Semana 7- aula 01
Linguagens de programação back-end
Aplicações práticas avançadas
Implementação de autenticação e autorização
Código da aula: [SIS]ANO2C2B1S7A1

Objetivos da Aula:

- ❖ Implementar sistemas de autenticação e autorização.
- ❖ Desenvolver a assertividade ao implementar sistemas de autenticação e autorização, garantindo a segurança dos usuários.
- ❖ Recurso audiovisual para exibição de vídeos e imagens; • Lápis e caderno para anotações

Exposição

Autenticação e autorização são dois processos distintos, porém interligados, cruciais para a segurança de sistemas e aplicações. Vamos definir cada um, destacar suas diferenças e ilustrar com exemplos.

Definição de Autenticação

Autenticação é o processo de verificar a identidade de um usuário, dispositivo ou processo. O objetivo é garantir que a entidade que está tentando acessar um sistema ou recurso é realmente quem ou o que alega ser. Em outras palavras, a autenticação responde à pergunta: "Quem é você?"

Os métodos de autenticação podem variar bastante, incluindo:

- Algo que você sabe: Senhas, PINs, perguntas de segurança.
- Algo que você tem: Cartões de acesso, tokens de segurança, chaves físicas, um dispositivo móvel.
- Algo que você é: Biometria (impressão digital, reconhecimento facial, escaneamento de retina).

Definição de Autorização

Autorização é o processo de determinar o que uma entidade autenticada tem permissão para fazer ou acessar dentro de um sistema ou recurso. Uma vez que a identidade do usuário é verificada (autenticação), a autorização define seus privilégios e restrições. Em outras palavras, a autorização responde à pergunta: "O que você tem permissão para fazer?"

As regras de autorização são definidas e aplicadas pelo sistema, geralmente baseadas em funções, políticas ou atributos associados ao usuário autenticado.

Diferenças entre Autenticação e Autorização

Característica	Autenticação	Autorização
Objetivo Principal	Verificar a identidade do usuário.	Determinar o que o usuário pode acessar/fazer.
Pergunta Chave	Quem é você?	O que você tem permissão para fazer?
Ocorrência	Geralmente ocorre uma vez por sessão.	Ocorre a cada tentativa de acesso a um recurso.
Baseada em	Credenciais fornecidas pelo usuário.	Regras e permissões definidas pelo sistema.
Visibilidade ao Usuário	O usuário geralmente participa ativamente.	O usuário geralmente não tem controle direto.
Dependência	É o primeiro passo para o controle de acesso.	Depende de uma autenticação bem-sucedida.

Exemplos

Cenário 1: Acesso a um prédio de escritórios

- Autenticação: Um funcionário passa um cartão de acesso (algo que ele tem) em um leitor na porta. O sistema verifica se o número do cartão corresponde a um funcionário registrado. Ou, ele pode digitar um código PIN (algo que ele sabe) em um teclado.
- Autorização: Depois de autenticado, o sistema verifica se aquele funcionário tem permissão para acessar aquele andar específico ou áreas restritas do prédio com base em sua função ou nível de acesso. Um visitante autenticado na recepção, por exemplo, pode ter autorização apenas para a área de espera.

Cenário 2: Acesso a um sistema bancário online

- Autenticação: Um cliente digita seu nome de usuário e senha (algo que ele sabe) na página de login do banco. O sistema verifica se essas credenciais

correspondem aos registros. Em seguida, pode solicitar um código de verificação enviado por SMS (algo que ele tem) como uma segunda camada de autenticação (autenticação de dois fatores).
- Autorização: Uma vez autenticado, o sistema determina o que o cliente pode fazer. Ele pode ter permissão para visualizar o saldo da conta, fazer transferências dentro de certos limites, pagar boletos, mas não para aprovar empréstimos ou acessar informações de outros clientes.

Cenário 3: Acesso a arquivos em um sistema operacional

- Autenticação: Ao ligar o computador, um usuário pode precisar digitar uma senha ou usar reconhecimento facial (algo que ele é) para fazer login no sistema operacional.
- Autorização: Depois de logado, o sistema operacional verifica as permissões associadas à conta do usuário para cada arquivo e pasta. Um usuário pode ter permissão para ler, escrever e executar arquivos em sua pasta pessoal, mas apenas ler arquivos compartilhados por outros usuários.

Em resumo, a autenticação garante quem está acessando, enquanto a autorização garante o que essa pessoa pode fazer após o acesso ser concedido. Ambos são pilares fundamentais para manter a segurança e a integridade de qualquer sistema ou aplicação.

Semana 7- aula 02
Linguagens de programação back-end
Aplicações práticas avançadas
Integração com API externas
Código da aula: [SIS]ANO2C2B1S7A2

Objetivos da Aula:

- ❖ Implementar sistemas de autenticação e autorização.
- ❖ Desenvolver a assertividade ao implementar sistemas de autenticação e autorização, garantindo a segurança dos usuários.
- ❖ Recurso audiovisual para exibição de vídeos e imagens; • Lápis e caderno para anotações

Exposição

Integração com APIs Externas e Implementação de Autenticação e Autorização

Integração com APIs Externas

Integração com APIs (Application Programming Interfaces) externas refere-se ao processo de conectar e interagir o seu sistema ou aplicação com serviços, dados ou funcionalidades fornecidas por outros sistemas através de suas APIs. Essas APIs atuam como "portas de entrada" padronizadas que permitem a troca de informações e a execução de ações entre diferentes softwares, mesmo que desenvolvidos por terceiros e executados em infraestruturas distintas.

Em essência, a integração com APIs externas permite que sua aplicação estenda suas capacidades, utilize serviços especializados e acesse dados relevantes sem a necessidade de construir toda a funcionalidade internamente.

Exemplos Práticos:

- E-commerce utilizando um gateway de pagamento: Uma loja virtual precisa processar pagamentos com cartão de crédito. Em vez de desenvolver seu próprio sistema de processamento, ela se integra com a API de um gateway de pagamento (como PayPal, Stripe ou PagSeguro). A API permite que a loja envie os detalhes da transação de forma segura para o gateway, que processa o pagamento e retorna o status (aprovado ou negado) para a loja.
- Aplicativo de previsão do tempo utilizando uma API meteorológica: Um aplicativo móvel que mostra a previsão do tempo não coleta dados meteorológicos diretamente. Ele se integra com a API de um serviço meteorológico (como OpenWeatherMap ou AccuWeather). O aplicativo envia a localização do usuário para a API, que responde com os dados de temperatura, umidade, vento, etc.
- Sistema de CRM integrando com uma API de mapas: Um sistema de Customer Relationship Management (CRM) pode se integrar com a API de um serviço de mapas (como Google Maps ou Mapbox) para exibir a localização de clientes, otimizar rotas de visitas de vendedores ou calcular distâncias entre contatos.
- Plataforma de mídia social utilizando uma API de compartilhamento: Um site pode se integrar com a API de plataformas de mídia social (como Twitter ou Facebook) para permitir que os usuários compartilhem conteúdo diretamente do site em seus perfis sociais.

Implementação de Autenticação e Autorização em Integrações com APIs Externas

Quando sua aplicação interage com APIs externas, é crucial implementar mecanismos robustos de autenticação e autorização para garantir a segurança da comunicação e dos dados envolvidos.

Autenticação no contexto de APIs Externas:

No contexto da integração com APIs externas, a autenticação visa verificar a identidade da sua aplicação (o "cliente" da API) para o serviço externo. Isso garante que apenas aplicações legítimas possam acessar a API.

Métodos comuns de autenticação em APIs Externas:

- Chaves de API (API Keys): Uma string secreta fornecida pelo provedor da API que sua aplicação inclui em cada requisição. Serve como uma identificação básica da sua aplicação.
- Tokens (OAuth 2.0, JWT): Um token de segurança temporário que sua aplicação obtém após um processo de autenticação inicial (que pode envolver credenciais da sua aplicação ou de um usuário final). Esse token é então incluído nas requisições subsequentes. OAuth 2.0 é frequentemente usado quando a aplicação precisa acessar recursos em nome de um usuário (sem ter acesso direto às credenciais do usuário). JWT (JSON Web Tokens) são tokens auto-contidos que carregam informações sobre a identidade e permissões.
- Autenticação Básica (Basic Auth): Envio de nome de usuário e senha codificados em Base64 no cabeçalho da requisição. Geralmente usado em cenários mais simples ou para APIs internas.
- Certificados TLS/SSL (Mutual TLS): Envolve a autenticação mútua entre o cliente e o servidor, onde ambos trocam e validam certificados digitais. Oferece um alto nível de segurança.

Autorização no contexto de APIs Externas:

Após a autenticação da sua aplicação pelo serviço externo, a autorização determina o que sua aplicação tem permissão para acessar ou fazer dentro da API. O provedor da API define as políticas de autorização.

Mecanismos comuns de autorização em APIs Externas:

- Escopos (Scopes) em OAuth 2.0: Definem as permissões específicas que um token de acesso concede à aplicação. Por exemplo, um token pode ter escopo para "ler dados do perfil" mas não para "publicar em nome do usuário".
- Funções (Roles) ou Permissões associadas à chave de API ou token: O provedor da API pode associar diferentes níveis de acesso ou funcionalidades a diferentes chaves de API ou tokens.
- Controles de acesso baseados em recursos (RBAC - Role-Based Access Control): A API pode ter um sistema interno que verifica se a aplicação autenticada (identificada pelo token ou chave) tem a função necessária para acessar um determinado recurso ou executar uma ação específica.
- Políticas de acesso (Access Policies): Regras mais complexas definidas pelo provedor da API que levam em consideração diversos fatores (como a origem

da requisição, o tipo de dado solicitado, etc.) para determinar se o acesso deve ser concedido.

Relação entre Integração com APIs Externas e Autenticação/Autorização

A autenticação e a autorização são componentes essenciais para uma integração segura e funcional com APIs externas. Sem esses mecanismos:

- Segurança seria comprometida: Qualquer pessoa ou aplicação maliciosa poderia potencialmente acessar dados sensíveis ou utilizar funcionalidades da API sem permissão.
- Uso e custos seriam difíceis de rastrear: O provedor da API não teria como identificar quem está utilizando seus serviços, dificultando o controle de uso e a cobrança (se aplicável).
- Integridade dos dados poderia ser afetada: Aplicações não autorizadas poderiam modificar ou corromper dados através da API.

Portanto, ao integrar sua aplicação com uma API externa, é fundamental compreender e implementar corretamente os métodos de autenticação e autorização exigidos pelo provedor da API. Isso garante a segurança da sua aplicação, a proteção dos dados e o uso adequado dos serviços externos.

Exemplos Práticos Combinados

Vamos revisitar os exemplos anteriores, focando agora nos aspectos de autenticação e autorização:

1. E-commerce e Gateway de Pagamento:
    - Autenticação: A loja virtual geralmente se autentica junto ao gateway de pagamento utilizando chaves de API fornecidas pelo gateway. Essas chaves identificam a loja como um cliente autorizado do serviço.
    - Autorização: A chave de API da loja pode ter permissões específicas, como "criar cobranças", "reembolsar pagamentos" ou "consultar transações". A loja só poderá realizar as ações para as quais sua chave tem autorização.
2. Aplicativo de Previsão do Tempo e API Meteorológica:
    - Autenticação: O aplicativo geralmente se autentica com a API meteorológica utilizando uma chave de API que é incluída em cada requisição. Essa chave permite que o provedor da API rastreie o uso e, em alguns casos, limite o número de requisições por dia/mês.
    - Autorização: A chave de API pode dar acesso a diferentes níveis de detalhe da previsão (por exemplo, dados básicos versus dados horários) ou a diferentes tipos de dados (temperatura, vento, poluição

do ar). O plano de assinatura do aplicativo pode determinar o nível de autorização concedido pela chave.

3. Sistema de CRM e API de Mapas:
    - Autenticação: O sistema de CRM se autentica com a API de mapas utilizando uma chave de API.
    - Autorização: A chave de API pode ter restrições quanto ao número de requisições de geocodificação (converter endereço em coordenadas) ou de cálculo de rotas por dia. Diferentes planos de assinatura podem oferecer diferentes limites de uso e, portanto, diferentes níveis de autorização.
4. Plataforma de Mídia Social e API de Compartilhamento:
    - Autenticação: Quando um usuário do site deseja compartilhar conteúdo em sua rede social, o site geralmente utiliza o protocolo OAuth 2.0 para autenticar o usuário junto à plataforma de mídia social e obter um token de acesso em nome desse usuário. O usuário precisa conceder permissão ao site para realizar a ação de compartilhamento.
    - Autorização: O token de acesso obtido via OAuth 2.0 terá escopos específicos definindo o que o site tem permissão para fazer em nome do usuário (por exemplo, "publicar na timeline", "enviar mensagens"). O site só poderá realizar as ações permitidas pelos escopos concedidos pelo usuário durante o processo de autenticação.

Em todos esses exemplos, a correta implementação de autenticação e autorização garante que apenas aplicações e usuários legítimos possam interagir com as APIs externas e que eles só possam realizar as ações para as quais têm permissão, protegendo assim os dados e a funcionalidade dos serviços envolvidos.

Semana 7- aula 03
Linguagens de programação back-end
Aplicações práticas avançadas
Implementação de WebSockets para comunicação em tempo real
Código da aula: [SIS]ANO2C2B1S7A3

Objetivos da Aula:

- Implementar sistemas de autenticação e autorização.
- Desenvolver a assertividade ao implementar sistemas de autenticação e autorização, garantindo a segurança dos usuários.

- ❖ Recurso audiovisual para exibição de vídeos e imagens; • Lápis e caderno para anotações

Exposição

Implementação de WebSockets para Comunicação em Tempo Real.

A implementação de WebSockets para comunicação em tempo real refere-se ao processo de utilizar a tecnologia WebSocket para estabelecer uma conexão persistente e bidirecional entre um cliente (geralmente um navegador web ou um aplicativo móvel) e um servidor. Essa conexão permite a troca instantânea de dados em ambos os sentidos, sem a necessidade de o cliente fazer requisições HTTP repetidas (polling) para obter novas informações.

Em essência, WebSockets proporcionam um canal de comunicação contínuo, ideal para aplicações que exigem atualizações de dados em tempo real, como chats, jogos online, painéis de controle com dados dinâmicos e notificações instantâneas.

Principais Características dos WebSockets:

- Conexão Persistente: Uma vez estabelecida, a conexão WebSocket permanece aberta até que uma das partes (cliente ou servidor) a feche explicitamente. Isso elimina a latência associada à abertura de uma nova conexão HTTP para cada troca de dados.
- Bidirecional: A comunicação ocorre em ambos os sentidos simultaneamente. O servidor pode enviar dados para o cliente assim que eles estiverem disponíveis, e o cliente também pode enviar dados para o servidor a qualquer momento, sem precisar esperar por uma requisição.
- Baseado em TCP: O WebSocket utiliza o protocolo TCP (Transmission Control Protocol) na camada de transporte, garantindo uma comunicação confiável e ordenada.
- Handshake HTTP Inicial: A conexão WebSocket começa com um "handshake" HTTP padrão. O cliente envia uma requisição HTTP especial de "upgrade" para o servidor, solicitando a mudança para o protocolo WebSocket. Se o servidor suportar WebSockets, ele responde com um código de status 101 (Switching Protocols), e a conexão é atualizada para WebSocket.
- Protocolo Próprio: Após o handshake, a comunicação segue um protocolo WebSocket específico (RFC 6455), que define o formato dos frames de dados trocados.
- Menor Overhead: Comparado ao polling ou outras técnicas de "simulação" de tempo real via HTTP (como Long Polling), o WebSocket geralmente possui um menor overhead de comunicação, pois os cabeçalhos são menores e a conexão é reutilizada.

Exemplos Práticos de Implementação de WebSockets:

1. Aplicações de Chat em Tempo Real:
    - Cenário: Um aplicativo de mensagens instantâneas onde os usuários enviam e recebem mensagens em tempo real.
    - Implementação com WebSockets: Quando um usuário envia uma mensagem, o cliente (navegador ou aplicativo) envia essa mensagem para o servidor através da conexão WebSocket estabelecida. O servidor, ao receber a mensagem, a distribui para os outros participantes da conversa, também através de suas respectivas conexões WebSocket. As mensagens aparecem instantaneamente para todos os usuários conectados, sem a necessidade de atualizar a página ou fazer novas requisições.
2. Jogos Online Multiplayer:
    - Cenário: Jogos que exigem interação em tempo real entre múltiplos jogadores, como jogos de estratégia, jogos de tiro ou jogos de tabuleiro online.
    - Implementação com WebSockets: As ações dos jogadores (movimento, ataques, interações com o ambiente) são transmitidas para o servidor através de WebSockets. O servidor processa essas ações e envia as atualizações do estado do jogo para todos os jogadores conectados em tempo real, garantindo uma experiência sincronizada.
3. Painéis de Controle e Dashboards com Dados Dinâmicos:
    - Cenário: Aplicações que exibem informações que são atualizadas continuamente, como monitoramento de sistemas, dados financeiros em tempo real, ou estatísticas de uso.
    - Implementação com WebSockets: O servidor, ao receber novas informações (por exemplo, métricas de um servidor, cotações da bolsa de valores), envia esses dados para os clientes conectados através de WebSockets. O painel de controle no navegador do usuário é atualizado instantaneamente, refletindo as mudanças sem a necessidade de recarregar a página.
4. Notificações em Tempo Real:
    - Cenário: Aplicações que precisam enviar notificações instantâneas aos usuários sobre eventos importantes (por exemplo, novas mensagens, atualizações de status, alertas).
    - Implementação com WebSockets: Quando um evento ocorre no servidor que requer uma notificação, o servidor envia essa notificação para os clientes relevantes através de suas conexões WebSocket. A

notificação pode ser exibida imediatamente ao usuário, sem que ele precise verificar manualmente por novas atualizações.

5. Ferramentas de Colaboração em Tempo Real:

    - Cenário: Aplicações onde múltiplos usuários precisam colaborar simultaneamente no mesmo documento ou projeto (por exemplo, editores de texto online, quadros brancos virtuais).
    - Implementação com WebSockets: As alterações feitas por um usuário são transmitidas para o servidor via WebSocket. O servidor, por sua vez, envia essas alterações para todos os outros usuários conectados ao mesmo documento em tempo real, permitindo que todos vejam as atualizações instantaneamente.

Em resumo, a implementação de WebSockets é fundamental para construir aplicações web e móveis que exigem comunicação instantânea e bidirecional entre clientes e servidores, proporcionando uma experiência de usuário mais fluida e responsiva em cenários de tempo real.