	<p align="center">Centro Federal de Educação Tecnológica de Minas Gerais Campus VIII – Varginha Curso Técnico em Informática</p>	
<p><i>Disciplina</i> Aplicações para Web I</p>	<p align="center">FrontEnd cadastrando e autenticando no backend</p>	<p><i>Professor</i> Lázaro Eduardo da Silva</p>

Na aula de hoje vamos colocar o projeto React com Typescript para autenticar no backend projeto AdonisJS. Você precisará rodar os dois projetos. Vamos começar pelo projeto AdonisJS.

Para continuar o projeto, inicialmente, crie uma nova pasta, abra o Visual Studio Code, abra a pasta criado no Visual Studio Code, abra o terminal e verifique a versão do nodejs que está instalado no computador que você está utilizando. Para isso, execute o comando abaixo:

```
node -v
```

A versão deve ser 14 ou maior. Conferida a versão, vamos continuar o projeto anterior. Para isso, você precisa clonar o seu projeto na pasta com o comando:

```
git clone https://github.com/...
```

Todo projeto AdonisJS possui um arquivo .gitignore. Este arquivo é responsável por não subir no github os arquivos das bibliotecas e arquivos de configuração referente ao lugar, no qual a aplicação está executando. Para instalar os pacotes deve-se executar o comando abaixo:

```
npm install
```

O arquivo .env não vai para o github e você precisa criá-lo e configurá-lo. O arquivo .env.example é um modelo das configurações que precisam ser realizadas. Faça uma cópia dele e altere o nome para .env. Configure o banco de dados e para gerar uma APP_KEY, execute o comando abaixo no terminal:

```
node ace generate:key
```

Feito isso, seu projeto está pronto para ser executando com o comando

```
node ace serve --watch
```

ou

```
npm run dev
```


Ambos os comandos fazem a mesma função, iniciam a execução do seu projeto.

Para que o FrontEnd acesse o seu backend, você precisará realizar uma única alteração no seu projeto.

Ela deve ser feita no arquivo config/cors.ts na linha 23. O seu enabled deve estar false e você deve trocar para true.

Cors é uma segurança implementada nos backends que controla a origem das requisições. Quando ele não está ativado, o backend permite requisições somente da própria máquina que ele está executando. Não permite a requisição por outros projetos.

A imagem abaixo apresenta o código com a alteração realizada.

	<p align="center">Centro Federal de Educação Tecnológica de Minas Gerais Campus VIII – Varginha Curso Técnico em Informática</p>	
<p><i>Disciplina</i> Aplicações para Web I</p>	<p align="center">FrontEnd cadastrando e autenticando no backend</p>	<p><i>Professor</i> Lázaro Eduardo da Silva</p>

```

TS cors.ts M X
config > TS cors.ts > ...
1  /**
2   * Config source: https://git.io/JfefC
3   *
4   * Feel free to let us know via PR, if you find something broken in this config
5   * file.
6   */
7
8  import { CorsConfig } from '@ioc:Adonis/Core/Cors'
9
10 const corsConfig: CorsConfig = {
11   /*
12    |-----|
13    | Enabled
14    |-----|
15    |
16    | A boolean to enable or disable CORS integration from your AdonisJs
17    | application.
18    |
19    | Setting the value to `true` will enable the CORS for all HTTP request. However,
20    | you can define a function to enable/disable it on per request basis as well.
21    |
22    */
23   enabled: true,

```

Feito isso, vamos para o frontend. Crie uma nova pasta. Abra uma nova janela do Visual Studio Code. Abra a nova pasta criada no Visual Studio Code, abra o terminal e verifique a versão do nodejs que está instalado no computador que você está utilizando. Para isso, execute o comando abaixo:

```
node -v
```

A versão deve ser 14 ou maior. Conferida a versão, vamos continuar o projeto anterior. Para isso, você precisa clonar o seu projeto na pasta com o comando:

```
git clone https://github.com/... .
```

Todo projeto React possui um arquivo .gitignore. Este arquivo é responsável por não subir no github os arquivos das bibliotecas. Para instalar os pacotes deve-se executar o comando abaixo:

```
npm install
```

Para iniciar a execução do seu projeto execute o comando abaixo.

```
npm run start
```

Uma janela do navegador deve abrir e o seu projeto deve estar executando de acordo com a última alteração que você fez.

Vamos fazer alterações no nosso frontend para realizar o cadastro e o login no backend.

Para isso, vamos começar pelo arquivo de interface. Crie-o dentro da pasta src/interfaces/user.interface.ts com o conteúdo abaixo:

```
TS user.interface.ts U X
src > interfaces > TS user.interface.ts > ...
1  import { ReactNode } from "react"
2  export interface IUser {
3    name?: string
4    email?: string
5    password?: string
6  }
7  export interface IResponseUser {
8    user: {
9      id: number
10     name: string
11     email: string
12   }
13   token: {
14     token: string
15     expires_at: string
16   }
17 }
18 export interface IAuthContextData {
19   signIn(credentials: IUser): Promise<void>
20   signOut(): Promise<void>
21   loadUserStorageData(): Promise<boolean>
22   user: {
23     id: number
24     email: string
25     name: string
26   }
27 }
28 export interface IAuthProvider {
29   children?: ReactNode
30 }
31 export interface IErrorResponse {
32   status: string
33   message: string
34   errors?: {
35     message: string
36   }[]
37 }
```

Feito isso, vamos instalar alguns pacotes que serão importantes para respectivamente:

- fazermos as requisições para a API;

- verificar se o acesso expirou analisando a data;

- construir um componente de Loading (carregando) para apresentarmos ao usuário enquanto uma chamada retorna os dados e;

- um pacote para apresentarmos mensagens ao usuário quando realizarmos as ações.

npm install axios

npm install date-fns

npm install lottie-react

npm install react-toastify


Os links de documentação das bibliotecas são respectivamente:

<https://github.com/axios/axios>

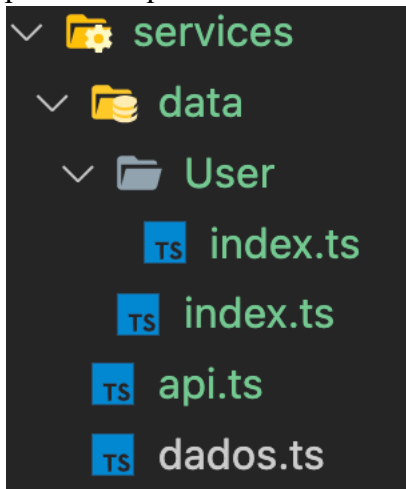
<https://date-fns.org/>

<https://lottiefiles.com/> (<https://lottiefiles.com/>)

<https://github.com/fkhadra/react-toastify>

	<p align="center">Centro Federal de Educação Tecnológica de Minas Gerais Campus VIII – Varginha Curso Técnico em Informática</p>	
<p><i>Disciplina</i> Aplicações para Web I</p>	<p align="center">FrontEnd cadastrando e autenticando no backend</p>	<p><i>Professor</i> Lázaro Eduardo da Silva</p>

Vamos para a API. Dentro da pasta services temos um único arquivo dados.ts. Vamos criar a estrutura de pastas e arquivos conforme imagem abaixo. O arquivo api.ts deve ter o seguinte conteúdo:



```

TS api.ts U X
src > services > TS api.ts > ...
1   import axios from 'axios'
2
3   const api = axios.create({
4     baseURL: 'http://127.0.0.1:3333',
5     headers: {
6       'Content-Type': 'application/json',
7     },
8   })
9
10  export default api

```

O arquivo que está na pasta src/services/data/User/index.ts deve ter o conteúdo abaixo. Este arquivo contém a classe com os métodos que farão as requisições aos backend.

```

TS index.ts U X
src > services > data > User > TS index.ts > ...
1   import { IResponseUser, IUser } from "interfaces/user.interface";
2   import api from "services/api";
3
4   class UserData {
5     register(data: IUser) {
6       return api.post<IResponseUser>('/register', data)
7     }
8     login(data: IUser) {
9       return api.post<IResponseUser>('/login', data)
10    }
11  }
12
13  export default new UserData()

```

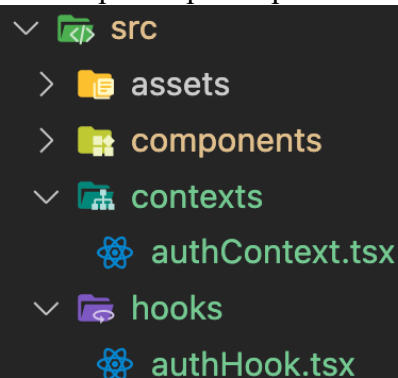
O arquivo que está na pasta src/services/data/index.ts deve ter o conteúdo abaixo.

```

TS index.ts U X
src > services > data > TS index.ts
1   export { default as apiUser } from './User'

```

Vamos passar para a parte da autenticação. Implementaremos dois recursos novos: contexto e hook.



Crie a pasta contexts e crie dentro dela o arquivo authContext.tsx.

Implementaremos neste arquivo um contexto que utilizaremos para autenticar os usuários e passar os dados para todas as páginas do frontend.

O arquivo abaixo está na pasta src/contexts/authContext.tsx.

Ele implementa o contexto e um provider, que utilizamos para passar os dados para outras páginas da aplicação.


```
authContext.tsx U X
src > contexts > authContext.tsx > ...
1  import {
2    IAuthContextData, IAuthProvider,
3    IResponseUser, IUser
4  } from '../interfaces/user.interface'
5  import React, { createContext, useState, useCallback } from 'react'
6  import api from '../services/api'
7  import { apiUser } from 'services/data'
8  import { isAfter, parseISO } from 'date-fns'
9  import { useNavigate } from 'react-router-dom'
10
11  const AuthContext = createContext<IAuthContextData>({} as IAuthContextData)
12
13  const AuthProvider: React.FC<IAuthProvider> = ({ children }) => {
14    const [auth, setAuth] = useState<IResponseUser>({} as IResponseUser)
15    const navigate = useNavigate();
16
17    const signIn = useCallback(async ({ email, password }: IUser) => {
18      const response = await apiUser.login({ email, password })
19      const { token, user } = response.data
20      api.defaults.headers.common.Authorization = `Bearer ${token}`
21      setAuth({ token, user })
22
23      localStorage.setItem('@web1:token', JSON.stringify(token))
24      localStorage.setItem('@web1:user', JSON.stringify(user))
25    }, [])
26
27    const removeLocalStorage = useCallback(async () => {
28      localStorage.removeItem('@web1:token')
29      localStorage.removeItem('@web1:user')
30    }, [])
31  }
```

authContext.tsx U X

src > contexts > authContext.tsx > ...

```
32   const signOut = useCallback(async () => {
33     setAuth({} as IResponseUser)
34     await removeLocalStorage()
35     delete api.defaults.headers.common.Authorization
36     navigate('/login')
37   }, [removeLocalStorage, navigate])
38
39   const loadUserStorageData = useCallback(async () => {
40     const token = JSON.parse(String(localStorage.getItem('@web1:token')))
41     const user = JSON.parse(String(localStorage.getItem('@web1:user')))
42     if (token && user) {
43       api.defaults.headers.common.Authorization = `Bearer ${token.token}`
44       setAuth({ token, user })
45       if (isAfter(parseISO(token.expires_at), new Date())) {
46         return true
47       } else {
48         return false
49       }
50     } else {
51       return false
52     }
53   }, [])
54
55   return (
56     <AuthContext.Provider
57       value={{
58         signIn,
59         signOut,
60         loadUserStorageData,
61         user: auth.user,
62       }}
63     >
64       {children}
65     </AuthContext.Provider>
66   )
67 }
68 export { AuthProvider, AuthContext }
```


O arquivo que está na pasta src/hooks/authHook.tsx será utilizado para recuperar os dados do contexto.

 **authHook.tsx** U X

src > hooks >  authHook.tsx > ...

```
1  import { AuthContext } from "contexts/authContext"
2  import { IAuthContextData } from "interfaces/user.interface"
3  import { useContext } from "react"
4
5  function useAuth(): IAuthContextData {
6      const context = useContext(AuthContext)
7      if (!context) {
8          throw new Error('useAuth deve ser utilizado com o AuthProvider')
9      }
10     return context
11 }
12
13 export { useAuth }
```

Feito isso, vamos criar um arquivo na pasta src/components/Loading/index.tsx. Ele será nosso componente de carregando.

 **index.tsx** U X

src > components > Loading >  index.tsx > ...

```
1  import { useLottie } from "lottie-react";
2  import loading from 'lottie/loading.json'
3  export default function Loading() {
4
5      const options = {
6          animationData: loading,
7          loop: true,
8          autoplay: true,
9      };
10
11     const { View } = useLottie(options, { height: 300 });
12
13     return View;
14 }
```


O arquivo lottie/loading.json é um arquivo que deve ser baixado no site <https://lottiefiles.com/>

Crie um pasta lottie dentro de src, escolha um loading no site, faça o download do json dele e salve na pasta lottie com o nome de loading.json.

Feito isso, vamos criar o componente que irá proteger a rota autenticada. Para isso, deve-se criar um arquivo na pasta src/routes/ProtectedRoute.tsx com o conteúdo abaixo:

```
ProtectedRoute.tsx U X

src > routes > ProtectedRoute.tsx > ...

1  import Loading from 'components/Loading';
2  import { useAuth } from 'hooks/authHook'
3  import { useEffect, useState } from 'react'
4  import { Navigate, Outlet } from 'react-router-dom'
5
6  export default function ProtectedRoute() {
7    const [isLoading, setIsLoading] = useState(true);
8    const [isAuth, setIsAuth] = useState(true);
9    const { loadUserStorageData } = useAuth()
10
11    useEffect(() => {
12      async function fetchData() {
13        const verifyAuth = await loadUserStorageData()
14        setIsAuth(verifyAuth)
15        setIsLoading(false)
16      }
17      fetchData()
18    }, [loadUserStorageData])
19
20    if (isLoading) {
21      return <Loading />
22    }
23
24    return (
25      <>
26        {isAuth ? (
27          <Outlet />
28        ) : (
29          <Navigate to='/login' replace={true} />
30        )}
31      </>
32    )
33  }
```


Para testarmos a área administrativa, vamos criar uma página bem simples, que modificaremos na próxima parte da atividade. Ela deve estar dentro de src/pages/Adm. O arquivo de estilo e a página estão abaixo:

```
src > pages > Adm > styles.ts > ...
1 import styled from "styled-components"
2
3 export const Adm = styled.section`
4   font-size: 2rem;
5 `
```

Inclua a página Adm no export das pages no arquivo src/pages/index.ts

```
src > pages > index.ts
1 export { default as AdmPage } from "./Adm"
```


```
src > pages > Adm > index.tsx > ...
1 import * as S from "../styles";
2
3 const Adm = () => {
4   return (
5     <S.Adm>
6       Área administrativa
7     </S.Adm>
8   );
9 };
10
11 export default Adm;
```

Para utilizarmos a rota privada e a página Adm, vamos fazer as modificações abaixo no arquivo de rotas em src/routes/index.tsx

```
src > routes > index.tsx > ...
1 import { Routes, Route } from "react-router-dom";
2 import { AdmPage, CadastrarPage, HomePage, LayoutPage, LoginPage } from "pages";
3 import ProtectedRoute from "../ProtectedRoute";
4 const Rotas = () => {
5   return (
6     <Routes>
7       <Route element={<LayoutPage />} />
8       <Route path="/" element={<HomePage />} />
9       <Route path="/cadastro" element={<CadastrarPage />} />
10      <Route path="/login" element={<LoginPage />} />
11      <Route element={<ProtectedRoute />} />
12        <Route path="/adm" element={<AdmPage />} />
13      </Route>
14    </Route>
15  </Routes>
16 );
17 };
18
19 export default Rotas;
```

Algumas modificações no arquivo principal da aplicação são necessárias para que os dados do contexto se propaguem para todas as páginas da aplicação.

O arquivo src/App.tsx deve ser modificado para ficar como o arquivo abaixo:

	<p align="center">Centro Federal de Educação Tecnológica de Minas Gerais Campus VIII – Varginha Curso Técnico em Informática</p>	
<p><i>Disciplina</i> Aplicações para Web I</p>	<p align="center">FrontEnd cadastrando e autenticando no backend</p>	<p><i>Professor</i> Lázaro Eduardo da Silva</p>

```

App.tsx M X
src > App.tsx > ...
1  import { BrowserRouter } from "react-router-dom";
2  import GlobalStyle from "styles/GlobalStyle";
3  import { ToastContainer } from "react-toastify";
4  import Routes from "routes";
5  import { AuthProvider } from "contexts/authContext";
6
7  function App() {
8      return (
9          <BrowserRouter>
10             <GlobalStyle />
11             <ToastContainer autoClose={3000} />
12             <AuthProvider>
13                 <Routes />
14             </AuthProvider>
15         </BrowserRouter>
16     );
17 }
18
19 export default App;

```

Para que as mensagens de alerta apareçam na tela, é necessário ainda incluir o import da linha 2 na pasta src/styles/GlobalStyles.ts

```

GlobalStyle.ts M X
src > styles > GlobalStyle.ts > ...
1  import { createGlobalStyle } from "styled-components"
2  import 'react-toastify/dist/ReactToastify.css';
3

```

Precisamos ainda fazer alterações no Cadastrar e no Login.

O arquivo src/pages/Cadastrar/index.tsx deve ficar como a imagem abaixo:

```

index.tsx M X
src > pages > Cadastrar > index.tsx > ...
1  import { FormEvent, useState } from "react";
2  import { Link, useNavigate } from "react-router-dom";
3  import { FaKey } from "react-icons/fa";
4  import { BsFillPersonFill } from "react-icons/bs";
5  import { MdEmail } from "react-icons/md";
6  import { toast } from "react-toastify";
7
8  import * as S from "../styles";
9  import { ButtonComponent } from "components";
10 import { IErrorResponse, IUser } from "interfaces/user.interface";
11 import { AxiosError } from "axios";
12 import { apiUser } from "services/data";

```

index.tsx M X

src > pages > Cadastrar > index.tsx > ...

```
13
14  const Cadastrar = () => {
15    const navigate = useNavigate();
16    const [formData, setFormData] = useState<IUser>({
17      name: '',
18      email: '',
19      password: '',
20    })
21    async function handleChange(e: IUser) {
22      setFormData((state: IUser | undefined) => ({ ...state, ...e }))
23    }
24    async function handleSubmit(event: FormEvent<HTMLFormElement>) {
25      event.preventDefault()
26      try {
27        await apiUser.register(formData);
28        toast.success("Cadastro realizado com sucesso!");
29        navigate('/login')
30      } catch (error) {
31        const err = error as AxiosError<IErrorResponse>
32        let messages = err.response?.data.message
33        if (err.response?.data.errors) {
34          messages = err.response?.data.errors?.map((i) => i.message)
35            .reduce((total, cur) => `${total} ${cur}`)
36        }
37        toast.error(messages)
38      }
39    }
```

```

index.tsx M X
src > pages > Cadastrar > index.tsx > ...
40     return (
41         <S.Section>
42             <h1>Cadastre-se</h1>
43             <form method="post" onSubmit={handleSubmit}>
44                 <label htmlFor="nome">Nome</label>
45                 <div>
46                     <BsFillPersonFill />
47                     <input type="text" name="name" id="nome" placeholder="Nome"
48                         onChange={(e) => handleChange({ name: e.target.value })}
49                         value={formData?.name}
50                     />
51                 </div>
52                 <label htmlFor="email">E-mail</label>
53                 <div>
54                     <MdEmail />
55                     <input type="email" name="email" id="email" placeholder="E-mail"
56                         onChange={(e) => handleChange({ email: e.target.value })}
57                         value={formData?.email}
58                     />
59                 </div>
60                 <label htmlFor="senha">Senha</label>
61                 <div>
62                     <FaKey />
63                     <input type="password" name="senha" id="senha" placeholder="Senha"
64                         onChange={(e) => handleChange({ password: e.target.value })}
65                         value={formData?.password}
66                     />
67                 </div>
68                 <p>
69                     Já possui conta? <Link to="/login">Faça o login</Link>
70                     <ButtonComponent>Salvar</ButtonComponent>
71                 </p>
72             </form>
73         </S.Section>
74     );
75 };
76 export default Cadastrar;

```

O arquivo src/pages/Login/index.tsx deve ficar como a imagem abaixo:

index.tsx M X

src > pages > Login > index.tsx > ...

```
1  import { FormEvent, useState } from "react";
2  import { Link, useNavigate } from "react-router-dom";
3  import { FaKey } from "react-icons/fa";
4  import { MdEmail } from "react-icons/md";
5  import { toast } from "react-toastify";
6
7  import * as S from "./styles";
8  import { ButtonComponent } from "components";
9  import { useAuth } from "hooks/authHook";
10 import { IErrorResponse, IUser } from "interfaces/user.interface";
11 import { AxiosError } from "axios";
12
13 const Login = () => {
14   const navigate = useNavigate();
15   const { signIn } = useAuth()
16   const [formData, setFormData] = useState<IUser>({
17     email: '',
18     password: '',
19   })
20   async function handleChange(e: IUser) {
21     setFormData((state: IUser | undefined) => ({ ...state, ...e }))
22   }
23   async function handleSubmit(event: FormEvent<HTMLFormElement>) {
24     event.preventDefault()
25     try {
26       const { email, password } = formData
27       await signIn({
28         email: String(email),
29         password: String(password),
30       })
31       toast.success("Login realizado com sucesso!");
32       navigate('/adm')
33     } catch (error) {
34       const err = error as AxiosError<IErrorResponse>
35       toast.error(String(err.response?.data))
36     }
37   }
```

index.tsx M X

src > pages > Login > index.tsx > ...

```
38
39   return (
40     <S.Section>
41       <h1>Login</h1>
42       <form method="post" onSubmit={handleSubmit}>
43         <label htmlFor="email">E-mail</label>
44         <div>
45           <MdEmail />
46           <input type="email" name="email" id="email" placeholder="E-mail"
47             onChange={(e) => handleChange({ email: e.target.value })}
48             value={formData?.email}
49           />
50         </div>
51         <label htmlFor="senha">Senha</label>
52         <div>
53           <FaKey />
54           <input type="password" name="senha" id="senha" placeholder="Senha"
55             onChange={(e) => handleChange({ password: e.target.value })}
56             value={formData?.password}
57           />
58         </div>
59         <p>
60           Não possui conta? <Link to="/cadastrar">Cadastre-se</Link>
61           <ButtonComponent>Entrar</ButtonComponent>
62         </p>
63       </form>
64     </S.Section>
65   );
66 };
67
68 export default Login;
```

Para finalizar a interface vamos fazer uma modificação no componente de Menu. Quando o usuário estiver autenticado, vamos mostrar o nome dele e colocar um botão para sair.


index.tsx M X

src > components > Menu > index.tsx > ...

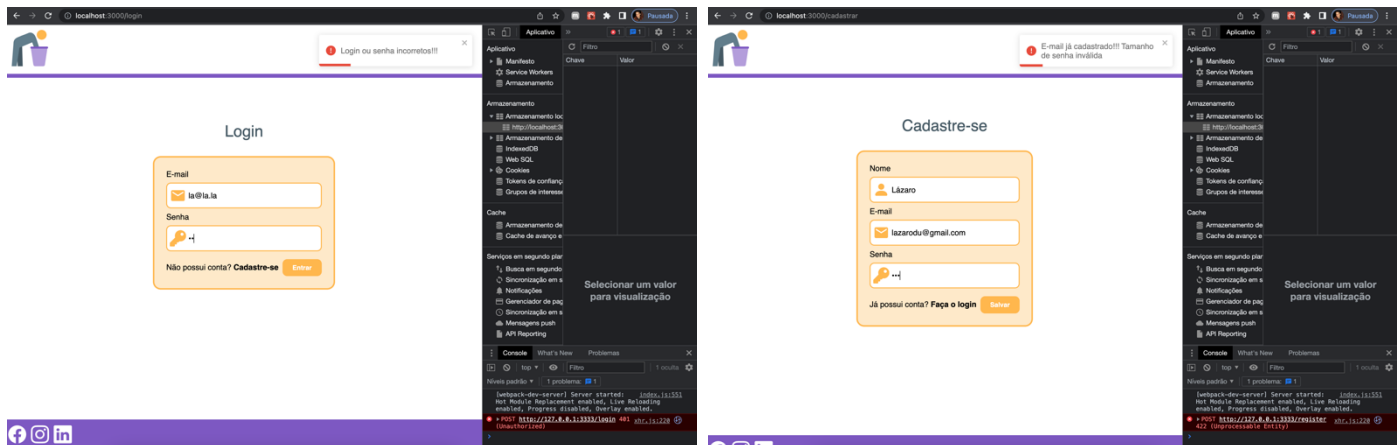
```
1  import { FcReuse } from "react-icons/fc";
2  import { GrLogout } from "react-icons/gr";
3  import * as S from "./styles";
4  import { Link, useNavigate } from "react-router-dom";
5  import { useAuth } from "hooks/authHook";
6  const Menu = () => {
7    const { user, signOut } = useAuth()
8    const navigate = useNavigate();
9    async function logout() {
10      await signOut()
11      navigate('/login')
12    }
13    return (
14      <S.Cabecalho>
15        <picture>
16          <Link to="/">
17            <FcReuse />
18          </Link>
19        </picture>
20        <nav>
21          {
22            user ? (
23              <ul>
24                <li>
25                  <button onClick={logout}>{user.name} <GrLogout /></button>
26                </li>
27              </ul>
28            ) : (
29              <ul>
30                <li>
31                  <Link to="/login">Login</Link>
32                </li>
33                <li>
34                  <Link to="/cadastrar">Cadastrar</Link>
35                </li>
36              </ul>
37            )
38          }
39        </nav>
40      </S.Cabecalho >
41    );
42  };
43  export default Menu;
```


O arquivo de estilo deve ter pequenas modificações também, são elas:

```
TS styles.ts M X
src > components > Menu > TS styles.ts > ...
1  import styled from "styled-components"
2  import { colors } from "styles/GlobalStyle"
3  export const Cabecalho = styled.header`
4      display: flex;
5      border-bottom: 0.5rem solid ${colors.secondary};
6      picture svg {
7          height: 6rem;
8          width: 6rem;
9      }
10     nav {
11         width: 100%;
12         display: flex;
13         justify-content: flex-end;
14         align-items: center;
15         ul {
16             list-style-type: none;
17             display: flex;
18             align-items: center;
19             li {
20                 margin: 0 1.5em;
21                 a {
22                     text-decoration: none;
23                     font-size: 1.2em;
24                     color: ${colors.primary};
25                 }
26                 button {
27                     display: flex;
28                     gap: 0.5rem;
29                     align-items: center;
30                     text-decoration: none;
31                     font-size: 1.2em;
32                     color: ${colors.primary};
33                     border: 0;
34                     background-color: transparent;
35                     cursor: pointer;
36                 }
37             }
38         }
39     }
40 }
```

	<p align="center">Centro Federal de Educação Tecnológica de Minas Gerais Campus VIII – Varginha Curso Técnico em Informática</p>	
<p>Disciplina Aplicações para Web I</p>	<p align="center">FrontEnd cadastrando e autenticando no backend</p>	<p>Professor Lázaro Eduardo da Silva</p>

Feito isso, podemos testar. Tente acessar o endereço localhost:3000/adm, o sistema deve te redirecionar para a tela de login. Abra o inspecionar, clique na aba Aplicativo e depois em armazenamento local. Você verá que ele fica vazio enquanto você não se autentica.



Ao fazer o login, observe que o local storage está preenchido com os dados do token e do usuário. O menu agora não possui as opções de Login e Cadastrar, mas tem a opção com o nome do usuário que fez o login com o botão de sair, que ao ser clicado, limpa o local storage.

