Universidad Tecnológica Metropolitana

Estructuras de Datos Aplicadas

ISC. Ruth Betsaida Martínez Domínguez, MGTI

Prácticas 9-10

Soberanis Acosta Jimena Monserrat

Desarrollo de Software Multiplataforma

Cuarto Cuatrimestre

4°B

Parcial I

Jueves, 28 de septiembre de 2023

PRACTICA 9

Primeramente, para este práctica definí cada matriz con lo que se trabajará

```
const matriz1 = [
    [10, 5],
    [8, 2]
];

const matriz2 = [
    [2, 4],
    [6, 8]
];
```

Después, creé una función para cada operación:

Para cada una de las funciones se creó una nueva matriz llamada resultado(nombre de la operación), el cual se utilizará para almacenar el resultado de dicha operación.

Se creó bucles for anidades para que recorran tanto las filas como columnas de las matrices, este bucle se repetirá solamente 2 veces, pues ese es el tamaño máximo.

La operación funciona de tal modo que se sumará cada elemento de matriz1[fila][columna] con su correspondiente posición en la matriz2[fila][columna], y este resultado se guardará en resultadoSuma y lo devolverá.

```
function sumaMatrices(){
    const resultadoSuma =[];

    for (let fila = 0; fila < 2; fila++){
        resultadoSuma.push([]);

        for (let columna = 0; columna < 2; columna++){
            resultadoSuma[fila].push(matriz1[fila][columna]+ matriz2[fila][columna]);
        }
    }
    return resultadoSuma;
}
```

La siguiente operación funciona de manera similar, solo que en esta se realizará una resta de la matriz1[fila][columna] con su correspondiente posición en la matriz2[fila][columna], este resultado se guardará en resultadoResta y se devolverá cuando termine el bucle.

```
function restaMatrices(){
    const resultadoResta =[];

    for (let fila = 0; fila < 2; fila++){
        resultadoResta.push([]);

        for (let columna = 0; columna < 2; columna++){
            resultadoResta[fila].push(matriz1[fila][columna] - matriz2[fila][columna]);
        }
    }
    return resultadoResta;
}
```

De la misma manera funciona la operación del producto, solo que en este en vez de que se realice una suma o resta, se realiza una multiplicación de la matriz1[fila][columna] con su correspondiente posición en la matriz2[fila][columna],

```
function productoMatrices(){
    const resultadoProducto =[];

    for (let fila = 0; fila < 2; fila++){
        resultadoProducto.push([]);

        for (let columna = 0; columna < 2; columna++){
            resultadoProducto[fila].push(matriz1[fila][columna] * matriz2[fila][columna]);
        }
    }
    return resultadoProducto;
}
```

En esta última operación se realiza una comprobación, pues la matriz2 es la divide a la matriz 1, y en tal caso que la matriz divisora (matriz2) sea igual a cero, se mostrará en la consola que no se pude dividir entre cero y se retornará un null como resultado; en caso contrario, si es diferente de 0 se realiza la operación normalmente, pero el resultado se dará con 2 decimales.

```
function divisonMatrices(){
    const resultadoDivision =[];

    for (let fila = 0; fila < 2; fila++){
        resultadoDivision.push([]);

        for (let columna = 0; columna < 2; columna++){
            if(matriz2[fila][columna]!==0){
                const resultado = matriz1[fila][columna] / matriz2[fila][columna];

                const resultadoDivisionRedondeado = resultado.toFixed(2);
                resultadoDivision[fila].push(parseFloat(resultadoDivisionRedondeado));
            }else{
                console.log("No se puede dividir entre cero.");
                return null;
            }
        }
    }
    return resultadoDivision;
}
```

Después de realizar los cálculos de las operaciones entre las 2 matrices, se mostrará primero las matrices con las que se trabajaron, y después los resultados de cada una de las operaciones, en cada apartado se llamó a las funciones con los parámetros de matriz1 y matriz2.

```
68    console.log("Matriz 1: ");
69    console.log(matriz1);
70
71    console.log("Matriz 2: ");
72    console.log(matriz2);
73    console.log('\t');
74
75    console.log("Suma de las matrices:");
76    console.log(sumaMatrices(matriz1, matriz2));
77    console.log('\t');
78
79    console.log("Resta de las matrices:");
80    console.log(restaMatrices(matriz1, matriz2));
81    console.log('\t');
82
83    console.log("Producto de las matrices:");
84    console.log(productoMatrices(matriz1, matriz2));
85    console.log('\t');
86
87    console.log("Division de las matrices:");
88    console.log(divisonMatrices(matriz1, matriz2));
89    console.log('\t');
90
```

Ejemplo:

En la consola, con las matrices definidas arriba, estos son los resultados:

```
PS C:\Users\Jimena\Documents\UTM\Cuatrimestre 4\Estructura de datos aplicados\PARCIAL I\Practica9> node script9.js
Matriz 1:
[ [ 10, 5 ], [ 8, 2 ] ]
Matriz 2:
[ [ 2, 4 ], [ 6, 8 ] ]

Suma de las matrices:
[ [ 12, 9 ], [ 14, 10 ] ]

Resta de las matrices:
[ [ 8, 1 ], [ 2, -6 ] ]

Producto de las matrices:
[ [ 20, 20 ], [ 48, 16 ] ]

Division de las matrices:
[ [ 5, 1.25 ], [ 1.33, 0.25 ] ]
```

Primero se muestran las matrices y después cada uno de los resultados de las operaciones.

PRÁCTICA 10

Comencé en un HTML simple en donde se crean dos formularios, uno para ingresar el tamaño de la matriz y otro para ingresar los valores de la matriz, y dos botones para enviar los datos.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Matriz Cuadrada</title>
</head>
<body>
    <form id="size-form">
        <label for="matrix-size">Tamaño de la matriz cuadrada:</label>
        <input type="number" id="matrix-size" required>
        <input type="submit" value="Ingresar Tamaño">
    </form>
    <div id="matrix-container"></div>
    <form id="input-form">
        <h2>Ingrese los valores de la matriz (separados por comas):</h2>
        <input type="text" id="matrix-values" required>
    </form>

    <button id="solve-button">Resolver</button>

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

Definí una clase llamada Matrix que se utiliza para crear y manipular las matrices cuadradas. Comienza con un constructor que toma el argumento size que representa el tamaño de la matriz cuadrada. Se asigna el tamaño pasado como argumento a this.size, se crea una matriz vacía utilizando el método createMatrix a this.matrix, también se llama al método renderMatrix para mostrar la matriz en la página web. Se obtienen referencias a elementos del HTML como el formulario de entrada de valores y al botón para resolverlo, y de último se crea un llamado a onSolveButtonClick cuando se hace click.

```javascript
class Matrix {
    constructor(size) {
        this.size = size;
        this.matrix = this.createMatrix();
        this.renderMatrix();
        this.inputForm = document.getElementById('input-form');
        this.solveButton = document.getElementById('solve-button');
        this.solveButton.addEventListener('click', () => this.onSolveButtonClick());
    }
```

Luego, está el método createMatrix que crea una matriz vacía de tamaño size x size, donde todos los elementos se inicializan a 0. Utiliza bucles for para recorrer las filas y columnas de la matriz, y devuelve la matrix creada.

```
createMatrix() {
    const matrix = [];
    for (let fila = 0; fila < this.size; fila++) {
        matrix[fila] = [];
        for (let columna = 0; columna < this.size; columna++) {
            matrix[fila][columna] = 0;
        }
    }
    return matrix;
}
```

Después, creé el método renderMatrix que se utiliza para actualizar la representación de la matriz en la página web. Borra el contenido actual del contenedor de la matriz, crea una tabla HTML y llena las celdas de la tabla con los valores de la matriz actual.

```
renderMatrix() {
    const matrixContainer = document.getElementById('matrix-container');
    matrixContainer.innerHTML = '';
    const table = document.createElement('table');
    for (let fila = 0; fila < this.size; fila++) {
        const row = document.createElement('tr');
        for (let columna = 0; columna < this.size; columna++) {
            const cell = document.createElement('td');
            cell.textContent = this.matrix[fila][columna];
            row.appendChild(cell);
        }
        table.appendChild(row);
    }
    matrixContainer.appendChild(table);
}
```

Creé el método solver que realiza la resolución de la matriz:

Obtiene los valores de la matriz de matrix-values, y los convierte en un array de números.

Comienza con un if que primero verifica si el número de valores ingresados coinciden con el tamaño de la matriz, sino muestra una alerta.

Después, se llena la matriz actual con esos valores ingresados y de ultimo llama renderMatrix para actalizar la representación,

```
38      async solve() {
39          const valuesInput = document.getElementById('matrix-values').value;
40          const values = valuesInput.split(',').map(val => parseFloat(val.trim()));
41
42          if (values.length !== this.size * this.size) {
43              alert('Ingrese la cantidad correcta de valores para la matriz.');
44              return;
45          }
46
47          let index = 0;
48          for (let fila = 0; fila < this.size; fila++) {
49              for (let columna = 0; columna < this.size; columna++) {
50                  this.matrix[fila][columna] = values[index];
51                  index++;
52              }
53          }
54
55          this.renderMatrix();
```

Comprueba si todos los elementos de la matriz son iguales. Si es así, reemplaza la matriz con una matriz identidad.

```
60          if (allEqual) {
61              // Si todos los elementos son iguales, reemplazar la matriz con una matriz identidad
62              for (let fila = 0; fila < this.size; fila++) {
63                  for (let columna = 0; columna < this.size; columna++) {
64                      this.matrix[fila][columna] = (fila === columna) ? 1 : 0;
65                  }
66                  this.renderMatrix();
67              }
68          } else {
69              // Resolver con el método de Gauss
```

Si no son iguales, resuelve la matriz utilizando el método de Gauss, realizando operaciones de eliminación gaussiana. Y al final muestra la matrix.

```
        } else {
            // Resolver con el método de Gauss
            for (let fila = 0; fila < this.size; fila++) {
                let pivot = this.matrix[fila][fila];
                for (let columna = 0; columna < this.size; columna++) {
                    this.matrix[fila][columna] /= pivot;
                }
                for (let k = 0; k < this.size; k++) {
                    if (k !== fila) {
                        let factor = this.matrix[k][fila];
                        for (let columna = 0; columna < this.size; columna++) {
                            this.matrix[k][columna] -= factor * this.matrix[fila][columna];
                        }
                    }
                }
                this.renderMatrix();
            }
        }
    }
```

El siguiente método simplemente llama al método solve cuando se hace clic en el botón de resolución.

```
    onSolveButtonClick() {
        this.solve();
    }
}
```

Finalmente, fuera de la clase Matrix, se hace un llamado al formulario y hace lo siguiente:

Previene el comportamiento por defecto del formulario para evitar que la página se recargue.

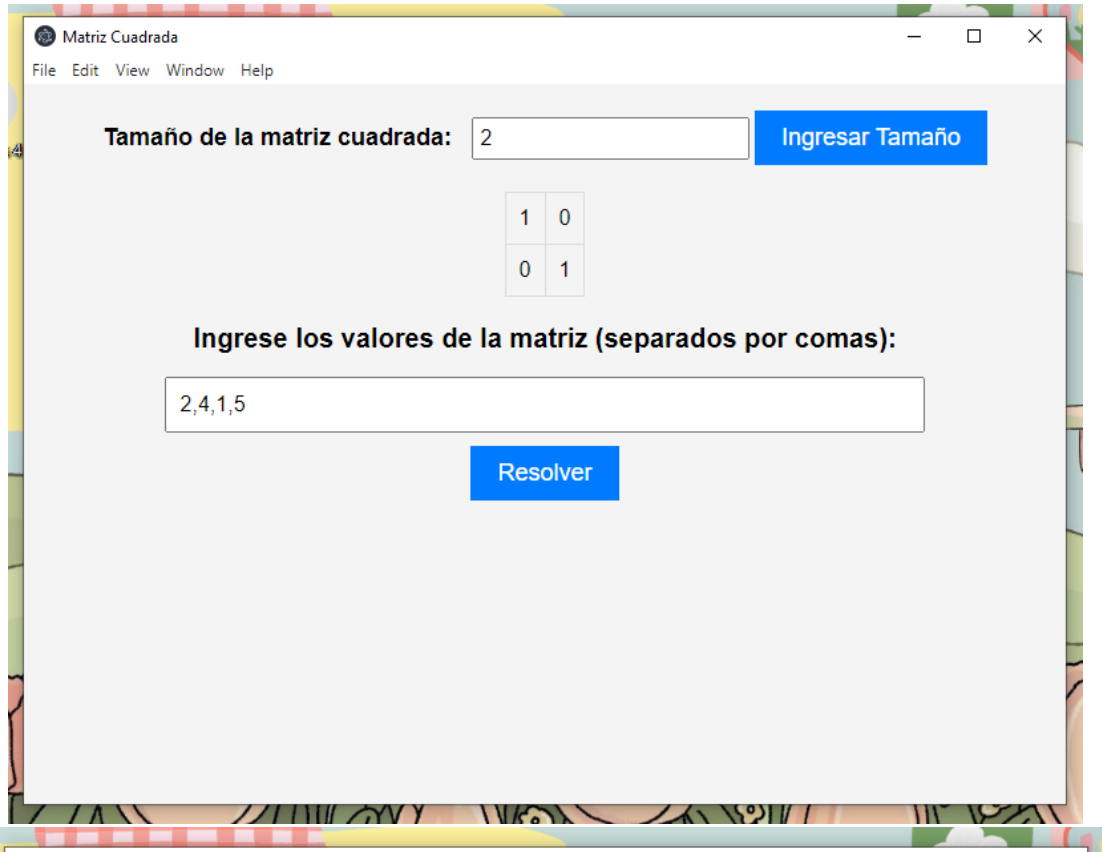
Obtiene el tamaño ingresado en el campo de entrada de tamaño.

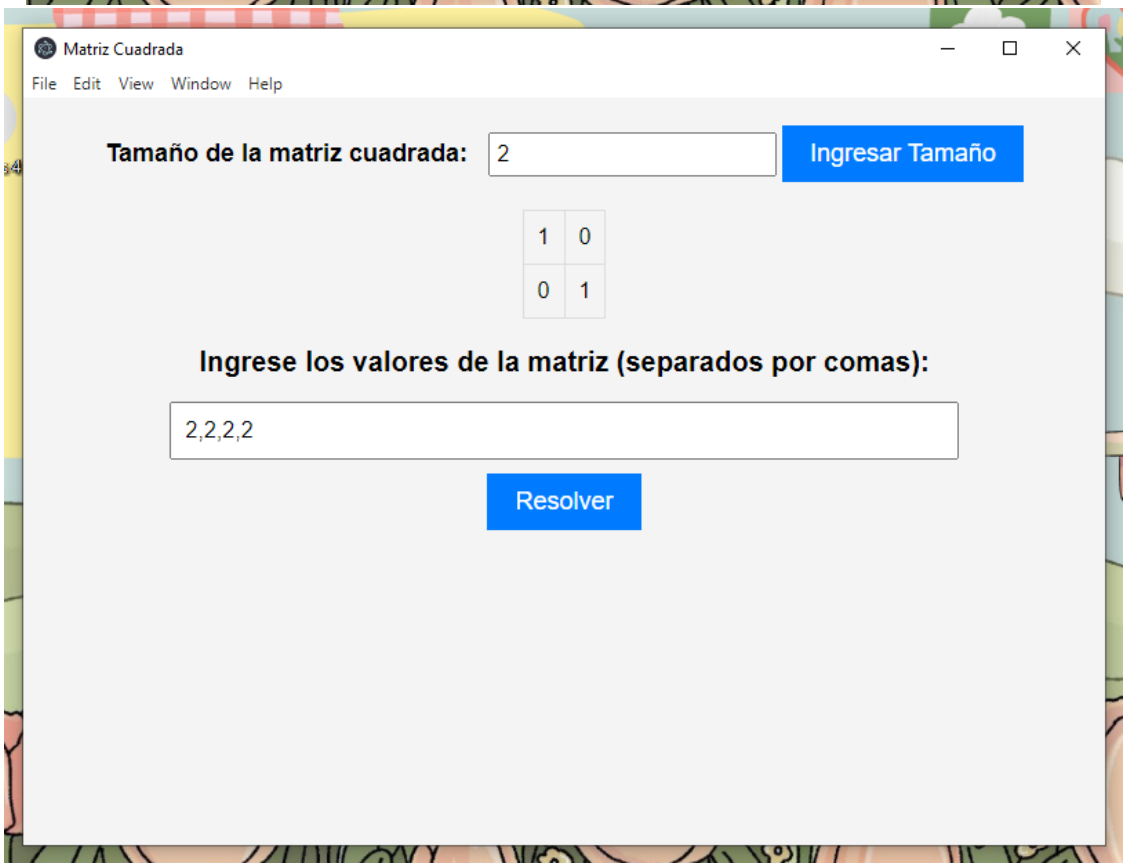
Si el tamaño es un número válido y mayor que cero, crea una instancia de la clase Matrix con ese tamaño. Si no es válido, muestra una alerta.

```
document.getElementById('size-form').addEventListener('submit', (event) => {
    event.preventDefault();
    const sizeInput = document.getElementById('matrix-size');
    const size = parseInt(sizeInput.value);
    if (!isNaN(size) && size > 0) {
        const matrix = new Matrix(size);
    } else {
        alert('Ingrese un tamaño válido para la matriz cuadrada.');
    }
});

```

Decidí colocarle un diseño con CSS y con ayuda Electron pude levantar la aplicación, el cual quedó de esta manera:


Matriz Cuadrada
File Edit View Window Help
Tamaño de la matriz cuadrada:
2
Ingresar Tamaño
1 0
0 1
Ingrese los valores de la matriz (separados por comas):
2,4,1,5
Resolver


Matriz Cuadrada
File Edit View Window Help
Tamaño de la matriz cuadrada:
2
Ingresar Tamaño
1 0
0 1
Ingrese los valores de la matriz (separados por comas):
2,2,2,2
Resolver