OJ在线判题系统简历写法

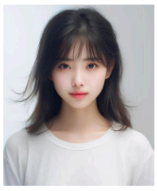
建议

注意，以下简历写法仅供参考，根据你自己的简历丰富度、以及对于项目的理解情况有选择地去写。如果你自己还没有实现项目或者不理解，建议赶紧跟着鱼皮的教程把它弄懂，再写到简历上！

此外，本项目系统设计的思路，设计模式、异步化、应用解耦的编程思想、Docker 和单体项目改造微服务的实践，其实是可以运用到你做的其他项目中的，可以把该项目的部分亮点和你之前的项目进行整合。

> 简历参考（后端版）：https://laoyujianli.com/share/lRBV5j <https://laoyujianli.com/share/lRBV5j>

老鱼简历



15372232131 mycv@gmail.com

应届毕业生 求职意向：后端开发工程师

教育经历

老鱼大学 计算机科学与技术 本科 2020-09 ~ 2024-07

全日制 计算机学院 杭州

专业技能

- 熟悉 Java 知识，如集合类、Java 安全管理器、进程类、编译执行原理、部分 JVM 参数
- 熟悉 SSM + Spring Boot 开发框架，能够使用 MyBatis Plus + MyBatis X 自动生成基础 CRUD 代码
- 熟悉 MySQL 数据库及库表设计，有过自定义索引的性能优化实践经历
- 熟悉并实践过多种设计模式，比如策略模式、工厂模式、模板方法模式、代理模式
- 熟悉 RabbitMQ 消息队列，有过手动消息确认、应用解耦、交换机队列定义、消息生产消费的实践
- 熟悉 Spring Cloud Alibaba 微服务相关技术，如 Nacos、Open Feign、Gateway 等，有过将单体项目改造为微服务项目的实践，并能使用网关实现全局跨域、全局聚合接口文档、全局鉴权等

项目经历

老鱼 OJ 在线判题系统 2023-04 ~ 2023-07

后端开发 杭州

项目介绍：

基于 Spring Cloud 微服务 + MQ + Docker 的编程题目评测系统。系统能够根据管理员预设的题目用例对用户提交的代码进行执行和评测；系统中 **自主实现的代码沙箱** 可作为独立服务供其他开发者调用。

主要工作：

1. 自主设计判题机模块的架构，定义了代码沙箱的抽象调用接口和多种实现类（比如远程 / 第三方代码沙箱），并通过 **静态工厂模式 + Spring 配置化** 的方式实现了对多种代码沙箱的灵活调用。
2. 使用 **代理模式** 对代码沙箱接口进行能力增强，统一实现了对代码沙箱调用前后的日志记录，减少重复代码。
3. 由于判题逻辑复杂、且不同题目的判题算法可能不同（比如 Java 题目额外增加空间限制），选用 **策略模式** 代替 if else 独立封装了不同语言的判题算法，提高系统的可维护性。
4. 通过编写 Java 脚本自测代码沙箱，模拟了多种程序异常情况并针对性解决，如使用 **黑白名单 + 字典树** 的方式实现了对敏感操作的限制。
5. 为保证沙箱宿主机的稳定性，选用 Docker 隔离用户代码，使用 **Docker Java** 库创建容器隔离执行代码，并通过 tty 和 Docker 进行传参交互，从而实现了更安全的代码沙箱。
6. 使用 Java 安全管理器和自定义的 **Security Manager** 对用户提交的代码进行权限控制，比如关闭写文件、执行文件权限，进一步提升了代码沙箱的安全性。

个人优势

- 有较强的文档阅读能力，曾阅读 Spring Cloud Alibaba 等官方文档自主学习，并能够运用到项目中
- 有较强的问题解决能力，能够利用 GitHub Issues 区、AI 工具、搜索引擎、Stack Overflow 等自主解决问题

> 简历参考（前端版）：https://laoyujianli.com/share/gyFwhi <https://laoyujianli.com/share/gyFwhi>

老鱼

13818996520 mycv@gmail.com
求职意向：前端开发工程师 上海市

教育经历

老鱼大学 计算机科学与技术 本科 2022-09 ~ 至今

专业技能

- 熟悉前端 Vue 3 开发，能够自定义通用的前端开发模板，包括 Vuex 状态管理、权限管理、布局切换、菜单生成等功能
- 熟悉前端代码规范，并能够使用 ESLint + Prettier + TypeScript 等技术保证前端项目质量
- 熟悉 Arco Design、ByteMD、 Monaco Editor 等组件的使用
- 能够使用 Vue-CLI 脚手架、openapi-typescript-codegen 代码生成器、VS Code、WebStorm IDE 等开发工具快速开发前端项目

项目经历

老鱼在线评测系统 2023-06 ~ 2023-10

项目介绍：

基于 Spring Boot + Spring Cloud 微服务 + Docker + Vue 3 + Arco Design 的编程题目在线评测系统。

在系统前台，管理员可以创建、管理题目；用户可以自由搜索题目、阅读题目、编写并提交代码。在系统后端，能够根据管理员设定的题目测试用例在 **自主实现的代码沙箱** 中对代码进行编译、运行、判断输出是否正确。其中，代码沙箱可以作为独立服务，提供给其他开发者使用。

主要工作：

1. 基于 Vue3 + Arco Design 组件库，自主实现了在线做题、题目检索和管理、提交列表、用户登录等页面。
2. 全局导航生成：基于 Vue Router 的路由配置文件自动生成导航菜单，并通过给路由的 meta 属性增加 hidden 字段实现集中控制页面的显隐。
3. 前后端联调：使用 openapi-typescript-codegen 工具根据后端 Swagger 接口文档自动生成请求后端的代码，大幅提高开发效率。
4. 基于 Webpack 整合了 Monaco Editor 开源代码编辑器组件，并进一步基于 ref 自行封装了可复用的 Editor 和 Viewer，实现了用户编写代码功能，支持多种语言的高亮。
5. 选用 ByteMD 开源 Markdown 文本编辑器组件，引入 gfm 插件并进一步自行封装了可复用的 Editor 和 Viewer，实现了题目内容及答案的编辑功能。
6. 使用 Arco Design 的 Table 组件实现了题目检索页面，并通过自定义插槽将后端返回的 JSON 数据解析为美观的格式。

个人优势

- 有较强的文档阅读能力，曾阅读 Spring Cloud Alibaba 等官方文档自主学习，并能够运用到项目中

- 有较强的问题解决能力，能够利用 GitHub Issues 区、AI 工具、搜索引擎、Stack Overflow 等自主解决问题

专业技能

后端

1. 熟悉 Java 知识，如集合类、Java 安全管理器、进程类、编译执行原理、部分 JVM 参数
2. 熟悉 SSM + Spring Boot 开发框架，能够使用 MyBatis Plus + MyBatis X 自动生成基础 CRUD 代码
3. 熟悉 MySQL 数据库及库表设计，有过自定义索引的性能优化实践经历
4. 熟悉常见的业务开发场景：比如用户分布式登录、操作异步化、API 接口服务设计等
5. 熟悉并实践过多种设计模式，比如策略模式、工厂模式、模板方法模式、代理模式
6. 熟悉 RabbitMQ 消息队列，有过手动消息确认、应用解耦、交换机队列定义、消息生产消费的实践
7. 熟悉 Spring Cloud Alibaba 微服务相关技术，如 Nacos、Open Feign、Gateway 等，有过将单体项目改造为微服务项目的实践，并能使用网关实现全局跨域、全局聚合接口文档、全局鉴权等
8. 实践过 Docker 技术，能够熟练运用命令行和 Docker Java 库两种方式操作 Docker
9. 熟练使用 Git、IDEA、Swagger、Navicat、VMware 虚拟机、远程开发等工具提高开发协作效率

前端

- 熟悉前端 Vue 3 开发，能够自定义通用的前端开发模板，包括 Vuex 状态管理、权限管理、布局切换、菜单生成等功能。
- 熟悉前端代码规范，并能够使用 ESLint + Prettier + TypeScript 等技术保证前端项目质量。
- 熟悉 Arco Design、ByteMD、 Monaco Editor 等组件的使用
- 能够使用 Vue-CLI 脚手架、openapi-typescript-codegen 代码生成器、VS Code、WebStorm IDE 等开发工具快速开发前端项目

项目经历

项目名称：XX OJ 在线判题系统

建议自己想个有区分度的名字，其他名称参考：

- XX 在线评测系统
- XX 题目测评系统
- XX 代码练习平台
- XX 编程学习系统
- XX OJ

在线访问：xxx（建议自己部署一下，提供可访问的、简短的线上地址）

GitHub：xxx（建议把项目放到代码仓库中，并且在主页文档里补充项目架构图等信息）

项目介绍

> 以下文字，括号里的内容表示可选项、或者鱼皮的备注，比如不熟悉前端的同学就不要写 Vue 等和前端相关的内容了

精简版

> 适合简历内容丰富的同学

基于 Spring Cloud 微服务 + MQ + Docker（+ Vue 3 + Arco Design）的编程题目评测系统。系统能够根据管理员预设的题目用例对用户提交的代码进行执行和评测；系统中 **自主实现的代码沙箱** 可作为独立服务供其他开发者调用。

详细版

> 适合简历内容不多的同学

基于 Spring Boot + Spring Cloud 微服务 + Docker（+ Vue 3 + Arco Design）的编程题目在线评测系统。

在系统前台，管理员可以创建、管理题目；用户可以自由搜索题目、阅读题目、编写并提交代码。

在系统后端，能够根据管理员设定的题目测试用例在 **自主实现的代码沙箱** 中对代码进行编译、运行、判断输出是否正确。

其中，代码沙箱可以作为独立服务，提供给其他开发者使用。

主要工作

> 根据自己的方向选 6 个左右去写并适当调整文案，灵活一点。 强烈建议结合下面的扩展思路多完善下项目，增加一些区分度！

前端

1. 基于 Vue3 + Arco Design 组件库，自主实现了在线做题、题目检索和管理、提交列表、用户登录等页面。
2. 使用 Vue-CLI 脚手架初始化项目，并自行开发了全局页面布局和通用前端项目模板，便于后续复用。
3. 使用 TypeScript + ESLint + Prettier + Husky 保证项目编码和提交规范，提高项目的质量。（虽然是由脚手架自动帮你整合了，但你要知道这些技术各自的作用）
4. 全局导航生成：基于 Vue Router 的路由配置文件自动生成导航菜单，并通过给路由的 meta 属性增加 hidden 字段实现集中控制页面的显隐。
5. 全局权限管理：通过给 Vue Router 路由的 meta 属性增加 access 字段来定义页面权限，然后通过 beforeEach 全局路由守卫集中校验用户进入页面的权限，并进一步将权限管理相关代码统一封装为 access.ts 模块，简化用户使用。
6. 全局状态管理：基于 Vuex 定义 User Module 实现了对登录用户的状态存储，并通过组合式 API（useStore）在页面中访问用户信息。
7. 前后端联调：使用 openapi-typescript-codegen 工具根据后端 Swagger 接口文档自动生成请求后端的代码，大幅提高开发效率。
8. 为提高前端开发效率，使用 IDEA 的 Live Templates 功能自定义了一套基础前端代码模板，能够通过快捷键高效生成代码。
9. 选用 ByteMD 开源 Markdown 文本编辑器组件，引入 gfm 插件（支持表格语法）并进一步自行封装了可复用的 Editor 和 Viewer，实现了题目内容及答案的编辑功能。
10. 基于 Webpack 整合了 Monaco Editor 开源代码编辑器组件，并进一步基于 ref 自行封装了可复用的 Editor 和 Viewer，实现了用户编写代码功能，支持多种语言的高亮。
11. 使用 Arco Design 的 Table 组件实现了题目检索页面，并通过自定义插槽将后端返回的 JSON 数据解析为美观的格式。

后端

1. 系统架构：根据功能职责，将系统划分为负责核心业务的后端模块、负责校验结果的判题模块、负责编译执行代码的可复用代码沙箱。各模块相互独立，并通过 API 接口和分包的方式实现协作。
2. 库表设计：根据业务流程自主设计用户表、题目表、题目提交表，并通过给题目表添加 userId 索引提升检索性能。（感兴趣的同学可以自己测试一下性能的提高比例）

3. 自主设计判题机模块的架构，定义了代码沙箱的抽象调用接口和多种实现类（比如远程 / 第三方代码沙箱），并通过 **静态工厂模式 + Spring 配置化** 的方式实现了对多种代码沙箱的灵活调用。
4. 使用 **代理模式** 对代码沙箱接口进行能力增强，统一实现了对代码沙箱调用前后的日志记录，减少重复代码。
5. 由于判题逻辑复杂、且不同题目的判题算法可能不同（比如 Java 题目额外增加空间限制），选用 **策略模式** 代替 if else 独立封装了不同语言的判题算法，提高系统的可维护性。
6. 使用 Java Runtime 对象的 exec 方法实现了对 Java 程序的编译和执行，并通过 **Process 类** 的输入流获取执行结果，实现了 Java 原生代码沙箱。
7. 通过编写 Java 脚本自测代码沙箱，模拟了多种程序异常情况并针对性解决，如使用守护线程 + Thread.sleep 等待机制实现了对进程的超时中断、使用 JVM -Xmx 参数限制用户程序占用的最大堆内存、使用 **黑白名单 + 字典树** 的方式实现了对敏感操作的限制。（选 1 - 2 种即可）
8. 使用 Java 安全管理器和自定义的 **Security Manager** 对用户提交的代码进行权限控制，比如关闭写文件、执行文件权限，进一步提升了代码沙箱的安全性。
9. 为保证沙箱宿主机的稳定性，选用 Docker 隔离用户代码，使用 **Docker Java** 库创建容器隔离执行代码，并通过 tty 和 Docker 进行传参交互，从而实现了更安全的代码沙箱。
10. 使用 VMware Workstation 虚拟机软件搭建 Ubuntu Linux + Docker 环境，并通过 JetBrains Client 连接虚拟机进行实时 **远程开发** ，提高了开发效率。
11. 为提高 Docker 代码沙箱的安全性，通过 HostConfig 限制了容器的内存限制和网络隔离，并通过设置容器执行超时时间解决资源未及时释放的问题。
12. 由于 Java 原生和 Docker 代码沙箱的实现流程完全一致（编译、执行、获取输出、清理），选用模板方法模式定义了一套标准的流程并允许子类自行扩展部分流程，提高代码一致性并大幅简化冗余代码。
13. 为防止用户恶意请求代码沙箱服务，（采用 API 签名认证的方式，）给调用方分配签名密钥，并通过校验请求头中的密钥保证了 API 调用安全。
14. 为保证项目各模块的稳定性，选用 Spring Cloud Alibaba 重构单体项目，（使用 Redis 分布式 Session 存储登录用户信息，并将项目）划分为用户服务、题目服务、判题服务、公共模块。
15. 使用阿里云原生脚手架初始化微服务项目，并结合 Maven 子父模块的配置，保证了微服务各模块依赖的版本一致性，避免依赖冲突。
16. 通过工具（JetBrains 的 Find Usage 功能 + 表格整理）梳理微服务间的调用关系，并通过 Nacos + OpenFeign 实现了各模块之间的相互调用，如判题服务调用题目服务来获取题目信息。
17. 使用 Spring Cloud Gateway 对各服务接口进行聚合和路由，保护服务的同时简化了客户端的调用（前端不用根据业务请求不同端口的服务），并通过自定义 CorsWebFilter Bean 全局解决了跨域问题。

18. 使用 Knife4j Gateway 在网关层实现了对各服务 Swagger 接口文档的统一聚合，无需通过切换地址查看各服务的文档。
19. 为保护内部服务接口，给接口路径统一设置 inner 前缀，并通过在网关自定义 GlobalFilter（全局请求拦截器）实现对内部请求的检测和拦截，集中解决了权限校验问题。
20. 为防止判题操作执行时间较长，系统选用异步的方式，在题目服务中将用户提交 id 发送给 RabbitMQ 消息队列，并通过 Direct 交换机转发给判题队列，由判题服务进行消费，异步更新提交状态。相比于同步，响应时长由 xx 秒减少至 xx 秒，且系统 qps 提升了 xx%（需要自己使用 JMeter 等工具进行测试）。
21. 基于自己二次开发的 Spring Boot 初始化模板 + MyBatis X 插件，快速生成图表、用户数据的增删改查。

通用

1. 在系统设计阶段，通过绘制时序图、功能模块图、流程图、分层架构图来帮助自己梳理业务流程。

扩展思路

> 需要大家自行实现

前端

1. 增强前端通用模板的能力，支持在路由文件中配置多套布局
2. 增强前端通用模板的能力，支持自动生成包含嵌套路由的多级导航栏
3. 优化题目管理页面，将 json 配置分为多列更美观地展示
4. 题目提交列表页面增加一个刷新、定时自动刷新的按钮，保证获取到题目提交的最新状态（前端轮询）
5. 增加对用户个人提交题目的管理页面
6. 增加对题目提交情况的统计分析页面，给管理员使用
7. 使用 Monaco Editor 的 diff editor 功能来对比用户代码和标准答案代码的区别

后端

1. 把微服务项目部署上线，参考教程：如何快速部署微服务项目？保姆级教程 <https://www.code-nav.cn/course/1788871761736998913/section/1789186997333499906?type=>
2. 增加题目的通过数、提交数统计，计算通过率
3. 限制代码沙箱中最多允许同时启动的 Docker 容器数，防止系统过载（甚至还可以用池化技术复用 Docker容器）
4. 更多类型的代码沙箱实现，比如使用 AI 进行判题？使用第三方服务（judge0 api）进行判题？
5. 反向压力：https://zhuanlan.zhihu.com/p/404993753 <https://zhuanlan.zhihu.com/p/404993753>，通过调用的服务状态来选择当前系统的策略（比如根据当前提交任务队列数来控制当前允许用户的最大提交数），从而最大化利用系统资源。
6. 限制单个用户的同时最大提交数，合理分配资源。
7. 限制单个用户的提交频率，可以通过 Redisson 或者 Sentinel 网关层限流实现。
8. 实现 ACM 模式（通过代码进行输入输出）的代码沙箱
9. 用同样的思路或者 Linux 的 cgroup 语法实现一种其他编程语言的代码沙箱
10. 实现 Special Judge 特判程序的逻辑
11. 给判题过程中的每个测试用例增加一个独立的内存、时间占用的统计
12. 可以使用 JWT Token 实现用户登录，在网关层面通过 token 获取登录信息，实现鉴权
13. 处理消息队列的消息重试，避免消息积压（可以选用死信队列）

个人评价

1. 有较强的文档阅读能力，曾阅读 Spring Cloud Alibaba 等官方文档自主学习，并能够运用到项目中。
2. 有较强的问题解决能力，能够利用 GitHub Issues 区、AI 工具、搜索引擎、Stack Overflow 等自主解决问题