

Informe de Laboratorio 03

Tema: Javascript

Nota

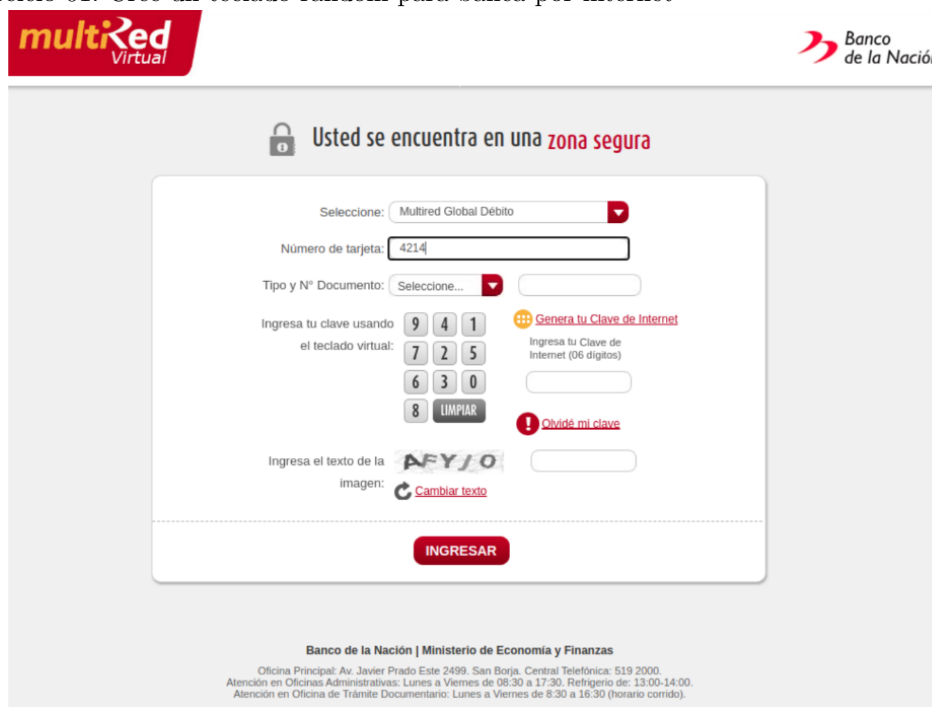
Estudiante	Escuela	Asignatura
Rodrigo Zarun Castillo Lazo rcastillola@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: I Código: 20231001

Laboratorio	Tema	Duración
03	Javascript	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 13 Mayo 2024	Al 19 Mayo 2024

1. Tarea

- Ejercicio 01: Cree un teclado random para banca por internet



multired Virtual

Banco de la Nación

Usted se encuentra en una zona segura

Seleccione: Multired Global Débito

Número de tarjeta: 4214

Tipo y N° Documento: Seleccione...

Ingresa tu clave usando el teclado virtual:

9 4 1
7 2 5
6 3 0
8 LIMPIAR

Ingresa tu Clave de Internet (06 dígitos)

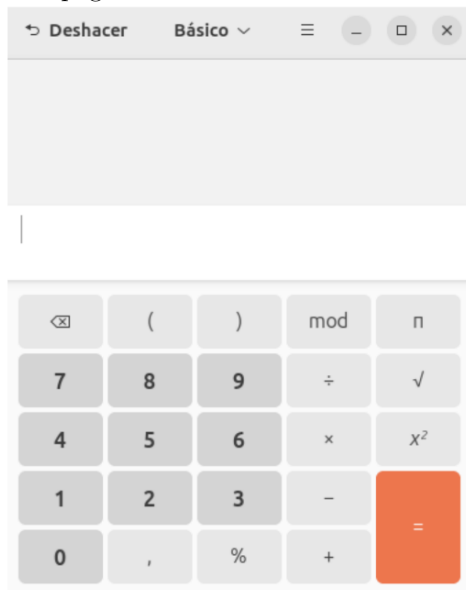
Ingresa el texto de la imagen: AFY/O

INGRESAR

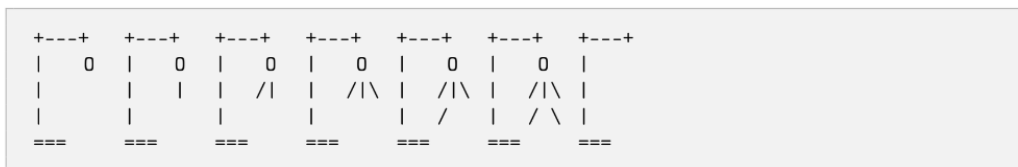
Banco de la Nación | Ministerio de Economía y Finanzas

Oficina Principal: Av. Javier Prado Este 2499, San Borja, Central Telefónica: 519 2000.
Atención en Oficinas Administrativas: Lunes a Viernes de 08:30 a 17:30. Refrigerio de: 13:00-14:00.
Atención en Oficina de Trámite Documentario: Lunes a Viernes de 8:30 a 16:30 (horario corrido).

- Ejercicio 02: Cree una calculadora básica como la de los sistemas operativos, que pueda utilizar la función eval() y que guarde todas las operaciones en una pila. Mostrar la pila al pie de la página web.



- Ejercicio 03: Cree una versión de el juego 'el ahorcado' que grafique con canvas paso a paso desde el evento onclick() de un botón.



2. Equipos, materiales y temas utilizados

- Sistema Operativo Manjaro 6.6.26-1-MANJARO
- git version 2.44.0
- Github

3. URL de Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/RodrigoCastilloLazo/pw2-24A.git>

4. Resolución

4.1. Ejercicio 01

- Se duplicó el index.html quitando agregados de UI/UX y archivos innecesarios.

- <https://github.com/RodrigoCastilloLazo/pw2-24A/blob/main/lab03/ejercicio01/index.html>
- Se duplicó el style.css y se obtuvo una versión simplificada.
- <https://github.com/RodrigoCastilloLazo/pw2-24A/blob/main/lab03/ejercicio01/style.css>
- Se implementó el siguiente código JavaScript:.

4.1.1. Inicialización de Eventos

```
document.getElementById('txtNumeroTarjeta').addEventListener('keypress', soloNumerosAll);
document.getElementById('txtNumDoc').addEventListener('keypress', soloNumerosDNI);
document.getElementById('boton_captcha').addEventListener('click', cambiarCaptcha);
document.addEventListener('DOMContentLoaded', function() {
    generarTeclado();
    cargarCaptcha();
    document.getElementById('limpiar').addEventListener('click', limpiarPassword);
});
```

- *Eventos keypress*: Estos eventos se añaden a los campos de texto txtNumeroTarjeta y txtNumDoc para asegurar que solo se permitan números.
- *Evento click*: El evento click en el botón de captcha llama a la función cambiarCaptcha.
- *Evento DOMContentLoaded*: Se asegura de que las funciones generarTeclado, cargarCaptcha, y el evento click para limpiar la contraseña se ejecuten una vez que el DOM esté completamente cargado.

4.1.2. Cambio de Tipo de Documento

```
function cambiarEspTextNumDoc() {
    limpiarNumDoc();
    cambiarEspacios();
}

function cambiarEspacios() {
    const tipoDoc = document.getElementById('cboTipoDoc').value;
    const txtNumDoc = document.getElementById('txtNumDoc');

    switch (tipoDoc) {
        case '1':
            txtNumDoc.maxLength = 8;
            break;
        case '2':
            txtNumDoc.maxLength = 20;
            break;
        case '3':
            txtNumDoc.maxLength = 20;
            break;
        case '4':
            txtNumDoc.maxLength = 11;
            break;
    }
}
```

```
}  
}
```

- **cambiarEspTextNumDoc**: Llama a **limpiarNumDoc** para limpiar el campo de número de documento y luego a **cambiarEspacios** para ajustar el **maxlength** del campo **txtNumDoc** según el tipo de documento seleccionado.
- **cambiarEspacios**: Ajusta el atributo **maxlength** del campo **txtNumDoc** según el valor seleccionado en el menú desplegable **cboTipoDoc**.

4.1.3. Captcha

```
const captchaData = [  
  { image: 'captchaImg/263S2V.jpg', answer: '263S2V' },  
  { image: 'captchaImg/6HJH6CTN.jpg', answer: '6HJH6CTN' },  
  { image: 'captchaImg/AAXUE.jpg', answer: 'AAXUE' },  
  { image: 'captchaImg/EXXTENHK.jpg', answer: 'EXXTENHK' }  
];  
  
function cargarCaptcha() {  
  const captcha = captchaData[Math.floor(Math.random() * captchaData.length)];  
  document.getElementById('captcha').src = captcha.image;  
  document.getElementById('captcha').setAttribute('data-answer', captcha.answer);  
}  
  
function cambiarCaptcha() {  
  const txtCaptcha = document.getElementById('txtCaptcha');  
  txtCaptcha.value = "";  
  cargarCaptcha();  
}  
  
window.onload = cargarCaptcha;  
  
function verificarCaptcha() {  
  const respuestaUsuario = document.getElementById('txtCaptcha').value.toUpperCase();  
  const respuestaCorrecta = document.getElementById('captcha').getAttribute('data-answer');  
  
  if (respuestaUsuario == respuestaCorrecta) {  
    return true;  
  } else {  
    alert('Incorrecto captcha');  
    cambiarCaptcha();  
    return false;  
  }  
}
```

- **captchaData**: Un array de objetos que contiene las imágenes de captcha y sus respuestas correctas.
- **cargarCaptcha**: Selecciona una imagen de captcha al azar y actualiza el **src** del elemento **img** correspondiente.
- **cambiarCaptcha**: Limpia el campo de texto del captcha y llama a **cargarCaptcha** para generar una nueva imagen de captcha.

- **verificarCaptcha:** Verifica si el texto ingresado por el usuario coincide con la respuesta correcta del captcha. Si es correcto, retorna **true**; de lo contrario, alerta al usuario y genera un nuevo captcha.

4.1.4. Autenticación

```
function autenticar() {
  const txtNumeroTarjeta = document.getElementById('txtNumeroTarjeta').value;
  const txtNumDoc = document.getElementById('txtNumDoc').value;
  const tipoDoc = document.getElementById('cboTipoDoc').value;
  const txtNumDocTamao = document.getElementById('txtNumDoc').maxLength;
  const txtPasswordTamao = document.getElementById('txtPassword').value.length;

  if (txtNumeroTarjeta.length === 16 && verificarCaptcha() && txtNumDocTamao ===
      txtNumDoc.length && txtPasswordTamao === 6) {
    alert('Ingreso Correcto');
    window.location.reload();
  } else {
    alert("Ingreso Incorrecto");
    window.location.reload();
  }
}
```

- **autenticar:** Valida que el número de tarjeta tenga 16 dígitos, que el captcha sea correcto, que el número de documento coincida con su longitud permitida y que la contraseña tenga 6 dígitos. Si todas las condiciones se cumplen, alerta Ingreso Correcto y recarga la página; de lo contrario, alerta Ingreso Incorrecto y recarga la página.

4.1.5. Generación del Teclado Virtual

```
function generarTeclado() {
  const numeros = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'];
  numeros.sort(() => Math.random() - 0.5);
  const teclado = document.getElementById('botones-clave');
  const limpiarBtn = document.getElementById('limpiar');
  teclado.innerHTML = '';

  numeros.forEach(num => {
    const boton = document.createElement('div');
    boton.textContent = num;
    boton.classList.add('boton-clave');
    boton.addEventListener('click', () => evalRanTable(num));
    teclado.appendChild(boton);
  });
  teclado.appendChild(limpiarBtn);
}

function evalRanTable(char) {
  const txtPassword = document.getElementById('txtPassword');
  if (txtPassword.value.length < 6) {
    txtPassword.value += char;
  }
}
```

```
function limpiarPassword() {
    const txtPassword = document.getElementById('txtPassword');
    txtPassword.value = '';
}
```

- **generarTeclado:** Genera un teclado virtual con números del 0 al 9 en orden aleatorio. Cada botón agrega el número correspondiente al campo de texto de la contraseña.
- **evalRanTable:** Añade un carácter al campo de texto de la contraseña si la longitud es menor a 6.
- **limpiarPassword:** Limpia el campo de texto de la contraseña.

4.1.6. Pagina ya implementada



4.2. Ejercicio 02

- Se crearon los archivos index.html y style.css correspondientes:
 - <https://github.com/RodrigoCastilloLazo/pw2-24A/blob/main/lab03/ejercicio02/index.html>
 - <https://github.com/RodrigoCastilloLazo/pw2-24A/blob/main/lab03/ejercicio02/style.css>

4.2.1. Explicación del Código JavaScript

Listing 1: Funciones Principales

```
document.addEventListener('DOMContentLoaded', () => {
    botones.forEach(boton => {
        boton.addEventListener('click', () => {
```

```
        const valor = boton.textContent;
        if (valor === '=') {
            if (estaAsignando) {
                asignarVariable();
            } else {
                calcularResultado();
            }
        } else if (valor === 'CE') {
            limpiarEntrada();
        } else if (valor === 'asig') {
            iniciarAsignacion();
        } else {
            aadirEntrada(valor);
        }
    });
});
function aadirEntrada(valor) {
    entradaActual += valor;
    actualizarPantalla(entradaActual);
}

function limpiarEntrada() {
    entradaActual = '';
    actualizarPantalla('0');
    estaAsignando = false;
}

function iniciarAsignacion() {
    if (entradaActual.trim() === 'x') {
        estaAsignando = true;
        entradaActual += ' = ';
        actualizarPantalla(entradaActual);
    } else {
        actualizarPantalla('Error: Use x para asignacin');
        entradaActual = '';
    }
}

function calcularResultado() {
    // Implementacin...
}

function asignarVariable() {
    // Implementacin...
}

function actualizarPantalla(valor) {
    pantalla.textContent = valor;
}
```

- Se utiliza DOMContentLoaded para asegurarse de que el código se ejecute una vez que el DOM esté completamente cargado.
- Se obtienen referencias a los elementos del DOM necesarios: la pantalla de la calculadora, el historial y los botones.

- Se inicializan variables de estado para rastrear la entrada actual, el historial de operaciones, las variables y el estado de asignación.
- Se asocian eventos de clic a los botones de la calculadora. Cada botón ejecutará una función específica dependiendo de su valor.

El código JavaScript asociado al archivo 'app.js' es el siguiente:

Listing 2: Código JavaScript de la calculadora

```
document.addEventListener('DOMContentLoaded', () => {
  const pantalla = document.getElementById('display');
  const elementoHistorial = document.getElementById('history');
  const botones = document.querySelectorAll('.buttons button');
  let entradaActual = '';
  let historial = [];
  let variables = {};
  let estaAsignando = false;

  botones.forEach(boton => {
    boton.addEventListener('click', () => {
      const valor = boton.textContent;

      // Manejo de los diferentes casos segun el valor del botn
      if (valor === '=') {
        if (estaAsignando) {
          asignarVariable();
        } else {
          calcularResultado();
        }
      } else if (valor === 'CE') {
        limpiarEntrada();
      } else if (valor === 'asig') {
        iniciarAsignacion();
      } else {
        aadirEntrada(valor);
      }
    });
  });

  elementoHistorial.addEventListener('click', (event) => {
    if (event.target && event.target.nodeName === "DIV") {
      entradaActual = event.target.textContent.split(' = ')[0];
      actualizarPantalla(entradaActual);
    }
  });

  function aadirEntrada(valor) {
    entradaActual += valor;
    actualizarPantalla(entradaActual);
  }

  function limpiarEntrada() {
    entradaActual = '';
    actualizarPantalla('0');
    estaAsignando = false;
  }
});
```



```
function iniciarAsignacion() {
  if (entradaActual.trim() === 'x') {
    estaAsignando = true;
    entradaActual += ' = ';
    actualizarPantalla(entradaActual);
  } else {
    actualizarPantalla('Error: Use x para asignacin');
    entradaActual = '';
  }
}

function calcularResultado() {
  if (!estaAsignando) {
    try {
      let expresion = entradaActual;
      for (let variable in variables) {
        let valor = variables[variable];
        let regex = new RegExp(`\\b${variable}\\b`, 'g');
        expresion = expresion.replace(regex, valor);
      }
      let resultado = eval(expresion);
      actualizarHistorial(entradaActual + ' = ' + resultado);
      entradaActual = resultado.toString();
      actualizarPantalla(resultado);
    } catch (error) {
      actualizarPantalla('Error');
      entradaActual = '';
    }
  }
}

function asignarVariable() {
  const partes = entradaActual.split('=');
  if (partes.length === 2 && partes[0].trim() === 'x') {
    try {
      let valor = eval(partes[1].trim());
      variables['x'] = valor;
      actualizarHistorial('x = ${valor}');
      actualizarPantalla('x = ${valor}');
      entradaActual = '';
      estaAsignando = false;
    } catch (error) {
      actualizarPantalla('Error en la asignacin');
      entradaActual = '';
      estaAsignando = false;
    }
  } else {
    actualizarPantalla('Error: Asignacin invalida');
    entradaActual = '';
    estaAsignando = false;
  }
}

function actualizarPantalla(valor) {
  pantalla.textContent = valor;
}
```

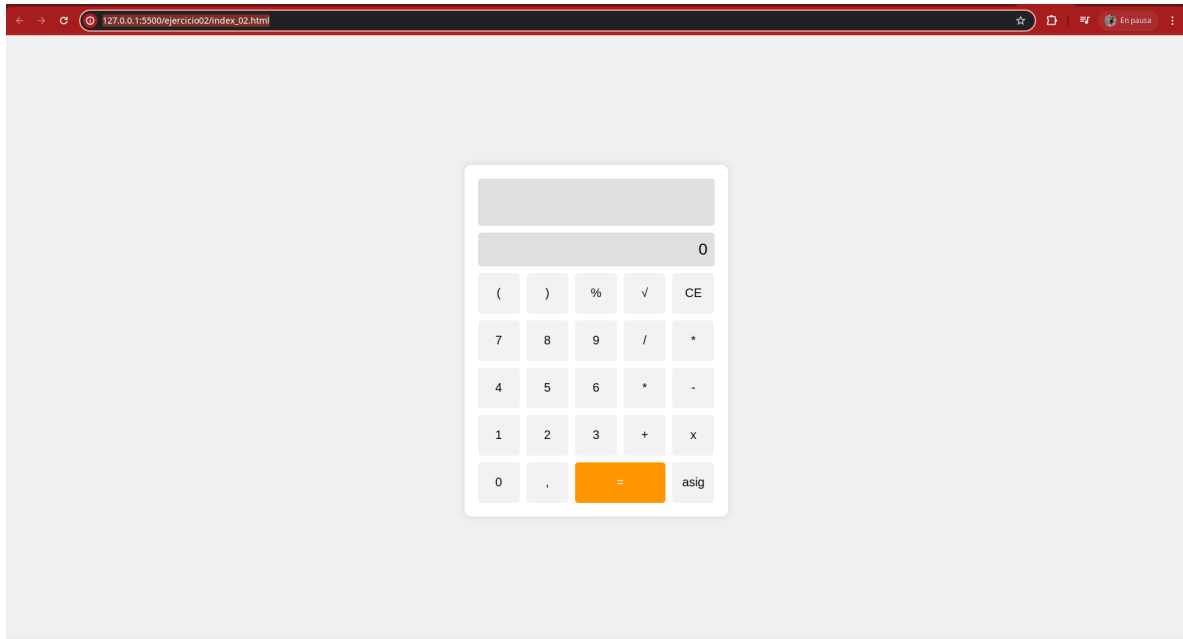
```
}

function actualizarHistorial(operacion) {
  if (historial.length >= 5) {
    historial.shift();
  }
  historial.push(operacion);
  renderizarHistorial();
}
function renderizarHistorial() {
  elementoHistorial.innerHTML = '';
  historial.forEach(item => {
    const elementoHistorialItem = document.createElement('div');
    elementoHistorialItem.textContent = item;
    elementoHistorial.appendChild(elementoHistorialItem);
  });
}

// Limpiar la entrada al cargar la pagina
limpiarEntrada();
});
```

- añadirEntrada(valor): Añade un valor a la entrada actual y actualiza la pantalla.
- limpiarEntrada(): Limpia la entrada actual y resetea la calculadora.
- iniciarAsignacion(): Inicia el proceso de asignación de una variable.
- calcularResultado(): Calcula el resultado de la operación ingresada.
- asignarVariable(): Asigna un valor a una variable.
- actualizarPantalla(valor): Actualiza el contenido de la pantalla de la calculadora con el valor proporcionado.

4.2.2. Implementacion en local



4.3. Ejercicio 03

- Se crearon los archivos index.html y style.css correspondientes:
 - <https://github.com/RodrigoCastilloLazo/pw2-24A/blob/main/lab03/ejercicio02/index.html>
 - <https://github.com/RodrigoCastilloLazo/pw2-24A/blob/main/lab03/ejercicio02/style.css>

4.3.1. Explicacion del codigo javascript

Listing 3: Variables Iniciales

```
const questions = [
  { question: "Capital de Francia", answer: "PARIS" },
  // ...otras preguntas
];

let selectedQuestion = questions[Math.floor(Math.random() * questions.length)];
let answer = selectedQuestion.answer;
let guessedLetters = [];
let mistakes = 0;

document.getElementById('question').innerText += selectedQuestion.question;
```

- questions: Array de preguntas y respuestas.
- selectedQuestion: Selecciona una pregunta aleatoria.
- answer: Respuesta a la pregunta seleccionada.
- guessedLetters: Almacena letras adivinadas.

- mistakes: Contador de errores.
- Actualiza el HTML para mostrar la pregunta.

Listing 4: Función para Mostrar la Palabra

```
function displayWord() {  
  let display = answer.split('').map(letter =>  
    (guessedLetters.includes(letter) ? letter : '_')).join(' ');  
  document.getElementById('word').innerText = display;  
}
```

- Muestra la palabra oculta con guiones bajos (_).

Listing 5: Función para Adivinar una Letra

```
function guessLetter() {  
  let input = document.getElementById('letter-input').value.toUpperCase();  
  document.getElementById('letter-input').value = '';  
  
  if (input && !guessedLetters.includes(input)) {  
    guessedLetters.push(input);  
    document.getElementById('used-letters').innerText = guessedLetters.join(' ');  
  };  
  
  if (!answer.includes(input)) {  
    mistakes++;  
    document.getElementById('hangman-image').src = 'images/${mistakes}.png';  
    if (mistakes === 6) {  
      alert('Perdiste! La respuesta era ' + answer);  
      resetGame();  
    }  
  } else {  
    displayWord();  
    if (!document.getElementById('word').innerText.includes('_')) {  
      alert('Ganaste! La respuesta es ' + answer);  
      resetGame();  
    }  
  }  
}
```

- input: Obtiene la letra ingresada por el jugador.
 - guessedLetters.push(input): Agrega la letra adivinada.
 - if (!answer.includes(input)): Incrementa errores si la letra no está en la respuesta y actualiza la imagen del ahorcado.
 - displayWord(): Actualiza y muestra la palabra adivinada.
 - resetGame(): Reinicia el juego en caso de ganar o perder.
- **Función para Reiniciar el Juego:**

Listing 6: Función para Reiniciar el Juego

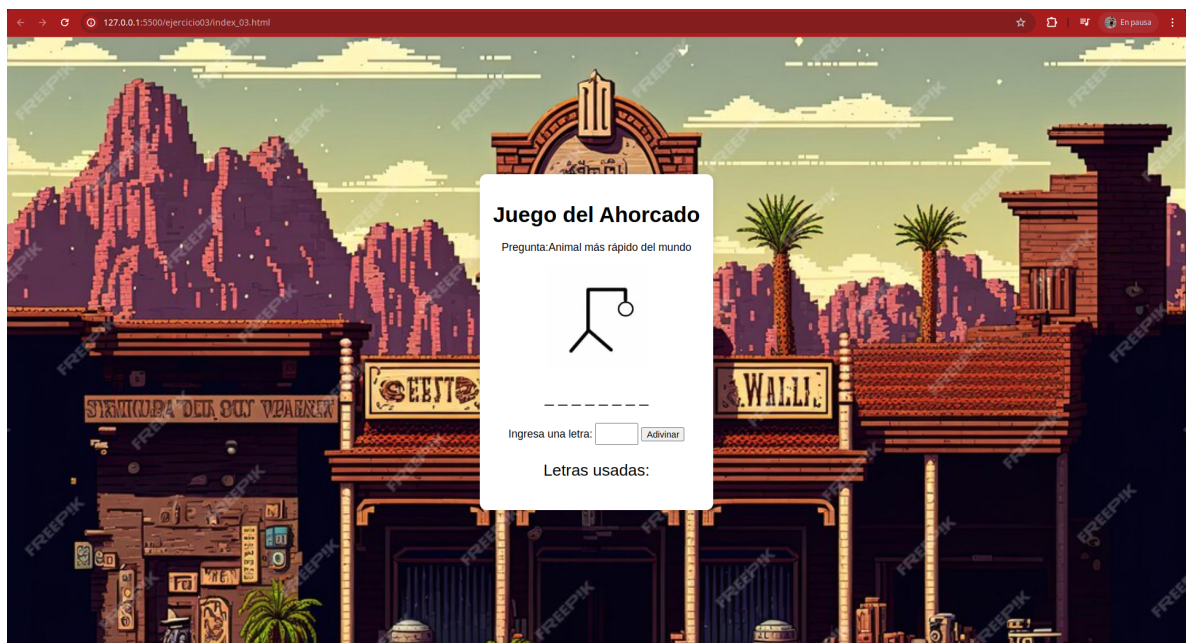
```
function resetGame() {  
  selectedQuestion = questions[Math.floor(Math.random() * questions.length)];  
  answer = selectedQuestion.answer;  
  guessedLetters = [];
```

```
mistakes = 0;
document.getElementById('question').innerText = 'Pregunta: ' +
    selectedQuestion.question;
document.getElementById('hangman-image').src = 'images/0.png';
document.getElementById('used-letters').innerText = '';
displayWord();
}

displayWord();
```

- `resetGame`: Selecciona una nueva pregunta, resetea variables y actualiza el HTML.
- `displayWord()`: Inicializa y muestra la palabra oculta al cargar la página.

4.3.2. Pagina ya implementada en local



5. Preguntas

5.1. Explique una herramienta para ofuzcar código JavaScript

- La herramienta usada fue javascript-obfuscator, un ofuscador de para javascript de código abierto
- <https://github.com/javascript-obfuscator/javascript-obfuscator/tree/master>
- Esta herramienta corre en node.js aplica una serie de transformaciones al código para cambiar su apariencia y hacerlo más difícil de entender. Estas transformaciones pueden incluir:
 - Minificación: Elimina espacios en blanco, comentarios y renombra variables a nombres más cortos para reducir el tamaño del archivo.
 - Renombrado de variables y funciones: Cambia los nombres de variables y funciones a nombres más cortos y menos descriptivos.
 - Encapsulamiento: Envuelve el código en funciones autoejecutables (IIFE) para limitar el alcance de las variables y funciones.
 - Cifrado: Aplica algún tipo de cifrado al código, como el cifrado Base64, para ocultar su contenido.
 - Técnicas de control de flujo: Modifica el control de flujo del programa para hacer que el código sea menos legible.

5.2. Ejemplo de ofuscación

- Se ofusco el javascript del primer ejercicio
- https://github.com/RodrigoCastilloLazo/pw2-24A/blob/main/lab03/ejercicio01/app_01.min.js

6. Rúbricas

6.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y fácil de leer.

Tabla 2: Niveles de desempeño

Puntos	Nivel			
	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

6.2. Autocalificación

Tabla 3: Rúbrica para contenido del Informe y demostración

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).	4	X	4	
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	2	
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 estan 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		19	

7. Referencias

- <https://github.com/javascript-obfuscator/javascript-obfuscator/tree/master>
- <https://www.youtube.com/watch?v=koiPxFFiqJ4>
- <https://nodejs.org/docs/latest/api/>