import { useEffect, useState } from "react";

import { Link, useNavigate, useParams } from "react-router-dom";

import Navbar from "../../components/navbar/navbar.jsx";

import api from "../../constantes/api.js";

1. **Importações:**
   * useEffect e useState são hooks do React para gerenciamento de efeitos colaterais e estado.
   * Link, useNavigate e useParams são componentes e hooks do react-router-dom para navegação e manipulação de parâmetros.
   * Navbar é um componente personalizado, e api é um módulo para fazer requisições HTTP.

javascript

function DoctorsAdd() {

const { id\_doctor } = useParams();

const navigate = useNavigate();

1. **Componente Funcional:**
   * DoctorsAdd é um componente funcional.
   * useParams recupera parâmetros da rota, especificamente id\_doctor.
   * useNavigate é usado para navegação programática.

javascript

const [doctor, setDoctor] = useState({ name: "", icon: "M", id\_service: "" });

const [services, setServices] = useState([]);

1. **Variáveis de Estado:**
   * O estado doctor armazena os detalhes do médico.
   * O estado services armazena a lista de serviços.

javascript

async function loadDoctor(id) {

try {

const response = await api.get(`/doctors/${id}`);

console.log("API Response:", response.data);

if (response.data) {

setDoctor(response.data);

console.log("Doctor data:", response.data);

}

} catch (error) {

alert("Erro ao carregar médico");

console.error("Erro ao carregar médico:", error);

}

}

1. **Função** loadDoctor**:**
   * Busca detalhes do médico com base no ID.
   * Se bem-sucedido, atualiza o estado doctor.
   * Lida com erros exibindo alertas e registrando no console.

javascript

async function loadServices() {

try {

const response = await api.get(`/services`);

if (response.data) {

setServices(response.data);

}

} catch (error) {

alert("Erro ao listar serviços");

console.error("Erro ao listar serviços:", error);

}

}

1. **Função** loadServices**:**
   * Busca a lista de serviços.
   * Se bem-sucedido, atualiza o estado services.
   * Lida com erros exibindo alertas e registrando no console.

javascript

async function saveDoctor() {

const json = {

id\_doctor: id\_doctor,

id\_service: doctor.id\_service,

name: doctor.name,

icon: doctor.icon,

};

try {

const response = id\_doctor

? await api.put(`/doctors/${id\_doctor}`, json)

: await api.post("/doctors", json);

if (response.data) {

navigate("/doctors");

}

} catch (error) {

alert("Erro ao salvar médico.");

console.error("Erro ao salvar médico:", error);

}

}

1. **Função** saveDoctor**:**
   * Constrói um objeto JSON com detalhes do médico.
   * Envia uma requisição POST ou PUT para salvar o médico.
   * Em caso de sucesso, navega de volta para a lista de médicos.
   * Lida com erros exibindo alertas e registrando no console.

javascript

useEffect(() => {

loadServices();

}, []);

1. useEffect **para Serviços:**
   * Carrega serviços quando o componente é montado.

javascript

useEffect(() => {

if (id\_doctor) {

loadDoctor(id\_doctor);

}

}, [id\_doctor]);

1. useEffect **para Médico:**
   * Carrega detalhes do médico se id\_doctor estiver presente.
   * Dependência de id\_doctor garante que a função seja executada sempre que o ID mudar.

javascript

return (

<>

<Navbar />

<div className="container-fluid mt-page">

<div className="row col-lg-4 offset-lg-4">

<div className="col-12 mt-2">

<h2>{id\_doctor ? "Editar Médico" : "Novo Médico"}</h2>

</div>

1. **Função de Renderização:**
   * Retorna o JSX para renderizar o componente.
   * Inclui o componente Navbar.

javascript

<div className="col-12 mt-4">

<label htmlFor="doctor" className="form-label">Médico</label>

<input

type="text"

name="doctor"

id="doctor"

className="form-control"

value={doctor.name || ""}

onChange={(e) =>

setDoctor((prev) => ({ ...prev, name: e.target.value }))

}

placeholder="Digite o nome do médico"

/>

</div>

1. **Campo de Entrada do Nome do Médico:**
   * Campo de entrada para o nome do médico.
   * Atualiza doctor.name ao mudar.

javascript

<div className="col-12 mt-3">

<label htmlFor="service" className="form-label">Serviço</label>

<select

name="service"

id="service"

className="form-select"

value={doctor.id\_service || ""}

onChange={(e) => {

setDoctor((prev) => ({ ...prev, id\_service: e.target.value }));

}}

>

<option value="">Selecione o serviço</option>

{services.map((s) => (

<option key={s.id\_service} value={s.id\_service}>

{s.description}

</option>

))}

</select>

</div>

1. **Dropdown de Serviço:**
   * Dropdown para selecionar um serviço.
   * Atualiza doctor.id\_service ao mudar.
   * As opções são populadas a partir do estado services.

javascript

<div className="col-12 mt-3">

<label htmlFor="icon" className="form-label">Gênero</label>

<select

name="icon"

id="icon"

className="form-select"

value={doctor.icon}

onChange={(e) =>

setDoctor((prev) => ({ ...prev, icon: e.target.value }))

}

>

<option value="M">Masculino</option>

<option value="F">Feminino</option>

</select>

</div>

1. **Dropdown de Gênero:**
   * Dropdown para selecionar o gênero do médico.
   * Atualiza doctor.icon ao mudar.

javascript

<div className="col-12 mt-4">

<div className="d-flex justify-content-end">

<Link to="/doctors" className="btn btn-outline-primary me-3">

Cancelar

</Link>

<button

onClick={saveDoctor}

type="button"

className="btn btn-primary ms-2"

>

Salvar Dados

</button>

</div>

</div>

1. **Botões do Formulário:**
   * O botão "Cancelar" navega de volta para a lista de médicos.
   * O botão "Salvar Dados" aciona a função saveDoctor.

javascript

</div>

</div>

</>

);

}

export default DoctorsAdd;

1. **Exportar Componente:**
   * Exporta o componente DoctorsAdd como exportação padrão.