四川大學
SICHUAN UNIVERSITY

危房改造管理系统项目介绍

汇报人：tian

时间：2024届招聘季

目录
CONTENTS

1

PART
ONE

01

背景

传统危房改造管理

四川大学 SICHUAN UNIVERSITY

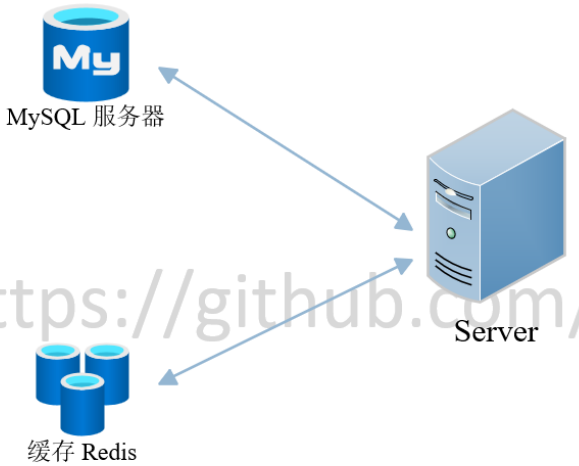
农户申请	政府审批	政府公示	建设跟踪	改造完成
提出申请	**多层审批**	**多层公示**	**多阶段跟踪**	**档案导出**
. 身份信息 . 低收入群体证明 . 房屋信息 . 建房计划	. 村集体评议 . 镇审核、危房改造认定 . 县审核 . 民政局审核 . 住建局审批	. 村公示 . 镇公示 . 县公示 . 民政局公示 . 住建局审批	. 地基、主体、封顶、验收照片 . 质量安全巡查记录 . 竣工验收监督检查表 . 事件通报	档案管理

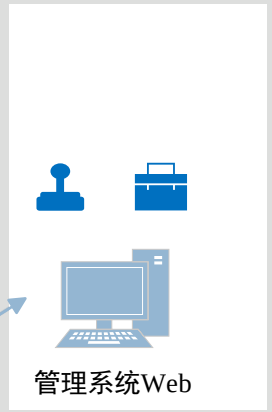
缺点：

- 流程复杂：农户进行危房审批时，需要跑多个政府部门进行审批，耗时且在审批流程中可能遗漏程序
- 文档管理困难：申请、审批、公示、跟踪流程中产生的记录较多，整理档案困难

解决方案：利用微服务小微快的特点，建设危房改造管理系统

危房改造管理系统





使用用户：

村、镇街、局县、民政局、住建局干部

主要功能：

1. 系统数据看板；
2. 进行系统用户管理，干部注册；
3. 发布新政策、违规警告；
4. 对农户的危房改造申请进行审核驳回；
5. 完工的危房改造建设项目生成 word 档案，并在线打印

使用用户：

村、镇街、局县、民政局、住建局干部;农户

主要功能：

农户： 1. 农户注册；2. 提出危房改造申请；
3. 改造申请通过后，建房过程中，提交房屋图片、建房计划、责任书等材料。

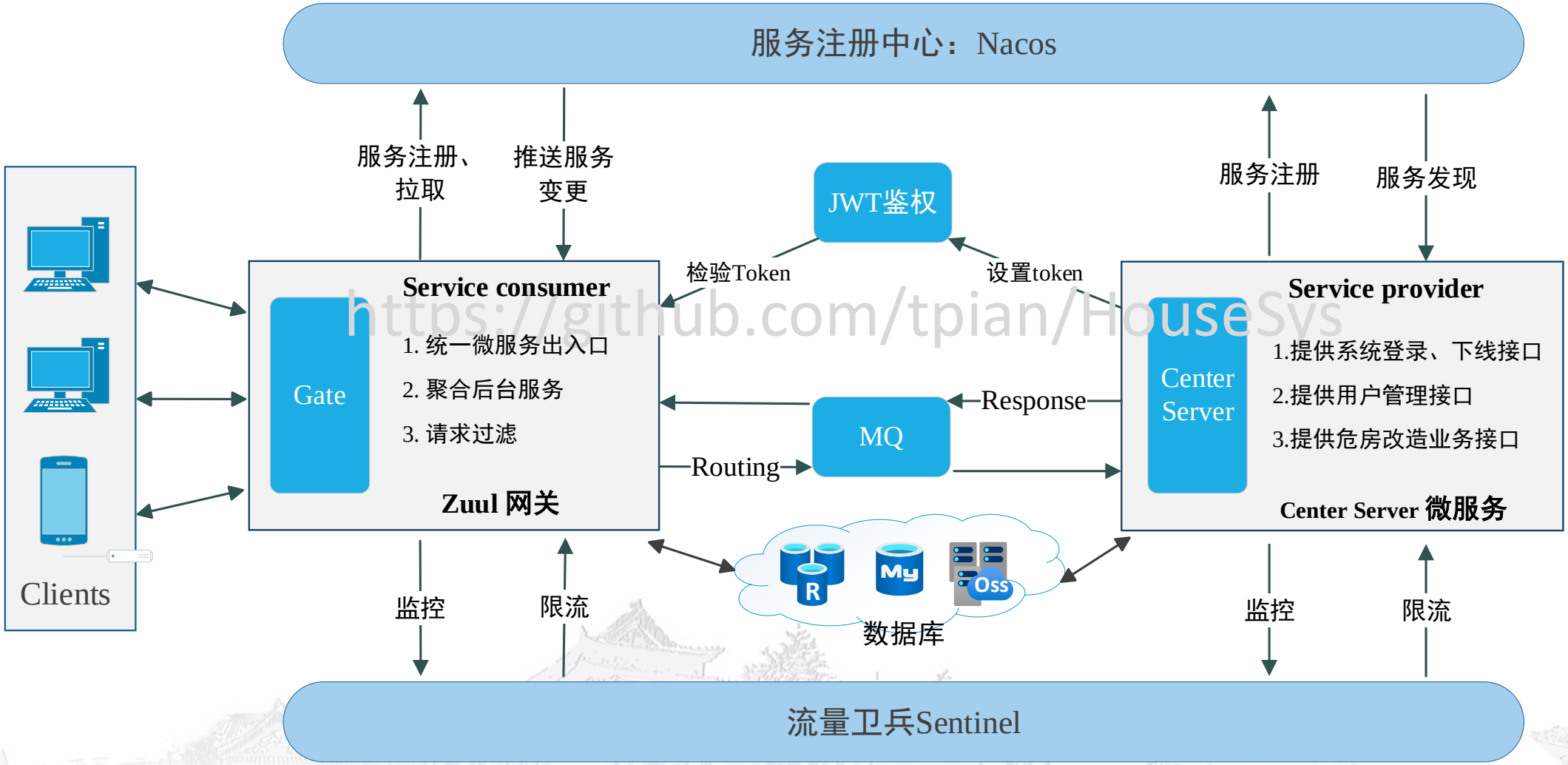
干部： 审核/驳回申请

2

PART
TWO

项目架构

网络架构



3

PART
Three

具体实现

环境配置

小程序：

uni-app，Vue2

Web管理系统前端：

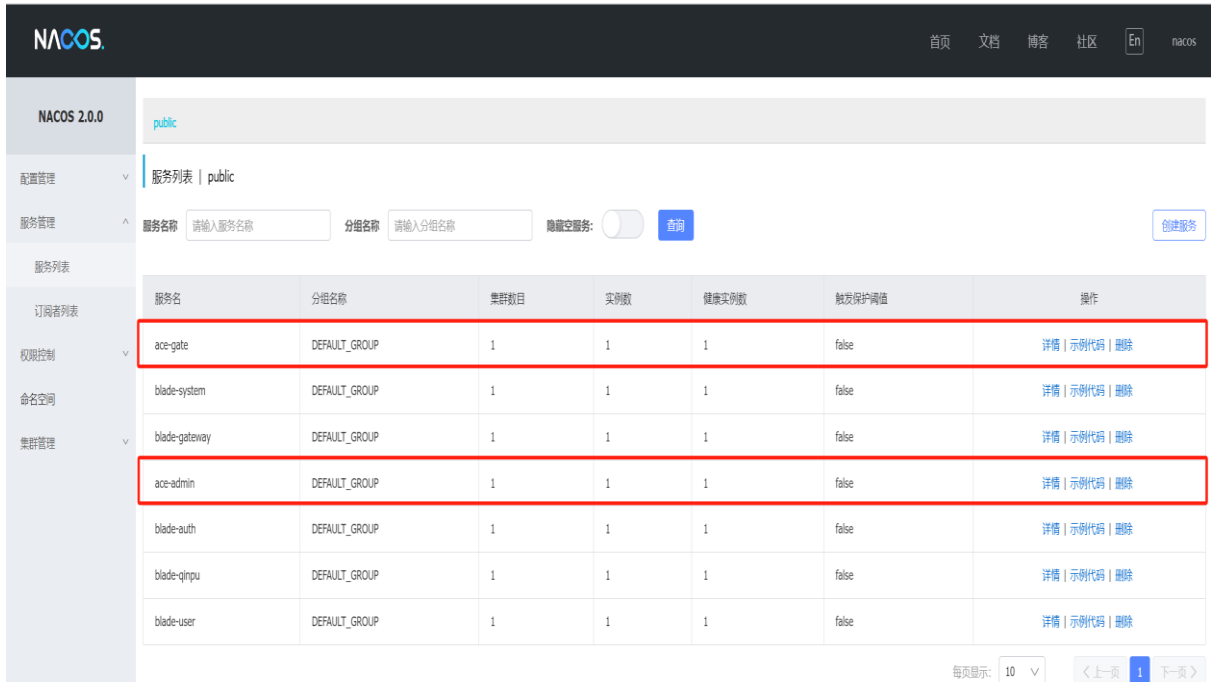
Vue2，Element-ui组件库，Node 9.4.0，Npm 6.10.2

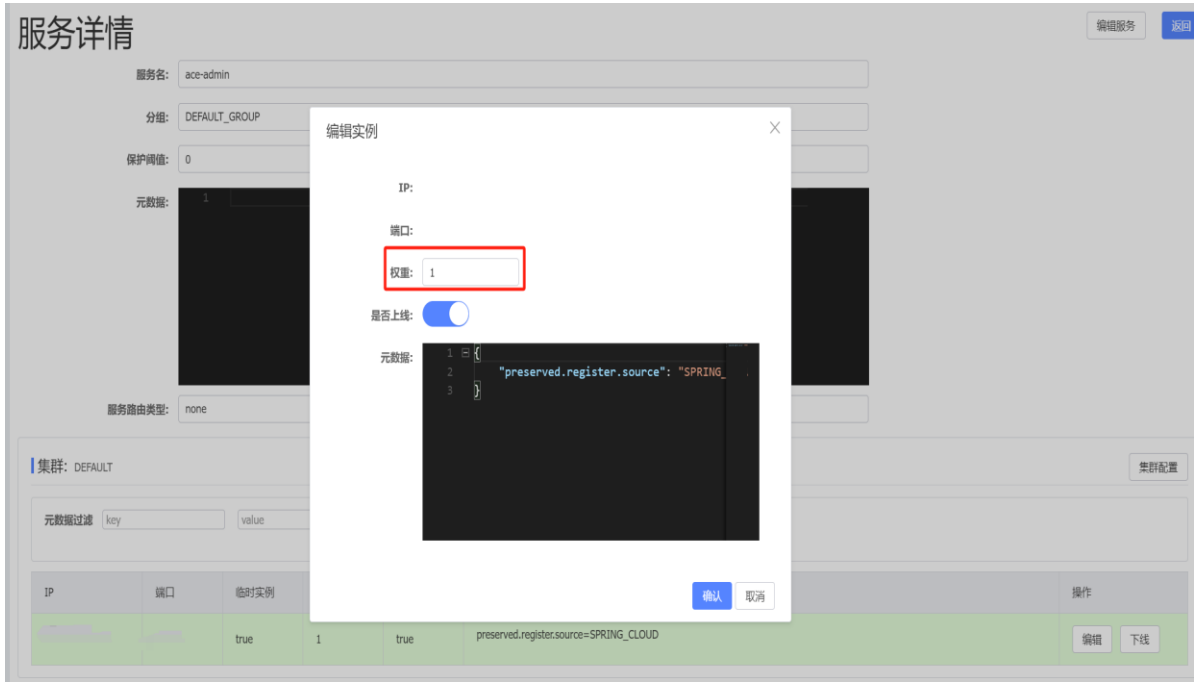
后端服务器：

主要环境	版本要求
JDK	1.8
MySQL	5.7
SpringBoot	2.4.1
SpringCloud	2020.0.0
Mybatis	3.3.1

阿里云Oss服务、Redis、lombok插件、maven插件均需完成安装配置，但对版本无要求，最终项目部署在华为云上。

服务注册中心：Nacos





作用：

1. 服务发现和服务健康监测

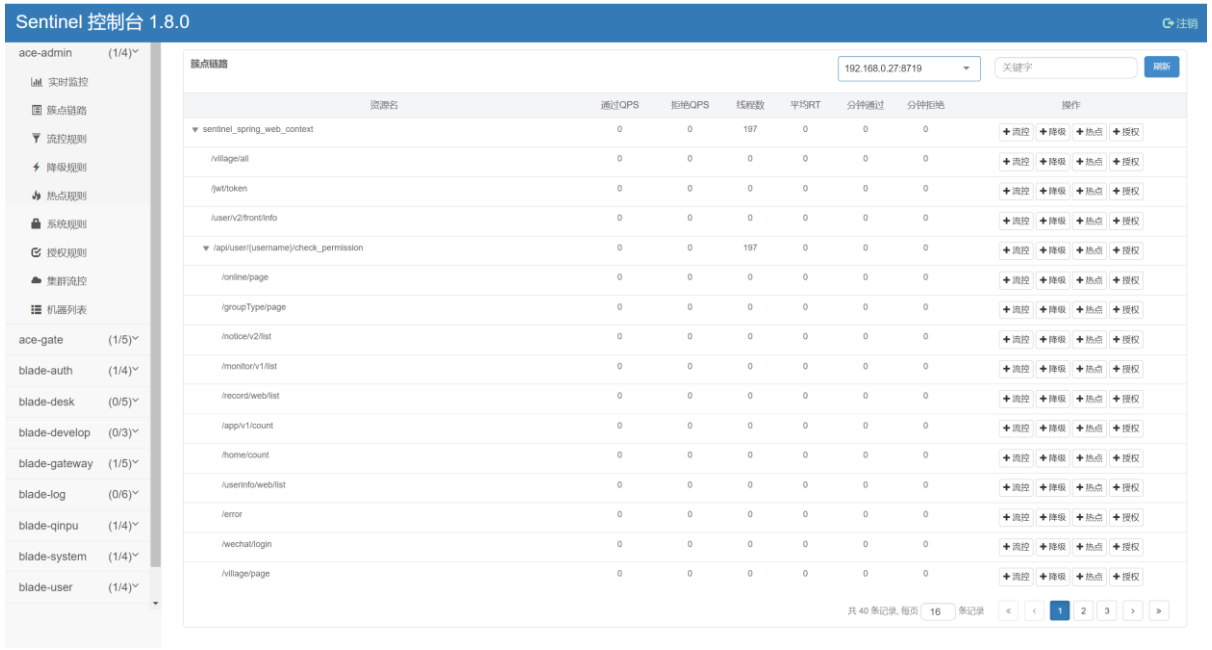
2. 动态配置服务

3. 动态DNS服务（权重路由）

4.服务及其元数据管理

流量卫兵：Sentinel

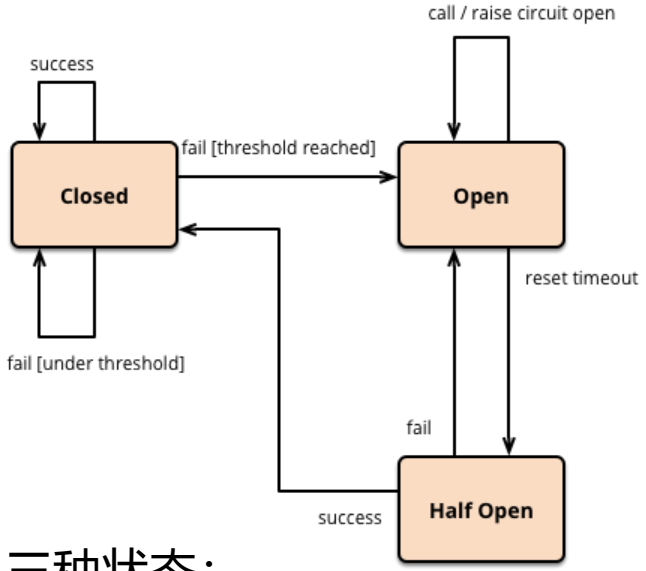
Sentinel核心作用就是流量控制，针对请求链路中的每一个环节做流量控制，从而保护系统。



实现要素：

- 资源：资源是指被Sentinel保护的目标对象，大部分情况，使用方法签名，URL，服务名称作为资源名来标识资源。
- 规则：针对目标资源所设定的流控规则，包括流量控制规则、熔断降级规则、系统保护规则，所有规则可以实时动态调整。

熔断机制：Hystrix




```
hystrix:
  threadpool:
    default:                    限流策略
      coreSize: 1000 ##并发执行的最大线程数，默认10
      maxQueueSize: 1000 ##BlockingQueue的最大队列数
      queueSizeRejectionThreshold: 500 ##即使maxQueueSize没有达到，达到queueSizeRejectionThreshold该值后，请求也会被拒绝
  command:
    default:                    熔断策略
      execution:
        isolation:
          thread:
            timeoutInMilliseconds: 10000 ##服务调用超时时间,THREAD隔离模式下是请求超时是会取消调用线程从而立即返回的，
              SEMAPHORE模式下会等待响应回来再判断是否超时。
```

三种状态：

- **熔断打开**-请求不再进行调用当前服务，内部设置时钟一般为MTTR（平均故障处理时间）当打开时长达到所设置时钟则进入半熔断状态。
- **熔断关闭**-熔断关闭，不会对服务进行熔断。
- **熔断半开**-部分请求根据规则调用当前服务，如果请求成功且符合规则认为当前服务恢复正常，关闭熔断，恢复链路。

当扇出链路的某个微服务出错不可用或者响应时间太长时，会进行服务的降级，进而熔断该节点微服务的调用，快速返回错误的响应信息。当检测到该节点微服务调用响应正常后，恢复调用链路。

消息队列：RabbitMQ

生产者往某个队列里面发送消息，一个队列可以存储多个生产者的消息，一个队列也可以有多个消费者，但是消费者之间是竞争关系，即每条消息只能被一个消费者消费。

作用：

1.异步处理

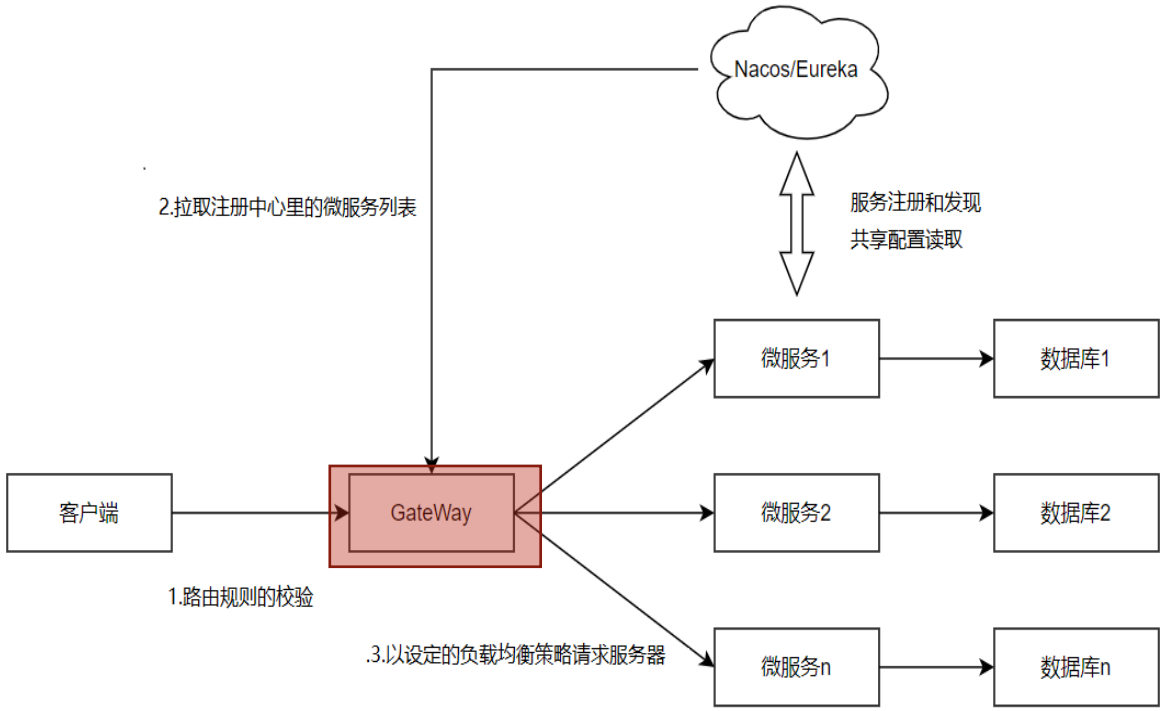
将消息写入消息队列，非必要的业务逻辑以异步的方式运行，加快响应速度。

2.服务解耦

某一个系统A要与其他系统通信，我们A系统将产生的数据发入消息队列中，其它的系统再去消息队列来进行消费，那么其他系统的减少或者新增系统即与A系统关系不大了，这样来实现解耦的功能。

3.流量控制

网关：概览



在危房改造管理系统中的**作用**:

1.路由转发

根据请求路由规则，向指定的微服务集群转发请求;

2.内部负载均衡

路由转发时，使用Robin负载均衡策略;

3.请求过滤

实现GlobalFilter接口，对请求的token进行过滤;

4.Feign服务

对鉴权微服务中的获取公钥api开启Feign服务。

网关：路由转发

```yaml
cloud:
  nacos:  # 注册中心
    discovery:
      server-addr: xxx.xxx.xxx.xxx:xxxx  # 需要根据实际配置nacos IP与端口号
  gateway:
    default-filters: # 默认过滤器配置(全局过滤器)
       - DedupeResponseHeader=Access-Control-Allow-Origin  # 允许跨域请求
    globalcors: #跨域配置
       add-to-simple-url-handler-mapping: true
       corsConfigurations:
         '[/**]':
           allowed-origins: "xxxxxx" #  运行跨域请求的网站
           allowed-methods: "*"  # 允许跨域的请求方式：所有
           allowed-headers: "*"  # 允许跨域请求携带的头信息头信息：所有
           allow-credentials: true  # 允许携带cookie
    discovery:
      locator:
        lowerCaseServiceId: true  # 请求路径上的服务名配置为小写
        enabled: true  # 通过注册中心创建路由
    routes:  # 自定义转发路径
      - id: ace-auth  # 目标服务ID
        uri: lb://ace-admin  # 在nacos中注册的目标服务URI
        order: 8000  # 路由顺序
        predicates: #断言，与下面条件相匹配的请求进行路由
        - RequestBody=true  #需要请求头
        - Path=/api/auth/**  #路径匹配
        filters:
        - StripPrefix=2  # 改写HTTP请求，转发时去掉前2个前缀
      - id: ace-admin
        uri: lb://ace-admin #注意和上一个服务是同一个URI，本质在同一个URI中
```

举例：

请求url：

xxx.xxx.xxx.xxx:xxxx/api/auth/captcha

xxx.xxx.xxx.xxx:xxxx为网关ip：port

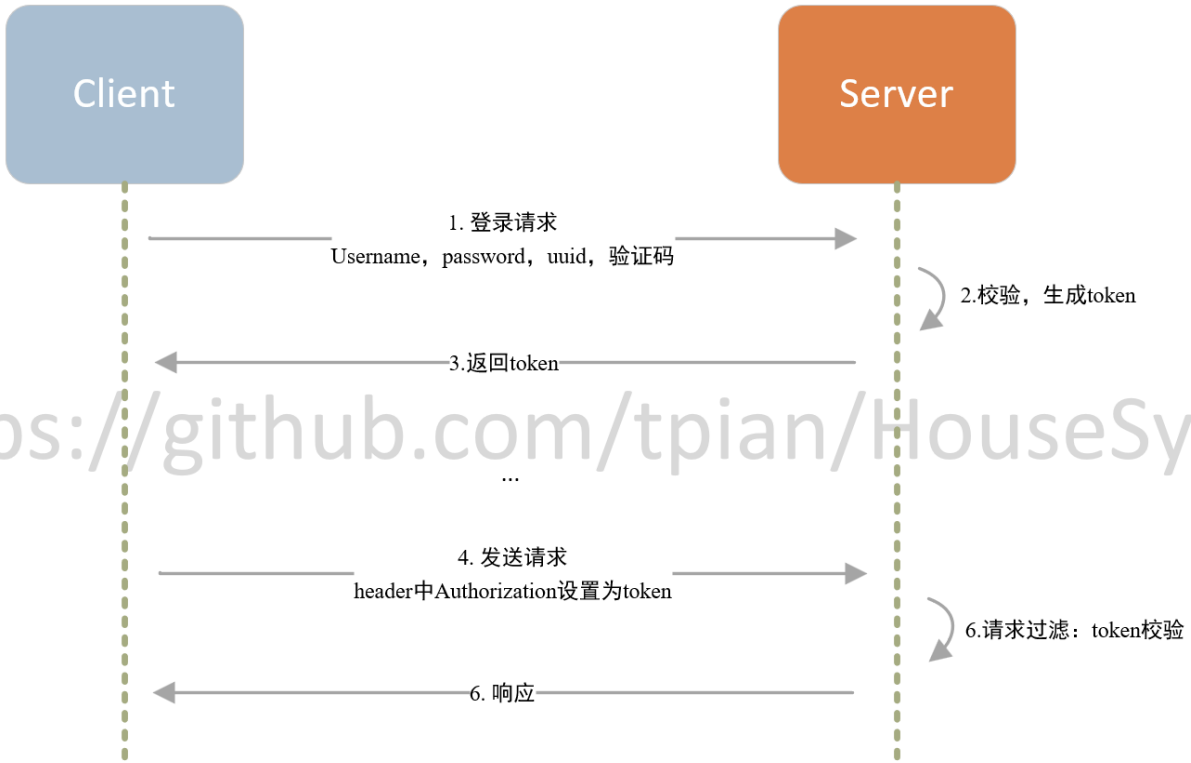
路径/api/auth/匹配到路由id：ace-auth，在**Nacos**中获取对应服务的ip与port

转发url：

xxx.xxx.xxx.xxx:xxxx/captcha

xxx.xxx.xxx.xxx:xxxx为ace-admin服务的ip：port

网关：请求过滤




```java
@Configuration
public class AccessGatewayFilter implements GlobalFilter {
   @Value("${gate.ignore.startWith}")
   private String startWith; # 通过application.yml配置忽略token校验的接口地址，如获取验证码、获取token接口
   @Autowired
   private StringRedisTemplate stringRedisTemplate; # token保存在redis中，存在过期时间

   // ...其他成员属性

   @Override
   public Mono<Void> filter(ServerWebExchange serverWebExchange, GatewayFilterChain gatewayFilterChain) {
      /**
         通过gatewayFilterChain.filter(exchange)编写前置滤器逻辑，在请求被路由之前调用，设计中仅用到前置过滤器
         通过gatewayFilterChain.filter(exchange).then(Mono.fromRunnable(() -> {过滤器逻辑}))编写后置滤器逻辑，在路由到微服务之后调用
         1.校验token是否过期、是否合法，从token获取用户信息校验用户是否具有接口权限
         2.检验用户权限时需要请求微服务中的对应接口
         3.接口地址开头为startWith的接口不需要校验token与权限
      */
   }

   // ...其他成员方法
}
```

过滤逻辑：

1. 校验uri是否需要检查token（uri开头判断）
2. 校验token是否合法、过期（需用到pubkey）
3. 获取用户信息，并校验该用户是否具有请求该uri权限（需访问微服务中的用户鉴权服务）

网关：Feign

四川大学 SICHUAN UNIVERSITY

1.依赖：

```
<dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>
```

2.Gateway启动类上的配置：

```
@EnableFeignClients({"com.github.wxiaoqi.security.auth.client.feign"})
```

表示当前模块需要用到 com.github.wxiaoqi.security.auth.client.feign包下的 feign 接口

3.包中feign接口的写法：

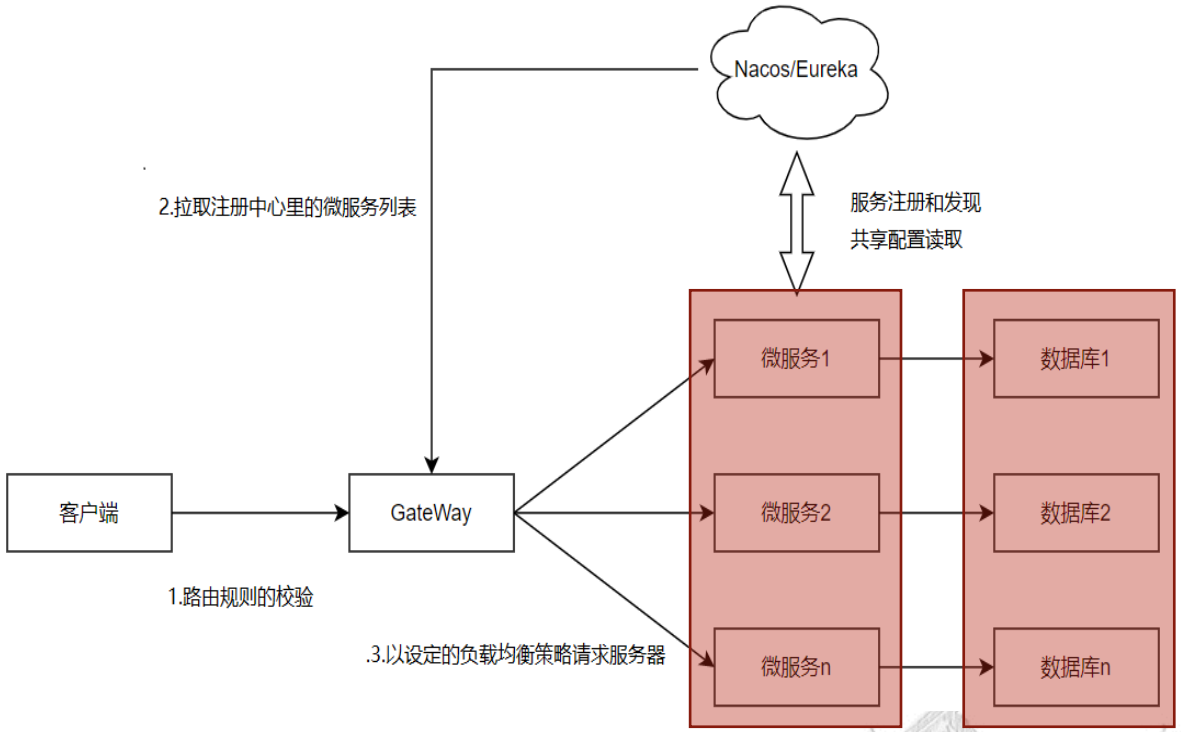
```
@FeignClient(value = "${auth.serviceId}",configuration = {})
public interface ServiceAuthFeign {
  @RequestMapping(value = "/client/userPubKey",method = RequestMethod.POST)
  public ObjectRestResponse<byte[]> getUserPublicKey(@RequestParam("clientId") String clientId, @RequestParam("secret") String secret);
```

4.在gateway中的类中调用feign接口中的方法：

```
@Scheduled(cron = "0 0/1 * * * ?")
public void refreshUserPubKey(){
    BaseResponse resp = serviceAuthFeign.getUserPublicKey(serviceAuthConfig.getClientId(), serviceAuthConfig.getClientSecret());
    if (resp.getStatusCode() == HttpStatus.OK.value()) {
        ObjectRestResponse<byte[]> userResponse = (ObjectRestResponse<byte[]>) resp;
        this.userAuthConfig.setPubKeyByte(userResponse.getData());
    }
}
```

效果：使得 gateway 在使用 HTTP 请求远程 center server 的 Auth 服务时能获得与调用本地方法一样的编码体验，开发者完全感知不到这是远程方法。

微服务：Center Sever



在危房改造管理系统中的**作用**：

1. 公钥私钥生成

启动服务时，随机生成公钥与对应私钥；

2. 请求拦截

继承HandlerInterceptorAdapter，重写preHandle方法；

3. 用户相关服务（个人工作）

JWT鉴权、用户登录退出，权限、角色、菜单分配；

4. 危房审批相关服务

提出申请、修改、查看、审批等其他政务。

在危房改造管理系统中的**数据库**：

1. MySQL　2. Redis　3. OSS

微服务：鉴权

JWT Token：

鉴权JWT

公钥密钥

- 用于对JWT头、有效载荷进行**签名**的非对称加密
- 启动服务时利用配置中的jwt.rsa-secret、随机数与KeyPairGenerator生成一对公、密钥，公钥会提供给**gateway**

用户信息

负载中的私有字段

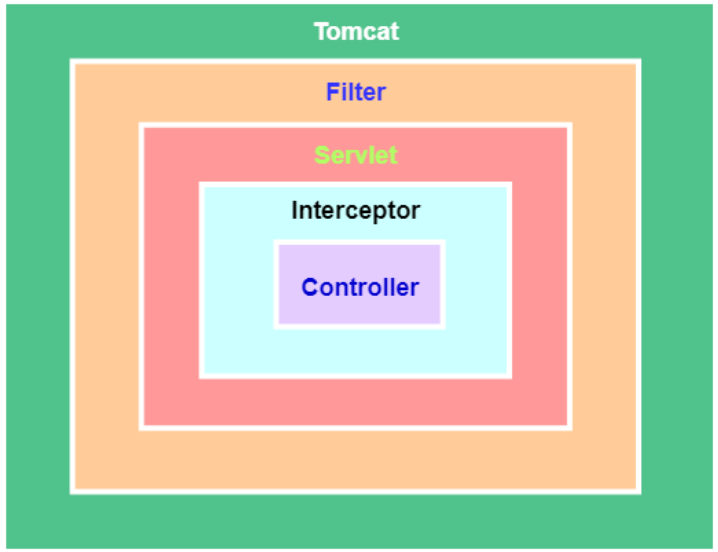
- 包含用户id、token id、用户姓名
- 用于在**拦截器**中解密获取发起请求的用户信息

过期信息

负载中的官方字段之一

- 标志token的过期时间
- 用于在**Gateway过滤器**中解密判断token是否过期

微服务：请求拦截



拦截器的**作用位置**：

Interceptor 是在请求进入servlet后，在进入Controller之前进行预处理的，Controller 中渲染了对应的视图之后请求结束。

在危房改造管理系统中的**作用**：获取用户信息、路径拦截、全局异常处理

1. 继承HandlerInterceptorAdapter，并重写preHandle方法

```
@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {
    // 配置该注解，说明不进行用户拦截
    String token = request.getHeader(userConfiguration.getUserTokenHeader());
    if (StringUtils.isEmpty(token)) {
        if (request.getCookies() != null) {
            for (Cookie cookie : request.getCookies()) {
                if (cookie.getName().equals(userConfiguration.getUserTokenHeader())) {
                    token = cookie.getValue();
                }
            }
        }
    }
    IJWTInfo infoFromToken = jwtTokenUtil.getInfoFromToken(token);
    BaseContextHandler.setUsername(infoFromToken.getUniqueName());
    BaseContextHandler.setName(infoFromToken.getName());
    BaseContextHandler.setUserID(infoFromToken.getId());
    return super.preHandle(request, response, handler);
}
```

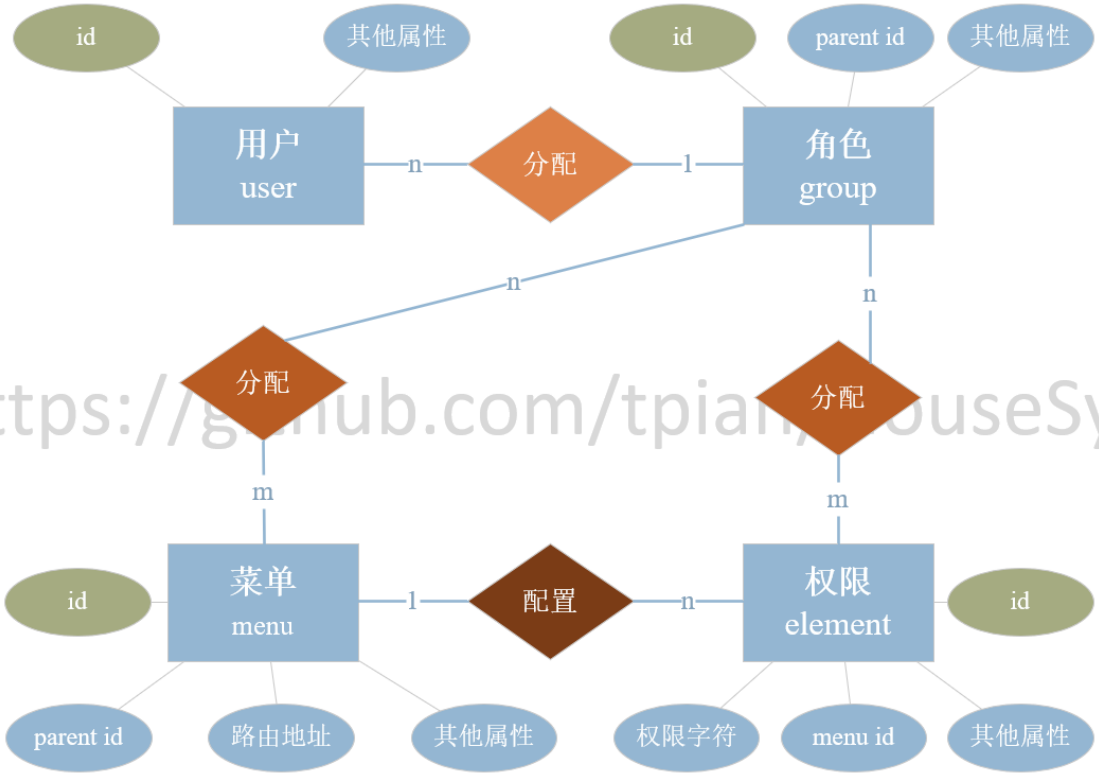
2. 实现WebMvcConfigurer，并重写addInterceptors方法

```
@Configuration("admimWebConfig")
@Primary
public class WebConfiguration implements WebMvcConfigurer {
    @Bean
    GlobalExceptionHandler getGlobalExceptionHandler() { return new GlobalExceptionHandler(); }

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(getUserAuthRestInterceptor()).addPathPatterns(getIncludePathPatterns());
    }
```

微服务：用户管理

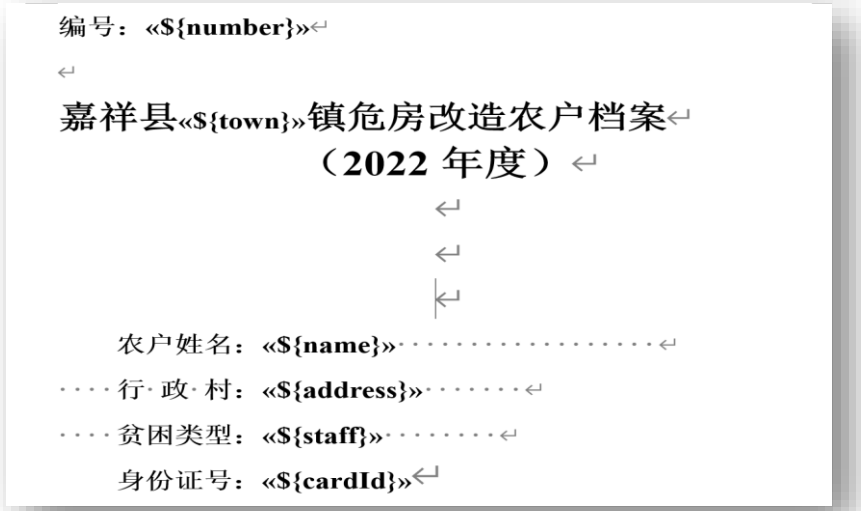
数据库UML



- 通过角色表，解除用户与菜单权限的耦合，减少用户与菜单修改时的影响；
- 为每个登录用户动态生成登录菜单，动态页面内容展示。

微服务：文档导出

规则：具备《农村危房改造建房备案书》的危房改造申请才能进行文档导出


编号：«${number}»

嘉祥县«${town}»镇危房改造农户档案
（2022 年度）

农户姓名：«${name}»

行政村：«${address}»

贫困类型：«${staff}»

身份证号：«${cardId}»

准备：docx模版文件

```
<dependency>
    <groupId>fr.opensagres.xdocreport</groupId>
    <artifactId>fr.opensagres.xdocreport.core</artifactId>
    <version>2.0.2</version>
</dependency>
```

准备：主要依赖xdocreport

```
1     public String word(ArchivesVO archivesVO, String docName) {
2         OutputStream out = null;
3         try {
4             InputStream ins = this.getClass().getResourceAsStream("/模板.docx");
5             IXDocReport report = XDocReportRegistry.getRegistry().loadReport(ins,
  TemplateEngineKind.Velocity);
6             IContext context = report.createContext();
7             FieldsMetadata fm = report.createFieldsMetadata();
8             fm.addFieldAsImage("front");
9             fm.addFieldAsImage("back");
```

1. 获取作用域

```
49            context.put("number", archivesVO.getUserId());
50            context.put("town", archivesVO.getTown());
51            context.put("town02", archivesVO.getTown());
52            context.put("town03", archivesVO.getTown());
```

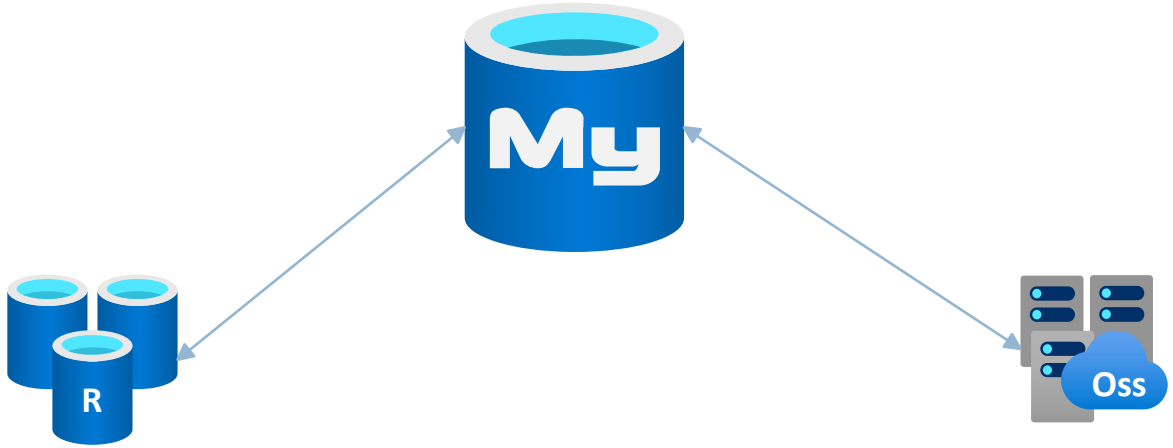
2.1 文本内容替换

```
165           context.put("urge", getImg(archivesVO.getUrge()));
166           context.put("filing", getImg(archivesVO.getFiling()));
167           context.put("transfer", getImg(archivesVO.getTransfer()));
168           String filename = encodingFilename(docName);
169           out = new FileOutputStream(getAbsoluteFile(file
```

2.2 图片内容替换

实现：作用域替换

微服务：数据库



	应用	举例
MySQL	主要使用的数据库，持久化数据存储。	用户信息、菜单信息等
Redis	非永久数据存储； 旁路缓存：对读多写少的数据，提供旁路缓存，加快响应速度。	token、pubkey、prikey、登录信息、验证码 将全部权限接口的响应以JSON串的格式存入Redis中并设置较长的过期时间如1h
OSS	阿里云的对象存储服务，用于数据库中的大对象存储	将图片存入OSS，在MySQL数据库中保存对应URL

4

PART
Four

使用情况

小程序

登录

注册

首页

事件通报

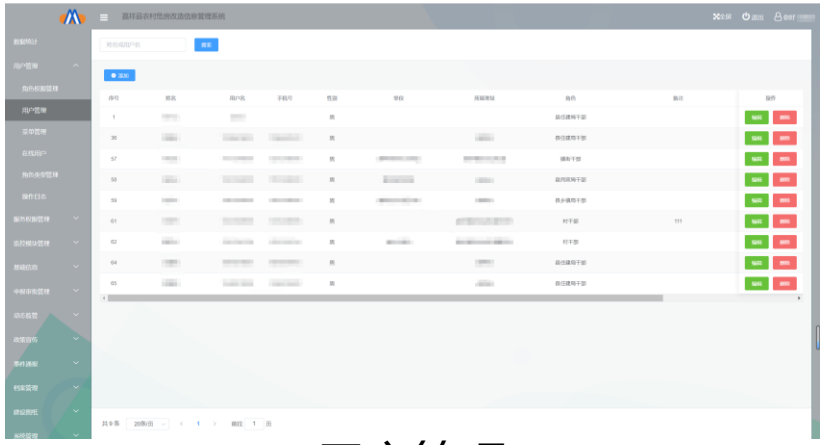
建设图纸

动态监管

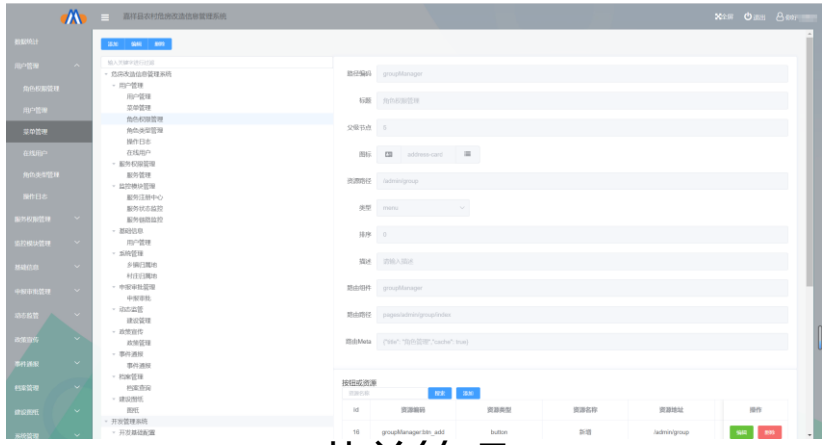
小程序端使用用户：198村民+9干部

Web管理

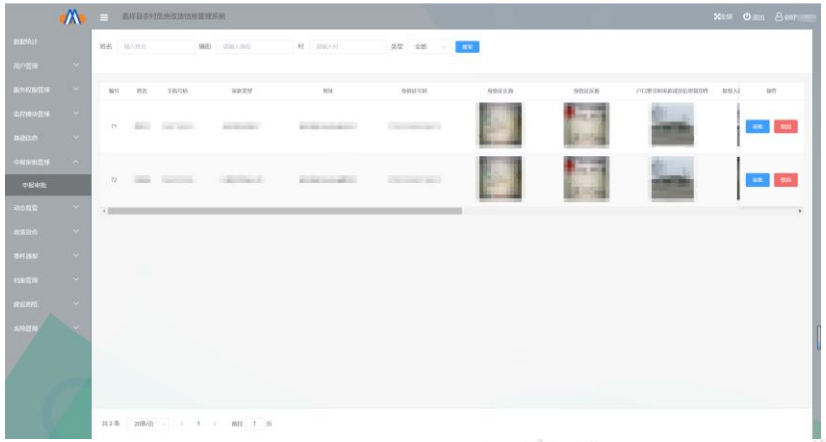
登录



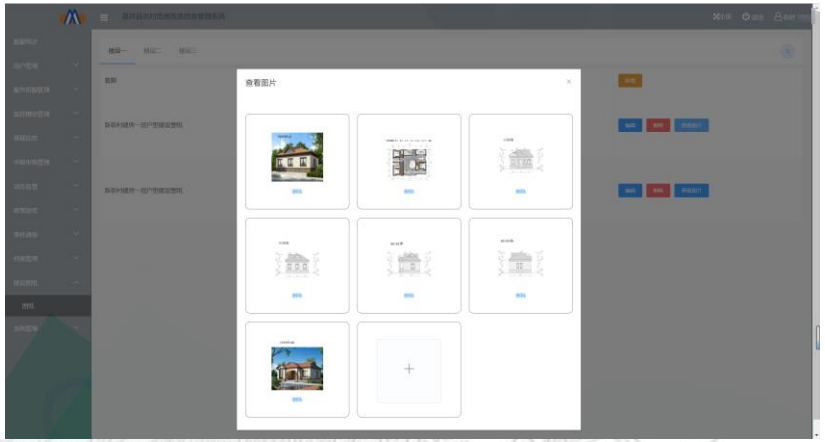
用户管理



菜单管理



申报审批



建设图纸管理

用户数据看板

Web管理端使用用户：9干部