El ejercicio consiste en una *demo* de una aplicación, *ListaApp*, para gestionar una lista de la compra. *Los números identificativos de la imagen aparecen como comentarios en el código html entregado.*

Obligatorio

Solo se puede editar el archivo de app.js. Usar de la cláusula `'use strict'`. La creación y eliminación de nodos deberá hacerse por medio de métodos adecuados, no con propiedades. Siempre que se trabaje con arrays, utilizar sus métodos. Los puntos 1, 2 y 3 deberán estar operativos con las 2 listas entregadas (carpeta *datos*). **La URL del servidor será** `http://192.168.14.101:3000` No cumplir estas normas acarreará la nulidad de esta parte de la prueba.

1. Inicio

- La app contará con una sola variable global, `listaApp`, y todo el código residirá en una función invocada de forma inmediata o IIFE que exportará el método `iniciar` al objeto `listaApp`.
- Al lanzarse la aplicación, ③ y ⑧ aparecen deshabilitados y ④ no aparece. El nombre de la lista a cargar se introduce en ①. Cuando se carga, al pulsar *enter* o en el botón de 'Cargar' se muestra el nombre en ②.
- La carga de los archivos se realizará mediante CORS siempre que no exista ya la lista en el *localeStorage* en cuyo caso se leerá del mismo. La función encargada de CORS deberá implementarse mediante una función con `2` argumentos: el archivo a cargar y la función a llamar cuando se reciban los datos. El único argumento de ésta última recibirá la lista en forma de texto: los campos del primer registro efectivo son los nombres de las propiedades del objeto. El separador de registros es el carácter `^` y el de campo, `|`

2. Listas

- La lista que se recibe debe convertirse en un array de objetos con las propiedades del primer registro (que no se añade al array). La conversión se debe realizar con el método **reduce** o **map** de arrays. Las normas para que se pueda integrar un registro en el array de objetos es que cuente con `3` campos y que éstos cumplan que: el primero tenga una longitud mayor a `1` y no puede existir ya en el array; el segundo debe tener un valor reflejado en los tipos entregados (ver archivo *app.js*); el tercero debe ser `si` o `no`. Sin influir, en ningún caso, si están en mayúsculas y/o minúsculas y eliminando los espacios al inicio y final de los campos. Además, se añadirá una cuarta propiedad, **`id`**, que se asignará de forma consecutiva desde `0` con formato de `3` caracteres (pe. `004`) Una vez convertida la lista en un array de objetos, se ordenará de forma ascendente por la primera propiedad. El array de objetos se guardará con el nombre de la lista en el *localStorage*.

3. Gestión del array de objetos

- El array del punto 2 se mostrará en④ según la estructura entregada en *app.js* y por medio de un método que lo reciba como argumento. La segunda propiedad de cada objeto se reflejará con el atributo *dataset*. Los elementos con la tercera propiedad a `si` aparecerán tachados, ⑤, por medio de la clase `encesta` en el *span* y no se podrán editar por lo que ⑥ tendrá aplicada la clase `esconder` en la etiqueta *big* correspondiente. Ambas clases ya están definidas en el css entregado. En ⑨ se reflejará los artículos que quedan por encestar respecto del total.

4. Acciones

Con la lista reflejada en ④, el usuario podrá realizar las siguientes acciones: agregar/modificar un elemento, eliminar un elemento o marcar un elemento como que está en la cesta. El gestor de eventos de todas ellas deberá residir en el nodo ul que contiene a los li. Los cambios se realizarán sobre el array de objetos ordenándose siempre que sea necesario. Una vez realizados los cambios, se volverá a mostrar el array en ④ (punto 3). Todas las modificaciones se registrarán en el objeto correspondiente del *localeStorage*

- *Agregar*. El usuario podrá introducir el texto correspondiente en ③ de la forma `elemento|tipo`. Si hay texto y cumple las normas dadas anteriormente para la primera y segunda propiedad del objeto, al pulsar *enter* o en ⑧, objeto se agregará al array. Su `id` será el primero que no esté asignado (de `0` en adelante). Si no hay huecos, el más alto más uno. El contenido de ③ se borrará.
- *Modificar*. Al pulsar el icono de lápiz (⑥), el artículo aparecerá en ③ y ⑧ cambiará su contenido a `Modificar`. Si hay texto y cumple las normas dadas anteriormente para la primera propiedad, al pulsar *enter* o en ⑧, el contenido del elemento original se sustituirá por el nuevo. El botón recuperará el valor de '`Agregar`'. El contenido de ③ se borrará.
- *Eliminar*. Si el usuario pulsa sobre la ⊗ en ⑦, el objeto correspondiente se eliminará del array.
- *Marcar*. Si el usuario pulsa sobre el espacio que ocupa un artículo (salvo la ⊗ o el lápiz), el artículo aparecerá tachado (ver ⑤) por medio de la clase `encesta`. Al elemento contenedor del lápiz, se le aplicará la clase `esconder`. Si pulsa sobre un artículo ya marcado, se desmarcará y aparecerá su lápiz.