

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

[摘要 3](#_Toc29340)

[Abstract 3](#_Toc13280)

[1 绪论 4](#_Toc26302)

[1.1 设计背景及意义 4](#_Toc31382)

[1.2 设计研究目的 4](#_Toc18532)

[1.3 本文主要工作和内容 4](#_Toc22253)

[1.4论文的组织方法 4](#_Toc18343)

[2 相关技术 5](#_Toc21480)

[2.1 NodeJS 5](#_Toc14732)

[2.2 Express 服务端框架 5](#_Toc14427)

[2.3 SocketIO 库 5](#_Toc32747)

[2.4 JOSN Web Token 5](#_Toc9136)

[2.5 Vue3 框架 5](#_Toc13749)

[2.6 MongoDB 数据库 5](#_Toc405)

[2.7 vscode 开发环境 5](#_Toc27227)

[3 需求分析与设计 5](#_Toc5198)

[3.1 业务流程分析 5](#_Toc19797)

[3.2 系统功能分析 7](#_Toc4749)

[3.3 系统用例分析 9](#_Toc29483)

[3.4 系统数据分析 10](#_Toc31986)

[3.4.1实体关系E-R图 10](#_Toc26470)

[3.4.2 数据表字段设计 11](#_Toc12452)

[4 系统实现 14](#_Toc6986)

[4.1系统开发环境 14](#_Toc10150)

[4.2注册登录的实现 14](#_Toc32603)

[4.3关注功能的实现 16](#_Toc7374)

[4.4实时会话的实现 17](#_Toc17903)

[4.5用户评论的实现 19](#_Toc14479)

[4.6苗木地址解析的实现 20](#_Toc32600)

[4.7苗木资源推荐的实现 22](#_Toc21745)

[4.8苗木交易功能的实现 23](#_Toc1141)

[5 系统测试 25](#_Toc6767)

[5.1系统测试概述 25](#_Toc2722)

[5.2系统功能测试 25](#_Toc1167)

[6 结束语 27](#_Toc21608)

[参考文献 28](#_Toc5231)

[致谢 29](#_Toc19541)

基于Vue3+NodeJS的苗木交易平台的设计与实现

摘要： 随着时间的推移，各地对环境条件的要求也在不断提高，对苗木的需求量也在不断增加，但由于苗木资源的零散，买卖关系的不明确，使得传统的苗木行业已经无法适应新的形势。本文采用Vscode开发工具结合NodeJS开发了一个基于Vue3的苗木交易平台。系统采用NodeJS和Vue3进行开发，应用MVC的分层思想设计模式，将软件程序分为3个核心模块，模型(Model)、视图(View)和控制器(Controller)[1]以及RESTful API 的Web 服务架构风格，这种开发模式的关键在于后端开发人员要设计出一个简洁、清晰、权限分明的通用程序接口, 用于与系统前端程序进行数据交互[2]。服务端使用Express框架和MongoDB数据库等。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

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

Abstract: With the passage of time, the requirements for environmental conditions in various places are also increasing, and the demand for seedlings is also increasing. However, due to the scattered resources of seedlings and the unclear relationship between buyers and sellers, the traditional seedling industry has been unable to adapt to the new environment. situation. This article uses the Vscode development tool combined with NodeJS to develop a Vue3-based seedling trading platform. The system uses NodeJS and Vue3 for development, and applies the layered thinking design pattern of MVC to divide the software program into three core modules, model (Model), view (View) and controller (Controller) and RESTful API Web Service architecture style, the key to this development model is that the back-end developers should design a concise, clear, and well-defined general program interface for data interaction with the front-end program of the system. The server uses the Express framework and MongoDB database, etc. The platform consists of two parts: client and server. The client mainly includes functional modules such as browsing seedling resources, seedling trading, real-time conversation, following users, and personal center. The server mainly includes functional modules such as statistical analysis, user management, seedling management, and order management. Through the test of various functional modules of the system, the test results are good and meet the needs of users.

Keywords: Vscode；Vue3；NodeJS；Express；MongoDB

# 1 绪论

## 1.1 设计背景及意义

近年来调查显示，目前大部分苗木产业仍然通过传统的模式运作。传统的苗木交易方式需要大量的人力、物力以及时间成本，且存在着信息不对称的问题，使得交易双方难以获得准确的信息。目前我国苗木相关的平台有：苗木通、苗木网等，但这些平台基本都是面向大规模的苗圃，对于小规模的个人苗木种植商以及苗木买家并没有提供良好的渠道。因此对于需求量并不大的苗木买家，仍是通过雇佣劳动工人耗时耗力的在散落的村户中，进行逐个的找寻农户进行交易，因此必须加快深度调整, 由传统产业向现代产业转型[3]。因此基于Vue3 + NodeJS的苗木交易平台的设计和实现具有重要的意义，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，具有广阔的应用前景。

## 1.2 设计研究目的

本设计是基于Vue3 + NodeJS的苗木交易平台的设计和实现，苗木交易平台将分散在各个村落中的苗木资源进行聚合，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，更快更准确的满足苗木买卖双方的需求。并且通过平台统一的交易，可以提高散户苗木交易的确定性以及安全性，避免了传统运作模式所带来的恶性竞争与交易不确定性。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以及更高效的读写。满足市场的需求，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

本文主要研究如何实现一个苗木交易平台，本设计主要运用NodeJS语言进行开发，MongoDB作为后台数据库存储数据。首先对系统需求以及可行性进行分析，确定系统的特点以及功能，然后根据功能数据库设计以及进行系统开发。系统实现中需要确定系统开发环境以及核心功能用户注册登陆、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。展示功能实现的核心代码。然后系统实现后，对其进行系统测试，测试系统运行是否稳定、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论文的组织方法

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、主要功能模块的分析概述、系统数据库设计；

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ChromeV8为基础的 JavaScript运行环境，使得 JavaScript不再受浏览器的局限，具有异步非阻塞I / O、单线程、事件机制、跨平台等特点[4]。随着近些年nodejs及其生态的发展，nodejs也逐渐成为后端开发的选择之一。

## 2.2 Express 服务端框架

Express是nodejs的生态之一，作为nodejs老牌的服务端框架，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，能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，同时缩短应用发布周期[5]。

## 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[6]。

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[7]，是目前用于用户鉴权较为流行的方式之一。

2.5 Vue3 框架

Vue3是一个流行的前端框架，采用MVVM[8]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 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 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3-1所示。

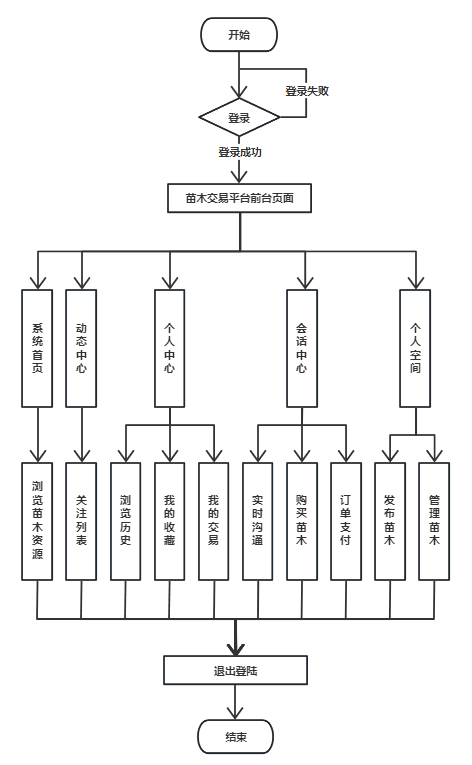


图3-1 系统前台功能流程图

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。

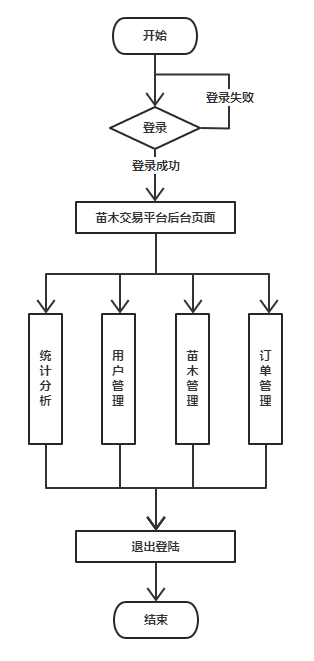


图3-2 系统后台功能流程图

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。

## 3.2 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：
   1. 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。
   2. 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。
   3. 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。
   4. 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。
   5. 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。
   6. 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。

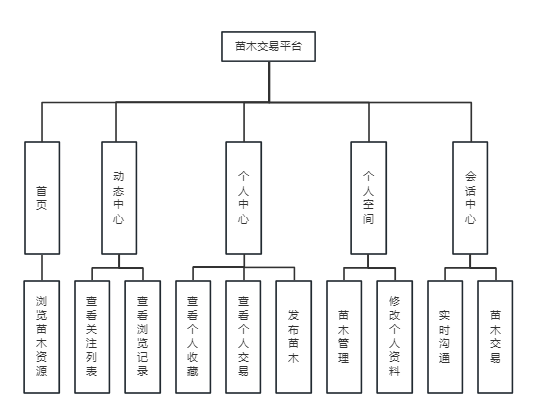


图3-3 前台功能模块图

1. 后台部分功能模块：
   1. 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。
   2. 用户管理模块：管理员可审核注册用户、管理用户信息。
   3. 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。
   4. 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。

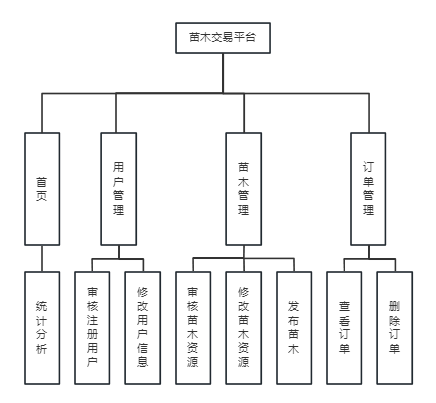


图3-4 后台功能模块图

3.3 系统用例分析

本系统主要包括普通用户和管理员用户两种用户角色，他们各有不同的用户需求，普通用户需要登录系统、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；而管理员用户需要能够对用户信息、苗木信息进行增删改查以及审核操作、查看订单信息以及查看统计分析的图表。

（1）普通用户需求用例如图3-5所示。

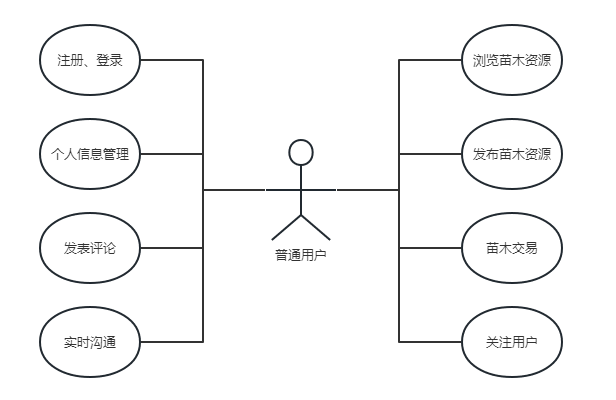


图3-5 普通用户需求用例图

（2）管理员需求用例如图3-6所示。

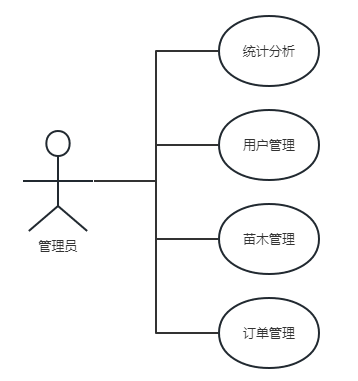


图3-6 管理员需求用例图

## 3.4 系统数据库设计

3.4.1实体关系E-R图

ER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[9]。本系统所有数据库表ER图如图3-7所示。

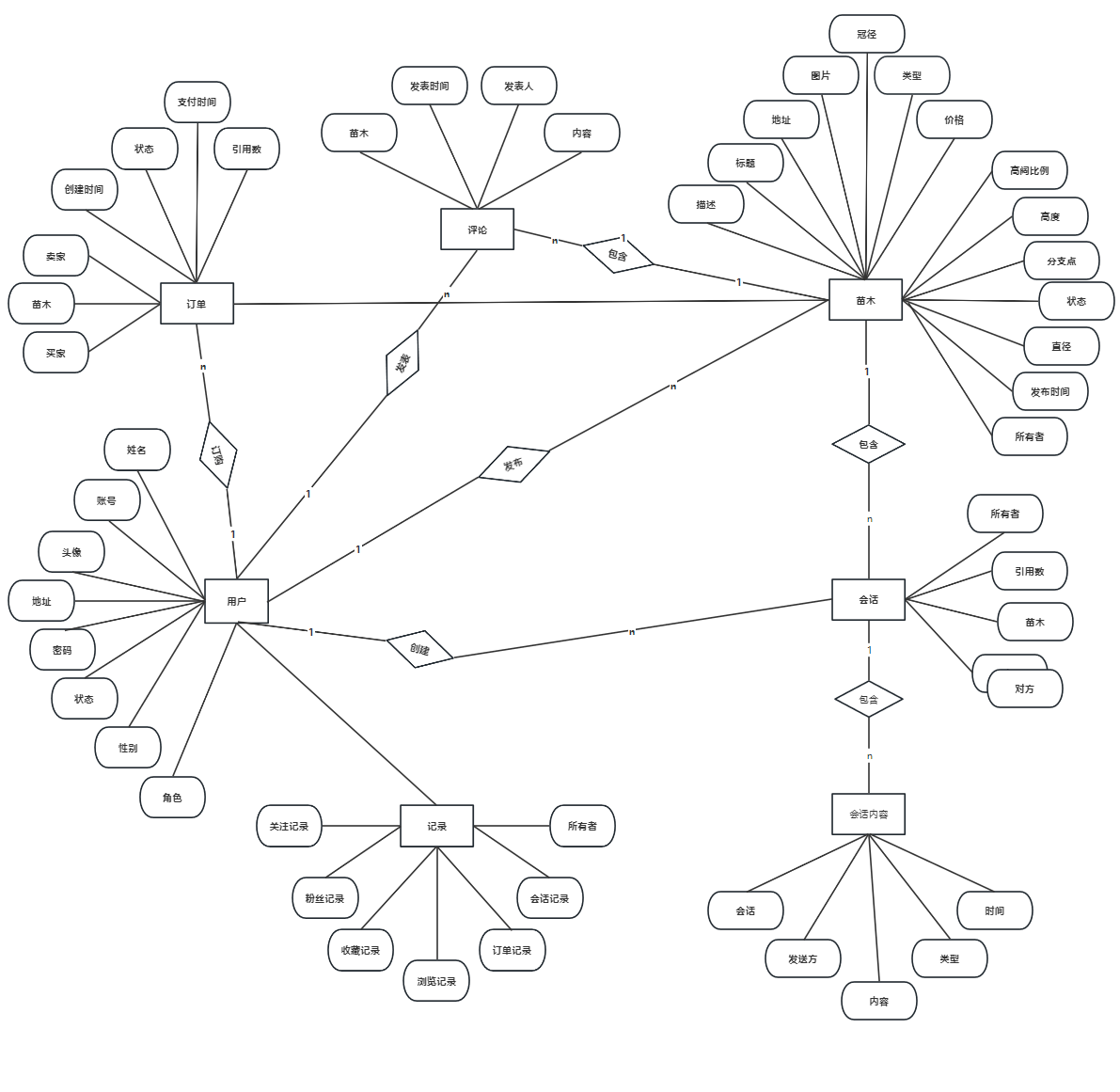


图3-7 用户信息实体图

### 3.4.2 数据表字段设计

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。具体展示部分如下所示。

表3-1 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3-2 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3-3 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3-4 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 |
| following | Array | 否 | 关注记录 |
| fans | Array | 否 | 粉丝记录 |
| collect | Array | 否 | 收藏记录 |
| browsingHistory | Array | 否 | 浏览记录 |
| order | Array | 否 | 订单记录 |

表3-5 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3-6 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3-7 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 |

# 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。

## 4.2注册登录的实现

在此实现用户登录注册功能，用户在进入系统主页前必须进行登录操作，若用户没有该系统账号，则需要进入注册页面进行注册。在进行登录操作时，用户需要输入正确的账户以及密码才能进入系统进行其他操作。在进行注册操作时，用户需要输入正确的用户名、账号、密码并选择用户类型（种植苗木用户或苗木买家），且账户不可重复，若账户已存在将注册失败，注册成功后用户进入待审核状态，等待管理员审核。注册页面如图4-1所示。登录页面如图4-2所示。

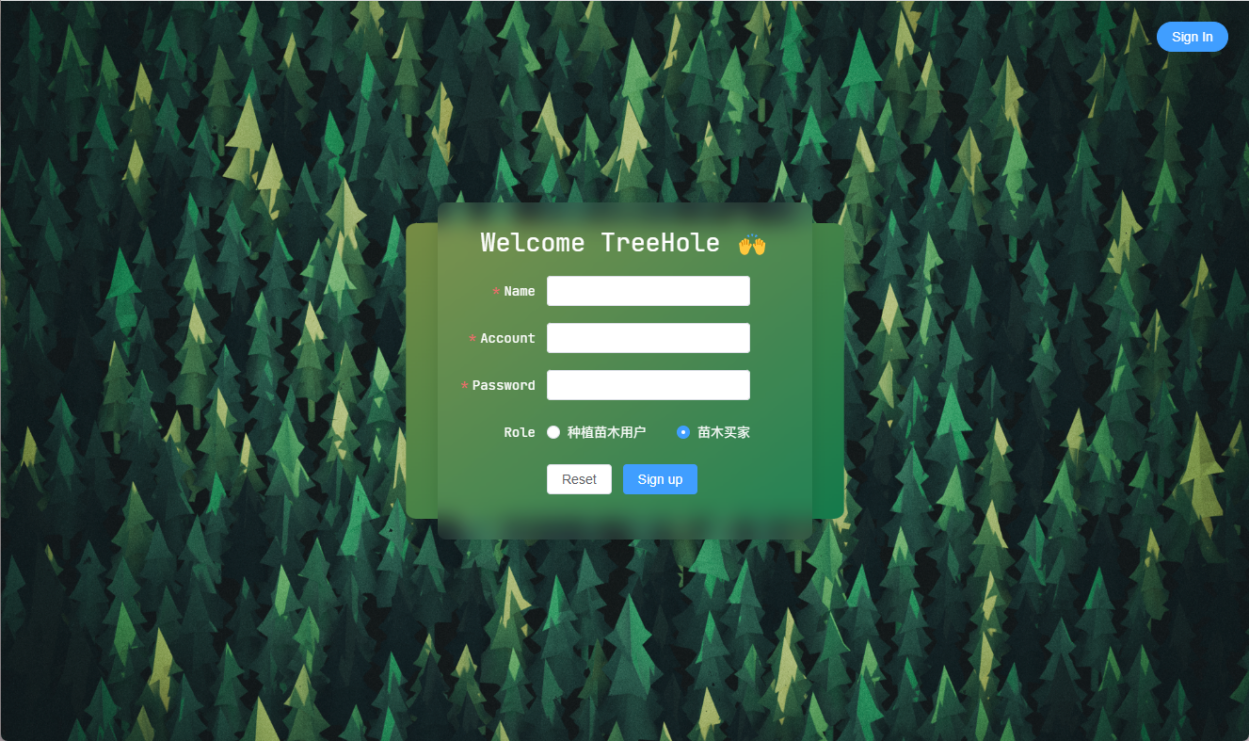


图4-1 用户注册页面

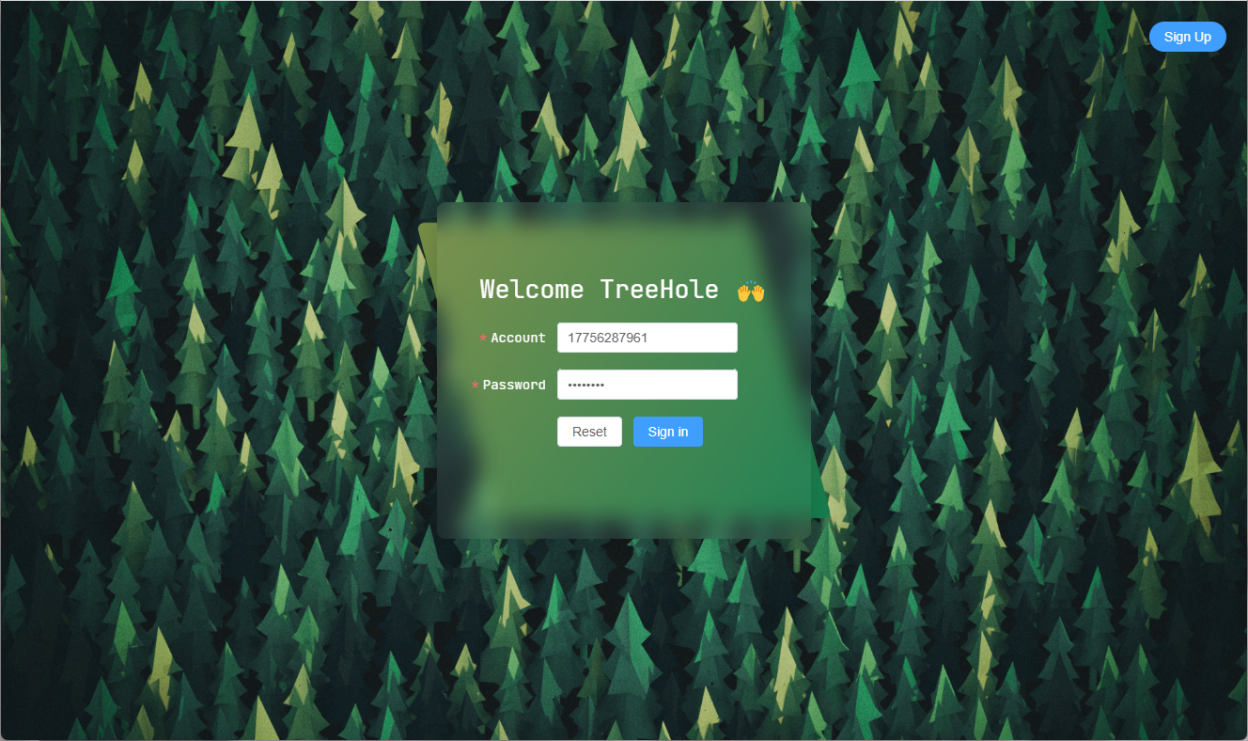


图4-2 用户登录页面

（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();

## 4.3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。主要实现方式，是通过数组的方式将关注用户的id进行存储，并且关注或取消关注涉及到用户双方的关注列表以及粉丝列表，故需要操作两个用户的记录数据，并且由于mongoDB查询顺序的特殊性，需对双方的记录进行一些顺序上的处理。关注与取消关注效果如图4-3所示。

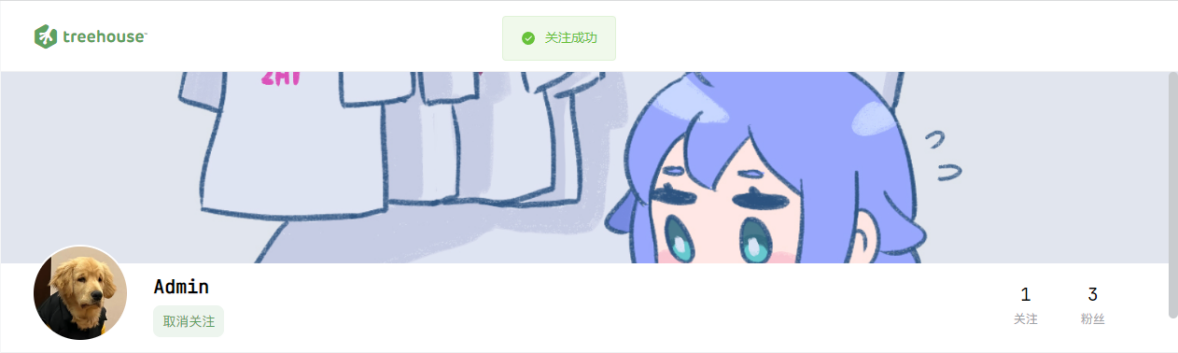


图4-3 关注用户功能

其核心代码如下：

const { userID1, userID2 } = req.body;

const records = await Record.find({

userID: { $in: [userID1, userID2] },

});

if (records.length != 2) {

next(err("记录有误", 403, ""));

return;

}

if (records[0].userID != userID1) records.reverse();

const index\_0 = records[0].following.indexOf(userID2);

const index\_1 = records[1].fans.indexOf(userID1);

if (index\_0 == -1) {

records[0].following.unshift(userID2);

records[1].fans.unshift(userID1);

} else {

records[0].following.splice(index\_0, 1);

records[1].fans.splice(index\_1, 1);

}

let res1 = await Record.findByIdAndUpdate(records[0].\_id, records[0]);

let res2 = await Record.findByIdAndUpdate(records[1].\_id, records[1]);

const data = [res1, res2];

if (data == []) {

next(err("Data update failed!", 403, ""));

return;

}

## 4.4实时会话的实现

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。实现方式主要通过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4-4所示。

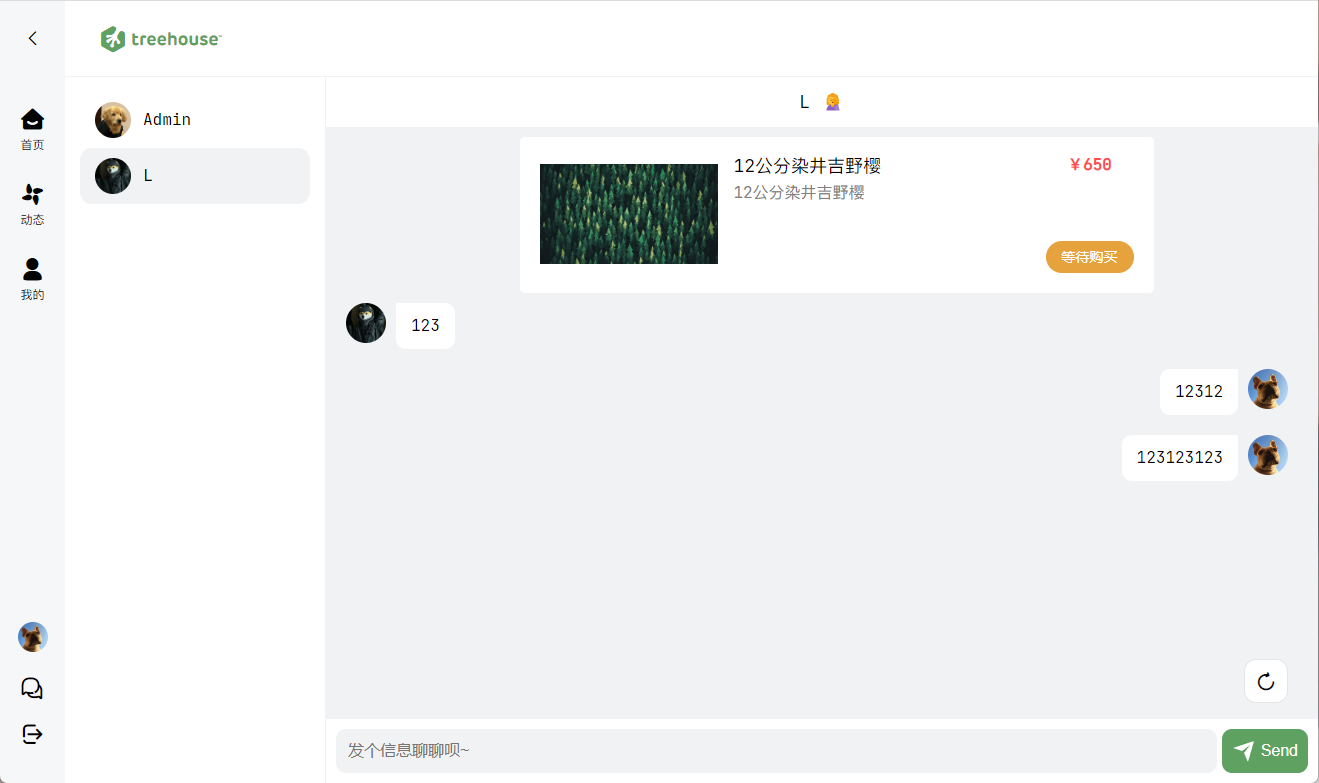


图4-4 会话页面

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. 数据存储

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);

}

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();

## 4.5用户评论的实现

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。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4-5所示。

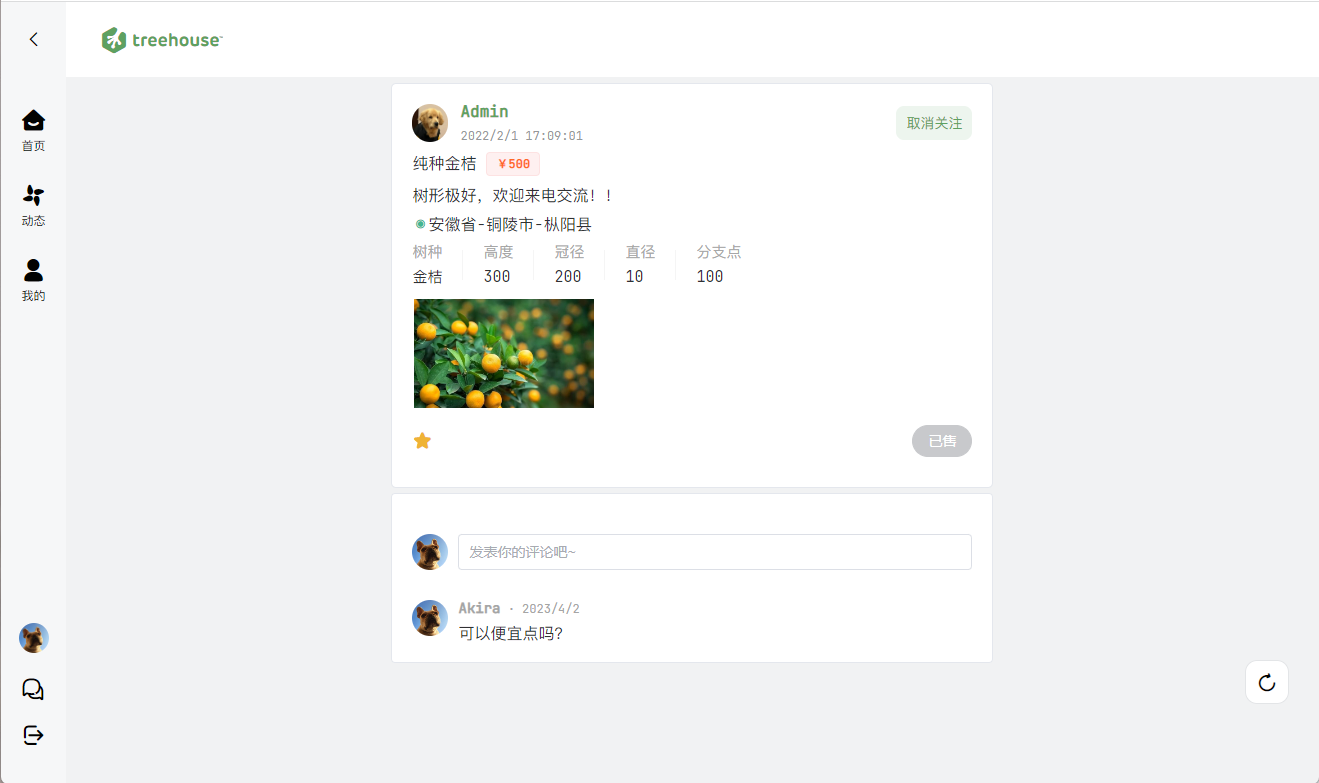


图4-5 苗木详情页面

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();

## 4.6苗木地址解析的实现

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。实现方式主要通过集成腾讯地图API，建立正逆地址解析类，对中文地址进行反向解析，重新渲染地图以及中心点位置，从而实现地图中查看苗木的地址。苗木地址页面如图4-6所示。



图4-6 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);

});

}

## 4.7苗木资源推荐的实现

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。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，再进行数据筛选展示。苗木资源推荐页面如图4-7所示。

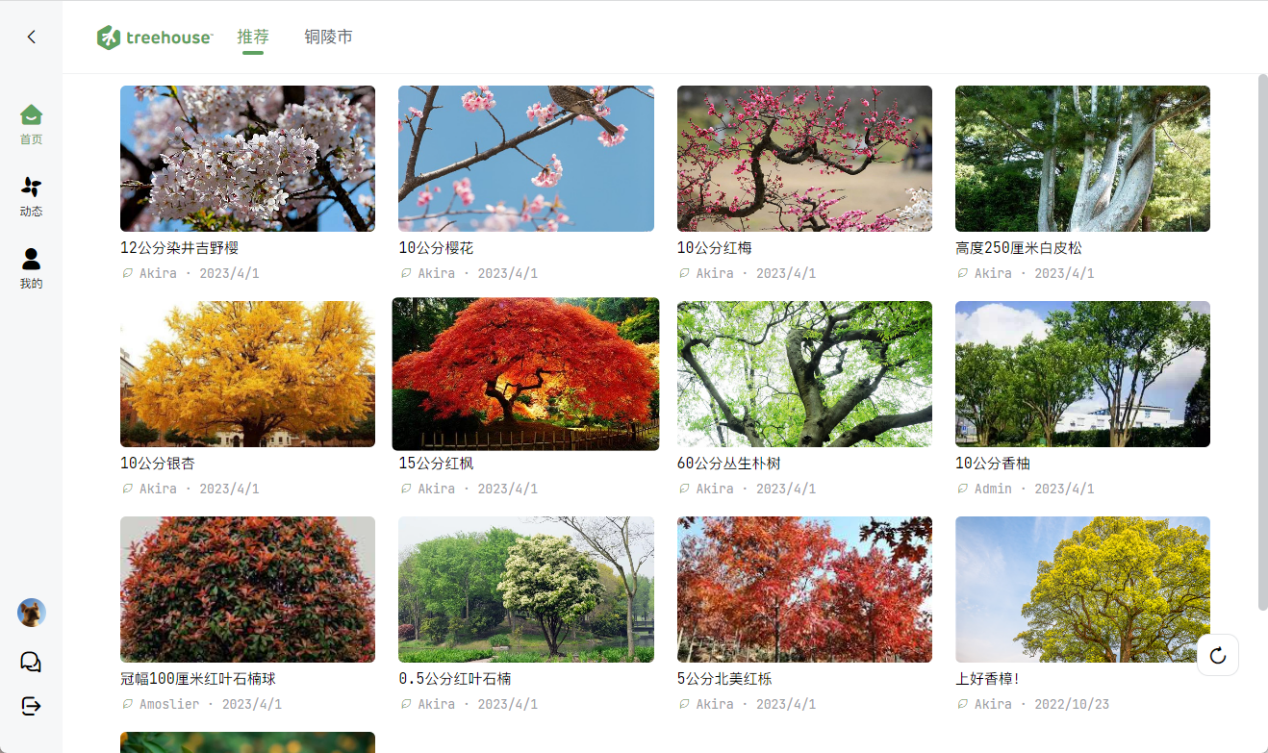


图4-7 苗木资源推荐页面

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);

## 4.8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，并且该苗木资源将不会出现在平台中，仅交易双方可见，进而避免传统苗木运行模式下出现的恶性竞争，主要实现方式是通过支付宝沙盒进行模拟支付，以及对苗木的状态的调控。苗木交易订单详情页面如图4-8所示。



图4-8 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"));

}

# 5 系统测试

5.1系统测试概述

对系统进行测试，是为了找出系统中的缺陷，并对其进行改进。对于一个完整的系统来说，系统开发的完成只是第一步，接下来还需要对系统的各个功能进行测试，这关系到用户的使用体验。系统测试目的就是要测试这个系统是否已经完成了系统设计中的全部功能，以及它是否能够正常工作。

## 5.2系统功能测试

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，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]。系统登录功能测试结果如表 5-1 所示。核心功能测试如表 5-2 所

示。

表5-1 登录功能测试表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 具体步骤 | 输入内容 | 预期效果 | 实际效果 | 说明 |
| 1 | 输入账号 | 账号：admin,  密码为空 | 出现“请输入密码！” | 出现“请输入密码！” | 结果准确 |
| 2 | 输入密码 | 密码：admin，账号为空 | 出现“请输入账号！” | 出现“请输入账号！” | 结果准确 |
| 3 | 输入账号和密码但不正确 | 账号：admin，  密码：xxxxx | 提示“密码错误，请重新输入！” | 提示“密码错误，请重新输入！” | 结果准确 |
| 4 | 输入密码和账号但不正确 | 账号：adminxx,  密码：admin | 提示“用户不存在，请先注册！” | 提示“用户不存在，请先注册！” | 结果准确 |
| 5 | 输入准确密码和账号 | 账号：admin,  密码：admin | 进入系统 | 进入系统 | 结果准确 |

表5-1 核心功能测试表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 检测功能 | 具体操作 | 预期结论 | 说明 |
| 1 | 登录 | 使用账号密码与注册时一致 | 进入系统 | 结果准确 |
| 2 | 注册 | 输入准确且未被注册的账号以及密码 | 返回登录 | 结果准确 |
| 3 | 关注用户 | 点击关注按钮 | 提示“关注成功” | 结果准确 |
| 4 | 取消关注用户 | 点击取消关注按钮 | 提示“取消关注成功” | 结果准确 |
| 5 | 收藏苗木资源 | 点击收藏按钮 | 提示“收藏成功” | 结果准确 |
| 6 | 取消收藏苗木 | 点击取消收藏按钮 | 提示“取消收藏成功” | 结果准确 |
| 7 | 实时通讯 | 点击联系卖家，对话框中输入信息，点击发送 | 卖家对话框实时接收到对方发送的数据 | 结果准确 |
| 8 | 发表评论 | 按回车发表评论 | 发表成功 | 结果准确 |
| 9 | 地址解析 | 点击苗木地址 | 进入地图页面，并中心点位为苗木地址 | 结果准确 |
| 10 | 购买苗木 | 点击立即购买按钮 | 跳转支付宝支付页面 | 结果准确 |
| 11 | 取消订单 | 点击取消订单按钮 | 若已支付将提示“取消订单并退款成功！”，若未支付提示“取消订单成功！” | 结果准确 |
| 12 | 用户浏览记录 | 用户点击查看苗木详情 | 个人中心中新增对应的苗木浏览记录 | 结果准确 |
| 13 | 发布苗木 | 点击发布按钮，填写相关苗木信息，点击保存 | 提示“发布成功” | 结果准确 |
| 14 | 统计分析 | - | 周交易通过折线图展示，苗木种类通过饼状图展示 | 结果准确 |
| 15 | 用户管理 | 用户信息增删改查 | 用户信息发生相应变化 | 结果准确 |
| 16 | 苗木管理 | 苗木信息增删改查 | 苗木信息发生相应变化 | 结果准确 |
| 17 | 订单管理 | 用户信息删查 | 订单信息发生相应变化 | 结果准确 |

# 6 结束语

本系统的设计基于 Vue3 + NodeJS的苗木交易平台的设计与实现已经全部完成，在开发系统的过程中遇到了不少的问题，同时也学习了很多的新知识，不断弥补自己的欠缺，在编写代码的过程中也做了很多功能的调研，比如如何使用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、如何内嵌腾讯地图并逆向解析等。本次课题的完成让我受益匪浅，不仅让我掌握了大量的信息，而且还提升了我的编程能力，让我可以更好地利用资源来实现自己的目标。然而，本平台系统设计还存在一些不足之处，需要进一步改善，以满足更高水平的要求。

# 参考文献

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 李志.论E-R图在数据库建模过程中的重要性[J].信息记录材料,2020,21(06):143-145.
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。