import React, { useState, useEffect } from 'react';

import { Search, Plus, Edit, Trash2, FileText, Filter } from 'lucide-react';

const AdvogadosNomeacaoApp = () => {

// Estado para armazenar dados

const [advogados, setAdvogados] = useState(() => {

const savedAdvogados = localStorage.getItem('advogados');

return savedAdvogados ? JSON.parse(savedAdvogados) : [];

});

const [processos, setProcessos] = useState(() => {

const savedProcessos = localStorage.getItem('processos');

return savedProcessos ? JSON.parse(savedProcessos) : [];

});

// Estados para gerenciar a interface

const [activeTab, setActiveTab] = useState('advogados');

const [searchTerm, setSearchTerm] = useState('');

const [areaFilter, setAreaFilter] = useState('');

// Estados para formulários

const [novoAdvogado, setNovoAdvogado] = useState({ nome: '', oab: '', areas: [], telefone: '', email: '' });

const [advogadoSelecionado, setAdvogadoSelecionado] = useState(null);

const [showAdvogadoForm, setShowAdvogadoForm] = useState(false);

const [novoProcesso, setNovoProcesso] = useState({ numero: '', advogado\_id: '', area: '', data\_nomeacao: '', status: 'Em andamento' });

const [processoSelecionado, setProcessoSelecionado] = useState(null);

const [showProcessoForm, setShowProcessoForm] = useState(false);

// Áreas de atuação disponíveis

const areasAtuacao = ['Criminal', 'Infância e Juventude', 'Júri', 'Execuções Penais', 'Cível'];

// Salvar dados no localStorage quando houver mudanças

useEffect(() => {

localStorage.setItem('advogados', JSON.stringify(advogados));

}, [advogados]);

useEffect(() => {

localStorage.setItem('processos', JSON.stringify(processos));

}, [processos]);

// Funções para gerenciar advogados

const handleAdvogadoSubmit = (e) => {

e.preventDefault();

if (advogadoSelecionado) {

// Atualizar advogado existente

setAdvogados(advogados.map(adv =>

adv.id === advogadoSelecionado.id ? { ...novoAdvogado, id: adv.id } : adv

));

} else {

// Adicionar novo advogado

const id = Date.now().toString();

setAdvogados([...advogados, { ...novoAdvogado, id }]);

}

resetAdvogadoForm();

};

const resetAdvogadoForm = () => {

setNovoAdvogado({ nome: '', oab: '', areas: [], telefone: '', email: '' });

setAdvogadoSelecionado(null);

setShowAdvogadoForm(false);

};

const editarAdvogado = (advogado) => {

setAdvogadoSelecionado(advogado);

setNovoAdvogado({ ...advogado });

setShowAdvogadoForm(true);

};

const excluirAdvogado = (id) => {

if (window.confirm('Tem certeza que deseja excluir este advogado?')) {

setAdvogados(advogados.filter(adv => adv.id !== id));

// Também excluir processos relacionados a este advogado

setProcessos(processos.filter(proc => proc.advogado\_id !== id));

}

};

// Funções para gerenciar processos

const handleProcessoSubmit = (e) => {

e.preventDefault();

if (processoSelecionado) {

// Atualizar processo existente

setProcessos(processos.map(proc =>

proc.id === processoSelecionado.id ? { ...novoProcesso, id: proc.id } : proc

));

} else {

// Adicionar novo processo

const id = Date.now().toString();

setProcessos([...processos, { ...novoProcesso, id }]);

}

resetProcessoForm();

};

const resetProcessoForm = () => {

setNovoProcesso({ numero: '', advogado\_id: '', area: '', data\_nomeacao: '', status: 'Em andamento' });

setProcessoSelecionado(null);

setShowProcessoForm(false);

};

const editarProcesso = (processo) => {

setProcessoSelecionado(processo);

setNovoProcesso({ ...processo });

setShowProcessoForm(true);

};

const excluirProcesso = (id) => {

if (window.confirm('Tem certeza que deseja excluir este processo?')) {

setProcessos(processos.filter(proc => proc.id !== id));

}

};

// Funções de filtro e estatísticas

const advogadosFiltrados = advogados.filter(adv =>

adv.nome.toLowerCase().includes(searchTerm.toLowerCase()) &&

(areaFilter === '' || adv.areas.includes(areaFilter))

);

const processosFiltrados = processos.filter(proc =>

proc.numero.toLowerCase().includes(searchTerm.toLowerCase()) &&

(areaFilter === '' || proc.area === areaFilter)

);

const contarProcessosPorAdvogado = (advogadoId) => {

return processos.filter(proc => proc.advogado\_id === advogadoId).length;

};

const encontrarAdvogado = (id) => {

return advogados.find(adv => adv.id === id) || { nome: 'Não encontrado' };

};

// Componente de estatísticas

const EstatisticasComponent = () => {

const advogadosComContagem = advogados.map(adv => ({

...adv,

totalProcessos: contarProcessosPorAdvogado(adv.id)

})).sort((a, b) => b.totalProcessos - a.totalProcessos);

const totalPorArea = areasAtuacao.map(area => ({

area,

total: processos.filter(proc => proc.area === area).length

}));

return (

<div className="space-y-6">

<div className="bg-white p-4 rounded-lg shadow">

<h3 className="text-lg font-medium mb-3">Distribuição de Processos por Advogado</h3>

<div className="overflow-auto max-h-96">

<table className="w-full text-sm">

<thead>

<tr className="bg-gray-100">

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

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

<th className="p-2 text-center">Total de Processos</th>

</tr>

</thead>

<tbody>

{advogadosComContagem.map(adv => (

<tr key={adv.id} className="border-b">

<td className="p-2">{adv.nome}</td>

<td className="p-2">{adv.oab}</td>

<td className="p-2 text-center">{adv.totalProcessos}</td>

</tr>

))}

</tbody>

</table>

</div>

</div>

<div className="bg-white p-4 rounded-lg shadow">

<h3 className="text-lg font-medium mb-3">Processos por Área</h3>

<div className="grid grid-cols-1 md:grid-cols-2 gap-4">

{totalPorArea.map(item => (

<div key={item.area} className="bg-gray-50 p-3 rounded">

<div className="font-medium">{item.area}</div>

<div className="text-2xl font-bold">{item.total}</div>

</div>

))}

</div>

</div>

</div>

);

};

return (

<div className="min-h-screen bg-gray-100">

<div className="bg-blue-800 text-white p-4 shadow-md">

<h1 className="text-xl font-bold">Sistema de Controle de Nomeação de Advogados Dativos</h1>

<h2 className="text-sm">Vara Criminal, Infância e Juventude, Júri e Execuções Penais - Comarca de Itanhém</h2>

</div>

<div className="container mx-auto p-4">

{/\* Tabs de navegação \*/}

<div className="flex border-b mb-4">

<button

className={`px-4 py-2 ${activeTab === 'advogados' ? 'border-b-2 border-blue-500 text-blue-600' : 'text-gray-500'}`}

onClick={() => setActiveTab('advogados')}

>

Advogados

</button>

<button

className={`px-4 py-2 ${activeTab === 'processos' ? 'border-b-2 border-blue-500 text-blue-600' : 'text-gray-500'}`}

onClick={() => setActiveTab('processos')}

>

Processos

</button>

<button

className={`px-4 py-2 ${activeTab === 'estatisticas' ? 'border-b-2 border-blue-500 text-blue-600' : 'text-gray-500'}`}

onClick={() => setActiveTab('estatisticas')}

>

Estatísticas

</button>

</div>

{/\* Barra de pesquisa e filtros \*/}

{activeTab !== 'estatisticas' && (

<div className="flex flex-col md:flex-row gap-4 mb-4">

<div className="flex-1 relative">

<Search className="absolute left-3 top-2.5 h-5 w-5 text-gray-400" />

<input

type="text"

placeholder="Pesquisar..."

className="pl-10 p-2 w-full border rounded"

value={searchTerm}

onChange={e => setSearchTerm(e.target.value)}

/>

</div>

<div className="flex items-center gap-2">

<Filter className="h-5 w-5 text-gray-500" />

<select

className="p-2 border rounded"

value={areaFilter}

onChange={e => setAreaFilter(e.target.value)}

>

<option value="">Todas as áreas</option>

{areasAtuacao.map(area => (

<option key={area} value={area}>{area}</option>

))}

</select>

<button

className="bg-blue-500 text-white p-2 rounded flex items-center gap-1"

onClick={() => {

if (activeTab === 'advogados') {

setShowAdvogadoForm(true);

} else {

setShowProcessoForm(true);

}

}}

>

<Plus className="h-5 w-5" />

Novo {activeTab === 'advogados' ? 'Advogado' : 'Processo'}

</button>

</div>

</div>

)}

{/\* Conteúdo da aba Advogados \*/}

{activeTab === 'advogados' && (

<div>

{showAdvogadoForm ? (

<div className="bg-white p-4 rounded-lg shadow mb-4">

<h3 className="text-lg font-medium mb-4">

{advogadoSelecionado ? 'Editar Advogado' : 'Novo Advogado'}

</h3>

<form onSubmit={handleAdvogadoSubmit} className="space-y-4">

<div className="grid grid-cols-1 md:grid-cols-2 gap-4">

<div>

<label className="block text-sm font-medium mb-1">Nome Completo</label>

<input

type="text"

className="w-full p-2 border rounded"

value={novoAdvogado.nome}

onChange={e => setNovoAdvogado({...novoAdvogado, nome: e.target.value})}

required

/>

</div>

<div>

<label className="block text-sm font-medium mb-1">Número OAB</label>

<input

type="text"

className="w-full p-2 border rounded"

value={novoAdvogado.oab}

onChange={e => setNovoAdvogado({...novoAdvogado, oab: e.target.value})}

required

/>

</div>

<div>

<label className="block text-sm font-medium mb-1">Telefone</label>

<input

type="tel"

className="w-full p-2 border rounded"

value={novoAdvogado.telefone}

onChange={e => setNovoAdvogado({...novoAdvogado, telefone: e.target.value})}

/>

</div>

<div>

<label className="block text-sm font-medium mb-1">E-mail</label>

<input

type="email"

className="w-full p-2 border rounded"

value={novoAdvogado.email}

onChange={e => setNovoAdvogado({...novoAdvogado, email: e.target.value})}

/>

</div>

</div>

<div>

<label className="block text-sm font-medium mb-1">Áreas de Atuação</label>

<div className="grid grid-cols-2 md:grid-cols-3 gap-2">

{areasAtuacao.map(area => (

<label key={area} className="flex items-center">

<input

type="checkbox"

className="mr-2"

checked={novoAdvogado.areas.includes(area)}

onChange={e => {

const updatedAreas = e.target.checked

? [...novoAdvogado.areas, area]

: novoAdvogado.areas.filter(a => a !== area);

setNovoAdvogado({...novoAdvogado, areas: updatedAreas});

}}

/>

{area}

</label>

))}

</div>

</div>

<div className="flex justify-end gap-2">

<button

type="button"

className="px-4 py-2 border rounded"

onClick={resetAdvogadoForm}

>

Cancelar

</button>

<button

type="submit"

className="px-4 py-2 bg-blue-500 text-white rounded"

>

Salvar

</button>

</div>

</form>

</div>

) : (

<div className="bg-white rounded-lg shadow overflow-hidden">

{advogadosFiltrados.length > 0 ? (

<table className="w-full text-sm">

<thead>

<tr className="bg-gray-100">

<th className="p-3 text-left">Nome</th>

<th className="p-3 text-left">OAB</th>

<th className="p-3 text-left">Áreas de Atuação</th>

<th className="p-3 text-center">Total de Processos</th>

<th className="p-3 text-right">Ações</th>

</tr>

</thead>

<tbody>

{advogadosFiltrados.map(advogado => (

<tr key={advogado.id} className="border-t hover:bg-gray-50">

<td className="p-3">{advogado.nome}</td>

<td className="p-3">{advogado.oab}</td>

<td className="p-3">

{advogado.areas.join(', ')}

</td>

<td className="p-3 text-center">

{contarProcessosPorAdvogado(advogado.id)}

</td>

<td className="p-3 text-right">

<div className="flex justify-end gap-2">

<button

onClick={() => editarAdvogado(advogado)}

className="p-1 text-blue-500 hover:bg-blue-100 rounded"

title="Editar"

>

<Edit className="h-5 w-5" />

</button>

<button

onClick={() => excluirAdvogado(advogado.id)}

className="p-1 text-red-500 hover:bg-red-100 rounded"

title="Excluir"

>

<Trash2 className="h-5 w-5" />

</button>

</div>

</td>

</tr>

))}

</tbody>

</table>

) : (

<div className="p-6 text-center text-gray-500">

Nenhum advogado encontrado.

</div>

)}

</div>

)}

</div>

)}

{/\* Conteúdo da aba Processos \*/}

{activeTab === 'processos' && (

<div>

{showProcessoForm ? (

<div className="bg-white p-4 rounded-lg shadow mb-4">

<h3 className="text-lg font-medium mb-4">

{processoSelecionado ? 'Editar Processo' : 'Novo Processo'}

</h3>

<form onSubmit={handleProcessoSubmit} className="space-y-4">

<div className="grid grid-cols-1 md:grid-cols-2 gap-4">

<div>

<label className="block text-sm font-medium mb-1">Número do Processo</label>

<input

type="text"

className="w-full p-2 border rounded"

value={novoProcesso.numero}

onChange={e => setNovoProcesso({...novoProcesso, numero: e.target.value})}

required

/>

</div>

<div>

<label className="block text-sm font-medium mb-1">Advogado</label>

<select

className="w-full p-2 border rounded"

value={novoProcesso.advogado\_id}

onChange={e => setNovoProcesso({...novoProcesso, advogado\_id: e.target.value})}

required

>

<option value="">Selecione um advogado</option>

{advogados.map(adv => (

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

{adv.nome} - OAB {adv.oab}

</option>

))}

</select>

</div>

<div>

<label className="block text-sm font-medium mb-1">Área</label>

<select

className="w-full p-2 border rounded"

value={novoProcesso.area}

onChange={e => setNovoProcesso({...novoProcesso, area: e.target.value})}

required

>

<option value="">Selecione uma área</option>

{areasAtuacao.map(area => (

<option key={area} value={area}>{area}</option>

))}

</select>

</div>

<div>

<label className="block text-sm font-medium mb-1">Data de Nomeação</label>

<input

type="date"

className="w-full p-2 border rounded"

value={novoProcesso.data\_nomeacao}

onChange={e => setNovoProcesso({...novoProcesso, data\_nomeacao: e.target.value})}

required

/>

</div>

<div>

<label className="block text-sm font-medium mb-1">Status</label>

<select

className="w-full p-2 border rounded"

value={novoProcesso.status}

onChange={e => setNovoProcesso({...novoProcesso, status: e.target.value})}

>

<option value="Em andamento">Em andamento</option>

<option value="Concluído">Concluído</option>

<option value="Suspenso">Suspenso</option>

</select>

</div>

</div>

<div className="flex justify-end gap-2">

<button

type="button"

className="px-4 py-2 border rounded"

onClick={resetProcessoForm}

>

Cancelar

</button>

<button

type="submit"

className="px-4 py-2 bg-blue-500 text-white rounded"

>

Salvar

</button>

</div>

</form>

</div>

) : (

<div className="bg-white rounded-lg shadow overflow-hidden">

{processosFiltrados.length > 0 ? (

<table className="w-full text-sm">

<thead>

<tr className="bg-gray-100">

<th className="p-3 text-left">Número</th>

<th className="p-3 text-left">Advogado</th>

<th className="p-3 text-left">Área</th>

<th className="p-3 text-left">Data de Nomeação</th>

<th className="p-3 text-left">Status</th>

<th className="p-3 text-right">Ações</th>

</tr>

</thead>

<tbody>

{processosFiltrados.map(processo => (

<tr key={processo.id} className="border-t hover:bg-gray-50">

<td className="p-3">{processo.numero}</td>

<td className="p-3">{encontrarAdvogado(processo.advogado\_id).nome}</td>

<td className="p-3">{processo.area}</td>

<td className="p-3">{processo.data\_nomeacao}</td>

<td className="p-3">

<span className={`px-2 py-1 rounded text-xs ${

processo.status === 'Em andamento' ? 'bg-blue-100 text-blue-800' :

processo.status === 'Concluído' ? 'bg-green-100 text-green-800' :

'bg-yellow-100 text-yellow-800'

}`}>

{processo.status}

</span>

</td>

<td className="p-3 text-right">

<div className="flex justify-end gap-2">

<button

onClick={() => editarProcesso(processo)}

className="p-1 text-blue-500 hover:bg-blue-100 rounded"

title="Editar"

>

<Edit className="h-5 w-5" />

</button>

<button

onClick={() => excluirProcesso(processo.id)}

className="p-1 text-red-500 hover:bg-red-100 rounded"

title="Excluir"

>

<Trash2 className="h-5 w-5" />

</button>

</div>

</td>

</tr>

))}

</tbody>

</table>

) : (

<div className="p-6 text-center text-gray-500">

Nenhum processo encontrado.

</div>

)}

</div>

)}

</div>

)}

{/\* Conteúdo da aba Estatísticas \*/}

{activeTab === 'estatisticas' && <EstatisticasComponent />}

</div>

</div>

);

};

export default AdvogadosNomeacaoApp;