南京工業職業技術大學
NANJING VOCATIONAL UNIVERSITY OF INDUSTRY TECHNOLOGY

毕业论文（设计）

题　目　基于 Vue+Element UI+Node.js 的电商后台管理系统

学生姓名：林伟杰

学　　号：210501100323

学　　院：计算机与软件学院

专　　业：软件工程

班　　级：软件 2136

指导教师：刘立阳

二〇二三年 4 月

南京工业职业技术大学

毕业论文（设计）诚信承诺书

本人郑重声明：所呈交的毕业论文（设计）是本人在指导教师的指导下独立开展工作所取得的成果。尽我所知，除了文中特别加以标注和致谢的内容外，本设计（论文）不包含任何其他个人或集体已经发表或撰写的成果作品。对本设计（论文）所涉及的研究工作作出贡献的其他个人和集体，均已在文中以明确方式标明。

作者签名：　　（手签）

日　　期：　20　　年　　月　　日

（此页底部已经设置分节符（奇数页），请勿删除分节符）

摘　　要

本次设计的研究目的是为了探究基于 Vue、Node.js、Element UI 的电商后台管理系统的开发流程与实现方式。电商后台管理系统是一个功能强大的平台，它能够帮助企业高效地管理商品、订单等信息，提高企业的管理效率和服务水平。

在需求分析的过程中，我们对电商后台管理系统的各个模块进行了详细的讨论和分析，针对不同的模块实现了不同的功能和操作方式，如商品管理模块实现了商品的添加、删除、修改和查询，订单模块实现了订单的管理,查询等，权限模块实现了角色信息的修改和权限的分配等等。

在系统架构设计上，我们采用了前后端分离的架构，使用了 Node.js 作为后端 API 接口，再通过 Vue 实现前端页面展示，并且采用了 Element UI 的组件库，进行了美化和布局设计，在系统的设计和开发中，采用了前后端分离的架构设计，使得系统具有更好的可维护性和扩展性。总之，本次设计成功实现了一套基于 Vue、Node.js 和 Element UI 的电商后台管理系统，不仅提供了强大的功能支持，也为企业的电子商务管理提供了有力的保障。

关键词：Vue、Node.js、Element UI、电商后台管理系统、需求分析、系统架构设计、前后端分离

ABSTRACT

The purpose of this design is to explore the development process and implementation of e-commerce background management system based on Vue, Node.js and Element UI. The e-commerce background management system is a powerful platform that can help enterprises efficiently manage goods, orders and other information, and improve the management efficiency and service level of enterprises.

In the process of demand analysis, we have discussed and analyzed each module of the e-commerce background management system in detail, and implemented different functions and operation methods for different modules, such as the commodity management module to realize the addition, deletion, modification and query of goods, the order module to realize the management of orders, query, etc., and the permission module to realize Modification of role information, assignment of permissions, etc.

In terms of system architecture design, we adopt the architecture of separation of front and back end, use Node .js as the backend API interface, and then realize front-end page display through Vue, and adopt the component library of Element UI, beautification and layout design, in the design and development of the system, adopt the architecture design of front-end and back-end separation, so that the system has better maintainability and scalability 。 In short, this design has successfully realized a set of e-commerce background management system based on Vue, Node.js and Element UI, which not only provides strong functional support, but also provides a strong guarantee for the e-commerce management of enterprises.

KEY WORDS: Vue, Node.js, Element UI, e-commerce background management system, demand analysis, system architecture design, front-end and back-end separation

目录

第一章 绪论

1.1 课题背景及研究意义

随着互联网技术的不断发展和普及，电子商务在全球范围内蓬勃发展，已成为经济增长的重要引擎，电商后台管理系统作为一个信息管理平台，它能够帮助企业高效地管理商品、订单、客户等信息，提高企业的管理效率和服务质量。由于市场竞争的加剧，电商后台管理系统也变得越来越重要，不仅影响了企业的经营模式和盈利模式，也关系到企业的品牌形象和口碑效应。因此，对于企业而言，如何设计和开发高效、稳定、易用的电商后台管理系统就成为了一项重大的课题。

本次设计以 Vue、Node.js 和 Element UI 等技术为基础，构建一套功能强大、操作简单、稳定可靠的电商后台管理系统。通过对需求分析、系统架构设计、开发流程、测试和优化等环节的深入研究和实践，本次设计不仅实现了电商后台管理系统的全面开发和演示，也为企业相关领域的学术研究和实际应用提供了有力的支持和参考。

总之，本次设计的背景和意义在于，探究如何提升电商后台管理系统的开发效率和管理水平，为企业的信息化建设提供有力保障。

1.2 国内外研究现状

目前国内外对于电商后台管理系统的研究，主要可以从以下几个方面来展开：

系统架构设计和技术选型：在电商后台管理系统中，选择适合自身需求的技术栈和架构设计，对于确保系统的稳定性和高效性至关重要。国内外研究者通过对不同的技术栈进行比较和分析，评估不同技术对于电商后台管理系统架构和性能的影响，例如前端框架 Vue、React 等，或者后端框架 Node.js 等的应用。

用户行为分析：研究者通过大量的实证研究和数据分析，探究消费者在电商平台上的行为特征、偏好和决策过程，为企业提供有力的市场营销和产品策略依据。基于人工智能和大数据技术的用户行为分析方法也得到了广泛的应用和研究。

功能优化和创新：近年来，随着电子商务的飞速发展，对于电商后台管理系统的各项功能进行深入优化或者增加新功能，已成为一个热点和难点。如推荐算法、客户服务机器人等等，使得整个系统更加智能化、用户友好、高效稳定。例如，基于语义分析的自然语言处理技术推出的智能客服，能够有效帮助企业降低人力成本，提高服务效率。

安全问题：由于电商平台涉及到大量的交易信息和用户隐私，安全问题已成为一个关键研究点。国内外研究者通过对电商平台进行安全测试和风险评估，提出了一系列的安全策略和方案，有效保障了电商平台的安全性。

总之，从国内外的研究现状来看，电商后台管理系统已经成为研究领域中一个重要的话题，不仅涉及到技术、市场、用户等多个方面，而且具有广泛的应用前景和研究价值。

1.3 开发工具介绍

（1） VUE

Vue.js 是一款流行的 JavaScript 前端框架，具有轻量、高效、易用的特点，广泛应用于构建单页面应用（SPA）以及大型 Web 应用程序。以下是 Vue.js 的技术介绍：

响应式数据绑定：Vue.js 的核心部分是响应式数据绑定，这意味着当数据发生变化时，视图会自动更新。此功能通过使用 Vue.js 的数据监听器和指令来实现。

组件化开发：Vue.js 支持组件化开发，可以将一个应用程序分解为若干个独立的、可复用的组件。每个组件都有自己的数据和方法，并且可以嵌套在其他组件中。

虚拟 DOM：Vue.js 采用虚拟 DOM 技术，通过将数据变更抽象成 DOM 操作，然后再进行一次比较来减少 DOM 操作的次数，从而提高应用程序的性能。

模板语法：Vue.js 的模板语法类似于 HTML，但是增加了许多指令和过滤器，从而使开发者能够更便捷地绑定数据和进行条件渲染等操作。

生态系统：Vue.js 拥有一个庞大的生态系统，包括第三方插件、工具和模块等，可以大幅提高开发效率和代码质量。

Vue.js 是一种极具灵活性和高效性的 JavaScript 前端框架，支持响应式数据绑定、组件化开发、虚拟 DOM、模板语法等特性。同时，Vue.js 拥有庞大的生态系统，并且不断更新版本和发展新特性，使得其具备较高的适应性和可扩展性，广泛应用于各类 Web 应用程序的开发中。

（2） Node.js

Node.js 是一款基于 Chrome V8 引擎的 JavaScript 运行时环境，通过使用事件驱动、非阻塞 I/O 模型，使 JavaScript 语言在后端实现服务器端编程。以下是 Node.js 的技术介绍：

事件驱动：Node.js 采用事件驱动的编程模型，代码通过注册事件回调函数来响应事件的发生和处理。这种模型使得程序较为简单，事件处理器之间的松散耦合也提高了应用程序的可扩展性。

非阻塞 I/O 模型：Node.js 中的 I/O 操作采用异步方式进行，不会阻塞主线程，从而使得系统全部或部分的 I/O 操作能够并发执行。这种模型的实现基于 Node.js 的事件循环机制，大幅提高了应用程序的处理能力和效率。

单线程：Node.js 采用单线程模型，所有的 I/O 请求都是异步地处理的，从而避免了多线程之间共享数据和同步的问题。此外，单线程的设计使得开发者可以更加简单地实现代码，也减少了系统出错的可能性。

轻量级：Node.js 是一款轻量级的运行时环境，其核心代码只占几百 KB，且没有任何依赖关系，因此具有很好的可移植性和易安装性。

模块化：Node.js 采用了模块化的开发方式，可以使用 require()函数导入其他 JavaScript 模块，也可以通过 exports 关键字将某个函数或类公开给其他模块使用。

Node.js 是一种快速、轻量级、高效的 JavaScript 运行时环境，具有事件驱动、非阻塞 I/O 模型、单线程、模块化等特点。Node.js 广泛应用于构建各类服务端应用程序，如 Web 服务器、API 服务器、实时数据通信等，并成为后端 JavaScript 开发的一个重要工具和技术。

（3） Element UI

Element UI 是一款基于 Vue.js 框架的 UI 库，提供丰富的组件和基础样式，方便开发者快速搭建美观、易用的 Web 界面。以下是 Element UI 的技术介绍：

基于 Vue.js：Element UI 是基于 Vue.js 框架构建的，具有 Vue.js 的响应式数据绑定、虚拟 DOM 等特性，使得组件可以动态更新，提高了应用程序的交互性和效率。

组件化：Element UI 采用组件化的开发方式，每个组件都是独立的模块，并且拥有自己的样式和行为。这种方式可以让开发者更加灵活地构建页面，提高了应用程序的可复用性和可维护性。

省时省力：Element UI 提供丰富的基础组件和样式，如按钮、表单、表格、布局等，开发者可以直接使用，减少了开发时间和成本。

外观统一：Element UI 的组件风格和样式都经过设计师的精心设计和把控，可以保证整个应用程序的外观统一性，提高了应用程序的用户体验。

可定制化：Element UI 提供了丰富的主题定制功能，可以根据项目需求和用户喜好自定义主题样式，从而实现个性化的 Web 界面。

Element UI 是一款基于 Vue.js 框架的轻量级 UI 库，具有组件化开发、响应式数据绑定、省时省力、外观统一、可定制化等特点。Element UI 在实际应用中可以大幅提高前端开发效率和代码质量，成为很多开发者喜爱的前端 UI 库之一。

（4） MySql

MySQL是一款开源的关系型数据库管理系统，被广泛应用于Web应用程序的开发中。以下是 MySQL 的技术介绍：

关系型：MySQL 是一款关系型数据库管理系统，数据以表格的形式进行组织和存储，这种方式使得数据之间具有明确的关系，从而方便了数据的查询和处理。

跨平台：MySQL 支持跨多个操作系统平台使用，包括 Windows、Linux、MacOS 等，且非常稳定可靠。

数据安全性：MySQL 提供强大的数据安全功能，如用户身份认证、数据加密、访问控制等，可以保证数据在传输和存储过程中的安全可靠。

可扩展性：MySQL 支持高度可扩展性，可以通过添加更多服务器或更改硬件配置来满足应用程序的扩展需求。

性能优化：MySQL 提供了多种性能优化机制，如索引优化、查询缓存、语句优化等，可以提高数据库系统的查询效率和响应速度。

MySQL 是一款功能强大的开源关系型数据库管理系统，具有跨平台、数据安全性、可扩展性和性能优化等特点。MySQL 在 Web 应用程序的开发中广泛应用，包括电子商务、社交网站、金融和医疗应用程序等。MySQL 的开源性和社区支持也使得其得到了广泛的开发者和用户的支持和认可。

（5） Vue-Router

Vue-router 是 Vue.js 官方提供的路由管理插件，用于实现前端页面的切换和跳转，并支持 RESTful 风格的 URL。以下是 Vue-router 的技术介绍：

基于 Vue.js：Vue-router 是基于 Vue.js 框架构建的路由管理插件，充分利用了 Vue.js 的响应式数据绑定、虚拟 DOM 等特性，实现页面跳转和数据传递。

组件化：Vue-router 采用组件化的开发方式，每个组件都可以通过 vue-router 的路由规则配置成一个独立的子页面，从而实现界面的动态切换和组合。

路由管理：Vue-router 可以帮助开发者管理页面的路由规则，如设置路由参数、查询参数、重定向、嵌套路由等，以及支持 history 模式和 hash 模式的路由。

导航守卫：Vue-router 的导航守卫机制可以让开发者在路由改变前或后进行拦截和处理，比如权限认证、用户登录等。

统一化 API：Vue-router 的 API 设计与 Vue.js 保持一致，统一简洁易懂，方便开发者快速上手和使用。

Vue-router 是一款功能强大的 Vue.js 路由管理插件，具有基于 Vue.js、组件化、路由管理、导航守卫、统一化 API 等特点。Vue-router 广泛应用于 Vue.js 开发中，可以实现复杂的单页面应用程序、多页面应用程序等，让前端开发更加简单、高效和灵活。

（6） Axios

Axios 是一款能够发送 HTTP 请求的 JavaScript 库，可用于在浏览器和 Node.js 中异步获取数据。它使用 Promise 机制来处理异步请求，可以让开发者更加方便、高效地编写代码。同时 Axios 支持拦截器、错误处理等特性，能够帮助开发者更好地处理数据请求和响应过程中的异常情况。Axios API 设计简洁易懂，并且能够无缝适配多种环境，开发者可以轻松上手使用。

（7） Echarts

Echarts 是一款基于 JavaScript 的开源图表库，可以用于在 Web 应用程序中创建各种样式丰富、交互性强的图表。Echarts 提供了多种类型的图表，如折线图、柱状图、散点图、饼图、地图等，并支持自定义配置和样式。Echarts 能够将复杂的数据以可视化的方式呈现出来，为用户分析、观察数据提供方便。Echarts 还具有高性能的渲染能力和流畅的交互体验，支持集成到各种框架中使用。总之，Echarts 是一个强大易用的图表库，被广泛应用于各种领域，如金融、物流、电商等，为用户提供了优秀的数据可视化工具。

第二章 电商后台管理系统需求分析

2.1 系统可行性分析

2.1.1 技术可行性

基于 Vue.js + Node.js + Element UI 的电商后台管理系统开发技术可行性非常高，因为 Vue.js 适合开发单页应用，具有高效率和易上手的特点；Node.js 拥有卓越的并发性能和高吞吐率，可以快速构建高性能的网络应用程序；Element UI 则提供了丰富的 UI 组件和交互式操作，并有着广泛的社区支持。这三个技术组合在电商后台管理系统的开发中可以满足稳定性、安全性、易用性、扩展性等多重需求，所以技术可行性非常高。

2.1.2 市场可行性

当考虑电商后台管理系统的市场可行性时，需要综合考虑目标市场、竞争对手、市场趋势等因素。通过市场调研和竞争分析，可以找出产品的差异化点和优势，制定明确的市场定位和营销策略。渠道拓展、用户反馈和不断改进产品也是提高市场可行性的关键因素。在开发电商后台管理系统之前，我们需要将市场可行性纳入到考虑范围中，制定切实可行的市场策略，优化产品功能和用户体验，提升用户使用的满意度和口碑，从而增强系统在市场中的竞争力和持续性。

2.1.3 操作可行性

确定需求与目标：在开始开发之前，要明确电商后台管理系统的需求和目标，从而为开发工作提供方向。

选择合适的技术栈：根据自己的实际情况和所掌握的技能，选择适合自己开发的技术栈。如果缺乏某些技能，可以通过学习或找到一些开源代码进行参考，以提升开发效率。

定制化开发：不同的电商后台管理系统可能拥有不同的需求和功能，因此需要根据自己的业务需求和用户需求进行定制化开发，尽可能减少代码复杂度和多余功能。

迭代开发：采用迭代式开发模式，每个迭代周期内打包完成一个小目标，并进行测试确认后再进行下一次开发。

2.2 系统需求分析

2.2.1 角色识别

为了实现角色识别并确保系统资源的安全性和稳定性，本项目采用了权限验证模块对服务请求进行拦截和角色权限验证。通过这种设计策略，我们能够有效区分不同角色的用户并控制他们对特定功能和数据的访问，经过分析可得，本系统分为五个角色，系统管理员，商品管理员，订单管理员，用户管理员，数据分析师。

2.2.2 系统管理员需求

系统管理员作为最高权限角色，负责整个系统的管理和维护，如图 2-1 具备以下权限：

1） 商品管理：包括商品列表、添加商品、修改商品、删除商品、更新商品图片、更新商品属性、更新商品状态以及获取商品详情。

2） 分类参数：包括获取参数列表、创建商品参数、删除商品参数。

3） 商品分类：包括添加分类、删除分类、编辑分类。

4） 订单管理：包括订单列表、修改订单信息、查看订单物流详情。

5） 权限管理：包括角色列表、添加角色、删除角色、角色授权、取消角色授权、获取角色列表、获取角色详情、更新角色信息以及更新角色权限。

6） 权限列表：包括查看权限。

7） 用户管理：包括用户列表、添加用户、删除用户、更新用户、获取用户详情、分配用户角色以及设置管理状态。

8） 数据统计：包括数据报表、查看数据。

如图 2-1 系统管理员

2.2.3 商品管理员需求

商品管理员在本项目中的权限主要集中在商品管理方面，负责商品的发布、管理、分类和库存等相关工作如图 2-2，具体包括以下几个方面：

1） 商品管理：负责商品列表的查看、添加商品、修改商品、删除商品、更新商品图片、更新商品属性、更新商品状态以及获取商品详情。

2） 分类参数：负责获取参数列表、创建商品参数、删除商品参数。

3） 商品分类：负责添加分类、删除分类、获取分类详情。

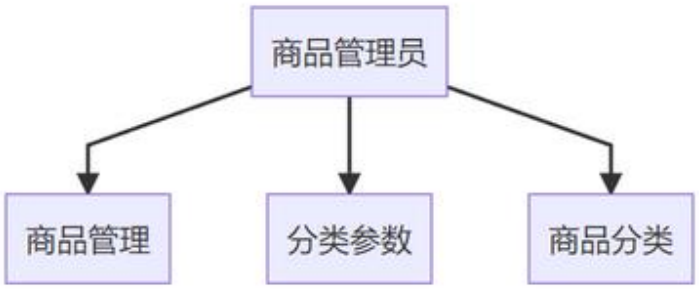

图 2-2 商品管理员

2.2.4 订单管理员需求

订单管理员在本项目中的权限主要关注订单管理方面：

1） 订单管理：负责查看订单列表、添加订单、更新订单以及获取订单详情。

2.2.5 用户管理员需求

用户管理员在本项目中的权限主要涉及用户及权限管理方面，具体如下：

1） 权限管理：负责查看角色列表、添加角色、删除角色、角色授权、取消角色授权、获取角色列表、获取角色详情、更新角色信息以及更新角色权限。

2） 权限列表：负责查看权限。

3） 用户管理：负责查看用户列表、添加用户、删除用户、更新用户、获取用户详情、分配用户角色以及设置管理状态。

2.2.6 数据分析师需求

数据分析师在本项目中的职责主要集中在数据统计、分析与可视化方面，具体内容如下：

1）数据统计：负责收集和整理各项数据，为分析和可视化做准备。

2）数据报表：负责生成数据报表，呈现各项关键数据。

3）查看数据：负责查看并分析数据，以便从中发现潜在的规律和趋势。

第三章 系统设计

3.1 总体架构

本系统是一个基于 Vue+Element UI+Node.js 的电商后台管理系统。系统的总体架构分为前端和后端两部分如图 3-1。前端使用 Vue.js 作为主要的 JavaScript 框架，搭配 Element UI 组件库构建用户界面。后端使用 Node.js 实现，主要负责处理 API 请求、连接数据库以及处理业务逻辑。

前端部分采用模块化开发，实现了登录、用户管理、权限管理、商品管理、订单管理和数据统计等功能模块。通过与后端接口的交互，实现数据的增删改查和权限控制等操作。Vue.js 和 Element UI 的结合使得系统具有良好的用户体验和易用性。

后端部分使用 Node.js 构建，为前端提供数据接口。后端负责处理前端发送的请求，实现用户认证、数据库操作以及业务逻辑处理等功能。同时，后端还需关注系统的性能、安全性和可扩展性。

系统采用模块化、分层的设计，将前端与后端解耦，有利于实现功能的快速开发、测试和维护。整体架构使得系统具有较高的灵活性和可扩展性，能够满足电商平台对后台管理的需求。

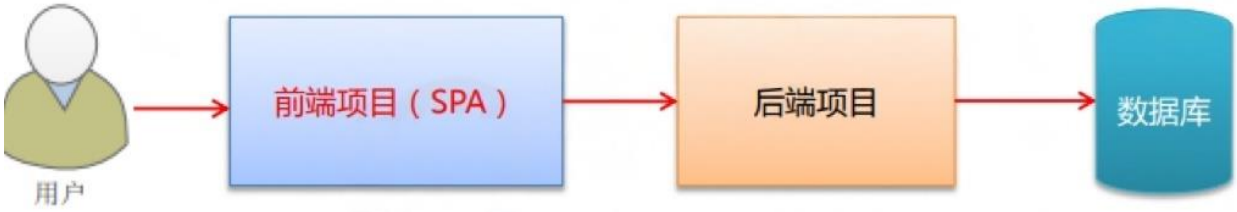

图 3-1 项目总体架构图

3.2 系统功能介绍

电商后台管理系统是电商平台的重要组成部分，其主要功能包括登录、用户管理、权限管理、商品管理、订单管理和数据统计。在这些模块中，每个模块都有其自身独特的需求和特点如图 3-2。

1.登录模块

登录模块是系统的入口，其主要任务是验证用户的身份信息并授权用户访问相应资源。其需求包括：

- 用户需要提供用户名和密码进行登录；
- 系统需要记录用户的登录日志，以便管理和监督。

2.用户管理模块

用户管理模块涉及到用户的添加、编辑、删除、搜索等功能，需要满足以下需求：

- 系统需要支持添加新用户及相关信息，例如用户名、手机号、邮箱等；
- 系统需要支持编辑用户信息；
- 系统需要支持搜索、分页显示用户列表；
- 系统需要支持对用户进行分配角色，以保证用户的权限与职责相对应；
- 系统需要提供更改用户状态的功能，以便管理员能够在必要时禁用或启用用户账号。

3.权限管理模块

权限管理模块的主要任务是为用户分配角色并授权。其需求包括：

- 系统需要支持添加新角色及相关信息，例如角色名称、描述等；
- 系统需要支持编辑角色信息；
- 系统需要支持删除角色以及删除相关权限；
- 系统需要支持对角色进行授权，以便用户能够使用相应功能；
- 系统需要提供权限分配和撤销的功能。

4.商品管理模块

商品管理模块是电商平台中的核心模块，其需求包括：

- 系统需要支持添加新商品及相关信息，例如商品名称、价格、描述、图片等；
- 系统需要支持编辑商品信息；
- 系统需要支持删除商品；
- 系统需要支持搜索商品、添加修改参数、添加编辑分类等功能。

5.订单管理模块

订单管理模块涉及到订单的搜索和编辑。其需求包括：

- 系统需要支持按照订单号、用户 ID 等条件搜索订单，并能实现分页显示；
- 系统需要支持编辑收货地址，查看物流情况等。

6.数据统计模块

数据统计模块负责分析和展示数据，并生成数据报表。其需求包括：

- 系统需要支持对不同维度的数据进行统计和分析，例如订单数量、交易额等；

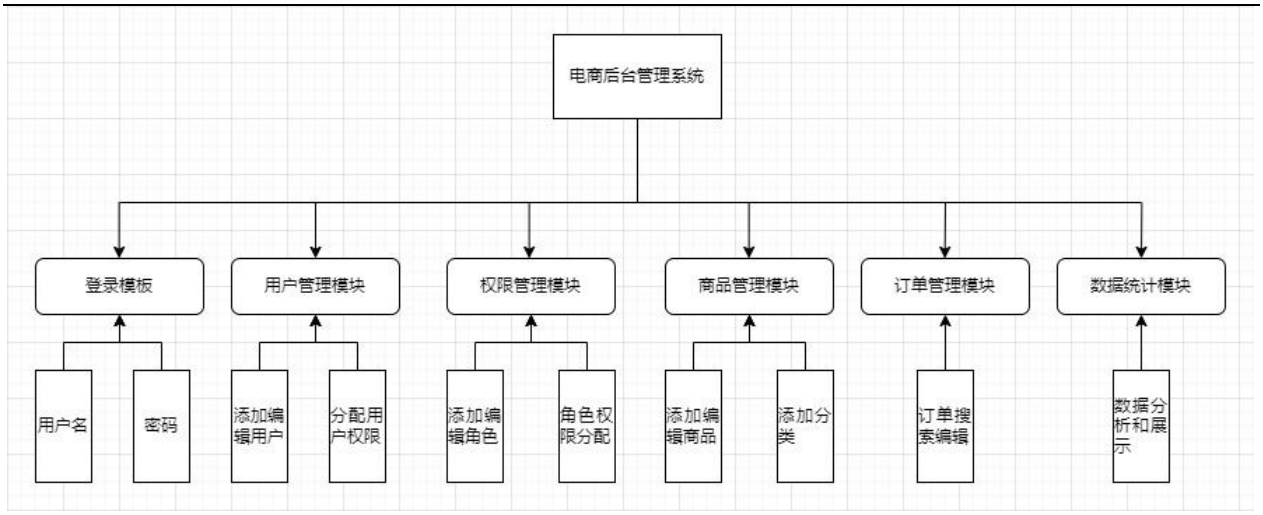

图 3-2 功能模块图

3.3 功能模块设计

3.3.1 登录功能设计

登录流程是电商后台系统中非常重要的一部分，用于确保用户身份的准确性并控制对系统的访问权限，以下是登录的流程如图 3-3：

1. 登录页面：用户在登录页面输入用户名和密码，这些信息将用于验证用户身份。
2. 调用后台接口：客户端将用户输入的用户名和密码打包成请求，发送到后端服务器进行验证。
3. 身份验证：后端服务器收到请求后，查询数据库以核实用户提供的凭据是否有效。如有效，服务器将生成一个访问令牌并返回给客户端。
4. 跳转至主页：客户端在收到访问令牌后，将其存储在本地，然后跳转到项目主页。后续请求中，客户端会将访问令牌附加到请求头，以证明用户已通过身份验证。
5. 保持登录状态：由于 HTTP 是无状态的，我们需要采用 Token 来保持登录状态。原理如图 3-4，在登录过程中，服务器生成 Token 并返回给客户端。客户端随后

将 Token 附加到请求头，并发送给服务器。服务器通过验证 Token 的有效性来判断用户的身份。

图 3-4Token 原理分析图

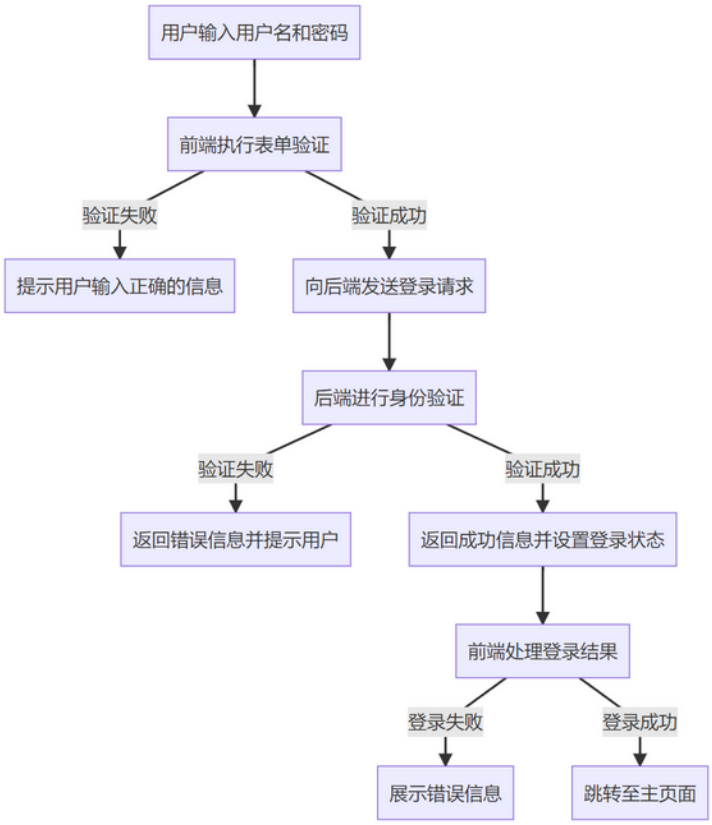

图 3-5 登录流程图

3.3.2 权限管理设计

权限管理模块是电商后台管理系统的关键组成部分，它通过控制不同用户可以执行的操作来保障系统的安全性和稳定性。通常，权限管理采用基于角色的访问控制策略，将权限分配给不同角色，然后将角色分配给用户如图 3-5 权限管理图。以下是权限管理业务分析的要点：

1） 角色管理：系统管理员可以创建、编辑和删除角色。每个角色具有一组特定的功能权限，这些权限决定了分配给该角色的用户可以执行哪些操作。

2） 权限分配：系统管理员可以为每个角色分配一组功能权限。这些权限可以包括访问特定页面、执行特定操作（如添加、修改或删除数据）等。

3） 用户角色分配：系统管理员为每个用户分配一个或多个角色。用户的操作权限取决于其所属角色的权限。通过更改用户的角色，可以灵活地调整用户的权限。

4） 权限验证：在用户尝试访问受保护资源或执行受限操作时，系统会检查用户所属角色的权限，确保用户具有执行该操作的权限。如果用户没有相应权限，系统将拒绝访问或执行操作。

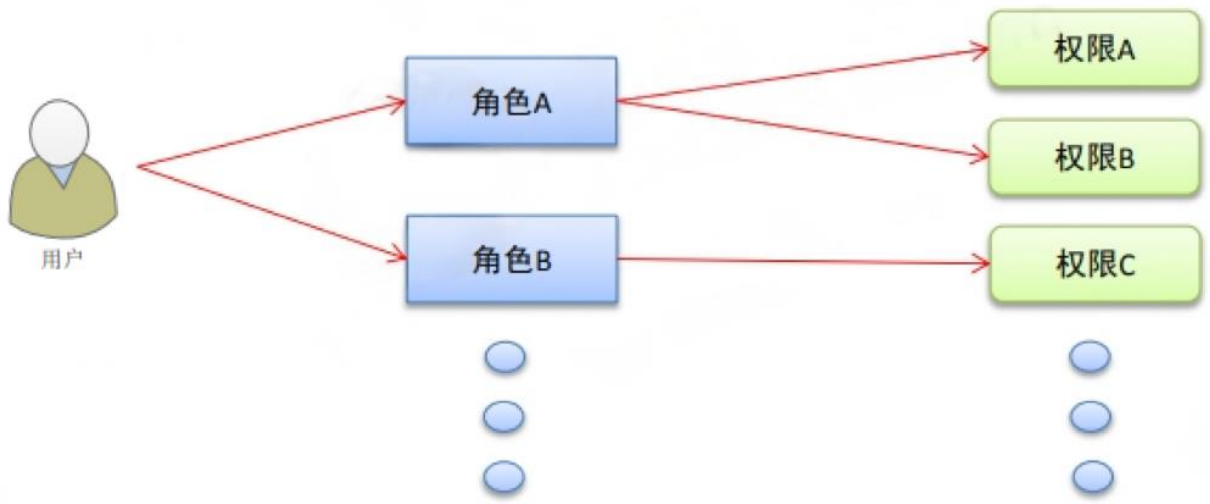

图 3-5 权限管理图

3.3.3 用户管理设计

用户管理模块是电商后台管理系统中的核心部分，它关注于对用户信息的维护和管理。通过用户管理，系统管理员可以对用户进行添加、删除、修改和查询等操作，从而实现对用户数据的有效控制。用户管理模块主要包括以下几个方面：

1） 用户列表展示：系统管理员可以查看所有用户的信息，包括用户名、邮箱、手机号等。用户列表以表格形式展示，便于管理员快速浏览和查找用户。

2） 分页功能：当用户数量较多时，系统会将用户列表分成多个页面进行展示。管理员可以根据需要选择每页显示的用户数量，以便更有效地管理用户数据。

3） 搜索功能：管理员可以通过输入关键字对用户列表进行筛选，快速找到需要管理的用户。

4） 添加用户：管理员可以创建新用户并添加到用户列表中。在添加过程中，需要对用户信息进行验证，以确保所输入的信息合法且唯一。

5） 修改用户：管理员可以对已有用户进行信息修改，包括用户名、密码、邮箱和手机号等。修改操作需要对用户信息进行验证，防止输入非法数据。

6） 删除用户：管理员可以从用户列表中删除不再需要的用户。在执行删除操作时，系统会弹出确认框，避免误操作导致用户数据丢失。

用户管理模块通过实现这些功能，为电商后台管理系统提供了一个有效的用户数据维护工具。它与权限管理模块相互配合，共同保障系统的安全性和稳定性。

3.3.4 商品管理设计

商品管理模块是电商平台的核心部分，负责处理与商品相关的所有信息和操作。以下是商品管理分析的要点：

1) 商品信息管理：系统管理员可以添加、编辑、查看和删除商品的基本信息，如名称、描述、价格、库存等。这些信息是电商平台向用户展示的商品属性，因此需要确保数据的准确性和完整性。
2) 商品分类：商品管理模块支持创建和管理商品分类，以便将具有相似特征的商品归类到一起，方便用户浏览和查找。管理员可以为每个商品分配一个或多个分类。
3) 商品参数：商品参数用于描述商品的详细特性，如尺寸、颜色、材质等。商品管理模块允许管理员为不同类型的商品设置各种参数，以便为用户提供丰富的商品信息。
4) 商品图片管理：商品图片是电商平台中展示商品外观的关键要素。商品管理模块应支持上传、编辑和删除商品图片，以便为用户提供清晰、准确的商品视觉效果。
5) 库存管理：商品管理模块需要支持对商品库存的实时监控和管理，以便在库存不足时及时补充，确保商品的正常销售。

3.4 数据库设计

3.4.1 数据库概念设计

数据库的设计是电商后台管理系统开发中的核心部分之一。它负责存储和管理系统中的各种数据，如商品信息、订单数据、用户信息等。一个良好设计的数据库可以提高系统的性能、安全性和可扩展性。

在电商后台管理系统的数据库设计中，主要包含以下实体类：商品实体、订单实体、用户实体、管理员实体等。这些实体之间通过不同的关系来建立联系，常见的关系有一对一（1：1）、多对一（m：1）和多对多（m：n）。

例如，商品实体和订单实体之间通过订单详情来建立多对一的关系，表示一个订单可以包含多个商品；用户实体和订单实体之间通过用户 ID 来建立多对一的关系，表示一个用户可以拥有多个订单；管理员实体和商品实体之间通过商品管理来建立多对一的关系，表示一个管理员可以管理多个商品。

此外，数据库设计还需考虑数据的完整性和约束条件。例如，商品价格不能为负数，用户 ID 和订单 ID 需要唯一且非空等。

通过合理的数据库设计，电商后台管理系统能够高效地存储和管理大量的数据，并支持系统的各种功能和业务需求。同时，合适的索引、表结构优化以及数据备份和恢复策略也是数据库设计中需要考虑的重要方面，以确保系统的稳定性和可靠性。

3.4.2 数据库逻辑设计

通过数据库各实体类之间对应的关系的分析，我们进行下一段数据库的逻辑设计，如下所示（括号内为属性加粗的部分为主键）：

管理员：Id 编号（主键自增长）、用户名、密码、新增时间、角色 id、手机号码、邮箱、状态

角色：Id 编号（主键自增长）、角色名、角色描述

商品：Id 编号（主键自增长）、商品名字、商品价格、商品数量、商品重量、分类编号、商品介绍

分类：Id 编号（主键自增长）、父 id、分类名称、状态、创建时间

权限：Id 编号（主键自增长）、权限名称、权限等级

3.4.3 数据库物理实现

通过数据库的概念设计和逻辑设计，我们对数据库进行物理实现。将数据库的各个表

以及对应的属性信息和关系来设计数据库的字段表。

1. manager 表主要存放的是用户的数据信息，以主键 Id 为约束，具体的数据表的结构设计如下所示：

表 3-6 用户 manager 表

字段名称	字段类型	字段长度	是否为空	是否为主键	字段名称
Mg_id	int	11	True	主键	是
mg_name	varchar	32	True	名称	否
mg_pwd	char	64	True	密码	否
mg_time	int	10	True	注册时间	否
role_id	tinyint	11	True	角色 id	否
mg_mobile	varchar	32	False	手机号	否
mg_email	varchar	64	False	邮箱	否
mg_state	tinyint	2	False	状态	否

2. role 表主要存放的是管理员的数据信息，以主键 Id 为约束，具体的数据表的结构设计如下所示：

表 3-7 管理员 role 表

字段名称	字段类型	字段长度	是否为空	是否为主键	字段名称
role_id	smallint	6	True	主键	是
role_name	varchar	20	True	角色名称	否
ps_ids	varchar	512	True	权限集合 id	否
ps_ca	text	1	False	权限分类	否
role_desc	text		False	权限描述	否

3. goods 表主要存放的是商品信息的数据信息，以主键 Id 为约束，具体的数据表的结构设计如下所示：

表 3-8 商品信息 goods 表

字段名称	字段类型	字段长度	是否为空	是否为主键	字段名称
goods_id	mediumint	8	True	主键	是
goods_name	varchar	255	True	商品名称	否
goods_price	decimal	10	True	商品价格	否
goods_number	int	8	False	商品数量	否
goods_weight	smallint	5	False	商品重量	否
cat_id	smallint	5	False	类型 id	否
goods_introduce	text		False	商品详情介绍	否
add_time	int	11	False	添加时间	否
upd_time	int	11	False	更新时间	否
cat_one_id	smallint	5	False	一级分类	否
cat_two_id	smallint	5	False	二级分类	否

4. permission 表主要存放的是权限的相关信息，以主键 id 为约束，具体的数据表结构设计如下所示：

表 3-9 权限信息 permission 表

字段名称	字段类型	字段长度	是否为空	是否为主键	字段名称
id	Smallint	6	True	主键	是
Name	Varchar	20	True	权限名称	否
Pid	smallint	6	True	父 id	否
Ps-c	varchar	32	True	权限集合	否
Ps-level	enum		True	权限等级	否

5. goods_cats 表主要存放的是商品分类的相关信息，以主键 id 为约束，具体的数据表结构设计如下所示：表 3-10 商品分类 goods_cats 表

字段名称	字段类型	字段长度	是否为空	是否为主键	字段名称
Cat_id	int	11	True	主键	是
Parent_id	int	11	True	分类 id	否
Cat_name	varchar	50	True	父级 id	否
Is_show	tinyint	4	True	是否显示	否
Cat_sort	int	11	True	分类排序	否
Data_flag	tinyint	4	True	数据标记	否
Create_time	int	11	True	创建时间	否

第四章 系统实现

4.1 系统登录

登录功能是电商后台管理系统中的一个关键功能，它涉及到用户身份验证、权限控制等方面，当用户可以根据自己的账户和密码才能进行正确的登录，如果是输入的账户或者是密码不当，将会出现“登录失败”的提示，并停留在本登录页面。只有用户名以及密码信息（包括所选择的用户权限都正确）才能进行登录完成页面的跳转。使用 Element UI 的表单组件构建登录界面。使用 Node.js 搭建后端服务器，处理登录请求和身份验证。页面实现和前端代码如图所示：

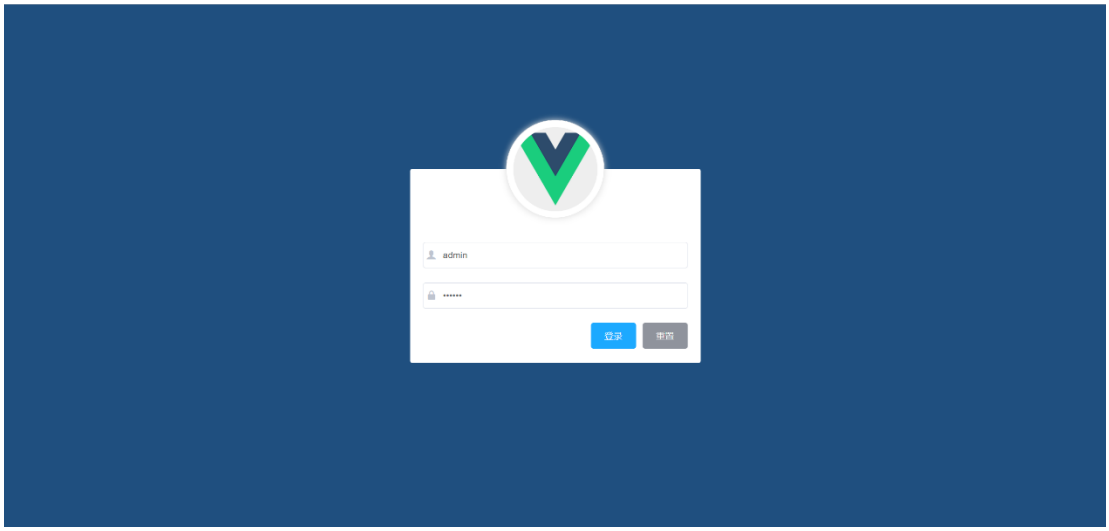

图 4-1 登录首页

图 4-2 登录前端代码

在登录的时候，用户通过客户端提交登录请求，传递用户名和密码，服务器收到请求，调用 login 函数来处理登录逻辑，login 函数负责根据提供的用户名和密码来查询数据库并验证用户身份，若用户和密码验证成功，服务器会生成一个 JWT token，这个 token 包含了用户 id 和角色 id 等信息，之后会将 token 和用户信息一起返回客户端，实现的部分代码如下所示：

```
module.exports.login = function(username, password, cb) {
    managersDAO.findOne({"mg_name": username}, function(err, manager) {
        if (err || !manager) return cb("用户名不存在");

        if (manager.role_id < 0) {
            return cb("该用户没有权限登录");
        }

        if (manager.role_id != 0 && manager.mg_state != 1) {
            return cb("该用户已经被禁用");
        }

        if (Password.verify(password, manager.mg_pwd)) {
            cb(
                null,
                {
                    "id": manager.mg_id,
                    "rid": manager.role_id,
                    "username": manager.mg_name,
                    "mobile": manager.mg_mobile,
                    "email": manager.mg_email,
                }
            );
        } else {
            return cb("密码错误");
        }
    });
}
```

```
module.exports.login = function(req, res, next) {
    passport.authenticate('local', function(err, user, info) {
        if (err) return res.sendResult(null, 400, err);
        if (!user) return res.sendResult(null, 400, "参数错误");

        // 获取角色信息
        var token = jwt.sign({"uid": user.id, "rid": user.rid}, jwt_config.get("secretKey"),
{"expiresIn": jwt_config.get("expiresIn")});
        user.token = "Bearer " + token;
        return res.sendResult(user, 200, '登录成功');
    })(req, res, next);
}
```

图 4-3 ManagerService.login 函数

图 4-4 module.exports.login 函数

4.2 用户管理

用户管理模块是电商后台管理系统中的核心部分，通过用户管理页面，可以实现查询用户列表，获取用户信息，创建用户，修改用户信息，删除用户信息，分配用户角色，设置用户状态等功能，以下是用户管理页面的实现页面和部分前端代码：

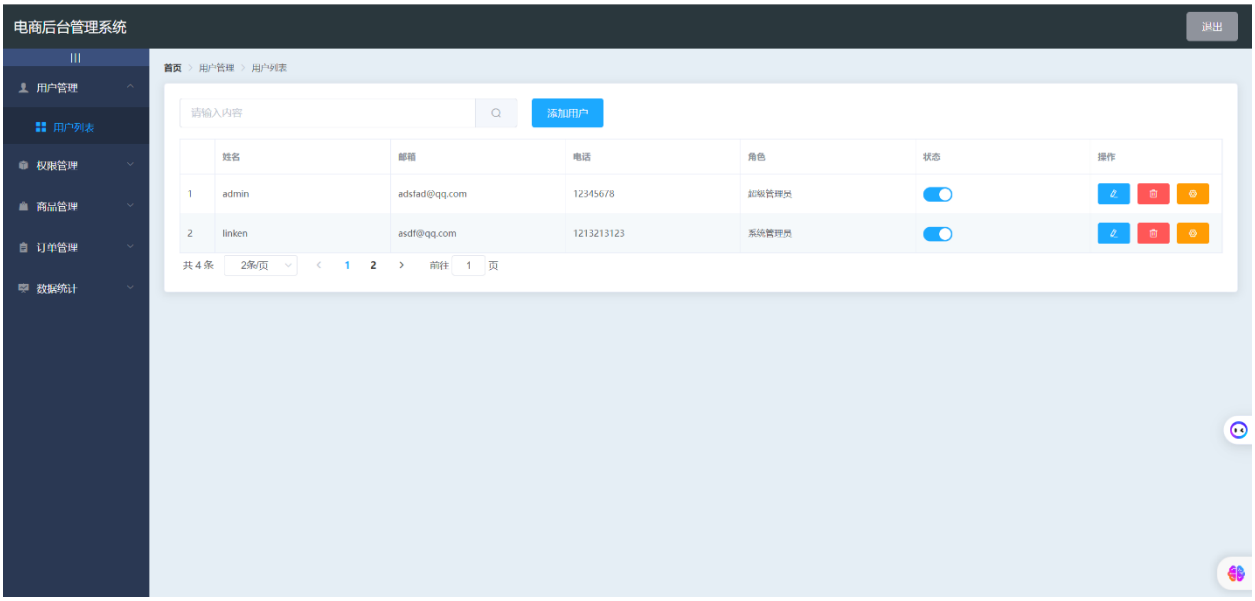

图 4-5 用户管理页面

```vue
<template>
  <div>
    <!-- 面包屑导航区域 -->
    <el-breadcrumb separator-class="el-icon-arrow-right">
      <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
      <el-breadcrumb-item>用户管理</el-breadcrumb-item>
      <el-breadcrumb-item>用户列表</el-breadcrumb-item>
    </el-breadcrumb>

    <!-- 卡片视图区域 -->
    <el-card>
      <!-- 搜索与添加区域 -->
      <el-row :gutter="20">
        <el-col :span="8">
          <el-input placeholder="请输入内容" v-model="queryInfo.query" clearable @clear="getUserList">
            <el-button slot="append" icon="el-icon-search" @click="getUserList"></el-button>
          </el-input>
        </el-col>
        <el-col :span="4">
          <el-button type="primary" @click="addDialogVisible = true">添加用户</el-button>
        </el-col>
      </el-row>
    ...
    ...
    </div>
</template>
```

图 4-6 用户管理前端部分代码

在用户管理页面种，主要是通过各个路由处理函数来完成实现不同的功能，首先接受客户端的请求，根据请求的方法和路径执行相应的路由处理函数，进行参数验证和业务逻辑处理，最后返回处理结果给客户端，实现用户管理的增删改查功能。以下是部分路由处理函数的代码：

```
router.post("/",
    // 验证参数
    function(req,res,next) {
        if(!req.body.username){
            return res.sendResult(null,400,"用户名不能为空");
        }
        if(!req.body.password) {
            return res.sendResult(null,400,"密码不能为空");
        }
        if(!req.body.rid) {
            req.body.rid = -1;
            //return res.sendResult(null,200,"角色ID不能为空");
        }
        if(isNaN(parseInt(req.body.rid))) req.body.rid = -1;//return res.sendResult(null,200,"角色ID必须
是数字");
        next();
    },
    // 处理业务逻辑
    function(req,res,next) {
        params = {
            "username":req.body.username,
            "password":req.body.password,
            "mobile":req.body.mobile,
            "email":req.body.email,
            "rid":req.body.rid
        }
        mgrServ.createManager(params,function(err,manager){
            if(err) return res.sendResult(null,400,err);
            res.sendResult(manager,201,"创建成功");
        })(req,res,next);
    }
);
```

图 4-7 创建用户路由代码

```
router.put("/:id",
    // 参数验证
    function(req,res,next) {
        if(!req.params.id) {
            return res.sendResult(null,400,"用户ID不能为空");
        }
        if(isNaN(parseInt(req.params.id))) return res.sendResult(null,400,"用户ID必须是数字");
        next();
    },
    // 处理业务逻辑
    function(req,res,next) {
        mgrServ.updateManager(
            {
                "id":req.params.id,
                "mobile":req.body.mobile,
                "email":req.body.email
            },
            function(err,manager) {
                if(err) return res.sendResult(null,400,err);
                res.sendResult(manager,200,"更新成功");
            }
        )(req,res,next);
    }
);
```

图 4-8 修改用户路由代码

```
router.put("/:id/role",
    // 参数验证
    function(req,res,next) {
        if(!req.params.id) {
            return res.sendResult(null,400,"用户ID不能为空");
        }
        if(isNaN(parseInt(req.params.id))) return res.sendResult(null,400,"用户ID必须是数字");

        if(req.params.id == 500) return res.sendResult(null,400,"不允许修改admin账户");

        if(!req.body.rid) res.sendResult(null,400,"权限ID不能为空");
        next();
    },
    // 处理业务逻辑
    function(req,res,next) {
        mgrServ.setRole(req.params.id,req.body.rid,function(err,manager){
            if(err) return res.sendResult(null,400,err);
            res.sendResult(manager,200,"设置角色成功");
        })(req,res,next);
    }
);
```

图 4-9 分配用户角色路由代码

4.3 权限管理

通过权限管理可以定义用户可以访问的功能和资源，它通过控制不同用户可以执行的操作来保障系统的安全性和稳定性。通常，权限管理采用基于角色的访问控制策略，将权限分配给不同角色，以下是权限管理的前端页面和部分代码：

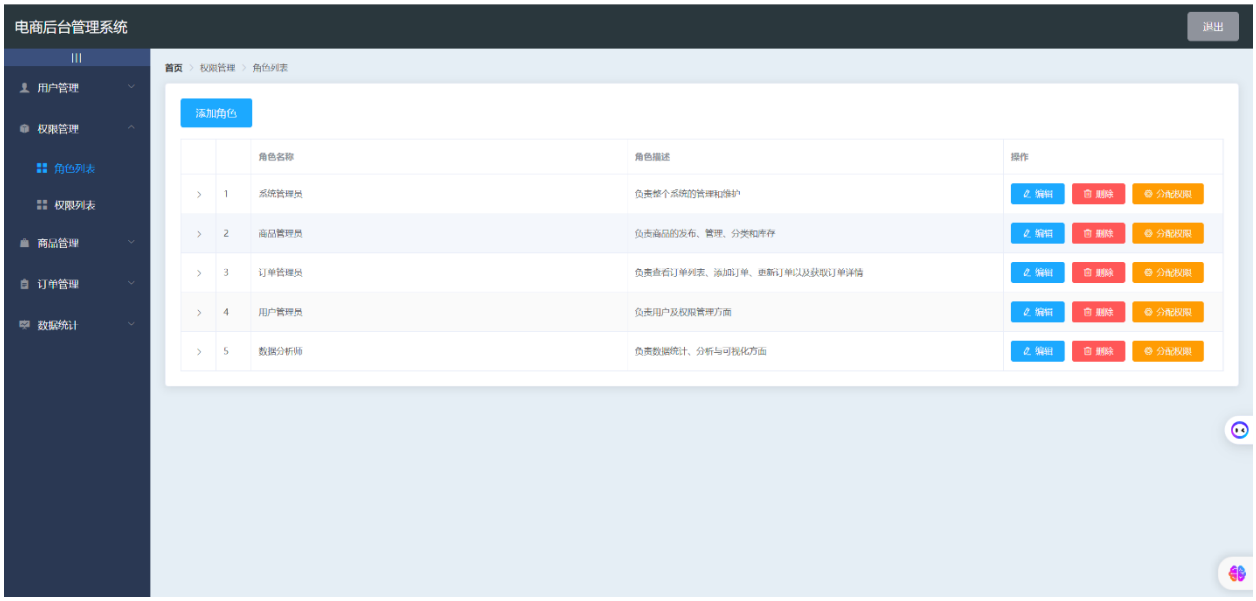

图 4-10 权限管理页面

```
<template>
  <div>
    <!-- 面包屑导航区域 -->
    <el-breadcrumb separator-class="el-icon-arrow-right">
      <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
      <el-breadcrumb-item>权限管理</el-breadcrumb-item>
      <el-breadcrumb-item>角色列表</el-breadcrumb-item>
    </el-breadcrumb>
    <!-- 卡片视图 -->
    <el-card>
      <!-- 添加角色按钮区域 -->
      <el-row>
        <el-col>
          <el-button type="primary" @click="dialogFormVisible = true">添加角色</el-button>
        </el-col>
      </el-row>
      <!-- 角色列表区域 -->
      <el-table :data="rolelist" border stripe>
        <!-- 展开列 -->
        <el-table-column type="expand">
          <template slot-scope="scope">
            <el-row :class="['bdbottom', i1 === 0 ? 'bdtop' : '', 'vcenter']" v-for="(item1, i1) in
scope.row.children"
              :key="item1.id">
              <!-- 渲染一级权限 -->
              <el-col :span="5">
                <el-tag closable @close="removeRightById(scope.row, item1.id)">{{ item1.authName }}
</el-tag>
                <i class="el-icon-caret-right"></i>
              </el-col>
              <!-- 渲染二级和三级权限 -->
              <el-col :span="19">
                <!-- 通过 for 循环 嵌套渲染二级权限 -->
                <el-row :class="[i2 === 0 ? '' : 'bdtop', 'vcenter']" v-for="(item2, i2) in
item1.children"
                  :key="item2.id">
                  <el-col :span="6">
                    <el-tag type="success" closable @close="removeRightById(scope.row, item2.id)">{{
item2.authName
                    }}</el-tag>
                    <i class="el-icon-caret-right"></i>
                  </el-col>
                  <el-col :span="18">
                    <el-tag type="warning" v-for="(item3, i3) in item2.children" :key="item3.id"
closable
                      @close="removeRightById(scope.row, item3.id)">{{ item3.authName }}</el-tag>
                  </el-col>
                </el-row>
              </el-col>
            </el-row>

            <!-- <pre>
              {{scope.row}}
            </pre> -->
          </template>
    ...
    ...
  </div>
</template>
```

图 4-11 权限管理前端代码

权限管理的核心是通过定义路由处理函数来实现相关业务，再通过调用 roleServ 方法对角色进行增删改查，授权和取消等操作，以下是部分相关核心代码：

```
router.get("/",
    // 参数验证
    function(req,res,next){
        next();
    },
    // 处理业务逻辑
    function(req,res,next) {
        roleServ.getAllRoles(function(err,result) {
            if(err) return res.sendResult(null,400,err);
            res.sendResult(result,200,"获取成功");
        })(req,res,next);
    }
);
```

图 4-12 获取角色列表

```
router.post("/:id/rights",
    // 参数校验
    function(req,res,next) {
        if(!req.params.id) return res.sendResult(null,400,"角色ID不能为空");
        if(isNaN(parseInt(req.params.id))) res.sendResult(null,400,"角色ID必须为数字");
        next();
    },
    // 业务逻辑
    function(req,res,next) {
        roleServ.updateRoleRight(req.params.id,req.body.rids,function(err,newRole){
            if(err) return res.sendResult(null,400,err);
            res.sendResult(null,200,"更新成功");
        })(req,res,next);
    }
);
```

图 4-13 角色授权

```
router.put("/:id",
    // 参数验证
    function(req,res,next) {
        if(!req.params.id) return res.sendResult(null,400,"角色ID不能为空");
        if(isNaN(parseInt(req.params.id))) return res.sendResult(null,400,"角色ID必须为数字");
        if(!req.body.roleName) return res.sendResult(null,400,"角色名称不能为空");
        next();
    },
    // 处理业务逻辑
    function(req,res,next) {
        roleServ.updateRole(
            {
                "id":req.params.id,
                "roleName":req.body.roleName,
                "roleDesc":req.body.roleDesc
            },
            function(err,result){
            if(err) return res.sendResult(null,400,err);
            res.sendResult(result,200,"获取成功");
        })(req,res,next);
    }
);
```

图 4-14 更新角色信息

4.4 商品管理

商品管理模块是电商后台管理系统的重要组成部分，通过商品管理页面，可以实现查询商品列表、获取商品详细信息、添加商品、修改商品信息、删除商品、更新商品图片、更新商品属性和更新商品状态等功能。以下是商品管理页面的实现图片和部分代码：

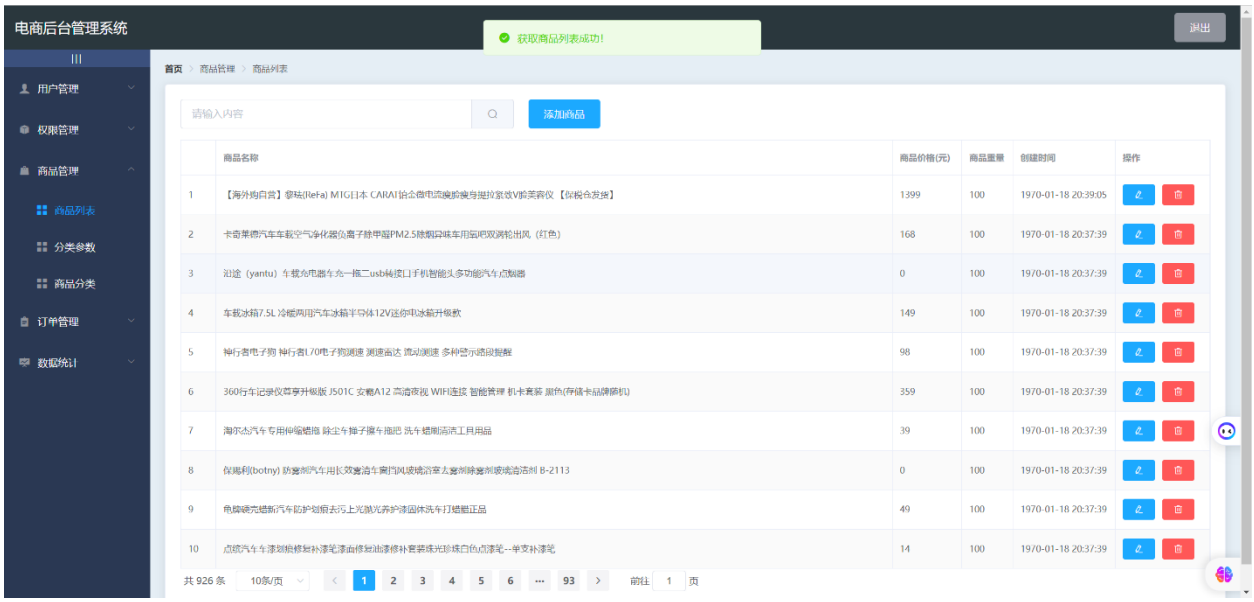

图 4-15 商品管理页面

```
<template>
    <div>
      <!-- 面包屑导航区域 -->
      <el-breadcrumb separator-class="el-icon-arrow-right">
        <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
        <el-breadcrumb-item>商品管理</el-breadcrumb-item>
        <el-breadcrumb-item>参数列表</el-breadcrumb-item>
      </el-breadcrumb>

      <!-- 卡片视图区域 -->
      <el-card>
        <!-- 警告区域 -->
        <el-alert show-icon title="注意：只允许为第二级分类设置相关参数！" type="warning" :closable="false">
</el-alert>

        <!-- 选择商品分类区域 -->
        <el-row class="cat_opt">
          <el-col>
            <span>选择商品分类：</span>
            <!-- 选择商品分类的级联选择框 -->
            <el-cascader expand-trigger="hover" :options="catelist" :props="cateProps" v-
model="selectedCateKeys" @change="handleChange">
            </el-cascader>
          </el-col>
        </el-row>

        <!-- tab 页签区域 -->
        <el-tabs v-model="activeName" @tab-click="handleTabClick">
          <!-- 添加动态参数的面板 -->
          <el-tab-pane label="动态参数" name="many">
            <!-- 添加参数的按钮 -->
            <el-button type="primary" size="mini" :disabled="isBtnDisabled"
@click="addDialogVisible=true">添加参数</el-button>
            <!-- 动态参数表格 -->
            <el-table :data="manyTableData" border stripe>
              <!-- 展开行 -->
              <el-table-column type="expand">
                <template slot-scope="scope">
                  <!-- 循环渲染Tag标签 -->
                  <el-tag v-for="(item, i) in scope.row.attr_vals" :key="i" closable
@close="handleClose(i, scope.row)">{{item}}</el-tag>
                  <!-- 输入的文本框 -->
                  <el-input class="input-new-tag" v-if="scope.row.inputVisible" v-
model="scope.row.inputValue" ref="saveTagInput" size="small"
@keyup.enter.native="handleInputConfirm(scope.row)" @blur="handleInputConfirm(scope.row)">
                  </el-input>
                  <!-- 添加按钮 -->
                  <el-button v-else class="button-new-tag" size="small" @click="showInput(scope.row)">+
New Tag</el-button>
                </template>
              ...
              ...

    </div>
  </template>
```

图 4-16 商品管理前端代码

在商品管理页面中，主要通过各个路由处理函数来完成实现不同功能。首先，服务器接收客户端的请求，根据请求的方法和路径执行相应的路由处理函数。接下来，进行参数验证和业务逻辑处理。最后，返回处理结果给客户端，实现商品管理的增删改查功能。部分代码如图所示：

```
router.get("/:id",
    // 参数验证
    function(req,res,next) {
        if(!req.params.id) {
            return res.sendResult(null,400,"商品ID不能为空");
        }
        if(isNaN(parseInt(req.params.id))) return res.sendResult(null,400,"商品ID必须是数字");
        next();
    },
    // 业务逻辑
    function(req,res,next) {
        goodServ.getGoodById(req.params.id,function(err,good){
            if(err) return res.sendResult(null,400,err);
            return res.sendResult(good,200,"获取成功");
        })(req,res,next);
    }
);
```

图 4-17 获取商品详情

```
router.put("/:id",
    // 参数验证
    function(req,res,next) {
        if(!req.params.id) {
            return res.sendResult(null,400,"商品ID不能为空");
        }
        if(isNaN(parseInt(req.params.id))) return res.sendResult(null,400,"商品ID必须是数字");
        next();
    },
    // 业务逻辑
    function(req,res,next) {
        var params = req.body;
        goodServ.updateGood(req.params.id,params,function(err,newGood){
            if(err) return res.sendResult(null,400,err);
            res.sendResult(newGood,200,"创建商品成功");
        })(req,res,next);
    }
);
```

图 4-18 更新商品

```
router.post("/",
    // 参数验证
    function(req,res,next) {
        next();
    },
    // 业务逻辑
    function(req,res,next) {
        var params = req.body;
        goodServ.createGood(params,function(err,newGood){
            if(err) return res.sendResult(null,400,err);
            res.sendResult(newGood,201,"创建商品成功");
        })(req,res,next);
    }
);
```

图 4-19 添加商品

第五章 系统测试

5.1 测试的目的与方法

系统的测试是为了发现系统内部存在的错误和检测出潜在的 Bug。通过测试，我们可以确保各个功能在系统中可以正常使用，并对数据的输入、输出以及系统的判断和容错能力进行验证。通过对错误出现的地方进行系统改进和优化，我们能够确保系统能够正确地运行并延长其生命周期。

在进行系统测试时，开发人员通常会采用黑盒测试和白盒测试两种方法，以实现全面的测试覆盖。

黑盒测试将整个系统看作一个不透明的盒子，在测试过程中不考虑内部的程序结构，而是关注系统接口和功能的使用。通过这种测试方法，我们可以验证系统是否按照预期的方式响应用户的输入，并检查系统是否产生正确的输出结果。黑盒测试帮助我们检测系统的功能是否符合需求，并发现潜在的逻辑错误或者与其他组件的集成问题。

白盒测试则是在了解系统内部结构和执行方式的基础上进行的全面测试。这种测试方法考虑到系统的程序结构、代码覆盖率等方面，通过检查代码逻辑、循环和条件判断等来验证系统的正确性。白盒测试可以帮助我们发现系统内部的错误、漏洞和潜在的性能问题，并优化系统的性能和稳定性。

通过综合应用黑盒测试和白盒测试，我们能够全面检验系统的功能、性能和可靠性。黑盒测试帮助我们验证系统是否满足用户需求，而白盒测试则有助于发现并修复系统内部的问题。这两种测试方法的结合将有助于确保系统的质量和可用性，从而提供稳定可靠的电商后台管理系统。

5.2 测试的内容和结论

本次的测试用例我们以登录功能以及权限的管理进行测试，通过测试登录功能是否可以正常使用。本次测试用例如下表所示：

测试用例编号	测试项	测试步骤	输入数据	预期结果	测试结果
登录测试流程	打开系统登录页面	使用谷歌浏览器打开管理系统登录页面	http://localhost:8080/	成功跳转至系统登录页面	成功跳转至系统登录页面
登录测试第一组	输入用户名和密码	1、输入正确的用户名，正确的密码 2、点击登录按钮	用户名：admin 密码：123456	成功跳转至系统首页	成功跳转至系统首页
登录测试第二组	输入用户名和密码	1、输入正确的用户名,不正确的密码 2、点击登录按钮	用户名：admin 密码：12345678	提示登陆失败	提示登陆失败
登录测试第三组	输入用户名和密码	1、输入不正确的用户名，正确的密码 2、点击登录按钮	用户名：user 密码：123456	提示登陆失败	提示登陆失败
登录测试第四组	输入用户名和密码	1、输入不正确的用户名,不正确的密码 2、点击登录按钮	用户名:xxx 密码:null	提示登陆失败	提示登陆失败

表 5-1 登陆测试用例表

测试组别	测试过程	预期结果	实际结果	与预期相比
添加测试 1 组	输入正确的用户数据信息，点击“确认”	正常添加用户信息	正常添加用户信息	一致
添加测试 2 组	输入正确的用户数据信息但是有部分空白，点击“确认”	用户信息添加失败	用户信息添加失败	一致
添加测试 3 组	输入正确的用户名及密码,但邮箱格式有误，点击“确定”	无法添加	提示“请输入合法的邮箱”没有反应	一致
添加测试 4 组	输入正确的用户名及密码,但手机号格式有误，点击“确定”	无法添加	提示“请输入合法的手机号”没有反应	一致

表 5-2 用户信息添加表

测试则别	测试过程	预期结果	实际结果	与预期相比
修改测试 1 组	输入正确的数据类型和数据信息，点击“确认修改”	修改信息成功，数据信息更新	修改信息成功，数据信息更新	一致
修改测试 2 组	输入正确的邮箱但留有空白手机号点击“确认”	无法修改	修改失败，“提示输入用户手机”	一致
修改测试 3 组	输入正确的手机号但输入错误格式的邮箱点击“确认”	无法修改	修改失败，“提示输入合法的邮箱”	一致

表 5-3 用户信息修改测试表

通过上述测试，我们发现本次登录功能测试能够正常使用，与我们的预期目标一致。系统能够正确进行页面跳转，并在用户添加、修改功能方面表现正常。我们对数据流进行了验证，并且功能运行良好。系统能够准确判断数据流，并基于相应的返回值进行处理。我们的设计不仅仅局限于登录测试，还对其他功能进行了全面的测试。

第六章 总结与展望

本文所论述的是一款基于 Vue+Element UI+Node.js 的电商后台管理系统。本文从设计背景开始进行论述，对相关技术也进行了介绍。本文论文论述的重点在于系统的需求分析和设计以及系统的实现上，对系统的测试也进行了描述。通过本次的设计让我对系统的开发流程有了深刻的认识，也在系统的开发过程中对注释书写的重要性和代码的规范性有了更深刻的认识，代码的规范可以保证系统进行后期的维护以及错误的排查。在设计上我也遇到了一些问题，比如在使用 Vue 和 Element UI 时遇到了组件的嵌套和交互问题。通过学习和实践，我最终解决了这些困难。

本次的设计虽然圆满完成，但由于个人开发能力和时间限制等原因，仍存在一些不足之处。例如，系统的某些功能可能还不够完善，用户体验方面还有待提高等。随着个人开发能力的提升和时间的增加，我将进一步改进和完善系统的功能和用户体验。

随着电商行业的快速发展，电商后台管理系统的需求也日益增加。本次设计的电商后台管理系统可以帮助商家管理商品、订单、用户等信息，提供数据统计和分析功能，以及其他管理功能。预计将来这样的系统将得到更广泛的应用和推广，因此本次设计具有很大的发展空间。