[GRUPO 5]

Ángel Bachiller García

Álvaro Barroso Mato

Miguel Ángel Galaz Sánchez

Juan Daniel Silva Mora



# Roles:

**HTML**: Álvaro

**CSS**: Juan

**Diseño y Portfolio**: Ángel

**JSON**: Miguel

# Descripción:

Se trata de un laberinto en primera persona, donde el jugador podrá moverse en varias direcciones. Durante su avance por el laberinto, se encontrará con diversos puzles (a cada cual más difícil en función de la dificultad seleccionada al comienzo del juego), los cuales deberá resolver para poder seguir avanzando y de este modo, finalmente, encontrar el tesoro del laberinto.

# Objetivos propuestos y cumplidos:

La mayoría de objetivos propuestos al principio del proyecto se han podido cumplir. Uno de los que no hemos conseguido cumplir ha sido el poner varios minijuegos en cada sala, debido a su largo desarrollo tanto en diseño como en programación, con lo que se decidió limitar a un minijuego al final del laberinto para conseguir abrir el tesoro, y varios durante el camino en forma de cuestionarios.

# Programación del HTML y CSS:

Agrupamos estas categorías, ya que no podemos hablar de uno sin el otro. Para la realización del HTML se han dividido sus elementos en diferentes **<div>**, con el fin de poder trabajar más cómodamente tanto a nivel de CSS como JS. Además, para llegar a la optimización de ambos, hemos agrupado ciertos elementos con propiedades similares (tales como las flechas de dirección utilizadas en el recorrido del laberinto) en clases comunes para así poder reducir el número de líneas de código. También debemos destacar el uso de Bootstrap en nuestro menú **Contactar**, pudiendo dividir nuestras fotos y enlaces en distinto número de columnas.

Pasando al CSS, y a bien de mejorar la interacción del usuario con nuestro juego, ya sea desde un dispositivo móvil o un ordenador, los elementos que se muestran en la escena se rigen no por medidas inalterables como pueden ser los pixeles, sino por porcentajes en función de la pantalla, o de los elementos que lo contengan.

# Realización del JSON y Mecánicas:

Hemos pensado en hacer toda la base del juego con Json, ya que es más rápido que XML , no obstante hemos utilizado un convertidor para ello, es decir, haciendo las salas y demás elementos en XML ya que estaba más organizado con el sistema de etiquetas <> que un Json.

El sistema es muy sencillo, tienes salas, que actúan como estados y en función de la dirección que pinches vas a una sala u otra (un estado u otro), metiendo salas de transición para simular el falso movimiento del laberinto.

Pensamos que cargar el laberinto así es un sistema mecánico, ya que hace que la aplicación fuera como la consola y los Json los cartuchos, es decir; es un sistema en el cual si el Json sigue la misma estructura, podrías cargar cuantos y cuan largos mapas quisieses, frente a cualquier otro sistema, ya que podría implicar más líneas de código y menos optimización.

Las preguntas son variadas y conocidas, ya que son un mero elemento que actúa de enemigo en nuestro juego, al no tener estos como tal, debido a que solo servirán para quitarte tiempo si fallas, teniendo preguntas incluso que no llevan a ninguna sala con el mero objetivo de que falles y no completes el nivel. Al ser de diferente temática puede hacer que cualquier jugador pueda responder algunas o todas, independientemente de su conocimiento previo.

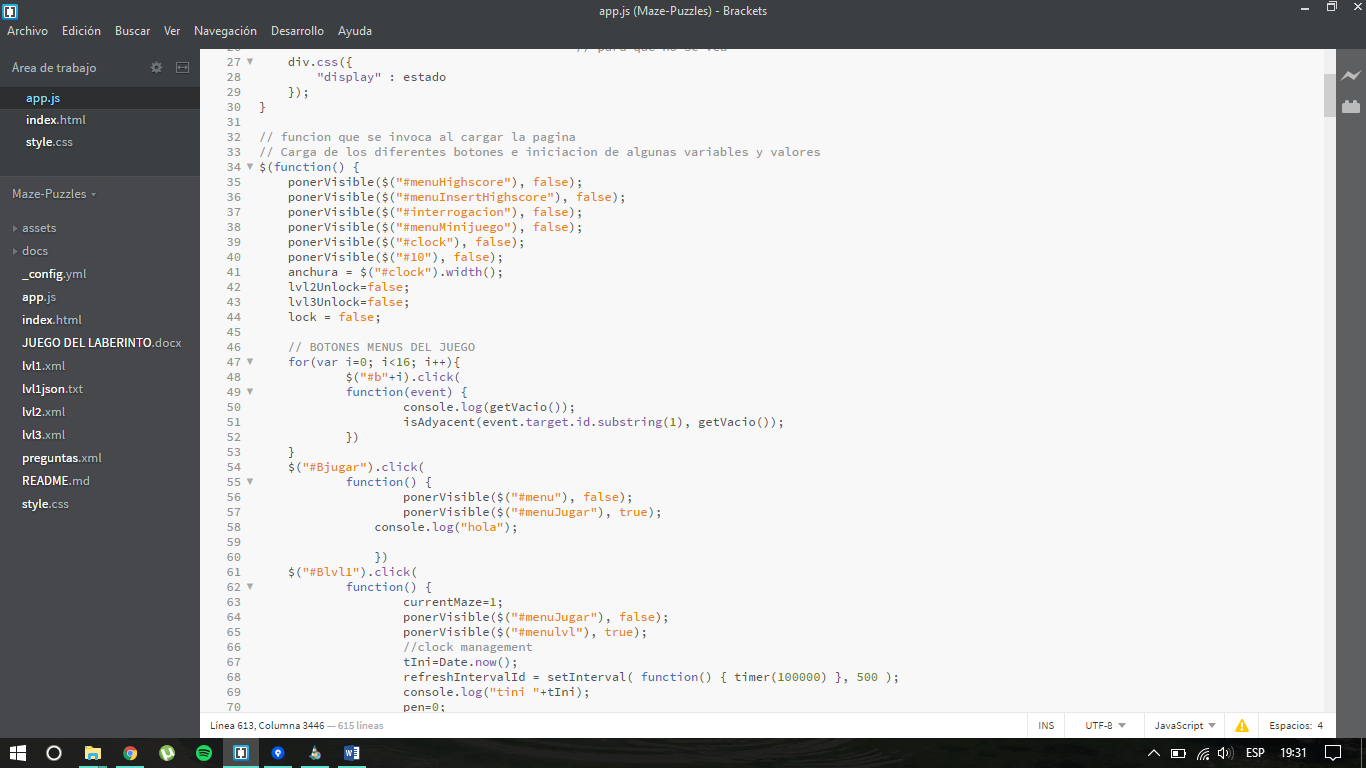
Como ya he mencionado el tiempo es otro enemigo a tener en cuenta, ya que si este acaba, el juego concluirá, pese a que te deje intentarlo de nuevo.

Para determinar que jugadores son mejores, lo hacemos en función del tiempo que han tardado en completar el laberinto, ya que al ser las preguntas elementos de resta del tiempo, es proporcional al tiempo que tardes, es decir a más preguntes falles peor será tu resultado. En conclusión, el tiempo es una forma de resumir tu habilidad por las salas recordando por cuales has ido o cuales no y la cantidad de preguntas que aciertes para no perder el tiempo, además de un factor suerte al coger los caminos correctos a la primera.

Hemos incluido una etiqueta asset dentro del XML en algunas salas, no como mero hecho decorativo, sino a modo de indicador de porque salas has pasado, ayudándote así a completar el nivel y tener compensado el equilibrio entre dificultad y frustración y que dejes de jugar al juego por esta última.

# Programación de JavaScript:

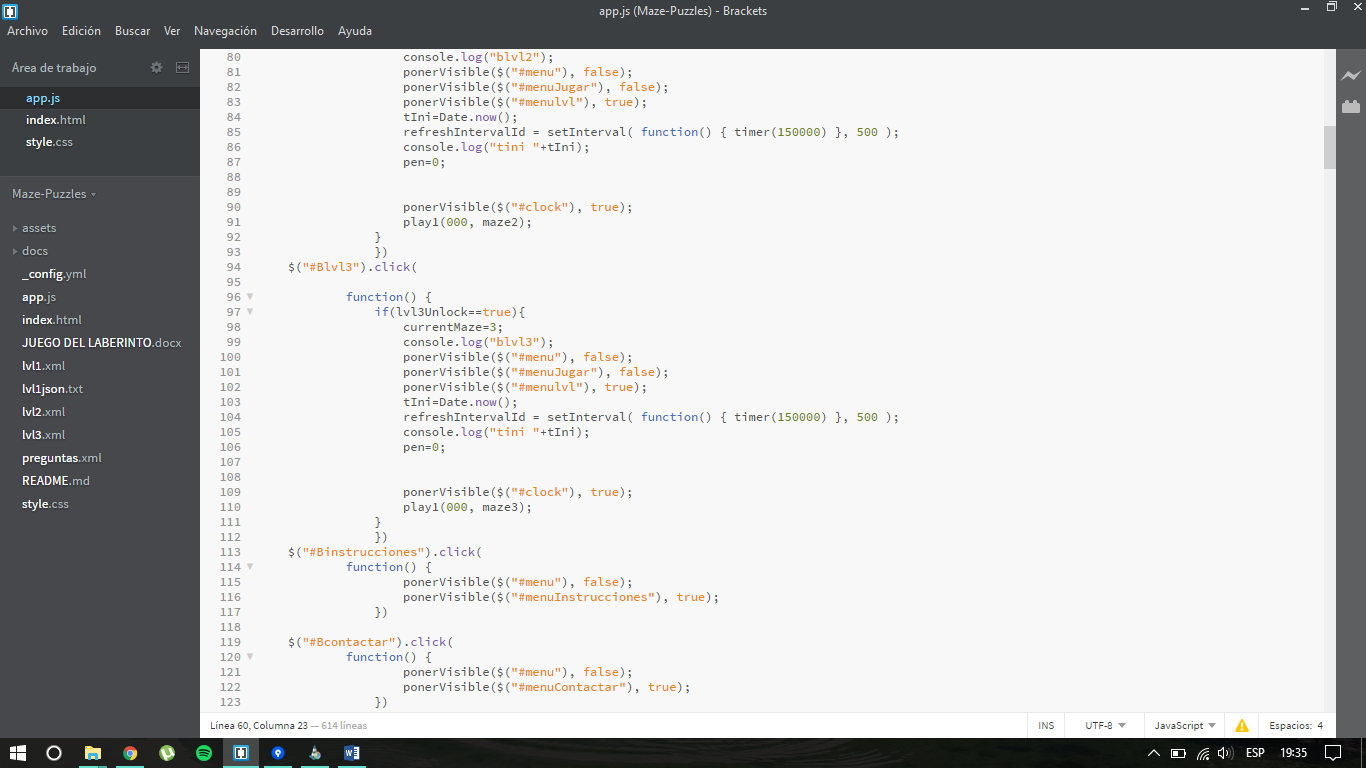
Para explicar mi parte dividiré el código según sus funcionalidades.



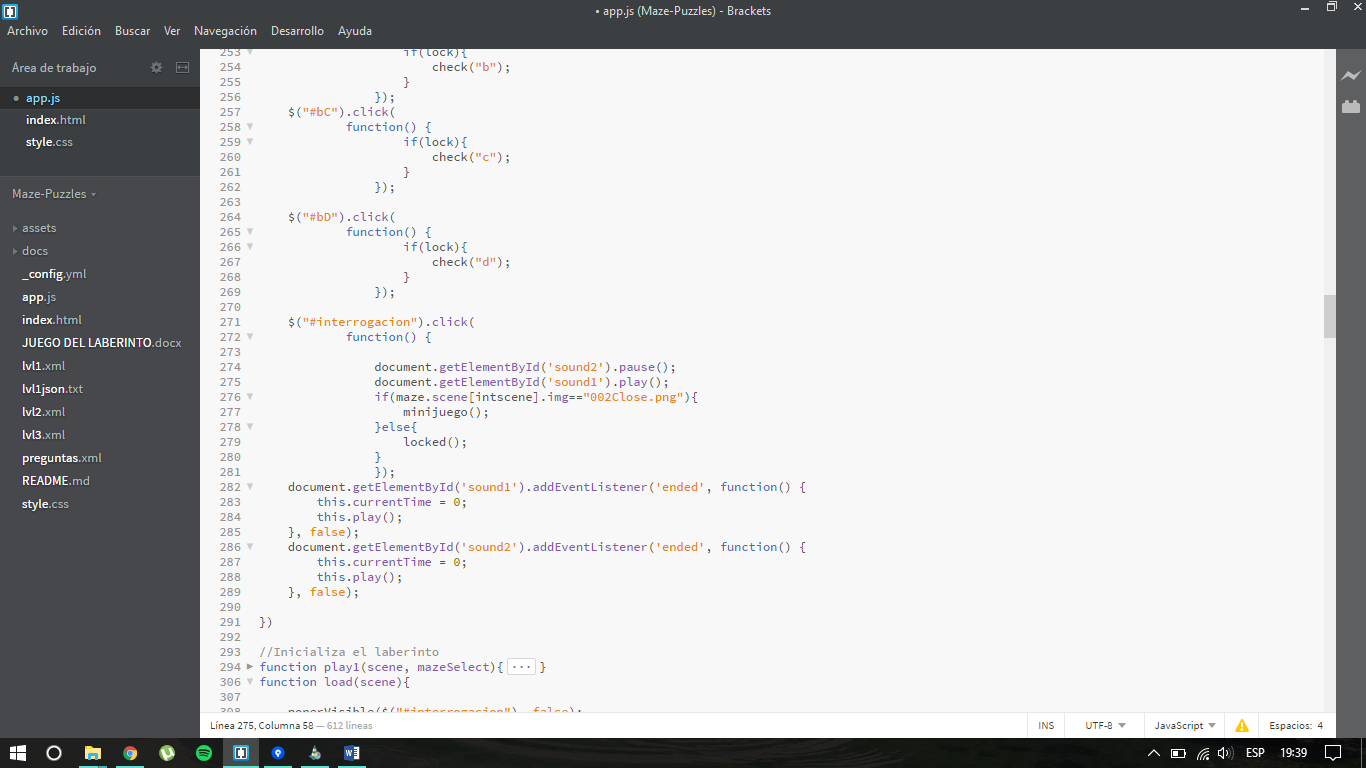
Las primeras sentencias se aseguran de que todos los elementos que no deseamos que aparezcan en primer instante se oculten, y se inicializan algunas de las variables.

A continuación, se procede a la declaración de los listeners de los botones:

En el primer caso se declaran los botones del minijuego final de cada nivel, en los que se realiza una llamada a un método que se explicara más abajo.

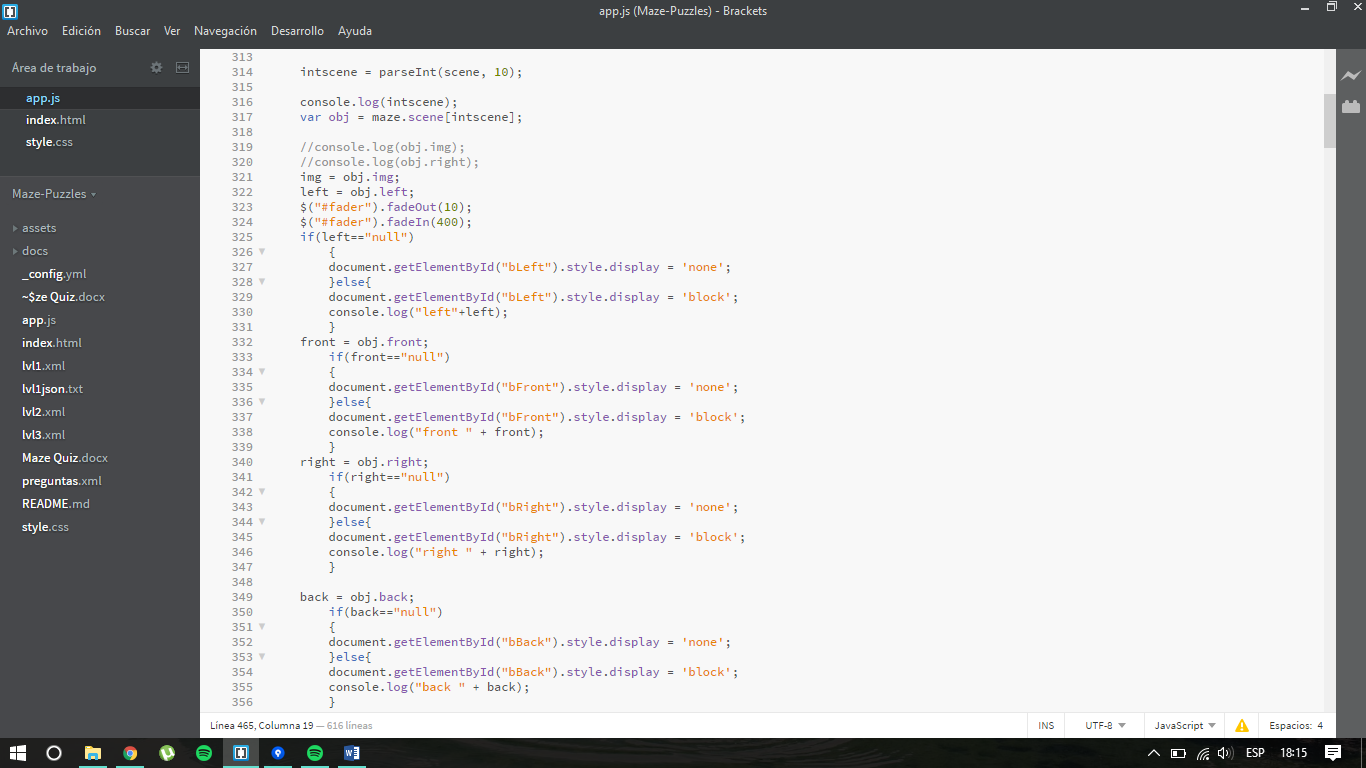


También se declara cada botón de acceso a cada nivel, para acceder a ellos deben de estar desbloqueados con las variables lvlxUnlock. En este fragmento se reinicia el reloj se ocultan capas indeseadas y se llama al constructor de la partida(*play1*) con los datos del nivel.



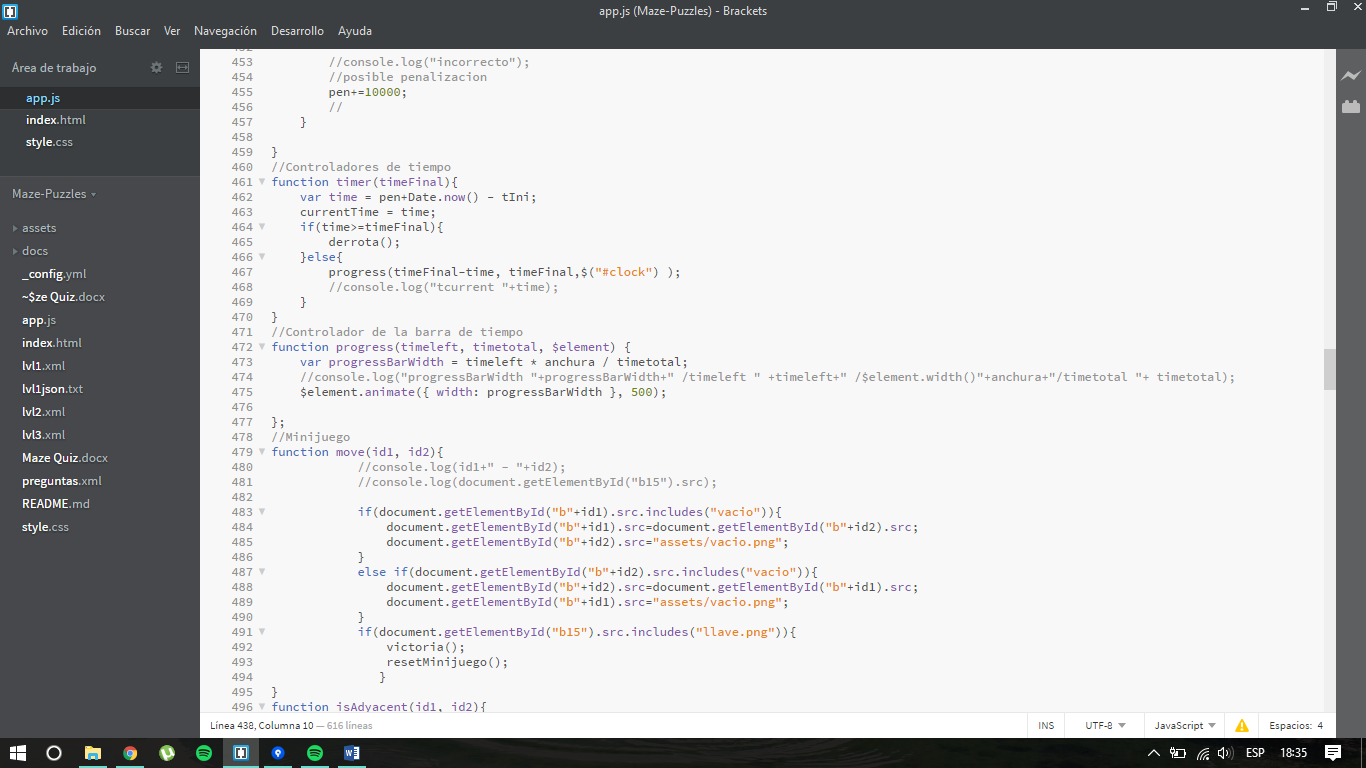
En esta sección de código se instala parte de la funcionalidad del sonido, en la cual se aplica la reproducción el bucle y se impone el cambio de sonido desde el sonido ambiental del laberinto al de las preguntas o minijuegos.

El Cargado del nivel se realiza en la función *Play1*, en la cual se parsea el JSON que contiene los niveles y las preguntas, y se carga la primera escena en pantalla.

Esta es la función *load* que sirve para cargar cada escena en pantalla además de interpretar y extraer los datos que corresponden a estas. Obtiene el diseño de la escena, además de sus salidas, los posibles adornos, minijuegos y escenas a las que llevan sus salidas.

*Prepregunta* Muestra la escena junto con una interrogación para pulsar y acceder a el acertijo o minijuego.

*Lock* pasa a la pantalla de pregunta, mostrando las respuestas y la pregunta junto con sus botones selectores, una vez se pulsa alguno de los botones selectores se llama a *check* que comprueba si la respuesta es correcta, en caso afirmativo carga la puerta abierta (*imgo*), en caso contrario aplica la penalizacion y mantiene la pantalla.



*Timer* es la función que controla el tiempo, esta función comprueba que el tiempo que lleva el jugador en el nivel no es superior a el estipulado como derrota. Esta función se llama cada 0.5 segundos y a su vez usa *progress* para editar la barra roja que indica el estado del tiempo.

Ahora vamos con las funciones del minijuego de extraer la llave. En primer lugar, tenemos *move*, que intercambia las imágenes que ocupan los elementos contenedores, este intercambio solo se produce si son adyacentes los cubos, así que para asegurar esta última condición esta *isAdyacent* que nos aseguran que estén pegados los cubos ya sea vertical u horizontalmente. Para intercambiar cubos (siguiendo las reglas del juego) se debe de escoger un cubo que este pegado a el cubo vacío, para obtener el cubo vacío tenemos *getVacio* que nos devuelve la posición del cubo vacío. Finalmente están *minijuego* y *resetMinijuego* que inicializan la pantalla del minijuego y resetean las posiciones de los cubos respectivamente.

Por último, está el guardado de puntuaciones que consta de 3 funciones: *InsertHighscore* que dispone la pantalla de introducción del nombre del jugador, *checkHighscores* que añade la puntuación actual a los datos guardados localmente del laberinto actual y actualiza la tabla ordenándola y descartando las ultimas posiciones hasta reducirla 10 puestos, *getHighscores* imprime los datos de mayores puntuaciones.

# Diseño:

El diseño escogido para el videojuego ha sido un estilo ráster, utilizando para ello la herramienta PhotoShop, ya que este software da mucha libertad y muchas facilidades a la hora de realizar dibujos.

La perspectiva y zonas escogidas para el laberinto han sido pensadas para dar tanto una mayor sensación de inmersión al jugador, como entendimiento y ubicación en el laberinto. Éste consta de tres fases: inicialmente a las afueras de un castillo, a continuación, accederemos a las mazmorras a través de un túnel, algo más cerca del tesoro. Por último, accederemos al interior del castillo a través de una puerta secreta.

Los assets extra (La pala, el ratón, la pelota…) se han utilizado para dar distintos aspectos a las diferentes salas, dando así a entender al usuario que se ha movido por una zona distinta. El tamaño de éstos, es del mismo que el de las pantallas de los laberintos, para así facilitar la parte de programación.

Todo el diseño sigue una estética parecida, con colores grises, blancos y marrones.

# Portfolio:

El portfolio contiene un carousel en la página de inicio con los dos juegos desarrollados por el grupo (uno todavía en fase de desarrollo), en los cuales se puede acceder mediante los botones, situados debajo de la imagen, a la información del juego, o al mismo juego. También hay una pestaña, en la barra de navegación, que contiene la información de contacto de los componentes del grupo.

Link al portfolio : [https://grupomaajwebyredessociales.github.io/Portfolio/#](https://grupomaajwebyredessociales.github.io/Portfolio/)

# Secuenciación del videojuego:

# Secuenciación de las pantalla:

El menú principal contiene 4 botones: uno para jugar, otro para las instrucciones, los contactos y para ver las mejores puntuaciones.

El botón de jugar te lleva a la elección de un laberinto/nivel, de los cuales solo se podrá acceder al primero, estando el segundo y el tercero bloqueados hasta que finalices progresivamente los distintos niveles.

