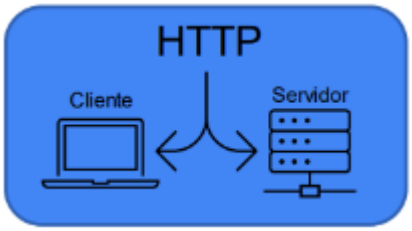
Protocolo de comunicação Web HTTP

Como funciona o protocolo HTTP?

Protocolo HTTP (HyperText Transfer Protocol)



Definido pelos RFCs 1945 e 2116 (mais antigas).

- Protocolo de Comunicação
- Browser → Implementa o cliente HTTP
  - Cliente HTTP pode não usar um browser
- Servidor → Host objetos web

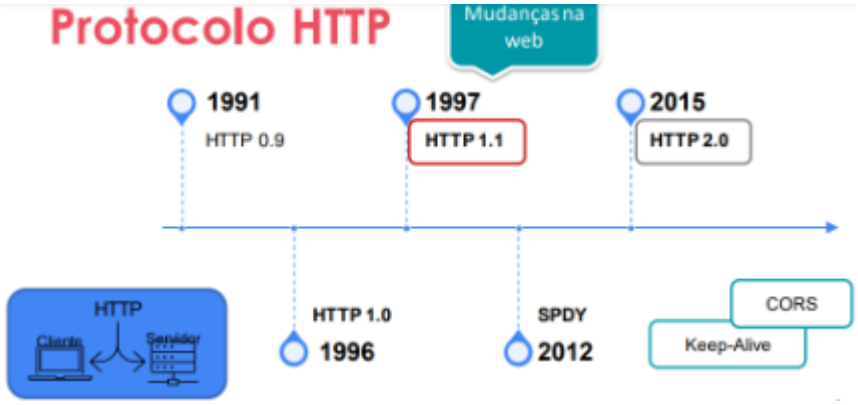
Arquitetura Client-Server

Cliente

Mensagens - Request HTTP (Objetos Web)

Servidor

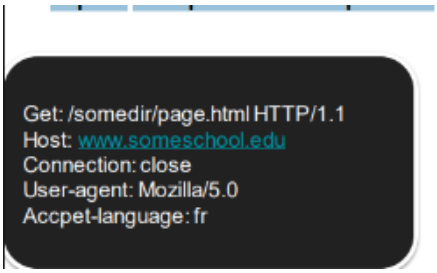
Mensagens – Response HTTP (Protocolo TCP)

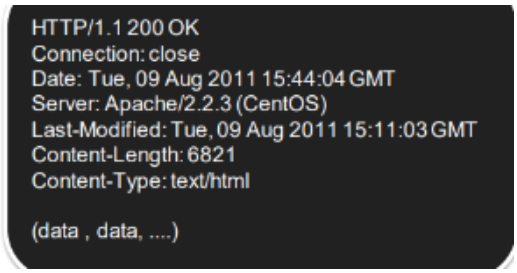


Mensagens HTTP

Tipos: request & response

As requisições do tipo **GET** são recomendadas para obter dados de um determinado recurso. Como em um formulário de busca ou em uma listagem de todos os produtos cadastrados.





Tipos: post

O **método** de requisição **POST** foi projetado para solicitar que o servidor web aceite os dados anexados no corpo da mensagem de requisição para armazenamento. Ele é normalmente usado quando se faz o upload de um arquivo ou envia-se um formulário web completo.

Formatos para Enviar Dados

- HTTP Entity Body

Formatos mais usados:

XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<authentication-context>
  <username>my_username</username>
  <password>my_password</password>
  <validation-factors>
    <validation-factor>
      <name>remote_address</name>
      <value>127.0.0.1</value>
    </validation-factor>
  </validation-factors>
</authentication-context>
```

JSON

```json
{
   "username" : "my_username",
   "password" : "my_password",
   "validation-factors" : {
      "validationFactors" : [
         {
            "name" : "remote_address",
            "value" : "127.0.0.1"
         }
      ]
   }
}
```

Qual é o proposito disso?
Armazenamento, **Transmissão**, **Reconstrução** desses dados.

Extensible Markup Language

Objetivo?
Serialization, serialização dos dados para haver **Comunicação** de **Metadados**.

JavaScript Object Notation

Características

- Lightweight: estrutura mais simples
- Independente de linguagem de programação.
- Simples

XML e JSON

Comum

- Auto-descritivos
- Hierárquicos
- Independentes de linguagem de programação
- Vasta utilização

Diferenças

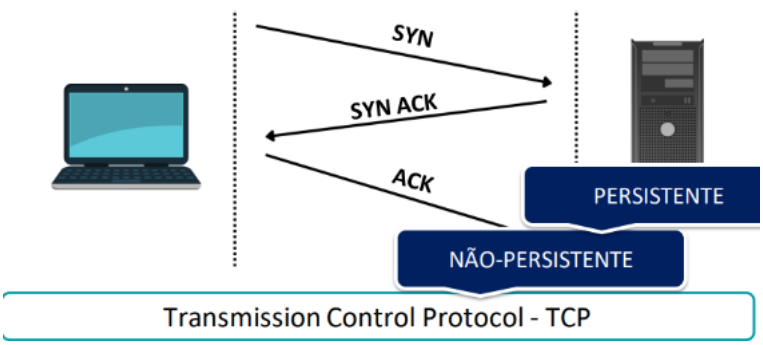
- Tags
- Legibilidade
- Sucinto
- Utilização de arrays

Por qual optar?

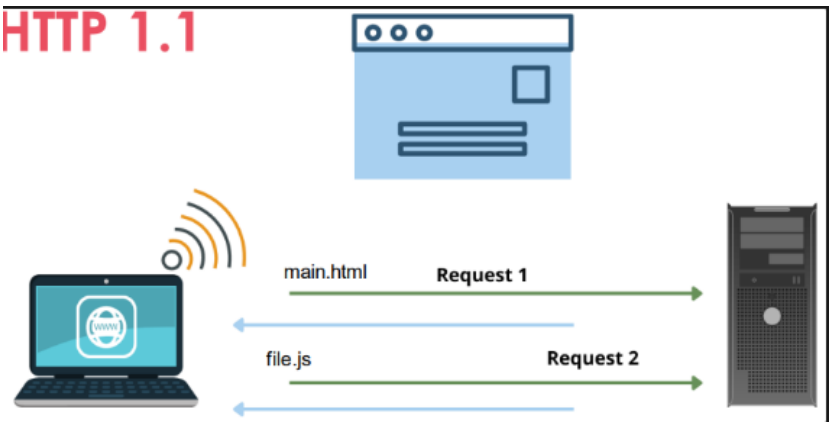
- Tecnologia
- Complexidade
- Metadados
- Aplicações leves

Particularidades do HTTP versão 1.1

Protocolo HTTP



HTTP 1.1

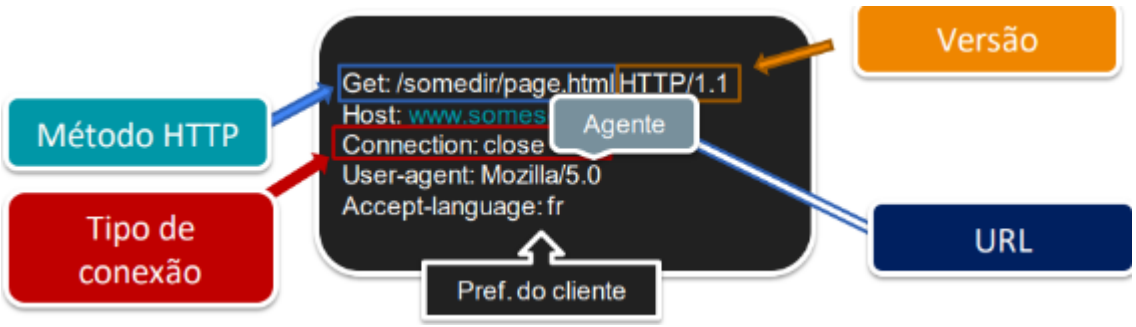


Mensagens HTTP – Request & Response

HTTP Request

Estrutura:

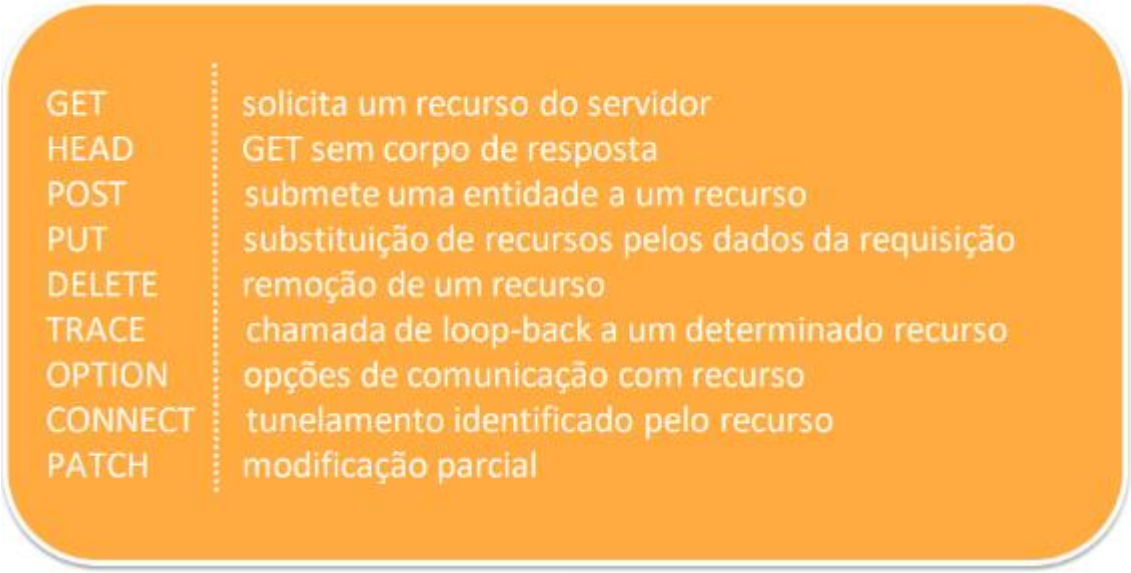
- Texto em ASCII



Métodos mais utilizados

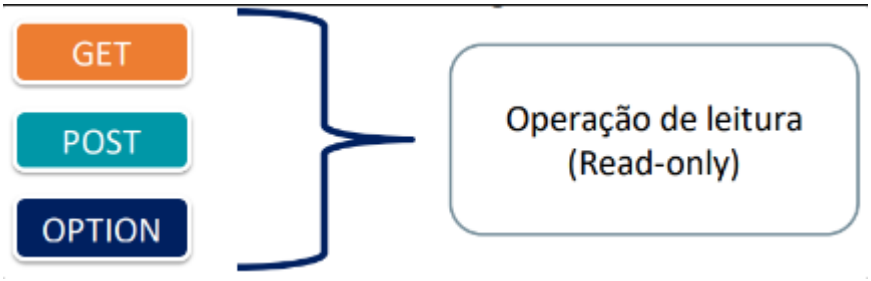
- GET
- POST

Outros métodos



Métodos seguros

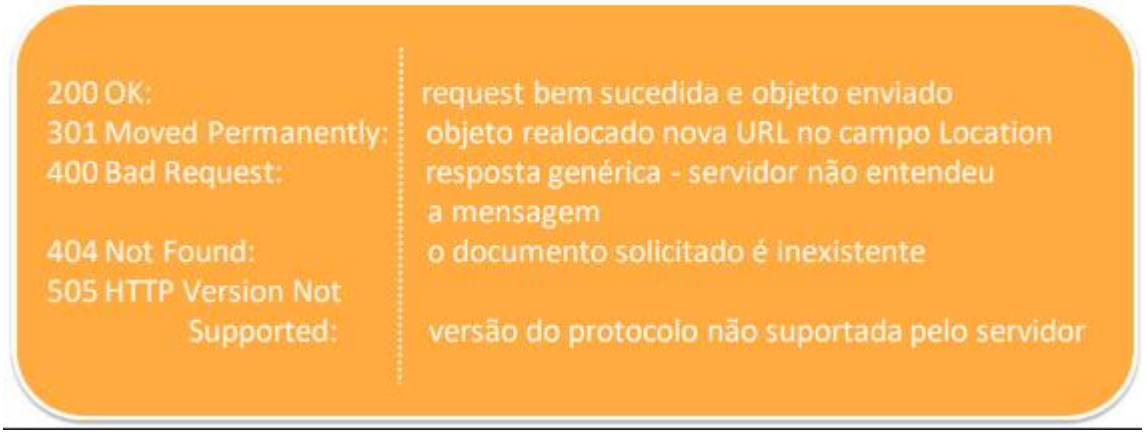
Eles não modificam estados, apenas de leitura.



HTTP: Response

Status Line

- Versão do protocolo



Status code

- Information response (100 – 199)
- Successful response (200 – 299)
- Redirection response (300 – 399)
- Client error response (400 – 499)
- Server error response (500 – 599)

Web Distributed Authoring and Versioning

- 102 – Processing
- 207 – Multi-status
- 208 – Already Reported
- 422 – Unprocessable Entity
- 423 – Locked
- 424 – Failed Dependency

Ele permite:

- Webpage Meta: Add, deletar, retrieve
- Link pages
- Criação de conjuntos de documentos
- Copy e move

- Lock: documento editado - 1

Status da mensagem

Header Lines
- Conexão encerrada
- Dados da mensagem: Data, servidor, …
- Content-type: Tipo de dado

Campos
- Entity Header (definições)
- Entity Body

Para que serve Cookie e Cache?

O HTTP PROTOCOL é:
- Stateless
  - Não guarda as informações do cliente, mas pesa na utilização de banda.
- Client/Server

O que são Cookies?
Pequenos pedaços ou blocos de dados criados e utilizados pelo servidor para persistir dados no dispositivo do cliente.

Especificação: RFC 6265

- Os cookies ficam armazenados no Header File

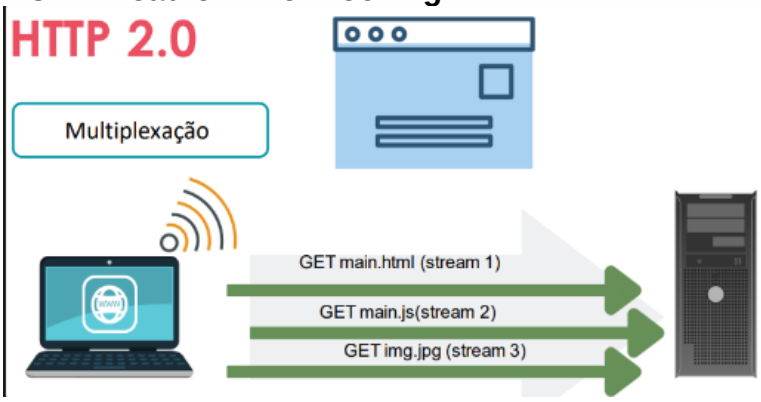
Tipos
- *Cookies de Sessão*
  - A partir do encerramento da sessão eles são apagados.
- *Cookies de persistentes*
  - Podem se manter por messes e anos.
  - Exemplo: Login Ativo, Carrinho de ecommerce, etc.

Cookies e Privacidade
- Está dentro da LGPD, ele pede sua autorização.
- Hacking

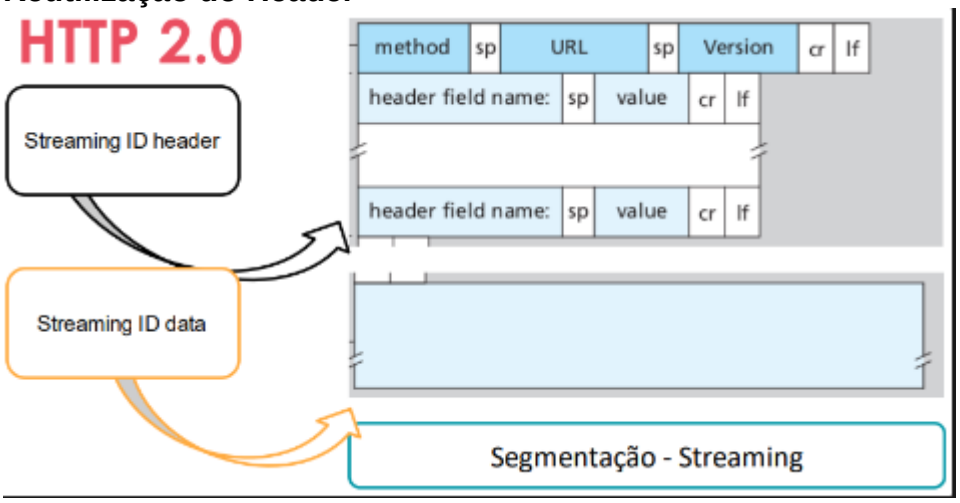
Caching
Caching **é uma técnica que guarda uma cópia de dado recurso e mostra de volta quando requisitado**.
Quando um web cache tem um recurso requerido em seu armazenamento, ele intercepta a solicitação e retorna sua cópia ao invés de fazer o download novamente do servidor original.
Praticamente um proxy intermediário.

HTTP 2.0 - Atualizações do protocolo

HOL – Head of Line Blocking

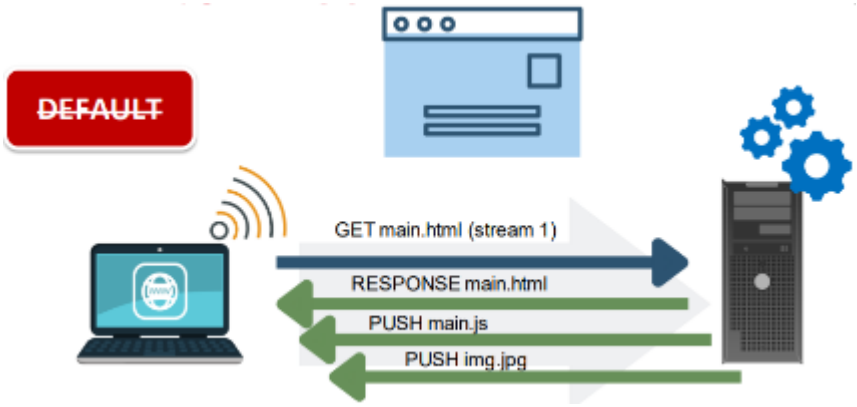


Reutilização do Header



HTTP 2.0 - Push

O servidor envia tudo relacionado as GET solicitado, facilitando na economia de processamento e banda.



O cliente suporta?

- Única conexão persistente
- Compressão de header
- Server push
- HTTPS por padrão - TLS
- Negociação no handshake

CONTRAS

PUSH

- Configuração incorreta

Mixe 1.1 e 2.0

- Lentidão
- Load balancer – HTTP 1.1

Servidores / Sistemas de aplicação

Apache

- Datado de 1995
- Contribuição c/ WWW
- Opensource & free
- Comunidade: Apache Software Foundation

Características

- Modular, dinâmico e carregável
- Multiple Request Processing modes (MPMs)

- Altamente escalável (+ 10,000 conexões simultâneas fácil)
- Lida com arquivos estáticos, index, auto -indexing e negociação de conteúdo
- TLS/SSL via OpenSSL ou wolfSSL.
- Servidores virtuais com endereços baseados em IP ou nome
- Compatibilidade com IPV6
- Suporte à versão 2.0 do HTTP
- Proxy reverso, websocket ...

Xampp

Voltado para testes.

Pacote/ambiente:

- Apache server
- MariaDB
- Interpretadores script – PHP/Perl

NGINX

Servidor / 2004:

- Web server
- Proxy reverso
- Load balancer
- Mail proxy
- HTTP cache

Recursos

- 10mil conexõessimultâneas
- Lida com arquivos estáticos, index, autoindexing
- Proxy reverso
- Load balance
- TLS/SSL com SNI via OpenSSL
- Suporte à HTTP 2.0
- Compatível com IPV6
- FastCGI SCGI, uWSGI com cache
- Suporte a gRPC (v. 1.13.10)
- Servidores virtuais baseados em IP e nome
- WebSocket desde 1.3.13
- URL rewriting and redirection[35][36]