

Desenvolvimento Web II

Aula 04 - Protocolos de Comunicação na Web

Prof. Fabricio Bizotto

Instituto Federal Catarinense
fabricio.bizotto@ifc.edu.br

Ciência da Computação
20 de março de 2025

1 Protocolos de Comunicação

- Conceito
- HTTP
 - Métodos HTTP
 - Códigos de Status
 - Requisição e Resposta
- HTTPS
- Websocket

2 Material Complementar

3 Experimentos

Protocolo de comunicação é um conjunto de regras que definem como a comunicação entre dois ou mais dispositivos deve ocorrer. Existem diversos protocolos de comunicação, cada um com suas características e finalidades específicas. Alguns exemplos de protocolos de comunicação são:

- **HTTP** - *Hypertext Transfer Protocol*
- **HTTPS** - *Hypertext Transfer Protocol Secure*
- **FTP** - *File Transfer Protocol*
- **SMTP** - *Simple Mail Transfer Protocol*
- **SSH** - *Secure Shell*
- **Websocket** - *Websocket Protocol*







Protocolos de Comunicação

HTTP

Hypertext Transfer Protocol


- **Protocolo** de comunicação utilizado para transferência de dados na **World Wide Web (WWW)**.
- Utiliza o protocolo **TCP**, normalmente na porta **80**.
- É um protocolo **stateless**, ou seja, não mantém estado entre requisições.
- É um protocolo **client-server**, ou seja, o cliente envia uma requisição e o servidor responde.
- A **URL** é utilizada para identificar o recurso que será acessado pelo cliente (browser, por exemplo).
- **Local Storage e Cookies** são utilizados para manter estado entre requisições e melhorar a experiência do usuário.

HTTP Request Methods

 GET	 POST	 PUT	 DELETE	 PATCH	 HEAD
retrieve data from server	add data to an existing file or resource	update(replace) an existing file or resource in server	delete data from server	update a resource partially (modify)	retrieve the resource's headers

- **CONNECT** is used to open a two-way socket connection to the remote server;
- **OPTIONS** is used to describe the communication options for specified resource;
- **TRACE** is designed for diagnostic purposes during the development.
- **HEAD** retrieves the resource's headers, without the resource itself.

Figura: Métodos HTTP ¹.

¹  Twitter Post: Top 9 HTTP Request Methods

HTTP - Códigos de Status

Status Code	Action	Description
200	OK	Successfully retrieved resource
201	Created	A new resource was created
204	No Content	Request has nothing to return
301 / 302	Moved	Moved to another location (redirect)
400	Bad Request	Invalid request / syntax error
401 / 403	Unauthorized	Authentication failed / Access denied
404	Not Found	Invalid resource was requested
409	Conflict	Conflict was detected, e.g. duplicated email
500 / 503	Server Error	Internal server error / Service unavailable

Figura: Códigos de status HTTP.

HTTP - Requisição e Resposta

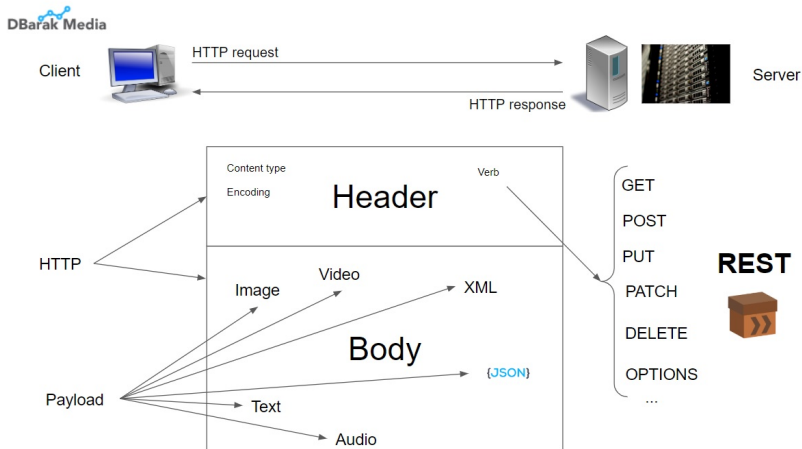


Figura: Ilustração do funcionamento do protocolo HTTP.

HTTP - Requisição e Resposta

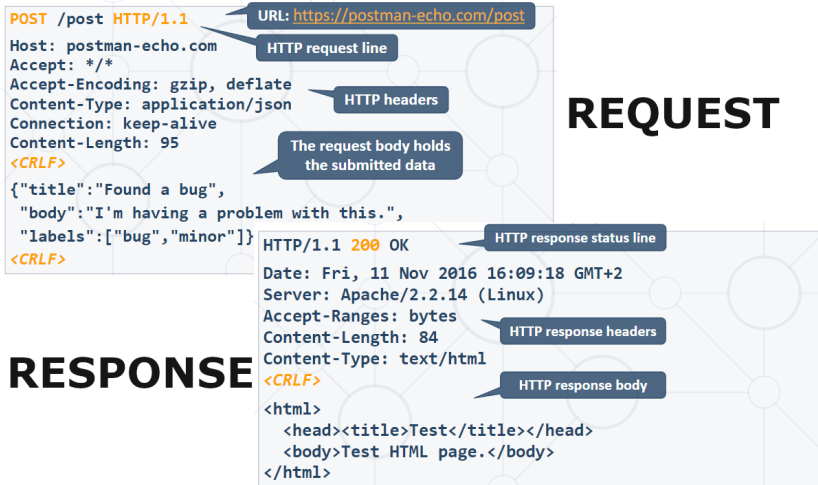


Figura: Requisição e resposta HTTP.

- **HTTPS** - *Hypertext Transfer Protocol* **Secure**.
- Os dados são criptografados antes de serem enviados. Evita que os dados sejam modificados durante a transmissão.
- Utiliza o protocolo **SSL** - *Secure Sockets Layer* ou **TLS** - *Transport Layer Security*.
- O certificado SSL/TLS (criptografia assimétrica) é emitido por uma **AC** - *Autoridade Certificadora* e instalado no servidor ¹.

O que HTTPS não faz?

- Não garante que o servidor é confiável.
- Não garante que o servidor não foi invadido.
- Não garante que o servidor não está enviando dados para terceiros.
- Não oculta o endereço do site que está sendo acessado.
- Não esconde sua identidade e localização.
- Não evita que o usuário pegue vírus. Não é um filtro de conteúdo.

¹  Let's Encrypt

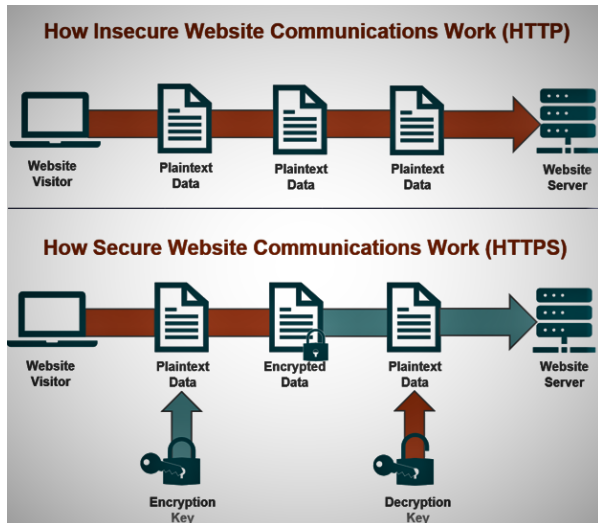



Figura: Comparação entre o funcionamento do protocolo HTTP e HTTPS.

HTTPS - Consultando o Certificado

exemplos > https >  main.py > ...

```
1 import ssl
2 import urllib.request
3 parsed_url = urllib.parse.urlparse("https://videira.ifc.edu.br")
4 print(ssl.get_server_certificate((parsed_url.hostname, 443)))
```

• (.venv) fabricio@DESKTOP-MG3SLC3:~/Projetos/Desenvolvimento-Web-II/exemplos/https\$ python main.py

-----BEGIN CERTIFICATE-----

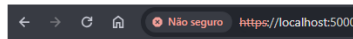
MIIG4zCCBcugAwIBAgIOMB6vLJx3jJAh0W06QMA0GCSqGSIb3DQEBCwUAMGQxCzAJBgNVBAYTAKJSMTEwLWYDVQQKEyhSZWR1IE5hY21vbmFsIGR1IEVuc21ubyB1IFB1c3F1aXNhIC0gUk5QM5IwIAYDVQQDEx1ST1AgSUNQRWR1IE9wIFNTTCBDQSAyMDE5MB4XDTEzMDEzMDE5MDEwM1oXDTE0MDMwMjE5MDEwM1owZz8xCzAJBgNVBAYTAKJSMRcwFQYDVQQIEw5TYW50YSBDYXRhcm1uYTERMA8GA1UEBxMIQmx1bWVudXUxRzBFBgNVBAOTPk1uc3RpdH0byBGZWR1cmFsIGR1IEVkdWlnY2FvIENpZW5jaWVhZGZSB1ZmNub2xvZ21hIENhdGFyaW51bnNlMRswGQYDVQQDExJ2aWR1aXJhLm1mYy51ZHUuYnIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCdD5KbDvDqxSoDyeuEPDktSKIzE11s3jBh2KUEkpg3jPYEPry158Aqp61MMXxIK4gk5gCo+T5Evd1z5yCSNNHJReRictREPhEf7nJNKn4LjrN0xyTZViGjWJ1AFG/JBCX5r4qineCDPNILiC+VVBtBcsyEBy0WLFtQArw00a4msiLnj5DtcwFeIYz1fJOXDSa8+c1IczZA+cX64Ps3ivfKBI8Qnt03p016s7z9iivXGB+A6NBNrHx4bqpIbW0HjC6kZPD5QCcsnqkoZ1ZX4JCyAgsmRHy/NegN91MHVs5Gzj/JODxSYhPZD3Y+205TA3dXnvpW3J5Pbe5Lo

Figura: Consultando o certificado SSL/TLS.

HTTPS - Criando e consumindo meu próprio certificado

```
(.venv) fabricio@DESKTOP-MG3SLC3:~/Projetos/Desenvolvimento-Web-II/exemplos/https$ openssl genpkey -algorithm RSA -out privada.pem
(.venv) fabricio@DESKTOP-MG3SLC3:~/Projetos/Desenvolvimento-Web-II/exemplos/https$ openssl req -new -x509 -key privada.pem -out publica.pem -days 7
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:BR
State or Province Name (full name) [Some-State]:SC
Locality Name (eg, city) []:Videira
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IFC
Organizational Unit Name (eg, section) []:Ensino
Common Name (e.g. server FQDN or YOUR name) []:Fabricio
Email Address []:fabricio.bizotto@ifc.edu.br
```

```
exemplos > https > main.py > ...
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route("/")
6 def index():
7     return "Hello, World!"
8
9 if __name__ == "__main__":
10 | app.run(debug=True, ssl_context=('publica.pem', 'privada.pem'))
..
```



Hello, World!

Figura: Criando e consumindo meu próprio certificado.

Protocolos de Comunicação

Web Socket

Web Socket Protocol

Antes de Websocket

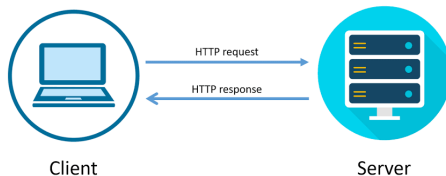


Figura: Fluxo normal de requisição e resposta HTTP.

- **Pooling** - O cliente faz requisições ao servidor em intervalos regulares. O servidor responde, mesmo que não tenha novos dados.
- **Long polling** - O cliente faz uma requisição e o servidor mantém a conexão aberta até que tenha novos dados. O servidor responde quando tiver novos dados ou quando o tempo limite for atingido.
- Essas abordagens são ineficientes e consomem muitos recursos. Cada requisição/resposta gera um *overhead* de comunicação.

HTTP - Estratégia de Pooling

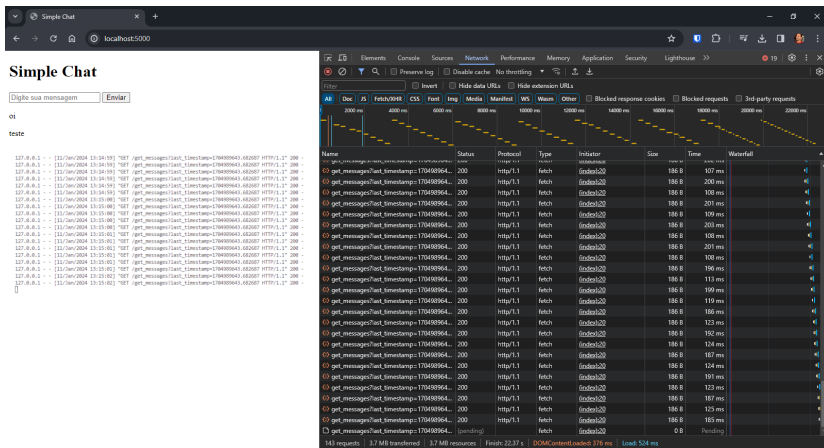


Figura: HTTP - Pooling.

- É um protocolo de comunicação **full-duplex**, ou seja, permite que o cliente e o servidor enviem dados simultaneamente.
- Utiliza o protocolo **TCP**, normalmente na porta **443**.
- É um protocolo **stateful**, ou seja, mantém estado entre requisições.
- A comunicação é iniciada com um **handshake** HTTP para garantir ambas as partes concordam em estabelecer uma conexão bidirecional persistente.
- Muito usado em aplicações que precisam de comunicação em tempo real, tais como:
 - Chat
 - Jogos
 - Streaming
 - Gráfico de ações em tempo real
 - etc

Onde não usar?

Não é recomendado usar em situações onde a comunicação é pontual ou para buscar dados antigos. Nesses casos, o HTTP é mais adequado.

Web Socket - Exemplo de uso

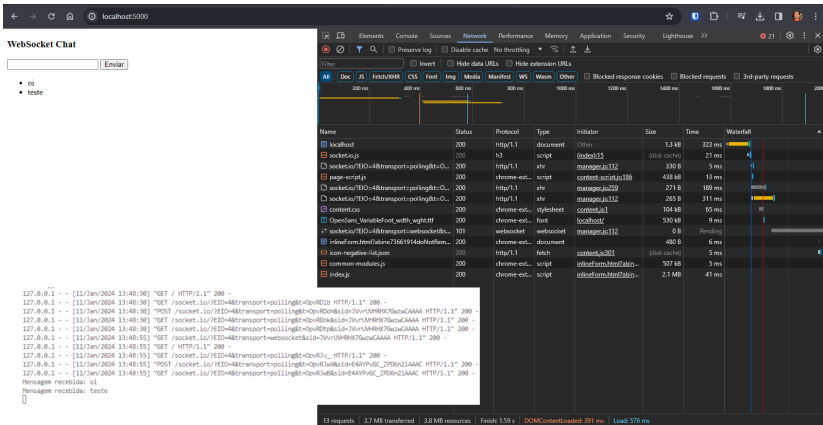


Figura: Exemplo de uso do protocolo WebSocket.

Comparação entre HTTP e WebSocket

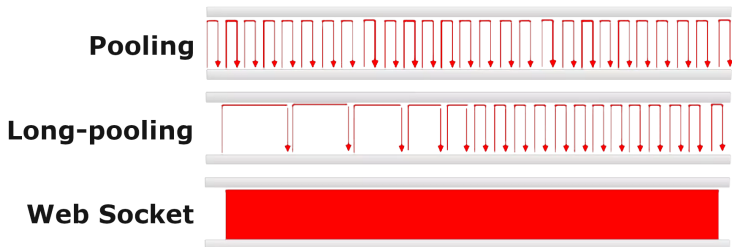






Figura: Comparação entre Pooling, Long Polling e WebSocket.

Chrome	Edge	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS	Samsung Internet	Opera Mini	Opera Mobile	UC Browser for Android	Android Browser	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
		3.1-4	2-3.6													
4-14		5-5.1	4-5	10.1			3.2-4.1									
15		6-6.1	6-10	11.5	6-9		4.2-5.1			12			2.1-4.3			
16-119	12-119	7-17.1	11-120	12.1-105	10		6-17.1	4-22		12.1		4.4-4.4.4				2.5
120	120	17.2	121	106	11	120	17.2	23	all	73	15.5	120	119	13.1	13.18	3.1
121-123	121	17.3-TP	122-124				17.3									


Figura: Can I use: - Web Socket Browser Support.

Material Complementar

-  Web Socket - Demo
-  HTTP2: magia com o novo protocolo – Hipsters (2016)
-  Web3 vale o hype? – Hipsters (2023)
-  Desconstruindo a Web: As tecnologias por trás de uma requisição

Lista de Experimentos

Escolha um dos experimentos abaixo para realizar:

 Lista de Experimentos