

NOMBRE ALUMNO/A _____

Curso 2018/2019	PORTADA Modelo de Objetos del Documento	CALIFICACIÓN
Evaluación: 1º		
Fecha: 28/12/18		
	Grupo: 2º DAW	

MÓDULO	UNIDAD DE TRABAJO	ACTIVIDAD
DEW	UT6	A15

CRITERIO DE EVALUACIÓN		VALORACIÓN MÁXIMA
Modelo (3 puntos)	Se implementan correctamente los métodos pedidos de Edificio	0,5 puntos
	Se implementa correctamente el método <code>getEdificio()</code> en Inmobiliaria	0,5 puntos
	Se realiza correctamente la importación de los edificios en Inmobiliaria	1 punto
	Se realiza correctamente la importación de los propietarios en Inmobiliaria	1 punto
Manejadores de eventos y tratamiento del DOM (7 puntos)	Los edificios y los pisos se generan correctamente.	0,75 puntos
	La distribución de pisos en columnas es correcta	0,5 puntos
	Los iconos se muestran correctamente en los pisos	0,75 puntos
	Los pisos vacíos se muestran correctamente	0,5 puntos
	Eliminación de propietarios correcta	1 punto
	Modificación de propietarios correcta	1,5 puntos
	Creación de nuevos usuarios correcta.	1,5 puntos

Dentro de cada valoración se incluye: **Estrategias de programación, eficiencia, elegancia y documentación del código.** Por lo que, todas las habilidades nombradas anteriormente, también, **serán valoradas.**

- Queremos hacer una aplicación en JavaScript para gestionar edificios con la información de la situación del edificio y de los propietarios de cada piso. Para ello vamos a utilizar las siguientes clases (en sombreado se indican los métodos que es necesario implementar):

Clase Propietario [Definida mediante objeto literal]			
Propiedades			
Nombre	Tipo	Visibilidad	Descripción
nombre	cadena	pública	Nombre y apellidos del propietario
genero	booleano	Pública	Género del propietario
tamFamilia	entero	Pública	Número de integrantes de la unidad familiar

Clase Edificio [Definida mediante prototipos]			
Propiedades			
Nombre	Tipo	Visibilidad	Descripción

calle	cadena	privada	Calle del edificio
numero	entero	Privada	Número del edificio
codigoPostal	cadena	Privada	Código postal
plantas	Propietario[][]	Privada	Plantas del edificio. Dentro de cada planta tendremos un número de puertas y para cada puerta almacenaremos el propietario , representado mediante una instancia de Propietario . Se implementará esta propiedad utilizando un array bidimensional.
Métodos			
Nombre	Parámetros	Devuelve	Descripción
agregaPlantasYPuertas	nPlantas:entero nPuertas:entero	-	Recibe el número de plantas que queremos crear en el edificio y el número de puertas por planta. Las plantas se añadirán a las ya creadas en el edificio. Todas las puertas se inicializan a null .
modificaNumero	numero:entero	-	Actualiza el número del edificio
modificaCalle	calle:cadena	-	Modifica el nombre de la calle.
modificaCP	cp:cadena	-	Modifica el CP
getCalle	-	cadena	Devuelve el nombre de la calle
getNumero	-	entero	Devuelve el número del edificio
getCP	-	cadena	Devuelve el CP
agregaPropietario	propietario:cadena planta:entero puerta:entero	-	Asigna el propietario al piso identificado por planta y puerta .
getNumeroPlantas	-	entero	Devuelve el número de plantas del edificio
getNumeroPuertas	planta:entero	entero	Devuelve el número de puertas de la planta planta
getPropietario	planta:entero puerta:entero	Propietario	Devuelve el propietario del piso identificado por planta y puerta .

Clase Inmobiliaria [Definida mediante prototipos]			
Propiedades			
Nombre	Tipo	Visibilidad	Descripción
edificios	Edificio[][]	Privada	Edificios de la inmobiliaria.
Métodos			
Nombre	Parámetros	Devuelve	Descripción
Inmobiliaria	-	-	Constructor sin parámetros
addEdificio	edificio: Edificio	-	Inserta un edificio en el array.
getEdificio	calle: cadena numero:entero	Edificio	Devuelve el edificio con la calle y número proporcionados, o null si no existe un edificio con los datos proporcionados.
9cargaDatos	-	-	Carga los datos del fichero importar.js en el array

El fichero **importar.js** contiene dos arrays:

- El array **edificiosImportar** contiene datos de varios edificios que tenemos que importar. Cada edificio que tenemos que importar se representa con un objeto literal que contiene los datos siguientes:
 - calle**: cadena. Calle del edificio.
 - numero**: entero. Número del edificio.

- **cp:** cadena. Código postal del edificio.
- **plantas:** array de enteros que representa las plantas que queremos crear en el edificio. Contiene tantas posiciones como plantas queremos en el edificio. La posición *i* del array contiene el número de puertas de la planta *i*.

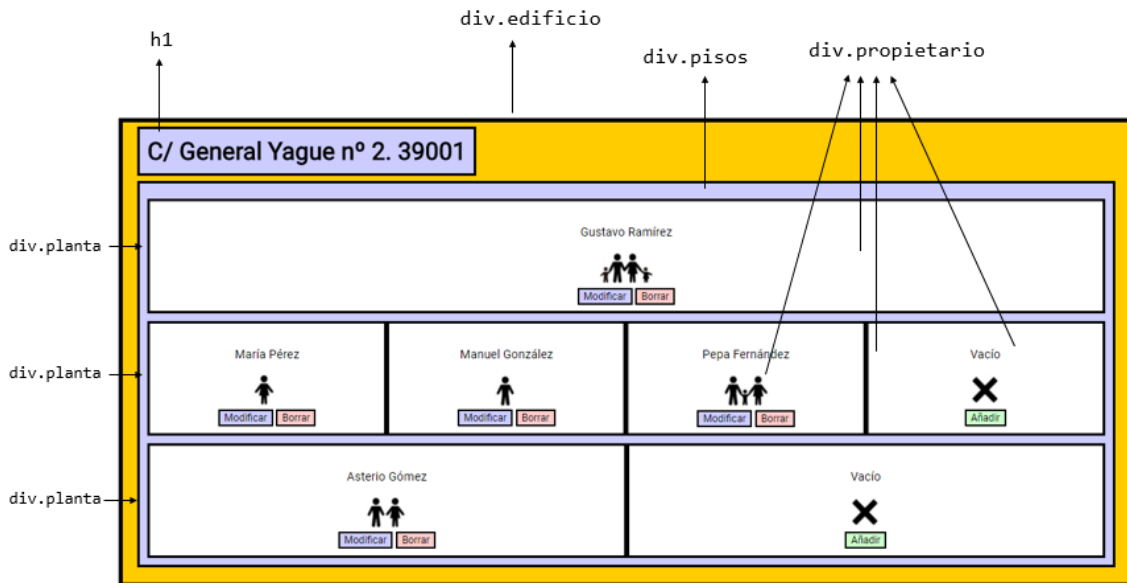
Por ejemplo, si **plantas** contiene [2,2,1,1], se generará un edificio con 4 plantas como el siguiente (las dos plantas superiores contienen dos puertas, y las dos inferiores contienen una puerta). Recordemos que las puertas se inicializan a **null**:

null	null
null	null
null	
null	

- El array **inquilinosImportar** contiene datos de varios inquilinos que tenemos que importar. Cada inquilino que tenemos que importar se representa con un objeto literal que contiene los datos siguientes:
 - **calle:** cadena. Calle del Edificio donde importaremos al inquilino
 - **numero:** entero. Número del Edificio donde importaremos al inquilino
 - **piso:** entero. Piso del Edificio donde importaremos al inquilino
 - **puerta:** entero. Puerta del piso anterior donde importaremos al inquilino
 - **nombre:** cadena. Nombre del inquilino.
 - **genero:** cadena. Género del inquilino.
 - **miembros:** entero. Número de miembros de la unidad familiar.

Desde el cuerpo del archivo **ejercicio1.js** realizaremos las siguientes tareas:

- Declaramos e instanciamos una variable global de tipo **Inmobiliaria**, en la cual cargaremos los datos a importar (llamando a **cargaDatos()**). **Está prohibido el uso de más variables globales.**
- A partir de la información de inmobiliaria, se generará de manera dinámica, **para edificio de la inmobiliaria**, generaremos una estructura como la siguiente:
 - Un **div** de clase **edificio**, que contiene:
 - Un **h1** con los datos del edificio.
 - Un **div** de clase **pisos** que contiene:
 - Un **div** de clase **planta** para cada planta del edificio.
 - Un **div** de clase **propietario** para cada propietario.



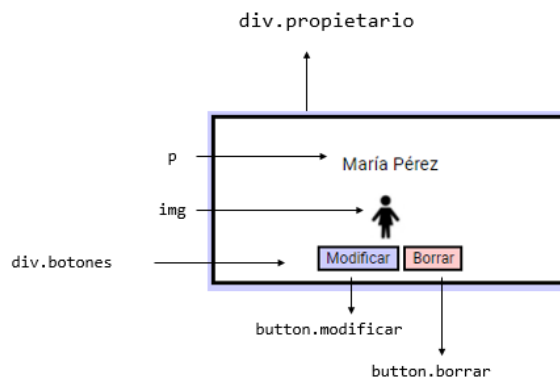
A cada div de clase propietario se le asignará una clase adicional que representa el número de columnas que ocupa el piso en la planta, y puede ser:

- **col-4** si es el único piso de la planta.
- **col-2** si hay 2 pisos en la planta.
- **col-4** si hay 4 pisos en la planta.

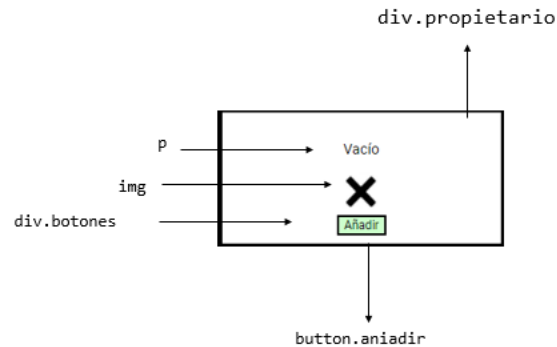
Todos los pisos de una planta tendrán el mismo valor para esta clase. En la captura anterior, los pisos de la planta inferior tienen **col-2**, los de la planta intermedia tienen **col-4** y el de la planta superior tiene **col-1**.

En caso de que exista el propietario, para cada capa propietario se generará la siguiente información,

- Un párrafo con el nombre del propietario.
- Una imagen que dependerá del género del propietario y/o el número de miembros de la unidad familiar, que puede ser una de las siguientes imágenes de la carpeta img:
 - **hombre.jpg** (familia de 1 miembro, género masculino).
 - **mujer.jpg** (familia de 1 miembro, género femenino).
 - **pareja.jpg** (familia de 2 miembros).
 - **familia-1.jpg** (familia de 3 miembros).
 - **familia-2.jpg** (familia de 4 ó más miembros).
- Un **div** de clase **botones** que contiene:
 - Un **button** de clase **modificar**
 - Un **button** de clase **borrar**.



En caso de que exista el propietario, para cada capa propietario se generará una estructura similar, con la salvedad de que la imagen será **vacio.jpg** y en lugar de dos botones se generará un único botón de clase **aniadir**.



En cuanto al comportamiento de los botones, será el siguiente:

- Al pulsar el botón *Borrar* en un propietario, automáticamente, y sin previa confirmación, se eliminará el propietario correspondiente, pasando a estar dicha puerta vacía y mostrando el piso como vacío (ver punto anterior).
- Al pulsar el botón *Añadir* en un propietario (vacío), se mostrará el formulario con id formulario (inicialmente oculto), modificando su propiedad de estilo **display** (pasará a tener el valor 'block'). En dicho formulario:
 - El botón *Modificar* estará deshabilitado.
 - Los valores de planta y puerta (atributo value) se corresponderán con los de la planta y puerta donde hemos pulsado.
 - Al pulsar el botón *Cerrar* se cerrará el formulario (display='none').
 - Al pulsar el botón *Añadir* se creará un nuevo propietario y se añadirá a la posición correspondiente. **No es necesario realizar validaciones de campos existentes ni coherencia de tipos en el formulario** (suponemos que todos los datos se introducen correctamente). Una vez añadido el propietario se cerrará el formulario.

- Al pulsar el botón *Modificar* en un propietario, se procederá de manera similar al caso anterior, con la diferencia de que:
 - El botón deshabilitado del formulario será el de *Añadir*
 - Los datos del propietario que hemos pulsado se mostrarán en el formulario.

Gestión de propietarios

Nombre:

Apellidos

Unidad familiar

Género ☒ Hombre ☐ Mujer

Planta

Puerta

Añadir

Modificar

Cerrar








Se proporcionan los siguientes archivos:

- **varios.js**. Definición de la clase enumerada **TIPO_FAMILIA** para distinguir los tipos de unidad familiar. En este archivo puedes incluir todas las funciones auxiliares que consideres necesarias.
- **edificio.js**. Definición de la clase **Edificio**.
- **importar.js**. Contiene los datos que importaremos desde la clase **Inmobiliaria**.
- **inmobiliaria.js**. Definición de la clase **Inmobiliaria**.
- **ejercicio1.js**. Definición de manejadores de eventos y funciones de acceso al DOM, además de manejar una variable global que consiste en una instancia de la clase **Inmobiliaria**.

NOTA: Con los datos proporcionados, la interfaz debería tener el siguiente aspecto:


Gestión de edificios

C/ General Yague nº 2. 39001

<p>Gustavo Ramírez</p>  <p>Modificar Borrar</p>			
<p>Maria Pérez</p>  <p>Modificar Borrar</p>	<p>Manuel González</p>  <p>Modificar Borrar</p>	<p>Pepa Fernández</p>  <p>Modificar Borrar</p>	<p>Vacio</p>  <p>Añadir</p>
<p>Asterio Gómez</p>  <p>Modificar Borrar</p>		<p>Vacio</p>  <p>Añadir</p>	

C/ Primero de Mayo nº 1. 39011


Eleuterio Gómez



Modificar

Borrar

Vacio



Añadir

C/ Paseo de Pereda nº 5. 39002	
<div>Vacio</div> <div>✕</div> <div>Añadir</div>	
<div>Vacio</div> <div>✕</div> <div>Añadir</div>	
<div>Vacio</div> <div>✕</div> <div>Añadir</div>	<div>Vacio</div> <div>✕</div> <div>Añadir</div>
<div>Vacio</div> <div>✕</div> <div>Añadir</div>	<div>Vacio</div> <div>✕</div> <div>Añadir</div>