***Manual de código*:**

**Wordlabras**

|  |
| --- |
| Logotipo  Descripción generada automáticamente |

*Realizado por:* Adrián López Pastor

Contenido

[HTML 3](#_Toc121938132)

[Index.html 3](#_Toc121938133)

[Wordlabras.html 4](#_Toc121938134)

[Registro.html 5](#_Toc121938135)

[Stats.html / rank.html / ayuda.html 6](#_Toc121938136)

[CSS 7](#_Toc121938137)

[Login.css 7](#_Toc121938138)

[Main.css 13](#_Toc121938139)

[JavaScript 16](#_Toc121938140)

# HTML

## Index.html

En el index.html tendremos el formulario de login con acceso al formulario de registro y olvidar contraseña. En dicho HTML lo primero que veremos serán enlaces de ayuda por medio de “link” para enlazar el CSS y bootstrap y “script” de Bootstrap además de JQuery.

Texto

Descripción generada automáticamente

La estructura que seguirá será de un div que bajará con efecto donde guardaremos todo el contenido, luego dentro de ese div padre de todos, tendremos un div con el logo de la aplicación y otro div con el formulario dentro. Antes del formulario dejamos un div en caso de querer meter o implementar una imagen o algo externo al formulario pero en éste caso estará en desuso. Una vez entramos al formulario tendremos 3 tipos de inputs, el del usuario, contraseña y el que envía los datos que será para acceder al juego. Debajo del formulario tendremos otro Div padre de dos enlaces, uno para registrarse como usuario en caso de no tener cuenta y otro por si olvidaste la contraseña (queda por desarrollar a si que lo hice sencillo con un alert para simular un caso real aunque poco práctico diciendo que envíes un correo solicitando la contraseña).

Texto

Descripción generada automáticamente

Al final tengo implementado el script de axios aunque no sé si al final utilizaré axios o node para la BBDD.

## Wordlabras.html

Éste html y los siguientes que voy a explicar están en la carpeta “HTML”. Éste HTML es el HTML del juego, pese a estar muy vacío para todo lo que tiene es porque la mayor parte de las cajas del juego se generan a través de JavaScript de forma interna (esto hace que no sea estático y que se puedan implementar cambios a futuro como que puedas escoger el número de letras que tenga la palabra a adivinar por ejemplo y generar una caja en función de ese número escogido, aunque para hacer esto habría que cambiar parte del código de generado de intentos)

Texto

Descripción generada automáticamente

En el juego tenemos un div entero del juego que tendrá unas medidas para no ocupar por completa la pantalla o sí, dependiendo de la vista, con eso jugaremos para la visualización. En él encontraremos el header que es la cabecera del juego, ésta es la que contiene lo botones para visualizar las opciones, en éste caso serían: Ayuda (completamente inútil actualmente, se haría un html con la explicación del juego sobre cómo jugar), el título del juego, y a la derecha cuenta con un botón para consultar las estadísticas en el cual tengo una función porque pensé en hacerla como un iframe o una ventana que está invisible pero al darle click se visualizará, pero no está realizada, la otra opción es que te redirija a la página de estadísticas. La siguiente es el ranking que visualizará la página con un ranking sacado de la base de datos. Cada botón tiene un emoticono/imagen para visualizar qué es cada cosa, si bien es cierto que pienso que tanto el de estadísticas y ranking quizás a primera vista y sin saber nada puede que no sean tan claros como me gustaría (saber qué son sólo con ver la imagen, es decir, que sean intuitivos).

Tras el header tenemos el “main” que es donde tenemos el juego básicamente, ahí se generarán las cajas de los intentos por medio de JavaScript y el teclado en el div de teclado.

Debajo tengo un div oculto de “stats” que como digo, mi idea inicial era tener una ventanita oculta con las estadísticas para luego consultar BBDD y visualizarla.

Tenemos importado al final el javascript que hará todo el proceso para la creación tanto de los intentos como del teclado entre otras cosas.

## Registro.html

La página de registro es muy similar a la del index.html (login y página principal e inicial) a si que no diré mucho más. Viene a ser lo mismo cambiando un par de cosas, como son los inputs de los datos que consultaremos en BBDD y el input submit con una función en “onclick” que hará todo el proceso de comprobación en BBDD y si es correcto el registro del usuario, si no es correcto, lanzará alertas con los posibles fallos.

Tengo importado axios también por si lo utilizo.

Texto

Descripción generada automáticamente

## Stats.html / rank.html / ayuda.html

WIP (Work In Progress). Actualmente no están hechas, pero contendrán lo comentado en los anteriores HTML’s.

Captura de pantalla de computadora

Descripción generada automáticamente

# CSS

## Login.css

Dicho login está sacado de Bootstrap, lo que hace es poner un fondo del html azul cyan (para nosotros invisible ya que lo que se verá será el fondo del body y sus hijos, se hace para diferenciar las capas mejor).

De lo importante a comentar, la estructura, básicamente hacemos un flex en columna y que esté alineado y justificado por el centro (posición centro) ocupando toda la pantalla, así distribuiremos los divs y todo el contenido el formulario en vertical y alineado en el centro de todo lo que ocupa el formulario (que es el 90% de la ventana). Esto creará esa ventana de login siempre alineada en el centro, dispuesta y ordenada de forma vertical.

Texto

Descripción generada automáticamente

El FormFooter serán los botones de “Registrarse” (añadido por mí) y el de “Olvidé la contraseña”, que aparecerán de forma diferente, con otros colores y efectos, por ejemplo tienen borde pero realmente será un color casi invisible para notar que tiene borde poruqe será también un efecto visual. El formulario posee nodos (los cuales me han molestado a la hora de trabajar con el formulario al recoger los datos) vacíos e inactivos para que queden bien vistosos, les deja en un color gris para que quede bonito el espacio entre los inputs del formulario por medio de etiquetas H2 inactivasm también dando efectos de activas por medio de clases.

Texto

Descripción generada automáticamente

Después tendremos lo visual sobre los botones de etiqueta “input” con los tipos correspondientes. Tendrán los valores dados, con efectos proporcionados con los “-webkit-box-X”, los “-X-transition” y el transition. Esto creará un efecto de aparecer y desaparecer con un tiempo estipulado, en este caso el efecto de aparición y desaparición será de 0.3 segundos. También tienen un efecto de cuando dejas el ratón encima de los inputs el color de fondo cambiará a azul claro. El efecto de “Scale” lo que hace es disminuir 0.05 en escala X e Y el botón.

Texto

Descripción generada automáticamente

Lo demás es un poco de lo mismo con los demás inputs (type text) y demás efectos/animaciones con duraciones, formas y modos hasta que llevamos a los “@KeyFrames”. Éstos KeyFrames son los que permiten controlar los pasos intermedios en una secuencia de animación CSS, en el caso de nuestro código sería para la calase “fadeInDown”, realiza una opacidad al 0% inicialmente hasta un 1 en el 100% del efecto (acabado) por ejemplo.

Texto

Descripción generada automáticamente

Podremos ver más efectos



Otros displays

Texto

Descripción generada automáticamente

Y por último unas líneas que añadí a mayores para que se viese el logo de la forma que yo quería estando donde lo puse (fuera del div del formulario), una mejor visibilidad del margen superior ya que el Bootstrap estaba hecho para contener una imagen superior encima del formulario pero dentro del mismo, y en éste caso la quité y la saqué fuera porque quedaba mejor y era otro propósito al inicial. Y por último una MediaQuery que hace que se visualice correctamente en vista móvil/Tablet donde el ancho máximo sea de 1000px. Lo que haré será hacer que ocupe toda la pantalla básicamente, lo demás lo vuelvo a declarar por si no se coge correctamente en algunos navegadores.

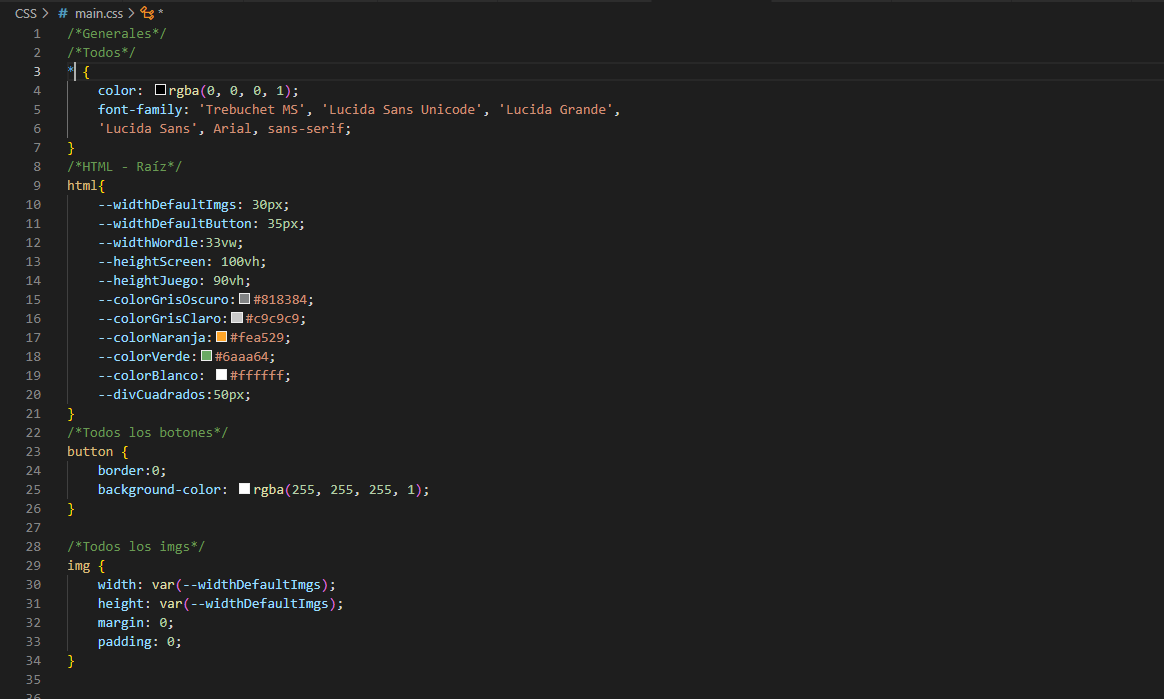
Texto

Descripción generada automáticamente

## Main.css

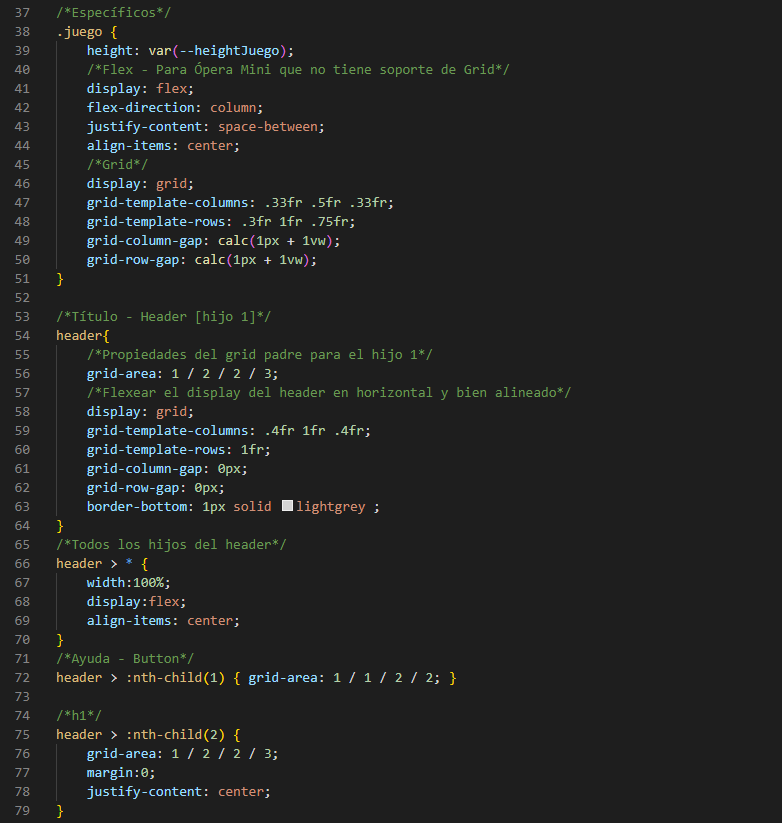
Éste CSS será para la página del juego. Sé que es mala práctica (sobretodo si se hace sin saber) el dar valor a todo con \* pero como lo he hecho a sabiendas, he puesto el color a negro y una fuente parecida al Wordle junto con otras parecidas en caso de que no las detecte el navegador a toda la página web (es decir, ha sido a sabiendas).

En la página he creado variables que utilizaré en el futuro de éste CSS. Los botones tendrán fondo blanco sin borde, las imágenes utilizarán un valor fijo dado por variable de medidas, junto con un margen y padding de 0 (que estén bien juntitas las imágenes sobretodo del header que serán a las que afecte).



Tendremos una clase “juego” que dejará el height en 90vh con una variable y creará la estructura del div del juego con un display flex, se distribuirá de forma vertical y crearemos un grid de 3 estructuras dentro del juego, éstas estructuras serán: Header (33fr), intentos (5fr) y teclado (33fr) con espacios entre ellos de 1px + 1vw para darles espacios dinámicos.

El header se distribuirá con una sección de 4 columnas (una columna ocupando como 2 columnas por lo que tendremos visualmente 3) y 1 fila. Todos los hijos del header ocuparán el 100% (todo lo que puedan) y se distribuirán con display flex y en el centro (es decir, que las 3 columnas ocuparán los “fr” asignados al 100% y se posicionarán en el centro por defecto). Después asigno por encima un grid área para distribuir los espacios de la misma manera que hice pero asignándolo de forma específica por si algún navegador no aceptase display flex pero sí grid (lo cual sería raro pero nunca está de más prevenir).

Texto

Descripción generada automáticamente

El teclado ocupará el 100%, estará centrado y se alineará de forma centrada horizontal y verticalmente. Tendrá una separación por cada hijo de 4px verticalmente. Cada cajita del teclado tendrá también su propio espacio y tendrá una separación de 4px también que hará la separación “horizontalmente” (y hará el efecto de teclado virtual gracias a esto y los 4px del padre de gap). Los divs tendrán botones, les daremos forma, así como al texto para que se vea bien la letra asignada del teclado.

Texto

Descripción generada automáticamente

Tenemos estética de las cajas de intentos, y unos efectos CSS para lo que se supone que iba a hacer inicialmente de unos stats que aparezcan habiendo estado invisibles hasta que acabe la partida como en el wordle original pero actualmente está sin uso ya que la parte de estadísticas no está hecha y no estará hecha así (lo dejo para que se vea cómo se podría hacer)

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Aquí tenemos los estilos para cuando introduzca letras (El “important” es para que sobrepase cualquier valor inicial y tenga prioridad sobre los otros valores dados), los efectos de “Flip” que son los giros de las cajas de los intentos y los MediaQuerys para la Tablet y vista móvil.

Texto

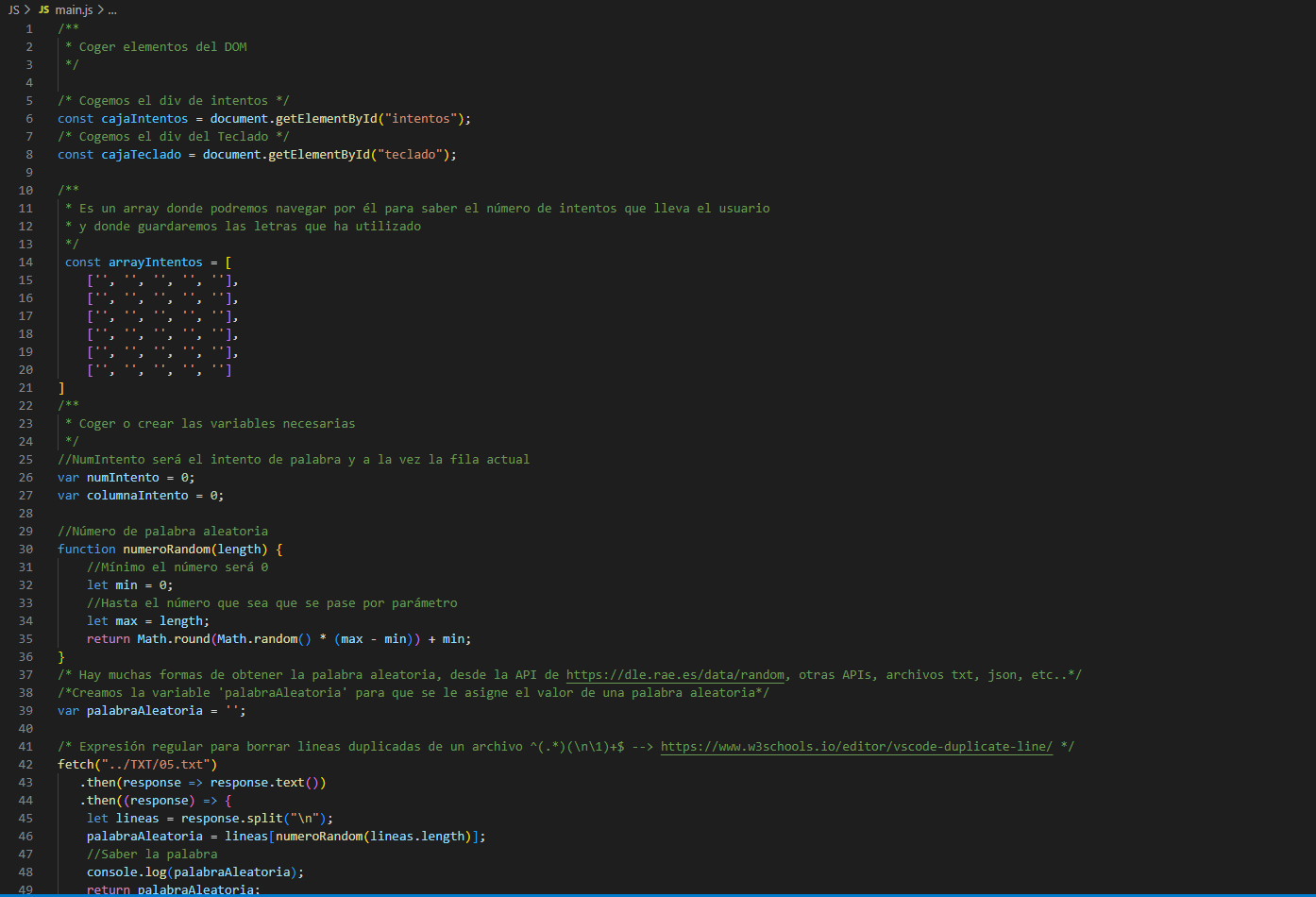
Descripción generada automáticamente

# JavaScript

## Main.js

Main.js será el JavaScript que se encargue del juego (que es la aplicación en sí, por lo tanto es el main). Primero cogeremos los divs de intentos y de teclado para poder generarlos. Creamos un array con los espacios necesarios de los intentos, en éste caso 5x6 (5 letras \* 6 intentos), que serán 6 arrays de 5 caracteres.

Inicializo el intento a 0 y la columna del intento a 0, éstas variables nos servirá para saber en qué intento estamos tanto de letra como de palabra (numIntento será el intento o fila y la columna la letra actual). La función “numeroRandom” será para generar un número aleatorio en función del número que le pasemos por parámetro (en éste caso será el número de letras que contiene el .txt que utilizaremos, pero realmente podría ser un .json como hice inicialmente o con una API utilizar sus funciones). Después tenemos el fetch para generar la palabra aleatoria y tenerla siempre accesible

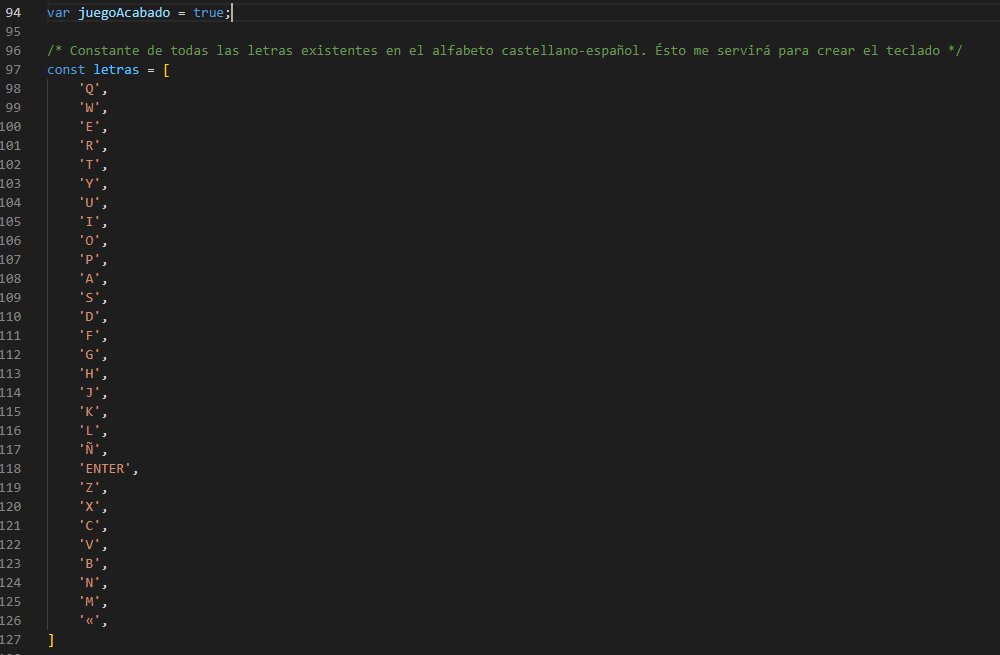


Después he dejado en comentarios la forma inicial que tenía que era con un archivo .json pero tenía pocas palabras y no me acabó de convencer, también tengo puesta la API de la RAE y un repositorio donde hacen uso de ésta API pero en Python y de donde saqué el txt y la propia carpeta para futuras implementaciones

Texto

Descripción generada automáticamente

Creé una variable de juego acabado y en los comentarios explico el porqué de esa variable, aunque actualmente en las capturas no se verá su uso. Luego he creado un Array con todas las letras del teclado y en el orden que está en el teclado para luego generarlo.



Dicho array también lo usaré para comprobar qué letras leerá el evento “onclick” y el del teclado para ver si coincide con las posibles letras del teclado virtual, ya que si no, con el teclado detectará teclas como F12, F1, 2, 3, 4, etc… Y no queremos que pinte en las cajas de intentos esas pulsaciones porque no forman ninguna palabra.

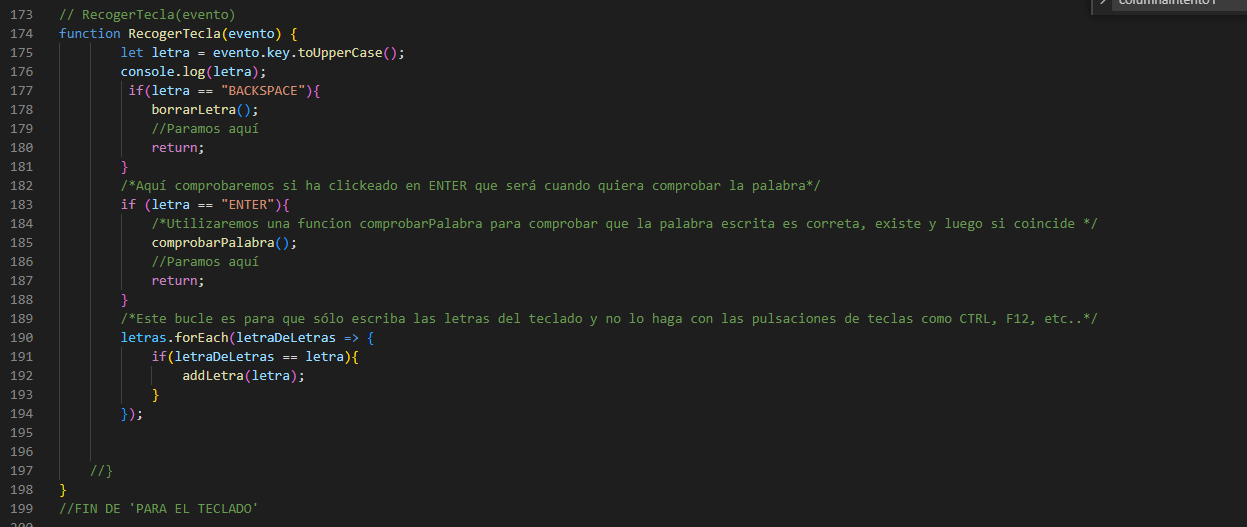
Después tendremos la creación de la estructura de los intentos y su visualización, lo crearemos dándoles atributos como id para luego poder manejarlos. Se han creado con un bucle for-each utilizando el arrayIntentos pero éste podría ser variable como una mejora y que el usuario escogiese el número de letras de la palabra y que se auto-generase éste cuadro de intentos en base a esa elección y no sería muy difícil de implementar (sería hacer un par de cambios).

Después tenemos el contador de botones creados, para poder simular un teclado real y que no ponga todos los botones en la misma fila (en éste caso, en el mismo div). Mientras los creamos también les damos los eventos de click, no lo haremos con el teclado porque si lo hacemos así, el usuario deberá dar click en el botón para tener el “focus” de la tecla virtual y poder detectar dicha pulsación del teclado real. Tendremos que dejar la lectura de teclado en “document” para que lo lea siempre la pulsación del teclado y que sea más amena la experiencia del usuario.

Texto

Descripción generada automáticamente

Posteriormente tendremos la función utilizada en el evento de ‘keydown’ que sería la pulsación, donde la función será “recogerTecla”, que lo que hará será comparar la tecla pulsada con las de ENTER o borrado para hacer una función y otra (enviar la palabra o borrar letra) o compararla con las letras del array del teclado como comenté en dicho apartado, para que no detecte pulsaciones o teclas como “F12” por ejemplo.



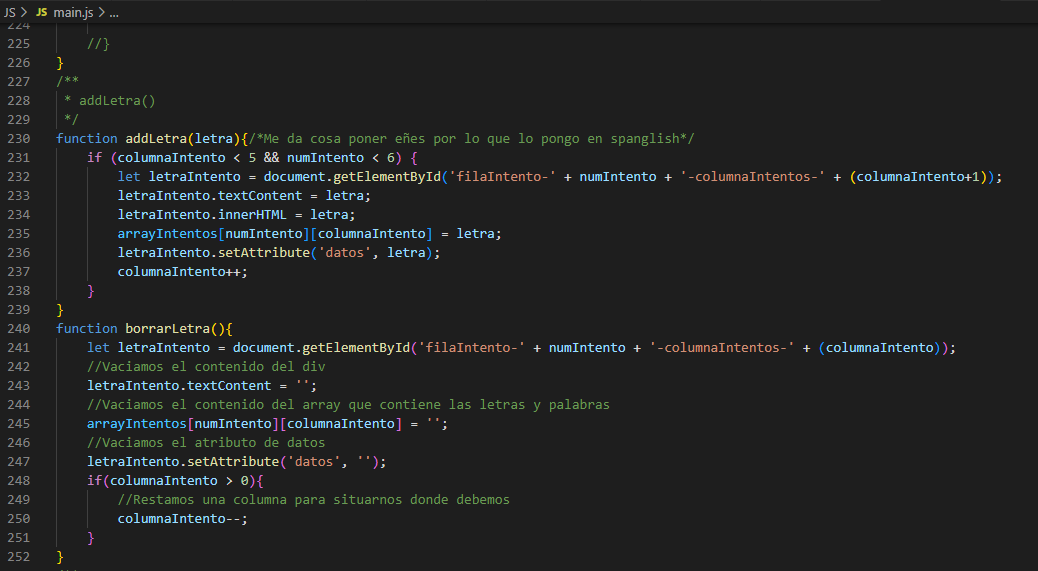
Después tendremos una función muy parecida pero con el evento click en cada letra del teclado virtual generado, no necesitamos recorrer aquí el array con las letras del teclado virtual porque el usuario va a tener que hacer click en cada letra directamente entonces no puede haber fallo con letras no gestionadas (o que no estén en dicho teclado). Cabe decir que el teclado no está preparado para poner tildes por ello mismo. Tampoco son necesarias porque eso podría detectarse a la hora de comparar la palabra sin tilde perfectamente.

Texto

Descripción generada automáticamente

La función “addLetra(param)” será para añadir la letra al div correspondiente (dependiendo de la columna y fila en la que nos encontremos). Usaremos el contador “columnaIntento” para manejar la posición en la fila o intento que nos encontramos. Le he añadido un atributo de ‘datos’ con la letra para un futuro uso con los efectos.

La función “borrarLetra()” borrará la letra de la posición actual y nos posicionará en la columna anterior en caso de que no estemos en la columna inicial.­­



El siguiente método a ver es el de “comprobarPalabra()”, este método se encarga de comparar la palabra introducida cuando se pulsa “ENTER”, primero comprueba si la columna actual es la última, si no lo es, no hace nada, así no hace todos los siguientes pasos en caso de que no sea una palabra de 5 letras. Una vez la palabra es de 5 letras, comprueba en el .txt por medio de una petición fetch, entre todas las palabras del archivo .txt separadas por saltos de línea si coincide con alguna palabra, si no coincide, diremos que la palabra no existe y deberá volver a escribir la palabra, pero no borraremos toda la fila por si sólo se equivocó en la última letra aunque podría haber sido una opción borrar todas las letras de la fila (sólo que no me gustaba y el Wordle original además no funciona de esa manera). En vez de utilizar un boolean para saber si la palabra existe o no, he utilizado un contador (0 = no existe, +1 = existe), porque inicialmente el .txt estaba mal y tenía las palabras repetidas aunque luego eso lo solucioné con un comando y una expresión regular para modificar el archivo. Usará también la función efectos que básicamente serán todos los efectos que hemos visto por CSS también de “flip” y los colores que asignaremos (lo veremos en más profundidad cuando vayamos a la función). Lo primero que comprueba es si la palabra intento coincide con la palabra aleatoria, así no miramos más, en ese caso el juego habrá acabado, actualmente no hace nada más pero debería guardar los datos (número de intentos y si es victoria o no) y enviárselos a la BBDD.

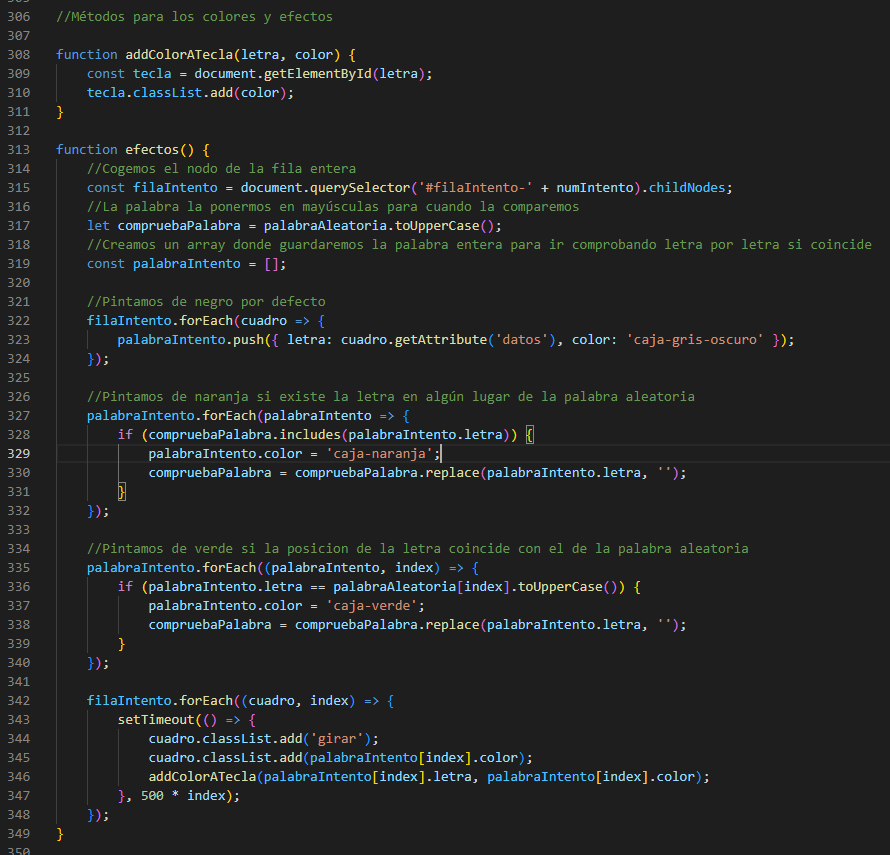
En caso de que no sea así, comprobaremos si el intento es el último, si lo es, es que ha perdido y sacaremos por pantalla un alert sencillo (mejorable con un mensaje más currado, ya sea con un div oculto o un iframe). En caso de que no sea su último intento, aumentamos el número de fila (intento) y reseteamos la columna intento a 0 (posición 1 en columna).

Texto

Descripción generada automáticamente

La función “addColoATecla(letra, color)” es una función que añade a todos los nodos (esto incluye tanto la tabla de intentos como la del teclado, por eso está hecho de ésta manera) de la letra que hayamos pasado por parámetros el parámetro que pasemos (verde, naranja o gris oscuro).

El método efectos es un método donde comprobamos letra por letra si coincide con la palabra y sus posiciones, primero comprobamos si no existe la letra, en cuyo caso la pintamos de color gris oscuro, en caso de que exista sin conocer aún la posición, obtendrá el color naranja, y si existe y en la misma posición, se pintará de verde (esto hará que sobrescriba el color naranja pero realmente se hace tan rápido que no será posible apreciarlo). Después en toda la fila (que hemos llamado “cuadro” la cual está pasada por parámetro cogida de “filaIntento” y la posición que será el “index”. Lo hacemos con un “setTimeout” para que no sea inmediato y se pueda apreciar el efecto de flip (giro), así como añadir los colores). Con esto conseguiremos el efecto tanto de los colores en ambas partes (teclado e intentos) como de giro de los cuadros de intentos.



## Mysql.js (WIP)

Actualmente usando node.js para realizar la conexión con la BBDD, comprobamos la conexión con una función “connect” sencilla y un mensaje por consola, tendremos funciones con posibilidad de exportar/importar (gracias a node.js y la declaración “export” en éste js y la declaración de “import” en el js que queramos importarlas) las funciones necesarias para hablar con la BBDD. Una será comprobarEmail que como dice la función comprobará que el email tiene formato email por medio de una expresión regular, después deberá comprobar dicho email en la BBDD para comprobar que no exista ya (ya que el campo de email sólo estará en el registro y no será para login). La función de comprobarUsuario que comprobará si existe el usuario, devolverá un booleano o un int de 0 o 1 (aún por hacer), en caso falso o 0 será que el usuario no existe, en caso verdadero o 1 será que existe, y dependiendo de ese return, en función de si estamos en el registro o en el login, haremos una cosa u otra (en registro dará error si existe porque ya está creado el usuario, y en login no hará nada y comprobará luego el siguiente dato, que será la contraseña). La función registrarUsuario() será para el registro, necesitará los parámetros que son los 3 datos para registrar un usuario (añadí el email pese a que en principio no debería estar ni ser necesario así como hacer el registro). Quedan por hacer además de éstas funciones la de login, la de sacar con un SELECT las estadísticas y un ranking, en el cual el SELECT para obtener el ranking sería:

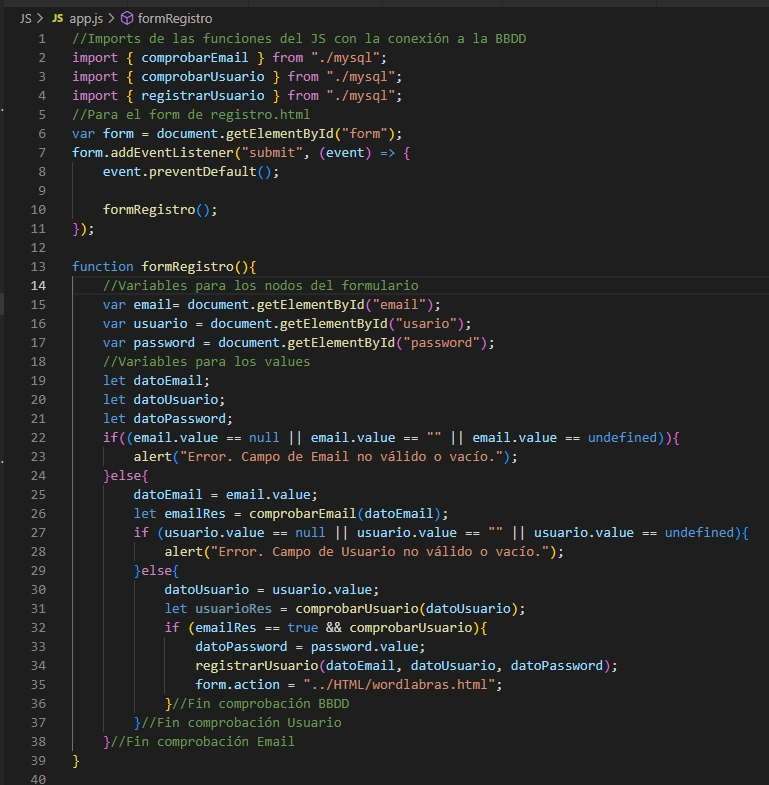
|  |
| --- |
| (SELECT usuario FROM `usuarios` WHERE usuario ='admin'), (SELECT SUM(victoria) FROM `estadisticas` WHERE estadisticas.victoria = 1), (SELECT SUM(intentos) FROM `estadisticas`)) |

Texto

Descripción generada automáticamente

## App.js (WIP)

Éste JS será para todo el funcionamiento y conexión/intercambio con la BBDD pero manteniendo toda la base en el js de mysql, es decir, aquí tendrá todas las funcionalidades más cercanas a la app (recogida de datos, uso de las funciones de mysql.js, etc…).



Pantalla negra con letras blancas

Descripción generada automáticamente