服务网关设计

评审

```
参与：立东、姜睿、志东、玉强
时间：2022.03.03
地点：公司办公室
```

结论

1. 技术方案可行。
2. 志东：详细需求原型提供。
3. 立东：熟悉控制面代码
4. 玉强：负责数据面

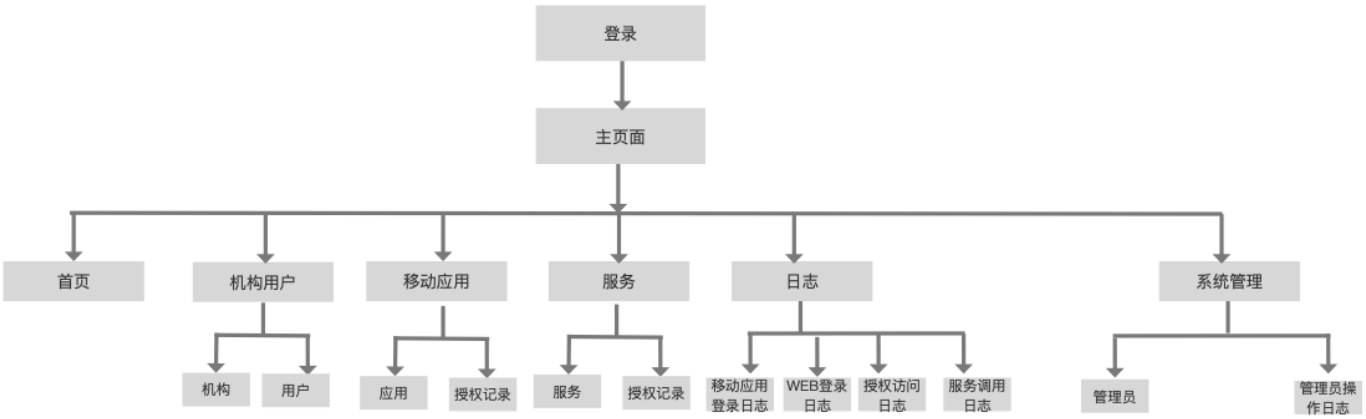
需求来源

jw项目中客户要求提供服务网关，服务调用者需要认证后才可以调用服务提供者，且可支持服务提供者授权，调用者可分为app、服务。具体如下图：

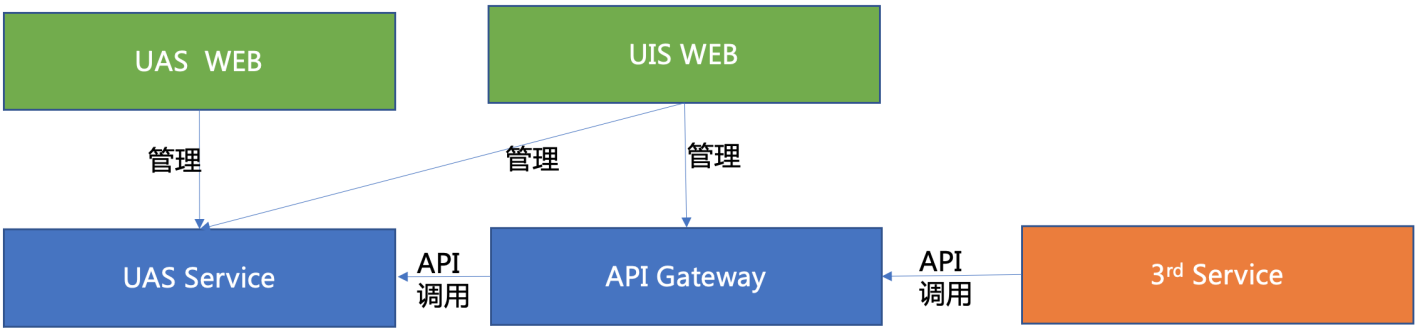
序号	指标项	指标要求	指标要求
1	总体要求	建设北京铁路公安局统一的人员信息管理服务，为北京铁路公安局各级单位提供组织机构和人员管理入口服务和接口调用服务	
2	产品形态	部署于虚拟化主机集群的服务系统，为用户提供WEB应用形态客户端	
3	人员管理	提供北京铁路公安局人员基础信息的导入、查询、删除、修改，并支持批量导入和批量导出	
4	组织管理	提供北京铁路公安局单位下组织机构信息的创建、修改、删除,支持提供人员和组织机构的管理，支持人员信息的批量附着	
6	分级管理	并提供分级、分权限管理能力	把“分权限管理” 拿掉，只保留分级管理
7	接口设计	提供请求服务能力接口，支持REST或WebService形式的网络接口调用封装服务；	
8	权限管控	支持针对第三方应用系统提供分权限、细粒度的接口调用和结果反馈	
9	日志功能	对系统的操作进行日志记录，具备完整的日志记录功能，日志保留6个月以上。	
10	部署环境	部署于北京铁路公安局移动警务虚拟化集群主机之上	
11	部署区域	北京铁路公安局移动警务平台公安信息网服务子平台	
12	数据导出	支持将人员信息、组织机构信息导出为excel格式文件	

架构

页面结构



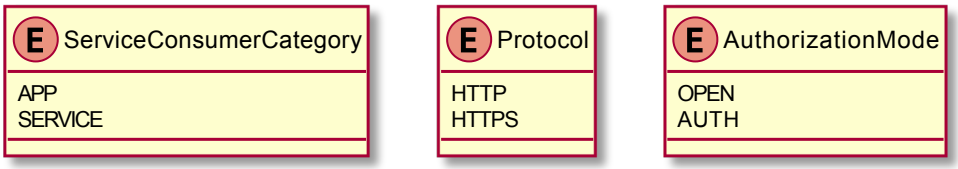
组件架构

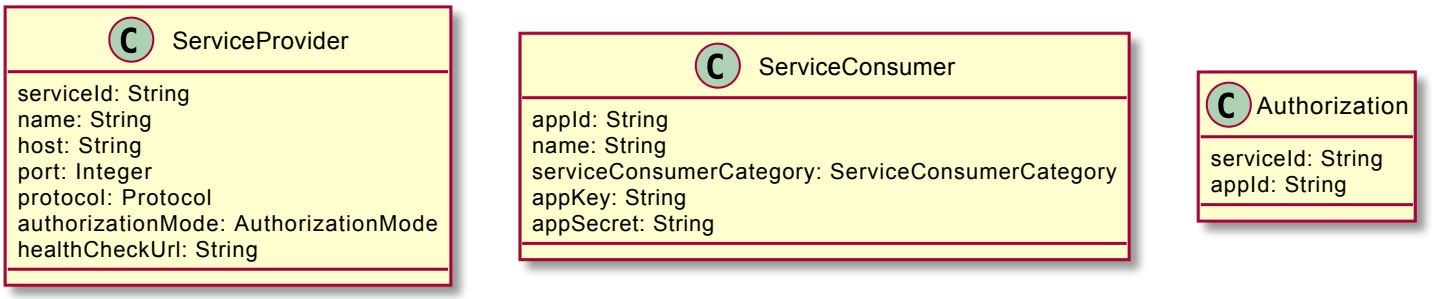


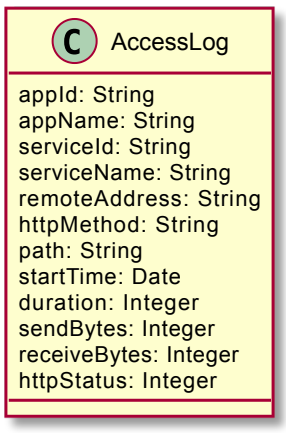
需求分析

按照网关技术特点，可将网关分为控制面、数据面，控制面来负责维护服务提供者、服务调用者，数据面根据控制面定义的策略项执行相关策略及功能需求。

定义







1. 服务提供者
2. 服务调用者
3. 授权
4. 调用日志

控制面

1. 服务提供者管理（增删改查）
2. 服务调用者管理（增删改查）
3. 服务提供者授权管理（授权、撤销授权）
4. 调用日志查看、统计
5. 服务调用方认证（采用HMAC）

数据面

1. 认证路由到控制面
2. 对服务调用请求认证校验（JWT）、授权校验
3. 服务调用根据请求路径路由到对应服务
4. 记录服务调用请求日志

技术组件

mag定制配置

envoy支持七层http流量的路由及过滤管理，利用此能力支持服务网关的根据路径路由服务的业务定制。http按路径路由请求，类似nginx中配置的路径匹配然后路由到后端服务，此需求可通过原始envoy配置文件相关配置项来支持。 从需求分析来看，主要需要envoy自身来提供的需求包括如下几点：

1. 对认证服务请求路由到后端控制面 >可通过配置http的route_config实现
2. 对服务调用请求认证校验（JWT） >可通过配置http过滤器envoy.filters.http.jwt_authn实现，校验通过后需要将jwt中相关应用信息配置到当前流信息的元数据中，用于后面授权校验使用
3. 对服务调用请求授权校验 >需要在envoy源码路由相关业务逻辑中增加授权校验逻辑，前面的过滤器JWT已经将认证通过的app相关信息放到了当前流信息的元数据中，根据配置做校验并执行响应的动作即可。
4. 对服务调用请求根据配置路由到对应服务 >可通过配置http的route_config实现
5. 记录服务调用请求日志 >可通过配置http的access_log实现

参考如下配置文件支持服务网关的业务需求：

```yaml
node:
  user_agent_name: mag
  user_agent_version: 1.0.0
#  配置自定义头，envoy内部定义的一些http请求响应头前缀
header_prefix: x-mag
static_resources:
  listeners:
    - name: mag_http
      address:
#         网关地址端口
        socket_address: { address: 0.0.0.0, port_value: 4502 }
      filter_chains:
        - filters:
          - name: envoy.http_connection_manager
            typed_config:
              "@type": type.googleapis.com/envoy.config.filter.network.http_conn
ection_manager.v2.HttpConnectionManager
#                统计信息前缀
              stat_prefix: mag_http
              codec_type: AUTO
#                路由表定义，可以在此处定义通过路径到某个后端服务的请求的转换与路径替换
              route_config:
                name: mag_http_route
                virtual_hosts:
                  - name: mag_http_service
                    domains: [ "*" ]
                    routes: # 路由表，固定前缀：/mag/
                      # 后面跟服务标识，服务标识应该在服务网关中唯一
                      - match: { prefix: '/mag/baidu/' }
                        route:
                          host_rewrite: 'www.baidu.com'
                          cluster: service_baidu
                          prefix_rewrite: '/'
                      - match: { prefix: '/mag/pm/' }
                        route:
                          prefix_rewrite: '/'
                          cluster: service_pm
              http_filters:
                - name: envoy.filters.http.jwt_authn
                  typed_config:
                    "@type": type.googleapis.com/envoy.extensions.filters.http.j
wt_authn.v3.JwtAuthentication
#                     配置jwt认证
                    providers:
                      mag_server:
                        issuer: https://www.pekall.com # 该JWT的签发者
                        audiences: # 接收该JWT的一方
```

```
                            - mag.pekall.com
                          forward: false # 不转发此jwt请求头
                          # 将校验通过的载荷json字符串结构放进StreamInfo DynamicMetadat
a,
                          # key是：envoy.filters.http.jwt_authn，属性名是jwt
                          payload_in_metadata: jwt
#                           jwt是protobuf的Struct结构体，我们设计jwt包含如下属性：
#                           aud: mag.pekall.com
#                           exp: 过期时间
#                           iat: 签发时间
#                           iss: 签发者
#                           nbf: 生效起始时间
#                           sub: 授权主体，此处设计用来配置为app标识，和授权关联的app标识
一致，以在授权校验中检测授权关系
                          local_jwks: # 此处定义了jwks，用于校验jwt，包含公钥及相关算法参
数即可
                            inline_string: |
                              {
                                "keys": [
                                  {
                                    "kty":"EC",
                                    "crv":"P-256",
                                    "x":"1eGbUrtZupfIq6fJGef4f9Zg0WLfIhp4e2KdOBNyh
JE",
                                    "y":"2Cvam8fdoC7SybTbOcd9tSFhZ-0ZhAORJwRaxxz1w
b8",
                                    "alg":"ES256"
                                  }
                                ]
                              }
                      rules:
                        - match:
                            prefix: /
                          requires:
                            provider_name: mag_server
                  - name: envoy.router
                server_name: mag
                access_log:
                  - name: "envoy.file_access_log"
                    config:
                      typed_json_format:
                        start: '%START_TIME%'
                        method: '%REQ(:METHOD)%'
                        path: '%REQ(X-MAG-ORIGINAL-PATH?:PATH)%'
                        proto: '%PROTOCOL%'
                        sent: '%BYTES_SENT%'
                        received: '%BYTES_RECEIVED%'
                        duration: '%DURATION%'
```

```
                    cluster: '%UPSTREAM_CLUSTER%'
                    host: '%UPSTREAM_HOST%'
                    remote_addr: '%DOWNSTREAM_REMOTE_ADDRESS%'
                    resp_status: '%RESPONSE_FLAGS%'
                    resp_code: '%RESPONSE_CODE%'
                    user_agent: '%REQ(USER-AGENT)%'
                    time: '%RESP(X-MAG-UPSTREAM-SERVICE-TIME)%'
                    forward: '%REQ(X-FORWARDED-FOR)%'
                    req_id: '%REQ(X-REQUEST-ID)%'
                    authority: '%REQ(:AUTHORITY)%'
                  path: "my/logs/access.log"

  clusters:
    - name: service_baidu
      connect_timeout: 0.25s
      type: LOGICAL_DNS
      lb_policy: ROUND_ROBIN
      metadata: # 服务授权管理通过此处配置服务集群的元数据来支持
        filter_metadata:
          mag:
            authorize: TRUE # TRUE：需授权，FALSE：开放，不需要授权
            app_list:
              - app1
              - app2
      load_assignment:
        cluster_name: service_baidu
        endpoints:
          - lb_endpoints:
            - endpoint:
              address:
                socket_address:
                  address: www.baidu.com
                  port_value: 443
      transport_socket:
        name: envoy.transport_sockets.tls
        typed_config:
          "@type": type.googleapis.com/envoy.api.v2.auth.UpstreamTlsContext
          sni: www.baidu.com
    - name: service_pm
      type: STATIC
      connect_timeout: 0.25s
      metadata: # 服务授权管理通过此处配置服务集群的元数据来支持
        filter_metadata:
          mag:
            authorize: TRUE # TRUE：需授权，FALSE：开放，不需要授权
            app_list:
              - app2
      load_assignment:
```

```
        cluster_name: service_pm
        endpoints:
          lb_endpoints:
            - endpoint:
              address:
                socket_address:
                  protocol: TCP
                  address: '192.168.11.34'
                  port_value: 80
admin:
  access_log_path: "my/logs/admin-access.log"
  address:
    socket_address:
      address: 0.0.0.0
      port_value: 4501
```

测试用秘钥对

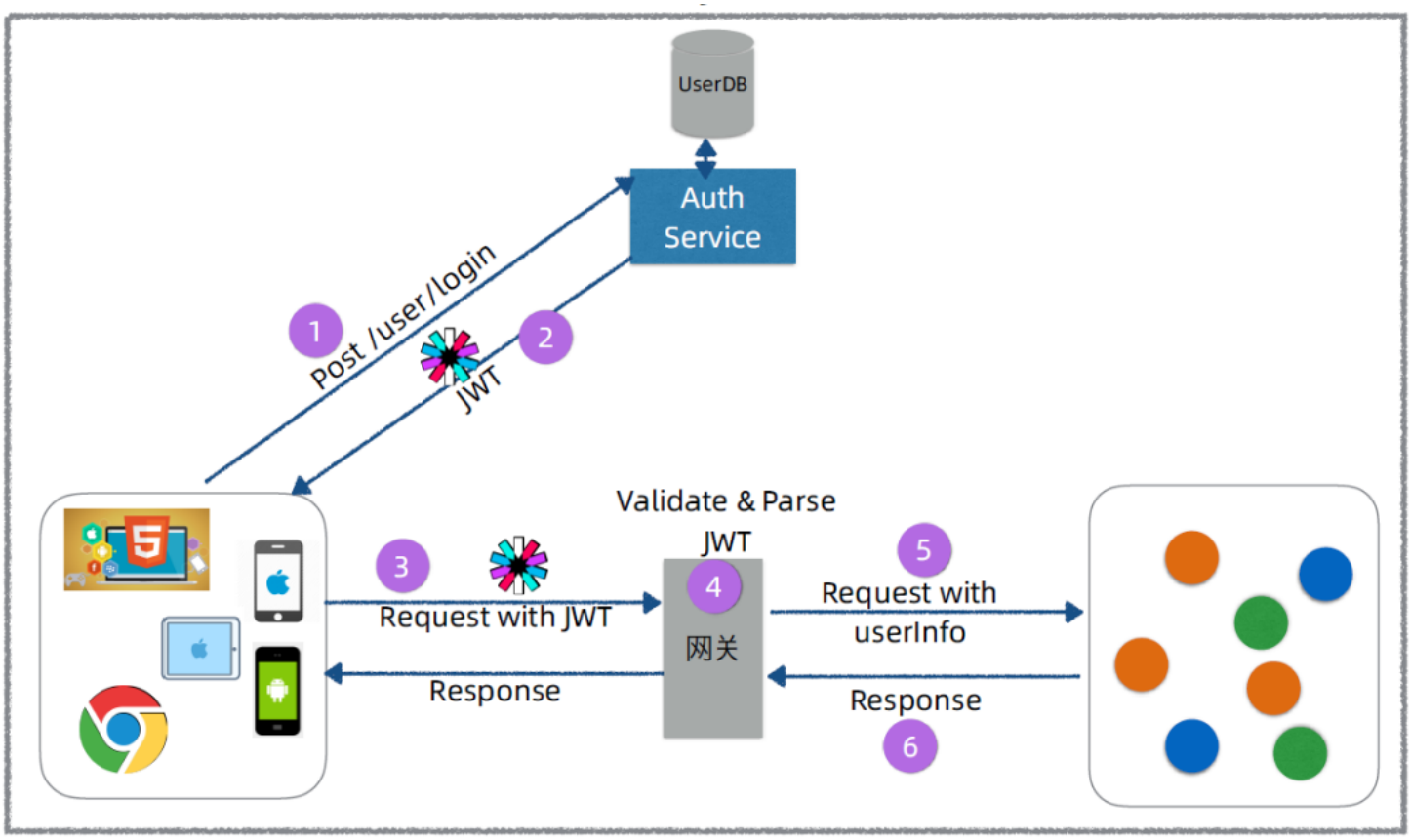
```
{
  "kty": "EC",
  "d": "yB6n8hX3qR89-p9EnKDIUpp7dBc7-1oJ5XEXjG5n6zQ",
  "crv": "P-256",
  "x": "1eGbUrtZupfIq6fJGef4f9Zg0WLfIhp4e2KdOBNyhJE",
  "y": "2Cvam8fdoC7SybTbOcd9tSFhZ-0ZhAORJwRaxxz1wb8",
  "alg": "ES256"
}
```

测试脚本

```
curl -kv http://localhost:4502/mag/baidu/ -H "Authorization: Bearer eyJhbGciOiJFUzI1NiIsImp3ayI6eyJrdHkiOiJFQyIsImNydiI6IlAtMjU2IiwieCI6IjFlR2JVcnRadXBmSXE2ZkpHZWY0ZjlaZzBXTGZJaHA0ZTJLZE9CTnloSkUiLCJ5IjoiMkN2YW04ZmRvQzdTeWJUYk9jZDl0U0ZoWi0wWmhBT1JKd1JheHh6MXdiOCIsImFsZyI6IkVTMjU2In19.eyJhdWQiOiJtYWcucGVrYWxsLmNvbSIsInN1YiI6ImFwcDEiLCJuYmYiOjE2NDQ0NTg0NDksImlzcyI6Imh0dHBzOlwvXC93d3cucGVrYWxsLmNvbSIsImV4cCI6MTk1OTk5MTI0OSwiaWF0IjoxNjQ0NDU4NDQ5fQ.cAhPPWEXZ-rJjvRN4R-SiXujED6GBJVPS2CRe_azVFw46vklf_QlTRjuPgmtBn2j5pFyOE6oW17J51NvjxOosg"
```

参考

JWT



OAuth-Client

```
+---------+                                  +---------------+
|         |                                  |               |
|         |>--(A)- Client Authentication --->| Authorization |
| Client  |                                  |     Server    |
|         |<--(B)---- Access Token ---------<|               |
|         |                                  |               |
+---------+                                  +---------------+
```

可以看到非常简单，他其实只要:

A、Client携带授权信息（client_id,client_scret，scopes,grant_type等）去Authorization Server 申领AccessToken；

B、Authorization Server 颁发AccessToken；

然后你就可以用这个AccessToken 调用 受保护的接口了；