

Desenvolvimento de Aplicações Web - Entrega 2 - Abner Lucas Pereira Cardoso Vera

```
$$$ public.index
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8"/>
    <meta name="viewport" content="width=device-width, initial-scale=1"/>
    <title>Propostas de Design de Logos</title>
  </head>
  <body><div id="root"></div></body>
</html>
```

```

$$$ 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 { estilizarBotão, estilizarBotãoRemover, estilizarDivCampo, estilizarInlineFlex,
  estilizarLabel, estilizarModal } from "../../utilitários/estilos";
import { serviçoAlterarUsuário, serviçoRemoverUsuário } from "../../serviços/serviços-
usuário";
import mostrarToast from "../../utilitários/mostrar-toast";
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 "empresário": return "Empresário";
      case "designer-gráfico": return "DesignerGráfico";
      default: return "";
    }
  };

  function fecharToast() {
    if (redirecionar) {
      setMostrarModalConfirmação(false);
      setConfirmaçãoUsuário({});
      if (confirmaçãoUsuário?.operação) setUsuárioLogado({});
      navegar("../pagina-inicial");
    }
  };
  function finalizarCadastro() {
    if (dados.perfil === "empresário") {

```

```

setUsuárioLogado({ ...dados, cadastrado: false });
setMostrarModalConfirmação(false);
navegar("../cadastrar-empresario");
} else if (dados.perfil === "designer-gráfico") {
setUsuárioLogado({ ...dados, cadastrado: false });
setMostrarModalConfirmação(false);
navegar("../cadastrar-designer-grafico");
}
};

function executarOperação() {
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);
}
};

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"); }
};

```

```

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(confirmaçãoUsuário?.cor_tema)}/>
      <Button label="Corrigir" onClick={ocultar}
        className={estilizarBotãoRemover(confirmaçãoUsuário?.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={estilizarPasswordTextInputBorder()}  
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 opçõesEmpresário = [
    { label: "Página Inicial", command: () => navegar("/pagina-inicial") },
    { label: "Menu", items: [
      { label: "Cadastrar Usuário", command: () => navegar("/atualizar-usuario"),
        disabled: usuárioLogado.status !== "ativo"},
      { label: "Cadastrar Empresário", command: () => navegar("/cadastrar-empresario")},
      { label: "Sair do Sistema", command: () => sairSistema()}
    ]},
  ];
  const opçõesDesignerGráfico = [
    { label: "Página Inicial", command: () => navegar("/pagina-inicial") },
    { label: "Menu", items: [
      { label: "Cadastrar Usuário", command: () => navegar("/atualizar-usuario"),
        disabled: usuárioLogado.status !== "ativo"},
      { label: "Cadastrar Designer Gráfico", command: () => navegar("/cadastrar-designer-grafico")},
      { label: "Sair do Sistema", command: () => sairSistema()}
    ]},
  ];

  function sairSistema() {
    setUsuárioLogado({});
    navegar("/");
  };
  function opçõesMenu() {
    switch (usuárioLogado.perfil) {
      case "empresário": return opçõesEmpresário;
      case "designer-gráfico": return opçõesDesignerGráfico;
      default: return;
    }
  }

```

```

};
function redimensionarJanela() {
  setWindowWidth(window.innerWidth);
};
function MenuServiços() {
  if (tamanhoDesktop) {
    return (
      <div className={estilizarMenuLateralDesktop(usuarioLogado?.cor_tema)}>
        <h1 className={estilizarTítulo(usuarioLogado?.cor_tema)}>{usuarioLogado?.nome}</h1>
        <h2 className={estilizarSubtítulo(usuarioLogado?.cor_tema)}>
          {formatarPerfil(usuarioLogado?.perfil)}</h2>
        <Menu className={estilizarMenu()} model={opçõesMenu()}>
        </div>
      );
    } else return (
      <>
        <div className={estilizarMenuLateralMobile(usuarioLogado?.cor_tema)}>
          <Button className={estilizarBotão(usuarioLogado?.cor_tema)} icon="pi pi-bars"
            aria-label="Filter" onClick={() => setVisible(true)}>
          <h1 className={estilizarTítulo(usuarioLogado?.cor_tema)}>{usuarioLogado?.nome}</h1>
          <h2 className={estilizarSubtítulo(usuarioLogado?.cor_tema)}>
            {formatarPerfil(usuarioLogado?.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(usuarioLogado?.cor_tema)}>
          <div className={estilizarGridColunaSidebar()}><MenuServiços/></div>
          <div className={estilizarColuna()}>{children}</div>
        </div>
      );
    }
  }
}

```



```
$$$ 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.imagem

```

$$$ src.páginas.designer-gráfico.cadastrar-designer-gráfico
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 { serviçoCadastrarDesignerGráfico, serviçoAtualizarDesignerGráfico,
serviçoBuscarDesignerGráfico }
  from "../serviços/serviços-designer-gráfico";
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 CadastrarDesignerGráfico() {
  const referênciaToast = useRef(null);
  const { usuárioLogado, setUsuárioLogado } = useContext(ContextoUsuário);
  const [dados, setDados] = useState({ disponibilidade: "", anos_experiência: "",
  data_nascimento: "",
  telefone: "" });
  const [erros, setErros] = useState({});
  const [cpfExistente, setCpfExistente] = useState(false);
  const navegar = useNavigate();
  const opçõesDisponibilidade = [{ label: "Freelancer", value: "freelancer" },
  { label: "Meio período", value: "meio período" }, { label: "Tempo integral", value: "tempo
integral"}];
  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 Designer Gráfico";
  }

```

```

else return "Cadastrar Designer Gráfico";
};
async function cadastrarDesignerGráfico() {
if (validarCampos()) {
try {
const response = await serviçoCadastrarDesignerGráfico({ ...dados, usuário_info:
usuárioLogado,
disponibilidade: dados.disponibilidade, anos_experiência: dados.anos_experiência,
data_nascimento: dados.data_nascimento, telefone: dados.telefone });
if (response.data)
setUsuárioLogado(usuário => ({ ...usuário, status: response.data.status,
token: response.data.token }));
mostrarToast(referênciaToast, "Designer Gráfico cadastrado com sucesso!", "sucesso");
} catch (error) {
setCpfExistente(true);
mostrarToast(referênciaToast, error.response.data.erro, "erro");
}
}
};
async function atualizarDesignerGráfico() {
if (validarCampos()) {
try {
const response = await serviçoAtualizarDesignerGráfico({ ...dados, cpf:
usuárioLogado.cpf });
if (response) mostrarToast(referênciaToast, "Designer Gráfico 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() {
if (usuárioLogado?.cadastrado) atualizarDesignerGráfico();
else cadastrarDesignerGráfico();
};
function redirecionar() {
if (cpfExistente) {
setUsuárioLogado(null);
navegar("/criar-usuario");
} else {
setUsuárioLogado(usuárioLogado => ({ ...usuárioLogado, cadastrado: true }));
navegar("/pagina-inicial");
}
};
useEffect(() => {
let desmontado = false;

```

```

async function buscarDadosDesignerGráfico() {
  try {
    const response = await serviçoBuscarDesignerGráfico(usuarioLogado.cpf);
    if (!desmontado && response.data) {
      setDados(dados => ({ ...dados, disponibilidade: response.data.disponibilidade,
        anos_experiência: response.data.anos_experiência,
        data_nascimento: response.data.data_nascimento, telefone: response.data.telefone }));
    }
  } catch (error) {
    const erro = error.response.data.erro;
    if (erro) mostrarToast(referênciaToast, erro, "erro");
  }
}

if (usuarioLogado?.cadastrado) buscarDadosDesignerGráfico();
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)}>Disponibilidade*:</label>
        <Dropdown name="disponibilidade" className={estilizarDropdown(erro.disponibilidade,
          usuarioLogado.cor_tema)}
          value={dados.disponibilidade} options={opçõesDisponibilidade} onChange={alterarEstado}
          placeholder="-- Selecione --"/>
        <MostrarMensagemErro mensagem={erro.disponibilidade}/>
      </div>
      <div className={estilizarDivCampo()}>
        <label className={estilizarLabel(usuarioLogado.cor_tema)}>Anos de Experiência*:</label>
        <InputMask name="anos_experiência" autoComplete="on" size={TAMANHOS.ANO}
          onChange={alterarEstado} //TODO: ?
          className={estilizarInputMask(erro.anos_experiência, usuarioLogado.cor_tema)}
          mask={ANO_MÁSCARA} value={dados.anos_experiência}/>
        <MostrarMensagemErro mensagem={erro.anos_experiência}/>
      </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(erro.data_nascimento, usuarioLogado.cor_tema)}
          onChange={alterarEstado}/>
        <MostrarMensagemErro mensagem={erro.data_nascimento}/>
      </div>
      <div className={estilizarDivCampo()}>
        <label className={estilizarLabel(usuarioLogado.cor_tema)}>Telefone*:</label>
        <InputMask name="telefone" autoComplete="on" size={TAMANHOS.TELEFONE}
          onChange={alterarEstado}
          className={estilizarInputMask(erro.telefone, usuarioLogado.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.empresa.cadastrar-empresa
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 { serviçoCadastrarEmpresa, serviçoBuscarEmpresa,
serviçoAtualizarEmpresa }
  from "../serviços/serviços-empresa";
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 CadastrarEmpresa() {
  const referênciaToast = useRef(null);
  const { usuárioLogado, setUsuárioLogado } = useContext(ContextoUsuário);
  const [dados, setDados] = useState({ classificação: "", anos_experiência_empresa: "" });
  const [erros, setErros] = useState({});
  const [cpfExistente, setCpfExistente] = useState(false);
  const navegar = useNavigate();
  const opçõesClassificação = [
    { label: "Pequeno", value: "pequeno" },
    { label: "Médio", value: "médio" },
    { label: "Grande", value: "grande" }
  ];
  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";
    else return "Cadastrar Empresa";
  };
  async function cadastrarEmpresa() {
    if (validarCampos()) {
      try {

```

```

const response = await serviçoCadastrarEmpresário({ ...dados, usuário_info:
usuárioLogado,
classificação: dados.classificação,
anos_experiência_empresarial: dados.anos_experiência_empresarial });
if (response.data)
setUsuárioLogado(usuário => ({ ...usuário, status: response.data.status,
token: response.data.token }));
mostrarToast(referênciaToast, "Empresário cadastrado com sucesso!", "sucesso");
} catch (error) {
setCpfExistente(true);
mostrarToast(referênciaToast, error.response.data.erro, "erro");
}
}
};

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

function açãoBotãoSalvar() {
if (usuárioLogado?.cadastrado) atualizarEmpresário();
else cadastrarEmpresário();
};

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

async function atualizarEmpresário() {
if (validarCampos()) {
try {
const response = await serviçoAtualizarEmpresário({ ...dados, cpf: usuárioLogado.cpf });
if (response) mostrarToast(referênciaToast, "Empresário atualizado com sucesso!",
"sucesso");
} catch (error) { mostrarToast(referênciaToast, error.response.data.erro, "erro"); }
}
};

useEffect(() => {
let desmontado = false;
async function buscarDadosEmpresário() {
try {
const response = await serviçoBuscarEmpresário(usuárioLogado.cpf);
if (!desmontado && response.data) {
setDados(dados => ({ ...dados, classificação: response.data.classificação,

```



```

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) buscarDadosEmpresário();
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="classificação"
className={estilizarDropdown(erro.classificação, usuárioLogado.cor_tema)}
value={dados.classificação} options={opçõesClassificação} onChange={alterarEstado}
placeholder="-- Selecione --"/>
<MostrarMensagemErro mensagem={erro.classificação}/>
</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(erro.anos_experiência_empresarial,
usuárioLogado.cor_tema)}/>
<MostrarMensagemErro mensagem={erro.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.usuario.cadastrar-usuario
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-usuario";
import ModalConfirmaçãoUsuário from "../../componentes/modais/modal-confirmação-
usuario";
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-usuario";
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: "Empresário", value: "empresário" },
{ label: "DesignerGráfico", value: "designer-gráfico" }];
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 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);
  };
function validarConfirmarAlteração() {
  const camposVálidos = validarCampos();
  if (camposVálidos) confirmarOperação("alterar");
};

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(erro.s.perfil, dados.cor_tema)}
value={dados.perfil} options={opçõesPerfis} onChange={alterarEstado}
placeholder="-- Selecione --" disabled={usuárioLogado?.perfil} />
<MostrarMensagemErro mensagem={erro.s.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" autoComplete="cpf" className={estilizarInputMask(erro.s.cpf,
dados.cor_tema)}
mask={CPF_MÁSCARA} size={TAMANHOS.CPF} value={dados.cpf}
onChange={alterarEstado} disabled={usuárioLogado?.perfil} />
<MostrarMensagemErro mensagem={erro.s.cpf} />
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(dados.cor_tema)}>Nome Completo*:</label>
<InputText name="nome" className={estilizarInputText(erro.s.nome, 400, dados.cor_tema)}
value={dados.nome} onChange={alterarEstado} disabled={usuárioLogado?.perfil} />
<MostrarMensagemErro mensagem={erro.s.nome} />
</div>
<div className={estilizarDivCampo()}>
<label className={estilizarLabel(dados.cor_tema)}>Email*:</label>
<InputText name="email" className={estilizarInputText(erro.s.email, 400, dados.cor_tema)}
value={dados.email} onChange={alterarEstado} />
<MostrarMensagemErro mensagem={erro.s.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={estilizarPasswordTextInputBorder(erro.s.senha, dados.cor_tema)}
className={estilizarPasswordInput(erro.s.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={estilizarPasswordTextInputBorder(erro.s.senha ||
erro.s.confirmação_senha,
dados.cor_tema)}
size={TAMANHOS.SENHA} feedback={false} value={dados.confirmação}
onChange={alterarEstado} />
<MostrarMensagemErro mensagem={erro.s.senha || erro.s.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.usuario.logar-usuario
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-usuario";
import { serviçoLogarUsuário } from "../../serviços/serviços-usuario";
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);
        setUsuárioLogado({ ...response.data?.usuarioLogado, 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 Design de Logos</h1>
      <Card title="Login" className={estilizarCard()}>

```

```
<div className={estilizarDivCampo()}>
  <label className={estilizarLabel()}>Usuário</label>
  <InputMask name="nome_login" size={TAMANHOS.CPF}
    className={estilizarInputMask(errores.nome_login)}
    autoClear mask={CPF_MÁSCARA} value={dados.nome_login} onChange={alterarEstado}/>
  <MostrarMensagemErro mensagem={errores.nome_login}/>
</div>
<div className={estilizarDivCampo()}>
  <label className={estilizarLabel()}>Senha</label>
  <Password name="senha" inputClassName={estilizarPasswordTextInputBorder()}
    className={estilizarPasswordInput(errores.senha)} size={TAMANHOS.SENHA}
    value={dados.senha} feedback={false} toggleMask onChange={alterarEstado}/>
  <MostrarMensagemErro mensagem={errores.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-usuario">Cadastrar Usuário</Link>
</div>
</Card>
</div>
);
}
```



```

$$$ src.páginas.usuario.página-inicial
import { useContext } from "react";
import { Card } from "primereact/card";
import { Image } from "primereact/image";
import ContextoUsuário from "../../contextos/contexto-usuario";
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 Design de Logos</div>)
  };
  return (
    <div className={estilizarPáginaÚnica()}>
      <Card header={HeaderCentralizado} className={estilizarCard(usuárioLogado.cor_tema)}>
        <Image src={computação} alt="Venha fazer a diferença!" width={1100} />
      </Card>
    </div>
  );
};

```

```

$$$ src.páginas.usuario.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-usuario";
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-usuario";
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);
    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(erro.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(erro.questão, 400)}
        value={dados.questão} disabled />
    </div>
  </div>
);

```

```
<div className={estilizarDivCampo()}>
  <label className={estilizarLabel()}>Resposta*:</label>
  <InputText name="resposta" className={estilizarInputText(errores.resposta, 350)}
    disabled={desabilitar} value={dados.resposta} onChange={alterarEstado} />
  <MostrarMensagemErro mensagem={errores.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 CadastrarEmpresário from "../páginas/empresário/cadastrar-empresário";
import RecuperarAcesso from "../páginas/usuário/recuperar-acesso";
import CadastrarDesignerGráfico from "../páginas/designer-gráfico/cadastrar-designer-gráfico";

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={<RotasUsuárioLogado/>} />
    <Route element={<CadastrarDesignerGráfico/>} path="cadastrar-designer-grafico" />
    <Route element={<PáginaInicial/>} path="pagina-inicial" />
    <Route element={<CadastrarUsuário/>} path="atualizar-usuario" />
    <Route element={<CadastrarEmpresário/>} path="cadastrar-empresario" />
    </Route>
    </Routes>
    </BrowserRouter>
  );
};

```

```
$$$ 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-designer-gráfico
import servidor from "../servidor";
export function serviçoCadastrarDesignerGráfico(designerGráfico) { return
servidor.post("/designers-grafico", designerGráfico); };
export function serviçoAtualizarDesignerGráfico(designerGráfico) { return
servidor.patch("/designers-grafico", designerGráfico); };
export function serviçoBuscarDesignerGráfico(cpf) { return servidor.get(`/designers-
grafico/${cpf}`); }
```

```
$$$ src.serviços.serviços-empresário
import servidor from "../servidor";
export function serviçoCadastrarEmpresário(empresário)
{ return servidor.post("/empresarios", empresário); };
export function serviçoBuscarEmpresário(cpf) { return servidor.get(`/empresarios/${cpf}`); };
export function serviçoAtualizarEmpresário(empresário)
{ return servidor.patch("/empresarios", empresário); };
```



```
$$$ 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-
usuario",
  usuário, { headers: { Authorization: `Bearer ${usuário.tokenRecuperação}` } }); };
export function serviçoRemoverUsuário(cpf) { return servidor.delete(`/usuarios/${cpf}`); };
export function serviçoBuscarQuestãoSegurança(cpf) { return servidor.get
(`/usuarios/questao/${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, 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 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 ${cor_borda}`;
};
export function estilizarInputNumber(erro, cor_tema) {
let cor_borda = `border-${cor_tema}-800`;
if (erro) cor_borda = "p-invalid";
return `py-0 ${cor_borda}`;
};
export function estilizarInputText(erro, input_classname, cor_tema=TEMA_PADRÃO) {
let cor_borda = `border-${cor_tema}-800`;
if (erro) cor_borda = "p-invalid";
return `input${input_classname} py-0 ${cor_borda}`;
};
export function estilizarLabel(cor_tema = TEMA_PADRÃO) {
return `w-auto text-md mr-4 md:text-base text-${cor_tema}-700 font-bold`;
};
export function estilizarLink(cor_tema) {
return `font-bold text-md mt-4 md:text-sm text-${cor_tema}-800`;
};

```

```

export function estilizarLogo() {
return `text-center text-2xl md:text-2xl mb-6 text-${TEMA_PADRÃO}-700`;
};
export function estilizarMenu() {
return "w-auto mt-2";
};
export function estilizarMenuLateralDesktop(cor_tema) {
return `w-15rem p-2 flex flex-column align-items-center h-screen fixed
surface-50 bg-${cor_tema}-100 text-${cor_tema}-800`;
};
export function estilizarMenuLateralMobile(cor_tema) {
return `w-full p-2 surface-50 bg-${cor_tema}-100 text-${cor_tema}-800`;
};
export function estilizarModal() {
return "flex flex-column p-3";
};
export function estilizarPáginaÚnica() {
return "flex flex-column align-items-center justify-content-center h-screen";
};
export function estilizarPasswordInput() {
return `w-auto mr-2 mt-2 lg:mt-0`;
};
export function estilizarPasswordTextInputBorder(erro, cor_tema = TEMA_PADRÃO) {
let cor_borda = `border-${cor_tema}-800`;
if (erro) cor_borda = "p-invalid";
return cor_borda;
};
export function estilizarSidebar() {
return "w-15rem";
};
export function estilizarSubtítulo(cor_tema) {
return `font-bold text-base align-self-start lg:mt-0 text-${cor_tema}-500`;
};
export function estilizarTítulo(cor_tema) {
return `text-base align-self-start text-${cor_tema}-800 mr-3`;
}
export function estilizarParágrafo() {
return "text-justify text-lg md:text-sm align-self-start text-gray-900";
};

```

```
$$$ src.utilitários.formatar-perfil
export default function formatarPerfil(perfil) {
  switch(perfil) {
    case "empresário": return "Empresário";
    case "designerGráfico": return "DesignerGráfico";
    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", "datels": "Data é", "datelsNot": "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", "firstPáginæLabel": "Primeira Página",
  "lastPáginæLabel": "Última Página", "nextPáginæLabel": "Próxima",
  "previousPáginæLabel": "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 validarConfirmaçãoSenhaOpcional(senha, confirmação_senha) {
  let errosConfirmaçãoSenhaOpcional = {};
  if (senha && confirmação_senha && senha !== confirmação_senha) {
    errosConfirmaçãoSenhaOpcional.confirmação_senha = 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;
};
export function validarCpf(cpf) {

```

```
    cpf = cpf.replace(/[^\d]/g, "");  
    if (cpf.length === 11) return true;  
    return false;  
};
```

```

$$$ 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-datatable-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 paginador */
.p-datatable .p-paginator-bottom {
height: 50px;

```

```

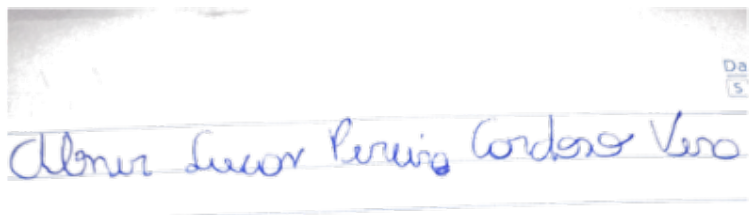
padding: 0;
}
/* Diminui altura da esfera ao redor do número da página atual do paginador */
.p-paginator .p-paginator-pages .p-paginator-page.p-highlight { height: 40px; }
/* desktop */
.input400 { width: 400px; } /* nome - email */
.input300 { width: 350px; } /* questão - resposta */
.input200 { width: 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 RotasAplicação from "./rotas/rotas-aplicação";
import { ProvedorUsuário } from './contextos/contexto-usuário';

addLocale("pt", português)
locale("pt")
const rootElement = document.getElementById("root");
const App = (<ProvedorUsuário><RotasAplicação/></ProvedorUsuário>);
if (rootElement.hasChildNodes()) hydrateRoot(App, rootElement);
else createRoot(rootElement).render(App);
```

Dourados, 21/09/2025 -- assinatura



Alenir Junior Pereira Cordens Vero