权限设计

权限系统是独立系统，负责所有的系统的功能权限控制，其他业务系统比如商品模块，订单模块，用户中心，每个系统用户都有自己的角色和权限，那么权限系统就可以配置其他系统的角色和权限。该系统的主要目的是对不同的用户访问资源进行权限的控制，避免因权限控制缺失或操作不当引发的风险问题。

# 涉及的模块概念

## 功能权限（需要分类）

功能权限是把应用系统中的资源按最小量化划分，表现为菜单的访问（页面级）、功能模块的操作（功能级）、文件上传的删改，甚至页面上某个按钮、图片是否可见等等都属于权限的范畴。可以根据业务的需要自定义系统中的权限。比如：

1. 菜单

用户登录后可以看到的菜单，可以根据需求定义到几级菜单。

1. 页面

如果想看到页面，用户仅仅只有菜单权限还不够，还需要有对应的页面权限。

1. 按钮

用户是否能看到页面上的功能按钮，包括比如查看，新增，修改，删除，审核等。

1. API

用户点击按钮时，即向后台发送请求，后台会校验用户角色下的所有权限是否包含该删除权限，如果是，就可以进行下一步操作，反之提示无权限

注：用户如果想要看到某个页面，必须同时具备菜单和页面两个功能权限；如果想要完成某个按钮功能，必须同时具备按钮和 API 功能权限。这样做的好处：

1. 配置灵活，如果用户没有按钮功能权限，但是有 API 的功能权限，那么用户可以不通过页面的按钮来完成请求，可以完美适应用户通过各种形式来完成请求（不局限于通过页面访问）。
2. 更加安全，可以避免用户直接通过 API 访问后台接口

## 功能集

对于某一个具体的应用系统，要完成一个功能所需要的所有功能权限的集合。

一个新增商品功能集

比如，对于一个网站应用而言，一个“新增商品（功能集）”需要包含以下功能

1. 菜单：需要进入到商品列表的菜单
2. 页面：列表页页面
3. 按钮：需要列表页的新增按钮
4. 页面：新增页面
5. 按钮：新增页面的保存按钮
6. API：保存按钮的 API （有可能多个）

## 角色

起到了桥梁的作用，连接了用户和功能集的关系，每个角色可以关联多个功能集，同时一个用户可以拥有多个角色，那么这个用户就有了多个角色的多个功能集。由于用户数量比较大，其中很多人的权限都是一样的，可以把拥有相同功能权限的人归为一类。这就引入了"角色"概念，也就是说，拥有相同功能集的人被视为是相同角色的用户。

一个角色也可以被多个用户拥有，管理员只需要把该角色赋予用户，那么用户就有了该角色下的所有功能集，这样设计既提升了效率，也有很大的拓展性。

## 用户组

和角色的作用类似，可以分配给用户哪些功能集。首先给用户组分配功能集或角色，然后把有相同功能集的用户添加进来，添加进来的用户自动获取用户组配置的功能集。

和角色相比，用户组更具临时性（给用户临时扩展某些功能集），灵活性，扩展性，使管理员对用户--功能集的操作更加便捷。

## 用户

系统中的用户：

1，OM 用户（系统管理员），负责功能权限，功能集，角色，用户组，用户的管理和维护。这类用户可以考虑“组织/职位”权限模式

2，B用户（系统使用者），他们可以拥有系统中角色和申请加入用户组（一个用户可以拥有多个角色，多个用户组）

3，开发人员，系统中还需要开发人员对功能集进行维护，因为功能集中的功能权限只有开发人员最清楚

用户同时拥有多个角色和用户组，用户组同时拥有多个角色和功能集。系统用户最终拥有的功能权限 = 用户角色拥有的权限 + 该用户所在用户组拥有的权限

# 模块功能

## 功能权限

### 分类管理

维护系统涉及的所有功能权限类型，根据实际业务需要把系统中所有资源、功能进行分类，比如：菜单、页面、页面上的按钮，文件等都可以作为分类

新增：

实现一：可以写死在后台，无需管理员维护。系统中的分类是固定的，可以为每个分类维护一张表，如果要新增分类，直接在数据库新增一个对应分类表。

实现二：nosql，采用所有的权限在一张表中

### 功能管理

新增：

1. 选择分类，不同的分类对应功能权限属性不一样，也就是页面表单内容不一样。比如，菜单分类具有“父菜单”属性，API 分类有“请求路径”属性。
2. 选择和其他分类下的功能的层级关系，目的是在新增功能集选择功能时，方便功能的级联显示

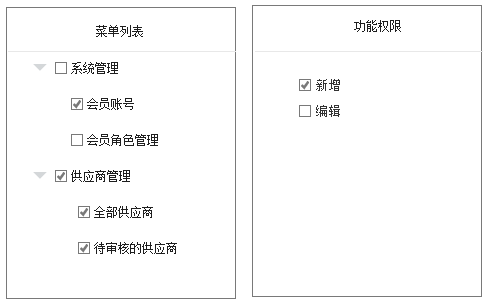
删除：

修改：修改基本信息，不能修改分类

查询：

## 功能集

需要根据应用系统的需求去维护功能集，需要维护人员（后台开发者）对系统中的功能和业务非常理解。比如：应用系统中给角色配置权限是如下设计：



为满足以上需求，功能集维护成这样：

“会员账号”功能集

“新增”功能集

当用户选择“会员账号”“新增”时，就拥有这两个功能集的所有权限了

新增：勾选不同功能权限分类下的功能权限，选择多个功能。

删除：

修改：修改功能集的基本信息，也可以修改功能包含的功能权限

查询：查询功能集下有哪些功能

## 角色

新增：选择多个功能集。

删除：如果角色被使用（已分配给用户）则不能删除

修改功能集：对已具有的功能集取消勾选，或为某功能集添加勾选，来修改角色的功能集，点击“保存”按钮保存修改信息。

给角色添加用户：点击某个角色，选择多个用户，实现给批量用户授予角色的目的

查询某个角色拥有哪些功能集

## 用户组

添加成员：选择用户，添加。

分配功能集：列出系统中的所有功能，选择添加。

分配角色：

编辑：只能修改组的基本信息，比如，组名称，描述。

删除：删除一个用户组，同时删除组内用户，和该用户组的功能权限

新增：新增一个用户组。

查询用户功能集

## 用户

配置角色：列出系统中的所有角色，勾选保存。

申请加入用户组：需要管理员审批

查询用户的功能权限

# 系统管理员系统维护流程

1. 新增功能（权限）
2. 新增功能集
3. 新增角色
4. 新增用户组
5. 给用户配置角色，用户组

# 授权流程

即给用户分配角色

## 手动授权

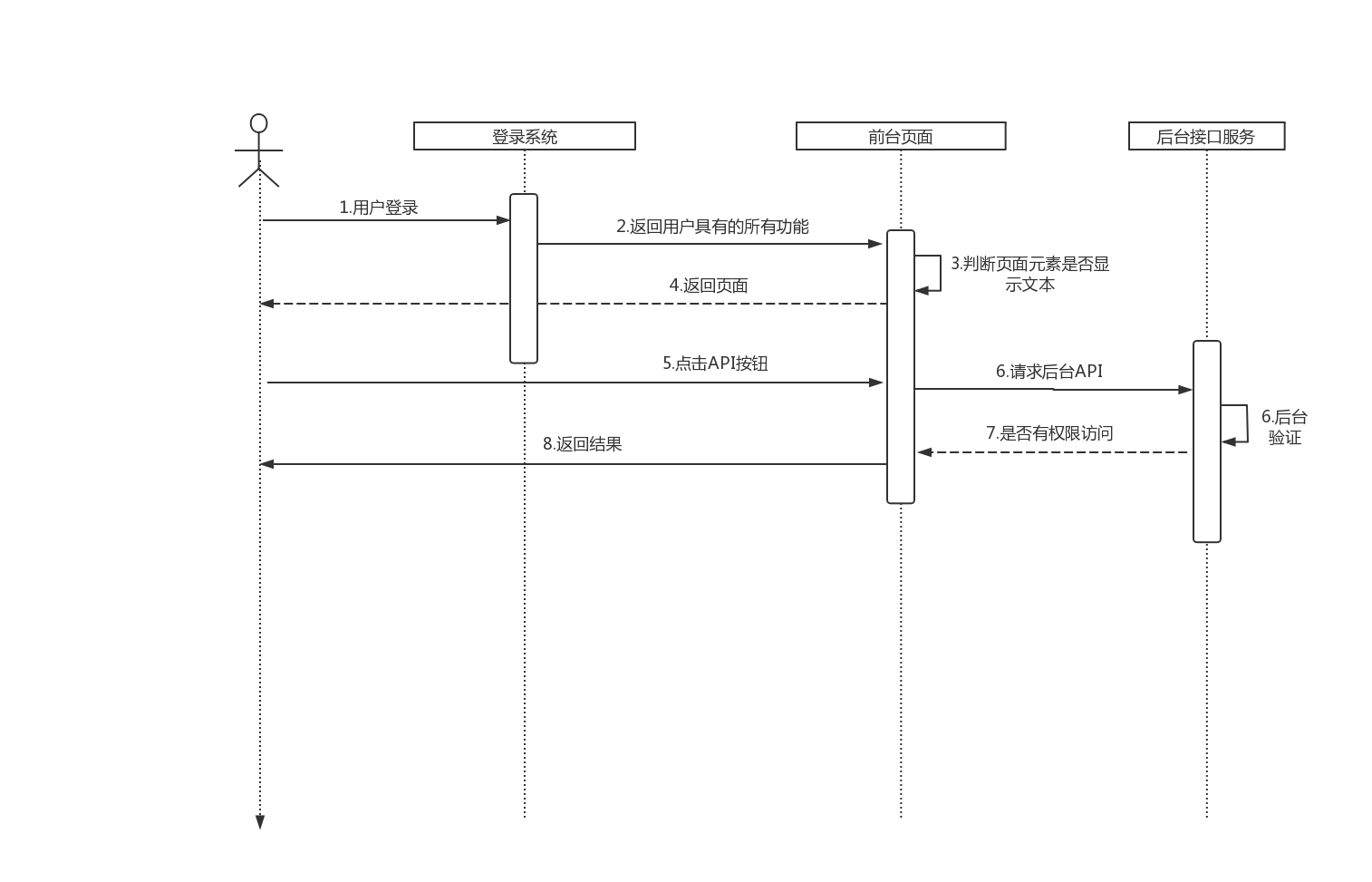
企业后台可以在用户管理中分配角色，也可以在角色管理中分配用户

## 审批授权

B用户申请某个角色，OM 后台审核

# 用户权限验证流程

1. 用户登录，登录接口返回该用户所有功能
2. 前台判断用户是否具备“菜单”，“页面”，“按钮”等功能，有则显示，没有不显示
3. 请求后台 API 时，带着登录时返回的“API”功能
4. 后台验证用户具有这个 API 功能



# 数据权限

规定用户可以看到哪数据，比如，我在系统中创建了一条订单，那么是否只有我自己可以看到？是否所有人都能看到？还是指定某些人可以看到？

怎么指定“某些人”？

前提条件（系统应具备的功能）：

1，用户所在企业必须有组织层级架构

目的是根据组织层级关系，找到和用户相关联的其他用户，比如，我创建的订单只有我和我的上级能看到

2，规定“某一类”数据的访问权限

常规情况下，应该为系统中的每一条数据都设置访问权限，比如，“本人”“本部门”，大多数系统中，只需为数据类别设置访问权限，比如，订单类数据，商品类数据，计划类数据。

作用是，在整体企业组织层级架构中，寻找某一段层级的用户

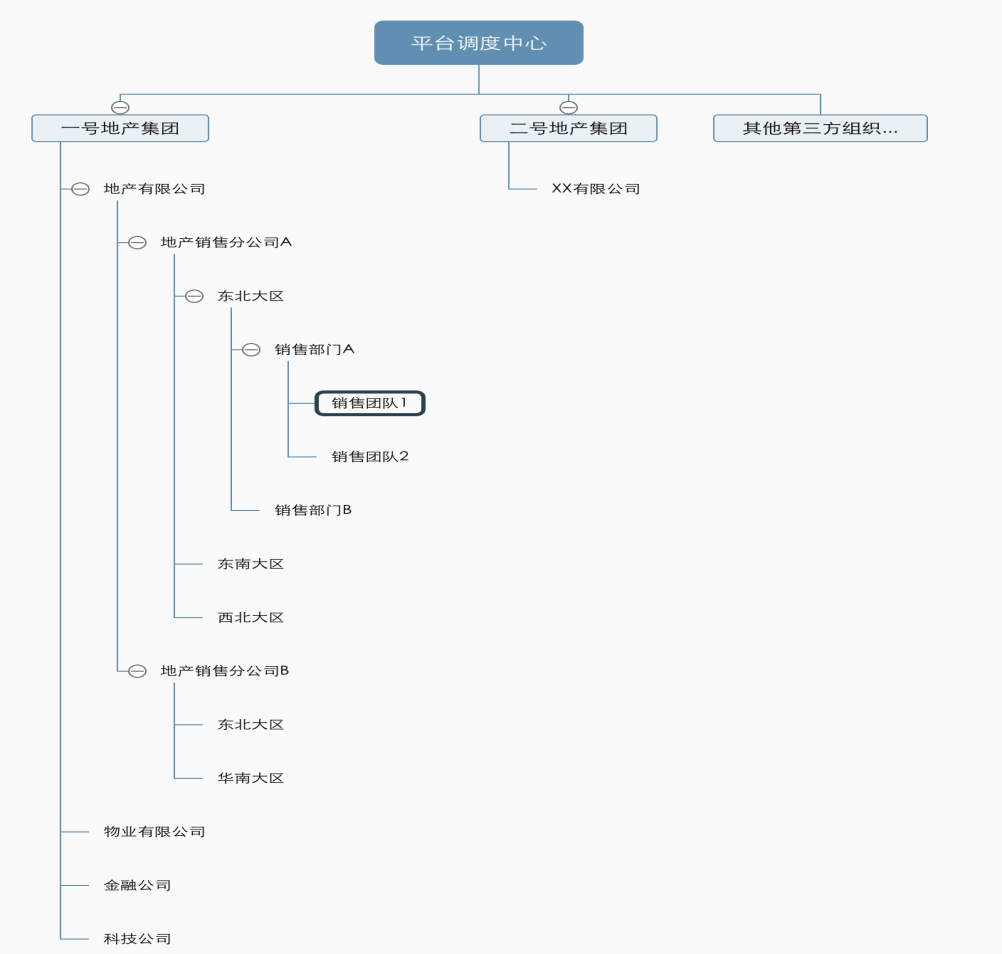
3，在创建数据时要有和该数据相关的用户信息

比如，一条订单中，要有买卖双方的用户信息。作用是，作为寻找“指定某些人”的根源

三个条件共同决定“某些人”

## 维护企业组织层级架构

这是实现数据权限的前提，类似下图结构：



## 给数据配置数据权限

### 前置配置

### 后置配置

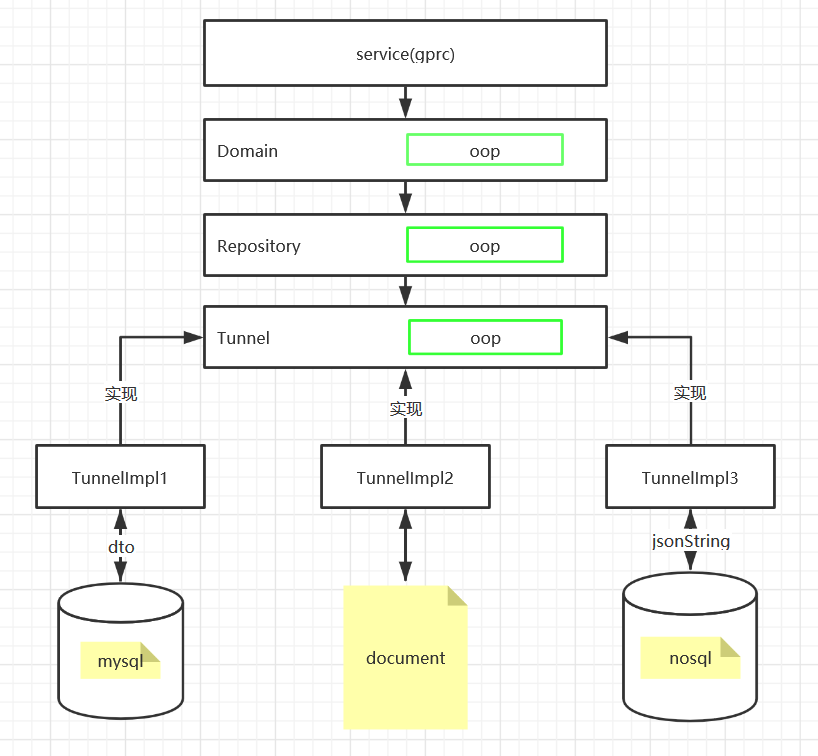
后台维护，

维护数据分类，把系统中涉及到数据进行分类，比如，订单数据类，商品数据类，计划数据类，可以理解成系统中的每个列表就是一个分类

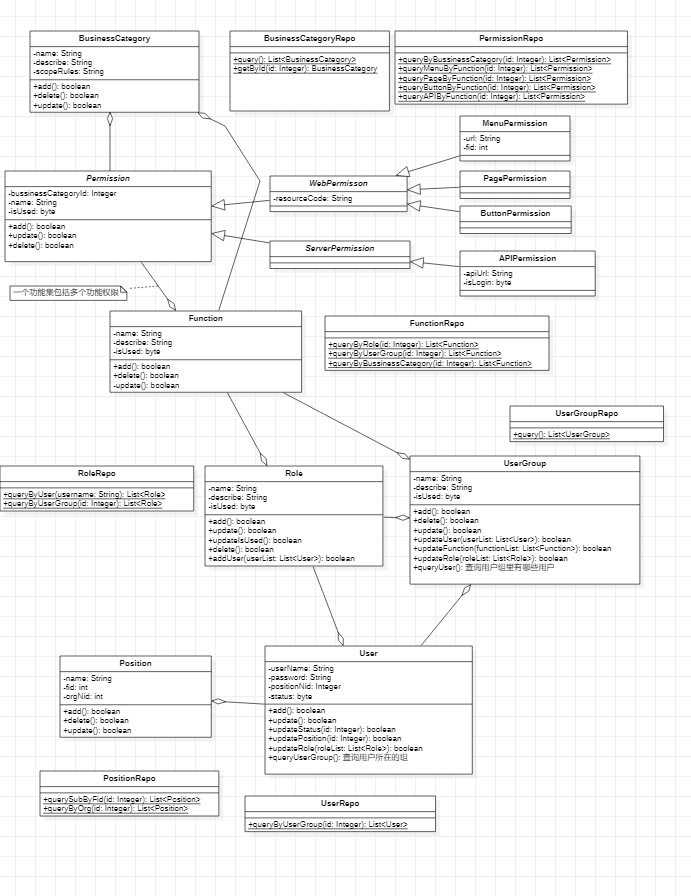
维护分类的访问权限，比如，商品类的数据的访问权限是“本人”，“上级”，“组内”“同职级”等。如是设置为“本人”，则只有发布商品本人可以看到；如果选择“上级”，则发布商品本人的上级可以看到

# 程序设计

基于 DDD（领域驱动模型）程序设计思想实现功能，基于 rpc 协议的框架，对外提供服务接口。整体架构如下图：



## 领域模型UML



## 数据库设计

