Exercício 3: Otimização de Performance no Frontend

Para os problemas de performance o React.memo pode ser usado para memorizar componentes e evitar as renderizações desnecessárias dos componentes, o useMemo para fazer a exibição da lista de produtos (sem precisar pegar novamente a lista quando o componente renderizar) e o useCallback para prevenir a recriação das funções e passar os memo callbacks pros componentes filhos, diminuindo assim os tempos de carregamento (já que apenas os componentes que precisam ser renderizados são atualizados na tela) e uma navegação mais suave.

Exemplo com o react.memo

```
const LinhaProduto = React.memo(({ produto }) => {
  console.log(`Renderizando linha do produto: ${produto.nome}`);
  return (
    <tr>
      <td>{produto.id}</td>
      <td>{produto.nome}</td>
      <td>{produto.preco}</td>
    </tr>
  );
});
```

Exemplo do useMemo

```
const TabelaProdutos = ({ produtos }) => {
  // recalcula apenas quando "produtos" muda
  const linhas = useMemo(() => {
    return produtos.map((produto) => (
      <LinhaProduto key={produto.id} produto={produto} />
    ));
  }, [produtos]);

  return (
    <table>
      <thead>
        <tr>
          <th>ID</th>
          <th>Nome</th>
          <th>Preço</th>
        </tr>
      </thead>
      <tbody>{linhas}</tbody>
    </table>
  );
};
```

Exemplo com useCallback em uma tabela existente

```
const [produtos, setProdutos] = useState([
  { id: 1, nome: "", preco: 5.80 },
  { id: 11, nome: "Produto B", preco: 2.0 },
  { id: 12, nome: "Produto C", preco: 7 },
]);

const adicionarProduto = useCallback(() => {
  setProdutos((prevProdutos) => [
    ...prevProdutos,
    { id: prevProdutos.length + 1, nome: `produto ${prevProdutos.length + 1}`, preco: 5.50 },
  ]);
}, []);
```