智能 BI 项目简历写法

智能BI项目简历写法

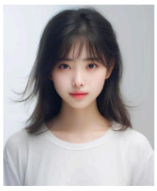
建议

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，本项目有关 AIGC 的思路和实践、异步化的编程思想，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/zC5Xcc
> <https://laoyujianli.com/share/zC5Xcc>

老鱼简历



13818996520 mycv@gmail.com
应届毕业生 求职意向：后端开发 深圳

教育经历

老鱼大学 计算机科学与技术 本科 2020-09 ~ 2024-07

专业技能

- 熟悉 Java 知识（如集合类、异常处理），能熟练运用 Lambda、Hutool、Easy Excel、Apache Utils 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、分库分表优化性能
- 熟悉并发编程，能够使用 CompletableFuture 等 JUC 类、自定义线程池实现并发和操作异步化
- 熟悉 Redis，实践过基于 Redis 的分布式缓存、分布式锁、基于 Redisson 的分布式限流
- 熟悉 RabbitMQ 消息队列，有手动消息确认、消息持久化、交换机队列定义、消息生产消费的实践

项目经历

老鱼智能数据分析平台 2023-12 ~ 2024-03

项目介绍：

基于 Spring Boot + MQ + AIGC + React 的智能数据分析平台。区别于传统 BI，用户只需要导入原始数据集、并输入分析诉求，就能自动生成可视化图表及分析结论，实现数据分析的降本增效。

主要工作：

1. 后端自定义 Prompt 预设模板并封装用户输入的数据和分析诉求，通过对接 AIGC 接口生成可视化图表 json 配置和分析结论，返回给前端渲染。
2. 由于 AIGC 的输入 Token 限制，使用 Easy Excel 解析用户上传的 XLSX 表格数据文件并压缩为 CSV，实测提高了 20% 的单次输入数据量、并节约了成本。
3. 为保证系统的安全性，对用户上传的原始数据文件进行了后缀名、大小、内容等多重校验。
4. 为防止某用户恶意占用系统资源，基于 Redisson 的 RateLimiter 实现分布式限流，控制单用户访问的频率。
5. 由于 AIGC 的响应时间较长，基于自定义 IO 密集型线程池 + 任务队列实现了 AIGC 的并发执行和异步化，提交任务后即可响应前端，提高用户体验。
6. 由于本地任务队列重启丢失数据，使用 RabbitMQ 来接受并持久化任务消息，通过 Direct 交换机转发给解耦的 AI 生成模块消费并处理任务，提高了系统的可靠性。

个人优势

- 有较强的文档阅读能力，曾阅读 RabbitMQ 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、异常处理），能熟练运用 Lambda、Hutool、Easy Excel、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、分库分表优化性能
4. 熟悉并发编程，能够使用 CompletableFuture 等 JUC 类、自定义线程池实现并发和操作异步化
5. 熟悉 Redis，实践过基于 Redis 的分布式缓存、分布式锁、基于 Redisson 的分布式限流
6. 熟悉 RabbitMQ 消息队列，有过手动消息确认、消息持久化、交换机队列定义、消息生产消费的实践
7. 熟练使用 Git、IDEA、ChatGPT、Swagger、Navicat 等工具提高开发协作效率

前端

- 熟悉 React 框架开发，能够根据业务定制前端模板，比如封装全局异常处理逻辑。
- 熟悉前端代码规范，并能够使用 ESLint + Prettier + TypeScript + Husky 等技术保证前端项目质量。
- 熟悉 Ant Design、Echarts、Lodash 等组件库和工具的使用
- 能够使用 Ant Design Pro 框架、Umi OpenAPI 代码生成、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 智能 BI 平台

建议自己想个有区分度的名字，其他名称参考：

其他名称参考：

- XX 智能数据分析平台
- XX AI 数据可视化平台

- XX AIGC BI 系统
- 数据小智

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 React 等和前端相关的内容了

基于 Spring Boot + MQ + AIGC（+ React）的智能数据分析平台。区别于传统 BI，用户只需要导入原始数据集、并输入分析诉求，就能自动生成可视化图表及分析结论，实现数据分析的降本增效（或者降低数据分析的人工成本、提高数据分析效率等）。

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 Ant Design Pro 脚手架快速搭建初始项目，并根据业务定制项目模板，如封装全局异常处理逻辑。
2. 使用 TypeScript + ESLint + Prettier + Husky 保证项目编码和提交规范，提高项目质量。
3. 使用 Umi OpenAPI 插件，根据后端 Swagger 接口文档自动生成请求 service 层代码，大幅提高开发效率。
4. 选用兼容性较好的 Echarts 库，接收后端 AI 生成的动态 json 自动渲染可视化图表

后端

1. （业务流程：）后端自定义 Prompt 预设模板并封装用户输入的数据和分析诉求，通过对接 AIGC 接口生成可视化图表 json 配置和分析结论，返回给前端渲染。
2. 由于 AIGC 的输入 Token 限制，使用 Easy Excel 解析用户上传的 XLSX 表格数据文件并压缩为 CSV，实测提高了 20% 的单次输入数据量、并节约了成本。

3．为保证系统的安全性，对用户上传的原始数据文件进行了后缀名、大小、内容等多重校验（其实能被 Easy Excel 解析成功，内容基本是没问题的）
4．为防止某用户恶意占用系统资源，基于 Redisson 的 RateLimiter 实现分布式限流，控制单用户访问的频率。
5．考虑到单个图表的原始数据量较大，基于 MyBatis + 业务层构建自定义 SQL 实现了对每份原始数据的分表存储，提高查询性能 XX %（可以实测一下，把所有图表的大量数据一起 select by Id，和直接 select from 单张数据表）和系统的可扩展性。
6．由于 AIGC 的响应时间较长，基于自定义 IO 密集型线程池（要能讲清楚为什么是 IO 密集型线程池）+ 任务队列实现了 AIGC 的并发执行和异步化，提交任务后即可响应前端，提高用户体验。（或者说：支持更多用户排队而不是无限给系统压力导致提交失败）
7．由于本地任务队列重启丢失数据，使用 RabbitMQ（分布式消息队列）来接受并持久化任务消息，通过 Direct 交换机转发给解耦的 AI 生成模块消费并处理任务，提高了系统的可靠性。

其他（几乎万用）：

1．基于自己二次开发的 Spring Boot 初始化模板 + MyBatis X 插件，快速生成图表、用户数据的增删改查
2．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。

扩展思路

> 需要大家自行实现

前端

1．支持用户查看图表原始数据，对于大量数据的展示，可以用开源长列表组件进行优化
2．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。
3．支持创建看板，可以将多个图表添加到看板中，并且拖拽布局图表的位置
4．支持分组（或者分标签）查看和检索图表。
5．增加更多可选参数来控制图表的生成，比如图表配色等。
6．支持用户对失败的图表进行手动重试
7．支持编辑生成后的图表的信息，比如可以使用代码编辑器来编辑 Echarts 图表配置代码。
8．图表管理页面增加一个刷新、定时自动刷新的按钮，保证获取到图表的最新状态（前端轮询）
9．补充更多传统 BI 拥有的功能，把智能分析作为其中一个子模块。

后端

1. 图表数据分表存储，提高查询灵活性和性能
2. 可以使用 AI 来整理和压缩原始数据，进一步提高输入的数据数量。
3. 限制用户同时生成图表的数量，防止单用户抢占系统资源
4. 统计用户生成图表的次数，甚至可以添加积分系统，消耗积分来智能分析
5. 由于图表数据是静态的，很适合使用 Redis 缓存来提高加载速度。
6. 使用死信队列来处理异常情况，将图表生成任务置为失败
7. 给任务的执行增加 guava Retrying 重试机制，保证系统可靠性和稳定性。
8. 提前考虑到 AI 生成错误的情况，在后端进行异常处理（比如 AI 说了多余的话，提取正确的字符串）
9. 如果说任务根本没提交到队列中（或者队列满了），可以用定时任务把失败状态的图表放到队列中（补偿）
10. 建议给任务的执行增加一个超时时间，超时自动标记为失败（超时控制）
11. 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 AI 服务的当前任务队列数来控制本系统的核心线程数），从而最大化利用系统资源。
12. 任务执行成功或失败，给用户发送实时消息通知（可以使用 websocket、server side event 等技术）

个人评价

1. 有较强的文档阅读能力，曾阅读 RabbitMQ 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