**Resumen funcional del simulador de compra**

Autor: **Prellezo Sergio Emanuel**

Sitio:[**https://mammoth-compras.netlify.app/**](https://mammoth-compras.netlify.app/)

Repositorio:<https://github.com/SerePrec/Mammoth-Compra>

**1. Introducción**

Preparé este resumen a modo de guía para enumerar las funcionalidades que se encuentran en el simulador. Lo que puede ser útil para recordar y poder reutilizar en proyectos futuros alguna de ellas.

Este proyecto nace como una extensión de mi anterior trabajo final en el curso de desarrollo web <https://mammoth-bicicleteria.netlify.app/> , si bien para esta entrega se trata la sección de compras de manera aislada para no mezclar los objetivos. Por eso varios de los links a distintas páginas de mi anterior proyecto se encuentran desvinculadas.

Se dividió el código JS en distintos archivos para organizarlo mejor:

* clase.js: contiene la definición de todas las clases.
* data.js: contiene las funciones AJAX que solicitan los datos con los que voy
* a trabajar en el simulador.
* productos.js: contiene los scripts de la página productos.html.
* compra.js: contiene los scripts de la página compra.html.
* base.js: contiene scripts básicos comunes a todas las páginas (no relacionados a la funcionalidad del simulador sino a elementos de diseño, como por ejemplo el botón fijo de scroll).

También se ordenó e introdujeron diversos comentarios a lo largo del código para hacer más fácil la comprensión de su funcionamiento.

El sito tiene responsive pasando por distintos break-points con el objeto de cubrir la visualización en pequeños celulares, tabletas, notebooks y computadoras de escritorio.

La codificación de los estilos la hice mediante SCSS, utilizando la división en parciales.

Trabajé con Git para versionar mi proyecto y llevar un control sobre los cambios que fue sufriendo. Generé una rama llamada *desarrollo* sobre la cual iba trabajando y a medida que veía que los cambios me parecían correctos, la fusionaba con la master (principal).

Todo el repositorio fue subido a Github, incluyendo las ramas y sub-ramas.

Finalmente, el proyecto fue subido al servidor Netlify para visualizar la funcionalidad del mismo, ya que si se prueba localmente, se debe utilizar algún plugin como el Live server de VSC para que la petición AJAX al JSON estático se lleve a cabo correctamente.

Links a terceros:

* Bootstrap
* Wow anímate
* Fontawesome
* Fuentes de Google

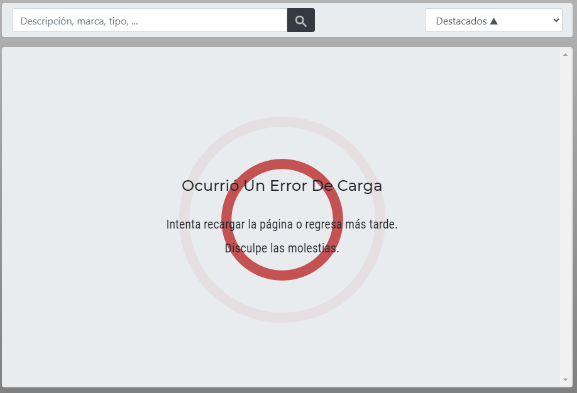
**2. Obtención de datos del simulador**

El simulador para su funcionamiento necesita la información de dos peticiones AJAX. Una de ellas se encarga de traer el array de productos (que luego se instancian con la clase Productos para poder disponer de métodos) desde un JSON estático ubicado en la carpeta /data.

La otra petición, se encarga de obtener el valor oficial del dólar actual desde la API dolarsi.com, ya que los precios de los productos se encuentran en dólares por ser material importado. Una vez obtenida se exhibe con una animación.

Mediante el método .when() de jQuery, se verifica al cargar la página que ambas peticiones resulten exitosas.

De ser así se procede a asignar la mayoría de los eventos relacionados al simulador (ya que de lo contrario no tendría sentido y podría ocasionar resultados no deseados) e inicializarlo con su función principal iniciar().

De lo contrario se muestra un mensaje con animación, de “error de carga”

**Nota:** ha ocurrido durante el desarrollo en donde muy seguido actualizaba la página para ver los cambios, que la API de dolarsi.com no enviaba respuesta (seguramente estaban realizando actualizaciones), por lo que la página daba el mensaje de “error de carga”

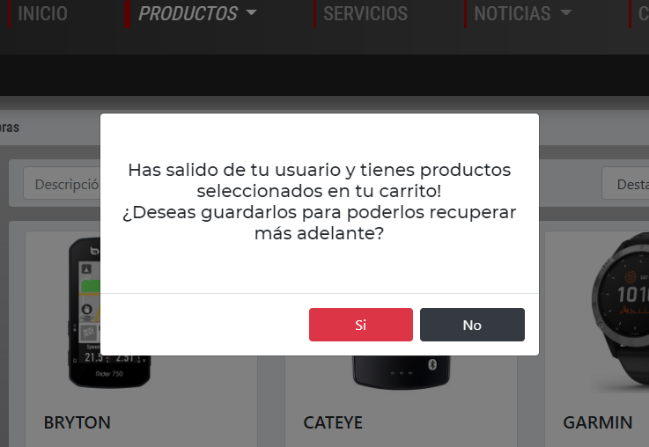
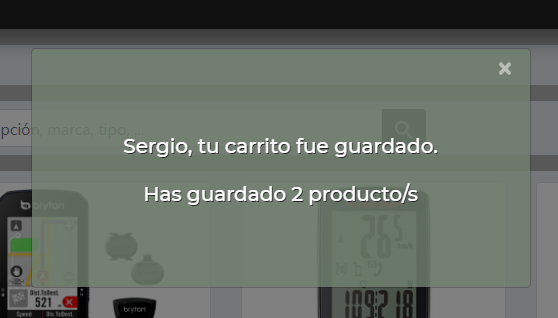
**3. Login / Logout del usuario y storage**

Decidí que la compra se puede realizar estando o no logueado, total al hacer el checkout se solicitan todos los datos necesarios.

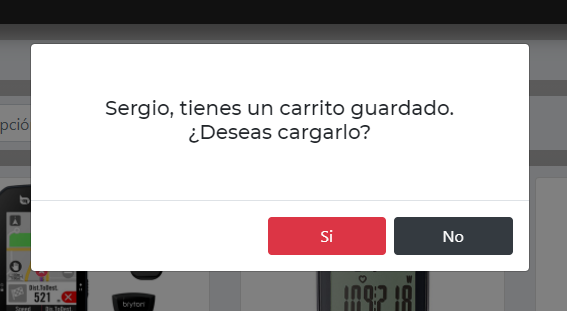
Si el usuario no se loguea el carrito actual se almacena en el sessionStorage permitiendo mantener la selección mientras no se cierre la pestaña del navegador.

Por el contrario si el usuario se loguea tiene dos ventajas:

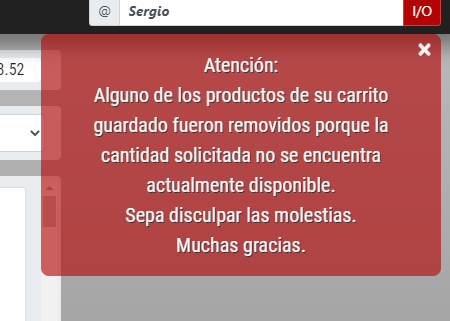
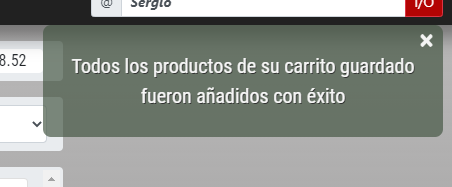
* Permite mantener el carrito actual a pesar de cerrar el navegador ya que el mismo se guarda en el localStorage siempre que permanezca logueado.
* Al desloguearse, si el usuario tiene productos seleccionados, permite guardar distintos carritos para distintos usuarios que se hayan logueado desde el dispositivo, dando la opción de recuperarlo la próxima vez que ese usuario se loguea.

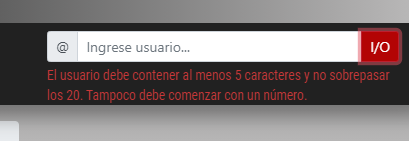


Si se opta por recuperarlo o no, ese carrito ya se borra del localStorage, permitiendo volver a guardar otro al desloguearse nuevamente.

Si el usuario comienza una selección y luego decide loguearse, si tiene un carrito guardado se pregunta que quiere hacer. Si lo carga, sigue con la selección de aquel carrito guardado, de lo contrario, se descarta y continúa con la selección actual.

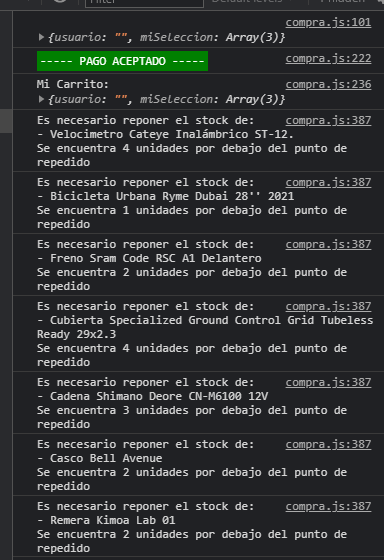
También incluí una verificación importante a la hora de cargar el carrito que tiene que ver con la disponibilidad actual del producto, lo que se puede simular localmente, agregando productos y luego poniendo su disponibilidad por debajo del valor agregado al carrito.

Si todos los productos están disponibles lo alerta con un mensaje verde, si algún producto no se encuentra disponible en la cantidad solicitada, lo advierte con un mensaje rojo y quita ese item por completo del carrito.

De manera arbitraria y para imponer ciertas condiciones, el usuario no debe comenzar con un número y no puede tener menos de 5 caracteres ni más de 20. Lo que advierte con una animación por debajo del input, si no se cumple dicha condición.

**4. Consola**

****A lo largo de todo el proceso se van registrando mensajes importantes en la consola asociados a los “movimientos” de la compra. Esto simularía ser como mensajes de control para a empresa.

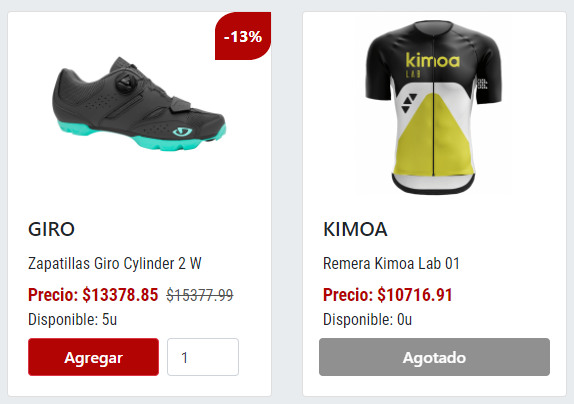
También al finalizar la misma, se realiza una verificación del stock de todos los productos, y se advierte aquellos en que su disponibilidad se encuentra por debajo de su punto de repedido

**5. Tarjetas de productos**

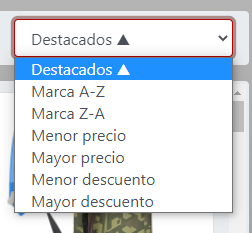
Se generan dinámicamente en función del archivo productos.JSON obtenido inicialmente.

Si posee descuento genera el html necesario para el “globo” del porcentaje de descuento y tacha el precio original exhibiendo el nuevo.

Si el producto no tiene disponibilidad, cambia el botón de agregar y el input de cantidad por un botón con otro formato y disabled

A su vez el rango de valores máximos de cada input asociado a cada producto se va estableciendo de acuerdo al valor de disponibilidad, si bien luego se valida la cantidad por otro lado también

**6. Orden de productos**

Existen 7 opciones de ordenamiento de los productos. Por defecto la página inicia con la opción “Destacados primero” donde ubica primero a los productos destacados ordenados por marca y descripción, luego los no destacados con el mismo orden.

Las otras opciones son orden alfabético A-Z o Z-A por marca y luego descripción, precio ascendente o descendente y por menor o mayor descuento.

Este ordenamiento se puede hacer en cualquier proceso del filtrado de productos

**7. Filtrado de productos**

Existen 5 formas de filtrar los productos:

* Por búsqueda de palabra
* Por categoría
* Por destacados
* Por marcas (correspondiente a la **selección Madre**)
* Por rango de precios (correspondiente a la **selección Madre**)

La **selección Madre** la compone la combinación de los tres primeros filtros, es decir: búsqueda, categoría y destacados.

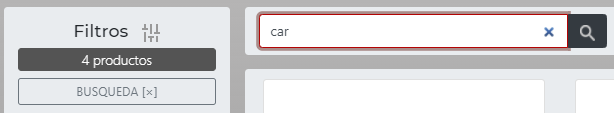
****Es en base a esta **selección madre** que se genera un array de productosFiltradosCliente.

Este array es el punto de partida “fijo” para los otros dos filtros: marcas y rango precios. Es decir estos dos filtros trabajan siempre sobre esa **selección Madre**, pero no la modifican, solo agregan un filtrado extra a partir de ella.

Por eso los extremos del rango de precio se mantienen inalterados siempre que no se toque ninguno de los tres primeros filtros.

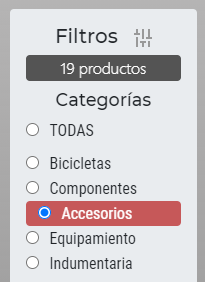
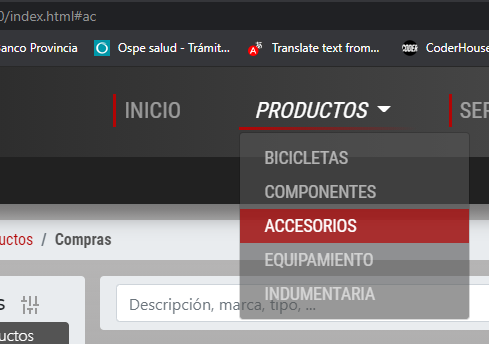
Lo mismo con el listado de marcas que se mantiene junto a las cantidades de productos de cada marca.

Es decir, ambas cosas se mantienen porque la **selección Madre** es la misma, pero si se toca cualquiera de los 3 filtros que la componen, se resetean los valores de los filtros de marcas y rango de precio, tomando los valores correspondientes a la nueva selección **Madre** escogida por el usuario.

*******Búsqueda por palabra:* se activa a medida que el usuario va introduciendo caracteres en el input correspondiente y también es posible activarla al presionar el botón de la lupa (por si algún navegador no fuese compatible con el método anterior).

Al activar esta búsqueda aparece un botón en la barra de filtros dando a entender esta activo éste método. Si se borra por completo los caracteres o se toca el botón de “Búsqueda [x]”, se limpia este filtro.

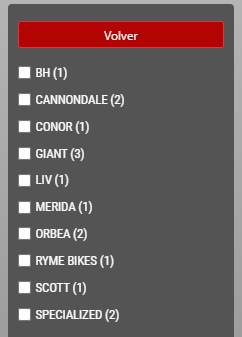
*Filtro por categoría:* Permite circunscribir los productos entre las categorías existentes o todas.

Este también tiene interacción con la carga de la página o cambio del Hash #, es decir que si se accede a la pagina a través de una url con el link al # correspondiente, se pasa a esa selección. Esto puede darse al seleccionar en el navbar a que parte de los productos quiero “ir”

*Filtro por destacados:* Solo muestra los productos que son destacados al activar el checkbox



*Filtrado por marcas:* Muestra un listado con las marcas que contiene la **selección Madre** como también la cantidad de productos que las componen.

****Se pueden seleccionar una o varias marcas y el filtro mostrará los productos de la selección madre que contengan dichas marcas y se encuentren dentro del rango de precios elegido.

Se puede volver a ingresar y cambiar dicha selección a gusto.

**Importante:** Esta selección se resetea automáticamente al cambiar cualquiera de los 3 primeros filtros enumerados.

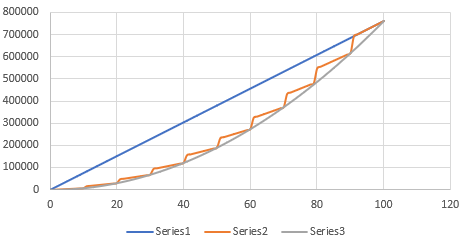
Para saber si tenemos activa alguna selección del filtro marcas el logo a la derecha del botón pasa del estado OFF al ON si hay marcas seleccionadas.

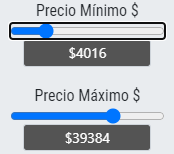
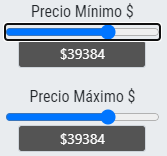


*Filtro por rango de precio:* Permite elegir el valor del producto desde un precio mínimo a uno máximo.

Respecto a la escala quiero decir dos cosas:

* Los valores mín y máx se setean de acuerdo a la **selección Madre.** Se recorre el array de productos que la componen y se sacan los valores extremos de los precios
* Como puede haber mucha diferencia entre los extremos de precio, generé una escale exponencial, en donde crece más suavemente al comienzo permitir más fácilmente precios del rango inferior y luego crece más rápido a medida que se acerca al extremo máximo.

En azul se ve una escala lineal y la que se adopto en la curva color gris.

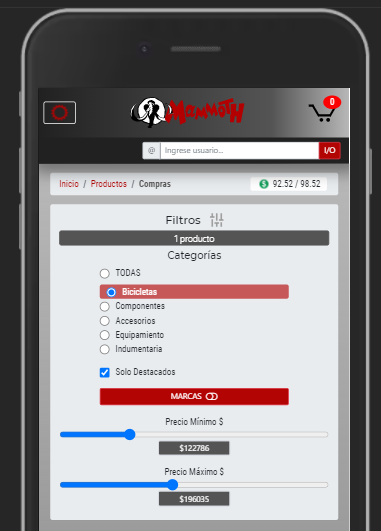
Realicé mediante funciones asociadas a los eventos del control del rango de precios la verificación para que no permita la inconsistencia de que el precio mín se elija por encima del máximo. En caso de suceder esto, inmediatamente al salir de la selección, los indicadores se acomodan automáticamente al valor máximo permitido para el cruce (min = max).

Ejemplo al querer pasar con el precio mínimo el valor del precio máximo.

Solo se ajusta al valor máximo válido para esta situación.

También generé eventos asociados al mouse y al teclado para estos input, así se pueden manejar también mediante Tab y las flechas del teclado logrando un ajuste fino.

Por último estos filtros en versión mobile, se ven contraídos y si se apaísa el dispositivo dando origen a la columna izquierda de filtros, vuelven a ser visibles.

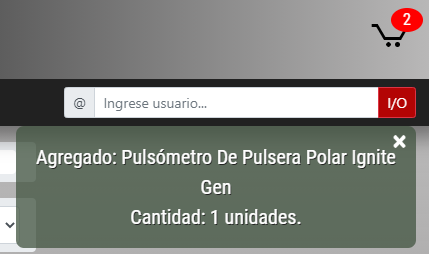
****En el caso del celular vertical, inicialmente aparece contraído y tocando sobre la leyenda de filtros, se despliega completamente haciendose visible. Lo mismo con el botón de marcas.

**8. Agregando o quitando elementos del carrito**

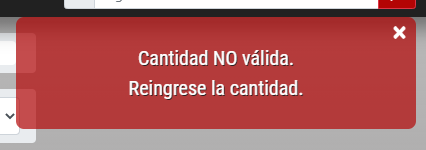
Agregando productos:

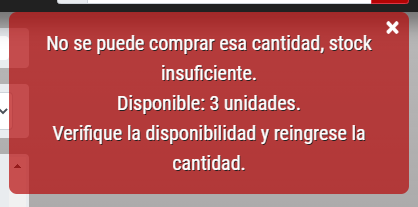
Para incorporar productos al carrito se puede simplemente apretar el botón agregar, la cantidad de veces correspondiente a la cantidad de unidades a comprar o seleccionar las unidades desde el input derecho y luego apretar agregar.

En cualquiera de los casos, las opciones repetidas se incrementan en cantidad en el carrito, sin repetir el mismo producto.

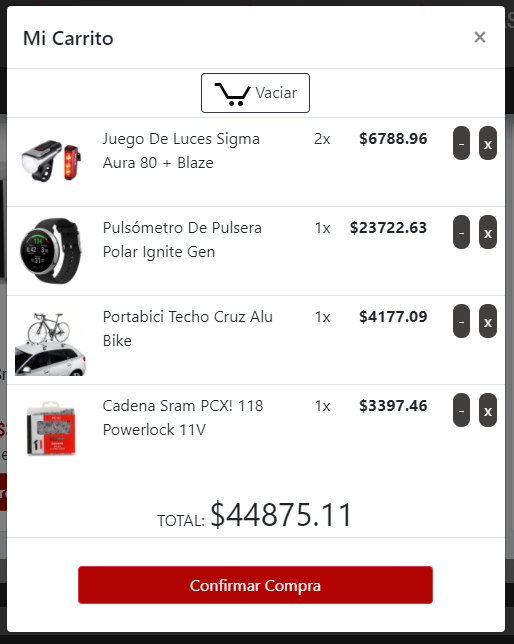
****Al agregar el producto exitosamente, aparece un mensaje confirmando la operación y el icono animado del carrito se actualiza.

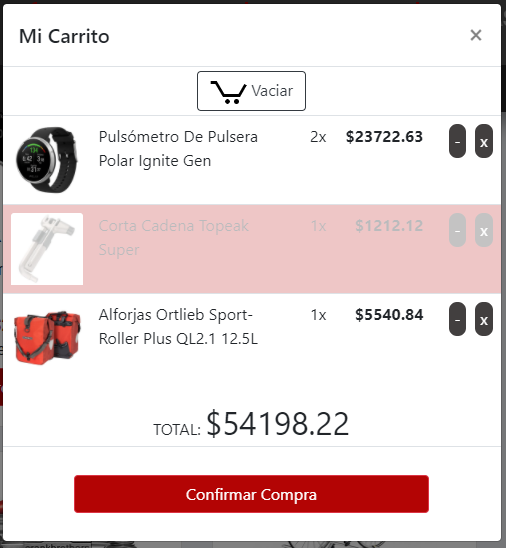
Si bien constantemente con cada operación se van actualizando los valores y atributos de las tarjetas y con las flechitas del input number no es posible seleccionar una cantidad no acorde, si es posible hacerlo al ingresar el valor a mano, por eso las siguientes validaciones.

****Si se intenta ingresar un valor inválido de cantidad (ej. Número negativo), se advierte de ello.

****Lo mismo si se intenta ingresar a mano una cantidad mayor a la que se encuentra disponible.

Quitando productos:

Hay 3 formas de quitar elementos del carrito: restando de a una unidad, eliminando el elemento completo o vaciando el carrito. Esto se hace a través de los correspondientes botones.

En cualquiera de estos casos, como sucede al agregar, se actualiza la información de las tarjetas, del contenido del carrito y del icono del carrito, visualizándose los cambios inmediatamente. Al desaparecer uno o más elementos, se produce una animación en donde se desvanecen rápidamente tornándose de color rojo y luego que se completa, se regenera el carrito junto con el importe total.

Como se mencionó anteriormente, con cada interacción del carrito ya sea agregando o quitando elementos, se regenera la información asociada.

Respecto a las tarjetas en lugar de usar la función mostrarProductos, generé una nueva que es actualizarInfoTarjetas, porque al tener imágenes y ser muchas, podía ocasionar parpadeos o demoras en la reconstrucción de toda la información, pero como en este caso todo sigue en el mismo lugar (no como al ordenar las tarjetas), era más optimo reescribir solo lo que se modificaba. Es más en algunos casos, con el efecto de regenerar todo el contenido aparte del parpadeo se perdía el scroll actual dentro del contenedor.

Entonces mediante comprobaciones respecto a la disponibilidad, se regeneran los atributos de máximo de los inputs cantidad, la disponibilidad y el diseño del botón. Pasando de uno a otro diseño de acuerdo a si el producto tiene o no stock disponible.

**A dif de función mostrar se act tarjetas apar ahorrar recursos**

**Act de valores extremo y botón con lo anterior**