

## Desenvolvimento de Aplicações Web - Entrega 4 - Marcos Henrique Almeida Lima

```
$$$ public.index
import React from 'react';
import { hydrateRoot, createRoot } from "react-dom/client";
import { locale, addLocale } from 'primereact/api';
import "primereact/resources/themes/lara-light-indigo/theme.css";
import "primereact/resources/primereact.min.css";
import "/node_modules/primeflex/primeflex.css";
import "primeicons/primeicons.css";
import "./global.css";
import português from "./utilitários/português.json"
import Rotas from "./rotas/rotas-aplicação";
import {ProvedorUsuário} from './contextos/contexto-usuário';
addLocale("pt", português)
locale("pt")
const rootElement = document.getElementById("root");
const App = (<ProvedorUsuário><Rotas/></ProvedorUsuário>);
if (rootElement.hasChildNodes()) hydrateRoot(App, rootElement);
else createRoot(rootElement).render(App);
```

```
$$$$ src/componentes/modais/modal-confirmação-usuário

import { useContext, useRef, useState } from "react";
import { useNavigate } from "react-router-dom";
import { Button } from "primereact/button";
import { Toast } from "primereact/toast";
import ContextoUsuário from "../../contextos/contexto-usuário";
//
import { serviçoAlterarUsuário, serviçoRemoverUsuário } from "../../../serviços/serviços-usuário";
import mostrarToast from "../../../utilitários/mostrar-toast";
//
import { estilizarBotão, estilizarBotãoRemover, estilizarDivCampo, estilizarInlineFlex,
estilizarLabel, estilizarModal } from "../../../utilitários/estilos";
export default function ModalConfirmaçãoUsuário() {
const referênciaToast = useRef(null);

const { setUsuárioLogado, confirmaçãoUsuário, setConfirmaçãoUsuário,
setMostrarModalConfirmação, usuárioLogado} = useContext(ContextoUsuário);
const dados = { cpf: confirmaçãoUsuário?.cpf, perfil: confirmaçãoUsuário?.perfil,
nome: confirmaçãoUsuário?.nome, senha: confirmaçãoUsuário?.senha,
email: confirmaçãoUsuário?.email, questão: confirmaçãoUsuário?.questão,
resposta: confirmaçãoUsuário?.resposta, cor_tema: confirmaçãoUsuário?.cor_tema };
const [redirecionar, setRedirecionar] = useState(false); //

const navegar = useNavigate();

//
function labelOperação() {
switch (confirmaçãoUsuário?.operação) {
case "salvar": return "Salvar";
case "alterar": return "Alterar";
case "remover": return "Remover";
default: return;
}
};

// 

function exibirPerfilFormatado() {
switch (dados.perfil) {
case "gerente-mineradora": return "GerenteMineradora";
case "gerente-Mineradora": return "GerenteMineradora";
default: return "";
}
};

// 

function fecharToast() {
if (redirecionar) {
```

```

setMostrarModalConfirmação(false);
setConfirmaçãoUsuário({});
if (confirmaçãoUsuário?.operação) setUsuárioLogado({}); // inseriu ?
navegar("../pagina-inicial");
}

};

function finalizarCadastro() {
  console.log("Valor da operação: ", dados.perfil); // Linha adicionada
  if (dados.perfil === "gerente-mineradora") {
    setUsuárioLogado({ ...dados, cadastrado: false });
    setMostrarModalConfirmação(false);
    navegar("../cadastrar-gerente-mineradora");
  } else if (dados.perfil === "gerente-tecnologia") {
    setUsuárioLogado({ ...dados, cadastrado: false });
    setMostrarModalConfirmação(false);
    navegar("../cadastrar-gerente-tecnologia");
  }
}
};

async function alterarUsuário(dadosAlterados) {
  try {
    const response = await serviçoAlterarUsuário({ ...dadosAlterados, cpf: usuárioLogado.cpf });
    setUsuárioLogado({ ...usuárioLogado, ...response.data });
    setRedirecionar(true);
    mostrarToast(referênciaToast, "Alterado com sucesso! Redirecionando à Página Inicial...", "sucesso");
  } catch (error) { mostrarToast(referênciaToast, error.response.data.erro, "erro"); }
};

async function removerUsuário() {
  try {
    await serviçoRemoverUsuário(usuárioLogado.cpf);
    setRedirecionar(true);
    mostrarToast(referênciaToast, "Removido com sucesso! Redirecionando ao Login.", "sucesso");
  } catch (error) { mostrarToast(referênciaToast, error.response.data.erro, "erro"); }
};

function executarOperação() {
  console.log("Valor da operação: ", confirmaçãoUsuário.operação); // Linha adicionada
  switch (confirmaçãoUsuário.operação) {
    case "salvar":
      finalizarCadastro();
      break;
    case "alterar":
      alterarUsuário({
        email: dados.email, senha: dados.senha, questão: dados.questão,
        resposta: dados.resposta, cor_tema: dados.cor_tema
      });
  }
};

```

```

        break;
    case "remover":
        removerUsuário();
        break;
    default:
        break;
    }
};

function ocultar() {
    if (!redirecionar) {
        setConfirmaçãoUsuário({});
        setMostrarModalConfirmação(false);
    }
};
return (
<div className={estilizarModal()}>
<Toast ref={referênciaToast} onHide={fecharToast} position="bottom-center"/>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(confirmaçãoUsuário?.cor_tema)}>Tipo de Perfil:</label>
<label>{exibirPerfilFormatado()}</label>
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(confirmaçãoUsuário?.cor_tema)}>
CPF -- nome de usuário:</label>
<label>{dados.cpf}</label>
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(confirmaçãoUsuário?.cor_tema)}>Nome Completo:</label>
<label>{dados.nome}</label>
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(confirmaçãoUsuário?.cor_tema)}>Email:</label>

<label>{dados.email}</label>
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(confirmaçãoUsuário?.cor_tema)}>
Questão de Segurança:</label>
<label>{dados.questão}</label>
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(confirmaçãoUsuário?.cor_tema)}>Resposta:</label>
<label>{dados.resposta}</label>
</div>
<div className={estilizarInlineFlex()}>

```

```
<Button label={labelOperação()} onClick={executarOperação}
className={estilizarBotão(confirmçãoUsuário?.cor_tema)}/>
<Button label="Corrigir" onClick={ocultar}
className={estilizarBotãoRemover(confirmçãoUsuário?.cor_tema)}/>
</div>
</div>
);
};
```

```
$$$ src.componentes.modais.modal-recuperar-acesso
import { useContext, useRef, useState } from "react";
import { useNavigate } from "react-router-dom";
import { Button } from "primereact/button";
import { Password } from "primereact/password";
import { Toast } from "primereact/toast";
import ContextoUsuário from "../../contextos/contexto-usuário";
import { serviçoAlterarUsuário } from "../../serviços/serviços-usuário";
import mostrarToast from "../../utilitários/mostrar-toast";
import { MostrarMensagemErro, checarListaVazia, validarCamposObrigatórios, validarConfirmaçãoSenha } from "../../utilitários/validações";
import { TAMANHOS, estilizarBotão, estilizarDivCampo, estilizarLabel, estilizarModal, estilizarPasswordInput, estilizarPasswordTextInputBorder } from "../../utilitários/estilos";
export default function ModalRecuperarAcesso() {
  const referênciaToast = useRef(null);
  const { cpfVerificado, setCpfVerificado, novaSenha, setNovaSenha, tokenRecuperação, setTokenRecuperação } = useContext(ContextoUsuário);
  const [dados, setDados] = useState({ senha: novaSenha?.senha || "", confirmação: novaSenha?.confirmação || "" });
  const [erros, setErros] = useState({});
  const navegar = useNavigate();
  function validarCampos() {
    let errosCamposObrigatórios = validarCamposObrigatórios(dados);
    let errosSenhasDiferentes = validarConfirmaçãoSenha(dados.senha, dados.confirmação);
    setErros({ ...errosCamposObrigatórios, ...errosSenhasDiferentes });
    return checarListaVazia(errosCamposObrigatórios) && checarListaVazia(errosSenhasDiferentes);
  };
  async function alterarSenha() {
    if (validarCampos()) {
      await serviçoAlterarUsuário({ cpf: cpfVerificado, senha: dados.senha, tokenRecuperação });
      setTokenRecuperação(null);
      mostrarToast(referênciaToast, "Senha redefinida com sucesso! Redirecionando ao Login...", "sucesso");
    }
  };
  function navegarLogarUsuário() {
    setCpfVerificado("");
    setNovaSenha({});
    navegar("/");
  };
  function alterarEstado(event) {
    const chave = event.target.name || event.value;
    const valor = event.target.value;
    setDados({ ...dados, [chave]: valor });
  };
  return (
    <div className={estilizarModal()}>
      <Toast ref={referênciaToast} onHide={navegarLogarUsuário} position="bottom-center" />
      <div className={estilizarDivCampo()}>
        <label className={estilizarLabel()}>Senha e Confirmar Senha*</label>
        <Password name="senha" inputClassName={estilizarPasswordTextInputBorder()} toggleMask className={estilizarPasswordInput(erros.senha)} size={TAMANHOS.SENHA}>
```

```
value={dados.senha} onChange={alterarEstado} />
<Password name="confirmação" inputClassName={estilizarPasswordInputBorder()} toggleMask
className={estilizarPasswordInput(erros.senha || erros.confirmação_senha)}
size={TAMANHOS.SENHA} feedback={false} value={dados.confirmação}
onChange={alterarEstado} />
<MostrarMensagemErro mensagem={erros.senha || erros.confirmação} />
</div>
<Button className={estilizarBotão()} label="Salvar" onClick={alterarSenha} />
</div>
);
};
```

```
$$$ src.componentes.menu-lateral
```

```
import { useContext, useEffect, useState } from "react";
import { useNavigate } from "react-router-dom";

import { Button } from "primereact/button";
import { Menu } from "primereact/menu";
import { Sidebar } from "primereact/sidebar";
import ContextoUsuário from "../contextos/contexto-usuário";
import formatarPerfil from "../utilitários/formatar-perfil";
import { estilizarBotão, estilizarColuna, estilizarGridColunaSidebar, estilizarGridSidebar, estilizarMenu, estilizarMenuLateralDesktop, estilizarMenuLateralMobile, estilizarSidebar, estilizarSubtítulo, estilizarTítulo } from "../utilitários/estilos";
export default function MenuLateral({ children }) {
  const { usuárioLogado, setUsuárioLogado } = useContext(ContextoUsuário);
  const [windowWidth, setWindowWidth] = useState(window.innerWidth);
  const [visible, setVisible] = useState(false);
  const tamanhoDesktop = windowWidth > 991;
  const navegar = useNavigate();
```

```
const opçõesGerenteMineradora = [
  { label: "Página Inicial", command: () => navegar("/pagina-inicial") },
  { label: "Menu", items: [
    { label: "Cadastrar Usuário", command: () => navegar("/atualizar-usuario"),
      disabled: usuárioLogado.status !== "ativo" },
    { label: "Cadastrar GerenteMineradora", command: () => navegar("/cadastrar-gerente-mineradora") },
    { label: "Administrar Patrocínios", command: () => navegar("/administrar-patrocinos") }, //
    { label: "Sair do Sistema", command: () => sairSistema() }
  ]},
];
```

```
const opçõesGerenteTecnologia = [
  { label: "Página Inicial", command: () => navegar("/pagina-inicial") },
  { label: "Menu", items: [
    { label: "Cadastrar Usuário", command: () => navegar("/atualizar-usuario"),
      disabled: usuárioLogado.status !== "ativo" },
    { label: "Cadastrar Gerente Tecnologia", command: () => navegar("/cadastrar-gerente-tecnologia") },
    { label: "Administrar Interesses", command: () => navegar("/administrar-Participacoes-Mineracao") },
    { label: "Sair do Sistema", command: () => sairSistema() }
  ]},
];
```

```
function sairSistema() {
  setUsuárioLogado({});
  navegar("/");
}

function opçõesMenu() {
```

```

switch (usuárioLogado.perfil) {
  case "gerente-mineradora": return opçõesGerenteMineradora;
  case "gerente-tecnologia": return opçõesGerenteTecnologia;
  default: return;
}

};

function redimensionarJanela() {
  setWindowWidth(window.innerWidth);
};

function MenuServiços() {
  if (tamanhoDesktop) {
    return (
      <div className={estilizarMenuLateralDesktop(usuárioLogado?.cor_tema)}>
        <h1 className={estilizarTítulo(usuárioLogado?.cor_tema)}>{usuárioLogado?.nome}</h1>
        <h2 className={estilizarSubtítulo(usuárioLogado?.cor_tema)}>
          {formatarPerfil(usuárioLogado?.perfil)}</h2>
        <Menu className={estilizarMenu()} model={opçõesMenu()} />
      </div>
    );
  } else return (
    <>
      <div className={estilizarMenuLateralMobile(usuárioLogado?.cor_tema)}>
        <Button className={estilizarBotão(usuárioLogado?.cor_tema)} icon="pi pi-bars" aria-label="Filter" onClick={() => setVisible(true)} />
        <h1 className={estilizarTítulo(usuárioLogado?.cor_tema)}>{usuárioLogado?.nome}</h1>
        <h2 className={estilizarSubtítulo(usuárioLogado?.cor_tema)}>
          {formatarPerfil(usuárioLogado?.perfil)}</h2>
      </div>
      <Sidebar className={estilizarSidebar()} visible={visible} onHide={() => setVisible(false)} showCloseIcon>
        <Menu className={estilizarMenu()} model={opçõesMenu()} />
      </Sidebar>
    </>
  );
}
};

useEffect(() => {
  window.addEventListener('resize', redimensionarJanela);
  return () => window.removeEventListener('resize', redimensionarJanela);
}, []);
return (
  <div className={estilizarGridSidebar(usuárioLogado?.cor_tema)}>
    <div className={estilizarGridColunaSidebar()}><MenuServiços/></div>
    <div className={estilizarColuna()}>{children}</div>
  </div>
);
}

```

```
$$$$ src.contextos.contexto-gerente-tecnologia
import { createContext, useState } from "react";
const ContextoGerenteTecnologia = createContext();
export default ContextoGerenteTecnologia;
export function ProvedorGerenteTecnologia({ children }) {
  const [participaçãoMineraçãoConsultado, setParticipaçãoMineraçãoConsultado] = useState({});
  const [PatrocínioConsultada, setPatrocínioConsultada] = useState({});
  const [PatrocínioSelecionada, setPatrocínioSelecionada] = useState({});
  const [PatrocínioCandidatura, setPatrocínioCandidatura] = useState({});
  const [gerenteMineradoraProponente, setGerenteMineradoraProponente] = useState({});
  return (
    <ContextoGerenteTecnologia.Provider value={{{
      participaçãoMineraçãoConsultado, setParticipaçãoMineraçãoConsultado, PatrocínioConsultada,
      setPatrocínioConsultada,
      PatrocínioSelecionada, setPatrocínioSelecionada, PatrocínioCandidatura, setPatrocínioCandidatura,
      gerenteMineradoraProponente, setGerenteMineradoraProponente
    }}>{children}</ContextoGerenteTecnologia.Provider>
  );
}
}
```

```
$$$$ src.contextos.contexto-gerente-mineradora
import { createContext, useState } from "react";
const ContextoGerenteMineradora = createContext();
export default ContextoGerenteMineradora;
export function ProvedorGerenteMineradora({ children }) {
  const [patrocínioConsultada, setPatrocínioConsultada] = useState({});
  const [participaçãoMineraçãoConsultado, setParticipaçãoMineraçãoConsultado] = useState(null);
  const [gerenteTecnologialInteressado, setGerenteTecnologialInteressado] = useState(null);
  return (
    <ContextoGerenteMineradora.Provider value={{patrocínioConsultada, setPatrocínioConsultada,
    participaçãoMineraçãoConsultado, setParticipaçãoMineraçãoConsultado,
    gerenteTecnologialInteressado, setGerenteTecnologialInteressado
  }}>{children}</ContextoGerenteMineradora.Provider>
  );
}
```

```
$$$$ src.contextos.contexto-usuário
import { createContext, useState } from "react";
const ContextoUsuário = createContext();
export default ContextoUsuário;
export function ProvedorUsuário({ children }) {
  const [usuárioLogado, setUsuárioLogado] = useState(null);
  const [confirmaçãoUsuário, setConfirmaçãoUsuário] = useState(null);
  const [mostrarModalConfirmação, setMostrarModalConfirmação] = useState(false);
  const [cpfVerificado, setCpfVerificado] = useState(null);
  const [novaSenha, setNovaSenha] = useState({});
  const [tokenRecuperação, setTokenRecuperação] = useState(null);

  return (
    <ContextoUsuário.Provider value={{ usuárioLogado, setUsuárioLogado,
      confirmaçãoUsuário, setConfirmaçãoUsuário, mostrarModalConfirmação,
      setMostrarModalConfirmação,
      cpfVerificado, setCpfVerificado, novaSenha, setNovaSenha, tokenRecuperação,
      setTokenRecuperação
    }}>{children}</ContextoUsuário.Provider>
  );
}
```

\$\$\$ src.imagens.image



```

$$$$ src.páginas.gerente-tecnologia.administrar-participações-mineração
import { useContext, useEffect, useRef, useState } from "react";
import { useNavigate } from "react-router-dom";
import { Button } from "primereact/button";
import { Card } from "primereact/card";
import { Column } from "primereact/column";
import { DataTable } from "primereact/datatable";
import { Divider } from "primereact/divider";
import { Dropdown } from "primereact/dropdown";
import { Toast } from "primereact/toast";
import { TriStateCheckbox } from "primereact/tristatecheckbox";

import ContextoUsuário from "../../contextos/contexto-usuário";
import ContextoGerenteTecnologia from "../../contextos/contexto-gerente-tecnologia";
import mostrarToast from "../../utilitários/mostrar-toast";
import { serviçoBuscarParticipaçõesMineraçãoGerenteTecnologia } from "../../serviços/serviços-gerente-tecnologia";

import {
    TAMANHOS,
    estilizarBotão,
    estilizarBotãoRetornar,
    estilizarBotãoTabela,
    estilizarCard,
    estilizarColumnHeader,
    estilizarColunaConsultar,
    estilizarDataTable,
    estilizarDataTablePaginator,
    estilizarDivider,
    estilizarFilterMenu,
    estilizarFlex,
    estilizarTriStateCheckbox
} from "../../utilitários/estilos";
}

export default function AdministrarParticipaçõesMineração() {
    const referênciaToast = useRef(null);
    const navegar = useNavigate();

    const { usuárioLogado } = useContext(ContextoUsuário);
    const { participaçãoMineraçãoConsultado, setParticipaçãoMineraçãoConsultado, setPatrocínioSelecionada } =
        useContext(ContextoGerenteTecnologia);

    const [listaParticipaçõesMineração, setListaParticipaçõesMineração] = useState([]);

    // Opções para filtros
    const opçõesCategoria = [
        { label: "Extração", value: "Extração" },
        { label: "Exploração", value: "Exploração" },
        { label: "Consultoria", value: "Consultoria" },
        { label: "Pesquisa Mineral", value: "Pesquisa Mineral" }
    ];

    // --- 1. ADICIONADO OPÇÕES DE RESULTADO ---
    const opçõesResultado = [
        { label: "Sucesso", value: "Sucesso" },
        { label: "Parcial", value: "Parcial" },
        { label: "Falha", value: "Falha" }
    ];

    // Navegar para a página inicial
    function retornarPáginaInicial() {
        navegar("/pagina-inicial");
    }

    // Navegar para adicionar novo interesse
    function adicionarParticipaçãoMineração() {
        setParticipaçãoMineraçãoConsultado(null);
        setPatrocínioSelecionada(null);
        navegar("./cadastrar-participacao-mineracao");
    }

    // Botão de consulta de interesse
    function ConsultarTemplate(participação) {
        function consultar() {
            setParticipaçãoMineraçãoConsultado(participação);
        }

```

```

        setPatrocínioSelecionada(null);
        navegar("../cadastrar-participacao-mineracao");
    }

    return (
        <Button
            icon="pi pi-search"
            className={estilizarBotãoTabela(
                usuárioLogado.cor_tema,
                participaçãoMineraçãoConsultado?.id === participação.id
            )}
            tooltip="Consultar Participação Mineração"
            tooltipOptions={{ position: 'top' }}
            onClick={consultar}
        />
    );
}

// Dropdown de categorias (Já existia)
function DropdownAreaTemplate(opções) {
    function alterarFiltroDropdown(event) {
        return opções.filterCallback(event.value, opções.index);
    }

    return (
        <Dropdown
            value={opções.value}
            options={opçõesCategoria}
            placeholder="Selecione"
            onChange={alterarFiltroDropdown}
            showClear
        />
    );
}

// --- 2. ADICIONADA FUNÇÃO PARA FILTRO DE RESULTADO ---
function DropdownResultadoTemplate(opções) {
    function alterarFiltroDropdown(event) {
        return opções.filterCallback(event.value, opções.index);
    }

    return (
        <Dropdown
            value={opções.value}
            options={opçõesResultado}
            placeholder="Selecione"
            onChange={alterarFiltroDropdown}
            showClear
        />
    );
}

// Buscar interesses do gerente tecnologia ao carregar o componente
useEffect(() => {
    let desmontado = false;

    async function buscarParticipaçõesMineração() {
        try {
            console.log("[CLIENTE] 🔍Buscando participações do CPF:", usuárioLogado.cpf);
            const response = await serviçoBuscarParticipaçõesMineraçãoGerenteTecnologia(usuárioLogado.cpf);

            console.log("[CLIENTE] 🔍Resposta bruta da API:", response);
            console.log("[CLIENTE] 🔍Dados recebidos (response.data):", response?.data);

            if (Array.isArray(response?.data)) {
                console.log(`[CLIENTE] ✅${response.data.length} participações recebidas`);
            } else {
                console.warn("[CLIENTE] O retorno não é um array:", typeof response?.data);
            }

            if (!desmontado && response.data) setListaParticipaçõesMineração(response.data);
        } catch (error) {
            console.error("[CLIENTE] ❌Erro ao buscar participações:", error);
            mostrarToast(referênciaToast, error.response?.data?.erro || "Erro desconhecido", "error");
        }
    }
})

```

```

    }

    buscarParticipaçõesMineração();
    return () => (desmontado = true);
}, [usuárioLogado.cpf]);

return (
<div className={estilizarFlex()}>
  <Toast ref={referênciaToast} position="bottom-center" />
  <Card title="Administrar Participações de Mineração" className={estilizarCard(usuárioLogado.cor_tema)}>
    <DataTable
      dataKey="id"
      size="small"
      paginator
      rows={TAMANHOS.MAX_LINHAS_TABELA}
      emptyMessage="Nenhuma participação encontrada."
      value={listaParticipaçõesMineração}
      responsiveLayout="scroll"
      breakpoint="490px"
      removableSort
      className={estilizarDataTable()}
      paginatorClassName={estilizarDataTablePaginator(usuárioLogado.cor_tema)}
    >
    <Column
      bodyClassName={estilizarColunaConsultar()}
      body={ConsultarTemplate}
      headerClassName={estilizarColumnHeader(usuárioLogado.cor_tema)}
    />
    {/* Campos reais conforme o backend */}
    <Column
      field="título"
      header="Título"
      sortable
      filter
      showFilterOperator={false}
      headerClassName={estilizarColumnHeader(usuárioLogado.cor_tema)}
    />
    {/* --- 3. ADICIONADA COLUNA CATEGORIA --- */}
    <Column
      field="categoria"
      header="Categoria"
      sortable
      filter
      filterMatchMode="equals"
      filterElement={DropdownÁreaTemplate}
      showClearButton={false}
      showFilterOperator={false}
      showFilterMatchModes={false}
      filterMenuClassName={estilizarFilterMenu()}
      showFilterMenuOptions={false}
      headerClassName={estilizarColumnHeader(usuárioLogado.cor_tema)}
    />
    <Column
      field="área_atuação"
      header="Área de Atuação"
      sortable
      filter
      showFilterOperator={false}
      headerClassName={estilizarColumnHeader(usuárioLogado.cor_tema)}
    />
    {/* --- 4. ADICIONADA COLUNA RESULTADO --- */}
    <Column
      field="resultado"
      header="Resultado"
      sortable
      filter
      filterMatchMode="equals"
      filterElement={DropdownResultadoTemplate}
      showClearButton={false}
      showFilterOperator={false}
      showFilterMatchModes={false}
    />

```

```

        filterMenuClassName={estilizarFilterMenu()}
        showFilterMenuOptions={false}
        headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
    />

    <Column
        field="data_início"
        header="Data de Início"
        sortable
        headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
        body={(rowData) => new Date(rowData.data_início).toLocaleDateString()}
    />

    {/* Descrição é muito longa para filtro, talvez remover? */}
    <Column
        field="descrição"
        header="Descrição"
        sortable
        headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
    />
</DataTable>

<Divider className={estilizarDivider()} />
<Button
    className={estilizarBotãoRetornar()}
    label="Retornar"
    onClick={retornarPáginaInicial}
/>
<Button
    className={estilizarBotão()}
    label="Adicionar"
    onClick={adicionarParticipaçãoMineração}
/>
</Card>
</div>
);
}
}

```

```

$$$$ src.páginas.gerente-tecnologia.cadastrar-gerente-tecnologia
import { useContext, useEffect, useRef, useState } from "react";
import { useNavigate } from "react-router-dom";
import { Button } from "primereact/button";
import { Card } from "primereact/card";
import { Divider } from "primereact/divider";
import { Dropdown } from "primereact/dropdown";
import { InputMask } from "primereact/inputmask";
import { InputText } from "primereact/inputtext";
import { Toast } from "primereact/toast";
import ContextoUsuário from "../../contextos/contexto-usuário";
import { ANO_MÁSCARA, TELEFONE_MÁSCARA } from "../../utilitários/máscaras";
import { serviçoCadastrarGerenteTecnologia, serviçoAtualizarGerenteTecnologia, serviçoBuscarGerenteTecnologia }
from "../../serviços/serviços-gerente-tecnologia";
import mostrarToast from "../../utilitários/mostrar-toast";
import { MostrarMensagemErro, checarListaVazia, validarCamposObrigatórios }
from "../../utilitários/validações";
import { TAMANHOS, estilizarBotão, estilizarBotãoRetornar, estilizarCard, estilizarDivCampo, estilizarDivider, estilizarDropdown, estilizarFlex, estilizarInlineFlex, estilizarInputMask, estilizarInputText, estilizarLabel } from "../../utilitários/estilos";

export default function CadastrarGerenteTecnologia() {
  const referênciaToast = useRef(null);
  const { usuárioLogado, setUsuárioLogado } = useContext(ContextoUsuário);
  const [dados, setDados] = useState({ curso: "", ano_ingresso: "", data_nascimento: "", telefone: "" });

  const [erros, setErros] = useState({});
  const [cpfExistente, setCpfExistente] = useState(false);
  const navegar = useNavigate();
  const opçõesCurso = [{ label: "Gerente Tecnologia" , value: "Gerente Tecnologia" }, { label: "Gerente Inovação" , value: "Gerente Inovação" }, { label: "Engenheiro Sistemas Sênior" , value: "Engenheiro Sistemas Sênior" }, { label: "Líder Equipe Desenvolvimento" , value: "Líder Equipe Desenvolvimento" }];

  function alterarEstado(event) {
    const chave = event.target.name || event.value;
    const valor = event.target.value;
    setDados({ ...dados, [chave]: valor });
  };

  // Adicione a função 'validarCampos' que estava faltando
  function validarCampos() {
    let errosCamposObrigatórios = validarCamposObrigatórios(dados);

    // Log para ver os erros antes de definir no estado
    console.log("Erros de validação encontrados:", errosCamposObrigatórios);

    setErros(errosCamposObrigatórios);
    return checarListaVazia(errosCamposObrigatórios);
  }
}

```

```

function títuloFormulário() {
    console.log("usuárioLogado dentro de títuloFormulário:", usuárioLogado.cadastrado);
    if (usuárioLogado?.cadastrado) return "Alterar Gerente Tecnologia";
    else return "Cadastrar Gerente Tecnologia";
};

async function cadastrarGerenteTecnologia() {
    if (validarCampos()) {
        try {
            // Log para ver os dados antes de enviar
            console.log("Enviando dados do gerente de tecnologia:", dados);

            const response = await serviçoCadastrarGerenteTecnologia({
                ...dados,
                usuário_info: usuárioLogado,
                curso: dados.curso,
                ano_ingresso: dados.ano_ingresso,
                data_nascimento: dados.data_nascimento,
                telefone: dados.telefone
            });

            if (response.data) {
                console.log("Cadastro realizado com sucesso!", response.data);
                setUsuárioLogado(usuário => ({
                    ...usuário,
                    status: response.data.status,
                    token: response.data.token
                }));
                // Atualize a mensagem do toast
                mostrarToast(referênciaToast, "Gerente de Tecnologia cadastrado com sucesso!",
                "sucesso");
            }
        } catch (error) {
            console.error("Erro ao cadastrar o gerente de tecnologia:", error);

            // Se o erro tiver uma resposta com dados, logue a resposta de erro
            if (error.response) {
                console.error("Erro na resposta do servidor:", error.response);
                setCpfExistente(true);
                mostrarToast(referênciaToast, error.response.data.erro, "erro");
            } else {
                // Se o erro não tiver resposta, logue ele diretamente
                console.error("Erro sem resposta do servidor:", error.message);
                mostrarToast(referênciaToast, "Ocorreu um erro inesperado", "erro");
            }
        }
    }
}

```

```

        }
    }

async function atualizarGerenteTecnologia() {
    console.log("atualizarGerenteTecnologia chamado");
    if (validarCampos()) {
        try {
            const response = await serviçoAtualizarGerenteTecnologia({ ...dados, cpf: usuárioLogado.cpf });
            if (response) mostrarToast(referênciaToast, "Gerente de Tecnologia atualizado com sucesso!", "sucesso");
        } catch (error) { mostrarToast(referênciaToast, error.response.data.erro, "erro"); }
    }
};

function labelBotãoSalvar() {
    if (usuárioLogado?.cadastrado) return "Alterar";
    else return "Cadastrar";
};

function açãoBotãoSalvar() {
    console.log(usuárioLogado);
    console.log("açãoBotãoSalvar chamado");
    if (usuárioLogado?.cadastrado) atualizarGerenteTecnologia();
    else cadastrarGerenteTecnologia();
};

function redirecionar() {
    if (cpfExistente) {
        setUsuárioLogado(null);
        navegar("/criar-usuario");
    } else {
        setUsuárioLogado(usuárioLogado => ({ ...usuárioLogado, cadastrado: true }));
        navegar("/pagina-inicial");
    }
};

useEffect(() => {
    let desmontado = false;

    async function buscarDadosGerenteTecnologia() {
        try {
            console.log("Buscando dados do gerente de tecnologia para o CPF:", usuárioLogado.cpf);
            console.log("usuárioLogado:", usuárioLogado);

            // Teste se a função existe antes de chamar
            if (typeof serviçoBuscarGerenteTecnologia !== 'function') {
                const msg = `Função serviçoBuscarGerenteTecnologia não encontrada! Esperado em: front-end/src/serviços/serviços-gerente-tecnologia.js`;
                console.error(msg);
                mostrarToast(referênciaToast, msg, "erro");
                return;
            }
        }
    }
});

```

```

const response = await serviçoBuscarGerenteTecnologia(usuarioLogado.cpf);
console.log("Teste");

console.log("Resposta recebida do serviço de busca:", response);

if (!desmontado && response?.data) {
    console.log("Dados do gerente de tecnologia encontrados:", response.data);
    setDados(dados => ({
        ...dados,
        curso: response.data.curso,
        ano_ingresso: response.data.ano_ingresso,
        data_nascimento: response.data.data_nascimento,
        telefone: response.data.telefone
    }));
} else {
    console.log("Nenhum dado encontrado para este gerente de tecnologia");
}
} catch (error) {
    console.error("Erro ao buscar os dados do gerente de tecnologia:", error);
    if (error.response) {
        console.error("Erro na resposta do servidor:", error.response);
        const erro = error.response.data.erro;
        if (erro) mostrarToast(referênciaToast, erro, "erro");
    } else {
        console.error("Erro sem resposta do servidor:", error.message);
        mostrarToast(referênciaToast, "Erro inesperado ao buscar os dados", "erro");
    }
}
}

if (usuarioLogado?.cadastrado) {
    buscarDadosGerenteTecnologia();
}

return () => desmontado = true;
}, [usuarioLogado?.cadastrado, usuarioLogado.cpf]);

return (
<div className={estilizarFlex()}>
<Toast ref={referênciaToast} onHide={redirecionar} position="bottom-center"/>
<Card title={títuloFormulário()} className={estilizarCard(usuarioLogado.cor_tema)}>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(usuarioLogado.cor_tema)}>Curso*</label>
<Dropdown name="curso" className={estilizarDropdown(erros.curso, usuarioLogado.cor_tema)}>
value={dados.curso} options={opçõesCurso} onChange={alterarEstado}
placeholder="-- Selecione --"/>
<MostrarMensagemErro mensagem={erros.curso}/>
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(usuarioLogado.cor_tema)}>Ano de Ingresso*</label>
<InputMask name="ano_ingresso" autoClear size={TAMANHOS.ANO} onChange={alterarEstado}>
className={estilizarInputMask(erros.ano_ingresso, usuarioLogado.cor_tema)}</InputMask>
</div>

```

```
mask={ANO_MÁSCARA} value={dados.ano_ingresso}/>
<MostrarMensagemErro mensagem={erros.ano_ingresso}/>
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(usuarioLogado.cor_tema)}>Data de Nascimento*</label>
<InputText name="data_nascimento" type="date" value={dados.data_nascimento}
className={estilizarInputText(erros.data_nascimento, usuárioLogado.cor_tema)}
onChange={alterarEstado}/>
<MostrarMensagemErro mensagem={erros.data_nascimento}/>
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(usuarioLogado.cor_tema)}>Telefone*</label>
<InputMask name="telefone" autoClear size={TAMANHOS.TELEFONE} onChange={alterarEstado}
className={estilizarInputMask(erros.telefone, usuárioLogado.cor_tema)}
mask={TELEFONE_MÁSCARA} value={dados.telefone}/>
<MostrarMensagemErro mensagem={erros.telefone}/>
</div>
<Divider className={estilizarDivider(dados.cor_tema)}/>
<div className={estilizarInlineFlex()}>
<Button className={estilizarBotãoRetornar()} label="Retornar" onClick={redirecionar} />
<Button className={estilizarBotão()} label={labelBotãoSalvar()} onClick={açãoBotãoSalvar}/>
</div>
</Card>
</div>
);
};
```

```

$$$$ src.páginas.gerente-tecnologia.cadastrar-participação-mineração
import { useContext, useRef, useState } from "react";
import { useNavigate } from "react-router-dom";
import { Button } from "primereact/button";
import { Card } from "primereact/card";
import { Checkbox } from "primereact/checkbox";
import { Divider } from "primereact/divider";
import { InputText } from "primereact/inputtext";
import { InputTextarea } from "primereact/inputtextarea";
import { Toast } from "primereact/toast";
import { Dropdown } from "primereact/dropdown"; // 1. IMPORTAR O DROPODOWN

import ContextoUsuário from "../../contextos/contexto-usuário";
import ContextoGerenteTecnologia from "../../contextos/contexto-gerente-tecnologia";

import {
    serviçoCadastrarParticipaçãoMineração,
    serviçoRemoverParticipaçãoMineração,
} from "../../../serviços/serviços-gerente-tecnologia";

import mostrarToast from "../../utilitários/mostrar-toast";
import {
    MostrarMensagemErro,
    checarListaVazia,
    validarCamposObrigatórios,
} from "../../utilitários/validações";

import {
    estilizarBotão,
    estilizarBotãoRetornar,
    estilizarBotãoRemover,
    estilizarCard,
    estilizarCheckbox,
    estilizarDivCampo,
    estilizarDivider,
    estilizarFlex,
    estilizarInlineFlex,
    estilizarInputText,
    estilizarInputTextarea,
    estilizarLabel,
} from "../../utilitários/estilos";

export default function CadastrarParticipaçãoMineração() {
    const referênciaToast = useRef(null);
    const { usuárioLogado } = useContext(ContextoUsuário);
    const {
        participaçãoMineraçãoConsultado,
        PatrocínioSelecionada,
        setPatrocínioConsultada
    } = useContext(ContextoGerenteTecnologia);

    // 2. DEFINIR AS OPÇÕES (COM BASE NA SUA ENTIDADE)
    const opçõesCategoria = [

```

```

{ label: "Extração", value: "Extração" },
{ label: "Exploração", value: "Exploração" },
{ label: "Consultoria", value: "Consultoria" },
{ label: "Pesquisa Mineral", value: "Pesquisa Mineral" }
];

const opçõesResultado = [
  { label: "Sucesso", value: "Sucesso" },
  { label: "Parcial", value: "Parcial" },
  { label: "Falha", value: "Falha" }
];

```

// 3. ADICIONAR CAMPOS AO ESTADO

```

const [dados, setDados] = useState({
  id_patrocínio: PatrocínioSelecionada?.id || "",
  título: participaçãoMineraçãoConsultado?.título || "", // Título da participação
  necessidade_bolsa: participaçãoMineraçãoConsultado?.necessidade_bolsa || false,
  justificativa: participaçãoMineraçãoConsultado?.justificativa || "",
  área_atuação: participaçãoMineraçãoConsultado?.área_atuação || "",
  data_início: participaçãoMineraçãoConsultado?.data_início || "",
  descrição: participaçãoMineraçãoConsultado?.descrição || "",
  categoria: participaçãoMineraçãoConsultado?.categoria || "", // <-- ADICIONADO
  resultado: participaçãoMineraçãoConsultado?.resultado || "" // <-- ADICIONADO
});

```

```

const [erros, setErros] = useState({});
const navegar = useNavigate();

function alterarEstado(event) {
  // Esta função já está correta para lidar com Dropdown também
  const chave = event.target.name;
  const valor = event.target.type === 'checkbox' ? event.target.checked : event.target.value;
  setDados({ ...dados, [chave]: valor });
}

```

// 4. ADICIONAR CAMPOS À VALIDAÇÃO

```

function validarCampos() {
  const { título, justificativa, categoria, resultado, área_atuação, data_início } = dados; // <-- ADICIONADOS
  const errosCamposObrigatórios = validarCamposObrigatórios({
    título,
    justificativa,
    categoria,
    resultado,
    área_atuação,
    data_início
  });
  setErros(errosCamposObrigatórios);
  return checarListaVazia(errosCamposObrigatórios);
}

```

```

function patrocínioLabel() {
  if (participaçãoMineraçãoConsultado?.título_patrocinio || PatrocínioSelecionada)
    return "Patrocínio Selecionado*";
  else return "Selecione um Patrocínio*";
}

function consultarPatrocínioParticipaçãoMineração() {
  console.log("Participação atual (consultarPatrocínio):", participaçãoMineraçãoConsultado);

  // O seu backend (buscarParticipaçõesMineraçãoGerenteTecnologia) agora retorna 'patrocínios'
  // (plural/array)
  const patrocinioArray = participaçãoMineraçãoConsultado?.patrocínios;

  // Pega o primeiro patrocínio do array, se existir
  const patrocinio = (Array.isArray(patrãoArray) && patrocinioArray.length > 0) ?
    patrocinioArray[0] : null;

  if (patrocinio) {
    console.log("Patrocínio encontrado:", patrocinio);
    setPatrocínioConsultada(patrão);
    navegar("../consultar-patrão");
  } else {
    console.warn("Nenhum patrocínio associado encontrado para esta participação.");
    mostrarToast(referênciaToast, "Esta participação não possui um patrocínio associado.", "info");
    setPatrocínioConsultada(null);
  }
};

function pesquisarPatrocínios() {
  navegar("../pesquisar-patrão");
}

function retornarAdministrarParticipações() {
  navegar("../administrar-participações-mineração");
}

async function cadastrarParticipação() {
  if (validarCampos()) {
    try {
      await serviçoCadastrarParticipaçãoMineração({
        ...dados,
        cpf: usuárioLogado.cpf,
        id_patrão: PatrocínioSelecionada?.id
      });
      mostrarToast(referênciaToast, "Participação cadastrada com sucesso!", "sucesso");
    } catch (error) {
      mostrarToast(referênciaToast, error.response.data.erro, "erro");
    }
  }
}

async function removerParticipação() {

```

```

try {
    await serviçoRemoverParticipaçãoMineração(participaçãoMineraçãoConsultado.id);
    mostrarToast(referênciaToast, "Participação removida com sucesso!", "sucesso");
} catch (error) {
    mostrarToast(referênciaToast, error.response.data.erro, "erro");
}
}

function BotõesAções() {
    if (participaçãoMineraçãoConsultado) {
        return (
            <div className={estilizarInlineFlex()}>
                <Button
                    className={estilizarBotãoRetornar()}
                    label="Retornar"
                    onClick={retornarAdministrarParticipações}
                />
                <Button
                    className={estilizarBotãoRemover()}
                    label="Remover"
                    onClick={removerParticipação}
                />
                <Button className={estilizarBotão()} label="Patrocínio"
                    onClick={consultarPatrocínioParticipaçãoMineração}/>
            </div>
        );
    } else {
        return (
            <div className={estilizarInlineFlex()}>
                <Button
                    className={estilizarBotãoRetornar()}
                    label="Retornar"
                    onClick={retornarAdministrarParticipações}
                />
                <Button
                    className={estilizarBotão()}
                    label="Cadastrar"
                    onClick={cadastrarParticipação}
                />
            </div>
        );
    }
}

function títuloFormulário() {
    if (participaçãoMineraçãoConsultado) return "Remover Participação";
    else return "Cadastrar Participação";
}

function PatrocínioInputText() {
    // 1. Verifica o patrocínio selecionado (quando você clica em "Selecionar/Substituir")
    if (PatrocínioSelecionada?.justificativa) {
        return (

```

```

<InputText
  name="nome_patrocínio"
  className={estilizarInputText(erros.nome_patrocínio, 400, usuárioLogado.cor_tema)}
  value={PatrocínioSelecionada?.justificativa}
  disabled
/>
);

// 2. Verifica o patrocínio de uma participação existente (quando você está editando)
//   Ajustado para o novo formato de 'patrocínios' (array)
} else if (participaçãoMineraçãoConsultado?.patrocínios &&
participaçãoMineraçãoConsultado.patrócínios.length > 0) {
  return (
    <InputText
      name="nome_patrocínio"
      className={estilizarInputText(erros.nome_patrocínio, 400, usuárioLogado.cor_tema)}
      value={participaçãoMineraçãoConsultado.patrócínios[0].justificativa} // Pega do primeiro
patorocínio
      disabled
    />
  );
} else return null; // Retorna nulo se nenhum patrocínio for encontrado
}

function BotãoSelecionar() {
if (!PatrocínioSelecionada && !participaçãoMineraçãoConsultado) {
  return (
    <Button
      className={estilizarBotão()}
      label="Selecionar"
      onClick={pesquisarPatrocínios}
    />
  );
} else if (PatrocínioSelecionada) {
  return (
    <Button
      className={estilizarBotão()}
      label="Substituir"
      onClick={pesquisarPatrocínios}
    />
  );
// Se estiver editando, não mostre "Selecionar" ou "Substituir",
// o patrocínio é visto pelo botão "Patorocínio"
} else if (participaçãoMineraçãoConsultado) {
  return null;
}

// Fallback caso nenhuma condição acima seja atendida (modo de cadastro inicial)
return (
  <Button
    className={estilizarBotão()}
    label="Selecionar"
    onClick={pesquisarPatrocínios}
  />

```

```

);
}

return (
<div className={estilizarFlex()}>
<Toast
ref={referênciaToast}
onHide={retornarAdministrarParticipações}
position="bottom-center"
/>
<Card title={títuloFormulário()} className={estilizarCard(usuarioLogado.cor_tema)}>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(usuarioLogado.cor_tema)}>
{patrocínioLabel()}
</label>
<BotãoSelecionar />
<PatrocínioInputText />
<MostrarMensagemErro mensagem={erros.id} />
</div>

<div className={estilizarDivCampo()}>
<label className={estilizarLabel(usuarioLogado.cor_tema)}>
Necessidade de Bolsa*:
</label>
<Checkbox
name="necessidade_bolsa"
checked={dados.necessidade_bolsa}
className={estilizarCheckbox()}
onChange={alterarEstado}
/>
</div>

<div className={estilizarDivCampo()}>
<label className={estilizarLabel(usuarioLogado.cor_tema)}>
Título*:
</label>
<InputText
name="título"
value={dados.título}
className={estilizarInputText(erros.título, 400, usuarioLogado.cor_tema)}
onChange={alterarEstado}
/>
<MostrarMensagemErro mensagem={erros.título} />
</div>

/* 5. ADICIONAR O JSX DO DROPDOWN DE CATEGORIA */
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(usuarioLogado.cor_tema)}>
Categoria*:
</label>
<Dropdown
name="categoria"

```

```

        value={dados.categoria}
        options={opçõesCategoria}
        onChange={alterarEstado}
        placeholder="Selecione uma categoria"
        // Você pode precisar de um 'estilizarDropdown' ou usar o 'estilizarInputText'
        className={estilizarInputText(erros.categoria, 400, usuárioLogado.cor_tema)}
    />
    <MostrarMensagemErro mensagem={erros.categoria} />
</div>

<div className={estilizarDivCampo()}>
    <label className={estilizarLabel(usuarioLogado.cor_tema)}>
        Área de Atuação*:
    </label>
    <InputText
        name="área_atuação"
        value={dados.área_atuação || ""}
        className={estilizarInputText(erros.área_atuação, 400, usuárioLogado.cor_tema)}
        onChange={alterarEstado}
    />
    <MostrarMensagemErro mensagem={erros.área_atuação} />
</div>

<div className={estilizarDivCampo()}>
    <label className={estilizarLabel(usuarioLogado.cor_tema)}>
        Descrição:
    </label>
    <InputTextarea
        name="descrição"
        value={dados.descrição || ""}
        className={estilizarInputTextarea(erros.descrição, usuárioLogado.cor_tema)}
        onChange={alterarEstado}
        autoResize
        cols={40}
    />
    <MostrarMensagemErro mensagem={erros.descrição} />
</div>

/* 5. ADICIONAR O JSX DO DROPDOWN DE RESULTADO */
<div className={estilizarDivCampo()}>
    <label className={estilizarLabel(usuarioLogado.cor_tema)}>
        Resultado*:
    </label>
    <Dropdown
        name="resultado"
        value={dados.resultado}
        options={opçõesResultado}
        onChange={alterarEstado}
        placeholder="Selecione um resultado"
        className={estilizarInputText(erros.resultado, 400, usuárioLogado.cor_tema)}
    />

```

```

<MostrarMensagemErro mensagem={erros.resultado} />
</div>

<div className={estilizarDivCampo()}>
  <label className={estilizarLabel(usuarioLogado.cor_tema)}>
    Data de Início*:
  </label>
  <InputText
    type="date"
    name="data_início"
    value={dados.data_início || ""}
    className={estilizarInputText(erros.data_início, 200, usuárioLogado.cor_tema)}
    onChange={alterarEstado}
  />
  <MostrarMensagemErro mensagem={erros.data_início} />
</div>

<div className={estilizarDivCampo()}>
  <label className={estilizarLabel(usuarioLogado.cor_tema)}>
    Justificativa*:
  </label>
  <InputTextarea
    name="justificativa"
    value={dados.justificativa}
    className={estilizarInputTextarea(erros.justificativa, usuárioLogado.cor_tema)}
    onChange={alterarEstado}
    autoResize
    cols={40}
  />
  <MostrarMensagemErro mensagem={erros.justificativa} />
</div>

<Divider className={estilizarDivider()} />
<BotõesAções />
</Card>
</div>
);
}

```

```

$$$$ src.páginas.gerente-tecnologia.consultar-gerente-mineradora
import { useContext } from "react";
import { useNavigate } from "react-router-dom";
import { Button } from "primereact/button";
import { Card } from "primereact/card";
import { Divider } from "primereact/divider";
import { InputNumber } from "primereact/inputnumber";
import { InputText } from "primereact/inputtext";
import ContextoUsuário from "../../contextos/contexto-usuário";
import ContextoGerenteTecnologia from "../../contextos/contexto-gerente-tecnologia";

import { estilizarBotãoRetornar, estilizarCard, estilizarDivCampo, estilizarDivider, estilizarFlex,
estilizarInlineFlex, estilizarInputText, estilizarLabel } from "../../utilitários/estilos";

export default function ConsultarGerenteMineradora() {
const { usuárioLogado } = useContext(ContextoUsuário);
const { gerenteMineradoraProponente } = useContext(ContextoGerenteTecnologia);
const dados = { nome_professor: gerenteMineradoraProponente?.usuário?.nome,
titulação: gerenteMineradoraProponente?.titulação,
anos_experiência_empresarial: gerenteMineradoraProponente?.anos_experiência_empresarial };
const navegar = useNavigate();
function retornarConsultarProposta() { navegar("/consultar-patrocínio"); }

return (
<div className={estilizarFlex()}>
<Card title="Consultar Gerente Mineradora" className={estilizarCard(usuárioLogado.cor_tema)}>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(usuárioLogado.cor_tema)}>Gerente Mineradora*</label>
<InputText name="nome_professor"
className={estilizarInputText(null, 400, usuárioLogado.cor_tema)}>
value={dados.nome_professor} disabled/>
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(usuárioLogado.cor_tema)}>Titulação*</label>
<InputText name="titulação"
className={estilizarInputText(null, 150, usuárioLogado.cor_tema)}>
value={dados.titulação} autoResize disabled/>
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(usuárioLogado.cor_tema)}>Anos de Experiência Empresarial*</label>
<InputNumber name="anos_experiência_empresarial" size={5}
value={dados.anos_experiência_empresarial}>
inputClassName={estilizarInputText(null, usuárioLogado.cor_tema)}
mode="decimal" min={1} disabled/>
</div>

```

```
<Divider className={estilizarDivider(dados.cor_tema)}/>
<div className={estilizarInlineFlex()}>
<Button className={estilizarBotãoRetornar()} label="Retornar"
onClick={retornarConsultarProposta}/>
</div>
</Card>
</div>
);
};
```

```
$$$$ src.páginas.gerente-tecnologia.consultar-patrocínio
// ARQUIVO: consultar-patrocínio.jsx

import { useContext } from "react";
import { useNavigate } from "react-router-dom";
import { Button } from "primereact/button";
import { Card } from "primereact/card";
import { Checkbox } from "primereact/checkbox";
import { Divider } from "primereact/divider";
import { InputText } from "primereact/inputtext";
import { InputTextarea } from "primereact/inputtextarea";

import ContextoUsuário from "../../contextos/contexto-usuário";
import ContextoGerenteTecnologia from "../../contextos/contexto-gerente-tecnologia";
import {
    estilizarBotão,
    estilizarBotãoRetornar,
    estilizarCard,
    estilizarCheckbox,
    estilizarDivCampo,
    estilizarDivider,
    estilizarFlex,
    estilizarInlineFlex,
    estilizarInputText,
    estilizarInputTextarea,
    estilizarLabel
} from "../../utilitários/estilos";

export default function ConsultarPatrocínio() {
    const { usuárioLogado } = useContext(ContextoUsuário);
    const { PatrocínioConsultada, setGerenteMineradoraProponente } =
        useContext(ContextoGerenteTecnologia);
    const navegar = useNavigate();
    console.log("Dados recebidos do backend (PatrocínioConsultada):", PatrocínioConsultada);
    const dados = {
        gerente: PatrocínioConsultada?.gerentemineradora?.usuário?.nome || "",
        justificativa: PatrocínioConsultada?.justificativa || "",
        necessidade_bolsa: PatrocínioConsultada?.necessidade_bolsa || false,
        categoria: PatrocínioConsultada?.categoria_participacao || ""
    };

    function retornar() {
        navegar("../pesquisar-patrocínios");
    }

    function consultarGerenteMineradora() {
        if (PatrocínioConsultada)
            setGerenteMineradoraProponente(PatrocínioConsultada.gerentemineradora);
        navegar("../consultar-gerente-mineradora");
    };

    return (
        <div className={estilizarFlex()}>
```

```

<Card
  title="Consultar Patrocínio"
  className={estilizarCard(usuárioLogado.cor_tema)}
>
  {/* Gerente */}
  <div className={estilizarDivCampo()}>
    <label className={estilizarLabel(usuárioLogado.cor_tema)}>Gerente Responsável:</label>
    <InputText name="gerente" value={dados.gerente} disabled />
  </div>

  {/* Justificativa */}
  <div className={estilizarDivCampo()}>
    <label className={estilizarLabel(usuárioLogado.cor_tema)}>Justificativa:</label>
    <InputTextarea name="justificativa" value={dados.justificativa} disabled />
  </div>

  {/* Necessidade de Bolsa */}
  <div className={estilizarDivCampo()}>
    <label className={estilizarLabel(usuárioLogado.cor_tema)}>Necessita Bolsa:</label>
    <Checkbox name="necessidade_bolsa" checked={dados.necessidade_bolsa} disabled />
  </div>

  {/* Categoria / Tipo */}
  <div className={estilizarDivCampo()}>
    <label className={estilizarLabel(usuárioLogado.cor_tema)}>Categoria:</label>
    <InputText name="categoria" value={dados.categoria} disabled />
  </div>

  <Divider className={estilizarDivider()} />

  <div className={estilizarInlineFlex()}>
    <Button
      label="Retornar"
      className={estilizarBotãoRetornar()}
      onClick={retornar}
    />
  </div>

  <div className={estilizarInlineFlex()}>
    <Button className={estilizarBotão()} label="Gerente Mineradora"
      onClick={consultarGerenteMineradora}/>
  </div>
</Card>
</div>
);
}

```

```

$$$$ src.páginas.gerente-tecnologia.pesquisar-patrocínios
// ARQUIVO: pesquisar-patrocínios.jsx

import { useContext, useEffect, useRef, useState } from "react";
import { useNavigate } from "react-router-dom";
import { Button } from "primereact/button";
import { Card } from "primereact/card";
import { Column } from "primereact/column";
import { DataTable } from "primereact/datatable";
import { Divider } from "primereact/divider";
import { Dropdown } from "primereact/dropdown";
import { Toast } from "primereact/toast";

import ContextoUsuário from "../../contextos/contexto-usuário";
import ContextoGerenteTecnologia from "../../contextos/contexto-gerente-tecnologia";
import { serviçoBuscarPatrocínios } from "../../serviços/serviços-gerente-tecnologia";
import mostrarToast from "../../utilitários/mostrar-toast";

import {
    TAMANHOS, estilizarBotãoRetornar, estilizarBotãoTabela, estilizarCard,
    estilizarColumnHeader, estilizarColunaConsultar, estilizarDataTable, estilizarDataTablePaginator,
    estilizarDivider, estilizarFilterMenu, estilizarFlex
} from "../../utilitários/estilos";

// 1. CORRIGIDO: O nome do componente deve ser sobre Patrocínios
export default function PesquisarPatrocínios() {
    const referênciaToast = useRef(null);
    const { usuárioLogado } = useContext(ContextoUsuário);

    // 2. CORRIGIDO: Desestruturar os estados de PATROCÍNIO do contexto
    // (Assumindo que os nomes no seu ProvedorGerenteTecnologia sejam estes)
    const {
        PatrocínioConsultada, // <-- Mudou
        setPatrocínioConsultada, // <-- Mudou
        setPatrocínioSelecionada // <-- Mudou (este nome já estava em administrar-participações)
    } = useContext(ContextoGerenteTecnologia);

    const [listaPatrocínios, setListaPatrocínios] = useState([]);
    const navegar = useNavigate();

    // Este Dropdown não parece ser relevante para Patrocínio,
    // mas vou mantê-lo caso você queira filtrar por categoria de participação associada
    const opçõesCategoria = [
        { label: "Extração", value: "Extração" },
        { label: "Exploração", value: "Exploração" },
        { label: "Consultoria", value: "Consultoria" },
        { label: "Pesquisa Mineral", value: "Pesquisa Mineral" }
    ];

    function retornarCadastrarPatrocínio(){
        // 3. CORRIGIDO: Define o PATROCÍNIO selecionado ao retornar
        // (O nome "ParticipaçãoMineraçãoConsultada" não existe mais aqui)
        if (setPatrocínioSelecionada && PatrocínioConsultada) {

```

```

        setPatrocínioSelecionada(PatrocínioConsultada);
    }
    setPatrocínioConsultada(null);
    navegar("../cadastrar-participacao-mineracao");
};

// 4. CORRIGIDO: O argumento agora é 'patrocínio'
function ConsultarTemplate(patrocínio) {
    return (
        <Button icon="pi pi-search"
            className={estilizarBotãoTabela(
                usuárioLogado.cor_tema,
                // 5. CORRIGIDO: Compara com o 'patrocínioConsultado'
                PatrocínioConsultada?.id === patrocínio.id
            )}
            tooltip="Consultar Patrocínio" // <-- Mudou
            tooltipOptions={{ position: 'top' }}
            onClick={() => {
                // 6. CORRIGIDO: Define o 'patrocínioConsultado'
                setPatrocínioConsultada(patrocínio);
                // 7. CORRIGIDO: Navega para a rota de CONSULTAR PATROCÍNIO
                navegar("../consultar-patrocínio");
            }}
        />
    );
}

// ... (DropdownÁreaTemplate permanece igual) ...
function DropdownÁreaTemplate(opções) {
    function alterarFiltroDropdown(event) {
        return opções.filterCallback(event.value, opções.index);
    }

    return <Dropdown value={opções.value} options={opçõesCategoria} placeholder="Selecione"
        onChange={alterarFiltroDropdown} showClear />;
}

// Este useEffect já estava CORRETO (buscando patrocínios)
useEffect(() => {
    let desmontado = false;
    async function buscarPatrocínios() {
        try {
            const response = await serviçoBuscarPatrocínios();
            if (!desmontado && response.data) setListaPatrocínios(response.data);
        } catch (error) {
            mostrarToast(referênciaToast, error.response?.data?.erro || "Erro ao buscar patrocínios",
            "error");
        }
        buscarPatrocínios();
        return () => { desmontado = true; };
    }, []);
}

```

```

return (
  <div className={estilizarFlex()}>
    <Toast ref={referênciaToast} position="bottom-center"/>
    <Card title="Pesquisar Patrocínios" className={estilizarCard(usuarioLogado.cor_tema)}>
      <DataTable
        dataKey="id"
        size="small"
        paginator
        rows={TAMANHOS.MAX_LINHAS_TABELA}
        emptyMessage="Nenhum patrocínio encontrado." // <-- Corrigido
        value={listaPatrocínios}
        // ...
      >

      <Column
        bodyClassName={estilizarColunaConsultar()}
        body={ConsultarTemplate}
        headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
      />

      {/* As colunas de Patrocínio (já estavam corretas) */}
      <Column
        field="gerente_mineradora.usuário.nome"
        header="Gerente Responsável"
        filter showFilterOperator={false}
        headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
        sortable
      />
      <Column
        field="justificativa"
        header="Justificativa"
        filter showFilterOperator={false}
        headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
        sortable
      />
      <Column
        field="necessidade_bolsa"
        header="Necessita Bolsa"
        sortable
        headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
      />
      <Column
        field="data_manifestação"
        header="Data"
        sortable
        headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
      />
    </DataTable>
    <Divider className={estilizarDivider()}>
    <Button
      className={estilizarBotãoRetornar()}
      label="Retornar"

```

```
        onClick={retornarCadastrarPatrocínio}
      />
    </Card>
  </div>
);
}
```

```

$$$$ src.páginas.gerente-mineradora.administrar-patrocínios
import { useContext, useEffect, useRef, useState } from "react";
import { useNavigate } from "react-router-dom";
import { Button } from "primereact/button";
import { Card } from "primereact/card";
import { Column } from "primereact/column";
import { DataTable } from "primereact/datatable";
import { Divider } from "primereact/divider";
import { Dropdown } from "primereact/dropdown";
import { TriStateCheckbox } from "primereact/tristatecheckbox";

// 🔍 Contextos
import ContextoGerenteMineradora from "../../contextos/contexto-gerente-mineradora";
import ContextoUsuário from "../../contextos/contexto-usuário";

// 🔍 Serviços e utilitários
import { serviçoBuscarPatrocínioGerenteMineradora } from "../../serviços/serviços-gerente-
mineradora";
import mostrarToast from "../../utilitários/mostrar-toast";
import {
  TAMANHOS,
  estilizarBotão,
  estilizarBotãoRetornar,
  estilizarBotãoTabela,
  estilizarCard,
  estilizarColunaConsultar,
  estilizarColumnHeader,
  estilizarDataTable,
  estilizarDataTablePaginator,
  estilizarDivider,
  estilizarFilterMenu,
  estilizarFlex,
  estilizarTriStateCheckbox
} from "../../utilitários/estilos";

// 🔍 Componente principal
export default function AdministrarPatrocínios() {
  const referênciaToast = useRef(null);
  const { usuárioLogado } = useContext(ContextoUsuário);
  const { patrocínioConsultada, setPatrócinioConsultada } =
    useContext(ContextoGerenteMineradora);
  const [listaPatrocínios, setListaPatrocínios] = useState([]);
  const navegar = useNavigate();

  // 🔍 Opções de Categoria para o filtro (CORRIGIDO)
  // (Valores baseados no seu arquivo cadastrar-patrocínio.jsx)
  const opçõesCategoria = [
    { label: "Mineração Lunar", value: "MineraçãoLunar" },
    { label: "Extração de Rochas", value: "ExtraçãoDeRochas" },
    { label: "Extração de Hélio-3", value: "ExtraçãoDeHelio3" }
  ];
}

// 🔍 Retorna à página inicial

```

```

function retornarPáginaInicial() {
    navegar("/pagina-inicial");
}

// Adiciona novo patrocínio
function adicionarPatrocínio() {
    setPatrocínioConsultada(null);
    navegar("../cadastrar-patrocínio");
}

// Template para botão de consulta na tabela
function ConsultarTemplate(patrão) {
    function consultar() {
        setPatrocínioConsultada(patrão);
        navegar("../cadastrar-patrocínio");
    }

    return (
        <Button
            icon="pi pi-search"
            className={estilizarBotãoTabela(
                usuárioLogado.cor_tema,
                patrocínioConsultada?.id === patrão.id
            )}
            tooltip="Consultar Patrocínio"
            tooltipOptions={{ position: "top" }}
            onClick={consultar}
        />
    );
}

// Template para filtro Dropdown (Categoria) (CORRIGIDO)
function DropdownCategoriaTemplate(opções) {
    function alterarFiltroDropdown(event) {
        return opções.filterCallback(event.value, opções.index);
    }

    return (
        <Dropdown
            value={opções.value}
            options={opçõesCategoria} // <-- Usando a lista correta
            placeholder="Selecione"
            onChange={alterarFiltroDropdown}
            showClear
        />
    );
}

// Exibe "Sim" ou "Não" na tabela conforme booleano
function BooleanBodyTemplate(patrão) {
    return patrão.necessidade_bolsa ? "Sim" : "Não";
}

```

```

// □Template para filtro booleano (tri-state)
function BooleanFilterTemplate(opções) {
  function alterarFiltroTriState(event) {
    return opções.filterCallback(event.value);
  }

  // O label aqui estava "Concorrendo ao contrato:",
  // mudei para "Necessidade de Bolsa:" para ficar consistente
  return (
    <div>
      <label>Necessidade de Bolsa:</label>
      <TriStateCheckbox
        className={estilizarTriStateCheckbox(usuarioLogado?.cor_tema)}
        value={opções.value}
        onChange={alterarFiltroTriState}
      />
    </div>
  );
}

// □Busca os patrocínios do gerente mineradora
useEffect(() => {
  let desmontado = false;

  async function buscarPatrocíniosGerente() {
    try {
      const response = await serviçoBuscarPatrocínioGerenteMineradora(usuarioLogado.cpf);
      if (!desmontado && response.data) {
        setListaPatrocínios(response.data);
      }
    } catch (error) {
      const erro = error.response?.data?.erro;
      if (erro) mostrarToast(referênciaToast, erro, "error");
    }
  }

  buscarPatrocíniosGerente();
  return () => (desmontado = true);
}, [usuarioLogado.cpf]);

// □Renderização
return (
  <div className={estilizarFlex()}>
    <Card
      title="Administrar Patrocínios"
      className={estilizarCard(usuarioLogado.cor_tema)}
    >
      <DataTable
        dataKey="id"
        size="small"
        paginator
        rows={TAMANHOS.MAX_LINHAS_TABELA}
        emptyMessage="Nenhum patrocínio encontrado."
      </DataTable>
    </Card>
  </div>
)

```

```

value={listaPatrocínios}
responsiveLayout="scroll"
breakpoint="490px"
removableSort
className={estilizarDataTable()}
paginatorClassName={estilizarDataTablePaginator(usuarioLogado.cor_tema)}
>
  /* Botão de consulta */
<Column
  bodyClassName={estilizarColunaConsultar()}
  body={ConsultarTemplate}
  headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
/>

/* Coluna: Nome do Gerente */
<Column
  field="gerentemineradora.usuário.nome"
  header="Nome do Gerente"
  filter
  showFilterOperator={false}
  headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
  sortable
/>

/* Coluna: Justificativa */
<Column
  field="justificativa"
  header="Justificativa"
  filter
  showFilterOperator={false}
  headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
  sortable
/>

/* Coluna: Categoria (ADICIONADA) */
<Column
  field="categoria_participacao"
  header="Categoria"
  sortable
  filter
  filterMatchMode="equals"
  filterElement={DropdownCategoriaTemplate}
  showClearButton={false}
  showFilterOperator={false}
  showFilterMatchModes={false}
  filterMenuClassName={estilizarFilterMenu()}
  showFilterMenuOptions={false}
  headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
/>

/* Coluna: Necessidade de Bolsa */
<Column
  field="necessidade_bolsa"

```

```
        header="Necessidade de Bolsa"
        filter
        showFilterOperator={false}
        headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
        sortable
        filterMatchMode="equals"
        filterElement={BooleanFilterTemplate}
        body={BooleanBodyTemplate}
        showClearButton={false}
        showAddButton={false}
        filterMenuClassName={estilizarFilterMenu()}
        dataType="boolean"
    />
</DataTable>

<Divider className={estilizarDivider()} />

/* Botões de ação */
<Button
    className={estilizarBotãoRetornar()}
    label="Retornar"
    onClick={retornarPáginaInicial}
/>
<Button
    className={estilizarBotão()}
    label="Adicionar"
    onClick={adicionarPatrocínio}
/>
</Card>
</div>
);
}
```

```

$$$$ src.páginas.gerente-mineradora.cadastrar-gerente-mineradora

import { useContext, useEffect, useRef, useState } from "react";
import { useNavigate } from "react-router-dom";
import { Button } from "primereact/button";
import { Card } from "primereact/card";
import { Divider } from "primereact/divider";
import { Dropdown } from "primereact/dropdown";
import { InputNumber } from "primereact/inputnumber";
import { Toast } from "primereact/toast";
import ContextoUsuário from "../../contextos/contexto-usuário";
import { serviçoCadastrarGerenteMineradora, serviçoBuscarGerenteMineradora, serviçoAtualizarGerenteMineradora }
from "../../serviços/serviços-gerente-mineradora";
import mostrarToast from "../../utilitários/mostrar-toast";
import { MostrarMensagemErro, checarListaVazia, validarCamposObrigatórios } from "../../utilitários/validações";

import {estilizarBotão, estilizarBotãoRetornar, estilizarCard, estilizarDivCampo, estilizarDivider,
estilizarDropdown, estilizarFlex, estilizarInlineFlex, estilizarInputNumber, estilizarLabel }
from "../../utilitários/estilos";
export default function CadastrarGerenteMineradora() {
const referênciaToast = useRef(null);
const { usuárioLogado, setUsuárioLogado } = useContext(ContextoUsuário);
const [dados, setDados] = useState({ titulação: "", anos_experiência_empresarial: "" });
const [erros, setErros] = useState({});
const [cpfExistente, setCpfExistente] = useState(false);
const navegar = useNavigate();
const opçõesTitulação = [{ label: "Diretor de Operações", value: "diretor de operações" },
{ label: "Supervisor de Lavragem", value: "supervisor de lavragem" }, { label: "Coordenador de Exploração", value: "coordenador de exploração" },
{ label: "Engenheiro de Minas", value: "engenheiro de minas" }, { label: "Técnico de Minas", value: "técnico de minas" }];

function alterarEstado(event) {
const chave = event.target.name || event.value;
const valor = event.target.value;
setDados({ ...dados, [chave]: valor });
};

function validarCampos() {
let errosCamposObrigatórios;
errosCamposObrigatórios = validarCamposObrigatórios(dados);
setErros(errosCamposObrigatórios);
return checarListaVazia(errosCamposObrigatórios);
};

function títuloFormulário() {
if (usuárioLogado?.cadastrado) return "Alterar Empresa mineradora";
else return "Cadastrar Empresa mineradora";
};

async function atualizarGerenteMineradora() {
if (validarCampos()) {

```

```

try {
const response = await serviçoAtualizarGerenteMineradora({ ...dados, cpf: usuárioLogado.cpf });
if (response) mostrarToast(referênciaToast, "gerente mineradora atualizado com sucesso!",
"sucesso");
} catch (error) { mostrarToast(referênciaToast, error.response.data.erro, "erro"); }
}
};

async function cadastrarGerenteMineradora() {
if (validarCampos()) {
try {
const response = await serviçoCadastrarGerenteMineradora({ ...dados, usuário_info: usuárioLogado,
titulação: dados.titulação,
anos_experiência_empresarial: dados.anos_experiência_empresarial });
if (response.data)
setUsuárioLogado(usuário => ({ ...usuário, status: response.data.status,
token: response.data.token }));
mostrarToast(referênciaToast, "Empresa mineradora cadastrado com sucesso!", "sucesso");
} catch (error) {
setCpfExistente(true);
mostrarToast(referênciaToast, error.response.data.erro, "erro");
}
}
}
};

function labelBotãoSalvar() {
console.log("labelBotãoSalvar");
console.log(usuárioLogado);

if (usuárioLogado?.cadastrado) return "Alterar";
else return "Cadastrar";
};

function açãoBotãoSalvar() {
console.log(usuárioLogado);
if (usuárioLogado?.cadastrado) {
atualizarGerenteMineradora();
} else {
cadastrarGerenteMineradora();
}
}

function redirecionar() {
if (cpfExistente) {
setUsuárioLogado(null);
navegar("/criar-usuario");
} else {
setUsuárioLogado(usuárioLogado => ({ ...usuárioLogado, cadastrado: true }));
navegar("/pagina-inicial");
}
}

```

```
};
```

```
useEffect(() => {
let desmontado = false;
async function buscarDadosGerenteMineradora() {
try {
const response = await serviçoBuscarGerenteMineradora (usuárioLogado.cpf);
if (!desmontado && response.data) {
setDados(dados => ({ ...dados, titulação: response.data.titulação,
anos_experiência_empresarial: response.data.anos_experiência_empresarial }));
}
} catch (error) {
const erro = error.response.data.erro;
if (erro) mostrarToast(referênciaToast, erro, "erro");
}
}
```

```
if (usuárioLogado?.cadastrado) buscarDadosGerenteMineradora();
return () => desmontado = true;
}, [usuárioLogado?.cadastrado, usuárioLogado.cpf]);
return (
<div className={estilizarFlex()}>
<Toast ref={referênciaToast} onHide={redirecionar} position="bottom-center"/>
<Card title={títuloFormulário()} className={estilizarCard(usuárioLogado.cor_tema)}>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(usuárioLogado.cor_tema)}>Titulação*</label>
<Dropdown name="titulação"
className={estilizarDropdown(erros.titulação, usuárioLogado.cor_tema)}
value={dados.titulação} options={opçõesTitulação} onChange={alterarEstado}

placeholder="-- Selecione --"/>
<MostrarMensagemErro mensagem={erros.titulação}/>
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(usuárioLogado.cor_tema)}>
Anos de Experiência Empresarial*</label>
<InputNumber name="anos_experiência_empresarial" size={5}
value={dados.anos_experiência_empresarial}
onValueChange={alterarEstado} mode="decimal"
inputClassName={estilizarInputNumber(erros.anos_experiência_empresarial,
usuárioLogado.cor_tema)}/>
<MostrarMensagemErro mensagem={erros.anos_experiência_empresarial}/>
</div>
<Divider className={estilizarDivider(dados.cor_tema)}/>
<div className={estilizarInlineFlex()}>
```

```
<Button className={estilizarBotãoRetornar()} label="Retornar" onClick={redirecionar} />
<Button className={estilizarBotão()} label={labelBotãoSalvar()} onClick={açãoBotãoSalvar}/>
</div>
</Card>
</div>
);
};
```

```
$$$$ src.páginas.gerente-mineradora.cadastrar-patrocínio
import { useContext, useEffect, useRef, useState } from "react";
import { useNavigate } from "react-router-dom";
import { Button } from "primereact/button";
import { Card } from "primereact/card";
import { InputText } from "primereact/inputtext";
import { InputTextarea } from "primereact/inputtextarea";
import { Dropdown } from "primereact/dropdown";
import { Toast } from "primereact/toast";
import { Checkbox } from "primereact/checkbox";

import ContextoUsuário from "../../contextos/contexto-usuário";
import ContextoGerente from "../../contextos/contexto-gerente-mineradora"; // ou o contexto correto

import {
    serviçoCadastrarPatrocínio,
    serviçoAlterarPatrocínio,
    serviçoRemoverPatrocínio,
    serviçoBuscarÁreasAtuaçãoPatrocínio
} from "../../serviços/serviços-gerente-mineradora";

import mostrarToast from "../../utilitários/mostrar-toast";
import { MostrarMensagemErro, checarListaVazia, validarCamposObrigatórios } from
"../../utilitários/validações";

import {
    estilizarBotão,
    estilizarBotãoRemover,
    estilizarBotãoRetornar,
    estilizarCard,
    estilizarDivCampo,
    estilizarFlex,
    estilizarInlineFlex,
    estilizarInputText,
    estilizarInputTextarea,
    estilizarLabel
} from "../../utilitários/estilos";

export default function CadastrarPatrocínio() {
    const referênciaToast = useRef(null);
    const { usuárioLogado } = useContext(ContextoUsuário);
    const { patrocínioConsultada } = useContext(ContextoGerente);

    const [dados, setDados] = useState({
        necessidade_bolsa: patrocínioConsultada?.necessidade_bolsa || false, // <-- Mude para 'false'
        justificativa: patrocínioConsultada?.justificativa || "",
        categoria_participacao: patrocínioConsultada?.categoria_participacao || null
    });
    const [erros, setErros] = useState({});
    const navegar = useNavigate();

    function alterarEstado(event) {
        const chave = event.target.name;
```

```

// Se for um checkbox, pega 'checked'. Se não, pega 'value'.
const valor = event.target.type === 'checkbox' ? event.target.checked : event.target.value;
setDados({ ...dados, [chave]: valor });
}

function validarCampos() {
  const { necessidade_bolsa, justificativa, categoria_participacao } = dados;
  const errosCampos = validarCamposObrigatórios({ necessidade_bolsa, justificativa,
categoria_participacao });
  setErros(errosCampos);
  return checarListaVazia(errosCampos);
}
function mostrarParticipaçõesMineração() { navegar("../pesquisar-participacao-mineracao"); }

function retornarAdministrarPatrocínios() {
  navegar("../administrar-patrocinos");
}

async function cadastrar() {
  if (validarCampos()) {
    try {
      await serviçoCadastrarPatrocínio({ ...dados, cpf: usuárioLogado.cpf });
      mostrarToast(referênciaToast, "Patrocínio cadastrado com sucesso!", "sucesso");
    } catch (error) {
      mostrarToast(referênciaToast, error.response.data.erro, "error");
    }
  }
}

async function alterar() {
  if (validarCampos()) {
    try {
      await serviçoAlterarPatrocínio({ ...dados, id: patrocínioConsultada.id });
      mostrarToast(referênciaToast, "Patrocínio alterado com sucesso!", "sucesso");
    } catch (error) {
      mostrarToast(referênciaToast, error.response.data.erro, "error");
    }
  }
}

async function remover() {
  try {
    await serviçoRemoverPatrocínio(patrocínioConsultada.id);
    mostrarToast(referênciaToast, "Patrocínio removido com sucesso!", "sucesso");
  } catch (error) {
    mostrarToast(referênciaToast, error.response.data.erro, "error");
  }
}

function BotõesAções() {
  if (patoocínioConsultada) {
    return (
      <div className={estilizarInlineFlex()}>

```

```

        <Button className={estilizarBotãoRetornar()} label="Retornar"
onClick={retornarAdministradorPatrocínios} />
        <Button className={estilizarBotãoRemover()} label="Remover" onClick={remover} />
        <Button className={estilizarBotão()} label="Alterar" onClick={alterar} />
        <Button className={estilizarBotão()} label="Participações Mineração"
onClick={mostrarParticipaçõesMineração}/>
    </div>
);
}
} else {
return (
<div className={estilizarInlineFlex()}>
    <Button className={estilizarBotãoRetornar()} label="Retornar"
onClick={retornarAdministradorPatrocínios} />
    <Button className={estilizarBotão()} label="Cadastrar" onClick={cadastrar} />
</div>
);
}
}

function títuloFormulário() {
    return patrocínioConsultada ? "Alterar Patrocínio" : "Cadastrar Patrocínio";
}

return (
<div className={estilizarFlex()}>
<Toast ref={referênciaToast} onHide={retornarAdministradorPatrocínios} position="bottom-center" />
<Card title={títuloFormulário()} className={estilizarCard(usuarioLogado.cor_tema)}>
    /* Categoria / Tipo de participação */
    <div className={estilizarDivCampo()}>
        <label className={estilizarLabel(usuarioLogado.cor_tema)}>Categoria (Tipo):</label>
        <Dropdown
            value={dados.categoria_participacao}
            options={[
                { label: "Mineração Lunar", value: "MineraçãoLunar" },
                { label: "Extração de Rochas", value: "ExtraçãoDeRochas" },
                { label: "Extração de Hélio-3", value: "ExtraçãoDeHelio3" }
            ]}
            onChange={(e) => setDados({ ...dados, categoria_participacao: e.value })}
            placeholder="Selecione a categoria"
        />
        <MostrarMensagemErro mensagem={erros.categoria_participacao} />
    </div>
    <div className={estilizarDivCampo()}>
        <label className={estilizarLabel(usuarioLogado.cor_tema)}>Necessidade de Bolsa*</label>
        <Checkbox
            name="necessidade_bolsa"
            checked={dados.necessidade_bolsa} // Use 'checked' em vez de 'value'
            onChange={alterarEstado}
        />
        <MostrarMensagemErro mensagem={erros.necessidade_bolsa} />
    </div>

    <div className={estilizarDivCampo()}>

```

```
<label className={estilizarLabel(usuarioLogado.cor_tema)}>Justificativa*</label>
<InputTextarea
  name="justificativa"
  value={dados.justificativa}
  className={estilizarInputTextarea(erros.justificativa, usuarioLogado.cor_tema)}
  onChange={alterarEstado}
  autoResize
  cols={40}
/>
<MostrarMensagemErro mensagem={erros.justificativa} />
</div>

<BotõesAções />
</Card>
</div>
);
}
```

```

$$$$ src.páginas.gerente-mineradora.consultar-gerente-tecnologia
import { useContext } from "react";
import { useNavigate } from "react-router-dom";
import { Button } from "primereact/button";
import { Card } from "primereact/card";
import { Divider } from "primereact/divider";
import { InputMask } from "primereact/inputmask";
import { InputText } from "primereact/inputtext";
import ContextoUsuário from "../../contextos/contexto-usuário";
import ContextoGerenteMineradora from "../../contextos/contexto-gerente-mineradora";
import { ANO_MÁSCARA, TELEFONE_MÁSCARA } from "../../utilitários/máscaras";
import {TAMANHOS, estilizarBotãoRetornar, estilizarCard, estilizarDivCampo, estilizarDivider,
estilizarFlex, estilizarInlineFlex, estilizarInputMask, estilizarInputText, estilizarLabel }
from "../../utilitários/estilos";

export default function ConsultarGerenteTenologia() {
    const { usuárioLogado } = useContext(ContextoUsuário);
    // O gerente que interessa aqui é o gerente de tecnologia (vindo do contexto de gerente-
    mineradora)
    const { gerenteTecnologialInteressado } = useContext(ContextoGerenteMineradora);

    // DEBUG COMPLETO
    console.log("█ Dados brutos recebidos em ConsultarGerenteTecnologia:",
    gerenteTecnologialInteressado);
    console.log("█ gerenteTecnologialInteressado?.usuário:", gerenteTecnologialInteressado?.usuário);
    console.log("█ Tipo do objeto:", typeof gerenteTecnologialInteressado);
    const dados = {
        nome: gerenteTecnologialInteressado?.usuário?.nome,
        titulacao: gerenteTecnologialInteressado?.titulacao,
        ano_ingresso: gerenteTecnologialInteressado?.ano_ingresso,
        data_nascimento: gerenteTecnologialInteressado?.data_nascimento,
        telefone: gerenteTecnologialInteressado?.telefone,
        email: gerenteTecnologialInteressado?.usuário?.email
    };
    const navegar = useNavigate();
    function retornarConsultarInteresse() { navegar("../consultar-participação-mineração"); }

    return (
        <div className={estilizarFlex()}>
            <Card title="Consultar Gerente Tecnologia" className={estilizarCard(usuárioLogado.cor_tema)}>
                <div className={estilizarDivCampo()}>
                    <label className={estilizarLabel(usuárioLogado.cor_tema)}>Nome:</label>
                    <InputText name="nome" className={estilizarInputText(null, 400, usuárioLogado.cor_tema)}>
                        value={dados.nome} disabled />
                </div>
                <div className={estilizarDivCampo()}>
                    <label className={estilizarLabel(usuárioLogado.cor_tema)}>Titulação:</label>
                    <InputText name="titulacao" className={estilizarInputText(null, 300,
                    usuárioLogado.cor_tema)} value={dados.titulacao} disabled />
                </div>
                <div className={estilizarDivCampo()}>
                    <label className={estilizarLabel(usuárioLogado.cor_tema)}>Ano de Ingresso:</label>
                    <InputMask name="ano_ingresso" autoClear size={TAMANHOS.ANO}>

```

```

        mask={ANO_MÁSCARA} value={dados.ano_ingresso} className={estilizarInputMask(null,
usuárioLogado.cor_tema)} disabled />
    </div>
    <div className={estilizarDivCampo()}>
        <label className={estilizarLabel(usuárioLogado.cor_tema)}>Data de Nascimento:</label>
        <InputText name="data_nascimento" type="date" value={dados.data_nascimento}
className={estilizarInputText(null, usuárioLogado.cor_tema)} disabled />
    </div>
    <div className={estilizarDivCampo()}>
        <label className={estilizarLabel(usuárioLogado.cor_tema)}>Telefone:</label>
        <InputMask name="telefone" autoClear size={TAMANHOS.TELEFONE}
mask={TELEFONE_MÁSCARA} className={estilizarInputMask(null, usuárioLogado.cor_tema)}
value={dados.telefone} disabled />
    </div>
    <div className={estilizarDivCampo()}>
        <label className={estilizarLabel(usuárioLogado.cor_tema)}>Email:</label>
        <InputText name="email" className={estilizarInputText(null, 400, usuárioLogado.cor_tema)}
value={dados.email} disabled />
    </div>
    <Divider className={estilizarDivider(usuárioLogado.cor_tema)} />
    <div className={estilizarInlineFlex()}>
        <Button className={estilizarBotãoRetornar()} label="Retornar"
onClick={retornarConsultarInteresse} />
    </div>
</Card>
</div>
);
};

```

\$\$\$ src.páginas.gerente-mineradora.consultar-participação-mineração

```

import { useContext } from "react";
import { useNavigate } from "react-router-dom";
import { Button } from "primereact/button";
import { Card } from "primereact/card";
import { Divider } from "primereact/divider";
import { InputText } from "primereact/inputtext";
import { InputTextarea } from "primereact/inputtextarea";
import ContextoUsuário from "../../contextos/contexto-usuário";
import ContextoGerenteMineradora from "../../contextos/contexto-gerente-mineradora";

import { estilizarBotão, estilizarBotãoRetornar, estilizarCard,
estilizarDivCampo, estilizarDivider, estilizarFlex, estilizarInlineFlex, estilizarInputText,
estilizarLabel } from "../../utilitários/estilos";
export default function ConsultarInteresse() {
const { usuárioLogado } = useContext(ContextoUsuário);
const { participaçãoMineraçãoConsultado, setGerenteTecnológicoInteressado } =
useContext(ContextoGerenteMineradora);
// Mapar os campos corretos da participação para o objeto "dados" usado pela UI.
const dados = {
    título: participaçãoMineraçãoConsultado?.título,
    categoria: participaçãoMineraçãoConsultado?.categoria,
    área_atuação: participaçãoMineraçãoConsultado?.área_atuação,
    data_início: participaçãoMineraçãoConsultado?.data_início,
    descrição: participaçãoMineraçãoConsultado?.descrição,
    resultado: participaçãoMineraçãoConsultado?.resultado,
    nome_gerente_mineradora:
participaçãoMineraçãoConsultado?.gerente_mineradora?.usuário?.nome
};
const navegar = useNavigate();
function retornarPesquisarInteresses() { navegar("../pesquisar-participacao-mineracao"); }

// --- LOG ADICIONADO AQUI ---
function consultarGerenteTecnológicoInteressado() {

    // 1. LOG PARA VERIFICAR O QUE ESTÁ SENDO ENVIADO
    console.log(
        "□ Enviando para o contexto (setGerenteTecnológicoInteressado):",
        participaçãoMineraçãoConsultado.gerente_tecnologia
    );

    // 2. PASSE O OBJETO DO GERENTE PARA O CONTEXTO
    setGerenteTecnológicoInteressado(participaçãoMineraçãoConsultado.gerente_tecnologia);

    // 3. NAVEGUE
    navegar("./consultar-gerente-tecnologia");
};

// --- FIM DA ADIÇÃO ---

return (
<div className={estilizarFlex()}>
<Card title="Consultar Interesse" className={estilizarCard(usuárioLogado.cor_tema)}>
<div className={estilizarDivCampo()}>

```

```

<label className={estilizarLabel(usuarioLogado.cor_tema)}>Título:</label>
<InputText name="título" className={estilizarInputText(null, 400, usuárioLogado.cor_tema)}
value={dados.título} disabled />
</div>
<div className={estilizarDivCampo()}>
  <label className={estilizarLabel(usuarioLogado.cor_tema)}>Categoria:</label>
  <InputText name="categoria" className={estilizarInputText(null, 400, usuárioLogado.cor_tema)}
value={dados.categoria} disabled />
</div>
<div className={estilizarDivCampo()}>
  <label className={estilizarLabel(usuarioLogado.cor_tema)}>Área de Atuação:</label>
  <InputText name="área_atuação" className={estilizarInputText(null, 400,
usuárioLogado.cor_tema)} value={dados.área_atuação} disabled />
</div>
<div className={estilizarDivCampo()}>
  <label className={estilizarLabel(usuarioLogado.cor_tema)}>Data Início:</label>
  <InputText name="data_início" className={estilizarInputText(null, 200, usuárioLogado.cor_tema)}
value={dados.data_início} disabled />
</div>
<div className={estilizarDivCampo()}>
  <label className={estilizarLabel(usuarioLogado.cor_tema)}>Descrição:</label>
  <InputTextarea name="descrição" className={estilizarInputText(null, 400,
usuárioLogado.cor_tema)} value={dados.descrição} disabled />
</div>
<div className={estilizarDivCampo()}>
  <label className={estilizarLabel(usuarioLogado.cor_tema)}>Resultado:</label>
  <InputText name="resultado" className={estilizarInputText(null, 200, usuárioLogado.cor_tema)}
value={dados.resultado} disabled />
</div>
<div className={estilizarDivCampo()}>
  <label className={estilizarLabel(usuarioLogado.cor_tema)}>Gerente Mineradora:</label>
  <InputText name="gerente_mineradora" className={estilizarInputText(null, 400,
usuárioLogado.cor_tema)} value={dados.nome_gerente_mineradora} disabled />
</div>
<Divider className={estilizarDivider()}>
<div className={estilizarInlineFlex()}>
  <Button className={estilizarBotãoRetornar()} label="Retornar"
onClick={retornarPesquisarInteresses} />
  <Button className={estilizarBotão()} label="Gerente Tecnologia"
onClick={consultarGerenteTecnologialInteressado} />
</div>
</Card>
</div>
);
}

```

\$\$\$ src.páginas.gerente-mineradora.pesquisar-pesquisar-participações-mineração

```

import { useContext, useEffect, useRef, useState } from "react";
import { useNavigate } from "react-router-dom";

import { Button } from "primereact/button";
import { Card } from "primereact/card";
import { Column } from "primereact/column";
import { DataTable } from "primereact/datatable";
import { Divider } from "primereact/divider";
import { Dropdown } from "primereact/dropdown";
import { Toast } from "primereact/toast";
import { TriStateCheckbox } from "primereact/tristatecheckbox";

import ContextoUsuário from "../../contextos/contexto-usuário";
import ContextoGerenteMineradora from "../../contextos/contexto-gerente-mineradora";

import mostrarToast from "../../utilitários/mostrar-toast";
import { serviçoBuscarParticipaçõesMineraçãoGerenteMineradora } from "../../serviços/serviços-gerente-mineradora";

import {
    TAMANHOS,
    estilizarBotãoRetornar,
    estilizarBotãoTabela,
    estilizarCard,
    estilizarColumnHeader,
    estilizarColunaConsultar,
    estilizarDataTable,
    estilizarDataTablePaginator,
    estilizarDivider,
    estilizarFilterMenu,
    estilizarFlex,
    estilizarTriStateCheckbox
} from "../../utilitários/estilos";

export default function PesquisarParticipaçãoMineração() {

    const referênciaToast = useRef(null);
    const { usuárioLogado } = useContext(ContextoUsuário);
    const {
        participaçãoMineraçãoConsultado,
        setParticipaçãoMineraçãoConsultado,
        patrocínioConsultada
    } = useContext(ContextoGerenteMineradora);

    const [listaPatrocínios, setListaPatrocínios] = useState([]);
    const navegar = useNavigate();

    const opçõesCategoria = [
        { label: "Extração", value: "Extração" },
        { label: "Exploração", value: "Exploração" },
        { label: "Consultoria", value: "Consultoria" },
        { label: "Pesquisa Mineral", value: "Pesquisa Mineral" }
    ];

```

```

function retornarCadastrarParticipaçãoMineração() {
    setParticipaçãoMineraçãoConsultado(null);
    navegar("/pagina-inicial");
}

function ConsultarTemplate(interesse) {
    return (
        <Button
            icon="pi pi-search"
            className={estilizarBotãoTabela(
                usuárioLogado.cor_tema,
                participaçãoMineraçãoConsultado?.id === interesse.id
            )}
            tooltip="Consultar Participação Mineração"
            tooltipOptions={{ position: "top" }}
            onClick={() => {
                console.log("⚠ Consultando item:", interesse);
                setParticipaçãoMineraçãoConsultado(interesse);
                navegar("../consultar-participacao-mineracao");
            }}
        />
    );
}

function DropdownÁreaTemplate(opções) {

    function alterarFiltroDropdown(event) {
        console.log("⚠ Filtro categoria:", event.value);
        return opções.filterCallback(event.value, opções.index);
    }

    return (
        <Dropdown
            value={opções.value}
            options={opçõesCategoria}
            placeholder="Selecione"
            onChange={alterarFiltroDropdown}
            showClear
        />
    );
}

function BooleanBodyTemplate(participacao) {
    return participacao.resultado ? participacao.resultado : "N/A";
}

function BooleanFilterTemplate(opções) {

    function alterarFiltroTriState(event) {
        console.log("⚠ Filtro resultado:", event.value);
        return opções.filterCallback(event.value);
    }
}

```

```

        return (
            <div>
                <label>Resultado:</label>
                <TriStateCheckbox
                    className={estilizarTriStateCheckbox(usuarioLogado?.cor_tema)}
                    value={opções.value}
                    onChange={alterarFiltroTriState}
                />
            </div>
        );
    }

useEffect(() => {
    let desmontado = false;

    async function buscarParticipaçõesMineraçãoGerenteMineradora() {
        try {
            const response = await serviçoBuscarParticipaçõesMineraçãoGerenteMineradora(
                usuárioLogado.cpf
            );

            // DEBUG COMPLETO
            console.log("Dados brutos do backend:", response);
            console.log(" response.data:", response.data);
            console.log(" Tipo de response.data:", typeof response.data);

            if (Array.isArray(response.data)) {
                console.log(" Tamanho da lista:", response.data.length);
                console.log(" Primeiro item:", response.data[0]);
            }

            if (!desmontado && response.data) {
                setListaPatrocínios(response.data);
            }
        } catch (error) {
            mostrarToast(
                referênciaToast,
                error.response?.data?.erro || "Erro desconhecido",
                "error"
            );
        }
    }

    buscarParticipaçõesMineraçãoGerenteMineradora();
    return () => (desmontado = true);
}, [usuárioLogado.cpf]);

return (
    <div className={estilizarFlex()}>
        <Toast ref={referênciaToast} position="bottom-center" />

```

```

<Card
  title="Interesses Cadastrados"
  className={estilizarCard(usuarioLogado.cor_tema)}
>

  <DataTable
    dataKey="id"
    size="small"
    paginator
    rows={TAMANHOS.MAX_LINHAS_TABELA}
    emptyMessage="Nenhum interesse encontrado."
    value={listaPatrocínios}
    responsiveLayout="scroll"
    breakpoint="490px"
    removableSort
    className={estilizarDataTable()}
    paginatorClassName={estilizarDataTablePaginator(
      usuárioLogado.cor_tema
    )}
  >

  <Column
    bodyClassName={estilizarColunaConsultar()}
    body={ConsultarTemplate}
    headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
  />

  <Column
    field="título"
    header="Título"
    filter
    showFilterOperator={false}
    sortable
    headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
  />

  <Column
    field="gerente_mineradora.usuário.nome"
    header="Gerente Mineradora"
    filter
    showFilterOperator={false}
    sortable
    headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
  />

  <Column
    field="categoria"
    header="Categoria"
    filter
    filterMatchMode="equals"
    filterElement={DropdownÁreaTemplate}
    showClearButton={false}
  />

```

```

        showFilterOperator={false}
        showFilterMatchModes={false}
        filterMenuClassName={estilizarFilterMenu()}
        showFilterMenuOptions={false}
        sortable
        headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
    />

<Column
    field="resultado"
    header="Resultado"
    dataType="string"
    filter
    showFilterOperator={false}
    body={BooleanBodyTemplate}
    filterElement={BooleanFilterTemplate}
    filterMatchMode="equals"
    showClearButton={false}
    showAddButton={false}
    filterMenuClassName={estilizarFilterMenu()}
    sortable
    headerClassName={estilizarColumnHeader(usuarioLogado.cor_tema)}
/>

</DataTable>

<Divider className={estilizarDivider()} />

<Button
    className={estilizarBotãoRetornar()}
    label="Retornar"
    onClick={retornarCadastrarParticipaçãoMineração}
/>

        </Card>
    </div>
);
}

```

```
$$$$ src.páginas.usuário.cadastrar-usuário
import { useContext, useRef, useState } from "react";
import { Link } from "react-router-dom";
import { Button } from "primereact/button";
import { Card } from "primereact/card";
import { Dialog } from "primereact/dialog";
import { Divider } from "primereact/divider";
import { Dropdown } from "primereact/dropdown";
import { InputMask } from "primereact/inputmask";
import { InputText } from "primereact/inputtext";
import { Password } from "primereact/password";
import { Toast } from "primereact/toast";
import ContextoUsuário from "../../contextos/contexto-usuário";
import ModalConfirmaçãoUsuário from "../../componentes/modais/modal-confirmação-usuário";
import mostrarToast from "../../utilitários/mostrar-toast";
import { CPF_MÁSCARA } from "../../utilitários/máscaras";
import { MostrarMensagemErro, checarListaVazia, validarCampoEmail, validarCamposObrigatórios, validarConfirmaçãoSenha, validarConfirmaçãoSenhaOpcional, validarRecuperaçãoAcessoOpcional } from "../../utilitários/validações";
```

```
import { TAMANHOS, TEMA_PADRÃO, estilizarBotão, estilizarCard, estilizarDialog,
estilizarBotãoRemover,
estilizarDivBotõesAção, estilizarDivCampo, estilizarDivider, estilizarDropdown, estilizarFlex,
estilizarFooterDialog, estilizarInputMask, estilizarInputText, estilizarLabel, estilizarLink,
estilizarPasswordInput, estilizarPasswordTextInputBorder, estilizarSubtítulo, opçõesCores }
from "../../utilitários/estilos";
import { serviçoVerificarCpfExistente } from "../../serviços/serviços-usuário";
export default function CadastrarUsuário() {
const referênciaToast = useRef(null);
const { usuárioLogado, mostrarModalConfirmação, setMostrarModalConfirmação,
setConfirmaçãoUsuário}
= useContext(ContextoUsuário);
const [dados, setDados] = useState({ cpf: usuárioLogado?.cpf || "", nome: usuárioLogado?.nome || "", perfil: usuárioLogado?.perfil || "", email: usuárioLogado?.email || "", senha: "", confirmação: "", questão: usuárioLogado?.questão || "", resposta: "", cor_tema: usuárioLogado?.cor_tema || TEMA_PADRÃO });
const [erros, setErros] = useState({});
const opçõesPerfis = [{ label: "GerenteMineradora", value: "gerente-mineradora" },
{ label: "Gerente-Tecnologia", value: "gerente-tecnologia" }];
function alterarEstado(event) {
const chave = event.target.name;
const valor = event.target.value;
setDados({ ...dados, [chave]: valor });
};
function validarCamposAdministrar() {
const { email, senha, confirmação, questão, resposta } = dados;
```

```

let errosCamposObrigatórios = validarCamposObrigatórios({ email });
let errosValidaçãoEmail = validarCampoEmail(email);
let errosConfirmaçãoSenhaOpcional = validarConfirmaçãoSenhaOpcional(senha, confirmação);
let errosRecuperaçãoAcessoOpcional = validarRecuperaçãoAcessoOpcional(questão, resposta);
setErros({ ...errosCamposObrigatórios, ...errosConfirmaçãoSenhaOpcional,
...errosRecuperaçãoAcessoOpcional, ...errosValidaçãoEmail });
return checarListaVazia(errosCamposObrigatórios)
&& checarListaVazia(errosConfirmaçãoSenhaOpcional)
&& checarListaVazia(errosValidaçãoEmail) && checarListaVazia(errosRecuperaçãoAcessoOpcional);
};

function validarCamposCadastrar() {
const { perfil, cpf, nome, questão, resposta, senha, confirmação, email } = dados;
console.log("CadastrarUsuário.validarCamposCadastrar:dados.nome -- " + dados.nome);
console.log(JSON.parse(JSON.stringify(dados)));
if (!usuárioLogado?.perfil) {
let errosCamposObrigatórios = validarCamposObrigatórios
({ perfil, cpf, nome, questão, resposta, senha, confirmação, email });
let errosValidaçãoEmail = validarCampoEmail(email);
let errosConfirmaçãoSenha = validarConfirmaçãoSenha(senha, confirmação);
setErros({ ...errosCamposObrigatórios, ...errosConfirmaçãoSenha, ...errosValidaçãoEmail });
return checarListaVazia(errosCamposObrigatórios) && checarListaVazia(errosConfirmaçãoSenha)
&& checarListaVazia(errosValidaçãoEmail);
}
};

function validarCampos() {
if (!usuárioLogado?.perfil) return validarCamposCadastrar();
else return validarCamposAdministrar();
};

function títuloFormulário() {
if (!usuárioLogado?.perfil) return "Cadastrar Usuário";
else return "Alterar Usuário";
};

function validarConfirmarAlteração() {
const camposVálidos = validarCampos();
if (camposVálidos) confirmarOperação("alterar");
};

function textoRetorno() {
if (!usuárioLogado?.perfil) return "Retornar para login";
else return "Retornar para página inicial";
};

function linkRetorno() {
if (!usuárioLogado?.perfil) return "/";
else return "/pagina-inicial";
};

function limparOcultar() {
setConfirmaçãoUsuário(null);
setMostrarModalConfirmação(false);
};

```

```

};

async function validarConfirmarCriação() {
  const camposVálidos = validarCampos();
  if (camposVálidos) {
    let response;
    try {
      response = await serviçoVerificarCpfExistente(dados.cpf);
      if (response) confirmarOperação("salvar");
    } catch (error) {
      if (error.response.data.erro)
        mostrarToast(referênciaToast, error.response.data.erro, "erro");
    }
  }
}

function confirmarOperação(operação) {
  setConfirmaçãoUsuário({ ...dados, operação });
  setMostrarModalConfirmação(true);
};

function ComandosConfirmação() {
  if (!usuárioLogado?.perfil) {
    return <Button className={estilizarBotão(dados.cor_tema)} label="Salvar"
      onClick={validarConfirmarCriação}/>;
  } else {
    return (
      <div className={estilizarDivBotõesAção()}>
        <Button className={estilizarBotão(dados.cor_tema)} label="Alterar"
          onClick={() => validarConfirmarAlteração()}/>
        <Button className={estilizarBotãoRemover(dados.cor_tema)} label="Remover"
          onClick={() => confirmarOperação("remover")}/>
      </div>
    );
  }
};

```

```

function alinharCentro() { if (!usuárioLogado?.cadastrado) return "center"; };
return (
  <div className={estilizarFlex(alinharCentro())}>
    <Toast ref={referênciaToast} position="bottom-center" />
    <Dialog visible={mostrarModalConfirmação} className={estilizarDialog()}>
      header="Confirme seus dados" onHide={limparOcultar} closable={false}
      footer={<div className={estilizarFooterDialog()}></div>}>
      <ModalConfirmaçãoUsuário />
    </Dialog>
    <Card title={títuloFormulário()} className={estilizarCard(dados.cor_tema)}>
      <div className={estilizarDivCampo()}>
        <label className={estilizarLabel(dados.cor_tema)}>Tipo de Perfil*</label>
        <Dropdown name="perfil" className={estilizarDropdown(erros.perfil, dados.cor_tema)}
          value={dados.perfil} options={opçõesPerfis} onChange={alterarEstado}
          placeholder="-- Selecione --" disabled={usuárioLogado?.perfil} />
      </div>
    </Card>
  </div>
);

```

```

<MostrarMensagemErro mensagem={erros.perfil} />
</div>
<Divider className={estilizarDivider(dados.cor_tema)} />
<h2 className={estilizarSubtítulo(dados.cor_tema)}>Dados Pessoais</h2>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(dados.cor_tema)}>CPF*</label>
<InputMask name="cpf" autoClear className={estilizarInputMask(erros.cpf, dados.cor_tema)}
mask={CPF_MÁSCARA} size={TAMANHOS.CPF} value={dados.cpf}
onChange={alterarEstado} disabled={usuárioLogado?.perfil} />
<MostrarMensagemErro mensagem={erros.cpf} />
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(dados.cor_tema)}>Nome Completo*</label>
<InputText name="nome" className={estilizarInputText(erros.nome, 400, dados.cor_tema)}
value={dados.nome} onChange={alterarEstado} disabled={usuárioLogado?.perfil} />
<MostrarMensagemErro mensagem={erros.nome} />
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(dados.cor_tema)}>Email*</label>
<InputText name="email" className={estilizarInputText(erros.email, 400, dados.cor_tema)}
value={dados.email} onChange={alterarEstado} />

<MostrarMensagemErro mensagem={erros.email} />
</div>
<Divider className={estilizarDivider(dados.cor_tema)} />
<h2 className={estilizarSubtítulo(dados.cor_tema)}>Dados de Login</h2>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(dados.cor_tema)}>Senha e Confirmação*</label>
<Password name="senha"
inputClassName={estilizarPasswordInputBorder(erros.senha, dados.cor_tema)}
className={estilizarPasswordInput(erros.senha)} toggleMask value={dados.senha}

onChange={alterarEstado} size={TAMANHOS.SENHA}

tooltip={usuárioLogado?.token

&& "Será alterada somente se a senha e a confirmação forem informadas."} />
<Password name="confirmação" className={estilizarPasswordInput(dados.cor_tema)} toggleMask
inputClassName={estilizarPasswordInputBorder(erros.senha || erros.confirmação_senha,
dados.cor_tema)}>

size={TAMANHOS.SENHA} feedback={false} value={dados.confirmação}

onChange={alterarEstado} />
<MostrarMensagemErro mensagem={erros.senha || erros.confirmação_senha} />
</div>
<Divider className={estilizarDivider(dados.cor_tema)} />
<h2 className={estilizarSubtítulo(dados.cor_tema)}>Recuperação da conta</h2>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(dados.cor_tema)}>Questão de Segurança*</label>
<InputText name="questão"

```

```

className={estilizarInputText(erros.questão, 400, dados.cor_tema)}
placeholder="Ex: Qual era o nome do meu primeiro pet?" value={dados.questão}
onChange={alterarEstado} tooltipOptions={{ position: 'top' }}
tooltip={usuárioLogado?.token

&& "Se a resposta não for informada: a alteração de questão será ignorada."}/>
<MostrarMensagemErro mensagem={erros.questão} />
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(dados.cor_tema)}>Resposta*:</label>
<InputText name="resposta"
className={estilizarInputText(erros.resposta, 400, dados.cor_tema)}

value={dados.resposta} onChange={alterarEstado} />
<MostrarMensagemErro mensagem={erros.resposta} />
</div>
<Divider className={estilizarDivider(dados.cor_tema)} />
<h2 className={estilizarSubtítulo(dados.cor_tema)}>Configurações*: </h2>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(dados.cor_tema)}>Cor do Tema*:</label>
<Dropdown name="cor_tema" className={estilizarDropdown(erros.cor_tema, dados.cor_tema)}
value={dados.cor_tema} options={opçõesCores} onChange={alterarEstado}

placeholder="-- Selecione --" />
<MostrarMensagemErro mensagem={erros.cor_tema} />
</div>
<ComandosConfirmação/>
<div className={estilizarFlex("center")}>
<Link to={linkRetorno()} className={estilizarLink(dados.cor_tema)}>{textoRetorno()}</Link>
</div>
</Card>
</div>
);
}

```

```

$$$ src.páginas.usuário.logar-usuário
import { useContext, useRef, useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { Button } from "primereact/button";
import { Card } from "primereact/card";
import { InputMask } from "primereact/inputmask";
import { Password } from "primereact/password";
import { Toast } from "primereact/toast";
import ContextoUsuário from "../../contextos/contexto-usuário";
import { serviçoLogarUsuário } from "../../serviços/serviços-usuário";
import mostrarToast from "../../utilitários/mostrar-toast";
import { CPF_MÁSCARA } from "../../utilitários/máscaras";
import { MostrarMensagemErro, checarListaVazia, validarCamposObrigatórios } 
```

```

from "../../utilitários/validações";
import { TAMANHOS, estilizarBotão, estilizarCard, estilizarDivCampo, estilizarFlex,
estilizarInputMask, estilizarLabel, estilizarLink, estilizarLogo, estilizarPasswordInput,
estilizarPasswordTextInputBorder, estilizarPáginaÚnica } from "../../utilitários/estilos";
export default function LogarUsuário() {
const referênciaToast = useRef(null);
const { setUsuárioLogado } = useContext(ContextoUsuário);
const [dados, setDados] = useState({ nome_login: "", senha: "" });
const [erros, setErros] = useState({});
const navegar = useNavigate();
function validarCampos() {
const erros = validarCamposObrigatórios(dados);
setErros(erros);
return checarListaVazia(erros);
};
async function logarUsuário() {
if (validarCampos()) {
try {
const response = await serviçoLogarUsuário(dados);
//Alteração feita aqui TODO
// Salva o token no localStorage
if (response.data?.usuárioLogado?.token) {
localStorage.setItem('token', response.data.usuárioLogado.token);
}
setUsuárioLogado({ ...response.data?.usuárioLogado, cpf: dados.nome_login,
cadastrado: true });
navegar("/pagina-inicial");
} catch (error) {
mostrarToast(referênciaToast, error.response.data.erro, "error");
}
}
};
function alterarEstado(event) {
const chave = event.target.name || event.value;
const valor = event.target.value;
setDados({ ...dados, [chave]: valor });
};
return (
<div className={estilizarPáginaÚnica()}>
<Toast ref={referênciaToast} position="bottom-center"/>
<h1 className={estilizarLogo()}>Propostas de Trabalhos de Computação</h1>
<Card title="Login" className={estilizarCard()}>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel()}>Usuário</label>
<InputMask name="nome_login" size={TAMANHOS.CPF}
className={estilizarInputMask(erros.nome_login)}>
autoClear mask={CPF_MÁSCARA} value={dados.nome_login} onChange={alterarEstado}/>
<MostrarMensagemErro mensagem={erros.nome_login}/>
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel()}>Senha</label>
<Password name="senha" inputClassName={estilizarPasswordTextInputBorder()}>

```

```

    className={estilizarPasswordInput(erros.senha)} size={TAMANHOS.SENHA}
    value={dados.senha} feedback={false} toggleMask onChange={alterarEstado}/>
    <MostrarMensagemErro mensagem={erros.senha}/>
</div>
<div className={estilizarFlex("center")}>
    <Button className={estilizarBotão()} label="Login" onClick={logarUsuário}/>
    <Link className={estilizarLink()} to="/recuperar-acesso">Recuperar Acesso de Usuário</Link>
    <Link className={estilizarLink()} to="/criar-usuário">Cadastrar Usuário</Link>
</div>
</Card>
</div>
);
}

```

```

$$$$ src.páginas.usuário.página-inicial
import { useContext } from "react";
import { Card } from "primereact/card";
import { Image } from "primereact/image";
import ContextoUsuário from "../../contextos/contexto-usuário";
import computação from "../../imagens/imagem.jpg";
import { estilizarCard, estilizarCardHeaderCentralizado, estilizarPáginaÚnica } from "../../utilitários/estilos";
export default function PáginaInicial() {
    const { usuárioLogado } = useContext(ContextoUsuário);
    function HeaderCentralizado() {
        return (<div className={estilizarCardHeaderCentralizado()}>
            Propostas de Trabalhos de Computação</div>)
    }
}

```

```
};

return (
<div className={estilizarPáginaÚnica()}>
<Card header={HeaderCentralizado} className={estilizarCard(usuarioLogado.cor_tema)}>
<Image src={computação} alt="Venha fazer a diferença!" width={1100} />
</Card>
</div>
);
};
```

```
$$$ src.páginas.usuário.recuperar-acesso
import { useContext, useRef, useState } from "react";
import { Link } from "react-router-dom";
import { Button } from "primereact/button";
import { Card } from "primereact/card";
import { Dialog } from "primereact/dialog";
import { InputMask } from "primereact/inputmask";
import { InputText } from "primereact/inputtext";
import { Toast } from "primereact/toast";
import ContextoUsuário from "../../contextos/contexto-usuário";
import ModalRecuperarAcesso from "../../componentes/modais/modal-recuperar-acesso";
import mostrarToast from "../../utilitários/mostrar-toast";
import { CPF_MÁSCARA } from "../../utilitários/máscaras";
```

```
import { serviçoBuscarQuestãoSegurança, serviçoVerificarRespostaCorreta }  
from "../../serviços/serviços-usuário";  
import { MostrarMensagemErro, checarListaVazia, validarCamposObrigatórios, validarCpf }  
from "../../utilitários/validações";  
import { TAMANHOS, estilizarBotão, estilizarCard, estilizarDialog, estilizarDivCampo, estilizarFlex,  
estilizarFooterDialog, estilizarInputMask, estilizarInputText, estilizarLabel, estilizarLink,  
estilizarParágrafo } from "../../utilitários/estilos";  
export default function RecuperarAcesso() {  
  const referênciaToast = useRef(null);  
  const { setCpfVerificado, setNovaSenha, setTokenRecuperação } = useContext(ContextoUsuário);  
  const [dados, setDados] = useState({ cpf: "", questão: "", resposta: "", token: "" });  
  const [mostrarModal, setMostrarModal] = useState(false);  
  const [desabilitar, setDesabilitar] = useState(true);  
  const [timer, setTimer] = useState(null);  
  const [erros, setErros] = useState({});  
  function alterarEstado(event) {  
    const chave = event.target.name || event.value;  
    const valor = event.target.value;  
    setDados({ ...dados, [chave]: valor });  
  };  
  function validarCampos() {  
    let errosCamposObrigatórios = validarCamposObrigatórios({ resposta: dados.resposta });  
    setErros(errosCamposObrigatórios);  
    return checarListaVazia(errosCamposObrigatórios);  
  }  
  function esconderModal() {  
    setNovaSenha({});  
    setMostrarModal(false);  
  };  
  async function buscarQuestãoSegurança(event) {  
    const cpf = event.target.value;  
    setDados({ ...dados, cpf });  
    clearTimeout(timer);  
    const novoTimer = setTimeout(async () => {  
      try {  
        if (validarCpf(event.target.value)) {  
          const response = await serviçoBuscarQuestãoSegurança(cpf);  
          setDesabilitar(false);  
          setDados({ ...dados, cpf, questão: response.data.questão });  
        }  
      } catch (error) {  
        mostrarToast(referênciaToast, error.response.data.mensagem, "erro");  
        setDados({ ...dados, questão: "" });  
      }  
    }, 1500);  
    setTimer(novoTimer);  
  };  
  async function verificarRespostaCorreta() {  
    try {  
      const cpf = dados.cpf;  
      const response = await serviçoVerificarRespostaCorreta({ cpf, resposta: dados.resposta });  
      setCpfVerificado(cpf);  
      setTokenRecuperação(response.data.token);  
    } catch (error) {  
      mostrarToast(referênciaToast, error.response.data.mensagem, "erro");  
    }  
  };  
}
```

```

        setMostrarModal(true);
    } catch (error) { mostrarToast(referênciaToast, error.response.data.mensagem, "erro"); }
};

async function validarConfirmarRecuperaçãoAcesso() {
if (validarCampos()) { await verificarRespostaCorreta(); }
};

return (
<div className={estilizarFlex("center")}>
<Toast ref={referênciaToast} position="bottom-center" />
<Dialog visible={mostrarModal} className={estilizarDialog()}>
header="Digite sua nova senha e confirme" onHide={esconderModal}
footer={<div className={estilizarFooterDialog()}></div>}>
<ModalRecuperarAcesso />
</Dialog>
<Card title="Recuperar Acesso de Usuário" className={estilizarCard()}>
<p className={estilizarParágrafo()}>
{'Para recuperar o acesso à sua conta, forneça as informações abaixo:'}</p>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel()}>CPF*</label>
<InputMask name="cpf" className={estilizarInputMask(erros.cpf)} size={TAMANHOS.CPF}
mask={CPF_MÁSCARA} autoClear value={dados.cpf} onChange={buscarQuestãoSegurança} />
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel()}>Questão de segurança*</label>
<InputText name="questão" className={estilizarInputText(erros.questão, 400)}
value={dados.questão} disabled />
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel()}>Resposta*</label>
<InputText name="resposta" className={estilizarInputText(erros.resposta, 350)}
disabled={desabilitar} value={dados.resposta} onChange={alterarEstado} />
<MostrarMensagemErro mensagem={erros.resposta} />
</div>
<div className={estilizarFlex()}>
<Button className={estilizarBotão()} label="Confirmar" disabled={desabilitar}
onClick={validarConfirmarRecuperaçãoAcesso} />
<Link to="/" className={estilizarLink()}>Voltar ao Login</Link>
</div>
</Card>
</div>
);
}
}

```

```

$$$$ src.rotas.rotas-aplicação
import { Route, BrowserRouter, Routes } from "react-router-dom";
import RotasUsuárioLogado from "./rotas-usuário-logado";
import LogarUsuário from "../páginas/usuário/logar-usuário";
import CadastrarUsuário from "../páginas/usuário/cadastrar-usuário";
import PáginaInicial from "../páginas/usuário/página-inicial";
import CadastrarGerenteMineradora from "../páginas/gerente-mineradora/cadastrar-gerente-
mineradora";
import RecuperarAcesso from "../páginas/usuário/recuperar-acesso";
import CadastrarGerenteTecnologia from "../páginas/gerente-tecnologia/cadastrar-gerente-
tecnologia";

```

```

//=====
//□

import RotasGerenteMineradora from "./rotas-gerente-mineradora";
import { ProvedorGerenteMineradora } from "../contextos/contexto-gerente-mineradora";
import AdministrarPatrocínios from "../páginas/gerente-mineradora/administrar-patrocínios";
import CadastrarPatrocínio from "../páginas/gerente-mineradora/cadastrar-patrocínio";

import RotasGerenteTecnologia from "./rotas-gerente-tecnologia";
import { ProvedorGerenteTecnologia } from "../contextos/contexto-gerente-tecnologia";
import AdministrarParticipaçõesMineração from "../páginas/gerente-tecnologia/administrar-participações-mineração";
import CadastrarParticipaçãoMineração from "../páginas/gerente-tecnologia/cadastrar-participação-mineração";
import PesquisarPatrocínios from "../páginas/gerente-tecnologia/pesquisar-patrocínios";
import ConsultarPatrocínio from "../páginas/gerente-tecnologia/consultar-patrocínio";

import PesquisarParticipaçõesMinerações from "../páginas/gerente-mineradora/pesquisar-participações-mineração";
import ConsultarParticipaçãoMineração from "../páginas/gerente-mineradora/consultar-participação-mineração";
import ConsultarGerenteTenologia from "../páginas/gerente-mineradora/consultar-gerente-tecnologia";
import ConsultarGerenteMineradora from "../páginas/gerente-tecnologia/consultar-gerente-mineradora";

//=====
//□

export default function RotasAplicação() {
  return (
    <BrowserRouter>
      <Routes>
        <Route element={<LogarUsuário/>} path="/" />
        <Route element={<CadastrarUsuário/>} path="criar-usuario" />
        <Route element={<RecuperarAcesso/>} path="recuperar-acesso" />
        <Route element={<CadastrarGerenteTecnologia/>} path="cadastrar-gerente-tecnologia" />

        <Route element={<RotasUsuárioLogado/>} />
        <Route element={<PáginaInicial/>} path="pagina-inicial" />
        <Route element={<CadastrarUsuário/>} path="atualizar-usuario" />
        <Route element={<CadastrarGerenteMineradora/>} path="cadastrar-gerente-mineradora" />

        <Route
          element={<ProvedorGerenteMineradora><RotasGerenteMineradora/></ProvedorGerenteMineradora

```

```

    >}>
    <Route element={<AdministrarPatrocínios/>} path="administrar-patrocínios"/>
    <Route element={<CadastrarPatrocínio/>} path="cadastrar-patrocínio"/>
    <Route element={<PesquisarParticipaçõesMinerações/>} path="pesquisar-participaçao-
mineração"/>
    <Route element={<ConsultarParticipaçãoMineração/>} path="consultar-participaçao-mineração"/>
    <Route element={<ConsultarGerenteTecnologia/>} path="consultar-gerente-tecnologia"/>

</Route>

<Route
element={<ProvedorGerenteTecnologia><RotasGerenteTecnologia/></ProvedorGerenteTecnologia>}
>
    <Route element={<AdministrarParticipaçõesMineração/>} path="administrar-Participaçoes-
Mineração"/>
    <Route element={<CadastrarParticipaçãoMineração/>} path="cadastrar-participaçao-
mineração"/>
    <Route element={<PesquisarPatrocínios/>} path="pesquisar-patrocínios"/>
    <Route element={<ConsultarPatrocínio/>} path="consultar-patrocínio"/>
    <Route element={<ConsultarGerenteMineradora/>} path="consultar-gerente-mineradora"/>
</Route>

</Route>
</Routes>
</BrowserRouter>
);
};

```

```

$$$$ src.rotas.rotas-gerente-tecnologia
import { useContext } from "react";
import { Navigate, Outlet } from "react-router-dom";
import UsuárioContext from "../contextos/contexto-usuário";

export default function RotasGerenteTecnologia() {
  const { usuárioLogado } = useContext(UsuárioContext);
  if (usuárioLogado.perfil === "gerente-tecnologia") return <Outlet/>
  else return <Navigate to="/" />;
}

```

```
$$$ src.rotas.rotas-gerente-mineradora
// Arquivo: src/rotas/rotas-gerente-mineradora.js

import { useContext } from "react";
import { Navigate, Outlet } from "react-router-dom";
import UsuárioContext from "../contextos/contexto-usuário"; // Você precisa importar seu contexto de usuário aqui

export default function RotasGerenteMineradora() {
  const { usuárioLogado } = useContext(UsuárioContext); // Pega o usuário logado do contexto [cite: 922]

  // Verifica se o perfil é o correto [cite: 922]
```

```
if (usuárioLogado.perfil === "gerente-mineradora") {
    return <Outlet />; // Se for, permite o acesso à rota filha (ex: AdministrarPatrocínios) [cite: 922]
} else {
    return <Navigate to="/pagina-inicial" />; // Se não for, redireciona para a página inicial [cite: 922]
}
}
```

```
$$$ src.rotas.rotas-usuário-logado
import { useContext, useEffect } from "react";
import { Navigate, Outlet } from "react-router-dom";
import ContextoUsuário from "../contextos/contexto-usuário";
import MenuLateral from "../componentes/menu-lateral";
import servidor from "../serviços/servidor";
export default function RotasUsuárioLogado() {
const { usuárioLogado } = useContext(ContextoUsuário);
useEffect(() => {
    if (usuárioLogado?.token) {
        const interceptadorNovo = servidor.interceptors.request.use((request) => {
            request.headers.Authorization = `Bearer ${usuárioLogado.token}`;
            return request;
        })
    }
})
```

```
    });
    return () => servidor.interceptors.request.eject(interceptadorNovo);
}
}, [usuárioLogado?.token]);
if (usuárioLogado?.perfil) return <MenuLateral><Outlet/></MenuLateral>;
else return <Navigate to="/" />;
}
```

```
$$$$ src.serviços.serviços-gerente-tecnologia
import servidor from "./servidor";

export function serviçoCadastrarGerenteTecnologia(gerente) {
  console.log("[serviçoCadastrarGerenteTecnologia] Dados enviados:", gerente);
  return servidor.post("/gerente-tecnologia", gerente, {
    headers: {
      // Força log do header Authorization
      Authorization: window?.localStorage?.getItem('token') || undefined
    }
  });
};
```

```

export function serviçoAtualizarGerenteTecnologia(gerente) {
    console.log("[serviçoAtualizarGerenteTecnologia] Dados enviados:", gerente);
    return servidor.patch("/gerente-tecnologia", gerente, {
        headers: {
            Authorization: window?.localStorage?.getItem('token') || undefined
        }
    });
}

export function serviçoBuscarGerenteTecnologia(cpf) {
    console.log("[serviçoBuscarGerenteTecnologia] CPF enviado:", cpf);
    // Loga o header Authorization antes de enviar
    const token = window?.localStorage?.getItem('token');
    console.log("[serviçoBuscarGerenteTecnologia] Token Authorization:", token);
    return servidor.get(`/gerente-tecnologia/${cpf}`, {
        headers: {
            Authorization: token || undefined
        }
    });
}

// -----
// Serviços para Participação Mineração (análogo ao Interesse do Aluno)

export function serviçoCadastrarParticipaçãoMineração(participacao) {
    console.log("Enviando participação:", participacao); // debug

    return servidor.post("/gerente-tecnologia/participacao-mineracao", participacao, {
        headers: {
            Authorization: window?.localStorage?.getItem('token') || undefined
        }
    });
}

export function serviçoRemoverParticipaçãoMineração(id) {
    // Corresponde à rota: RotasGerenteTecnologia.delete("/participação-mineração/:id", ...)
    return servidor.delete(`/gerente-tecnologia/participacao-mineracao/${id}`, {
        headers: {
            Authorization: window?.localStorage?.getItem('token') || undefined
        }
    });
}

export function serviçoBuscarParticipaçõesMineraçãoGerenteTecnologia(cpf) {
    // Corresponde à rota: RotasGerenteTecnologia.get("/participação-mineração/gerente-tecnologia/:cpf", ...)
    return servidor.get(`/gerente-tecnologia/participacao-mineracao/gerente-tecnologia/${cpf}`, {
        headers: {

```

```
        Authorization: window?.localStorage?.getItem('token') || undefined
    }
});
};

export function serviçoBuscarPatrocínios() {
    // Corresponde à rota: RotasGerenteTecnologia.get("/participação-mineração/Patrocinio/", ...)
    // ATENÇÃO: A URL usa "Patrocínio" (com dois 'o') para bater com a sua rota do backend.
    return servidor.get("/gerente-tecnologia/participacao-mineracao/Patrocincio/", {
        headers: {
            Authorization: window?.localStorage?.getItem('token') || undefined
        }
    });
}
// -----
```

\$\$\$ src.serviços.serviços-gerente-mineradora

```
import servidor from "./servidor";
export function serviçoCadastrarGerenteMineradora(gerentemineradora)
{ return servidor.post("/gerente-mineradora", gerentemineradora); };
export function serviçoBuscarGerenteMineradora(cpf) { return servidor.get('/gerente-mineradora/$
{cpf}'); };
export function serviçoAtualizarGerenteMineradora(gerentemineradora)
{ return servidor.patch("/gerente-mineradora", gerentemineradora); }
```

// -----

```

export function serviçoCadastrarPatrocínio(patrocínio) {
    // Corresponde à rota: RotasGerenteMineradora.post("/patrocínio", ...)
    return servidor.post("/gerente-mineradora/patrocínio", patrocínio);
};

export function serviçoAlterarPatrocínio(patrocínio) {
    // Corresponde à rota: RotasGerenteMineradora.patch("/patrocínio", ...)
    return servidor.patch("/gerente-mineradora/patrocínio", patrocínio);
};

export function serviçoRemoverPatrocínio(id) {
    // Corresponde à rota: RotasGerenteMineradora.delete("/patrocínio/:id", ...)
    return servidor.delete('/gerente-mineradora/patrocínio/${id}');
};

export function serviçoBuscarPatrocínioGerenteMineradora(cpf) {
    // Corresponde à rota: RotasGerenteMineradora.get("/patrocínio/gerente-mineradora/:cpf", ...)
    return servidor.get('/gerente-mineradora/patrocínio/gerente-mineradora/${cpf}');
};

export function serviçoBuscarÁreasAtuaçãoPatrocínio() {
    // Corresponde à rota: RotasGerenteMineradora.get("/patrocínio/áreas-atuacao/", ...)
    return servidor.get("/gerente-mineradora/patrocínio/áreas-atuacao/");
};

export function serviçoBuscarParticipaçõesMineraçãoPatrocínio(id_patrocínio) {
    return servidor.get('/professores/participação-mineração/${id_patrocínio}'); };

export function serviçoBuscarParticipaçõesMineraçãoGerenteMineradora(cpf) {
    return servidor.get('/gerente-mineradora/participação-mineração/gerente-mineradora/${cpf}'); }

// -----

```

```

$$$$ src.serviços.serviços-usuário
import servidor from "./servidor";
export function serviçoLogarUsuário(login) { return servidor.post("/usuarios/login", login); };
export function serviçoVerificarCpfExistente(cpf) { return servidor.post
('/usuarios/verificar-cpf/${cpf}'); };

```

```

export function serviçoAlterarUsuário(usuário) { return servidor.patch("/usuarios/alterar-usuário",
usuário, { headers: { Authorization: `Bearer ${usuário.tokenRecuperação}` } }); };
export function serviçoRemoverUsuário(cpf) { return servidor.delete('/usuarios/${cpf}'); };
export function serviçoBuscarQuestãoSegurança(cpf) { return servidor.get
('/usuarios/questão/${cpf}'); };
export function serviçoVerificarRespostaCorreta(resposta) { return servidor.post

```

```
("/usuarios/verificar-resposta", resposta); };
```

```
$$$$ src.serviços.servidor
import axios from "axios";
const servidor = axios.create({ baseURL: process.env.REACT_APP_API_URL });
export default servidor;
```

```
$$$ src.utilitários.estilos
export const opçõesCores = [
{ label: "Amarelo", value: "yellow" },
{ label: "Anil", value: "indigo" },
{ label: "Azul", value: "blue" },
{ label: "Azul Piscina", value: "cyan" },
{ label: "Laranja", value: "orange" },
{ label: "Preto", value: "bluegray" },
{ label: "Rosa", value: "pink" },
{ label: "Roxo", value: "purple" },
{ label: "Verde", value: "green" },
{ label: "Verde Azulado", value: "teal" }
];
```

```
export const TAMANHOS = { ANO: 4, CPF: 13, MAX_LINHAS_TABELA: 10, SENHA: 15 ,  
TELEFONE: 12};  
export const TEMA_PADRÃO = "bluegray";  
export function estilizarBotão() {  
const cor_botão = "green";  
return `p-button-sm h-2rem text-base w-auto md:w-min mr-2  
bg-${cor_botão}-600 border-${cor_botão}-800 shadow-6`;  
  
};  
  
export function estilizarParágrafo() {  
return "text-justify text-lg md:text-sm align-self-start text-gray-900";  
};  
export function estilizarBotãoTabela(cor_tema, consultado = false) {  
let cor_botão = `text-${cor_tema}-900`;  
if (consultado) cor_botão = "p-button-danger";  
return `w-1rem p-button-sm p-button-text w-1 h-1rem ${cor_botão} shadow-6`;  
};  
export function estilizarCheckbox(erro) {  
return `w-auto mr-2 ${erro && "p-invalid"} `;  
};  
export function estilizarColunaConsultar() {  
return "w-1rem";  
};  
export function estilizarColumnHeader(cor_tema) {  
return `bg-${cor_tema}-100 text-${cor_tema}-700`;  
};  
export function estilizarDataTable() {  
return "border-1 shadow-6";  
};  
export function estilizarDataTablePaginator(cor_tema) {  
return `bg-${cor_tema}-100 text-${cor_tema}-700`;  
};  
export function estilizarFilterMenu() {  
return "flex flex-column align-items-center";  
};  
  
export function estilizarInputTextarea(erro, cor_tema) {  
let cor_borda = `border-${cor_tema}-800`;  
if (erro) cor_borda = "p-invalid";  
return `w-full lg:w-auto py-0 ${cor_borda}`;  
};  
export function estilizarTriStateCheckbox(cor_tema) {  
return `ml-2 text-${cor_tema}-800`;  
};  
  
export function estilizarBotãoRemover() {  
const cor_borda = "bluegray";  
return `p-button-sm h-2rem text-base w-auto md:w-min mr-2 p-button-danger  
border-${cor_borda}-800 shadow-6`;  
};
```

```
export function estilizarBotãoRetornar() {
  const cor_botão = "yellow";
  return `p-button-sm h-2rem text-base w-auto md:w-min mr-2
  bg-${cor_botão}-600 border-${cor_botão}-800 shadow-6`;
};

export function estilizarCard(cor_tema) {
  return `w-10 lg:w-auto overflow-auto pt-2 pb-3 m-4 text-${cor_tema}-700 border-2 shadow-8`;
};

export function estilizarCardHeaderCentralizado() {
  return "flex justify-content-center font-bold text-2xl mb-2";
};

export function estilizarColuna() {
  return "col";
};

export function estilizarDialog() {
  return "w-auto p-2";
};

export function estilizarDivBotõesAção() {
  return "align-self-center my-4";
};

export function estilizarDivCampo() {
  return "mb-3 flex flex-column sm:align-items-start md:flex-row md:align-items-center";
};

export function estilizarDivider(cor_tema = TEMA_PADRÃO) {
  return `mt-5 mb-3 border-1 border-${cor_tema}-800`;
};

export function estilizarDropdown(erro, cor_tema) {
  let cor_borda = `border-${cor_tema}-800`;
  if (erro) cor_borda = "p-invalid";
  return `w-auto ${cor_borda}`;
};

export function estilizarErro() {
  return "w-auto p-error flex-wrap text-base md:text-sm font-bold ml-2";
};

export function estilizarFlex(alinhamento = "start") {
  return `flex flex-column align-items-${alinhamento}`;
};

export function estilizarFooterDialog() {
  return "border-round mt-0";
};

export function estilizarGridColunaSidebar() {
  return "lg:col-fixed lg:w-15rem shadow-6";
};

export function estilizarGridSidebar(cor_tema) {
  return `lg:grid-nogutter lg:flex bg-${cor_tema}-100 h-screen`;
};

export function estilizarInlineFlex() {
  return "flex flex-row align-items-center mt-2";
};

export function estilizarInputMask(erro, cor_tema) {
  let cor_borda = `border-${cor_tema}-800`;
  if (erro) cor_borda = "p-invalid";
}
```

```
return `w-auto ${cor_borda}`;
};

export function estilarInputNumber(erro, cor_tema) {
let cor_borda = `border-${cor_tema}-800`;
if (erro) cor_borda = "p-invalid";
return `py-0 ${cor_borda}`;
};

export function estilarInputText(erro, input_classname, cor_tema=TEMA_PADRÃO) {
let cor_borda = `border-${cor_tema}-800`;
if (erro) cor_borda = "p-invalid";
return `input${input_classname} py-0 ${cor_borda}`;
};

export function estilarLabel(cor_tema = TEMA_PADRÃO) {
return `w-auto text-md mr-4 md:text-base text-${cor_tema}-700 font-bold`;
};

export function estilarLink(cor_tema) {
return `font-bold text-md mt-4 md:text-sm text-${cor_tema}-800`;
};

export function estilarLogo() {
return `text-center text-2xl md:text-2xl mb-6 text-${TEMA_PADRÃO}-700`;
};

export function estilarMenu() {
return "w-auto mt-2";
};

export function estilarMenuLateralDesktop(cor_tema) {
return `w-15rem p-2 flex flex-column align-items-center h-screen fixed
surface-50 bg-${cor_tema}-100 text-${cor_tema}-800`;
};

export function estilarMenuLateralMobile(cor_tema) {
return `w-full p-2 surface-50 bg-${cor_tema}-100 text-${cor_tema}-800`;
};

export function estilarModal() {
return "flex flex-column p-3";
};

export function estilarPáginaÚnica() {
return "flex flex-column align-items-center justify-content-center h-screen";
};

export function estilarPasswordInput() {
return `w-auto mr-2 mt-2 lg:mt-0`;
};

export function estilarPasswordTextInputBorder(erro, cor_tema = TEMA_PADRÃO) {
let cor_borda = `border-${cor_tema}-800`;
if (erro) cor_borda = "p-invalid";
return cor_borda;
};

export function estilarSidebar() {
return "w-15rem";
};

export function estilarSubtítulo(cor_tema) {
return `font-bold text-base align-self-start lg:mt-0 text-${cor_tema}-500`;
};

export function estilarTítulo(cor_tema) {
```

```
return `text-base align-self-start text-${cor_tema}-800 mr-3`;
}
```

```
$$$ src.utilitarios.formatar-perfil
```

```
export default function formatarPerfil(perfil) {
  switch(perfil) {
    case "gerente-mineradora": return "GerenteMineradora";
    case "gerente-tecnologia": return "Gerente-Tecnologia";
    default: return;
  }
};
```

```
$$$ src.utilitários.máscaras
const ANO_MÁSCARA = "2099";
const CPF_MÁSCARA = "999.999.999-99";
const TELEFONE_MÁSCARA = "(99) 99999-9999";
export { ANO_MÁSCARA, CPF_MÁSCARA, TELEFONE_MÁSCARA };
```

```
$$$ src.utilitários.mostrar-toast
export default function mostrarToast(referênciaToast, mensagem, tipo) {
    referênciaToast.current.show({
        severity: tipo === "sucesso" ? "success" : "error",
        summary: tipo === "sucesso" ? "Sucesso" : "Erro",
        detail: mensagem,
        life: 2000
    });
}
```

\$\$\$ src.utilitários.português

```
{  
  "startsWith": "Começa com", "contains": "Contém", "notContains": "Não contém",  
  "endsWith": "Termina com", "equals": "Igual", "notEquals": "Diferente",  
  "noFilter": "Sem filtro", "lt": "Menor que", "lte": "Menor que ou igual a", "gt": "Maior que",  
  "gte": "Maior que ou igual a", "dateIs": "Data é", "dateIsNot": "Data não é",  
  "dateBefore": "Data é anterior", "dateAfter": "Data é posterior", "custom": "Customizado",  
  "clear": "Limpar", "apply": "Aplicar", "matchAll": "Todos", "matchAny": "Qualquer",  
  "addRule": "Adicionar Regra", "removeRule": "Remover Regra", "accept": "Sim", "reject": "Não",  
  "choose": "Escolha", "upload": "Upload", "cancel": "Cancelar",  
  "dayNames": ["domingo", "segunda", "terça", "quarta", "quinta", "sexta", "sábado"],  
  "dayNamesShort": ["dom", "seg", "ter", "qua", "qui", "sex", "sáb"],
```

```

"dayNamesMin": ["Do", "Se", "Te", "Qa", "Qi", "Sx", "Sa"],
"monthNames": ["Janeiro", "Fevereiro", "Março", "Abril", "Maio", "Junho", "Julho", "Agosto",
"Setembro", "Outubro", "Novembro", "Dezembro" ],
"monthNamesShort": ["Jan", "Fev", "Mar", "Abr", "Mai", "Jun", "Jul", "Ago", "Set", "Out",
"Nov", "Dez"],
"today": "Hoje", "weekHeader": "Sem", "firstDayOfWeek": 0, "dateFormat": "dd/mm/yy",
"weak": "Fraco", "medium": "Médio", "strong": "Forte", "passwordPrompt": "Digite uma senha",
"emptyFilterMessage": "Sem opções disponíveis", "emptyMessage": "Sem resultados",
"aria": { "trueLabel": "True", "falseLabel": "False", "nullLabel": "Não selecionado",
"pageLabel": "Página", "firstPáginaeLabel": "Primeira Página",
"lastPáginaeLabel": "Última Página", "nextPáginaeLabel": "Próxima",
"previousPáginaeLabel": "Anterior"
}
}

```

```

$$$ src.utilitários.validações
import { estilizarErro } from "./estilos";
const ERRO_CAMPO_OBRIGATÓRIO = "Campo obrigatório não preenchido";
const ERRO_CONFIRMAÇÃO_SENHA = "Senha não confere";
const ERRO_FORMATO_INVÁLIDO = "Campo com formato inválido";
const ERRO_QUESTÃO = "Resposta sem questão";
export function validarCamposObrigatórios(campos) {
let errosCamposObrigatórios = {};
for (let nomeCampo in campos) {
if (campos[nomeCampo] === "" || campos[nomeCampo] === null)
errosCamposObrigatórios[nomeCampo] = ERRO_CAMPO_OBRIGATÓRIO;
}
return errosCamposObrigatórios;
}

```

```

};

export function validarConfirmaçãoSenha(senha, confirmação_senha) {
let errosConfirmaçãoSenhaOpcional = {};
if (senha !== confirmação_senha) {
errosConfirmaçãoSenhaOpcional.confirmação_senha = ERRO_CONFIRMAÇÃO_SENHA;
}
return errosConfirmaçãoSenhaOpcional;
};

export function validarCpf(cpf) {
cpf = cpf.replace(/\d/g, " ");
if (cpf.length === 11) return true;
return false;
};

export function validarConfirmaçãoSenhaOpcional(senha, confirmação_senha) {
let errosConfirmaçãoSenhaOpcional = {};
if (senha && confirmação_senha && senha !== confirmação_senha) {
errosConfirmaçãoSenhaOpcional.confirmaçãoSenha = ERRO_CONFIRMAÇÃO_SENHA;
}
return errosConfirmaçãoSenhaOpcional;
};

export function validarCampoEmail(email) {
const FORMATO_EMAIL = /^w+([.-]?\w+)*@\w+([.-]?\w+)*(\.\w{3}(\.\w{2})?)$/;
let erroEmail = {};
if (!email) erroEmail.email = ERRO_CAMPO_OBRIGATÓRIO;
else if (!FORMATO_EMAIL.test(email)) erroEmail.email = ERRO_FORMATO_INVÁLIDO;
return erroEmail;
};

export function validarRecuperaçãoAcessoOpcional(questão, resposta) {
let errosRecuperaçãoAcessoOpcional = {};
if (resposta && !questão) errosRecuperaçãoAcessoOpcional.questão = ERRO_QUESTÃO;
return errosRecuperaçãoAcessoOpcional;
};

export function checarListaVazia(listaErros) {
return Object.keys(listaErros).length === 0
};

export function MostrarMensagemErro({ mensagem }) {
if (mensagem) return <small className={estilizarErro()}>{mensagem}</small>;
else return null;
};

```

```
$$$ src.global
body {
margin: 0;
padding: 0;
font-family: var(--font-family);
background-color: var(--surface-100);
}
h1 { margin: 0; } /* Reset da margem padrão de parágrafos */
h1, label, a { color: var(--text-color); } /* Aplica cor primária de texto do tema */
.p-inputtext { height: 32px; } /* Alterar altura de InputText */
.field label { padding: 0; } /* Remover padding dos labels da classe field */
.field.grid { margin: 0; } /* Remover margin padrão do field grid */
.p-inputnumber-button { height: 16px; } /* Alterar altura dos botões + e - do inputnumber */
```

```
.p-dropdown-label { padding: 3px 10px; } /* Padding do dropdown */
/* Margem abaixo do título do card */
.p-card-title {
font-size: 1.4rem !important;
margin-bottom: 24px !important;
}
.p-card-body { padding: 0 15px !important; } /* Remove padding dos cards */
.p-card-content { padding: 0 !important; }
.p-dialog-content { padding: 5px !important; } /* Diminui padding do modal de confirmação */
/* Remove padding inferior do header dos modais e seta pt para padronização */
.p-dialog-header {
padding-top: 5px !important;
padding-bottom: 0 !important;
}
/* Alinha o texto do footer do modal de confirmação no começo e seta pb para padronização*/
.p-dialog-footer {
text-align: start !important;
padding-bottom: 5px !important;
}
.field-radiobutton { margin-bottom: 0; } /* Remove margem inferior do radio button */
.p-column-title { font-size: 14px; } /* Altera o tamanho da fonte do título da coluna nas tabelas */
td { font-size: 14px; } /* Altera o tamanho da fonte nas linhas nas tabelas */
/* Remove padding que aumenta tamanho da linha das tabelas */
.p-datable-tbody td { padding: 2px 10px !important; }
/* Deixar o ícone de filtro próximo do título da coluna */
.p-column-header-content {
display: block !important;
margin-left: 0 !important;
}
/* Diminui tamanho da linha do paginator */
.p-datable .p-paginator-bottom {
height: 50px;
padding: 0;
}
/* Diminui altura da esfera ao redor do número da página atual do paginator */
.p-paginator .p-paginator-pages .p-paginator-page.p-highlight { height: 40px; }
/* desktop */
.input400 { width: 400px; } /* nome - email */
.input300 { width: 350px; } /* questão - resposta */
.input200 { width: 200px; } /* cidade - país */
.input70 { width: 70px; } /* ano - sigla */
.input30 { width: 30px; } /* estado */
@media (max-width: 576px) {
.input400 { width: auto; } /* nome - email */
.input300 { width: auto; } /* questão - resposta */
.input200 { width: auto; } /* cidade - país */
.input70 { width: auto; } /* ano - sigla */
.input30 { width: auto; } /* estado */
img {
max-width: 100%;
height: 100%;
}
.p-sidebar .p-sidebar-header {
```

```
padding-top: 0 !important;
padding-bottom: 0 !important;
}
.p-card-title {
font-size: 1.2rem !important;
}
}
```

```
$$$ src.index
import React from 'react';
import { hydrateRoot, createRoot } from "react-dom/client";
import { locale, addLocale } from 'primereact/api';
import "primereact/resources/themes/lara-light-indigo/theme.css";
import "primereact/resources/primereact.min.css";
import "/node_modules/primeflex/primeflex.css";
import "primeicons/primeicons.css";
import "./global.css";
import português from "./utilitários/português.json"
import Rotas from "./rotas/rotas-aplicação";
import {ProvedorUsuário} from './contextos/contexto-usuário';
addLocale("pt", português)
locale("pt")
const rootElement = document.getElementById("root");
```

```
const App = (<ProvedorUsuário><Rotas/></ProvedorUsuário>);  
if (rootElement.hasChildNodes()) hydrateRoot(App, rootElement);  
else createRoot(rootElement).render(App);
```

Dourados, 18/11/25

Marcos Henrique Almeida Lima