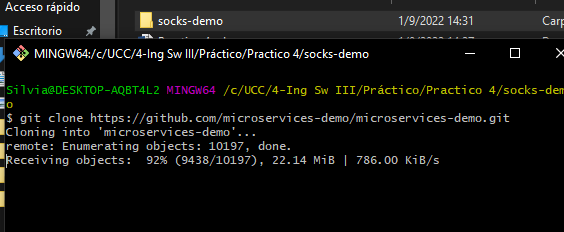
1. **Instanciación del sistema**

• Clonar el repositorio https://github.com/microservices-demo/microservicesdemo mkdir -p socks-demo

cd socks-demo

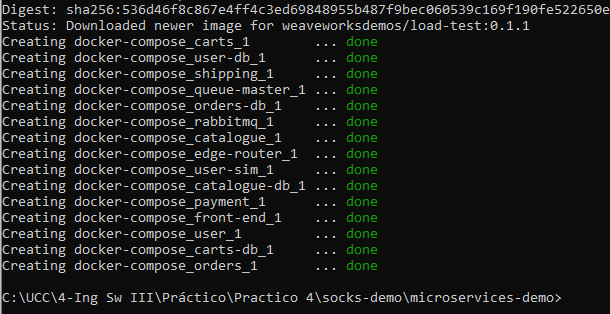
git clone https://github.com/microservices-demo/microservices-demo.git



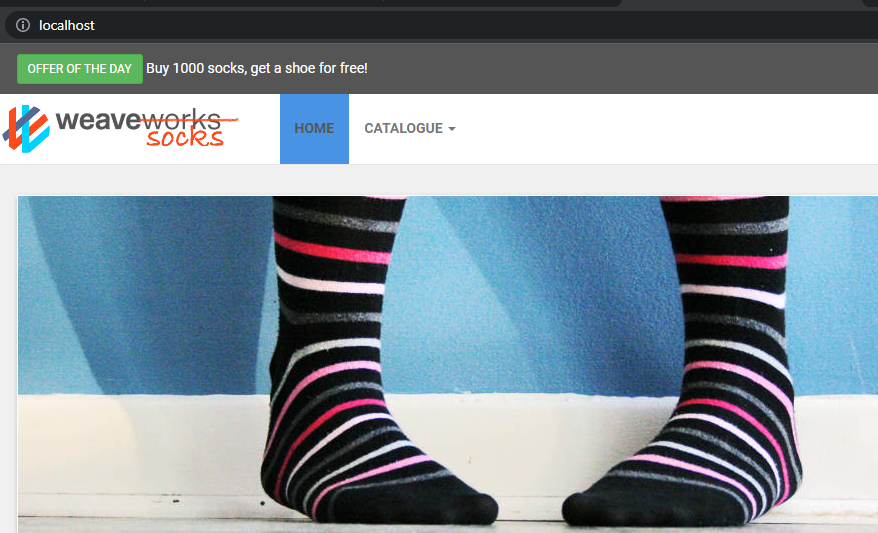
• Ejecutar lo siguiente

cd microservices-demo

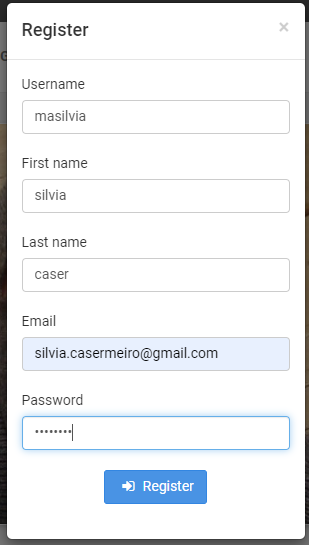
docker-compose -f deploy/docker-compose/docker-compose.yml up -d



• Una vez terminado el comando docker-compose acceder a <http://localhost>

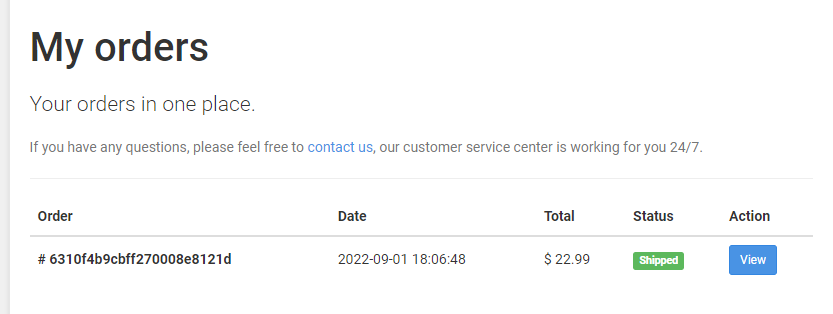


• Generar un usuario



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

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



1. **Investigación de los componentes**
2. Describa los contenedores creados, indicando cuales son los puntos de ingreso del sistema

Front-end: contenedor que carga el FE de la página usando la imagen weaveworksdemos/front-end:0.3.12 y nos permite visualizar la misma

Edge-router: api Gateway que mediante cap\_add redirecciona a catalogue, carts, orders, payment y user. Utiliza el Puerto 80 (visualización) y el 8080 (monitoreo) Esto es el ingreso del sistema

Catalogue: contenedor que carga el FE y los servicios de la opción catalogue en la página

Catalogue-db: contenedor que posee todos los datos del catálogo en la db. El nombre de la db es socksdb, el user es root y password null o vacía.

Carts: contenedor que carga el FE y los servicios de la opción del carrito de compras para añadir mas productos en la página

Carts-db: contenedor que posee todos los datos del carrito de compras en la DB. Utiliza una imagen de la db no relacional Mongo.

Orders: contenedor que carga el FE y los servicios de la opción orders en la página y que permite visualizar, una vez registrado, todas las órdenes de compra.

Orders-db: contenedor que posee todos los datos de las orders de la db. Usa una imagen de una db no relacional Mongo.

Shipping: contenedor que carga el FE y los servicios de la opción envíos. No se visualiza porque no podemos realizar compras.

Queue-master: lógica de las queue. Implementa una queue de envíos y simula el proceso de envío.

Rabbitmq: una redis que maneja queues. El sistema usa esto para la carga de mensajes.

Payment: contenedor que carga el FE y los servicios de la opción payment en la página y que permite añadir los datos de la tarjeta de crédito/débito para realizar el pago.

User: contenedor que carga el FE y los servicios de la opción login en la página permitiendo añadir los datos del usuario para iniciar sesión y acceder a la página

User-db: contenedor que posee todos los datos de los usuarios en la db.

User-sim: contenedor que posee los datos de un user test que es usado por el desarrollador para probar o testear funcionalidades.

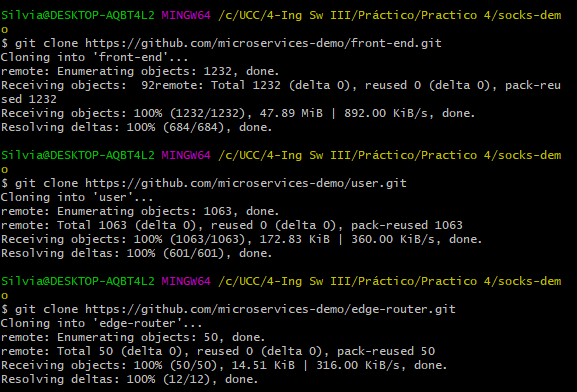
1. Clonar algunos de los repositorios con el código de las aplicaciones

cd socks-demo

git clone https://github.com/microservices-demo/front-end.git

git clone <https://github.com/microservices-demo/user.git>

git clone https://github.com/microservices-demo/edge-router.git



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.

Se tienen repos separados, uno para el código, otro para la config del sistema. Es así porque son servicios diferentes y funcionan diferente manejando código base independiente, pero comunicándose a través de Apis definidas. Los detalles de la implementación interna se ocultan frente a los otros.

Ventajas: equipo de trabajo mínimo, escalabilidad y funcionalidad modular, módulos independientes. Libertad del desarrollador para desarrollar y desplegar servicios de forma independiente. Uso de contenedores permitiendo el despliegue y desarrollo de la app rápidamente. Independizamos la lógica, en caso de algunos lenguajes

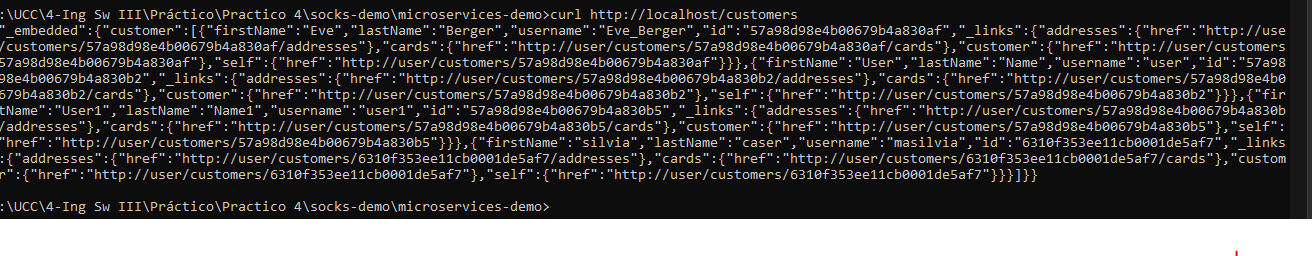
Desventajas: mayor consumo de recursos (cada microservicio tiene su propio SO y dependencias) Por eso se usa docker. La complejidad del despliegue es difícil de gestionar. Hay que lidiar con la complejidad adicional de los sistemas distribuidos. Implementar comunicación interna entre los servicios, dependencias entre servicios, solicitudes que pueden extenderse a varios servicios, etc. Esto es, la config inicial.

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

Edge-router. Traefik es la herramienta de API gateway

5. Cuando ejecuto este comando:

curl <http://localhost/customers>

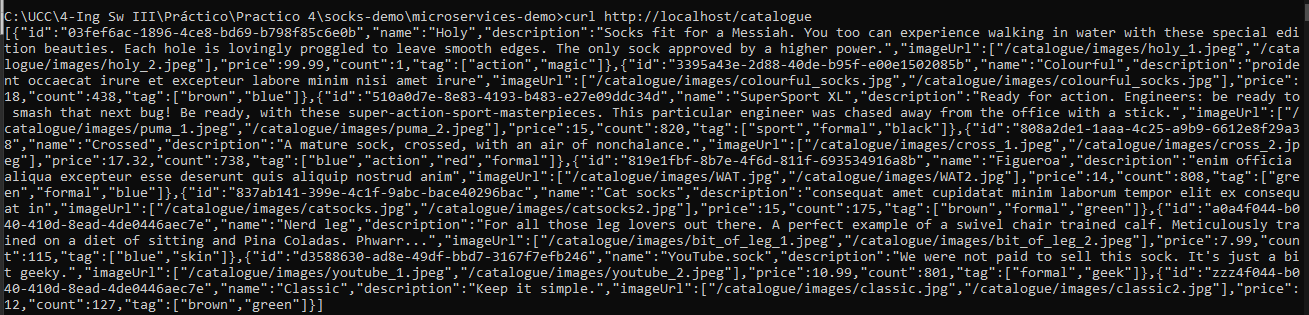


6. ¿Cuál de todos los servicios está procesando la operación?

Servicio user

7. ¿Y para los siguientes casos?

curl <http://localhost/catalogue>



Servicio Catalogue

curl <http://localhost/tags>



Servicio Catalogue

8. ¿Cómo persisten los datos los servicios?

Cada contenedor tiene su propio servicio con su db, lo que permite persistir los datos de cada uno. Las db Mongo almacenan info de usr, carrito de compra… La db MySql que se usa para guardar info del catálogo. Si se clonan, se tendrán los mismos datos, los originales.

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

Rabbitmq es un código que se usa para el manejo de mensajería.

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

La interfaz que usan estos microservicios para comunicarse es del tipo API REST.