Vídeo Aula:
https://app.rocketseat.com.br/node/guia-estelar-de-http

Visão Geral

- O que é HTTP?
    - HyperText Transfer Protocol: Protocolo de Transferência de HiperTexto;
    - Protocolo: são conjuntos de regras;
    - HiperTexto são textos que podem conter imagens, links para outros textos, etc, cujos são transferidos através do protocolo.
- O que faz?
    - Permite a troca de informações/dados na internet;
    - Utilizamos isso diariamente, por exemplo, acessar o site do google: ao inserir a url no browser estou fazendo uma requisição, a requisição será enviada para o servidor do google através do protocolo HTTP e como resposta terei um HTML (página da google).

Conceitos

HTTP

- Simples: Deve ser legível para qualquer pessoa;
- Baseado em cliente/servidor, ou seja, o servidor irá servir o cliente. Exemplo: eu cliente cheguei em uma lanchonete e pedi um lanche, o servidor para me servir me entrega o lanche;
- Stateless:
    - Não guarda estado/ informação;
    - Não guarda relações entre as conexões: uma requisição não depende da outra;
    - Faz uso de sessões: manter um usuário logado armazenando a sessão do mesmo no cookie/storage. Essa informação é enviada para as próximas requisições após usuário ter realizado login.
- Extensível:
    - Header (Cabeçalho): Podemos utilizar o cabeçalho para fazer diversas trocas de informações entre o cliente e servidor;
    - Body (corpo): podemos receber e enviar muitas informações através dele.

Cliente

- Quem é o cliente?
    - Uder-Agent: Browser, curl;
    - Entidade que dá início a comunicação.
- Ações do cliente:
    - Pedidos: GET, POST, PUT, DELETE.

Servidor

- Quem é o servidor?
    - É uma máquina, cuja pode estar em qualquer lugar do mundo e está preparada para ouvir as requisições e processar os dados para a resposta;
    - Vários servidores podem representar um único computador, ou seja, um computador pode ter vários servidores rodando nele. Ou posso ter um computador representando vários servidores;
    - É responsável por fornecer a resposta e dentro do header dessa resposta existe o Status Code - Status da Resposta (404, 500), etc...;
    - Essa resposta pode conter também um body.

Proxies

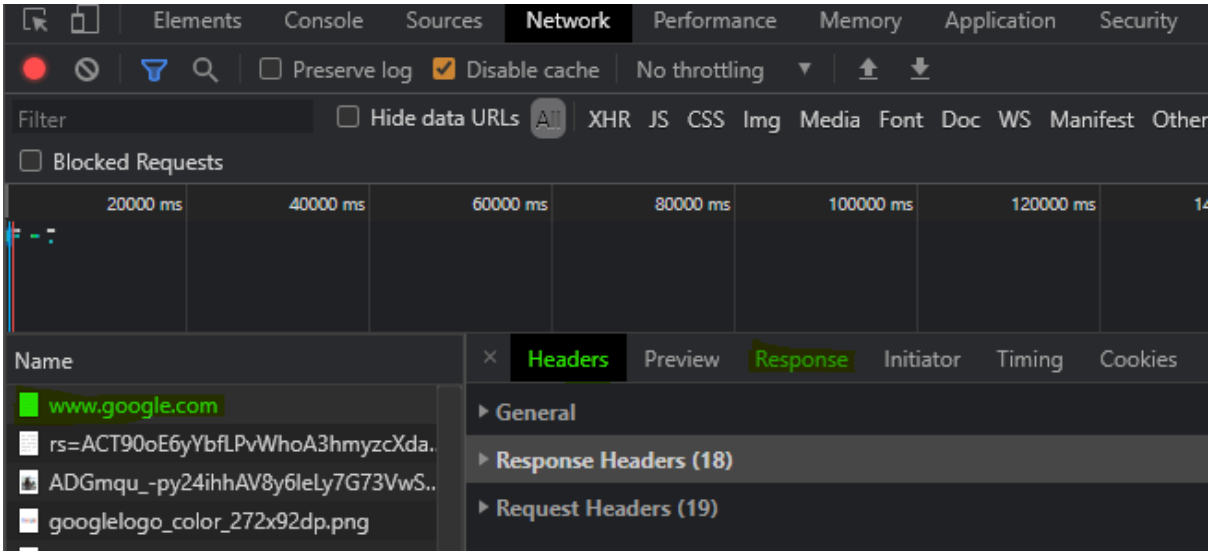
- Quem são os proxies?
    - Fica entre o cliente e o servidor auxiliando no transporte da mensagem, ex: roteador de wifi;
- Funções:
    - Cache: para nos dar a informação de forma mais rápida;
    - Filtro (antivírus, controle parental): Por exemplo, configurar o roteador para bloquear determinados sites;
    - Load balancing: distribuição da carga;
    - Autenticação;
    - Autorização.

Ferramentas/ Recursos

Algumas ferramentas e recursos que vamos utilizar nessa trilha: DevTools, cURL e JSON Server.

DevTools

- É uma ferramenta do próprio Google;
- Como usar?
    - Abrir o endereço/recurso que deseja acessar: https://www.google.com/;
    - Aperte a tecla F12 do seu teclado;
    - Algumas configurações que você pode selecionar:
        - Preserv log: armazena todas as informações durante minha navegação na web;
        - Disable cache: Não pegar nada que tem na minha máquina, apenas da internet, ou seja, não utiliza o cache.
    - Vá até a aba Network e encontre a url que você digitou no log de acesso: https://www.google.com/ para que possa visualizar:
        - Headers;
        - Request;
        - Response, etc...



cURL

- É uma ferramenta que nos permite utilizar diversos protocolos: DICT, FILE, FTP, FTPS, GOPHER, GOPHERS, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP. curl supports SSL certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form based upload, proxies, HTTP/2, HTTP/3, cookies, user+password authentication (Basic, Plain, Digest, CRAM-MD5, SCRAM-SHA, NTLM, Negotiate and Kerberos), file transfer resume, proxy tunneling e mais;
- Para Windows recomenda-se utilizar o git bash para execução dos seus comandos;
- cURL representa o cliente, ou seja, realiza as requisições e obtém respostas;
- Alguns comandos:

Comandos	Descrição
curl https://google.com	Fazendo uma requisição e ontém a resposta
curl -i https://google.com	Obtendo os headers de resposta e o corpo da requisição
curl -v https://google.com	Obtendo todos os headers, tanto de saída (>), quanto de chegada(<)

JSON Server

Vamos instalar o JSON Server, servidor que irá nos responder dados no formato JSON.

- Instalando:
  - Abra o Node.js prompt de comando;
  - Digite: npm install -g json-server;
  - Crie um diretório utilizando o seguinte comando: mkdir http-course. Ou crie uma pasta manualmente no diretório cujo seu prompt json está apontando, por exemplo: C:\Users\Vanessa>;
  - Acesse o diretório no prompt de comando Node.js utilizando o seguinte comando: cd + nomeDiretorio;
  - Crie um arquivo dentro do diretório utilizando o seguinte comando: vim db.json. Ou crie um arquivo com extensão json de forma manual;
  - Insira o seguinte conteúdo dentro do arquivo:

```
{
  "posts": [
    { "id": 1, "title": "json-server", "author": "typicode" }
  ],
  "comments": [
    { "id": 1, "body": "some comment", "postId": 1 }
  ],
  "profile": { "name": "typicode" }
}
```

  - Inicie o servidor com o arquivo criado utilizando o seguinte comando: http-course>json-server --watch db.json;

Ajuda, acesse: https://github.com/typicode/json-server

URI

- Conceito: Uniform Resource Identifier - identificador de recurso utilizado de maneira uniforme/única, através do nome ou localização.

Resource

- Recurso é o alvo do pedido HTTP;
- Pode ser qualquer coisa identificável. Exemplos:
    - Digital: e-mail, acessado via protocolo malito:email@dominio.com;
    - Abstrata:
        - Autenticação: login e senha do usuário;
        - Sessão: Após o usuário autenticado é mantida uma sessão dessa autenticação.
    - Física:
        - Produtos;
        - Usuários.

URL

- URL: Uniform Resource Locator;
- É um tipo de localizador através de um endereço, cujo me proporciona o acesso a um determinado recurso;
- Deve ser composta por 2 componentes obrigatórios e 5 opcionais (não significa que irei utilizar ou não, mas fará uma diferença para que eu possa encontrar o recurso). Exemplo de URL para descrever os campos obrigatórios e opcionais - https://www.rocketseat.com.br/blog:
    - Obrigatórios:
        - Protocolo: https;
        - Domínio: //www.rocketseat.com.br
    - Opcionais:
        - Subdomínio: www;
        - Path: /blog (caminho);
        - Parâmetros: https://www.youtube.com/**watch?v=dVzJ3bx68FA&list=PLx4x_zx8csUglgKTmgfVFEhWWBQCasNGi** **(? + chave = valor);**
        - Porta (local do servidor que disponibilizará o acesso a determinado recurso): 80 http/ 443 https, etc... Exemplo: http://127.0.0.1:**3333**/index.html;
        - Âncora (local/lugar dentro de um documento): http://127.0.0.1:3333/index.html**#algumlugar.**

URN/URI

- Uniform resource name;
- Encontramos um recurso pelo nome sem saber a sua localização;
- Exemplo: urn:isbn:0451450523 / urn:oasis:names:specification:docbook:dtd:xml:4.1.2;
- Sempre começa com urn;
- No exemplo estamos buscando um livro.

Mensagens

- Tanto no request quanto no response nós temos as mensagens para comunicar o cliente com o servidor;
- Versões do HTTP:
    - HTTP 1.1: Ainda é muito utilizada, bem legível e textual;
    - HTTP 1.2: É feita de estrutura binária (não legível), o que permite maior otimização e compressão. No final é a mesma coisa que a versão 1.1, só é mascarada.

Request

O que compõe o request:

- Método: verbo HTTP que irá dizer a intenção do pedido (deletar, cadastrar, verificar se a conexão está aberta, etc..);
- Versão do protocolo;
- URI;
- Body: dependendo do tipo de método pode conter ou não;
- Headers;
- Exemplo utilizando DevTools (www.google.com):

```
Request URL: https://www.google.com/
Request Method: GET
Status Code: 200
Remote Address: [2800:3f0:4001:802::2004]:443
Referrer Policy: strict-origin-when-cross-origin
```

- Exemplo utilizando cURL(www.google.com):

Comando	Resposta
curl -v https://google.com	***#REQUISIÇÃO*** **> GET / HTTP/2 *#VERSÃO HTTP*** **> Host: google.com** **> user-agent: curl/7.75.0** **> accept: */*** ***#RESPOSTA*** **< HTTP/2 301** **< location: https://www.google.com/** **< content-type: text/html; charset=UTF-8** **< date: Sun, 28 Mar 2021 14:03:02 GMT** **< expires: Tue, 27 Apr 2021 14:03:02 GMT** **< cache-control: public, max-age=2592000** **< server: gws** **< content-length: 220** **< x-xss-protection: 0** **< x-frame-options: SAMEORIGIN** **< alt-svc: h3-29=":443"; ma=2592000,h3-T051=":443";** **ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443";** **ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"** **<** ***#CORPO DA RESPOSTA*** **<TITLE>301 Moved</TITLE></HEAD><BODY>** **<H1>301 Moved</H1>** **The document has moved** **<A HREF="https://www.google.com/">here</A>.** **</BODY></HTML>**

Response

O que compõe o response:

- Versão do protocolo;
- Status Code: se deu tudo certo ou algum erro no meio do caminho;
- Headers e;
- Status Mensagem (corpo);
- Exemplo utilizando DevTools (www.google.com): Faça você mesmo!!!!
- Exemplo utilizando cURL(www.google.com): Faça você mesmo através do comando curl -i https://www.google.com!!!!

Methods

- São conjuntos de métodos HTTP;
- É a identificação de uma ação que o cliente deseja operar;
- Podem ser chamados de Verbos HTTP;

- Cada um possui sua semântica/significado/própria maneira de operar;
- Características:
    - Nem todos são seguros, com exceção dos de apenas leitura, pois, não altera o estado no servidor, ou seja, quando eu acesso hhtp://www.google.com.br eu estou apenas recebendo a página para leitura, não estou solicitando nenhuma alteração, não há carga extra no servidor (GET, HEAD, OPTIONS);
    - Nem todos são idempotentes. O que seria um método idempotente:
        - Ao executar o método a resposta deverá ser sempre a mesma;
        - Quais os métodos idempotentes?: Todos os seguros, PUT e DELETE;
        - Status code poderá ser diferente;
        - O servidor tem a responsabilidade de retornar os dados da mesma maneira;
        - Essa especificação não é a garantia de que todos os servidores irão aplicar o conceito idempotente corretamente.

    - Nem todos possuem cacheable, ou seja, mantém resposta em memória para ser utilizada em próximas requisições.

OPTIONS

- Serve para nos informar a disponibilidade da requisição;
- É seguro;
- É idempotente;
- Não possui corpo na requisição ou na resposta;
- Não é utilizado em formulário HTML;
- Não possui cacheable.

Comando	curl -X OPTIONS http://localhost:3000/posts -i
Retorno	% Total % Received % Xferd Average Speed Time Time Time Current Dload Upload Total Spent Left Speed 0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0HTTP/1.1 204 No Content X-Powered-By: Express Vary: Origin, Access-Control-Request-Headers Access-Control-Allow-Credentials: true **Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,POST,DELETE** Content-Length: 0 Date: Sun, 28 Mar 2021 15:17:45 GMT Connection: keep-alive Keep-Alive: timeout=5

GET

- Serve para pegar um recurso;

- Somente vai receber dados;
- Características:
    - É seguro;
    - É idempotente;
    - Não posso enviar nada no body, apenas receber;
    - É cacheable, ou seja, pode manter respostas em cache para serem utilizadas em novas requisições;
    - É usado em formulários HTML.

Comando	Resposta

curl http://localhost:3000/posts	***#body da resposta***
	[
	  {
	  "id": 1,
	  "title": "json-server",
	  "author": "typicode"
	  }
	]

curl -v http://localhost:3000/posts
#ver detalhes da requisição
	% Total % Received % Xferd Average Speed Time Time Time Current
	Dload Upload Total Spent Left Speed
	0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0* Trying ::1:3000...
	* Trying 127.0.0.1:3000...
	#fazendo a conexão na url e porta
	* Connected to localhost (127.0.0.1) port 3000 (#0)
	#cabeçalho: método/rota/versão HTTP
	> GET /posts HTTP/1.1
	> Host: localhost:3000
	> User-Agent: curl/7.75.0
	> Accept: */* ***#aceita qualquer resposta***
	>
	* Mark bundle as not supporting multiuse
	#resposta
	< HTTP/1.1 200 OK
	< X-Powered-By: Express
	< Vary: Origin, Accept-Encoding
	< Access-Control-Allow-Credentials: true
	< Cache-Control: no-cache
	< Pragma: no-cache
	< Expires: -1
	< X-Content-Type-Options: nosniff
	< Content-Type: application/json; charset=utf-8
	< Content-Length: 77
	< ETag: W/"4d-49G7XbVRP2NKipc5uj9Z4hcUq3Y"
	< Date: Sun, 28 Mar 2021 15:28:32 GMT
	< Connection: keep-alive
	< Keep-Alive: timeout=5
	
	{ [77 bytes data]
	100 77 100 77 0 0 313 0 --:--:-- --:--:-- --:--:-- 314[
	#corpo da resposta ao meu pedido
	  {
	  "id": 1,

	"title": "json-server", "author": "typicode" }]

HEAD

- Se assemelha ao GET, porém, recebemos apenas o cabeçalho;
- Características:
    - É seguro;
    - É idempotente;
    - Não possui body tanto no response quanto no request;
    - Não é utilizado em formulários HTML;
    - Podemos fazer cach para termos uma resposta mais rápida.

Comando	Resposta
curl -I http://localhost:3000/posts	% Total % Received % Xferd Average Speed Time Time Time Current Dload Upload Total Spent Left Speed 0 77 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0HTTP/1.1 200 OK X-Powered-By: Express Vary: Origin, Accept-Encoding Access-Control-Allow-Credentials: true Cache-Control: no-cache Pragma: no-cache Expires: -1 X-Content-Type-Options: nosniff Content-Type: application/json; charset=utf-8 ***#tamanho do conteúdo. Posso usar antes de um GET para ver o tamanho do conteúdo*** Content-Length: 77 ETag: W/"4d-49G7XbVRP2NKipc5uj9Z4hcUq3Y" Date: Sun, 28 Mar 2021 15:55:46 GMT Connection: keep-alive Keep-Alive: timeout=5

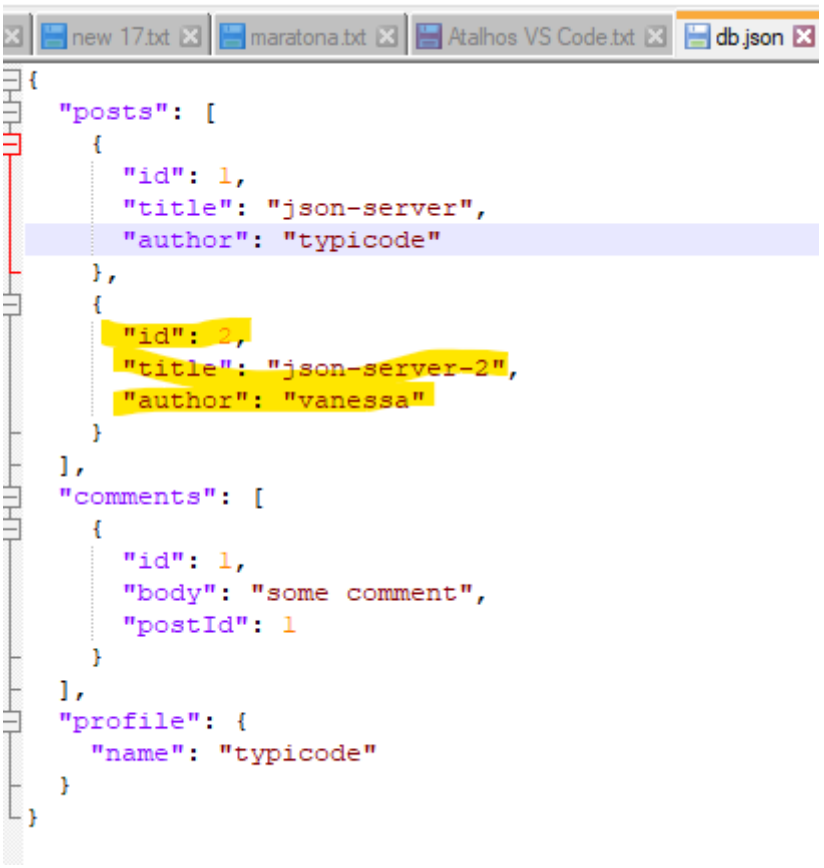
POST

- Significa cadastrar/incluir alguma coisa/ um recurso em algum lugar;
- Caracterísitcas:
    - Não é seguro;
    - Não é idempotente, pois, sempre que eu executá-lo, apesar de usar a mesma rota, vamos ter cadastros diferentes (body se altera no request e response);
    - O body:
        - Irá existir na request: envio do recurso;
        - Pode ou não existir na resposta, ou seja, posso receber um body ou apenas o status code;
    - É utilizado em formulários HTML;
    - Pode ser cacheble.

Comando	Resposta
# -d significa dados, é o body da minha	% Total % Received % Xferd Average

requisição
# -H é o cabeçalho, estou avisando que os dados estão no formato json 	
# -X estou avisando o tipo de método que estou usando 	
#Observe que a rota é a mesma utilizada no GET, porém, como estou informando que o método é do tipo POST o processamento no servidor será diferente
 	
$ curl -d '{ "id": 2, "title": "json-server-2", "author": "vanessa" }' -H "Content-type: application/json" -X POST http://localhost:3000/posts
	Speed Time Time Time Current
	Dload Upload Total
	Spent Left Speed
	100 122 100 64 100 58 200 181 --:--:--
	--:--:-- --:--:-- 383{
	"id": 2,
	"title": "json-server-2",
	"author": "vanessa"
	}

*Observe a alteração no arquivo db.json



PUT

- Serve para criar/atualizar um recurso;
- Verbo: POST;
- A diferença entre PUT e POST:
  - PUT é idempotente, ou seja, ele não faz alteração na resposta;
  - É muito mais utilizado para atualizar;
  - Seu status code para criação é 201 e não 200. Sendo 200(OK com conteúdo do body) /204(OK e não possui conteúdo de retorno no body) para atualização.
- Características:
  - Não é seguro;

- É idempotente;
- Não tem body na resposta mas tem no pedido;
- Não é utilizado em formulários e;
- Não é cacheble.

Comando	Resposta

$ curl -d '{ "name": "vanessa"}' -H "Content-type: application/json" -X PUT http://localhost:3000/profile	***#Observe que esse modelo de PUT deu uma resposta no body***
	% Total % Received % Xferd Average Speed Time Time Time Current
	Dload Upload Total Spent Left Speed
	100 43 100 23 100 20 92 80 --:--:-- --:--:-- --:--:-- 171{
	"name": "vanessa"
	}

#Acrescentando -i consigo visualizar as informações do header de resposta
$ curl -d '{ "name": "vanessa"}' -H "Content-type: application/json" -X PUT http://localhost:3000/profile -i
	% Total % Received % Xferd Average Speed Time Time Time Current
	Dload Upload Total Spent Left Speed
	#Status code
	100 43 100 23 100 20 95 83 --:--:-- --:--:-- --:--:-- 178HTTP/1.1 **200 OK**
	X-Powered-By: Express
	Vary: Origin, Accept-Encoding
	Access-Control-Allow-Credentials: true
	Cache-Control: no-cache
	Pragma: no-cache
	Expires: -1
	X-Content-Type-Options: nosniff
	Content-Type: application/json; charset=utf-8
	Content-Length: 23
	ETag: W/"17-4/EB5CoNnSMV1zDjp/gIvVqF+k8"
	Date: Sun, 28 Mar 2021 16:21:50 GMT
	Connection: keep-alive
	Keep-Alive: timeout=5
	
	{
	"name": "vanessa"
	}

PATCH

- Serve para modificar parcialmente um recurso;
- **No mundo real** dificilmente alguém utiliza o PATCH. Geralmente utilizam o PUT mesmo;
- Qual sua diferença para o PUT?: o PUT serve para atualizar o recurso por inteiro, já o PATCH apenas parcialmente.
- Características:
  - Não é seguro;
  - Não é idempotente;
  - Possui body tanto no envio quanto na resposta;
  - Não é utilizado em formulários e;
  - Não é cacheble.

Comando	Resposta

$ curl -d '{ "title": "my-title"}' -H "Content-type: application/json" -X PATCH http://localhost:3000/posts/1	***#Observe que apenas o título foi alterado***
	% Total % Received % Xferd Average Speed Time Time Time Current
	Dload Upload Total Spent Left Speed
	100 82 100 60 100 22 239 87 --:--:-- --:--:-- --:--:-- 328{
	"id": 1,
	"title": "my-title",
	"author": "typicode"
	}

DELETE

- Serve para remover um recurso;
- Características:
    - Não é seguro;
    - É idempotente;
    - Body, pode ser que seja necessário enviar no pedido e pode ser que tenhamos um na resposta;
    - Não é usado em formulários HTML;
    - Não fazemos cache dele.

Comando	Resposta

#Apenas consultando o recurso que vamos deletar para validar que o mesmo existia
curl http://localhost:3000/posts/2
	% Total % Received % Xferd Average Speed Time Time Time Current
	Dload Upload Total Spent Left Speed
	100 64 100 64 0 0 250 0 --:--:-- --:--:-- --:--:-- 251{
	"id": 2,
	"title": "json-server-2",
	"author": "vanessa"
	}

$ curl -X DELETE http://localhost:3000/posts/2	***#Não temos resposta, mas podemos ver na execução do servidor que a requisição foi executada com sucesso: DELETE /posts/2 200 37.784 ms - 2***
#Consultando o recurso deletado para validar sua deleção $ curl http://localhost:3000/posts/2 -i	***#Não temos resposta, mas podemos ver no status code que o recurso não existe mais*** % Total % Received % Xferd Average Speed Time Time Time Current Dload Upload Total Spent Left Speed 100 2 100 2 0 0 8 0 --:--:-- --:--:-- --:--:-- 8HTTP/1.1 **404 Not Found** X-Powered-By: Express Vary: Origin, Accept-Encoding Access-Control-Allow-Credentials: true Cache-Control: no-cache Pragma: no-cache Expires: -1 X-Content-Type-Options: nosniff Content-Type: application/json; charset=utf-8

	Content-Length: 2 ETag: W/"2-vyGp6PvFo4RvsFtPoIWeCReyIC8" Date: Sun, 28 Mar 2021 16:54:00 GMT Connection: keep-alive Keep-Alive: timeout=5 {}

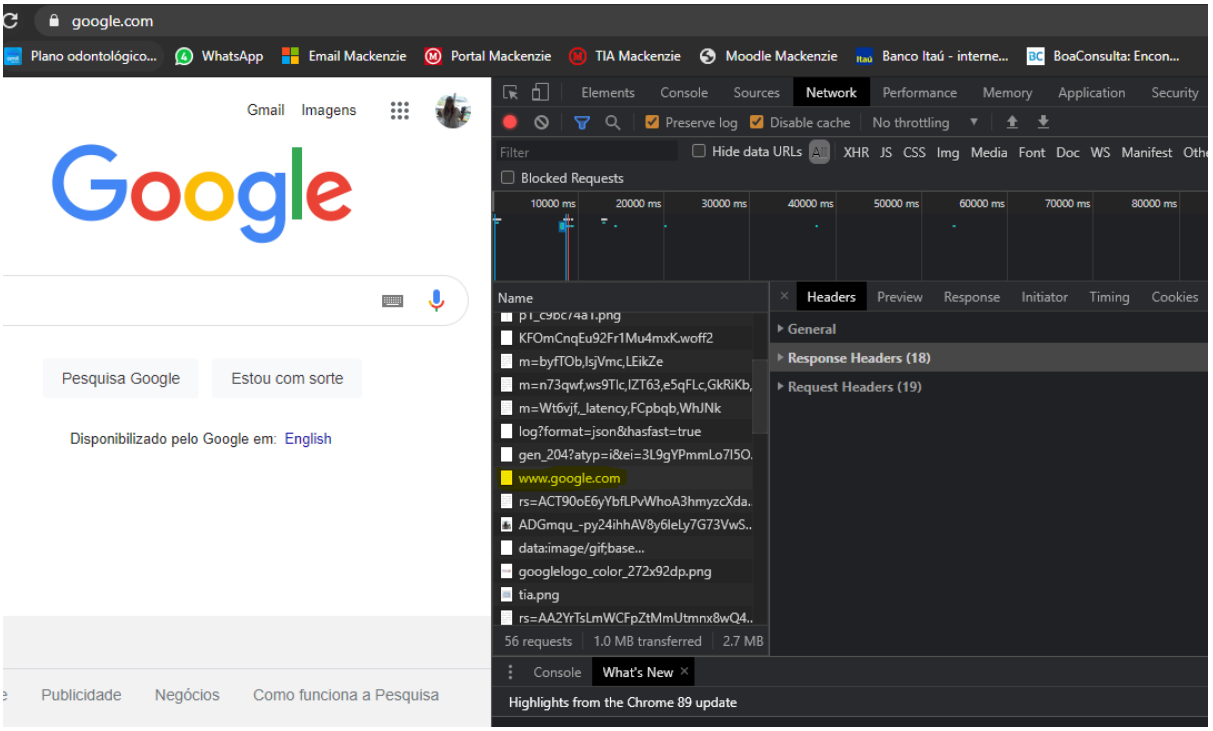
Headers

O que são Headers?

- Na sua tradução significa cabeçalhos;
- São informações adicionais que utilizamos para pedido ou resposta;
- Geralmente ele é construído da seguinte forma: nome : value:
    - Content-type: application/json;
    - Content-Type: text/html;
    - content-Type: text/html;
    - content-type: text/html.

Entendendo pelo contexto

- Headers por contexto:
    - Abra o google: www.google.com.br;
    - Abra o devTool (F12);
    - Abra o google: www.google.com.br;
    - Vá até o link www.google.com.br:



General

- Headers do General servem tanto pro request quanto para o response:

#URL do nosso pedido
Request URL: https://www.google.com/
#Método do nosso pedido
Request Method: GET
#Status da nossa requisição 200 -> OK
Status Code: 200
#Endereço físico da máquina
Remote Address: [2800:3f0:4001:818::2004]:443
#Política para referenciados: ao clicarmos em um link o mesmo nos levará para algum lugar, ou seja, de onde veio essa requisição? Do twitter?
Referrer Policy: strict-origin-when-cross-origin

Request

#Autoridade do Google.com
:authority: www.google.com
#Método
:method: GET
#Caminho: /
:path: /
#Protocolo
:scheme: https
#O que é aceito nesse pedido
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=0.9
#Estamos comprimindo o corpo dessa requisição em gzip. Quem compacta isso é o Chrome
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
#Rastros/ detalhes que quero deixar para podermos utilizar nas próximas requisições
cookie:
ANID=AHWqTUklU83j9kxTMYajhm4dvHAsJfoC6cfF147Yn8m3HNDpWzLVsot1Pl_1Q68K;
S=billing-ui-v3=3BW78lxsHgn1UhvUXbis6ts6Fz0hkyWs:billing-ui-v3-efe=3BW78lxsHgn1UhvUXbis6ts6Fz0hkyWs; SEARCH_SAMESITE=CgQI8pEB;
OTZ=5889581_68_64_73560_68_416340;
SID=7ge14_kGuM3nnANJstqCJ-8WB3fpcU4NSCI_HhaBfV7UvIBrBC7VphS7993FP8Hpv9ae5g.;
__Secure-3PSID=7ge14_kGuM3nnANJstqCJ-8WB3fpcU4NSCI_HhaBfV7UvIBriLjATFIq_lWaqmKiQL6QdA.; HSID=AQtn5H0x4KqhPrBMk; SSID=A_ckm7r-4Ntb_egXp;
APISID=dC6ipqAn5AuDieTS/A5kEBnxmr3_Syfl8-;
SAPISID=11-7yXhwXvsXGouk/AxzpI1v9rdszNGo_v;
__Secure-3PAPISID=11-7yXhwXvsXGouk/AxzpI1v9rdszNGo_v;
NID=212=ufPIHkfEoeq4ev8Xa1iSWEEKjKbVPEJ7P89EpLrxRJaS7S4Je6ljQvr8Y1FVDM8Vvu3w_t0n0om39FbFdw1n-BHH8ixrmQbgFKC3iM4DotHDJ0Dmb6JuIvX0VyeySbWNJnTZ91yCJynU2MWV-BNabzuT2iUNlx8lg8lFhzil5PQstDa2mwlZnXTBjuqfCPAXm_yzRamENZW8hm8wjanlF0n6_Yuv7QuVTnkgSkqaos3hUNBAeGpfOpTB5Q8ixQU6uLpDVUSPuD7pFJQojyk; 1P_JAR=2021-03-28-17;
SIDCC=AJi4QfErawRoVXROzps68p5wGajcP-Eg9RzPHIWe6pwkVzzp-aeS4LdAecJotkkqXHRZcxqrEwCW;
__Secure-3PSIDCC=AJi4QfGFxz2MlMy7gQbQ0IG5HvD7FU9pSQFDm8AYTMACSuj_6srud-4-AG6G-BzKqZYh_F0c8lXg
pragma: no-cache
sec-ch-ua: "Google Chrome";v="89", "Chromium";v="89", ";Not A Brand";v="99"
sec-ch-ua-mobile: ?0
sec-fetch-dest: document
sec-fetch-mode: navigate

sec-fetch-site: same-origin
sec-fetch-user: ?1
upgrade-insecure-requests: 1
#Tipo de cliente que estou usando
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.90 Safari/537.36
x-client-data:
CJG2yQEIo7bJAQjBtskBCKmdygEIlqzKAQiiw8oBCPjHygEIyovLAQiymssBCNScywEI45zLAQioncsBCOidywEI3+/LARjgmssB
Decoded:
message ClientVariations {
  // Active client experiment variation IDs.
  repeated int32 variation_id = [3300113, 3300131, 3300161, 3313321, 3315222, 3318178, 3318776, 3327434, 3329330, 3329620, 3329635, 3329704, 3329768, 3340255];
  // Active client experiment variation IDs that trigger server-side behavior.
  repeated int32 trigger_variation_id = [3329376];
}

Response

alt-svc: h3-29=":443"; ma=2592000,h3-T051=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
#Está voltando do servidor falando que o controle de cache é público e tem uma idade máxima de x segundos. Como meu cache está desabilitado eu sempre receberei conteúdos novos
cache-control: private, max-age=0
#Tipo de codificação para esse conteúdo: português BR
content-encoding: br
#Tamanho do conteúdo de volta (bytes)
content-length: 38911
#Tipo do conteúdo
content-type: text/html; charset=UTF-8
date: Sun, 28 Mar 2021 17:42:01 GMT
expires: -1
server: gws
#Cookie que ficará no browser para que possa ser usado no próximo pedido
set-cookie: 1P_JAR=2021-03-28-17; expires=Tue, 27-Apr-2021 17:42:01 GMT; path=/; domain=.google.com; Secure; SameSite=none
set-cookie: OTZ=; expires=Mon, 01-Jan-1990 00:00:00 GMT; path=/; domain=www.google.com
set-cookie: OTZ=; expires=Mon, 01-Jan-1990 00:00:00 GMT; path=/; domain=.www.google.com
set-cookie: OTZ=; expires=Mon, 01-Jan-1990 00:00:00 GMT; path=/; domain=google.com
set-cookie: OTZ=; expires=Mon, 01-Jan-1990 00:00:00 GMT; path=/; domain=.google.com
set-cookie:
SIDCC=AJi4QfHwuBDBssBuli-SplWptufJWPPj-_DDmjmQYtLe1ZLKEtP92rgF6Z9xyC75ZLwoifQW0xtt; expires=Mon, 28-Mar-2022 17:42:01 GMT; path=/; domain=.google.com; priority=high
set-cookie:
__Secure-3PSIDCC=AJi4QfHHtPWfkA976CfVwRdmsh9TSe_OSQX8SfAI68_XNOKeywJgnUY290DS-4KQMctD-Y7bkMxV; expires=Mon, 28-Mar-2022 17:42:01 GMT; path=/; domain=.google.com; Secure; HttpOnly; priority=high; SameSite=none
strict-transport-security: max-age=31536000
x-frame-options: SAMEORIGIN
x-xss-protection: 0

Obtendo ajuda

- Acesse o portal https://devdocs.io/http-headers/. Ele busca conteúdo de diversos frameworks/ bibliotecas/ tecnologias;
- Pesquise por http headers;
- Note que ele traz uma listagem enorme de headers. Sua grande maioria já é configurada pelos frameworks;
- Clique em Accept: Note que ele traz a sintaxe, diretivas, etc...
- Clique em outro - Access-Control-Allow-Origin: Traz a sintaxe, diretivas, o que cada elemento da sintaxe significa, respostas padrão, etc...

Status

Status code mais comuns

O objetivo do status code é fazer uma comunicação mais clara entre o servidor e o cliente.
Mais comuns:

	Status Code	Descrição
100	100: Continue	O servidor está dizendo que está tudo OK, podemos continuar
200	200: OK {GET, POST}	O servidor está dizendo que está tudo OK
	201: CREATED {PUT}	O servidor está dizendo que está tudo OK
	204: No Content {DELETE, PUT}	O servidor está dizendo que está tudo OK. O recurso foi deletado
300	301: Moved Permanently	O recurso que estamos acessando foi movido. Geralmente vem com um header e um location para sabermos pra onde ele foi e utilizamos mais para o GET
	308: Permanently Redirect	O recurso que estamos acessando foi movido. Geralmente utilizamos mais no método POST
	302: Found	O recurso foi encontrado, mas foi movido temporariamente. Mai usado para o GET
	307: Temporarily Redirect	O recurso foi encontrado, mas foi movido temporariamente e através do location eu vou para a localização temporária do recurso. Mais usado para os outros métodos com exceção do GET
	400: Bad Request	É um pedido mal efetuado

400	401: Unauthorized	Não estou autorizado a receber o recurso. Eu fiz o pedido, ocorreu tudo certo, porém, faltou eu enviar algum tipo de autorização, exemplo: chave de autorização (header de autorização com chave de autorização)
	403: Forbidden	Não tenho permissão de acesso ao recurso
	404: Not Found	Recurso não encontrado
	405: Method Not Allowed	O método não está permitido para determinado escopo
	429: Too Many Request	Fui bloqueado porque estou fazendo muitos pedidos
500	500: Internal Server Error	Ocorreu um erro na parte do servidor
	503: Service Unavailable	Serviço não disponível nesse momento

Buscando mais informações

- Respostas da parte do servidor: https://devdocs.io/http-status/