Nombre de la práctica	Ordenamiento por Selección			No.	1
Asignatura:	Estructura de Datos	Carrera:	Ing. Sistemas Computacionales	Duración de la práctica (Hrs)	2 hrs

Nombre del Alumno: IVAN MORA GARCIA
Grupo: 3401

I. Competencia(s) específica(s):

Conoce, comprende y aplica los algoritmos de ordenamiento para el uso adecuado en el desarrollo de aplicaciones que permita solucionar problemas en entornos.

II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):

Salón de clases

III. **Material empleado:**

- Laptop
- Visual Studio Code
- Internet

IV. Desarrollo de la práctica:

ALGORITMO DEL MÉTODO DE ORDENAMIENTO (PSeint)

OrdenamientoSeleccion.psc*

```
1  Algoritmo OrdenamientoSeleccion
2      Definir n, i, j, min, temp Como Entero
3      Dimension arr[100]
4
5      Escribir "Ingresa el tamaño del arreglo: "
6      Leer n
7
8      Para i ← 1 Hasta n Hacer
9          Escribir "Ingresa el elemento ", i, " del arreglo: "
10         Leer arr[i]
11     FinPara
```

Este algoritmo implementa el método de ordenamiento por selección para ordenar un arreglo de números enteros.

1. **Definir variables necesarias para el algoritmo:**
   - n: Almacena el tamaño del arreglo.
   - i: Variable de control para el bucle externo.
   - j: Variable de control para el bucle interno.
   - min: Índice del elemento mínimo en el rango actual.
   - temp: Variable temporal para intercambio de elementos.
   - Declaración de un arreglo llamado 'arr' que puede contener hasta 100 elementos.

2. **Solicitar el tamaño del arreglo y leerlo.**

3. **Llenar el arreglo.**
   - Utiliza un bucle para iterar desde 1 hasta 'n' e ingresa los elementos en el arreglo 'arr'.
   - Muestra un mensaje al usuario para ingresar cada elemento y lo almacena en la posición 'i' del arreglo.

```
Para i ← 1 Hasta n - 1 Hacer
    min ← i
    Para j ← i + 1 Hasta n Hacer
        Si arr[j] < arr[min] Entonces
            min ← j
        FinSi
    FinPara
```

4. **Ordenamiento por Selección.**
   - Inicializa 'min' con el valor actual de 'i'.
   - 'min' almacena el índice del elemento mínimo en el rango actual.
   - Utiliza un bucle interno para buscar el índice del elemento mínimo en el rango no ordenado.

```
    temp ← arr[i]
    arr[i] ← arr[min]
    arr[min] ← temp
FinPara
```

5. **Intercambio de variables.**
   - Intercambia el elemento actual con el elemento mínimo encontrado en el rango no ordenado.
   - Esto coloca el elemento mínimo en su posición correcta

```
    Escribir "Arreglo ordenado:"
    Para i ← 1 Hasta n Hacer
        Escribir arr[i]
    FinPara
FinAlgoritmo
```

6. **Mostrar el arreglo:**
   - Muestra el arreglo ordenado después de haber completado el proceso de ordenamiento.

ARCHIVO HTML

Para la creación de nuestro archivo HTML comenzamos con los siguientes pasos.

1. Iniciamos con la declaración de tipo de documento `<!DOCTYPE html>` indica que el documento es un documento HTML5.

2. Seguido del elemento `<html>` es el elemento raíz de la página HTML, y el atributo `lang` se establece en "es" para especificar que el contenido está en español.

3. La sección `<head>` contiene metainformación sobre el documento. Aquí tenemos el título de la página, que se mostrará en la barra de título del navegador, en este caso será Ordenamiento por Selección.

4. Dentro de `<head>`, hay un enlace a un archivo CSS externo (`css/styles.css`) que contiene los estilos de la página.

seleccion.html × seleccion.js

Estrutura de Datos > Unidad 3 > Metodos de Ordenamiento > 02_Seleccion > seleccion.ht

```
1  <!DOCTYPE html>
2  <html lang="es">
3      <head>
4          <title>Ordenamiento por seleccion</title>
5          <link rel="stylesheet" href="css/styles.css">
6      </head>
```

5. Creamos nuestro elemento `<body>` que contiene el contenido visible de la página. Comienza con `<div>` con la clase "login".

6. Dentro del div "login", hay un elemento `<h1>` con la clase "login_heading" que mostrará el texto "EJERCICIOS" como encabezado de la sección de inicio de sesión.

7. Creamos un elemento decorativo que usa `<span>` con la clase "login_circuit-mask".

```
<body>
    <div class="login">
            <h1 class="login_heading">Ordenamiento por SELECCION</h1>
            <span class="login_circuit-mask"></span>
```

8. ** :** Aquí se está creando un contenedor con una clase llamada "cantidad".

9. **<label class="form-label">Candidad</label>:** Esto crea una etiqueta de texto que dice "Cantidad". Se utiliza como una etiqueta descriptiva para el campo de entrada que viene a continuación.

10. **<input class="form-control" type="text" placeholder="Ingrese No. de Elementos" name="txtCantidad">:** Este es un campo de entrada de texto. El atributo type está configurado como "text", lo que significa que se espera texto como entrada. El atributo placeholder muestra un mensaje dentro del campo antes de que se ingrese cualquier texto. El atributo name establece el nombre del campo que se utilizará cuando se envíe el formulario.

```
<div class="cantidad">
    <label class="form-label">Candidad</label>
    <input class="form-control" type="text" placeholder="Ingrese No. de Elementos" name="txtCantidad">
</div><br>
```

11. **
:** Aquí se crea otro contenedor con un atributo id establecido en "output". El atributo id se utiliza para seleccionar elementos de manera única en JavaScript.

12. ** :** Crea otro contenedor con un atributo id establecido en "output2". Al igual que el anterior, este contenedor parece destinado a mostrar algún tipo de salida, pero el código relacionado podría estar en otro lugar.

```
<div id="output"></div><br>
<div id="output2"></div>
```

13. ** :** Creamos un contenedor div. Con una clase que esta definida en CSS para darle estilo.

14. **<button type="button" class="btn btn-ligth" onclick="ingresar()">:** Pedir Elementos</button>: Este es el primer botón dentro del contenedor. Aquí están los detalles:
    - **type="button":** Define el tipo de botón como "button".

- **class="btn btn-ligth":** Asigna dos clases CSS al botón. Estas clases podrían ser parte de un marco de diseño como Bootstrap. "btn" podría ser una clase que indica que es un botón, y "btn-ligth" podría ser una clase que controla su estilo de fondo.
- **onclick="ingresar()":** Define un atributo de evento. Cuando el botón se hace clic, se ejecutará la función JavaScript llamada "ingresar()". Esto significa que la función "ingresar()" debe estar definida en algún lugar del código JavaScript asociado a esta página web.

15. **<button type="button" class="btn btn-ligth" onclick="procesarArray()">Mostrar Array</button>:** Este es el segundo botón y su funcionamiento es similar al primero, pero en lugar de "ingresar()", hace referencia a la función "procesarArray()".

16. **<button type="button" onclick="ordenarSeleccion()">Ordenar Array</button>:** Este es el tercer botón. No tiene la clase "btn" ni "btn-ligth", por lo que podría tener un aspecto diferente a los dos primeros botones. Al hacer clic en este botón, se ejecutará la función "ordenarSeleccion()".

17. **<button type="reset" onclick="borrarArray()">Borrar</button>:** Este es el cuarto botón. Tiene el atributo type="reset", lo que significa que restablecerá los valores del formulario si hubiera uno. Al hacer clic en este botón, se ejecutará la función "borrarArray()".

```
<div class="botones"> <br>
    <button type="button" class="btn btn-ligth" onclick="ingresar()">Pedir Elementos</button>
    <button type="button" class="btn btn-ligth" onclick="procesarArray()" >Mostrar Array</button>
    <button type="button" onclick="ordenarSeleccion()">Ordenar Array</button>
    <br><br><button type="reset" onclick="borrarArray()">Borrar</button><br>
</div>
```

18. **<script src="js/seleccion.js"></script>:** Esta línea importa un archivo JavaScript en la página HTML. Veamos en detalle:

    - **<script>:** Esta etiqueta se utiliza para incluir código JavaScript en la página HTML.
    - **src="js/seleccion.js":** El atributo src especifica la ruta del archivo JavaScript que se va a incluir. En este caso, el archivo se llama "seleccion.js" y está ubicado en la carpeta "js" en el directorio del proyecto.
    - **</script>:** Esta etiqueta cierra el bloque de código JavaScript.

```
    <!-- //Enlazar el archivo js con html -->
    <script src="js/seleccion.js"></script>
    </body>
</html>
```

CÓDIGO DENTRO DEL ARCHIVO JS

I. Declaración y Llenado del Array:

- El código comienza declarando un array llamado “a” en el ámbito global.
- La función ingresar se encarga de llenar este array con números. Toma la cantidad de elementos a ingresar desde un campo de entrada en el HTML (txtCantidad). Luego, solicita al usuario que ingrese cada elemento y los agrega al array “a”.

seleccion.html | seleccion.js

Estrutura de Datos > Unidad 3 > Metodos de Ordenamiento > 02_Seleccion > js > seleccion.js > ...

```
1   // Declarar el array a en el ámbito global
2   var a = [];
3
4   const ingresar = () => {
5       let n = parseInt(document.getElementsByName('txtCantidad')[0].value)
6
7       if (isNaN(n) || n <= 0) {
8           alert('Ingrese cantidad de elementos válida');
9           return;
10      }
11
12      // Limpiar el array antes de llenarlo nuevamente
13      a = [];
14
15      for (c = 0; c < n; c++) {
16          elemento = parseInt(prompt('Ingrese el elemento'));
17          a.push(elemento);
18      }
19  }
```

1. **const ingresar = () => {:** Esto define una función anónima usando la sintaxis de funciones flecha. La función se llama ingresar.

1. **let n = parseInt(document.getElementsByName('txtCantidad')[0].value):** Esta línea obtiene el valor ingresado por el usuario en un campo de entrada con el nombre 'txtCantidad'. Luego, convierte ese valor en un número entero usando la función parseInt() y lo guarda en la variable n.

2. **if (isNaN(n) || n <= 0) {:** Esta línea verifica si n no es un número válido (usando isNaN()) o si es menor o igual a cero.

3. **alert('Ingrese cantidad de elementos válida');:** Si el valor ingresado no es un número válido o es menor o igual a cero, se muestra una alerta que indica al usuario que ingrese una cantidad de elementos válida.

4. **return:** Esto hace que la función se detenga en este punto y no continúe con el proceso de llenado del array si el valor ingresado no es válido.

5. **a = []:** Esta línea reinicia el array a, eliminando todos los elementos previamente almacenados en él. Esto asegura que el array esté limpio antes de llenarlo nuevamente.

6. **for (c = 0; c < n; c++):** Aquí comienza un ciclo for que se ejecutará n veces, donde n es la cantidad de elementos que el usuario ha ingresado previamente.

7. **elemento = parseInt(prompt('Ingrese el elemento')):** En cada iteración del ciclo, se muestra un cuadro de diálogo (prompt) que le pide al usuario que ingrese un valor. Luego, se convierte ese valor en un número entero usando parseInt() y se guarda en la variable elemento.

8. **a.push(elemento):** Después de convertir el valor ingresado en un número entero, se agrega ese valor al array a utilizando el método push(), lo que aumenta el tamaño del array y almacena el nuevo valor.

II. Procesamiento y mostrado del array:

- La función procesarArray simplemente muestra el contenido del array en la consola, en un cuadro de alerta y en un elemento HTML con el id "output". Utiliza el método join para crear una cadena separada por comas a partir de los elementos del array.

```
const procesarArray = () => {
    console.log('Array recibido:', a);
    alert('Array recibido:\n' + a.join(', '));
    var outputDiv = document.getElementById('output');
    outputDiv.innerHTML = 'Array recibido: ' + a.join(', ');
}
```

1. **console.log('Array recibido:', a):** Esta línea imprime en la consola del navegador el contenido del array a junto con el mensaje "Array recibido:". Esto es útil para depurar y verificar el estado del array en la consola del navegador.

2. **alert('Array recibido:\n' + a.join(', ')):** Esta línea muestra una alerta en el navegador con el contenido del array a, formateado como una cadena en la que los elementos están separados por comas y un espacio. El uso de \n introduce un salto de línea en el mensaje de la alerta.

3. **var outputDiv = document.getElementById('output'):** Aquí se busca un elemento HTML en el documento con el atributo id igual a "output". Esto generalmente se usa para modificar o mostrar contenido en ese elemento.

4. **outputDiv.innerHTML = 'Array recibido: ' + a.join(', '):** Esta línea modifica el contenido del elemento HTML identificado por "output". El contenido se establece como una cadena que indica "Array recibido:" seguido del contenido del array a, formateado de manera similar a la alerta anterior.

5. En resumen, la función procesarArray tiene como objetivo mostrar el contenido del array a en la consola del navegador, en una alerta para el usuario y en un elemento HTML con el id "output".

III. Ordenamiento por Selección:

- La función ordenarSeleccion realiza un ordenamiento por selección en el array. En este algoritmo, busca el valor mínimo en el segmento no ordenado del array y lo intercambia con el primer valor del segmento no ordenado. Esto se repite hasta que todo el array esté ordenado de manera ascendente.

```
const ordenarSeleccion = () => {
    for (let i = 0; i < a.length - 1; i++) {
        let minIndex = i;

        for (let j = i + 1; j < a.length; j++) {
            if (a[j] < a[minIndex]) {
                minIndex = j;
            }
        }
```

1. **for (let i = 0; i < a.length - 1; i++):** Inicia un bucle for que itera a través del array a. El índice i representa la posición del elemento actual en el proceso de ordenación. El bucle se ejecuta hasta a.length - 1 porque el último elemento ya estará en su lugar cuando los demás estén ordenados.

    - **let minIndex = i:** Se establece minIndex como i, lo que significa que asumimos inicialmente que el elemento en la posición i es el mínimo en el segmento no ordenado.

- El siguiente bucle for anidado, con la variable j, busca en el segmento no ordenado (a partir de i + 1) para encontrar el índice del elemento más pequeño.

```
        if (minIndex !== i) {
            // Intercambiar los valores en las posiciones i y minIndex
            let temp = a[i];
            a[i] = a[minIndex];
            a[minIndex] = temp;
        }
    }
    var outputDiv = document.getElementById('output2');
    outputDiv.innerHTML = 'Array recibido: ' + a.join(', ');
    alert('Array ordenado por selección: ' + a);
}
```

2. **if (minIndex !== i) { ... }:** Después de encontrar el elemento mínimo en el segmento no ordenado, se verifica si es diferente del elemento actual en la posición i. Si son diferentes, esto significa que hay un elemento más pequeño que el actual en el segmento no ordenado, y se procede a intercambiar los elementos.

3. **let temp = a[i]; a[i] = a[minIndex]; a[minIndex] = temp:** Se realiza el intercambio de valores entre los elementos en las posiciones i y minIndex utilizando una variable temporal (temp) para facilitar el intercambio.

4. **var outputDiv = document.getElementById('output2'):** Aquí se busca un elemento HTML en el documento con el atributo id igual a "output2". Esto se hace para modificar o mostrar contenido en ese elemento.

5. **outputDiv.innerHTML = 'Array recibido: ' + a.join(', '):** Similar al fragmento anterior, esta línea modifica el contenido del elemento HTML identificado por "output2", mostrando el contenido del array a después de haber sido ordenado por selección.

6. **alert('Array ordenado por selección: ' + a):** Muestra una alerta en el navegador con el contenido del array a después de haber sido ordenado por selección.

En resumen, la función ordenarSeleccion implementa el algoritmo de ordenación por selección en el array a y muestra el array antes y después de la ordenación en elementos HTML y cuadros de alerta.

IV. Borrado del Array y Contenido HTML:

- La función borrarArray simplemente borra todos los elementos del array a, además de limpiar el campo de entrada (txtCantidad) y los contenidos mostrados en los elementos HTML con los ids "output" y "output2". Muestra un cuadro de alerta para confirmar que el array y la cantidad han sido borrados.

```
const borrarArray = () => {
    a = [];

    // Borrar el contenido del input "txtCantidad"
    var inputCantidad = document.getElementsByName('txtCantidad')[0];
    inputCantidad.value = '';

    // Borrar el contenido mostrado en el div de salida
    var outputDiv = document.getElementById('output');
    outputDiv.innerHTML = ('Array borrado');
    var outputDiv = document.getElementById('output2');
    outputDiv.innerHTML = ('Array borrado');
    alert('Array y cantidad borrados');
}
```

1. **a = []:** Esta línea asigna un array vacío a la variable a, lo que elimina todos los elementos previamente almacenados en el array.

2. **var inputCantidad = document.getElementsByName('txtCantidad')[0]:** Aquí se busca el campo de entrada en el HTML con el nombre 'txtCantidad' y se almacena en la variable inputCantidad.

3. i**nputCantidad.value = '':** Esto establece el valor del campo de entrada inputCantidad en una cadena vacía, eliminando así cualquier valor que haya sido ingresado previamente.

4. **var outputDiv = document.getElementById('output'):** Aquí se busca un elemento HTML en el documento con el atributo id igual a "output". Esto generalmente se hace para modificar o mostrar contenido en ese elemento.

5. **outputDiv.innerHTML = ('Array borrado'):** Esta línea cambia el contenido del elemento HTML identificado por "output" para mostrar "Array borrado". Esto limpia el contenido que previamente mostraba el array.

6. **var outputDiv = document.getElementById('output2'):** Se busca otro elemento HTML con el atributo id igual a "output2". Esto se hace para modificar o mostrar contenido en ese elemento.

7. **outputDiv.innerHTML = ('Array borrado'):** Similar al punto 5, esto cambia el contenido del elemento HTML identificado por "output2" para mostrar "Array borrado", limpiando el contenido que previamente mostraba el array ordenado.

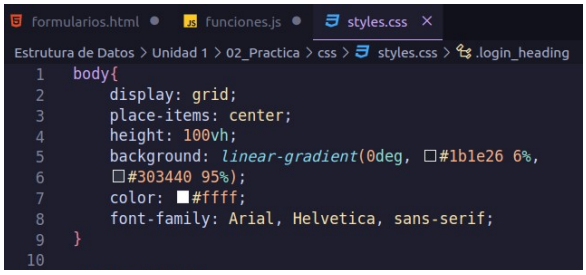
8. **alert('Array y cantidad borrados'):** Muestra una alerta en el navegador que indica que el array y la cantidad han sido borrados.

En resumen, la función borrarArray se encarga de limpiar el array a, borrar el contenido del campo de entrada y limpiar los contenidos mostrados en los elementos HTML con los ids "output" y "output2". También muestra una alerta para informar al usuario que el array y la cantidad han sido borrados.

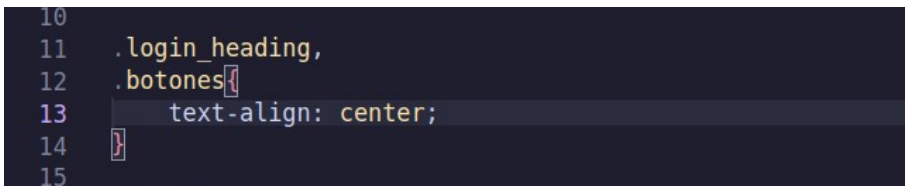
CÓDIGO DENTRO DEL ARCHIVO CSS

Ahora definiremos el diseño y la apariencia de nuestra página web para la compra de Hamburguesas.

1. En la primera parte definiremos los estilos generales del cuerpo (`body`) de la página:
   - `display: grid;`: Hace que el cuerpo utilice el modelo de diseño en cuadrícula (grid).
   - `place-items: center;`: Centra los elementos en el centro tanto vertical como horizontalmente.
   - `height: 100vh;`: Establece la altura del cuerpo (viewport) al 100% del alto de la ventana del navegador, es decir, ocupa todo el espacio vertical disponible.
   - `background: linear-gradient(0deg, #1b1e26 6%, #303440 95%);`: Crea un fondo con un degradado lineal que va desde el color #1b1e26 al 6% del alto de la ventana, hasta el color #303440 al 95% del alto de la ventana.
   - `color: #ffff;`: Define el color de texto como blanco (#ffffff).
   - `font-family: Arial, Helvetica, sans-serif;`: Establece la fuente utilizada para el texto como Arial, Helvetica o fuentes sans-serif genéricas.



2. Luego, creamos la clase `.login_heading` que se refiere al encabezado del formulario de inicio de sesión y `.botones` que hará referencia a los botones presentes en el formulario. Ambos tienen `text-align: center;`, lo que centra el texto en el centro.

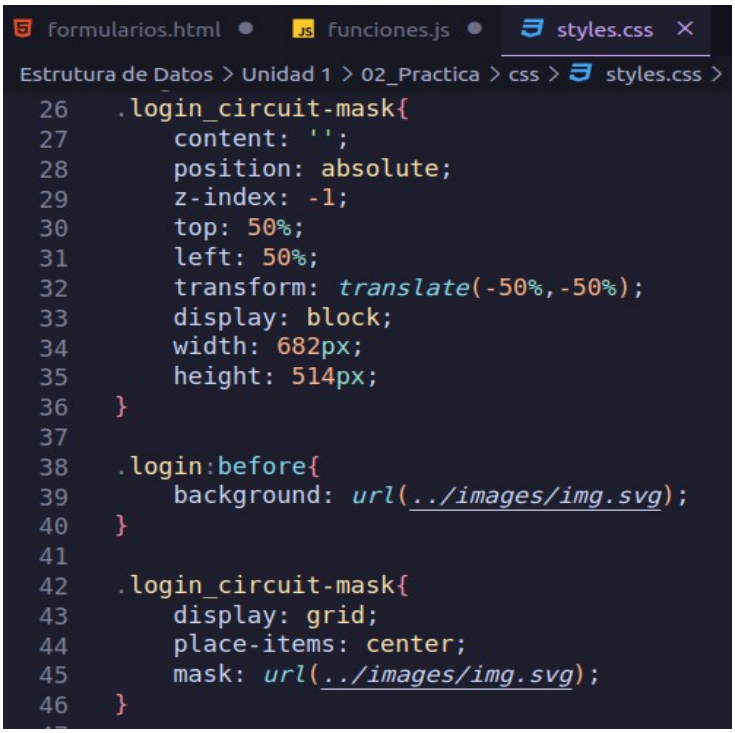


3. La clase `.login` definirá el diseño y los estilos del formulario de inicio de sesión:
   - position: relative;`: Hace que los elementos dentro del formulario sean posicionados de manera relativa con respecto a su posición original.
   - `width: 360px;`: Establece el ancho del formulario a 360 píxeles.
   - `height: 300px;`: Establece la altura del formulario a 300 píxeles.
   - `padding: 10px;`: Agrega un espacio de relleno de 10 píxeles alrededor del contenido dentro del formulario.
   - `border-radius: 10px;`: Crea bordes redondeados en las esquinas del formulario con un radio de 10 píxeles.
   - background: #15171d;`: Establece un fondo de color #15171d para el formulario.

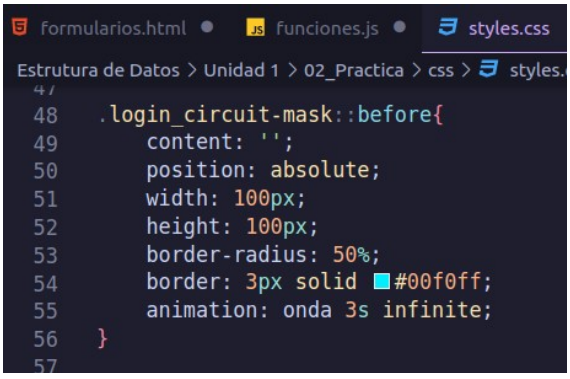
```
15
16  .login{
17      position: relative;
18      width: 360px;
19      height: 300px;
20      padding: 10px;
21      border-radius: 10px ;
22      background: #15171d;
23  }
24
```

4. Ahora creamos las pseudoclases `:before` son utilizadas para agregar elementos adicionales antes de los contenidos de los elementos con la clase `.login` y `.login_circuit-mask`.
   - `.login:before`: Agrega un elemento antes del contenido del formulario de inicio de sesión.
   - `.login_circuit-mask`: Agrega un elemento antes del contenido del elemento con la clase `.login_circuit-mask`.

5. `.login:before` y `.login_circuit-mask` tendrán estilos similares:
   - - `content: '';`: Define el contenido del elemento creado.
   - - `position: absolute;`: Establece una posición absoluta en relación con el primer elemento padre que tiene una posición no estática.
   - - `z-index: -1;`: Define el valor de la propiedad z-index para que los elementos estén por debajo del contenido del formulario.
   - - `top: 50%; left: 50%; transform: translate(-50%, -50%);`: Centra el elemento creado horizontal y verticalmente en el formulario.
   - - `width: 682px; height: 514px;`: Establece el ancho y la altura del elemento creado.
   - - `background: url(../images/img.svg);`: Establece una imagen de fondo que se encentra en el directorio "../images/" con el nombre "img.svg".



6. La pseudoclase `.login_circuit-mask::before` agregará un círculo animado sobre la imagen de fondo:
   - `content: '';`: Define el contenido del círculo creado.
   - `position: absolute;`: Establece una posición absoluta en relación con el primer elemento padre que tiene una posición no estática.
   - `width: 100px; height: 100px;`: Establece el ancho y la altura del círculo.
   - `border-radius: 50%;`: Crea un círculo alrededor del elemento con un radio de 50% (un círculo perfecto).
   - `border: 3px solid #00f0ff;`: Establece un borde de color #00f0ff alrededor del círculo.
   - `animation: onda 3s infinite;`: Aplica una animación llamada "onda" al círculo, con una duración de 3 segundos y se repite infinitamente.



7. Por último, creamos la definición de la animación `@keyframes onda`:
   - `0%`: Define el estado inicial de la animación (escala de 1).
   - `100%`: Define el estado final de la animación (escala de 8).

```
57
58  @keyframes onda {
59      0%{
60          transform: scale(1);
61      }
62      100%{
63          transform: scale(8);
64      }
65
66  }
```

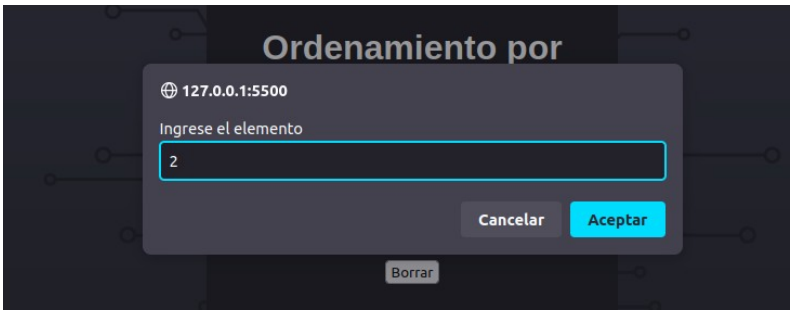
La animación hace que el círculo aumente de tamaño (escala) desde 1 hasta 8 durante un período de 3 segundos, y luego vuelve al tamaño inicial y se repite infinitamente.

PAGINA WEB

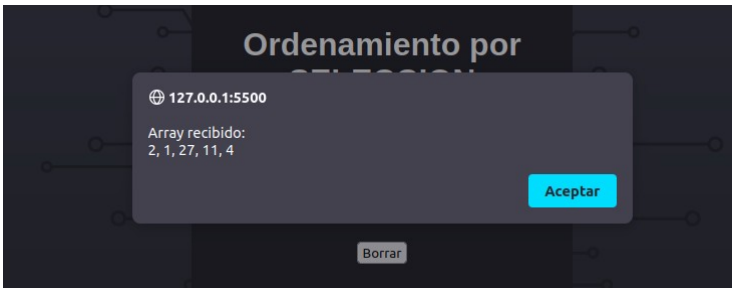
1. Diseño principal de la pagina implementando el método de ordenamiento.

2. Alerta al presionar el botón de pedir elementos sin a ver ingresado el tamaño del array.

3. Prompt para pedir cada elemento del array.

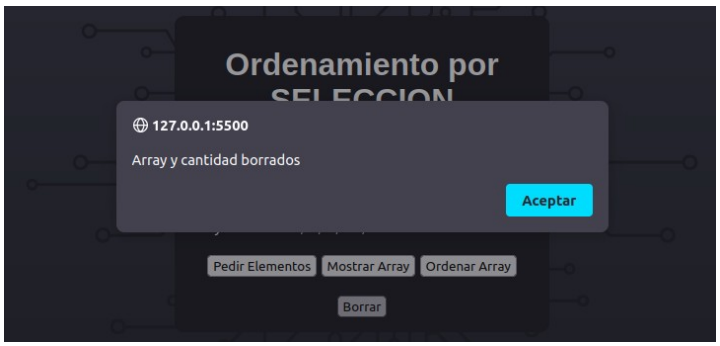


4. Alerta al presionar el array, ademas de mostrarlo en mi elemento div.



5. Alerta del array ordenado mostrándolo en un div.

6. Alerta al presionar el botón de borrar.



CONCLUSIÓN

El método de ordenamiento por selección es una técnica fundamental en la programación para organizar elementos en un orden específico dentro de un array. Aunque no es el algoritmo más eficiente para grandes conjuntos de datos, el ordenamiento por selección brinda una comprensión sólida de los conceptos básicos de ordenación y es útil para propósitos educativos y cuando el tamaño de los datos es pequeño. Sin embargo, sigue siendo una herramienta valiosa para enseñar conceptos de algoritmos y lógica de programación, y puede ser útil en situaciones específicas donde la cantidad de datos es pequeña o donde se busca una implementación sencilla y fácil de entender.

En JavaScript, el algoritmo de ordenamiento por selección se implementa en un ciclo anidado, donde se busca el elemento mínimo en el segmento no ordenado y se intercambia con el primer elemento del segmento no ordenado. Esto se repite hasta que todo el array esté ordenado. Aunque no es eficiente para grandes conjuntos de datos debido a su complejidad de tiempo O(n^2), el algoritmo es sencillo de entender y de implementar.