### 

### 成都职业技术学院

### 毕业设计（项目技术报告）

|  |  |
| --- | --- |
| 题目 | 赛博集市 |
| 学生姓名 | 吴臻海 |
| 专业班级 | 软件203-2班 |
| 学生学号 | 20302030230 |
| 院（部） | 软件分院 |
| 指导教师 | 邓达 |

2023年4 月 22 日

目录

[1 引言 4](#_Toc133416619)

[1.1 编写目的 4](#_Toc133416620)

[1.1.1 项目背景 4](#_Toc133416621)

[1.1.2 专门定义 5](#_Toc133416622)

[1.1.3 参考资料 5](#_Toc133416623)

[2项目概述 7](#_Toc133416624)

[2.1 设计目标 7](#_Toc133416625)

[2.1.1 运行环境 7](#_Toc133416626)

[2.1.2 需求概述 7](#_Toc133416627)

[2.1.3 条件和限制 8](#_Toc133416628)

[2.2 总体设计 8](#_Toc133416629)

[2.2.1 处理流程 8](#_Toc133416630)

[2.2.2 总体结构和模块外部设计 10](#_Toc133416631)

[2.2.3 功能分配表明各项功能与程序结构的关系 12](#_Toc133416632)

[2.3 接口设计 12](#_Toc133416633)

[2.3.1 管理端路由 12](#_Toc133416634)

[2.3.2 内部接口（部分） 14](#_Toc133416635)

[2.4 数据库结构设计 18](#_Toc133416636)

[2.4.1 数据库表字段设计（Mongoose Schema） 18](#_Toc133416637)

[3数据结构和处理流向 21](#_Toc133416638)

[3.1运行设计 21](#_Toc133416639)

[3.1.1 运行模块的组合 21](#_Toc133416640)

[3.1.2 客户端页面展示（部分） 23](#_Toc133416641)

[3.1.2 管理端页面展示 27](#_Toc133416642)

[3.2 出错处理设计 30](#_Toc133416643)

[3.2.1 出错输出信息 30](#_Toc133416644)

[3.2.2 出错处理对策 30](#_Toc133416645)

[**4 项目总结** 31](#_Toc133416646)

[4.1 感想 31](#_Toc133416647)

[4.2总结 31](#_Toc133416648)

摘 要

赛博集市交易平台主要发挥二手交易和相关经验之分享，客户端通过向后端添加数据，得以在前台进行显示，所以就要结合前台的业务流程以及前台的展示进行测试。最终目的就是数据可以正确的在前台展示出来。  
 该平台最大的作用就是用户自主交易，相较于传统的交易平台，该平台给予用户较大的自由，可以自主发布商品、与买家协商交易金额、交易方式，所有的活动都会留存于服务器，方便维权，而管理员则在后台监控用户所提交的数据，并对部分违规违法的、异常的数据进行封锁、清理。

该平台的后台管理系统的模式相对简单，基本都会有一个功能菜单，然后每个菜单对应着一个功能模块，这个模式很固定。不同于传统的电商管理系统，本平台的后台没有提供增改功能，所以在后端管理系统里的功能会相对简单，即使复杂也是服务器层面的，不需要大幅修改管理端，而逻辑上一般不会特别麻烦。

关键字：（Node.js、MongoDB、Vue、React）

# 1 引言

## 1.1 编写目的

### 项目背景

目前，随着商品经济之发展，出现了闲置之物品，这些闲置品日积月累，增加了维护成本，极大造成了资源浪费，一些二手交易平台应运而生。但基于当前电商平台的惯性，传统的二手交易仍然需要货币交易，在面临货币严重贬值等特定环境时无法有效进行。本平台除了提供货币交易，还可以通过买卖双方协商确定其他等价交换物（如以物易物）来推进交易，同时留存交易凭证以维护双方权益。

### 专门定义

#### （1）用户自主权

用户自主权主要体现在用户可自主发布商品、帖子、评论等数据，并在处理交易时拥有更多样化的手段，以实现复杂的、经济危机条件下的交易。

#### （2）社区化

在商品、帖子下，用户可以围绕某个特定的对象、特定的时间，以评论之方式参与讨论，而商品发布（发帖）者亦可以参与其中，在交易中可以形成买卖双方特定的互信关系以及默契，也可以将一些经验拓展开来，对其有更深刻的认识。同时，为维护社区良好和谐之环境，客户端亦有举报通道，用户可以对违反规定的数据提交举报材料，等待管理员审核结果。

#### （3）管理员监督

管理员在该平台中扮演着平台秩序的主要维持者和服务器的维护者，并不具备发布、修改数据之权限。此系统主要负责收集、存储和分析市场、销售、服务及整个企业的各类信息，对客户进行全方位的了解，为企业市场决策提供依据，从而理顺企业资源与客户需求之间的关系，提高客户满意度，实现挖掘新客户、支持交叉销售、保持和挽留老客户、发现重点客户、支持面向特定客户的个性化服务等目标。

### 参考资料

在本文档中，引用了如下的文档、资料及相关技术栈网站

-- 《软件文档编写规范 - GB2020》

-- Vue：<https://cn.vuejs.org/>

-- React：<https://zh-hans.react.dev/>

-- Tauri: https://tauri.app/zh-cn/

-- Ant Design：https://ant.design/components/overview-cn

-- ECharts: https://echarts.apache.org/zh/index.html

-- Uni-App: https://uniapp.dcloud.net.cn/

-- Node.js: https://nodejs.org/zh-cn

-- MongoDB: https://www.mongodb.com/

-- Mongoose ODM: https://mongoosejs.com/

-- Express.js: <http://expressjs.com/zh-cn/>

-- Socket.IO: <https://socket.io/zh-CN/>

-- Redis: https://redis.io/

# 2项目概述

## 2.1 设计目标

### 2.1.1 运行环境

系统：Windows10+、Linux系统、MacOS（该平台在Windows11下开发）

运行时：Node.js

数据库：MongoDB数据库

缓存：Redis缓存（Linux版）

开发环境：HBuilderX（客户端）、JetBrains WebStorm（管理端和服务器）

注：由于某些原因，开发机无法运行Docker虚拟化容器.

### 2.1.2 需求概述

**该平台的需求只要包括以下的总结性要求。本文档的设计目的就是要通过编制软件来达到这些要求**

1. 管理员需要验证身份进入系统，对本平台的商品、帖子、用户等信息进行管理；
2. 数据量过大，可以使用关键字和过滤选择来进行筛选，进行分页展示；
3. 数据简洁化，没有繁琐的数据，对数据进行整理展示，更便于理解；
4. 前后端分离，前端页面可以更改，后端使用一些中间件；
5. 部分数据利用图表进行展示，很清晰看到数据的变化；
6. 管理端部分需要实时令牌操作，提高了安全性；
7. 每个部分都有相应的接口，缺少相应的参数不能继续；
8. 每条数据都可以获取到相关的数据信息，方便查看（客户端的大部分以及管理端的举报）

### 2.1.3 条件和限制

1.运行限制

因为是前后端分离项目，所以必须要两个端口同时运行时该项目才能正常运转。当后端未运行时，一般不影响运行，但部分页面无法正常渲染，故需要注意。此外，由于使用了Redis，需要安装与之相适应的环境（Linux子系统或者Docker均可），否则管理端无法使用。

1. 使用限制

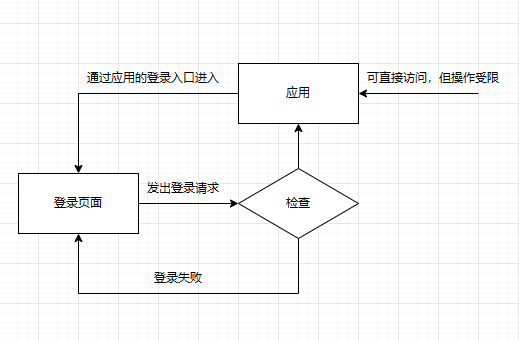
项目涉及的技术栈相对较多，启动流程上相对复杂，使用上需要注意，如有报错则请及时保存报错信息，虽然操作逻辑上借鉴了市面主流的二手交易平台，上手容易，但在交易方式上需要多加留意，以免出现不必要的损失。

管理端部分，如果强制以浏览器方式访问，则会因为缺少Tauri模块而无法正常运行。

## 2.2 总体设计

### 2.2.1 处理流程

#### 1、登录判断



客户端方面，在未登录状态下可以访问部分页面，但大多数操作受限，通过应用内部的登录入口发送登录请求并且获得登录成功的响应数据后，才能执行受限的操作。客户端一旦登录，将会存储用户的登录信息，除非手动登出，用户不需要再次执行相同的登录操作（上图为客户端登录流程图）。

管理端方面，由于权限设置，只有登录成功才能访问主页面，直接访问主页面将导致没有有效的令牌，进而无法获取主页面信息，此时切换页面则会跳转至登录页面（仅在Tauri环境下正常判断）。

#### 客户端操作

1. 查询商品信息，可通过搜索、首页加载
2. 商品详情获取，同步获取评论
3. 查询帖子信息，可通过搜索、发现页加载
4. 帖子详情获取，同步获取评论
5. 在商品、帖子下发表评论
6. 发布或修改商品、帖子
7. 收藏商品、帖子
8. 对违规的商品、帖子或评论进行举报
9. 在特定商品页面下进行交易
10. 查看消息及不同类别的消息详情
11. 修改个人信息和密码

#### 管理端操作

1. 在首页查看近段时间所发布的数据，处理近段时间的举报，实时监控服务器状态。
2. 查看用户列表，通过用户名确定查找条件，支持分页，可进行锁定、封禁、删除等操作。
3. 查看不同类型的数据列表，通过时间段确定查找条件，支持分页，可进行锁定、删除等操作。
4. 查看举报列表，可根据时间段、其他原因确定查找条件，支持分页，管理员对举报信息进行判断后，可处理也可以否决。

### 2.2.2 总体结构和模块外部设计

#### 1、Node.js

Node.js是一个跨平台的、事件驱动I/O服务端的JavaScript运行时，其封装了谷歌V8引擎，拥有较高的性能，但添加了网络、I/O相关的功能，使得其具备搭建服务端的能力。同时，Node.js也是前端项目的主要开发环境和运行时，主流的前端框架都在此拥有相应的脚手架工具。Node.js对于前端工程师而言，由于使用JavaScript，大大降低了学习成本，而且很多前端搭建也依赖Node.js，为快速成长为全栈开发者提供了便捷之环境。

#### Express.js

Express.js是一个基于Node.js的轻量级Web框架，其封装了，使得开发人员能够关注业务逻辑，提高了开发效率。同时，Express.js开发团队也提供了更加轻量的Web框架Koa.js，其按需导入的特性、与Express.js相似的开发流程，使得其学习成本大大降低，同时最小化体积。

#### MongoDB与Mongoose

MongoDB是一种基于分布式文件存储的非关系型数据库，旨在为WEB应用提供可扩展的高性能数据存储解决方案。与SQL数据库不同，由于采用了与json格式相似的bson格式，MongoDB可以存储比较复杂的数据类型，并且存储的模式更加自由。MongoDB的查询语言非常强大，支持聚合管道查询及对数据添加索引，查询效率高。

Mongoose是运行于Node.js的MongoDB ODM库，功能上与Mybatis相似，与MongoDB官方提供的驱动程序对比，Mongoose提供了Schema和Model，为增删改查等操作进行了数据规范，以保证数据的一致性。同时，Mongoose的操作优雅，使用ES6+特性可以最大程度避免回调地狱，提升代码的可读性。

#### Redis

Redis是一种缓存数据库，以键值对的形式存储于内存中。由于官方仅提供Linux版本，需要安装相应的Linux运行环境，社区有windows版本，但使用Linux版本能够获得最多的技术支持。

#### Socket.IO

WebSocket的升级封装，可用于实时监听。不同于WebSocket，Socket.IO还提供了HTTP长轮询方式，以兼容较老的浏览器，而且在事件处理上可以自定义，拓展了可用性。目前，使用Socket.IO需要前后端同时安装相应的版本，以保证WebSocket通信正常进行。

#### Vue

前端框架，采用了虚拟DOM技术，不用频繁操作DOM树。特有的响应式编程和组件化，双向的数据绑定，完美的数据和结构分离，极高的提升了运行速度。

Vue是单页面应用，通过VueRouter及ajax请求，使页面局部刷新，不用每次跳转页面都要请求所有数据和DOM，这样大大加快了访问速度和提升用户体验。而且其第三方UI库，如Element UI，能够节省很多开发时间。同时，Vue有相应的全栈框架，比如Nuxt.js，能够快速搭建全栈项目而不用频繁地向传统后端工程师请求技术支持。

#### React

前端框架，同Vue一样，采用了虚拟DOM技术，不需要频繁操作DOM。React侧重于MVC中的V（即View）。由于其采用单向数据绑定，需要对绑定数据的表单组件进行一些处理。其最新版本的函数组件采用hook钩子函数，取代了较为庞大的类组件。相对于Vue，React更适合构建大型Web项目。

目前，React是国外主流的前端框架，国内主要在互联网巨头中应用（如知乎、抖音），由于React学习难度较大，加之前端培训机构以Vue框架为主，React人才相对匮乏。

#### Ant Design

Ant Design是前端UI库，由阿里出品和维护，搭配React使用更加方便的展示前端优美的页面，而且内置很多API，可以提高和用户的交互需求，是国内React项目主流的UI库之一。

#### Tauri

与Electron类似，Tauri同样使用前端技术构建桌面应用程序，但不同于Electron，其后台使用Rust语言，相较于Electron后台使用的Node.js，性能、安全性都有大幅度的提高，同时使用系统自带的WebView库，使得其安装包和安装后的大小大幅度减小（可通过配置文件设置是否在安装程序内嵌入WebView），Tauri未来会将移动端之构建引入其生产环境，大幅减小开发人员跨平台构建之压力。

#### ECharts

ECharts是前端图表库，由百度开发，现由Apache基金会维护。ECharts提供了强大的绘图API，能够满足不同数据可视化的需求，并且使用原生js，理论上兼容目前以及未来的前端框架（传统如JQuery，主流如Vue、React）\*。

\*部分前端框架需要相应的封装，如React，与ECharts相对应的EChartsForReact库。

### 2.2.3 功能分配表明各项功能与程序结构的关系

#### 1、后台

后台不在对页面进行操作，也不在进行页面的跳转和重定向，后台只需要复制的是数据的传递，前端访问不同的接口，后台操作数据库对数据进行格式化，然后通过RESTful风格封装数据，并发送给前端，后台做到的是权限的校验和对数据的验证，以及反馈前端的请求。最主要的是后台对各个用户的权限进行规定和限制，后台才是最安全的权限检验。

#### 前端

前端需要进行数据的展示和获取，并且处理页面的跳转和访问请求，采用的是局部刷新，所以获取数据时是靠请求来获取，当获取到数据之后，更改界面或者提示用户，不再繁琐地进行刷新和获取，根据不用的请求接口来获取不同数据，利用前端来进行展示来和用户进行交互，提升使用感受。

## 2.3 接口设计

### 2.3.1 管理端路由

#### 1.登录主页面

请求路径：/login

请求参数：无

成功返回：登录页面，如果拥有有效的令牌，则跳转至首页

#### 2.获取各个操作页面

请求路径：/xxx（login除外）

请求参数：无

成功返回：如果拥有有效的令牌，则跳转至对应各个操作的页面，否则跳转至登录页面

#### 3.404页面

请求不存在的路径

此时跳转404页面

### 2.3.2 内部接口（部分）

#### 1.统一请求返回格式

返回格式：

{

code: <Number>,

msg: <String>,

data: <Any>

}

#### 2.客户端登录

请求接口：/api/user/login

请求参数：

{

login\_form: {

username: <String>,

password: <String>

}

}

成功返回：

{

code: <Number>,

msg: <String>,

data: <User>

}

#### 3.获取商品信息

请求接口：/api/goods/goods\_info

请求参数：

{

goods\_id: <String>

}

成功返回：

{

code: <Number>,

msg: <String>,

data: {

goods\_id: <String>,

owner: <String>,

goods\_name: <String>,

desc: <String>,

price: <Number>,

status: <String>,

imgs: <Array<Imgs>>,

comments: <Array<Comment>>

}

}

#### 4.加载商品列表

请求接口：/api/goods/goods\_display

请求参数：

{

filter: {

start\_at: <Number>,

amount: <Number>

}

}

成功返回：

{

code: <Number>,

msg: <String>,

data: {

data: <Array<Goods>>,

next\_index: <Number>

}

}

#### 5.其他请求接口格式规范（客户端）

请求接口：/对应类型/对应操作

请求参数：

{

verify: <userid>(通过客户端拦截器添加),

……(其他参数)

}

成功返回：

{

code: <Number>,

msg: <String>,

(data: <Any>)

}

客户端请求时会默认带上verify参数，当服务器接收到参数时，首先检查所请求的路径，并通过封禁白名单、未登录白名单确认是否需要检查用户，如果未在白名单中且未通过封禁、未登录检查，客户端将被拒绝访问，否则接收服务器的响应数据。

后台返回的数据格式是规定的Result格式化，有code编码，data数据（部分接口不返回），msg消息通知，客户端接收的一般都是data数据里面的存放的不同的数据，后台将数据编写为后端JSON格式返回，客户端可直接接收。

#### 6.其他请求接口格式规范（管理端）

请求接口：/对应类型/对应操作

请求参数：

{

token: <token\_id>,

……(其他参数)

}

成功返回：

{

code: <Number>,

msg: <String>,

(data: <Any>)

}

管理端请求时会默认带上token参数，当服务器接收到参数时，首先检查Redis缓存，如果不存在则转至MongoDB数据库检查，若存在结果且未过期，则写入Redis后再允许管理端进行请求，否则解决。在此过程中，如果Redis缓存和MongoDB都没有有效的令牌，则管理端的请求无法继续，以阻止在无授权的情况下随意操作数据之可能性。

管理端接收的数据格式与客户端较为相似，都经过相同的处理流程后渲染之，此处不再展开。

## 2.4 数据库结构设计

### 2.4.1 数据库表字段设计（Mongoose Schema）

#### 1.商品表goods

{  
 goods\_id : { type: String, unique: true },  
 owner: { type: String },  
 goods\_name: { type: String },  
 desc: { type: String },  
 price: { type: Number },  
 status: { type: String },  
 imgs: { type: Array },  
 post\_date: { type: Date },  
 isDel: { type: Boolean }  
}

#### 2.帖子表 post

{  
 post\_id: { type: String, unique: true },  
 post\_by: { type: String },  
 title: { type: String },  
 content: { type: String },  
 imgs: { type: Array },  
 post\_date: { type: Date },  
 isDel: { type: Boolean }  
}

#### 3.评论表 comment

{  
 comment\_id: { type: String, unique: true },  
 comment\_by: { type: String },  
 comment\_to: { type: String },  
 content: { type: String },  
 post\_date: { type: Date },  
 isDel: { type: Boolean }  
}

#### 4.交易表 exchange

{  
 exchange\_id: { type: String, unique: true },  
 seller: { type: String },  
 buyer: { type: String },  
 addon: { type: String },  
 goods: { type: String },  
 *//提供商品快照，以保证商品被删除时无法正常显示之问题* goods\_snapshot: { type: Object },  
 desc: { type: String },  
 price: { type: Number },  
 pay: { type: String },  
 status: { type: Number },log: { type: Object },  
 post\_date: { type: Date },  
 isDel: { type: Boolean }  
}

#### 5.收藏表 favorite

{  
 favorite\_id: { type: String, unique: true },  
 favorite\_by: { type: String },  
 refer\_to: { type: String },  
 post\_date: { type: Date }  
}

#### 浏览历史表 history

{  
 history\_id: { type: String, unique: true },  
 viewer: { type: String },  
 history\_to: { type: String },  
 post\_date: { type: Date }  
}

#### 支付表 pay

{  
 pay\_id: { type: String, unique: true },  
 *//指向哪个交易* pay\_to: { type: String },  
 *//包含支付方法名称和支付内容（价格或等值商品）* pay\_content: { type: Object },  
 post\_date: { type: Date }  
}

#### 用户表 user

{  
 userid: { type: String, unique: true },  
 username: { type: String, unique: true },  
 headImg: { type: String },  
 email: { type: String, unique: true },  
 *//通过sha1加密* password: { type: String },  
 gender: { type: String },  
 birthday: { type: Date },  
 city: { type: String },  
 desc: { type: String },  
 isDel: { type: Boolean }  
}

#### 消息表 message

{  
 message\_id: { type: String, unique: true },  
 receiver: { type: String },  
 *//商品、帖子、交易、账号* type: { type: String },  
 title: {type: String },  
 content: { type: String },  
 post\_date: { type: Date }  
}

#### 黑名单表 blacklist

{  
 black\_id: { type: String, unique: true },  
 refer\_to: { type: String },  
}

#### 举报表 report

{  
 report\_id: { type: String, unique: true },  
 refer\_to: { type: String },  
 report\_by: { type: String },  
 reason: { type: String },  
 other\_reason: { type: String },  
 isDone: { type: Boolean },  
 post\_date: { type: Date }  
}

#### 管理员令牌表 auth\_token

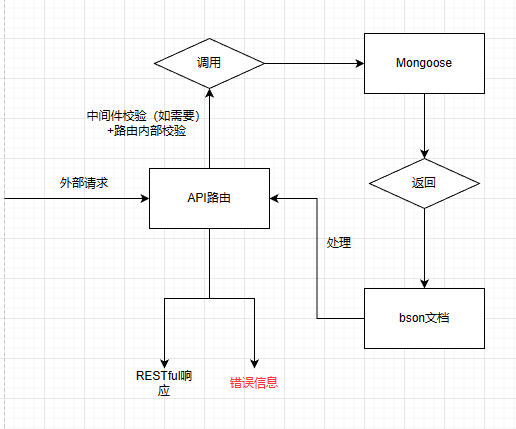
{  
 token\_id: { type: String, unique: true },  
 valid\_time: { type: Date }  
}

# 3数据结构和处理流向

## 3.1运行设计

### 3.1.1 运行模块的组合

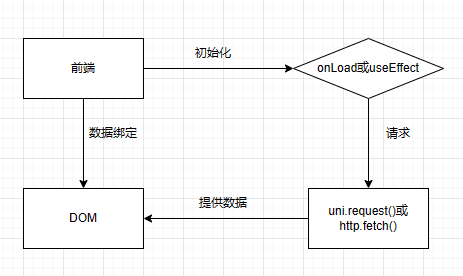
#### 1.后端框架运行



根据不同的请求接口路径，来触发不同的API路由，并进行参数校验，当参数全部正确，或者没有传入参数但是参数有默认值时，则调用Mongoose方法，在Mongoose方法里面使用相应的参数返回特定的文档事件，获取数据之后API路由会把数据进行处理，最后通过封装的请求规范化模块把返回的数据解析成规范的格式输入给前台的请求。

Mongoose通过定义Schema和Model规范数据，由于MongoDB的数据组织自由度较高，可以首先定义然后创建数据集合，如有进一步的需求，可以通过修改Schema和Model来对字段进行增减。实现Mongoose Schema的Model可以直接调用MongoDB的常规增删改查方法，也可以使用聚合管道进行高级查询。

#### 2.前端框架运行



前端根据不同的角色（客户端和管理端），分别采用Vue（此处着重uni-app）和React（请求使用Tauri封装的API），当页面加载完毕之后，前者调用onLoad方法，而后者灵活使用useEffect Hook钩子，对一些属性进行初始化和加载操作，通常是请求后台的数据渲染在页面上面，此时获取的数据全是默认值获取，也不带条件查询，对于后台的请求，前者使用uni-app自带的ajax请求方法，后者则使用Tauri框架封装的高性能fetch请求，更加方便地处理AJAX请求。当每次用户有点击、选择等操作时，有可能会需要更新数据，此时调用uni.request()获取后台返回数据，然后通过vue的双向绑定数据，对DOM树进行修改，不再跳转页面，实现局部刷新；而后者则通过fetch() 获取数据后，使用state setter更新数据，同时渲染新的页面。

### 3.1.2 客户端页面展示（部分）

#### 1.首页



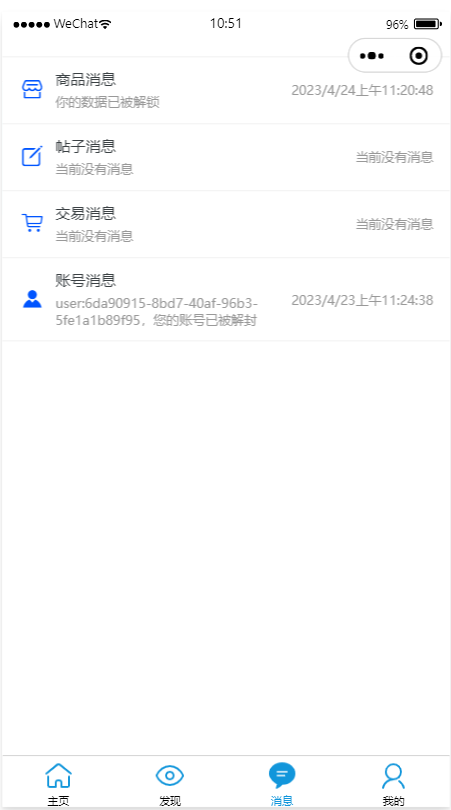
首页主要展示商品信息，支持下拉刷新和底部请求。顶层的搜索框点击后跳转至搜索页面，搜索商品内容。其下为推荐商品，在后端使用聚合管道随机抽取4个展示。所有包含商品信息的数据单元都可以跳转至相应的商品界面，跳转时会添加商品id。

#### 2.发现（即帖子）



发现页面主要显示帖子，与首页一样，支持下拉刷新和底部请求，数据拼接方式也相似。同样地，当点击任意一个帖子时，会跳转至相应的帖子页面，此时附上帖子id。而顶部的搜索框则以搜索贴子的方式跳转至搜索页面。

#### 3.消息



消息页面主要查看消息。该页面是客户端中唯一需要使用Socket.IO的页面，通过Socket.IO实时接收最新消息（即触发请求最新消息的方法），并渲染之。点击不同类别的消息列表项就会进入相应类别的消息详情页面。

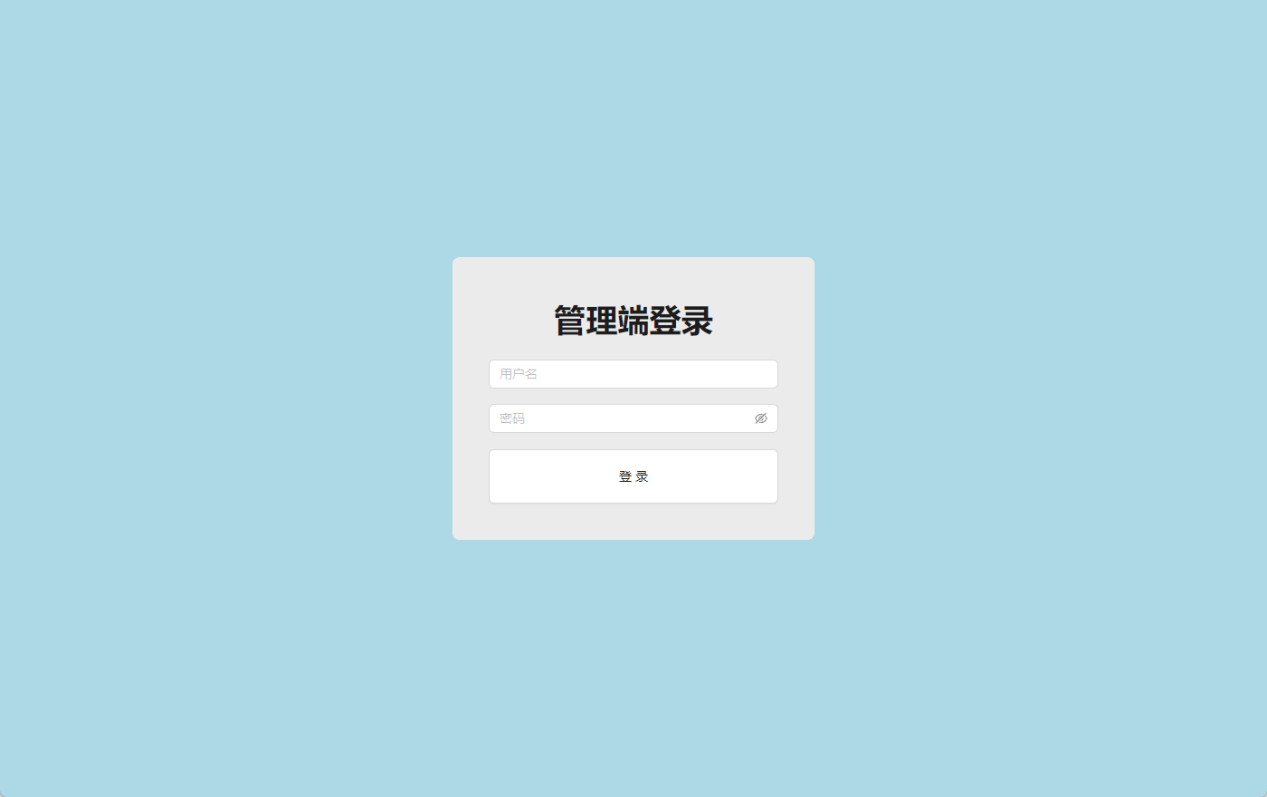
#### 4.“我的”



“我的”页面可以查看当前登录的账号，如未登录，用户信息部分则显示登录按钮，点击后跳转至登录界面；登录后，点击头像可直接替换之（可拍照亦可选择文件，首先上传，然后使用返回的文件路径修改个人信息以实现），点击用户信息部分即可查看自己的账号信息。在本页面，用户可以选择查看收藏的数据，历史浏览，发布的数据，进行中的交易、历史交易（买卖双方）。

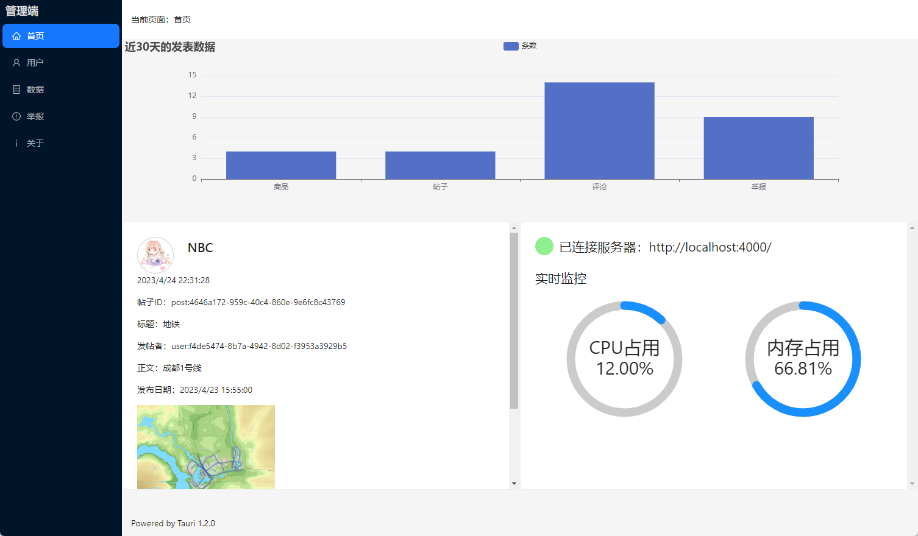
### 3.1.2 管理端页面展示

#### 1.登录，验证身份



在登陆页面键入账号和密码，管理员账户固定为admin，密码为123456。登录信息正确，则写入MongoDB（有效期为1天）和Redis（有效期为4小时）中。

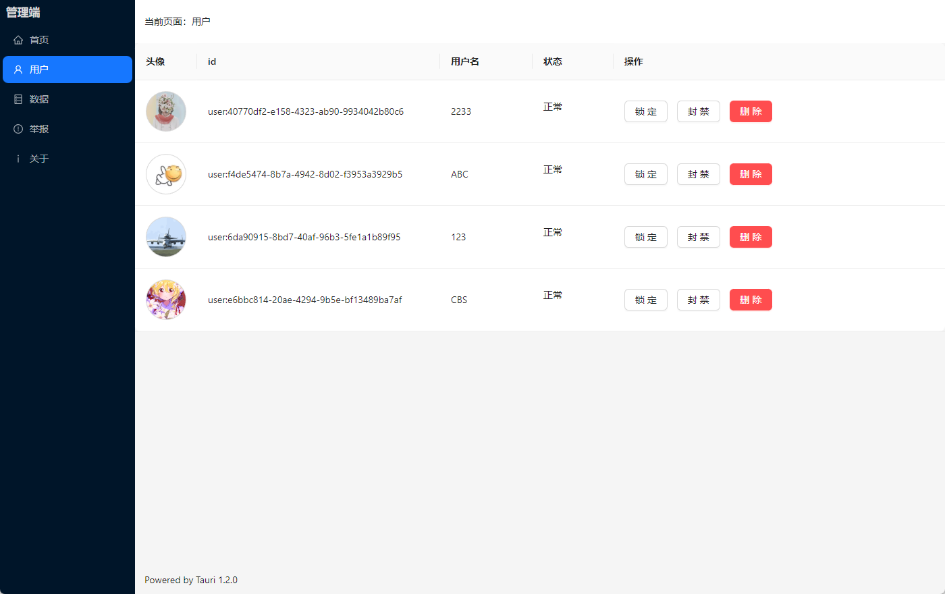
#### 2.主页



在主页面中，左侧栏有所有的操作选择，选择每一个操作可在右边的空间进行不同的操作，此时右边头部会出现导航栏，以提示当前操作页面和当前已打开的页面浏览，可以通过点击左侧侧边栏进行更换不同的页面。

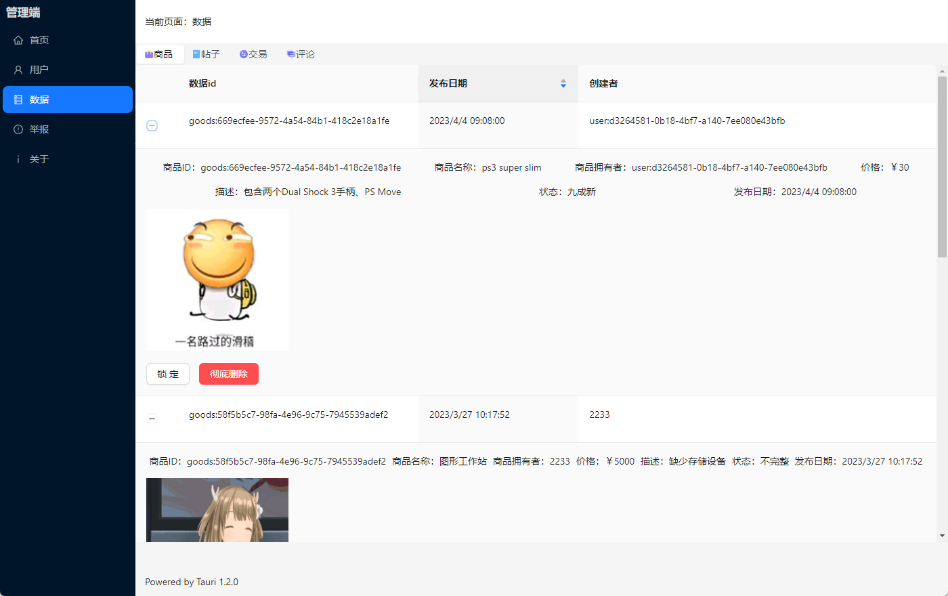
首页主要展示近段时间所有用户所发布的数据，并以图表的形式可视化。下方可以对最近发布的举报进行快速之处理，不用先跳转至举报页面再处理。右侧则显示服务器实时状态，通过Socket.IO，可以查看当前连接状态、服务器CPU占用率和内存占用。

#### 3.用户管理



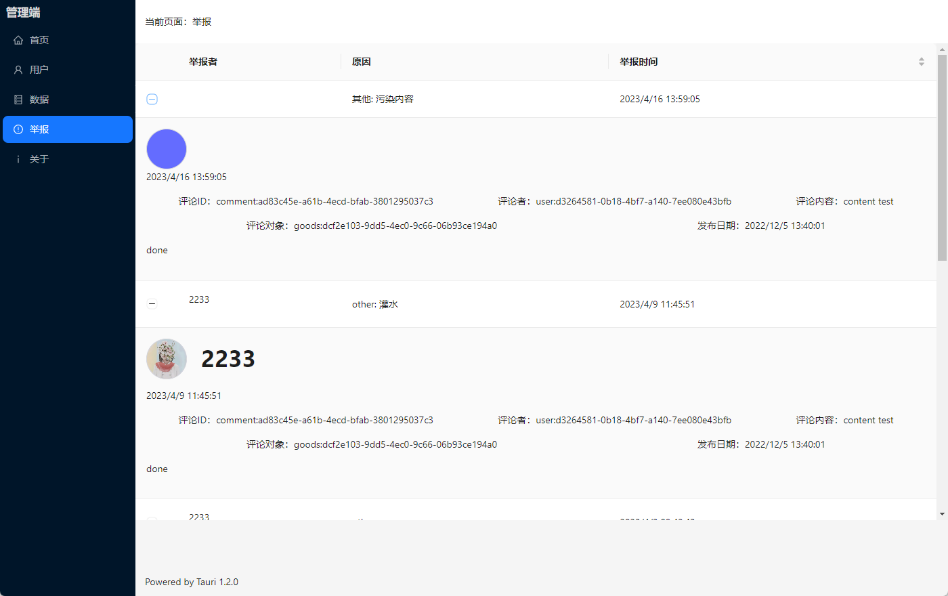
用户管理界面，在这里你可以看到所有的用户信息，为了保护用户隐私，一些信息没有在此显示，本页面可以查看用户的id、用户名、头像和状态，可以进行锁定、封禁、彻底删除等操作，可以通过用户名模糊检索。

#### 4.数据管理



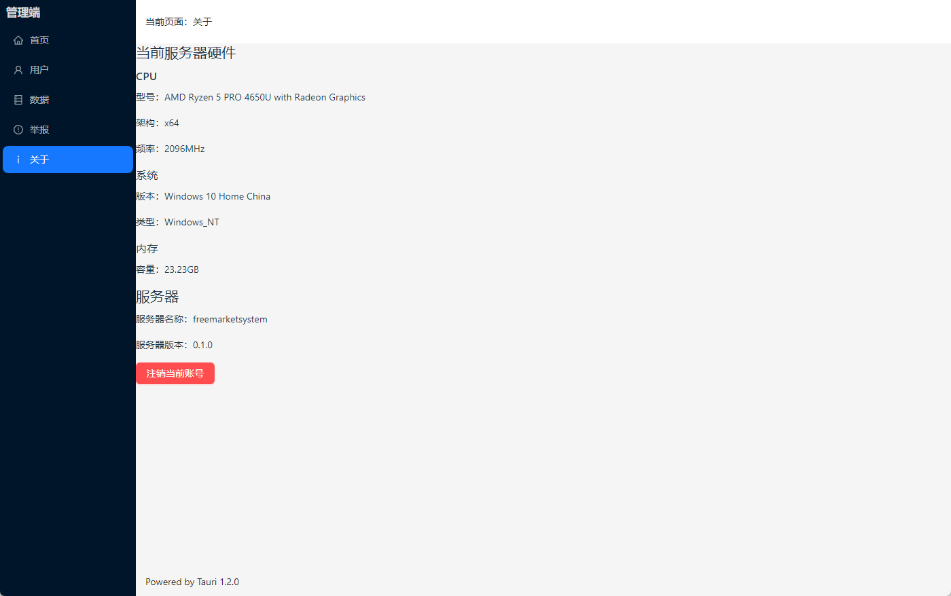
数据管理页面，本页面展示了对商品、帖子、评论和交易数据之管理，通过切换表格上的选项来预览不同类型的数据（会重新向服务器请求数据），可以变换日期排序，也可以设置时间范围以查看该范围内的数据，该页面展示了不同数据类型的基本信息，你可以通过展开表格的一个行来查看详细数据和进行一些操作。

#### 5.举报管理



举报管理界面，展示所有的举报信息，本页面可以查看举报者和被举报的数据，可以通过其他原因和时间段缩小查询范围，选择一条举报，可以查看该举报的详细信息，同时进行操作（规范管理员的处理举报流程：不管有多急迫，首先需要查看全貌）。同样地，可以更改举报时间的排序。

#### 6.关于



关于页面，可以查看当前服务器的一些信息，如基本的配置信息、服务器名称和版本。该页面提供了注销当前账号的选项，执行后，MongoDB和Redis中的令牌会被清除，并跳转至登录页面。

## 3.2 出错处理设计

### 3.2.1 出错输出信息

#### 1.控制台输出

在WebStorm和类谷歌浏览器开发人员工具的运行控制台中，输出代码错误信息，一般在控制台输出的信息，除了人为，其他都未预料到或者说未捕获到的异常错误，在控制台会输出错误索引和错误信息简介，可以根据提示消息找寻错误的代码所在位置，以做修正。

#### 2.接口返回信息输出

在前端请求数据时，如果未带必选参数，或者进行无权操作，获取访问不属于自己可以接触的信息时，后台会向前端发送错误提示，并且带上信息和错误编码，这种错误都是可预料的、可捕获的，属于项目中可以预料和判断的错误。

### 3.2.2 出错处理对策

#### 1.全局异常捕获

Express.js通过中间件采用全局异常捕获。通过Express-Builder构建的项目默认打印错误信息并停止，此时使用nodemon依赖可以快速重启（输入rs）。而部分模块由于采用了特殊的异常，在抛出时不会导致后端停止运行。

#### 2.弹窗或其他方式

前端是采用前后端分离处理方法，所以每次获取数据的时候，不是进行刷新页面，而是用AJAX来进行异步获取数据，所以当获取数据失败的时候，我可以提前预测这个情况的发生，在JS文件中，在每次接收到返回的response时，后台返回的都是规定的result格式，所以判断返回的code就知道是否是错误或者失败的请求，当code编码大于等于400时，代表请求出现错误，然后根据错误的信息和编码，通过uni.showToast()等方式来提醒操作人员，操作人员就可以知道自己的错误操作所在。

4 项目总结

## 4.1 感想

今年形势异常困难，由于疫情，IT行业裁员，就业难度较大，尽管如此，我仍然克服了这些不利条件，以较高的水准完成了这个毕业设计（主要是对心理影响较大）。

技术上，我之前经常使用Java技术栈，但因为一些原因，转到了Node.js技术栈上，并为了完成iOS开发学习了一段时间的Swift语言，不过基于较强的自学能力，我不仅度过了转型期，也在空闲时间中拓展了一些技术栈知识（如React、Koa.js），未来虽然会扎根于Node.js技术栈上，但Go、Rust等技术也是未来拓展之方向。

对于这个毕业设计，技术方面，虽然在其中使用了新的技术栈，并且踩过不少的坑，但这些坑在另外一方面确实是一个宝贵的开发经验。未来，我可能会继续使用一些技术栈，有可能更新一些，也有可能抛弃一些，但无论如何，这些技术栈都要落实到生产环境上，所以一定要认真、深度学习应用。业务方面，由于没有参与过真正的实战开发项目，开发思路上会有一些欠缺，这点也需要注意。

## 4.2总结

三年过去了，突然面临毕业，无论何人都有话说不清。大学生活历历在目，在这三年的学习生活中，收获了很多，而这些成绩的取得，除了不断的自学外，还有一直关心帮助我的人的支持。在大学最后的时间里，很认真的做完了毕业设计，因为它是我大学的最后一次作业了，也是检验我学习成果的最后一次机会。完成了它我大学生活接近尾声了，但对技术的学习却不会停止。回想我的三年大学生活，感触很大。在这里的三年时间我从一个空有理想且懵懂的少年成长为一个懂事懂礼懂得做人的人，在这里我不但学到了文化知识，实践动手能力，更形成了我的许多价值观（虽然我这一代有提前形成自身价值观的），学会了许多做人做事的道理。最后谢谢我的母校对我的教育，我将以更加饱满的热情投入今后的工作中，去完成自己的理想!