PRIMERA EVALUACIÓN

NOTA: No está permitida la modificación de los archivos proporcionados (HTML, CSS y JS), salvo en el HTML para comentar/decomentar las referencias a archivos JS correspondientes a pruebas unitarias.

1. Queremos implementar un script que nos permita manejar la gestión de las plazas de una empresa de autobuses. El script manejará las estructuras de datos descritas a continuación.

La clase `Autobus` nos permite gestionar las plazas de un autobús. Se definirá en el archivo `autobus.js`. Si se desea realizar una prueba unitaria de la clase (no obligatorio pero recomendable), se realizará en el archivo `autobus_prueba.js`

Clase **Autobus**			[Definida mediante prototipo]
Propiedades			
Nombre	Tipo	Visibilidad	Descripción
matricula	cadena	Pública	Matrícula del autobús
origen	cadena	Pública	Origen del viaje
destino	cadena	Pública	Destino del viaje
fecha	Date	Pública	Fecha del viaje
precioBase	entero	**Privada**	Precio base para los billetes.
plazas	Billete[]	**Privada**	Plazas del autobús.
Métodos			
Nombre	Parámetros	Devuelve	Descripción
Autobus(matricula, origen, destino, fecha, numeroFilas, precioBase);	matricula:cadena origen:cadena destino:cadena fecha:Date, numeroFilas:entero, precio	-	Constructor. Se inicializa el array de plazas (ver estructura del mismo más adelante).
generaLocalizador()	-	Cadena	Genera un localizador de manera aleatoria. Un localizador tiene un formato LLLNN, siendo L una letra y N un número[1]
generaPrecio()	-	real	Genera el precio para un billete. Para calcular el precio de un billete sumaremos al precio base 0.2 por cada billete que se ha reservado (es decir, se suma 0.2 por cada posición ocupada en el array de plazas).
reservaPlaza(billete)	billete:Billete	entero	Inserta el billete en la primera plaza libre. Para ello habrá que tener en cuenta si el billete iba con preferencia de pasillo o ventana. Se devolverá un código de reserva de entre los definidos en la clase **CODIGOS** del archivo `varios.js` para indicar: - Reserva correcta - Reserva realizada pero no se respeta preferencia de ventanilla / pasillo - Reserva no realizada.
getPlaza(localizador)	localizador:cadena	entero[2]	A partir del localizador, obtiene el asiento asignado (fila/asiento), que se devolverá en un array de dos posiciones (la primera contiene la fila y la segunda el asiento).

[1] Para generar una letra de manera aleatoria se puede utilizar el array LETRAS definido en el archivo `varios.js`.

getPorcentajeOcupacion()	-	entero	Devuelve el porcentaje de ocupación (redondeado a entero) del autobús.
toHTML()	-	cadena	Devuelve una representación de las plazas como tabla HTML. Para cada plaza se mostrará el nombre y apellidos del pasajero o * si está vacía.

Para guardar las plazas asignadas del autobús, se mantendrá una estructura de datos como la descrita a continuación. Supongamos que el autobús se ha inicializado a 6 filas (dato proporcionado en el constructor). Todas las filas de todos los autobuses tienen 4 plazas, luego la propiedad plazas contendrá una tabla de 7 filas x 4 columnas donde la primera fila contendrá la letra asignada al asiento (A, B,C,D).

	0	1	2	3
0	A	B	C	D
1	{Billete}	{Billete}	{Billete}	{Billete}
2	{Billete}	{Billete}	*	{Billete}
3	{Billete}	{Billete}	{Billete}	*
4	{Billete}	*	*	{Billete}
5	{Billete}	{Billete}	{Billete}	{Billete}
6	{Billete}	{Billete}	{Billete}	{Bilete}

Inicialmente todas las celdas contendrán la cadena *, y a medida que se vayan asignando plazas se irán insertando objetos de tipo `Billete`. Las letras A y D se corresponden con ventanilla, mientras que las B y C se corresponden con pasillo.

La clase `Billete` nos permite representar datos de un billete adquirido por un pasajero. Recordemos que al definirse mediante objetos literales no es necesario definir previamente la clase, sino que los objetos se irán creando a definiendo que se necesitan.

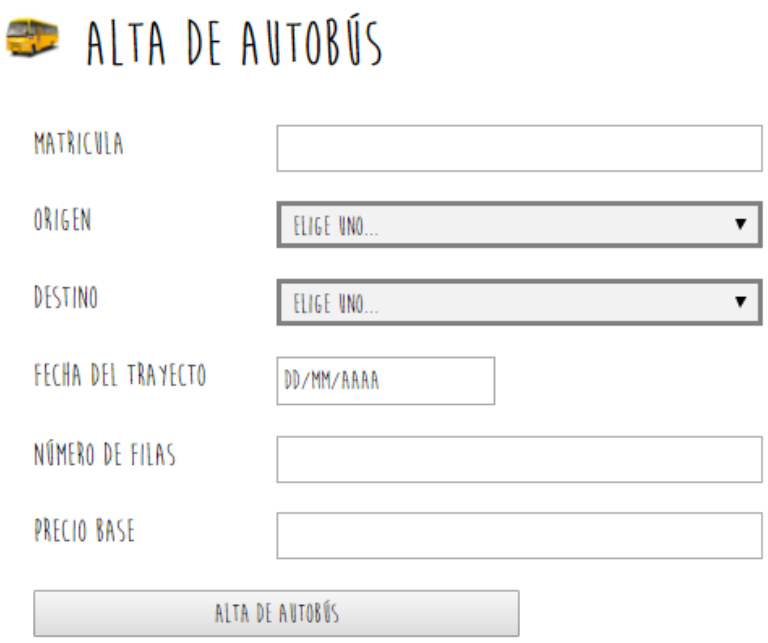
Clase **Billete**			[Definida mediante objetos literales]
Propiedades			
Nombre	Tipo	Visibilidad	Descripción
nombre	cadena	Pública	Nombre del pasajero
apellidos	cadena	Pública	Apellidos del pasajero
dni	cadena	Pública	DNI del pasajero
precio	real	Pública	Precio del billete
localizador	cadena	Pública	Localizador del billete (6 letras).
ventanilla	booleano	Pública	Determina si el billete es preferiblemente de ventanilla (valor true) o de pasillo (valor false).

La clase Flota nos permite almacenar información acerca de un conjunto de autobuses. Se proporciona la clase ya implementada en el archivo `flota.js` (y el archivo de prueba en **`flota_prueba.js`**)

Clase **Flota**			[Definida mediante prototipo]
Propiedades			
Nombre	Tipo	Visibilidad	Descripción
autobuses	Autobus[]	Privada	Listado de autobuses de la flota.
Métodos			
Nombre	Parámetros	Devuelve	Descripción
Flota()	-	-	Constructor sin parámetros

addAutobus(autobus)	autobus:Autobus	boolean	Añade el autobús al listado de autobuses. Devuelve verdadero si el autobús se ha podido insertar correctamente y falso si no se puede insertar, por ya existir un autobús con la misma matrícula.
getAutobus(origen,destino)	origen:cadena destino:cadena	Autobus	Devuelve el primer autobús que opera entre origen y destino cuyo porcentaje de ocupación sea menor al 100%. Si no se encuentra ningún autobús con dichas características devuelve `null`.
getAutobusPosicion(posicion)	posicion:entero	Autobus	Devuelve el autobús que ocupa la posición **`posicion`**, y **`null`** si no existe autobús en dicha posición
getNumeroAutobuses()	-	entero	Devuelve el número de autobuses

Para dar de alta nuevos autobuses se usará el siguiente formulario, proporcionado en **`autobus.html`**. Asociado al mismo tenemos el archivo **`autobus-vista.js`**, que contiene una instancia de **`Flota`** y los manejadores de eventos correspondientes (siguiendo el modelo W3C):



Al pulsar el botón *Alta de autobús* se validarán los siguientes aspectos:

- Todos los campos son obligatorios (matricula, origen, destino, fecha, filas y precio base). En caso de que alguno de los campos no se haya introducido, en la capa **`mensaje`** se mostrará un mensaje de error con el formato de error correspondiente:

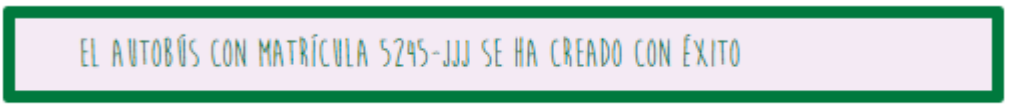
- La matrícula tendrá el formato NNNN-LLL, siendo N dígito y L letra. En caso de no seguir dicho formato, se mostrará un mensaje de error indicativo:

- El campo número de filas deberá ser un entero cuyo valor puede estar entre 1 y 14. En caso de no ser así, se mostrará un mensaje de error indicativo al igual que en los casos anteriores.

- El campo precio base deberá ser un entero. En caso de no ser así, se mostrará un mensaje de error indicativo al igual que en los casos anteriores.

- NOTA: No es necesario validar el formato del campo de fecha.

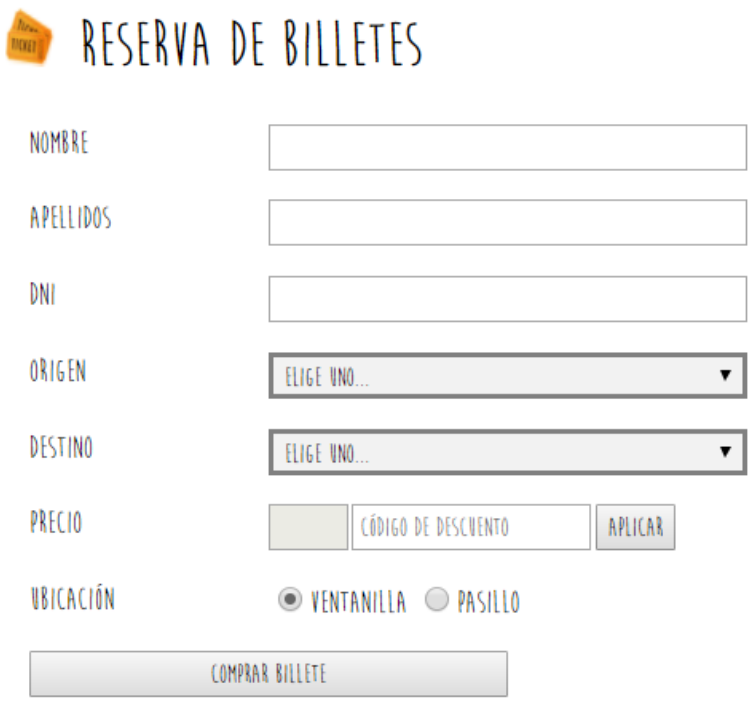
Si todas las validaciones son correctas, se instanciará un nuevo objeto de tipo **`Autobus`** que se añadirá a la flota, y se mostrará un mensaje de información indicativo con el formato adecuado:



Si no es posible guardar el autobús por ya existir otro con la misma matrícula, se mostrará un mensaje de información especial con el formato adecuado:

Por otra parte, al cerrar la página, se guardará en localStorage la información de la flota, usando una variable cuya clave es 'flota' y su valor es la instancia de `Flota` convertida a JSON.

Para dar de alta nuevos billetes se usará el siguiente formulario, proporcionado en **`billetes.html`**. Asociado al mismo tenemos el archivo **`billetes-vista.js`**, que contiene una instancia de **`Flota`** y los manejadores de eventos correspondientes (siguiendo el modelo W3C):

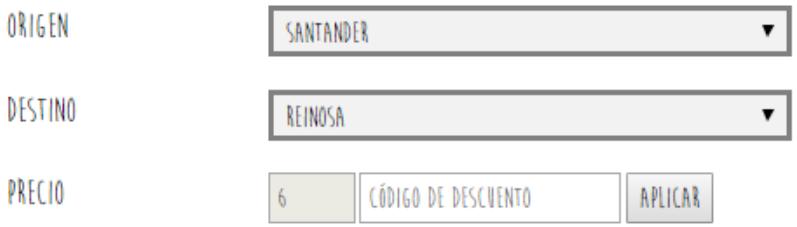


Al cargar la página, se recuperará desde JSON la instancia de **`Flota`** guardada (en caso de existir) y se utilizará para recuperar los datos de los autobuses guardados previamente.

NOTA: En caso de que el alumno no sepa o no consiga hacer correctamente este paso, puede introducir de manera manual varias instancias de `Autobus` en la instancia de **`Flota`**. Se puede coger el ejemplo de `flota_prueba.js`.

Por otra parte, se tendrán en cuenta los siguientes aspectos:

- Todos los campos (nombre, apellidos, DNI, origen, destino y ubicación) son obligatorios. En caso de no introducir un campo se mostrará un mensaje de error indicativo, al igual que en el formulario anterior.
- El DNI debe seguir un formato NNNNNNNN-L. En caso de ser incorrecto se mostrará un mensaje de error indicativo, al igual que en el formulario anterior.
- Cuando se hayan escogido tanto el origen como el destino (el evento onChange de `select` se desencadena cuando ha cambiado su valor) se comprobará si hay un autobús en la flota que opere entre origen y destino con plazas:
    - En caso negativo se mostrará un mensaje de error indicando que el trayecto no se cubre, usando el mismo formato que en el formulario anterior.
    - En caso afirmativo, se obtendrá de dicho autobús el precio del billete y se mostrará en la caja **precio**.



- Se puede introducir un código de descuento en la caja de texto código-descuento. Al pulsar el botón *Aplicar* se comprobará si el código está entre las cadenas definidas en el array CODIGOS_DESCUENTO, definidas en varios.js:
    - En caso afirmativo, se aplicará un descuento del 10% al precio de la caja precio (de haberlo). Se mostrará un mensaje de tipo **especial** informando de que se ha aplicado un descuento del 10% al precio y se actualizará la caja con el precio.
    - En caso negativo, se mostrará un mensaje de error indicativo con el mismo formato que en el ejercicio anterior, indicando que no existe el código de descuento introducido.

Si las validaciones son correctas, al pulsar *Comprar billete* se realizarán las siguientes acciones:

- Se creará un nuevo objeto de tipo `Billete` con los datos introducidos.
- Se obtendrá un localizador de manera aleatoria del autobús donde operamos, que se asignará al billete.
- Se reservará plaza en el autobús y a continuación:
    - Si se ha podido reservar plaza:
        - Se mostrará un mensaje de **informacion** con el localizador, la fila y asiento asignados. En caso de que no se haya respetado la preferencia de pasillo/ventanilla habrá que informar.
        - Se mostrará en la capa con id **plazas** una tabla que representará las plazas libres/ocupadas.
    - Si no se ha podido reservar la plaza se mostrará un mensaje de error informando de que el autobús está lleno actualmente.

CRITERIOS DE EVALUACION

CRITERIO DE EVALUACIÓN		VALORACIÓN MÁXIMA
Clase **Autobus** (4 puntos)	Constructor e inicialización del número de plazas correcto.	1 punto
	Método `generaLocalizador()` correcto.	0,5 puntos
	Método `generaPrecio()` correcto.	0,5 puntos
	Método `reservaPlaza()` correcto	1 punto
	Método `getPorcentajeOcupacion()` correcto	0,5 puntos
	Método `getPlaza()` correcto	0,5 puntos
Formulario de alta de autobús (2,5 puntos)	Validación de campos obligatorios	0,5 puntos
	Validación de matrícula	0,5 puntos
	Validación de número de filas y precio	0,5 puntos
	El autobús se añade a la flota correctamente La matrícula repetida se trata correctamente	0,5 puntos
	La flota se guarda en localStorage correctamente	0,5 puntos
Formulario de compra de billetes (3,5 puntos)	La instancia de flota se recupera correctamente desde localStorage	1 punto
	Validación de campos obligatorios y DNI	0,5 puntos
	El precio se muestra correctamente al cambiar la ruta	0,5 puntos
	El código de descuento se aplica correctamente.	0,5 puntos
	El billete se compra correctamente, mostrando la información y la tabla correctamente.	1 punto