

# 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

BE Server N

API

缓存

请求

响应

API

Mock

View+各种组件

Redux

响应

用户

指导老师：徐浩

成员：史洲航、饶家梁、赵子豪、李新宇

日志管理

性能监控

埋点统计

# 目录

1 前期准备

2 后端开发

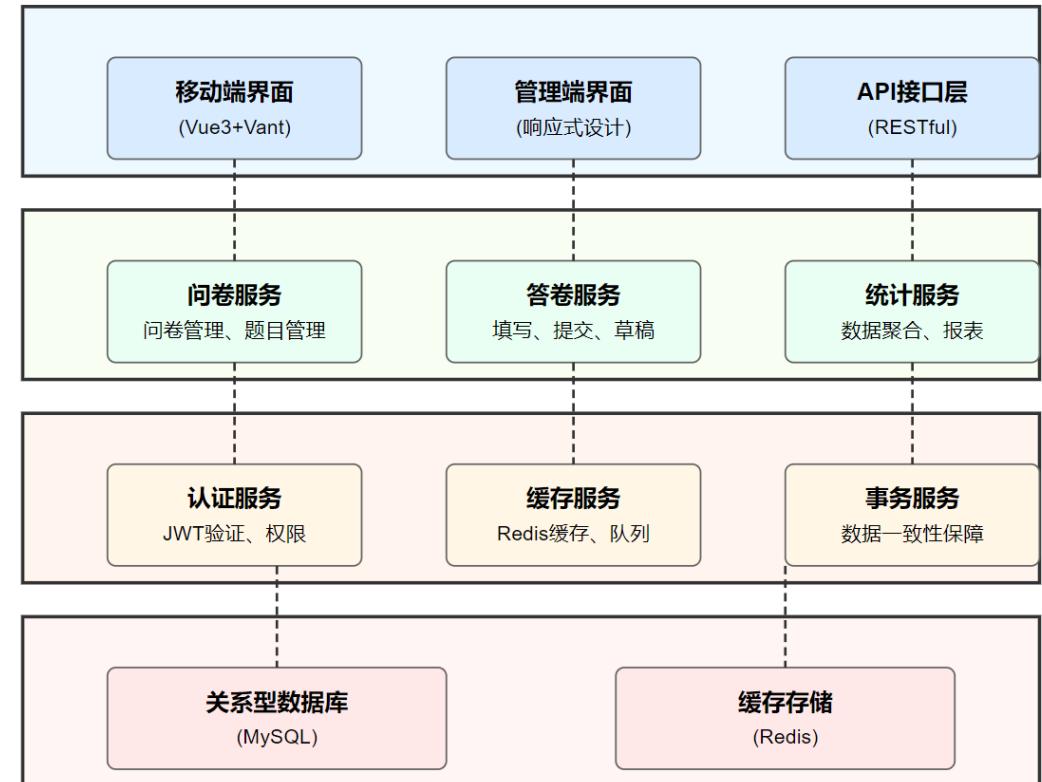
3 前端开发

4 前后端对接

5 测试部署

6 总结

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



@稀土掘金技术社区

# 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

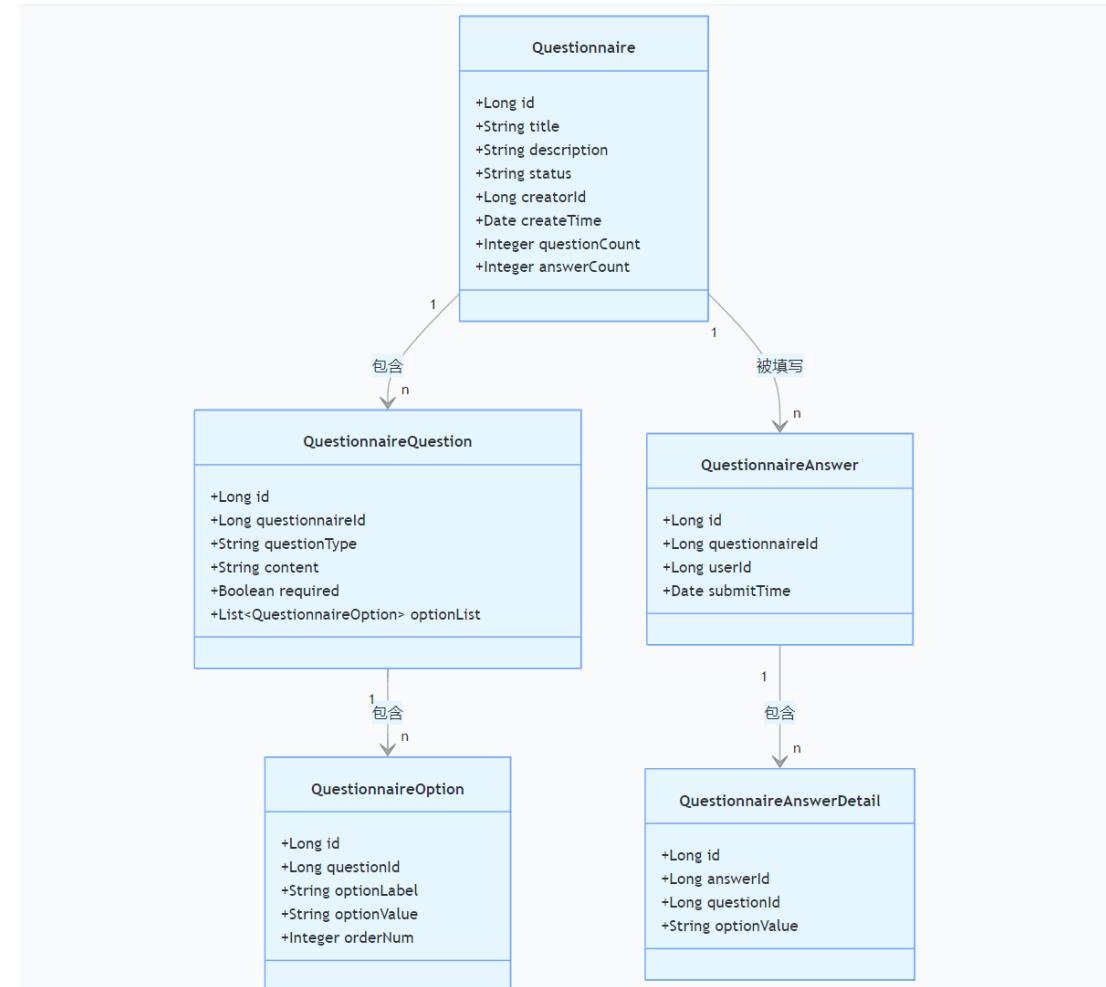
The screenshot shows a web-based questionnaire management system. The left sidebar has a dark blue theme with white text, listing '首页', '系统管理', '系统监控', '系统工具', '问卷管理' (selected), '题目管理', '答卷管理', and '问卷设计'. The main content area has a light gray background. At the top, there's a search bar with placeholder '请输入问卷标题' and dropdown filters for '问卷状态' (status) and '创建时间' (creation time). Below these are buttons for '+ 新增问卷' (Add New Questionnaire), '修改' (Edit), '删除' (Delete), '预览' (Preview), and '统计' (Statistics). A table lists 10 questionnaires with columns: '问卷编号' (Questionnaire ID), '问卷标题' (Title), '问卷描述' (Description), '状态' (Status), '题目数量' (Number of Questions), '答卷数量' (Number of Answers), '创建时间' (Creation Time), and '操作' (Operations). Each row includes a checkbox and icons for edit, delete, preview, and more. At the bottom right, there are pagination controls: '共 9 条' (Total 9 items), '10条/页' (10 items per page), and buttons for '上一页' (Previous page), '下一页' (Next page), '前往' (Go to), and '1 页' (Page 1).

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

## 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

```
1 import request from '@/utils/request'  
2  
3 // 获取问卷列表  
4 export function getQuestionnaireList(params) { Show usages  
5   return request({  
6     url: '/questionnaire/list',  
7     method: 'get',  
8     params  
9   })  
10 }  
11  
12 // 获取问卷详情  
13 export function getQuestionnaireDetail(id) { no usages  
14   return request({  
15     url: `/questionnaire/${id}`,  
16     method: 'get'  
17   })  
18 }  
19  
20 // 获取问卷题目  
21 export function getQuestionList(questionnaireId) { no usages  
22   return request({  
23     url: '/questionnaire/question/list',  
24     method: 'get',  
25     params: { questionnaireId }  
26   })  
27 }  
28  
29 // 获取题目选项  
30 export function getOptionList(questionId) { no usages  
31   return request({  
32     url: `/questionnaire/question/${questionId}/option`,  
33     method: 'get'  
34   })  
35 }
```

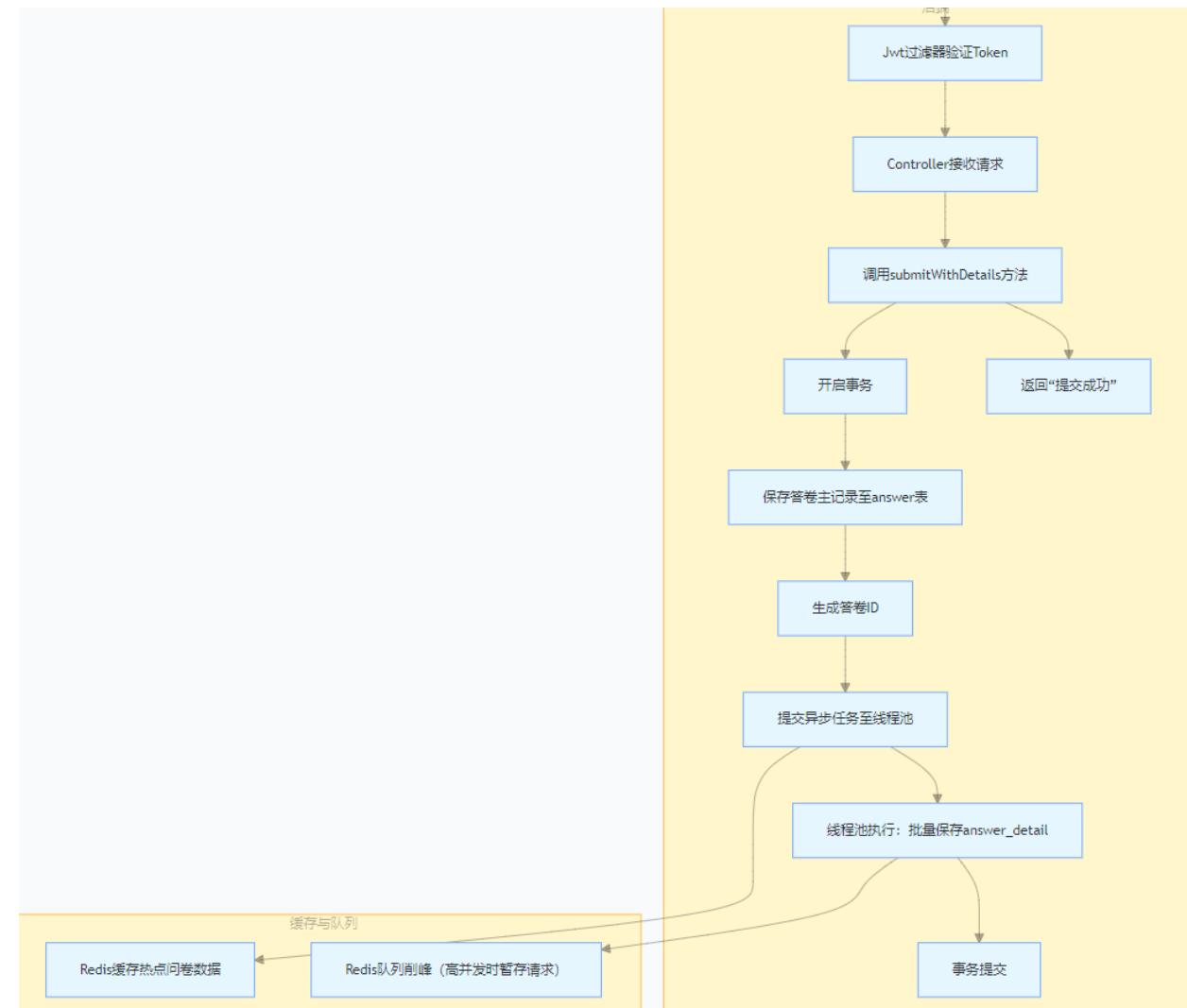
## 高并发解决方案

### 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



The screenshot shows a Java code editor with the file `AsyncManager.java` open. The code defines a class `AsyncManager` with various methods and fields. The code includes Chinese comments and annotations. The editor interface shows tabs for other files like `questionnaire.js`, `jobLog.js`, etc., and a status bar with a warning icon and the number 3.

```
public class AsyncManager 20 usages  ZH Shi
{
    /**
     * 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

```
loginfor.js    online.js    operlog.js    server.js    AsyncManager.java    README.md    ThreadPoolConfig.java    ▾ 7 :  
17  @Configuration  ♫ ZH Shi  
18  public class ThreadPoolConfig  
19  {  
20      // 核心线程池大小  
21      private int corePoolSize = 50;  2 usages  
22  
23      // 最大可创建的线程数  
24      private int maxPoolSize = 200;  1 usage  
25  
26      // 队列最大长度  
27      private int queueCapacity = 1000;  1 usage  
28  
29      // 线程池维护线程所允许的空闲时间  
30      private int keepAliveSeconds = 300;  1 usage  
31  
32  @Bean(name = "threadPoolTaskExecutor")  ♫ ZH Shi  
33  public ThreadPoolTaskExecutor threadPoolTaskExecutor()  
34  {  
35      ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor();  
36      executor.setMaxPoolSize(maxPoolSize);  
37      executor.setCorePoolSize(corePoolSize);  
38      executor.setQueueCapacity(queueCapacity);  
39      executor.setKeepAliveSeconds(keepAliveSeconds);  
40      // 线程池对拒绝任务(无线程可用)的处理策略  
41      executor.setRejectedExecutionHandler(new ThreadPoolExecutor.CallerRunsPolicy());  
42      return executor;  
43  }  
44  
45  /**  
46      * 执行周期性或定时任务  
47  */
```

## 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

```
> └─ public
└─ src
    > └─ api
    > └─ assets
    > └─ components
    > └─ directive
    > └─ layout
    > └─ plugins
    > └─ router
    > └─ store
    > └─ utils
    > └─ views
    └─ App.vue
    └─ main.js
    └─ permission.js
    └─ settings.js
```

设计工具

问卷设计区域

\* 问卷标题

\* 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

实时预览

问卷标题  题目类型  问卷

操作	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
+ 新增题目	29	测试	单选题	测试	必填	0	2025-07-08 16:02:29
修改	26	选择题1	单选题	Test_123	必填	1	2025-07-08 14:36:11
删除	30	测试	单选题	测试	必填	1	2025-07-08 16:02:29
选项管理	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	2025-07-08 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

填与问卷

问卷管理

问卷状态  + 创建时间  - 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	<span>...</span>
36	ceshi	qwr	草稿	3	0	2025-07-16 15:06:31	<span>...</span>
35	ceshi	qwr	草稿	3	0	2025-07-16 15:06:28	<span>...</span>
33	t0	aa	草稿	3	0	2025-07-16 08:49:21	<span>...</span>
32	t0	aa	草稿	3	0	2025-07-16 08:49:14	<span>...</span>
30	111	3333	已发布	1	1	2025-07-15 17:07:57	<span>...</span>
29	上传者测试	上传者测试	已发布	3	0	2025-07-09 11:41:34	<span>...</span>
28	测试	测试	已发布	4	0		<span>...</span>
27	Test_123	Test_123	草稿	3	0	2025-07-06 14:35:45	<span>...</span>

共 9 条 10条/页 < 1 > 前往 1

答卷管理

问卷  提交者  提交时间  - 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 >

## 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<any> => 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  $\leq 500\text{ms}$

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

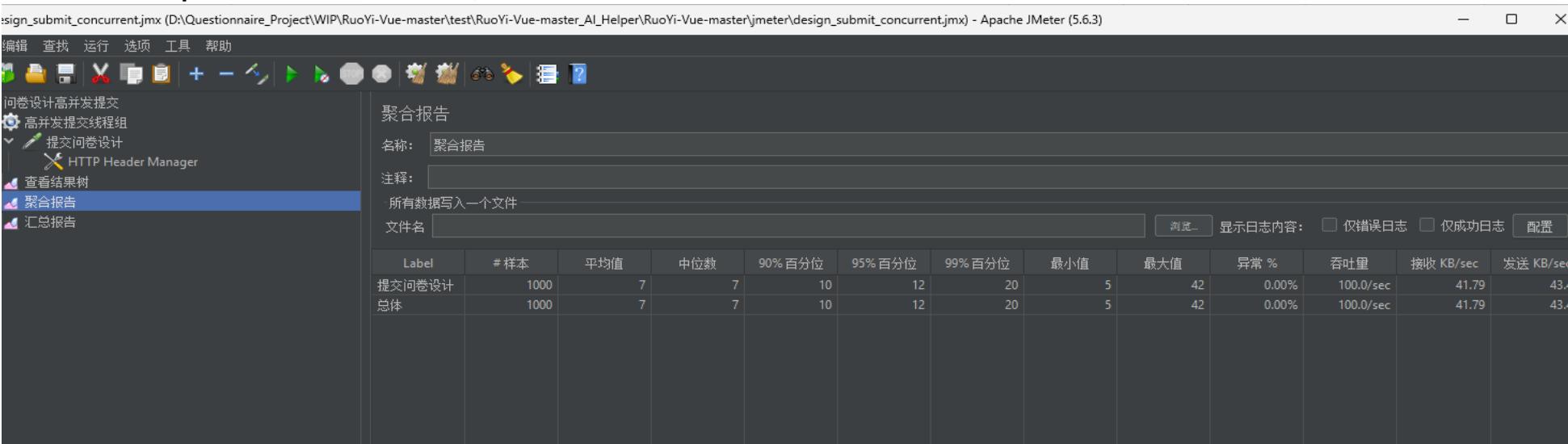
不同屏幕尺寸响应式

# 压力测试

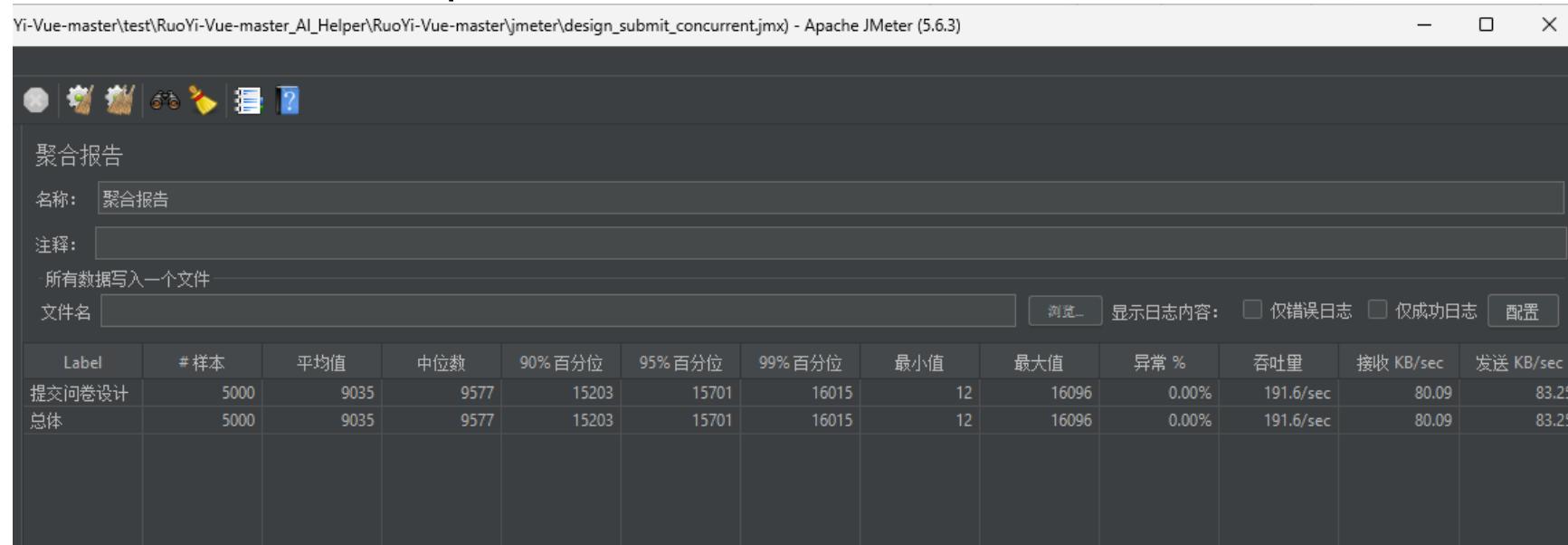
请求体内容：

名称: 提交问卷设计  
注释:  
高级  
Web服务器  
协议: http 服务器名称或IP: localhost 端口号: 8080  
HTTP请求  
POST 路径: /questionnaire/designer/save 内容编码:  
 自动重定向  跟随重定向  使用 KeepAlive  对POST使用multipart / form-data  与浏览器兼容的头  
参数 消息体数据 文件上传  
1 [{"questionnaire": {"title": "#随机生成问卷号"}, "description": "基础类测试", "status": "draft", "questions": [{"content": "问题1", "questionType": "single", "optionList": [{"optionLabel": "A"}, {"optionLabel": "B"}]}]}]

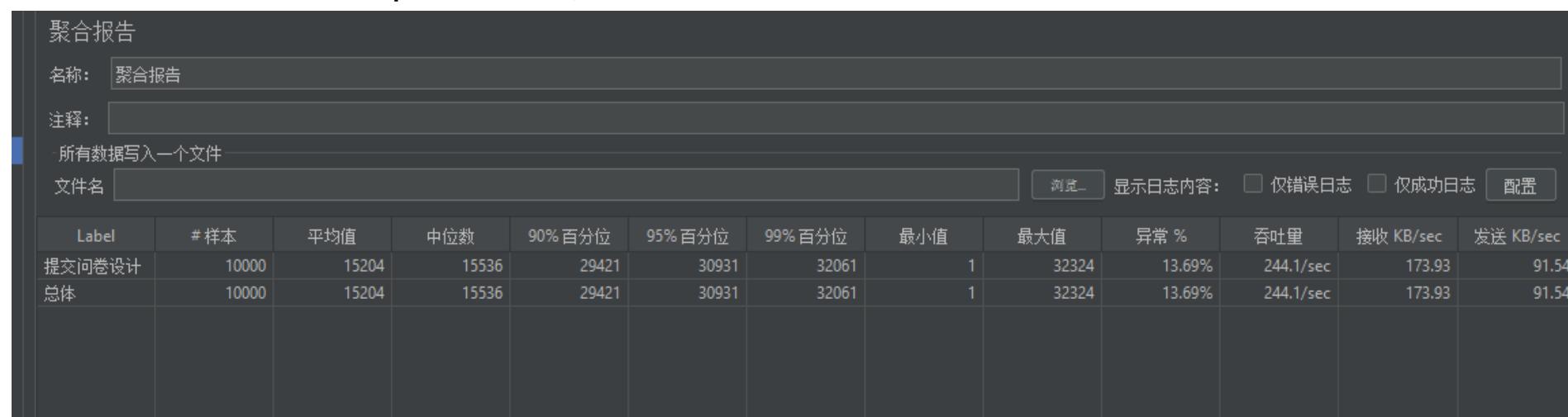
线程数1000 run up时间10s 无任何异常



线程数5000 run up时间10s 无任何异常



线程数10000 run up时间10s 异常率: 13.69%



线程数20000 run up时间10s 异常率: 55.23%

# 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

## 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