

**广州城建职业学院**

**毕业设计**

**题 目 基于区块链加密技术在web开发中的设计与实现**

**所 在 系 信息工程学院**

**专业班级 计算机应用技术1班**

**学生姓名 邹威鹏**

**学生学号 \_\_1804010155**

**指导老师 陈焰**

**广州城建职业学院教务处制**

摘 要

在2020年，web技术更新日益频繁，谷歌，蚂蚁金服等越来越多的企业和开发者投身研发技术中。2020年疫情带来了更多的机遇与挑战，疫情中网课需求量上升，虚拟货币市场波动，让 IM即时通讯、货币加密 等技术不再是新潮，让我意识到技术和创新才是新的方向。

而现在，隐私也是用户所注重的一个痛点，越来越多的人们发现自己的隐私被互联网窥视，也许在一个app中输入的一段文字，在网购平台中就会给你推荐，用户信息在大数据分析下无所遁形。

因此我开发了A Blog博客平台，该网站能进行一些文章编写和上传储存功能，文章通过区块链技术进行加密，杜绝了被修改的风险。

该网站，由首页，文章页，待办页，编辑页，关于页组成。主要用于解决用户隐私问题和轻度办公需求。

关键词**：区块链，加密，博客，javascript ,web**

目录

[绪论 5](#_Toc58126847)

[研究背景 5](#_Toc58126848)

[相关领域已有的研究成果 7](#_Toc58126849)

[研究目的 11](#_Toc58126850)

[论文结构 11](#_Toc58126851)

[技术理论 12](#_Toc58126852)

[技术栈 12](#_Toc58126853)

[前端 12](#_Toc58126854)

[ReactJS 12](#_Toc58126855)

[Redux 12](#_Toc58126856)

[Typescript 12](#_Toc58126857)

[Antd 13](#_Toc58126858)

[Less 13](#_Toc58126859)

[Echartjs 13](#_Toc58126860)

[Axios 13](#_Toc58126861)

[Cryptojs 13](#_Toc58126862)

[Tinymce 13](#_Toc58126863)

[后端 15](#_Toc58126864)

[Nestjs 15](#_Toc58126865)

[Passport 15](#_Toc58126866)

[JWT 16](#_Toc58126867)

[Mongoose 16](#_Toc58126868)

[Swagger 16](#_Toc58126869)

[服务器 17](#_Toc58126870)

[Ecs 17](#_Toc58126871)

[Nginx 17](#_Toc58126872)

[MongoDB 17](#_Toc58126873)

[Pm2 17](#_Toc58126874)

[程序设计思路 18](#_Toc58126875)

[小结 19](#_Toc58126876)

[技术实现 20](#_Toc58126877)

[设计思路 20](#_Toc58126878)

[前端 21](#_Toc58126879)

[页面功能展示 21](#_Toc58126880)

[主要功能代码 27](#_Toc58126881)

[后端 31](#_Toc58126882)

[代码 31](#_Toc58126883)

[服务器 37](#_Toc58126884)

[总结与展望 38](#_Toc58126885)

# 绪论

## 研究背景

要说区块链产生的背景，就不得不说一说人类的进步。

文字的发明标志着知识的传承，信息的流通和发展促进着人类社会的进步。货币的产生代表着价值体系和信用体系的建立，价值的传递促进了商品的流通，也就标志着物质的不断丰富。

区块链被列入国家战略

2019 年 10 月 24 日下午，中共中央政治局就区块链技术发展现状和趋势进行第 18 次集体学习，习近平总书记发表了重要讲话，将区块链作为国家核心技术自主创新的重要突破口。

区块链技术得到国家层面如此高规格的定位尚属首次，这标志着区块链技术已被上升为国家战略，也意味着以大数据、人工智能和区块链为主要支撑技术的中国数字经济新战略的基本确立。

党的十九大以来，中共中央政治局迄今已经进行 19 次集体学习，其中三次与数字经济相关，主题分别是大数据、人工智能和区块链。其中，对于区块链的定位是最高的。

习近平总书记在主持学习时强调，区块链技术的集成应用在新的技术革新和产业变革中起着重要作用。我们要把区块链作为核心技术自主创新的重要突破口，明确主攻方向，加大投入力度，着力攻克一批关键核心技术，加快推动区块链技术和产业创新发展。习近平总书记指出，区块链技术应用已延伸到数字金融、物联网、智能制造、供应链管理、数字资产交易等领域。目前，全球主要国家都在加快区块链布局。我国在区块链领域拥有良好的基础，要加快推动区块链技术和产业创新发展，积极推进区块链和经济社会融合发展。会上还总结了区块链的五大作用，分别是促进数据共享、优化业务流程、降低运营成本、提升协同效率、建设可信体系，要抓住区块链技术融合、功能拓展、产业细分的契机，发挥区块链的作用。

香农量创造了Bit,量化了信息，发布了《通信的数字原理》为作为信息传递网络的互联网打下了基础，而互联网作为信息传递网络已经大大的影响了我们的生活，极大地促进社会和人类的进步。

而货币，早在原始人时代就存在，原始的贝壳货币，圆石货币（为记账货币）等，货币的最早期是以商品货币形式存在，再到后来的记账货币，再到现在的信用货币。随着金本位制度崩溃，货币金属论不攻自破，货币名目论被实际证明了。在货币名目论中对货币的认识是：货币不是财富，主张货币只是一个符号，一种票证，是名目上的存在，是便利交换的技术工具。即将货币定性为信用货币。而信用货币的本质就是‘信用’与‘货币’的结合，正如《人类简史》说的“Money is a system of mutual trust, and it is not a system that follows it - money is the most common and effective system of trust in history.”。当没有信用体系的时候只能使用商品型货币，但是信用体系建立后货币可以作为信用体系的一个产物来起到货币的作用。

## 相关领域已有的研究成果

区块链的应用。首先先说央行数字货币——央行数字货币的技术其实早就成熟了，都用不到区块链，因为区块链的目的是去中心化，而央行数字货币用不到这个。当然，采用区块链的央行数字货币技术也早就有了（参见RSCoin），这里面可能会在某些环节采用区块链技术去中心化——例如货币的发行仍旧是中心化的，但是流转中的管理和验证可以通过区块链技术去中心化到其他银行。同时，据我所知，官方早就有一个数字货币研究所在研究这个很久了。我在很早就写过，从大势上，货币的数字化是必然的，但这和比特币是两码事，看起来可能会更像支付宝。我觉得从很大程度上推动了这次区块链的新闻，也推动了央行数字货币的推出。libra本身只代表facebook的野心，目前来看，他们的野心很有可能不会成功，但是也许是这种野心和指明的技术方向让国家产生了紧迫感——因为就算现在libra的所有成员都退出得差不多了，libra本身也前途未卜，但是万一类似的框架被搭起来又吸引了其他的重量级成员加入，那么国家在这上面就失去了主导权。因此，根据最近的新闻，央行的数字货币应该已经非常接近了。

但这点，并不足以突然让“区块链”成为国家战略，这里说的区块链，一定是包含数字货币之外的其他应用的。

这次战略中最主要提出的，是“要掌握区块链核心技术”，那么区块链的核心技术是什么呢？

区块链相关技术有很多方面，我可以把它大致分成4类：

1），共识算法；

2），隐私保护技术和相应密码学技术；

3），智能合约相关技术；

4），面向应用相关技术。其中，前三者可以认为是底层，后两者是上层。

1），其中，共识算法是区块链最核心，也是整个技术发展和学术界最热衷的领域：大量的公链项目都追求共识算法的创新，同时，大量的研究者也投身于这个领域。

但是共识算法中的涵盖面非常广，其中有一个概念非常重要——有许可和无许可，有许可共识算法要求所有参与节点的身份已知，主要采用类BFT算法。而无许可共识算法则允许任何节点只要达到某些条件就能进入，无需事先在任何系统中注册身份，而POW和后来衍生出的POS以及其他许多POx类算法都在此列。前者，多用于联盟链，而后者多用于公有链。其中，两者发展至今日，在速度和延迟上的差异已经不大了，归根结底的差别就是前面说的有无许可的差别。而无许可共识算法由于无许可的特性，又引出了一个很重要的概念叫“激励机制”，比如比特币的挖矿，于是，又衍生出了“货币”。于是，就又衍生出许多非常有趣的方向，例如：ASIC矿机，链上治理，加密经济学等等。

2），其实区块链技术本身采用的密码学技术并不高深也不复杂，无非是哈希函数和数字签名这两个非常经典和简单的密码学工具而已。然而，由于区块链技术和密码学天生的契合性（大部分早期研究区块链的学者都来自于密码学领域），同时，由于“加密货币”这个词，即：cryptocurrency，（注：在英文中，这个词被用来指代所有类似于比特币之类的货币，而不是中文领域中常用的“数字货币”），导致大部分人都认为密码学天生和区块链是联系到一起的。

实际上，区块链技术本身并用不到特别先进的密码学工具（当然，一些新的共识算法中采用了一些新的密码学工具）。但是，其中最重要的密码学挑战，是隐私保护技术，而其中，可能会用到的，包括零知识证明，同态加密等目前还在发展之中的密码学工具。这点，其实并不是区块链本身的需求，但又和区块链紧紧联系在一起——原因，还得说回之前区块链的性质和区块链的应用上：

首先，区块链最大的特点是所有上链的数据必须通过所有节点的验证，然而，如果区块链得到了大规模应用，这就代表了所有的节点都会获取所有的数据。好吧，我们现在不信任中心，有很大的一部分原因是中心获取了我们所有的数据，而现在倒好，用了区块链去中心之后，所有人都可以获取我们所有的数据了。

因此，如何在能够保证隐私的同时还可以享受区块链带来的去中心化优势呢？这需要所有节点能够在不知道数据内容的情况下，还能够对数据的有效性进行验证，例如：在不知道你是谁，具体有多少钱，你想要支出多少钱的情况下，验证你确实有一笔钱可以来完成这笔支出……

这听起来像是天方夜谭，然而，实际上我们有一种密码学工具可以实现它——叫做零知识证明。

然而，目前零知识证明尽管一直在发展，但是目前仍存在很大的局限性，这种局限性是效率——我们有了一些高效的证明方式，例如：我们可以很快地证明某个未知的数比另一个已知数大，或者我们可以证明某个未知的值来自于某个集合。然而，这些高效的零知识证明，还不足以覆盖区块链的所有应用。因此，可想而知，这些新加密技术，尽管不是区块链技术中的必须品，却是区块链技术能够得到广泛应用的必需品，所以一定是“核心技术”中的一部分。

3），智能合约相关的技术，实际上包括语言，包括编译……这些，实际上相当于是当我们已经有了区块链作为底层之后，如何使用它的问题。在这上面目前还没有一个非常明确的方向，因为，其实目前在尝试的思路有几种：

i，以太坊式的，高度自由的图灵完备的系统。然而问题是，写程序太容易出错了，而且出错的代价太大。

ii，Hyperledger-fabric式的，高度模块化的系统，然而，极度复杂，复杂到想要采用HF的公司，要么放弃，要么花钱请IBM的人帮他们做。

iii，Libra式的，专注于某项应用的系统，例如libra的move语言就明白地说了，我们不追求图灵完备，我们的目标是做一个更方便于实现各种金融场景的语言——这可能是个趋势，但是还没有经过足够的检验。

这些方向，在目前的尝试还很少。大部分区块链项目仍旧采用以太坊式的系统，或者加以改进。然而，我认为这一点很有可能成为“区块链的核心技术”中最重要的部分——因为一旦区块链上升到了国家战略的高度，那么势必会诞生大量的需求、场景和用户，于是才会有充分的尝试和实现的平台。由此，可能会衍生出一个最适合区块链的语言和开发工具，而这个，会如同OS一样，成为一个新的世界范围内的标准。而从目前来看，在其他的区块链项目都在挣扎着寻求落地的时候，可能只有中国有这样的土壤能够诞生出这样的标准。这大概就是把区块链作为国家战略的原因。

4），应用相关的技术。其实在区块链应用方面最大的技术难点并不在技术上。例如：区块链存证，如果我们做一条联盟链将相关各方作为共识节点然后把证据上链，那么，这个证据就是无法篡改的了。这道理大家都懂，然而问题在于：谁来做这个事呢？证据怎么数字化？怎么运行区块链？给出数据我的好处在哪里？最重要的是，谁掏钱？

区块链目前落地最大的难点，在于其相比传统的中心化方案，除了在存证，溯源方面的一些场景下，找不到明显的优势，也就找不到可行的商业模式。于是，也就很难通过市场的方式推行出去，近两年币市的低迷已经证明了这一点。然而，这一切都会在国家的支持下产生变化，于是，就如同中国互联网产业从模仿学习国外到开始向外输出抖音一样，中国希望能在区块链领域，能够激发出更多的尝试和场景。于是，在这些场景中，我们会积累相关的经验、技术和商业模式，从而将这种模式在出口到国外去，而这点，也会是区块链核心技术中非常重要的部分。

## 研究目的

通过本次研究实现区块链的实际应用场景之一，开发了区块链博客平台，用户能在平台上创建博客、预览他人博客、轻度办公等需求。区块链的不可篡改特性保证了用户的文章信息安全和隐私安全。

## 论文结构

第一章主要介绍了背景和目的。第二章介绍了所使用的技术栈。第三章展示了实现的过程。第四章总结。

# 技术理论

## 技术栈

前端

ReactJS

React 起源于 Facebook 的内部项目，因为该公司对市场上所有 JavaScript MVC 框架，都不满意，就决定自己写一套，用来架设 Instagram 的网站。做出来以后，发现这套东西很好用，就在2013年5月开源了。由于 React 的设计思想极其独特，属于革命性创新，性能出众，代码逻辑却非常简单。所以，越来越多的人开始关注和使用，认为它可能是将来 Web 开发的主流工具。

Redux

Redux 是 JavaScript 状态容器，提供可预测化的状态管理。可以让你构建一致化的应用，运行于不同的环境（客户端、服务器、原生应用），并且易于测试。

Typescript:

TypeScript是微软开发的一门编程语言，它是JavaScript的一个超集,它遵循最新的ES6脚本语言规范，TypeScript扩展了JavaScript的语法，已经存在的JavaScript程序可以不经任何改动的情况下TypeScript环境下运行。TypeScript只是想JavaScript添加了一些新的遵循ES6规范的语法、以及基于类的面向对象编程的特性，其次2016年9月底发布的Angular2框架是由TypeScript语言来编写的，Angular2是谷歌公司发布的一套前端框架，也就是说TypeScript是由微软和谷歌两家大型科技公司在背后做技术支持的。基于这一点我们有理由相信在未来一段时间内，TypeScript有可能成为前端脚本语言发展的一个主流方向。

Antd

Ant Design设计语言和组件库蚂蚁金服体验技术部经过大量的项目实践和总结，沉淀出设计语言 Ant Design，这可不单纯只是设计原则、控件规范和视觉尺寸，还配套有前端代码实现方案。也就是说采用Ant Design后，UI设计和前端界面研发可同步完成，效率大大提升。目前有阿里、美团、滴滴、简书采用。Ant Design有Web版和Moblie版。

Less

Less 是一门 CSS 预处理语言，它扩展了 CSS 语言，增加了变量、Mixin、函数等特性，使 CSS 更易维护和扩展。Less 可以运行在 Node 或浏览器端。

Echartjs

harts，一个使用 JavaScript 实现的开源可视化库，可以流畅的运行在 PC 和移动设备上，兼容当前绝大部分浏览器（IE8/9/10/11，Chrome，Firefox，Safari等），底层依赖轻量级的矢量图形库 [ZRender](https://github.com/ecomfe/zrender)，提供直观，交互丰富，可高度个性化定制的数据可视化图表。  
ECharts 提供了常规的折线图、柱状图、散点图、饼图、K线图，用于统计的盒形图，用于地理数据可视化的地图、热力图、线图，用于关系数据可视化的关系图、旭日图，多维数据可视化的平行坐标，还有用于 BI 的漏斗图，仪表盘，并且支持图与图之间的混搭。

Axios

Axios 是一个基于 promise 的 HTTP 库，可以用在浏览器和 node.js 中。

Cryptojs

JavaScript library of crypto standards，Js加密库。

#### Tinymce

TinyMCE是一个轻量级的基于浏览器的所见即所得编辑器，由JavaScript写成。它对IE6+和Firefox1.5+都有着非常良好的支持。功能方面虽然不能称得上是最强，但绝对能够满足大部分网站的需求，并且功能配置灵活简单。另一特点是加载速度非常快。

后端

Nestjs

Nest (NestJS) 是一个用于构建高效、可扩展的 [Node.js](https://nodejs.org/) 服务器端应用程序的开发框架。它利用 JavaScript 的渐进增强的能力，使用并完全支持 [TypeScript](http://www.typescriptlang.org/) （仍然允许开发者使用纯 JavaScript 进行开发），并结合了 OOP （面向对象编程）、FP （函数式编程）和 FRP （函数响应式编程）。

在底层，Nest 构建在强大的 HTTP 服务器框架上，例如 [Express](https://expressjs.com/) （默认），并且还可以通过配置从而使用 [Fastify](https://github.com/fastify/fastify) 。

Nest 在这些常见的 Node.js 框架 (Express/Fastify) 之上提高了一个抽象级别，但仍然向开发者直接暴露了底层框架的 API。这使得开发者可以自由地使用适用于底层平台的无数的第三方模块。

近年来，由于 Node.js、JavaScript 已经成为 web 前端和后端应用程序的“通用开发语言”。这促成了诸如 [Angular](https://angular.io/)、[React](https://github.com/facebook/react) 和 [Vue](https://github.com/vuejs/vue) 等优秀项目的出现，他们提高了开发者的工作效率，并能够创建快速、可测试和可扩展的前端应用程序。然而，尽管 Node（和服务器端 JavaScript）拥有大量优秀的软件库、辅助程序和工具，但没有一个能够有效地解决我们所面对的主要问题，即架构。

Nest 提供了一个开箱即用的应用程序体系结构，允许开发者及其团队创建高度可测试、可扩展、松散耦合且易于维护的应用程序。这种架构深受Angular 的启发。

Passport

Passport is authentication middleware for Node.js. Extremely flexible and modular, Passport can be unobtrusively dropped in to any Express-based web application. A comprehensive set of strategies support authentication using a username and password, Facebook, Twitter, and more. passport.js

是Nodejs中的一个做登录验证的中间件，极其灵活和模块化，并且可与Express、Sails等Web框架无缝集成。Passport功能单一，即只能做登录验证，但非常强大，支持本地账号验证和第三方账号登录验证（OAuth和OpenID等），支持大多数Web网站和服务。

JWT

Json web token (JWT), 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（[(RFC 7519](https://link.jianshu.com?t=https:/tools.ietf.org/html/rfc7519)).该token被设计为紧凑且安全的，特别适用于分布式站点的单点登录（SSO）场景。JWT的声明一般被用来在身份提供者和服务提供者间传递被认证的用户身份信息，以便于从资源服务器获取资源，也可以增加一些额外的其它业务逻辑所必须的声明信息，该token也可直接被用于认证，也可被加密。

Mongoose

Mongoose是nodejs的MongoDB Object Modeling Map简称ODM，它类似于关系型数据库ORM，用于抽象化mongodb数据表模型，以简化对数据库交互的代码，增强代码的可读性和可维护性。

Swagger

Swagger?的目标是为REST APIs 定义一个标准的，与语言无关的接口，使人和计算机在看不到源码或者看不到文档或者不能通过网络流量检测的情况下能发现和理解各种服务的功能。当服务通过Swagger定义，消费者就能与远程的服务互动通过少量的实现逻辑。类似于低级编程接口，Swagger去掉了调用服务时的很多猜测。

服务器

Ecs

[云服务器 ECS（Elastic Compute Service）是一种弹性可伸缩的计算服务，助您降低 IT 成本，提升运维效率，使您更专注于核心业务创新。](https://www.aliyun.com/product/ecs?spm=5176.13910061.744563.2.7b2a2f57Xu4Orw)

Nginx

Nginx是一个高性能的HTTP和反向代理服务。Nginx 是由伊戈尔·赛索耶夫为俄罗斯访问量第二的 [http://Rambler.ru](https://link.zhihu.com/?target=http%3A//Rambler.ru) 站点（俄文：Рамблер）开发的，第一个公开版本0.1.0发布于 2004 年 10 月 4 日。

Nginx 是一个很强大的高性能Web 和反向代理服务，它具有很多非常优越的特性：在连接高并发的情况下，Nginx 是 Apache 服务不错的替代品。

MongoDB

MongoDB是专为可扩展性，高性能和高可用性而设计的数据库。它可以从单服务器部署扩展到大型、复杂的多数据中心架构。利用内存计算的优势，MongoDB能够提供高性能的数据读写操作。 MongoDB的本地复制和自动故障转移功能使您的应用程序具有企业级的可靠性和操作灵活性。

Pm2

PM2（Process Manager 2 ）是具有内置负载均衡器的Node.js应用程序的生产运行时和进程管理器。 它允许您永久保持应用程序活跃，无需停机即可重新加载它们，并促进常见的Devops任务。

## 程序设计思路

**前端**：

前端使用react框架，和redux状态管理，

有视图层(view)，动作层(action)，服务层(service)，状态层(state)。用户在视图层触发动作，请求服务，更新状态，渲染视图。

在用户、待办、编辑三个页面使用路由守卫。

从用户页面注册后，在编辑页面可以编辑自己的文章，提交时使用cryptojs进行非对称加密，提交成功后可以在文章页查看最新的文章。

文章页可以切换查看全球模式，这个模式下能展示全球不同地区的博主的位置，和链接下一个文章的地图路径。

待办页面考虑频繁增删改，所以不上传到数据库中，只在客户端本地缓存中存放和展示使用。

**后端**：

后端使用Nestjs，实现权限注册，登录；获取用户信息；查询文章，创建文章。

passport调用接口登录，实现多平台登录。

jwt进行签发令牌授权。

mongoDB + mongoose 连接数据库和实例化数据模型。

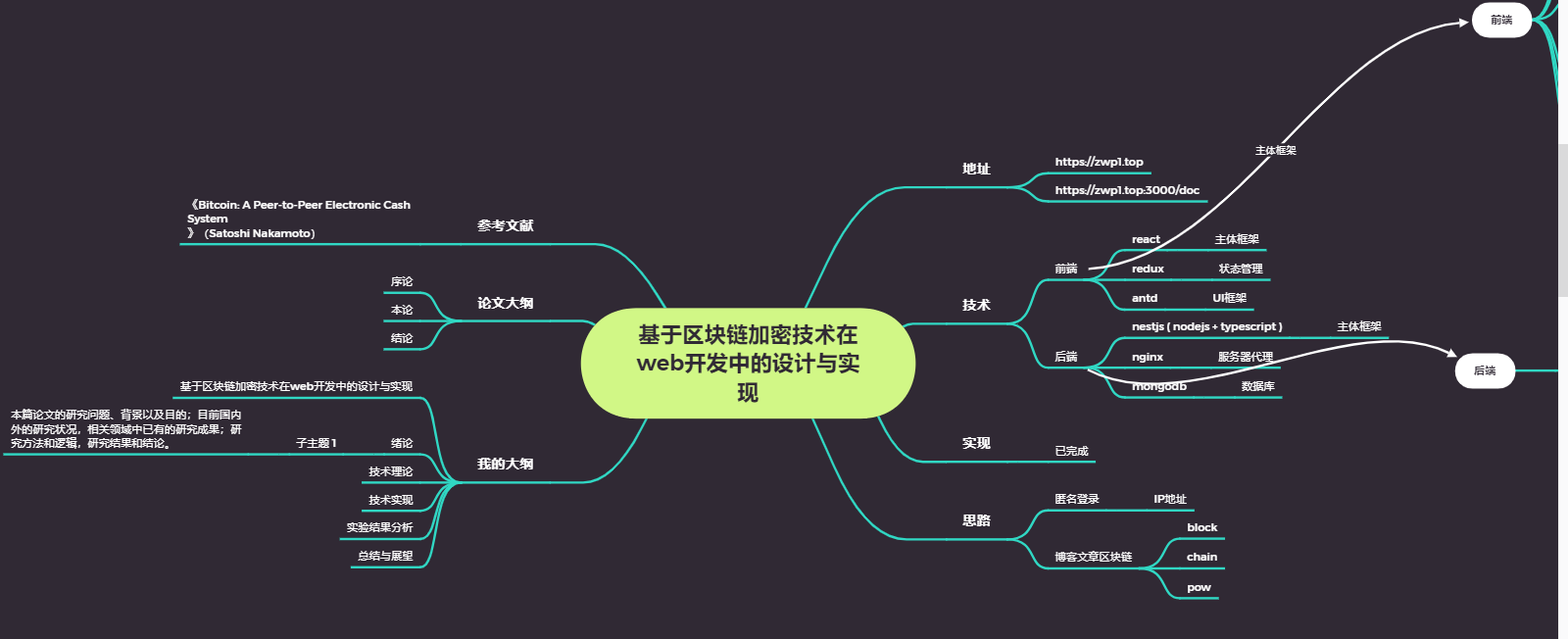
swagger 生成后端接口文档。

**服务器**

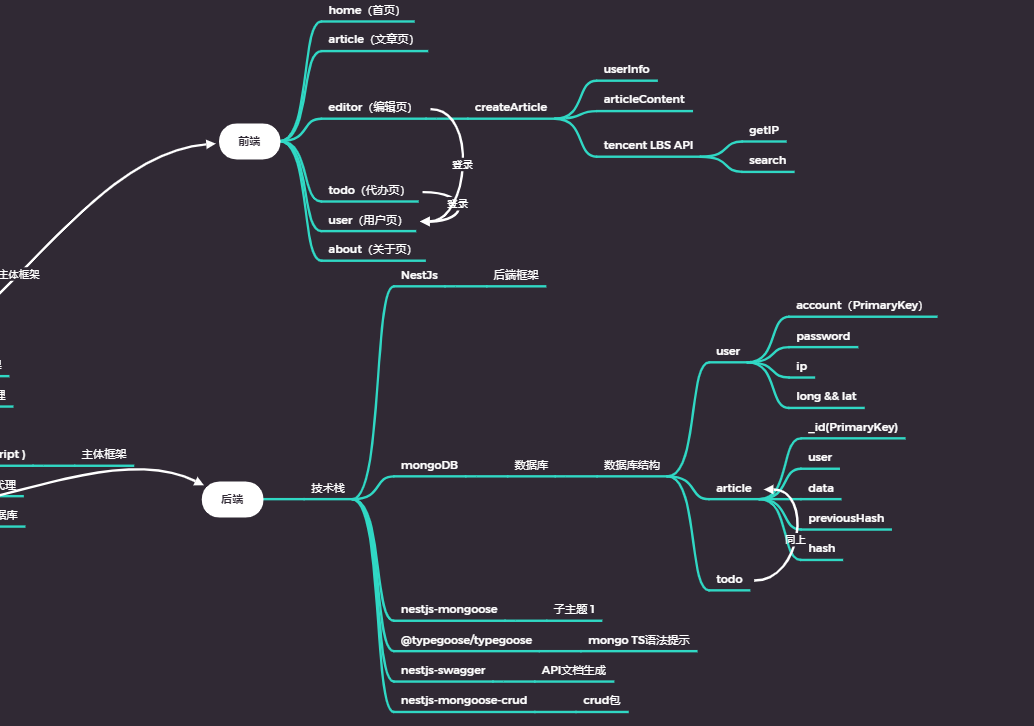
服务端使用阿里云ecs，通过nginx进行服务器代理443端口，指向存放前端代码文件位置。

使用PM2包部署后端代码服务，实现开机自启动，和服务器负载均衡。

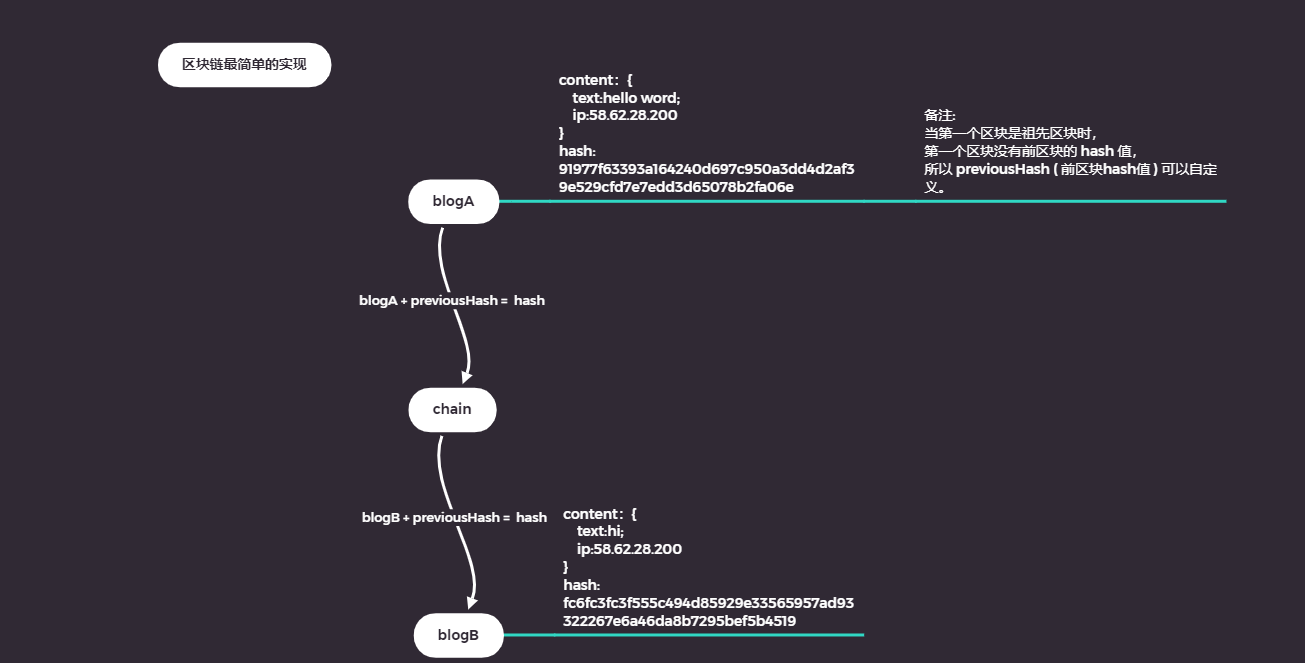
## 设计思路



确定主题



技术选型



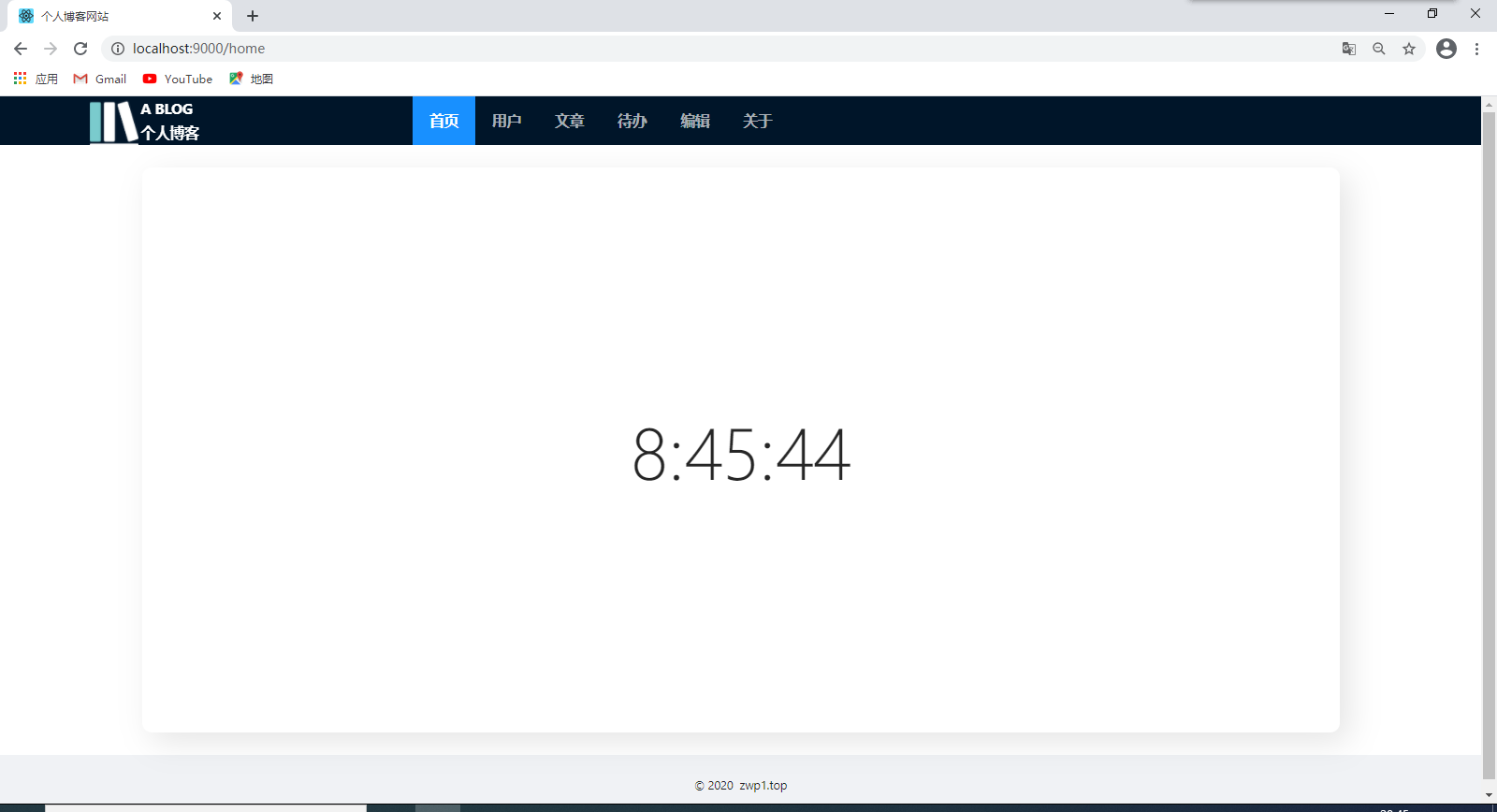
小结：本章展示所用到的主要技术栈和区块链模型，仅展示较为主要的技术内容，其余小框架插件不作展示。在下一章展示具体代码实现。

# 技术实现

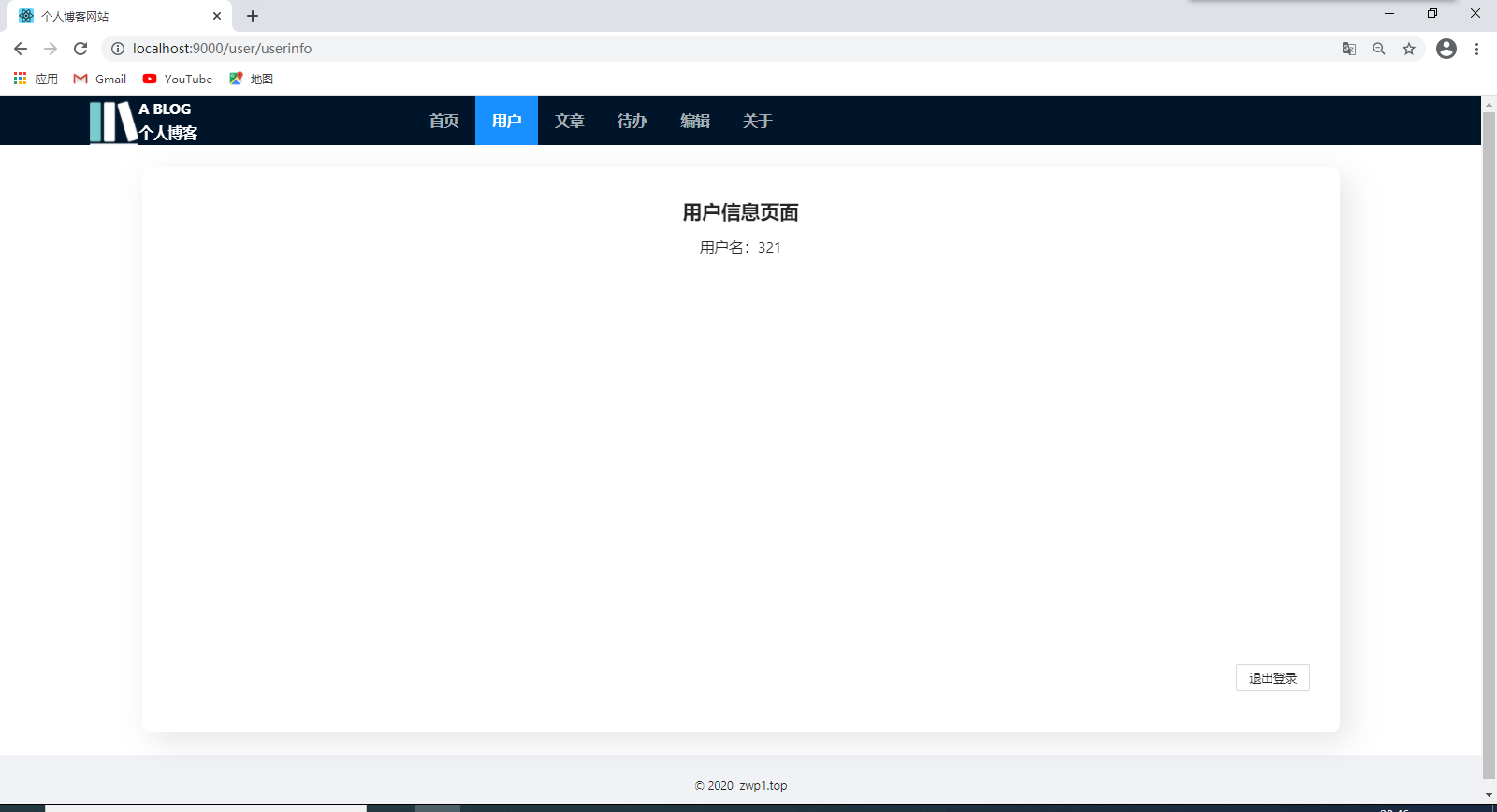
本章展示重要功能的实现。环境配置、初始化项目 等不做过多展示。

## 前端

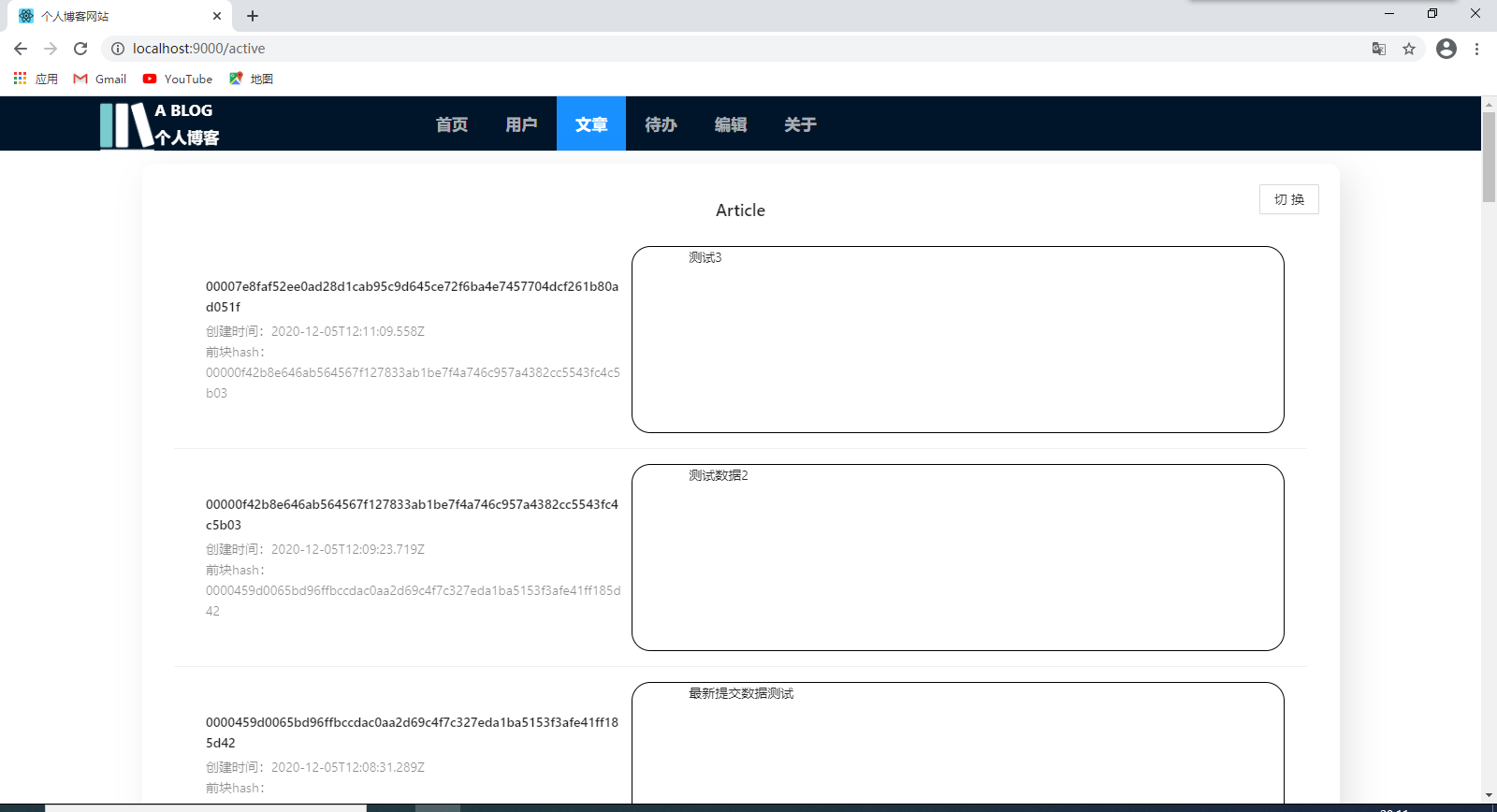
### 页面功能展示



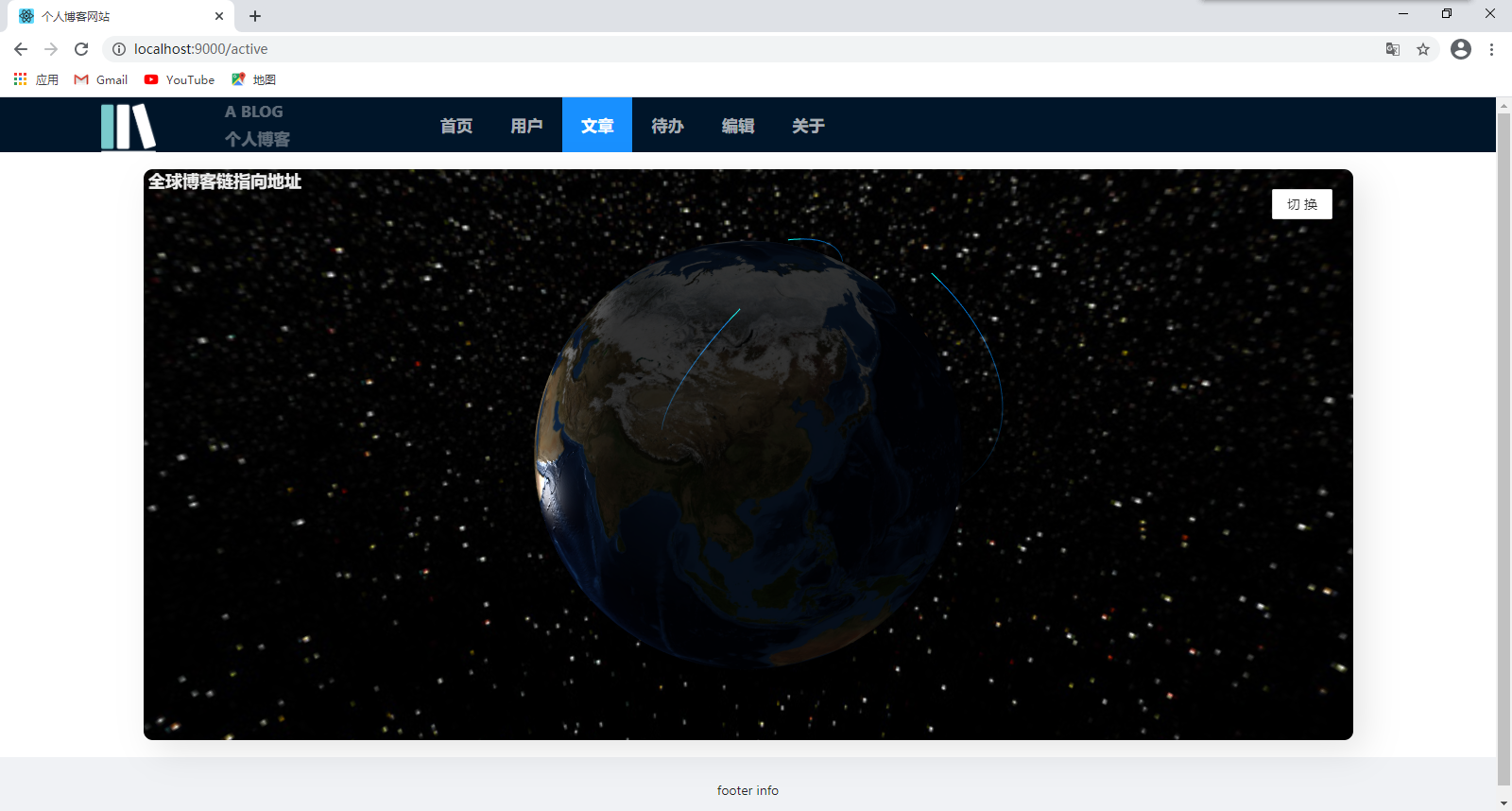
1-1 首页



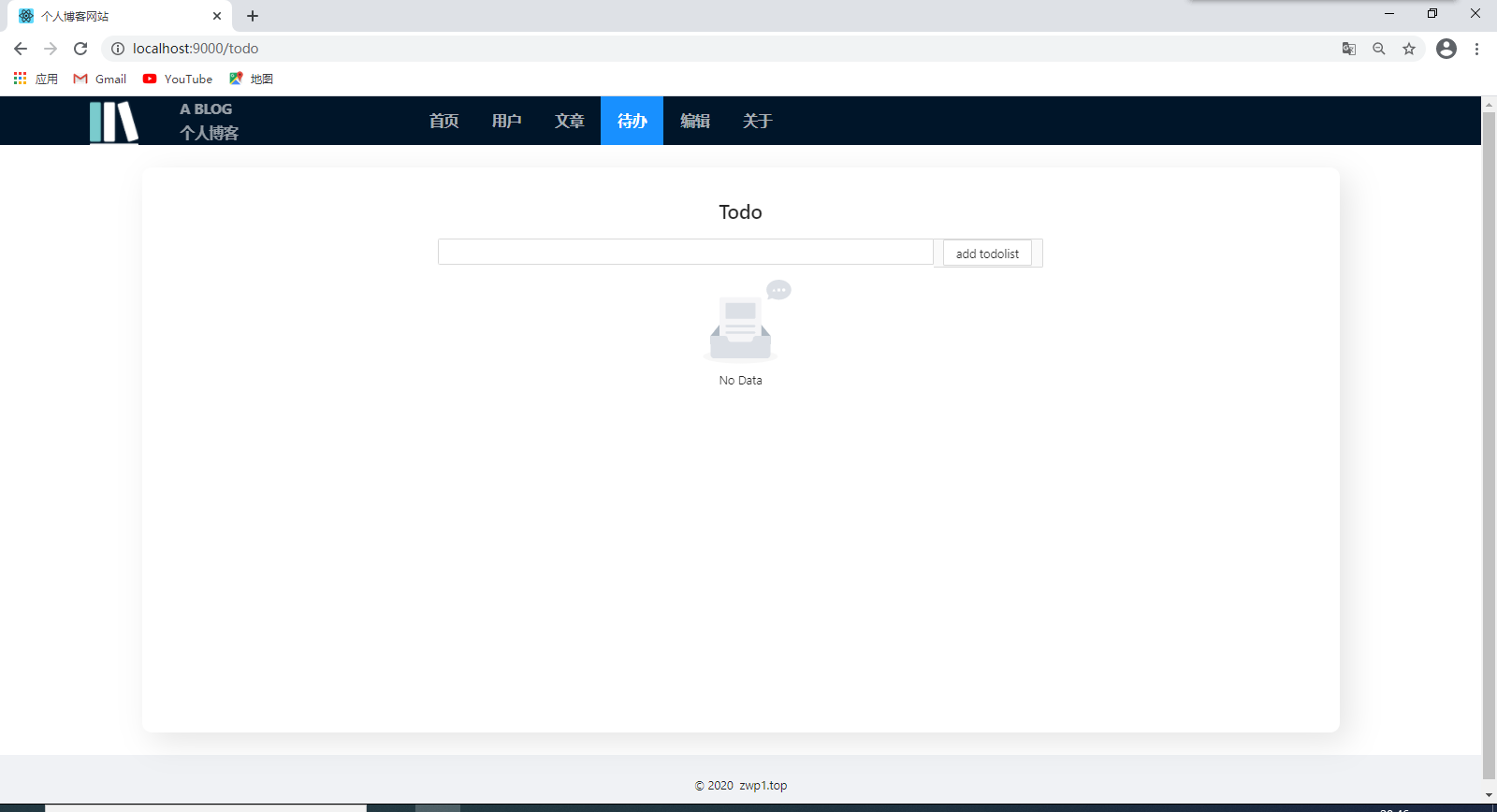
1-2 用户页



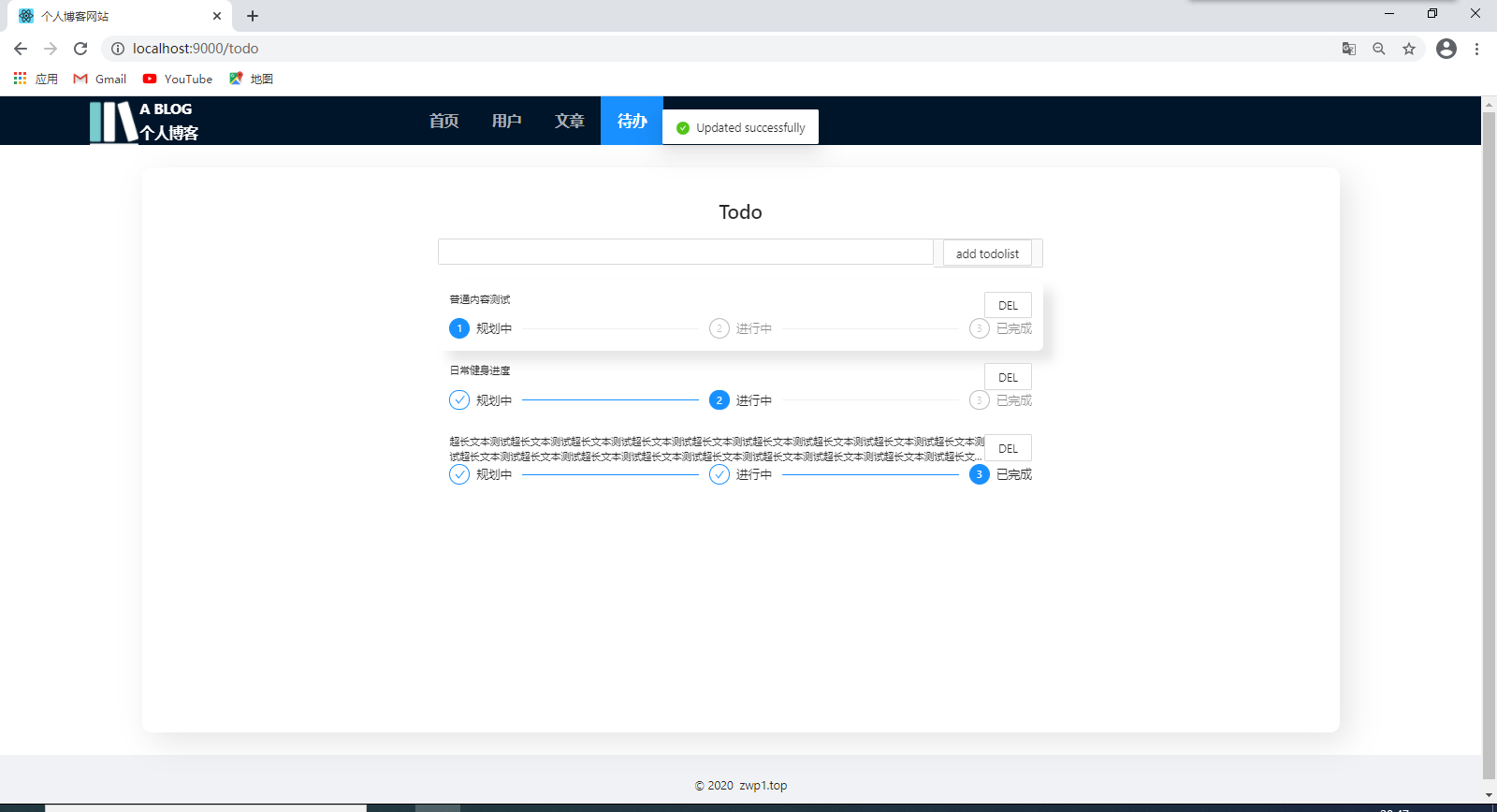
1-3 文章页



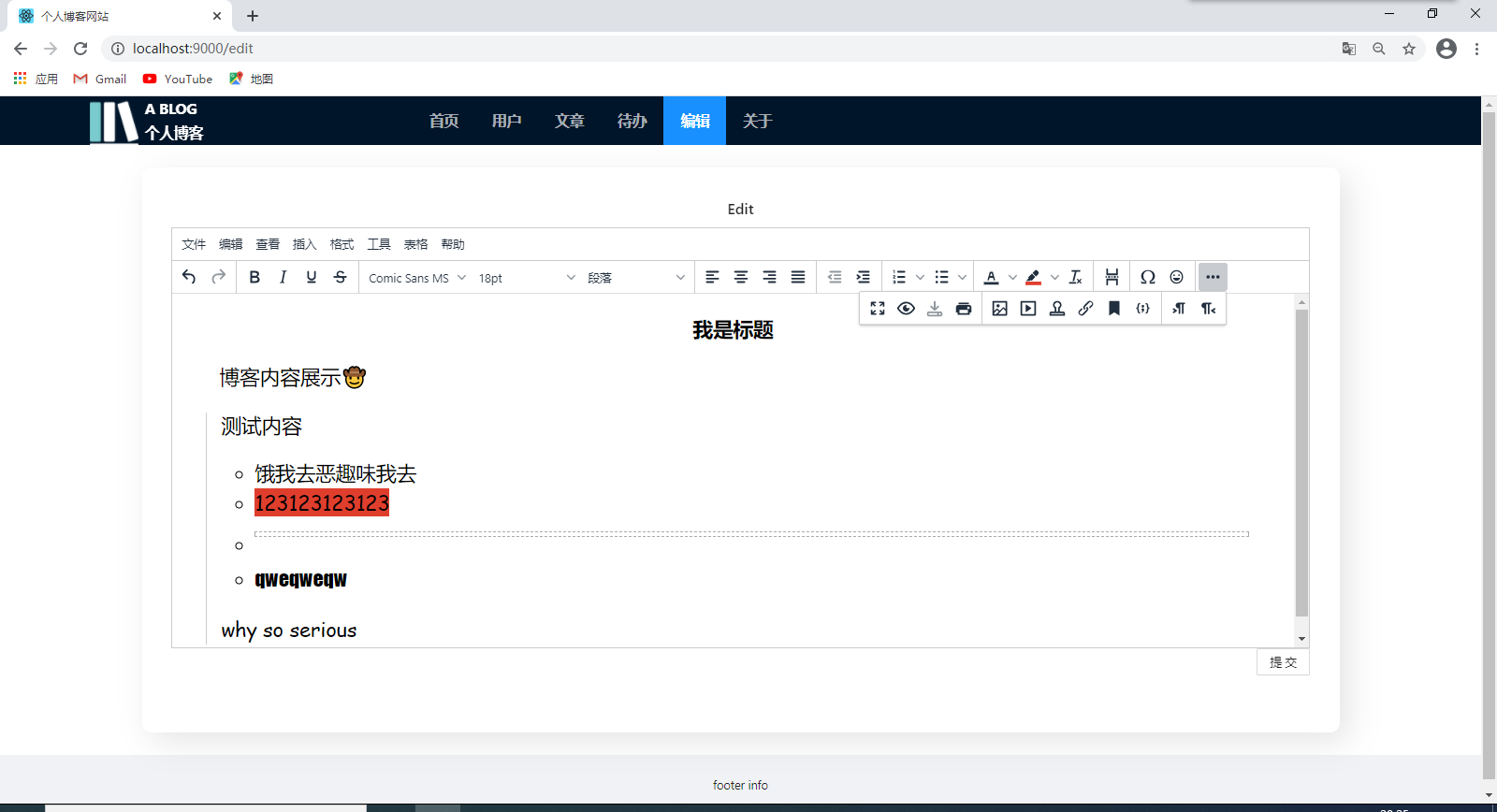
1-4 文章页



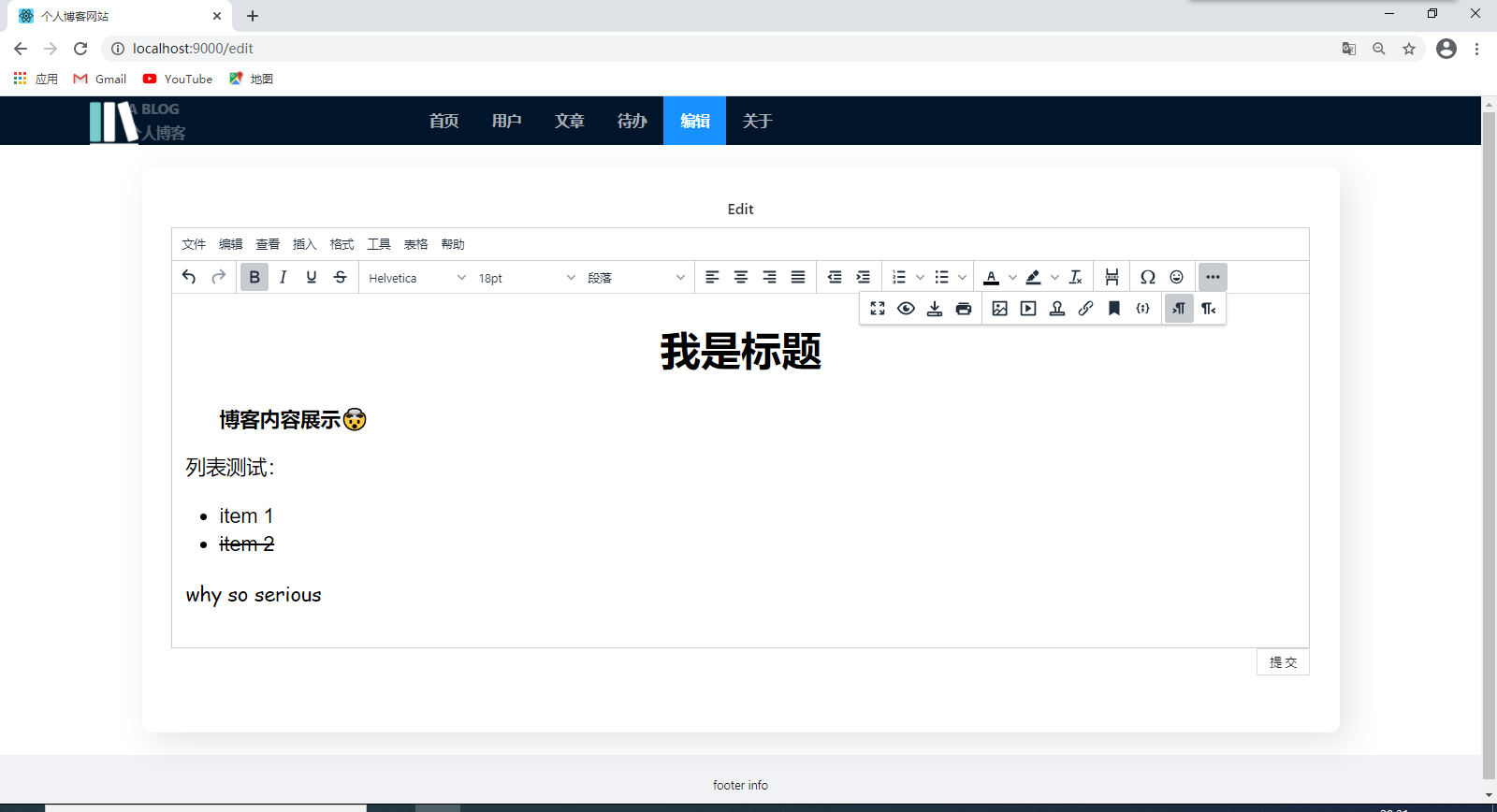
1-5 待办页



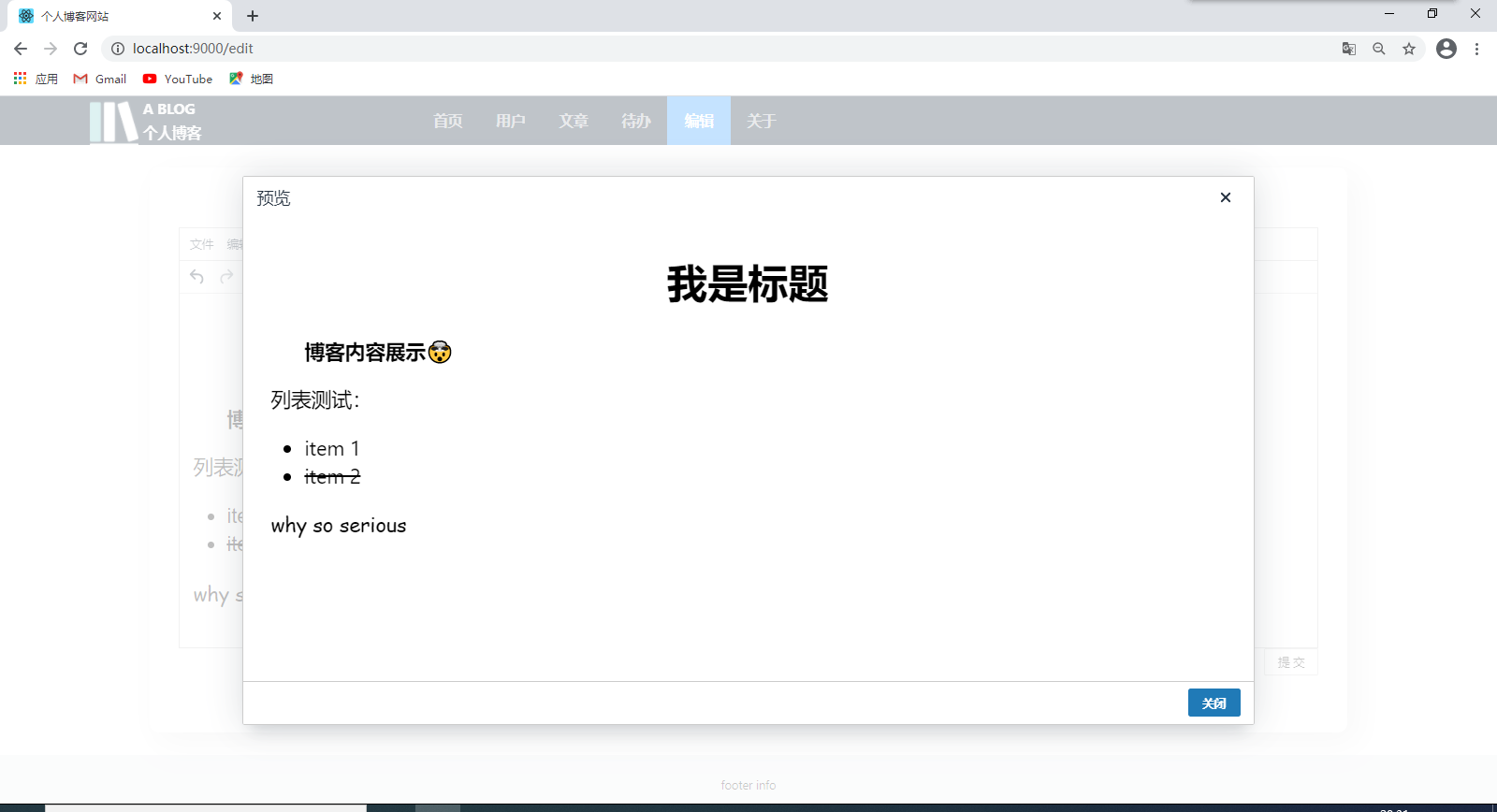
1-6 代办页



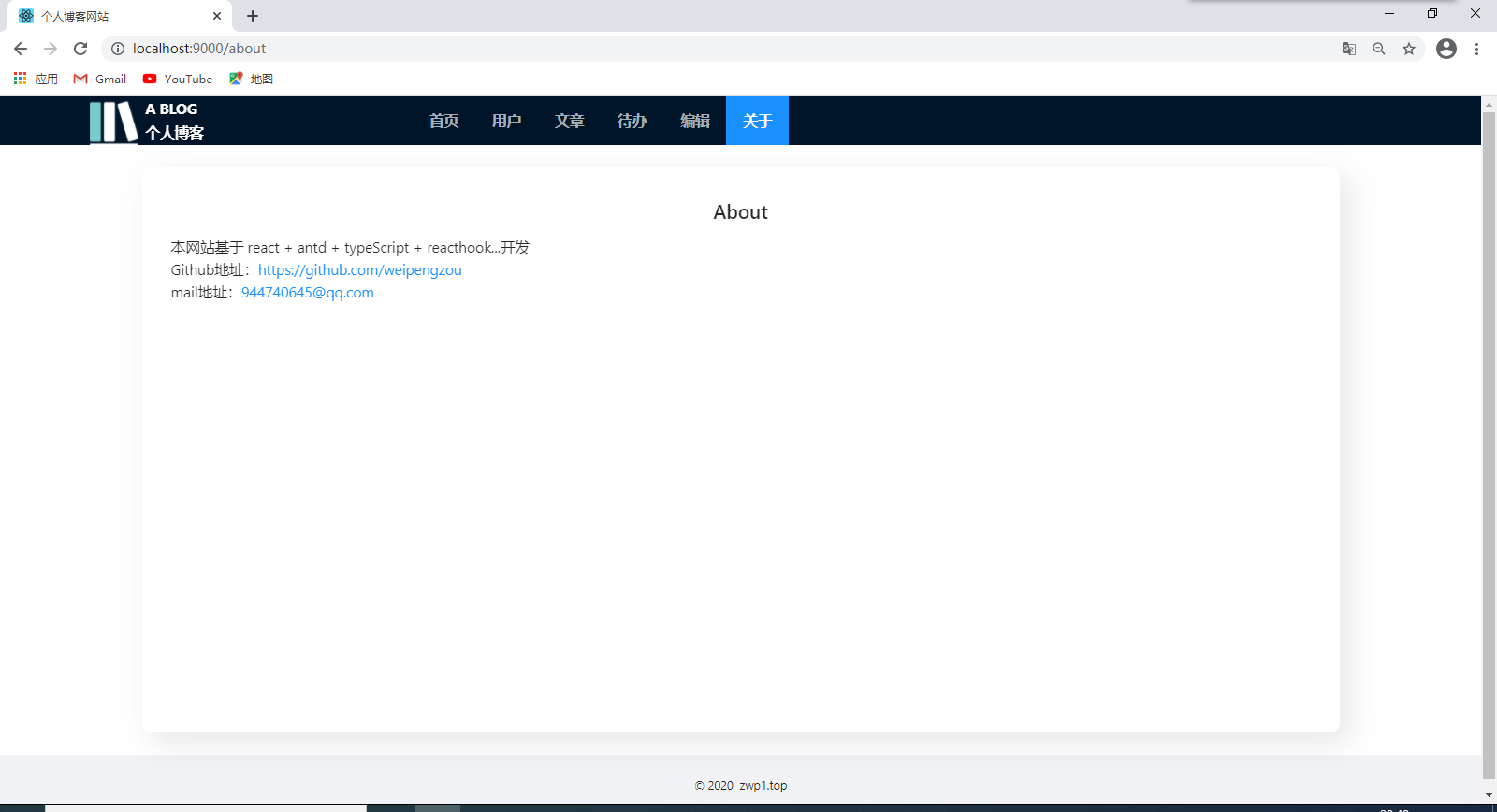
1-7 编辑页



1-8 编辑页



1-9 编辑页



1-10 关于页

### 主要功能代码

const loginAction = ({ account, password }: UserLoginInterFace) => async (dispatch: any) => {

  message.destroy()

  message.loading('登录中...')

  if (account && password) {

    try {

      let res = await userLogin({ account, password })

      console.log('登录回调', res)

      dispatch({

        type: actionTypes.USER\_LOGIN,

        payload: {

          data: res.data,

        },

      })

    } catch (error) {

      message.destroy()

      message.warning(String(error))

    }

  } else {

    message.destroy()

    message.warn('请输入账号或密码')

  }

}

const loginOutAction = () => async (dispatch: any) => {

  try {

    await localStorage.clear()

    dispatch({

      type: actionTypes.USER\_LOGINOUT,

      payload: {

        isLoginOut: true,

      },

    })

  } catch (error) {

    message.warn('login out error', error)

  }

}

const registerAction = ({ account, password }: UserLoginInterFace) => async (dispatch: any) => {

  message.loading('wating...')

  try {

    let res: any = await userRegister({ account, password })

    console.log('action', res)

    if (res.\_id) {

      message.destroy()

      message.success('注册成功')

    }

  } catch (error) {

    message.destroy()

    message.warning('注册失败', error)

  }

}

const getUserAction = (token: string) => async (dispatch: any) => {

  try {

    let res: any = await getUserInfo(token)

    dispatch({

      type: actionTypes.GET\_USERINFO,

      payload: {

        data: res.data,

      },

    })

  } catch (error) {

    console.log('getUserAction ERR', error)

  }

}

用户注册、登录、登出、token登录

// 创建文章

const createArticleAction = (payload: { articleContent: string; user: string; local: object }) => async (dispatch: any) => {

  message.destroy()

  message.loading('请稍等，计算文章区块中...')

  let res = await getArticle()

  // console.log('getArticleAPI',res);

  let blogChain = res.data

  let lastBlog = blogChain[blogChain.length - 1]

  // 文章数据,

  let data: string = crypto.encryptoAES(payload.articleContent)

  // 用户信息

  let user: string = payload.user

  // 地理信息

  let local: object = payload.local

  // 前块hash

  let previousHash: string = ''

  // 本块hash

  let hash: string = ''

  // 1.判断祖先区块

  if (!lastBlog) previousHash = ''

  else previousHash = lastBlog.hash

  // proof of work

  let randomNumb = 1

  let curTime = Date.now() // 设置时间

  let stopTime

  let diffNumb = 1

  while (true) {

    // 加一个随机数，在不影响其他参数情况下实现不同输出

    hash = crypto.encryptoSHA256(data + user + previousHash + randomNumb).toString()

    if (hash.substring(0, diffNumb) === '0'.repeat(diffNumb)) {

      stopTime = Date.now()

      // 防止算力过强的电脑短时间内提交多个文章块

      // 规定300ms内的

      if (stopTime - curTime < 300) {

        console.log(stopTime - curTime, 'ms')

        ++diffNumb

      } else break

    } else randomNumb++

  }

  let createArtilceRes = await createArticle({

    data,

    user,

    previousHash,

    hash,

    local,

  })

  message.destroy()

  if (createArtilceRes.data) {

    message.success('文章区块上传成功')

    dispatch({

      type: actionTypes.CREATE\_ARTICLE,

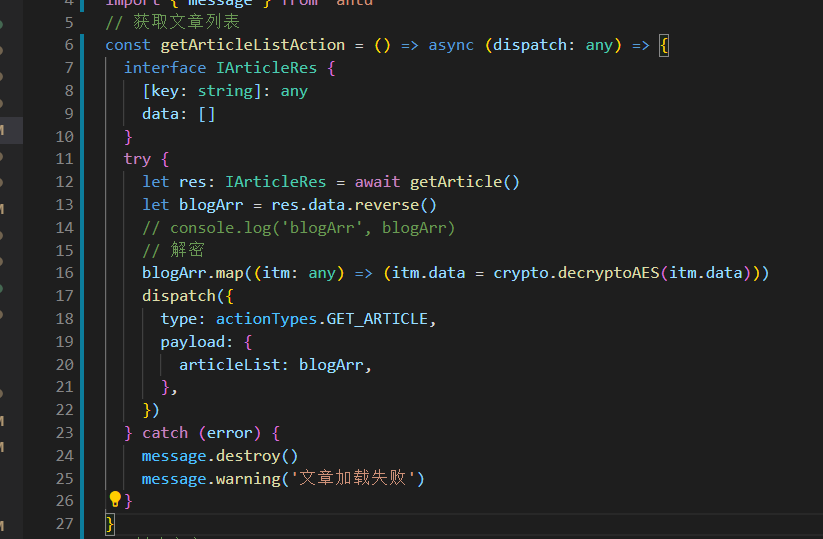
      payload: createArtilceRes.data,

    })

  } else message.warning('文章创建失败')

}

创建文章

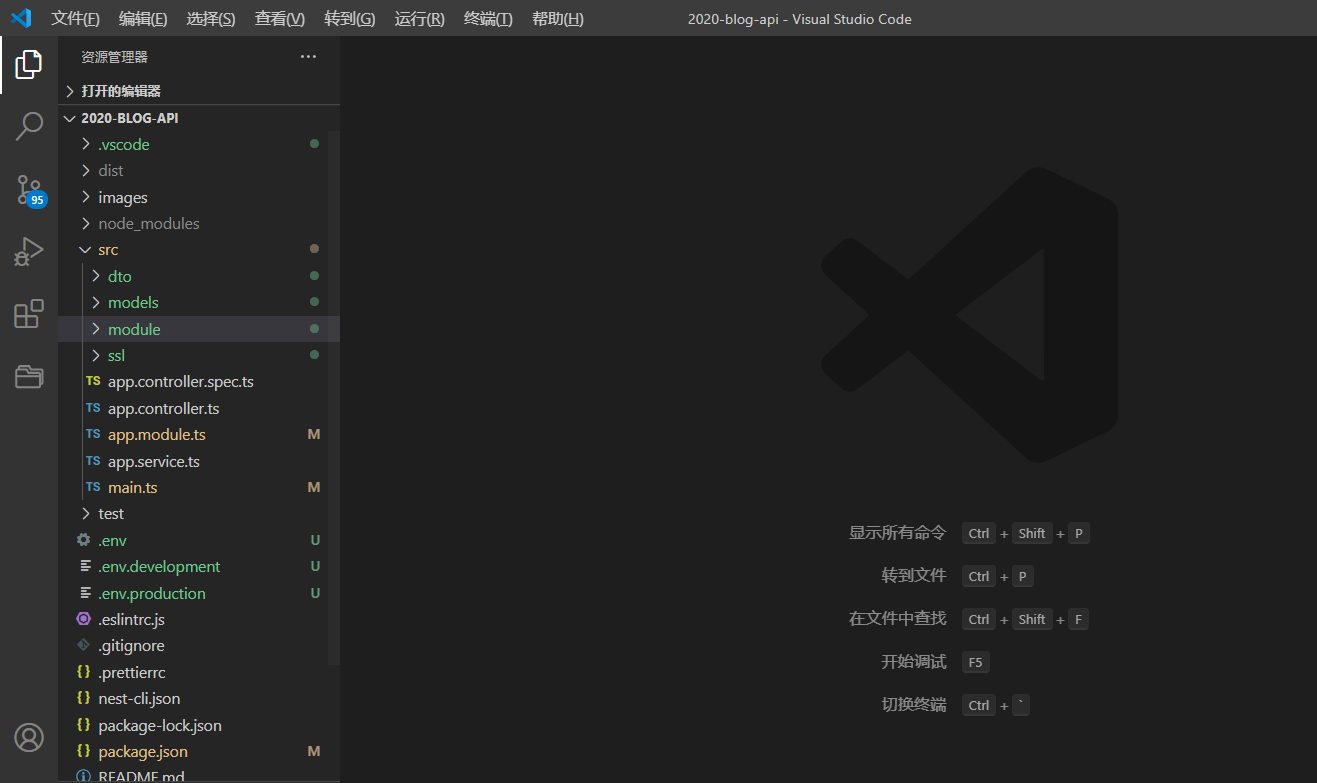


获取文章

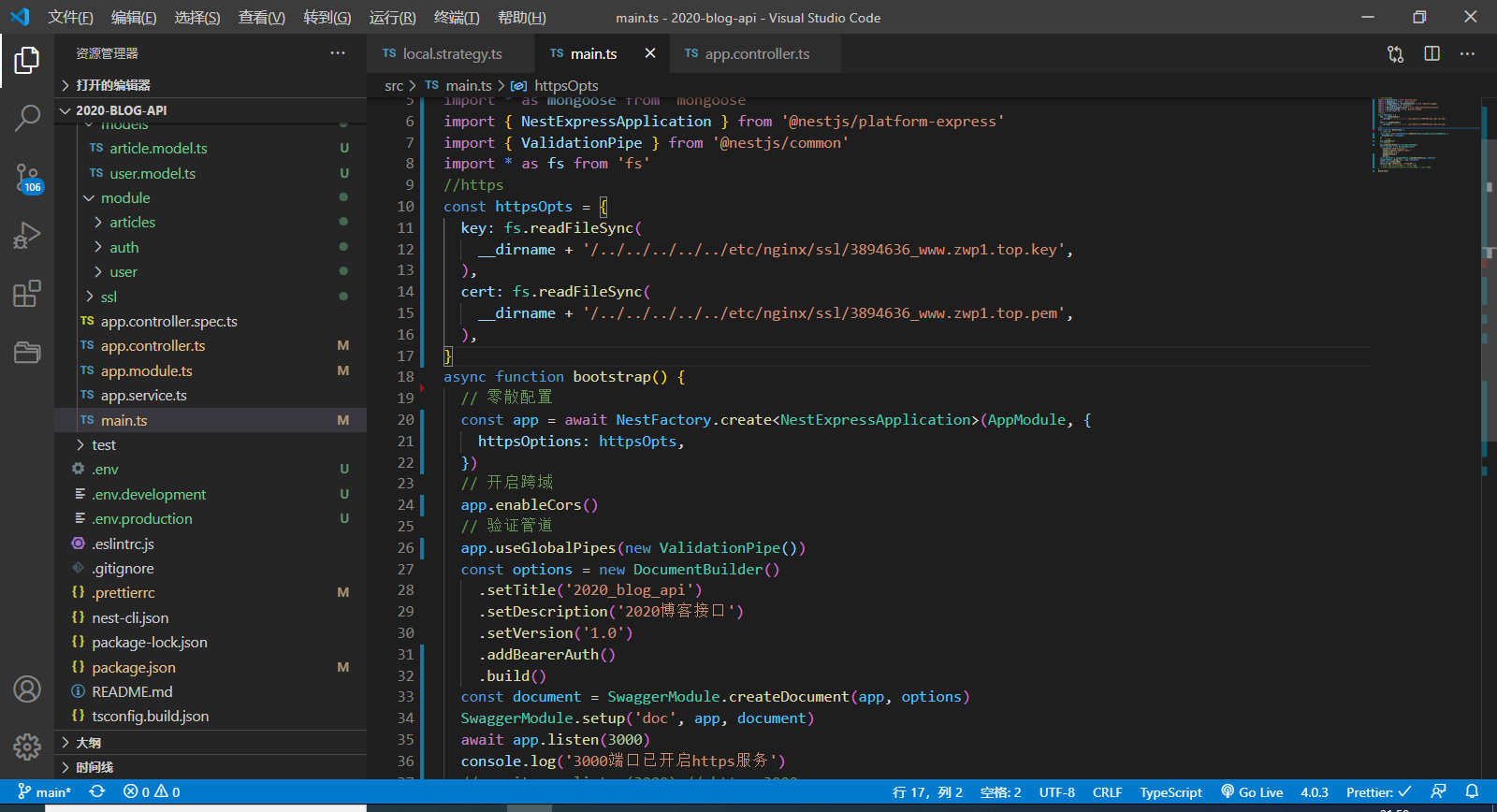
## 后端

### 代码

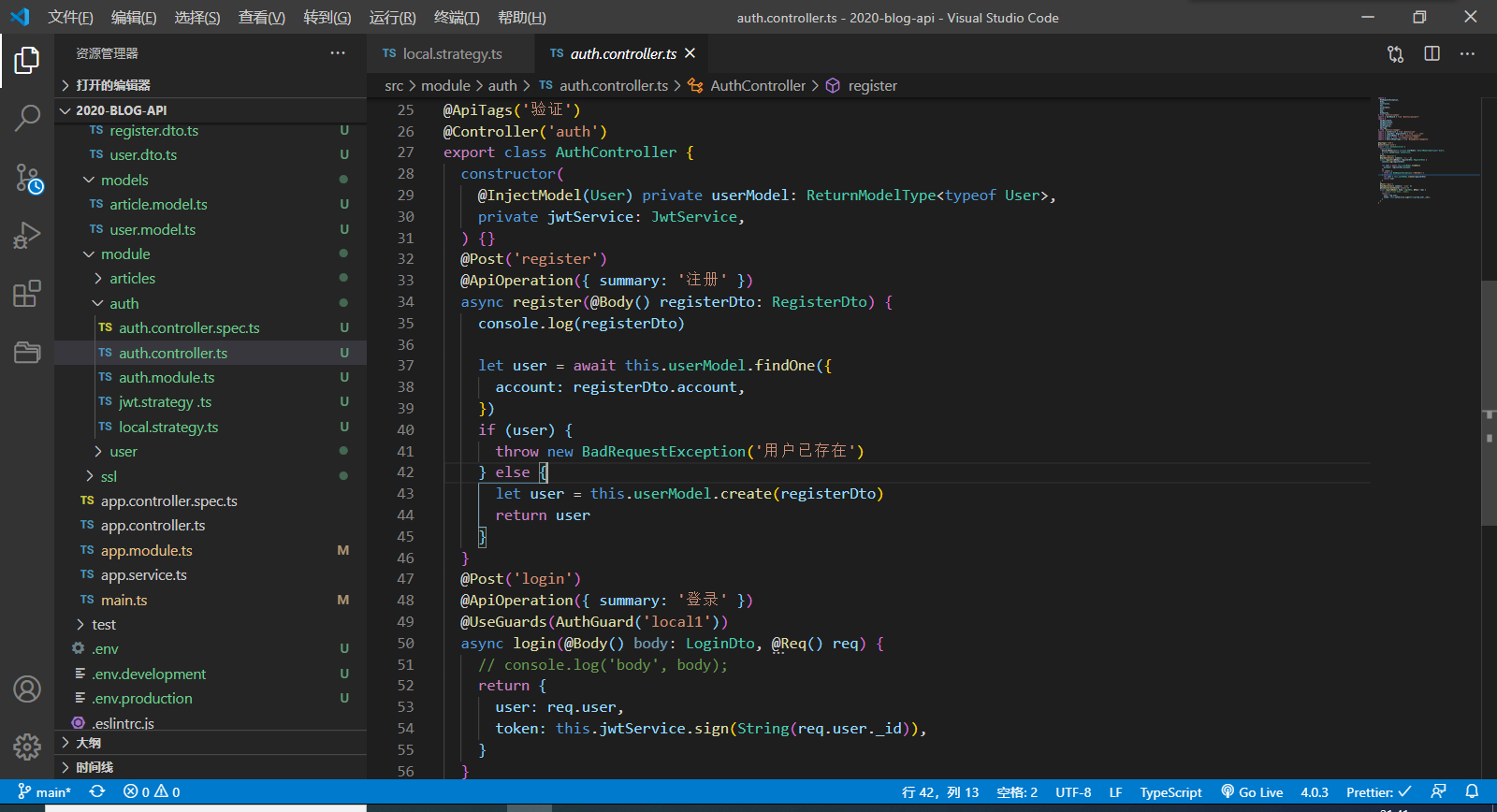
初始化项目：



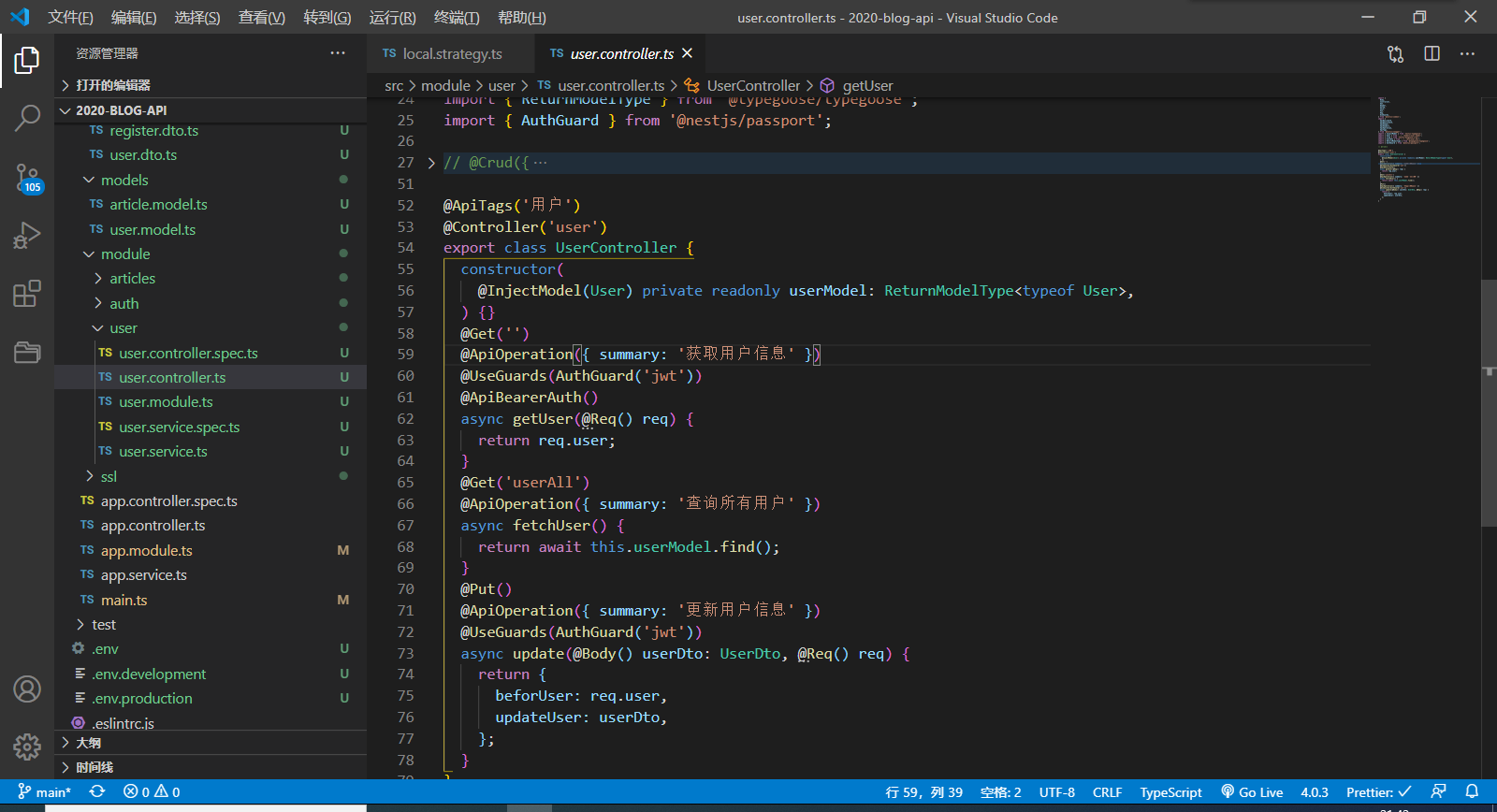
项目结构



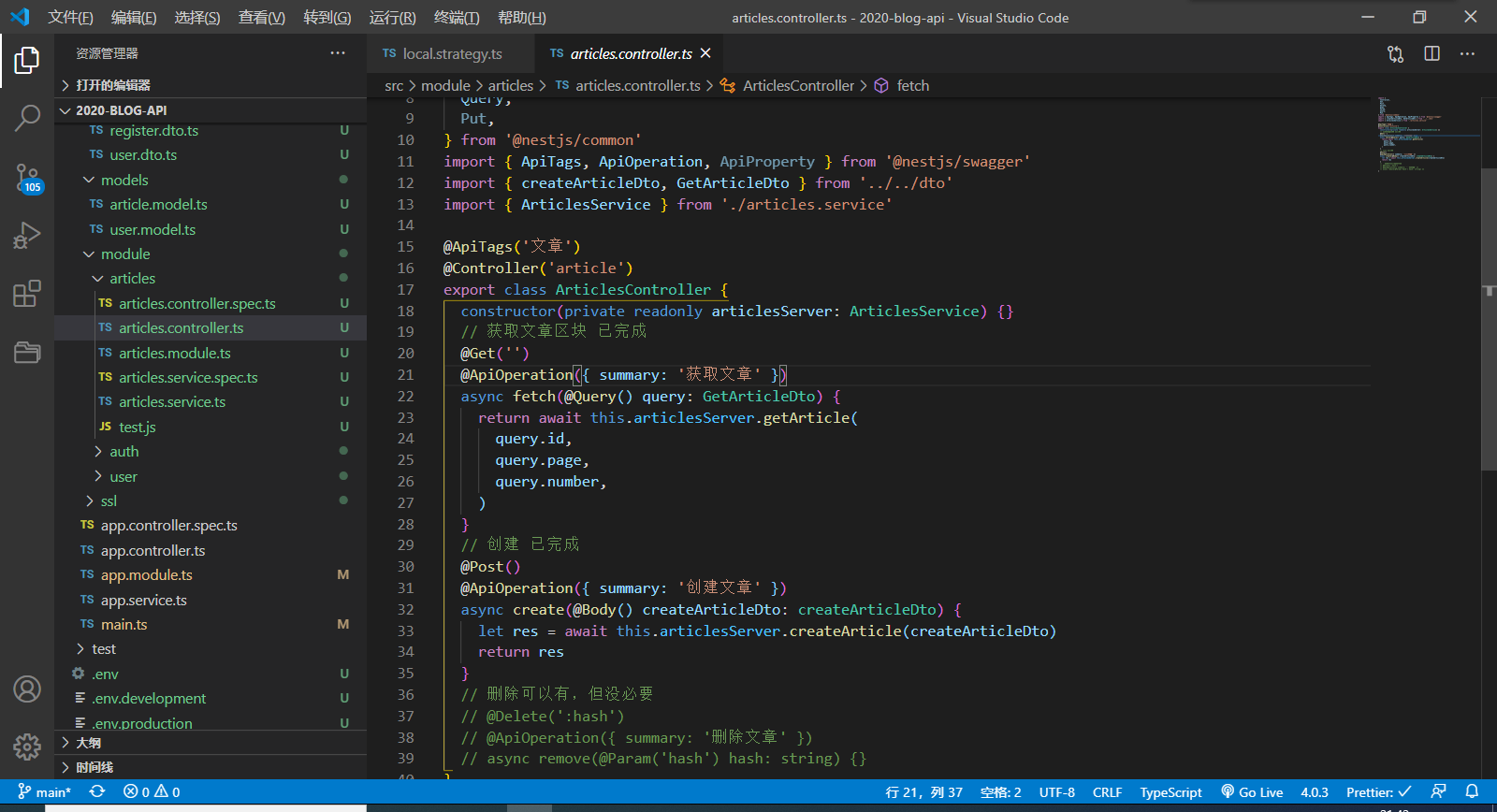
入口文件



Auth模块



User模块



文章模块

// 验证区块正确性（是否被篡改过）

  async verifyArticle(): Promise<Boolean> {

    // 三个库对比

    var res: any

    res = await this.blogModel.find()

    let blog = res

    res = await this.blogBackupAModel.find()

    let blogA = res

    res = await this.blogBackupBModel.find()

    let blogB = res

    console.log('blog', blog)

    if ((blog.length === 0, blogA.length === 0, blogB.length === 0)) return true

    // 过滤唯一\_id

    blog.map(itm => (itm.\_id = null))

    blogA.map(itm => (itm.\_id = null))

    blogB.map(itm => (itm.\_id = null))

    // 获取诚实节点

    let dbArr = [blog, blogA, blogB]

    let modelArr = [

      this.blogModel,

      this.blogBackupAModel,

      this.blogBackupBModel,

    ]

    // let dbArr = [blog, blogA, blogB, 1,2,3,4,5];

    // 获取数组长度一半以上的相同的值的下标

    let maxVal = [] // 正确数组的下标

    let m = new Map()

    // 通过相同数据结构加密出来的hash值相同判断是否同一数据库

    dbArr.map((itm1, idx) => {

      itm1 = itm1.map(itm2 => JSON.stringify(itm2))

      let hashItm = SHA256(itm1.join(',')).toString()

      if (!m.get(hashItm)) {

        m.set(hashItm, [idx])

        if ((maxVal.length = 0)) maxVal = [idx]

      } else {

        let arr = m.get(hashItm)

        m.set(hashItm, [...arr, idx])

        if (arr.length >= maxVal.length) maxVal = [...arr, idx]

      }

    })

    // 打印诚实节点数组的下标

    console.log(maxVal)

    // 如果诚实节点数量大于总数量

    if (maxVal.length > Math.floor(dbArr.length / 2)) {

      // 有个别叛逆节点

      if (maxVal.length !== dbArr.length) {

        // 从诚实节点中复制数据给叛逆节点

        let trueMudel = modelArr[maxVal[0]]

        // 存下正确数据

        let trueData = await trueMudel.find({})

        // 过滤叛逆节点

        let badNodeArr = modelArr.filter((itm, idx) =>

          maxVal.indexOf(idx) == -1 ? true : false,

        )

        console.log('叛逆节点', badNodeArr)

        // 恢复数据

        // modelArr.map

        badNodeArr.map(async item => {

          // this.blogModel.remove({}) 删除叛逆节点所有数据

          let removeRes = await item.remove({})

          console.log('removeRes', removeRes)

          let insertRes = await item.insertMany(trueData)

          console.log('insertRes', insertRes)

          console.log('数据恢复')

          // await trueData.map(trueItm => item.insertMany(trueItm))

        })

      }

      return true

    } else {

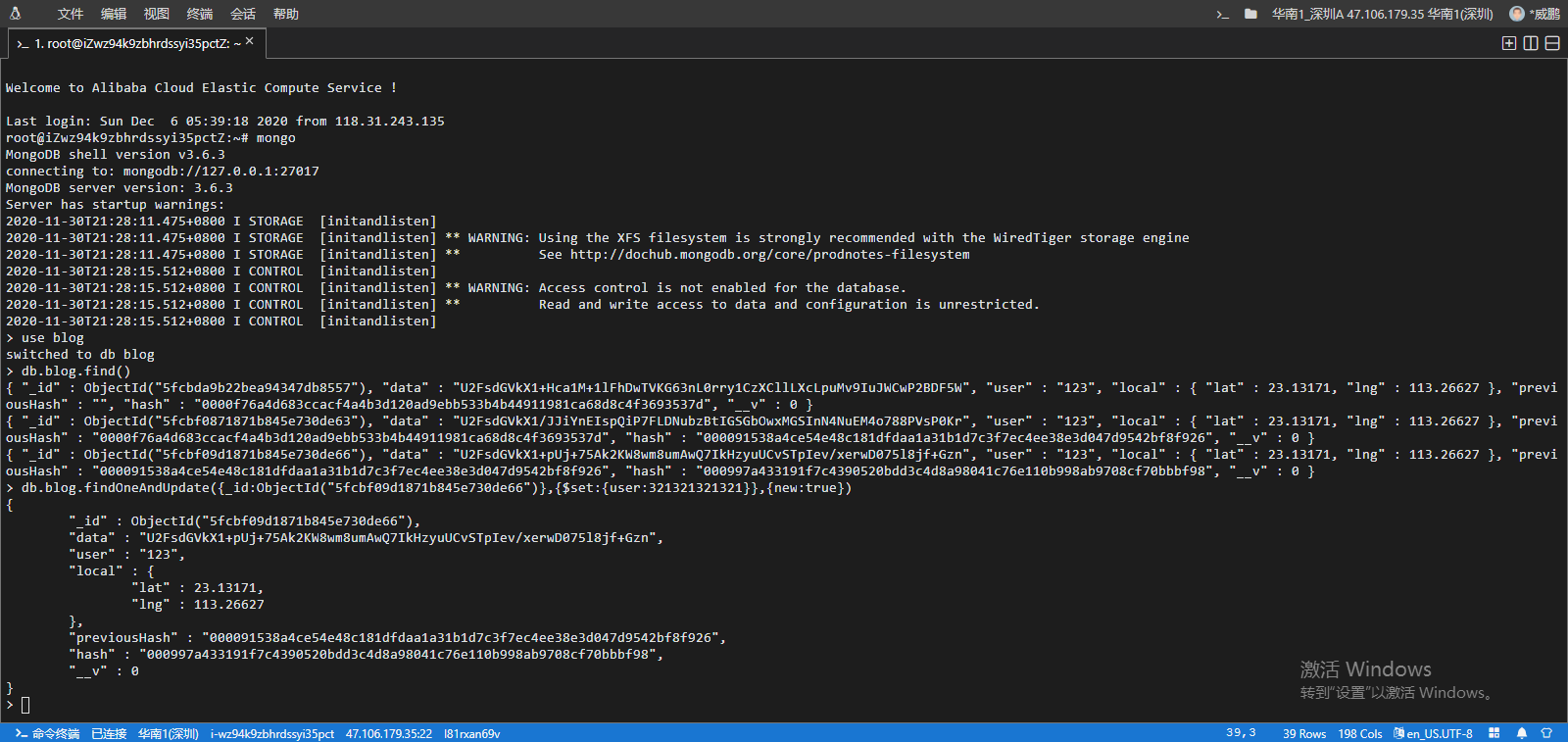
      console.log('数据库异常请检查')

      return false

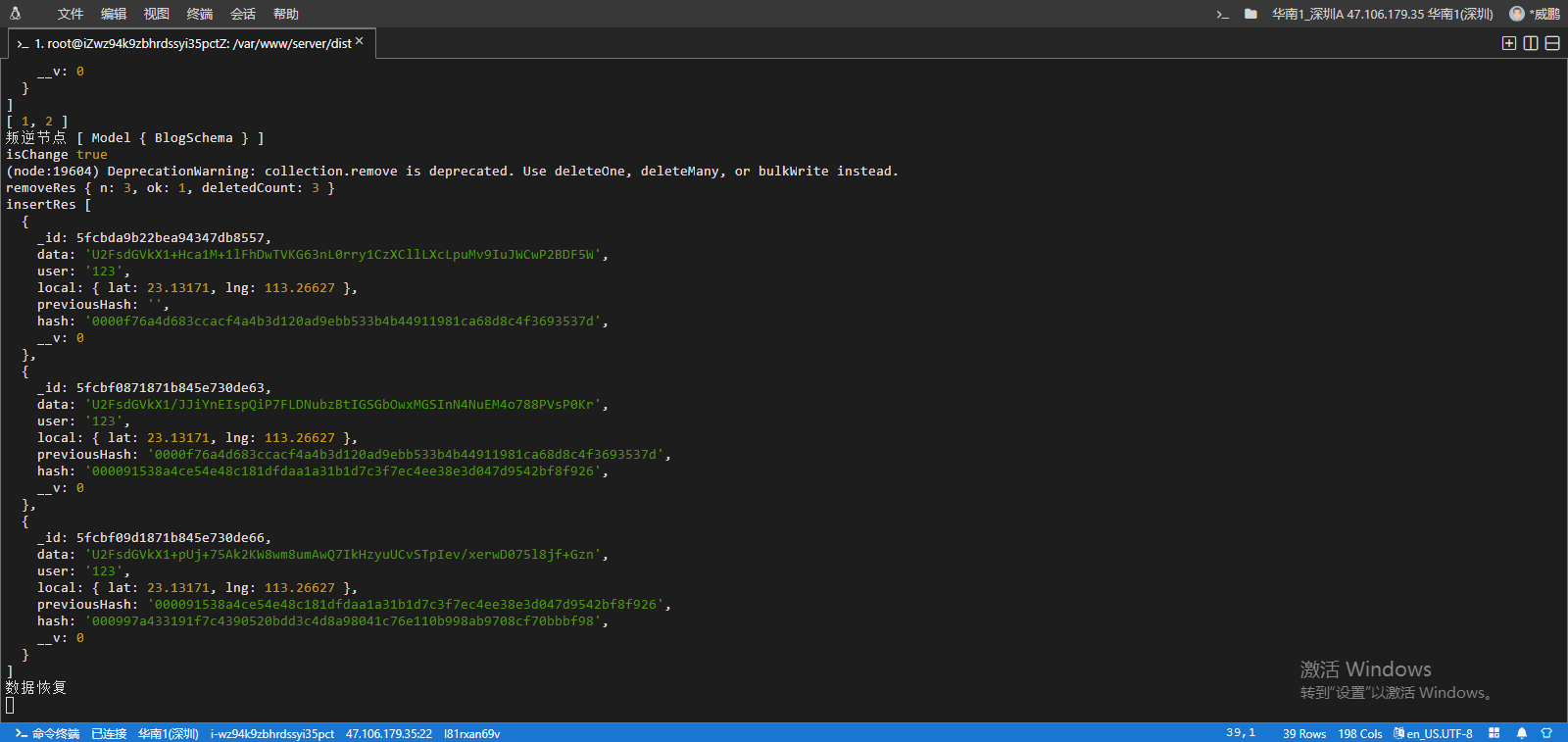
    }

  }

验证算法（重要功能）

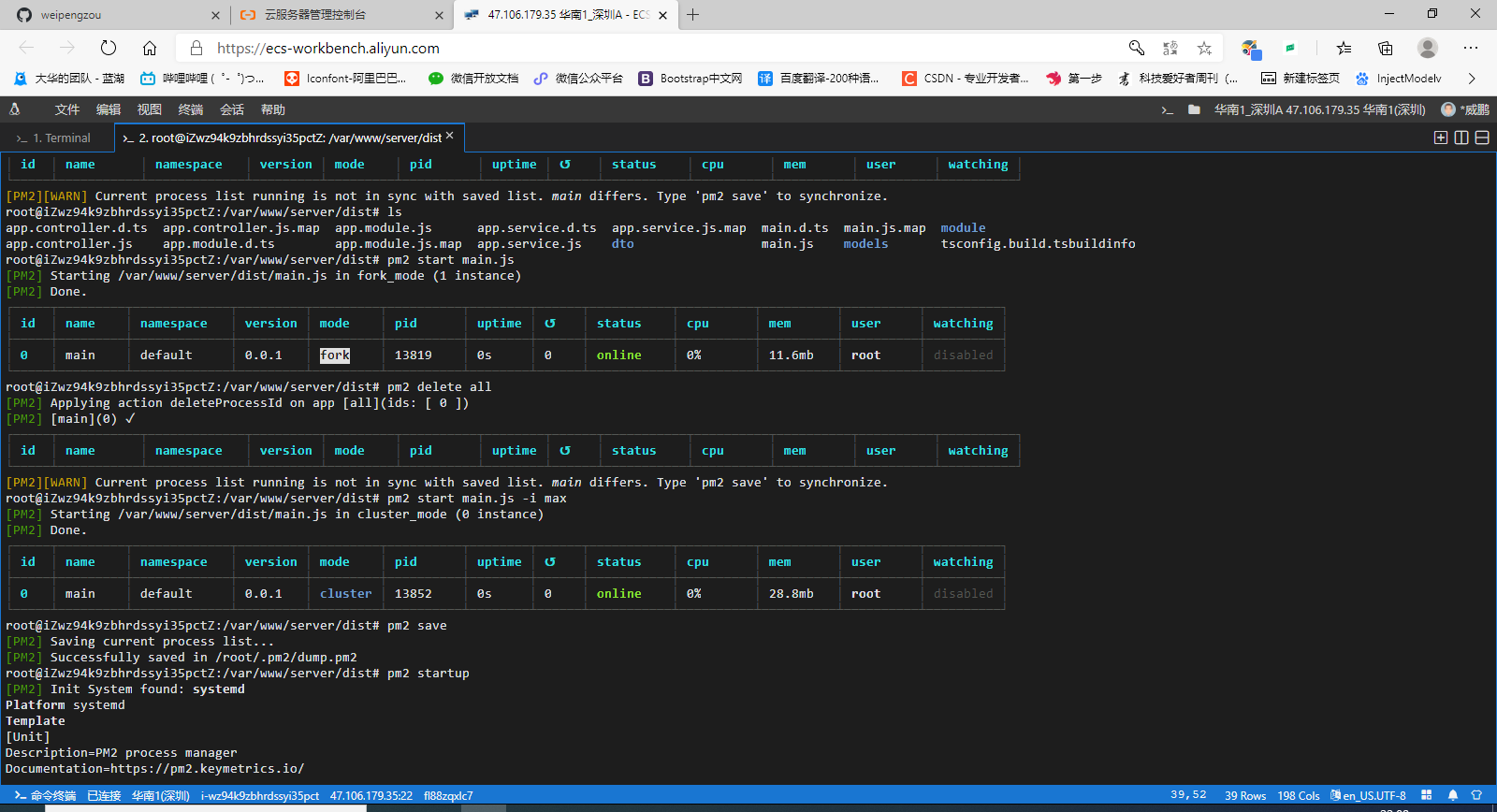


改数据测试

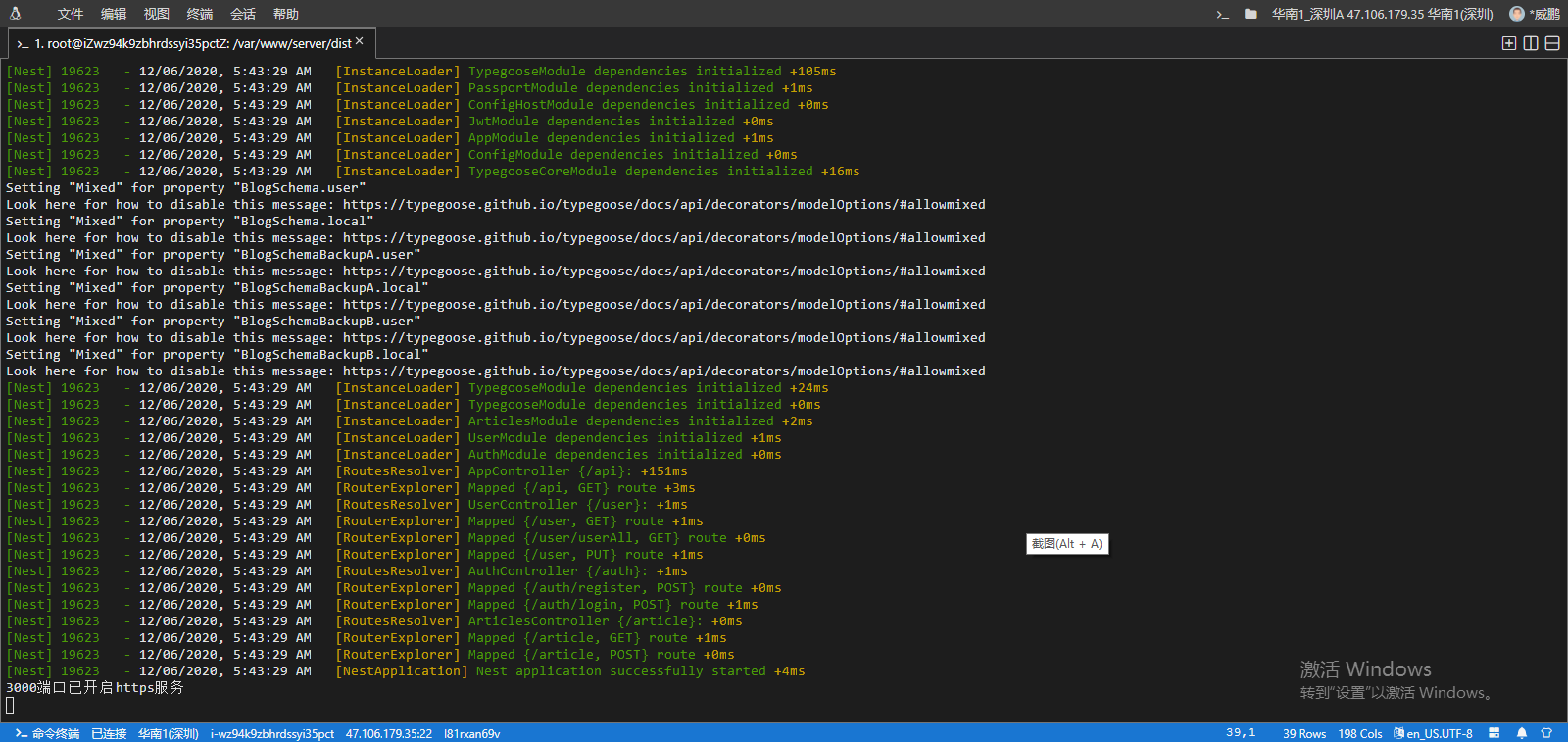


检测到数据变化，自动修改

## 服务器



开启服务&设置负载均衡



运行情况

# 总结与展望

该项目难度在于文章的加密和保证文章的不可篡改特性，使用了类似拜占庭将军问题的诚实节点特性。

使用多个数据库备份文章链，在前端页面设置了计算哈希值必须大于300ms的算法限制，定时查询和对比数据库，保证数据源的正确性。

多个数据库牺牲了代码效率，但是能分区管理验证，防止被入侵者一锅端的情况。

基于区块链技术的特性，文章链会以数据最多的链作为准。

例子：

当前有6个数据库，每个库只有200条文章数据，每条需要>500ms计算出hash值，那么需要获取4个服务器的权限，代码文件的私钥，再用电脑计算8分钟才能修改一条数据，并且8分钟内无人提交文章。

该项目用户操作的数据使用对称加密（aes）、散列加密（hash），在数据传输中均已密文传输，数据在客户端渲染DOM节点，搜索引擎爬虫无法获取到网站页面数据，在绝大多数情况下保证用户隐私问题。

现在大多数人都沉浸在大数据时代下带来的便捷福利，往往忽视了自身在大数据分析下的危险，往往按下一个按钮或在某个应用下停留的时间过长，就有可能被算法和埋点获取到用户画像，通过绑定手机、社交软件等方法给人推送商品或推荐业务，我希望能通过该项目帮助、提醒和启发到能多人和开发者去关注隐私方面的问题。

参考论文：《Bitcoin: A Peer-to-Peer Electronic Cash System》（Satoshi Nakamoto）