PRÁCTICA 3.04 Mi primera aplicación en REACT

Normas de entrega

- En cuanto al **código**:
  - en la **presentación interna**, importan los **comentarios**, la claridad del código, la significación de los nombres elegidos; todo esto debe permitir considerar al programa como **autodocumentado**. No será necesario explicar que es un `if` un `for` pero sí su funcionalidad. Hay que comentar las cosas destacadas y, sobre todo, las **funciones** y **clases** empleadas. La ausencia de comentarios será penalizada,
  - en la **presentación externa**, importan las leyendas aclaratorias, información en pantalla y avisos de ejecución que favorezcan el uso de la aplicación,
  - si no se especifica lo contrario, la información resultante del programa debe aparecer en la consola del navegador `console.log(información)`,
  - los ejercicios deben realizarse usando ***JavaScript* ES6** y usar el modo estricto (`use strict`) No se podrá utilizar *jQuery* ni cualquier otra biblioteca (si no se especifica lo contrario en el enunciado),
  - para el nombre de **variables**, **constantes** y **funciones** se utilizará *lowerCamelCase,*
  - el nombre de los componentes debe comenzar con letra **mayúscula**.
- En cuanto a la **entrega** de los archivos que componen los ejercicios de `React`:
  - entrega **la práctica** en un sólo proyecto (el nombre a tu discreción),
  - los componentes creados deben estar separados en carpetas (los creados en el Ejercicio1 dentro de una carpeta denominada `Ejercicio1`),
  - el código contendrá ejemplos de ejecución, si procede,
  - comprime la carpeta `src` junto con los ficheros `package.json` y `package-lock.json` en un fichero `ZIP` , y
  - sube a **Aules** el fichero comprimido.

Ejercicio 1 - Aplicación REACT

El objetivo de esta actividad es desarrollar en React la creación de componentes reutilizables y modulares para representar información compleja.

En este caso, vamos a rediseñarán un sistema de componentes para mostrar películas, donde cada componente maneja una parte específica de la información (título, cartelera, elenco, etc.). Partiremos del ejercicio de la unidad anterior.

Emplearemos con **props** para pasar datos entre componentes y utilizaremos **children** para gestionar el contenido dinámico, reforzando así la importancia del flujo de datos en aplicaciones React.

También vamos a apoyarnos en estructurar el proyecto de manera eficiente, usando funciones auxiliares para mejorar la funcionalidad de los componentes.

Paso 1 – Definir y crear una buena estructura de la aplicación

Estructurar las carpetas y fichero que contendrán. La estructura recomendada del proyecto será algo similar a esto donde se agrupan los componentes en una carpeta, las funciones en una biblioteca y los objetos en otra carpeta

```
├── src
│ ├── componentes
│ │ ├── Pelicula.jsx
│ │ ├── Peliculas.jsx
│ │ ├── Interprete.jsx
│ │ ├── Interpretes.jsx
│ ├── bibliotecas
│ │ └── funciones.js
│ ├── objetos
│ │ └── peliculas.json
│ └── App.jsx
└── index.html
```

Paso 2: Crear el archivo `peliculas.json`

Este archivo contendrá los datos de las películas y su elenco. Cada película tendrá su título, cartelera, resumen, y una lista de actores con su respectiva información.

Se adjunta a la práctica fichero de ejemplo, a continuación se incluye pequeño extracto

```
{
    "peliculas": [
      {
        "id": 1,
        "nombre": "El sexto sentido",
        "director": "M. Night Shyamalan",
        "clasificacion": "Drama",
        "recaudacion": "$245 millones",
        "cartelera": "https://www.ecartelera.com/carteles/5400/5409/001.jpg",
        "nota": 8,
        "actores": [
          {
            "nombre": "Marlon Brando",
            "fechaNacimiento": "03/04/1924",
            "biografia": "Marlon Brando fue un influyente actor estadounidense...",
            "imagen": "https://upload.wikimedia.org/Marlon_Brando.png"
          }
          ...
        ],
        "resumen":"El Dr. Malcom Crowe …."
      },
      {
        "id": 2,
        "nombre": "Pulp Fiction",
        "director": "Tarantino",
        "clasificacion": "Acción",
        "recaudacion": "$2.2 mil millones",
        "cartelera": "https://www.ecartelera.com/carteles/200/286/001.jpg",
        "nota": 9,
        "actores": [
          {
            "nombre": "Kate Winslet",
            "fechaNacimiento": "05/10/1975",
            "biografia": "Kate Winslet es una actriz británica...",
            "imagen": "https://upload.wikimedia.org/Kate_Winslet.jpg"
          }...
        ],
        "resumen":"La historia principal cuenta la historia de Vincent Vega …."
      }...
    ]
```

```
}
```

Paso 3: Crear el componente App.jsx

Este componente es el punto de entrada y renderizará el componente principal.

- Importa el archivo JSON con los datos de las películas.
- Importa el componente Peliculas que será el encargado de mostrar la lista de películas.
- Pasa la lista de películas como prop al componente Peliculas. Así, este último podrá acceder a cada película individualmente.

Paso 4: Crear el componente Peliculas.jsx

Este componente se encargará de recorrer el listado de películas y pasar los datos correspondientes a cada película a través de props al componente Pelicula.

- Recibe la lista de películas a través de props.
- Utiliza un método de mapeo (como map()) para recorrer la lista y renderizar un componente Pelicula por cada película en el listado.
- Pasa al componente Pelicula los datos necesarios:
    a. El nombre de la película.
    b. La imagen de la cartelera.
    c. El elenco de actores.
    d. El resumen de la película como children.
- Es recomendable incorporar el operador ternario, para controlar posibles errores. En este caso deberíamos evaluar que si el campo viene vacío o con longitud 0 que imprima "No se han encontrado películas."

Paso 5: Crear el componente Pelicula.jsx

El componente Pelicula recibirá los datos de una película a través de props y mostrará su título, cartelera y resumen. También incluirá una lista de intérpretes (actores/actrices), que será gestionada por el componente Interpretes.

- Recibe a través de props los siguientes parámetros:
    a. nombre: Título de la película.
    b. cartelera: URL de la imagen de la cartelera.
    c. actores: Elenco de la película.
    d. children: El resumen de la película (contenido que se envía dentro de las etiquetas del componente).
- Renderiza la cartelera (imagen) y el resumen de la película.
- Llama al componente Interpretes para mostrar el elenco de la película, pasándole la lista de actores como prop.

Paso 6: Función auxiliar para generar UUIDs

La función generarUuidAleatorio se encargará de generar identificadores únicos para los componentes que no tengan un ID en los datos.

Usa la función en funciones.js para generar identificadores únicos (por ejemplo, utilizando crypto.randomUUID()).

Importa esta función en el componente Interpretes y úsala para generar un identificador único para cada actor dentro de la lista.

```
//Devuelve un UUID aleatorio para identificar elementos.
const generarUuidAleatorio = () => {
  return crypto.randomUUID();
};

export { generarUuidAleatorio };
```

Paso 7: Crear el componente Interpretes.jsx

Este componente se encargará de recibir un listado de intérpretes (actores) y recorrerlos, pasando cada uno al componente Interprete.

- Recibe a través de props una lista de actores.
- Utiliza un método de mapeo para recorrer la lista de actores y renderizar un componente Interprete por cada uno.
- Como no tienes identificador único en los datos de los actores, puedes generar un identificador único usando una función auxiliar o cualquier otro método para crear claves únicas para los componentes. ***Ver nota***

```
return <Interprete key={generarUuidAleatorio()} datos={valor} />;
```

Paso 8: Crear el componente Interprete.jsx

Este componente mostrará la información de cada actor o actriz, incluyendo su nombre, imagen y biografía.

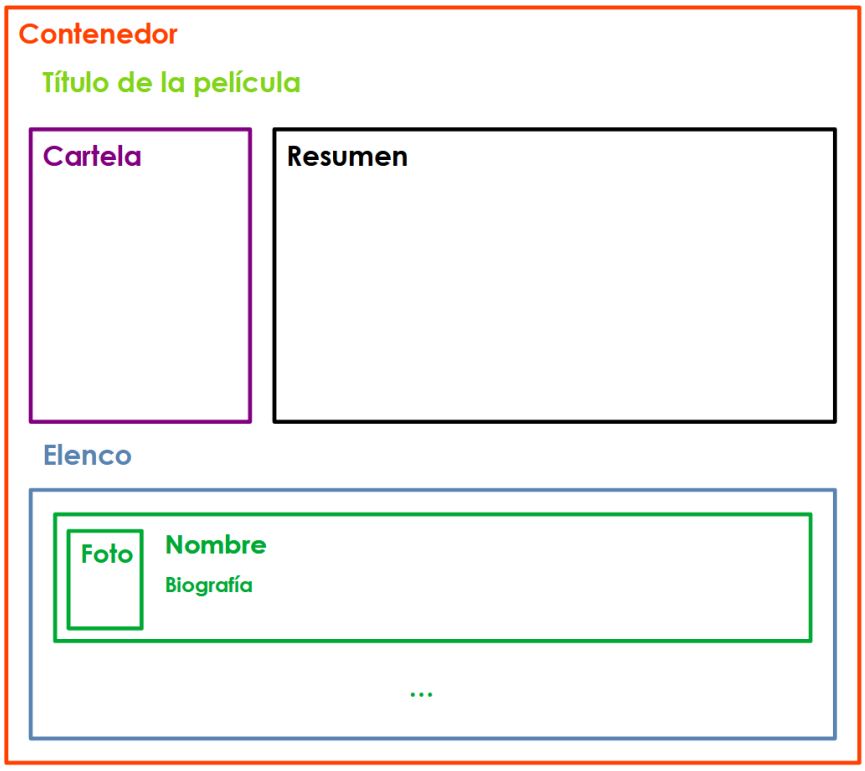
- Recibe a través de props un objeto con los siguientes atributos:
  - nombre: Nombre del actor/actriz.
  - imagen: URL de la imagen del actor/actriz.
  - biografia: Breve descripción o biografía.
- Muestra la imagen del actor, su nombre y biografía en el componente.

Paso 9: Aplicar estilos (opcional)

Puedes definir los estilos CSS en el archivo App.css o en otro archivo de estilos para mejorar la presentación de los componentes, como dar formato al título, la cartelera, o al elenco.

- Define estilos básicos para que el diseño del listado de películas sea claro y esté organizado.
    - Destaca el título de la película.
    - Estiliza la cartelera para que la imagen tenga un tamaño adecuado.
    - Asegúrate de que el elenco se vea correctamente alineado y formateado.
- Añade estilos específicos para los actores y sus imágenes, y asegúrate de que la biografía tenga una presentación legible.

Esto es un ejemplo (**sólo un ejemplo**) de composición para la `<Pelicula>`:



Nota

En React, los identificadores únicos son fundamentales para el correcto funcionamiento de las listas de componentes dinámicos. Cuando renderizamos elementos en un bucle (por ejemplo, una lista de actores en una película), React necesita una manera eficiente de identificar qué elementos han cambiado, añadido o eliminado. Este proceso se conoce como **reconciliación**.