POKEMONLIST.JS NO CONTEXT

```
import React from "react";
// import "../App.css"; // Importa el archivo CSS para los estilos
import Product from "../componentes/Product";
import CalculateTotal from "../componentes/CalculateTotal";
import { useEffect, useState } from "react";
// fUENTE internet: Importa el icono de carrito de React Icons
import { FaShoppingCart } from "react-icons/fa";
import { FaTrashAlt } from "react-icons/fa";

// Componente principal que maneja la lista de productos y el carrito
export default function PokemonList() {

const [pokemons, setPokemons] = useState([]);
const [cart, setCart] = useState([]); // Inicializa el estado del carrito como un array vacío
// const [pokemon, setPokemon] = useState{};

// Función para obtener datos de la PokéAPI

useEffect(() => {
// Obtenemos una lista de los primeros 10 Pokémon
fetch(https://pokeapi.co/api/v2/pokemon?limit=10)
.then((response) => response.json())
.then((data) => {
const results = data.results;
```

```
      // Obtener detalles adicionales de cada Pokemon
      const promises = results.map((pokemon) =>
        fetch(pokemon.url)
          .then((res) => res.json())  // recoge la respuesta json
          .then((resData) => ({
            id: resData.id,
            name: resData.name,
            price: `$${(resData.id * 10).toFixed(2)}`, // Asignamos un precio
basado en su id
            image: resData.sprites.front_default, // Obtenemos la imagen
          }))
      );
      // Promise.all Espera que todas las promesas se resuelvan
    Promise.all(promises).then((pokemonData) => {
        setPokemons(pokemonData);
        console.log("pokemonData:"+pokemonData) ;
    });
  })
    .catch((error) => console.error("Error fetching the pokemons: ", error));

}, []);
```

```
  // return pokemons;

  // Función para agregar un producto (parámetro = pokemon) al carrito: agrupando
  por producto
  const handleAddToCart = (pokemon) => {
    //setCart es una función de React setState para actualizar el estado basado
  en su valor anterior, Esta función recibe el estado anterior como
  argumento.prevCart = previousCart (carrito anterior) Es el estado actual del
  carrito justo antes de que se realice la actualización.
    setCart((prevCart) => {
      //busca en el carrito previo (prevCart) un producto que tenga el mismo id
  que el producto que se quiere agregar. Si lo encuentra, lo almacena en la
  variable existingProduct.

      const existingProduct = prevCart.find((item) => item.id === pokemon.id);
      if (existingProduct) {
        // Si el producto ya está en el carrito, incrementa la cantidad
        return prevCart.map((item) =>
          item.id === pokemon.id ? { ...item, quantity: item.quantity + 1 } :
  item
        );
      } else {
        // Si el producto no está en el carrito, agrégalo con cantidad 1
        return [...prevCart, { ...pokemon, quantity: 1 }];
      }
    });
  };


  const handleRemoveFromCart = (indexToRemove) => {
      setCart(cart.filter((_, index) => index !== indexToRemove));
  };
```

return (

# Catálogo de Pokemon

{pokemons.map((pokemon) => ( ))}

{/_

## Carrito

_/}

## {/_ Carrito _/} {/_ Fuente Internet: icono de carrito _/}

{/_ Mostar producto seleccionado: nombre del producto: el precio unitario, la cantidad y el precio total _/}

{cart.map((item, index) => (

- {item.name} - {item.quantity} x {item.price} = ${item.quantity * parseFloat(item.price.replace("$", ""))} handleRemoveFromCart(index)}> Eliminar

))}

```
</div>
```

);
};

ShoppingCart.JS NO CONTEXT

import React from "react";
import { FaShoppingCart } from "react-icons/fa";
import { FaTrashAlt } from "react-icons/fa";
import CalculateTotal from "../componentes/CalculateTotal";
import { useState } from "react";

export default function ShoppingCart() {
const [cart, setCart] = useState([]); // Inicializa el estado del carrito como un array vacío

```
  // return pokemons;

// Función para agregar un producto (parámetro = pokemon) al carrito: agrupando por producto
const handleAddToCart = (pokemon) => {
    //setCart es una función de React setState para actualizar el estado basado en su valor anterior, Esta función recibe el estado anterior como argumento.prevCart = previousCart (carrito anterior) Es el estado actual del carrito justo antes de que se realice la actualización.
    setCart((prevCart) => {
      //busca en el carrito previo (prevCart) un producto que tenga el mismo id que el producto que se quiere agregar. Si lo encuentra, lo almacena en la variable existingProduct.

      const existingProduct = prevCart.find((item) => item.id === pokemon.id);
      if (existingProduct) {
        // Si el producto ya está en el carrito, incrementa la cantidad
        return prevCart.map((item) =>
          item.id === pokemon.id ? { ...item, quantity: item.quantity + 1 } : item
        );
      } else {
        // Si el producto no está en el carrito, agrégalo con cantidad 1
        return [...prevCart, { ...pokemon, quantity: 1 }];
      }
    });
```

```
    };


    const handleRemoveFromCart = (indexToRemove) => {
        setCart(cart.filter((_, index) => index !== indexToRemove));
    };
```

return (

{/_

## Carrito

_/}

## {/_ Carrito _/} {/_ Fuente Internet: icono de carrito _/}

{/_ Mostar producto seleccionado: nombre del producto: el precio unitario, la cantidad y el precio total _/}

{cart.map((item, index) => (

- {item.name} - {item.quantity} x {item.price} = ${item.quantity * parseFloat(item.price.replace("$", ""))} handleRemoveFromCart(index)}> Eliminar

))}

); }