# Aula 7 - Segurança

**Docupedia Export**

Author:Ferro Alisson (CtP/ETS)
Date:11-Sep-2023 13:18

# Table of Contents

# 1 Protected Route

Até o momento nosso login ainda não serve para muita coisa, realmente é enviado para a API, mas não tem um protetor para as rotas.
Para isso, iremos criar o arquivo de protected route recebendo como props 2 parâmetros, errorPage e targetPage. sendo o primeiro a página caso ocorra erro, e o segundo caso ocorra sucesso.
Vamos também criar um useState page e setPage que por default vem uma tag vazia.
Agora iremos criar a função mesmo que verifica o token recebido, para isso precisaremos instalar a biblioteca jwt-decode com o comando

```
npm i jwt-decode
```

e importa-lo no ProtectedRoute

Por fim precisamos retornar a page. Ficando assim nosso protectedRoute

```
import { useEffect, useState } from "react";
import jwt_decode from 'jwt-decode';

export default function ProtectedRoute({ errorPage, targetPage, func }){
    var [page, setPage] = useState(<></>);

    function renderPage(){
        const token = sessionStorage.getItem('token');

        if(!token) {
            setPage(errorPage)
            return
        }

        const decodeToken = jwt_decode(token)
        const { exp } = decodeToken;

        if(exp+'000' - Date.now()){
            setPage(errorPage)
            return
        }
        setPage(targetPage)
```

```
    }

    useEffect(() => {
        renderPage()
    }, [])

    return page;
}
```

Agora iremos colocar a protectRoute nas rotas em que queremos proteger;
Em **App.js**

```
function App() {
  return (
    <>
        <Routes>
          <Route path='/' element={<LoginPage />} />
          <Route path='/register' element={<RegisterPage />} />
          <Route path='/main' element={
            <ProtectedRoute
              errorPage={<AccessDenied />}
              targetPage={<NavBar />}
            />
          }>
            <Route path='' element={<HomePage />} />
          </Route>
            <Route path='*' element={<NotFoundPage />} />
        </Routes>
    </>
  );
}
```

Onde o protectedRoute vai englobar todas as rotas que são protegidas, no nosso caso somente a HomePage por enquanto é protegida

Vamos utilizar uma criptografia para enviar o login para a API, para isso, instalamos a biblioteca CryptoJS com

```
npm i crypto-js
```

e após a verificação do login, antes de enviar para a API vamos criptografar os dados.
Agora é com você:

## 1.1 Desafio: Com a API de person criada na aula de NodeJS, crie uma página para o usuário se registrar, e fazer o login, protegendo assim a página Home e com criptografia tanto no login, quanto no register.