**Título:** Diseño y Desarrollo del Juego de Tablero

# Introducción:

Este documento describe las decisiones de diseño y los desafíos encontrados durante el desarrollo del juego de tablero. El objetivo principal era crear un juego utilizando HTML, CSS y JavaScript.

# Inicio del Juego

## Eventos iniciales:

* Al cargar la página (DOMContentLoaded), se ejecuta la función inicio, que configura los elementos iniciales del juego y enlaza eventos a los botones.

## Formulario de validación:

* Se utiliza un formulario para ingresar el nombre del jugador, validando que tenga al menos cuatro letras y no contenga números. Si el nombre no cumple los requisitos, se muestran mensajes de error específicos en pantalla.

## Decisiones tomadas:

* **Validar el nombre**: Se priorizó asegurar que el jugador tenga un nombre válido para mejorar la inmersión en el juego.
* **Separar mensajes de error**: Se eligió manejar errores específicos para dar una mejor retroalimentación al usuario.

# Creación del Tablero

## Método: crearTablero

* Se crea dinámicamente una cuadrícula de 10x10 utilizando una tabla HTML. Cada celda tiene coordenadas (data-x y data-y) que permiten identificar su posición.
* La posición inicial del héroe se marca en la celda (0, 0), mientras que el cofre se ubica en (9, 9).

## Decisiones tomadas:

* **Uso de atributos personalizados (data-x y data-y)**: Permiten identificar rápidamente las celdas y manejarlas de manera eficiente sin necesidad de cálculos complejos.
* **Posición inicial y final fija**: Facilita el control del juego y asegura un objetivo claro para el jugador.

# Interacción del Jugador

## Tirar el dado:

* Al presionar el botón, se genera un número aleatorio entre 1 y 6. Este número representa las posibles posiciones a las que el héroe puede moverse.

## Mostrar movimientos posibles:

* Se resaltan las celdas accesibles desde la posición actual del héroe, calculando los movimientos hacia arriba, abajo, izquierda y derecha dentro de los límites del tablero.

## Decisiones tomadas:

* **Número aleatorio controlado**: Se eligió limitar los valores del dado para mantener el juego equilibrado y dinámico.
* **Celdas resaltadas**: Se usa una clase específica para identificar los movimientos posibles visualmente, facilitando la experiencia del usuario.

# Movimiento del Héroe

## Método: moverHeroe:

* El jugador hace clic en una celda resaltada para mover al héroe. Se actualiza la posición del héroe y se verifica si se alcanzó el objetivo (el cofre).

## Decisiones tomadas:

* **Validación del movimiento**: Se asegura que el héroe solo pueda moverse a las celdas resaltadas, evitando movimientos inválidos.
* **Actualización visual y lógica**: La posición del héroe se refleja tanto en el DOM como en la variable posicionHeroe.

# Verificar Victoria

## Método: verificarVictoria:

* Comprueba si el héroe llegó a la posición (9, 9). Si es así, se desactiva el botón para tirar el dado y se verifica si el jugador estableció un nuevo récord.

## Decisiones tomadas:

* **Uso de localStorage**: Se almacena el récord en el navegador, permitiendo mantener la información entre sesiones.
* **Mensajes personalizados**: El jugador recibe mensajes diferentes si supera o no el récord actual, haciendo el juego más competitivo.

# Manejo de Eventos

## Diseño modular:

* Los eventos, como clics o envíos de formulario, están distribuidos en funciones específicas para garantizar que cada componente del código sea fácil de entender y modificar.

## Decisiones tomadas:

* **Evitar recargar la página**: Se usa preventDefault en eventos como submit para evitar interrupciones en la experiencia del juego.
* **Desacoplar funciones**: Cada función tiene una tarea clara y específica, facilitando su mantenimiento.

# Problemas Encontrados

* **Uso de LocalStorage: En LocalStorage me he encontrado problemas al implementarlo en mi código, tanto para guardar información como para luego trabajar con ella y mostrarla. Por este motivo es que el código en la parte que le corresponde flojea un poco mas.**