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

import { Button, Container, TextField, Table, TableHead, TableRow, TableCell, TableBody } from '@mui/material';

import { ArrowDownward } from '@mui/icons-material';

import api from '../services/api';

import '../styles.css';

import '../tabela2.css';

function Home() {

const [file, setFile] = useState(null);

const [data, setData] = useState([]);

const [filteredData, setFilteredData] = useState([]);

const [anos, setAnos] = useState({ anoInicio: '', anoFim: '' });

const [nomePessoa, setNomePessoa] = useState('');

const [pontuacaoQualis, setPontuacaoQualis] = useState([]);

const [ponderacaoDiscente, setPonderacaoDiscente] = useState(75);

// Filtragem dos dados por ano

useEffect(() => {

if (anos.anoInicio && anos.anoFim) {

const filtered = data.filter(item => item.Ano >= parseInt(anos.anoInicio) && item.Ano <= parseInt(anos.anoFim));

setFilteredData(filtered);

recalcularPontuacao(filtered);

} else {

setFilteredData(data);

}

}, [anos, data]);

// Função de upload de arquivo

const handleFileChange = (e) => setFile(e.target.files[0]);

const handleAnoChange = (e) => setAnos(prev => ({ ...prev, [e.target.name]: e.target.value }));

const handlePonderacaoChange = (e) => {

setPonderacaoDiscente(e.target.value);

// Recalcula a Tabela 1 com os novos valores

const updatedData = filteredData.map(item => ({

...item,

Pts: calcularPontuacao(item.Qualis, item.DP, item.DIS)

}));

setFilteredData(updatedData);

// Recalcula a Tabela 2 com base nos novos valores recalculados

recalcularTabela2(updatedData); // Atualiza a Tabela 2

};

const recalcularPontuacao = (data) => {

const updatedData = data.map(item => ({

...item,

Pts: calcularPontuacao(item.Qualis, item.DP, item.DIS)

}));

setFilteredData(updatedData);

recalcularTabela2(updatedData); // Atualiza a Tabela 2 em tempo real

};

// Função para recalcular a Tabela 2 utilizando a coluna Pts da Tabela 1

const recalcularTabela2 = (data) => {

const pontuacaoPorAno = {};

data.forEach(item => {

if (!pontuacaoPorAno[item.Ano]) {

pontuacaoPorAno[item.Ano] = {

A1: 0, A2: 0, A3: 0, A4: 0, B1: 0, B2: 0, B3: 0, B4: 0, C: 0, NP: 0,

TotalA: 0, TotalB: 0, PtsAno: 0

};

}

// Incrementa a pontuação baseada no Qualis do artigo

pontuacaoPorAno[item.Ano][item.Qualis] += item.Pts;

// Acumula totais de A e B

if (['A1', 'A2', 'A3', 'A4'].includes(item.Qualis)) {

pontuacaoPorAno[item.Ano].TotalA += item.Pts;

} else if (['B1', 'B2', 'B3', 'B4'].includes(item.Qualis)) {

pontuacaoPorAno[item.Ano].TotalB += item.Pts;

}

// Acumula a pontuação total por ano

pontuacaoPorAno[item.Ano].PtsAno += item.Pts;

});

// Agora calcula os percentuais de A e B

Object.keys(pontuacaoPorAno).forEach(ano => {

const totalAno = pontuacaoPorAno[ano].PtsAno;

pontuacaoPorAno[ano]["% A"] = totalAno > 0 ? (pontuacaoPorAno[ano].TotalA / totalAno) \* 100 : 0;

pontuacaoPorAno[ano]["% B"] = totalAno > 0 ? (pontuacaoPorAno[ano].TotalB / totalAno) \* 100 : 0;

});

// Atualiza o estado da Tabela 2 com base nos valores calculados

setPontuacaoQualis(Object.keys(pontuacaoPorAno).map(ano => ({

Ano: ano,

A1: pontuacaoPorAno[ano].A1,

A2: pontuacaoPorAno[ano].A2,

A3: pontuacaoPorAno[ano].A3,

A4: pontuacaoPorAno[ano].A4,

B1: pontuacaoPorAno[ano].B1,

B2: pontuacaoPorAno[ano].B2,

B3: pontuacaoPorAno[ano].B3,

B4: pontuacaoPorAno[ano].B4,

C: pontuacaoPorAno[ano].C,

NP: pontuacaoPorAno[ano].NP,

TotalA: pontuacaoPorAno[ano].TotalA,

TotalB: pontuacaoPorAno[ano].TotalB,

"% A": pontuacaoPorAno[ano]["% A"],

"% B": pontuacaoPorAno[ano]["% B"]

})));

};

// Função para calcular a pontuação da Tabela 1

const calcularPontuacao = (qualis, dp, dis) => {

const qualisPts = { A1: 1, A2: 0.9, A3: 0.75, A4: 0.6, B1: 0.4, B2: 0.3, B3: 0.15, B4: 0.05, C: 0 };

const basePts = qualisPts[qualis] || 0;

// Se DIS for 0, aplica a ponderação, caso contrário, usa o valor original

const ponderacao = dis === 0 ? (ponderacaoDiscente / 100) : 1;

return (basePts \* ponderacao) / Math.max(dp, 1);

};

// Função de upload de arquivo

const handleUpload = async () => {

if (!file) return alert('Selecione o arquivo XML!');

const formData = new FormData();

formData.append('file', file);

try {

const response = await api.post('/upload/', formData);

const { nomePessoa, artigos, pontuacaoQualis } = response.data;

setNomePessoa(nomePessoa);

const adjustedData = artigos.map(item => ({

...item,

Qualis: item.Qualis || "NP",

'2023 JIF': item['2023 JIF'] || "---",

'JIF Quartile': item['JIF Quartile'] || "---",

SJR: item.SJR || "---",

'SJR Best Quartile': item['SJR Best Quartile'] || "---",

DOI: `http://dx.doi.org/${item.DOI}`

}));

setData(adjustedData);

setPontuacaoQualis(pontuacaoQualis);

alert('Arquivo enviado com sucesso!');

} catch (error) {

console.error('Erro ao enviar o arquivo:', error);

alert('Erro ao enviar o arquivo.');

}

};

// Manipulação de input editável (DP ou DIS)

const handleInputChange = (e, index, field) => {

const updatedData = [...filteredData];

const newValue = e.target.value === '' ? 0 : parseInt(e.target.value, 10);

// Atualiza o valor do campo (DP ou DIS)

updatedData[index][field] = isNaN(newValue) ? 0 : newValue;

// Recalcula a pontuação com o novo valor de DP ou DIS e atualiza a Tabela 1

recalcularPontuacao(updatedData);

setFilteredData(updatedData);

};

// Recalcula a Tabela 2 sempre que filteredData ou ponderacaoDiscente forem alterados

useEffect(() => {

if (filteredData.length > 0) {

recalcularTabela2(filteredData); // Atualiza a Tabela 2

}

}, [filteredData, ponderacaoDiscente]);

return (

<Container className="container-ppgec" maxWidth={false} style={{ paddingLeft: 70, paddingRight: 70, width: '100vw' }}>

<div className="logo-container">

<img src={require('../assets/PPGEC Tipo 2.png')} alt="Logo PPGEC" className="ppgec-logo" />

<h1 className="ppgec-title">Programa de Pós-graduação em Engenharia Civil</h1>

<h2 className="ppgec-subtitle">Analisador de Artigos</h2>

</div>

{nomePessoa && <h2 className="nome-pessoa">{nomePessoa}</h2>}

<div className="anos-container">

<TextField

label="Ano - Início"

name="anoInicio"

type="number"

value={anos.anoInicio}

onChange={handleAnoChange}

variant="outlined"

style={{ fontSize: 12, fontWeight: 'bold', height: 60, width: 100, textAlign: 'center' }}

/>

<TextField

label="Ano - Fim"

name="anoFim"

type="number"

value={anos.anoFim}

onChange={handleAnoChange}

variant="outlined"

style={{ fontSize: 12, fontWeight: 'bold', height: 60, width: 100, textAlign: 'center' }}

/>

<TextField

label="Ponderação Discente (%)"

type="number"

value={ponderacaoDiscente}

onChange={handlePonderacaoChange}

variant="outlined"

style={{ fontSize: 12, fontWeight: 'bold', height: 60, width: 170, textAlign: 'center' }}

/>

<div className="upload-container">

<div className="upload-label">

<label>Lattes (XML .zip)</label>

<input type="file" name="file" onChange={handleFileChange} className="file-input" />

</div>

<Button

variant="contained"

color="primary"

onClick={handleUpload}

className="upload-button"

style={{ marginLeft: '10px', height: '40px' }}

>

Enviar XML

</Button>

</div>

</div>

{/\* Tabela 1 \*/}

<h2 className="table-title">Tabela 1 - Listagem de artigos</h2>

{filteredData.length > 0 && (

<Table className="custom-table">

<TableHead>

<TableRow className="table-header">

{['Ano', 'DOI', 'ISSN', 'Periódico', 'Título', 'Autores', 'Qualis', 'JCR', 'Quartil', 'SJR', 'Quartil2', 'DP', 'DC', 'DIS', 'Pts'].map(header => (

<TableCell

key={header}

className={`col-${header.toLowerCase().replace(' ', '-'}`}

>

{header === 'Quartil' ? <>Q<sub>jcr</sub></> : header === 'Quartil2' ? <>Q<sub>sjr</sub></> : header}

</TableCell>

))}

</TableRow>

</TableHead>

<TableBody>

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

<TableRow key={index} className={index % 2 === 0 ? 'even-row' : 'odd-row'}>

<TableCell className="col-ano">{item.Ano}</TableCell>

<TableCell className="col-doi">

<a href={item.DOI} target="\_blank" rel="noopener noreferrer">

<ArrowDownward style={{ color: 'rgb(40, 161, 219)' }} />

</a>

</TableCell>

<TableCell className="col-issn">{item.ISSN}</TableCell>

<TableCell className="col-periodico">{item['Título do Periódico']}</TableCell>

<TableCell className="col-titulo">{item.Título}</TableCell>

<TableCell className="col-autores">{item.Autores}</TableCell>

<TableCell className="col-qualis">{item.Qualis}</TableCell>

<TableCell className="col-jcr">{item['2023 JIF']}</TableCell>

<TableCell className="col-quartil-jcr">{item['JIF Quartile']}</TableCell>

<TableCell className="col-sjr">

{item.SJR && !isNaN(parseFloat(item.SJR.replace(',', '.').trim())) ?

parseFloat(item.SJR.replace(',', '.').trim()).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })

: "---"

}

</TableCell>

<TableCell className="col-quartil-sjr">{item['SJR Best Quartile']}</TableCell>

<TableCell className="col-dp" align="center">

<input type="text" value={item.DP} onChange={(e) => handleInputChange(e, index, 'DP')} className="input-editable" />

</TableCell>

<TableCell className="col-dc" align="center">

<input type="text" value={item.DC} onChange={(e) => handleInputChange(e, index, 'DC')} className="input-editable" />

</TableCell>

<TableCell className="col-dis" align="center">

<input type="text" value={item.DIS} onChange={(e) => handleInputChange(e, index, 'DIS')} className="input-editable" />

</TableCell>

<TableCell className="col-pts">

{calcularPontuacao(item.Qualis, item.DP, item.DIS).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}

</TableCell>

</TableRow>

))}

</TableBody>

</Table>

)}

{/\* Tabela 2 - Pontuação Qualis \*/}

<h2 className="table-qualis-title">Tabela 2 - Pontuação Qualis</h2>

<Table className="table-qualis">

<TableHead>

<TableRow className="table-header">

{['Ano', 'A1', 'A2', 'A3', 'A4', 'B1', 'B2', 'B3', 'B4', 'C', 'NP', 'Total A', 'Total B', '% A', '% B'].map(header => (

<TableCell key={header} align="center"><strong>{header}</strong></TableCell>

))}

</TableRow>

</TableHead>

<TableBody>

{pontuacaoQualis.map((row, index) => (

<TableRow key={index}>

<TableCell align="center">{row.Ano}</TableCell>

<TableCell align="center">{(row.A1 || 0).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}</TableCell>

<TableCell align="center">{(row.A2 || 0).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}</TableCell>

<TableCell align="center">{(row.A3 || 0).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}</TableCell>

<TableCell align="center">{(row.A4 || 0).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}</TableCell>

<TableCell align="center">{(row.B1 || 0).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}</TableCell>

<TableCell align="center">{(row.B2 || 0).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}</TableCell>

<TableCell align="center">{(row.B3 || 0).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}</TableCell>

<TableCell align="center">{(row.B4 || 0).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}</TableCell>

<TableCell align="center">{(row.C || 0).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}</TableCell>

<TableCell align="center">{(row.NP || 0).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}</TableCell>

<TableCell align="center">{(row.TotalA || 0).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}</TableCell>

<TableCell align="center">{(row.TotalB || 0).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}</TableCell>

<TableCell align="center">{(row["% A"] || 0).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}</TableCell>

<TableCell align="center">{(row["% B"] || 0).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}</TableCell>

</TableRow>

))}

</TableBody>

</Table>

</Container>

);

}

export default Home;