Laboratorio #3

Jeisson Andrés Vergara Vargas
Arquitectura de Aplicaciones Web
2021

El laboratorio debe ser desarrollado en parejas.

Actividades

i. Requisitos

Laboratorio #2 finalizado en su totalidad.

ii. Proxy Inverso

a. Nginx

Nginx es un servidor web y proxy inverso, es decir que su función es ser un intermediario en las peticiones de recursos. De esta manera, se pueden implementar medidas de control de acceso, registro del tráfico, restricción a determinados tipos de archivos, mejora de rendimiento, caché web, entre otras.

b. Implementación

1. Crear un directorio llamado **supermarket_proxy**.

2. En la raíz, crear el siguiente archivo **Dockerfile**:

```
FROM nginx

RUN apt-get update -qq && apt-get -y install apache2-utils
ENV NODE_ROOT /var/www/api-gateway
WORKDIR $NODE_ROOT
RUN mkdir log
```

```
COPY app.conf /tmp/app.nginx
RUN envsubst '$NODE_ROOT' < /tmp/app.nginx > /etc/nginx/conf.d/default.conf

EXPOSE 80

CMD [ "nginx", "-g", "daemon off;" ]
```

3. Crear un archivo llamado **app.conf**:

```
upstream api_gateway_node {
    server localhost:5000;
}

server {
    listen 80;
    proxy_buffers 64 16k;
    proxy_max_temp_file_size 1024m;
    proxy_connect_timeout 5s;
    proxy_send_timeout 10s;
    proxy_read_timeout 10s;

    location ~ /\. {
        deny all;
    }

    location ~* ^.+\.(rb|log)$ {
        deny all;
    }

    # serve static (compiled) assets directly if they exist (for node production)
    location ~ ^/(assets|images|javascripts|stylesheets|swfs|system)/ {
        try_files $uri @api_gateway_node;

        access_log off;
        gzip_static on; # to serve pre-gzipped version

        expires max;
        add_header Cache-Control public;

        # Some browsers still send conditional-GET requests if there's a
        # Last-Modified header or an ETag header even if they haven't
        # reached the expiry date sent in the Expires header.
        add_header Last-Modified "";
        add_header ETag "";
        break;
    }

    location / {
        try_files $uri $uri/ @api_gateway_node;
```

```
        }

        location @api_gateway_node {
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header Host $http_host;
            proxy_redirect off;
            proxy_pass http://api_gateway_node;
            access_log /var/www/api-gateway/log/nginx.access.log;
            error_log /var/www/api-gateway/log/nginx.error.log;
        }
    }
```

Nota: en la segunda línea, reemplazar *localhost* por la IP del contenedor del *API Gateway* desplegado (supermarket_api). Para obtener la IP del contenedor, ejecutar el siguiente comando:

> docker inspect CONTAINER_ID

c. Despliegue

1. Desplegar el componente:

```
docker build -t supermarket_proxy .

docker run -p 80:80 supermarket_proxy
```

Antes de realizar esta acción, asegurarse de que el API Gateway se encuentre desplegado correctamente.

2. Acceder a la ruta http://localhost/graphiql y verificar que el **Proxy Inverso** recibe la petición y la traslada al **API Gateway**.