HTTP

1 – Como Funciona? Forma de comunicação.

O HTTP opera em um modelo cliente-servidor, onde os clientes, geralmente navegadores da web, solicitam recursos e os servidores web respondem a essas solicitações. A comunicação entre cliente e servidor ocorre por meio de mensagens individuais, que são trocadas em um formato específico definido pelo protocolo.

Quando um usuário digita um URL em seu navegador e pressiona Enter, o navegador inicia uma solicitação HTTP para o servidor correspondente ao URL fornecido. Essa solicitação é composta por um método HTTP, que indica a ação a ser realizada no recurso solicitado, além de informações adicionais, como cabeçalhos e um corpo de mensagem.

O servidor recebe a solicitação e processa-a de acordo com o método HTTP especificado. Em seguida, o servidor envia uma resposta de volta ao cliente. Esta resposta também é composta por um código de status HTTP, cabeçalhos de resposta e um corpo de mensagem contendo os dados solicitados.

A comunicação entre cliente e servidor ocorre tipicamente sobre o protocolo TCP/IP, que fornece uma conexão confiável e orientada a conexão para a transmissão das mensagens HTTP. No entanto, o HTTP também pode ser transmitido sobre uma conexão TCP criptografada com TLS, garantindo a segurança e a integridade dos dados transmitidos.

Além disso, o HTTP é um protocolo extensível, o que significa que novas funcionalidades podem ser adicionadas a ele ao longo do tempo. Isso permite que o HTTP evolua para atender às demandas crescentes da web moderna, incluindo suporte para novos tipos de mídia, como streaming de vídeo e áudio, e melhores práticas de segurança.

2 – O que são métodos? Explique os principais.

Os métodos HTTP, são a parte fundamental do protocolo de comunicação HTTP, responsáveis por indicar a ação que deve ser realizada em um recurso específico no servidor. Cada método define uma operação diferente

que pode ser executada sobre um recurso, permitindo uma variedade de interações entre clientes e servidores web. Abaixo alguns dos métodos mais utilizados:

1. O método GET é usado para solicitar dados de um recurso específico no servidor. Quando um cliente envia uma solicitação GET, ele está pedindo ao servidor para enviar os dados do recurso solicitado de volta para o cliente. Esse método é amplamente utilizado para recuperar páginas da web, imagens, arquivos de estilo e outros tipos de conteúdo estático.
2. O método POST é usado para enviar dados para o servidor, geralmente para submeter formulários HTML ou para enviar dados para serem processados e armazenados. Ao contrário do método GET, que envia os parâmetros na URL, o método POST envia os dados no corpo da mensagem HTTP, tornando-o mais adequado para o envio de grandes quantidades de dados ou dados sensíveis.
3. O método PUT é usado para enviar dados para serem armazenados em um recurso específico no servidor. Ele substitui completamente os dados existentes no recurso pelo novo conteúdo enviado na solicitação. O método PUT é comumente utilizado para atualizar recursos existentes, como arquivos ou documentos.
4. DELETE: O método DELETE é usado para solicitar a remoção de um recurso específico no servidor. Quando um cliente envia uma solicitação DELETE, ele está solicitando ao servidor que remova permanentemente o recurso especificado. Esse método é útil para remover dados ou arquivos que não são mais necessários.
5. O método PATCH é usado para aplicar modificações parciais a um recurso no servidor. Ao contrário do método PUT, que substitui completamente os dados do recurso, o método PATCH permite que o cliente envie apenas as alterações necessárias para serem aplicadas ao recurso existente. Isso é útil quando apenas uma parte do recurso precisa ser atualizada.

Além desses métodos principais, existem outros métodos menos comuns, como HEAD, OPTIONS e TRACE, que têm funções específicas, como

obter informações de cabeçalho, determinar as opções de comunicação disponíveis e rastrear a rota de uma solicitação, respectivamente.

3 – Quais os problemas do protocolo?

Embora o Protocolo de Transferência de Hipertexto (HTTP) tenha sido uma peça fundamental no desenvolvimento da World Wide Web, ele apresenta alguns problemas e desafios que precisam ser abordados para melhorar a segurança, eficiência e confiabilidade da comunicação na web.

1. Falta de Segurança: O HTTP opera sobre uma conexão não criptografada, o que significa que os dados transmitidos entre o cliente e o servidor são enviados em texto simples e podem ser interceptados por terceiros mal-intencionados. Isso coloca em risco a privacidade e a integridade dos dados transmitidos.

2. Vulnerabilidade a Ataques de Interceptação: Devido à falta de criptografia, o HTTP está sujeito a ataques de interceptação, nos quais um invasor pode capturar e modificar os dados transmitidos entre o cliente e o servidor sem o conhecimento do usuário.

3. Ausência de Controle de Integridade dos Dados: O HTTP não oferece um mecanismo embutido para verificar a integridade dos dados transmitidos entre o cliente e o servidor. Isso significa que não há garantia de que os dados recebidos pelo cliente são idênticos aos dados enviados pelo servidor, o que pode levar a erros de transmissão, corrupção de dados e ataques de falsificação.

4. Desperdício de Recursos de Rede: O HTTP frequentemente requer a abertura de uma nova conexão TCP para cada solicitação e resposta, o que pode resultar em um desperdício significativo de recursos de rede, especialmente em sites que carregam vários recursos, como imagens, scripts e folhas de estilo.

4 – Cite exemplos de requisições.

Outra forma de realizar solicitações HTTP é através da ferramenta de linha de comando curl. O curl é uma ferramenta amplamente utilizada para transferência de dados com suporte a diversos protocolos, incluindo HTTP.

No exemplo abaixo, temos um comando curl que envia uma solicitação GET para o URL 'www.leagueoflegends.com':

```
curl -X GET -H "Content-type: application/json" 'www.leagueoflegends.com'
```

Neste comando:
-X GET específica explicitamente o método GET, embora GET seja o método padrão e, portanto, não precisaria ser especificado novamente.
-H "Content-type: application/json" define um cabeçalho adicional na solicitação, indicando que o tipo de conteúdo esperado é JSON.
'www.leagueoflegends.com' é o URL para o qual a solicitação GET está sendo enviada.

Como exemplo de como usar o curl para enviar uma solicitação PATCH para o URL 'www.leagueoflegends.com':

```
curl -X PATCH -H "Content-type: application/json" 'www.leagueoflegends.com'
```

Neste comando:
-X PATCH especifica explicitamente o método PATCH, indicando que estamos enviando uma solicitação para aplicar modificações parciais a um recurso no servidor.
-H "Content-type: application/json" define um cabeçalho adicional na solicitação, informando que o tipo de conteúdo esperado é JSON.
'www.leagueoflegends.com' é o URL para o qual a solicitação PATCH está sendo enviada.

HTTPS

1 – Como Funciona? Forma de comunicação.

A sigla HTTPs significa Hypertext Transfer Protocol Secure, ou Protocolo de Transferência de Hipertexto Seguro. O termo é muito parecido com a conhecida sigla HTTP, mas com a diferença do “s” de segurança incluído no final.

Para proteger o seu site com HTTPs, é necessário obter um Certificado de Segurança SSL (Camada de Soquetes Segura) ou TLS (Segurança na Camada de Transporte).

Ao adquirir um certificado SSL/TLS para o seu site, a tecnologia gera algumas chaves de proteção para garantir a segurança do seu visitante. Primeiramente, o seu site envia uma chave pública ao navegador do visitante para se identificar e comprovar que é uma página segura, verificada por uma autoridade de certificação válida. Vale lembrar que a autoridade de certificação é a entidade que emite o certificado SSL/TLS.

Ao começar a usar o HTTPS, seu site também terá uma chave privada, que deve ser mantida em segurança e não é compartilhada com nenhum navegador ou elemento externo. Essa é a chave que comanda a comunicação do seu site com os seus visitantes, portanto qualquer pessoa em posse dessa chave poderá decifrar os dados transmitidos. Cuidado com ela!

Por fim, a combinação da sua chave pública com a sua chave privada gera as chaves de sessão, que valem apenas para cada visita feita ao seu site e fazem a criptografia em si da comunicação entre o servidor web e o navegador do usuário.

2 – Qual diferença entre HTTP e HTTPS?

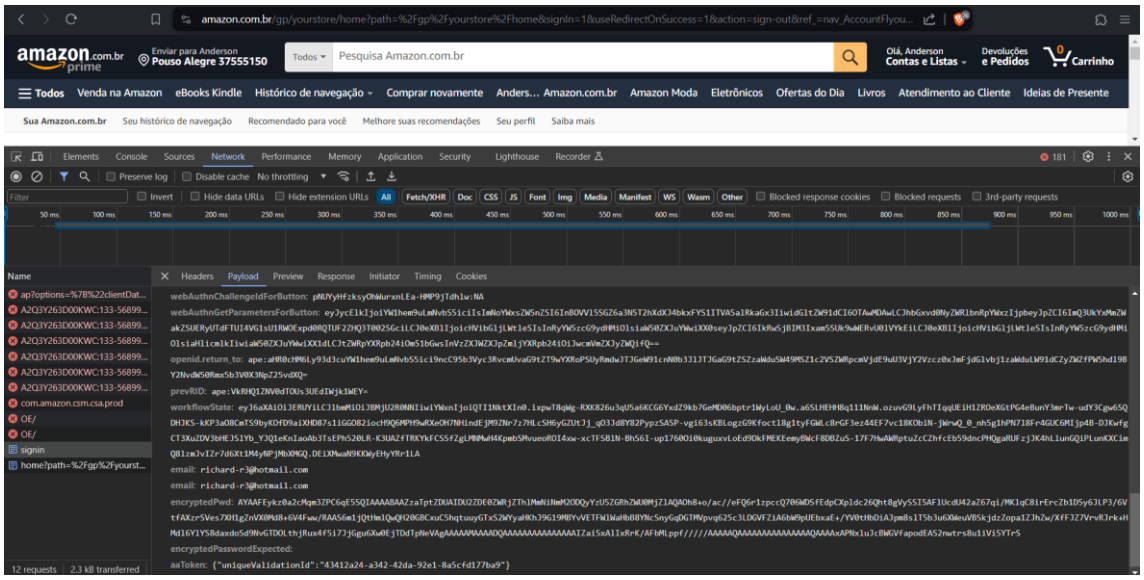
Enquanto o HTTP (Hypertext Transfer Protocol) é um protocolo de comunicação aberto amplamente usado na internet, onde transfere dados entre navegadores e servidores em texto simples, o que pode deixar as informações vulneráveis a interceptações por terceiros mal-intencionados. Por outro lado, o HTTPS (Hypertext Transfer Protocol Secure) é uma versão segura do HTTP. Ele criptografa os dados transferidos entre o navegador e o servidor, tornando-os ilegíveis para qualquer pessoa que não possua a chave de criptografia. Isso oferece uma camada adicional de segurança e proteção para os usuários, especialmente ao enviar informações sensíveis, como senhas e informações financeiras.

3 – Conexões simultâneas

No contexto do HTTPS (Hypertext Transfer Protocol Secure), o número de conexões simultâneas refere-se à quantidade de conexões ativas que podem ser estabelecidas entre um cliente (como um navegador da web) e um servidor seguro.

Sendo assim o número de conexões simultâneas no contexto do HTTPS pode variar com base na configuração específica do servidor ou aplicativo utilizado.

4 – Simulação



Na imagem acima é possível verificar a criptografia que ocorre quando é feito o login na página do AmazonPrime.

Campos de Conteúdo: Informações como URLs, cookies e dados de formulários são criptografados durante a transmissão.

Emissão de Certificado: Autoridades como Let's Encrypt oferecem certificados SSL/TLS gratuitos e automatizados.

Fluxo de Validação: Inclui verificação da integridade do certificado, validade, status de revogação e autenticidade do emissor