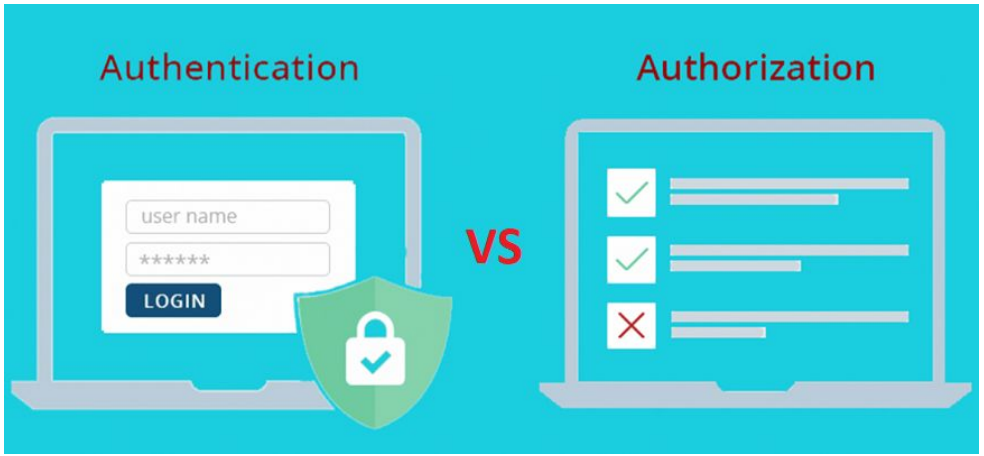
Funcionalidades avançadas e segurança em APIs RESTful

José Glauber UFCG 2024.1

Qual a diferença entre autenticação e autorização?



importância?

Autenticação

O objetivo geral é identificar se o usuário é quem diz ser.

- Administrador do sistema
- Usuário comum
- Professor em um sistema acadêmico

email? senha?

Autenticação

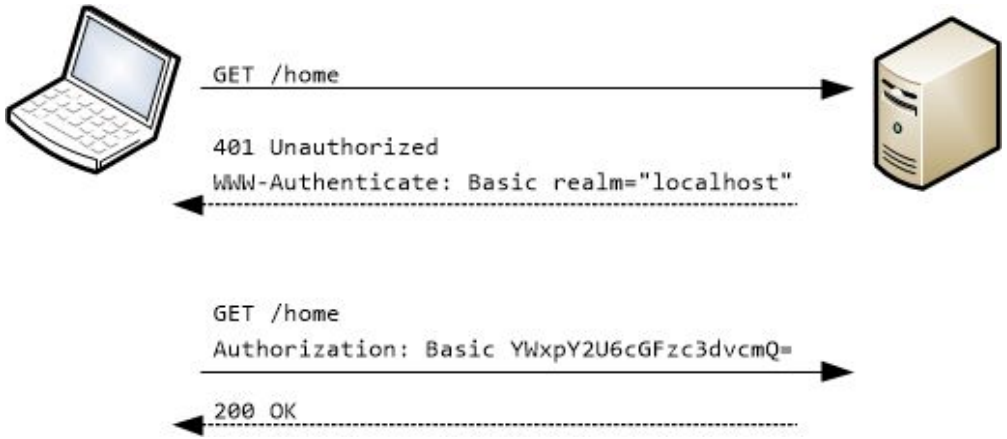
Alguns tipos de autenticação..

- Autenticação baseada em senhas
- Autenticação baseada em Multi-fator (MFA)
- Autenticação baseada em token
- OAuth 2.0

Autenticação baseada em senhas

Neste modelo a segurança depende do usuário ao definir nome de usuário e senha

ex: Basic Auth



cabeçalho de autorização com base64

Autenticação baseada em senhas

De forma geral é a autenticação mais básica e simples de ser implementada.

Quando utilizar?

Simplicidade…

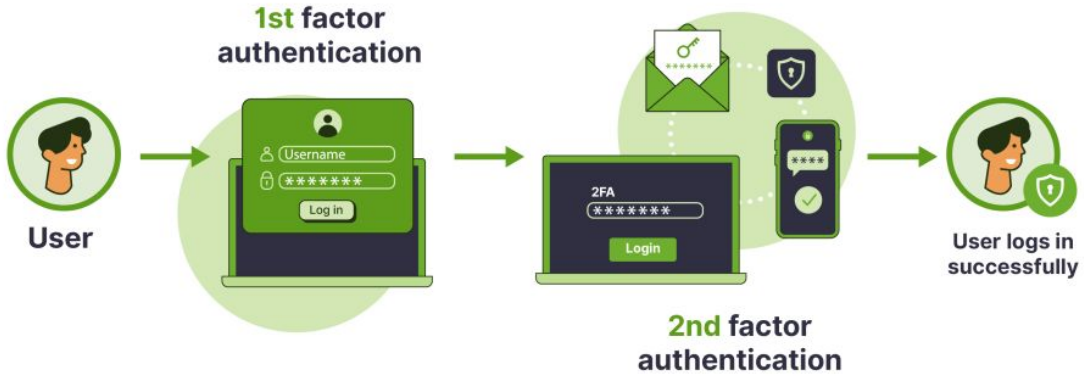
Está transportando dados sigilosos?

Problema principal:

Bastante suscetível a ataque de hackers

Autenticação de dois fatores (2FA)

Este modelo traz mais uma camada de segurança ao acesso das nossas aplicações. A 2FA dá aos negócios a capacidade de monitorar e ajudar a proteger suas informações e redes mais vulneráveis.



Autenticação de dois fatores (2FA) - Benefícios

- Não é necessário usar tokens físicos para autenticação de dois fatores (2FA), pois esses dispositivos podem ser perdidos. Hoje, há métodos 2FA mais práticos e convenientes.

- Geradores de senhas são mais seguros e eficientes que senhas tradicionais, já que cada senha gerada é única.

- Limitar o número de tentativas de senha ajuda a impedir que hackers acessem dados confidenciais.

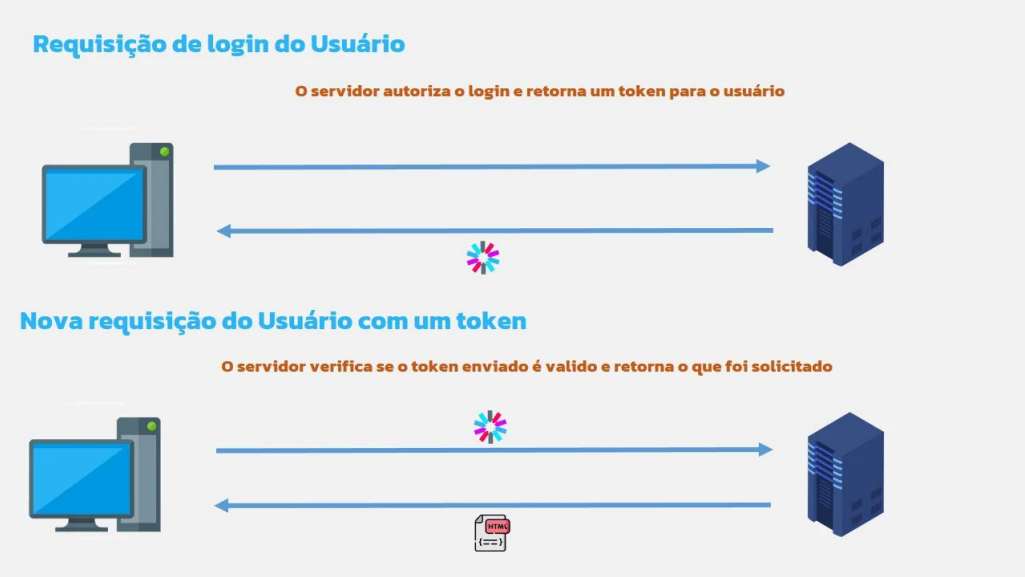
- O processo de segurança é fácil de gerenciar e usar.

Autenticação por sessão

Um dos primeiros métodos de autenticação. Após a requisição feita pelo cliente o servidor, por sua vez, cria uma sessão em sua memória ou banco e devolve a informação de usuário através de um cookie com o identificador da sessão criada.

Autenticação por token

Após ter login e senha validados pelo servidor, é criado um token e que permitirá o acesso à algum recurso. O padrão mais adotado é o JWT (JSON Web Token).



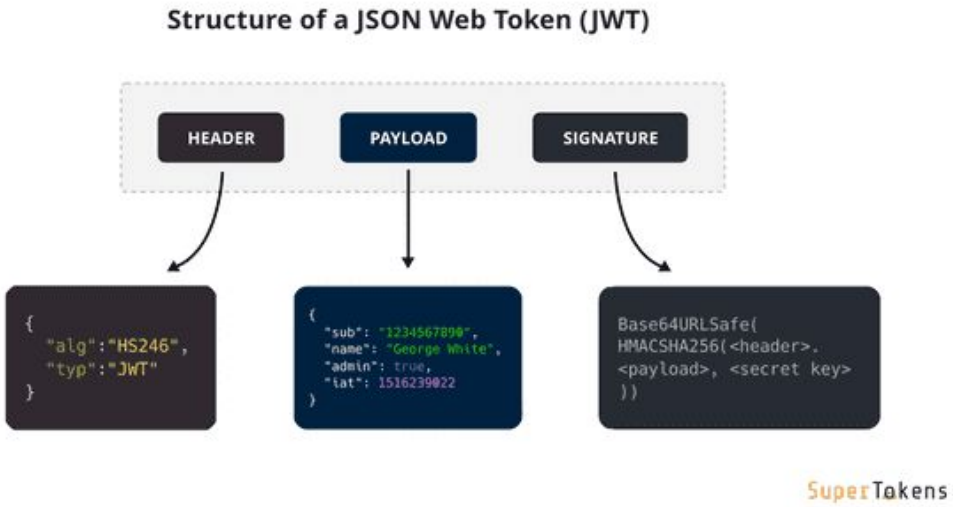
Autenticação por token

O JWT contém informações relevantes sobre o usuário, identificação, permissões e etc. Esse token é assinado digitalmente a fim de garantir a sua autenticidade e integridade.

O servidor não precisa armazenar nenhum estado relacionado a sessão do usuário. O token é incluído automaticamente em cada requisição subsequente ao servidor, o que faz com que não exista a necessidade de consultar um armazenamento de sessões.

Autenticação por token - JWT

É um padrão da indústria definido pela RFC7519 que tem como objetivo transmitir ou armazenar de forma compacta e segura objetos JSON entre diferentes aplicações.



Autenticação por token - JWT

Vantagens:

1. Independência do estado do servidor;
2. Tokens carrega consigo todas as informações referentes aos usuários;
3. Revogação eficiente de tokens;
4. Interoperabilidade, portabilidade e flexibilidade;

Desvantagens:

1. Necessidade de estratégias eficazes para revogar tokens;
2. Estratégias de sincronização e gerenciamento de tokens em diferentes dispositivos precisam ser cuidadosamente implementadas;
3. Prazo de validade para tokens;

Autenticação por sessão X token

Sessão: estado mantido pelo servidor, podendo ser armazenado em um bd ou na memória (Stateful).

- Limites de hardware;
- Muitas chamadas ao garbage collector;

Tokens: não são mantidas no servidor (Stateless). Token possui tudo que é importante.

- Id do usuário;
- Assinatura;
- Data de expiração;
- Método de autenticação utilizado.

Autorização

O objetivo geral é determinar permissões para acesso a determinados recursos, de usuários já autenticados.

Acessar configurações da conta

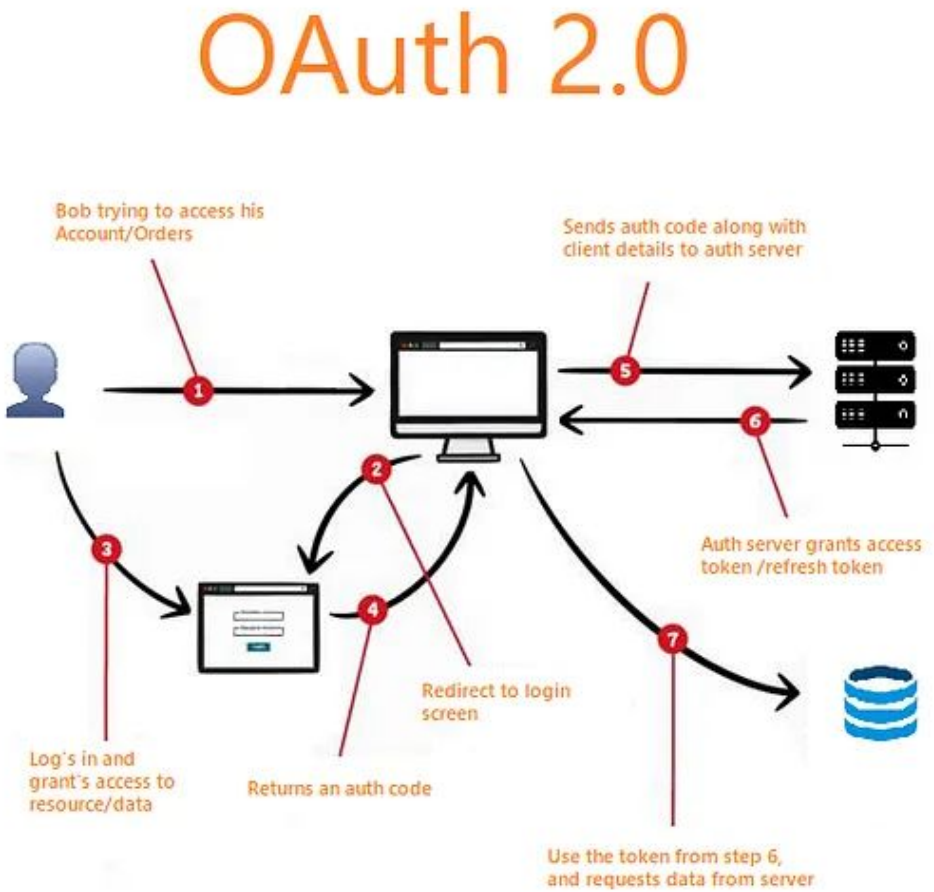
excluir e-mails

acessar lista de documentos dos usuários..

Autorização

OAuth 2.0 -> é um protocolo de autorização que permite que uma aplicação acesse recursos em nome de um usuário sem que o usuário precise fornecer suas credenciais diretamente à aplicação

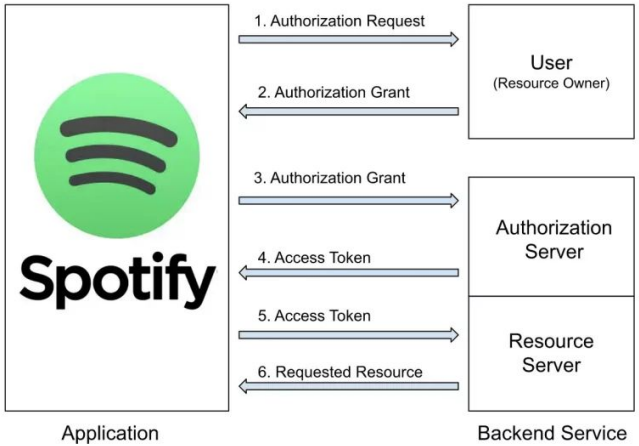
é um protocolo de autorização e NÃO um protocolo de autenticação



Autorização

Problema antes do OAuth -> Usuários tinham que compartilhar suas senhas com diferentes aplicativos para permitir o acesso aos seus dados.

- Risco de roubo de senha e violações de segurança;
- Comprometimento de todas as contas vinculadas àquela senha;



Quando ele é útil?

OAuth 2.0 é útil em situações em que os usuários desejam conceder acesso a seus recursos a aplicativos de terceiros, mas não desejam fornecer suas credenciais ou senhas.

é bastante útil para garantir que os usuários mantenham o controle sobre seus recursos e possam revogar o acesso a qualquer momento

Integração OAuth 2.0 com JWT

Quando OAuth 2.0 é usado com JWT, o servidor de autorização pode emitir um JWT como um token de acesso. Esse token pode então ser usado pelo cliente para acessar recursos no servidor de recursos.

como funciona?

1. **Autenticação do Usuário:** O usuário faz login no servidor de autorização (ex.: Google) e concede permissão ao cliente.
2. **Autorização:** O servidor de autorização emite um Access Token (um JWT) que autoriza o cliente a acessar os recursos do usuário.
3. **Acesso ao Recurso:** O cliente usa o JWT (Access Token) para acessar os recursos protegidos no servidor de recursos.
4. **Validação:** O servidor de recursos valida o JWT para verificar a autorização antes de conceder acesso.

Benefícios OAuth 2.0

- Segurança aprimorada
- Controle granular de permissões
- Experiência de usuário melhorada
- Suporte a diversos tipos de clientes
- Interoperabilidade

Benefícios JWT

- Autenticação e Autorização seguras
- Eficiência e desempenho
- Flexibilidade e escalabilidade
- Independência da plataforma e linguagem
- Suporte para expiração e renovação

OAuth 2.0 e JWT

	JWT	OAuth
Purpose	Token-based authentication mechanism for transmitting claims	Protocol for authorization and authentication in web and mobile apps
Centralized Server	None	Authorization server
Used For	Authentication and authorization	Authorization, not authentication
Applications	Single-page apps, mobile apps	Apps that rely on external APIs or services
Relationship	Directly between parties	Between third-party apps and resource owners
Authentication	Yes	No
Authorization	Yes	Yes

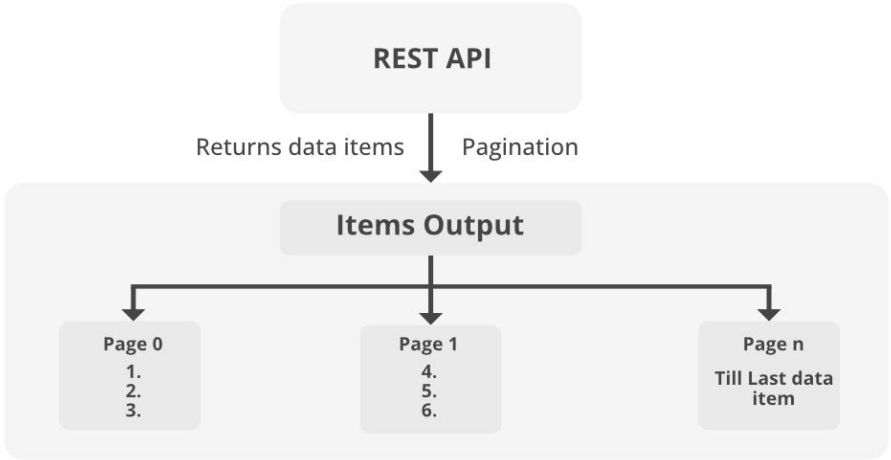
Paginação, filtros e ordenação

- Imagine uma API com muitos dados, retornando mais de 10k de registros em uma única requisição..

É a implementação desses recursos que faz sua API trabalhar de forma mais eficiente, trafegando apenas o necessário entre o cliente e servidor.

Paginação

- Endpoints que retornam muitos dados devem suportar a paginação como forma de obter uma melhor performance nas suas buscas.



Paginação

limit: identificar a quantidades de itens a ser obtido na consulta.

offset: identifica a página de dados da coleção a ser obtida.

```
GET

http://localhost:3000/api/products?
offset=1&limit=10
```

O corpo do mensagem de resposta deve obrigatoriamente retornar no JSON os campos:

- totalPages: total de páginas baseadas no campo size.
- currentPage: página atual dos dados retornados
- totalITems: total de registros encontrados.
- data: lista de registros encontrados.

Paginação

C: > Users > glaub > Documents > aulas-web > JS teste.js > app.get('/api/products') callback > products

```
app.get('/api/products', async (req, res) => {
    const { page = 1, limit = 10 } = req.query; // Valores padrão: página 1, 10 itens por página

    try {
        const pageNumber = parseInt(page);
        const limitNumber = parseInt(limit);

        const products = await Product.find()
            .limit(limitNumber) // Limite de itens por página
            .skip((pageNumber - 1) * limitNumber); // Pula os itens das páginas anteriores

        const total = await Product.countDocuments();

        res.json({
            totalPages: Math.ceil(total / limitNumber),
            currentPage: pageNumber,
            totalItems: total,
            data: products,
        });
    } catch (error) {
        res.status(500).json({ error: 'Erro ao buscar produtos' });
    }
});
```

Filtros e ordenação

Via parâmetros de query string, você consegue dar mais inteligência a sua API, para trabalhar com dados sob demanda.

```
HTTP GET
https://api.minhagastronomia/vinhos?pais=Brasil&estado=RS&de_preco=100&ate_preco=200
&status=disponivel
```

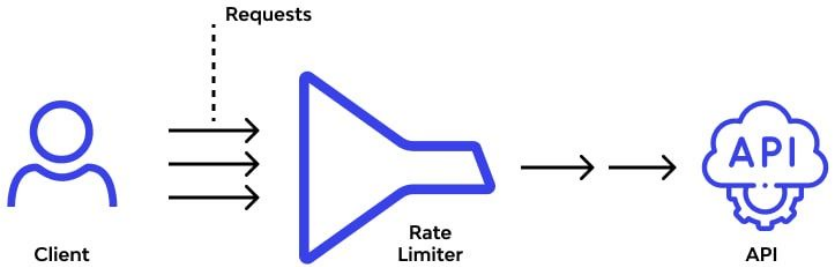
```
HTTP GET
https://api.minhagastronomia/vinhos?sort=pais:asc,estado
```

Boas práticas:

Tenha valores defaults para seus limites.

Não exagera em parâmetros de Query String.

Rate limiting

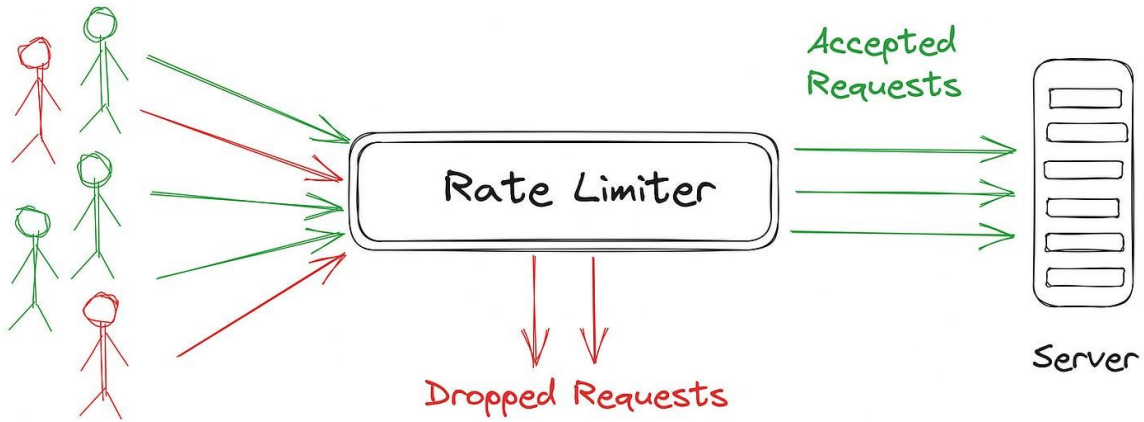


Técnica utilizada para controlar o número de solicitações que um usuário pode fazer a uma API em um determinado período de tempo.

Cenário real: Usuário redefinindo senha. A maioria dos sites/aplicativos usa limitação de taxa em seu processo de login e bloqueará se 3 ou 4 tentativas de login erradas forem feitas.

Rate limiting - Como funciona?

A ideia principal é baseada no rastreamento do endereço IP que fez as solicitações da API em um determinado intervalo de tempo. Quando está ativa, a solução controla o total de solicitações feitas e o tempo entre duas solicitações, feitas a partir de um único endereço IP. Se houver muitas solicitações e forem feitas com muita frequência, a ferramenta imporá limitações predefinidas ao referido endereço IP.



Rate limiting - Benefícios de implementar

- Evitar consumo excessivo de recursos
  - Ajuda a proteger a infraestrutura da API contra ataques de negação de serviço (DDoS), onde um grande número de requisições é enviado para sobrecarregar o sistema.
  - Permite uma melhor alocação de recursos, garantindo um acesso equitativo À API.

- Controle de fluxo inteligente
  - Uma maneira de gerenciar de forma eficiente a quantidade de requisições ou dados que um sistema processa em um determinado período de tempo.

Rate limiting

- Backend
  - É aplicado diretamente nas rotas da API;
  - Implementação mais simples e com controle total sobre lógica de rate limiting;
  - express-rate-limit

- API Gateway
  - O rate limiting é aplicado antes que as requisições alcancem o backend. O API Gateway ou o proxy monitora o tráfego, aplica as regras de rate limiting e bloqueia requisições que excedem os limites.
  - Útil quando existem muitos microsserviços;

Boas práticas

Autenticação segura

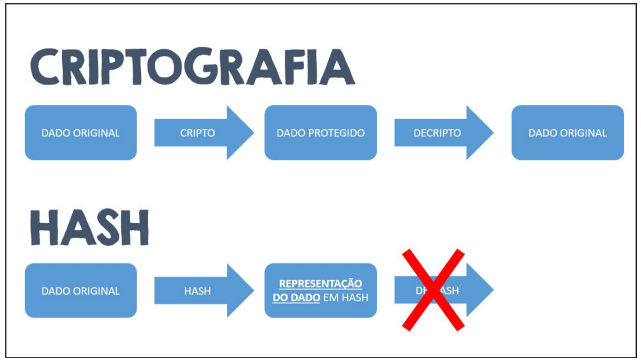
➢ Armazenamento seguro de senhas:

o jeito errado: armazenar senhas em texto simples.

- apenas comparação do que é inserido com o que está no banco de dados.

o jeito um pouco melhor: senhas criptografadas;

- precisa ser descriptografada sempre, isso necessita uma chave e precisa de certa segurança ao guardar essa chave de descriptografia.

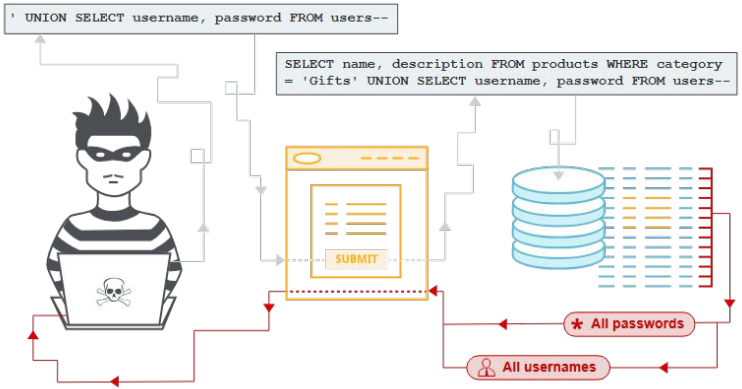


o jeito certo: armazenamento com hashes

- algoritmos criptográficos embaralham quaisquer dados em uma sequência de bits de comprimento fixo, de maneira irreversível.
  - o que é armazenado no bd é um hash, e não a senha em si. Por fim, na tentativa de login é comparado os hashes.

Proteção contra vulnerabilidades comuns

- SQL injection
  - ocorre quando é injetado comandos SQL em uma query executada por um bd. Isso pode permitir ao atacante manipular consultas, acessar dados sensíveis, modificar ou excluir dados e até mesmo assumir o controle total do banco de dados.




```sql
SELECT * FROM users WHERE username = 'user' AND password = 'pass';
```

```sql
SELECT * FROM users WHERE username = 'user' AND password = '' OR '1'='1' --';
```

Proteção contra vulnerabilidades comuns

- SQL injection - Como se prevenir?

Valide sempre os dados digitados pelo usuário

- Rejeite dados que são conhecidamente inválidos;
- Aceite somente dados que são conhecidamente válidos;

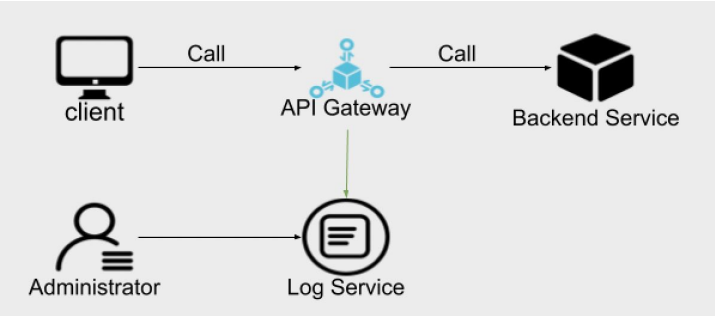
Crie usuários com permissões adequadas

Nunca retorne as mensagens do servidor SQL para o usuário

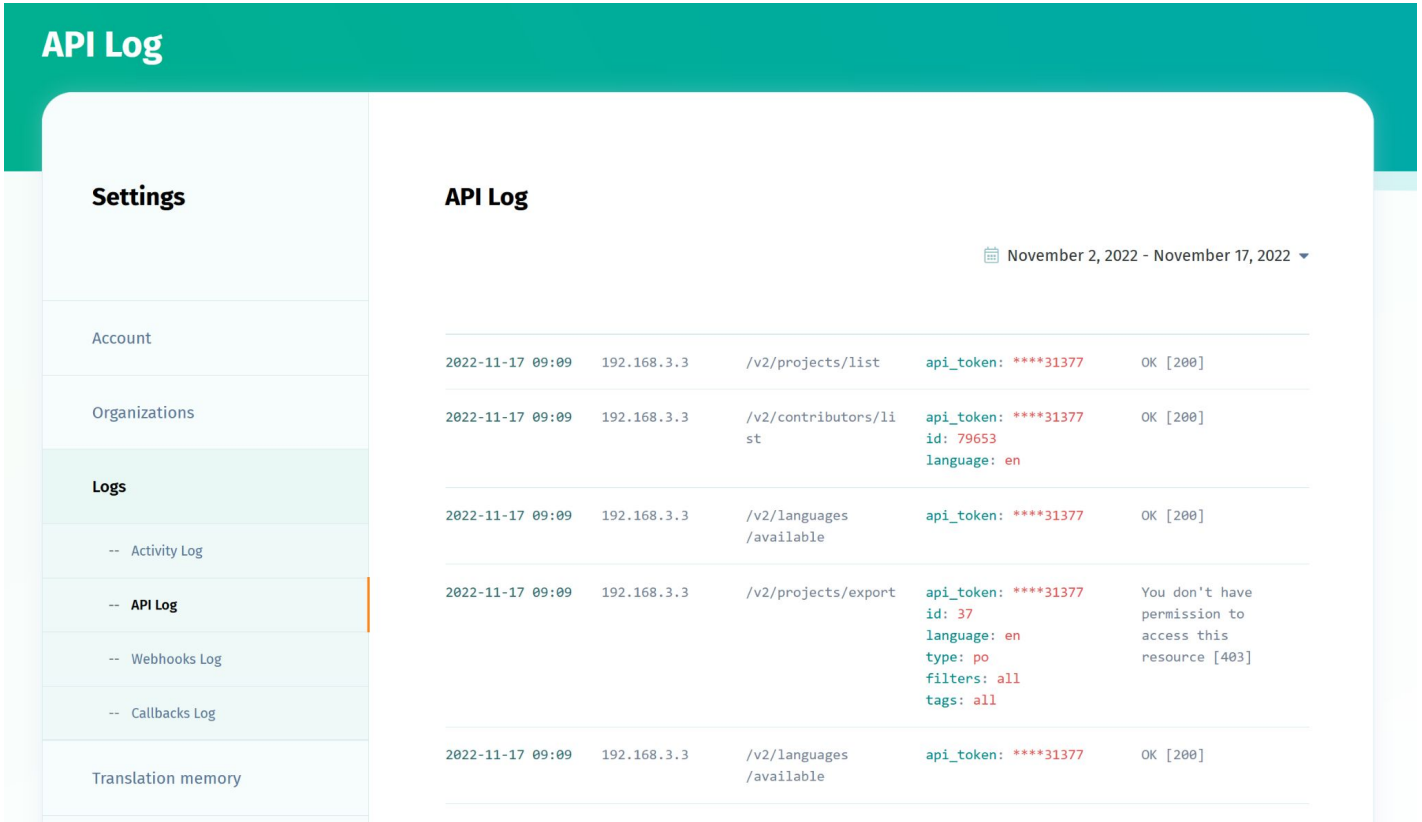
Habilite logs de segurança no servidor

Auditorias e logs

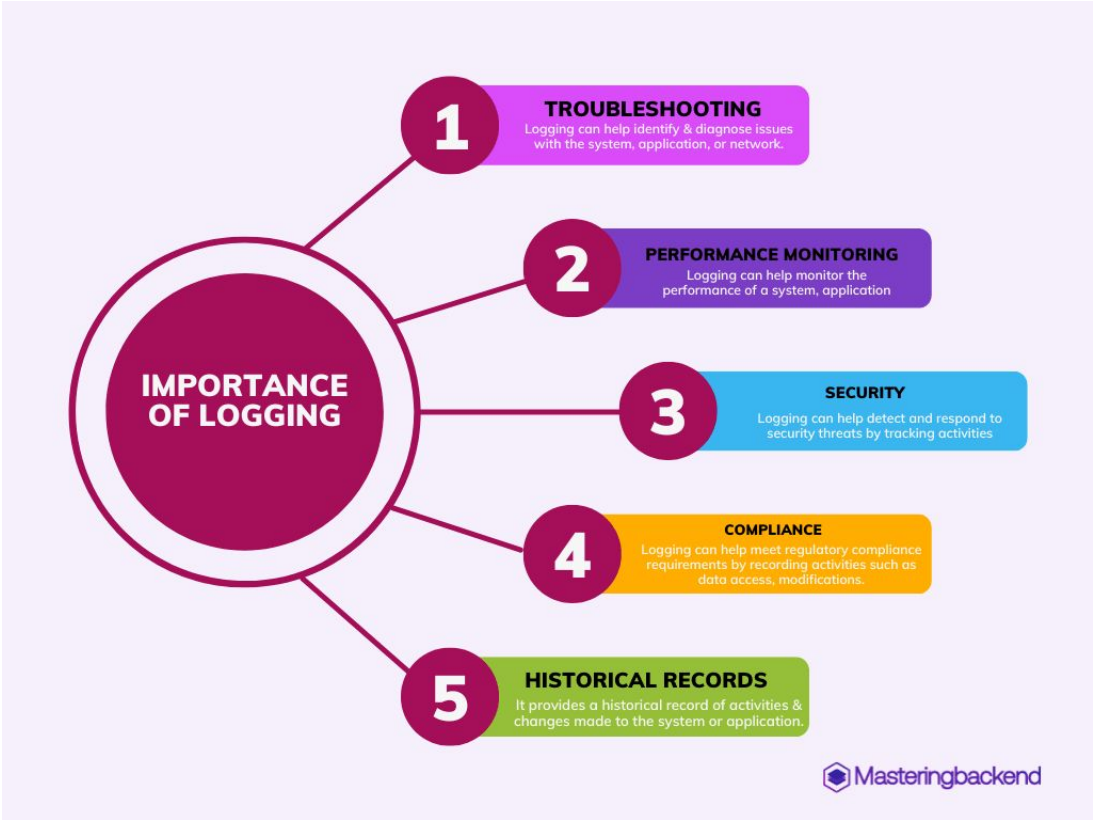
- Revisão sistemática de logs para garantir a conformidade com políticas e detectar atividades suspeitas. Capturam informações sobre eventos que ocorrem dentro de um sistema, como acessos, transações e operações realizadas pelos usuários e pelo próprio sistema.



Auditorias e logs



Auditorias e logs



Auditorias - monitoramento de acessos

- Observação e análise contínua de quem acessa o sistema, o que acessa, e como acessa. Este processo é crucial para garantir que apenas usuários autorizados tenham acesso aos recursos apropriados e para detectar acessos anômalos ou maliciosos.

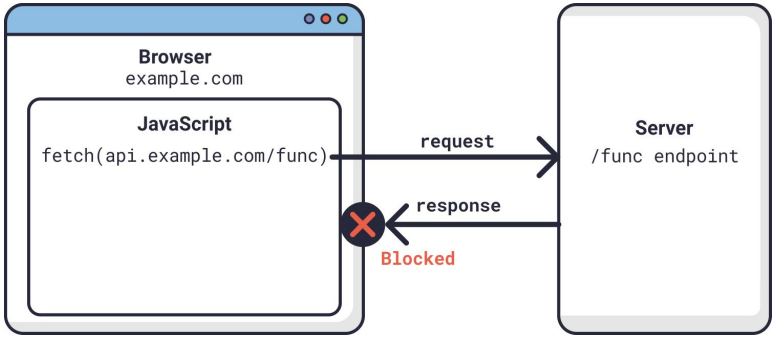
  - **Logs de acesso**
  - **Análise de comportamento através de ferramentas e dados**
  - **Alertas e notificações**

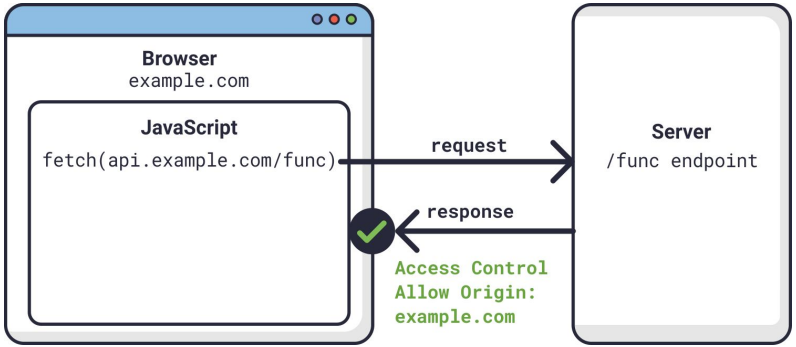
CORS - Cross Origin Resource Sharing

- CORS (Cross-Origin Resource Sharing) é um mecanismo de segurança que permite ou restringe recursos em um servidor da web a serem acessados a partir de origens diferentes da sua própria.

É usado para controlar quais sites podem fazer solicitações HTTP para um servidor em domínio diferente da página web.

CORS - Cross Origin Resource Sharing





Testes…

Prática fundamental no desenvolvimento de software. Garante qualidade e confiabilidade do produto.

Vamos focar em dois tipos específicos de testes:

Testes de unidade: verificam as unidades individuais do nosso código (métodos, classes..), garantindo que ela execute a tarefa desejada e produza os resultados esperados.

Validar lógica interna do código.

Aprender sintaxe de código de teste.

Testes…

Prática fundamental no desenvolvimento de software. Garante qualidade e confiabilidade do produto.

Vamos focar em dois tipos específicos de testes:

Testes de rota:

usados para verificar o comportamento das rotas em uma aplicação web ou API. Eles garantem que as rotas (ou endpoints) da aplicação respondam corretamente às solicitações e retornem os resultados esperados.

Validar retorno dos endpoints.

Aprender sintaxe de código de teste.

Prática

Form: https://forms.gle/KZ2uoTX9wwh1qa4F6

Referências

1. https://oauth.net/2/
2. https://jwt.io/introduction
3. https://openid.net/developers/how-connect-works/#:~:text=OAuth%202.0%2C%20the%20substrate%20for,structures%20when%20signatures%20are%20required.
4. Designing Web APIs: Building APIs That Developers Love - Designing Web APIs: Building APIs That Developers Love
5. RESTful Web APIs - Leonard Richardson, Mike Amundsen, Sam Ruby
6. O que é a 2FA (Autenticação de Dois Fatores)? | Segurança da Microsoft
7. O Que é API Rate Limiting: Entenda A Limitação De Requisições (lbodev.com.br)
8. What is Rate Limiting? Meaning and Definition (wallarm.com)