por Gabriel Viegas Dantas | 84.7k xp | 2 posts

Olá, sei que não é assunto do curso, mas gostaria de uma ajuda: Como funciona a autenticação por meio de JWT com React. Estou desenvolvendo uma API que vai gerar esse token e vai mandar pro Front (feito com React) no momento do Login mas como fazer? E o que fazer depois?

3 respostas

por Renan Lima | 3364.2k xp | 7111 posts Alura Scuba Team

Apoio Educacional - Alura

3 meses atrás

Olá Gabriel.

Tudo bem?

Entendo sua dúvida e vou tentar te ajudar com um exemplo prático sobre como implementar a autenticação usando JWT (JSON Web Token) com React.

Passo 1: Gerar o JWT na API (Back-end)

No seu back-end (Node.js), você precisa criar uma rota de login que vai gerar o token JWT quando o usuário fornecer credenciais válidas. Aqui está um exemplo usando o pacote jsonwebtoken:

const express = require('express');

const jwt = require('jsonwebtoken');

const app = express();

app.use(express.json());

const users = [{ id: 1, username: 'user', password: 'password' }]; // Exemplo de usuário

app.post('/login', (req, res) => {

const { username, password } = req.body;

const user = users.find(u => u.username === username && u.password === password);

if (user) {

const token = jwt.sign({ id: user.id }, 'seu\_segredo', { expiresIn: '1h' });

res.json({ token });

} else {

res.status(401).json({ error: 'Credenciais inválidas' });

}

});

app.listen(3000, () => {

console.log('Servidor rodando na porta 3000');

});

Copiar código

Passo 2: Consumir o JWT no Front-end (React)

No seu front-end em React, você precisa fazer a requisição para a API de login e armazenar o token JWT recebido. Aqui está um exemplo usando fetch:

import React, { useState } from 'react';

function Login() {

const [username, setUsername] = useState('');

const [password, setPassword] = useState('');

const [token, setToken] = useState(null);

const handleLogin = async () => {

const response = await fetch('http://localhost:3000/login', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ username, password }),

});

const data = await response.json();

if (response.ok) {

setToken(data.token);

localStorage.setItem('token', data.token); // Armazenar o token no localStorage

} else {

alert(data.error);

}

};

return (

<div>

<input

type="text"

value={username}

onChange={(e) => setUsername(e.target.value)}

placeholder="Username"

/>

<input

type="password"

value={password}

onChange={(e) => setPassword(e.target.value)}

placeholder="Password"

/>

<button onClick={handleLogin}>Login</button>

{token && <p>Token: {token}</p>}

</div>

);

}

export default Login;

Copiar código

Passo 3: Utilizar o JWT para acessar rotas protegidas

Para acessar rotas protegidas na sua API, você precisa enviar o token JWT no cabeçalho das requisições. Aqui está um exemplo de como fazer isso:

const fetchProtectedData = async () => {

const token = localStorage.getItem('token');

const response = await fetch('http://localhost:3000/protected', {

headers: {

'Authorization': `Bearer ${token}`,

},

});

const data = await response.json();

if (response.ok) {

console.log(data);

} else {

console.error('Erro ao acessar a rota protegida', data);

}

};

Copiar código

E no seu back-end, você precisa verificar o token nas rotas protegidas:

const verifyToken = (req, res, next) => {

const token = req.headers['authorization']?.split(' ')[1];

if (!token) return res.status(403).json({ error: 'Token não fornecido' });

jwt.verify(token, 'seu\_segredo', (err, decoded) => {

if (err) return res.status(401).json({ error: 'Token inválido' });

req.userId = decoded.id;

next();

});

};

app.get('/protected', verifyToken, (req, res) => {

res.json({ message: 'Acesso concedido', userId: req.userId });

});

Copiar código

É só um exemplo para você visualizar e ter uma noção de como é feito.

Espero ter ajudado de alguma forma. Qualquer dúvida manda aqui. Bons estudos.

por Gabriel Viegas Dantas | 84.7k xp | 2 posts

3 meses atrás

Entendi. Agora, só mais uma dúvida: No curso, não é usado um banco de dados para armazenar as imagens. Estava pesquisando sobre isso e consegui armazenar as imagens no meu projeto em formato longblob (usei o mysql). Mas ao fazer um GET para mostrar as imagens no navegador, vi que só aparecia o texto alternativo. Pelo que procurei a respeito, eu devo transformar essa imagem em um 'base64' para ai sim conseguir mostrar a imagem no navegador, mas não encontro muitas informações na internet sobre isso, consegue me dar um exemplo a respeito?

solução!

por Renan Lima | 3364.2k xp | 7111 posts Alura Scuba Team

Apoio Educacional - Alura

3 meses atrás

Claro, posso te ajudar com um exemplo convertendo para base64, precisa converter mesmo quando pega do banco:

Passo 1: Armazenar Imagens no Banco de Dados

Você já conseguiu armazenar as imagens no formato longblob no MySQL, então o próximo passo é recuperar essas imagens e transformá-las em um formato que o navegador possa exibir.

Passo 2: Recuperar e Transformar Imagens para Base64

Quando você recuperar a imagem do banco de dados, você deve convertê-la para uma string base64 antes de enviá-la para o frontend. Aqui está um exemplo de como fazer isso no seu backend com Node.js e Express:

Backend (Node.js)

const express = require('express');

const mysql = require('mysql');

const app = express();

// Configuração do banco de dados

const db = mysql.createConnection({

host: 'localhost',

user: 'seu\_usuario',

password: 'sua\_senha',

database: 'seu\_banco\_de\_dados'

});

db.connect(err => {

if (err) {

console.error('Erro ao conectar ao banco de dados:', err);

return;

}

console.log('Conectado ao banco de dados');

});

// Rota para obter a imagem

app.get('/image/:id', (req, res) => {

const { id } = req.params;

const query = 'SELECT image FROM images WHERE id = ?';

db.query(query, [id], (err, results) => {

if (err) {

res.status(500).json({ error: 'Erro ao buscar imagem' });

return;

}

if (results.length > 0) {

const imageBuffer = results[0].image;

const base64Image = imageBuffer.toString('base64');

res.json({ image: base64Image });

} else {

res.status(404).json({ error: 'Imagem não encontrada' });

}

});

});

app.listen(3000, () => {

console.log('Servidor rodando na porta 3000');

});

Copiar código

Passo 3: Exibir Imagem no Frontend (React)

No frontend, você pode consumir essa rota para buscar a imagem e exibi-la. Aqui está um exemplo de como fazer isso em React:

Frontend (React)

import React, { useEffect, useState } from 'react';

function ImageDisplay({ imageId }) {

const [image, setImage] = useState(null);

useEffect(() => {

const fetchImage = async () => {

try {

const response = await fetch(`http://localhost:3000/image/${imageId}`);

const data = await response.json();

if (response.ok) {

setImage(`data:image/jpeg;base64,${data.image}`);

} else {

console.error(data.error);

}

} catch (error) {

console.error('Erro ao buscar imagem:', error);

}

};

fetchImage();

}, [imageId]);

return (

<div>

{image ? <img src={image} alt="Imagem" /> : <p>Carregando imagem...</p>}

</div>

);

}

export default ImageDisplay;

Copiar código

Explicação

Backend:

No backend, recuperamos a imagem do banco de dados como um buffer de dados.

Convertendo esse buffer em uma string base64 usando toString('base64').

Enviamos essa string base64 como parte da resposta JSON.

Frontend:

No frontend, fazemos uma requisição para a API para obter a imagem.

Convertendo a string base64 recebida em uma URL de dados (data URL) que o elemento <img> pode usar.

Exibimos a imagem no navegador.

Certifique-se de que o tipo de imagem (image/jpeg) no data URL corresponda ao tipo de imagem armazenada no banco de dados. Se você estiver usando um tipo de imagem diferente (por exemplo, image/png), ajuste o tipo conforme necessário.

Esse exemplo pressupõe que a imagem é do tipo JPEG. Para outros tipos de imagem, altere image/jpeg para o tipo MIME correto.

Espero ter ajudado. Qualquer dúvida manda aqui. Bons estudos.