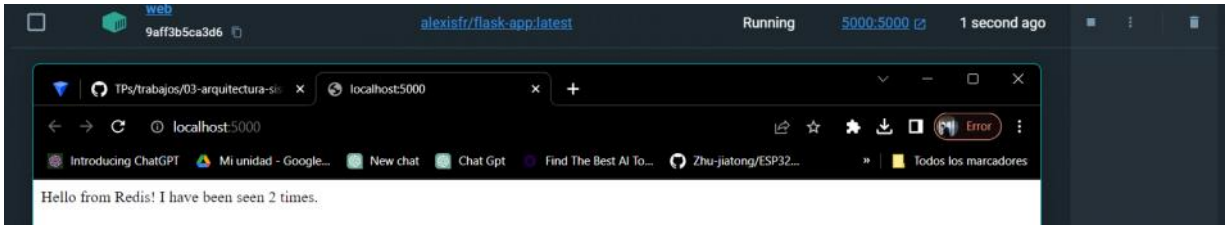
Practico 3

lunes, 16 de octubre de 2023 13:24

```
C:\Users\Benja>docker network create -d bridge mybridge
Error response from daemon: network with name mybridge already exists

C:\Users\Benja> docker run -d --net mybridge --name db redis:alpine
d5450a20bb955b7c5634e76e7612c532f0d4e9123cf22705392e7a9a37b585cd

C:\Users\Benja>  docker run -d --net mybridge -e REDIS_HOST=db -e REDIS_PORT=6379 -p 5000:5000 --name web alexisfr/flask
-app:latest
9aff3b5ca3d6e9e2c612ced41a54eec53c602ab6b0b79cdc4b58b5fb4dd20e8b

C:\Users\Benja>  docker run -d --net mybridge -e REDIS_HOST=db -e REDIS_PORT=6379 -p 5000:5000 --name web alexisfr/flask
-app:latest
docker: Error response from daemon: Conflict. The container name "/web" is already in use by container "9aff3b5ca3d6e9e2
c612ced41a54eec53c602ab6b0b79cdc4b58b5fb4dd20e8b". You have to remove (or rename) that container to be able to reuse tha
t name.
See 'docker run --help'.

C:\Users\Benja>  docker run -d --net mybridge -e REDIS_HOST=db -e REDIS_PORT=6379 -p 5000:5000 --name web alexisfr/flask
-app:latest
docker: Error response from daemon: Conflict. The container name "/web" is already in use by container "9aff3b5ca3d6e9e2
c612ced41a54eec53c602ab6b0b79cdc4b58b5fb4dd20e8b". You have to remove (or rename) that container to be able to reuse tha
t name.
See 'docker run --help'.

C:\Users\Benja>
```




```python
import os

from flask import Flask
from redis import Redis


app = Flask(__name__)
redis = Redis(host=os.environ['REDIS_HOST'], port=os.environ['REDIS_POR
bind_port = int(os.environ['BIND_PORT'])


@app.route('/')
def hello():
    redis.incr('hits')
    total_hits = redis.get('hits').decode()
    return f'Hello from Redis! I have been seen {total_hits} times.'


if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True, port=bind_port)
```

Explicacion del contador de visitas

Se crea una instancia de la aplicación Flask (app).
Se crea una instancia de Redis (redis) para conectarse al servidor Redis. Las direcciones del servidor y el puerto se toman de variables de entorno (REDIS_HOST y REDIS_PORT).
Definición de una ruta ("/") y una función asociada (hello):

Cuando se accede a la ruta principal ("/"), la función hello se ejecuta.
Dentro de la función, se incrementa el contador de visitas almacenado en Redis y se recupera el valor actual del contador.
Ejecución de la aplicación Flask:

Si el script se ejecuta directamente (es decir, no se importa como un módulo), la aplicación Flask se ejecutará.
El servidor se iniciará en "0.0.0.0" (lo que significa que estará disponible para conexiones desde cualquier dirección IP), en el puerto especificado por la variable de entorno BIND_PORT.
Docker y parámetros -e en el segundo Docker run del ejercicio 1:

En el contexto de Docker, el parámetro -e se utiliza para pasar variables de entorno al contenedor.
En el segundo docker run del ejercicio 1, se están pasando las variables de entorno REDIS_HOST, REDIS_PORT, y BIND_PORT al contenedor. Esto permite configurar dinámicamente la dirección y el puerto del servidor Redis, así como el puerto en el que el servidor web escuchará, sin necesidad de modificar el código fuente del script Flask.

Ejecutar docker rm -f web y luego docker run -d --net mybridge -e REDIS_HOST=db -e REDIS_PORT=6379 -p 5000:5000 --name web alexisfr/flask-app:latest:

Al ejecutar docker rm -f web, estás eliminando el contenedor llamado "web".
Luego, al ejecutar el segundo comando, estás iniciando un nuevo contenedor llamado "web" con la imagen alexisfr/flask-app:latest, configurando las variables de entorno REDIS_HOST y REDIS_PORT para apuntar al contenedor Redis llamado "db", exponiendo el puerto 5000 del contenedor al puerto 5000 del host.
Borrar el contenedor de Redis con docker rm -f db:

Al ejecutar este comando, estás eliminando el contenedor Redis llamado "db". Esto detendrá y eliminará el contenedor.
Levantar nuevamente el contenedor de Redis con docker run -d --net mybridge --name db redis:alpine:

Al ejecutar este comando, estás iniciando un nuevo contenedor Redis llamado "db" utilizando la imagen redis:alpine.
¿Qué ocurre en la página web cuando borro el contenedor de Redis?

Si el contenedor de Redis es eliminado, la aplicación Flask no podrá conectarse a la base de datos, y

cada vez que se haga una visita, no se incrementará el contador en Redis. La página web mostrará el mensaje, pero el contador de visitas no cambiará.
¿Qué considera usted que haría falta para no perder la cuenta de las visitas?

Para no perder la cuenta de las visitas, se necesita una solución de almacenamiento persistente para el contador de visitas. Actualmente, la aplicación Flask utiliza Redis, pero si el contenedor de Redis se detiene o se elimina, se pierde la información.
Una solución sería utilizar un servicio de Redis que persista los datos en el disco, como configurar Redis para que realice snapshots periódicos o utilizar una base de datos más persistente para el seguimiento de las visitas.

```
C:\Users\Benja\MiNotSoSimpleApp>docker-compose up -d
[+] Running 4/4
 ✓ Network minotsosimpleapp_default      Created                                    0.3s
 ✓ Volume "minotsosimpleapp_redis_data"  Created                                    0.1s
 ✓ Container minotsosimpleapp-db-1       Started                                    4.4s
 ✓ Container minotsosimpleapp-app-1      Started                                    5.1s

C:\Users\Benja\MiNotSoSimpleApp>docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED              STATUS              PORTS
               NAMES
4430f857a589   alexisfr/flask-app:latest  "python /app.py"         About a minute ago   Up About a minute   0.0.0.0:500
0->5000/tcp    minotsosimpleapp-app-1
e3e7634e2710   redis:alpine               "docker-entrypoint.s…"   About a minute ago   Up About a minute   6379/tcp
               minotsosimpleapp-db-1

C:\Users\Benja\MiNotSoSimpleApp>docker network ls
NETWORK ID     NAME                       DRIVER    SCOPE
019b9e25e164   bridge                     bridge    local
d96ec2a47bec   host                       host      local
3a8066b3c153   minotsosimpleapp_default   bridge    local
1ad387855115   none                       null      local

C:\Users\Benja\MiNotSoSimpleApp>docker volume ls
DRIVER    VOLUME NAME
local     8b71a19294736123bb7cc98b5adea48a0949e9e2e4a4f69e6ac639ec1db9928e
local     e5324973fd9e70cb8642c740d9412312ee6de0313a71739592ed654a4e5f1d32
local     minotsosimpleapp_redis_data

C:\Users\Benja\MiNotSoSimpleApp>
```

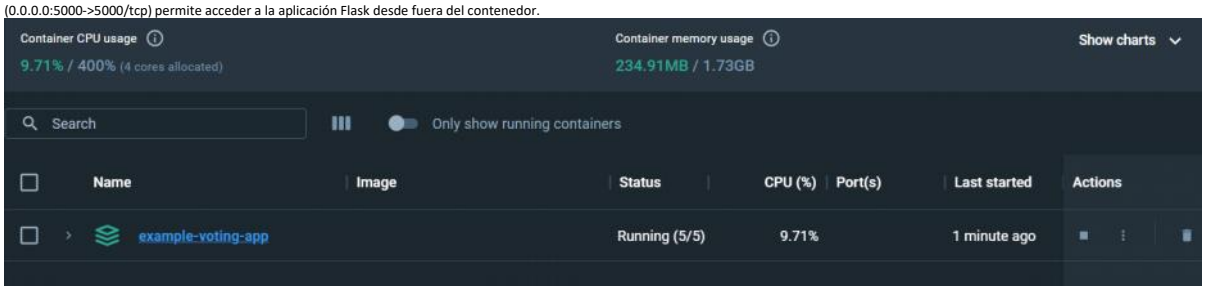
Docker Compose creó una red llamada minotsosimpleapp_default para permitir que los contenedores definidos en el archivo de composición se comuniquen entre sí. Esta red es de tipo "bridge".
Creación de volúmenes (minotsosimpleapp_redis_data):

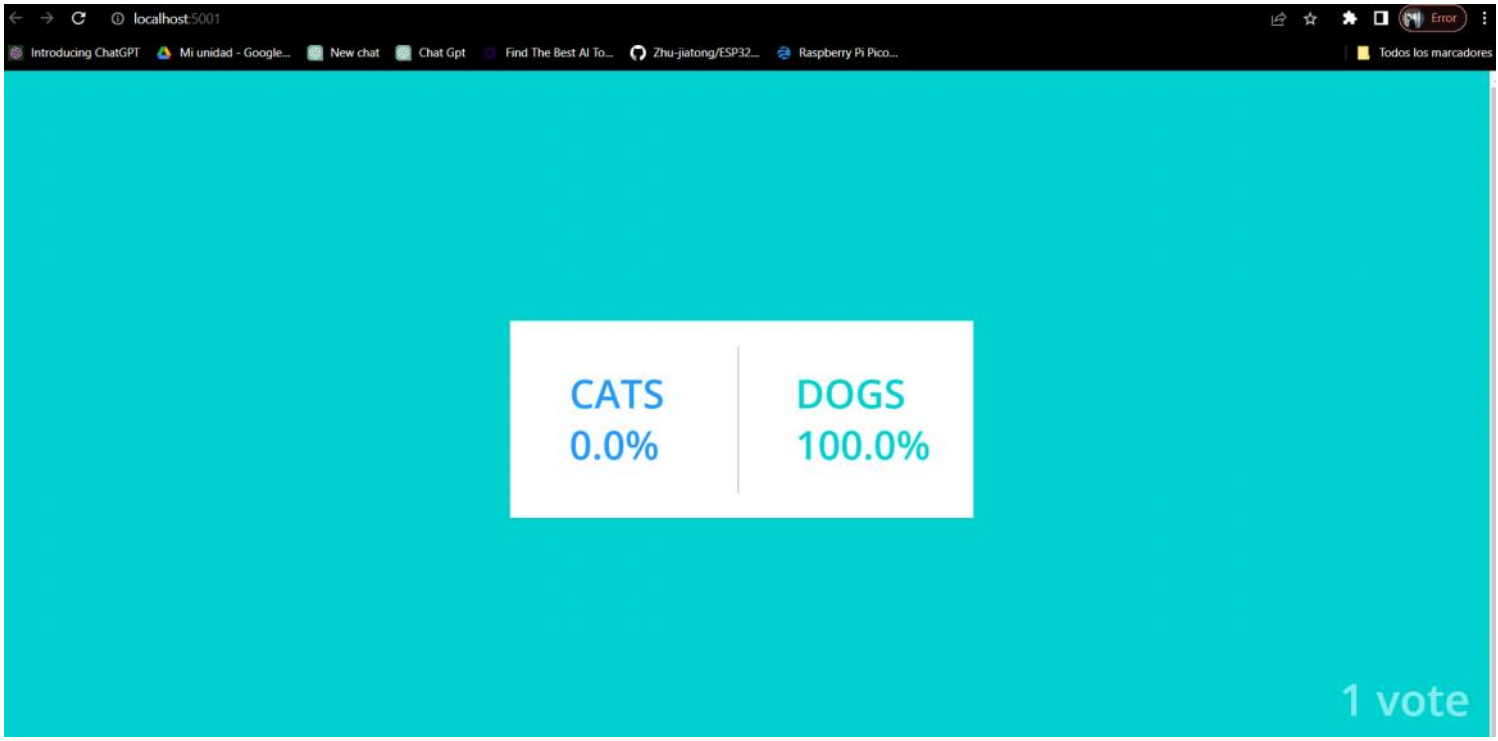
Docker Compose creó un volumen llamado minotsosimpleapp_redis_data. Los volúmenes son utilizados para persistir datos entre reinicios o recreación de contenedores. En este caso, parece que se está utilizando para persistir los datos de Redis.
Inicio de contenedores (minotsosimpleapp-db-1 y minotsosimpleapp-app-1):

Docker Compose inició dos contenedores, uno para la aplicación Flask (minotsosimpleapp-app-1) y otro para la base de datos Redis (minotsosimpleapp-db-1).
Ambos contenedores están conectados a la red minotsosimpleapp_default, lo que les permite comunicarse entre sí.
Asignación de nombres a los contenedores:

Docker Compose asignó nombres específicos a los contenedores, como minotsosimpleapp-app-1 y minotsosimpleapp-db-1. Esto facilita la referencia a los contenedores en lugar de usar identificadores de contenedor más largos y difíciles de recordar.
Exposición de puertos:

La aplicación Flask se está ejecutando en el contenedor minotsosimpleapp-app-1 y se ha configurado para exponer el puerto 5000. La exposición del puerto 5000 del contenedor al puerto 5000 del host (0.0.0.0:5000->5000/tcp) permite acceder a la aplicación Flask desde fuera del contenedor.




localhost:5001
Error
Introducing ChatGPT
Mi unidad - Google...
New chat
Chat Gpt
Find The Best AI To...
Zhu-jiatong/ESP32...
Raspberry Pi Pico...
Todos los marcadores
CATS
0.0%
DOGS
100.0%
1 vote