Exemplo completo e funcional que demonstra todos os passos para proteger sua aplicação contra CSRF usando `Express`, `cookie-parser` e `csurf`.

1. Instale as dependências:

npm install express cookie-parser csurf

2. Código completo: servidor com proteção CSRF

```
// app.js
const express = require('express');
const cookieParser = require('cookie-parser');
const csrf = require('csurf');

const app = express();

// Middlewares básicos
app.use(express.urlencoded({ extended: true }));
app.use(express.json());
app.use(cookieParser());

// Habilita proteção CSRF via cookie
const csrfProtection = csrf({ cookie: true });
app.use(csrfProtection);

// Rota pública: entrega o token CSRF para o frontend
app.get('/csrf-token', (req, res) => {
  res.json({ csrfToken: req.csrfToken() });
});

// Rota protegida: exige token CSRF para POST
app.post('/envio-dados', (req, res) => {
  const { nome, mensagem } = req.body;
  res.json({
    status: 'ok',
    nome,
    mensagem,
    aviso: 'Requisição protegida e validada com sucesso.'
  });
});

// Inicia o servidor
app.listen(3000, () => {
  console.log('Servidor com CSRF rodando na porta 3000');
});
```

3. Requisição do lado do cliente (exemplo com fetch):

1. Primeiro, obtenha o token:

```
const csrf = await fetch('/csrf-token').then(res => res.json());
```

2. Depois, envie no corpo da requisição:

```
fetch('/envio-dados', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    nome: 'Maria',
    mensagem: 'Formulário seguro!',
    _csrf: csrf.csrfToken
  })
});
```

Aqui está o mesmo exemplo em React:

```
// App.jsx
import React, { useEffect, useState } from 'react';

function App() {
  const [csrfToken, setCsrfToken] = useState('');
  const [mensagem, setMensagem] = useState('');
  const [nome, setNome] = useState('');
  const [resposta, setResposta] = useState(null);

  // Busca o token CSRF uma única vez
  useEffect(() => {
    fetch('http://localhost:3000/csrf-token', {
      credentials: 'include' // inclui o cookie CSRF automaticamente
    })
      .then(res => res.json())
      .then(data => {
        setCsrfToken(data.csrfToken);
      });
  }, []);
```

```
  // Envia formulário com o token CSRF
  const enviar = async (e) => {
    e.preventDefault();

    const resposta = await fetch('http://localhost:3000/envio-dados', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      credentials: 'include', // necessário para enviar o cookie
      body: JSON.stringify({
        nome,
        mensagem,
        _csrf: csrfToken // obrigatório!
      })
    });

    const data = await resposta.json();
    setResposta(data);
  };

  return (
    <div style={{ padding: 20 }}>
      <h2>Formulário com proteção CSRF</h2>
      <form onSubmit={enviar}>
        <input
          type="text"
          placeholder="Seu nome"
          value={nome}
          onChange={e => setNome(e.target.value)}
        />
        <br />
        <textarea
          placeholder="Mensagem"
          value={mensagem}
          onChange={e => setMensagem(e.target.value)}
        />
        <br />
        <button type="submit">Enviar</button>
      </form>

      {resposta && (
        <pre style={{ marginTop: 20 }}>
          {JSON.stringify(resposta, null, 2)}
        </pre>
      )}
    </div>
  );
}
```

```
export default App;
```

Resultado

Se o token `_csrf` estiver ausente ou incorreto, a API retorna erro 403. Se estiver presente e válido, a requisição segue normalmente.

Um exemplo funcional de como proteger uma aplicação Express contra CSRF (Cross-Site Request Forgery) usando a biblioteca `csurf`. Ela garante que cada requisição POST (ou mutável) contenha um token CSRF válido, prevenindo requisições forjadas vindas de outros sites.

Passo a passo: Protegendo rota com CSRF

Instale as bibliotecas: npm install express cookie-parser csurf

```
// app.js
const express = require('express');
const cookieParser = require('cookie-parser');
const csrf = require('csurf');

const app = express();

// Middlewares
app.use(express.urlencoded({ extended: true }));
app.use(express.json());
app.use(cookieParser());

// Proteção CSRF via cookie
const csrfProtection = csrf({ cookie: true });
app.use(csrfProtection);

// Rota GET que fornece o token CSRF para o frontend
app.get('/form-token', (req, res) => {
  res.json({ csrfToken: req.csrfToken() });
});

// Rota protegida com token CSRF (ex: formulário simulando envio)
app.post('/comentario', (req, res) => {
  const { nome, mensagem } = req.body;
  res.json({
    mensagem: 'Comentário recebido com segurança!',
    nome,
    conteudo: mensagem
  });
});

app.listen(3000, () => {
  console.log('Servidor protegido com CSRF rodando na porta 3000');
});
```

Como funciona:

- O middleware `csrf` gera um token CSRF exclusivo por sessão.

- O token é entregue ao cliente (ex: via rota `/form-token`) e precisa ser enviado no corpo da requisição POST, no header ou como campo oculto de formulário.

- Se o token estiver ausente ou inválido, a requisição será automaticamente rejeitada com erro 403.

Exemplo de requisição POST simulada (com token):

```bash
curl -X POST http://localhost:3000/comentario \
  -H "Content-Type: application/json" \
  -H "Cookie: _csrf=SEU_COOKIE_AQUI" \
  -d '{"nome":"João", "mensagem":"Olá mundo!", "_csrf":"SEU_TOKEN_AQUI"}'
```