
🎯 Guia de Possíveis Perguntas sobre Autenticação, Segurança, Entrada de Informações do Usuário e APIs

1. Autenticação de Usuário

História

- **O que é autenticação de usuário?**
    - A autenticação de usuário é o processo de verificar se um usuário é quem ele diz ser, geralmente por meio de credenciais como **nome de usuário** e **senha**.

Questões Técnicas

- O que são as diferentes **formas de autenticação**? (ex: **autenticação por senha**, **autenticação multifatorial**)
- Qual a diferença entre **autenticação** e **autorização**?
- O que é o **OAuth 2.0** e como ele funciona? Quais são seus principais usos?
- Explique como o **JWT (JSON Web Token)** funciona na autenticação de usuários.
- O que é a **autenticação baseada em tokens** e como ela é implementada em APIs?
- O que são **cookies de sessão** e como eles são usados na autenticação de usuários?
- Quais são as melhores práticas para **armazenamento seguro de senhas**? Explique o uso de **hashing** e **salting**.

---

2. Segurança na Web

História

- **Qual a importância da segurança no desenvolvimento de websites?**
    - A segurança é vital para proteger dados sensíveis, como informações pessoais, bancárias e credenciais de login. Sites vulneráveis são alvos fáceis para **ataques cibernéticos** que podem resultar em danos à reputação, vazamento de dados ou prejuízos financeiros.

Questões Técnicas

- O que é um **ataque de injeção SQL (SQL Injection)** e como preveni-lo?
- Explique o que é **Cross-Site Scripting (XSS)** e como protegemos nossos sites contra esse tipo de ataque.
- O que é **Cross-Site Request Forgery (CSRF)** e como podemos evitá-lo?
- O que são **cookies seguros** e qual sua importância na segurança de um site?
- O que é o **HSTS (HTTP Strict Transport Security)** e como ele contribui para a segurança do site?
- Como proteger um **site contra ataques de força bruta** durante a autenticação?
- O que são **CORS (Cross-Origin Resource Sharing)** e como eles são usados para controlar o acesso a recursos entre diferentes origens?

---

3. Entrada de Informações do Usuário

História

- **Por que a entrada de informações do usuário precisa ser tratada de forma segura?**
    - As informações fornecidas pelos usuários em formulários ou interações com o site podem ser sensíveis. Uma validação e sanitização adequada é essencial para prevenir ataques, como injeção de código e garantir que os dados sejam tratados corretamente.

Questões Técnicas

- O que é **validação de dados** e qual a diferença entre **validação do lado do cliente** e **validação do lado do servidor**?
- O que é a **sanitização de entradas** e como ela protege contra ataques como **SQL Injection**?
- Quais são as melhores práticas para validar **endereços de e-mail** e **senhas** em formulários de entrada?
- O que são **expressões regulares** e como elas podem ser usadas na validação de entradas?
- O que é o conceito de **rate limiting** e como ele pode ser aplicado para proteger formulários de login e entradas de dados?
- O que são **tokens CSRF** e como eles ajudam a proteger contra ataques de falsificação de requisições?

---

4. APIs (Application Programming Interfaces)

História

- **O que é uma API e qual sua importância no desenvolvimento de aplicações web?**
    - Uma **API** é um conjunto de definições e protocolos que permite que softwares diferentes se comuniquem entre si. No contexto da web, APIs são usadas para permitir que sistemas externos interajam com o servidor, seja para **recuperar dados** ou **enviar comandos**.

Questões Técnicas

- O que é uma **API RESTful** e como ela se diferencia de outros tipos de API (ex: SOAP)?
- O que são os **métodos HTTP** principais usados em APIs REST (GET, POST, PUT, DELETE)?
- Como o **JSON** é usado para formatar dados em APIs e qual sua importância?
- O que é **autenticação via API** e como o **OAuth 2.0** é usado para autenticar usuários em uma API?
- Como garantir a **segurança de uma API**? Quais são as melhores práticas para evitar ataques como **injeção** ou **acesso não autorizado**?
- O que é o **CORS (Cross-Origin Resource Sharing)** e como ele afeta o consumo de uma API na web?
- O que é **rate limiting** em uma API e por que é importante para evitar abusos e garantir performance?
- O que é uma **API GraphQL** e como ela difere de uma API RESTful?

---

5. Autenticação e Segurança em APIs

Questões Técnicas

- Como implementar a **autenticação de usuário** usando **JWT (JSON Web Tokens)** em uma API?
- O que são **scopes** no contexto do OAuth 2.0 e como eles são usados para limitar o acesso de uma API?
- Explique como a **autenticação multifatorial (MFA)** pode ser implementada em uma API.
- Como o **OAuth 2.0** melhora a segurança das APIs ao permitir **autenticação delegada**?
- Como você pode usar **API keys** para controlar o acesso a uma API?
- O que são **webhooks** e como eles podem ser usados em APIs para notificar eventos externos?

---

6. Boas Práticas de Segurança para Desenvolvimento Web

Questões Técnicas

- O que é o **princípio do menor privilégio** e como ele deve ser aplicado no desenvolvimento de um site?
- O que são **headers de segurança HTTP** como **Content-Security-Policy (CSP)** e **X-Content-Type-Options**?
- Como você pode usar a **criação de sessões seguras** para proteger a autenticação do usuário?
- O que são **backups regulares** e qual a sua importância na segurança dos dados de um site?
- O que é **hacking ético** e como ele pode ser usado para identificar vulnerabilidades em seu sistema?

⚡ Dicas Rápidas de Segurança e Autenticação

Ação	Dica
Armazenamento de Senhas	Use **bcrypt** para hashing e **salt** das senhas dos usuários
Proteção de API	Use **OAuth 2.0** e **JWT** para autenticação e autorização seguras
Proteção de Dados	Sempre use **HTTPS** para criptografar o tráfego de dados entre o cliente e o servidor
Validação de Entrada	Valide todas as entradas do usuário, tanto no **lado do cliente** quanto no **lado do servidor**
APIs Seguras	Implemente **rate limiting** e **autenticação baseada em chave** nas suas APIs

---