# PRÁCTICA 5.05 Formularios y persistencia de datos

### Normas de entrega

- En cuanto al **código**:
  - en la **presentación interna**, importan los **comentarios**, la claridad del código, la significación de los nombres elegidos; todo esto debe permitir considerar al programa como **autodocumentado**. No será necesario explicar que es un `if` un `for` pero sí su funcionalidad. Hay que comentar las cosas destacadas y, sobre todo, las **funciones** y **clases** empleadas. La ausencia de comentarios será penalizada,
  - en la **presentación externa**, importan las leyendas aclaratorias, información en pantalla y avisos de ejecución que favorezcan el uso de la aplicación,
  - todo el código debe estar situado dentro del evento `window.onload = () => {};` o a través del evento `document.addEventListener("DOMContentLoaded", () => {});` ,
  - si no se especifica lo contrario, la información resultante del programa debe aparecer en la consola del navegador `console.log(información)`,
  - los ejercicios deben realizarse usando ***JavaScript* ES6** y usar el modo estricto (`use strict`) No se podrá utilizar *jQuery* ni cualquier otra biblioteca (si no se especifica lo contrario en el enunciado),
  - para el nombre de **variables**, **constantes** y **funciones** se utilizará *lowerCamelCase,*
  - para la asignación de eventos se utilizará `addEventListener()` indicando sus tres parámetros en su definición,
  - se usarán las funcionalidades `import` y `export` para crear bibliotecas de funciones temáticas (no debe haber declaración de funciones ni objetos en el documento principal).
- En cuanto a la **entrega** de los archivos que componen los ejercicios:
  - todos los ejercicios en **una carpeta** (creando las **subcarpetas** necesarias para documentación anexa como imágenes o estilos) cuyo nombre queda a discreción del discente,
  - el nombre de los ficheros necesarios para resolver el ejercicio será el número de ejercicio que contenga,
  - el código contendrá ejemplos de ejecución, si procede, y
  - la carpeta será comprimida en formato **ZIP** y será subida a **Aules** de forma puntual.

**Ejercicio 1 - Mi colección de discos**

**Parte I. Prepara un formulario** diseñado para almacenar los discos con los datos listados a continuación:

- nombre del disco,
- carátula del disco (su **URL**),
- grupo de música o intérprete,
- año de publicación,
- género de música mínimo cuatro),
- localización que guardará un código (inventado pero que contiene números y letras),
- prestado que almacenará un valor booleano (por defecto será `false`).

Tendrá un botón **Guardar** que añadirá el disco a un listado almacenado en un objeto **JSON** (que tendrás que diseñar) y otro **Mostrar** que mostrará el listado de discos debidamente formateado (usa **CSS**).

**Parte II.** Antes de añadir el disco al objeto **JSON** debe ser comprobado. Crea las **funciones necesarias para validarlo** teniendo en cuenta:

- nombre del disco tiene, al menos, cinco caracteres y es obligatorio,
- grupo de música o intérprete posee, al menos, cinco caracteres y es obligatorio,
- año de publicación dispone de cuatro caracteres numéricos,
- tipo de música comprobará si se ha seleccionado alguno,
- localización tiene el formato **ES-001AA** donde **001** es el número de la estantería y **AA** la balda (combinación de dos letras mayúsculas),
- prestado y carátula no tienen comprobación.

En caso de que se produzca un error en la validación, el campo del formulario implicado será destacado con un estilo **CSS** adecuado. En cuanto ese campo contenga un valor válido, volverá a su estilo original.

Además existe un **contenedor de información** que, si se ha producido un error, mostrará un mensaje informando de qué campo (o campos) es el incorrecto y cómo solucionarlo (los insultos son opcionales). Ubícalo donde estimes oportuno y añádele el formato **CSS** que necesites.

**Parte III.** Añade un **input** texto y un botón **Buscar** que permita **filtrar** los discos según el texto introducido por el usuario. Junto a él existirá otro denominado **Limpiar** que volverá el listado a su formato original.

**Parte IV.** Crea un sistema que permita **eliminar un disco** de la colección. Para ello añade un botón para eliminar (un icono puede ser una buen idea) en cada uno de ellos que permita quitar ese disco del listado. Se debe confirmar la acción de borrado.

**Parte V.** Da **persistencia** a los datos del listado utilizando **localStorage**. Los datos se cargarán en un objeto **JSON** en la carga de la página y se guardarán cada vez que se modifique el listado (se añada o se elimine un disco).