1. **O QUE É HTTP?**

HTTP: Hypertext Transfer Protocol

Protocolo HTTP segue o modelo Client-Server, o navegador é o cliente.

O HTTP estabelece regras de comunicação.

Outro modelo de comunicação é o Peer-to-Peer (P2P) onde neste, os clientes distribuem o trabalho ou divide as obrigações entre os participantes da comunicação.

FTP: File Transfer Protocol

SQL: Structured Query Language, é uma linguagem de consulta à banco de dados.

BitTorrent: programa P2P

SMTP: Simple Mail Transfer Protocol.

1. **A web Segura – HTTPS**

Utilizar as ferramentas de desenvolvedor para avaliar as informações.

SSL/TLS: Secure Socket Layer/Transport Layer Security é uma camada de proteção de dados, com criptografia.

Para ser HTTPS precisa de um certificado digital, uma chave (pública e privada) que criptografa os dados para o servidor.

1. **Endereços**

Protocolo (https://) em seguida o domínio (ssfsfdfsg.com.br) o último elemento é o domínio raíz.

DNS: Domain Name System, um Sistema que converte um endereço textual em endereço numérico.

No DNS pode especificar a porta de acesso explicitamente colocando no final “:XX” onde XX é o número da porta. A porta padrão para o protocolo HTTP é :80, e para o HTTPS é :443. A porta padrão do FTP é 21 e do SSH é 22.

A construção de uma Uniform Resource Locator (URL) é: protocolo://domínio:porta/caminho/recurso.

Uniform Resource Identifier (URI) é a versão mais geral da URL, outro tipo de URI é a Uniforme Resource Name (URN).

1. **O cliente pede e o servidor responde**

Request: requisição navegador 🡪 servidor.

Responde: resposta servidor 🡪 navegador.

Cada recurso e requisição no HTTP é independente do anterior. O HTTP não mantém o estado das requisições.

Em um login é gerado uma credencial de identificação que o navegador armazena para futuras requisições. Essas informações armazenadas, em sessões, são os conhecidos “cookies”. Os cookies são associados a um domínio.

1. **Depurando a requisição HTTP**

Na requisição HTTP pelas “ferramentas de desenvolver 🡪 rede”. Na requisição de um HTML temos diversas requisições em seguida.

Método HTTP GET: Receber as informações.

Alguns status code:

200: tudo ok!

301: redirecionamento do lado do cliente, força o cliente a fazer outra requisição.

404: Não encontrado.

500: Erro interno do servidor.

2XX: resposta esperada.

3XX: redirecionamento da requisição.

4XX: erro do lado do cliente.

5XX: erro do lado do servidor.

1. **Parâmetros de Requisição**

Nas requisições do tipo GET passamos a busca dentro da URL, começando por “?” e separando os termos de busca com &, exemplo: https://generico.com/?a=2&b=3

Método HTTP POST: Enviar informações dentro da requisição, permite que não apareça as informações na URL. Para tal, é preciso usar a tag “form”, configurando o atributo “method” como POST. Por padrão o method é GET.

É possível implementar o método GET na URL da forma: https://generico.com/a/parametro1/bOUparametro2/...

Método HTTP DELETE: Deletar informação.

Método HTTP PUT: Atualização informação.

Métodos DELETE e PUT são implementados por meio de JavaScript.

Esses últimos são mais utilizados para Web Services (WS). Em um WS o resultado vem em formato XML ou JSON.

1. **Serviços na WEB com REST**

Em aplicações web não é interessante retornar um HTML, pelo problema de achar as informações necessárias. O uso de arquivos XML ou JSON é melhor numa requisição de aplicações.

Em alguns momentos se faz necessário informar ao protocolo HTTP qual o tipo de formato do arquivo esperado, para isso utilizamos o cabeçalho “Accept”. Ex.: Accept: text/html

O REST (REpresentational State Transfer) é o modelo de operação de serviços web. Possui três elementos:

- Recurso (URI)

- Operações (GET/POST/PUT/DELETE)

- Representação (JSON/XML/HTML)

Tipos de dados:

MIME Types são formatos do cabeçalho HTTP, a estrutura de definição é “tipo/subtipo”. Alguns tipos:

* text / image / application / audio / vídeo

Alguns subtipos

* text: plain / html / css / javascript /
* Image: gif / png / jpeg
* Audio: midi / mpeg / webm / ogg / wav
* Video: mp4
* Application: xml / pdf

**8. HTTP2 – Por uma web mais eficiente**

No HTTP2 existe o algoritmo de compactação GZIP. Além disso, o HTTP2 envia e requisita documentos em binário compactados pelo algoritmo HPACK. O HTTP2 traz a criptografia TLS como padrão.

No HTTP2 não é preciso enviar os mesmos headers a cada GET. No caso de necessitar uma mudança de cabeçalho, basta mudar o trecho necessário.

O HTTP2 é stateful, ou seja, guarda o estado.

No HTTP2 é possível o servidor já enviar todos os arquivos necessários para uma requisição de cliente antes do cliente requisitá-los de fato. Por exemplo, enviar já o css e js junto com o html requisitado.

NP HTTP1.1 cada requisição abre uma conexão TCP e ao final fecha a conexão. Um método usado é manter a conexão TCP aberta por um tempo para desonerar o tempo de abertura e fechamento do TCP. Geralmente, deixa aberto entre 4 e 8 requisições. No HTTP2, o arquivo requisitado em um pedido pode chegar depois de outra requisição, mantendo um serviço paralelo.