FACULTAD DE INGENIERÍA

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

Universidad Autónoma del Estado de México. Facultad de Ingeniería. Ingeniería de computación. Tecnologías computacionales.

Trabajo: Desarrollo y Dockerización de un Blog Académico con PHP, MySQL y Apache.

Profesor: Dr. Jose Antonio Hernández Servin.

Equipo:

+ Josue Yael Ávila Figueroa 2020953 + Emmanuel Santiago Dattoli 2021018

Fecha de entrega: 27 de Noviembre 2025

Periodo 2025B

1. Objetivo

Desarrollar e implementar una aplicación web de tipo blog académico que permita la administración integral de usuarios, publicaciones y comentarios, utilizando tecnologías como PHP, MySQL y Docker. El proyecto buscará establecer un entorno de desarrollo y despliegue altamente portable, reproducible y consistente, que facilite su instalación y operación en cualquier sistema operativo. Asimismo, se pretende garantizar una arquitectura modular, segura y escalable que permita una experiencia de uso eficiente, favorezca la colaboración académica y asegure la mantenibilidad del sistema a largo plazo.

—

2. Introducción

Un blog es una aplicación web que permite publicar, almacenar y consultar contenido de manera dinámica, constituyéndose como una de las herramientas digitales más utilizadas en el ámbito académico para la difusión de información, la publicación de recursos educativos y la comunicación institucional. Su naturaleza flexible y su estructura orientada a la interacción lo convierten en un medio eficaz para organizar y presentar contenido de forma accesible y actualizable.

En el presente proyecto se desarrolló un blog académico básico como ejercicio práctico orientado a comprender de manera integral el funcionamiento de una aplicación web bajo el modelo de arquitectura cliente-servidor. A través de este desarrollo se buscó reforzar conceptos fundamentales relacionados con la gestión de contenido, la interacción con bases de datos, la comunicación entre componentes del sistema y los principios esenciales de diseño y despliegue de aplicaciones web modernas.

Tradicionalmente, el desarrollo de aplicaciones web basadas en PHP y MySQL se lleva a cabo mediante herramientas como XAMPP o WAMP, las cuales proporcionan entornos locales de servidor relativamente fáciles de configurar. No obstante, estas soluciones presentan limitaciones importantes relacionadas con la portabilidad del proyecto, la compatibilidad entre distintos sistemas operativos y la dependencia de configuraciones específicas de cada equipo. Como resultado, reproducir el mismo entorno de ejecución en diferentes máquinas puede volverse complejo, especialmente cuando se requiere uniformidad para fines académicos, colaborativos o de despliegue multiplataforma.

La problemática principal identificada radicó en la necesidad de que el proyecto pudiera ejecutarse de manera consistente en diversos equipos —incluyendo Windows, Linux y macOS— sin depender de instalaciones adicionales, ajustes manuales o configuraciones particulares del entorno. Ante este desafío, se optó por emplear **Docker** como solución tecnológica. Docker permite encapsular aplicaciones dentro de contenedores aislados que incluyen todos los componentes necesarios para su correcto funcionamiento, garantizando que el comportamiento del sistema sea idéntico independientemente del dispositivo o sistema operativo donde se ejecute.

Para este proyecto, la aplicación se estructuró en tres servicios principales: un contenedor para PHP con Apache, otro para MySQL y un tercero para phpMyAdmin. Estos componentes fueron orquestados mediante `docker-compose`, lo que permitió automatizar su despliegue, facilitar la gestión de dependencias y asegurar que todo el entorno pudiera levantarse de forma rápida, coherente y reproducible con un solo comando. Esta aproximación no solo resolvió los problemas de portabilidad, sino que también aportó buenas prácticas de desarrollo orientadas a la modularidad y al uso de entornos estandarizados.

El uso de contenedores se ha convertido en un estándar en la industria del software debido a su eficiencia, portabilidad y reproducibilidad. Tal como lo menciona Merkel (2014), Docker permite empaquetar aplicaciones junto con todas sus dependencias, asegurando que la ejecución sea consistente sin importar el sistema donde se implemente.

Durante el proyecto se investigaron los conceptos fundamentales de arquitectura web, contenedores, servicios, redes internas, volúmenes persistentes y vinculación entre aplicaciones y bases de datos.

locate(loc => { if counter(page).at(loc).page() != 1 { align(right, «Trabajo: Blog Académico con Docker») } })

También se practicó la manipulación de scripts SQL, la integración de PHP con MySQL mediante la extensión **mysqli**, y la gestión de rutas dentro de contenedores.

La introducción de Docker en el proyecto resolvió completamente la problemática inicial y permitió que la aplicación pudiera ser fácilmente compartida, ejecutada y evaluada por el docente en otro sistema operativo utilizando exactamente el mismo entorno.

3. Modelo

Placeholder: Diagrama Relacional de la base de datos (usuarios, posts, comentarios)

Figura 1: Diagrama relacional de la base de datos del blog en phpMyAdmin.
Muestra el diagrama relacional generado en phpMyAdmin para la base de datos `blogdb`, utilizada por la aplicación web del blog académico.

En el diseño se representan las tres tablas principales del sistema: `usuarios`, `posts` y `comentarios`. La tabla `usuarios` contiene la información básica de los autores, mientras que la tabla `posts` almacena las publicaciones creadas, vinculándose a la tabla de `usuarios` mediante la clave foránea `id_usuario`. A su vez, la tabla `comentarios` registra las opiniones asociadas a cada publicación, relacionándose tanto con `posts` como con `usuarios` para identificar el contenido comentado y el autor del comentario. El diagrama evidencia la estructura lógica del modelo de datos y las relaciones que permiten garantizar la integridad referencial del sistema, mostrando visualmente cómo interactúan los elementos fundamentales del blog dentro del motor MySQL.

4. Código y Ejecución de Docker

4.1. Código del Docker-compose

Placeholder: Contenido del archivo docker-compose.yml

Figura 2: Muestra la estructura completa del archivo `docker-compose.yml` utilizado para definir y orquestar los distintos servicios que conforman la aplicación web.

En primer lugar, se observa la configuración del servicio correspondiente al servidor web, el cual se construye a partir del `Dockerfile` ubicado en el proyecto. Este servicio monta, mediante volúmenes, el código fuente de la aplicación dentro del contenedor, permitiendo que los cambios realizados en los archivos locales se reflejen inmediatamente sin necesidad de recompilar la imagen. Asimismo, se expone el puerto `8080`, que redirige al puerto interno `80` de Apache, y se declara la dependencia directa del servicio de base de datos, garantizando que este último se inicie previamente.

A continuación, se detalla el servicio asociado al motor de base de datos MySQL, en el que se especifica la imagen a utilizar, las credenciales de acceso, el nombre de la base de datos por defecto y otros valores de entorno necesarios para su inicialización. También se define un volumen persistente para almacenar los datos, evitando su pérdida incluso si el contenedor es eliminado. Además, se incluye la carga automática del archivo `init.sql` mediante el mecanismo de `docker-entrypoint`, lo que permite crear tablas o insertar datos iniciales al momento de levantar el servicio. El puerto `3306` se expone para permitir conexiones externas.

Finalmente, se observa la configuración del servicio phpMyAdmin, herramienta que facilita la gestión visual de la base de datos. Este servicio utiliza la imagen oficial y se conecta al contenedor de MySQL mediante variables de entorno que establecen el host del servidor. En conjunto, el archivo refleja una arquitectura modular, permitiendo un despliegue uniforme, reproducible y eficiente de la aplicación web.

4.2. Ejecución del comando `docker compose up --build`

Placeholder: Salida de la terminal con el comando `docker compose up --build`

Listado 1: Muestra la ejecución del comando `docker compose up --build` dentro del directorio del proyecto, proceso mediante el cual Docker construye y levanta los contenedores definidos en el archivo `docker-compose.yml`.

En primer lugar, el sistema inicia la fase de construcción de imágenes, cargando las definiciones del `Dockerfile` y transfiriendo el contexto. Posteriormente, se descargan o reutilizan las capas base de la imagen de PHP con Apache, tras lo cual se ejecutan las instrucciones internas del contenedor, como la instalación de librerías y la habilitación de extensiones para MySQL.

Una vez finalizado el proceso de construcción, Docker procede a crear la red predeterminada del proyecto y a inicializar cada uno de los servicios: la base de datos MySQL, la interfaz de administración phpMyAdmin y el servidor web encargado de ejecutar el código PHP. Cada contenedor se marca con el estado correspondiente (`Created`, `Started` o `Built`), indicando que los servicios han sido desplegados

locate(loc => { if counter(page).at(loc).page() != 1 { align(right, «Trabajo: Blog Académico con Docker») } })

correctamente. La terminal refleja todo el flujo automatizado de compilación, configuración y arranque de la aplicación web mediante la orquestación proporcionada por Docker.

locate(loc => { if counter(page).at(loc).page() != 1 { align(right, «Trabajo: Blog Académico con Docker») } })

4.3. Código del Dockerfile

La configuración base del entorno se construye a partir de la imagen oficial de PHP 8.2 con Apache. Se define la sección de instalación de extensiones: mediante una única instrucción `RUN`, se actualizan los repositorios, se instalan dependencias necesarias, y se activan las extensiones de PHP `mysqli` y `pdo_mysql` para permitir la conexión a bases de datos. Se realiza una limpieza para reducir el tamaño de la imagen.

Posteriormente, se incluye una instrucción `RUN a2enmod rewrite` para habilitar el módulo `mod_rewrite` de Apache, esencial para el manejo de **URL amigables**. La instrucción `WORKDIR /var/www/html` establece el directorio de trabajo predeterminado dentro del contenedor. Las instrucciones `COPY` realizan la copia inicial del código del proyecto. Asimismo, se declara el puerto interno que expondrá el servicio: `EXPOSE 80`.

4.4. Visualización de los contenedores activos

5. Arquitectura y Funcionamiento

5.1. Arquitectura general del sistema

El proyecto está compuesto por tres servicios:

1. **Web (PHP + Apache):** Ejecuta el código del blog y atiende peticiones HTTP en el puerto 8080.
2. **Base de datos (MySQL 8):** Almacena usuarios, posts y comentarios, utilizando un volumen Docker para persistencia.
3. **phpMyAdmin:** Cliente web para administrar la base, disponible en el puerto 8081.

Los tres servicios están interconectados mediante una red interna creada automáticamente por Docker Compose.

5.2. Estructura del proyecto

blog-web/ │── src/ │ ├── index.php │ ├── login.php │ ├── register.php │ ├── new_post.php │ ├── view_post.php │ ├── conexion.php │ └── logout.php │── styles/ │── db/ │ └── init.sql │── Dockerfile └── docker-compose.yml

5.3. Base de datos del proyecto (Modelo entidad-relación)

1. **usuarios:** id (PK), nombre, correo, contraseña.
2. **posts:** id (PK), id_usuario (FK), título, contenido, fecha.
3. **comentarios:** id (PK), id_post (FK), id_usuario (FK), contenido, fecha.

5.4. Funcionamiento de la Conexión

La conexión a la base de datos desde la aplicación PHP se realiza mediante el siguiente comando:

conexion = new mysqli(«db», «Yael», «password», «blogdb»)

El parámetro `"db"` corresponde al nombre del servicio definido en `docker-compose.yml`, lo cual permite que PHP resuelva automáticamente la red interna generada por Docker sin necesidad de utilizar direcciones IP locales.

5.4.1. Dockerfile

Define el entorno exacto en el que se ejecutará la aplicación, asegurando consistencia. Utiliza como base la imagen `php:8.2-apache` e instala la extensión `mysqli` para permitir la comunicación con el servidor MySQL.

5.4.2. docker-compose.yml

Actúa como el orquestador principal del sistema. Al ejecutar `docker-compose up -d`, realiza automáticamente la construcción de la imagen PHP–Apache, la inicialización del servicio MySQL y la puesta en marcha de phpMyAdmin. Expone los puertos hacia el host, permitiendo el acceso a los servicios en `http://localhost:8080` (Blog) y `http://localhost:8081` (phpMyAdmin).

6. Discusión del proyecto

El funcionamiento del sistema fue validado mediante diversas pruebas operativas que incluyeron el registro de nuevos usuarios, el inicio de sesión y la creación de publicaciones, confirmando la integridad del flujo completo de datos.

Uno de los resultados más relevantes fue la verificación de que la conexión a MySQL no debe realizarse empleando `"localhost"`, sino utilizando el nombre del servicio `"db"`. Este ajuste es fundamental, ya que los servicios se identifican por sus nombres dentro de la red interna de Docker. El uso de volúmenes en MySQL garantizó la persistencia de los datos.

El blog pudo ejecutarse correctamente tanto en Windows como en Linux sin requerir modificaciones, validando el objetivo de lograr un entorno completamente portable. Este enfoque eliminó los errores recurrentes asociados a herramientas tradicionales como XAMPP y permitió estructurar la aplicación y la base de datos para admitir futuras mejoras.

7. Conclusión

Se cumplió plenamente con el objetivo planteado, demostrando que la aplicación es funcional, estable y capaz de ejecutarse de manera uniforme en distintos sistemas operativos mediante Docker, lo que confirma su carácter portable y reproducible. El uso de Docker permitió abstraer las limitaciones de las dependencias locales y las configuraciones específicas de cada equipo, resolviendo la problemática inicial.

La experiencia proporcionó una comprensión profunda sobre la importancia de un diseño modular, la orquestación de servicios y la persistencia de datos.

7.1. Posibles Mejoras Futuras

1. Implementar un sistema de roles para diferenciar privilegios entre administradores y usuarios.
2. Incorporar herramientas para el manejo de archivos e imágenes.
3. Adoptar una arquitectura más formal basada en el patrón MVC para mejorar la estructura del proyecto y fortalecer su mantenibilidad.

8. Referencias

1. Merkel D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux Journal. 2014;239(2).
2. Robbins, A. Learning PHP, MySQL & JavaScript. O'Reilly Media; 2018.
3. Sitio oficial de Docker. ¿Qué es Docker? Disponible en: https://www.docker.com/resources/what-container/
4. Welling, L., Thomson, L. PHP and MySQL Web Development. Addison Wesley Professional; 2016.