ACTIVIDAD 8

EL EDIFICIO (PARTE 2)

Queremos hacer una aplicación en JavaScript para gestionar los propietarios de un edificio. Para ello vamos a utilizar la siguiente estructura de datos creada en la parte 1 de este ejercicio (realizada en la unidad anterior):

Clase `Edificio`

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. Inicialmente está vacío

Métodos

Nombre	Parámetros	Devuelve	Descripción
Getters y setters de las propiedades			
`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`**.

`agregaPropietario`	`prop:Propietario`
	`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`**.

Para almacenar la información relativa a los propietarios (nombre, género y miembros de la familia) usaremos **objetos literales** por lo que no será necesario declarar la clase (descarta el fichero **Propietario.js** en esta versión).

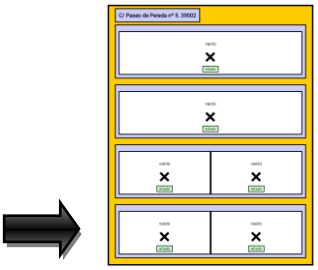
En el archivo **`ejercicioEdificio.js`** tendremos todas las funciones que se relacionan con el DOM, y realizaremos las siguientes tareas:

- Crearemos un edificio que almacenaremos en una variable global, proporcionando los datos sobre la calle, número y cp. Sobre este edificio crearemos varias plantas, por ejemplo, si queremos un edificio con 4 plantas, 2 de ellas con 1 puerta y las otras 2 con 2 puertas tendríamos algo así en el array Puertas de la clase Edificio (las puertas se inicializan a `null`:

`null`	`null`
`null`	`null`
`null`	
`null`	

Ten en cuenta que al mostrar con console.table esta estructura, las plantas más altas se muestran más abajo:

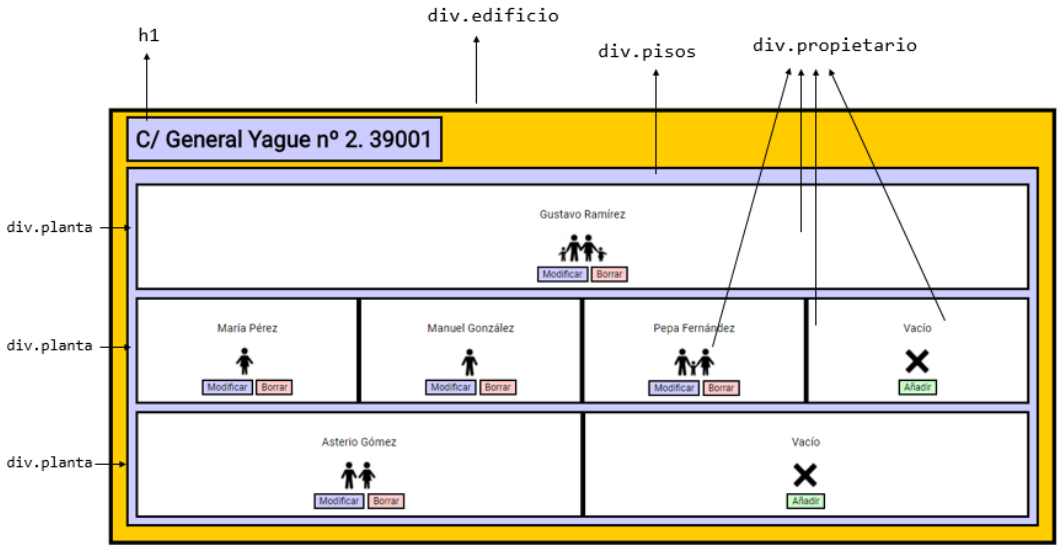
(index)	0	1
0	null	null
1	null	null
2	null	
3	null	



- Para tener unos datos iniciales en el edificio, usaremos un array **`inquilinosImportar`** con los datos de varios inquilinos:
  - **`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.

  Será necesario procesar este array usando la función map.

- En la interfaz, 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.



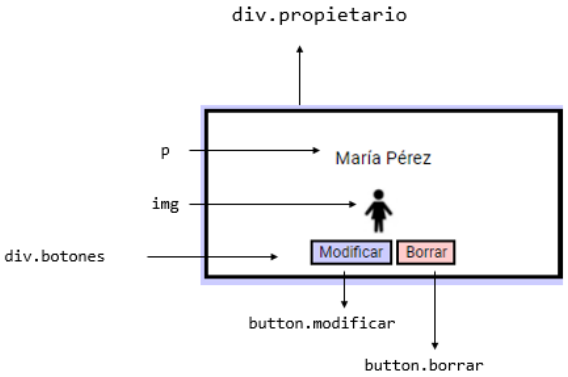
A cada div de clase propietario se le asignará una clase adicional que representa el número de columnas que ocupa el piso en la planta, y puede ser:

- `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.

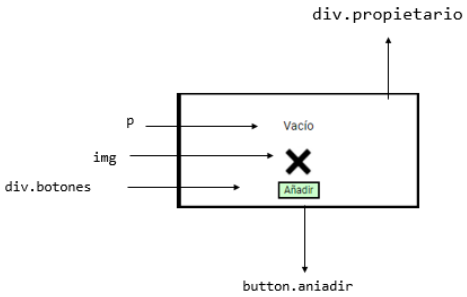
Todos los pisos de una planta tendrán el mismo valor para esta clase. En la captura anterior, los pisos de la planta inferior tienen `col-2`, los de la planta intermedia tienen `col-4` y el de la planta superior tiene `col-1`.

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 comprobará que se hayan introducido los campos obligatorios (nombre, apellidos y unidad familiar), si es así se creará un nuevo propietario y se añadirá a la posición correspondiente. 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.

RETO OPCIONAL

Añade al ejercicio la posibilidad de gestionar más de un edificio, para ello deberás usar la siguiente clase:

Clase **`Inmobiliaria`**			
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.

La interfaz entonces tendría un div padre "inmobiliaria" que contendría un div para cada edificio de la siguiente manera: