

Joc d'escacs

Introducción:

Este proyecto consiste en una aplicación web de ajedrez para jugar online contra oponentes humanos. Al basarse enteramente en la arquitectura cliente - servidor, los usuarios no tienen que descargar ni instalar nada en sus dispositivos.

Aunque la temática del proyecto no es en absoluto novedosa, sí suele ser compleja y por tanto difícil de implementar, por lo que entendimos que desarrollarlo es un logro en sí mismo.

Hemos querido centrarnos en la arquitectura que conecta a los jugadores entre sí y cómo gestiona los mensajes que los clientes se mandan a través del servidor, así que la posibilidad de jugar contra una IA nunca ha formado parte del proyecto.

Requisitos y casos de uso:

Los usuarios deben registrarse a través de la página web de la aplicación, definiendo obligatoriamente un nombre de usuario y una contraseña. Una vez registrado, cada usuario puede personalizar su perfil adjuntando una foto y/o escribiendo una descripción, pudiendo mostrarse ambos durante las partidas. El usuario también empieza con una puntuación de 800 como jugador, imitando el sistema ELO. Además tiene una sección donde ver las notaciones de sus partidas.

La aplicación empareja los jugadores mediante sockets en función de su ELO (máximo 100 puntos de diferencia entre ambos) y les crea una partida, cuyo tablero y piezas se pintan en pantalla mediante canvas. Cada jugador tiene un cronómetro de 10 minutos para jugar la partida. El tiempo cuenta hacia atrás mientras está en su turno y se para durante el de su oponente.

Cuando la partida termina, cada jugador gana o pierde puntos de ELO según corresponda y se actualiza su puntuación; además, se añade la notación de la partida a ambos perfiles. En caso de tablas, los puntos ELO no se modifican. Si un jugador se desconecta o deja que su tiempo total se agote, la partida termina automáticamente y le cuenta como derrota.

La aplicación dispone de un ranking donde aparecen los mejores jugadores ordenados por puntuación ELO. Cualquier visitante a la web puede ver este ranking. Si se trata de un usuario registrado y ha hecho login, su posición le aparece destacada en colores.

Para la **interfaz de usuario** hemos planteado ciertas características:

- Cada jugador ve el tablero orientado para que su bando aparezca en el lado inferior
- El tablero tiene marcadas las coordenadas de las filas y columnas
- Los colores de las casillas del tablero pueden personalizarse (dando a elegir entre una lista)
- Las piezas se mueven haciendo clic principal sobre ellas y arrastrando
- Mientras se mantiene clic sobre una pieza propia, el tablero marca a qué casillas puede moverse
- Hacer clic secundario sobre una casilla y arrastrar el ratón hacia otra pinta una flecha
 - * Se pueden pintar múltiples flechas a la vez
 - * Hacer clic principal sobre el tablero las borra todas
 - * Cada jugador puede ver sólo sus propias flechas
- La última pieza que se ha movido tiene su casilla de destino resaltada con otro color
- Si un rey está en jaque, su casilla actual se pinta de rojo
- Cada jugador tiene un botón que puede pulsar para pedir tablas, su oponente puede aceptar o no
 - * 15 segundos para responder sí o no, si se acaba el tiempo se toma como un no
 - * El jugador que debe responder puede ver el tiempo que le queda
 - * Si se rechaza la propuesta de tablas, no se puede pedir otra hasta dentro de 10 turnos

La **lógica del juego** debe tener en cuenta varios parámetros:

- Rechazar los movimientos de un jugador si no está en su turno
- Movimiento general de cada tipo de pieza
- Detectar qué casillas que por defecto serían viables están bloqueadas
 - * Para los reyes se incluye qué casillas están siendo atacadas por piezas enemigas
- Cada pieza tiene una lista actualizada de a qué casillas puede moverse y qué casillas puede atacar
- Detectar cuándo un rey está en jaque
 - * Permitir sólo los movimientos que resuelvan la situación

- Tener en cuenta los movimientos que siguen **reglas especiales**

* Enroque

* Avance inicial de 2 casillas para los peones

* Captura *en passant* de peones

* Coronación de peones (para simplificar esta regla, asumimos que siempre coronan a dama)

También se deben tener en cuenta las situaciones especiales dentro de la partida, como los jaques y los mates, además de las reglas especiales del juego que no están directamente ligadas a los movimientos de las piezas; hablamos principalmente de las situaciones que derivan en tablas:

* Rey ahogado (uno de los jugadores, sin estar en jaque, no puede hacer ningún movimiento legal)

* Tablas muertas (ningún jugador tiene material y/o espacio suficiente en el tablero para ganar)

* Repetición triple (una determinada posición de partida se repite un mínimo de 3 veces)

* Regla de los 50 movimientos (todos seguidos y sin que se avancen peones ni capturen piezas)

Las dos últimas reglas mencionadas no provocan que la partida termine automáticamente en tablas, sino que otorga al siguiente jugador al que toque mover el derecho a reclamarlas, sin estar obligado a ello; no obstante, para simplificar el proyecto, la partida se declara en tablas automáticamente.

La **base de datos** contiene 3 tipos de tabla:

- Usuario: nombre, foto, contraseña, descripción, ELO (default 800)

- Partida: id_juego, id_usuario_blancas, id_usuario_negras, tiempo_blancas, tiempo_negras

- Movimiento: num_movimiento, id_partida, id_jugador, movimiento

Cada partida genera su tabla 'movimiento' detallando todos los movimientos. Esto se convertirá luego en la notación de la partida, que sus jugadores podrán consultar.

Planteamiento inicial sobre la lógica del juego

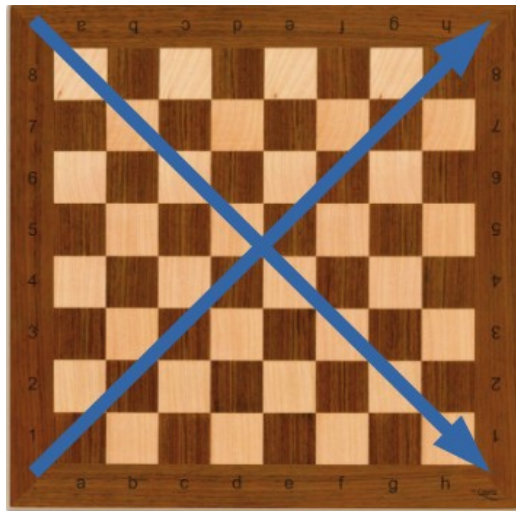
Movimiento general (tablero vacío):

En nuestro planteamiento inicial, el tablero es una matriz de 8 filas y 8 columnas. Las coordenadas van del [0,0] al [7,7].

Cada pieza tiene un array con las coordenadas de las casillas a las que puede moverse. Partiendo de la base que la pieza está en la casilla [x,y]:

- **Horizontalmente:** se añaden al array las casillas de [0,y] a [7,y] sin contar la casilla actual
- **Verticalmente:** se añaden al array las casillas de [x,0] a [x,7] sin contar la casilla actual
- **Diagonalmente:** Cada pieza que puede moverse en diagonal tiene 2 diagonales disponibles, una ascendiente y otra descendiente. Elegiremos recorrerlas de izquierda a derecha.

Esto significa que el origen de las diagonales se buscará lo más a la izquierda posible en el tablero.



La figura muestra las 2 grandes diagonales del tablero, que van de esquina a esquina. La de casillas blancas empieza en [0,0] y acaba en [7,7]. La de casillas negras empieza en [0,7] y acaba en [7,0]. Para más comodidad las llamaremos 'diagonal descendiente' y 'diagonal ascendiente'.

Las demás diagonales del tablero son más cortas porque no empiezan en las esquinas, pero todas tienen en común que empiezan en los bordes del tablero.

Esto significa que una de sus coordenadas será siempre 0 o 7, según qué diagonal estemos calculando. También sabemos que ninguna coordenada podrá ser negativa ni superior a 7 porque implicaría salirse del tablero.

Además, sabemos que una diagonal se traza a base de sumar o restar ambas coordenadas siempre por un mismo valor. Para trazar diagonales descendientes, hacemos $[x+1,y+1]$ para cada casilla que queramos anotar. Para las ascendientes hacemos $[x+1,y-1]$. Haciendo así las operaciones recorreremos las diagonales de izquierda a derecha, tal como nos hemos propuesto.

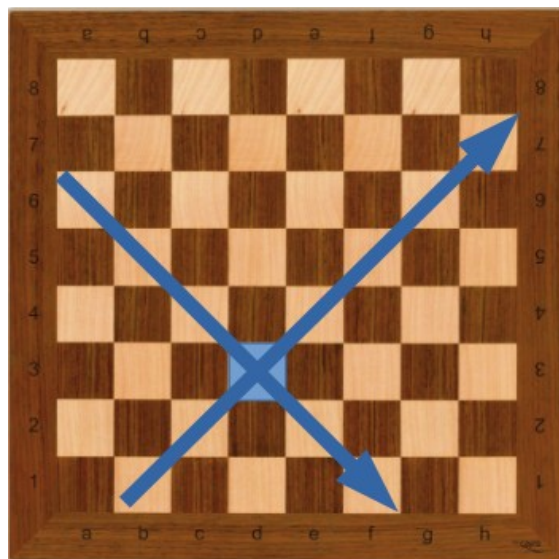
Volviendo al planteamiento inicial de que tenemos una casilla cualquiera $[x,y]$, para calcular sus diagonales hacemos lo siguiente:

- Diagonal descendiente: debemos encontrar qué casilla se acerca más a $[0,0]$. Para hacerlo medimos la distancia entre 0 y el valor de cada coordenada, es decir, $x-0$ e $y-0$. Dicho de otra forma, debemos averiguar qué valor es más pequeño. Luego sólo debemos restar ese valor a ambas coordenadas.

Ejemplo: $[3,5]$. 3 es más pequeño, por tanto $[3-3,5-3] =$ su diagonal descendiente empieza en $[0,2]$.

- Diagonal ascendiente: debemos encontrar qué casilla se acerca más a $[0,7]$. La distancia de 0 hasta x es igual que en el caso anterior. La distancia de y hasta 7 es $7-y$. Por tanto, hay que comparar qué valor es más pequeño: x , o $7-y$. Ese valor debe restarse a x y sumarse a y .

Ejemplo: $[3,5]$. $7-5=2$, por tanto es más pequeño. $[3-2,5+2] =$ su diagonal ascendiente empieza en $[1,7]$.



- **Salto de caballo:** hay varias formas de calcularlos. Optaremos por calcular las coordenadas 2 casillas hacia cada dirección horizontal y vertical. Es posible que algunas direcciones sean inviables si el caballo está demasiado cerca del borde del tablero. Un primer esquema para $[x,y]$ es:

* Direcciones horizontales: $[x-2,y]$ y $[x+2,y]$

* Direcciones verticales: $[x,y-2]$ y $[x,y+2]$

Para cada coordenada que se ha podido calcular, se comprueban las casillas adyacentes a esa coordenada en dirección perpendicular. Es decir, si hemos trazado una dirección horizontal, miramos las casillas verticales adyacentes, y viceversa.

Ejemplo: $[4,0]$. Las primeras direcciones a mirar serían $[2,0]$ y $[6,0]$ en sentido horizontal, y $[4,-2]$ y $[4,2]$ en sentido vertical. Todas son viables excepto $[4,-2]$.

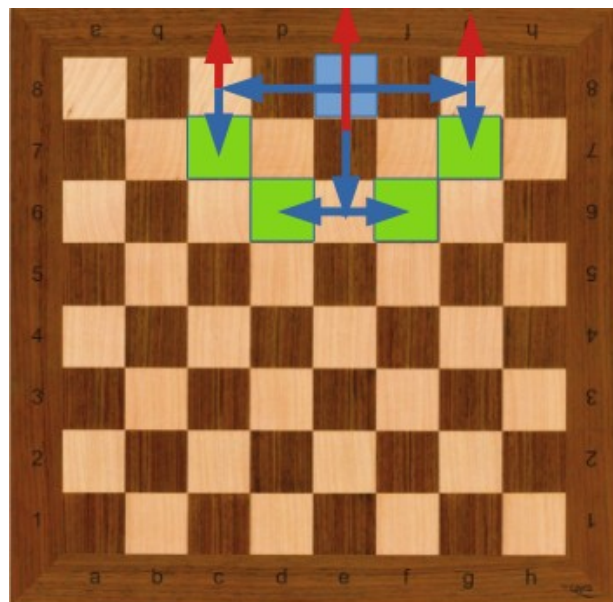
Para $[2,0]$, al venir de una dirección horizontal, sus adyacentes verticales a mirar son $[2,-1]$ y $[2,1]$.

Para $[6,0]$ serían $[6,-1]$ y $[6,1]$.

Para $[4,2]$, al ser vertical, sus adyacentes horizontales son $[3,2]$ y $[5,2]$.

Todas las casillas que se han mirado son viables excepto $[2,-1]$ y $[6,-1]$.

La siguiente figura ilustra estos cálculos:



- **Peones:** como sólo se mueven hacia adelante, es cuestión de sumar o restar 1 a la coordenada y en función de si el peón es blanco o negro.

- **Reyes:** sólo se miran las 8 casillas adyacentes a la actual, aunque si bien no es complicado mirarlas todas, podemos simplificar los cálculos mirando primero las verticales y horizontales; si la casilla actual está en un borde del tablero, habrá direcciones horizontales y/o verticales que no serán válidas, y por tanto sus diagonales tampoco lo serán.

Se tratará de comprobar si las coordenadas de la casilla actual contienen algún 0 o 7.

$x=0$: la casilla está en el borde izquierdo del tablero. Invalida moverse a la izquierda $[x-1,y]$ y por tanto las diagonales izquierdas $[x-1,y-1]$ y $[x-1,y+1]$.

$x=7$: la casilla está en el borde derecho. Invalida $[x+1,y]$ y las diagonales $[x+1,y-1]$ y $[x+1,y+1]$.

$y=0$: la casilla está en el borde superior. Invalida $[x,y-1]$ y las diagonales $[x-1,y-1]$ y $[x+1,y-1]$.

$y=7$: la casilla está en el borde inferior. Invalida $[x,y+1]$ y las diagonales $[x-1,y+1]$ y $[x+1,y+1]$.

Lo más sencillo para cada dirección será realizar sus cálculos sólo si sus coordenadas no son 0 ni 7. A partir de ahí, se añadirán las casillas al array de movimientos.

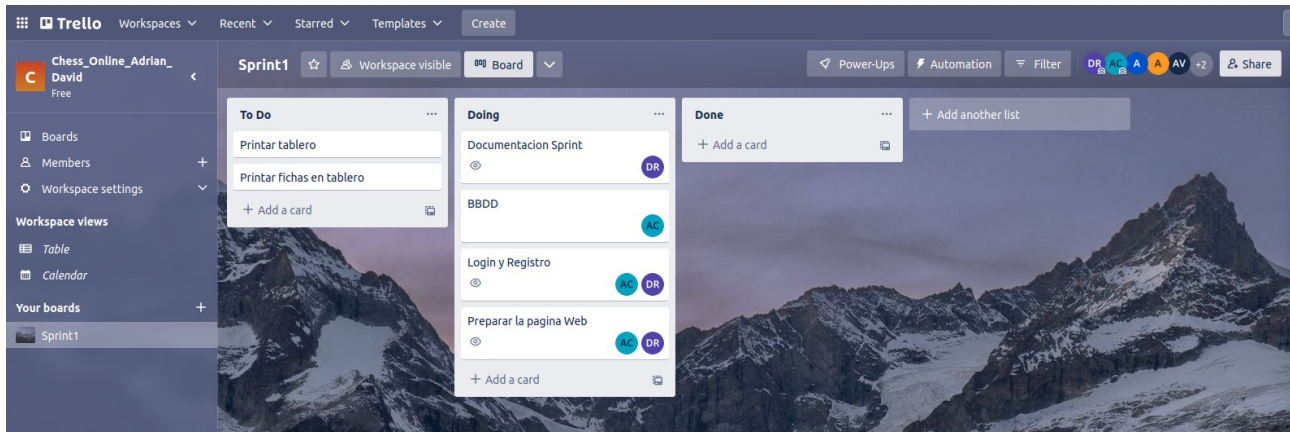
Movimientos especiales:

Peones, reyes y torres pueden ejecutar determinados movimientos cuyo común denominador es que debe ser su primer movimiento de la partida. Para los peones es la capacidad de moverse 2 casillas en lugar de 1, y en el caso de reyes y torres es el enroque.

Bastará con asignarles individualmente una variable booleana, por ejemplo 'primerMovimiento = true', y requerir que tenga ese valor para que puedan ejecutar esos movimientos. Una vez esas piezas ejecutan su primer movimiento, sea cual sea, ese booleano cambia de valor ('false' en este caso) para el resto de la partida.

Implementación:

La parte del front end ha sido implementada con **Html**, **Css** y **JavaScript**, y la parte del back end, con **Php**. No se han utilizado frameworks para ningún lenguaje, aunque sí el IDE de **Visual Studio** para gestionar la organización de los ficheros y vincular el proyecto con un repositorio remoto en **GitHub** mediante la versión de **Git** de la que dispone Visual Studio. También hemos coordinado las tareas iniciales a través de Trello.



La base de datos elegida es **MySQL**. El fichero sql que crea las tablas está adjunto en el proyecto.

Los cimientos de la lógica de la aplicación y la parte de la interfaz relacionada con la creación del tablero y las piezas se han importado a través de las librerías chess.js y chessboard.js, respectivamente.

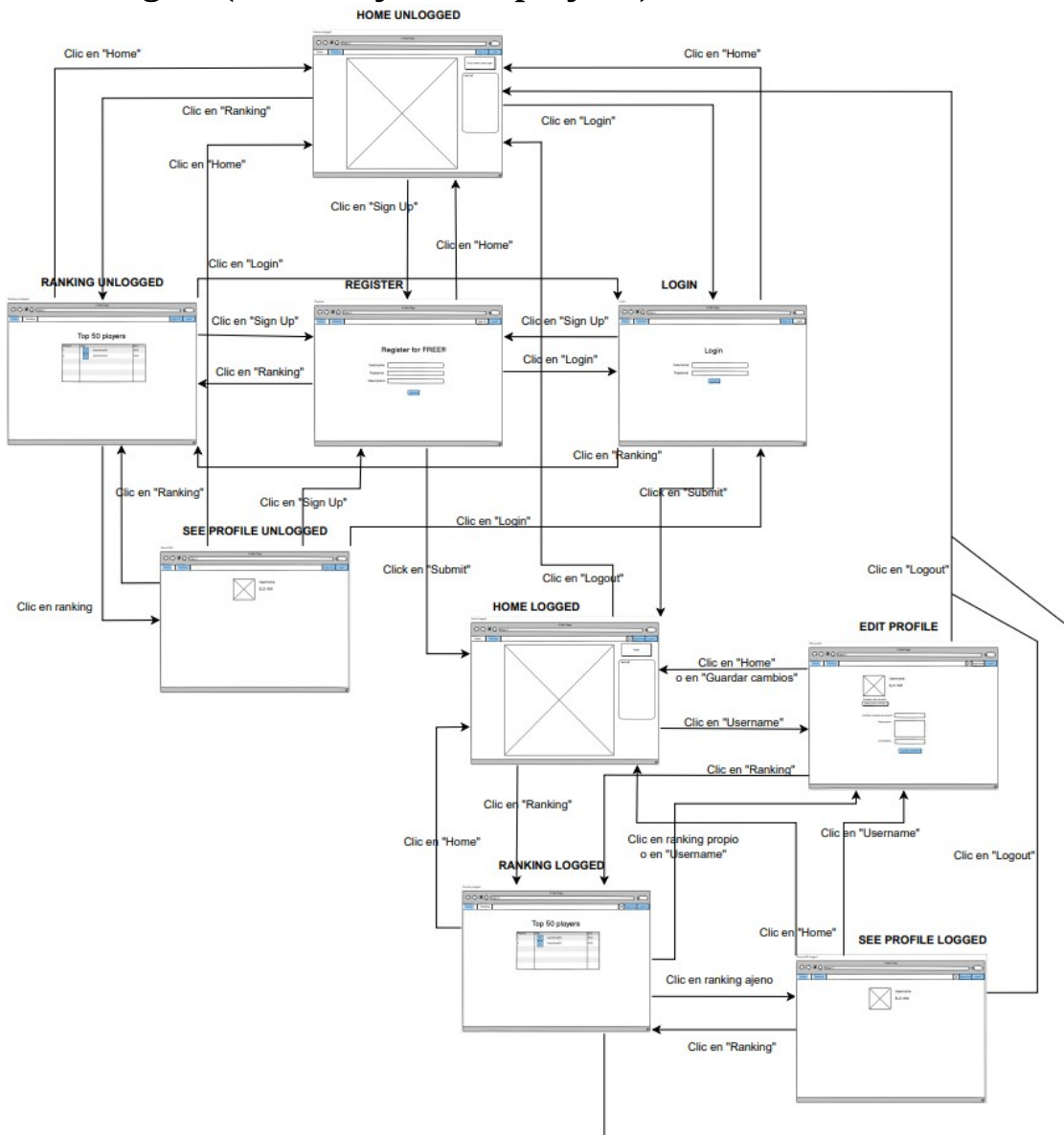
Diseño:

El proyecto incluye diagramas de casos de uso, navegación y entidad-relación, además de un wireframe creado con Balsamiq. Todos estos ficheros han sido adjuntados aparte de este documento.

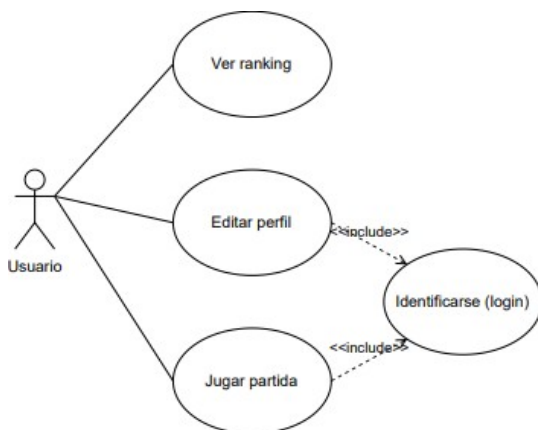
También tenemos un mockup usable en el pdf Balsamiq-chess.

Diagramas:

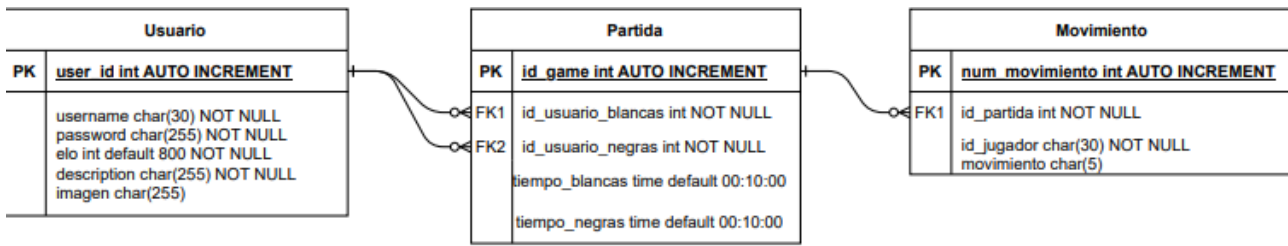
Navegaci3(tambi3 adjuntat al projecte):



Cas 3s:



Entitat-relació:



Capturas de paginas:

Index.php

La principal pagina del proyecto, donde tienes un tablero para jugar en local con tu compañero de al lado o para practicar. Tambien incluye un botón que solo está operativo si tienes session iniciada. El recuadro de negro es un div donde con cada movimiento que haces lo printas en notación de ajedrez. Abajo tienes un boton para reiniciar el tablero y el registro de los movimientos. ¡¡¡Tambien hay un boton para buscar partida online, pero actualmente no està operativo!!!

Sin iniciar sesion:



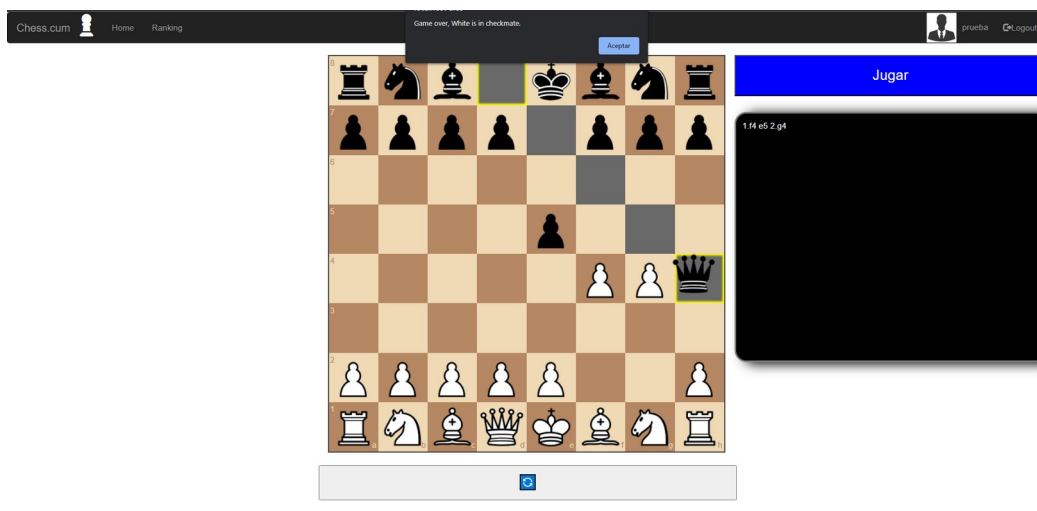
Iniciando sesion:



Jugando:



Hay ganador:



BBDD

Tabla usuario:

+ Opciones

			user_id	username	password	elo	description	imagen
<input type="checkbox"/>				73	prueba	\$2y\$10\$EdejeQEbaRK3cqpnQbjShJ.2dmquWZ471.fE7o8nmwqF...	900 soy el mejor	profilepic/default.jpg
<input type="checkbox"/>				74	prueba2	\$2y\$10\$SrAZysj2y1h6oVbRxoA1zd.6vVWZZ1a7X7GR0VxDA8n...	800	profilepic/default.jpg
<input type="checkbox"/>				75	prueba3	\$2y\$10\$OxzY9aL4h/8aLgXz6fMOBu6G62xzigtgJUAL0eeQd.Q...	800 soy promedio	profilepic/default.jpg
<input type="checkbox"/>				76	prueba4	\$2y\$10\$IPTK6shpwJcgUjavedZtUu/7iGMg0P0o2drFrWTx05K...	800	profilepic/default.jpg
<input type="checkbox"/>				77	prueba5	\$2y\$10\$zM6NRN7w10o0sInsjuym.1DdnsZLXMmBIFoXgEr7ds...	700 soy el peor	profilepic/default.jpg

Servidor: 127.0.0.1 » Base de datos: chess » Tabla: usuario

Examinar

Estructura

SQL

Buscar

Insertar

Exportar

Importar

Privilegios

Operaciones

Seguimiento

Disparadores

Estructura de tabla

Vista de relaciones

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1 user_id	int(5)			No	Ninguna		AUTO_INCREMENT	<div><div></div><div>Cambiar</div><div>Eliminar</div><div>Más</div></div>
<input type="checkbox"/>	2 username	varchar(30)	utf8_general_ci		No	Ninguna			<div><div></div><div>Cambiar</div><div>Eliminar</div><div>Más</div></div>
<input type="checkbox"/>	3 password	varchar(255)	utf8_general_ci		No	Ninguna			<div><div></div><div>Cambiar</div><div>Eliminar</div><div>Más</div></div>
<input type="checkbox"/>	4 elo	int(4)			No	800			<div><div></div><div>Cambiar</div><div>Eliminar</div><div>Más</div></div>
<input type="checkbox"/>	5 description	varchar(255)	utf8_general_ci		No	Ninguna			<div><div></div><div>Cambiar</div><div>Eliminar</div><div>Más</div></div>
<input type="checkbox"/>	6 imagen	varchar(255)	utf8_general_ci		No	profilepic/default.jpg			<div><div></div><div>Cambiar</div><div>Eliminar</div><div>Más</div></div>

Tabla partida:



Servidor: 127.0.0.1 » Base de datos: chess » Tabla: partida

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Seguimiento Disparadores

Estructura de tabla Vista de relaciones

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1 id_game	int(9)			No	Ninguna		AUTO_INCREMENT	Cambiar Eliminar Más
<input type="checkbox"/>	2 id_usuario_blancas	int(5)			No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	3 id_usuario_negras	int(5)			No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	4 tiempo_blancas	time			No	00:10:00			Cambiar Eliminar Más
<input type="checkbox"/>	5 tiempo_negras	time			No	00:10:00			Cambiar Eliminar Más

Tabla movimiento:



Servidor: 127.0.0.1 » Base de datos: chess » Tabla: movimiento

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Seguimiento Disparadores

Estructura de tabla Vista de relaciones

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1 num_movimiento	int(3)			No	Ninguna		AUTO_INCREMENT	Cambiar Eliminar Más
<input type="checkbox"/>	2 id_partida	int(9)			No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	3 id_jugador	varchar(30)	utf8_general_ci		No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/>	4 movimiento	varchar(5)	utf8_general_ci		No	Ninguna			Cambiar Eliminar Más

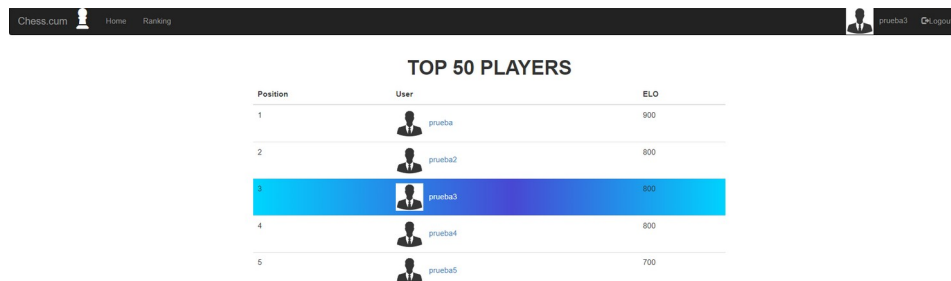
Ranking.php

Página donde muestras los 50 usuarios con más ELO de la BBDD, aunque ahora solo hay 5 porque son los que hay ahora en la BBDD.

En caso de que tu usuario de sesión esté en el Top 50, sale destacado con Azul.

Al pulsar en uno de estos usuarios, te redirige a su perfil

Usuario iniciado








Chess.cum Home Ranking

TOP 50 PLAYERS

Position	User	ELO
1	prueba	900
2	prueba2	800
3	prueba3	800
4	prueba4	800
5	prueba5	700

Usuario no iniciado

TOP 50 PLAYERS		
Position	User	ELO
1	 prueba	900
2	 prueba2	800
3	 prueba3	800
4	 prueba4	800
5	 prueba5	700

Signup.php(crear usuarios)

Solo se puede acceder a él SIN la sesion iniciada!!!

Al crear un usuario eres redirigido a index.php CON session iniciada.

Usuario no existente y contraseña segura

Register for FREE!!!

Username:

Password:

Descripcion:

Usuario ya existente(detectedo con el fichero signupCheck.js)

Register for FREE!!!

Username:

Password:



El usuario ya existe

Descripcion:

Contraseña insegura(detectedo con el fichero signupCheck.js)

Register for FREE!!!

Username:

Password:

Descripcion:

! Contraseña insegura

login.php(iniciar sesion)

Solo se puede acceder a él SIN la sesion iniciada!!!

Al iniciar sesion eres redirigido a index.php CON session iniciada.

Usuario existente(detectedo con el fichero loginCheck.js)

Login

Username:

Password:

Usuario no existente(detectedo con el fichero loginCheck.js)

Login

Username:

Password:

! El usuario no existe

Usuario existente pero la contraseña es incorrecta(redireccion a la misma página).

Login

Username:

Password:

Submit

logout.php(cerrar sesion)

Solo se puede acceder a él CON la sesion iniciada!!!

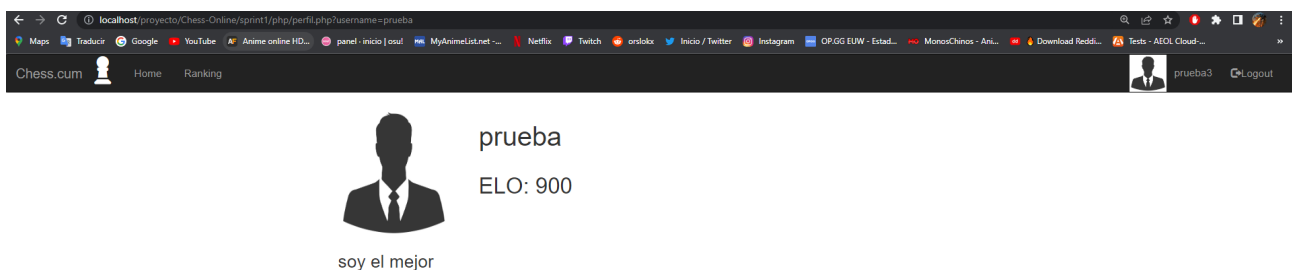
Al iniciar sesion eres redirigido a index.php SIN session iniciada.

Perfil.php

A traves de la variable username recupera y presenta toda la informacion de un usuario, menos obviamente la contraseña.

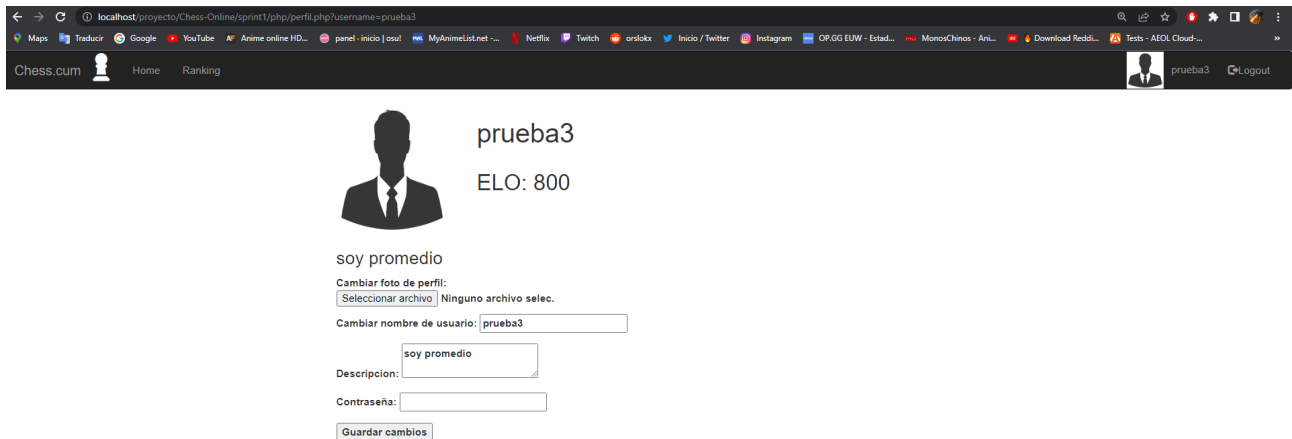
Se puede acceder a el pulsando en el nombre de un usuario en Ranking.php, o bien pulsando en tu propio perfil arriba a la derecha.

Otro perfil




Mi perfil

Si el perfil seleccionado es el mismo que tu session iniciada, obtienes tambien un formulario para actualizar varios datos del usuario a traves del fichero perfilModificar.php



Chess.cum Home Ranking prueba3 Logout

 prueba3
ELO: 800

soy promedio

Cambiar foto de perfil:
 Ninguno archivo selec.

Cambiar nombre de usuario:

Descripcion:

Contraseña:

perfilModificar.php

Solo se puede acceder a él CON la sesion iniciada y si coincide tu usuario con el usuario enviado a perfil.php.

Para modificar tu usuario tienes que poner tu contraseña en el campo contraseña. Si no es correcta los cambios no tendran lugar.

¡¡¡Aunque seria facil de implementar, hemos optado por no añadir la opcion de borrar usuario, ya que registrarse no requiere correo.!!!

Modificamos correctamente el nombre de usuario, la descripcion y la foto.



Chess.cum Home Ranking prueba3modificado Logout

 prueba3modificado
ELO: 800

no soy promedio

Cambiar foto de perfil:
 Ninguno archivo selec.

Cambiar nombre de usuario:

Descripcion:

Contraseña:

El cambio lo que modifica es el usuario de la BBDD y el username de la session.

+ Opciones			user_id	username	password	elo	description	imagen
<input type="checkbox"/>	Editar	Copiar	Borrar	73	prueba	\$2y\$10\$EdjeQEbaRK3cqQnQbjShJ.2dmquWZ471.fE7o8nmwqF...	900 soy el mejor	profilepic/default.jpg
<input type="checkbox"/>	Editar	Copiar	Borrar	74	prueba2	\$2y\$10\$rAZysj2y1h6oVbRxoA1zd.6vVWtZZ1a7X7GR0VxDA8n...	800	profilepic/default.jpg
<input type="checkbox"/>	Editar	Copiar	Borrar	75	prueba3modificado	\$2y\$10\$zrUuQM4eP/r7m7tCoNI7deaok2QWz2D7S.ZkRerUJNa...	800 no soy promedio	profilepic/10773947_1685902610.jpg
<input type="checkbox"/>	Editar	Copiar	Borrar	76	prueba4	\$2y\$10\$IPTK6shpwJcgUjavedZtUu/7tGMg0P0o2drFrWTx05K...	800	profilepic/default.jpg
<input type="checkbox"/>	Editar	Copiar	Borrar	77	prueba5	\$2y\$10\$VzZM6NRN7w1lo0slnsjuym.1DdnsZLXMmBlFoXgEr7ds...	700 soy el peor	profilepic/default.jpg

Codigo:

Index.js: El codigo principal de nuestro programa es el siguiente:

```
$(document).ready(function() {
  iniciarTablero();

  //Reiniciamos el tablero
  //Limpiamos el texto de la notación
  document.getElementById('startBtn').addEventListener('click', function() {
    //Location.reload();
    document.getElementById('playLog').innerHTML = "";
    iniciarTablero();
  });
});
```

La funcion iniciar tablero es llamada al cargar la pagina por primera vez, y al pulsar el boton de reiniciar.

Esta funcion se encarga de crear un objeto llamado Chess, con sus características.El objeto Chess proviene de la librería Chess.js, la cual nos dio algunos problemas para la implementacion. Este problema lo arreglamos importando la librería de Chess.js mediante cdn, en lugar de instalarla.

```
//Logica de ajedrez
function iniciarTablero() {
  var board = null
  var game = new Chess()
  var whiteSquareGrey = '#a9a9a9'
  var blackSquareGrey = '#696969'
  var $status = $('#status')
  var $fen = $('#fen')
  var $pgn = $('#pgn')
  var contColor = 1;
  var contPrint = 1;

  var config = {
    draggable: true,
    position: 'start',
    onDragStart: onDragStart,
    onDrop: onDrop,
    onMouseoutSquare: onMouseoutSquare,
    onMouseoverSquare: onMouseoverSquare,
    onSnapEnd: onSnapEnd
  }

  board = Chessboard('board1', config)

  updateStatus()
```

La funcion `updateStatus` tambien és muy importante, ya que es llamada cada vez que se realiza un movimiento válido, por lo que es muy necesaria para validar los movimientos siguientes al recién hecho, a parte de detectar jaques y jaque mates.

El resto del codigo se puede revisar mirando el fichero `index.js`, esto es un resumen general!

UserCheck.php: Fichero que se encarga únicamente de comprobar si un usuario enviado por formulario existe o no. Lo usamos para realizar llamadas AJAX sobre él con los ficheros `loginCheck.js` y `signupCheck.js`.

```
<?php
include_once "connexio.php";
$connex = new mysqli($lloc, $usuari, $pwd, $bbdd);
// check connection
if ($connex->connect_error) {
    die("Connection failed: " . $connex->connect_error);
}
$username = $_POST["username"];

if (!empty($username)) {
    $stmt = $connex->prepare("SELECT username FROM usuario WHERE username=?");
    $stmt->bind_param('s', $username);
    $stmt->execute();
    $stmt->store_result();
    $num_rows = $stmt->num_rows;

    if ($num_rows > 0) {
        echo "existe";
    } else {
        echo "noexiste";
    }
    $stmt->close();
} else {
    echo "noexiste";
}

?>
```

LoginCheck.js: Comprueba mediante el fichero anterior que un usuario SI exista.
En caso de existir, pinta el borde del input texto de username a verde.
En caso de NO existir, lo pinta de rojo y muestra un error personalizado.

```
window.onload = function(){

    document.loginForm.username.addEventListener("blur",databaseCheck);
    //document.loginForm.password.addEventListener("blur",passwordCheck);

    function databaseCheck(){
        var username = this;
        var usernameData = username.value;
        var datos = "username=" + encodeURIComponent(usernameData);
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "../php/userCheck.php", true);
        xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        xhr.onreadystatechange = function() {
            if (xhr.readyState == 4 && xhr.status == 200) {
                var resultado = xhr.responseText;
                if (resultado == "existe") {
                    // El valor existe, borrar mensaje de error
                    username.setCustomValidity("");
                    username.style.border = "2px solid green";
                }
                else {
                    // El valor no existe, mostrar mensaje de error
                    username.setCustomValidity("El usuario no existe");
                    username.reportValidity();
                    username.style.border = "2px solid red";
                }
            }
        }
        xhr.send(datos);
    }
}
```

SignupCheck.js: Comprueba mediante el fichero anterior que un usuario NO exista.
En caso de NO existir, pinta el borde del input texto de username a verde.
En caso de existir, lo pinta de rojo y muestra un error personalizado

```
window.onload = function(){

    document.signupForm.username.addEventListener("blur",databaseCheck);
    document.signupForm.password.addEventListener("blur",passwordCheck);

    function databaseCheck(){
        var username = this;
        var usernameData = username.value;
        var datos = "username=" + encodeURIComponent(usernameData);
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "../php/userCheck.php", true);
        xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        xhr.onreadystatechange = function() {
            if (xhr.readyState == 4 && xhr.status == 200) {
                var resultado = xhr.responseText;
                if (resultado == "existe") {
                    // El valor existe, mostrar mensaje de error
                    username.setCustomValidity("El usuario ya existe");
                    username.reportValidity();
                    username.style.border = "2px solid red";
                } else {
                    // El valor no existe, borrar mensaje de error
                    username.setCustomValidity("");
                    username.style.border = "2px solid green";
                }
            }
        }
        xhr.send(datos);
    }
}
```

Pero este código no acaba aquí, ya que tiene una segunda parte reservada a medir la complejidad de la contraseña, en caso de no obtener 2 o más puntos de complejidad la dará por nula, cambiará el color del borde a rojo y mostrará un error personalizado.

Si la contraseña obtiene al menos 2 puntos, pintará el borde de verde y la dará por válida.

También hay una barra de colores debajo del campo de contraseña, que nos indica del 0 al 4 los puntos totales que tiene nuestra contraseña en ese momento.

```

function passwordCheck(){
    var password = this;
    var passwordData = password.value;

    if (meter.value <= 2) {
        password.setCustomValidity("Contrasenya insegura");
        password.reportValidity();
        password.style.border = "2px solid red";
    }

    else {
        // El valor no existe, borrar mensaje de error
        password.setCustomValidity("");
        password.style.border = "2px solid green";
    }
}

//Barra de força de contrasenya
var password = document.getElementById('password');
var meter = document.getElementById('password-strength-meter');

document.signupForm.password.addEventListener('input', function(){
    var val = password.value;
    var result = zxcvbn(val);

    // Canviem el valor del meter
    meter.value = result.score;
});

```

login.php: Comprueba mediante una consulta a la BBDD que el usuario introducido existe y que la contraseña introducida equivale a la que tiene ese usuario en concreto.

En caso de ser correcto, inicias session con esos datos, y eres redirigido a la pagina principal.

En caso de no ser correcto alguno de los datos, eres redirigido otra vez al formulario

```
login.php M X
sprint1 > php > login.php
6  include_once "connexio.php";
7  $connex = new mysqli($lloc, $usuari, $pwd, $bbdd);
8  // check connection
9  if ($connex->connect_error) {
10     die("Connection failed: " . $connex->connect_error);
11 }
12 // get username and password from form
13 $username = $_POST["username"];
14 $password = $_POST["password"];
15 // prepare statement
16 $stmt = $connex->prepare("SELECT * FROM usuario WHERE username = ?");
17 $stmt->bind_param("s", $username);
18 $stmt->execute();
19 $result = $stmt->get_result();
20 // check if user exists
21 if ($result->num_rows == 1) {
22     // user exists, check password
23     $row = $result->fetch_assoc();
24     //Obtenemos la contraseña cifrada almacenada en la base de datos
25     $bbdd_password = $row["password"];
26     //Comparamos la contraseña introducida con la cifrada de la bbdd
27     if (password_verify($password, $bbdd_password)) {
28         // password is correct, start session and redirect to home page
29         session_start();
30         $_SESSION["username"] = $username;
31         header("Location: index.php");
32     } else {
33         // password is incorrect, show error message
34         header("Location: login.html");
35     }
36 } else {
37     // user does not exist, show error message
38     header("Location: login.html");
39 }
```

signup.php: Comprovamos que el usuario introducido no existe y que la contraseña introducida cumple con los requisitos de seguridad.

También nos encargamos de cifrar la contraseña enviada para mayor seguridad en caso de que el usuario no exista.

```

<?php
include_once "connexio.php";
$connex = new mysqli($lloc, $usuari, $pwd, $bbdd);

// check connection
if ($connex->connect_error) {
    die("Connection failed: " . $connex->connect_error);
}

// get username and password from form
$username = $_POST["username"];
$password = $_POST["password"];
$hashed_password = password_hash($password, PASSWORD_DEFAULT);
$description = $_POST["description"];
if(empty($_POST["username"])){
    header("Location: signup.html");
}

//Crear la consulta SQL
$sql = "INSERT INTO usuario (username, password, description) VALUES ('$username', '$hashed_password', '$description')";
// Insertar Los datos en La base de datos
if ($connex->query($sql) === TRUE) {
    session_start();
    $_SESSION["username"] = $username;
    header("Location: index.php");
}
else {
    header("Location: signup.html");
}

// Cerrar la conexión a La base de datos
mysqli_close($connex);
?>

```

Perfil.php: Esto es solo lo principal del fichero. Printas los datos del usuario introducido en la URL y si coincide con el tuyo, a parte printas un formulario de modificacion de datos.

```

perfil.php x
sprintl > php > perfil.php
70 <div class="col-sm-6 col-sm-offset-3 col-md-6 col-md-offset-3 col-lg-6 col-lg-offset-3 col-xs-6 col-xs-offset-3">
71 <div class="row">
72 ">
73 <h1 id="username" class="col-sm-8 col-md-8 col-lg-8 col-xs-8"><?php echo $user_data["username"]?></h1>
74 <div class="row">
75 <h2 id="elo" class="col-sm-6 col-md-6 col-lg-6 col-xs-6">ELO: <?php echo $user_data["elo"]?></h2>
76 </div>
77 <h3 id="description" class="col-sm-6 col-md-6 col-lg-6 col-xs-6"><?php echo $user_data["description"]?></h3>
78 </div>
79 <?php
80 //Si el perfil revisat és el propi, afegim el formulari de modificació
81 if (isset($_SESSION["username"])){
82     if($_SESSION["username"] == $user_data["username"]){
83         ?>
84         <form method="post" action="perfilModificar.php" enctype="multipart/form-data">
85             <label for="image">Cambiar foto de perfil:<p>
86             <input type="file" name="image" id="image">
87             <p>
88             <label for="name">Cambiar nombre de usuario:
89             <input type="text" name="name" value="<?php echo $myuser_data["username"]?>">
90             <p>
91             <label for="description">Descripcion:
92             <textarea name="description"><?php echo $myuser_data["description"]?></textarea>
93             <p>
94             <label for="password">Contraseña:
95             <input type="password" name="password">
96             <p>
97             <label for="id">
98             <input type="hidden" name="id" value="<?php echo $myuser_data["user_id"]?>">
99             <label for="submit">
100             <input type="submit" value="Guardar cambios">
101         </form>
102     }

```

```

<?php
session_start();
include_once "connexio.php";
$connex = new mysqli($lloc, $usuari, $pwd, $bbdd);

// pick the username from URL
$username = $_GET['username'];
//pick the info of my session(bugs occurred without that)
if(isset($_SESSION["username"])){
    $myuser = $_SESSION['username'];
    $myquery = "SELECT * FROM usuario WHERE username = '$myuser'";
    $myresult = mysqli_query($connex, $myquery);

    if (!$myresult) {
        die("Error en la consulta: " . mysqli_error($connex));
    }
    $myuser_data = mysqli_fetch_assoc($myresult);
}
// prepare query
$query = "SELECT * FROM usuario WHERE username = '$username'";

$result = mysqli_query($connex, $query);
// check connection
if (!$result) {
    die("Error en la consulta: " . mysqli_error($connex));
}
//in this variable we took all data from the query
$user_data = mysqli_fetch_assoc($result);
?>

```

PerfilModificar.php: Llegamos aquí desde perfil.php si el usuario introducido coincide con el de session, y además envías el formulario de modificación.

Lo principal que se hace aquí es formatear el nombre de imagen introducido para no sobrescribir imágenes subidas por otros usuarios.

También validas que la contraseña sea la correcta, ya que en ese caso no se realizara ninguna modificación.


```

11 //Cogemos todos los datos del usuario
12 $myusername = $_SESSION["username"];
13 $myquery = "SELECT * FROM usuario WHERE username = '$myusername'";
14 $myresult = mysqli_query($connex, $myquery);
15 $myuser_data = mysqli_fetch_assoc($myresult);
16 //Cogemos los datos del formulario y le hacemos un control rapido
17
18 //Este ID es de campo oculto y nos sirve para no joder la base de datos: l y actualizar solo el usuario que toca
19 $id = $_POST["id"];
20 $description = $_POST["description"];
21 //Contraseña sin cifrar introducida en el formulario
22 $password = $_POST["password"];
23 //Contraseña cifrada de la bbdd
24 $bbdd_password = $myuser_data["password"];
25 //Comparamos ambas contraseñas
26 if (password_verify($password, $bbdd_password)) {
27     //Ciframos la nueva contraseña que será la que introduciremos en la bbdd
28     $hashed_password = password_hash($password, PASSWORD_DEFAULT);
29     //Si no hay foto nos quedamos con la de la bbdd, si hay foto se cambia el valor
30     if(isset($_FILES["image"])&& $_FILES["image"]["type"] == 'image/jpeg'){
31         //Aquí tenemos que subir a la carpeta /profilepic la foto para que funcione
32         $imageName = $_FILES["image"]["name"];
33         $imageTemp=$_FILES["image"]["tmp_name"];
34         $new_file_name = rand(1000000,1000000000)."_" .time().".jpg";
35         $image = "profilepic/".$new_file_name;
36         move_uploaded_file($imageTemp, $image);
37     }
38     else{
39         $image = $myuser_data["imagen"];
40     }

```

En ambos casos seremos redirigidos al perfil, pero si decides modificar el nombre de usuario, tienes que ser redirigido mediante el nombre nuevo, no el antiguo, ya que es la referencia que tomamos para hacer las redirecciones a los perfiles.

```

//Cogemos el nombre de usuario y validamos que exista
if(!isset($_POST["name"])){
    $name = $_SESSION["username"];
}
else{
    $name = $_POST["name"];
}
//Creamos y ejecutamos la query
$queryUserUpdate = "UPDATE usuario SET username = '$name',imagen = '$image',password = '$hashed_password',descripti";
$result = mysqli_query($connex, $queryUserUpdate);
$_SESSION["username"] = $name;
//Regresamos al perfil
header("Location: perfil.php?username=$name");
}
else{
    header("Location: perfil.php?username=$myusername");
}
}

```

Ranking.php: Lo principal de este fichero es que printamos una tabla iterando un array que cogemos seleccionando todos los usuarios existentes y ordenandolos por ELO. Aquí es donde indicamos que el usuario en cuestión es el nuestro pindándolo de Azul, o donde indicamos los href correspondientes a cada usuario.

```

<h1 style="text-align: center;"><strong>TOP 50 PLAYERS</strong></h1>
<table class="table">
  <thead class="thead-dark">
    <tr>
      <th>Position</th>
      <th>User</th>
      <th>ELO</th>
    </tr>
  </thead>
  <tbody>
    <?php
    $suma =1;

    if ($listar_usuarios) {
      while($row = mysqli_fetch_array($listar_usuarios, MYSQLI_ASSOC)){
        echo "<tr id='" . $suma . "'><td>" . $suma . "</td>";
        echo "<td><a href='perfil.php?username=" . $row['username'] . "'><span class='glyphicon'></span><img src='";
        echo "<td>" . $row["elo"] . "</td></tr>";
        if (isset($_SESSION['username'])){
          if ($row['username'] == $_SESSION['username']){?>
            <script>
              window.onload = function(){
                var myUser = document.getElementById(<?php echo $suma ?>);
                myUser.style.background = "linear-gradient(90deg, rgba(0,212,255,1) 1%, rgba(73,73,212,1)";
                var myUserColor = myUser.childNodes[1].children[0];
                myUserColor.style.color = "white";
              };
            </script>
            <?php }
          }
          $suma = $suma +1;
        }
      }
    }
  }

```

Aquí es importante poner nombres claros a las variables, ya que tenemos que hacer 2 consultas diferentes a la vez y usarlas para cosas diferentes.

```

<?php
session_start();
include_once "connexio.php";
$connex = new mysqli($lloc, $usuari, $pwd, $bbdd);

// pick the username from URL
if (isset($_SESSION["username"])) {
  $username = $_SESSION['username'];
  // prepare query
  $query = "SELECT * FROM usuario WHERE username = '$username'";
  $result = mysqli_query($connex, $query);
  // check connection
  if (!$result) {
    die("Error en la consulta: " . mysqli_error($connex));
  }
  //in this variable we took all data from the query
  $user_data = mysqli_fetch_assoc($result);
}
//Aquí creem la llista de tots els usuaris ordenats per elo,
$query2 = "SELECT * FROM usuario ORDER BY elo DESC LIMIT 50";
$listar_usuarios = mysqli_query($connex, $query2);

?>

```

<nav>: Practicamente todos nuestros ficheros php y html tienen esta etiqueta. Una buena forma de reutilizar código sería meterla en un fichero php a parte y importarla desde cada fichero que sea necesario, pero por falta de tiempo y algunos errores en el proceso, optamos por no hacerlo.

```
<nav class="navbar navbar-inverse">
  <div class="container-fluid">
    <div class="navbar-header">
      <i class="fas fa-chess-pawn" style='font-size:40px;color:white'></i>
      <a class="navbar-brand" href="index.php">Chess.cum</a>
    </div>
    <ul class="nav navbar-nav" style="margin-left:15px">
      <li><a href="index.php">Home</a></li>
      <li><a href="ranking.php">Ranking</a></li>
    </ul>
    <?php
      if (isset($_SESSION["username"])) {
      ?>
      <ul class="nav navbar-nav navbar-right" style="margin-right: 0px">
        <li><a href="perfil.php?username=?php echo $_SESSION["username"]?>" style="padding: 0px;padding-right:
        <li><a href="logout.php"><span class="glyphicon glyphicon-log-out"></span>Logout</a></li>
      </ul>
    <?php
    }
    else {
    ?>
      <ul class="nav navbar-nav navbar-right">
        <li><a href="signup.html"><span class="glyphicon glyphicon-user"></span>Sign Up</a></li>
        <li><a href="login.html"><span class="glyphicon glyphicon-log-in"></span>Login</a></li>
      </ul>
    <?php
    }
    ?>
  </div>
</nav>
```

Conclusiones:

Al plantear que queríamos desarrollar una aplicación de ajedrez, fuimos advertidos por el profesorado de que programar la lógica del juego es más complejo de lo que parece, y tras la experiencia con este proyecto podemos confirmar que, efectivamente, las apariencias engañan.

También hemos constatado que, si bien sabíamos de antemano que en los ciclos se nos proporciona una base, la cual luego vamos ampliando por nuestra cuenta y en función de lo que pida el mercado laboral, hemos visto que entre el entorno controlado de las clases y lo que es luego la programación en el mundo real hay un salto más grande de lo que habíamos imaginado, incluso teniendo en cuenta que este proyecto sólo es un esbozo comparado con el entorno real de trabajo.

Quizá lo que más nos ha sorprendido no es sólo la envergadura que pueden llegar a tomar los proyectos a poco que empiezas a profundizar en lo que hay que hacer, sino también en la facilidad con que puede tomar más tiempo del que en un principio se había previsto. Nos ha parecido algo a tener en cuenta en el futuro de cara a, por ejemplo, explicar a potenciales clientes el tiempo que puede tomar el desarrollo de sus aplicaciones.

Además, hemos visto que en ocasiones hay que abandonar ciertas ideas o aspectos en pos de una alternativa más eficiente en uno o varios sentidos que pueda conseguir resultados similares. En nuestro caso nos hemos visto en situación de elegir entre una lógica de juego propia pero deficiente e incompleta, o una funcional pero hecha por otros e importada.

Relacionando esto con la realidad del mundo laboral, hemos optado por presentar una lógica de juego funcional que no es nuestra, concretamente las librerías ‘chessboard.js’ y ‘chess.js’, mencionadas en la bibliografía; somos conscientes de que no es para nada deseable en el contexto de este proyecto, pero también entendemos que siempre es preferible presentar algo que funcione, ya que los clientes a quienes podamos atender en el futuro no se van a preocupar por cómo está hecho su programa por dentro, sólo van a querer que haga lo que debe y que lo haga correctamente.

En cualquier caso, las complicaciones que nos han surgido con este proyecto nos han servido para aprender a prepararnos de cara al futuro.

Bibliografía:

Nuestro proyecto: <https://github.com/El69Man/Chess-Online>

<https://chessboardjs.com/examples#1000>

<https://github.com/jhlywa/chess.js/>

<https://chat.openai.com/>