1. Web
2. Mysql
3. Mybatis Plus
4. OAuth2 + JWT
5. Redis
6. Ngixs
7. RabbitMq
8. Swagger Knife4j

用户管理 ：用户列表、权限列表、角色列表

菜单管理

权限认证

负载均衡

消息队列

## 数据库设计规范

### 表设计规范

1、表名全部小写，单词间通过'\_'间隔

2、主键命名为'id'，类型为serial自增长主键，会默认创建名为[表名\_id\_seq]的序列，约定初始值(start)为1000000

3、必须包含4个审计字段且不能为空。created\_time、updated\_time、created\_by、updated\_by。

4、关键词要求大写，使用IDE如idea进行格式化

5、常量枚举全部用大写

### 外键及索引命名规范

1、唯一索引：ux\_表名\_索引字段。如：ux\_resource\_code

2、普通索引：ix\_表名\_索引字段。如：ix\_role\_name

3、外键命名：fk\_表名\_字段名。如：fk\_orders\_product\_id

### 字段长度规则

| 名称类 | 类型 | 长度 | 备注 |

|--------|---------|------|--------|

| 编码类 | varchar | 100 | |

| 账号类 | varchar | 100 | 如email，username |

| 状态类 | varchar | 5 | 如订单状态等 |

| 名称类 | varchar | 200 | 中文名称，如产品名 |

| 手机电话| varchar | 20 | |

| 描述简介| varchar | 500 | |

| 网址类 | varchar | 500 | 如url |

| 时间类 | timestamp | | |

## URL和方法命名规范

### RESTFUL URL命名规范

API URI design

API URI 设计最重要的一个原则： nouns (not verbs!) ，名词（而不是动词）。

CRUD 简单 URI：

| 方法 | URL | 功能 |

|--------|-----------|------------------|

| GET | /users | 获取用户列表 |

| GET | /users/1 | 获取 id 为 1 的用户|

| POST | /users | 创建一个用户 |

| PUT | /users/1 | 替换 id 为 1 的用户|

| PATCH | /users/1 | 修改 id 为 1 的用户|

| DELETE | /users/1 | 删除 id 为 1 的用户|

上面是对某一种资源进行操作的 URI，那如果是有关联的资源，或者称为级联的资源，该如何设计 URI 呢？比如某一用户下的产品：

| 方法 | URL | 功能 |

|--------|---------------------|------------------------------------|

| GET | /users/1/products | 获取 Id 为 1 用户下的产品列表 |

| GET | /users/1/products/2 | 获取 Id 为 1 用户下 Id 为 2 的产品 |

| POST | /users/1/products | 在 Id 为 1 用户下，创建一个产品 |

| PUT | /users/1/products/2 | 在 Id 为 1 用户下，替换 Id 为 2 的产品|

| PATCH | /users/1/products/2 | 修改 Id 为 1 的用户下 Id 为 2 的产品 |

| DELETE | /users/1/products/2 | 删除 Id 为 1 的用户下 Id 为 2 的产品 |

### 方法命名规范

### Mapper

insert/add

delete

update

query

search

### Service

add

get

delete

update

save

query

search

### Rest

add

get

delete

update

save

query

search