9 - Adicionando um carrinho

E aí, programador! o/

Nessa aula nós vamos começar a parte final da nossa aplicação adicionando um carrinho de compras funcional através de um hook que nós mesmos vamos criar. Assim você poderá ver como o Next.js é, em sua essência, o React que você já conhece.

1. Vamos começar atualizando o visual da nossa página de carrinho atual que está bem vazia e sem graça:

```
// pages/cart.tsx

import { NextPage } from "next"
import Head from "next/head"
import { Container } from "reactstrap"
import Header from "../src/components/Header"

const Cart: NextPage = () => {
  return (
    <>
      <Head>
        <title>Carrinho</title>
        <meta name="description" content="Meu carrinho de compras" />
        <link rel="icon" href="/favicon.ico" />
      </Head>

      <Header />

      <main>
        <Container className="mb-5">
          <h1 className="my-5">
            Carrinho
          </h1>

        </Container>
      </main>
    </>
  )
}

export default Cart
```

2. Agora já podemos começar a criar um contexto para lidar com nosso carrinho. Ele vai ajudar a ter acesso ao que precisamos para trabalhar com o carrinho de qualquer

componente de nível mais interno evitando *prop drilling*. Vamos também utilizar nosso próprio hook para deixar tudo mais organizado. Crie uma pasta “hooks” dentro de “src” e nela crie o arquivo useCart.tsx. Dentro desse arquivo vamos começar criando o contexto em si:

Obs.: Repare que já definimos que nosso contexto terá 3 propriedades, o próprio carrinho com a lista dos produtos e funções para adicionar e remover produtos dele.

```
// src/hooks/useCart.tsx

import React, { createContext, ReactNode, useContext, useEffect, useState } from "react";
import { ProductType } from "../services/products";

type CartContextType = {
  cart: ProductType[]
  addProduct: (product: ProductType) => void
  removeProduct: (productId: number) => void
}

const CartContext = createContext<CartContextType>({} as CartContextType)
```

3. Com o contexto criado, vamos criar o provider do contexto e exportá-lo para facilitar a inserção desse provider lá no wrapper das páginas (_app.tsx). Por fim, também vamos criar o nosso hook que retorna o contexto, assim podemos importar apenas esse hook e já teremos acesso ao contexto:

   Obs.: Esse nosso contexto vai ser bem simples. Ele vai ter apenas um state para o carrinho em si, e funções de adicionar e remover itens desse carrinho. Para deixar mais interessante podemos incluir nosso carrinho no localStorage, assim teremos alguma persistência de dados na aplicação. Nada de novo aqui se você já trabalhou com React antes.

```
// src/hooks/useCart.tsx

// ...

export const CartContextProvider = (props: {
  children: ReactNode
}) => {
  const [cart, setCart] = useState<ProductType[]>([])

  useEffect(() => {
    const storedCart = localStorage.getItem('shopping-cart')
```

```
    if (storedCart) {
      setCart(JSON.parse(storedCart))
    }
  }, [])

  const addProduct = (product: ProductType) => {
    const updatedCart = [...cart, product]
    localStorage.setItem('shopping-cart', JSON.stringify(updatedCart))
    setCart(updatedCart)
  }

  const removeProduct = (productId: number) => {
    const productIndex = cart.findIndex(product => product.id === productId)

    if (productIndex !== -1) {
      const updatedCart = [...cart]
      updatedCart.splice(productIndex, 1)
      localStorage.setItem('shopping-cart', JSON.stringify(updatedCart))
      setCart(updatedCart)
    }
  }

  return (
    <CartContext.Provider
      value={{ cart, addProduct, removeProduct }}
    >
      {props.children}
    </CartContext.Provider>
  );
}

export const useCart = () => useContext(CartContext)
```

4. Agora que já temos nosso contexto e hook criados vamos incluí-los no nosso código. Como vamos utilizar o carrinho em múltiplas páginas, podemos incluir o provider no arquivo _app.tsx, deixando disponível em todas as páginas:

```
// pages/_app.tsx

import 'bootstrap/dist/css/bootstrap.min.css'
import type { AppProps } from 'next/app'
import { CartContextProvider } from '../src/hooks/useCart'

function MyApp({ Component, pageProps }: AppProps) {
  return (
    <CartContextProvider>
      <Component {...pageProps} />
    </CartContextProvider>
  )
}
```

```
export default MyApp
```

5. Já podemos incluir a função de adicionar ao carrinho nos botões lá dos produtos, tanto dos cards quanto da página individual:

```
// src/components/ProductCard.tsx

// ...

import { useCart } from "../hooks/useCart"

// ...

const ProductCard: React.FC<ProductCardProps> = ({ product }) => {
  const [toastIsOpen, setToastIsOpen] = useState(false)
  const { id, name, imageUrl, price } = product
  const { addProduct } = useCart()

  return (

    // ...

          <Button
            color="dark"
            className="pb-2"
            block
            onClick={() => {
              addProduct(product)
              setToastIsOpen(true)
              setTimeout(() => setToastIsOpen(false), 1000 * 3)
            }}
          >
            Adicionar ao Carrinho
          </Button>

    // ...
```

```
// src/components/ProductDetails.tsx

// ...

import { useCart } from "../hooks/useCart";

// ...

const ProductDetails: React.FC<ProductDetailsProps> = ({ product }) => {
```

```
  const [toastIsOpen, setToastIsOpen] = useState(false)
  const { addProduct } = useCart()

  return (

    // ...

        <Button
          color="dark"
          className="my-3 pb-2"
          onClick={() => {
            addProduct(product)
            setToastIsOpen(true)
            setTimeout(() => setToastIsOpen(false), 1000 * 3)
          }}
        >
          Compre agora
        </Button>

    // ...
```

6. Já temos a funcionalidade de adicionar ao carrinho pronta. Se você testar a aplicação agora vai ver que temos produtos sendo adicionados ao localStorage. Agora vamos mostrar esses produtos na página do carrinho através de uma tabela. Para isso, crie o componente CartTable.tsx na pasta “components”. Como esse componente é um pouco maior vamos criar separadamente um componente para a linha da tabela com cada entrada de produto, sua quantidade e os botões de adicionar/remover:

```
// src/components/CartTable.tsx

import Image from "next/image";
import { useEffect, useState } from "react";
import { Button, Col, Row, Table } from "reactstrap";
import { useCart } from "../hooks/useCart";
import { ProductType } from "../services/products";

type CartEntry = {
  product: ProductType
  quantity: number
}

const CartTableRow = (props: {
  entry: CartEntry
}) => {
  const { addProduct, removeProduct } = useCart()

  return (
    <tr>
```

```
      <td>
        <Row className="align-items-center">
          <Col xs={4} md={2} lg={1}>
            <Image
              src={props.entry.product.imageUrl}
              alt={props.entry.product.name}
              height={500}
              width={600}
            />
          </Col>
          <Col xs={8} md={10} lg={11}>
            {props.entry.product.name}
          </Col>
        </Row>
      </td>
      <td>R$ {props.entry.product.price}</td>
      <td>{props.entry.quantity}</td>
      <td>R$ {(props.entry.product.price * props.entry.quantity)}</td>
      <td>
        <Button
          color="primary"
          size="sm"
          onClick={() => addProduct(props.entry.product)}
        >
          +
        </Button>
        {' '}
        <Button
          color="danger"
          size="sm"
          onClick={() => removeProduct(props.entry.product.id)}
        >
          -
        </Button>
      </td>
    </tr>
  )
}
```

7. E agora podemos criar o componente da tabela, onde obtemos todos os produtos do carrinho e transformamos eles em uma lista que possui a quantidade de cada produto repetido:

```
// src/components/CartTable.tsx

// ...

export default function CartTable() {
  const [cartEntries, setCartEntries] = useState<CartEntry[]>([])
  const { cart } = useCart()
```

```
  useEffect(() => {
    const entriesList = cart.reduce((list, product) => {
      const entryIndex = list.findIndex(entry => entry.product.id === product.id)

      if (entryIndex === -1) {
        return [
          ...list,
          {
            product,
            quantity: 1
          }
        ]
      }

      list[entryIndex].quantity++
      return list

    }, [] as CartEntry[])

    entriesList.sort((a, b) => a.product.id - b.product.id)
    setCartEntries(entriesList)

  }, [cart])

  return (
    <Table responsive className="align-middle" style={{ minWidth: '32rem' }}>
      <thead>
        <tr>
          <th>Produto</th>
          <th>Preço</th>
          <th>Qtd.</th>
          <th>Total</th>
        </tr>
      </thead>
      <tbody>
        {cartEntries.map(entry => <CartTableRow key={entry.product.id} entry={entry} />)}
      </tbody>
    </Table>
  )
}
```

8. Se incluirmos nossa tabela na página do carrinho vamos ver que já está tudo funcionando corretamente:

```
// pages/cart.tsx

// ...

import CartTable from "../src/components/CartTable"

// ...
```

```
      <main>
        <Container className="mb-5">
          <h1 className="my-5">
            Carrinho
          </h1>

          <CartTable />
        </Container>
      </main>

// ...
```

9. Para finalizar, podemos criar um pequeno card mostrando o total do carrinho, só para deixar nossa aplicação mais completa. Crie um componente CartTotal.tsx:

```
// src/components/CartTotal.tsx

import { Card, CardBody } from "reactstrap";
import { useCart } from "../hooks/useCart";

const CartTotal = () => {
  const { cart } = useCart()

  return (
    <Card className="ms-auto" style={{ maxWidth: '20rem' }}>
      <CardBody className="d-flex justify-content-between">
        <strong>
          Total:
        </strong>
        <span>
          R$ {cart.reduce((total, product) => total + product.price, 0)}
        </span>
      </CardBody>
    </Card>
  )
}

export default CartTotal
```

10. E inclua esse componente na página do carrinho como acabamos de fazer com a tabela:

```
// pages/cart.tsx

// ...
```

```
import CartTable from "../src/components/CartTable"
import CartTotal from "../src/components/CartTotal"

// ...

      <main>
        <Container className="mb-5">
          <h1 className="my-5">
            Carrinho
          </h1>

          <CartTable />
          <CartTotal />
        </Container>
      </main>

// ...
```

11. Com isso finalizamos! Agora você já pode subir a aplicação para o GitHub e deixar seu projeto completo disponível na Vercel.