中国大学生计算机设计大赛



软件开发类作品文档简要要求

作品编号：　　　　　　2022003293

作品名称：　　　　　　基于容器技术的云OJ平台

作　　者：　　　　　　熊舟桐 徐荣鑫 杨琦琛

版本编号：　　　　　　1.0.1

填写日期：　　　　　　2022/4/8

填写说明：

1. 本文档适用于**所有**涉及软件开发的作品，包括：软件应用与开发、大数据应用、人工智能应用、物联网应用；
2. 正文一律用五号宋体，一级标题为二号黑体，其他级别标题如有需要，可根据需要设置；
3. 本文档为简要文档，不宜长篇大论，简明扼要为上；
4. 提交文档时，以PDF格式提交本文档；
5. 本文档内容是正式参赛内容组成部分，务必真实填写。如不属实，将导致奖项等级降低甚至终止本作品参加比赛。

目 录

[第一章 需求分析 3](#_Toc6230586)

[第二章 概要设计 4](#_Toc6230587)

[第三章 详细设计 5](#_Toc6230588)

[第四章 测试报告 12](#_Toc6230589)

[第五章 安装及使用 15](#_Toc6230590)

[第六章 项目总结 1](#_Toc6230591)7

# 需求分析

1.1 设计初衷

现在市面上，OJ产品不少，像力扣、PTA。他们针对的用户为整个编程社区，人员水平参差不齐，用户多而杂，对于低年级同学或是初接触编程的同学来说，很难去区分和获取有价值的信息，或对于一些非计算机专业的学生，没有必要去深入算法研究，本作品提供一个更加简洁且纯粹的判题平台和综合交流空间，在能够有效的锻炼学生编程思维的同时减少学生压力以及缓解学科焦虑。并促进形成和谐、进步的校内技术交流社区环境。

1.2 作品简介

参赛作品是一个面向校内学生综合技术学习的平台，以云判题为核心，专注于基础编程知识的掌握；提供多元化的探讨、交流空间；另外设计了HTML画板以供休闲之需。在如今以大数据和人工智能为代表的技术普及的时代，各行各业都或多或少的需要相关的人才，在高校的人才培养中体现出来的就是很多专业，尤其是工科类专业的学生都开始学习编程技术。甚至在许多城市的中小学都开始开设人工智能相关的编程课程。本作品主要服务于学习编程技术的初级学员，包括但不限于高等学校低年级的学生。作品以线上判题为主要功能，旨在打好学生的编程基础，为大家提供一个良好的校内交流平台。

1.2.1 主要功能

(1)线上判题模块

(2)学习讨论模块

(3)前端画板模块

(4)用户管理模块

(5)数据统计模块

1.2.2 优势和创新点

(1) 受众广。作品主要是服务于学习编程技术的初级学员，如高等学校低年级的工科类专业学生，想锻炼编程思维、打好编程基础的初高中学生等。

(2) 效率高，鲁棒性强。在传统的线上判题，即黑盒测试的基础上，使用了更为高效的云容器技术，有效地提高了判题效率和稳定性。

(3) 平台功能丰富，具有可扩展性。平台除主要的判题需求，还有相应的评论区、讨论区、组队咨询和前端页面在线编写功能，有利于形成良好的校内编程社区环境，同时不失为一个良好的学习交流平台。未来还可以根据需求继续增加更多有意义的功能。

# 概要设计

整个云OJ社区平台总体分为五大模块：最主要的判题模块、用户信息管理模块、休闲讨论模块、后台数据统计模块、前端画板模块。模块设计图如图1所示：

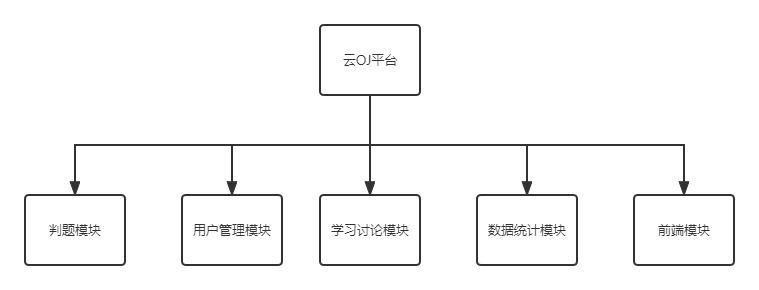


图1 总体模块图

OJ的主要核心功能是判题，即在线编译执行用户代码完成某项任务并做出判断。判题核心功能流程图如图2所示：

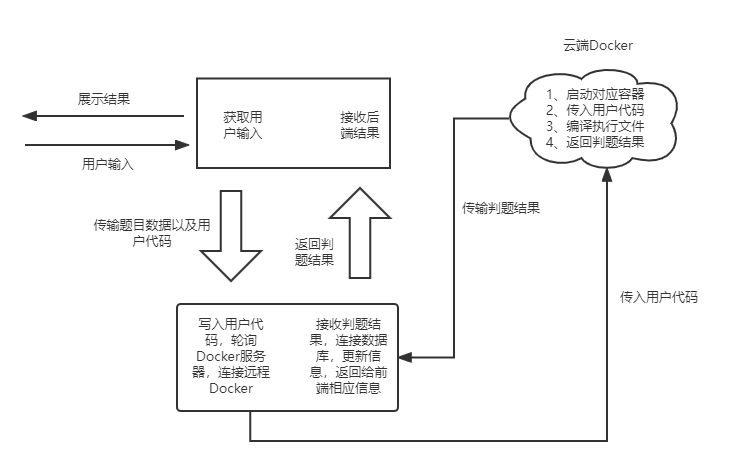


图2 核心功能流程图

在架构选择方面，采用经典的MVC架构，前后端分离开发部署：其中视图层使用Vue编写；后端使用Java技术栈SpringBoot及其相关组件；在数据库方面，使用开源关系型数据库MySQL进行用户数据及判题数据的存储，另外使用开源非关系型数据库Redis存储点赞、订阅等实时性较强的数据；前后端交互通过Nginx实现负载均衡、跨域请求等操作。架构图如图3所示。

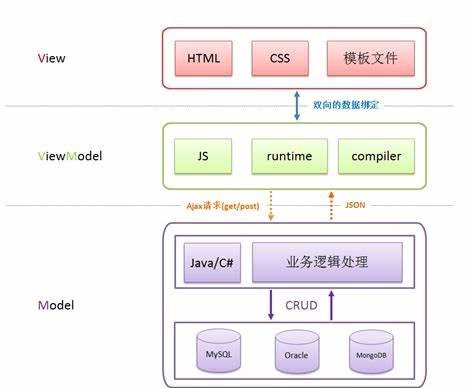


图3 架构图

# 详细设计

3.1 界面设计

前端页面使用Vue框架进行开发，单页面应用，使用开源组件Element-UI进行美化，极少使用jQuery，DOM操作几乎没有，页面反馈迅速，界面简洁优美、一目了然。部分图示如图4-图6所示。

3.2 数据库设计

主要使用关系型数据库MySQL对平台数据进行储存。在最初设计时，本着校内平台的想法，选择以“一校一库”的方式进行存储，即一个学校的数据绑在一个数据库中，每个数据库中（以NEUQ为例），包含用户数据、题目数据等共9张表，均遵循数据库第一、二、三设计范式，具体ER图如图7所示。

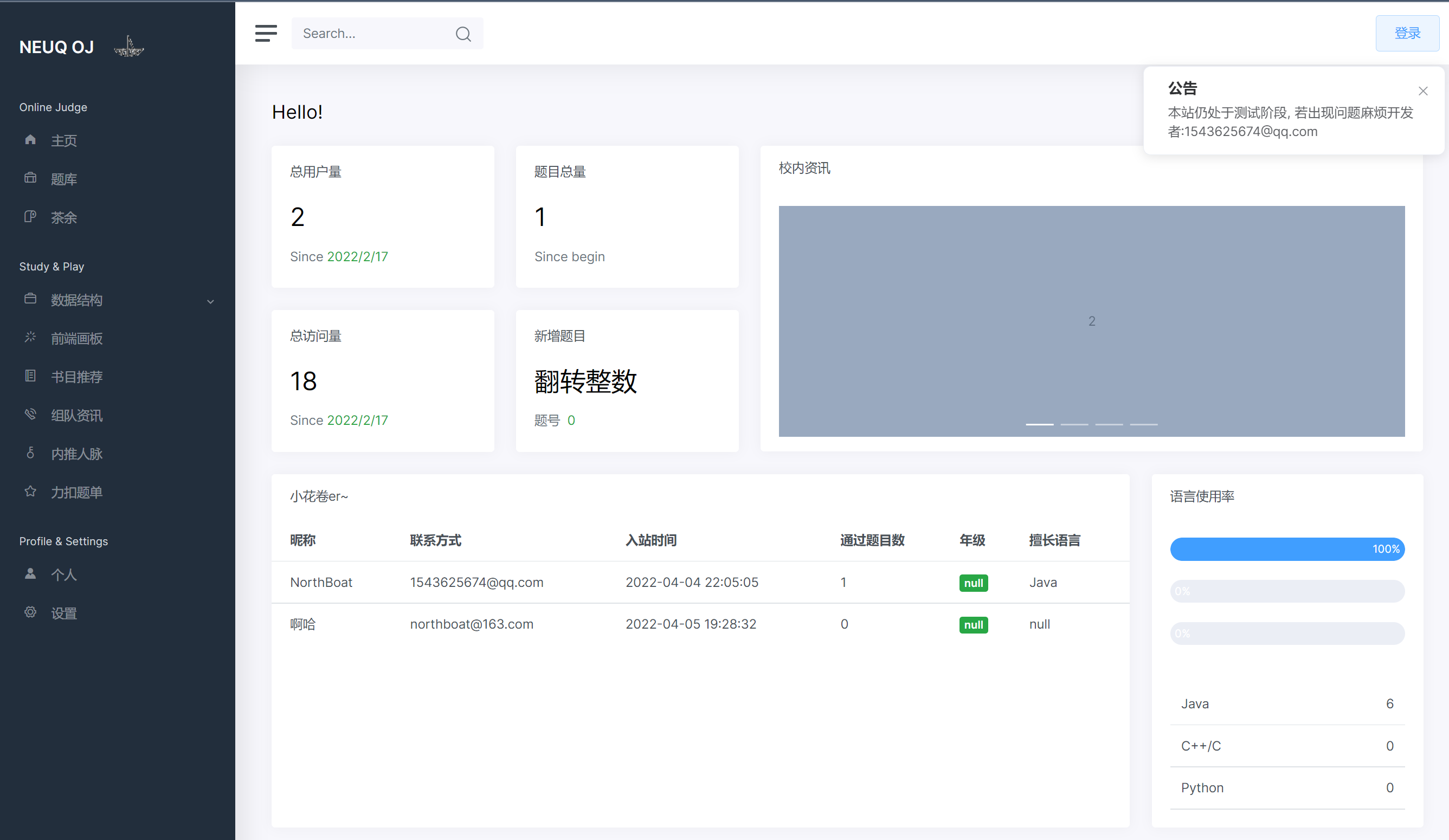


图4 平台主页

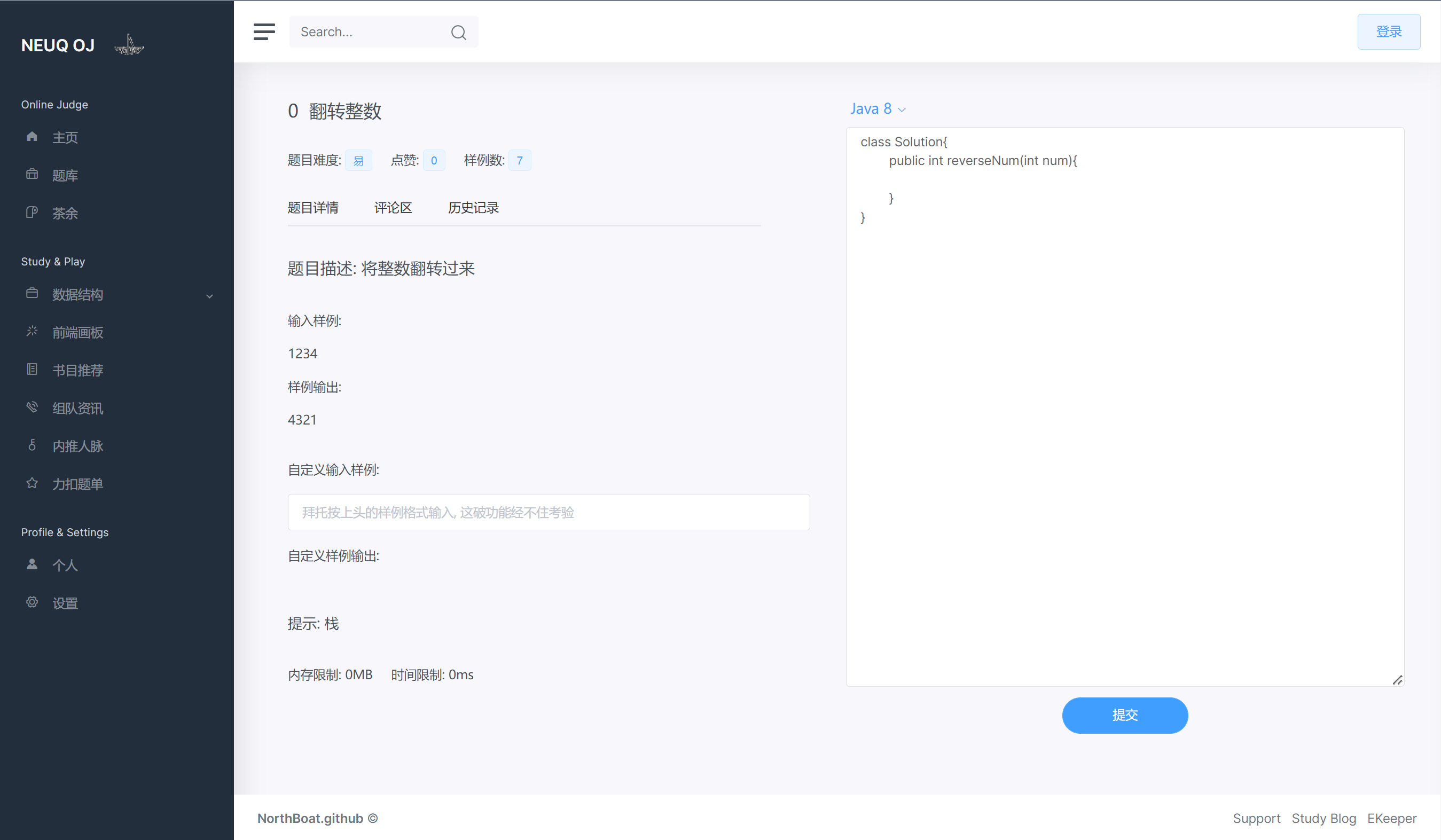


图5 判题页面

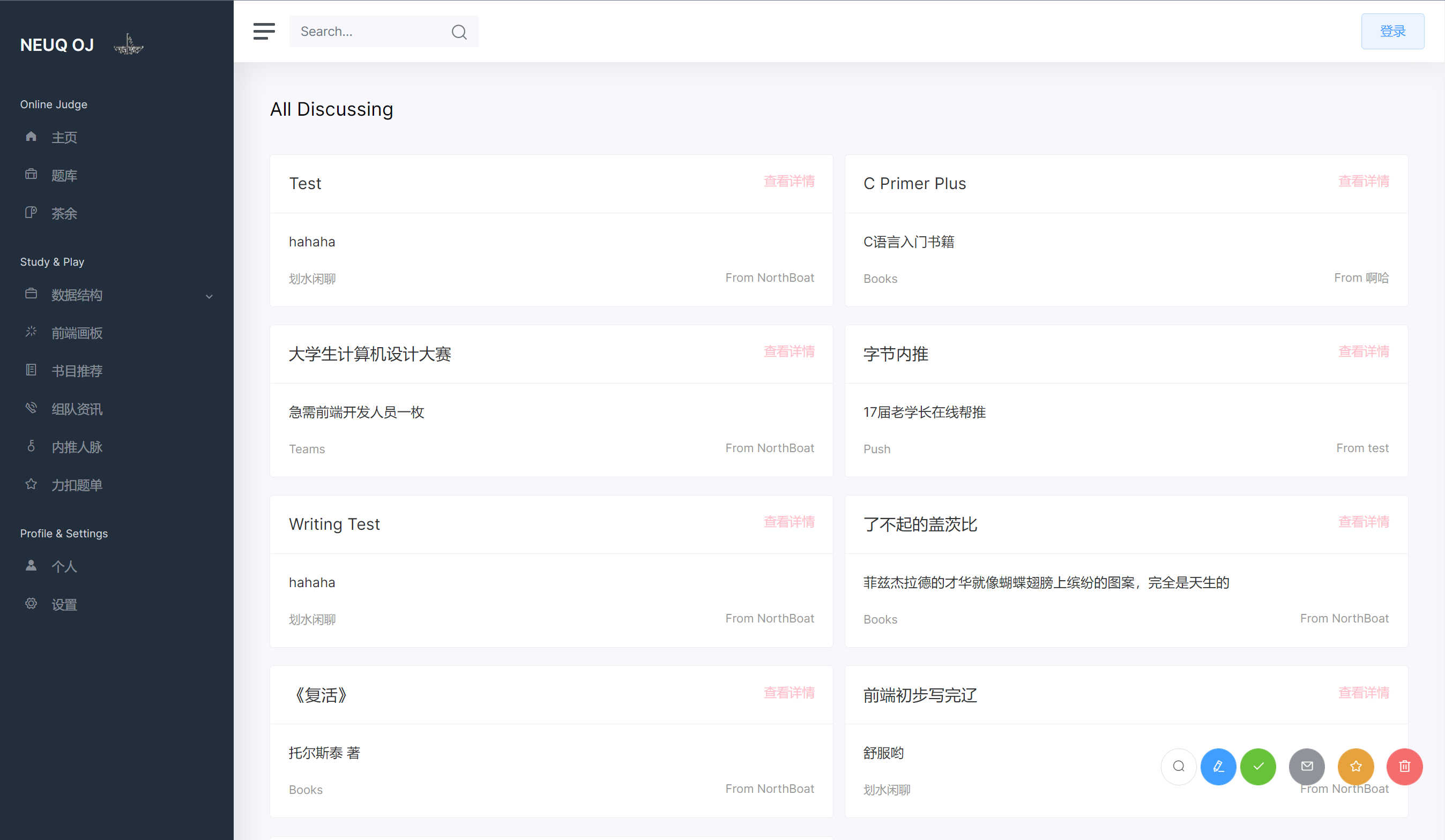


图6 讨论区

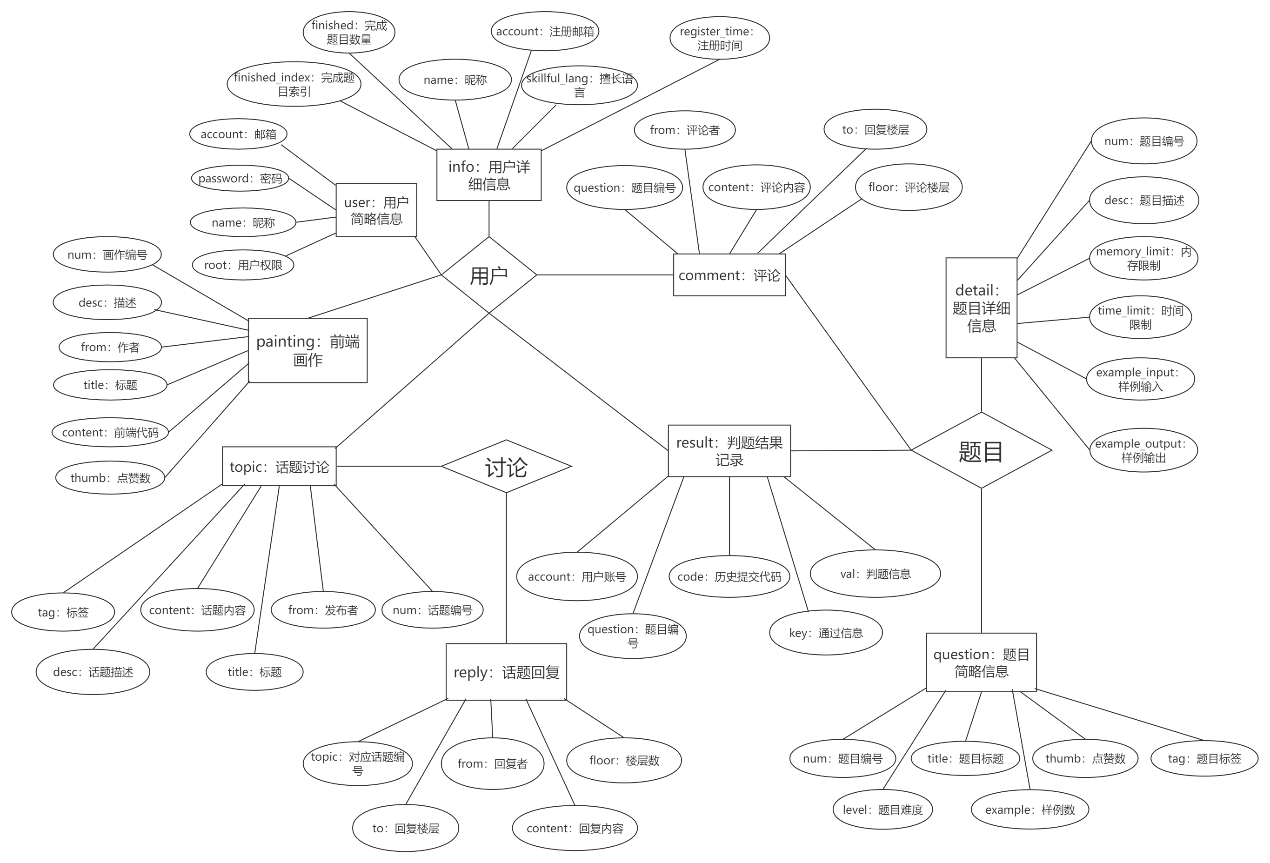


图7 数据库设计ER图

3.3 架构设计

采用经典的MVC架构，前后端分离部署，前端使用Vue技术框架，后端使用Java SpringBoot技术栈，分离部署造成的跨域请求使用Nginx进行反向代理处理，同时解决负载均衡问题。如图8所示：

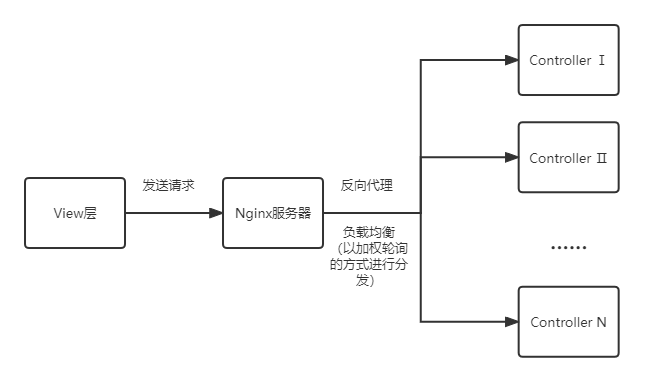


图8 架构设计

3.4 关键技术设计

判题是本平台的核心技术。使用Java调用Docker-Client API远程操控Docker服务器，传入用户代码编译执行，实现黑盒测试。大致判题流程图如图9所示。

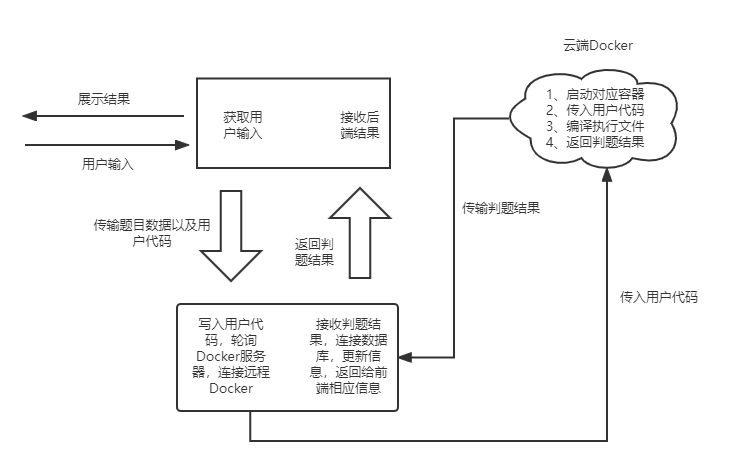


图9 判题流程图

在远程调用Docker时，在后端手动加了一层轮询，将请求分发到不同的Docker服务器，降低单个Docker服务器压力，充分利用云算力，提高判题效率和判题稳定性。如图10是其调用示意图，图11是实现的核心代码。

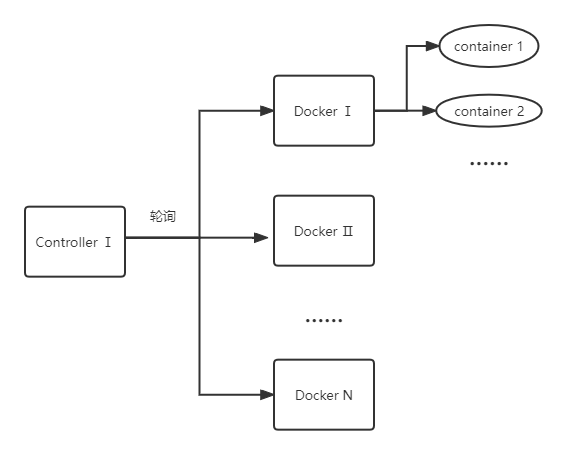


图10 轮询Docker服务器示意图

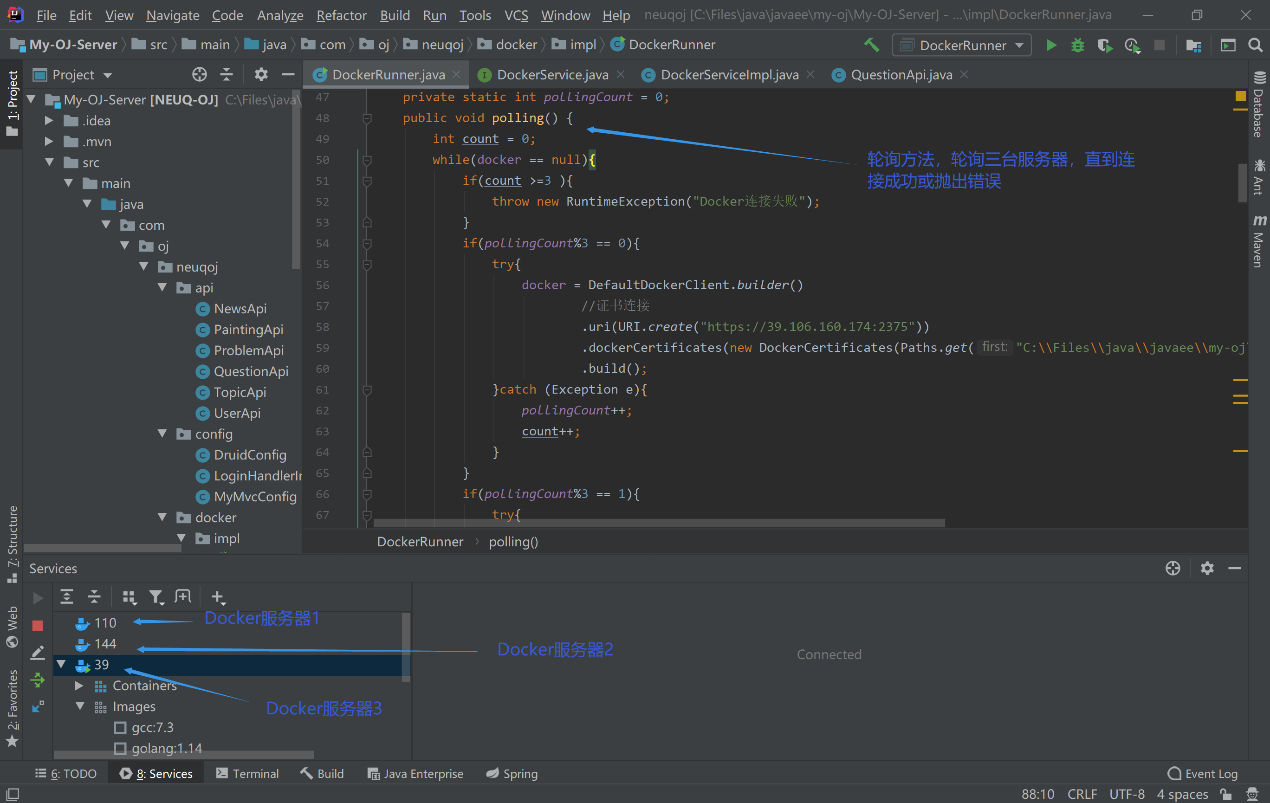


图11 远程操控Docker核心代码

Docker容器采用线程池的形式提供服务：一台Docker服务器配置以下两个属性：MaximumContainerSize, CoreContainerSize。其中，MaximumContainerSize为Docker允许的最大容器数，CoreContainerSize为活跃的Docker容器数。

当多请求同时打到同一Docker服务器时，会优先使用活跃的Docker容器，当请求数大于CoreContainerSize，服务器将提供新的容器，直到请求数大于MaximumContainerSize，这时Docker容器达到饱和，转而令请求入队，等待有空闲容器时出队。当请求数下降一定时间后，后端将自动销毁容器数量至CoreContainerSize。示意图如图12

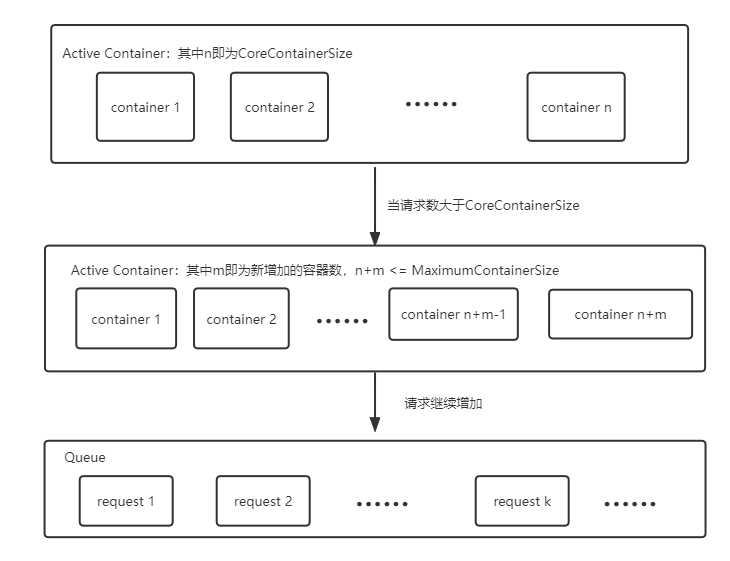


图12 请求队列

3.5 点赞功能设计

点赞这类数据实时性很强，选择用Redis储存点赞数据，以处理多线程、高并发、实时的要求，Redis中具体存储结构如图13所示。

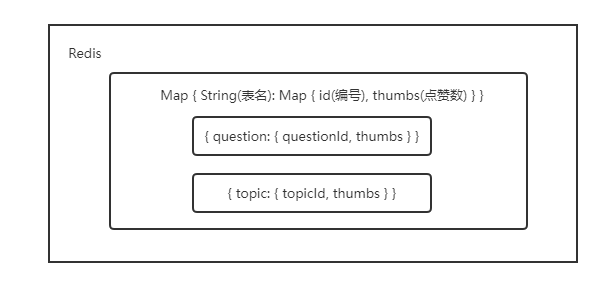


图13 点赞存储设计

3.6 HTML画板功能设计

用户在线编写HTML代码，平台提供Element-UI弹窗组件在线编译HTML文件并展示，为防止页面渲染过度造成服务卡顿，限制了HTML代码长度，如图14为渲染流程图。

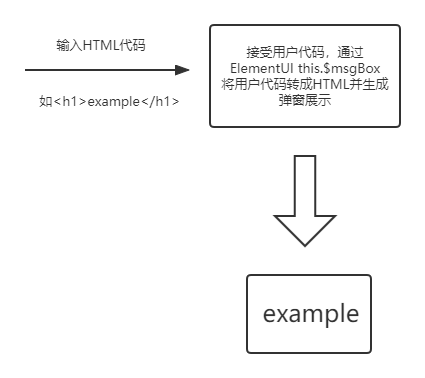


图14 渲染流程图

3.7 评论功能设计

评论的数据关联度很高，这里使用话题编号和评论楼层作为联合主键以保证每条评论的唯一性，并附上回复楼层的字段精准定位，方便用户交流，相应存储设计如图15所示。

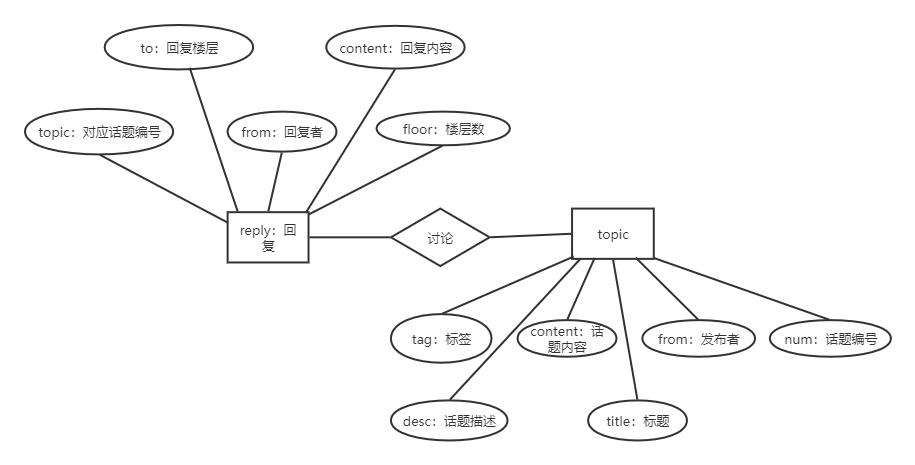


图15 存储设计

3.8 注册功能设计

验证码的时效性通过Redis数据库实现。用户提交注册请求后，后端生成六位随机验证码验证码，通过POP3/SMTP服务器将邮件发送到用户邮箱，同时将验证码以邮箱、验证码的键值对存入Redis，有效时间为10分钟，验证从Reids库中取相应数据即可。为防止恶意测试，前端限制验证码按钮功能60s发送一次。注册流程如图-16所示

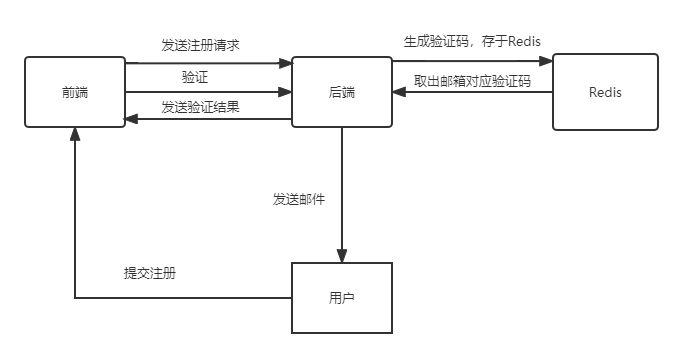


图16 注册流程图

# 测试报告

4.1 Docker测试

我们对Docker容器判题的性能进行高压运行测试，样例题目为一个简单的深度优先搜索算法，代码类型为Java8，服务器配置为1核2G，系统为CentOS7.6。在分别连续执行1、2、5、10、20、50、100、200、500、1000次的情况下，观察在高压下Docker服务器创建容器、停止容器、编译时长、执行时长与连续执行次数的关系，分析判题的效率以及稳定性，各种数据如图17-图19所示，其中时间单位均为毫秒。



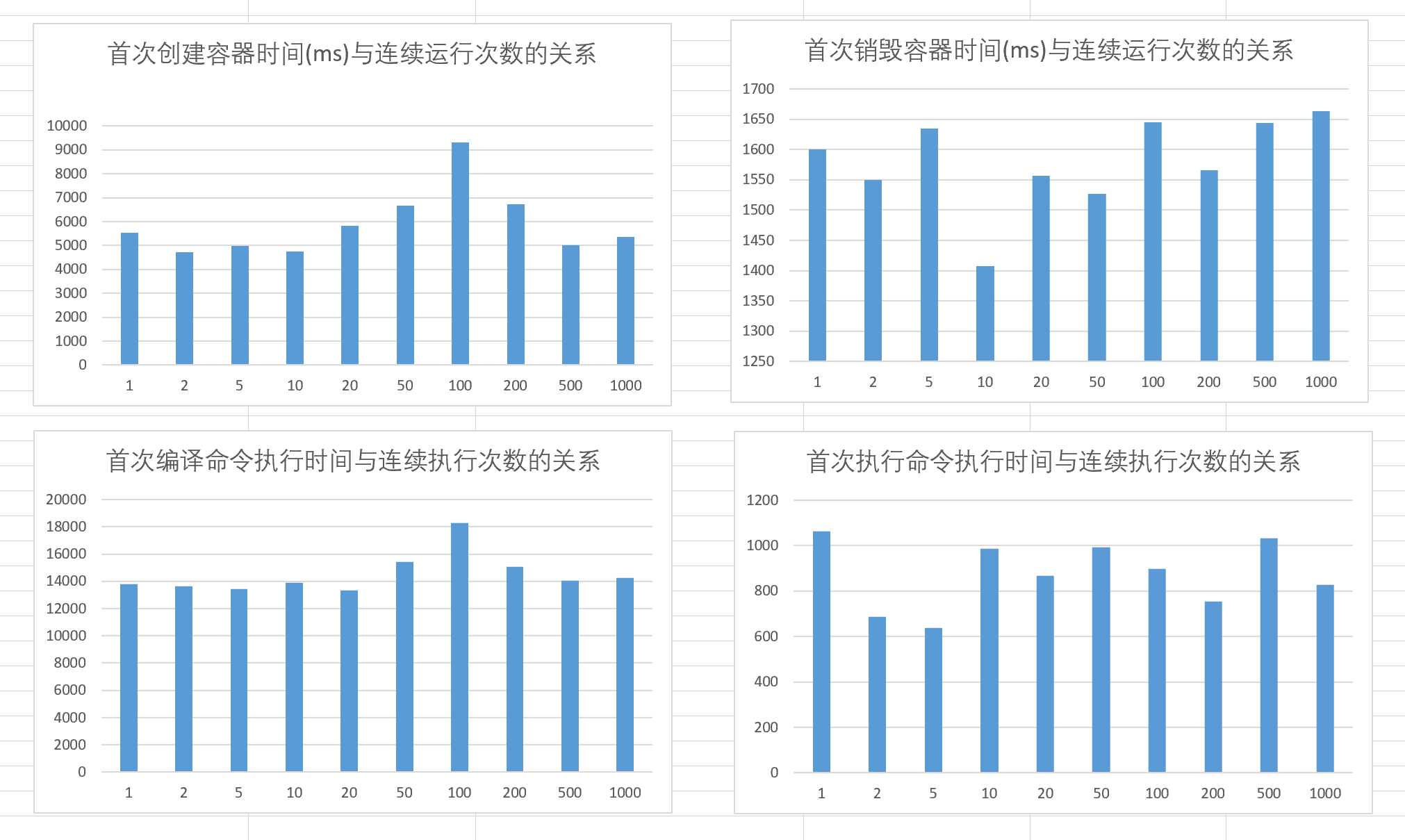
图17 Docker性能测试表格

图18 Docker性能测试柱形图

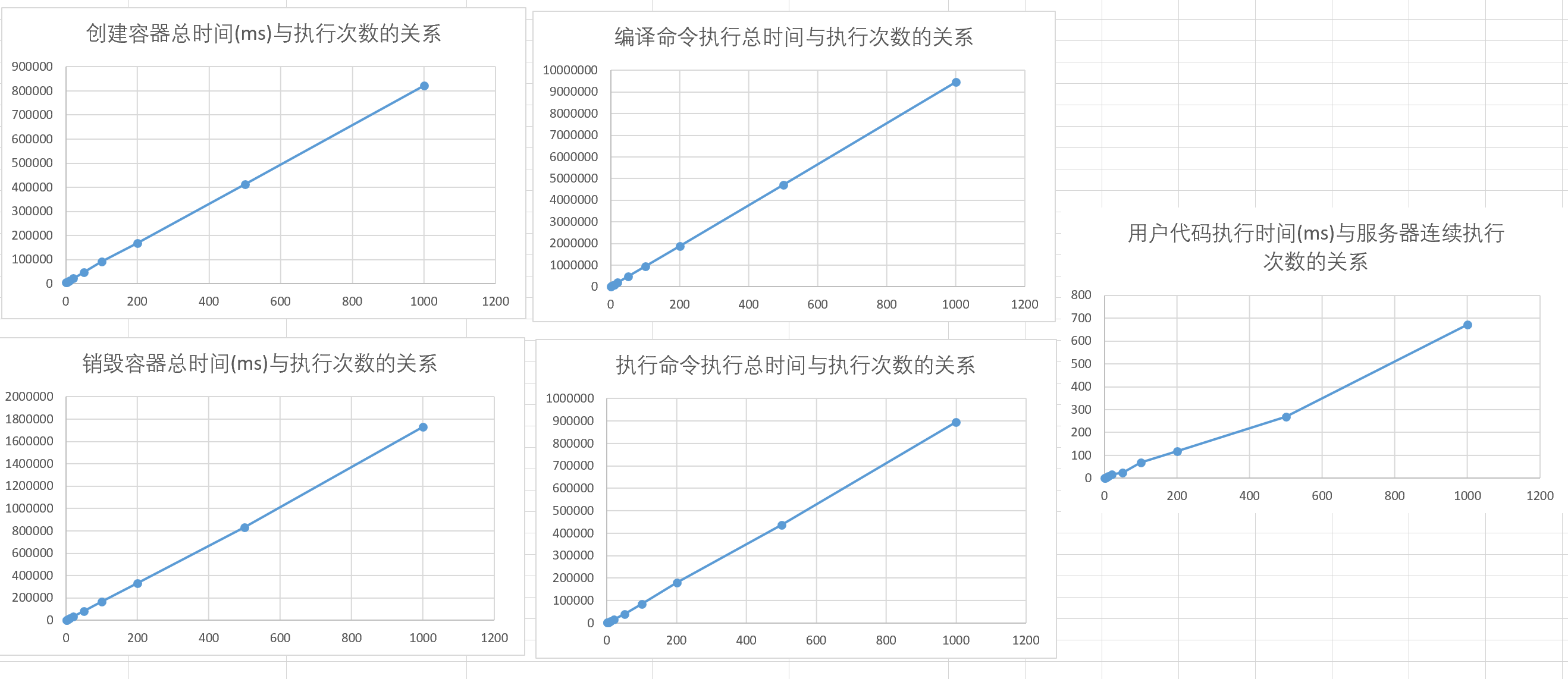


图19 Docker性能测试折线图

可以发现，在服务器高负荷连续运转下每条服务的运作时间在误差允许内保持一致，与运行次数呈正比例关系，不会因为服务器长时间运作造成明显性能下降的情况，并且对用户代码执行的时间几乎毫无影响，判题耗时的准确性很高；另外发现在首次运行时，创建、销毁容器时间约为之后每次的1.4倍，分析判断为Docker镜像构建时的缓存机制造成，参考文档：[Best practices for writing Dockerfiles | Docker Documentation](https://docs.docker.com/develop/develop-images/dockerfile_best-practices/)

Docker安全性试错，必须手动生成SSL证书并通过它远程连接Docker，若直接开放2375端口采用TCP直连，不法分子会有机会通过这一端口直接攻击你的服务器，包括但不限于挖矿病毒、DNS攻击，亲测百分百被攻击，只是时间问题。

4.2 数据库测试

关系型数据库方面使用阿里Druid数据源的SQL监控。后台监控程序部署在：<http://39.106.160.174:8089/druid>，该监控程序将对每条SQL从性能到安全进行自动化的统计分析，运行图如图20-图21

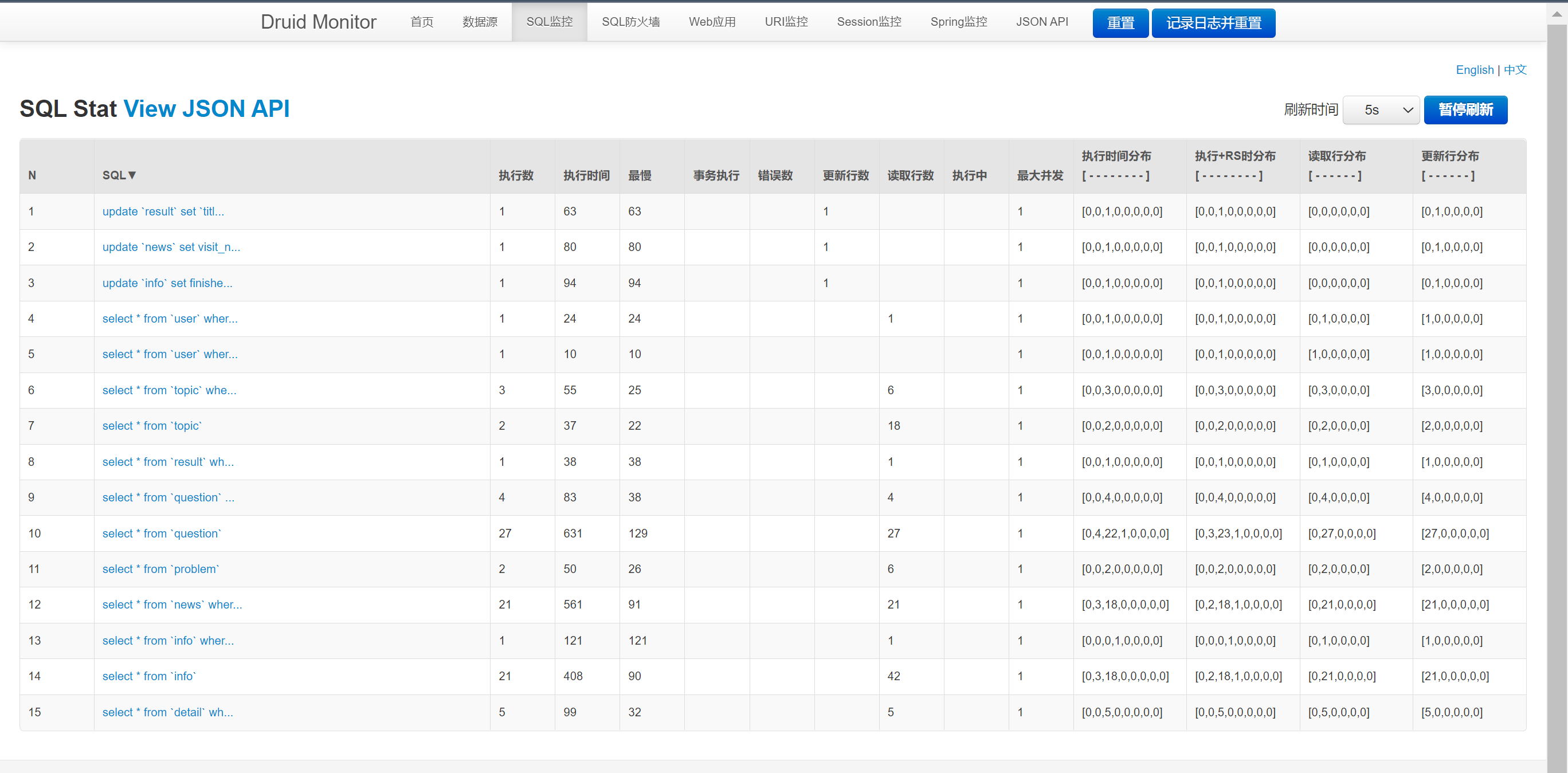


图20 SQL监控图1

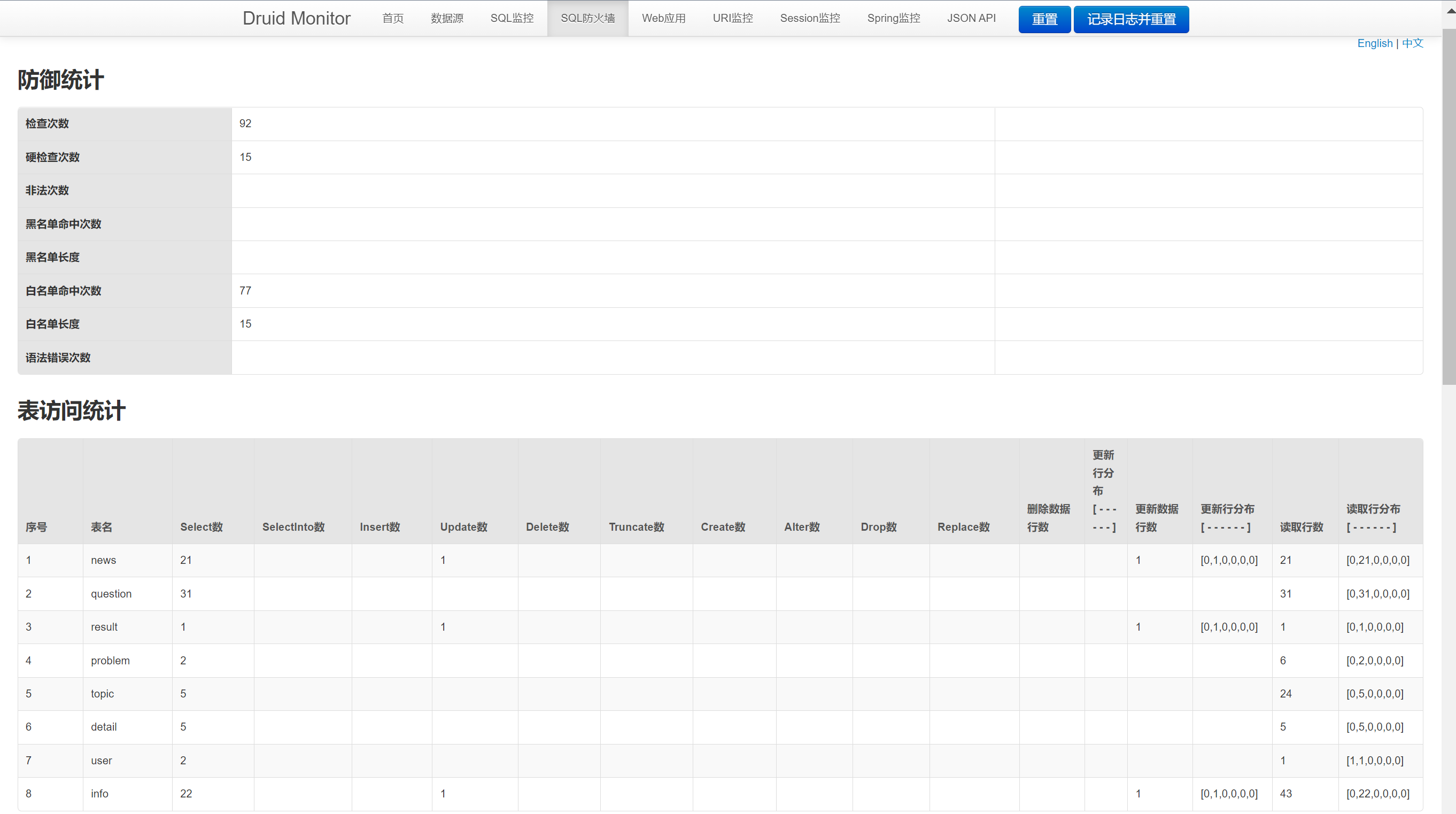


图21 SQL监控图2

非关系型数据库Redis压测，使用henchmark进行并发量为10000的测试，测试用服务器为后台服务器其一，1核2G，测试结果为throughput summary: 23041.47 requests per second，即平均每秒处理请求23041.47。测试结果如图-22所示

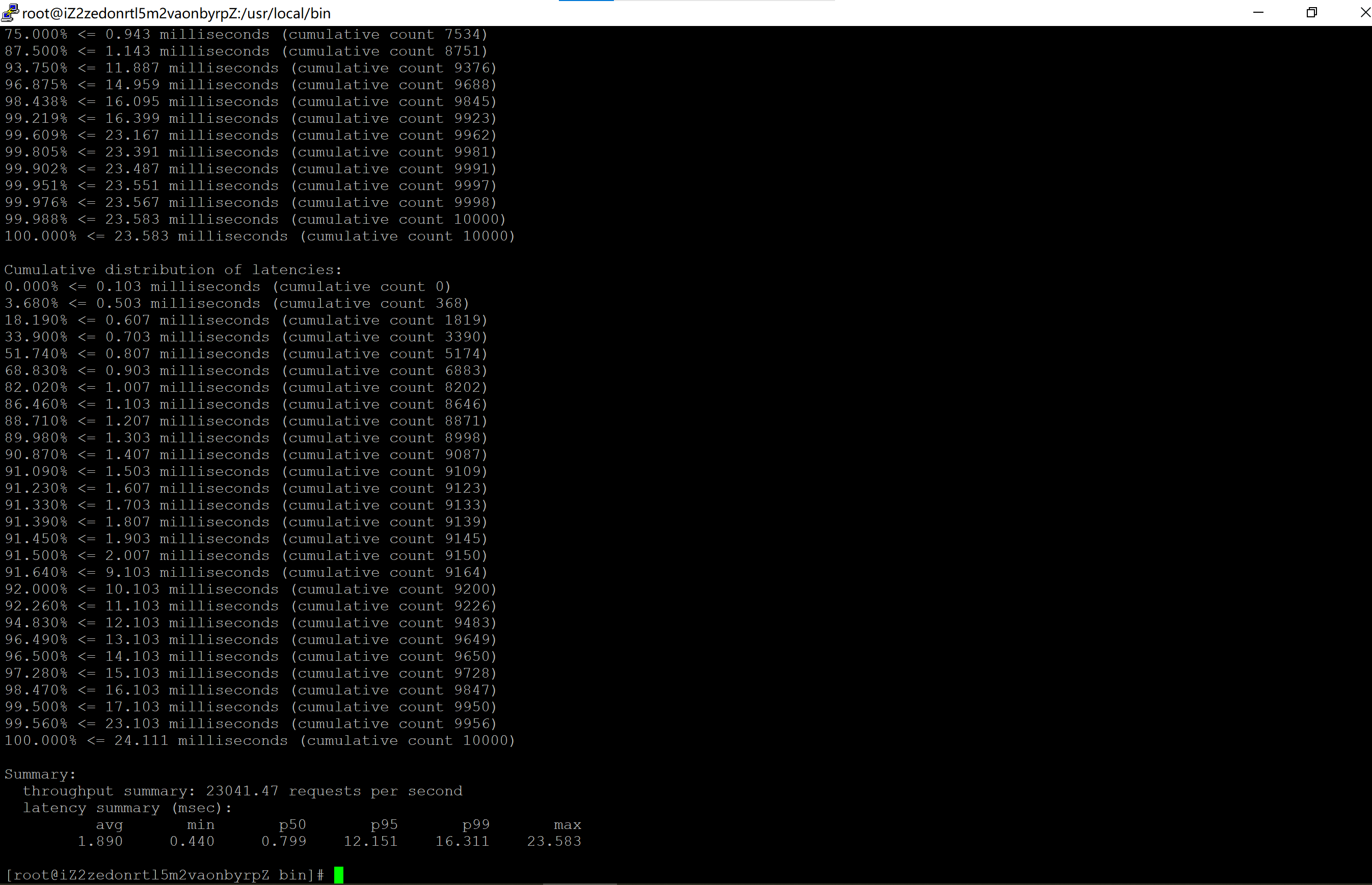


图22 Redis压测

4.3 部署与扩展

使用Nginx实现后台服务和前端网页的动静分离，手动进行部署，前端网页已在两个站点试运行（[neuq-oj-client (lin233.cc)](https://miao.lin233.cc/#/main/hello)、[neuq-oj-client](http://39.106.160.174:8087/#/main/hello)），后端服务通过java -jar命令手动部署，日志输出至本地文件夹oj.log，如图23所示。

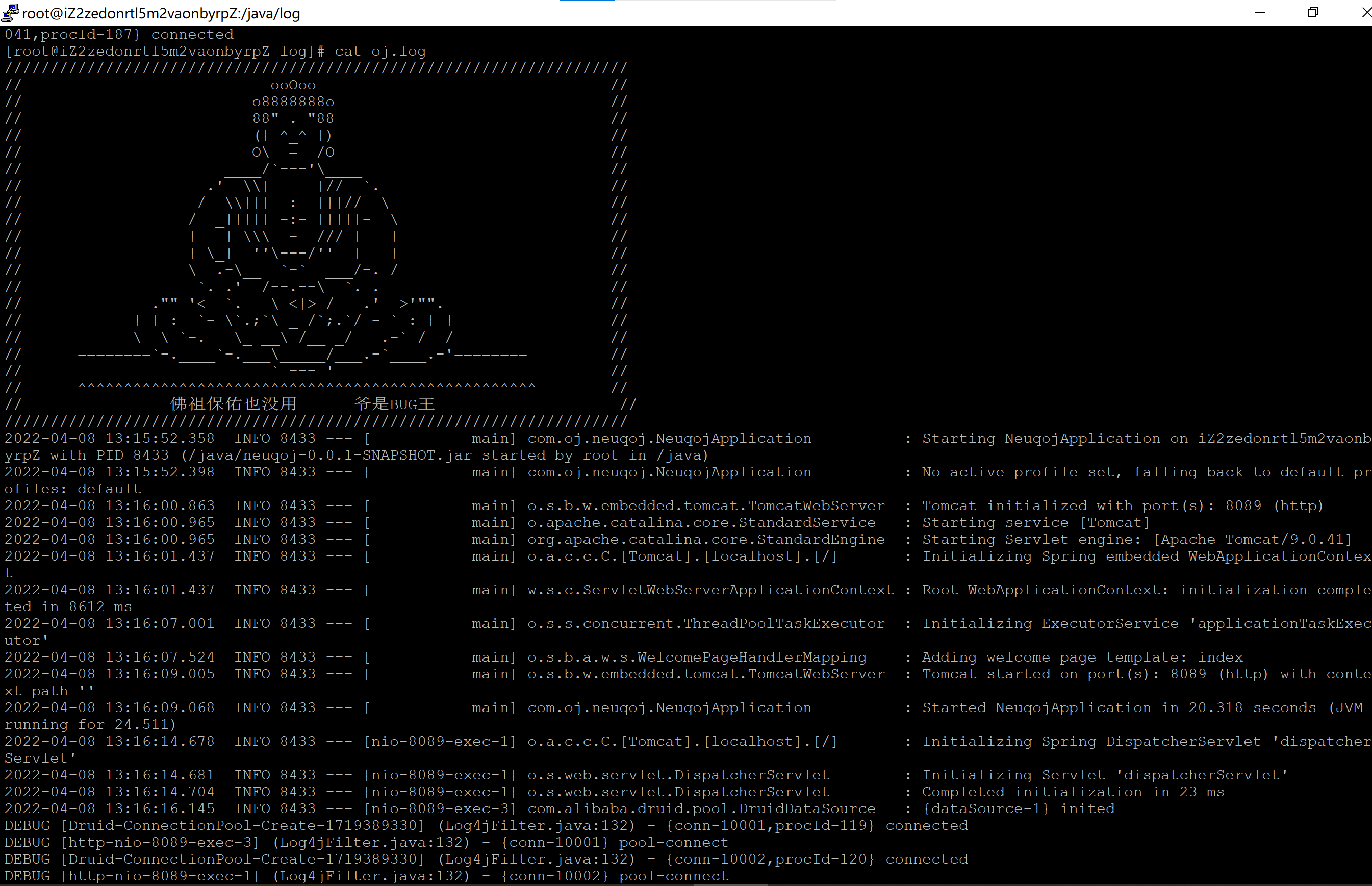


图23 后端日志

扩展性方面，由于前后端分离，功能的添加分工变得十分明确，后端提供接口，前端设置请求、展示结果。比如为用户添加一个订阅功能，只需在用户信息旁添加一个[subscribe]按钮，将请求发送到[/subscribe]接口，后端人员实现该接口功能即可，可谓十分便捷，另外Nginx提供的负载均衡以及热部署能供使后台服务“悄无声息”地重启，更新服务。

# 安装及使用

应用可直接通过URL访问:

部署链接一：[neuq-oj-client](http://39.106.160.174:8087/#/main/hello)

部署链接二：[neuq-oj-client](http://95.179.141.144/#/main/hello)

未登录权限为Only Read，无法进行本站的绝大部分操作，登录后可解锁判题、发布评论、发布绘画等SQL操作权限。账号通过邮箱进行注册，访问网站主页后，点击右上角登录按钮可找到注册链接。如图24-图25所示

登录完成后便可直接开始做题或发表评论，如图26-图28所示。

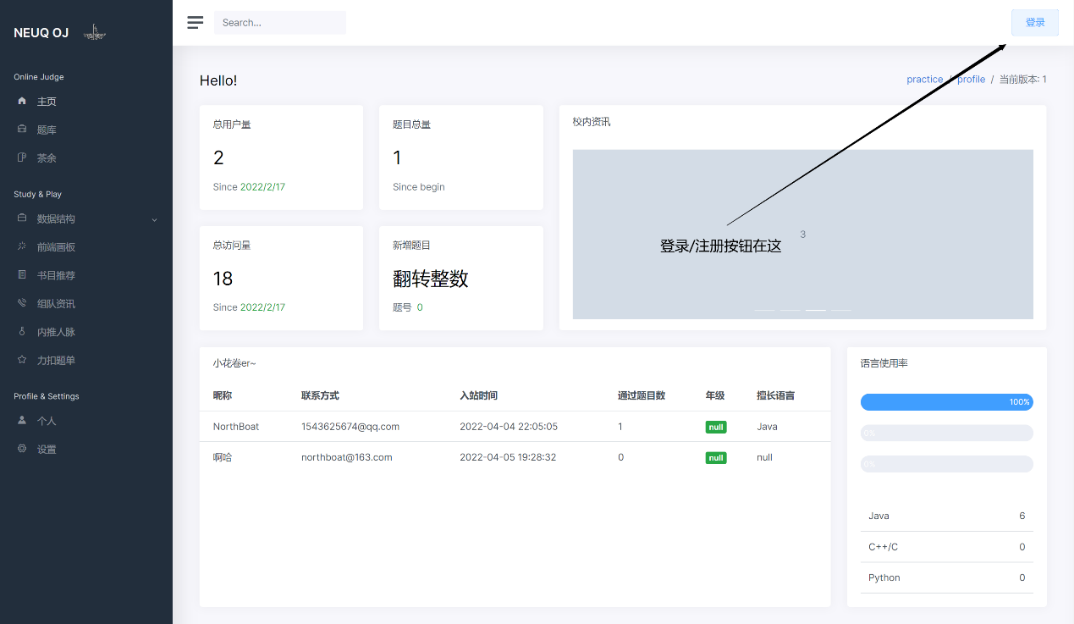


图24 主页登录

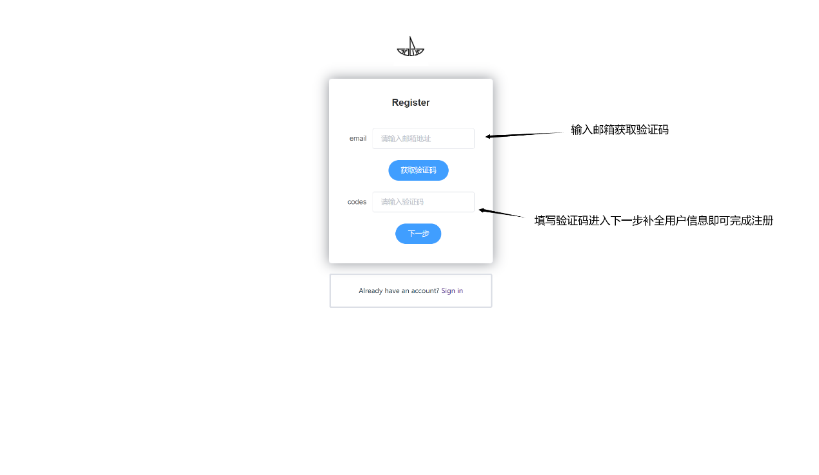


图25 登陆页面

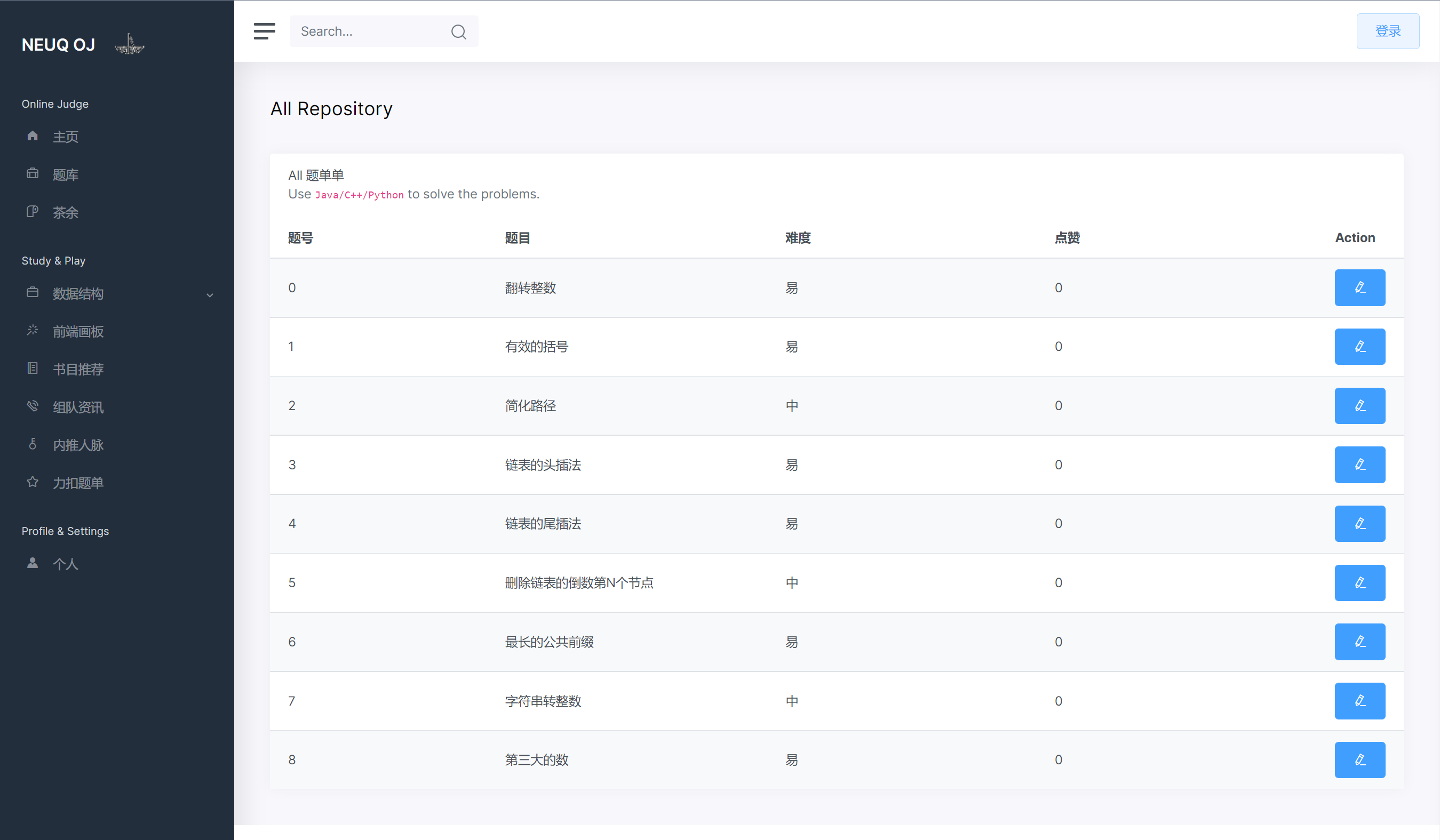


图26 题库页

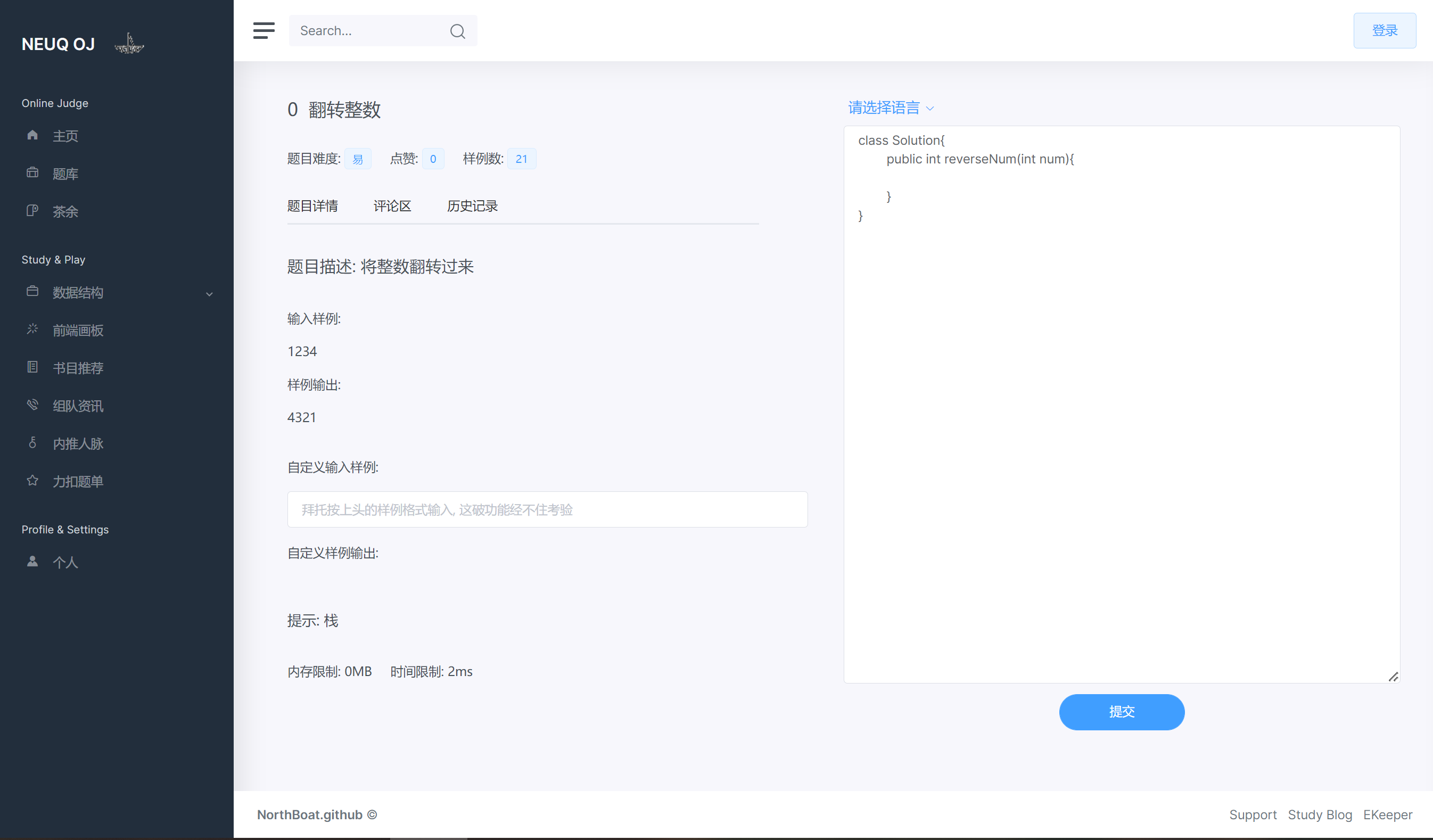


图27 判题工作页面

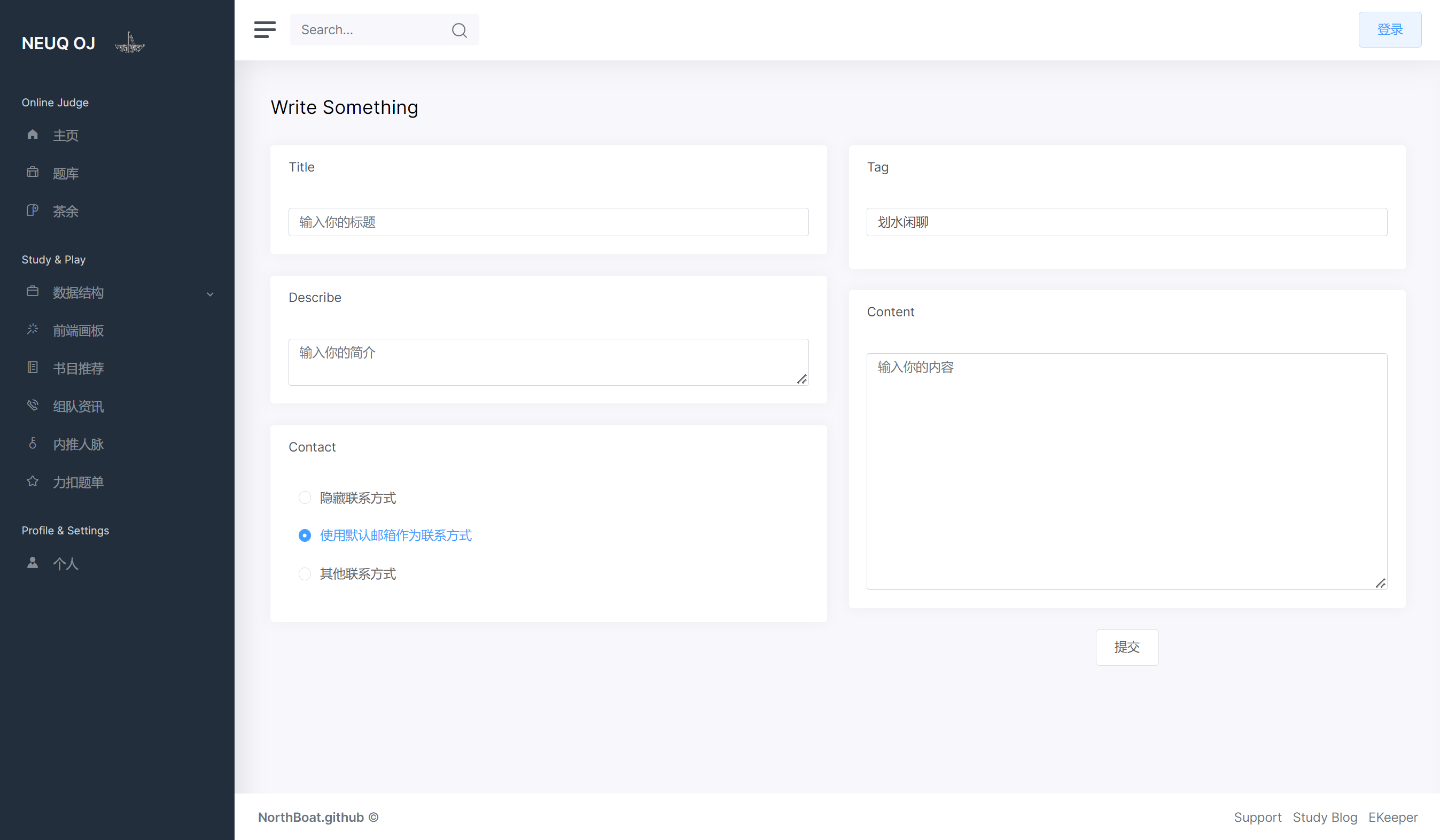


图28 发布话题

# 项目总结

在项目协调方面，主要的技术开发由熊舟桐同学解决，前后端开发工作配合十分融洽；运维工作由杨琦琛同学负责，诸如Docker服务器配置、前端网页部署、Nginx做反向代理以及负载均衡、测试工作；项目的后期工作，PPT制作、文档撰写、项目提交由徐荣鑫同学负责。总体看分工明确，成员各司其职，为彼此分担压力。在完善各自工作的同时，我们也向彼此学习了很多自己并不擅长的东西，综合能力有很大的提升，沟通交流也更为默契，结下了深厚的友谊。

本作品做商业推广是存在可能性的，数据库一开始便设计为“一校一库”的模式。但需要建立一套更加完善的后台管理系统，方便题库的录入以及用户信息管理；为了保证用户体验，服务器配置也得有相应提升，后台部署更多的后台服务以及Docker服务器，继续使用Nginx做好负载均衡，另外利用Redis解决分布式Session问题，将后台服务全部进行分布式部署。考虑保证网站的纯净性，并未预留广告位，盈利方式可能只有一开始的购买当前学校独有的数据库以及管理员权限。但一个良好的学习氛围是无价之宝，精神层面的培养往往是最难的，也是再多金钱也无法购买的。

由于项目前后端分离、分布式处理，鲁棒性极强，在扩展功能时，代码层面修改量极小，只需在现基础上不断丰富接口，于是平台的扩展变得异常容易。在后续考虑中，我们想设计一个智能推荐接口，帮助用户选择适合自己学习进度的题目、话题。另外，在同校若有其他的Web项目，如本校的编译原理在线展示平台“千机”，可以一同开辟一个导航页，为有需求的学生提供更为全面、优质的学习资源，共同创建一个优秀的编程社区环境，共同学习、进步、完善自己。