DIFERENÇA ENTRE ESSAS ABORDAGENS - (não estava limpando os campos titulo e descrição)

**Abordagem 1 (não funcionando - c**om setAtividades Dentro de uma Função de Atualização**)**

setAtividades((prevAtividades) => {

const novasAtividades = [...prevAtividades, atividade];

// Após adicionar a atividade, limpe os campos

setDescricao('');

setTitulo('');

setPrioridade('0');

return novasAtividades;

#### **O que acontece aqui:**

* **Atualização do estado de atividades**: setAtividades é chamado com uma função de atualização, onde você usa prevAtividades para calcular o novo estado.
* **Limpeza dos campos (setDescricao, setTitulo, setPrioridade) dentro da função de atualização**: Você chama as funções de limpeza logo após calcular o novo estado de atividades.

#### **Por que isso não funcionava corretamente:**

O que ocorre é que, ao usar o padrão de **função de atualização** em setAtividades, React faz uma atualização assíncrona, ou seja, o **estado de atividades não é atualizado imediatamente**. A função setAtividades começa o processo de atualização, mas o que acontece é que o React ainda não completou essa atualização até o momento em que você está chamando as funções setDescricao, setTitulo e setPrioridade.

O React **não garante que as atualizações de estado sejam sincronizadas**. Então, ao chamar setDescricao(''), setTitulo('') e setPrioridade('0') dentro dessa função, eles são **agendados para a próxima renderização** e **não são executados de forma imediata**. Isso pode causar uma situação onde os campos não são limpos na mesma renderização, e a interface ainda mantém os valores anteriores até que o estado seja efetivamente atualizado e o componente seja re-renderizado.

**Abordagem 2 (funcionando -** Sem a Função de Atualização Dentro de setAtividades**)**

setAtividades(prevAtividades => [...prevAtividades, atividade]);

// Limpa os campos imediatamente após adicionar a atividade

setDescricao('');

setTitulo('');

setPrioridade('0');

#### **O que acontece aqui:**

* **Atualização do estado de atividades**: setAtividades é chamado diretamente com uma função de atualização, mas sem a parte de limpeza dentro dela.
* **Limpeza dos campos após setAtividades**: Após a chamada de setAtividades, você limpa os campos imediatamente.

#### **Por que isso funciona:**

Ao separar a atualização de atividades e a limpeza dos campos em chamadas distintas, você está **garantindo que a limpeza dos campos aconteça logo após a adição da atividade** e **antes de qualquer re-renderização do componente**. Essa abordagem evita o problema do estado assíncrono e garante que, ao clicar no botão de adicionar, o React limpe os campos antes de exibir a nova lista de atividades.

**Importante**: A função setAtividades ainda é assíncrona, mas você não depende de uma lógica interna para limpar os campos dentro da mesma função. Ou seja, o React atualiza o estado de atividades, e depois, o código que limpa os campos (setDescricao, setTitulo, setPrioridade) é executado de forma direta, **imediatamente** após essa chamada de atualização.

### **Diferença-chave**

* **Com a função de atualização dentro de setAtividades**: A limpeza dos campos é agendada depois da atualização do estado. Como o estado é atualizado de forma assíncrona, a limpeza pode não ocorrer imediatamente, causando um comportamento indesejado.
* **Com a limpeza fora da função de atualização**: A limpeza dos campos é feita imediatamente após a atualização do estado. Dessa forma, você garante que os campos sejam limpos antes de a interface ser atualizada com a nova atividade.

### **Conclusão**

A principal diferença é que, ao **manter a limpeza de campos fora da função de atualização de setAtividades**, você consegue controlar o fluxo da atualização de estado de forma mais previsível, evitando que o React não atualize os campos imediatamente após a adição da atividade.