EBAC – Engenharia Frontend

Módulo 48 – Exercício

React e tecnologias de transição de Estado

Reatividade e Estado

A reatividade web – isto é, a dinâmica de uma página web que não é necessariamente dinâmica, mas reage fortemente com a interação do usuário em tempo real – exige, na maioria das infraestruturas de aplicação, um aporte de atributos sobre o que está sendo manipulado, e o que será modificado, e a que tempo.

No contexto do React Framework, foi estabelecido o conceito de Estado (doravante e mais comumente chamado de “State”), o qual define um critério flexível para a presença, valor, ou interoperabilidade de um Objeto em um contexto dinâmico de uma aplicação. A utilização de States garante performance e segurança, pois reduz a lógica necessária para manter a dinâmica que outrora dependeria de JavaScript embutido em DOM (JQuery, por exemplo), o que é naturalmente mais complexo considerando que React manipula DOM em segundo plano.

API React de transição de Estado (useState())

A fim de facilitar e manter um padrão convencional de uso do State, mantendo a performance e a fácil manutenibilidade, o React promove o uso de uma API de transição de Estado, um hook de nome ‘useState()’.

Por definição, useState() é um hook que obrigatoriamente requer dois valores estruturados em array, os quais um é sempre uma variável que recebe o State, e o outro é sempre uma função (chamada de setState()) que permite a edição dessa variável. Como parâmetro opcional do hook, pode ser utilizado um valor inicial, o qual no conceito de State é chamado de initialState.

Dessa forma, um Componente pode ser inicializado já recebendo um valor inicial o qual pode ter seu Estado transitado, ou seja, alterado dentro da dinâmica do usuário, sem requerer o transporte da informação para outras páginas ou carregando outros componentes. É possível, ainda, se utilizar de uma função, tanto no initialState quanto no nextState (o valor a ser alterado), o que criará uma fila de alterações após a “re-renderização” do Componente.

O useState() permite que o Componente tenha uma renderização atualizada após uma mudança usando chamadas de renderização quando a variável tem seu valor alterado. Porém, o useState() não pode ser usado em Componentes de Classe do React. Nesse caso, seria necessário utilizar a função setState() diretamente um com handler o qual esteja sob binding e faça a alteração.

Apesar da existência da Context API para criar States menos transitórios e mais globais, ainda possuem limitações e complicam a visibilidade, quando comparado a soluções mais ideais para aplicações de longa escala, como o uso de bibliotecas baseadas em Flux Architeture.

Flux Architeture

Introduzida pelo Facebook em 2014, é uma arquitetura de gerenciamento de State e, apesar de possuir mesmo propósito que o hook de useState() do React, possuem algumas características a serem consideradas que fazem este diferencial.

Em primeiro lugar, é importante observar que useState() lida com os States internos dos Componentes e, apesar de poder ser transmitido de maneira hierarquicamente vertical (de pai para filho na hierarquia de componentes), trata de transições de State geralmente que importam a Componentes mais simples e pequenos.

Flux é designado para trabalhar principalmente com States os quais precisem ser acessados de qualquer lugar, com maior abrangência e importância em um projeto, onde useState() limitaria demais o escopo de atuação desses States em questão.

O useState() por ser um hook React, depende deste. Flux Architeture pode ser utilizado com outros frameworks e não é totalmente dependente do React para sua atuação, apesar de recomendado.

Flux direciona as responsabilidades do State para sua estrutura única, removendo a participação específica de algum Componente nesse papel, centralizando os States e direcionando a atenção, manuseio e manutenção desse State para uma forma global de gerenciamento dentro de um projeto. Esse aspecto torna Flux ideal para States de importância geral no projeto, principalmente para projetos grandes os quais necessitam com mais frequência desse tipo de infraestrutura.

Redux

Redux é uma biblioteca de gerenciamento de State para JavaScript, geralmente utilizada com React, que permite que aplicações web de grande escala possuam um gerenciamento de State mais centralizado dentro da arquitetura Flux de fácil acesso, escalabilidade e manutenção.

Uma das premissas do Redux é que todo o State de um aplicativo é armazenado na árvore de objetos do Store, facilitando a localização e inicialização dos States.

O segundo princípio fundamental é que todo State é somente leitura, e a única modificação do State acontece através de disparos de ações, componentes de modificação localizados na função reducer.

O terceiro princípio é o uso das funções reducers para centralizar as ações e executar a modificação do estado de maneira indireta.