



---

# DESPLIEGUE DE UNA APLICACIÓN MULTI-CONTENEDOR CON DOCKER COMPOSE

---

Realizado por: Miguel Gutiérrez García y Adrian Dumitru



19 DE ABRIL DE 2025

# índice

Objetivo del trabajo.....	2
Creación de la máquina virtual.....	2
Instalación de docker.....	3
Instalación docker compose y permisos .....	4
Docker compose .....	6
Despliegue de los contenedores .....	7
Modificación de la Aplicación Web para Conexión a la Base de Datos .....	9
Prueba Final: Envío y Almacenamiento de Datos .....	13
Conclusiones .....	15

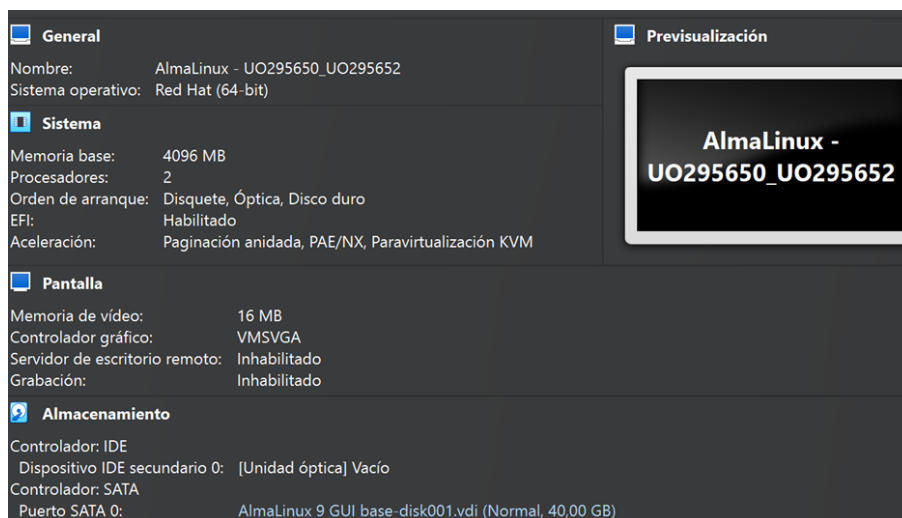
## Objetivo del trabajo

Los objetivos principales de este trabajo son:

- Aprender a usar Docker Compose para definir y lanzar aplicaciones con múltiples contenedores.
- Saber crear archivos docker-compose.yml para configurar los servicios, redes y volúmenes.
- Gestionar la comunicación y los datos entre los contenedores de la aplicación.
- Manejar el ciclo de vida de la aplicación (iniciar, parar, ver logs) con los comandos de docker-compose.
- Entender su utilidad para crear entornos de desarrollo consistentes.

## Creación de la máquina virtual

Todo el trabajo práctico se realizó dentro de una máquina virtual creada con Oracle VirtualBox. Utilizamos AlmaLinux 9 como sistema operativo invitado, asignándole 4GB de RAM y 2 CPUs. Esta VM nos proporcionó el entorno Linux aislado y controlado necesario para instalar Docker/Docker Compose y llevar a cabo todo el desarrollo, despliegue y pruebas de la aplicación multi-contenedor.



## Instalación de docker

Una vez que tuvimos nuestra máquina virtual AlmaLinux preparada y funcionando, el primer paso fundamental para nuestro proyecto fue instalar Docker Engine.

Docker es la plataforma de contenedores sobre la que se basa Docker Compose y toda nuestra aplicación.

Comandos para instalar docker:

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

```
sudo yum-config-manager --add-repo
```

```
https://download.docker.com/linux/centos/docker-ce.repo
```

```
sudo yum install -y docker-ce docker-ce-cli containerd.io
```

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

Docker corriendo:

```
AlmaLinux - UO295650_UO295652 [Corriendo] - Oracle VirtualBox
root@vbox ~]# sudo systemctl status docker
docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: disabled)
   Active: active (running) since Sat 2025-04-19 16:14:14 CEST; 1min 36s ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 4452 (dockerd)
       Tasks: 8
      Memory: 23.7M
         CPU: 379ms
    CGroup: /system.slice/docker.service
           └─4452 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

br 19 16:14:11 vbox dockerd[4452]: time="2025-04-19T16:14:11.815110602+02:00" level=info
br 19 16:14:11 vbox dockerd[4452]: time="2025-04-19T16:14:11.914871847+02:00" level=info
br 19 16:14:11 vbox dockerd[4452]: time="2025-04-19T16:14:11.985714327+02:00" level=info
br 19 16:14:14 vbox dockerd[4452]: time="2025-04-19T16:14:14.362620017+02:00" level=info
br 19 16:14:14 vbox dockerd[4452]: time="2025-04-19T16:14:14.423711128+02:00" level=info
br 19 16:14:14 vbox dockerd[4452]: time="2025-04-19T16:14:14.423972018+02:00" level=info
br 19 16:14:14 vbox dockerd[4452]: time="2025-04-19T16:14:14.522777731+02:00" level=info
br 19 16:14:14 vbox dockerd[4452]: time="2025-04-19T16:14:14.532944606+02:00" level=info
br 19 16:14:14 vbox dockerd[4452]: time="2025-04-19T16:14:14.534494666+02:00" level=info
br 19 16:14:14 vbox systemd[1]: Started Docker Application Container Engine.
lines 1-22/22 (END)
```

## Instalación docker compose y permisos

Después de instalar Docker Engine, procedimos a instalar Docker Compose descargando directamente el binario de la versión 1.27.4 desde GitHub y dándole permisos de ejecución, lo cual verificamos con `docker-compose --version`. A continuación, creamos la estructura de directorios base para nuestro proyecto (`red/sitio_web/`) y generamos los archivos iniciales `index.html` y `script.js`, que contenían una página estática simple con un script para una primera prueba de despliegue.

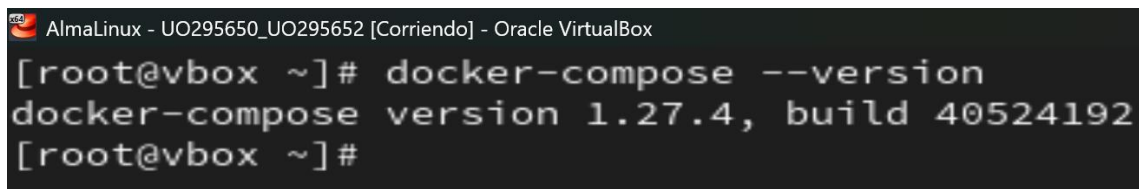
Comandos utilizados:

```
sudo curl -L
```

```
"https://github.com/docker/compose/releases/download/1.27.4/docker-  
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

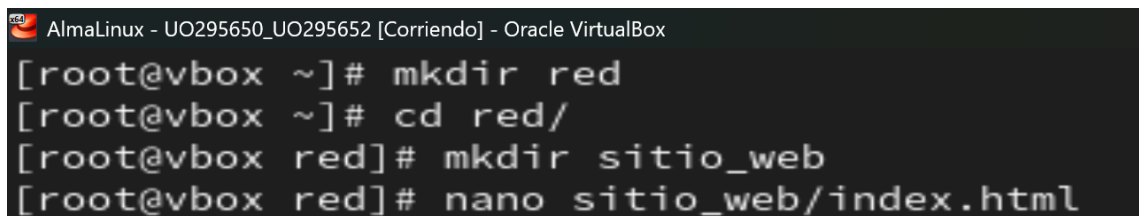
```
sudo chmod +x /usr/local/bin/docker-compose
```

Instalado:



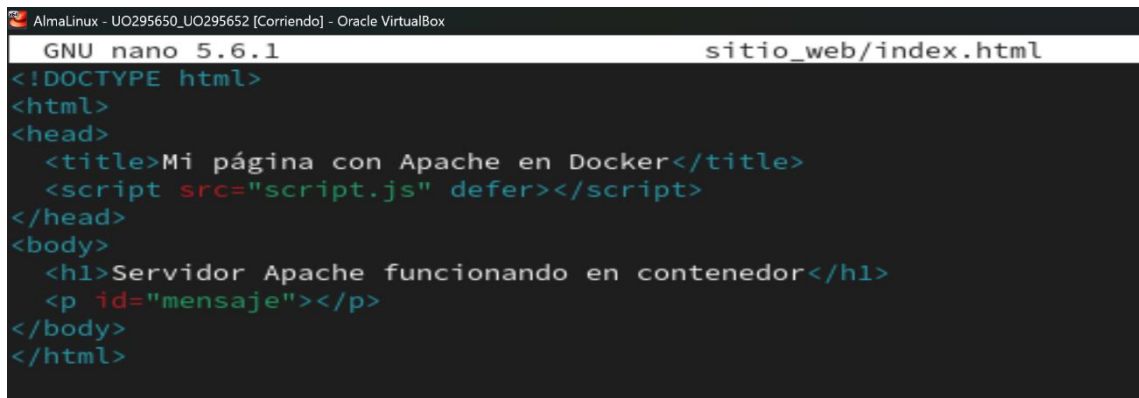
```
AlmaLinux - UO295650_UO295652 [Corriendo] - Oracle VirtualBox  
[root@vbox ~]# docker-compose --version  
docker-compose version 1.27.4, build 40524192  
[root@vbox ~]#
```

Creacion de index para el servicio web /(que esta en apache)



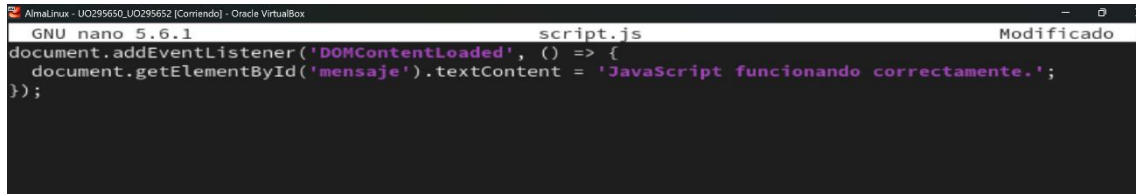
```
AlmaLinux - UO295650_UO295652 [Corriendo] - Oracle VirtualBox  
[root@vbox ~]# mkdir red  
[root@vbox ~]# cd red/  
[root@vbox red]# mkdir sitio_web  
[root@vbox red]# nano sitio_web/index.html
```

Codigo:



```
AlmaLinux - UO295650_UO295652 [Corriendo] - Oracle VirtualBox  
GNU nano 5.6.1 sitio_web/index.html  
<!DOCTYPE html>  
<html>  
<head>  
  <title>Mi página con Apache en Docker</title>  
  <script src="script.js" defer></script>  
</head>  
<body>  
  <h1>Servidor Apache funcionando en contenedor</h1>  
  <p id="mensaje"></p>  
</body>  
</html>
```

Script.js:



```
GNU nano 5.6.1 script.js Modificado
document.addEventListener('DOMContentLoaded', () => {
  document.getElementById('mensaje').textContent = 'JavaScript funcionando correctamente.';
});
```

Explicación del script:

1. Espera a que se cargue todo el contenido HTML (DOMContentLoaded)
  - a. Así se asegura de que los elementos del DOM existen antes de intentar modificarlos.
2. Busca el elemento con ID mensaje
  - a. Este es el `<p id="mensaje"></p>` del index.html.
3. Le cambia el texto interno
  - a. Sustituye el contenido por:  
"JavaScript funcionando correctamente."

Utilización: para demostrar que el navegador puede ejecutar JS estático desde Apache.

## Docker compose

Para gestionar nuestra aplicación compuesta por un servidor web PHP y una base de datos MariaDB, hemos utilizado Docker Compose, definiendo toda la configuración en el archivo docker-compose.yml. En este archivo especificamos los servicios: apache, usando la imagen php:8.1-apache para poder ejecutar nuestros scripts PHP, mapeando el puerto 8080 para el acceso y montando nuestro código local en /var/www/html para reflejar los cambios al instante; y mariadb, usando la imagen oficial, configurando las credenciales y la base de datos inicial mediante variables de entorno y asegurando la persistencia de los datos con el volumen nombrado db\_data. Ambos servicios los conectamos a una red común redweb, lo que permite que el contenedor web se comuniquen con la base de datos usando simplemente el nombre del servicio mariadb, y todo el conjunto se despliega y gestiona de forma unificada con los comandos de docker-compose.

Archivo Docker-compose.yml:

```
AlmaLinux - UO295650_UO295652 [Corriendo] - Oracle VirtualBox
[root@vbox red]# ls
docker-compose.yml  sitio_web
[root@vbox red]# cat docker-compose.yml
version: '3.3'

services:
  apache:
    image: httpd:latest
    ports:
      - "8080:80"
    volumes:
      - ./sitio_web:/usr/local/apache2/htdocs/
    networks:
      - redweb

  mariadb:
    image: mariadb:latest
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: ADMSIS123_
      MYSQL_DATABASE: Database_ASR
      MYSQL_USER: uo295650
      MYSQL_PASSWORD: ADMSIS123_
    volumes:
      - db_data:/var/lib/mysql
    networks:
      - redweb

volumes:
  db_data:

networks:
  redweb:
    driver: bridge
[root@vbox red]#
```

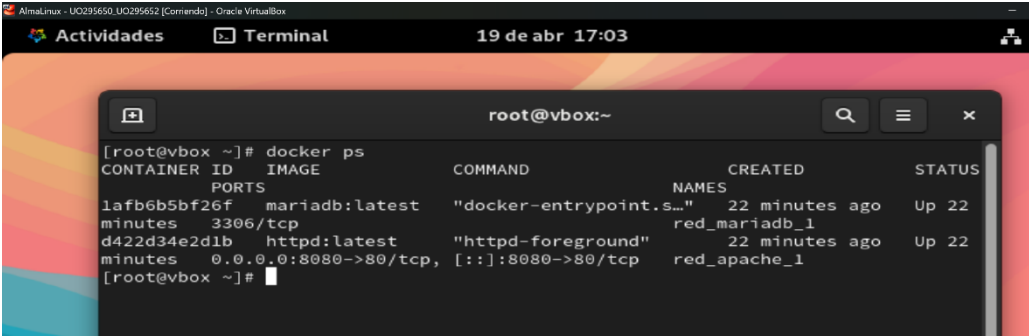
## Despliegue de los contenedores

Una vez que tuvimos nuestro archivo `docker-compose.yml` definido con los servicios `apache` y `mariadb`, el siguiente paso fue poner en marcha la aplicación. Para ello, ejecutamos el siguiente comando desde el directorio `/root/red/` donde se encuentra nuestro archivo `docker-compose.yml`:

- `docker-compose up -d`

```
[root@vbox red]# docker-compose up -d
Creating network "red_redweb" with driver "bridge"
Creating volume "red_db_data" with default driver
Pulling apache (httpd:latest)...
latest: Pulling from library/httpd
8a628cdd7ccc: Pull complete
60ba3d18ad64: Pull complete
4f4fb700ef54: Pull complete
03e322382f93: Pull complete
4ad6b63c403f: Pull complete
c613327bbca6: Pull complete
Digest: sha256:4564ca7604957765bd2598e14134a1c6812067f0dadd7dc5a484431dd03832b
Status: Downloaded newer image for httpd:latest
Pulling mariadb (mariadb:latest)...
latest: Pulling from library/mariadb
2726e237d1a3: Pull complete
0b86886c6aaa: Pull complete
2b221cf763a8: Pull complete
5e4180757702: Pull complete
43028b9f5f8e: Pull complete
bbef7eafa75b: Pull complete
ab732728101f: Pull complete
0c9f57c1bb30: Pull complete
Digest: sha256:81e893032978c4bf8ad43710b7a979774ed90787fa32d199162148ce28fe3b76
Status: Downloaded newer image for mariadb:latest
Creating red_mariadb_1 ... done
Creating red_apache_1 ... done
[root@vbox red]#
```

Tras ejecutar `docker-compose up -d`, queríamos asegurarnos de que ambos contenedores (el servidor web y la base de datos) se habían iniciado correctamente y estaban en ejecución. Para ello, utilizamos el comando `docker ps`. Este comando lista todos los contenedores que se encuentran actualmente activos en el sistema.



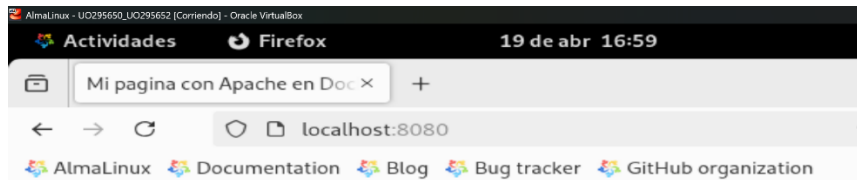
```
AlmaLinux - UO295650_UO295652 [Corriendo] - Oracle VirtualBox
Actividades Terminal 19 de abr 17:03

root@vbox:~
[root@vbox ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
PORTS         NAMES
1afb6b5bf26f   mariadb:latest "docker-entrypoint.s..." 22 minutes ago Up 22
minutes      red_mariadb_1
d422d34e2d1b   httpd:latest   "httpd-foreground"        22 minutes ago Up 22
minutes      red_apache_1
0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
```



Una vez confirmamos con `docker ps` que nuestros contenedores estaban en ejecución, el siguiente paso era verificar si el servicio web Apache estaba funcionando correctamente y sirviendo nuestro contenido.

Para ello abrimos un navegador web en nuestra máquina host y navegamos a la dirección `http://localhost:8080`



## Servidor Apache funcionando en contenedor

JavaScript funcionando correctamente.

Habiendo comprobado que el servidor web estaba operativo, el siguiente paso fue verificar que el contenedor de la base de datos (mariadb) también se había iniciado correctamente y que la configuración inicial definida en el `docker-compose.yml` se había aplicado.

Para interactuar directamente con la base de datos dentro de su contenedor, utilizamos el comando `docker exec`. Este comando nos permite ejecutar un comando dentro de un contenedor que ya está en ejecución.

```
AlmaLinux - UO295650_UO295652 [Corriendo] - Oracle VirtualBox
[root@vbox red]# docker exec -it red_mariadb_1 mariadb -u root
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
[root@vbox red]# docker exec -it red_mariadb_1 mariadb -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 6
Server version: 11.7.2-MariaDB-ubu2404 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| Database_ASR |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.001 sec)

MariaDB [(none)]> █
```

## Modificación de la Aplicación Web para Conexión a la Base de Datos

Con ambos contenedores ya desplegados y correctamente conectados a la red redweb, el siguiente paso es habilitar la comunicación entre ellos a nivel de la propia aplicación. Es decir, aunque técnicamente están en red, nuestro sitio web (aún estático en este punto) no envía ni recibe información de la base de datos MariaDB. Para implementar esta funcionalidad, vamos a modificar el sitio web incorporando un formulario y el script PHP necesario para que las respuestas se guarden en la base de datos.

Nueva estructura

red/

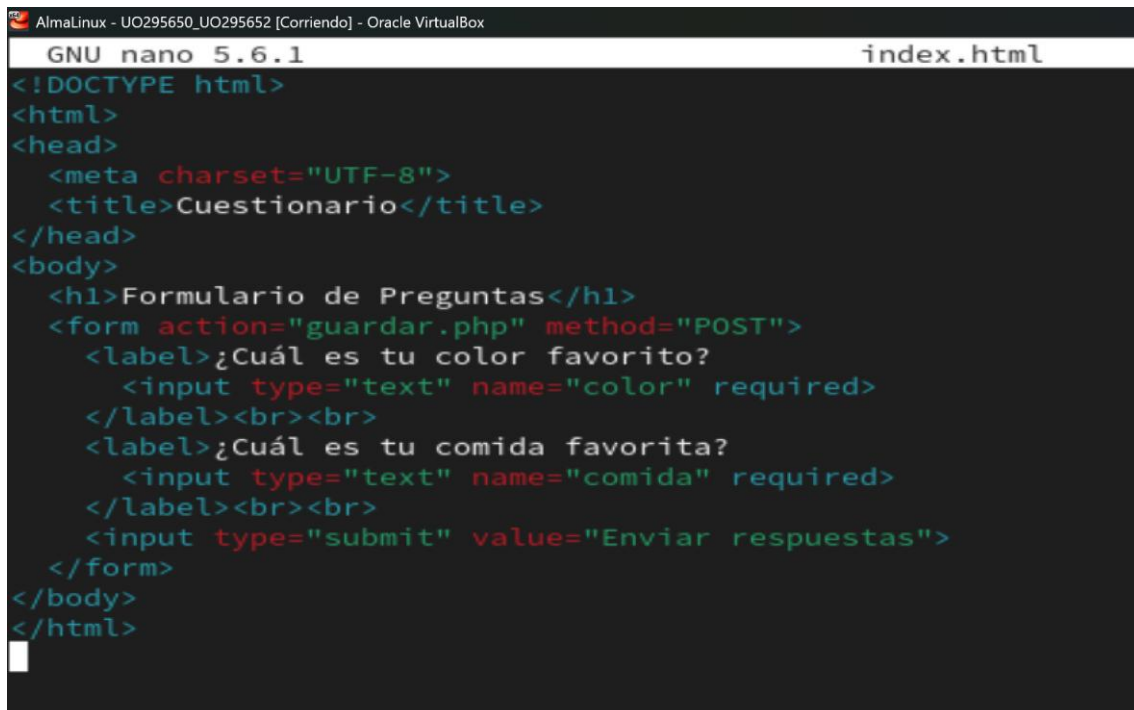
└─ docker-compose.yml

└─ sitio\_web/

└─ index.html

└─ guardar.php

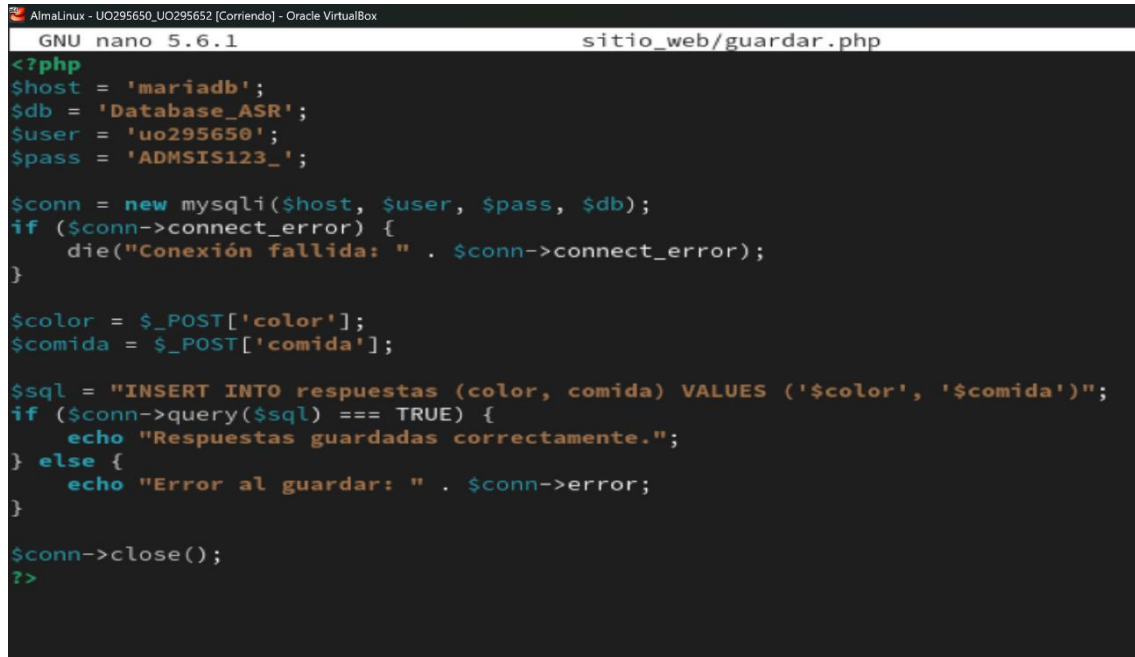
Nuevo index:



```
AlmaLinux - UO295650_UO295652 [Corriendo] - Oracle VirtualBox
GNU nano 5.6.1 index.html
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Cuestionario</title>
</head>
<body>
  <h1>Formulario de Preguntas</h1>
  <form action="guardar.php" method="POST">
    <label>¿Cuál es tu color favorito?
      <input type="text" name="color" required>
    </label><br><br>
    <label>¿Cuál es tu comida favorita?
      <input type="text" name="comida" required>
    </label><br><br>
    <input type="submit" value="Enviar respuestas">
  </form>
</body>
</html>

```

El archivo guardar.php, contiene el código PHP que implementamos para actuar como el backend de nuestro formulario. Su función principal es recibir los datos enviados desde el index.html, conectarse a nuestra base de datos MariaDB (que se ejecuta en su propio contenedor) e insertar esos datos en la tabla correspondiente.



```
AlmaLinux - UO295650_UO295652 [Corriendo] - Oracle VirtualBox
GNU nano 5.6.1                                sitio_web/guardar.php
<?php
$host = 'mariadb';
$db = 'Database_ASR';
$user = 'uo295650';
$pass = 'ADMSIS123_';

$conn = new mysqli($host, $user, $pass, $db);
if ($conn->connect_error) {
    die("Conexión fallida: " . $conn->connect_error);
}

$color = $_POST['color'];
$comida = $_POST['comida'];

$sql = "INSERT INTO respuestas (color, comida) VALUES ('$color', '$comida')";
if ($conn->query($sql) === TRUE) {
    echo "Respuestas guardadas correctamente.";
} else {
    echo "Error al guardar: " . $conn->error;
}

$conn->close();
?>
```

Para poder ejecutar nuestro script guardar.php, tuvimos que realizar dos ajustes esenciales en la configuración del servicio web (apache) dentro del archivo docker-compose.yml.

- Cambio de Imagen: Reemplazamos image: httpd:latest por image: php:8.1-apache. Este cambio fue crucial porque la nueva imagen incluye el intérprete de PHP necesario para procesar nuestro script.
- Ajuste del Volumen: Modificamos la ruta de destino del volumen a ./sitio\_web:/var/www/html, ya que esta es la carpeta raíz web por defecto en la imagen php:8.1-apache, asegurando que Apache encontrara nuestros archivos PHP y HTML.

Nuevo docker-compose.yml:

```
AlmaLinux - UO295650_UO295652 [Corriendo] - Oracle VirtualBox
[root@vbox red]# cat
docker-compose.yml sitio_web/
[root@vbox red]# cat docker-compose.yml
version: '3.3'

services:
  apache:
    image: php:8.1-apache
    ports:
      - "8080:80"
    volumes:
      - ./sitio_web:/var/www/html
    networks:
      - redweb

  mariadb:
    image: mariadb:latest
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: 'ADMSIS123_'
      MYSQL_DATABASE: 'Database_ASR'
      MYSQL_USER: 'uo295650'
      MYSQL_PASSWORD: 'ADMSIS123_'
    volumes:
      - db_data:/var/lib/mysql
    networks:
      - redweb

volumes:
  db_data:

networks:
  redweb:
    driver: bridge
```

Desplegamos los contenedores con:

```
docker-compose up -d
```

Como último paso de configuración, necesitábamos crear la tabla donde se guardarían los datos del formulario. Para ello, accedimos al contenedor mariadb con `docker exec`, nos conectamos a nuestra base de datos `Database_ASR` y ejecutamos el comando SQL `CREATE TABLE respuestas`.

- `docker exec -it red_mariadb_1 mariadb -u root -p`
- `CREATE TABLE respuestas ( id INT AUTO_INCREMENT PRIMARY KEY, color VARCHAR(100), comida VARCHAR(100) );`

Esta acción fue indispensable porque nuestro script `guardar.php` necesita que la tabla `respuestas` exista para poder insertar los datos correctamente. Una vez creada la tabla, toda la aplicación quedó lista para funcionar.

```
MariaDB [Database_ASR]> drop table respuestas;  
Query OK, 0 rows affected (0.059 sec)  
  
MariaDB [Database_ASR]> CREATE TABLE respuestas (id INT AUTO_INCREMENT PRIMARY KEY, color VARCHAR(  
100), comida VARCHAR(100));  
Query OK, 0 rows affected (0.041 sec)
```

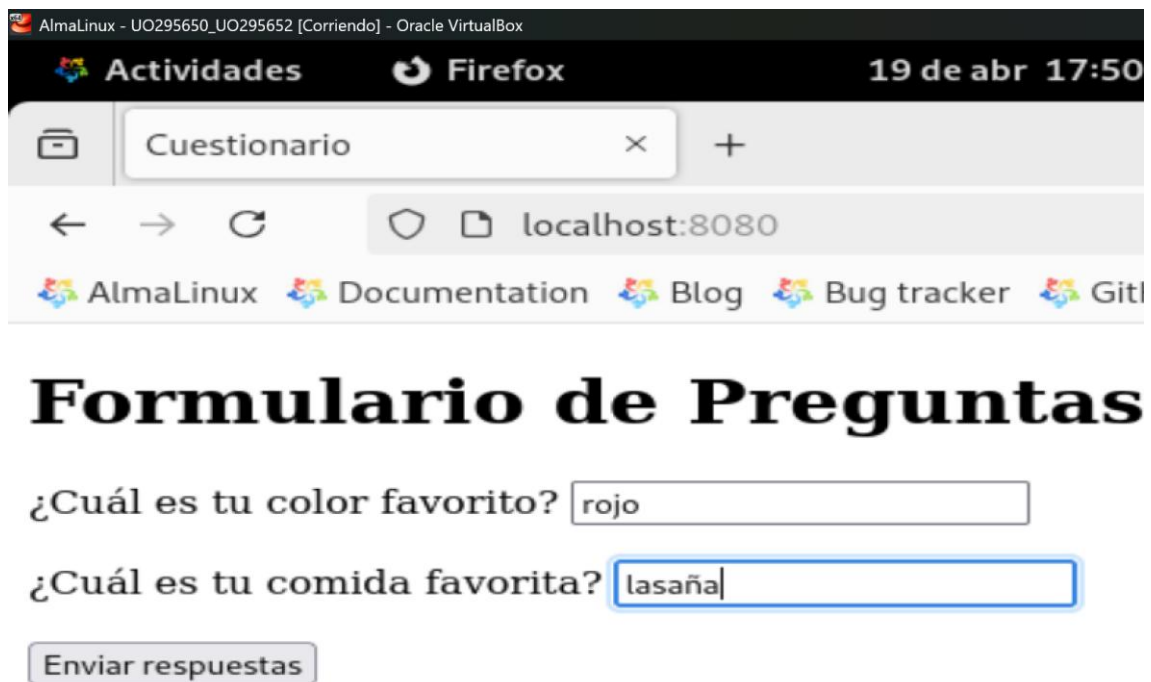
Detectamos que la imagen base `php:8.1-apache` no incluía la extensión `mysqli` de PHP, necesaria para conectar con `MariaDB` desde nuestro script `guardar.php`. Para solucionarlo, creamos el `Dockerfile` mostrado: simplemente parte de la imagen `php:8.1-apache` (FROM) y le añade la extensión `mysqli` usando el comando `RUN docker-php-ext-install mysqli`

```
[root@vbox red]# cat Dockerfile  
FROM php:8.1-apache  
RUN docker-php-ext-install mysqli  
[root@vbox red]#
```

## Prueba Final: Envío y Almacenamiento de Datos

Tras haber configurado todos los componentes (servidor web con PHP y mysql, base de datos con la tabla respuestas creada) y haberlos desplegado con Docker Compose, realizamos la prueba definitiva para verificar el flujo completo de la aplicación.

1. Accedimos a nuestra aplicación web en `http://localhost:8080`.
2. Rellenamos el formulario introduciendo "rojo" como color favorito y "lasaña" como comida favorita.
3. Pulsamos el botón "Enviar respuestas".



The screenshot shows a web browser window titled "AlmaLinux - UO295650\_UO295652 [Corriendo] - Oracle VirtualBox". The browser is Firefox, and the address bar shows "localhost:8080". The page title is "Cuestionario". The form contains two questions: "¿Cuál es tu color favorito?" with the answer "rojo" and "¿Cuál es tu comida favorita?" with the answer "lasaña". A button labeled "Enviar respuestas" is at the bottom.

AlmaLinux - UO295650\_UO295652 [Corriendo] - Oracle VirtualBox

Actividades Firefox 19 de abr 17:50

Cuestionario

localhost:8080

AlmaLinux Documentation Blog Bug tracker Git

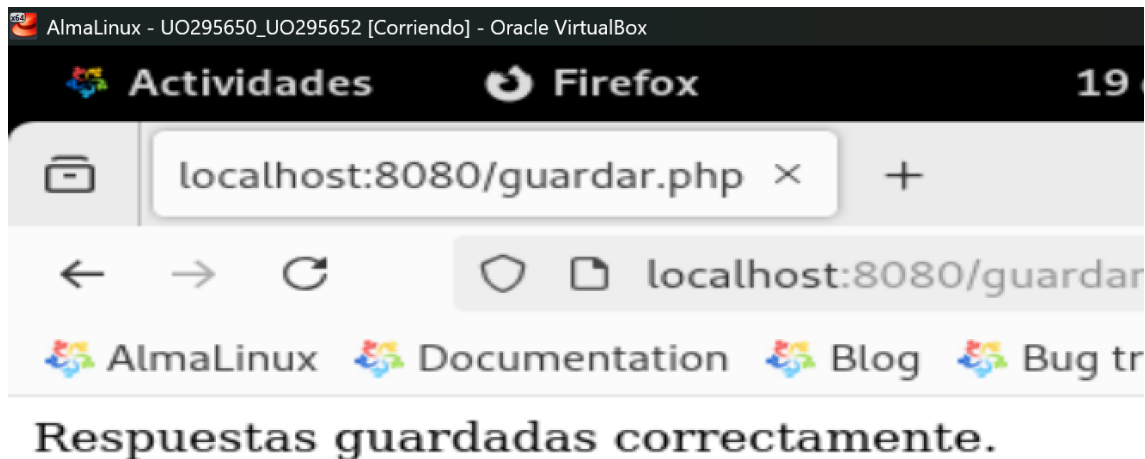
# Formulario de Preguntas

¿Cuál es tu color favorito?

¿Cuál es tu comida favorita?

Al enviar el formulario, el navegador nos redirigió a la página `http://localhost:8080/guardar.php`, tal como especificaba el atributo `action` de nuestro formulario.

La página resultante mostró el mensaje: "Respuestas guardadas correctamente."



Finalmente, para asegurar que los datos enviados desde el formulario web realmente se guardaron, nos conectamos al contenedor `mariadb` y ejecutamos la consulta `SELECT * FROM respuestas`; La captura muestra que la tabla contenía exactamente la fila con `id=1`, `color='rojo'` y `comida='lasaña'`, coincidiendo con los datos que habíamos introducido. Esto nos dio la prueba definitiva de que la aplicación completa, desde el frontend hasta la base de datos, funcionaba correctamente.

```
Database changed
MariaDB [Database_ASR]> SELECT * FROM respuestas;
+-----+-----+-----+
| id | color | comida |
+-----+-----+-----+
| 1 | rojo | lasaña |
+-----+-----+-----+
1 row in set (0.001 sec)

MariaDB [Database_ASR]> 
```

## Conclusiones

Este proyecto demostró con éxito cómo usar Docker Compose para poner en marcha una aplicación web completa, compuesta por un servidor web (PHP/Apache) y una base de datos (MariaDB). Conseguimos los objetivos que teníamos:

- Aprendimos a definir en el archivo docker-compose.yml qué servicios necesita la aplicación (web y base de datos).
- Configuramos cómo deben conectarse entre ellos a través de redes virtuales (permitiendo que el contenedor web encuentre a la base de datos por su nombre, mariadb).
- Aseguramos que los datos de la base de datos se guarden de forma permanente usando volúmenes.