PRÁCTICA 6.04 *Star Wars* 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,
  - para el nombre de **variables**, **constantes** y **funciones** se utilizará *lowerCamelCase,*
  - el nombre de los componentes debe comenzar con letra **mayúscula**.
  - la **componentización** debe ser lógica y estar adaptada a las necesidades de la aplicación,
  - todos los formularios en `React` deben ser del tipo **controlados** (a través del **estado**). En caso contrario debe ser debidamente justificado,
  - toda comunicación con el exterior del `DOM` debe ser realizada con funciones asíncronas que dispongan de control de errores (`try{} catch{}`),
  - se debe implementar el uso de `async/await` para la comunicación asíncrona.
- 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 - Añadiendo contextos en `React`

Parte 1

Modifica la enciclopedia de ***Star Wars*** para que utilice **contextos** destinados a contener la información y funcionalidades que se repitan. Se pueden usar los que creas conveniente (mínimo uno).

Parte 2

Añade, en los datos de cada intérprete, un **listado de las naves y vehículos** (si los hubiera) con los que tenga una relación cada **personaje**. Al pulsar sobre el botón Pilota (que deberás crear y mostrar en la ficha del personaje) se mostrará una ficha con los **datos básicos** de cada nave y vehículo. Si no hay ninguno se debe informar al usuario.

Utiliza la llamada en **conjunto** de `promise` en aquellas solicitudes que deban resolverse como una.

Recuerda que se sugirió una estructura parecida a esta:

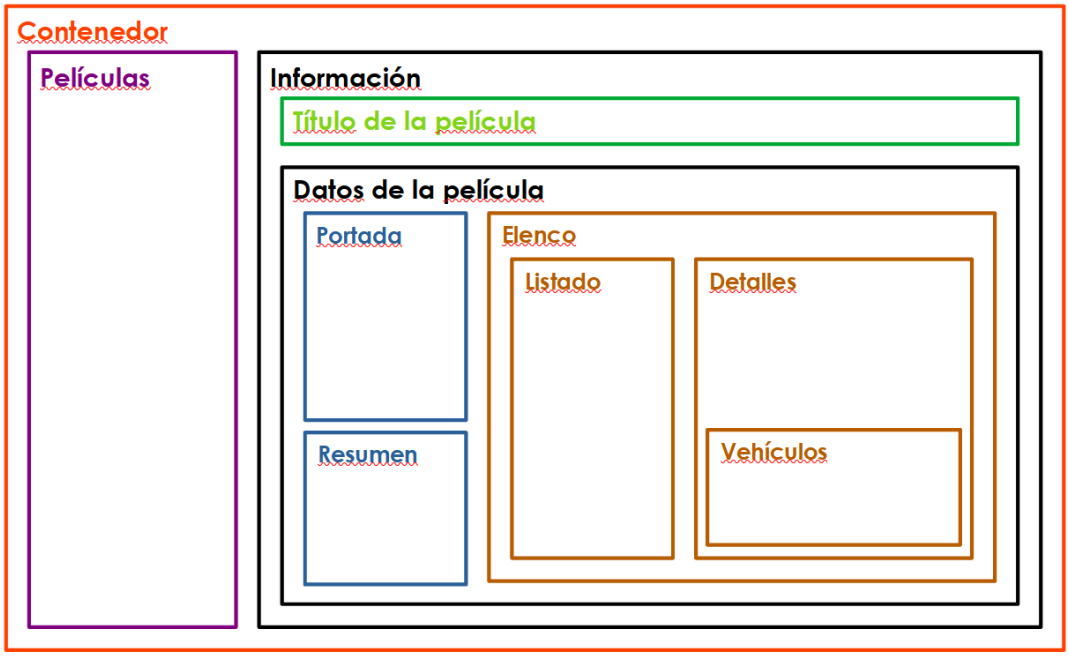

Figura 1: Propuesta (no obligatoria) de organización de la información.