

本 科 生 毕 业 设 计（或论文）

论 文 题 目 基于Vue3+NodeJS的苗木交易

平台的设计与实现

作 者 姓 名 李鑫

所学专业名称 软件工程

指 导 教 师 张燕玲

2023年4月3日

学 生： （签字）

学 号：

论文答辩日期：

指 导 教 师： （签字）

目录

[Abstract 3](#_Toc29569)

[1 绪论 4](#_Toc13358)

[1.1 研究背景及意义 4](#_Toc21050)

[1.2 研究目的 4](#_Toc4815)

[1.3 本文主要工作和内容 4](#_Toc4115)

[1.4 论文组织结构 4](#_Toc28082)

[2 相关技术 5](#_Toc23876)

[2.1 NodeJS 5](#_Toc6956)

[2.2 Express 服务端框架 5](#_Toc2508)

[2.3 SocketIO 库 5](#_Toc6744)

[2.4 JOSN Web Token 5](#_Toc4895)

[2.5 Vue3 框架 5](#_Toc10872)

[2.6 MongoDB 数据库 5](#_Toc7314)

[2.7 MVC 5](#_Toc9790)

[2.8 RESTful API 5](#_Toc4093)

[2.9 vscode 开发环境 6](#_Toc2667)

[3 需求分析与设计 6](#_Toc20564)

[3.1 可行性分析 6](#_Toc30300)

[3.2 系统用例分析 6](#_Toc27354)

[3.3 系统功能分析 7](#_Toc3996)

[3.4 系统数据分析 9](#_Toc22800)

[3.5 业务流程分析 10](#_Toc23097)

[4 系统实现 13](#_Toc28708)

[4.1系统开发环境 13](#_Toc3172)

[4.2数据功能模块的实现 13](#_Toc24333)

[4.3注册登录功能的实现 17](#_Toc27776)

[4.4关注功能的实现 19](#_Toc204)

[4.5实时会话功能的实现 20](#_Toc26411)

[4.6用户评论功能的实现 22](#_Toc32439)

[4.7苗木地址解析功能的实现 23](#_Toc31432)

[4.8苗木资源推荐功能的实现 25](#_Toc5735)

[4.9苗木交易功能的实现 26](#_Toc19528)

[4.10统计分析功能的实现 28](#_Toc10201)

[4.11用户管理功能的实现 29](#_Toc5949)

[4.12苗木管理功能的实现 30](#_Toc5561)

[4.13订单管理功能的实现 30](#_Toc31613)

[5 系统测试 31](#_Toc4345)

[5.1系统测试原理 31](#_Toc31965)

[5.2注册登录功能测试用例 31](#_Toc29211)

[5.3关注功能测试用例 32](#_Toc28111)

[5.4实时会话功能测试用例 32](#_Toc5596)

[5.5用户评论功能测试用例 33](#_Toc1000)

[5.6苗木地址解析功能测试用例 33](#_Toc8223)

[5.7苗木交易功能测试用例 34](#_Toc3477)

[5.8用户管理功能测试用例 34](#_Toc24396)

[5.9订单管理功能测试用例 35](#_Toc14993)

[结束语 36](#_Toc7031)

[参考文献 36](#_Toc13520)

[致谢 38](#_Toc14360)

基于Vue3+NodeJS的苗木交易平台的设计与实现

摘要： 针对传统苗木产业中苗木资源分散、交易不明确、信息不对称和市场信息不透明等问题，设计与实现了一款基于Vue3 + NodeJS 的苗木交易平台。该平台基于MVC分层设计思想和RESTful API风格的Web服务架构，应用Vscode和NodeJS进行系统开发，使用MongoDB进行数据存储，特别说明的是系统支付采用了支付宝沙盒实现模拟支付，客户端通信采用了websocket通信技术，系统开发过程中聚焦数据交互。该平台由客户端和服务端两部分组成，客户端主要包括浏览苗木资源、苗木交易、实时会话、关注用户、个人中心等功能模块。服务端主要包括统计分析、用户管理、苗木管理、订单管理等功能模块。通过对系统各项功能模块的测试，测试结果良好，满足用户使用需求。

关键词：Vscode；Vue3；NodeJS；Express；MongoDB；MVC；MVVM

Design and Implementation of a Seedling Trading Platform Based on Vue3+NodeJS

Abstract: A tree trading platform based on Vue3 + NodeJS was designed and implemented to address the problems of scattered resources, unclear transactions, information asymmetry, and opaque market information in the traditional tree planting industry. The platform was based on the MVC layered design concept and the RESTful API style web service architecture, developed using Vscode and NodeJS, and used MongoDB for data storage. It is worth noting that the system payment used Alipay sandbox to simulate payment, and the client communication used websocket communication technology. The system development process focused on data interaction. The platform was composed of two parts: the client and the server. The client mainly included functions such as browsing tree resources, tree trading, real-time chat, following users, and personal center. The server mainly included functions such as statistical analysis, user management, tree management, and order management. Through testing of the various functional modules of the system, the test results were satisfactory and met user requirements.

Keywords: Vscode；Vue3；NodeJS；Express；MongoDB；MVC；MVVM

# 1 绪论

## 1.1 研究背景及意义

近年来的调查显示，目前苗木产业仍然大量采用传统模式进行运作。传统的苗木交易方式需要耗费大量人力、物力和时间成本，并且存在信息不对称的问题，导致交易双方难以获得准确的信息。目前我国有一些苗木相关平台，如苗木通、苗木网等，但这些平台主要面向大规模苗圃，对于小规模的个人苗木种植商和苗木买家提供渠道较少。因此，对于需求量并不大的苗木买家而言，仍然需要耗费大量时间在散落的村户中逐个寻找农户进行交易，这需要加快深度调整，将传统产业转型为现代产业[1]。因此，基于 Vue3+NodeJS 的苗木交易平台的设计与实现具有重要意义。该平台可以提高苗木交易的效率和质量，推动苗木行业的发展，满足市场需求，并具有广阔的应用前景。

## 1.2 研究目的

本设计是基于 Vue3+NodeJS 的苗木交易平台的设计与实现。该平台旨在聚合分散在各个村落中的苗木资源，实现苗木资源的信息化，降低需要雇佣劳动工所带来的成本，并更快更准确地满足苗木买卖双方的需求。通过平台统一的交易，可以提高散户苗木交易的确定性和安全性，避免了传统运作模式所带来的交易不确定性。该平台以 NodeJS 技术为基础进行开发，数据库使用 MongoDB，具有更灵活的数据模型和更高效的读写。同时，该平台还满足市场需求，探索新型苗木交易模型。平台实现了用户浏览、收藏、发布苗木资源、关注其他用户、苗木交易、在线会话等功能，并集成地图以满足用户大致了解苗木方位的需求，方便用户使用。

## 1.3 本文主要工作和内容

本文旨在研究如何实现一个苗木交易平台，该设计主要运用 NodeJS 语言进行开发，MongoDB 作为后台数据库存储数据。首先进行系统需求和可行性分析，确定系统特点和功能。接下来，根据数据库设计和系统开发需求进行实施。系统实现中需要确定系统开发环境以及核心功能，包括用户注册登陆、关注用户、在线沟通、苗木交易、苗木资源推荐、用户评论、苗木地址解析的实现。同时，还需要展示实现这些功能的核心代码。完成系统实现后，对其进行系统测试，测试系统运行是否稳定和正常，有无错误。最后进行简单总结，并标注引用到的参考文献。

## 1.4 论文组织结构

本文设计基于Vue3+NodeJS的苗木交易平台，其主要内容安排如下：

第一部分，绪论，包括设计背景、设计意义、研究目的、研究内容；

第二部分，相关技术，包括本系统开发过程中将使用到的技术与开发工具；

第三部分，需求分析与设计，包括业务流程分析、系统功能分析、系统数据设计；

第四部分，系统实现，包括开发环境相关配置与系统核心功能实现步骤；

第五部分，系统测试，包括对已完成系统各项功能的测试；

第六部分，结束语。

# 2 相关技术

## 2.1 NodeJS

NodeJS 是以 Chrome V8 为基础的 JavaScript 运行环境，使得 JavaScript 不再受浏览器的限制，并具有异步非阻塞 I/O、单线程、事件机制和跨平台等特点[2]。随着近年来 NodeJS 及其生态的发展，它已经成为后端开发的选择之一。

## 2.2 Express 服务端框架

Express 是 NodeJS 生态中的一个重要组成部分，作为 NodeJS 的老牌服务端框架之一，它提供了简单易用的 API 和实现机制，包括路由、中间件、错误处理等。Express 可以降低开发者开发应用的难度，提高应用开发效率，减少研发成本，并缩短应用发布周期[3]。

## 2.3 SocketIO 库

Socket.IO是一个基于NodeJS的实时网络通信库，通过浏览器websocket的新特性，它实现了客户端和服务器之间的实时双向通信。同时Socket.IO具有很好的可靠性，支持自动重连和容错机制，可以保证通信的可靠性和稳定性，并且封装了网络底层通信,支持多种网络通信协议,易于实现跨浏览器通信[4]。

2.4 JOSN Web Token

JWT全称为JSON Web Token，是为了在网络应用环境间传递声明而执行的一种基于JSON的开放标准[5]。JWT是目前广泛应用于用户鉴权的方式之一。

2.5 Vue3 框架

Vue3是一个流行的前端框架，采用MVVM[6]的设计模式，用于构建用户界面，数据驱动视图，从而让开发者更专注数据的流向，组合式API也将原先的结构模块化转变为功能模块化，让开发者更加灵活的组织构建应用。

2.6 MongoDB 数据库

MongoDB是一种流行的NoSQL（非关系型）数据库，是一个文档型数据库，具有高可扩展性、高性能、高灵活性等优点，还支持复杂的查询和聚合操作，可以处理大量数据，并支持横向扩展，可以轻松地扩展到各个节点。

2.7 MVC

MVC是一种常见的软件架构模式，它将程序分为三个核心部分：模型、视图和控制器。这三个部分各自处理不同的功能，彼此之间独立，但又相互协作，共同实现系统的目标[7]。

2.8 RESTful API

RESTful API是一种基于REST（Representational State Transfer）架构风格的应用程序接口设计规范。它使用HTTP协议中的GET、POST、PUT、DELETE等方法来实现对Web资源的增删改查操作，且风格简单、易于理解和扩展[8]。

2.9 vscode 开发环境

Visual Studio Code是一款轻量级文本编辑器和集成开发环境（IDE），它支持多种编程语言，并拥有丰富的扩展和插件来满足开发需求，可以满足从单个文件编辑到大型项目开发的需求，具有良好的性能和可扩展性。

# 3 需求分析与设计

3.1 可行性分析

（1）技术可行性

Vue3和NodeJS都是当今非常流行的技术栈，具有广泛的应用和强大的社区支持。同时，这两个技术都具有良好的性能和可扩展性，可以支持苗木交易平台的需求。

1. 数据库可行性

苗木交易平台需要处理大量的数据，包括商品信息、用户信息、订单信息等等。因此，需要选择一个可靠的数据库来存储和管理这些数据。目前，常用的数据库包括MySQL、MongoDB等，这些数据库都可以与NodeJS很好地集成。

1. 竞争对手分析

在设计和实现苗木交易平台之前，需要进行竞争对手分析，了解市场上已经存在的苗木交易平台，了解它们的特点和优势，从而更好地设计自己的平台。

1. 用户需求分析

苗木交易平台的设计和实现需要考虑到用户的需求，包括易用性、安全性、速度等等。需要进行用户调研，了解用户的需求和偏好，从而更好地设计和实现苗木交易平台。

1. 成本可行性

苗木交易平台的设计和实现需要投入一定的人力和财力。需要进行成本分析，了解开发和维护苗木交易平台的成本，并进行预算和规划。

3.2 系统用例分析

本系统包括两种不同的用户角色：普通用户和管理员用户，他们各有不同的需求。对于普通用户而言，他们需要登录系统后可以查看苗木资源、对相关苗木进行评论、进行苗木交易、发布苗木资源、关注其他用户等操作。而管理员用户则具备更多的权限，例如能够对用户信息和苗木信息进行增删改查以及审核操作，查看订单信息和统计分析图表等。

（1）普通用户需求用例如图3-1所示。

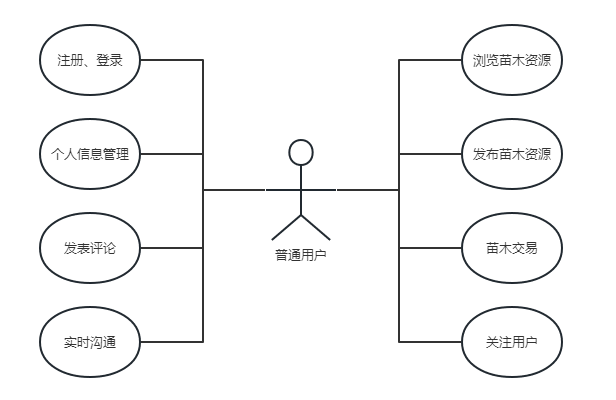


图3-1 普通用户需求用例图

（2）管理员需求用例如图3-2所示。

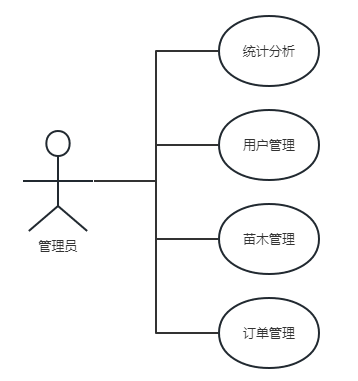


图3-2 管理员需求用例图

## 3.3 系统功能分析

基于Vue3 + NodeJS 的苗木交易平台是一款真实、安全、透明的平台，以实现苗木资源共享、在线会话、苗木交易。使用MongoDB作为后台数据库，使用Vue3进行前端页面开发，使用NodeJS作为后台技术，实现页面的数据展示以及数据交互。通过分析苗木种植商和苗木买家的需求，制定系统的主要功能与模块，最终逐步实现系统。

基于Vue3 + NodeJS 的苗木交易平台的主要功能详细描述如下：

（1）前台部分功能模块：

注册登录模块：游客进入注册页面后，输入必填的信息即可完成注册，用户进入登录界面，输入账号密码即可完成登录操作。

首页模块：用户可登录查看苗木资源，系统根据用户数据可进行推荐苗木资源，也可以选择浏览用户所在地区的苗木资源。

动态中心模块：用户可登录查看关注列表以及被关注用户所发布的苗木资源。

个人中心模块：用户可登录查看个人浏览历史记录、查看个人收藏的苗木资源、查看个人交易。

个人空间模块：用户可登录发布苗木资源、管理个人苗木资源、修改个人资料。

会话中心模块：用户可登录联系用户，进行会话，在线聊天，进行苗木交易。

本系统所有前台功能模块如图3-3所示。

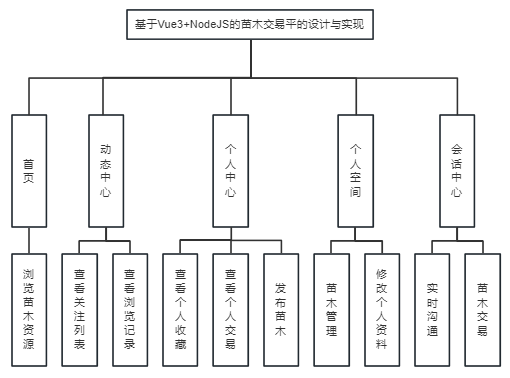


图3-3 前台功能模块图

（2）后台部分功能模块：

首页模块：管理员可查看统计的用户数量、苗木数量、周交易量、以图表的形式展示苗木种类以及每周交易数量。

用户管理模块：管理员可审核注册用户、管理用户信息。

苗木管理模块：管理员可发布苗木资源、管理苗木信息、审核用户发布的苗木资源。

订单管理模块：管理员可查看订单的详细信息，删除相关订单。

本系统所有后台功能模块如图3-4所示。

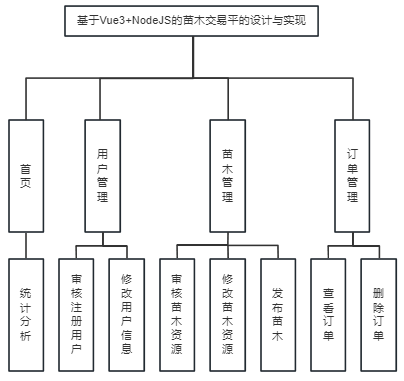


图3-4 后台功能模块图

## 3.4 系统数据分析

E-R图（Entity-Relationship Diagram）又称实体-关系图，是一种用于表示实体、属性和它们之间关系的图形化工具，在数据库设计过程中，E-R图是MIS数据库结构设计常用的方法之一。合理的E-R图设计会直接减少数据库中表的冗余，加大数据库资源的有效利用。本系统所有数据库表E-R图如图3-5所示。

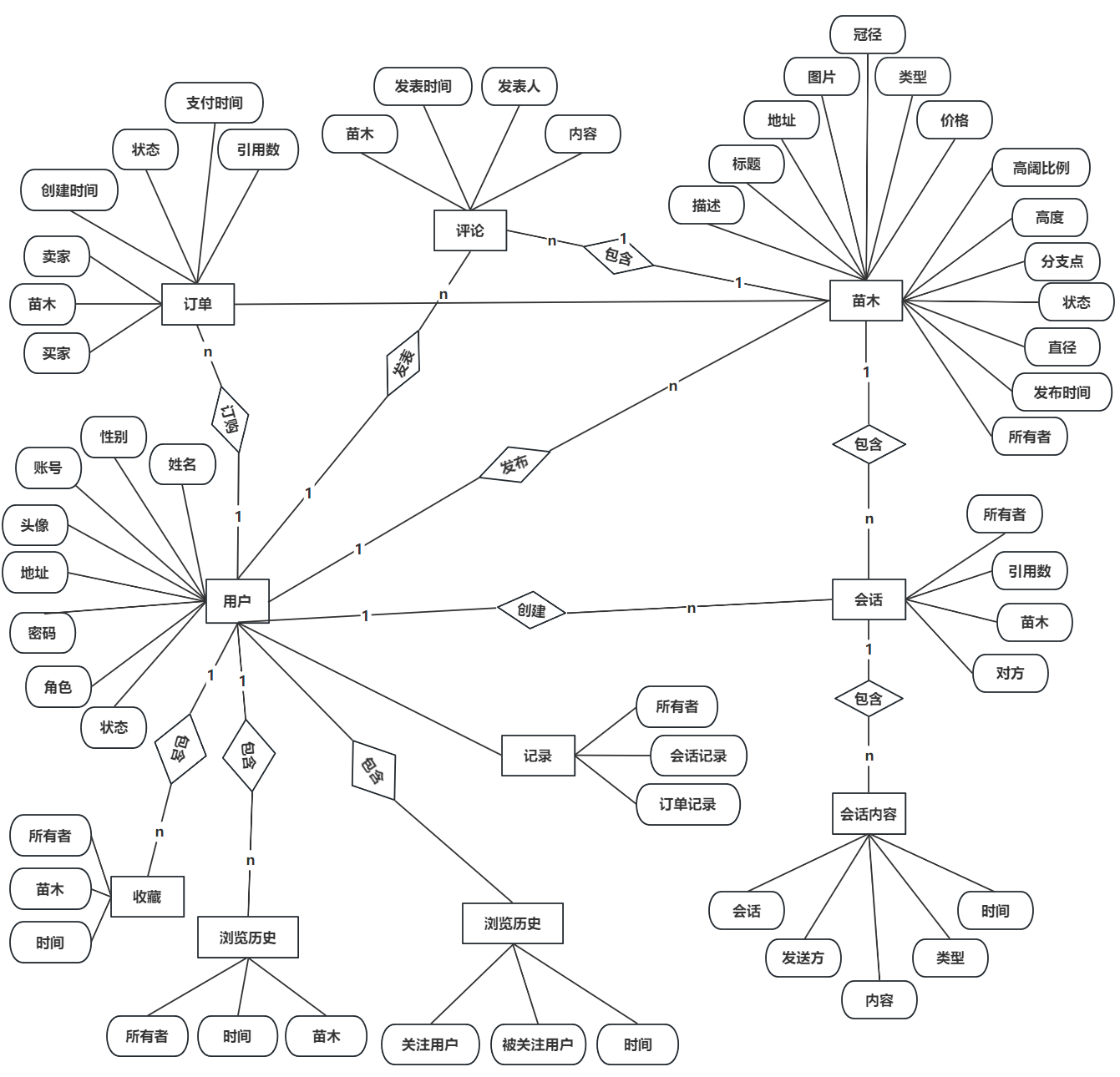


图3-5 用户信息实体图

3.5 业务流程分析

前台功能主要包括浏览苗木资源、查看苗木地址、收藏苗木、关注用户、个人中心、发布苗木资源、实时会话等功能。本系统前台功能流程图如图3-6所示。

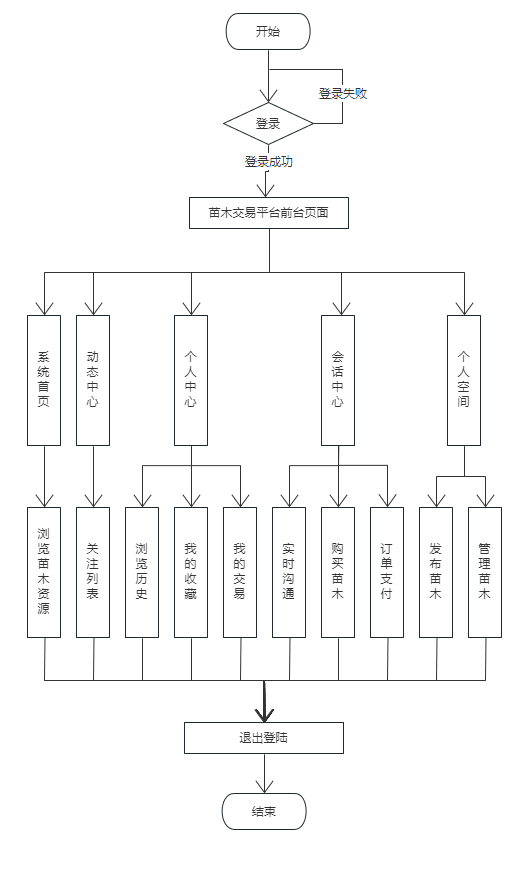


图3-6 系统前台功能流程图

前台部分：

（1）在浏览器中输入网址进入网站首页，进行登录或者注册。

（2）用户可以点击侧边栏导航，选择平台首页、动态中心、个人中心、个人空间、会话中心。

后台功能主要包括统计分析、用户管理、苗木管理、订单管理。本系统后台功能流程图如图3-2所示。

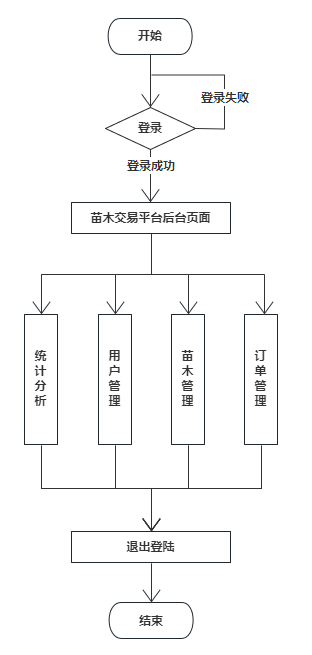


图3-2 系统后台功能流程图

后台部分：

1. 在浏览器中输入网址进入网站首页。
2. 输入账号密码，登录到苗木交易平台的后台，在侧边栏可选择操作对应的功能。
3. 根据相关的功能模块，获取对应的数据进行修改。

# 4 系统实现

## 4.1系统开发环境

本系统基于vscode环境开发，利用MongoDB作为数据库存储数据、socket.io 实现实时通讯、使用 JWT（JSON Web Token）实现用户认证与授权。

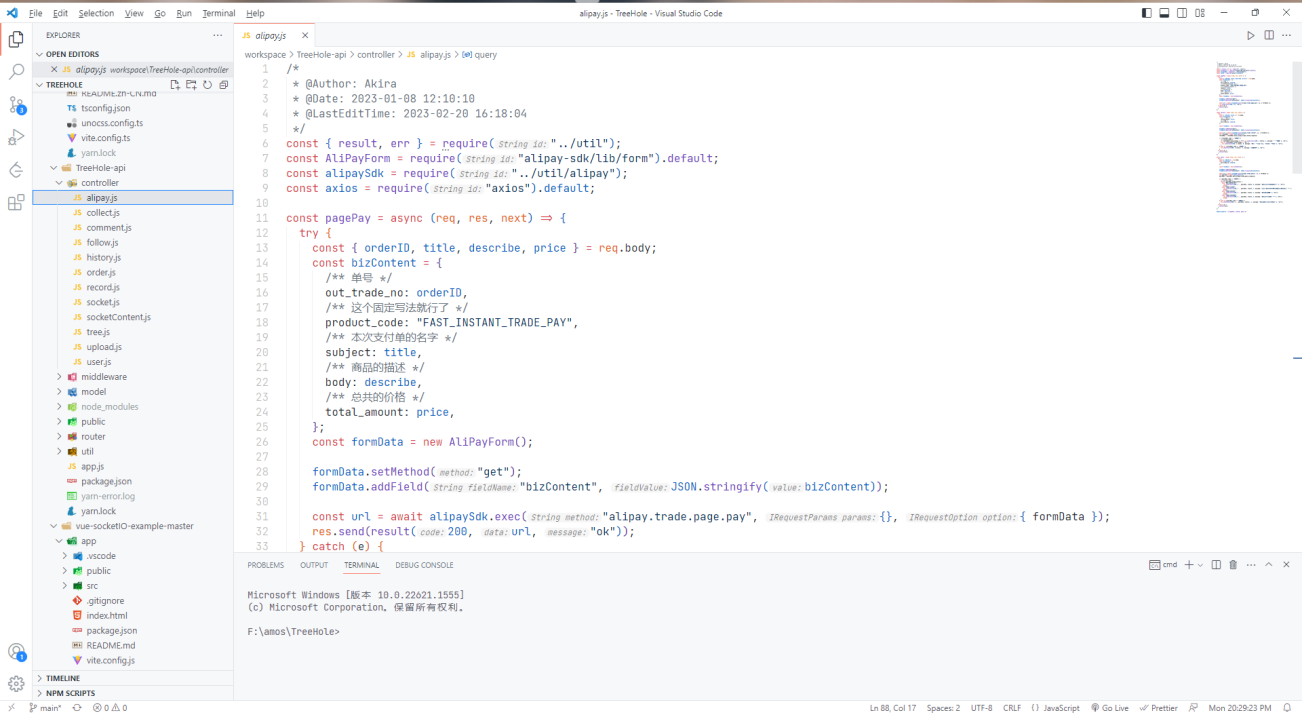


图4-1 vscode 开发环境

## 4.2数据功能模块的实现

本系统采用的是MongoDB数据库，是一个非关系型数据库。在本系统中使用到了多张表，如用户信息表、苗木信息表、评论表、个人记录表、会话信息表、会话内容表、订单信息表、浏览记录信息表、关注信息表、收藏信息表。数据库连接部分通过调用mongoose.connect进行数据库连接，调用mongoose.model创建数据库表。

1. 数据库连接

mongoose.connect(URL, {

useNewUrlParser: true,

});

mongoose.connection.on("connected", () => {

console.log("mongoose connection success");

});

mongoose.connection.on("error", (error) => {

console.log(`mongoose connection error: ${error}`);

});

mongoose.connection.on("disconnected", () => {

console.log("Mongoose connection disconnected");

});

1. 数据表的创建

User: mongoose.model("User", require("./User"), "User"),

Tree: mongoose.model("Tree", require("./Tree"), "Tree"),

Record: mongoose.model("Record", require("./Record"), "Record"),

Order: mongoose.model("Order", require("./Order"), "Order"),

Socket: mongoose.model("Socket", require("./Socket"), "Socket"),

Comment: mongoose.model("Comment", require("./Comment"), "Comment"),

SocketContent: mongoose.model("SocketContent", require("./SocketContent"), "SocketContent"),

History: mongoose.model("History", require("./History"), "History"),

Collect: mongoose.model("Collect", require("./Collect"), "Collect"),

Follow: mongoose.model("Follow", require("./Follow"), "Follow"),

（3）具体展示部分如下所示。

表4-1 用户信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 是否主键 | 说明 |
| \_id | ObjectId | 是 | 编号 |
| account | String | 是 | 账号 |
| password | String | 否 | 密码 |
| name | String | 否 | 用户名 |
| sex | String | 否 | 性别 |
| location | String | 否 | 地区 |
| avator | String | 否 | 头像 |
| role | String | 否 | 角色 |
| status | String | 否 | 状态 |

表4-2 苗木信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 是否主键 | 说明 |
| \_id | ObjectId | 是 | 编号 |
| ownerID | String | 否 | 所有者id |
| type | String | 否 | 种类 |
| height | String | 否 | 高度 |
| diameter | String | 否 | 直径 |
| branchPoint | String | 否 | 分支点 |
| crownDiameter | String | 否 | 冠径 |
| location | String | 否 | 地址 |
| title | String | 否 | 标题 |
| describe | String | 否 | 描述 |
| price | String | 否 | 价格 |
| imgs | Array | 否 | 图片 |
| time | String | 否 | 时间 |
| hci | Double | 否 | 高阔比例 |
| status | String | 否 | 状态 |

表4-3 评论表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 是否主键 | 说明 |
| \_id | ObjectId | 是 | 编号 |
| treeID | String | 否 | 苗木id |
| senderID | String | 否 | 评论人id |
| context | String | 否 | 评论内容 |
| time | String | 否 | 评论时间 |

表4-4 个人记录表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 是否主键 | 说明 |
| \_id | ObjectId | 是 | 编号 |
| userID | String | 否 | 用户id |
| order | Array | 否 | 订单记录 |
| Socket | Array | 否 | 会话记录 |

表4-5 会话信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 是否主键 | 说明 |
| \_id | ObjectId | 是 | 编号 |
| userID1 | String | 是 | 会话用户1 |
| userID2 | String | 是 | 会话用户2 |
| treeID | String | 是 | 苗木id |
| tree | Object | 否 | 苗木快照 |
| refer | Int32 | 否 | 会用引用数 |

表4-6 会话内容信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 是否主键 | 说明 |
| \_id | ObjectId | 是 | 编号 |
| socketID | String | 否 | 会话id |
| senderID | String | 否 | 发送方id |
| content | String | 否 | 会话内容 |
| type | String | 否 | 内容类型 |
| time | String | 否 | 时间 |

表4-7 订单信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 是否主键 | 说明 |
| \_id | ObjectId | 是 | 编号 |
| treeID | String | 是 | 苗木id |
| buyerID | String | 否 | 买家id |
| sellerID | String | 否 | 卖家id |
| tree | Object | 否 | 苗木快照 |
| time | String | 否 | 创建时间 |
| payTime | String | 否 | 付款时间 |
| refer | Int32 | 否 | 订单引用数 |
| status | String | 否 | 状态 |

表4-8 关注信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 是否主键 | 说明 |
| \_id | ObjectId | 是 | 编号 |
| fromUserID | String | 是 | 关注者id |
| toUserID | String | 是 | 被关注者id |
| Time | String | 否 | 时间 |

表4-9 收藏信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 是否主键 | 说明 |
| \_id | ObjectId | 是 | 编号 |
| userID | String | 是 | 用户id |
| treeID | String | 是 | 苗木id |
| Time | String | 否 | 时间 |

表4-10 浏览记录信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 是否主键 | 说明 |
| \_id | ObjectId | 是 | 编号 |
| userID | String | 是 | 用户id |
| treeID | String | 是 | 苗木id |
| Time | String | 否 | 时间 |

## 4.3注册登录功能的实现

在此实现用户登录注册功能，要求用户在进入系统主页前必须进行登录操作。如果用户没有该系统账号，则需要进入注册页面进行注册。其中，登录功能的实现，定义了Login函数，通过调用接口将账号密码传递给服务端，服务端使用User模型的findOne方法进行匹配验证。而注册功能的实现，定义了Register函数，服务端将前端传递的账号通过User模型的findOne方法判断是否重复，如果不存在重复，则调用save方法进行数据存储。注册页面如图4-2所示。登录页面如图4-3所示。

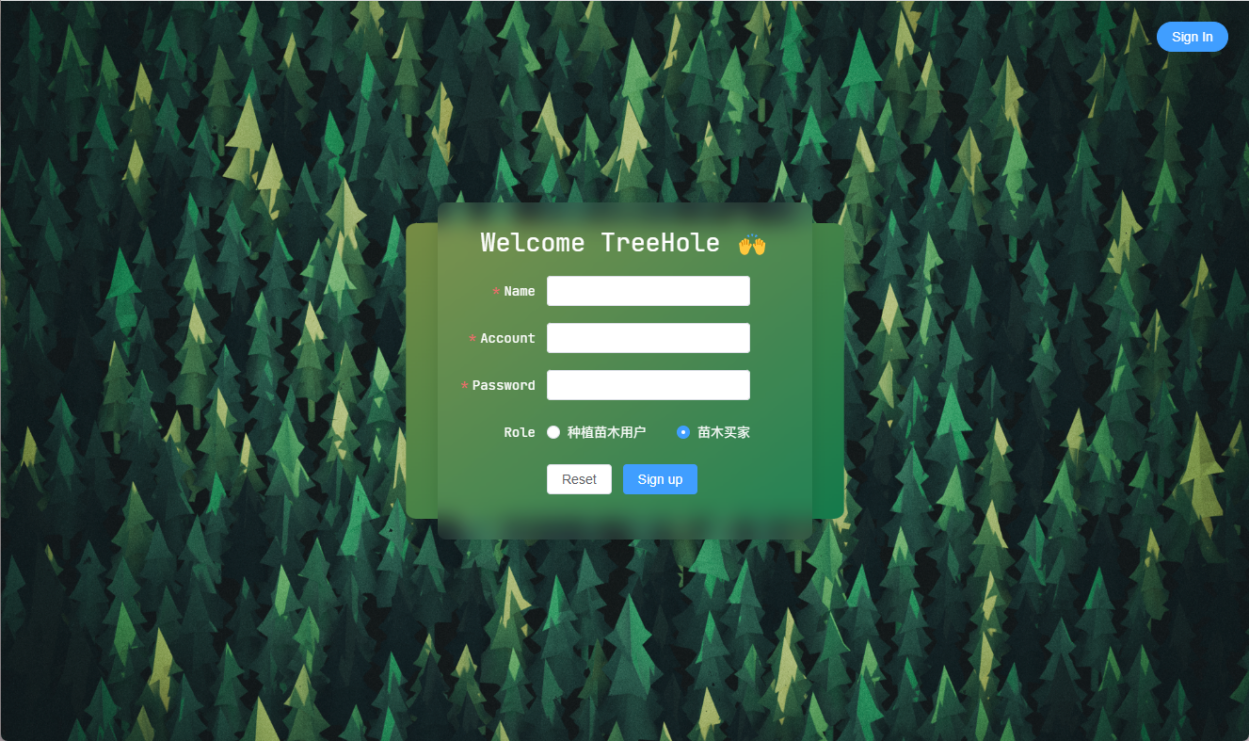


图4-2 用户注册页面

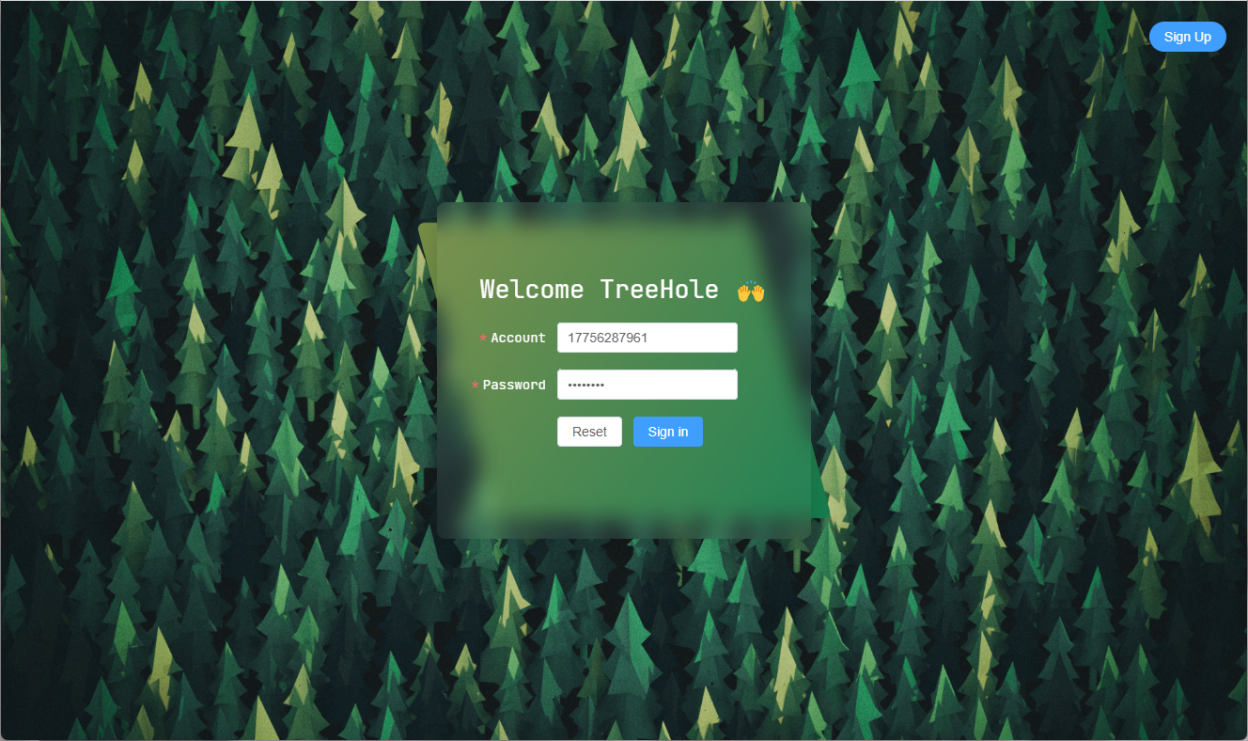


图4-3 用户登录页面

（1）登录功能核心代码如下：

const { account, password } = req.body;

const user = await User.findOne({ account });

if (!user) {

next(err("用户不存在，请先注册！", 401, null));

return;

}

if (password !== user.password) {

next(err("密码错误，请重新输入！", 401, null));

return;

}

if(user.status == '0') {

next(err("用户待审核中...", 401, null));

return;

}

（2）注册功能核心代码如下：

const { account } = req.body;

let user = await User.findOne({ account });

if (user) {

next(err("注册用户已存在！", 403, null));

return;

}

user = new User(req.body);

user = await user.save();

const record = new Record({ userID: user.\_id });

await record.save();

## 4.4关注功能的实现

用户可通过点击关注或者取消关注按钮，实现对其他用户的关注或取消关注，然后在动态页面可以查看当前用户的关注列表以及被关注用户所发布的苗木。关注功能的实现，定义了addFollow、removeFollow函数，调用Follow模型的findOne方法获取用户关注记录，若未关注该用户，则new Follow新建实例，通过调用save方法保存数据，若已关注，提示“该用户已被关注”。removeFollow函数中，通过调用findOneAndRemove进行关注数据删除，实现取消关注功能。关注与取消关注效果如图4-4所示。

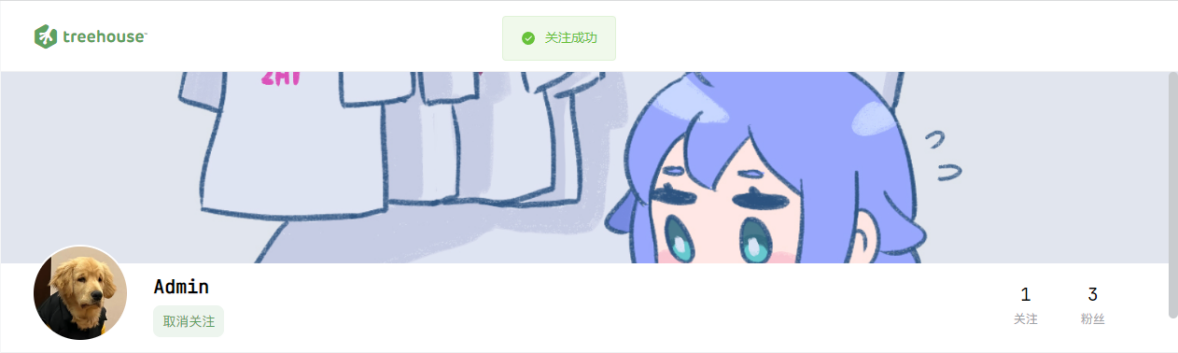


图4-4 关注用户功能

其核心代码如下：

1. 关注

const { fromUserID, toUserID } = req.body;

const follow = await Follow.findOne({ fromUserID, toUserID });

if (follow) {

res.send(result(401, follow, "已关注该用户！"));

return;

}

const newFollow = new Follow(req.body);

const data = await newFollow.save();

res.send(result(200, data, "ok"));

1. 取消关注

const { fromUserID, toUserID } = req.body;

const follow = await Follow.findOne({ fromUserID, toUserID });

if (follow) {

res.send(result(401, follow, "已关注该用户！"));

return;

}

const newFollow = new Follow(req.body);

const data = await newFollow.save();

res.send(result(200, data, "ok"));

## 4.5实时会话功能的实现

用户可点击联系卖家进入到会话页面，与用户进行在线会话，实时沟通。实时会话功能实现，首先调用socket.io原型上的on方法创建长连接服务器，并在前台监听sendMsg事件以及定义回调socketCallback，回调函数中调用socket.emit方法触发sendMsg事件，进而会话服务器会通知每一个参与者进行数据缓存与数据存储，数据存储方面则是服务端将前端传递的会话内容创建新的socketContent实例，并调用save方法存储到数据库。实现原理是通过socket.io利用websocket进行服务端事件监听，并让客户端与服务端进行连接，并且在必要时取消事件的监听并且断开连接，避免重复监听，从而实现客户端间实时收发数据。会话页面如图4-5所示。

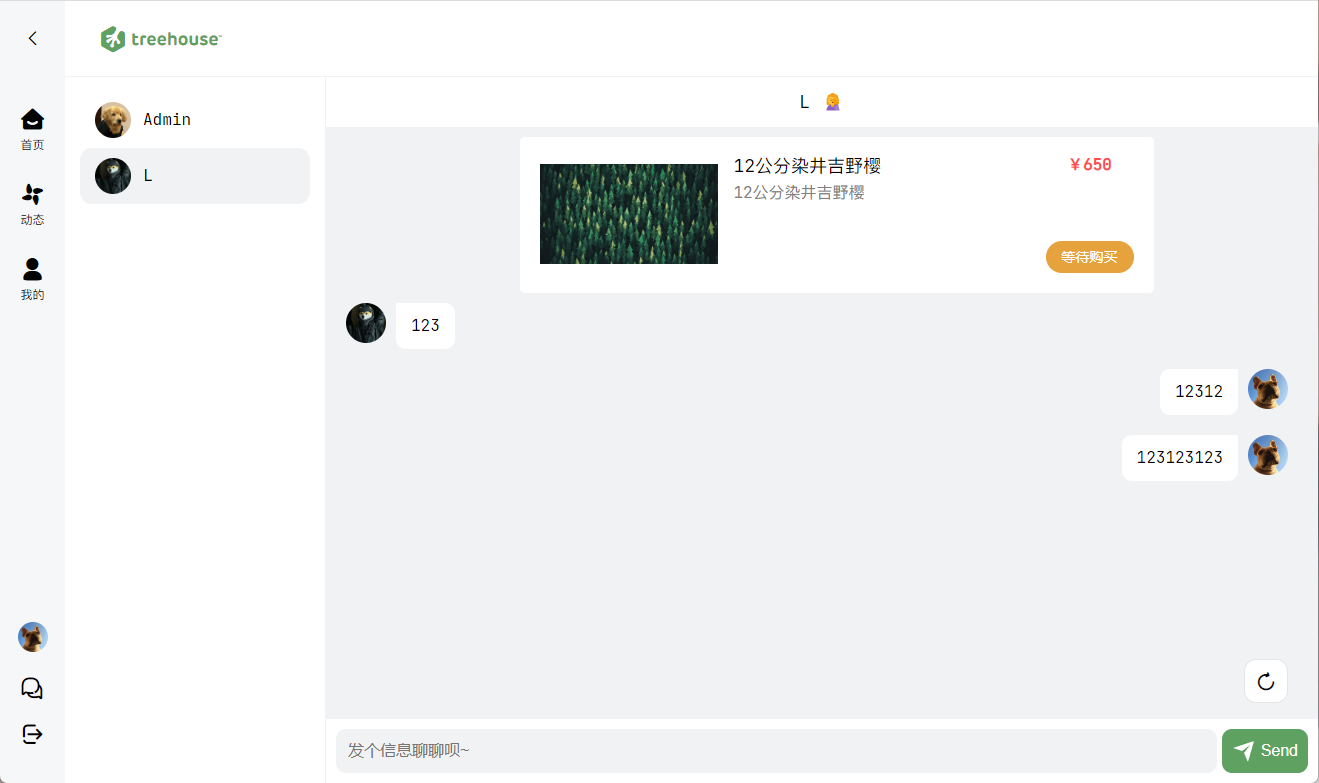


图4-5 会话页面

其核心代码如下：

（1）Socket服务器

const io = require("socket.io")(http, { cors: true });

const port = process.env.PORT || 3000;

io.on("connection", (socket) => {

socket.on("sendMessage", (content) => {

io.emit("sendMessage", content);

});

});

http.listen(port, () => {

console.log(`Socket.IO server running at http://localhost:${port}/`);

});

（2）客户端连接与事件监听

const socket = io("ws://localhost:3000");

const socketCallback = async (content) => {

if (currentSocket.value.\_id == content.socketID) state.socketContent.push(content);

if (content.senderID == loginUser.\_id) await request.post(api.socketContent.addSocketContent, content);

downScroll();

};

socket.on("sendMessage", socketCallback);

1. 客户端发送信息

const sendMsg = async () => {

if (state.text == "") {

ElMessage.warning("信息内容不能为空喔~");

return;

}

const content = {

socketID: currentSocket.value.\_id,

senderID: loginUser.\_id,

context: state.text,

type: 1,

};

socket.emit("sendMessage", content);

state.text = "";

};

1. 会话存储

const companionSocket = async (data) => {

const { userID1, userID2 } = data;

const records = await Record.find({ userID: { $in: [userID1, userID2] } });

records[0].socket.unshift(data.\_id.toString());

records[1].socket.unshift(data.\_id.toString());

await Record.findByIdAndUpdate(records[0].\_id, records[0]);

await Record.findByIdAndUpdate(records[1].\_id, records[1]);

};

const { userID1, userID2, treeID } = req.body;

let socket = await Socket.findOne({ $or: [{ $and: [{ userID1 }, { userID2 }, { treeID }] }, { $and: [{ userID1: userID2 }, { userID2: userID1 }, { treeID }] }] });

if (socket) {

if (socket.refer != 2) {

const records = await Record.find({ userID: { $in: [userID1, userID2] } });

const index\_0 = records[0].socket.indexOf(socket.\_id);

const index\_1 = records[1].socket.indexOf(socket.\_id);

if (index\_0 != -1) records[0].socket.splice(index\_0, 1);

if (index\_1 != -1) records[1].socket.splice(index\_1, 1);

records[0].socket.unshift(socket.\_id.toString());

records[1].socket.unshift(socket.\_id.toString());

await Record.findByIdAndUpdate(records[0].\_id, records[0]);

await Record.findByIdAndUpdate(records[1].\_id, records[1]);

socket.refer = 2;

await Socket.findByIdAndUpdate(socket.\_id, { refer: 2 });

}

} else {

socket = new Socket(req.body);

const data = await socket.save();

await companionSocket(data);

}

（5）会话内容存储

const { socketID } = req.body;

const socket = await Socket.findOne({ \_id: socketID });

if (!socket) {

next(result(401, socket, "会话不存在！"));

return;

}

const socketContent = new SocketContent(req.body);

const content = await socketContent.save();

## 4.6用户评论功能的实现

用户可在苗木详情页面发表评论，并且用户可删除自己发表的评论。评论功能的实现，定义了addComment函数，服务端接收前台传递的评论内容，通过创建新的Comment模型的实例，并调用save方法进行数据存储。实现方式主要通过系统调用接口将评论内容传给服务端，首先判断苗木是否存在，再通过save方法进行数据存储，删除评论则是传递评论id进行相关评论的删除。苗木详情页面如图4-6所示。

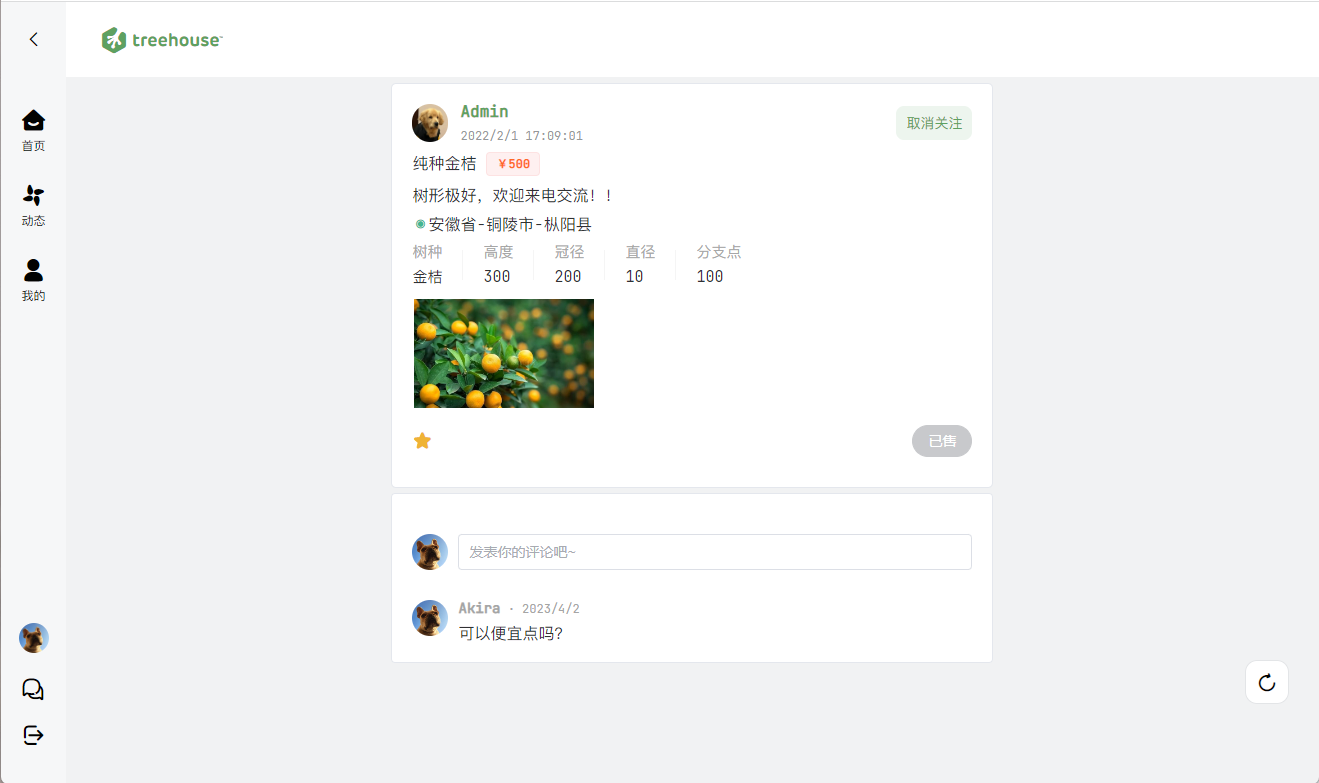


图4-6 苗木详情页面

const { treeID } = req.body;

const tree = await Tree.findOne({ \_id: treeID });

if (!tree) {

next(result(401, tree, "该苗木已被删除！"));

return;

}

const comment = new Comment(req.body);

const data = await comment.save();

## 4.7苗木地址解析功能的实现

用户可在苗木详情页面点击苗木详细地址，在地图中查看苗木的大体位置。地址解析功能的实现，定义了geocoder正逆解析类以及convert转换函数，通过集成腾讯地图API，建立正逆地址解析类，调用geocoder.getLocation获取中文地址解析的结果，并通过setCenter以及updateGeometries方法重新渲染地图以及中心点，从而实现地图中查看苗木的地址。苗木地址页面如图4-7所示。



图4-7 苗木地址页面

其核心代码如下：

const map = new TMap.Map("container", {

zoom: 14,

center: new TMap.LatLng(39.986785, 116.301012),

});

const geocoder = new TMap.service.Geocoder();

const markers = new TMap.MultiMarker({

map: map,

geometries: [],

});

function convert(location) {

markers.setGeometries([]);

geocoder.getLocation({ address: location }).then((result) => {

markers.updateGeometries([

{

id: "container",

position: result.result.location,

},

]);

map.setCenter(result.result.location);

console.log(result.result.location);

});

}

## 4.8苗木资源推荐功能的实现

系统可根据用户的历史浏览记录进行推荐相关的苗木资源。苗木资源推荐功能的实现，定义了getRecommendTreeList函数，服务端接收前台传递的用户浏览记录，进行数据分析，主要根据用户浏览记录的部分数据进行分析计算用户近期喜好苗木的高阔比例均值，调用Tree模型的find方法数据筛选并将数据返回给前台展示。苗木资源推荐页面如图4-8所示。

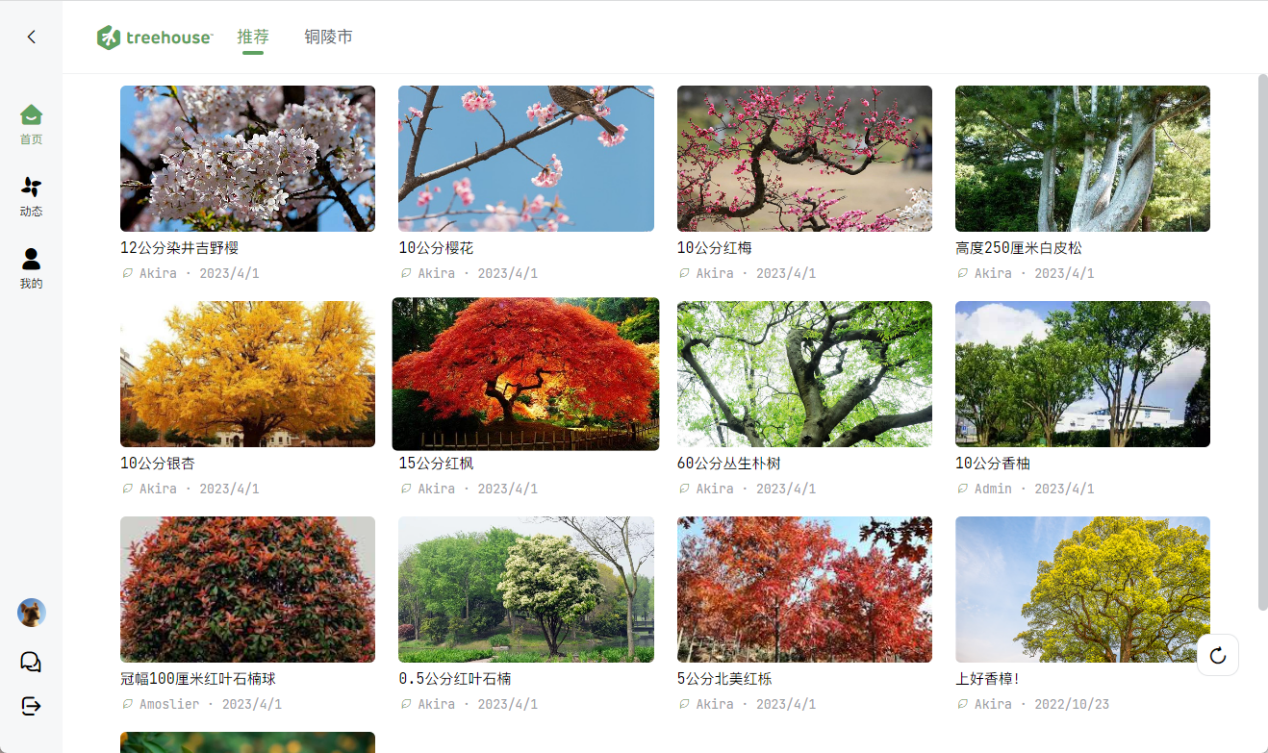


图4-8 苗木资源推荐页面

let hci\_gt = 0;

let hci\_lt = 5;

let data;

if (trees.length >= 10) {

trees = trees.slice(0, 10);

data = await Tree.find({ \_id: { $in: trees } });

let hci = 0;

data.forEach((item) => {

hci += item.hci;

});

hci = hci / data.length;

hci\_gt = (hci - 0.5).toFixed(2);

hci\_lt = (hci + 0.5).toFixed(2);

if (hci\_gt < 0) hci\_gt = 0;

}

data = await Tree.find({ $and: [{ hci: { $gt: hci\_gt, $lt: hci\_lt } }, { status: "0" }] })

.sort({ \_id: -1 })

.skip((pageNo - 1) \* limit)

.limit(limit);

const list = await mergeTrees(data);

## 4.9苗木交易功能的实现

用户可通过平台进行苗木交易，在会话页面点击立即购买并跳转支付宝支付页面进行支付购买苗木，并且在苗木交易未完成前，买卖双方均可以进行取消订单，并退还钱款，且当某用户购买过此苗木后，其他用户将无法购买，仅交易双方可见，这也是避免传统苗木运行模式下出现的恶性竞争的一种方式。苗木交易功能的实现，定义了pagePay、refund、query三个函数，分别调用支付宝沙盒的页面支付、退款、支付查询接口，首先调用AliAPayForm的构造函数创建新的实例，调用setMethod方法确定请求方式，通过addField添加请求参数，再通过axios调用接口并获取交易结果。进而模拟整个支付流程。苗木交易订单详情页面如图4-9所示。



图4-9 苗木交易详情页面

其核心代码如下：

1. 页面支付

const bizContent = {

out\_trade\_no: orderID, //单号

product\_code: "FAST\_INSTANT\_TRADE\_PAY",

subject: title,

body: describe,

total\_amount: price,

};

const formData = new AliPayForm();

formData.setMethod("get");

formData.addField("bizContent", JSON.stringify(bizContent));

const url = await alipaySdk.exec("alipay.trade.page.pay", {}, { formData });

1. 支付查询

const bizContent = {

refund\_amount: price,

out\_trade\_no: orderID,

};

const formData = new AliPayForm();

formData.setMethod("get");

formData.addField("bizContent", JSON.stringify(bizContent));

const url = await alipaySdk.exec("alipay.trade.refund", {}, { formData });

let refundRes = await axios.get(url);

refundRes = refundRes.data.alipay\_trade\_refund\_response;

if (refundRes.code == "10000") {

if (refundRes?.fund\_change == "Y") res.send(result(200, { status: 1, message: "退款成功" }, "ok"));

else if (refundRes?.fund\_change == "N") {

res.send(result(200, { status: 0, message: "正在退款，请稍后进一步确认退款状态" }, "ok"));

}

} else if (refundRes.code == "20000") {

res.send(result(500, { status: -1, message: "系统繁忙" }, "ok"));

}

1. 退款

const bizContent = {

out\_trade\_no: orderID,

};

const formData = new AliPayForm();

formData.setMethod("get");

formData.addField("bizContent", JSON.stringify(bizContent));

const url = await alipaySdk.exec("alipay.trade.query", {}, { formData });

let queryRes = await axios.get(url);

queryRes = queryRes.data.alipay\_trade\_query\_response;

if (queryRes.code == "10000") {

switch (queryRes.trade\_status) {

case "WAIT\_BUYER\_PAY":

res.send(result(200, { ...queryRes, status: 0, massage: "交易创建，等待买家付款" }, "ok"));

break;

case "TRADE\_CLOSED":

res.send(result(200, { ...queryRes, status: 1, massage: "未付款交易超时关闭，或支付完成后全额退款" }, "ok"));

break;

case "TRADE\_SUCCESS":

res.send(result(200, { ...queryRes, status: 2, massage: "交易支付成功" }, "ok"));

break;

case "TRADE\_FINISHED":

res.send(result(200, { ...queryRes, status: 3, massage: "交易结束，不可退款" }, "ok"));

break;

}

} else if (queryRes.code == "40004") {

res.send(result(200, { ...queryRes, status: -1, massage: "交易不存在，请立即支付" }, "ok"));

}

## 4.10统计分析功能的实现

管理员用户可登录管理系统，点击统计分析按钮，查看整个苗木交易平台的数据统计。统计分析功能的实现，定义了dataAnalysis函数，通过new Data() 方法获取近一周的起止时间，再通过Promise.all() 分别执行各个模型的find方法获取相应的数据，以及通过map集合进行整合，由于服务端不能传递map类型的数据，最后使用Array.from()进行转换为普通数组，并返回给客户端。统计分析页面如图4-10所示。



图4-10 统计分析页面

其核心代码如下：

const endTime = new Date();

let startTime = new Date(endTime.getTime() - 24 \* 60 \* 60 \* 10 \* 1000);

const data = await Promise.all([User.count(), Tree.count(), Order.count(), Order.find({ time: { $gte: startTime, $lte: endTime } }), Tree.find().sort({ \_id: -1 }).limit(100)]);

const orders = data[3];

const trees = data[4];

const weeklyVolume = [0, 0, 0, 0, 0, 0, 0];

orders.forEach((item) => { weeklyVolume[item.time.getDay()]++; });

let popularType = new Map();

trees.forEach((item) => {

let type = item.type;

if (popularType.has(type)) popularType.set(type, popularType.get(type) + 1);

else popularType.set(type, 1);

});

popularType = Array.from(popularType, ([key, value]) => ({ name: key, value }));

## 4.11用户管理功能的实现

管理员用户可登录管理系统，点击用户管理按钮，对用户数据进行修改，包括新增用户、删除用户、修改用户信息、审核用户，且可以根据账号或者用户名查找相关用户。用户管理功能的实现，分别定义了addUser、removeById、modifyById、getUserList的函数。addUser中将客户端传递的数据作为参数，通过调用new User() 新建实例，再通过save方法存储数据。removeById 中通过调用findByIdAndDelete方法进行删除用户。modifyById中将客户端传递来的id以及需要修改用户信息作为参数，通过调用findByIdAndUpdate方法实现用户信息修改。getUserList中通过将客户端传递的account或name的数据，采用new RegExp() 创建正则表达式，并通过User.find方法查询对应的用户并返回。用户管理页面如图4-11所示。

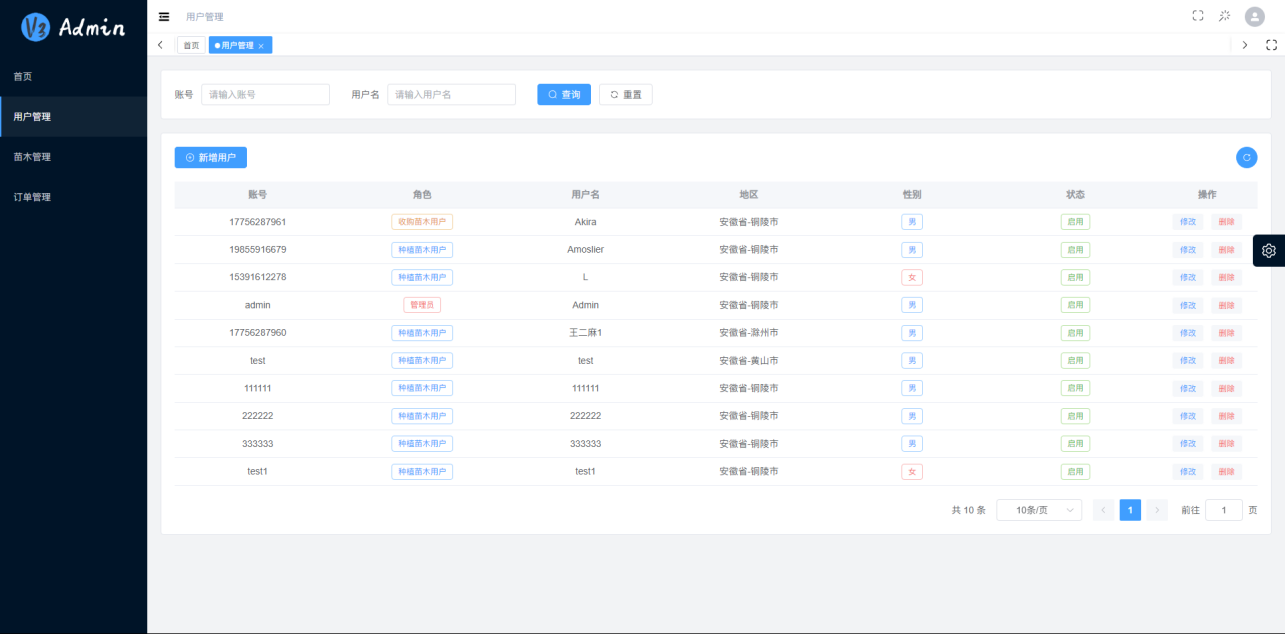


图4-11 用户管理页面

## 4.12苗木管理功能的实现

管理员用户可登录管理系统，点击苗木管理按钮，对苗木数据进行修改，包括新增苗木、删除苗木、修改苗木信息、审核苗木，且可以根据苗木类型或者地区查找相关苗木。苗木管理功能的实现，分别定义了addTree、removeById、modifyById、getTreeList的函数。addTree中将客户端传递的数据作为参数，通过调用new Tree() 新建实例，再通过save方法存储数据。removeById 中通过调用findByIdAndDelete方法进行删除苗木。modifyById中将客户端传递来的id以及需要修改的苗木信息作为参数，通过调用findByIdAndUpdate方法实现苗木信息修改。getTreeList中通过将客户端传递的type或location的数据，采用new RegExp() 创建正则表达式，并通过Tree.find方法查询对应的苗木并返回。苗木管理页面如图4-12所示。

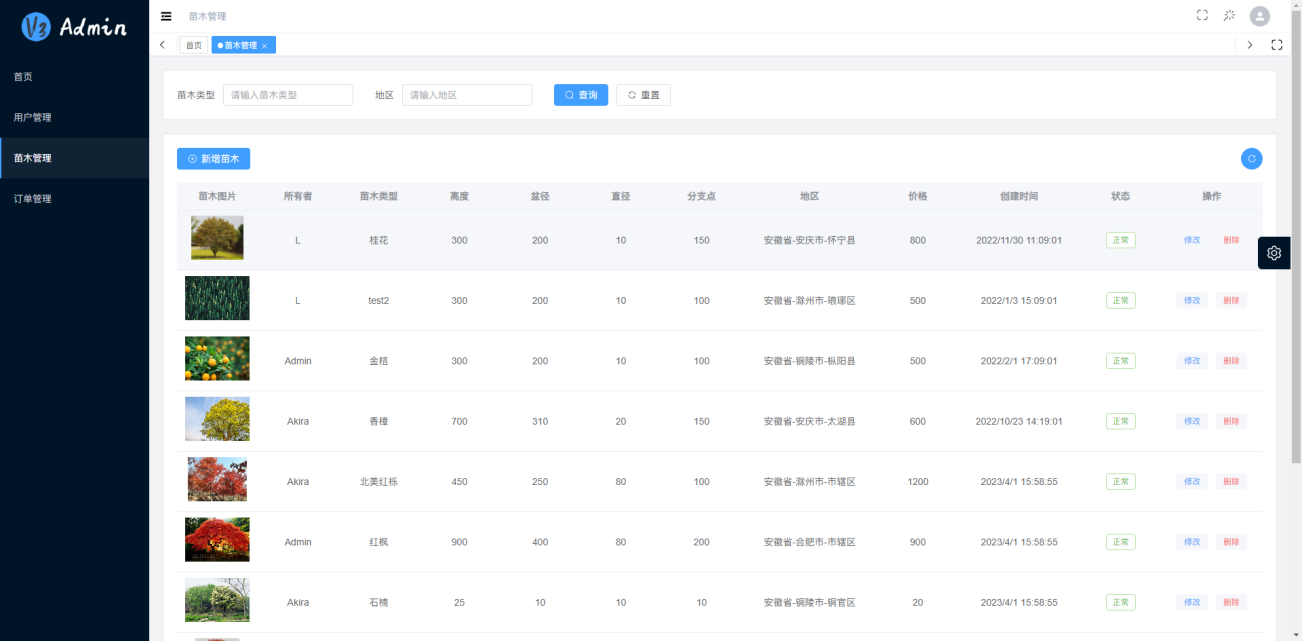


图4-12 苗木管理页面

## 4.13订单管理功能的实现

管理员用户可登录管理系统，点击订单管理按钮，查看订单信息以及删除相关订单，且可以根据苗木类型或者苗木所有者查找相关订单。订单管理功能的实现，分别定义了getOrderList、removeById、的函数。getOrderList中通过将客户端传递的type或userID的数据，采用new RegExp() 创建正则表达式，并通过Order.find方法查询对应的订单并返回。removeById 中通过调用findByIdAndDelete方法进行删除苗木，并且客户端会判断订单的状态，仅已完成的订单可以删除，否则将提示“订单正在进行中...”。订单管理页面如图4-13所示。

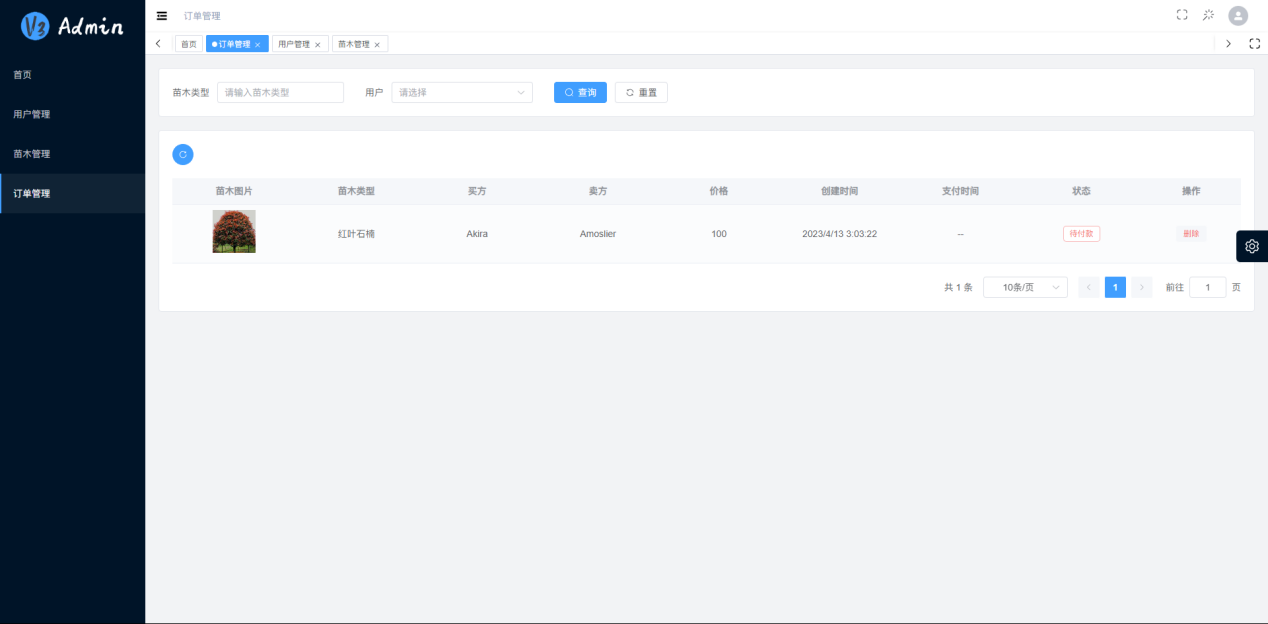


图4-13 订单管理页面

# 5 系统测试

5.1系统测试原理

随着信息产业的不断发展完善，在软件研发中，软件检测的重要性不断凸显，同软件测试的相关技术以及流程也不断完善，这在很大程度上提升软件研发质量的稳定性。在这个过程中，软件品质的概念在软件研发中渗透，也促进了软件研发专业性水平的提升[9]对于开发系统而言，完成了开发的流程只是开始，接下来要做的就是对所开发的系统进行功能的测试，每一个功能都要进行测试，测试方法一般有黑盒测试和白盒测试，白盒测试是基于对软件内部结构和实现细节的了解来设计测试用例和评估软件的质量。在白盒测试中，测试人员可以查看源代码、设计文档和算法等软件内部细节，从而理解软件的内部工作原理。本次测试采用黑盒测试，测试过程中，通常会将程序看作一个不能打开的黑盒子，在测试人员无法从外面看见产品内部结构的情况下，对软件界面和软件功能进行测试[10]。

## 5.2注册登录功能测试用例

注册登录功能测试包括：用户是否能够成功登录、用户是否能够成功注册。用户登录测试用例如表 5-1 所示。用户注册测试用例如表 5-2 所示。

表5-1 用户登录测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名称 | 用户登录 | 测试序号 | 01 |
| 测试时间 | 2023年04月1日 | 测试人员 | 开发者 |
| 测试目的 | 测试系统登录功能是否正常、稳定 | | |
| 测试步骤 | （1）用户登录本系统并在登录界面输入用户名“admin”，密码“admin”点击登录按钮  （2）用户要登录本系统并在登录界面输入用户名“admin”，密码“adminx”点击登录按钮 | | |
| 预测结果 | （1）提示“登录成功”，并进入主界面，用户admin可以正常登录系统。  （2）提示“密码错误，请重新输入”。 | | |
| 实际结果 | （1）提示“登录成功”，并进入主界面，用户admin可以正常登录系统。  （2）提示“密码错误，请重新输入” | | |

表5-2 用户注册测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名称 | 用户注册 | 测试序号 | 02 |
| 测试时间 | 2023年04月1日 | 测试人员 | 开发者 |
| 测试目的 | 测试系统注册功能是否正常、稳定 | | |
| 测试步骤 | （1）用户在注册界面输入用户名“admin”，账号“admin”，密码“admin”点击注册按钮  （2）用户在注册界面输入用户名“admin1”，账号“admin1”，密码“admin1”点击注册按钮 | | |
| 预测结果 | （1）提示“用户已存在”。  （2）提示“注册成功”，并跳转登录页面。 | | |
| 实际结果 | （1）提示“用户已存在”。  （2）提示“注册成功”，并跳转登录页面。 | | |

## 5.3关注功能测试用例

关注功能测试主包括：是否能成功关注用户、是否能成功取消关注用户。用户关注功能测试用例如表5-3所示。

表5-3 用户关注测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名称 | 用户关注 | 测试序号 | 03 |
| 测试时间 | 2023年04月1日 | 测试人员 | 开发者 |
| 测试目的 | 测试系统关注功能是否正常、稳定 | | |
| 测试步骤 | （1）用户在用户个人空间点击关注按钮。  （2）用户在用户个人空间点击取消关注按钮。 | | |
| 预测结果 | （1）提示“关注成功”。  （2）提示“取消关注成功”。 | | |
| 实际结果 | （1）提示“关注成功”。  （2）提示“取消关注成功”。 | | |

## 5.4实时会话功能测试用例

实时会话功能测试包括：用户的发送功能是否正常、用户的接收功能是否正常、是否存在延迟等。系统实时会话功能测试用例如表5-4所示。

表5-4 实时会话测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名称 | 实时会话 | 测试序号 | 04 |
| 测试时间 | 2023年04月1日 | 测试人员 | 开发者 |
| 测试目的 | 测试系统实时会话功能是否正常、稳定 | | |
| 测试步骤 | （1）用户在会话中心，不输入内容，点击发送按钮。  （2）用户在会话中心，输入内容“你好老板”，点击发送按钮。 | | |
| 预测结果 | （1）提示“发送内容不能为空”。  （2）对方收到用户发送的内容。 | | |
| 实际结果 | （1）提示“发送内容不能为空”。  （2）对方收到用户发送的内容。 | | |

## 5.5用户评论功能测试用例

用户评论功能测试包括：用户是否能够发表评论、用户是否能够删除自己发布的评论。系统用户评论功能测试结果如表5-5所示。

表5-5 用户评论测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名称 | 用户评论 | 测试序号 | 04 |
| 测试时间 | 2023年04月1日 | 测试人员 | 开发者 |
| 测试目的 | 测试系统用户评论功能是否正常、稳定 | | |
| 测试步骤 | （1）用户在苗木详情页面，不输入内容，按下回车按钮发表评论。  （2）用户在苗木详情页面，输入内容“好苗子”，按下回车按钮发表评论。  （3）用户点击删除按钮，删除评论。 | | |
| 预测结果 | （1）提示“你还没有评论！”。  （2）提示“发表成功！”。  （3）提示“删除成功！” | | |
| 实际结果 | （1）提示“你还没有评论！”。  （2）提示“发表成功！”。  （3）提示“删除成功！” | | |

## 5.6苗木地址解析功能测试用例

苗木地址解析功能测试包括：用户能否通过苗木地址在地图中查看具体的地址位置、是否能成功复制苗木地址。系统苗木地址解析功能测试用例如表5-6所示。

表5-6 苗木地址解析测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名称 | 苗木地址解析 | 测试序号 | 04 |
| 测试时间 | 2023年04月1日 | 测试人员 | 开发者 |
| 测试目的 | 测试系统苗木地址解析功能是否正常、稳定 | | |
| 测试步骤 | （1）用户在苗木详情页面，点击苗木地址。 | | |
| 预测结果 | （1）提示“复制成功！”，并跳转地图页面，地图中心点为苗木详细地址。 | | |
| 实际结果 | （1）提示“复制成功！”，并跳转地图页面，地图中心点为苗木详细地址。 | | |

## 5.7苗木交易功能测试用例

苗木交易功能测试包括：用户是否能够购买苗木、用户是否能够取消订单、用户是否能够支付、苗木被购买后是否仍能被购买。系统苗木交易功能测试用例如表5-6所示。

表5-7 苗木交易测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名称 | 苗木交易 | 测试序号 | 04 |
| 测试时间 | 2023年04月1日 | 测试人员 | 开发者 |
| 测试目的 | 测试系统苗木交易功能是否正常、稳定 | | |
| 测试步骤 | 1. 用户在会话中心，点击立即购买按钮并支付。 2. 用户点击立即购买后再次点击立即购买。 3. 用户点击立即购买后，不立即支付，查看订单，并点击取消订单。 4. 用户点击立即购买后，立即支付，查看订单，并点击取消订单。 5. 用户点击立即购买后，立即支付，查看订单，并点击确认收货。 | | |
| 预测结果 | 1. 提示“购买成功！”，并跳转支付宝支付页面。 2. 提示“该苗木已被购买！”。 3. 提示“取消订单成功！”。 4. 提示“取消订单并退款成功！”。 5. 提示“确认收货成功！”。 | | |
| 实际结果 | （1）提示“购买成功！”，并跳转支付宝支付页面。  （2）提示“该苗木已被购买！”。  （3）提示“取消订单成功！”。  （4）提示“取消订单并退款成功！”。  （5）提示“确认收货成功！”。 | | |

## 5.8用户管理功能测试用例

用户管理功能测试包括：管理员是否能够新增用户、管理员是否能够审核注册用户、管理员是否能够修改用户信息、管理员是否能够删除用户。系统用户管理功能测试用例如表5-6所示。

表5-8 用户管理测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名称 | 用户管理 | 测试序号 | 04 |
| 测试时间 | 2023年04月1日 | 测试人员 | 开发者 |
| 测试目的 | 测试系统用户管理功能是否正常、稳定 | | |
| 测试步骤 | 1. 管理员在用户管理页面，点击新增用户，并填写正确的用户信息，点击保存按钮。 2. 管理员在用户管理页面，点击新增用户，并填写的用户信息，但空缺用户账号，点击保存按钮。 3. 管理员在用户管理页面，点击删除按钮，删除用户。 4. 管理员在用户管理页面，点击修改按钮，修改用户的相关信息，点击保存按钮。 5. 管理员在用户管理页面，点击修改按钮，审核注册，点击保存按钮。 | | |
| 预测结果 | （1）提示“保存成功”。  （2）提示“请输入账号”。  （3）提示“删除成功”。  （4）提示“修改成功”，并且相应用户信息修被改。  （5）提示“保存成功”，且相关用户可正常登录。 | | |
| 实际结果 | 1. 提示“保存成功”。 2. 提示“请输入账号”。 3. 提示“删除成功”。 4. 提示“修改成功”，并且相应用户信息修被改。 5. 提示“保存成功”，且相关用户可正常登录。 | | |

## 5.9订单管理功能测试用例

订单管理功能测试包括：管理员是否能够删除已完成的订单、管理员是否能够查询相关苗木类型的订单。系统订单管理功能测试用例如表5-9所示。

表5-9 订单管理测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名称 | 订单管理 | 测试序号 | 04 |
| 测试时间 | 2023年04月1日 | 测试人员 | 开发者 |
| 测试目的 | 测试系统订单管理功能是否正常、稳定 | | |
| 测试步骤 | 1. 管理员在订单管理页面，点击删除按钮，且此订单已完成。 2. 管理员在订单管理页面，点击删除按钮，且此订单未完成。 3. 管理员在订单管理页面，搜索框中搜索”桂花”，且点击搜索按钮。 | | |
| 预测结果 | （1）提示“删除成功”。  （2）提示“订单正在进行中...”。  （3）订单表中出现相关苗木类型的订单。 | | |
| 实际结果 | （1）提示“删除成功”。  （2）提示“订单正在进行中...”。  （3）订单表中出现相关苗木类型的订单。 | | |

# 结束语

本系统基于Vue3和NodeJS的苗木交易平台经过设计与实现，已完成的功能包括注册登录、关注功能、实时会话、用户评论、苗木地址解析、苗木资源推荐、苗木交易、统计分析、用户管理、苗木管理、订单管理。在系统开发过程中，我们遇到了许多问题，并学习了很多新知识，不断提升自己的编程能力。在编写代码的过程中，我们还进行了大量的功能调研，例如如何使用socket.io进行实时通信、使用multer和GridStorage实现图床、使用无限滚动进行分页操作以及如何内嵌腾讯地图并逆向解析等。通过该课题的完成，我们受益匪浅，掌握了大量信息并提升了编程能力，使我们能够更好地利用资源来实现自己的目标。尽管我们已经完成了该平台系统的设计和实现，但我们也认识到它仍存在一些不足之处，例如用户间实时会话无法进行实时通知、目前苗木交易仅支持支付宝支付、苗木的地址仅支持到县级等，需要进一步改善以满足更高水平的要求。

# 参考文献

1. 卞文志.苗木花卉产业亟需由传统向现代转型[J].云南林业,2015,36(05):61.
2. 许会元,何利力.NodeJS的异步非阻塞I/O研究[J].工业控制计算机,2015,28(03):127-129.
3. 王伶俐,张传国.基于NodeJS+Express框架的轻应用定制平台的设计与实现[J].计算机科学,2017,44(S2):596-599.
4. 陈希球.基于Socket.IO框架的服务器与浏览器双向实时通信的实现[J].长江工程职业技术学院学报,2016,33(01):31-32.
5. 肖双林,何迎生,田杰等.基于JWT+Spring Security的动态权限管理系统[J].信息与电脑(理论版),2021,33(14):131-134.
6. 方生.基于“MVVM”模式的“Web”前端的设计与实现[J].电脑知识与技术,2021,17(20):147-149.
7. 刘伟,李树文.MVC模式下的软件开发框架设计[J].电脑知识与技术,2021,17(01):71-72.
8. 张云飞,杨明光.基于Django的RESTful通用程序接口研究与实践[J].电脑知识与技术,2018,14(28):30-32.
9. 何琼月.谈软件工程中软件测试的重要性及方法[J].电子元器件与信息技术,2020,4(11):148-149.
10. 妥泽花.基于黑盒测试与白盒测试的比较探究[J].电子世界,2021,No.617(11):55-56.

# 致谢

毕业论文执笔至此，亦意味着我的大学时光也将划上句号，时光荏苒，感慨顿生。

首先，我要真诚的感谢张燕玲老师在论文的选题、研究方法和论文写作过程中给予了我很多宝贵的指导和建议，使我能够顺利地完成毕业设计，感谢她给予我的热情关怀与细心指导。

其次，我要感谢我的家人、朋友、同学一路以来对我的支持、鼓励与帮助。

最后，感谢滁州学院给了我宝贵的学习机会与条件，感谢各位老师四年来的谆谆教诲和无私奉献。

在这里再次向他们致以最诚挚的感谢，日后的生活和工作中，我会铭记老师学长的教诲，不懈努力，奋勇向前。