DOCUMENTACIÓN TRABAJO HLC, RELACIONAR SITIO WEB CON UNA BASE DE DATOS MEDIANTE PHP.

Alumno: Alejandro Ojeda Morillo

2º ASIR I.E.S LUIS VÉLEZ DE GUEVARA

Curso 2024/2025

Sumario

Introducción.

En mi caso, he decidido crear mi sitio web sobre el videojuego *elden ring* ya que es un juego que se caracteriza por tener una gran cantidad de objetos, personajes y demás.

Elegí este videojuego justamente porque al tener tantos datos, se podrían crear varias tablas a manejar con muchos datos, a pesar de que yo solo use 3 (sin contar la de usuarios), este juego da a pie para muchos más.

Base de datos.

La base de datos que voy a usar la he denominado *eldenring_db* y ella albergara las siguientes 4 tablas:

- Armas: Hace referencia a las armas del juego, con las siguientes columnas.

- Id (autoincremental), nombre, tipo (que es ligera, media o pesada), dano_base, y escalado.

- Armadura: Hace referencia a las armaduras del juego, con las columnas:

- Id (autoincremental), nombre, tipo,defensa_fisica y peso.

- Jefes: Hace referencia a algunos enemigos importantes del juego, sus columnas son:

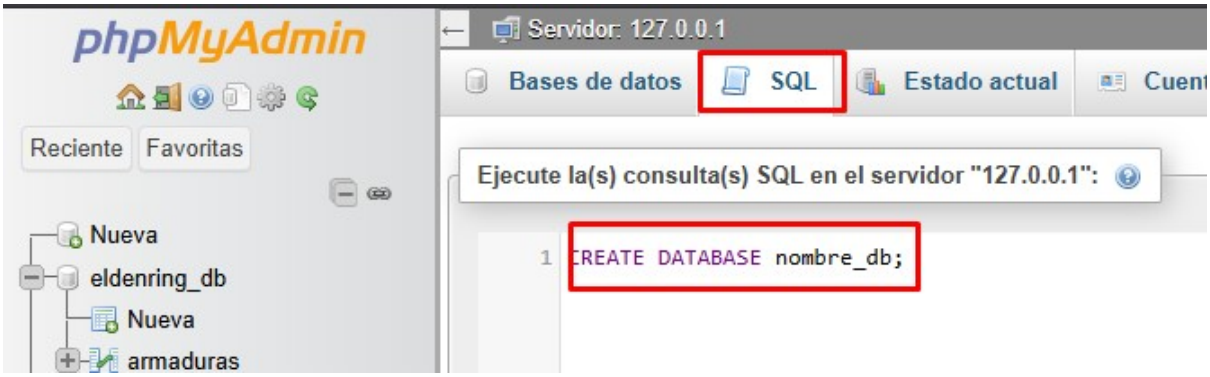
- Id (autoincremental), nombre, ubicación y requisito_historia (que debe ser si o no)

- Usuarios: Hace referencia a los usuarios registrados en la página:

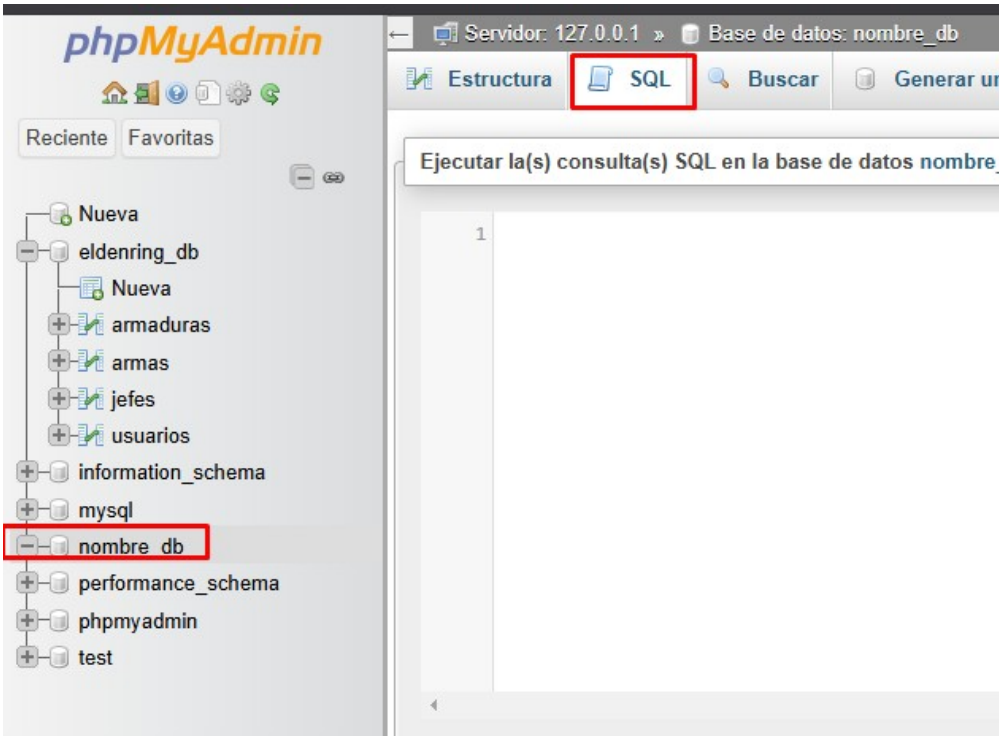
- Id (autoincremental), nombre_usuario, email, contraseña y rol (usuario o admin)

Para crear dicha base de datos, la mejor forma es yendo directamente al administrador de PHPMyAdmin.

Vamos a SQL y usamos esta sentencia para crear la base de datos.



Luego, damos clic a la base de datos creada y damos otra vez clic en SQL y usamos la setencia SQL para crear las tablas.



Que en mi caso, esta seria mi sentencia.

```sql
CREATE TABLE armaduras (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    tipo ENUM('Ligera', 'Media', 'Pesada') NOT NULL,
    defensa_fisica DECIMAL(5,2) NOT NULL,
    peso DECIMAL(5,2) NOT NULL
);
CREATE TABLE armas (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    tipo ENUM('Espada', 'Lanza', 'Hacha', 'Daga', 'Arco', 'Catalizador') NOT NULL,
    daño_base DECIMAL(5,2) NOT NULL,
    escalado VARCHAR(10) NOT NULL
);
CREATE TABLE jefes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    ubicacion VARCHAR(150) NOT NULL,
    requisito_historia ENUM('Sí', 'No') NOT NULL
);
CREATE TABLE usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre_usuario VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE,
    contraseña VARCHAR(255) NOT NULL,
    rol ENUM('usuario', 'admin') NOT NULL DEFAULT 'usuario');
```

Explicación web.

Ahora, voy a pasar a explicar las partes más importantes del código de cada archivo.

No enseñaré todo el código, solo lo que sea PHP.

index.php

Esta es la primera página que se encuentra nada más entrar al sitio, yo me decante que la primera sea simple, con un texto dando la bienvenida y dos botones, uno para iniciar sesión en el caso de que dispongas de una cuenta y otro botón para ir a registrarte.

```php
<?php
session_start();
if (isset($_SESSION["nombre_usuario"])) {
    header("Location: home.php");
    exit();
}
?>
```

En esta página, lo único de código en PHP es esto, que verifica si la sesión está abierta, ya que si accedes al index.php y iniciaste sesión con anterioridad, te redirige directamente al home.php, que es la página donde esta todo el contenido y enlaces a las tablas.

Conexion.php

Este archivo es esencial, ya que crea una conexión entre la base de datos que hemos creado y almacenamos los datos, con los archivos donde necesitamos acceder a esos datos que hemos almacenado.

```php
<?php
// Variables de conexión
$servidor = "localhost:3306"; // Nombre del servidor (por defecto localhost)
$usuario = "root";       // Usuario (por defecto root en XAMPP)
$contrasena = "";        // Contraseña (por defecto vacío en XAMPP)
$nombre_base_datos = "eldenring_db"; // Nombre de la base de datos

// Establecer la conexión
$conexion = mysqli_connect($servidor, $usuario, $contrasena, $nombre_base_datos);

// Verificar si la conexión fue exitosa
if (!$conexion) {
    die("Error de conexión al servidor MySQL: " . mysqli_connect_error());
}

// Verificar si la base de datos se ha seleccionado correctamente
if (!mysqli_select_db($conexion, $nombre_base_datos)) {
    die("Error al seleccionar la base de datos: " . mysqli_error($conexion));
}

// aSi llegamos aquí, la conexión y la selección de base de datos fueron exitosas
//echo "Conexión exitosa a la base de datos '$nombre_base_datos' en el servidor '$servidor'.";

// Cerrar la conexión cuando ya no sea necesaria
// mysqli_close($conexion);
?>
```

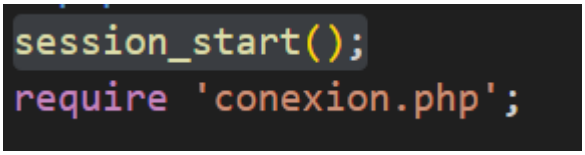
Almacenamos todos los datos necesarios en variables para luego poder usarlos en el mysqli_connect, que es una función que se usa para conectarse a la base de datos que indicamos posteriormente con las variables que hemos creado y le hemos indicado dentro los paréntesis de la propia función.

login.php y auth.php

Aquí he decidido explicar ambos archivos, ya que están estrechamente relacionados uno con el otro.

El archivo auth.php sea usa principalmente para que verifique si está la sesión iniciada, y si no es así, nos lleva automáticamente hacia el login.php, por ejemplo el uso sería que si queremos acceder directamente a home.php a través de la URL, nos lleve al login.php, ya que es obligatorio tener una cuenta e iniciar sesión para acceder al resto del sitio.

Luego en el login.php tenemos el siguiente código de php.



Para empezar, tenemos el session_start() y el require ‘conexion.php’; el primero se usa para almacenar datos del usuario durante su navegación. Luego, el require es totalmente necesario para indicar que nos conectaremos a la base de datos que se definió en el archivo conexion.php.

Registro.php

Este es el código PHP del registro, vemos que lo primero que hace es comprobar que todo se envia mediante el metodo post, y también comprueba que el usuario y el email a la hora de registrar también lo reciba por ese mismo metodo. Por último, al llegar a la contraseña, vemos que la hashea para hacerla más segura.

```php
<?php
require 'conexion.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $nombre_usuario = $_POST['nombre_usuario'];
    $email = $_POST['email'];
    $contraseña = password_hash($_POST['contraseña'], PASSWORD_DEFAULT); // Hasheamos la contraseña
```

Luego, vemos que comprueba si el usuario existe, si es así, en la variable $resultado vemos que almacena un dato booleano, que si devuelve false (en este caso 0) significa que ese usuario ya existe. Si el usuario no existe, realiza la consulta para insertar el nuevo usuario en la tabla de usuarios.

```php
    // Verificar si el usuario o email ya existen
    $query = "SELECT id FROM usuarios WHERE nombre_usuario = '$nombre_usuario' OR email = '$email'";
    $resultado = mysqli_query($conexion, $query);

    if (mysqli_num_rows($resultado) > 0) {
        $error = "El nombre de usuario o el email ya están registrados.";
    } else {
        // Insertar nuevo usuario
        $query_insert = "INSERT INTO usuarios (nombre_usuario, email, contraseña) VALUES ('$nombre_usuario', '$email', '$contraseña')";
        if (mysqli_query($conexion, $query_insert)) {
            header("Location: login.php");
            exit();
        } else {
            $error = "Hubo un error al registrar el usuario.";
        }
    }
}
?>
```

Home.php

Como partes más importante del codigo PHP en el home tenemos lo siguiente.

```php
<?php
session_start();

// Si el usuario no ha iniciado sesión, redirigir al login
if (!isset($_SESSION["nombre_usuario"])) {
    header("Location: login.php");
    exit();
}

// Establecer cookies con los datos de sesión si no existen
if (!isset($_COOKIE["nombre_usuario"])) {
    setcookie("nombre_usuario", $_SESSION["nombre_usuario"], time() + 3600, "/"); // 1 hora
}
if (!isset($_COOKIE["rol"])) {
    setcookie("rol", $_SESSION["rol"], time() + 3600, "/");
}
?>
```

El primer if hace que si intentamos entrar al home sin iniciar sesión (mediante la URL) nos lleve directamente al login.php para que iniciemos sesion.

Luego esa parte del código nos crea las cookies, que este caso duran 1h.

Por otro lado, también tenemos este codigo.

```php
<?php if ($_SESSION["rol"] == "admin"): ?>
    <li class="nav-item"><a class="nav-link" href="admin.php">Panel Admin</a></li>
<?php endif; ?>
```

Se encuentra la barra superior, y se usa para que si iniciamos sesión con alguna cuenta con el rom de admin, nos salga un botón extra para ir al panel de administrador.

Admin.txt

Aquí es simple. Solo verifica que al entrar desde la URL, eres admin, si no, te lleva de vuelta al login.

```php
<?php
session_start();
if (!isset($_SESSION["nombre_usuario"]) || $_SESSION["rol"] !== "admin") {
    header("Location: login.php");
    exit();
}

require_once "conexion.php";
?>
```

crud_armaduras, crud_armas, crud_jefes y crud_usuarios.php

Crud, que viene de las siglas create, read, update y delete, son los archivos que vamos a usar para hacer las modificaciones en la base de datos sin necesidad de tener que hacerlo mediante código SQL.

Los 4 son iguales, solo se diferencia por cual base de datos hace referencia.

Esto nos consigue todos los datos de la tabla de armaduras, para luego mediante HTML, poder insertar esos datos en una tabla.

```php
// Obtener lista de armaduras
$resultado = mysqli_query($conexion, "SELECT * FROM armaduras");
$armaduras = mysqli_fetch_all($resultado, MYSQLI_ASSOC);
```

Esto hace que elimine una de las filas al hacer clic en el boton de eliminar.

```php
// Manejo de eliminación de armadura
if (isset($_POST['accion']) && $_POST['accion'] == 'eliminar') {
    $id = $_POST['id'];
    $query = "DELETE FROM armaduras WHERE id = $id";
    mysqli_query($conexion, $query);
}
```

Esto hace que al darle a actualizar y rellenar con nueva información, se actualice la fila que hemos decidido modificar.

```php
// Manejo de actualización de armadura
if (isset($_POST['accion']) && $_POST['accion'] == 'actualizar') {
    $id = $_POST['id'];
    $nombre = $_POST['nombre_armadura'];
    $tipo = $_POST['tipo'];
    $defensa_fisica = $_POST['defensa_fisica'];
    $peso = $_POST['peso'];

    $query = "UPDATE armaduras SET nombre = '$nombre', tipo = '$tipo', defensa_fisica = $defensa_fisica, peso = $peso WHERE id = $id";
    mysqli_query($conexion, $query);
}
```

Esto lo que hace es conseguir los datos desde la base de datos de la fila que queremos actualizar.

```php
// Obtener armadura a actualizar
$armadura_para_actualizar = null;
if (isset($_GET['editar'])) {
    $id = $_GET['editar'];
    $query = "SELECT * FROM armaduras WHERE id = $id";
    $resultado = mysqli_query($conexion, $query);
    $armadura_para_actualizar = mysqli_fetch_assoc($resultado);
}
?>
```

Leerjefes.php, leerarmas.php y leerarmaduras.php

Estos 3 archivos tienen practicamente tienen la misma estrucutura, pero variando para según a que tabla se refieran.
Yo por ejemplo lo voy a explicar con el de jefes.

```php
// Obtener valores de los filtros
$filtro_nombre = isset($_GET['nombre']) ? $_GET['nombre'] : '';
$filtro_ubicacion = isset($_GET['ubicacion']) ? $_GET['ubicacion'] : '';
$filtro_historia = isset($_GET['requisito_historia']) ? $_GET['requisito_historia'] : '';
```

Los parametros que se elijan al filtrar se envian mediante la URL, por eso se usa el metodo GET, para conseguir esos parametros desde ahí.

```php
// Construcción de la consulta con filtros
$query = "SELECT id, nombre, ubicacion, requisito_historia FROM jefes WHERE 1=1";

if (!empty($filtro_nombre)) {
    $query .= " AND nombre LIKE '%" . mysqli_real_escape_string($conexion, $filtro_nombre) . "%'";
}
if (!empty($filtro_ubicacion)) {
    $query .= " AND ubicacion LIKE '%" . mysqli_real_escape_string($conexion, $filtro_ubicacion) . "%'";
}
if (!empty($filtro_historia)) {
    $query .= " AND requisito_historia = '" . mysqli_real_escape_string($conexion, $filtro_historia) . "'";
}
```

En esta parte del código se aplica el filtrado.

```php
// Ejecutar la consulta
$resultado = mysqli_query($conexion, $query);
```

Esto te devuelve la tabla ya con los filtros aplicados.

```php
// Verificar si la consulta fue exitosa
if (!$resultado) {
    die('Error en la consulta: ' . mysqli_error($conexion));
}
?>
```

Los 3 archivos son prácticamente iguales, así que no creo que sea necesario explicar los 3 ya que son idénticos, solo cambia a que tabla se refiere.

Noticia1.php y noticia2.php

Estos dos archivos solo tienen este codigo, que sirve para verificar que has iniciado sesión correctamente.

```php
<?php
session_start();
if (!isset($_SESSION["nombre_usuario"])) {
    header("Location: login.php");
    exit();
}
?>
```

logout.php

El archivo de logout, que nos permite cerrar sesión, es bastante corto, ya que lo que hace es destruir las cookies y las variables de sesión y llevarnos de vuelva al index.php

logout.php

```php
<?php
session_start();

// Destruir las variables de sesión
session_destroy();

// Eliminar cookies si existen
if (isset($_COOKIE['nombre_usuario'])) {
    setcookie('nombre_usuario', '', time() - 3600, "/");
}
if (isset($_COOKIE['rol'])) {
    setcookie('rol', '', time() - 3600, "/");
}

// Redirigir al home
header("Location: index.php");
exit;
?>
```