# **《人才招募系统软件需求规格说明》设计风格分析**

## **一、常见软件设计风格对比**

| ****风格**** | ****核心特点**** | ****典型应用场景**** | ****模块交互方式**** | ****扩展性**** | ****性能特点**** |
| --- | --- | --- | --- | --- | --- |
| ****分层风格**** | 将系统划分为表现层、业务层、数据层等，每层职责明确，上层通过接口依赖下层服务 | 企业级应用（如电商平台、ERP 系统） | 接口调用，层间松耦合 | 良好，可独立扩展层 | 层间调用可能引入延迟，需优化接口 |
| ****管道 - 过滤器风格**** | 数据流经独立过滤器处理，通过管道传递数据，过滤器间无直接依赖 | 数据处理流水线（如编译器、日志处理） | 数据流驱动，通过缓冲区传递数据 | 优秀，可动态增减过滤器 | 数据转换开销较高，适合批处理场景 |
| ****面向对象风格**** | 基于对象封装数据与行为，通过继承、多态实现代码复用 | 通用应用开发（如桌面软件、移动应用） | 对象间消息传递，依赖接口实现 | 良好，可通过子类扩展功能 | 运行时开销较低，需设计合理类结构 |
| ****事件驱动风格**** | 通过事件发布 - 订阅机制触发组件交互，组件监听特定事件并响应 | 实时系统（如消息队列、实时通信） | 事件监听与回调 | 灵活，可动态注册事件监听器 | 异步处理性能高，需管理事件顺序 |
| ****微服务风格**** | 将系统拆分为独立部署的微服务，通过轻量级协议（如 HTTP/REST）通信 | 分布式系统（如 Netflix、云原生应用） | 服务间 HTTP / 消息通信 | 极强，可独立扩展服务 | 网络延迟可能影响性能，需服务治理 |
| ****批处理风格**** | 按顺序处理数据批次，前一阶段输出为后一阶段输入，强依赖处理顺序 | 批量数据处理（如报表生成、日志分析） | 文件 / 数据集传递，顺序执行 | 较差，修改需重跑全流程 | 适合离线处理，吞吐量较高 |

## **二、《人才招募系统软件需求规格说明》风格分析**

****（一）关键特征提取****

分层架构明确性

前端层：负责用户界面交互，采用响应式设计适配 PC / 移动端，包含个人用户端、企业用户端、管理端三类界面。

后端服务层：通过模块化设计实现业务逻辑，划分为认证模块、用户模块、招聘流程模块、数据统计模块等，模块间通过接口提供服务（如用户模块为其他模块提供身份验证接口）。

数据存储层：采用混合存储方案：

MySQL：存储用户信息、岗位详情等结构化数据，支持事务和复杂查询；

阿里云 OSS：存储简历 PDF、营业执照等非结构化文件，数据库仅记录文件路径；

Redis：缓存用户登录态、热门岗位列表等高频访问数据，提升响应速度。

模块接口标准化

前后端交互：遵循 RESTful 风格，通过 HTTP 方法操作资源，数据格式为 JSON。

示例：个人用户搜索职位调用GET /api/positions?keyword=Java&city=北京；

企业用户发布岗位通过POST /api/jobs接口提交结构化数据。

内部模块交互：通过接口约定实现松耦合，如简历管理模块向招聘流程模块提供getResumeById(resumeId)接口，避免直接操作数据库。

数据处理分层逻辑

存储分层：根据数据访问频率和类型划分存储介质，热数据（如登录令牌）存于 Redis，冷数据（如历史简历）归档至 OSS。

流程分层：简历解析功能分为 “文件读取→文本解析→信息提取→入库存储” 多层处理，每层由独立组件完成（如 Apache Tika 解析文件，NLP 模型提取关键词）。

****（二）排除其他风格的依据****

| 风格 | 排除理由 |
| --- | --- |
| 管道 - 过滤器 | 系统核心流程（如招聘流程状态机管理）依赖业务逻辑状态，而非无状态的数据流过滤，过滤器间需共享数据库状态。 |
| 事件驱动 | 核心交互为同步接口调用（如登录后立即跳转页面），仅通知功能采用异步事件（如邮件通知），非架构主导。 |
| 微服务 | 系统为单体应用，模块间未拆分为独立服务，共享同一数据库和代码库，不符合微服务 “独立部署” 原则。 |
| 批处理 | 主要处理实时请求（如即时职位搜索、简历投递），非批量离线数据处理，流程需实时反馈结果。 |

### **三、结论：分层风格主导，辅以面向对象设计**

该文档体现的软件设计风格以 \*\* 分层架构（Layered Architecture）\*\* 为核心，通过 “前端 - 后端 - 数据” 三层划分实现职责分离，并利用面向对象设计封装模块内部逻辑（如用户类、岗位类）。其设计特点如下：

优势：分层结构清晰，便于团队分工开发（如前端组、后端组、数据库组）；模块松耦合，支持独立测试和维护（如替换简历解析引擎不影响其他模块）；数据分层存储优化性能（如 Redis 缓存减少数据库压力）。

适配场景：中小型 Web 应用开发，满足 “结构清晰、可维护性强” 的教学项目需求，同时支持后期功能扩展（如增加 AI 推荐模块时仅需修改业务层）。