DOCUMENTACIÓN

¿Qué es PHP? ¿Qué relación tiene con Apache? ¿Dónde se usa Apache+PHP+MariaDB/MySQL? Documenta tu respuesta y busca varios ejemplos de webs donde esté presente esta combinación.

PHP es un lenguaje de programación de código abierto, especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. Apache es un servidor web que se utiliza para alojar sitios web y aplicaciones web. PHP se ejecuta en el servidor y Apache es el servidor que maneja las solicitudes HTTP y ejecuta los scripts PHP.

La combinación de Apache, PHP y MariaDB/MySQL es muy común en el desarrollo web. MariaDB y MySQL son sistemas de gestión de bases de datos relacionales que se utilizan para almacenar y gestionar datos. Esta combinación se usa en muchas aplicaciones web debido a su flexibilidad, eficiencia y facilidad de uso.

Ejemplos de webs que usan esta combinación:

- WordPress: Un sistema de gestión de contenidos (CMS) muy popular.
- Drupal: Otro CMS ampliamente utilizado.
- Joomla: Un CMS que permite crear sitios web y aplicaciones online.

¿Qué es GET? ¿Y POST? ¿Cómo accedemos a los datos enviados por estos canales?

GET y POST son métodos de solicitud HTTP utilizados para enviar datos desde el cliente al servidor.

- GET: Envía datos a través de la URL. Los datos se pueden acceder en PHP usando $_GET.
- POST: Envía datos a través del cuerpo de la solicitud HTTP. Los datos se pueden acceder en PHP usando $_POST.

¿Qué son las sesiones? ¿Para qué sirven? ¿Cómo funcionan?

Las sesiones en PHP permiten almacenar datos en el servidor para ser utilizados en múltiples páginas. Sirven para mantener el estado de un usuario a lo largo de su navegación en un sitio web, como mantenerlo logueado.

Funcionamiento:

1. Se inicia la sesión con session_start().
2. Se almacenan datos en la sesión usando $_SESSION.
3. Los datos se pueden acceder en cualquier página mientras la sesión esté activa.

¿Cómo insertamos datos en la base de datos mediante PHP?

Para insertar datos en una base de datos mediante PHP, generalmente se utilizan las funciones de PDO (PHP Data Objects) o mysqli para interactuar con la base de datos.

Ejemplo en dashboard_games_add.php:

```php
$query = "SELECT * FROM creators WHERE id_creator=" . $_SESSION["id_creator"];
$conn = mysqli_connect($db_server, $db_user, $db_pass, $db_name);

if (!$conn) {
    die("ERROR: No se pudo conectar a la base de datos.");
}

$result = mysqli_query($conn, $query);

if (!$result || mysqli_num_rows($result) != 1) {
    header("Location: login.php");
    exit();
}

$creator = mysqli_fetch_array($result);

printHead("Dashboard de " . $creator["name"]);
openBody("Dashboard de " . $creator["name"]);
require_once("dashboard_template.php");
openDashboard();
```

EXPLICACION DEL CODIGO

1. **db_config.php**

Este archivo contiene la configuración de la base de datos para la aplicación. Define las variables necesarias para conectarse a la base de datos MySQL.

Variables

- $db_server: La dirección del servidor de la base de datos. En este caso, es "localhost", lo que significa que la base de datos está alojada en el mismo servidor que la aplicación.
- $db_user: El nombre de usuario utilizado para conectarse a la base de datos. En este caso, es "entich".
- $db_pass: La contraseña del usuario de la base de datos. En este caso, es "entich".
- $db_name: El nombre de la base de datos a la que se conectará la aplicación. En este caso, es `"entich"`.

2. **template.php**

Este archivo contiene funciones para generar la estructura HTML básica de la aplicación. Incluye funciones para imprimir el encabezado, abrir el cuerpo del documento y cerrarlo.

Funciones

1. printHead($title): Esta función imprime el encabezado HTML del documento, incluyendo la etiqueta <title> y un enlace a la hoja de estilos CSS.
    - **Parámetros**:
        - $title: El título de la página que se mostrará en la etiqueta <title>.
2. getLoginOptions(): Esta función devuelve las opciones de navegación dependiendo de si el usuario está logueado o no.
    - **Retorno**:
        - Si el usuario está logueado, devuelve enlaces al dashboard y a la página de logout.
        - Si el usuario no está logueado, devuelve un enlace a la página de login.
3. openBody($title="ENTIch"): Esta función abre el cuerpo del documento HTML, incluyendo el encabezado y la navegación.
    - **Parámetros**:
        - $title: El título de la página que se mostrará en el encabezado. Por defecto, es "ENTIch".
4. closeBody(): Esta función cierra el cuerpo del documento HTML, incluyendo el pie de página.

3. register_check.php

Este archivo se encarga de procesar el formulario de registro de usuarios. Realizo varias validaciones y comprobaciones antes de insertar el nuevo usuario en la base de datos.

Pasos del proceso

1. **Comprobar que todos los campos estén rellenados**:
   - Verifico que los campos register_name, register_username, register_email, register_password y register_repass estén presentes en la solicitud POST. Si falta alguno, muestro un mensaje de error y detengo la ejecución.
2. **Limpiar espacios y validar longitud de los campos**:
   - Elimino los espacios en blanco al principio y al final de los valores de los campos.
   - Verifico que el nombre tenga al menos 3 caracteres, el nombre de usuario al menos 5 caracteres, el correo electrónico al menos 6 caracteres y la contraseña al menos 7 caracteres. Si alguno de estos valores no cumple con los requisitos, muestro un mensaje de error y detengo la ejecución.
3. **Comprobar caracteres problemáticos**:
   - Escapo caracteres especiales y convierto caracteres especiales en entidades HTML para evitar inyecciones de código.
4. **Validar formato de email**:
   - Verifico que el correo electrónico tenga un formato válido utilizando la función filter_var.
5. **Comprobar que el password y repassword coinciden**:
   - Verifico que la contraseña y la confirmación de la contraseña coincidan. Si no coinciden, muestro un mensaje de error y detengo la ejecución.
6. **Generar el hash MD5 del password**:
   - Genero un hash MD5 de la contraseña para almacenarla de forma segura en la base de datos.
7. **Conectar a la base de datos**:
   - Incluyo el archivo db_config.php y establezco una conexión con la base de datos utilizando las credenciales definidas en ese archivo.
8. **Comprobar si el usuario o el email ya existen**:

    - Realizo una consulta a la base de datos para verificar si el nombre de usuario o el correo electrónico ya están registrados. Si ya existen, muestro un mensaje de error y detengo la ejecución.
9. **Insertar nuevo usuario**:
    - Inserto el nuevo usuario en la tabla creators de la base de datos con los valores proporcionados.
10. **Obtener ID del usuario insertado y crear sesión**:
    - Obtengo el ID del usuario recién insertado y creo una sesión para el usuario.
11. **Redireccionar al dashboard**:
    - Redirecciono al usuario a la página del dashboard.

4. profile_update.php

Este archivo se encarga de procesar el formulario de actualización del perfil de usuario. Realizo varias validaciones y comprobaciones antes de actualizar los datos del usuario en la base de datos.

Pasos del proceso

1. **Redirigir si no hay una sesión activa:**
   - Verifico si hay una sesión activa. Si no la hay, redirijo al usuario a la página de login y detengo la ejecución.
2. **Obtener el ID del usuario activo:**
   - Obtengo el ID del usuario activo desde la sesión.
3. **Comprobar que todos los campos obligatorios estén rellenados:**
   - Verifico que los campos name, username, email y description estén presentes en la solicitud POST. Si falta alguno, muestro un mensaje de error y detengo la ejecución.
4. **Limpiar espacios y validar longitud de los campos:**
   - Elimino los espacios en blanco al principio y al final de los valores de los campos.
   - Verifico que el nombre tenga al menos 3 caracteres, el nombre de usuario al menos 5 caracteres y el correo electrónico al menos 6 caracteres. Si alguno de estos valores no cumple con los requisitos, muestro un mensaje de error y detengo la ejecución.
5. **La descripción puede estar vacía:**
   - La descripción es opcional, por lo que puede estar vacía. Si está presente, la limpio y convierto caracteres especiales en entidades HTML.
6. **Comprobar caracteres problemáticos:**
   - Escapo caracteres especiales y convierto caracteres especiales en entidades HTML para evitar inyecciones de código.
7. **Validar formato de email:**
   - Verifico que el correo electrónico tenga un formato válido utilizando la función filter_var.
8. **Conectar a la base de datos:**
   - Incluyo el archivo db_config.php y establezco una conexión con la base de datos utilizando las credenciales definidas en ese archivo.
9. **Actualizar datos del usuario activo:**

- Preparo una consulta SQL para actualizar los datos del usuario activo en la tabla creators de la base de datos con los valores proporcionados.

10. **Redireccionar al dashboard:**
    - Redirecciono al usuario a la página del dashboard.

5. logout.php

Este archivo se encarga de cerrar la sesión del usuario y redirigirlo a la página de inicio.

Pasos del proceso

1. **Iniciar la sesión:**
   - Utilizo session_start() para asegurarme de que la sesión esté iniciada y pueda acceder a las variables de sesión.
2. **Destruir la sesión:**
   - Utilizo session_destroy() para destruir todas las variables de sesión y cerrar la sesión del usuario.
3. **Redirigir a la página de inicio:**
   - Utilizo header("Location: index.php") para redirigir al usuario a la página de inicio (index.php).
   - Utilizo exit() para asegurarme de que el script se detenga después de la redirección.

6. login.php

Este archivo se encarga de mostrar los formularios de inicio de sesión y registro de usuarios. Utilizo funciones para generar la estructura HTML básica y los formularios necesarios.

Pasos del proceso

1. **Incluir el archivo template.php:**
    - Incluyo el archivo template.php que contiene funciones para generar la estructura HTML básica de la aplicación.
2. **Imprimir el encabezado HTML:**
    - Utilizo la función printHead("ENTIch: Login") para imprimir el encabezado HTML del documento, incluyendo la etiqueta <title> y un enlace a la hoja de estilos CSS.
3. **Abrir el cuerpo del documento HTML:**
    - Utilizo la función openBody("ENTIch") para abrir el cuerpo del documento HTML, incluyendo el encabezado y la navegación.
4. **Formulario de inicio de sesión:**
    - Genero un formulario HTML para que los usuarios puedan iniciar sesión. El formulario incluye campos para el nombre de usuario y la contraseña, y se envía al archivo login_check.php para su procesamiento.
5. **Formulario de registro:**
    - Genero un formulario HTML para que los usuarios puedan registrarse. El formulario incluye campos para el nombre, nombre de usuario, correo electrónico, contraseña y confirmación de contraseña, y se envía al archivo register_check.php para su procesamiento.
6. **Cerrar el cuerpo del documento HTML:**
    - Utilizo la función closeBody() para cerrar el cuerpo del documento HTML, incluyendo el pie de página.

7. login_check.php

Este archivo se encarga de procesar el formulario de inicio de sesión de usuarios. Realizo varias validaciones y comprobaciones antes de autenticar al usuario y crear una sesión.

Pasos del proceso

1. **Comprobar que todos los campos estén rellenados**:
    - Verifico que los campos login_username y login_password estén presentes en la solicitud POST. Si falta alguno, muestro un mensaje de error y detengo la ejecución.
2. **Limpiar espacios y validar longitud de los campos**:
    - Elimino los espacios en blanco al principio y al final de los valores de los campos.
    - Verifico que el nombre de usuario tenga al menos 5 caracteres y la contraseña al menos 7 caracteres. Si alguno de estos valores no cumple con los requisitos, muestro un mensaje de error y detengo la ejecución.
3. **Limpiar y encriptar los datos**:
    - Escapo caracteres especiales y convierto caracteres especiales en entidades HTML para evitar inyecciones de código.
    - Genero un hash MD5 de la contraseña para compararla con la almacenada en la base de datos. Nota: MD5 no es seguro para contraseñas críticas, se recomienda usar bcrypt en producción.
4. **Conectar a la base de datos**:
    - Incluyo el archivo db_config.php y establezco una conexión con la base de datos utilizando las credenciales definidas en ese archivo.
5. **Consulta segura para buscar el usuario y contraseña**:
    - Preparo una consulta SQL para buscar el usuario y la contraseña en la tabla creators de la base de datos. Utilizo bind_param para evitar inyecciones SQL.
6. **Verificar resultado**:
    - Ejecuto la consulta y verifico si se encontró un único resultado. Si no se encuentra el usuario o la contraseña es incorrecta, muestro un mensaje de error y detengo la ejecución.
7. **Obtener el ID del usuario**:

    - Obtengo el ID del usuario desde el resultado de la consulta y creo una sesión para el usuario.
8. **Redireccionar al dashboard**:
    - Redirecciono al usuario a la página del dashboard (dashboard.php).
9. **Cerrar conexiones**:
    - Cierro la consulta preparada y la conexión a la base de datos.

8. index.php

Este archivo se encarga de procesar el formulario de inicio de sesión de usuarios. Realizo varias validaciones y comprobaciones antes de autenticar al usuario y crear una sesión.

Pasos del proceso

1. **Comprobar que todos los campos estén rellenados**:
    - Verifico que los campos login_username y login_password estén presentes en la solicitud POST. Si falta alguno, muestro un mensaje de error y detengo la ejecución.
2. **Limpiar espacios y validar longitud de los campos**:
    - Elimino los espacios en blanco al principio y al final de los valores de los campos.
    - Verifico que el nombre de usuario tenga al menos 5 caracteres y la contraseña al menos 7 caracteres. Si alguno de estos valores no cumple con los requisitos, muestro un mensaje de error y detengo la ejecución.
3. **Limpiar y encriptar los datos**:
    - Escapo caracteres especiales y convierto caracteres especiales en entidades HTML para evitar inyecciones de código.
    - Genero un hash MD5 de la contraseña para compararla con la almacenada en la base de datos. Nota: MD5 no es seguro para contraseñas críticas, se recomienda usar bcrypt en producción.
4. **Conectar a la base de datos**:
    - Incluyo el archivo db_config.php y establezco una conexión con la base de datos utilizando las credenciales definidas en ese archivo.
5. **Consulta segura para buscar el usuario y contraseña**:
    - Preparo una consulta SQL para buscar el usuario y la contraseña en la tabla creators de la base de datos. Utilizo bind_param para evitar inyecciones SQL.
6. **Verificar resultado**:
    - Ejecuto la consulta y verifico si se encontró un único resultado. Si no se encuentra el usuario o la contraseña es incorrecta, muestro un mensaje de error y detengo la ejecución.
7. **Obtener el ID del usuario**:

    - Obtengo el ID del usuario desde el resultado de la consulta y creo una sesión para el usuario.
8. **Redireccionar al dashboard**:
    - Redirecciono al usuario a la página del dashboard (dashboard.php).
9. **Cerrar conexiones**:
    - Cierro la consulta preparada y la conexión a la base de datos.

9. game.php

Este archivo se encarga de mostrar la información detallada de un juego específico y sus comentarios. Realizo varias validaciones y consultas a la base de datos para obtener y mostrar la información del juego y los comentarios asociados.

Pasos del proceso

1. **Iniciar la sesión**:
    - Utilizo session_start() para asegurarme de que la sesión esté iniciada y pueda acceder a las variables de sesión.
2. **Incluir el archivo** db_config.php:
    - Incluyo el archivo db_config.php que contiene las credenciales necesarias para conectarse a la base de datos.
3. **Comprobar que se ha especificado un juego**:
    - Verifico que el parámetro id_game esté presente en la URL. Si no está presente, muestro un mensaje de error y detengo la ejecución.
4. **Conectar a la base de datos**:
    - Establezco una conexión con la base de datos utilizando las credenciales definidas en db_config.php.
5. **Obtener la información del juego**:
    - Realizo una consulta a la base de datos para obtener la información del juego especificado por id_game. Si no se encuentra el juego, muestro un mensaje de error y detengo la ejecución.
6. **Incluir el archivo** template.php:
    - Incluyo el archivo template.php que contiene funciones para generar la estructura HTML básica de la aplicación.
7. **Imprimir el encabezado HTML**:
    - Utilizo la función printHead() para imprimir el encabezado HTML del documento, incluyendo la etiqueta <title> con el título del juego.
8. **Abrir el cuerpo del documento HTML**:
    - Utilizo la función openBody() para abrir el cuerpo del documento HTML, incluyendo el encabezado y la navegación.
9. **Mostrar la información del juego**:

    - Muestro el título, la imagen de cabecera, el precio, el enlace a Steam y el tráiler de YouTube del juego.
10. **Mostrar los comentarios del juego**:
    - Realizo una consulta a la base de datos para obtener los comentarios asociados al juego. Si hay comentarios, los muestro en una lista. Si no hay comentarios, muestro un mensaje indicando que no hay comentarios.
11. **Formulario para añadir comentarios**:
    - Si el usuario está logueado, muestro un formulario para que pueda añadir comentarios al juego. El formulario se envía al archivo comment_insert.php para su procesamiento.
12. **Cerrar el cuerpo del documento HTML**:
    - Utilizo la función closeBody() para cerrar el cuerpo del documento HTML, incluyendo el pie de página.
13. **Cerrar la conexión a la base de datos**:
    - Cierro la conexión a la base de datos utilizando mysqli_close().

10. dashboard.php

Este archivo se encarga de mostrar el dashboard del usuario logueado. Realizo varias validaciones y consultas a la base de datos para obtener y mostrar la información del perfil del usuario, permitiendo que este pueda actualizar sus datos.

Pasos del proceso

1. **Iniciar la sesión**:
   - Utilizo session_start() para asegurarme de que la sesión esté iniciada y pueda acceder a las variables de sesión.
2. **Redirigir si no hay una sesión activa**:
   - Verifico si hay una sesión activa. Si no la hay, redirijo al usuario a la página de login y detengo la ejecución.
3. **Obtener el ID del usuario activo**:
   - Obtengo el ID del usuario activo desde la sesión.
4. **Incluir el archivo** template.php:
   - Incluyo el archivo template.php que contiene funciones para generar la estructura HTML básica de la aplicación.
5. **Realizar una consulta para obtener la información del usuario**:
   - Realizo una consulta a la base de datos para obtener la información del usuario activo utilizando el ID del usuario.
6. **Conectar a la base de datos**:
   - Incluyo el archivo db_config.php y establezco una conexión con la base de datos utilizando las credenciales definidas en ese archivo.
7. **Verificar el resultado de la consulta**:
   - Verifico si la consulta se ejecutó correctamente y si se encontró un único resultado. Si no se encuentra el usuario, redirijo al usuario a la página de login y detengo la ejecución.
8. **Obtener la información del usuario**:
   - Utilizo mysqli_fetch_array() para obtener la información del usuario desde el resultado de la consulta.
9. **Imprimir el encabezado HTML**:
   - Utilizo la función printHead() para imprimir el encabezado HTML del documento, incluyendo la etiqueta <title> con el nombre del usuario.
10. **Abrir el cuerpo del documento HTML**:

    - Utilizo la función openBody() para abrir el cuerpo del documento HTML, incluyendo el encabezado y la navegación.
11. **Incluir el archivo** dashboard_template.php:
    - Incluyo el archivo dashboard_template.php que contiene funciones para generar la estructura del dashboard.
12. **Abrir el dashboard**:
    - Utilizo la función openDashboard() para abrir la sección del dashboard.
13. **Mostrar el formulario de actualización del perfil**:
    - Genero un formulario HTML para que el usuario pueda actualizar su perfil. El formulario incluye campos para el nombre, nombre de usuario, correo electrónico y descripción, y se envía al archivo profile_update.php para su procesamiento.
14. **Cerrar el dashboard**:
    - Utilizo la función closeDashboard() para cerrar la sección del dashboard.
15. **Cerrar el cuerpo del documento HTML**:
    - Utilizo la función closeBody() para cerrar el cuerpo del documento HTML, incluyendo el pie de página.

11. dashboard_template.php

Este archivo contiene funciones para generar la estructura del dashboard de la aplicación. Incluye funciones para abrir y cerrar la sección del dashboard.

Funciones

1. openDashboard(): Esta función abre la sección del dashboard, incluyendo un menú de navegación con enlaces a diferentes opciones del dashboard.
    - **Contenido**:
        - Un elemento <aside> que contiene un <nav> con un título "OPCIONES".
        - Una lista <ul> con enlaces a "Perfil", "Juegos" y "Añadir un Juego".
2. closeDashboard(): Esta función cierra la sección del dashboard.
    - **Contenido**:
        - Un elemento <article> que cierra la sección del dashboard.

12. dashboard_games.php

Este archivo se encarga de mostrar el dashboard de juegos del usuario logueado. Realizo varias validaciones y consultas a la base de datos para obtener y mostrar la información de los juegos del usuario, permitiendo que este pueda añadir, modificar y ver comentarios de sus juegos.

Pasos del proceso

1. **Iniciar la sesión**:
    - Utilizo session_start() para asegurarme de que la sesión esté iniciada y pueda acceder a las variables de sesión.
2. **Redirigir si no hay una sesión activa**:
    - Verifico si hay una sesión activa. Si no la hay, redirijo al usuario a la página de login y detengo la ejecución.
3. **Incluir los archivos** template.php **y** db_config.php:
    - Incluyo el archivo template.php que contiene funciones para generar la estructura HTML básica de la aplicación.
    - Incluyo el archivo db_config.php que contiene las credenciales necesarias para conectarse a la base de datos.
4. **Realizar una consulta para obtener la información del usuario**:
    - Realizo una consulta a la base de datos para obtener la información del usuario activo utilizando el ID del usuario.
5. **Conectar a la base de datos**:
    - Establezco una conexión con la base de datos utilizando las credenciales definidas en db_config.php.
6. **Verificar el resultado de la consulta**:
    - Verifico si la consulta se ejecutó correctamente y si se encontró un único resultado. Si no se encuentra el usuario, redirijo al usuario a la página de login y detengo la ejecución.
7. **Obtener la información del usuario**:
    - Utilizo mysqli_fetch_array() para obtener la información del usuario desde el resultado de la consulta.
8. **Imprimir el encabezado HTML**:
    - Utilizo la función printHead() para imprimir el encabezado HTML del documento, incluyendo la etiqueta <title> con el nombre del usuario.
9. **Abrir el cuerpo del documento HTML**:

    - Utilizo la función openBody() para abrir el cuerpo del documento HTML, incluyendo el encabezado y la navegación.
10. **Incluir el archivo** dashboard_template.php:
    - Incluyo el archivo dashboard_template.php que contiene funciones para generar la estructura del dashboard.
11. **Abrir el dashboard**:
    - Utilizo la función openDashboard() para abrir la sección del dashboard.
12. **Mostrar el formulario para añadir o modificar juegos**:
    - Si el usuario ha solicitado añadir un nuevo juego (add_game), muestro un formulario HTML para que el usuario pueda añadir un nuevo juego. El formulario incluye campos para el título, enlace a Steam, imagen, precio y enlace al tráiler de YouTube, y se envía al archivo dashboard_games_add.php para su procesamiento.
    - Si el usuario ha solicitado modificar un juego existente (id_game), realizo una consulta a la base de datos para obtener la información del juego y muestro un formulario HTML con los datos del juego para que el usuario pueda modificarlos. El formulario se envía al archivo dashboard_games_update.php para su procesamiento.
13. **Mostrar los juegos del usuario**:
    - Realizo una consulta a la base de datos para obtener los juegos del usuario. Si hay juegos, los muestro en una lista con su título, imagen, precio, enlace a Steam y tráiler de YouTube. También incluyo un enlace para modificar cada juego.
14. **Mostrar los comentarios de los juegos**:
    - Realizo una consulta a la base de datos para obtener los comentarios de cada juego. Si hay comentarios, los muestro en una lista con el nombre del autor, el comentario y la fecha de creación.
15. **Cerrar el dashboard**:
    - Utilizo la función closeDashboard() para cerrar la sección del dashboard.
16. **Cerrar el cuerpo del documento HTML**:
    - Utilizo la función closeBody() para cerrar el cuerpo del documento HTML, incluyendo el pie de página.

13. dashboard_games_update.php

Este archivo se encarga de procesar el formulario de actualización de juegos en el dashboard del usuario. Realizo varias validaciones y consultas a la base de datos para actualizar la información del juego seleccionado.

Pasos del proceso

1. **Iniciar la sesión**:
    - Utilizo session_start() para asegurarme de que la sesión esté iniciada y pueda acceder a las variables de sesión.
2. **Redirigir si no hay una sesión activa**:
    - Verifico si hay una sesión activa. Si no la hay, redirijo al usuario a la página de login y detengo la ejecución.
3. **Incluir el archivo** db_config.php:
    - Incluyo el archivo db_config.php que contiene las credenciales necesarias para conectarse a la base de datos.
4. **Comprobar que la solicitud es POST y que se ha enviado el formulario de actualización**:
    - Verifico que la solicitud sea de tipo POST y que el formulario de actualización (update_game_submit) haya sido enviado. Si no es así, redirijo al usuario a la página del dashboard de juegos y detengo la ejecución.
5. **Conectar a la base de datos**:
    - Establezco una conexión con la base de datos utilizando las credenciales definidas en db_config.php.
6. **Obtener y limpiar los datos del formulario**:
    - Obtengo los datos del formulario (id_game, add_title, add_link, add_image, add_price, add_trailer) y los limpio utilizando mysqli_real_escape_string para evitar inyecciones SQL. También convierto el precio a un valor flotante utilizando floatval.
7. **Actualizar la información del juego en la base de datos**:
    - Preparo una consulta SQL para actualizar la información del juego en la tabla games de la base de datos con los valores proporcionados.
8. **Ejecutar la consulta y verificar el resultado**:
    - Ejecuto la consulta y verifico si se actualizó correctamente. Si la actualización fue exitosa, redirijo al usuario a la página del

dashboard de juegos. Si hubo un error, muestro un mensaje de error con el detalle del error.

9. **Cerrar la conexión a la base de datos**:
    - Cierro la conexión a la base de datos utilizando mysqli_close().
10. **Redirigir si la solicitud no es válida**:
    - Si la solicitud no es válida, redirijo al usuario a la página del dashboard de juegos y detengo la ejecución.

14. dashboard_games_add.php

Este archivo se encarga de mostrar el formulario para añadir un nuevo juego al dashboard del usuario logueado. Realizo varias validaciones y consultas a la base de datos para obtener y mostrar la información del usuario y sus juegos.

Pasos del proceso

1. **Iniciar la sesión**:
    - Utilizo session_start() para asegurarme de que la sesión esté iniciada y pueda acceder a las variables de sesión.
2. **Redirigir si no hay una sesión activa**:
    - Verifico si hay una sesión activa. Si no la hay, redirijo al usuario a la página de login y detengo la ejecución.
3. **Incluir los archivos** template.php **y** db_config.php:
    - Incluyo el archivo template.php que contiene funciones para generar la estructura HTML básica de la aplicación.
    - Incluyo el archivo db_config.php que contiene las credenciales necesarias para conectarse a la base de datos.
4. **Realizar una consulta para obtener la información del usuario**:
    - Realizo una consulta a la base de datos para obtener la información del usuario activo utilizando el ID del usuario.
5. **Conectar a la base de datos**:
    - Establezco una conexión con la base de datos utilizando las credenciales definidas en db_config.php.
6. **Verificar el resultado de la consulta**:
    - Verifico si la consulta se ejecutó correctamente y si se encontró un único resultado. Si no se encuentra el usuario, redirijo al usuario a la página de login y detengo la ejecución.
7. **Obtener la información del usuario**:
    - Utilizo mysqli_fetch_array() para obtener la información del usuario desde el resultado de la consulta.
8. **Imprimir el encabezado HTML**:
    - Utilizo la función printHead() para imprimir el encabezado HTML del documento, incluyendo la etiqueta <title> con el nombre del usuario.
9. **Abrir el cuerpo del documento HTML**:

    - Utilizo la función openBody() para abrir el cuerpo del documento HTML, incluyendo el encabezado y la navegación.
10. **Incluir el archivo** dashboard_template.php:
    - Incluyo el archivo dashboard_template.php que contiene funciones para generar la estructura del dashboard.
11. **Abrir el dashboard**:
    - Utilizo la función openDashboard() para abrir la sección del dashboard.
12. **Mostrar el formulario para añadir un nuevo juego**:
    - Si el usuario ha solicitado añadir un nuevo juego (add_game), muestro un formulario HTML para que el usuario pueda añadir un nuevo juego. El formulario incluye campos para el título, enlace a Steam, imagen, precio y enlace al tráiler de YouTube, y se envía al archivo dashboard_games_add.php para su procesamiento.
13. **Mostrar el formulario para modificar un juego existente**:
    - Si el usuario ha solicitado modificar un juego existente (id_game), realizo una consulta a la base de datos para obtener la información del juego y muestro un formulario HTML con los datos del juego para que el usuario pueda modificarlos. El formulario se envía al archivo dashboard_games_update.php para su procesamiento.
14. **Mostrar los juegos del usuario**:
    - Realizo una consulta a la base de datos para obtener los juegos del usuario. Si hay juegos, los muestro en una lista con su título, imagen, precio, enlace a Steam y tráiler de YouTube. También incluyo un enlace para modificar cada juego.
15. **Mostrar los comentarios de los juegos**:
    - Realizo una consulta a la base de datos para obtener los comentarios de cada juego. Si hay comentarios, los muestro en una lista con el nombre del autor, el comentario y la fecha de creación.
16. **Cerrar el dashboard**:
    - Utilizo la función closeDashboard() para cerrar la sección del dashboard.
17. **Cerrar el cuerpo del documento HTML**:
    - Utilizo la función closeBody() para cerrar el cuerpo del documento HTML, incluyendo el pie de página.

18. **Función** printHead($title):
    - Esta función imprime el encabezado HTML del documento, incluyendo la etiqueta <title> y un enlace a la hoja de estilos CSS.

15. comment_insert.php

Este archivo se encarga de procesar el formulario de comentarios de los juegos. Realizo varias validaciones y consultas a la base de datos para insertar un nuevo comentario asociado a un juego específico.

Pasos del proceso

1. **Iniciar la sesión**:
    - Utilizo session_start() para asegurarme de que la sesión esté iniciada y pueda acceder a las variables de sesión.
2. **Redirigir si no hay una sesión activa**:
    - Verifico si hay una sesión activa. Si no la hay, redirijo al usuario a la página de login y detengo la ejecución.
3. **Incluir el archivo** db_config.php:
    - Incluyo el archivo db_config.php que contiene las credenciales necesarias para conectarse a la base de datos.
4. **Comprobar que la solicitud es POST y que se han enviado los campos necesarios**:
    - Verifico que la solicitud sea de tipo POST y que los campos comment e id_game estén presentes en la solicitud. Si no es así, redirijo al usuario a la página de inicio y detengo la ejecución.
5. **Obtener y limpiar los datos del formulario**:
    - Obtengo los datos del formulario (id_game y comment) y los limpio eliminando los espacios en blanco al principio y al final de los valores.
6. **Validar el comentario**:
    - Verifico que el comentario no esté vacío. Si está vacío, muestro un mensaje de error y detengo la ejecución.
7. **Conectar a la base de datos**:
    - Establezco una conexión con la base de datos utilizando las credenciales definidas en db_config.php.
8. **Escapar caracteres especiales en el comentario**:
    - Escapo caracteres especiales en el comentario utilizando htmlspecialchars para evitar inyecciones de código.
9. **Insertar el comentario en la base de datos**:
    - Preparo una consulta SQL para insertar el comentario en la tabla comments de la base de datos con los valores proporcionados.

10. **Ejecutar la consulta y verificar el resultado**:
    - Ejecuto la consulta y verifico si se insertó correctamente. Si la inserción fue exitosa, redirijo al usuario a la página del juego. Si hubo un error, muestro un mensaje de error.
11. **Cerrar la conexión a la base de datos**:
    - Cierro la conexión a la base de datos utilizando mysqli_close().
12. **Redirigir si la solicitud no es válida**:
    - Si la solicitud no es válida, redirijo al usuario a la página de inicio y detengo la ejecución.