



Curso: Sistema para internet
Disciplina: Padrão de projetos
Professora: Angelo Dias
Turma: P3

Grupo de autenticação de acesso ao sistema para gestão de empresas.
(Padrão de projeto proxy de autenticação)

Edson Henrique
João Gabriel
Juan Victor
Lucas Gusmão
Samuel Vinícius
Vitor Aranha

Proxy de autenticação

Um proxy de autenticação é como um "guardião" que verifica se você tem permissão para acessar um determinado sistema ou recurso online. Ele atua como um intermediário entre você e o sistema, garantindo que sua identidade seja verificada corretamente antes de permitir o acesso. É como mostrar um documento de identificação para entrar em um lugar - o proxy verifica se você é quem diz ser antes de conceder acesso. Isso ajuda a proteger o sistema contra acessos não autorizados e garante que apenas usuários autenticados possam utilizá-lo. O proxy de autenticação é importante para manter a segurança online, evitando que pessoas não autorizadas tenham acesso a informações ou recursos restritos.

// Objeto real

```
const sistema = {  
  acessarRecursos: function(usuario) {  
    console.log(`Acesso concedido para ${usuario}`);  
  }  
};
```

// Proxy de autenticação

```
const proxy = {  
  acessoPermitido: ['usuario1', 'usuario2'],  
  acessarRecursos: function(usuario) {  
    if (this.acessoPermitido.includes(usuario)) {  
      sistema.acessarRecursos(usuario);  
    } else {  
      console.log(`Acesso negado para ${usuario}`);  
    }  
  }  
};
```

// Utilizando o proxy

```
proxy.acessarRecursos('usuario1'); // Acesso concedido para usuário 1  
proxy.acessarRecursos('usuario3'); // Acesso negado para usuário 3
```

Algumas vantagens do padrão de projeto proxy:

- **Segurança aprimorada:** o Proxy de autenticação valida as credenciais do usuário e evita que usuários não autenticados ou não autorizados acessem recursos sensíveis.
- **Controle de acesso:** o Proxy de autenticação define regras e restrições para limitar quem pode acessar determinados recursos específicos.
- **Redução de carga:** o Proxy de autenticação usa cache para reduzir as consultas ao objeto real e melhorando o desempenho geral do sistema.
- **Escalabilidade:** o Proxy de autenticação permite adicionar novos recursos de autenticação sem afetar o objeto real. Útil em cenários que a autenticação precisa ser distribuída em várias camadas ou serviços.
- **Flexibilidade:** o Proxy de autenticação permite implementar lógica personalizada de autenticação sem modificar o objeto real. Permitindo usar mais formas de confirmar quem é o usuário, aumentando a segurança.

Visão geral

Em resumo, o padrão de projeto Proxy de autenticação oferece segurança aprimorada, controle de acesso, redução de carga, escalabilidade, flexibilidade e reutilização de código. Ele é útil em cenários em que é necessário controlar o acesso a recursos com base em autenticação e implementar lógica personalizada sem modificar o objeto real. O Proxy de Autenticação é um padrão de projeto que traz benefícios, mas também desafios (desvantagens). No entanto, esse padrão de projeto também traz alguns desafios, como complexidade de implementação, desempenho comprometido, ponto único de falha, necessidade de medidas de segurança adicionais e requisitos de configuração e gerenciamento. Esses desafios

devem ser considerados antes de escolher o proxy de autenticação para um sistema.

Algumas desvantagens do padrão de projeto proxy:

- **Complexidade de implementação:** requer a criação de uma interface comum e a implementação da lógica de autenticação e cache no objeto intermediário.
- **Desempenho comprometido:** adiciona uma camada extra de comunicação entre o cliente e o objeto real, o que pode afetar o tempo de resposta e a latência.
- **Ponto único de falha:** é responsável por gerenciar o acesso ao objeto real, o que significa que se ele falhar ou ficar indisponível, o cliente não poderá acessar o objeto real.
- **Necessidade de medidas de segurança adicionais:** armazena as credenciais do usuário e os dados do objeto real em cache, podendo torná-lo um alvo para ataques maliciosos. Sendo necessário proteger o objeto intermediário com criptografia, firewall entre outros.
- **Requisitos de configuração e gerenciamento:** requer configuração adequada para estabelecer conexão com o objeto real e definir regras e restrições de acesso. Faz-se necessário gerenciar o cache para garantir que os dados estejam atualizados e consistentes com o objeto real.

Apresentação do Código

Objetivo

Nosso código tem como objetivo principal implementar um sistema de gestão de empresas com funcionalidades de cadastro e autenticação de dois tipos de acesso: administrador e gestor.

Ao acessar a tela de administrador, o usuário terá permissão para visualizar e gerenciar solicitações de ordens de serviço, bem como cadastrar técnicos e equipamentos. Isso permite um controle mais eficiente e centralizado das atividades da empresa.

Por outro lado, na tela do gestor, o usuário terá acesso a informações detalhadas sobre os gastos e operações realizadas na empresa. Essa funcionalidade permite que o gestor acompanhe de perto o desempenho financeiro da empresa, identificando áreas que requerem melhorias e tomando decisões estratégicas embasadas em dados concretos.

Funcionalidades

Cadastro e autenticação de administrador e gestor.

Tela de administrador com acesso a solicitações de ordens de serviço, cadastro de técnicos e cadastro de equipamentos.

Tela do gestor com informações detalhadas sobre gastos e operações da empresa

Código (Proxy)

```
projeto_js > JS proxy.js > ...
1  const express = require('express');
2  const app = express();
3
4  // Middleware de autenticação
5  const authenticate = (req, res, next) => {
6    // Realize a autenticação conforme necessário
7    // Aqui você pode verificar as credenciais do usuário ou o token de acesso, etc.
8    // Se a autenticação falhar, retorne uma resposta de erro ou redirecione o usuário para uma página de login.
9
10   // Se a autenticação for bem-sucedida, chame next() para prosseguir com a solicitação
11   next();
12 };
13
14 // Rota inicial
15 app.get('/', (req, res) => {
16   res.sendFile(__dirname + '/index.html');
17 });
18
19 // Rota para processar o login
20 app.post('/api/login', (req, res) => {
21   // Lógica de autenticação
22   // ...
23
24   // Exemplo de resposta bem-sucedida
25   res.status(200).json({ isAdmin: true });
26 });
27
28 // Iniciar o servidor na porta desejada
29 app.listen(3000, () => {
30   console.log('Servidor em execução na porta 3000');
31 });
```

Este código utiliza o framework Express para criar um servidor web em Node.js.

Primeiro, ele importa o módulo `express` e cria uma instância do aplicativo Express usando `const app = express();`.

Em seguida, define um middleware de autenticação chamado `authenticate`. Um middleware é uma função que pode manipular a requisição antes de ser processada pela rota correspondente. Neste caso, o middleware de autenticação é usado para verificar as credenciais do usuário ou o token de acesso. Se a autenticação falhar, uma resposta de erro pode ser enviada ou o usuário pode ser redirecionado para uma página de login. Se a autenticação for bem-sucedida, a função `next()` é chamada para prosseguir com a solicitação.

O código define uma rota inicial acessível através do método GET e o caminho `/`. Quando essa rota é acessada, o servidor envia o arquivo `index.html` localizado no diretório atual (usando `__dirname`) como resposta.

O código também define uma rota para processar o login. Essa rota é acessada através do método POST e o caminho `/api/login`. Quando essa

rota é acessada, é esperado que haja uma lógica de autenticação que verifica as credenciais do usuário. Neste exemplo, a resposta enviada é um objeto JSON com a propriedade `isAdmin` definida como `true` para demonstrar uma resposta bem-sucedida.

Por fim, o servidor é inicializado para escutar na porta 3000 usando `app.listen(3000, () => { ... });`. Quando o servidor é iniciado, a mensagem "Servidor em execução na porta 3000" é exibida no console.

Resumidamente, este código cria um servidor web básico usando o Express, define rotas para a página inicial e para processar o login, e usa um middleware de autenticação para verificar as credenciais do usuário antes de processar as solicitações.

Código (Validação do Login)

```
projeto_js > JS login.js > ...
1  function acessar() {
2      var lista_emails = ["adminvitor@gmail.com", "gestorsamuel@gmail.com"];
3      var lista_senhas = ["123", "456"];
4      var get_email = document.querySelector("input#email");
5      var get_senha = document.querySelector("input#senha");
6
7      // Verificar se o email e a senha estão na lista
8      var email = get_email.value.toLowerCase();
9      var senha = get_senha.value.toLowerCase();
10     var index = lista_emails.indexOf(email);
11     if (index !== -1 && lista_senhas[index] === senha) {
12         // Sucesso na autenticação, redirecionar o usuário para a página adequada
13         if (email.includes("admin")) {
14             window.location.href = "/admin/index.html";
15         } else if (email.includes("gestor")) {
16             window.location.href = "/gestor/index.html";
17         }
18     } else {
19         // Erro na autenticação
20         window.alert("ERRO! Verifique se seu email e/ou senha estão corretos");
21     }
22 }
```

Este código é uma função chamada `acessar()`, que é executada quando um determinado evento ocorre (no caso, ao clicar em um botão).

A função começa definindo duas variáveis: `lista_emails` e `lista_senhas`. Elas são arrays que armazenam os emails e senhas permitidos para autenticação.

Em seguida, a função obtém os elementos de entrada de email e senha do documento HTML usando `document.querySelector("input#email")` e `document.querySelector("input#senha")`, respectivamente.

O código verifica se o email e a senha fornecidos pelo usuário estão na lista permitida. Ele converte o email e a senha para letras minúsculas para

evitar problemas de comparação. Em seguida, ele usa o método `indexOf()` para encontrar o índice do email na lista de emails permitidos. Se o email estiver na lista e a senha correspondente também estiver correta, a autenticação é considerada bem-sucedida.

Se a autenticação for bem-sucedida, o código verifica o tipo de usuário com base no email fornecido. Se o email incluir a palavra "admin", o usuário é redirecionado para a página `/admin/index.html`. Se o email incluir a palavra "gestor", o usuário é redirecionado para a página `/gestor/index.html`. O redirecionamento é realizado usando `window.location.href`.

Se a autenticação falhar, ou seja, o email e/ou senha fornecidos não estiverem na lista permitida, um alerta é exibido na janela do navegador com a mensagem "ERRO! Verifique se seu email e/ou senha estão corretos".

Resumidamente, essa função é responsável por verificar as credenciais de login fornecidas pelo usuário em relação a uma lista predefinida de emails e senhas permitidos. Se a autenticação for bem-sucedida, o usuário é redirecionado para a página apropriada com base no tipo de usuário identificado no email. Caso contrário, é exibido um alerta de erro.