Cuaderno de ejercicios

Tema 4 - Docker

1. Recupera la aplicación web de la calculadora que desarrollaste en el tema 1. En el último ejercicio creaste un Docker personalizado a partir de la aplicación. Ahora debes subir el Docker a tu cuenta de Docker Hub. A continuación, descarga el Docker en tu instancia EC2 de AWS y ejecútala. Por último, accede a la aplicación desde el navegador de tu PC y comprueba que funciona correctamente.

De los ejercicios del Tema 1, creamos el fichero Dockerfile:

```
# Usa una imagen base oficial de PHP con Apache

FROM php:8.2-apache

# Copia el contenido de la carpeta "src" en el contenedor

COPY src/ /var/www/html/
```

A continuación, creamos la imagen. Como la idea es distribuirla, vamos a indicar que se trata de la primera versión:

```
docker build -t calculadora-web:v1 .
```

Debes tener ya creada una cuenta de Docker Hub. Si no lo tienes, créala.

Inicia sesión en Docker Hub:

```
docker login
```

Etiqueta la imagen:

```
docker tag calculadora-web:v1 tuUsuarioDockerHub/calculadora-web:v1
```

Subir la imagen a Docker Hub:

```
docker push tuUsuarioDockerHub/calculadora-web:v1
```

Comprueba en tu cuenta de Docker Hub que la imagen se ha subido correctamente.

Desde la instancia de EC2 de AWS (debes haber instalado Docker, si no lo has hecho, sigue los pasos que se indican en el tema 1 -Windows- o en el tema 4 -Linux-), importa la imagen de Docker Hub:

```
docker pull tuUsuarioDockerHub/calculadora-web:v1
```

Y ejecútala:

```
docker run -d -p 8080:80 rsanzfloridauni/mi_imagen.v1
```

La aplicación ya debe ser accesible desde http://IP_maquina_EC2:8080

2. Crea un formulario web sencillo que permita añadir o borrar películas de la base de datos que has utilizado en este tema. A continuación, integra este formulario en la aplicación web de ejemplo del tema. Puede ser en la misma página que muestra la lista de películas o a través de un botón que abra otra página en el navegador. Crea también los scripts PHP necesarios. Comprueba que funciona correctamente en local.

Ejemplo de "index.php" actualizado con las nuevas funcionalidades:

```
<?php
$servername = "localhost"; // Host de la BDD
$username = "usuario1"; // Nombre de usuario de MySQL
$password = "contrasenyaUsuario1"; // Contraseña de MySQL
$dbname = "cine"; // Nombre de la base de datos

// Crear la conexión
$conn = mysqli_connect($servername, $username, $password, $dbname);

// Verificar la conexión
if ($conn->connect_error) {
    die("Conexión fallida: " . $conn->connect_error);
}

// Lógica para eliminar el registro
if (isset($_POST['id_borrar'])) {
    $id_borrar = intval($_POST['id_borrar']);
    $sql_delete = "DELETE FROM peliculas WHERE id = ?";
    $stmt = $conn->prepare($sql_delete);
    $stmt->bind_param("i", $id_borrar);

    if ($stmt->execute()) {
        echo "<p>Registro eliminado correctamente.</p>";
    } else {
        echo "<p>Error al eliminar el registro: " . $conn->error . "</p>";
    }

    $stmt->close();
}

// Lógica para insertar un nuevo registro
if (isset($_POST['titulo'])) {
    $titulo = $_POST['titulo'];
    $director = $_POST['director'];
    $nota = $_POST['nota'];
    $anyo = intval($_POST['anyo']);
    $presupuesto = intval($_POST['presupuesto']);
    $url_trailer = $_POST['url_trailer'];

    // Procesar la imagen si se ha cargado
    if (isset($_FILES['imagen']) && $_FILES['imagen']['error'] === UPLOAD_ERR_OK) {
        $imagen_temp = $_FILES['imagen']['tmp_name'];
        $img_data = file_get_contents($imagen_temp);
        $img_base64 = base64_encode($img_data);
    } else {
        $img_base64 = ""; // Si no se carga imagen, se guarda como vacío
    }

    $sql_insert = "INSERT INTO peliculas (titulo, director, nota, anyo, presupuesto, img_base64, url_trailer) VALUES (?,
?, ?, ?, ?, ?, ?)";
    $stmt = $conn->prepare($sql_insert);
    $stmt->bind_param("sssisss", $titulo, $director, $nota, $anyo, $presupuesto, $img_base64, $url_trailer);

    if ($stmt->execute()) {
        echo "<p>Nueva película añadida correctamente.</p>";
    } else {
        echo "<p>Error al añadir la película: " . $conn->error . "</p>";
    }
```

```
    $stmt->close();
}

// Consulta SQL para seleccionar todo el contenido de la tabla peliculas
$sql = "SELECT * FROM peliculas";
$result = $conn->query($sql);

?>
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Listado de Películas</title>
</head>
<body>
    <h1>Listado de Películas</h1>
    <table>
        <thead>
            <tr>
                <th>ID</th>
                <th>Título</th>
                <th>Director</th>
                <th>Nota</th>
                <th>Año</th>
                <th>Presupuesto</th>
                <th>Imagen (Base64)</th>
                <th>URL del Trailer</th>
                            <th>Acciones</th>
            </tr>
        </thead>
        <tbody>
            <?php
            if ($result->num_rows > 0) {
                // Salida de cada fila de la tabla
                while($row = $result->fetch_assoc()) {
                    echo "<tr>";
                    echo "<td>" . htmlspecialchars($row["id"]) . "</td>";
                    echo "<td>" . htmlspecialchars($row["titulo"]) . "</td>";
                    echo "<td>" . htmlspecialchars($row["director"]) . "</td>";
                    echo "<td>" . htmlspecialchars($row["nota"]) . "</td>";
                    echo "<td>" . htmlspecialchars($row["anyo"]) . "</td>";
                    echo "<td>" . htmlspecialchars($row["presupuesto"]) . "</td>";
                    echo "<td><img src='" . htmlspecialchars($row["img_base64"]) . "' alt='Imagen'
width='100' height='100'></td>";
                    echo   "<td><a   href='"   .   htmlspecialchars($row["url_trailer"])   .   "'   target='_blank'>Ver
Trailer</a></td>";
                                        echo "<td>";
                    echo "<form method='post' action=''>";
                    echo "<input type='hidden' name='id_borrar' value='" . $row['id'] . "'>";
                    echo  "<input  type='submit'  value='Eliminar'  onclick='return  confirm(\"¿Estás  seguro  de  que  deseas
eliminar este registro?\");'>";
                    echo "</form>";
                    echo "</td>";
                    echo "</tr>";
                }
            } else {
                echo "<tr><td colspan='8'>No hay registros</td></tr>";
            }
            ?>
        </tbody>
    </table>

        <h2>Añadir Nueva Película</h2>
    <form method="post" enctype="multipart/form-data">
        <label for="titulo">Título:</label><br>
        <input type="text" id="titulo" name="titulo" required><br><br>

        <label for="director">Director:</label><br>
        <input type="text" id="director" name="director" required><br><br>

        <label for="nota">Nota:</label><br>
        <input type="text" id="nota" name="nota" required><br><br>
```

```
        <label for="anyo">Año:</label><br>
        <input type="number" id="anyo" name="anyo" required><br><br>

        <label for="presupuesto">Presupuesto:</label><br>
        <input type="number" id="presupuesto" name="presupuesto" required><br><br>

        <label for="imagen">Imagen:</label><br>
        <input type="file" id="imagen" name="imagen" accept="image/*"><br><br>

        <label for="url_trailer">URL del Trailer:</label><br>
        <input type="text" id="url_trailer" name="url_trailer" required><br><br>

        <input type="submit" value="Añadir Película">
    </form>

</body>
</html>

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

3. Crea un Docker con la versión nueva de la aplicación web y súbelo a tu cuenta de Docker Hub.

Recuerda que para que la aplicación que has desarrollado “encuentre” la base de datos, no puede ser que la variable “$servername” apunte a “localhost”, ya que la aplicación se va a ejecutar en un entorno de contenedores de Docker. Siguiendo el ejemplo visto en clase, debes asignar el nombre del servicio que luego tendrá en el fichero de Docker Compose, por ejemplo “db”.

La línea debe queda, por tanto,

```
$servername = “db”; //Host de la BDD
```

A continuación, construye el fichero Dockerfile (igual que en el ejemplo del tema):

```
FROM php:8.2-apache

# Instala y habilita la extensión mysqli

RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli

COPY . /var/www/html/mi_app_cine
```

Por último, crea la imagen, etiquétala y súbela a tu Docker Hub (puedes modificar la versión para no sobrescribir la del ejemplo del tema):

```
docker build -t mi_app_cine:v2 .

docker tag mi_app_cine:v2 tuUsuarioDockerHub/mi_app_cine:v2

docker push tuUsuarioDockerHub/mi_app_cine:v2
```

4. Crea un Docker con la base de datos en el estado en que se encuentre después de haber probado el formulario anterior (deberías haber borrado/añadido algunas películas). Para hacerlo, tendrás que exportar primero el contenido de la base de datos a un fichero .sql. A continuación, crea el Docker personalizado y súbelo a tu cuenta de Docker Hub.

El proceso que hay que seguir en este caso es prácticamente igual que el ejemplo visto en el tema. Lo único que va a cambiar es el contenido del fichero “cine.sql”, que va a tener unos datos distintos al que había en el tema, en función de lo que hayas modificado con tu aplicación. Sin embargo, la creación de la base de datos, de la tabla y del usuario y sus permisos es igual. Por tanto, puedes utilizar de base el fichero “cine.sql” y editarlo o generarlo de nuevo.

Por otro lado, el fichero Dockerfile sí que es exactamente:

```
FROM mysql:latest
ENV MYSQL_ROOT_PASSWORD=contrasenyaRoot
ENV MYSQL_DATABASE=cine
ENV MYSQL_USER=usuario1
ENV MYSQL_PASSWORD=contrasenyaUsuario1
COPY cine.sql /docker-entrypoint-initdb.d/
```

Y los pasos siguientes también serían iguales (puedes también modificar los nombres de las imágenes y las versiones, según tu conveniencia):

Construir la imagen:

```
docker build -t mi_cine:v2 .
```

Etiquetar la imagen:

```
docker tag mi_cine:v2 tuUsuarioDockerHub/mi_cine:v2
```

Subir la imagen a Docker Hub:

```
docker push tuUsuarioDockerHub/mi_cine:v2
```

5. Crea un nuevo fichero de Docker Compose con las imágenes actualizadas de la aplicación y de la base de datos. Incluye también phpMyAdmin para facilitar la gestión de la base de datos.

El fichero “compose.yaml” también sería muy parecido al visto en el ejemplo del tema (hemos quitado las variables de entorno para la imagen de la base de datos porque ya están incluidas en la imagen que hemos subido a DockerHub):

```
services:
  db:
    image: tuUsuarioDockerHub/mi_cine:v2
    container_name: contenedorMiCine
    ports:
      - "3306:3306"

  php:
    image: tuUsuarioDockerHub/mi_app_cine:v2
```

```
    container_name: contenedorMiApp
    ports:
      - "8080:80"
    depends_on:
      - db

  phpmyadmin:
    image: phpmyadmin/phpmyadmin:latest
    container_name: contenedorPhpMyAdmin
    environment:
      PMA_HOST: db
      MYSQL_ROOT_PASSWORD: contrasenyaRoot
    ports:
      - "8081:80"
    depends_on:
      - db
```

6. Ejecuta el Docker Compose que acabas de crear en local para comprobar que funciona correctamente. A continuación, ejecútalo en tu instancia EC2 de AWS. Accede desde el navegador de tu PC y comprueba que todas las funcionalidades se ejecutan correctamente.

Transfiere el fichero (o créalo) en algún directorio de la instancia EC2. Para ejecutar el Docker Compose, recuerda que hay que situarse en el directorio donde esté el fichero "compose.yaml" y ejecutar las instrucciones:

```
sudo chmod 666 /var/run/docker.sock
docker compose up -d
```

Y luego ya puedes acceder a la aplicación desde la ruta: http://IP_EC2:8080

Y si quieres acceder a phpMyAdmin: http://IP_EC2:8081

NOTAS:

- Al levantar los contenedores con Docker Compose se los trae (hace un "pull") de Docker Hub. Si a continuación los paras (con "docker compose down") y los vuelves a levantar de nuevo ("docker compose up -d"), ten en cuenta que puede utilizar las mismas imágenes que ya se ha traído anteriormente, es decir, si has hecho alguna corrección en tus imágenes y las has subido de nuevo a Docker Hub, es posible que no se las traiga de nuevo solo con esta instrucción. Para hacer correctamente, tendrías que parar los contenedores que estén en ejecución (incluso borrarlos), tal y como has visto en el tema.
- Para el acceso a la aplicación y a phpMyAdmin, ten en cuenta que los puertos (8080 y 8081 en este caso) deben estar abiertos al tráfico en la configuración de seguridad de tu instancia EC2.