

UNIVERSIDAD DON BOSCO
FACULTAD DE CIENCIAS BÁSICAS



Trabajo de investigación

ASIGNATURA: Diseño y Programación de Software Multiplataforma

GRUPO: G01T

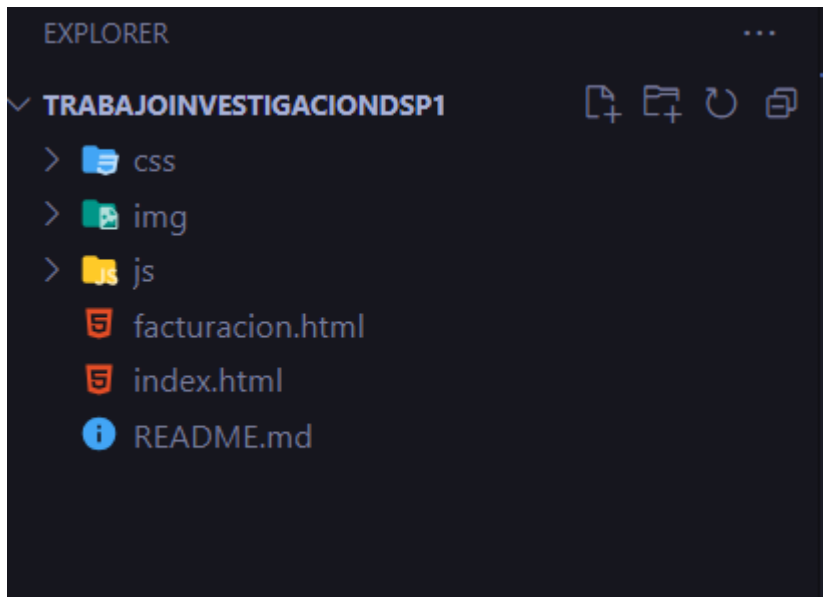
DOCENTE: ING. Alexander Alberto Sigüenza Campos

INTEGRANTES:

NOMBRES	CARNET
Menéndez Zepeda, Josué Erasmo	MZ142582
Vásquez Villalta, Luis Eduardo	VV121782

FECHA DE ENTREGA: 03/03/2024.

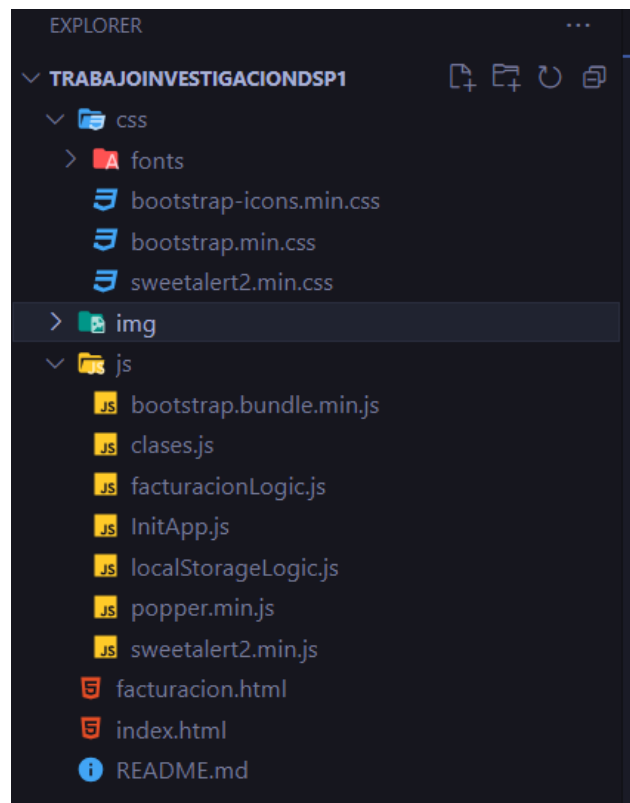
La estructura de carpeta del proyecto es la siguiente:



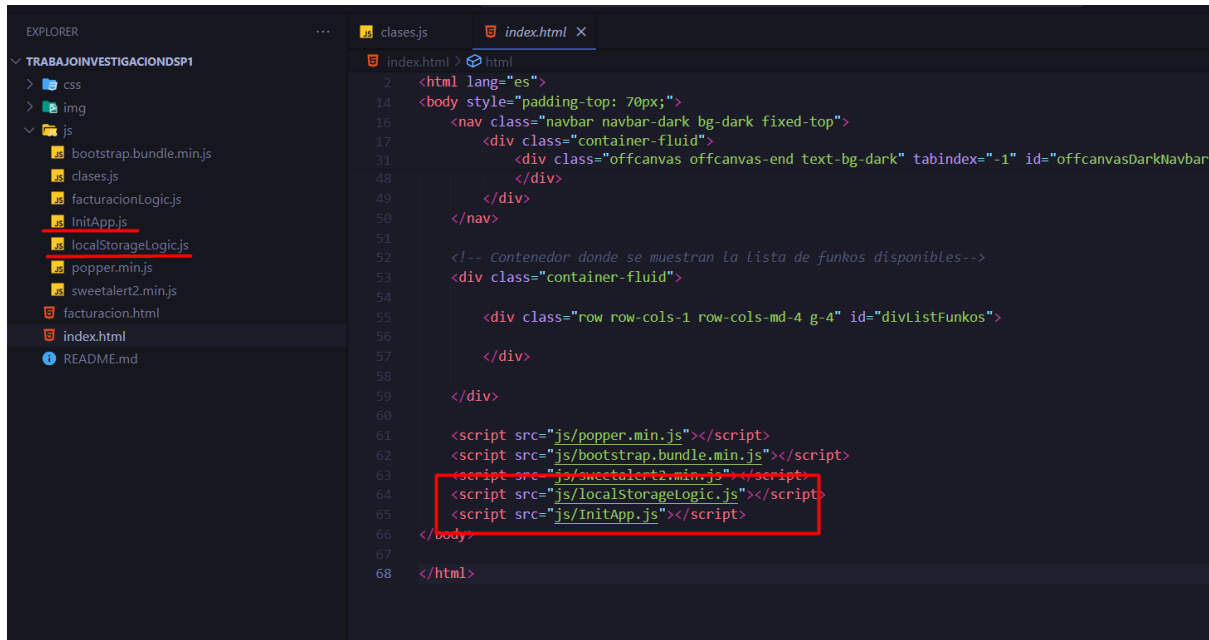
css: En esta carpeta esta conteniendo los distintos estilos (librerías) que se ocupan para darle mejor presentación al siteo web, siendo todas las versiones minimizadas para reducir la carga de grandes tamaños de archivos.

img: En ella se encuentra las distintas imágenes en formato webp, que se cargan dinámicamente dependiente de las acciones que se le apliquen.

js: Aquí se encuentran tanto los archivos de lógica que se ocupan de las distintas librerías que se importaron, así como los archivos de lógica de cada una de las pantallas (archivos html) creadas.

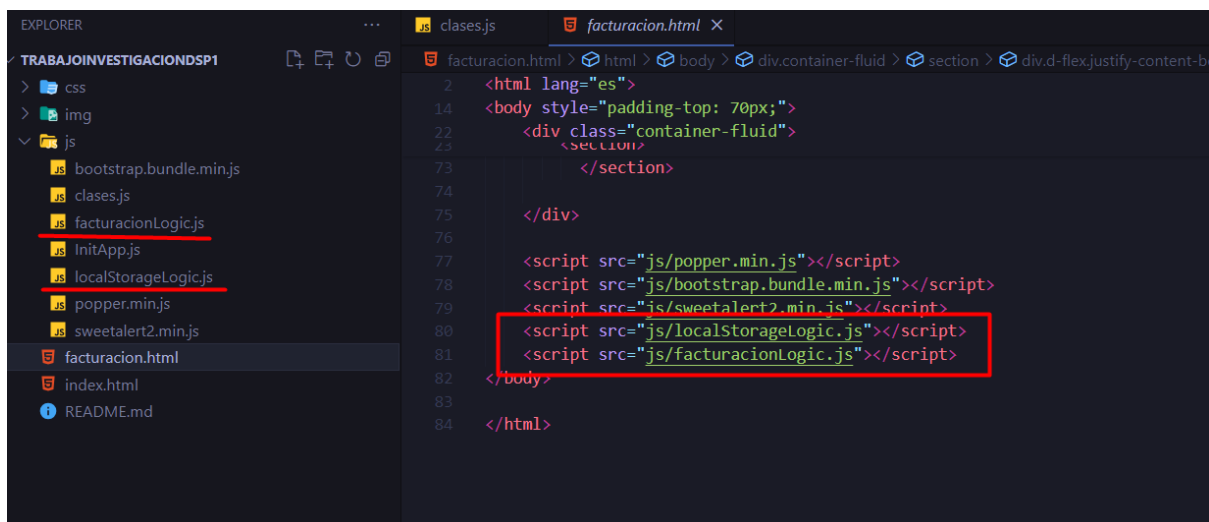


La aplicación trabaja por medio del localStorage que proporcionan los navegadores para crear dinámicamente los elementos en el html con la ayuda del javascript, se explicara la finalidad de la estructuración que se le dio al código:



```
2 <html lang="es">
14 <body style="padding-top: 70px;">
16 <nav class="navbar navbar-dark bg-dark fixed-top">
17 <div class="container-fluid">
31 <div class="offcanvas offcanvas-end text-bg-dark tabindex="-1" id="offcanvasDarkNavbar">
48 </div>
49 </div>
50 </nav>
51
52 <!-- Contenedor donde se muestran la lista de funks disponibles-->
53 <div class="container-fluid">
54
55 <div class="row row-cols-1 row-cols-md-4 g-4" id="divListFunks">
56
57 </div>
58
59 </div>
60
61 <script src="js/popper.min.js"></script>
62 <script src="js/bootstrap.bundle.min.js"></script>
63 <script src="js/sweetalert2.min.js"></script>
64 <script src="js/localStorageLogic.js"></script>
65 <script src="js/InitApp.js"></script>
66 </body>
67
68 </html>
```

Se separo el código en mini módulos en los cuales se tendrían la lógica específica para que se importaran en el html específico con su finalidad.



```
2 <html lang="es">
14 <body style="padding-top: 70px;">
22 <div class="container-fluid">
23 <section>
73 </section>
74
75 </div>
76
77 <script src="js/popper.min.js"></script>
78 <script src="js/bootstrap.bundle.min.js"></script>
79 <script src="js/sweetalert2.min.js"></script>
80 <script src="js/localStorageLogic.js"></script>
81 <script src="js/facturacionLogic.js"></script>
82 </body>
83
84 </html>
```

Se construyo de igual forma un archivo que contendría cada uno de las clases que se ocuparían que sería más genérico en toda la aplicación, con el objetivo de minimizar el código de cada uno de los objetos ya que los elementos html se crean dinámicamente con javascript:

```
EXPLORER
TRABAJOINVESTIGACIONDSP1
  > css
  > img
  > js
    bootstrap.bundle.min.js
    clases.js
    facturacionLogic.js
    InitApp.js
    localStorageLogic.js
    popper.min.js
    sweetalert2.min.js
    facturacion.html
    index.html
    README.md

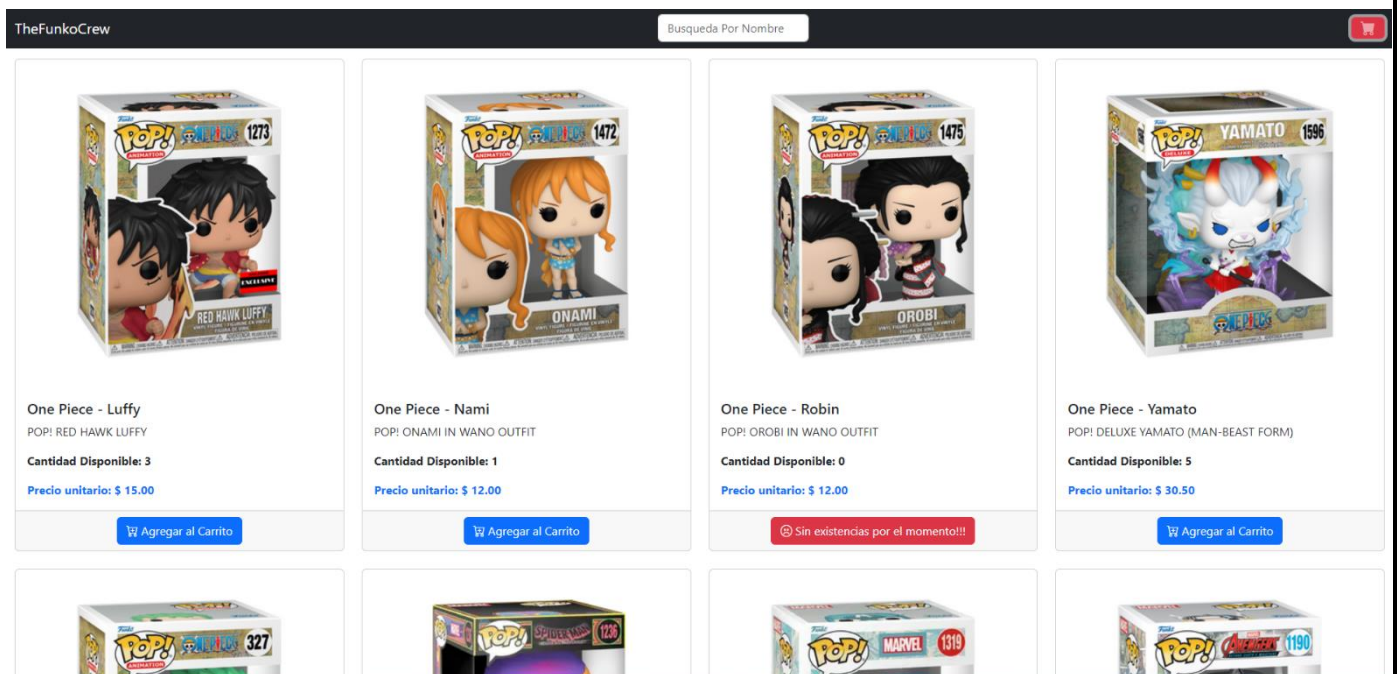
clases.js
//Clase del Tipo Funko para el manejo de los objetos y su renderizado en el HTML
class Funko {
  constructor(id, nombre, descripcion, precio, cantidadDisponible, nombreImg){
    this.id = id;
    this.nombre = nombre;
    this.descripcion = descripcion;
    this.precio = precio;
    this.cantidadDisponible = cantidadDisponible;
    this.nombreImg = nombreImg;
  }

  get urlImagen(){
    return `img/${this.nombreImg}.webp`;
  }

  get precioFormateado(){
    return `$ ${Number.parseFloat(this.precio).toFixed(2)}`;
  }

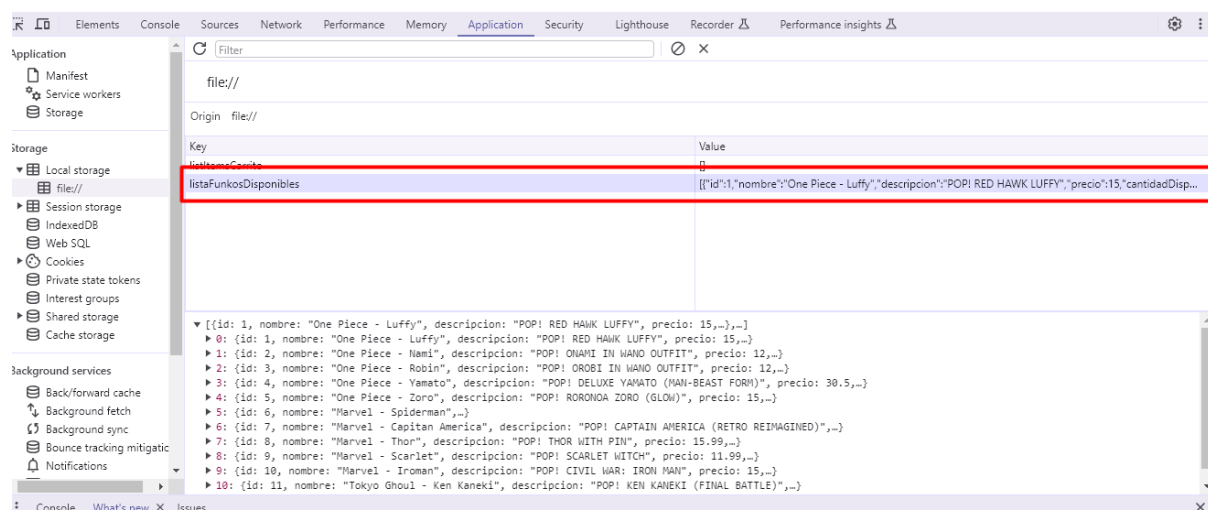
  //funcion que devuelve el html ya con la transpolacion de los valores del objeto
  get getDivContainerAllInformation() {
    return `<div class="col">
      <div class="card h-100" id-funko="${this.id}">
        
        <div class="card-body">
          <h5 class="card-title">${this.nombre}</h5>
          <p class="card-text">${this.descripcion}</p>
          <p class="card-text fw-bold">Cantidad Disponible: ${this.cantidadDisponible}</p>
          <p class="card-text fw-bold text-primary">Precio unitario: ${this.precioFormateado}</p>
        </div>
        <div class="card-footer text-center">
          ${this.#getBotonPorDisponibilidad()}
        </div>
      </div>
    </div>`;
  }
}
```

La pagina inicial de la aplicación sería “index.html” en el cual nos mostraría algo como lo siguiente:

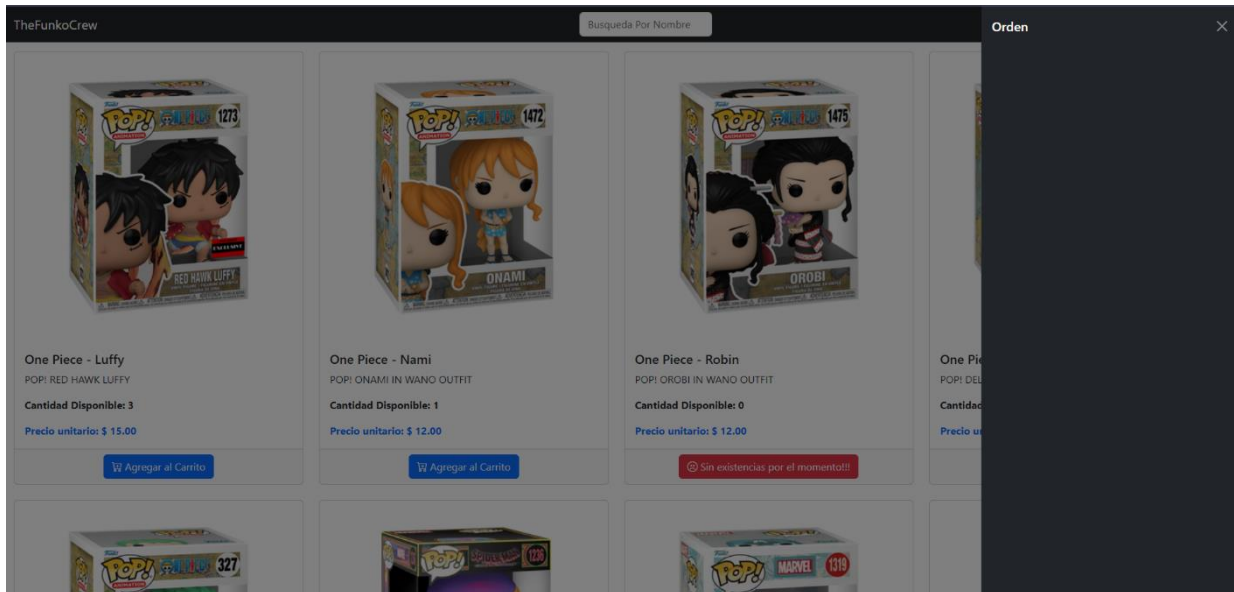


La lista de productos que se muestran están configurados en un JSON que se guarda en el localStorage para una simulación del lado del cliente del servidor en el cual se dejó de una forma que se puedan modificar en el siguiente archivo “localStorageLogic.js”.

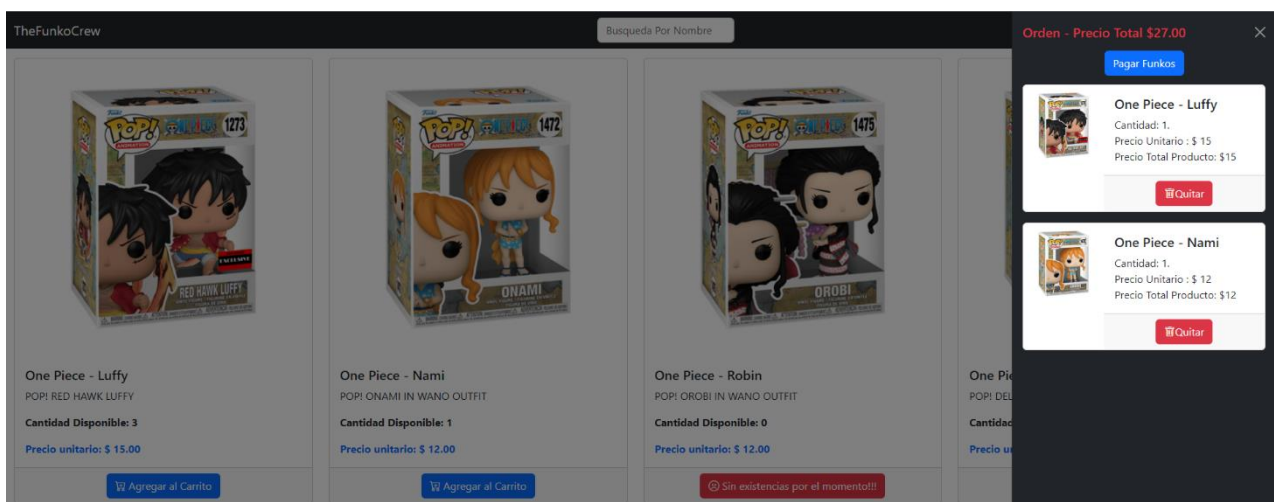
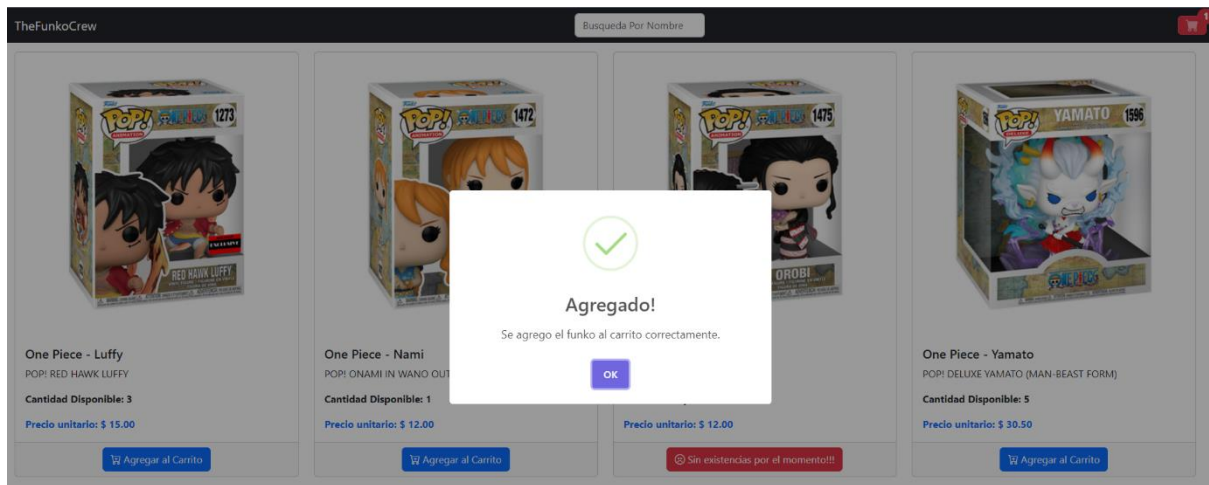
```
js > localStorageLogic.js > ...
2 function verificarCargaInformacionFunkos(){
3   const coleccionFunkosEnMemoria = localStorage.getItem('listaFunkosDisponibles');
4   if (coleccionFunkosEnMemoria == null || coleccionFunkosEnMemoria == undefined){
5     cargarListaFunkos();
6   }
7
8   const coleccionItemsCarrito: string | null = localStorage.getItem("listItemsCarrito");
9   if (coleccionItemsCarrito == null || coleccionItemsCarrito == undefined){
10    cargarCarritoVacio();
11  }
12 }
13
14 //Funcion que carga una lista Json en el localStorage para darle dinamismo al aplicativo
15 function cargarListaFunkos(){
16   const listafunkos = [
17     {id:1, nombre: 'One Piece - Luffy',descripcion:'POP! RED HAWK LUFFY', precio: 15.00, cantidadDisponible: 3, nombreImg: 'OPLuffy'},
18     {id:2, nombre: 'One Piece - Nami',descripcion:'POP! ONAMI IN WANO OUTFIT', precio: 12.00, cantidadDisponible: 1, nombreImg: 'OPnami'},
19     {id:3, nombre: 'One Piece - Robin',descripcion:'POP! OROBI IN WANO OUTFIT', precio: 12.00, cantidadDisponible: 0, nombreImg: 'OProbin'},
20     {id:4, nombre: 'One Piece - Yamato',descripcion:'POP! DELUXE YAMATO (MAN-BEAST FORM)', precio: 30.50, cantidadDisponible: 5, nombreImg: 'OPY'},
21     {id:5, nombre: 'One Piece - Zoro',descripcion:'POP! RORONOA ZORO (GLOW)', precio: 15.00, cantidadDisponible: 5, nombreImg: 'OPzoro'},
22     {id:6, nombre: 'Marvel - Spiderman',descripcion:'POP! JUMBO SPIDER-MAN (MILES MORALES) (BLACK LIGHT)', precio: 45.50, cantidadDisponible: 5,
23     {id:7, nombre: 'Marvel - Capitan America',descripcion:'POP! CAPTAIN AMERICA (RETRO REIMAGINED)', precio: 15.99, cantidadDisponible: 10, nomb
24     {id:8, nombre: 'Marvel - Thor',descripcion:'POP! THOR WITH PIN', precio: 15.99, cantidadDisponible: 10, nombreImg: 'MarvelThor'},
25     {id:9, nombre: 'Marvel - Scarlet',descripcion:'POP! SCARLET WITCH', precio: 11.99, cantidadDisponible: 10, nombreImg: 'MarvelScarlet'},
26     {id:10, nombre: 'Marvel - Ironman',descripcion:'POP! CIVIL WAR: IRON MAN', precio: 15.00, cantidadDisponible: 10, nombreImg: 'MarvelIronMan'}
27     {id:11, nombre: 'Tokyo Ghoul - Ken Kaneki',descripcion:'POP! KEN KANEKI (FINAL BATTLE)', precio: 19.99, cantidadDisponible: 10, nombreImg: '
28     {id:12, nombre: 'Tokyo Ghoul - Juzo',descripcion:'POP! JUZO SUZUYA', precio: 19.99, cantidadDisponible: 0, nombreImg: 'TokyoGhoulJuzoSuzuya'
29   ];
30
31   localStorage.setItem("listaFunkosDisponibles", JSON.stringify(listafunkos));
32 }
33 }
```



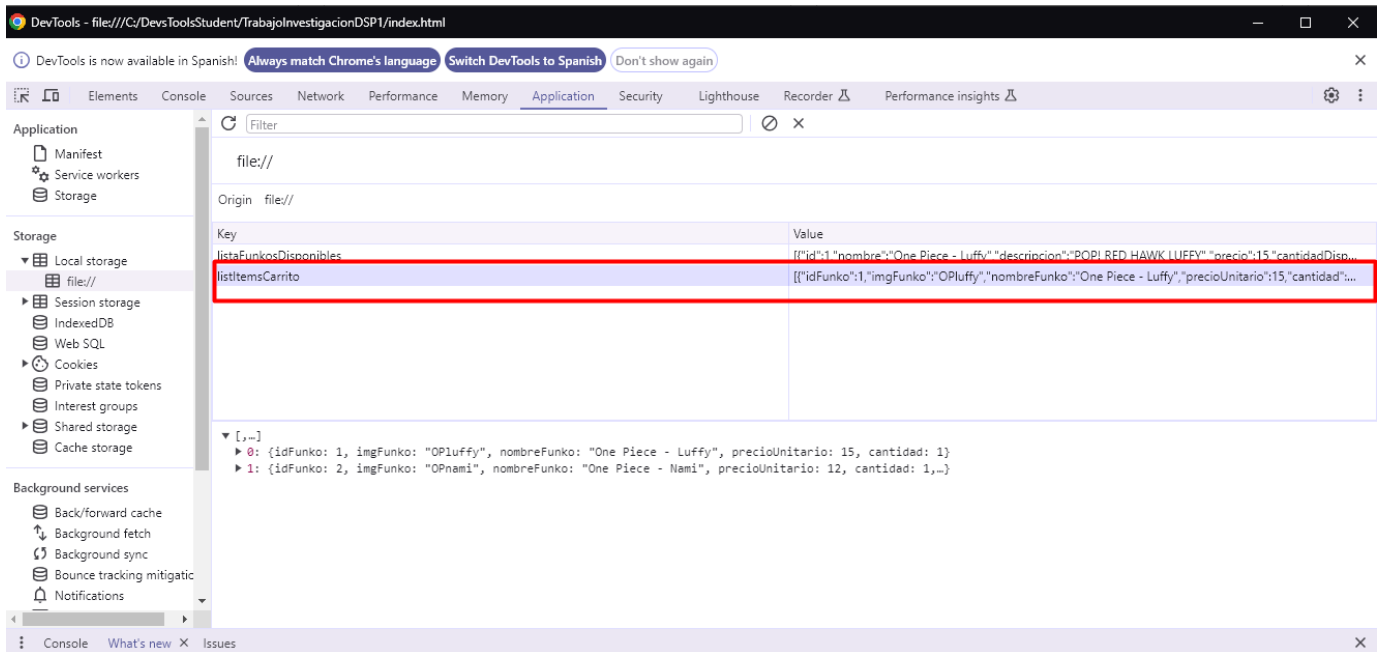
De igual forma con el carrito de compra se realiza la misma lógica del manejo de un json para los objetos en memoria desde el localStorage desde el lado del cliente (navegador)



Que reacciona de acuerdo con lo que se va añadiendo en él.



Y este igual se almacena de la misma forma que la lista de productos.



Teniendo en cuenta que en el localStorage siempre se guardan String (cadena de texto), por lo cual desde la lógica del manejo de objeto en javascript se realizaran las siguientes tareas de transformaciones de datos.

```
29     ];  
30  
31     localStorage.setItem("listaFunkosDisponibles", JSON.stringify(listafunkos); Transformacion de un objeto a un String  
32  
33 }  
34  
35 function cargarCarritoVacio(){  
36     localStorage.setItem("listItemsCarrito", "[]");  
37 }  
38  
39 //funcion para retorna una lista de la clase Funko que seria List<Funko>  
40 function getListFunkosLocalStorage(){ Transformacion de un String a un Json  
41     return JSON.parse(localStorage.getItem("listaFunkosDisponibles"))  
42     .map(a => new Funko(a.id, a.nombre, a.descripcion, a.precio, a.cantidadDisponible, a.nombreImg));  
43 } Transformacion de un elemento del JSON hacia un objeto de la clase especifica Funko por medio del .r  
44  
45 //funcion que retorna los datos del funko seleccionado  
46 function getOneFunkoById(idFunko){
```

Ejecutar la aplicación:

Para ejecutar solo se necesita descargar el código fuente y de una vez dar clic sobre el archivo "index.html".

Este equipo > Disco local (C:) > DevsToolsStudent > TrabajoInvestigacionDSP1

Nombre		Fecha de modificación	Tipo	Tamaño
io	.git	03/03/2024 10:38 p. m.	Carpeta de archivos	
	css	01/03/2024 02:03 p. m.	Carpeta de archivos	
	img	01/03/2024 02:03 p. m.	Carpeta de archivos	
	js	03/03/2024 10:17 p. m.	Carpeta de archivos	
	facturacion.html	03/03/2024 10:21 p. m.	Chrome HTML Do...	4 KB
	index.html	01/03/2024 02:03 p. m.	Chrome HTML Do...	3 KB
	README.md	03/03/2024 10:17 p. m.	Archivo MD	1 KB