TIC em trilhas

Módulo 03

CORS

TIC em trilhas

Aula 1 - CORS

Conceitos, funcionamento e exemplo prático

Imagem extraída através do acervo gratuito do Freepik

1.1 O que é?

CORS, ou Cross-Origin Resource Sharing, é uma política de segurança nos navegadores que controla como os recursos de um site podem ser acessados por outros domínios.

Essa política permite que um site conceda permissão para que outros domínios acessem seus recursos, ajudando a proteger contra possíveis ataques, ao mesmo tempo em que permite interações entre diferentes sites de forma controlada.

Essa permissão é concedida através do cabeçalho HTTP Access-Control-Allow-Origin, que o servidor envia para indicar quais domínios têm permissão para acessar o recurso.

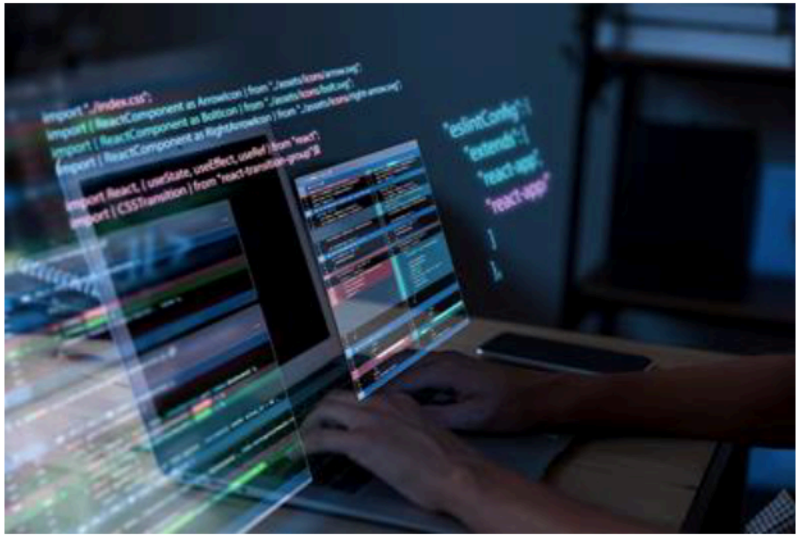

Imagem extraída através do acervo gratuito do Freepik

1.2 Como ocorre

Primeiro, quando o navegador faz uma solicitação HTTP, ele adiciona um cabeçalho chamado "Origin" à solicitação, indicando de onde ela está vindo.

Em seguida, o servidor que hospeda o recurso verifica se a origem (indicada pelo cabeçalho "Origin") tem permissão para acessar o recurso. Se o servidor determinar que a origem tem permissão, ele incluirá um cabeçalho "Access-Control-Allow-Origin" na resposta HTTP. Este cabeçalho indica explicitamente que aquele domínio específico tem permissão para acessar o recurso.

Assim, quando o navegador recebe a resposta com o cabeçalho "Access-Control-Allow-Origin", ele permite que o recurso seja carregado e usado pela página que fez a solicitação. No entanto, se o servidor não incluir o cabeçalho "Access-Control-Allow-Origin" ou se o domínio que faz a solicitação não estiver na lista permitida, o navegador bloqueará a solicitação por motivos de segurança.

TIC em trilhas

1.3 Exemplo prático

- Criar uma aplicação em Node.js com uma rota simples.
- Realizar uma requisição para essa rota via JavaScript.
- Implementar a dependência CORS para lidar com o problema de CORS.

No terminal, crie uma nova pasta para o projeto e inicie o Node.js:

```
mkdir cors-example
cd cors-example
npm init -y
```

Instale o Express para criar o servidor e o CORS para lidar com as políticas de CORS:

```
npm install express cors
```

E crie um arquivo `index.js` com o seguinte código:

```
const express = require('express');
const cors = require('cors');

const app = express();
const PORT = 3000;

// Configurar o CORS para permitir todas as origens
app.use(cors());

// Rota simples
app.get('/api/dados', (req, res) => {
    res.json({ mensagem: "Dados recebidos do servidor!" });
});

// Iniciar o servidor
app.listen(PORT, () => {
    console.log(`Servidor rodando na porta ${PORT}`);
});
```

Do autor

TIC em trilhas

Agora, crie um arquivo HTML simples para realizar uma requisição para a rota que criamos no Node.js.

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemplo de CORS</title>
</head>
<body>
    <h1>Teste de CORS</h1>
    <button onclick="fazerRequisicao()">Fazer Requisição</button>
    <p id="resultado"></p>

    <script>
   </body>
</html>
```

TIC em trilhas

E dentro da tag <body> </body> adicione:

```
<script>
    function fazerRequisicao() {
        fetch('http://localhost:3000/api/dados')
            .then(response => response.json())
            .then(data => {
                document.getElementById('resultado').textContent = data.mensagem;
            })
            .catch(error => {
                console.error('Erro:', error);
            });
    }
</script>
```

Para testar o código no terminal execute o comando:

```
node index.js
```

Abra o arquivo index.html no navegador (você pode simplesmente abrir o arquivo localmente no seu navegador) e quando você clicar no botão "Fazer Requisição", o navegador fará uma requisição ao servidor Node.js, e a resposta será exibida na página.

1.4 Por que CORS é necessário?

O CORS (Cross-Origin Resource Sharing) é um mecanismo que permite que recursos restritos em uma página da web sejam solicitados de outro domínio fora do domínio ao qual o recurso pertence. Sem configurar CORS, o navegador bloquearia a requisição por questões de segurança, especialmente em aplicações que acessam APIs em servidores diferentes.

Com o código do exemplo, o problema de CORS é tratado adequadamente, permitindo que o frontend (navegador) faça requisições para o backend (servidor Node.js) sem ser bloqueado.

1.5 Explore mais

O problema de CORS é algo normal no cotidiano dos desenvolvedores, neste módulo aprendemos o conceito do CORS, funcionamento e como resolver, abaixo estão alguns materiais complementares sobre este assunto:

- MDN Web Docs - CORS
- AWS (Amazon) - O que é CORS

Bons estudos 😉

Imagem extraída através do acervo gratuito do Freepik

TIC em trilhas

Referências bibliográficas

MDN Web Docs: CORS. Disponível em: https://developer.mozilla.org/pt-BR/docs/Glossary/CORS. Acesso em fevereiro de 2024.

AWS: O que é CORS. Disponível em: https://aws.amazon.com/pt/what-is/cross-origin-resource-sharing. Acesso em fevereiro de 2024.

Alura: O que é CORS e como resolver um erro de Cross-Origin Resource Sharing. Disponível em: https://www.alura.com.br/artigos/como-resolver-erro-de-cross-origin-resource-sharing. Acesso em fevereiro de 2024.

Treinaweb: O que é CORS e como resolver os principais erros. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-cors-e-como-resolver-os-principais-erros. Acesso em fevereiro de 2024.