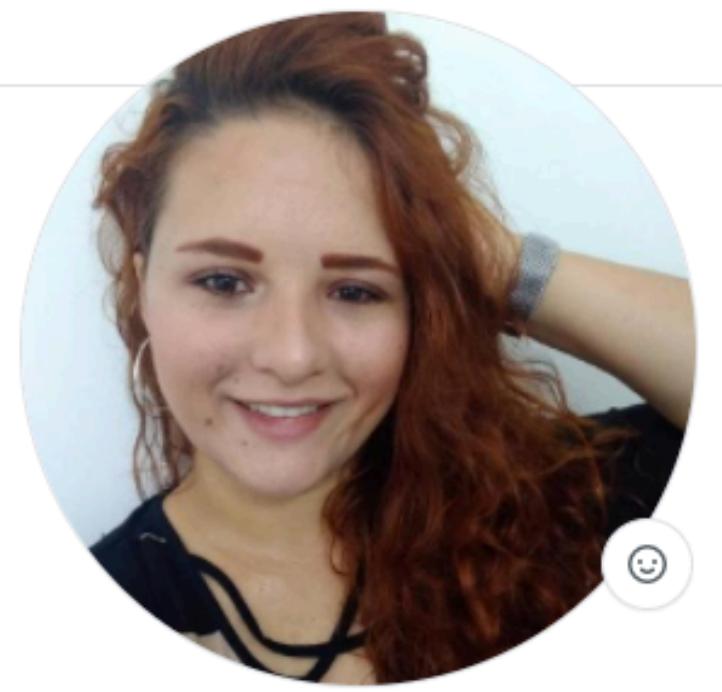


LETICIA SOARES

INTRODUÇÃO AO NODEJS



Overview Repositories 24 Projects Packages

Find a repository...

get-started-with-node-js

Repositório para aula sobre Node no Reprograma

Updated 13 hours ago

create-react-app-typescript-todo-example-2020

Template

Forked from laststance/create-react-app-typescript-todo-example-2020

Create React App TypeScript Todo Example 2020

TypeScript 30 MIT License Updated on Aug 4

Letícia Soares

LeticiaSoares

Frontend Developer

Edit profile

27 followers · 26 following · 10

Gympass

São Paulo-SP

lele_leticia.94@hotmail.com

Highlights

PRO

Organizations



EMAIL

LELE_LETICIA.94@HOTMAIL.COM

GITHUB

[HTTPS://GITHUB.COM/LETICIASOARES](https://github.com/leticiasoares)

WHATSAPP

11963781553

react-project

React app without create react app

JavaScript Apache License 2.0 Updated on Jul 31

get_started_with_nextjs

JavaScript Updated on Jul 31

graphql-with-apollo-client

JavaScript 1 Apache License 2.0 Updated on Jul 20

node-api

JavaScript MIT License Updated on Jul 20

ROTEIRO DA AULA

- Overview HTTP
- Backend e Frontend
- Como o HTTP Funciona
- O que são APIs
- Estrutura do HTTP
- HTTP Status Code
- Introdução ao Node Js
- Criando APIs com Node JS
- Consumindo APIs com Postman

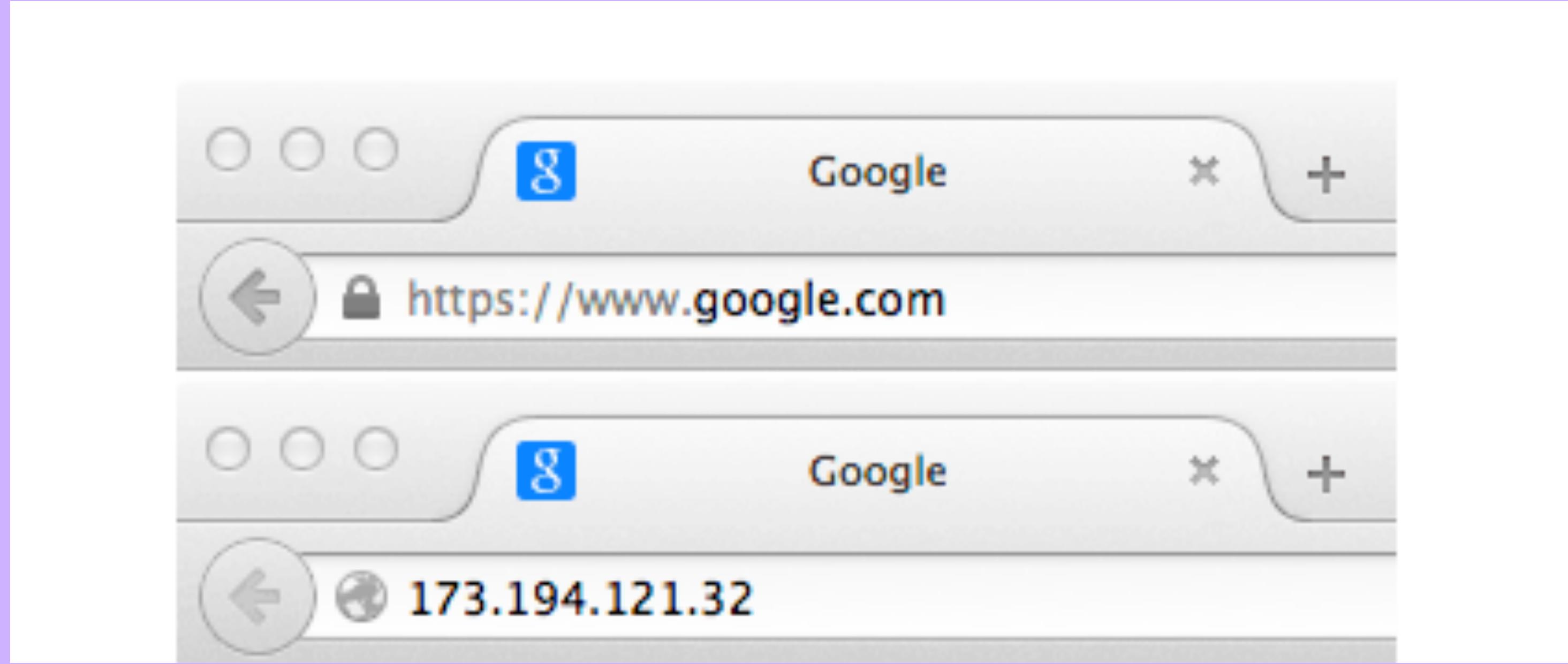
**COMO FUNCIONA A INTERNET QUE
CONHECEMOS?**

A INTERNET É A INFRAESTRUTURA
QUE POSSIBILITA TER UMA
GIGANTESCA REDE DE
COMPUTADORES QUE SE
COMUNICAM.

A WEB É O SERVIÇO CONSTRUÍDO
NA INTERNET...

A WEB QUE CONHECEMOS HOJE
FUNCIONA ATRAVÉS...

URLS



POR EXEMPLO: GOOGLE.COM É UM “APELIDO DO ENDEREÇO
173.194.121.32(ENDEREÇO IP)

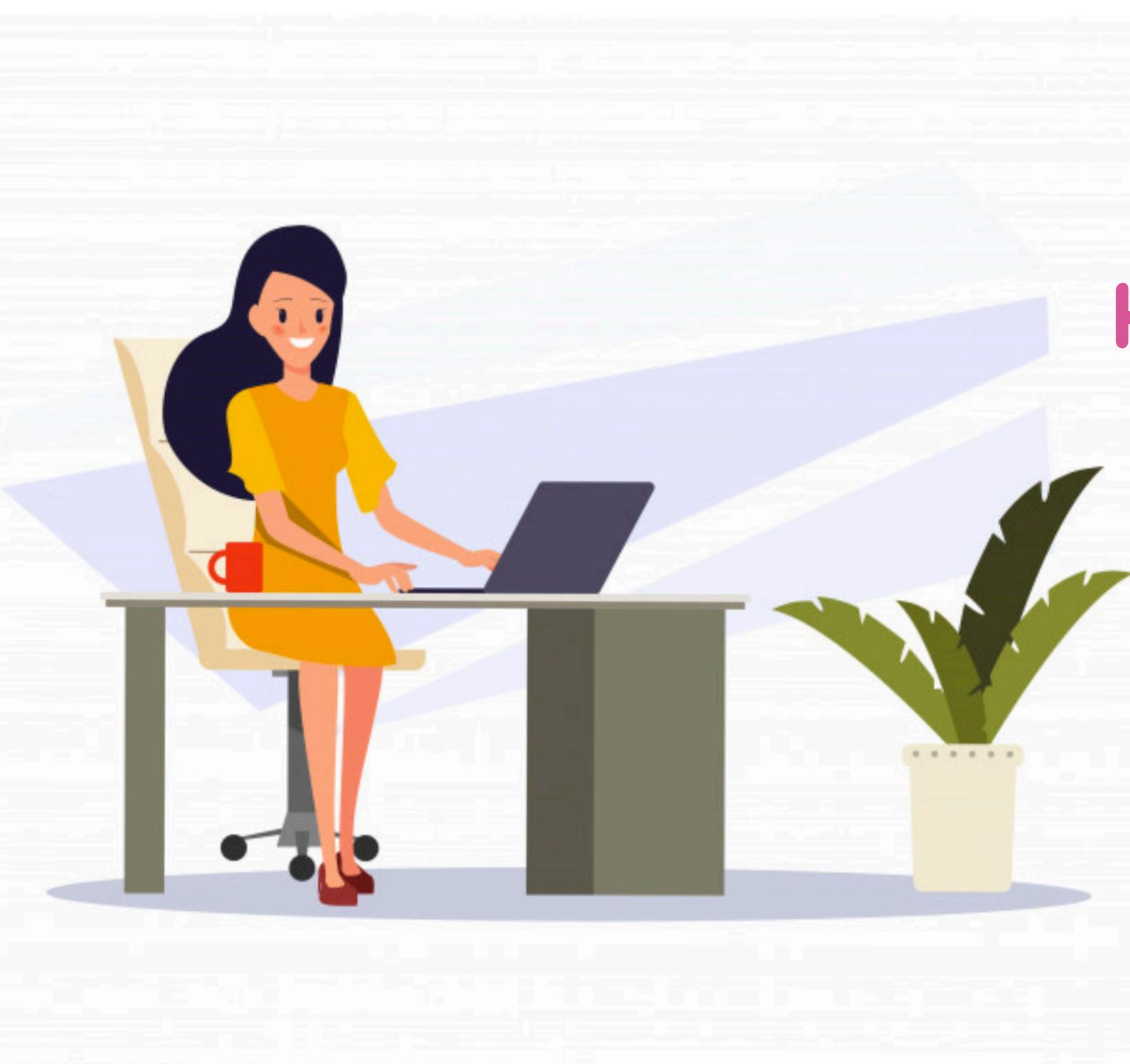
USANDO OS “APELIDOS”,
OU SEJA OS DOMÍNIOS
SE TORNA MAIS SIMPLES
NAVEGAR NA WEB.

QUANDO ACESSAMOS UM SITE

[HTTPS://WWW.GOOGLE.COM/](https://www.google.com/)

ESTAMOS ACESSANDO NADA
MAIS DO QUE UM
COMPUTADOR

A WEB NADA MAIS É DO QUE COMPUTADORES SE COMUNICANDO



VOCÊ DO SEU
COMPUTADOR

ACESSA A O ENDEREÇO
[HTTPS://WWW.GOOGLE.COM/](https://www.google.com/)



QUE É UM ENDEREÇO PARA O
SERVIDOR(COMPUTADOR) DO GOOGLE



WWW - É SIGLA PARA WORLD
WIDE WEB.

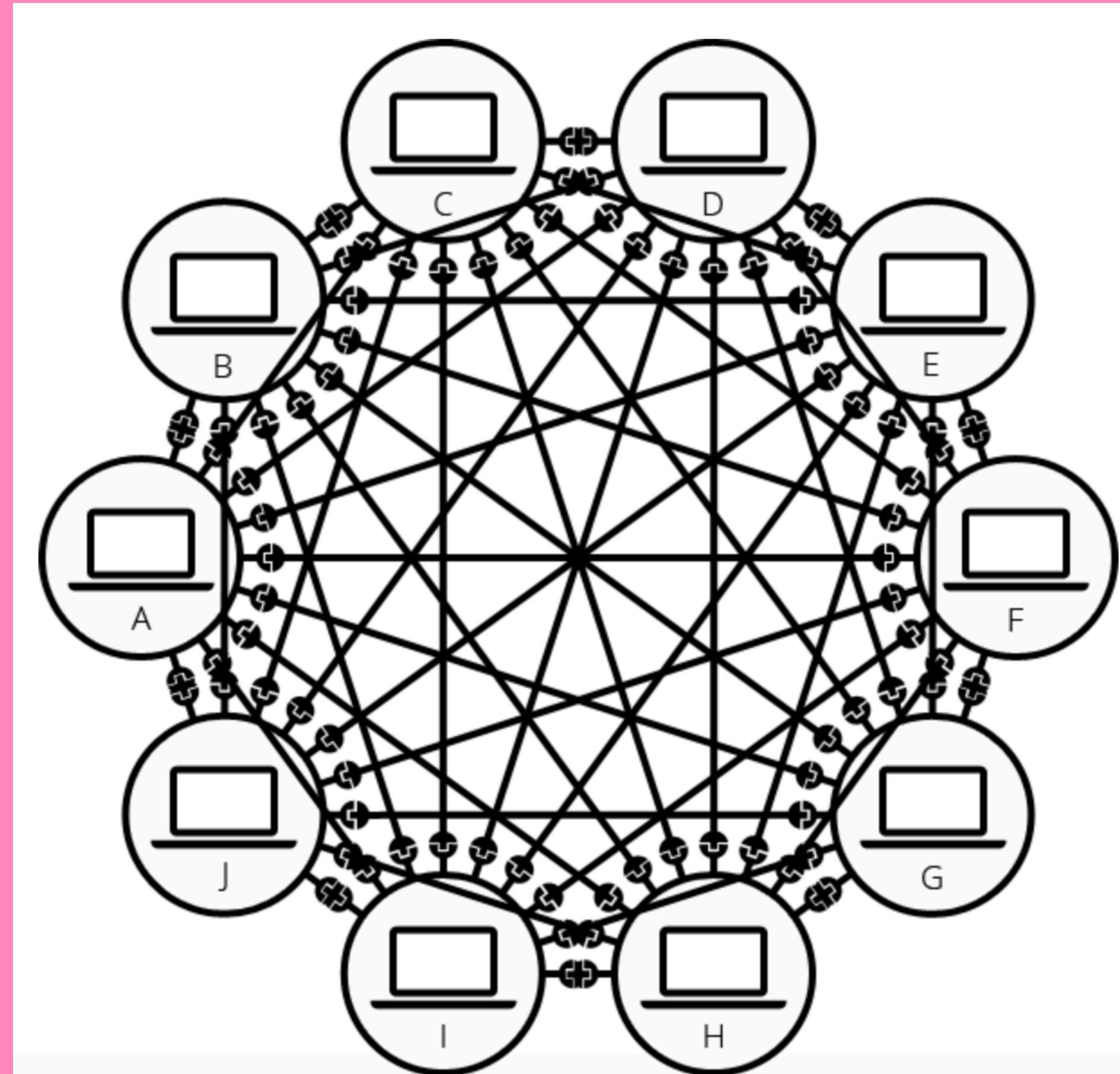
QUE SIGNIFICA O ALCANCE DA
REDE MUNDIAL

OU SEJA, O MUNDO CONECTADO!

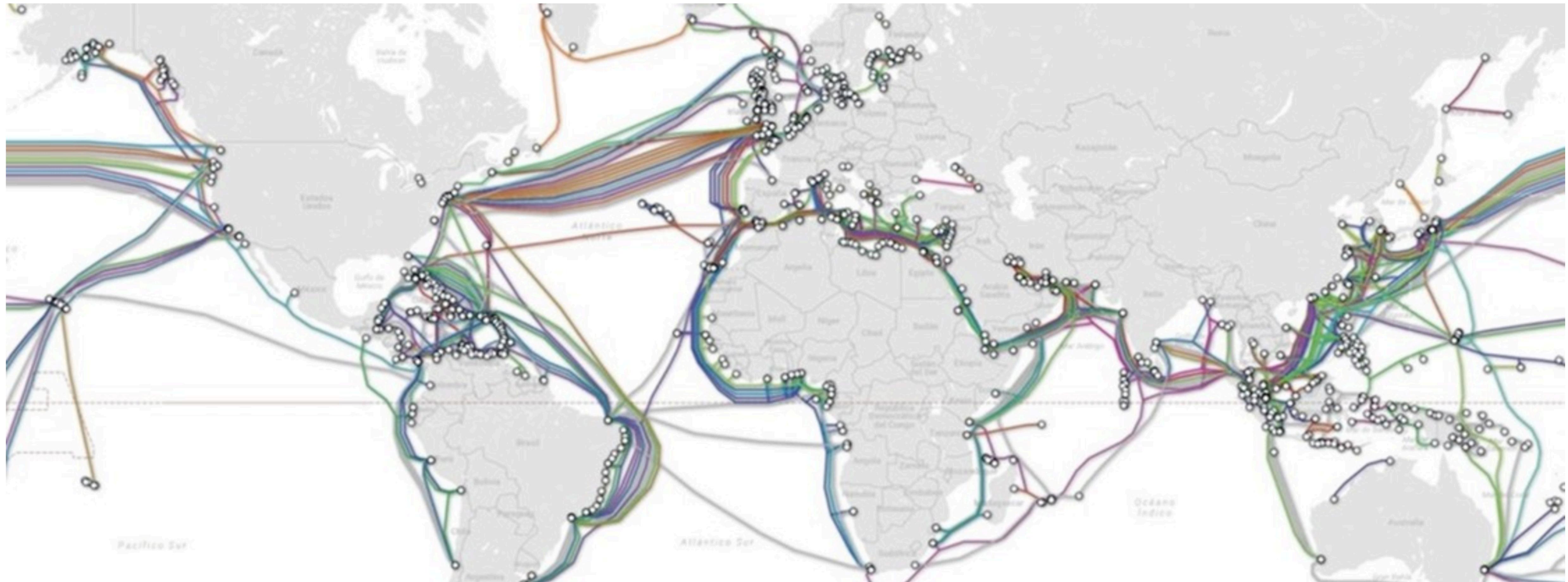


E O HTTPS:// É O
PROTOCOLO(CONJUNTO DE
REGRAS), QUE PERMITE A TROCA
DE INFORMAÇÃO PELA WEB.

A INTERNET É A REDE DE
COMUNICAÇÃO ENTRE
VÁRIOS COMPUTADORES
ESPALHADOS PELO MUNDO



E NÃO É MAGIA



HÁ CABOS SUBMARIÑOS!



PARA CRIAR APLICAÇÕES
WEB PRECISAMOS SABER
SOBRE HTTP

MAS...O QUE É HTTP?

HTTP

HYPER TEXT TRANSFER PROTOCOL
(PROTOCOLO DE TRANSFERÊNCIA DE
HIPERTEXTO)

HTTP É UM PROTOCOLO
DE COMUNICAÇÃO

...OU SEJA, É UM SISTEMA
DE REGRAS QUE DEFINE
COMO O DADO É
TRAFEGADO NA WEB...

**QUANDO ABRIMOS UM SITE
CARREGAMOS UMA PÁGINA HTML
QUE TEM IMAGENS, VIDEOS E ETC...**

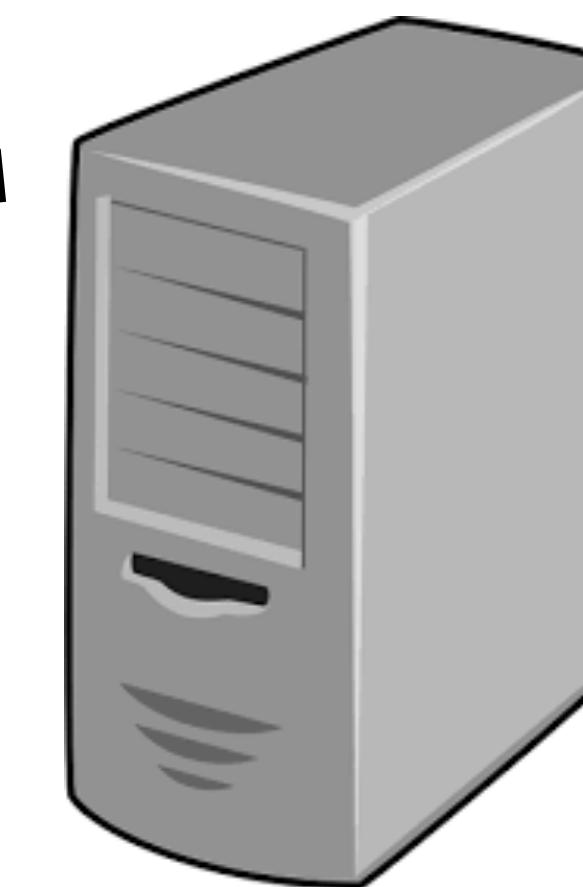
ACESSANDO UM ENDEREÇO WEB

[HTTPS://WWW.GOOGLE.COM/](https://www.google.com/)

CLIENT



RESPONSE - PÁGINA HTML

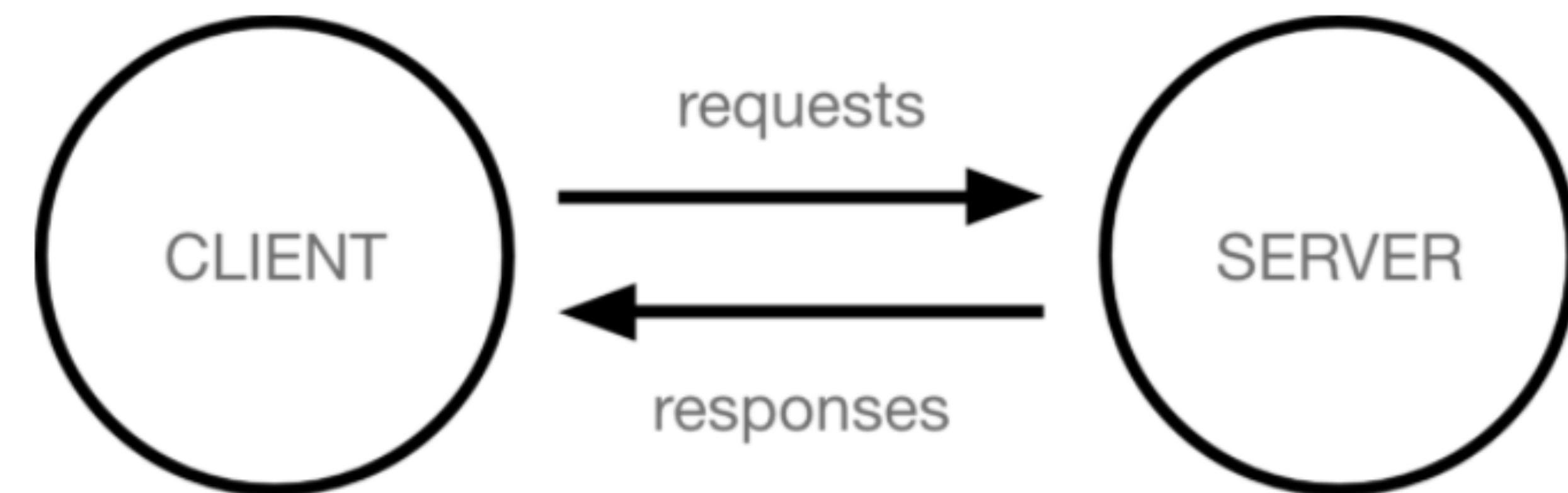


SERVER

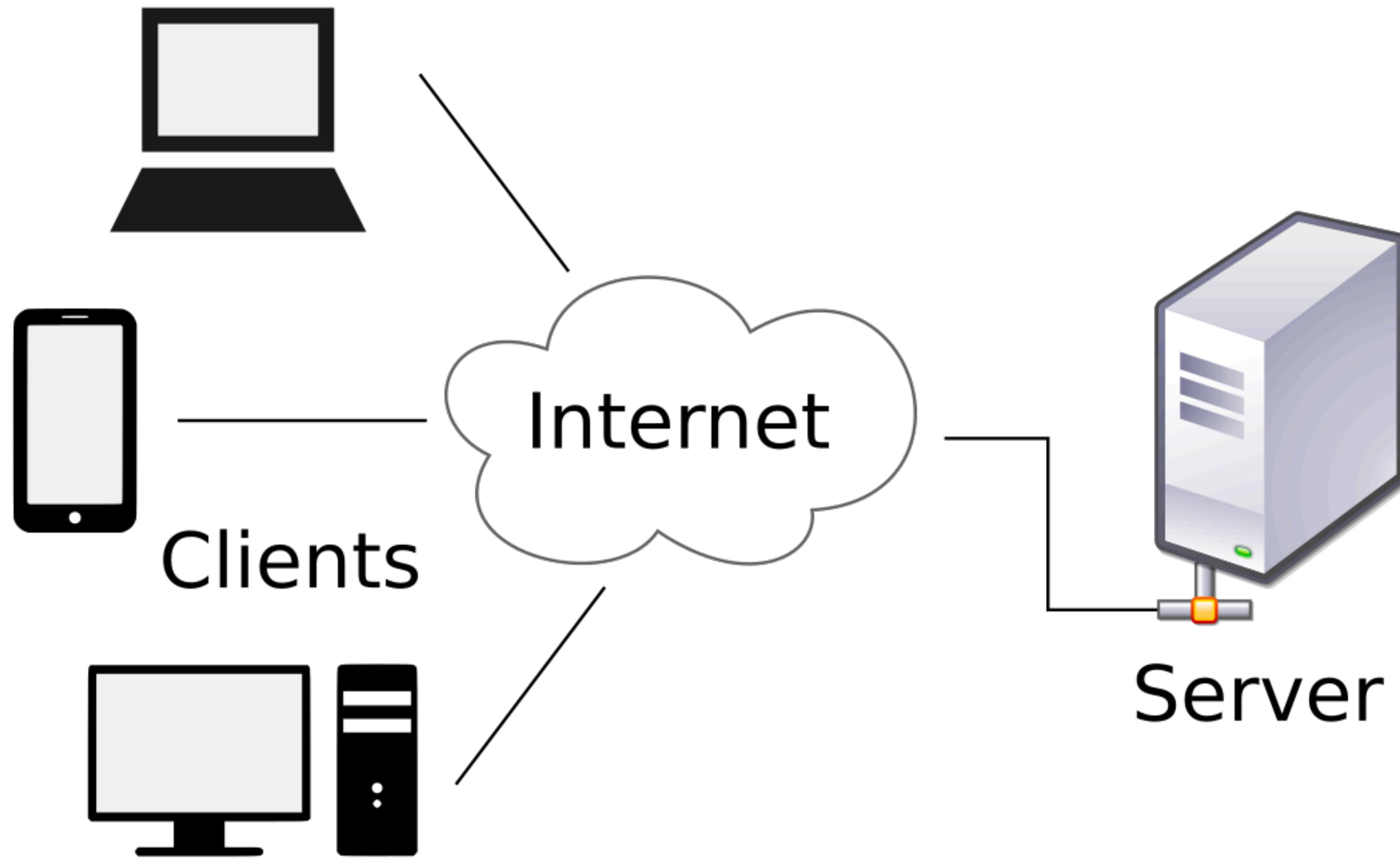
REQUEST - SOLICITA VER UMA PÁGINA

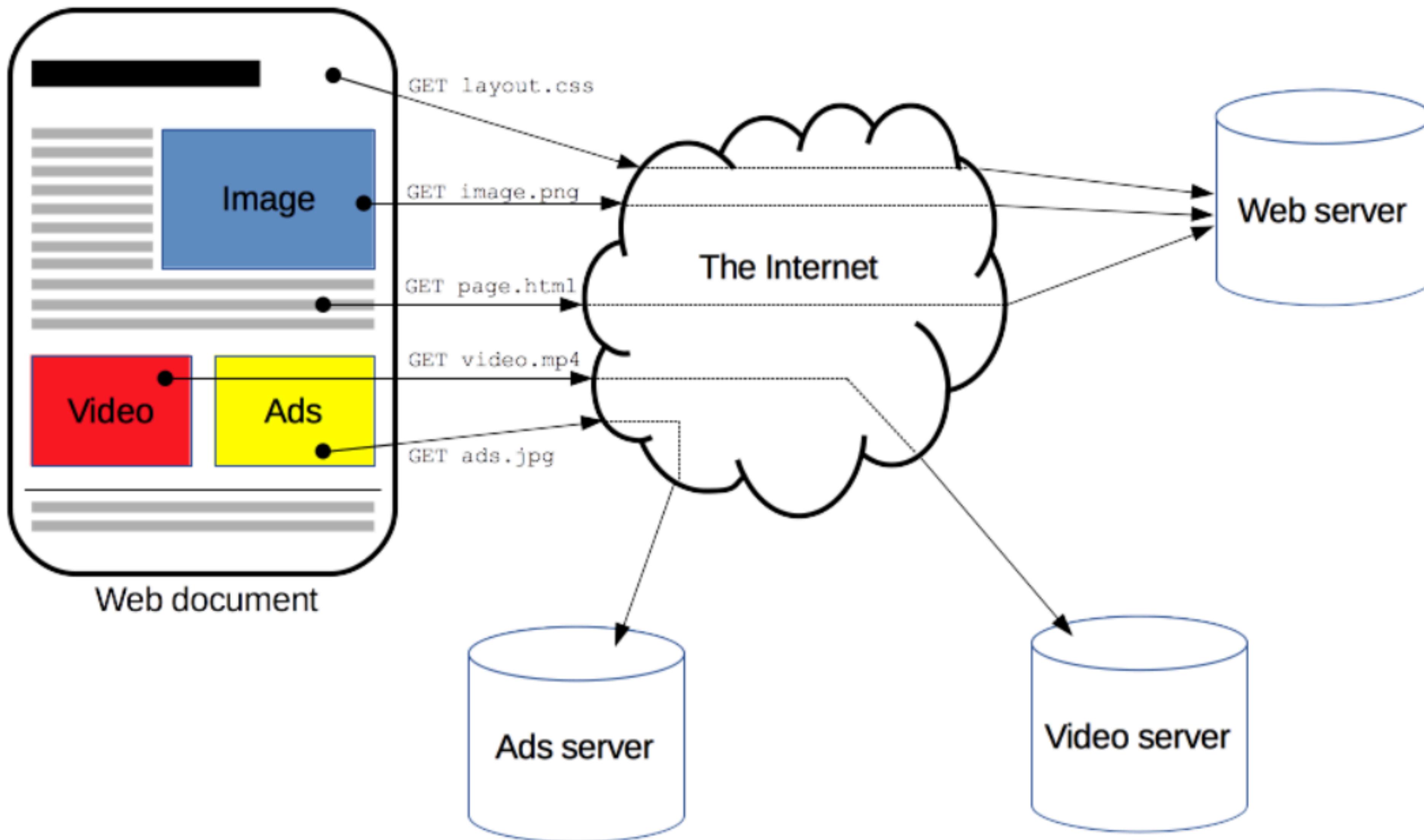


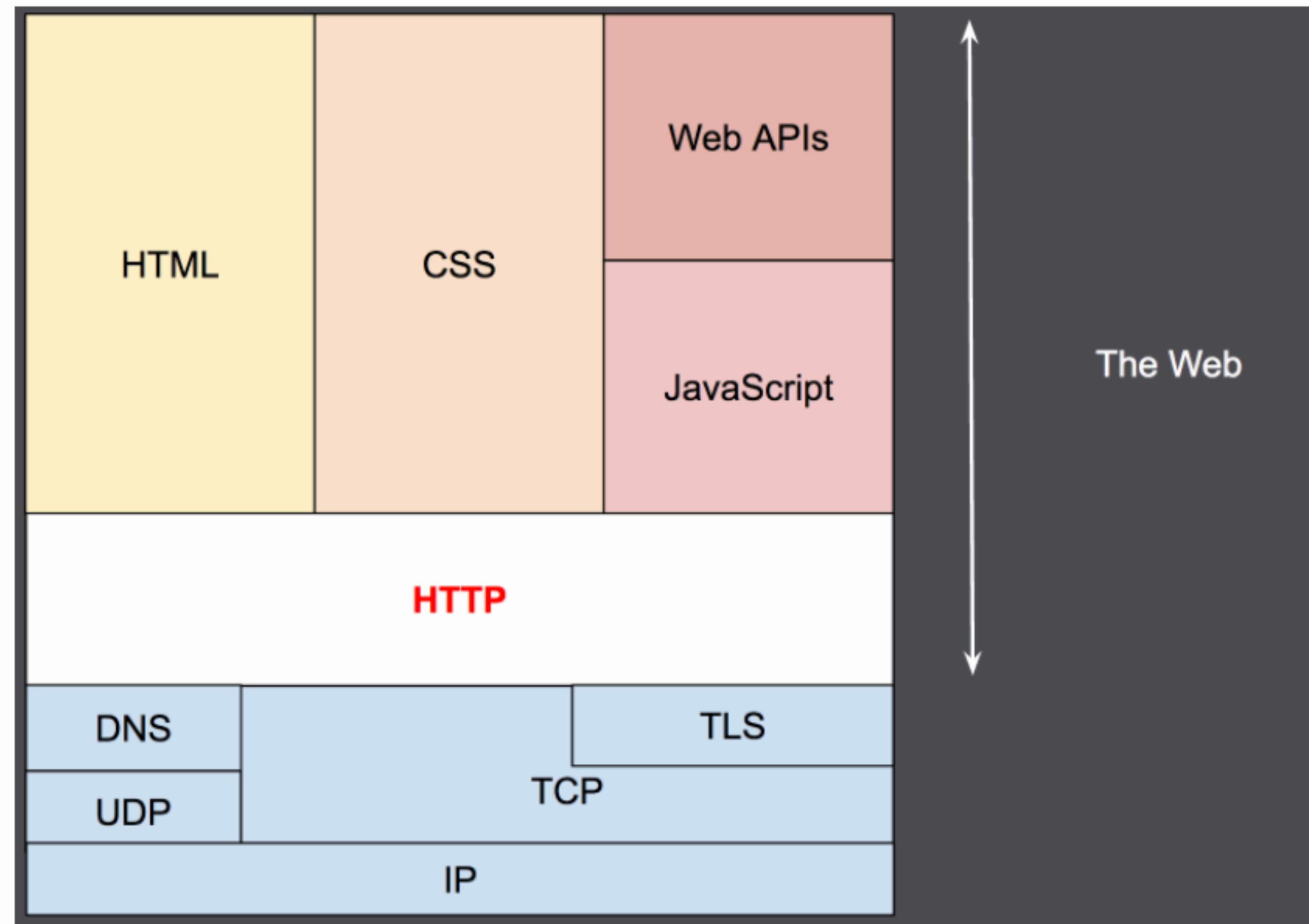
ESSE MODELO SE CHAMA CLIENTE SERVIDOR...



ONDE TEMOS UM CLIENT(BROWSER) QUE
FAZ UMA REQUISIÇÃO(REQUEST) A UM
SERVIDOR QUE DÁ UMA
RESPOSTA(RESPONSE) AO CLIENTE(CLIENT)





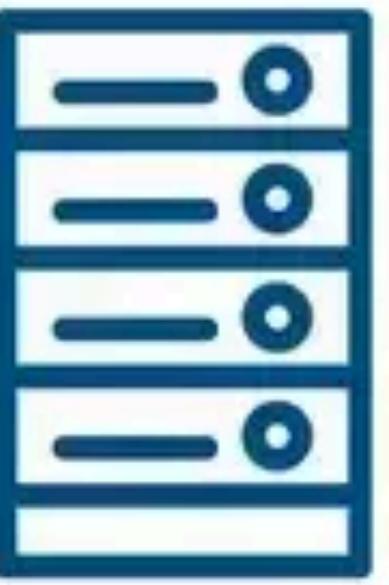


COM ESSE MODELO NOS
ENTENDEMOS A DIFERENÇA
ENTRE BACKEND E FRONTEND

FRONTEND

BACKEND





Front End

- Markup and web languages such as HTML, CSS and Javascript
- Asynchronous requests and Ajax
- Specialized web editing software
- Image editing
- Accessibility
- Cross-browser issues
- Search engine optimisation

Back End

- Programming and scripting such as Python, Ruby and/or Perl
- Server architecture
- Database administration
- Scalability
- Security
- Data transformation
- Backup

AGORA QUE ENTENDEMOS A DIFERENÇA
ENTRE FRONT E BACK...PODEMOS
APRENDER COMO ELES SE COMUNICAM...

O CLIENT(FRONT) SE COMUNICA COM
O SERVER(BACK) ATRAVÉS DE
APIS(PONTES DE ACESSOS AO SERVER)

**APIS SÃO “PORTAS DE ACESSO” QUE
PERMITEM NOSSA COMUNICAÇÃO
COMO SERVIÇOS E APLICAÇÕES.**

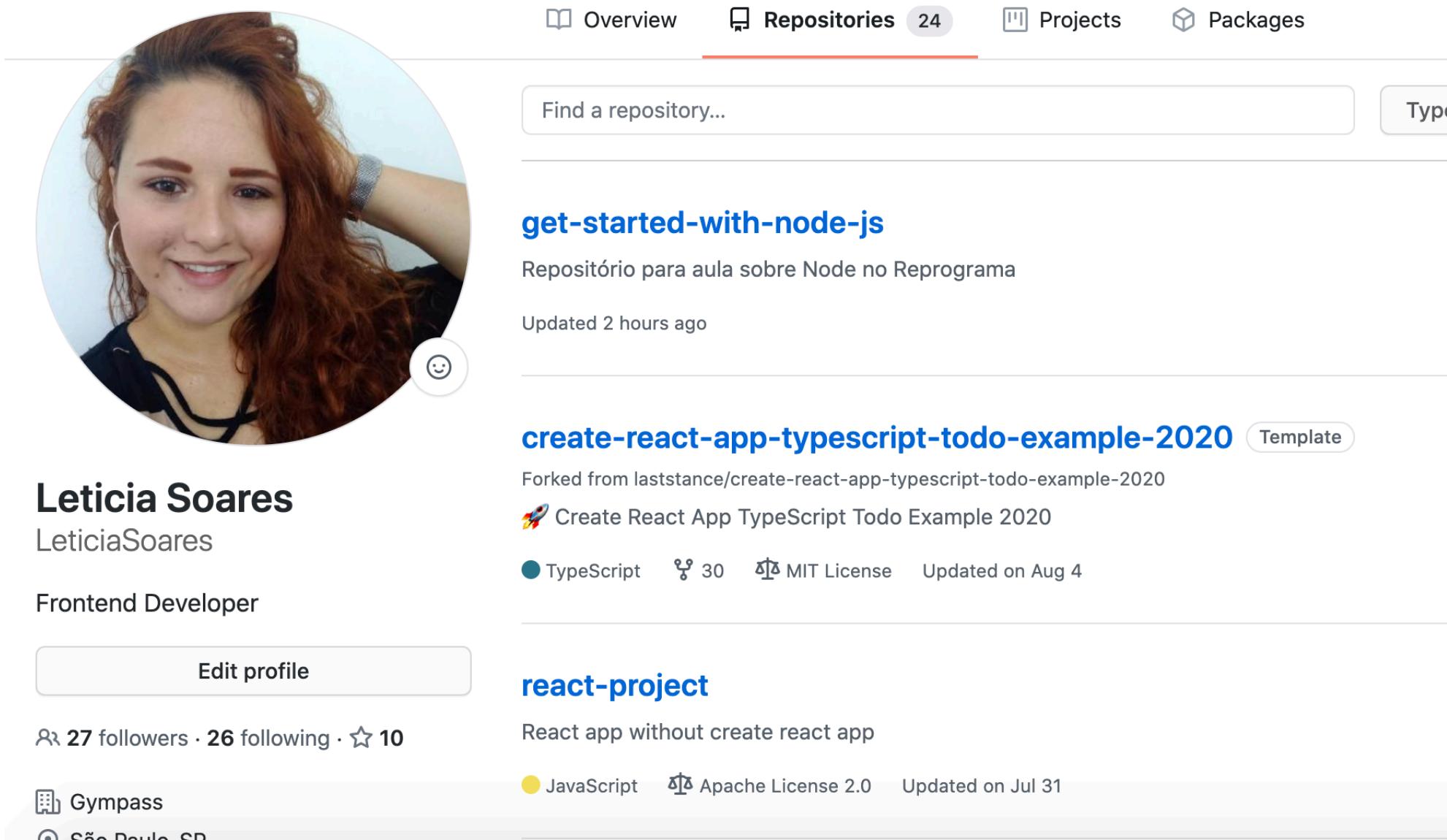
POR EXEMPLO:
O **GITHUB** TEM UMA API QUE É
POSSÍVEL RETORNAR OS REPOSITÓRIOS
DE UM USUÁRIO

<https://api.github.com/users/LeticiaSoares/repos>

```
{  
  "id": 293954013,  
  "node_id": "MDEwOlJlcG9zaXRvcnky0TM5NTQwMTM=",  
  "name": "get-started-with-node-js",  
  "full_name": "LeticiaSoares/get-started-with-node-js",  
  "private": false,  
  "owner": {  
    "login": "LeticiaSoares",  
    "id": 11762938,  
    "node_id": "MDQ6VXNlcjExNzYy0TM4",  
    "avatar_url": "https://avatars1.githubusercontent.com/u/11762938?v=4",  
    "gravatar_id": "",  
    "url": "https://api.github.com/users/LeticiaSoares",  
    "html_url": "https://github.com/LeticiaSoares",  
    "followers_url": "https://api.github.com/users/LeticiaSoares/followers",  
    "following_url": "https://api.github.com/users/LeticiaSoares/following{/other_user}",  
    "gists_url": "https://api.github.com/users/LeticiaSoares/gists{/gist_id}",  
    "starred_url": "https://api.github.com/users/LeticiaSoares/starred{/owner}{/repo}",  
    "subscriptions_url": "https://api.github.com/users/LeticiaSoares/subscriptions",  
    "organizations_url": "https://api.github.com/users/LeticiaSoares/orgs",  
    "repos_url": "https://api.github.com/users/LeticiaSoares/repos",  
    "events_url": "https://api.github.com/users/LeticiaSoares/events{/privacy}",  
    "received_events_url": "https://api.github.com/users/LeticiaSoares/received_events",  
  }  
}
```

OU SEJA, ATRAVÉS DA **API**S EU
FRONTEND CONSIGO ME CONECTAR COM
O SERVIDOR ATRAVÉS DE UMA
REQUISIÇÃO HTTP

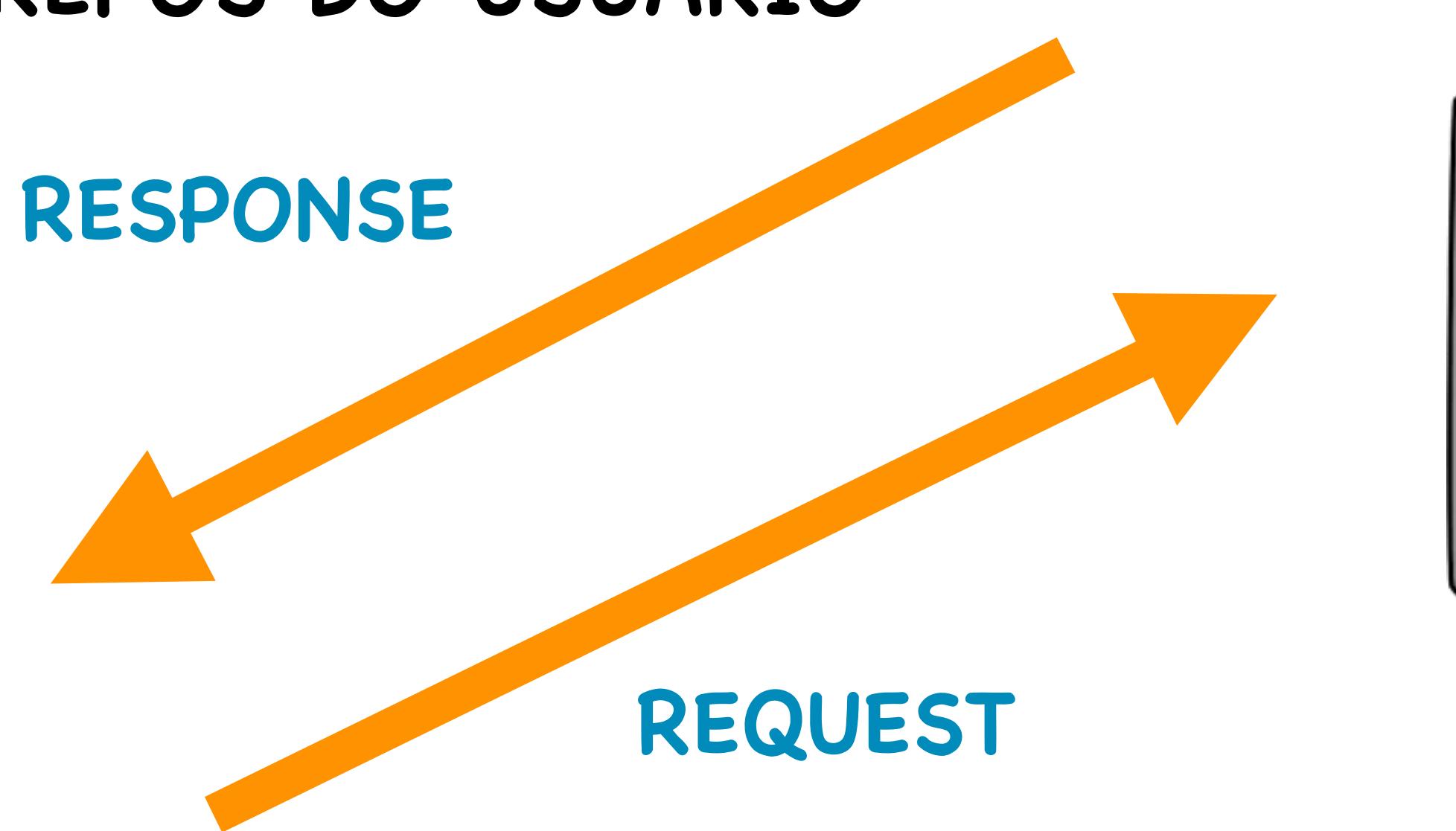
FRONTEND - CLIENT



A screenshot of a GitHub user profile for 'Leticia Soares'. The profile picture is a circular photo of a woman with red hair. The profile name is 'Leticia Soares' and the username is 'LeticiaSoares'. The title is 'Frontend Developer'. Below the profile, there is an 'Edit profile' button. At the bottom, it shows '27 followers · 26 following · 10' stars. The main content area shows the user's repositories: 'get-started-with-node-js' (24 stars), 'create-react-app-typescript-todo-example-2020' (Template, 30 stars), and 'react-project' (Apache License 2.0, 1 star). Each repository has a brief description, the number of stars, and the last update date.

**SERVER RETORNA A LISTA
DE REPOS DO USUÁRIO**

BACKEND - SERVER



**FAZ UMA
REQUISIÇÃO HTTP
PARA UMA API**

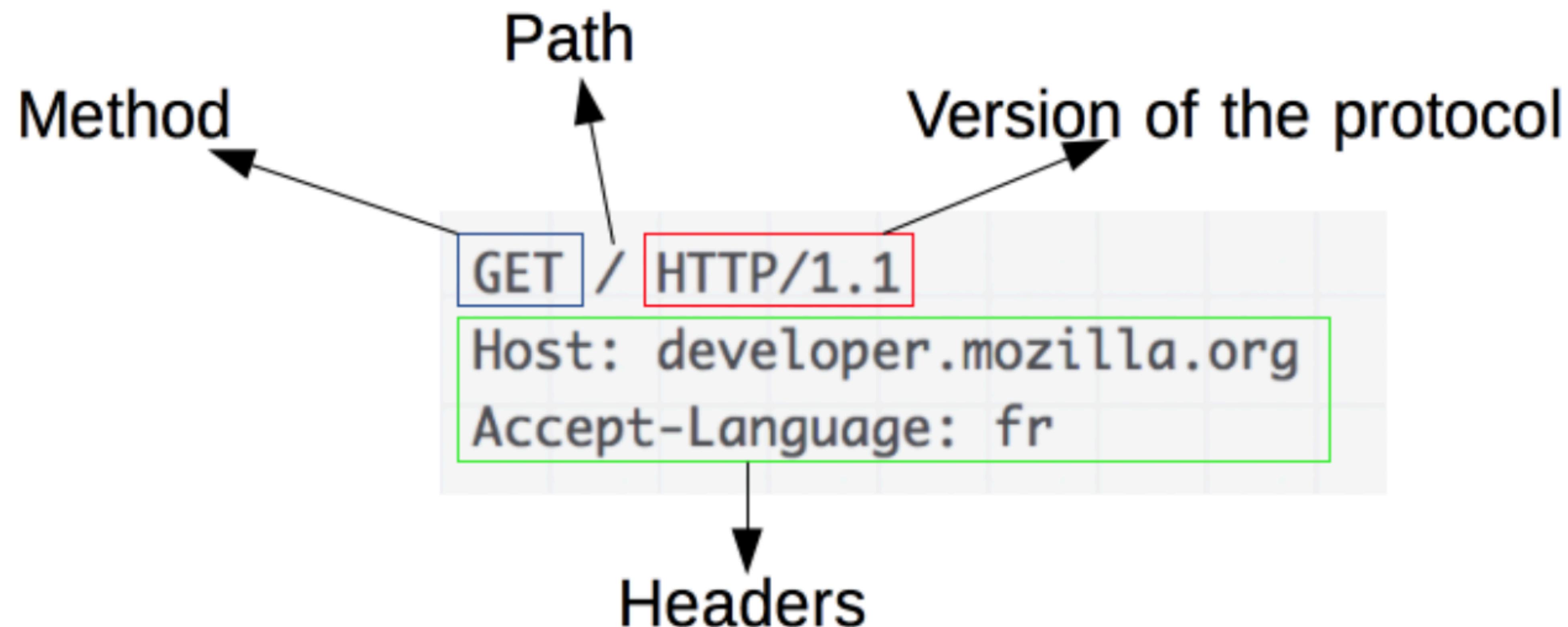
**[https://api.github.com/
users/LeticiaSoares/repos](https://api.github.com/users/LeticiaSoares/repos)**

GITHUB

AGORA VAMOS ENTENDER UM
POUCO SOBRE A ESTRUTURA DO
HTTP

PODEMOS DIZER QUE O HTTP É
COMO SE FOSSE “CARTAS” QUE O
CLIENTE E O SERVIDOR “TROCAM”

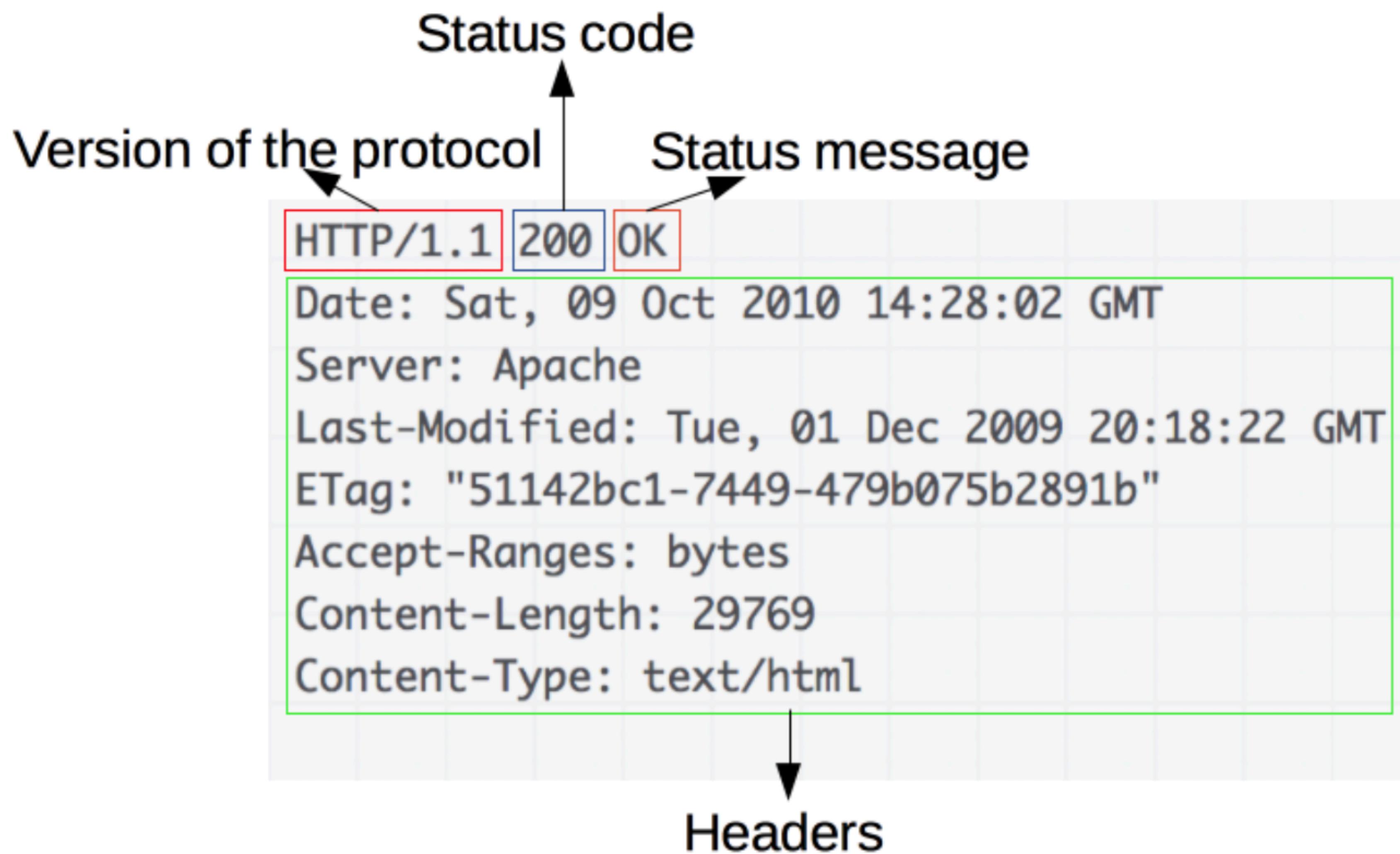
Exemplo de uma requisição HTTP:



As requisições consistem dos seguintes elementos:

- Um **método** HTTP, geralmente é um verbo como `GET`, `POST`, `DELETE`, `PUT`, etc, ou um substantivo como `OPTIONS` ou `HEAD` que define qual operação o cliente quer fazer. Tipicamente, um cliente que pegar um recurso (usando `GET`) ou publicar dados de um formulário HTML (usando `POST`), embora mais operações podem ser necessárias em outros casos.
- O caminho do recurso a ser buscado; a URL do recurso sem os elementos que são de contexto, por exemplo sem o protocolo protocol (`http://`), o domínio **domain** (aqui como `developer.mozilla.org`), ou a porta **port** TCP (aqui indicada pelo `80` que é ocultado por ser o número da porta padrão)
- A versão do protocolo HTTP.
- **Cabeçalhos** opcionais que contém informações adicionais para os servidores.
- Ou um corpo de dados, para alguns métodos como `POST`, similares aos corpos das respostas, que contém o recurso requisitado.

Exemplo de resposta HTTP:



Respostas consistem dos seguintes elementos:

- A versão do protocolo HTTP que elas seguem.
- Um **código de status**, indicando se a requisição foi bem sucedida, ou não, e por quê.
- Uma mensagem de status, uma pequena descrição informal sobre o código de status.
- **Cabeçalhos** HTTP, como aqueles das requisições.
- Opcionalmente, um corpo com dados do recurso requisitado.

Elements Console Sources Network Performance Memory Application Security Lighthouse AdBlock Redux

Preserve log Disable cache Online

Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other Has blocked cookies Blocked Requests

10 ms 20 ms 30 ms 40 ms 50 ms 60 ms 70 ms 80 ms 90 ms 100 ms 110 ms

Name	Headers	Preview	Response	Initiator	Timing
index.html					
index.js					
repos					

General

Request URL: https://api.github.com/users/LeticiaSoares/repos

Request Method: GET

Status Code: 200 OK (from disk cache)

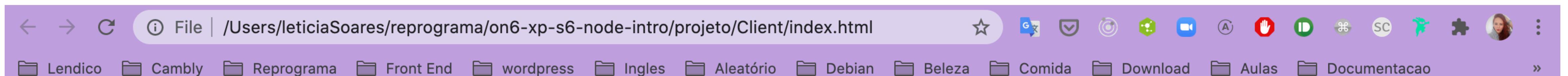
Remote Address: 140.82.112.6:443

Referrer Policy: no-referrer-when-downgrade

Response Headers view source

Accept-Ranges: bytes
access-control-allow-origin: *
access-control-expose-headers: ETag, Link, Location, Retry-After, X-GitHub-OTP, X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Used, X-RateLimit-Reset, X-OAuth-Scopes, X-Accepted-OAuth-Scopes, X-Poll-Interval, X-GitHub-Media-Type, Deprecation, Sunset
cache-control: public, max-age=60, s-maxage=60
content-encoding: gzip
content-security-policy: default-src 'none'
content-type: application/json; charset=utf-8
date: Fri, 11 Sep 2020 00:09:09 GMT
etag: W/"0bcef7efcf02b86a8d5a01249fdaa45c"
referrer-policy: origin-when-cross-origin, strict-origin-when-cross-origin
server: GitHub.com
status: 200 OK
vary: Accept, Accept-Encoding, Accept, X-Requested-With
x-content-type-options: nosniff
x-frame-options: deny
x-github-media-type: github.v3; format=json
X-GitHub-Request-Id: FB6F:0917:1BAD04:2E2345:5F5AC025
X-Ratelimit-Limit: 60
X-Ratelimit-Remaining: 51

3 requests | 711 B transferred



HTTP METHOD O QUE SÃO?

SÃO VERBOS QUE ENVIAMOS NO NOSSO
REQUEST QUE INDICAM QUAL AÇÃO VAI
SER EXECUTADA NO SERVIDOR

POR EXEMPLO:
NO CASO DA API DO GITHUB QUE
VIMOS NOS ENVIAMOS A REQUISIÇÃO
COM O METHOD GET

Elements Console Sources Network **Performance** Memory Application Security Lighthouse AdBlock Redux

Preserve log Disable cache Online

Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other Has blocked cookies Blocked Requests

10 ms 20 ms 30 ms 40 ms 50 ms 60 ms 70 ms 80 ms 90 ms 100 ms 110 ms

Name	Headers	Preview	Response	Initiator	Timing
index.html					
index.js					
repos					

General

Request URL: https://api.github.com/users/LeticiaSoares/repos

Request Method: GET

Status Code: 200 OK (from disk cache)

Remote Address: 140.82.112.6:443

Referrer Policy: no-referrer-when-downgrade

Response Headers view source

Accept-Ranges: bytes
access-control-allow-origin: *
access-control-expose-headers: ETag, Link, Location, Retry-After, X-GitHub-OTP, X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Used, X-RateLimit-Reset, X-OAuth-Scopes, X-Accepted-OAuth-Scopes, X-Poll-Interval, X-GitHub-Media-Type, Deprecation, Sunset
cache-control: public, max-age=60, s-maxage=60
content-encoding: gzip
content-security-policy: default-src 'none'
content-type: application/json; charset=utf-8
date: Fri, 11 Sep 2020 00:09:09 GMT
etag: W/"0bcef7efcf02b86a8d5a01249fdaa45c"
referrer-policy: origin-when-cross-origin, strict-origin-when-cross-origin
server: GitHub.com
status: 200 OK
vary: Accept, Accept-Encoding, Accept, X-Requested-With
x-content-type-options: nosniff
x-frame-options: deny
x-github-media-type: github.v3; format=json
X-GitHub-Request-Id: FB6F:0917:1BAD04:2E2345:5F5AC025
X-Ratelimit-Limit: 60
X-Ratelimit-Remaining: 51

3 requests | 711 B transferred

O METHOD GET SIGNIFICA QUE
QUEREMOS RETORNAR DADOS
DO SERVIDOR

NO CASO DA API DO GITHUB
RETORNAMOS A LISTA DE
REPOS DO USUÁRIO

NOS TEMOS ALGUNS METHODS
QUE UTILIZAMOS PARA CADA
SITUAÇÃO...

GET - QUANDO VAMOS RETORNAR DADOS.

POR EXEMPLO: RETORNAR A LISTA DE REPOS DO GITHUB

POST - QUANDO VAMOS ENVIAR UM DADO PARA SER CRIADO.

POR EXEMPLO: CADASTRAR UM USUÁRIO

PUT - QUANDO QUEREMOS ATUALIZAR UM DADO DE UM REGISTRO QUE JÁ EXISTE

POR EXEMPLO: ATUALIZAR DADOS DO PROFILE DE UM USUÁRIO

DELETE - QUANDO QUEREMOS DELETAR UM DADO

POR EXEMPLO : EXCLUIR UM REPOSITÓRIO DO GITHUB

NOS CASO DO RESPONSE, OU
SEJA O RETORNO DO SERVER.

NÓS TEMOS OS **STATUS CODES**

QUE VAI INDICAR SUCESSO
OU FALHA NA REQUISIÇÃO

POR EXEMPLO...CASO EU ENVIE UM
usuário que não existe para a API
do Github ele vai retornar um
status code de erro.

Repositórios

The screenshot shows the Network tab in the Chrome DevTools developer console. The main area displays a timeline of network requests, with the first few items visible:

Name	Time
index.html	5 ms
index.js	10 ms
repos	15 ms
index.html	20 ms
index.js	25 ms
repos	30 ms
index.html	35 ms
index.js	40 ms
repos	45 ms
index.html	50 ms
index.js	55 ms
repos	60 ms
index.html	65 ms

Below the timeline, a specific request for "index.html" is expanded to show its details:

General

- Request URL: <https://api.github.com/users/testeerrro/repos>
- Request Method: GET
- Status Code: 404 Not Found
- Remote Address: 140.82.112.6:443
- Referrer Policy: no-referrer-when-downgrade

Response Headers (view source)

- access-control-allow-origin: *
- access-control-expose-headers: ETag, Link, Location, Retry-After, X-GitHub-OTP, X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Used, X-Ratelimit-Reset
- content-encoding: gzip
- Content-Length: 127
- content-security-policy: default-src 'none'
- content-type: application/json; charset=utf-8
- date: Fri, 11 Sep 2020 01:41:45 GMT
- referrer-policy: origin-when-cross-origin, strict-origin-when-cross-origin
- server: GitHub.com
- status: 404 Not Found
- strict-transport-security: max-age=31536000; includeSubdomains; preload
- vary: Accept-Encoding, Accept, X-Requested-With
- x-content-type-options: nosniff
- x-frame-options: deny
- x-github-media-type: github.v3; format=json

At the bottom of the Network tab, it shows "45 requests | 20.7 kB transferred".

O ERRO 404 INDICA QUE O SERVER
NÃO CONSEGUIU ENCONTRAR O
DADO QUE VC SOLICITOU

NO TEMOS OUTROS STATUS
CODES PARA CADA TIPO DE
SITUAÇÃO

200- QUANDO UM GET, OU SEJA UM RETORNO DE DADOS FOI BEM SUCEDIDO.

POR EXEMPLO O RETORNO DE REPOS DA API DO GITHUB

201 - QUANDO O DADO É CRIADO COM SUCESSO.

POR EXEMPLO UM CADASTRO USANDO POST

404 - QUANDO NÃO FOR POSSÍVEL ENCONTRAR O DADO.

POR EXEMPLO TENTAR LISTAR OS REPOS DE UM USUÁRIO QUE NÃO EXISTE

400 -QUANDO O SERVIDOR NÃO PODERÁ POR ALGUM ERRO PROCESSAR UM
REQUISIÇÃO. POR EXEMPLO ENVIAR UM REQUEST COM ALGUM DADO
FALTANDO.

500 - QUANDO OCORRE ALGUM ERRO INESPERADO NO SERVIDOR E O MESMO NÃO PODE PROCESSAR A REQUISIÇÃO.POR EXEMPLO: ERRO DE SINTAXE DE CÓDIGO NO SERVIDOR

VERBOS	significado
GET	Obter os dados de um recurso.
POST	Criar um novo recurso.
PUT	Substituir os dados de um determinado recurso.
PATCH	Atualizar parcialmente um determinado recurso.
DELETE	Excluir um determinado recurso.

O NODEJS É AMBIENTE DE EXECUÇÃO
MULTI-PLATAFORMA EM JAVASCRIPT
QUE PERMITE CRIAR APLICAÇÕES
SERVER-SIDE(BACKEND)

OU SEJA....É O **JAVASCRIPT** QUE
CONHECEMOS NO FRONTEND
SENDO USADO NO BACKEND

```
const http = require('http');
```

```
const hostname = '127.0.0.1';
const port = 3000;
```

```
const server = http.createServer((request, response) => {
  response.statusCode = 200;
  response.setHeader('Content-Type', 'text/plain');
  response.end('Hello, World!\n');
});
```

```
server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

PARA INICIAR NOSSO PROJETO EM NODEJS NOS
TEMOS DUAS OPÇÕES DE GERENCIADORES DE
DEPENDÊNCIAS, OU SEJA ELES AJUDAM A GENTE
INSTALAR LIBS QUE PRECISAMOS NO PROJETO



yarn



DEPOIS IREMOS TAMBÉM UTILIZAR O
EXPRESS FRAMEWORK QUE NOS
INSTALAMOS ATRAVÉS DOS
GERENCIADORES YARN OU NPM

O EXPRESS FACILITA NA
CRIAÇÃO DE APIs COM NODEJS

```
server > src > js index.js > ...
```

```
1  var express = require('express');
2
3
4  const app = express();
5
6
7  app.get("/ping", (request, response) => {
8    |   res.send("pong");
9  });
10
11
12 const PORT = 3001;
13
14 app.listen(PORT, () => console.log("Listening on port " + PORT));
15 |
```