

INTRODUÇÃO ARQUITETURA CLIENTE E SERVIDOR



Douglas Nassif Roma Junior

 /douglasjunior

 /in/douglasjunior

 nassifroma@gmail.com

Slides: <https://github.com/douglasjunior/Catalisa-DB1-2023>



AGENDA

- Internet
- Cliente e Servidor
- Request e Response
- Protocolo HTTP
- Métodos GET e POST
- Características
- Referências

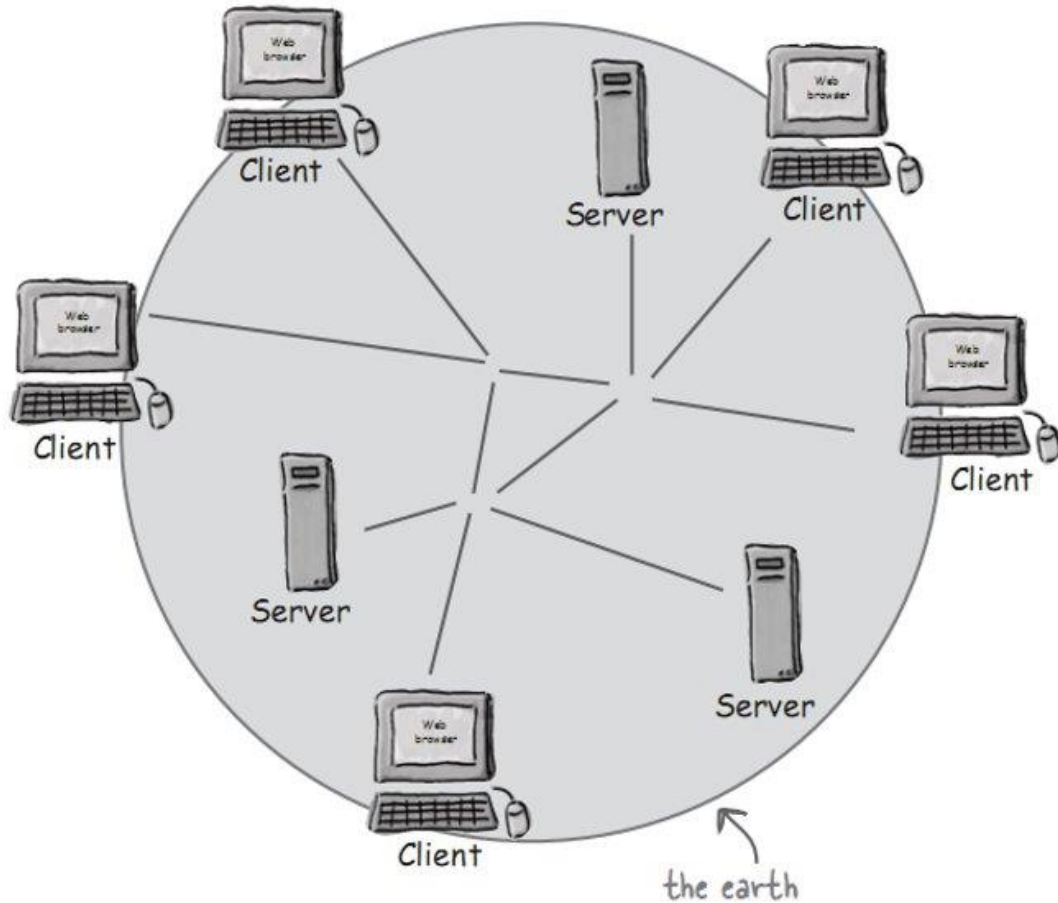
INTERNET



INTERNET

- Criada a ARPANET em 1970 com a finalidade de conectar departamentos de Pesquisa nos EUA
- Protocolo inicial *Network Control Protocol* (NCP)
- Em 1975 criação do TCP/IP
- Em 1990 a Internet passa a ter tendência comercial e não apenas pesquisa.

INTERNET



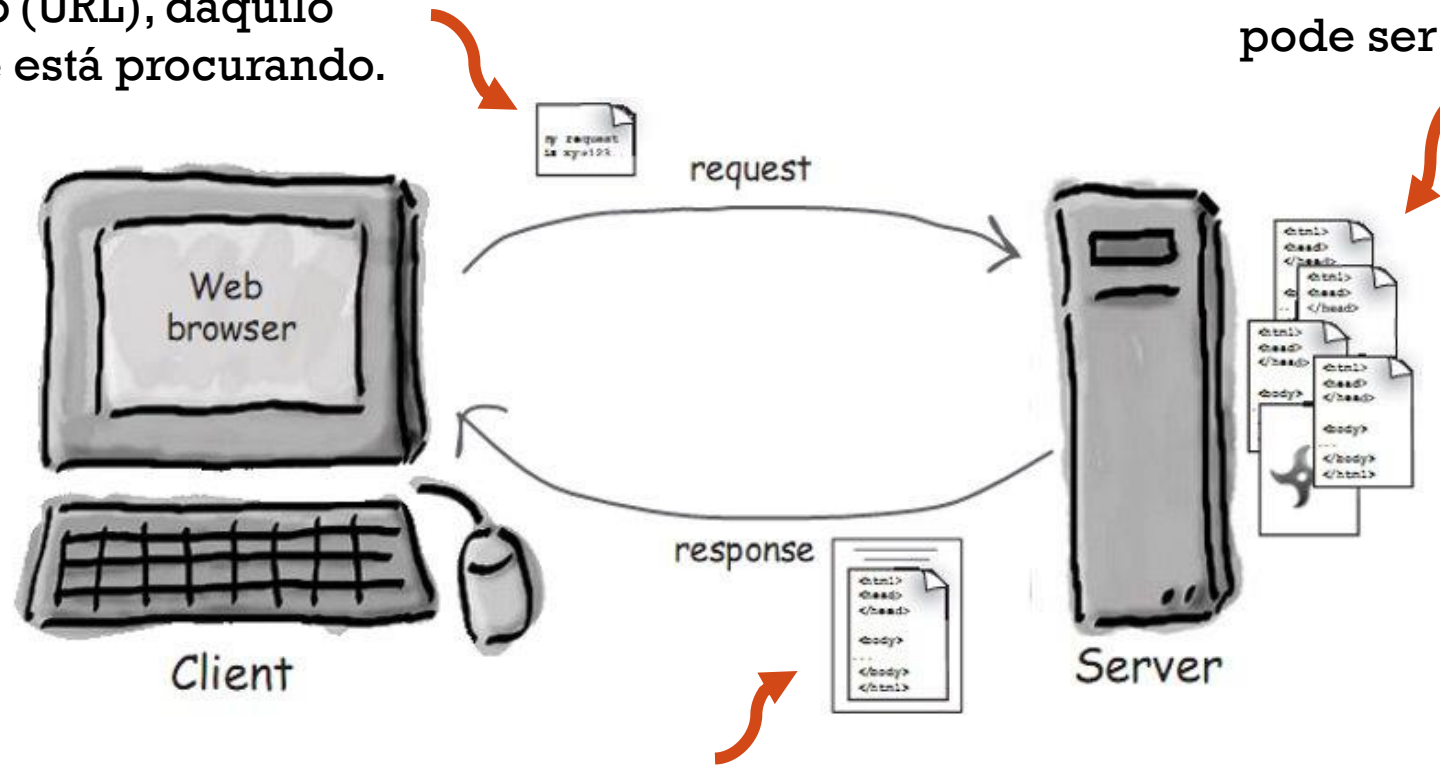
- Clientes (usando browsers como Firefox ou Chrome)
- Servidores (rodando aplicações como o Apache ou Node JS)
- Conectados através de redes com fio ou wireless
- Nosso objetivo é construir uma aplicação que os clientes ao redor do mundo possam acessar.

CLIENTE E SERVIDOR

CLIENTE E SERVIDOR

A solicitação do cliente contém o endereço (URL), daquilo que o cliente está procurando.

Geralmente, o servidor tem muito conteúdo que pode ser retornado para o usuário. Este conteúdo pode ser páginas, imagens, etc.



A resposta do servidor contém o documento que o cliente solicitou (ou um código de erro se o pedido não puder ser processado).

REQUEST E RESPONSE

REQUEST (REQUISIÇÃO)

1

O usuário clica em um link



User

2

O browser formata a requisição e envia pela a rede



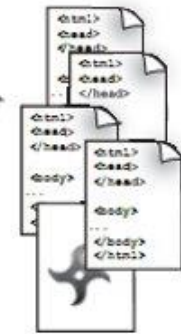
Browser

3

O servidor recebe a requisição e procura pelo recurso solicitado



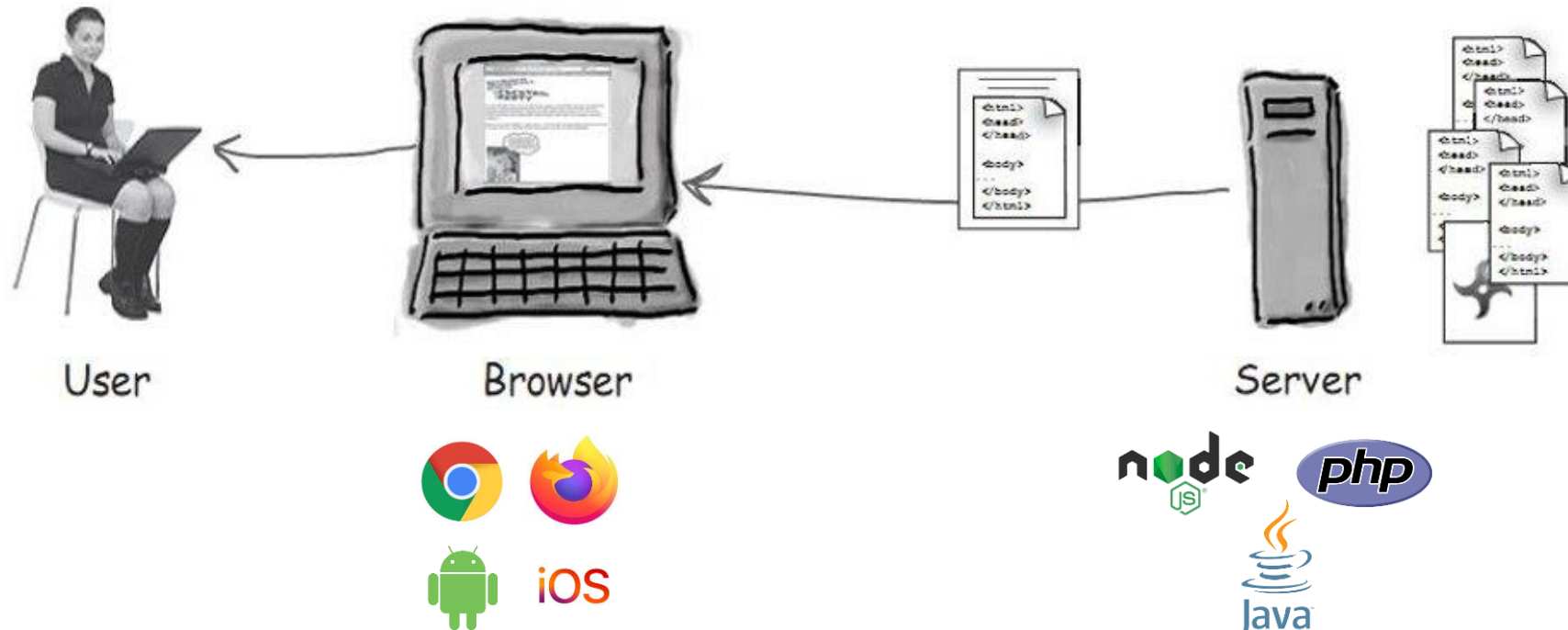
Server



RESPONSE (RESPOSTA)

5 O browser recebe a resposta, interpreta e exibe para o usuário

4 O servidor formata a resposta e envia pela rede

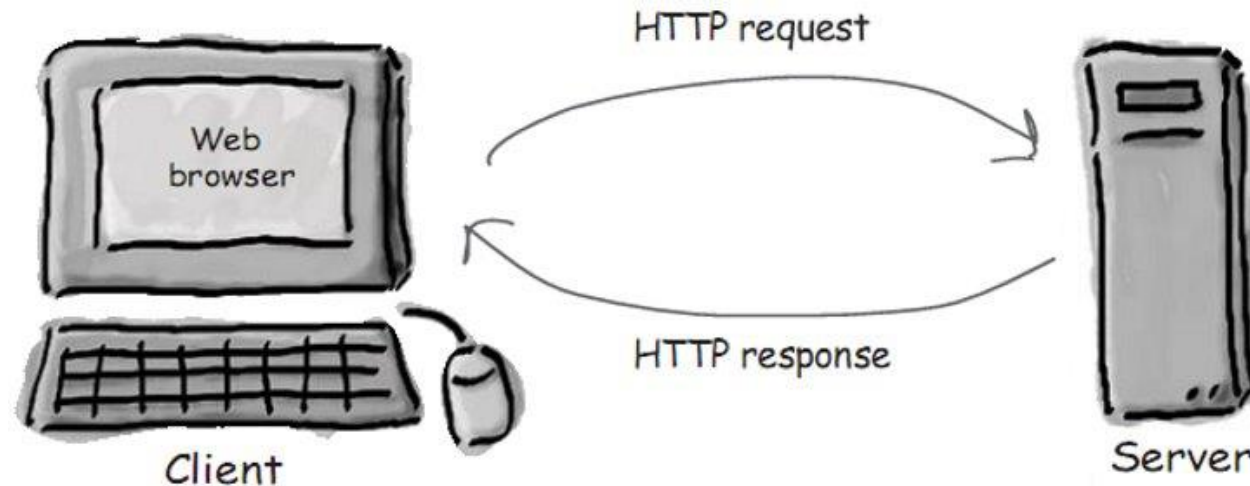


PROTOCOLLO HTTP

PROTOCOLO HTTP

Principais elementos de uma **requisição**:

- O método HTTP (a ação de ser executada).
- O recurso que será acessada (URL)
- Conteúdo (formulário, parâmetros, etc).

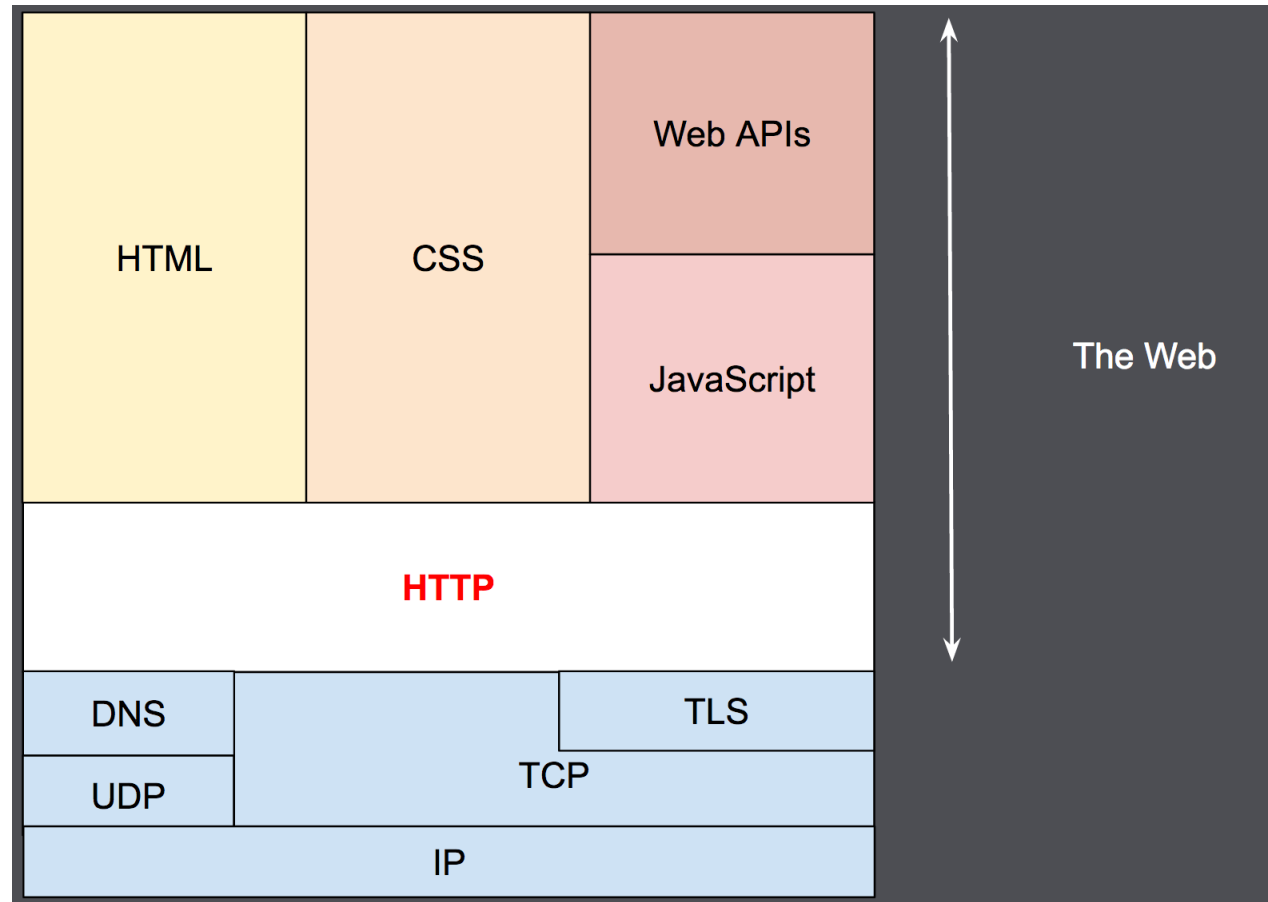


Principais elementos de uma **resposta**:

- Código de status (200, 404, 500, etc)
- Tipo do conteúdo (texto, imagem, HTML, etc)
- Conteúdo (texto, imagem, HTML, etc).



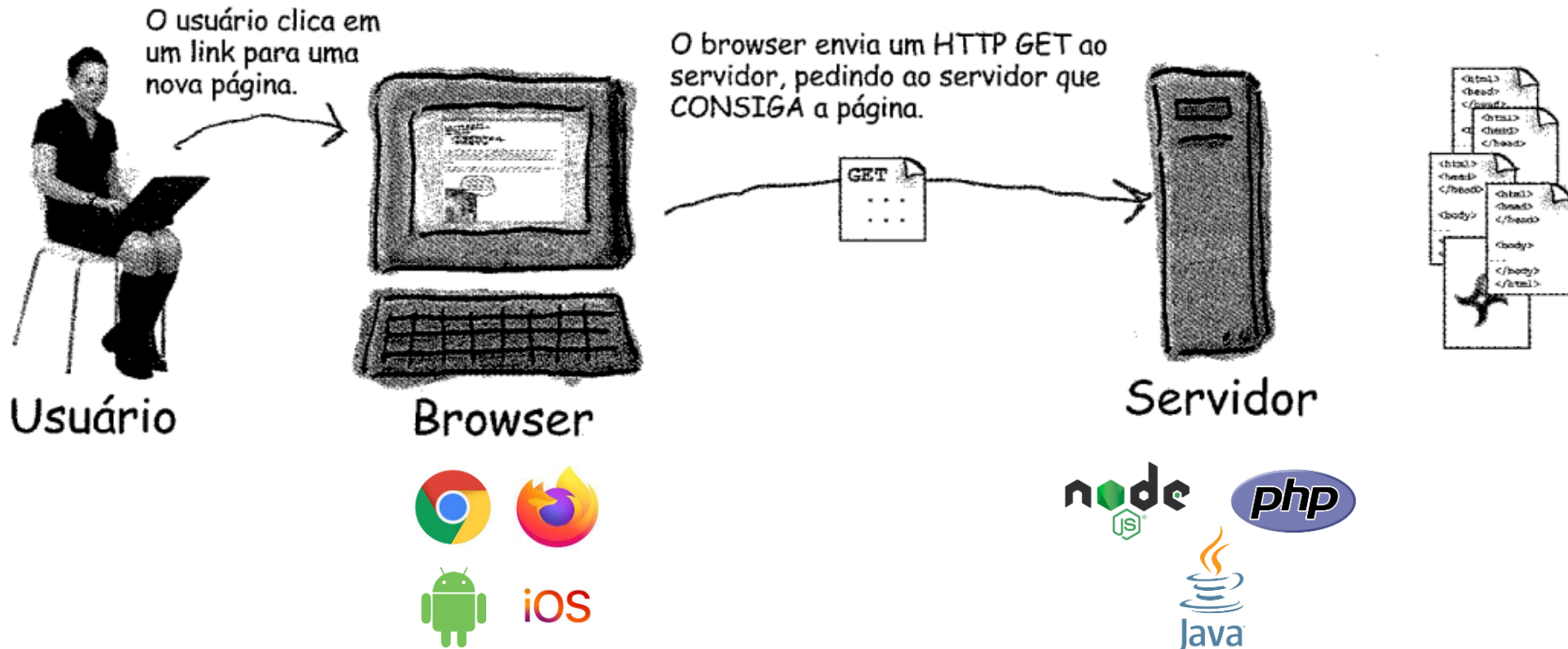
PROTOCOLLO HTTP



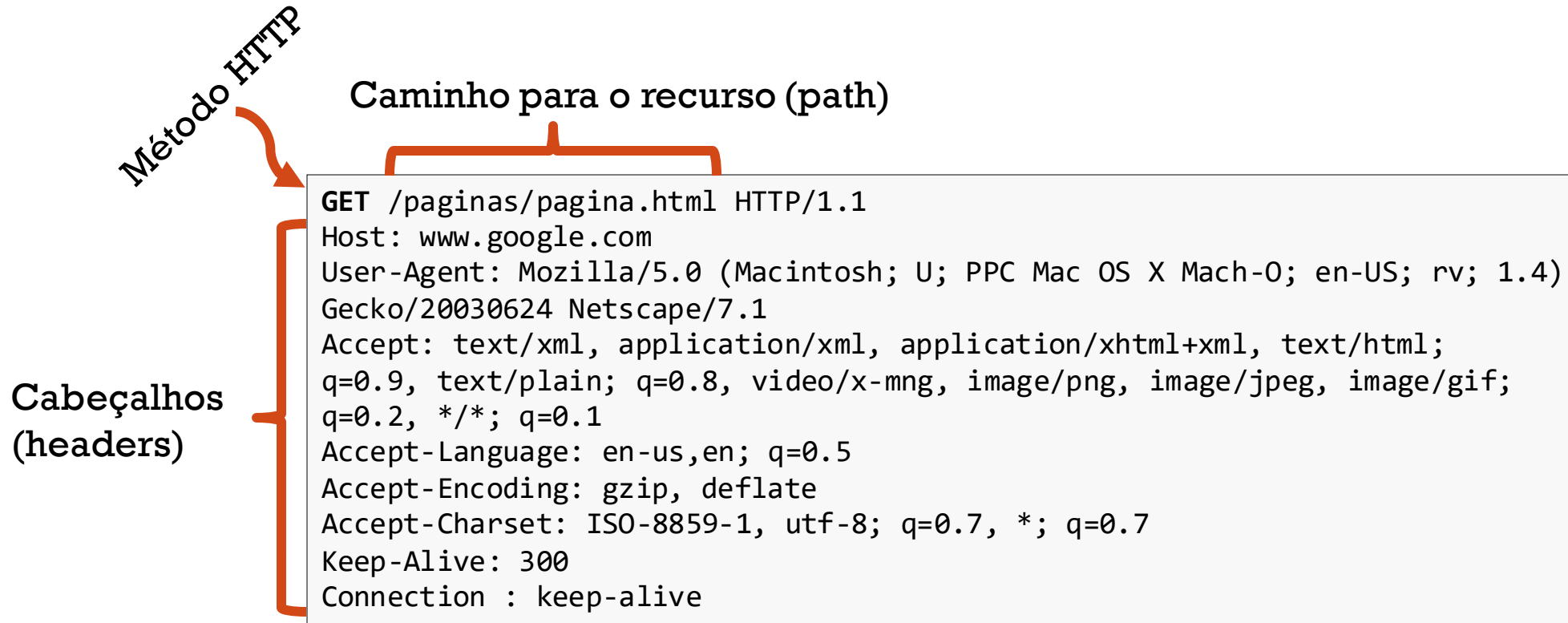
MÉTODOS GET E POST

MÉTODO GET

GET



MÉTODO GET





Sobre Nós

Smarppy, aplicações inteligentes.

Missão

Criar, de forma transparente, soluções robustas que contribuam para o objetivo de nossos clientes.

Visão

Ser reconhecida como uma empresa ágil, transparente e comprometida com a entrega de software de qualidade.

Valores

DevTools is now available in Portuguese! [Always match Chrome's language](#) [Switch DevTools to Portuguese](#) [Don't show again](#)

Elements Console Sources **Network** Performance Memory Application Security Lighthouse >> 4 1

Filter ☐ Invert ☐ Hide data URLs ☒ All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other ☐ Has blocked cookies

☐ Blocked Requests ☐ 3rd-party requests

☐ Use large request rows ☐ Group by frame

☒ Show overview ☐ Capture screenshots

10000 ms 20000 ms 30000 ms 40000 ms 50000 ms 60000 ms 70000 ms 80000 ms

Name

- smarppy.com
- css?family=Source+Sans+Pro:300,...
- js?id=UA-129645957-1
- main.053b2d4a.chunk.css
- runtime~main.d53d57e4.js
- 2.e0ee1d52.js
- main.7feef7b3.js
- js?id=G-NBX74N5X8R&l=dataLaye
- analytics.js
- c1a53753a8ceb0d1c88607be67e40
- maps?f=q&source=s_q&geocode...
- 6xK3dSBYKcSV-LCoeQqfX1RYOo3q
- 6xKwdSBYKcSV-LCoeQqfX1RYOo3.
- 6xKydsBYKcSV-LCoeQqfX1RYOo3i.
- 6xKydsBYKcSV-LCoeQqfX1RYOo3ik
- b7c9e1e479de3b53f1e4e30ebac24
- 9a2323f6d0ec91369e9319590d52b
- 05df0ea76b34489e684aa2df991e4.
- 788640584e7f8b2601482b061b2d.
- a8d6655a5be3c0cb39482bc8adf79
- 462a625fb61f49854c61b2298b62d
- 3c9b38d27e149bf8b15ab066d9cf8
- 412e0cec81297b730e94ac6ce225a
- 653c54f460eb272c44d70e23da44d
- 0b31b7b1b693559981560c819959
- 7c350f07ad36e19dcebb499701821
- 895e463315e65637bc7571acc71dc
- 2b1770d51662861709bc774518ce.
- f7dcba784d35c00ba95c9528019f.
- ...

59 requests 2.2 MB transferred 3.7

Headers Preview Response Initiator Timing Cookies

x-fastly-request-id: 3ae1be2a53ffff4be8610f52fc3add9aa37773540

x-github-request-id: EB7C:7314:54F1E4:5F3FDE:6485E115

x-proxy-cache: MISS

x-served-by: cache-cgh11163-CGH

x-timer: S1686495510.285321,V50,VE124

▼ Request Headers

:authority: smarppy.com

:method: GET

:path: /

:scheme: https

accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7

accept-encoding: gzip, deflate, br

accept-language: pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7

cache-control: no-cache

cookie: _ga=GA1.1.486299082.1625426458; _ga_6Z6YLH1MER=GS1.1.1625426458.1.1.1625426461.0; _ga_8EGSMH27W1=GS1.1.1625429213.2.1.1625429683.0

dnt: 1

pragma: no-cache

sec-ch-ua: "Opera";v="99", "Chromium";v="113", "Not-A.Brand";v="24"

sec-ch-ua-mobile: ?0

sec-ch-ua-platform: "Windows"

sec-fetch-dest: document

sec-fetch-mode: navigate

sec-fetch-site: none

sec-fetch-user: ?1

upgrade-insecure-requests: 1

user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36 OPR/99.0.0.0


MÉTODO GET



Espera um momento...
Eu poderia jurar que já vi
solicitações GET que enviavam
alguns dados por parâmetros ao
servidor.

MÉTODO GET

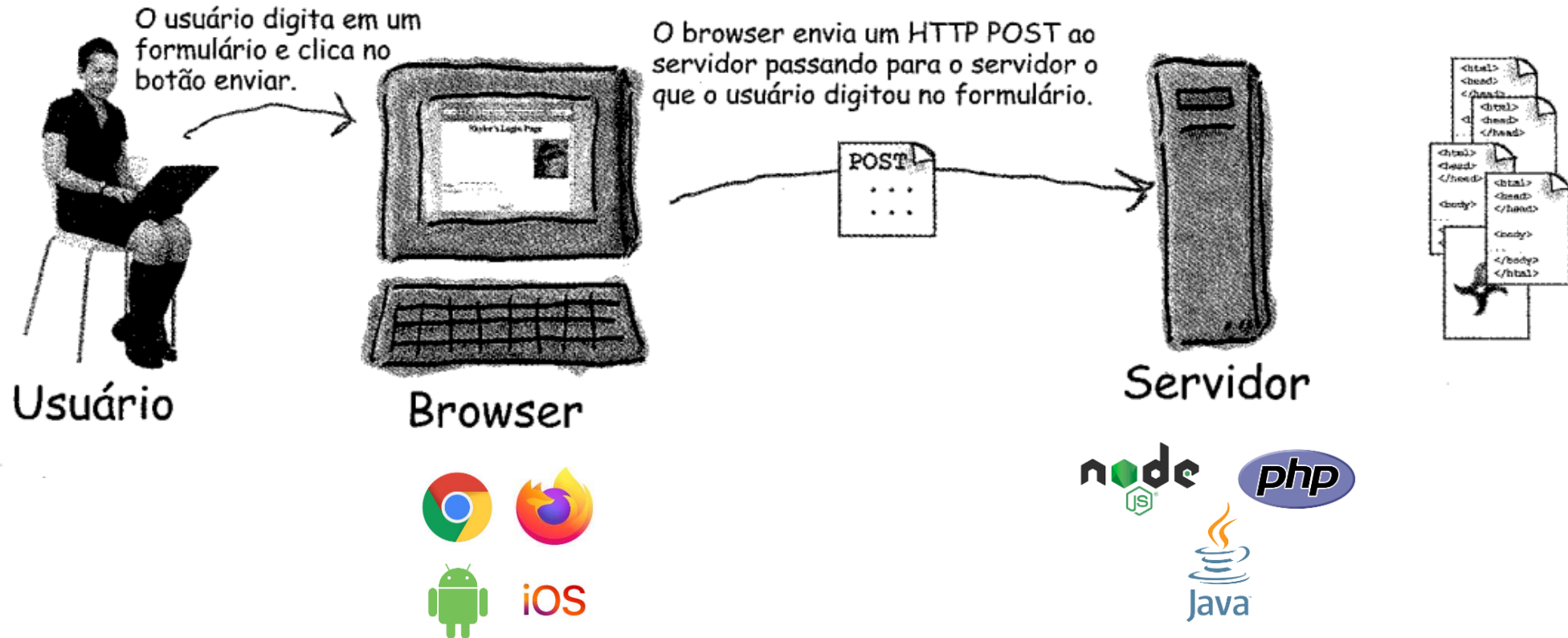
Em uma solicitação GET, os parâmetros (se existir algum) serão anexados à primeira parte da solicitação URL, iniciando-se por uma "?". Os parâmetros são separados usando-se o "&".



```
GET /paginas/pagina.html?id=1&aluno=Jack HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-0; en-US; rv; 1.4)
Gecko/20030624 Netscape/7.1
Accept: text/xml, application/xml, application/xhtml+xml, text/html;
q=0.9, text/plain; q=0.8, video/x-mng, image/png, image/jpeg, image/gif;
q=0.2, */*; q=0.1
Accept-Language: en-us,en; q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1, utf-8; q=0.7, *; q=0.7
Keep-Alive: 300
Connection : keep-alive
```

MÉTODO POST

POST



METODO POST

Método HTTP

Caminho para o recurso (path)

Cabeçalhos
(headers)

Conteúdo da
requisição
(payload/body)

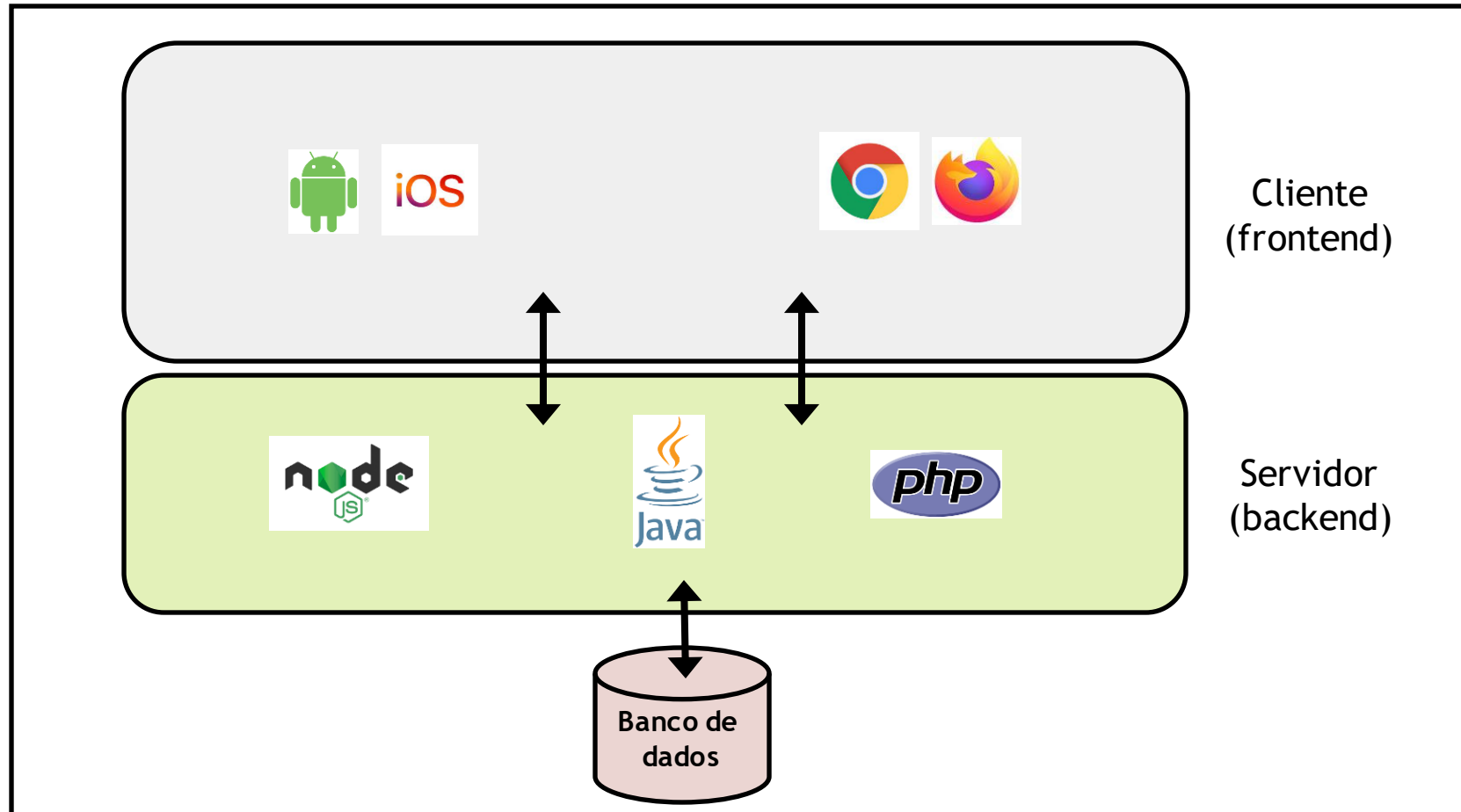
```
POST /paginas/recebeDados HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US; rv; 1.4)
Gecko/20030624 Netscape/7.1
Accept: text/xml, application/xml, application/xhtml+xml, text/html;
q=0.9, text/plain; q=0.8, video/x-mng, image/png, image/jpeg, image/gif;
q=0.2, */*; q=0.1
Accept-Language: en-us,en; q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1, utf-8; q=0.7, *; q=0.7
Keep-Alive: 300
Connection : keep-alive

aluno=Jack&idade=17
```

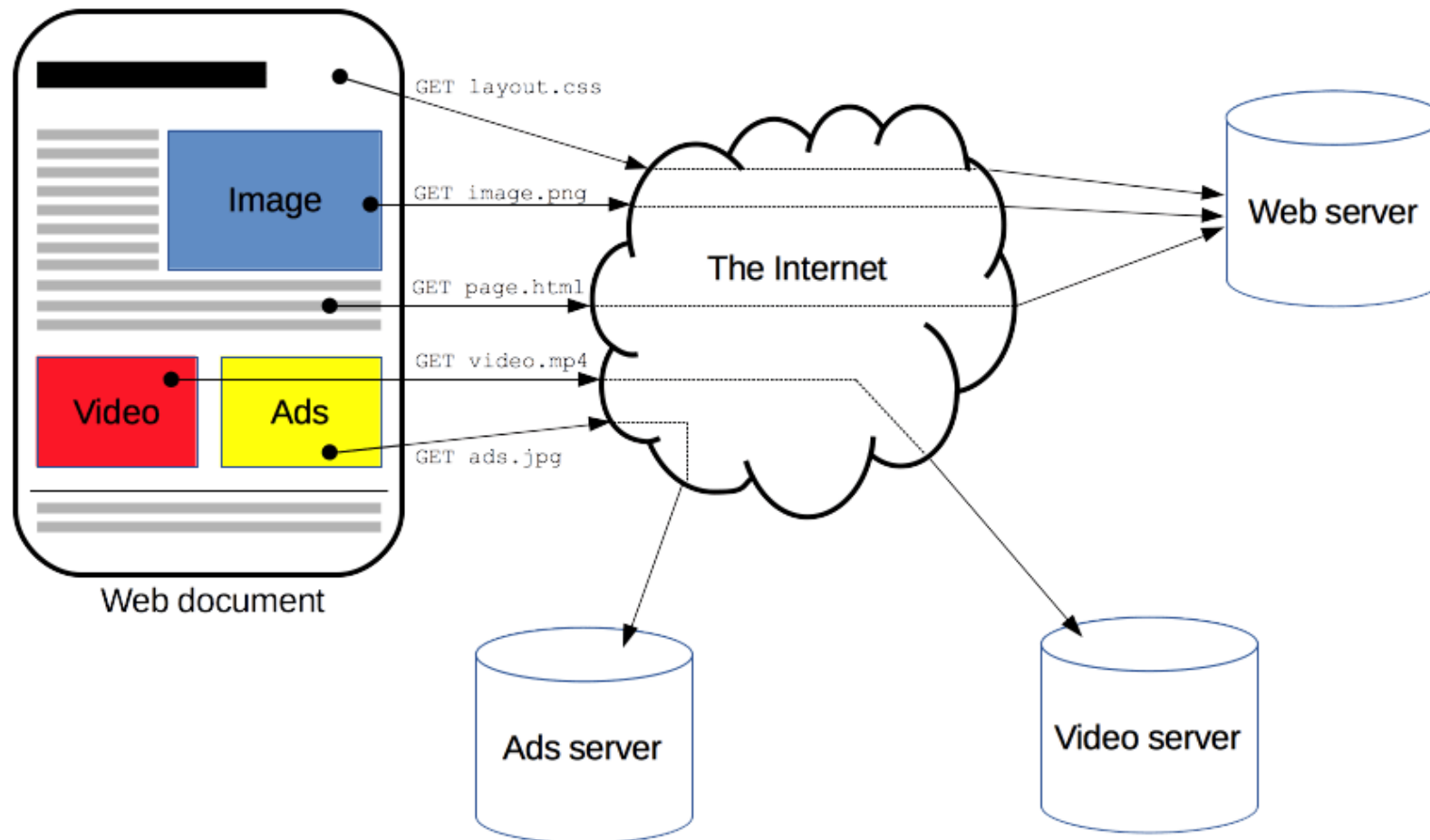
EXERCÍCIO

- Um usuário entrando com login e uma senha em um formulário
- Um usuário solicitando uma nova página via hyperlink
- Um usuário em uma sala de bate-papo enviando uma resposta
- Um usuário clica no botão "próximo" para ver a próxima página
- Um usuário clica no botão de "Sair" num site seguro de um banco
- Um usuário clica em "Voltar" no browser
- Um usuário envia nome e endereço em uma página de contato
- Um usuário faz uma escolha em um botão de seleção

CARACTERÍSTICAS DE UMA APLICAÇÃO WEB



CARACTERÍSTICAS DE UMA APLICAÇÃO WEB



REFERÊNCIAS

- ARPANET: <https://pt.wikipedia.org/wiki/ARPANET>
- Internet: <https://pt.wikipedia.org/wiki/Internet>
- TCP/IP: <https://pt.wikipedia.org/wiki/TCP/IP>
- Protocolo HTTP: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>
- Métodos HTTP: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>

DÚVIDAS?



Douglas Nassif Roma Junior

 /douglasjunior

 /in/douglasjunior

 nassifroma@gmail.com

Slides: <https://github.com/douglasjunior/Catalisa-DB1-2023>