



Especialización en Front End II

Patrones de diseño - Parte I

¡Buenas, buenas! Ahora que hemos completado el repaso acerca de las funcionalidades y el uso de los patrones de diseño, vamos a poner en práctica todo lo que aprendimos.

Para hacerlo, te proponemos comenzar a trabajar en nuestro proyecto integrador: la profesionalización de la **base de datos interplanetaria de Rick**.

- Ejercitación individual 
- Nivel de complejidad: intermedio 

Objetivo

¡**Rick necesita ayuda!** Ha creado rápidamente una aplicación que le permite saber qué planetas visitó cada personaje de la serie, pero la aplicación es un caos de mantener.

En esta práctica vamos a realizar una mejora incremental, creando un custom Hook para manejar el estado de la ventana modal de los personajes, y otro para manejar las traducciones a diferentes idiomas, que además va a utilizar el provider pattern.

Consigna de trabajo

¡Uff, qué desorden es esta aplicación! Si bien se encuentra funcionando —e increíblemente sin errores (al parecer)—, está llena de malas prácticas.

El **primer desafío** es extraer la lógica que se encarga de abrir la ventana modal del personaje seleccionado en un custom Hook, de forma de que sea reutilizable y podamos aprovecharla más adelante. De esta manera, nos vamos familiarizando con la creación de Hooks independientes.

Una vez terminado ese Hook —y con la aplicación todavía funcionando correctamente—, el **segundo desafío** es solucionar el props drilling que hay por toda la aplicación, para permitir

a la misma cambiar de idiomas. Para ello, nuestra **Navbar**, que contiene el componente **LanguageSwitcher**, está manipulando el estado del idioma seleccionado y lo pasa como props a cada componente que lo necesita, dando lugar a miles de props iguales por toda la aplicación. Este es un caso ideal para ser resuelto con lo que vimos en clase acerca de provider pattern y el uso de un custom Hook con la API de Context. Debemos almacenar este **estado** en el **Context** y, utilizando el **useSelector**, podremos acceder a él en cada componente que requiera la traducción.

Debemos tener en cuenta lo explicado durante la clase, ya que será de mucha utilidad para poder realizar este ejercicio.



IMPORTANTE

Te dejamos el código base para que puedas comenzar a trabajar en el siguiente [enlace](#).

Ahora sí, ¡manos a la obra!