|  |  |
| --- | --- |
| **Week 1 – 17 Ene** | **Sprint Planning (Sprint 2)** |
| **Week 2 – 24 Ene** | **Review (Sprint 1)** |
| **Week 3 – 31 Ene** | **Sprint Planning (Sprint 3)** |
| **Week 4 – 7 Feb** | **Review (Sprint 2)** |
| **Week 5 – 14 Feb** |  |
| **Week 6 – 21 Feb** | **Review (Sprint 3)** |

**STORY 0: CALIDAD DE CÓDIGO**

* Código comentado, explicando bien que hace cada función +1 punto por story
* Optimización de código: + 3 puntos por story
  + Se debe seguir las buenas prácticas de javascript indicadas en la siguiente página:
  + <https://dev.opera.com/articles/javascript-best-practices/>
  + Además se debe tener en cuenta que:
    - No haya variables no declaradas
    - Variables globales bien usadas, no usar si no son necesarias.
    - Uso de funciones: código separado en funciones
    - Código limpio, sin variables que no se usan, ni funciones o código comentado
* Uso de Typescript: + 1 punto por story

**STORY 1: PÁGINA DE INICIO DEL JUEGO. ELECCIÓN DE TIPO DE DIFICULTAD (10 PUNTOS)**

En la página de inicio del juego el jugador deberá indicar su nombre y la dificultad que desea para el juego. Habrá 4 opciones de dificultad:

* fácil, jugará con 6 cartas, y tendrá un tiempo máximo de 2 minutos
* intermedio, jugará con 8 cartas y tendrá un tiempo máximo de 2 minutos
* difícil jugará con 10 cartas y tendrá un tiempo máximo de 3 minutos
* muy difícil jugará con 14 cartas y tendrá un tiempo máximo de 3 minutos

Habrá un botón Jugar que iniciará el juego.

El botón jugar deberá llamar a una función iniciarJuego, que llamará a su vez a 3 funciones:

1. IniciarCartas

creará un array de 6, 8, 10 o 14 cartas, en función del número indicado por el jugador.

El array contendrá objetos de tipo carta. Cada objeto de tipo carta deberá tener la siguiente información:

* ID: id de la carta para identificar si es igual a otra, dos cartas con el mismo id serán iguales.
* IMAGEN: src de la imagen con la que se corresponde la imagen al darse la vuelta.
* GIRADA: será un booleano, que almacene la información true si está girada, y false si está oculta. Por defecto todas estarán en false

La posición de la carta, será su posición en el array.

Consejos: para el src de la imagen, lo ideal es que coincida el nombre de la imagen con su id, así al inicializar el array, tendrán el mismo id y nombre de la imagen, por ejemplo id: 1, su imagen sería 1.jpg

1. IniciarTemporizador

Iniciará el tiempo a 2 o 3 minutos en función de la dificultad elegida por el jugador.

Tras esto, se llamará a las funciones, mostrarTablero y mostrarTemporizador, que se van a realizar en las siguientes stories.

**STORY 2: TEMPORIZADOR REGRESIVO (18 PUNTOS)**

Al iniciar el juego, se mostrará el nombre del jugador, y un contador con el tiempo que le queda.

Para ello, se debe crear una función iniciarTemporizador la cual debe usar la función setInterval que permite llamar a una función cada cierto tiempo. En este caso se debe llamar cada segundo. El jugador verá en todo momento los minutos y segundos que le quedan de juego, podéis seguir el siguiente ejemplo para llevar a cabo el temporizador:

<https://programadorwebvalencia.com/Javascript-cuenta-atras-o-contador-regresivo/>

Si el total de segundos es igual a 0, se debe finalizar el juego. (se debe llamar a la función finalizar juego)

**STORY 3: INICIAR TABLERO (12 PUNTOS)**

Tras elegir la dificultad, se mostrarán el tablero con el número de cartas elegidas. Para ello:

Crear una función IniciarTablero que recibirá el array creado de cartas.

1. Llamar a una función llamada ordenaleatorio, que ordenará las cartas de forma aleatoria
2. Llamar a la función mostrar cartas, que recorrerá el array de cartas y las mostrará todas boca abajo. Cada carta deberá tener un id con su id, y un src con la imagen que le corresponde. Además de una función onclick que llame a mostrarCarta con su posición, que coincidirá con su posición en el array.
3. Las imágenes deben ser responsive usando flexbox igual que en el anterior.

**STORY4: MOSTRAR CARTA (8 PUNTOS)**

Mostrar carta: recibirá la posición de la carta que se ha clickado. Se actualizará el atributo girada de la carta que se le ha pasado. Y se mostrará dicha carta en el tablero. Se eliminará también la funcion onclick de esa carta mostrada, para que si el usuario vuelve a clickar sobre ella, no llame a ninguna función.

Desde aquí se llamará a la funcion comprobarCartasGiradas. (se detalla en la story 6)

**STORY5: OCULTAR CARTA (8 PUNTOS)**

Ocultar carta: recibirá la posición de la carta que se quiere girar. Se actualizará el atributo de la carta que se le ha pasado y se ocultará dicha carta en el tablero. Deberá tener la función onclick para que pueda volver a pulsarse con la posición que se le ha pasado por parámetro.

**STORY6: COMPROBAR CARTAS (8 PUNTOS)**

Actualice la función comprobar carta, para que pasada la posición de 2 cartas compruebe si son iguales o no.

Si ambas cartas son iguales, devolverá true, y si son diferentes, devolverá false.

Esta función sólo debe hacer eso, **importante**. El resto se debe hacer en comprobarcartasgiradas (story 6)

**STORY7: COMPROBAR TOTAL CARTAS GIRADAS (12 PUNTOS)**

Se deberá también llamar a la función cartasGiradas, la cual debe realizar lo siguiente:

* Debe tener un array que guardará la posición de la carta girada. Si el array es igual a 2, es que se han girado 2 cartas y llamará a la función comprobar cartas. Si no es igual a 2, no hará nada.
* Tras la llamada a comprobar cartas, tendrá una variable true si son iguales, o false si son diferentes.
* Si las cartas son iguales:
  + Llamará a la función sumarPuntos, esta función sumará 2 puntos al marcador
  + Tras esto llamará a la función creada en el sprint anterior, comprobarJuegoFinalizado (la cual actualizaremos en la siguiente story)
* Si las cartas son diferentes:
  + Llamará a la función ocultarCartas que girará de nuevo las 2 cartas que no son iguales.
* En ambos casos, se vaciará el array de cartas giradas, para seguir jugando con otras 2 cartas diferentes.

**STORY8: COMPROBAR JUEGOFINALIZADO (15 PUNTOS)**

Para comprobar que el juego ha finalizado, se debe actualizar la función comprobarJuegoFinalizado, para que:

Comprobar el total de segundos que le quedan. Si es mayor a 0:

Comprobar el total de puntos, si es igual al total de cartas:

* + Parar el temporizador con la función clearInterval: <https://stackoverflow.com/questions/109086/stop-setinterval-call-in-javascript>
  + Sumar al total de cartas giradas, los segundos que le quedan, y multiplicalo por 1 si es dificultad fácil, por 2 si es media, por 3 si es difícil por 4 si es muy difícil y esta será la puntuación del jugador.
  + Mostrar un mensaje, enhorabuena, ha ganado su puntuación es (y mostrar el total de puntuación)

Si es menor a 0:

Mostrar un mensaje, lo sentimos ha terminado su tiempo.

Mostrar un mensaje con la puntuación del jugador que será el total de cartas giradas hasta ese momento.

**STORY9: MULTIJUGADOR PANTALLA INICIO ( 18 PUNTOS)**

Actualiza el menú de inicio, para que permita jugar a varios jugadores. Para ello:

Actualiza la pantalla de inicio para que se pueda elegir la dificultad y el número de jugadores.

En función del número de jugadores, se mostrará un formulario donde se debe introducir el nombre o nick de cada jugador. Estos jugadores se guardarán en un array de objetos jugador. Cada objeto jugador deberá guardar el nombre y la puntuación del jugador.

A continuación, se deberá mostrar el juego completo para cada uno de los jugadores introducidos, es decir:

Se recorrerá el array de jugadores, y se llamará a la función iniciarJuego.

En la función finalizarJuego, se deberá guardar la puntuación del jugador en el array.

Al finalizar el juego completo todos los jugadores, se llamará a la función mostrarRanking, que recorrerá el array de los jugadores con sus nombres y puntuaciones y los mostrará en orden de mayor a menor puntuación.