜索功能 38](#_Toc484090528)

[4.3.2 商品分组管理功能 40](#_Toc484090529)

[4.3.3 头像组件的封装 43](#_Toc484090530)

[5 系统测试 44](#_Toc484090531)

[5.1 功能测试 44](#_Toc484090532)

[5.2 健壮性测试 44](#_Toc484090533)

[5.3 单元测试 45](#_Toc484090534)

[5.4 测试总结 45](#_Toc484090535)

[结 论 46](#_Toc484090536)

[参 考 文 献 47](#_Toc484090537)

[致 谢 48](#_Toc484090538)

# 引 言

过去的餐饮模式无非两种，到店餐饮、电话外卖。到店餐饮的主要问题在于时间花费，必须花时间收拾出门，如果家离餐厅还有一段距离的话，还会做公交打车等等，另一个问题则在于如果餐厅比较火爆那么不得不排队等待；电话外卖的主要问题在于信息共享不对称。不能浏览餐厅商品的样式及价格，很多点餐都抱着尝试的心理，结果餐品无论色香味都没达到自己的期望，或者只能重复点吃过的那家店那个菜。而今互联网发展迅速，外卖app正是致力于解决这些问题，不但解决了客户花费时间的问题，还提高了餐厅的工作效率，商户依次接单解决了排队的苦恼，当你吃到好吃的菜品时，还可以一键分享给身边朋友，既方便了商家又方便了客户。

基于Vue实现的外卖商家子系统主要解决了商家与客户信息共享不对称的问题，商家只要有商品的更新，商品的促销就可以提供信息给管理员，管理员通过操作数据库实时更新，用户就可以在任何时间点所处任何地方查看商家的商品信息。该系统的主要角色为管理员与用户，管理员与用户分别对应不同web界面。

角色的主要职责；

1. 系统管理员：负责接收商家提供的信息，实现商家信息的增删改查。
2. 用户：移动端访问商家界面，可以查看到商家的商品信息及促销等。

商家子系统网站采用前后端完全分离的开发方式。所谓的前后端分离[1]，后台只需要提供API接口，前端调用AJAX实现数据的呈现。组件化以及工程化不需要再依赖后端去实现。前端利用Vue实现组件化，数据驱动，后端的业务逻辑层使用node.js去操作mongodb，再使用node.js提供api接口访问到数据库的数据，前端就可以通过AJAX调用到这些数据更新页面的呈现。

这样开发的主要优点在于：

1. 彻底解放前端：前端不需要再向后台提供模版或是在前端html中嵌入后抬代码。分离之后提高可读性，解耦。
2. 提高工作效率，分工更加明确：前端只用关注前端的事，后台也只用提供api给前端。
3. 局部性能提升：按需加载
4. 维护成本降低：MVVM框架，更替数据改变页面。

后台主要采用的是node.js语言进行开发，选用了express基于Node.js平台的web应用开发框架，它提供了丰富的HTTP快捷的方法和Connect中间件，包装了HTTP请求，快速搭建起了一个拥有极本web功能的应用，再由此不断扩展api，满足系统需要的基本功能。

在数据库的选用上，选取了非关系型数据库mongodb。先采用了关系型数据库表的设计模式，使用了PowerDesigner进行了表基础的ER图的设计，然后将设计好的关系型数据库表转化为非关系型数据库集合。

在前端上主要采用了Vue进行了组建式开发，将页面拆分为了几个组件，再通过组建通过封装好的异步AJAX请求后台数据，父组件再将数据传递给子组件，从而渲染整个网页系统，达到高内聚低耦合的结构，易维护可扩展。

# 1 背景及技术介绍

## 1.1 项目背景及意义

### 1.1.1 项目背景

在过往的非自己做饭的情况下，大家吃饭的方式无非是下馆子，或者电话订餐。并非重大意义聚餐，只是单纯的为了解决温饱，但却要出门走到餐厅吃饭，这是一件很浪费时间的事情。后者电话订餐解决了这个时间浪费的问题，但是又出现了新的问题就是，信息共享不对称，客人只会记住固定几家店的电话，而这固定几家店的商品也只能了解部分而已，每次都是重复的点餐，或许又是失望的尝试。随着时代的进步，由于人类智慧的结晶­——互联网的出现，外卖app被创造了出来，它既解决了时间浪费的问题，又解决了商家商品信息获取的问题。

外卖app给我们带来的好处还远不止解决时间，信息共享等问题。对于商家，提高了商家的知名度，对于一些开店很久的商家，可能因为位置偏远，招牌不醒目而被忽略，实际上商品的质量却很高，却不为人所知，现在大家只需要地区定位搜索，就能搜索到周围所有的商家，列表呈现；提高了商家的工作效率，平时商家只有在饭点，大学生下课，上班族下班时，才有客人来，从而忙碌起来，而现在通过大家网上订餐，时间范围扩展，商家有更充足的时间准备商品，即提高了质量又提高了数量，利润达到了倍级地提升。对于用户，解决了时间成本，我可以花这些时间去干更多自己喜欢的事情了，学习或者娱乐[2]；得到优惠，当商家利益增大的时候，就会放出很多优惠以吸引更多的顾客，而这些优惠的受益者无非就是用户。对于社会，提高了互联网的普及率，很多商家并不会什么手机，电脑，更不愿意把钱存在网上，对网络及其地不信任，通过使用外卖app，让他们得到了外卖app所给他们带来的利益时，他们会更情愿了解互联网，使用互联网；提高了社会就业率，当外卖app被大面积普及时，必定会出现一个新职业，那就是送餐员，送餐员会有大量需求，更多人群将会获得就业机会。

### 1.1.2 项目意义

外卖app商家模块子系统，主要解决了商家用户信息共享不对称的问题，商家有更新，用户不能得到及时的通知。而这个项目正是解决了这个问题，商家可以提交信息给管理员，管理员可以通过信息操作外卖商家信息管理系统更新数据库商家数据。当数据库被更新了以后，用户访问商家移动端系统时，就能获取到商家一手的更新信息，从而下单选到自己心意的美食商品。

丰富多元的商家给用户提供了更多的就餐选择，在不同的商家不选择不同的商品。就餐体验也是不相同的。相比传统的餐饮方式，在线点餐更容易选择到自己喜欢的商品，商家也更容易通过用户点餐的大数据记录用户的商品需求方向，以制定出更美味更廉价的商品，达到商户用户双赢。

## 1.2 相关技术

基于Vue实现的外卖商家子系统主要运用了Vue以及Vue相关插件包括vuex，vue-resource，vue-router来搭建前端页面，主要运用的组件式开发方式，把整个移动端系统构建成了一个单页应用；运用了node.js编写后台，主要为express包搭建web服务，mongoose包操作数据库；运用了mongodb数据库进行数据存储；运用了vue-cli 脚手架 + webpack使开发流程化，代码分包等。

### 1.2.1 Vue.js框架

Vue.js是一个用来开发网页界面的前端库，由js封装而成，主要采用MVVM的思想，它有配套的周边工具，例如vuex，vue-resource，vue-router等，算上这些东西，我们可以把Vue叫做前端框架。它的主要特点为：

1. 响应式编程：声明一个js对象，然后在模版里用花括号引用，界面就能显示出这个js对象所对应属性的值，并且是双向绑定的。
2. 组件化：在组织复杂界面的问题上，页面上的一切都是组件。
3. 模块化：可以把每一个Vue组件都看作是一个独立的模块，以.vue为后缀来编写这个软件，最后再使用webpack等打包就能变成一段可执行的html，css，js代码了。
4. 动画：Vue自带简洁的动画的过渡系统，很多获奖的互动类网站都是用Vue开发的。
5. 路由：vue-router插件就是负责组织路由的，可以将嵌套的路径映射到嵌套的组件里，并且提供了细致的路径跳转控制。
6. 插件：vue-router就是一个控制路由的插件；vuex，是一个专门为vue构建的应用程序设计的状态管理模式。它采用了集中式的方法存储管理应用组件的状态，并且以一定的规则保证状态以一种可以预测的方式发生变化；vue-resource，也是专门为vue所构建的应用提供web请求和响应操作的工具，是AJAX的一层封装，支持Promise，URI Templates，兼容火狐，chrome，safari，opera 和IE9以上。

### 1.2.2 node.js以及express

谁都没有想过JavaScript可以用来写服务端，而通过Node.js它可以了。Node本身运行V8 JavaScript。而V8 JavaScript引擎是谷歌公司开发用于Chrome浏览器的底层JavaScript引擎[3]。谷歌使用了V8创造了一个用C++编写而成的超快解释器，并且这个解释器还有一个特别之处，那就是你可以下载这个引擎，并且将其嵌入任何的应用程序之中。所以，Node.js就是利用这个引擎，将它重建为可以在服务器上使用。Node.js的主要特点为：

1. 异步I/O：不需要等待服务器的响应，继续执行之后的动作，等到服务器响应了之后，再做响应后需要执行的动作。
2. 事件循环：当收到web server的请求，然后立即关闭进行处理，然后去服务下一个web server请求。当这个请求完成时，再放回处理队列，当队列达到队头，则放回给用户。与传统的相比：传统的处理好比银行柜台，人坐在柜台前，一个窗口只能服务一个人，其他人只有等待，这个人等待服务完成后才离开，而柜台人员也才能继续服务其他人。而node.js则好比麦当劳，每个人只需要点餐，点餐完成后去旁边等待，其他人继续点餐，然后厨房做完了餐点后送到旁边，排队的人依次领取。
3. 单线程：没有死锁存在，也没有线程切换所带来的性能上的开销。
4. npm包：由node.js官方提供的第三方包管理工具，可以下载使用第三方由node.js编写的包。例如express，他是一个node.js关于创建web应用的框架，提供了丰富的方法来帮你创建应用。特点主要是，可以用中间件来响应HTTP  
   的请求，定义不同的路由表来表示不同的HTTP动作，还可以通过模版渲染HTML的页面。

### 1.2.3 Mongodb数据库

Mongdb是使用C++进行编写的数据库，是一个基于分布式存储的非关系型数据库。Mongodb主要为了解决web应用可扩展高性能的数据库解决方案。Mongodb将数据以文档的方式存储，数据结构主要是键值对，类似于JSON对象[4]。而Mongodb的操作主要是依靠node.js关于Mongodb的包Mongoose。

Mongoose是一个node.js操作Mongodb的包，主要提供了简易的api解决了创建表的数据约束问题，以及原生操作db所带来异步的不便。

### 1.2.4 Webpack

Webpack也是通过npm下载到的一个包，但是他的主要作用是创建工作流，快速创建我们所需要的工作场景，也可以说是环境，且能快速打包。它的主要思想是一切皆模块，正如上面我们所用的vue-resource，vue-router他们都可以看作是一个模块，或者说bootstrap等css文件也可以说是模块，当这些模块使用得繁多的时候，那么维护上线都是一个困难的问题，webpack都可以帮你解决。它主要分为开发模式和生产模式。主要提供：代码分包，静态资源整合打，异步加载等功能，主要特点有：

1. 模块化：可以把系统复杂的程序化为细小的文件。
2. 编译预处理：有人喜欢使用TypeScript等编写js，sass等编写css，但是这些东西并不能被浏览器所解析，那么当我们编写这些东西的时候，按下保存，webpack可以帮你预编译，然后可以在浏览器中等直接浏览效果。

## 1.3 本章小结

本章主要还是介绍了传统的餐饮习惯给我们带来的不便之处，而外卖app又给大家带来了哪些改变。还主要介绍了本项目中主要使用的几项技术包括vue，node.js，mongodb还有webpack和相关周边的一些技术，以及这些技术的主要作用，特点，还有给我们所带来的编程上的优点。

# 2 系统可行性与需求分析

## 2.1 可行性分析

可行性分析是分析开发出来的软件是否值得开发、然后计算软件成本、估计软件未来可带来的收入、并且与用户群体分析讨论然后征求意见、最终总汇而成的可行性研究的报告。简言之，就是对软件开发完后所得到的利益进行评估，以判断软件是否值得开发，可行性分析必须具有可预见性、公平公正、有据可循的特点[5]。该项目的可行性主要包括了以下几点：

### 2.1.1 经济可行性

经济可行性主要是判断可以使用的资源的可能性包括人力资源、自然拥有资源、资金。主要采用的方法是成本——效益分析发。这个项目完成后，随着订单量的提升，商家的利润就增大，增加过后给出优惠活动，用户将会继续增多，从而形成了一个正向循环，在适宜的时候从中抽取单比费用，从而达到盈利；在人力上可以通过人才市场寻找大批待职人员参加到送餐员的行列中。由此可见，在经济上是完全可行的。

### 2.1.2 开发环境可行性

开发环境可行性主要指开发人员意见是否一直，资金链是否稳定，是否能抽出骨干来带领开发，简单地说就是新项目的开发及维护是否能够得到一个长期的良好环境。本项目属于重点项目，由核心部门人员带领开发，大家技术性选型都很同意，平台很大，开发环境完全可行。

### 2.1.3 管理可行性

本项目主要有两大模块，其中一大模块就是外卖商家信息管理系统，本管理系统只有在内网能够访问，可支持多点同时修改数据，由卖家提供信息，管理员进行数据更改，在管理上完全可行。

### 2.1.4 技术可行性

技术可行性主要考量的是当代技术是否能够实现该系统的所有需求的功能以及是否能满足用户的需求。本项目主要采用Vue.js来构建前端界面，该框架已经相当成熟并且已经有了很多配套的插件，页面上的交互完全能够满足需求。数据库是开源的mongodb，可以满足基本存储，并且不收取任何费用，这也极大地节约了成本，各种便利的包也都是开源的，轻易地就能满足需求，技术上也完全可行。

## 2.2 需求分析

需求分析是指对要解决的问题进行的详细分析，弄清楚问题到底需要解决什么，包括需要什么数据，呈现出来的是什么样的一个状态。以“软件工程”中的术语来说的话，就是要弄清楚所设计的这个软件到底做什么，想要达到一个什么样的效果，可以说需求分析是设计项目前必须做的一项工作，主要为功能性需求分析。

### 2.2.1 功能性需求分析

功能性需求指的是项目中具体需要给用户提供的功能，而本系统可以分为两个模块，外卖商家信息管理系统和用户点单页面，不同模块的需求分析是分开的。

不同模块用户划分如图2.1：

点餐用户

主要用户划分

外卖商家信息管理员

图2.1 系统角色划分

图2.1给出了两个模块的不同角色，接下来将从不同角色所对应的模块，来做具体的需求分析，两个模块的完全分离完全解耦，有利于后期维护。

（一）点餐模块

本系统主要采用的是组件式开发模式[6]，不同的组件对应着客户不同的具体行为操作，故本节主要叙述的是点餐用户界面的六个组件的需求设计，主要包括：头部组件，根组件，商品组件，评价组件，商家组件，购物车组件。

1. 头部组件：

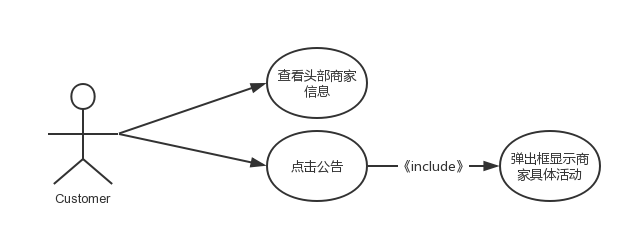


图2.2 头部组件用户用例图

头部组件为固定组件，固定组件指的随着路由的变化，依然保持固定位置的组件。头部组件的主要作用在于给用户提供基础信息，例如商家的名字，主要参与的促销活动，送达时间等等，主要的交互功能为用户点击公告栏，会弹出浮层，显示商家更为详细的信息，包括商家的总得分，参与的所有促销活动等等，然后点击关闭按钮浮层关闭。

表2.2 头部组件用户用例图描述

|  |  |
| --- | --- |
| 项目 | 内容 |
| 用例名称 | 用户 |
| 用例描述 | 用户对头部组件的操作 |
| 参与者 | 用户 |
| 前置条件 | 无 |
| 后置条件 | 如果该用例成功，用户可以查看商家详细信息 |
| 基本操作 | 1. 用户查看头部商家信息 2. 用户点击公告查看详细信息 3. 用户关闭弹窗 |

1. 根组件

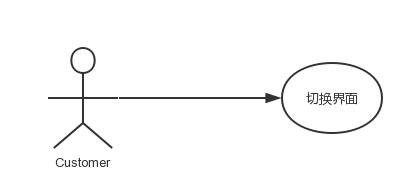


图2.3 根组件用户用例图

根主组件指的是其他组件所挂载的主要父组件，然后这个根组件又实际挂载在一个具体的HTML元素上，实现真正的组件化开发。他对用户的主要功能在于提供了界面的切换，点击了菜单栏以后路由会发生改变，界面将在商品组件，评论组件，商家组件间发生变化。

表2.3 根组件用户用例图描述

|  |  |
| --- | --- |
| 项目 | 内容 |
| 用例名称 | 用户 |
| 用例描述 | 用户对根组件的操作 |
| 参与者 | 用户 |
| 前置条件 | 无 |
| 后置条件 | 如果该用例成功，用户可以切换不同组件 |
| 基本操作 | 1. 切换界面 |

1. 商品组件

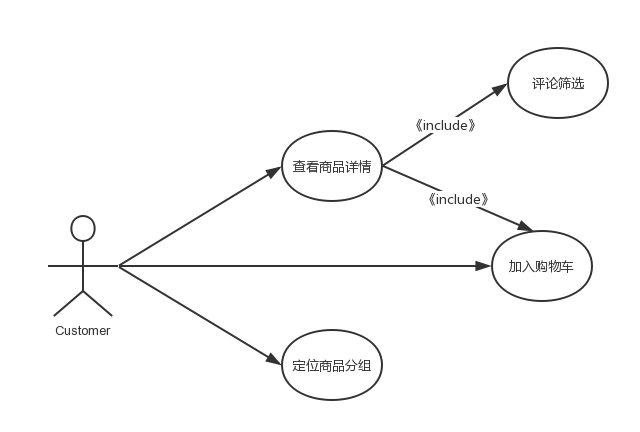


图2.4 商品组件用户用例图

商品组件主要展示商家所有商品，以及每个商品的所在分组，同时可以根据分组直接查看该分组的信息，然后用户可以对商品进行加入购物车操作，点击进入商品详情之后可以查看商品的详细信息，并且也可以进行加入购物车操作，还可以看见商品评价，以及对商品实现一个筛选。

表2.4 商品组件用户用例图描述

|  |  |
| --- | --- |
| 项目 | 内容 |
| 用例名称 | 用户 |
| 用例描述 | 用户对商品组件的操作 |
| 参与者 | 用户 |
| 前置条件 | 用户切换到商品界面 |
| 后置条件 | 如果该用例成功，用户可以进行下单操作 |
| 基本操作 | 1. 点选分组栏，直接查看分组商品 2. 点击商品查看商品详情 3. 将商品加入购物车 |

1. 评论组件

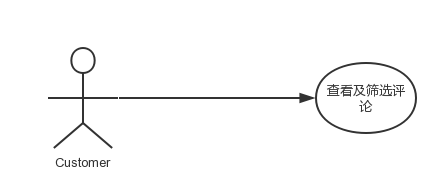


图2.5 评论组件用户用例图

评论组件主要展示商家所有获得的评论，然后用户可以从赞踩全部这三个选项来选择查看某种评价。

表2.5 评论组件用户用例图描述

|  |  |
| --- | --- |
| 项目 | 内容 |
| 用例名称 | 用户 |
| 用例描述 | 用户对评价组件的操作 |
| 参与者 | 用户 |
| 前置条件 | 用户切换到评价界面 |
| 后置条件 | 如果该用例成功，用户可以查看不同种类评价 |
| 基本操作 | 1. 对评价进行筛选查看 |

1. 商家组件

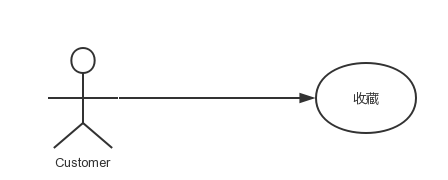


图2.6 商家组件用户用例图

商家组件主要展示商家所有相关信息，包含各项评分，商家店内照片，商家所提供的信息包含地址电话等待，这里给用户提供了一个收藏功能。

表2.6 商家组件用户用例图描述

|  |  |
| --- | --- |
| 项目 | 内容 |
| 用例名称 | 用户 |
| 用例描述 | 用户对商家组件的操作 |
| 参与者 | 用户 |
| 前置条件 | 用户切换到商家界面 |
| 后置条件 | 如果该用例成功，用户可以成功收藏店铺 |
| 基本操作 | 1. 收藏店铺 |

1. 购物车组件

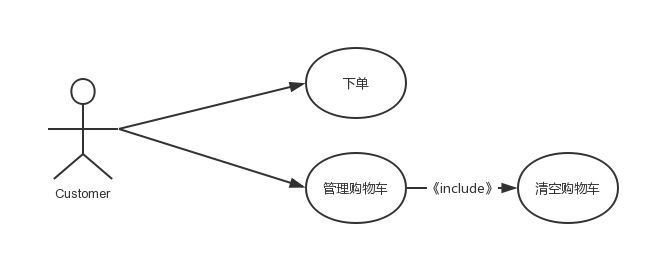


图2.7 购物车组件用户用例图

购物车组件为用户提供选择好商品后的管理以及下单行为。

表2.7 购物车组件用户用例图描述

|  |  |
| --- | --- |
| 项目 | 内容 |
| 用例名称 | 用户 |
| 用例描述 | 用户对购物车组件的操作 |
| 参与者 | 用户 |
| 前置条件 | 用户添加商品到购物车 |
| 后置条件 | 如果该用例成功，用户可以清空购物车或者成功下单 |
| 基本操作 | 1. 下单 2. 管理购物车内商品 |

1. 外卖商家信息管理模块

外卖商家信息管理模块的用户是管理员，为了防止有商家恶意删除更改别家信息，或者上传一些非法信息，该管理系统专为内网管理员设计，只允许管理员对商家的信息进行管理，该模块主要建立在pc端上，同样采用了前端的组件化开发模式。主要包括根组件，商家基本信息组件，商家商品信息组件。

1. 根组件

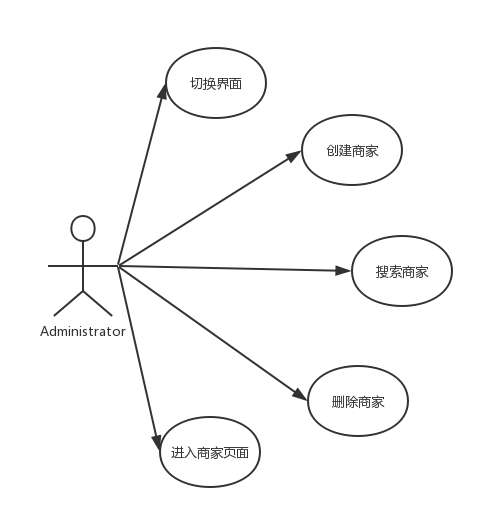


图2.8 根组件管理员用例图

根组件主要提供的是商家级别上的管理，比如有新的商家提交了入驻信息，又比如有什么商家想要关店了想要删除商店的网上信息等等，当商家到达一定数量级别的时候，单单是列表翻页查找具体商家是一件非常麻烦的事情，故需要提供搜索功能搜索商家。除了商家级别上的功能外，还需要提供信息更替后查看商家新的界面的功能。商家的信息又分为了两块，需要提供路由导航在这两块来回切换。

表2.8 根组件管理员用例图描述

|  |  |
| --- | --- |
| 项目 | 内容 |
| 用例名称 | 管理员 |
| 用例描述 | 管理员对根组件的操作 |
| 参与者 | 管理员 |
| 前置条件 | 无 |
| 后置条件 | 如果该用例成功，可完成商家级别上的切换 |
| 基本操作 | 1. 创建商家 2. 搜索商家 3. 进入商家节目 4. 切换界面 |

1. 商家基本信息组件

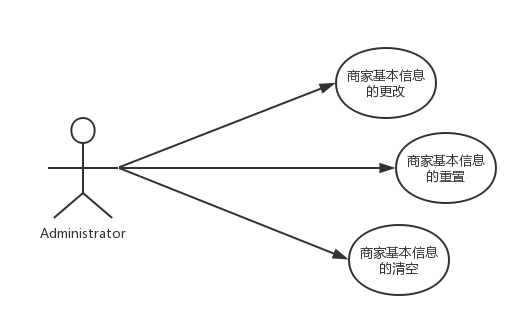


图2.9 商家基本信息组件管理员用例图

商家基本信息的管理主要包括，对基本信息的更改，比如名字，描述，公告，参与的促销活动等。如果修改完了还未保存，突然不想更改了，那么需要提供一个重置功能，则回到之前为修改的信息状态。如果商家想要更换种类，比如从一个粥品店转换到川菜，那么需要提供一个将所有信息清空的功能，以便于管理员重新添加数据。

表2.9 商家基本信息组件管理员用例图描述

|  |  |
| --- | --- |
| 项目 | 内容 |
| 用例名称 | 管理员 |
| 用例描述 | 管理员对根组件的操作 |
| 参与者 | 管理员 |
| 前置条件 | 切换到商家基本信息组件 |
| 后置条件 | 如果该用例成功，可更改商家信息 |
| 基本操作 | 1. 更改信息 2. 重置 3. 清空 |

1. 商家商品信息组件

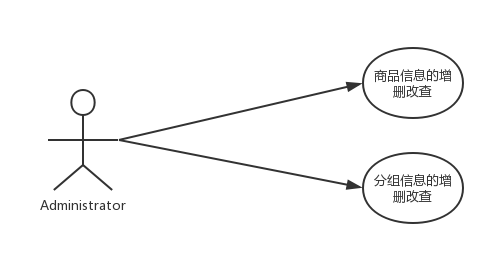


图2.10 商家商品信息组件管理员用例图

商家商品信息除了包括商品以外，还有商品的分组，该组件主要需求是对这两部分的增删改查，除此之外，还要给商品添加分组删除分组等。

表2.10 商家商品信息组件管理员用例图描述

|  |  |
| --- | --- |
| 项目 | 内容 |
| 用例名称 | 管理员 |
| 用例描述 | 管理员对根组件的操作 |
| 参与者 | 管理员 |
| 前置条件 | 切换到商家商品信息组件 |
| 后置条件 | 如果该用例成功，可更改商家商品信息 |
| 基本操作 | 1. 分组的增删改查 2. 商品的怎删改查 |

### 非功能性需求

一个软件产品除了功能性需求还有非功能性需求，就算功能性需求被设计得相当好，但是如果把非功能性的需求忽视了话，往往会导致商品质量很差，很大程度上影响功能性需求上的定义，甚至可能淹没功能性需求所带给用户的体验和价值。我们主要从以下几个方面来定义非功能性需求：

1. 效率：本项目的两个模块主要都采用的是组件化开发，组件化开发有一个极大的好处就在于组件的重用上，例如商家基本信息页面有3个div，商家商品信息需要4个div，那么不会把信息页面的3个div删除掉，而是直接替换内容，再重新添加1个div，这样的话div利用率就会很高，网站的响应速度也会非常快。
2. 维护性：由于采用的组件化开发方式，各个组件之间高内聚、低耦合，各个逻辑分开互不影响，修改了一个组件不会影响到其他组件，易维护。
3. 易用性：组件开发，页面结构清晰，用户很快就能掌握整个app的流程，而且适合各个年龄段的人使用。

# 3 系统设计

## 系统整体架构设计

该项目的整体架构分为了两大模块，pc端页面外卖商家管理模块与用户点单模块。两个模块的基本流程一致，但外卖商家管理模块与后端的交互更多一些、用户点单模块的前端逻辑复杂得多，但是与后台的交互相对简单。这里就以外卖商家管理模块为例来描述整改过程。

进入界面时，由前端界面发送http请求给后台获取数据，后台连接上数据库，接受到请求后从数据库中读取数据并格式化返回给前端界面。同时在前端界面发生增删改查时都会给后台发送请求，后台根据请求修改数据库并且返回给前端界面结果，这时当数据库的数据发生改变时，用户再进入点单系统时，获取到的数据则是更新后的数据，由此达到系统的管理与呈现。

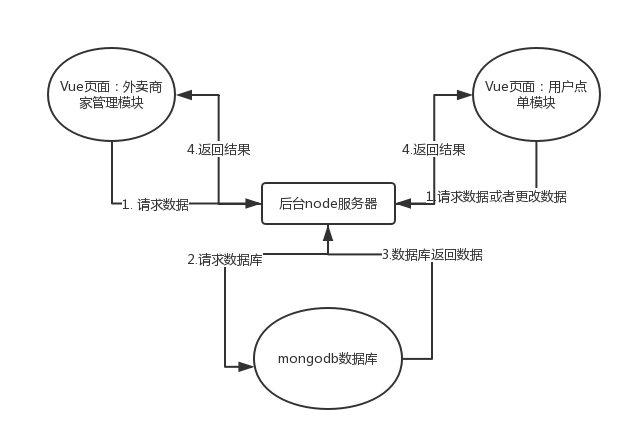


图3.1 外卖商家子系统顶层设计

## 系统功能模块设计

根据之前的需求分析，将各个组件整合为模块之后，系统功能模块主要为外卖商家管理模块和用户点单模块，接下来将详细介绍各个模块的设计：

### 3.2.1 用户点单模块

用户点单模块是用户进入系统后可以使用的所有功能模块，主要包括了购物车操作，查看店铺基础信息，查看商品列表，查看评论列表，查看商家具体信息这五大基础的功能，具体功能设计详见图3.2：

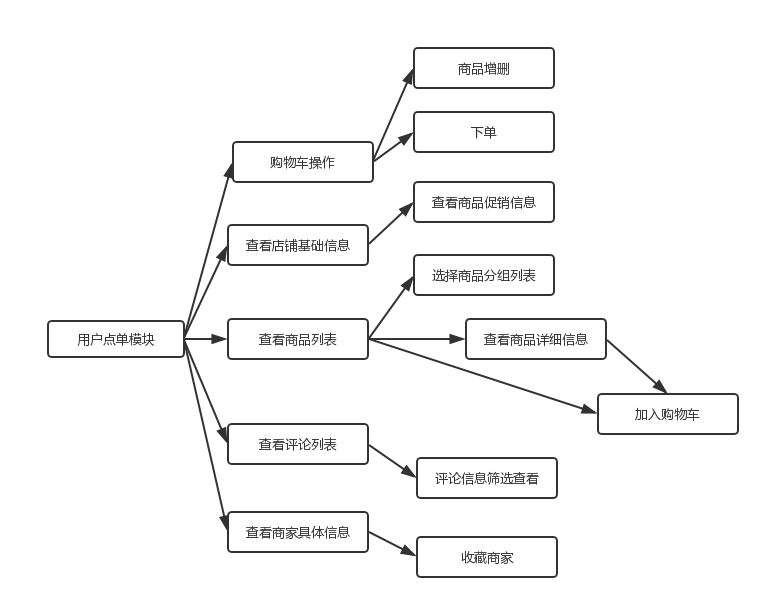


图3.2 用户点单模块功能

用户通过与不同组件的交互来完成各项具体功能。主要功能为：

1. 点击底部购物车时，可以看见自己已选择的所有商品及数量，如果不需要了可以删除，如果需要更多份那么可以继续增添份数，决定好后就可以点击下单；
2. 查看店铺的基础信息在系统头部就能看见，如果想要更多得了解促销信息和公告则需要点击头部；
3. 查看商品列表，分为左栏右栏，左栏为商品的分组，有利于用户可以直接定位到自己感兴趣的商品种类位置，右栏为所有的商品信息按分组排列，可以直接选择添加到购物车中，也可以点击查看商品的详情包括商品的评价等等再判断是否加入购物车；
4. 查看评论列表，默认显示所有评论，但是可以通过按钮筛选出想看的好的评论或者是不好的评论。
5. 查看商家详情，可以获得一些商家的详细信息，如果觉得会长期在这个商家消费的话可以收藏该商家。

### 外卖商家管理模块

外卖商家管理模块是管理员进入了之后能够行使的功能，主要包括查询商家，商家基础信息管理，商家商品信息管理，具体功能设计详见图3.3：

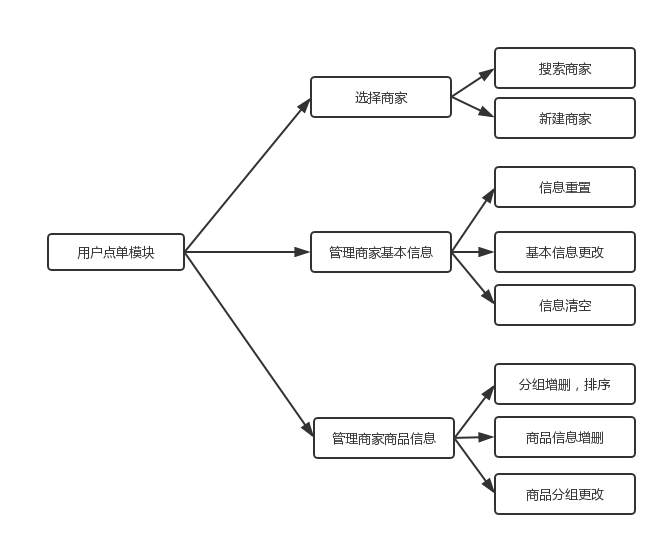


图3.3 外卖商家管理模块功能

管理与不同组件的交互来完成各项具体功能。主要功能为：

1. 在选择商家时，可以按照商家名字包含字符搜索到指定商家，也可以新建商家，那么正文部分将会自动转换到新建商家界面，之前的商家如果没有做更改的话记得保存信息后再新建商家；
2. 管理商家基本信息主要包括，头像图片的更换­、名字等基础信息、商家所参与的促销活动、商家想要提供给用户的基本信息、商家的店内图片上传与删除等等；
3. 管理商家商品信息主要包括，创建分组给分组起名字以及该分组是否参加促销活动，由于商品的总列表是按造分组排序的，故可以替换分组的顺序，创建商品更改已有商品信息包括图片描述等。

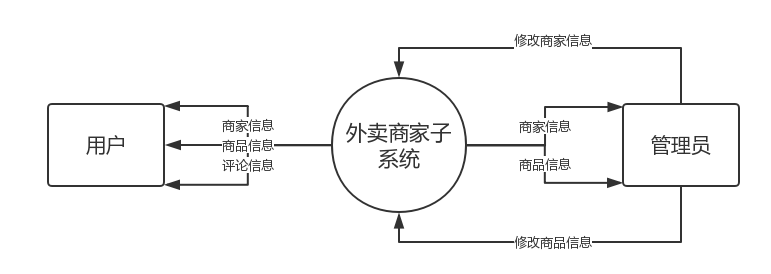
## 数据库设计

数据库的设计是对一个确定的环境，设计出最优秀的数据库模型，通过建立数据库以及相应的系统，让它能够最有效地存储数据以满足用户对于应用的各种需求。由于该项目事务与约束性不强，查询的需求比较小，切数据价值并非重点，故采用的是非关系型数据库mongodb，先通过关系型数据库的设计模式来设计最终将这些关系转化为一个非关系集合。

### 3.3.1 数据库需求分析及E-R图设计

在设计一个性能完善的数据库系统之前，掌握系统环境和系统的要求是至关重要的。所以第一步不应该匆匆忙忙立即开始设计，而是应该收集用户需求、分析数据库要素。故之前的需求分析是收集信息的关键，通过需求分析找出数据要素，分析数据关系，画出数据流向[7]。

由需求分析可以看出，所有的信息都围绕着商家，故商家表是第一要素，而从用户所获取的信息来看，又包含商家的基本信息，商品信息，还有统计而来的评价信息。故我们需要从这几个大点且结合前端设计图来获得更小更细的表单信息，首先我们要明确首要的数据流向，如图3.4：

图3.4 系统顶层数据流图

通过顶层数据流可以确定实体应该有商家，商品，评论，可以由系统的更多需求得出商品又包括两个实体应该为分组实体和商品实体，一个分组可以拥有多个商品，一个商品也可以被多个分组拥有；而评论也包含两个实体，一个是评论的内容，一个是评论的用户，每一条评论只能由一个用户发出。故分析了这些实体关系之后可以画出E-R图3.5：

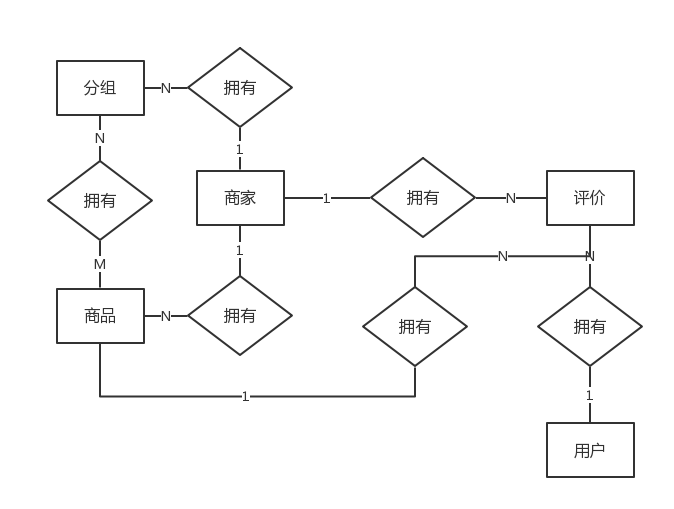


图3.5 系统ER图分析

再了解了这些关系之后，我们需了进一步的了解每个实体所包含哪些属性，由于管理员能操作的商家信息及商品信息都是点单模块返回给用户信息的一部分，故这里只需要重点分析用户与系统的交互数据即可，由于具体数据复杂，将由前端组件层面分析出数据具体属性，如图3.6所示：

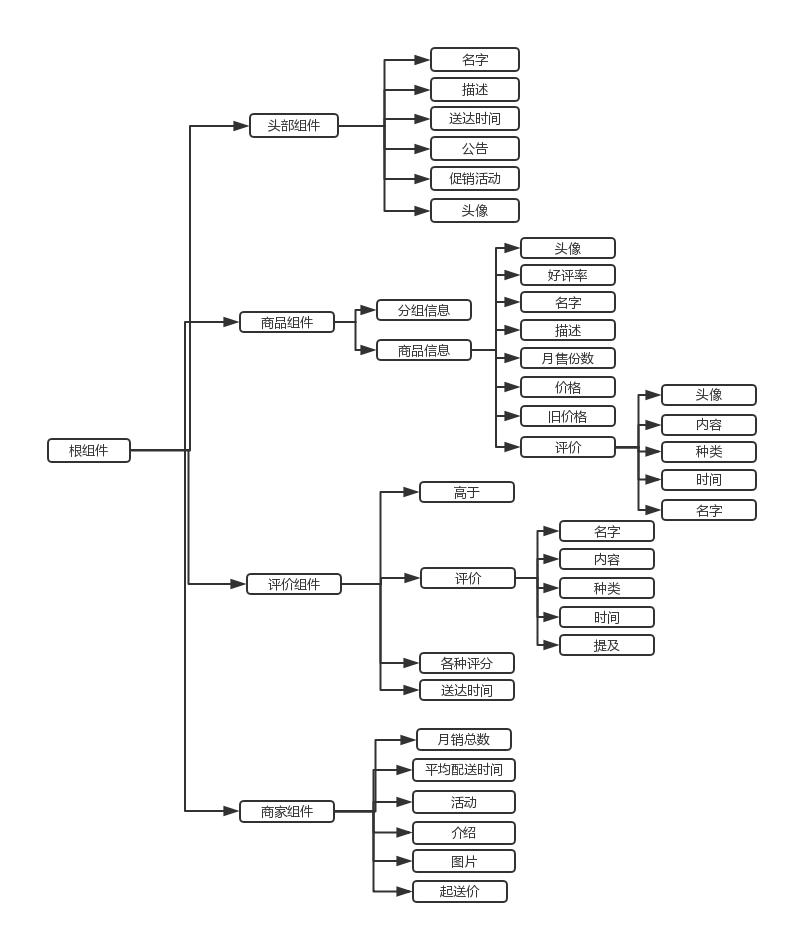


图3.6 组件的数据关系

从顶层数据流出发，研究需求分析，画出E-R图，之后确定好了每个实体所包含的属性，最后就应该来设计表了。

### 数据库表设计

我们已经采用了关系型数据库设计的分析方法得到了该系统所拥有的数据结构，接下来就该设计数据库表了。在mongodb中，并没有表这个概念，而是指的是集合。在传统数据库中的表，例如：表一（id，名字，电话），表二（id，店铺名，总销量），表一与表二是一对多的关系，这样的两个表会在mongodb中设计为这样的集合（id，名字，电话，[店铺名，总销量]），仔细分析可以看出，多的一方成为了一的一方的一个数组，就是因为有这样的转化关系，所以可以采用传统的数据库设计方式，最终再转化为一个集合。从E-R图中可以发现实体关系中大多为多对一的关系，当遇见多对多的关系时，则把数量级小的一方作为实体，数量级多的一方作为属性。故最终的设计表为：

表3.1 商家表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 字段名 | 字段描述 | 数据类型 | 说明 |
| 1 | id | 主键 | ObjectId | 主键，随机 |
| 2 | description | 描述 | String | NULL |
| 3 | deliveryTime | 送达时间 | Number | NOT NULL |
| 4 | score | 得分 | Number | NOT NULL，0-5 |
| 5 | serviceScore | 服务得分 | Number | NOT NULL，0-5 |
| 6 | foodScore | 食品得分 | Number | NOT NULL，0-5 |
| 7  8  9  10  11  12  13  14  15  16  17  18 | rankRate  minPrice  deliveryPrice  ratingCount  sellCount  bulletin  supports  avatar  pics  infos  goods  ratings | 高于  送餐价  起送价  评价数  卖出数  公告  支持种类  头像  照片  信息  货品  评价 | Number  Number  Number  Number  Number  String  Array  String  [String]  [String]  Array  Array | NOT NULL，0-100  NOT NULL，0-100  NOT NULL，0-100  NOT NULL，默认0  NOT NULL，默认0  NOT NULL  support表的数组  默认  地址字符数组  信息字符数组  good表的数组  rating表的数组 |

表3.2 分组表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 字段名 | 字段描述 | 数据类型 | 说明 |
| 1 | id | 主键 | ObjectId | 主键，随机 |
| 2 | name | 名字 | String | NOT NULL |
| 3  4 | type  foods | 促销类型  商品 | Number  Array | NOT NULL，-1-5  food表的数组 |

表3.3 商品表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 字段名 | 字段描述 | 数据类型 | 说明 |
| 1 | id | 主键 | ObjectId | 主键，随机 |
| 2 | name | 名字 | String | NOT NULL |
| 3 | price | 现价 | Number | NOT NULL |
| 4 | oldPrice | 原价 | Number | NOT NULL |
| 5 | description | 描述 | String | NULL |
| 6 | rating | 评价数 | Number | NULL |
| 7  8  9  10 | sellCount  info  ratings  image | 卖出数  商品信息  评价  头像 | Number  String  Array  String | NOT NULL，默认0  NULL  rating表的数组  NOT NULL |

表3.4 评价表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 字段名 | 字段描述 | 数据类型 | 说明 |
| 1 | id | 主键 | ObjectId | 主键，随机 |
| 2 | username | 名字 | String | NOT NULL |
| 3 | rateTime | 评论时间 | Number | NOT NULL |
| 4 | rateType | 评论类型 | Number | NOT NULL，0-2 |
| 5 | text | 评论内容 | String | NULL |
| 6 | avatar | 头像 | String | NOT NULL |

这里主要列举一些重要的表，商品表其实是分组表的一部分，而分组表，评价表正又是商家表的一部分，这样整个数据结构其实只有商家表这个集合，这也正是mongodb数据库的特点。

## 前端界面设计

### 3.4.1 用户点单模块前端设计

用户点单模块的前端界面的设计主要由设计部门给出，然后将页面的各个部分抽象成各个组件，再采用组件化开发。



图3.7 设计稿首页

由图3.7首页可知，当用户进入app时默认进入的是商品界面，点击评价、商家切换其他页面。内容整体采用的是三段式结构，分别为头部、导航、内容，头部与导航固定不动，内容随着点击的导航栏位置而切换界面。在商品页中又是左右两栏布局，并且全局固定了一个购物车，详细的用户界面设计可以参考系统实现中的界面设计。

### 外卖商家管理模块

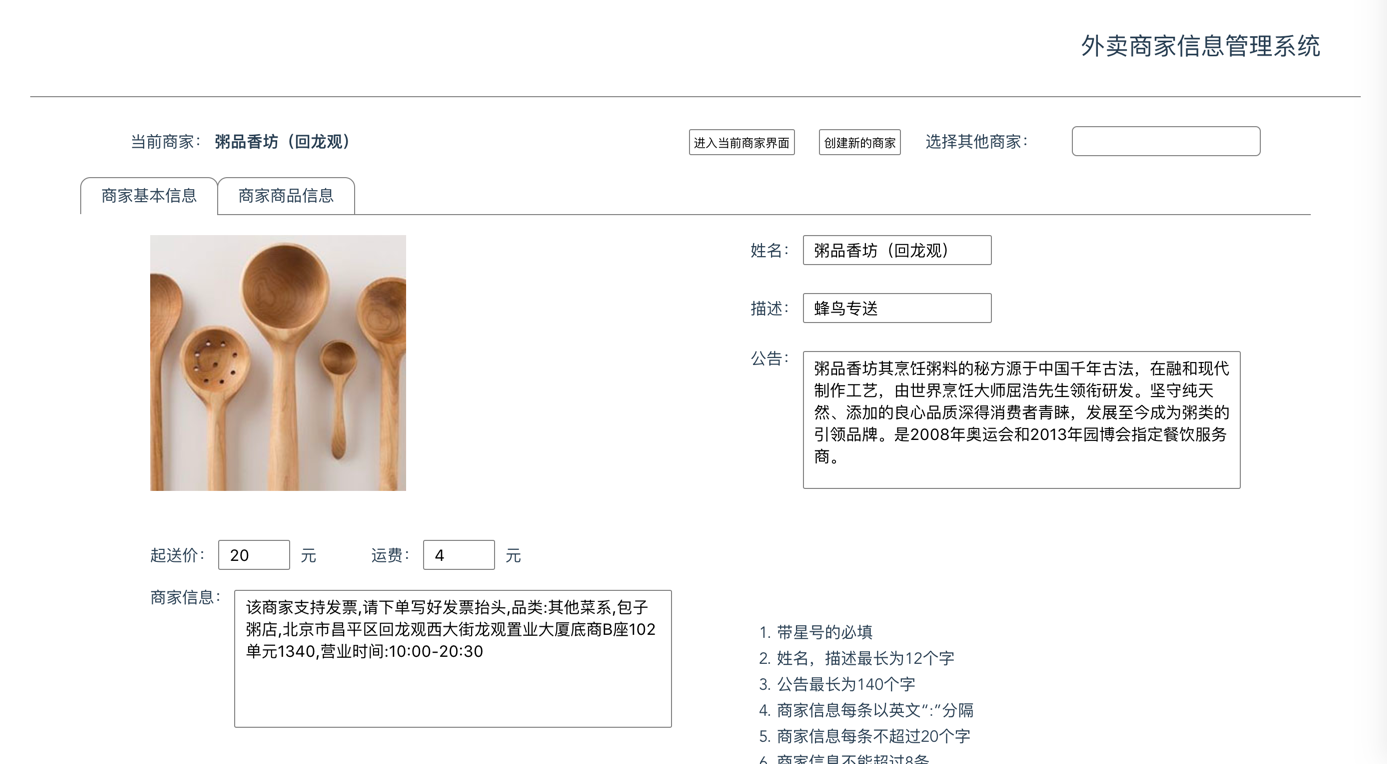


图3.8 外卖商家管理模块首页

外卖商家模块的前端界面主要面向的是管理人员，在界面上的美观需求比较低，没有提供设计稿，但是主要采用的仍然是三段式，内容区域纵向排列，超出视口范围的内容中滚动查看，详细的管理员界面设计可以参考系统实现中的界面设计。

这样的设计方案是现代的SPA（Signal Page App）单页应用的设计方法[8]，实现了切换导航，页面不刷新的效果，提高了用户体验。

# 4 系统实现

## 系统实现说明

本项目在OS环境下开发，使用Node.js的Express搭建的服务器，数据库使用了非关系型数据库Mogodb，前端采用的Vue.js框架进行开发，前后端完全分离，前端依靠Rest API获取后端数据[9]，本章依然从用户点单模块以及外卖商家管理模块的角度介绍该系统是如何实现的。

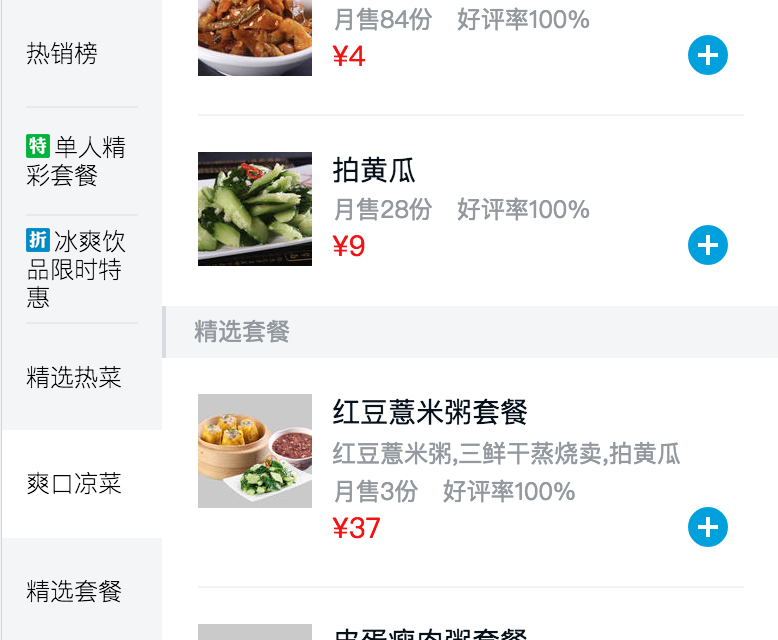
## 用户点单模块

该模块最主要的作用是提供给用户商家商品的信息，然后用户执行点单操作。这里的一个重点就在于需要给用户良好的体验，需要界面使用流畅，布局合理，所以这个模块的主要难点在于前端界面的构建，其中牵扯到的前端功能复杂繁多，这里选择几个重要的前端功能点进行讲解。

### 分组菜单联动功能



（a）



（b）

图4.1 商品组件联动效果图

如图4.1所示，分组菜单联动指的是，左边分组和右边商品的一个联动效果，当点击左侧分组时，属于该分组的第一个商品定位到内容区域顶部；当滑动右侧商品栏时，距离内容区域顶部最近的商品的所属分组会高亮，随着右侧商品的高速滑动，左侧分组也会依次高亮[10]。

结构设计：左侧是只有一层的列表，每个表项对应一个index，包含着每个分组的文本（图4.1（a）中左侧的热销榜，单人精彩套餐等）。右侧是一个有两层的列表，第一层的每个表项和左侧的每个表项相对应（图4.1（a）中右侧的热销榜，单人精彩套餐等），然后每个表项里又包含第二层列表，该列表的每个表项就是上层表项所对应分组所包含的商品（图4.1（b）中右侧的拍黄瓜，红豆薏米粥套餐等）。

点击左侧分组定位功能的实现：点击右侧分组，触发事件，获得触发事件分组的index，根据这个index找到右侧所对应的分组，然后获取右侧所对应分组的二级列表，再取得该列表距离内容区域顶部的位置，然后滚动到指定位置。在点击元素上添加这个事件，由于渲染时，点击的元素自带index，可以把该index传入事件函数中，事件响应函数获取到了该index后，就可以用js原生方法获取右侧的这个列表，然后根据index找到对应的列表项，然后这里调用了一个接口，这个接口可以接收某个元素，然后计算出元素距离内容区顶部位置，然后进行滚动。

该部分代码：

      selectMenu(index, event) {

        if (!event.\_constructed) { //判断事件是否是原生事件

          return;

        }

        let foodList = this.$refs.foodsWrapper.getElementsByClassName('food-list-hook');

        let el = foodList[index];

        this.foodsScroll.scrollToElement(el, 300);

      }

右侧滑动，左侧依次高亮的实现：获取右侧列表，在浏览器进行dom元素渲染时，计算右侧列表每个表项的高度，然后累加放入一个数组里，在这里渲染好右侧列表时，所构成的数组为[0, 1037, 1191, 1313, 1624, 1840, 2076, 2388, 2889, 3585]，这个数组代表的是当表项距离内容框顶部0～1037px时，所对应左侧列表index=0的表项，然后依次类推。然后当滑动事件触发时，获取实时的表项距离内容框顶部的距离，当距离属于数组某个范围，则该范围所对应的index的左侧列表表项就进行高亮。

该部分代码：

计算列表高度数组：

      \_calculateHeight() {

        let foodList = this.$refs.foodsWrapper.getElementsByClassName('food-list-hook');

        let height = 0;

        this.listHeight.push(height);

        for (let i = 0; i < foodList.length; i++) {

          let item = foodList[i];

          height += item.clientHeight;

          this.listHeight.push(height);

        }

}

根据实时距离返回应该高亮的index：

      currentIndex() {

        for (let i = 0; i < this.listHeight.length; i++) {

          let height1 = this.listHeight[i];

          let height2 = this.listHeight[i + 1];

          if (!height2 || (this.scrollY >= height1 && this.scrollY < height2)) {

            return i;

          }

        }

        return 0;

}

在这个地方，用到了Vue的一个特性，计算属性[11]。当你有一些数据需要随着其它数据变动而变动时，一般的方法是监听源数据，源数据触发时更新新的数据，而Vue直接封装了这样的方法，直接写出源数据与新数据的对应关系，源数据变化时，新数据就自动发生变化，正好应用在这里距离发生了变化，高亮的index也就发生变化。

根据左对应右，右对应左这样的两侧对应，就能达到整个商品列表的联动效果，故商品的排序必须是按照分组的顺序进行排序。

### 收藏商家功能

收藏商家功能是这类应用中最基础的一个功能之一，该功能分为两个部分，一个部分是将商家添加到收藏列表，该功能比较简单，点击按钮后，将商家id压入一个已收藏的商家id数组中即可。在这里主要想要介绍的功能是当收藏了商家之后，再次进入商家界面的时候显示的是已收藏见图4.2：



图4.2 收藏商家后效果图

现在浏览器都支持一个H5的新性质叫做localStorage和sessionStorage，这两个性质是全局对象window的两个属性[12]，属性的作用是存储数据，session的生命周期为当窗口关闭，而local则会一直存在，直到被清空，这些数据都存储在浏览器端。故可以利用localStorage，每当用户进入这个单页应用时，就可以通过window对象获取到存储在浏览器端关于这个页面的数据。

主要逻辑为，当进入到这个页面时，不需要马上读取localStorage，而是当切换到商家组件时，读取该属性，读取后的主要流程为：

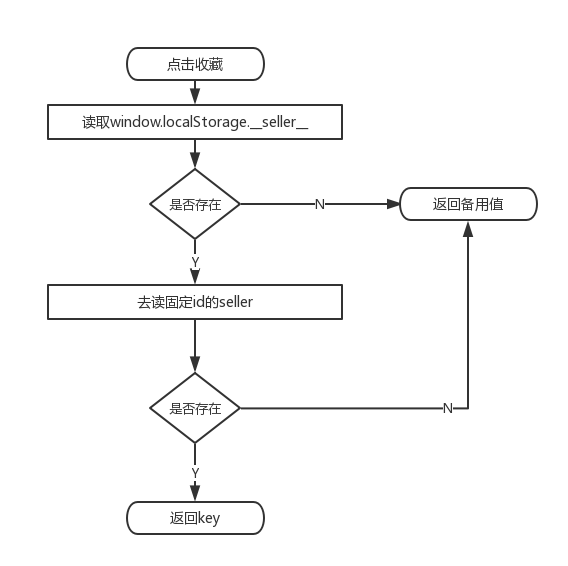


图4.3 读取localStorage流程图

该部分代码：

export function loadFromLocal(id, key, def) {

  let seller = window.localStorage.\_\_seller\_\_;

  if (!seller) {

    return def;

  }

  seller = JSON.parse(seller)[id];

  if (!seller) {

    return def;

  }

  let ret = seller[key];

  return ret || def;

};

如果这个key存在，那么切换到商家页面时，就可以把商家置为已收藏，但是这里只是读取，一定是把数据存储了之后才能有读取，那么我们在点击收藏商家时，不但应该把商家id存入已收藏商家数组，还要在localStorage中设置key：

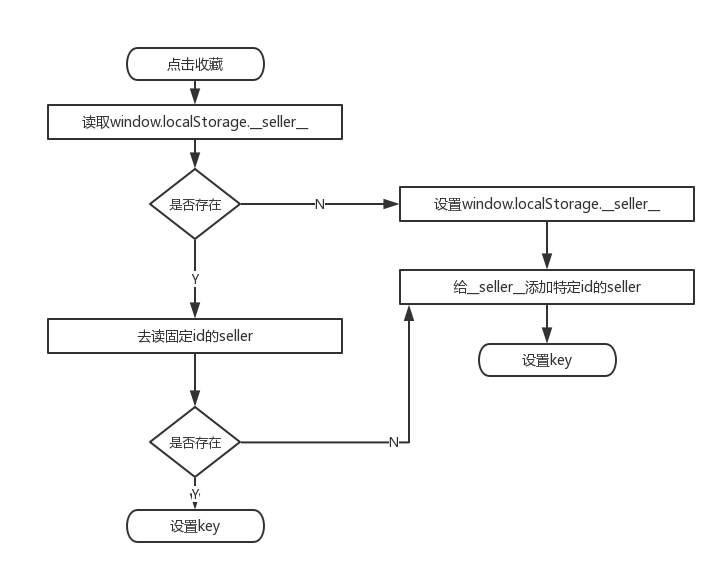


图4.4 设置localStorage流程图

该部分代码：

export function saveToLocal(id, key, value) {

  let seller = window.localStorage.\_\_seller\_\_;

  if (!seller) {

    seller = {};

    seller[id] = {};

  } else {

    seller = JSON.parse(seller);

    if (!seller[id]) {

      seller[id] = {};

    }

  }

  seller[id][key] = value;

  window.localStorage.\_\_seller\_\_ = JSON.stringify(seller);

};

### 评论筛选功能

评论筛选功能给用户提供功能筛选想看的评论种类，评论分为两种类型，满意与不满意，然后评论根据内容还可以分为有内容无内容，现在需要根据这些条件对评论进行一个筛选。

功能描述：默认显示只有内容的所有评论，包括满意与不满意，提供四个按钮，全部，满意，不满意，只显示有内容的评价这些按钮点击后回显示对应条件内容。



图4.5 评论组件效果图

这里主要的特别之处在于组件封装，在E-R中就可以发现，拥有评价的除了商家还有每个商品，然而在设计稿中商家评论模块和商品的评论模块基本一致，那么我们就应该把这个评论抽象为一个功能组件，另外一个重点在于，传统的可刷新的界面中，当评论列表作出了更新，那么整个页面也会一并更新，这里并不会请求后台数据，直接就可以实现无刷新更新。

由于Vue的数据采用的是双向绑定[13]，指的是数据和视图是相互绑定的，视图更改，那么数据更改，视图更改，数据也会更改，并且数据发生任何变化时，页面并不会刷新。每条评论都含有一条属性叫做type，可以根据这个属性进行筛选。首先为每一条评论的html绑定一个v-show，这个也是Vue的一个特性，当v-show里的值为真，该条评论才显示。

      <li v-show="needShow(rating.rateType,rating.text)" v-for="rating in ratings" class="rating-item border-1px">

评论相关内容

      </li>

而这里绑定的v-show为一个函数，函数中传入两个参数一个为需要显示的评论的属性，一个为评论的内容，然后四个按钮对应着改变这两个参数的值，参数可以看作为数据，数据改变视图改变，然后每当参数发生改变时都会触发计算这个函数，然后根据返回值判断是否显示这条评论，从而达到了无刷新评论筛选的功能。

该部分代码：

      needShow(type, text) {

        if (this.activeOptions.onlyContent && !text) {

          return false;

        }

        if (this.activeOptions.selectType === ALL) {

          return true;

        } else {

          return type === this.activeOptions.selectType;

        }

      }

传统的列表更新都是依靠前端传信号给后台，后台数组去除不合格的部分，再返回前台渲染，这样的体验在单页应用程序非常不友好，以这样的方式实现更能带给用户更优雅的体验。

### 添加至购物车小球飞过效果

当我们点击了每个商品上的加号时，就会把该商品添加到购物车中，如果每次都是点击过后，购物车就自动添加了商品，那么这样的效果并不友好，可能用户并不能马上反应过来自己的商品已经添加购物车成功，那么这个时候就需要一个效果，点击添加时，添加按钮会产生一个小球飞抛物线角度入购物车。如同4.6：



（a）



（b）

图4.6 小球飞出效果图

小球的飞向目标购物车，由需求分析所知，添加购物车动作不仅在这个右侧商品栏会发生，而且在点击商品，进入商品详情后也可以触发添加至购物车这个动作，故在两个地方都有这个操作，所以添加按钮应该封装为一个添加按钮组件。

这个动作的难点一在于每次都是从不同的位置飞入购物车，所以要实时计算小球的位置；难点二在于如果有人手快，连续点了几次添加，那么也应该同时出现几个小球，同时执行这个飞入购物车效果。

首先我们应该明确一点，v-show关联着数据，数据发生改变，将会促发v-show里面的动作，如果返回值为真，那么含有v-show属性的元素就会显示。现在又利用了Vue的一个特性，叫做过度动画，当一个元素从显示变为消失，或者从小事变为显示，那么就可以定义消失或者显示前后的小球样式，从而产生动画效果，利用这一点，首先把就小球置放在结束点，结束点就是购物车中心点，且结束点位置不会发生变化，并且把小球此时置于消失的状态，那么当控制v-show属性让小球显示的时候，计算点击按钮的位置，就可以使小球显示前在起点位置，显示后在终点位置了。

接下来需要定义过渡动画，过渡是一个抛物线，垂直方向变速，水平方向匀速，就可以达到抛物线效果。因为有两个方向的移动所以需要给小球外表再套一层div，小球本身往一个方向移动，外层再往一个方向移动，达成抛物线效果。

此部分代码为：

显示前代码：

      beforeEnter(el) {

        this.dropBalls.forEach((ball) => {

          let rect = ball.el.getBoundingClientRect();

          let x = rect.left - 32;

          let y = -(window.innerHeight - rect.top - 22);

          el.style.display = '';

          el.style.webkitTransform = `translate3d(0,${y}px,0)`;

          el.style.transform = `translate3d(0,${y}px,0)`;

          el.style.transition = 'all 0.4s cubic-bezier(0.49,-0.29,0.75,0.41)';

          let inner = el.getElementsByClassName('inner-hook')[0];

          inner.style.webkitTransform = `translate3d(${x}px,0,0)`;

          inner.style.transform = `translate3d(${x}px,0,0)`;

          inner.style.transition = 'all 0.4s linear';

        });

}

显示后代码：

      enter(el, done) {

        /\* eslint-disable no-unused-vars \*/

        // 手动重绘画

        let rf = el.offsetHeight;

        el.style.webkitTransform = 'translate3d(0,0,0)';

        el.style.transform = 'translate3d(0,0,0)';

        let inner = el.getElementsByClassName('inner-hook')[0];

        inner.style.webkitTransform = 'translate3d(0,0,0)';

        inner.style.transform = 'translate3d(0,0,0)';

        setTimeout(done, 400);

}

解决了动画效果，但是只能支持单个小球移动，经过时间测试，手的连续点击速度，最多在屏幕上只能同时出现五个小球，那么就声明五个div作为小球，然后用两个数组来控制这些小球，一个数组用来表示这个五个小球的状态，另一个数组用来作为队列接收执行动画中的小球。每次点击添加后，就从数组中抽出一个没有执行动画的小球，将它的状态由消失变为执行（this.show = true），然后把这个小球添加到队列中，当这个小球动画执行完后，再将该小球弹出，把状态变消失（this.show=false）。连续按，就连续将小球添加到队列中，然后依次弹出。

该部分代码：

小球状态控制数组：

  balls: [{

    show: false

  }, {

    show: false

  }, {

    show: false

  }, {

    show: false}, {

    show: false}],

  dropBalls: []

小球压入队列函数：

BAll\_JUMP(state, el) {

    let ball = state.balls.find((ball) => {

      return ball.show === false;

    });

    ball.show = true;

    ball.el = el;

    state.dropBalls.push(ball);

  }

小球执行完动画后的动作；

      afterEnter(el) {

        let ball = this.dropBalls.shift();

        if (ball) {

          ball.show = false;

          el.style.display = 'none';

        }

      }

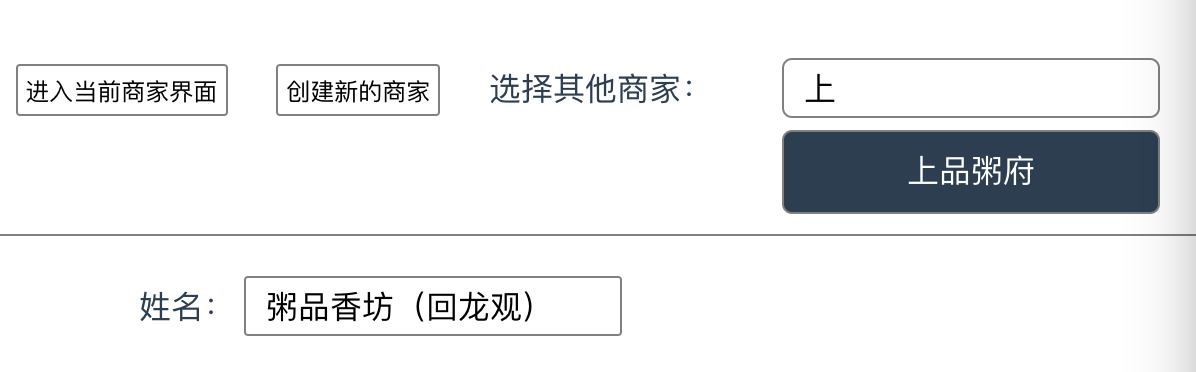
五个小球完全能够满足手连续点的执行时间，所以当连续点的次数大于五时，最开始执行动画的小球也已经从队列里弹出，有能力执行新的一轮动画了。

## 外卖商家管理模块

该模块的主要角色为管理员，管理员接受到商家发来的需求然后根据需求对商家信息进行一个管理，大致为增删改查，该模块的一些基本功能都是很多系统常用功能，在这里不作介绍。该模块的也是采用组件化开发方式，可以给管理员带来更好的操作体验，更是提高了管理员的操作速度等，故这里主要也从前端角度介绍几个具体功能的设计与实现。

### 4.3.1 商家搜索功能

商家搜索功能指的是，在搜索框输入关键词，就能搜索出名字中包含该关键词的所有卖家，在传统的应用中，输入关键词后，必须点击确认按钮，然后才向后台请求数据，虽然现在的AJAX不用刷新页面，但是当数据特别大时，这样的体验仍然不是特别好，在这里我们主要采用的一种方式为：当界面载入时，向后台请求所有商家的数据，但是该数据是一个只包含商家名字和商家ID的键值队数组，并不包含商家的所有信息，然后利用Vue的双向数据绑定的特性，将该数据数组绑定在搜索框上，然后根据关键字进行数组筛选，保留筛选通过的商家在数组上，然后会自动渲染在弹框上。这只是一条搜索出商家信息的主要逻辑，实际完成这个控件还和前端逻辑操作有关：



（a）



（b）

图4.7 搜索商家效果图

如图所示，默认状态应该没有这个列表栏，当输入数据后，列表才会执行上面所说的那个逻辑，然后当删除所有关键词后，此时列表并没有消失，而是显示了所有商家信息，如果继续输入关键词会重复之前的动作，当我们点击除了弹框和搜索框以外的地方，弹框才会消失，然后点击弹框上你想找的商家名，就能搜索到该商家并把该商家信息立即渲染到商家基本信息组件和商家商品组件上。流程图：

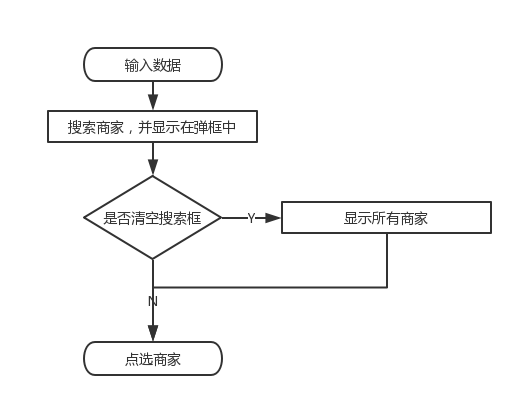


图4.8 搜索商家流程图

前端逻辑和搜索逻辑相结合才能完成这个搜索功能，并且需要把搜索的逻辑放在前端逻辑过程中的合适的位置。

HTML元素绑定事件代码：

      <input type="text" class="selectinput" @click.prevent="" @input="inputName" ref="selectinput" />

      <ul class="select-list" v-show="isShowlist" @click.prevent="">

        <li class="list-item" v-for="item in currentList" @click="selectSeller(item)">

          {{ Object(item).name }}

        </li>

      </ul>

发生input事件触发的函数：

      inputName() {

        this.setAllitem();

        let currentValue = this.$refs.selectinput.value;

        if (currentValue === '') {

          this.setAllitem();

        } else {

          this.isShowlist = true;

        }

        this.currentList = this.currentList.filter((item) => {

          return item.name.indexOf(currentValue) !== -1;

        });

      }

每次执行输入事件时都要把数组还原为包含所有商家信息的数组，不然再次筛选会造成商家信息丢失。

到这里，搜索的逻辑就执行完了，但是搜索完成了以后必须让弹框消失，第一种情况就是选中商家表项后消失，这样只需要把消失动作添加到点击事件中即可；第二种情况是，搜索商家搜素到了一半，不想继续搜索了，那么此时应该点击弹框和输入框以外的所有事物都可以让弹框消失，那么就在body上添加事件：

      document.body.onclick = function () {

        this.isShowlist = false;

      }.bind(this);

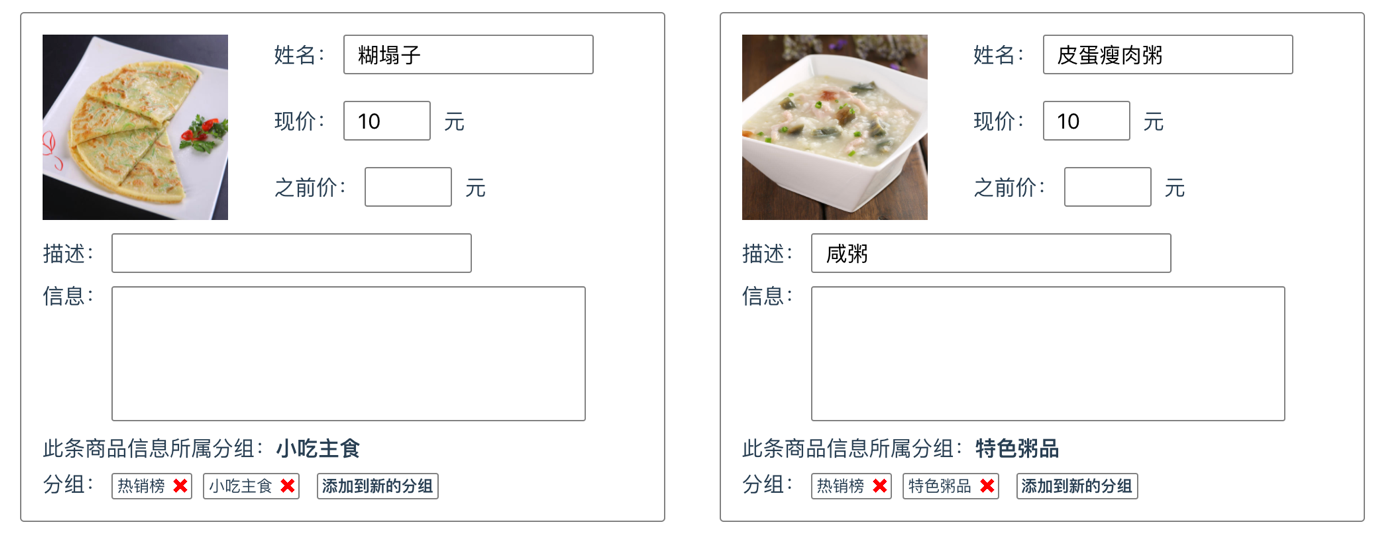
然后不在输入框和弹框上执行该动作即可（HTML上@click.prevent=""）。

### 4.3.2 商品分组管理功能

由E-R图可以得，数据库的实体中除了商品还要分组，这两个是多对多点关系，一个商品可以包含在多个分组当中，一个分组也可以包含多个商品，然后由于我们采用的是Mongodb，由数量级少的包含数量级多的表。故会出现这样的情况，例如：A、B、C是分组，1、2、3、4是商品，A包含123，B包含12，C包含34，那么123都有两份拷贝。又由于页面管理商品信息是按照客户端的商品信息顺序排列的，所以当一次性渲染所有信息后，同一个商品（名字相同）可能会出现两次，那么每个商品除了显示该商品被哪些分组包含以外，还要显示该份拷贝是属于哪个分组的，如图：



（a）



（b）

图4.8 商品信息管理图

如图所示皮蛋瘦肉粥商品有两份，一份有更多的信息，拷贝属于热销榜分组；另一份信息相对少一些，拷贝属于特色粥品。我们定义为同名同照片的商品为同一个商品（点餐后，无论点哪个，送到的都是一模一样的餐品），这样做的原因是，两份拷贝，第一个分组里的拷贝属于热销榜，那么他可以在商品页面中靠前显示，并且可以给他添加优惠或者信息描述，让他一目了然，但是同时该商品有自己的种类分组，以防用户分组定位时找不到该商品。同时防止特色商品不小心被删除难以找回。

故信息的管理应该是在逻辑上关联的，如果改其中任何一份拷贝的名字，现价，头像，都会造成另外一份拷贝的改变。描述，信息，之前价的改变都不会造成另一份拷贝的改变，分组操作是关联的，如果移除属于热销榜的皮蛋瘦肉粥的热销榜分组，那么该分拷贝将会消失，如果移除的是粥品分组，那么该份拷贝会被爆了，粥品分组的拷贝将会消失。操作属于粥品分组的皮蛋瘦肉粥效果亦然。

除了商品上的操作还有分组上的操作，删除一个分组，属于该分组的所有商品将会消失；新建一个分组，可以在某个拷贝上执行添加到新的分组操作，那么该拷贝会再拷贝一份放到新的分组。

新建一个商品默认放在热销榜分组里。

该部分代码：

将商品从某个分组中移除：

      removeOut(item, label) {

        let goods = this.goods.filter((good) => {

          return good.name === label;

        });

        goods[0].foods.forEach((food, index) => {

          if (food.name === item.name) {

            goods[0].foods.splice(index, 1);

          }

        }, this);

      }

将分组删除：

          this.goods.forEach((good, index) => {

            if (good.name === item.name) {

              this.goods.splice(index, 1);

            }

故当想新建一个商品放在一个全新的分组里的：

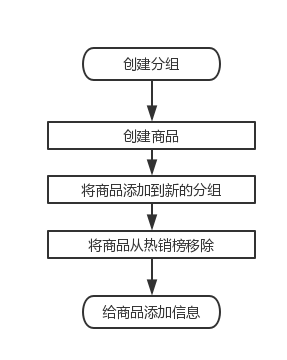


图4.8 新商品添加到新分组流程图

由于Vue是双向绑定，数据驱动，那么渲染在界面上的东西只需要写好固定的样式，然后具体操作被渲染的数据即可，故代码部分都只是在this.goods和this.foods这两个对象上进行的具体的数据操作，数据操作成功后，视图将会进行自动更新。

### 4.3.3 头像组件的封装



图4.9 商品信息管理界面

如图4.9所示，项目中多处有头像框，头像必须实现图片上传功能。首先因为多处需要使用此功能，故首先封装为一个组件，其次实现的功能细化为，上传图片，并且视图发生改变。但是，如果没有保存，刷新过后，图片是上传到服务器上了，但是商品的信息并不会发生改变。最后，就是保存过后，再刷新页面时，图片才会显示最新上传的图片。该部分代码：

      preview(event) {

        upload(event, this, (response) => {

          response = response.body;

          if (response.errno === ERR\_OK) {

            let pathArr = response.imagePath.split('/');

            let path = 'http://localhost:3000/' + pathArr[pathArr.length - 2] + '/' + pathArr[pathArr.length - 1];

            this.father.$emit('changeAvatar', path, this.item);

          } else {

            alert(response.data.message);}});}

5 系统测试

系统测试[14]是将经过集成测试的软件，作为计算机系统的一个部分，与系统中其他部分结合起来，在实际运行环境下对计算机系统进行的一系列严格有效地测试，以发现软件潜在的问题，保证系统的正常运行。

## 5.1 功能测试

功能测试也叫黑盒测试或数据驱动测试，测试软件系统的功能是否正确，其依据是需求文档，不需要考虑整个软件的内部结构及代码.一般从软件产品的界面、架构出发，按照需求编写出来的测试用例，输入数据在预期结果和实际结果之间进行评测，进而提出更加使产品达到用户使用的要求。

本文功能测试主要才采用等价类划分和边界值分析法进行测试。

等价类划分[15]是把所有可能的输入数据，即程序的输入域划分成若干部分（子集），然后从每一个子集中选取少数具有代表性的数据作为测试用例.该方法是一种重要的，常用的黑盒测试用例设计方法。

边界值分析[16]方法是对等价类划分方法的补充，主要是考虑系统错误大多是发生在输入或输出范围的边界上，而不是发生在输入输出范围的内部。因此针对各种边界情况设计测试用例，可以查出更多的错误。

本次功能测试的环境如表5.1：

表5.1 系统测试的环境

|  |  |
| --- | --- |
| 服务器端操作系统 | CentOS 6 mini |
| 搭建环境 | 搬瓦工 |
| 数据库客户端 | mongodb 3.4.4 |
| 编程语言 | node.js express框架 |
| 版本控制 | Git |
| 编码格式 | UTF-8 |

由于系统的功能测试较多，在此只列出用户端系统模块的功能测试。

## 5.2 健壮性测试

健壮性测试指的是系统在不同情况下能否正常运行的能力。健壮性一般有两层意义：容错能力和恢复能力[14]。

该程序主要用户运行环境是微信，故只要微信下能正常使用即可，但是为了满足部分用户电脑上使用微信进入外卖子系统，故也测试各大浏览器的兼容性，具体测试情况见表5.3：

表5.3 浏览器的兼容性测试

|  |  |  |
| --- | --- | --- |
| 浏览器 | 版本 | 测试结果 |
| Google Chrome | 56 | 正常 |
| Safari | 10.1 | 正常 |
| Opera | 37 | 正常 |
| IE | 8及以上 | 正常 |
| IE  微信 | 7及以下  6.5.8 | 功能不能完全正常使用  完全正常 |

以上为用户模块的测试结果，外卖商家管理模块采用的是同样的开发方式，并且开发中css自动会加上兼容性，故两个模块的兼容性是一致的。通过上表可以见得，除了老旧的IE7及以下，功能显示表示正常，健壮性测试通过。

## 单元测试

互联网的快速发展，给web开发人员带来了前所未有的挑战。对于前端开发，前端开发者所需要编写的js早已不是那些寥寥几行的视觉效果代码。代码量的大增，多人协同，人员素质悬殊不齐，这都需要一个标准，来对代码的规范性进行控制。Jasmine作为一个前端团队使用的测试框架，便运应而生[15]。本项目release预计会进行单元测试。

## 5.4 测试总结

通过对系统进行相应的功能测试和健壮性测试，确实发现了系统存在的问题，并及时进行了修正，修正后，系统的功能性更加稳定，出错率明显降低，使系统更加稳定的运行。

在所有测试结束后，该系统基本能保持稳定运行，能确保一定的兼容性，确保功能的稳定运行，满足了用户的需求。

# 结 论

本系统是基于Vue实现的外卖商家子系统的设计与实现，分为两大模块，客户端和后台管理。客户端是移动端的用户点单模块，后台管理是PC端的外卖商家管理模块。商家通过线下手段提供更改需求给管理员，后端管理员接受商家的请求，然后根据请求内容线上利用外卖商家管理模块修改商家信息，然后用户点单模块更新。

用户点单模块主要植入在微信应用中，通过微信可以关注商家的公众号，然后公众号可以推送实时的信息，如果用户想要点单就可以直接点击点单进入商家的界面，进行下单处理。

该文章描述了一个外卖商家子系统的构建过程，包括了以下几个方面：

1. 讲述了为什么要开发这个系统，这个系统的意义与背景。
2. 通过具体的几项可行性分析了该系统的事实可行性，因为采用的是组件化开发，故最后细化到了组件进行需求分析。
3. 把抽象的组件整合，设计了用户点单模块的功能设计和外卖商家管理模块的功能设计，然后对它们进行了数据库设计和前端界面的一个设计。
4. 项目的实现才是项目的血液，之后讲述了该项目几个重点功能的实现，分别从为什么要开发这个功能、开发这个功能利用的原理是什么、如何利用原理进行的开发、具体的代码实现，这四个部分来详细描述了每个功能是如何实现的，最后分析了一下传统的实现和本项目实现的区别，由于是单页应用，追求的是流畅性，简易性。
5. 系统测试是对系统的验收测试，包括功能性测试和健壮性测试，在此过程中，不断修复BUG，完善系统功能和稳定性。

该系统基本能保持稳定运行，能确保一定的兼容性，确保功能的稳定运行，满足了用户的需求。

参 考 文 献

[1] 吕大豹.参考淘宝团队实践：一个简单粗暴的前后端分离方案[J/OL].2015-01-08. http://developer.51cto.com/art/201501/463338.htm.

[2] 李鲁静. 大学生网络外卖消费现状及发展研究[J]. 商场现代化, 2015(2):25-25.

[3] Chaniotis I K, Kyriakou K I D, Tselikas N D. Is Node.js a viable option for building modern web applications? A performance evaluation study[J]. Computing, 2015, 97(10):1023-1044.

[4] Chodorow K, Dirolf M. MongoDB: The Definitive Guide[J]. 2010.

[5] 李曼, 李斌. 软件工程可行性分析部分的教学设计实践与研究[J]. 计算机光盘软件与应用, 2013(20):285-285.

[6] 邓超. WEB前端应用组件化开发研究[J]. 科研, 2016(11):00042-00043.

[7] 陈文宇. 面向对象的关系数据库设计[J]. 电子科技大学学报, 2002, 31(1):53-56.

[8] 米可夫斯基, 鲍威尔包勇明. 单页Web应用 : Single page web applications : JavaScript从前端到后端[M]. 人：民邮电出版社, 2014.

[9] Mardan A. REST API[M]// Pro Express.js. Apress, 2014:249-261.

[10] 闫娜. “滑动门”技术在网页导航栏中的设计与应用[J]. 信息与电脑:理论版, 2016(16):86-86.

[11] 尤雨溪. Vue.js官方文档：计算属性[J/OL].

[12] Fox P. The localStorage API[J]. Informationweek.

[13] 杜震洪, 张丰, 刘仁义,等. 一种基于访问器劫持的前端数据双向绑定实现方法:, CN 105867919 A[P]. 2016.

[14] 赵则章, 江建慧. 操作系统健壮性测试方法研究[J]. 计算机工程与应用, 2007, 43(7):93-97.

[15] 侯海先. 基于QUnit的JavaScript自动化单元测试框架的设计与实现[D]. 北京大学, 2012.

# 致 谢

时间悄无声息地流逝着，还没等反应，大四也已到结尾。大学求学之路漫长且艰辛，但是艰辛的路途中，也充满了欢笑汗水悔恨，这一切切大学的细节让我难以忘怀，让我沉迷于中。刚入学从一个懵懂的少年，充满朝气充满活力、每天只是想着如何才能玩得更开心、如何才能在学校拿个一官半职，到了一个成熟的少年，经过了求职的洗礼，知道努力奋斗学习才是硬道理，减少了玩耍的时间，更多的时间投入在了学习和与人的相处之中，大学改变了我，我也要从大学毕业了，感谢大学。

该论文是在导师的耐心帮助下才得以完成。导师不辞辛苦，手头还有自己的工作忙不过来，却仍然定时约我们出来开会，一个人一个人轮流地检查我们的论文进度，项目完成的进度，并对我们的论文和项目都提出了指导性意见，认真细致，无微不至。我意识到有了导师的帮助，这个论文不仅仅只是完成毕业任务，更是自身学习提升的一个过程，学习更新更多的知识，学习和导师的交流，学习和周围同学的合作。导师所带给我的提升并不是单单几句话就能描述的，我想对导师表达我最真诚的感谢以及由心的尊敬与爱戴。

我很庆幸自己高考能够选择大连理工大学，也很庆幸被调剂到了软件工程这个专业，如果当年就被分配到按照自己的想法所填写到的专业，我现在还能找得到工作吗？正是有了大连理工大学，我生命中重要的一站，我才有这样的成长，向大连理工大学致以我最真心的感谢。

在求学的路途中，遇见了很多前辈，他们没有高人一等，他们耐心给予你指导，他们告诉你哪个方向热门，哪些技术厉害，没有他们的指引，我恐怕还是一介小卒，感谢前辈们。

感谢室友同学，大学生活因你们而快乐，因你们而成长，没有朋友的日子，一个人如何过得下去呢，向我们刷夜的夜晚致敬。

感谢妈妈爸爸奶奶姑姑妹妹等等家人，没有你们的支持，没有你们的信任，我是无法坚持下去的。

感谢我的女朋友一直鼓励我、帮助我。

感谢老师，感谢你们传授给我的知识。

最后，感谢文献作者，让我完成了这篇论文。