

# Estado Global usando Context

Labenu\_



# O que vamos ver hoje?

- Revisão
- Estados Globais com useContext
- Exercício Prático



# Estado Global

Labenu\_



# Estado Global

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



# Problemas que queremos resolver

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



# React Context

Labenu\_

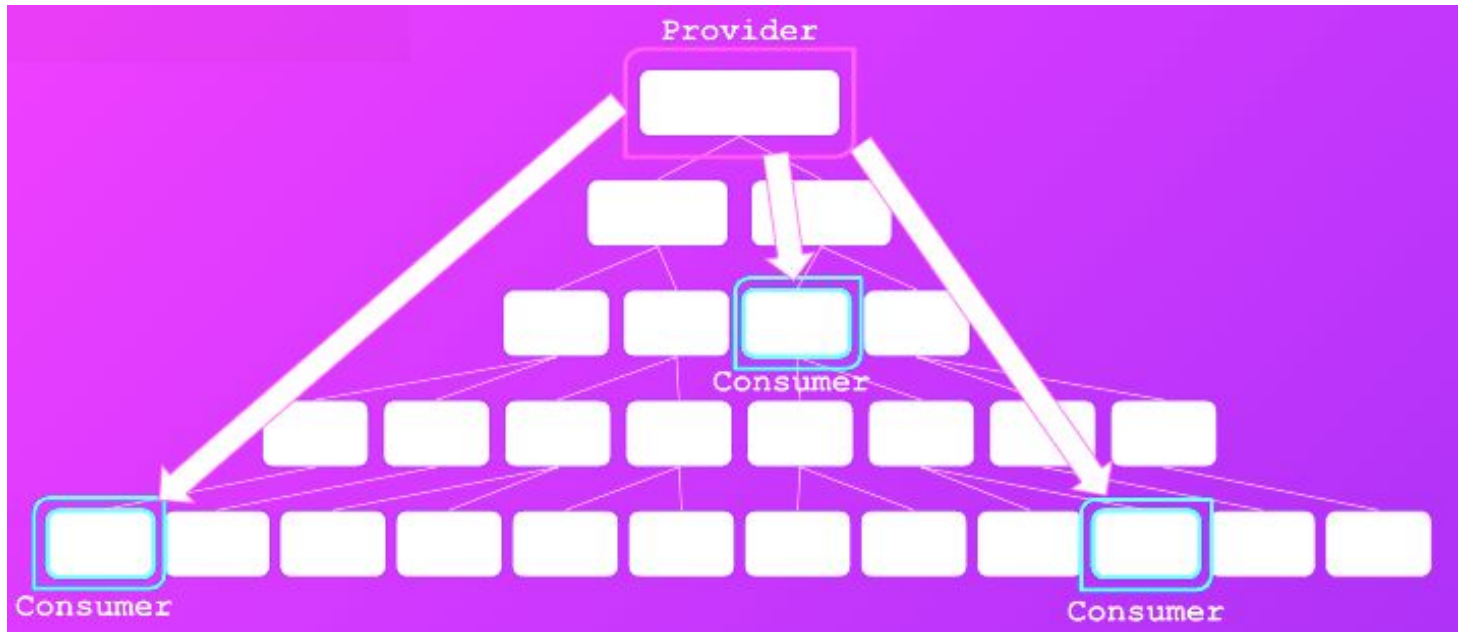


# React Context - Definição

- O React Context nos permite criar um **contexto**, que é como um **estado** que podemos utilizar em qualquer componente **sem precisar usar props**
- Todo contexto do React tem 2 peças importantes:
  - **Provider:** guarda dados e disponibiliza para os Consumers
  - **Consumer:** tem acesso aos dados de um provider



# React Context - Definição





# React Context - Contexto

- **Contexto** é como um **estado** que pode ser acessado diretamente de qualquer componente

```
1 export const ContextoBola = React.createContext()
```



# React Context - Provider

- **Provider** é o componente que guarda o dado

```
1 import PaiSemBola from "../components/PaiSemBola"
2 import { ContextoBola } from "../context"
3
4 export default function App() {
5   const bola = {
6     tipo: "basquete",
7     imagem: "https://www.bolas.com/basquete.png"
8   }
9
10  return (
11    
```



# React Context - Consumer

- **Consumers** são os componentes que recebem esses dados através do hook `useContext()`

```
1 import React, {useContext} from "react"
2 import {ContextoBola} from "../context"
3
4 const FilhoComBola = () => {
5   ➡ const bola = useContext(ContextoBola)
6
7   return (
8     <div>
9       <h1>{'Bola: ${bola.tipo}'}</h1>
10      <img alt="bola" src={bola.imagem} />
11    </div>
12  )
13 }
```



# Estados Globais com Context

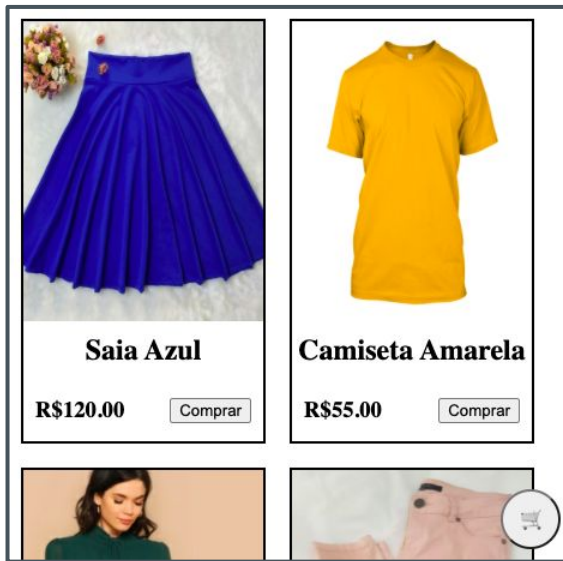
Labenu\_

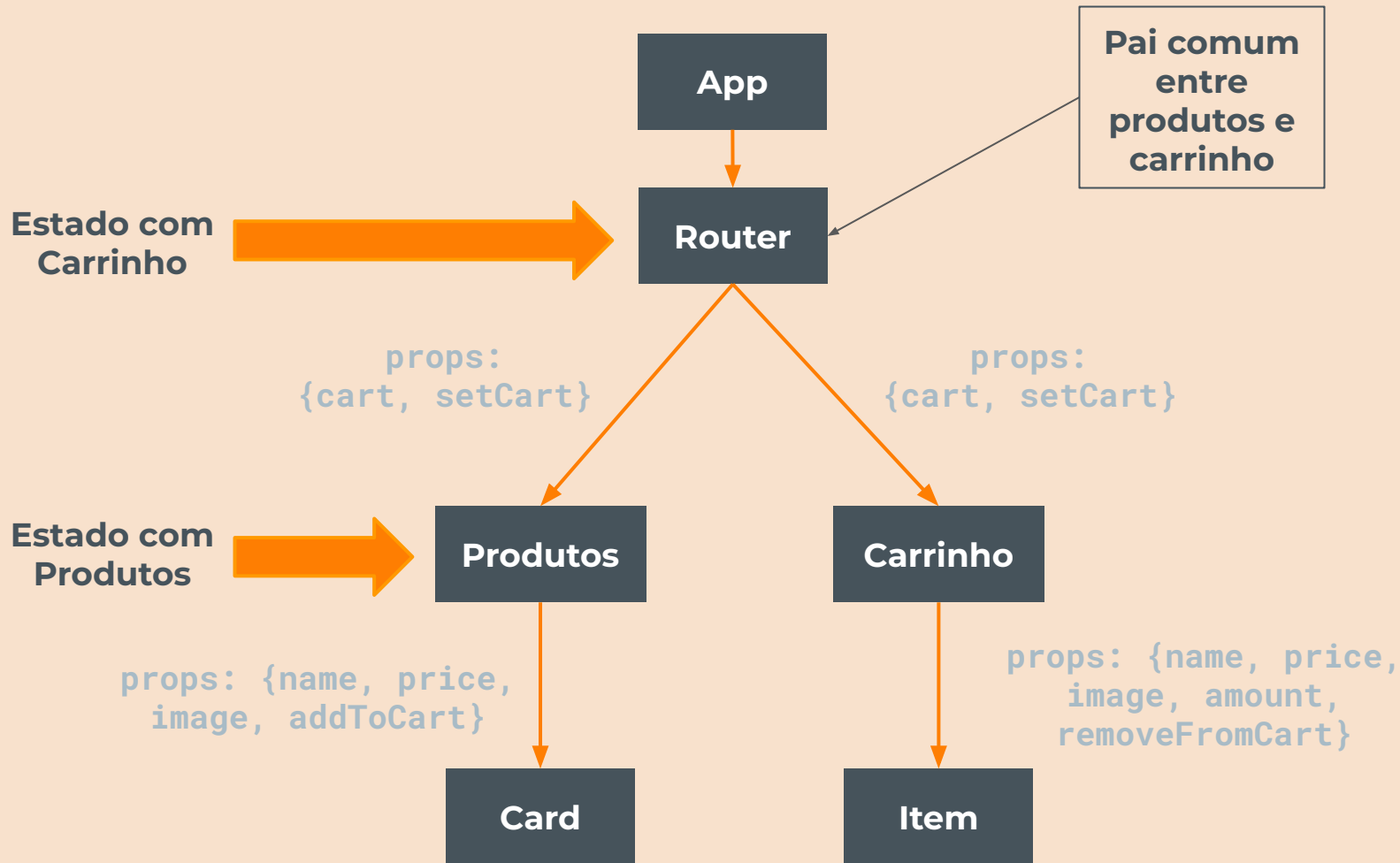




# Exercício 1

- Refatorar nosso site do e-commerce para utilizar um Estado Global
- Utilizar a ferramenta de Context para tornar os dados disponíveis para toda a árvore
- [Link do Projeto](#) (de onde paramos na outra aula)





# Estado Global

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

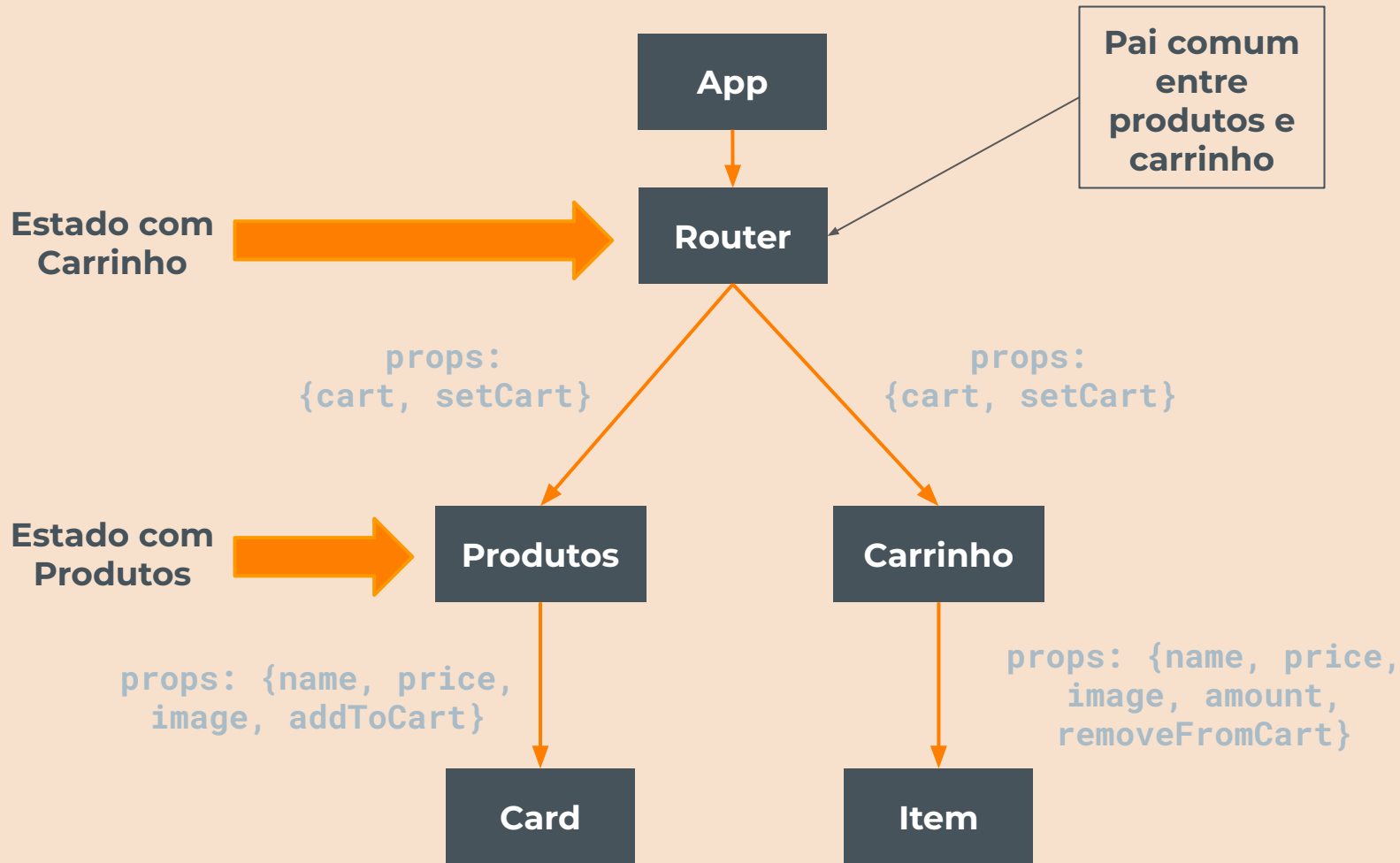


# Pausa para relaxar 🧘

10 min







Tudo relacionado aos dados da aplicação:

- Estados
- Setar Estados
- Requisições



App

Global

Router

Responsável apenas pelo roteamento



Produtos

Carrinho

props: {name, price,  
image, addToCart}

Card

props: {name, price, image,  
amount, removeFromCart}

Item



# Organizando o que a gente fez

Labenu\_



# Criação do Estado Global

- Criamos uma pasta **global** com dois arquivos:
  - **GlobalStateContext**  $\Rightarrow$  `React.createContext()`
  - **GlobalState**  $\Rightarrow$  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, estado2 }
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, estado2 }
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

```
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, estado2 }
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
```



# 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

```
1 // Desestruturação do objeto passado pelo Provider no Contexto
2 const { states, setters, requests } = useContext(GlobalStateContext)
3
4 // Fazer uma requisição:
5 useEffect(() => {
6     requests.getInformacaoA();
7 }, [requests])
8
9 // Acessar um estado
10 states.estado1
11
12 // Setar um estado
13 setters.setEstado1(["banana", "maçã", "morango"])
```



# Conclusões

Labenu\_




# Problemas que queríamos resolver

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





# Problemas que queríamos resolver

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






# Problemas que queríamos resolver

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





# Problemas que queríamos resolver

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



# Problemas que queríamos resolver ⚠

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



# Resumo

Labenu\_



# Resumo

- Juntando o conceito de estado global com a ferramenta React Context, conseguimos criar um estado global que:
  - **Unifica todos os nossos dados** em uma única camada da aplicação
  - É acessível por todos os componentes **diretamente**, sem a necessidade de usar props
  - **Isola as responsabilidades** das telas e dos dados



# GlobalStateContext

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



# GlobalState

```
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, estado2 }
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
```



# 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
```



# Consumer

```
1 // Desestruturação do objeto passado pelo Provider no Contexto
2 const { states, setters, requests } = useContext(GlobalStateContext)
3
4 // Fazer uma requisição:
5 useEffect(() => {
6     requests.getInformacaoA();
7 }, [requests])
8
9 // Acessar um estado
10 states.estado1
11
12 // Setar um estado
13 setters.setEstado1(["banana", "maçã", "morango"])
```





# Dúvidas? 🧐

Labenu\_





Obrigado(a)!