

# Primer parcial de Laboratorio III

Crear una aplicación Web que permita realizar un ABM de distintas ropas para una tienda.

Crear la siguiente jerarquía de clases en **Typescript** (en el namespace **Entidades**):

- a. **Ropa**: código(entero), nombre(string) y precio(floatante) como atributos. Un constructor que reciba tres parámetros. Un método, **ropaToString** que retorne la representación de la clase en formato cadena.
- b. **Campera**, hereda de **ropa**, posee como atributos **talle(cadena)** y **color(string)**. Un constructor para inicializar los atributos. Un método **camperaToJson** que retornará la representación del objeto en formato JSON. Se debe de reutilizar el método **ropaToString**.

Crear en **TypeScript** la clase **Manejadora** (en el namespace **Test**) que posea los siguientes métodos y funcionalidades:

**1.- AgregarCampera.** Tomará los distintos valores desde la página *index.html*, creará un objeto de tipo **Campera**, que se enviará (por **AJAX**) junto al parámetro **caso** (con valor "agregar"), hacia **"/BACKEND/adminstrar.php"**. En esta página se guardará al ciudadano en el archivo **"/BACKEND/camperas.json"**.

**2.- MostrarCamperas.** Recuperará (por **AJAX**) a todas las camperas del archivo .json (caso="mostrar") y generará un listado dinámico (en el **FRONTEND**) que mostrará toda la información de cada una de las camperas. Agregar columnas al listado que permitan: **Eliminar** y **Modificar** a la campera elegida.

**3.- EliminarCampera.** Eliminará a la campera del archivo (por **AJAX**) (caso="eliminar"). Recibe como parámetro al objeto **JSON** que se ha de eliminar. Pedir confirmación, mostrando código y talle, antes de eliminar. Refrescar el listado para visualizar los cambios.

**4.- ModificarCampera.** Mostrará todos los datos de la campera que recibe por parámetro (objeto **JSON**), en el formulario. Permitirá modificar cualquier campo, a excepción del código. Modificar el método **AgregarCampera** para cambiar el caso de "agregar" a "modificar", según corresponda.

**5.- FiltrarCamperasPorColor.** Mostrará (por **AJAX**) (caso="mostrar") un listado dinámico (en el **FRONTEND**) de todas las camperas según el color seleccionado en el combo (cboColor).

**6.- CargarColoresJSON.** Cargará (por **AJAX**) (caso="colores") en el combo (cboColor) con el contenido del archivo **"/BACKEND/colores.json"**. Si se invoca varias veces, no se deberán repetir los colores.

**7.-** Crear en TypeScript las funciones necesarias para verificar que **todos** los campos de la página, **index.html**, sean enviados correctamente.

La función se llamará **AdministrarValidaciones** y será la encargada de invocar a otras funciones que verifiquen:

Campos no vacíos (todos).

**\*ValidarCamposVacios(string): boolean.** Recibe como parámetro el valor del atributo id del campo a ser validado. Retorna true si no está vacío o false caso contrario.

Talles correctos (talle).

**\*ValidarTalles(string, string[]): boolean.** Recibe como parámetro el valor a ser validado y los valores permitidos para los talles (S, M, L, X, XL, XXL). Retorna true si el valor pertenece a los talles o false caso contrario.

Selección del código (código).

**\*ValidarCodigo(number): boolean.** Recibe como parámetro el valor del código a ser validado y retorna true si el mismo es mayor o igual a 523 y menor a 1000. False caso contrario.

Si algún campo no pasa la validación, se mostrará un \* (asterisco) al lado de dicho campo y no se permitirá el envío de la información. Si todos los campos pasan todas las validaciones, se enviará la información correspondiente.

**Aplicarlo tanto para el alta como para la modificación de las camperas.**

**8.-** Generar, luego de la transpilación, solo un archivo .js (denominado **app\_primer\_parcial.js**) que contenga todo el contenido necesario para interactuar con *index.html*.

**9.- LimpiarForm.** Vaciará todos los campos del formulario y colocará el combo (cboColor) en “Azul”. Este método se invocará cada vez que se realice una acción sobre el formulario.

**10.- AdministrarSpinner.** Este método mostrará u ocultará el archivo “./BACKEND/gif-load.gif”, de acuerdo al parámetro booleano que recibe como parámetro. Aplicar la llamada de este método en cada acción que invoque a un AJAX.

**11.-** Agregar a la clase Campera el campo foto. Generar un ABM incluyendo la foto, que se guardarán en “./BACKEND/fotos”.

Al modificar o eliminar, se deberá colocar la foto (modificada o borrada) en la carpeta “./BACKEND/fotos” o “./BACKEND/fotos”, según corresponda.

**12.-** Agregar una pre-visualización de la foto tanto para el alta como para la modificación.

## IMPORTANTE:

***Se pueden bajar templates de internet o traer código hecho, pero en ningún caso se debe incluir código obsoleto o que no cumpla ninguna función dentro del parcial.***

***Toda la comunicación con el backend, se realizará con AJAX. El pasaje de datos, con JSON.***

**Los puntos se evaluarán de manera descendente y consecutiva.**