Universidad de Guadalajara

Ingeniería informática

Programación para Internet D03 2024A

Proyecto IA

Profesor: MIICHEL EMANUEL LOPEZ FRANCO

Equipo: Herrera Covarrubias Kevin Shalom, Ramírez Gómez Kevin Ramsés.

INDICE

Introducción.

Nuestro proyecto consiste en un ajedrez con inteligencia artificial (IA), el cual ofrece una experiencia enriquecedora para los aficionados al juego. En primer lugar, hemos desarrollado un sistema de IA que permite a los jugadores enfrentarse a un oponente virtual con un nivel moderado en partidas de ajedrez, asegurando desafíos adaptados a diferentes niveles de habilidad.

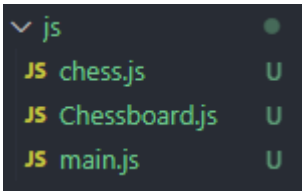
Además de enfrentarse a la IA, otra característica distintiva de nuestro proyecto es la posibilidad de presenciar enfrentamientos entre dos IA. Esto significa que los usuarios tienen la oportunidad de observar cómo dos programas de IA compiten entre sí en el tablero de ajedrez. Este aspecto no solo es fascinante de ver, sino que también brinda una oportunidad invaluable para estudiar tácticas y estrategias de IA de manera práctica y dinámica.

Para facilitar aún más la experiencia de juego, nuestro tablero de ajedrez también ofrece una función de asistencia, que muestra los posibles movimientos de las piezas en el tablero. Esta característica proporciona a los jugadores la oportunidad de explorar y comprender mejor las posibilidades tácticas disponibles en cada situación, al tiempo que les ayuda a mejorar su juego y tomar decisiones más informadas durante la partida.

Explicación General del proyecto.

Como funciona el ajedrez:

El funcionamiento del ajedrez depende de estos tres archivos js:



Empecemos por uno de los tres el chess.js:

El código proporciona una implementación del motor de ajedrez en JavaScript, utilizando una combinación de matrices y objetos para representar el estado del juego y realizar operaciones relacionadas con el ajedrez.

Representación del tablero: Utiliza una matriz bidimensional llamada ar para representar el tablero de ajedrez. Cada elemento de esta matriz representa una casilla del tablero y contiene información sobre la pieza presente en esa casilla.

Objeto Chess: Es el constructor principal del motor de ajedrez. Se utiliza para inicializar el estado del juego, incluyendo la posición de las piezas en el tablero, el turno actual, las opciones de enroque, la captura al paso y otros detalles importantes del juego.

Funciones de movimiento: El código proporciona varias funciones para realizar y validar movimientos de piezas en el tablero. Estas funciones son esenciales para garantizar que los movimientos realizados por los jugadores sigan las reglas del ajedrez.

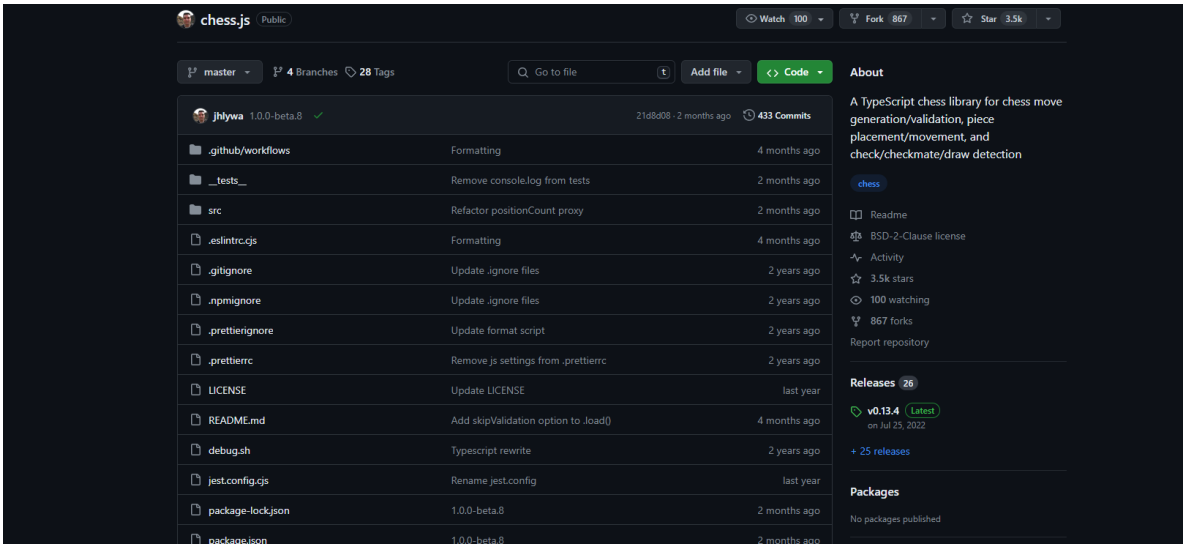
Notación FEN: El motor utiliza la notación FEN (Forsyth–Edwards Notation) para representar el estado del juego. Esto permite establecer la posición inicial del tablero o cargar posiciones específicas mediante una cadena de texto que describe el estado del juego.

Historial de movimientos: El motor mantiene un historial de movimientos realizados en la partida. Esto se logra mediante la utilización de una matriz para almacenar los movimientos en orden cronológico. Esta función es crucial para el análisis retrospectivo del juego y para deshacer movimientos si es necesario.

Otras características: El código también incluye funciones para manejar aspectos específicos del juego, como el enroque, la captura al paso, la promoción de peones, entre otros. Estas funciones garantizan que el motor sea capaz de manejar todas las situaciones posibles que puedan surgir durante una partida de ajedrez.

Nosotros nos basamos en un repositorio que viene ya preparado con una lógica establecida para el ajedrez el cual nos ayudó en gran parte para lograr el funcionamiento del proyecto:

Enlace a ese repositorio de GIT: https://github.com/jhlywa/chess.js#api



Ahora el chessboard.js:

Este código implementa una biblioteca de ajedrez en JavaScript que permite crear y gestionar un tablero de ajedrez interactivo en una página web. La biblioteca proporciona varias funcionalidades clave, como representar el estado del tablero, permitir que los usuarios realicen movimientos de piezas, y manejar diferentes interacciones del usuario, como hacer clic en una casilla o arrastrar y soltar piezas.

Algunos puntos destacados del código incluyen:

Definición de variables y configuración: Al principio del código se definen varias variables y parámetros de configuración, como la representación de las piezas en el tablero, la velocidad de animación y el tema de las piezas.

```
!(function () {
    "use strict";
    var z = window.jQuery,
        F = "abcdefgh".split(""),
        r = 20,
        A = "…",
        W = "1.8.3",
        e = "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR",
        G = pe(e),
        n = 200,
        t = 200,
        o = 60,
        a = 30,
        i = 100,
        H = {};
    function V(e, r, n) {
        function t() {
            (o = 0), a && ((a = !1), s());
        }
        var o = 0,
            a = !1,
            i = [],
            s = function () {
                (o = window.setTimeout(t, r)), e.apply(n, i);
            };
        return function (e) {
            (i = arguments), o ? (a = !0) : s();
        };
    }
    function Z() {
        return "xxxx-xxxx-xxxx-xxxx-xxxx-xxxx-xxxx-xxxx".replace(
            /x/g,
            function (e) {
                return ((16 * Math.random()) | 0).toString(16);
            }
        );
```

Variables Globales y Constantes:

Se definen varias variables globales, como z para jQuery, F para los nombres de las columnas del tablero, r para el tamaño de la cuadrícula del tablero, etc.

A se define como un carácter especial (…) que se utiliza para separar mensajes de error.

W representa la versión de jQuery requerida.

Funciones de Utilidad:

V: Esta función devuelve un nuevo método que, cuando se invoca, espera hasta que no haya llamadas repetidas durante un período de tiempo (r) y luego ejecuta la función dada.

Z: Genera un identificador único.

ee: Sustituye en una cadena los marcadores de posición (por ejemplo, {nombre}) con los valores correspondientes de un objeto.

re, ne, p, c, te, oe, u, ae, ie: Son funciones de validación para varios tipos de datos y formatos.

Funciones de Transformación de Datos:

pe: Convierte una notación de tablero FEN (Notación Forsyth-Edwards) en un objeto que representa la posición de las piezas en el tablero.

ce: Realiza la operación inversa de pe, convirtiendo un objeto de posición de piezas en notación FEN.

_: Realiza una copia profunda de un objeto utilizando JSON para evitar la referencia compartida.

Funciones de Inicialización y Configuración:

fe: Configura y valida las opciones del tablero de ajedrez.

se: Verifica si la versión de jQuery cargada es compatible con la requerida por la librería.

Funciones de Interfaz de Usuario:

q: Dibuja el tablero en la interfaz de usuario, mostrando las piezas y las coordenadas si está habilitado.

T: Coloca las piezas en el tablero según la posición actual.

I: Inicia un arrastre de una pieza en una posición específica.

M: Mueve una pieza arrastrada a una nueva posición durante el arrastre.

N: Suelta una pieza arrastrada en una nueva posición.

B: Elimina una pieza del tablero.

Funciones de Control de Eventos:

D, R: Controlan el inicio del arrastre de una pieza con el ratón o el dedo.

L, U: Controlan el movimiento de arrastre de una pieza con el ratón o el dedo.

j: Previene el comportamiento predeterminado de un evento (por ejemplo, el arrastre de una imagen en un navegador).

X, Q: Controlan la selección de piezas desde las piezas de repuesto con el ratón o el dedo.

Para mas detalles de lo hablado consulte el código del repositorio en git del proyecto:

https://github.com/KevinShalom/Proyect_IA

Ahora pasaremos al main.js:

Este código implementa una versión simple de un motor de ajedrez en JavaScript. Aquí hay una explicación general de las principales funciones y partes del código:

Función evaluateBoard: Esta función evalúa el estado actual del tablero de ajedrez y asigna un valor numérico basado en diferentes criterios, como la presencia de jaque mate, empate, cantidad de piezas capturadas, etc. Esto se utiliza en el algoritmo minimax para evaluar la posición de las piezas y tomar decisiones de movimiento.

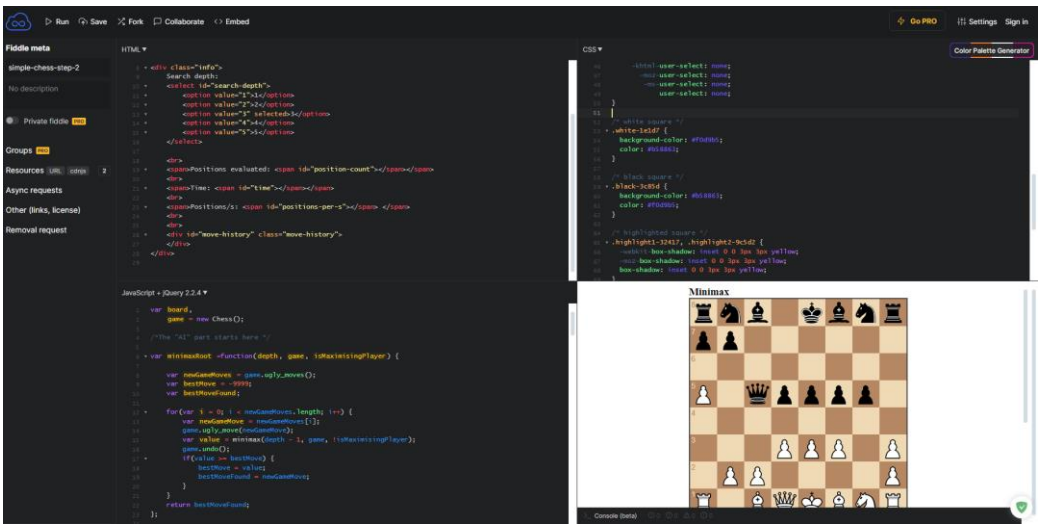
```
function evaluateBoard(e, o, t, a) {
  if (e.in_checkmate()) return o.color === a ? 1e10 : -1e10;
  if (e.in_draw() || e.in_threefold_repetition() || e.in_stalemate()) return 0;
  e.in_check() && (o.color === a ? (t += 50) : (t -= 50));
  var n = [8 - parseInt(o.from[1]), o.from.charCodeAt(0) - "a".charCodeAt(0)],
    i = [8 - parseInt(o.to[1]), o.to.charCodeAt(0) - "a".charCodeAt(0)];
  return (
    t < -1500 && "k" === o.piece && (o.piece = "k_e"),
    "captured" in o &&
      (o.color === a
        ? (t +=
            weights[o.captured] + pstOpponent[o.color][o.captured][i[0]][i[1]])
        : (t -=
            weights[o.captured] + pstSelf[o.color][o.captured][i[0]][i[1]])),
    o.flags.includes("p")
      ? ((o.promotion = "q"),
        o.color === a
          ? ((t -= weights[o.piece] + pstSelf[o.color][o.piece][n[0]][n[1]]),
            (t +=
              weights[o.promotion] + pstSelf[o.color][o.promotion][i[0]][i[1]]))
          : ((t += weights[o.piece] + pstSelf[o.color][o.piece][n[0]][n[1]]),
            (t -=
              weights[o.promotion] +
              pstSelf[o.color][o.promotion][i[0]][i[1]])))
      : o.color !== a
      ? ((t += pstSelf[o.color][o.piece][n[0]][n[1]]),
        (t -= pstSelf[o.color][o.piece][i[0]][i[1]]))
      : ((t -= pstSelf[o.color][o.piece][n[0]][n[1]]),
        (t += pstSelf[o.color][o.piece][i[0]][i[1]])),
    t
  );
```

Función minimax: Implementa el algoritmo minimax, que es un método de búsqueda exhaustiva que se utiliza en juegos de dos jugadores, como el ajedrez, para encontrar el mejor movimiento posible. La función realiza una búsqueda en profundidad del árbol de posibilidades de movimiento, evaluando cada posición mediante la función evaluateBoard. Luego, selecciona el mejor movimiento posible basado en la evaluación y el número de niveles de profundidad especificados.

(Captura de lo hablado en la siguiente hoja)

```
function minimax(e, o, t, a, n, i, r) {
  positionCount++;
  var s,
    l = e.ugly_moves({ verbose: !0 });
  if (
    (l.sort(function (e, o) {
      return 0.5 - Math.random();
    }),
    0 === o || 0 === l.length)
  )
    return [null, i];
  for (
    var u, g = Number.NEGATIVE_INFINITY, c = Number.POSITIVE_INFINITY, h = 0;
    h < l.length;
    h++
  ) {
    s = l[h];
    var d = e.ugly_move(s),
      m = evaluateBoard(e, d, i, r),
      [p, b] = minimax(e, o - 1, t, a, !n, m, r);
    if (
      (e.undo(),
      n
        ? (b > g && ((g = b), (u = d)), b > t && (t = b))
        : (b < c && ((c = b), (u = d)), b < a && (a = b)),
      t >= a)
    )
      break;
  }
  return n ? [u, g] : [u, c];
}
```

Para poder lograr este algoritmo nos basamos en uno de ejemplo el cual es el siguiente:



Enlace: https://jsfiddle.net/k96eoqoq/1/

Mas enlace de información de como se logro el algoritmo para el ajedrez:

https://www.microsiervos.com/archivo/ordenadores/aventura-programa-ajedrez-desde-cero.html

https://www.youtube.com/watch?v=HRAxpCi-RTY

Funciones de control del juego: Se incluyen funciones como checkStatus, que comprueba el estado del juego (jaque mate, empate, etc.) y actualiza la interfaz de usuario en consecuencia. También hay funciones para realizar movimientos (makeBestMove), deshacer movimientos (undo), rehacer movimientos (redo), mostrar pistas (showHint), y restablecer el juego (reset).

```
}
function checkStatus(e) {
  if (game.in_checkmate())
    $("#status").html(`<b>Jaque Mate!</b>, <b>${e}</b> perdio.`);
  else if (game.insufficient_material())
    $("#status").html("Es un <b>empate!</b>");
  else if (game.in_threefold_repetition())
    $("#status").html("Es un <b>empate!</b>");
  else if (game.in_stalemate()) $("#status").html("Es un <b>empate!</b>");
  else {
    if (!game.in_draw())
```

```
function getBestMove(e, o, t) {
  if (((positionCount = 0), "b" === o))
    var a = parseInt($("#search-depth").find(":selected").text());
  else a = parseInt($("#search-depth-white").find(":selected").text());
  var n = new Date().getTime(),
    [i, r] = minimax(
      e,
      a,
      Number.NEGATIVE_INFINITY,
```

```
function undo() {
  var e = game.undo();
  undo_stack.push(e),
    undo_stack.length > STACK_SIZE && undo_stack.shift(),
    board.position(game.fen());
```

```
function redo() {
  game.move(undo_stack.pop()), board.position(game.fen());
}
```

```
function showHint() {
  var e = document.getElementById("showHint");
  if (
    ($board.find("." + squareClass).removeClass("highlight-hint"), e.checked)
  ) {
    var o = getBestMove(game, "w", -globalSum)[0];
    $board.find(".square-" + o.from).addClass("highlight-hint"),
      $board.find(".square-" + o.to).addClass("highlight-hint");
  }
}
```

```
function reset() {
  game.reset(),
    (globalSum = 0),
    $board.find("." + squareClass).removeClass("highlight-white"),
    $board.find("." + squareClass).removeClass("highlight-black"),
    $board.find("." + squareClass).removeClass("highlight-hint"),
    board.position(game.fen()),
    $("#advantageColor").text("Neither side"),
    $("#advantageNumber").text(globalSum),
    timer && (clearTimeout(timer), (timer = null));
}
```

Función compVsComp: La función compVsComp permite que dos computadoras jueguen una contra la otra sin intervención humana. Aquí está el funcionamiento detallado pero breve:

Parámetro e: Indica el color de la computadora que debe realizar el próximo movimiento: "w" para el jugador blanco y "b" para el jugador negro.

Llamadas recursivas: La función se llama a sí misma de manera recursiva. Primero, verifica el estado del juego usando checkStatus. Si el juego no ha terminado, programa una llamada a makeBestMove después de un breve retraso usando setTimeout. Esta llamada simula el movimiento de la computadora y cambia el turno del jugador.

Alternancia de colores: Antes de cada llamada recursiva, el color del jugador se invierte, asegurando que la función alternará entre los jugadores blanco y negro en cada llamada.

Temporizador: Utiliza setTimeout para agregar un pequeño retraso entre los movimientos de las computadoras, lo que hace que el juego sea visualmente más comprensible.

```
function compVsComp(e) {
  checkStatus({ w: "Blanco", b: "Negro" }[e]) ||
    (timer = window.setTimeout(function () {
      makeBestMove(e), (e = "w" === e ? "b" : "w"), compVsComp(e);
    }, 250));
}
```

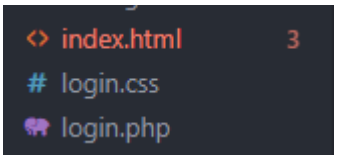
Interfaz de usuario y eventos: Se utilizan eventos de interacción del usuario, como hacer clic en una casilla, arrastrar y soltar piezas, para realizar movimientos en el tablero. También se incluyen botones para iniciar nuevas partidas, deshacer movimientos, rehacer movimientos, etc.

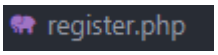
Con eso acabaríamos la explicación breve de los tres archivos js para el funcionamiento lógico del ajedrez.

Inicio de sesión.

Ahora hablemos de un poquito mas cositas por ejemplo el login.

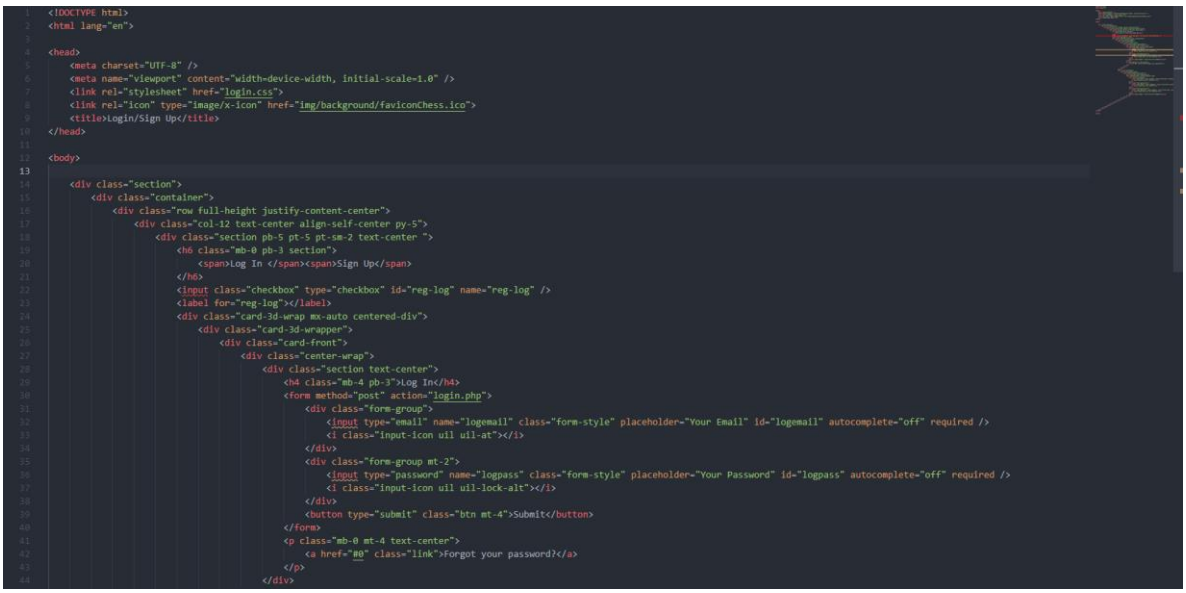
Los archivos que configuramos para el login son los siguientes:





No hay mucho de que hablar de esto pero será breve en el index.html será el encargado de desplegar el sistema del login:

(una captura del código)



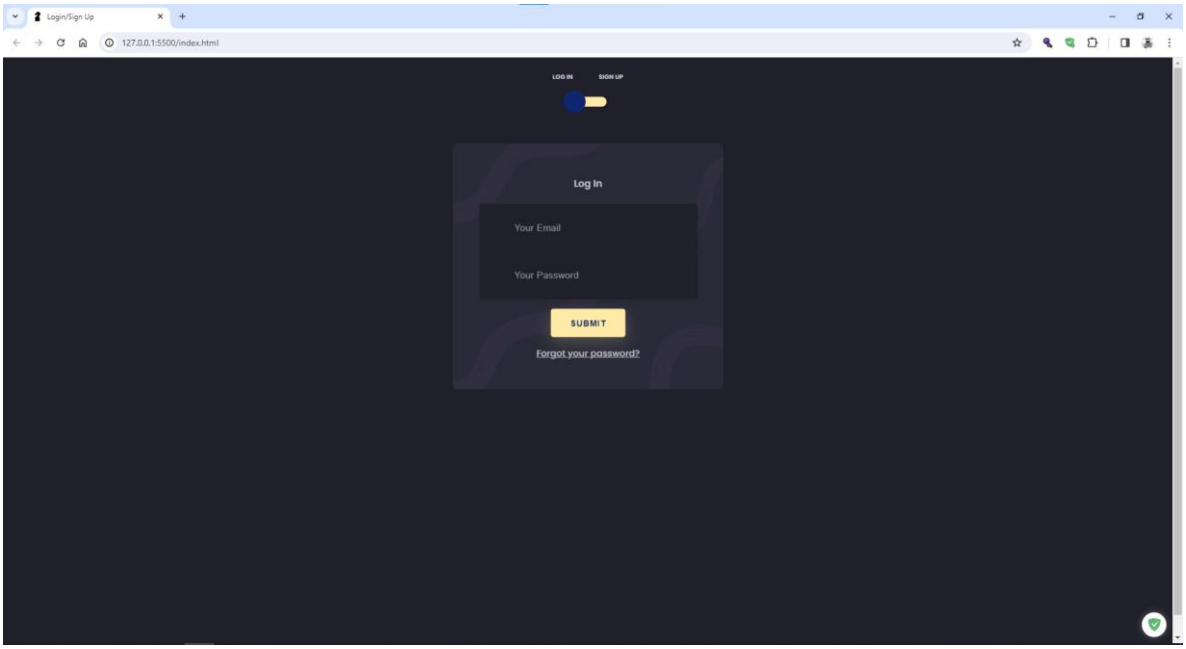
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="login.css">
    <link rel="icon" type="image/x-icon" href="img/background/favIconChess.ico">
    <title>Login/Sign Up</title>
</head>

<body>

    <div class="section">
        <div class="container">
            <div class="row full-height justify-content-center">
                <div class="col-12 text-center align-self-center py-5">
                    <div class="section pb-5 pt-5 pt-sm-2 text-center ">
                        <h6 class="mb-0 pb-3 section">
                            <span>Log In </span><span>Sign Up</span>
                        </h6>
                        <input class="checkbox" type="checkbox" id="reg-log" name="reg-log" />
                        <label for="reg-log"></label>
                        <div class="card-3d-wrap mx-auto centered-div">
                            <div class="card-3d-wrapper">
                                <div class="card-front">
                                    <div class="center-wrap">
                                        <div class="section text-center">
                                            <h4 class="mb-4 pb-3">Log In</h4>
                                            <form method="post" action="login.php">
                                                <div class="form-group">
                                                    <input type="email" name="logemail" class="form-style" placeholder="Your Email" id="logemail" autocomplete="off" required />
                                                    <i class="input-icon uil uil-at"></i>
                                                </div>
                                                <div class="form-group mt-2">
                                                    <input type="password" name="logpass" class="form-style" placeholder="Your Password" id="logpass" autocomplete="off" required />
                                                    <i class="input-icon uil uil-lock-alt"></i>
                                                </div>
                                                <button type="submit" class="btn mt-4">Submit</button>
                                            </form>
                                            <p class="mb-0 mt-4 text-center">
                                                <a href="#0" class="link">Forgot your password?</a>
                                            </p>
                                        </div>
```

A la vez hicimos una conexión a una base de datos para poder almacenar un correo y una contraseña para poder acceder al ajedrez el cual son los .php:

```
$conn = new mysqli($servername, $username, $password, $dbname);

// Verificar la conexión
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Procesar el formulario de inicio de sesión
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $email = $_POST['logemail'];
    $password = $_POST['logpass'];

    // Consulta para verificar las credenciales
    $sql = "SELECT * FROM usuarios WHERE email='$email' AND password='$password'";
    $result = $conn->query($sql);

    if ($result->num_rows == 1) {
        // Inicio de sesión exitoso
        // Redirigir a la página de inicio o hacer alguna otra acción
        header("Location: chessgame.html");
        exit; // Importante: asegúrate de salir del script después de la redirección
    } else {
        // Credenciales incorrectas
        header("Location: index.html");
    }
}

// Cerrar la conexión a la base de datos
$conn->close();
?>
```

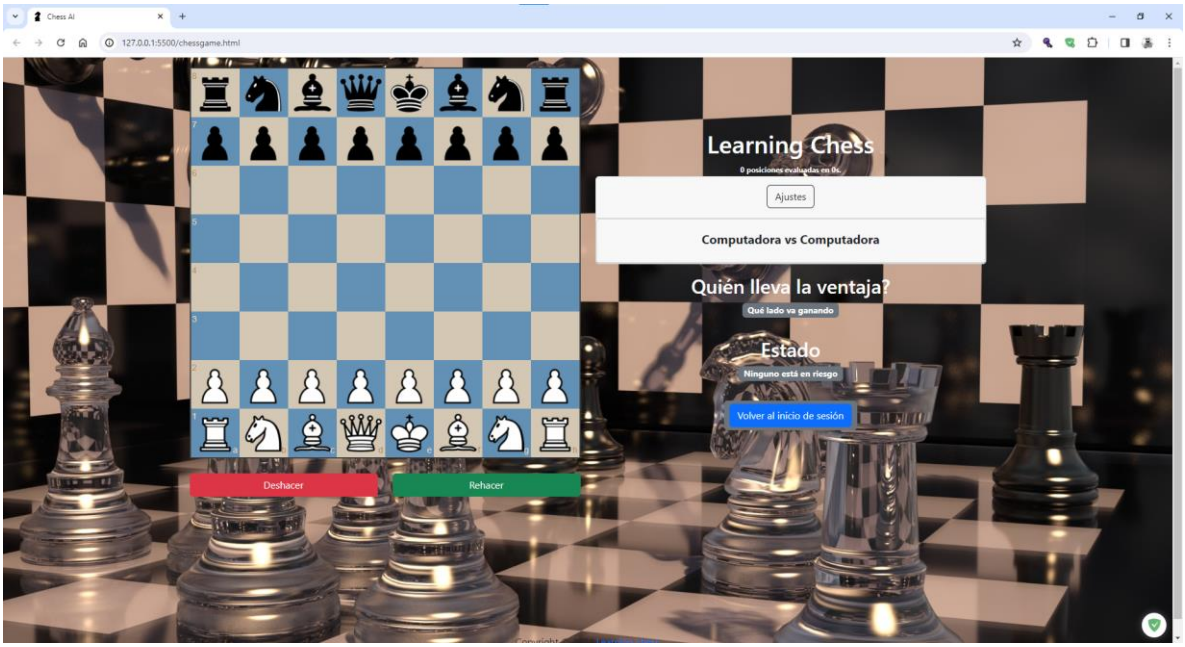
```php
$conn = new mysqli($servername, $username, $password, $dbname);

// Verificar la conexión
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Procesar el formulario de registro
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['logname'];
    $email = $_POST['logemail'];
    $password = $_POST['logpass'];

    // Consulta para insertar el nuevo usuario en la base de datos
    $sql = "INSERT INTO usuarios (username, email, password) VALUES ('$username', '$email', '$passwor

    if ($conn->query($sql) === TRUE) {
        header("Location: index.html");
    } else {
        header("Location: index.html");
    }
}

// Cerrar la conexión a la base de datos
$conn->close();
?>
```

Ahora pasaremos a como se ve la interfaz del proyecto terminada. Primero que nada enseñaremos el login:

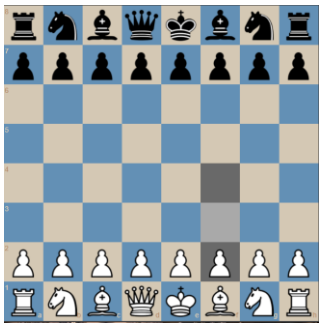


Uno puede tanto registrarse como ya acceder con una cuenta previamente creada como la que esta registrada para hacer las pruebas del login.

Pero ahora pasemos mejor una vez se inicia sesión, vamos a enseñar el juego:



Así se vera la interfaz principal del juego y como dijimos anteriormente tiene asistencia de ayuda para saber como se mueven las piezas:



A la vez que disponemos de unos ajustes adicionales para modificar la dificultad de la inteligencia del bot (IA) del ajedrez, tanto para el lado blanco como negro dependiendo si estas jugando contra el bot (IA) o si fuera vs otro bot (IA).

A la vez que dispone de unas características adicionales:

Referencias:

Jhlywa. (s. f.). *GitHub - jhlywa/chess.js: A TypeScript chess library for chess move generation/validation, piece placement/movement, and check/checkmate/draw detection*. GitHub. https://github.com/jhlywa/chess.js#api

https://jsfiddle.net/k96eoq0q/1/

@Alvy. (s. f.). *La aventura de crear desde cero un programa para jugar al ajedrez, explicada paso a paso*. Microsiervos. https://www.microsiervos.com/archivo/ordenadores/aventura-programa-ajedrez-desde-cero.html

Guerrero, S. (2023, 19 junio). *Crea un tablero de ajedrez con HTML y Grid Layout CSS | Espai*. Blog Escola Espai. https://www.espai.es/blog/2023/06/tablero-ajedrez-html-grid-layout-css/

Quant Dani. (2022, 18 octubre). *Creando una IA que JUEGA al AJEDREZ con Python* [Vídeo]. YouTube. https://www.youtube.com/watch?v=HRAxpCi-RTY