基于Spring Boot + Vue3的前后端分离项目构建方案

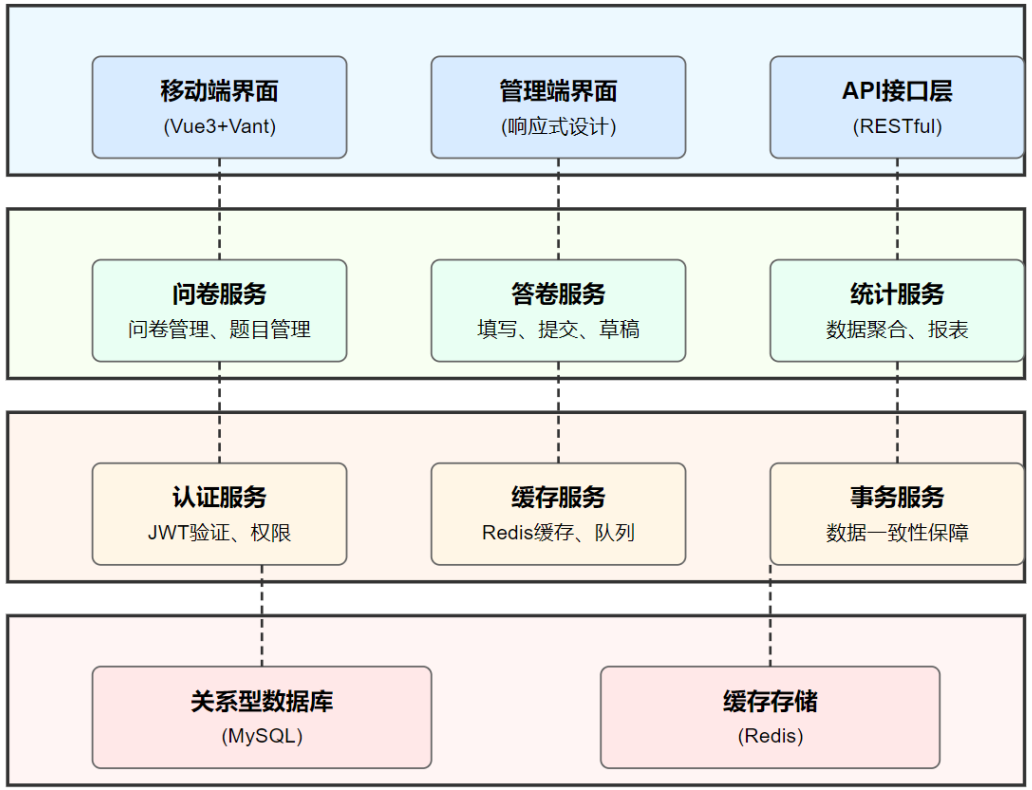
以问卷调查系统为例

指导老师：徐浩

成员：史洲航、饶家梁、赵子豪、李新宇

目录

移动端问卷调查系统 - 分层架构图



前期准备：技术栈选型

核心技术栈

技术	作用
Spring Boot	后端开发框架
Vue3	前端开发框架
RuoYi-Vue	快速开发脚手架
MySQL	关系型数据库
Redis	缓存数据库
Nginx	反向代理服务器

前期准备：环境配置

F 前端环境

N Node.js v16+
前端运行环境

V Vue CLI
项目构建工具

B 后端环境

J JDK 11+
Java开发环境

M Maven
项目管理工具

T 开发工具

I IntelliJ IDEA
Java开发IDE

V VS Code
前端开发IDE

N Navicat
数据库管理工具

后端开发：框架整合

RuoYi-Vue初始化流程

1. **下载脚手架**

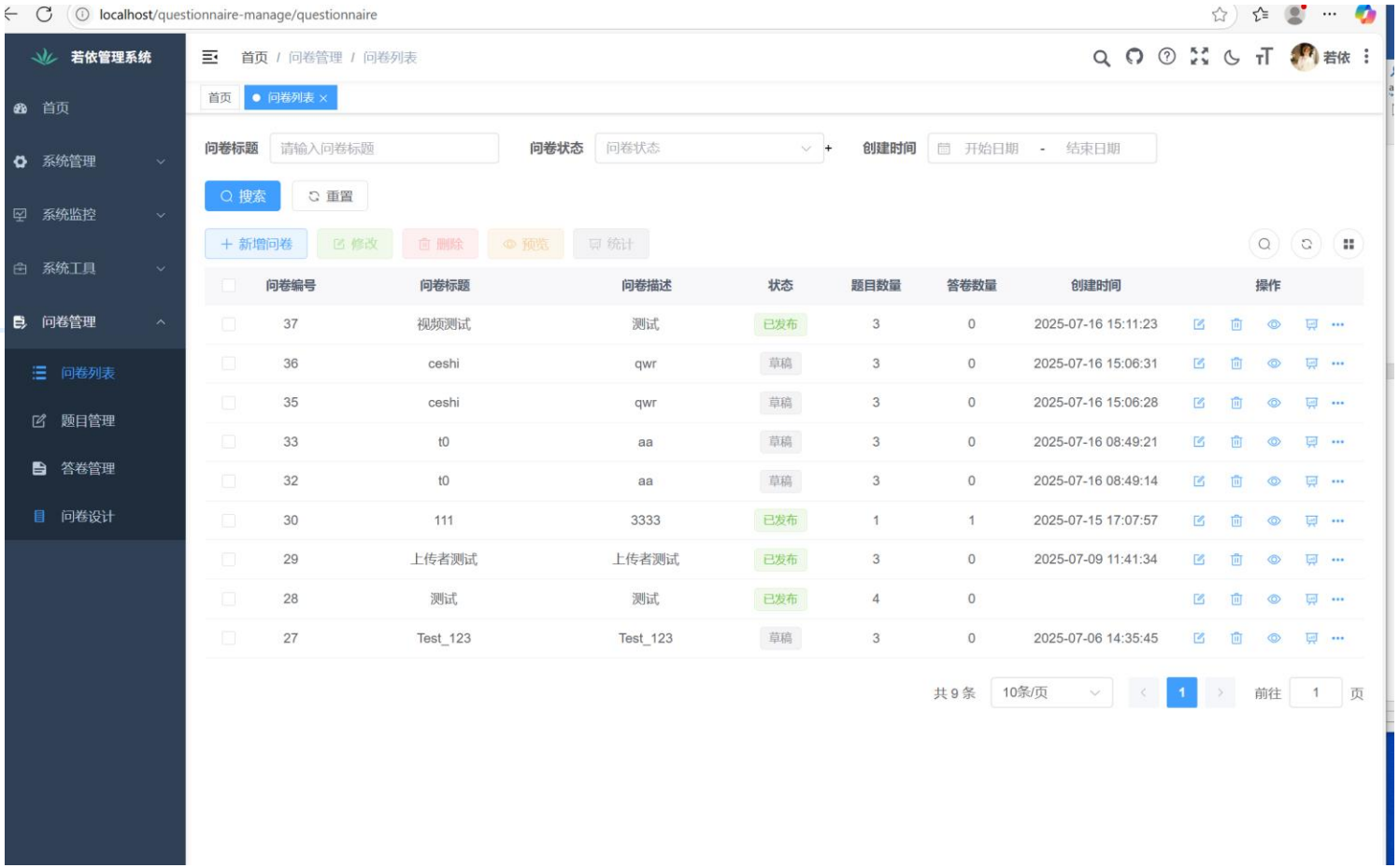
   从GitHub获取RuoYi-Vue最新版本

2. **配置数据库**

   修改application.yml文件设置数据库连接

3. **启动服务**

   运行Application主类启动后端服务

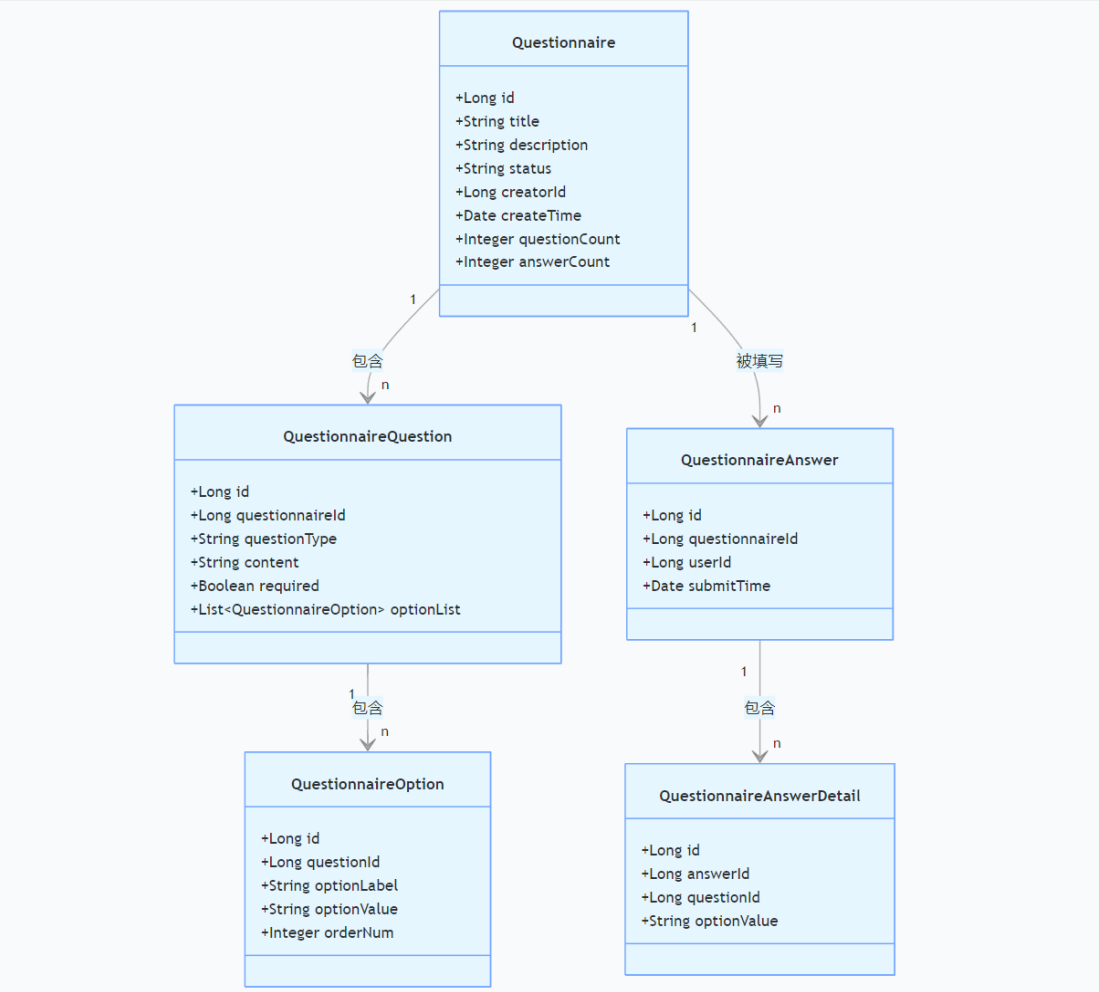


RuoYi-Vue 网页界面

后端开发：核心功能 -数据库设计

核心数据实体

表名	核心字段	说明
questionnaire（问卷表）	id（主键）、title（标题）、status（状态：0 草稿 / 1 发布）、creator_id（创建者 ID）	存储问卷基本信息
question（题目表）	id、questionnaire_id（关联问卷）、content（题干）、type（类型：single/multiple/text）	存储题目内容与类型
option（选项表）	id、question_id（关联题目）、label（显示文本：如 “A. 满意”）、value（存储值：如 “A”）	存储选择题的选项
answer（答卷表）	id、questionnaire_id（关联问卷）、user_id（填写用户）、submit_time（提交时间）	存储答卷主记录
answer_detail（答卷详情表）	id、answer_id（关联答卷）、question_id（关联题目）、option_value（答案值）	存储用户对每个题目的具回答



实体关系图

后端开发：核心功能　　-接口开发

RESTful　接口示例

GET

/api/questionnaire/{id}

获取问卷详情

POST

//questionnaire/list

获取问卷列表

PUT

/questionnaire/answer/submit

提交答卷

DELETE

/questionnaire/stats/${id}

获取问卷统计

```
import request from '@/utils/request'

// 获取问卷列表
export function getQuestionnaireList(params) {  Show usages
  return request({
    url: '/questionnaire/list',
    method: 'get',
    params
  })
}

// 获取问卷详情
export function getQuestionnaireDetail(id) {  no usages
  return request({
    url: `/questionnaire/${id}`,
    method: 'get'
  })
}

// 获取问卷题目
export function getQuestionList(questionnaireId) {  no usages
  return request({
    url: '/questionnaire/question/list',
    method: 'get',
    params: { questionnaireId }
  })
}

// 获取题目选项
export function getOptionList(questionId) {  no usages
  return request({
```

后端开发：高并发处理

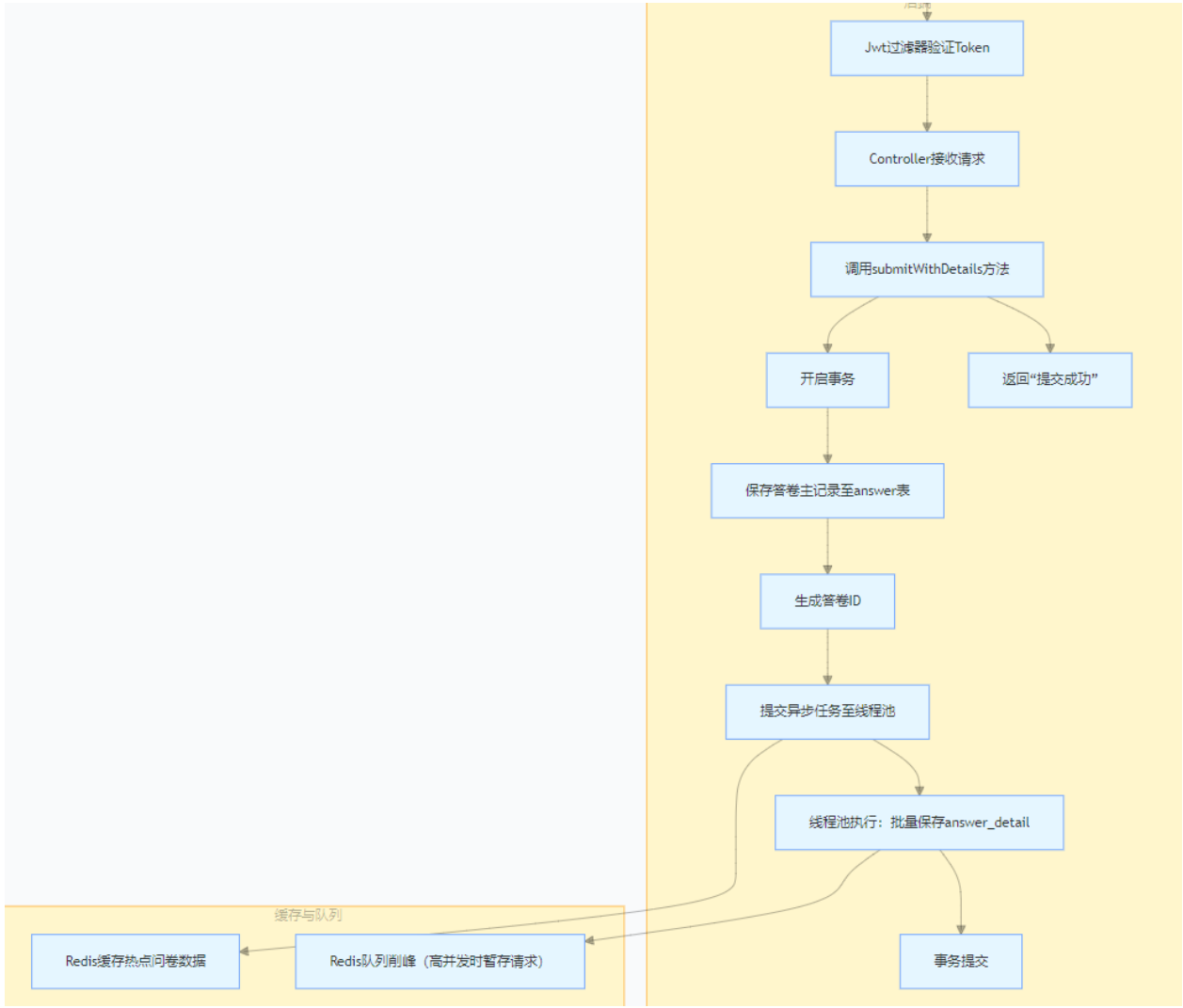
高并发解决方案

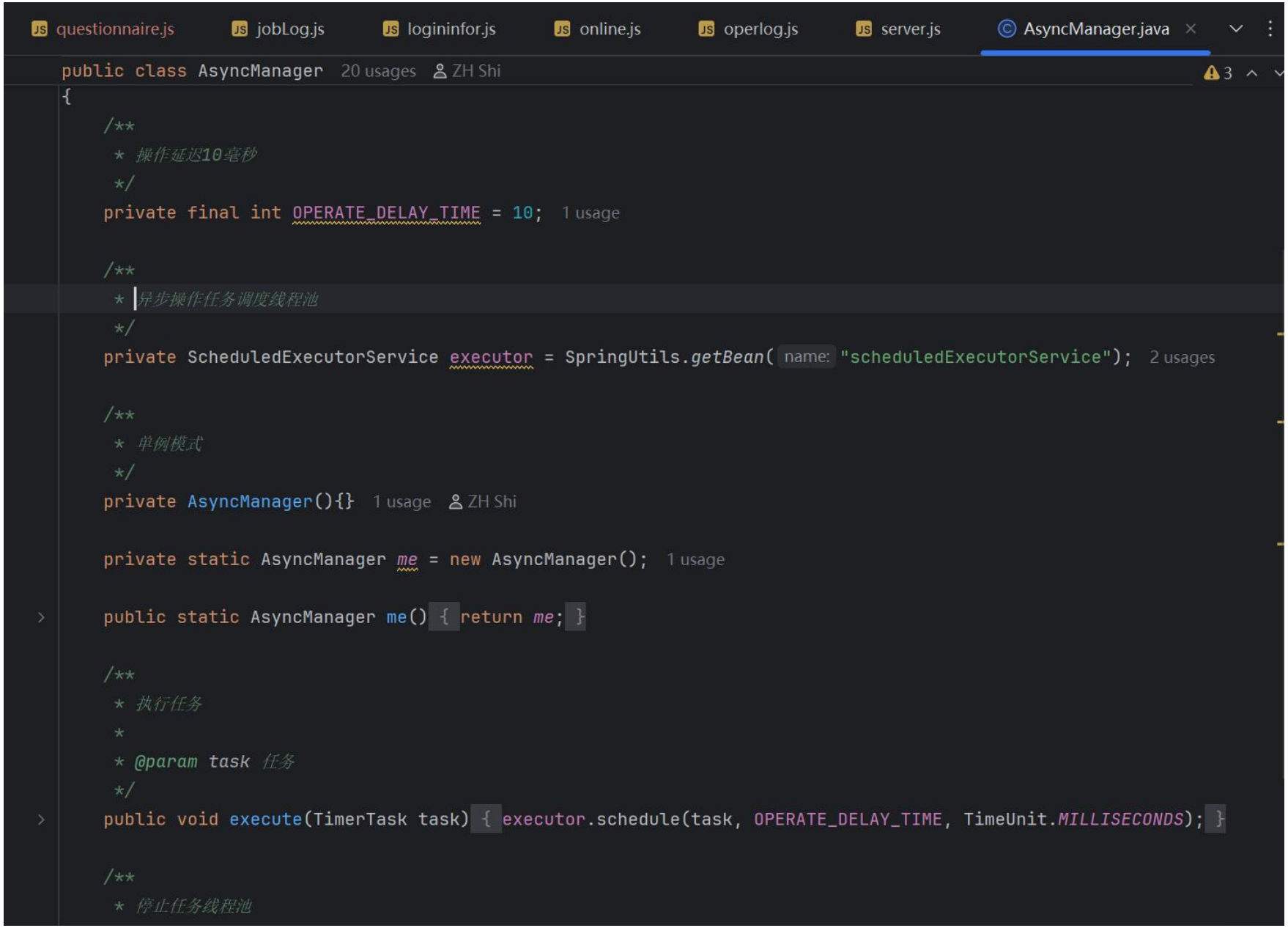
1. **Redis缓存热门数据**

   使用Redis缓存热门问卷数据，减少数据库查询压力，提高响应速度

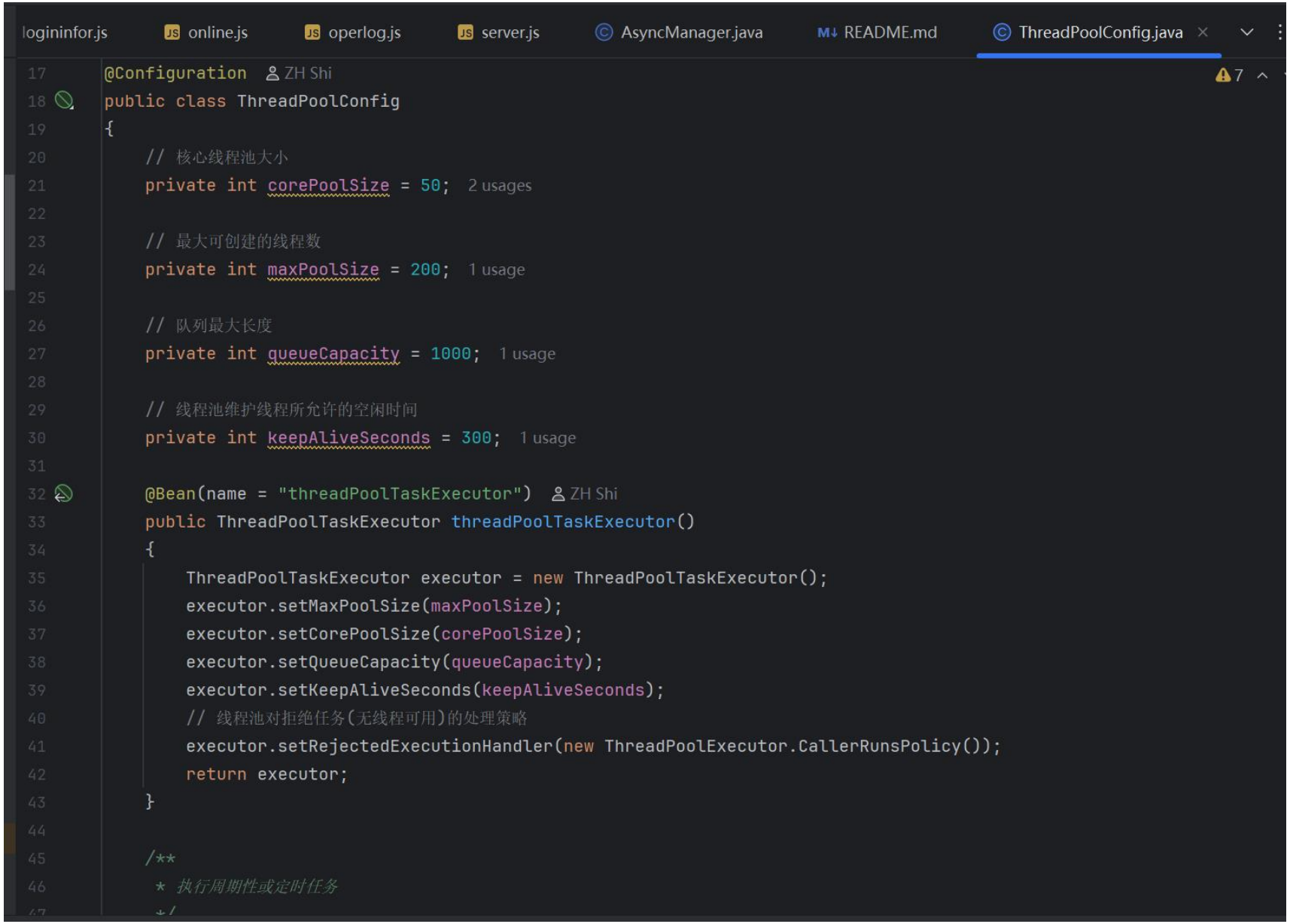
2. **线程池异步处理**

   通过线程池异步处理问卷提交结果，避免主线程阻塞，提高系统吞吐量




questionnaire.js
jobLog.js
logininfor.js
online.js
operlog.js
server.js
AsyncManager.java
public class AsyncManager 20 usages ZH Shi
3
{
/**
* 操作延迟10毫秒
*/
private final int OPERATE_DELAY_TIME = 10; 1 usage
/**
* 异步操作任务调度线程池
*/
private ScheduledExecutorService executor = SpringUtils.getBean(name: "scheduledExecutorService"); 2 usages
/**
* 单例模式
*/
private AsyncManager(){} 1 usage ZH Shi
private static AsyncManager me = new AsyncManager(); 1 usage
public static AsyncManager me() { return me; }
/**
* 执行任务
*
* @param task 任务
*/
public void execute(TimerTask task) { executor.schedule(task, OPERATE_DELAY_TIME, TimeUnit.MILLISECONDS); }
/**
* 停止任务线程池

异步操作代码示例


logininfor.js
online.js
operlog.js
server.js
AsyncManager.java
README.md
ThreadPoolConfig.java
@Configuration ZH Shi
public class ThreadPoolConfig
{
// 核心线程池大小
private int corePoolSize = 50; 2 usages
// 最大可创建的线程数
private int maxPoolSize = 200; 1 usage
// 队列最大长度
private int queueCapacity = 1000; 1 usage
// 线程池维护线程所允许的空闲时间
private int keepAliveSeconds = 300; 1 usage
@Bean(name = "threadPoolTaskExecutor") ZH Shi
public ThreadPoolTaskExecutor threadPoolTaskExecutor()
{
ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor();
executor.setMaxPoolSize(maxPoolSize);
executor.setCorePoolSize(corePoolSize);
executor.setQueueCapacity(queueCapacity);
executor.setKeepAliveSeconds(keepAliveSeconds);
// 线程池对拒绝任务(无线程可用)的处理策略
executor.setRejectedExecutionHandler(new ThreadPoolExecutor.CallerRunsPolicy());
return executor;
}
/**
* 执行周期性或定时任务

线程池代码实例

前端开发：项目结构

src 目录核心结构

A api

接口封装模块，统一管理 API 请求

C components

公共组件库，可复用 UI 组件

V views

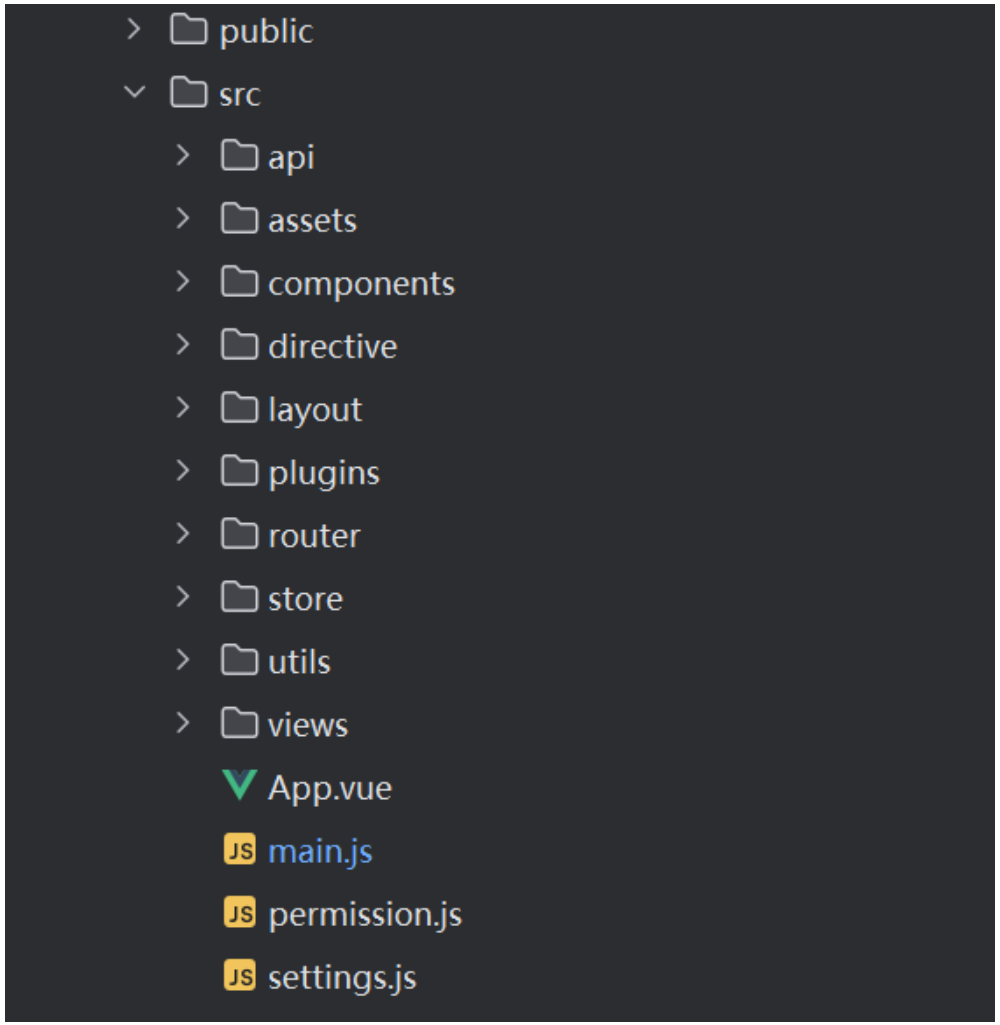
页面组件，包含各业务页面

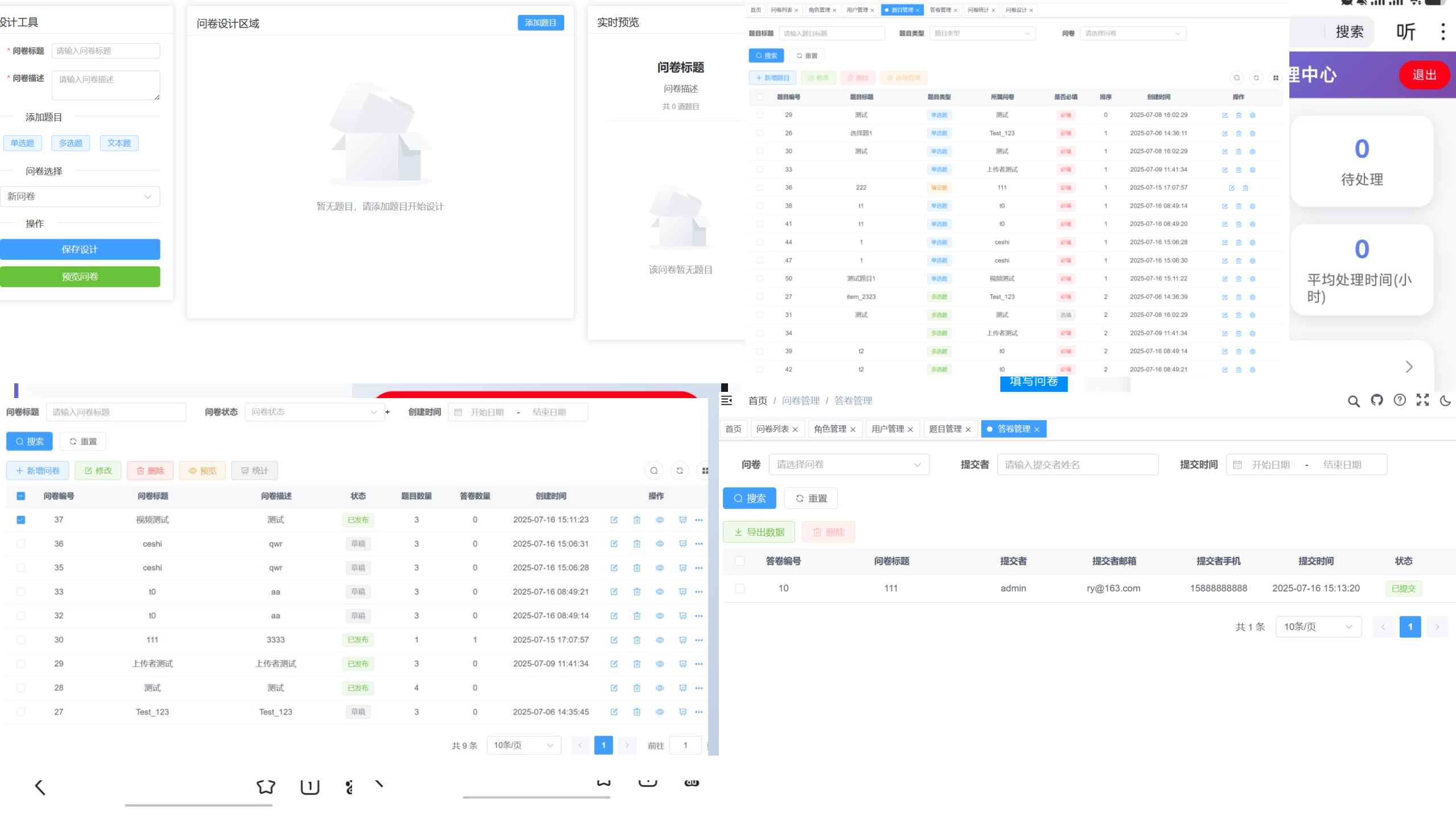
R router

路由配置，管理页面导航

S store

状态管理，使用 Pinia 管理全局状态




设计工具
* 问卷标题
请输入问卷标题
* 问卷描述
请输入问卷描述
添加题目
单选题
多选题
文本题
问卷选择
新问卷
操作
保存设计
预览问卷
问卷设计区域
添加题目
暂无题目，请添加题目开始设计
实时预览
问卷标题
问卷描述
共 0 道题目
该问卷暂无题目
搜索
听
理中心
退出
0
待处理
0
平均处理时间(小时)
题目标题
题目类型
问卷
搜索
重置
新增题目
修改
删除
选项管理
题目编号
题目标题
题目类型
所属问卷
是否必填
排序
创建时间
操作
29 测试 单选题 测试 必填 0 2025-07-08 16:02:29
26 选择题1 单选题 Test_123 必填 1 2025-07-06 14:36:11
30 测试 单选题 测试 必填 1 2025-07-08 16:02:29
33 单选题 上传者测试 必填 1 2025-07-09 11:41:34
36 222 填空题 111 必填 1 2025-07-15 17:07:57
38 t1 单选题 t0 必填 1 2025-07-16 08:49:14
41 t1 单选题 t0 必填 1 2025-07-16 08:49:20
44 1 单选题 ceshi 必填 1 2025-07-16 15:06:28
47 1 单选题 ceshi 必填 1 2025-07-16 15:06:30
50 测试题目1 单选题 视频测试 必填 1 2025-07-16 15:11:22
27 item_2323 多选题 Test_123 必填 2 2025-07-06 14:36:39
31 测试 多选题 测试 选填 2 2025-07-08 16:02:29
34 多选题 上传者测试 必填 2 2025-07-09 11:41:34
39 t2 多选题 t0 必填 2 2025-07-16 08:49:14
42 t2 多选题 t0 必填 2 2025-07-16 08:49:21
填写问卷
问卷标题
请输入问卷标题
问卷状态
问卷状态
创建时间
开始日期
结束日期
搜索
重置
新增问卷
修改
删除
预览
统计
问卷编号
问卷标题
问卷描述
状态
题目数量
答卷数量
创建时间
操作
37 视频测试 测试 已发布 3 0 2025-07-16 15:11:23
36 ceshi qwr 草稿 3 0 2025-07-16 15:06:31
35 ceshi qwr 草稿 3 0 2025-07-16 15:06:28
33 t0 aa 草稿 3 0 2025-07-16 08:49:21
32 t0 aa 草稿 3 0 2025-07-16 08:49:14
30 111 3333 已发布 1 1 2025-07-15 17:07:57
29 上传者测试 上传者测试 已发布 3 0 2025-07-09 11:41:34
28 测试 测试 已发布 4 0
27 Test_123 Test_123 草稿 3 0 2025-07-06 14:35:45
共 9 条
10条/页
前往
首页 / 问卷管理 / 答卷管理
首页
问卷列表
角色管理
用户管理
题目管理
答卷管理
问卷
请选择问卷
提交者
请输入提交者姓名
提交时间
开始日期
结束日期
搜索
重置
导出数据
删除
答卷编号
问卷标题
提交者
提交者邮箱
提交者手机
提交时间
状态
10 111 admin ry@163.com 15888888888 2025-07-16 15:13:20 已提交
共 1 条
10条/页
1

前端开发：路由与状态

路由与状态管理

路由配置　RuoYi-Vue-master\ruoyi-ui-vue3\src\router\index.js

使用Vue Router实现页面路由，支持动态路由加载

```
*/
@GetMapping("getRouters")  ZH Shi
public AjaxResult getRouters()
{
    Long userId = SecurityUtils.getUserId();
    List<SysMenu> menus = menuService.selectMenuTreeByUserId(userId);
    return AjaxResult.success(menuService.buildMenus(menus));
}
```

```
/////// 移动端路由 ///////

{
  path: '/redirect',
  component: Layout,
  hidden: true,
  children: [
    {
      path: '/redirect/:path(.*)',
      component: () : Promise<... => import('@/views/redirect/index.vue')
    }
  ]
},
{
  path: '/login',
  component: () : Promise<any> => import('@/views/login'),
  hidden: true
},
{
  path: '/register',
  component: () : Promise<any> => import('@/views/register'),
  hidden: true
},
```

测试部署：测试方案

F 功能测试

问卷创建
验证问卷创建流程完整性

问卷提交
测试用户提交问卷功能

数据统计
验证统计结果准确性

P 性能测试

10000并发用户
模拟高并发访问场景

响应时间
API平均响应时间 ≤ 500ms

资源占用
监控CPU/内存使用率

C 兼容性测试

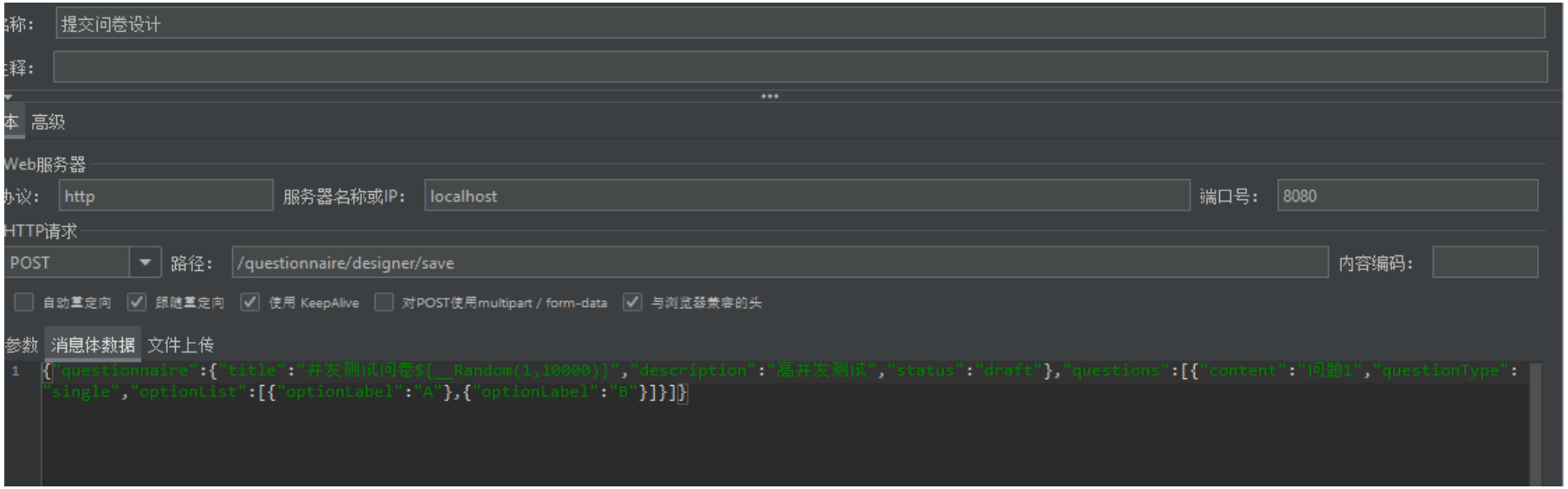
Chrome/Edge
主流桌面浏览器

移动端浏览器
Safari/Chrome移动版

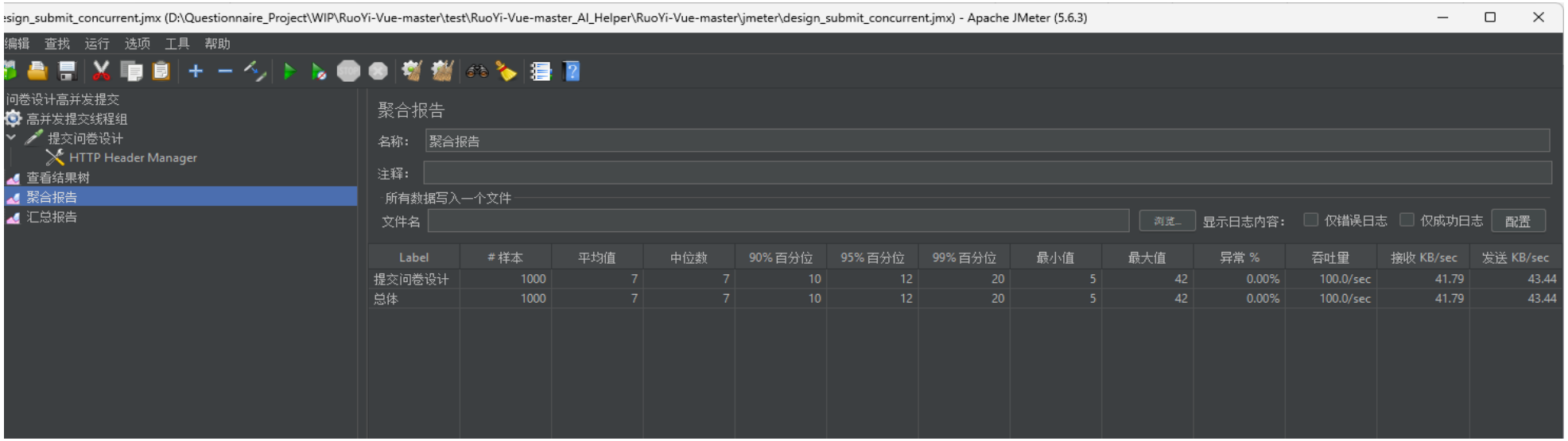
分辨率适配
不同屏幕尺寸响应式

压力测试

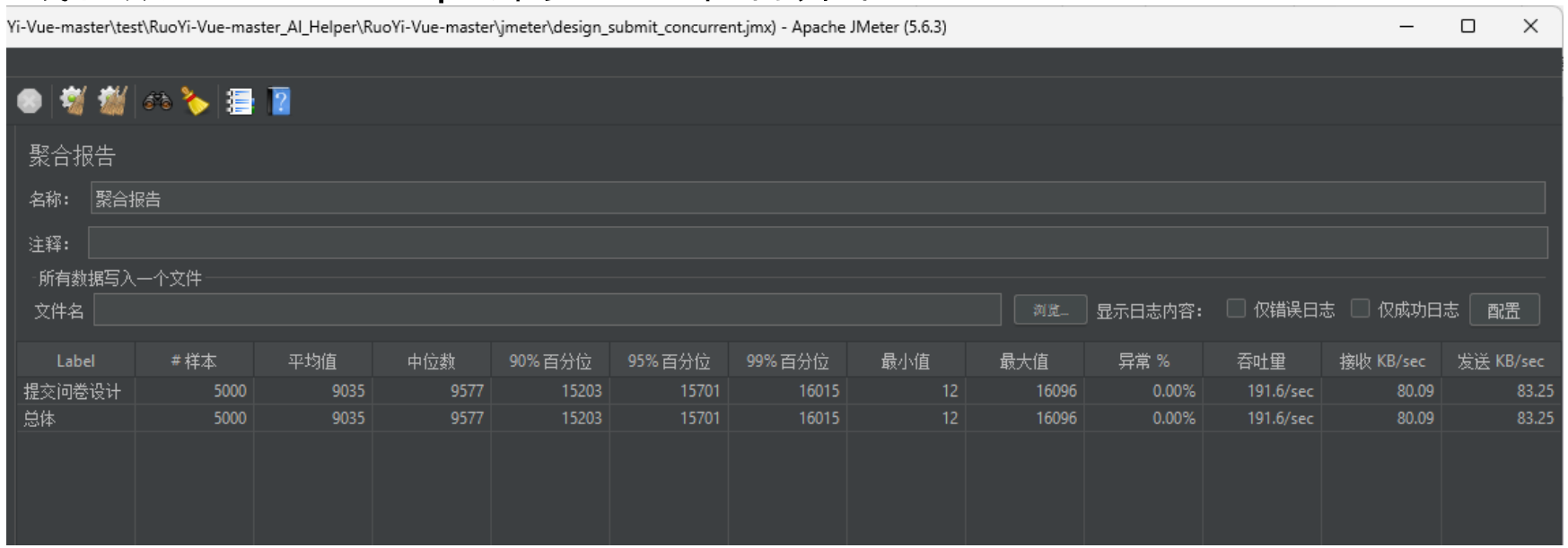
请求体内容:



线程数1000　run up时间10s 无任何异常

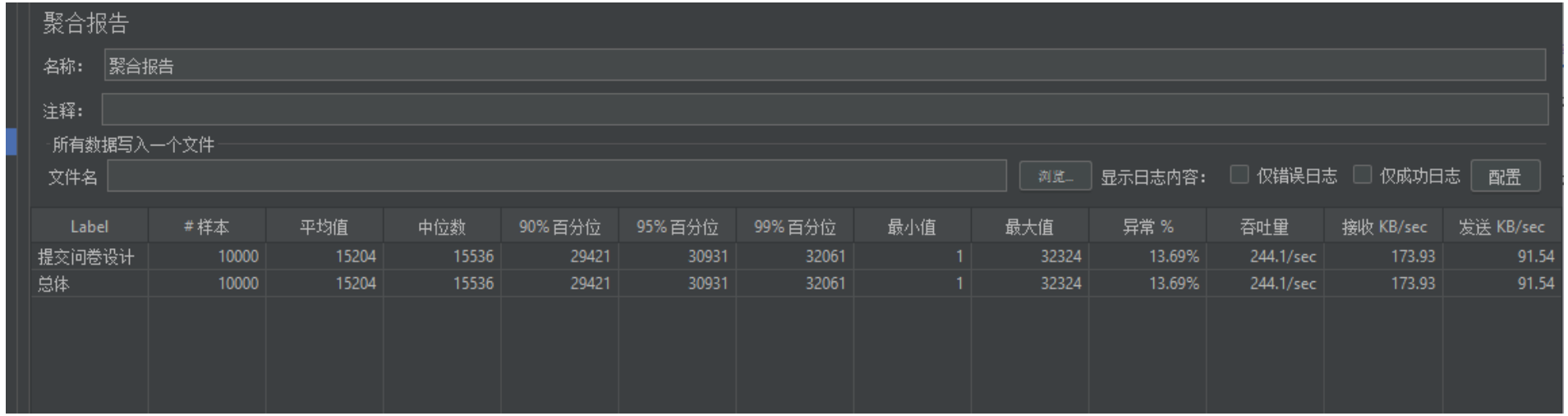


线程数5000　run up时间10s 无任何异常



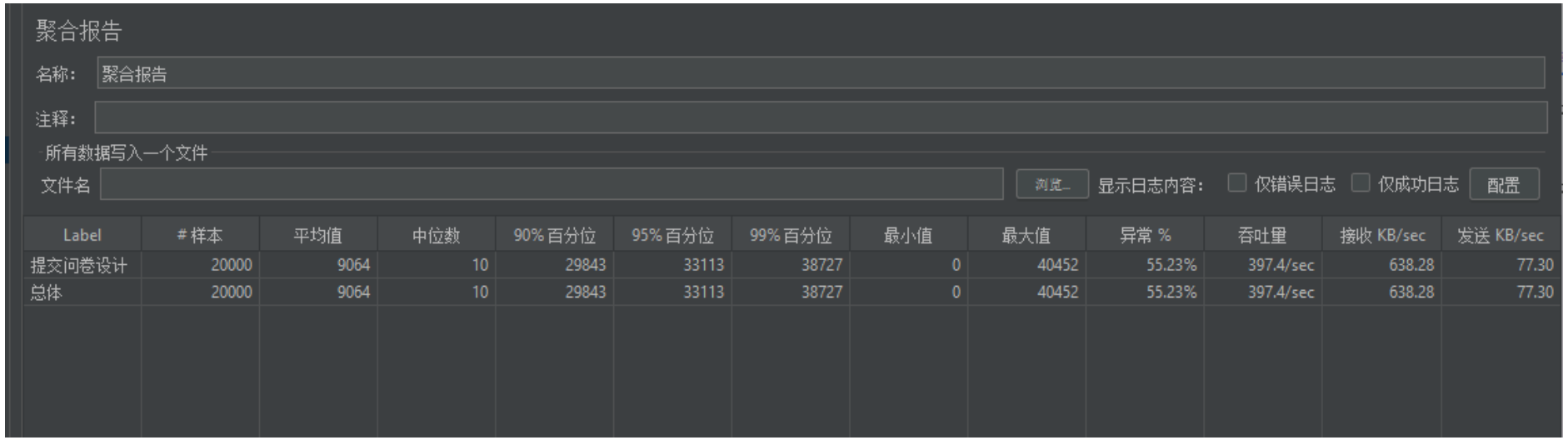
Label	# 样本	平均值	中位数	90% 百分位	95% 百分位	99% 百分位	最小值	最大值	异常 %	吞吐量	接收 KB/sec	发送 KB/sec
提交问卷设计	5000	9035	9577	15203	15701	16015	12	16096	0.00%	191.6/sec	80.09	83.25
总体	5000	9035	9577	15203	15701	16015	12	16096	0.00%	191.6/sec	80.09	83.25

线程数10000　run up时间10s 异常率：13.69%



Label	# 样本	平均值	中位数	90% 百分位	95% 百分位	99% 百分位	最小值	最大值	异常 %	吞吐量	接收 KB/sec	发送 KB/sec
提交问卷设计	10000	15204	15536	29421	30931	32061	1	32324	13.69%	244.1/sec	173.93	91.54
总体	10000	15204	15536	29421	30931	32061	1	32324	13.69%	244.1/sec	173.93	91.54

线程数20000　run up时间10s 异常率：55.23%



Label	# 样本	平均值	中位数	90% 百分位	95% 百分位	99% 百分位	最小值	最大值	异常 %	吞吐量	接收 KB/sec	发送 KB/sec
提交问卷设计	20000	9064	10	29843	33113	38727	0	40452	55.23%	397.4/sec	638.28	77.30
总体	20000	9064	10	29843	33113	38727	0	40452	55.23%	397.4/sec	638.28	77.30

测试部署：部署流程

F 前端部署

1. **构建生产包**
   运行 npm run build 生成静态资源
2. **配置Nginx**
   设置反向代理和静态资源路径
3. **部署上线**
   将dist目录部署到Nginx服务器

B 后端部署

1. **项目打包**
   使用Maven打包生成JAR文件
2. **上传服务器**
   将JAR文件上传到生产环境
3. **启动服务**
   运行 java -jar 命令启动服务

总结

核心亮点

1. **模块化设计**
   前后端分离架构，组件化开发，便于维护和扩展
2. **高并发支持**
   Redis+线程池方案，有效应对高并发访问场景
3. **安全控制机制**
   完善的权限管理和数据加密机制，保障系统安全

优化方向

1. **Redis缓存策略**
   实现动态调整的缓存策略，提高资源利用率
2. **前端性能优化**
   组件懒加载和代码分割，提升页面加载速度
3. **自动化部署**
   集成CI/CD流水线，实现一键部署