# 后端实操大题：微服务架构用户权限管理系统实现指南

## 一、项目目标

基于微服务架构实现简化版用户权限管理系统，包含以下核心功能：

1. 角色分级管理：系统预设普通用户、管理员、超级管理员三种角色，初始化包含超级管理员账号
2. 操作日志异步记录：通过消息队列（MQ）实现关键操作日志的持久化存储
3. 微服务间协作：用户服务与权限服务通过远程过程调用（RPC）实现数据交互
4. 分库分表实践：对用户表进行水平分片处理，提升大数据量下的存储与查询性能
5. 分布式事务保障：确保用户注册与角色绑定操作的原子性，避免数据不一致

## 二、系统架构设计

|  |
| --- |
| +-------------------+ +---------------------+ +----------------------+  | User Service |<--->| Permission Service |<--->| Logging Service |  | (HTTP API + MQ) | | (RPC服务端) | | (MQ消费者) |  +-------------------+ +---------------------+ +----------------------+ |

## 三、服务职责与实现细节

### 1. 用户服务（user-service）

**核心职责**：

* 处理用户注册 / 登录鉴权（基于 JWT 实现）
* 管理分库分表的用户数据存储与查询
* 通过 RPC 调用权限服务完成角色绑定
* 作为消息生产者，发送操作日志至 MQ

**关键接口设计**：

|  |  |  |  |
| --- | --- | --- | --- |
| 接口路径 | 请求方法 | 功能描述 | 技术实现要点 |
| /user/register | POST | 用户注册 | 分库分表写入用户数据 → RPC 调用绑定默认角色 → 发送日志消息至 MQ |
| /user/login | POST | 登录生成 JWT Token | 校验用户密码 → 生成并返回 JWT 认证令牌 |
| /users | GET | 分页用户列表 | 基于用户角色权限控制返回范围：普通用户仅自己，管理员可见所有普通用户，超管可见全部 |
| /user/{userId} | GET | 查询用户信息 | 权限控制逻辑同用户列表接口 |
| /user/{userId} | PUT | 修改用户信息 | 权限限制：普通用户仅能修改自己信息，管理员可修改普通用户，超管可修改所有 |
| /user/reset-password | POST | 密码重置 | 权限限制：普通用户重置自己密码，管理员可重置普通用户，超管可重置所有用户 |

**数据库设计**：

|  |
| --- |
| -- 用户表（水平分表设计）  CREATE TABLE users (  user\_id BIGINT PRIMARY KEY,  username VARCHAR(50),  password VARCHAR(255),  email VARCHAR(100),  phone VARCHAR(20),  gmt\_create TIMESTAMP  ); |

### 2. 权限服务（permission-service）

**核心职责**：

* 管理用户角色绑定关系（普通用户 / 管理员 / 超管）
* 提供 RPC 接口供外部服务查询用户角色信息
* 支持角色升级 / 降级操作

**RPC 接口定义**：

|  |
| --- |
| // 权限服务RPC接口  public interface PermissionService {  // 为新用户绑定默认角色（普通用户）  void bindDefaultRole(Long userId);  // 查询用户角色编码（返回role\_code）  String getUserRoleCode(Long userId);  // 超级管理员调用：升级用户为管理员角色  void upgradeToAdmin(Long userId);  // 超级管理员调用：降级用户为普通角色  void downgradeToUser(Long userId);  } |

**数据库设计**：

|  |
| --- |
| -- 角色表（权限服务单库存储）  CREATE TABLE roles (  role\_id INT PRIMARY KEY, -- 1:超级管理员 2:普通用户 3:管理员  role\_code VARCHAR(20) UNIQUE -- 角色编码：super\_admin/user/admin  );  -- 用户-角色关系表  CREATE TABLE user\_roles (  id BIGINT PRIMARY KEY AUTO\_INCREMENT,  user\_id BIGINT,  role\_id INT,  UNIQUE KEY uk\_user\_role (user\_id) -- 每个用户仅绑定一个角色  ); |

### 3. 日志服务（logging-service）

**核心职责**：

* 异步消费 MQ 中的日志消息
* 将操作日志持久化到数据库

**关键实现逻辑**：

|  |
| --- |
| // 日志消费处理示例  OperationLog log = new OperationLog();  log.setUserId(event.getUserId());  log.setAction(event.getAction());  log.setIp(event.getIp());  log.setDetail(event.getDetail()); // 记录操作详情（如修改字段）  logRepository.save(log); |

**数据库设计**：

|  |
| --- |
| -- 操作日志表（单库存储）  CREATE TABLE operation\_logs (  log\_id BIGINT PRIMARY KEY AUTO\_INCREMENT,  user\_id BIGINT,  action VARCHAR(50), -- 操作类型标识，如"update\_user"  ip VARCHAR(15),  detail TEXT -- 记录修改内容（如JSON格式的字段变更）  ); |

## 四、核心流程设计

### 1. 用户注册流程

|  |
| --- |
| 客户端 -> 用户服务: POST /user/register  用户服务 -> 分库分表: 写入users表  用户服务 -> RPC调用: permissionService.bindDefaultRole(userId)  用户服务 -> MQ: 发送"REGISTER"日志消息  日志服务 -> 消费消息: 写入operation\_logs表 |

### 2. 权限校验流程

|  |
| --- |
| 客户端 -> 用户服务: 请求需权限接口（如GET /user/123）  用户服务 -> RPC调用: permissionService.getUserRoleCode(userId=123)  权限服务 -> 查询user\_roles表 → 关联roles.role\_code  权限服务 -> 返回角色码（如"admin"）  用户服务 -> 本地逻辑校验角色权限：  - 若接口需管理员权限，检查role\_code是否为"admin"  - 若接口需超管权限，检查role\_code是否为"super\_admin" |

### 3. 分布式事务设计

**应用场景**：用户注册时需保证用户创建与角色绑定的原子性

**实现方案**：采用 Seata 框架的 AT 模式，通过全局事务协调器（TC）管理跨服务事务，确保两阶段提交的一致性

## 五、技术实现要求

### 核心技术点实现说明

|  |  |
| --- | --- |
| 技术点 | 实现要求 |
| 分库分表 | 对用户表进行水平分片，需提供分片规则与 SQL 验证方法 |
| RPC 调用 | 用户服务通过 RPC 调用权限服务接口，需实现服务发现与负载均衡 |
| 消息队列 | 用户服务发送日志消息至 MQ，日志服务消费后落库，需保证消息可靠性 |
| 分布式事务 | 实现用户注册与角色绑定的分布式事务，确保数据一致性 |

### 技术选型建议

|  |  |  |
| --- | --- | --- |
| 技术模块 | 推荐选型 | 选型理由 |
| 服务注册与发现 | Nacos | 支持服务注册、发现及动态配置管理，与 Spring Cloud Alibaba 深度集成 |
| RPC 通信 | OpenFeign + Nacos | Feign 提供声明式 REST 调用，集成 Nacos 实现服务发现与负载均衡 |
| 消息队列 | RocketMQ | 高吞吐、低延迟，支持事务消息机制，适合异步日志处理和分布式事务场景 |
| 分库分表 | ShardingSphere + MySQL | 支持水平分片、分布式主键生成（Snowflake 算法）及复杂查询路由 |
| 分布式事务 | Seata（AT 模式） | 对业务无侵入，通过 TC 和 RM 保障事务原子性，与微服务框架集成良好 |
| 配置中心 | Nacos | 提供动态配置管理，支持微服务配置的集中管理和实时更新 |

## 六、评分标准

|  |  |  |
| --- | --- | --- |
| 项目类别 | 具体要求 | 分值 |
| 功能实现（必做） | 用户注册 / 登录、角色权限控制、操作日志落库 | 50 |
| 业务日志打印（必做） | 核心业务流程日志打印，包含关键参数与执行结果 | 10 |
| 分库分表 | 用户表正确实现分片逻辑，提供分片规则说明与 SQL 验证案例 | 10 |
| RPC 调用 + 分布式事务 | 用户服务成功调用权限服务接口，实现跨服务事务一致性保障 | 10 |
| 消息队列 | 操作日志通过 MQ 异步处理，确保消息不丢失、不重复消费 | 10 |
| 文档 | 包含架构图、接口说明、技术难点总结、自测报告等完整文档 | 10 |
| 技术亮点 | 任何能体现技术能力的创新点（如性能优化、安全增强、可观测性设计等） | 20 |

## 七、提交要求

### 1. 代码结构规范

需提交三个独立项目：

* user-service：用户服务核心实现
* permission-service：权限服务核心实现
* logging-service：日志服务核心实现

每个项目需包含：

* 核心业务逻辑类（Service 层）
* 数据库实体类（Entity）
* RPC 接口定义（Feign Client 或 Dubbo 接口）
* 配置文件与启动类

### 2. 文档要求

* 架构图：清晰标注服务间调用关系与数据流向
* 技术实现说明：
  + 分库分表路由逻辑详细说明
  + RPC 调用链路设计与负载均衡策略
  + MQ 消息可靠性保障策略（如重试机制、幂等性处理）
  + 完整接口文档（包含请求 / 响应格式、权限控制说明）
  + 自测方案与测试用例

### 3. 提交格式规范

|  |
| --- |
| 姓名\_学校\_后端/  ├── 后端代码.zip  │ ├── user-service/ # 用户服务完整项目目录  │ ├── permission-service/ # 权限服务完整项目目录  │ └── logging-service/ # 日志服务完整项目目录  └── 文档/ # 所有相关文档文件  ├── 架构设计图.png  ├── 技术实现说明.md  ├── 接口文档.md  └── 自测报告.md |

**注意事项**：

* 压缩包命名格式：姓名\_学校\_后端.zip（示例：张三\_XX 大学\_后端.zip）
* 严禁抄袭或共享代码，需体现个人开发痕迹（如代码提交记录）
* 允许借助 AI 工具辅助开发，但需在文档 / 代码中标注 AI 生成内容，未标注将视为作弊

（注：文档部分内容可能由 AI 生成）