

1- Objetivos de Aprendizaje

- Familiarizarse con conceptos de Microservicios

2- Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a la unidad N°: 2 (Libro Ingeniería de Software: Unidad 18)

3- Consignas a desarrollar en el trabajo práctico:

A continuación, se presentarán algunos conceptos generales de la arquitectura de microservicios y exploraremos un ejemplo completo y funcional de ejemplo de este tipo de sistemas.

1- Instanciación del sistema

- Clonar el repositorio
<https://github.com/microservices-demo/microservices-demo>

```
mkdir -p socks-demo
```

```
juan-pablo@juanpablo-OMEN-Laptop-15-ek0xxx:~/Escritorio/tpn4clon{{ $ mkdir -p socks-demo
```

creo directorio socks demo, entro dentro y clono ahi.

```
cd socks-demo
```

```
git clone https://github.com/microservices-demo/microservices-demo.git
```



Ejecutar lo siguiente

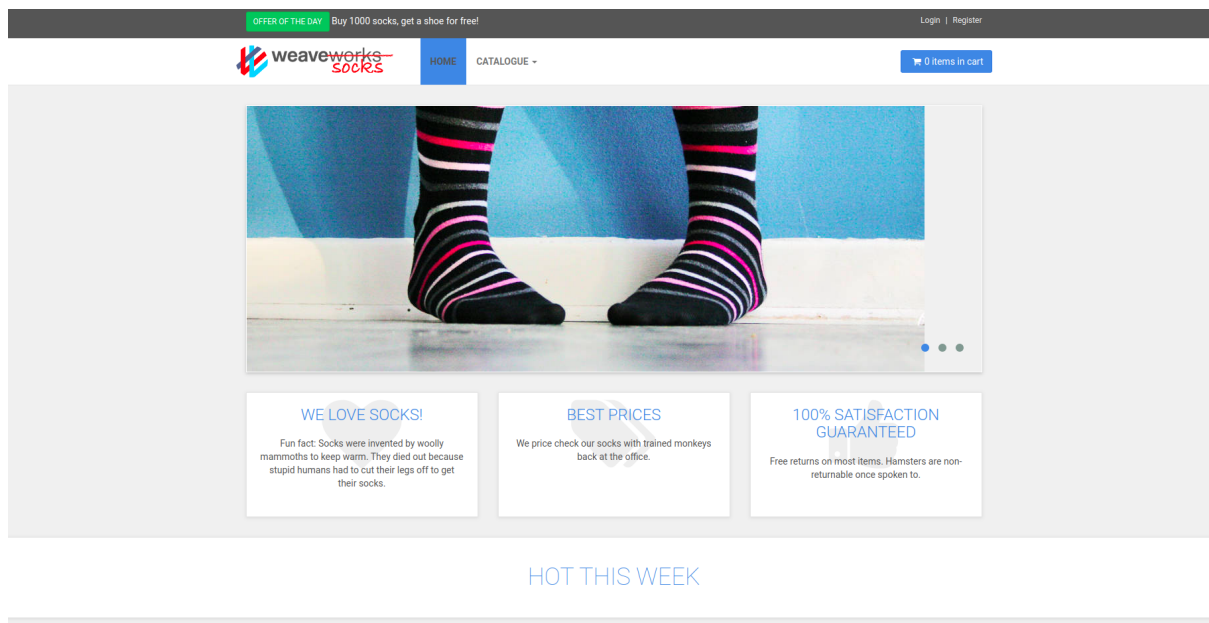
```
cd microservices-demo
```

levanto los contenedores del docker compose como vimos en el practico anterior, es decir el compose permite tener mas de un contenedor y levantar todo automático de una. "Docker Compose **ayuda a definir y compartir aplicaciones de varios contenedores.**"

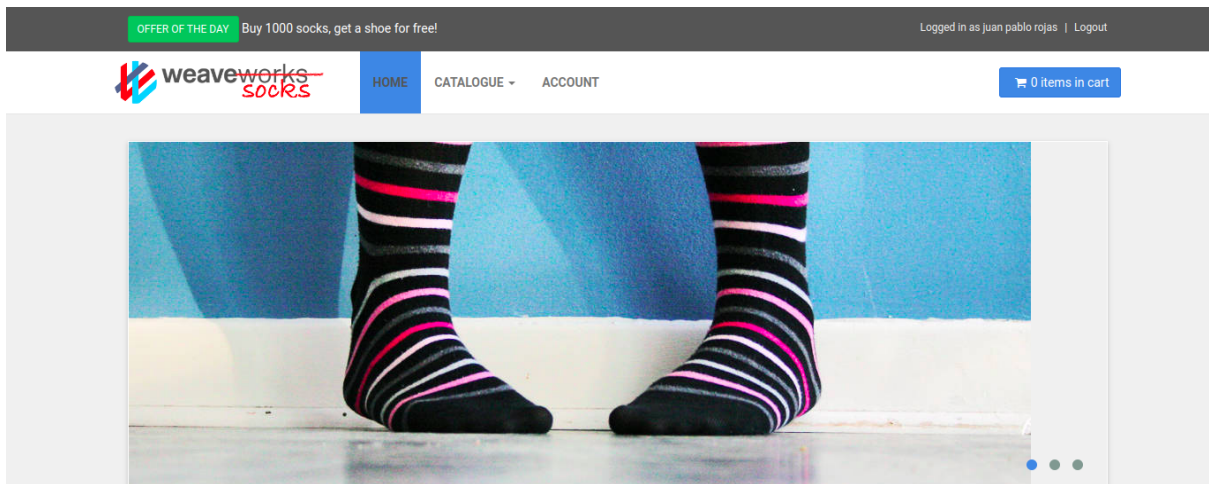
```
docker-compose -f deploy/docker-compose/docker-compose.yml up -d
```

```
juan-pablo@juanpablo-OMEN-Laptop-15-ek0xxx:~/Escritorio/tpn4clon{/socks-demo/microservices-demo$ docker-compose -f deploy/docker-compose/docker-compose.yml up -d
WARNING: The MYSQL_ROOT_PASSWORD variable is not set. Defaulting to a blank string.
Creating network "docker-compose_default" with the default driver
Pulling front-end (weaveworksdemos/front-end:0.3.12)...
0.3.12: Pulling from weaveworksdemos/front-end
709515475419: Pull complete
8a7a0a7b6b80: Pull complete
79dadb976eeb: Pull complete
f48430971a7c: Pull complete
59ee1f19577a: Pull complete
89bb0a032ada: Pull complete
01d99281a40a: Pull complete
07a3ae088a4f: Pull complete
a9efb8659a27: Pull complete
Digest: sha256:26a2d9b6b291dee2dca32fca3f5bfff6c2fa07bb5954359afcbc8001cc70eac71
Status: Downloaded newer image for weaveworksdemos/front-end:0.3.12
Pulling edge-router (weaveworksdemos/edge-router:0.1.1)...
0.1.1: Pulling from weaveworksdemos/edge-router
b7f33cc0b48e: Already exists
```

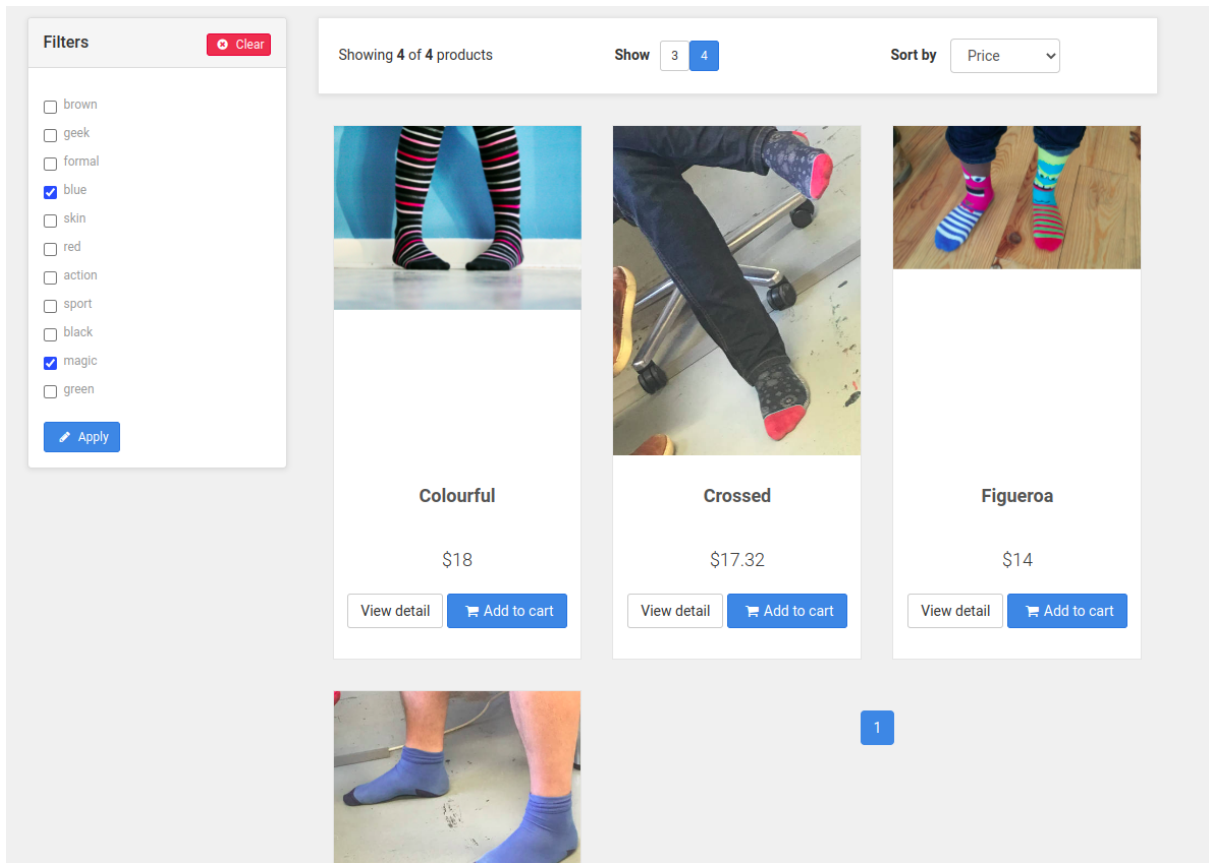
- una vez termina, tuve que liberar el puerto 80 porque tenia apache, lo que hice para matar ese proceso en ese puerto fue: `sudo kill -15 PID` y para ver el pid use el `netstat -tupln`.



Generar un usuario



Realizar búsquedas por tipo de media, color, etc



Peticion GET

☐ topbar.html

☐ navbar.html

☐ footer.html

☐ size?tags=blue

☐ tags

☐ catalogue?pa...

☐ HsXT9Oe1Wo...

☐ cart

▼ [{id: "3395a43e-2d88-40de-b95f-e00e1502085b", name: "Colourful",...},...]

▶ 0: {id: "3395a43e-2d88-40de-b95f-e00e1502085b", name: "Colourful",...}

▶ 1: {id: "808a2de1-1aaa-4c25-a9b9-6612e8f29a38", name: "Crossed",...}

▶ 2: {id: "819e1fbf-8b7e-4f6d-811f-693534916a8b", name: "Figueroa",...}

▶ 3: {id: "a0a4f044-b040-410d-8ead-4de0446aec7e", name: "Nerd leg",...}

Hacer una compra

Card Number


3453453454

Expires

06/36

CCV

466

 Update

Home > My orders

Customer section

My orders

My orders

Your orders in one place.

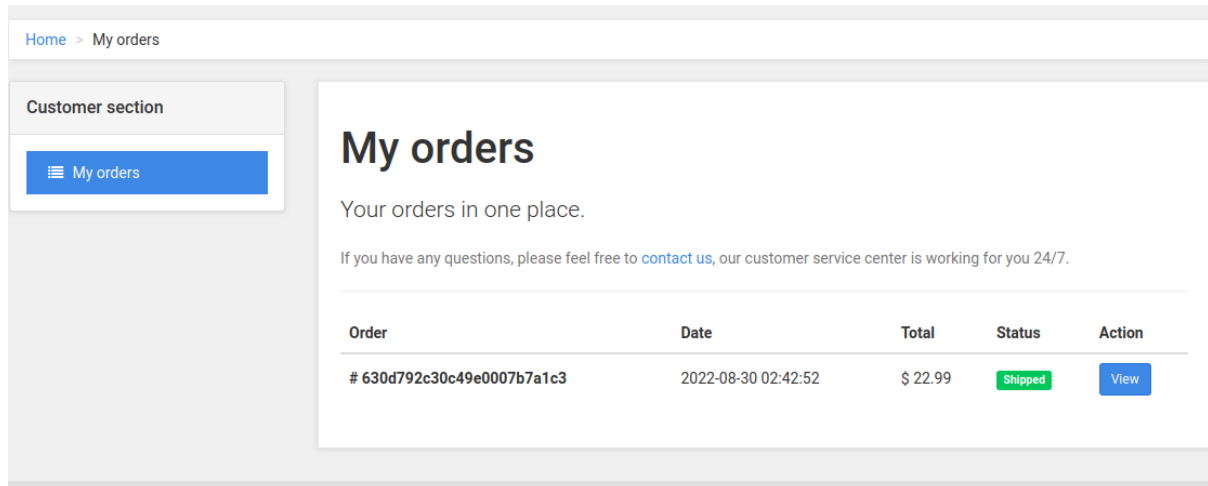
If you have any questions, please feel free to [contact us](#), our customer service center is working for you 24/7.

Order	Date	Total	Status	Action
# 630d792c30c49e0007b7a1c3	2022-08-30 02:42:52	\$ 22.99	Shipped	View

2 Investigacion de los componentes:

1 Describa los contenedores creados, indicando cuales son puntos de ingreso del sistema

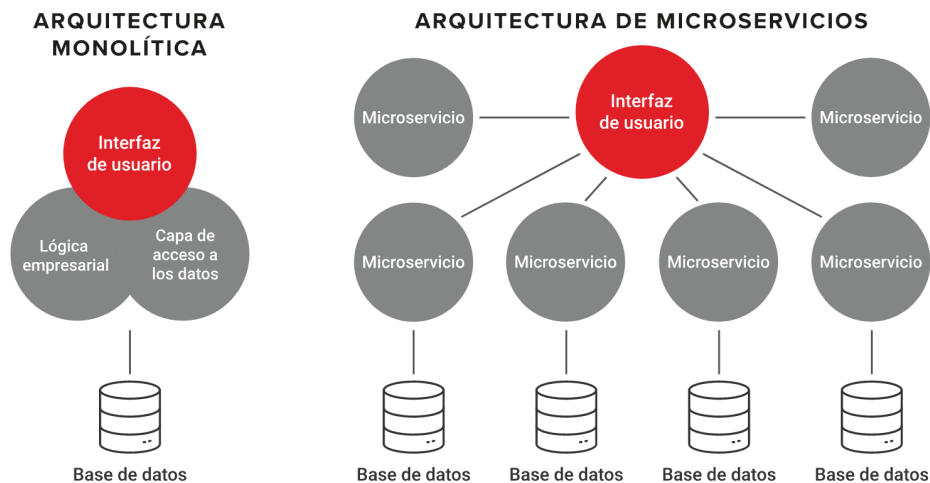
Ejecuto `docker -ps` y tambien desde el `complete-demo.yaml`, para ver los contenedores levantados:



- **Carts:** Servicio de carrito de compras
- **queue-master:** consume los mensajes de la cola de rabbitmq
- **user:** Servicio de usuarios con sus datos
- **catalogue:** Servicio de catalogo
- **shipping:** Servicio de compra
- **front-end:** Interfaz, puerta de interacción con la API, se invocan los diferentes servicios apartir de los protocolos http.
- **user-db:** Base de datos de los usuarios
- **edge-router:** es el punto de acceso al sistema??
- **payment:** Servicio de pago
- **catalogue-db:** Base de datos que contiene los productos
- **orders:** Servicio o logica de negocio de las ordenes
- **orders-db:** Base de datos de las ordenes
- **rabbitmq:** RabbitMQ es un software de negociación de mensajes de código abierto que funciona como un middleware de mensajería. Implementa el estándar Advanced Message Queuing Protocol (middleware de mensajería).

investigando un poco mas sobre estas colas de mensajes, podemos ver que se usan en arquitecturas de microservicios y servidor, es asíncrona de servicio a servicio, estos msg se pushean a la cola y despues de procesarse se eliminan.

imagen representativa de una arquitectura de micro servicios



2 clono los repositorios del front, user, edge-router

3-¿Por qué cree usted que se está utilizando repositorios separados para el código y/o la configuración del sistema? Explique puntos a favor y en contra. Están separados porque cada microservicio contiene su propia funcionalidad y estructura de forma independiente, luego para vincular estos microservicios apartir de las configuraciones del sistema. modularidad

Porque permite contener diferentes microservicios de forma independiente, es decir evolucionan sobre el medio de cada microservicio de forma propia, permite que no todas los microservicios se desarrollen de la misa manera o con las mismas tecnologías por ejemplo, o con los mismos flujos de trabajo. estos microservicios se pueden utilizar para construir otras aplicaciones por ejemplo el servicio de autenticacion, lo podemos replicar en varias aplicaciones, es modular por lo que hablabamos de los modulos o de que cada uno funciona de forma independiente, podemos utilizar los contenedores como docker-compose para desarrollar y desplegar de forma rapida la aplicacion

-Algunas de las desventajas pueden ser la complejidad de configurar los vinculos o gestionar un numero elevados de microservicios

4¿Cuál contenedor hace las veces de API Gateway?

El front end, ya que es el que interactua con los servicios desde la interfaz. **docker_compose_front-end_1**

5 Cuando ejecuto este comando:

```
curl http://localhost/customers
```

```
juan-pablo@juanpablo-OMEN-Laptop-15-ek0xxx:~/Escritorio/cpn4clon{{/socks-demo$ curl http://localhost/customers
{"_embedded":{"customer":[{"firstName":"Eve","lastName":"Berger","username":"Eve_Berger","id":"57a98d98e4b00679b4a830af",
"_links":{"addresses":{"href":"http://user/customers/57a98d98e4b00679b4a830af/addresses"},"cards":{"href":"http://user/
customers/57a98d98e4b00679b4a830af/cards"},"customer":{"href":"http://user/customers/57a98d98e4b00679b4a830af"},"self":{"
href":"http://user/customers/57a98d98e4b00679b4a830af"}}}],{"firstName":"User","lastName":"Name","username":"user","id":
"57a98d98e4b00679b4a830b2","_links":{"addresses":{"href":"http://user/customers/57a98d98e4b00679b4a830b2/addresses"},"ca
rds":{"href":"http://user/customers/57a98d98e4b00679b4a830b2/cards"},"customer":{"href":"http://user/customers/57a98d98e
4b00679b4a830b2"},"self":{"href":"http://user/customers/57a98d98e4b00679b4a830b2"}}}],{"firstName":"User1","lastName":"Na
me1","username":"user1","id":"57a98d98e4b00679b4a830b5","_links":{"addresses":{"href":"http://user/customers/57a98d98e4b
00679b4a830b5/addresses"},"cards":{"href":"http://user/customers/57a98d98e4b00679b4a830b5/cards"},"customer":{"href":"ht
tp://user/customers/57a98d98e4b00679b4a830b5"},"self":{"href":"http://user/customers/57a98d98e4b00679b4a830b5"}}}],{"firs
tName":"juan pablo","lastName":"rojas","username":"juanpablorojas7","id":"630d7875ee11cb000147edc7","_links":{"addresses
":{"href":"http://user/customers/630d7875ee11cb000147edc7/addresses"},"cards":{"href":"http://user/customers/630d7875ee1
1cb000147edc7/cards"},"customer":{"href":"http://user/customers/630d7875ee11cb000147edc7"},"self":{"href":"http://user/c
ustomers/630d7875ee11cb000147edc7"}}}}]}
```

podemos verlo de mejor forma al json del get a esa url desde las siguientes imagenes.

```
{
  "_embedded": {
    "customer": [
      {
        "firstName": "Eve",
        "lastName": "Berger",
        "username": "Eve_Berger",
        "id": "57a98d98e4b00679b4a830af",
        "_links": {
          "addresses": {
            "href": "http://user/customers/57a98d98e4b00679b4a830af/addresses"
          },
          "cards": {
            "href": "http://user/customers/57a98d98e4b00679b4a830af/cards"
          },
          "customer": {
            "href": "http://user/customers/57a98d98e4b00679b4a830af"
          },
          "self": {
            "href": "http://user/customers/57a98d98e4b00679b4a830af"
          }
        }
      },
      {
        "firstName": "User",
        "lastName": "Name",
        "username": "user",
        "id": "57a98d98e4b00679b4a830b2",
        "_links": {
          "addresses": {
            "href": "http://user/customers/57a98d98e4b00679b4a830b2/addresses"
          },
          "cards": {
            "href": "http://user/customers/57a98d98e4b00679b4a830b2/cards"
          },
          "customer": {
            "href": "http://user/customers/57a98d98e4b00679b4a830b2"
          },
          "self": {
            "href": "http://user/customers/57a98d98e4b00679b4a830b2"
          }
        }
      },
      {
        "firstName": "User1",
        "lastName": "Name1",
        "username": "user1",
        "id": "57a98d98e4b00679b4a830b5",
        "_links": {
          "addresses": {
            "href": "http://user/customers/57a98d98e4b00679b4a830b5/addresses"
          },
          "cards": {
            "href": "http://user/customers/57a98d98e4b00679b4a830b5/cards"
          },
          "customer": {
            "href": "http://user/customers/57a98d98e4b00679b4a830b5"
          },
          "self": {
            "href": "http://user/customers/57a98d98e4b00679b4a830b5"
          }
        }
      },
      {
        "firstName": "juan pablo",
        "lastName": "rojas",
        "username": "juanpablorojas7",
        "id": "630d7875ee11cb000147edc7",
        "_links": {
          "addresses": {
            "href": "http://user/customers/630d7875ee11cb000147edc7/addresses"
          },
          "cards": {
            "href": "http://user/customers/630d7875ee11cb000147edc7/cards"
          },
          "customer": {
            "href": "http://user/customers/630d7875ee11cb000147edc7"
          },
          "self": {
            "href": "http://user/customers/630d7875ee11cb000147edc7"
          }
        }
      }
    ]
  }
}
```

► GET http://localhost/customers

Estado	200 OK ?
Versión	HTTP/1.1
Transferido	1,70 KB (tamaño 1,56 KB)
Prioridad de solicitud	Highest

▼ Encabezados de respuesta (142 B)

Bruto

- Content-Length: 1601
- Content-Type: text/plain; charset=utf-8
- Date: Wed, 31 Aug 2022 17:30:57 GMT
- X-Powered-By: Express

6.¿Cuál de todos los servicios está procesando la operación?

5ba4bbec	weaveworksdemos/user:0.4.4	"/user -port=80"	39 hours ago	U
				docker-compose_user_1

```
curl http://localhost/catalogue
```

vemos como nos devuelve el servicio la url, que el que porcesa este
 microservicio seria el contenedor del catalogo que es

```

docker-compose catalogue 1
  
```



```
curl http://localhost/tags
```

```
// 20220831143911
// http://localhost/tags

{
  "tags": [
    "brown",
    "geek",
    "formal",
    "blue",
    "skin",
    "red",
    "action",
    "sport",
    "black",
    "magic",
    "green"
  ],
  "err": null
}
```

► GET http://localhost/tags

Estado	200 OK ?
Versión	HTTP/1.1
Transferido	248 B (tamaño 107 B)
Prioridad de solicitud	Highest

También lo procesa el catálogo porque si vemos el catálogo, tiene sus atributos y tiene una `List<Tags>` tag, como clave foránea sobre la tabla de tags, donde en esa lista de tipo tags se le pasarían los datos que va a contener esa tupla del producto del catálogo, cuando yo haga un post tengo atributos de la clase por ejemplo catálogo y tengo un atributo tag a llenar, que le paso a un array el contenido de los mismos, por eso lo procesa también catalogue.

8¿Cómo persisten los datos los servicios?

tenemos 3 bases de datos, una para catalogo (contiene los productos), otra para los usuarios y otra para las ordenes.

si se frenan los servicios o se dan de baja, se pierden, excepto que se le pase un volumen.

¿Cuál es el componente encargado del procesamiento de la cola de mensajes? como nombramos arriba, el que se encarga es **queue-master**: consume los mensajes de la cola de rabbitmq y esto es **rabbitmq**: RabbitMQ es un software de negociación de mensajes de código abierto que funciona como un middleware de mensajería. Implementa el estándar Advanced Message Queuing Protocol (middleware de mensajería).

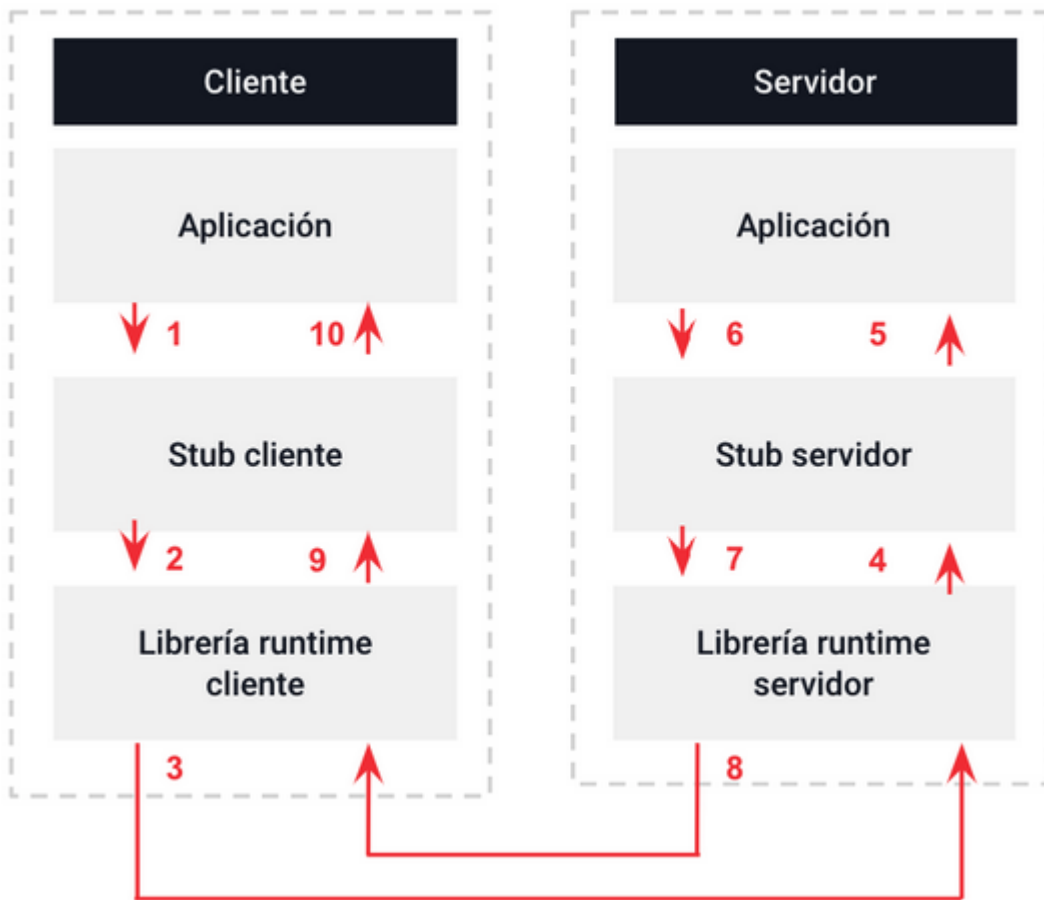
¿Qué tipo de interfaz utilizan estos microservicios para comunicarse?

Hay varios tipos de interfaz que se utiliza para la comunicacion del vinculo de todos estos microservicios, yo personalmente conozco RESTful que apartir de una url, devuelve un json o xml, tambien en el teorico vimos algunos como gRPC

Una de las opciones que está adquiriendo fuerza es el uso de [gRPC](#). **gRPC es una implementación de llamadas a procedimiento remoto (RPC)** diseñado originalmente por Google, de ahí su nombre: g (Google) + RPC.

Actualmente es un componente open source y su distribución la realiza la [Cloud Native Computing Foundation](#) y para cada nueva versión cambia el significado de la "g", hasta el punto en que incluso hicieron un [README](#) para enumerar todos sus significados.

Se emplea en comunicaciones cliente-servidor distribuidas por su eficiencia gracias a la ingeniería de procesos basada en RPC. Este mecanismo permite la comunicación de una forma tan sencilla como si se tratara de una comunicación local entre procesos de una misma máquina



stub es código que no está implementado en una parte pero cuando se comunica con la otra, en esa sí está implementado y no da error porque es del tipo stub