CENTRO UNIVERSITÁRIO UNISATC

BACK-END

Prof. Bruno Kurzawe

HTTP, Requisições e Principais Protocolos

HTTP (Hypertext Transfer Protocol) é um protocolo de comunicação utilizado na World Wide Web. Ele define a forma como as mensagens são formatadas e transmitidas pela internet, permitindo a comunicação entre um cliente (como um navegador web) e um servidor.

Natureza Cliente-Servidor:

Cliente: É o software que faz a solicitação de algum recurso na web. Pode ser um navegador (como Chrome, Firefox, etc.), um aplicativo móvel ou qualquer outra aplicação que faça requisições HTTP.

Servidor: É o software que atende às solicitações dos clientes e envia de volta a resposta. Pode ser um servidor web como Apache, Nginx, entre outros.

Comunicação:

Pedido (Request): Um cliente envia uma requisição HTTP para um servidor. Esta requisição contém informações como o tipo de requisição (GET, POST, etc.), o caminho do recurso desejado e outros cabeçalhos (como informações sobre o cliente, cookies, etc.).

Resposta (Response): O servidor processa a requisição e envia de volta uma resposta HTTP. Esta resposta contém um código de status (indicando se a requisição foi bem-sucedida, redirecionada, etc.) e o conteúdo solicitado (como uma página web).

Métodos HTTP Comuns:

GET	Obtém informações do servidor. Usado principalmente para solicitar páginas web.
POST	Envia dados ao servidor. Usado para submeter formulários e enviar dados para processamento.
PUT	Atualiza um recurso no servidor. Substitui completamente o recurso existente.
DELETE	Remove um recurso no servidor.
PATCH	Atualiza parcialmente um recurso no servidor.
HEAD	Similar ao GET, mas retorna apenas os cabeçalhos, sem o conteúdo.
OPTIONS	Obtém informações sobre os métodos suportados pelo servidor.

Estadoless (Sem Estado):

O HTTP é um protocolo stateless, o que significa que cada requisição entre cliente e servidor é independente das anteriores. O servidor não mantém informações sobre requisições passadas. Isso implica que, para manter o estado entre requisições (como em um sistema de login), técnicas como cookies ou sessões são usadas.

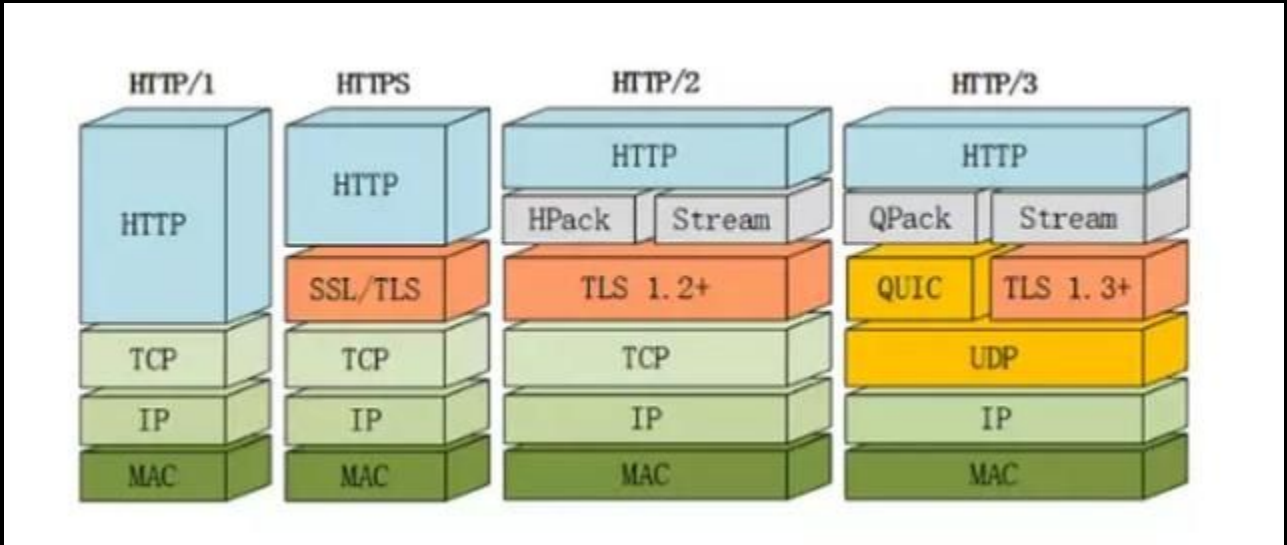
URLs (Uniform Resource Locators):

As URLs são endereços que identificam recursos na web. Elas contêm o protocolo (http:// ou https://), o domínio (exemplo.com) e o caminho para o recurso (/caminho/do/recurso).

Exemplo de URL:

https://www.exemplo.com/pagina

Versões e Evolução do HTTP



HTTP/1.0 (1996):

- Versão inicial do HTTP.
- Cada requisição abre uma nova conexão.
- Sem suporte para cabeçalhos persistentes.
- Limitações de desempenho devido à abertura constante de conexões.

HTTP/1.1 (1997):

- Introduziu a persistência de conexão, permitindo múltiplas requisições em uma única conexão.
- Melhorou significativamente a eficiência e desempenho em comparação com a versão 1.0.
- Introduziu cabeçalhos de controle de cache (Cache-Control, ETag, Last-Modified).
- É a versão predominante por um longo período e ainda amplamente utilizada.

HTTP/2 (2015):

- Introduziu o conceito de multiplexação, que permite várias requisições e respostas simultâneas em uma única conexão TCP.

- Utiliza compressão de cabeçalhos para reduzir a sobrecarga de largura de banda.

- Priorização de requisições para otimizar a entrega de recursos críticos.

- Foi projetado para melhorar significativamente o desempenho e a eficiência, especialmente em ambientes modernos de alta largura de banda.

HTTP/3 (2018):

- Introduzido para resolver algumas limitações do HTTP/2.
- Usa o novo protocolo de transporte QUIC (Quick UDP Internet Connections) em vez do TCP, o que reduz a latência.
- Mantém as melhorias do HTTP/2, como a multiplexação, mas com uma ênfase ainda maior na eficiência e na redução de atrasos.
- Ainda está em processo de adoção generalizada, mas promete melhorias significativas em relação ao HTTP/2.

Respostas HTTP

As Respostas HTTP são mensagens enviadas pelo servidor em resposta a uma requisição feita por um cliente (como um navegador web, aplicativo móvel, etc.). Elas contêm informações sobre o resultado da requisição e permitem que o cliente saiba se a operação foi bem-sucedida, se houve um erro ou se há alguma ação adicional que precisa ser tomada.

Uma resposta HTTP consiste em três partes principais:

- Linha de Status
- Cabeçalhos
- Corpo da Resposta

Linha de Status:

Esta linha indica o resultado da requisição e é composta por um código de status e uma mensagem associada. Exemplos de códigos de status incluem 200 (OK), 404 (Not Found) e 500 (Internal Server Error).

Linha de Status:

200 - OK	Significa que a requisição foi bem-sucedida. Esta é a resposta padrão para uma requisição GET.
201 - Created	Indica que a requisição foi bem-sucedida e que um novo recurso foi criado como resultado (usado em POST).
204 - No Content	Significa que a requisição foi bem-sucedida, mas não há conteúdo para enviar de volta.
400 - Bad Request	Indica que a requisição feita pelo cliente é inválida ou mal formada.
401 - Unauthorized	Informa que a requisição requer autenticação do cliente (por exemplo, login e senha) para ser processada.
403 - Forbidden	Indica que o servidor entende a requisição, mas o cliente não tem permissão para acessar o recurso.
404 - Not Found	Indica que o recurso solicitado não foi encontrado no servidor.
500 - Internal Server Error	Indica que ocorreu um erro no servidor ao processar a requisição
503 - Service Unavailable	Indica que o servidor não está pronto para lidar com a requisição. Isso pode ser devido a sobrecarga ou manutenção temporária.
301 - Moved Permanently	Informa que o recurso solicitado foi permanentemente movido para uma nova localização. O cliente deve usar a nova URL.

Cabeçalhos:

Os cabeçalhos fornecem informações adicionais sobre a resposta, como o tipo de conteúdo, a data de modificação, a codificação e muito mais. Eles são chave-valor pares que fornecem metadados sobre a resposta.

Cabeçalhos:

Content-Type	Indica o tipo de mídia do conteúdo retornado na resposta. Pode ser HTML, JSON, XML, etc. Exemplo: Content-Type: application/json
Content-Length	Especifica o tamanho, em bytes, do conteúdo retornado na resposta. Exemplo: Content-Length: 1024
Cache-Control	Controla a política de cache tanto no lado do cliente quanto do servidor. Exemplo: Cache-Control: max-age=3600
Set-Cookie	Define um cookie no navegador do cliente. Pode ser usado para manter o estado de sessões. Exemplo: Set-Cookie: session-id=123456; Path=/; Expires=Wed, 21 Jul 2021 12:30:45 GMT
Location	Especifica o URI para o qual o cliente deve ser redirecionado após uma operação bem-sucedida. Exemplo: Location: https://www.exemplo.com/nova-pagina
Allow	Indica os métodos HTTP permitidos para o recurso. Exemplo: Allow: GET, POST, PUT
Expires	Define uma data e hora de expiração para a resposta, após a qual o conteúdo não é mais considerado válido. Exemplo: Expires: Wed, 21 Jul 2021 12:30:45 GMT

Corpo da Resposta:

O corpo contém o conteúdo da resposta em si. Pode ser HTML para uma página web, JSON para dados estruturados, um arquivo binário, ou qualquer outro tipo de conteúdo.

Exemplos de retorno

```json
{
  "nome": "John Doe",
  "email": "john@exemplo.com",
  "idade": 30
}
```

```xml
<pessoa>
  <nome>John Doe</nome>
  <email>john@exemplo.com</email>
  <idade>30</idade>
</pessoa>
```

```
Este é um exemplo de corpo de requisição em texto
```

Fim da aula 08…