Actividad- Proyecto
[GRUPO]
Juego del Ahorcado

Resultados de Aprendizaje y Criterios de Evaluación aplicados

RA5	**Realiza operaciones de entrada y salida de información, utilizando procedimientos específicos del lenguaje y librerías de clases.**
b	Se han aplicado formatos en la visualización de la información.
d	Se han utilizado ficheros para almacenar y recuperar información.
e	Se han creado programas que utilicen diversos métodos de acceso al contenido de los ficheros.
f	Se han utilizado las herramientas del entorno de desarrollo para crear interfaces gráficos de usuario simples.
g	Se han programado controladores de eventos.
h	Se han escrito programas que utilicen interfaces gráficos para la entrada y salida de información.
RA6	**Escribe programas que manipulen información, seleccionando y utilizando tipos avanzados de datos.**
a	Se han escrito programas que utilicen arrays.
b	Se han reconocido las librerías de clases relacionadas con tipos de datos avanzados.
c	Se han utilizado listas para almacenar y procesar información.
d	Se han utilizado iteradores para recorrer los elementos de las listas.
e	Se han reconocido las características y ventajas de cada una de las colecciones de datos disponibles.
f	Se han creado clases y métodos genéricos.
g	Se han utilizado expresiones regulares en la búsqueda de patrones en cadenas de texto.
h	Se han identificado las clases relacionadas con el tratamiento de documentos escritos en diferentes lenguajes de intercambio de datos
i	Se han realizado programas que realicen manipulaciones sobre documentos escritos en diferentes lenguajes de intercambio de datos.
j	Se han utilizado operaciones agregadas para el manejo de información almacenada en colecciones.

Todos los RA y CE se aplican correspondientemente en los apartados indicados.

Enunciado

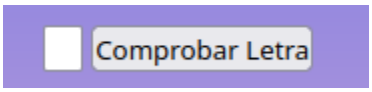
Por equipos (de **3 miembros máximo**) se debe desarrollar el juego conocido como "**El Ahorcado**" (o "**Hanged**"), en **HTML, CSS y JS**.

Para ello, deberás realizar lo siguiente:

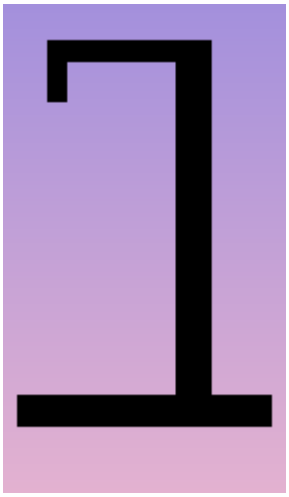
- Crear un directorio con el nombre "**Act_JS_Ahorcado_Nombre_Apellidos**", donde el nombre y apellidos serán los de uno de los miembros del equipo (la persona que realizará la entrega en Aules).
- Dentro del directorio, tendréis que crear tantos archivos HTML, CSS y JS como consideréis.
- Inicialmente, al ejecutar la aplicación, **se leerá el archivo JSON adjunto en Aules, que previamente deberá ser descargado** en el directorio raíz. Su contenido deberá ser cargado en un array. **(RA5de RA6hij)**
- Del array anterior deberemos, **aleatoriamente**, obtener una palabra secreta.
- Mostraremos en pantalla tantos guiones como huecos tenga la palabra secreta. Ejemplo:

- La pantalla deberá tener un cuadro de texto (input text pequeño) donde podremos introducir una letra, junto a un botón "Comprobar Letra" (del que más tarde explicaremos su funcionalidad). Ejemplo:

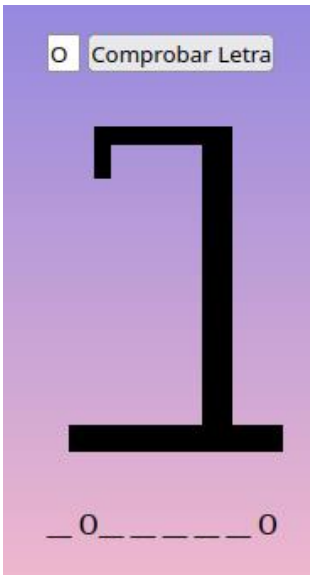
- Una imagen fija mostrará el estado inicial del juego. Dicha imagen será "Hangman-0.png", cargada de la **carpeta images, descargada, previamente de Aules**, en la ruta raíz de la aplicación.

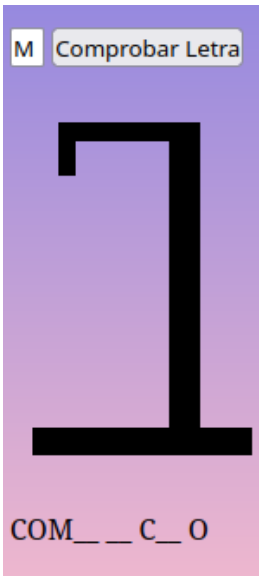
  

- Por último, tendremos una cabecera, con el título "**El Abogado...**", arriba del todo. Ejemplo:

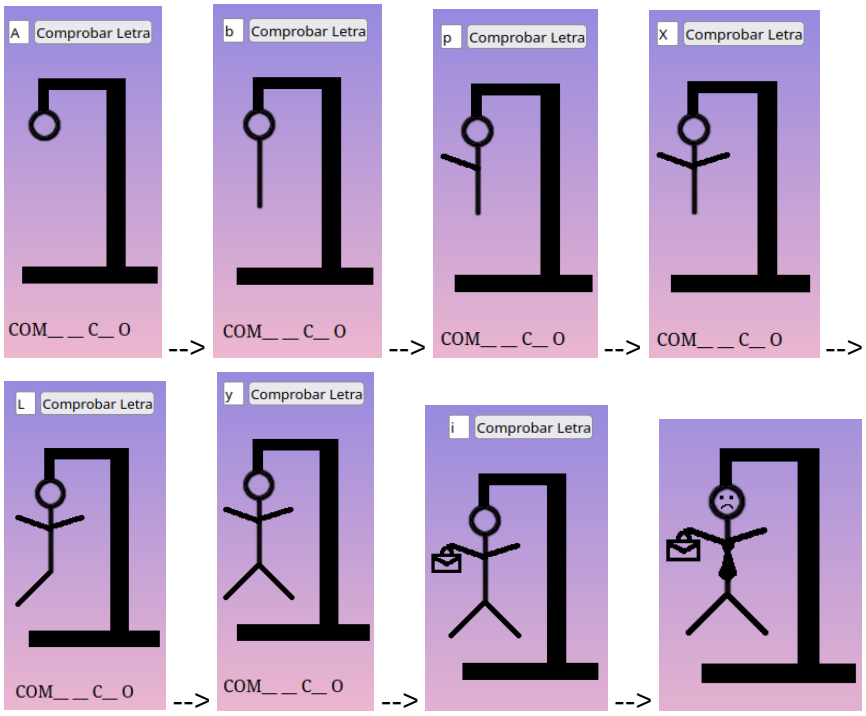
El Abogado...

- Funcionamiento del juego **(RA5g RA6abcdefg)**:
  - Escribiremos una única letra en el cuadro de texto (input text pequeño) y haremos clic en el botón "Comprobar Letra".
  - Si existe la letra dentro de la palabra secreta, ya sea mayúscula o minúscula (no haremos distinción), descubriremos su hueco (o huecos) en los guiones que la ocultan visualmente. Ejemplos:





  - Si no existe la letra, cambiaremos dinámicamente la imagen por la siguiente existente, siguiendo un orden pintando la figura de un **abogado**:
    - Hangman-1.png
    - Hangman-2.png
    - Hangman-3.png
    - ...
    - Hangman-8.png (última imagen donde se pierde)
    - *Ejemplo de secuencia donde las letras no existen:*



- **FIN DEL JUEGO**. El juego termina si:
  - El abogado es ahorcado, llegando a la última imagen y cumpliendo con los **8 INTENTOS posibles**. Llegados a este punto, mostraremos el resto del texto del encabezado "***...que tengo aquí colgado***". Ejemplo:

- Acertamos y descubrimos todas las letras de la palabra secreta. Entonces mostraremos un mensaje similar al siguiente:

"***...a quien has Ayudado***"

- **ESTADO INICIAL DEL JUEGO**. **(RA5fh)** Ejemplo:
  - Archivo JSON cargado en array y, de este, palabra secreta aleatoria extraída y visualizada de forma oculta a través de guiones. La imagen inicial es "Hangman-0.png":

- **Consideraciones/ Requisitos generales**

  - Aplica estilos CSS para mejorar la visualización de tu juego. Utiliza frameworks CSS como Bootstrap 5 si lo consideráis oportuno. Cuanto mejor estilo visual tenga la aplicación, mejor calificación en su CE y RA. **(RA5b)**

  - Como siempre: Comenta bien tu código, utiliza nombres a las variables autoexplicativos, utiliza constantes, haz uso del depurador...

- **Control de Errores.**

  - Utiliza TRY CATCH donde creas oportuno con el objetivo que la aplicación no se "rompa".
  - A la hora de introducir valores de entrada (input text) comprueba que sólo podemos introducir una letra (mayúscula o minúscula). No debe permitir introducir números ni una cadena de más de un caracter.
  - En el JSON todas las palabras están escritas en mayúscula. Tenedlo en cuenta ya que NO hay distinción entre introducir una 'A' o una 'a' por ejemplo para adivinar la letra de una palabra secreta. Ambos casos deberían funcionar igual.

- **Uso de módulos y funciones (utilizando la IMPORTACIÓN):**

  - Crea e importa un archivo de funciones ("funciones.js", por ejemplo) con el objetivo de modular tu proyecto lo máximo posible y así repartir mejor las tareas con los compañeros del equipo. Hay funciones que podrás utilizar en módulos aparte y otras que tendrás que hacerlo en el archivo principal ("app.js", por ejemplo).
  - Debéis **diseñar y crear las funciones** que necesitéis y consideréis para cada una de las características y funcionalidades del juego. Sois libres de organizar vuestro código de la mejor manera posible.

- **Entrega**:
  - Comprime tu directorio en un archivo ZIP con exactamente la misma nomenclatura, pero extensión ZIP (**sólo un miembro por grupo**):
    - **Act_JS_Ahorcado_Nombre_Apellidos.zip**
    - Ejemplo: "*Act_JS_Ahorcado_Miguel_Angel_Sanchez_Benito.zip*"
    - Sube tu archivo ZIP a Aules
  - Aparte, trabaja en equipo con GIT
  - **El incumplimiento de alguna de estas normas será penalizado en la calificación**.

- ***Rúbricas → Ver calificación en Aules***

- ***Autoevaluación y Coevaluación***

  Vinculado a la tarea, existirá un cuestionario de autoevaluación y coevaluación del grupo. De esta forma, cada integrante podrá autoevaluarse explicando qué ha hecho y podrá coevaluar a sus compañeros.
  Este cuestionario es totalmente confidencial y servirá para equilibrar el peso de la calificación final entre los miembros del equipo.