versao 2

"use client";

import { useEffect, useState } from "react";

import { ToastContainer, toast } from "react-toastify";

import "react-toastify/dist/ReactToastify.css";

type Produto = {

id: number;

nome: string;

quantidade: number;

preco: number;

};

type Venda = {

id: number;

produtoId: number;

produto: Produto;

quantidade: number;

preco: number;

criadoEm: string;

};

export default function VendasPage() {

const [produtos, setProdutos] = useState<Produto[]>([]);

const [vendas, setVendas] = useState<Venda[]>([]);

const [produtoId, setProdutoId] = useState<number | "">("");

const [quantidade, setQuantidade] = useState<number | "">("");

const [precoUnitario, setPrecoUnitario] = useState<number>(0);

useEffect(() => {

fetch("/api/produtos")

.then((res) => res.json())

.then((data) => setProdutos(data))

.catch(console.error);

fetch("/api/vendas")

.then((res) => res.json())

.then((data) => setVendas(data))

.catch(console.error);

}, []);

// Soma o total das vendas para mostrar no topo

const totalVendas = vendas.reduce((acc, venda) => acc + Number(venda.preco), 0);

const precoTotal =

quantidade && precoUnitario

? Number(quantidade) \* Number(precoUnitario)

: 0;

function handleProdutoChange(e: React.ChangeEvent<HTMLSelectElement>) {

const id = Number(e.target.value);

setProdutoId(id);

const produtoSelecionado = produtos.find((p) => p.id === id);

setPrecoUnitario(Number(produtoSelecionado?.preco ?? 0));

}

async function handleSubmit(e: React.FormEvent<HTMLFormElement>) {

e.preventDefault();

if (produtoId === "" || quantidade === "") {

alert("Preencha todos os campos.");

return;

}

const novaVenda = {

produtoId,

quantidade: Number(quantidade),

preco: precoTotal,

};

// Verificar estoque para alerta de estoque baixo

const produtoAtual = produtos.find((p) => p.id === produtoId);

if (produtoAtual && produtoAtual.quantidade - quantidade <= 5) {

toast.warning("⚠ Estoque baixo! Reponha o produto.");

}

const res = await fetch("/api/vendas", {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(novaVenda),

});

if (res.ok) {

const vendaCriada = await res.json();

setVendas((prev) => [vendaCriada, ...prev]);

setProdutoId("");

setQuantidade("");

setPrecoUnitario(0);

toast.success("Venda registrada com sucesso!"); // <-- toast de sucesso adicionado aqui

} else {

const erro = await res.json();

alert(erro.message || "Erro ao registrar venda.");

}

}

return (

<div>

{/\* ToastContainer para mostrar os toasts \*/}

<ToastContainer position="top-right" autoClose={4000} />

{/\* Exibe o total das vendas no topo \*/}

<div className="mb-6 p-4 bg-pink-100 rounded shadow text-pink-800 font-semibold text-lg">

Total de Vendas: R$ {totalVendas.toFixed(2)}

</div>

<h1 className="text-2xl font-bold text-pink-700 mb-4">Registro de Vendas</h1>

<p className="text-gray-600 mb-6">

Cadastre as vendas realizadas e acompanhe o histórico.

</p>

<form

onSubmit={handleSubmit}

className="mb-8 space-y-4 bg-pink-50 bg-opacity-70 p-6 rounded-md shadow-sm text-gray-700"

>

<div>

<label className="block text-sm text-gray-700 mb-1">Produto</label>

<select

value={produtoId}

onChange={handleProdutoChange}

required

className="w-full border border-pink-300 rounded px-3 py-2 bg-white"

>

<option value="" disabled>

Selecione o produto

</option>

{produtos.map((p) => (

<option key={p.id} value={p.id}>

{p.nome}

</option>

))}

</select>

</div>

<div className="grid grid-cols-2 gap-4">

<div>

<label className="block text-sm text-gray-700 mb-1">Quantidade</label>

<input

type="number"

min={1}

value={quantidade}

onChange={(e) => setQuantidade(Number(e.target.value))}

required

className="w-full border border-pink-300 rounded px-3 py-2 bg-white"

/>

</div>

<div>

<label className="block text-sm text-gray-700 mb-1">Preço Unitário</label>

<input

type="text"

readOnly

value={`R$ ${Number(precoUnitario).toFixed(2)}`}

className="w-full border border-pink-300 rounded px-3 py-2 bg-white text-gray-700"

/>

</div>

</div>

<div>

<label className="block text-sm text-gray-700 mb-1">Total</label>

<input

type="text"

readOnly

value={`R$ ${precoTotal.toFixed(2)}`}

className="w-full border border-pink-300 rounded px-3 py-2 bg-white text-gray-800"

/>

</div>

<button

type="submit"

className="bg-pink-600 text-white px-4 py-2 rounded hover:bg-pink-700"

>

Registrar Venda

</button>

</form>

<div className="overflow-auto">

<table className="min-w-full border border-pink-300 text-gray-700">

<thead className="bg-pink-100 text-pink-700">

<tr>

<th className="p-2 text-left">Produto</th>

<th className="p-2 text-left">Qtd</th>

<th className="p-2 text-left">Preço Total (R$)</th>

<th className="p-2 text-left">Data</th>

</tr>

</thead>

<tbody>

{vendas.length === 0 && (

<tr>

<td colSpan={4} className="text-center p-4 text-gray-500">

Nenhuma venda registrada.

</td>

</tr>

)}

{vendas.map((v) => (

<tr key={v.id} className="border-t">

<td className="p-2">{v.produto?.nome || "Produto removido"}</td>

<td className="p-2">

{v.produto?.quantidade !== undefined && v.produto.quantidade <= 5 ? (

<span className="text-red-600 font-semibold">

⚠ {v.produto.quantidade} (baixo)

</span>

) : (

v.quantidade

)}

</td>

<td className="p-2">R$ {Number(v.preco).toFixed(2)}</td>

<td className="p-2">

{typeof window !== "undefined"

? new Date(v.criadoEm).toLocaleString("pt-BR")

: ""}

</td>

</tr>

))}

</tbody>

</table>

</div>

</div>

);

}