

---

# CLASSIC GAMES

---



Trabajo Fin de Grado  
Curso 2022 - 2023

VICTOR JIMENEZ GOMEZ-AREVALILLO

---

## **OBJETIVO**

La decisión por la que he decidido realizar este proyecto en web viene concebida de mi experiencia y formación en el desarrollo web.

En este proyecto se presenta la creación de una página web que incluye una variedad de juegos clásicos y retro, como Pong, 3 en raya, Piedra, Papel y Tijera, Dados, Blackjack y Adivinar el Número. El objetivo principal es desarrollar una plataforma accesible y fácil de usar para que los usuarios puedan disfrutar de estos juegos de una manera divertida y entretenida.

Además de proporcionar una experiencia de juego fluida, también se prestó atención al diseño y la experiencia de usuario para asegurar una navegación intuitiva y una interfaz atractiva. El desarrollo de esta página web ha requerido una gran creatividad y tiempo.

La memoria tiene como objetivo presentar el proceso de creación de la página web y los juegos incluidos, así como reflexionar sobre los desafíos y lecciones aprendidas a lo largo del proyecto.

## **ABSTRACT**

The decision to undertake this project comes from my experience and training in web development. This project presents the creation of a website that includes a variety of classic and retro games such as Pong, Tic-Tac-Toe, Rock-Paper-Scissors, Dice, Blackjack and Guess the Number. The main objective is to develop an accessible and user-friendly platform for users to enjoy these games in a fun and entertaining way.

In addition to providing a smooth gaming experience, attention was also paid to the design and user experience to ensure intuitive navigation and an attractive interface. The development of this website required a significant amount of creativity and time. The objective of this report is to present the process of creating the website and the included games, as well as reflect on the challenges and lessons learned throughout the project.

## INDICE

### **Índice de usuario**

PROPIEDADES DETALLADA.....	5
OBJETIVOS.....	6
REVISAR.....	7
Servidores.....	7
Lenguajes de programación y Diseño.....	7
IDES.....	8
Otras Herramientas.....	8
ANÁLISIS DEL SISTEMA.....	9
Definición.....	9
Identificación de requisitos.....	9
Casos de uso.....	10
DISEÑO DEL SISTEMA.....	11
Arquitectura.....	11
Diseño y configuración.....	12
Pantallas de la web.....	12
Pantalla inicio.....	14
Pantallas De Los Juegos.....	18
IMPLEMENTACIÓN Y JUEGOS.....	21
Adivina El Número.....	21
Interfaz.....	21
Casos de prueba definidos y resultados.....	22
Escalabilidad.....	25
Blackjack.....	26
Interfaz.....	26
Casos de prueba definidos y resultados.....	28
Escalabilidad.....	35
Dados.....	36
Interfaz.....	36

Casos de prueba definidos y resultados.....	37
Escalabilidad.....	41
Piedra, Papel y Tijera.....	42
Interfaz.....	42
Casos de prueba definidos y resultados.....	43
Escalabilidad.....	49
Pong.....	49
Interfaz.....	50
Casos de prueba definidos y resultados.....	52
Escalabilidad.....	61
Tres en Raya.....	62
Interfaz.....	62
Casos de prueba definidos y resultados.....	63
Escalabilidad.....	69
Elementos de seguridad.....	70
GESTIÓN DEL PROYECTO.....	70
Ciclo de vida.....	70
Planificación inicial.....	70
Planificación final.....	71
CONCLUSIONES.....	71
Valoración del proyecto.....	71
Posibles mejoras futuras.....	72
BIBLIOGRAFÍA.....	72

## 1. PROUESTA DETALLADA

En un principio, consideré varias opciones para mi proyecto final de grado, incluyendo la creación de diferentes aplicaciones. Sin embargo, después de evaluar varios factores decidí enfocarme en la creación de una página web que incluyera una variedad de juegos clásicos y retro, que actualmente ya no podemos encontrar tan fácilmente en el sistema operativo Windows.

La principal motivación detrás de esta decisión fue la oportunidad de poner en práctica mis habilidades aprendidas en desarrollo web y mostrar mi capacidad para crear una plataforma completa y funcional. Además, me pareció interesante la idea de crear una experiencia de juego en línea que fuera accesible y fácil de usar para los usuarios.

También consideré que en el mercado hay muchas aplicaciones de juegos, pero pocas páginas web que ofrezcan una variedad de juegos clásicos y retro de manera accesible, por eso me pareció una buena oportunidad para llenar esa necesidad y ofrecer algo diferente.

Todos los juegos están disponibles en una página principal desde la cual el usuario puede acceder a cada juego individualmente.

Los juegos de Classic Games:

1. Juego de **Adivinar el Número**: Un juego en el que el usuario tiene que adivinar un número entre 1 y 100, recibiendo indicaciones de si el número es alto o bajo.
2. Juego de **Blackjack**: Un juego de cartas en el que el objetivo es obtener una mano con un puntaje lo más cercano posible a 21 en comparación con la computadora.
3. Juego de **Dados**: Un juego en el que el usuario elige si quiere jugar a "mayor" o "menor" y se enfrenta a la computadora para ver quién saca el número más cercano al elegido.
4. Juego de **Piedra, Papel o Tijera**: Un juego en el que el usuario elige entre piedra, papel o tijera mediante botones y se enfrenta a la computadora.
5. Juego de **Pong** : Un juego clásico en el que se enfrenta al usuario contra la computadora en un juego de tenis de mesa.
6. Juego de **3 en Raya** : Un juego clásico en el que se enfrenta al usuario contra la computadora en un juego de lógica y estrategia.

En cuanto al desarrollo técnico, la página web de Classic Games y retro fue construida utilizando HTML, CSS y JavaScript como lenguajes principales.

HTML se utilizó para estructurar el contenido de la página, CSS se utilizó para dar estilo visual y JavaScript se utilizó para agregar interactividad y lógica a los juegos. Estos lenguajes y herramientas son los que se han ido aprendiendo y/o conociendo durante el Grado Superior, y han sido esenciales para llevar a cabo este proyecto.

Los desafíos técnicos encontrados durante el desarrollo incluyeron la creación de un sistema de puntuación para cada juego, mensajes de victoria y derrota para así fomentar mas competitividad dentro del juego .

En resumen, mi decisión de enfocarme en la creación de una página web de juegos clásicos y retro se basó en mi infancia jugando a los juegos tradicionales del sistema operativo Windows y también en mi experiencia y formación en desarrollo web, así como en la oportunidad de poner en práctica mis habilidades y ofrecer una experiencia de juego en línea accesible y diferente al mercado.

## **2. OBJETIVOS**

La finalidad del proyecto es proporcionar una plataforma accesible y fácil de usar para los usuarios para disfrutar de una variedad de juegos clásicos y retro, también buscábamos esa nostalgia al volver a jugar a esos juegos que jugaba la gente en la época de los primeros ordenadores .

En cuanto a la implementación en entorno laboral, este proyecto podría ser utilizado en empresas de juegos o entretenimiento, ya que proporciona una plataforma de juegos divertidos y entretenidos para los usuarios. También podría ser utilizado en empresas que buscan proporcionar una experiencia de juego en línea para sus empleados o clientes, como una forma de relajación y distracción. También podría ser utilizado en entornos educativos, especialmente en colegios para desarrollar habilidades en los niños mediante juegos fáciles y divertidos. Los juegos incluidos en esta plataforma, como 3 en Raya, Piedra, Papel y Tijera, y Adivinar el Número son juegos simples que pueden ayudar a los niños a desarrollar sus habilidades lógicas y de razonamiento. Además, el juego de Blackjack podría ser utilizado para enseñar a los niños estrategias básicas y cómo tomar decisiones bajo presión. En general, este proyecto podría ser utilizado como una herramienta educativa para ayudar a los niños a desarrollar habilidades importantes mientras se divierten jugando juegos.

En cuanto a la escalabilidad, este proyecto tiene un gran potencial de escalabilidad. Se podrían agregar más juegos y funcionalidades a la plataforma, como la posibilidad de jugar contra otros jugadores en línea, o la posibilidad de personalizar la experiencia de juego mediante la selección de diferentes niveles de dificultad. También se podría implementar un sistema de puntuación y clasificación para que los usuarios puedan comparar sus habilidades con las de otros jugadores.

### 3. REVISAR

#### 3.1. Servidores

Intenté crear el sitio web utilizando diferentes opciones

Primero con un dominio gratuito con Freenom .ml, .cf o .tk y un hosting gratuito como Byethost, pero tuve problemas con la conexión DNS.

Luego, compré un dominio en Hostalia con la extensión .es y lo subí a un hosting de pago como Hostinger, pero el explorador de archivos no reconoció mi página principal Index y por mas que lo intento me carga la pagina o como una predeterminada de hostinger o de hostalia.

También intenté utilizar WordPress y Elementor, a través de un elementor canvas e insertar el código y lanzar por páginas la web, pero esta opción la termine descartando.

Finalmente, utilicé 000webhost, una sub-página gratuita de Hostinger, donde logré mostrar correctamente mi página principal y navegar por ella. Esta fue mi opción final.

Classic Games: <https://classicgamesbyvictor.000webhostapp.com/Index.html>

#### 3.2. Lenguajes de programación y Diseño

1. **HTML 5** fue utilizado como lenguaje estructurar el contenido de la página web. Se utilizaron etiquetas HTML estándar como head, div, etc. para estructurar el contenido de la página web y proporcionar una estructura lógica para los juegos y el diseño. Además, se utilizaron etiquetas específicas para los juegos como canvas para el juego de Pong, y tablas para en el juego de 3 en Raya.
2. **JavaScript** fue utilizado para proporcionar funcionalidad interactiva a los juegos. Se utilizó JavaScript para la lógica de juego, como la detección de colisiones en el juego de Pong, o la selección de cartas en el juego de Blackjack.
3. **CSS** fue utilizado para proporcionar estilos visuales a la página web. Se utilizaron reglas CSS para establecer el diseño de la página web, como la posición de los elementos, los colores, las fuentes, etc. Además, se utilizaron medidas para adaptar el diseño a diferentes tamaños de pantalla y dispositivos, para garantizar una experiencia de usuario consistente en todos los dispositivos. Además, se utilizaron animaciones y transiciones con CSS para proporcionar una experiencia de juego más fluida y atractiva.

En resumen, HTML, JavaScript y CSS son los tres pilares fundamentales del desarrollo web, y en este proyecto se utilizaron estos lenguajes de manera conjunta para crear una página web de juegos retro que proporciona una experiencia de usuario atractiva y fácil de usar. HTML se utilizó para estructurar el contenido de la página, JavaScript para proporcionar funcionalidad interactiva y lógica de juego, y CSS para proporcionar estilos visuales y adaptabilidad a diferentes dispositivos.

### **3.3. IDES**

En el proyecto probe a utilizar el entorno de desarrollo Visual Studio debido a la utilidad de este mismo IDE en las prácticas laborales. Sin embargo, también considere el uso de **Sublime Text** debido a la familiaridad y comodidad con este entorno y sus características adicionales como los atajos y plugins.

En lugar de limitarme al primer entorno de desarrollo, tome la decisión de utilizar la herramienta que mejor se adapta a las necesidades del proyecto y a mi. Esto se consideró para maximizar la eficiencia y calidad del desarrollo.

Además, al elegir esta herramienta logre una mayor productividad y una mejor comprensión del proyecto. Esto permitió un mejor desempeño en el desarrollo y una mejor calidad en el resultado final.

### **3.4. Otras Herramientas**

Diseñadores gráficos, editores de video, motores de video juegos ...

En este proyecto, se ha utilizado el software de diseño gráfico **Photoshop CS6** para crear un logo y fondos de pantalla personalizados.

A través de mis conocimientos en diseño gráfico, he utilizado las herramientas y funciones de Photoshop para crear diseños atractivos y profesionales. He utilizado técnicas de edición de imágenes para crear un logo único, y he creado una variedad de fondos de pantalla con diferentes diseños y estilos para adaptarse a diferentes necesidades y preferencias.

## 4. ANÁLISIS DEL SISTEMA

### 4.1. Definición

El objetivo principal de este proyecto es crear una experiencia de juego divertida y entretenida para los usuarios, al proporcionar una plataforma de juegos clásicos y retro en línea. A través del desarrollo de esta plataforma, buscamos revivir esa nostalgia de los juegos de antaño, aquellos que jugaban en los primeros ordenadores y consolas.

La utilidad de este proyecto radica en la posibilidad de proporcionar una forma divertida y accesible de jugar a estos juegos clásicos, ya sea para pasar el tiempo libre o para desarrollar habilidades de estrategia y pensamiento lógico a través del juego. Además, también se ha prestado atención al diseño y la experiencia de usuario para asegurar una navegación intuitiva y una interfaz atractiva.

### 4.2. Identificación de requisitos

FUNCIONALES:	
Requisitos del sistema	Descripción
Compatibilidad con navegadores	La página web debe ser compatible con los navegadores web más populares (Chrome, Firefox, Safari, Edge, etc.)
Interfaz de usuario	La interfaz de usuario debe ser fácil de usar e intuitiva para el usuario
Acceso a juegos	La página web debe ofrecer acceso a una variedad de juegos clásicos y retro
Compatibilidad con dispositivos	La página web debe ser compatible con diferentes dispositivos (escritorio, tablet, móvil)
Interacción con el juego	El usuario debe poder interaccionar con el juego (manejar, velocidad, etc.)
Jugar contra la CPU	La página web debe permitir jugar contra la CPU
Puntuaciones y estadísticas	La página web debe ofrecer la posibilidad de obtener puntuaciones y estadísticas del juego

NO FUNCIONALES:	
Requisitos del sistema	Descripción
Tiempo de carga	La página web debe cargar rápidamente para una mejor experiencia del usuario
Diseño responsive	La página web debe ser diseño responsive y adaptable a diferentes resoluciones de pantalla
Escalabilidad	La página web debe ser escalable para soportar un gran número de jugadores
Accesibilidad	La web debe ser accesible para personas con discapacidades
Seguridad	La página web debe ser segura para proteger la información del usuario

### 4.3. Casos de uso

A nivel de **usuario**:

Caso de uso	Descripción	Actores
Acceder a la página web	El usuario ingresa a la página web y se presenta con una pantalla de inicio con una lista de juegos disponibles.	Usuario
Seleccionar un juego	El usuario selecciona un juego de la lista y se inicia la pantalla de juego correspondiente.	Usuario
Jugar al juego	El usuario juega al juego seleccionado y se presentan las instrucciones para jugar.	Usuario
Puntuación/estadísticas	El usuario puede ver su puntuación y estadísticas de juego en tiempo real.	Usuario
Finalizar juego	El usuario finaliza el juego y se presenta la opción de volver a jugar..	Usuario
Volver a inicio	El usuario acaba y se redirige a la pantalla de inicio.	Usuario

A nivel de **administradores**:

Caso de uso	Descripción
Administrador usuarios	El administrador puede crear, editar y eliminar usuarios en la plataforma en un futuro.
Administrador juegos	El administrador puede crear, editar y eliminar juegos en la plataforma.
Administrador contenido	El administrador puede crear, editar y eliminar contenido en la plataforma, como imágenes o textos.
Revisar estadísticas	El administrador puede revisar estadísticas sobre el uso de la plataforma, como el número de usuarios registrados o el número de partidas jugadas.

## 5. DISEÑO DEL SISTEMA

### 5.1. Arquitectura

La arquitectura utilizada en este proyecto es la arquitectura cliente-servidor. La parte del cliente está compuesta por el navegador web del usuario, que se encarga de mostrar la interfaz gráfica de la página web y recibir las interacciones del usuario. La parte del servidor está compuesta por un servidor web, que se encarga de procesar las solicitudes del cliente y enviar las respuestas adecuadas.

En cuanto al hosting, se ha utilizado un servicio de alojamiento web para publicar la página en Internet, de esta manera los usuarios pueden acceder a ella desde cualquier lugar con conexión a Internet..

## 5.2. Diseño y configuración

### 5.2.1. Pantallas de la web

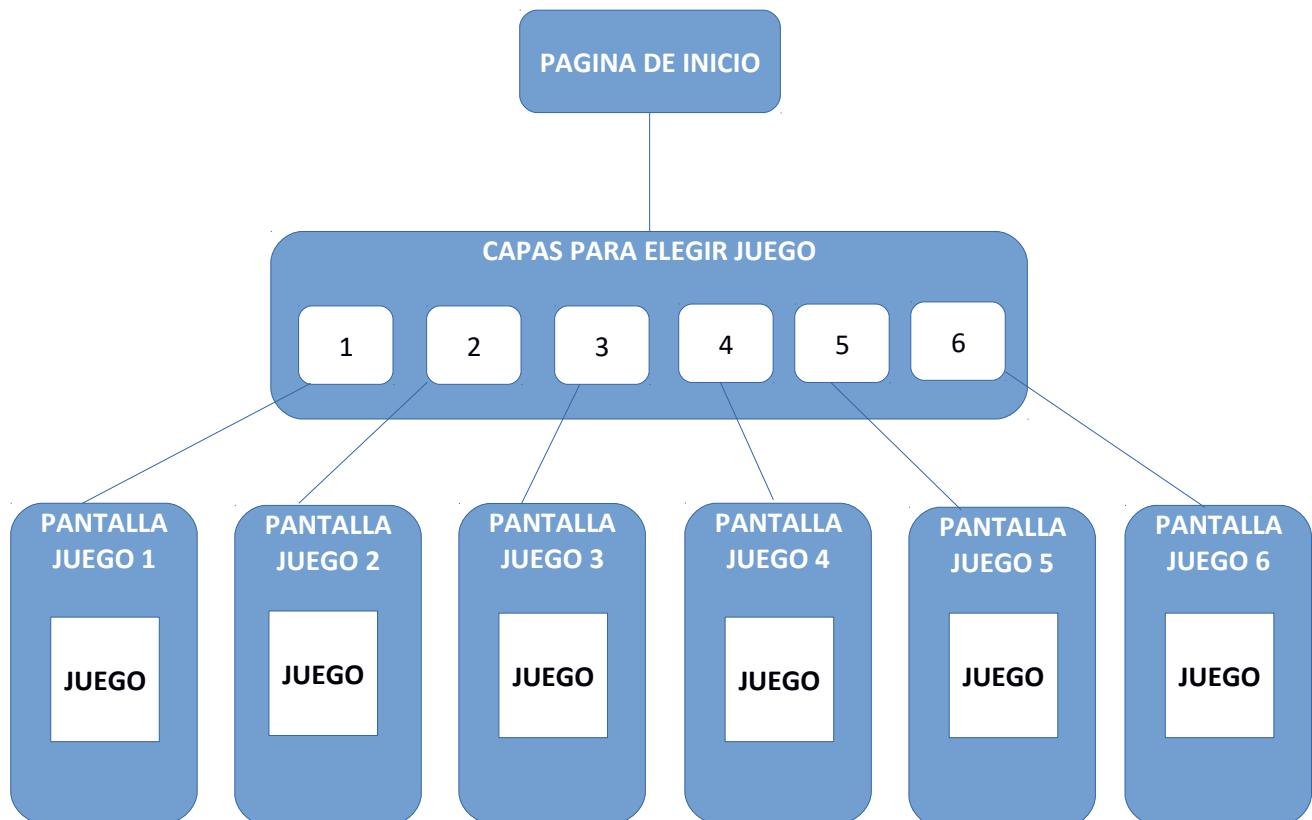
Los clientes serían los navegadores web compatibles, como Google Chrome, Firefox, Safari, entre otros. Estos clientes permitirán a los usuarios interactuar con la aplicación a través de su navegador web.

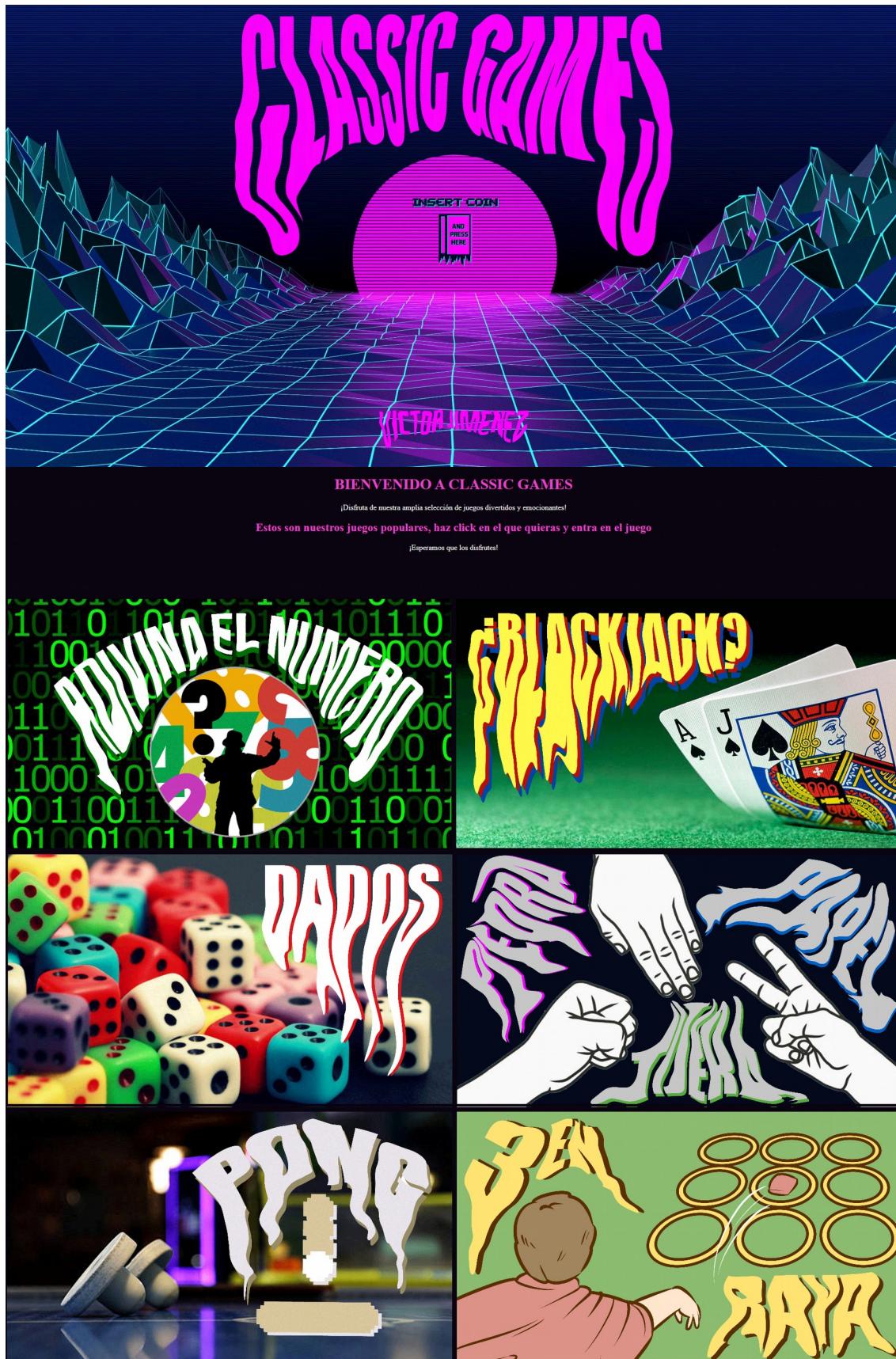
En cuanto al diseño de pantallas, se ha optado por un diseño sencillo y atractivo, con una interfaz de usuario intuitiva y fácil de navegar. Las pantallas incluyen una página de inicio con acceso a todos los juegos, en esta pantalla de inicio hay una selección de juegos que te lanza a una pantalla del juego con tan solo pulsar la opción.

La pantalla del juego, obtiene el juego, las estadísticas y marcadores. La arquitectura utilizada es la arquitectura cliente-servidor.

El lado del cliente se encarga de proporcionar una interfaz gráfica de usuario para interactuar con los juegos, mientras que el lado del servidor se encarga de procesar las solicitudes y responder con los resultados del juego. El proyecto se aloja en un servidor web y utiliza JavaScript para procesar y responder a las solicitudes del usuario.

Esquema de pantallas de la web:





### **5.2.2. Pantalla inicio**



La página de inicio del proyecto cuenta con un fondo de pantalla diseñado específicamente para crear un ambiente divertido y entretenido.

En la parte superior de la página se encuentra un botón ( Insert Coin )que permite al usuario acceder rápidamente a la pantalla de inicio de los juegos, donde se encuentran disponibles una serie de juegos. La pantalla de inicio incluye una descripción, así como las opciones para entrar a la experiencia del juego.



La página de inicio de la web cuenta con un botón gráfico elaborado con el objetivo de evocar una sensación nostálgica en los usuarios.

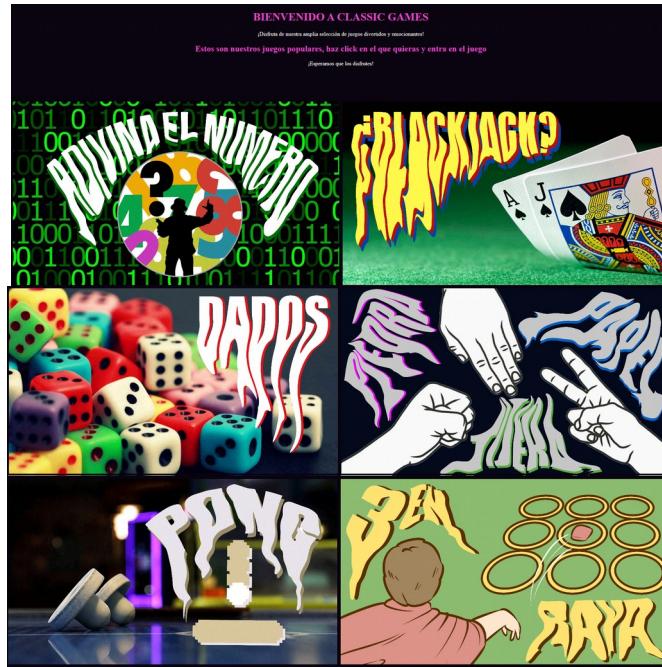
```
.button {  
    border: none;  
    position: fixed;  
    top: 50%;  
    left: 50%;  
    transform: translate(-50%, -50%);  
    background-color: rgba(11, 4, 17, 0);  
    color: #EA44CF;  
    padding: 35px 0px;  
    text-align: center;  
    text-decoration: none;  
    display: inline-block;  
    font-size: 16px;  
}
```

Utilizamos herramientas de diseño gráfico como Photoshop Cs6 para crear un fondo de pantalla que simula una máquina recreativa con un botón de insertar moneda.

Después implementamos un botón transparente en HTML que al ser presionado, permite al usuario bajar directamente a la pantalla de inicio, donde se encuentran las descripciones y opciones de los juegos disponibles. Esto hace la intención de aumentar la inmersión en la experiencia de juego



En la página principal, se ha implementado un diseño que consiste en una capa principal que sirve como introducción visual al sitio web. En esta capa, se presenta una breve descripción de la página y disponibles.



En la página principal, se ha implementado un diseño que consiste en una capa principal que sirve como introducción visual al sitio web. En esta capa, se presenta una breve descripción de la pagina y disponibles.

```
<div class="bigpage">
    <div class="layerwelcome">
        <h1> BIENVENIDO A CLASSIC GAMES </h1>
        <p> ¡Disfruta de nuestra amplia selección de juegos divertidos y emocionantes! </p>
    </div>
    <div class="layerwelcome">
        <h2> Estos son nuestros juegos populares, haz click en el que quieras y entra en el juego </h2>
        <p> ¡Esperamos que los disfrutes! </p>
    </div>

    <div class="games">
        <div class="container">
            <div class="layer adivina" onclick="window.location.href='adivinaelnumero.html'"></div>
            <div class="layer blackjack" onclick="window.location.href='blackjack.html'"></div>
        </div>
        <div class="container">
            <div class="layer dados" onclick="window.location.href='dados.html'"></div>
            <div class="layer piedrapapeltiljera" onclick="window.location.href='piedrapapeltiljera.html'"></div>
        </div>
        <div class="container">
            <div class="layer pong" onclick="window.location.href='juegopong.html'"></div>
            <div class="layer tresenraya" onclick="window.location.href='3enraya.html'"></div>
        </div>
    </div>
</div>
```

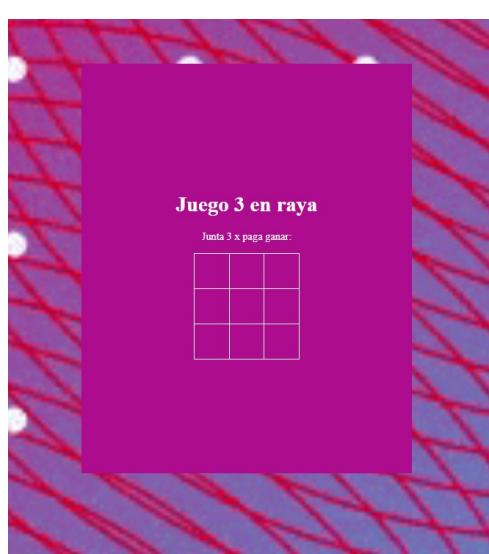
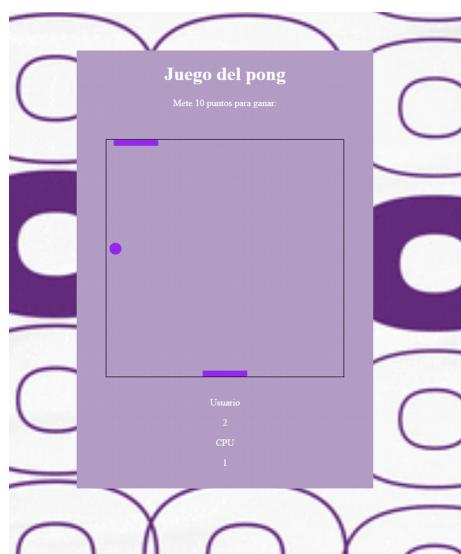
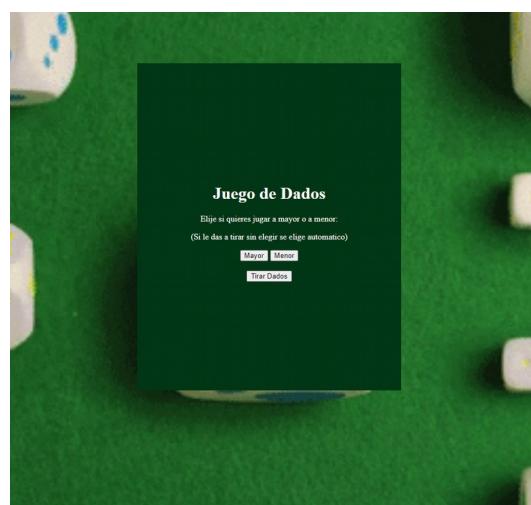
Debajo de esta capa principal, se encuentran seis capas adicionales, cada una con un fondo personalizado que presenta una portada y vista previa de cada juego.

Al hacer clic en cada capa, el usuario será dirigido a la página del juego correspondiente, donde podrá interactuar y jugar.



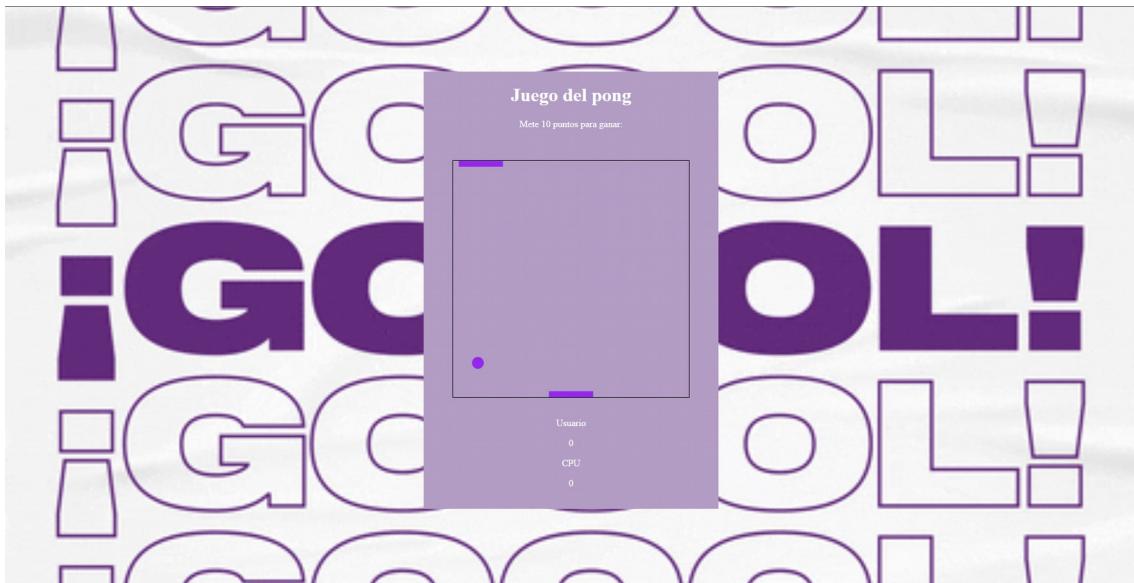
Este diseño no solo es atractivo visualmente, sino que también facilita la navegación y accesibilidad de los juegos disponibles en la plataforma.

### 5.3. Pantallas De Los Juegos



Cada juego cuenta con su propia pantalla, diseñada para crear una experiencia única y personalizada para el usuario.

El fondo de cada pantalla está compuesto por un Gift específico para cada juego, que se ajusta perfectamente al ambiente y la temática del juego en cuestión. Esto permite al usuario sentirse sumergido en el juego desde el primer momento en que lo abre.



Además de los fondos personalizados, cada pantalla cuenta con una capa central que se encuentra en el centro de la pantalla.

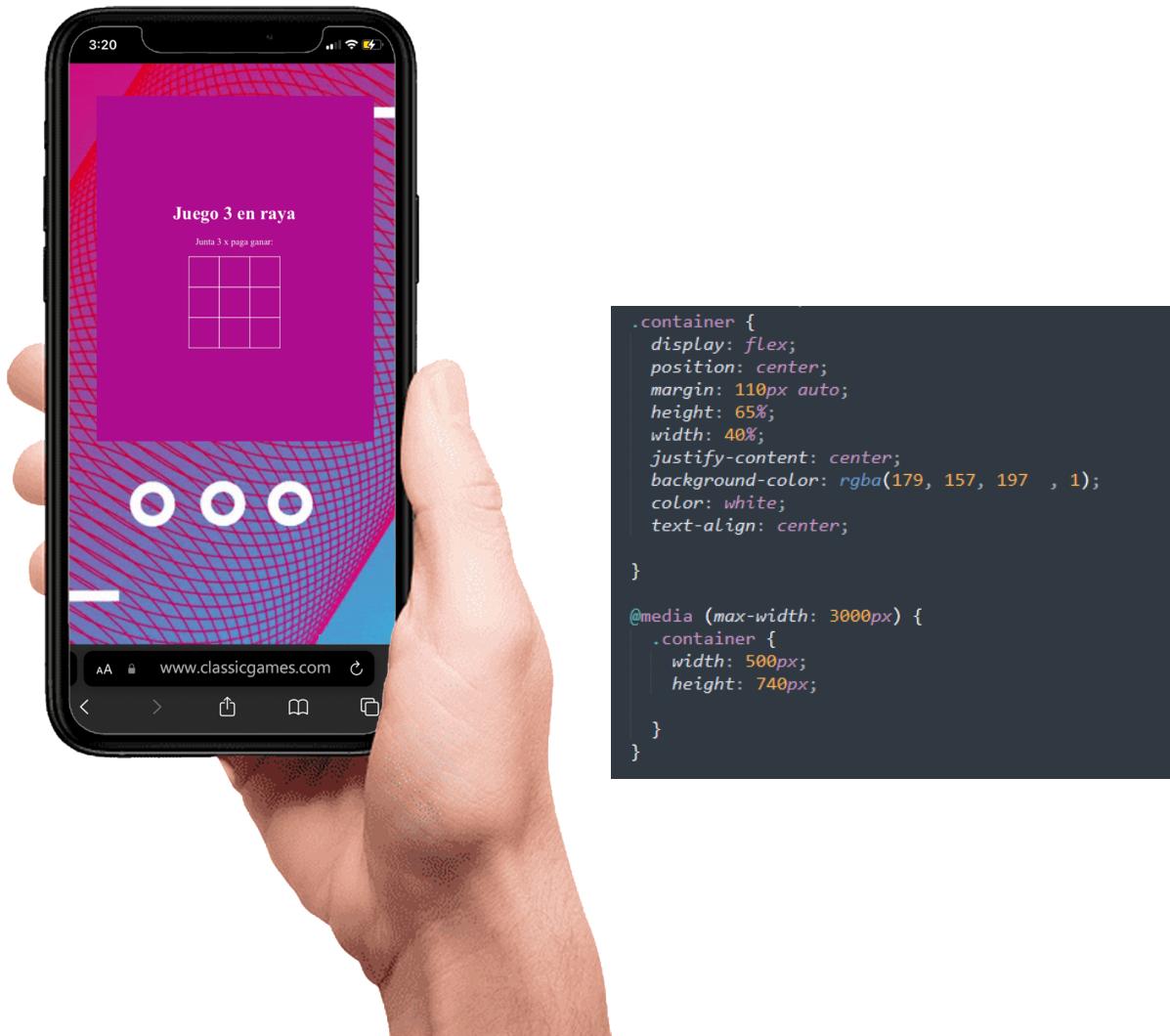
Esta capa está diseñada para mostrar el juego en sí, y se ajusta de manera que se vea perfectamente centrada en la pantalla. El color de esta capa es específico para cada juego y se elige de manera que complementa el fondo de pantalla, creando un ambiente visualmente atractivo y coherente.

La combinación de estos elementos permite al usuario sentirse completamente sumergido en el juego desde el primer momento en que lo abre. El usuario se sentirá como si estuviera dentro del juego, y no sólo jugando en una pantalla.

Los fondos y capas personalizadas crean un ambiente atractivo y coherente, lo que aumenta la inmersión y la experiencia de juego en general.

En el esfuerzo por ofrecer una experiencia de juego óptima a los usuarios, he ajustado cada pantalla y juego de manera que sean totalmente adaptables a diferentes tipos de pantalla.

Esto significa que los juegos se verá increíblemente bien en cualquier dispositivo, ya sea un teléfono móvil, una tableta o una computadora de escritorio.



He adaptado las proporciones de pantalla de cada dispositivo , lo que garantiza que el juego se vea nítido y claro en cualquier pantalla.

## 6. IMPLEMENTACIÓN Y JUEGOS

### 6.1. Adivina El Numero



#### 6.1.1. Interfaz

El juego de Adivinar el Número es una emocionante y desafiante experiencia de juego que pone a prueba la habilidad del usuario para adivinar un número específico entre el rango de 1 y 100.

El objetivo del juego es adivinar el número correcto en el menor número de intentos posible, recibiendo indicaciones de si el número es demasiado alto o demasiado bajo en cada intento.

Este juego ofrece una experiencia única ya que el usuario debe usar su habilidad lógica y su capacidad para analizar y resolver problemas para adivinar el número correcto.

Con una interfaz fácil de usar y controles intuitivos, este juego es perfecto tanto para jugadores experimentados como para aquellos que están empezando a jugar.

El juego cuenta con una interfaz de usuario clara y fácil de usar, este se encuentra en una capa central en la pantalla, donde esta el título del juego y una pequeña descripción del mismo. Esto permite al usuario comprender rápidamente qué es lo que se espera de él y cómo funciona el juego.



En la parte inferior de la pantalla se encuentra un campo de entrada donde el usuario puede introducir su número de adivinanza, y un botón para ejecutar el juego. El campo de entrada es fácil de usar y se ajusta automáticamente a cualquier número introducido, lo que permite al usuario jugar de manera rápida y sencilla.

El botón de ejecución del juego está bien destacado y es fácil de encontrar, lo que permite al usuario iniciar el juego con facilidad.

```
<div class="container">
  <div id="layer">
    <h1>Juego de adivinanza de números</h1>
    <p>Elije un número entre 1 y 100 y trata de adivinar cuál es:</p>
    <input type="text" id="guess"/>
    <button onclick="checkGuess()">Comprobar adivinanza</button>
    <h3 id="result"></h3>
  </div>
</div>
```

### 6.1.2. Casos de prueba definidos y resultados

Para asegurarnos de que el juego funcione correctamente y proporcione una experiencia de juego satisfactoria, hemos definido varios casos de prueba específicos.

1. Uno de los casos de prueba consiste en comprobar si el campo de entrada del número de adivinanza está funcionando correctamente y aceptando sólo números válidos dentro del rango de 1 a 100.

## CASO DE PRUEBA

Validación de los datos del campo de entrada. Para ello, introducimos un valor no numérico, como una letra o un símbolo. Esto nos permite evaluar cómo el juego maneja y reporta estos errores, y asegurarnos de que el usuario recibe una respuesta adecuada y comprensible.

### Juego de adivinanza de números

Elije un número entre 1 y 100 y trata de adivinar cuál es:

asd

Comprobar adivinanza

## RESULTADO

### Juego de adivinanza de números

Elije un número entre 1 y 100 y trata de adivinar cuál es:

asd

Comprobar adivinanza

**Por favor, introduce un número válido**

El juego a sido programado con una línea específica de código que permite detectar y manejar errores en el campo de entrada del número de adivinanza.

Cuando el usuario introduce un valor que no es un número, el juego detecta automáticamente este error y emite un mensaje de error específico.

```
if (isNaN(guess)) {  
    document.getElementById("result").innerHTML = "Por favor, introduce un número válido";  
    return;  
}
```

2. También comprobamos si el botón de ejecución del juego está funcionando correctamente y si proporciona la respuesta adecuada al usuario, ya sea "demasiado alto" o "demasiado bajo" dependiendo de la adivinanza del usuario.

### CASO DE PRUEBA

#### Juego de adivinanza de números

Elije un número entre 1 y 100 y trata de adivinar cuál es:

 Comprobar adivinanza

#### Juego de adivinanza de números

Elije un número entre 1 y 100 y trata de adivinar cuál es:

 Comprobar adivinanza

Validación de los mensajes de adivinanza alta o baja. Para ello, introducimos el valor numérico mas alto y mas bajo. Esto nos permite evaluar cómo el juego maneja y reporta estos errores, y asegurarnos de que el usuario recibe una respuesta adecuada y comprensible.

### RESULTADO

#### Juego de adivinanza de números

Elije un número entre 1 y 100 y trata de adivinar cuál es:

 Comprobar adivinanza

Tu adivinanza es demasiado alta

#### Juego de adivinanza de números

Elije un número entre 1 y 100 y trata de adivinar cuál es:

 Comprobar adivinanza

Tu adivinanza es demasiado baja

El sistema de generación de números aleatorios se utiliza para elegir el número objetivo que el usuario debe adivinar.

```
var correctNumber = Math.floor(Math.random() * 100) + 1;
```

Introduce número de adivinanza, el juego lo compara con el número objetivo y emite un mensaje, "demasiado alto" o "demasiado bajo", dependiendo de si el número de adivinanza es mayor o menor que el número objetivo.

```
if (guess > correctNumber) {  
    document.getElementById("result").innerHTML = "Tu adivinanza es demasiado alta";  
} else if (guess < correctNumber) {  
    document.getElementById("result").innerHTML = "Tu adivinanza es demasiado baja";  
}
```

- 
3. Otro caso de prueba es comprobar si la interfaz de usuario está funcionando correctamente, si es fácil de usar e intuitiva para el usuario y si el usuario puede ganar el juego y no salga el mismo numero siempre.

### CASO DE PRUEBA

Validación del mensaje de victoria. Para ello, introducimos valores hasta acertar. Esto nos permite evaluar cómo el juego maneja y reporta estos errores, y asegurarnos de que el usuario recibe una respuesta adecuada y comprensible.

### RESULTADO

The image contains two side-by-side screenshots of a web-based number guessing game. Both screenshots have a black header with the title 'Juego de adivinanza de números' and a light blue footer bar at the bottom.

**Screenshot 1 (Left):** The input field contains '70'. Below it, a red box highlights the error message 'Tu adivinanza es demasiado baja'. At the bottom, a green button displays the success message '¡Felicitaciones, adivinaste el número!'. The footer bar is light blue.

**Screenshot 2 (Right):** The input field contains '90'. Below it, a red box highlights the error message 'Tu adivinanza es demasiado alta'. At the bottom, a green button displays the success message '¡Felicitaciones, adivinaste el número!'. The footer bar is light blue.

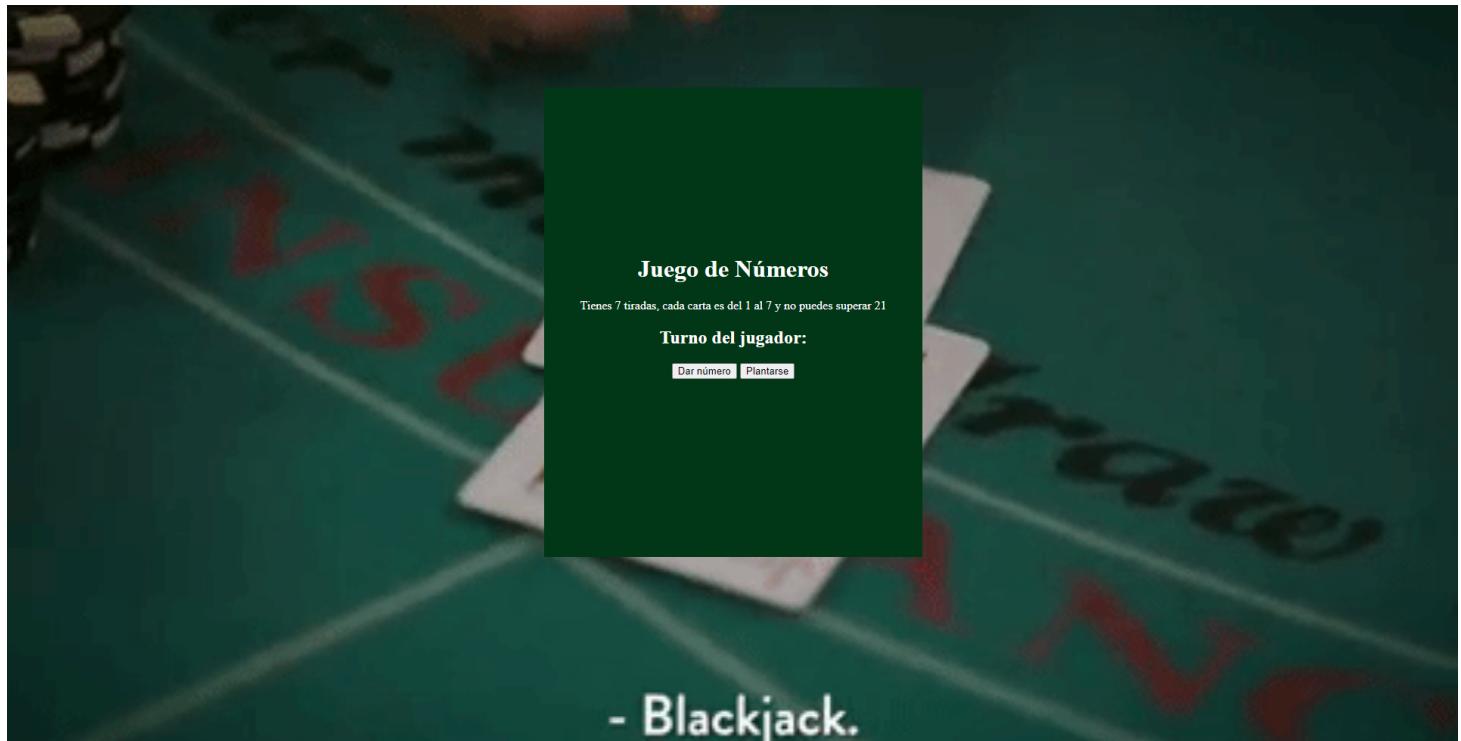
Si el usuario introduce el número objetivo correcto, el juego emitirá un mensaje de "Felicitaciones" indicando que el usuario ha adivinado correctamente el número.

```
else {
    // Si el número es adivinado, se elimina todo lo que hay en la página y se muestra un mensaje de felicitaciones
    var successDiv = document.createElement("div");
    successDiv.classList.add("success-message");
    successDiv.innerHTML = "¡Felicitaciones, adivinaste el número!";
    document.body.appendChild(successDiv);
}
```

### 6.1.3. Escalabilidad

Este juego cuenta con una gran escalabilidad implementando un sistema de puntuación que permite al usuario medir su progreso, ver cuando a sido la vez que mas rápido acertó y competir con amigos y familiares para ver quién puede adivinar el número correcto en el menor número de intentos.

## 6.2. Blackjack



### 6.2.1. Interfaz

Blackjack es una versión electrónica del popular juego de cartas en el que los jugadores compiten contra la CPU para obtener una mano con un puntaje lo más cercano posible a 21 sin pasarse.

El objetivo del juego es superar al la CPU, mediante la estrategia y la suerte.

Este juego cuenta con una interfaz de usuario atractiva y fácil de usar, diseñada para brindar una experiencia de juego intuitiva y fluida.

El juego se divide en dos partes, una para el jugador y otra para la computadora, donde se pueden ver las cartas de cada uno y el puntaje obtenido.

El juego ofrece a los jugadores una gran cantidad de opciones y estrategias para maximizar sus posibilidades de ganar.

Una de las principales características es la capacidad del usuario de elegir si desea recibir las 7 cartas del 1 al 7 o plantarse en cualquier momento, cuando vea que estratégicamente está cerca del 21. Esto es posible a través de 2 botones; Dar numero o Plantarse

La opción de recibir las 7 cartas ofrece al jugador la posibilidad de obtener una mejor mano, aumentando sus posibilidades de ganar. Sin embargo, también aumenta el riesgo de pasarse de 21 y perder la partida.

Por otro lado, la opción de plantarse es ideal para aquellos jugadores que prefieren jugar con precaución y evitar el riesgo de pasarse de 21. Esta opción les permite mantener su mano actual y competir contra la computadora con las cartas que tienen.



```
<div class="container">
  <div id="layer">
    <h1>Juego de Números</h1>
    <div id="game">
      <p>Tienes 7 tiradas, cada carta es del 1 al 7 y no puedes superar 21</p>
      <h2>Turno del jugador:</h2>
      <button id="give-number">Dar número</button>
      <button id="stand">Plantarse</button>
      <br><br>
      <div id="player-score"></div>
    </div>
    <div id="player-numbers"></div>
    <br>
    <div id="result"></div>
    <div id="cpu">
      <p>Turno de la CPU:</p>
      <div id="cpu-score"></div>
    </div>
  </div>
</div>
```

## **6.2.2. Casos de prueba definidos y resultados**

1. Prueba de límite: en esta prueba se comprueba que el juego no permite al jugador recibir más de 7 cartas del 1 al 7.

### **CASO DE PRUEBA**

En esta prueba, el objetivo es comprobar que el juego no permite al jugador recibir más de 7 cartas en una partida.



### **RESULTADO**

<p><b>Juego de Números</b></p> <p>Tienes 7 tiradas, cada carta es del 1 al 7 y no puedes superar 21</p> <p><b>Turno del jugador:</b></p> <p>Dar número Plantarse</p> <p>Resultado: 28 Cartas: 3, 5, 7, 6, 1, 4, 2</p>	<p><b>Juego de Números</b></p> <p>Cartas: 3, 5, 7, 6, 1, 4, 2</p> <p><b>¡Perdiste!</b></p> <p>Turno de la CPU: 16</p>
---	---

Durante esta prueba, simulamos una partida de juego en la que el jugador pre-siona repetidamente el botón para recibir cartas, hasta un total de 7 veces.

Una vez que el jugador ha recibido su séptima carta, el juego debe finalizar automáticamente, y se debe comparar el puntaje obtenido por el jugador con el puntaje obtenido por la computadora.

El objetivo de esta prueba es asegurarnos de que el juego está programado para finalizar correctamente una vez que el jugador ha recibido su séptima carta, y que el puntaje del jugador es comparado correctamente con el puntaje de la computadora

```
let numbers = [1, 2, 3, 4, 5, 6, 7];
let playerScore = 0;
let computerScore = 0;
let playerTurn = true;
let playerNumbers = [];

document.getElementById("give-number").addEventListener("click", function() {

  if (playerNumbers.length < 7) {
    let number = Math.floor(Math.random() * 7) + 1;
    playerScore += number;
    playerNumbers.push(number);
    document.getElementById("player-score").innerHTML = "Resultado: " + playerScore;
    document.getElementById("player-numbers").innerHTML = "Cartas: " + playerNumbers.join(", ");
  } else {
    endTurn();
  }
});
```

El juego a sido programado con un código que solo de 7 números del 1 al 7 aleatorios al usuario.

Con una variable para almacenar los números permitidos en el juego para el jugador y la CPU

Se comprueba si aún no se han dado todos los números, se da otro número al jugador, que se genera de un numero aleatorio entre 1 y 7, luego se suma el numero generado al puntaje del jugador, se actualiza el resultado de las cartas que te se da al usuario y el total que lleva.

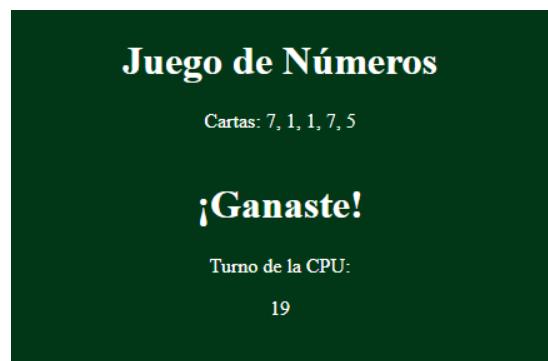
2. Prueba de estrategia: en esta prueba se comprueba que el juego permite al jugador utilizar diferentes estrategias y tácticas para maximizar sus posibilidades de ganar, como la opción de plantarse cuando está cerca del 21.

### CASO DE PRUEBA

En esta prueba, el objetivo es comprobar que el juego finalice y se sumen todas las cartas una vez que el usuario pulse el botón de plantarse.



### RESULTADO



Esta prueba comprueba que el juego permite al jugador elegir la opción de plantarse en cualquier momento, cuando vea que estratégicamente está cerca del 21,

Simulamos una partida de juego en la que el jugador recibe 6 cartas, con el objetivo de acabar lo más cerca de 21 posible, plantarse y que el juego finaliza correctamente una vez que el jugador ha elegido la opción de plantarse.

```
let playerTurn = true;
document.getElementById("stand").addEventListener("click", endTurn);

function endTurn() {
    document.getElementById("game").style.display = "none";
    document.getElementById("cpu").style.display = "block";
    endGame();
}
function endGame() {
    document.getElementById("result").style.display = "block";
}
```

El juego a sido programado con un código que recoge una variable otra para indicar si es turno del jugador.

Añade un evento al botón de "Plantarse" para finalizar turno, finaliza el turno del jugador, oculta los botones de control, muestra el turno de la CPU y con esto finaliza el juego.

- 
3. Prueba de barajar las cartas: en esta prueba se comprueba que el juego baraja las cartas de manera aleatoria y justa para todos los jugadores.

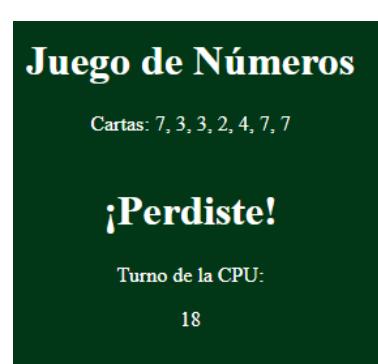
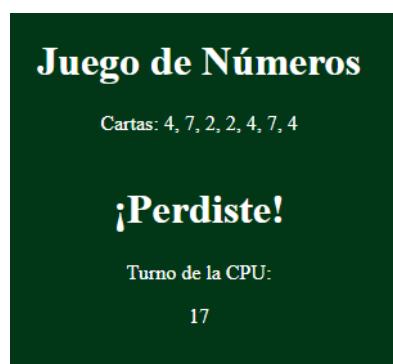
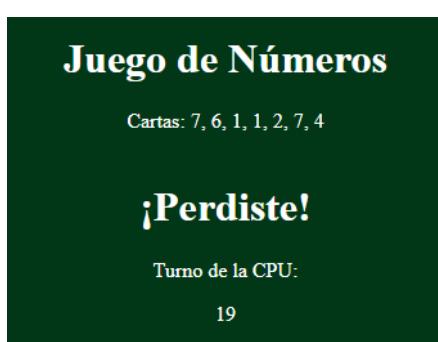
### CASO DE PRUEBA

En esta prueba, el objetivo es comprobar que el juego reparte diferentes cartas en varias partidas diferentes, tanto al jugador como en el resultado final de la CPU.



## RESULTADO

Esta prueba comprueba que el sistema de barajar las cartas es efectivo, para asegurar que las cartas son generadas de manera aleatoria y justa para todos los jugadores he reiniciado varias veces el juego y repartido las 7 cartas para comprobar que cada vez sea diferente.



```
let numbers = [1, 2, 3, 4, 5, 6, 7];
let playerNumbers = [];

document.getElementById("give-number").addEventListener("click", function() {
  if (playerNumbers.length < 7) {
    let number = Math.floor(Math.random() * 7) + 1;
    playerScore += number;
    playerNumbers.push(number);
    document.getElementById("player-score").innerHTML = "Resultado: " + playerScore;
    document.getElementById("player-numbers").innerHTML = "Cartas: " + playerNumbers.join(", ");
  } else {
    endTurn();
  }
});
function endTurn() {
  document.getElementById("game").style.display = "none";
  document.getElementById("cpu").style.display = "block";

  while (computerScore < 16) {
    let number = Math.floor(Math.random() * 7) + 1;
    computerScore += number;
    document.getElementById("cpu-score").innerHTML = computerScore;
  }
  // Finaliza el juego
  endGame();
}
```

El juego a sido programado con un código para que recoja una variable para almacenar los números permitidos en el juego para el jugador y la CPU y otra para almacenar todos los números del jugador.

Se comprueba si aún no se han dado todos los números, si es así se da otro número al jugador, que se genera de un numero aleatorio entre 1 y 7, que se almacenan en la variable.

Una vez finalizado el turno del usuario oculta los botones de control, muestra el turno de la CPU, dando números aleatorios del 1 al 7 a la CPU hasta que su puntuación sea mayor o igual a 16 para que la CPU no se quede corta en cartas. Después de esto se suma a resultado final.

- 
4. Prueba de usabilidad: en esta prueba se comprueba que el juego es fácil de usar y comprender para el jugador, que la interfaz es intuitiva y que haya una victoria y una derrota.

#### CASO DE PRUEBA

En esta prueba, el objetivo es comprobar que haya un mensaje de Victoria y Derrota cuando el juego finalice y se compare los resultados.



## RESULTADO



En esta prueba he reiniciado varias veces el juego para intentar ganar uno y perder otro para comprobar que sale un mensaje de Victoria o Derrota dependiendo del resultado.

```
let playerScore = 0;
let computerScore = 0;

function endGame() {
    document.getElementById("result").style.display = "block";
    if (playerScore > computerScore && playerScore <= 21 || computerScore > 21) {
        document.getElementById("result").innerHTML = "<h1>¡Ganaste!</h1>";
    } else {
        // En cualquier otro caso, muestra un mensaje de derrota
        document.getElementById("result").innerHTML = "<h1>¡Perdiste!</h1>";
    }
}
```

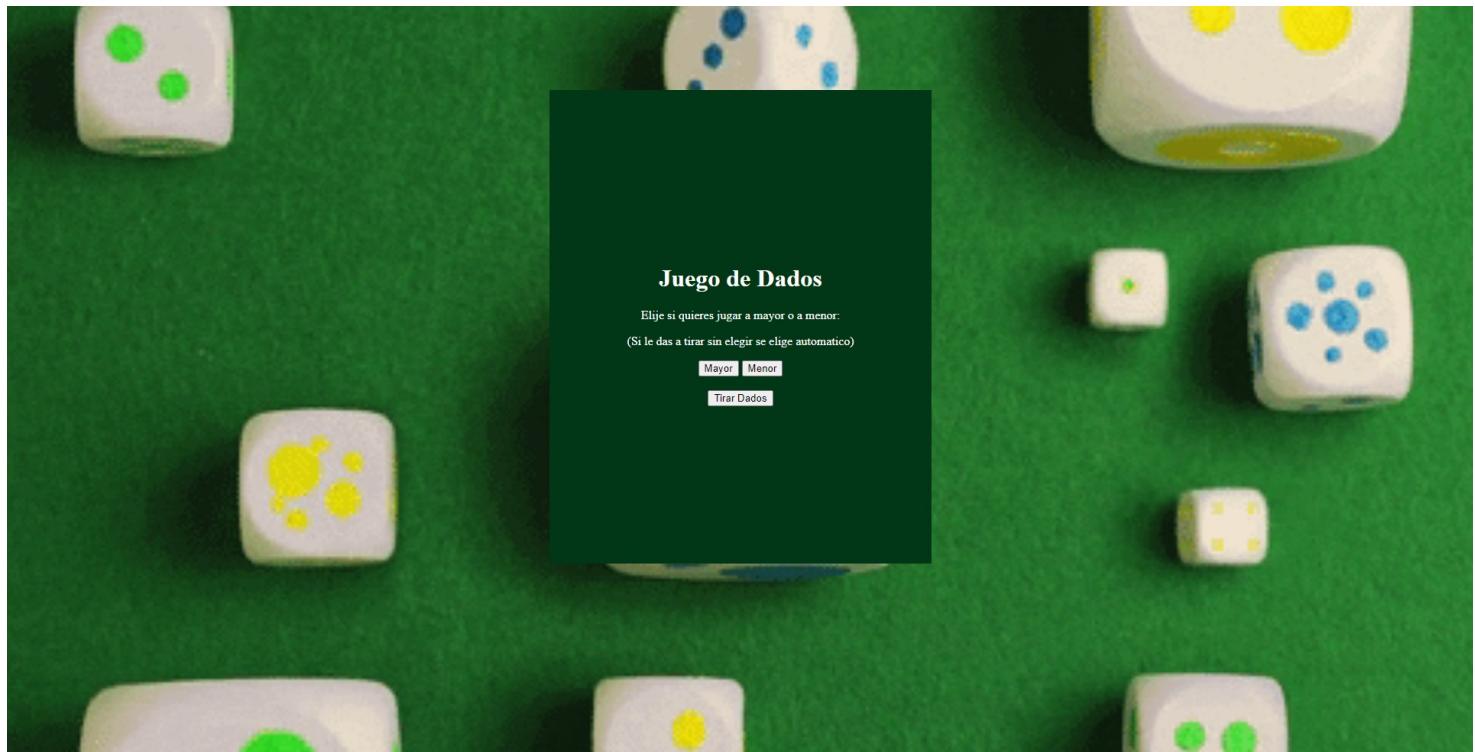
El juego a sido programado con un código para recoger variables de marcador del jugador y CPU, que son comparadas en una función para lanzar un mensaje de Victoria y Derrota dependiendo del resultado final que ha sido sumado anteriormente en las funciones respectivamente de cada jugador.

### 6.2.3. Escalabilidad

El juego cuenta con un sistema escalabilidad alta debido a la implementación de una moneda virtual que se consigue jugando al juego diariamente, permitiendo usar al jugador la moneda a su propio disfrute dentro del juego.

El usuario deberá elegir la cantidad de monedas que quiere apostar antes de comenzar a jugar, esto añade un elemento de emoción y estrategia adicional al juego. De recompensa podrá conseguir mas monedas y canjearlas por diferentes premios físicos dentro de la plataforma

### 6.3. Dados



#### 6.3.1. Interfaz

El juego de Dados ofrece a los jugadores la emoción de un juego de azar y la oportunidad de poner a prueba su habilidad para predecir el resultado de un lanzamiento de dados.

El objetivo del juego es elegir entre jugar a "mayor" o "menor" y luego competir contra la computadora para ver quién saca el número más cercano al elegido. En caso de seleccionar "Tirar Dados" sin seleccionar "mayor" o "menor" se generara el juego aleatoriamente como quiera la CPU

El juego comienza con la selección de la modalidad de juego, "mayor" o "menor", después de lo cual el jugador y la computadora lanzan un dado y el número obtenido se compara con el objetivo elegido.

El jugador que tenga el número más cercano al objetivo elegido gana la ronda.

El juego cuenta con una interfaz intuitiva y fácil de usar, con un botón de lanzamiento de dado y una pantalla que muestra el resultado del lanzamiento. Además, el juego cuenta con un sistema de puntuación para llevar un registro de los puntos obtenidos por cada jugador y así lanzar un mensaje de victoria o derrota.

The screenshot shows a dark-themed web application for a dice game. On the left, there's a dark sidebar with the title "Juego de Dados". The main content area has a light background. It displays a message: "Elije si quieres jugar a mayor o a menor: (Si le das a tirar sin elegir se elige automatico)". Below this are three buttons: "Mayor", "Menor", and "Tirar Dados". To the right of the screenshot is the corresponding HTML code:

```
<div class="container">
  <div id="layer">
    <h1>Juego de Dados</h1>
    <div id="game">
      <p>Elije si quieres jugar a mayor o a menor:</p>
      <p>(Si le das a tirar sin elegir se elige automatico)</p>
      <button id="higher">Mayor</button>
      <button id="lower">Menor</button>
    </div>
    <br>
    <div id="dice">
      <button id="roll">Tirar Dados</button>
      <div id="result"></div>
    </div>
  </div>
</div>
```

### **6.3.2. Casos de prueba definidos y resultados**

1. Prueba de modalidad de juego: en esta prueba se comprueba que el juego permite al jugador elegir entre jugar a "mayor" o "menor" antes de comenzar la partida.

#### **CASO DE PRUEBA**

En esta prueba, el objetivo es comprobar que tanto el botón de Mayor como el de Menor cumplan su función y haga que la partida fluya según lo seleccionado.

The screenshot shows the same dice game interface as before, but now it displays the results of the test case. The message "Elije si quieres jugar a mayor o a menor: (Si le das a tirar sin elegir se elige automatico)" is still present. The "Tirar Dados" button is highlighted in pink, indicating it has been interacted with. The "Mayor" and "Menor" buttons are also visible.

## RESULTADO

Al seleccionar la modalidad de juego "mayor" o "menor" antes de comenzar la partida, el juego asigna correctamente el objetivo de la partida y permite al jugador comenzar el juego.

Esto se debe al código que se encarga de registrar la selección del jugador y asignar el objetivo correcto.

### CASO MAYOR



### CASO MENOR



Este juego fue programado con un código para recoger variables de tipo juego para saber si se juega a "mayor" o "menor".

Con un evento al pulsar el botón de "mayor" o "menor" se activa la función con la variable de tipo juego para ocultar el menú y que se lance el dado con el botón da lanzar dado.

```
let gameType = "";
document.getElementById("higher").addEventListener("click", function() {
    gameType = "higher";
    document.getElementById("game").style.display = "none";
    document.getElementById("dice").style.display = "block";
});

document.getElementById("lower").addEventListener("click", function() {
    gameType = "lower";
    document.getElementById("game").style.display = "none";
    document.getElementById("dice").style.display = "block";
});
```

- 
2. Prueba de lanzamiento de dado: en esta prueba se comprueba que el juego lanza el dado de manera aleatoria y justa para ambos jugadores.

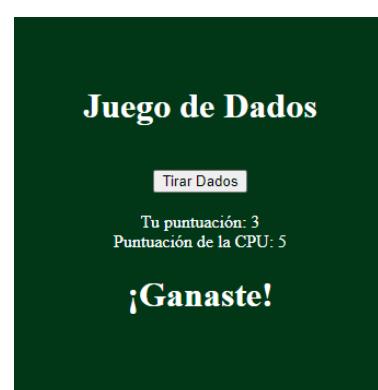
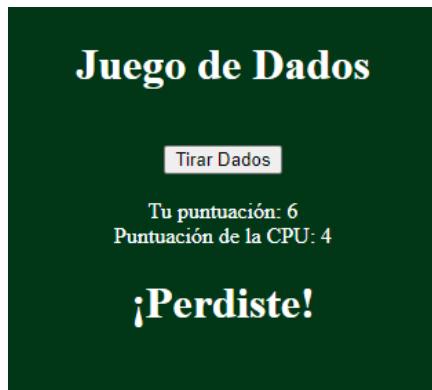
### CASO DE PRUEBA

En esta prueba, reiniciar el juego varias veces para comprobar que el dado se lanza de manera aleatoria y justa para ambos jugadores. La prueba se hará en el mismo modo de juego "menor".



### RESULTADO

Se lanza el dado de manera aleatoria y justa para ambos jugadores, esto se puede comprobar al jugar varias veces y ver que los números salen de manera aleatoria.



Esto se debe al código que se encarga de generar un número aleatorio para el lanzamiento del dado.

```
const dice = [1, 2, 3, 4, 5, 6];

document.getElementById("roll").addEventListener("click", function() {

    playerScore = dice[Math.floor(Math.random() * dice.length)];
    computerScore = dice[Math.floor(Math.random() * dice.length)];

    document.getElementById("result").innerHTML = `<br>Tu puntuación: ${playerScore}<br>Puntuación de la CPU: ${computerScore}`;
});
```

Se hace una variable para almacenar los números permitidos en el juego para el jugador y la CPU.

Se añade un evento al botón roll para lanzar los dados y determinar el resultado del juego, se establece el puntaje del jugador y de la CPU mediante un número aleatorio de la variable después de darle al botón de lanzar dado.

- 
3. Prueba de comparación de resultados: en esta prueba se comprueba que el juego compara correctamente los resultados del lanzamiento de dado con el objetivo elegido y determina al ganador de la ronda.

### CASO DE PRUEBA

En esta prueba, el objetivo es comprobar que haya un mensaje de Victoria y Derrota cuando el juego finalice y se compare los resultados.



## RESULTADO

En esta prueba he reiniciado varias veces el juego para intentar ganar uno y perder otro para comprobar que sale un mensaje de Victoria o Derrota dependiendo del resultado.



El juego compara correctamente los resultados del lanzamiento de dado con el objetivo elegido y determina al ganador de la ronda, esto se puede comprobar al ver que el juego indica correctamente al ganador de cada ronda.

```
if ((gameType === "higher" && playerScore > computerScore) || (gameType === "lower" && playerScore < computerScore)) {  
    document.getElementById("result").innerHTML += "<br><h1>¡Ganaste!</h1>";  
} else if ((gameType === "higher" && playerScore < computerScore) || (gameType === "lower" && playerScore > computerScore)) {  
    document.getElementById("result").innerHTML += "<br><h1>¡Perdiste!</h1>";  
} else {  
    document.getElementById("result").innerHTML += "<br><h1>¡Empate!</h1>";  
}
```

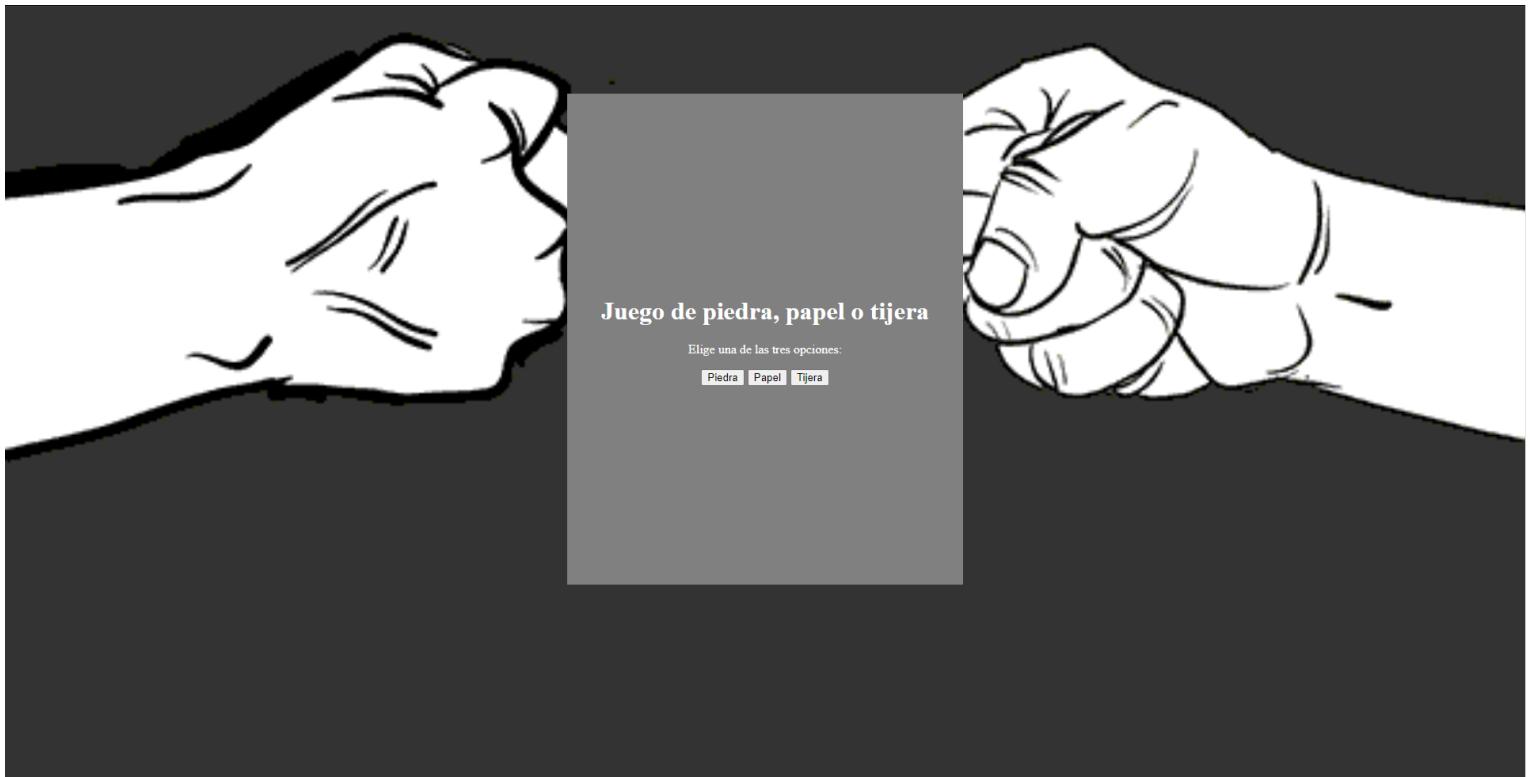
Esto se debe al código que se encarga de comparar los resultados y determinar al ganador de cada ronda.

Se verifica si el jugador ganó, perdió o empató de acuerdo al modo de juego seleccionado y comparándolo con el numero de la CPU y lanza un mensaje según el resultado.

### 6.3.3. Escalabilidad

El juego podría escalar fácilmente para soportar un mayor número de jugadores en linea jugando entre si para generar torneos o un ultimo en pie gana, ya que se habría considerado la escalabilidad en su diseño. También podría escalar guardando numero de victorias y derrotas para hacer un top 100.

## 6.4. Piedra, Papel y Tijera



### 6.4.1. Interfaz

Piedra, Papel o Tijera es un juego clásico en el que el usuario se enfrenta a la CPU en una competencia de estrategia y habilidad. El objetivo del juego es elegir correctamente entre piedra, papel o tijera para vencer al oponente.

En este juego, el usuario tiene la opción de elegir entre piedra, papel o tijera mediante botones específicos en la pantalla.



```
<div class="container">
  <div id="layer">
    <h1>Juego de piedra, papel o tijera</h1>
    <p>Elige una de las tres opciones:</p>
    <button onclick="play('piedra')">Piedra</button>
    <button onclick="play('papel')">Papel</button>
    <button onclick="play('tijera')">Tijera</button>
    <p id="result"></p>
  </div>
</div>
```

Cada opción tiene una ventaja o desventaja en relación a las otras, y el usuario debe elegir sabiamente para tener una mejor oportunidad de ganar.

La CPU también elige al azar entre piedra, papel o tijera y el juego compara los resultados para determinar al ganador de la ronda.

El juego también lleva un registro de las victorias, derrotas .

El juego es fácil de usar y comprender, con una interfaz intuitiva y gráficos atractivos. Además, se han considerado las reglas del juego de Piedra, Papel o Tijera en su desarrollo, para que sea aplicado correctamente durante la partida.

#### **6.4.2. Casos de prueba definidos y resultados**

1. Prueba de selección: Verificar que el usuario puede elegir correctamente entre piedra, papel o tijera mediante los botones de selección.

##### **CASO DE PRUEBA**

En esta prueba, el objetivo es comprobar que los tres botones funcionen correctamente y se ejecuten bien para comparar el resultado y seguir con el juego.



##### **RESULTADO**

Esta prueba se realiza para verificar que el usuario puede elegir correctamente entre piedra, papel o tijera mediante los botones de selección.

Es importante comprobar que el juego registra correctamente la elección del usuario, ya que esto es esencial para determinar al ganador de la ronda.

## Juego de piedra, papel o tijera

Elige una de las tres opciones:

CPU eligió: papel  
Usted eligió: piedra

PIEDRA

## Juego de piedra, papel o tijera

Elige una de las tres opciones:

CPU eligió: piedra  
Usted eligió: papel

PAPEL

## Juego de piedra, papel o tijera

Elige una de las tres opciones:

CPU eligió: papel  
Usted eligió: tijera

TIJERA

```
function play(playerChoice) {
    var result = "";

    if (computerChoice == playerChoice) {
        result = "<h2>Empate</h2>";
    } else if (computerChoice == "piedra") {
        if (playerChoice == "tijera") {
            result = "<h2>CPU gana</h2>";
            computerWins++;
        } else {
            result = "<h2>Jugador gana</h2>";
            playerWins++;
        }
    } else if (computerChoice == "papel") {
        if (playerChoice == "piedra") {
            result = "<h2>CPU gana</h2>";
            computerWins++;
        } else {
            result = "<h2>Jugador gana</h2>";
            playerWins++;
        }
    } else if (computerChoice == "tijera") {
        if (playerChoice == "papel") {
            result = "<h2>CPU gana</h2>";
            computerWins++;
        } else {
            result = "<h2>Jugador gana</h2>";
            playerWins++;
        }
    }
}
```

```
<button onclick="play('piedra')">Piedra</button>
<button onclick="play('papel')">Papel</button>
<button onclick="play('tijera')">Tijera</button>
```

Se genera 3 Botones play que se aplicarán en la función del jugador, cada uno para elegir la opciones del jugador.

Se hace la función para que cuando se pulse se compare con la opción y que salga con posibles resultados.

- 
1. Prueba de elección de la CPU: Verificar que la computadora elige al azar entre piedra, papel o tijera.

### CASO DE PRUEBA

En esta prueba, el objetivo es comprobar que la CPU saque valores aleatorios funcionen correctamente para comparar el resultado y seguir con el juego.



### RESULTADO

Esta prueba se lleva a cabo para verificar que la computadora elige al azar entre piedra, papel o tijera.

Es importante asegurar que la elección de la computadora sea aleatoria para garantizar un juego justo y emocionante.



TIJERA

## Juego de piedra, papel o tijera

Elige una de las tres opciones:

Piedra Papel Tijera

CPU eligió: papel  
Usted eligió: tijera

PAPEL

## Juego de piedra, papel o tijera

Elige una de las tres opciones:

Piedra Papel Tijera

CPU eligió: piedra  
Usted eligió: papel

PIEDRA

Se genera una variable es inicializada con un número aleatorio generado mediante la función Math.random()

Si el número aleatorio es menor a 0,34, entonces la elección de la computadora es "piedra".

Si el número aleatorio es mayor o igual a 0,34 y menor o igual a 0,67, entonces la elección de la computadora es "papel".

De lo contrario, la elección de la computadora es "tijera".

```
var computerChoice = Math.random();
if (computerChoice < 0.34) {
    computerChoice = "piedra";
} else if (computerChoice <= 0.67) {
    computerChoice = "papel";
} else {
    computerChoice = "tijera";
}
```

1. Prueba de comparación: Verificar que el juego compara correctamente las elecciones del usuario y la computadora para determinar al ganador de la ronda.

### CASO DE PRUEBA

En esta prueba, el objetivo es comprobar que el juego compara correctamente las elecciones del usuario y la computadora para determinar al ganador de la ronda . El juego lo ganara el mejor de 3 partidas, solo saliendo así el mensaje de Victoria o Derrota cuando alguien llegue a 3



### RESULTADO

Esta prueba se realiza para verificar que el juego compara correctamente las elecciones del usuario y la computadora para determinar al ganador de la ronda.

Es importante comprobar que el código de comparación está funcionando correctamente para garantizar que el juego sea justo y preciso.

**Juego de piedra, papel o tijera**

Elige una de las tres opciones:

Piedra Papel Tijera

CPU eligió: piedra  
Usted eligió: papel

**Jugador gana**

Jugador: 3 CPU: 1

**¡Felicitaciones, ganaste el juego!**

**Juego de piedra, papel o tijera**

Elige una de las tres opciones:

Piedra Papel Tijera

CPU eligió: papel  
Usted eligió: piedra

**CPU gana**

Jugador: 2 CPU: 3

**¡Ohh, perdiste! La CPU gano el juego.**

```
var playerWins = 0;
var computerWins = 0;
```

Variables para almacenar las victorias que van teniendo cada jugador.

```
var result = "";

if (computerChoice == playerChoice) {
    result = "<h2>Empate</h2>";
} else if (computerChoice == "piedra") {
    if (playerChoice == "tijera") {
        result = "<h2>CPU gana</h2>";
        computerWins++;
    } else {
        result = "<h2>Jugador gana</h2>";
        playerWins++;
    }
} else if (computerChoice == "papel") {
    if (playerChoice == "piedra") {
        result = "<h2>CPU gana</h2>";
        computerWins++;
    } else {
        result = "<h2>Jugador gana</h2>";
        playerWins++;
    }
} else if (computerChoice == "tijera") {
    if (playerChoice == "papel") {
        result = "<h2>CPU gana</h2>";
        computerWins++;
    } else {
        result = "<h2>Jugador gana</h2>";
        playerWins++;
    }
}
```

Este código está diseñado para comparar las opciones elegidas por el jugador y la computadora y determinar quién gana en un juego de "Piedra, Papel o Tijera".

Primero, se verifica si ambas partes han elegido la misma opción, en cuyo caso el juego termina en empate.

Si no es así, el código entra a comparar cada opción del jugador con las opciones posibles de la computadora.

Si hay una coincidencia, se establece al jugador como el ganador de la ronda.

Después de determinar al ganador, se genera una capa visual en la pantalla del juego, si el jugador gana se genera una capa verde para indicar su victoria, y si gana la CPU se genera una capa roja para indicar su victoria.

Esto ayuda al usuario a entender rápidamente quien gano la partida.

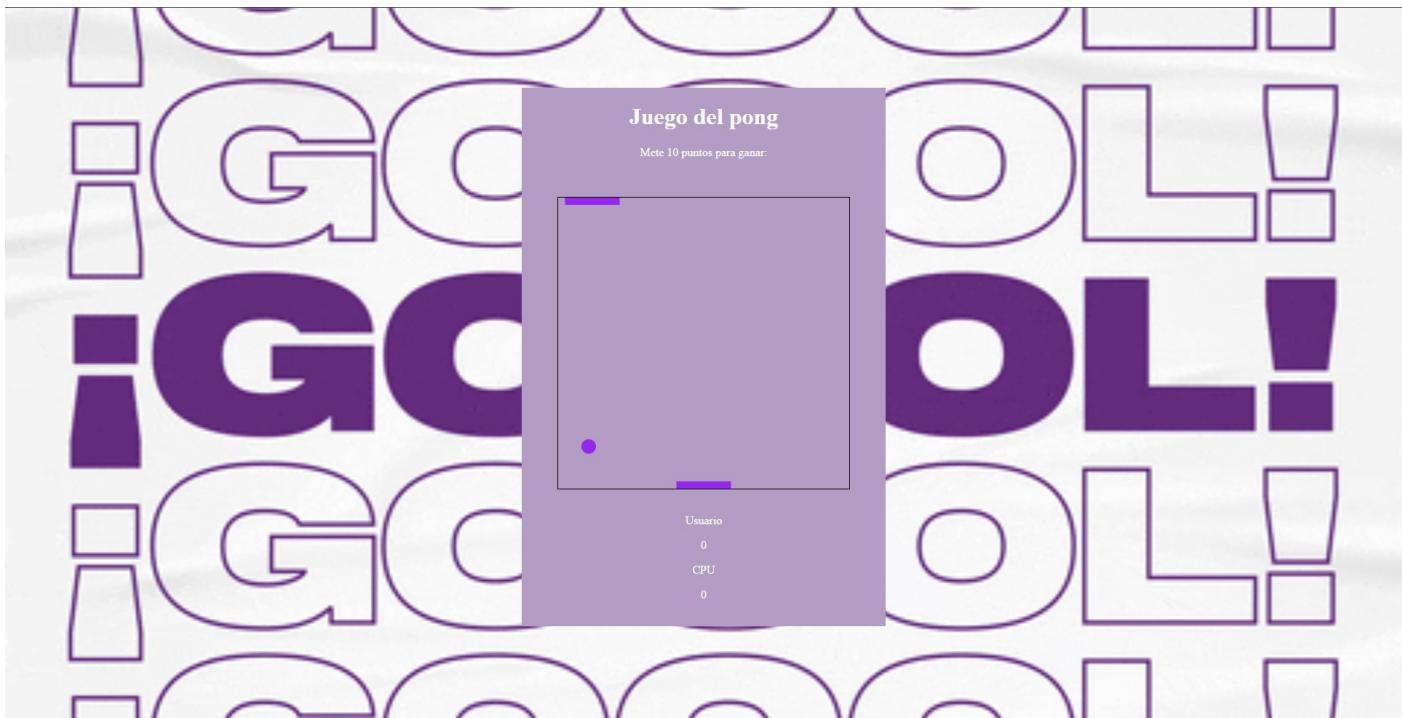
```
document.getElementById("result").innerHTML = "CPU eligió: " + computerChoice + "<br>" + "Usted eligió: " +
playerChoice + "<br>" + result + "Jugador: " + playerWins + " CPU: " + computerWins;
if (playerWins == 3) {
    var successDiv = document.createElement("div");
    successDiv.classList.add("success-message");
    successDiv.innerHTML = "¡Felicidades, ganaste el juego!";
    document.body.appendChild(successDiv);
} else if (computerWins == 3) {
    var successDiv2 = document.createElement("div");
    successDiv2.classList.add("success-message2");
    successDiv2.innerHTML = "¡Ohh, perdiste! La CPU gano el juego.";
    document.body.appendChild(successDiv2);
}
```

### **6.4.3. Escalabilidad**

El juego podría escalar fácilmente para soportar un mayor número de jugadores en linea jugando entre si para generar torneos, ya que se habría considerado la escalabilidad en su diseño.

También podría escalar guardando numero de victorias y derrotas para hacer un top 100 y competiciones a nivel mas profesionales.

## 6.5. Pong



### 6.5.1. Interfaz

Nuestro juego de Pong contra la CPU es una emocionante y desafiante experiencia de juego que ofrece al usuario la oportunidad de enfrentarse a un oponente de alta tecnología como es la CPU.

Este juego clásico se basa en el juego de tenis de mesa, en el cual el objetivo es golpear una pelota hacia el oponente, usando una barra para evitar que la pelota salga de los límites de la pantalla.

Este juego ofrece una experiencia de juego divertida y desafiante, ya que el usuario se enfrenta a un oponente CPU que se adapta y evoluciona a medida que el juego avanza.

Con una interfaz fácil de usar y controles intuitivos, este juego es perfecto tanto para jugadores experimentados como para aquellos que están empezando a jugar.

El objetivo del juego es alcanzar 10 puntos antes de la computadora. El juego se juega con una raqueta controlada por las teclas del teclado, y el usuario debe mover la raqueta para golpear la pelota y evitar que la pelota salga del campo de juego. Este juego es una versión clásica del juego de tenis de mesa en la que el usuario se enfrenta a una computadora.

```
<div class="container">
<div id="layer">
  <h1>Juego del pong</h1>
  <p>Mete 10 puntos para ganar:</p>
  <br><br>
  <!-- Se crea un canvas donde se mostrara el juego -->
  <canvas height="400" id="gameCanvas" width="400">
  </canvas>
  <br><br>
  <!-- Se crea una capa donde se muestra el puntaje del usuario -->
  <p>Usuario</p>
  <div id="userScore">
    0
  </div>
  <!-- Se crea una capa donde se muestra el puntaje de la CPU -->
  <p>CPU</p>
  <div id="cpuScore">
    0
  </div>
</div>
</div>
```

```
// Dibuja la raqueta
function drawPaddle() {
  context.beginPath();
  context.rect(paddleX, canvas.height - paddleHeight, paddleWidth, paddleHeight);
  context.fillStyle = "#9628EC";
  context.fill();
  context.closePath();
}

// Dibujar la raqueta de la CPU
function drawCpuPaddle() {
  context.beginPath();
  context.rect(cpuPaddleX, 0, cpuPaddleWidth, cpuPaddleHeight);
  context.fillStyle = "#9628EC";
  context.fill();
  context.closePath();
}

// Dibujar la pelota
function drawBall() {
  context.beginPath();
  context.arc(x, y, 10, 0, Math.PI * 2);
  context.fillStyle = "#9628EC";
  context.fill();
  context.closePath();
}
```

```
// Obtener una referencia del canvas y su contexto
var canvas = document.getElementById("gameCanvas");
var context = canvas.getContext("2d");

// Marcadores iniciados en 0
var userScore = 0;
var cpuScore = 0;

// Posiciones iniciales de la pelota
var x = canvas.width / 2;
var y = canvas.height / 2;

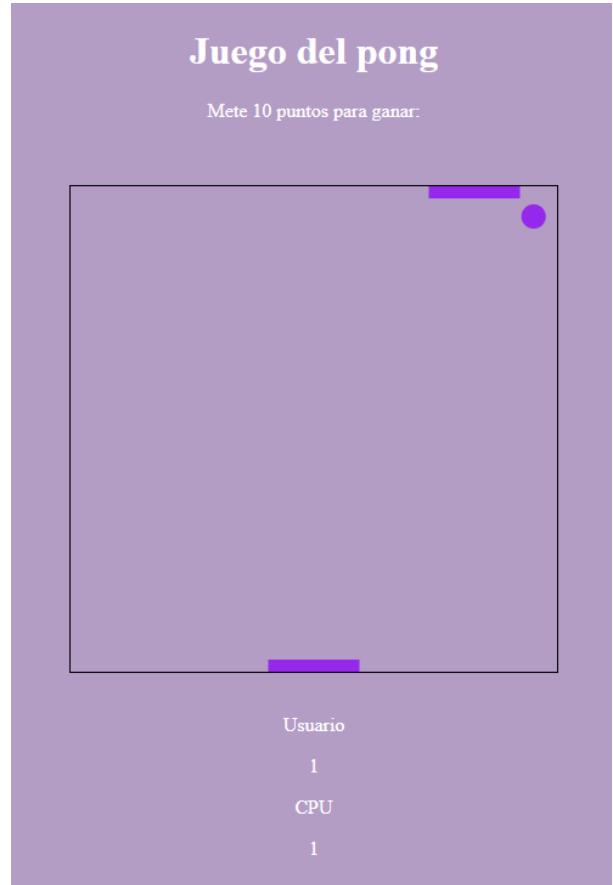
// Velocidad inicial de la pelota
var dx = 2;
var dy = -2;

// Ancho y alto de la raqueta del jugador con la que jugara
var paddleWidth = 75;
var paddleHeight = 10;

// Posición inicial de la raqueta
var paddleX = (canvas.width - paddleWidth) / 2;

// Ancho y alto de la raqueta de la CPU
var cpuPaddleWidth = 75;
var cpuPaddleHeight = 10;

// Posición inicial de la raqueta de la CPU
var cpuPaddleX = (canvas.width - cpuPaddleWidth) / 2;
```

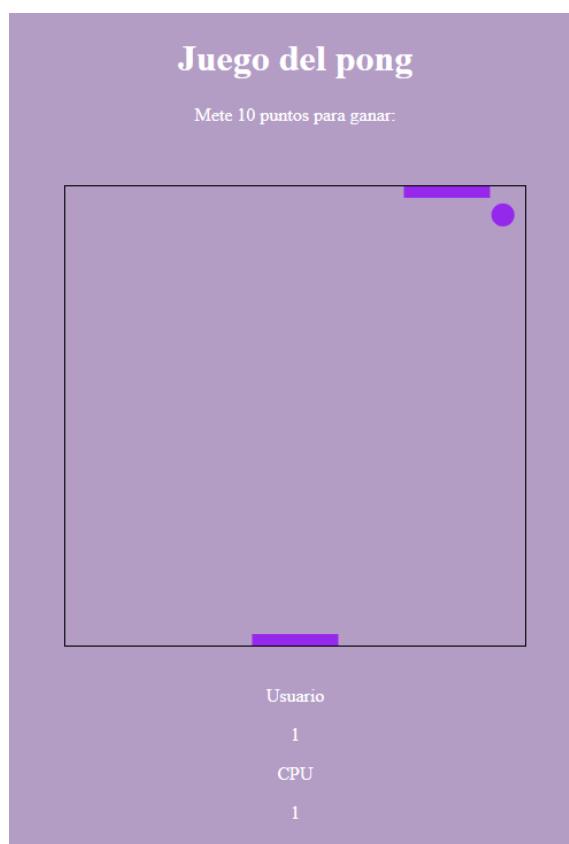


### **6.5.2. Casos de prueba definidos y resultados**

1. Prueba de movimiento de raqueta: Al presionar las teclas del teclado, se observa que la raqueta se mueve correctamente en la dirección deseada.

#### **CASO DE PRUEBA**

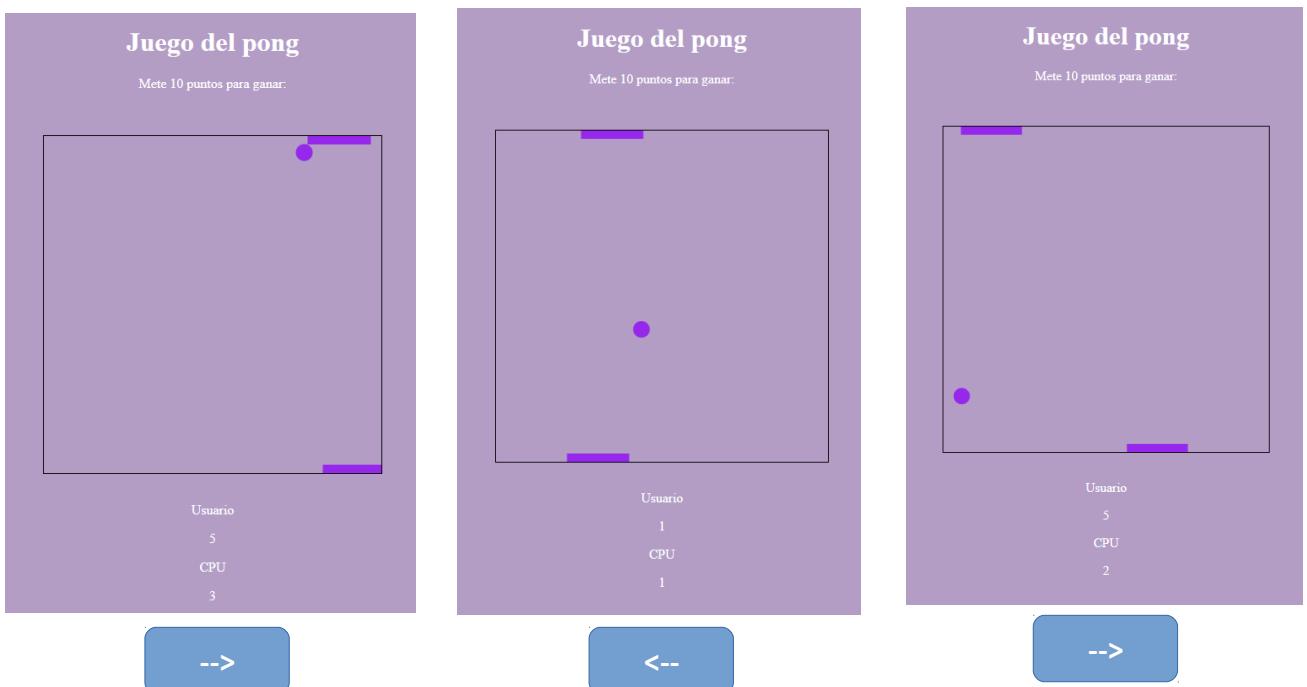
En esta prueba se comprobará que al presionar las teclas del teclado, la raqueta se mueve correctamente en la dirección deseada. Se pueden probar las teclas de izquierda, derecha, arriba y abajo, para comprobar que la raqueta se mueve correctamente en cada dirección.



## RESULTADO

Realizamos la prueba presionando las teclas del teclado, y se observó que la raqueta se movió correctamente en la dirección deseada.

1.



Esto es gracias al código que establece las funciones de movimiento para cada tecla presionada.

Creamos el evento de la escucha del teclado.

Manejo de eventos por teclado, keyDownHandler se activa cuando el usuario pulse una tecla y keyUpHandler se activa al levantar la tecla

Si la tecla presionada es derecha se activa rightPressed y si es izquierda leftPressed con un true .

Si la tecla soltada es derecha se establece la variable rightPressed y si es izquierda leftPressed con un false .

```
document.addEventListener("keydown", keyDownHandler);
document.addEventListener("keyup", keyUpHandler);

function keyDownHandler(e) {
    if (e.key == "Right" || e.key == "ArrowRight") {
        rightPressed = true;
    } else if (e.key == "Left" || e.key == "ArrowLeft") {
        leftPressed = true;
    }
}

function keyUpHandler(e) {
    if (e.key == "Right" || e.key == "ArrowRight") {
        rightPressed = false;
    } else if (e.key == "Left" || e.key == "ArrowLeft") {
        leftPressed = false;
    }
}
```

- 
2. Prueba de golpeo de la pelota CPU: Cuando la pelota golpee contra la raqueta de la CPU que salga disparada en dirección el usuario y pueda seguir aleatoriamente a la pelota.

### CASO DE PRUEBA

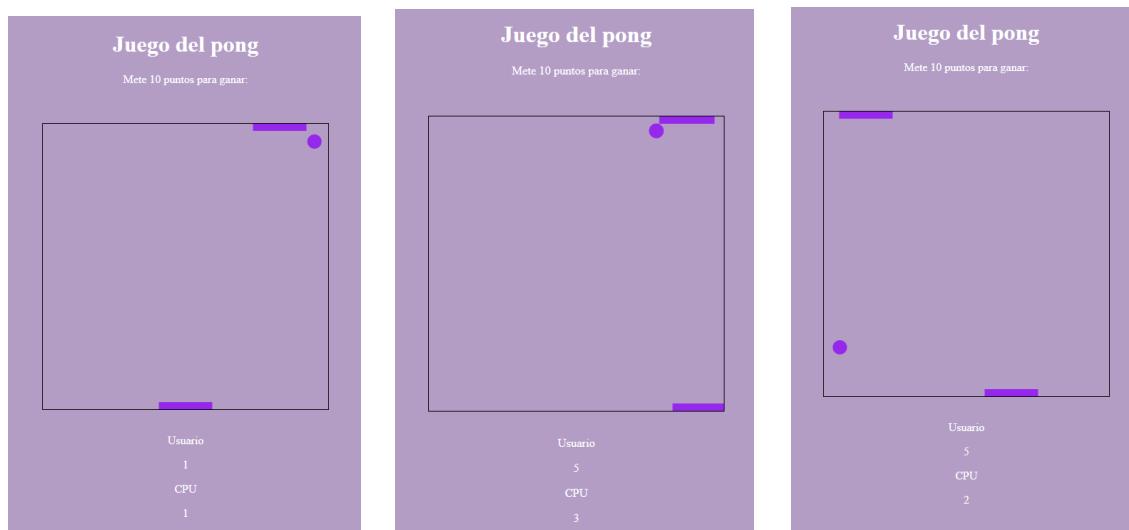
Cuando la pelota golpee contra la raqueta de la CPU que salga disparada en dirección el usuario y pueda seguir aleatoriamente a la pelota

## RESULTADO

Realizamos la prueba jugando contra la computadora y observamos que la CPU movió la raqueta de forma adecuada para tocar la pelota y devolverla al usuario.

En el código implemente un algoritmo que permitía a la CPU seguir la pelota y mover la raqueta de manera efectiva para tocar la pelota.

Además, reajuste la dificultad a un 70% para que fuera un desafío para el usuario, ya que mas de un 70 era muy favorable para la CPU y menos de 70 para el usuario.



Hacemos que la raqueta de la CPU se mueva hacia la pelota. Luego que la maquina no gane siempre con un 70% de posibilidad de moverse hacia la pelota y que aumenta y decremente la velocidad de la raqueta de la CPU en la velocidad de X que es la pelota para así moverse.

```
if (Math.random() < 0.7) {
    if (cpuPaddleX + cpuPaddleWidth < x) {
        cpuPaddleX += cpuPaddleSpeed;
    } else if (cpuPaddleX > x) {
        cpuPaddleX -= cpuPaddleSpeed;
    }
}
```

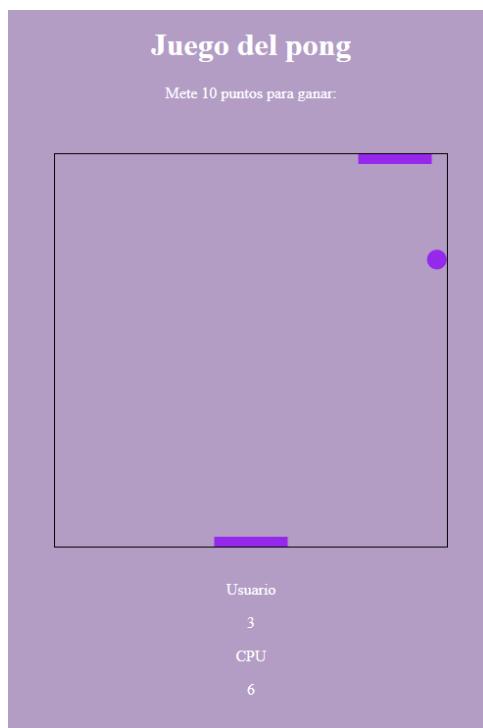
Además , se le puso colisión para repeler la pelota hacia el usuario, siendo posición de Y menos el radio de la pelota (10) es menor o igual que el alto de la raqueta CPU y si la posición X de la pelota esta entre el inicio y fin de la raqueta invierte el eje de Y para que rebote en otra dirección

```
if (y - 10 <= cpuPaddleHeight) {
    if (x > cpuPaddleX && x < cpuPaddleX + cpuPaddleWidth) {
        dy = -dy;
    }
}
```

- 
3. Prueba de límites del campo de juego: Al golpear la pelota varias veces, se observa que la pelota no sale del campo de juego debido al código que limita los límites del campo.

### CASO DE PRUEBA

Prueba de límites del campo de juego: En esta prueba se comprobará que la pelota no sale del campo de juego. Se puede hacer esto golpeando la pelota varias veces y observando si la pelota sigue dentro del campo de juego.

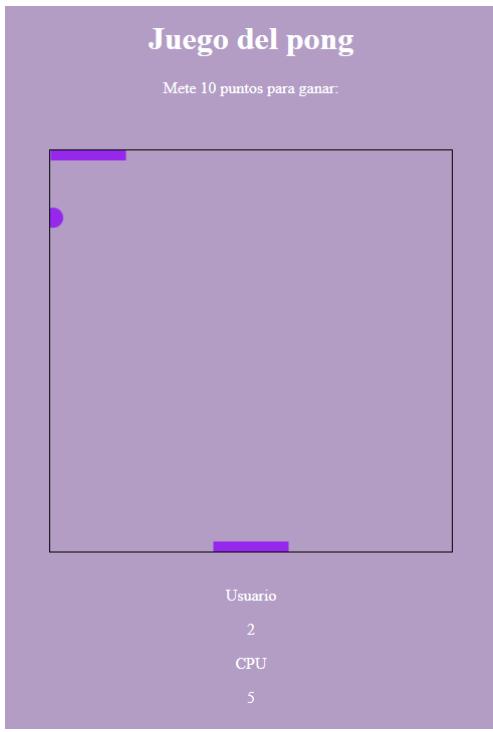


### RESULTADO

Realizamos la prueba golpeando la pelota varias veces, y se observó que la pelota no salió del campo de juego.

Esto es gracias al código que limita los límites del campo y hace que la pelota rebote en caso de tocar los límites.

1. Colisión con la pared derecha e invierte el sentido de la pelota.
2. Colisión con la pared izquierda e invierte el sentido de la pelota
3. Colisión con la pared superior, si la toca anota punto para el usuario e inicia el juego
4. Si la pelota no colisiona contra la raqueta del usuario significa que ha colisionado contra la pared de abajo y seria punto para la CPU



```
// Colisión con la pared derecha e invierte el sentido de la pelota
if (x + dx > canvas.width - 10) {
    dx = -dx;
}

// Colisión con la pared izquierda e invierte el sentido de la pelota
if (x + dx < 10) {
    dx = -dx;
}

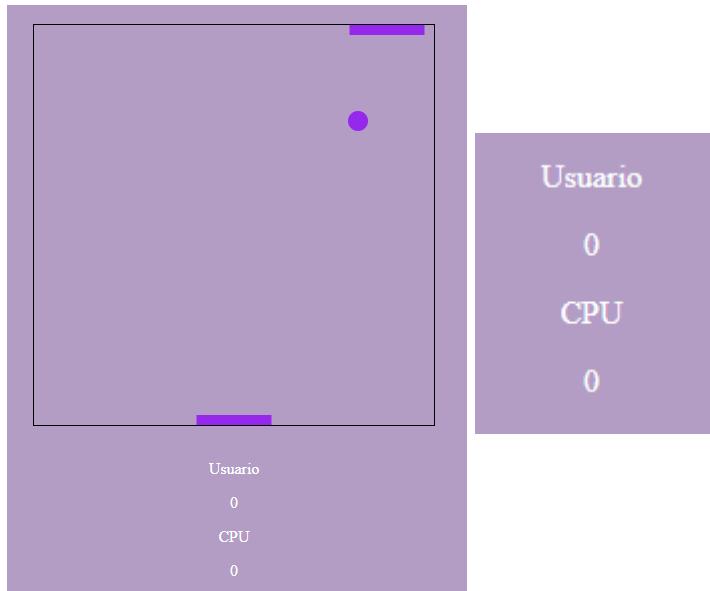
// Colisión con la pared superior, si la toca anota punto para el usuario e inicia el juego
if (y + dy < 10) {
    // Aumentar marcador del usuario
    userScore++;
    document.getElementById("userScore").innerHTML = userScore;
    initGame();
}
```

```
if (y + dy > canvas.height - paddleHeight - 10) {
    if (x > paddleX && x < paddleX + paddleWidth) {
        dy = -dy;
    } else {
        // Aumentar marcador de la CPU y reinicia el juego
        cpuScore++;
        document.getElementById("cpuScore").innerHTML = cpuScore;
        initGame();
    }
}
```

- 
4. Prueba de puntos: Al golpear la pelota varias veces, se observa que el contador de puntos se actualiza correctamente para ambos jugadores.

### CASO DE PRUEBA

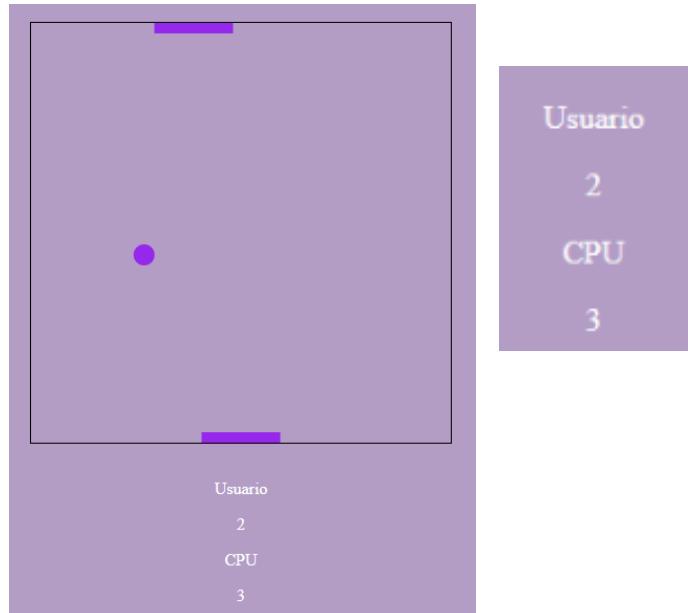
En esta prueba se comprobará que el contador de puntos se actualiza correctamente para ambos jugadores. Se pueden hacer varios golpes para asegurarse de que el contador de puntos se actualiza correctamente cada vez que alguien golpea la pelota dentro del campo de juego.



### RESULTADO

Realizamos varios golpes y se observó que el contador de puntos se actualizó correctamente para ambos jugadores.

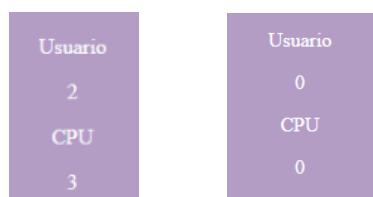
Esto es gracias al código que lleva el registro de los puntos y los actualiza correctamente cada vez que alguien golpea la pelota dentro del campo de juego.



- 
5. Prueba de victoria: Al alcanzar los 10 puntos antes de la computadora, se observa que el juego termina y se muestra un mensaje de victoria o derrota en pantalla.

#### CASO DE PRUEBA

En esta prueba se comprobará que el juego termina y se muestra un mensaje de victoria o derrota en pantalla al alcanzar los 10 puntos antes de la CPU.



## RESULTADO

Realizamos varias partidas y se observó que al alcanzar los 10 puntos antes de la computadora, el juego termina y se muestra un mensaje de victoria o derrota en pantalla.

Esto es gracias al código que establece la condición de victoria y derrota y despliega un mensaje en consecuencia.



Inicia las variables a 0

```
// Marcadores iniciados en 0
var userScore = 0;
var cpuScore = 0;
```

Comprueba si alguien ha llegado a 10 para ver quien gana.

Aumenta el marcador y borra el juego de la pantalla si el usuario hace 10 puntos y lanza el mensaje de “Felicitaciones ganador del juego” en una capa en el centro de la pantalla en color verde.

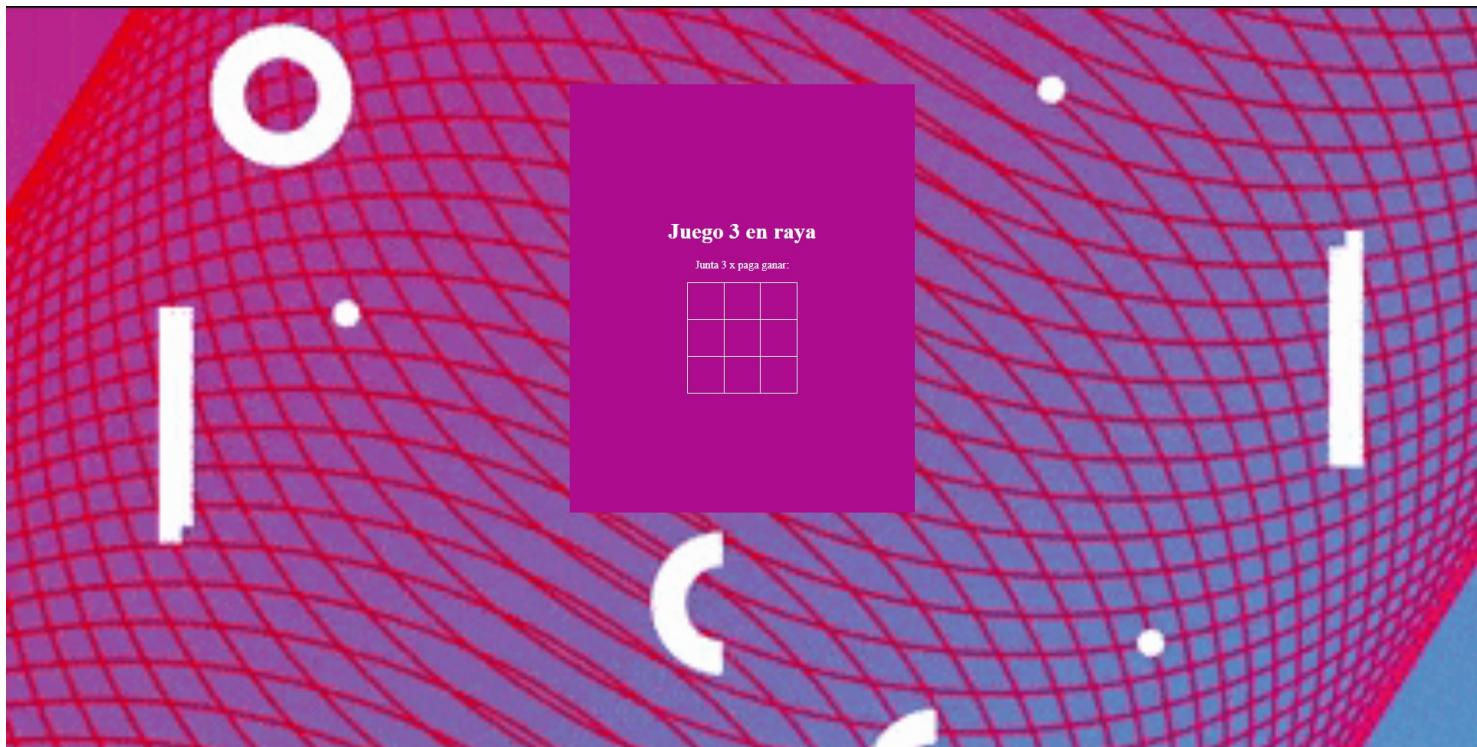
Aumenta el marcador y borra el juego de la pantalla si la CPU hace 10 puntos y lanza el mensaje de "Ohh has perdido" del juego en una capa en el centro de la pantalla en color verde.

```
if (userScore == 10) {  
    cpuScore++;  
    document.body.innerHTML = '';  
    var successDiv = document.createElement("div");  
    successDiv.classList.add("success-message");  
    successDiv.innerHTML = "¡Felicidades, ganaste el juego!";  
    document.body.appendChild(successDiv);  
} else if (cpuScore == 10) {  
    cpuScore++;  
    document.body.innerHTML = '';  
    var successDiv2 = document.createElement("div");  
    successDiv2.classList.add("success-message2");  
    successDiv2.innerHTML = "¡Ohh, perdiste! La CPU gano el juego.";  
    document.body.appendChild(successDiv2);  
}
```

### 6.5.3. Escalabilidad

El juego podría escalar fácilmente para soportar un mayor número de jugadores en linea jugando entre si para generar torneos y partidas mas profesionales, ya que se habría considerado la escalabilidad en su diseño. También podría escalar guardando numero de victorias y derrotas para hacer un top 100 y un conteo del tiempo jugado.

## 6.6. Tres en Raya



### 6.6.1. Interfaz

En el que el objetivo es colocar tres símbolos consecutivos en un tablero de 3x3 antes que la CPU.

El usuario es el símbolo "X" y la CPU es la "O" y el juego utiliza un algoritmo para determinar los movimientos de la CPU.

Con un diseño bien pensado, este juego es una excelente opción para aquellos que quieren desafiar su lógica y estrategia contra un adversario virtual.



```
<div class="container">
  <div id="layer">
    <h1>Juego 3 en raya</h1>
    <p>Junta 3 x paga ganar:</p>
    <table id="board">
      <tr>
        <td onclick="markCell(0, 0)"></td>
        <td onclick="markCell(0, 1)"></td>
        <td onclick="markCell(0, 2)"></td>
      </tr>
      <tr>
        <td onclick="markCell(1, 0)"></td>
        <td onclick="markCell(1, 1)"></td>
        <td onclick="markCell(1, 2)"></td>
      </tr>
      <tr>
        <td onclick="markCell(2, 0)"></td>
        <td onclick="markCell(2, 1)"></td>
        <td onclick="markCell(2, 2)"></td>
      </tr>
    </table>
  </div>
</div>
```

### **6.6.2. Casos de prueba definidos y resultados**

1. Prueba de selección de símbolo: Realizamos la prueba seleccionando el símbolo "X" mediante los botones de selección.

#### **CASO DE PRUEBA**

Prueba seleccionando el símbolo "X" en la celda seleccionada mediante los botones de selección.



#### **RESULTADO**

Realizamos la prueba seleccionando el símbolo "X" mediante los botones de selección, y se observó que el juego registró correctamente la elección del usuario.

Podemos ver que se ha implementado un sistema de validación que garantiza que al hacer click en una celda se marque la casilla con "X".



Se hace una función para marcar una celda del tablero y se obtiene la celda que se va a marcar

Luego se verifica que la celda este vacía y si es así se marca con una "X".

```
// Función para marcar una celda del tablero
function markCell(row, col) {
    // Obtiene la celda a marcar
    var cell = document.getElementById("board").rows[row].cells[col];

    // Verifica si la celda está vacía
    if (cell.innerHTML == "") {
        // Marca la celda con una X si esta vacia
        cell.innerHTML = "X";
        numX++;
    }
}
```

También se ha puesto en el código la opción de si hay mas de 3 "X" que se elimine la primera en poner, en el que se recorre todas las celdas buscando la "X" mas antigua

```
// Si hay 4 o más X, elimina la X más antigua
if (numX >= 4) {
    removeOldestX();
}

// Función para eliminar la X más antigua
function removeOldestX() {
    // Recorre las celdas del tablero
    for (var row = 0; row < 3; row++) {
        for (var col = 0; col < 3; col++) {
            var cell = document.getElementById("board").rows[row].cells[col];
            // Si la celda contiene una X, la elimina y sale de la función
            if (cell.innerHTML == "X") {
                cell.innerHTML = "";
                numX--;
                return;
            }
        }
    }
}
```

- Prueba de movimiento de la CPU: Realizamos la prueba jugando contra la computadora en un nivel de dificultad intermedio, y se observó que la CPU realizó movimientos estratégicos y coherentes en el tablero.

### CASO DE PRUEBA

Prueba jugando contra la computadora en un nivel de dificultad intermedio para valorar si tiene pensamiento propio.



### RESULTADO

Realizamos la prueba jugando contra la CPU, y se observó que la CPU realizó movimientos estratégicos y coherentes en el tablero.



Se ha implementado un algoritmo inteligente que analizaba el tablero y elegía la mejor opción de movimiento para la CPU.

Se hace la función para marcar una celda aleatoriamente para la CPU y se obtiene la celda que se va a marcar.

Luego se verifica que la celda este vacía y si es así se marca con una "O".

```
// Marca otra celda aleatoriamente con una O de la CPU  
markRandomCell("O");
```

```
// Función para marcar una celda aleatoriamente para la CPU  
function markRandomCell(mark) {  
    // Elige una celda aleatoriamente  
    var row = Math.floor(Math.random() * 3);  
    var col = Math.floor(Math.random() * 3);  
  
    // Obtiene la celda a marcar  
    var cell = document.getElementById("board").rows[row].cells[col];  
  
    // Verifica si la celda está vacía  
    if (cell.innerHTML == "") {  
        // Marca la celda  
        cell.innerHTML = mark;  
        if (mark == "O") {  
            numO++;  
        }  
    } else {  
        // Si la celda no está vacía, vuelve a llamar a la función  
        markRandomCell(mark);  
    }  
}
```

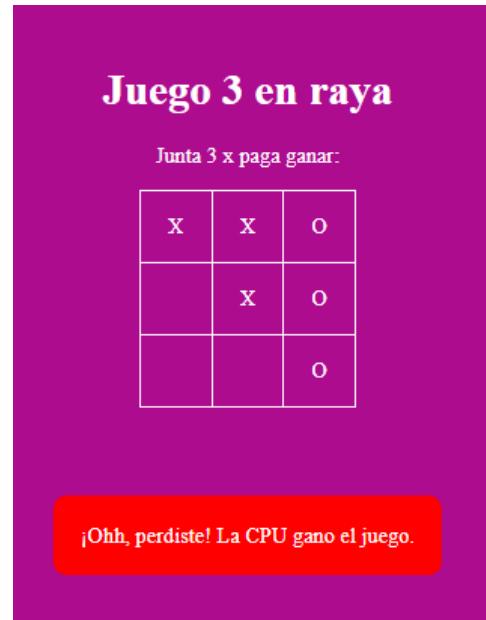
- 
3. Prueba de detección de victoria y derrotas: Realizamos la prueba jugando varias partidas analizar si cuando el usuario gane o pierda se llegara a mostrar algún mensaje de victoria o derrota en consecuencia.

### CASO DE PRUEBA

Prueba jugando varias partidas analizando si el usuario gane o pierda se llegara a mostrar algún mensaje de victoria o derrota.

## RESULTADO

Realizamos la prueba jugando varias partidas y se observó que el juego detectó correctamente cuando había una línea o diagonal de tres símbolos consecutivos en el tablero y mostró un mensaje de victoria o derrota en consecuencia.



En el código, podemos ver que se ha implementado un sistema de detección de victoria que revisaba constantemente con la función en busca de una línea o diagonal de tres símbolos consecutivos el tablero.

```
// Función para verificar quien ha ganado
function checkWin() {
    // Busca en las filas
    for (var row = 0; row < 3; row++) {
        if (checkRowWin(row)) {
            return true;
        }
    }

    // Busca en las columnas
    for (var col = 0; col < 3; col++) {
        if (checkColWin(col)) {
            return true;
        }
    }

    // Busca en las diagonales
    if (checkDiagonalWin(0, 2)) {
        return true;
    }
    if (checkDiagonalWin(2, 0)) {
        return true;
    }
    return false;
}
```

1. Busca en las filas si están los tres símbolos consecutivos

Comprueba si el contenido de las tres celdas es igual y no está vacío, devuelve true si es así, false en caso contrario.

2. Busca en las columnas si están los tres símbolos consecutivos.

Comprueba si el contenido de las tres celdas es igual y no está vacío, devuelve true si es así, false en caso contrario.

3. Busca en las diagonales si están los tres símbolos consecutivos.

Comprueba si el contenido de las tres celdas es igual y no está vacío, devuelve true si es así, false en caso contrario.

```
// Función para verificar si hay una fila ganadora
// Comprueba si el contenido de las tres celdas es igual y no está vacío, devuelve true si es así, false en caso contrario
function checkRowWin(row) {
    var cell1 = document.getElementById("board").rows[row].cells[0];
    var cell2 = document.getElementById("board").rows[row].cells[1];
    var cell3 = document.getElementById("board").rows[row].cells[2];
    return cell1.innerHTML == cell2.innerHTML && cell2.innerHTML == cell3.innerHTML && cell1.innerHTML != "";
}

// Función para verificar si hay una columna ganadora
// Comprueba si el contenido de las tres celdas es igual y no está vacío, devuelve true si es así, false en caso contrario
function checkColWin(col) {
    var cell1 = document.getElementById("board").rows[0].cells[col];
    var cell2 = document.getElementById("board").rows[1].cells[col];
    var cell3 = document.getElementById("board").rows[2].cells[col];
    return cell1.innerHTML == cell2.innerHTML && cell2.innerHTML == cell3.innerHTML && cell1.innerHTML != "";
}

// Función para verificar si hay una diagonal ganadora
// Comprueba si el contenido de las tres celdas es igual y no está vacío, devuelve true si es así, false en caso contrario
function checkDiagonalWin(row1, row2) {
    var cell1 = document.getElementById("board").rows[row1].cells[0];
    var cell2 = document.getElementById("board").rows[row1].cells[1];
    var cell3 = document.getElementById("board").rows[row2].cells[2];
    return cell1.innerHTML == cell2.innerHTML && cell2.innerHTML == cell3.innerHTML && cell1.innerHTML != "";
}
```

Una vez se ha conseguido encontrar al ganador se genera una capa en el centro de la pantalla que lanza el mensaje de "Felicitaciones, has ganado" del juego en una capa en color verde.

Una vez se ha conseguido encontrar al ganador se genera una capa en el centro de la pantalla que lanza el mensaje de "Ohh has perdido" del juego en una capa en color rojo.

```
// Verifica si hay una victoria y te lanza una capa con felicidades
if (checkWin()) {
    var successDiv = document.createElement("div");
    successDiv.classList.add("success-message");
    successDiv.innerHTML = "¡Felicidades, ganaste el juego!";
    document.body.appendChild(successDiv);
    return;
}

// Verifica si hay una derrota y te lanza una capa con derrota
if (checkWin()) {
    var successDiv2 = document.createElement("div");
    successDiv2.classList.add("success-message2");
    successDiv2.innerHTML = "¡Ohh, perdiste! La CPU gano el juego.";
    document.body.appendChild(successDiv2);
    return;
}
```

### 6.6.3. Escalabilidad

El juego podría escalar fácilmente para soportar un mayor número de jugadores en linea jugando entre si para generar torneos y partidas mas profesionales, ya que se habría considerado la escalabilidad en su diseño. También podría escalar guardando numero de victorias y derrotas para hacer un top 100 y un conteo del tiempo jugado.

## **6.7. Elementos de seguridad**

El SSL de la web iba a ser gestionado con CloudFlare que es una servidor de SSL gratuito y que en mas de 10 webs gestionadas por mi he confiado en este servidor y me ha funcionado bastante bien, pero al haber tenido el problema con la subida de la pagina, al final 000webhost me gestiona automáticamente el SSL de la pagina.

# **7. GESTIÓN DEL PROYECTO**

## **7.1. Ciclo de vida**

El ciclo de vida de mi proyecto de página web de juegos recreativos incluyó una fase de planificación, donde se establecieron los objetivos y requisitos del proyecto; una fase de análisis, donde se determinó la mejor manera de implementar los requisitos en la página web; una fase de diseño, donde se creó el diseño visual de la página web y se diseñaron las interfaces de los juegos; una fase de implementación, donde se escribió el código y se implementaron los juegos; una fase de validación, donde se realizaron pruebas y se corrigieron errores para asegurar el correcto funcionamiento de la página web y los juegos en diferentes navegadores y dispositivos. Y finalmente, una fase de documentación, donde se documentó el proyecto, se describieron los procesos y se detallaron las decisiones tomadas, se documentó el código.

## **7.2. Planificación inicial**

Se recolectó información sobre los objetivos del proyecto, que era desarrollar una página web de juegos recreativos. Se definieron las funcionalidades necesarias, como los juegos Pong, Blackjack, Adivina números, Dados, Tres en Raya y Piedra, Papel y Tijera. Además se estableció el alcance del proyecto, es decir, los lenguajes y herramientas a utilizar para el desarrollo de la página web.

Se analizaron los requisitos recolectados en la fase anterior y se determinó la mejor manera de implementarlos en la página web. Se estudió la competencia y se hizo un análisis de viabilidad del proyecto, determinando que era factible desarrollar una página web de juegos recreativos con las herramientas y lenguajes elegidos.

En esta fase se creó el diseño visual de la página web, se estableció la estructura de navegación, se determinó el esquema de color, la tipografía y los elementos de diseño necesarios para los juegos. Además se diseñaron las interfaces y layouts para cada uno de los juegos.

### **7.3. Planificación final**

Se escribió el código para hacer funcionar la página web y se implementaron los juegos. Se utilizaron lenguajes de programación como HTML, CSS y JavaScript para construir la página web y hacerla funcionar. Para cada juego se implementó la lógica y las interacciones necesarias para que los usuarios pudieran jugar

Se realizaron pruebas para asegurar que la página web y los juegos funcionan correctamente en diferentes navegadores y dispositivos. Se buscaron y corrigieron errores y se aseguró que la página web cumple con los requisitos establecidos. Se realizaron pruebas de usabilidad..

Se documentó el proyecto, se describieron los procesos y se detallaron las decisiones tomadas. Se documentó el código y se creó un manual de usuario para ayudar a otros desarrolladores a entender y mantener el proyecto.

## **8. CONCLUSIONES**

### **8.1. Valoración del proyecto**

En conclusión, estoy muy satisfecho con el desarrollo de esta página web de juegos re-creativos.

El proceso de diseño y programación fue difícil, pero al final logré crear una página web atractiva y fácil de navegar cumpliendo todos los objetivos y afrontando los problemas con ilusión.

Los juegos son divertidos y ofrecen una amplia variedad de opciones para los usuarios, a la misma vez fue divertido programarlos y hacerle pruebas.

Estoy orgulloso del trabajo realizado en el diseño y la estructura del código, ya que creo que he logrado hacerlo limpio y bien organizado, lo que facilita su mantenimiento y escalabilidad en el futuro.

En general, considero que este proyecto ha sido un gran éxito y una valiosa experiencia en mi formación como desarrollador web.

## **8.2. Posibles mejoras futuras**

Una posible mejora a futuro para esta página web podría ser la implementación de un sistema de cuentas de usuario y puntuaciones. Esto permitiría a los usuarios registrarse en la página web, guardar sus puntuaciones y compararlas con las de otros usuarios. También se podrían agregar opciones de personalización de perfil, como elegir un avatar o un fondo de pantalla.

Además, se podrían agregar nuevos juegos o categorías de juegos para mantener el interés de los usuarios y aumentar la duración de su tiempo en la página web.

Otra posible mejora sería la implementación de un sistema de recompensas, donde los usuarios puedan ganar puntos o premios por jugar y alcanzar ciertos logros en los juegos. Esto podría motivar a los usuarios a jugar más y aumentar su compromiso con la página web.

Además, se podría mejorar el aspecto visual de la página, agregando más efectos y animaciones para hacer que la experiencia de juego sea más atractiva, por ejemplo, se podría agregar alguna transición al cambiar de juego o al ganar.

En general, hay muchas posibles mejoras que se podrían hacer para esta página web, y sería importante evaluar cuáles serían las más importantes y relevantes para los usuarios y el objetivo del sitio.

## **9. BIBLIOGRAFÍA**

Para este trabajo utilice diferentes páginas web

- **Codecademy** - <https://www.codecademy.com/>
- **FreeCodeCamp** - <https://www.freecodecamp.org/>
- **Stack Overflow** - <https://stackoverflow.com/>
- **GitHub** - <https://github.com/>
- **CSS-Tricks** - <https://css-tricks.com/>
- **W3Schools** - <https://www.w3schools.com/>
- **MDN web docs** - <https://developer.mozilla.org/es/>