

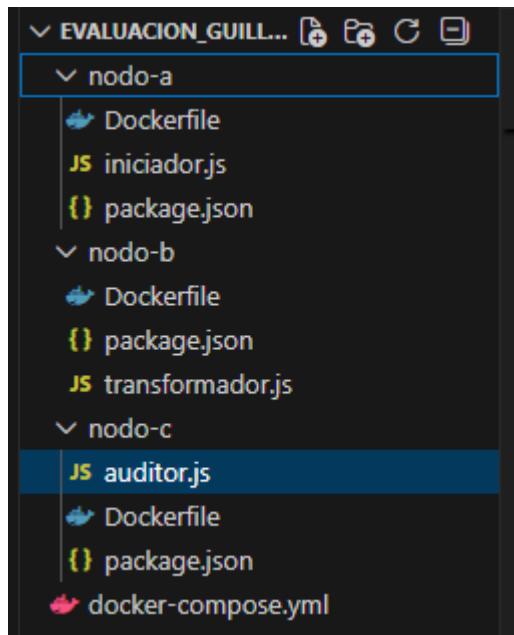
INFORME DE FUNCIONAMIENTO

Prueba de Sistemas

NOMBRE: Juan Guillen

LINK DEL GITHUB: https://github.com/JuanGuillenA/Evaluacion_Guillen.git

La arquitectura que se siguió es la que se pidió de ANILLO creando tres archivos con el nombre de los nodos: A, B y C.



La arquitectura que se usó es de ANILLO: de nodo A --> nodo B --> nodo C --> nodo A. Se cierra el anillo y se muestra el mensaje

Cada archivo cuenta con su Dockerfile para crear la imagen de cada uno y que se gestione por separado. Además de eso cada uno tiene su package.json con las versiones que se ocupan para crear cada uno de los códigos (los tres son lo mismo) solo que cada uno tiene el nombre del código creado en javascript con la lógica de lo que pedia las instrucciones.

El iniciador lo que hace es conecta a websocket para enviar los datos al nodo B, también valida el mensaje que se envío que sea con el formato correcto de JSON. También en el iniciador del nodo A creo el url para que según el valor que se envíe se haga la vuelta el anillo y se pueda calcular el valor si es par o impar en el nodo B. Agrega mensajes en la consola para que se vea como se va conectando y qué es lo que se pide para completar el ciclo

El transformador lo que hace es que se conecta a websocket para recibir el mensaje del nodo A y luego que lo verifica calcula si es el numero ingresado Par o Impar. Como se pido si es par se multiplica x2 y si es impar se suma 1 y se envia al nodo C. Tambien se agrega el mensaje B_PROCESSED al la lista del audit_trail, eso se puede ver en la consola como se agrega ese mensaje “B_processed” en el audit_trail

En el auditor del nodo C asi mismo se conecta al websocket para recibir y enviar los mensajes, cuando recibe el numero par o impar se le resta 5 y se envia el valor que seria e valor final del ciclo al nodo A para que muestre el mensaje. Tambien aqui se agrega a la lista del audit_trail el mensaje “C_verified” para que al final se muestre con el valor final ese mensaje.

En el docker compose.yml lo que hagoo es colcocar cada nodo para que sea un contenedor por separado y coloco que cada nodo depende del otro, y colco que solo el nodo A se conecte al puerto 8080 para poder escribir y mirar el valor de la operacion.

Capturas con resultados:

The screenshot shows the Docker Desktop interface. The left sidebar has options like Ask Gordon, Containers (selected), Images, Volumes, Builds, Models, Docker Hub, Docker Scout, and Extensions. The main area is titled "Containers" with a sub-header "View all your running containers and applications". It displays container usage: Container CPU usage (0.00% / 800% (8 CPUs available)) and Container memory usage (40.26MB / 30.46GB). A "Show charts" button is present. Below is a table of running containers:

	Name	Container ID	Image	Port(s)	CPU (%)	Last std	Actions
<input type="checkbox"/>	evaluacion_guill -	-	-	-	0.03%	1 minu	... Remove
<input type="checkbox"/>	nodo-c	d97b846d1bf3	evaluacion_	-	0%	1 minu	... Remove
<input type="checkbox"/>	nodo-b	0bf3880f453d	evaluacion_	-	0%	1 minu	... Remove
<input type="checkbox"/>	nodo-a	5e5cbe031907	evaluacion_	8080:8080 ↗	0.03%	1 minu	... Remove

Below the table, there's a "Walkthroughs" section with "Multi-container applications" and "Containerize your application" links. At the bottom, it shows system stats: Engine running, RAM 0.89 GB, CPU 0.63%, Disk: 2.18 GB used (limit 1006.85 GB).

Para el ejemplo coloque el numero 40

The screenshot shows a browser window with the URL "localhost:8080/start?value=40". The response is a JSON object: {"sent": {"_id": "71f933bf-472e-44e5-a7db-40bff020e479", "power_level": 40, "audit_trail": []}}. The browser interface includes back, forward, search, and other navigation buttons.

```
Attaching to nodo-a, nodo-b, nodo-c
nodo-c | Nodo C esta en el puerto 3002
nodo-b | Nodo B esta escuchando en el puerto 3001
nodo-b | Conectando a nodo-c
nodo-a | Nodo A esta en: http://localhost:8080/start?value= (EL NUMERO QUE SE QUIERA PROBAR)
nodo-a | Conectado a nodo-b
nodo-c | Conectado al nodo-a
[]

> ▾ TERMINAL
✓ Container nodo-b           Created          0.1s
✓ Container nodo-a           Created          0.1s
Attaching to nodo-a, nodo-b, nodo-c
nodo-c | Nodo C esta en el puerto 3002
nodo-b | Nodo B esta escuchando en el puerto 3001
nodo-b | Conectando a nodo-c
nodo-a | Nodo A esta en: http://localhost:8080/start?value= (EL NUMERO QUE SE QUIERA PROBAR)
nodo-a | Conectado a nodo-b
nodo-c | Conectado al nodo-a
nodo-b | Nodo B: con B_processed en audit_trail -> {"_id":"46b7421d-ad53-4a77-aeda-e571761e920b","power_level":0,"audit_trail":["B_processed"]}
nodo-c | Nodo C: con C_verified en audit_trail -> {"_id":"46b7421d-ad53-4a77-aeda-e571761e920b","power_level":-5,"audit_trail":["B_processed","C_verified"]}
nodo-a | CICLO COMPLETADO: -5
nodo-b | Nodo B: con B_processed en audit_trail -> {"_id":"dd3634e1-ea79-4fa7-bfab-5bcab03b6197","power_level":80,"audit_trail":["B_processed"]}
nodo-c | Nodo C: con C_verified en audit_trail -> {"_id":"dd3634e1-ea79-4fa7-bfab-5bcab03b6197","power_level":75,"audit_trail":["B_processed","C_verified"]}
nodo-a | CICLO COMPLETADO: 75
[]

v View in Docker Desktop  o View Config  w Enable Watch
```