Proyecto AS

Diego Marta Hurtado

03/12/2023

1. INTRODUCCIÓN

En la segunda parte evaluativa de la asignatura de Administración de Sistemas (AS), se nos ha solicitado llevar a cabo un proyecto en el cual aplicaremos los conocimientos adquiridos en la asignatura. El objetivo es gestionar y orquestar un conjunto de contenedores.

Se nos pide utilizar una imagen de servidor y otra de cliente, y además utilizar una imagen que utilizaremos de base de datos. Todas las imágenes que utilicemos deben ser imágenes de Docker Hub.

Por otro lado, deberemos incluir otras funcionalidades a la aplicación en forma de contenedor y generar un despliegue conjunto con docker-compose.

2. LOS CONTENEDORES

2.1 Mysql

MySQL es un sistema de gestión de bases de datos relacional (RDBMS, por sus siglas en inglés) de código abierto que utiliza el lenguaje de consulta estructurado (SQL)

2.2 Aplicación cliente

La aplicación cliente nos conectara con la base de datos Mysql para almacenar los datos necesarios y que esta se visualice en el frontend.

Se ha utilizado un lenguaje Python que permite, por un lado, recibir peticiones mediante la librería *Flask* y, por otro lado, realizar operaciones SQL en nuestra base de datos mediante el paquete *mysql.connector*. La aplicación quedará estará almacenada en el archivo main.py que se muestra a continuación:

```
import mysql.connector
from flask import Flask, request, jsonify
from flask_cors import CORS
from flask import send_from_directory
import time

print("Iniciando la aplicación...")
```

```
app = Flask(__name__)
CORS(app)

@app.route('/')
        def home():

                return "¡Bienvenido a mi aplicación Flask!"



@app.route('/favicon.ico')
        def favicon():

                return jsonify({"message": "¡Favicon no encontrado!"})

@app.route('/commands/', methods=['GET'])
        def comandas_data():
                connection= mysql.connector.connect(
                        host="10.5.0.4",
                        user="DiegoAS",
                        password="ProyectoAS2324",
                        database="ProyectoAS2324Restaurante",
                        port=3306)

                if connection:
                        cursor = connection.cursor()
                        cursor.execute("select table_name from INFORMATION_SCHEMA.tables
            where table_name = 'Comandas ' limit 50")
                        exists = cursor.fetchone()

                        if exists:
                                connection.close()
                                return jsonify({"message": "La tabla Comandas ya existe"})
                        else:
                                cursor.execute("CREATE TABLE Comandas (id INT NOT NULL
                                        AUTO_INCREMENT, primerPlato
                                VARCHAR(255) NOT NULL, segundoPlato
                VARCHAR(255) NOT NULL, nMesa int,
        PRIMARY KEY (id))")
                                cursor.execute("INSERT INTO Comandas (primerPlato,
                                segundoPlato, nMesa) VALUES ('Paella de
        marisco', 'Ensalada mixta', 1)")
                                cursor.execute("INSERT INTO Comandas (primerPlato,
                                segundoPlato, nMesa) VALUES ('Arroz con
                bogavante' , 'Ensalada mixta', 3)")
                                cursor.execute("INSERT INTO Comandas (primerPlato,
                                segundoPlato, nMesa) VALUES ('Alubias
                pintas' , 'Chuleton a la brasa', 4)")
                                cursor.execute("INSERT INTO Comandas (primerPlato,
                                segundoPlato, nMesa) VALUES ('Tortilla de
                bacalado' , 'Lubina a la plancha', 2)")
                                connection.commit()
```

```
                    time.sleep(1)
                    cursor.execute("SELECT * FROM Comandas")
                    comandas = cursor.fetchall()
                    connection.close()
                    retorno = {"lComandas": []}
                    for t in comandas:
                        retorno["lComandas"].append({"id":t[0],"primerPlato":
                t[1], "segundoPlato": t[2], "nMesa": t[3]})

                    return jsonify(retorno)

@app.route('/nuevacomanda', methods=['POST'])
    def nuevaComanda():
        primerPlato = request.json['primerPlato']
        segundoPlato = request.json['segundoPlato']
        nMesa = request.json['nMesa']


        connection = mysql.connector.connect(
            host="10.5.0.4",
            user="DiegoAS",
            password="ProyectoAS2324",
            database="ProyectoAS2324Restaurante",
            port=3306)

        if connection:
            cursor = connection.cursor()
            cursor.execute("INSERT INTO Comandas (primerPlato, segundoPlato,
        nMesa) VALUES (%s, %s, %s)", (primerPlato,
    segundoPlato, nMesa))
            connection.commit()
            connection.close()

            return jsonify({"message": "Comanda creada con exito"})
        else:
            return jsonify({"message": "No se pudo conectar con la base de
        datos"})


app.run(host='0.0.0.0', port=4201)
```

Code 1: main.py

La aplicación cuenta con dos peticiones diferentes, una con método GET y otra con POST con diferentes funcionalidades:

- */commands: se ejecuta cuando carga el frontend. Crea la tabla en la base de datos en caso de que no exista y carga los datos almacenados en el volumen. A continuación, se realiza una operación de SELECT mediante la cual, se recuperan los datos que después se mostrarán en el frontend.*

- /nuevacomanda:recupera los datos del frontend y, mediante una operación de INSERT, los añade a la base de datos.

La aplicación cliente se ha creado el siguiente Dockerfile, cuyo repositorio en Docker Hub es **https://hub.docker.com/repository/docker/dmarta14/proyectoas/general**

```
FROM alpine

RUN apk update && \
apk add --no-cache python3 py3-pip mariadb-dev build-base python3-dev

RUN pip3 install --upgrade pip && \
pip3 install mysql-connector-python flask flask_cors docker

WORKDIR /cliente
COPY ./main.py .

CMD ["python3", "main.py"]
```

Code 2 : Dockerfile

2.3 Frontend

La tercera aplicación desarrollada es un servidor web, implementado mediante la imagen Docker de httpd. En esta aplicación, se incluye un archivo HTML simple que contiene tres etiquetas de tipo input, las cuales permiten la entrada de información. La información ingresada se procesará a través de un archivo JavaScript, el cual realizará las solicitudes correspondientes al backend (aplicación cliente) para almacenar o mostrar la información.

3 Archivo docker-compose

```
version: "3.8"

services:
    mysqldb:
        image: mysql
        environment:
            - MYSQL_ROOT_PASSWORD=ProyectoAS2324
            - MYSQL_DATABASE=ProyectoAS2324Restaurante
```

```
            - MYSQL_USER=DiegoAS
            - MYSQL_PASSWORD=ProyectoAS2324
        ports:
            - 3310:3306
        networks:
            proyectoas_mired:
                ipv4_address: 10.5.0.4
        volumes:
            - ./mysql-data:/var/lib/mysql

    cliente:
        build: .
        depends_on:
            - mysqldb
        links:
            - mysqldb
        ports:
            - 4201:4201
        networks:
            proyectoas_mired:
                ipv4_address: 10.5.0.5

    web-httpd:
        image: httpd
        volumes:
            - ./public:/usr/local/apache2/htdocs
        ports:
            - 8080:80
        networks:
            proyectoas_mired:
                ipv4_address: 10.5.0.6
        depends_on:
            - cliente

networks:
    proyectoas_mired:
        driver: bridge
        ipam:
            driver: default
            config:
                - subnet: 10.5.0.0/16
                  gateway: 10.5.0.1
```

Code 3: docker-compose.yml

4. Kubernets

NO he podido desarrollar este apartado por falta de organización a la hora de desarrollar el proyecto.

5. Link a repositorio gitHub

https://github.com/Dmarta14/ProyectoAS.git