# 架构设计文档

## 1. 架构概述

本Web应用采用分层架构设计，遵循高内聚低耦合的原则，主要分为数据层、服务层、表示层及安全层。系统通过RESTful API实现前后端分离，确保系统的可维护性、扩展性与可测试性，从而提供灵活的服务和良好的用户体验。

## 2. 系统组件

### 2.1 数据层

* **数据库管理系统**：生产环境可选择MySQL作为主要的关系型数据库，负责存储用户信息、选手信息和点赞数据。
* **数据库设计**：运用Entity-Relationship (ER) 模型进行数据建模，利用ORM框架（如Hibernate）简化与数据库的交互，并提高代码的可读性和可维护性。同时，考虑使用数据库索引和视图优化查询性能。

### 2.2 服务层

* **服务模块**：
  + 功能：实现信息管理。
  + 设计：通过RESTful API暴露服务接口，支持多种用户角色的逻辑。

### 2.3 表示层

* **前端框架**：使用Vue 3框架构建SPA（单页面应用），结合Vue Router和Vuex实现组件化管理和状态管理，确保良好的用户体验。
* **用户界面**：设计响应式用户界面，包括登录页、注册页、主页、选手详情页等，同时优化为适配不同设备以提升可用性。

## 3. 安全和隐私

* **身份认证**：采用JWT（JSON Web Tokens）进行用户身份验证，确保数据安全传输与用户会话管理。
* **数据加密**：用户密码采用加密存储，并设置适当的工作因子以增强安全性。
* **注入防护**：使用参数化查询防止SQL注入，前端利用Vue 3的安全特性，以及后端实施输入验证和输出编码，防御XSS与CSRF攻击。

## 4. 技术栈

* **前端**：Vue 3 / Vue Router / Vuex
* **后端**：Java / Spring Boot
* **数据库**：MySQL
* **安全**：JWT / 加密存储
* **容器化**：Docker

## 5. 部署和维护

* **容器化部署**：可利用Docker实现服务的容器化，确保开发、测试与生产环境的一致性，以及快速的服务交付。
* **CI/CD流程**：后期可结合Jenkins或GitLab CI实现自动化构建、测试及部署，减少人为干预，提高部署效率和软件质量。

## 6. 测试策略

* **单元测试**：采用JUnit框架为后端服务编写单元测试，确保核心业务逻辑的正确性。
* **集成测试**：可使用Postman及Spring MockMvc对RESTful API进行集成测试，以验证各模块间的交互。
* **性能测试**：可使用Apache JMeter进行负载测试与性能监控，评估系统在高并发场景下的表现与稳定性。
* **安全测试**：可使用扫描工具进行安全漏洞扫描，并定期进行渗透测试，以发现并修复安全隐患。