
INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	*PROGRAMACIÓN WEB 2*				
TÍTULO DE LA PRÁCTICA:	*Git y GitHub*				
NÚMERO DE PRÁCTICA:	*3*	**AÑO LECTIVO:**	*2024*	**NRO. SEMESTRE:**	*II*
FECHA DE PRESENTACIÓN	*5/10/2024*	**HORA DE PRESENTACIÓN**	*10/mm/ss*		
INTEGRANTE (s)					
Marron Puma ,Maritza Claudia *Pacheco Milene* *Paredes Jordan*				**NOTA (0-20)**	
DOCENTE(s): *Lino José Pinto Oppe*					

INFORMACIÓN BÁSICA
RESULTADOS Y PRUEBAS

URL del Repositorio GitHub

- **URL para clonar o recuperar el repositorio**: https://github.com/LINOPINTO2023/PWII-2024B.git

URL para el Laboratorio 03 en el Repositorio GitHub

- **URL para el laboratorio 03 en el repositorio**:
  https://github.com/LINOPINTO2023/PWII-2024B/tree/Grupo-Miau/LABORATORIO_III_EJERCICIOS

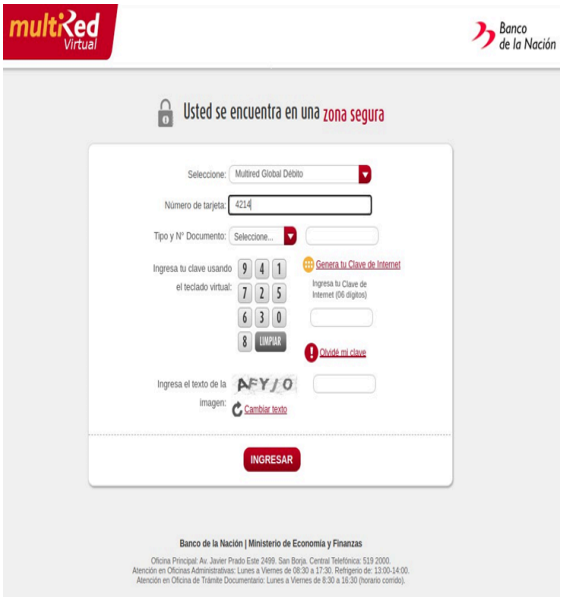
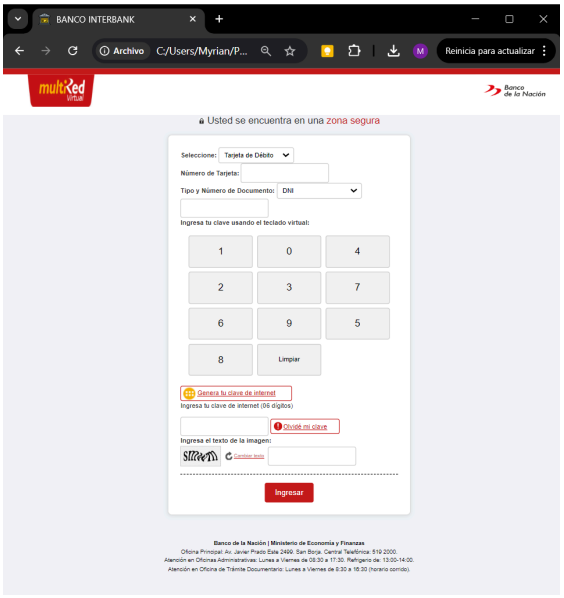
I. EJERCICIOS RESUELTOS:

El estudiante coloca la evidencia de los ejercicios propuestos realizados en la sesión de laboratorio, en el tiempo o duración indicado por el docente.

El docente debe colocar la retroalimentación por cada ejercicio que el estudiante/grupo ha presentado

EJERCICIO 01

Implementación de un Teclado Virtual Aleatorio del Banco Interbank

VISTA PREVIA	*PÁGINA WEB REALIZADA POR EL GRUPO*
	

Este informe detalla la creación y configuración de un sistema de autenticación web para el Banco Interbank. El objetivo del código es proporcionar una interfaz que simule la página de inicio de sesión del banco, manteniendo sus estilos, imágenes y colores. Para la funcionalidad del teclado virtual, se implementó un teclado aleatorio utilizando JavaScript, junto con HTML y CSS. A continuación, se explica cada parte del código con porciones relevantes.

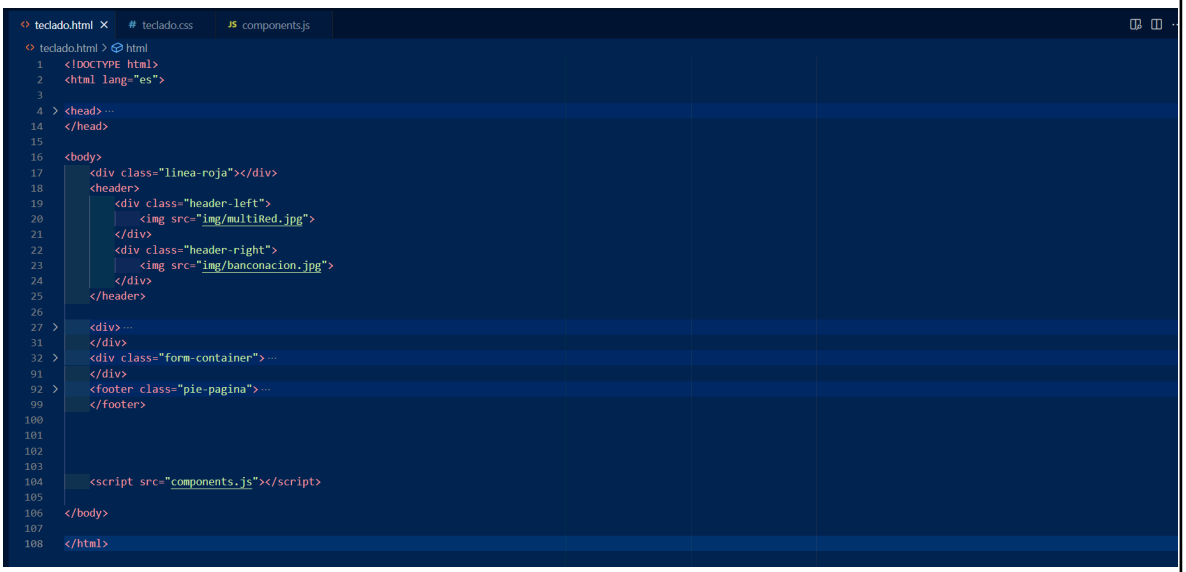
Creación del Documento HTML

Primero, se crea la estructura básica del documento HTML, incluyendo enlaces a hojas de estilo CSS y la configuración inicial del encabezado.

En esta parte del código, se configura el título de la página y se enlazan las hojas de estilo necesarias

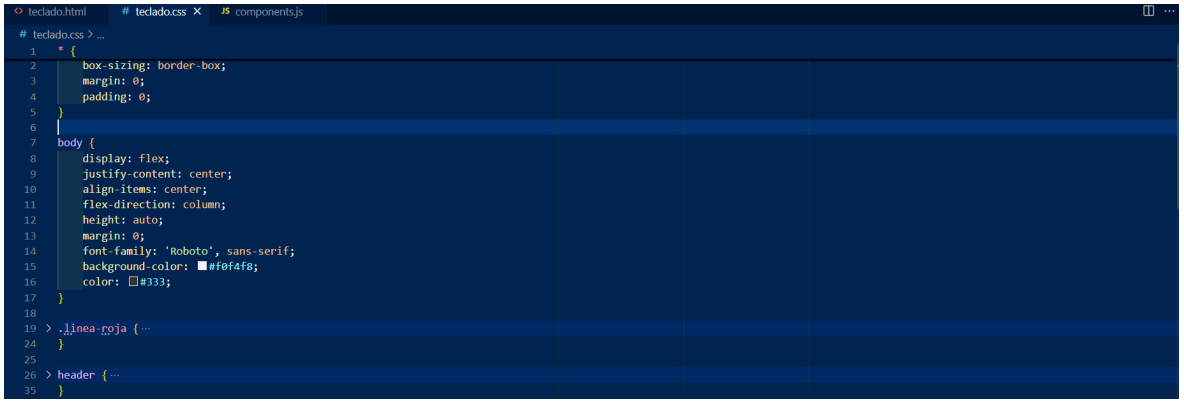
Creación del Encabezado

A continuación, se crea el encabezado de la página, que incluye imágenes representativas del banco



Estilo del Contenido

Se aplica un estilo CSS básico para el cuerpo del documento y los elementos, asegurando una presentación clara.

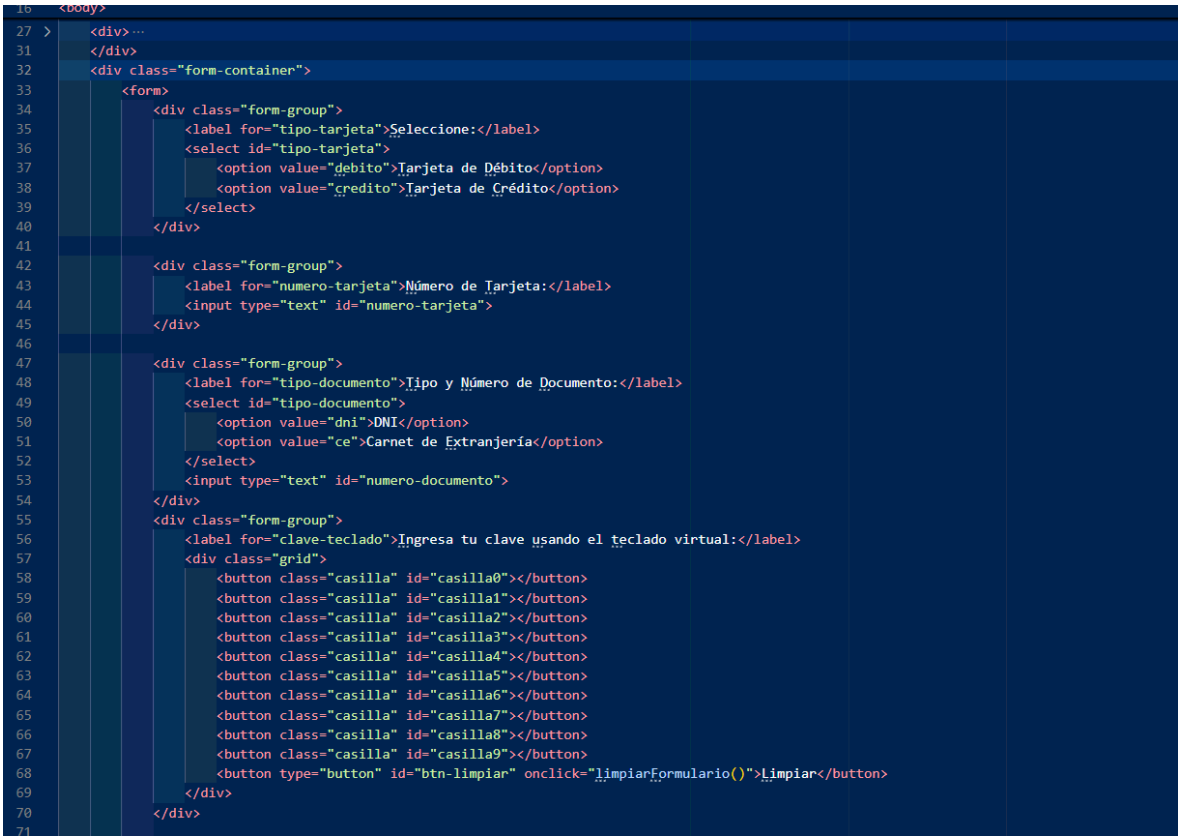


Creación del Formulario de Autenticación

Se crea un formulario que permite a los usuarios ingresar su información, como el tipo de tarjeta y el número de tarjeta.

Creación del Teclado Virtual

Dentro del formulario, se incluye un teclado virtual para la entrada de la clave de internet.



```
<body>
    <div>···
    </div>
    <div class="form-container">
        <form>
            <div class="form-group">
                <label for="tipo-tarjeta">Seleccione:</label>
                <select id="tipo-tarjeta">
                    <option value="debito">Tarjeta de Débito</option>
                    <option value="credito">Tarjeta de Crédito</option>
                </select>
            </div>

            <div class="form-group">
                <label for="numero-tarjeta">Número de Tarjeta:</label>
                <input type="text" id="numero-tarjeta">
            </div>

            <div class="form-group">
                <label for="tipo-documento">Tipo y Número de Documento:</label>
                <select id="tipo-documento">
                    <option value="dni">DNI</option>
                    <option value="ce">Carnet de Extranjería</option>
                </select>
                <input type="text" id="numero-documento">
            </div>
            <div class="form-group">
                <label for="clave-teclado">Ingresa tu clave usando el teclado virtual:</label>
                <div class="grid">
                    <button class="casilla" id="casilla0"></button>
                    <button class="casilla" id="casilla1"></button>
                    <button class="casilla" id="casilla2"></button>
                    <button class="casilla" id="casilla3"></button>
                    <button class="casilla" id="casilla4"></button>
                    <button class="casilla" id="casilla5"></button>
                    <button class="casilla" id="casilla6"></button>
                    <button class="casilla" id="casilla7"></button>
                    <button class="casilla" id="casilla8"></button>
                    <button class="casilla" id="casilla9"></button>
                    <button type="button" id="btn-limpiar" onclick="limpiarFormulario()">Limpiar</button>
                </div>
            </div>
```

Funciones de JavaScript

Se define una función para manejar el llenado del campo de la clave usando el teclado virtual. Los botones se llenan con números aleatorios.

Cuando el usuario accede a la página, el teclado virtual se presenta con números del 0 al 9 dispuestos de manera aleatoria. Al hacer clic en cualquier botón, el número correspondiente se añade al campo de contraseña, pero solo si la longitud de la contraseña es menor de seis dígitos. Si el usuario desea borrar el campo, puede hacerlo mediante un botón de limpiar.

1. window.onload
    a. Función General: Se ejecuta automáticamente cuando la página web ha terminado de cargar.
    b. Propósito: Asegurar que el DOM esté completamente cargado antes de intentar manipularlo.
2. const buttons
    a. Función General: Selecciona todos los botones del teclado virtual.

b. Propósito: Recopilar todos los elementos HTML con la clase "casilla" en un array llamado buttons, facilitando su manipulación posterior.

3. const numbers
    a. Función General: Genera un arreglo de números del 0 al 9 y los desordena aleatoriamente.
    b. Propósito: Crear un conjunto de números que se asignan a los botones, garantizando que su orden sea aleatorio cada vez que se carga la página.
4. Iteración sobre los botones
    a. Función General: Asigna un número a cada botón y establece un evento de clic.
    b. Propósito: Cada botón recibe un número del array numbers, y se agrega un evento que previene el comportamiento predeterminado y llama a la función para agregar el número al campo de contraseña.
5. function agregarNumero(numero)
    a. Función General: Agrega un número al campo de contraseña.
    b. Propósito: Controlar el ingreso de números, asegurando que solo se puedan agregar hasta 6 dígitos al campo de contraseña.
6. function limpiarFormulario()
    a. Función General: Limpia el contenido del campo de contraseña.
    b. Propósito: Proveer una manera de restablecer el campo de entrada, permitiendo que el usuario comience de nuevo si así lo desea.

Estilos Adicionales para el Documento

Se aplicaron estilos adicionales para mejorar la apariencia del formulario y sus elementos. Estos estilos se pueden ver con más precisión en el código disponible en nuestro repositorio remoto.

teclado.css > ...

```
17   }
18
19 > .linea-roja {…
24   }
25
26 > header {…
35   }
36
37   .header-left img,
38 > .header-right img {…
42   }
43
44 > h1 {…
47   }
48
49
50 > .form-container {…
57   }
58
59
60 > label {…
62   }
63
64   input,
65   select,
66 > button {…
71   }
72
73 > .grid {…
78   }
79
80 > .casilla {…
87   }
88
89 > .casilla:hover {…
91   }
92
93 > .captcha-container {…
97   }
98
99 > .captcha-container img {…
103  }
104
105 > .boton_captcha {…
```

teclado.css > ...

```
72
73 > .grid {…
78   }
79
80 > .casilla {…
87   }
88
89 > .casilla:hover {…
91   }
92
93 > .captcha-container {…
97   }
98
99 > .captcha-container img {…
103  }
104
105 > .boton_captcha {…
115  }
116
117 > .boton_clave {…
127  }
128
129 > .boton_genera {…
140  }
141
142
143 > .btn-ingresar {…
154  }
155
156
157
158 > .pie-pagina {…
169  }
170
171 > .pie-pagina p {…
174  }
175
176 > .pie-pagina strong {…
178  }
179
180
181 > .linea-puntos {…
184  }
```

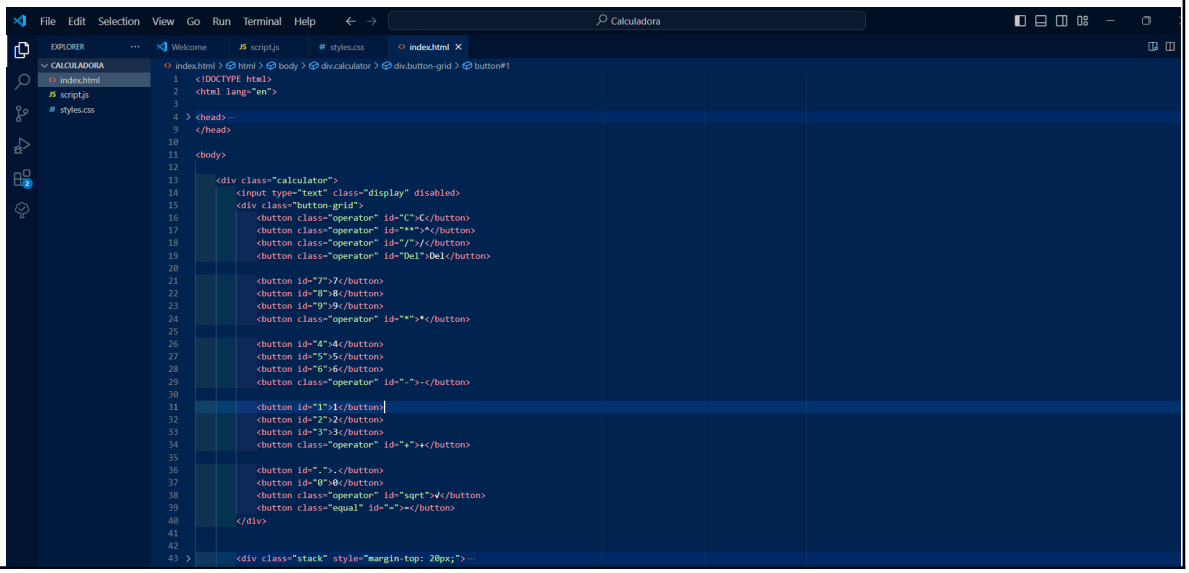
EJERCICIO 02:

Implementación de una calculadora básica utilizando HTML, CSS y JavaScript. Se detallaron las funciones y estilos que hacen que la calculadora sea funcional y

visualmente atractiva, así como la gestión de un historial de operaciones que proporciona un valor adicional a los usuarios.

Creación de la Interfaz de Usuario

Primero, se crea la estructura HTML de la calculadora, incluyendo el área de visualización y los botones para los números y operaciones.



En esta parte del código, se establece un contenedor principal que incluye un campo de entrada para mostrar el resultado y botones para las operaciones matemáticas. También se agrega un área para mostrar el historial de operaciones.

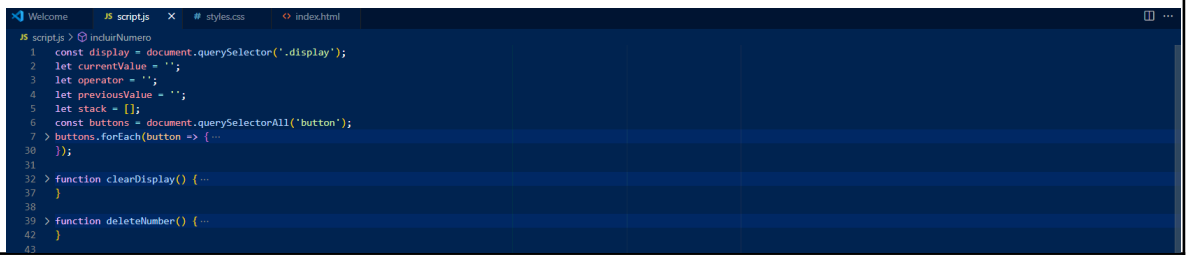
Estilos para la Calculadora

Se aplican estilos CSS para mejorar la presentación de la calculadora y hacerla más atractiva visualmente.

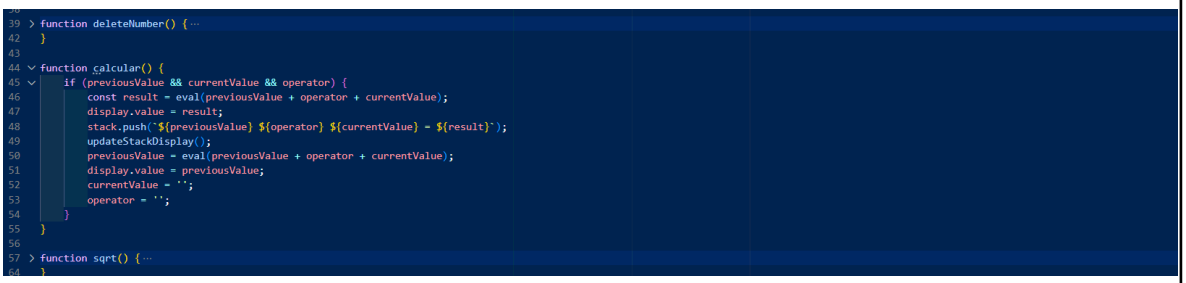
Se utilizan estilos para el cuerpo y el contenedor de la calculadora para asegurarse de que se vea bien en diferentes dispositivos. El fondo es suave, y los botones tienen un diseño atractivo que mejora la experiencia del usuario.

Funciones de la Calculadora

Se definen varias funciones en JavaScript para manejar las operaciones matemáticas y la interacción del usuario.



La función principal `calcular` se encarga de evaluar la expresión ingresada cuando el usuario presiona el botón de igual. También actualiza la pila de operaciones para que el usuario pueda ver un historial de sus cálculos



```
function deleteNumber() {
}

function calcular() {
    if (previousValue && currentValue && operator) {
        const result = eval(previousValue + operator + currentValue);
        display.value = result;
        stack.push(`${previousValue} ${operator} ${currentValue} = ${result}`);
        updateStackDisplay();
        previousValue = eval(previousValue + operator + currentValue);
        display.value = previousValue;
        currentValue = '';
        operator = '';
    }
}

function sqrt() {
```

La función `sqrt` permite calcular la raíz cuadrada del valor actual y lo agrega a la pila de operaciones.

```
}

function sqrt() {
    if (currentValue) {
        currentValue = Math.sqrt(parseFloat(currentValue)).toString();
        display.value = currentValue;
        stack.push(`sqrt(${currentValue}) = ${currentValue}`);
        updateStackDisplay();
    }
}

function operador(op) {
```

Actualización de la Pila de Operaciones

La función `updateStackDisplay` es responsable de mostrar el historial de operaciones en la interfaz de usuario.

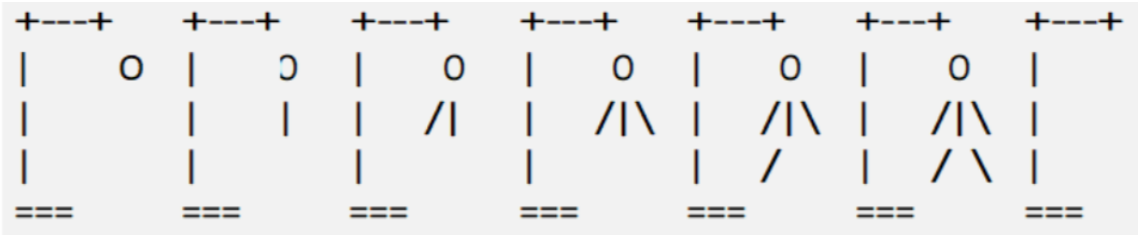
```
function incluirNumero(num) {
}
function updateStackDisplay() {
    const stackDisplay = document.querySelector('.stack');
    stackDisplay.innerHTML = stack.join('<br>');
}
```

Esto permite que el usuario vea claramente las operaciones que ha realizado, lo que mejora la usabilidad de la calculadora.

EJERCICIO 03

Cree una versión de el juego 'el ahorcado' que grafique con canvas paso a paso desde el evento onclick() de un botón.

VISTA PREVIA



PÁGINA WEB REALIZADA POR EL GRUPO

Este informe detalla la creación y configuración de un juego de 'El Ahorcado' que grafica paso a paso utilizando el elemento “canvas”de HTML5. El objetivo es proporcionar una interfaz donde el usuario pueda jugar al ahorcado, con las distintas partes del cuerpo del personaje siendo dibujadas progresivamente a medida que se falla en adivinar las letras. El proceso se controla mediante eventos onclick() asociados a un botón. El código está implementado utilizando JavaScript, junto con HTML y CSS para el diseño y la estructura. A continuación, se explica cada parte del código con porciones relevantes.

Creación del Documento HTML

Primero, se crea la estructura básica del documento HTML, incluyendo enlaces a hojas de estilo CSS y la configuración inicial del encabezado.

styles.css script.js index.html

index.html > ...

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ahorcado Moderno</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>

    <div class="game-container">
        <h1>Juego de Ahorcado</h1>
        <canvas id="ahorcadoCanvas" width="150" height="150"></canvas>
        <div id="wordContainer" class="word-display">_ _ _ _ _</div>
        <div id="message" class="message"></div>
        <div class="keyboard" id="keyboard"></div>
        <button id="resetButton" onclick="resetGame()">Reiniciar Juego</button>
    </div>

    <script src="script.js"></script>
</body>
</html>

```

En esta parte del código, se configura el título de la página y se enlazan las hojas de estilo necesarias

Estilo del Contenido

Se aplica un estilo CSS básico para el cuerpo del documento y los elementos, asegurando una presentación clara.

styles.css script.js index.html

styles.css > body

```css
body {
    font-family: 'Arial', sans-serif;
    background-color: #f0f0f0;
    margin: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
```


Funciones de JavaScript

Se define una función para manejar el llenado del campo de la clave usando el teclado virtual. Los botones se llenan con números aleatorios.

Se eligieron 5 palabras que serán las que se usarán para adivinar, que son elegidas aleatoriamente al iniciar el juego.

script.js > ...

```
const palabras = ["JAVASCRIPT", "HTML", "CSS", "AHORCADO", "CODIGO"];
let palabraSeleccionada = '';
let letrasAdivinadas = [];
let intentos = 6;
let juegoTerminado = false;
```

Estilos Adicionales para el Documento

Creamos la función iniciarJuego(); este método se encarga de iniciar el juego, seleccionando una palabra aleatoria de la lista predefinida, inicializando un arreglo con guiones bajos que representan las letras no adivinadas, y configurando el número de intentos en 6. También limpia cualquier mensaje anterior en la interfaz y actualiza la pantalla mostrando los guiones bajos en lugar de las letras. Finalmente, llama a la función generarTeclado() para crear el teclado virtual que los jugadores usarán para adivinar la palabra.

```
qodo Gen: Options | Test this function
function iniciarJuego() {
    palabraSeleccionada = palabras[Math.floor(Math.random() * palabras.length)];
    letrasAdivinadas = Array(palabraSeleccionada.length).fill('_');
    intentos = 6;
    juegoTerminado = false;
    document.getElementById('message').textContent = '';
    mostrarPalabra();
    dibujarAhorcado();
    generarTeclado();
}
```

Creamos la función mostrarPalabra; actualiza la interfaz mostrando las letras adivinadas hasta el momento, separadas por espacios. Si una letra ha sido correctamente adivinada, se muestra en su posición correspondiente, y las letras aún no adivinadas se muestran como guiones bajos.

```
qodo Gen: Options | Test this function
function mostrarPalabra() {
    document.getElementById('wordContainer').textContent = letrasAdivinadas.join(' ');
}
```

Estilos Adicionales para el Documento

Creamos la función generarTeclado; genera un teclado virtual creando botones para cada letra del alfabeto. Cada botón tiene un evento onclick() que llama a la función manejarLetra() cuando el usuario selecciona una letra. El teclado se muestra dinámicamente en la página, permitiendo al usuario interactuar con el juego.

```
25  function generarTeclado() {
26      const teclado = document.getElementById('keyboard');
27      teclado.innerHTML = '';
28      const letras = 'ABCDEFGHIJKLMNÑOPQRSTUVWXYZ';
29
30      for (let letra of letras) {
31          const button = document.createElement('button');
32          button.textContent = letra;
33          button.onclick = () => manejarLetra(letra);
34          teclado.appendChild(button);
35      }
36  }
```

Creamos la función manejarLetra; verifica si la letra seleccionada por el jugador está en la palabra. Si es correcta, actualiza la palabra adivinada en pantalla; si no, decrementa los intentos y llama a dibujarAhorcado() para añadir una parte del dibujo en el canvas. Finalmente, se llama a verificarEstadoJuego() para comprobar si el juego ha terminado.


```
qodo Gen: Options | Test this function
function manejarLetra(letra) {
    if (juegoTerminado) return;

    let acierto = false;
    for (let i = 0; i < palabraSeleccionada.length; i++) {
        if (palabraSeleccionada[i] === letra) {
            letrasAdivinadas[i] = letra;
            acierto = true;
        }
    }

    if (!acierto) {
        intentos--;
        dibujarAhorcado();
    }

    mostrarPalabra();
    verificarEstadoJuego();
}
```

Creamos la función verificarEstadoJuego; comprueba si el jugador ha adivinado todas las letras correctamente, en cuyo caso muestra un mensaje de victoria, o si se han agotado los intentos, lo que significa que el jugador ha perdido y se revela la palabra completa. En ambos casos, el juego se detiene.

```
function verificarEstadoJuego() {
    if (letrasAdivinadas.join('') === palabraSeleccionada) {
        document.getElementById('message').textContent = '¡Ganaste!';
        juegoTerminado = true;
    } else if (intentos === 0) {
        document.getElementById('message').textContent = 'Perdiste. La palabra era ' + palabraSeleccionada;
        juegoTerminado = true;
    }
}
```

Creamos la función dibujarAhorcado; dibuja el estado actual del ahorcado en el canvas. Dependiendo de los intentos restantes, se añaden diferentes partes del cuerpo del personaje, comenzando por la cabeza y continuando con el cuerpo, brazos y piernas, hasta completar el dibujo s el jugador pierde.


```
qodo Gen: Options | Test this function
function dibujarAhorcado() {
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    ctx.lineWidth = 4;
    ctx.strokeStyle = '#333';
    ctx.beginPath();
    ctx.moveTo(10, 140);
    ctx.lineTo(140, 140);
    ctx.moveTo(40, 140);
    ctx.lineTo(40, 20);
    ctx.lineTo(100, 20);
    ctx.lineTo(100, 40);
    ctx.stroke();

    if (intentos <= 5) {
        ctx.beginPath();
        ctx.arc(100, 50, 10, 0, Math.PI * 2);
        ctx.stroke();
    }
    if (intentos <= 4) {
        ctx.moveTo(100, 60);
        ctx.lineTo(100, 100);
        ctx.stroke();
    }
    if (intentos <= 3) {
        ctx.moveTo(100, 70);
        ctx.lineTo(80, 90);
        ctx.stroke();
    }
    if (intentos <= 2) {
        ctx.moveTo(100, 70);
        ctx.lineTo(120, 90);
        ctx.stroke();
```

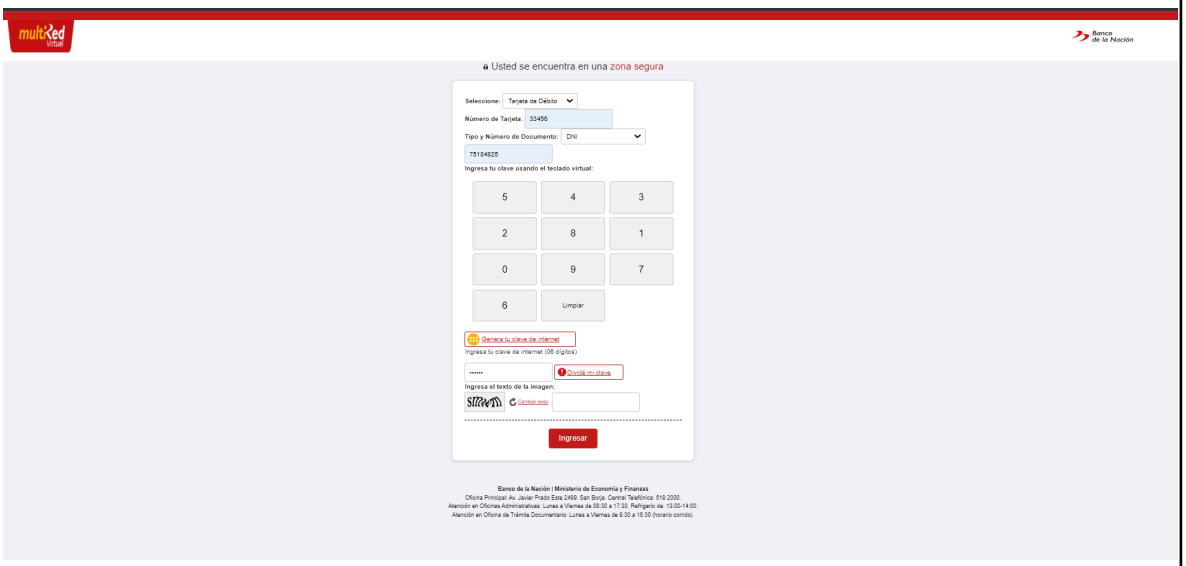
```
}
if (intentos <= 1) {
    ctx.moveTo(100, 100);
    ctx.lineTo(80, 120);
    ctx.stroke();
}
if (intentos <= 0) {
    ctx.moveTo(100, 100);
    ctx.lineTo(120, 120);
    ctx.stroke();
}
```

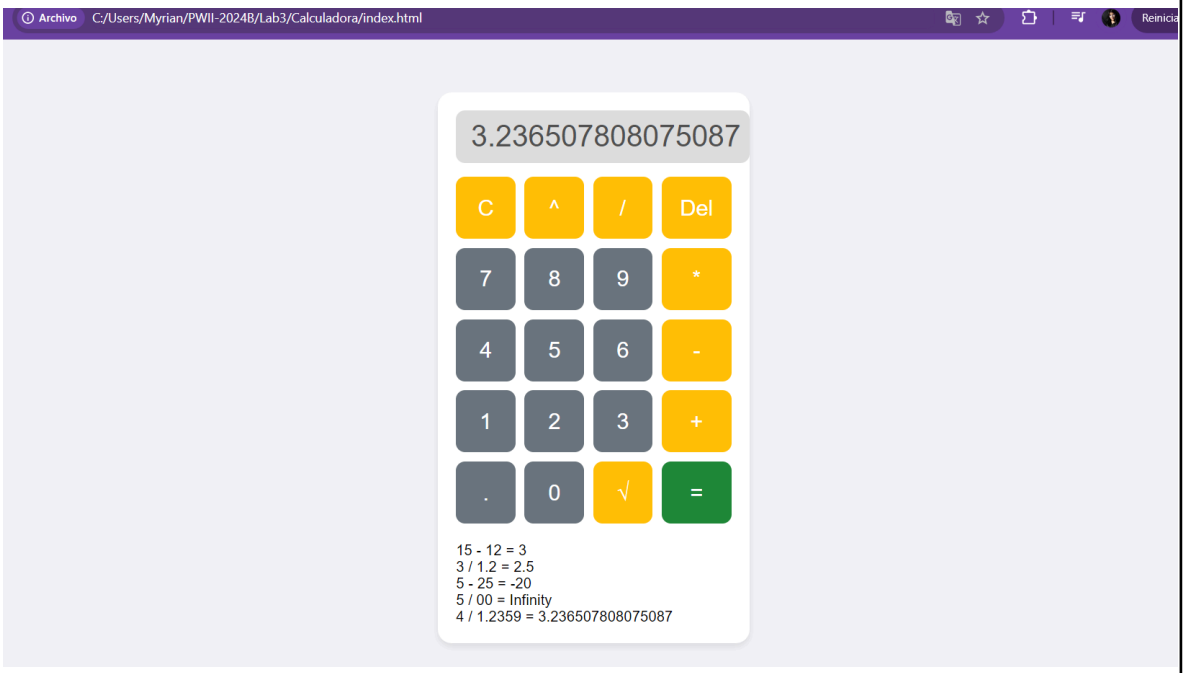
Creamos la función resetGame(); reinicia el juego llamando a iniciarJuego(), lo que resetea todas las variables y reinicia la interfaz para que el jugador pueda comenzar una nueva partida.

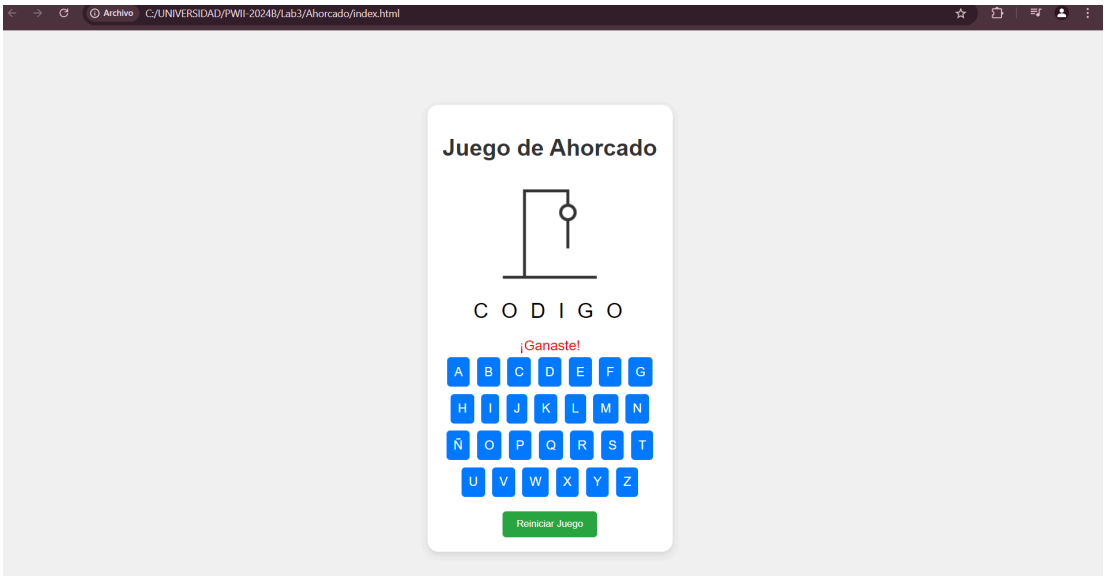
```
qodo Gen: Options | Test this function
114  function resetGame() {
115      iniciarJuego();
116  }
```

II. PRUEBAS

¿Con que valores comprobaste que tu práctica estuviera correcta? ¿Qué resultado esperabas obtener para cada valor de entrada? ¿Qué valor o comportamiento obtuviste para cada valor de entrada?







III. CUESTIONARIO:

Ofuscamiento de Código en JavaScript

El ofuscamiento de código en JavaScript es el proceso de transformar el código fuente legible y entendible en un código más difícil de interpretar. Este proceso tiene múltiples objetivos, entre los cuales destacan:

1. **Protección de la propiedad intelectual**: Dificulta el acceso no autorizado a los detalles internos del código.
2. **Prevención de la ingeniería inversa**: Complica el análisis del código, reduciendo el riesgo de que los hackers puedan replicar o modificar el software.
3. **Reducción del tamaño del archivo**: Mejora el rendimiento de la aplicación al disminuir el tamaño del código que se debe descargar y ejecutar.

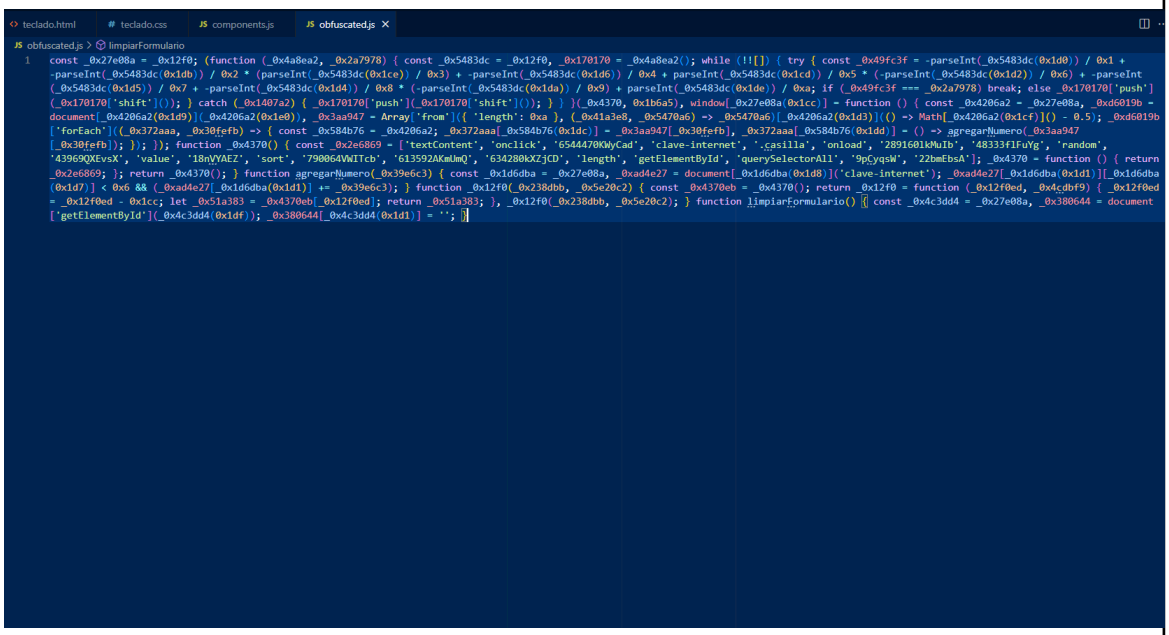
El ofuscamiento ayuda a proteger el código contra el robo y la modificación no autorizada. Además, puede hacer que sea más complicado para los hackers encontrar y explotar vulnerabilidades en el software.

Ejemplo de Ofuscamiento

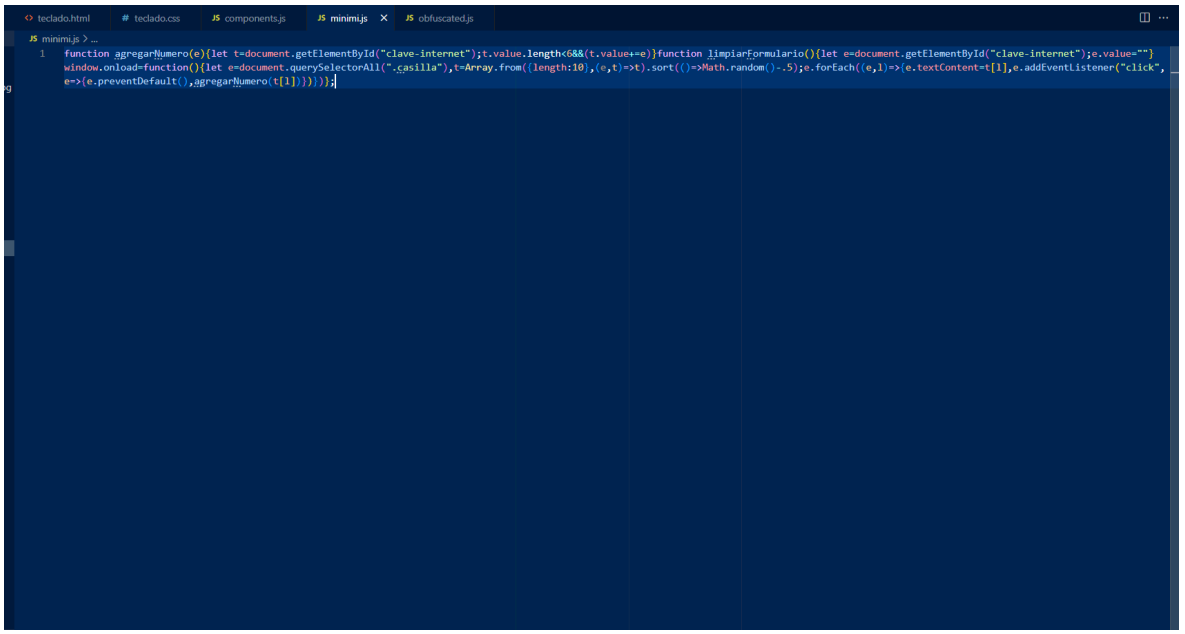
Aquí se muestra un ejemplo del proceso de ofuscamiento utilizando **JavaScriptObfuscator Tool,** una herramienta que permite transformar el código JavaScript en una versión más difícil de entender.

1. Código Original

Supongamos que tenemos el siguiente código de una calculadora simple en un archivo llamado obfuscated.js



- y minimizado con la herramienta https://www.toptal.com/developers/javascript-minifier:


teclado.html
teclado.css
components.js
minimi.js
obfuscated.js
minimi.js > ...
1
function agregarNumero(e){let t=document.getElementById("clave-internet");t.value.length<6&&(t.value+=e)}function limpiarformulario(){let e=document.getElementById("clave-internet");e.value=""}
window.onload=function(){let e=document.querySelectorAll(".casilla"),t=Array.from({length:10},(e,t)=>t).sort(()=>Math.random()-.5);e.forEach((e,l)=>{e.textContent=t[l],e.addEventListener("click",
e=>{e.preventDefault(),agregarNumero(t[l])})})};

COMMITS

Commits Más Recientes

A continuación se presentan los commits más recientes realizados en el repositorio:

ommits del 5 de octubre de 2024

- ***Último commit:*** *Añadidas líneas para centrar el botón en el CSS. Se añadió 'display: block;' y 'margin: 0 auto;'.*
  - ***Autor:*** *maritza1818*
  - ***Tiempo:*** *Hace 1 hora*
- ***Corregido error de sintaxis en el script que causaba problemas de funcionamiento.*** *Reestructuré las rutas de las imágenes para organizarlas en una carpeta 'img', mejorando así la organización de mi proyecto.*
  - ***Autor:*** *maritza1818*
  - ***Tiempo:*** *Hace 1 hora*
- ***Fix:*** *Evitar que el formulario se envíe al hacer clic en los botones del teclado virtual.*
  - ***Autor:*** *maritza1818*
  - ***Tiempo:*** *Hace 1 hora*
- ***Mejorando el teclado random.***
  - ***Autor:*** *maritza1818*
  - ***Tiempo:*** *Hace 17 horas*
- ***Agregar favicon a nuestra página web.***
  - ***Autor:*** *maritza1818*
  - ***Tiempo:*** *Hace 17 horas*
- ***Subir el archivo ofuscado de components.js.***
  - ***Autor:*** *maritza1818*
  - ***Tiempo:*** *Hace 17 horas*
- ***Implementar funciones de JavaScript como limpiar y agregarNumero().***
  - ***Autor:*** *maritza1818*
  - ***Tiempo:*** *Hace 17 horas*
- ***Mejoras en header, footer y estilos generales de la interfaz.***
  - ***Autor:*** *maritza1818*
  - ***Tiempo:*** *Hace 20 horas*
- ***Merge branch 'Grupo-Miau' of https://github.com/LINOPINTO2023/PWII-2024B into Grupo-Miau.***
  - ***Autor:*** *maritza1818*
  - ***Tiempo:*** *Ayer*
- ***Añadidas imágenes banconacion.jpg y descarga.png.***
  - ***Autor:*** *maritza1818*
  - ***Tiempo:*** *Ayer*

- ***Modificaciones en teclado.css.***
  - ***Autor:*** *maritza1818*
  - ***Tiempo:*** *Ayer*
- ***Resuelto conflicto en teclado.html.***
  - ***Autor:*** *maritza1818*
  - ***Tiempo:*** *Ayer*

ommits del 4 de octubre de 2024

- ***Últimos cambios.***
  - ***Autor:*** *JP-777*
  - ***Tiempo:*** *Ayer*
- ***Lab3.***
  - ***Autor:*** *JP-777*
  - ***Tiempo:*** *Ayer*
- ***Descripción de los cambios realizados en teclado.html y teclado.css.***
  - ***Autor:*** *maritza1818*
  - ***Tiempo:*** *Ayer*
- ***Actualización del formulario con nuevos campos y funcionalidades.***
  - ***Autor:*** *maritza1818*
  - ***Tiempo:*** *Ayer*
- ***Mover GUÍA DE LABORATORIO 1 - GRUPO 3.pdf a LABORATORIO_01_EJERCICIOS y renombrar carpeta src.***
  - ***Autor:*** *maritza1818*
  - ***Tiempo:*** *Ayer*

REFERENCIAS Y BIBLIOGRAFÍA

[1] Bootstrap, “Spinners,” Bootstrap Documentation, 2024. [Enlace: tps://getbootstrap.com/docs/5.3/components/spinners/].

¡OJO! Anexar a su informe

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	x	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	3	x	2	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	x	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	x	1	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	x	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	x	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	x	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	x	3	
TOTAL		20		16	

RUBRICA PARA EL CONTENIDO DEL INFORME Y DEMOSTRACIÓN

El alumno debe marcar o dejar en blanco en celdas de la columna Checklist si cumplió con el ítem correspondiente.

Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo ítems.

El alumno debe autocalificarse en la columna Estudiante de acuerdo a la siguiente tabla:

Tabla 1: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 2: Rúbrica para contenido del Informe y demostración