



INSTITUTO TECNOLÓGICO DEL VALLE DE OAXACA

PROGRAMACION II



Alumno:

Javier Ezequiel García Florean

DOCENTE: Ambrosio Cardoso Jiménez

TEMA: 3.1 Ejercicios sobre operaciones de Crear, Leer, Actualizar y Borrar datos (100%)

SEMESTRE: 4°

GRUPO: "A"

CARRERA: Ingeniería en Tics

FECHA DE ENTREGA: 08 de mayo de 2023

Ejercicio 1:

A continuacion se muestran los codigos que se utilizaron para la realizacion de las interfaces de una pagina.



Producto.html



DataManager.js



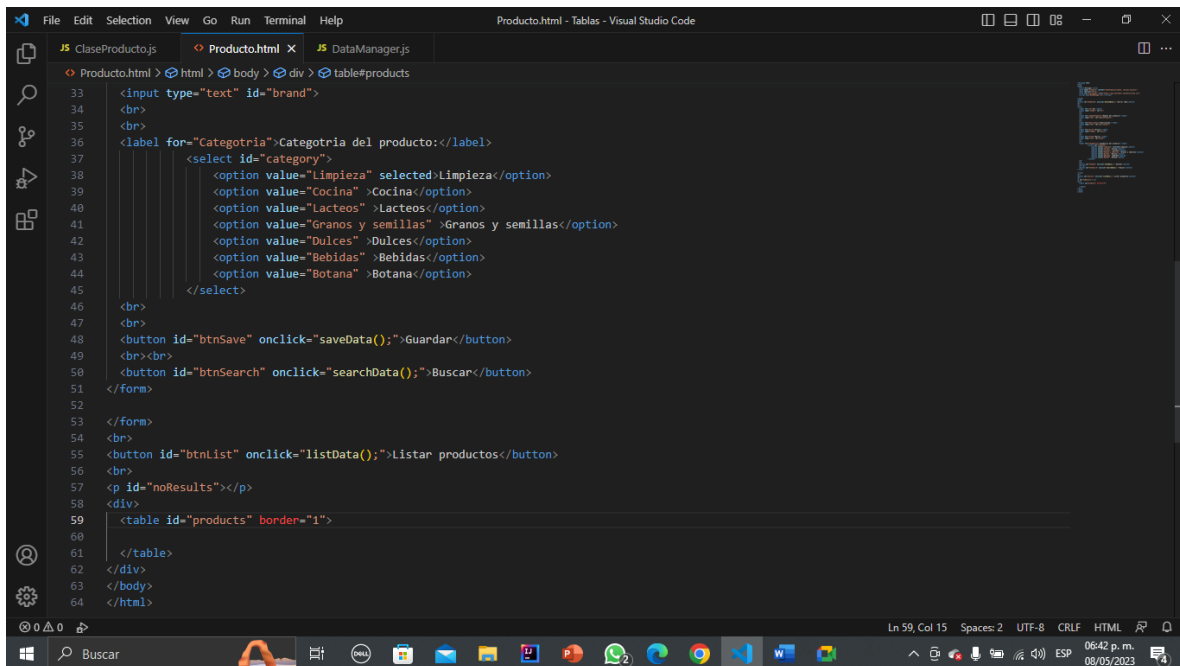
ClaseProducto.js

Como podemos observar en esta primera parte de código se inserto en la parte de arriba un botón de borrar todo el contenido, además de eso se insertaron las celdas para ingresar los datos correspondientes del producto los cuales son:

- Id del producto
- Nombre del producto
- Descripción
- Precio
- Marca
- Categoría: En este caso en las categorías se realizó una lista desplegable con las diferentes categorías como son: Limpieza, Cocina, Lácteos, Granos y semillas, Dulces, Bebidas y por último Botana

```
1 <!doctype html>
2 <html>
3 <head>
4 <title>Storage</title>
5 <meta name="viewport" content="width=device-width, initial-scale=1">
6 <meta charset="utf-8">
7 <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
8 <script src="DataManager.js"></script>
9
10 </head>
11 <body>
12 <button id="btnDelete" onclick="deleteData();">Borrar todo</button>
13 <br>
14 <br>
15 <form>
16 <label for="id">ID:</label>
17 <input type="number" id="id">
18 <br>
19 <br>
20 <label for="nombreProducto">Nombre del producto:</label>
21 <input type="text" id="nombreProducto">
22 <br>
23 <br>
24 <label for="description">Descripción:</label>
25 <input type="text" id="description">
26 <br>
27 <br>
28 <label for="price">Precio:</label>
29 <input type="number" id="price">
30 <br>
31 <br>
32 <label for="brand">Marca:</label>
33 <input type="text" id="brand">
34 </form>
35 </body>
36 </html>
```

En la siguiente imagen podemos visualizar lo antes mencionado de la parte del listado de categoría, también se ingresó un botón de guardar los datos que se introdujeron, contiene un botón de buscar, el cual nos ayudara a buscar un producto dependiendo de que opción deseamos ya sea por id del producto, nombre, precio etc. También tiene un botón de Listar productos el cual al darle clic en listara todos los productos que se hayan introducido.



```
33 <input type="text" id="brand">
34 <br>
35 <br>
36 <label for="Categoria">Categoría del producto:</label>
37 <select id="category">
38 <option value="Limpieza" selected>Limpieza</option>
39 <option value="Cocina">Cocina</option>
40 <option value="Lacteos">Lacteos</option>
41 <option value="Granos y semillas">Granos y semillas</option>
42 <option value="Dulces">Dulces</option>
43 <option value="Bebidas">Bebidas</option>
44 <option value="Botana">Botana</option>
45 </select>
46 <br>
47 <br>
48 <button id="btnSave" onclick="saveData();">Guardar</button>
49 <br><br>
50 <button id="btnSearch" onclick="searchData();">Buscar</button>
51 </form>
52
53 </form>
54 <br>
55 <button id="btnList" onclick="listData();">Listar productos</button>
56 <br>
57 <p id="noResults"></p>
58 <div>
59 <table id="products" border="1">
60
61 </table>
62 </div>
63 </body>
64 </html>
```

En esta segundo codigo podemos observar la creacion de la base de datos con la palabra reservada `localStorage`, realizando un compartido entre todas las pestañas y ventanas del mismo origen. Los datos no expiran. Persisten a los reinicios de navegador y hasta del sistema operativo.

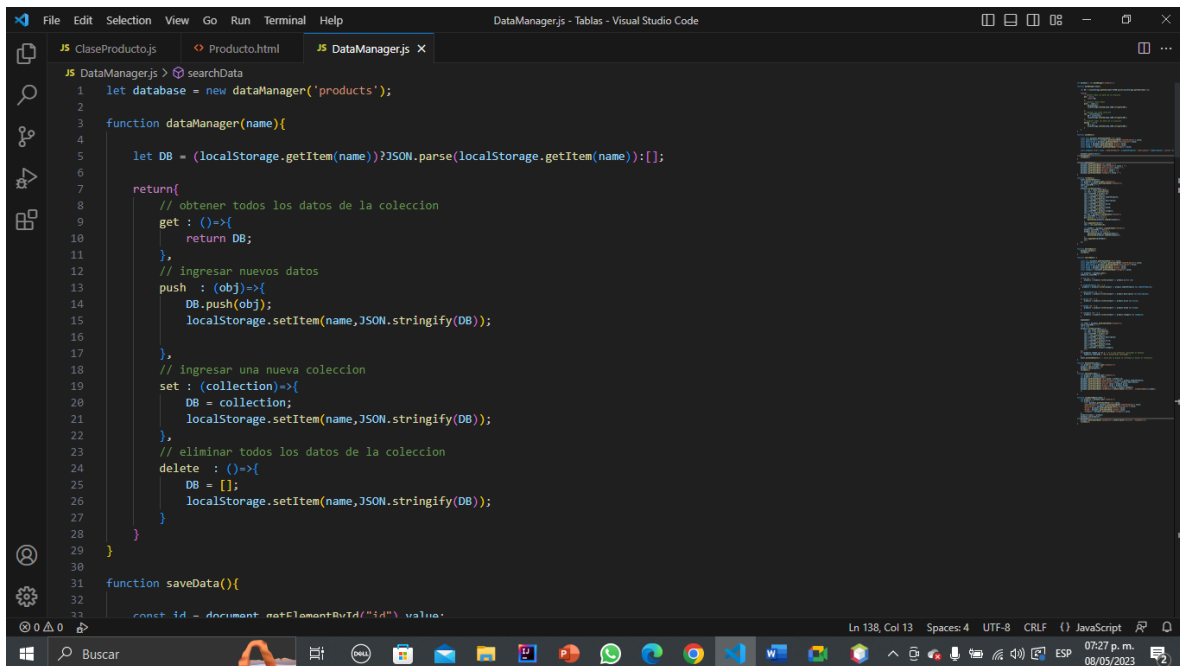
Tambien se realizo con la palabra reservada `sessionStorage`, las propiedades y métodos son los mismos, pero es mucho más limitado:

`sessionStorage` solo existe dentro de la pestaña actual del navegador.

Otra pestaña con la misma página tendrá un almacenaje distinto.

Pero se comparte entre iframes en la pestaña (asumiendo que tengan el mismo origen).

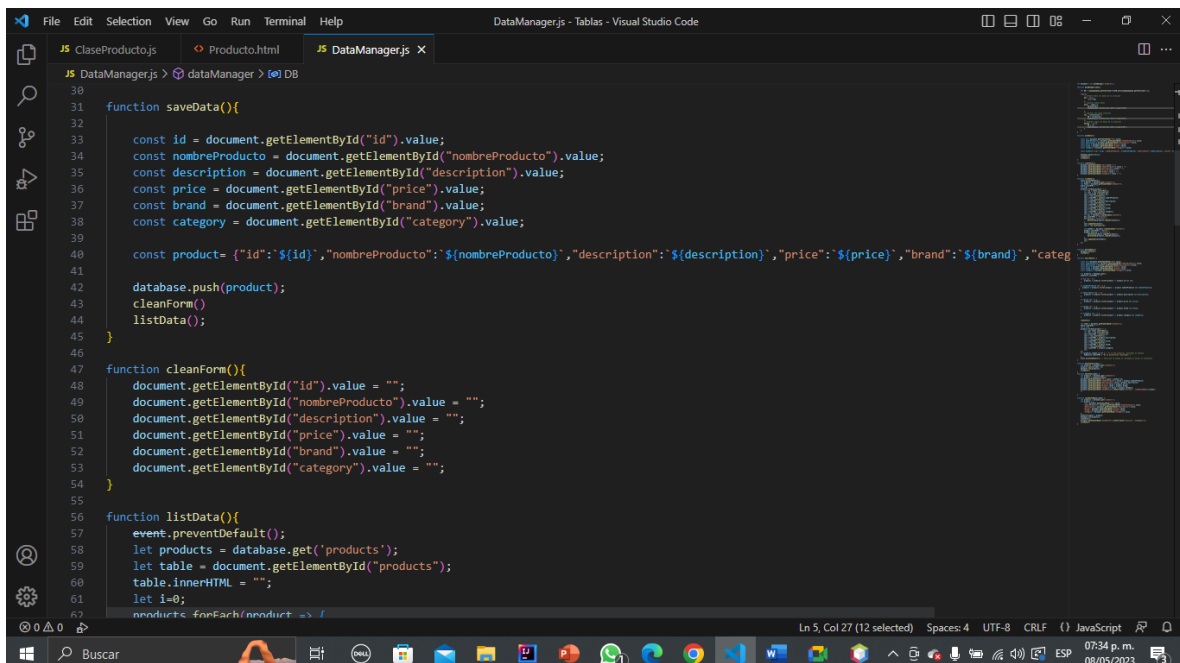
Los datos sobreviven un refresco de página, pero no cerrar/abrir la pestaña.



```
1 let database = new DataManager('products');
2
3 function DataManager(name){
4
5     let DB = (localStorage.getItem(name)) ? JSON.parse(localStorage.getItem(name)) : [];
6
7     return{
8         // obtener todos los datos de la coleccion
9         get : ()=>{
10             return DB;
11         },
12         // ingresar nuevos datos
13         push : (obj)>=>{
14             DB.push(obj);
15             localStorage.setItem(name,JSON.stringify(DB));
16         },
17         // ingresar una nueva coleccion
18         set : (collection)>=>{
19             DB = collection;
20             localStorage.setItem(name,JSON.stringify(DB));
21         },
22         // eliminar todos los datos de la coleccion
23         delete : ()=>{
24             DB = [];
25             localStorage.setItem(name,JSON.stringify(DB));
26         }
27     }
28 }
29
30
31 function saveData(){
32     const id = document.getElementById("id").value;
```

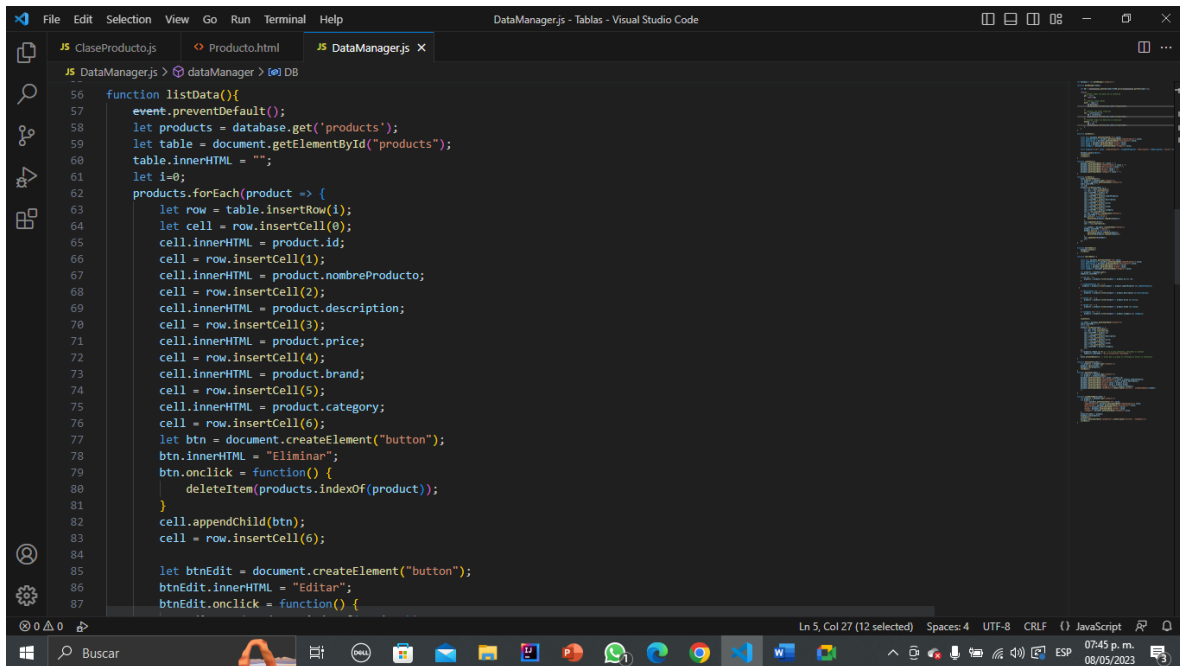
A continuación, podemos ver la función `saveData`, su funcionamiento es guardar los datos que introduce el usuario por medio de las interfaces de HTML guardándolas en la base de datos antes creada.

También podemos observar la función `clearForm`, su función es limpiar el formulario ya que se haya rellenado y guardado en la base esto facilitando el control de ingresar productos.



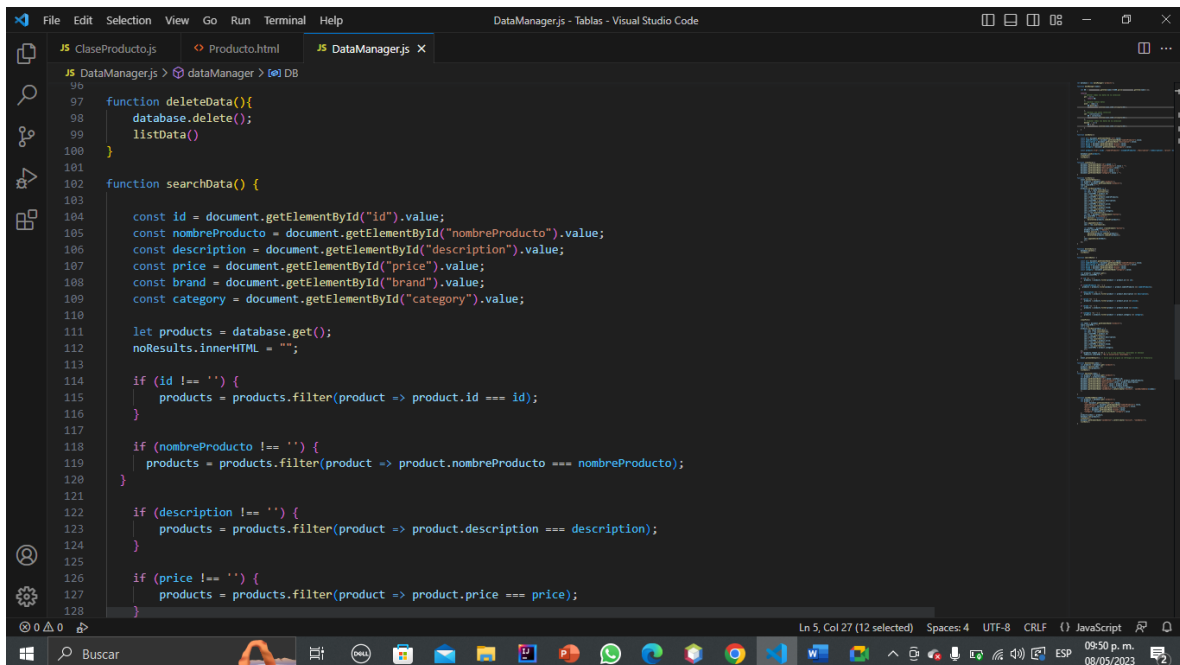
```
30
31 function saveData(){
32     const id = document.getElementById("id").value;
33     const nombreProducto = document.getElementById("nombreProducto").value;
34     const description = document.getElementById("description").value;
35     const price = document.getElementById("price").value;
36     const brand = document.getElementById("brand").value;
37     const category = document.getElementById("category").value;
38
39     const product= { "id":`${id}`, "nombreProducto":`${nombreProducto}`, "description":`${description}`, "price":`${price}`, "brand":`${brand}`, "category":`${category}` };
40
41     database.push(product);
42     clearForm();
43     listData();
44 }
45
46
47 function clearForm(){
48     document.getElementById("id").value = "";
49     document.getElementById("nombreProducto").value = "";
50     document.getElementById("description").value = "";
51     document.getElementById("price").value = "";
52     document.getElementById("brand").value = "";
53     document.getElementById("category").value = "";
54 }
55
56 function listData(){
57     event.preventDefault();
58     let products = database.get('products');
59     let table = document.getElementById("products");
60     table.innerHTML = "";
61     let i=0;
62     products.forEach(product => {
```

Como podemos observar en este código se implementó una función `listaData`, el cual realiza el listado de todos los productos que se ingresaron en el formulario anterior para poder mostrarse los a los usuarios por medio de la interfaz, además se implementaron dos botones el cual consta de eliminar de la lista algún producto o editar su contenido.

A screenshot of the Visual Studio Code editor with the file 'DataManager.js' open. The code defines a function 'listData()' that interacts with a database to fetch products and display them in a table. It includes logic for creating 'Eliminar' (Delete) and 'Editar' (Edit) buttons for each product row. The status bar at the bottom shows 'Ln 5, Col 27 (12 selected)' and the date '08/05/2023'.

```
56 function listData(){
57   event.preventDefault();
58   let products = database.get('products');
59   let table = document.getElementById("products");
60   table.innerHTML = "";
61   let i=0;
62   products.forEach(product => {
63     let row = table.insertRow(i);
64     let cell = row.insertCell(0);
65     cell.innerHTML = product.id;
66     cell = row.insertCell(1);
67     cell.innerHTML = product.nombreProducto;
68     cell = row.insertCell(2);
69     cell.innerHTML = product.description;
70     cell = row.insertCell(3);
71     cell.innerHTML = product.price;
72     cell = row.insertCell(4);
73     cell.innerHTML = product.brand;
74     cell = row.insertCell(5);
75     cell.innerHTML = product.category;
76     cell = row.insertCell(6);
77     let btn = document.createElement("button");
78     btn.innerHTML = "Eliminar";
79     btn.onclick = function() {
80       deleteItem(products.indexOf(product));
81     }
82     cell.appendChild(btn);
83     cell = row.insertCell(6);
84
85     let btnEdit = document.createElement("button");
86     btnEdit.innerHTML = "Editar";
87     btnEdit.onclick = function() {
```

Así mismo tenemos una función `deleteData`, su función es eliminar todos los datos de la lista que se creó, dejando vacío la lista, también podemos observar una función `searchData` su función es buscar es buscar los productos en la base de datos por medio de las variables declaradas.

A screenshot of the Visual Studio Code editor showing the 'deleteData()' and 'searchData()' functions in 'DataManager.js'. 'deleteData()' calls 'database.delete()' and 'listData()'. 'searchData()' filters the 'products' array based on user input for id, nombreProducto, description, price, and category. The status bar at the bottom shows 'Ln 5, Col 27 (12 selected)' and the date '08/05/2023'.

```
96
97 function deleteData(){
98   database.delete();
99   listData()
100 }
101
102 function searchData() {
103
104   const id = document.getElementById("id").value;
105   const nombreProducto = document.getElementById("nombreProducto").value;
106   const description = document.getElementById("description").value;
107   const price = document.getElementById("price").value;
108   const brand = document.getElementById("brand").value;
109   const category = document.getElementById("category").value;
110
111   let products = database.get();
112   noResults.innerHTML = "";
113
114   if (id !== '') {
115     products = products.filter(product => product.id === id);
116   }
117
118   if (nombreProducto !== '') {
119     products = products.filter(product => product.nombreProducto === nombreProducto);
120   }
121
122   if (description !== '') {
123     products = products.filter(product => product.description === description);
124   }
125
126   if (price !== '') {
127     products = products.filter(product => product.price === price);
128   }
129 }
```

Esta función de clearForm permite limpiar el formulario donde se introducen los datos de los productos facilitando el poder seguir introduciendo mas productos sin la necesidad de estar borrando dato por dato en el formulario.

The image shows a Visual Studio Code editor window with a dark theme. The title bar at the top reads "DataManager.js - Tablas - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The sidebar on the left contains icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The main editor area displays a JavaScript file named "DataManager.js" with the following code:

```
137  
138     cleanForm()  
139  
140     let table = document.getElementById("products");  
141     table.innerHTML = "";  
142     let i = 0;  
143     products.forEach(product => {  
144         let row= table.insertRow(i);  
145         let cell = row.insertCell(0);  
146         cell.innerHTML = product.id;  
147         cell = row.insertCell(1);  
148         cell.innerHTML = product.description;  
149         cell = row.insertCell(2);  
150         cell.innerHTML = product.price;  
151         cell = row.insertCell(3);  
152         cell.innerHTML = product.brand;  
153         cell = row.insertCell(4);  
154         cell.innerHTML = product.category;  
155         i++;  
156     });  
157     if (products.length === 0) { // Si no hay productos, mostramos el mensaje  
158         noResults.innerHTML = "No se encontraron resultados.";   
159     }  
160     event.preventDefault(); // evita que la página se refresque al enviar el formulario  
161 }  
162  
163 function deleteItem(index) {  
164     let products = database.get('products');  
165     products.splice(index, 1);  
166     database.set(products);  
167     listData();  
168 }  
169 function editItem(index) {
```

The status bar at the bottom indicates "Ln 5, Col 27 (12 selected)", "Spaces: 4", "UTF-8", "CRLF", and "JavaScript". The system tray at the bottom right shows the date and time as "10:23 p.m. 08/05/2023".

Por ultimo tenemos la función `deleteItem`, el cual realiza la eliminación de datos de la lista que se crea al estar introduciendo datos en la base de datos, también podemos observar que tenemos la función `editItem` el cual nos permite editar el contenido de los productos en la lista, y para finalizar tenemos la función `saveEditedData` el cual guarda los cambios que se realizan a la lista de datos dependiendo que correcciones se realizan.

The screenshot displays the Visual Studio Code editor with a JavaScript file named 'DataManager.js' open. The code is as follows:

```

function deleteItem(index) {
    let products = database.get('products');
    products.splice(index, 1);
    database.set(products);
    listData();
}

function editItem(index) {
    let products = database.get('products');
    let product = products[index];
    document.getElementById("id").value = product.id;
    document.getElementById("nombreProducto").value = product.nombreProducto;
    document.getElementById("description").value = product.description;
    document.getElementById("price").value = product.price;
    document.getElementById("brand").value = product.brand;
    document.getElementById("category").value = product.category;
    document.getElementById("saveButton").setAttribute("onclick", `saveEditedData(${index})`);
};

function saveEditedData(index) {
    let products = database.get('products');
    let product = {
        "id": document.getElementById("id").value,
        "nombreProducto": document.getElementById("nombreProducto").value,
        "description": document.getElementById("description").value,
        "price": document.getElementById("price").value,
        "brand": document.getElementById("brand").value,
        "category": document.getElementById("category").value
    };
    products[index] = product;
    database.set(products);
}

```

The interface includes a sidebar on the left with icons for Explorer, Search, Source Control, and Run and Debug. The bottom status bar shows 'Ln 5, Col 27 (12 selected)', 'Spaces: 4', 'UTF-8', 'CRLF', and 'JavaScript'.

Como podemos observar se muestran los espacios del formulario para introducir los datos correspondientes del producto, también podemos visualizar los botones de guardar, buscar, borrar todo y listar productos, así mismo se puede observar el listado de productos que se han introducido.

Storage x +

← → ↻ Archivo | C:/Users/jgri/OneDrive/Documentos/progra_II/Tablas/Producto.html

Borrar todo

ID:

Nombre del producto:

Descripción:

Precio:

Marca:

Categoría del producto:

Guardar

Buscar

Listar productos

1	cheetos	botana	15	sabritas	Botana	Editar	Eliminar
2	Fabuloso	Aromatizante	25	Fabuloso	Limpieza	Editar	Eliminar
3	Pake taxo	botana	17	sabritas	Botana	Editar	Eliminar
3	NutriLeche	Leche de uso humano	20	NutriLeche	Lacteos	Editar	Eliminar
4	queso	Comestible	28	Patito	Lacteos	Editar	Eliminar

Windows Buscar

10:31 p. m.
08/05/2023