

1)

<input type="checkbox"/>		docker-compose		Running (14/1)	78.41%	1 minute ago			
<input type="checkbox"/>		payment-1	weaveworksdemos/payment:0.4.3	Running	0%	4 minutes ago			
<input type="checkbox"/>		front-end-1	weaveworksdemos/front-end:0.3.12	Running	0.01%	4 minutes ago			
<input type="checkbox"/>		carts-1	weaveworksdemos/carts:0.4.8	Running	74.74%	4 minutes ago			
<input type="checkbox"/>		carts-db-1	mongo:3.4	Running	0.37%	4 minutes ago			
<input type="checkbox"/>		edge-router-1	weaveworksdemos/edge-router:0.1.1	Running	0.04%	4 minutes ago			
<input type="checkbox"/>		user-1	weaveworksdemos/user:0.4.4	Running	0.03%	4 minutes ago			
<input type="checkbox"/>		queue-master-1	weaveworksdemos/queue-master:0.3.1	Running	0.19%	4 minutes ago			
<input type="checkbox"/>		orders-db-1	mongo:3.4	Running	0.38%	4 minutes ago			
<input type="checkbox"/>		rabbitmq-1	rabbitmq:3.6.8	Running	0.84%	4 minutes ago			
<input type="checkbox"/>		catalogue-1	weaveworksdemos/catalogue:0.3.5	Running	0%	4 minutes ago			
<input type="checkbox"/>		user-sim-1	weaveworksdemos/load-test:0.1.1	Exited	0%	1 minute ago			

OFFER OF THE DAY Buy 1000 socks, get a shoe for free!

Logged in as Lautaro Saenz | Logout

HOME CATALOGUE ACCOUNT

0 items in cart

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
# 652835fa2ab79c0007de9644	2023-10-12 18:07:54	\$ 22.99	Shipped	<a href="#">View</a>

2)

Los contenedores creados son los siguientes:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
e5116e5c9e7c	weaveworksdemos/catalogue-db:0.3.0	"docker-entrypoint.s..."	41 minutes ago	Up 25 minutes	3306/tcp	docker-compose-catalogue-db-1
f21059ca182d	weaveworksdemos/front-end:0.3.12	"/usr/local/bin/npm ..."	41 minutes ago	Up 25 minutes	8079/tcp	docker-compose-front-end-1
e16d57a775b6	mongo:3.4	"docker-entrypoint.s..."	41 minutes ago	Up 25 minutes	27017/tcp	docker-compose-carts-db-1
2dd540d5fd64	weaveworksdemos/edge-router:0.1.1	"traefik"	41 minutes ago	Up 25 minutes	0.0.0.0:80->80/tcp, 0.0.0.0:8080->8080/tcp	docker-compose-edge-router-1
ea45eb14928a	weaveworksdemos/carts:0.4.8	"/usr/local/bin/java..."	41 minutes ago	Up 25 minutes	80/tcp	docker-compose-carts-1
34b280952ad9	weaveworksdemos/catalogue:0.3.5	"/app -port=80"	41 minutes ago	Up 25 minutes	4369/tcp, 5671-5672/tcp, 25672/tcp	docker-compose-catalogue-1
6f4c739f0528	rabbitmq:3.6.8	"docker-entrypoint.s..."	41 minutes ago	Up 25 minutes	27017/tcp	docker-compose-rabbitmq-1
9fbd351aeb11	weaveworksdemos/user-db:0.4.0	"/entrypoint.sh mong..."	41 minutes ago	Up 25 minutes	27017/tcp	docker-compose-user-db-1
5ffa78402cba	weaveworksdemos/orders:0.4.7	"/usr/local/bin/java..."	41 minutes ago	Up 25 minutes	27017/tcp	docker-compose-orders-1
8b5641ebfde8	mongo:3.4	"docker-entrypoint.s..."	41 minutes ago	Up 25 minutes	27017/tcp	docker-compose-carts-db-1
744e4db8b94f	weaveworksdemos/queue-master:0.3.1	"/usr/local/bin/java..."	41 minutes ago	Up 25 minutes	80/tcp	docker-compose-queue-master-1
a97450f4a8d5	weaveworksdemos/user:0.4.4	"/user -port=80"	41 minutes ago	Up 25 minutes	80/tcp	docker-compose-user-1
8b167989cb1e	weaveworksdemos/payment:0.4.3	"/app -port=80"	41 minutes ago	Up 25 minutes	80/tcp	docker-compose-payment-1
18eb32dcfef	weaveworksdemos/shipping:0.4.8	"/usr/local/bin/java..."	41 minutes ago	Up 25 minutes		docker-compose-shipping-1

CONTAINER ID	IMAGE
e5116e5c9e7c	weaveworksdemos/catalogue-db:0.3.0
f21059ca182d	weaveworksdemos/front-end:0.3.12
e16d57a775b6	mongo:3.4
2dd540d5fd64	weaveworksdemos/edge-router:0.1.1
ea45eb14928a	weaveworksdemos/carts:0.4.8
34b280952ad9	weaveworksdemos/catalogue:0.3.5
6f4c739f0528	rabbitmq:3.6.8
9fbd351aeb11	weaveworksdemos/user-db:0.4.0
5ffa78402cba	weaveworksdemos/orders:0.4.7
8b5641ebfde8	mongo:3.4
744e4bd8b94f	weaveworksdemos/queue-master:0.3.1
a97450f4a8d5	weaveworksdemos/user:0.4.4
8b167989cb1e	weaveworksdemos/payment:0.4.3
18eb32d8cfef	weaveworksdemos/shipping:0.4.8

**Catalogue-db:** Es el contenedor encargado de persistir los datos del catálogo.

**front-end:** el front end del proyecto.

**mongo:** base de datos no relacional.

**edge-router:** Este es el punto de entrada, o el equivalente al API Gateway.

**carts:** Carrito de compras.

**Catalogue:** servicio de catálogo.

**Rabbitmq:** rabbitmq es un software de cola de mensajes.

**user-db:** base de datos de usuarios.

**orders:** servicio encargado de la lógica de las órdenes.

**queue-master:** procesa los mensajes de la cola de rabbitmq.

**user:** servicio encargado de los usuarios.

**payment:** servicio encargado de los pagos.

**shipping:** servicio encargado de los envíos.

El punto de ingreso del sistema es el edge-router.

3)

Se utilizan repositorios separados para individualizar las responsabilidades de cada parte del sistema, así si se necesita modificar algo, o si alguna parte da un error se identifica más fácilmente y el resto del código puede seguir funcionando normalmente. Algunas ventajas de esto son:

1. Separación de Responsabilidades: Al dividir el código fuente en múltiples repositorios, cada uno puede tener una responsabilidad clara y estar enfocado en una parte específica del sistema. Esto facilita la colaboración y el mantenimiento, ya que los equipos pueden trabajar de forma independiente en sus componentes.
2. Escalabilidad y Despliegue Independiente: Si el proyecto se compone de varios microservicios o componentes independientes, cada uno puede tener su propio ciclo de vida y escalar de manera independiente. Esto permite desplegar y escalar partes específicas de la aplicación sin tener que tocar todo el proyecto.
3. Reutilización y Flexibilidad: Los componentes o servicios que se encapsulan en contenedores Docker pueden ser reutilizados en diferentes proyectos si se almacenan en repositorios separados. Esto fomenta la modularidad y la flexibilidad, ya que los mismos contenedores pueden ser utilizados en múltiples contextos.
4. Integración Continua y Entregas Continuas (CI/CD): Al separar los repositorios, es más fácil configurar flujos de trabajo de CI/CD para cada componente por separado. Esto permite la automatización del proceso de construcción, prueba y despliegue para cada parte del sistema.
5. Seguridad y Control de Acceso: En un entorno de desarrollo empresarial, puede haber necesidad de controlar el acceso a diferentes partes del sistema. Al tener repositorios separados, es posible establecer políticas de acceso específicas para cada componente, lo que mejora la seguridad y el control de versiones.
6. Versionado Independiente: Cada componente puede tener su propio ciclo de versionado. Esto es especialmente útil cuando diferentes partes del sistema se desarrollan y evolucionan a diferentes velocidades.
7. Facilidad de Mantenimiento: Cuando un componente necesita ser actualizado o corregido, los cambios se pueden hacer en el repositorio específico sin afectar a otras partes del sistema. Esto facilita el mantenimiento y la solución de problemas.

Y algunas desventajas son:

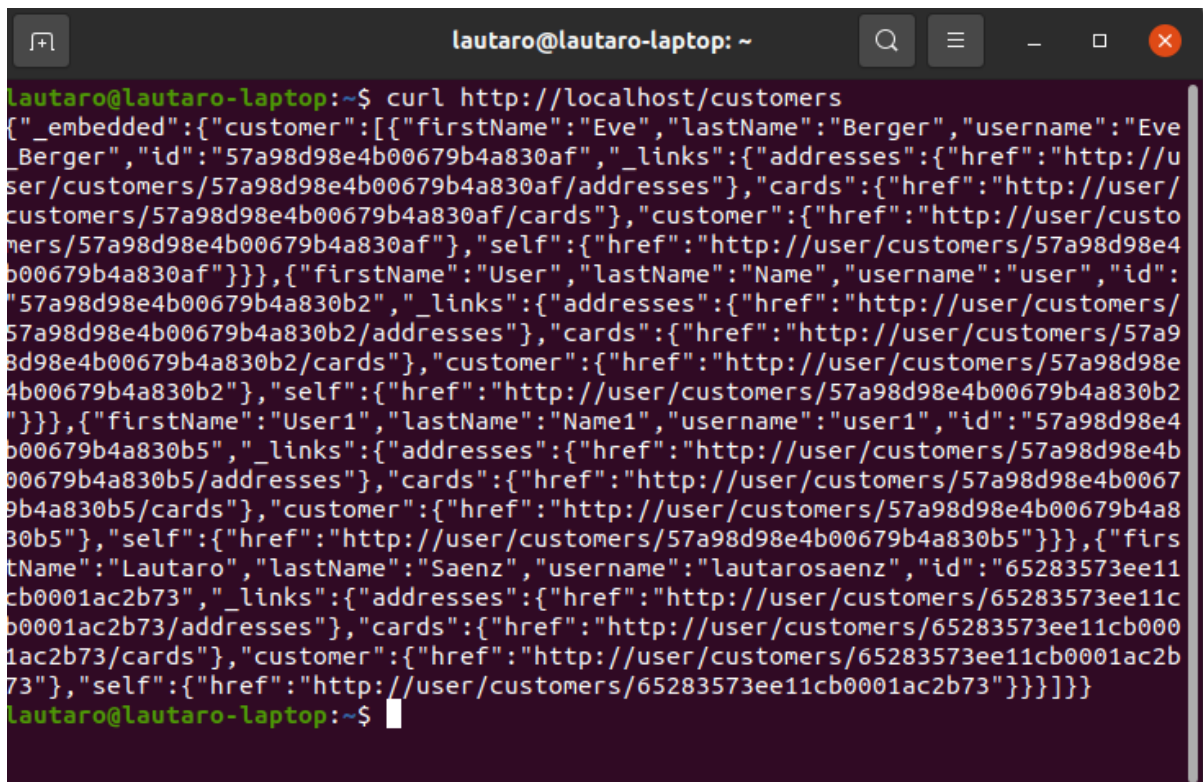
1. Complejidad en la gestión: Mantener múltiples repositorios implica una mayor complejidad en términos de gestión y coordinación. Coordinar cambios entre los componentes y asegurarse de que todas las partes sean compatibles puede ser un desafío.

2. Dificultades en la Integración: Si los componentes interactúan fuertemente entre sí, puede haber desafíos en la integración y las pruebas. Asegurar que las actualizaciones en un componente no rompan la funcionalidad en otros componentes puede requerir un esfuerzo adicional de integración y prueba.
3. Problemas de Versionamiento: A veces, las dependencias entre componentes pueden generar problemas de versionamiento. Asegurar que todas las partes del sistema estén en la misma versión puede ser complicado, especialmente si hay diferencias en los ciclos de desarrollo y lanzamiento.
4. Desafíos en CI/CD: Configurar flujos de trabajo de CI/CD para múltiples repositorios puede ser más complejo que hacerlo para un solo repositorio. La coordinación de los despliegues y las pruebas puede requerir una planificación cuidadosa y una infraestructura adecuada.
5. Dificultades en la comunicación: La comunicación efectiva entre equipos y componentes es crucial. Si no hay una comunicación adecuada, los equipos pueden tomar decisiones que afecten a otros componentes sin darse cuenta, lo que puede llevar a problemas de compatibilidad y coherencia en el sistema.
6. Aumento en la complejidad del desarrollo local: Los desarrolladores que trabajan en varios componentes pueden necesitar administrar múltiples repositorios localmente, lo que puede complicar los flujos de trabajo de desarrollo y pruebas en los entornos locales.
7. Dificultades en la Depuración y Rastreo de Problemas: Rastrear problemas que involucren múltiples componentes puede ser complicado. Identificar la causa raíz de un problema que se manifiesta en la interacción entre componentes puede llevar más tiempo y esfuerzo.
8. Posible Duplicación de Código: Si no hay una gestión adecuada, los equipos pueden terminar replicando ciertas funcionalidades en múltiples componentes, lo que lleva a la duplicación innecesaria del código.
9. Aumento en la complejidad de las políticas de seguridad: La implementación de políticas de seguridad coherentes en un entorno con múltiples repositorios puede requerir más esfuerzo, especialmente si se necesita controlar el acceso de forma granular.
10. Mayor Tiempo de Configuración Inicial: Configurar adecuadamente la estructura de múltiples repositorios y las interdependencias puede llevar más tiempo al principio del proyecto.

4)

El contenedor que simula ser el API Gateway es el que se denomina edge-router

5)

A terminal window titled 'lautaro@lautaro-laptop: ~' with standard window controls. The prompt is 'lautaro@lautaro-laptop:~\$'. The command 'curl http://localhost/customers' has been executed. The output is a large JSON array containing five customer objects. Each object has fields for '\_embedded' (a nested customer object), '\_links' (links to addresses and cards), 'firstName', 'lastName', 'username', and 'id'. The customers are: Eve Berger, User, User1, and Lautaro Saenz. The JSON is truncated on the right side of the terminal window.

```
lautaro@lautaro-laptop:~$ 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"},"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":"Lautaro","lastName":"Saenz","username":"lautarosaenz","id":"65283573ee11cb0001ac2b73","_links":{"addresses":{"href":"http://user/customers/65283573ee11cb0001ac2b73/addresses"},"cards":{"href":"http://user/customers/65283573ee11cb0001ac2b73/cards"},"customer":{"href":"http://user/customers/65283573ee11cb0001ac2b73"},"self":{"href":"http://user/customers/65283573ee11cb0001ac2b73"}}}]}}
lautaro@lautaro-laptop:~$
```

6)

El servicio que se encarga de obtener la solicitud es front-end. Pero se nutre del servicio user para obtener los datos, los cuales están guardados en user-db.

7)

Con estos dos servicios sucede lo mismo, primero el front-end obtiene la solicitud y este utiliza el servicio catalogue.

8)

Los datos de los servicios persisten en 4 bases de datos, una para cada servicio, los cuales son: Usuarios, Catálogo, Carritos y Órdenes.

9)

El componente encargado del procesamiento de la cola de mensajes es queue-master.

10)

Los microservicios se comunican mediante API Rest.

