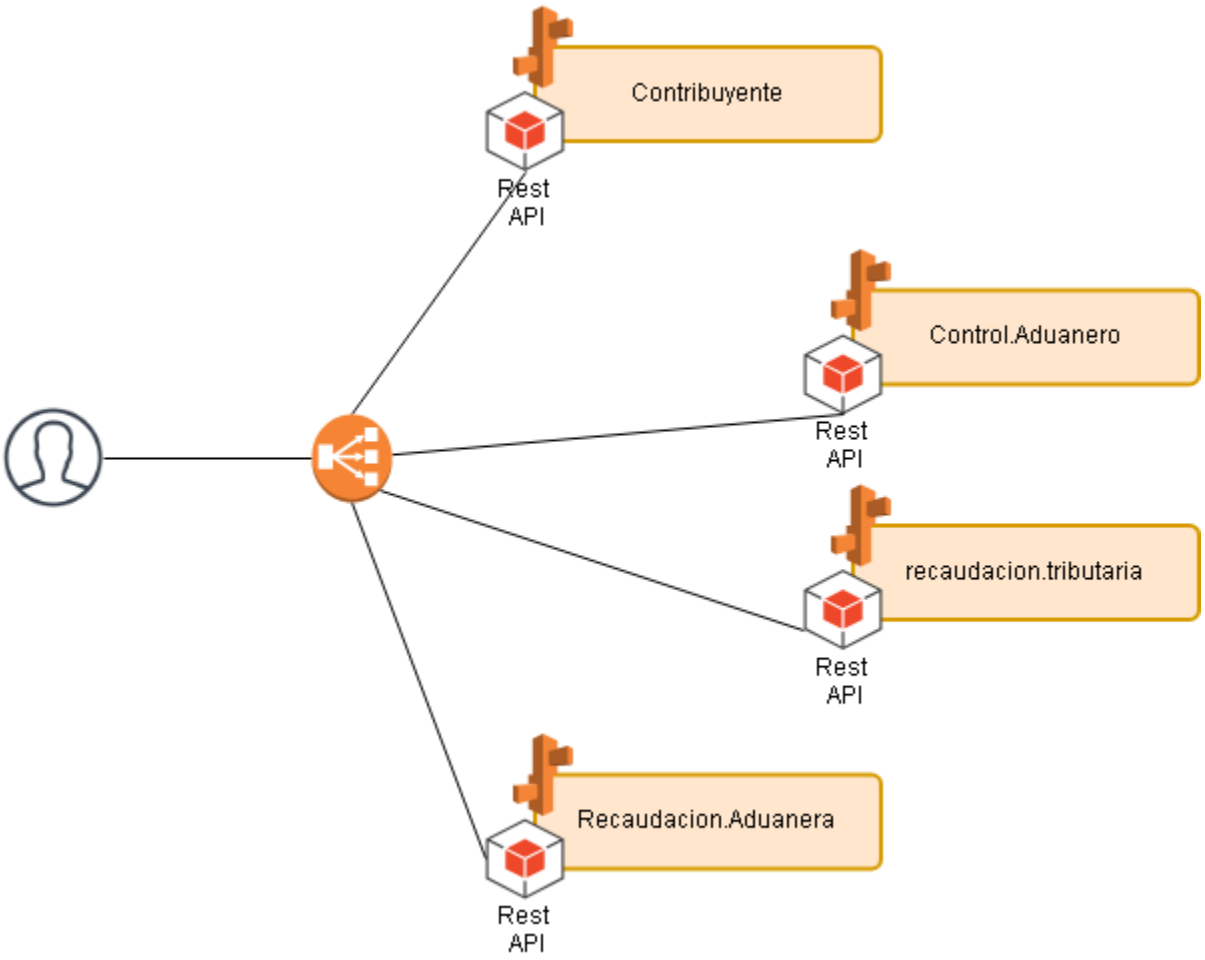
API Gateway

1.0. Introducción

La función principal de un API Gateway es proporcionar un único punto de entrada constante para múltiples API, independientemente de cómo se implementen o implementen en el back-end. No todas las API son aplicaciones de microservicios. Nuestra puerta de enlace API necesita administrar API, monolitos y aplicaciones existentes que experimentan una transición parcial a microservicios.

Un API Gateway es un servidor que es el único punto de entrada al sistema. Es similar al patrón de Facade del diseño orientado a objetos. API Gateway encapsula la arquitectura interna del sistema y proporciona una API que se adapta a cada cliente. Puede tener otras responsabilidades como la autenticación, la supervisión, el equilibrio de carga, el almacenamiento en caché, la configuración y administración de solicitudes y el manejo de respuestas estáticas.

El siguiente diagrama muestra cómo una puerta de enlace API normalmente se ajusta a la arquitectura:



API Gateway es responsable del enrutamiento de solicitudes, la composición y la traducción de protocolos. Todas las solicitudes de los clientes pasan primero a través de la API Gateway. Luego dirige las solicitudes al servicio apropiado. La API Gateway a menudo manejará una solicitud invocando múltiples servicios y agregando los resultados.

2.0. Configuración

La presente sección describe los pasos referenciales para configurar un servidor NGINX como API GateWay, teniendo como directorio base "/etc/nginx" al que denominaremos como "**$NGINX_HOME**".

1. ## Registro "alias" en el archivo "hosts" del Servidor NGINX

   En el archivo "**/etc/hosts**" registramos la siguiente información:
   Asociamos la IP del servidor NGINX con el SubDominio que recibirá las peticiones.

   ```
   <IP servidor>       <server.name>
   <IP servidor NGINX> <subdominio>
   ```

   Por ejemplo:

   ```
   192.168.56.xx  api.sunat.gob.pe
   192.168.56.1xx api.sunat.peru
   ```

2. ## Configuración de NGINX como API GateWay

   Dentro del directorio **$NGINX_HOME**, encontraremos los siguientes recursos:

Recurso	Tipo	Descripción
nginx.conf	Archivo	Archivo de configuración principal del servidor. No realizamos ninguna modificación en este archivo.
conf.d	Directorio	Directorio en el que se colocan archivos de configuración personalizados que serán cargados por el servidor. En este directorio colocaremos los archivos con la configuración (*.conf) de nuestros API GateWay.

3. ## Organizando la configuración de NGINX como API GateWay

   La siguiente es la estructura de archivos al finalizar la configuración.

   ```
   nginx/
    +-- conf.d
    ¦   +-- api_backends.conf
    ¦   +-- api.sunat.gob.pe.443.conf
    ¦   +-- api.sunat.gob.pe.443.conf.d
    ¦   ¦   +-- api_contribuyente.conf
    ¦   ¦   +-- api_error_messages.conf
    +-- jwk
    ¦   +-- api_secret.jwk
   ```

Recurso	Tipo	Descripción
conf.d	Directorio	Directorio en el que se colocan archivos de configuración personalizados que serán cargados por el servidor. En este directorio colocaremos los archivos con la configuración (*.conf) de nuestros API GateWay.

api_backends.conf	Archivo	Define el servidor virtual que expone NGINX Plus como una puerta de enlace API a los clientes.
api.sunat.gob.pe.443.conf	Archivo	Describe con más precisión los recursos implementados por los servicios de backends
api.sunat.gob.pe.443.conf.d	Directorio	Directorio en el que se colocan archivos de configuración personalizados que serán cargados por el servidor. En este directorio colocaremos los archivos con la configuración (*.conf) de nuestros API GateWay. Cada "Sub-Dominio" debe tener asociado un solo archivo de configuración.
api_contribuyente.conf	Archivo	Configuración del servicio para cada uno de los sub-Dominio
api_error_messages.conf	Archivo	Respuestas de error HTTP en formato JSON
jwk	Directorio	Directorio en el que se colocan archivos de configuración para el Servicio de Autenticación.
api_secret.jwk	Archivo	Especificación JWK para representar las claves criptográficas utilizadas para firmar tokens RS256

4. ## Contenido de un archivo de configuración:

A continuación se describen las secciones mínimas que se necesitan para configurar el API GateWay.

Sección:	**api_backends.conf**

	Aquí utilizamos múltiples pares de dirección IP-puerto en cada bloque `upstream` para indicar dónde se implementa el código
	`upstream upstreamBackendAPI1 {`
	`  zone inventory_service 64k;`
	`  server 10.0.0.1:80;`
	`  server 10.0.0.2:80;`
	`  server 10.0.0.3:80;`
	`}`
	`upstream upstreamBackendAPI2 {`
	`  zone pricing_service 64k;`
	`  server 10.0.0.7:80;`
	`  server 10.0.0.8:80;`
	`  server 10.0.0.9:80;`
	`}`
	Por ejemplo,"api.sunat.peru" quedaría definido de la siguiente forma:
	`upstream upstreamApiSunatPeru {`
	`  zone upstreamApiSunatPeru 64k ;`
	`  server api.sunat.peru;`
	`}`
	Donde
	Línea / **Descripción**
	1-4 / Definición de API.

Sección:	**api.sunat.gob.pe.443.conf**
	Definimos las siguientes instrucciones:

```
log_format api_main '$remote_addr - $remote_user [$time_local] "$requ
                    '$status $body_bytes_sent "$http_referer" "$http_
                    '"$http_x_forwarded_for" "$api_name"';

server {
    set $api_name -;
    access_log /var/log/nginx/api_access.log api_main;

    listen 443 ssl;
    server_name api.example.com;

    # TLS config
    ssl_certificate      /etc/ssl/certs/api.example.com.crt;
    ssl_certificate_key  /etc/ssl/private/api.example.com.key;
    ssl_session_cache    shared:SSL:10m;
    ssl_session_timeout  5m;
    ssl_ciphers          HIGH:!aNULL:!MD5;
    ssl_protocols        TLSv1.1 TLSv1.2;

    # API definitions, one per file
    include api_conf.d/*.conf;

    # Error responses
    error_page 404 = @400;         # Invalid paths are treated as bad
    proxy_intercept_errors on;     # Do not send backend errors to th
    include api_json_errors.conf;  # API client friendly JSON error r
    default_type application/json; # If no content-type then assume J
}
```

Donde:

Línea	Descripción
1-3	directiva para cambiar el formato de los mensajes registrados
6	Comienza con un nombre de API no definido, cada API actualizará este valor.
7	Cada API también puede iniciar sesión en un archivo separado
9	Describe todas las direcciones y puertos que deberían aceptar conexiones para el servidor,
13-18	Protegido por TLS como está configurado en las líneas.
21	Los datos del API individuales y sus servicios backend se especifican en los archivos referenciados por la `include`
24-27	Configuración de manejo de errores.

Por ejemplo,"api.sunat.peru" quedaría definido de la siguiente forma:

```
log_format api_main '$remote_addr - $remote_user [$time_local] "$requ
                    '$status $body_bytes_sent "$http_referer" "$http_
                    '"$http_x_forwarded_for" ** dominio: "$api_domain

server {
    set $api_domain -;
    set $apiSunatPeruHost "api.sunat.peru";

    auth_jwt "SUNAT_API";
    auth_jwt_key_file jwk/api_secret.jwk;

    access_log /var/log/nginx/api.sunat.gob.pe.443.access.log api_mai
    error_log /var/log/nginx/api.sunat.gob.pe.443.error.log debug;

    listen 443 ssl;

    server_name api.sunat.gob.pe;
    status_zone api.sunat.gob.pe.443;

    ssl on;
    ssl_certificate      /etc/ssl/certs/api.example.com.crt;
    ssl_certificate_key  /etc/ssl/private/api.example.com.key;
    ssl_prefer_server_ciphers   on;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers
ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-A

    # API definitions, one per file
    include conf.d/api.sunat.gob.pe.443.conf.d/*.conf;

    # Error responses
    error_page 404 = @400;         # Invalid paths are treated as bad
    proxy_intercept_errors on;     # Do not send backend errors to th
    default_type application/json; # If no content-type then assume J
}
```

Donde

Línea	Descripción
9-10	implementa la autorización del cliente validando el JSON Web Token(JWT) provisto, utilizando las claves especifica

Seccion	**Dominio : Contribuyente (api_contribuyente.conf)**

Esta parte de la configuración primero define los URI válidos para la API y luego define una política común para manejar las s

```
# API definition
location /api/upstreamBackendAPI1 {
    set $upstream FirstAPI;
    rewrite ^ /_API last;
}

location /api/upstreamBackendAPI2 {
    set $upstream SecondAPI;
    rewrite ^ /_upstreamBackendAPI last;
}
# Policy section
location = /_API {
    internal;
    set $api_name "upstreamBackendAPI";
    # Policy configuration here (authentication, rate limiting, loggi
    proxy_pass http://$upstream$request_uri;
}
```

Línea	Descripción
2 y 7	definimos dos prefijos de ruta. En cada caso, la `$upstream` variable se configura con el nombre del `upstream` bl

Por ejemplo,"api.sunat.peru" quedaría definido de la siguiente forma:

```
location ~ "^\/v1\/contribuyente\/contribuyentes\/(\d{11})\/scores\/(
 set $api_domain "contribuyente";
 proxy_set_header Host $apiSunatPeruHost;
 proxy_set_header X-Real-IP $remote_addr;
 proxy_set_header X-Forwarded-Host $host;
 proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
 proxy_set_header X-Forwarded-Proto $scheme;
 rewrite "^\/v1\/contribuyente\/contribuyentes\/(\d{11}(|\/))$" /v1/c
 proxy_pass http://upstreamApiSunatPeru;
 proxy_http_version 1.1;
}
```

Donde

Línea	Descripción
1	se usa para definir cómo Nginx debe manejar las solicitudes de diferentes recursos y URI para el servidor principal
~	Activar Expresiones Regulares para la directiva **location**
8	Reescribiendo las solicitudes del cliente, re-direccionamiento hacia el servicio correspondiente.
9	Enviar la solicitud a un servidor especifico.

Seccion	api_secret.jwk

configuración de api_secret.

```
{
 "alg": "",
 "e": "",
 "kid": "",
 "kty": "",
 "n": "",
 "use": ""
 }
```

Donde

Línea	Descripción
2	El parámetro "alg" (algoritmo) identifica el algoritmo destinado a usar con la llave.
4	El parámetro "kid" (ID de clave) se usa para hacer coincidir una clave específica. Est se utiliza, por ejemplo, para elegir entre un conjunto de claves dentro de un conjunto durante la transferencia de clave.
5	El parámetro "kty" (tipo de clave) identifica el algoritmo criptográfico familia utilizada con la clave, como "RSA" o "EC".
7	El parámetro "use" se emplea para indicar si una clave pública se utiliza para cifrar datos o verificar la firma en los datos. Los valores definidos por esta especificación son: o "sig" (firma) o "enc" (cifrado)

Por ejemplo,"api.sunat.peru" quedaría definido de la siguiente forma:

```
{
 "alg": "RS256",
 "e": "AQAB",
 "kid": "sunatKeyRSA002",
 "kty": "RSA",
 "n": "mYrEMREykeBOZtpouorHg8W6tBfn3o_bR5mzCQDVTE51y547dmJXH7a108T--D
ceGRHCPcfnFPLAzxoCRwjB_CubhrUjZoXH4-0gN8skMGle3dl478vg-5MZXWjua7MdrTz
8S3U7S9Qq_tEcrHHCKVUV1czBpy8GSzi4lXLdQJ-7zK5uPFZiamI5Q3N4WmFZVtSbQ",
 "use": "sig"
 }
```