Universidad Tecnológica Metropolitana

Estructuras de Datos Aplicadas

ISC. Ruth Betsaida Martínez Domínguez, MGTI

Prácticas 7-8

Soberanis Acosta Jimena Monserrat

Desarrollo de Software Multiplataforma

Cuarto Cuatrimestre

4°B

Parcial I

Viernes, 22 de septiembre de 2023

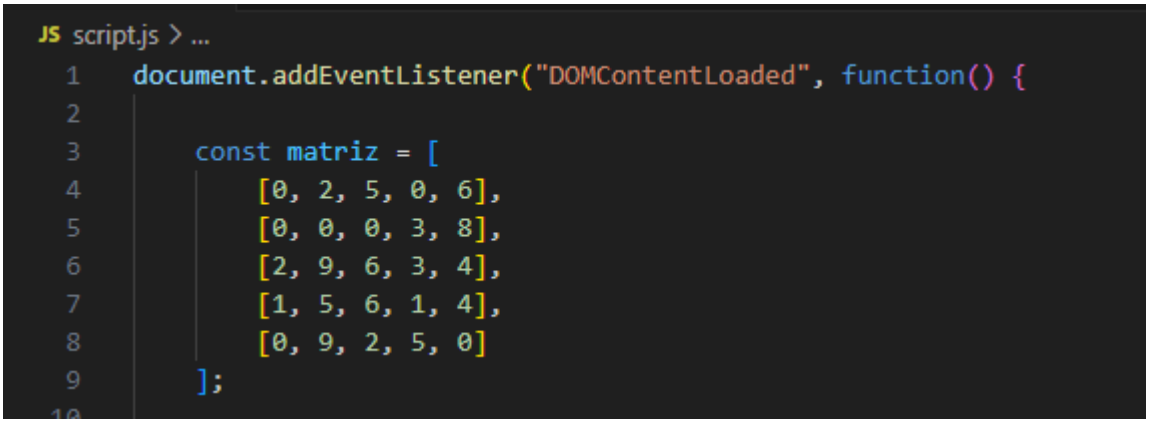
PRÁCTICA 7

Para la base de la práctica fue el html:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Contador de Ceros</title>
</head>
<body>
    <div class="container">
        <h1>Contador de Ceros por Fila</h1>

        <h2> Tarea 7</h2>
        <table>
            <thead>
                <tr>
                    <th>Fila</th>
                    <th>Cantidad de Ceros</th>
                    <th>Matriz</th>
                </tr>
            </thead>
            <tbody id="resultado"></tbody>
        </table>

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

En este mismo, creé la tabla que se utilizará para poder mostrar los resultados del contador de ceros por fila de la matriz dada.

Luego, comencé con el script, en el cual primero declaré una matriz, el cual será el que se analice la cantidad de ceros por fila.



```javascript
document.addEventListener("DOMContentLoaded", function() {

    const matriz = [
        [0, 2, 5, 0, 6],
        [0, 0, 0, 3, 8],
        [2, 9, 6, 3, 4],
        [1, 5, 6, 1, 4],
        [0, 9, 2, 5, 0]
    ];
```

Después, creé la clase ContadorCeros, este tiene un constructor que recibe el parámetro Matriz, este se almacena como una propiedad de clase con this.matriz=matriz, lo que permite usar matriz desde cualquier lado.

Está el método contadorCerosPorFila que recibe el argumento fila (de la matriz), este método usa el método filter, el cual crea un nuevo array que toma todos los elementos que cumplen la condición de ser exactamente igual a 0 y se devuelve el número de elementos (ceros) contados.

El método contadorCerosPorTodasFilas, el cual primero crea un array vacío de resultado (el número de ceros), después de inicia un bucle que recorre la matriz buscando a los ceros con el contadorCerosPorFila(i), donde i es el 0 y se almacena en resultado, el cual también se devuelve con el return.

```
class ContadorCeros{
    constructor(matriz){
        this.matriz=matriz;
    }

    contadorCerosPorFila(fila){
        return this.matriz[fila].filter(num => num === 0).length;
    }

    contadorCerosPorTodasFila(){
        const resultado = [];

        for (let i = 0; i < this.matriz.length; i++){
            const contador = this.contadorCerosPorFila(i);
            resultado.push(contador);
        }
        return resultado;
    }
```

Por último, está mostrarResultados(), que muestra los resultados en una tabla en el HTML. Se crea un const cantidadCeros que utiliza el método contadorCerosPorTdoasFilas que obtiene el array respectivo. Se crea un bucle que crea cada fila de la tabla. Dentro de cada fila también se forma cada celda de la misma, incluyendo el número de la fila, la cantidad de ceros y los elementos de la fila.

```
    mostrarResultados() {
        const tablaResultado = document.getElementById("resultado");
        const cantidadCeros = this.contadorCerosPorTodasFila();

        for (let i = 0; i < cantidadCeros.length; i++) {
            const fila = document.createElement("tr");
            const indexFila = document.createElement("td");
            const contadorCelda = document.createElement("td");
            const matrizCelda = document.createElement("td");

            indexFila.textContent = `Fila ${i + 1}`;
            contadorCelda.textContent = cantidadCeros[i];
            matrizCelda.textContent = this.matriz[i].join(",");

            fila.appendChild(indexFila);
            fila.appendChild(contadorCelda);
            fila.appendChild(matrizCelda);

            tablaResultado.appendChild(fila);
        }
    }
}
```

Por último, en el script, creo una nueva instancia que termina llamando al método mostrarResultados().

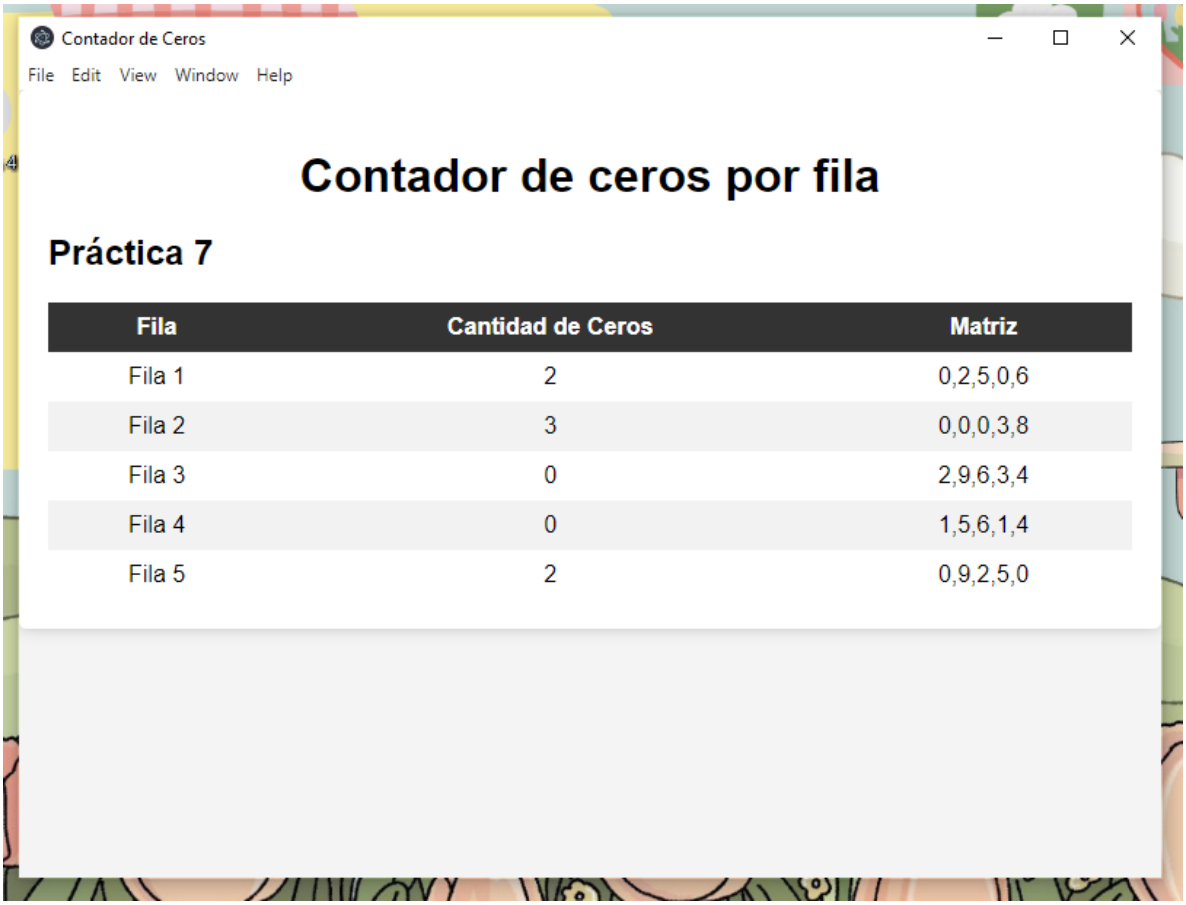
Solo faltaba darle un formato con css:

```css
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
}

.container {
    max-width: 800px;
    margin: 0 auto;
    padding: 20px;
    background-color: #fff;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
}

h1 {
    text-align: center;
    margin-bottom: 20px;
}
```

```css
table {
    width: 100%;
    border-collapse: collapse;
}

th, td {
    padding: 8px 12px;
    text-align: center;
}

th {
    background-color: #333;
    color: #fff;
}

tr:nth-child(even) {
    background-color: #f2f2f2;
}
```

Y luego, lo hice aplicación con el framework Electron:

Solo muestra la tabla, pero si cambio el arreglo desde el script, la cantidad cambiará:


Contador de Ceros
File Edit View Window Help
Contador de ceros por fila
Práctica 7
Fila
Cantidad de Ceros
Matriz
Fila 1
2
0,2,5,0,6
Fila 2
3
0,0,0,3,8
Fila 3
0
2,9,6,3,4
Fila 4
0
1,5,6,1,4
Fila 5
2
0,9,2,5,0

PRÁCTICA 8

Comencé con el HTML, como siempre:

Para la base de la práctica fue el 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">
    <link rel="stylesheet" href="styles.css">
    <title>Cuadro Mágico</title>
</head>
<body>
    <h1>Verificador de Cuadro Mágico</h1>

    <input type="text" id="tamañoInput" placeholder="Ingrese el tamaño del cuadro mágico (n)"><br>
    <button id="ingresarDatosBtn">Ingresar Datos del Cuadro Mágico</button>

    <div id="matrizIngresada"></div>
    <div id="resultado"></div>

    <div id="inputContainer" style="display: none;">
        <h2>Ingrese los valores:</h2>
        <table id="inputTable"></table>
        <button id="calcularBtn">Calcular</button>
    </div>

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

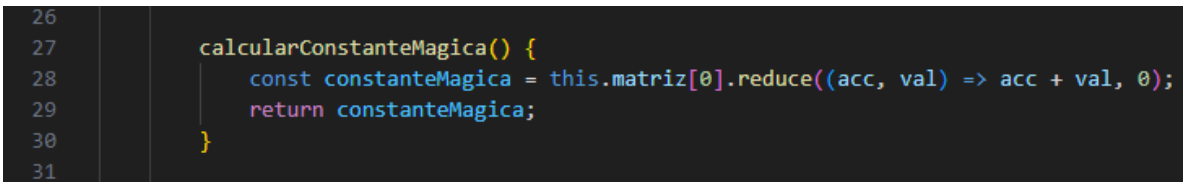
En este mismo, comienza con lo primero que se verá, que es un input para ingresar el tamaño del cuadro y debajo el botón para ingresar los datos, cuando se le da click es cuando las otras divisiones aparecerán, tales como una pequeña tabla para poder ingresar los valores en cada celda, un botón para calcular, y después nos dará la matriz y el resultado.

Proseguí con el script, el cual solo puede iniciar hasta que el documento haya cargado por completo.

Después, está un evento listener que buscar el botón ingresarDatosBtn, que al hacer click en él, se llama a la función mostrarInputs.

Creé la clase CuadroMagico, en cual tiene el constructor que recibe el argumento n que representará el tamaño del cuadro. De igual manera, se crea matriz, el cual es un array inicializado en 0, y esta crea una matriz cuadrada, multiplica la nxn, por ejemplo, si n es = 2, el tamaño del cuadro será 2x2 y tendrá 4 celdas.

Después, creé el método llenarMatriz(), este funciona cuando la tabla para ingresar el dato de cada celda aparece, recibe un array llamado valores, los cuales son los que llenarán a la matriz y son proporcionados por el usuario. El idx es como un índice que recorre el array valores, y con el for recorre las filas (i) y las columnas(j) de los datos ingresados en la tablita que apareció, cada recorrido se obtienen los elementos del array y los almacena en la variable valor, el bucle continua hasta que se llenen todas las celdas de la matriz.



El siguiente método calcularConsanteMagica suma todos los datos de la primera fila y obtiene lo que debería ser la constante Magica real del cuadro, pues cualquier suma ya sea de una columna o una diagonal debería ser la misma a la de una fila, por lo que este método es la base de comparación a otras sumas.

```
esCuadroMagico() {
    const constanteMagica = this.calcularConstanteMagica();

    for (let i = 0; i < this.n; i++) {
        const sumaFila = this.matriz[i].reduce((acc, val) => acc + val, 0);

        const sumaColumna = this.matriz.reduce((acc, fila) => acc + fila[i], 0);
        if (sumaFila !== constanteMagica || sumaColumna !== constanteMagica) {
            return false;
        }
    }

    const sumaDiagonalPrincipal = this.matriz.reduce((acc, fila, i) => acc + fila[i], 0);
    if (sumaDiagonalPrincipal !== constanteMagica) {
        return false;
    }

    const sumaDiagonalSecundaria = this.matriz.reduce((acc, fila, i) => acc + fila[this.n - 1 - i], 0);
    if (sumaDiagonalSecundaria !== constanteMagica) {
        return false;
    }

    return true;
}
```

El método que sigue es la comprobación de si el cuadro es mágico, el cual llamada al método calcularConstanteMagica, este mismo ayuda comparándose así mismo con la suma de los elementos de las filas, la suma de columna y la suma de diagonales, y si el número es el mismo (que la constante mágica), entonces es un cuadro mágico y devuelve un true, en caso contrario devuelve false.

```
generarTabla() {
    const tabla = document.createElement("table");
    for (let i = 0; i < this.n; i++) {
        const fila = document.createElement("tr");
        for (let j = 0; j < this.n; j++) {
            const celda = document.createElement("td");
            celda.textContent = this.matriz[i][j];
            fila.appendChild(celda);
        }
        tabla.appendChild(fila);
    }
    return tabla;
}
}
```

El método siguiente generarTabla, como su nombre lo dice, genera la tabla desde el mismo script, primero creando un elemento llamado tabla, y con el bucle for y el valor de n (tamaño del cuadro nxn) crea cada parte del cuadro como sus filas (tr) y con el otro for es cuando crea las celdas (td). Después se le asigna a la celdas que tendrán un futuro contenido dependiendo de su posición (i,j), y poco a poco se va creando las filas con sus respectivas celdas.

```
function mostrarInputs() {
    const tamañoInput = document.getElementById("tamañoInput");
    const tamaño = parseInt(tamañoInput.value);

    if (!isNaN(tamaño) && tamaño > 0) {
        const inputContainer = document.getElementById("inputContainer");
        const inputTable = document.getElementById("inputTable");
        inputTable.innerHTML = "";

        for (let i = 0; i < tamaño; i++) {
            const row = document.createElement("tr");
            for (let j = 0; j < tamaño; j++) {
                const cell = document.createElement("td");
                const input = document.createElement("input");
                input.type = "number";
                cell.appendChild(input);
                row.appendChild(cell);
            }
            inputTable.appendChild(row);
        }
```

Después existe la función mostrarInputs, que crea un input para escribir el tamaño del cuadro y luego se convierte a la variable tamaño.

Se verifica si es un número y también mayor a 0 para proseguir, entonces se prepara el espacio para las tablas para ingresar los datos, así como el botón para enviar esos datos.

El bucle for hace exactamente lo mismo que ya se ha explicado, solo que aquí se crea los espacios nada más.

```
            const calcularBtn = document.getElementById("calcularBtn");
            calcularBtn.addEventListener("click", function () {
                const inputValues = [];
                const inputs = document.querySelectorAll("#inputTable input");
                inputs.forEach(input => {
                    inputValues.push(parseInt(input.value));
                });

                const cuadro = new CuadroMagico(tamaño);
                cuadro.llenarMatriz(inputValues);
                mostrarMatrizIngresada(cuadro);
                verificarCuadroMagico(cuadro);
            });

            inputContainer.style.display = "block";
        } else {
            alert("Tamaño inválido. Debe ser un número mayor que cero.");
        }
    }
```

Se agrega el botón calcular y se llama este con el evento listener que al darle click nos recopila estos valores de entrada ingresados dentro de un array vació llamado inputValues. Se toman los valores de entrada que capturó el usuario,y el foreach lo que hace es convertirlos en enteros y obtenerlos para agregarlos al array inputValues.

Se crea una instancia para cuadro que tiene el argumento tamaño, este cuadro se le pasa el array inputValues. Esto llena la matriz del cuadro mágico con los valores ingresados por el usuario. En caso de no cumplir con las condiciones de los datos, manda una alerta.

```
112
113     function mostrarMatrizIngresada(cuadro) {
114         const matrizIngresada = document.getElementById("matrizIngresada");
115         matrizIngresada.innerHTML = ""; // Limpiar contenido anterior
116         const tabla = cuadro.generarTabla();
117         matrizIngresada.appendChild(tabla);
118     }
119
```

Esta función toma el objeto cuadro como argumento y con él muestra la matriz que se ingresó.

Se obtiene referencia al espacio designado para la matriz ingresada y dentro de este, con el método generarTabla se coloca esta misma la información de cuadro.

Para finalizar con este script se encuentra el método verificarCuadroMagico que toma de parámetro a cuadro, este mismo llama al método esCuadroMagico para comprobarlo y en caso de que sí lo sea, imprime la confirmación y la constante mágica que se obtuvo, esta constante se le llama al calcularConstanteMagica para obtenerlo, en caso contrario de esto, dice que no es un cuadro mágico.

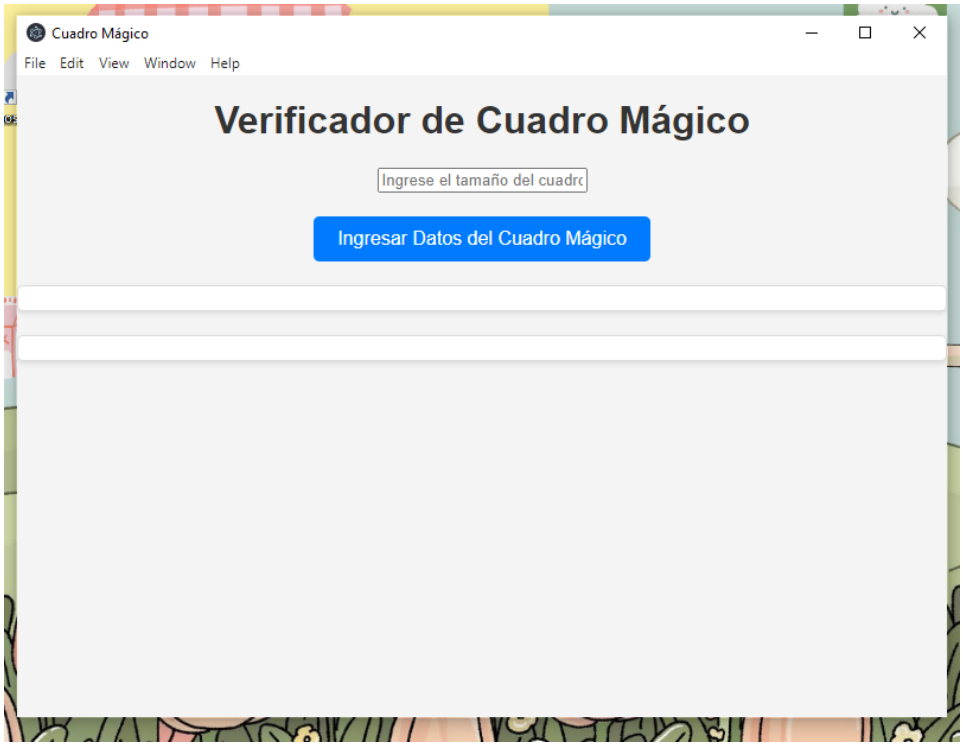
```
120     function verificarCuadroMagico(cuadro) {
121         if (cuadro.esCuadroMagico()) {
122             document.getElementById("resultado").textContent = `Es un cuadro mágico. La constante mágica es: ${cuadro.calcularConstanteMagica()}`;
123         } else {
124             document.getElementById("resultado").textContent = "No es un cuadro mágico.";
125         }
126     }
127
128 });
```

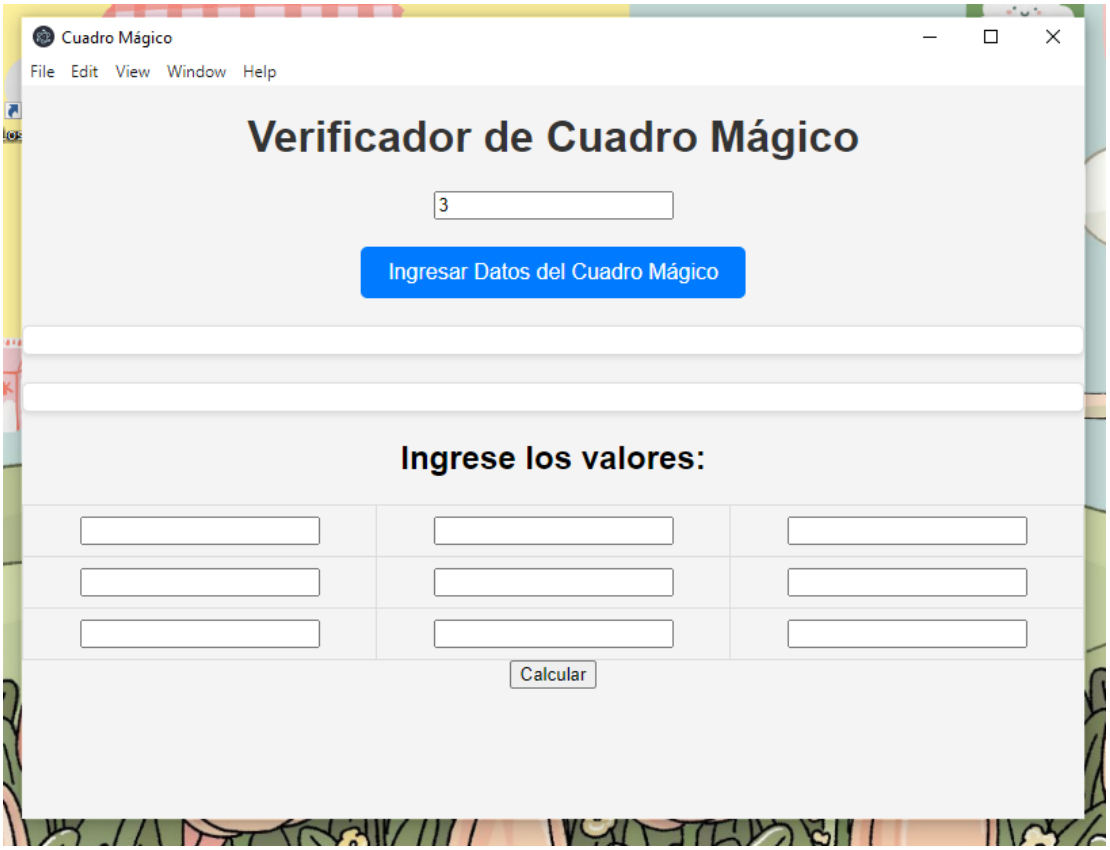
Para darle el estilo que quería para la generación de la tabla, le hice un css:

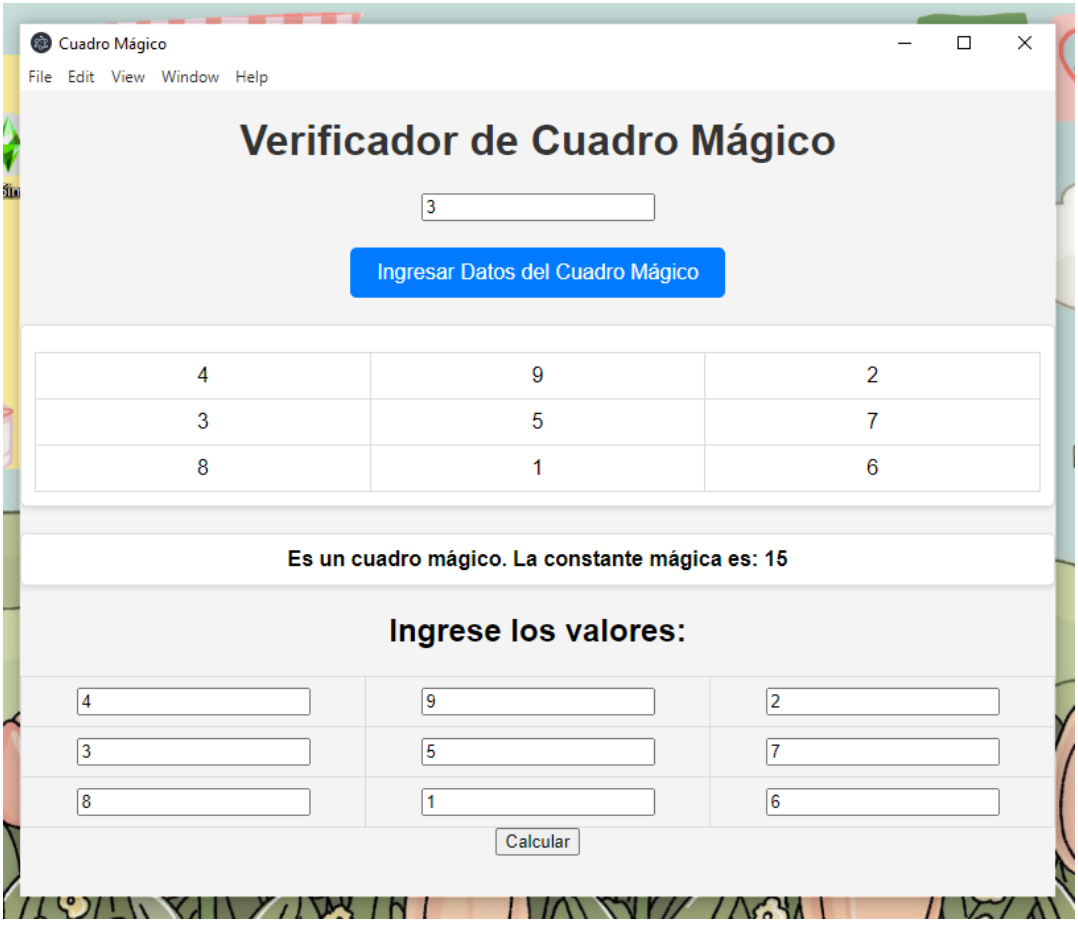
```css
body {
    font-family: Arial, sans-serif;
    text-align: center;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
}

h1 {
    color: #333;
    margin-top: 20px;
}

#ingresarDatosBtn {
    background-color: #007BFF;
    color: #fff;
    font-size: 16px;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    margin-top: 20px;
}

#matrizIngresada, #resultado {
    margin-top: 20px;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 5px;
    background-color: #fff;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}
```

```css
table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 10px;
}

table, th, td {
    border: 1px solid #ddd;
}

th, td {
    padding: 8px;
    text-align: center;
}

#resultado {
    font-weight: bold;
    margin-top: 20px;
}
```

Con la ayuda, otra vez del framework Electron, logré hacerlo una aplicación, el cual quedó así:




Cuadro Mágico
File Edit View Window Help
Verificador de Cuadro Mágico
3
Ingresar Datos del Cuadro Mágico
Ingrese los valores:
Calcular


Cuadro Mágico
File Edit View Window Help
Verificador de Cuadro Mágico
3
Ingresar Datos del Cuadro Mágico
4 9 2
3 5 7
8 1 6
Es un cuadro mágico. La constante mágica es: 15
Ingrese los valores:
4 9 2
3 5 7
8 1 6
Calcular