

Sugestão de resolução

LabECommerce

Labenu_



O que vamos ver hoje?

- Análise do enunciado
- Preparação do ambiente de desenvolvimento
- Implementação do LabECommerce
- Finalização do projeto



Analizando o enunciado

Labenu_



Analizando o enunciado

- Quando o problema é muito complexo para resolvermos, a solução é quebrá-lo em pequenas partes.
- Pequenas partes que podem ser resolvidas por pessoas diferentes e é essa uma das vantagens em desenvolver uma aplicação em grupo.
- Vamos observar isso na proposta de hoje: O LabEcommerce



Analizando o enunciado

▼ Home

- ☐ Mostrando todos os produtos
- ☐ Deve haver alguma forma de ordenar os produtos por ordem crescente ou decrescente de preço (pode ser na home em si ou junto dos filtros)

+ :: ▼ Produtos:

- + :: ☐ Devem ter um botão que permita adicioná-los ao carrinho
- ☐ Devem exibir o nome, preço e imagem em um card

▼ Carrinho

- ☐ Mostrar todos os produtos e quantidades adicionadas
- ☐ Capacidade de remover itens do carrinho
- ☐ Mostrar o valor total do carrinho

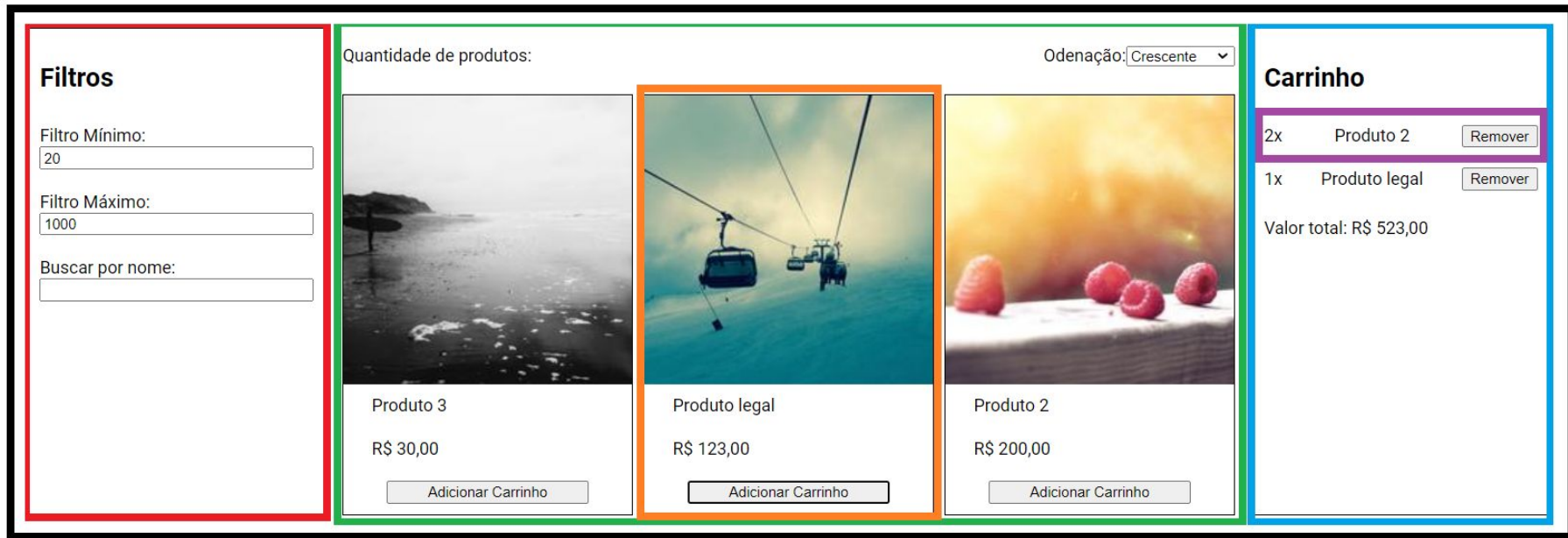
▼ Filtro

- ☐ Por valor mínimo e máximo
- ☐ Por nome do produto



Analizando o enunciado

■ App.js



Filtros



Home/produtos



Cartão dos produtos



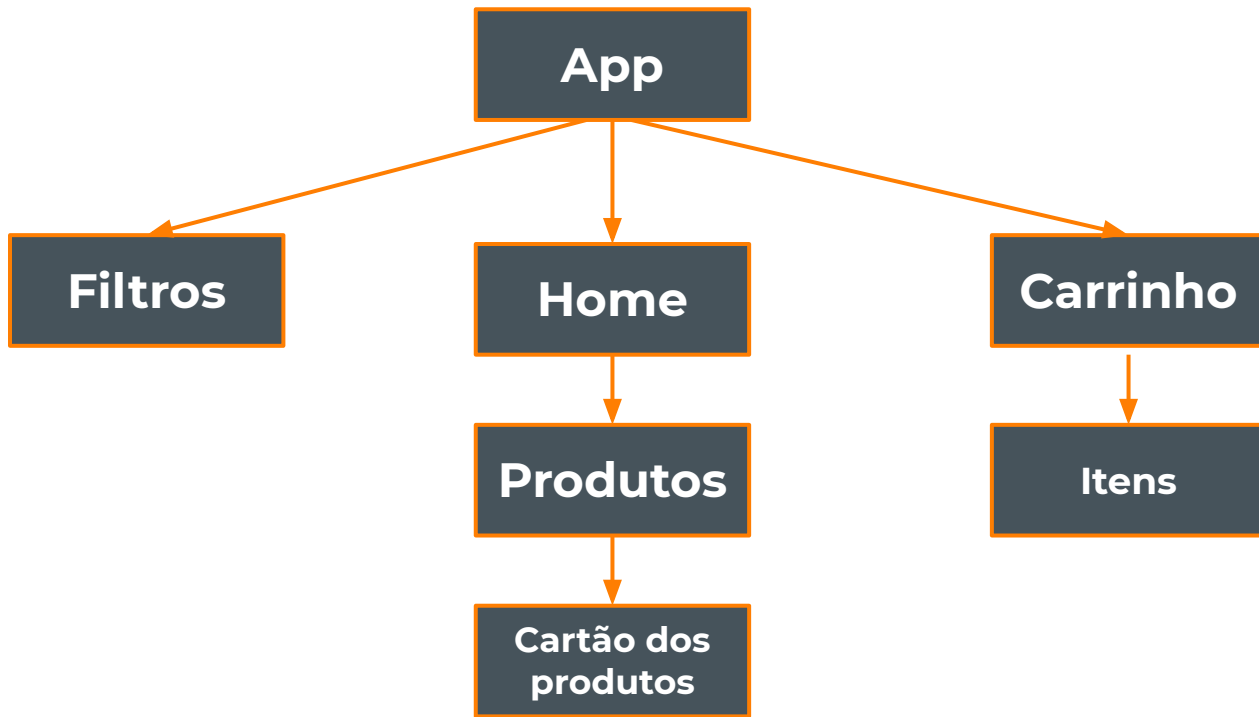
Carrinho



Itens



Analizando o enunciado



Preparação do ambiente de desenvolvimento

Labenu_



Preparando o ambiente

- Agora que temos uma ideia de como resolver o problema, vamos organizar o nosso ambiente de desenvolvimento instalando bibliotecas, extensões e estruturando pastas. Além disso definindo uma língua para o projeto
 - Língua que será usada no projeto (português, inglês...)
 - Criar projeto `npx create-react-app nome-do-programa`
 - Instalação das libs (styled-components)
 - Instalação de extensões (styled-components)

Vamos ver na prática! 



Preparando o ambiente

- Com o projeto funcionando, exclua todos os arquivos que vem por padrão do React e que não vamos precisar:
 - logo192.png, logo512.png, manifest.json, App.css, App.test.js, logo.svg, reportWebVitals.js, robots.txt e favicon.icon.
- Ao excluir todos esses arquivos a sua aplicação vai quebrar e precisaremos fazer ajustes nos arquivos com erros.

Vamos ver na prática! 



Implementação do Projeto

Labenu_



Implementação do Projeto

- Para desenvolvermos a nossa aplicação vamos seguir um padrão na construção do projeto:
 - Estruturas arquivos/pastas
 - Estilização básica para cada componente
 - Separar a estilização em outro arquivo.js
 - Implementar as funcionalidades mínimas
 - Testes



Filtros

Labenu_



Implementação - Filtro

Vamos começar pelo componente **Filtros** observando a nossa lista de tarefas encontrada no enunciado do projeto

- Filtros
 - Filtro por valor mínimo
 - Filtro por valor máximo
 - Filtro por nome

Vamos ver na prática! 



Implementação - Filtro

1 - Implementação do componente

```
1 import React from "react";
2 import { GrupoDeFiltros, Filtro } from "../estilo";
3
4 export class Filtros extends React.Component {
5   render() {
6     return (
7       <GrupoDeFiltros>
8         <h2>Filtros</h2>
9
10        <Filtro>
11          Filtro Mínimo:
12          <input value={this.props.minimo} onChange={this.props.onChangeMinimo} min={0} type="number" />
13        </Filtro>
14
15        <Filtro>
16          Filtro Máximo:
17          <input value={this.props.maximo} onChange={this.props.onChangeMaximo} min={0} type="number" />
18        </Filtro>
19
20        <Filtro>
21          Buscar por nome:
22          <input onChange={this.props.onChangeBuscaPorNome} />
23        </Filtro>
24      </GrupoDeFiltros>
25    );
26  }
27 }
28
```

2 - Estilização básica

```
1 import styled from 'styled-components'
2
3 export const GrupoDeFiltros = styled.div`
4   border: 1px solid black;
5   display: flex;
6   flex-direction: column;
7   padding: 10px;
8 `
9
10 export const Filtro = styled.label`
11   display: flex;
12   flex-direction: column;
13   padding: 10px 0;
14 `
```

Vamos ver na prática!



Implementação - Filtro

Resultado

Filtros

Filtro Mínimo:

Filtro Máximo:

Buscar por nome:



Implementação - Filtro

1- Estado

```
class App extends React.Component {  
  state = {  
    filtroMaximo: 1000,  
    filtroMinimo: 20,  
    filtroBuscaPorNome: "",  
  };  
}
```

2 - Funções para o input controlado

```
manipularValorDoFiltroMinimo = (event) => {  
  this.setState({  
    filtroMinimo: event.target.value,  
  });  
};  
  
manipularValorDoFiltroMaximo = (event) => {  
  this.setState({  
    filtroMaximo: event.target.value,  
  });  
};  
  
manipularValorDoBuscaPorNome = (event) => {  
  this.setState({  
    filtroBuscaPorNome: event.target.value,  
  });  
};
```

3 - Usando as propriedades do componente

```
<Filtros  
  minimo={this.state.filtroMinimo}  
  onChangeMinimo={this.manipularValorDoFiltroMinimo}  
  
  maximo={this.state.filtroMaximo}  
  onChangeMaximo={this.manipularValorDoFiltroMaximo}  
  
  buscaPorNome={this.state.filtroBuscaPorNome}  
  onChangeBuscaPorNome={this.manipularValorDoBuscaPorNome}  
>
```

Vamos ver na prática!



Implementação - Filtro

1 - Produtos para testar

```
export const pacoteDeProdutos = [
  {
    id: 1,
    name: 'Produto legal',
    price: 123,
    photo: 'https://picsum.photos/200/200?a=1',
  },
  {
    id: 2,
    name: 'Produto 2',
    price: 200,
    photo: 'https://picsum.photos/200/200?a=2'
  },
  {
    id: 3,
    name: 'Produto 3',
    price: 30,
    photo: 'https://picsum.photos/200/200?a=3'
  },
  {
    id: 4,
    name: 'Produto 4',
    price: 10,
    photo: 'https://picsum.photos/200/200?a=4'
  }
]
```

2 - funções de filtro

```
filtrarProdutos = () => {
  const produtoFiltradoPorNome = pacoteDeProdutos.filter((produto) => {
    return produto.name.includes(this.state.filtroBuscaPorNome);
  });

  const produtoFiltradoMaximo = produtoFiltradoPorNome.filter((produto) => {
    if(this.state.filtroMaximo){
      return produto.price <= this.state.filtroMaximo
    }else{
      return produto
    }
  });

  const produtosFiltrados = produtoFiltradoMaximo.filter((produto) => {
    if(this.state.filtroMinimo){
      return produto.price >= this.state.filtroMinimo
    }else{
      return produto
    }
  });
  return produtosFiltrados;
};
```

Vamos ver na prática!



Produtos e o Cartão de produtos

Labenu_



Implementação - Produtos

Vamos para o **Produtos** observando a nossa lista de tarefas encontrada no enunciado do projeto

- Produtos
 - Mostrar todos os produtos
 - Devem exibir o nome, preço e imagem em um card
 - Deve haver alguma forma de ordenar os produtos por ordem crescente ou decrescente de preço
 - Devem ter um botão que permita adicioná-los ao carrinho



Implementação - Produtos

1 - componente de Produtos inicial

```
1 import React from "react";
2 import { Cabecalho } from "../estilizacaoDosProdutos";
3
4 export class Produtos extends React.Component {
5   render() {
6
7     return (
8       <div>
9         <Cabecalho>
10         <p>Quantidade de produtos: {this.props.quantidade}</p>
11         <label>
12           Odenação:
13           <select onChange={this.props.onChangeCabecalho}>
14             <option value={"Crescente"}>Crescente</option>
15             <option value={"Decrescente"}>Decrescente</option>
16           </select>
17         </label>
18       </Cabecalho>
19     </div>
20   );
21 }
22 }
23
```

2 - Estilização básica

```
import styled from "styled-components";

export const Cabecalho = styled.div`
  display: flex;
  justify-content: space-between;
  align-items: center;
`;
```

Vamos ver na prática!



Implementação - Produtos

Resultado

Quantidade de produtos:

Odenação: Crescente ▼

Vamos ver na prática! 



Implementação - Cartão dos Produtos

1 - componente Cartão dos Produtos

```
import React from "react";
import { Cartao, TextoDoCartao } from "../estiloDoCartao";

export class CartaoDosProdutos extends React.Component {
  render() {
    return (
      <Cartao>
        <img src={this.props.imagem} alt={this.props.alt} />
        <TextoDoCartao>
          <p>{this.props.nome}</p>
          <p>R$ {this.props.valor},00</p>
          <button onClick = {this.props.onClick}>Adicionar Carrinho</button>
        </TextoDoCartao>
      </Cartao>
    );
  }
}
```

2 - Estilização básica

```
import styled from "styled-components";

export const Cartao = styled.div`
  border: 1px solid black;
  display: flex;
  flex-direction: column;
`;

export const TextoDoCartao = styled.div`
  display: flex;
  flex-direction: column;

  p {
    margin: 0 16px;
    padding: 10px;
  }

  button {
    margin: 10px;
    align-self: center;
    width: 70%;
  }
`;
```

Vamos ver na prática!



Implementação - Produtos

- Por último chamamos o componente **CartaoDosProdutos** no componente **Produtos** para testar

```
import React from "react";
import { CartaoDosProdutos } from "../CartaoDoProduto/CartaoDosProdutos";
import { Cabecalho, GrupoDeCartoes } from "../estilizacaoDosProdutos";

export class Produtos extends React.Component {
  render() {
    return (
      <div>
        <Cabecalho>
          <p>Quantidade de produtos: {this.props.quantidade}</p>
          <label>
            Ordenação:
            <select onChange={this.props.onChangeCabecalho}>
              <option value={"Crescente"}>Crescente</option>
              <option value={"Decrescente"}>Decrescente</option>
            </select>
          </label>
        </Cabecalho>
        <GrupoDeCartoes>
          <CartaoDosProdutos
            nome = {"Produto 1"}
            alt = {"Produto 1"}
            imagem={"https://picsum.photos/200/200?a=2"}
            valor={200}
            onClick={() => this.props.onClick()}
          />
        </GrupoDeCartoes>
      </div>
    );
  }
}
```

Estilização

```
import styled from "styled-components";

export const Cabecalho = styled.div`
  display: flex;
  justify-content: space-between;
  align-items: center;
`;

export const GrupoDeCartoes = styled.div`
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  gap: 10px;
  margin-top: 10px;
`;
```

Vamos ver na prática!




Implementação - Produtos

Resultado

Quantidade de produtos:

Odenação: Crescente ▼



Produto 1

R\$ 200,00

Adicionar Carrinho

Vamos ver na prática! 



Implementação - Produtos

- Então fazemos o **map()** usando o componente de **CartaoDosProdutos**

Mapeando os produtos

```
const produtosMapeados =  
  pacoteDeProdutos &&  
  pacoteDeProdutos.map((produto) => {  
    return (  
      <CartaoDosProdutos key={produto.id}  
        nome = {produto.name}  
        alt = {produto.name}  
        imagem={produto.photo}  
        valor={produto.price}  
        onClick={() => this.props.onClick(produto)}  
      );  
    );  
  });
```

Produtos serão renderizados

```
<GrupoDeCartoes>{produtosMapeados}</GrupoDeCartoes>
```

```
import React from "react";  
import { pacoteDeProdutos } from "../../pacoteDeProdutos";  
import { CartaoDosProdutos } from "../CartaoDoProduto/CartaoDosProdutos";  
import { Cabecalho, GrupoDeCartoes } from "../estilizacaoDosProdutos";  
  
export class Produtos extends React.Component {  
  render() {  
    const produtosMapeados =  
      pacoteDeProdutos &&  
      pacoteDeProdutos.map((produto) => {  
        return (  
          <CartaoDosProdutos key={produto.id}  
            nome = {produto.name}  
            alt = {produto.name}  
            imagem={produto.photo}  
            valor={produto.price}  
            onClick={() => this.props.onClick(produto)}  
          );  
        });  
      });  
    return (  
      <div>  
        <Cabecalho>  
          <p>Quantidade de produtos: {this.props.quantidade}</p>  
          <label>  
            Ordenação:  
            <select onChange={this.props.onChangeCabecalho}>  
              <option value={"Crescente"}>Crescente</option>  
              <option value={"Decrescente"}>Decrescente</option>  
            </select>  
          </label>  
        </Cabecalho>  
        <GrupoDeCartoes>{produtosMapeados}</GrupoDeCartoes>  
      </div>  
    );  
  }  
}
```

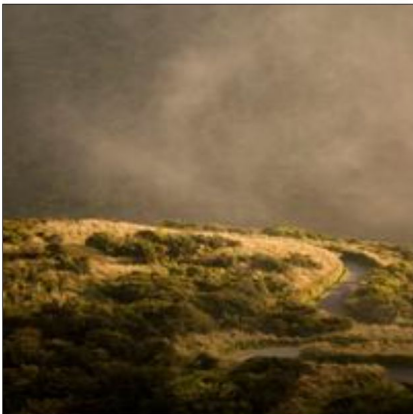

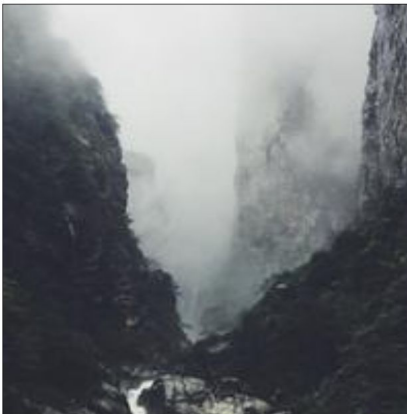
Vamos ver na prática!



Implementação - Produtos

Resultado

Quantidade de produtos: Ordenação: Crescente ▾

		
Produto legal R\$ 123,00 <button>Adicionar Carrinho</button>	Produto 2 R\$ 200,00 <button>Adicionar Carrinho</button>	Produto 3 R\$ 30,00 <button>Adicionar Carrinho</button>

Vamos ver na prática! 



Ordenação

Labenu_



Implementação - ordenação

- Enfim, vamos mudar um pouco o nosso código para começarmos a implementar a ordenação
- Para isso usaremos a função **sort()** no array de produtos **enviando por props** pelo **App.js**

```
const produtosOrdenados =  
  this.props.produtos &&  
  this.props.produtos.sort((a, b) => {  
    return this.props.ordenacao === "Crescente"  
      ? a.price - b.price  
      : b.price - a.price;  
  });
```

```
import React from "react";  
import { CartaoDoProduto } from "../CartaoDoProduto/CartaoDoProduto";  
import { Cabecalho, GrupoDeCartoes } from "../estilizacaoDosProdutos";  
  
export class Produtos extends React.Component {  
  render() {  
  
    const produtosOrdenados =  
      this.props.produtos &&  
      this.props.produtos.sort((a, b) => {  
        return this.props.ordenacao === "Crescente"  
          ? a.price - b.price  
          : b.price - a.price;  
      });  
  
    const produtosMapeados =  
      produtosOrdenados &&  
      produtosOrdenados.map((produto) => {  
        return (  
          <CartaoDoProdutos key={produto.id}  
            nome={produto.name}  
            alt={produto.name}  
            imagem={produto.photo}  
            valor={produto.price}  
            onClick={() => this.props.onClick(produto)}  
          />  
        );  
      });  
  
    return (  
      <div>  
        <Cabecalho>  
          <p>Quantidade de produtos: {this.props.quantidade}</p>  
          <label>  
            Ordenação:  
            <select onChange={this.props.onChangeCabecalho}>  
              <option value="Crescente">Crescente</option>  
              <option value="Decrescente">Decrescente</option>  
            </select>  
          </label>  
        </Cabecalho>  
        <GrupoDeCartoes>{produtosMapeados}</GrupoDeCartoes>  
      </div>  
    );  
  }  
}
```

Vamos ver na prática!



Implementação - ordenação

- **Dentro do App.js** criamos um estado **ordenacao** e enviamos os produtos **por props**.

```
class App extends React.Component {  
  state = {  
    ordenacao: "Crescente",  
  };  
}
```

```
<Produtos  
  onChangeCabecalho={this.ordenarProdutos}  
  produtos={CartaoDosProdutos}  
  ordenacao={this.state.ordenacao}  
/>
```

Pegamos as opções de ordenação pela propriedade **onChangeCabecalho**, utilizando a função **ordenarProdutos** e alteramos a opção/ordenação no **estado com o setState()**

```
<Produtos  
  onChangeCabecalho={this.ordenarProdutos}  
  produtos={CartaoDosProdutos}  
  ordenacao={this.state.ordenacao}  
/>
```

```
ordenarProdutos = (event) => {  
  this.setState({  
    ordenacao: event.target.value,  
  });  
};
```

Vamos ver na prática! 



Função do botão “adicionar Carrinho”

Labenu_



Implementação - Botão/função

- Por último vamos deixar a **função do botão “adicionar carrinho”** pronta para ser utilizada no **App.js**

Vamos usar a propriedade **onClick** passado via **props** do Componente **Produtos**


```
<Produtos  
  onChangeCabecalho={this.ordenarProdutos}  
  produtos={CartaoDosProdutos}  
  onClick={this.adicionarProdutoNoCarrinho}  
  ordenacao={this.state.ordenacao}  
/>
```

E adicionar a função **adicionarProdutoNoCarrinho** para testar

```
adicionarProdutoNoCarrinho = (produto) => {  
  console.log(produto)  
};
```

A ideia é que quando você clicar no botão será passado as informações do produto no **console**

```
► {id: 4, name: "Produto 4", price: 10, photo: "https://picsum.photos/200/200?a=4"}
```

Vamos ver na prática! 



Carrinho

Labenu_



Implementação - Carrinho

Vamos para o **Carrinho** observando a nossa lista de tarefas encontrada no enunciado do projeto

- Carrinho
 - Mostrar todos os produtos e quantidades adicionais
 - Capacidade de remover itens do carrinho
 - mostrar o valor total do carrinho



Implementação - carrinho

1 - implementação básica do **Carrinho**

```
import React from "react";
import { ConjuntoDoCarrinho } from "../estiloDoCarrinho";

export class Carrinho extends React.Component {
  render() {
    return (
      <ConjuntoDoCarrinho>
        <h2>Carrinho</h2>
        <p>Valor total: R$ {this.props.valorTotal},00</p>
      </ConjuntoDoCarrinho>
    );
  }
}
```

2 - estilização básica

```
import styled from "styled-components";

export const ConjuntoDoCarrinho = styled.div`
  border: 1px solid black;
  display: flex;
  flex-direction: column;
  padding: 10px;
```

Vamos ver na prática! 




Implementação - carrinho

Resultado

Carrinho

Valor total: R\$,00

Vamos ver na prática! 



Implementação - carrinho

1 - implementação básica do **Item**

```
import React from "react";
import { ConjutoDeItens } from "../estiloDoItem";

export class Item extends React.Component {
  render() {
    return <ConjutoDeItens>
      <p>{this.props.quantidade}x</p>
      <p>{this.props.nome}</p>
      <button onClick={this.props.onClick}>Remover</button>
    </ConjutoDeItens>
  }
}
```

2 - estilização básica

```
import styled from "styled-components";

export const ConjutoDeItens = styled.div`
  display: flex;
  align-items: center;
  justify-content: space-between;
  p {
    margin: 10px 0px;
  }
`;
```

Vamos ver na prática!



Implementação - carrinho

Adicionamos o **Item** no **Carrinho**, utilizando as suas propriedades para testarmos o código.

```
import React from "react";
import { ConjuntoDoCarrinho } from "../estiloDoCarrinho";
import { Item } from "../Item";

export class Carrinho extends React.Component {
  render() {
    return (
      <ConjuntoDoCarrinho>
        <h2>Carrinho</h2>
        <div>
          <Item
            nome={"produto"}
            quantidade={"1"}
            onClick={() => this.props.onClick()}
          />
          <Item
            nome={"produto"}
            quantidade={"1"}
            onClick={() => this.props.onClick()}
          />
        </div>
        <p>Valor total: R$ {this.props.valorTotal},00</p>
      </ConjuntoDoCarrinho>
    );
  }
}
```

Carrinho

1x	produto	<button>Remover</button>
1x	produto	<button>Remover</button>
Valor total: R\$,00		

Vamos ver na prática!



Implementação - carrinho

Agora que já testamos o **Item** podemos fazer o **map()** dos itens que serão enviados do **App.js**

```
const itens =  
  this.props.itensDoCarrinho &&  
  this.props.itensDoCarrinho.map(item => {  
    return <Item  
      nome={item.name}  
      quantidade={item.quantidade}  
      onClick={() => this.props.onClick(item)}  
    />;  
  });
```

```
import React from "react";  
import { ConjuntoDoCarrinho } from "../estiloDoCarrinho";  
import { Item } from "../Item";  
  
export class Carrinho extends React.Component {  
  render() {  
    const itens =  
      this.props.itensDoCarrinho &&  
      this.props.itensDoCarrinho.map(item => {  
        return <Item  
          nome={item.name}  
          quantidade={item.quantidade}  
          onClick={() => this.props.onClick(item)}  
        />;  
      });  
  
    return (  
      <ConjuntoDoCarrinho>  
        <h2>Carrinho</h2>  
        <div>{itens}</div>  
        <p>Valor total: R$ {this.props.valorTotal},00</p>  
      </ConjuntoDoCarrinho>  
    );  
  }  
}
```

Vamos ver na prática!



Implementação - carrinho

No **App.js** criamos 2 estados, **carrinho** e **valorTotal**, que serão passados para o componente **Carrinho** utilizando as propriedades passadas por **props**.

```
class App extends React.Component {  
  state = {  
    carrinho: [],  
    valorTotal: 0,  
  };  
}
```

```
<Carrinho  
  valorTotal={this.state.valorTotal}  
  itensDoCarrinho={this.state.carrinho}  
  onClick={this.removerItemDoCarrinho}  
>
```

```
import React from "react";  
import { Carrinho } from "../Components/Carrinho/Carrinho";  
  
class App extends React.Component {  
  state = {  
    carrinho: [],  
    valorTotal: 0,  
  };  
  
  render() {  
    return (  
      <div>  
        <Carrinho  
          valorTotal={this.state.valorTotal}  
          itensDoCarrinho={this.state.carrinho}  
          onClick={this.removerItemDoCarrinho}  
        />  
      </div>  
    );  
  }  
}  
  
export default App;
```

Vamos ver na prática!



Implementação - carrinho

E deixamos pronta a função de remover item do **Carrinho** para ser usado quando todos os componentes estiverem presentes.

```
<Carrinho  
  valorTotal={this.state.valorTotal}  
  itensDoCarrinho={this.state.carrinho}  
  onClick={this.removerItemDoCarrinho}  
/>
```

```
class App extends React.Component {  
  state = {  
    carrinho: [],  
    valorTotal: 0,  
  };  
  
  removerItemDoCarrinho = (itemParaRemover) => {  
    console.log(itemParaRemover);  
  };  
  
  render() {  
    return (  
      <div>  
        <Carrinho  
          valorTotal={this.state.valorTotal}  
          itensDoCarrinho={this.state.carrinho}  
          onClick={this.removerItemDoCarrinho}  
        />  
      </div>  
    );  
  }  
}  
  
export default App;
```

Vamos ver na prática!



Implementação - carrinho

Podemos testar a função **removerItemDoCarrinho** colocando itens fixo no array do estado "carrinho" e clicando no botão.

```
carrinho: [  
  {  
    id: 1,  
    name: "Produto legal",  
    price: 123,  
    photo: "https://picsum.photos/200/200?a=1",  
    quantidade: 1  
  },  
  {  
    id: 2,  
    name: "Produto 2",  
    price: 200,  
    photo: "https://picsum.photos/200/200?a=2",  
    quantidade: 2  
  }  
]
```

veja o resultado no **console**

```
► {id: 1, name: "Produto legal", price: 123, photo: "https://picsum.photos/200/200?a=1", quantidade: 1}
```

Vamos ver na prática!



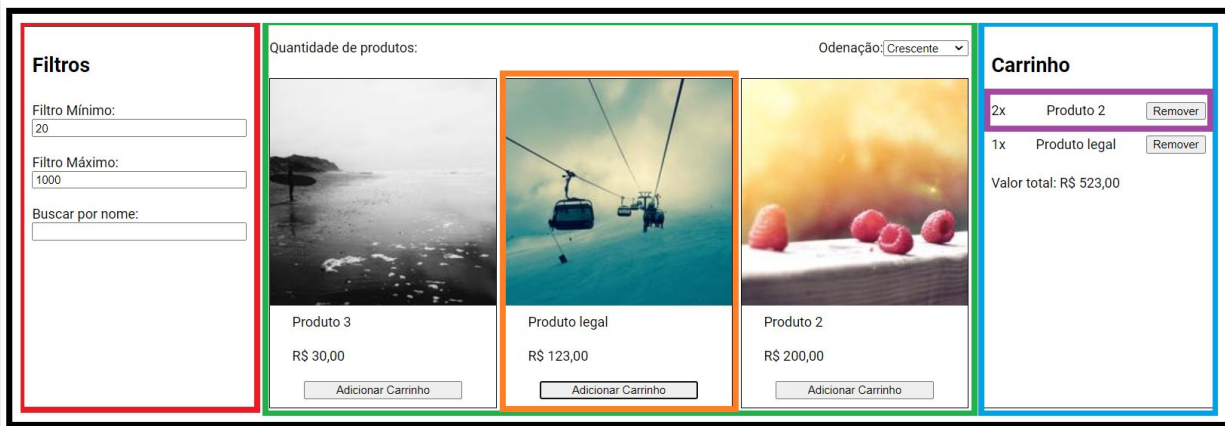
Finalizando o projeto: Unindo os Componentes

Labenu_



Implementação - unindo componentes

- Agora é o momento de todos os componentes trabalharem juntos, cada um com as suas funcionalidades, mas complementando um aos outros.



Vamos ver na prática!

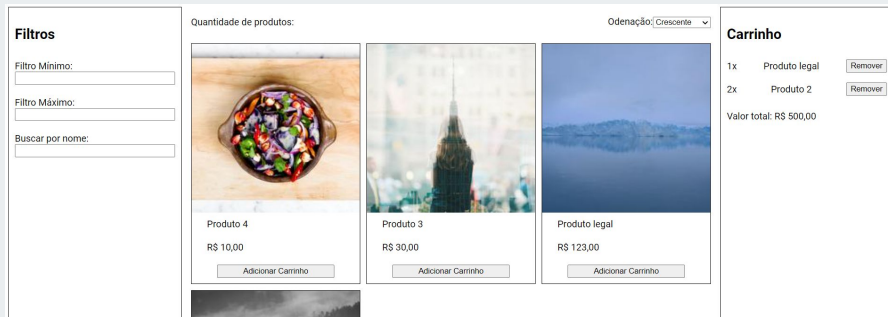


Implementação - unindo componentes

Começaremos importamos todos os componentes e em seguida estilizando o **App.js**. Enfim utilizamos as implementações que fizemos de cada componente.

```
render() {  
  const produtosFiltrados = this.filtrarProdutos();  
  
  return (  
    <ConjuntoDeComponentes>  
      <Filtros  
        minimo={this.state.filtroMinimo}  
        onChangeMinimo={this.manipularValorDoFiltroMinimo}  
        maximo={this.state.filtroMaximo}  
        onChangeMaximo={this.manipularValorDoFiltroMaximo}  
        buscaPorNome={this.state.filtroBuscaPorNome}  
        onChangeBuscaPorNome={this.manipularValorDoBuscaPorNome}  
      />  
  
      <Produtos  
        onChangeCabecalho={this.ordenarProdutos}  
        produtos={produtosFiltrados}  
        onClick={this.adicionarProdutoNoCarrinho}  
        ordenacao={this.state.ordenacao}  
      />  
  
      <Carrinho  
        valorTotal={this.state.valorTotal}  
        itensDoCarrinho={this.state.carrinho}  
        onClick={this.removerItemDoCarrinho}  
      />  
    </ConjuntoDeComponentes>  
  );  
}
```

```
export const ConjuntoDeComponentes = styled.div`  
  display: grid;  
  grid-template-columns: 1fr 3fr 1fr;  
  gap: 16px;  
`;
```



Vamos ver na prática!





Vamos unir as funcionalidades

- Filtrar os produtos
- Adicionar itens para o Carrinho
- Remover itens do Carrinho



Filtrar produtos

Labenu_



Implementação - Filtrar produtos

Vamos usar o componente **Filtros** com o componente de **Produtos**:

1 - A **propriedade produtos** é responsável por receber os produtos e renderizar na tela do componente **Produtos**

3- O mesmo acontece com a **propriedade quantidade** de **Produtos**, observe.

```
<Produtos
  quantidade={produtosFiltrados.length}
  onChangeCabecalho={this.ordenarProdutos}
  ordenacao={this.state.ordenacao}
  produtos={produtosFiltrados}
  onClick={this.adicionarProdutoNoCarrinho}
/>
```

2 - é só inserir a variável **produtosFiltrados**, retorno da função **filtrarProdutos**, e ele deve renderizar os produtos já filtrados. Teste o resultado.

```
render() {
  const produtosFiltrados = this.filtrarProdutos();

  return (
    <ConjuntoDeComponentes>
      <Filtros
        minimo={this.state.filtroMinimo}
        maximo={this.state.filtroMaximo}
        buscaPorNome={this.state.filtroBuscaPorNome}
        onChangeMinimo={this.manipularValorDoFiltroMinimo}
        onChangeMaximo={this.manipularValorDoFiltroMaximo}
        onChangeBuscaPorNome={this.manipularValorDoFiltroBuscaPorNome}
      />
      <Produtos
        quantidade={produtosFiltrados.length}
        onChangeCabecalho={this.ordenarProdutos}
        ordenacao={this.state.ordenacao}
        produtos={produtosFiltrados}
        onClick={this.adicionarProdutoNoCarrinho}
      />
      <Carrinho
        carrinho={this.state.carrinho}
        valorTotal={this.state.valorTotal}
        removerItemDoCarrinho={this.removerItemDoCarrinho}
      />
    </ConjuntoDeComponentes>
  );
}
```



Adicionar itens no Carrinho

Labenu_



Implementação - Adicionar itens no carrinho

Vamos usar o componente **Produtos** com o componente de **Carrinho**:

1 - Implemente a função **adicionarProdutoNoCarrinho** no App.js para adicionar itens no **Carrinho**

```
adicionarProdutoNoCarrinho = (produto) => {  
  const produtoNoCarrinho = this.state.carrinho.filter((item) => {  
    if (item.id === produto.id) {  
      return item;  
    } else {  
      return false;  
    }  
  });  
  
  if (produtoNoCarrinho.length === 0) {  
    produto.quantidade = 1;  
    const novoCarrinho = [produto, ...this.state.carrinho];  
    this.setState({  
      carrinho: novoCarrinho,  
    });  
  } else {  
    const novoCarrinho = this.state.carrinho.map((item) => {  
      if (produto.id === item.id) {  
        return { ...item, quantidade: item.quantidade + 1 };  
      } else {  
        return item;  
      }  
    });  
  
    this.setState({  
      carrinho: novoCarrinho,  
    });  
  }  
};
```

2- conferimos se o produto clicado está no **estado do carrinho** e se não estiver, adicionamos 1 na **propriedade quantidade** do produto.

```
const produtoNoCarrinho = this.state.carrinho.filter(item => {  
  if (item.id === produto.id) {  
    return item;  
  }  
});  
  
if (produtoNoCarrinho.length === 0) {  
  produto.quantidade = 1;
```

3 - Em seguida copiamos o estado do carrinho e o novo produto e os adicionamos em **novoCarrinho**. Por último alteramos o estado com o **novoCarrinho**.

```
const novoCarrinho = [produto, ...this.state.carrinho];  
this.setState({  
  carrinho: novoCarrinho,  
});
```



Implementação - Adicionar itens no carrinho

Implementação da função **adicionarProdutoNoCarrinho** no App.js para adicionar itens no Carrinho

```
adicionarProdutoNoCarrinho = (produto) => {  
  const produtoNoCarrinho = this.state.carrinho.filter((item) => {  
    if (item.id === produto.id) {  
      return item;  
    } else {  
      return false;  
    }  
  });  
  
  if (produtoNoCarrinho.length === 0) {  
    produto.quantidade = 1;  
    const novoCarrinho = [produto, ...this.state.carrinho];  
    this.setState({  
      carrinho: novoCarrinho,  
    });  
  } else {  
    const novoCarrinho = this.state.carrinho.map((item) => {  
      if (produto.id === item.id) {  
        return { ...item, quantidade: item.quantidade + 1 };  
      } else {  
        return item;  
      }  
    });  
  
    this.setState({  
      carrinho: novoCarrinho,  
    });  
  }  
};
```

4- se houver item no estado do carrinho, então: será comparado o produto clicado com o **estado do carrinho**.

```
} else {  
  const novoCarrinho = this.state.carrinho.map((item) => {  
    if (produto.id === item.id && item.quantidade >= 1) {
```

5 - Caso sejam iguais, será copiado todo o objeto e a **propriedade quantidade** será alterada, recebendo +1.

```
    return { ...item, quantidade: item.quantidade + 1 };  
  } else {  
    return item;  
  }  
};
```

No final alteramos o **estado do carrinho** com o **novoCarrinho**.

```
this.setState({  
  carrinho: novoCarrinho,  
});
```



Remover itens do Carrinho

Labenu_



Implementação - Remover itens do carrinho

Vamos usar o componente de **Carrinho**:

1 - Implemente a função **removerProdutoDoCarrinho** no App.js para adicionar itens no **Carrinho**

```
removerItemDoCarrinho = (itemParaRemover) => {  
  if (itemParaRemover.quantidade === 1) {  
    const novoCarrinho = this.state.carrinho.filter((item) => {  
      if (item.id !== itemParaRemover.id) {  
        return item;  
      } else {  
        return false  
      }  
    });  
    this.setState({  
      carrinho: novoCarrinho,  
    });  
  } else {  
    const novoCarrinho = this.state.carrinho.map((item) => {  
      if (itemParaRemover.id === item.id && item.quantidade >= 1) {  
        return { ...item, quantidade: item.quantidade - 1 };  
      } else {  
        return item;  
      }  
    });  
    this.setState({  
      carrinho: novoCarrinho,  
    });  
  }  
};
```

2- Se a **quantidade** do produto for igual a 1, então removemos totalmente do **carrinho**

```
if (itemParaRemover.quantidade === 1) {  
  const novoCarrinho = this.state.carrinho.filter((item) => {  
    if (item.id !== itemParaRemover.id) {  
      return item;  
    } else {  
      return false  
    }  
  });  
  this.setState({  
    carrinho: novoCarrinho,  
  });  
};
```

3 - Se não, copie o produto inteiro e retire 1 da propriedade **quantidade**.

```
} else {  
  const novoCarrinho = this.state.carrinho.map((item) => {  
    if (itemParaRemover.id === item.id && item.quantidade >= 1) {  
      return { ...item, quantidade: item.quantidade - 1 };  
    } else {  
      return item;  
    }  
  });  
  this.setState({  
    carrinho: novoCarrinho,  
  });  
};
```



Somar/subtrair o valor total do Carrinho

Labenu_



Implementação - Somar/subtrair o valor total

Vamos usar o componente de **Carrinho**:

1 - Implemente as funções abaixo (note que são bem parecidas e muda somente os sinais + e -)

```
adicionarValorTotal = (valor) => {  
  this.setState({  
    valorTotal: this.state.valorTotal + valor  
  })  
}
```

```
removerValorTotal = (valor) => {  
  this.setState({  
    valorTotal: this.state.valorTotal - valor  
  })  
}
```

2- as funções recebem um **valor** e altera o **estado valorTotal** que está sendo utilizado na propriedade do **Carrinho**.

```
<Carrinho  
  carrinho={this.state.carrinho}  
  valorTotal={this.state.valorTotal}  
  removerItemDoCarrinho={this.removerItemDoCarrinho}  
/>
```



Implementação - Somar/subtrair o valor total

As funções precisam ser chamadas dentro de cada função que corresponde a sua ação passando o valor do produto:

```
adicionarProdutoNoCarrinho = (produto) => {  
  const produtoNoCarrinho = this.state.carrinho.filter((item) => {  
    if (item.id === produto.id) {  
      return item;  
    } else {  
      return false  
    }  
  });  
  
  if (produtoNoCarrinho.length === 0) {  
    produto.quantidade = 1;  
    const novoCarrinho = [produto, ...this.state.carrinho];  
    this.setState({  
      carrinho: novoCarrinho,  
    });  
  } else {  
    const novoCarrinho = this.state.carrinho.map((item) => {  
      if (produto.id === item.id) {  
        return { ...item, quantidade: item.quantidade + 1 };  
      } else {  
        return item;  
      }  
    });  
  
    this.setState({  
      carrinho: novoCarrinho,  
    });  
  }  
  this.adicionarValorTotal(produto.price);  
};
```

```
removerItemDoCarrinho = (itemParaRemover) => {  
  if (itemParaRemover.quantidade === 1) {  
    const novoCarrinho = this.state.carrinho.filter((item) => {  
      if (item.id !== itemParaRemover.id) {  
        return item;  
      } else {  
        return false  
      }  
    });  
    this.setState({  
      carrinho: novoCarrinho,  
    });  
  } else {  
    const novoCarrinho = this.state.carrinho.map((item) => {  
      if (itemParaRemover.id === item.id && item.quantidade >= 1) {  
        return { ...item, quantidade: item.quantidade - 1 };  
      } else {  
        return item;  
      }  
    });  
    this.setState({  
      carrinho: novoCarrinho,  
    });  
  }  
  this.removerValorTotal(itemParaRemover.price);  
};
```

* **adicionarProdutoNoCarrinho** soma o valor no Total chamando **adicionarValorTotal**, e **removerItemDoCarrinho** vai subtrair o valor total, chamando **removerValorTotal**.



LabECommerce

Agora podemos fazer as nossas comprinhas 😊

Filtros

Filtro Mínimo:

Filtro Máximo:

Busca por nome:

Quantidade de produtos: 3

Ordenação: Crescente



Produto 3

R\$ 30,00

Adicionar carrinho



Produto legal

R\$ 123,00

Adicionar carrinho



Produto 2

R\$ 200,00

Adicionar carrinho

Carrinho:

5x Produto 2

Remover

4x Produto legal

Remover

6x Produto 3

Remover

Valor total: R\$ 1672,00





Obrigado(a)!