



用户表 用户ID 姓名 部门 状态 创建时间 更新时间

元数据表 元数据ID 元数据信息 元数据描述 父ID 子类型 状态 创建时间 更新时间 1. 元数据分为两类:前端元素和后 台URL,通过元数据类型区分 2. 元数据有层级关系,在赋权限 时,拥有父亲的权限后,自动拥有 所有孩子的权限 3. 前端负责前端元素的显示与否 , 同时发起后台请求也需要查看是否 有此URL的请求。 4. api-gateway负责校验URL是否 在权限内 5. 对于URL的元数据,可通过子类 型来标识是 GET/POST/PUT/DELETE请求

角色表 角色ID 角色名

父角色ID tag 状态 角色描述 创建时间 更新时间

如超级管理员

1. 父角色ID用来标识角色的上下级 2. 可通过tag , 来标识特殊角色 , 比

角色元数据映射表

角色ID 元数据ID 状态 创建时间 更新时间

1. 子角色拥有的元数据, 父角色自 2. 拥有父元数据的角色,自动拥有 子元数据的权限。

用户角色映射表

角色ID 用户ID 状态 创建时间 更新时间

操作日志表

用户ID 角色ID 元数据信息 元数据描述 请求信息 应答信息 创建时间 更新时间

蛋和鸡:管理权限的权限

对于超级管理员的角色 , 前端和后台除了登 陆校验和每次access-token外,其他不做 任何权限校验。

对于管理权限的页面,至少需要以下功 🔼 能:

1. 展示可授予权限的用户列表

2. 展示可授予角色的角色列表 3. 展示可授予角色的元数据列表

4. 授予用户某某角色

另外,可考虑和普通权限进行 分离的方案。

数据一致性 高并发

对于用户权限发生动态变化的情况,简单宝 现的话:只有用户重新登陆才生效。 严格一致性的话:每次都去权限服务查询, 如果在权限之外, 把最新的权限列表返回前 端。

前端根据用户账号缓存权限列表,登陆时,将校 验和或者获取权限的时间一并带到apigateway, api-gateway校验是否发生变化,如 果没有变化,不用返回权限列表。

api-gateway启动时,从权限服务拉取所有信息 缓存到本地/Redis,每次获取权限或者校验权限 时,不用再去权限服务请求。

api-gateway定时去权限服务拉取最新权限信息,更新缓存信息。

权限信息发生变更时,由权限服务主动通知apigatway,