Arquitecturas para aplicaciones React y patrones de diseño

Autor:

Kevin Cárdenas.

2024

Índice

1. Patrones de diseño

Los patrones de diseño son soluciones probadas para problemas comunes en el desarrollo de software. En el contexto de React, existen varios patrones de diseño que se utilizan para estructurar y organizar aplicaciones React de manera eficiente y sostenible.

1.1. Reducer

Reducer es un patron de diseño que se utiliza para manejar el estado de una aplicación. En el contexto de React, un reducer es una función que toma el estado actual y una acción, y devuelve un nuevo estado. Los reducers son comunes en aplicaciones basadas en Redux, una biblioteca de gestión de estado para aplicaciones JavaScript o Typescript.

La idea principal detrás de los reducers es que permiten manejar el estado de una manera predecible y escalable. Al dividir el estado en piezas más pequeñas y manejar las actualizaciones a través de acciones, los reducers pueden simplificar la lógica de la aplicación y facilitar el mantenimiento.

Para nuestra Arquitetura tendremos 5 modulos principales:

- **Models** (Modelos): Contiene la definición de los datos, un modelo en este caso es un objeto que representa una entidad de la aplicación, como un usuario, un producto, una publicación, etc. No necesitan pruebas unitarias.
- **Services:** (Servicios): Contiene servicios transparentes con el backend, estos servicios se encargan de realizar peticiones HTTP, usualmene no contienen lógica de negocio, y tampoco utilidades. No necesitan pruebas unitarias.
- **Context:** (Contexto): Contiene el contexto de la aplicación, es decir, el estado global de la aplicación, y las acciones que pueden modificar ese estado. Necesitan pruebas unitarias, pero es puramente funcional (.ts o .js).
- **Reducers:** (Reductores): Contiene los reductores, que son funciones puras que toman el estado actual y una acción, y devuelven un nuevo estado. Necesitan pruebas unitarias, pero es puramente funcional (.ts o .js).
- **UI:** (Interfaz de Usuario): Contiene los componentes de la interfaz de usuario, estos componentes se encargan de mostrar la información y de manejar la interacción del usuario. Necesitan pruebas unitarias, en este caso no es puramente funcional (.tsx o .jsx).

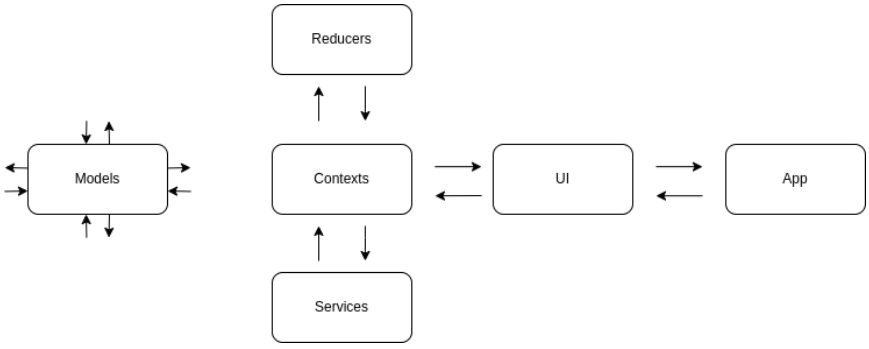

Figura 1: Diagrama de flujo de la arquitectura con el patrón de diseño de Reducer

- **App:** (Aplicación): Contiene el punto de entrada de la aplicación, y se encarga de inicializar el contexto y renderizar la interfaz de usuario.

El patrón de diseño de reductor es especialmente útil para manejar el estado global de una aplicación, ya que permite dividir la lógica de la aplicación en piezas más pequeñas y manejar las actualizaciones de estado de manera predecible y escalable.

En este diagrama se muestra el flujo de la arquitectura con el patrón de diseño de Reducer. La aplicación comienza en el punto de entrada, donde se inicializa el contexto y se renderiza la interfaz de usuario. Los componentes de la interfaz de usuario interactúan con el contexto para leer y actualizar el estado global de la aplicación. Los reductores manejan las actualizaciones de estado en respuesta a las acciones, y los servicios se encargan de realizar peticiones al backend. Los modelos definen la estructura de los datos que se utilizan en la aplicación.