UD2E6 POO – Gestión red de bibliotecas

Vamos a desarrollar una aplicación web dinámica para la gestión de bibliotecas utilizando JavaScript orientado a objetos (POO), módulos ES6 y un enfoque de Single Page Application (SPA). El objetivo es trabajar con clases, módulos, generación dinámica de HTML y manejo de datos.

El sistema que vamos a desarrollar se dividirá en varias clases: **Libro**, **Autor**, y **Biblioteca**, cada una en su propio módulo. Además, contaremos con un módulo central llamado **gestorBiblioteca.js** que actuará como punto de entrada para la aplicación.

La aplicación se cargará inicialmente desde un único archivo HTML (index.html) que contendrá el mínimo código necesario, y todo el contenido será gestionado dinámicamente desde JavaScript. Los datos iniciales (libros, autores y bibliotecas) estarán definidos en un archivo aparte (**datos.js**), permitiendo que la aplicación cargue su estado inicial desde ahí.

Clase Libro

Propiedades: libroId, titulo, ISBN, autorId, bibliotecaId, prestamos (lista con la fecha de préstamo y fecha de devolución), estaDisponible (calculada).

Métodos:

- **generarHTMLCreacion():** Formulario para crear un nuevo libro.
- **generarHTMLPropiedades():** Detalles del libro. Incluye opciones de menú para: editar, borrar, listar prestamos, crear préstamo, devolver préstamo, listar préstamos.
- **generarHTMLEdicion():** Formulario para editar un libro.
- **generarHTMLListadoPrestamos():** Listado de préstamos.

NOTA: te doy libertad para que pienses como gestionar la solicitud de préstamo y devolución.

Clase Autor

Propiedades: autorId, nombre, nacionalidad, biografía, libros (lista con los títulos de los libros publicados (UN TÍTULO NO ES UN ID)).

Métodos:

- **generarHTMLCreacion():** Formulario para crear un nuevo autor.
- **generarHTMLPropiedades():** Detalles del autor y una tabla con los libros publicados. Incluye opciones de menú para: añadir y eliminar libros.
- **generarHTMLEdicion():** Formulario para editar un autor.

NOTA: te doy libertad para que pienses como gestionar la añadir y eliminar libros.

Clase Biblioteca

Propiedades: bibliotecaId, nombre, ubicación, libros (lista de objetos libro asignados a la biblioteca, calculada a partir de libros).

Métodos:

- **generarHTMLCreacion():** Formulario para crear una nueva biblioteca.
- **generarHTMLEdicion():** Formulario para editar una biblioteca.

Módulo gestorBiblioteca.js

Crea una función autoinvocada que devuelve un objeto **$biblio** que contendrá todos los métodos públicos para gestionar la aplicación, y de manera privada el código auxiliar.

Métodos expuestos por $biblio:

- **generarHTMLListadoAutores():** Listado con los autores. Incluye opciones de menú para: crear, ver, editar, borrar.
- **generarHTMLListadoBibliotecas():** Listado con las bibliotecas. Incluye opciones de menú para: crear, ver, editar, borrar.
- **generarHTMLListadoLibros():** Listado con los libros. Incluye opciones de menú para: crear, ver, editar, borrar.
- **buscarLibrosPorTitulo(parámetro/s), buscarLibrosPorAutor(parámetro/s):** Devuelven la colección de libros o autores que cumplen con los criterios de filtrado.
- **generarHTMLResultadoBuscador(parámetro/s):** Listado con los resultados de una búsqueda, el formato varía según el tipo de búsqueda.
- **buscarLibro(libroId), buscarAutor(autorId), buscarBiblioteca(bibliotecaId):** Buscan en las colecciones el objeto correspondiente según su ID.
- **crearLibro(libro), crearAutor(autor), crearBiblioteca(biblioteca):** Añade en las colecciones el objeto correspondiente. Debes gestionar con un incremental por clase el valor del ID.
- **borrarLibro(libroId), borrarAutor(autorId), borrarBiblioteca(bibliotecaId):** Elimina en las colecciones el objeto correspondiente según su ID.
- **devolverPrestamo(libro), crearPrestamo(libro):** Devuelve el libro si actualmente está prestado. Crea un nuevo préstamo si el libro esta disponible.

NOTA: no necesitas métodos para editar, ya que actualizas el objeto con el ID indicado.

Método privados:

- Gestión de datos.js
- Colecciones de datos: libros, autores, bibliotecas

Página “index.html”

La página tendrá un menú con las opciones para: listado de autores, listado de libros, listado de bibliotecas.

Un formulario buscador con la posibilidad de filtrar por libros o por autores.

Un div con el id=”app” en el que insertarás dinámicamente los formularios según necesitados. **RECUERDA**, cada vez que cargues un formulario dinámicamente tendrás que gestionar los eventos de los elementos HTML incluidos en este div.

- ⇨ Vas a necesitar código auxiliar para mapear los eventos y para cambiar el contenido de “app” dinámicamente.
- ⇨ document.querySelector y document.querySelectorAll son tus amigos, no le tengas miedo a los selectores CSS.

Recomendaciones.

Lee todo y hasta que no lo hayas leído no hagas nada.

Haz un esquema del modelo de datos.

Haz unos dibujos con lo que vas a construir.

Crea la estructura del proyecto, y que compile.

Crea páginas estáticas de ejemplo con la estructura de datos a generar dinámicamente. Piensa que necesitarías para poner eventos a elementos. Luego replica esa estructura en javascript.

- ⇨ CONSEJO: define un formato de clases para identificar a los elementos y luego asignar fácilmente los eventos, algo del estilo “biblio-libro-editar”, “biblio-autor-listar”, etc...
- ⇨ La propiedad HTML “data” puede ser de utilidad.