华南师范大学
SOUTH CHINA NORMAL UNIVERSITY

本科毕业论文

论文题目：知识产权交易平台的设计与实现

指导老师：王　腾、朱定局

学生姓名：马逸青

学　　号：20142100008

学　　院：计算机学院

专　　业：计算机科学与技术数据库方向

班　　级：2014 级本科 2 班

摘要

在新知识创造日新月异的新时代，知识产权逐渐为人们所重视，为了给我们企业与我们的知识创造者提供更加简单直观的交易体验，我们团队决心开发基于大数据的知识产权交易平台，为知识产权创造者和企业带来方便和效益。

本文主要介绍知识产权交易平台的设计与实现，该平台给用户提供查看知识产权产品、与发布者取得联系、发布知识产权信息的功能，给管理员提供用户管理、数据维护的功能，其开发包括后台数据库的开发、后端 API 服务的开发和前端界面的开发。

本平台后端 API 服务运行于 Node.js 运行环境，速度快、处理能力强，适合知识产权交易平台此类 I/O 密集型 Web 应用，其丰富的模块和 npm 包也为平台开发提供便利化和结构清晰化；采用 MongoDB 作为网站数据库，性能高，扩展能力强，为平台提供稳固保障的数据支撑，也为后续功能开发提供广阔空间；采用 Vue.js 作为前端开发框架，其数据驱动和组件化的特点为平台的开发提供便利的 MVVM 开发模式；采用 Element-UI 响应式布局框架作为前端界面，其提供的丰富功能组件实现数据校验、页面自适应等多种功能。平台通过前端页面与后端服务分离为平台的开发提供更多可能，也可以有效减少 web 数据输送流量，同时还可以将平台进行功能化区分，将数据库与 API 服务器与静态资源服务器分开，为平台的高稳定性和高并发性提供支持。

关键词　:知识产权，交易平台，Node.js，MongoDB，Vue.js

Abstract

In today's continuous innovation of knowledge, intellectual property has gradually been valued by people. In order to provide our companies and our knowledge creators with a simpler and more intuitive trading experience, our team is determined to develop an intellectual property trading platform based on big data to provides convenience for intellectual property creators and businesses.

This article mainly introduces the development of the intellectual property trading platform. The platform provides users with the ability to view intellectual property products, communicate with publishers, and provides administrators with user management and data maintenance functions.

This platform uses Node_js as the backend API server. Node_js is based on the Chrome V8 engine, it is suitable for I/O-intensive web applications like intellectual property trading platforms, its rich module and npm package also provide platform development with ease and structure clarity. The platform uses MongoDB as the website database and MongoDB is a database based on distributed file storage. It has the dual characteristics of non-relational databases and relational databases. It provides powerful data support for the intellectual property trading platform and also provides ample space for follow-up function development. The platform uses Vue_js as the front-end framework. It has data-driven and componentized features, and provides a convenient MVVM development mode for the development of intellectual property trading platforms. The platform uses the Element-UI responsive layout framework as the front-end interface, and utilizes the rich functional components it provides to implement data verification, page adaptation, and other functions. The platform provides more possibilities for the development of the platform through the separation of the front and back ends. At the same time, it is conducive to differentiation of platform functions, separating the database from the API server and the static resource server, and providing support for the platform's high stability and high concurrency.

Keywords :Intellectual Property, Trading Platform, Node.js, MongoDB, Vue.js

目录

1. 绪论

1.1 选题背景

在新的知识、技术、创新创造日新月异的现代，知识产权逐渐为人们所重视，更多的知识创造者开始申请专利，这一方面是为了通过法律的手段保护自身的知识产权专利，另一方面是为了知识产权可以得到尊重，可以在交易市场中流通创造价值。近年来，国家也在不断出台新法律、加强监管措施、完善举报渠道以保护知识产权，这为鼓励知识创新、创造打下了基础，然而在保护知识产权的过程中，我们也要做到疏与堵结合，不仅要堵住知识产权的抄袭盗用行为，也要疏通知识产权的交易流通，让知识产权造福我们的企业、产品，也造福我们的知识创造者。

国家也对互联网加以重视，推出了“互联网+”的创新理念，我们现在已踏入“互联网＋”的新时代，许多传统的产业、工作方式通过与“互联网+”结合，成功实现了新发展。

在新的 Web 网络技术不断被提出并应用的今天，计算机硬件和软件不断发展，Web 开发不再遥不可及，学生也可以通过自身的知识学习，借助成熟的开发框架和设计方法完成 Web 网站的开发。

为了给企业与知识产权创造者提供更好的交易环境，市场上相继诞生了一些知识产权交易平台，但这些平台大多运营时间短，缺少专业的运营团队，在平台更新和用户体验上普遍落后，急需开发一个安全可靠、数据齐全的知识产权交易平台。

1.2 选题意义

为了给我们企业与我们的知识创造者提供更加简单直观的交易体验，我们团队决心开发基于大数据的知识产权交易平台。知识产权创造者可以通过平台发布自有知识产权信息，既增加知识产权的影响力，扩大创造者知名度，也可以为知识创造者寻得有意合作的合作方。企业可以通过该知识产权交易平台检索需要的知识产权，了解知识产权的详细信息并进行对比，还可与发布者取得联系，详细了解有关知识产权的具体细节。而我们平台作为可信的第三方，可以为他们双方提供政策上的帮助指引，避免不必要的的交易纠纷。知识产权创造者、企业与我们平台三方通过这一平台，跨越现实的地理间隔，实现知识产权交易全球化、简便化、信息化。我们团队也在项目中引入大数据工具，将互联网上现有的知识产权信息进行分析整合，方便需求者检索。

平台提供知识产权发布功能，知识创造者可以在平台发布自己的知识产权产品，而游客和其他用户可以对这些知识产权信息进行详情查看，还可以通过发送意向的方式将自己的联系方式和留言发送给知识产权创造者，知识产权创造者可以根据意向者发送的信息与意向者取得联系，并与满意的意向者私下完成交易，而我们平台可以在这个过程中为双方提供可靠的政策服务。

平台采用 B/S 架构，企业与知识创造者都可以方便的通过电脑或手机的浏览器进行浏览，平台采用自适应的界面设计，在电脑端和手机端都有良好的体验，交互性强，且网站在运营阶段升级功能无需升级客户端，只需更新服务器代码即可，而 Node.js + MongoDB + Vue.js 的组合学习难度低，开发简便，扩展性强，还能在网站访问量逐渐提升后适时地进行扩容，这样的开发技术已渐渐为开发者接受与应用。

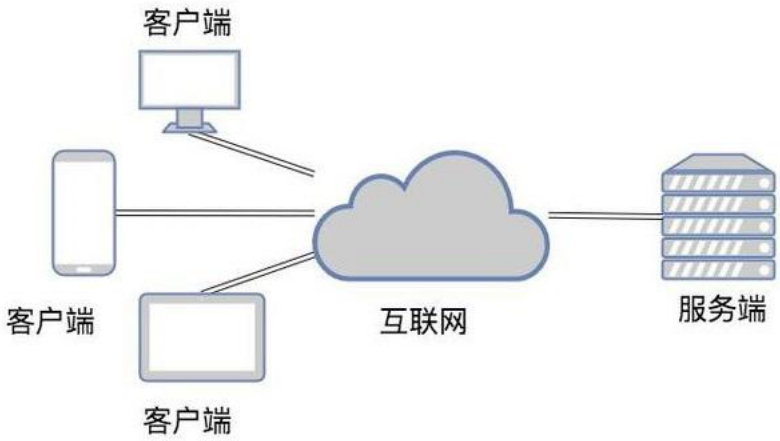

图 1：B/S 架构图

2. 需求分析

2.1 用户需求

知识产权交易平台作为服务知识创造者和企业的开放式平台，主要用户就是我们的知识创造者和我们的企业，针对这两大类用户的特点我们对平台的需求进行了分析：

1. 界面清爽：清爽优雅的平台界面是吸引知识创造者和企业的第一步，只有平台界面功能清晰明确才能吸引双方继续选择我们的平台，继而在平台上发布产品和交易。
2. 账号信息安全：平台用户在使用过程中会对平台安全性进行留意，如果平台存在安全漏洞，用户会对平台可靠性产生怀疑，不利于平台长期运营，因此平台在用户认证机制和密码存储需要做特别的考虑。
3. 操作简便：平台有两个主要的功能，一是知识产权发布，二是交易意向发送，在这两个功能的使用上既要保证功能齐全也要保证操作的简便，过于繁琐的操作过程会让用户无从下手甚至放弃我们平台的使用。
4. 产权发布快速：知识创造者希望产权在发布后，其他用户可以快速地从平台首页查看到产权的详细信息。
5. 意向发送便捷：当用户在平台查看到合适的知识产权信息时会希望能够第一时间将自己的意向信息发送给知识产权所有者，所有者可以在收到意向后及时与意向者取得联系。
6. 意向信息直观：产权所有者在收到众多交易意向后希望能够直观地看到所有意向信息并做对比，并与满意的意向者完成知识产权的交易。

2.2 功能需求

知识产权交易平台将根据购物网站的设计经验进行开发，平台包括三部分：前台展示部分、用户中心部分、管理员中心部分，具体实现以下功能：

1. 用户登录模块：通过生成识别用户的 token 存放至用户的浏览器中，以此辨别用户是否登录。

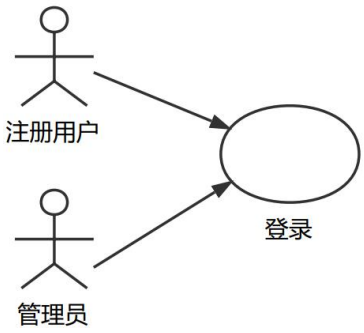

图 2：用户登录模块用例图

2. 用户登出模块：通过清除用户浏览器中的 token，重置用户的登录状态

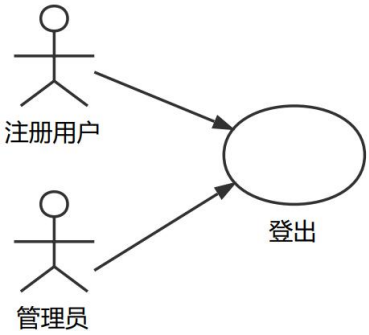

图 3：用户登出模块用例图

3. 用户注册模块：对用户信息进行检查，检查无误可进行注册。

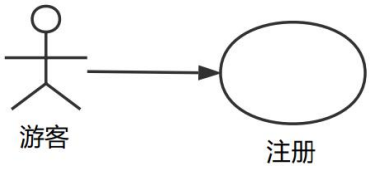

图 4：用户注册模块用例图

4. 获取用户状态模块：根据用户请求头中 cookie 里的 token 信息，判别用户登录状态并返回一些用户基本数据。

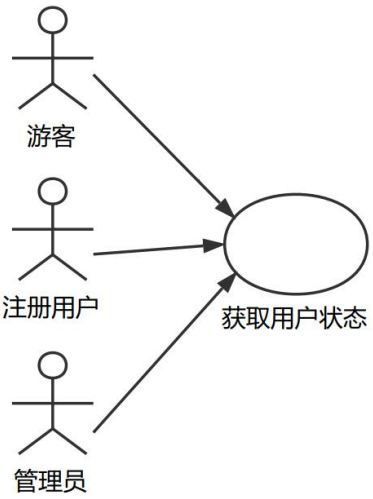

图 5：获取用户状态模块用例图

5. 获取用户资料模块：查询用户基本资料并在用户中心展示。

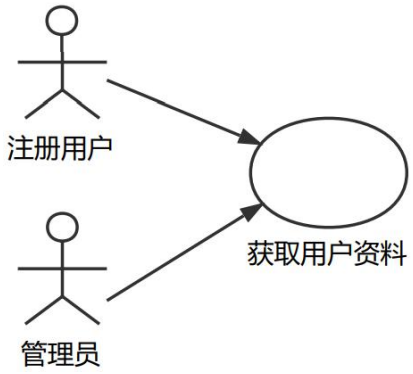

图 6：获取用户资料模块用例图

6. 用户资料修改模块：用户对个人资料进行修改。

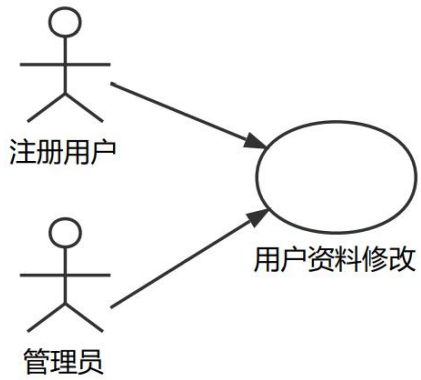

图 7：用户资料修改模块用例图

7. 获取验证邮件模块：为保证平台安全，用户需要在验证邮箱后才可进行平台信息发布和修改操作。

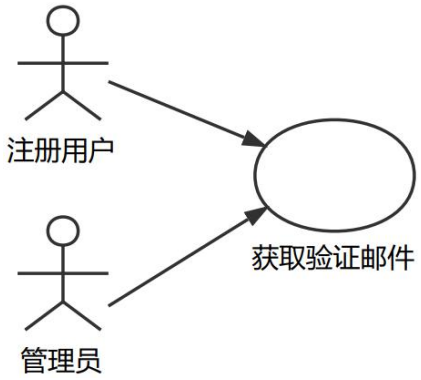

图 8：获取验证邮件模块用例图

8. 找回密码模块：用户可以通过注册账号获取网页链接来重置密码。

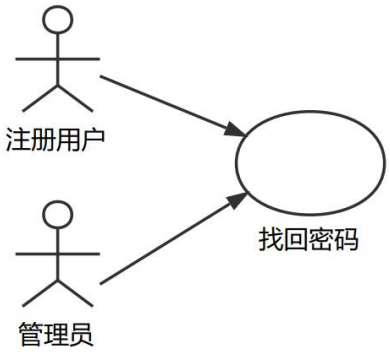

图 9：找回密码模块用例图

9. 修改邮箱模块：邮箱修改涉及新旧邮箱的替换机制，平台只在用户通过新邮件验证后才对邮箱进行更新。

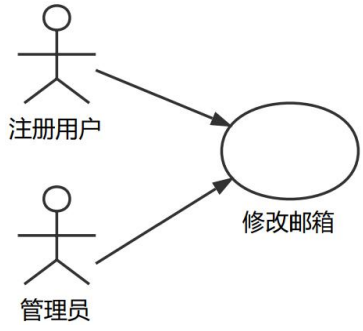

图 10：修改邮箱模块用例图

10. 修改密码模块：在验证原密码后对新密码加密后进行存储。

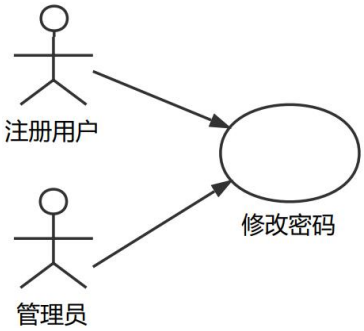

图 11：修改密码模块用例图

11. 获取所有用户信息模块：管理员获取用户信息列表。

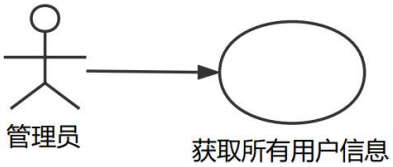

图 12：获取所有用户信息模块用例图

12. 产权发布、修改模块：用户发布新的知识产权信息或对已发布的知识产权信息进行更新。

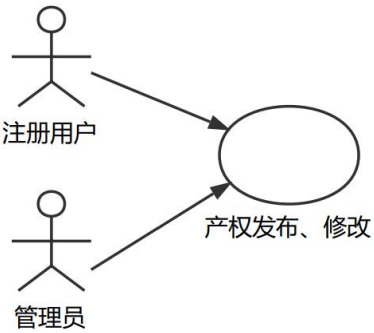

图 13：产权发布、修改模块用例图

13. 获取产权列表模块：用户查阅个人发布的产权信息列表，管理员查看平台所有用户的产权列表，游客可以获取产权的部分列表信息。

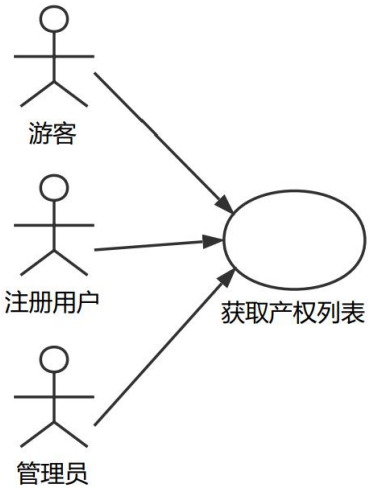

图 14：获取产权列表模块用例图

14. 获取产权详情模块：用户可以查看产权详情，管理员可以查看所有产权的详情，游客能够查阅产权的部分信息。

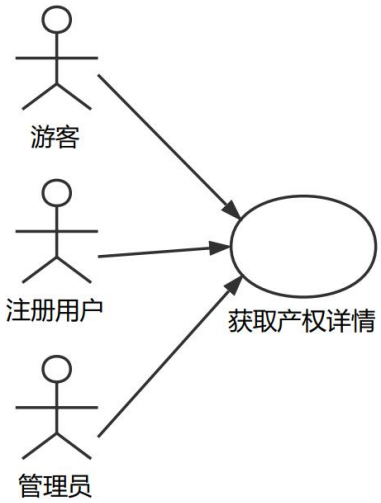

图 15：获取产权详情模块用例图

15. 意向发布模块：用户可以在产权详情页对他人发布的产权信息发送交易意向。

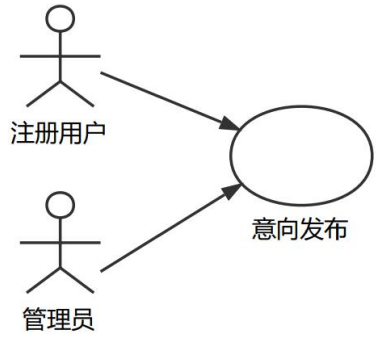

图 16：意向发布模块用例图

16. 获取意向模块：用户查阅自己收到的意向信息，管理员查看平台所有的意向信息。

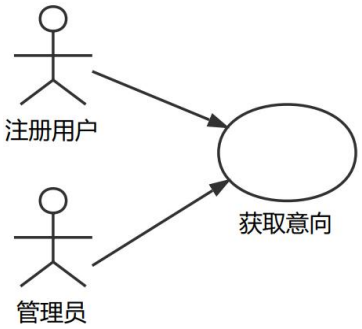

图 17：获取意向模块用例图

2.3 非功能需求

2.3.1 安全需求

平台的安全可靠、用户的账户安全是平台开发工作中的重点，对此，平台在安全性上做了以下考量：

第一，平台根据用户唯一的 user_id 生成 token 并将其存在用户浏览器 cookie 中，根据用户请求头部的 token 对用户身份进行验证。

第二，在用户输入邮箱、密码等涉及关键数据的操作，平台在前端页面和后端服务都采用了严格、同标准的信息校验机制，确保用户账户安全。

第三，平台在存储密码时，对密码进行加盐和哈希，用户校验密码时同样对密码进行加盐和哈希，根据比对哈希结果判断密码是否正确，避免再数据库中存放明文密码带来的风险。

第四，在发送用户验证邮件或密码重置邮件时，平台同样采用 token 验证的形式，token 采用与用户登录不一样的密匙生成，安全可靠。

第五，平台采用中间件的方式对用户权限进行控制，数据请求的权限检测过程安全而且简明易懂。

2.3.2 性能需求

平台作为开放的知识产权交易平台，用户访问量多，且要能够应对突发的大量访问，需要极高的性能。平台按照前端页面与后端服务分离的实现方案，其前端页面不管是在电脑还是在移动端都能稳定运行，平台后端服务运用了异步 IO 的事件驱动模式，可以应对高并发量，对服务器性能要求低，采用的 Koa2 框架具有稳定性高的特点，可以提供全天候的服务。

2.4 可行性分析

2.4.1 经济可行性

知识产权交易平台开发过程中使用的软件都是开源软件，无需任何授权收费负担，且平台采用成熟方案构建，对后端服务要求不高。在开发过程中参考了学校图书馆和电子资源库中的资料，在平台开发过程中，学生使用自有计算机进行开发，使用自有移动端设备测试，且平台项目可在个人计算机上运行。因此，知识产权交易平台的开发、建设具有极强的经济可行性。

2.4.2 社会可行性

社会上知识产权交易需求日趋旺盛，急需这样一个知识产权交易平台来提供服务，平台的开发迎合了社会的发展需求，必将为知识产权创造者和企业带来新体验。平台依托于互联网技术，上线后可以在全互联网范围内访问，受众广、潜在用户多，且平台准入门槛低，只有拥有普通电脑或移动端设备皆可访问，人人都可参与到知识产权交易中来，具有良好的社会意义。

2.4.3 开发可行性

开发前经过讨论与方案的制定，并与指导老师进行了交流，开发方案在讨论中明朗确定。开发前做了详细的准备工作，对方案中未使用过的新鲜工具进行了深入的学习，并参考了许多项目开发经验。在开发过程中使用了版本控制工具，在每天开发工作完成时都将代码提交至 Github 私有库中，保证了代码的数据可靠性，且依托可靠的开发技术，开发可行性极高。

2.4.4 操作可行性

功能上，平台功能模块清晰明了，前台、用户中心、管理员中心模块功能分明，用户不用过多摸索就能学会平台的操作，学习成本低。界面上，平台基于响应式布局，在电脑端和手机端均有良好的浏览效果，且在开发过程中多次对平台进行浏览体验，针对一些体验不佳的地方及时进行了调整。因此，本知识产权交易平台具有很高的操作可行性，可以被我们团队实现。

2.4.5 技术可行性

平台开发运用到了成熟稳定的先进技术，后台服务运行在高性能 Node.js 上，依托异步语法更先进的 Koa2 框架开发，保证后端服务的安全可靠，采用性能优异的 MongoDB 数据库，为网站的开发提供多种功能，前端页面使用 Vue.js 开发，双向数据绑定的开发模式极大地简便了开发过程。

2.4.5.1 Node.js 平台分析

JavaScript 语言运行平台 Node.js 是基于谷歌浏览器 JS 模块开发。Chrome V8 引擎由 Google 公司采用 C++开发，通过在编译方法上的技术改进大大提高了 JavaScript 代码的执行速度，同时其专注于网络功能，在 TCP、HTTP、DNS 等方面相当成熟。Node.js 的开发者 Ryan Dahl 巧妙地借助了 Chrome V8 引擎的高效性和 JavaScript 语言的事件驱动模式开发了 Node.js，Node.js 作为单线程、事件驱动的运行平台，开发时只能使用异步 IO，虽然 Web 开发者更喜欢写同步 IO，但是使用异步 IO 却能获得更高的性能、更高的并发。

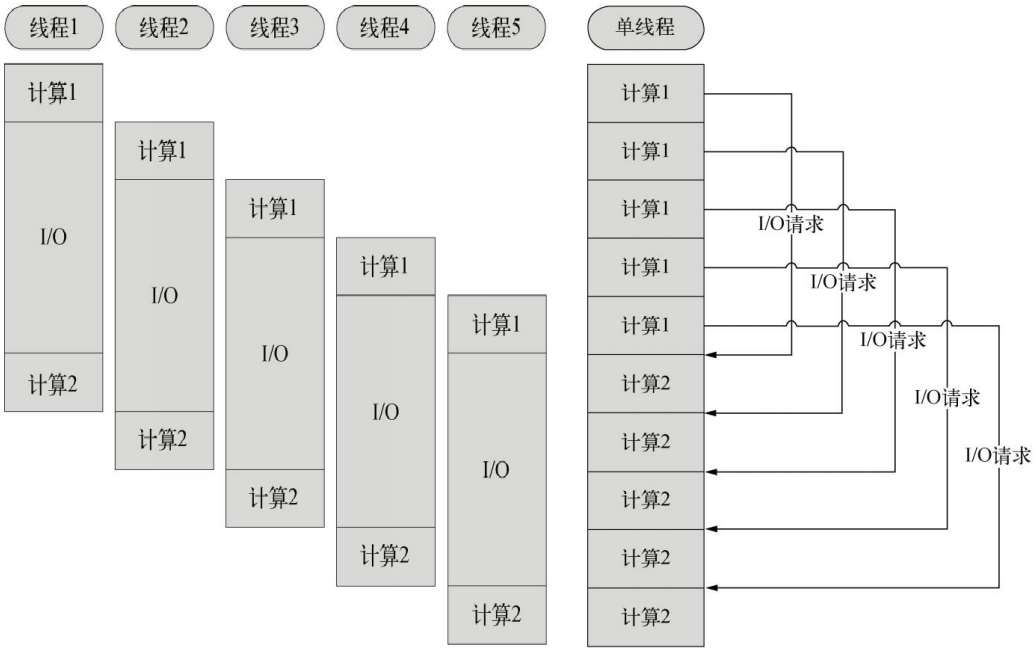

图 18：多线程与单线程的区别

同时，Node.js 作为一个通用的平台，用户可以通过其配套的 npm 管理器免费下载其他用户分享的 npm 包，也可以分享自己的代码包，目前其已成为互联网上最大的开源库系统。当用户发布程序时只需要附上程序使用的 npm 包，其他用户便可自行安装依赖，提高了开发速度。Node.js 不仅可以用来开发 Web 网络服务，得益于其前端、后端统一的模型和接口，开发人员可以在前端和后端开发中灵活地穿梭，现在许多的前端开发框架也开始依赖 Node.js 来编译开发。

传统的服务器开发常常要为高并发而烦恼，多线程在切换线程时会浪费一定的资源，在等待资源时会阻塞线程。Node.js 摈弃了以往 Web 开发语言的线程驱动编程，而采用异步 IO 和事件循环机制，适合用来非阻塞式的平台开发。这里可以这样理解事件驱动，当 Node.js 收到一个新请求时会将其放到事件队列尾部，系统只需集中资源专注处理队列中的一个事件，因而其具有高的运行效率。

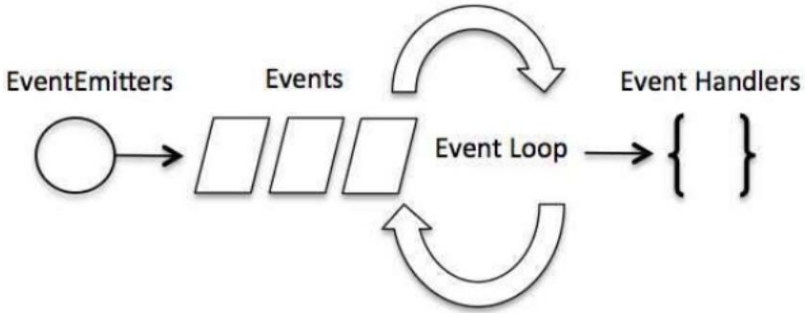

图 19：Node.js 的事件循环机制

知识产权交易平台采用 Node.js 开发，可以应对大量的网络请求，便于以后的升级扩容，且有强大的开发社区支持，为平台提供有力的保障。

2.4.5.2 Koa2 框架分析

Koa2 是继 Express 后的新一代 Node.js 的 Web 开发框架，提供了一部分辅助工具来协助开发后端服务，可以用来快速开发 Web 应用。网络开发框架 Express 对 Node.js 的网络功能进行了封装，使开发者可以用这些封装后的 API 进行开发，但这些 API 的语法基于需要用回调来实现异步的 ES5，在经过多次异步嵌套后代码维护性和阅读性差，给开发工作带来负担。在 Node.js 支持 ES6 语法后，该团队又制作了网络开发框架 Koa1.0，运用 ES6 的 generator 完成非同步操作，代码变得简洁明了。Koa2 是在 Koa1.0 基础上的升级，Koa2 使用了 ES7 语法中的 async 和 await 而不再使用本意并非用于实现异步的 generator，更简洁且易于使用。

Koa2 框架有以下特征：

第一，独特的中间件设计理念，通过中间件来对 http 请求进行处理，开发者可以针对相似的请求开发中间件，轻松实现代码的复用。开发者还可以从 npm 包管理器中引用一些成熟的中间件，做到对文件上传和下载参数格式化、请求权限检查等操作，后端服务接受到的请求就像进入工厂中的流水线般逐一被中间件处理，中间件中调用了 next()下一步函数，在前一中间件处理完成后会被送入下一中间件处理。

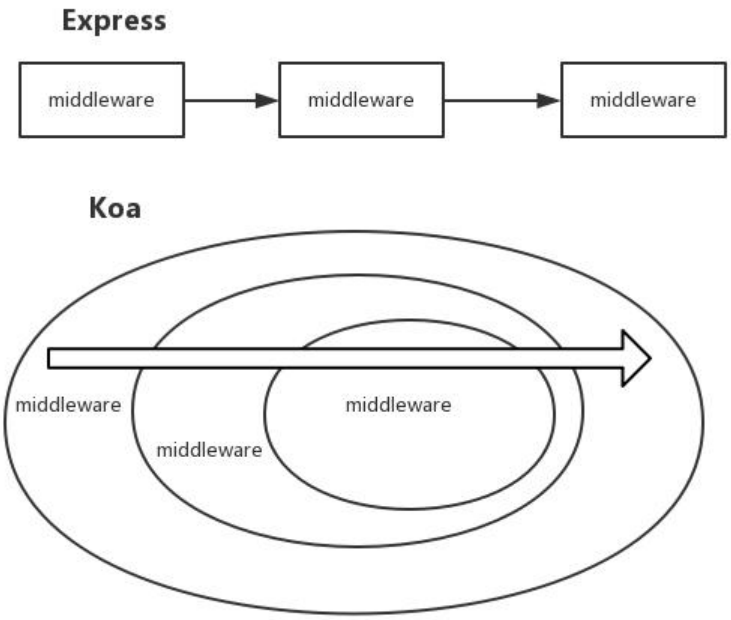

图 20：Express 与 Koa 的中间件对比

第二，通过路由表对不同的请求进行分别处理，将不同的请求地址和请求方式分别传到不同的处理函数，对于路由表中不存在的请求格式给予拒绝，确保了网络服务的安全统一，在定义路由时还可以指定每一路由使用哪些中间件，让整一项目清晰明了。

第三，Koa2 框架支持页面模板功能，在一些前后端不分离的项目中可以方便的输出页面，还可以用来定制邮件模板，发送带有网页样式的邮件。

知识产权交易平台采用 Koa2 框架可以使用上最新的 ES7 异步语法，代码层次更加鲜明，不仅带来了开发上的便利且符合未来技术的发展趋势，为平台的长久运行提供保障。

2.4.5.3 MongoDB 数据库分析

MongoDB 为开发者提供了完整的传统数据库功能，其拥有以下特征：

第一，MongoDB 作为致力于文件储存的数据库，使用简便，可以方便地存储 Json 对象，不需要对数据进行特别的转换。

第二，在 MongoDB 中可以为记录中的属性增加索引，可以给数据快速进行排序，读取速度更快。

第三，MongoDB 具有强大扩展能力，在数据库中数据集增长到一定的高度后，可以对数据库进行分布式扩容，给平台后续建设提供更多可能性。

第四，利用 MongoDB 提供的用来存放小文件的 GridFS 功能，平台的图片文件可以交由 MongoDB 存储，管理图片文件更加清晰简明。

第五，MongoDB 支持许多计算机开发语言，平台可以利用数据库进行大数据开发，为平台提供更丰富的资源。

知识产权交易平台采用 MongoDB 数据库，既为网站用户、产权数据提供稳固的支持，也为网站未来更复杂、更加多样化数据存储提供可能。

2.4.5.4 Vue.js 前端框架分析

Vue.js 前端框架是用于创建用户网页界面的开源 JavaScript 框架，更加关心 MVC 模式中的视图层，通过组件中的方法实现视图与模型间数据的绑定，使得创建单页面应用时页面数据更新更加简便，很大地便利了前端页面的开发，开发者可以将精力更多地放在数据的获取、处理和发送。

Vue.js 具有以下特征：

第一，Vue.js 前端框架具有先进的组件开发思想，过去在后端才能实现的组件开发、组件复用成功的在浏览器前端实现，组件开发者可以更集中精力于组件的开发与复用，每一组件可以完整地包含网页代码，而且开发者可以方便地将组件打包上传到 npm 包管理器，与其他开发者共享组件。

第二，Vue.js 前端框架采用了 responsive 设计，开发者只要将视图与模型进行绑定，Vue 会在数据中加入监视函数，当数据变化时视图也会相应的改变，而且 Vue.js 框架提供了许多操作视图的功能，html 代码也能实现 if 条件判断、for 循环等功能，前端开发人员不再需要操作复杂的 html 代码拼接。

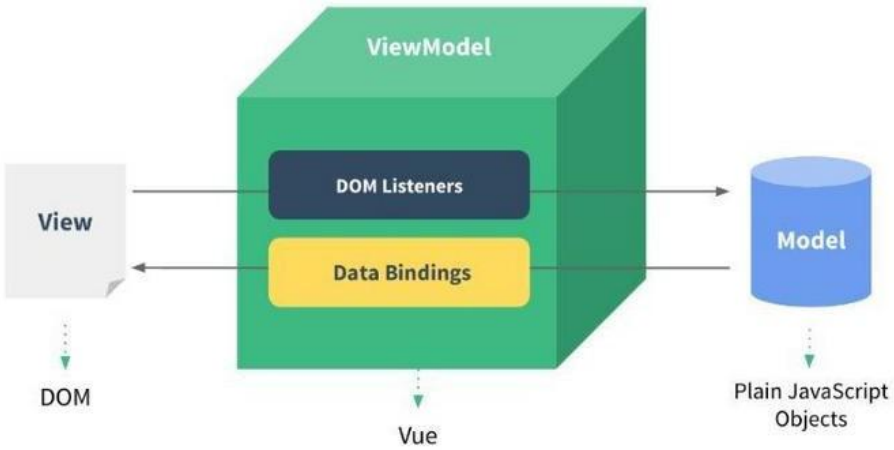

图 21：Vue.js 的双向数据绑定模型

第三，Vue.js 前端框架提供了一系列配套的包，包括用来构建复杂多页面应用的 vue-router、用来实现数据共享的 vuex、用来构架包含 Vue.js 的 webpack 项目的 vue-cli 脚手架等包功能，开发者可以根据开发需求在项目中使用这些包，还可以用上 axios 来实现 http 请求的发送与接收。

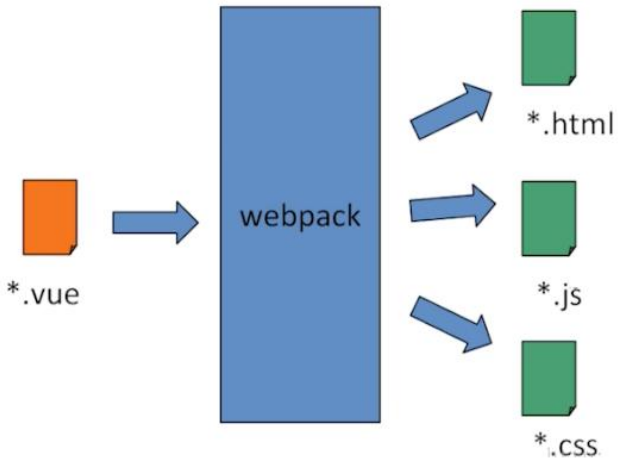

图 22：Vue.js 组件通过 webpack 打包成网页代码

第四，Vue.js 前端框架的开发团队中有不少中文开发者，遇到问题可以在社区中提问交流，开发团队也在持续更新框架功能，一些框架中存在的小问题都可以迅速得到修补。

第五，Vue.js 前端框架虽然是 2014 年新发布的框架，但是其星标数在开源代码分析网站 Github 上排名前十，互联网上关于它的问题讨论与文章数量多，许多新网站项目也用上了这一框架。

第六，Vue.js 前端框架的良好生态催生了一批基于 Vue.js 框架开发的项目，Element-UI 就是基于它开发的前端页面 UI，前端开发者可以借用其成熟的组件进行前端开发，其组件不仅提供了良好的配色，还提供数据校验等功能。

知识产权交易平台采用 Vue.js 前端框架搭配 Element-UI 组件库开发，有利于开发高效化和维护条理化，是平台开发的创新尝试。

2.4.5.5 前后端分离分析

在 MVC 架构中，后端服务在收到浏览器发送的请求后对请求进行解析，向数据库查询所需的数据，在对数据进行格式化后填充至模板，返回经过填充的网站页面至用户浏览器，完成这一请求功能。这种做法虽然可以减轻浏览器端的压力，却对后端服务器提出了巨大的性能要求，后端需要耗费大量的资源，而传输的数据有很大一部分是重复的，真正有关用户的数据却很少。随着互联网技术的发展，人们对网页的需求不再仅仅停留在能用，而要既好用又好看，网页内容的增大对后端服务提出了更严峻的业务需求。这种情况下，静态页面通过 JS 发出异步 http 请求，后端服务不用再对每个请求都返回页面而只要返回部分 JSON 数据。

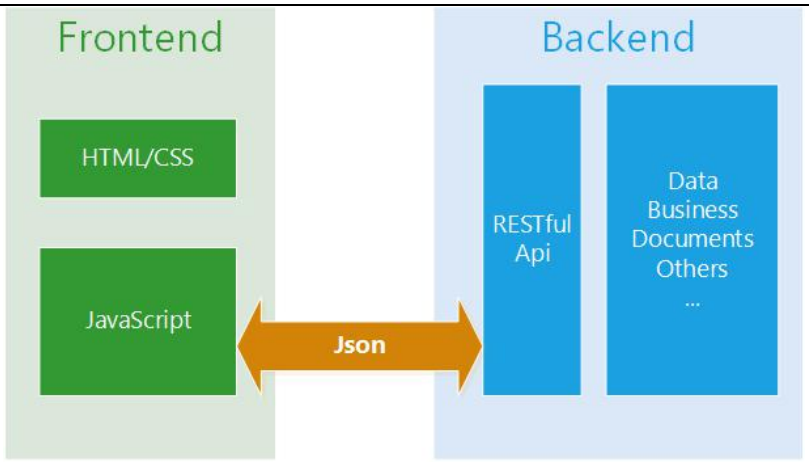

图 23：前后端分离架构图

而在电脑设备、移动端设备处理能力不断提升的今天，前端处理能力已不再是网络技术的瓶颈，人们越来越多地关注后端服务的高并发性和高可用性，渐渐希望能将尽可能多的请求由过去的返回页面转变为返回格式化的 JSON 数据。在网络技术的支撑下，前端页面与后端服务分离的想法慢慢地被开发者实现，用户访问到的网页完全可以存放在静态服务器，而针对数据处理的业务可以放在后端服务器，后端服务器只处理数据和传输数据，很大程度减轻了其压力，可以同时处理更多请求。

前后端的分离也有利于网页的开发，前端工程师专注于前端页面开发，后端工程师专注于后端服务开发，双方只要通过接口文档进行数据对接，同时，后端接口不仅可被前端界面使用，还可以根据这些接口开发对应功能的移动端 APP，实现接口的复用，一举多得。

知识产权交易平台正式采用了前端页面与后端服务分离的设计方案，前端界面 APP 化，与后端服务器通过 json 数据格式交换，使得后端服务稳定且高效。

3. 总体设计

3.1 系统结构设计

根据前端页面与后端服务分离的设计思想，平台前端页面存放在静态资源服务器上，平台后端存放在数据处理服务器，在条件允许的情况下可将数据库放在单独的数据库服务器。当用户通过互联网访问我们的知识产权交易平台时，网址指向静态资源服务器，用户浏览器会从静态资源服务器载入前端页面，前端页面中包含了页面的框架、页面的风格样式和请求数据的方法。在加载完成前端页面后，浏览器会根据页面中包含的接口信息和触发条件依次向后端服务器发起请求，后端服务器接受到请求后对请求合法性进行判断，对不合法请求给予错误响应，对合法请求根据需求向数据库服务器查询数据，将请求到的数据以 json 格式返回给前端页面。前端页面在接受到数据后，按照视图与模型绑定的关系，将数据显示到视图上，这样就完成了页面的显示。

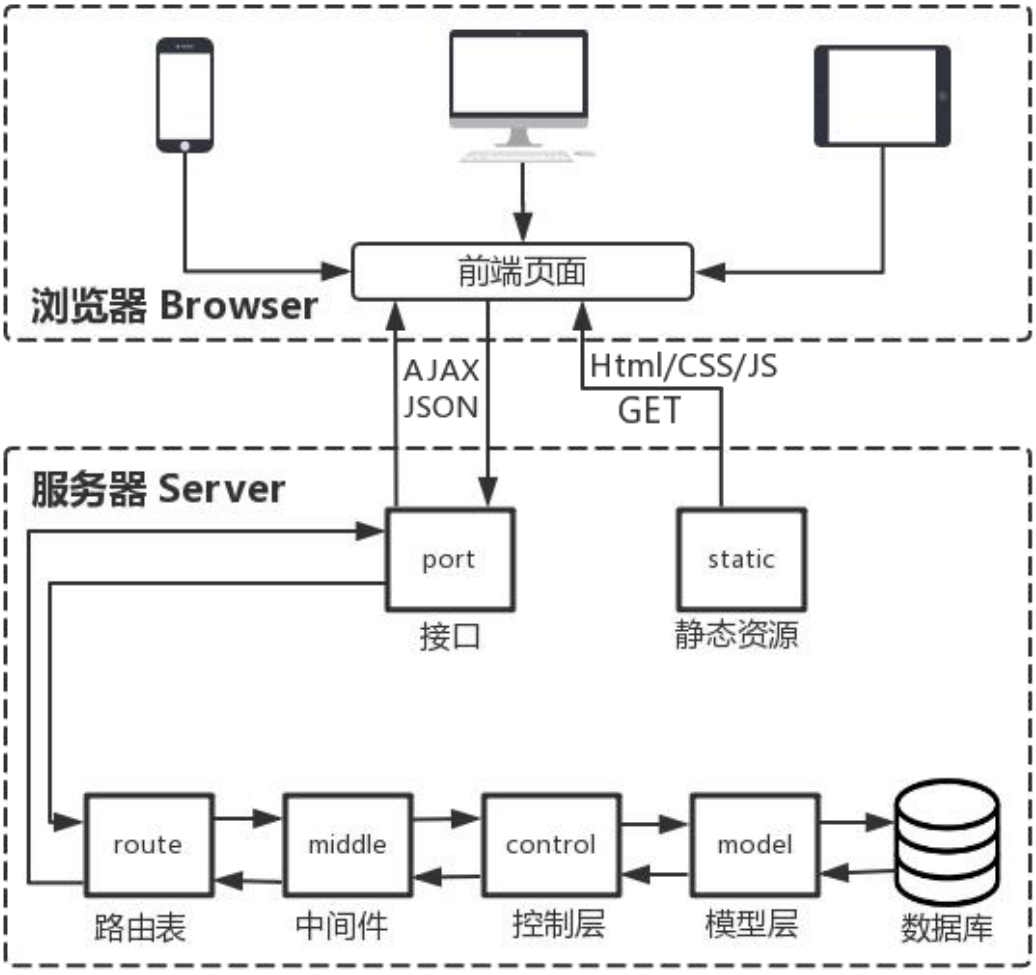

图 24：系统架构图

知识产权交易平台后端采用基于 Node.js 的 JavaScript 运行环境的新一代 Web 网页框架 Koa2 开发，在官方模板范例的基础上，引入了用于密码哈希的 bcrypt、用于连接 MongoDB 数据库的 mongoose 驱动、用于发送电子邮件的 nodemailer、用于对请求格式化的 body-parser 和用于 token 生成的 Json Web Token 等第三方库，根据需求编写检查用户 token 的 checkToken 中间件、用于检查管理员权限的 checkAdmin 中间件、用于检查用户是否验证邮件的 checkEmail 中间件、用于检查用户是否被禁用的 isDisabled 中间件和对管理员和用户公用接口进行处理的 isAdmin 中间件，对于不一样的请求分别调用不同的函数方法进行处理。这样，当一个请求发送至平台后端服务后，会到路由表中根据请求的内容查看需要经过哪些中间件处理，在中间件处理完数据后，可以在请求体中插入一些下一步需要用到的资料，最后送至对应的函数进行处理，处理过程中可能就会用到第三方库功能，最后将响应信息返回给用户。而在这一过程中，任何不合法的请求都会被拦截，返回错误信息。

使用第三方库可以提高后端的开发效率，这里对一些第三方库的使用方法进行举例：

Mongoose：提供多种方法用来方便 MongoDB 数据库的操作，首先建立与数据库的连接并定义集合 Collection 的文档 document 结构 Schema，根据这个结构 Schema 构造出模型 Model，通过这个 Model 即可实现文档的增删改查。Schema 定义时要指明 document 对象的属性，还可以给属性赋默认值，方便新建数据时的自动赋予初始值；Model 提供了多个操作 MongoDB 数据库的方法，比如：插入新数据的 save 方法、查找数据的 find，findById（根据_id 查询），findOne（只查询一条结果）方法、修改数据的 update，updateMany（更新多条数据），updateOne（只更新一条数据）方法、删除数据的 remove 等，通过这些方法可以简化许多数据的操作方法。

bcrypt：用来增强密码存储的安全性，它通过加盐来防御彩虹表攻击，其流程是：随机生成盐值 salt，将 salt 附加到密码后共同哈希，最后返回 salt 与哈希后的值的拼接结果。而其校验流程则是：将 salt 从加密后的密码中剪切出来。将待校验的密码与先前随机生成的 salt 共同哈希，并与先前的哈希结果进行比对，得出密码的正确与否。存储在数据库中的密码由三部分组成，加密轮数、salt 和哈希后的结果，巧妙地设计兼顾了安全与简洁。

图 25：经 Bcrypt 哈希后的密码

JsonWebToken：用来在用户和后端服务之间传递经后端服务签名认证的用户身份信息，方便从后端服务拿到需要权限的内容。为了在无状态的 http 请求中辨认用户，传统的后端服务采用 Session 认证机制来给连接到后端服务的用户分配 session 值，session 值会储存在用户的浏览器中，后端服务根据用户请求时携带的 session 值辨别用户，后端服务会在内存中存放用户的 session 信息，随着用户数量的增加，后端服务负担会变重，而且这样的认证机制要求用户请求的是同一台服务器，后端服务无法实现分布式布局，抑制了负载均衡的能力。使用 token 后后端服务就不再需要在内存中记录用户的登录状态了，给平台扩展提供了可能。Json Web Token 由三段信息组成：头部(header)记录 token 采用的加密算法，载荷(payload)存放用于辨别用户的信息、签发时间、过期时间，这部分内容是用户和后端服务都可以解析的，签证(signature)是后端服务对头部和载荷的加盐哈希，后端服务可以签发和验证，但用户无法伪造。知识产权交易平台在用户登录和邮件认证上都用了 Json Web Token，用户登录后 token 储存在用户浏览器的 cookie 中，且设置了“http-only”，保证其仅可被后端服务获取与修改。

图 26：Json Web Token 的组成

平台前端采用 Vue.js 前端框架开发，实现了前端页面 app 化，在官网提供的 vue-cli 模板的基础上，引入了用于异步数据请求与接收的 axios、用于快速构建页面的 Element-UI、用于构建多页面应用的 vue-router、用于方便样式表 css 书写的 stylus，通过模板的分层级嵌套，实现了前台页面与后台页面的功能区分。当用户打开平台网站时会加载前端页面，前端页面根据业务需求向后端服务发起请求，将请求到的数据在前端展示，如果返回错误的信息，前端页面要对错误进行拦截，并将错误信息提示给用户，在用户填写用户名、邮箱、密码时，前端页面要实现对数据的校验，拦截错误的请求信息，减轻后端服务端压力。

这里对前端使用到的一些第三方库进行使用举例：

axios：用于异步数据的请求与接收，在前端页面与后端服务分离的应用中，数据的获取都要靠 JS 代码异步实现，原生的 JS 获取非同步数据代码操作复杂，且在不同版本浏览器实现也不同，因此诞生了 JQuery 等 JS 库，而 axios 是 Vue.js 请求数据的不二选择，其具有简明的 GET、POST 等请求操作方法，还可以添加拦截器对请求数据或响应数据做拦截。在知识产权交易平台前端页面中，就设置了响应拦截器，对状态码非 200 请求成功的接口进行了拦截，并在页面上显示后端服务返回的错误的信息。

Element-UI：用于快速构建网站的组件库，知识产权交易平台中使用了其提供的消息提示、输入框验证、弹窗等功能，可以不用花费精力在这些功能的具体实现上，在这些组件的基础上进行再创作和风格改造。

3.2 系统功能设计

知识产权交易平台主要包含用户的管理和知识产权的管理版块这两大版块又由许多模块组成，下面详细介绍这些模块：

1. 用户登录模块：登录接口对前端页面传输过来的账号、密码格式进行校验，如果格式正确再在数据库中检索账号，再比对数据库中该账号的密码与请求中的密码，如果正确则生成 token 并回复给前端页面。
2. 用户登出模块：登出接口则给前端返回空的 token，覆盖先前存在的 token，完成退出。
3. 用户注册模块：用户注册接口对前端页面传输过来的用户名、邮箱和密码格式进行校验，如果格式正确再在数据库中查找请求的用户名或邮箱是否已经存有，如果都不存在则将用户数据写入数据库，生成 token 返回给前端，并向邮箱发送验证邮件。
4. 获取用户状态模块：获取用户状态模块不需要传入参数，后端服务会根据请求头中的 token 返回当前是否已经登录、登录的用户名和是否管理员。
5. 获取用户资料模块：获取用户资料模块不需要传入参数，后端服务会返回当前用户的资料信息，包括用户的用户 ID、用户名、邮箱、电话、QQ 号码、微信号、邮箱是否通过验证、是否接收通知邮件、账号是否被禁用、账户创建时间。
6. 用户资料修改模块：用户资料修改接口需要传入用户名、电话、QQ 号码、微信号、是否接收通知邮件参数，并再用户名校验无误后存入数据库。

7. 获取验证邮件模块：获取验证邮件接口不需要传入参数，后端服务会检查当前用户是否未验证邮箱，如果未验证则发送验证邮件，邮件链接内的 token 会通过 Json Web Token 生成，载荷当中包含用户 ID 和待验证邮箱，安全性高。
8. 找回密码模块：找回密码分为三个接口。发送重置密码邮件接口传入用户名或邮件名，如果账号存在则发送邮件；检查 token 是否合法接口需传入 token 参数，如果 token 有效返回正确消息；重置密码接口需传入 token 和新密码，如果 token 有效且密码格式正确则修改数据库中密码。
9. 修改邮箱模块：修改邮箱模块分为两个接口。发送验证邮件接口传入旧密码与新邮箱，在旧密码正确且新邮箱不存在于数据库的情况下发送验证邮件至新邮箱，邮件链接内的 token 通过 Json Web Token 生成，载荷当中包含用户 ID 与待验证邮箱；检验 token 接口传入 token 参数，如果 token 有效则修改用户邮箱为载荷中的新邮箱。
10. 修改密码模块：密码修改接口需要传入旧密码与新密码，如果旧密码正确则将新密码写入数据库。
11. 获取所有用户信息模块：检查用户是否有管理员权限，如果有管理员权限将所有用户信息进行返回。
12. 产权发布、修改模块：注册用户可以在用户中心发布自己的产权信息，也可以对已经发布的产权信息进行再编辑。
13. 获取产权列表模块：游客可以通过网站首页获取已经发布的产权信息，注册用户可以查看所有自己发布的知识产权，管理员可以获取到平台所有的知识信息。
14. 获取产权详情模块：游客可以查看已经发布的产权信息详情，注册用户可以查阅个人发布的产权信息详情，管理员可以查看所有产权信息详情。
15. 意向发布模块：注册用户可以对发布的未交易的知识产权发送自己的交易意向，产权所有者就可以接收到意向信息。
16. 获取意向模块：注册用户可以在个人中心查看自己发布的意向信息和查看接收到的产权信息，管理员则可以对平台上所有的产权信息进行查看。

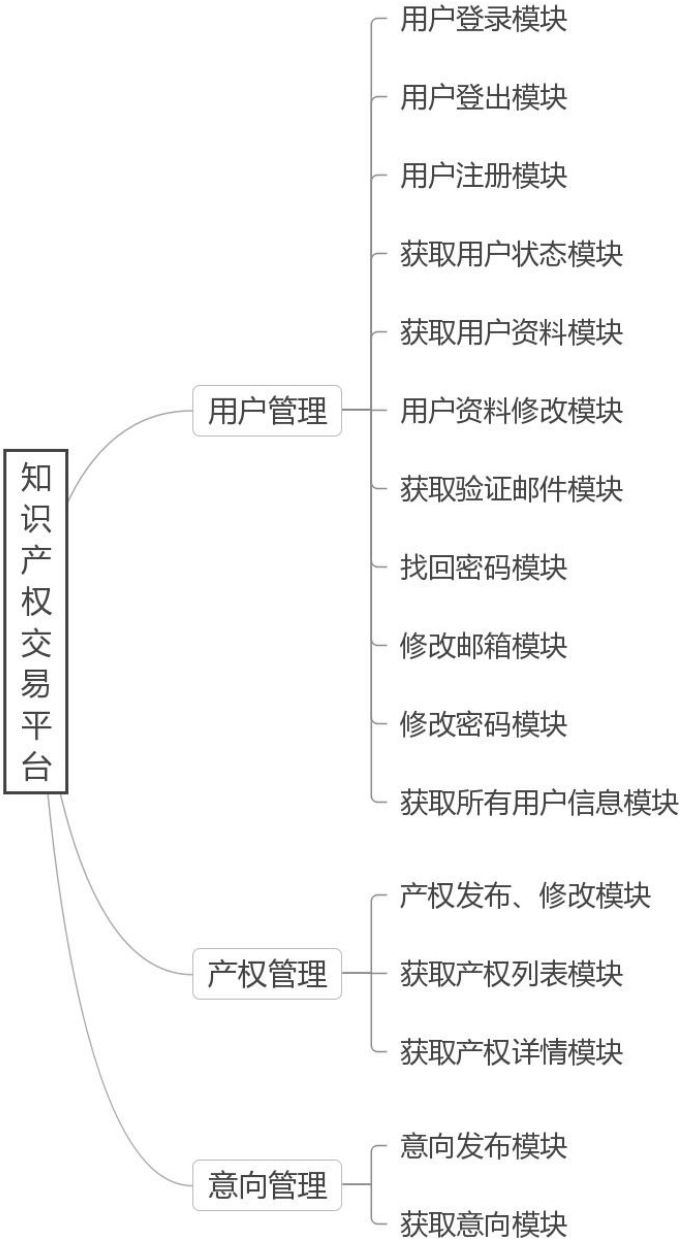

图 27：系统模块结构图

3.3 系统数据库设计

知识产权交易平台设计得实体有：用户 User 实体和知识产权 Property 集合。

用户 User 集合：记录用户的用户名、邮箱、密码、电话、QQ 号码、微信号码、是否接收提醒邮件、是否管理员、是否已验证邮箱、是否被禁用、是否被删除、创建用户时间。

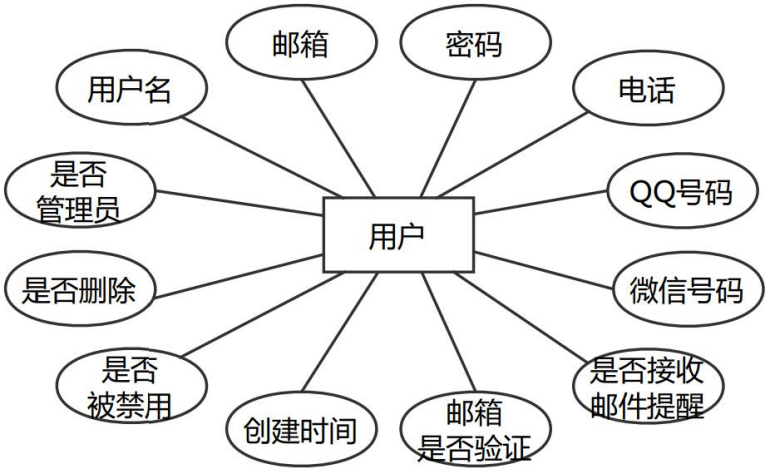

图 28：用户实体图

产权 Property 集合：记录知识产权的创建者用户 ID、产权名、摘要、详情、是否发布、是否售出、创建时间、修改时间、是否被禁用、是否删除。

图 29：产权实体图

意向 Want 集合：记录知识产权 ID、意向者的用户 ID、产权所有者的用户 ID、留言、留言时间、是否删除。

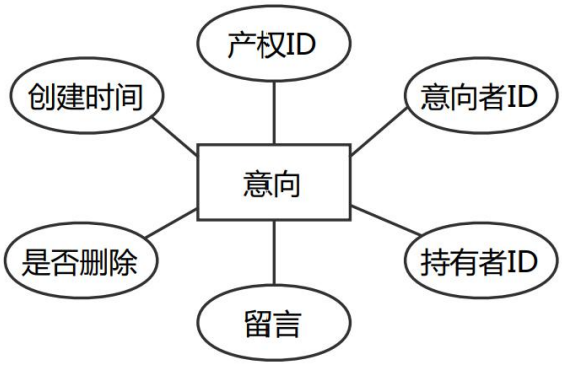

图 30：意向实体图

4. 详细设计

4.1 视图层设计

平台采用简明轻快的设计风格，使用媒体查询，页面支持自适应。首页由导航栏和内容这两成分组成；用户中心由导航栏，功能菜单和功能区组成。其中，导航栏使用 fixed 位置参数，固定于浏览器页面顶端，并限制了内容的宽度为 1000 像素，避免内容过于分散。

图 31：用户中心设计图

用户中心的菜单通过 js 代码监听浏览器的“document.documentElement.clientWidth”，实现菜单的自动收缩与展开，当浏览器页面宽度小于 1000px 时收缩，节省页面空间，当页面宽度大于 1000px 时展开。

图 32：用户中心菜单收缩(左)与展开(右)对比

平台中有多处功能需要用户输入信息，在用户输入完成后，若检测到用户输入不正确的格式，会提醒用户修改，在用户输入正确后会显示绿色的提示。

图 33：输入框校验的提醒效果

4.2 控制层设计

4.2.1 后端中间件设计

平台虽然功能模块多，但是有一些请求的功能是有类似的或有做一些相同操作的，针对这些类似操作，后台系统中开发了一些中间件，下面对这些中间件进行介绍：

checkToken：用于对需要用户权限的操作进行权限的检测，这个中间件会先对请求中的 token 进行检查，如果不存在 token 则返回错误信息，若包含 token 再接着对 token 进行合法性验证，token 签名错误或过期都会返回错误，在确认 token 合法后会根据 token 中的 user_id 在数据库中进行检索，如果用户不存在或用户已被删除都会返回错误，如果找到用户，会将用户信息放入请求体中，避免之后的操作需要再从数据库检索用户信息。只有请求正确，程序才会进入下一函数。

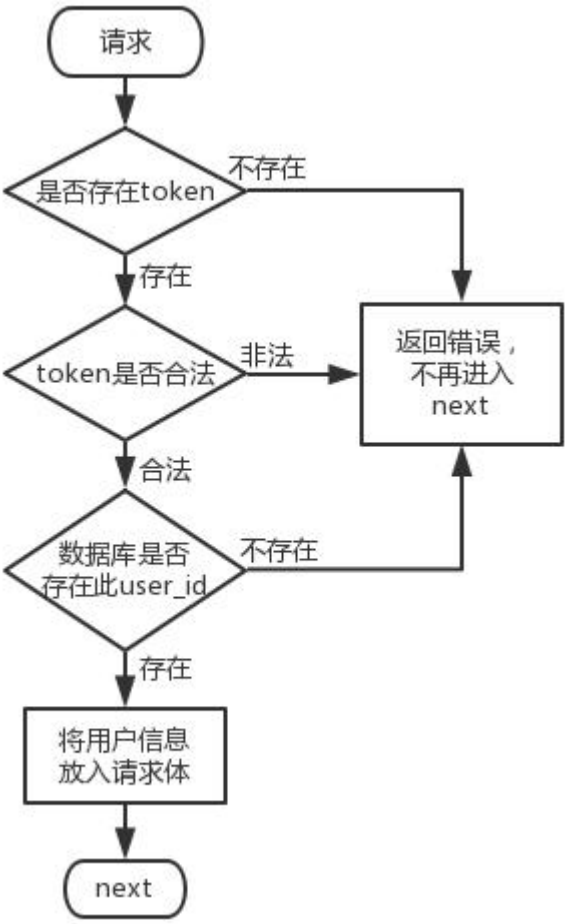

图 34：checkToken 中间件流程图

checkEmail：对用户是否已验证邮箱进行检测，出于安全考虑，未经邮箱验证的用户无法进行知识产权发布等更多功能。因为检测邮箱是否登录建立在用户已经登录的基础上，中间件直接从 request 请求体中拿出 checkToken 放入的用户信息，检查邮件是否经过验证，未验证返回错误信息，已验证再传入下一函数。

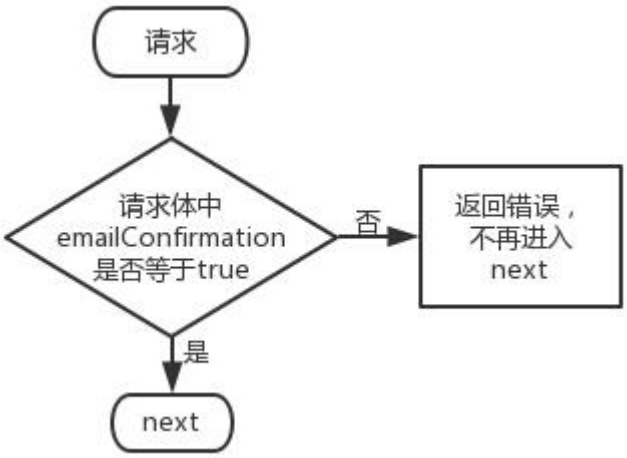

图 35：checkEmail 中间件流程图

checkAdmin：对用户是否是管理员进行检测。检测是否管理员同样建立在用户已经登录的基础上，这里同样对请求体中的用户信息里的‘isAdmin’进行检查。

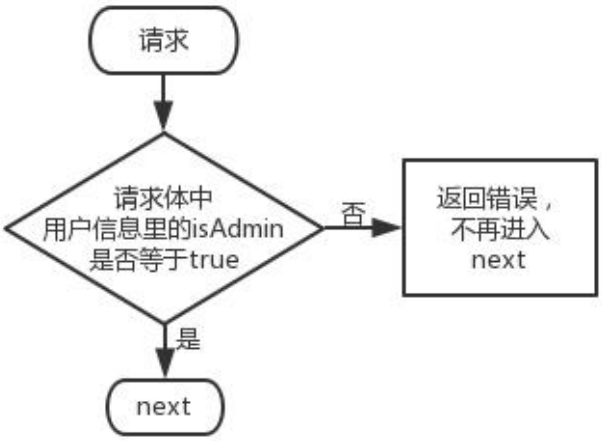

图 36：checkAdmin 中间件流程图

4.2.2 前端消息拦截

知识产权交易平台采用了前端界面与后端服务分离的设计方式，错误信息不再直接在页面中显示，前端页面在请求 API 接口时遇到的错误会通过页面消息的方式提示用户。为此，平台针对多种可能出现的错误情况，统一设计了消息的提醒方式：

Code	消息提示形式
1	请求成功，不显示消息提醒
2	请求成功，将‘msg’以文本的形式显示在绿色 success 消息提醒中
3	请求成功，将‘msg’以 HTML 的形式显示在绿色 success 消息提醒中
4	将‘msg’以文本的形式显示在黄色 warning 消息提醒中
5	将‘msg’以 HTML 的形式显示在黄色消 warning 息提醒中
6	请求错误，将‘msg’以文本的形式显示在红色 error 消息提醒中
7	请求错误，将‘msg’以 HTML 的形式显示在红色 error 消息提醒中
8	将‘msg’以文本的形式显示在灰色消息提醒中
9	请求错误，将‘msg’以文本的形式显示在红色 error 消息提醒中，并跳转至首页

表 1：前后端消息码含义对照表

前端页面根据后台响应中的 code 响应码，在页面上方醒目位置显示消息提醒，借助前端框架中引入的 axios 异步请求模块提供的拦截器功能，在拦截器中对每一条响应信息进行预处理，按照对照表的设计进行对应的弹窗。其中，在一些页面访问后即刻加载数据且成功的请求采用响应码 1，不提示用户；大多数消息采用文本的形式展示，HTML 消息用于在消息中加入超链接指引用户前往其它页面；黄色的 warning 消息目前用于提醒用户验证邮箱；灰色消息用于获取用于获取用户状态失败的情况；响应码 9 用于用户权限不足的

异常情况，前端页面接收到此响应码会跳转至首页。在平台以后的开发过程中，也会对消息状态码表不断完善。

图 37：消息提醒效果图

4.2.3 用户登录模块

平台共设有三种用户权限：平台游客、平台注册用户和平台管理员，他们共用同一个数据库存储，注册用户拥有游客的功能，管理员拥有注册用户和游客的功能，其中注册用户和管理员需要登录，而区分注册用户与管理员的权限靠的是数据库中的“isAdmin”字段。因为，用户权限的控制是由后台控制，所以注册用户和管理员可以共用同一个登录系统，后端服务通过识别获取到的 token 来确认用户。

接口	/login	方法	POST
请求参数	account：用户名或邮箱	响应参数	code：响应码
	password：密码		msg：消息

表 2：用户登录接口请求、响应参数

平台对用户名、账号、密码做了格式上的要求，在平台的前后端都用了正则表达式的校验方法，保证了用户名、邮箱和密码格式的一致性。

参数	要求	正则表达式
用户名	至少 4 个字符，只能由汉字、英文、数字和下划线构成，不能全由数字构成	^(?![0-9]+$)(?!_)(?!.*?_$)[a-zA-Z0-9_\u4e00-\u9fa5]{4,20}$
邮箱	——	^[a-z0-9]+([._\\-]*[a-z0-9])*@([a-z0-9]+[-a-z0-9]*[a-z0-9]+.){1,63}[a-z0-9]+$
密码	至少 8 个字符、由字母和数字组成、必须包含字母和数字组成	(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,20}$

表 3：用户名、邮箱和密码格式要求

当用户在登录弹窗中完成平台注册账号和密码的输入后，前端页面会对账号和密码进行格式验证，其中账号可以是用户名或邮箱，如果账号或密码格式错误，前端页面会在输入框下方用红色字体提醒用户更正，在格式输入正确之前用户无法提交登录。在用户输入正确格式的账号和密码并点击登录后，页面会以账号和密码为参数向后端服务的‘/login’接口发送登录的 POST 请求。

后端服务在收到请求后首先对平台账号和密码格式利用正则表达式进行校验，如果账号或密码格式错误就返回 error 信息。在确认账号密码格式正确后，对账号进行判断，如果账号中包含“@”，则认为账号为邮箱，否则为用户名，接着按照查找邮箱和用户名的方法分别在数据库中查找用户，如果用户不存在则返回错误。在找到用户之后取出用户信息，将数据库中存储的密码与请求中的密码交给 Bcrypt 函数进行比对，如果错误返回错误，正确则使用 Json Web Token 根据用户的 user_id 生成 token，再将 token 作为响应头与成功信息一起返回给用户。因为 token 设置了 http-only，前端页面不会获取到 token 信息，token 会由浏览器存储，并在每次请求时自动将 token 附加在请求头中。

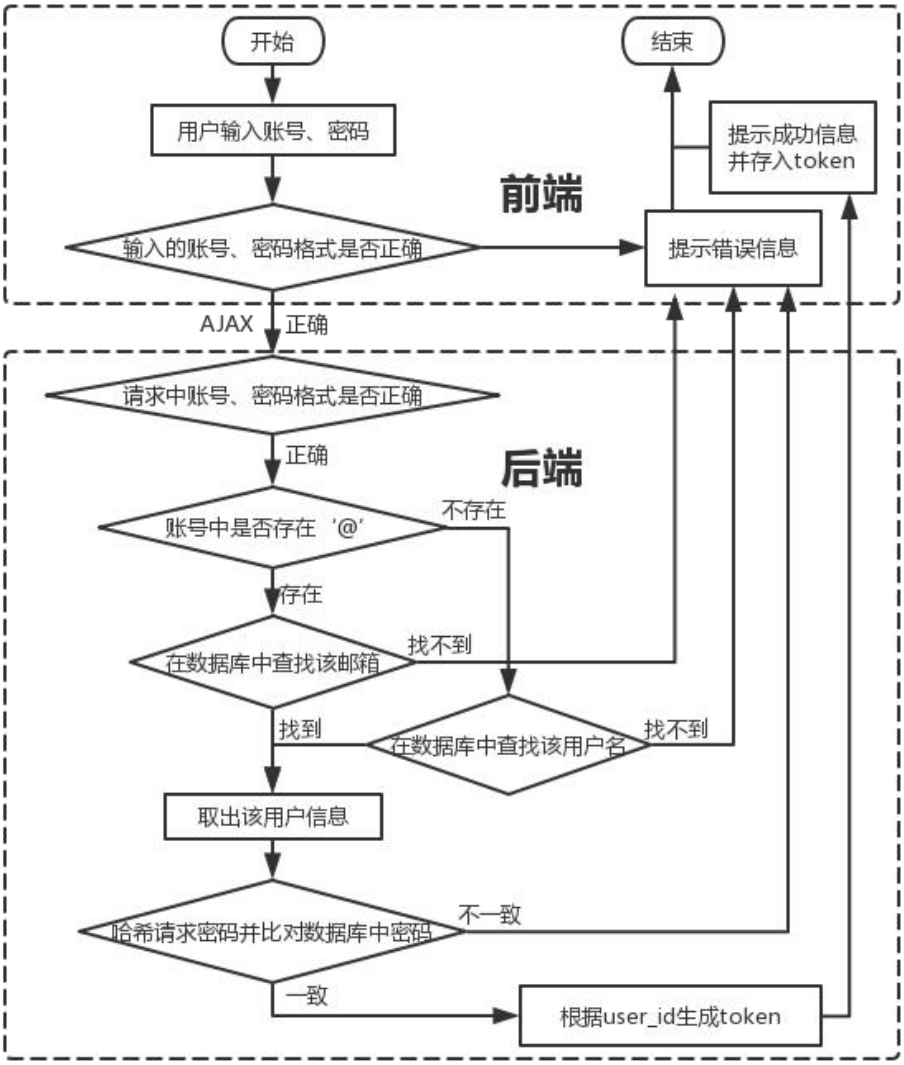

图 38：用户登录流程图

4.2.4 用户登出模块

平台通过在 cookies 中存储 token 来保存用户的登录状态，所以用户退出登录只需要清除 cookies 中的 token 即可，由于出于安全性考虑，平台在用户存储 token 时加了 http-only 参数，因此，用户的退出登录必需由后端响应空的 token 进行清除。

接口	/logout	方法	POST
请求参数		响应参数	code：响应码
			msg：消息

表 4：用户登出接口请求、响应参数

当用户点击退出登录按钮时，前端页面会向后台的‘/logout’接口发送无参数的 POST 请求，后端服务会返回带有空 token 的响应头和请求成功的信息。

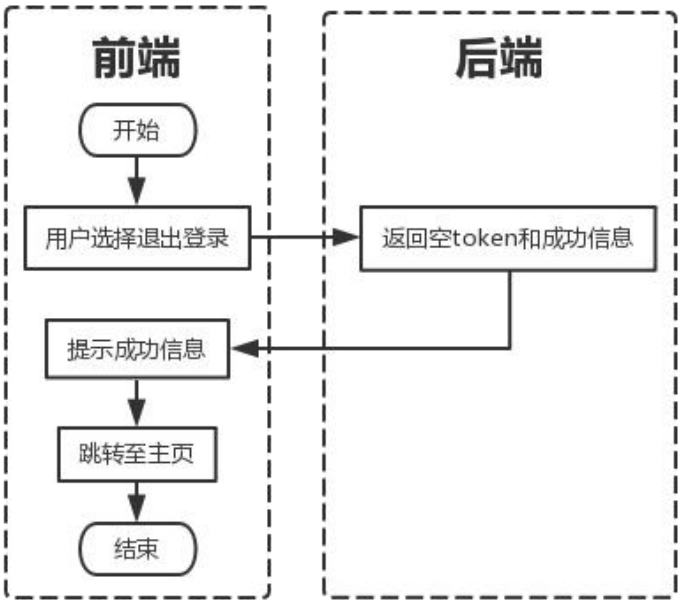

图 39：用户登出流程图

4.2.5 用户注册模块

用户在注册平台账号时要在注册界面中填入用户名、邮箱和密码，前台页面同样会先使用正则表达式对它们分别进行校验，在格式正确后用户点击注册，网页会将这些信息通过“/register”接口使用 POST 方法发送给后端服务。

接口	/register	方法	POST
请求参数	username：用户名	响应参数	code：响应码
	email：邮箱		msg：消息
	password：密码		

表 5：用户注册接口请求、响应参数

而后端服务会再对上述三者的格式进行校验，只有在格式正确的情况下，后端服务会接着在数据库中依次查找是否已存在请求中的用户名和邮箱，如果已存在会返回信息提示用户修改，如果不存有则会将这些信息录入数据库，并向用户当前注册的邮箱发送带有 token 的验证邮件，最后将存储数据库后生成的 user_id 进行 token 签发，以响应头的形式和成功的响应内容一同返回给用户，此时用户自动跳转至用户中心，完成注册操作。

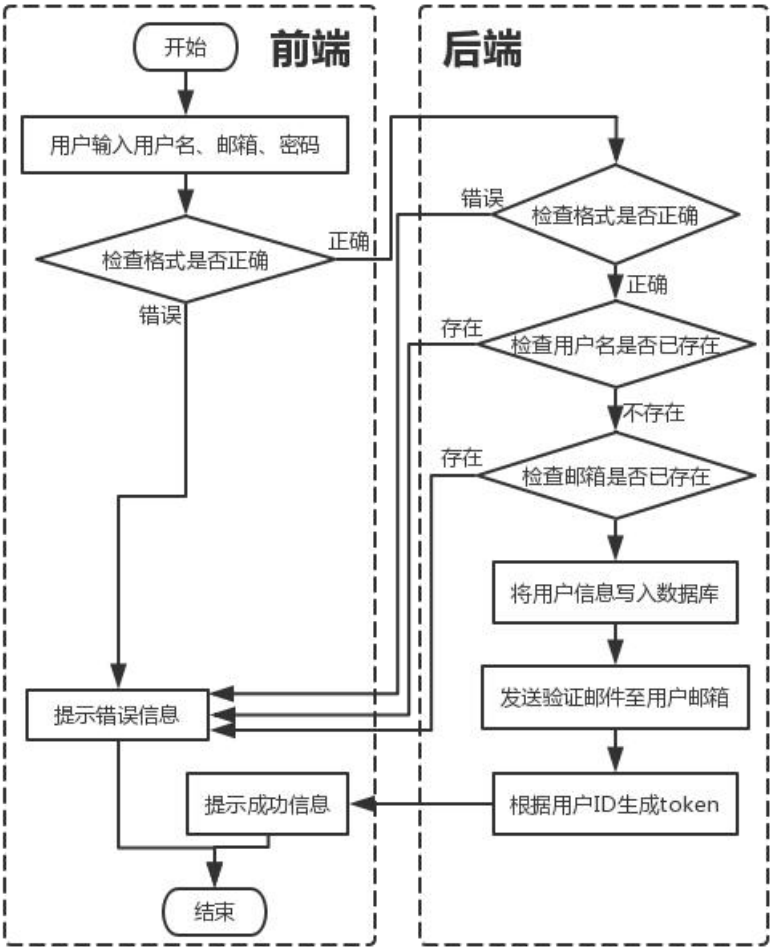

图 40：用户注册流程图

4.2.6 获取用户状态模块

在前端页面与后端服务分离的状态下，前端需要通过向后端服务请求当前用户的权限信息，才能做相应的界面展示，比如平台右上角的个人中心是否显示，个人中心中管理员功能是否展示，都要依赖对用户状态进行查询。

接口	/userStatus	方法	GET
响应参数	code：响应码	响应参数	isAdmin：是否管理员
	msg：消息		username：用户名
	isLogin：是否登录		

表 6：获取用户状态响应参数

在用户加载页面时，前端页面会同时向“/userStatus”接口发起用户状态查询的 GET 请求，平台在接收到请求后，会先判断请求是否带有 token 信息，不存在则表示未登录，若存在还要继续对 token 的合法性做检查，首先检查 token 是否由平台的密匙签发，再检查 token 是否在有效期内，在确认 token 合法性后根据 token 中的 user_id 到数据库中检索该用户，如果用户不存在或用户已被删除都返回未登录，如果用户存在且正常则返回用户的用户名、是否登录、是否管理员。

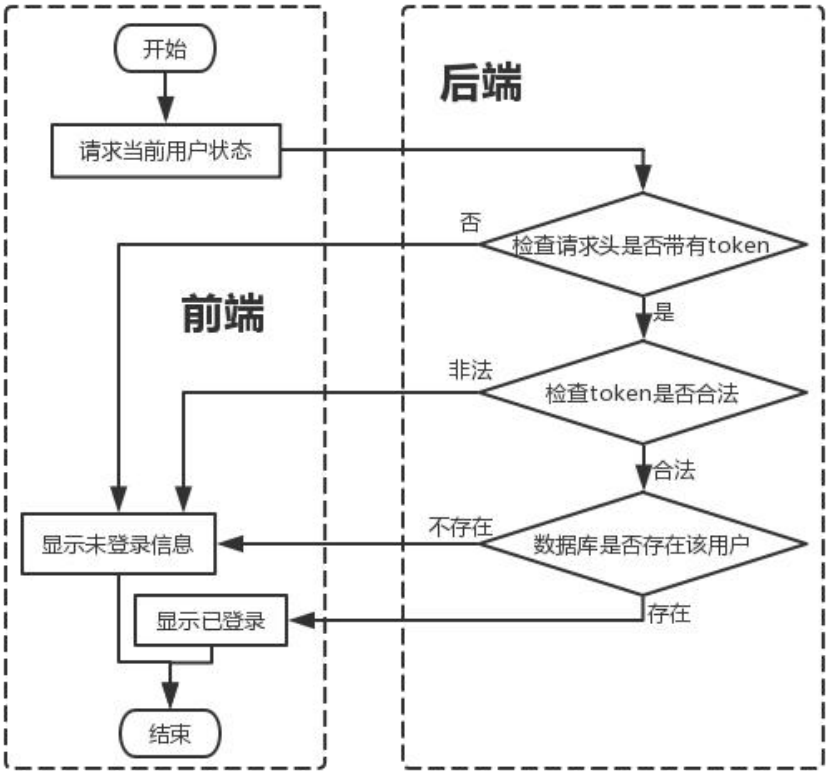

图 41：获取用户状态流程图

4.2.7 获取用户资料模块

注册用户打开个人中心资料页时，前端页面会自动向“/userInfo”接口发起资料查询的 GET 请求。

接口	/userInfo	方法	GET
响应参数	code：响应码	响应参数	qqNumber：QQ 号码
	msg：消息		wechat：微信号
	user_id：用户 ID		message：是否接收提醒邮件
	username：用户名		emailConfirmation：邮件是否验证
	email：邮箱		isDisabled：账号是否被禁用
	telephone：电话		createTime：用户创建时间

表 7：获取用户资料接口响应参数

后端服务在接到请求后会先将请求交给 checkToken 中间件处理，之后只需直接将 checkToken 获取到的用户信息返回给前端页面，前端页面再将信息显示到页面中来。

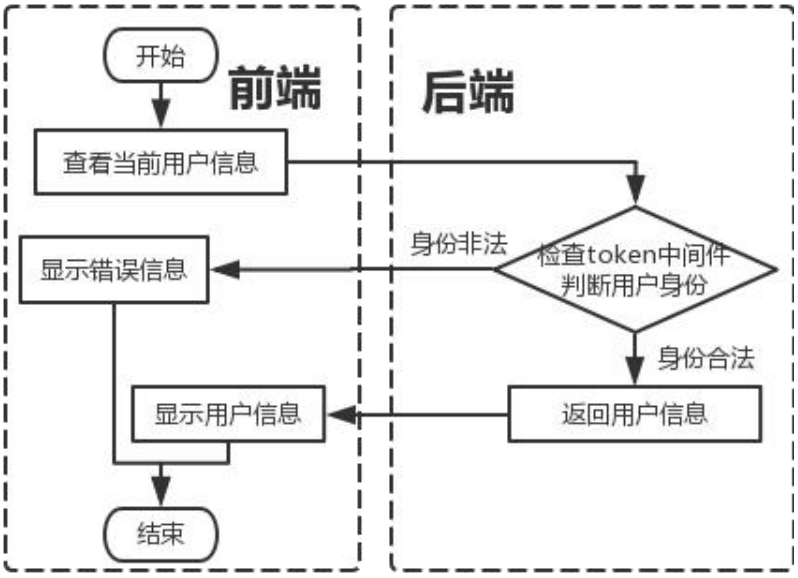

图 42：获取用户资料模块

4.2.8 用户资料修改模块

平台在用户资料查看页面提供了修改部分用户资料的功能，用户可以在修改后进行提交，其中用户名同样做了格式的校验，确认用户输入的用户名合法后可以进行提交，页面会向后端服务的“/userInfo”接口发送 POST 请求。基于安全上的考虑，后端服务不允许被禁用的账户和未经过邮件验证的账户修改个人资料。

接口	/userInfo	方法	POST
请求参数	username：用户名	请求参数	createTime：注册时间
	telephone：电话		emailConfirmation：邮件是否验证
	qqNumber：QQ 号码		isAdmin：是否管理员
	wechat：微信号		isDisabled：是否禁用
	message：是否接收提醒邮件	响应参数	code：响应码
	email：邮箱		msg：消息
备注	email、createTime、emailConfirmation、isAdmin、isDisabled 仅管理员可用		

表 8：用户资料修改接口请求、响应参数

请求到达后端服务后首先会通过 checkToken 检查用户合法性，再经过，会对用户名格式进行检查，在确认合法性后依次由 checkEmail 中间件和 checkDisabled 中间件检查，在检查都无误后对用户名格式做检查，如果用户名格式也正确则将用户更新后的资料写入数据库中，并返回成功信息。浏览器再收到成功信息后再次更新页面信息，做到与后端服务数据一致。

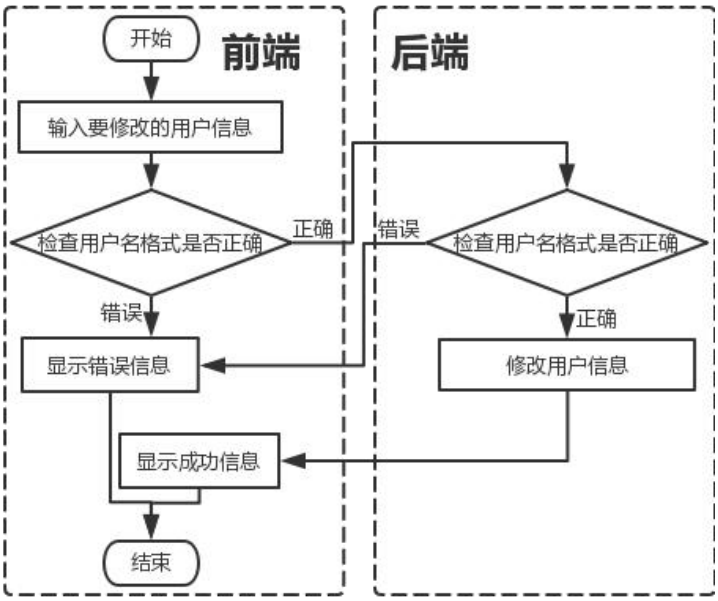

图 43：用户资料修改流程图

4.2.9 获取验证邮件模块

平台为避免因邮件系统异常而导致用户不能收到邮件，在资料中心提供了重新接收验证邮件的功能按钮，用户在按下该按钮后，页面会向后台的“/resendEmail”接口发送 GET 请求。

接口	/resendEmail	方法	GET
请求参数		响应参数	code：响应码
			msg：消息

表 9：获取验证邮件接口请求、响应参数

后端服务在接收到该请求后先将其交由 checkToken 中间件检查用户合法性，如果用户合法再检查用户是否已通过邮件认证,如果通过则返回错误信息,为通过则通过 Json Web Token 将 user_id 和邮件地址打包生成 token，放入邮件地址的好处是方便之后与修改邮箱验证新邮箱功能的功能共用，再将 token 与前端页面地址进行拼接成用于验证邮箱的 URL，通过 nodemailer 模块发送带有该 URL 的验证邮件至用户邮箱。

接口	/checkEmailToken	方法	POST
请求参数	token	响应参数	code：响应码
			msg：消息

表 10：验证修改邮件 token 接口请求、响应参数

用户在打开邮件 URL 后会跳转到带有参数的前端地址，前端页面根据地址中的参数向后端服务的“checkEmailToken”接口通过 POST 方法查询 token 是否合法，在确认 token 合法后修改用户邮箱为 token 中的邮箱，并返回成功信息给前端页面。

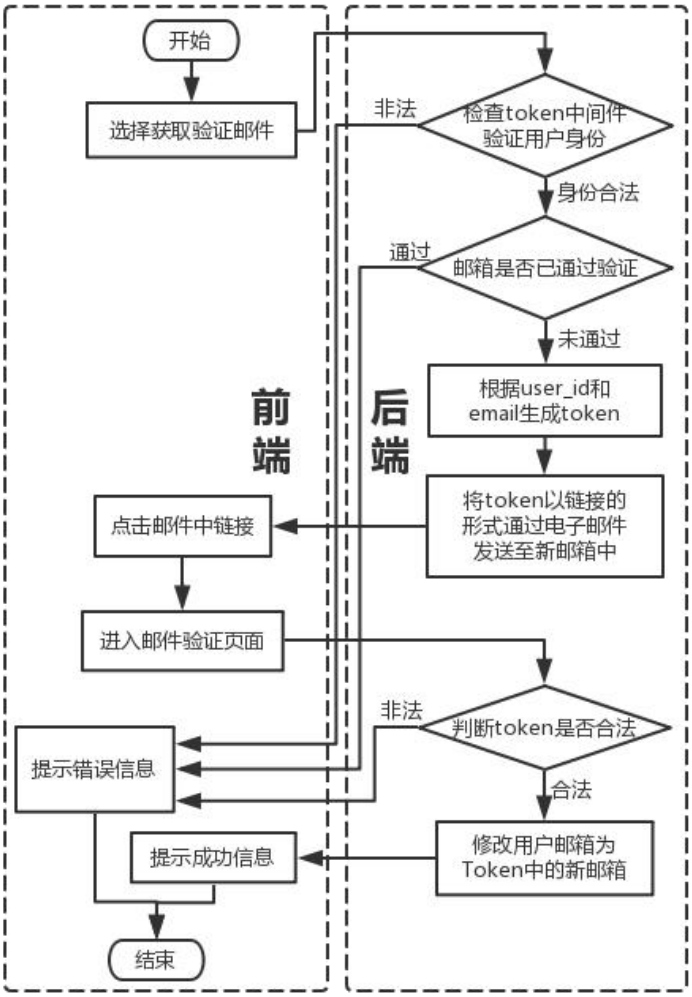

图 44：获取验证邮件流程图

4.2.10 找回密码模块

考虑到用户长时间不登录可能会忘记用户密码，平台提供找回密码功能，找回密码时用户可以在找回密码输入框输入用户名或邮箱并点击找回密码，在页面校验账号格式无误后会向后端服务的“/forgetPass”接口发送带有账号参数的 POST 请求。

接口	/forgetPass	方法	POST
请求参数	account：用户名或邮箱	响应参数	code：响应码
			msg：消息

表 11：找回密码接口一的请求、响应参数

后端服务在接收到该找回密码的请求后，同样会对账号格式进行校验，在账号符合用户名格式或邮箱格式后进入下一步。后端服务会判断账号中是否存在“@”，如果存在则判断其为邮箱，不存在则判断其为用户名，再根据情况在数据库中检索该用户，如果找到该用户则根据用户的 user_id 生成 token，将 token 与找回密码的前端页面链接进行拼接，将此拼接后的 URL 以邮件的形式发送给用户。用户在收到邮件后点击链接即可进入找回密

码页面，页面会将链接中的 token 以参数的形式通过 POST 方法发送至后端服务的“/resetPass1”接口。

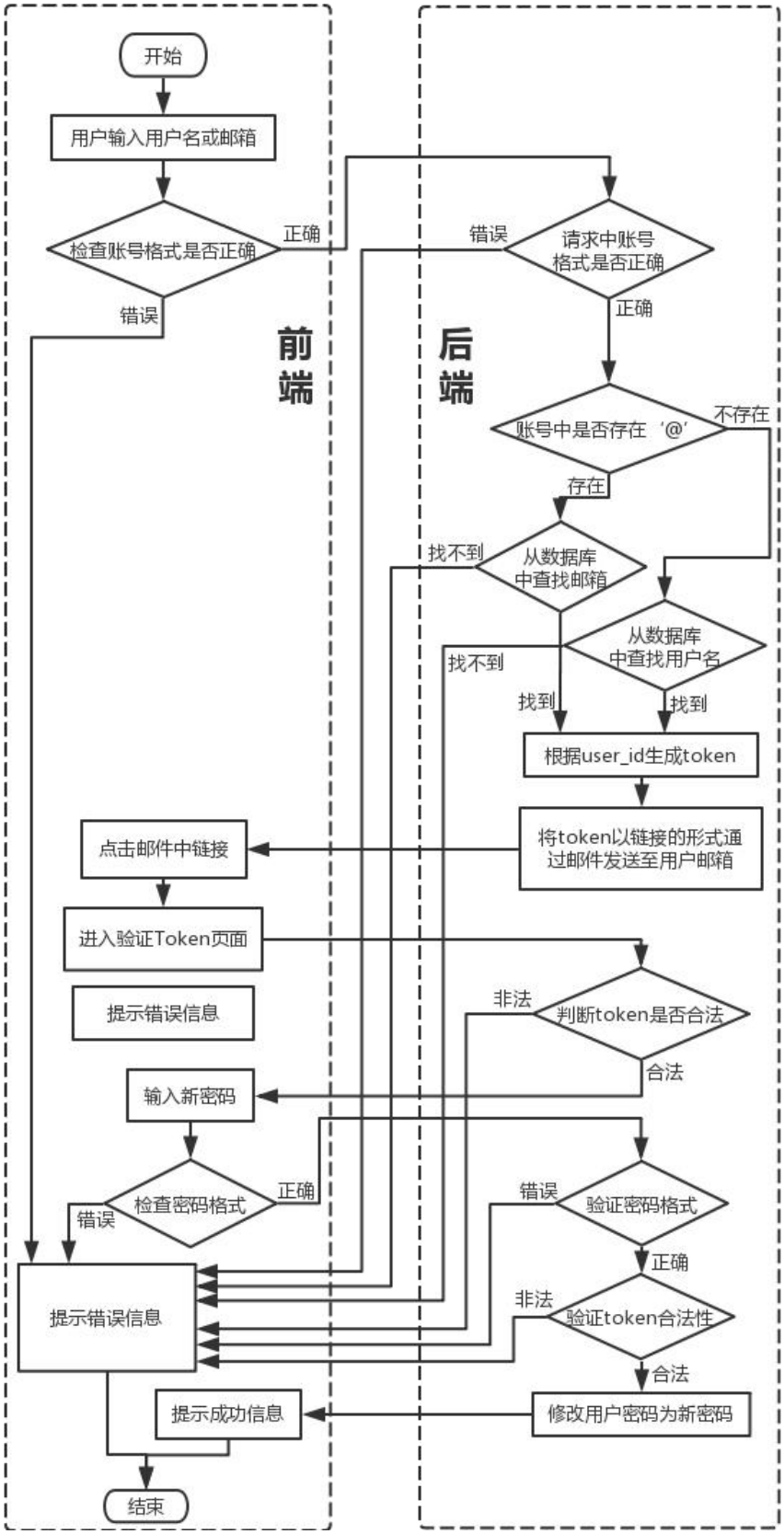

图 45：重置密码流程图

接口	/resetPass1	方法	POST
请求参数	token	响应参数	code：响应码
			msg：消息

表 12：找回密码接口二的请求、响应参数

后端服务在接收到该用于重置密码的 token 后会对该 token 的合法性进行检验，包括 signature 是否正确、token 是否到期，用户是否存在，在确定 token 合法后后端服务会返回成功信息，这时前端页面会显示新密码设置界面，用户可以进行新密码的输入。用户输入新密码后前端页面同样会对密码格式进行校验，在校验无误后，将 token 和新密码一同以参数形式，通过 POST 请求发送至“/resetPass2”接口。

接口	/resetPass2	方法	POST
请求参数	token	响应参数	code：响应码
	password：新密码		msg：消息

表 13：找回密码接口三的请求、响应参数

后端服务会对密码格式进行校验，并再一次检验用于重置密码的 token 的合法性，如果均无问题再将密码使用 Bcrypt 进行哈希，之后存入数据库并给页面返回成功信息，前端页面会提示用户密码已成功修改。

4.2.11 修改邮箱模块

用户可以在用户的安全中心进行邮箱的修改，或在邮箱无法收到验证邮件的情况下修改。当用户输入密码和新邮箱后选择确定，页面会在密码和新邮箱格式正确的情况下向后端服务的“/changeEmail”通过 POST 方法发送用户的密码和新邮箱。

接口	/changeEmail	方法	POST
请求参数	password：密码	响应参数	code：响应码
	email：新邮箱		msg：消息

表 14：修改邮箱接口请求、响应参数

后端服务在收到请求后先使用 checkToken 中间件进行合法性检查，在确认请求权限合法后检测密码、邮箱格式是否正确，再检查新设邮箱是否与旧邮箱相同，如果密码、邮箱格式正确且新旧邮箱不一样，使用 Bcrypt 对请求中的密码和数据库中的密码进行比对，在确认用户密码正确后，为 user_id 和新邮箱签发 token，将 token 与验证邮箱的前端网址进行拼接，拼接后以邮件的方式发送至用户新邮箱。接下来的操作就与前面验证邮箱的方法相同，共用同样的功能。

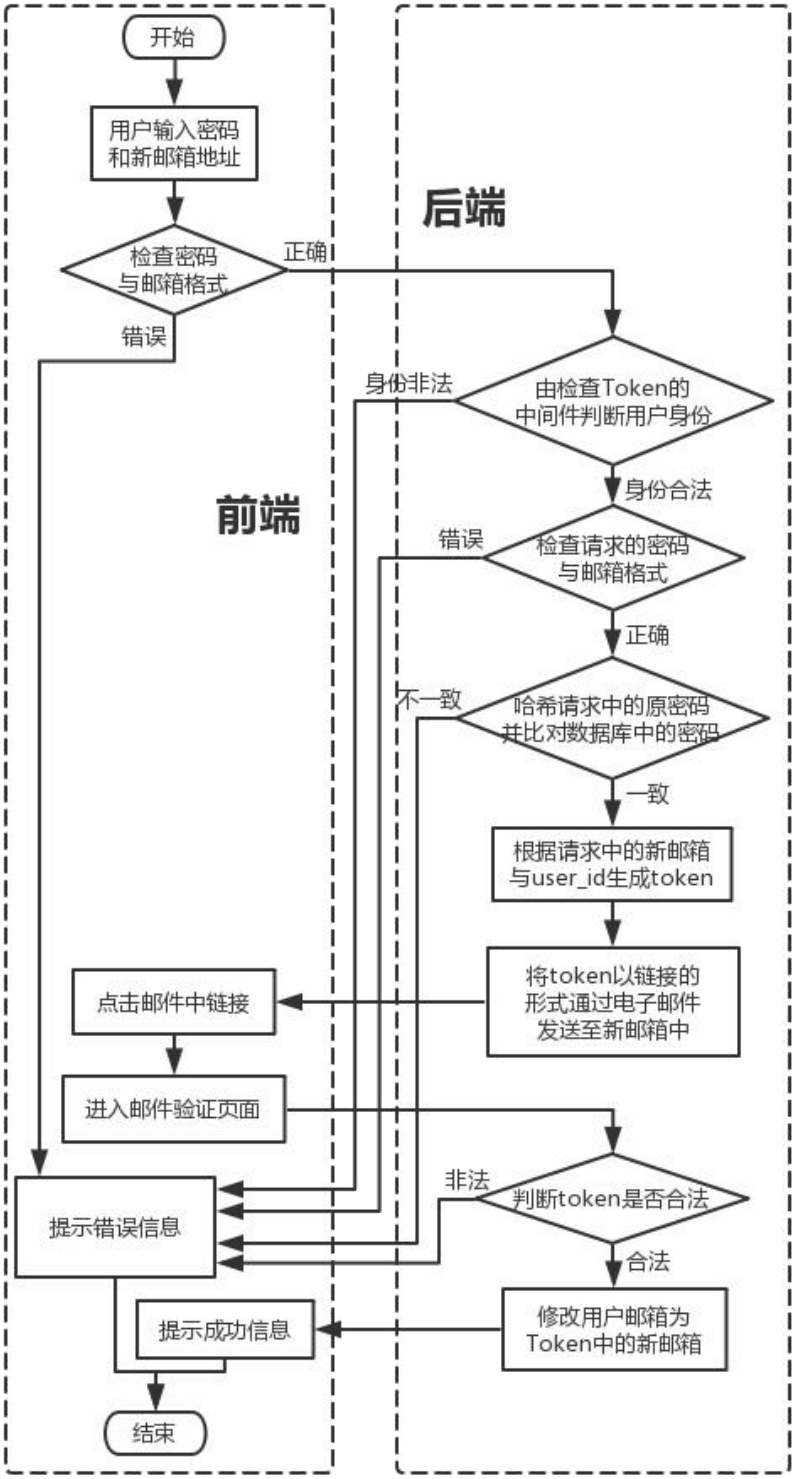
开始
用户输入密码
和新邮箱地址
检查密码
与邮箱格式
正确
错误
前端
后端
身份非法
由检查Token的
中间件判断用户身份
身份合法
错误
检查请求的密码
与邮箱格式
正确
不一致
哈希请求中的原密码
并比对数据库中的密码
一致
根据请求中的新邮箱
与user_id生成token
将token以链接的
形式通过电子邮件
发送至新邮箱中
点击邮件中链接
进入邮件验证页面
非法
判断token是否合法
合法
提示错误信息
提示成功信息
修改用户邮箱为
Token中的新邮箱
结束

图 46：修改邮箱流程图

4.2.12 修改密码模块

用户中心还提供密码修改功能，用户在填写旧密码和新密码后选择确定，页面会在检查新旧密码格式无误后，将新旧密码通过 POST 请求发送至后端服务的“/changePass”接口。

接口	/changePass	方法	POST
请求参数	oldPassword：旧密码 newPassword：新密码	响应参数	code：响应码 msg：消息

表 15：修改密码接口请求、响应参数

后端服务在接收到请求后会先用 checkToken 中间件检查用户权限，之后检查新旧密码格式与新旧密码是否相同，再使用 Bcrypt 对请求中的旧密码与数据库中的用户密码进行一致性比对，在确认用户旧密码后将用户新密码进行加盐哈希保存至数据库中，并返回成功信息，在前端页面中显示出来。

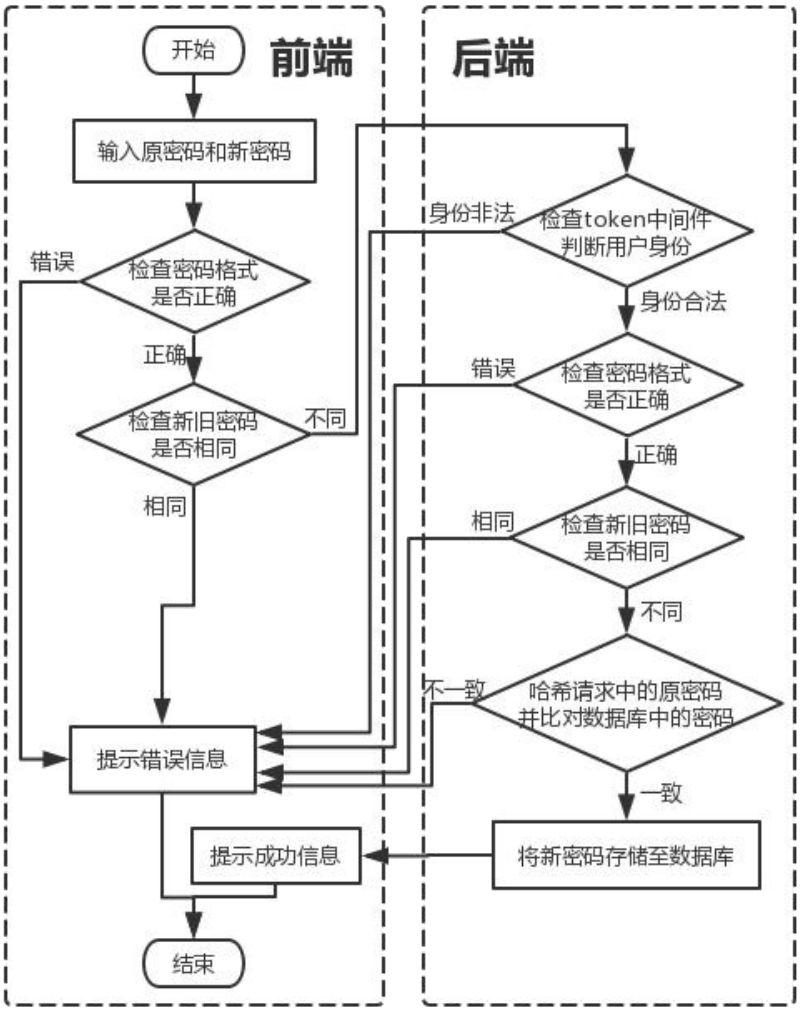

图 47：修改密码流程图

4.2.13 获取所有用户信息模块

管理员拥有查看当前平台中所有注册用户信息的功能，管理员可以在用户中心以表格的形式查看所有用户的信息，当页面打开时，页面会自动向后端服务的“/usersInfo”接口发送 GET 请求。

接口	/usersInfo	方法	GET
响应参数	code：响应码	响应参数	users：用户数据数组
	msg：消息		

表 16：获取所有用户信息接口响应参数

后端服务在接收到该请求后首先会使用 checkToken 中间件检查用户权限，再由 checkAdmin 中间件检查是否具有管理员权限，接着从数据库中取出所有注册用户的数据并使用 JSON 格式返回给前台，前台会将数据以表格的形式展现给管理员用户。

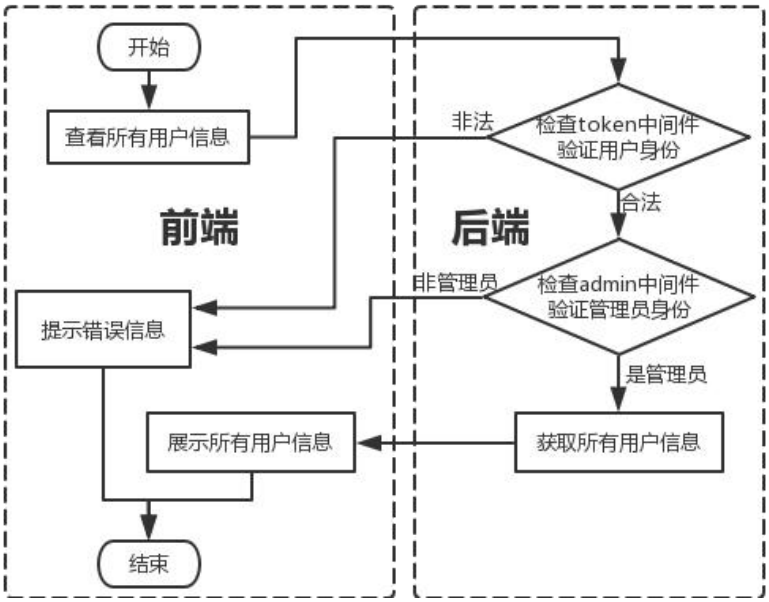

图 48：获取所有用户信息流程图

4.2.14 产权发布、修改模块

注册用户可以在用户中心进行产权的发布和修改，两者使用同一接口，当参数带有property_id时代表编辑产权，不带有代表新建产权信息。

接口	/property	方法	POST
请求参数	_id：产权的编号（可选）	请求参数	isDisabled：是否禁用
	propertyName：产权名		createTime：创建时间
	summary：摘要		editTime：编辑时间
	detail：详情	响应参数	code：响应码
	isPublish：是否发布		msg：消息
	isSelt：是否销售		
备注	isDisabled、createTime、editTime仅管理员可用		

表17：产权发布、修改接口请求、响应参数

后端服务在接收到请求后会先通过checkToken中间件检查用户权限，再根据发布或修改的不同情况对信息存入数据库，并返回成功结果，如果用户具有管理员身份，还可以对是否禁用、创建时间和编辑时间进行修改。

4.2.15 获取产权列表模块

注册用户可以在个人中心中查看已发布的知识产权信息列表，列表中包含产权名、摘要、是否发布、是否售出等信息。

接口	/propertys	方法	GET
响应参数	code：响应码	响应参数	propertys：产权列表
	msg：消息		

表18：用户获取个人产权列表接口响应参数

管理员可以在个人中心查看平台的所有产权信息，前端页面会通过向“/allPropertys”接口发送GET请求获取到参数。

接口	/allPropertys	方法	GET
响应参数	code：响应码	响应参数	propertys：产权列表
	msg：消息		

表19：管理员获取平台所有产权列表接口响应参数

游客可以在网站首页看到注册用户发布的产权信息，当游客打开首页就会向“/indexPropertys”接口发送GET请求获取到首页数据。

接口	/indexPropertys	方法	GET
请求参数	userId：查看某一用户发布的产权（可选）	响应参数	code：响应码
			msg：消息
			propertys：产权列表

表 20：游客获取首页产权列表接口请求、响应参数

后端服务会在检查完用户权限后取出对应的产权信息返回给用户，前端页面会将这些信息以列表的形式展示给用户。

4.2.16 获取产权详情模块

注册用户可以在个人中心对产权进行修改，修改产权前前端页面会通过向“/property”接口发送 GET 请求得到产权的详细信息并将它们填入输入框中以便用户修改。

接口	/property	方法	GET
请求参数	_id：产权编号	响应参数	detail：详情
			isPublish：是否发布
响应参数	code：响应码		isSelt：是否售出
	msg：消息		isDisabled：是否禁用
	propertyName：产权名		createTime：创建时间
	summary：摘要		editTime：编辑时间
备注	isDisabled、createTime、editTime 仅由管理员获得		

表 21：用户获取个人产权详情接口请求、响应参数

游客可以在首页对感兴趣的产权进行详情查看，查看详情会进入产权详情页，页面在获取到数据后会根据响应中的“isSlet”字段来显示“我有意向”按钮。

接口	/indexProperty	方法	GET
请求参数	_id：产权编号	响应参数	detail：详情
			isSelt：是否售出
响应参数	code：响应码		createTime：创建时间
	msg：消息		editTime：编辑时间
	propertyName：产权名		publisher：产权发布者 ID
	summary：摘要		publisherName：发布者用户名

表 22：游客获取产权详情接口请求、响应参数

4.2.17 意向发布模块

注册用户可以在其它用户的产权详情页点击“我有意向”发送意向请求。

接口	/indexWant	方法	POST
请求参数	message：留言	响应参数	code：响应码
	property：产权编号		msg：消息

表 23：意向发布接口请求、响应参数

后端服务会对产权编号、留言内容和产权发布者进行存储，方便之后的检索，同时会向产权发布者发送邮件提醒。

4.2.18 获取意向模块

注册用户可以在个人中心查看自己收到的所有意向信息，也可以就某一产权单独查看收到的意向信息。

接口	/wants	方法	GET
请求参数	_id：查看某一产权接收到的意向（可选）	响应参数	code：响应码
			msg：消息
			wants：意向信息数组

表 24：用户获取收到的意向接口请求、响应参数

注册用户也可以在个人中心查看自己发送过的意向信息。

接口	/needs	方法	GET
响应参数	code：响应码	响应参数	wants：意向信息数组
	msg：消息		

表 25：用户获取发出的意向接口响应参数

管理员可以查看到平台所有的意向信息。

接口	/allWants	方法	GET
请求参数	code：响应码	响应参数	wants：意向信息数组
	msg：消息		

表 26：管理员获取平台所有意向接口响应参数

4.3 模型层设计

模型层采用了 Mongoose 模块后可以直接使用面向对象的方式来操作 MongoDB 而无需拼接 SQL 语句。这里通过简单的代码段演示在处理请求的过程中如何高效快捷地操作数据库：

```
let result = await User
  .findOne({
    username: ctx.request.body.account,
    isDelete: false
  })
  .exec()
  .catch(err => {
    ctx.throw(500, 'find username error');
  });
```

图 49：从数据库中查找用户信息

图 31 展示了从数据库中异步查找用户的方法，Mongoose 使得查找工作变得简单，使用 findOne 方法查找内容，并对异常情况进行处理，接下来就可以通过 result 读到用户数据，但当用户数据不存在时，result 为空，所以执行完代码后要对 result 进行判断。

```
let user = new User({
  username: ctx.request.body.username,
  email: ctx.request.body.email,
  password
});
let result = await user
  .save()
  .catch(err => {
    ctx.throw(500, 'register user error');
  });
```

图 50：往数据库中添加用户信息

图 32 展示了从数据库中异步添加用户的方法，使用 Mongoose 创建的 User 模型新建一个 user，往 user 中添加用户信息，调用 save 方法就可实现用户数据的存储。

```
await User
  .findByIdAndUpdate(ctx.request.user._id, {
    username: ctx.request.body.username
    message: ctx.request.body.message
  })
  .exec()
  .catch(err => {
    ctx.throw(500, 'write user info error');
});
```

图 51：往数据库中更新用户信息

图 33 展示了从数据库中异步修改用户数据的方法，调用 User 模型的 findByIdAndUpdate 方法，以 user_id 和用户信息为参数，就可以对用户数据进行修改。

4.4 数据库设计

数据库的设计按照知识产权平台的用户、产权和意向这三大版块进行分别设计，依次设计了用户表、产权表和意向表，下面对这三个数据表进行展示：

字段名	数据类型	描述
_id	ObjectId	用户 ID
email	String	邮箱
username	String	用户名
password	String	密码
telephone	String	电话
qqNumber	String	QQ 号码
wechat	String	微信
message	Boolean	是否接收邮件消息
isAdmin	Boolean	是否管理员
emailConfirmation	Boolean	邮箱是否已验证
isDisabled	Boolean	是否被禁用
isDelete	Boolean	是否被删除
createTime	Date	创建时间

表 27：用户数据表

字段名	数据类型	描述
_id	ObjectId	产权 ID
propertyName	String	产权名
summary	String	摘要
detail	String	详情
publilsher	ObjectId	产权发布者
isPublisher	Boolean	是否发布
isSelt	Boolean	是否售出
isDisabled	Boolean	是否被禁用
isDelete	Boolean	是否被删除
createTime	Date	创建时间
editTime	Date	编辑时间

表 28：产权数据表

字段名	数据类型	描述
_id	ObjectId	意向 ID
property	ObjectId	产权 ID
wanter	ObjectId	意向者 ID
keeper	ObjectId	产权持有者 ID
message	String	留言内容
isDelete	Boolean	是否被删除
createTime	Date	创建时间

表 29：意向数据表

这三个数据表之间的相互关系可以通过数据库 E-R 图直观了解：

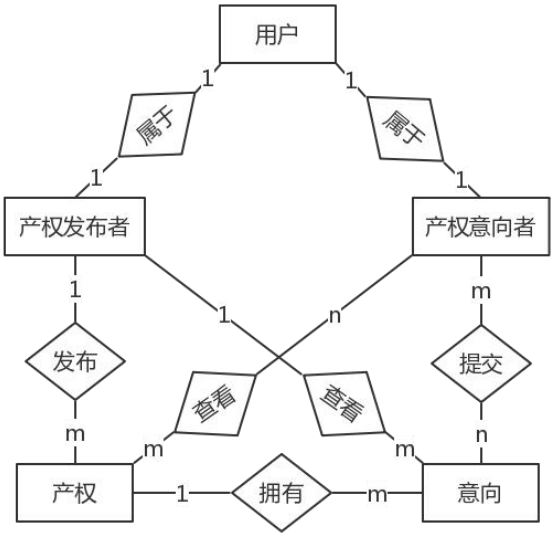

图 52：数据库 E-R 图

5. 系统测试与维护

5.1 白盒测试

团队在结束开发知识产权交易平台的工作后对先前整个开发思路进行了回顾与梳理，按照当初设计的思路，重点针对后端服务进行测试，保证接口的安全性。逐一对 API 接口的安全性进行讨论，画出每个 API 的流程图，讨论所有可能的异常情况，并按照代码为接口设计测试用例，保证测试用例可以完整的跑过程序的每一条代码，跑过每一个判断语句和条件语句，保证每个语句都能被执行，验证语句是否能按照设计的想法正常运行。

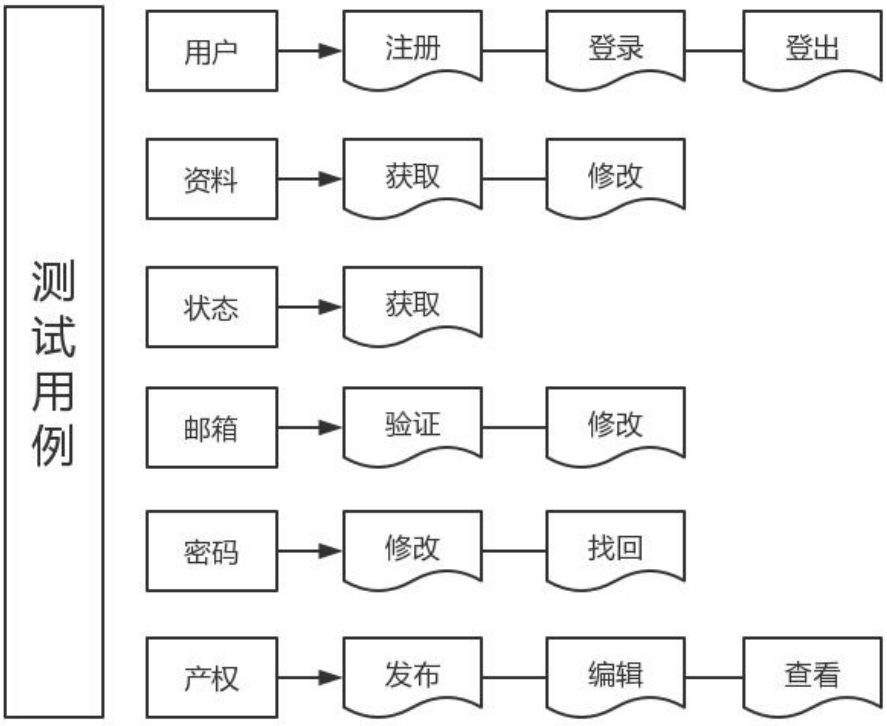

图 53：测试用例

平台按照上方的测试用例进行了测试，通过监视浏览器的控制台和后端服务控制台的输出，均为发现有异常或错误的情况出现，因此交易平台的功能逻辑正确，成功通过白盒测试。

5.2 黑盒测试

在白盒测试后团队又为平台进行了黑盒测试，前端页面能够拦截绝大数的错误操作，比如页面中各处需要输入邮箱、用户名和密码的地方都使用了正则表达式验证，为进一步检测平台的安全性，团队使用 postman 工具向后端服务接口发送不合法的邮箱、用户名或密码数据，检验后端服务能否拦截这些恶意请求。

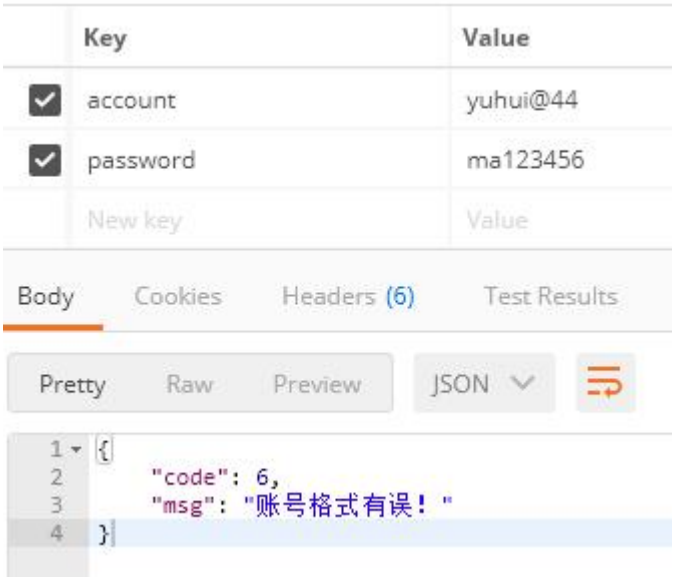

图 54：错误账号无法进行登录

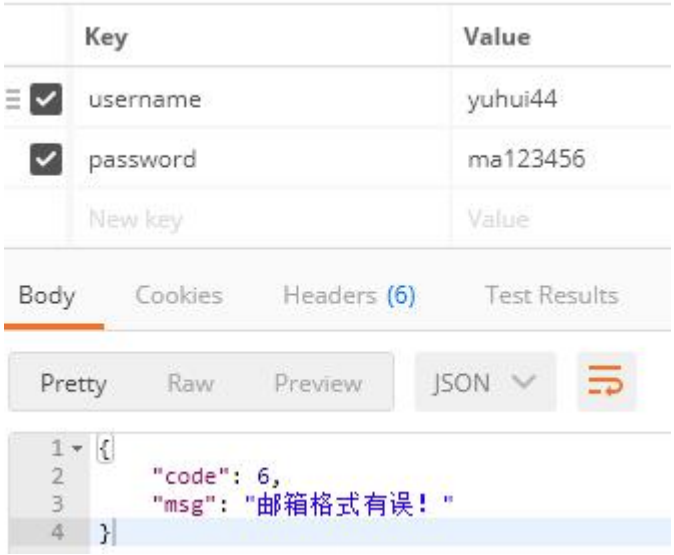

图 55：缺少邮箱参数无法进行注册

在测试时，团队仿照前端发送的请求进行接口请求，但只要参数不符合要求都会被拦截，据此说明网站的前后端在安全性上和可用性上都十分可靠，网站的各个功能都不存在问题。

5.3 系统维护

平台的开发虽然告一段落，但是平台的运营是长期的持续工作，在网站上线后要做到：

1. 及时对网站服务器进行安全的补丁更新，定期进行数据库备份。
2. 根据网站的需求积极开发新功能，通过供给侧改革，为用户提供更多实用的功能，真正助力知识产权的交易。
3. 安排专人对网站进行管理，对违法网站规定的用户进行禁用，对危害国家安全的不良信息进行删除，同时使用第三方服务对网站进行监控，以及时发现网站的异常状况。

6. 结论

此次毕业设计从确定到完成设计历时近一年，在这个过程中，虽然团队成员都有其他工作要忙但也一直没有耽搁知识产权交易平台的设计。在这一过程中我们查阅了有关网站建设的书籍，多次线下交流探讨，起初在平台的设计上考虑得不是很周全，我们在实际开发中逐步对设计进行的修正。在平台的建设中我们也渐渐发现了网站建设的趣味性，我们自己的许多想法可以直接在开发中进行实现，许多疑惑也在实操中得到了答案。

在平台开发过程中团队得到了导师和同学们的帮助和意见建议，在这里给予他们衷心的感谢，同时我们团队也会继续加强自身的知识储备，聆听导师和同学们的意见建议，一同将平台完善好。

参考文献

[1]. 罗琴. 知识产权交易一站式全程服务模式研究[D].中国科学技术大学,2017.

[2]. 邓志云,管怀明,吴达,高续波,严静.知识产权交易平台建设[J].天津科技,2015,42(09):95-97.

[3]. 苏玉慧. 基于B/S架构的高校二手网络交易平台的设计与实现[D].南昌大学,2016.

[4]. 刘旭.Chrome V8引擎中的JavaScript数组实现分析与性能优化[J].计算机与现代化,2014(10):66-70.

[5]. 陈浩.基于Javascript的异步编程分析[J].电脑知识与技术,2015,11(13):80-81.

[6]. 王金龙,宋斌,丁锐.Node.js:一种新的Web应用构建技术[J].现代电子技术,2015,38(06):70-73.

[7]. 王光磊.MongoDB数据库的应用研究和方案优化[J].中国科技信息,2011(20):93-94+96.

[8]. Chodorow, K. MongoDB: The Definitive Guide: Powerful and Scalable Data Storage[J]. "O'Reilly Media, Inc.", 2013

[9]. 朱二华.基于Vue.js的Web前端应用研究[J].科技与创新,2017(20):119-121.

[10]. 麦冬,陈涛,梁宗湾.轻量级响应式框架Vue.js应用分析[J].信息与电脑(理论版),2017(07):58-59.

[11]. 路雯雯. 支持前后端分离的JavaScript开发框架的研究及在内容管理系统中的应用[D].山东大学,2017.

[12]. 张智,郑卉,蒋依伶,袁欢欢,郑明清.使用Github实现高效的团队协作开发[J].电脑知识与技术,2015,11(07):206-208.

[13]. 林波. 基于Web2.0的网络交易平台研究与实现[D].上海交通大学,2009.

[14]. 程桂花. MVVM前后端数据交互中安全机制的研究与实现[D].浙江理工大学,2017.

[15]. 冯伟华,刘亚丽.基于Cookie的统一认证系统的设计与实现[J].计算机工程与设计,2010,31(23):4971-4975.

[16]. 王惠.基于Node.js的旅游网站设计[J].科技经济导刊,2017(17):32.

[17]. M. Jones, J. Bradley, N. Sakimura. JSON Web Token (JWT)[J], MAY 2015

[18]. 李凯.基于JSON Web Token的无状态账户系统的设计[J].现代计算机(专业版),2016(16):59-62.

[19]. 郝进义.数据库设计规范及设计技巧研究[J].计算机光盘软件与应用,2012(12):176-177.

[20]. 杨颖.基于媒体查询的自适应 Web 设计方法研究与实现[J].电脑知识与技术,2017,13(12):78-79+91.

[21]. 杨焕. 智能手机移动互联网应用的界面设计研究[D].武汉理工大学,2013.

[22]. 黄龙泉.媒体查询在响应式网站中的应用[J].电脑编程技巧与维护,2017(15):77-79.

[23]. 杜冬梅,许彩欣,苏健.浅谈正则表达式在 web 系统中的应用[J].计算机系统应用,2007(08):87-90.

[24]. Mike Cantelon, Marc Harter, TJ Holowaychuk, Nathan Rajlich. Node.js in Action[J]. Manning Publications Co. Greenwich, CT, USA 2013

[25]. 李素铎,马仲海.浅谈网站测试的基本方法[J].计算机时代,2008(08):14-15.

[26]. 胡静.浅析黑盒测试与白盒测试[J].衡水学院学报,2008(01):30-32.

[27]. 袁泉.网站维护和安全管理的重要性分析[J].计算机光盘软件与应用,2014,17(11):194+196.

[28]. 石永革,程乐.Web 网站维护的一种方法[J].计算机与现代化,2001(01):41-44+47.

致谢

在毕业设计即将完成的时候，回首这四年充实的学习时光，欣慰而不舍。我要对这四年来所有帮助过我、鼓励过我、教导过我的老师和同学致以诚挚的感谢，感谢他们在我迷茫时指引我方向，在我受挫时给予我鼓励，在我取得成就时告诉我学无止境。你们的陪伴和背后的默默付出，让我有充实的大学生活和今天的学识，而这期间培养的学习态度更将让我受益终身。

此次毕业设计的顺利完成，首先要感谢的是我的两位指导老师：王腾老师和朱定局老师。入学时就非常敬佩两位老师的教学态度，在大四学年有幸这两位老师能成为我的论文导师，更是给了我更多向他们学习的机会。在我们迷茫于做什么毕业设计时，两位导师对我们的学习情况和兴趣进行了了解，给了我们提供了许多设计思路，为我们的毕业设计指明了方向。而在我们项目中期，导师也对我们的项目完成情况进行了了解，并勉励我们抓紧完成，是我们很受鼓舞。在毕业论文的攥写过程中，导师也给了我们很多意见建议，许多不够科学的词语表述都被导师一一指出。

其次我要感谢我的组员詹坤展同学，在我忙于国考笔试和面试时，坤展同学给予我充分的理解与鼓励。在项目开发的过程中我们互相探讨、高效配合、分工明确，我在大数据方面许多不懂的地方他都能详尽解答。

再者我要感谢华南师大和计算机学院对我的栽培，学校倾其所能为我们提供了优异的学习环境，图书馆里的汗牛充栋的书籍、课室大楼里传道授业解惑的老师、学院里先进的实验室、饭堂里香甜的饭菜，这些都必将成为我难忘的回忆。感谢母校，祝福母校，相信母校一定会在建设高水平大学的道路上坚实迈步。

最后我要感谢我的父母，是他们在背后默默地为家庭付出，没有他们就没有我心无旁骛的学习环境，不愁吃喝的日常起居，我一天天长大的同时他们也在一天天老去，如今我学有所成，是时候凭借我所学的知识挑起家庭的重担，减轻他们的负担，让他们也过上不愁吃喝的生活。

马逸青

二〇一八年四月五日