	<p>UNIVERSIDAD NACIONAL DEL ALTIPLANO ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS</p>	
<p>CURSO: SISTEMAS DE INFORMACION</p>		
<p>Fecha: 16-04-24</p>	<p>M.Sc. Marga Isabel Ingaluque Arapa</p>	<p>Página: 1</p>

## INFORME DE LABORATORIO

### (formato estudiante)

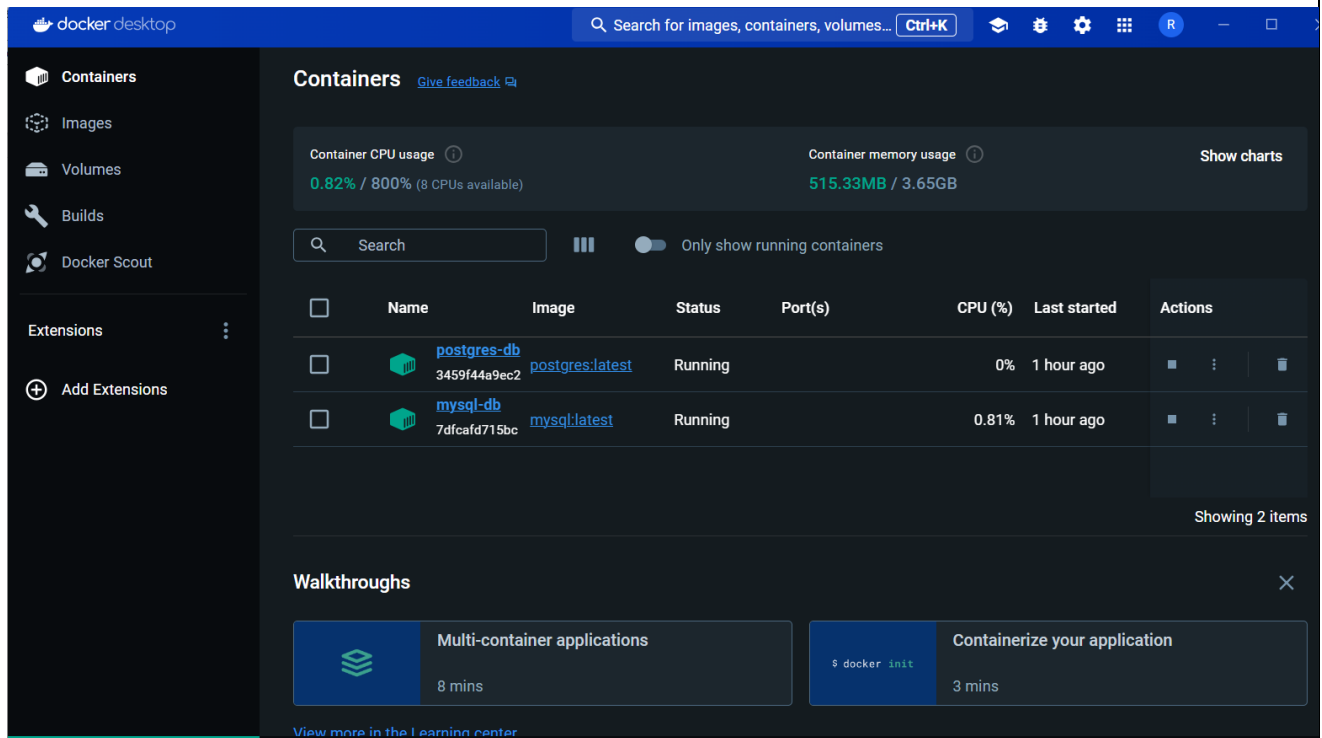
<p><b>INTEGRANTE (s): ARACAYO MAMANI JHON MARCO</b></p>	<p align="center"><b>NOTA:</b></p>	
---	------------------------------------	--

INFORMACIÓN BÁSICA					
<p><b>TÍTULO DE LA PRÁCTICA:</b></p>	<p><b>Integración de Sistemas y Flujos de Trabajo</b></p>				
<p><b>NÚMERO DE PRÁCTICA:</b></p>	<p align="center">8</p>	<p><b>AÑO LECTIVO:</b></p>	<p align="center">2024</p>	<p><b>NRO. SEMESTRE:</b></p>	<p align="center">VI</p>
<p><b>FECHA DE PRESENTACIÓN</b></p>	<p align="center">22/06/24</p>				

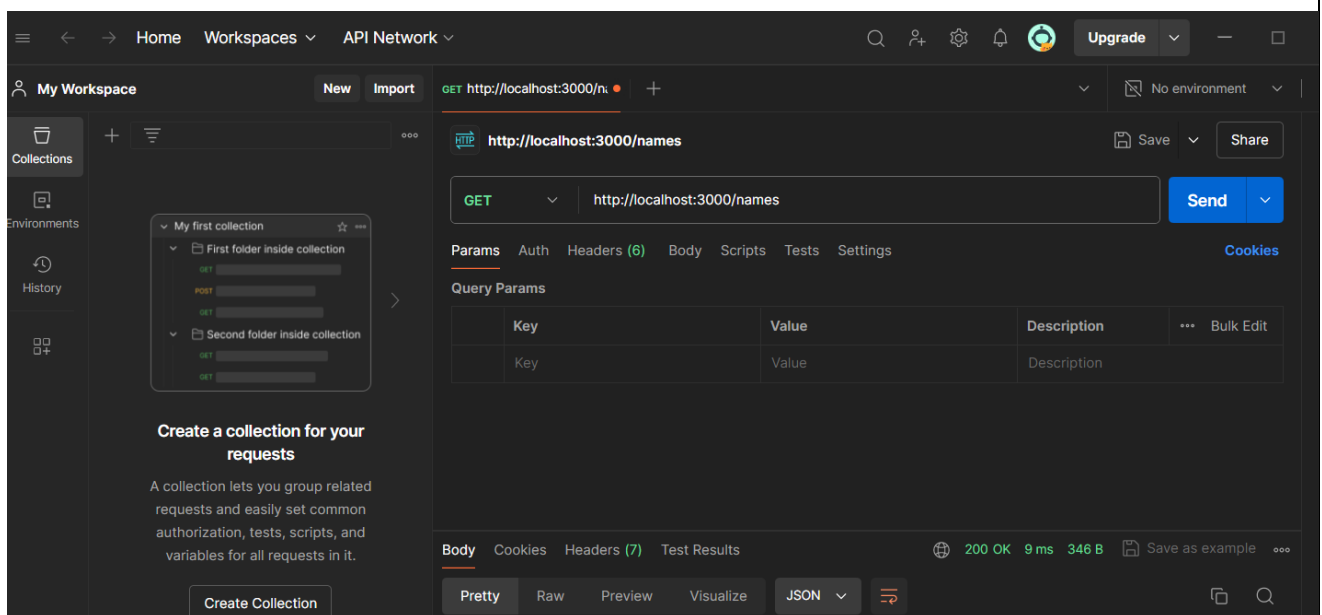
SOLUCIÓN Y RESULTADOS	
<p><b>Parte 1: Configuración del Entorno (30 minutos)</b></p> <p><b>1.- Instalacion del docker</b></p> <p><b>Instalamos el docker correctamente y lo inicializacmos con los comandos correspondientes:</b></p> <pre>docker run --name mysql-db -e MYSQL_ROOT_PASSWORD=root -d mysql:latest docker run --name postgres-db -e POSTGRES_PASSWORD=root -d postgres:latest</pre>	



## I.



## 2.- Instalación de Postman:





## II. Creación y Consumo de APIs (

En esta ocasión usaremos node.js para crear una API básica que pueda interactuar con la base de datos, iniciando dando los datos de mi base de datos.

```
const express = require('express');
const mysql = require('mysql');

const app = express();

// Configurar la conexión a la base de datos MySQL
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'usuarios'
});

db.connect((err) => {
  if (err) {
    console.error('Error connecting to MySQL:', err);
    return;
  }
  console.log('Connected to MySQL database');
});

// Definir una ruta para obtener datos desde la base de datos
app.get('/names', (req, res) => {
  db.query('SELECT * FROM names', (err, results) => {
    if (err) {
      console.error('Error executing query:', err);
      res.status(500).json({ error: 'Failed to retrieve data', details:
err.message });
      return;
    }
    res.json(results); // Devolver los resultados en formato JSON
  });
});

app.listen(3000, () => { // Iniciar el servidor en el puerto 3000
  console.log('Server started on port 3000');
});
```



### III. Probamos la API en el postman con nuestros datos ingresados:

Servidor: 127.0.0.1 » Base de datos: usuarios » Tabla: names

[Examinar](#) [Estructura](#) [SQL](#) [Buscar](#) [Insertar](#) [Exportar](#)

✓ Mostrando filas 0 - 1 (total de 2, La consulta tardó 0,0004 segundos.)

```
SELECT * FROM `names`
```

☐ Perfilando [ [Editar en línea](#) ] [ [Editar](#) ] [ [Explicar SQL](#) ] [ [Crear código PHP](#) ] [ [Actualizar](#) ]

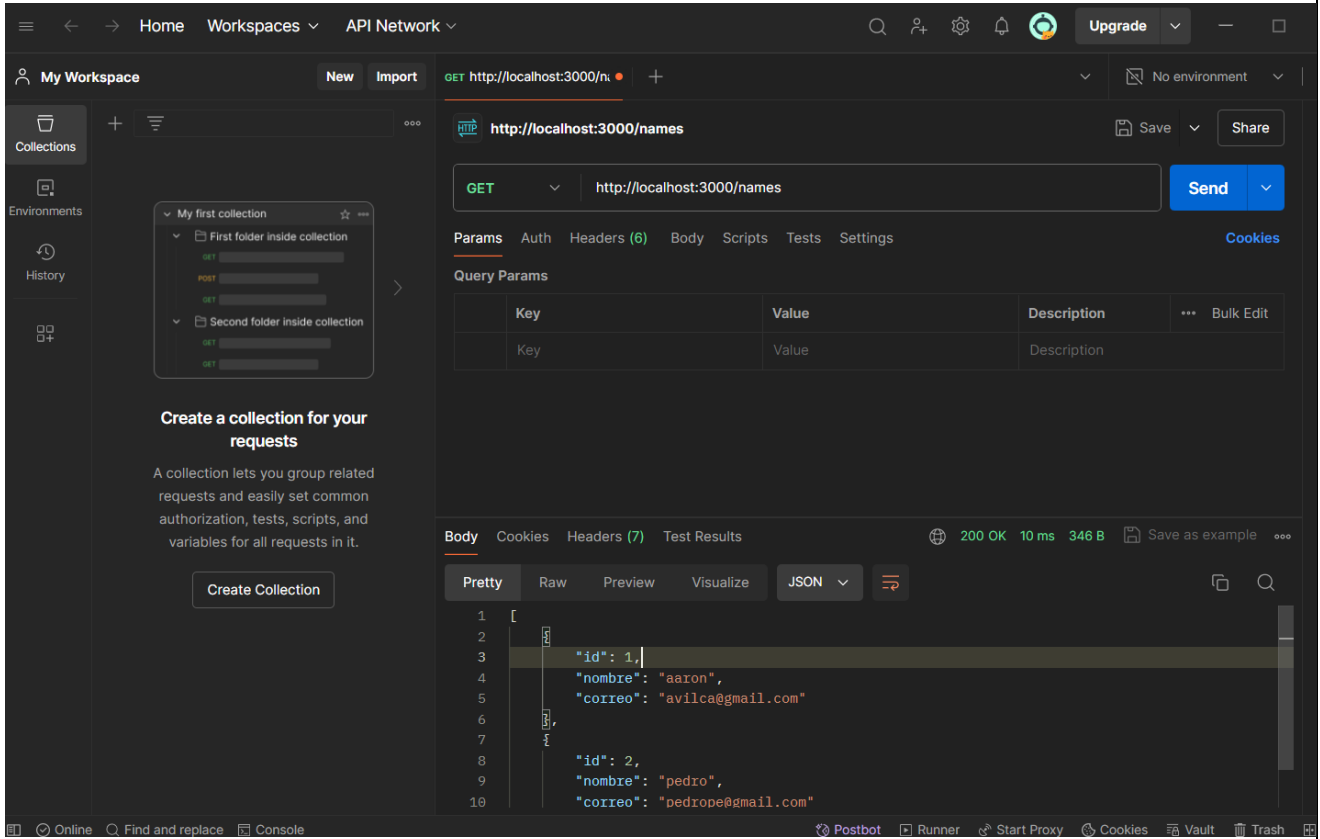
☐ Mostrar todo | Número de filas: 25 ▼ Filtrar filas:

Opciones extra

				id	nombre	correo			
<input type="checkbox"/>		Editar		Copiar		Borrar	1	aaron	avilca@gmail.com
<input type="checkbox"/>		Editar		Copiar		Borrar	2	pedro	pedrope@gmail.com



### Corroboramos con el postman que nuestra API funcione correctamente:



### IV. Implementacion de un flujo de trabajo basico:

#### - Ejemplo de codigo de automatizacion de tareas

```
import requests
import csv
import time

while True: # Bucle infinito para ejecutar la tarea repetidamente
    try:
        response = requests.get('http://localhost:3000/names') # Hacer una solicitud
        # GET a La API
        response.raise_for_status() # Verificar si la solicitud fue exitosa (código de
        # estado 200)
        try:
            data = response.json() # Obtener Los datos en formato JSON
        except requests.exceptions.JSONDecodeError:
            print("Error: No se pudo decodificar la respuesta JSON.")
            data = []

        if data: # Verificar si data no está vacío
```



```
with open('data.csv', mode='w', newline='') as file: # Abrir un archivo
CSV en modo escritura
    writer = csv.writer(file) # Crear un escritor de CSV
    # Escribir los encabezados de las columnas
    writer.writerow(['id', 'nombre', 'correo'])
    # Iterar sobre los datos obtenidos y escribir cada fila en el archivo
CSV
    for row in data:
        writer.writerow([row['id'], row['nombre'], row['correo']])
    print("Datos guardados correctamente en data.csv")
else:
    print("No se recibieron datos válidos de la API.")
except requests.exceptions.RequestException as e:
    print(f"Error en la solicitud HTTP: {e}")

time.sleep(3600) # Esperar una hora antes de la siguiente ejecución
```

## V. Evaluacion y documentacion

Gracias al postman sabemos que nuestra API funciona correctamente, ahora ejecutaremos el código python de automatización para crear un archivo .CSV que guarde los datos que obtenga de la API:

```
data.csv
1 id,nombre,correo
2 1,aaron,avilca@gmail.com
3 2,pedro,pedrope@gmail.com
4
```

```
localhost:3000/names
Facebook - Inicia se... YouTube Aula Virtual | UNAP Intranet | UNAP Trovo Coding Blocks IDE Online C++ Cor
Impresión con formato estilístico
[{"id":1,"nombre":"aaron","correo":"avilca@gmail.com"}, {"id":2,"nombre":"pedro","correo":"pedrope@gmail.com"}]
```

VI.



Link del repositorio del proyecto:

<https://github.com/tamoi13/practica8.git>

VII.

API-BDD Public

master had recent pushes 33 seconds ago [Compare & pull request](#)

master 2 Branches 0 Tags  [Add file](#) [Code](#)

This branch is 1 commit ahead of, 1 commit behind main . [Contribute](#)

rogervc1 final a617bdc · 2 minutes ago 1 Commit

node_modules	final	2 minutes ago
automatizacion.py	final	2 minutes ago
data.csv	final	2 minutes ago
gaaa.js	final	2 minutes ago
package-lock.json	final	2 minutes ago
package.json	final	2 minutes ago

[README](#)

About  
No description, website

Activity  
0 stars  
1 watching  
0 forks

Releases  
No releases published  
[Create a new release](#)

Packages  
No packages published  
[Publish your first package](#)

## REFERENCIAS Y BIBLIOGRAFÍA