伙伴匹配系统简历写法

建议

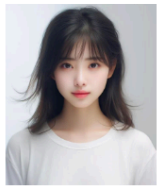
注意，以下简历写法仅供参考，根据你自己的简历丰富度、以及对于项目的理解情况有选择地去写。如果你自己还没有实现项目或者不理解，建议赶紧跟着鱼皮的教程把它弄懂，再写到简历上！

此外，伙伴匹配系统中的数据导入、数据匹配、队伍的加入管理逻辑，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，写同款：https://laoyujianli.com/share/1JJSAO
> <https://laoyujianli.com/share/1JJSAO>

老鱼

13818996520 mycv@gmail.com
实习中 求职意向：后端开发



教育经历

老鱼大学 计算机科学与技术 本科 2021-09 ~ 2025-06

专业技能

- 熟悉 Java 知识（如集合类、异常处理），能熟练运用 Lambda 表达式、Easy Excel、Hutool 工具库编程
- 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，能够通过创建索引、Explain 分析等方式优化性能
- 熟悉 Redis，实践过基于 Redis 的分布式缓存、分布式 Session 登录、基于 Redisson 的分布式锁
- 熟悉常见业务问题的解决方案：比如数据批量导入、基于标签的查询、缓存预热、定时任务等
- 熟练使用 Git、IDEA、ChatGPT、Swagger、Navicat 等工具提高开发协作效率

项目经历

老鱼伙伴匹配系统 2023-03 ~ 2023-07

项目介绍：

基于 Vue 3 + Spring Boot 2 的移动端网站，实现了用户管理、按标签检索用户、推荐相似用户、组队等功能。

主要工作：

1. 对于项目中复杂的集合处理（比如为队伍列表关联已加入队伍的用户），使用 Java 8 Stream API 和 Lambda 表达式来简化编码。
2. 为解决首次访问系统的用户主页加载过慢的问题，使用 Spring Scheduler 定时任务来实现缓存预热，并通过分布式锁保证多机部署时定时任务不会重复执行。
3. 为解决同一用户重复加入队伍、入队人数超限的问题，使用 Redisson 分布式锁来实现操作互斥，保证了接口幂等性。
4. 使用编辑距离算法实现了根据标签匹配最相似用户的功能，并通过优先队列来减少 TOP N 运算过程中的内存占用。
5. 自主编写 Dockerfile，并通过第三方容器托管平台实现自动化镜像构建及容器部署，提高部署上线效率。
6. 使用 Knife4j + Swagger 自动生成后端接口文档，并通过编写 ApiOperation 等注解补充接口注释，避免了人工编写维护文档的麻烦。

个人优势

- 有较强的文档阅读能力，曾阅读 Redisson 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、异常处理），能熟练运用 Lambda 表达式、Easy Excel、Hutool 工具库编程
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，能够通过创建索引、Explain 分析等方式优化性能
4. 熟悉 Redis，实践过基于 Redis 的分布式缓存、分布式 Session 登录、基于 Redisson 的分布式锁
5. 熟悉常见业务问题的解决方案：比如数据批量导入、基于标签的查询、缓存预热、定时任务等
6. 熟练使用 Git、IDEA、ChatGPT、Swagger、Navicat 等工具提高开发协作效率

前端

- 熟悉前端 Vue 3 开发，能熟练运用 Vue Router、Vant UI 等组件完成响应式页面开发
- 熟悉前端代码规范，并能够使用 TypeScript 等技术保证前端项目质量。
- 能够使用 Vite 脚手架、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 伙伴匹配系统

建议自己想个有区分度的名字，其他名称参考：

- XX 找伙伴
- XX 匹配系统
- 伙伴推荐站

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信息）

项目介绍

基于 Vue 3 + Spring Boot 2 的移动端网站，实现了用户管理、按标签检索用户、推荐相似用户、组队等功能。

主要工作

> 根据自己的方向选择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！

后端

1. 用户登录：使用 Redis 实现分布式 Session，解决集群间登录态同步问题；并使用 Hash 代替 String 来存储用户信息，节约了 xx% 的内存并便于单字段的修改。（需要自己实际测试对比数据，节省内存的原因是不用保存序列化对象信息或者 JSON 的一些额外字符串）
2. 对于项目中复杂的集合处理（比如为队伍列表关联已加入队伍的用户），使用 Java 8 Stream API 和 Lambda 表达式来简化编码。
3. 使用 Easy Excel 读取收集来的基础用户信息，并通过自定义线程池 + CompletableFuture 并发编程提高批量导入数据库的性能。实测导入 100 万行的时间从 xx 秒缩短至 xx 秒。（需要自己实际测试对比数据）
4. 使用 Redis 缓存首页高频访问的用户信息列表，将接口响应时长从 xx 秒缩短至 xx 秒。且通过自定义 Redis 序列化器来解决数据乱码、空间浪费的问题。
5. 为解决首次访问系统的用户主页加载过慢的问题，使用 Spring Scheduler 定时任务来实现缓存预热，并通过分布式锁保证多机部署时定时任务不会重复执行。
6. 为解决同一用户重复加入队伍、入队人数超限的问题，使用 Redisson 分布式锁来实现操作互斥，保证了接口幂等性。
7. 使用编辑距离算法实现了根据标签匹配最相似用户的功能，并通过优先队列来减少 TOP N 运算过程中的内存占用。
8. 自主编写 Dockerfile，并通过第三方容器托管平台实现自动化镜像构建及容器部署，提高部署上线效率。
9. 使用 Knife4j + Swagger 自动生成后端接口文档，并通过编写 ApiOperation 等注解补充接口注释，避免了人工编写维护文档的麻烦。

前端

1. 前端使用 Vant UI 组件库，并封装了全局通用的 Layout 组件，使主页、搜索页、组队页布局一致、并减少重复代码。
2. 基于 Vue Router 全局路由守卫实现了根据不同页面来动态切换导航栏标题， 并通过在全局路由配置文件扩展 title 字段来减少无意义的 if else 代码。
3. 使用 TypeScript 类型定义保证项目编码规范，提高项目的质量

扩展思路

> 需要大家自行实现

前端

1. 使用 Vercel Serverless 服务部署前端项目
2. 使用 cordova、taro、uniapp 等跨端框架，将前端项目改造为小程序
3. 支持通过地理位置来找附近的伙伴，前端可引入第三方地图组件（比如高德地图）

后端

1. 使用微信容器托管服务部署后端项目
2. 增加私聊、群聊功能，可以使用 WebSocket、Spring Boot STOMP、Vertx、Netty 实现
3. 支持通过地理位置来找附近的伙伴，后端使用 Redis GEO 数据结构来实现
4. 增加用户信息检测、举报恶意用户等功能

个人评价

1. 有较强的文档阅读能力，曾阅读 Redisson 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