

本科生学年论文(设计)

****

**题 目 基于NodeJS的手机商城平台的设计与实现**

**学生姓名 戴肖**

**学 号** 201633070041

**院 系 应用技术学院**

**专 业 软件工程**

**指导教师 许晓龙**

**二Ｏ二Ｏ 年 五月 二十 日**

**声 明**

本人郑重声明：

1. 持以“求实、创新”的科学精神从事研究工作。
2. 本论文是我个人在导师指导下进行的研究工作和取得的研究成果。
3. 本论文中除引文外，所有实验、数据和有关材料均是真实的。
4. 本论文中除引文和致谢的内容外，没有抄袭其他人或其他机构已经发表或撰写过的研究成果。
5. 其他同志对本研究所做的贡献均已在论文中作了声明并表示谢意。

作者签名：

日 期：2020.5.20

基于NodeJS的手机商城平台的设计与实现

戴肖

南京信息工程大学应用技术学院，江苏 南京 210044

摘要：网络不断发展的今天，人们的生活方式也随之发生了改变，大家在网上交流、购物、娱乐等等，其中用的最多的设备就是智能手机，人们也习惯了智能手机的存在，有了需求，就有了供应，随之出现了大量手机厂商，如小米、魅族、华为以及国外的苹果三星等等。我们中国是属于发展中国家，老一辈生活简朴的观念普遍的继承到下一代，所以与外国人显著不同的是，中国人对智能手机的挑选的主要因素就是非常的注重性价比这种东西，大家都很希望物美价廉这个词可以应用到手机上，所以，早期小米公司利用中国人的这种心理开始主打性价比，一直爬到了今天这个位置，时至今日，手机行业出现了百花齐放的局面，越来越多的厂商纷纷模仿小米的营销模式，从而导致了消费者对于手机的挑选开始出现了迷茫。为了解决消费者的这个困扰，本系统聚合了各个厂商的手机，以及用户可以对各种手机进行综合参数对比，在很短的时间内，消费者可以挑到自己最心仪的手机。本系统是基于NodeJS的手机商城PC端，前端利用目前最火的框架VueJS进行编写，后端是用express网络框架以及使用了很具有发展的前景的MongoDB数据库。

关键词：NodeJS，VueJS，手机

Design and implementation of mobile phone mall platform based on NodeJS

DaiXiao

Application Technology College of Nanjing University of Information Science & Technology, Nanjing 210044, China

**Abstract**：With the continuous development of the Internet, people's lifestyles have also changed. Everyone communicates, shop, and entertains on the Internet. The most commonly used device is a smart phone. People are also used to the existence of smart phones and have demand. There is a supply, followed by a large number of mobile phone manufacturers, such as Xiaomi, Meizu, Huawei, and Apple and Samsung abroad. We in China are a developing country. The older generation ’s simple life concept is generally inherited to the next generation. So, unlike foreigners, the main factor in the selection of Chinese smartphones is the emphasis on cost-effective things. Everyone very much hopes that the word of good quality and low price can be applied to mobile phones. Therefore, Xiaomi Company in the early days used the Chinese people's mentality to begin to focus on price and performance, and has climbed to this position today. Today, the mobile phone industry has witnessed a blossoming situation. More and more manufacturers have imitated Xiaomi's marketing model, which has led to consumers' confusion about the choice of mobile phones. In order to solve this problem for consumers, this system aggregates mobile phones of various manufacturers, and users can compare comprehensive parameters of various mobile phones. In a short period of time, consumers can pick their favorite mobile phone. This system is based on the NodeJS mobile phone PC. The front-end is written using the most popular framework VueJS at the moment. The back-end is an express network framework and uses a very promising MongoDB database.

**Keyword**：NodeJS，VueJS，Phone

1. 绪论
2. 系统概论与工具选择

本系统分为PC端的前台系统与后台管理系统。前台页面的样式主要是自己一点点手写，后台页面用的是ElementUI搭建而成，页面简洁，通俗易懂，而且使用的是目前前端最火的框架VueJS，相对于传统开发而言，效率显著提高。后端的是采用NodeJS开发环境，使用的是MongoDB数据库，而MongoDB数据非传统数据库，是文档型数据库，编写方式类似于JavaScript的数组对象，总的来看，前后端采用的都是基于JavaScript语言编写，减少了学习成本。

* 1. B/S架构

B/S架构全称是Browser/Server，意思用户只需要安装一个浏览器如Chrome或者Firefox浏览器或者使用本机自带的浏览器就可以使用系统，本系统无论是前台系统还是后台管理系统，均采用的是B/S架构，相比较C/S架构，无需安装额外的客户端，使用起来更加方便快捷。

* 1. JavaScript语言

JavaScript世界上使用最多的脚本语言，它诞生于1995年，由网景公司的布兰登 • 艾奇负责编写，起初命名为LiveScript，但是由于当时的Java这个语言特别火，所以为了提高自身热度，决定改名为JavaScript，类似于国内的“北大”与“北大青鸟”关系，在同时期还有其他的网页语言VBScript、JScript等等，但是后来均被JavaScript打败，所以在现在的浏览器中，只运行一种脚本语言就是JavaScript，但是几年来延伸出了好多生态，不单单局限于浏览器，使得JavaScript更趋向于编程语言。

* 1. NodeJS

NodeJS非语言，也非框架，是一个JavaScript的运行环境或者是平台，可以称作为JavaScript虚拟机。与PHP、JSP、Ruby等等是平台也是语言不同，NodeJS特点是JavaScript进行编程，运行在Chrome的V8引擎上，运行效率极快。当用JavaScript进行编程时，是不存在BOM和DOM的，只有EcmaScript基本的JavaScript语言部分。在Node中为JavaScript提供一些服务器级别的API，如文件操作以及http服务等等。

* 1. MongoDB数据库

MongoDB是为了能高效率开发Web应用而设计出来的数据库系统，其设计理念以极简、灵活著称，隶属于Web应用栈。MongoDB与传统关系型数据库明显的不同在于，它是面向文档的，文档型数据库与JSON结构相似，不严地说，也可以理解为把不同的JSON数据存放在MongoDB数据库中。

* 1. VueJS

VueJS于2014年诞生，作者是前华裔Google工程师尤雨溪，目前在中国是前端最火的的框架，诞生虽晚，但与老牌Angular、React齐肩，造成三足鼎立的局面，但是与Angular稍微不同的是，其核心概念是组件与双向数据流(基于ES5中的defineProperty来实现的)。

* 1. Visual Studio Code工具

VS Code由微软公司开发，几乎是目前前端工程师的标配，主要优势在于比Atom编辑器流畅 ，比WebStorm更轻量，其本身有着十分丰富的插件系统，无论我们是编辑HTML、CSS、TS、Vue等前端代码，还是JAVA，Python后端代码，我们都可以找到相对应的插件，在开发效率上有显著的提高。

1. 需求分析
   1. 前台系统需求分析
2. 用户注册及登录功能

用户通过输入账户密码进行注册，通过bcryptjs进行密码的加密，它是一个第三方加密库，用来实现Node环境下的密码加密，当密码验证成功后方可进入系统。

1. 首页功能

用户或者游客可以从首页看到商品的轮播、商品的分类。

1. 商品搜索功能

用户或者游客可以输入关键字进行商品搜索，商品的呈现可以是默认顺序排序、按销量升序、按销量降序、按价格升序、按价格降序等。

1. 商品详情功能

用户或者游客可以对商品进行型号及颜色选择、商品参数对比。

1. 购物车管理功能

用户可以进入购物车进行商品的管理，包括选择商品、全部选择或者反选商品、删除商品以及商品的结算。

1. 地址管理功能

用户可以新建地址、删除地址、修改地址、地址列表查看、选择默认地址。

1. 商品结算功能

用户可以选择收货地址、新建地址、修改地址、删除地址、选择付款方式等。

1. 订单管理功能

用户可以查看全部订单、查看待发货订单、查看已发货订单、确认收货订单。

* 1. 后台系统需求分析

1. 品牌管理功能

管理员可以进行品牌的添加、删除品牌、修改品牌以及查看品牌列表。

1. 用户管理功能

管理员可以查看用户列表。

1. 商品管理功能

管理员可以对商品进行添加、删除、修改、查看商品信息。

1. 订单管理功能

管理员可以对订单进行查看、发货处理、查看已发货订单、查看待收货订单。

1. 销量管理功能

管理员可以对查看各个品牌的销量。

1. 库存管理功能

管理员可以对查看各个品牌的库存。

1. 管理员管理功能

管理员可以添加管理员、删除管理员、修改管理员密码、查看管理员列表。

* 1. 可行性分析

可行性分析是对在立项起到运营过程中可能遇到的一些因素进行分析，如社会因素、技术因素以及经济因素等等。

1. 需求可行性：

目前使用人们离不开智能手机，特别是青少年更把数码手机的关注当成一种潮流文化，又因为本系统属于电商，对于手机而言，线上的需求往往大于线下。

1. 技术可行性：

根据客户的需求，选择目前较流行且较稳定的技术，精确细化系统功能，进行可行性研究，并有效的利用现有技术确定并实现开发任务。

1. 操作可行性：

前台页面和后台页面部分使用最新的UI框架，大部分参考其他电商页面进行手写，界面清晰，通俗易懂，让用户很容易地清楚哪个功能是的作用，提高用户体验。

1. 系统概要设计
   1. 前台系统设计

通过用户的需求分析，前台系统是主要分为注册登录功能模块、首页功能模块、商品详情功能模块、购物车管理功能模块、地址管理功能模块、商品结算功能模块、订单管理功能模块与地址管理功能模块。这些功能模块都是递进的，一个结果的输出是另一个条件的输入，功能模块如此划分，主要是把系统精确划分。.如此一来，前台层次结构图如下:

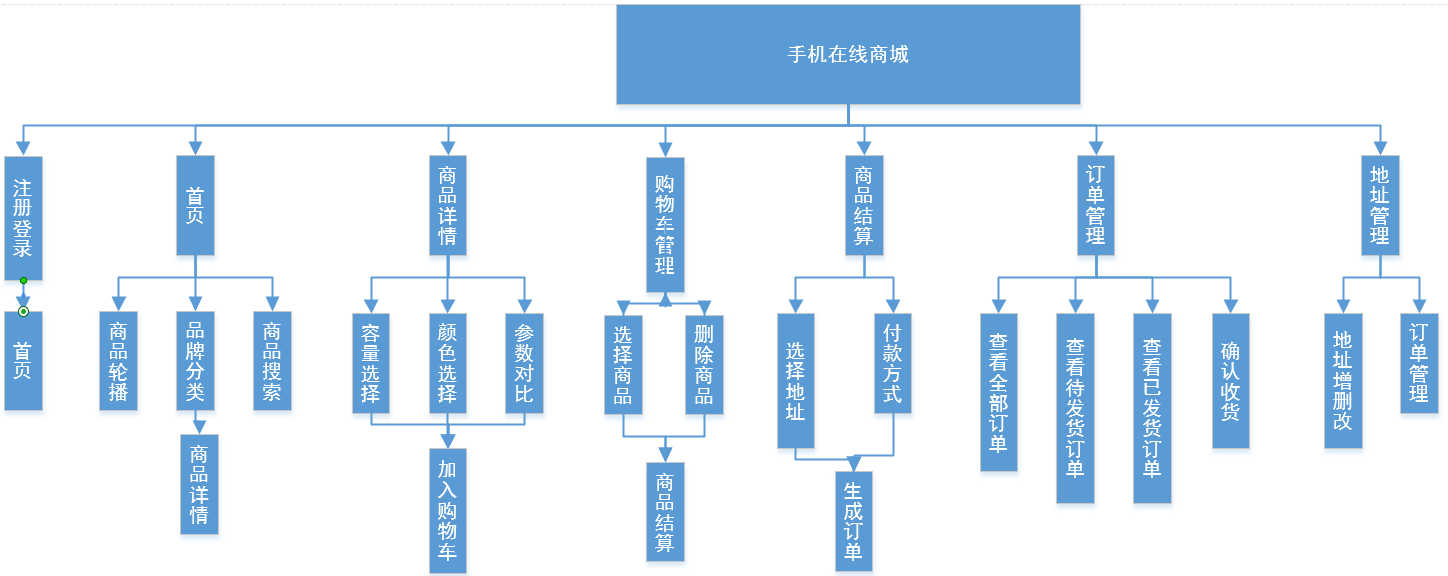


图 4.1 前台系统层次结构

* + 1. 注册/登录功能模块

当用户进行登录的时候，用户输入账户密码，然后前端把账户密码发送给后端，后端去数据库中寻找数据是否匹配，如果匹配则跳转首页，若不匹配则有相关提示；当用户进行注册的时候，用户输入用户名，密码及其确认密码，系统进行数据检查，若检查通过，则跳转登录页。

注册/登录功能模块流程图如下:

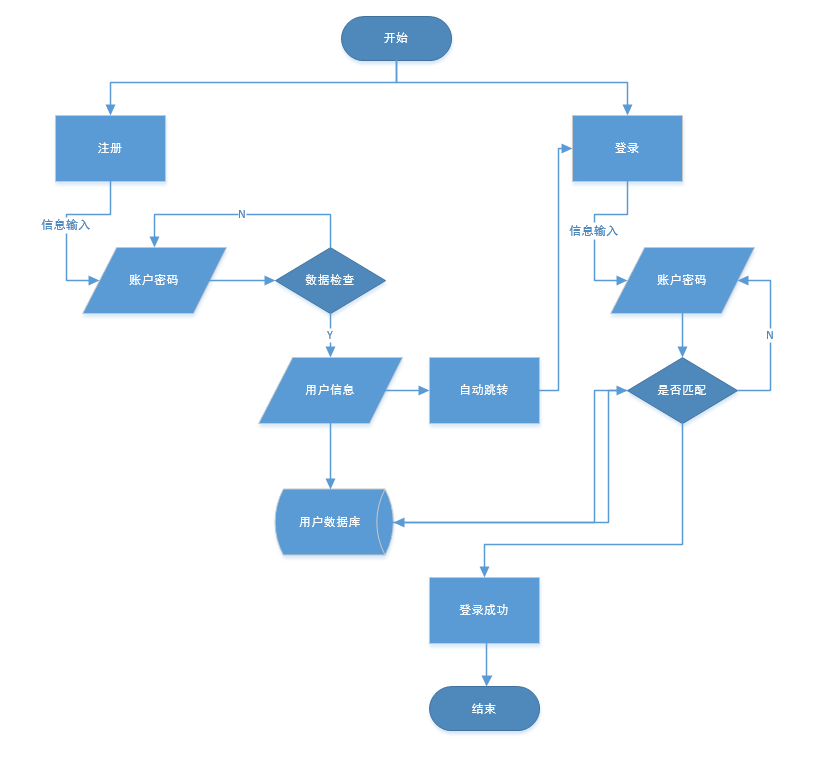


图4.2注册/登录功能模块流程图

* + 1. 首页功能模块

首页功能模块分为三个功能：首页轮播图、品牌的分类以及商品的搜索，如果用户点击商品，则跳转商品详情页。

商品搜索的流程图如下;

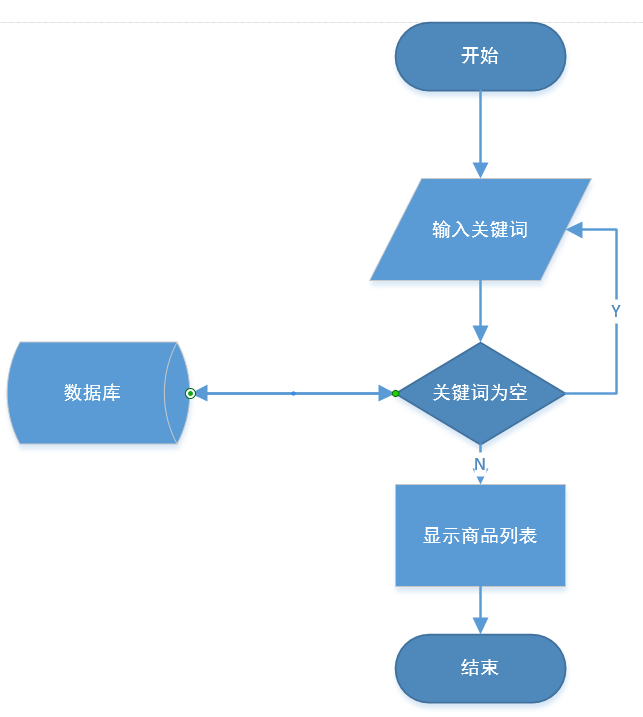


图.4.3商品搜索的流程图

* + 1. 商品详情功能模块

当进入商品详情功能模块的时候，用户选择商品的机身存储和颜色，也可以进行商品的参数的对比，若没有选择机身存储和颜色，则会选择默认选项，然后把商品加入购物车。

添加商品进入购物车的流程图如下:

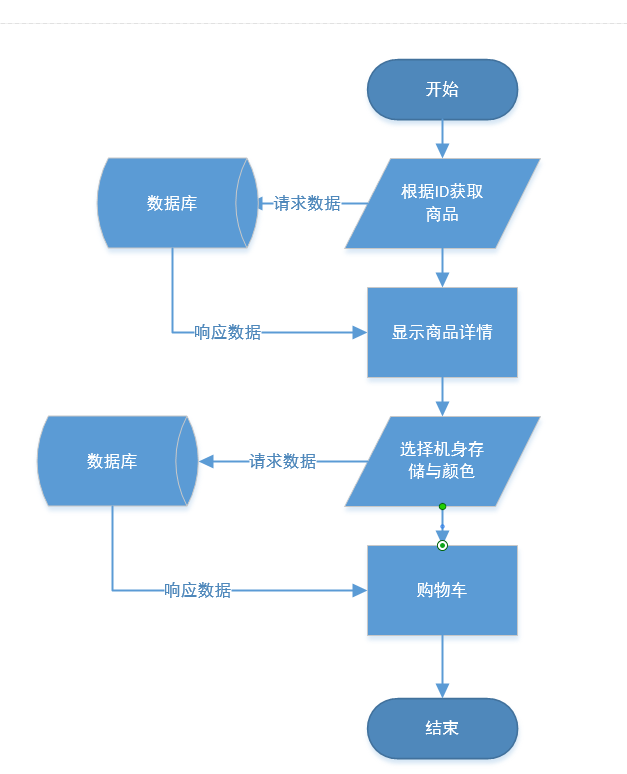


图4.4添加商品进入购物车的流程图

* + 1. 购物车管理功能模块

用户把商品添加入购物车以后，对里面的商品进行选择，或者删除，若没有商品都选中，或者商品全部删除，则都无法进入结算页。

购物车管理功能模块的流程图如下:

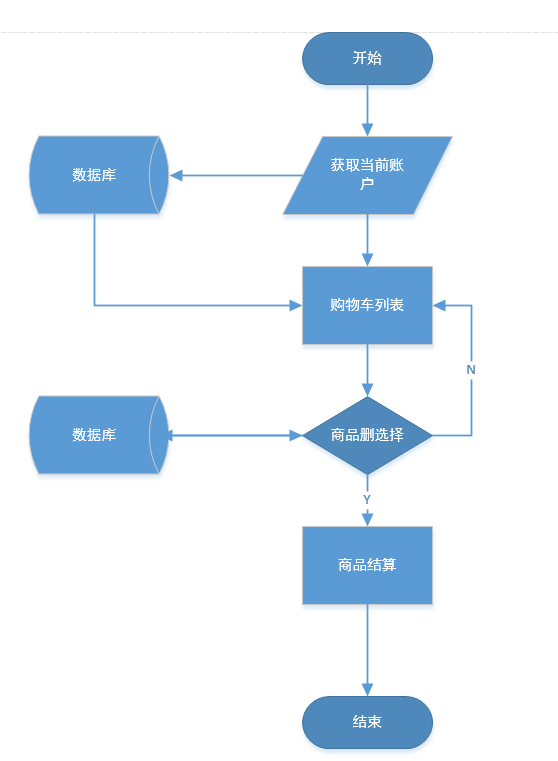


图4.5购物车管理功能模块的流程图

* + 1. 商品结算功能模块

当用户从购物车进行商品结算时，必须要有收货地址，如果没有收货地址可以新建，收款方式，一种是在线支付，另一种是货到付款，如果不做选择默认是在线付款。

商品结算功能模块的流程图如下:

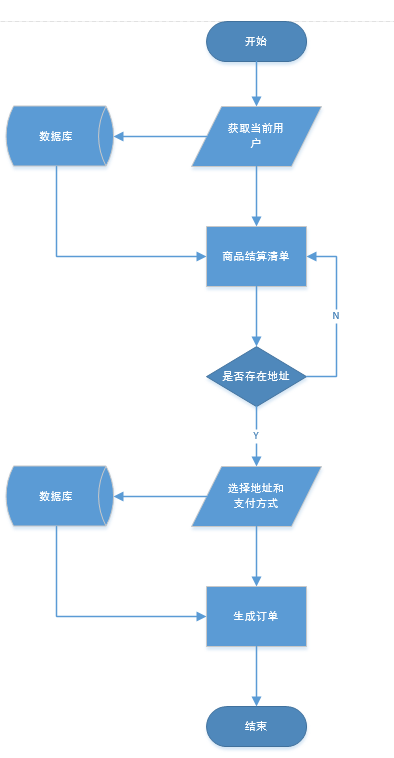


图4.6商品结算功能模块的流程图

* + 1. 订单管理功能模块

订单管理功能模块分为全部订单，通过筛选可以分为已发货、待发货、待收货，当点击确认收货的时候，订单会从待收货转成已收货。

订单管理功能模块的流程图如下:

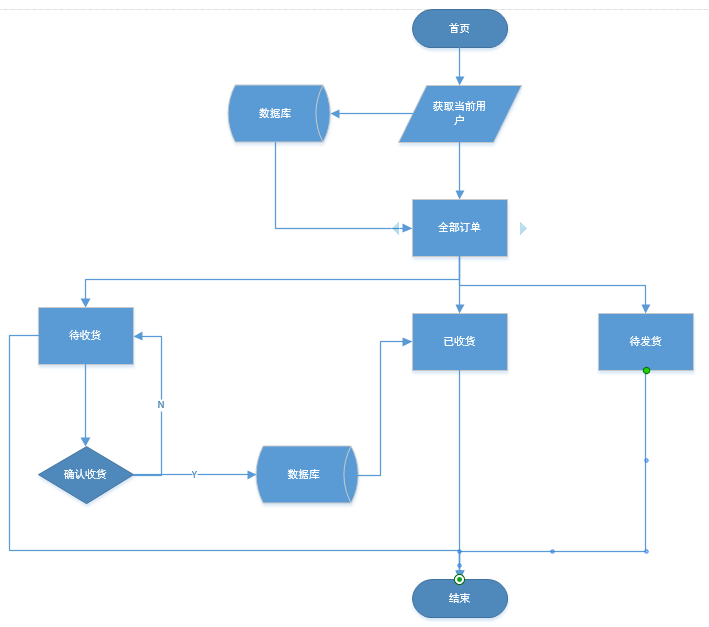


图4.7订单管理功能模块的流程图

* + 1. 地址管理功能模块

地址管理功能模块可以对地址进行增删改查，也可以设为默认地址，当地址条目不存在，是无法设置默认地址的。

地址管理功能模块的流程图如下:

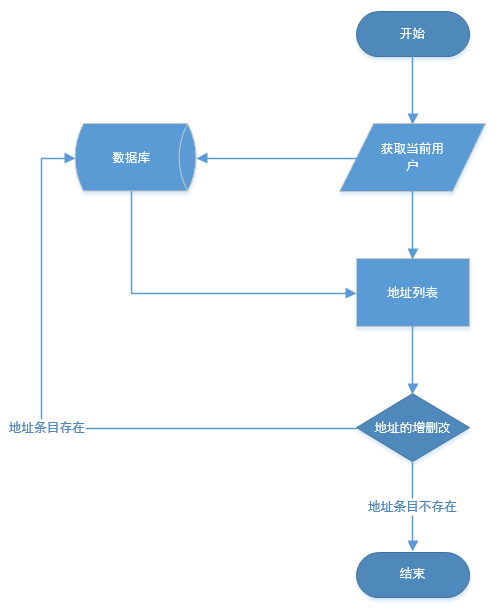


图4.8地址管理功能模块流程图

* 1. 后台系统设计
     1. 登录功能模块

管理员登录的时候必须输入账户和密码，前端组件进行校验如果校验成功，则会把账户密码发送给后端，后端会去数据库进行信息匹配，如果匹配成功则登录成功。

登录功能模块的流程图如下:

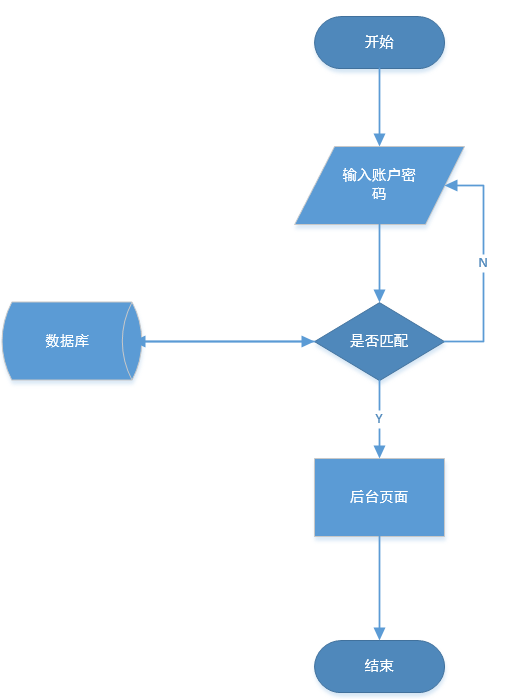


图4.9登录功能模块的流程图

* + 1. 商品管理功能模块

商品管理包括新建品牌、品牌列表、新建商品、商品列表、销量和库存。新建品牌以后会跳转到品牌列表，对品牌列表里的品牌进行编辑和删除；新建商品包括基本信息、参数配置、详细配置，新建商品后会跳转到商品列表，对商品列表的商品进行编辑和删除；销量是各个品牌的销售量，库存是各个品牌的库存。

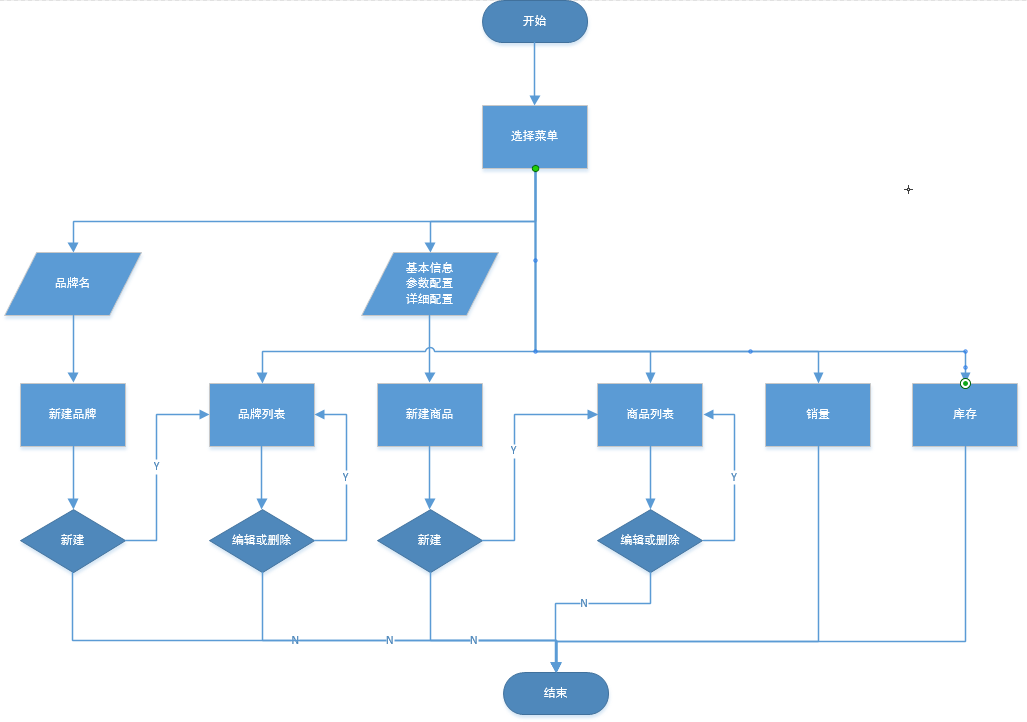


图4.10商品管理功能模块

* + 1. 运营管理功能模块

运营管理包括新建广告位和广告位列表，也就是前台的轮播图，新建广告位需要输入名称、广告位封面、所要链接的商品，新建广告位会跳转到广告位列表，对广告位列表的广告位进行编辑与删除。

运营管理功能模块的流程图如下:

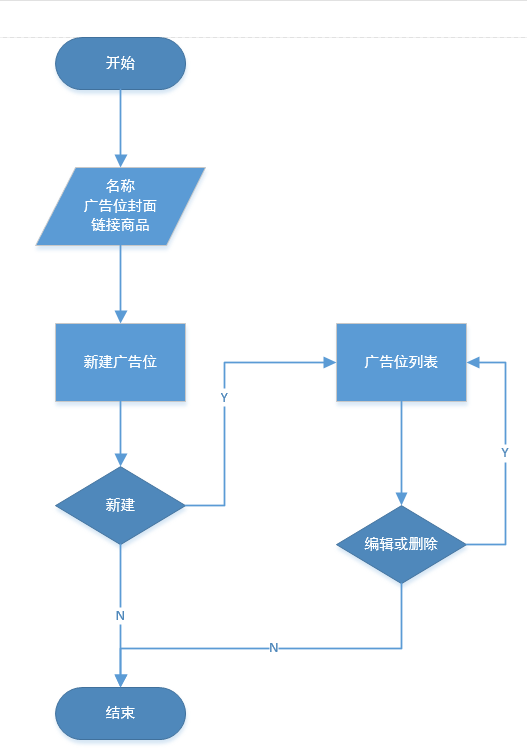


图4.11运营管理功能模块的流程图如下

* + 1. 用户管理功能模块

用户管理功能模块只有用户列表，只能查看用户的ID、用户名。

* + 1. 订单管理功能模块

订单列表可以看到订单编号、支付方式、数量、总价、及其状态。状态可分为待发货、已发货、已收货，当订单处于待发货时，点击发货可转为已发货。

订单管理功能模块的流程图如下:

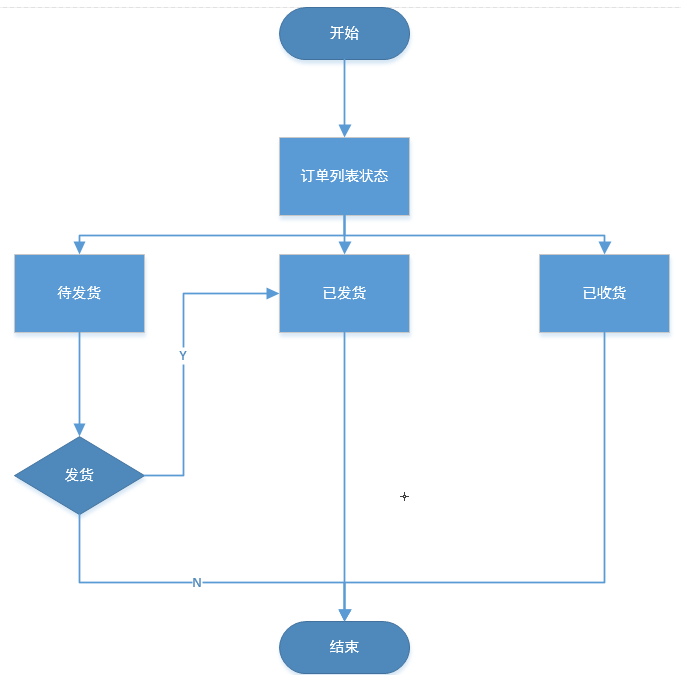


图4.12订单管理功能模块流程图

* + 1. 系统管理功能模块

系统管理功能模块包括新建管理员和管理员列表。新建管理员通过输入账户密码保存以后可以跳转到管理员列表，可以对管理员列表里的条目进行编辑与删除。

系统管理功能模块的流程图如下:

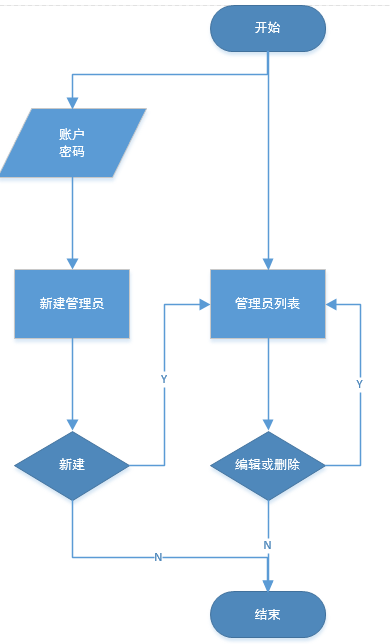


图 4.13系统管理功能模块的流程图

* 1. 关系型数据库设计
     1. 概念结构设计
        1. 局部E-R图

1. 首页广告位信息实体集E-R图

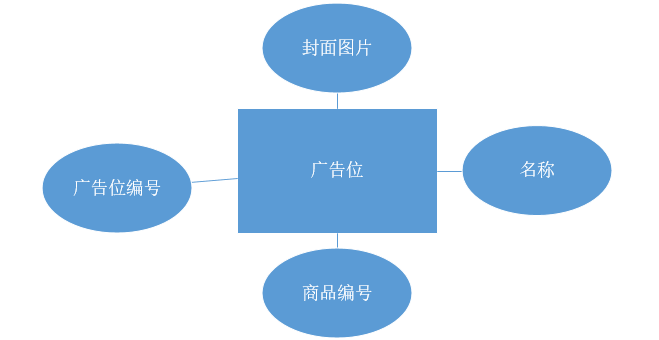


图 4.14首页广告位信息实体集E-R图

1. 品牌信息实体集E-R图

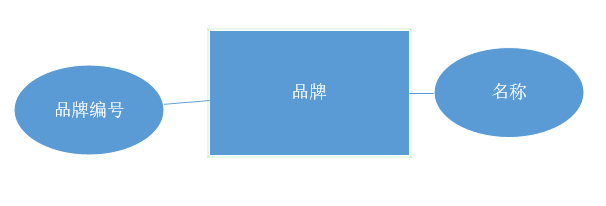


图4.15品牌信息实体集E-R图

1. 管理员信息实体集E-R图

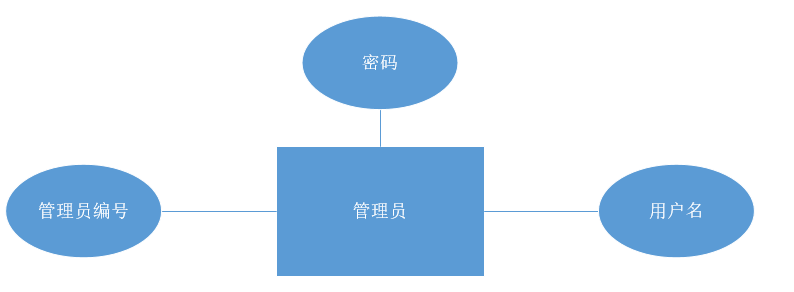


图4.16管理员信息实体集E-R图

1. 用户信息实体集E-R图

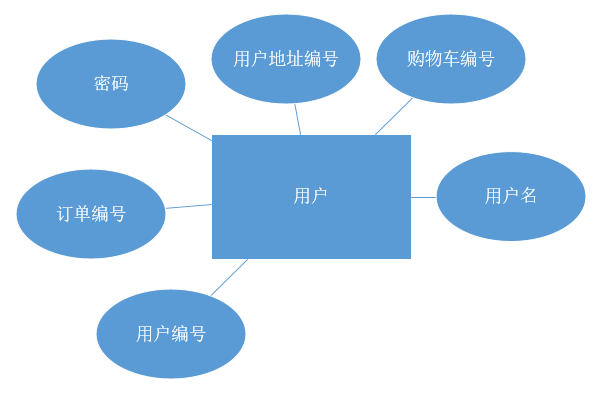


图4.17用户信息实体集E-R图

1. 地址信息实体集E-R图

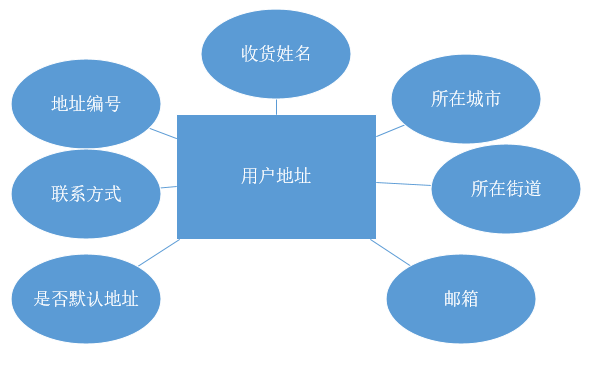


图4.18地址信息实体集E-R图

1. 购物车信息实体集E-R图

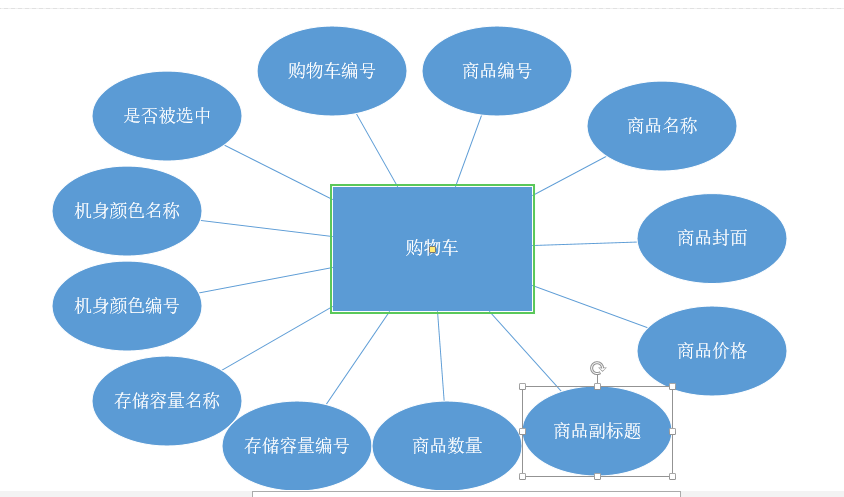


图4.19购物车信息实体集E-R图

1. 订单信息实体集E-R图

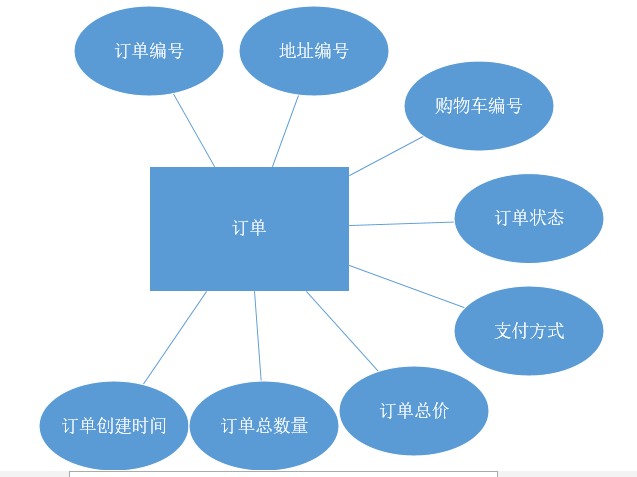


图4.20订单信息实体集E-R图

1. 商品信息实体集E-R图

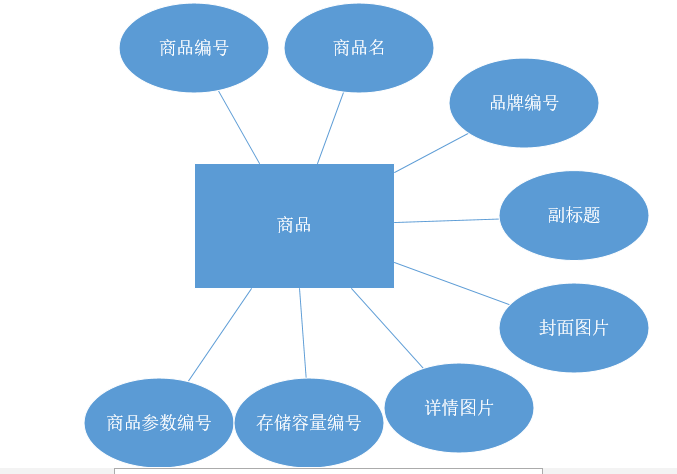


图4.21商品信息实体集E-R图

1. 存储容量信息实体集E-R图

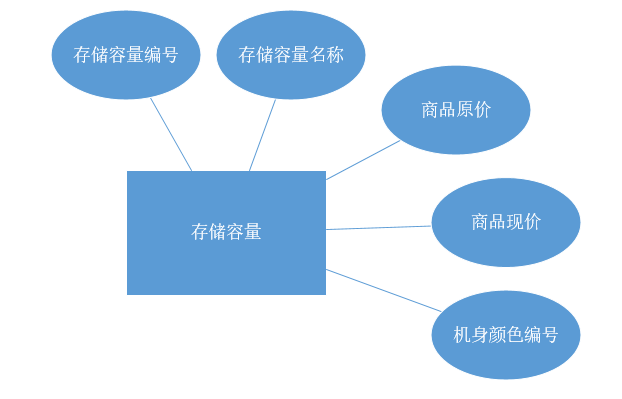


图4.22存储容量信息实体集E-R图

1. 机身颜色信息实体集E-R图

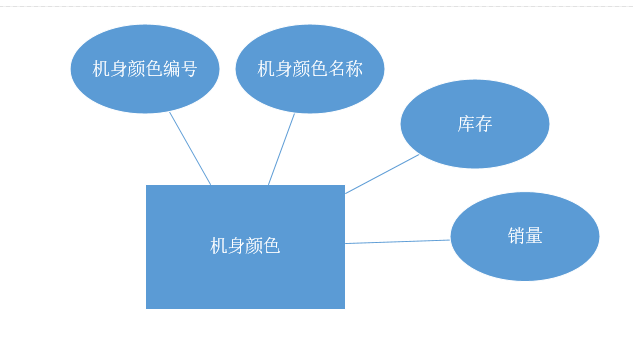


图4.23机身颜色信息实体集E-R图

1. 商品参数信息实体集E-R图

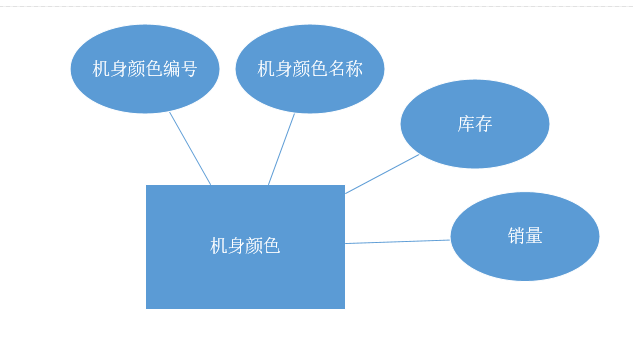


图4.24商品参数信息实体集E-R图

* + - 1. 全局E-R图

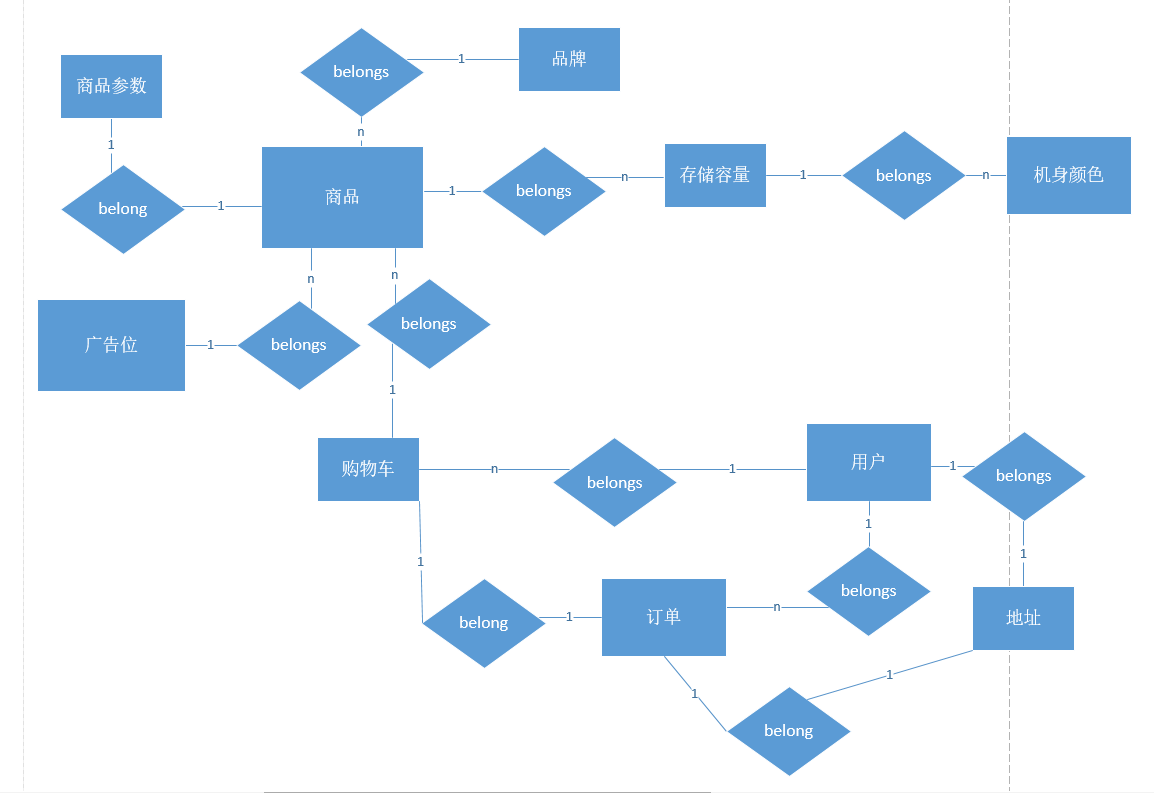


图4.25全局E-R

* + 1. 逻辑结构设计

1. 首页广告位表

表4.1 首页广告位表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空值 | 说明 | 主外键 |
| \_id | String | 否 | 默认编号 | 主键 |
| name | String | 否 | 名称 | 否 |
| coverImage | String | 否 | 封面图片 | 否 |
| porductId | String | 否 | 商品编号 | 外键 |

1. 品牌表

表4.2 品牌表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空值 | 说明 | 主外键 |
| \_id | String | 否 | 默认编号 | 主键 |
| name | String | 否 | 名称 | 否 |

1. 管理员表

表4.3 管理员表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空值 | 说明 | 主外键 |
| \_id | String | 否 | 默认编号 | 主键 |
| username | String | 否 | 用户名 | 否 |
| password | String | 否 | 密码 | 否 |

1. 用户表

表格 4.4 用户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空值 | 说明 | 主外键 |
| \_id | String | 否 | 默认编号 | 主键 |
| username | String | 否 | 用户名 | 否 |
| password | String | 否 | 密码 | 否 |
| addressId | String | 否 | 地址编号 | 外键 |
| cartId | String | 否 | 购物车编号 | 外键 |
| orderId | String | 否 | 订单编号 | 外键 |

1. 地址表

表4.5 地址表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空值 | 说明 | 主外键 |
| \_id | String | 否 | 默认编号 | 主键 |
| username | String | 否 | 收货姓名 | 否 |
| city | String | 否 | 所在城市 | 否 |
| streetName | String | 否 | 所在街道 | 否 |
| postCode | Number | 否 | 邮编 | 否 |
| tel | Number | 否 | 联系方式 | 否 |
| isDefault | Boolean | 否 | 是否为默认 | 否 |

1. 购物车表

表4.6 购物车表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空值 | 说明 | 主外键 |
| \_id | String | 否 | 默认编号 | 主键 |
| productId | String | 否 | 商品编号 | 外键 |
| productName | String | 否 | 商品名称 | 否 |
| productCoverIamge | String | 否 | 商品封面 | 否 |
| productPrice | Number | 否 | 商品价格 | 否 |
| productSubtitle | String | 否 | 商品副标题 | 否 |
| productNum | Number | 否 | 商品数量 | 否 |
| storageId | String | 否 | 存储容量编号 | 外键 |
| storageName | String | 否 | 存储容量名称 | 否 |
| colorId | String | 否 | 机身颜色编号 | 外键 |
| colorName | String | 否 | 机身颜色名称 | 否 |
| checked | Boolean | 否 | 是否被选中 | 否 |

1. 订单表

表4.7 订单表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空值 | 说明 | 主外键 |
| \_id | String | 否 | 默认编号 | 主键 |
| addressId | String | 否 | 地址编号 | 外键 |
| cartId | String | 否 | 购物车编号 | 外键 |
| status | Number | 否 | 状态（0是待发货，1是已发货/待收货，2是收货成功） | 否 |
| paymentMethod | Number | 否 | 付款方式（0是在线付款，1是货到付款） | 否 |
| totalPrice | Number | 否 | 商品总价 | 否 |
| totalNumber | Number | 否 | 商品总数量 | 否 |
| createTine | String | 否 | 创建时间 | 否 |

1. 商品表

表4.8 商品表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空值 | 说明 | 主外键 |
| \_id | String | 否 | 默认编号 | 主键 |
| name | String | 否 | 商品名 | 否 |
| brandId | String | 否 | 品牌编号 | 外键 |
| subtitle | String | 否 | 副标题 | 否 |
| coverIamge | String | 否 | 封面图片 | 否 |
| detailImage | String | 否 | 详情图片 | 否 |
| storageId | String | 否 | 存储容量编号 | 外键 |
| parameterId | String | 否 | 商品参数编号 | 外键 |

1. 商品参数表

表4.9 商品参数表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空值 | 说明 | 主外键 |
| \_id | String | 否 | 默认编号 | 主键 |
| socre | Number | 否 | 评分 | 否 |
| productionDate | String | 否 | 生产日期 | 否 |
| screenMaterial | String | 否 | 屏幕材质 | 否 |
| screenSize | String | 否 | 屏幕尺寸 | 否 |
| rosolution | String | 否 | 分辨率 | 否 |
| screenRatio | String | 否 | 屏幕比例 | 否 |
| cpuType | String | 否 | CPU型号 | 否 |
| ramSize | String | 否 | 内存大小 | 否 |
| ramType | String | 否 | 内存类型 | 否 |
| romSize | String | 否 | 闪存大小 | 否 |
| romType | String | 否 | 闪存类型 | 否 |
| batteryCapacity | String | 否 | 电池容量 | 否 |
| rearCamera | String | 否 | 后置摄像头 | 否 |
| frontCamera | String | 否 | 前置摄像头 | 否 |
| weight | String | 否 | 机身重量 | 否 |

1. 存储容量表

表4.10 存储容量表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空值 | 说明 | 主外键 |
| \_id | String | 否 | 默认编号 | 主键 |
| name | String | 否 | 存储容量名称 | 否 |
| oldPrice | Number | 否 | 商品原价 | 否 |
| nowPrice | Number | 否 | 商品现价 | 否 |
| colorId | String | 否 | 机身颜色编号 | 外键 |

1. 机身颜色表

表4.11 机身颜色表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空值 | 说明 | 主外键 |
| \_id | String | 否 | 默认编号 | 主键 |
| name | String | 否 | 机身颜色名称 | 否 |
| stock | Number | 否 | 库存 | 否 |
| salesVolume | Number | 否 | 销量 | 否 |

* 1. 文档型数据库设计

由于要减轻学习成本，所以选择MongoDB数据库，通过从关系型数据库转变成文档型数据库需要做出以下改变：

1. 若存在B实体，只作用于A实体中，无法与其他实体共享，则把B实体内嵌在A实体中。如：用户的地址列表是独一无二的，无法与其他用于共享，则地址列表内嵌在用户表中。
2. 若存在B实体，可以作用于多个实体中，则单独建表，通过id进行关联(类似于关系型数据库的主外键)，如：多个商品都属于一个品牌，则品牌单独建表，通过id与商品表进行关联。

根据以上规则，对关系型数据库表进行合并。

* + 1. 首页广告位模型

{

\_id: {type: ObjectId}//系统默认生成唯一id

name: {type: String},//名称

coverImage: { type: String },//封面

productId: { type: mongoose.SchemaTypes.ObjectId, ref: 'Product' },//关联商品的id

}

* + 1. 品牌模型

{

\_id: {type: ObjectId}//系统默认生成唯一id

name:{type:String}//名称

}

* + 1. 管理员模型

{

\_id: {type: ObjectId}//系统默认生成唯一id

username: { type: String },//用户名

password: {//密码

type: String,

select: false,

set(val) {

return bcrypt.hashSync(val, 10)

}

}

}

* + 1. 用户模型

{

\_id: {type: ObjectId}//系统默认生成唯一id

username: { type: String },//用户名

password: {//密码

type: String,

select: false,

set(val) {

return bcrypt.hashSync(val, 10)

}

},

addressList: [//地址列表

{

\_id: {type: ObjectId}//系统默认生成唯一id

username: { type: String },//收货姓名

city: { type: String },//所在城市

streetName: { type: String },//所在街道

postCode: { type: Number },//邮编

tel: { type: Number },//联系方式

isDefault: { type: Boolean }//是否为默认

}

],

cartList: [//购物车列表

{

\_id: {type: ObjectId}//系统默认生成唯一id

productId: { type: String },//商品id

productName: { type: String },//商品名称

productCoverImage: { type: String },//商品封面

productPrice: { type: Number },//商品价格

productSubtitle: { type: String },//商品副标题

productNum: { type: Number },//商品数量

productMaxNum: { type: Number },//商品最大最大数量

storageId: { type: String },//存储容量id

storageName: { type: String },//存储容量名称

colorId: { type: String },//机身颜色id

colorName: { type: String },//机身名称

checked: { type: Boolean },//是否被选中

}

],

orderList: [//订单列表

{

\_id: {type: ObjectId}//系统默认生成唯一id

address: { type: Object },//地址

cartList: { type: Array },//购物车列表

status: { type: Number }, //订单状态：0是待发货，1是已发货/待收货，2是收货成功

paymentMethod: { type: Number },//付款方式：0是在线付款，1是货到付款

totalPrice: { type: Number },//商品总价

totalNumber: { type: Number },//商品总数量

createTime:{type:String}//创建时间

},

{

timestamps: true//时间戳创建时间

}

]

}

* + 1. 商品模型

{

\_id: {type: ObjectId}//系统默认生成唯一id

name: { type: String },//商品名

brand: { type: mongoose.SchemaTypes.ObjectId, ref: 'Brand' },//关联品牌id

subtitle: { type: String },//副标题

coverImage: { type: String },//封面图片

detailImage: { type: String },//详情图片

storages: [//存储容量数组

{

\_id: {type: ObjectId}//系统默认生成唯一id

name: { type: String },//存储容量名称

oldPrice: { type: Number },//商品原价

nowPrice: { type: Number },//商品现价

colors: [//机身颜色素组

{

name: { type: String },//机身颜色名称

stock: { type: Number },//库存

salesVolume: { type: Number },//销量

}

]

}

],

parameter: {//商品参数

score: { type: Number },//评分

productionDate: { type: String },//生产日期

screenMaterial: { type: String },//屏幕材质

screenSize: { type: String },//屏幕尺寸

resolution: { type: String },//分辨率

screenRatio: { type: String },//屏幕比例

cpuType: { type: String },//cpu型号

ramSize: { type: String },//内存大小

ramType: { type: String },//内存类型

romSize: { type: String },//闪存大小

romType: { type: String },//闪存类型

batteryCapacity: { type: String },//电池容量

rearCamera: { type: String },//后置摄像头

frontCamera: { type: String },//前置摄像头

weight: { type: String }//机身重量

}

},

{

timestamps: true//时间戳创建时间

}

}

1. 详细设计
   1. 前台系统详细设计

前台系统主要分为注册/登录功能模块、首页功能模块、商品详情功能模块、购物车管理功能模块、商品结算功能模块、订单管理功能模块、地址管理功能模块。当单纯的用于页面显示的话就是用GET请求，例如轮播图的轮播；与后端频繁交互的话，如增删改，则发起POST/DELETE/PUT请求，出于效率考量，处理数据大都是用通用CRUD。

* + 1. 注册/登录功能模块
       1. 注册功能

点击首页的免费注册按钮，调转到注册界面，输入用户名、密码、确认密码；当输入用户名时，后端会去数据库匹配，如果用户名匹配成功，则会提示用户名存在；若三个input标签都不输入，则会有相关提示，提示用户必须输入才能完成注册，注册成功后会跳转到登录页面。

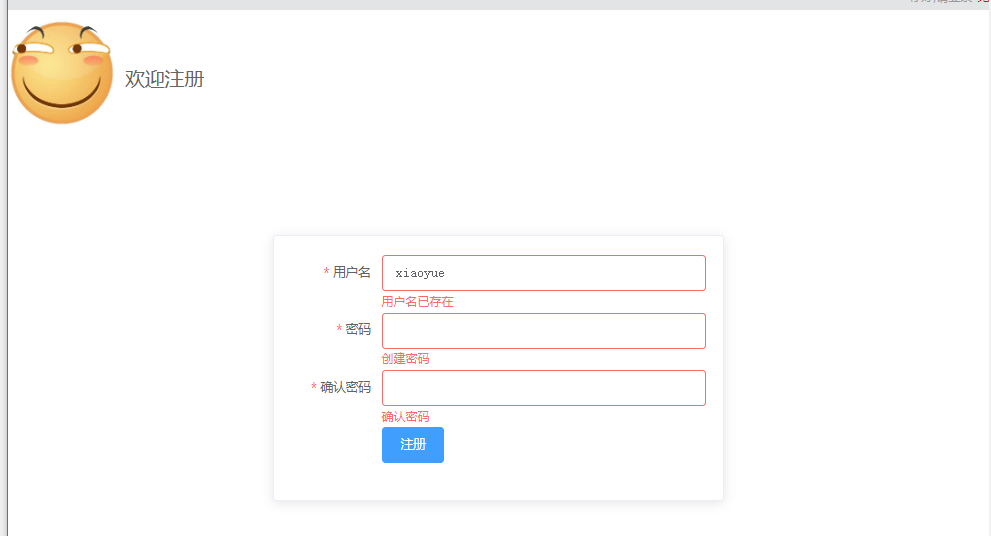


图5.1注册功能

注册功能部分代码如下:

register() {

//验证整张表是否通过

this.$refs["ruleForm"].validate(async valid => {

if (valid) {//如果表单验证不通过则无法注册

let model = {

username: this.ruleForm.name,

password: this.ruleForm.pwd

};

//发起注册请求

await this.$http.post("rest/web\_users", model);

this.$router.push("/login");//跳转到注册页面

this.$message({

type: "success",

message: "注册成功"

});

}

});

},

//去数据库找用户名，判断是否存在

async findUser() {

const res = await this.$http.get("rest/web\_users");

this.usernames = res.data.map(user => {

return user.username;

});

}

* + - 1. 登录功能

点击首页登录按钮跳转到登录页面，当用户无任何输入的时候，会有表单校验，并弹出相关提示，当输入账户密码通过校验，点击登录，则会提交表单到后端写入数据库中，登录成功后则跳转到首页。

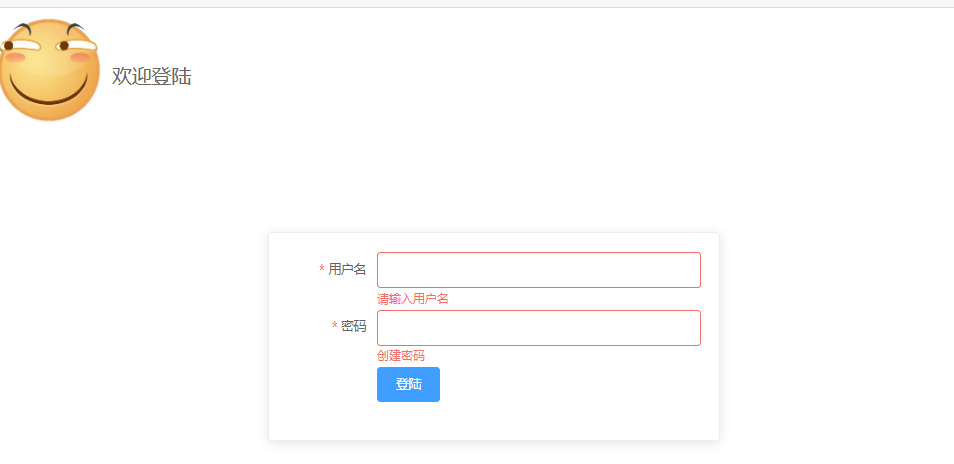


图5.2登录功能

登录功能部分代码如下：

login() {

//验证整张表是否通过

this.$refs["ruleForm"].validate(async valid => {

if (valid) {

let model = {

username: this.ruleForm.name,

password: this.ruleForm.pwd

};

//发起登录请求

const res = await this.$http.post("login", model);

localStorage.token = res.data.token;

this.$router.push("/");

this.$message({

type: "success",

message: "登陆成功"

});

}

});

后端部分代码如如下：

//登陆接口

app.post('/web/api/login', async (req, res) => {

const { username, password } = req.body

const user = await WebUser.findOne({ username }).select('+password')

assert(user, 422, '用户名不存在')

//校验密码

const isValid = require('bcryptjs').compareSync(password, user.password)

//如果密码不对

assert(isValid, 422, '密码错误')

//返回token

const jwt = require('jsonwebtoken')

const token = jwt.sign({ id: user.\_id }, app.get('secret'))

res.send({ token })

})

* + 1. 首页功能模块

首页主要分为轮播图、全部品牌、商品搜索，轮播图与全部品牌都是用的GET请求，然后把请求的数据渲染到页面上。

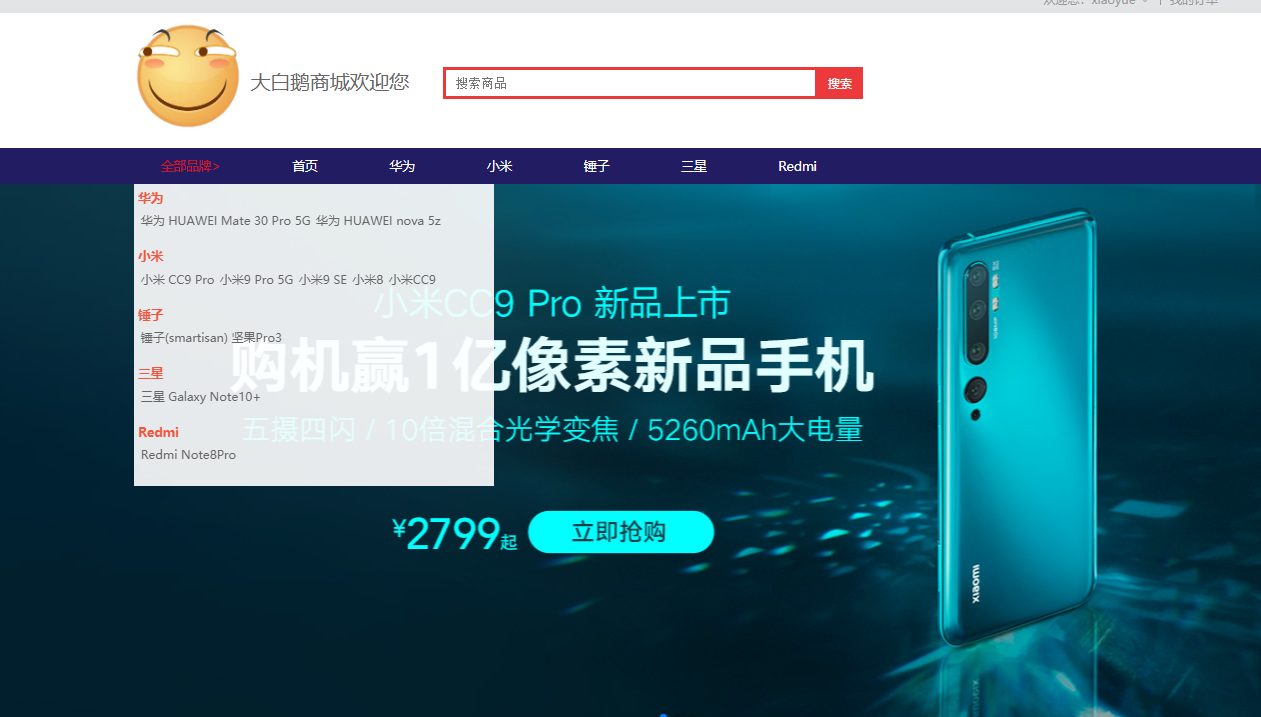


图5.3首页图

轮播图的部分代码如下:

async \_fetchAds() {

const res = await this.$http.get("rest/ads");

this.ads = res.data;

}

品牌分类的部分代码如下:

async \_fetchBrands() {

//获取品牌数据

const res = await this.$http.get("brandList");

this.brandList = res.data;

}

品牌分类的后端代码如下：

// 品牌列表

app.use('/web/api/brandList', async (req, res) => {

//聚合函数

const brands = await Brand.aggregate([

{

$lookup: {

from: 'products',

localField: '\_id',

foreignField: 'brand',

as: 'productList'

}

}

])

brands.map(brand => {

brand.productList.map(product => {

product.brandName = brand.name

})

})

res.send(brands)

})

* + - 1. 商品搜索

1. 当输入关键字时候，前端实时把关键字传到后端，后端根据正则表达式，把数据库中匹配成功的数据返回前端。



图5.4商品搜索一

商品搜索一代码部分如下:

async input() {

//如果输入为空，则返回

if (!this.query.trim()) {

this.resultList = [];

return;

}

const res = await this.$http.post("keyword", {

query: this.query

});

this.resultList = res.data;

},

当