

Sección
SW72

Alumno
Sharon Antuanet Ivet Barrial Marin

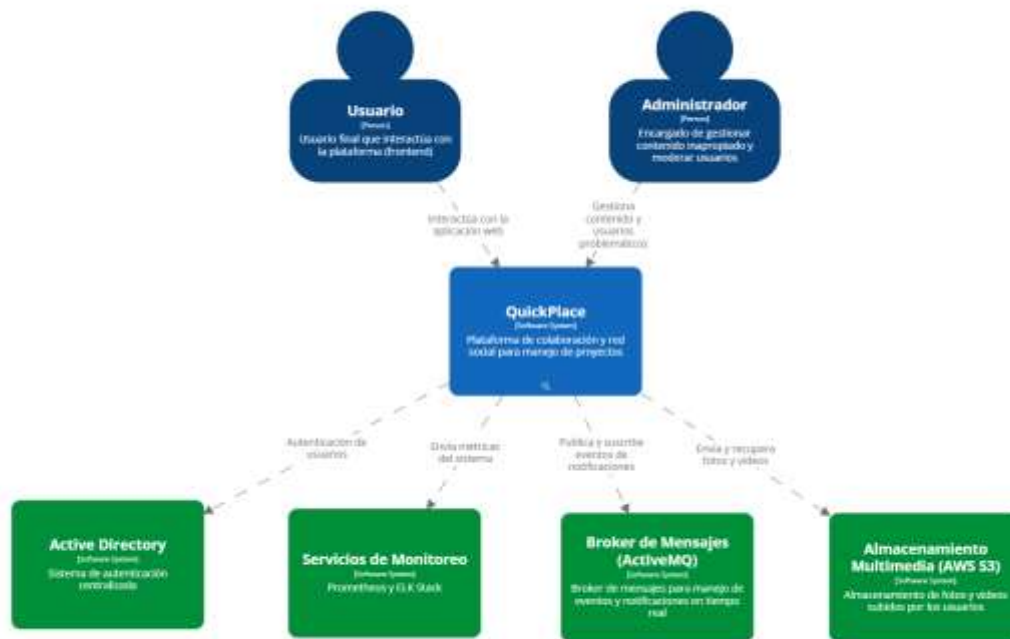
VERSION DE EXAMEN
B

SI657 - Fundamentos de Arquitectura de Software
Examen Final
202402

Pregunta 1 C4 Context Diagram

Criterios y sustento de decisión

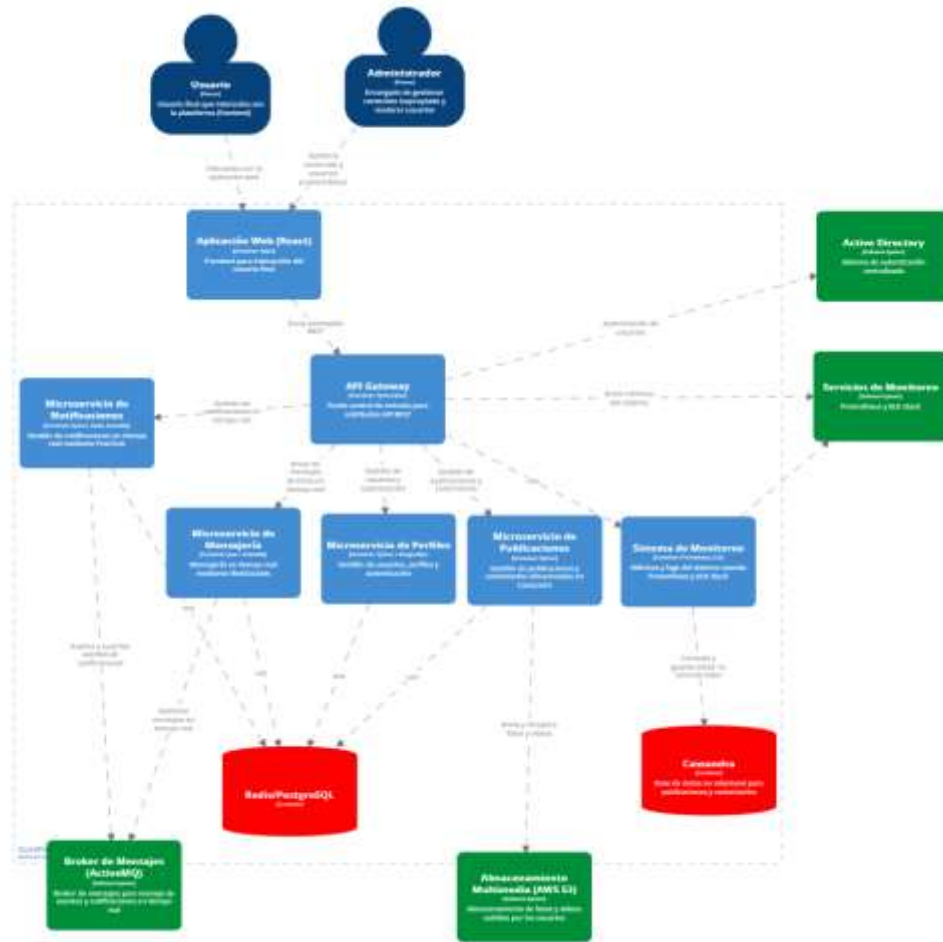
Se consideraron como usuarios a usuarios y administrador que se comunican con Quickplace donde se utiliza los sistemas externos active directory para la autenticación, servicios de monitoreo (prometheus y ELK stack, broker de mensajes (activeMQ) y almacenamiento multimedia AWS S3



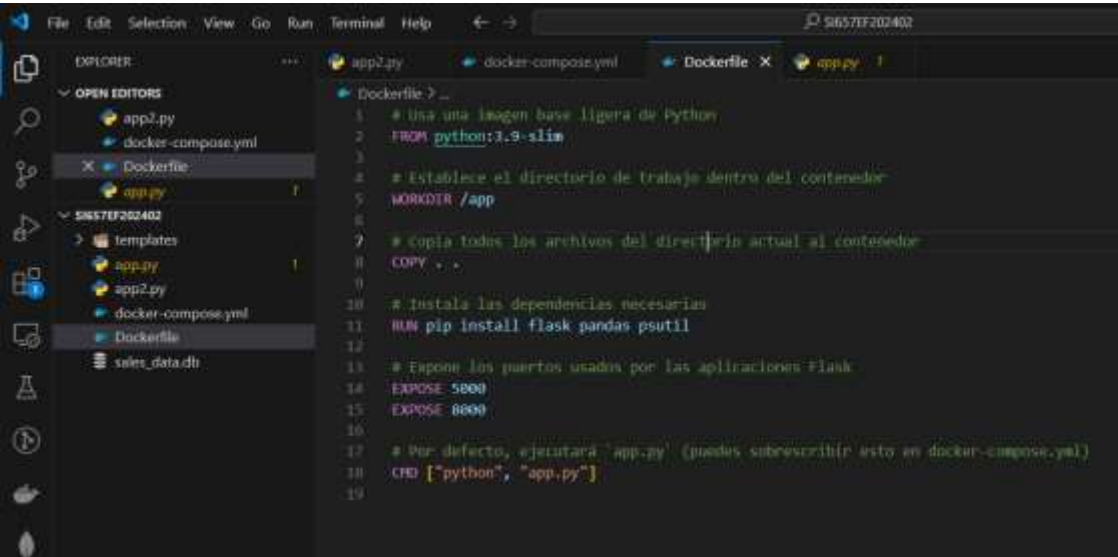
[System Context] QuickPlace

Diagrama de contexto de QuickPlace

Criterios y sustento de decisión

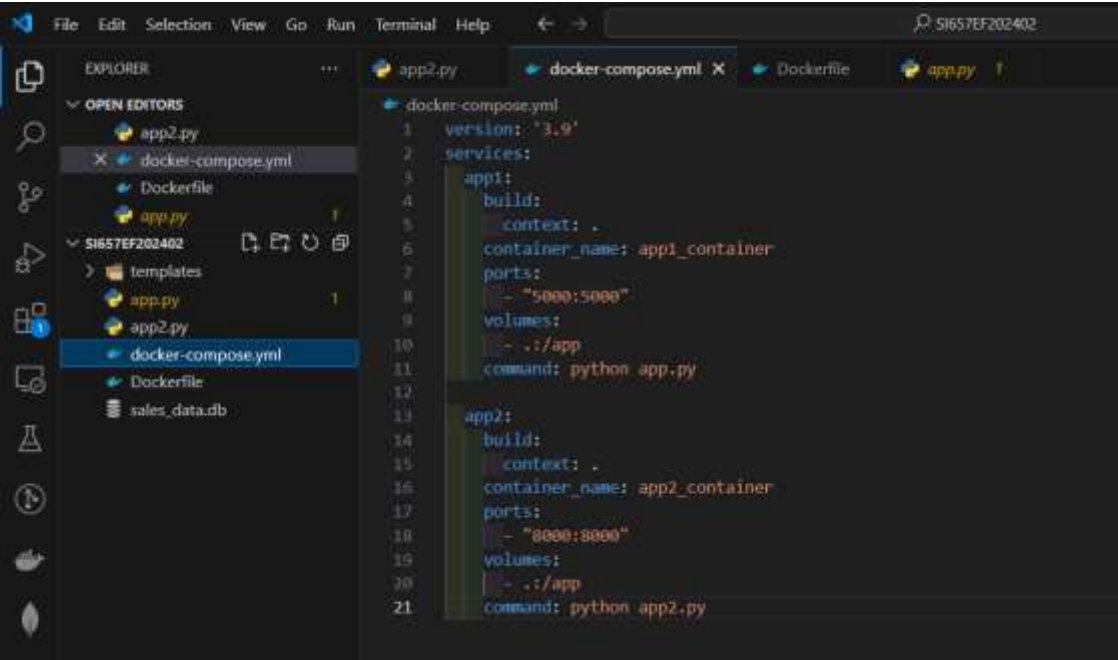


Se hizo uso de una arquitectura de microservicios contando con el apigateway como el punto central de entrada para las peticiones rest, se usó a Cassandra para el Sistema monitoro (logs) para datos no estructurados. Redis para el cache de mensaje y PostgreSQL para los mensajes.



The screenshot shows the VS Code interface with the Dockerfile open in the editor. The Explorer sidebar on the left shows the project structure with files like app2.py, docker-compose.yml, Dockerfile, and app.py. The Dockerfile content is as follows:

```
1 # Usa una imagen base ligera de Python
2 FROM python:3.9-slim
3
4 # Establece el directorio de trabajo dentro del contenedor
5 WORKDIR /app
6
7 # copia todos los archivos del directorio actual al contenedor
8 COPY . .
9
10 # instala las dependencias necesarias
11 RUN pip install flask pandas psutil
12
13 # Expone los puertos usados por las aplicaciones flask
14 EXPOSE 5000
15 EXPOSE 8000
16
17 # Por defecto, ejecutara 'app.py' (puedes sobrescribir esto en docker-compose.yml)
18 CMD ["python", "app.py"]
19
```



The screenshot shows the VS Code interface with the docker-compose.yml file open in the editor. The Explorer sidebar on the left shows the project structure. The docker-compose.yml content is as follows:

```
1 version: '3.9'
2 services:
3   app1:
4     build:
5       context: .
6     container_name: app1_container
7     ports:
8       - "5000:5000"
9     volumes:
10      - ./app
11     command: python app.py
12
13   app2:
14     build:
15       context: .
16     container_name: app2_container
17     ports:
18       - "8000:8000"
19     volumes:
20      - ./app
21     command: python app2.py

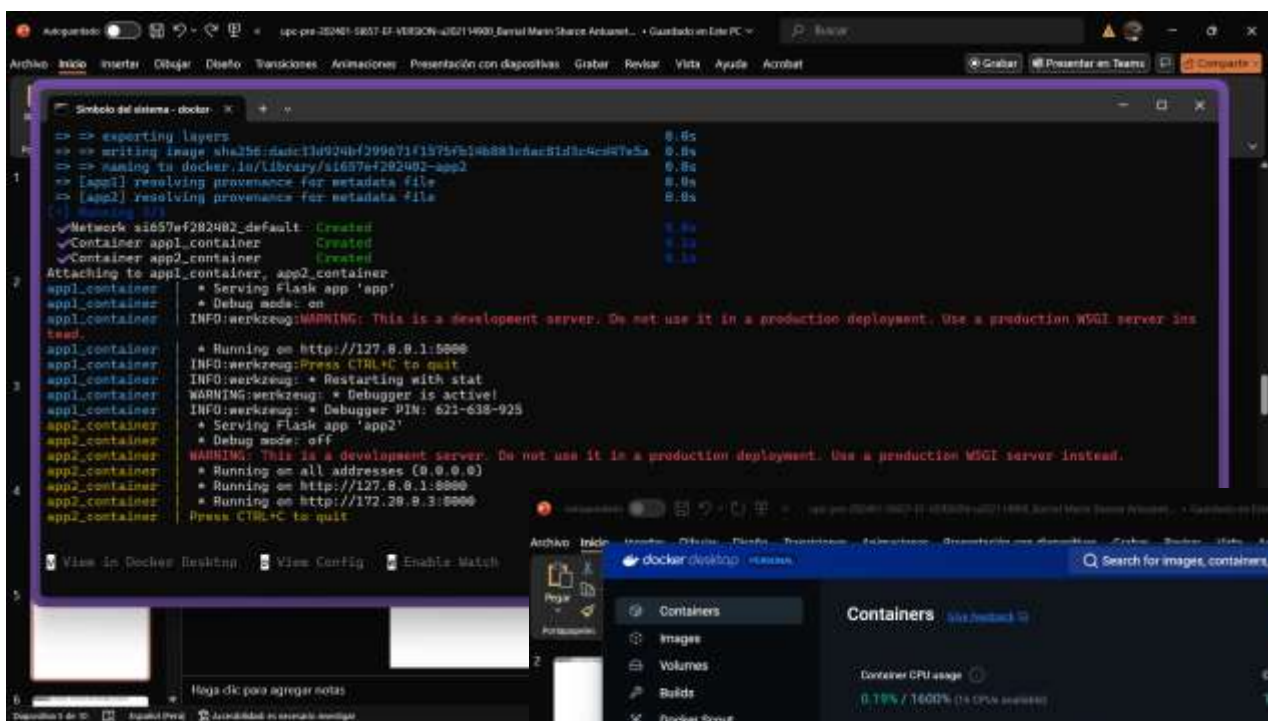
```

Criterios y sustento de decisión

Se hizo la creación del archivo Dockerfile para definir la construcción y ejecución del contenedor de la aplicación app.py

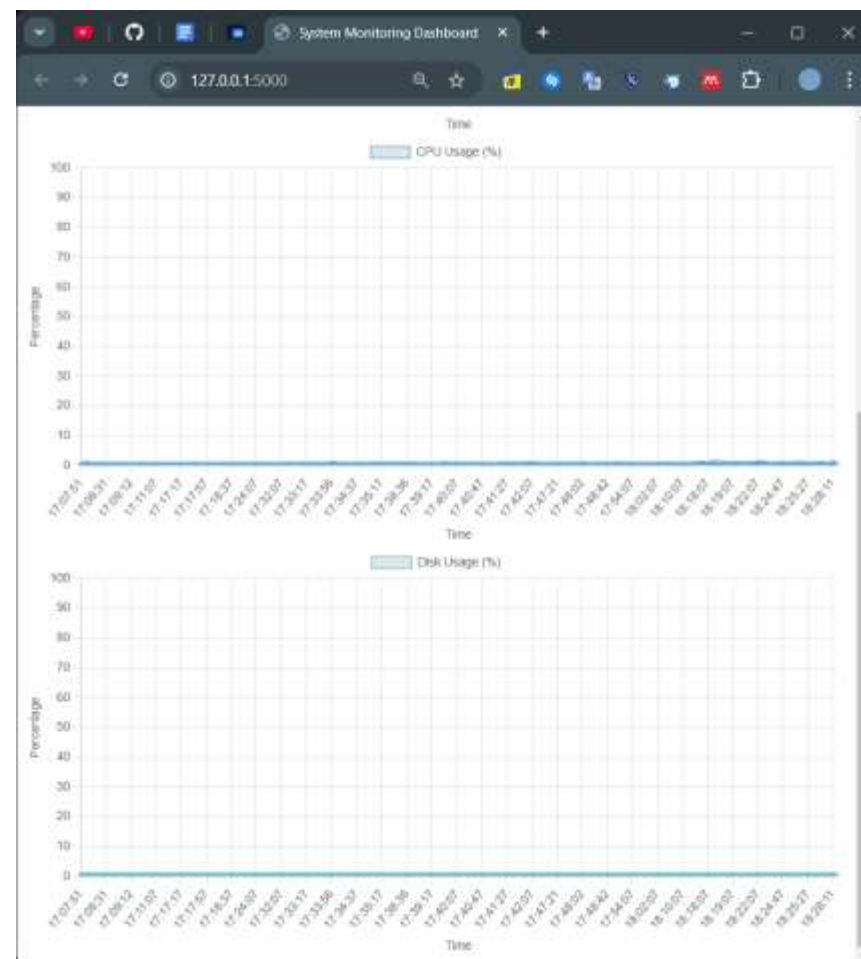
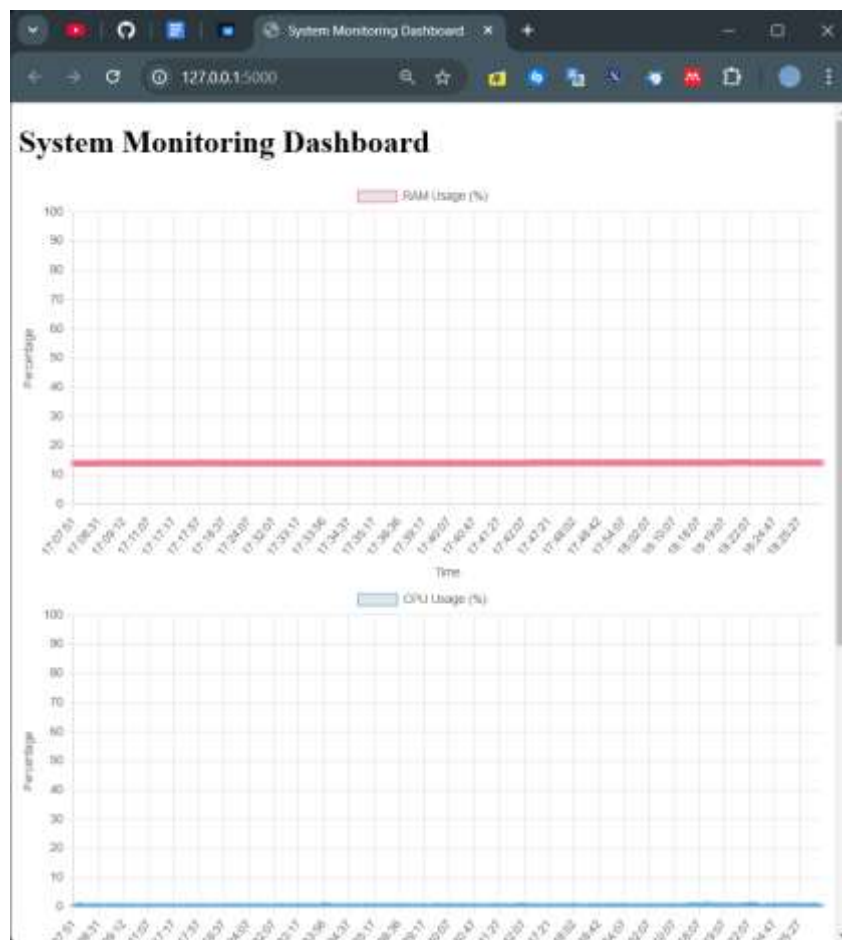
La creación de compose.yml se hizo para orquestar y gestionar multiples contenedores de docker en este caso Puerto 5000 y Puerto 8000

Pregunta 3

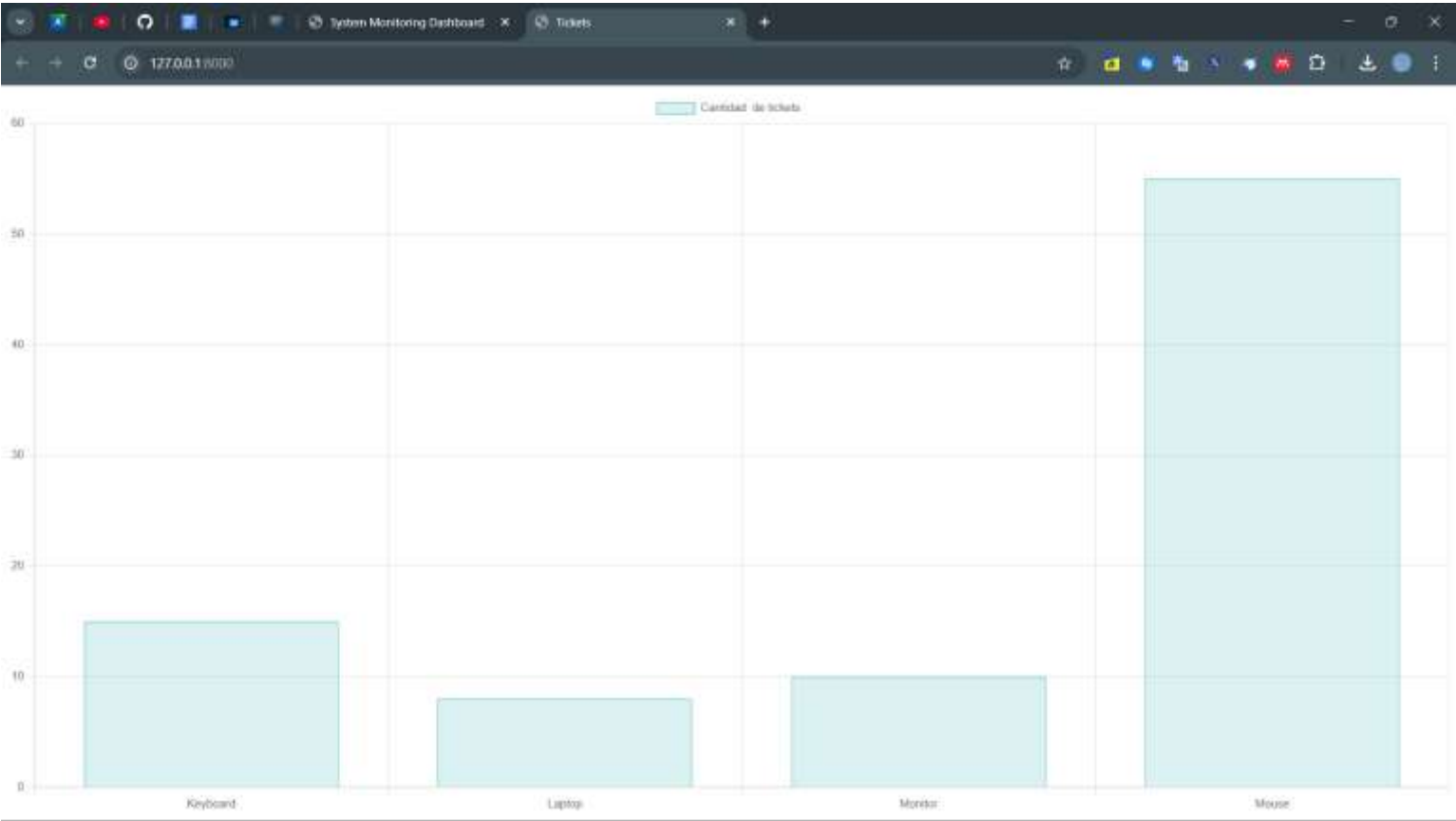


	Name	Container ID	Image	Upgrade	ns
	redis-server	482971d1aed2	redis:latest	Sign out	  
	si657ef202402	-	-	-	0.21   
	app2_container	3792360e53b4	si657ef202402-app: 8000-8000	  	6.01   
	app1_container	5d2a0867d31	si657ef202402-app: 5000-5000	  	0.2   

Puerto: 127.0.0.1:5000 para la visualización del dashboard

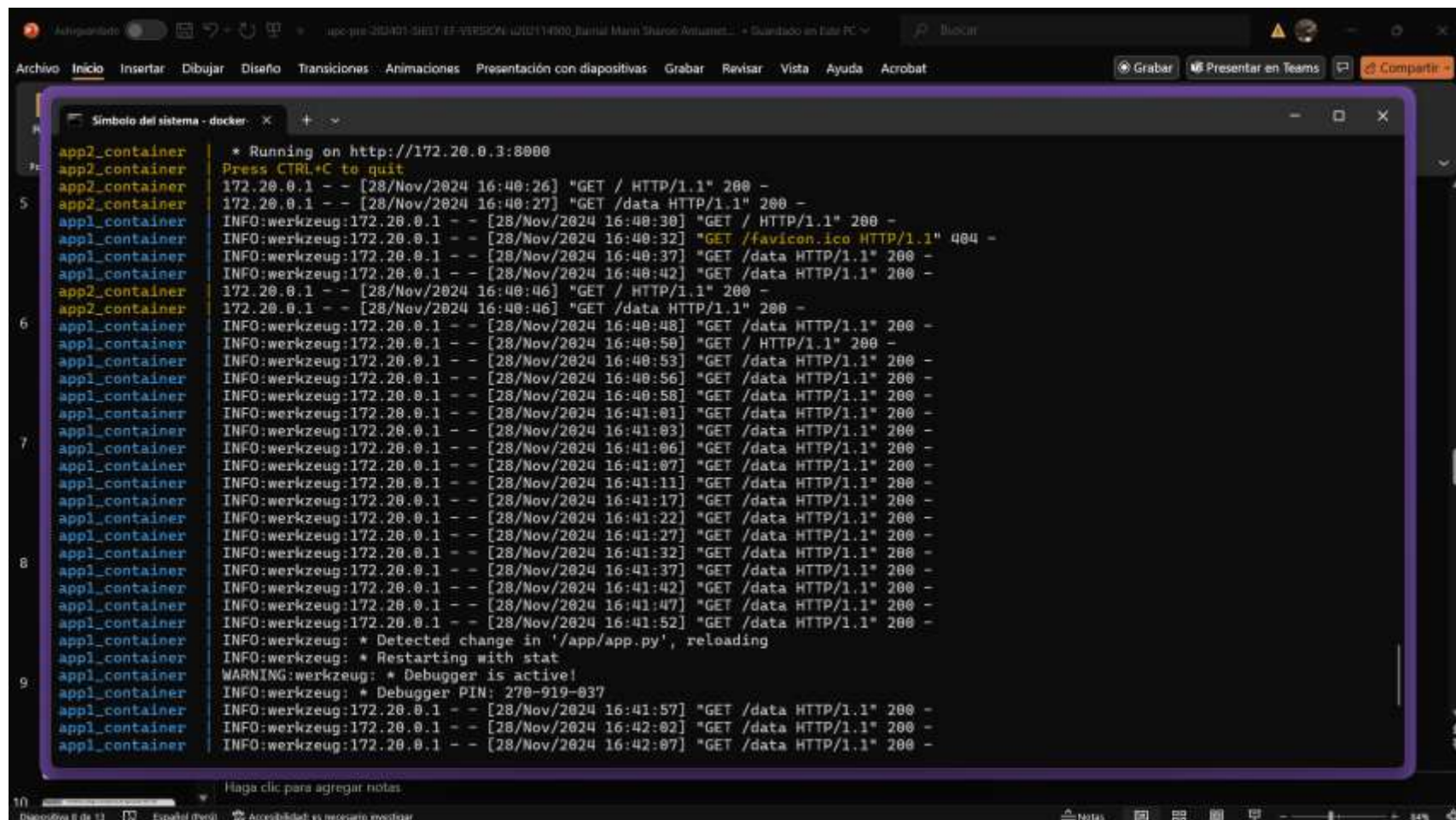


Puerto 127.0.0.1:8000 para la visualización de tickets



Pregunta 3 Verificación de la ejecución DASHBOARD

Vistas de funcionamiento

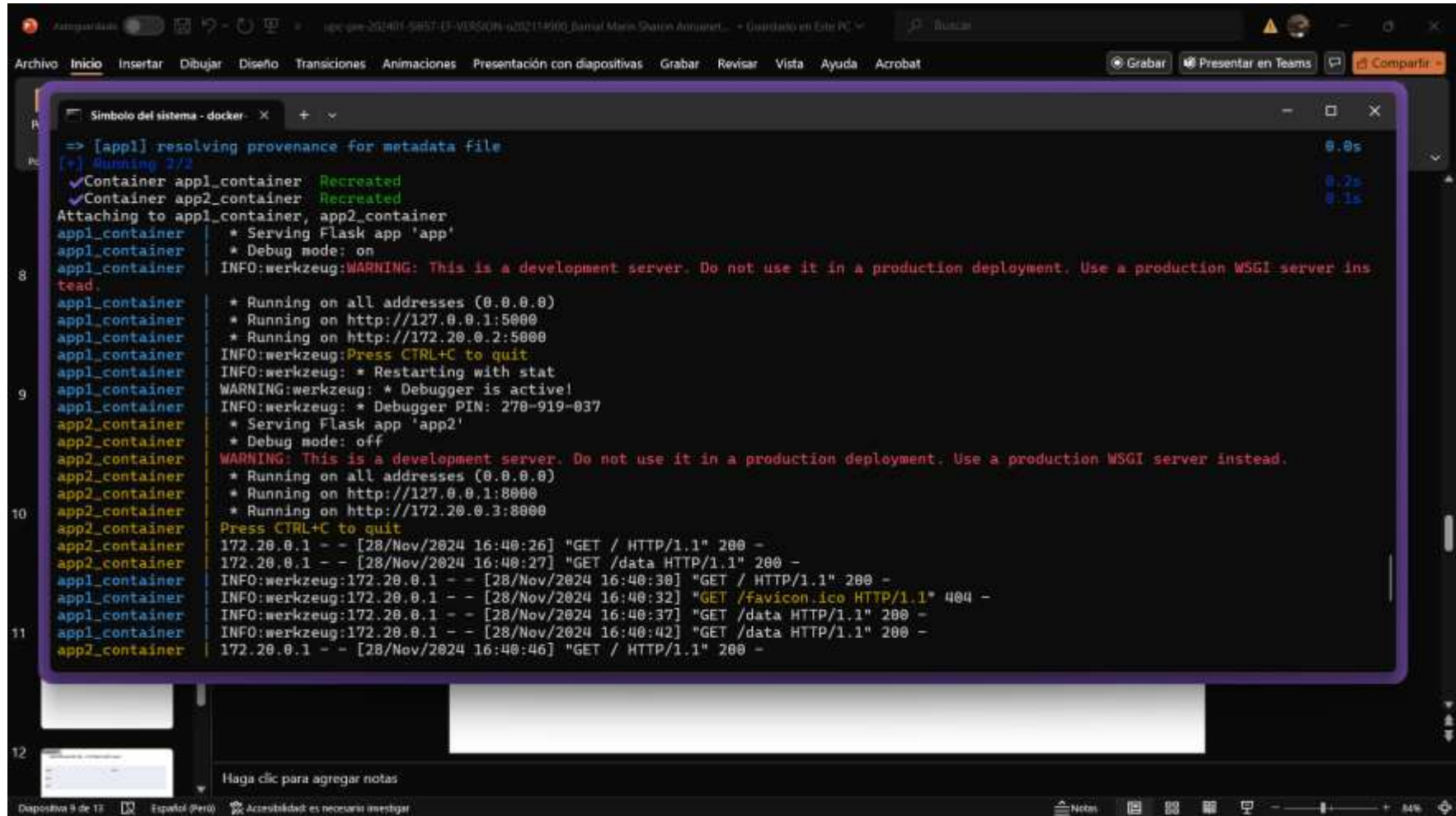


The screenshot shows a Beamer presentation slide titled "Pregunta 3 Verificación de la ejecución DASHBOARD". The slide content is a terminal window titled "Símbolo del sistema - docker" displaying logs for a Docker container named "app2_container". The logs show the container running on http://172.20.0.3:8000 and receiving various HTTP requests. The logs are as follows:

```
* Running on http://172.20.0.3:8000
Press CTRL+C to quit
172.20.0.1 - - [28/Nov/2024 16:40:26] "GET / HTTP/1.1" 200 -
172.20.0.1 - - [28/Nov/2024 16:40:27] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:40:30] "GET / HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:40:32] "GET /favicon.ico HTTP/1.1" 404 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:40:37] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:40:42] "GET /data HTTP/1.1" 200 -
172.20.0.1 - - [28/Nov/2024 16:40:46] "GET / HTTP/1.1" 200 -
172.20.0.1 - - [28/Nov/2024 16:40:46] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:40:48] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:40:50] "GET / HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:40:53] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:40:56] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:40:58] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:41:01] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:41:03] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:41:06] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:41:07] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:41:11] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:41:17] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:41:22] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:41:27] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:41:32] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:41:37] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:41:42] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:41:47] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:41:52] "GET /data HTTP/1.1" 200 -
INFO:werkzeug: * Detected change in '/app/app.py', reloading
INFO:werkzeug: * Restarting with stat
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 270-919-037
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:41:57] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:42:02] "GET /data HTTP/1.1" 200 -
INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:42:07] "GET /data HTTP/1.1" 200 -
```


Pregunta 3 Comandos necesarios para desplegar en Docker

Docker-compose up --build para desplegar en docker



```
Simbolo del sistema - docker - x + v
=> [app1] resolving provenance for metadata file
[+] Running 2/2
✓Container app1_container Recreated
✓Container app2_container Recreated
Attaching to app1_container, app2_container
app1_container | * Serving Flask app 'app'
app1_container | * Debug mode: on
app1_container | INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
app1_container | * Running on all addresses (0.0.0.0)
app1_container | * Running on http://127.0.0.1:5000
app1_container | * Running on http://172.20.0.2:5000
app1_container | INFO:werkzeug:Press CTRL+C to quit
app1_container | INFO:werkzeug: * Restarting with stat
app1_container | WARNING:werkzeug: * Debugger is active!
app1_container | INFO:werkzeug: * Debugger PIN: 270-919-037
app2_container | * Serving Flask app 'app2'
app2_container | * Debug mode: off
app2_container | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
app2_container | * Running on all addresses (0.0.0.0)
app2_container | * Running on http://127.0.0.1:8000
app2_container | * Running on http://172.20.0.3:8000
app2_container | Press CTRL+C to quit
app2_container | 172.20.0.1 - - [28/Nov/2024 16:40:26] "GET / HTTP/1.1" 200 -
app2_container | 172.20.0.1 - - [28/Nov/2024 16:40:27] "GET /data HTTP/1.1" 200 -
app1_container | INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:40:30] "GET / HTTP/1.1" 200 -
app1_container | INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:40:32] "GET /favicon.ico HTTP/1.1" 404 -
app1_container | INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:40:37] "GET /data HTTP/1.1" 200 -
app1_container | INFO:werkzeug:172.20.0.1 - - [28/Nov/2024 16:40:42] "GET /data HTTP/1.1" 200 -
app2_container | 172.20.0.1 - - [28/Nov/2024 16:40:46] "GET / HTTP/1.1" 200 -
```

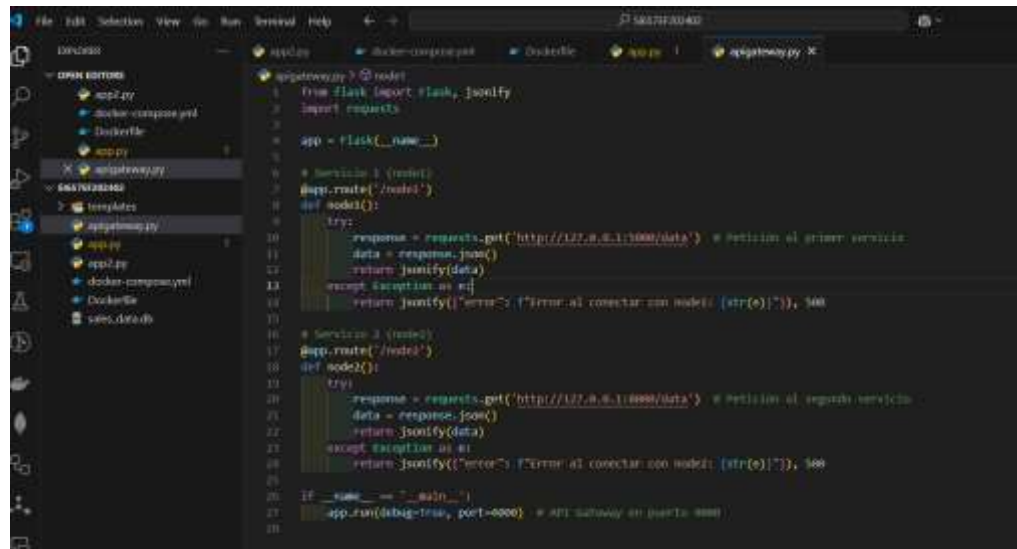
Haga clic para agregar notas

Diapositiva 9 de 13 Español (Perú) Accesibilidad es necesaria investigar

Que uso le daría en el escenario mencionado? Explique brevemente

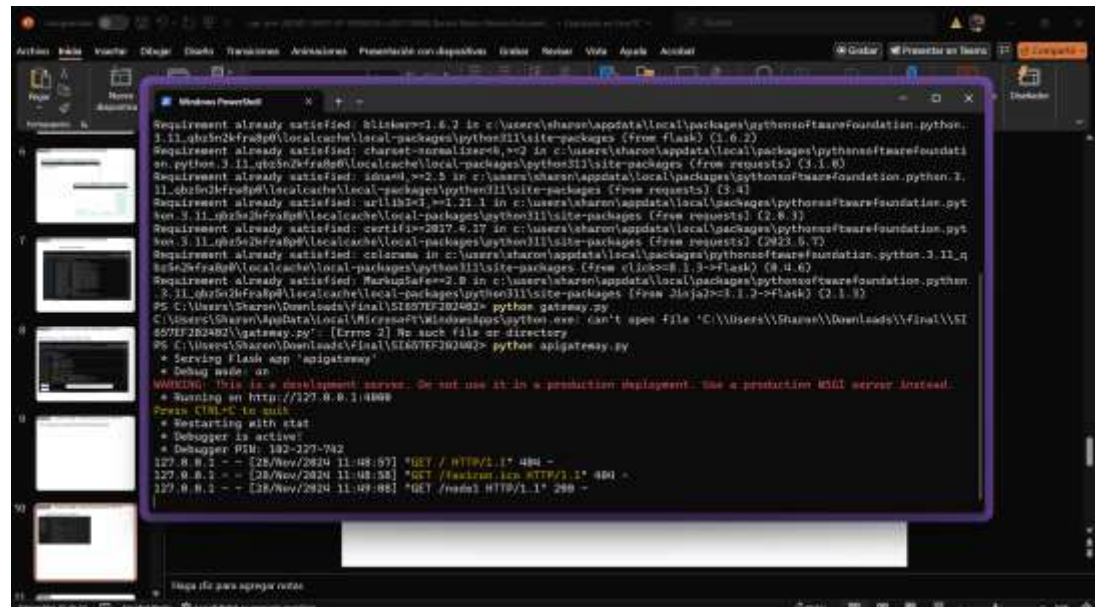
El api Gateway sirve como un punto de acceso único para diferentes servicios. En lugar de que los usuarios tengan que conectar a varios puertos (por ejemplo, 5000 y 8000), el API Gateway recibe todas las peticiones y las dirige al servicio correspondiente. Esto simplifica la comunicación, ya que los usuarios solo interactúan con un único punto. Además, se puede añadir funcionalidades como seguridad y monitoreo en el Gateway sin tener que modificar los servicios individuales. Esto hace que la arquitectura sea más fácil de gestionar y escalar.

Creación un servicio Api gateway



```
1 from flask import flask, jsonify
2 import requests
3
4 app = flask(__name__)
5
6 # Servicio 1 (node)
7 @app.route('/node1')
8 def node1():
9     try:
10         response = requests.get('http://127.0.0.1:3000/data') # Petición al primer servicio
11         data = response.json()
12         return jsonify(data)
13     except Exception as e:
14         return jsonify({"error": f"Error al conectar con node1: {str(e)}"}, 500)
15
16 # Servicio 2 (node)
17 @app.route('/node2')
18 def node2():
19     try:
20         response = requests.get('http://127.0.0.1:3000/data') # Petición al segundo servicio
21         data = response.json()
22         return jsonify(data)
23     except Exception as e:
24         return jsonify({"error": f"Error al conectar con node2: {str(e)}"}, 500)
25
26 if __name__ == '__main__':
27     app.run(debug=True, port=8000) # API Gateway en puerto 8000
28
```

Corriendo el servicio apigateway



```
Requirement already satisfied: blinker<1.6.2 in c:\users\sharon\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfr8q6\localcache\local-packages\python311\site-packages (from flask) (1.6.2)
Requirement already satisfied: charset-normalizer<4,=>2 in c:\users\sharon\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfr8q6\localcache\local-packages\python311\site-packages (from requests) (3.1.0)
Requirement already satisfied: idna<=2.5 in c:\users\sharon\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfr8q6\localcache\local-packages\python311\site-packages (from requests) (3.4)
Requirement already satisfied: urllib3<3,=>1.21.1 in c:\users\sharon\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfr8q6\localcache\local-packages\python311\site-packages (from requests) (2.0.3)
Requirement already satisfied: certifi<=2023.6.17 in c:\users\sharon\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfr8q6\localcache\local-packages\python311\site-packages (from requests) (2023.6.17)
Requirement already satisfied: click==8.1.3--flask in c:\users\sharon\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfr8q6\localcache\local-packages\python311\site-packages (from flask) (8.1.3)
Requirement already satisfied: MarkupSafe<2.0 in c:\users\sharon\appdata\local\packages\pythonsoftwarefoundation.python.3.11.qbz5n2kfr8q6\localcache\local-packages\python311\site-packages (from flask) (2.1.0)
PS C:\Users\Sharon\Downloads\Final\SI657EF202402> python gateway.py
PS C:\Users\Sharon\Downloads\Final\SI657EF202402> python apigateway.py
* Serving Flask app 'apigateway'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:8000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 182-297-792
127.0.0.1 - - [28/Nov/2024 11:48:57] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [28/Nov/2024 11:48:58] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [28/Nov/2024 11:49:08] "GET /node1 HTTP/1.1" 200 -
```

Prueba de los edpoints tanto en Boomerang como en la web

The screenshot shows the Boomerang extension interface. The top browser window displays a JSON response from the endpoint `127.0.0.1:4000/node1`. The response is:

```
{  "cpu": 0.1,  "disk": 0.2,  "ram": 14.1,  "time": "17:00:26"}
```

The bottom panel shows the request details for `GET http://127.0.0.1:4000/node1`. The response body is displayed in raw format:

```
1 { 2   "cpu": 0.1, 3   "disk": 0.2, 4   "ram": 14.1, 5   "time": "17:01:56" 6 }
```

The response took 1015 ms and returned 68 bytes.

The screenshot shows the Boomerang extension interface. The top browser window displays a JSON response from the endpoint `127.0.0.1:5000/data`. The response is:

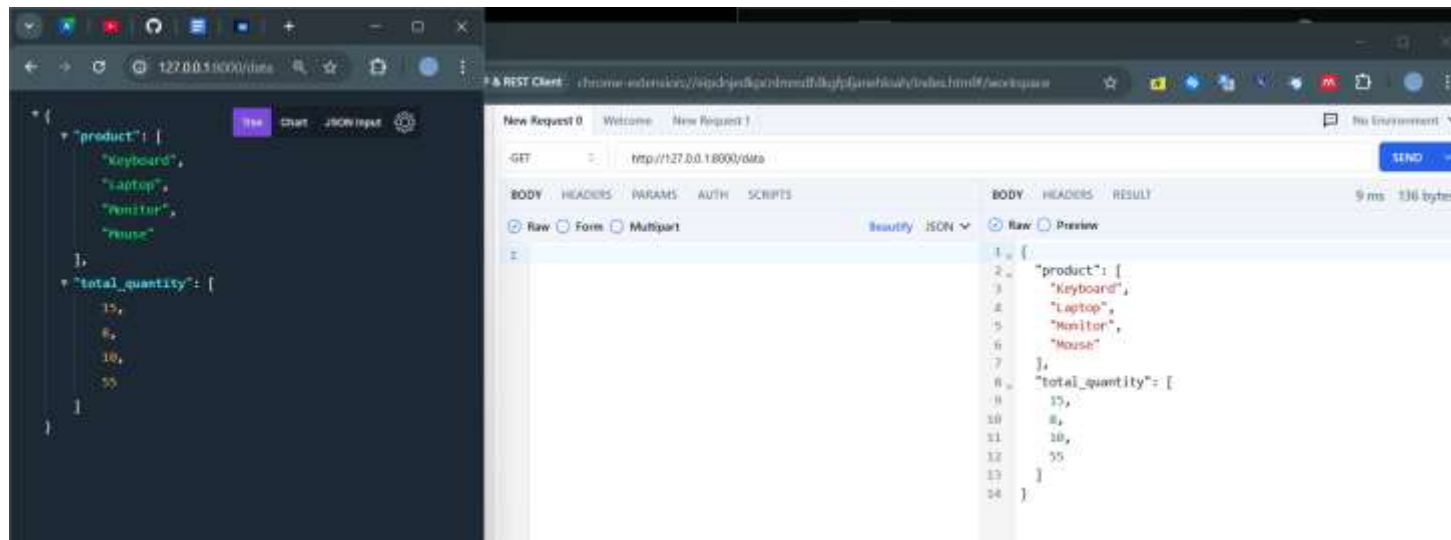
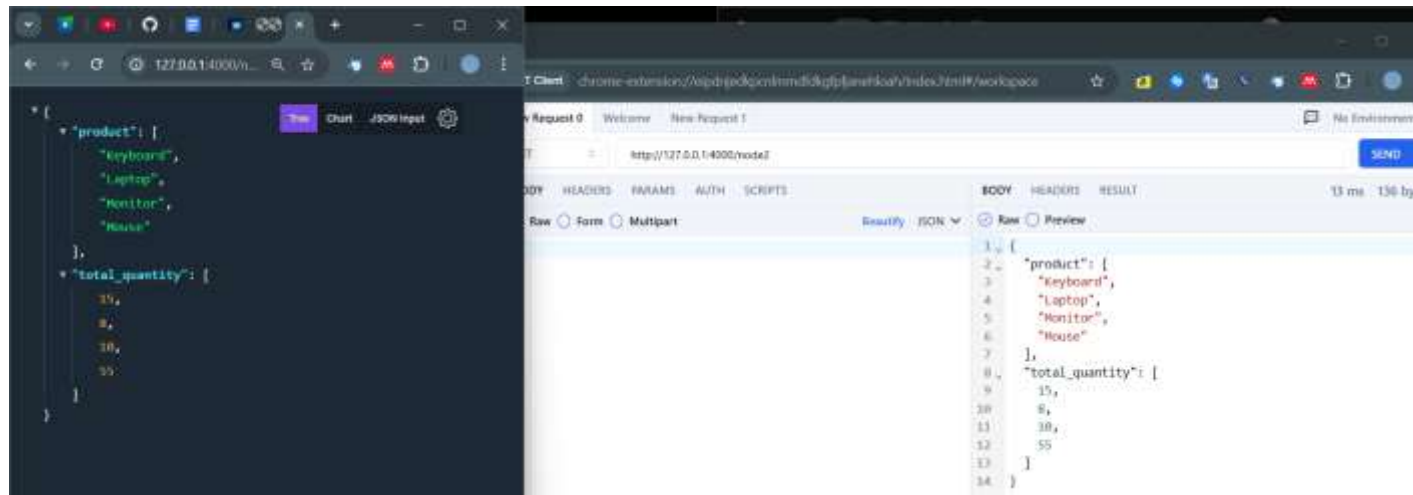
```
{  "cpu": 0.4,  "disk": 0.2,  "ram": 14.1,  "time": "17:00:52"}
```

The bottom panel shows the request details for `GET http://127.0.0.1:5000/data`. The response body is displayed in raw format:

```
1 { 2   "cpu": 0.2, 3   "disk": 0.2, 4   "ram": 14.2, 5   "time": "17:02:28" 6 }
```

The response took 1012 ms and returned 68 bytes.

Ejecución tanto en Boomerang como en la web



Identificación de 2 drivers del caso

Código	Drivers
R_001	Seguridad
R_002	Escalabilidad

<p>Tactica</p> <p>Cifrado y autenticación multifactor</p>	<p>Artefacto y Descripción de la táctica</p> <p>Al usar WebSockets para la comunicación, se garantiza que los datos estén cifrados tanto cuando se envían como cuando se guardan, protegiendo la información. Además, el sistema cuenta con autenticación de múltiples pasos para asegurarse de que solo los usuarios correctos puedan acceder al sistema.</p>
<p>Escalabilidad horizontal y microservicios</p>	<p>El sistema debe ser escalable para manejar un número creciente de usuarios y actividad. Para ello, se utilizará Prometheus y ELK Stack para monitorear el rendimiento y los logs del sistema.</p>