Middleware de Autenticação

Introdução

Nesta aula, adentraremos um território fundamental na salvaguarda da integridade e segurança das aplicações digitais. Aqui, mergulharemos no estudo dos mecanismos de autenticação, um pilar essencial na construção de softwares confiáveis e seguros. Nossa exploração cobrirá os diversos métodos de autenticação, desde senhas e tokens até soluções mais avançadas como autenticação biométrica e chaves públicas e privadas. O foco estará no entendimento de como estes métodos asseguram a identidade dos usuários e sistemas, permitindo o acesso apenas a entidades verificadas, enquanto protegem contra acessos não autorizados e potenciais vulnerabilidades. Essa introdução serve como base para a implementação prática de um middleware de autenticação no projeto da disciplina, combinando teoria e aplicação para um aprendizado integral sobre a importância crítica da autenticação no mundo da tecnologia.

Conceito dos Tipos de Autenticação

Iniciando nossa jornada, mergulharemos em um dos aspectos mais vitais da segurança e da arquitetura de software moderna. Desvendaremos os diferentes tipos de autenticação, uma ferramenta essencial na proteção de dados e na validação de identidades em sistemas digitais. Esta parte da aula visa esclarecer o vasto leque de opções disponíveis para autenticação, destacando sua importância não apenas para a segurança, mas também para a experiência do usuário e a integração de sistemas.

Tipos de Autenticação:

1. **Senha e PIN (Personal Identification Number):** os métodos tradicionais de autenticação, onde uma sequência de caracteres ou números é utilizada para verificar a identidade do usuário.

2. **Autenticação de Dois Fatores (2FA):** adiciona uma camada de segurança requerendo, além da senha, um código temporário enviado via SMS ou gerado por aplicativos dedicados.

3. **Chaves Públicas e Privadas:** um sistema criptográfico que utiliza um par de chaves para autenticação, onde a chave privada é mantida em segredo pelo usuário e a chave pública é compartilhada.

4. **Biométrica:** Utiliza características físicas ou comportamentais únicas dos indivíduos, como impressão digital, reconhecimento facial ou de voz, para autenticar o usuário.

5. **Identidade Digital e Tokens:** emprega identificadores digitais ou tokens que podem ser verificados eletronicamente para provar a identidade ou o direito de acesso de um usuário.

Cada método apresenta características distintas, adaptando-se a diferentes necessidades de segurança e contextos de uso. Por exemplo, a autenticação biométrica oferece conveniência e alta segurança ao garantir que o acesso seja concedido exclusivamente ao usuário legítimo, enquanto a autenticação baseada em chaves é ideal para proteger a transmissão de dados sensíveis.

A autenticação, no cerne da discussão, serve como um filtro que assegura a veracidade da identidade de usuários e sistemas, permitindo ou negando acesso com base em credenciais verificáveis. Exploraremos os conceitos e a aplicação de vários métodos, incluindo tokens, senhas, PINs (Personal Identification Numbers), autenticação baseada em chaves públicas e privadas, SMS, biométrica e identidade digital. Cada um destes métodos

possui características únicas, adaptando-se a diferentes cenários e exigências de segurança.

Por exemplo, os tokens são frequentemente utilizados em processos de autenticação de dois fatores, oferecendo uma camada adicional de segurança ao exigir que o usuário forneça, além da senha, um código gerado temporariamente enviado por SMS ou por aplicativos dedicados. Já a autenticação baseada em chaves públicas e privadas é um método sofisticado que utiliza um par de chaves para criptografar e descriptografar informações, garantindo a segurança na transmissão de dados sensíveis.

Os métodos de autenticação biométrica, como o reconhecimento facial ou de impressões digitais, são cada vez mais comuns em dispositivos móveis e sistemas de segurança, proporcionando conveniência e uma forte garantia de que o acesso é concedido exclusivamente ao usuário legítimo.

Ao final desta aula, espera-se que os alunos compreendam a diversidade de técnicas de autenticação disponíveis, reconhecendo sua relevância para a construção de sistemas seguros e confiáveis. Será destacada a importância de escolher o método de autenticação mais adequado para cada aplicação, equilibrando segurança, usabilidade e performance.

Ao compreender os tipos de autenticação, é fundamental também explorar como esses métodos se integram aos modelos de negócios. A escolha de um método de autenticação adequado impacta diretamente na segurança, na experiência do usuário e na eficiência operacional de uma empresa ou sistema.

Por exemplo:

- **Negócios Digitais:** Em plataformas de e-commerce ou serviços financeiros, a autenticação robusta, como a biométrica ou 2FA, é crucial para proteger as transações e os dados dos clientes.

- **Empresas de Tecnologia:** Para desenvolvedores de software e empresas de tecnologia, a autenticação baseada em chaves é essencial para assegurar a comunicação segura entre diferentes sistemas e serviços.

- **Dispositivos Móveis:** Fabricantes de smartphones e tablets frequentemente integram autenticação biométrica para facilitar o acesso seguro e rápido aos dispositivos.

Este módulo visa aprofundar o conhecimento teórico sobre autenticação e aplicá-lo em cenários práticos, incentivando os alunos a experimentar e testar diferentes métodos de autenticação, como parte integrante do processo de aprendizagem em desenvolvimento de software. A aplicação prática, por meio de ferramentas como o Postman, é fundamental para solidificar a compreensão dos conceitos e preparar os alunos para enfrentar desafios reais na segurança digital e integração de sistemas.

Assim, este módulo sobre Middleware de Autenticação é projetado para ser um marco no aprendizado dos alunos, alinhando teoria e prática para equipá-los com as habilidades necessárias para implementar soluções de autenticação eficazes em seus projetos tecnológicos.

Definição de JWT e quando usar

Nesta sessão, exploraremos o JSON Web Token (JWT), um padrão crucial na internet para a geração de dados com assinatura e/ou criptografia. Representado por um código em formato JSON, ele inclui um conjunto de chave-valor e passa por criptografia, garantindo a segurança dos dados. O JWT é fundamental na autenticação de usuários e na autorização de acessos em aplicações web.

O JWT se destaca por poder ser assinado com um par de chaves pública e privada. A chave pública pode ser distribuída para validar o token, enquanto a chave privada, mantida em segredo, é usada para assinar o token. Esta

abordagem assegura que somente entidades autorizadas possam emitir ou verificar os tokens, reforçando a segurança na troca de informações.

Estrutura do JWT:

- Header: Define o tipo do token e o algoritmo de criptografia.
- Payload: Contém informações importantes, como os dados do usuário ou do sistema.
- Signature: Verifica a autenticidade do token.

O JWT tem um período de validade limitado, prevenindo seu uso indevido após certo tempo.

Exemplos Práticos de Uso do JWT:

1. Autenticação em Aplicações Web:

- Num sistema de login, após a verificação das credenciais do usuário (e-mail e senha), um JWT é gerado e retornado ao usuário. Esse token é usado nas solicitações subsequentes para acessar recursos protegidos.

```
const jwt = require('jsonwebtoken');

const user = { id: 1, username: 'usuarioTeste' };

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

console.log(token);
```

2. Comunicação Entre Serviços:

- Em arquiteturas de microserviços, o JWT facilita a comunicação segura entre serviços distintos, onde cada serviço verifica o token para autenticar as solicitações.

3. APIs Restringidas:

- Para APIs que oferecem dados sensíveis ou funcionalidades específicas, o JWT pode ser usado para assegurar que somente usuários com tokens válidos tenham acesso.

Para exemplificar, integramos o JWT em um projeto Node.js. Imagine um cenário de aplicação onde necessitamos autenticar usuários. Após a validação das credenciais do usuário (por exemplo, e-mail e senha contra um banco de dados), um token JWT é gerado usando uma chave secreta. Veja:

```
const jwt = require('jsonwebtoken');

// Usuário autenticado com sucesso

const user = { id: 1, username: 'usuarioTeste' };

// Gerando um token JWT

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

console.log(token);
```

Esse token é então enviado de volta ao usuário e deve ser incluído nos cabeçalhos das solicitações subsequentes para acessar rotas protegidas.

Essa base será fundamental para o desenvolvimento de soluções tecnológicas seguras e eficientes, preparando os alunos para enfrentar os desafios de segurança

e autenticação no mundo digital. A seguir, avançaremos ainda mais na aplicação dos conceitos de autenticação, consolidando o conhecimento adquirido e explorando novas possibilidades.

Entendendo um Middleware de Autenticação

Agora, abordaremos a prática essencial de entender um middleware de autenticação. Após explorarmos as bases do middleware e os diferentes tipos de autenticação, incluindo o JWT (JSON Web Token), agora focaremos no papel específico do middleware de autenticação. Esse componente crítico visa validar a identidade de usuários para garantir o acesso seguro a sistemas e proteger contra vulnerabilidades. Desvendaremos como um simples token pode exercer um papel tão crucial na segurança das aplicações.

Figurando um processo de autenticação em um cenário com cliente, servidor e middleware operando conjuntamente. Esse middleware atua intermediando solicitações de entrada (GET) e respostas (response), utilizando informações como usuário, senha e token para verificar a autenticidade de quem solicita o acesso.

Na prática, quando uma solicitação de acesso é feita, o middleware de autenticação analisa os dados fornecidos. Se a verificação for bem-sucedida, o servidor responde com um status HTTP 200, indicando sucesso e concedendo acesso. Essa validação acompanha o usuário por toda a interação com o sistema, com mecanismos de segurança adicionais, como a expiração de tokens, para garantir que a sessão permaneça segura e atualizada.

Por outro lado, caso a autenticação falhe devido a dados incorretos ou tokens expirados, o servidor responderá com um erro HTTP 401 ou 403, indicando que a solicitação não possui autorização. Essa resposta exige que o usuário corrija os dados fornecidos ou, em alguns sistemas, realize um novo login para atualizar a sessão.

Esse mecanismo de autenticação é fundamental para proteger as aplicações contra acessos não autorizados e invasões, limitando a exposição a possíveis

vulnerabilidades de segurança. Ao compreender a estrutura básica de um middleware de autenticação, os alunos estarão melhor preparados para implementar essa camada de segurança em seus próprios projetos.

No decorrer, aprofundaremos na integração do middleware de autenticação em nossa aplicação de projeto, preparando o terreno para as próximas aulas, onde traremos essa estrutura para o contexto real do desenvolvimento de software. Com isso, espera-se que os alunos adquiram uma sólida compreensão do funcionamento e da importância da autenticação como uma medida de segurança indispensável nas aplicações modernas.

Adição do Middleware de Autenticação na aplicação

Nesta etapa crucial, vamos finalmente colocar em prática a adição do middleware de autenticação em nosso projeto. Esta parte da aula, tem como objetivo integrar o middleware de autenticação à nossa aplicação, essencial para garantir que apenas usuários autenticados tenham acesso. Esta etapa é fundamental para manter a segurança e a integridade da aplicação, usando tokens válidos para validar cada acesso.

Vamos iniciar abrindo o código no Visual Studio Code, certificando-nos de que todas as dependências necessárias estejam corretamente instaladas. Se necessário, realize o comando `npm install` para atualizar e instalar as dependências. Para a autenticação com JWT (JSON Web Token), introduziremos um novo módulo ao Node, que trabalhará em conjunto com o Bcrypt para a encriptação de senhas. Este módulo é essencial para a criação e validação de tokens seguros.

Neste momento, a nossa tarefa consiste em verificar as dependências atuais do projeto, incluindo o Bcrypt, o Express e o Mongoose, e introduzir a dependência do JWT. Após a instalação do JWT, implementaremos o middleware de autenticação, criando um arquivo específico para isso. Esse arquivo, nomeado como `authMiddleware.js`, será responsável por importar o JWT e definir a função de verificação de tokens.

A verificação do token será feita através de uma função chamada `VerifyToken`, que checará a existência e a validade do token em cada requisição ao servidor. Caso o token seja inválido ou não esteja presente, a função retornará um erro HTTP 401, indicando que o acesso é negado.

Após definir a função de verificação, prosseguiremos com a implementação do middleware no projeto. Inicialmente, adicionaremos a dependência do JWT e ajustaremos o `authMiddleware.js` para incluir a lógica de verificação do token. Em seguida, integraremos esse middleware ao nosso projeto, aplicando-o nas rotas necessárias para garantir que apenas solicitações autenticadas sejam processadas.

Essa integração requer a adição do middleware de autenticação em pontos estratégicos do código, assegurando que a autenticação seja realizada eficazmente. A implementação detalhada do middleware nos permitirá validar os tokens de autenticação, garantindo que apenas usuários com credenciais válidas acessem as funcionalidades protegidas da aplicação.

Ao finalizar essa integração, espera-se que os alunos compreendam como integrar o middleware de autenticação em suas próprias aplicações, utilizando JWT para a segurança e integridade dos acessos. Essa habilidade é essencial para o desenvolvimento de aplicações web modernas e seguras, representando um componente vital na proteção de dados e na gestão de acessos em sistemas complexos. Nosso próximo passo será aplicar esses conceitos, preparando nossas ferramentas, como o Postman, para testar e validar a autenticação implementada.

Validação e uso da autenticação na aplicação

Neste segmento final da nossa aula, realizaremos a crucial etapa de validação e uso da autenticação em nossa aplicação. Exploraremos a prática implementada anteriormente, enfatizando o processo de verificação de identidade dos usuários que acessam o sistema, seja através de senha, token, PIN, SMS, ou qualquer outro método de autenticação escolhido.

Concentraremos nossos esforços no middleware de autenticação adicionado ao projeto, empregando o Postman para a validação das mensagens de autenticação. Um ajuste necessário, previamente esquecido, envolve a importação do JWT como uma dependência no mapeamento da rota, essencial para a integração do "secret key" no processo de autenticação.

Após garantir a inclusão do import do JWT, prosseguiremos com a execução do projeto através do VSCode, utilizando o Postman para criar novas requisições de login. É vital lembrar as credenciais de um médico previamente cadastrado no banco de dados, pois as senhas agora estão protegidas por hash, tornando-se inacessíveis diretamente.

Criaremos uma nova requisição de login no Postman, especificando o mapeamento POST correto e fornecendo o corpo da requisição em formato JSON, contendo as informações de login e senha. Após enviar uma requisição vazia ou com credenciais inválidas, esperamos receber mensagens indicando falha na autenticação, demonstrando que o sistema de autenticação está funcionando corretamente.

Ao submeter credenciais válidas, o sistema deve retornar um token de segurança, que será essencial para acessar outras partes da aplicação. Esse token deve ser inserido no cabeçalho de autorização das subsequentes requisições no Postman, assegurando que apenas solicitações autenticadas sejam processadas. A implementação correta resultará na criação de novos registros na aplicação, como a adição de médicos e pacientes, somente após a validação bem-sucedida do token de autenticação.

Para validar o token nas solicitações subsequentes, usamos um middleware que verifica o token usando a mesma chave secreta.

```
const jwt = require('jsonwebtoken');

const verificarToken = (req, res, next) => {
```

```
  const token = req.headers['authorization'];

   if (!token) return res.status(401).send('Acesso negado. Nenhum token fornecido.');

  try {

    const verificado = jwt.verify(token, 'chave_secreta');

    req.usuario = verificado;

       next(); // prosseguir para a próxima função no pipeline

  } catch (erro) {

    res.status(400).send('Token inválido.');

  }

}
```

Esse processo de validação e uso da autenticação na aplicação não apenas fortalece a segurança, mas também integra uma camada adicional de proteção, garantindo que somente usuários autenticados possam interagir com o sistema. Ao final deste módulo, espera-se que os alunos tenham uma compreensão prática de como implementar e validar a autenticação em aplicações web, utilizando middleware de autenticação para reforçar a segurança e integridade dos acessos.

Este encerramento da nossa série sobre middleware de autenticação não marca o fim da jornada de aprendizado, mas prepara o terreno para futuras explorações no desenvolvimento front-end, testes e gestão de arquivos, enriquecendo ainda mais nosso projeto para um portfólio robusto e dinâmico.

GitHub da Disciplina

Você pode acessar o repositório da disciplina no GitHub a partir do seguinte link: https://github.com/FaculdadeDescomplica/ProgramacaoII. Esse repositório tem como principal objetivo guardar os códigos das aulas práticas da disciplina para aprimorar suas habilidades em vários tópicos, incluindo a criação e consumo de APIs com controle de autenticação utilizando Node.js e utilizando boas práticas de programação e mercado.

Conteúdo Bônus

Para complementar o conhecimento adquirido em nossa aula, recomendo como conteúdo bônus a visualização do vídeo "Middleware de Autenticação" do canal Thi Code no YouTube. Esse vídeo é uma excelente oportunidade para aprofundar seu entendimento sobre como os middlewares funcionam na prática, especialmente no contexto de autenticação de usuários em aplicações web.

Referências Bibliográficas

Bibliografia Básica:

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados.** 7.ed. Pearson: 2018.

MEDEIROS, L. F. de. **Banco de dados:** princípios e prática. Intersaberes: 2013.

VICCI, C. (Org.). **Banco de dados**. Pearson: 2014.

Bibliografia Complementar:

CARDOSO, L. da C. **Design de aplicativos**. Intersaberes: 2022.

JOÃO, B. do N. **Usabilidade e interface homem-máquina.** Pearson: 2017.

LEAL, G. C. L. **Linguagem, programação e banco de dados:** guia prático de aprendizagem. Intersaberes: 2015.

PUGA, S.; FRANÇA, E.; GOYA, M. **Banco de dados:** implementação em SQL, PL/SQL e Oracle 11g. Pearson: 2013.

SETZER, V. W.; SILVA, F. S. C. **Bancos de dados.** Blucher: 2005.

Ir para exercício