

Webservices

Que são webservices?

- Webservices são sistemas de software projetados no ano 2000, para suportar a interoperabilidade entre diferentes máquinas na rede.
- Permitem a comunicação e a troca de dados entre diferentes aplicações, independentemente da linguagem de programação ou plataforma utilizada.
- Os webservices são baseados em padrões abertos, como XML, SOAP e REST, e são amplamente utilizados para integrar sistemas e disponibilizar serviços online.
- Facilitam a comunicação entre aplicações distribuídas e permitem que diferentes sistemas se comuniquem de forma eficiente pela internet.

Que são webservices?

- Usados principalmente como um meio para as empresas se comunicarem entre si e com os clientes.
- Os webservices permitem que as organizações comuniquem dados sem o conhecimento profundo dos sistemas de TI uns dos outros por trás do firewall.



Exemplo de webservices?

- Uma forma de se entender a importância dos webservices para redução nos preços dos sites é considerar o seguinte exemplo: antes de fechar uma compra num e-commerce qualquer, certamente você vai querer saber o valor do frete.
- Assim, você enche seu carrinho de compras, insere seus dados e insere seu CEP para que o valor do frete seja dado.
- Em geral, empresas utilizam um webservice dos Correios que é disponibilizado de forma gratuita para calcular o frete.
- Caso este webservice não fosse disponibilizado gratuitamente, a empresa que faz seu site teria que desenvolver um sistema próprio para calcular o frete em seu site ou você teria que comprá-lo, o que encarece o custo do seu site.



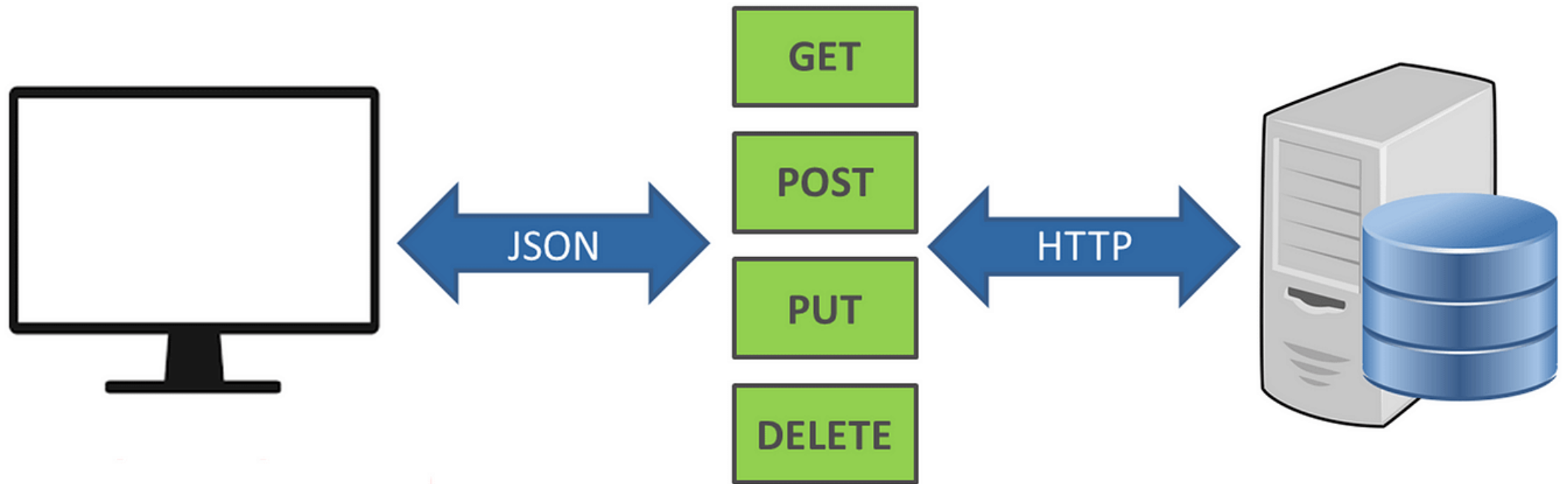
SOAP x REST

- Utiliza HTTP para fazer chamadas RPC trafegando XML;
 - Utiliza **WSDL** na comunicação entre cliente e servidor;
 - Invoca serviços através de chamadas de método **RPC**
 - Não retorna um resultado facilmente legível para humanos;
 - Comunicação feita por HTTP, mas pode usar outros protocolos como SMTP, FTP, etc.
 - JavaScript pode invocar um serviço **SOAP**, mas a implementação é complexa;
 - Comparado com REST, não tem um bom desempenho.
- Utiliza Http para fazer as requisições suportando diversos formatos de arquivos;
 - Utiliza XML, JSON, etc. para enviar e receber dados;
 - Simplesmente chama serviços via URL PATH;
 - Resultado legível por humanos, já que é simplesmente JSON ou XML por exemplo;
 - Comunicação feita unicamente por HTTP;
 - Fácil de invocar via JavaScript;
 - Comparado com SOAP, o desempenho é melhor e consome menos recursos de processamento, código mais simples e enxuto;

Que é REST?

- REST (Representational State Transfer), é um estilo arquitetural de desenvolvimento de APIs (Application Programming Interfaces) muito utilizado na construção de serviços web.
- Se baseia em princípios simples e bem definidos, como a utilização dos métodos HTTP de forma adequada (GET, POST, PUT, DELETE, etc.) para realizar operações em recursos (como dados ou objetos) de forma padronizada e sem estado, ou seja, cada requisição feita pelo cliente contém toda a informação necessária para ser compreendida pelo servidor, sem depender do contexto das requisições anteriores.

Que é REST?



Cliente envia um requisição (request)

Métodos HTTP

Servidor devolve a resposta da requisição (request)

Restrições para uso do REST

1. Cliente/Servidor

- Clientes e servidores separados

2. Stateless server

- O servidor não guarda nenhuma informação do estados do cliente;
- Cada *request* devem conter informações necessárias para ser atendida.

3. Cacheable

- O cliente deve ser informado sobre as propriedades do cache de um recurso para que possa decidir quando deve ou não utilizar **cache**

4. Interface uniforme

- Identificação de recursos (URI)
- Manipulação de recursos a partir de suas representações
- Mensagem auto descritivas
- Hypermedia as the engine of application state - **HATEOAS**

5. Sistema em camadas

- Deve suportar recursos como balanceamento de carga, proxies e firewalls

Formatos suportados pelo REST?

Diversos formatos ou MIME Types por exemplo:

- XML
- JSON
- CSV
- Texto
- Imagens
- HTML
- Binario

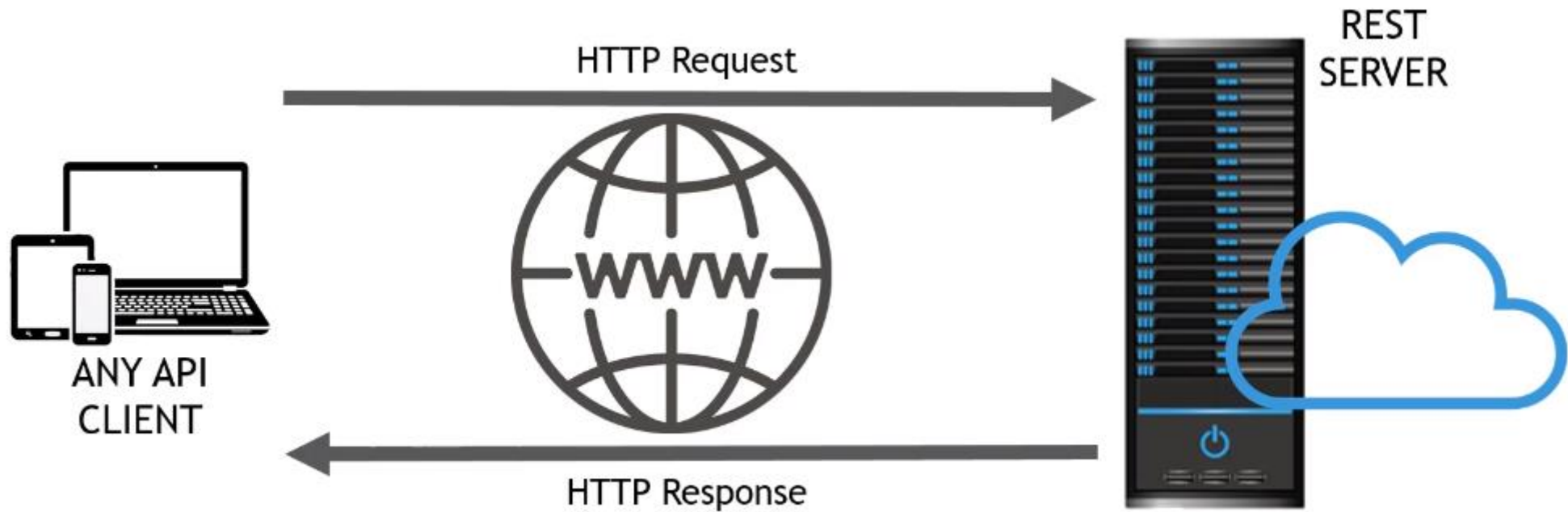
Vantagens dos webservices RESTful

- REST é um padrão arquitetural basicamente leve por natureza. Muito utilizado quando tem limitações de banda;
- Desenvolvimento fácil e rápido;
- Aplicativos mobile tem ganhado cada vez mais espaço e precisam interagir rapidamente com os servidores e o padrão REST é mais rápido no processamento de dados das *requests* e *responses*.

Exemplo use de requisições REST

- **~1 Bilhão** de *requests* por dia
 - Netflix
 - Ebay
 - AccuWeather
 - Sabre/TravelNetwork
- **~5 Bilhões** de *requests* por dia
 - X (antigo Twitter)
 - Google
 - Uber
 - AWS
- **~7 Bilhões** de *requests* por dia
 - Mercado Livre

Request e Response



Request

```
GET /doc/test.html HTTP/1.1
```

```
Host: www.test101.com
```

```
Accept: image/gif, image/jpeg, */*
```

```
Accept-Language: en-us
```

```
Accept-Encoding: gzip, deflate
```

```
User-Agent: Mozilla/4.0
```

```
Content-Length: 35
```

```
bookId=12345&author=Tan+Ah+Teck
```

Request Line

Request Headers

Request
Message
Header

A blank line separates header & body

Request Message Body

Request (Atividade)

- Identifique: Request line, Header e Body Message

```
POST /?id=1 HTTP/1.1
Host: www.swingvy.com
Content-Type: application/json; charset=utf-8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:53.0)
Gecko/20100101 Firefox/53.0
Connection: close
Content-Length: 136
```

```
{
  "status": "ok",
  "extended": true,
  "results": [
    {"value": 0, "type": "int64"},
    {"value": 1.0e+3, "type": "decimal"}
  ]
}
```

Request (Atividade/Gabarito)

- Identifique: Request line, Header e Body Message

`POST /?id=1 HTTP/1.1`

Request line

`Host: www.swingvy.com`

`Content-Type: application/json; charset=utf-8`

`User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:53.0)`

`Gecko/20100101 Firefox/53.0`

`Connection: close`

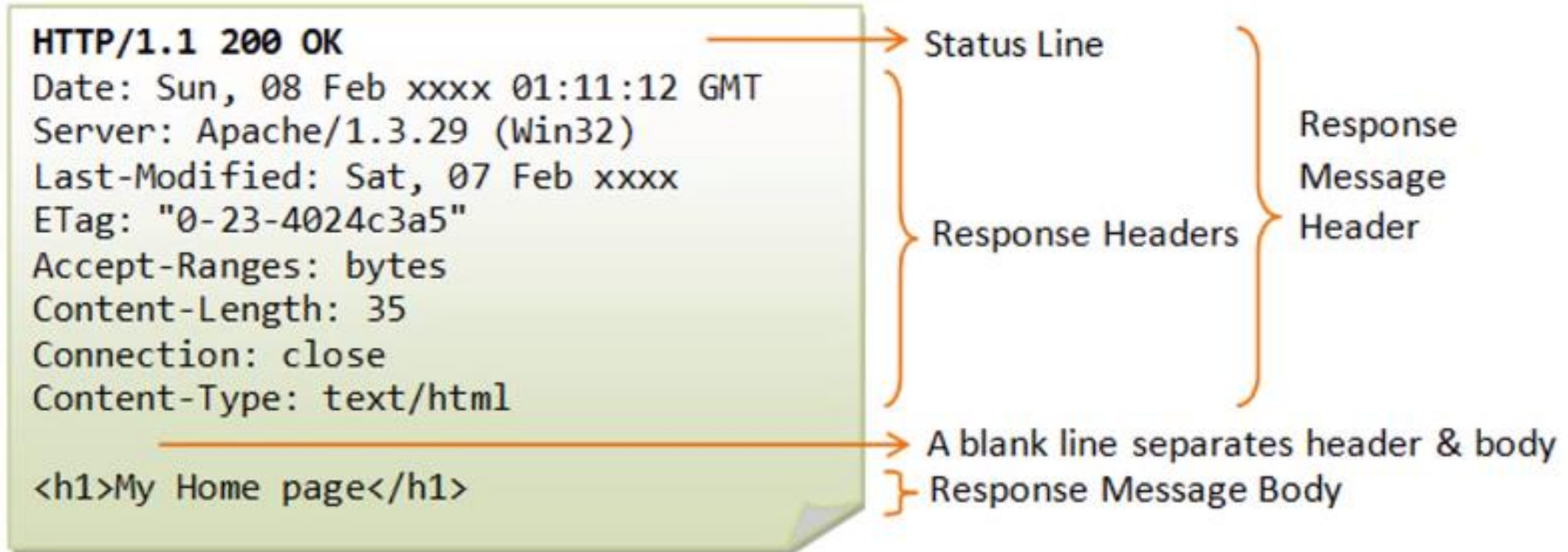
`Content-Length: 136`

Header

```
{
  "status": "ok",
  "extended": true,
  "results": [
    {"value": 0, "type": "int64"},
    {"value": 1.0e+3, "type": "decimal"}
  ]
}
```

Body message

Response



Path params

Os parâmetros são passados via path na URL

`https://server/api/usuários/v1/usuario-busca/asc/155/1`

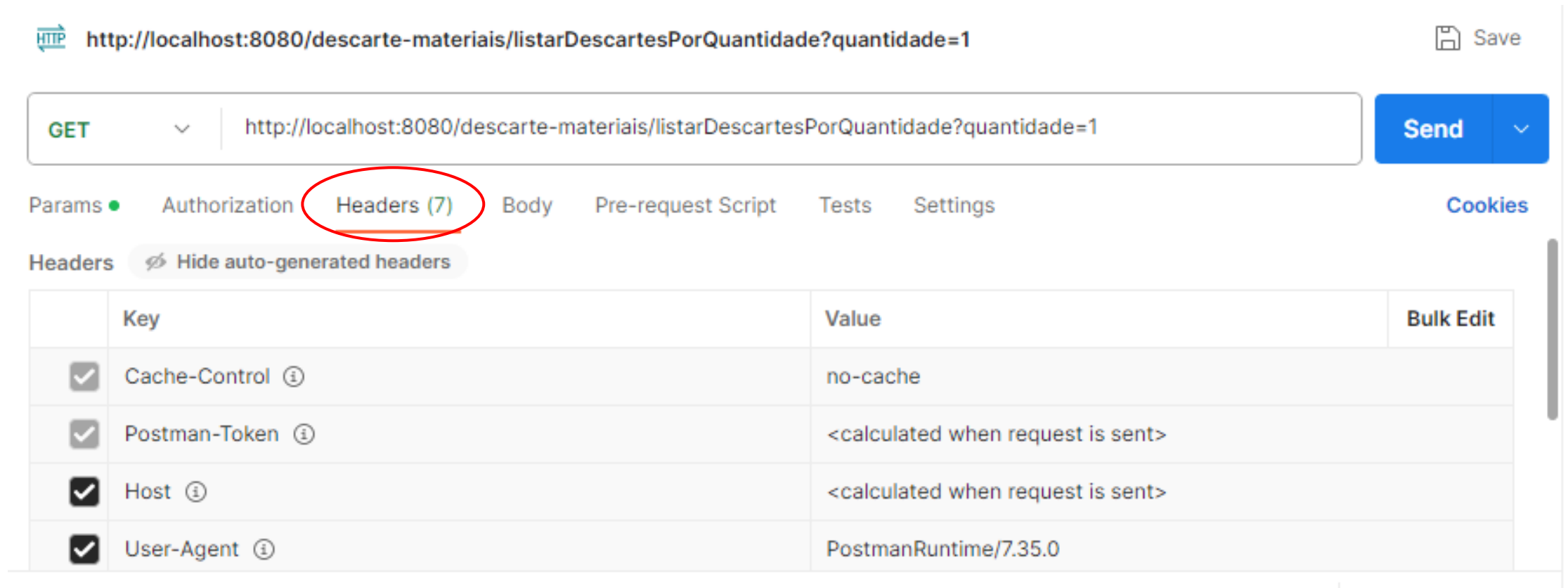
Query params

Os parâmetros são passados com ?variavel=x&variavel=y na URL

`https://server/api/usuários/v1/usuario-busca?codigo=155&tipo=1`

Header params

Os parâmetros são passados no cabeçalho da requisição
(exemplo no **Postman**)

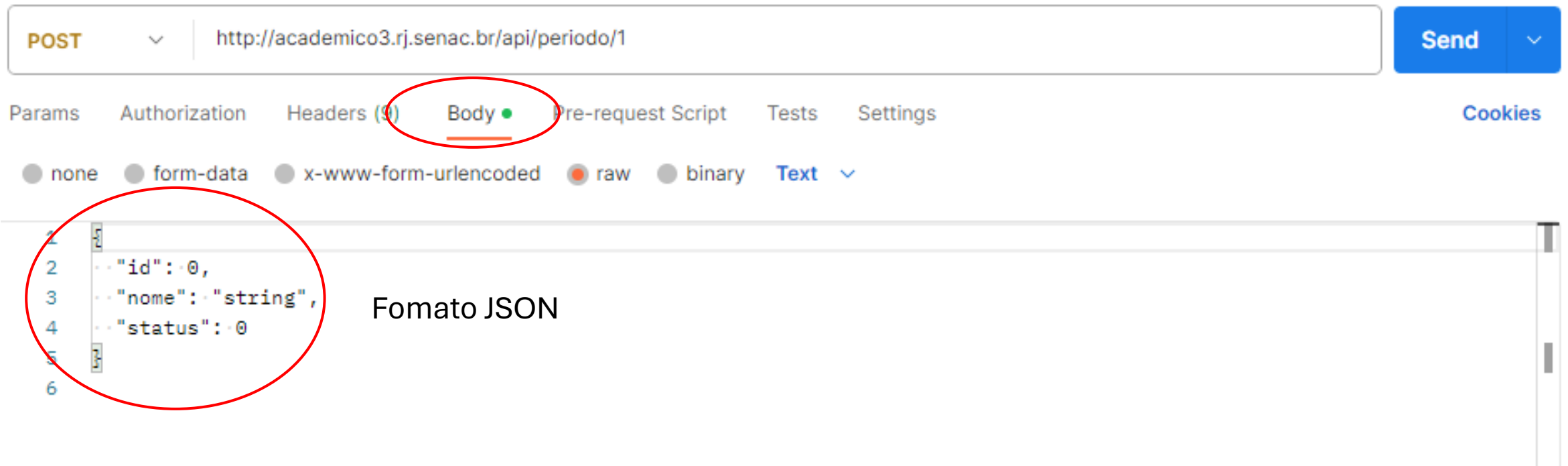


The screenshot shows the Postman interface for a GET request. The URL bar contains `http://localhost:8080/descarte-materiais/listarDescartesPorQuantidade?quantidade=1`. The 'Headers' tab is selected and circled in red. Below the tabs, there is a table of headers.

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	Cache-Control ⓘ	no-cache	
<input checked="" type="checkbox"/>	Postman-Token ⓘ	<calculated when request is sent>	
<input checked="" type="checkbox"/>	Host ⓘ	<calculated when request is sent>	
<input checked="" type="checkbox"/>	User-Agent ⓘ	PostmanRuntime/7.35.0	

Body params

Os parâmetros são passados no corpo (body) da requisição (exemplo no **Postman**)



Status code

Status-code são os três dígitos que indicam qual o erro para o servidor e navegador enquanto a frase razão é uma curta descrição do que este erro significa para melhor compreensão dos usuários. Estes códigos e razões estão descritos na tabela abaixo para o seu conhecimento.

Lista de Código de Status HTTP

1XX: Informativo – a solicitação foi aceita ou o processo continua em andamento;

Código do Status HTTP (Status-code)	Significado do código HTTP (Reason-Phrase)	Significado do código HTTP
100	Continue	Continuar
101	Switching Protocols	Mudando Protocolos
102	Processing	Processando

2XX: Confirmação – a ação foi concluída ou entendida;

Código do Status HTTP (Status-code)	Significado do código HTTP (Reason-Phrase)	Significado do código HTTP
200	Ok	Ok
201	Created	Criado
202	Accepted	Aceito
203	Non-Authoritative Information	Não autorizado
204	No Content	Nenhum Conteúdo
205	Reset Content	Resetar Conteúdo
206	Partial Content	Conteúdo Parcial

3XX: Redirecionamento – indica que algo mais precisa ser feito ou precisou ser feito para completar a solicitação;

Código do Status HTTP (Status-code)	Significado do código HTTP (Reason-Phrase)	Significado do código HTTP
300	Multiple Choices	Múltipla Escolha
301	Moved Permanently	Movido Permanentemente
302	Found	Encontrado
303	See Other	Veja outro
304	Not Modified	Não modificado
305	Use Proxy	Use Proxy
306	Proxy Switch	Proxy Trocado

Lista de Código de Status HTTP

4XX: Erro do cliente- indica que a solicitação não pode ser concluída ou contém a sintaxe incorreta;

Código do Status HTTP (Status-code)	Significado do código HTTP (Reason-Phrase)	Significado do código HTTP
400	Bad Request	Solicitação Inválida
401	Unauthorized	Não autorizado
402	Payment Required	Pagamento necessário
403	Forbidden	Proibido
404	Not Found	Não encontrado
405	Method Not Allowed	Método não permitido
406	Not Acceptable	Não aceito
407	Proxy Authentication Required	Autenticação de Proxy Necessária
408	Request Time-out	Tempo de solicitação esgotado
409	Conflict	Conflito
410	Gone	Perdido
411	Length Required	Duração necessária
412	Precondition Failed	Falha de pré-condição
413	Request Entity Too Large	Solicitação da entidade muito extensa

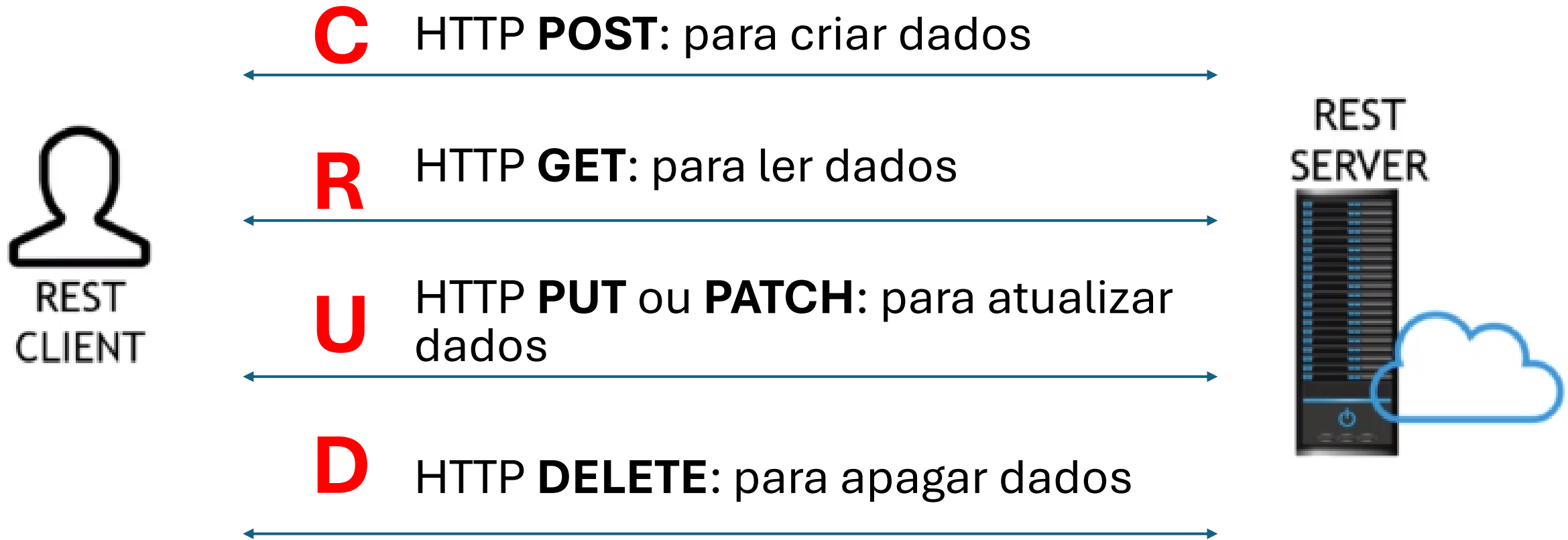
5XX: Erro no servidor – o servidor falhou ao concluir a solicitação.

Código do Status HTTP (Status-code)	Significado do código HTTP (Reason-Phrase)	Significado do código HTTP
414	Request-URL Too Large	Solicitação de URL muito Longa
415	Unsupported Media Type	Tipo de mídia não suportado
416	Request Range Not Satisfiable	Solicitação de faixa não satisfatória
417	Expectation Failed	Falha na expectativa
500	Internal Server Error	Erro do Servidor Interno
501	Not Implemented	Não implementado
502	Bad Gateway	Porta de entrada ruim
503	Service Unavailable	Serviço Indisponível
504	Gateway Time-out	Tempo limite da Porta de Entrada
505	HTTP Version Not Supported	Versão HTTP não suportada

Status code (Atividade)

- 1) Entre na linha de comando (CMD) do Windows
- 2) Digite: `telnet google.com 80`
- 3) Na tela preta digite: `GET /`
- 4) Verifique qual foi o status code da resposta e explique o resultado do comando

Verbos HTTP e o REST (método ou operação)



Glossário

- **WDSL:** é uma notação XML para descrever um serviço da web. Uma definição WSDL indica a um cliente como compor uma solicitação de serviço da web e descreve a interface que é fornecida pelo provedor de serviços da web.
- **RPC:** significa Remote Procedure Call, que em português é chamado de Chamada de Procedimento Remoto. É um protocolo de comunicação que permite a um programa solicitar um serviço de um programa localizado em outro computador em uma rede, sem precisar entender os detalhes da implementação desse serviço. Basicamente, o RPC permite a execução de procedimentos em um sistema remoto como se fossem locais, facilitando a comunicação entre diferentes sistemas distribuídos.

Glossário

- **SOAP:** Um serviço da web SOAP (Simple Object Access Protocol) é um protocolo de comunicação que permite a troca de informações estruturadas em um ambiente distribuído. Ele utiliza XML (Extensible Markup Language) como formato para enviar mensagens entre sistemas. Esse tipo de serviço é amplamente utilizado para integração de sistemas e comunicação entre aplicações em diferentes plataformas;

Glossário

- **API:** Um API, ou Interface de Programação de Aplicações, é um conjunto de regras e protocolos que permite a comunicação entre diferentes softwares. As APIs são utilizadas para facilitar a integração de sistemas e o compartilhamento de dados entre aplicações distintas. Elas definem como os componentes de software devem interagir uns com os outros, permitindo que desenvolvedores criem novas funcionalidades com base em funcionalidades existentes;
- **Cache:** Um "cache" é uma área de armazenamento temporário de dados que são frequentemente acessados ou usados, a fim de acelerar o processo de recuperação dessas informações;

Glossário

- **HATEOAS:** (Hypermedia as the Engine of Application State) é um princípio de design de API (Application Programming Interface) que promove a interconexão entre os recursos de uma aplicação através do uso de links hipermídia. Esse princípio é amplamente utilizado em arquiteturas RESTful para criar APIs mais flexíveis e escaláveis. A ideia central do HATEOAS é que, além de retornar os dados solicitados pelo cliente, a API também fornece links para outros recursos relacionados. Isso permite que o cliente navegue pela API de forma dinâmica, descobrindo e interagindo com os recursos disponíveis.

Exemplo: <https://api.com/usuario>

Glossário

- **Postman:** é uma aplicação que permite testar APIs web.