人人微服务平台-系统架构文档

系统名称	人人微服务平台
项目负责人	Mark
作者	Mark
最后更新日期	2020-06-03

人人开源(https://www.renren.io)

目 录

0 版权说明

1 背景

在传统的 IT 行业软件大多都是各种独立系统的堆砌，这些系统的问题总结来说就是扩展性差，可靠性不高，维护成本高。如 OA、CRM、ERP 等大型应用，随着新需求的不断增加，所有模块全部耦合在一块，代码量大，企业维护大型整体式应用，变得越来越困难。

微服务是一个新兴的软件架构，就是把一个大型的单个应用程序和服务拆分为数十个微小的应用，每个应用都相当于一个单独的项目，且代码量明显减少，遇到问题也相对来说比较好解决，可以使用不同的开发技术，开发模式更灵活，同时也可以使用不同的存储方式（比如 Oracle、MySQL、Mongo 等）。这种架构使得每个服务都可以有专门开发团队来开发，开发者可以自由选择开发技术。

2 名词解释

微服务：是一种架构风格，一个大型复杂软件应用由一个或多个微服务组成。系统中的每个微服务可被独立部署，每个微服务之间是松耦合的。每个微服务仅关注于完成一件任务并很好地完成该任务。在所有情况下，每个任务代表着一个小的业务功能。

Spring Cloud：是一系列框架的有序集合。它利用 Spring Boot 的开发便利性巧妙地简化了分布式系统基础设施的开发，如服务发现注册、配置中心、消息总线、负载均衡、断路器、数据监控等，都可以用 Spring Boot 的开发方式做到一键启动和部署。

3 设计目标

3.1 实现功能

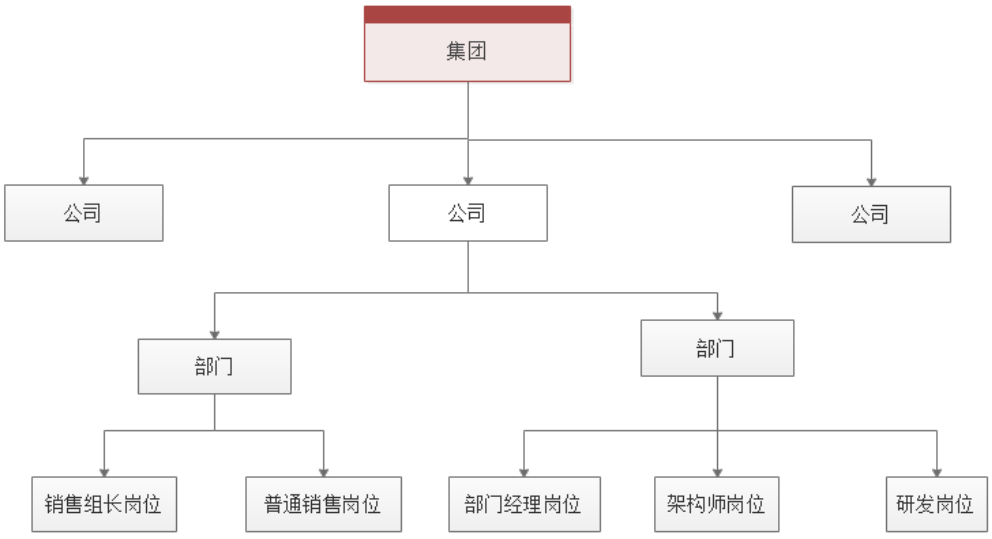
人人微服务平台，主要实现用户管理、部门管理、角色管理、功能权限（菜单、按钮权）、数据权限、日志管理、菜单管理、参数管理、数据字典、国际化等核心功能。

3.2 用户管理

用户管理，一般是管理企业在职人员，管理员可以新增、编辑、查询、删除、授权等操作，在职人员也可以登录系统，根据赋予的权限，进行相应的操作。

3.3 部门管理

部门管理，负责管理每个部门、岗位等，管理员可以对部门，进行新增、编辑、删除等操作，组织架构图如下：



3.4 角色管理

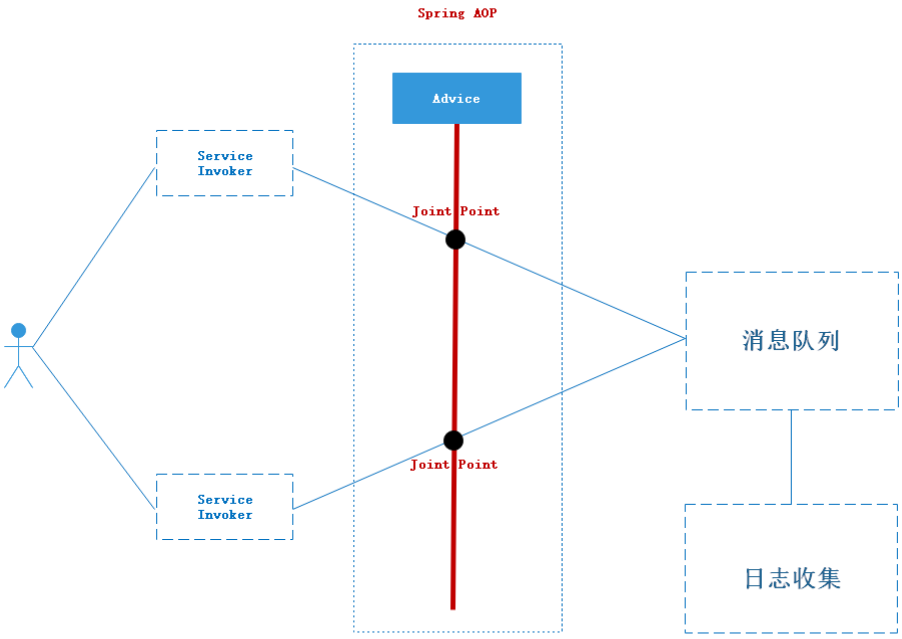
角色管理，主要用于权限的统一管理，人员与角色的关系是一对多的，每个人可以拥有多个角色。管理员可以对角色进行新增、编辑、删除等操作。

功能权限，就是有没有做某种操作的权限，具体表现形式就是，你看不到某个菜单或按钮。主要针对菜单、按钮进行权限管理。当新增、编辑角色时，当前角色可以勾选对应的菜单及按钮。赋予该角色权限的用户，就拥有该菜单及按钮的权限，其他菜单及按钮则没有权限操作。

数据权限，就是有没有对某些数据的访问权限，具体表现形式，当某用户有操作权限的时候，但不代表其对所有的数据，都有查看权限。

3.5 日志管理

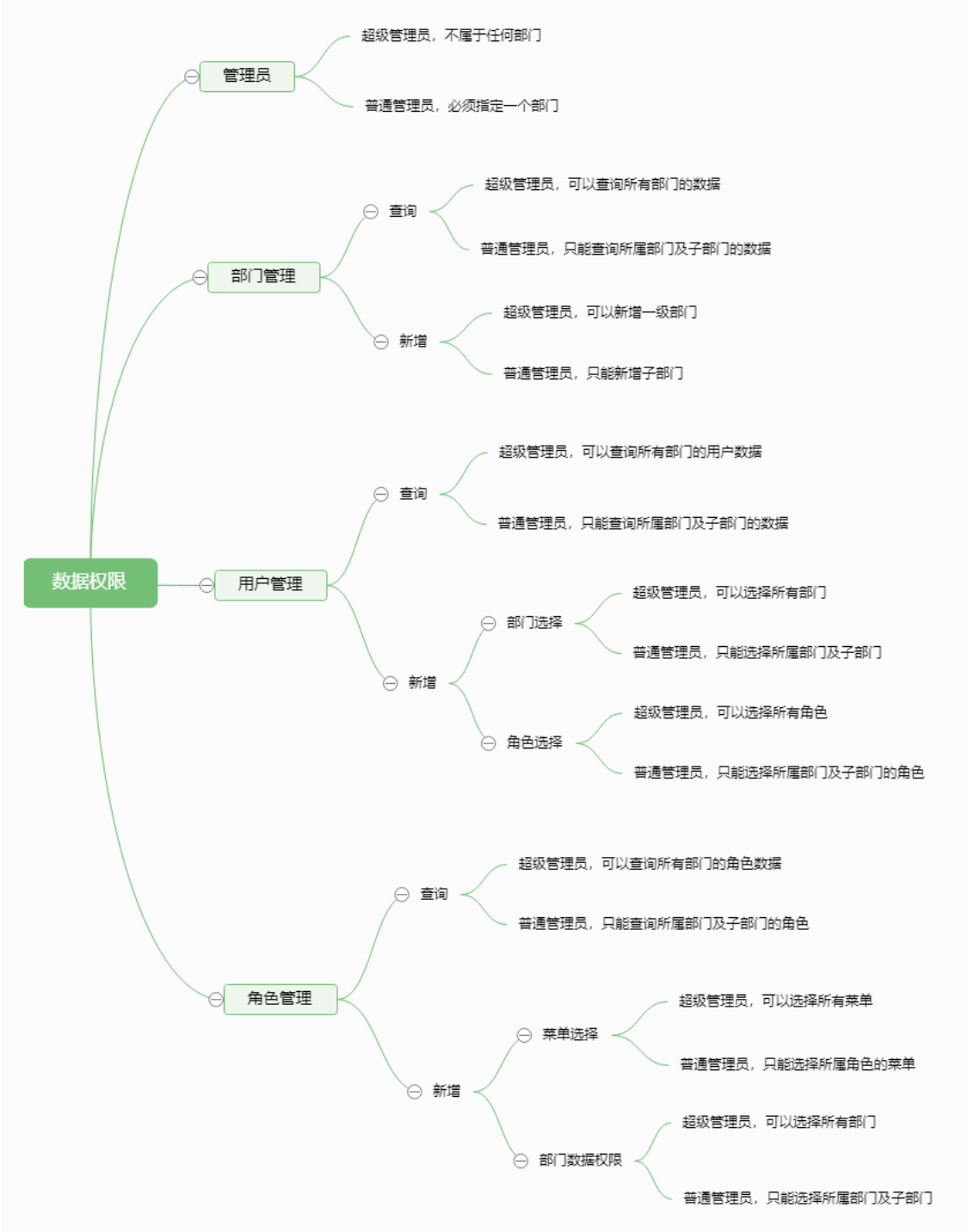
日志管理，主要保存登录、操作、异常等日志记录，方便管理人员追踪问题，可以查询、导出日志；系统采用 Spring AOP、Redis 消息队列，实现系统日志的收集，如下图所示：



3.6 权限控制

系统分为超级管理员和普通管理员，超级管理员拥有所有功能权限及数据权限，主要职责是创建普通管理员账号，并给普通管理员分配权限。在系统里，拥有用户管理、角色管理的账号，我们就可以理解成普通管理员，因为可以新建用户及分配权限。

管理员新建用户、角色、部门时，会有一定的约束，保证不会有越权操作，约束如下所示：



4 系统设计

4.1 技术选型

采用 Spring Cloud 进行开发，Spring Cloud 是一个基于 Spring Boot 实现的微服务开发平台，它提供了配置管理、服务发现、断路器、智能路由、微代理、控制总线、全局锁、决策竞选、分布式会话和集群状态管理等操作，为我们提供了一种简单的微服务开发模式。

4.2 前后端分离

在传统的网站开发中，前端一般扮演的只是切图的工作，简单地将效果图转变成 HTML 页面的过程。而与后端的数据交互工作，则可以由前端完成，也可以由后端完成，造成分工就不明确，前后端相互推卸责任等，且前后端代码还紧紧的耦合在一起，不仅仅开发效率低，而且代码难以维护，定位 BUG 困难，分工不均等缺点。

前后端分离后，则可以很好的解决前后端分工不均的问题，将交互逻辑分配给前端处理，而后端则可以专注于其本职工作，比如提供 API 接口，进行权限控制以及进行运算工作。而前端开发人员则可以利用 Node.js 来搭建本地服务器，直接在本地开发，然后通过 axios 将 API 请求转发到后端，这样就可以与后端交互，并且与后端完全解耦。前端可以根据后端提供的 API 文档，开发前端交互逻辑。这样，前后端就可以同时开工，且互不依赖，开发效率更高，更加专注。

4.3 Restful 风格

接口按照 Restful 分格设计，如下所示：

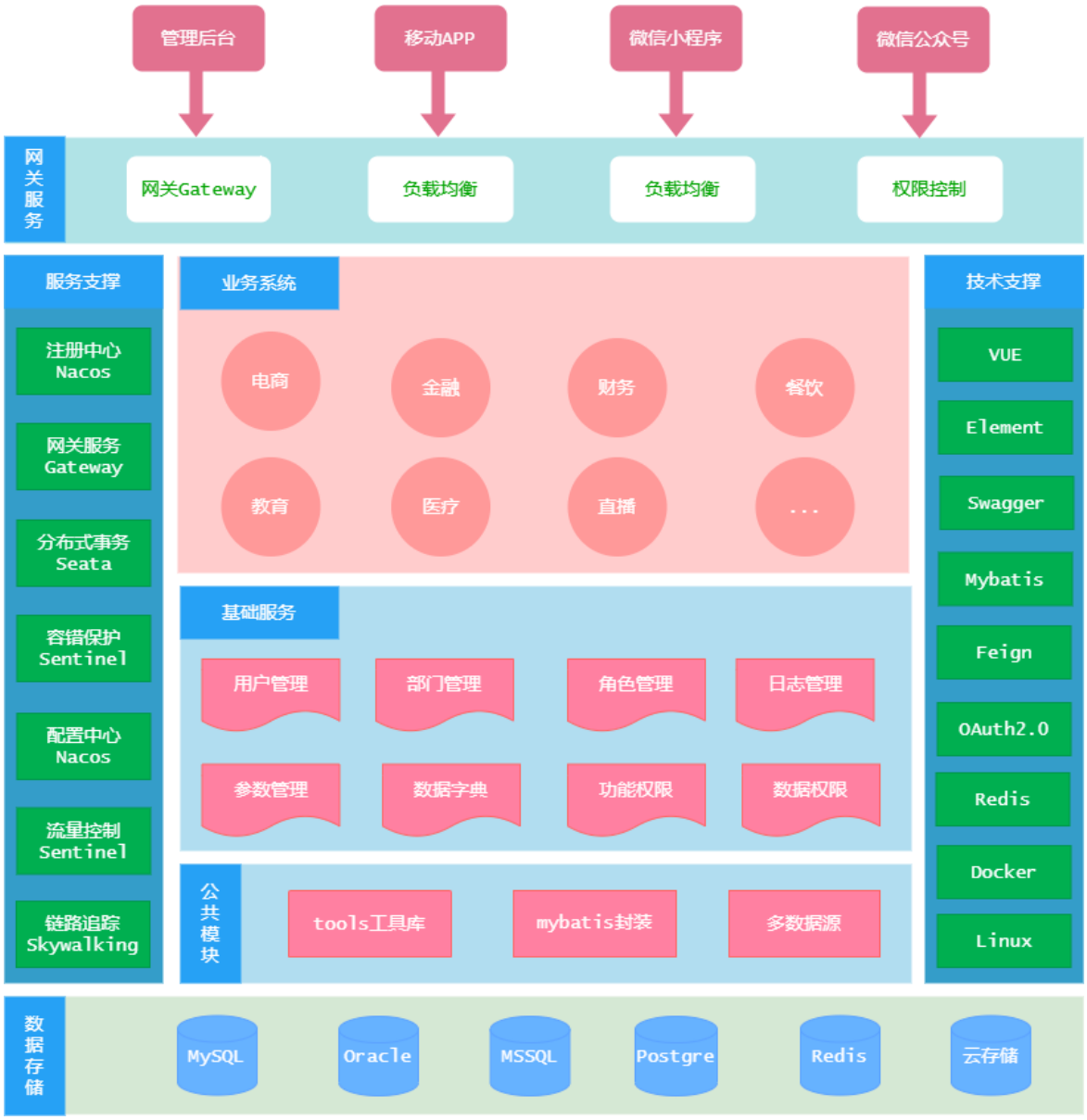
资源名称	资源地址	请求方式
获取用户	/sys/user/{id}	GET
新增用户	/sys/user	POST
修改用户	/sys/user	PUT
删除用户	/sys/user/{id}	DELETE

4.4 错误码定义

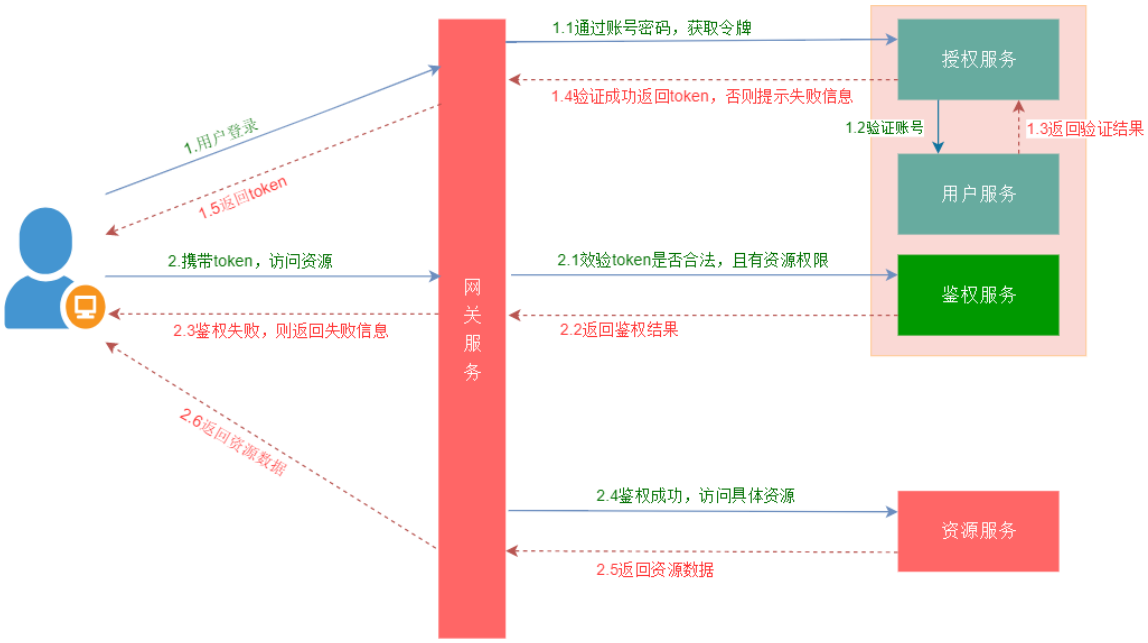
调用接口数据时，无论后端是否发生异常，HTTP 状态码，都只会返回 200，具体的错误，需在返回的数据里描述清楚，如调用接口，没有权限，则返回：

{"msg":"没有权限访问","code":401}

4.5 系统架构图



4.6 系统认证流程图及说明

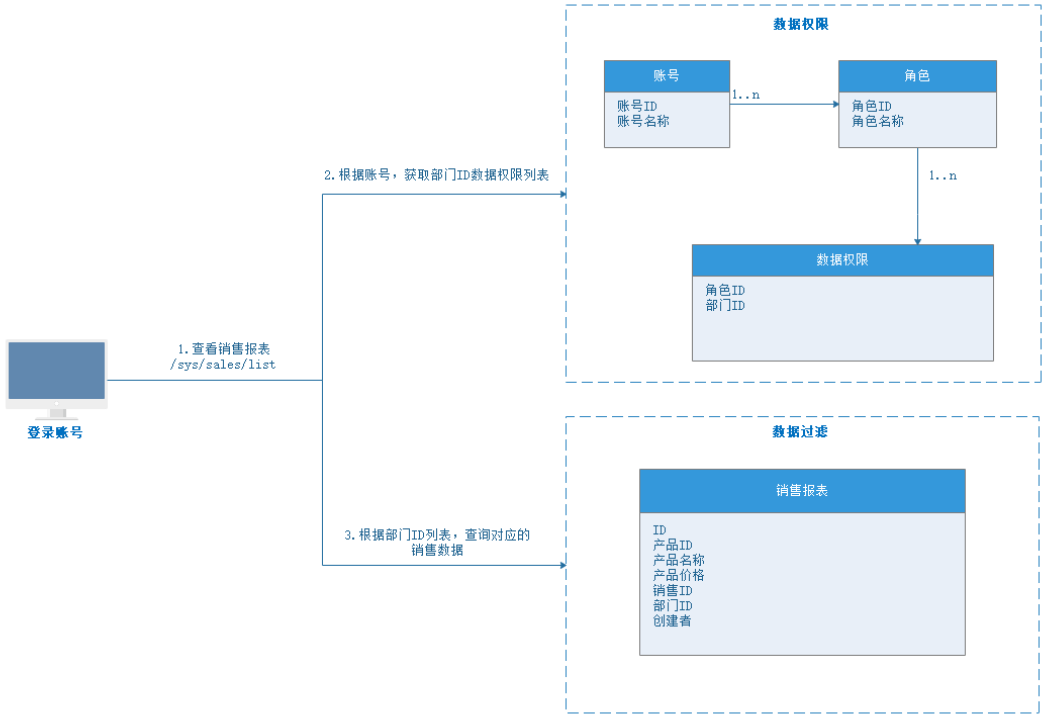


客户端想访问资源接口时，都需要经过鉴权，才能正常访问，没有经过鉴权的请求，都会被拒绝。本系统采用 SpringSecurity OAuth2.0，对客户端进行授权及鉴权，步骤如下：

1、 客户端携带账号、密码，请求授权服务，授权服务验证账号、密码是否正确，如果正确返回 token 令牌，否则告知客户端账号或密码不正确等

2、 客户端携带 token 令牌请求资源接口，网关会请求鉴权服务，判断用户是否有权限访问该资源，如果有权限，则返回资源的响应数据，否则告知客户端无权访问该资源

4.7 数据权限架构图及说明



数据权限，主要进行数据的过滤，没相应的数据权限，则不能查询、操作相应的数据。

数据权限实现方式如下：

1. 用户登录后，获取部门 ID 数据权限列表
2. 根据部门 ID 列表进行数据过滤，数据过滤表，都需要有部门 ID 字段