MODULO(A)	DWC		
IKASLEA - ALUMNO/A	Ander Sanchez	TALDEA - GRUPO	2DAW3

Documentacion Reto

Gestion de Datos en el Cliente:

En el cliente se usa el localStorage y cookies para almacenar datos en local del usuario.

El localStorage se usa para almacenar la información de las empresas seleccionadas y saber en todo momento cual es la que desea operar; con la función siguiente se almacenan las empresas seleccionadas en el localStorage:

```
function extraerEmpresas() {
  const empresas = document.querySelectorAll('#dRecive .envia');
  if (empresas.length !== 0) {
    let array = [];
    for (let i = 0; i < empresas.length; i++) {
      array.push(empresas[i].id);
      }
    localStorage.setItem('empresas', array);
    router('operar')
    crearEmpresas();
  } else {
    alert('Debe seleccionar alguna empresa para poder operar')
  }}
```

Y en la siguiente funcion extraemos esas empresas y generamos el html y hacemos la llamada a la api para obtener los datos:

```
function crearEmpresas() {

  const empresas = localStorage.getItem('empresas').split(',');

  const contenedor = document.getElementById('listaEmpresas');



  const fecha = new Date();

  const fechaFormateada = fecha.toISOString().slice(0,
10).replace(/-/g, "-");



  empresas.map(function (empresa) {

    let div = document.createElement('div');

    card = `<div class="flex justify-center">

        <div class="rounded-lg shadow-lg bg-white max-w-sm">

          <a href="#!" data-mdb-ripple="true"
data-mdb-ripple-color="light">

            <img class="rounded-t-lg"
src="../../imagenes/${empresa}.png" alt="${empresa}"/>

          </a>

          <div class="p-6">

            <h5 class="text-sky-900 text-xl font-medium
mb-2">${empresa.toUpperCase()}</h5>

            <p class="text-gray-700 text-base mb-4">

              Precio: <span class="text-rose-800"
id="precio${empresa}">${datos}>0</span>

            </p>

            <a href="#!"
onclick="router('grafico','${empresa}');pararSetInterval()">

            <button id="${empresa}"  type="button" class=" inline-block
px-6 py-2.5 bg-sky-600 text-white font-medium text-xl leading-tight
```

```
uppercase rounded shadow-md hover:bg-sky-700 hover:shadow-lg
focus:bg-sky-700 focus:shadow-lg focus:outline-none focus:ring-0
active:bg-sky-800 active:shadow-lg transition duration-150
ease-in-out">Grafico</button>

            </a>

          </div>

        </div>

      </div>`

    div.innerHTML = card;

    contenedor.appendChild(div);

  })

  console.log(empresas, fechaFormateada);

  pedir(empresas, fechaFormateada, 'a')

}
```

De igual manera utilizamos las cookies para almacenar el token que permitirá hacer la petición de la API; donde tenemos un archivo llamado cookies.js que tiene todas las funciones que manejan esta cookie.

Gestion de Panel Drag & Drop:

La Gestión de Drag & Drop la realizo con JQuery de la siguiente manera:

```
$(function () {

        $(".envia").draggable({

            helper: "clone",

            appendTo: "body",

            start: function (event, ui) {

                $(ui.helper).addClass("dragging");
```

```
            },

            stop: function (event, ui) {

                $(ui.helper).removeClass("dragging");

            }

        });

        $("#dRecive,#dEnvia").droppable({

            accept: ".envia",

            drop: function (event, ui) {

                const dragged = $(ui.draggable);

                $(this).append(dragged);

            }

        });

    });
```

Gestión de Promesas:

Para las promesas he utilizado funciones asincronas para poder gestionar mejor el tiempo de respuesta de la API al realizar el fetch, ejemplo:

```
async function apiLogin() {

    try {

        var pass = document.querySelector('#pass').value;

        var email = document.querySelector('#email').value;

        var myHeaders = new Headers();

        myHeaders.append("Content-Type",
"application/x-www-form-urlencoded");

        var urlencoded = new URLSearchParams();

        urlencoded.append("email", email);

        urlencoded.append("password", pass);
```

```
        var requestOptions = {
            method: 'POST',
            headers: myHeaders,
            body: urlencoded,
            signal: AbortSignal.timeout(4000)
        };
        const response = await fetch("http://localhost/api/login",
requestOptions)
        if (response.status === 200) {
            const json = await response.json();
            setCookie('token', json.token, 20);
            location.reload()
        } else {
            throw new Error(`Error en la respuesta del servidor:
${response.status}`);
        }
    } catch (error) {
        console.error(error);
        alert('Error, verifica tus datos e intenta nuevamente');
    }
}
```

También he usado try catch para que la gestión de errores no me falle todo el programa.

Tengo un AbortController que en caso de que la petición tarde más de 4s se anula completamente.

En el intervalo siguiente gestiono la respuesta de la API y en caso de que suba o baje el precio cambio los colores; esta petición se realiza cada 10s:

```
interval = setInterval(async () => {

    datos = await apiPeticiones(empresas, fecha, hora);

    datos.Empresas.map((empresa) => {

      let preioViejo =
Number(document.getElementById(`precio${empresa.empresa}`).text);



      if (Number(empresa.cierre) >= preioViejo)

        document.getElementById(`precio${empresa.empresa}`).style.color
= 'green';

      else

        document.getElementById(`precio${empresa.empresa}`).style.color
= 'red';

      document.getElementById(`precio${empresa.empresa}`).innerHTML =
`${empresa.cierre}`;

    });

  }, 10000);
```