智能面试刷题平台简历写法

建议

注意，以下简历写法仅供参考，根据自己的简历丰富度、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！**

本项目教程前 4 节为基础版，后面为扩展版，如果时间紧迫，可以先做完基础版，就可以将部分内容写到简历上了。

此外，本项目的部分知识，其实可以运用到你做的其他项目中，可以把该项目的部分亮点和你之前的项目进行整合。比如：

- Elasticsearch 分词搜索
- HotKey 热 key 缓存
- Sentinel 限流熔断
- Sa-Token 权限认证
- Nacos 动态黑名单

专业技能

后端

1. 熟悉 Java 特性，如集合类、自定义注解、异常处理、Lambda 编程等，遵循阿里 Java 规范保证项目质量。
2. 熟悉 Java 常用类库，如 Hutool 工具库、Easy Excel 表格处理、Logback 日志框架等。（Spring Boot 万用模板已经整合了这些库）
3. 熟悉 Spring Boot 框架，能利用 MyBatis Plus + MyBatis X 提高开发效率、基于 Sa-Token 实现鉴权。
4. 熟悉 MySQL 库表设计，能熟练编写高性能的 SQL，有自定义索引的性能优化实践。
5. 熟悉 Redis 及缓存设计，结合 Redisson 库实现各类业务（用 BitMap 实现签到、Lua 脚本实现计数）。
6. 熟悉 Elastic Stack，能基于 ES 实现分词搜索、Kibana 搭建数据看板，并掌握 ES 数据同步方案。
7. 掌握企业性能优化方案，如批处理操作优化、数据库连接池优化、多级缓存、HotKey 热点探测等。

8. 掌握企业安全优化方案，如 RBAC 鉴权、Sentinel 流控熔断、同端互斥登录、反爬虫、Nacos 黑白名单等。
9. 掌握基于 Nginx + Linux 管理面板的项目上线方法，并能通过配置反向代理解决跨域问题。
10. 掌握 Git、Maven、IDEA、Swagger、浏览器控制台等工具，并能通过 AI 编程助手提高开发效率。

其他可选（仅为推荐写法，本项目教程并没有讲解）：

1）熟悉并实践过多种设计模式，比如策略模式、工厂模式、单例模式、建造者模式等。

都是主流的设计模式，学习成本也不高，其中单例模式要重点掌握，多学习几种实现方案，推荐阅读 这道面试题 <https://www.mianshiya.com/bank/1801559627969929217/question/1801818854546284546> 。

2）熟悉 AI 应用开发，自主封装过通用 AI 模块，并能基于 SSE + RxJava 实现高性能的流式推送。

这是 AI 答题应用平台项目 <https://www.code-nav.cn/course/1790274408835506178> 中带大家实践过的，之所以建议大家写，因为 AI 是目前非常流行的前沿技术。

3）掌握基于 Serverless + Docker + CI / CD 的项目快速上线方法，并实操过灰度发布、自动扩缩容、回滚。

这是 AI 答题应用平台项目 <https://www.code-nav.cn/course/1790274408835506178> 中带大家实践过的，之所以建议大家写，因为项目上线也是企业特别看重的技能，增加区分度。

前端

1. 熟悉 React，开发过前端响应式项目模板，包括 Redux 状态管理、权限管理、布局切换、菜单生成等功能。
2. 熟悉服务端渲染和静态生成，自主开发上线过 Next.js 网站，掌握其 App Router 开发模式及特性（如缓存）。
3. 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。
4. 熟悉 Axios 请求库，能够自定义 Axios 实例，如通过全局响应拦截器实现统一的请求错误消息提示、根据环境区分请求域名等。
5. 熟悉 Ant Design、Ant Design Components、Echarts、ByteMD 组件库；以及 Day.js 等工具库。
6. 熟练运用脚手架、OpenAPI 代码生成器、VS Code、WebStorm、pnpm、Git 等工具快速开发前端项目。
7. 掌握基于 Nginx + Linux 管理面板的项目上线方法，并能通过配置反向代理解决跨域问题。

项目经历

项目名称：XX 面试刷题平台（比如 “鱼皮面试刷题平台”）

建议根据自己对项目的学习理解程度，自己想个有区分度的名字，其他名称参考：

- XX 刷题平台
- XX 学习助手
- XX 刷题助手
- XX 题库大师
- XX 刷题通
- XX 面试猎手
- 面试 X（比如面试鸭）
- XX 题海导航
- XX 面试达人
- XX 面试智库

如果能够自行给项目增加一些 AI 功能（比如 AI 生成题解），那么就可以给项目标题增加 “智能” 这两个字了，会更吸引面试官一些。比如 XX 智能面试刷题平台。

GitHub 代码地址：https://github.com/liyupi/mianshiya-next <https://github.com/liyupi/mianshiya-next>

建议大家也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、项目功能模块、技术选型等介绍信息。

但是注意不要抄袭鱼皮的写法！一定要加上自己的理解和扩展！ 否则一下就被面试官查出来了。

项目介绍

后端

基于 Spring Boot + Redis + MySQL + Elasticsearch（+ Next.js 服务端渲染）的面试刷题平台。管理员可以创建题库并批量关联题目；用户可以分词检索题目、在线刷题并查看刷题记录日历等。

项目已部署上线，并运用 Druid + HotKey + Sa-Token + Sentinel + Nacos 全面优化性能和安全性。

前端

基于 React + Next.js 服务端渲染 + Ant Design + Redux 的响应式面试刷题平台。管理员可以创建题库并批量关联题目；用户可以分词检索题目、在线刷题并查看刷题记录日历等。

开发中运用 ESLint + Prettier + TS 保证代码质量，并通过 OpenAPI 生成请求代码；最终基于 Nginx 部署上线。

💡 如果简历内容较少，可以多补充介绍一些功能。比如管理员可以批量关联题目到题库、批量删除题目；用户可以通过目录树快速切换题目等。

主要工作

根据自己对项目的掌握程度，选 6 个左右去写并适当调整文案，灵活一点。 **强烈建议参考鱼皮的项目扩展思路多完善下项目，增加一些区分度！**

前端

1. 基于 React + Ant Design 组件库，自主实现了在线刷题、题目搜索、题目管理、刷题记录等 10+ 页面。
2. 基于 Next.js 开发服务端渲染网站，并自主开发了响应式的通用前端项目模板，便于快速开发新项目。
3. 使用 TypeScript + ESLint + Prettier 保证项目编码规范，并自定义了 ESLint 规则，提高项目质量。（虽然是由脚手架自动帮忙整合了，但要知道这些技术各自的作用，也建议自己尝试下修改 ESLint 规则）
4. 全局通用布局：基于 Ant Design 的 ProLayout 组件实现，并在 Next.js 根布局中封装全局初始化逻辑。
5. 导航菜单生成：封装菜单配置文件 + 修改布局的菜单渲染函数实现，并通过扩展配置（如 hideInMenu）集中控制菜单的显隐。
6. 全局权限管理：封装 HOC 权限校验组件并装饰原有布局，开发者可通过修改菜单项的 access 为页面配置权限。
7. 全局状态管理：基于 Redux Toolkit 定义 UserStore 实现了对用户登录态的存储，并通过组合式 API（useSelector）在各页面中访问用户信息。
8. 全局请求：自定义 Axios 实例作为多环境通用的请求工具，并通过全局响应拦截器实现统一的请求错误消息提示。

9. 前后端联调：使用 OpenAPI 工具根据后端 Swagger 接口文档自动生成请求代码和 TS 数据模型，大幅提高开发效率。
10. 题目批量管理：通过 ProTable 的 render 函数 + 封装 Modal 组件实现，管理员可选中多题并添加到题库，提升操作效率。
11. 题目搜索：为优化 SEO，采用服务端渲染页面，并在首屏加载后，通过客户端渲染后续用户的分页查询。
12. 刷题记录：将添加记录功能封装为 hooks 简化调用；使用 ECharts 日历图展示全年刷题记录，并通过定制图表选项实现宽度自适应。
13. 题目展示：基于 MdViewer 组件实现 Markdown 题目内容展示，并通过导入主题 + CSS 样式覆盖优化阅读体验。
14. 性能优化：利用 Next.js 将题库大全页生成为静态网站，并通过分段配置选项控制缓存行为和时间。
15. 部署上线：基于 Next.js 的 standalone 模式构建，上传至 Linux 执行命令启动，并通过 Nginx 反向代理解决跨域。

后端

1. 基于自己开发的 Spring Boot 项目模板 + MyBatis X 插件 + 自定义代码生成器，快速生成各表基础业务代码。
2. 刷题记录：基于 Redis BitMap + Redisson 实现用户年度刷题记录的统计，相比数据库存储节约几百倍空间。并通过本地缓存 + 返回值优化 + 位运算进一步提升接口性能。（感兴趣的同学可以自行测试优化性能的效果）
3. 分词搜索：自主搭建 ES 代替 MySQL 模糊查询，并通过为索引绑定 ik 分词器实现了更灵活的分词搜索。（有时间的同学可以补充：使用 JMeter 测试后发现搜索性能提升 xx%）
4. 采用动静分离策略构建 ES 题目索引，仅存储需要检索且修改不频繁的字段，而将修改频繁的字段（如点赞数）从数据库中关联查询，从而减少了 ES 数据同步更新的成本、保证数据一致性。
5. 开发搜索功能时，使用 Kibana DevTools + DSL 调试 ES 的搜索效果，并使用 Spring Data Elasticsearch 的 QueryBuilder 组合查询条件，实现对题目的灵活查询。
6. 使用 Spring Scheduler 定时同步近期发生更新的 MySQL 题目到 ES，并通过唯一 id 保证每条数据同步的准确性。
7. 基于 MyBatis 的 batch 操作实现题目批量管理，并通过任务拆分 + CompletableFuture 并发编程提升批处理性能。（有时间的同学可以补充：实测批量删除 xx 道题目消耗的时间从 xx 秒缩短至 xx 秒）
8. 引入 Druid 连接池来监控慢 SQL，并通过调整连接数配置，进一步提升了批量操作的性能。
9. 使用 Caffeine 本地缓存提升题库查询性能，并通过接入 Hotkey 并配置热 key 探测规则来自动缓存热门题目，防止瞬时流量击垮数据库。

10. 为保护系统，基于 Sentinel 注解 + Dashboard 对获取题库列表接口进行限流，并通过 fallbackHandler 配置熔断，调用异常率超过 10% 时直接返回本地缓存。
11. 为保护系统，基于 Sentinel 的热点参数限流机制对单 IP 获取题目进行流控，并通过拉模式配置将规则持久化到本地文件。
12. 为限制恶意用户访问，基于 WebFilter + BloomFilter 实现 IP 黑名单拦截，并通过 Nacos 配置中心动态更新黑名单，便于维护。
13. 为防止账号共享，通过 UserAgent 识别用户设备，并基于 Sa-Token 快速实现同端登录冲突检测。
14. 设计分级反爬虫策略：基于 Redis 实现用户访问题目频率统计，并通过 Lua 脚本保证原子更新，超限时自动给管理员发送告警和封禁用户，有效防止内容盗取。
15. 通过宝塔 Linux 的 Java 项目管理器部署 jar 包，并通过 Nginx 配置反向代理解决跨域问题。
16. 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。

适合入门同学补充：

1. 库表设计：根据业务设计用户、题库、题目、关联表，其中题目标签采用 JSON 数组存储，便于维护；并通过给题库题目关联表添加联合索引来提升检索性能。
2. 通过 MyBatis Plus 的 LambdaQueryWrapper 构造数据库查询，简化编码。
3. 通过 @Lazy 注解解决题目服务和题库服务互相引用导致的循环依赖问题。
4. 通过实现 CommandLineRunner 接口单次执行全量同步数据到 ES 的任务。
5. 使用 AopContext.currentProxy 获取服务代理类，解决了 Spring 事务失效的问题。

有时间的同学可自行扩展实现：

1. 自主搭建 Kibana 并配置 index pattern 和看板，实现对 Elasticsearch 中题目数据的可视化管理。

个人评价

1. 自学和理解能力较强，能阅读文档高效自学，如 ES、HotKey、Sentinel、Nacos 等，并快速编写 Demo。
2. 问题解决能力较强，能够利用 GitHub Issues 区、AI 工具、搜索引擎、Stack Overflow 等自主解决问题。
3. 有企业项目实践经验，能通过调研和对比来设计解决方案，并从性能、安全性、可用性等多个角度优化项目。

4．注重积累沉淀，维护了一套项目开发模板和个人工具箱，可通过复用快速完成新项目开发。

5．有不错的产品思维和数据保护意识，比如能够想到给项目增加同端互斥登录和反爬虫。