目 录

[摘 要 4](#_Toc195254856)

[Abstract 5](#_Toc195254857)

[1 绪论 1](#_Toc195254858)

[1.1研究背景 1](#_Toc195254859)

[1.2研究目的和意义 1](#_Toc195254860)

[1.3国内外研究文献综述 1](#_Toc195254861)

[1.3.1基于微信小程序的直播带货的现状分析 1](#_Toc195254862)

[1.3.2基于机器学习的数据挖掘 2](#_Toc195254863)

[1.4本章小结 4](#_Toc195254864)

[2 技术理论基础 5](#_Toc195254865)

[2.1MySQL数据库 5](#_Toc195254866)

[2.2Python语言 5](#_Toc195254867)

[2.3Django框架 6](#_Toc195254868)

[2.4微信开发者工具 6](#_Toc195254869)

[2.5小程序目录结构以及框架介绍 6](#_Toc195254870)

[2.6本章小结 7](#_Toc195254871)

[3 系统分析 8](#_Toc195254872)

[3.1可行性分析 8](#_Toc195254873)

[3.1.1技术可行性分析 8](#_Toc195254874)

[3.1.2经济可行性分析 8](#_Toc195254875)

[3.1.3法律可行性分析 9](#_Toc195254876)

[3.2系统性能分析 9](#_Toc195254877)

[3.2.1系统安全性 9](#_Toc195254878)

[3.2.2可维护性和适应性 9](#_Toc195254879)

[3.2.3可靠性 10](#_Toc195254880)

[3.3功能需求分析 10](#_Toc195254881)

[3.3.1微信小程序用户端功能需求 10](#_Toc195254882)

[3.3.2管理员后台功能需求 11](#_Toc195254883)

[3.3.3商家后台功能需求 12](#_Toc195254884)

[3.4本章小结 12](#_Toc195254885)

[4系统设计 14](#_Toc195254886)

[4.1整体结构设计 14](#_Toc195254887)

[4.2数据库设计 14](#_Toc195254888)

[4.2.1数据库概念设计 14](#_Toc195254889)

[4.2.2数据库主要表E-R图 14](#_Toc195254890)

[4.2.3数据库表设计 15](#_Toc195254891)

[4.3本章小结 19](#_Toc195254892)

[5系统实现 20](#_Toc195254893)

[5.1系统开发环境以及运行环境 20](#_Toc195254894)

[5.2管理员后台设计与实现 20](#_Toc195254895)

[5.2.1登录设计与实现 20](#_Toc195254896)

[5.2.2数据分析与可视化设计与实现 22](#_Toc195254897)

[5.2.3数据爬取流程 33](#_Toc195254898)

[5.2.4通知公告发布流程 34](#_Toc195254899)

[5.3微信小程序用户端功能实现 35](#_Toc195254900)

[5.3.1注册与登录流程 35](#_Toc195254901)

[5.3.2商品浏览流程 37](#_Toc195254902)

[5.3.3留言反馈流程 38](#_Toc195254903)

[5.4商家后台功能实现 41](#_Toc195254904)

[5.4.1商品管理流程 41](#_Toc195254905)

[5.4.2订单管理流程 42](#_Toc195254906)

[5.5本章小结 43](#_Toc195254907)

[6 功能测试 44](#_Toc195254908)

[6.1测试用例 44](#_Toc195254909)

[6.2测试结果 45](#_Toc195254910)

[6.3本章小结 45](#_Toc195254911)

[参考文献 46](#_Toc195254912)

[致 谢 48](#_Toc195254913)

[附件一：核心代码 49](#_Toc195254914)

# 摘 要

随着互联网和微信小程序的发展，直播带货成为电商热门趋势，但直播带货数据的有效分析和利用成为关键问题。本研究旨在设计并实现一个基于微信小程序的直播带货商品数据分析系统。首先，通过可行性分析、性能分析和功能需求分析，明确系统的开发方向。接着，采用前后端分离架构，结合 MySQL 数据库、Python 语言、Django 框架等技术进行系统设计与实现。最后，对系统进行功能测试，结果表明系统功能运行稳定，能有效实现直播带货数据的分析、可视化以及相关业务管理，为提升直播效果、优化商品推荐策略提供有力支持。

**关键词：**微信小程序；直播带货；数据分析；机器学习；前后端分离

# Abstract

With the development of the Internet and WeChat Mini Programs, live - streaming e - commerce has become a popular trend in the e - commerce field. However, the effective analysis and utilization of live - streaming e - commerce data have become key issues. This research aims to design and implement a commodity data analysis system for live - streaming e - commerce based on WeChat Mini Programs. Firstly, through feasibility analysis, performance analysis, and functional requirements analysis, the development direction of the system is clarified. Then, a front - end and back - end separated architecture is adopted, and technologies such as MySQL database, Python language, and Django framework are combined for system design and implementation. Finally, the functional testing of the system is carried out. The results show that the system runs stably, can effectively achieve the analysis and visualization of live - streaming e - commerce data, as well as related business management, providing strong support for enhancing the live - streaming effect and optimizing the commodity recommendation strategy.

**Keywords：**WeChat Mini Program; Live - streaming E - commerce; Data Analysis; Machine Learning; Front - end and Back - end Separation

# 1 绪论

## 1.1研究背景

随着互联网技术的快速发展，直播带货已成为电商领域的一大热门趋势。微信小程序作为微信生态内的重要一环，凭借其便捷性和用户粘性，为直播带货提供了广阔的平台。2024年11月13日，腾讯发布的2024年第三季度财报显示，微信及WeChat的合并月活跃账户数增至13.82亿。小程序交易生态活跃，季度内交易额突破两万亿元；据微信公开课披露的数据，2023年视频号直播带货GMV（商品交易总额）规模相比2022年增长了2倍，订单数量增长超244%。然而，直播带货过程中产生的海量商品数据如何有效分析、利用，成为提升直播效果、优化商品推荐策略的关键。本课题旨在利用现代信息技术，特别是大数据技术，设计并实现一个基于微信小程序的直播带货商品数据分析系统，以实现对直播数据的深度挖掘和有效利用。

## 1.2研究目的和意义

通过该毕业设计课题训练和规范的毕业设计说明书撰写，掌握数据科学与大数据技术的基本知识和基本技能，能够运用机器学习，在需求分析基础上，设计基于微信小程序的直播带货商品数据分析系统、实现直播带货商品数据分析及可视化。通过对直播带货商品的数据采集、处理、可视化和数据分析，全面了解直播带货效果，优化直播策略和商品选择，提升用户体验和销售效果。实现基于微信小程序的直播带货商品数据分析系统的需求分析、总体设计、详细设计、系统实现和测试，并能够撰写规范的毕业设计说明书。

## 1.3国内外研究文献综述

### 1.3.1基于微信小程序的直播带货的现状分析

电子商务在国内当前的市场环境下迅速发展,其中主要依赖于国内的大数据处理技术。而正是在大数据处理技术的支持下,电商运营的地位更为稳固,发展也更为多元化。国内对于直播带货的现状也有比较全面的分析。如董彦喆梳理了国内学者从多维度对直播电商的阐述和解释，对国内外有关直播电商的文献进行回顾，从实证角度归纳出直播电商情景下的影响因素和反应机制[1]。魏玉平等探究了直播带货营销手段对消费者购买意愿的影响机制[2]。王靓靓基于大数据,分析了直播电商带货模式[3]。陈宏君等提出了构建基于大数据的农产品直播带货决策支持系统、利用大数据优化选品策略和直播内容策划、实现精准推送和个性化服务以提高转化率、强化供应链管理以降低运营成本等提升路径[4]。而微信小程序作为微信生态内的一个重要组成部分，凭借其无需下载安装、即用即走的便捷性，以及微信庞大的用户基础，迅速成为直播带货的新兴平台。随着消费者对线上购物体验要求的不断提高，微信小程序直播带货以其独特的社交属性和私域流量优势，逐渐受到品牌和商家的青睐。相较于原生应用，小程序的开发成本更低，具有广泛的应用的前景。如张晓燕系统分析微信"小程序"的开发,并对其未来发展前景以及将要面临的问题进行了探析[5]。刘玉佳基于微信"小程序"开发的系统架构、关键技术和市场前景进行研究，为当前微信"小程序"开发的理论与实践研究提供参考[6]。关于微信小程序在商品展示系统方面的应用也较为广泛。如南方航空把销售卖场搬上云端,在微信小程序发起了一场"暑期游专场"网络直播带货,南航也成为全球首家在微信小程序进行直播销售的航空公司[7]。闫烁应用微信小程序技术，开发出一款便于买家卖家使用的商品展示系统，实现了商品信息的展示、分类、搜索等功能，并提供了用户管理、商品管理、订单管理等后台管理功能[8]。晁琼等对构建基于微信小程序的“指上农庄”农产品交易平台，帮助农民推广农产品，发展特色农业[9]。

### 1.3.2基于机器学习的数据挖掘

数据挖掘是一种复杂而强大的数据处理和分析方法，包含了数据收集、数据预处理、数据分析、模型建立、模型评估等步骤。通过数据挖掘，可以从大量的数据中提取有价值的信息和知识，为决策提供科学依据。对于数据的获取，国内外也有许多相关的研究，如孙隽韬针对商品评论信息的获取与分析,使用基于Python的爬虫与分词技术，并且设计了一套评论内容分析流程[10]。通过对京东商城关于显卡的部分商品的评论获取与分析实验，实验结果得出该方法具有可行性及识别准确性。胡晓青等使用Python语言开发爬虫程序，获得电商平台的销售数据加以分析[11]。对于模型的建立，机器学习为模型建立提供了丰富的算法和技术支持。这些算法和技术可以根据不同的任务需求进行选择和调整，以构建出最合适的模型。如王坦等基于S-R理论构建购物行为影响因素模型，然后依据影响因素模型建立logistic回归模型、支持向量机模型、随机森林模型以及结合以上3种算法的融合模型，并应用以上模型进行实证分析，最后使用预测效果最好的模型对各因素进行特征偏好和相关性分析，为直播电商行业发展提出针对性建议[12]。李翠萍基于决策树、支持向量机和XGBoost三种机器学习算法，从客户画像特征、行为特征、情感特征和价值特征四个方面选取指标构建模型并评价，以对比不同模型在同一个数据集上的预测结果[13]。Ishrat Jahan等通过开发一个综合客户分析平台,结合客户细分、推荐和流失预测技术来对抗客户流失，基于准确率和F1值，CatBoost在数据集中表现最好，并应用递归特征消除对客户细分的特征进行排序以构建聚类[14]。P.Vijayaragavan等使用自然语言处理(NLP)和机器学习(ML)方法从产品评价中自动提取和识别主观数据。这些统计数字最终可能会揭示消费者基础的有利、中性或消极态度的信息[15]。多层感知器（Multi-Layer Perceptron）,以下简称为MLP，该结构可以根据具体任务的需要进行调整。对于经过处理的数据集，可以使用MLP对数据集进行训练以及建模，并通过不断调整参数来取得良好的效果。例如，可以选择不同数量的隐藏层、神经元和激活函数，以适应不同的问题。这种灵活性使得MLP能够广泛应用于不同的领域和场景。目前国内外有许多将其应用于对于商品数据的处理。如赵娟禾利用构造的电商商品特征集群数据对多层混合深度神经网络模型的预测效果进行验证，对电商商品短期需求量进行预测[16]。江涛使用PCA降低维度,最后使用MLP深度学习方法，进行评论的正负评价分类,在实验过程中尝试调整各项参数值进行多次训练,以训练出更有效的深度学习模型[17]。S.John Livingston基于神经网络对亚马逊产品评论情感进行分析[18]。

通过对国内外相关文献的查阅，可以了解到基于微信小程序的直播带货商品数据分析系统的设计与实现具有广阔的前景和巨大的市场潜力。但目前部分系统可能仅提供基础的数据统计和可视化功能，而缺乏更深入的数据挖掘和机器学习算法支持。这限制了系统对复杂用户行为和市场趋势的预测能力。本研究将对于处理过的数据集，投入数据挖掘和机器学习算法，通过直播带货的销售数据进行预测，如使用线性回归算法，通过拟合历史销量数据的线性趋势来预测未来的销量变化趋势。从而达到全面了解直播带货效果、优化直播策略和商品选择、提升用户体验和销售效果的目的。

## 1.4本章小结

随着互联网与微信小程序的蓬勃发展，直播带货成为电商热门，但数据有效分析利用问题凸显。本研究旨在设计并实现相关数据分析系统，通过掌握数据科学技术，实现数据处理与可视化，优化直播及商品策略。国内外研究显示，微信小程序直播带货前景良好，数据挖掘成果丰富，但现有系统存在不足。本研究将运用算法预测销售数据，填补部分系统功能空白。

# 2 技术理论基础

## 2.1MySQL数据库

该研究开发的应用程序在数据运行过程中存在一定的不确定因素，甚至有时会发生改变。在Word中无法直接写入数据信息，不仅不安全，而且应用程序的预期功能也很难实现。要想达到所需的文档存储功能，专业的数据库存储软件是必须要选择的。尽管大多数数据库软件能够满足应用程序的需求，但MySQL数据库因其小巧的安装包、快速的安装过程和便捷的使用体验脱颖而出。即使安装时出现问题，也不需要重新安装操作系统，而不会影响其他第三方软件在计算机上运行，网络资源消耗更少[19]。最关键的是，MySQL的功能与设计需求完全吻合，因此应用软件开发最终确定采用MySQL数据库作为核心数据库。

## 2.2Python语言

Python是一种解释型脚本语言，程序在运行过程中会被转化为字节码和机器语言。这意味着，Python程序不需要在运行前进行编译，而是通过专门的解释器逐步翻译执行。与此相对的是编译型语言，它们需要先经过编译生成可执行文件[20]。

同时，这是一种跨平台的编程语言，结合了编译、交互和面向对象的特性，是一种脚本语言（script language）。

（1）首先，它是一种简洁易学的编程语言，对于初学者来说也非常容易上手。与其他编程语言相比，编写和阅读代码时更贴近人类的思维方式，具有较强的可读性和易于理解的特点。

（2）由于其底层逻辑是用C语言实现的，因此在运行速度上表现得非常迅速。正如许多开发者所了解的，许多标准库及第三方依赖都是基于C语言开发的，因此其执行效率较高。

（3）它始终是一种面向对象的程序设计语言，既支持面向过程编程，也支持面向对象编程。在面向过程的语言中，程序是建立在一个过程或简单的可重用代码功能之上的；而在面向对象的编程中，程序则是基于数据和函数构造的对象体系[21]。

（4）在需要扩展代码或快速查看时，其格式清晰且易于维护。在不同开发人员协作的过程中，能够更方便地理解和使用所编写的内容。

（5）此外，由于其开源特性，它可以轻松移植到多种平台上使用。同时，语言本身完全支持重载、派生和继承等特性，这些都极大增强了代码的复用性。此外，还有两个标准库（functools 和 itertools）为开发提供了有力支持。

## 2.3Django框架

Django是基于Python语言开发的框架，采用了MVT（模型-视图-模板）架构[22]。它将Python中一些繁琐的操作进行封装，使得开发者能够更高效地调用，从而显著缩短开发周期。

## 2.4微信开发者工具

目前腾讯发布的微信官方网站上所使用的软件开发工具主要是微信开发工具，并且有官方网站为它提供了一个前端开发平台。由于软件不能在浏览器上运行，因此在网络上是不允许使用Document，Window等方法的。而在以往对HTML5进行加载的过程中，会受到网络环境的限制，需要对HTML、CSS以及JS进行一步又一步的加载，然后返回信息，从而渲染页面，将其在浏览器当中进行呈现[23]。这样的方式用户需要等待许久，而且会降低用户的体验感。微信Web开发者工具与H5开发工具的预览模式存在着一定的差异，其属于一项自己的工具，因此可有效地实现同步本地文件开发和调试等流程。此外小程序开发了相关的标签语言和样式语言，并没有使用相关的HTML5+CSS3。在微信小程序软件的开发过程中，还进一步提供了原生APP的组件。而在HTML5+CSS3当中，这是需要通过模拟才能够有效实现的相关功能，但在小程序当中可以通过组件来得到有效的实现[24]。

## 2.5小程序目录结构以及框架介绍

一个小程序主要由位于项目根目录的三个文件构成主体部分。其中，app.js是小程序逻辑的载体，必不可少；app.json负责小程序的公共配置，同样不可或缺；app.wxss用于设置小程序公共样式表，为非必需文件。而一个小程序页面通常由四个文件组成，js文件掌控页面逻辑，wxml决定页面结构，二者是必需的，json用于页面配置，wxss负责页面样式表，这两者均为可选。从框架层面来看，小程序架构涵盖逻辑层和视图层[25]。逻辑层AppService主要运用Pyton语言编写，承担着业务流程控制的重任，具备数据初始化、云端存储与调用、添加待处理逻辑事务、执行逻辑交易以及展示结果等功能[26]。视图层View则通过WXML与WXSS编写，借助组件进行展示，其作用是将逻辑层的数据呈现为视图，同时把视图层的事件传递给逻辑层，以此实现数据传递和事件处理，让开发者能够聚焦于数据和逻辑层面[27]。总的来说，小程序凭借这样的目录结构和框架，为开发者打造出高效便捷的开发环境。

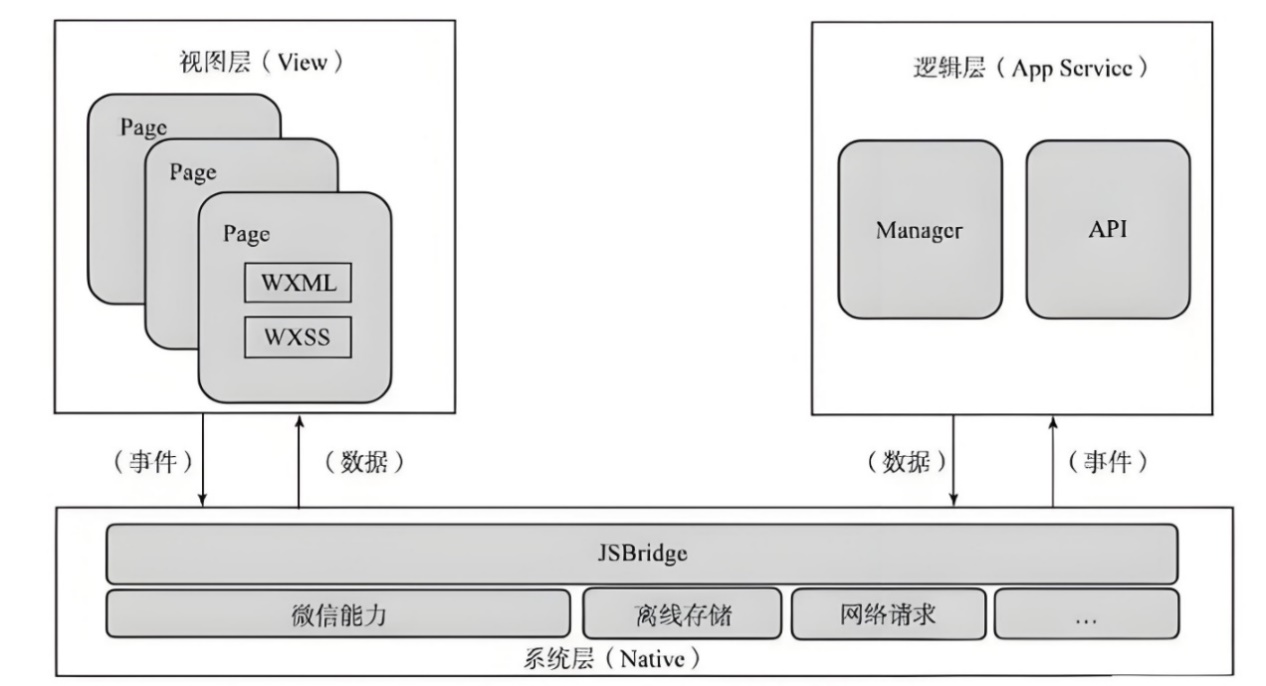


图2-1 微信小程序的结构

## 2.6本章小结

MySQL数据库因安装便捷、资源消耗低、功能适配，成为本系统核心数据库。Python语言凭借简洁易学、运行高效、面向对象、易维护及开源等特性，为系统开发提供有力支持。Django框架基于Python，采用MVT架构，提升开发效率。微信开发者工具助力小程序开发，其独特框架和目录结构，使逻辑层与视图层协同工作，为系统开发奠定坚实技术基础。

# 3 系统分析

需求分析就是研究、分析之后研发人员对用户需求有准确的了解，并对系统所需要的功能加以确定。然后对系统进行可行性及功能分析，保证后续系统创建有显著帮助，从而满足本系统设计的实现价值。

## 3.1可行性分析

### 3.1.1技术可行性分析

对于本系统的开发，所涉及的开发工具与技术在开发者的知识储备与实践经验范围内具备可操作性。开发过程中常用的软件，如集成开发环境、应用服务器以及数据库管理系统等，开发者已在过往学习和实践中深入接触，并熟练掌握了其基本操作与应用技巧。在技术层面，本系统所依赖的核心技术，包括主流编程语言、架构模式，以及前端技术等，均有丰富的学习资源可供参考。图书馆藏有大量与之相关的专业书籍，这些书籍对各项技术的原理、应用场景及实践案例进行了全面且深入的讲解，为开发者学习和掌握这些技术提供了坚实的理论基础。同时，在日常编程实践项目以及课程设计作业中，老师对这些技术进行了细致的教学和指导，开发者不仅积累了实际编写代码的经验，还在不断解决问题的过程中提升了自己的编程思维与调试能力。因此，基于现有的技术知识储备、丰富的学习资源以及一定的实践经验，开发者有充分的信心能够运用这些技术完成系统的编程开发工作。

### 3.1.2经济可行性分析

从经济成本角度来看，本系统的开发与运行具备较高的可行性。该系统主要面向特定的应用场景，数据管理规模相对可控，无需处理海量的并发数据请求，这意味着在系统资源配置方面的要求并不苛刻。普通计算机设备即可作为系统客户端，满足日常使用需求，无需额外购置高性能、高成本的硬件设备。在软件开发成本上，项目所使用的所有开发工具与技术均为开源资源，无需支付任何软件授权费用。这使得在系统开发过程中，能够将全部精力集中于功能实现与优化，无需担忧高昂的软件购买成本。同时，系统投入运行后的维护费用也相对较低，主要集中在服务器的日常运维与数据存储管理方面，不会给开发者或使用方带来沉重的经济负担。此外，系统的应用能够显著提升资源的利用效率。与传统方式相比，它减少了某些资源的消耗，避免了大量人力在相关事务处理过程中的浪费，从长期来看，能够为使用方节省可观的资源成本，具有良好的经济效益与资源节约效益。

### 3.1.3法律可行性分析

本系统是开发者独立设计开发的项目，在开发过程中严格遵循知识产权相关法律法规，不存在侵犯他人、集体或国家利益的情况。系统开发所采用的软件工具均为正版授权或开源免费资源，所有参考资料均来源于正规、合法的渠道，通过对专业技术网站、开源社区以及学术数据库的查询与分析获取，完全符合国家法律法规的要求。在整个开发过程中，开发者始终秉持合法合规的原则，确保系统的架构设计、功能实现以及数据处理等各个环节均不违背国家的方针政策与法律规定。因此，从法律层面来看，本系统的开发、部署与使用具备充分的可行性与合法性保障。

## 3.2系统性能分析

### 3.2.1系统安全性

系统安全在本系统的应用场景中是至关重要的保障。在资料存储方面，需采用先进的加密技术对系统资料进行加密处理，防止数据在存储过程中被非法窃取或篡改。例如，对于用户的敏感信息，如账号密码、个人隐私等，运用对称加密算法将其转换为密文存储，只有经过授权的操作才能进行解密读取。资料传送过程的安全同样不可忽视。通过建立安全的传输通道，如采用SSL/TLS协议对数据传输进行加密，确保数据在网络传输过程中的完整性和保密性。同时，设置防火墙等安全设备，对外部网络的访问进行严格过滤，阻止非法的网络攻击。合理的用户权限管理也是系统安全的关键。根据用户的角色和职责，为其分配不同的操作权限，避免越权操作。例如，普通用户只能进行数据查询和简单的操作，而管理员则拥有更高的权限，可以进行系统配置和数据管理等操作。当出现系统故障、数据损坏等事故导致系统资料缺损时，完善的历史数据备份机制能够快速还原数据，确保系统的正常运行。

### 3.2.2可维护性和适应性

随着时代的发展和行业的进步，本系统可能会面临新的需求和挑战。一个具有良好可维护性的系统，其代码结构应清晰，注释详细，便于开发人员进行代码的理解和修改。同时，采用模块化设计思想，将系统划分为多个独立的模块，每个模块负责特定的功能，当需要对系统进行功能扩展或修改时，只需对相应的模块进行调整，而不会影响到其他模块的正常运行。系统的适应性则体现在能够快速响应市场变化和用户需求的能力上。通过建立灵活的配置机制，系统可以根据不同的业务需求进行灵活配置，无需对代码进行大规模修改。例如，当业务规则发生变化时，只需在系统配置中进行相应的调整，即可实现新的业务逻辑。这样，本系统无论是在当前还是未来，都能够长期稳定地使用，持续满足用户不断变化的需求。

### 3.2.3可靠性

随着系统的不断发展，用户数量会逐渐增多，系统的访问量也会大幅增加。为了确保系统在高访问量情况下依然能够保持优越的运行速度和容错能力，需要采用一系列的技术手段。例如，采用分布式架构，将系统的负载均衡到多个服务器上，避免单点故障，提高系统的并发处理能力。同时，使用缓存技术，将常用的数据存储在缓存中，减少对数据库的访问次数，提高系统的响应速度。此外，系统还应具备完善的容错机制。当出现硬件故障、网络中断等异常情况时，系统能够自动进行故障转移和恢复，确保服务的连续性。通过定期进行性能测试和压力测试，及时发现系统中的性能瓶颈和潜在问题，并进行优化和改进，以保障系统在高访问量的情况下依然具备足够的可靠性，为用户提供稳定、高效的服务。

## 3.3功能需求分析

### 3.3.1微信小程序用户端功能需求

（1）首页：展示系统发布的最新公告和通知，包括直播活动预告、商品促销信息等。公告内容应按发布时间排序，最新的公告显示在最上方。用户点击公告标题可查看公告详情。

（2）商品信息：展示直播带货商品的详细信息，包括商品名称、价格、规格、图片、库存等。支持按商品类别、价格区间、销量等条件进行筛选和排序。提供商品搜索功能，用户可通过关键词搜索所需商品。点击商品可查看商品详情页，包括商品介绍、用户评价等。

（3）留言反馈：用户可在此模块向管理员或商家发送留言反馈，包括对商品的意见、对直播的建议等。留言内容应包含用户昵称、留言时间、留言内容等信息。支持用户查看自己的留言历史记录。

（4）购物车：用户可将感兴趣的商品加入购物车，购物车中显示商品名称、价格、数量、小计等信息。用户可在购物车中修改商品数量、删除商品。提供购物车商品总价计算功能。用户可在购物车中结算商品，生成订单。

（5）我的：直播带货：展示用户参与过的直播带货活动记录，包括直播时间、商品名称、购买数量等。充值记录：显示用户的充值历史记录，包括充值时间、充值金额、充值方式等。我的订单：展示用户的订单列表，包括订单编号、订单状态、商品名称、订单金额等信息。用户可查看订单详情、取消未支付订单、申请退款等。充值：提供多种充值方式，如微信支付、支付宝支付等。用户可输入充值金额进行充值操作。我的收藏：显示用户收藏的商品列表，用户可在收藏列表中快速查看和购买收藏的商品。修改密码：用户可在此模块修改自己的登录密码，需输入原密码、新密码和确认新密码进行验证。

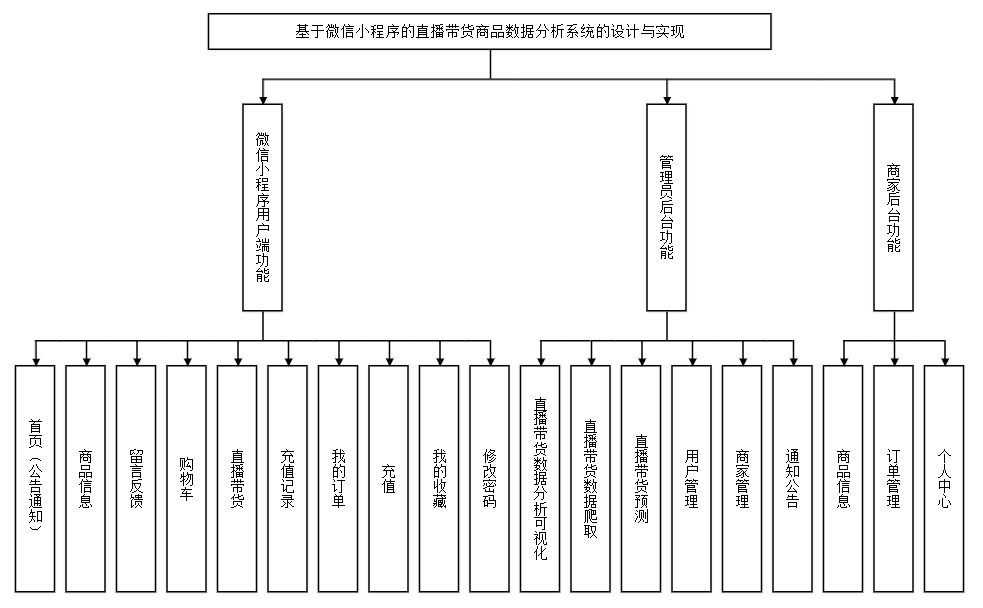


图3.1 系统功能模块结构

### 3.3.2管理员后台功能需求

（1）直播带货数据分析可视化：运用机器学习中的神经网络算法对直播带货数据进行分析，通过调整 solver、random\_state、hidden\_layer\_sizes 等参数，提高分析的准确性。将分析结果以可视化的方式展示，如柱状图、折线图、饼图等，直观呈现商品销量、销售额、用户参与度等数据。提供数据筛选和查询功能，管理员可按时间、商品类别、直播活动等条件筛选数据进行分析。

（2）直播带货（数据爬取）：使用爬虫技术从各大电商平台、直播平台爬取直播带货相关数据，包括商品信息、用户评价、销售数据等。对爬取的数据进行清洗和预处理，去除重复数据、无效数据等。将处理后的数据存储到 MySQL 数据库中，以便后续分析和使用。

（3）直播带货预测：基于历史直播带货数据，使用神经网络算法进行预测，调整相关参数以提高预测的准确性。预测内容包括商品销量、销售额、用户参与度等，为直播带货活动提供决策支持。以可视化的方式展示预测结果，如趋势图、预测值等。

（4）用户管理：管理微信小程序用户的信息，包括用户注册、登录、注销等操作。查看用户的基本信息、消费记录、留言反馈等。对违规用户进行封禁、警告等处理。

（5）商家管理：管理商家的信息，包括商家注册、审核、注销等操作。查看商家的基本信息、商品信息、订单信息等。对违规商家进行处罚、下架商品等处理。

（6）通知公告：发布系统公告和通知，包括直播活动预告、商品促销信息等。管理公告和通知的发布时间、内容、状态等。可对已发布的公告和通知进行编辑、删除等操作。

### 3.3.3商家后台功能需求

（1）商品信息：管理商家自己的商品信息，包括商品添加、编辑、删除等操作。上传商品图片、填写商品名称、价格、规格、库存等信息。对商品进行分类管理，设置商品的推荐状态、促销活动等。

（2）订单管理：查看商家的订单列表，包括订单编号、订单状态、商品名称、订单金额等信息。处理订单，包括发货、退款、售后等操作。统计订单数据，如订单数量、销售额、客户评价等。

（3）个人中心：展示商家的基本信息，包括商家名称、联系方式、注册时间等。商家可在此模块修改自己的基本信息和登录密码。查看商家的账户余额、充值记录、提现记录等。

## 3.4本章小结

从技术、经济、法律三方面看，本系统开发可行。技术上开发者经验丰富且有学习资源；经济上成本低且能增效；法律上遵循法规使用正版或开源资源。系统注重安全性，采用加密、权限管理等措施；具备良好可维护性与适应性，代码结构清晰且配置灵活；可靠性高，通过分布式架构等技术保障运行。此外，明确了用户端、管理员后台、商家后台的功能需求，为系统设计提供方向。

# 4系统设计

## 4.1整体结构设计

本系统采用前后端分离的架构模式，这种架构模式能够有效提高系统的可维护性、可扩展性和开发效率，使得前端和后端的开发工作可以独立进行，互不干扰。

前端部分基于微信小程序进行开发，微信小程序作为一种轻量级的应用程序，具有无需下载安装、即开即用的特点，能够为用户提供便捷的访问体验。微信小程序采用了类似于Web开发的技术栈，包括WXML（WeiXinMarkupLanguage）、WXSS（WeiXinStyleSheets）和Python，这使得前端开发人员可以利用现有的前端开发技能进行开发。

## 4.2数据库设计

开发一个高效的系统，建立一个性能优良的数据库是关键。这个数据库需具备强大的数据容纳能力，能支持多种类型的数据存储。同时，要运用先进的存储技术，有效降低数据冗余率，为各类应用程序（AP）提供更可靠的服务。数据库在信息存储领域起着至关重要的作用，它在程序间能够保持较高的独立性。随着技术的不断发展，数据库凭借其良好的理论基础，成为了备受青睐的存储工具。

### 4.2.1数据库概念设计

在数据库构建过程中，除了遵循必要的步骤，还可借助Visio等图形化工具，清晰展示复杂的网络结构及各部分之间的联系，以更好地满足系统在功能和性能方面的需求。构建数据库时，应紧密结合系统的架构、功能和性能特点，精心设计出适配系统的网络结构。实体-关系图（E-R图）是一种有效的图形化处理技术。无论是使用亿图软件还是Visio工具，它们都采用相同的符号体系来描述实体间的相互关系。其中，矩形代表实体，菱形表示实体之间的联系，而实体的属性则用椭圆来表示。通过绘制E-R图，将矩形、菱形、椭圆等几何图形有机组合，实现对数据库概念结构的清晰呈现。

### 4.2.2数据库主要表E-R图

在数据库设计中，包含一系列紧密关联的数据表。购物车表，主键为id，记录用户购物车信息，如商品表名、用户与商品id、商品详情、购买数量及商户名称等，支撑用户购物车功能。直播带货预测表，以id为主键，存储创建时间、标题、作者昵称、时长及收藏、点赞数，助力直播带货预测分析。

商品信息表的主键是id，全面涵盖商品名称、类型、图片、产地、品牌等基础信息，以及上架时间、详情、商家信息，还有库存、评分等状态数据，为商品管理提供核心依据。直播带货表，主键id关联直播创建时间、标题、作者昵称、图片、时长、互动数据及详情地址，记录直播带货全过程数据。

商家表与用户表分别以id为主键，前者存储商家账号、密码、姓名、联系方式、头像及余额，后者记录用户账号、密码、个人信息及余额，用于商家与用户的基本信息管理。订单表，主键id串联订单编号、用户与商品id、商品详情、购买详情、支付与物流信息、审核状态及退货原因等，是订单业务流转的数据载体。收藏表，主键id关联用户与商品id、表名、名称等，管理用户收藏数据。通知公告表，主键id包含标题、简介、发布人、互动数据及内容等，实现通知公告的发布与管理。

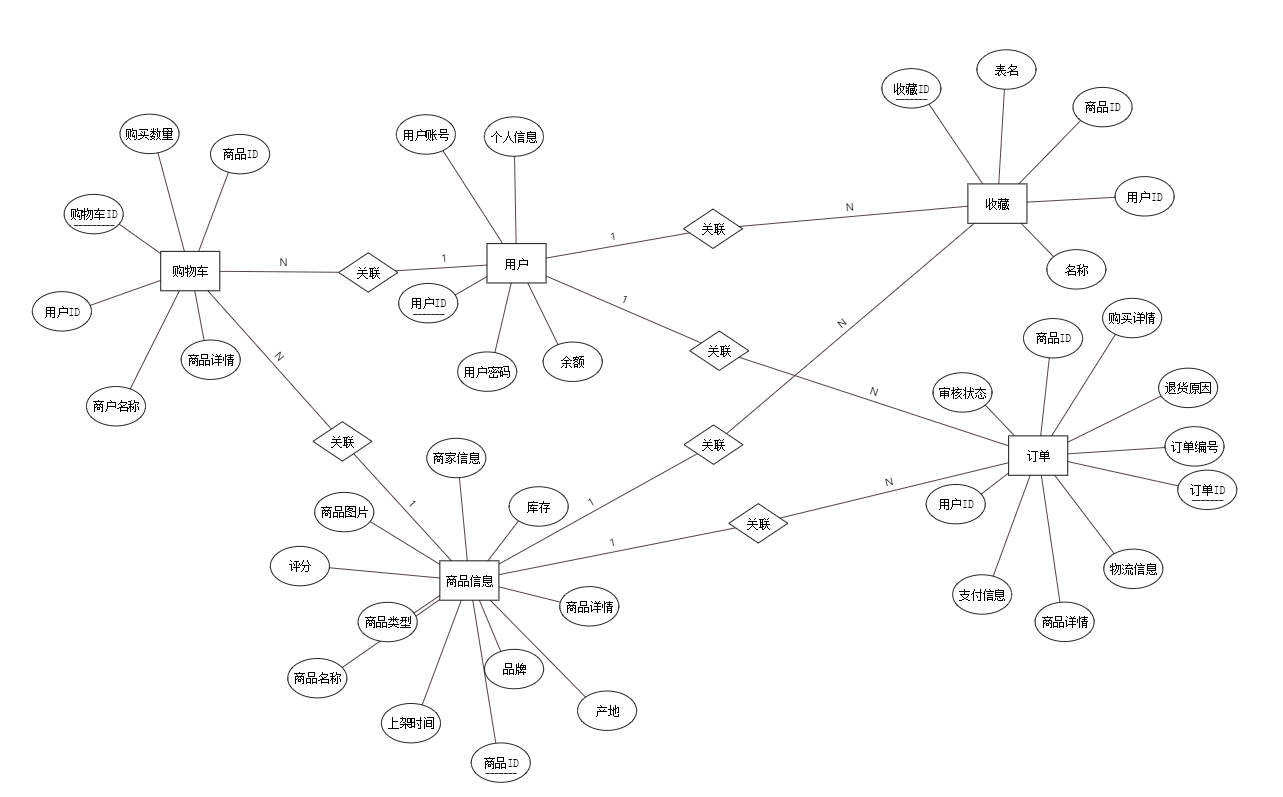


图4.1 数据库E-R图

### 4.2.3数据库表设计

表4.1 表名：cart

功能：购物车表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 字段说明 | 主键 | 默认值 |
| id | bigint |  | 主键 | 主键 |  |
| addtime | timestamp |  | 创建时间 |  |  |
| tablename | varchar | 200 | 商品表名 |  |  |
| userid | bigint |  | 用户id |  |  |
| goodid | bigint |  | 商品id |  |  |
| goodname | varchar | 200 | 商品名称 |  |  |
| picture | longtext | 4294967295 | 图片 |  |  |
| buynumber | int |  | 购买数量 |  |  |
| price | double |  | 单价 |  |  |
| shangjiazhanghao | varchar | 200 | 商户名称 |  |  |

表4.2 表名：zhibodaihuoforecast

功能：直播带货预测

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 字段说明 | 主键 | 默认值 |
| id | bigint |  | 主键 | 主键 |  |
| addtime | timestamp |  | 创建时间 |  |  |
| title | varchar | 200 | 标题 |  |  |
| nickname | varchar | 200 | 作者昵称 |  |  |
| duration | int |  | 时长 |  |  |
| collectcount | int |  | 收藏数 |  |  |
| diggcount | int |  | 点赞数 |  |  |

表4.3 表名：shangpinxinxi

功能：商品信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 字段说明 | 主键 | 默认值 |
| id | bigint |  | 主键 | 主键 |  |
| addtime | timestamp |  | 创建时间 |  |  |
| shangpinmingcheng | varchar | 200 | 商品名称 |  |  |
| shangpinleixing | varchar | 200 | 商品类型 |  |  |
| shangpintupian | longtext | 4294967295 | 商品图片 |  |  |
| shangguiguige | varchar | 200 | 商规规格 |  |  |
| shangpinchandi | varchar | 200 | 商品产地 |  |  |
| shangpinpinpai | varchar | 200 | 商品品牌 |  |  |
| shangjiashijian | date |  | 上架时间 |  |  |
| shangpinxiangqing | longtext | 4294967295 | 商品详情 |  |  |
| shangjiazhanghao | varchar | 200 | 商家账号 |  |  |
| shangjiaxingming | varchar | 200 | 商家姓名 |  |  |
| onelimittimes | int |  | 单限 |  |  |
| alllimittimes | int |  | 库存 |  |  |
| discussnum | int |  | 评论数 |  | 0 |
| totalscore | double |  | 评分 |  | 0 |
| price | double |  | 价格 |  |  |
| onshelves | int |  | 是否上架(1:上架，0:下架) |  | 1 |
| storeupnum | int |  | 收藏数 |  | 0 |

表4.4 表名：zhibodaihuo

功能：直播带货

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 字段说明 | 主键 | 默认值 |
| id | bigint |  | 主键 | 主键 |  |
| addtime | timestamp |  | 创建时间 |  |  |
| title | longtext | 4294967295 | 标题 |  |  |
| nickname | varchar | 200 | 作者昵称 |  |  |
| imgurl | longtext | 4294967295 | 图片 |  |  |
| duration | int |  | 时长 |  |  |
| ratio | varchar | 200 | 分辨率 |  |  |
| collectcount | int |  | 收藏数 |  |  |
| commentcount | int |  | 评论数 |  |  |
| diggcount | int |  | 点赞数 |  |  |
| sharecount | int |  | 分享数 |  |  |
| cjtime | varchar | 200 | 创建时间 |  |  |
| detailurl | longtext | 4294967295 | 详情地址 |  |  |

表4.5 表名：shangjia

功能：商家

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 字段说明 | 主键 | 默认值 |
| id | bigint |  | 主键 | 主键 |  |
| addtime | timestamp |  | 创建时间 |  |  |
| shangjiazhanghao | varchar | 200 | 商家账号 |  |  |
| mima | varchar | 200 | 密码 |  |  |
| shangjiaxingming | varchar | 200 | 商家姓名 |  |  |
| xingbie | varchar | 200 | 性别 |  |  |
| shouji | varchar | 200 | 手机 |  |  |
| touxiang | longtext | 4294967295 | 头像 |  |  |
| money | double |  | 余额 |  | 0 |

表4.6 表名：yonghu

功能：用户

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 字段说明 | 主键 | 默认值 |
| id | bigint |  | 主键 | 主键 |  |
| addtime | timestamp |  | 创建时间 |  |  |
| yonghuzhanghao | varchar | 200 | 用户账号 |  |  |
| mima | varchar | 200 | 密码 |  |  |
| yonghuxingming | varchar | 200 | 用户姓名 |  |  |
| xingbie | varchar | 200 | 性别 |  |  |
| shouji | varchar | 200 | 手机 |  |  |
| touxiang | longtext | 4294967295 | 头像 |  |  |
| money | double |  | 余额 |  | 0 |

表4.7 表名：orders

功能：订单

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 字段说明 | 主键 | 默认值 |
| id | bigint |  | 主键 | 主键 |  |
| addtime | timestamp |  | 创建时间 |  |  |
| orderid | varchar | 200 | 订单编号 |  |  |
| tablename | varchar | 200 | 商品表名 |  | shangpinxinxi |
| userid | bigint |  | 用户id |  |  |
| goodid | bigint |  | 商品id |  |  |
| goodname | varchar | 200 | 商品名称 |  |  |
| picture | longtext | 4294967295 | 商品图片 |  |  |
| buynumber | int |  | 购买数量 |  |  |
| price | double |  | 价格 |  | 0 |
| total | double |  | 总价格 |  | 0 |
| type | int |  | 支付类型 |  | 1 |
| status | varchar | 200 | 状态 |  |  |
| address | varchar | 200 | 地址 |  |  |
| tel | varchar | 200 | 电话 |  |  |
| consignee | varchar | 200 | 收货人 |  |  |
| logistics | longtext | 4294967295 | 物流 |  |  |
| remark | varchar | 200 | 备注 |  |  |
| shangjiazhanghao | varchar | 200 | 商户名称 |  |  |
| sfsh | varchar | 200 | 是否审核 |  |  |
| shhf | longtext | 4294967295 | 审核回复 |  |  |
| role | varchar | 200 | 用户角色 |  |  |
| returnreason | varchar | 200 | 退货原因 |  |  |

表4.8 表名：storeup

功能：收藏表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 字段说明 | 主键 | 默认值 |
| id | bigint |  | 主键 | 主键 |  |
| addtime | timestamp |  | 创建时间 |  |  |
| userid | bigint |  | 用户id |  |  |
| refid | bigint |  | 商品id |  |  |
| tablename | varchar | 200 | 表名 |  |  |
| name | varchar | 200 | 名称 |  |  |
| picture | longtext | 4294967295 | 图片 |  |  |
| type | varchar | 200 | 类型 |  | 1 |
| inteltype | varchar | 200 | 推荐类型 |  |  |
| remark | varchar | 200 | 备注 |  |  |

表4.9 表名：news

功能：通知公告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 字段说明 | 主键 | 默认值 |
| id | bigint |  | 主键 | 主键 |  |
| addtime | timestamp |  | 创建时间 |  |  |
| title | varchar | 200 | 标题 |  |  |
| introduction | longtext | 4294967295 | 简介 |  |  |
| typename | varchar | 200 | 分类名称 |  |  |
| name | varchar | 200 | 发布人 |  |  |
| headportrait | longtext | 4294967295 | 头像 |  |  |
| clicknum | int |  | 点击次数 |  | 0 |
| clicktime | datetime |  | 最近点击时间 |  |  |
| thumbsupnum | int |  | 赞 |  | 0 |
| crazilynum | int |  | 踩 |  | 0 |
| storeupnum | int |  | 收藏数 |  | 0 |
| picture | longtext | 4294967295 | 图片 |  |  |
| content | longtext | 4294967295 | 内容 |  |  |

## 4.3本章小结

系统采用前后端分离架构，前端基于微信小程序开发，为用户提供便捷体验，后端负责业务逻辑与数据存储。数据库设计是重点，通过概念设计明确结构，利用E-R图展示实体关系，详细设计各数据表，包括购物车、商品信息等表，确定字段属性，保障数据管理高效有序，提升系统整体性能。

# 5系统实现

## 5.1系统开发环境以及运行环境

本系统设计借助先进工具与环境，实现高效开发与稳定运行。开发工具上，后端选PyCharm，其智能代码补全、代码分析及调试功能强大。

## 5.2管理员后台设计与实现

### 5.2.1登录设计与实现

管理员输入账号和密码登录系统，系统验证账号密码的有效性，若验证通过则进入管理员后台主界面；若验证失败则提示重新输入。

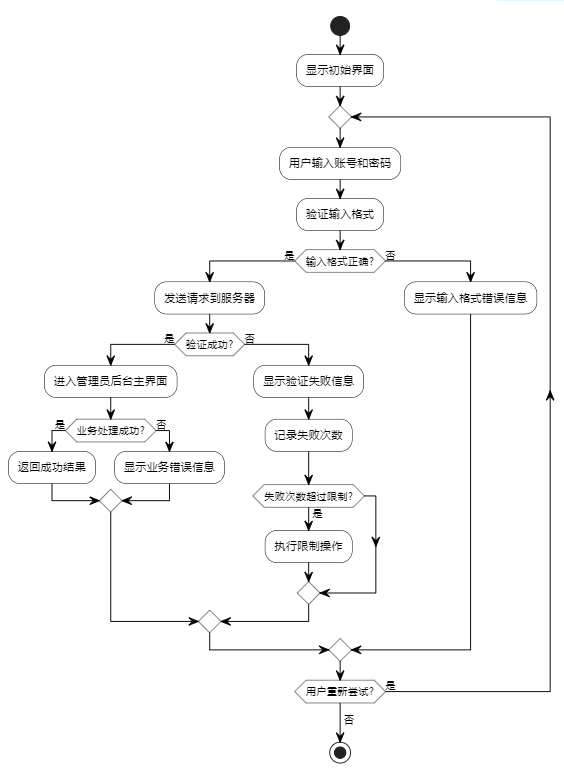


图5.1 登录流程

使用Python的Flask或Django框架搭建后端服务器，创建用户登录接口。前端页面使用HTML、CSS和JavaScript实现登录表单，用户输入账号密码后通过AJAX请求将数据发送到后端接口进行验证。

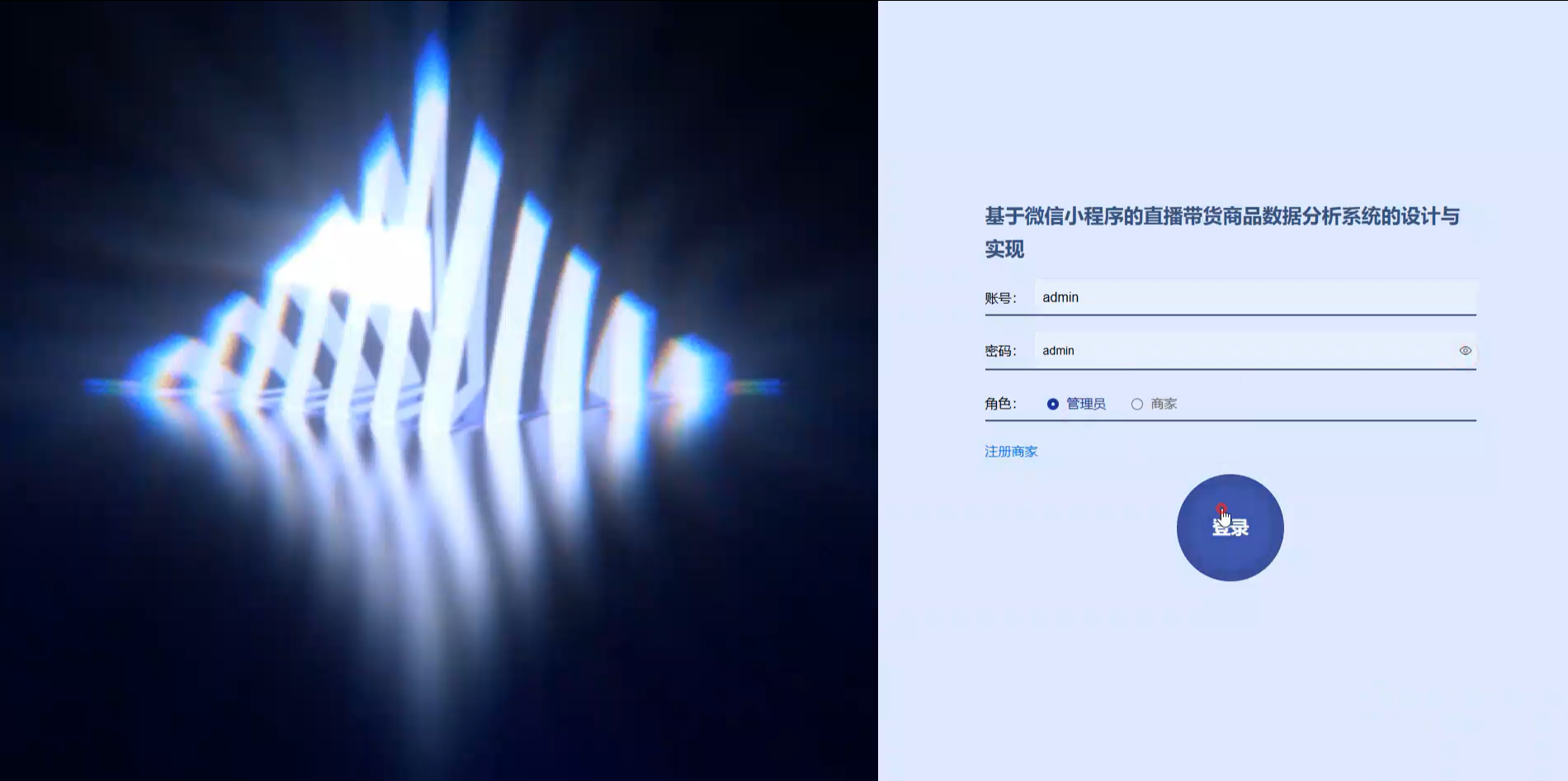


图5.2 登录界面

### 5.2.2数据分析与可视化设计与实现

系统从数据库中提取直播带货相关数据，使用机器学习的神经网络算法对数据进行分析，调整solver、random\_state、hidden\_layer\_sizes等参数以提高分析准确性，最后将分析结果以柱状图、折线图、饼图等可视化方式展示在界面上。

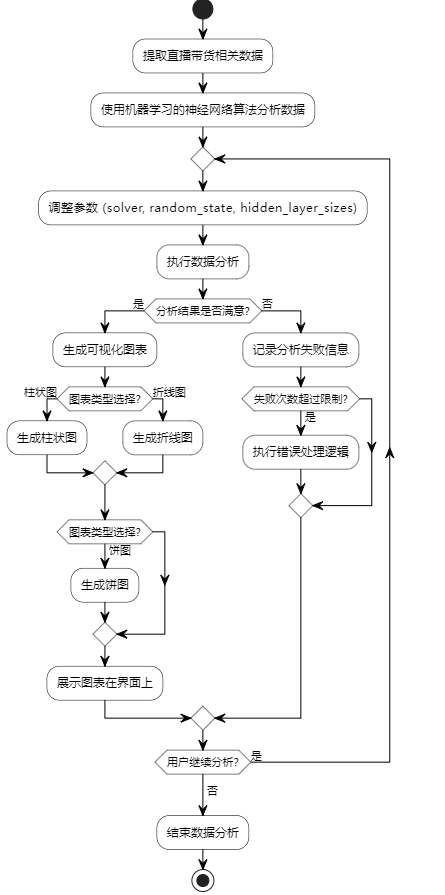


图5.3 数据分析与可视化流程

使用Python的scikit-learn库实现神经网络算法，通过循环调整solver、random\_state、hidden\_layer\_sizes等参数，找到最优参数组合。使用matplotlib或seaborn库将分析结果进行可视化展示。



图5.4 可视化实现

（1）直播收藏数可视化实现

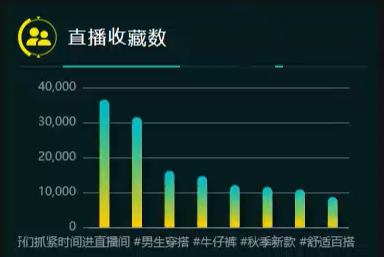


图5.5 直播收藏数实现图

图5.5中显示直播收藏数的柱状图，用于分析不同直播标题对应的收藏数量情况。用户在页面操作触发相关事件后，即可查看各直播标题的收藏数情况。鼠标悬停于柱状图上，可根据设置的tooltip查看具体的收藏数情况。

初始化图表：使用echarts.init方法初始化一个macarons主题的柱状图对象。

javascript

var collectcountChart2 = echarts.init(document.getElementById("collectcountChart2"),'macarons');

请求数据：通过this.$http发送GET请求获取直播收藏数的数据。

javascript

this.$http({

url: `zhibodaihuo/value/title/collectcount`,

method: "get",

}).then(({ data }) => {

// 处理数据

});

处理数据：将返回的数据中的标题和收藏数分别存储在xAxis和yAxis数组中。

javascript

let res = data.data;

let xAxis = [];

let yAxis = [];

let pArray = [];

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

if(this.boardBase&&i==this.boardBase.barNum){

break;

}

xAxis.push(res[i].title);

yAxis.push(parseFloat((res[i].total)));

pArray.push({

value: parseFloat((res[i].total)),

name: res[i].title

})

}

配置图表选项：设置图表的标题、图例、坐标轴、系列等选项。

javascript

let titleObj = this.bar.title;

titleObj.text = '直播收藏数';

const legendObj = this.bar.legend;

let tooltipObj = {trigger: 'item',formatter: '{b} : {c}'};

tooltipObj = Object.assign(tooltipObj , this.bar.tooltip?this.bar.tooltip:{});

let xAxisObj = this.bar.xAxis;

xAxisObj.type = 'category';

xAxisObj.data = xAxis;

let yAxisObj = this.bar.yAxis;

yAxisObj.type = 'value';

let seriesObj = {

data: yAxis,

type: 'bar',

};

seriesObj = Object.assign(seriesObj , this.bar.series);

const gridObj = this.bar.grid;

option = {

backgroundColor: this.bar.backgroundColor,

color: this.bar.color,

title: titleObj,

legend: legendObj,

tooltip: tooltipObj,

grid: gridObj,

xAxis: xAxisObj,

yAxis: yAxisObj,

series: [seriesObj]

};

显示图表：使用setOption方法将配置项应用到图表上，并监听窗口大小变化，动态调整图表大小。

javascript

collectcountChart2.setOption(option);

window.onresize = function() {

collectcountChart2.resize();

};

结果分析说明：通过直播收藏数的柱状图，可以直观地看出不同直播标题的收藏数差异。柱状图较高的直播标题说明其收藏数较多，该直播的内容、主播等因素更受观众喜爱。可以根据这些数据，对收藏数高的直播进行分析，总结成功经验，应用到后续的直播策划中；对于收藏数低的直播，可以考虑改进直播内容、推广方式等，以提高直播的吸引力和收藏数。

（2）直播分享数可视化实现

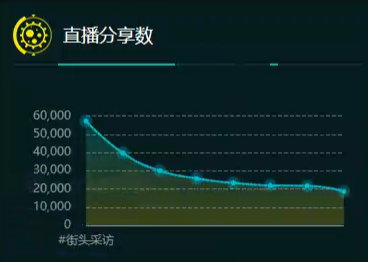


图5.6 直播分享数实现图

图5.6中显示直播分享数折线图，用于分析不同直播标题对应的分享数量随不同直播商家的变化情况。用户在页面操作触发相关事件后，即可查看各直播标题的分享数变化情况。鼠标悬停于折线图上，可根据设置的tooltip查看具体的分享数情况。

初始化图表：使用echarts.init方法初始化一个macarons主题的折线图对象。

javascript

var sharecountChart5 = echarts.init(document.getElementById("sharecountChart5"),'macarons');

请求数据：通过this.$http发送GET请求获取直播分享数的数据。

javascript

this.$http({

url: `zhibodaihuo/value/title/sharecount`,

method: "get",

}).then(({ data }) => {

// 处理数据

});

处理数据：将返回的数据中的标题和分享数分别存储在xAxis和yAxis数组中。

javascript

let res = data.data;

let xAxis = [];

let yAxis = [];

let pArray = [];

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

if(this.boardBase&&i==this.boardBase.lineNum){

break;

}

xAxis.push(res[i].title);

yAxis.push(parseFloat((res[i].total)));

pArray.push({

value: parseFloat((res[i].total)),

name: res[i].title

})

}

配置图表选项：设置图表的标题、图例、坐标轴、系列等选项。

javascript

let titleObj = this.line.title;

titleObj.text = '直播分享数';

const legendObj = this.line.legend;

let tooltipObj = { trigger: 'item',formatter: '{b} : {c}'};

tooltipObj = Object.assign(tooltipObj , this.line.tooltip?this.line.tooltip:{});

let xAxisObj = this.line.xAxis;

xAxisObj.type = 'category';

xAxisObj.boundaryGap = false;

xAxisObj.data = xAxis;

let yAxisObj = this.line.yAxis;

yAxisObj.type = 'value';

let seriesObj = {

data: yAxis,

type: 'line',

areaStyle: {},

smooth: true

};

seriesObj = Object.assign(seriesObj , this.line.series);

const gridObj = this.line.grid;

option = {

backgroundColor: this.line.backgroundColor,

color: this.line.color,

title: titleObj,

legend: legendObj,

tooltip: tooltipObj,

xAxis: xAxisObj,

yAxis: yAxisObj,

series: [seriesObj],

grid: gridObj

};

显示图表：使用setOption方法将配置项应用到图表上，并监听窗口大小变化，动态调整图表大小。

javascript

sharecountChart5.setOption(option);

window.onresize = function() {

sharecountChart5.resize();

};

结果分析说明：通过直播分享数折线图，可以观察到不同直播标题的分享数的变化趋势。如果折线呈上升趋势，说明该直播的分享数在增加，直播内容逐渐受到更多人的关注和认可；如果折线呈下降趋势，则需要分析原因，是否是直播内容质量下降、竞争对手出现等因素导致。同时，对比不同折线的高低，可以看出哪些直播的分享数整体较高，哪些较低，以便对直播进行针对性的优化和推广。

（3）作者直播时长可视化实现

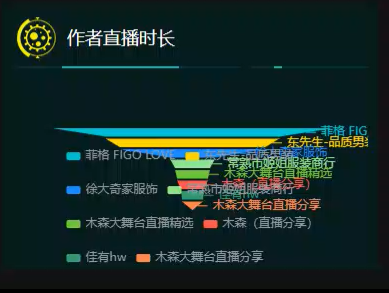


图5.7 作者直播时长实现

图5.7中显示作者直播时长漏斗图，用于分析不同作者的直播时长占比情况。用户在页面操作触发相关事件后，即可查看各作者的直播时长占比情况。鼠标悬停于漏斗图上，可根据设置的tooltip查看具体的直播时长和占比情况。

实现过程：初始化图表：使用echarts.init方法初始化一个macarons主题的漏斗图对象。

javascript

var durationChart6 = echarts.init(document.getElementById("durationChart6"),'macarons');

请求数据：通过this.$http发送GET请求获取作者直播时长的数据。

javascript

this.$http({

url: `zhibodaihuo/value/nickname/duration`,

method: "get",

}).then(({ data }) => {

// 处理数据

});

处理数据：将返回的数据中的作者昵称和直播时长分别存储在xAxis和yAxis数组中，并将数据整理成适合漏斗图的格式。

javascript

let res = data.data;

let xAxis = [];

let yAxis = [];

let pArray = [];

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

if(this.boardBase&&i==this.boardBase.funnelNum){

break;

}

xAxis.push(res[i].nickname);

yAxis.push(parseFloat((res[i].total)));

pArray.push({

value: parseFloat((res[i].total)),

name: res[i].nickname

})

}

配置图表选项：设置图表的标题、图例、坐标轴、系列等选项。

javascript

var option = {};

let titleObj = this.funnel.title;

titleObj.text = '作者直播时长';

let legendObj = {

data: xAxis,

};

legendObj = Object.assign(legendObj , this.funnel.legend);

let tooltipObj = {trigger: 'item',formatter: '{b} : {c}'};

tooltipObj = Object.assign(tooltipObj , this.funnel.tooltip?this.funnel.tooltip:{});

let seriesObj = {

name: '作者直播时长',

data: pArray,

type: 'funnel',

left: '10%',

top: 60,

bottom: 60,

width: '80%',

minSize: '0%',

maxSize: '100%',

};

option = {

backgroundColor: this.funnel.backgroundColor,

color: this.funnel.color,

title: titleObj,

legend: legendObj,

tooltip: tooltipObj,

series: [seriesObj]

};

显示图表：使用setOption方法将配置项应用到图表上，并监听窗口大小变化，动态调整图表大小。

javascript

durationChart6.setOption(option);

window.onresize = function() {

durationChart6.resize();

};

结果分析说明：通过作者直播时长漏斗图，可以直观地看出不同作者的直播时长占比情况。漏斗图顶部较宽的部分表示直播时长较长的作者，占比较大；底部较窄的部分表示直播时长较短的作者，占比较小。可以根据这些数据，对直播时长较长的作者进行分析，了解他们的直播策略和优势，为其他作者提供参考；对于直播时长较短的作者，可以提供培训和支持，帮助他们提高直播时长和质量。

### 5.2.3数据爬取流程

启动爬虫程序，从各大电商平台、直播平台爬取直播带货相关数据，对爬取的数据进行清洗和预处理，去除重复数据、无效数据等，将处理后的数据存储到MySQL数据库中。

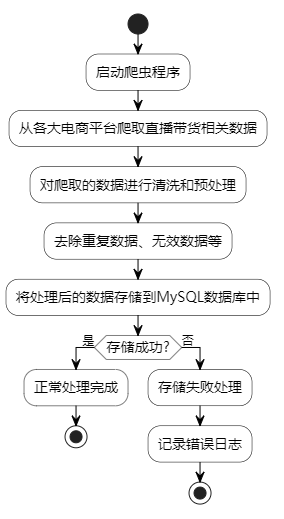


图5.5 数据爬取流程

使用Python的Scrapy框架编写爬虫程序，配置好爬取的网站地址和规则，使用BeautifulSoup或lxml库对爬取的数据进行解析和清洗。使用pymysql库将处理后的数据存储到MySQL数据库中。



图5.6 爬取实现

### 5.2.4通知公告发布流程

管理员在界面上输入公告标题、内容等信息，点击发布按钮，系统将公告信息存储到数据库中，并在小程序首页展示。

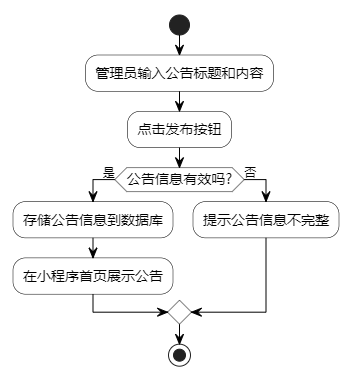


图5.7 通知公告发布流程

在后端服务器中创建通知公告发布接口，接收前端传递的公告信息并存储到数据库中。前端页面提供公告发布表单，用户输入信息后通过AJAX请求调用接口进行发布。

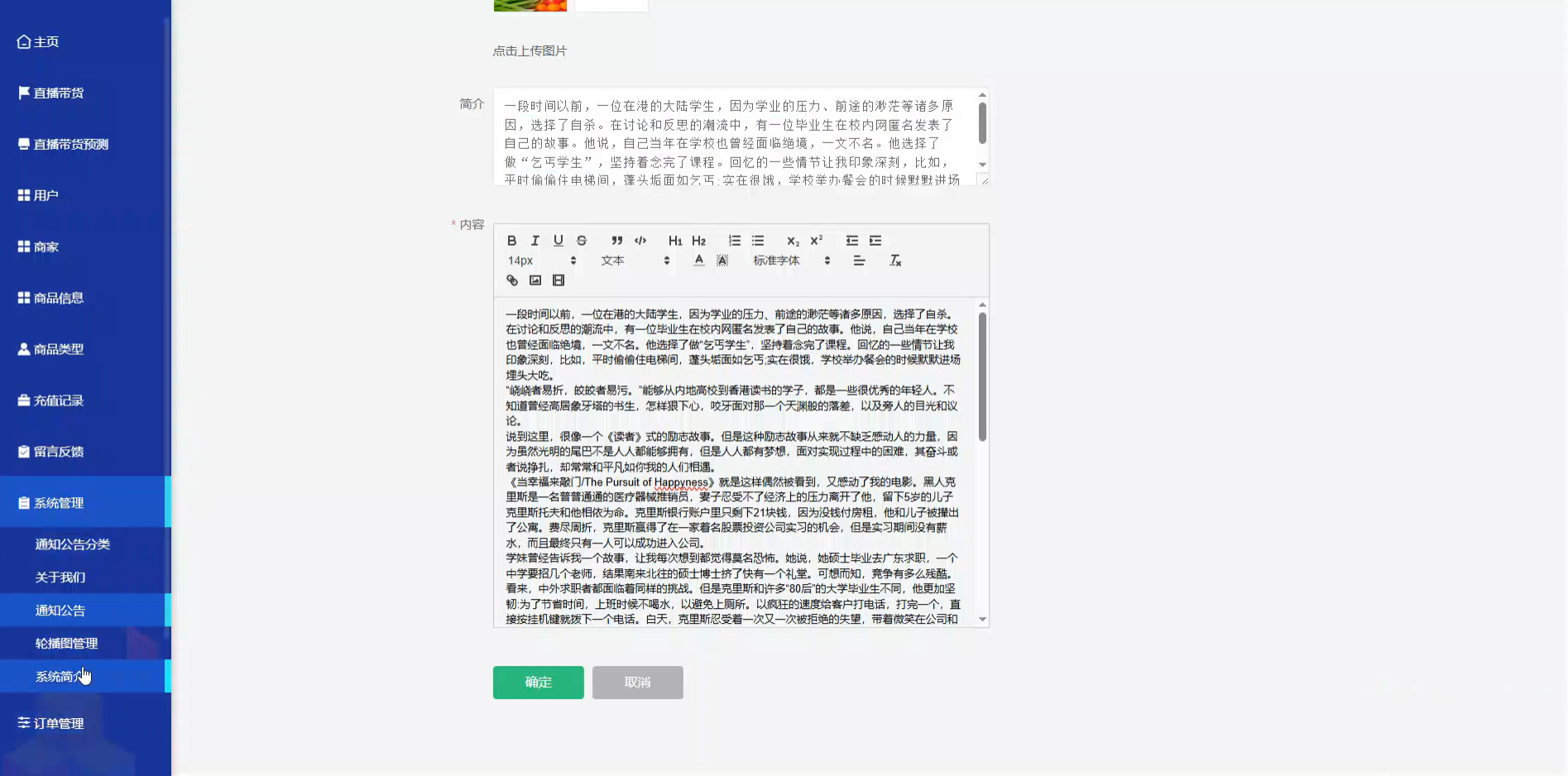


图5.8 通知公告发布实现

## 5.3微信小程序用户端功能实现

### 5.3.1注册与登录流程

用户打开微信小程序，点击注册按钮，输入手机号码、验证码、密码等信息进行注册。注册成功后，用户可以使用手机号码和密码登录小程序。

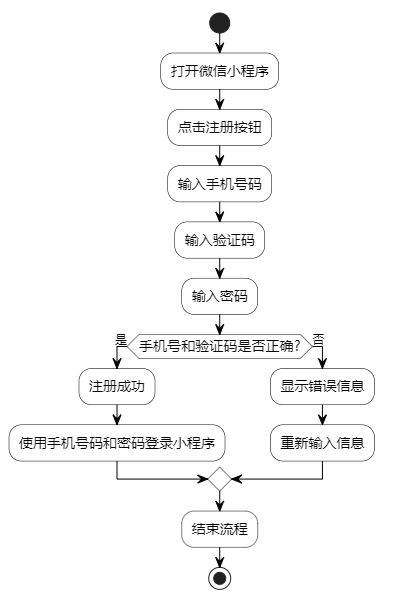


图5.9 注册与登录流程

使用微信小程序的API获取用户手机号码和验证码，通过AJAX请求将注册信息发送到后端服务器进行注册。登录时，用户输入手机号码和密码，后端服务器验证信息的有效性。

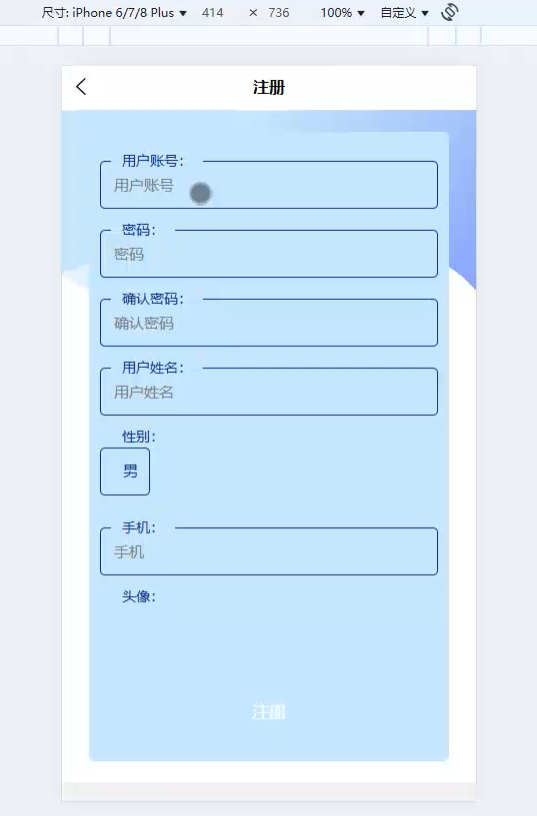


图5.10 注册与登录实现

### 5.3.2商品浏览流程

用户登录后进入小程序首页，浏览公告通知和商品信息。可以通过分类筛选、搜索等方式查找感兴趣的商品，点击商品进入商品详情页查看商品详细信息。

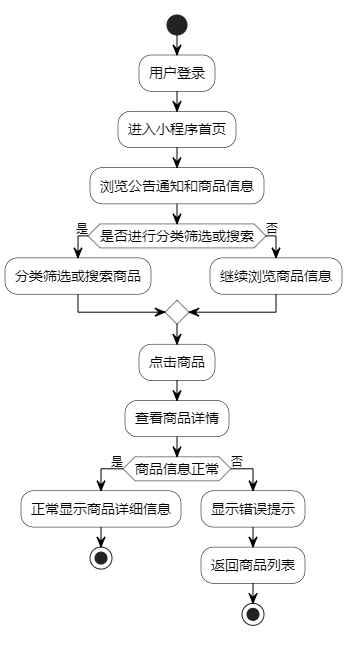


图5.11 商品浏览流程

用户在商品详情页点击加入购物车按钮，商品信息添加到购物车中。用户可以在购物车中修改商品数量、删除商品，点击结算按钮生成订单。

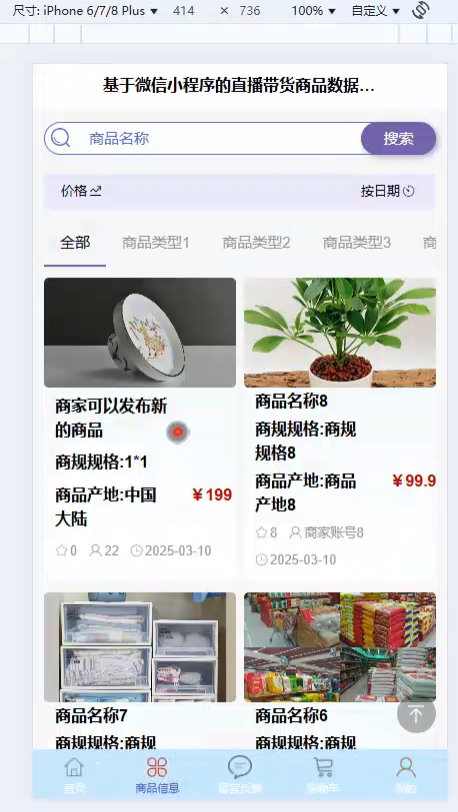


图5.12 商品浏览实现

### 5.3.3留言反馈流程

用户在商品详情页或专门的留言反馈页面输入留言内容，点击提交按钮，系统将留言信息存储到数据库中。

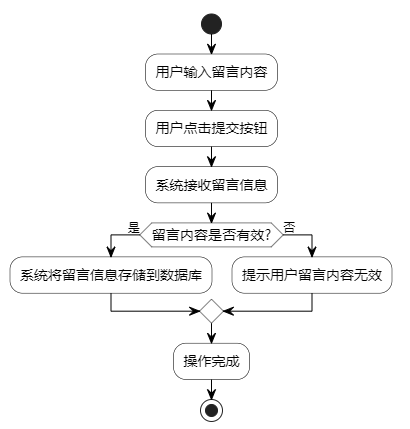


图5.13 留言反馈流程

前端页面提供留言输入框和提交按钮，用户输入留言内容后通过AJAX请求将留言信息发送到后端服务器进行存储。



图5.14 留言反馈实现

## 5.4商家后台功能实现

### 5.4.1商品管理流程

商家在商品管理页面可以添加、编辑、删除商品信息，上传商品图片，设置商品价格、库存等信息。添加商品时，系统将商品信息存储到数据库中；编辑商品时，系统更新数据库中的商品信息；删除商品时，系统从数据库中删除商品信息。

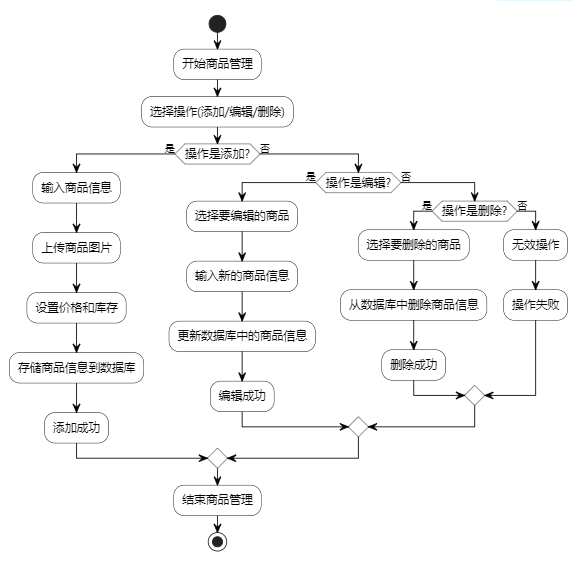


图5.15 商品管理流程

前端页面使用HTML、CSS和JavaScript实现商品管理页面，提供商品添加、编辑、删除的表单。后端服务器提供商品管理接口，前端通过AJAX请求调用这些接口对商品信息进行操作。

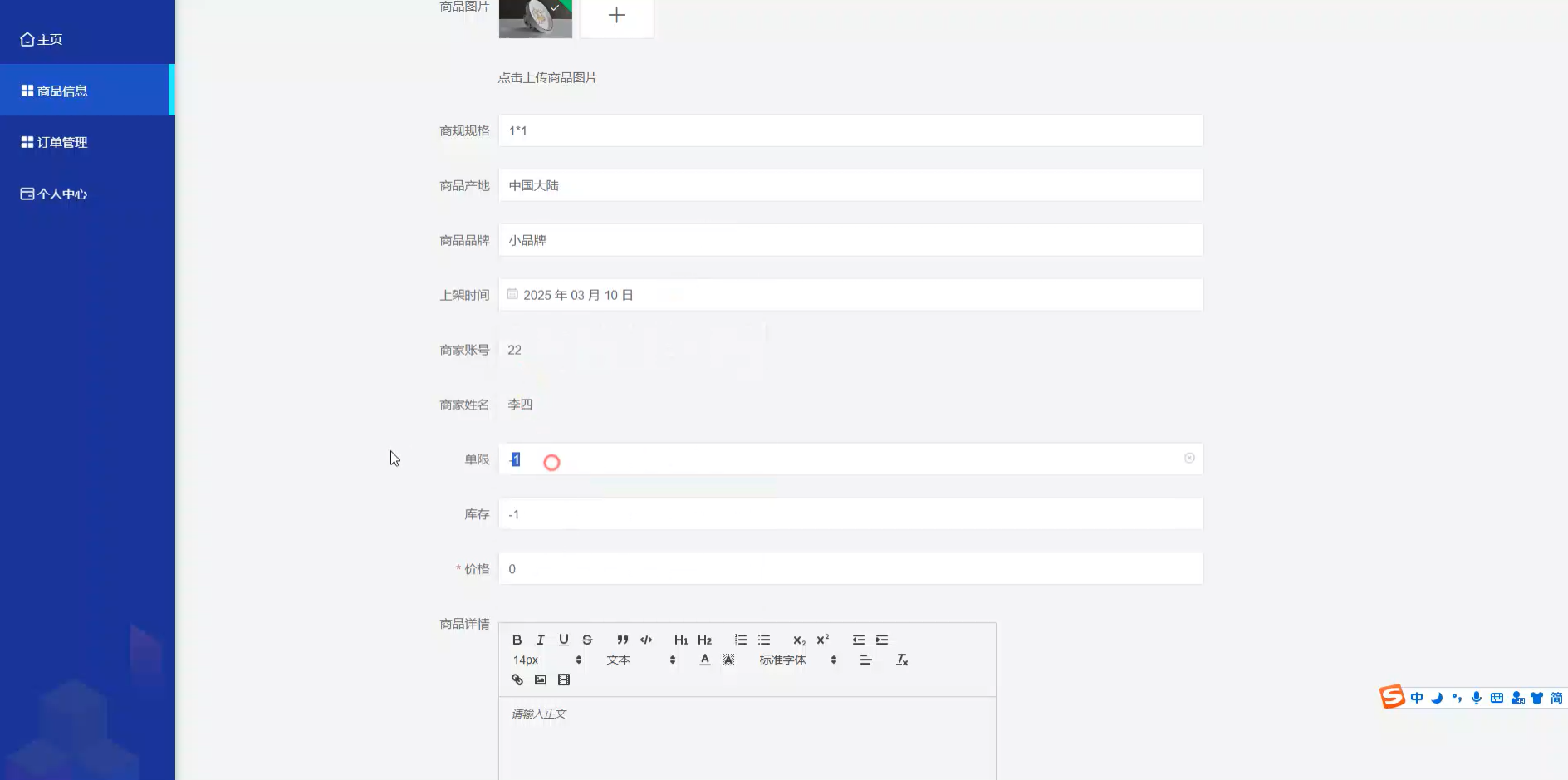


图5.16 商品管理实现

### 5.4.2订单管理流程

商家在订单管理页面查看订单列表，包括订单编号、订单状态、商品名称、订单金额等信息。可以对订单进行发货、退款、售后等操作，操作完成后系统更新数据库中的订单状态。

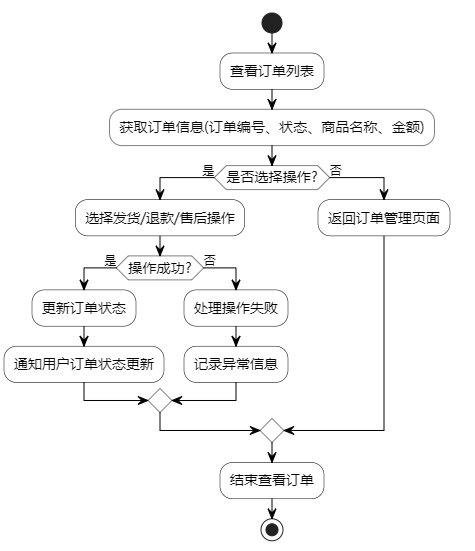


图5.17 订单管理流程

前端页面使用HTML、CSS和JavaScript实现订单管理页面，展示订单列表和订单详情。后端服务器提供订单管理接口，前端通过AJAX请求调用这些接口对订单进行操作。

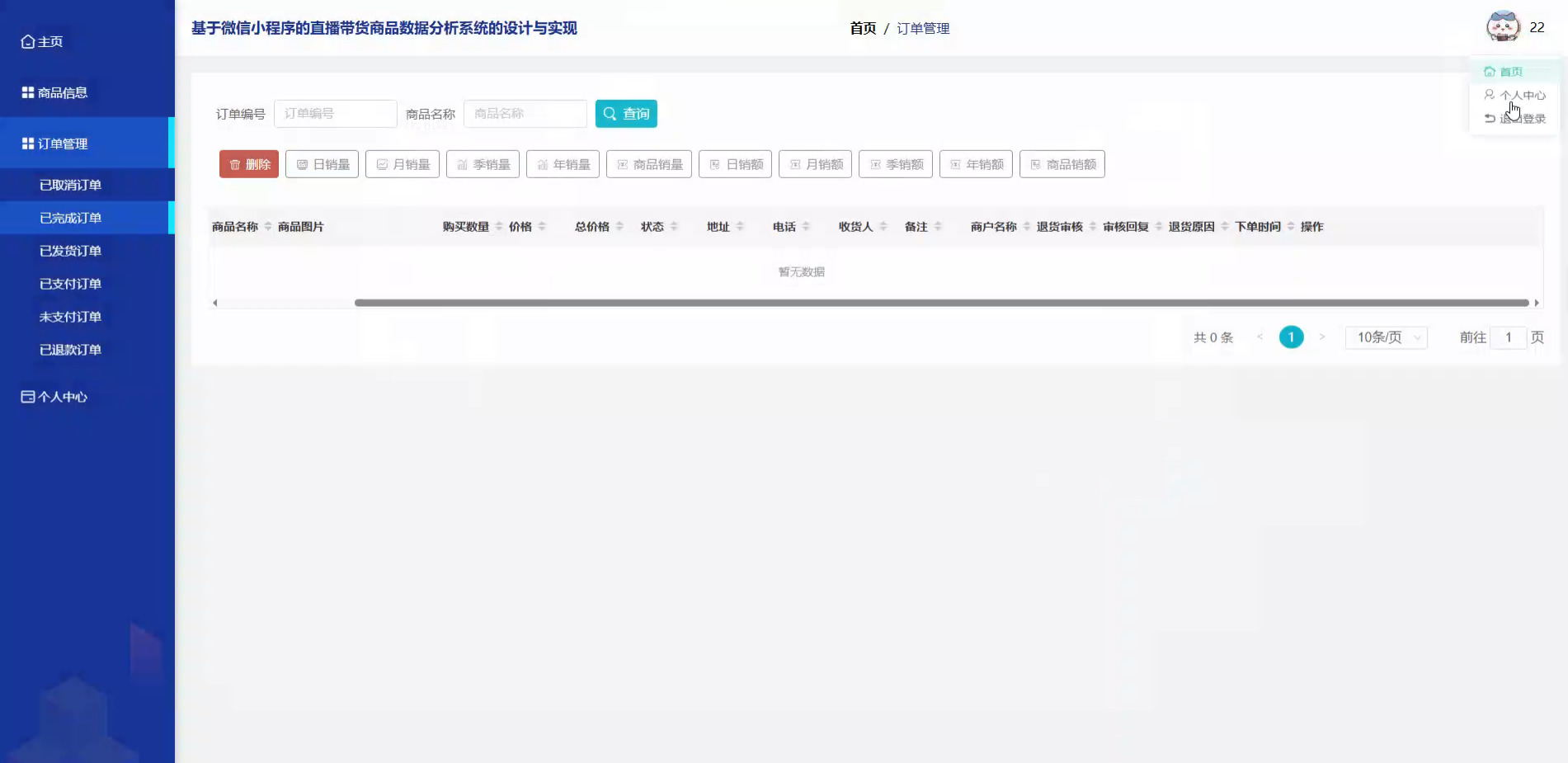


图5.18 订单管理实现

## 5.5本章小结

系统开发选用PyCharm工具，在管理员后台实现登录验证、数据分析可视化、数据爬取和公告发布功能；微信小程序用户端完成注册登录、商品浏览、留言反馈等功能；商家后台实现商品管理和订单管理功能。各功能通过相应技术和流程实现，前后端协作确保系统稳定运行，满足用户和商家的多样化需求。

# 6 功能测试

## 6.1测试用例

表6.1 管理员后台功能测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试模块 | 测试用例描述 | 预期结果 | 测试数据 |
| 管理员后台-登录 | 输入正确的管理员账号和密码点击登录 | 成功进入管理员后台主界面 | 管理员账号：admin，密码：123456 |
| 管理员后台-数据分析与可视化 | 点击数据分析按钮，查看直播带货相关数据的可视化图表 | 图表正确展示直播带货的销量、销售额等数据趋势 | 无（系统自动从数据库取数） |
| 管理员后台-数据爬取 | 点击数据爬取按钮，启动爬虫程序 | 爬虫程序正常运行，数据库中新增爬取的数据 | 无（爬虫按预设规则爬取电商、直播平台数据） |
| 管理员后台-用户管理 | 在用户列表中选择一个用户，点击封禁按钮 | 该用户在小程序端无法登录，系统提示账号已封禁 | 随机选择一个已注册用户 |
| 管理员后台-通知公告发布 | 输入公告标题“限时折扣活动”，内容“全场商品八折”，点击发布 | 小程序首页成功显示该公告 | 标题：限时折扣活动，内容：全场商品八折 |

表6.2 微信小程序用户端功能测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试模块 | 测试用例描述 | 预期结果 | 测试数据 |
| 微信小程序用户端-注册 | 输入手机号码13800138000，验证码1234，密码abcdef，确认密码abcdef，点击注册 | 注册成功，提示注册成功信息，可使用该账号登录 | 手机号码：13800138000，验证码：1234，密码：abcdef |
| 微信小程序用户端-商品浏览 | 在小程序首页点击商品分类为“电子产品”，浏览商品列表 | 显示所有分类为电子产品的商品信息 | 无（系统按分类从数据库取商品数据） |
| 微信小程序用户端-留言反馈 | 在商品详情页点击留言反馈，输入留言“商品质量不错”，点击提交 | 留言成功，数据库中新增该条留言记录 | 留言：商品质量不错 |
| 微信小程序用户端-购物车 | 在商品详情页点击加入购物车，进入购物车页面，修改商品数量为2，点击结算 | 购物车商品数量更新为2，跳转到订单结算页面 | 商品ID：1001（假设） |
| 微信小程序用户端-个人中心 | 点击个人中心，查看充值记录 | 显示该用户的充值历史记录 | 无（系统从数据库取该用户充值记录） |

6.3 商家后台功能测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试模块 | 测试用例描述 | 预期结果 | 测试数据 |
| 商家后台-登录 | 输入正确的商家账号和密码点击登录 | 成功进入商家后台主界面 | 商家账号：seller1，密码：sellerpass |
| 商家后台-商品管理 | 点击商品添加，输入商品名称“新款手机”，价格3999，库存100，上传商品图片，点击保存 | 商品成功添加到数据库，商品管理列表中显示该商品 | 商品名称：新款手机，价格：3999，库存：100 |
| 商家后台-订单管理 | 在订单列表中选择一个订单，点击发货按钮 | 订单状态更新为已发货，数据库中订单状态字段变更 | 随机选择一个未发货订单 |
| 商家后台-个人中心 | 点击个人中心，修改商家联系方式为13900139000，点击保存 | 商家联系方式修改成功，数据库中商家信息更新 | 新联系方式：13900139000 |

表6.4 数据爬取专项测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试模块 | 测试用例描述 | 预期结果 | 测试数据 |
| 数据爬取模块 | 启动数据爬取模块，设置爬取电商平台A的直播带货商品数据 | 成功从电商平台A爬取商品数据，并存储到数据库中 | 电商平台A网址及相关爬取规则 |

## 6.2测试结果

经过全面测试，各模块表现良好。管理员后台登录顺畅，数据分析与可视化图表精准呈现数据，数据爬取成功填充数据库，用户管理封禁操作有效，通知公告发布及时展示于小程序首页。微信小程序用户端注册登录便捷，商品浏览分类清晰，留言反馈记录完整，购物车数量修改及结算正常，个人中心充值记录准确呈现。商家后台登录无阻碍，商品管理添加商品无误，订单管理发货操作顺利更新状态，个人中心联系方式修改成功。数据爬取模块从指定电商平台成功获取数据并正确存储，整体功能测试均顺利通过，系统功能运行稳定。

## 6.3本章小结

针对系统各模块制定测试用例，涵盖管理员、用户端、商家端及数据爬取功能。测试结果表明，各模块运行稳定。管理员后台登录、数据处理和管理功能正常；用户端注册登录、购物等操作顺畅；商家端登录、商品与订单管理有效；数据爬取成功获取并存储数据，系统整体满足设计要求，可投入使用。

# 参考文献

[1]董彦喆.直播电商现状、问题与优化路径的综述[J].河北企业,2024,(12):19-23.

[2]魏玉平,李嘉钰,李满慧等.直播带货营销手段和消费者购买意愿——感知有用性的中介作用[J].时代经贸,2024,21(11):59-63.

[3]王靓靓,罗雅云.基于大数据的直播带货电商模式分析——以美妆直播为例[J].现代营销(学苑版),2021,(05):116-118.

[4]陈宏君,刘永超,张明初等.大数据背景下湖南农产品直播带货经济效益提升路径研究[J].南方农机,2024,55(24):88-91.

[5]张晓燕.微信“小程序”开发的系统实现及前景[J].电子技术与软件工程,2018,(12):49-50.

[6]刘玉佳.微信“小程序”开发的系统实现及前景分析[J].信息通信,2017,(01):260-261.

[7]南航业界首创微信小程序直播带货[J].空运商务,2020,(07):7.

[8]闫烁.基于微信小程序的商品展示系统的设计与实现[D].大连海事大学,2019.

[9]晁琼,焦博妮,齐琳等.基于微信小程序的“指上农庄”农产品交易平台设计与实现[J].电脑知识与技术,2023,19(29):40-42.

[10]孙隽韬,陈炳丰,黄爱华.一种基于Python的商品评论数据智能获取与分析技术[J].现代计算机(专业版),2018,(14):16-19+23.

[11]胡晓青,曹宇.基于爬虫技术的网购信息获取与营销策略分析[J].中国储运,2018,(09):122-125.

[12]王坦,周祺胤,毛莎.基于机器学习算法的直播电商用户购物行为预测模型的构建与应用[J].湖北大学学报(自然科学版),2023,45(06):872-878.

[13]李翠萍.基于机器学习的直播电商客户流失风险预测[J].互联网周刊,2023,(10):40-42.

[14]Jahan I ,Sanam F T .A comprehensive framework for customer retention in E-commerce using machine learning based on churn prediction, customer segmentation, and recommendation[J].Electronic Commerce Research,2024,(prepublish):1-44.

[15]Vijayaragavan P ,Suresh C ,Maheshwari A , et al.Sustainable sentiment analysis on E-commerce platforms using a weighted parallel hybrid deep learning approach for smart cities applications[J].Scientific Reports,2024,14(1):26508-26508.

[16]赵娟禾.基于多层混合深度神经网络的电商商品短期需求量预测方法研究[D].长安大学,2019.

[17]江涛.基于MLP神经网络的商品评论情感分析[J].电子制作,2021,(12):36-37+78.

[18]Livingston J S ,Selvi T S B ,Thabeetha M , etal.A Neural Network Based Approach for Sentimental Analysis on Amazon Product Reviews[J].International Journal of Innovative Technology and Exploring Engineering (IJITEE),2019,8(6s):469-473.

[19]贾靖仪. 大数据技术与MySQL数据库的应用[J]. 集成电路应用, 2024, 41 (08): 78-79.

[20]周洪亮. Python在实验数据处理中的应用[J]. 信息与电脑, 2024, 36 (23): 209-211.

[21]韩家超. 基于Python的一站式自动化测试平台设计与实现[J]. 大数据时代, 2024, (11): 43-47.

[22]杨媛. 基于Django技术和MVS网络的无人机遥感数据三维重建可视化平台设计[J]. 计算机测量与控制, 2025, 33 (02): 221-228.

[23]王相喜, 周晓波, 倪甜弟. 基于微信小程序的预约挂号系统的设计与实现[J]. 现代计算机, 2024, 30 (08): 116-120.

[24]邓选发. 微信小程序仿冒发现及安全检测技术研究[D]. 北京邮电大学, 2024.

[25]陈欣苹, 赵子潇, 程文莉. 基于大数据的题库微信小程序的设计与开发[J]. 电脑知识与技术, 2024, 20 (03): 65-68.

[26]叶文婵. 基于OMO教学模式的微信小程序开发与教学实践研究[D]. 深圳大学, 2023.

[27]陈怡婧, 郑晓溪, 李芳. 基于微信云开发的校园二手交易平台小程序的设计与实现[J]. 电脑知识与技术, 2022, 18 (32): 51-54.

# 致 谢

在完成这篇论文的过程中，我得到了许多人的帮助，心中满是感激。

我要深深感谢我的父母亲人，他们是我成长道路上最坚实的后盾。生活里，他们无微不至的关怀让我毫无后顾之忧；学习中，无论遇到什么困难，他们都给予我鼓励和支持，让我有勇气不断前行。是他们的爱与付出，成就了现在的我。

感谢我的学校，为我提供了良好的学习环境和丰富的学习资源。宽敞明亮的教室、藏书丰富的图书馆、先进的实验设备，都为我的学习和研究创造了有利条件。在这里，我不仅获取了专业知识，还参与了各种丰富多彩的活动，收获了成长与进步。

学院的培育之恩，我铭记于心。学院精心安排的课程体系，让我逐步构建起扎实的专业知识框架。专业的教学团队和浓厚的学术氛围，不断激发着我的学习热情和创新思维，让我在专业领域不断探索。

特别要感谢我的指导老师，在论文的选题、研究和撰写过程中，他给予了我悉心的指导。每一次的交流，他都耐心解答我的疑问，提出宝贵的建议，帮助我开拓思路、完善论文。他严谨的治学态度和渊博的学识，让我受益匪浅，激励着我不断追求卓越。同时，也要感谢所有授课老师，是他们用精彩的课程，为我打开了知识的大门，传授给我专业知识和技能，为我的学术道路奠定了坚实基础。

还要感谢我的同学们，与他们一起度过的大学时光充满了欢乐和回忆。课堂上，我们共同探讨问题、相互启发；生活中，我们彼此陪伴、分享喜怒哀乐。是他们让我的大学生活变得丰富多彩，这份情谊我会永远珍惜。

感恩所有的遇见与帮助，我将带着这份温暖与力量，继续勇敢前行。

# 附件一：核心代码

#coding:utf-8  
import base64, copy, logging, os, sys, time, xlrd, json, datetime, configparser  
from django.http import JsonResponse  
from django.apps import apps  
import numbers  
from django.db.models.aggregates import Count,Sum  
from django.db.models import Case, When, IntegerField, F  
from django.forms import model\_to\_dict  
import requests  
from util.CustomJSONEncoder import CustomJsonEncoder  
from .models import news  
from util.codes import \*  
from util.auth import Auth  
from util.common import Common  
import util.message as mes  
from django.db import connection  
import random  
from django.core.mail import send\_mail  
from django.conf import settings  
from django.shortcuts import redirect  
from django.db.models import Q  
from util.baidubce\_api import BaiDuBce  
from .config\_model import config  
  
  
def news\_register(request):  
 if request.method in ["POST", "GET"]:  
 msg = {'code': normal\_code, "msg": mes.normal\_code}  
 req\_dict = request.session.get("req\_dict")  
  
  
 error = news.createbyreq(news, news, req\_dict)  
 if error is Exception or (type(error) is str and "Exception" in error):  
 msg['code'] = crud\_error\_code  
 msg['msg'] = "用户已存在,请勿重复注册!"  
 else:  
 msg['data'] = error  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
def news\_login(request):  
 if request.method in ["POST", "GET"]:  
 msg = {'code': normal\_code, "msg": mes.normal\_code}  
 req\_dict = request.session.get("req\_dict")  
 datas = news.getbyparams(news, news, req\_dict)  
 if not datas:  
 msg['code'] = password\_error\_code  
 msg['msg'] = mes.password\_error\_code  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
 try:  
 \_\_sfsh\_\_= news.\_\_sfsh\_\_  
 except:  
 \_\_sfsh\_\_=None  
  
 if \_\_sfsh\_\_=='是':  
 if datas[0].get('sfsh')!='是':  
 msg['code']=other\_code  
 msg['msg'] = "账号已锁定，请联系管理员审核!"  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
   
 req\_dict['id'] = datas[0].get('id')  
  
  
 return Auth.authenticate(Auth, news, req\_dict)  
  
  
def news\_logout(request):  
 if request.method in ["POST", "GET"]:  
 msg = {  
 "msg": "登出成功",  
 "code": 0  
 }  
  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
  
def news\_resetPass(request):  
 *'''  
 '''* if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": mes.normal\_code}  
  
 req\_dict = request.session.get("req\_dict")  
  
 columns= news.getallcolumn( news, news)  
  
 try:  
 \_\_loginUserColumn\_\_= news.\_\_loginUserColumn\_\_  
 except:  
 \_\_loginUserColumn\_\_=None  
 username=req\_dict.get(list(req\_dict.keys())[0])  
 if \_\_loginUserColumn\_\_:  
 username\_str=\_\_loginUserColumn\_\_  
 else:  
 username\_str=username  
 if 'mima' in columns:  
 password\_str='mima'  
 else:  
 password\_str='password'  
  
 init\_pwd = '123456'  
 recordsParam = {}  
 recordsParam[username\_str] = req\_dict.get("username")  
 records=news.getbyparams(news, news, recordsParam)  
 if len(records)<1:  
 msg['code'] = 400  
 msg['msg'] = '用户不存在'  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
 eval('''news.objects.filter({}='{}').update({}='{}')'''.format(username\_str,username,password\_str,init\_pwd))  
   
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
  
  
def news\_session(request):  
 *'''  
 '''* if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code,"msg": mes.normal\_code, "data": {}}  
  
 req\_dict={"id":request.session.get('params').get("id")}  
 msg['data'] = news.getbyparams(news, news, req\_dict)[0]  
  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
  
def news\_default(request):  
  
 if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code,"msg": mes.normal\_code, "data": {}}  
 req\_dict = request.session.get("req\_dict")  
 req\_dict.update({"isdefault":"是"})  
 data=news.getbyparams(news, news, req\_dict)  
 if len(data)>0:  
 msg['data'] = data[0]  
 else:  
 msg['data'] = {}  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
def news\_page(request):  
 *'''  
 '''* if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": mes.normal\_code, "data":{"currPage":1,"totalPage":1,"total":1,"pageSize":10,"list":[]}}  
 req\_dict = request.session.get("req\_dict")  
  
 global news  
  
 #获取全部列名  
 columns= news.getallcolumn( news, news)  
  
 if "vipread" in req\_dict and "vipread" not in columns:  
 del req\_dict["vipread"]  
  
 #当前登录用户所在表  
 tablename = request.session.get("tablename")  
  
 '''\_\_authSeparate\_\_此属性为真，params添加userid，后台只查询个人数据'''  
 try:  
 \_\_authSeparate\_\_=news.\_\_authSeparate\_\_  
 except:  
 \_\_authSeparate\_\_=None  
  
 if \_\_authSeparate\_\_=="是":  
 tablename=request.session.get("tablename")  
 if tablename!="users" and 'userid' in columns and 'userid' not in req\_dict:  
 try:  
 req\_dict['userid']=request.session.get("params").get("id")  
 except:  
 pass  
  
 #当项目属性hasMessage为”是”，生成系统自动生成留言板的表messages，同时该表的表属性hasMessage也被设置为”是”,字段包括userid（用户id），username(用户名)，content（留言内容），reply（回复）  
 #接口page需要区分权限，普通用户查看自己的留言和回复记录，管理员查看所有的留言和回复记录  
 try:  
 \_\_hasMessage\_\_=news.\_\_hasMessage\_\_  
 except:  
 \_\_hasMessage\_\_=None  
 if \_\_hasMessage\_\_=="是":  
 tablename=request.session.get("tablename")  
 if tablename!="users":  
 req\_dict["userid"]=request.session.get("params").get("id")  
  
 # 判断当前表的表属性isAdmin,为真则是管理员表  
 # 当表属性isAdmin=”是”,刷出来的用户表也是管理员，即page和list可以查看所有人的考试记录(同时应用于其他表)  
 \_\_isAdmin\_\_ = None  
  
 allModels = apps.get\_app\_config('main').get\_models()  
 for m in allModels:  
 if m.\_\_tablename\_\_==tablename:  
  
 try:  
 \_\_isAdmin\_\_ = m.\_\_isAdmin\_\_  
 except:  
 \_\_isAdmin\_\_ = None  
 break  
  
 # 当前表也是有管理员权限的表  
 if \_\_isAdmin\_\_ == "是" and 'news' != 'forum' :  
 if req\_dict.get("userid") and 'news' != 'chat' and 'news' != 'examrecord':  
 del req\_dict["userid"]  
 else:  
 if tablename!="users" and tablename!="jdfnl" and 'news'[:7]!='discuss' and "userid" in news.getallcolumn(news,news):  
 req\_dict["userid"] = request.session.get("params").get("id")  
  
 #当列属性authTable有值(某个用户表)[该列的列名必须和该用户表的登陆字段名一致]，则对应的表有个隐藏属性authTable为”是”，那么该用户查看该表信息时，只能查看自己的  
 try:  
 \_\_authTables\_\_=news.\_\_authTables\_\_  
 except:  
 \_\_authTables\_\_=None  
  
 if \_\_authTables\_\_!=None and \_\_authTables\_\_!={} and \_\_isAdmin\_\_ == "是":  
 for authColumn,authTable in \_\_authTables\_\_.items():  
 if authTable==tablename:  
 params = request.session.get("params")  
 req\_dict[authColumn]=params.get(authColumn)  
 username=params.get(authColumn)  
 break  
 q = Q()  
  
 msg['data']['list'], msg['data']['currPage'], msg['data']['totalPage'], msg['data']['total'], \  
 msg['data']['pageSize'] =news.page(news, news, req\_dict, request, q)  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
def news\_autoSort(request):  
 *'''  
 ．智能推荐功能(表属性：[intelRecom（是/否）],新增clicktime[前端不显示该字段]字段（调用info/detail接口的时候更新），按clicktime排序查询)  
主要信息列表（如商品列表，新闻列表）中使用，显示最近点击的或最新添加的5条记录就行  
 '''* if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": mes.normal\_code, "data":{"currPage":1,"totalPage":1,"total":1,"pageSize":10,"list":[]}}  
 req\_dict = request.session.get("req\_dict")  
 if "clicknum" in news.getallcolumn(news,news):  
 req\_dict['sort']='clicknum'  
 elif "browseduration" in news.getallcolumn(news,news):  
 req\_dict['sort']='browseduration'  
 else:  
 req\_dict['sort']='clicktime'  
 req\_dict['order']='desc'  
 msg['data']['list'], msg['data']['currPage'], msg['data']['totalPage'], msg['data']['total'], \  
 msg['data']['pageSize'] = news.page(news,news, req\_dict)  
  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
#分类列表  
def news\_lists(request):  
 if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": mes.normal\_code, "data":[]}  
 msg['data'],\_,\_,\_,\_ = news.page(news, news, {})  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
def news\_query(request):  
 *'''  
 '''* if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": mes.normal\_code, "data": {}}  
 try:  
 query\_result = news.objects.filter(\*\*request.session.get("req\_dict")).values()  
 msg['data'] = query\_result[0]  
 except Exception as e:  
  
 msg['code'] = crud\_error\_code  
 msg['msg'] = f"发生错误：{e}"  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
def news\_list(request):  
 *'''  
 前台分页  
 '''* if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": mes.normal\_code, "data":{"currPage":1,"totalPage":1,"total":1,"pageSize":10,"list":[]}}  
 req\_dict = request.session.get("req\_dict")  
 #获取全部列名  
 columns= news.getallcolumn( news, news)  
 if "vipread" in req\_dict and "vipread" not in columns:  
 del req\_dict["vipread"]  
 #表属性[foreEndList]前台list:和后台默认的list列表页相似,只是摆在前台,否:指没有此页,是:表示有此页(不需要登陆即可查看),前要登:表示有此页且需要登陆后才能查看  
 try:  
 \_\_foreEndList\_\_=news.\_\_foreEndList\_\_  
 except:  
 \_\_foreEndList\_\_=None  
 try:  
 \_\_foreEndListAuth\_\_=news.\_\_foreEndListAuth\_\_  
 except:  
 \_\_foreEndListAuth\_\_=None  
  
 #authSeparate  
 try:  
 \_\_authSeparate\_\_=news.\_\_authSeparate\_\_  
 except:  
 \_\_authSeparate\_\_=None  
  
 if \_\_foreEndListAuth\_\_ =="是" and \_\_authSeparate\_\_=="是":  
 tablename=request.session.get("tablename")  
 if tablename!="users" and request.session.get("params") is not None:  
 req\_dict['userid']=request.session.get("params").get("id")  
  
 tablename = request.session.get("tablename")  
 if tablename == "users" and req\_dict.get("userid") != None:#判断是否存在userid列名  
 del req\_dict["userid"]  
 else:  
 \_\_isAdmin\_\_ = None  
  
 allModels = apps.get\_app\_config('main').get\_models()  
 for m in allModels:  
 if m.\_\_tablename\_\_==tablename:  
  
 try:  
 \_\_isAdmin\_\_ = m.\_\_isAdmin\_\_  
 except:  
 \_\_isAdmin\_\_ = None  
 break  
  
 if \_\_isAdmin\_\_ == "是":  
 if req\_dict.get("userid"):  
 # del req\_dict["userid"]  
 pass  
 else:  
 #非管理员权限的表,判断当前表字段名是否有userid  
 if "userid" in columns:  
 try:  
 pass  
 except:  
 pass  
 #当列属性authTable有值(某个用户表)[该列的列名必须和该用户表的登陆字段名一致]，则对应的表有个隐藏属性authTable为”是”，那么该用户查看该表信息时，只能查看自己的  
 try:  
 \_\_authTables\_\_=news.\_\_authTables\_\_  
 except:  
 \_\_authTables\_\_=None  
  
 if \_\_authTables\_\_!=None and \_\_authTables\_\_!={} and \_\_foreEndListAuth\_\_=="是":  
 for authColumn,authTable in \_\_authTables\_\_.items():  
 if authTable==tablename:  
 try:  
 del req\_dict['userid']  
 except:  
 pass  
 params = request.session.get("params")  
 req\_dict[authColumn]=params.get(authColumn)  
 username=params.get(authColumn)  
 break  
   
 if news.\_\_tablename\_\_[:7]=="discuss":  
 try:  
 del req\_dict['userid']  
 except:  
 pass  
  
  
 q = Q()  
 msg['data']['list'], msg['data']['currPage'], msg['data']['totalPage'], msg['data']['total'], \  
 msg['data']['pageSize'] = news.page(news, news, req\_dict, request, q)  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
def news\_save(request):  
 *'''  
 后台新增  
 '''* if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": mes.normal\_code, "data": {}}  
 req\_dict = request.session.get("req\_dict")  
 if 'clicktime' in req\_dict.keys():  
 del req\_dict['clicktime']  
 tablename=request.session.get("tablename")  
 \_\_isAdmin\_\_ = None  
 allModels = apps.get\_app\_config('main').get\_models()  
 for m in allModels:  
 if m.\_\_tablename\_\_==tablename:  
  
 try:  
 \_\_isAdmin\_\_ = m.\_\_isAdmin\_\_  
 except:  
 \_\_isAdmin\_\_ = None  
 break  
  
 #获取全部列名  
 columns= news.getallcolumn( news, news)  
 if tablename!='users' and req\_dict.get("userid")==None and 'userid' in columns and \_\_isAdmin\_\_!='是':  
 params=request.session.get("params")  
 req\_dict['userid']=params.get('id')  
  
  
 if 'addtime' in req\_dict.keys():  
 del req\_dict['addtime']  
  
 idOrErr= news.createbyreq(news,news, req\_dict)  
 if idOrErr is Exception:  
 msg['code'] = crud\_error\_code  
 msg['msg'] = idOrErr  
 else:  
 msg['data'] = idOrErr  
  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
def news\_add(request):  
 *'''  
 前台新增  
 '''* if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": mes.normal\_code, "data": {}}  
 req\_dict = request.session.get("req\_dict")  
 tablename=request.session.get("tablename")  
  
 #获取全部列名  
 columns= news.getallcolumn( news, news)  
 try:  
 \_\_authSeparate\_\_=news.\_\_authSeparate\_\_  
 except:  
 \_\_authSeparate\_\_=None  
  
 if \_\_authSeparate\_\_=="是":  
 tablename=request.session.get("tablename")  
 if tablename!="users" and 'userid' in columns:  
 try:  
 req\_dict['userid']=request.session.get("params").get("id")  
 except:  
 pass  
  
 try:  
 \_\_foreEndListAuth\_\_=news.\_\_foreEndListAuth\_\_  
 except:  
 \_\_foreEndListAuth\_\_=None  
  
 if \_\_foreEndListAuth\_\_ and \_\_foreEndListAuth\_\_!="否":  
 tablename=request.session.get("tablename")  
 if tablename!="users":  
 req\_dict['userid']=request.session.get("params").get("id")  
  
  
 if 'addtime' in req\_dict.keys():  
 del req\_dict['addtime']  
 error= news.createbyreq(news,news, req\_dict)  
 if error is Exception:  
 msg['code'] = crud\_error\_code  
 msg['msg'] = error  
 else:  
 msg['data'] = error  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
def news\_thumbsup(request,id\_):  
 *'''  
 点赞：表属性thumbsUp[是/否]，刷表新增thumbsupnum赞和crazilynum踩字段，  
 '''* if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": mes.normal\_code, "data": {}}  
 req\_dict = request.session.get("req\_dict")  
 id\_=int(id\_)  
 type\_=int(req\_dict.get("type",0))  
 rets=news.getbyid(news,news,id\_)  
  
 update\_dict={  
 "id":id\_,  
 }  
 if type\_==1:#赞  
 update\_dict["thumbsupnum"]=int(rets[0].get('thumbsupnum'))+1  
 elif type\_==2:#踩  
 update\_dict["crazilynum"]=int(rets[0].get('crazilynum'))+1  
 error = news.updatebyparams(news,news, update\_dict)  
 if error!=None:  
 msg['code'] = crud\_error\_code  
 msg['msg'] = error  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
  
def news\_info(request,id\_):  
 *'''  
 '''* if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": mes.normal\_code, "data": {}}  
  
 data = news.getbyid(news,news, int(id\_))  
 if len(data)>0:  
 msg['data']=data[0]  
 if msg['data'].\_\_contains\_\_("reversetime"):  
 if isinstance(msg['data']['reversetime'], datetime.datetime):  
 msg['data']['reversetime'] = msg['data']['reversetime'].strftime("%Y-%m-%d %H:%M:%S")  
 else:  
 if msg['data']['reversetime'] != None:  
 reversetime = datetime.datetime.strptime(msg['data']['reversetime'], '%Y-%m-%d %H:%M:%S')  
 msg['data']['reversetime'] = reversetime.strftime("%Y-%m-%d %H:%M:%S")  
  
 #浏览点击次数  
 try:  
 \_\_browseClick\_\_= news.\_\_browseClick\_\_  
 except:  
 \_\_browseClick\_\_=None  
  
 if \_\_browseClick\_\_=="是" and "clicknum" in news.getallcolumn(news,news):  
 try:  
 clicknum=int(data[0].get("clicknum",0))+1  
 except:  
 clicknum=0+1  
 click\_dict={"id":int(id\_),"clicknum":clicknum,"clicktime":datetime.datetime.now()}  
 ret=news.updatebyparams(news,news,click\_dict)  
 if ret!=None:  
 msg['code'] = crud\_error\_code  
 msg['msg'] = ret  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
def news\_detail(request,id\_):  
 *'''  
 '''* if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": mes.normal\_code, "data": {}}  
  
 data =news.getbyid(news,news, int(id\_))  
 if len(data)>0:  
 msg['data']=data[0]  
 if msg['data'].\_\_contains\_\_("reversetime"):  
 if isinstance(msg['data']['reversetime'], datetime.datetime):  
 msg['data']['reversetime'] = msg['data']['reversetime'].strftime("%Y-%m-%d %H:%M:%S")  
 else:  
 if msg['data']['reversetime'] != None:  
 reversetime = datetime.datetime.strptime(msg['data']['reversetime'], '%Y-%m-%d %H:%M:%S')  
 msg['data']['reversetime'] = reversetime.strftime("%Y-%m-%d %H:%M:%S")  
  
 #浏览点击次数  
 try:  
 \_\_browseClick\_\_= news.\_\_browseClick\_\_  
 except:  
 \_\_browseClick\_\_=None  
  
 if \_\_browseClick\_\_=="是" and "clicknum" in news.getallcolumn(news,news):  
 try:  
 clicknum=int(data[0].get("clicknum",0))+1  
 except:  
 clicknum=0+1  
 click\_dict={"id":int(id\_),"clicknum":clicknum,"clicktime":datetime.datetime.now()}  
  
 ret=news.updatebyparams(news,news,click\_dict)  
 if ret!=None:  
 msg['code'] = crud\_error\_code  
 msg['msg'] = ret  
 return JsonResponse(msg, encoder=CustomJsonEncoder)  
  
def news\_update(request):  
 *'''  
 '''* if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": mes.normal\_code, "data": {}}  
 req\_dict = request.session.get("req\_dict")  
 if 'clicktime' in req\_dict.keys() and req\_dict['clicktime']=="None":  
 del req\_dict['clicktime']  
 if req\_dict.get("mima") and "mima" not in news.getallcolumn(news,news) :  
 del req\_dict["mima"]  
 if req\_dict.get("password") and "password" not in news.getallcolumn(news,news) :  
 del req\_dict["password"]  
 try:  
 del req\_dict["clicknum"]  
 except:  
 pass  
  
  
 error = news.updatebyparams(news, news, req\_dict)  
 if error!=None:  
 msg['code'] = crud\_error\_code  
 msg['msg'] = error  
  
 return JsonResponse(msg)  
  
  
def news\_delete(request):  
 *'''  
 批量删除  
 '''* if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": mes.normal\_code, "data": {}}  
 req\_dict = request.session.get("req\_dict")  
  
 error=news.deletes(news,  
 news,  
 req\_dict.get("ids")  
 )  
 if error!=None:  
 msg['code'] = crud\_error\_code  
 msg['msg'] = error  
 return JsonResponse(msg)  
  
  
def news\_vote(request,id\_):  
 *'''  
 浏览点击次数（表属性[browseClick:是/否]，点击字段（clicknum），调用info/detail接口的时候后端自动+1）、投票功能（表属性[vote:是/否]，投票字段（votenum）,调用vote接口后端votenum+1）  
统计商品或新闻的点击次数；提供新闻的投票功能  
 '''* if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": mes.normal\_code}  
  
  
 data= news.getbyid(news, news, int(id\_))  
 for i in data:  
 votenum=i.get('votenum')  
 if votenum!=None:  
 params={"id":int(id\_),"votenum":votenum+1}  
 error=news.updatebyparams(news,news,params)  
 if error!=None:  
 msg['code'] = crud\_error\_code  
 msg['msg'] = error  
 return JsonResponse(msg)  
  
def news\_importExcel(request):  
 if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": "成功", "data": {}}  
  
 excel\_file = request.FILES.get("file", "")  
 if excel\_file.size > 100 \* 1024 \* 1024: # 限制为 100MB  
 msg['code'] = 400  
 msg["msg"] = '文件大小不能超过100MB'  
 return JsonResponse(msg)  
  
 file\_type = excel\_file.name.split('.')[1]  
   
 if file\_type in ['xlsx', 'xls']:  
 data = xlrd.open\_workbook(filename=None, file\_contents=excel\_file.read())  
 table = data.sheets()[0]  
 rows = table.nrows  
   
 try:  
 for row in range(1, rows):  
 row\_values = table.row\_values(row)  
 req\_dict = {}  
 news.createbyreq(news, news, req\_dict)  
   
 except:  
 pass  
   
 else:  
 msg = {  
 "msg": "文件类型错误",  
 "code": 500  
 }  
   
 return JsonResponse(msg)  
  
  
# 收藏推荐算法接口  
def news\_autoSort2(request):  
   
 if request.method in ["POST", "GET"]:  
 req\_dict = request.session.get("req\_dict")  
 cursor = connection.cursor()  
 leixing = set()  
 try:  
 cursor.execute("select inteltype from storeup where userid = %d"%(request.session.get("params").get("id"))+" and tablename = 'news' order by addtime desc")  
 rows = cursor.fetchall()  
 for row in rows:  
 for item in row:  
 if item != None:  
 leixing.add(item)  
 except:  
 leixing = set()  
   
 L = []  
  
 where = " AND ".join([f"{key} = '{value}'" for key, value in req\_dict.items() if key!="page" and key!="limit" and key!="order"and key!="sort"])  
 if where:  
 sql = f'''SELECT \* FROM (SELECT \* FROM news WHERE {where}) AS table1 WHERE typename IN ('{"','".join(leixing)}') union all SELECT \* FROM (SELECT \* FROM news WHERE {where}) AS table1 WHERE typename NOT IN ('{"','".join(leixing)}')'''  
 else:  
 sql = f'''select \* from news where typename in ('{"','".join(leixing)}') union all select \* from news where typename not in('{"','".join(leixing)}')'''  
 cursor.execute(sql)  
 desc = cursor.description  
 data\_dict = [dict(zip([col[0] for col in desc], row)) for row in cursor.fetchall()]   
 for online\_dict in data\_dict:  
 for key in online\_dict:  
 if 'datetime.datetime' in str(type(online\_dict[key])):  
 online\_dict[key] = online\_dict[key].strftime(  
 "%Y-%m-%d %H:%M:%S")  
 else:  
 pass  
 L.append(online\_dict)  
  
  
 return JsonResponse({"code": 0, "msg": '', "data":{"currPage":1,"totalPage":1,"total":1,"pageSize":5,"list": L[0:int(req\_dict["limit"])]}}, encoder=CustomJsonEncoder)  
  
  
  
  
def news\_security(request):  
 *'''  
 获取密保接口  
 '''* if request.method in ["POST", "GET"]:  
 msg = {"code": normal\_code, "msg": "成功", "data": {}}  
 req\_dict = request.session.get("req\_dict")  
   
 sql = "SELECT \* FROM news where ='{0}'".format(req\_dict['username'])  
 record = {}  
 cursor = connection.cursor()  
 cursor.execute(sql)  
 desc = cursor.description  
 data\_dict = [dict(zip([col[0] for col in desc], row)) for row in cursor.fetchall()]   
 for online\_dict in data\_dict:  
 record = online\_dict  
 msg['data'] = record  
  
 return JsonResponse(msg)