Frontend React — Exercício Guiado (Listas, Chaves e Ordenação)

Prof. Me. Tássio Auad

Laboratório de Programação FullStack

Contexto

No VassCommerce, o frontend (React) exibirá **categorias** e seus **produtos**. Antes de integrar com a API, trabalharemos com dados locais para dominar: **renderização de listas**, **uso correto de `key`** e **ordenação** (asc/desc) sem quebrar o estado dos componentes.

Objetivo

Construir uma tela `<CategoriasPage/>` que:

- Renderize **categorias** e, dentro de cada uma, a lista de **produtos**.
- Use **chaves estáveis** (`key`) tanto para categorias quanto para produtos.
- Destaque em **verde** os produtos com preço < R$ 100,00.
- Ofereça um **select** para ordenar produtos por preço (`asc`/`desc`) preservando `keys`.

Modelo de dados (mock)

Crie um arquivo `src/data/categorias.js` com o mock abaixo:

```
export const categorias = [
  {
    id: 10,
    nome: "Periféricos",
    produtos: [
      { id: 101, nome: "Teclado Mecânico", preco: 150.90 },
      { id: 102, nome: "Mouse Óptico",     preco: 89.50  }
    ]
  },
```

```
  {
    id: 20,
    nome: "Monitores",
    produtos: [
      { id: 201, nome: "Monitor 24\" FullHD", preco: 749.00 },
      { id: 202, nome: "Monitor 27\" QHD",    preco: 1299.90 }
    ]
  }
]
```

Passo a passo (guiado)

Passo 1 — Criar a página `CategoriasPage`

Crie `src/pages/CategoriasPage.jsx`:

```
import { useMemo, useState } from 'react'
import { categorias as base } from '../data/categorias'

export default function CategoriasPage() {
  const [ordem, setOrdem] = useState('asc') // 'asc' ou 'desc'

  // NUNCA altere o array original in-place.
  // Use useMemo para derivar uma versão ordenada quando 'ordem' mudar.
  const categoriasOrdenadas = useMemo(() => {
    // copia rasa das categorias
    const copia = base.map(cat => ({
      ...cat,
      produtos: [...cat.produtos] // copia a lista de produtos
    }))
    // ordena cada lista de produtos por preco
    for (const cat of copia) {
      cat.produtos.sort((a, b) =>
        ordem === 'asc' ? a.preco - b.preco : b.preco - a.preco
      )
    }
    return copia
  }, [ordem])

  return (
    <main>
      <header style={{display:'flex', gap:12, alignItems:'center'}}>
        <h2>Categorias</h2>
        <label>
          Ordenar por preço:
          <select value={ordem} onChange={e => setOrdem(e.target.value)}>
```

```
            <option value="asc">Menor para maior</option>
            <option value="desc">Maior para menor</option>
          </select>
        </label>
      </header>

      {categoriasOrdenadas.map(cat => (
        <section key={cat.id}>
          <h3>{cat.nome}</h3>
          <ul>
            {cat.produtos.map(p => (
              <li key={p.id}
                  style={{ color: p.preco < 100 ? 'green' : 'inherit' }}>
                {p.nome} - R$ {p.preco.toFixed(2)}
              </li>
            ))}
          </ul>
        </section>
      ))}
    </main>
  )
}
```

Checkpoint: Rodar a aplicação e verificar se:

- A página lista as categorias e produtos.
- O select altera a ordem dos preços por categoria.
- Produtos < R$ 100,00 aparecem verdes.

Passo 2 — Componentizar (boa prática)

Crie `src/components/CategoriaSection.jsx`:

```
export default function CategoriaSection({ categoria }) {
  return (
    <section>
      <h3>{categoria.nome}</h3>
      <ul>
        {categoria.produtos.map(p => (
          <li key={p.id}
              style={{ color: p.preco < 100 ? 'green' : 'inherit' }}>
            {p.nome} - R$ {p.preco.toFixed(2)}
          </li>
        ))}
      </ul>
    </section>
```

```
  )
}
```

Ajuste `CategoriasPage.jsx` para usar o componente:

```
import CategoriaSection from '../components/CategoriaSection'
// ...
{categoriasOrdenadas.map(cat => (
  <CategoriaSection key={cat.id} categoria={cat} />
))}
```

Checkpoint: Nada muda visualmente, mas o código fica mais claro e reutilizável.

Passo 3 — Extra: extrair o item de produto (refino)

Crie `src/components/ProdutoItem.jsx`:

```
export default function ProdutoItem({ produto }) {
  const barato = produto.preco < 100
  return (
    <li style={{ color: barato ? 'green' : 'inherit' }}>
      {produto.nome} - R$ {produto.preco.toFixed(2)}
    </li>
  )
}
```

Atualize `CategoriaSection.jsx`:

```
import ProdutoItem from './ProdutoItem'

export default function CategoriaSection({ categoria }) {
  return (
    <section>
      <h3>{categoria.nome}</h3>
      <ul>
        {categoria.produtos.map(p => (
          <ProdutoItem key={p.id} produto={p} />
        ))}
      </ul>
    </section>
  )
}
```

Checkpoint: Código mais modular. `key` permanece no **map** (ponto onde a lista é criada).

Ajuda visual (CSS mínimo)

Crie `src/index.css` (ou ajuste o seu):

```
main { padding: 16px; font-family: system-ui, Arial, sans-serif; }
h2, h3 { margin: 8px 0; }
ul { padding-left: 18px; }
header select { margin-left: 8px; }
section { margin-bottom: 18px; }
```

Erros comuns (e como resolver)

- **Usar índice como `key`**: evita bugs de reconciliação do React. Use sempre `id` único.
- **Fazer `sort` in-place no array original**: isso muta o *mock* e pode causar efeitos colaterais. Faça cópias com `[...arr].sort(...)` ou copie antes (como no `useMemo`).
- **Trocar `key` quando ordenar**: a `key` *não* muda porque o `id` não mudou. Apenas a **posição** muda.
- **Formatar preço com vírgula**: use `toFixed(2)` para duas casas. Caso queira separador local, depois veremos `Intl.NumberFormat`.
- **Re-render desnecessário**: o `useMemo` reduz processamento quando o valor de `ordem` não muda.

Checklist de entrega

- ( ) **Categorias** renderizadas com `key` estável (`id`).
- ( ) **Produtos** renderizados com `key` estável (`id`).
- ( ) Produtos < R$ 100,00 destacados em **verde**.
- ( ) **Select** funcional para ordenar por preço (`asc`/`desc`).
- ( ) **Componentização** aplicada (`CategoriaSection`, `ProdutoItem`).
- ( ) **Sem** mutar arrays originais; cópias e `useMemo` corretos.

Desafios bônus (opcional)

- Adicionar filtro por `nome` do produto (input controlado).
- Mostrar `quantidade de produtos` por categoria.
- Exibir `tag` “promo” para itens < R$ 100,00.

Entrega

- Código-fonte (pasta do projeto React) com `CategoriasPage.jsx` e componentes auxiliares.
- **README.md** curto explicando como rodar e onde está a página.
- (Opcional) Print da tela em ambos os estados de ordenação.