Autor: Lincon Santos Viana Barbosa
Versão: 0.0.1

Sumário

1. Criar projeto react com typescript

npx create-react-app frontend --template typescript

2. Add dependências do projeto

npm i react-router-dom@6.21.2 @types/react-router-dom@5.3.3
npm i axios@1.6.7
npm i react-bootstrap bootstrap
npm i vanilla-masker remask
npm i react-hook-form@7.49.3
npm i moment
npm i @fortawesome/fontawesome-svg-core
npm i @fortawesome/free-regular-svg-icons
npm i @fortawesome/free-brands-svg-icons
npm i @fortawesome/free-solid-svg-icons
npm i @fortawesome/react-fontawesome

3. Add comando no arquivo tsconfig.json

```
"compilerOptions": {
"baseUrl": "./src",
....
}
```

4. Extensões para VS Code

- Material Icon Theme
- IntelliCode
- ESLint
- Color Highlight

5. Instalar Dependências Projeto

```
npm install
```

6. Iniciar Projeto

```
npm start
```

7. Limpeza inicial do projeto

Dentro da pasta src, apagar os arquivos: logo.svg, index.css, App.test.tsx, setupTests.ts, reportWebVittals.ts

No arquivo index.tsx : remover as linhas

```
import './index.css';
import reportWebVitals from './reportWebVitals';
reportWebVitals();
```

No arquivo App.tsx : remover as linhas

```
import React from 'react';
import logo from './logo.svg';
todo conteúdo dentro da div
```

No arquivo index.tsx : adicionar importação do bootstrap

```
import 'bootstrap/dist/css/bootstrap.css';
```

8. Criar estrutura de pasta das Pages

8.1. Criar uma pasta Pages e dentro da pasta Pages, criar arquivo" login.tsx" e add o seguinte código

```
const Login = () => {
   return(
      <>
         <h1>Bem vindo a Login</h1>
      </>
   );
}
export default Login;
```

Dentro da pasta Pages criar os arquivos “cadastro.tsx”, “listagem.tsx” e pageNotFound.tsx e add o criar a função padrão para exibir páginas e add um texto qualquer.

9. **** Criar arquivo styles.css dentro da pasta pages

```
.fade-in-message {
   -webkit-animation: fadeinout 4s linear forwards;
   animation: fadeinout 4s linear forwards;
}
@-webkit-keyframes fadeinout {
   0%,100% { opacity: 0; }
   50% { opacity: 1; }
}

@keyframes fadeinout {
   0%,100% { opacity: 0; }
   50% { opacity: 1; }
}

.error-mensagem {
   color: red;
   background: rgba(255, 0, 0, 0.2);
   border: 1px solid red;
   padding: 10px 20px;
   margin: 2em 1em;
   height: unset;
   display: flex;
   justify-content: center;
}

#shadow-error {
   box-shadow: 2px 2px 5px rgb(255 0 0 / 40%);
}

.form-cadastro {
   height:calc(100vh - 150px);
   overflow-y:auto;
}

.table-listagem {
   height: calc(100vh - 300px);
   overflow-y: auto;
}

.container-form-senha {
   display:flex;
   justify-content:center;
   align-items:center;
   cursor:pointer;
}

.container-form-senha .eye:hover {
   color: green;
}
```

```
.container-form-star {
   display:flex;
   justify-content:center;
   align-items:flex-end;
}

.container-page-not-found {
   margin: auto;
   display: flex;
   justify-content: center;
   align-items: center;
   flex-direction: column;
   margin-top: 80px;
}

p.msg-senha{
   margin-bottom: 0;
}
```

10. ***Add componentes de navbar, sidebar e footer junto com a tela de home no arquivo App.tsx.

Criar dentro da pasta src , uma pasta components, e dentro desta pasta criar navbar, sidebar e footer e add o criar a função padrão para exibir páginas e add um texto qualquer.

11. Criar componente de rotas dentro da pasta src um arquivo routers.tsx e adicionar o seguinte conteúdo

```
import Footer from "components/footer";
import Navbars from "components/navbar";
import Cadastro from "pages/cadastro";
import Listagem from "pages/listagem";
import Login from "pages/login";
import PageNotFound from "pages/pageNotFound";
import { BrowserRouter, Route, Routes } from "react-router-dom";

const Routers = () => {
   return(
      <BrowserRouter>
         <Navbars />
         <Routes>
            <Route path="/" element={<Cadastro />}/>
            <Route path="/update/:id" element={<Cadastro />}/>
            <Route path="/login" element={<Login />}/>
            <Route path="/listagem" element={<Listagem />}/>
            <Route path="*" element={< PageNotFound />}/>
         </Routes>
         <Footer />
      </BrowserRouter>
   );
}
export default Routers;
```

12. No arquivo App.tsx adicionar o componente de Routers.

```
import Routers from 'routers';
import './App.css';

function App() {
  return (
    <div>
      <Routers />
    </div>
  );
}
export default App;
```

13. Vamos customizar os arquivos de footer, navbar e sidebar.

13.1. FOOTER

```
import { useState } from "react";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faGithub, faLinkedin, faYoutube } from "@fortawesome/free-brands-svg-icons";
import { Link } from "react-router-dom";
import './styles.css';
const Footer = () => {
   const [year] = useState(new Date().getFullYear());
   return(
      <footer className="footer mt-auto py-4 bg-primary footer-position">
         <div className="container">
            <div className="row">
               <div className="col-md-11">
                  <span className="text-light" style={{float:'right'}}>Fatec - Todos os direitos reservados - {year}</span>
               </div>
               <div className="col-md-1" style={{display:'flex', justifyContent: "space-around"}}>
                  <Link className='text-light' to={`https://www.linkedin.com/feed/`} target="_blanck"
style={{textDecoration:'none', fontSize:'25px'}}>
                     <FontAwesomeIcon icon={faLinkedin} className='fas fa-plus'></FontAwesomeIcon>
                  </Link>
                  <Link className='text-light' to={`https://github.com/linconviana`} target="_blanck"
style={{textDecoration:'none', fontSize:'25px'}}>
                     <FontAwesomeIcon icon={faGithub} className='fas fa-plus'></FontAwesomeIcon>
                  </Link>
                  <Link className='text-light' to={`https://www.youtube.com/channel/UCUJqY7s9wlgRwQx5BJS7dvA`}
style={{textDecoration:'none', fontSize:'25px'}}>
                     <FontAwesomeIcon icon={faYoutube} className='fas fa-plus'></FontAwesomeIcon>
                  </Link>
               </div>
            </div>
         </div>
      </footer>
   );
}
export default Footer;
```

13.2. Criar um arquivo styles.css dentro da pasta components

```
.footer-position {
```

```
  bottom:0;
  position:fixed;
  width:100%;
}
```

13.3. NAVBAR

```
import { faHome, faSignOut } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import Container from 'react-bootstrap/Container';
import Navbar from 'react-bootstrap/Navbar';
import NavDropdown from 'react-bootstrap/NavDropdown';
import { Link } from 'react-router-dom';
import SideBar from './sidebar';

const Navbars = () => {

  const swagger = () => {
    //const url = `${BASE_URL}/swagger-ui.html`;
    //window.open(url, '_blank');
  }

  return(
    <Navbar bg="primary" variant="dark">
      <Container>
        <Link className='text-light' to={`/`} style={{textDecoration:'none', fontSize:'25px'}}>
          <FontAwesomeIcon icon={faHome} className='fas fa-plus'></FontAwesomeIcon>
          <span style={{marginLeft:'5px'}}>Home</span>
        </Link>

        <NavDropdown title="Paginas" id="nav-dropdown" style={{color:'#fff', marginRight:'2%'}}>
          <NavDropdown.Item eventKey="4.1">
            <Link to={`/`} style={{textDecoration: 'none'}}>
              Cadastro
            </Link>
          </NavDropdown.Item>
          <NavDropdown.Item eventKey="4.2">
            <Link to={`/login`} >
              Login
            </Link>
          </NavDropdown.Item>
          <NavDropdown.Item eventKey="4.3">
            <Link to={`/listagem`} >
              Listagem
            </Link>
          </NavDropdown.Item>
          <NavDropdown.Item eventKey="4.5">
            <span onClick={swagger} style={{color: '#0d6efd'}}>
              Swagger API
            </span>
          </NavDropdown.Item>

        </NavDropdown>
```

```
          <SideBar />

        </Container>
      </Navbar>

    );
}
export default Navbars;
```

13.4. SIDEBAR

```
import { faBars, faHome, faList, faUser } from "@fortawesome/free-solid-svg-icons";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { useState } from "react";
import Offcanvas from 'react-bootstrap/Offcanvas';
import { Link } from "react-router-dom";

const SideBar = () => {

  const [show, setShow] = useState(false);
  const handleClose = () => setShow(false);
  const handleShow = () => setShow(true);

  return (
    <>
      <span onClick={handleShow}>
        <FontAwesomeIcon icon={faBars} className='fas fa-plus' style={{color:'fff',
fontSize:'25px'}}></FontAwesomeIcon>
      </span>

      <Offcanvas show={show} onHide={handleClose} placement="end" className='bg-primary'>
        <Offcanvas.Header closeButton>
          <Offcanvas.Title>Menu</Offcanvas.Title>
        </Offcanvas.Header>
        <Offcanvas.Body>
          <div style={{display:"grid"}}>
            <Link className='text-light' to={`/`} style={{textDecoration:'none', fontSize:'25px'}}>
              <FontAwesomeIcon icon={faHome} className='fas fa-plus'></FontAwesomeIcon>
              <strong style={{marginLeft:'2%'}}>Cadastro </strong>
            </Link>
            <Link className='text-light' to={`/login`} style={{textDecoration:'none', fontSize:'25px'}}>
              <FontAwesomeIcon icon={faUser} className='fas fa-plus'></FontAwesomeIcon>
              <strong style={{marginLeft:'2%'}}>Login </strong>
            </Link>
            <Link className='text-light' to={`/listagem`} style={{textDecoration:'none', fontSize:'25px'}}>
              <FontAwesomeIcon icon={faList} className='fas fa-plus'></FontAwesomeIcon>
              <strong style={{marginLeft:'2%'}}>Lista de Usuários </strong>
            </Link>
          </div>
        </Offcanvas.Body>
      </Offcanvas>
    </>
  );
}
export default SideBar;
```

14. Pasta Util, e dentro criar arquivo validar.ts, mascaras.ts, formatter.ts, api.ts, request.ts, getErrorMessage.ts

14.1. Arquivo Validar

```
/// :: Validar email
export const ValidarEmail = (email: string) => {
   var res =
   /[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-
9](?:[a-z0-9-]*[a-z0-9])?/gi;
   return res.test(email);
};
/// :: Validar email
export const ValidarForcaSenha = (senha: string) => {
   var res = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[$*&@#])[0-9a-zA-Z$*&@#]{8,}$/;
   return res.test(senha);
};
/// :: Validar CPF
export const ValidarCPF = (cpf: string) => {
   cpf = cpf.replace(/[^\d]+/g, '');
   if (cpf === '') return false;
   /// :: Elimina cpfs invalidos conhecidos
   if (
    cpf.length !== 11 ||
    cpf === '00000000000' ||
    cpf === '11111111111' ||
    cpf === '22222222222' ||
    cpf === '33333333333' ||
    cpf === '44444444444' ||
    cpf === '55555555555' ||
    cpf === '66666666666' ||
    cpf === '77777777777' ||
    cpf === '88888888888' ||
    cpf === '99999999999'
   )
    return false;
   /// :: Valida 1º digito
   let add = 0;
   for (let i = 0; i < 9; i++) add += parseInt(cpf.charAt(i)) * (10 - i);
   let rev = 11 - (add % 11);
   if (rev === 10 || rev === 11) rev = 0;
   if (rev !== parseInt(cpf.charAt(9))) return false;
   /// :: Valida 2º digito
   add = 0;
   for (let i = 0; i < 10; i++) add += parseInt(cpf.charAt(i)) * (11 - i);
   rev = 11 - (add % 11);
   if (rev === 10 || rev === 11) rev = 0;
   if (rev !== parseInt(cpf.charAt(10))) return false;
   return true;
};
/// :: Validar CNPJ
export const ValidarCNPJ = (cnpj: string) => {
   cnpj = cnpj.replace(/[^\d]+/g, '');
   if (cnpj === '') return false;
   if (cnpj.length !== 14) return false;
   /// :: Elimina CNPJs invalidos conhecidos
   if (
```

```
    cnpj === '00000000000000' ||
    cnpj === '11111111111111' ||
    cnpj === '22222222222222' ||
    cnpj === '33333333333333' ||
    cnpj === '44444444444444' ||
    cnpj === '55555555555555' ||
    cnpj === '66666666666666' ||
    cnpj === '77777777777777' ||
    cnpj === '88888888888888' ||
    cnpj === '99999999999999'
  )
    return false;
  /// :: Valida DVs
  let tamanho = cnpj.length - 2;
  let numeros = cnpj.substring(0, tamanho);
  let digitos = cnpj.substring(tamanho);
  let soma = 0;
  let pos = tamanho - 7;
  for (let i = tamanho; i >= 1; i--) {
    soma += parseInt(numeros.charAt(tamanho - i)) * pos--;
    if (pos < 2) pos = 9;
  }
  let resultado = soma % 11 < 2 ? 0 : 11 - (soma % 11);
  if (resultado !== parseInt(digitos.charAt(0))) return false;
  tamanho = tamanho + 1;
  numeros = cnpj.substring(0, tamanho);
  soma = 0;
  pos = tamanho - 7;
  for (let i = tamanho; i >= 1; i--) {
    soma += parseInt(numeros.charAt(tamanho - i)) * pos--;
    if (pos < 2) pos = 9;
  }
  resultado = soma % 11 < 2 ? 0 : 11 - (soma % 11);
  if (resultado !== parseInt(digitos.charAt(1))) return false;
  return true;
};
```

14.2. Arquivo de Mascaras

```
import {mask, unMask} from 'remask';

/// :: Criar mascaras para inputs
export const mascara = (value: string, name: string) => {
  let cleanMask = unMask(value);
  let getMask = '';
  if(name === 'contato' || name === 'recado'){
    getMask = mask(cleanMask, ["(99) 9999-9999", "(99) 99999-9999"]);
  }
  else if(name === 'nascimento'){
    getMask = mask(cleanMask, ["99/99/9999"]);
  }
  else if(name === 'documento'){
    getMask = mask(cleanMask, ["999.999.999-99", "99.999.999/9999-99"]);
  }
  return getMask;
}
```

14.3. Arquivo Formatter

```
/// :: Formatar data padrão brasileiro
export const dataLocalPtBr = (data: string) => {

   if(data)
      return (moment(data).format('DD/MM/YYYY'));
}

/// :: Formatar data padrão americano
export const dataServidor = (data: string) => {
   let newDate = '';
   if(data){
      let baseDate = data.split('/');
      newDate = baseDate[2] + '-' + baseDate[1] + '-' + baseDate[0];
   }
   return newDate;
}
```

14.4. Arquivo Request

```
export const BASE_URL = process.env.REACT_APP_BACKEND_URL ?? 'http://localhost:8080';
```

14.5. Arquivo API

```
import axios from 'axios';
import { BASE_URL } from './request';

/// :: Sair do sistema
export const Logout = () => {

  // limpar localStorage
  localStorage.clear();

  // redirecionar para login
  window.location.href = '/login';
}

export const UserLoginAuth = (email: string, senha: string, callback: any) =>{

   const url = `${BASE_URL}/login`;

   const data = JSON.stringify({
      email: email,
      senha: senha,
   });

   const headers = {
      "Content-Type": "application/json",
   };

   axios({
      method: 'post',
```

```
        url: url,
        data: data,
        headers: headers
    }).then((response) => {
        callback(response);
    }).catch((error) => {
        callback(error);
    });
}

export const PostPackage = (service: string, data: string, callback: any) => {

    /// :: Base
    const url = `${BASE_URL}/${service}`;

    const headers = {
        "Content-Type": "application/json",
    };

    axios({
        method: "post",
        url: url,
        data: data,
        headers: headers,
    }).then((response) => {
        callback(response);
    }).catch((error) => {
        callback(error);
    });
}
export const GetPackage = (id: any, service: string, callback: any) =>{

    /// :: Base.
    let url = `${BASE_URL}/${service}/${id}`;

    const headers = {
        "Content-Type": "application/json",
    };

    axios({
        method: 'get',
        url: url,
        headers: headers
    }).then((response) => {
        callback(response);
    }).catch((error) => {
        callback(error);
    });
};
export const GetAllPackage = (service: string, param: any, callback: any) =>{

    /// :: Base.
    let url = `${BASE_URL}/${service}?${param}`;

    const headers = {
        "Content-Type": "application/json",
    };
```

```
    axios({
        method: 'get',
        url: url,
        headers: headers
    }).then((response) => {
        callback(response);
    }).catch((error) => {
        callback(error);
    });
};


export const PutPackage = (id: any, service: string, data: string, callback: any) => {

/// :: Base.
    let url = `${BASE_URL}/${service}/${id}`;

    const headers = {
        "Content-Type": "application/json",
    };

    axios({
        method: "put",
        url: url,
        data: data,
        headers: headers,
    }).then((response) => {
        callback(response);
    }).catch((error) => {
        callback(error);
    });
}

export const DeletePackage = (id: any, service: string, callback: any) =>{

    let url = `${BASE_URL}/${service}/${id}`;

    const headers = {
        "Content-Type": "application/json",
    };

    axios({
        method: "delete",
        url: url,
        headers: headers,
    }).then((response) => {
        callback(response);
    }).catch((error) => {
        callback(error);
    });
}
```

14.6. Arquivo GetErrorMessage

```
import { Erros, Response } from "types/resultServidor";
```

```
/// :: Retornar mensagem de erro do backend.
export const getErrorMessage = (result: Response) => {
   let message = '';

   if(result.status === 500)
      message = 'Error Servidor (500)!';
   else if(result.status === 400)
      message = result.message;
   else if(result.status === 404)
      message = result.message;
   else if(result.status === 422){
      let resultado = '';

      if(result.error === "Validation Exception"){
         resultado += result.erros.map( (x:Erros) => x.message + ' ' )
      }
      else {
         resultado = 'Dados já cadastrados no sistema: ';
         resultado += result.erros.map( (x:Erros) => x.message + ' ' )
      }
      message = resultado;
   }
   else
      message = 'Ocorreu um erro inesperado!';

   return message;
}
```

15. Criar pasta Types, com arquivos usuário.ts, resultServidor.ts, modalTemplate.ts

15.1. Arquivo usuário.ts

```
export type Usuario= {
   id: number,
   nome: string,
   documento: string,
   nascimento: string,
   periodo: string,
   sexo: string,
   userStatus: boolean,
   recado: string,
   email: string,
   senha: string,
   validarSenha?: string,
   observacao?: string,
   contato: string,
}
export type UsuarioPage = {
   content?: Usuario[],
   totalPages: number,
   totalElements: number,
   last: boolean,
   size?: number,
   number: number,
```

```
    first: boolean,
    numberOfElements?: number,
    empty?: boolean
}
export type UserLogin= {
    email: string,
    senha: string,
}
```

15.2. Arquivo ResultServidor

```
export type Result = {
    data: any,
        response : Response,
}
export type Response = {
    data ?: Data,
        error: string,
        erros: Erros[],
        message : string ,
        status : number,
        timestamp : string,
}
export type Data = {
        error : string,
        message : string ,
        status : number,
        timestamp : string,
}
export type Erros = {
        fieldName: string,
        message: string
}
```

15.3. Arquivo ModalTemplate

```
export type ModalTemplate = {
    title: string,
    message: string,
    template: string,
    icon: JSX.Element,
    height: string,
    width: string,
    placeholder?: string,
    bgColor: string,
    callback: Function
}
export type CallbackTemplate = {
    status: boolean,
    button: string,
    inputText: string
}
```

16. Pagina Cadastro

```
import { faEye } from "@fortawesome/free-solid-svg-icons";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { useState } from "react";
import { useForm } from "react-hook-form";
import { Usuario } from "types/usuario";
import { mascara } from "utils/mascaras";
import { ValidarCNPJ, ValidarCPF, ValidarForcaSenha } from "utils/validar";
import { getErrorMessage } from "utils/getErrorMessage";
import { PostPackage} from "utils/api";
import './styles.css';

const Cadastro = () => {

   const { register, handleSubmit, formState: { errors }, setValue, getValues, reset } = useForm<Usuario>({});
   const [totalCaracters, setTotalCaracters] = useState(0);
   const [contatoImcompleto, setContatoImcompleto] = useState(false);
   const [validaSenha, setValidaSenha] = useState(false);
   const [validaForcaSenha, setValidaForcaSenha] = useState(false);
   const [cpfError, setCpfError] = useState(false);
   const [cnpjError, setCnpjError] = useState(false);
   const charLimit = 200;

   const [btnName, setBtnName] = useState("Salvar");

   const handleChange = (event: any) => {
      const name = event.target.name;
      const value = event.target.value;
      setValue(name, (mascara(value, name)));
   }

   const onBlur = (event: any) => {
      const name = event.target.name;
      const value = event.target.value;

      if(name === 'contato' || name === 'recado'){

         if(value.length === 14 || value.length === 15){
            setContatoImcompleto(false);
         }
         else {
            setContatoImcompleto(true);
            setValue(name, '');
         }
      }
      else if (name === 'senha' || name === 'validarSenha'){

         const getSenhas = getValues();
         const senha = getSenhas.senha;
         const validarSenha = getSenhas.validarSenha;

         if(senha !== null && senha !== undefined && senha !== ''
            && validarSenha !== null && validarSenha !== undefined && validarSenha !== ''
            && senha !== validarSenha){
               setValidaSenha(true);
         }
```

```
      else if(senha !== '' && validarSenha !== ''){
        setValidaSenha(false);

        if(!ValidarForcaSenha(senha)){
          setValidaForcaSenha(true);
        }
        else {
          setValidaForcaSenha(false);
        }
      }
    }
    else if(name === 'documento'){

      if(value.length === 14){
        if(ValidarCPF(value)){
          setCpfError(false);
        }
        else{
          setValue(name, '');
          setCpfError(true);
          setCnpjError(false);
        }
      }
      else if(value.length === 18){
        if(ValidarCNPJ(value)){
          setCnpjError(false);
        }
        else{
          setValue(name, '');
          setCnpjError(true);
          setCpfError(false);
        }
      }
    }

}

const showPassword = () => {
    const senha =  document.getElementById('senha') as HTMLInputElement;
    const validarSenha =  document.getElementById('validarSenha') as HTMLInputElement;

    senha.setAttribute('type', 'text');
    validarSenha.setAttribute('type', 'text');
}

const hidePassword = () => {
    const senha =  document.getElementById('senha') as HTMLInputElement;
    const validarSenha =  document.getElementById('validarSenha') as HTMLInputElement;

    senha.setAttribute('type', 'password');
    validarSenha.setAttribute('type', 'password');
}

const onSubmit = (formData: Usuario) => {

    formData.nascimento = dataServidor(formData.nascimento);
    const data = JSON.stringify(formData);
```

```
    PostPackage("usuario", data, getResult);    }

const getResult = (result: any) => {

    setTimeout(() => {
        var message = null;

        if(result.status === 201){

            message = 'Usuario cadastrado com sucesso!';
            alert(message);

            reset();
            setTotalCaracters(0);
        }
        else if(result.status === 200){

            message = 'Usuario atualizado com sucesso!';
            alert(message);
            reset();
            setTotalCaracters(0);
            setTimeout(() => {
                window.location.href = "/listagem";
            },2000);
        }
        else {
            message = getErrorMessage(result.response.data);
            alert(message);
        }
    }, 3000);
}
return(
    <div className="container form-cadastro">
        <form onSubmit={handleSubmit(onSubmit)}>
            <div className="row">
                <div className="col-md-6 offset-3 mt-3">
                    <h2>Cadastro de funcionários</h2>
                </div>
                <div className="col-md-6 offset-3 mt-2">
                    <label htmlFor="nome"><strong>Nome</strong></label>
                    <input className={`form-control my-2 ${errors.nome ? 'is-invalid' : ''}`} type="text"
                        {...register("nome", { required: 'campo nome é obrigatório' })}
                        id="nome" name="nome" placeholder="Nome Completo"
                    />
                    {errors.nome && <div className="invalid-feedback d-block">{errors.nome.message}</div>}
                </div>
                <div className="col-md-6 offset-3 mt-2">
                    <div className="row">
                        <div className="col-md-6">
                            <label htmlFor="documento"><strong>Documento</strong></label>
                            <input className={`form-control my-2 ${errors.documento ? 'is-invalid' : ''}`} type="text"
                                {...register("documento", { required: 'campo documento é obrigatório' })}
                                id="documento" name="documento" placeholder="Digite o nº CPF / CPNJ"
                                onChange={handleChange}
                                onBlur={onBlur}
                            />
```

```
                {errors.documento && <div className="invalid-feedback d-
block">{errors.documento.message}</div>}
                {cpfError && <div className="invalid-feedback d-block">Nº de CPF inválido!</div>}
                {cnpjError && <div className="invalid-feedback d-block">Nº de CNPJ inválido!</div>}
              </div>
              <div className="col-md-6">
                <label htmlFor="nascimento"><strong>Data de nascimento</strong></label>
                <input className={`form-control my-2 ${errors.documento ? 'is-invalid' : ''}`} type="text"
                  {...register("nascimento", { required: 'campo Data de nascimento é obrigatório' })}
                  id="nascimento" name="nascimento" placeholder="dd/mm/aaaa"
                  onChange={handleChange}
                />
                {errors.nascimento && <div className="invalid-feedback d-
block">{errors.nascimento.message}</div>}
              </div>
            </div>
          </div>
          <div className="col-md-6 offset-3 mt-2">
            <label htmlFor="email"><strong>E-mail</strong></label>
            <input className={`form-control my-2 ${errors.email ? 'is-invalid' : ''}`} type="text"
              {...register("email", {
                required: 'campo email é obrigatório',
                pattern: {
                  value: /^[^\s@]+@[^\s@]+\.[^\s@]+$/,
                  message: 'e-mail inválido'
                }
              })}
              id="email" name="email" placeholder="E-mail"
            />
            {errors.email && <div className="invalid-feedback d-block">{errors.email.message}</div>}
          </div>
          <div className="col-md-6 offset-3 mt-2">
            <div className="row">
              <div className="col-md-5">
                <label htmlFor="senha"><strong>Senha</strong></label>
                <input
                  className={`form-control ${errors.senha ? 'is-invalid' : ''}`} type="password"
                  {...register("senha", { required: true })}
                  id="senha" name="senha" placeholder="Digite sua senha ..."
                  onBlur={onBlur}
                />
              </div>
              <div className="col-md-5">
                <label htmlFor="validarSenha"><strong>Validar Senha</strong></label>
                <input
                  className={`form-control ${errors.validarSenha ? 'is-invalid' : ''}`} type="password"
                  {...register("validarSenha", { required: true })}
                  id="validarSenha" name="validarSenha" placeholder="Repita sua senha ..."
                  onBlur={onBlur}
                />
              </div>
              <div className="col-md-1 offset-1 container-form-senha">
                <FontAwesomeIcon icon={faEye} className="fas fa-eye eye"
                  onMouseDown={showPassword}
                  onMouseUp={hidePassword}
                ></FontAwesomeIcon>
              </div>
```

```
    </div>

    {errors.senha && <div className="invalid-feedback d-block">Senha é obrigatório</div>}
    {validaSenha && <div className="invalid-feedback d-block">As senhas não são identicas!</div>}
    {validaForcaSenha && <div className="valid-feedback d-block">
      <p className="msg-senha">deve conter ao menos um dígito!</p>
      <p className="msg-senha">deve conter ao menos uma letra minúscula!</p>
      <p className="msg-senha">deve conter ao menos uma letra maiúscula!</p>
      <p className="msg-senha">deve conter ao menos um caractere especial!</p>
      <p className="msg-senha">deve conter ao menos 8 dos caracteres mencionados!</p>
    </div>}
  </div>
  <div className="col-md-6 offset-3 mt-2">
    <div className="row">
      <div className="col-md-6">
        <label htmlFor="periodo"><strong>Periodo de Trabalho</strong></label>
        <select
          className={`form-select ${errors.periodo ? 'is-invalid' : ''}`}
          {...register("periodo", { required: true })}
          defaultValue=""
          name="periodo"
          onChange={handleChange} >
          <option value="" disabled>Escolha o periodo</option>
          <option key="1" value="Manhã">Manhã</option>
          <option key="2" value="Tarde">Tarde</option>
          <option key="3" value="Noite">Noite</option>
        </select>
        {errors.periodo && <div className="invalid-feedback d-block">Perido de trabalho obrigatorio</div>}
      </div>
      <div className="col-md-6">
        <label htmlFor=""><strong>Sexo</strong></label>
        <div style={{display:'flex', justifyContent:'space-evenly'}}>
          <label htmlFor="masculino">
            <input
              {...register("sexo", { required: true })}
              type="radio"
              value="masculino"
              id="masculino"
            />
            <span style={{marginLeft:'3px'}}>Masculino</span>
          </label>
          <label htmlFor="feminino">
            <input
              {...register("sexo", { required: true })}
              type="radio"
              value="feminino"
              id="feminino"
            />
            <span style={{marginLeft:'3px'}}>Feminino </span>
          </label>
          <label htmlFor="outros">
            <input
              {...register("sexo", { required: true })}
              type="radio"
              value="outros"
              id="outros"
            />
```

```
                    <span style={{marginLeft:'3px'}}>Outros</span>
                  </label>
                </div>
                {errors.sexo && <div className="invalid-feedback d-block">Sexo obrigatorio</div>}
              </div>
            </div>

          </div>
          <div className="col-md-6 offset-3 mt-2">
            <div className="row">
              <div className="col-md-6">
                <label htmlFor="contato"><strong>Telefone de Contato</strong></label>
                <input className={`form-control ${errors.contato ? 'is-invalid' : ''}`} type="text"
                  {...register("contato", { required: 'campo contato é obrigatório' })}
                  id="contato" name="contato" placeholder="(99) 99999-9999"
                  onChange={handleChange}
                  onBlur={onBlur}
                />
                {errors.contato && <div className="invalid-feedback d-block">Nº contato é obrigatório</div>}
                {contatoImcompleto && <div className="invalid-feedback d-block">Nº de contato
imcompleto!</div>}
              </div>
              <div className="col-md-6">
                <label htmlFor="recado"><strong>Telefone de Recado</strong></label>
                <input className={`form-control ${errors.recado ? 'is-invalid' : ''}`} type="text"
                  {...register("recado")}
                  id="recado" name="recado" placeholder="(99) 99999-9999"
                  onChange={handleChange}
                  onBlur={onBlur}
                />
                {contatoImcompleto && <div className="invalid-feedback d-block">Nº de recado
imcompleto!</div>}
              </div>
            </div>

          </div>
          <div className="col-md-6 offset-3 mt-2">
            <label htmlFor="observacao"><strong>Observação</strong></label>
            <textarea className="form-control"
              {...register("observacao")}
              id="observacao" name="observacao"
              placeholder="Digite sua observação ..."
              onChange={(e) => setTotalCaracters(e.target.value.length)}
              maxLength={200}
              rows={5}
            />
            <span>{totalCaracters}/{charLimit} caracteres</span>
          </div>
          <div className="col-md-6 offset-3 mt-2">
            <input {...register("userStatus")} type="checkbox" value="true" />
            <strong style={{marginLeft:'5px'}}>Funcionário Ativo?</strong>
          </div>
          <div className="col-md-6 offset-3 mt-2">
            <button className={'btn btn-primary'} type="submit">{btnName}</button>
          </div>
        </div>
      </form>
```

```
      </div>
   );
}
export default Cadastro;
```

17. Pagina de Listagem

```
import { useState } from "react";
import { Link } from "react-router-dom";
import { DeletePackage } from "utils/api";
import { getErrorMessage } from "utils/getErrorMessage";

let currentId = 0;

const Listagem = () => {
   const confirmDelete = (id: number) => {

      currentId = id;
      const resultado = window.confirm("Ao clicar em confirmar todas as informações sobre este usuario serão excluídas.
Tem certeza que deseja continuar?");
      if (resultado === true)
         excluirCadastro(currentId);
   }

   const excluirCadastro = (currentId: number) =>{

      if(currentId !== null && currentId !== 0){
         DeletePackage(currentId, "usuario", getResult);
      }
   }

   const getResult = (result: any) => {

      setTimeout(() => {
         let message = null;

         if(result.status === 204){

            message = 'Cadastro excluído com sucesso!';
            alert(message);
         }
         else{
            message = getErrorMessage(result.response.data);
            alert(message);
         }

      }, 3000);
   }

   return(

      <div className="container mt-5" style={{height: 'calc(100vh - 175px)', position: 'relative'}}>

         <div className="row">
            <div className="table-listagem">
            <table className="table table-striped table-bordered">
```

```
            <thead>
              <tr style={{cursor:'pointer'}}>
                <th>Nome</th>
                <th>Nº Documento</th>
                <th>Data Nascimento</th>
                <th>E-mail</th>
                <th>Periodo</th>
                <th>Sexo</th>
                <th>Nº Contato</th>
                <th>Status</th>
                <th className="text-center">Ação</th>
              </tr>
            </thead>
            <tbody>
                 <tr key={1} >
                <td>Lincon Viana</td>
                <td>123.456.789-00</td>
                <td>30/11/1982</td>
                <td>lincon@gmail.com</td>
                <td>Manhã</td>
                <td>Masculino</td>
                <td>(12) 98745-1212</td>
                <td>Ativo</td>
                <td style={{display:'flex', justifyContent:'space-around'}}>
                  <Link to={`/update/1`}><button className="btn btn-success btn-sm">Atualizar</button></Link>
                  <button className="btn btn-danger btn-sm" onClick={() => confirmDelete(1)}>Excluir</button><br/>
                </td>
              </tr>
                                                  <tr key={2} >
                <td>Heitor Viana</td>
                <td>023.456.789-99</td>
                <td>26/02/2015</td>
                <td>heitor@gmail.com</td>
                <td>tarde</td>
                <td>Masculino</td>
                <td>(12) 98745-1212</td>
                <td>Inativo</td>
                <td style={{display:'flex', justifyContent:'space-around'}}>
                  <Link to={`/update/2`}><button className="btn btn-success btn-sm">Atualizar</button></Link>
                  <button className="btn btn-danger btn-sm" onClick={() => confirmDelete(2)}>Excluir</button><br/>
                </td>
              </tr>
            </tbody>
          </table>
        </div>

      </div>

    </div>
  );
}
export default Listagem;
```

18. Atualizar pagina de listagem com dados do banco de dados

18.1. Adicionar variável baseParams

```
let baseParams = '';
```

18.2. Adicionar funções de buscar lista de usuários no banco de dados

```
const [page, setPage] = useState<UsuarioPage>({first: true, last: true, number: 0, totalElements: 0, totalPages: 0});
const [reloadTable, setReloadTable] = useState(false);

const [activePage, setActivePage] = useState(0);
const onPageChange = (index: number) => {
    setActivePage(index);
}

const [selectSize, setSelectSize] = useState(10);
const changeSize = (event: any) => {
    setSelectSize(event);
}

useEffect(() =>{

    const getResult = (result: any) => {
            setPage(result.data);
            setReloadTable(false);
    }

    baseParams = `page=${activePage}&size=${selectSize}&sort=${'id'},${'desc'}`;
    GetAllPackage("usuario", baseParams, getResult);

}, [activePage, selectSize, reloadTable]);
```

18.3. Adicionar lista de usuários na tabela de forma dinâmica

```
<tbody>
    {page.content?.map(x => (
            <tr key={x.id} >
                    <td>{x.nome}</td>
                    <td>{x.documento}</td>
                    <td>{dataLocalPtBr(x.nascimento)}</td>
                    <td>{x.email}</td>
                    <td>{x.periodo}</td>
                    <td>{x.sexo}</td>
                    <td>{x.contato}</td>
                    <td>{x.userStatus ? "Ativo" : "Inativo"}</td>
                    <td style={{display:'flex', justifyContent:'space-around'}}>
                            <Link to={`/update/${x.id}`}><button className="btn btn-success btn-
sm">Atualizar</button></Link>
                            <button className="btn btn-danger btn-sm" onClick={() =>
confirmDelete(x.id)}>Excluir</button><br/>
                    </td>
            </tr>
    ))}
</tbody>
```

18.4. Adicionar função para atualizar a tabela após excluir um registro

```
const getResult = (result: any) => {

    setTimeout(() => {
            let message = null;
            if(result.status === 204){

                    message = 'Cadastro excluído com sucesso!';
                    alert(message);
                    setReloadTable(true);
            }
            else{
                    message = getErrorMessage(result.response.data);
                    alert(message);
            }

    }, 3000);
}
```

19. Adicionar função para buscar usuário e atualizar o registro

19.1. Adicionar useParams

```
type UrlParams = {
   id: string;
}
let userPassword = '';
```

19.2. Pegar o id do usuário vindo da url e buscar usuário no banco de dados

```
const {id} = useParams<UrlParams>();
const userId = id ? parseInt(id) : null;
const [btnName, setBtnName] = useState("Salvar");

useEffect(() =>{

    const getResult = (result: any) => {

            const data = result.data as Usuario;
            setValue('id', data.id);
            setValue('nome', data.nome);
            setValue('documento', data.documento);
            const nascimento = dataLocalPtBr(data.nascimento);
            setValue('nascimento', nascimento as string);
            setValue('email', data.email);
            setValue('periodo', data.periodo);
            setValue('sexo', data.sexo);
            setValue('contato', data.contato);
            setValue('recado', data.recado);
            setValue('userStatus', data.userStatus);
            setValue('observacao', data.observacao);
                userPassword = data.senha;
```

```
            if(data.observacao !== ''){
                const totalCaracter = data.observacao?.length;
                setTotalCaracters(totalCaracter as number);
            }

            setBtnName("Atualizar");
        }

        if(userId !== null){
            GetPackage(userId, "usuario", getResult);
        }

    }, [userId, setValue]);
```

20. Adicionar função para atualizar usuário

```
const onSubmit = (formData: Usuario) => {

    formData.nascimento = dataServidor(formData.nascimento);

    if(formData.senha === null || formData.senha === '')
        formData.senha = userPassword;

    const data = JSON.stringify(formData);

    if(userId === null)
        PostPackage("usuario", data, getResult);
    else
        PutPackage(formData.id, "usuario" , data, getResult);
}
```

20.1. Bloquear campo de email ao atualizar pois o e-mail não pode ser alterado

```
<input className={`form-control my-2 ${errors.email ? 'is-invalid' : ''}`} type="text"
    {...register("email", {
        required: 'campo email é obrigatório',
        pattern: {
            value: /^[^\s@]+@[^\s@]+\.[^\s@]+$/,
            message: 'e-mail inválido'
        }
    })}
    id="email" name="email" placeholder="E-mail" disabled={userId !== null}
/>
```

21. Remover a obrigatoriedade dos campos de senha ao atualizar

```
<div className="col-md-5">
    <label htmlFor="senha">Senha</label>
    <input
        className={`form-control ${errors.senha ? 'is-invalid' : ''}`} type="password"
        {...register("senha", { required: userId === null })}
        id="senha" name="senha" placeholder="Digite sua senha ..."
        onBlur={onBlur}
    />
```

```
</div>
<div className="col-md-5">
    <label htmlFor="senha">Validar Senha</label>
    <input
            className={`form-control ${errors.validarSenha ? 'is-invalid' : ''}`} type="password"
            {...register("validarSenha", { required: userId === null })}
            id="validarSenha" name="validarSenha" placeholder="Repita sua senha ..."
            onBlur={onBlur}
    />
</div>
```

21.1. Mudar texto e cor do botão de forma dinâmica

```
<div className="col-md-6 offset-3 mt-2">
    <button className={`btn ${userId === null ? 'btn-primary' : 'btn-success'}`} type="submit">{btnName}</button>
</div>
```

22. Criar e add componente de paginação

22.1. Criar na pasta components o arquivo navigation.tsx

```
type Props = {
   page: any,
   onPageChange: Function
}

const Navigation = ({page, onPageChange}:Props) => {

   return(
      <nav style={{padding:'0'}}>
         <ul className="pagination" style={{float:'inline-end'}}>
            <li className={`page-item ${page.first ? 'disabled' : ''}`}>
               <button className="page-link" onClick={()=>onPageChange(page.number - 1)}>Anterior</button>
            </li>
            <li className="page-item active" aria-current="page">
               <span className="page-link">{page.number + 1}</span>
            </li>
            <li className={`page-item ${page.last ? 'disabled' : ''}`}>
               <button className="page-link" onClick={()=>onPageChange(page.number + 1)}>Próximo</button>
            </li>
         </ul>
      </nav>
   );
}

export default Navigation;
```

22.2. Adicionar componente navigation na pagina de listagem de usuários

```
<div style={{bottom: '0', position: 'absolute', padding: '0'}}>
    <Navigation page={page} onPageChange={onPageChange}/>
```

```
</div>
```

23. Criar e add componente de search

23.1. Criar na pasta components o arquivo search.tsx

```
type Props = {
   changeSize: Function
   value: number
}

const Search = ({changeSize, value}:Props) => {

   return(
      <>
         <div className="col-md-3">
            <label style={{display:'flex', alignItems:'center', justifyContent:'space-between'}}>

               <select style={{width:'40%'}} className="form-select" value={value} onChange={(e) =>
changeSize(e.target.value)}>
                  <option key="2" value="2">2</option>
                  <option key="4" value="4">4</option>
                  <option key="6" value="6">6</option>
                  <option key="10" value="10">10</option>
                  <option key="25" value="25">25</option>
                  <option key="50" value="50">50</option>
                  <option key="100" value="100">100</option>
               </select>
               <span>Resultados por pagina</span>
            </label>
         </div>
         <div className="col-md-4"></div>
      </>
   );
}

export default Search;
```

23.2. Adicionar componente search na página de listagem de usuários

```
<div className="row mt-3 mb-3">
     <Search changeSize={changeSize} value={selectSize}/>
</div>
```

24. Criar e add componente de progressbar

24.1. Criar na pasta components o arquivo progressbar.tsx

```
import { forwardRef, useEffect, useImperativeHandle, useState } from "react";
import { ProgressBar } from "react-bootstrap";

export interface HandleProgressBar {
```

```
    openProgressBar: () => void;
    closeProgressBar: () => void;
}

let index = 0;

const ProgressBars: React.ForwardRefRenderFunction<HandleProgressBar> = (props, ref) => {

    const [percentBar, setPercentBar] = useState(0);
    const [visibleBar, setVisibleBar] = useState(false);
    const [flag, setFlag] = useState(false);

    useEffect(() => {

        let interval = setInterval(() => {

            /// :: Atualizar contador.
            index += 10;
            if(index < 110)
                setPercentBar(index);

        }, 300);
        return () => clearInterval(interval);

    },[flag]);

    const openProgressBar = () => {
      index = 0;
        setPercentBar(0);
        setFlag(true);
        setVisibleBar(true);
    }

    const closeProgressBar = () => {
        index = 0;
        setPercentBar(0);
        setVisibleBar(false);
    }

    useImperativeHandle(ref, () => {
        return{
            openProgressBar,
            closeProgressBar
        }
    })

    return(
        <div className="col-md-6 offset-3 mt-2 mb-2" style={visibleBar ? {display:'block'} : {display:'none'}}>
            <div style={visibleBar ? {width:'100%', display:'block'} : {display:'none'}}>
                <ProgressBar animated now={percentBar} label={`${percentBar}%`} />
            </div>
        </div>
    )
}

export default forwardRef(ProgressBars);
```

24.2. Adicionar componente progressbar na página de cadastro e listagem de usuários

```
const progressBarRef = useRef<HandleProgressBar>(null);
const OpenProgressBar = useCallback(() => {
    progressBarRef.current?.openProgressBar();
}, []);

const CloseProgressBar = useCallback(() => {
    progressBarRef.current?.closeProgressBar();
}, []);

const onSubmit = (formData: Usuario) => {

     OpenProgressBar();
}

const getResult = (result: any) => {

  setTimeout(() => {

            CloseProgressBar();
  }, 3000);
}

<ProgressBars ref={progressBarRef}/>
```

25. Criar Componente de Modal e adicionar nas páginas usuário e listagem

25.1. Na pasta de components criar um arquivo modalCallback.tsx

```
import { faCheck, faTimes } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { forwardRef, useCallback, useImperativeHandle, useState } from 'react';
import Modal from 'react-bootstrap/Modal';

export interface ModalCallbackHandles {
  openModal: () => void;
}

type Props = {
  object: any,
  callback: Function
}

const ModalCallback: React.ForwardRefRenderFunction<ModalCallbackHandles, Props> = (props, ref) => {

  const [isOpen, setIsOpen] = useState(true);
  const [title, setTitle] = useState('');
  const [message, setMessage] = useState('');
  const [template, setTemplate] = useState('');
  const [bgColor, setBgColor] = useState('');
  const [icon, setIcon] = useState('');

  const refreshSetupModal = () => {
```

```
    setTitle(props.object.title);
    setMessage(props.object.message);
    setTemplate(props.object.template);
    setBgColor(props.object.bgColor);
    setIcon(props.object.icon);
}

setTimeout(() => {
    refreshSetupModal();
}, 20);

const openModal = useCallback(() => {
    setIsOpen(true);
}, []);

useImperativeHandle(ref, () => {
    return{
        openModal
    }
});

const hideModal = useCallback(() => {
    setIsOpen(false);
}, []);

if(!isOpen){
    return null;
}

const Confirm = (status: boolean) => {

    let button = status ? 'SIM' : 'NAO';

    if(template === 'confirm'){
        props.callback({'status': status, 'button': button, 'inputText': ''});
    }

    hideModal();
}

return (

    <Modal show={isOpen} onHide={hideModal}>
        <Modal.Header closeButton className={`bg-${bgColor} text-light`}>
            <Modal.Title>{title}</Modal.Title>
        </Modal.Header>
        <Modal.Body className='text-center'>
            <p dangerouslySetInnerHTML={{__html : message}}/>
            <div>
                {template === 'alert'
                ?
                    <div style={{display:'flex', justifyContent:'center', fontSize:'25px'}}>
                        <span onClick={hideModal} style={{cursor:'pointer'}}>{icon}</span>
                    </div>
                :
                    <div style={{display:'flex', justifyContent:'flex-end', width:'100%'}}>
                        <button className='btn btn-success btn-sm' onClick={() => Confirm(true)}>
```

```
                    <FontAwesomeIcon icon={faCheck} className='fas fa-plus'></FontAwesomeIcon>
                    <span>Sim</span>
                  </button>
                  <button className='btn btn-danger btn-sm' onClick={() => Confirm(false)} style={{marginLeft:'2%'}}>
                    <FontAwesomeIcon icon={faTimes} className='fas fa-plus'></FontAwesomeIcon>
                    <span>Não</span>
                  </button>
                </div>
              }
            </div>
          </Modal.Body>
        </Modal>
    );
}

export default forwardRef(ModalCallback);
```

25.2. Adicionar o componente de modalCallback nas paginas

```
/// :: Modal Callback
const [openModalCallback, setOpenModalCallback] = useState(false);
const [messageModal, setMessageModal] = useState<ModalTemplate>({} as ModalTemplate);
const modalCallbackRef = useRef<ModalCallbackHandles>(null);
const popupModalCallback = useCallback(() => {
    modalCallbackRef.current?.openModal();
}, []);
/// :: Modal Callback


const getResult = (result: any) => {

    let icon = <FontAwesomeIcon icon={faCheck} style={{color:'#198754'}}></FontAwesomeIcon>
    let width = '400px';
    let height = '120px';
    let message = '';
    let template = 'alert';
    let title = '';
    let bgColor = 'primary';

    setTimeout(() => {
            let message = null;

            if(result.status === 201){

                    title = 'Sucesso';
                    message = 'Nivel de satisfação cadastrado com sucesso!';
                    bgColor = 'primary';
                    createMessageModal(title, message, template, icon, height, width, 'placeholder', bgColor,
Function);

                    reset();
                    setTotalCaracters(0);
            }
            else if(result.status === 200){

                    title = 'Sucesso';
```

```
                        message = 'Nivel de satisfação Atualizado com sucesso!';
                        bgColor = 'primary';
                        createMessageModal(title, message, template, icon, height, width, 'placeholder', bgColor,
Function);
                        reset();
                        setTotalCaracters(0);
                        setTimeout(() => {
                                window.location.href = "/listagem";
                        },2000);
                }
                else {
                        title = 'Erro';
                        message = getErrorMessage(result.response.data);
                        bgColor = 'danger';
                        icon = <FontAwesomeIcon icon={faTimes} style={{color:'#dc3545'}}></FontAwesomeIcon>
                        createMessageModal(title, message, template, icon, height, width, 'placeholder', bgColor,
Function);
                }

                CloseProgressBar();
        }, 3000);
}

const createMessageModal = (title: string, message: string, template: string, icon: JSX.Element, height: string, width:
string, placeholder: string, bgColor: string, callback: Function) => {
        setMessageModal({title: title, message: message, template: template, icon: icon, height: height, width: width,
placeholder: placeholder, bgColor, callback: callback});
        setOpenModalCallback(true);
        popupModalCallback();
}

{openModalCallback && <ModalCallback object={messageModal} callback={messageModal.callback}
ref={modalCallbackRef} />}
```

25.3. Adicionar o componente de modalCallback na pagina de listagem para excluir registro

```
const confirmDelete = (id: number) => {

        currentId = id;
        let title = "Confirmação";
        let message = 'Ao clicar em confirmar esta avaliação será excluída. Tem certeza que deseja continuar?';
        let bgColor = 'warning';
        let icon = <FontAwesomeIcon icon={faWarning} style={{color:'#ffc107'}}></FontAwesomeIcon>
        createMessageModal(title, message, 'confirm', icon, '120px', '400px', 'placeholder', bgColor, excluirCadastro);
}

const excluirCadastro = (result: CallbackTemplate) =>{

        if(result.button === 'SIM'){
                OpenProgressBar();
                DeletePackage(currentId, "usuario", getResult);
        }
}

const getResult = (result: any) => {
```

```
        let icon = <FontAwesomeIcon icon={faCheck} style={{color:'#198754'}}></FontAwesomeIcon>
        let width = '400px';
        let height = '120px';
        let message = '';
        let template = 'alert';
        let title = '';
        let bgColor = 'primary';

        setTimeout(() => {

                if(result.status === 204){

                        title = 'Sucesso';
                        message = 'Avaliação excluída com sucesso!';
                        bgColor = 'primary';
                        createMessageModal(title, message, template, icon, height, width, 'placeholder', bgColor,
    Function);
                        setReloadTable(true);
                }
                else{
                        title = 'Erro';
                        message = getErrorMessage(result.response.data);
                        bgColor = 'danger';
                        icon = <FontAwesomeIcon icon={faTimes} style={{color:'#dc3545'}}></FontAwesomeIcon>
                        createMessageModal(title, message, template, icon, height, width, 'placeholder', bgColor,
    Function);
                }
        CloseProgressBar();
        }, 3000);
    }
```

26. Bônus Página de Login

```
import { useCallback, useRef, useState } from "react";
import { useForm } from "react-hook-form";
import { UserLogin } from "types/usuario";
import { UserLoginAuth} from "utils/api";
import { getErrorMessage } from "utils/getErrorMessage";
import { Result } from "types/resultServidor";
import ModalCallback, { ModalCallbackHandles } from "components/modalCallback";
import { ModalTemplate } from "types/modalTemplate";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faTimes } from "@fortawesome/free-solid-svg-icons";
import ProgressBars, { HandleProgressBar } from "components/progressbar";
import './styles.css';

const Login = () => {

   const { register, handleSubmit, formState: { errors } } = useForm<UserLogin>();
   const [errorMessage, setErrorMessage] = useState('');

   /// :: Modal Callback
   const [openModalCallback, setOpenModalCallback] = useState(false);
   const [messageModal, setMessageModal] = useState<ModalTemplate>({} as ModalTemplate);
   const modalCallbackRef = useRef<ModalCallbackHandles>(null);
   const popupModalCallback = useCallback(() => {
```

```
    modalCallbackRef.current?.openModal();
  }, []);
  /// :: Modal Callback

  const progressBarRef = useRef<HandleProgressBar>(null);
  const OpenProgressBar = useCallback(() => {
    progressBarRef.current?.openProgressBar();
  }, []);

  const CloseProgressBar = useCallback(() => {
    progressBarRef.current?.closeProgressBar();
  }, []);

  const onSubmit = (formData: UserLogin) => {

    OpenProgressBar();
    let email = formData.email.trim();
    let senha = formData.senha.trim();

    UserLoginAuth(email, senha, getResult);
  }

  const getResult = (result: Result) => {

    setTimeout(() => {

      if (result.data === "OK"){

        localStorage.clear();

        localStorage.setItem('token',"$2a$10$lmh.I.KXu7XCub");

        window.location.href = '/listagem';
      }
      else {

        let width = '400px';
        let height = '120px';
        let message = '';
        let template = 'alert';
        let title = 'Error';
        let bgColor = 'danger';
        const icon = <FontAwesomeIcon icon={faTimes} style={{color:'#dc3545'}}></FontAwesomeIcon>

        message = getErrorMessage(result.response.data as any);

        createMessageModal(title, message, template, icon, height, width, 'placeholder', bgColor, Function);
      }
      CloseProgressBar();
    }, 3000);
  }

  const createMessageModal = (title: string, message: string, template: string, icon: JSX.Element, height: string, width:
string, placeholder: string, bgColor: string, callback: Function) => {
    setMessageModal({title: title, message: message, template: template, icon: icon, height: height, width: width,
placeholder: placeholder, bgColor, callback: callback});
    setOpenModalCallback(true);
```

```
    popupModalCallback();
  }

  return(

    <>
      {openModalCallback && <ModalCallback object={messageModal} callback={messageModal.callback}
ref={modalCallbackRef} />}

      <div className="container">
        <div className="row mt-5">
          <div className={errorMessage ? "col-md-4 offset-4 error-mensagem auth-mensagem fade-in-message" :
'col-md-4 offset-4 auth-mensagem mensagem'} id={errorMessage ? 'shadow-error' : ''}>
            {errorMessage}
          </div>
        </div>
        <form onSubmit={handleSubmit(onSubmit)}>
          <div className="row">
            <div className="col-md-4 offset-4 mt-5">
              <h2>Login do Usúario</h2>
            </div>
            <div className="col-md-4 offset-4 mt-3">
              <label htmlFor="email">E-mail</label>
              <input className={`form-control my-2 ${errors.email ? 'is-invalid' : ''}`} type="text"
                {...register("email", {
                  required: 'email obrigatorio',
                  pattern: {
                    value: /^[^\s@]+@[^\s@]+\.[^\s@]+$/,
                    message: 'e-mail inválido'
                  }
                })}
                name="email"
                placeholder="E-mail"
              />
              {errors.email && <div className="invalid-feedback d-block">{errors.email.message}</div>}
            </div>
            <div className="col-md-4 offset-4 mt-3">
              <label htmlFor="senha">Senha</label>
              <input
                className={`form-control ${errors.senha ? 'is-invalid' : ''}`} type="password"
                {...register("senha", { required: true })}
                id="senha" name="senha" placeholder="Digite sua senha ..."
              />
              {errors.senha && <div className="invalid-feedback d-block">Senha é obrigatório</div>}
            </div>
            <div className="col-md-4 offset-4 mt-3">
              <button type="submit" className="btn btn-primary">Entrar</button>
            </div>
          </div>
        </form>

        <ProgressBars ref={progressBarRef}/>
      </div>
    </>
  );
}
```

```
export default Login;
```