DESARROLLO WEB EN ENTORNO CLIENTE

PREPARACIÓN PRIMERA EVALUACIÓN

1. Queremos implementar un script en Javascript que nos permita gestionar temperaturas mensuales [suponemos que el mes tiene 30 días] relativas a varias estaciones meteorológicas situadas en distintas ciudades. Para ello vamos a crear la clase **`CentralMedidas`** con las siguientes características:

Clase **`CentralMedidas`**			[Definida mediante función prototipos]
Propiedades			
Nombre	Tipo	Visibilidad	Descripción
`Medidas`	`entero[][]`	privada	Contiene una tabla de enteros correspondientes a las medidas tomadas durante un mes en distintas ciudades. Inicialmente el array está vacío.
Métodos			
Nombre	Parámetros	Devuelve	Descripción
`insertaMedidas(ciudad,valores)`	`ciudad:cadena` `valores:entero[]`	`booleano`	Crea una nueva fila en la tabla de medidas correspondiente a la ciudad **`ciudad`** con los valores del array **`valores`** Devuelve verdadero si se ha podido insertar las medidas o falso si no se han podido insertar (pues la ciudad ya existe, o el segundo parámetro no contiene 30 valores).

`actualizaMedida(ciudad,valor,dia)`	`ciudad:cadena`
	`valor:entero`
	`dia:entero`
		`booleano`	Actualiza la medida correspondiente a la ciudad **`ciudad`** y el dia **`dia`** con el valor **`valor`**.
			Devuelve verdadero si se ha podido actualizar la medida o falso si no se ha podido actualizar (pues la ciudad no existe).

`mediaMedidas(ciudad)`	`ciudad:cadena`	`entero`	Devuelve la temperatura media de la ciudad indicada durante el mes
`eliminaCiudad(ciudad)`	`ciudad:cadena`	`booleano`	Elimina las medidas correspondientes a la ciudad.. Devuelve verdadero si se ha podido eliminar la fila o falso si no se ha podido eliminar (pues la ciudad no existe).
`imprime()`	-	-	Se muestran por consola las medidas correspondientes a cada ciudad.
`toHTML()`	-	`cadena`	Devuelve una cadena que representa una tabla HTML con la misma estructura que la propiedad **`medidas`**, añadiendo una columna adicional que representa la media de las medidas del mes para esa fila.

Además, podrán crearse aquellos métodos que se considere necesario para la consecución del ejercicio.

La tabla **medidas** tendrá 31 columnas (una primera columna para el nombre de la ciudad y las 30 siguientes que contienen las medidas del mes) y una fila por cada ciudad. Por ejemplo, si tenemos 5 ciudades (Santander, Oviedo, Madrid, Valencia y Cádiz) la tabla sería la siguiente:

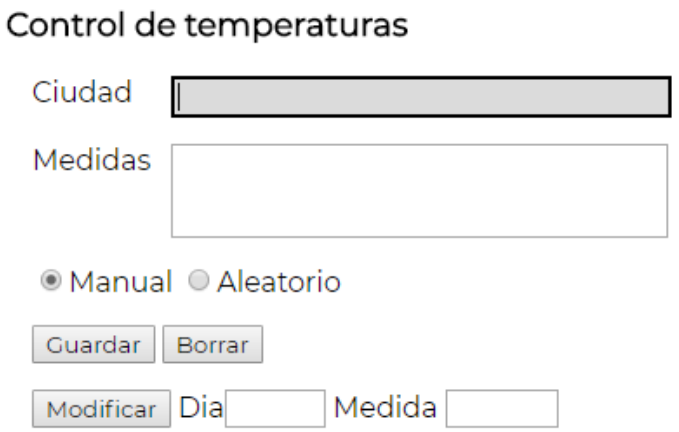
	0	1	2	3	...	30
0	Santander	5	6	10	...	2
1	Oviedo	4	2	8	...	0
2	Madrid	2	1	-1	...	-5
3	Valencia	8	10	15	...	5
4	Cádiz	10	12	18	...	7

Las medidas pueden tener un valor en el rango comprendido entre -5 y 40.

Deberán implementarse los siguientes archivos:

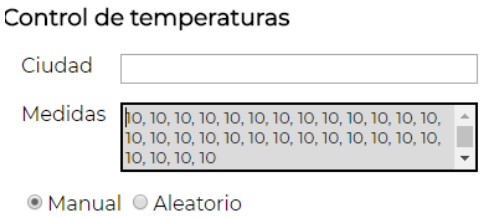
- **`centralmedidas.js`**. Definición de la clase **`CentralMedidas`**.
- **`central_medidas_prueba.js`**. Prueba de la clase.

Por otra parte, contamos con el siguiente formulario, definido en **`temperaturas.html`**:

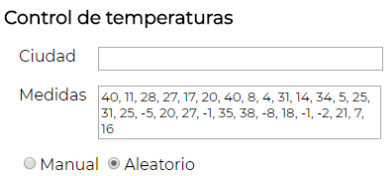


El formulario nos va a permitir añadir nuevas ciudades (campo **Ciudad**) y medidas (campo **Medidas**).

- El texto que escribamos en el campo de Ciudad se convertirá a mayúsculas de manera automática.
- En función de los botones de radio se podrán introducir las medidas de dos maneras:
  - **Manual**. Se introducirán 30 valores, separados entre comas:



  - **Aleatorio**. Los 30 valores se generarán de manera automática (con valores en el rango permitido), pasando el campo a ser de sólo lectura:

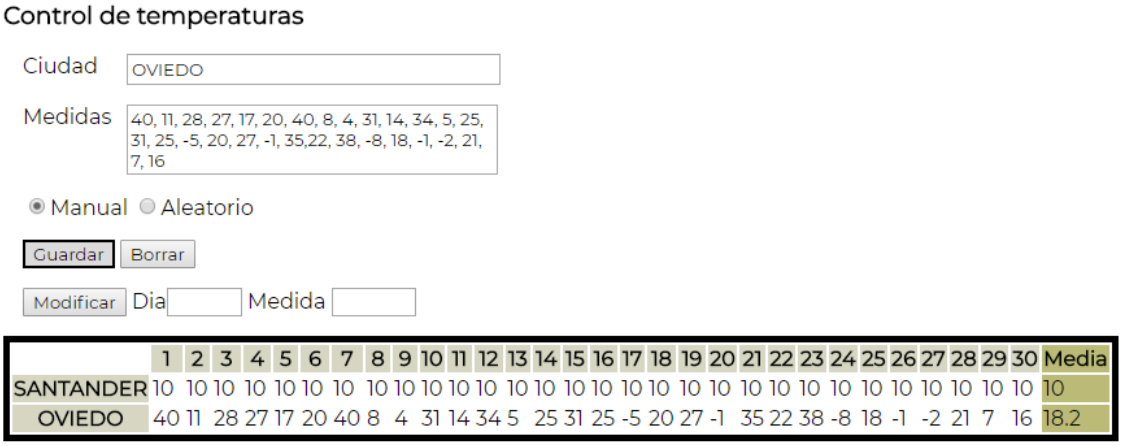


Al pulsar uno de los botones de radio, el valor del campo medidas deberá borrarse.

- Al pulsar en el botón ***Guardar,*** se deberán realizar las siguientes validaciones:
  - El campo de nombre tiene contenido.
  - El campo de nombre no está repetido (no existe en la tabla).
  - El campo de medidas contiene 30 valores separados por comas.
  - Todos los valores están en el rango de temperaturas permitido.

  En caso de haber algún error, se mostrará en la capa con identificador **`errores`** (que debe ocultarse una vez que se han subsanado los errores):

  Si las validaciones son correctas, se mostrará en la capa con id **`tabla-medidas`** una tabla con las ciudades que se han insertado hasta el momento, incluyendo la ciudad actual. Por ejemplo, en la siguiente captura se han insertado dos ciudades:



Además, la capa con id **`temperatura-media`** mostrará la temperatura media de todas las ciudades:

- Al pulsar el botón ***Borrar*** se consultará el campo Ciudad. En caso de no estar vacío y contener un nombre de ciudad existente, se borrará la fila correspondiente, actualizando la tabla y la temperatura media. Si las validaciones no son correctas, se mostrará el error en el campo de error.
- Al pulsar el botón ***Modificar*****,** se permitirá modificar la medida correspondiente a la Ciudad y Día correspondiente. Para ello:
  - Los tres campos deben contener un valor
  - El campo *Ciudad* debe tener una ciudad existente.
  - El campo *Día* debe tener un valor entero entre 1 y 30.
  - El campo *Medida* debe tener un valor entero entre -5 y 40.

  Si las validaciones son incorrectas, se mostrará un mensaje en la capa de error. En caso contrario, se actualizará la tabla de valores y la temperatura media.

2. Vamos a gestionar una lista de países haciendo uso de una interfaz Web. Para ello se van a implementar los siguientes elementos:
   - Clase `Pais` (archivo `pais.js`). Encapsula las propiedades de un país.
     - Contiene propiedades públicas que permiten almacenar los datos correspondientes a un país:
       - Código único de 2 letras
       - Nombre
       - Población en millones de habitantes.
     - Además se implementarán los siguientes métodos:
       - Constructor con parámetros
       - Método `toString()`. Devuelve una representación en formato cadena de las propiedades concatenadas.

     En el archivo **`pais_prueba.js`** se instanciarán 3 países de ejemplo y se mostrarán sus propiedades por la consola de depuración.

   - Clase `ListaPaises` (archivo `lista_paises.js`). Permite manejar un listado de países. Incluye:
     - Una propiedad privada que permite almacenar un listado de países.
     - Métodos que permitan:
       - Devolver el número de elementos del array.
       - Devolver el elemento que está en la posición i-esima.
       - Añadir un elemento al principio del array.
       - Añadir un elemento al final del array.
       - Comprobar si existe un elemento con un código determinado.
       - Borrar el elemento con un código determinado.
       - Borrar un elemento al principio del array (devolviendo el nombre del mismo).
       - Borrar un elemento al final del array (devolviendo el nombre del mismo).
       - Muestra la posición en la que se encuentra un elemento dado su nombre.
       - Devolver el país de mayor población
       - Devolver el país de menor población.

     En el archivo `lista_paises_prueba.js` se instanciará un objeto de la clase `ListaPaises` y se realizarán pruebas que permitan comprobar si los métodos funcionan correctamente. Toda la información correspondiente a las pruebas se mostrará en la consola de depuración.

   - Crea una página HTML que llamarás **`paises.html`**. La página tendrá un script asociado llamado **`paises.js`** que contendrá las funciones de manejo de eventos y carga de la página. Está permitido el uso de una única variable global con el listado de países (tipo `ListaPaises`). La página HTML contendrá un formulario que constará de los siguientes elementos:
     - Campos de texto con las etiquetas Código, Nombre y Población.
       - Aunque el campo código se introduzca en minúsculas, una vez introducido se convertirá automáticamente a mayúsculas.
     - Un botón que permita ***Añadir País***
       - Al pulsar el botón deberíamos comprobar (si es posible hacer uso de expresiones regulares):
         - Que el código tiene dos letras.
         - Que el campo población es numérico.
         - Que todos los campos se han introducido
         - Que no existe el país con el código introducido en el campo de texto

       En caso de que las validaciones sean correctas se guardará el país, actualizando el listado de países y borrando los campos de texto. Si alguna de las validaciones es incorrecta se mostrará la información en la capa pertinente.

- Un botón que permita ***Eliminar País***.
    - Al pulsar el botón se eliminará país cuyo código coincide con el introducido en el campo código. Si no se introduce ningún código, se eliminará el primer país. En cualquier caso, se actualizará la información del listado de países.
        - Si no hay países se mostrará el mensaje en la capa de información.
    - Una capa con el identificador información que mostrará retroalimentación de la última acción que se ha realizado (si se ha insertado/borrado un país), así como los correspondientes errores que se vayan produciendo. Dicha capa debería tener un formato diferenciado.
    - Una capa con el identificador listado que mostrará el listado de países actualizado (o un mensaje indicativo si no hay ningún país). Dicha capa debería tener un formato diferenciado.

  Ten en cuenta que el formulario no va a tener botón de envío (*submit*) ni atributos *action* ni *method*, pues no lo enviaremos a ningún script de servidor (toda la información de este ejercicio es gestionada en local).

  Si se desea pueden utilizarse entradas de formulario de HTML5, pero deberán realizarse las validaciones oportunas en previsión de que el navegador no implemente el tipo de entrada.

3. Modifica el ejercicio anterior para añadir un botón que permita ***Guardar Países***. Al pulsar el botón se guardará el listado de países en una cookie con caducidad de un día (en el formato que se prefiera). Al cargar la página, habrá que consultar si existe la cookie con el listado de países, actualizando tanto el array como la información que se muestra en pantalla.

4. Modifica el ejercicio anterior para que la funcionalidad se logre a través de las funciones de la API WebStorage.