

**Desenvolvimento de Aplicações Orientadas a Objetos**

**Prof. Ms. Vinícius Magnus - @vinimagnus**

**Atividade semipresencial**

O React possui alguns padrões bem difundidos, assim como o usando Hooks, temos outras boas práticas no desenvolvimento de aplicações utilizando essa biblioteca.

Relate um pouco sobre os principais padrões do react.

Nome: Gustavo Padilha

Professor: Vinicius Magnus

**React Provider Pattern**

É um conceito poderoso. Você não o encontra frequentemente ao usar apenas React, mas pode considerar usá-lo ao planejar seu aplicativo. Basicamente, ele pula a hierarquia obrigatória de passar props para cada componente, na sua árvore de componentes.

Muitas vezes, esse padrão pode ser visto em ação ao usar uma biblioteca de gerenciamento de estado externa. No Redux ou MobX, muitas vezes você acaba com um componente Provider no topo da hierarquia de componentes que cria a sua camada de estado (Redux/MobX) para ser usada na sua camada de exibição (React).

**Composição**

A principal característica do React são os componentes.

Componentes escritos por diferentes pessoas, devem funcionar bem juntos.

É importante para nós que você possa adicionar funcionalidade a um componente, sem gerar efeitos colaterais no código.

Por exemplo, é possível introduzir um state (estado) local dentro de um componente, sem alterar nenhum dos componentes que usam o mesmo.

Da mesma forma, será possível adicionar código de inicialização e “destruição” em qualquer componente quando necessário.

Componentes geralmente são descritos como “apenas funções”. Porém, pelo nosso ponto de vista, eles precisam ser mais do que isso para serem úteis. No React, os componentes descrevem qualquer comportamento “composable”, incluindo rendering (renderização), lifecycle (ciclo de vida) e state. Algumas bibliotecas, como Relay (Retransmissão) aumentam a responsabilidade dos componentes, como por exemplo descrever a dependência de dados. É possível que essas ideias possam voltar para o React também de alguma forma.

**Abstração comum**

O React não quer encher as suas aplicações com código de bibliotecas desnecessárias. No entanto, há algumas exceções.

Por exemplo, se o React não fornecesse suporte para state (estado) local e métodos do ciclo de vida, as pessoas criaram abstrações personalizadas para eles. E, quando há várias abstrações competindo, o React não pode forçar ou tirar vantagem das propriedades de nenhuma dessas abstrações. Deve-se ter um mínimo padrão a ser seguido.

**Estabilidade**

No Facebook, tem mais de 50 mil componentes utilizando React. Muitas outras empresas, como Twitter e Airbnb, também utilizam bastante o React. Estes são os motivos de evitarmos alterações nas APIs públicas ou no comportamento do React.

Entretanto, quando pensamos em estabilidade, não dizemos que “não haverá mudanças”. Pois assim, rapidamente irá paralisar. Em vez disso, preferimos definir estabilidade como "amplamente utilizado em produção e quando ocorrer alguma alteração, haverá um caminho claro (e preferencialmente automatizado) para realizar a migração.”

**Depuração**

Quando algo acontece de errado é importante que você tenha uma trilha para rastrear o erro até a sua origem no código. No React, props e state (estado) são essas trilhas.

Se verificar algo de errado em sua tela, você pode abrir o React DevTools, encontrar o componente responsável pela renderização, e verificar se as props e o state estão corretas. Se eles estiverem, você sabe que o problema está na função render() do componente, ou alguma função que é chamado pelo render(). É um problema isolado.