

# Trabajo Práctico N°4

## Ejercicio 1

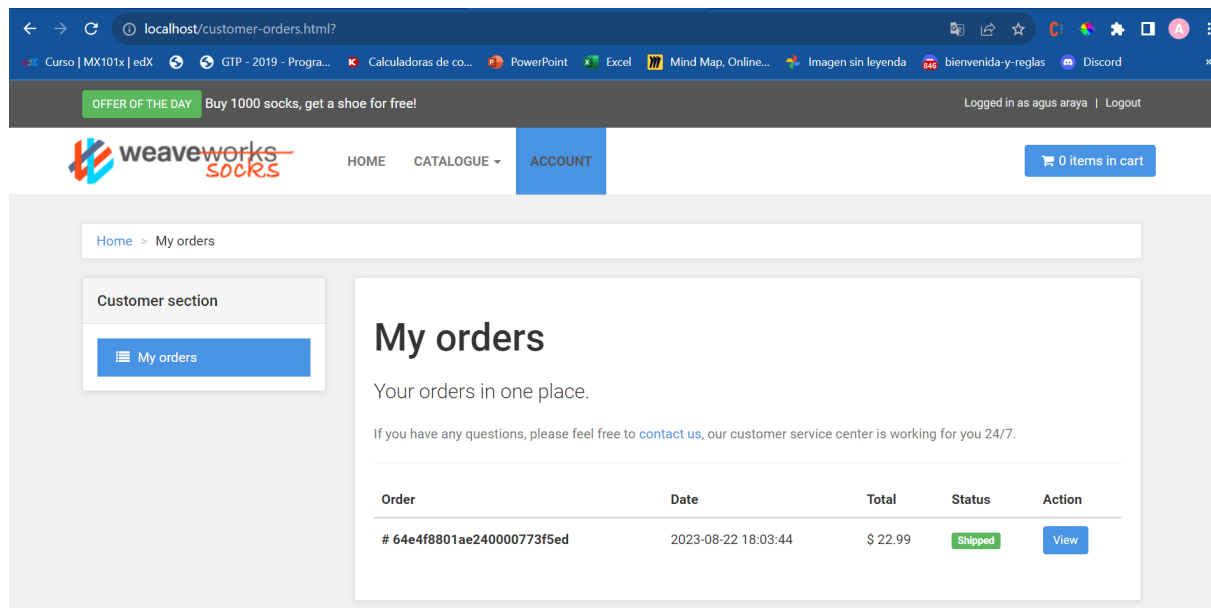
```
Usuario@DESKTOP-8G1HL0R MINGW64 ~/OneDrive/Escritorio/2023/Ing SW 3
$ mkdir -p socks-demo

Usuario@DESKTOP-8G1HL0R MINGW64 ~/OneDrive/Escritorio/2023/Ing SW 3
$ cd socks-demo

Usuario@DESKTOP-8G1HL0R MINGW64 ~/OneDrive/Escritorio/2023/Ing SW 3/socks-demo
$ git clone https://github.com/microservices-demo/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 | 1.10 MiB/s, done.
Resolving deltas: 100% (6208/6208), done.

Usuario@DESKTOP-8G1HL0R MINGW64 ~/OneDrive/Escritorio/2023/Ing SW 3/socks-demo
$ cd microservices-demo

Usuario@DESKTOP-8G1HL0R MINGW64 ~/OneDrive/Escritorio/2023/Ing SW 3/socks-demo/microservices-demo (master)
$ docker-compose -f deploy/docker-compose/docker-compose.yml up -d
time="2023-08-22T14:37:09-03:00" level=warning msg="The \"MYSQL_ROOT_PASSWORD\" variable is not set. Defaulting to a blank string."
[+] Running 23/20
 - user-sim 11 layers [██████████] 0B/0B Pulling
 - rabbitmq 14 layers [██████████] 29.42MB/79.37MB Pulling
 - catalogue 4 layers [██████] 0B/0B Pulled
 - orders-db Pulling
 - front-end 9 layers [██████████] 0B/0B Pulled
 - shipping 9 layers [██████████] 0B/0B Pulling
[+] Running 117/20 layers [██████] 0B/0B Pulling
```






















## Ejercicio 2

Estos son los contenedores creados al ejecutar docker-compose.yml

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

<div>   <div> <b>docker-compose</b>  C:\Users\Usuario\OneDrive\Escritorio\2023 </div> </div>				
	<b>docker-compose-rab...</b> <a href="#">rabbitmq:3.6.8</a> Running			
	<b>docker-compose-fro...</b> <a href="#">weaveworksdemos/fro</a> Running			
	<b>docker-compose-ord...</b> <a href="#">mongo:3.4</a> Running			
	<b>docker-compose-us...</b> <a href="#">weaveworksdemos/use</a> Running			
	<b>docker-compose-car...</b> <a href="#">mongo:3.4</a> Running			
	<b>docker-compose-us...</b> <a href="#">weaveworksdemos/use</a> Running			
	<b>docker-compose-cat...</b> <a href="#">weaveworksdemos/cat</a>			

```

2023-08-22 15:03:43 2023-08-22 18:03:43.013 INFO [orders,33aa4d352f8b4090,e41535d0ea35991f,false] 7 --- [p-nio-80-exec-2] w.w.s.o.controllers.OrdersControlle
r : Sending payment request: PaymentRequest{address=Address{id=null, number='11', street='calle 1', city='Capital', country='Argentina', postcode='50000'}
, card=Card{id=null, longNum='1234567890', expires='12/03', ccv='333'}, customer=Customer{id=null, firstName='agus', lastName='araya', username='aaraya0', add
resses=[], cards=[]}}
2023-08-22 15:03:43 2023-08-22 18:03:43.064 INFO [orders,33aa4d352f8b4090,e41535d0ea35991f,false] 7 --- [p-nio-80-exec-2] w.w.s.o.controllers.OrdersControlle
r : Received payment response: PaymentResponse{authorised=true, message=Payment authorised}

```

```

usuario@DESKTOP-8G1HL0R MINGW64 ~/OneDrive/Escritorio/2023/Ing SW 3/socks-demo/microservices-demo/deploy/docker-compose (master)
$ docker-compose ps
time="2023-08-22T15:31:40-03:00" level=warning msg="The \"MYSQL_ROOT_PASSWORD\" variable is not set. Defaulting to a blank string."
NAME                                IMAGE                                COMMAND                                SERVICE    CREATED         STATUS         PORTS
docker-compose-carts-1              weaveworksdemos/carts:0.4.8        "/usr/local/bin/java..."          carts       39 minutes ago  Up 39 minutes  27017/tcp
docker-compose-carts-db-1           mongo:3.4                           "docker-entrypoint.s..."          carts-db    39 minutes ago  Up 39 minutes  27017/tcp
docker-compose-catalogue-1          weaveworksdemos/catalogue:0.3.5    "/app -port=80"                    catalogue   39 minutes ago  Up 39 minutes  3306/tcp
docker-compose-catalogue-db-1       weaveworksdemos/catalogue-db:0.3.0 "docker-entrypoint.s..."          catalogue-db 39 minutes ago  Up 39 minutes  0.0.0.0:80->80/tcp
docker-compose-edge-router-1        weaveworksdemos/edge-router:0.1.1  "traefik"                           edge-router 39 minutes ago  Up 39 minutes  0.0.0.0:80->80/tcp
docker-compose-front-end-1          weaveworksdemos/front-end:0.3.12   "/usr/local/bin/npm ..."        front-end   39 minutes ago  Up 39 minutes  8079/tcp
docker-compose-orders-1             weaveworksdemos/orders:0.4.7       "/usr/local/bin/java..."          orders      39 minutes ago  Up 39 minutes  27017/tcp
docker-compose-orders-db-1          mongo:3.4                           "docker-entrypoint.s..."          orders-db   39 minutes ago  Up 39 minutes  27017/tcp
docker-compose-payment-1            weaveworksdemos/payment:0.4.3      "/app -port=80"                    payment     39 minutes ago  Up 39 minutes  80/tcp
docker-compose-queue-master-1       weaveworksdemos/queue-master:0.3.1 "/usr/local/bin/java..."          queue-master 39 minutes ago  Up 39 minutes  4369/tcp, 5671-567
2/tcp, 25672/tcp
docker-compose-rabbitmq-1           rabbitmq:3.6.8                     "docker-entrypoint.s..."          rabbitmq    39 minutes ago  Up 39 minutes  4369/tcp, 5671-567
2/tcp, 25672/tcp
docker-compose-shipping-1           weaveworksdemos/shipping:0.4.8     "/usr/local/bin/java..."          shipping    39 minutes ago  Up 39 minutes  80/tcp
docker-compose-user-1              weaveworksdemos/user:0.4.4         "/user -port=80"                    user        39 minutes ago  Up 39 minutes  80/tcp
docker-compose-user-db-1            weaveworksdemos/user-db:0.4.0     "/entrypoint.sh mong..."          user-db     39 minutes ago  Up 39 minutes  27017/tcp

```

## Endpoints:

```

module.exports = {
  catalogueUrl: util.format("http://catalogue%s", domain),
  tagsUrl:      util.format("http://catalogue%s/tags", domain),
  cartsUrl:     util.format("http://carts%s/carts", domain),
  ordersUrl:    util.format("http://orders%s", domain),
  customersUrl: util.format("http://user%s/customers", domain),
  addressUrl:   util.format("http://user%s/addresses", domain),
  cardsUrl:     util.format("http://user%s/cards", domain),
  loginUrl:     util.format("http://user%s/login", domain),
  registerUrl:  util.format("http://user%s/register", domain),
};

```

¿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:

1. **Separación de Preocupaciones:** Tener repositorios separados permite una clara separación entre el código de la aplicación y la configuración del sistema. Esto facilita la gestión y el mantenimiento de ambos aspectos de la aplicación de forma independiente.
2. **Gestión de Versiones:** Los cambios en el código y en la configuración del sistema a menudo tienen ciclos de vida diferentes. Tener repositorios separados permite gestionar las versiones de manera más específica y adecuada para cada componente.

3. **Colaboración más Fácil:** Diferentes equipos pueden trabajar en el código y la configuración de forma simultánea sin interferir entre sí. Esto mejora la colaboración y permite a los equipos especializarse en sus respectivas áreas.
4. **Automatización y Despliegue Continuo:** Al separar la configuración del sistema, se puede automatizar el proceso de implementación y configuración. Esto facilita la adopción de prácticas de despliegue continuo y orquestación de infraestructura.
5. **Mayor Flexibilidad en Despliegue:** Al tener la configuración separada, es más fácil ajustar y cambiar la infraestructura subyacente o los recursos sin necesariamente cambiar el código de la aplicación. Esto facilita la adaptación a cambios en la demanda o requisitos.

#### **Desventajas:**

1. **Complejidad Adicional:** Gestionar múltiples repositorios puede aumentar la complejidad, especialmente para equipos pequeños o proyectos menos sofisticados.
2. **Sincronización:** Es importante mantener sincronizados los cambios en el código y en la configuración para asegurarse de que funcionen bien juntos. Esto puede requerir un esfuerzo adicional.
3. **Dificultad en el Descubrimiento:** Para los nuevos miembros del equipo, puede ser complicado descubrir dónde se encuentra la configuración relevante, lo que podría llevar a retrasos y confusiones.
4. **Posible Desalineación:** Si los cambios en la configuración y el código no se coordinan adecuadamente, podría surgir una desalineación entre ambos, lo que podría causar problemas de compatibilidad.
5. **Mayor Complejidad en la Implementación Local:** Durante el desarrollo local o las pruebas, gestionar y mantener ambos repositorios puede requerir esfuerzo adicional y puede ser un proceso más complejo.

---

#### *¿Cuál contenedor hace las veces de API Gateway?*

El contenedor que hace de API Gateway es edge-router. El repositorio que corresponde a este contenedor tiene un archivo `traefik.toml`. Traefik es un *Edge Router*; esto significa que es 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 qué servicio es el encargado de gestionar cada petición.

---

¿Cuál de todos los servicios está procesando la operación?

```
usuario@DESKTOP-861HL08 MINGW64 ~/OneDrive/Escritorio/2023/Ing SW 3/socks-demo
$ curl http://localhost/customers
{"_embedded":{"customer":[{"firstName":"Eve","lastName":"Berger","username":"Eve_Berger","id":"57a98d98e4b00679b4a830af","links":{"addresses":{"href":"http://user/customer57a98d98e4b00679b4a830af/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":"agus","lastName":"araya","username":"aaraya0","id":"64e4f5fcee11cb001774fae","links":{"addresses":{"href":"http://user/customers/64e4f5fcee11cb001774fae/addresses"},"cards":{"href":"http://user/customers/64e4f5fcee11cb001774fae/cards"},"customer":{"href":"http://user/customers/64e4f5fcee11cb001774fae"},"self":{"href":"http://user/customers/64e4f5fcee11cb001774fae"}}}]}}
```

El servicio de **user** realiza la operación y se comunica con el **front-end**.

```
2023-08-22 15:49:55 docker-compose-user-1 | ts=2023-08-22T18:49:55Z caller=middlewares.go:75 method=GETUsers id=all result=4 took=925.9µs
2023-08-22 15:49:55 docker-compose-front-end-1 | GET /customers 200 7.310 ms - -
```

¿Cuál de todos los servicios está procesando la operación?

```
$ 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 edition beats. Each hole is lovingly progged to leave smooth edges. The only sock approved by a higher power.", "imageUrl":["/catalogue/images/holy.1.jpeg","/catalogue/images/holy.2.jpeg"],"price":99.99,"count":1,"tag":{"action","magic"}}, {"id":"3395a43e-2d88-40de-b95f-e00e1502085b","name":"Colourful","description":"proident occaecat irure et excepteur labore minim nisi amet irure", "imageUrl":["/catalogue/images/colourful_socks.jpg","/catalogue/images/colourful_socks.jpg"],"price":18,"count":438,"tag":{"brown","blue"}}, {"id":"510a0d7e-8e83-4193-b483-e27e09ddc34d","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-a9b9-6612e8f29a38","name":"Crossed","description":"A mature sock, crossed, with an air of nonchalance.", "imageUrl":["/catalogue/images/cross.1.jpeg","/catalogue/images/cross.2.jpeg"],"price":17.32,"count":738,"tag":{"blue","action","red","formal"}}, {"id":"819e1fbf-8b7e-4f6d-811f-e93534916a8b","name":"Figueroa","description":"enim officia aliquam excepteur esse deserunt quis aliquip nostrud anim", "imageUrl":["/catalogue/images/WAT.jpg","/catalogue/images/WAT2.jpg"],"price":14,"count":808,"tag":{"green","formal","blue"}}, {"id":"037ab141-399e-4c1f-9abc-bace48296bac","name":"Cat socks","description":"consequat amet cupidatat minim laborum tempor elite ex consequat in", "imageUrl":["/catalogue/images/catsocks.jpg","/catalogue/images/catsocks2.jpg"],"price":15,"count":175,"tag":{"brown","formal","green"}}, {"id":"a0a4f044-b040-410d-8ead-4de0446ac7e","name":"Nerd leg","description":"For all those leg lovers out there. A perfect example of a s wivel chair trained calf. Meticulously trained on a diet of sitting and Pina Colodas. Phwrrr...", "imageUrl":["/catalogue/images/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 bit geeky.", "imageUrl":["/catalogue/images/youtube.1.jpeg","/catalogue/images/youtube.2.jpeg"],"price":10.99,"count":801,"tag":{"formal","geek"}}, {"id":"zzz4f044-b040-410d-8ead-4de0446ac7e","name":"Classic","description":"Keep it simple.", "imageUrl":["/catalogue/images/classic.jpg","/catalogue/images/classic2.jpg"],"price":12,"count":127,"tag":{"brown","green"}}
```

El servicio de **catalogue** realiza la operación y se comunica con el **front-end**.

```
2023-08-22 15:54:30 docker-compose-catalogue-1 | ts=2023-08-22T18:54:30Z caller=logging.go:36 method=List tags= order=id pageNum=1 pageSize=10 result=9 err=null took=1.6094ms
2023-08-22 15:54:30 docker-compose-front-end-1 | GET /catalogue 200 8.570 ms - -
```

¿Cuál de todos los servicios está procesando la operación?

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

El servicio de **catalogue** realiza la operación y se comunica con el **front-end**.

```
2023-08-22 15:55:57 docker-compose-catalogue-1 | ts=2023-08-22T18:55:57Z caller=logging.go:74 method=Tags result=11 err=null took=518.9µs
2023-08-22 15:55:57 docker-compose-front-end-1 | GET /tags 200 4.009 ms - -
```

¿Como persisten los datos los servicios?

**Volúmenes de Docker:** Los volúmenes son una forma de persistir datos en Docker. Los volúmenes son directorios o archivos que están fuera del sistema de archivos del contenedor y se montan en el contenedor. Esto permite que los datos persistan incluso cuando el contenedor se detiene o se elimina. Los volúmenes pueden ser administrados por Docker y pueden compartirse entre varios contenedores.

Los servicios que deben almacenar datos, en este caso, user-db y catalogue-db, tienen creados volúmenes de Docker para la persistencia de datos.

Dockerfile de user-db:

```
FROM mongo:3
ADD ./scripts /tmp/scripts

# Modify child mongo to use /data/db-users as dbpath (because /data/db wont persist the build because it is already a VOLUME)
RUN mkdir -p /data/db-users \
    && echo "dbpath = /data/db-users" > /etc/mongodb.conf \
    && chown -R mongodb:mongodb /data/db-users

RUN su - mongodb && mongod --fork --logpath /var/log/mongodb.log --dbpath /data/db-users \
    && /tmp/scripts/mongo_create_insert.sh \
    && mongod --dbpath /data/db-users --shutdown \
    && chown -R mongodb /data/db-users

# Make the new dir a VOLUME to persist it
VOLUME /data/db-users

CMD ["mongod", "--config", "/etc/mongodb.conf", "--smallfiles"]
```

**VOLUME /data/db-users:**

Declara /data/db-users como un volumen. *Los volúmenes en Docker son puntos de montaje que pueden persistir datos incluso después de que el contenedor se detenga o se elimine.*

---

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

El componente encargado del procesamiento de la cola de mensajes es el servicio llamado `queue-master`

---

¿Qué tipo de interfaz utilizan estos microservicios para comunicarse?

Estos microservicios utilizan principalmente comunicación basada en **HTTP/HTTPS** para interactuar entre sí. Esto se logra a través de interfaces API expuestas por los servicios que permiten a otros microservicios realizar solicitudes HTTP a rutas específicas para obtener o enviar datos.

Por ejemplo:

```

2023-08-22 16:21:02 docker-compose-catalogue-1 | ts=2023-08-22T19:21:02Z caller=logging.go:62 method=Get i
d=510a0d7e-8e83-4193-b483-e27e09ddc34d sock=510a0d7e-8e83-4193-b483-e27e09ddc34d err=null took=1.9849ms
2023-08-22 16:21:02 docker-compose-front-end-1 | GET /catalogue/510a0d7e-8e83-4193-b483-e27e09ddc34d 200 1
1.044 ms - -
2023-08-22 16:21:03 docker-compose-front-end-1 | Request received: /cart, undefined
2023-08-22 16:21:03 docker-compose-front-end-1 | Customer ID: 9Wa5D9lbL8pg32Rl06D8lXKeUMbNfs2_
2023-08-22 16:21:03 docker-compose-catalogue-1 | ts=2023-08-22T19:21:03Z caller=logging.go:36 method=List
tags=sport order=id pageNum=1 pageSize=3 result=1 err=null took=1.9206ms
2023-08-22 16:21:03 docker-compose-front-end-1 | GET /catalogue?sort=id&size=3&tags=sport 200 7.953 ms - -
2023-08-22 16:21:03 docker-compose-front-end-1 | GET /cart 200 108.897 ms - -

```