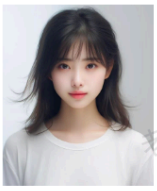
聚合搜索平台简历写法

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，这个项目的聚合搜索、数据抓取、数据同步等功能，其实是可以集成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/8ah1dx
> <https://laoyujianli.com/share/8ah1dx>



老鱼

13818996520 mycv@gmail.com

应届毕业生 意向职位：后端开发 上海

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 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
- 熟悉常见的业务开发场景：比如 jsoup 爬虫、分词搜索功能、数据同步、多数据源管理等
- 熟悉并实践过多种设计模式，比如门面模式、适配器模式、注册器模式
- 熟悉 Elastic Stack 技术栈，能用 Elasticsearch 实现分词搜索、用 Kibana 实现 ES 数据看板搭建
- 能够使用 Canal、Logstash、定时任务、双写等方式实现 Elasticsearch 和 MySQL 的同步
- 熟练使用 Git、IDEA、Swagger、Navicat 等工具提高开发协作效率，使用 JMeter 压测系统

项目经历

老鱼聚合搜索平台 2023-12 ~ 2024-03

项目介绍：

基于 Spring Boot + Elastic Stack 的一站式信息聚合搜索平台。用户可在同一页面集中搜索出不同来源、不同类型的内容，提升搜索体验。企业也可以直接将各项目的数据接入搜索平台，复用同一套搜索后端，提升开发效率、降低系统维护成本。

主要工作：

1. 基于自己二次开发的 Spring Boot 初始化模板 + MyBatis X 插件，快速生成基本数据源的增删改查（比如用户、文章）。
2. 使用 HttpClient 请求 **离线** 获取外部网站的文章，并使用 Hutool 的 JSONUtil 解析和预处理文章，最终入库。
3. 为解决文章搜不出的问题，自主搭建 Elasticsearch 来代替 MySQL 的模糊查询，并通过为索引绑定 ik 分词器实现了更灵活的分词搜索，且使用 JMeter 测试后发现搜索性能提升 30%。
4. 为了更方便地管理 Elasticsearch 中的数据，自主搭建 Kibana 并配置 index pattern 和看板，实现对文章数据的可视化管理。
5. 为了更方便地管理 Elasticsearch 中的数据，自主搭建 Kibana 并配置 index pattern 和看板，实现对文章数据的可视化管理。
6. 构建 ES 文章索引时，采用动静分离的策略，只在 ES 中存储要检索的、修改不频繁字段（比如文章）用于检索，而修改频繁的字段（比如点赞数）从数据库中关联查出，从而减少了 ES 数据更新和同步的成本、保证数据一致性。

个人优势

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

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、Hutool、HttpClient、Apache Utils 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. 熟悉常见的业务开发场景：比如 jsoup 爬虫、分词搜索功能、数据同步、多数据源管理等
5. 熟悉并实践过多种设计模式，比如门面模式、适配器模式、注册器模式
6. 熟悉 Elastic Stack 技术栈，能用 Elasticsearch 实现分词搜索、用 Kibana 实现 ES 数据看板搭建
7. 能够使用 Canal、Logstash、定时任务、双写等方式实现 Elasticsearch 和 MySQL 的同步
8. 熟练使用 Git、IDEA、Swagger、Navicat 等工具提高开发协作效率，使用 JMeter 压测系统。

前端

- 熟悉前端 Vue 3 开发，能熟练运用 Vue Router、Ant Design Vue 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 ESLint + Prettier + TypeScript 等技术保证前端项目质量。
- 能够使用 Vue-CLI 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 聚合搜索平台

建议自己想个有区分度的名字，其他名称参考：

- XX 搜索平台
- XX 信息聚合平台
- XX 搜索

- XX 信息墙
- XX 聚合网
- XX 聚合搜索中台

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

> 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 3 和前端相关的内容了

基于 Spring Boot + Elastic Stack（+ Vue 3）的一站式 XX 信息聚合搜索平台。用户可在同一页面集中搜索出不同来源、不同类型的内容（建议具体列举具体的数据类别，比如文章、图片、用户、专栏、视频等），提升搜索体验。

企业也可以直接将各项目的数据接入搜索平台，复用同一套搜索后端，提升开发效率、降低系统维护成本。（这里可以补充：自己已经将这个搜索后端应用到了其他的项目中）

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！

后端

1. 基于自己二次开发的 Spring Boot 初始化模板 + MyBatis X 插件，快速生成基本数据源的增删改查（比如用户、文章）。
2. 数据源获取：
    - 使用 HttpClient 请求 **离线** 获取外部网站的文章，并使用 Hutool 的 JSONUtil 解析和预处理文章，最终入库。
    - 使用 jsoup **实时** 请求 bing 搜索接口获取图片，并使用 CSS Selector 语法解析和预处理图片信息，最终返回给前端。
3. 为实现多类数据源的整体搜索，使用 **门面模式** 在后端对各类数据源的搜索结果进行聚合，统一返回给前端，减少了前端请求次数（N 次到 1 次）以及前端开发复杂度。并通过 CompletableFuture 并发搜索各数据源进一步提升搜索接口性能，实测整体响应时长 **由 300ms 减少为 xx ms** （一定要自己测试，如果效果不明显，可以加大数据量）。

4. 为提高聚合搜索接口的通用性，首先通过定义数据源接口来实现统一的数据源接入标准（比如新数据源必须支持分页）；当新数据源（比如视频）要接入时，只需使用适配器模式对其数据查询接口进行封装、以适配数据源接口，无须修改原有代码，提高了系统的可扩展性。
5. 为减少代码的圈复杂度，使用注册器模式代替 if else 来管理多个数据源对象，调用方可根据名称轻松获取对象，（使用 IDEA MetricReloaded 插件）实测圈复杂度由 XX 减少为 XX。
6. 为解决文章搜不出的问题，自主搭建 Elasticsearch 来代替 MySQL 的模糊查询，并通过为索引绑定 ik 分词器实现了更灵活的分词搜索，且使用 JMeter 测试后发现搜索性能提升 xx%（xx qps 到 xx qps）。
7. 构建 ES 文章索引时，采用动静分离的策略，只在 ES 中存储要检索的、修改不频繁字段（比如文章）用于检索，而修改频繁的字段（比如点赞数）从数据库中关联查出，从而减少了 ES 数据更新和同步的成本、保证数据一致性。
8. 为了更方便地管理 Elasticsearch 中的数据，自主搭建 Kibana 并配置 index pattern 和看板，实现对文章数据的可视化管理。
9. 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 ES 内文章的灵活查询（比如查询同时查询标题和文章中带有指定关键字的内容）。
10. 以下 2 选 1：
    - 使用 Spring Scheduler 定时同步近 5 分钟内发生更新的 MySQL 的文章数据到 ES，通过唯一 id 来保证每条数据同步的准确性。
    - 自主搭建 Logstash 实现每分钟同步 MySQL 的文章数据到 ES，并通过指定 tracking_column 为更新时间字段解决重复更新的问题。

其他：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. 基于 Vue 3 + Ant Design Vue 实现响应式页面开发，使用 Tab 组件 + Vue Router 动态路由实现统一页面布局，通过用户点击 Tab 时更改路由来切换各类数据（文章、图片、用户）的搜索结果，并选用不同的组件进行展示。
2. 为解决刷新页面后搜索结果丢失的问题，定义 searchParams 响应式变量来集中保存当前的搜索条件（比如关键词、搜索类别、分页），并通过 Vue Router 的 query 参数将搜索条件同步到 url 的 querystring 中。
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）

扩展思路

> 需要大家自行实现

后端

1. 可以使用 Canal 监听 MySQL，并将数据实时同步到 Elasticsearch（一定要能突出实时同步的必要性，比如要搜索的数据频繁改动）
2. 可以使用 Redis 对查询数据进行缓存，并测试下缓存命中率和性能提升的效果
3. 可以记录并统计用户的搜索词，从而实现热搜、词云图、看板分析等功能。
4. 为 Elasticsearch 的 ik 插件自定义词典，实现系统内一些关键词的灵活分词查询。（词典如何生成也是需要考虑的问题，可以从热搜词下手）
5. 使用 Elasticsearch 的 highlight 语法实现搜索词高亮。
6. 使用 Elasticsearch 的 suggest 语法实现搜索建议。
7. 聚合搜索接口中，可以使用自定义线程池来实现并发，线程池的参数配置为 IO 密集型，即线程数多一些。
8. 可以使用 Guava Retrying 重试库来保证调用第三方接口的稳定性，可以实测一下接口可用性有几个 9。
9. 可以使用定时任务定期去做离线文章爬虫。

前端

1. 为解决用户频繁输入带来的搜索请求性能浪费，使用 Lodash 库实现搜索功能的防抖。
2. 可以尝试使用 umi 的 one-api 插件，根据后端的 swagger 接口文档自动生成请求代码。
3. 对于各类数据的查询请求，可以使用懒加载或骨架屏来优化用户体验。
4. 可以尝试实现文章和图片的无限滚动列表，并解决数据量过大带来的性能问题。
5. 可以在前端使用 Vuex 或 Pinia 等状态管理工具实现搜索记录功能。

个人评价

1. 有较强的文档阅读能力，曾阅读 Elastic Stack、Spring Data Elasticesarch 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

url=https%3A%2F%2Fwww.yuque.com%2Fu37765561%2Fak85bt%2Fa119be2fcaaaa1c99d0206c9c