Universidad Tecnológica Metropolitana

Estructuras de Datos Aplicadas

ISC. Ruth Betsaida Martínez Domínguez, MGTI

Práctica 1. ABB

- Soberanis Acosta Jimena Monserrat

Desarrollo de Software Multiplataforma

Cuarto Cuatrimestre

4°B

Parcial III

Jueves, 16 de noviembre de 2023

Comencé con un HTML simple en donde hay dos entradas para escribir y 4 botones necesarios para la 'Biblioteca'.

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>Catálogo de Libros</title>
  <link rel="stylesheet" href="style.css">

</head>

<body>
  <h1>Catálogo de Libros</h1>

  <label for="isbn">ISBN:</label>
  <input type="number" id="isbn" placeholder="Ingrese el ISBN">
  <label for="titulo">Título:</label>
  <input type="text" id="titulo" placeholder="Ingrese el título">
  <button onclick="mostrarCatalogo()">Catalago</button>
  <button onclick="agregarLibro()">Agregar Libro</button>
  <button onclick="buscarLibro()">Buscar Libro</button>
  <button onclick="eliminarLibro()">Eliminar Libro</button>

  <div id="output"></div>
  <script type="module" src="js/app.js"></script>
</body>

</html>
```

Proseguí con el JavaScript y su lógica

js > JS libro.js > ...

```javascript
export class Libro {
    constructor(isbn, titulo){
        this.isbn = isbn;
        this.titulo = titulo;
        this.izquierda = null;
        this.derecha = null;
    }
}
```

Existe primeramente la clase ´libro' que representa un libro en la biblioteca y tiene las propiedades de 'ISBN', 'titulo', 'izquierda' y 'derecha'. Estas últimas dos se utilizan para la construcción de búsqueda binaria.

```javascript
import { Libro } from "./libro.js";

export class CatalogoLibro {
    constructor() {
        this.raiz = null;
    }
```

Existe el constructor que inicia la variable raíz del árbol como null cada que se crea una instancia de la clase CatalogoLibro.

```
8       agregarRecursivo(nodo, nuevoLibro) {
9           if (nuevoLibro.isbn < nodo.isbn) {
10              if (!nodo.izquierda) {
11                  nodo.izquierda = nuevoLibro;
12              } else {
13                  this.agregarRecursivo(this.izquierda, nuevoLibro);
14              }
15          } else {
16              if (!nodo.derecha) {
17                  nodo.derecha = nuevoLibro;
18              } else {
19                  this.agregarRecursivo(nodo.derecha, nuevoLibro);
20              }
21          }
22      }
```

Después está el método agregarRecursivo que se utiliza para agregar un nuevo libro de manera recursiva al árbol de búsqueda binario. Compara el ISBN del nuevo libro con el ISBN del nodo actual y decide que si el ISBN del libro es menor que el nodo actual debe colocarse a la izquierda y si es lo contrario (mayor que el nodo actual) se coloca a la derecha del nodo.

```
24      agregarLibro(isbn, titulo) {
25          const nuevoLibro = new Libro(isbn, titulo);
26
27          if (!this.raiz) {
28              this.raiz = nuevoLibro;
29          } else {
30              this.agregarRecursivo(this.raiz, nuevoLibro);
31          }
32      }
```

Junto al método explicado está su par llamado agregarLibro que llama al método agregarRecursivo. Este crea un nuevo libro con el ISBN y titulo que se proporciona. Si el árbol está vacío (que la raíz es null), ese nodo se vuelve en la raíz. Si no está vacío, entonces se llama al método recursivo mencionado antes, que ayuda a agregar cada nuevo libro en la posición adecuada en el árbol.

```
34      buscarRecursivo(nodo, isbn) {
35          if (!nodo) {
36              return null;
37          }
38
39          if (isbn === nodo.isbn) {
40              return nodo;
41          } else if (isbn < nodo.isbn) {
42              return this.buscarRecursivo(nodo.izquierda, isbn);
43          } else {
44              return this.buscarRecursivo(nodo.derecha, isbn);
45          }
46      }
```

Este método funciona como búsqueda recursiva en el árbol para poder encontrar un libro con el ISBN enviado. Comienza comparando desde la raíz en donde si no es un nodo devuelve un null, en caso contrario comienza comparando el ISBN con el ISBN del nodo, en ese caso devuelve el nodo que se encontró., sino ocurre este suceso, el ISBN se compara con el ISBN del nodo, en caso de que sea menor, se busca del lado izquierdo, si es mayor se busca del lado derecho.

```
48      buscarLibro(isbn) {
49          return this.buscarRecursivo(this.raiz, isbn);
50      }
51
```

Después, está el par de esté método, que realiza la búsqueda del libro llamando al método anterior.

```
51
52      encontrarSucesor(nodo) {
53          while (nodo.izquierda) {
54              nodo = nodo.izquierda;
55          }
56          return nodo;
57      }
```

Este método encuentra el sucesor de un nodo en el árbol. En este caso, el sucesor es el nodo más a la izquierda en el subárbol derecho del nodo dado.

```
59      inOrdenRecursivo(nodo, catalogoOrdenado) {
60          if (nodo) {
61              this.inOrdenRecursivo(nodo.izquierda, catalogoOrdenado);
62              catalogoOrdenado.push({ isbn: nodo.isbn, titulo: nodo.titulo });
63              this.inOrdenRecursivo(nodo.derecha, catalogoOrdenado);
64          }
65      }
```

En este método se utiliza el recorrido inorden del árbol. Aquí toma el nodo dado que recorre el árbol de manera ascendente visitando primero al subárbol izquierdo llamando recursivamente a este método, luego recorre el nodo actual para de último recorrer el subárbol derecho.

```
67      obtenerCatalogoOrdenado() {
68          const catalogoOrdenado = [];
69          this.inOrdenRecursivo(this.raiz, catalogoOrdenado);
70          return catalogoOrdenado;
71      }
```

Existe esté método que devuelve un array con todos los libros en el árbol, los ordena de acuerdo según su posición en el árbol.

```
73      eliminarRecursivo(nodo, isbn) {
74          if (!nodo) {
75              return null;
76          }
77
78          if (isbn === nodo.isbn) {
79
80              if (!nodo.izquierda) {
81                  return nodo.derecha;
82              } else if (!nodo.derecha) {
83                  return nodo.izquierda;
84              }
85
86              const sucesor = this.encontrarSucesor(nodo.derecha);
87
88              nodo.isbn = sucesor.isbn;
89              nodo.titulo = sucesor.titulo;
90              nodo.derecha = this.eliminarRecursivo(nodo.derecha, sucesor.isbn);
91          }else if(isbn < nodo.isbn){
92              nodo.izquierda = this.eliminarRecursivo(nodo.izquierda, isbn);
93          }else{
94              nodo.derecha = this.eliminarRecursivo(nodo.derecha, isbn);
95          }
96
97          return nodo;
98      }
```

Está el método de eliminación recursivo de un libro a través del ISBN dado, si el libro a eliminar tiene un solo hijo, ese hijo se convierte en el nuevo hijo del nodo padre. Si tiene dos hijos, se encuentra el sucesor en el subárbol derecho y se realiza una copia de sus datos al nodo a eliminar.

```
    eliminarLibro(isbn){
        this.raiz = this.eliminarRecursivo(this.raiz, isbn);
    }
}
```

Por último, está el par de eliminación, que llama al método eliminarRecursivo y actualiza la raíz del árbol con el nuevo árbol resultante después de la eliminación.

Para el archivo app.js que es la parte interactiva con el usuario, tiene las siguientes funciones:

```
import { CatalogoLibro } from "./catalogoLibro.js";

const catalogo = new CatalogoLibro();
const outputDiv = document.getElementById("output");

document.querySelector("button:nth-of-type(1)").addEventListener("click", mostrarCatalogo);
document.querySelector("button:nth-of-type(2)").addEventListener("click", agregarLibro);
document.querySelector("button:nth-of-type(3)").addEventListener("click", buscarLibro);
document.querySelector("button:nth-of-type(4)").addEventListener("click", eliminarLibro);

```

En esta sección es la que llama a cada uno los botones del documento.

```
function mostrarCatalogo(){
    const catalogoCompleto = catalogo.obtenerCatalogoOrdenado();
    outputDiv.innerHTML = "<h2>Catalogo</2>";

    if (catalogoCompleto.length === 0) {
        outputDiv.innerHTML += "<p> El catálogo está vacío</p>";
    } else {
        catalogoCompleto.forEach((libro) => {
            outputDiv.innerHTML += `<p>${libro.isbn}: ${libro.titulo}</p>`;
        });
    }
}
```

Función mostrarCatalogo. Esta función se ejecuta al hacer clic en el botón correspondiente en la interfaz en donde se obtiene el catálogo completo ordenado utilizando el método obtenerCatalogoOrdenado de la instancia catálogo, y actualiza el contenido del elemento HTML con id "outputDiv" para mostrar el catálogo ordenado.

```
function agregarLibro(){
    const isbn = document.getElementById("isbn").value;
    const titulo = document.getElementById("titulo").value;

    if(catalogo.buscarLibro(isbn)){
        mensaje("El libro con este IBN ya existe en el catálogo");
    }else{
        catalogo.agregarLibro(isbn, titulo);
        actualizarOutput();
    }
}
```

Función agregarLibro. Esta función se ejecuta al hacer clic en el botón correspondiente en la interfaz, se obtiene el ISBN y el título del nuevo libro desde los elementos HTML, utiliza el método buscarLibro para verificar si el libro ya existe en el catálogo. Si el libro no existe, agrega el nuevo libro al catálogo utilizando agregarLibro y luego actualiza la salida.

```
36  function buscarLibro(){
37      const isbn = document.getElementById("isbn").value;
38      const libro = catalogo.buscarLibro(isbn);
39
40      if(libro){
41          mensaje(`Libro encontrado: ${libro.titulo}`);
42      }else{
43          mensaje(`Libro no encontrado`);
44      }
45  }
```

Función buscarLibro: Esta función se ejecuta al hacer clic en el botón correspondiente en la interfaz, obtiene el ISBN del libro a buscar desde el elemento HTML. Utiliza el método buscarLibro para buscar el libro en el catálogo y muestra un mensaje indicando si el libro fue encontrado o no.

```
47  function eliminarLibro(){
48      const isbn = document.getElementById("isbn").value;
49
50      catalogo.eliminarLibro(isbn);
51      actualizarOutput();
52  }
```

Función eliminarLibro. Esta función se ejecuta al hacer clic en el botón correspondiente en la interfaz en donde se obtiene el ISBN del libro a eliminar desde el elemento HTML. Utiliza el método eliminarLibro para eliminar el libro del catálogo y luego, actualiza la salida para reflejar los cambios.

```
54  function actualizarOutput(){
55      const catalogoOrdenado = catalogo.obtenerCatalogoOrdenado();
56      outputDiv.innerHTML = "<h2>Catálogo</h2>";
57
58      if(catalogoOrdenado.length === 0){
59          outputDiv.innerHTML += "<p> El catálogo está vacío</p>";
60      }else{
61          catalogoOrdenado.forEach((libro) =>{
62              outputDiv.innerHTML += `<p>${libro.isbn}: ${libro.titulo}</p>`;
63          });
64      }
65  }
```

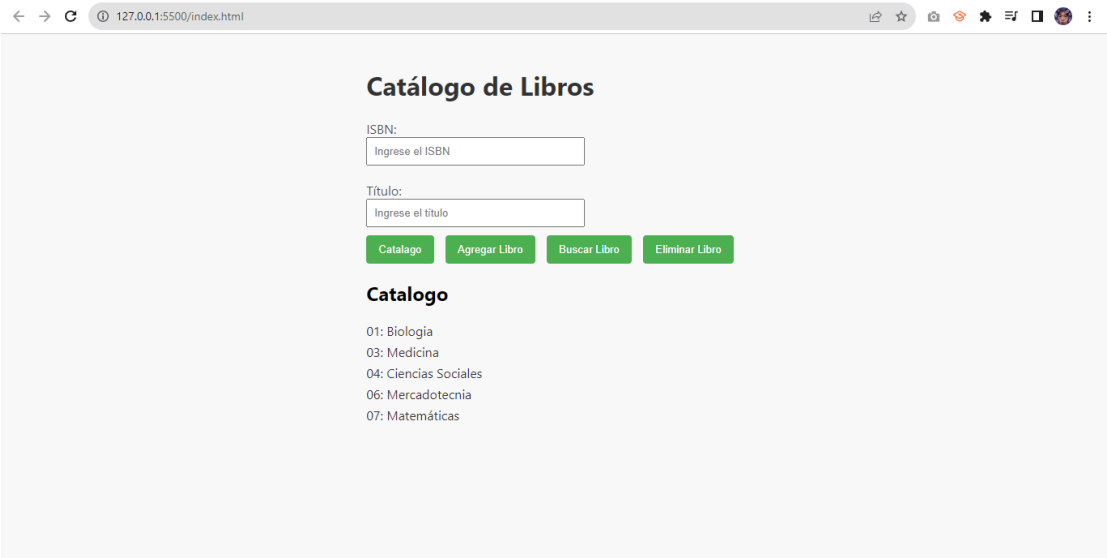
Función actualizarOutput. Esta función actualiza la salida en el HTML con el catálogo ordenado. Obtiene el catálogo ordenado utilizando el método obtenerCatalogoOrdenado de la instancia catálogo, y limpia el contenido actual del elemento HTML y luego agrega los elementos del catálogo ordenado.

```
66
67  function mensaje(mensaje){
68      outputDiv.innerHTML = `<p>${mensaje}</p>`;
69  }
```

Función mensaje. Esta función se utiliza para mostrar mensajes en la interfaz que recibe un mensaje como parámetro y lo muestra en el elemento HTML con id "outputDiv".

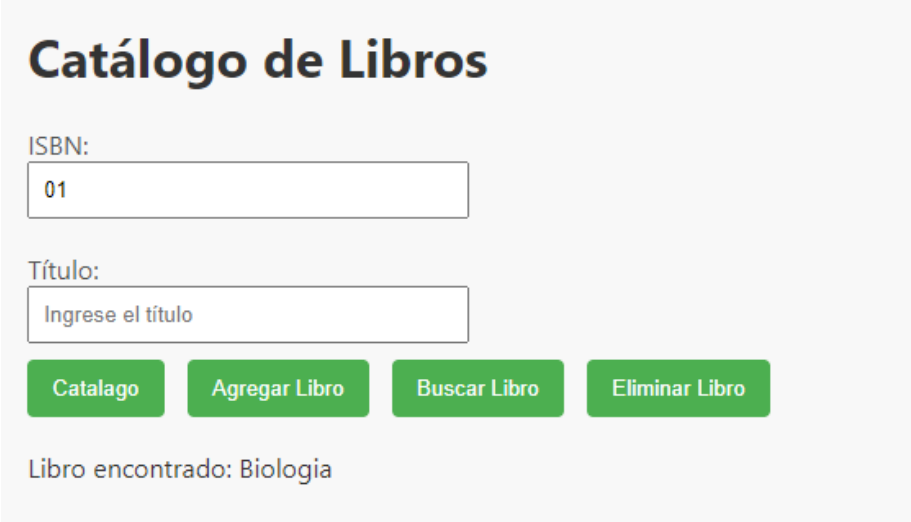
Después de agregarle un estilo con CSS, así aparece en su aplicación:

Agregué ciertos libros para la demostración.



Puedo agregar libros:

Puedo buscar algún libro a través de su ISBN:



Y puedo eliminar un libro con su ISBN: