7 - Obtendo os produtos da API

E aí, programador! o/

Nessa aula vamos criar uma página para exibir os produtos obtidos na API e começar a dar uma cara mais interessante para nossa aplicação.

1. Vamos começar criando uma barra de navegação no topo das páginas. Crie uma pasta “src” e dentro dela uma pasta “components”. Na pasta “components” crie o componente Header.tsx:

   Obs.: Repare que estamos usando o componente Link do Next.js para navegar entre páginas sem atualizar, usando navegação do lado do cliente.

```
// src/components/Header.tsx

import Link from "next/link";
import { Nav, Navbar } from "reactstrap";

const Header = () => {
  return (
    <Navbar container="md" color="dark" dark>
      <Link href="/" passHref>
        <a className="navbar-brand">
          Início
        </a>
      </Link>
      <Nav className="flex-row" navbar>
        <Link href="/products">
          <a className="nav-link me-2">
            Produtos
          </a>
        </Link>
        <Link href="/cart">
          <a className="nav-link">
            Carrinho
          </a>
        </Link>
      </Nav>
    </Navbar>
  )
}

export default Header
```

2. Agora vamos deixar nossa página inicial um pouco mais interessante e incluir a barra de navegação nela:

```
// pages/index.tsx

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

const Home: NextPage = () => {
  return (
    <>
      <Head>
        <title>Início</title>
        <meta name="description" content="Generated by create next app" />
        <link rel="icon" href="/favicon.ico" />
      </Head>

      <Header />

      <main >
        <Container className="py-5 text-center">
          <h1 className="mt-5 display-1">
```

```
            O melhor jeito de comprar o que você ama
          </h1>
          <p className="my-4">
            Lorem ipsum, dolor sit amet consectetur adipisicing elit. Molestiae iusto voluptatem obcaecati omnis error architecto
          </p>
          <Link href="/products">
            <Button color="dark" className="px-4 pb-2">
              Conheça nossos produtos!
            </Button>
          </Link>
        </Container>
      </main>
    </>
  )
}

export default Home
```

3. Antes de prosseguirmos vamos criar uma pasta “services” onde guardaremos as funções para chamar a API. Nela iremos criar um arquivo products.ts e exportar as seguintes funções:

```
// src/services/products.ts

export type ProductType = {
  id: number
  name: string
  description: string
  price: number
  imageUrl: string
  inStock: number
}

export const fetchProducts = async () => {
  const products: ProductType[] = await fetch(`${process.env.NEXT_PUBLIC_API_BASE_URL}/api/products`).then(res => res.json())
  return products
}

export const fetchProduct = async (id: string | number) => {
  const product: ProductType = await fetch(`${process.env.NEXT_PUBLIC_API_BASE_URL}/api/products/${id}`).then(res => res.json())
  return product
}
```

4. Vamos criar um componente agora que será um card de produto. Na página de produtos teremos uma lista cheia deles, um para cada produto vindo da API. Crie o componente ProductCard.tsx na pasta “components”:

   Obs.: Repare que estamos utilizando os componentes Image e Link para otimizar nossa página.

   Obs².: Repare que temos uma prop no componente que é o produto que virá da API.

   Obs³.: Também estamos incluindo um Toast, que é aquela caixinha de sucesso verde de “Produto adicionado ao carrinho”. Quando clicarmos no botão essa função será chamada exibindo a mensagem. Não se preocupe com o erro, criaremos o componente a seguir.

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

import Image from "next/image"
import Link from "next/link"
import React, { useState } from "react"
import { Button, Card, CardBody, CardSubtitle } from "reactstrap"
import { ProductType } from "../services/products"
import SuccessToast from "./SuccessToast"

type ProductCardProps = {
  product: ProductType
}

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

```
  return (
    <Card>
      <Link href={`/products/${id}`}>
        <Image className="card-img-top" src={imageUrl} alt="Product" height={500} width={600} />
      </Link>

      <CardBody>
        <Link href={`/products/${id}`}>
          <h5 className="card-title" style={{ cursor: 'pointer' }}>
            {name}
          </h5>
        </Link>

        <CardSubtitle className="mb-3 text-muted" tag="h6">
          R$ {price}
        </CardSubtitle>

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

      </CardBody>
    </Card>

    <SuccessToast toastIsOpen={toastIsOpen} setToastIsOpen={setToastIsOpen} />
  )
}

export default ProductCard
```

5. Vamos criar o componente do toast de sucesso, assim podemos reutilizá-lo no futuro:

```
// src/components/SuccessToast.tsx

import { Button, Toast, ToastBody } from "reactstrap"

const SuccessToast = (props: {
  toastIsOpen: boolean
  setToastIsOpen: (isOpen: boolean) => void
}) => {
  return (
    <Toast
      className="bg-success text-white fixed-bottom ms-auto me-4 mb-4"
      isOpen={props.toastIsOpen}
      fade
    >
      <ToastBody className="d-flex justify-content-between">
        Produto adicionado ao carrinho.
        <Button
          close
          className="btn-close-white"
          onClick={() => props.setToastIsOpen(false)}
        ></Button>
      </ToastBody>
    </Toast>
  )
}

export default SuccessToast
```

6. Agora vamos criar um componente que será a lista de todos os cards de produtos, podemos chamá-lo de ProductsList:

```
// src/components/ProductsList.tsx

import React from "react"
import { Col, Row } from "reactstrap"
import { ProductType } from "../services/products"
import ProductCard from "./ProductCard"

type ProductListProps = {
  products: ProductType[]
}

const ProductsList: React.FC<ProductListProps> = ({ products }) => {

  return (
    <>
      <Row className="g-5">
        {products.map(product => (
          <Col md={6} lg={4} xl={3} key={product.id}>
            <ProductCard
              product={product}
            />
          </Col>
        ))}
      </Row>
    </>
  )
}

export default ProductsList
```

7. Agora só precisamos atualizar nossa página de produtos para fazer a chamada a API, receber os produtos e incluir o componente ProductsList:

```
// pages/products.tsx

import { GetStaticProps, NextPage } from 'next'
import Head from 'next/head'
import { ReactNode } from 'react'
import { Container } from 'reactstrap'
import Header from '../src/components/Header'
import ProductsList from '../src/components/ProductsList'
import { fetchProducts, ProductType } from '../src/services/products'

export const getStaticProps: GetStaticProps = async () => {
  const products = await fetchProducts()
  return { props: { products } }
}

const Products: NextPage = (props: {
  children?: ReactNode
  products?: ProductType[]
}) => {
  return (
    <>
      <Head>
        <title>Nossos Produtos</title>
        <meta name="description" content="Conheça todos os nossos produtos" />
        <link rel="icon" href="/favicon.ico" />
      </Head>

      <Header />

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

          {<ProductsList products={props.products!} />}
        </Container>
      </main>{
    </>
  )
}
```

```
export default Products
```

8. E como estamos utilizando imagens de um servidor externo no componente Image precisamos especificar no arquivo next.config.js na raiz do projeto o domínio do servidor:

```
// next.config.js

/** @type {import('next').NextConfig} */
const nextConfig = {
  reactStrictMode: true,
  images: {
    domains: ['store.storeimages.cdn-apple.com'],
  },
}

module.exports = nextConfig
```

9. Se testarmos nossa aplicação agora veremos que tudo funciona corretamente. A navegação do lado do cliente funciona, não havendo atualização entre uma página e outra. Nossas imagens de produtos também estão otimizadas, prevenindo o CLS e carregando apenas quando entram na viewport. Além disso, ao clicar no botão vemos que nossa mensagem de sucesso aparece no canto inferior da tela e desaparece depois de 3 segundos.
10. Também já podemos subir a aplicação para o github e ver as mudanças na Vercel.