O que é um Estado Global?

Labenu_

O que vamos ver hoje?

- Como pensamos em estados até agora
- Problemas com essa abordagem
- Estado Global
- Como fazer um Estado Global?

Como pensamos em estados até agora?

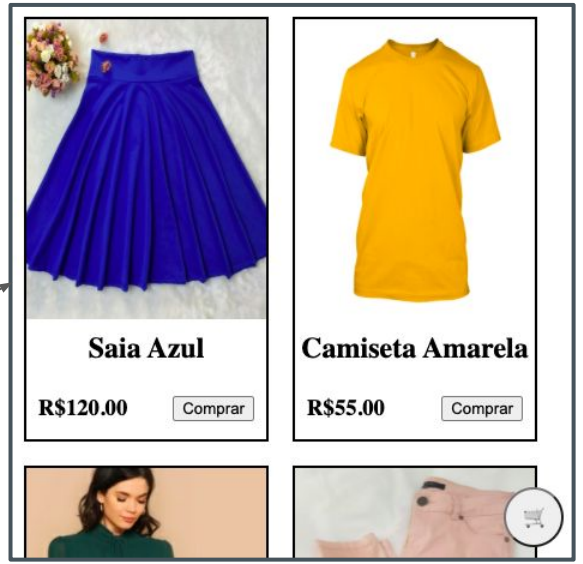
Labenu_

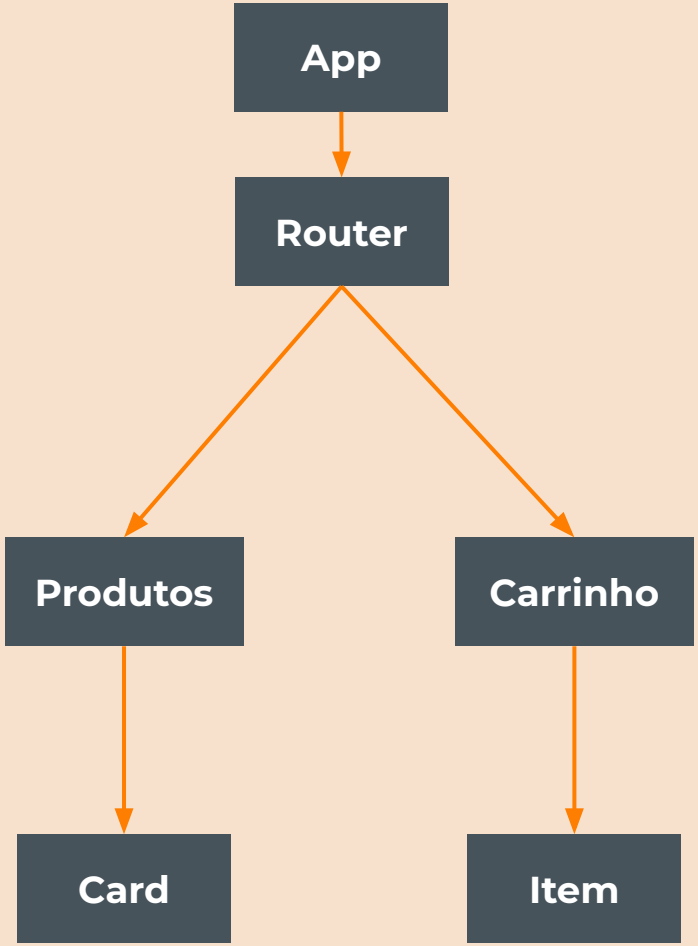
Exercício 1

Com tudo que vimos até hoje, criar as duas funcionalidades básicas de um e-commerce:

- Lista de Produtos
- Carrinho

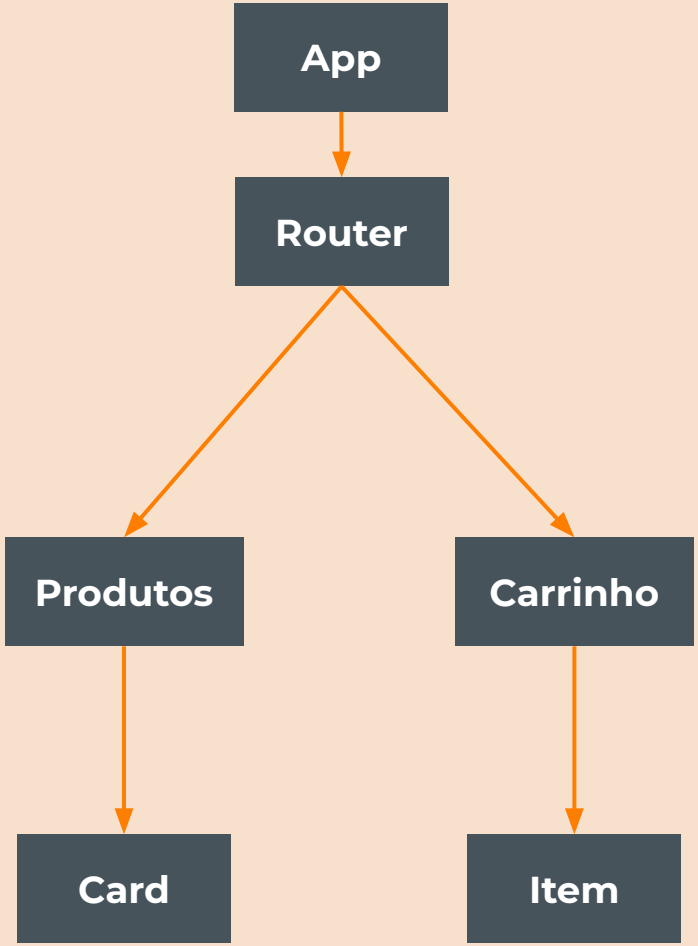
Link do Template e Link da API

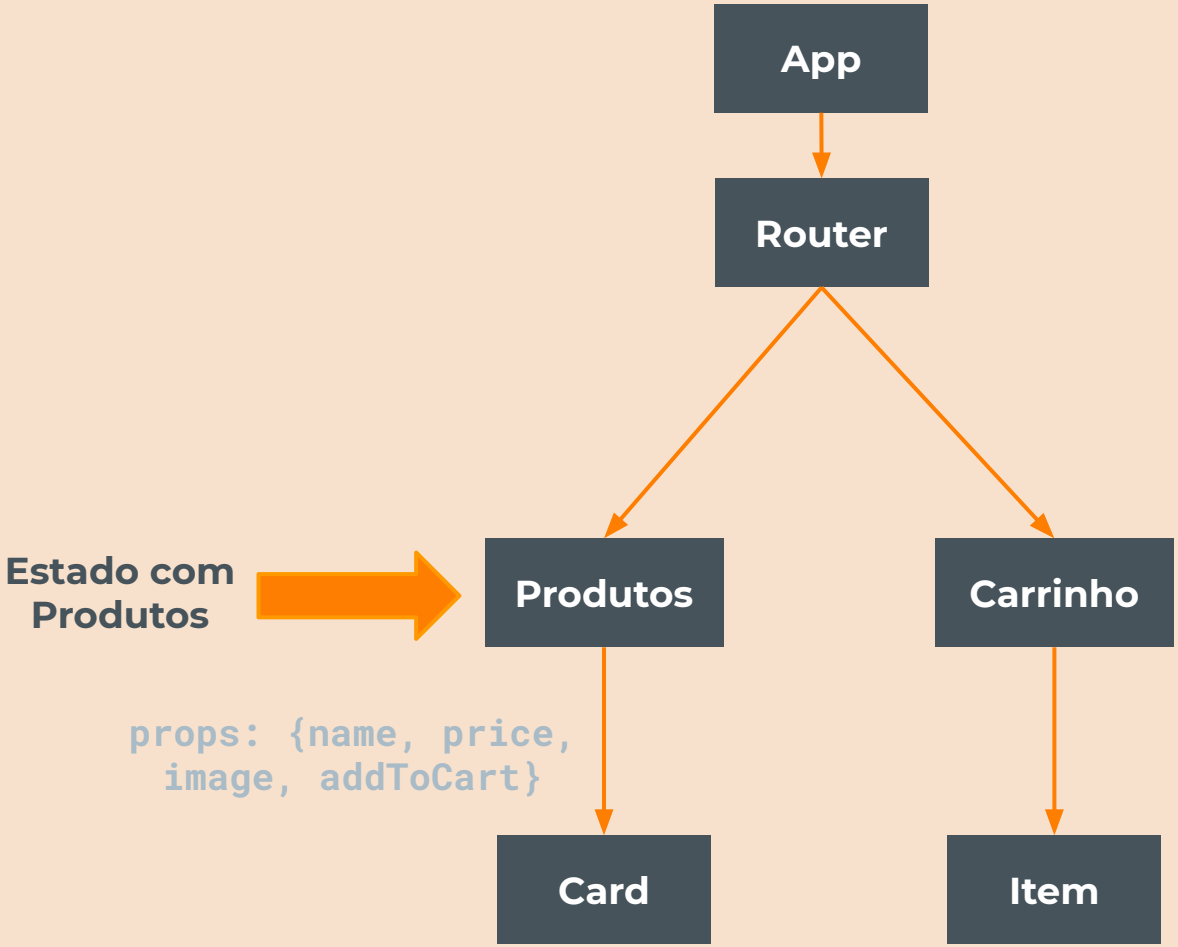


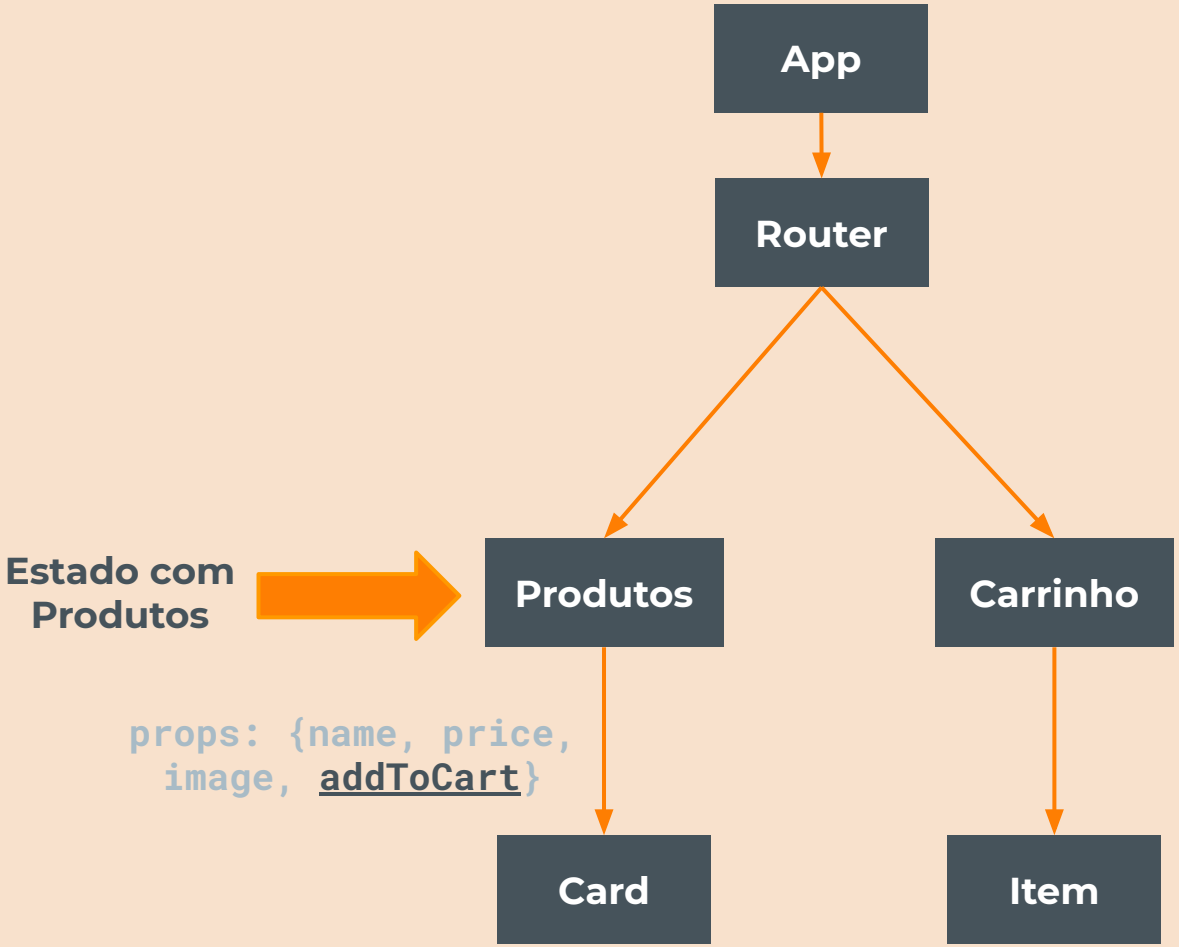

App
Router
Produtos
Carrinho
Card
Item

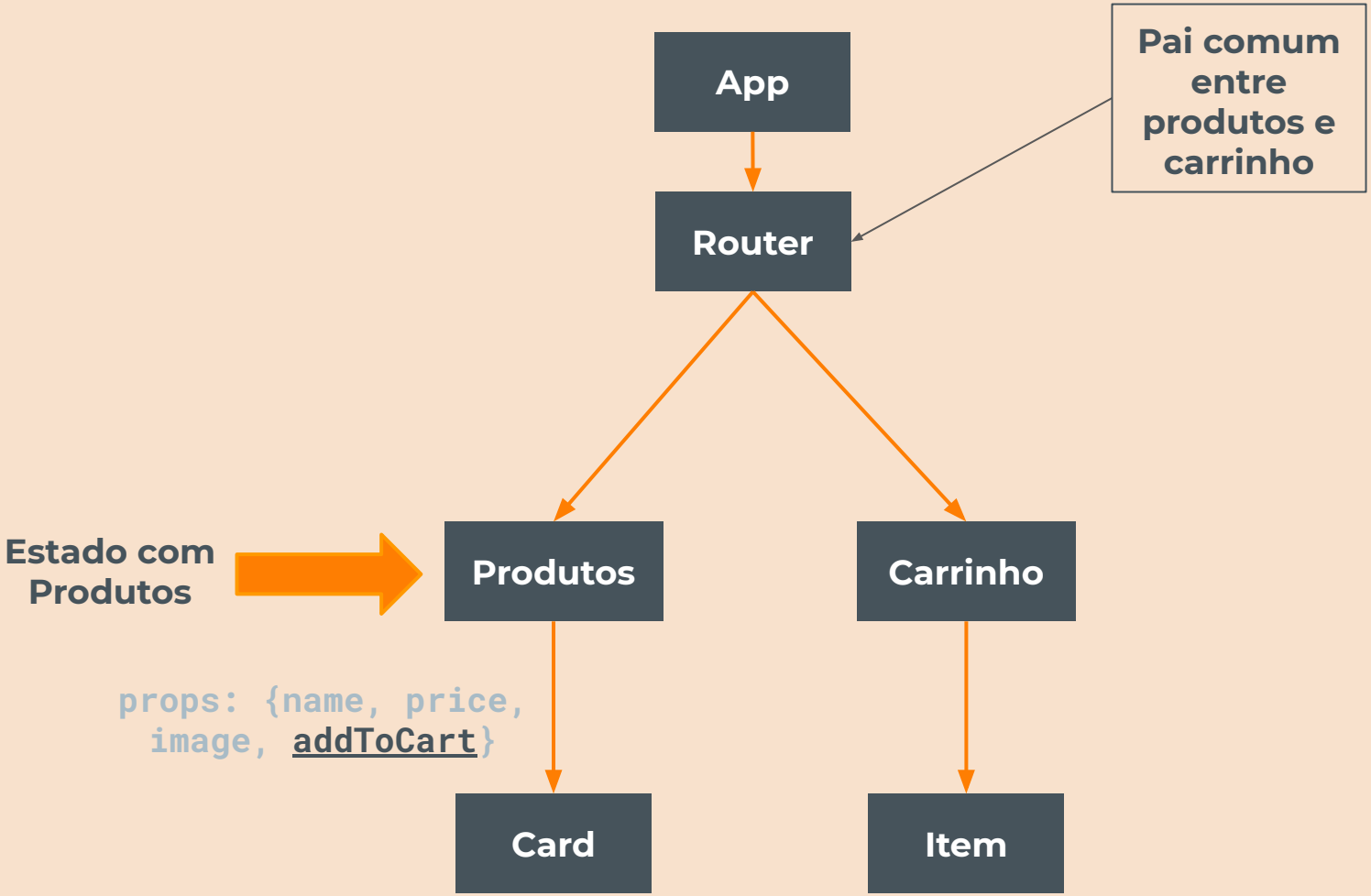
Pausa para relaxar 😴

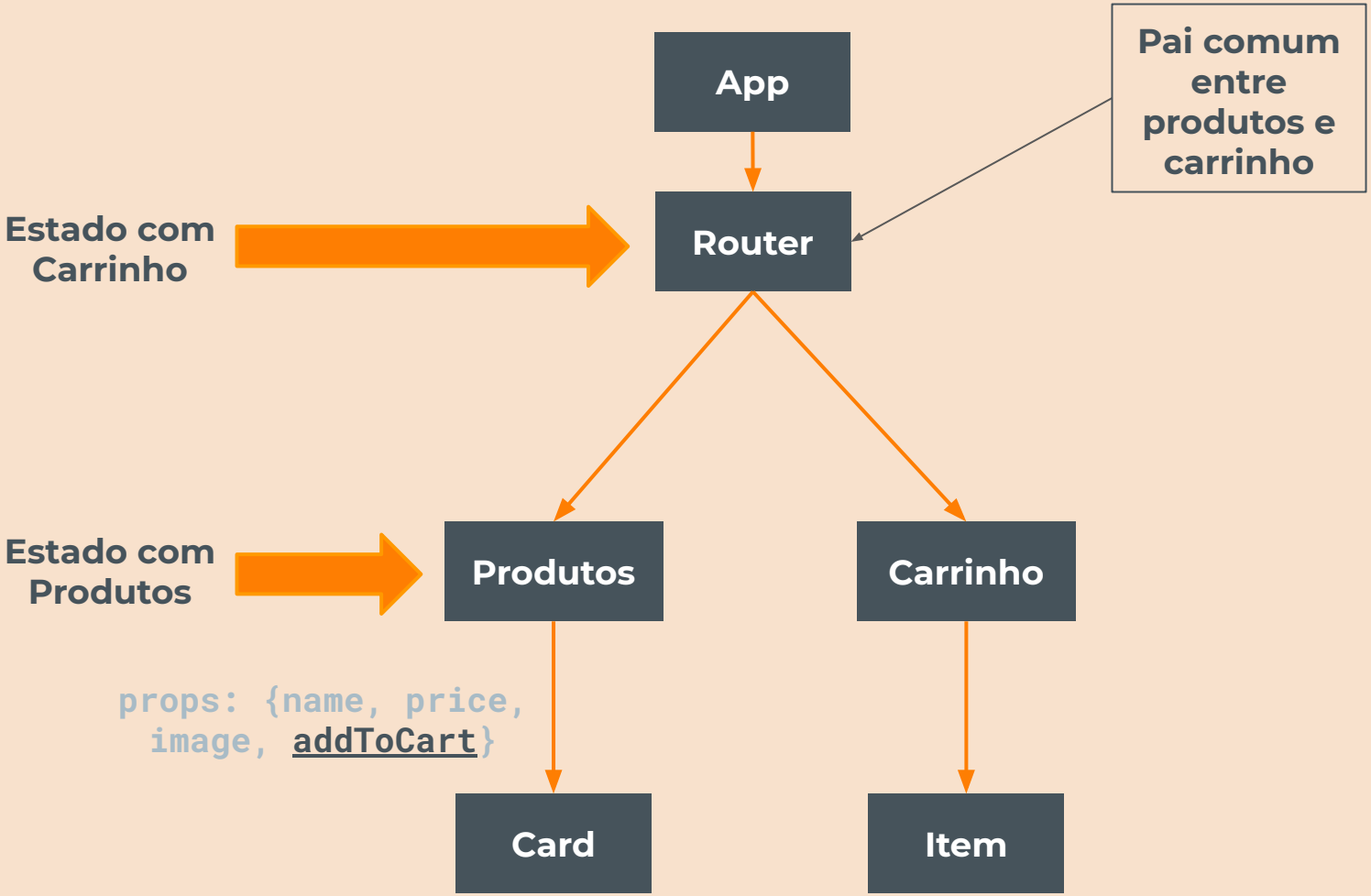
10 min

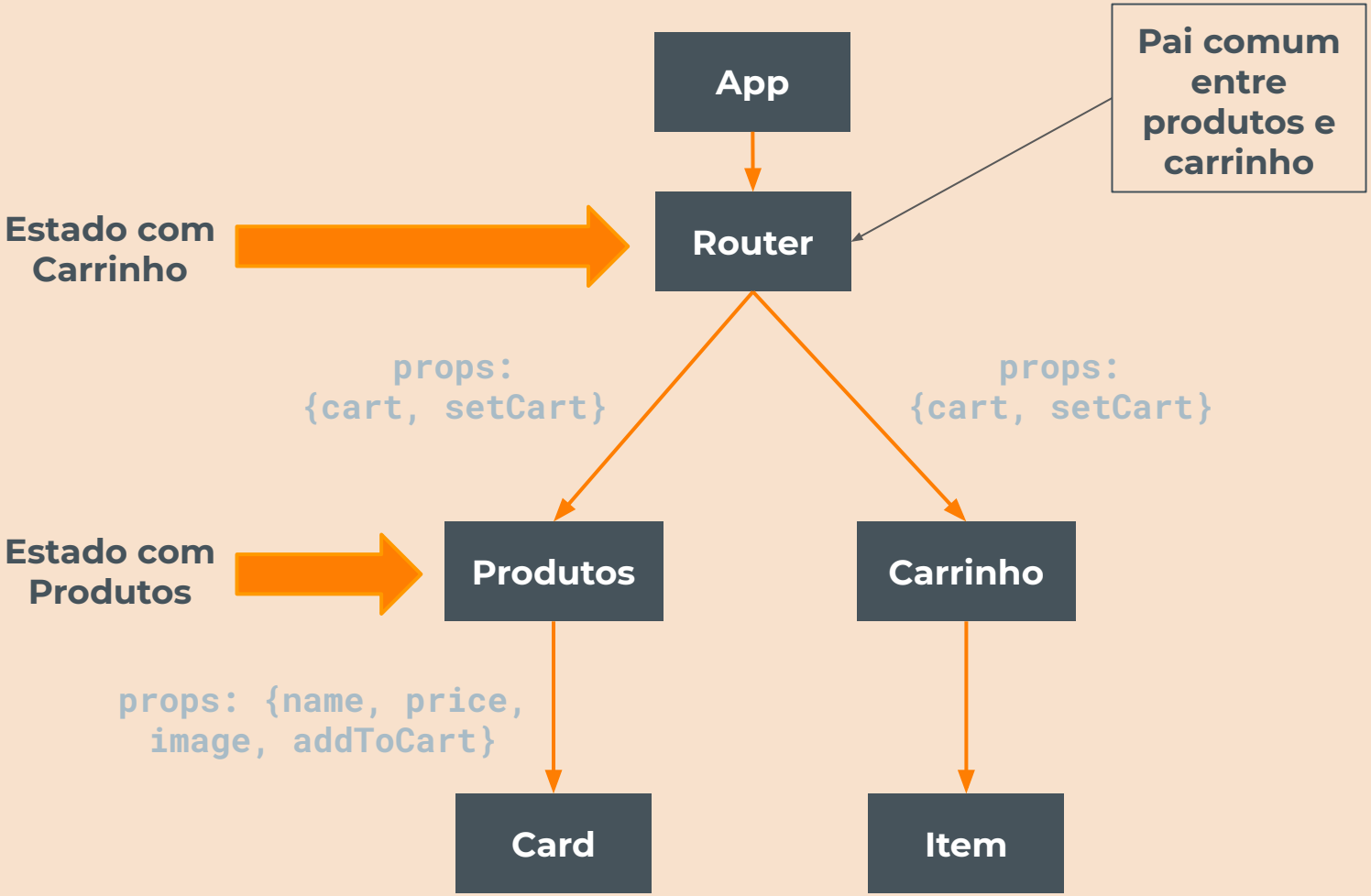

App
Router
Produtos
Carrinho
Card
Item

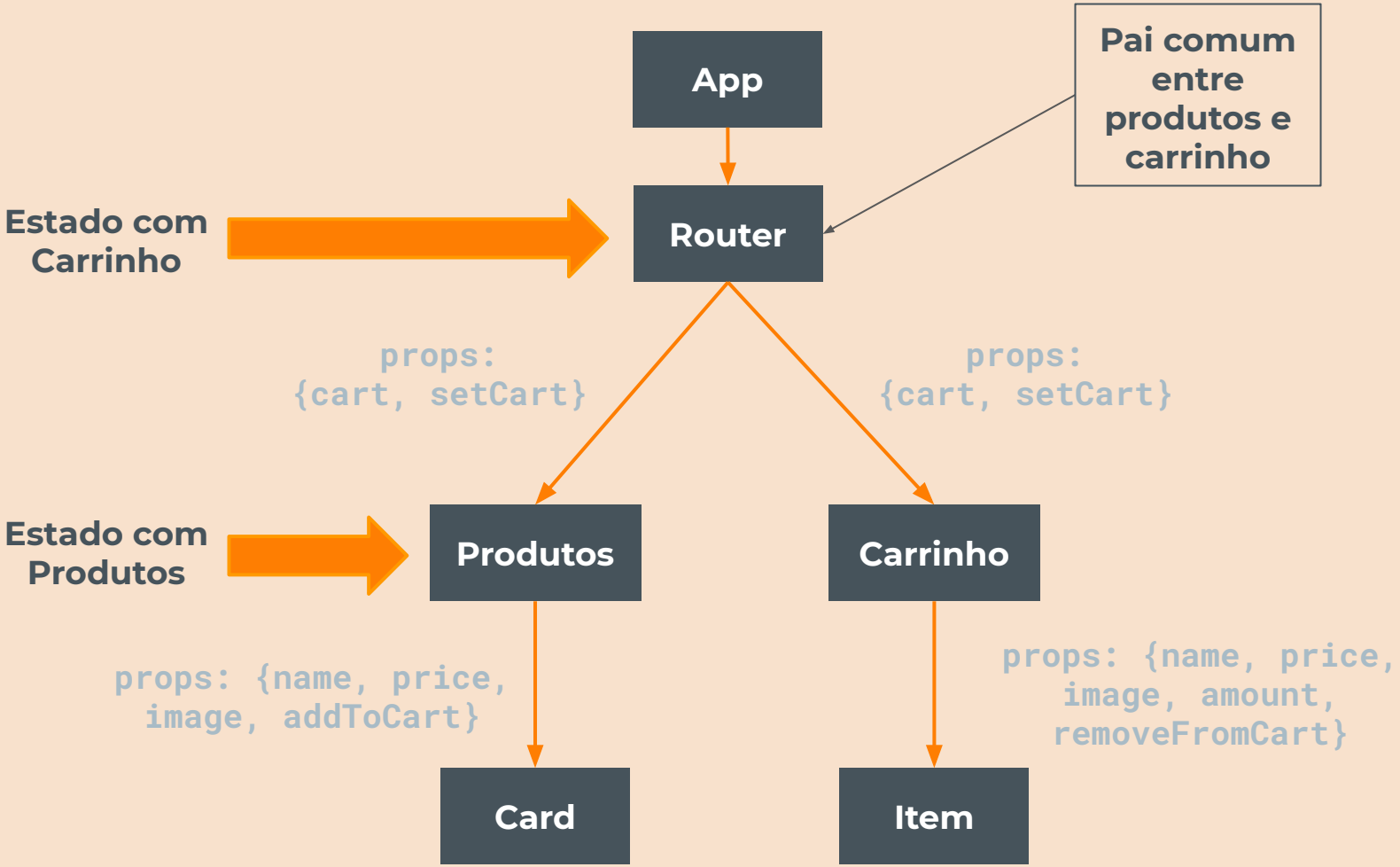

App
Router
Estado com Produtos
Produtos
Carrinho
props: {name, price, image, addToCart}
Card
Item


App
Router
Estado com Produtos
Produtos
Carrinho
props: {name, price, image, addToCart}
Card
Item


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


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


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}
Card
Item


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

Problemas com essa abordagem

Labenu_

Problemas ⚠️

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

Dados Espalhados 🎲

- No nosso exemplo do e-commerce, que é bem simples, nós temos os dados da nossa aplicação guardados em **estados**

- Esses estados estão em três componentes diferentes: **Router**, **ProductsScreen** e **CartScreen**

- Em uma aplicação real, que é muito maior, isso ficaria ainda mais espalhado e confuso de achar

Separação de Responsabilidades 🤹

- Ainda pensando no exemplo do e-commerce, a **responsabilidade** do **Router** deveria ser unicamente lidar com as rotas

- Pensando nisso, faz algum sentido o estado do carrinho estar dentro dele?

- Um bom código sempre tenta **isolar** trechos com responsabilidades diferentes

Fonte única de verdade ⛲

- Vamos pensar em um aplicativo grande para uma rede social onde o usuário troca a sua foto de perfil

- Essa foto é usada em **vários momentos diferentes**:
  - Perfil da própria pessoa
  - Lista de amigos de outras pessoas
  - Posts feitos por ela
  - ...

Fonte única de verdade ⛲

- Quando o usuário troca a foto em um lugar, **ela deve ser trocada em todos**

- Chamamos isso de **propagação de dados**

- Se por algum motivo esse gerenciamento for feito no **front**, com estados espalhados em **componentes diversos**, fica difícil saber todos os locais onde essa mudança deve ser feita

Props Drilling 🏢

- Quanto mais **profunda** fica nossa árvore de componentes, mais difícil é de realizar a **passagem de props**

- Esse problema é mais evidente quando **muitos componentes acessam a mesma informação**, ou simplesmente, estão “no caminho”.

Props Drilling - Analogia 🏢

- Você mora em um prédio com **50 apartamentos**, sendo um apartamento por andar

- O morador do 1° andar recebe todas as cartas do prédio, separa as dele e **repassa as restantes** para o morador do 2° andar, que faz a mesma coisa e passa as restantes para o 3° andar...

- Isso se repete até chegar ao 50° andar

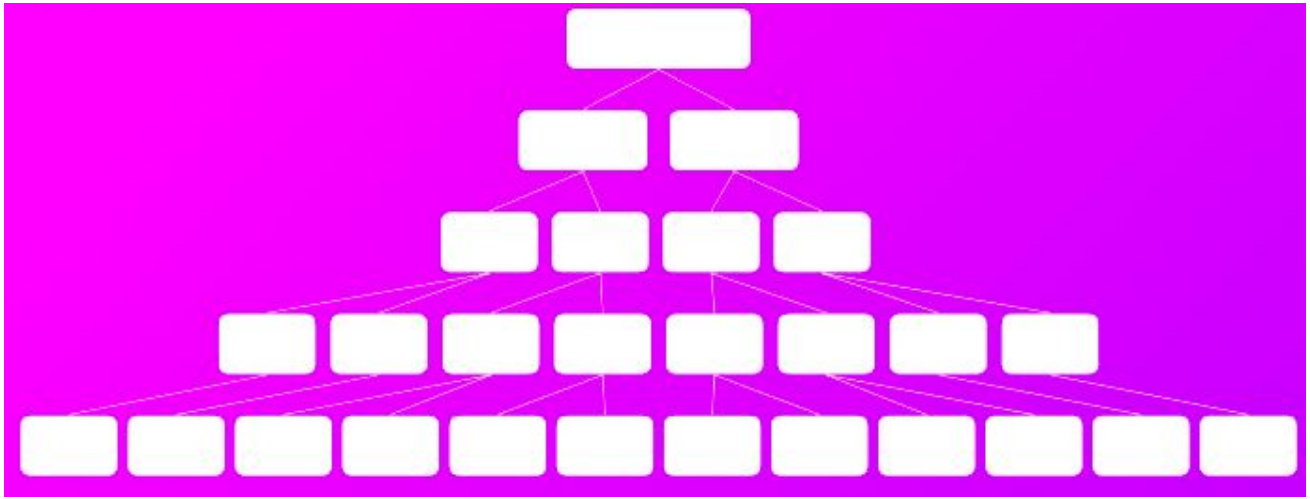
Props Drilling - Analogia 🏢

- Muitos moradores do prédio precisam lidar com cartas mesmo que não sejam deles, eles só repassam porque estão "no meio do caminho"

- Aplicando nossa analogia ao React, os apartamentos são **componentes** e as cartas são **props**

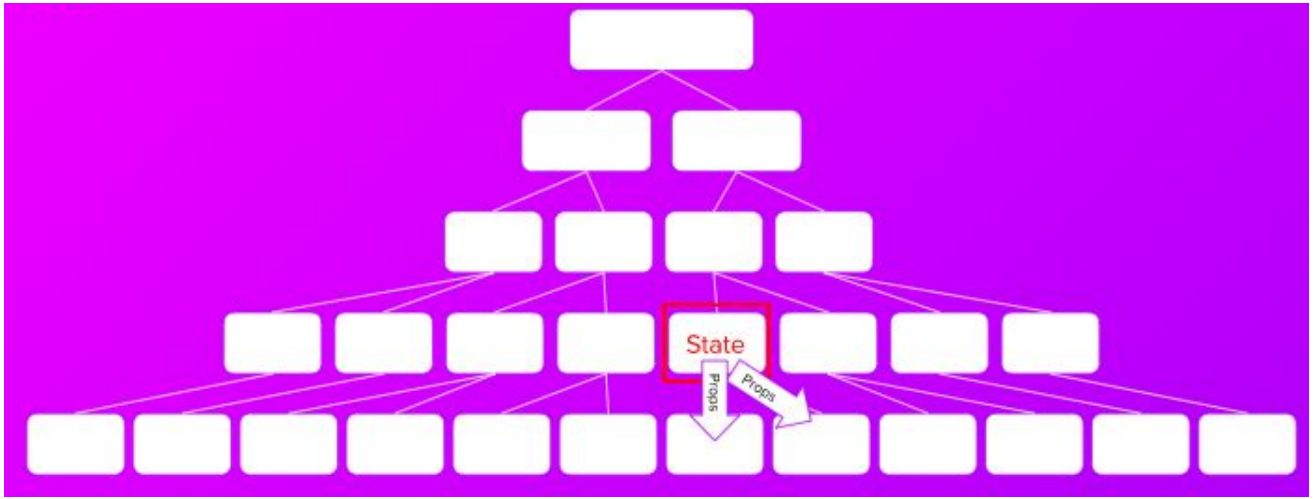
Props Drilling - Exemplo Visual 🏢

- Trazendo nossa analogia para o React, cada caixa branca representa um componente



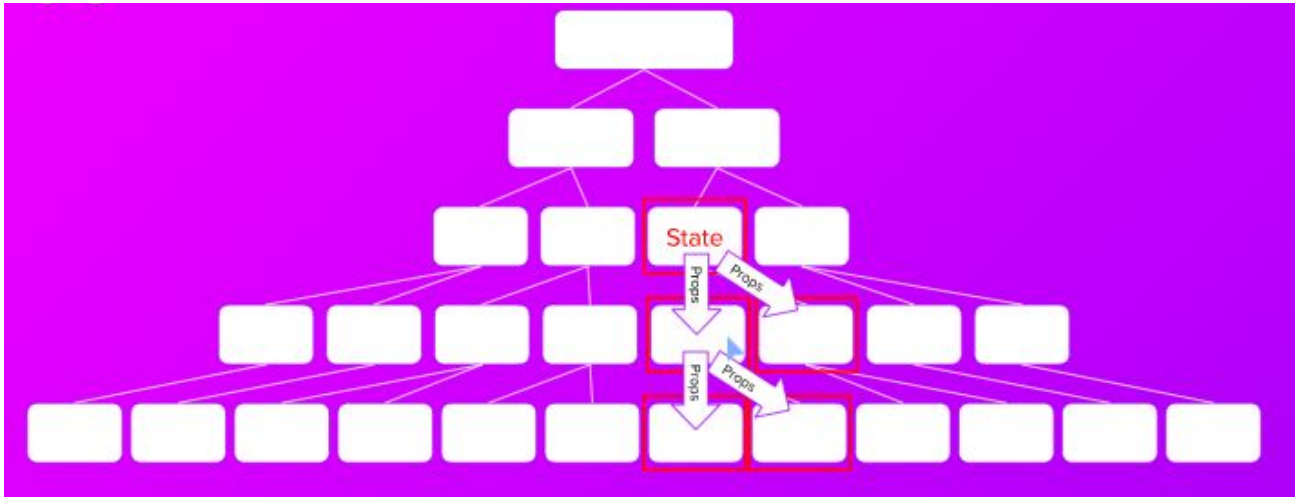
Props Drilling - Exemplo Visual 🏢

- Nesse exemplo, temos um componente pai passando props para seus 2 filhos



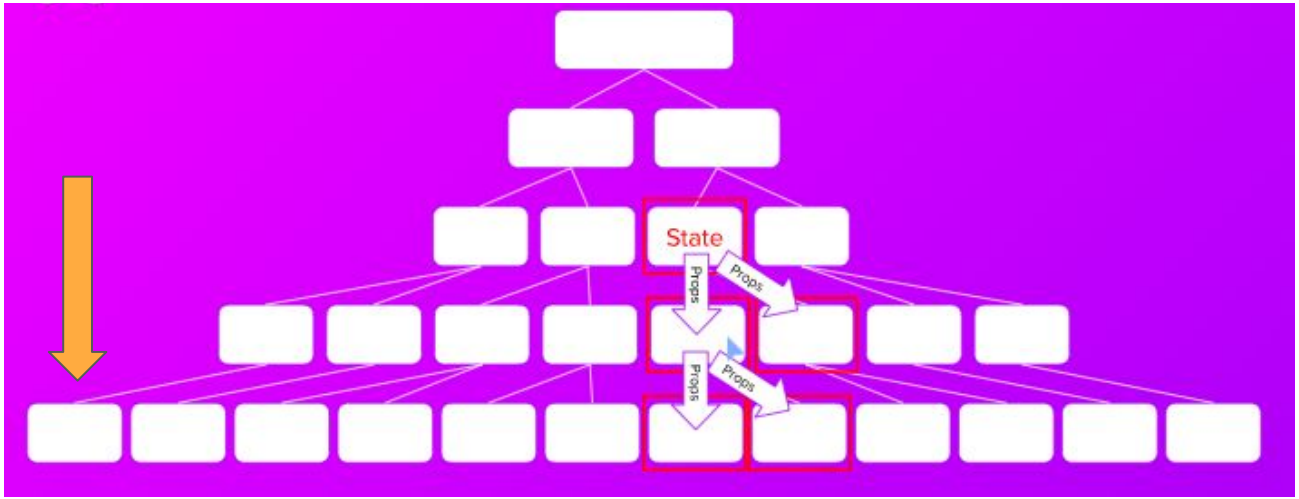
Props Drilling - Exemplo Visual 🏢

- A aplicação cresce e mais componentes precisam de dados, props vão de **pais** para **filhos e netos**



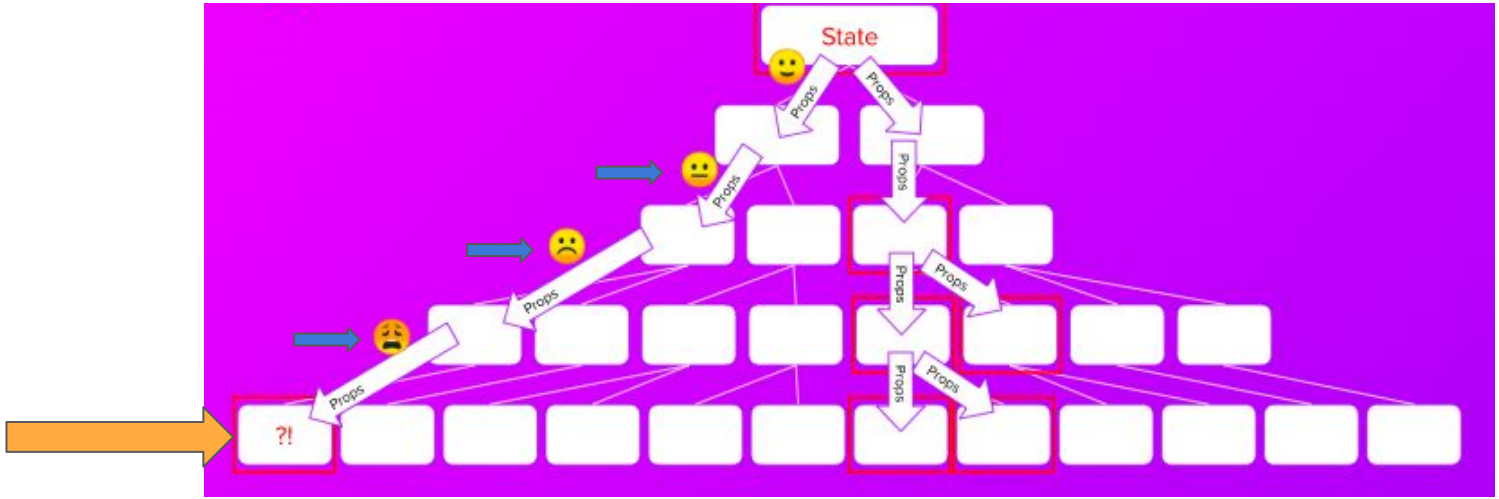
Props Drilling - Exemplo Visual 🏢

- E se quiséssemos ter acesso ao state no componente indicado?



Props Drilling - Exemplo Visual 🏢

- Teríamos que "subir" o dado para o componente pai e passá-los como prop por três componentes intermediários antes de atingir o componente final



Estado Global

Labenu_

Estado Global - O que é? 🌎

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

Estado Global 🌎

- Nossos componentes **não se comunicam mais diretamente com a API**

- Isso nos permite **isolar todos os dados** em um único local, sendo mais fácil de organizá-los e visualizá-los

- **Todos os componentes** têm acesso à essas informações de maneira **direta**, sem necessidade de passar nada por props

Estado Global 🌎

Como fazer um Estado Global?

Labenu_

Como fazer um Estado Global? 💡

- Temos duas abordagens para implementar estados globais nas nossas aplicações:
    - **Usando bibliotecas:** existem muitas bibliotecas robustas e amplamente usadas para este fim
    - **Fazendo "na mão":** criar seu estado global utilizando ferramentas nativas do React

Bibliotecas 

- Existem diversas bibliotecas para lidar com estados globais, alguns exemplos são:
  - Redux - um pouco mais complexa, cada dia menos usada mas ainda é bastante relevante
  - MobX - agnóstica à frameworks, tem sido amplamente adotada pelas empresas
  - ReactN - uso super simples mas não é tão apropriada em caso de estados muito complexos

Criando nosso Estado Global 🌎

- Para criar nosso estado global do zero com todos os requisitos que vimos anteriormente, iremos utilizar uma ferramenta nativa do React, o **Context**

- É uma das ferramentas por trás de todas as libs vistas

- Estudaremos o Context com mais detalhes na próxima aula 😄

Resumo

Labenu_

Resumo 📓

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

Resumo 📓

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

Dúvidas? 

Labenu_

Labenu_

Obrigado(a)!