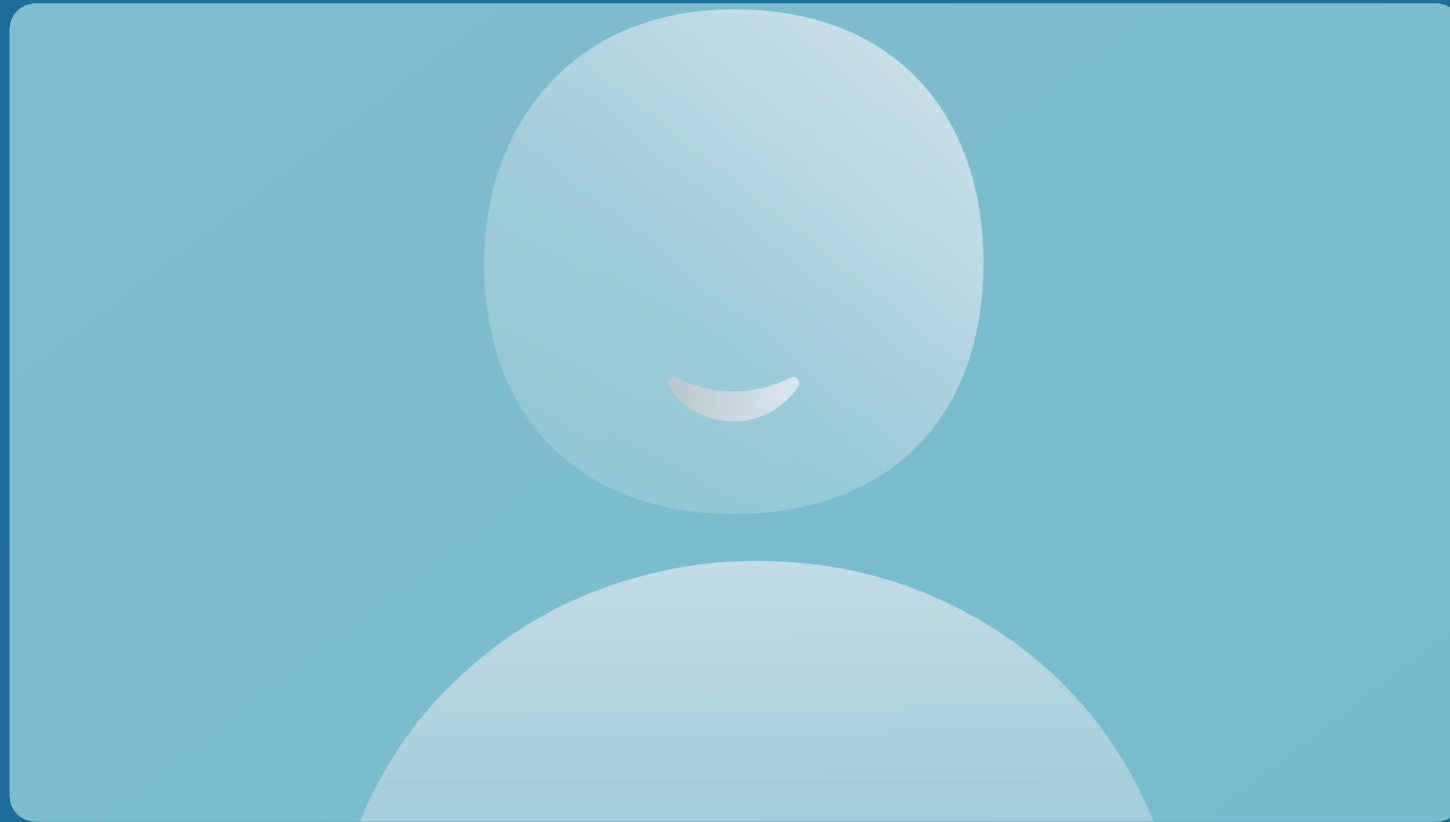
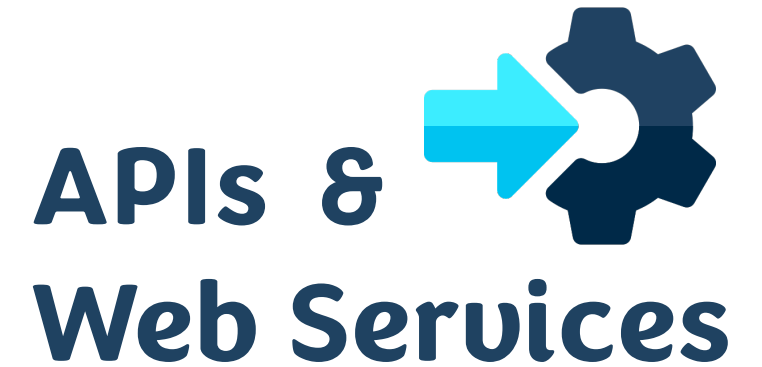


APIs e Web Services



Tópicos

- Unidade 1 - Introdução
 - Introdução ao Mundo das APIs
 - Protocolo HTTP
- Unidade 2 - Web APIs
 - Aspectos arquiteturais de APIs
 - Estilos: SOAP, REST, GraphQL, Webhooks e Websockets
- Unidade 3 - Segurança em APIs
 - Fundamentos de Segurança
 - Autenticação no Protocolo HTTP
 - Tecnologias e Frameworks: JWT, OAuth e CORS
- Unidade 4 - Gerenciamento de APIs
 - API Lifecycle Management (ALM)
 - Arquitetura de Microserviços e API Gateway
 - Boas práticas no desenvolvimento de APIs



Informações Gerais

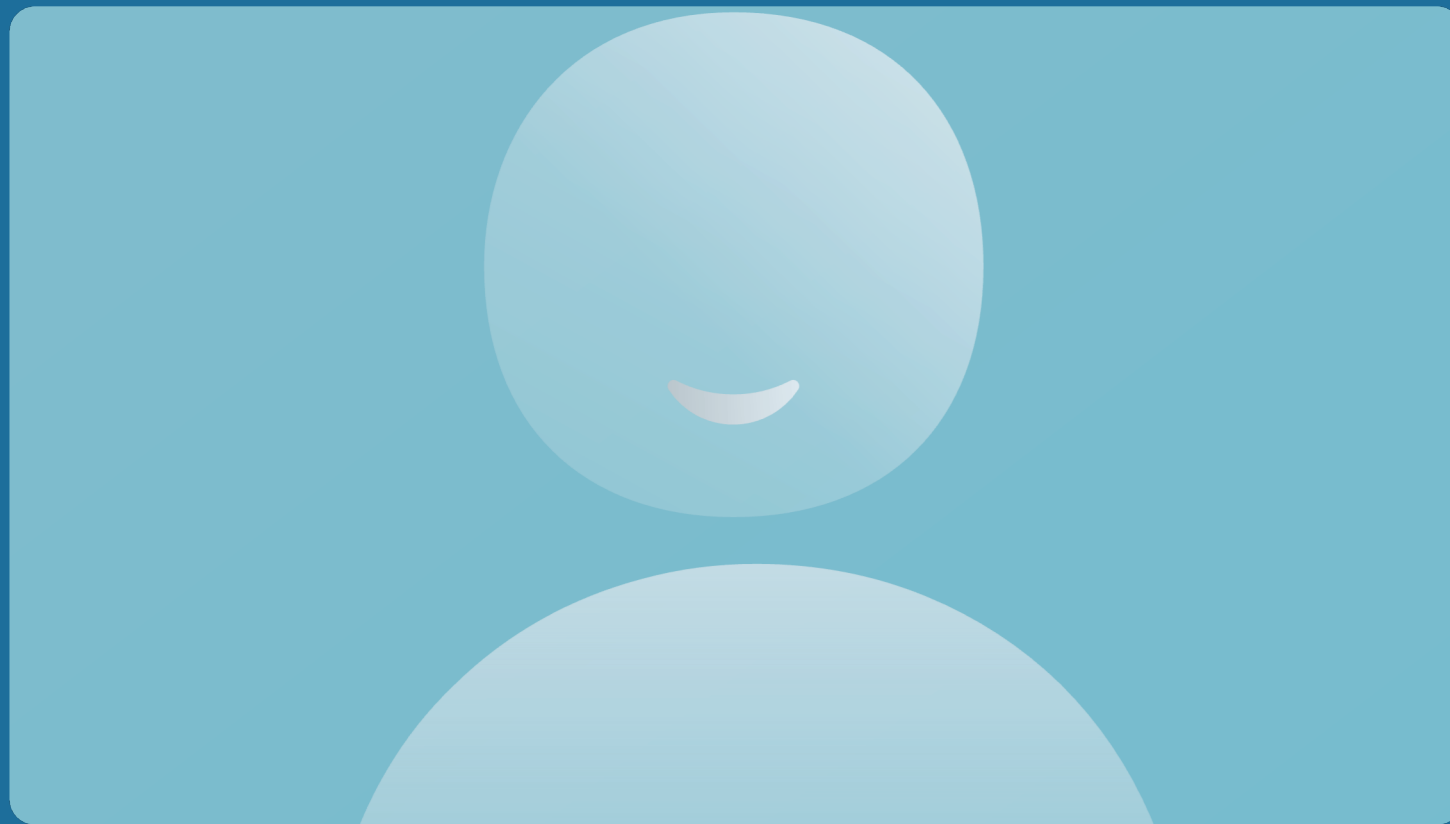
Ementa

Evolução das APIs. Gestão do ciclo de vida das APIs. Melhores práticas no projeto de API ferramentas para documentação de APIs. Mecanismos de segurança: autenticação, vulnerabilidades. Abordagens arquiteturais de APIs: RESTful, GraphQL, WebSockets, WebF Streaming.

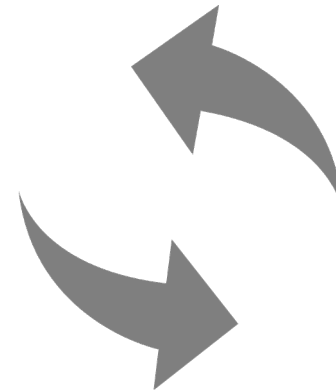
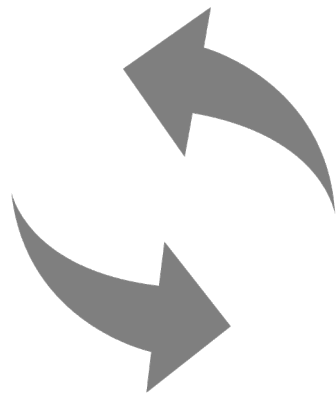
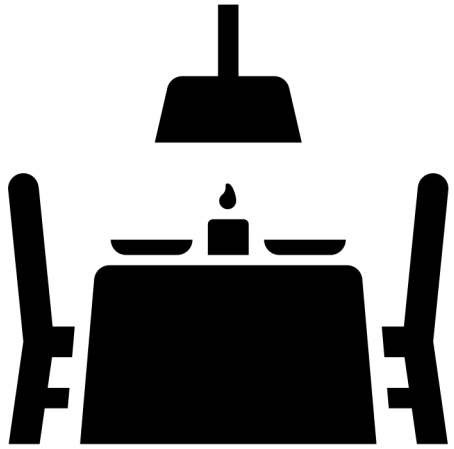
Bibliografia

- [Designing Web APIs](#). Brenda Jin, Saurabh Sahni, Amir Shevat. O'Reilly Media, Inc. (2018)
- [Mastering API Architecture](#). James Gough, Daniel Bryant, Matthew Auburn. O'Reilly Media, Inc. (2022)
- [API Design Patterns](#). John J. (JJ) Geewax. Manning Publications (2021)
- [Designing APIs with Swagger and OpenAPI](#). Josh Ponelat, Lukas Rosenstock. Manning Publications (2022)
- [Microservice APIs](#). Jose Haro. Manning Publications (2023)

Introdução ao mundo das APIs

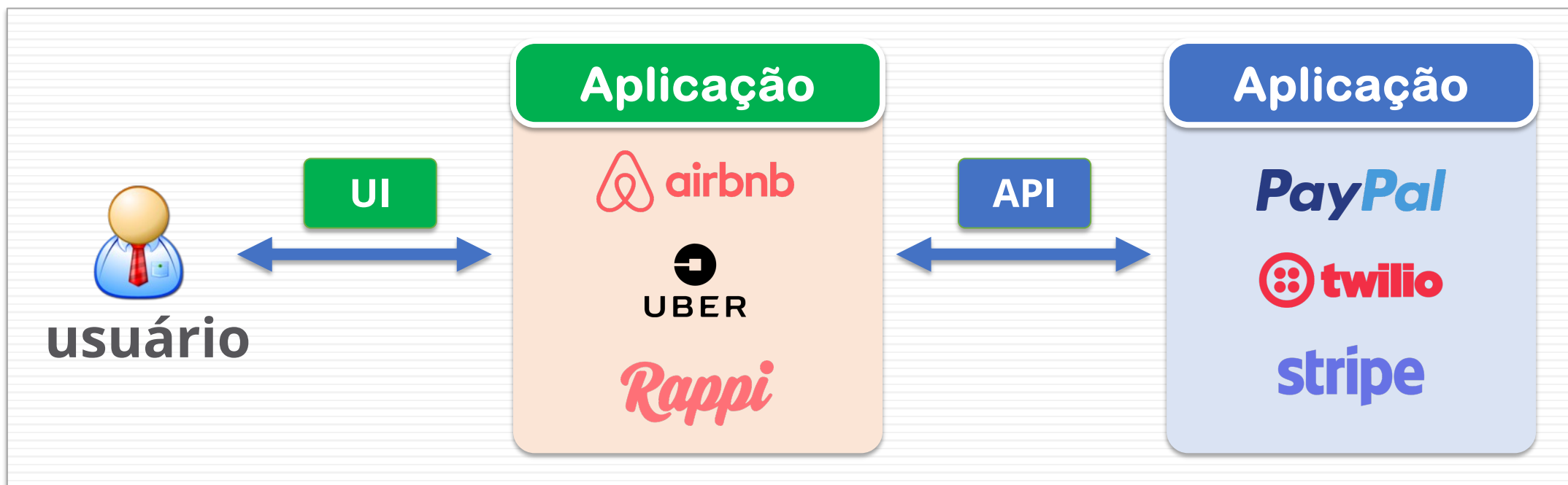


Introdução ao mundo das APIs



Introdução ao mundo das APIs

Application Programming Interface (API) é uma forma de permitir a integração de uma aplicação com outra.

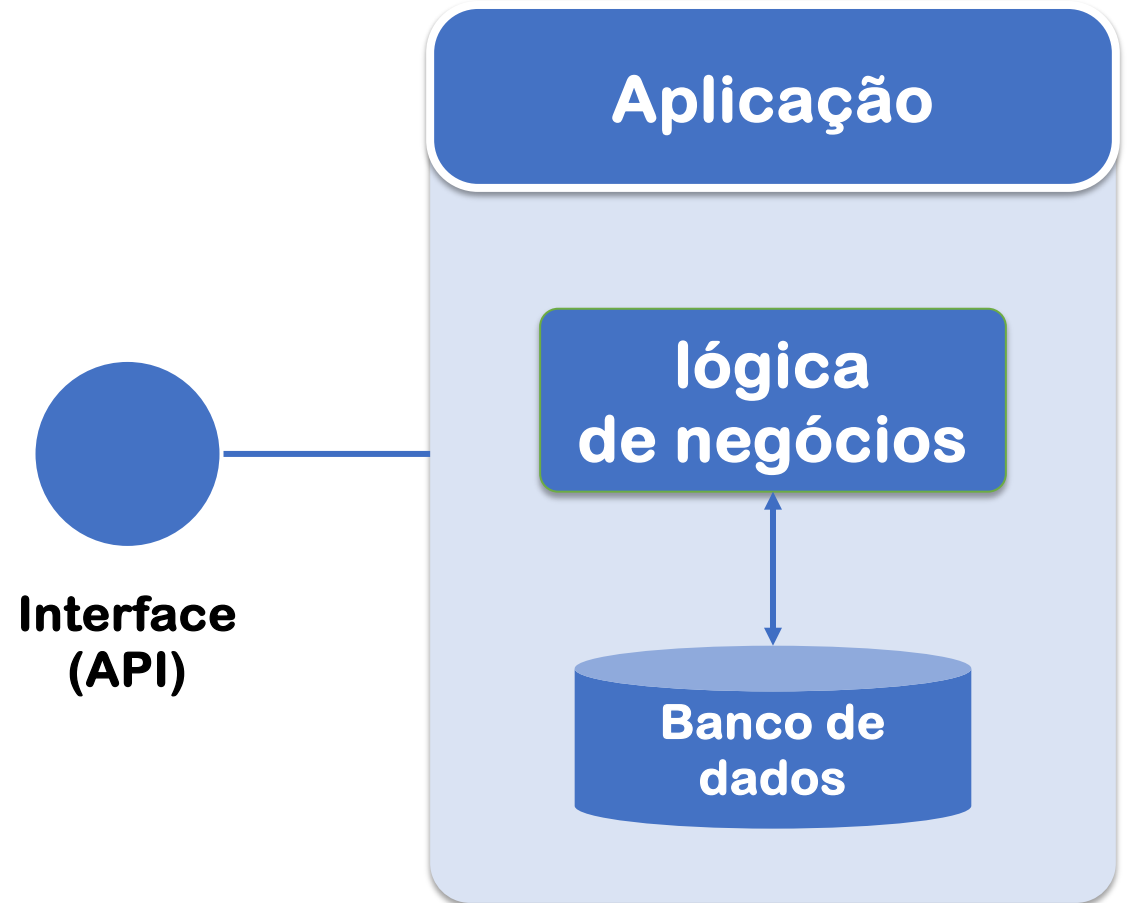


Introdução ao mundo das APIs

Na definição de APIs, o principal termo é **interface**.

Podemos pensar a interface como os **termos contratuais** que uma aplicação oferece para que outras aplicações possam se comunicar de **forma programática**.

Em síntese permitem a integração entre softwares distintos.



Introdução ao mundo das APIs

Web Services

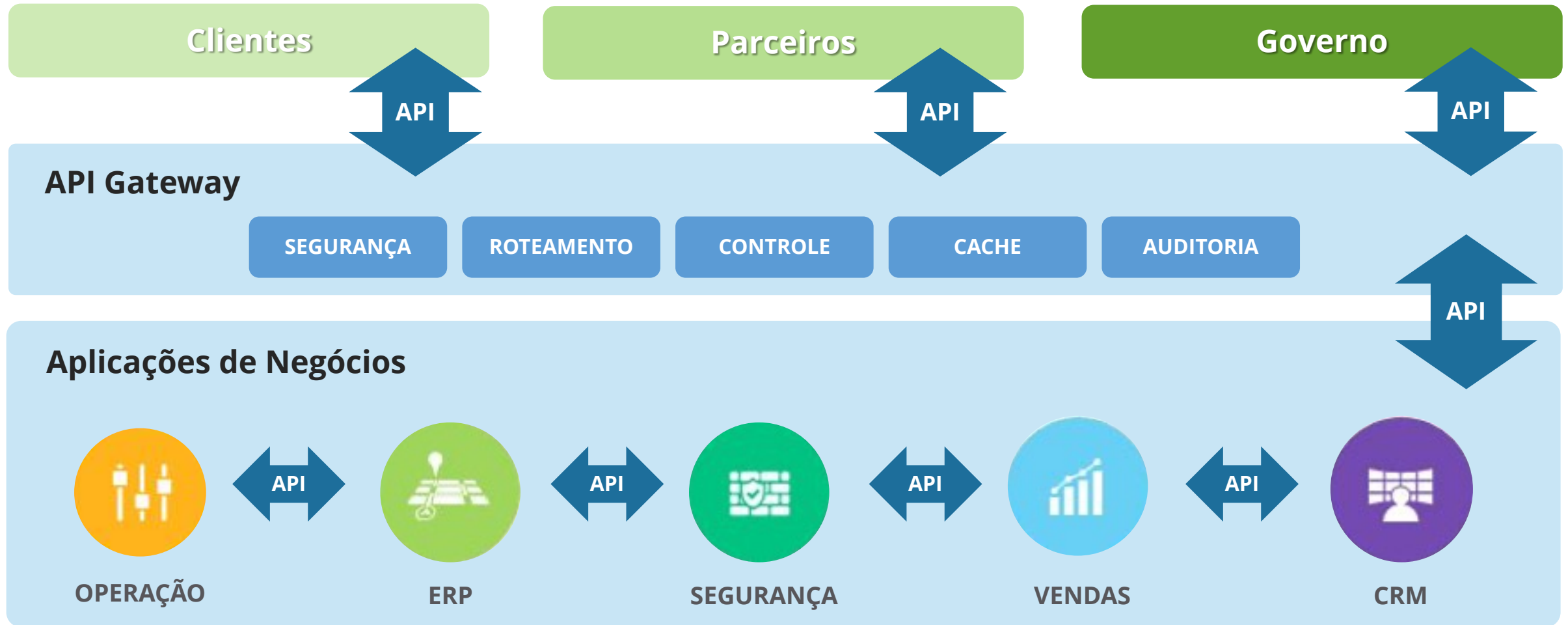
“Um **sistema de software** projetado para suportar **comunicação interoperável, máquina-a-máquina** através de uma rede.”

W3C – World Wide Web Consortium

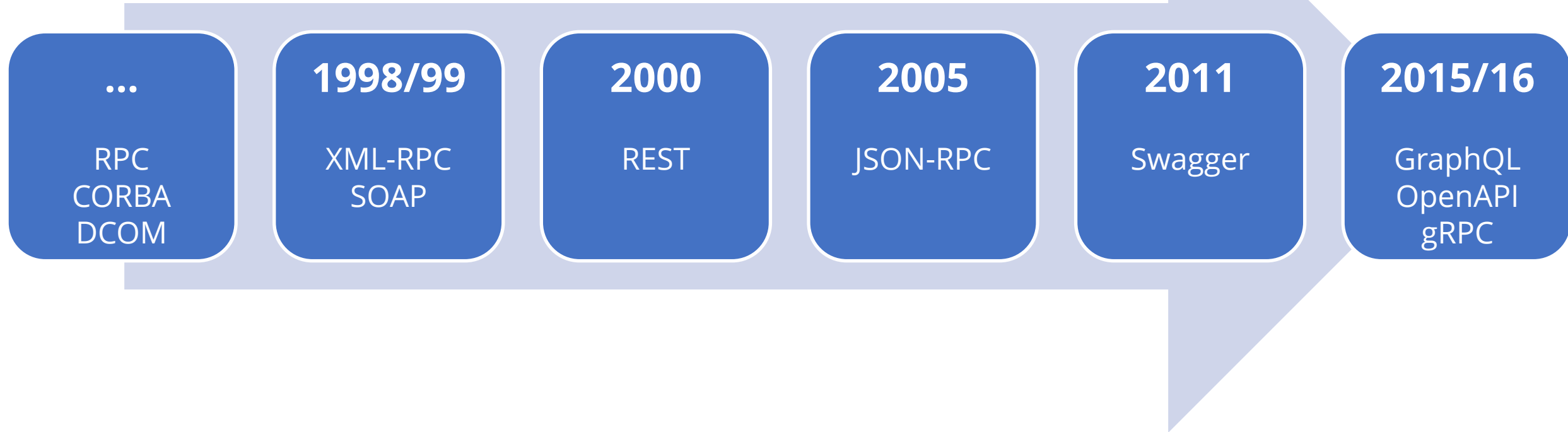
APIs e Web Services

- Os *Web Services* são voltados para atender a outras aplicações
- Todo *Web Service* é uma API. Mas nem toda API é um *Web Service*.
- *Web Services* utilizam, basicamente, o protocolo HTTP como forma de comunicação (Web)










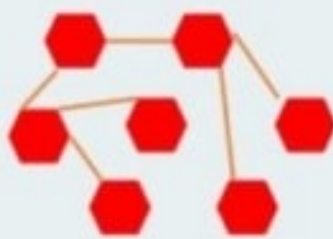


Introdução ao mundo das APIs



Evolução do Mundo das APIs

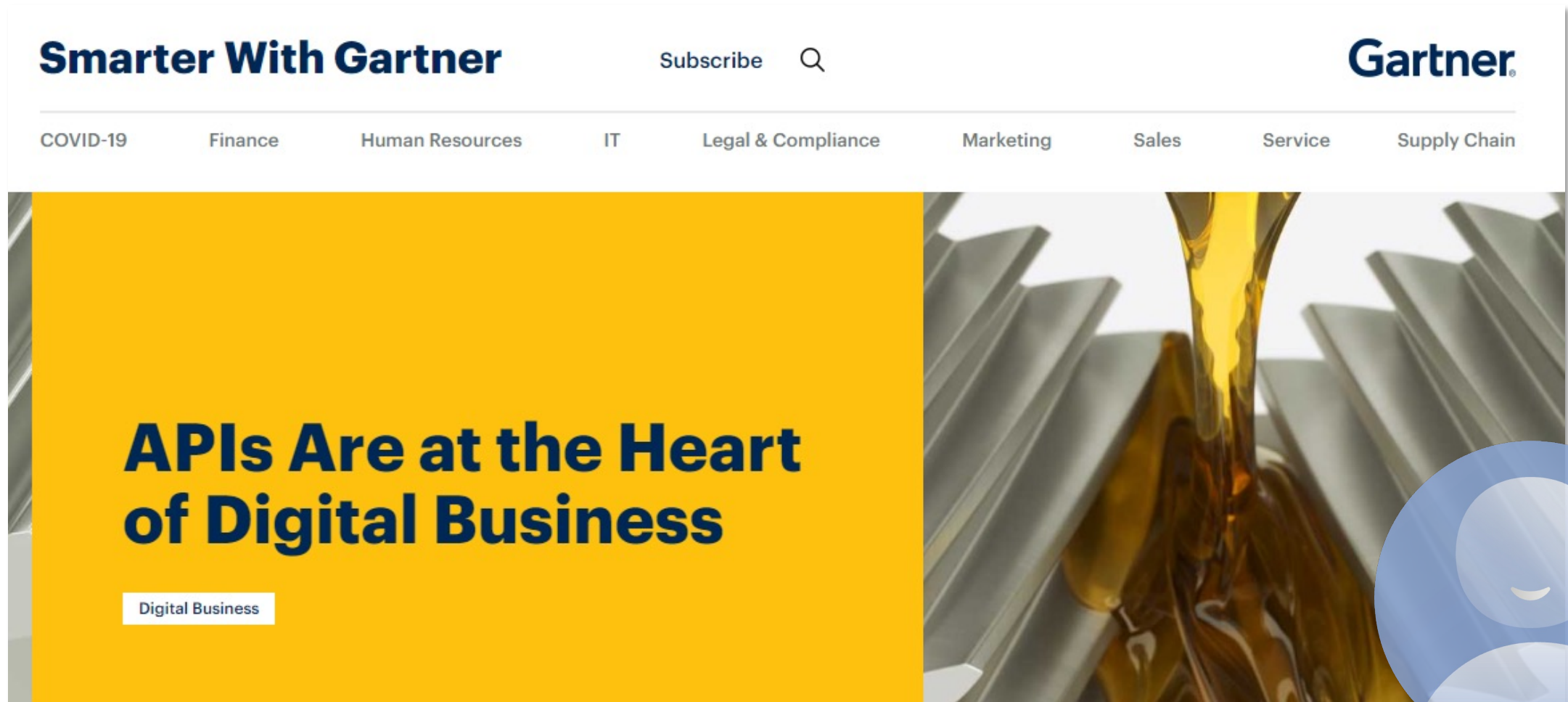


Evolução do Mundo das APIs

	Development Process	Application Architecture	Deployment and Packaging	Application Infrastructure	APIs
~ 1980	Waterfall 	Monolithic 	Physical Server 	Datacenter 	RPC CORBA DCOM XML-RPC
~ 1990					
~ 2000	Agile 	N-Tier 	Virtual Servers 	Hosted 	SOA SOAP REST JSON-RPC
~ 2010	DevOps 	Microservices 	Containers 	Cloud 	Swagger OAI GraphQL gRPC
Now					

Fonte: Adaptado pelo autor de [What is cloud-native?](#)

Mundo das APIs e a Transformação Digital



Fonte: [APIs Are at the Heart of Digital Business - Smarter With Gartner](#)

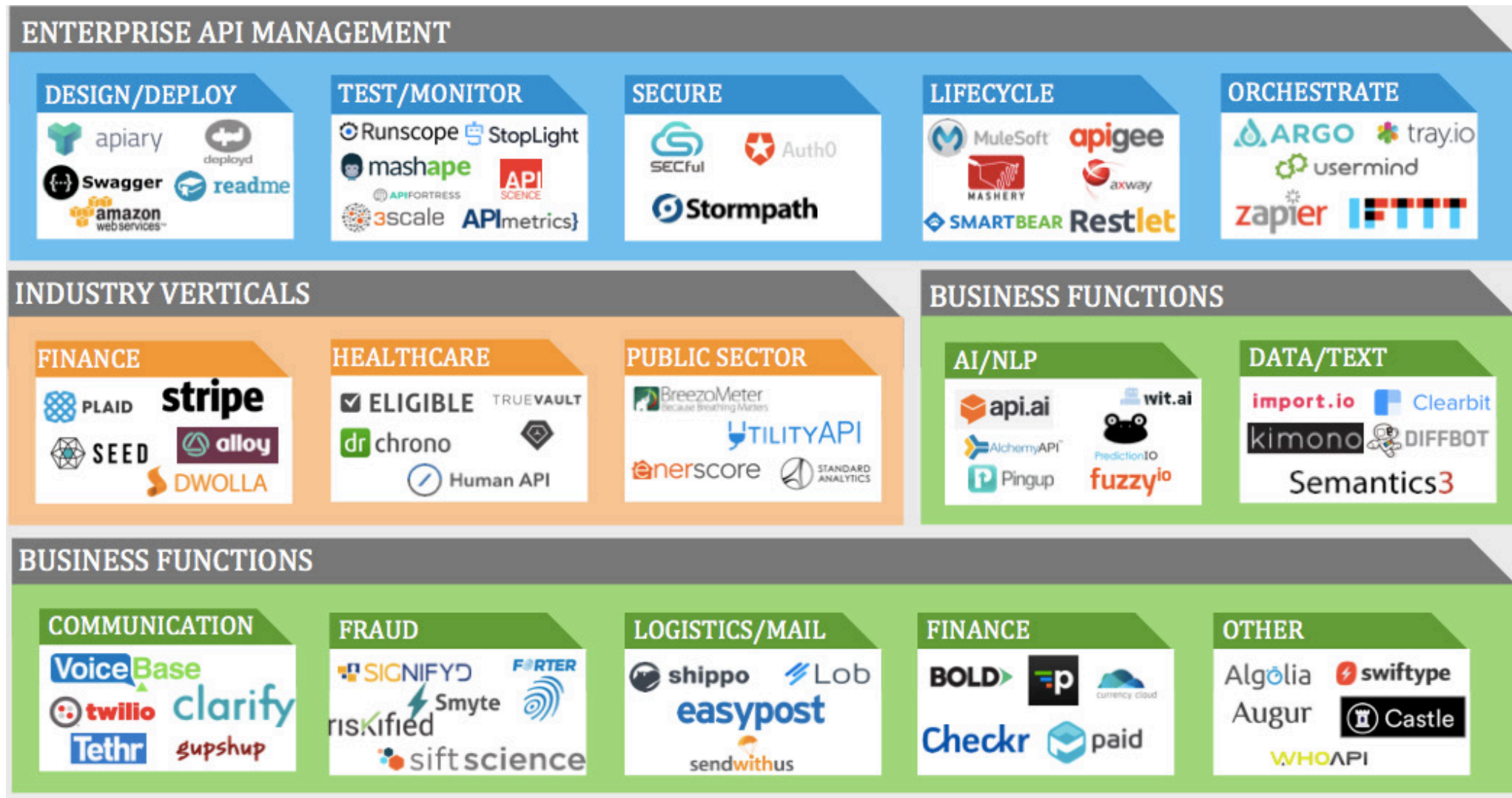


12



Prof. Rommel Vieira Carneiro

Mundo das APIs e a Transformação Digital



Tipos de APIs

Private APIs

- APIs are solely used within a certain organization.
- Apps are mostly built for company employees.
- Integration of company systems/apps and development of new systems using existing resources.

Partner APIs

- APIs are openly promoted but available for known business partners.
- End customers or business users are potential target audiences for such apps.
- The most popular use case for these APIs is software integration between two organizations

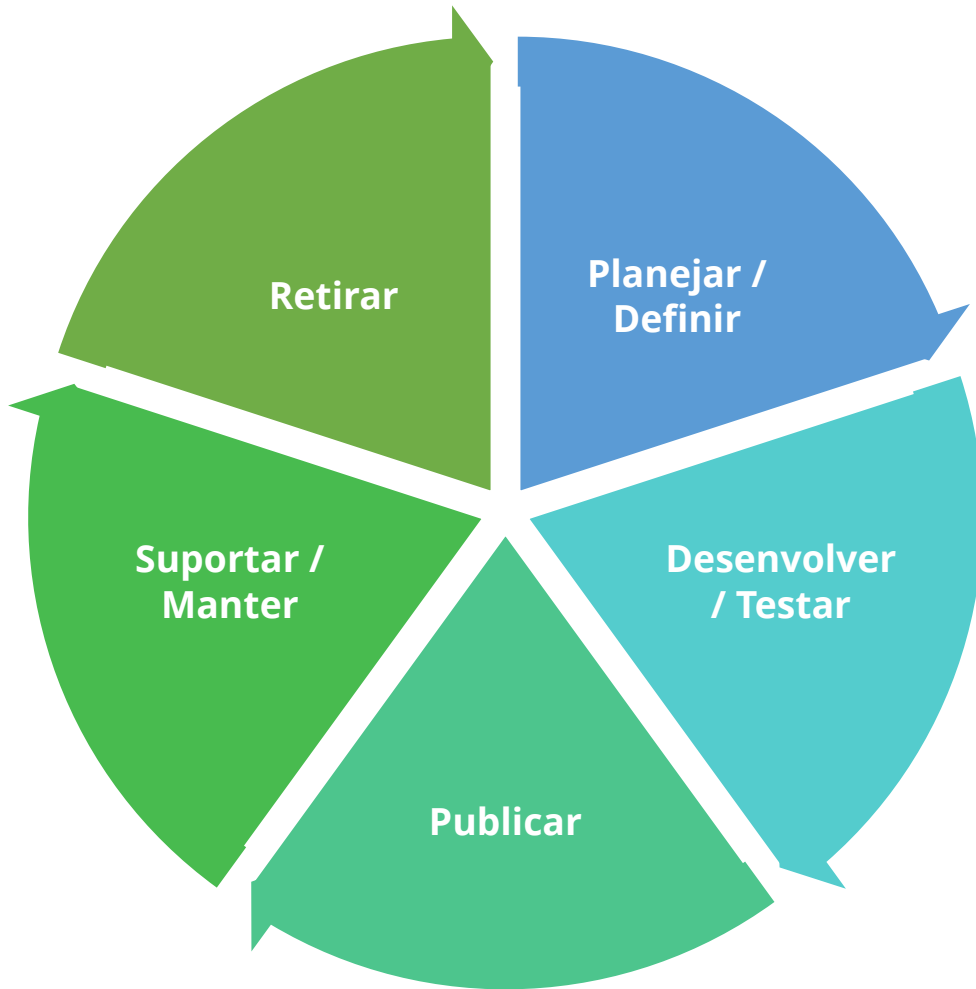
Public or external APIs

- APIs are available to any third-party developers. Can be open and commercial.
- Apps are mostly designed for end customers.
- This API release policy allows for increasing awareness and fostering external innovation.



Fonte: [What is an API: Definition, Types, Specifications, Documentation](#)

Life Cycle API Management



Life Cycle API Management

- Planejar / Definir
- Desenvolver / Testar
- Publicar
- Suportar / Manter
- Retirar



Fontes:

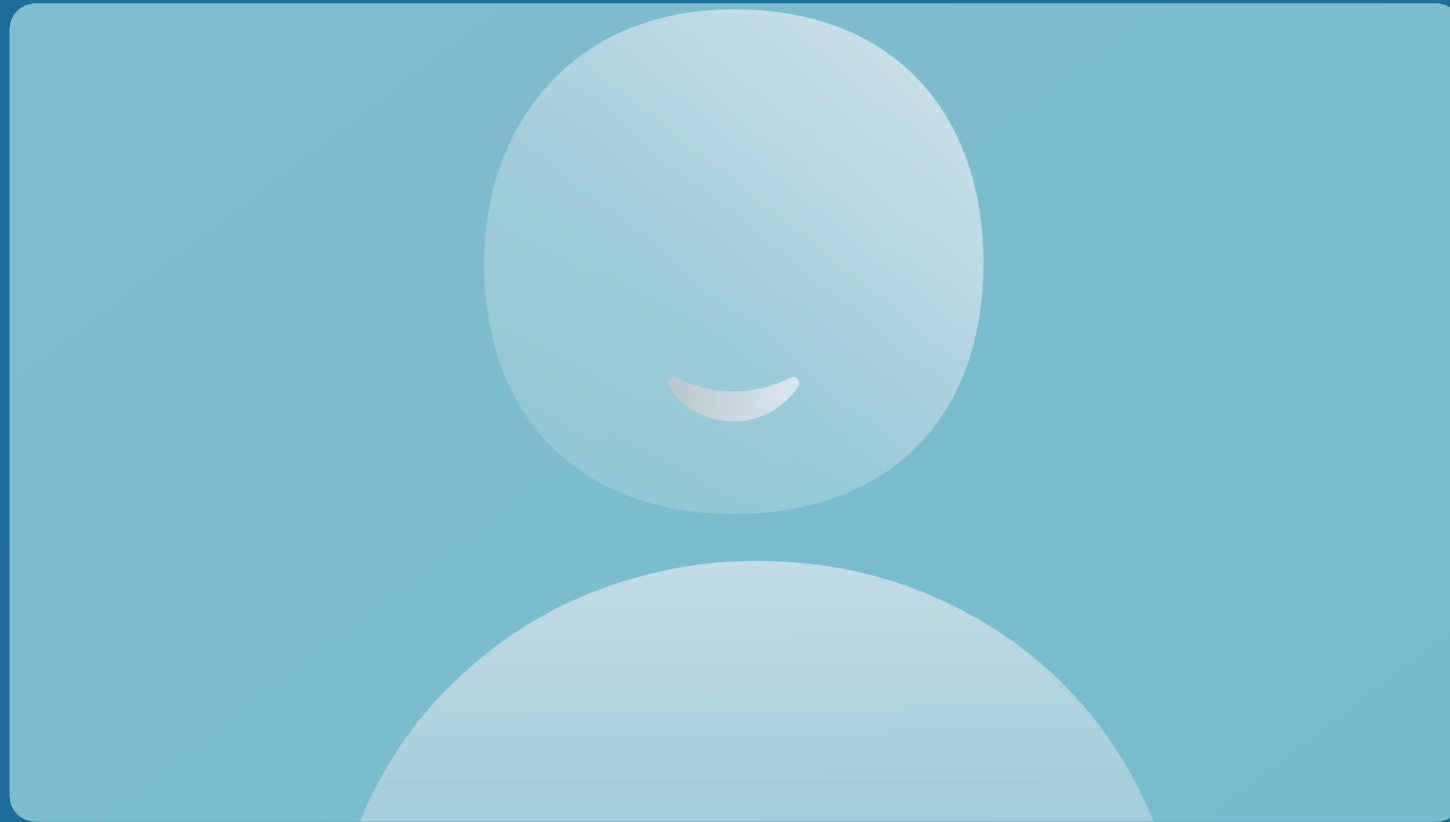
- What is API Lifecycle Management?
<https://swagger.io/blog/api-strategy/what-is-api-lifecycle-management/>
- API and APP Ecosystems
<https://www.slideshare.net/CiscoDevNet/devnet-1133-api-and-app-ecosystems-build-the-best>

Resumo

- Definição de uma APIs e a integração entre as diversas aplicações de negócios
- A evolução do mundo das APIs diante das mudanças no processo de desenvolvimento de software e da TI em geral
- As APIs como elemento fundamental no processo de transformação digital
- O ciclo de vida de criação e gerenciamento das APIs em uma organização.



Protocolo HTTP



Protocolo HTTP

O *Hypertext Transfer Protocol* (HTTP) é um protocolo da camada de aplicação para sistemas distribuídos e colaborativos de informação no formato de hipertextos. (RFC – 2068)

Características

- Requer a atuação de dois programas: Cliente e Servidor
- Atua na camada de aplicação da pilha TCP/IP
- A comunicação utiliza conexões TCP (e UDP no caso do HTTP v3.0)
- O servidor HTTP, por padrão, utiliza a porta 80
- Protocolo que não guarda estado do cliente (stateless)



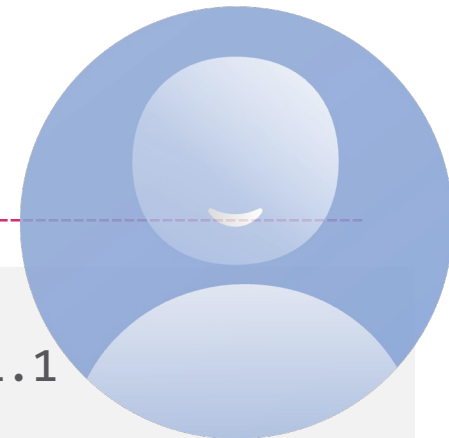
Histórico

1991 • HTTP/0.9
1994 • HTTPS
1996 • HTTP/1.0
1999 • HTTP/1.1

2009 • SPDY 1.0

2015 • HTTP/2
2016 • QUIC
2018 • HTTP/3

Protocolo HTTP – Requisição



Linha de Requisição -----

método

recurso

versão HTTP

POST /app/processamento HTTP/1.1

Linhas de Cabeçalho -----

campo cabeçalho:

valor

...

campo cabeçalho:

valor

User-Agent: Mozilla/4.0 (compatible...)

Host: www.pucminas.br

Content-Type: text/xml; charset=utf-8

Content-Length: 88

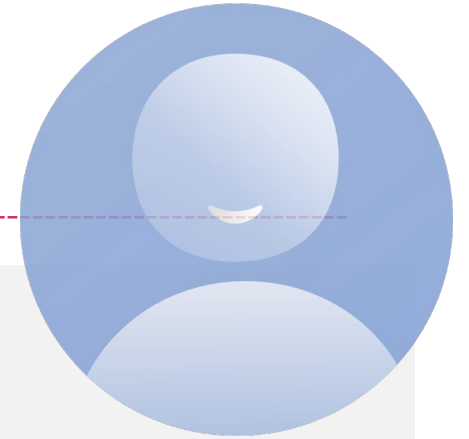
Accept-Language: en-us

Connection: Keep-Alive

Corpo da entidade -----

```
<?xml version="1.0" encoding="utf-8"?>
<string>Conteúdo do arquivo</string>
```

Protocolo HTTP – Resposta



Linha de Resposta -----

versão HTTP

code status

msg status

HTTP/1.1 200 OK

Linhas de Cabeçalho -----

campo cabeçalho:

valor

...

campo cabeçalho:

valor

Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Accept-Ranges: bytes
Content-Length: 88
Content-Type: text/html
Connection: Closed

Corpo da entidade -----

```
<html><body>
<h1>Request Processed Successfully</h1>
</body></html>
```

Protocolo HTTP – Códigos de Retorno

Código	Propósito	Descrição
1xx	Informacional	Requisição recebida, processo em continuidade
2xx	Sucesso	A ação foi recebida, entendida e aceita
3xx	Redirecionamento	Ações adicionais devem ser executadas para completar o pedido
4xx	Erro no cliente	O pedido contém erro de sintaxe ou não pode ser completado
5xx	Erro no servidor	O servidor falhou em completar um pedido aparentemente válido

Exemplos mais comuns

- 200 – Ok
- 403 – Acesso negado
- 404 – Página não encontrada
- 500 – Erro interno do servidor

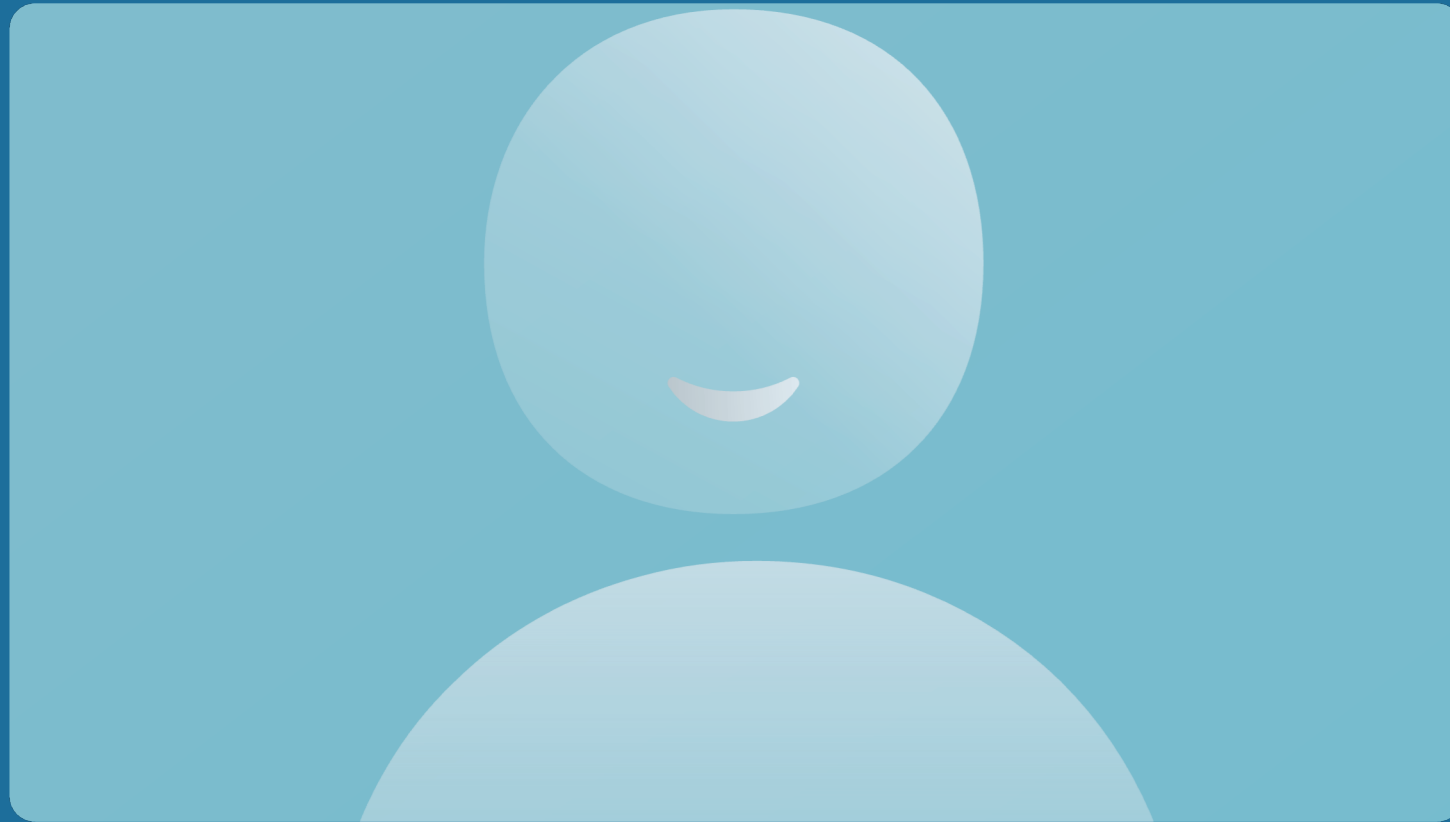


Resumo

- Visão geral sobre o protocolo HTTP e sua importância na criação de aplicações Web e APIs.
- A comunicação HTTP baseada em requisição/resposta, além do formato das requisições e respostas trocadas entre clientes e servidores na Web.
- Os métodos do protocolo HTTP, suas características e suas funcionalidades em uma aplicação Web.
- Os cabeçalhos do protocolo HTTP, seus tipos e alguns exemplos, destacando a sua aplicabilidade.
- Mudanças nas novas versões do protocolo HTTP



Protocolo HTTP – Métodos (*verbs*)



Protocolo HTTP – Métodos

Método	Propósito	Safe (readonly)	Idempotente
GET	Requisitar a representação de um recurso específico	Sim	Sim
POST	Enviar dados a serem processados por um recurso. Usado para incluir recursos ou submeter dados de processamento	Não	Não
HEAD	Similar ao GET, porém retorno deve ser somente do conjunto de cabeçalhos associados ao recurso solicitado	Sim	Sim
PUT	Requisitar a criação ou atualização de um recurso no servidor a partir dos dados no corpo da requisição	Não	Sim
DELETE	Excluir um recurso do servidor	Não	Sim
TRACE	Solicita ao servidor uma cópia (eco) da requisição. Usado para testar se a requisição foi alterada no caminho	Sim	Sim
PATCH	Utilizado para realizar alterações parciais de um recurso	Não	Não
OPTIONS	Usado pelo cliente para entender, ou descobrir, os métodos HTTP e outras opções suportadas por um servidor web	Sim	Sim
CONNECT	Usado quando o cliente estabelece uma conexão HTTPS com um servidor via um proxy	Não	Não

Protocolo HTTP – Métodos – GET



Método GET

- Tem por objetivo requisitar a representação de um recurso ao servidor
- **Por definição, não deve alterar o estado do servidor (*safe*)**
- As requisições podem ser mantidas em cache (favoritos ou bookmarks)
- Envia dados ao servidor via parâmetros na *query string* que ficam visíveis na URL
- Tem restrição quanto ao tamanho e ao formato das informações enviadas ao servidor
 - Formato: limitado a caracteres textuais (ASCII) incluídos na *query string*
 - Tamanho:
 - Apache: 4.000 caracteres
 - MS IIS: 16.384 caracteres
 - Tomcat: padrão 8.192 podendo chegar até 65.536 caracteres

Protocolo HTTP – Métodos – GET



Método GET

- Este é o método mais utilizado em aplicações Web.
- Ao informar uma URL em um navegador, o usuário está disparando uma requisição do tipo GET

Requisição

```
GET /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01)
Host: www.pucminas.br
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Accept-Ranges: bytes
Content-Length: 88
Content-Type: text/html
Connection: Closed

<html>
  <body> <h1>Hello, World!</h1> </body>
</html>
```

Protocolo HTTP – Métodos – POST

Método POST

- Envia dados ao servidor para serem processados
- **Por definição tem objetivo de alterar o estado do servidor** (*not safe*)
- Pode enviar dados via query string ou via corpo da requisição
 - Os dados enviados pelo corpo **não** ficam visíveis na URL
 - Muito utilizado para envio de dados sensíveis como senhas de acesso
- Não podem ser ‘favoritados’ (bookmarked)
- Não possuem restrição quanto ao tamanho e ao tipo de dados a serem enviados ao servidor



Protocolo HTTP – Métodos – POST



Método POST

- Normalmente é utilizado em conjunto com formulários HTML
- Observe os dados enviados no corpo da Requisição

Requisição

```
POST /cgi-bin/process.cgi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01)
Host: www.pucminas.br
Content-Type: text/xml; charset=utf-8
Content-Length: 88
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

```
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://clearforest.com/">string
</string>
```

Resposta

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Accept-Ranges: bytes
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

```
<html><body><h1>Request Processed
Successfully</h1></body></html>
```

Protocolo HTTP – Métodos – HEAD

Método HEAD

Possui estrutura e objetivo similar às requisições de GET, porém o servidor deve enviar apenas o conjunto de cabeçalhos associados ao recurso informado.



Requisição

```
HEAD /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01)
Host: www.pucminas.br
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

Resposta

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Vary: Authorization,Accept
Accept-Ranges: bytes
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

Protocolo HTTP – Métodos – PUT

Método PUT

Requisita a criação ou atualização de um recurso no servidor a partir dos dados no corpo da requisição. Utilizado no upload de arquivos para servidores Web.



Requisição

```
PUT /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01)
Host: www.pucminas.br
Accept-Language: en-us
Connection: Keep-Alive
Content-type: text/html
Content-Length: 182
```

```
<html><body>
<h1>Hello, World!</h1>
</body></html>
```

Resposta

```
HTTP/1.1 201 Created
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Content-type: text/html
Content-length: 30
Connection: Closed
```

```
<html>
<body>
<h1>The file was created.</h1>
</body>
</html>
```

Protocolo HTTP – Métodos – DELETE

Método DELETE

Solicita ao servidor a exclusão de dados ou representações associados ao recurso informado.



Requisição

```
DELETE /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01)
Host: www.pucminas.br
Accept-Language: en-us
Connection: Keep-Alive
```

Resposta

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Content-type: text/html
Content-length: 30
Connection: Closed

<html><body><h1>URL deleted.</h1></body></html>
```

Protocolo HTTP – Métodos – TRACE



Método TRACE

Usado para ecoar o conteúdo de uma requisição HTTP ao servidor. Usado para verificar se a requisição é alterada no caminho por agentes intermediários (servidores de cache ou proxy).

Requisição

```
TRACE / HTTP/1.1
Host: www.pucminas.br
User-Agent: Mozilla/4.0 (compatible; MSIE5.01)
```

Resposta

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Connection: close
Content-Type: message/http
Content-Length: 39

TRACE / HTTP/1.1
Host: www.pucminas.br
User-Agent: Mozilla/4.0 (compatible; MSIE5.01)
```


Protocolo HTTP – Métodos – OPTIONS

Método OPTIONS

Usado para descobrir métodos HTTP e outras opções suportados pelo servidor. O cliente pode especificar uma URL para o método de opções ou um asterisco (*) para se referir a todo o servidor.



Requisição

```
OPTIONS * HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01)
```

Resposta

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Allow: GET,HEAD,POST,OPTIONS,TRACE
Content-Type: httpd/unix-directory
```

Protocolo HTTP – Métodos – CONNECT

Método CONNECT

Usado pelo cliente para estabelecer uma conexão com o servidor web que pode ser via protocolo seguro (TLS). É utilizado no caso de requisições a proxies.



Requisição

```
CONNECT www.pucminas.br HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01)
```

Resposta

```
HTTP/1.1 200 Connection established
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
```

GET vs POST

http://www.w3schools.com/tags/ref_httpmethods.asp

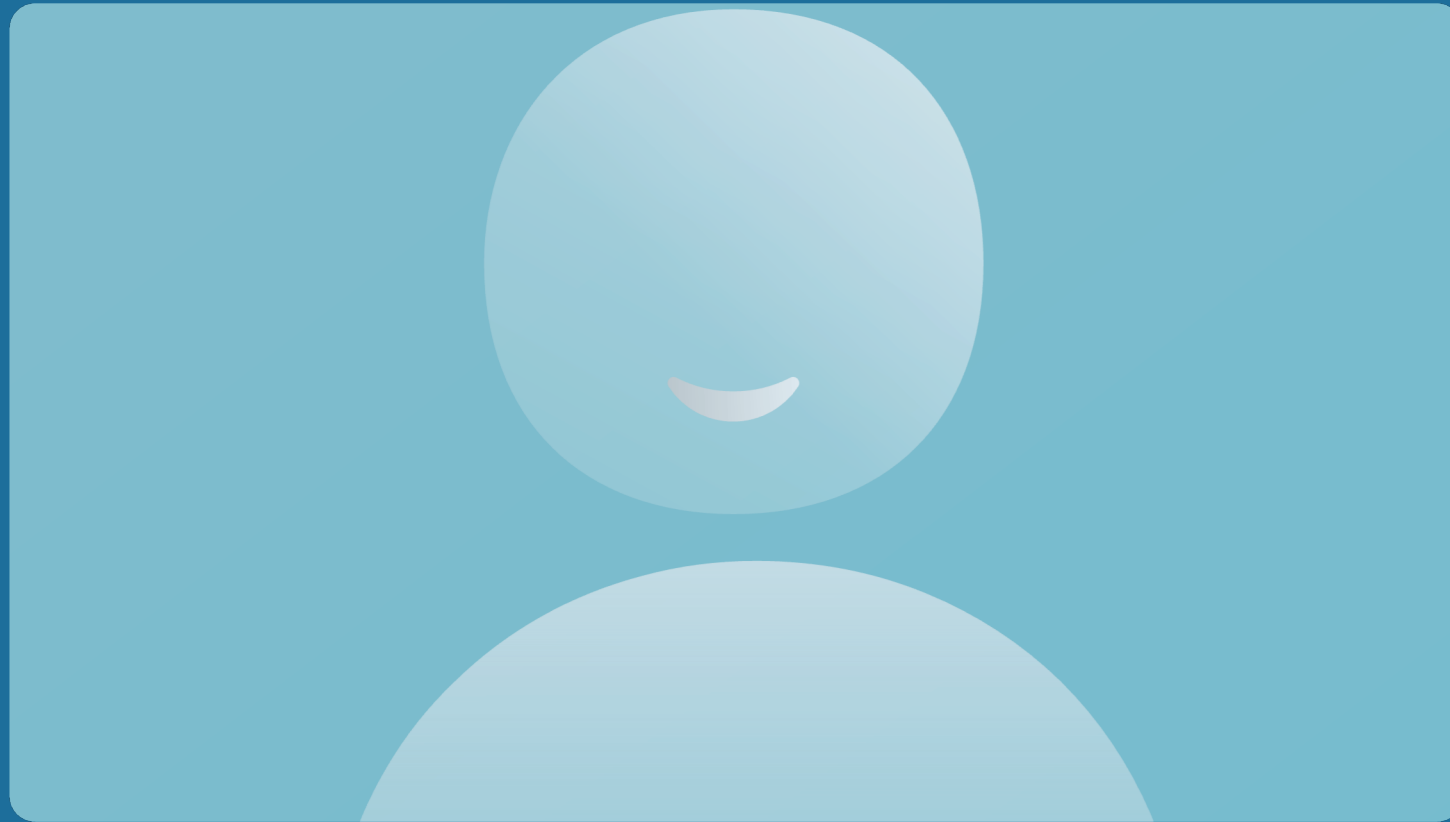
	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Resumo

- Visão geral sobre o protocolo HTTP e sua importância na criação de aplicações Web e APIs.
- A comunicação HTTP baseada em requisição/resposta, além do formato das requisições e respostas trocadas entre clientes e servidores na Web.
- Os métodos do protocolo HTTP, suas características e suas funcionalidades em uma aplicação Web.
- Os cabeçalhos do protocolo HTTP, seus tipos e alguns exemplos, destacando a sua aplicabilidade.
- Mudanças nas novas versões do protocolo HTTP



Protocolo HTTP – Cabeçalhos (*headers*)



Protocolo HTTP – Cabeçalhos

Os cabeçalhos utilizados em requisições e respostas do protocolo HTTP carregam informações adicionais sobre a comunicação entre cliente e servidor.



Tipos de Cabeçalhos

- **Request header:** informações sobre a requisição feita ou sobre o cliente Web.
- **Response header:** informações sobre a resposta encaminhada ou sobre o servidor Web.
- **Entity header:** informações sobre o conteúdo da entidade trocada como tamanho e tipo.
- **General header:** Usado tanto em requisições quanto em respostas.

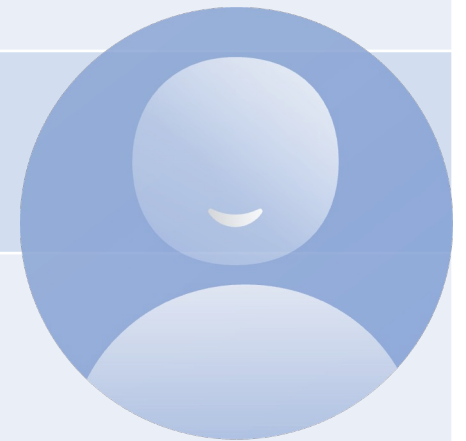
```
POST /app/processamento HTTP/1.1
```

```
User-Agent: Mozilla/4.0 (compatible...)  
Host: www.pucminas.br  
Content-Type: text/xml; charset=utf-8  
Content-Length: 88  
Accept-Language: en-us  
Connection: Keep-Alive
```

```
<?xml version="1.0" encoding="utf-8"?>  
<string>Conteúdo do arquivo</string>
```

Protocolo HTTP – Cabeçalhos de Requisição

Cabeçalho	Utilidade e exemplos	requisição
Accept	<p>Lista os tipos de mídia aceitáveis para a resposta. Indica que a solicitação está limitada a um pequeno conjunto de tipos desejados.</p> <p>Accept: application/json Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8</p>	
Accept-Charset	<p>Lista os conjuntos de caracteres que são aceitáveis para a resposta.</p> <p>Accept-Charset: utf-8, iso-8859-1;q=0.5</p>	
Accept-Encoding	<p>Lista conjuntos de codificações que são aceitáveis para a resposta.</p> <p>Accept-Encoding: gzip, deflate</p>	
Accept-Language	<p>Lista os conjuntos de idiomas naturais aceitáveis e preferidos pelo usuário para a resposta.</p> <p>Accept-Language: pt-BT, en;q=0.9, /*;q=0.8</p>	



Protocolo HTTP – Cabeçalhos de Requisição

Cabeçalho	Utilidade e exemplos	requisição
Authorization	Informa as credenciais de autenticação do User Agent Authorization: Basic SGxsdfRp32hgIKVrw5VzW1	
Host	Indica o host e a porta de onde o recurso está sendo solicitado Host: pucminas.br	
Referer	Informa a URL do recurso de origem, ou visitado antes da requisição atual e que, possivelmente, direcionou o usuário para este recurso referer: https://acesso.gov.br/login?id=acesso.gov.br	
User-Agent	Informa, ao servidor, detalhes sobre o user agent (cliente Web) que está enviando esta requisição user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.163 Safari/537.36	

Protocolo HTTP – Cabeçalhos de Resposta

Cabeçalho	Utilidade e exemplos	resposta
Server	Informa detalhes do software que implementa o servidor Web Server: Apache/2.4.34 OpenSSL/1.0.2k-fips PHP/5.5.38	
Set-Cookie	Apresenta cookies a serem armazenados pelo cliente e que devem ser enviados ao servidor nas próximas requisições. set-cookie: MLPRICING=1; Domain=magazineluiza.com.br;	
Etag	Traz um identificador da versão do recurso que altera toda vez que este for alterado. É utilizado pelo controle de cache. Etag: "353-527867f65e8ad"	

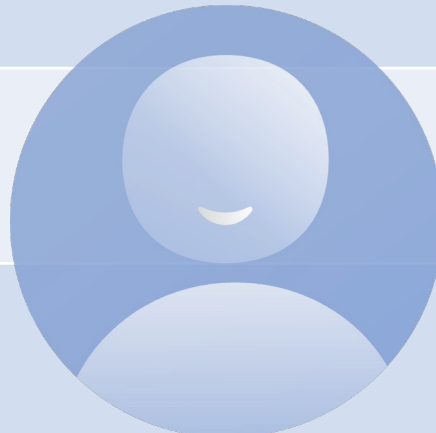


Protocolo HTTP – Cabeçalhos de Resposta

Cabeçalho	Utilidade e exemplos	resposta
Location	Redireciona o cliente Web outra URI Location: https://www.pucminas.br/Paginas/main.aspx	
WWW-Authenticate	indica que o servidor requer a autenticação do usuário para ter acesso ao recurso e de que forma WWW-Authenticate: Basic realm="Site X", charset="UTF-8"	

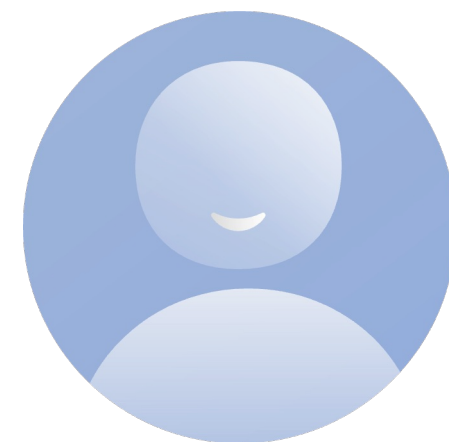


Protocolo HTTP – Cabeçalhos de Entidade

Cabeçalho	Utilidade e exemplos	entidade
Content-Encoding	Indica uma modificação ao tipo de mídia empregado no conteúdo Content-Encoding: gzip	
Content-Language	Descreve a linguagem na qual o conteúdo foi criado (en, pt, etc) Content-Language: pt-br	
Content-Length	Indica a quantidade em número de bytes na notação decimal Content-Length: 17515	
Content-Location	Local alternativo para o recurso solicitado Content-Location: /index.htm	
Content-Type	Indica o tipo de mídia do conteúdo Content-Type: text/html; charset=utf-8	

Protocolo HTTP – Cabeçalhos de Entidade

Cabeçalho	Utilidade e exemplos	entidade
Expires	Informa a data de expiração do recurso recebido Expires: Sun, 31 Jul 2016 05:00:00 GMT	
Last-Modified	Informa a data e hora de última modificação do recurso no servidor Last-Modified: Tue, 06 Nov 2018 22:45:26 GMT	



Resumo

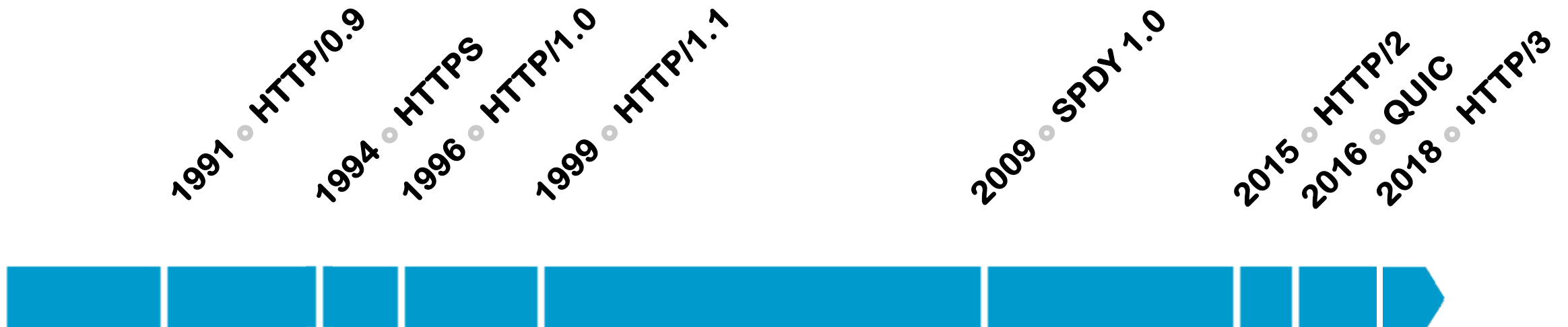
- Visão geral sobre o protocolo HTTP e sua importância na criação de aplicações Web e APIs.
- A comunicação HTTP baseada em requisição/resposta, além do formato das requisições e respostas trocadas entre clientes e servidores na Web.
- Os métodos do protocolo HTTP, suas características e suas funcionalidades em uma aplicação Web.
- Os cabeçalhos do protocolo HTTP, seus tipos e alguns exemplos, destacando a sua aplicabilidade.
- Mudanças nas novas versões do protocolo HTTP



Protocolo HTTP – Versões



Protocolo HTTP – Histórico de Versões



Protocolo HTTP – Histórico de Versões

- **1991** - O HTTP 0.9 é lançado
- **1994** - O HTTPS foi criado pela Netscape
- **1996** - O HTTP 1.0 foi lançado
 - Conceito de cabeçalhos
 - Códigos de Status
- **1999** - O HTTP 1.1 foi lançado
 - Conexões TCP persistentes
 - Suporte a Virtual Host (Cabeçalho Host)
 - Autenticação Digest
 - Controle de cache
 - Possibilidade de compressão de dados
- **2009** - Google propõe o SPDY
- **2015** - O HTTP 2.0 é lançado
 - Baseado no SPDY
 - Compressão de dados obrigatória
 - Cabeçalhos binários
 - Requisições paralelas
 - Envio apenas de cabeçalhos alterados nas próximas requisições
 - Priorização de requisições
 - Server PUSH – Envio automático de arquivos adicionais.
- **2018** – O HTTP 3 é lançado
 - Protocolo de transporte QUIC baseado em UDP

Protocolo HTTP – HTTP/1.1 vs HTTP/2

HTTP 1.1	X	HTTP 2.0
Protocolo textual.		Protocolo binário.
Protocolo sequencial. Requer mais de uma conexão para simular o paralelismo de requisições		Protocolo assíncrono. Utiliza multiplexação para realizar requisições paralelas em uma única conexão
Não prioriza requisições		Possui priorização de requisições
Apenas o cliente pode iniciar uma requisição.		Possui o mecanismo de server push (servidor infere requisições futuras e realizar o envio antecipado)
Compressão de dados é opcional		Compressão de dados é padrão e obrigatória
Envia todos os dados de cabeçalho em cada mensagem		Comprime cabeçalhos para enviar apenas os dados que sofreram alteração ou são desconhecidos

Protocolo HTTP

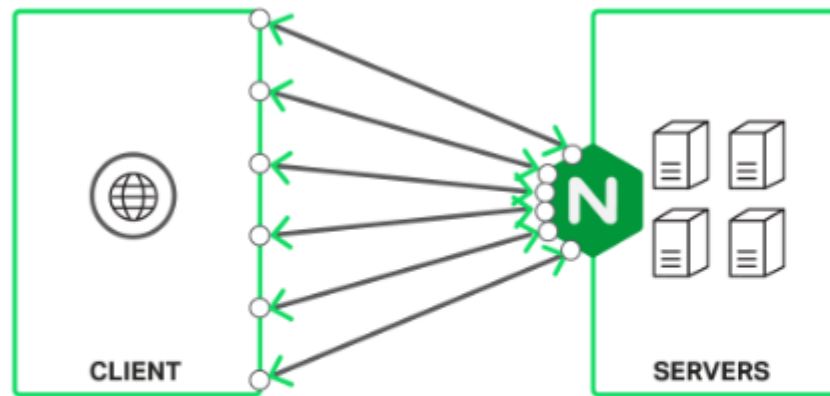
HTTP/2 – Novos recursos

- **Conexão única e persistente** - apenas uma conexão é usada para cada página da web. A mesma conexão é usada enquanto a página da Web estiver aberta.
- **Multiplexação** – requisições e respostas são paralelas e assíncronas, o navegador solicita vários arquivos e os recebe assim que estiverem prontos na mesma conexão.
- **Compressão de cabeçalhos e codificação binária** – os cabeçalhos são comprimidos usando um novo padrão separado e seguro de compressão, chamado HPACK, que reduz a quantidade de dados que cruzam a rede. As informações de cabeçalho são enviadas em formato compacto e binário, não como texto simples.
- **Priorização** – As solicitações recebem níveis de dependência e solicitações no mesmo nível são priorizadas. O servidor utiliza essas informações para ordenar e atribuir recursos para atender às solicitações.
- **Criptografia SSL** – O HTTP/2 permite adicionar suporte SSL com, em alguns casos, nenhuma penalidade de desempenho, tornando o seu site mais seguro.

Fonte: [HTTP/2 for Web Application Developers](#)

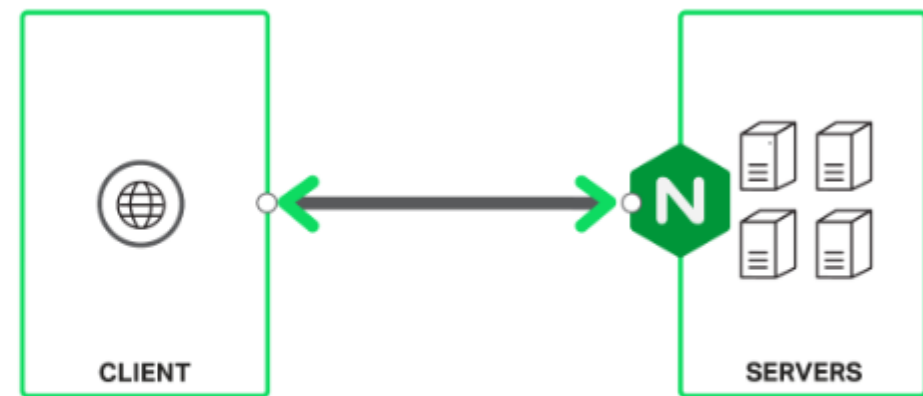
Protocolo HTTP

HTTP/2



HTTP/1.X

VS



HTTP/2

Fonte: [HTTP/2 for Web Application Developers](#)

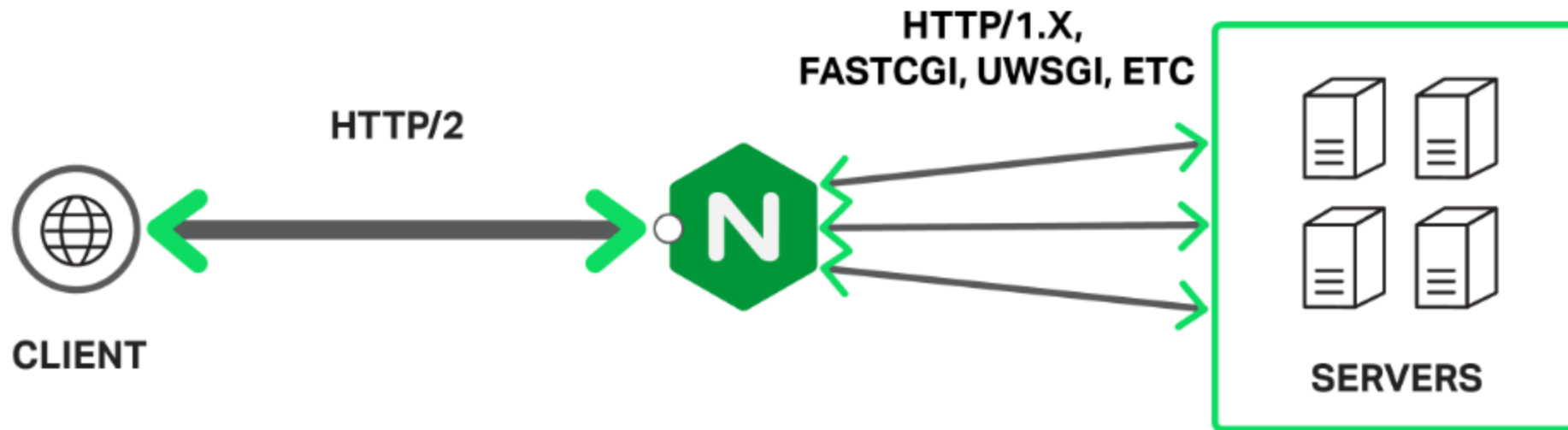
Protocolo HTTP

HTTP/2 x HTTP/1.1

HTTP/1.1 Page Load	HTTP/2 Page Load
1. Create six to eight connections.	1. Create a single connection.
2. Request HTML page.	2. Request HTML page.
3. Receive HTML page.	3. Receive HTML page.
4. Decode HTML page.	4. Decode HTML page.
5. Request first six to eight files included in the HTML page, no priorities or dependencies. (Requests have uncompressed, plain-text headers.)	5. Request all files included in the HTML page, with priorities and dependencies. (Requests have compressed, binary headers.)
6. On each connection, wait for requested file to arrive.	(Files returned, multiplexed on single connection, as ready.)
7. Request next file on now-open connection.	--
8. Repeat 6-7 for each remaining file.	--
9. Close six connections.	8. Close single connection.

Protocolo HTTP

HTTP/2 – Proxy Reverso HTTP/2 com NGINX



Fonte: [HTTP/2 for Web Application Developers](#)

Resumo

- Visão geral sobre o protocolo HTTP e sua importância na criação de aplicações Web e APIs.
- A comunicação HTTP baseada em requisição/resposta, além do formato das requisições e respostas trocadas entre clientes e servidores na Web.
- Os métodos do protocolo HTTP, suas características e suas funcionalidades em uma aplicação Web.
- Os cabeçalhos do protocolo HTTP, seus tipos e alguns exemplos, destacando a sua aplicabilidade.
- Mudanças nas novas versões do protocolo HTTP



Obrigado!



Prof. Rommel Vieira Carneiro

in [/rommelcarneiro](https://www.linkedin.com/in/rommelcarneiro)

 <http://rommelcarneiro.me/>