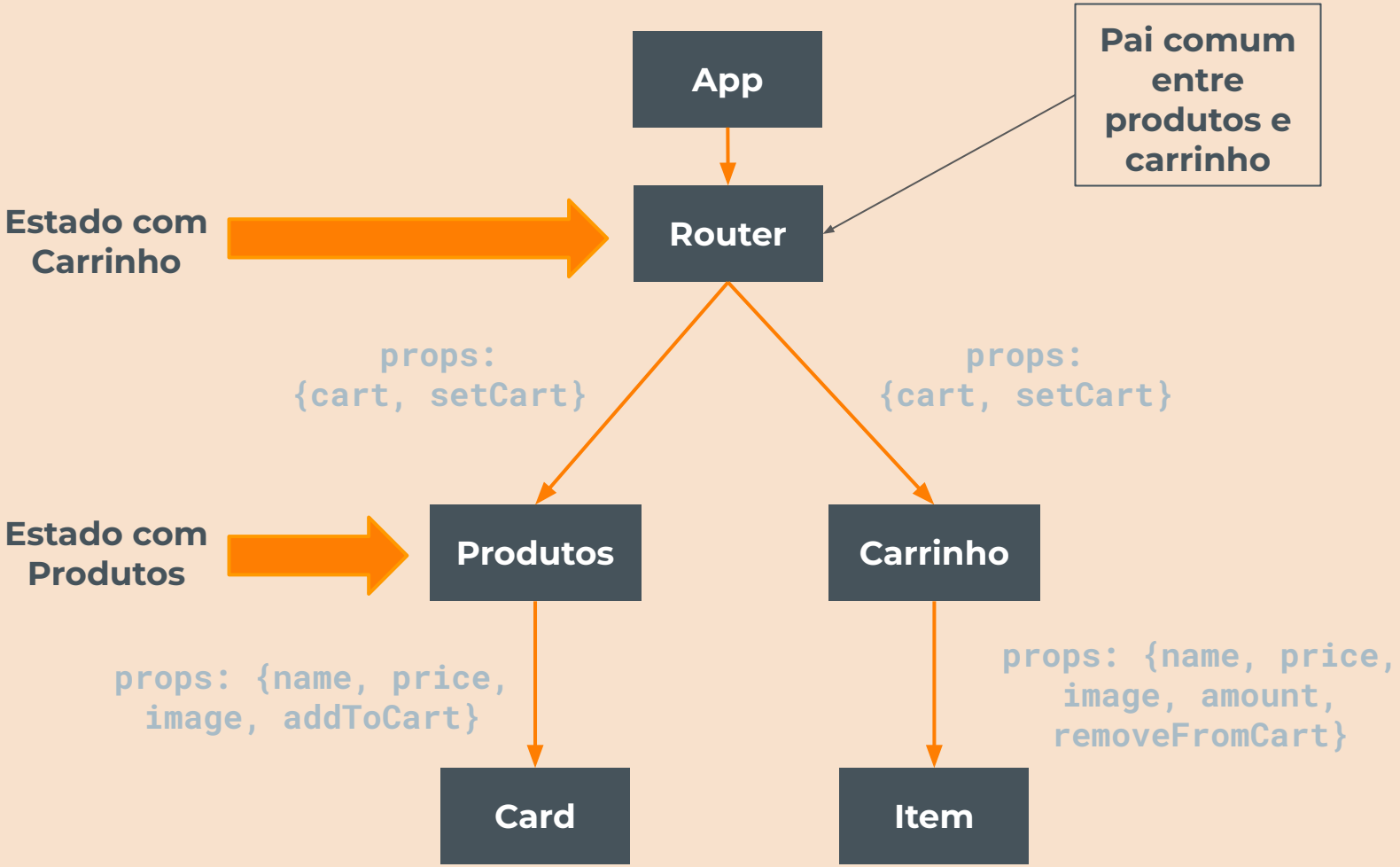
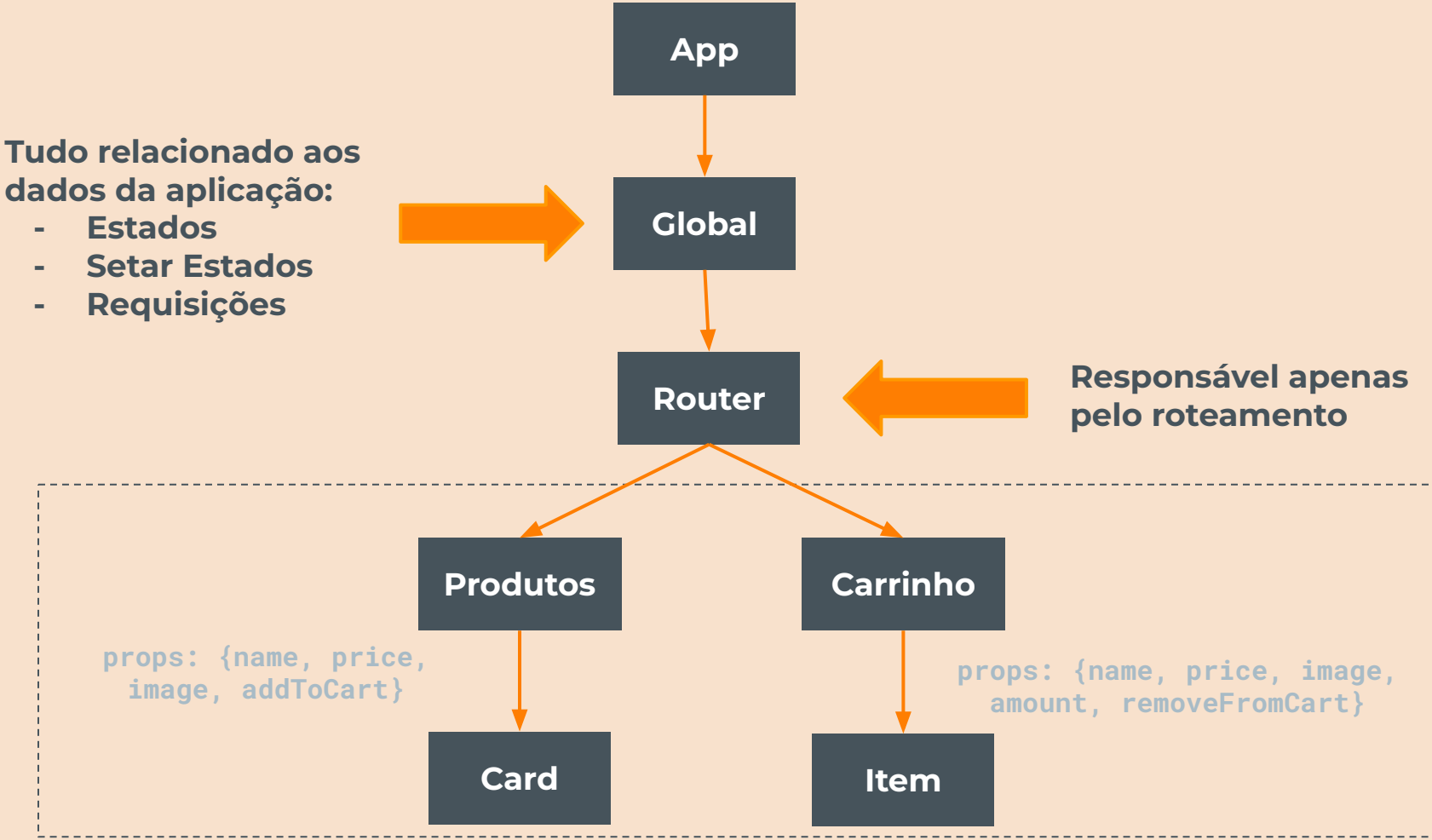
Estado Global usando Context

Estados Globais com Context


App
Pai comum entre produtos e carrinho
Estado com Carrinho
Router
props: {cart, setCart}
props: {cart, setCart}
Estado com Produtos
Produtos
Carrinho
props: {name, price, image, addToCart}
props: {name, price, image, amount, removeFromCart}
Card
Item

Estado Global 🌎

- É uma **camada** de dados **única** que pode ser acessada e modificada **diretamente** por **qualquer componente** da nossa aplicação


App
Global
Router
Tudo relacionado aos dados da aplicação:
- Estados
- Setar Estados
- Requisições
Responsável apenas pelo roteamento
Produtos
Carrinho
props: {name, price, image, addToCart}
props: {name, price, image, amount, removeFromCart}
Card
Item

Criação do Estado Global 🌎

- Criamos uma pasta **global** com dois arquivos:
  - **GlobalStateContext** ⇒ `React.createContext()`
  - **GlobalState** ⇒ Componente onde guardamos os estados e requisições da aplicação

- Chamamos o componente **GlobalState** como pai de todos os outros dentro do App.js

GlobalStateContext

```
1 import React from "react"
2
3 // Criação do contexto
4 const GlobalStateContext = React.createContext()
5
6 export default GlobalStateContext
```

GlobalState

Todos os estados que são relevantes para a aplicação como um todo

```
1  import React, { useState } from "react"
2  import GlobalStateContext from "./GlobalStateContext"
3
4  const GlobalState = (props) => {
5
6    // Criação de todos os estados gerais da aplicação
7    const [estado1, setEstado1] = useState([]);
8    const [estado2, setEstado2] = useState({});
9
10   // Funções para todas as requisições que serão feitas
11   // Caso elas retornem algum dado, esses dados devem ser guardados
12   // nos estados criados logo acima
13   const getInformacaoA = () => {
14     // ... Request A
15   }
16
17   const getInformacaoB = () => {
18     // ... Request B
19   }
20
21   // Organização das informações: criamos três categorias para facilitar
22   // achar essas informações nos componentes consumers
23   const states = { estado1, estado1 }
24   const setters = { setEstado1, setEstado2 }
25   const requests = { getInformacaoA, getInformacaoB }
26
27   // Provider do estado que irá receber como filhos os componentes que
28   // precisarão dessas informações
29   return (
30     <GlobalStateContext.Provider value={{ states, setters, requests }}>
31       {props.children}
32     </GlobalStateContext.Provider>
33   )
34 }
35
36 export default GlobalState;
37
```

GlobalState

Todos os estados que são relevantes para a aplicação como um todo

Todas as requisições que fazemos para o backend ficam centralizadas aqui

```
1  import React, { useState } from "react"
2  import GlobalStateContext from "./GlobalStateContext"
3
4  const GlobalState = (props) => {
5
6    // Criação de todos os estados gerais da aplicação
7    const [estado1, setEstado1] = useState([]);
8    const [estado2, setEstado2] = useState({});
9
10   // Funções para todas as requisições que serão feitas
11   // Caso elas retornem algum dado, esses dados devem ser guardados
12   // nos estados criados logo acima
13   const getInformacaoA = () => {
14     // ... Request A
15   }
16
17   const getInformacaoB = () => {
18     // ... Request B
19   }
20
21   // Organização das informações: criamos três categorias para facilitar
22   // achar essas informações nos componentes consumers
23   const states = { estado1, estado1 }
24   const setters = { setEstado1, setEstado2 }
25   const requests = { getInformacaoA, getInformacaoB }
26
27   // Provider do estado que irá receber como filhos os componentes que
28   // precisarão dessas informações
29   return (
30     <GlobalStateContext.Provider value={{ states, setters, requests }}>
31       {props.children}
32     </GlobalStateContext.Provider>
33   )
34 }
35
36 export default GlobalState;
37
```

GlobalState

Todos os estados que são relevantes para a aplicação como um todo

Todas as requisições que fazemos para o backend ficam centralizadas aqui

Essa parte é opcional! Apenas pra organização do código

```
1  import React, { useState } from "react"
2  import GlobalStateContext from "./GlobalStateContext"
3
4  const GlobalState = (props) => {
5
6    // Criação de todos os estados gerais da aplicação
7    const [estado1, setEstado1] = useState([]);
8    const [estado2, setEstado2] = useState({});
9
10   // Funções para todas as requisições que serão feitas
11   // Caso elas retornem algum dado, esses dados devem ser guardados
12   // nos estados criados logo acima
13   const getInformacaoA = () => {
14     // ... Request A
15   }
16
17   const getInformacaoB = () => {
18     // ... Request B
19   }
20
21   // Organização das informações: criamos três categorias para facilitar
22   // achar essas informações nos componentes consumers
23   const states = { estado1, estado1 }
24   const setters = { setEstado1, setEstado2 }
25   const requests = { getInformacaoA, getInformacaoB }
26
27   // Provider do estado que irá receber como filhos os componentes que
28   // precisarão dessas informações
29   return (
30     <GlobalStateContext.Provider value={{ states, setters, requests }}>
31       {props.children}
32     </GlobalStateContext.Provider>
33   )
34 }
35
36 export default GlobalState;
37
```

GlobalState

Todos os estados que são relevantes para a aplicação como um todo

Todas as requisições que fazemos para o backend ficam centralizadas aqui

Essa parte é opcional! Apenas pra organização do código

Componente que retorna o provider

```jsx
import React, { useState } from "react"
import GlobalStateContext from "./GlobalStateContext"

const GlobalState = (props) => {

  // Criação de todos os estados gerais da aplicação
  const [estado1, setEstado1] = useState([]);
  const [estado2, setEstado2] = useState({});

  // Funções para todas as requisições que serão feitas
  // Caso elas retornem algum dado, esses dados devem ser guardados
  // nos estados criados logo acima
  const getInformacaoA = () => {
    // ... Request A
  }

  const getInformacaoB = () => {
    // ... Request B
  }

  // Organização das informações: criamos três categorias para facilitar
  // achar essas informações nos componentes consumers
  const states = { estado1, estado1 }
  const setters = { setEstado1, setEstado2 }
  const requests = { getInformacaoA, getInformacaoB }

  // Provider do estado que irá receber como filhos os componentes que
  // precisarão dessas informações
  return (
    <GlobalStateContext.Provider value={{ states, setters, requests }}>
      {props.children}
    </GlobalStateContext.Provider>
  )
}

export default GlobalState;
```

App.js

```
1  import React from "react"
2  import Router from "./routes/Router"
3  import GlobalState from "./global/GlobalState"
4
5  const App = () => {
6    // Todos os componentes dentro do GlobalState
7    // têm acesso ao contexto GlobalStateContext
8    return (
9      <GlobalState>
10       <Router />
11     </GlobalState>
12   )
13 }
14
15 export default App
```

Uso do Estado Global 🌎

- Nos componentes que precisam acessar os dados (Consumers), precisamos apenas:
  - Usar o hook **useContext** para guardar os dados de {states, setters, requests} em variáveis
  - **Acessar** esses dados e **usá-los** da maneira que precisar!

Consumer

```
// Desestruturação do objeto passado pelo Provider no Contexto
const { states, setters, requests } = useContext(GlobalStateContext)

// Fazer uma requisição:
useEffect(() => {
    requests.getInformacaoA();
}, [requests])

// Acessar um estado
states.estado1

// Setar um estado
setters.setEstado1(["banana", "maçã", "morango"])
```

Conclusões

Problemas que queríamos resolver ⚠️

- **Dados espalhados** entre vários componentes
- Separação de **Responsabilidades**
- Não há uma **fonte única de verdade**
- **Props Drilling**