Universidad Autónoma del Estado de México.

Facultad de Ingeniería.

Ingeniería de computación.

Tecnologías computacionales.

Dr. Jose Antonio Hernández Servin.

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

Equipo: Josue Yael Ávila Figueroa 2020953.

Emmanuel Santiago Dattoli 2021018.

Fecha de entrega: 27 de Noviembre 2025

Periodo 2025B.

índice

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.

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 al modularidad y al uso de entornos estandarizados.

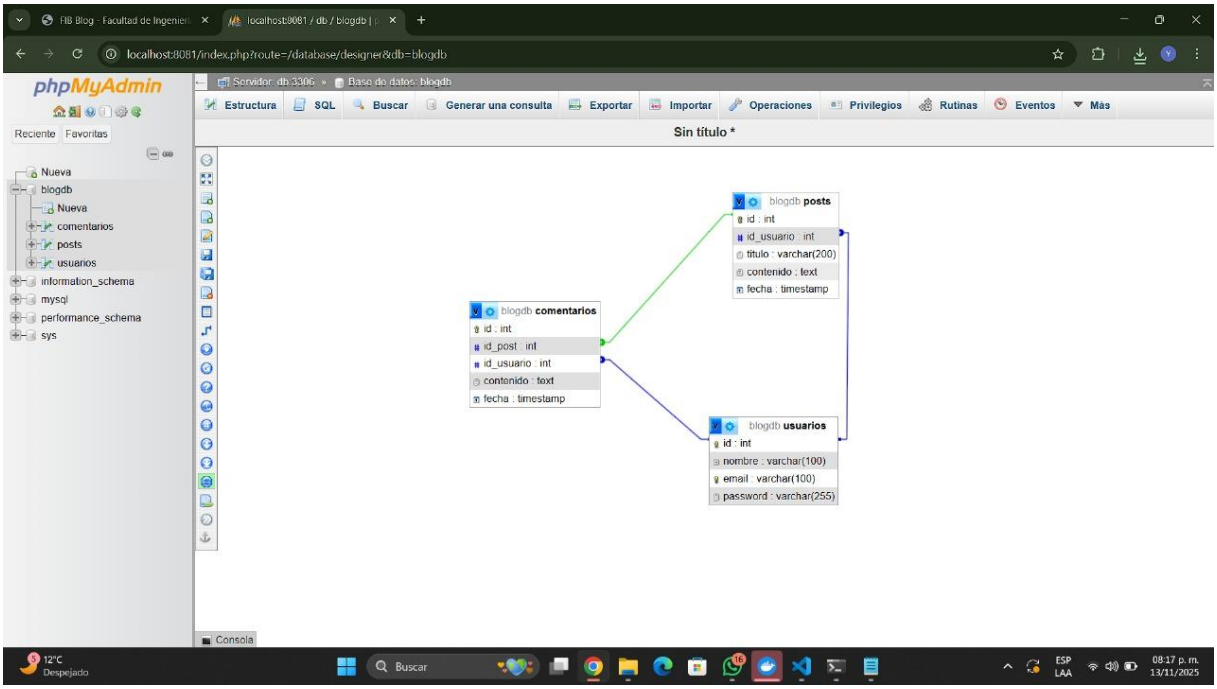
El uso de contenedores se ha convertido en un estándar en la industria del software debido a su eficiencia, portabilidad y reproducibilidad. Tal como lo menciona Merkel (2014), Docker permite empaquetar aplicaciones junto con todas sus dependencias, asegurando que la ejecución sea consistente sin importar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

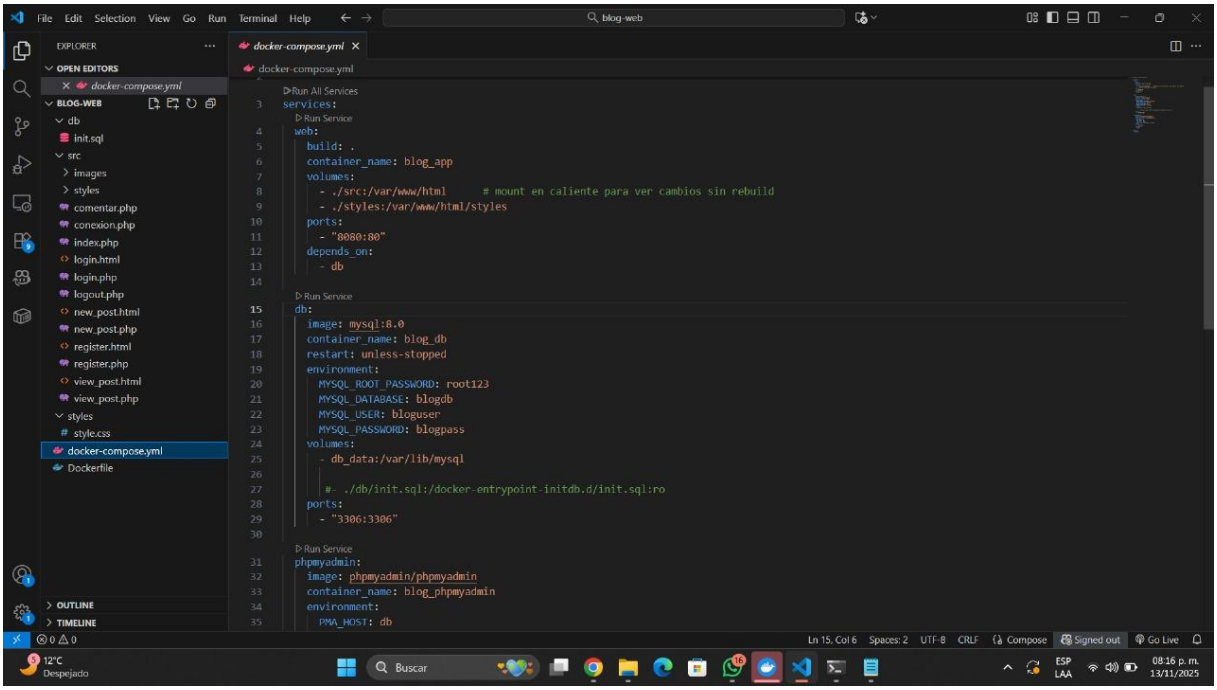
Modelo.

- 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.

- Código del Docker-compose.



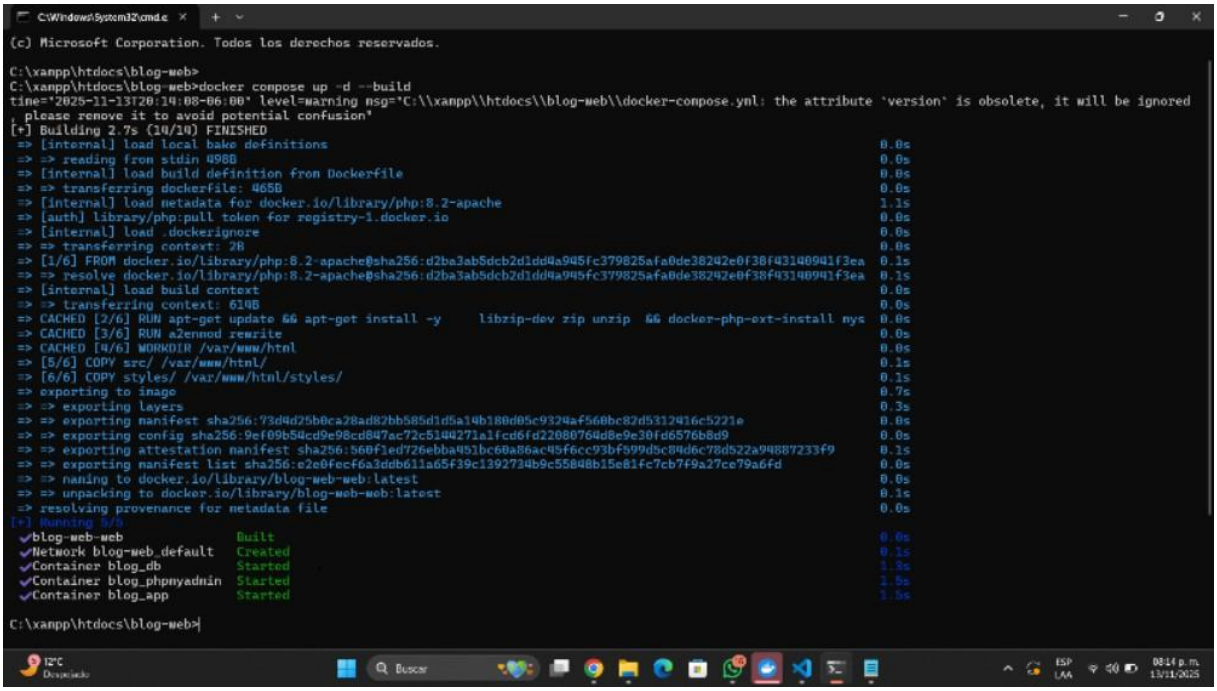
Muestra la estructura completa del archivo docker-compose.yml utilizado para definir y orquestar los distintos servicios que conforman la aplicación web. En primer lugar, se observa la configuración del servicio correspondiente al servidor web, el cual se construye a partir del Dockerfile ubicado en 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, como las realizadas desde clientes SQL o herramientas administrativas.

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 del

proyecto y se conecta al contenedor de MySQL mediante variables de entorno que establecen el host del servidor. También se expone el puerto correspondiente para que pueda accederse desde un navegador web. En conjunto, el archivo refleja una arquitectura modular en la que cada servicio está claramente definido, aislado y automatizado, permitiendo un despliegue uniforme, reproducible y eficiente de la aplicación web.

- Ejecución del comando Docker compose up –build.



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 necesario para la compilación. 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 actualización de paquetes, la instalación de librerías adicionales y la habilitación de extensiones requeridas 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 correctamente y que la aplicación está lista para ejecutarse en un entorno aislado, portable y reproducible. En síntesis, la terminal refleja todo el flujo automatizado de compilación, configuración y arranque de la aplicación web mediante la orquestación proporcionada por Docker.

- Código del Dockerfile.

```dockerfile
FROM php:8.2-apache

# extensiones
RUN apt-get update && apt-get install -y \
    libzip-dev zip unzip \
 && docker-php-ext-install mysqli pdo pdo_mysql \
 && apt-get clean && rm -rf /var/lib/apt/lists/*

# habilitar mod_rewrite
RUN a2enmod rewrite

WORKDIR /var/www/html

# copia inicial del proyecto - bind mounts desde docker-compose
COPY src/ /var/www/html/
COPY styles/ /var/www/html/styles/

EXPOSE 80
```

Muestra la estructura completa del archivo Dockerfile utilizado para definir y configurar la imagen del servicio del servidor web que conformará parte de la aplicación. En primer lugar, se observa la configuración base del entorno, la cual se construye a partir de la imagen oficial de PHP 8.2 con Apache

A continuación, se define la sección de instalación de extensiones. Mediante una única instrucción RUN, se actualizan los repositorios (apt-get update), se instalan dependencias necesarias (apt-get install -y libzip-dev zip unzip), y se activan las extensiones de PHP mysqli y pdo_mysql para permitir la conexión a bases de datos MySQL/MariaDB (docker-php-ext-install). Finalmente, se realiza una limpieza para reducir el tamaño de la imagen (apt-get clean && rm -rf /var/lib/apt/lists/*).

Posteriormente, se incluye una instrucción RUN a2enmod rewrite para habilitar el módulo mod_rewrite de Apache, esencial para el manejo de URL amigables o *pretty URLs*.

La instrucción WORKDIR /var/www/html establece el directorio de trabajo predeterminado dentro del contenedor, que es donde Apache sirve los archivos.

Las siguientes instrucciones COPY realizan la copia inicial del código del proyecto dentro del contenedor, en el *path* establecido como *webroot*. En un entorno con docker-compose.yml, esta copia inicial puede ser complementada o sobrescrita por bind mounts (volúmenes de montaje) definidos en el archivo de orquestación, permitiendo que los cambios realizados en los archivos locales se reflejen inmediatamente sin necesidad de recompilar la imagen.

Asimismo, se declara el puerto interno que expondrá el servicio: EXPOSE 80, que es el puerto predeterminado de Apache dentro del contenedor. Este puerto interno será típicamente mapeado (redireccionado) a un puerto del *host* (como el 8080) mediante la configuración del archivo docker-compose.yml, garantizando el acceso a la aplicación.

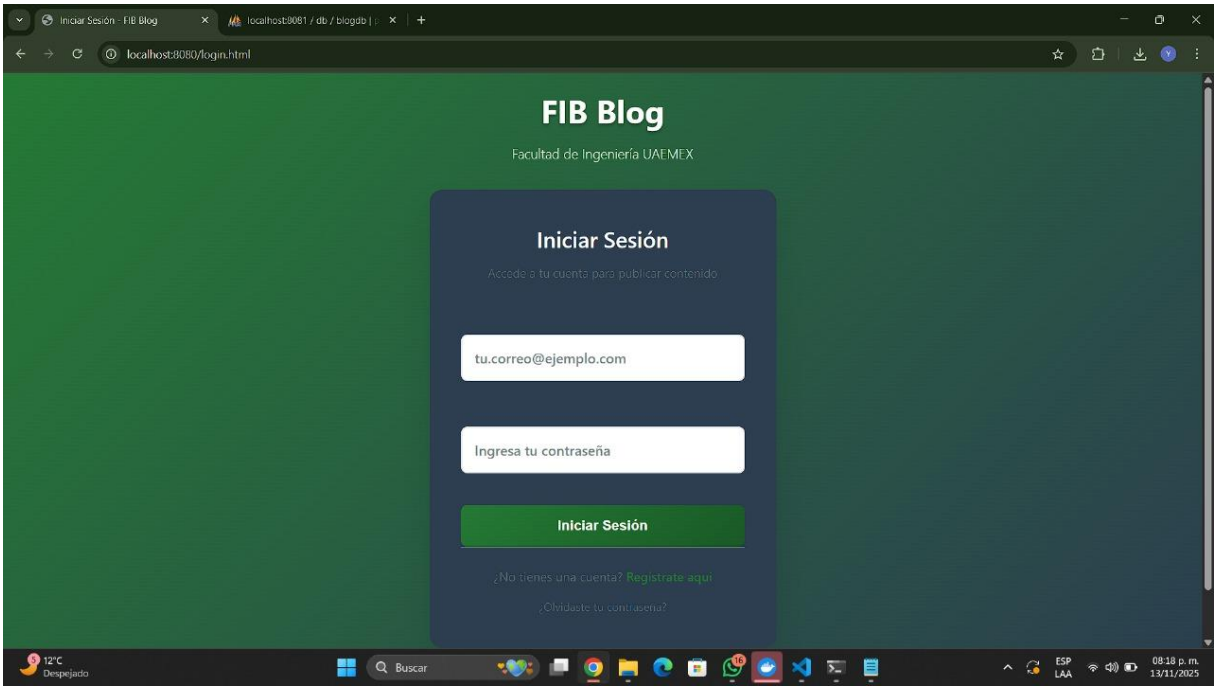
- Visualización de los contenedores activos mediante el comando docker ps.

```
C:\xampp\htdocs\blog-web>docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS          PORTS
                         NAMES
ca11874f3d5e   blog-web-web           "docker-php-entrypoi…"   42 seconds ago   Up 40 seconds   0.0.0.0:8080
->80/tcp, [::]:8080->80/tcp       blog_app
1bc5939fc9e0   phpmyadmin/phpmyadmin   "/docker-entrypoint.…"   42 seconds ago   Up 40 seconds   0.0.0.0:8081
->80/tcp, [::]:8081->80/tcp       blog_phpmyadmin
77866c3201b5   mysql:8.0              "docker-entrypoint.s…"   42 seconds ago   Up 41 seconds   0.0.0.0:3306
->3306/tcp, [::]:3306->3306/tcp   blog_db

C:\xampp\htdocs\blog-web>
```

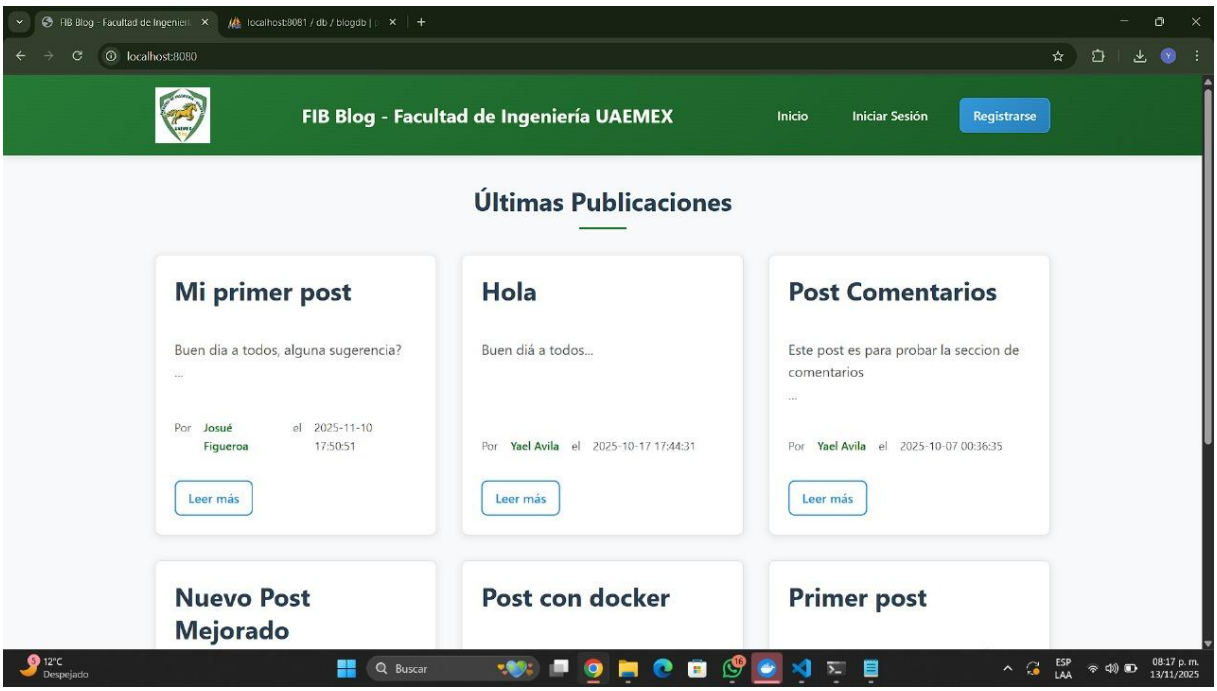
Muestra el resultado de ejecutar el comando docker ps, el cual permite listar los contenedores que se encuentran en funcionamiento dentro del entorno Docker. En este caso, se observan los tres servicios definidos para la aplicación: el servidor web basado en PHP y Apache (blog_app), la herramienta de administración phpMyAdmin (blog_phpmyadmin) y la base de datos MySQL (blog_db). Para cada contenedor se detallan la imagen utilizada, el comando de entrada, el tiempo desde su creación, su estado actual —indicando que todos se mantienen en ejecución— y los puertos expuestos hacia el sistema anfitrión. Esta salida confirma que todos los servicios esenciales del proyecto han sido desplegados correctamente y están operando de manera coordinada dentro del entorno de contenedores.

- Página principal del proyecto.



La imagen muestra el inicio de sesión del proyecto, al registrar el proyecto, se guardan los correos, así como nombres que decide utilizar el usuario.

- Imagen de las publicaciones realizadas en la página.

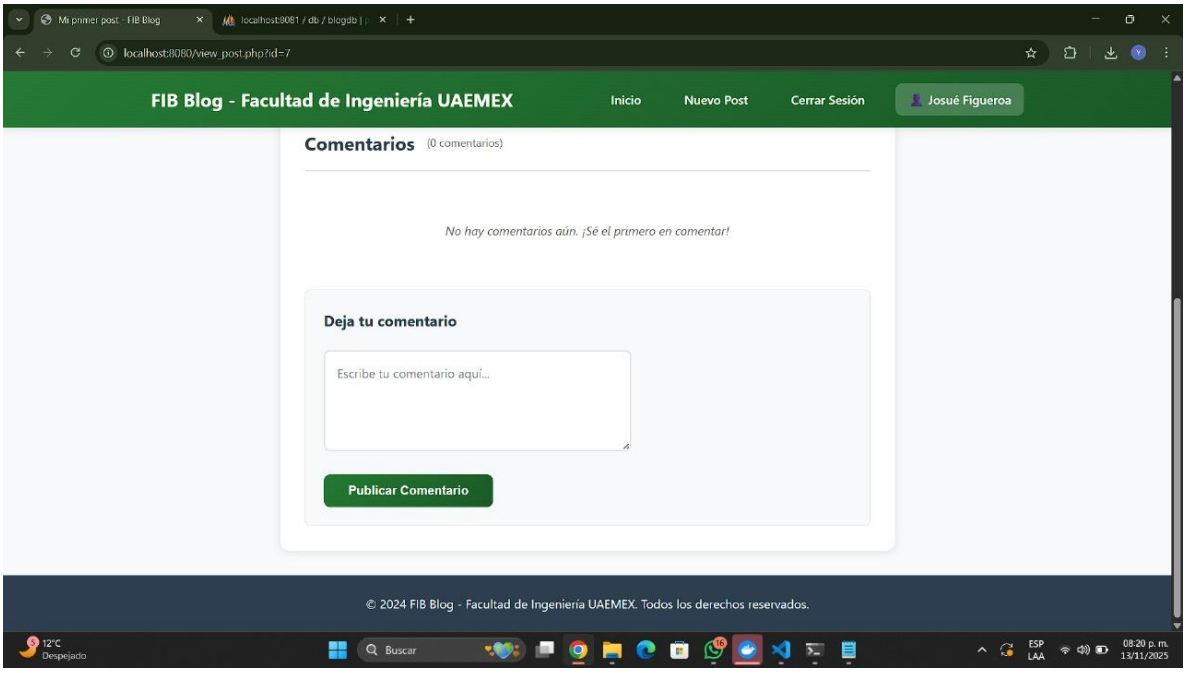


Actualmente aún no se desarrolla la importancia de las publicaciones o mostrar las que se consideren más importantes.

- Imagen de la página para realizar las publicaciones.

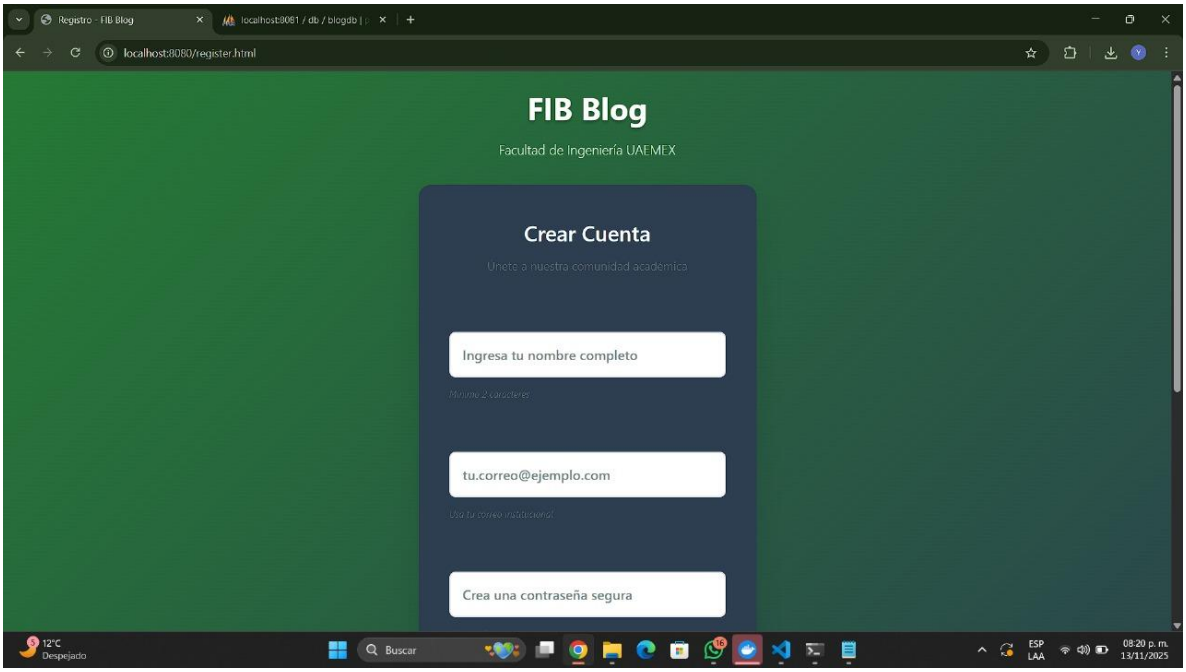
Muestra los apartados a llenar para realizar la publicación.

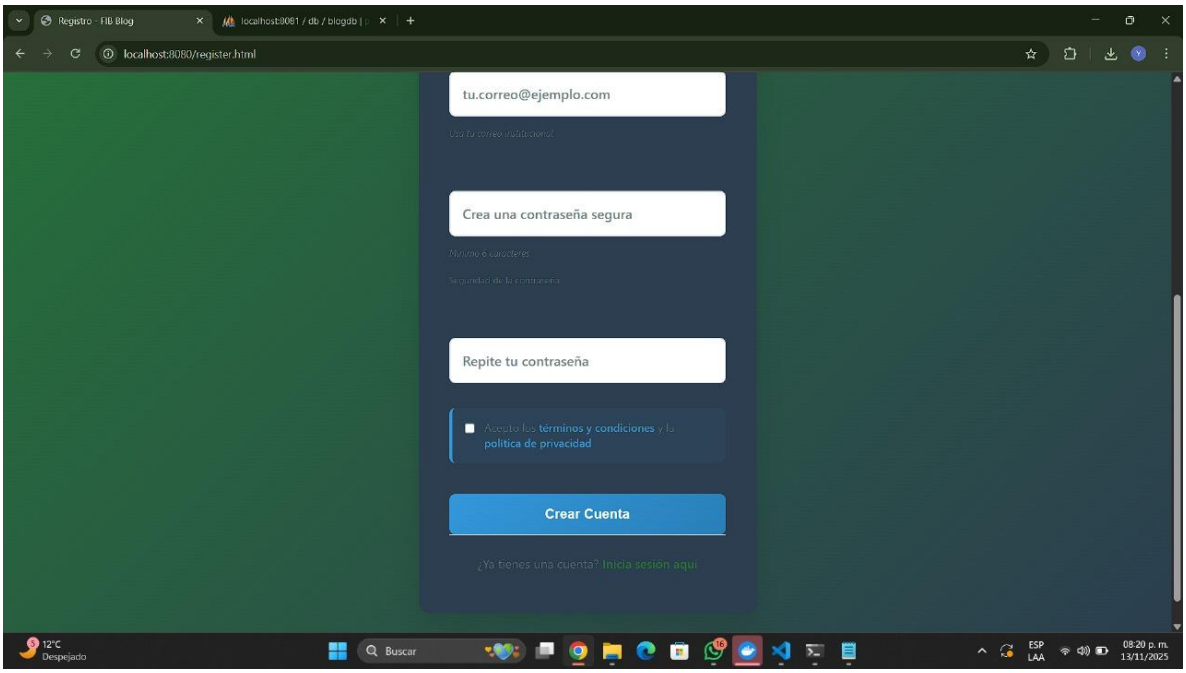
- Apartado de comentarios.



Muestra el apartado de los comentarios dentro del proyecto.

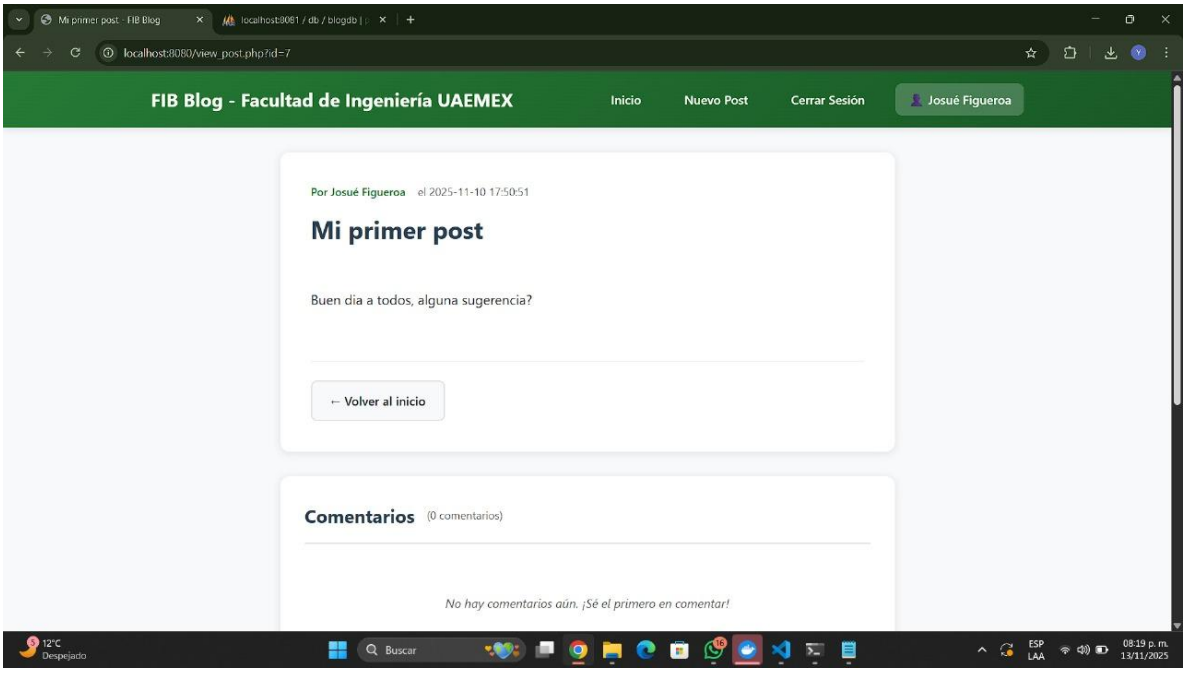
- Apartado de crear cuenta.





Muestra el apartado de la creación de cuenta y que campos se deben llenar para tener la cuenta creada.

- Apartado perfil para revisar las publicaciones realizadas.



Muestra el apartado que realizo la cuenta en la historia que lleva la cuenta creada.

Arquitectura general del sistema

El proyecto está compuesto por tres servicios:

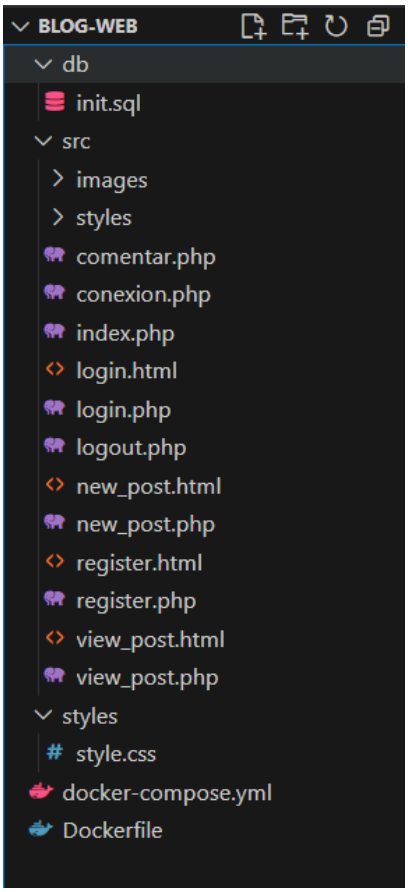
- Web (PHP + Apache)
    - Ejecuta el código del blog.
    - Atiende peticiones HTTP en el puerto 8080.
- Base de datos (MySQL 8)
    - Almacena usuarios, posts y comentarios.
    - Utiliza un volumen Docker para persistencia.
- phpMyAdmin
    - Cliente web para administrar la base.
    - Disponible en el puerto 8081.

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

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
```



Base de datos del proyecto (Modelo entidad-relación)

- usuarios
  - id (PK)
  - nombre
  - email
  - password (hash)
- posts
  - id (PK)
  - id_usuario (FK)

- titulo
- contenido
- fecha

- comentarios
  - id (PK)
  - id_post (FK)
  - id_usuario (FK)
  - contenido
  - fecha

Código utilizado solo los fragmentos clave

La conexión a la base de datos a docker se realiza mediante el comando:

$conexion = new mysqli("db", "Yael", "password", "blogdb");

Este fragmento establece una conexión desde la aplicación PHP hacia el contenedor de MySQL ejecutado dentro de Docker. 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. Los parámetros restantes especifican el usuario, la contraseña y el nombre de la base de datos, completando el proceso de autenticación dentro del contenedor de MySQL.

El archivo Dockerfile se encarga de construir la imagen personalizada para el servicio web. En este proyecto se utiliza como base la imagen php:8.2-apache, que integra el servidor Apache junto con PHP en su versión 8.2. Dentro del Dockerfile se instalan extensiones necesarias para el funcionamiento de la aplicación —principalmente la extensión mysqli— lo que permite que PHP pueda comunicarse adecuadamente con el servidor MySQL alojado en otro contenedor.

Por su parte, el archivo docker-compose.yml define la arquitectura completa del entorno. En él se declaran los tres servicios fundamentales: el servidor web (PHP + Apache), la base de datos MySQL y el panel de administración phpMyAdmin. También se especifican los puertos expuestos para el acceso externo, así como los volúmenes utilizados para persistir datos y permitir el desarrollo en caliente (hot reload). Este

archivo actúa como el orquestador principal, garantizando que los servicios se construyan, configuren e inicien de forma coherente y automatizada.

Dockerfile

El Dockerfile es el archivo encargado de construir la imagen personalizada que utilizará el servicio web. En este proyecto, se emplea como base la imagen oficial php:8.2-apache, la cual integra el servidor Apache junto con la versión 8.2 de PHP. Dentro del Dockerfile se instalan las dependencias necesarias para que la aplicación funcione correctamente, en particular la extensión mysqli, que permite a PHP establecer conexiones con el servidor MySQL. En conjunto, el Dockerfile define el entorno exacto en el que se ejecutará la aplicación, asegurando consistencia, compatibilidad y reproducibilidad en cualquier sistema donde se implemente.

docker-compose.yml

El archivo *docker-compose.yml* actúa como el orquestador del sistema, ya que define y coordina todos los servicios que conforman la aplicación. En este caso, especifica tres servicios principales:

- el servidor web (PHP + Apache),
- el motor de base de datos MySQL,
- y la herramienta de administración phpMyAdmin.

Además, establece los puertos necesarios para permitir el acceso externo a cada servicio y configura los volúmenes que permiten tanto la persistencia de datos como el montaje del código fuente, facilitando el desarrollo sin necesidad de reconstruir la imagen en cada cambio. En esencia, este archivo automatiza la creación, configuración y ejecución de todo el entorno del blog académico dentro de Docker.

Explicación de la dockerización.

Elemento	Función
Dockerfile	Construye la imagen con Apache + PHP + extensiones necesarias
docker-compose.yml	Orquesta los contenedores del proyecto
Volumen MySQL	Permite que la BD no se borre al apagar contenedores
Bind-mount del código	Permite editar en VSCode mientras se ejecuta
Red interna Docker	Conecta PHP ↔ MySQL con el hostname db

Funcionamiento.

Al ejecutar el comando:

docker-compose up -d

Docker inicia automáticamente todo el entorno definido en *docker-compose.yml*. Durante este proceso, se realizan las siguientes acciones:

1. Construcción de la imagen PHP–Apache:
   Docker genera la imagen personalizada a partir del *Dockerfile*, instalando PHP 8.2 junto con el servidor Apache y la extensión *mysqli* requerida por la aplicación.
2. Inicialización del servicio MySQL:
   Se levanta el contenedor de la base de datos utilizando las credenciales, el nombre de la base de datos y el archivo de inicialización especificados en la configuración.
3. Puesta en marcha de phpMyAdmin:
   Se inicia la herramienta de administración web que permite gestionar la base de datos de manera visual, conectándose automáticamente al contenedor de MySQL.
4. Creación de una red interna de comunicación:
   Docker genera una red virtual donde todos los contenedores pueden

comunicarse entre sí mediante sus nombres de servicio, garantizando aislamiento y conectividad estable.

5. Exposición de puertos al sistema anfitrión: Docker mapea los puertos internos de los contenedores hacia el host, permitiendo acceder a los servicios desde el navegador:

Servicio	URL
Blog	http://localhost:8080
phpMyAdmin	http://localhost:8081

En conjunto, este proceso automatiza la construcción, configuración y despliegue de toda la aplicación, asegurando un entorno coherente, portable y completamente funcional con un solo comando.

Discusión del proyecto.

El funcionamiento del sistema fue validado mediante diversas pruebas operativas que incluyeron el registro de nuevos usuarios, el inicio de sesión con credenciales previamente almacenadas, la creación de publicaciones y el envío de comentarios asociados a dichas entradas. En todos los casos, la información generada por la aplicación se almacenó correctamente en la base de datos MySQL y pudo ser visualizada, modificada y administrada sin inconvenientes a través de phpMyAdmin, confirmando así la integridad del flujo completo de datos.

Uno de los resultados más relevantes obtenidos durante las pruebas fue la verificación de que la conexión a MySQL no debe realizarse empleando “localhost”, sino utilizando el nombre del servicio "db", tal como está definido en el archivo docker-compose.yml. Este ajuste es fundamental en entornos basados en Docker, ya que cada contenedor opera dentro de una red interna propia, en la cual los servicios se identifican por sus nombres y no por direcciones IP locales. Al establecer la conexión mediante el identificador “db”, el servidor PHP logró localizar correctamente el contenedor de MySQL y establecer la comunicación necesaria, asegurando así el funcionamiento adecuado del sistema dentro de la arquitectura de contenedores.

Se observó que el uso de volúmenes en el servicio de MySQL garantiza la persistencia de los datos, independientemente de cuántas veces se reinicien o se reconstruyan los contenedores, lo cual asegura la continuidad de la información incluso durante procesos de mantenimiento o actualización. Asimismo, el blog pudo ejecutarse

correctamente tanto en Windows como en Linux sin requerir modificaciones adicionales, validando de manera directa el objetivo planteado de lograr un entorno completamente portable.

La principal tendencia identificada durante el desarrollo es que un entorno web complejo puede ser encapsulado de manera sencilla mediante contenedores, lo que coincide plenamente con las prácticas modernas de despliegue utilizadas en la industria. Este enfoque permitió mejorar de forma significativa el flujo de trabajo, eliminando los errores recurrentes asociados a herramientas tradicionales como XAMPP y evitando conflictos de versiones entre configuraciones locales.

Por otra parte, se destacó la importancia de estructurar adecuadamente la aplicación, separando módulos de funcionalidad como el inicio de sesión, las vistas y controladores básicos. Del mismo modo, se resaltó la necesidad de mantener una base de datos organizada, de manera que el sistema pueda admitir futuras mejoras, tales como la edición de publicaciones, la actualización de perfiles o la ampliación de funcionalidades sin comprometer la integridad del proyecto.

Conclusión.

A lo largo del desarrollo y la dockerización del blog académico, fue posible cumplir plenamente con el objetivo planteado para este proyecto. La aplicación demostró ser funcional, estable y capaz de ejecutarse de manera uniforme en distintos sistemas operativos mediante el uso de Docker, lo que confirma su carácter portable y reproducible. Este resultado valida la elección de los contenedores como la solución más adecuada para la problemática inicial, la cual se centraba en las dificultades asociadas a las dependencias locales, las configuraciones específicas de cada equipo y la falta de compatibilidad entre entornos tradicionales de desarrollo como XAMPP o WAMP.

El uso de Docker permitió abstraer estas limitaciones al encapsular todos los servicios necesarios —PHP, Apache, MySQL y phpMyAdmin— dentro de contenedores aislados, facilitando el despliegue, el mantenimiento y la reproducibilidad del sistema, sin requerir instalaciones complejas ni configuraciones adicionales por parte del usuario. Este enfoque se alinea con las prácticas modernas de la industria, donde el empleo de contenedores se ha convertido en un estándar para garantizar entornos consistentes y altamente controlados.

Asimismo, durante el proceso de implementación se evidenció la importancia de definir una estructura clara dentro de la aplicación, separar responsabilidades por

módulos y mantener una base de datos organizada, aspectos esenciales para asegurar la escalabilidad del sistema y su capacidad de evolucionar de manera ordenada. A partir de esta experiencia, se identificaron diversas oportunidades de mejora que podrían potenciar las funcionalidades del blog, entre ellas la implementación de un sistema de roles para diferenciar privilegios entre administradores y usuarios, la incorporación de herramientas para el manejo de archivos e imágenes, y la adopción de una arquitectura más formal basada en el patrón MVC con el fin de mejorar la estructura del proyecto y fortalecer su mantenibilidad.

En conjunto, el desarrollo del blog académico no solo permitió cumplir los objetivos establecidos, sino que también proporcionó una comprensión más profunda sobre el uso de contenedores y la importancia de un diseño modular y bien estructurado. Los resultados obtenidos confirman que Docker es una herramienta efectiva para construir entornos portables y robustos, y que su aplicación en proyectos web representa una mejora significativa frente a los métodos tradicionales de despliegue.

Como posible mejora futura, se propone:

- Implementar un sistema de roles (administrador y usuarios).
- Agregar manejo de archivos e imágenes.
- Integrar un sistema MVC básico para mejorar la estructura del proyecto.

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.