

Os protocolos HTTP e HTTPS e seus principais métodos

Prof.º Msc. Gustavo Nunes Rocha

Objetivos da Aula

- Explicar o funcionamento do HTTP e do HTTPS, destacando suas principais diferenças.
- Apresentar os principais métodos de requisição HTTP (GET, POST, PUT, DELETE, etc.).
- Demonstrar como o HTTPS protege a comunicação web através de criptografia (SSL/TLS).
- Capacitar os alunos a identificar e utilizar corretamente os métodos HTTP em aplicações web.

Sumário

- Introdução aos Protocolos HTTP e HTTPS;
- Histórico das versões HTTP: HTTP/1, 2 e 3;
- HTTPS e suas características;
- Introdução aos métodos básicos do HTTP;
- Recomendações de leitura;
- Referências Bibliográficas.

Introdução aos Protocolos HTTP e HTTPS

O que são protocolos HTTP e HTTPS?

Regras que permitem a comunicação entre dispositivos em redes.

- *HTTP (Hypertext Transfer Protocol)*: Protocolo de comunicação sem estado.
- *HTTPS (Hypertext Transfer Protocol Secure)*: Protocolo seguro, usa SSL/TLS para criptografar os dados.

“O HTTP é um protocolo sem estado, em que cada requisição é independente das anteriores.” (KUROSE; ROSS, 2013)

O nascimento do protocolo HTTP

- 1991: O HTTP 0.9 foi criado por Tim Berners-Lee para a transferência de hipertexto.
- 1996: Surgimento do HTTP/1.0, a primeira versão amplamente utilizada, suportando mais tipos de conteúdo além de texto.

“O HTTP foi inicialmente projetado para transferir documentos simples de hipertexto, com uma estrutura muito básica.”(TANENBAUM; WETHERALL, 2021)

HTTP/1.1

- 1997: HTTP/1.1 introduziu várias melhorias:
 - Suporte a conexões persistentes.
 - Melhor gestão de cache.
 - Requisições "pipelining" (envio de várias requisições sem esperar pelas respostas).

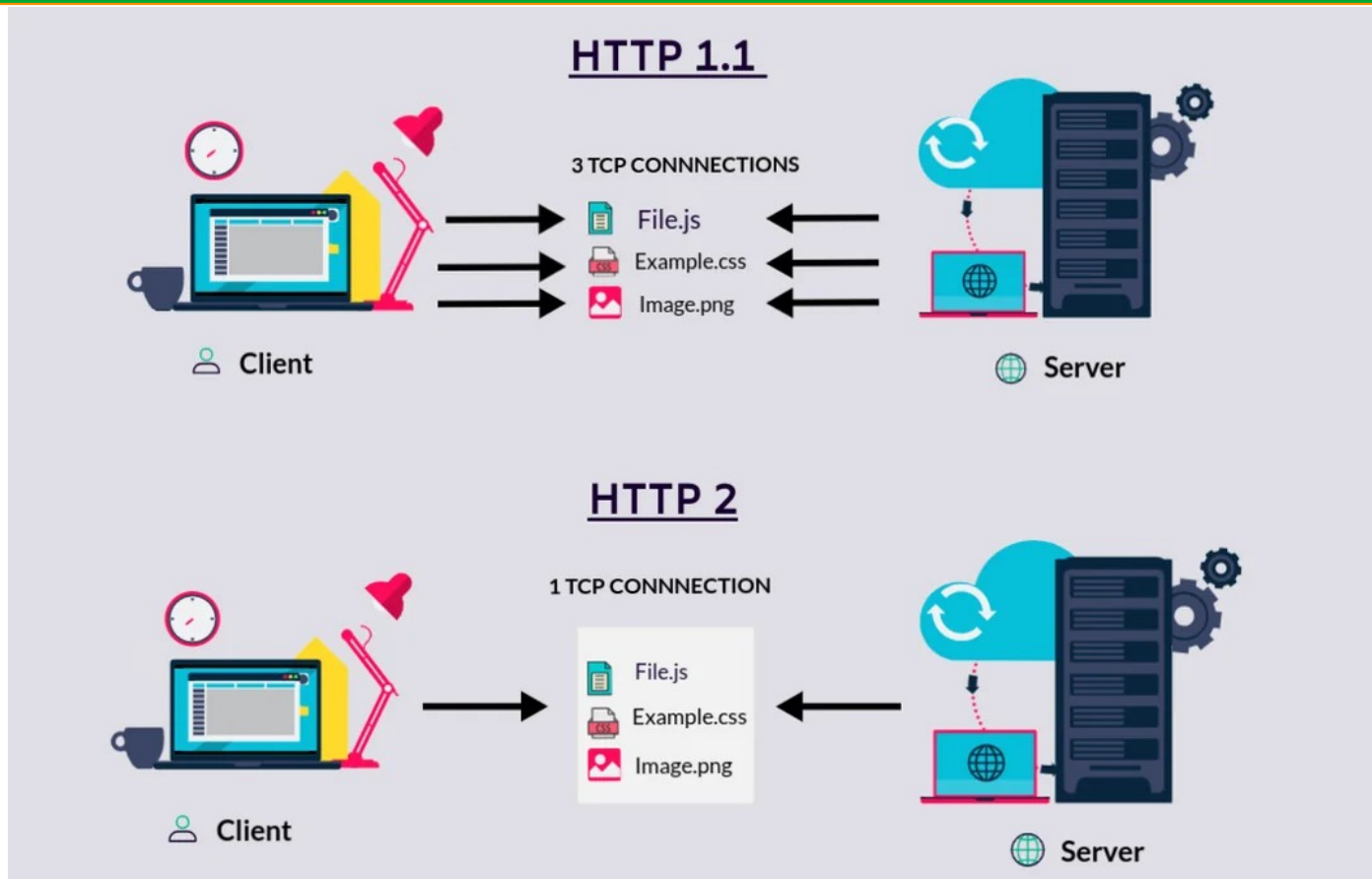
“O HTTP/1.1 tornou a web mais eficiente e rápida, permitindo que múltiplos recursos fossem carregados com menos demora.” (KUROSE; ROSS, 2013)

HTTP/2

- 2015: HTTP/2 foi lançado, trazendo mudanças significativas:
 - Multiplexação: várias requisições simultâneas pela mesma conexão.
 - Compressão de Cabeçalhos: melhora na eficiência do tráfego de dados.
 - Suporte nativo para HTTPS.

“HTTP/2 foi um divisor de águas, reduzindo a latência e acelerando o carregamento de páginas web.” (TANENBAUM; WETHERALL, 2021)

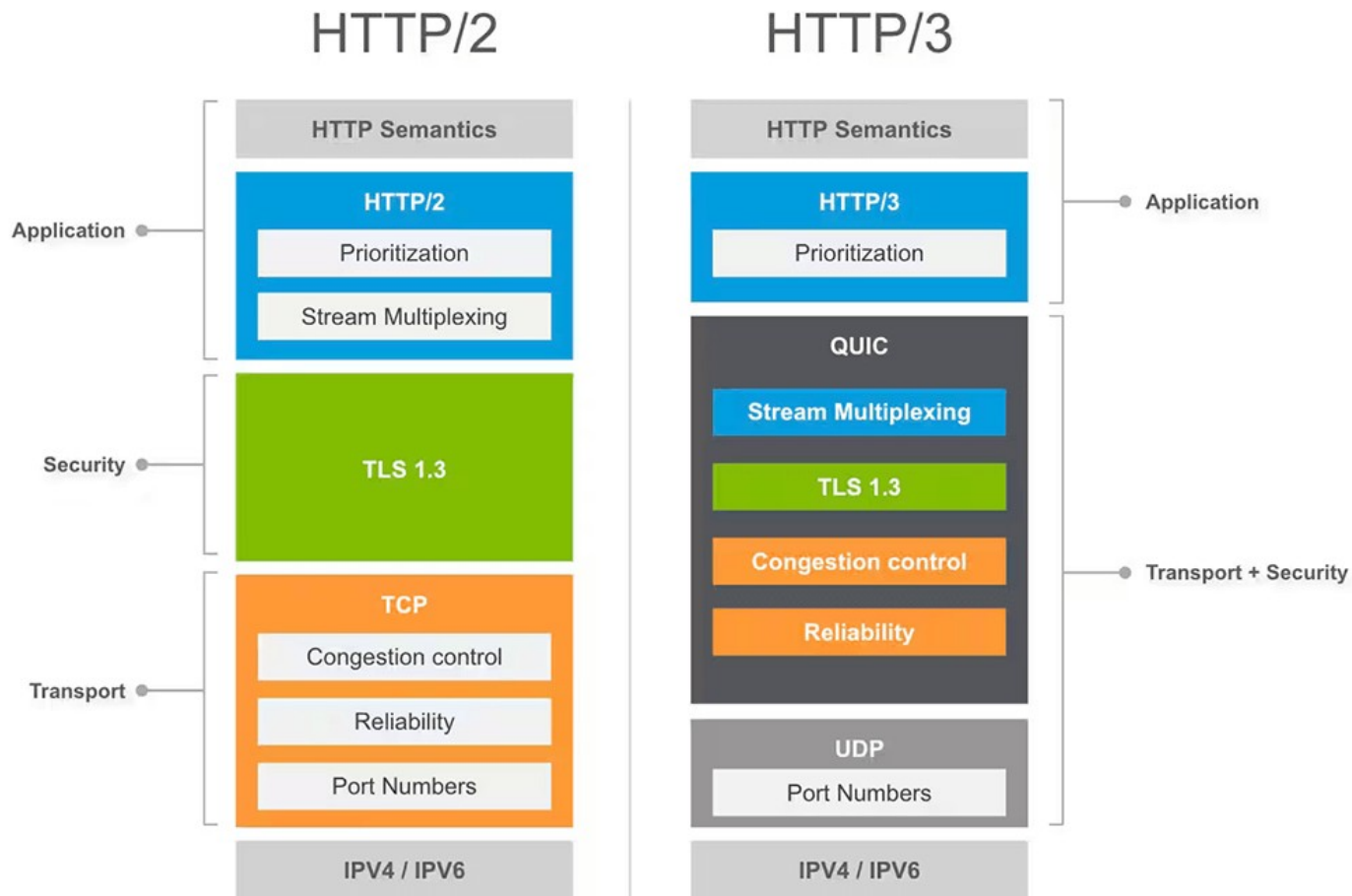
HTTP 1.1 X HTTP/2



HTTP/3

- 2022: O HTTP/3 foi oficialmente lançado em 2022, introduzindo melhorias significativas em relação ao seu predecessor, o HTTP/2.
 - Baseado no Protocolo QUIC: Opera sobre UDP e reduz o número de "handshakes" necessários para estabelecer uma conexão, o tempo de latência é significativamente diminuído.
 - Multiplexação Sem Bloqueio: Permite múltiplas requisições simultâneas em uma única conexão sem que uma requisição lenta atrase as demais, eliminando o problema de "head-of-line blocking" do HTTP/2.
 - Segurança Integrada: O HTTP/3 possui criptografia TLS 1.3 embutida, garantindo que todos os dados transmitidos sejam seguros desde o início.

HTTP/2 X HTTP/3



HTTPS – Hyper Text Protocol Secure

Por que HTTPS?

Com o crescimento do comércio eletrônico e da troca de informações sensíveis, surgiu a necessidade de um protocolo seguro.

- 1994: O SSL (Secure Sockets Layer) foi introduzido pela Netscape, criando a base para o HTTPS.
- 2000: O TLS (Transport Layer Security) substituiu o SSL, trazendo mais segurança e eficiência.

“O HTTPS foi desenvolvido para impedir a interceptação de dados e garantir a autenticidade dos sites.” (KUROSE; ROSS, 2013)

HTTPS – Hyper Text Protocol Secure

HTTPS utiliza criptografia SSL/TLS para proteger os dados.

Processo de Criptografia:

- Chave pública do servidor criptografa os dados.
- Chave privada do servidor decripta os dados recebidos.

Certificados Digitais: Emitidos por uma Autoridade Certificadora (CA) para garantir a autenticidade do site.

“Os certificados digitais são a base da confiança no HTTPS, permitindo que os usuários verifiquem a legitimidade dos sites.” (KUROSE; ROSS, 2013)

SSL vs. TLS

Critério	SSL	TLS	Vantagens do TLS
Desenvolvimento	Netscape, anos 90	IETF, sucessor do SSL	Evolução com melhorias de segurança
Última versão	SSL 3.0 (1996) – descontinuado	TLS 1.3 (2018) – padrão atual	TLS 1.3 é mais seguro e eficiente
Handshake	Lento e menos seguro	Rápido e seguro	Menos latência e maior segurança
Compatibilidade	Não é compatível com TLS	Compatível com versões anteriores	Flexibilidade para sistemas antigos
Segurança	Vulnerável a vários ataques	Corrige vulnerabilidades do SSL	Melhor resistência a ataques
Extensões	Suporte limitado	Suporta extensões modernas	Melhora a flexibilidade em ambientes complexos
Performance	Lento devido a negociações	Mais rápido e eficiente	Melhor performance com menos round-trips


FONTE: Elaboração própria

Protocolos HTTP x HTTPS

HTTP – Porta Padrão 80:

- Comunicação em texto puro.
- Vulnerável a ataques de "man-in-the-middle" e roubo de dados.

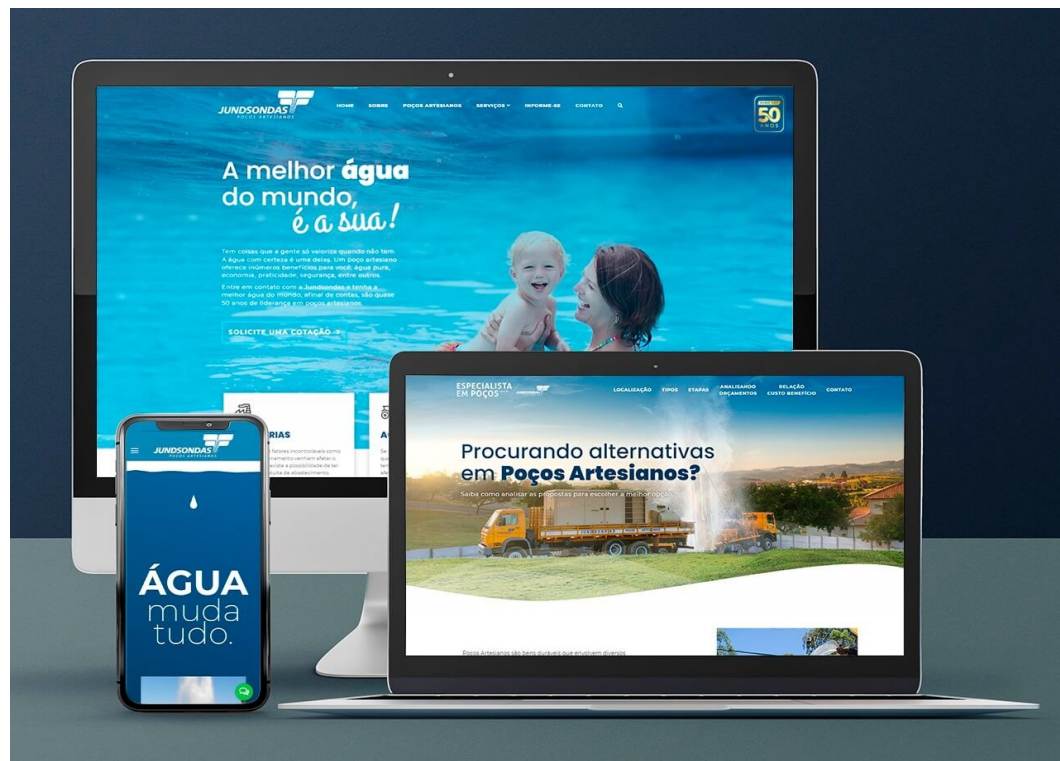
HTTPS - Porta Padrão 443:

- Criptografia via SSL/TLS. 
- Protege a confidencialidade, integridade e autenticidade dos dados.
- Essencial para transações financeiras e informações pessoais.

“O HTTPS se tornou indispensável para qualquer site que lide com informações sensíveis ou privadas.” (SILVA, 2015)

Onde usar o protocolo HTTP?

- *Sites Informativos*
- *Conteúdo Público*
- *Ambientes de Desenvolvimento*
- *Recursos Estáticos*

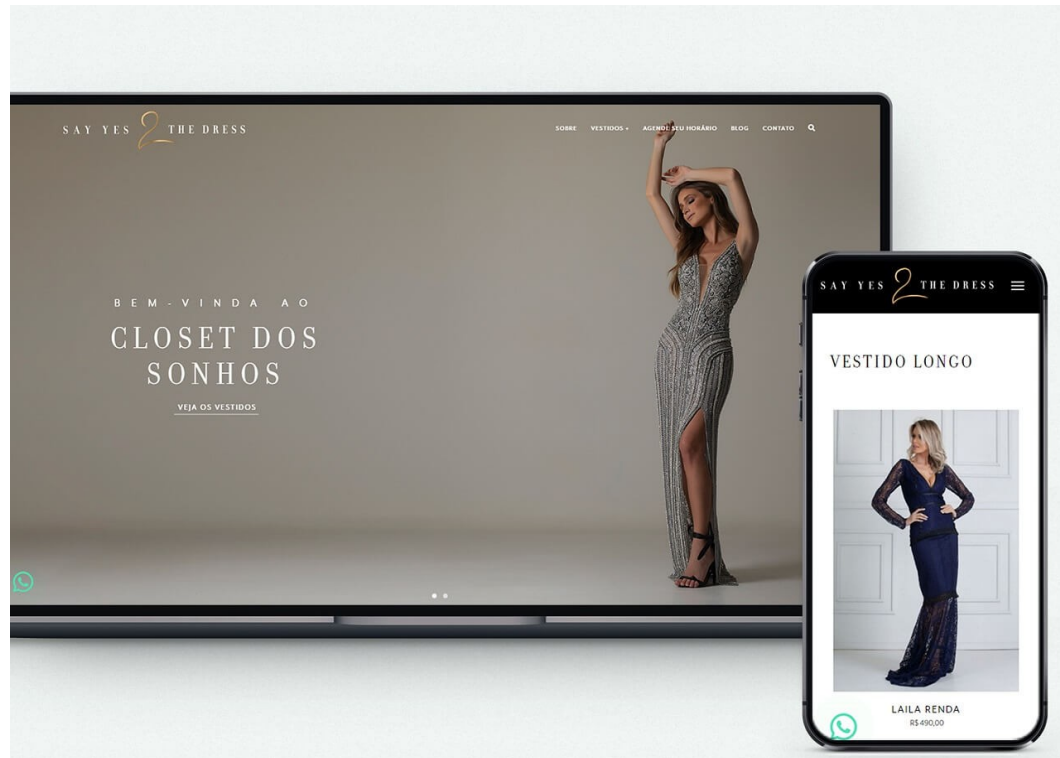


Fonte: <https://highsales.digital/blog/tipos-de-sites>

Onde usar o protocolo HTTPS?

- *Sites de Comércio Eletrônico;*
- *Tráfego de Dados Sensíveis.*

“Com o aumento das ameaças online, o uso de HTTPS é fundamental para proteger a privacidade e a segurança dos usuários na web.” (BASHAM; SIERRA, 2009)



Fonte: <https://highsales.digital/blog/tipos-de-sites>

Métodos HTTP – As Operações Fundamentais

O que são métodos HTTP?

Os métodos HTTP definem as ações que um cliente pode solicitar de um servidor. Cada método possui características e propósitos específicos.

Principais métodos:

- GET
- POST
- PUT
- DELETE
- HEAD
- OPTIONS
- PATCH

Métodos HTTP – MÉTODO GET

Solicita dados de um recurso específico. O método GET não altera o estado do servidor, apenas recupera informações.

Usabilidade:

- Consultar APIs para obter informações.
- Recuperar dados de páginas web.

Características principais:

- Idempotente: Repetir a mesma solicitação não altera o estado do recurso.
- Cacheáveis: Respostas podem ser armazenadas em cache.

EXEMPLO:

```
GET /api/users HTTP/1.1
```

```
Host: example.com
```

```
Accept: application/json
```

Descrição: Solicitação de informações sobre o usuário com o ID 123. O esperado é que o servidor responda com os dados desse usuário.

Métodos HTTP – MÉTODO POST

Envia dados ao servidor para criar ou atualizar um recurso.

Usabilidade:

- Envio de formulários.
- Carregar arquivos em servidores.

Características principais:

- Idempotente: Repetir a mesma solicitação não altera o estado do recurso.
- Cacheável: Respostas podem ser armazenadas em cache.

EXEMPLO:

```
POST /api/usuarios
```

```
Content-Type: application/json
```

```
{  
  "nome": "João",  
  "email": "joao@example.com"  
}
```

Descrição: Envia uma nova entrada de usuário ao servidor. Os dados (nome e email) são enviados no corpo da solicitação. O servidor pode criar um novo usuário e retornar a informação criada, como um ID gerado.

Métodos HTTP – MÉTODO PUT

Atualiza ou substitui um recurso existente no servidor.

Usabilidade:

- Atualizar informações em bancos de dados.

Características principais:

- Idempotente: Repetir a solicitação resulta no mesmo estado do recurso.

EXEMPLO:

```
PUT /api/users/1
```

```
Host: example.com
```

```
Content-Type: application/json
```

```
{
```

```
  "name": "Alice Smith",
```

```
  "email": "alice.smith@example.com"
```

```
}
```

Descrição: Esta solicitação é feita para atualizar as informações do usuário com ID 1. Se o usuário existir, suas informações são atualizadas; caso contrário, o servidor pode retornar um erro.

Métodos HTTP – MÉTODO DELETE

Remove um recurso específico do servidor.

Usabilidade:

- Excluir informações em bancos de dados.

Características principais:

- Idempotente: Tentar deletar um recurso que já foi removido não altera o estado.

EXEMPLO:

DELETE /api/users/1

Host: example.com

Descrição: Esta solicitação é feita para excluir o usuário com ID 1.

Se o usuário existir, o servidor remove o registro e pode retornar uma mensagem de sucesso.

Se o usuário já tiver sido excluído, o servidor pode retornar um status indicando que o recurso não foi encontrado.

Métodos HTTP – MÉTODO HEAD

Solicita apenas os cabeçalhos da resposta, sem o corpo.

Usabilidade:

- Verificar a existência de um recurso ou obter metadados.

Características principais:

- Idempotente: A mesma solicitação não altera o estado do recurso.
- Verificação rápida: Útil para verificar a existência de um recurso sem transferir dados adicionais. Retorna os mesmos cabeçalhos que o método GET, mas sem o corpo da resposta.

EXEMPLO:

```
HEAD /api/users
```

```
Host: example.com
```

Descrição: O cliente solicita os cabeçalhos do recurso de usuários, sem querer o corpo da resposta.

Métodos HTTP – MÉTODO OPTIONS

Retorna os métodos HTTP suportados pelo servidor para um recurso específico.

Usabilidade:

- Descobrir opções e requisitos de comunicação.

Características principais:

- Idempotente: A mesma solicitação não altera o estado do recurso.
- Informativo: Útil para descobrir quais métodos estão disponíveis para um recurso. O servidor pode fornecer informações adicionais, como requisitos de autenticação.

EXEMPLO:

```
OPTIONS /api/users
```

```
Host: example.com
```

Descrição: O cliente solicita informações sobre os métodos que pode usar para interagir com o recurso de usuários.

Métodos HTTP – MÉTODO PATCH

Aplica modificações parciais a um recurso existente.

Usabilidade:

- Atualizar partes específicas de um recurso.

Características principais:

- Idempotente: A mesma solicitação resulta na mesma modificação aplicada ao recurso.
- Parcialidade: Permite atualizar apenas os campos que precisam ser alterados, ao invés de enviar todos os dados do recurso.
- Versatilidade: Útil para modificações frequentes em recursos que não exigem reenvio de dados completos.

EXEMPLO:

```
PATCH /api/users HTTP/1.1
```

```
Host: example.com
```

```
Content-Type: application/json
```

```
{  
  "email": "new.email@example.com"  
}
```

Descrição: O cliente está solicitando uma atualização parcial do usuário com ID 1, mudando apenas o e-mail. Essa abordagem é eficiente porque não requer o envio de todos os dados do recurso.

Recomendações de leitura – Livros

- ALBUQUERQUE, E., Fernando. TCP/IP: Internet Programação de Sistemas Distribuídos HTML, JavaScript e Java. Rio de Janeiro: Axcel, 2001.
- BASHAM, Bryan; SIERRA, Kathy; BATES, Bert. Use a Cabeça! Servlets & JSP. 2. ed. Rio de Janeiro: Alta Books, 2011. Acesso em: 13 out. 2024.
- GEARY, David M. Dominando JavaServer Pages Avançado. São Paulo: Ciência Moderna, 2002.
- GONÇALVES, Edson. Desenvolvendo Aplicações Web com JSP, Servlets, Javasever Faces, Hibernate, EJB 3, Persistência e AJAX. Rio de Janeiro: Ciência Moderna, 2007.
- HAYWOOD, Dan; LAW, Debbie; LONGSHAW, Andy; ROXBURGH, Peter; BOND, Martin. Aprenda J2EE em 21 dias. São Paulo: Pearson Educação do Brasil, 2003.

Recomendações de leitura - Internet

Protocolo HTTP e HTTPS:

- <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>
- <https://aws.amazon.com/pt/compare/the-difference-between-https-and-http/>

Métodos HTTP:

- <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>
- <https://www.dio.me/articles/principais-metodos-http-um-guia-rapido>

Lista de Exercícios – Para Casa

A lista de exercícios, juntamente com essa apresentação, já está disponível no Github da disciplina:

- **Link de acesso para o github da disciplina para aula de hoje:**

https://github.com/gnrochabr/prog_web/tree/main/Protocolo_HTTP

As correções serão realizadas na próxima aula.

Referências Bibliográficas

- BASHAM, B.; SIERRA, K. Use a Cabeça! Servlets & JSP. 2ª edição. Alta Books, 2009.
- KUROSE, J.; ROSS, K. Redes de Computadores e a Internet: uma abordagem top-down. 6ª edição. Pearson, 2013.
- SILVA, M. S. Fundamentos de HTML5 e CSS3. 1ª edição. Novatec Editora, 2015.
- TANENBAUM, A.; WETHERALL, D. Redes de Computadores. 6ª edição. Pearson, 2021.