



## CONTENIDO

Introducción:.....	2
Principales adversidades en la creación del código:.....	2
Partes principales del código: .....	3
Partes más importantes del código y sus detalles:.....	7
Aplicación:.....	9
Inicio:.....	9
Ver entradas: .....	10
Crear entradas: .....	10
Registro:.....	11
Inicio de sesión:.....	11
Editar perfil: .....	11
Próximas mejoras: .....	12
Conclusión:.....	13
Bibliografía:.....	13

## INTRODUCCIÓN:

Nuestra aplicación es un blog en línea que permite a los usuarios registrados publicar y compartir sus entradas con otros usuarios. Hemos creado esta aplicación utilizando tecnologías web estándar, como HTML, CSS, JavaScript y PHP, junto con una base de datos MySQL para almacenar los datos.

Los usuarios pueden registrarse en la plataforma proporcionando un nombre de usuario, correo electrónico y contraseña. Una vez registrados, pueden iniciar sesión en sus cuentas para acceder a su perfil, donde pueden ver y editar su información personal.

Además, los usuarios pueden crear nuevas entradas en el blog, que consisten en un título, contenido de texto y opcionalmente una URL de imagen. Estas entradas se almacenan en la base de datos y se muestran en la página principal del blog, donde otros usuarios pueden verlas y comentarlas.

Para garantizar la seguridad de la aplicación, hemos implementado medidas como el almacenamiento seguro de contraseñas utilizando algoritmos de hashing, la validación de datos de entrada para prevenir ataques de inyección de SQL y la protección de sesiones contra ataques de falsificación de solicitudes entre sitios.

En resumen, nuestra aplicación proporciona una plataforma simple y segura para que los usuarios compartan sus ideas, experiencias y pensamientos a través de un blog en línea.

## PRINCIPALES ADVERSIDADES EN LA CREACIÓN DEL CÓDIGO:

1. **Errores de sintaxis y lógica:** Es común cometer errores de sintaxis o lógica al escribir el código. La solución es revisar el código cuidadosamente y utilizar herramientas de depuración para identificar y corregir estos errores.
2. **Problemas de conexión a la base de datos:** La conexión a la base de datos puede fallar debido a credenciales incorrectas, problemas de red o configuraciones incorrectas. Es importante verificar las credenciales de conexión y asegurarse de que el servidor de la base de datos esté en funcionamiento.
3. **Errores al manipular datos del formulario:** Al manipular datos del formulario, como datos de usuario o de entrada, pueden surgir errores al acceder a los valores o al procesarlos incorrectamente. Es esencial realizar una validación adecuada de los datos del formulario y manejar posibles valores nulos o vacíos de manera segura.
4. **Problemas de seguridad:** Pueden surgir vulnerabilidades de seguridad, como la inyección de SQL o la exposición de datos confidenciales, si no se implementan medidas de seguridad adecuadas. Es crucial utilizar consultas preparadas y funciones de hash seguras para evitar estos problemas.
5. **Problemas de rendimiento:** A medida que la aplicación crece en complejidad y volumen de datos, pueden surgir problemas de rendimiento. Optimizar consultas de base de datos, caché de datos y mejorar la eficiencia del código son formas de abordar estos problemas.
6. **Errores de diseño de la interfaz de usuario:** Los problemas con el diseño de la interfaz de usuario pueden surgir al intentar que la aplicación sea fácil de usar y estéticamente atractiva. Recopilar comentarios de los usuarios y realizar pruebas de usabilidad pueden ayudar a identificar y solucionar estos problemas.



## PARTES PRINCIPALES DEL CÓDIGO:

### 1) Inicio (inicio.php):

- Este archivo es la página de inicio de tu aplicación web.
- Almacena la dirección IP del usuario y la fecha de acceso en cookies.
- Contiene un texto explicativo sobre diferentes enfoques de programación.
- Muestra una imagen relacionada con la programación orientada a objetos.

```
// PHP
// Almacenar la IP del usuario y la fecha de acceso en una cookie
$ip = $_SERVER['REMOTE_ADDR'];
$date = date('format: Y-m-d H:i:s');
setcookie('ip', $ip, time() + (86400 * 30), '/'); // 86400 = 1 día
setcookie('date', $date, time() + (86400 * 30), '/');

// Incluir el texto explicativo
<?php
<doctype html>
<html>
<head>
<title>Inicio</title>
<!-- CSS de Bootstrap -->
<link href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">MI Blog</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="navbar-nav">
      <li class="nav-item active">
        <a class="nav-link" href="inicio.php">Inicio</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="view_post.php">Ver entradas</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="index.php">Crear entradas</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="register.php">Registrar</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="login.php">Iniciar sesión</a>
      </li>
    </ul>
  </div>
</nav>
```

### 2) Iniciar Sesión (login.php):

- Proporciona un formulario para que los usuarios inicien sesión en la aplicación.
- Los usuarios deben ingresar su correo electrónico y contraseña.
- Una vez enviado el formulario, los datos se procesan en "login\_process.php".

```
</nav>
<div class="container">
  <h1 class="my-4">Iniciar sesión</h1>
  <form action="login_process.php" method="post">
    <div class="form-group">
      <label for="email">Correo electrónico:</label>
      <input type="email" id="email" name="email" class="form-control" required>
    </div>
    <div class="form-group">
      <label for="password">Contraseña:</label>
      <input type="password" id="password" name="password" class="form-control" required>
    </div>
    <button type="submit" class="btn btn-primary">Iniciar sesión</button>
    <a href="edit_profile.php" class="btn btn-secondary">Modificar perfil</a>
  </form>
</div>
</div>
<!-- JS de Bootstrap -->
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
</body>
</html>
```

### 3) Proceso de Inicio de Sesión (login\_process.php):

- Verifica las credenciales del usuario en la base de datos.
- Inicia una sesión para el usuario si las credenciales son válidas.
- Establece cookies para recordar al usuario si elige la opción "Recordar".

```
<?php
// Conectar a la base de datos
$db = new mysqli(hostname: 'localhost', username: 'root', password: 'curso', database: 'blog');

// Comprobar la conexión
if ($db->connect_error) {
    die("La conexión falló: " . $db->connect_error);
}

// Recibir los datos del formulario
$email = $_POST['email'];
$password = $_POST['password'];

// Preparar la consulta SQL para obtener la información del usuario
$stmt = $db->prepare("SELECT * FROM users WHERE email = ?");
$stmt->bind_param("s", $email);

// Ejecutar la consulta
$stmt->execute();

// Obtener los resultados
$result = $stmt->get_result();

// Comprobar si se encontró el usuario
if ($result->num_rows > 0) {
    $user = $result->fetch_assoc();

    // Comprobar la contraseña
    if (password_verify($password, $user['password'])) {
        // Iniciar la sesión
        session_start();
        $_SESSION['user_id'] = $user['id'];

        // Establecer las cookies para recordar al usuario
        setcookie("user_id", $user['id'], time() + (86400 * 30), "/"); // 86400 = 1 día
        setcookie("user_key", hash("sha256", $user['password']), time() + (86400 * 30), "/");

        // Redirigir al usuario a la página de inicio
        header("location: inicio.php");
        exit;
    }
}
```

### 4) Gestionar Entradas (manage\_post.php):

- Muestra una lista de todas las entradas de blog existentes.
- Permite a los usuarios actualizar o eliminar las entradas existentes.

```
<?php
<body>
<div class="container">
<h1 class="my-4">Gestionar entradas</h1>
<?php while ($row = $result->fetch_assoc()): ?>
<div class="card mb-4">
<div class="card-body">
<h2 class="card-title"><? $row['title'] ?></h2>
<p class="card-text"><? $row['content'] ?></p>
<p class="card-text"><small class="text-muted">Publicado por <? $row['email'] ?> el <? $row['publish_date'] ?></small></p>
</div>
<?php if (empty($row['image'])): ?>

<?php endif; ?>
<div class="card-footer">
<a href="update_post.php?id=<? $row['id'] ?>" class="btn btn-primary">Actualizar</a>
<a href="delete_post.php?id=<? $row['id'] ?>" class="btn btn-danger">Eliminar</a>
</div>
</div>
<?php endwhile; ?>
</div>
<!-- JS de Bootstrap -->
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
</body>
</html>
```

## 5) Perfil de Usuario (profile.php):

- Permite a los usuarios ver y modificar su información de perfil.
- Muestra el nombre de usuario y el correo electrónico actuales del usuario.
- Los usuarios pueden actualizar su nombre de usuario y correo electrónico.

```
<?php
include_once 'functions.php';
check_user_logged_in();

// Conectar a la base de datos
mysqli = new mysqli(hostname: 'localhost', username: 'root', password: 'curso', database: 'blog');

// Comprobar la conexión
if ($mysqli->connect_error) {
    die('La conexión falló: ' . $mysqli->connect_error);
}

// Obtener el ID del usuario logueado
$user_id = $_SESSION['user_id'];

// Preparar la consulta SQL para obtener la información del usuario
$stmt = $mysqli->prepare( query: "SELECT * FROM users WHERE id = ?");
$stmt->bind_param( type: 'i', &$user_id);

// Ejecutar la consulta
$stmt->execute();

// Obtener los resultados
$result = $stmt->get_result();

// Obtener la información del usuario
$user = $result->fetch_assoc();

// Cerrar la conexión
mysqli->close();

<body>
<div class="container">
    <h1 class="my-4">Perfil de usuario</h1>
    <form action="update_profile.php" method="post">
        <div class="form-group">
            <label for="username">Nombre de usuario</label>
            <input type="text" class="form-control" id="username" name="username" value="<?= $user['username'] ?>">
        </div>
        <div class="form-group">
            <label for="email">Correo electrónico</label>
            <input type="email" id="email" name="email" class="form-control" value="<?= $user['email'] ?>">
        </div>
        <button type="submit" class="btn btn-primary">Actualizar perfil</button>
    </form>
</div>
```

## 6) Publicar (publish.php):

- Proporciona un formulario para que los usuarios publiquen nuevas entradas en el blog.
- Los usuarios deben ingresar un título, contenido, fecha de publicación e URL de la imagen.
- Una vez enviado el formulario, los datos se procesan en "publish\_process.php".

```
// Recibir los datos del formulario
$email = $_POST['email'];
$title = $_POST['title'];
$content = $_POST['content'];
$publish_date = $_POST['publish_date'];
$image = $_POST['image']; // Aquí recibimos la URL de la imagen

// Validar los datos (aquí puedes agregar más validaciones si lo deseas)
if (empty($email) || empty($title) || empty($content) || empty($publish_date) || empty($image)) {
    die("Por favor, completa todos los campos.");
}

// Preparar la consulta SQL para insertar la nueva entrada
$stmt = $mysqli->prepare( query: "INSERT INTO posts (email, title, content, publish_date, image) VALUES (?, ?, ?, ?, ?)");
$stmt->bind_param( type: 'sssss', &$email, &$title, $content, $publish_date, $image);

// Ejecutar la consulta
if ($stmt->execute()) {
    echo "Tu entrada ha sido publicada.";
} else {
    echo "Hubo un error al publicar tu entrada.";
}
```

## 7) Registro (register.php):

- Permite a los nuevos usuarios registrarse en la aplicación.
- Los usuarios deben proporcionar un nombre, correo electrónico y contraseña.
- Una vez enviado el formulario, los datos se procesan en "register\_process.php".

```
</nav>
<div class="container">
  <h1 class="my-4">Registrarse</h1>
  <form action="register_process.php" method="post">
    <div class="form-group">
      <label for="name">Nombre:</label>
      <input type="text" id="name" name="name" class="form-control" required>
    </div>
    <div class="form-group">
      <label for="email">Correo electrónico:</label>
      <input type="email" id="email" name="email" class="form-control" required>
    </div>
    <div class="form-group">
      <label for="password">Contraseña:</label>
      <input type="password" id="password" name="password" class="form-control" required>
    </div>
    <input type="submit" value="Registrarse" class="btn btn-primary">
  </form>
</div>
```

## 8) Actualizar Entrada (update\_post.php y update\_post\_process.php):

- Permite a los usuarios actualizar una entrada existente en el blog.
- El archivo "update\_post.php" muestra un formulario con los detalles de la entrada.
- El archivo "update\_post\_process.php" procesa los datos enviados desde el formulario y actualiza la entrada en la base de datos.

```
// Recibir los datos del formulario
$id = $_POST['id'];
$title = $_POST['title'];
$content = $_POST['content'];
$publish_date = $_POST['publish_date'];
$image = $_POST['image']; // Ahora es una URL

// Preparar la consulta SQL para actualizar la entrada
$stmt = $mysqli->prepare("UPDATE posts SET title = ?, content = ?, publish_date = ?, image = ? WHERE id = ?");
$stmt->bind_param("ssssi", $id, $title, $content, $publish_date, $image, $id);

// Ejecutar la consulta
if ($stmt->execute()) {
  echo "La entrada ha sido actualizada.";
} else {
  echo "Hubo un error al actualizar la entrada.";
}

// Cerrar la conexión
$stmt->close();
$mysqli->close();
exit;
```

## 9) Ver Entradas (view\_post.php):

- Muestra una lista de todas las entradas de blog existentes.
- Cada entrada muestra su título, contenido, autor, fecha de publicación y una imagen (si está disponible).

```
<div class="container">
  <h1 class="my-4">Entradas del blog</h1>
  <?php while ($row = $result->fetch_assoc()): ?>
    <div class="card mb-4">
      <div class="card-body">
        <h2 class="card-title"><?php echo $row['title']; ?></h2>
        <p class="card-text"><?php echo $row['content']; ?></p>
        <p class="card-text"><small class="text-muted">Publicado por <?php echo $row['email']; ?> el <?php echo $row['publish_date']; ?></small></p>
      </div>
      <?php if (empty($row['image'])): ?>
        <img src="" alt="Imagen de la entrada" class="card-img-bottom" />
      <?php endif: ?>
    </div>
    <?php while ($row = $result->fetch_assoc()): ?>
      <div class="card mb-4">
        <div class="card-body">
          <h2 class="card-title"><?php echo $row['title']; ?></h2>
          <p class="card-text"><?php echo $row['content']; ?></p>
          <p class="card-text"><small class="text-muted">Publicado por <?php echo $row['email']; ?> el <?php echo $row['publish_date']; ?></small></p>
          <div>
            <a href="update_post.php?id=<?php echo $row['id']; ?>">Editar</a> | <a href="delete_post.php?id=<?php echo $row['id']; ?>">Eliminar</a>
          </div>
        </div>
        <?php if (empty($row['image'])): ?>
          <img src="" alt="Imagen de la entrada" class="card-img-bottom" />
        <?php endif: ?>
      </div>
    </div>
  </div>
</div>
```

## PARTES MÁS IMPORTANTES DEL CÓDIGO Y SUS DETALLES:

### 1) Conexión a la Base de Datos

- En varios archivos, como "login\_process.php", "register\_process.php", "manage\_post.php", entre otros, se establece una conexión a la base de datos MySQL utilizando la clase mysqli.
- Se comprueba si la conexión es exitosa. Si no lo es, el script muestra un mensaje de error y finaliza la ejecución.

```
// Conectar a la base de datos
$mysqli = new mysqli( hostname: 'localhost', username: 'root', password: 'curso', database: 'blog');

// Comprobar la conexión
if ($mysqli->connect_error) {
    die("La conexión falló: " . $mysqli->connect_error);
}
```

### 2) Manejo de Sesiones:

- En los archivos que requieren autenticación, como "login\_process.php" y "profile.php", se inicia una sesión utilizando session\_start() si aún no se ha iniciado.
- Después de iniciar sesión correctamente, se establece el ID de usuario en la sesión para mantener al usuario autenticado durante su visita al sitio.

```
session_start();
$_SESSION['user_id'] = $user['id'];
```

### 3) Registro de Usuarios (register.php y register\_process.php):

- En "register.php", se muestra un formulario para que los usuarios se registren proporcionando su nombre, correo electrónico y contraseña.
- En "register\_process.php", se reciben los datos del formulario y se validan para garantizar que ningún campo esté vacío.
- La contraseña proporcionada se cifra utilizando la función password\_hash() antes de almacenarla en la base de datos para mayor seguridad.
- Si el registro es exitoso, se muestra un mensaje de confirmación al usuario.

```
$hashed_password = password_hash($password, algo: PASSWORD_DEFAULT);
```

### 4) Iniciar Sesión de Usuarios (login.php y login\_process.php):

- "login.php" muestra un formulario de inicio de sesión donde los usuarios ingresan su correo electrónico y contraseña.
- En "login\_process.php", se recuperan los datos del formulario y se verifican en la base de datos para autenticar al usuario.
- Si las credenciales son válidas, se inicia una sesión para el usuario y se establecen cookies para recordarlo en futuras visitas.



```
// Comprobar la contraseña
if (password_verify($password, $user['password'])) {
    // Iniciar la sesión
    session_start();
    $_SESSION['user_id'] = $user['id'];

    // Establecer las cookies para recordar al usuario
    setcookie("user_id", $user['id'], time() + (86400 * 30), "/"); // 86400 = 1 día
    setcookie("user_key", hash('sha256', $user['password']), time() + (86400 * 30), "/");
}
```

**5) Publicación de Entradas (publish.php y publish\_process.php):**

- "publish.php" muestra un formulario para que los usuarios publiquen nuevas entradas en el blog.
- "publish\_process.php" recibe los datos del formulario y los procesa para insertar una nueva entrada en la base de datos.
- Se valida que todos los campos del formulario estén completos antes de procesar la publicación.

```
// Preparar la consulta SQL para insertar la nueva entrada
$stmt = $mysqli->prepare( query: "INSERT INTO posts (email, title, content, publish_date, image) VALUES (?, ?, ?, ?, ?)");
$stmt->bind_param( types: 'sssss', &var1: $email, &var2: $title, $content, $publish_date, $image);
```

**6) Gestión de Entradas (manage\_post.php):**

- Este archivo muestra una lista de todas las entradas existentes en el blog.
- Cada entrada se presenta con opciones para editar o eliminar, lo que permite a los usuarios gestionar su contenido.

```
<?php while ($row = $result->fetch_assoc()): ?>
```

**7) Actualizar Entradas (update\_post.php y update\_post\_process.php):**

- "update\_post.php" muestra un formulario prellenado con los detalles de una entrada existente para que los usuarios la actualicen.
- "update\_post\_process.php" procesa los datos del formulario enviado por los usuarios y actualiza la entrada correspondiente en la base de datos.

```
// Preparar la consulta SQL para actualizar la entrada
$stmt = $mysqli->prepare( query: "UPDATE posts SET title = ?, content = ?, publish_date = ?, image = ? WHERE id = ?");
$stmt->bind_param( types: 'ssssi', &var1: $title, &var2: $content, $publish_date, $image, $id);
```

## APLICACIÓN:

## INICIO:

[Mi Blog](#) [Inicio](#) [Ver entradas](#) [Crear entradas](#) [Registro](#) [Iniciar sesión](#)

## Bienvenido

Aquí va la explicación de las diferencias entre lenguajes de programación orientada a objetos, a eventos y lenguajes procedimentales:

**Procedimental:** En la programación procedimental, el énfasis está en los procedimientos o funciones que manipulan los datos. El código se organiza en una secuencia de instrucciones, donde se definen funciones que realizan tareas específicas. Los datos se pasan entre estas funciones como parámetros, y las variables pueden tener un alcance global o local. Ejemplos de lenguajes procedimentales son C y Pascal.

**Orientada a Objetos:** La programación orientada a objetos (POO) se basa en el concepto de "objetos", que son entidades que contienen datos en forma de atributos y procedimientos en forma de métodos. Estos objetos interactúan entre sí mediante el intercambio de mensajes. La POO se centra en la encapsulación, la herencia y el polimorfismo como principios fundamentales. Ejemplos de lenguajes orientados a objetos son Java, Python y C++.

**Basada en Eventos:** En la programación basada en eventos, el flujo de control del programa está determinado por eventos que ocurren en el sistema, como clics de ratón, pulsaciones de teclas o temporizadores. El programa responde a estos eventos ejecutando ciertas acciones o manejadores de eventos asociados a cada evento. Este enfoque es común en el desarrollo de interfaces de usuario y aplicaciones que requieren una respuesta en tiempo real. Ejemplos de lenguajes que admiten programación basada en eventos incluyen JavaScript (para desarrollo web), Visual Basic y C# (para desarrollo de aplicaciones de escritorio).

### Esquema de diferencias de programación estructurada y programación orientada a objetos

Programación estructurada	programación orientada a objetos
1. Los programas son más fáciles de entender.	1. Fomenta la reutilización y extensión del código.
2. Reducción del esfuerzo en las pruebas.	2. Permite crear sistemas más complejos.
3. Programas más sencillos y más rápidos.	3. Relacionar el sistema al mundo real.
4. Aumento de la productividad del programador.	4. Facilita la creación de programas visuales.
5. Se facilita la utilización de las otras técnicas para el mejoramiento de la productividad en programación.	5. Construcción de prototipos
6. Los programas quedan mejor documentados internamente.	6. Agiliza el desarrollo de software
	7. Facilita el trabajo en equipo
	8. Facilita el mantenimiento del software

CONCLUSION: En mi punto de vista cada una tiene ventajas y desventajas pero ambas son útiles para uno u otro trabajo.  
Yared Susette Fernández García.


VER ENTRADAS:

titulo 1

contenido 1

Publicado por ltsl@gmail.com el 2024-02-16

[Editar](#) | [Eliminar](#)



CREAR ENTRADAS:

Publicar entrada

Email:

Título:

Contenido:

Fecha de publicación:

dd/mm/aaaa

Imagen:

Introduce la URL de la imagen

Publicar

REGISTRO:

# Registrarse

Nombre:

Correo electrónico:

Contraseña:

Registrarse

INICIO DE SESIÓN:

# Iniciar sesión

Correo electrónico:

Contraseña:

Iniciar sesión

Modificar perfil

EDITAR PERFIL:

# Editar perfil

Nombre de usuario

Correo electrónico

Actualizar perfil

Cerrar sesión

#### PRÓXIMAS MEJORAS:

1. **Seguridad de la contraseña:** Aunque el código utiliza la función `password_hash()` para almacenar contraseñas de forma segura, siempre es importante asegurarse de que se están siguiendo las mejores prácticas de seguridad en el manejo de contraseñas y en la protección contra ataques de fuerza bruta.
2. **Validación de datos de entrada:** Aunque se realizan algunas validaciones de datos de entrada en los formularios, podría ser necesario implementar más validaciones para evitar problemas como la inyección de SQL o la introducción de datos maliciosos.
3. **Protección contra ataques de sesión:** La gestión de sesiones es crítica para la seguridad de la aplicación. Es importante asegurarse de que las sesiones estén bien protegidas contra ataques como la suplantación de identidad o la manipulación de cookies.
4. **Manejo de errores y excepciones:** El código podría beneficiarse de una gestión más robusta de errores y excepciones para manejar de manera adecuada situaciones inesperadas y evitar la exposición de información sensible en caso de fallos.
5. **Escalabilidad y rendimiento:** A medida que el blog crezca en contenido y usuarios, podría surgir la necesidad de optimizar la base de datos y el código para mejorar el rendimiento y garantizar una experiencia de usuario fluida.
6. **Actualizaciones de seguridad y mantenimiento:** Es importante estar al tanto de las actualizaciones de seguridad de las herramientas y bibliotecas utilizadas en el proyecto y realizar un mantenimiento regular para mantener la aplicación segura y funcionando sin problemas.



## CONCLUSIÓN:

Al desarrollar esta aplicación web, nos hemos enfrentado a diversos desafíos y hemos encontrado soluciones que han permitido mejorar la funcionalidad y la experiencia del usuario. A lo largo del proceso, hemos identificado tanto ventajas como desventajas en el diseño y la implementación del código.

Entre las ventajas principales, destaca la capacidad de proporcionar una plataforma interactiva y dinámica donde los usuarios pueden registrar, iniciar sesión, publicar y gestionar entradas en un blog personalizado. La aplicación también permite la administración eficiente de la información del usuario y la entrada de datos a través de formularios intuitivos.

No obstante, durante el desarrollo del código, nos hemos encontrado con ciertas adversidades. Por ejemplo, la gestión de sesiones y la seguridad de contraseñas han sido aspectos críticos que hemos abordado con cuidado para garantizar la privacidad y la integridad de los datos de los usuarios. Además, hemos tenido que enfrentar desafíos técnicos relacionados con la conexión a la base de datos, la validación de formularios y el manejo de errores.

A pesar de estos obstáculos, hemos logrado superarlos mediante una combinación de investigación, prueba y error, y optimización del código. La aplicación ahora funciona de manera eficiente y segura, brindando una experiencia satisfactoria tanto para los usuarios como para los administradores.

En resumen, el desarrollo de esta aplicación web ha sido un proceso enriquecedor que nos ha permitido aprender y mejorar nuestras habilidades de programación. A través de la identificación y resolución de problemas, hemos creado una aplicación funcional y robusta que cumple con los objetivos establecidos.

## BIBLIOGRAFÍA:

### ChatGPT:

OpenAI. (s.f.). ChatGPT. Recuperado de <https://chat.openai.com/auth/login>

### PHP:

PHP Manual. (s.f.). PHP: Documentación. Recuperado de <https://www.php.net/manual/es/index.php>

### Copilot:

GitHub. (s.f.). Copilot: Your AI pair programmer. Recuperado de <https://github.com/features/copilot>

### PHPStorm:

JetBrains. (s.f.). PhpStorm: The Lightning-Smart IDE for PHP Programming by JetBrains. Recuperado de <https://www.jetbrains.com>

### MySQL Workbench:

MySQL. (s.f.). MySQL Workbench. Recuperado de <https://www.mysql.com/products/workbench/>

### GitHub:

GitHub. (s.f.). GitHub: Where the world builds software. Recuperado de <https://github.com/>