Centro Universitario de Ciencias Exactas e Ingenierías

CUCEI

Computación Tolerante a fallas

2023B

Arquitecturas monolíticas vs Microservicios

Michel Emanuel López Franco

Gutiérrez Galan Ruben Alejandro

Codigo: 214798315

Contenido

Introducción:

Un microservicio es una arquitectura de desarrollo de software que estructura una aplicación como un conjunto de servicios pequeños e independientes, cada uno ejecutándose en su propio proceso y comunicándose a través de mecanismos ligeros, como HTTP o mensajes. Estos microservicios están diseñados para realizar funciones específicas y pueden escalar y desplegarse de manera independiente. La idea es descomponer una aplicación monolítica en componentes más pequeños y manejables.

Docker, por otro lado, es una plataforma de contenedorización que permite empaquetar y distribuir aplicaciones junto con todas sus dependencias y configuraciones en contenedores. Los contenedores son unidades ligeras y portátiles que pueden ejecutarse en cualquier entorno que tenga Docker instalado.

Al combinar microservicios con Docker, se puede lograr una implementación eficiente y escalable. Cada microservicio se empaqueta en un contenedor Docker, lo que facilita la gestión de dependencias y la implementación consistente en diferentes entornos. Además, Docker proporciona herramientas para la orquestación de contenedores, como Docker Compose o Kubernetes, que facilitan la gestión, escalabilidad y mantenimiento de un conjunto de microservicios.

Programa desarrollado:

Codigo:

Este es un programa sencillo el cual nos permitirá usar una calculadora básica como un microservicio, la idea es poder enviar una serie de datos por medio de un servidor el cual nos permitirá realizar operaciones matemáticas y nos brindará de los resultados en su forma decimal y hexadecimal.

```
# app.py
from flask import Flask, request, jsonify

app = Flask(__name__)

def guardar_resultado(resultado_decimal, resultado_hexa):
    with open('resultados.txt', 'a') as file:
        file.write(f"Decimal: {resultado_decimal}, Hexadecimal:
{resultado_hexa}\n")
    file.close()

@app.route('/calcular', methods=['POST'])
def calcular():
    data = request.get_json()

    operacion = data['operacion']
    numero1 = data['numero1']
```

```
    numero2 = data['numero2']

    if operacion == 'sumar':
        resultado = numero1 + numero2
    elif operacion == 'restar':
        resultado = numero1 - numero2
    elif operacion == 'multiplicar':
        resultado = numero1 * numero2
    elif operacion == 'dividir':
        if not isinstance(numero1, (int, float)) or not isinstance(numero2,
(int, float)):
            return jsonify({"error": "Los números deben ser enteros o de
punto flotante"}), 400
        if numero2 != 0:
            resultado = numero1 / numero2
        else:
            return jsonify({"error": "No se puede dividir por cero"}), 400
    else:
        return jsonify({"error": "Operación no válida"}), 400
    resultado=int(resultado)
    resultado_hexa = hex(resultado)  # Convierte el resultado a hexadecimal
    guardar_resultado(resultado, resultado_hexa)  # Guarda el resultado en
el archivo

    return jsonify({"resultado": resultado, "resultado_hexa":
resultado_hexa})


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

Archivos importantes:

Dockerfile: contiene las especificaciones para descarga de extensiones o archivos necesarios para el arranque del microservicio, contiene el nombre de los archivos de arranque y de donde tomar las configuraciones.

app.py Dockerfile

Dockerfile

```
# Dockerfile

FROM python:3.8

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD [ "python", "./app.py" ]
```

requeriments.txt: contiene las configuraciones que vamos a necesitar para algunas extensiones, el lenguaje o programas que estamos utilizando.

app.py requirements.txt

requirements.txt

```
Flask==2.0.1
Werkzeug==2.0.1
```

Comandos de Docker usados:

Comando para crear el microservicio:

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS COMENTARIOS

```
PS D:\Programacion\Classroom\Computacion_Tolerante_a_fallas_1\Microservicio> docker build -t calculadora-microservicio .
```

Comando para verificar que el servicio se creó:

```
PS D:\Programacion\Classroom\Computacion_Tolerante_a_fallas_1\Microservicio> docker images
REPOSITORY                TAG       IMAGE ID       CREATED             SIZE
calculadora-microservicio   latest    d6122a5a8092   40 minutes ago      1.01GB
```

Comando para ejecutar el servidor:

```
PS D:\Programacion\Classroom\Computacion_Tolerante_a_fallas_1\Microservicio> docker run -p 5000:5000 --rm calculadora-microservicio
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.17.0.2:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 545-505-468
```

Ejecución del microservicio:

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  COMENTARIOS

PS D:\Programacion\Classroom\Computacion_Tolerante_a_fallas_1\Microservicio> Invoke-RestMethod -Uri "http://localhost:5000/calcular" -Metho
d Post -Headers @{"Content-Type"="application/json"} -Body '{"operacion": "dividir", "numero1": 50, "numero2": 10}'
>>

resultado resultado_hexa
--------- --------------
        5 0x5
```

```
resultado resultado_hexa
--------- --------------
        5 0x5


PS D:\Programacion\Classroom\Computacion_Tolerante_a_fallas_1\Microservicio> Invoke-RestMethod -Uri "http://localhost:5000/calcular" -Metho
d Post -Headers @{"Content-Type"="application/json"} -Body '{"operacion": "multiplicar", "numero1": 1000, "numero2": 20}'
>>

resultado resultado_hexa
--------- --------------
    20000 0x4e20
```

```
PS D:\Programacion\Classroom\Computacion_Tolerante_a_fallas_1\Microservicio> Invoke-RestMethod -Uri "http://localhost:5000/calcular" -Method Post
 -Headers @{"Content-Type"="application/json"} -Body '{"operacion": "muInvoke-RestMethod -Uri "http://localhost:5000/calcular" -Method Post -Head
ers @{"Content-Type"="application/json"} -Body '{"operacion": "multiplicar", "numero1": 1000, "numero2": 20,"operacion":"dividir", "numero3": 15}
'
>>

resultado resultado_hexa
--------- --------------
       50 0x32
```

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  COMENTARIOS

 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.17.0.2:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 545-505-468
172.17.0.1 - - [12/Nov/2023 23:28:32] "POST /calcular HTTP/1.1" 200 -
172.17.0.1 - - [12/Nov/2023 23:29:15] "POST /calcular HTTP/1.1" 200 -
172.17.0.1 - - [12/Nov/2023 23:29:50] "POST /calcular HTTP/1.1" 200 -
172.17.0.1 - - [12/Nov/2023 23:30:33] "POST /calcular HTTP/1.1" 200 -
```

Conclusión:

Este fue un microservicio bastante sencillo, aun asi me costo entender algunas cosas por lo que tarde en desarrollarlo, si bien lo que hace no es muy complejo, tuve dificultades con algunas de los comandos de Docker debido a que no lograba hacer que funcionara el servidor, además de que cree las imágenes de manera incorrecta por lo cual no me reconocía el microservicio y constantemente me mandaba error del servidor, tuve que investigar algunos comando porque no sabia bien exactamente que hacían y un poco acerca de los archivos de Docker que permiten tener configuraciones predeterminadas para poder correr en diferentes sistemas operativos con lo que me funciono de buena manera, aprendí mucho de esta actividad y logre comprender un poco más acerca de Docker y su uso.

Bibliografía:

- *Docker: Accelerated Container Application Development*. (2023b, October 18). Docker. https://www.docker.com/
- *Welcome to Flask — Flask Documentation (3.0.X)*. (s. f.). https://flask.palletsprojects.com/en/3.0.x/