

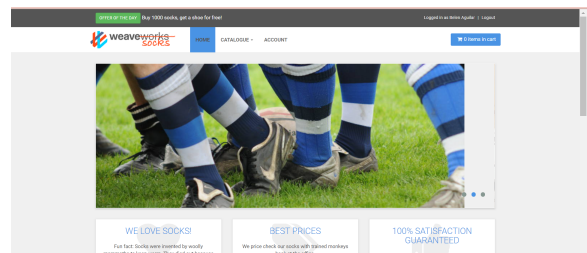
TP4 - Microservicios

```
MINGW64~/Users/BELU/go/src/github.com/belenaguiarv/IngenieriaDeSoftware3/TP4...
BELU@belenaguiarv MINGW64 ~/go/src/github.com/belenaguiarv/IngenieriaDeSoftware3/TP4...
P4 - Microservicios (main)
$ cd socks-demo

BELU@belenaguiarv MINGW64 ~/go/src/github.com/belenaguiarv/IngenieriaDeSoftware3/TP4...
P4 - Microservicios/socks-demo (main)
$ git clone https://github.com/belenaguiarv/microservices-demo.git
Cloning into 'microservices-demo'...
remote: Enumerating objects: 10197, done.
remote: Total 10197 (delta 0), reused 0 (delta 0), pack-reused 10197
Receiving objects: 100% (10197/10197), 52.95 MiB | 1.13 MiB/s, done.
Resolving deltas: 100% (6208/6208), done.

BELU@belenaguiarv MINGW64 ~/go/src/github.com/belenaguiarv/IngenieriaDeSoftware3/TP4...
P4 - Microservicios/socks-demo (main)
$ cd microservices-demo



BELU@belenaguiarv MINGW64 ~/go/src/github.com/belenaguiarv/IngenieriaDeSoftware3/TP4...
P4 - Microservicios/socks-demo/microservices-demo (master)
$ 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 docker-compose_shipping_1 ... done
Creating docker-compose_catalogue-db_1 ... done
Creating docker-compose_user-db_1 ... done
Creating docker-compose_queue-master_1 ... done
Creating docker-compose_carts_1 ... done
Creating docker-compose_orders_1 ... done
Creating docker-compose_edge-router_1 ... done
Creating docker-compose_payment_1 ... done
Creating docker-compose_orders-db_1 ... done
Creating docker-compose_front-end_1 ... done
Creating docker-compose_carts-db_1 ... done
Creating docker-compose_rabbitmq_1 ... done
Creating docker-compose_user-sim_1 ... done
Creating docker-compose_catalogue_1 ... done
Creating docker-compose_user_1 ... done
```



- Generar un usuario

Logged in as Belen Aguilar | Logout

- Realizar búsquedas por tipo de media, color, etc.

Shopping cart				
Product	Quantity	Unit price	Discount	Total
 Figuras	1	\$14.00	\$0.00	\$14.00
 Colorful	1	\$18.00	\$0.00	\$18.00
Total				\$32.00

- Hacer una compra - poner datos falsos de tarjeta de crédito ;)

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
# 64e798f0b9d8210007924dcd	2023-08-24 17:52:48	\$ 36.99	Shipped	View
# 64e798f1b9d8210007924dce	2023-08-24 17:52:49	\$ 4.99	Shipped	View

2- Investigación de los componentes

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

Se crearon 15 contenedores:

1. front-end

2. edge-router
3. catalogue
4. catalogue-db
5. carts
6. carts-db
7. orders
8. orders-db
9. shipping
10. queue-master
11. rabbitmq
12. payment
13. user
14. user-db
15. user-sim

2. Clonar algunos de los repositorios con el código de las aplicaciones

```

MINGW64/c/Users/BELU/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP4 ...
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP4 - Microservicios/socks-demo (main)
$ cd socks-demo
bash: cd: socks-demo: No such file or directory

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP4 - Microservicios/socks-demo (main)
$ git clone https://github.com/microservices-demo/front-end.git
Cloning into 'front-end'...
remote: Enumerating objects: 1236, done.
remote: Total 1236 (delta 0), reused 0 (delta 0), pack-reused 1236
Receiving objects: 47.40 MiB | 1.12 MiB/s
Receiving objects: 100% (1236/1236), 47.90 MiB | 1.09 MiB/s, done.
Resolving deltas: 100% (687/687), done.

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP4 - Microservicios/socks-demo (main)
$ git clone https://github.com/microservices-demo/user.git
Cloning into 'user'...
remote: Enumerating objects: 1063, done.
remote: Total 1063 (delta 0), reused 0 (delta 0), pack-reused 1063
Receiving objects: 100% (1063/1063), 172.83 KiB | 177.00 KiB/s, done.
Resolving deltas: 100% (601/601), done.

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/TP4 - Microservicios/socks-demo (main)
$ git clone https://github.com/microservices-demo/edge-router.git
Cloning into 'edge-router'...
remote: Enumerating objects: 50, done.
remote: Total 50 (delta 0), reused 0 (delta 0), pack-reused 50
Receiving objects: 54% (27/50)
Receiving objects: 100% (50/50), 14.51 KiB | 412.00 KiB/s, done.
Resolving deltas: 100% (12/12), done.

```

1. ¿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.

Ventajas

- Separación clara en aspectos de desarrollo e infraestructura que facilita la gestión y mantenimiento.

Desventajas

- Puede aumentar la complejidad por lo que no es ideal para equipos poco experimentados.

- Al separar repositorios permitimos que cada componente (código y configuración) tenga su propio flujo de versionamiento.
- Se necesita coordinación y que los cambios sean compatibles para que no causen problemas en el sistema completo.
- Diferentes equipos pueden trabajar en el código en forma paralela, se mejora la colaboración y acelera el desarrollo ya que cada uno se puede centrar en su area de aexperiencia.

1. ¿Cuál contenedor hace las veces de API Gateway?

El **edge-router**, el repositorio que corresponde a este contenedor tiene un archivo *traefik.toml*

Traefik es un Edge router, es decir, “la puerta a la plataforma”. Se encarga de interceptar cada petición que se realiza y enrutarla al servicio correcto. Traefik sabe la lógica y las reglas que determinan que el servicio es el encargado de gestionar cada petición.

2. Cuando ejecuto este comando:

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

```
curl http://localhost/customers
{
  "embedded": {
    "customers": [
      {
        "firstName": "Eve",
        "lastName": "Bergan",
        "username": "Eve_Bergan",
        "id": "57a9d98e4b0679b4a838af",
        "links": {
          "addresses": {
            "href": "http://user/customers/57a9d98e4b0679b4a838af/addresses"
          },
          "cards": {
            "href": "http://user/customers/57a9d98e4b0679b4a838af/cards"
          },
          "customer": {
            "href": "http://user/customers/57a9d98e4b0679b4a838af"
          }
        }
      },
      {
        "firstName": "User",
        "lastName": "Name",
        "username": "user",
        "id": "57a9d98e4b0679b4a838b2",
        "links": {
          "addresses": {
            "href": "http://user/customers/57a9d98e4b0679b4a838b2/addresses"
          },
          "cards": {
            "href": "http://user/customers/57a9d98e4b0679b4a838b2/cards"
          },
          "customer": {
            "href": "http://user/customers/57a9d98e4b0679b4a838b2"
          }
        }
      },
      {
        "firstName": "User",
        "lastName": "Name1",
        "username": "user",
        "id": "57a9d98e4b0679b4a838b5",
        "links": {
          "addresses": {
            "href": "http://user/customers/57a9d98e4b0679b4a838b5/addresses"
          },
          "cards": {
            "href": "http://user/customers/57a9d98e4b0679b4a838b5/cards"
          },
          "customer": {
            "href": "http://user/customers/57a9d98e4b0679b4a838b5"
          }
        }
      },
      {
        "firstName": "Belén",
        "lastName": "Aguilar",
        "username": "belenaguilar",
        "id": "64e79807ee11c0801ff7128",
        "links": {
          "addresses": {
            "href": "http://user/customers/64e79807ee11c0801ff7128/addresses"
          },
          "cards": {
            "href": "http://user/customers/64e79807ee11c0801ff7128/cards"
          },
          "customer": {
            "href": "http://user/customers/64e79807ee11c0801ff7128"
          }
        }
      }
    ]
  }
}
```

Nos devuelve la respuesta con todo el contenido

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

El servicio que está procesando la operación es el de **user**

2. ¿Y para los siguientes casos?

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

El servicio que está procesando la operación es el de **catalogue**

```

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware
3/TP4 - Microservicios/socks-demo/microservices-demo (master)
$ curl http://localhost/catalogue
[{"id":"03fef6ac-1896-4ce8-bd69-b798f85c6e0b","name":"Holy","description":"Socks
fit for a Messiah. You too can experience walking in water with these special edi
tion beauties. Each hole is lovingly progged to leave smooth edges. The only soc
k approved by a higher power.", "imageUrl":["/catalogue/images/holy_1.jpeg","/cata
logue/images/holy_2.jpeg"], "price":99.99, "count":1, "tag":["action", "magic"]}, {"id
":"3395a43e-2d88-40de-b95f-e00e1502085b","name":"Colourful","description":"proide
nt occaecat irure et excepteur labore minim nisi amet irure", "imageUrl":["/catalo
gue/images/colourful_socks.jpg", "/catalogue/images/colourful_socks.jpg"], "price":
18, "count":438, "tag":["brown", "blue"]}, {"id":"510a0d7e-8e83-4193-b483-e27e09ddc34
d","name":"SuperSport XL","description":"Ready for action. Engineers: be ready to
smash that next bug! Be ready, with these super-action-sport-masterpieces. This
particular engineer was chased away from the office with a stick.", "imageUrl":["/
catalogue/images/puma_1.jpeg", "/catalogue/images/puma_2.jpeg"], "price":15, "count"
:820, "tag":["sport", "formal", "black"]}, {"id":"808a2de1-1aaa-4c25-a9b9-6612e8f29a3
8","name":"Crossed","description":"A mature sock, crossed, with an air of nonchal
ance.", "imageUrl":["/catalogue/images/cross_1.jpeg", "/catalogue/images/cross_2.jp
eg"], "price":17.32, "count":738, "tag":["blue", "action", "red", "formal"]}, {"id":"819
e1fbf-8b7e-4f6d-811f-693534916a8b","name":"Figueroa","description":"enim officia
aliqua excepteur esse deserunt quis aliquip nostrud anim", "imageUrl":["/catalogue
/images/WAT.jpg", "/catalogue/images/WAT2.jpg"], "price":14, "count":808, "tag":["gre
en", "formal", "blue"]}, {"id":"837ab141-399e-4c1f-9abc-bace40296bac","name":"Cat so
cks","description":"consequat amet cupidatat minim laborum tempor elit ex consequ
at in", "imageUrl":["/catalogue/images/catsocks.jpg", "/catalogue/images/catsocks2.
jpg"], "price":15, "count":175, "tag":["brown", "formal", "green"]}, {"id":"a0a4f044-b0
40-410d-8ead-4de0446aec7e","name":"Nerd leg","description":"For all those leg lov
ers out there. A perfect example of a swivel chair trained calf. Meticulously tra
ined on a diet of sitting and Pina Colodas. Phwarr...", "imageUrl":["/catalogue/im
ages/bit_of_leg_1.jpeg", "/catalogue/images/bit_of_leg_2.jpeg"], "price":7.99, "count"
:115, "tag":["blue", "skin"]}, {"id":"d3588630-ad8e-49df-bbd7-3167f7efb246","name"
:"YouTube sock","description":"We were not paid to sell this sock. It's just a bi
t geeky.", "imageUrl":["/catalogue/images/youtube_1.jpeg", "/catalogue/images/youtu
be_2.jpeg"], "price":10.99, "count":801, "tag":["formal", "geek"]}, {"id":"zzz4f044-b0
40-410d-8ead-4de0446aec7e","name":"Classic","description":"Keep it simple.", "imag
eUrl":["/catalogue/images/classic.jpg", "/catalogue/images/classic2.jpg"], "price":
12, "count":127, "tag":["brown", "green"]}

```

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

```

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware
3/TP4 - Microservicios/socks-demo/microservices-demo (master)
$ curl http://localhost/tags
{"tags":["brown","geek","formal","blue","skin","red","action","sport","black","ma
gic","green"],"err":null}

```

El servicio que está procesando la operación es el de **front-end**

1. ¿Como persisten los datos los servicios?

Algunos servicios utilizan DB para persistir los datos. Por ejemplo: **catalogue-db**, **carts-db**, **orders-db**, **user-db**, **user-sim** utilizan **MongoDB** o **MySQL** para almacenar y gestionar los datos relacionados con el catálogo de productos, carritos de compras, pedidos y usuarios. Estas DB proporcionan la capacidad de almacenar datos de manera duradera entre las operaciones del sistema.

2. ¿Cuál es el componente encargado del procesamiento de la cola de mensajes?

En la configuración, el contenedor **rabbitmq** está destinado a actuar como un sistema de cola de mensajes. RabbitMQ es un software de mensajería que permite la comunicación asíncrona. Los microservicios pueden enviar y recibir mensajes a través de la cola para manejar tareas en segundo plano, comunicación entre servicios o procesamiento de eventos.

3. ¿Qué tipo de interfaz utilizan estos microservicios para comunicarse?

En esta configuración, los microservicios utilizan solicitudes HTTP para enviar datos y recibir respuestas de otros microservicios.