Applicant Technical Test

Una vez resuelto el test, se debe enviar el código fuente con los comentarios necesarios para correr el juego a los siguientes contactos: dario@spieldev.com y fernando@spieldev.com.

Objetivo.

El objetivo del presente test de aptitud es comprobar la manera en la que el aspirante al puesto de desarrollador resuelve el problema planteado.
Se tendrán en cuenta diversos aspectos técnicos al momento de evaluar el desarrollo. A saber:

- Comprensión de la consigna.
- Nivel técnico.
- Calidad del código escrito.
- Bugs y/o excepciones.
- Tiempos de entrega.

Consigna.

Se debe desarrollar un juego de slot que cumpla con los siguientes requisitos:

- El juego debe ser jugable desde un browser desktop o mobile.
- El juego debe desarrollarse en HTML5 Canvas. Preferentemente, pero no excluyente, utilizando alguno de los siguientes frameworks: PixiJS, Phaser, CreateJS.
- El juego debe mostrar un preloader de los assets del mismo.
- Una vez cargados los assets del juego, el usuario debe poder realizar un spin, y ver el resultado de la jugada.
- El usuario debe poder volver a jugar luego de mostrarse el resultado de la jugada.

Flow del juego.

1. Se lanza el juego en un browser.
2. Se piden los reels a la api.
3. Se piden las paylines a la api.
4. Se pide el paytable a la api.
5. Se realiza un spin (giro de reels) presionando el botón **spin**. El spin debe durar en promedio 3 segundos desde que comenzaron a girar los reels hasta que se detienen.
6. Se detienen los reels mostrando el layout sorteado.

7. En el caso de haber premios, se muestran las líneas ganadoras sobre los reels y el total ganado en la ventana correspondiente.
8. Si hubieran líneas ganadoras, quedan resaltadas hasta el siguiente spin. Al volver a clickear el botón spin, deben desmarcarse.
9. Se deben poder repetir los pasos desde el punto 5.

API. Descripción y modo de uso.

La API necesaria para este desarrollo es un archivo javascript que debe incluirse en el html que lanza el juego. La API se auto-inicializa, por lo cual basta con sólo incluir el .js en el documento html. Por ejemplo:
<script type="text/javascript" src="lib/api.js"></script>

La API cuenta con una serie de métodos mediante los cuales se simula la interacción con una plataforma de casino.

Para hacer referencia a la API debe invocarse el objeto *wrapper*. Por ejemplo:
wrapper.getReels();
wrapper.spin();

Métodos requeridos.

getReels():Array<Array<number>>

El método *getReels* devuelve una matriz con los 3 reels (columnas de símbolos) que componen el juego. Estos símbolos son los que deben utilizarse en cada reel al momento de iniciar una jugada; es decir, no se puede mostrar un símbolo que no se haya definido para ese reel.

getPaylines():Array<Array<number>>

El método *getPaylines* devuelve una matriz con las 5 paylines que contempla el juego. Cada payline está compuesta por una celda de cada reel. Si hubiera una cantidad mínima de símbolos iguales en esas celdas (en este caso 3), se considera que es una línea ganadora.
Las celdas se numeran de la siguiente manera:
0 1 2
3 4 5
6 7 8

Líneas de pago:
- Línea 1 (horizontal medio): celdas 3, 4 y 5.
- Línea 2 (horizontal superior): celdas 0, 1 y 2.

- Línea 3 (horizontal inferior): celdas 3, 4 y 5.
- Línea 4 (diagonal descendente): celdas 0, 4 y 8.
- Línea 5 (diagonal ascendente): celdas 6, 4 y 2.

getPaytable():Array<Object>

El método *getPaytable* devuelve un listado de objetos que define cuánto paga cada combinación de símbolos. En este caso la cantidad mínima de símbolos necesarios es 3.
Cada elemento del listado contiene los siguientes datos:

- *symbol: string;* es el identificador del símbolo.
- *prize: number;* es el premio que paga el símbolo con 3 ocurrencias.

Nota: No es estrictamente necesario llamar a este método, ya que la API se encarga de calcular las líneas ganadoras.

spin():Object

El método *spin* debe llamarse al momento de generar una jugada nueva. El usuario presiona el botón **spin,** que da comienzo al giro de los reels, e inmediatamente debe llamarse a éste método.
El método *spin* devuelve un objeto con los siguientes datos:

- *stopPoints: Array<number>;* son las posiciones en que debe detenerse cada reel.
- *layout: Array<string>;* es el layout sorteado. Se muestran todos los símbolos sorteados de corrido, siendo el primero el correspondiente a la celda 0 y el último el correspondiente a la celda 8.
- *reelsLayout: Array<Array<number>>;* es el layout sorteado. A diferencia del método *layout* se recibe separado por reels (columnas). Sirve para comprobar de manera rápida cómo debe quedar conformado el layout en pantalla.
- *prizes: Array<Object>;* son las líneas ganadoras. El objeto de cada elemento del array contiene los siguientes datos:
    - *lineId: number;* es el id de la línea que se está pagando.
    - *symId: string;* es el identificador del símbolo que se está pagando.
    - *winnings: number;* es el premio pagado para esa línea.
- *winnings: number;* es la suma de los pagos de todas las líneas ganadoras.