**Pol\_fork**

**Web**

**includes**

**Explicación footer:**

1. **<footer>**: Define el pie de página del documento.
2. **<div class="f-links">**: Un contenedor div para los enlaces a redes sociales.
   * **<a href="#" target="\_blank">**: Define un enlace. El atributo href="#" indica que no tiene una URL específica (esto debería ser reemplazado por la URL real). target="\_blank" hace que el enlace se abra en una nueva pestaña.
   * **<i class="fab fa-facebook-f"></i>**: Utiliza un icono de la librería de iconos FontAwesome. En este caso, es el icono de Facebook. Los demás iconos funcionan de manera similar: Instagram, LinkedIn y GitHub.
3. **<div class="copy">**: Un contenedor div para la información de copyright y créditos.
   * **<p>Desarrollado por: Alba, Marc, Pol</p>**: Un párrafo que indica los desarrolladores.
   * **<p>Todos los derechos reservados</p>**: Un párrafo con una declaración de derechos.
   * **<p>Copyright &copy; 2024</p>**: Un párrafo que muestra el aviso de copyright con el símbolo © seguido del año 2024.

**Explicación header:**

1. **<header>**: Define el encabezado del documento.
2. **<h1>Chess Game</h1>**: Un encabezado de nivel 1 con el título "Chess Game".
3. **<nav id="menu">**: Un contenedor nav con el id menu que agrupa enlaces de navegación.
   * **<a href="playS.php"><button id="classic">Classic</button></a>**: Un enlace que contiene un botón. El href="playS.php" indica que al hacer clic en el botón se redirigirá a la página playS.php. El botón tiene el id classic y el texto "Classic".
   * **<a href="blitz.php"><button id="blitz">Blitz</button></a>**: Similar al anterior, pero redirige a blitz.php y el botón tiene el id blitz con el texto "Blitz".
   * **<a href="logout.php"><button id="logout">Tanca sesió</button></a>**: Otro enlace con un botón, redirige a logout.php y el botón tiene el id logout con el texto "Tanca sesió" (que significa "Cerrar sesión" en catalán).

**Pol fork**

**Web**

**Js:blitz.js**

### General Overview

Este script es parte de un juego de ajedrez en HTML y JavaScript. Implementa la creación del tablero de ajedrez, la colocación inicial de las piezas, el movimiento de las piezas (incluyendo las reglas básicas de movimiento), la detección de jaque, un historial de movimientos y un temporizador de juego para ambos jugadores.

1. Variables y Objetos Iniciales

 Se seleccionan los elementos HTML necesarios (chessboard, newGame, historialLeft, historialRight).

 Se definen los caracteres para representar las piezas de ajedrez.

 Se crean arrays para las letras y números del tablero.

2. Creación del Tablero

 createBoard crea el tablero de ajedrez de 8x8.

 Se crean y se agregan las filas y las casillas, asignando clases y data-attributes para su identificación.

 Se añaden letras a la primera fila y números a la primera columna.

 Se añaden listeners para los eventos de arrastrar y soltar.

Colocación de las Piezas

 placePieces coloca las piezas en sus posiciones iniciales.

 Se definen las filas iniciales de piezas para ambos jugadores.

 Se crean elementos div para las piezas, se añaden las clases y data-attributes correspondientes, y se agregan listeners para el arrastre de piezas.

Inicio de un Nuevo Juego

 El botón de "Nuevo Juego" reinicia el tablero y el historial de movimientos.

 Se vuelven a crear el tablero y las piezas, y se inicializan las variables de turno y el historial.

. Selección y Movimiento de Piezas

 Se gestionan los eventos de selección de piezas, arrastre y soltar.

 Se valida que las piezas solo se seleccionen si es su turno y se deseleccionan después de moverlas.

Movimiento de Piezas y Validación de Movimientos

 movePiece mueve la pieza a la casilla objetivo si el movimiento es válido y no pone al rey en jaque.

 isValidMove es la lógica que valida si un movimiento es permitido para cada tipo de pieza.

 isKingInCheck verifica si el rey está en jaque.

Historial de Movimientos

* historialMove registra cada movimiento en el historial correspondiente.
* rotatePieces rota las piezas según el turno.
* updateTurnIndicator actualiza el indicador de turno.
* playTimer gestiona el temporizador de cada jugador.

### Resumen

Este script gestiona un juego de ajedrez básico en una página web. Maneja la creación del tablero, colocación de piezas, validación y ejecución de movimientos, detección de jaque, y mantiene un historial de movimientos y un temporizador para ambos jugadores.

**Pol fork**

**Web**

**Js:matrix**

Configuración del Lienzo (canvas)

 Se obtiene una referencia al elemento <canvas> en el HTML usando getElementById.

 Se obtiene el contexto 2D del canvas (ctx), que se utilizará para dibujar.

. Ajuste de Tamaño del Lienzo

 El ancho (w) del canvas se establece al ancho del cuerpo del documento.

 La altura (h) del canvas se establece a la altura del cuerpo del documento.

1. Inicialización de Columnas y Posiciones Y

 cols calcula el número de columnas necesarias dividiendo el ancho del canvas por 10 y sumando 1.

1.  ypos es un array que contiene la posición Y inicial (0) para cada columna.
2. . Fondo Inicial

. Fondo Inicial

 Se establece el color de relleno a negro (#000).

 Se dibuja un rectángulo negro que cubre todo el canvas.

1. Función matrix
2. **Fondo semitransparente:**
   * Se establece el color de relleno a #fff1 (blanco con opacidad del 1%).
   * Se dibuja un rectángulo semitransparente que cubre todo el canvas, creando un efecto de desvanecimiento.
3. **Texto:**
   * Se establece el color de relleno a gris (#777).
   * Se establece la fuente a 37pt monospace.
4. **Dibujo del texto:**
   * Para cada posición y en ypos, se selecciona un carácter aleatorio del string "♔♕♖♗♘♙".
   * text2 es el carácter seleccionado aleatoriamente.
   * x se calcula multiplicando el índice (ind) por 90.
   * Se dibuja el carácter (text2) en la posición (x, y).
5. **Actualización de ypos:**
   * Si y es mayor que 10 + Math.random() \* 300000, la posición y de la columna se reinicia a 0.
   * De lo contrario, y se incrementa en 100.

### 6. Configuración del Intervalo

Se llama a la función matrix cada 70 milisegundos, creando una animación continua.

**Pol fork**

**Web**

**Js:options**

### Función on

* Esta función selecciona el elemento con el ID overlay y cambia su estilo de visualización (display) a grid.
* Esto hace que el elemento sea visible y lo configura para usar el diseño de cuadrícula CSS.

### Función off

* Esta función selecciona el mismo elemento (overlay) y cambia su estilo de visualización a none.
* Esto oculta el elemento, eliminándolo del flujo de diseño del documento.

### Llamada Inicial a off()

* Esta llamada inicial a off() asegura que el elemento overlay esté oculto cuando se carga la página.
* **Pol fork**
* **Web**
* **Js:script**

Espera a que el DOM se cargue

document.addEventListener("DOMContentLoaded", () => {

Esta línea asegura que el código se ejecutará solo después de que todo el contenido HTML se haya cargado.

### 2. Declaración de variables y configuración inicial

javascript

Copiar código

const chessboard = document.getElementById("chessboard");

const newGame = document.getElementById("new-game");

const historialLeft = document.getElementById('left-historial');

const historialRight = document.getElementById('right-historial');

const pieces = {

Rey: "♔",

Reina: "♕",

Torre: "♖",

Alfil: "♗",

Caballo: "♘",

Peón: "♙",

};

letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'];

numbers = [1, 2, 3, 4, 5, 6, 7, 8];

Estas líneas inicializan las referencias a elementos HTML y crean un objeto para representar las piezas del ajedrez.

### 3. Creación del tablero

javascript

Copiar código

function createBoard() {

for (let i = 0; i < 8; i++) {

const row = document.createElement("div");

row.classList.add(i % 2 === 0 ? 'row-even' : 'row-odd');

for (let j = 0; j < 8; j++) {

const square = document.createElement("div");

square.classList.add("square");

square.dataset.x = j;

square.dataset.y = i;

square.dataset.l = letters[j];

square.dataset.n = numbers[i];

square.addEventListener('dragover', allowDrop);

square.addEventListener('drop', drop);

if (i === 0) {

const letter = document.createElement("div");

letter.textContent = letters[j];

letter.classList.add("letter");

square.appendChild(letter);

}

if (j === 0) {

const number = document.createElement("div");

number.textContent = numbers[i];

number.classList.add("number");

square.appendChild(number);

}

row.appendChild(square);

}

chessboard.appendChild(row);

}

}

Esta función crea el tablero de ajedrez de 8x8 y asigna los atributos de coordenadas a cada casilla. También agrega los eventos de arrastrar y soltar.

### 4. Colocación inicial de las piezas

javascript

Copiar código

function placePieces() {

const initialSetup = [

["Torre", "Caballo", "Alfil", "Reina", "Rey", "Alfil", "Caballo", "Torre"],

["Peón", "Peón", "Peón", "Peón", "Peón", "Peón", "Peón", "Peón"],

[],

[],

[],

[],

["Peón", "Peón", "Peón", "Peón", "Peón", "Peón", "Peón", "Peón"],

["Torre", "Caballo", "Alfil", "Reina", "Rey", "Alfil", "Caballo", "Torre"],

];

for (let i = 0; i < initialSetup.length; i++) {

for (let j = 0; j < initialSetup[i].length; j++) {

if (initialSetup[i][j]) {

const piece = document.createElement("div");

piece.textContent = pieces[initialSetup[i][j]];

piece.classList.add("piece");

piece.id = 'rotated';

piece.dataset.type = initialSetup[i][j];

piece.dataset.color = i < 2 ? "white" : "black";

piece.setAttribute('draggable', true);

piece.addEventListener('dragstart', dragStart);

piece.addEventListener('dragend', dragEnd);

chessboard.children[i].children[j].appendChild(piece);

}

}

}

}

Esta función coloca las piezas de ajedrez en sus posiciones iniciales.

### 5. Creación y reinicio del juego

javascript

Copiar código

createBoard();

placePieces();

newGame.addEventListener("click", () => {

chessboard.innerHTML = "";

historialLeft.textContent = "";

historialRight.textContent = "";

createBoard();

placePieces();

whiteTurn = true;

updateTurnIndicator();

num = 1;

chessboard.style.transform = "rotate(-90deg)";

});

Aquí se crean el tablero y las piezas cuando se carga la página y se reinicia el juego al hacer clic en el botón "new-game".

### 6. Variables y funciones auxiliares para el juego

javascript

Copiar código

var num = 1;

let selectedPiece = null;

let whiteTurn = true;

const turnIndicator = document.getElementById('turn-indicator');

function isPieceTurn(piece) {

const pieceColor = piece.dataset.color;

return (whiteTurn && pieceColor === 'white') || (!whiteTurn && pieceColor === 'black');

}

function selectPiece(event) {

const target = event.target;

if (target.classList.contains("piece") && isPieceTurn(target)) {

if (selectedPiece) {

selectedPiece.classList.remove("selected");

}

selectedPiece = target;

selectedPiece.classList.add("selected");

} else if (selectedPiece && target.classList.contains("square")) {

movePiece(selectedPiece, target);

} else if (selectedPiece && target.classList.contains("piece") && !isPieceTurn(target)) {

const targetSquare = target.parentElement;

if(isValidMove(selectedPiece.dataset.type, selectedPiece.dataset.color, parseInt(selectedPiece.parentElement.dataset.x, 10), parseInt(selectedPiece.parentElement.dataset.y, 10), parseInt(targetSquare.dataset.x, 10), parseInt(targetSquare.dataset.y, 10))){

movePiece(selectedPiece, targetSquare);

target.remove();

if (!isKingInCheck);

movePiece(selectedPiece, targetSquare);

} else {

selectedPiece.classList.remove("selected");

}

}

}

Estas funciones manejan la lógica de selección y movimiento de las piezas, asegurándose de que se sigan las reglas del ajedrez.

### 7. Funciones de arrastrar y soltar

javascript

Copiar código

function allowDrop(event) {

event.preventDefault();

}

function dragStart(event) {

if (isPieceTurn(event.target)) {

selectedPiece = event.target;

event.dataTransfer.setData("text/plain", event.target.dataset.type);

} else {

event.preventDefault();

}

}

function drop(event) {

event.preventDefault();

const targetSquare = event.target;

if (selectedPiece && targetSquare.classList.contains('square')) {

movePiece(selectedPiece, targetSquare);

} else if (selectedPiece && targetSquare.classList.contains("piece") && !isPieceTurn(targetSquare)) {

const targetSquare = event.target.parentElement;

if(isValidMove(selectedPiece.dataset.type, selectedPiece.dataset.color, parseInt(selectedPiece.parentElement.dataset.x, 10), parseInt(selectedPiece.parentElement.dataset.y, 10), parseInt(targetSquare.dataset.x, 10), parseInt(targetSquare.dataset.y, 10))){

movePiece(selectedPiece, targetSquare);

event.target.remove();

movePiece(selectedPiece, targetSquare);

} else {

selectedPiece.classList.remove("selected");

}

}

}

function dragEnd(event) {

event.target.classList.remove("selected");

selectedPiece = null;

}

Estas funciones permiten arrastrar y soltar piezas en el tablero.

### 8. Registro del movimiento en el historial

javascript

Copiar código

function historialMove(selectedPiece, targetSquare) {

const piece = selectedPiece.dataset.type;

const toL = targetSquare.dataset.l;

const toN = targetSquare.dataset.n;

if (whiteTurn) {

const historialText = num + '. ' + piece.toUpperCase().charAt(0) + toL + toN;

const historialUndo = historialLeft.textContent.split('\n');

historialUndo.push(historialText);

historialLeft.textContent = historialUndo.join('\n');

} else {

const historialText = num + '. ' + piece.toUpperCase().charAt(0) + toL + toN;

const historialUndo = historialRight.textContent.split('\n');

historialUndo.push(historialText);

historialRight.textContent = historialUndo.join('\n');

}

num++;

}

Esta función registra los movimientos en el historial de la partida.

### 9. Mover piezas y validar movimientos

javascript

Copiar código

function movePiece(piece, targetSquare) {

const fromX = parseInt(piece.parentElement.dataset.x, 10);

const fromY = parseInt(piece.parentElement.dataset.y, 10);

const type = piece.dataset.type;

const color = piece.dataset.color;

const toX = parseInt(targetSquare.dataset.x, 10);

const toY = parseInt(targetSquare.dataset.y, 10);

if (isValidMove(type, color, fromX, fromY, toX, toY)) {

const originalParent = piece.parentElement;

targetSquare.appendChild(piece);

selectedPiece.classList.remove("selected");

selectedPiece = null;

whiteTurn = !whiteTurn;

updateTurnIndicator();

historialMove(piece, targetSquare);

}

}

Esta función mueve las piezas en el tablero y valida los movimientos según las reglas de ajedrez.

### 10. Indicador de turno

javascript

Copiar código

function updateTurnIndicator() {

if (whiteTurn) {

turnIndicator.textContent = "Turno de las Blancas";

} else {

turnIndicator.textContent = "Turno de las Negras";

}

}

Actualiza el indicador de turno para mostrar de quién es el turno.

### 11. Validación de movimientos

javascript

Copiar código

function isValidMove(type, color, fromX, fromY, toX, toY) {

switch (type) {

case "Rey":

return Math.abs(fromX - toX) <= 1 && Math.abs(fromY - toY) <= 1;

case "Reina":

return (fromX === toX || fromY === toY || Math.abs(fromX - toX) === Math.abs(fromY - toY));

case "Torre":

return fromX === toX || fromY === toY;

case "Alfil":

return Math.abs(fromX - toX) === Math.abs(fromY - toY);

case "Caballo":

return (Math.abs(fromX - toX) === 2 && Math.abs(fromY - toY) === 1) || (Math.abs(fromX - toX) === 1 && Math.abs(fromY - toY) === 2);

case "Peón":

if (color === "white") {

return fromY - toY === 1 && fromX === toX;

} else {

return toY - fromY === 1 && fromX === toX;

}

}

}

Esta función valida los movimientos de las piezas según las reglas estándar del ajedrez.

### Conclusión

Este código crea un tablero de ajedrez interactivo que permite mover piezas y reiniciar el juego. Se aseguran de que se sigan las reglas del ajedrez y que los movimientos se registren en un historial.

**Pol fork**

**Webç**

**Js:validation**

### 1. Envolviendo el código en una IIFE (Immediately Invoked Function Expression)

javascript

Copiar código

(() => {

'use strict'

Se utiliza una IIFE para evitar contaminar el ámbito global. use strict activa el modo estricto de JavaScript, que ayuda a detectar errores comunes.

### 2. Selección de todos los formularios con la clase needs-validation

javascript

Copiar código

const forms = document.querySelectorAll('.needs-validation')

Esta línea selecciona todos los formularios que tienen la clase needs-validation. Esta clase se utiliza para identificar los formularios a los que se aplicará la validación personalizada.

### 3. Iteración sobre cada formulario y adición de un evento de escucha para el evento submit

javascript

Copiar código

Array.from(forms).forEach(form => {

form.addEventListener('submit', event => {

if (!form.checkValidity()) {

event.preventDefault()

event.stopPropagation()

}

form.classList.add('was-validated')

}, false)

})

})()

* Array.from(forms).forEach(form => { ... }): Convierte la lista de formularios en un array y luego itera sobre cada uno.
* form.addEventListener('submit', event => { ... }, false): Añade un evento de escucha para el evento submit de cada formulario.
  + if (!form.checkValidity()) { ... }: Comprueba si el formulario es válido usando la función checkValidity(). Esta función es nativa de los formularios en HTML5 y verifica si todos los campos requeridos son válidos.
    - event.preventDefault(): Previene el envío del formulario si no es válido.
    - event.stopPropagation(): Evita que el evento se propague más allá del formulario.
  + form.classList.add('was-validated'): Añade la clase was-validated al formulario. Esta clase es utilizada por Bootstrap para aplicar estilos a los campos según su estado de validación (válido o inválido).

### Resumen

Este script asegura que los formularios con la clase needs-validation no se envíen si hay campos inválidos. Utiliza la validación nativa de HTML5 y aplica estilos de Bootstrap para proporcionar una retroalimentación visual al usuario. Al añadir la clase was-validated, Bootstrap puede mostrar mensajes de error o éxito apropiados para cada campo del formulario.

Este enfoque es muy útil para mejorar la experiencia del usuario, asegurando que todos los campos obligatorios se llenen correctamente antes de permitir el envío del formulario.

**Pol fork**

**Web**

**Php**

**Db\_conn**

1. <?php: Esto indica que el código PHP comienza aquí.
2. Comentarios: Los comentarios en PHP se escriben con // para una línea o /\* \*/ para comentarios de varias líneas. Estos comentarios proporcionan información sobre el propósito del código.
3. Definición de variables: Se definen cuatro variables: $sName, $uName, $pass y $db\_name, que contienen la información necesaria para conectarse al servidor de la base de datos y seleccionar la base de datos específica.
   * $sName: Contiene la dirección del servidor de la base de datos (en este caso, localhost).
   * $uName: Contiene el nombre de usuario para acceder a la base de datos (en este caso, root).
   * $pass: Contiene la contraseña del usuario de la base de datos (en este caso, vacía).
   * $db\_name: Contiene el nombre de la base de datos a la que se quiere conectar (en este caso, escacs\_db).
4. Bloque try-catch: Se utiliza un bloque try-catch para manejar excepciones que pueden ocurrir durante el proceso de conexión a la base de datos.
   * En el bloque try, se intenta establecer una conexión PDO utilizando la información proporcionada en las variables anteriores.
   * Si la conexión es exitosa, se establece el modo de error en ERRMODE\_EXCEPTION, que significa que PDO lanzará excepciones en caso de errores.
   * Si ocurre algún error durante el intento de conexión, se captura la excepción en el bloque catch y se muestra un mensaje de error.
5. ?>: Esto indica el final del código PHP.

En resumen, este código PHP intenta establecer una conexión a la base de datos MySQL utilizando PDO y maneja posibles errores que puedan ocurrir durante este proceso. Si la conexión es exitosa, se establece la conexión para su uso posterior en el script.

**Pol fork**

**Web**

**Php:funcions**

 **Función neteja:** Esta función se utiliza para limpiar los datos ingresados en formularios con el fin de prevenir ataques de Cross-Site Scripting (XSS). Realiza las siguientes operaciones en la cadena de entrada:

* trim($dada): Elimina espacios en blanco al principio y al final de la cadena.
* stripslashes($dada): Elimina las barras invertidas añadidas por la función addslashes.
* htmlspecialchars($dada): Convierte caracteres especiales en entidades HTML. Por ejemplo, convierte < en &lt; y > en &gt;.
* Devuelve la cadena de datos limpiada.

Esta función es útil para procesar datos ingresados por usuarios antes de almacenarlos en una base de datos o mostrarlos en una página web para evitar ataques XSS.

 **Función xifrar:** Esta función se utiliza para cifrar contraseñas antes de almacenarlas en una base de datos. Utiliza el algoritmo Argon2id para el cifrado, que es uno de los algoritmos de hashing más seguros disponibles en la actualidad. Se establecen opciones para el costo de memoria y el tiempo de iteración, lo que afecta a la resistencia a los ataques de fuerza bruta. La función password\_hash toma la contraseña sin cifrar como primer argumento y el algoritmo de cifrado y las opciones como argumentos adicionales, y devuelve la contraseña cifrada.

Esta función es crucial para almacenar contraseñas de forma segura en la base de datos, ya que cifra las contraseñas de manera que no se puedan revertir a su forma original, proporcionando una capa adicional de seguridad en caso de que la base de datos sea comprometida.

**Pol fork**

**Web**

**Php:login**

1. **Inicio de sesión y obtención de datos del formulario:**
   * El script comprueba si se han enviado tanto el nombre de usuario (nom\_usuari) como la contraseña (contrassenya) mediante el método POST.
   * Limpia los datos recibidos utilizando la función neteja definida en el archivo funcions.php. Esta función se encarga de eliminar espacios en blanco, barras invertidas y convertir caracteres especiales a entidades HTML para prevenir ataques XSS.
2. **Validación de los campos enviados:**
   * Si el nombre de usuario o la contraseña están vacíos, se redirige al usuario de vuelta a la página de inicio de sesión (index.php) con un mensaje de error adecuado.
3. **Consulta a la base de datos:**
   * Se prepara una consulta SQL para seleccionar el usuario con el nombre de usuario proporcionado de la tabla usuaris.
   * Se ejecuta la consulta y se obtienen los resultados.
   * Si se encuentra un usuario en la base de datos que coincida con el nombre de usuario proporcionado, se procede a verificar la contraseña.
4. **Verificación de la contraseña:**
   * Se utiliza la función password\_verify para comparar la contraseña enviada con la contraseña almacenada en la base de datos.
   * Si las contraseñas coinciden, se establecen las variables de sesión ($\_SESSION) para el usuario y se redirige a la página de juego (playS.php).
   * Si las contraseñas no coinciden, se redirige de vuelta a la página de inicio de sesión con un mensaje de error.
5. **Manejo de otros casos:**
   * Si no se encuentra ningún usuario con el nombre de usuario proporcionado, se redirige de vuelta a la página de inicio de sesión con un mensaje de error.
   * Si alguien intenta acceder directamente a este script sin enviar los datos del formulario, se le redirige a la página de inicio de sesión con un mensaje de error.

Este script asegura que los usuarios solo puedan acceder al sistema si proporcionan credenciales válidas y maneja varios casos de error de manera adecuada.

**Pol fork**

**Web**

**Php:signupcheck.php**

 **Inclusión de archivos y verificación de datos recibidos:**

* Se incluyen los archivos funcions.php y db\_conn.php que contienen funciones y la conexión a la base de datos, respectivamente.
* Se comprueba si se han enviado los datos necesarios para el registro del usuario a través del método POST: nombre completo (nom\_public), nombre de usuario (nom\_usuari), contraseña (contrassenya), y confirmación de contraseña (rpass).

 **Limpieza y validación de los datos recibidos:**

* Se utilizan las funciones definidas en funcions.php para limpiar los datos recibidos y prevenir ataques XSS.
* Se realizan varias comprobaciones para asegurarse de que todos los campos requeridos se han completado y de que las contraseñas coinciden.

 **Consulta a la base de datos para verificar la disponibilidad del nombre de usuario:**

* Se prepara una consulta SQL para seleccionar usuarios que coincidan con el nombre de usuario proporcionado.
* Se ejecuta la consulta y se comprueba si ya existe un usuario con ese nombre de usuario en la base de datos.
* Si se encuentra un usuario con el mismo nombre de usuario, se redirige de vuelta a la página de registro con un mensaje de error.

 **Proceso de registro:**

* Si el nombre de usuario está disponible y las contraseñas coinciden, se procede a cifrar la contraseña usando la función xifrar definida en funcions.php.
* Se prepara una consulta SQL para insertar un nuevo usuario en la tabla usuaris con los datos proporcionados.
* Se ejecuta la consulta y se registra el nuevo usuario en la base de datos.
* Se redirige de vuelta a la página de registro con un mensaje de éxito.

 **Manejo de otros casos:**

* Si no se reciben todos los datos necesarios para el registro, se redirige de vuelta a la página de registro con un mensaje de error.
* Si se produce algún otro tipo de error, se redirige de vuelta a la página de registro con un mensaje de error genérico.

**Pol fork web : blitz.php**

1. **Verificación de la sesión**: Comprueba si existe una sesión activa para el usuario. Si existe una sesión, muestra el contenido de la página de ajedrez. Si no, redirige al usuario a la página de inicio de sesión.
2. **Contenido de la página de ajedrez**: Si el usuario ha iniciado sesión, muestra el contenido de la página de ajedrez, que incluye:
   * Un encabezado que incluye un saludo personalizado para el usuario.
   * Un tablero de ajedrez representado por un canvas HTML.
   * Indicadores de turno y temporizador para cada jugador.
   * Información sobre el número de piezas restantes para cada jugador.
   * Un botón para iniciar un nuevo juego.
3. **Scripts JavaScript**: Incluye los scripts JavaScript necesarios para el funcionamiento del juego, como blitz.js y matrix.js.
4. **Redirección si no hay sesión**: Si no se encuentra una sesión activa, redirige al usuario a la página de inicio de sesión en lugar de mostrar el contenido del juego de ajedrez.

En resumen, este archivo PHP asegura que solo los usuarios que han iniciado sesión puedan acceder al juego de ajedrez, y redirige a los usuarios no autenticados a la página de inicio de sesión.

**Pol fork web : home.php**

1. **Verificación de la sesión**: Comprueba si existe una sesión activa para el usuario. Si existe una sesión, muestra el contenido de la página de perfil. Si no, redirige al usuario a la página de inicio de sesión.
2. **Contenido de la página de perfil**: Si el usuario ha iniciado sesión, muestra el contenido de la página de perfil, que incluye:
   * Un menú de navegación con opciones como "Jugar" y "Cerrar sesión".
   * Un saludo personalizado para el usuario.
   * Un título "Historial".
   * Una tarjeta de perfil que muestra la foto de perfil y el nombre de usuario.
3. **Redirección si no hay sesión**: Si no se encuentra una sesión activa, redirige al usuario a la página de inicio de sesión en lugar de mostrar el contenido del perfil.

Este archivo PHP sigue el mismo patrón que el anterior, asegurando que solo los usuarios autenticados puedan acceder al perfil y redirigiendo a los usuarios no autenticados a la página de inicio de sesión.

**Pol fork web logout,php**

1. **Inicio y destrucción de la sesión**: Comienza la sesión si no está iniciada, luego libera todas las variables de sesión actualmente registradas y finalmente destruye la sesión.
2. **Redirección**: Después de destruir la sesión, redirige al usuario a la página de inicio (index.php) con un mensaje de éxito en la URL (success=Gracias+por+Jugar!). Esto se hace usando header("Location: ...").
3. **Salida**: La función exit se utiliza después de redirigir al usuario para asegurarse de que no se ejecute más código después de la redirección. Esto garantiza que la redirección sea efectiva y que no se ejecute ningún código adicional en la página actual.

En resumen, este archivo limpia y destruye la sesión del usuario, luego lo redirige a la página de inicio con un mensaje de éxito.

**PLAY.PHP**

1. **DOCTYPE y HTML Tags**: Establecen el tipo de documento y el idioma de la página.

html

Copiar código

<!DOCTYPE html>

<html lang="en">

1. **Head Section**: Aquí se definen metadatos como la codificación de caracteres, la escala de la vista en dispositivos móviles, y se incluyen enlaces a archivos CSS y a la biblioteca de iconos Font Awesome.

html

Copiar código

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="shortcut icon" href="favicon.ico" type="image/x-icon">

<link rel="stylesheet" href="../css/styles.css">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.2/css/all.min.css" integrity="sha512-SnH5WK+bZxgPHs44uWIX+LLJAJ9/2PkPKZ5QiAj6Ta86w+fsb2TkcmfRyVX3pBnMFcV7oQPJkl9QevSCWr3W6A==" crossorigin="anonymous" referrerpolicy="no-referrer" />

<title>Chess Game</title>

</head>

1. **Body Section**: Esta sección contiene el contenido principal de la página. Empieza con la inclusión de un archivo PHP llamado "header.php", que probablemente contiene el encabezado de la página, como la barra de navegación. Luego, tenemos un elemento canvas con el id "canv" y una clase "canvas", que probablemente se utilizará para dibujar el tablero de ajedrez. Después, hay un div con id "container" que contiene toda la interfaz del juego de ajedrez, incluido el tablero, la información sobre el turno, el historial de movimientos y los botones para comenzar un nuevo juego.

html

Copiar código

<body>

<?php include("includes/header.php"); ?>

<main>

<canvas id="canv" class="canvas"></canvas>

<div id="container">

<!-- Contenido de la interfaz del juego de ajedrez -->

</div>

</main>

</body>

1. **Script Tags**: Aquí se incluyen los archivos JavaScript necesarios para el funcionamiento del juego de ajedrez. El archivo "matrix.js" probablemente contiene la lógica del juego, mientras que el archivo "script.js" probablemente maneja la interacción del usuario con la interfaz del juego.

html

Copiar código

<script src="js/matrix.js"></script>

<script src="js/script.js"></script>

En resumen, esta página HTML muestra la interfaz de un juego de ajedrez, con un tablero interactivo y funcionalidades para iniciar un nuevo juego y realizar movimientos. La inclusión de archivos PHP sugiere que el juego puede estar integrado con un sistema de inicio de sesión o con otras funcionalidades dinámicas.

**PLAYS.PHP**

1. **Inicio de Sesión**: Se utiliza session\_start() para iniciar la sesión PHP. Luego, se comprueba si hay una sesión activa y si existen las variables de sesión $\_SESSION['id'] y $\_SESSION['nom\_public']. Si estas variables están definidas, se permite el acceso al contenido de la página.

php

Copiar código

<?php

session\_start();

if(isset($\_SESSION['id'])&& isset($\_SESSION['nom\_public'])):

?>

1. **Etiqueta HTML**: Se inicia el documento HTML con la declaración del tipo de documento y el idioma.

html

Copiar código

<!DOCTYPE html>

<html lang="en">

1. **Head Section**: Aquí se definen metadatos como la codificación de caracteres, la escala de la vista en dispositivos móviles, el icono de la página y los estilos CSS utilizados.

html

Copiar código

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="shortcut icon" href="favicon.ico" type="image/x-icon">

<link rel="stylesheet" href="../css/styles.css">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.2/css/all.min.css" integrity="sha512-SnH5WK+bZxgPHs44uWIX+LLJAJ9/2PkPKZ5QiAj6Ta86w+fsb2TkcmfRyVX3pBnMFcV7oQPJkl9QevSCWr3W6A==" crossorigin="anonymous" referrerpolicy="no-referrer" />

<title>Chess Game</title>

</head>

1. **Body Section**: Aquí se encuentra el contenido principal de la página. Incluye el encabezado (probablemente con un menú de navegación) mediante la inclusión de un archivo PHP llamado "header.php". Luego, tenemos el contenido principal del juego de ajedrez, incluido el tablero, la información sobre el turno, las piezas de ajedrez y los botones para iniciar un nuevo juego.

html

Copiar código

<body>

<?php include("includes/header.php"); ?>

<main>

<!-- Contenido del juego de ajedrez -->

</main>

</body>

1. **Script Tags**: Al final del documento HTML, se incluyen los archivos JavaScript necesarios para el funcionamiento del juego de ajedrez. El archivo "script.js" probablemente maneja la interacción del usuario con la interfaz del juego, mientras que "matrix.js" puede contener la lógica del juego.

html

Copiar código

<script src="js/script.js"></script>

<script src="js/matrix.js"></script>

1. **Final de la Estructura PHP**: Se cierra la estructura de control de flujo PHP con endif.

php

Copiar código

<?php else: ?>

1. **Redireccionamiento**: Si no hay una sesión activa, el usuario es redirigido a la página de inicio de sesión (login.php).

php

Copiar código

<?php

header("Location:php/login.php");

?>

Espero que esta explicación te ayude a comprender

el funcionamiento de este código. Si tienes alguna otra pregunta o necesitas más detalles sobre algún aspecto específico, no dudes en preguntar.