

The illustration shows a web browser window with a blue header bar containing navigation icons (back, forward, refresh) and a search bar. The main content area features a 'Contact Us' form with a grey header, a subtitle, and three input fields: 'Full Name', 'E-mail', and 'Message'. A 'SUBMIT' button is located at the bottom of the form.

Contact Us
Please fill this form in a decent manner

Full Name *

E-mail *

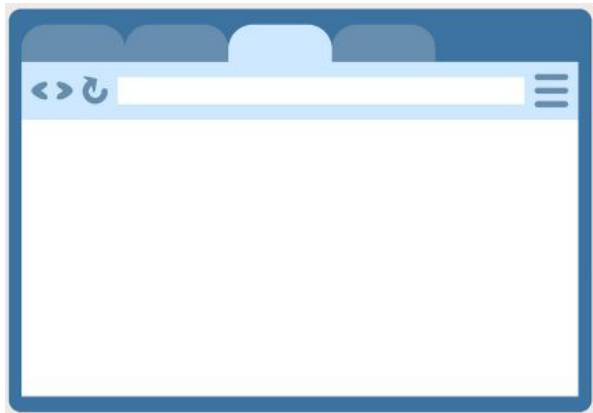
example@example.com

Message *

SUBMIT

Web Request Response

CLIENT



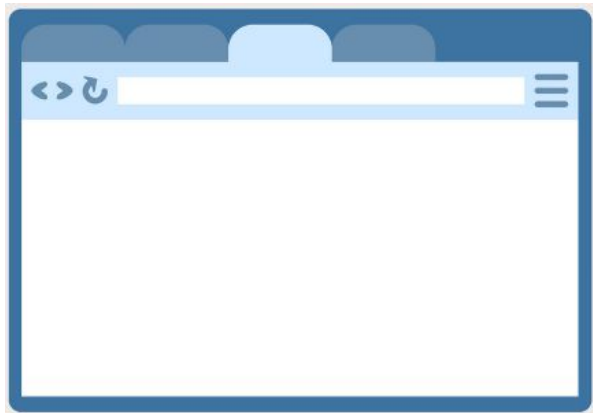
/index.html



SERVER



CLIENT



/index.html



REQUEST

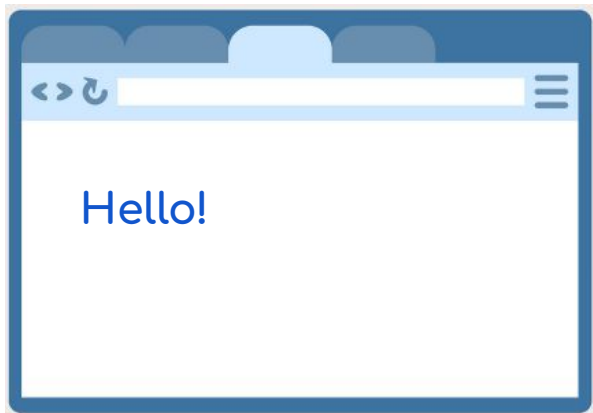
SERVER



```
<html>
  <h1>Hello!</h1>
</html>
```



CLIENT



SERVER

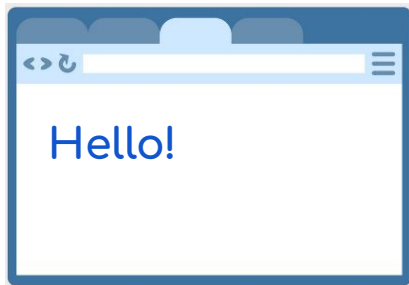


RESPONSE

```
b' <html><h1>Hello!</h1></html>'
```



CLIENT



/index.html

REQUEST

RESPONSE

SERVER



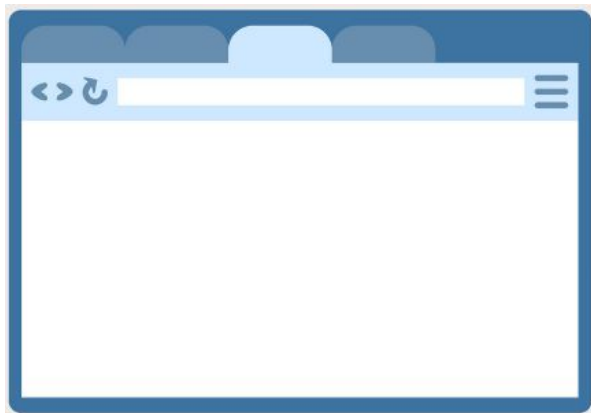
script

App
mobile

Podemos ter mais “clientes” além dos navegadores...

Mais detalhes
da requisição web

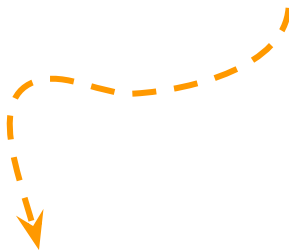
CLIENT



/index.html



REQUEST



SERVER

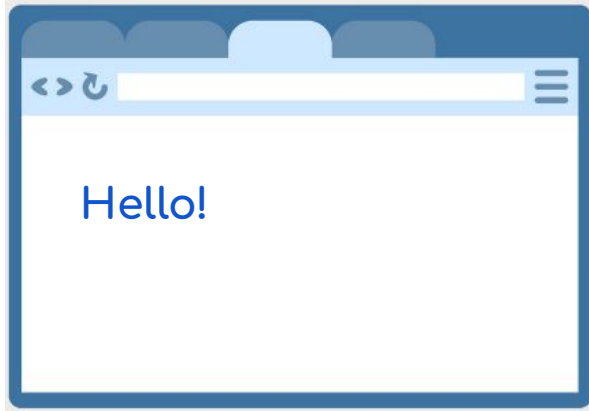


```
GET https://mydomain.com/index.html
```

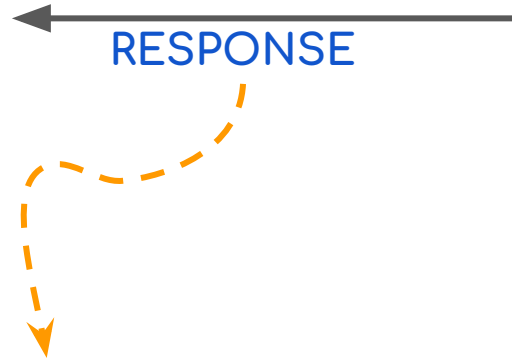
```
HTTP/[versão]
```

```
User-Agent: Chrome/80 (Ubuntu 20.04)
```

CLIENT



SERVER



200 OK

Date: Tue, 15 Nov 2012 08:12:31 GMT

Server: CERN/3.0 libwww/2.17

Content-Type: text/html

<HTML>

...

</HTML>



HTTP:

Um protocolo que veio para padronizar a comunicação entre cliente e servidor.

Importante pesquisar e entender um pouco mais deste protocolo de comunicação...

O que acontece quando vem na resposta um HTML deste?

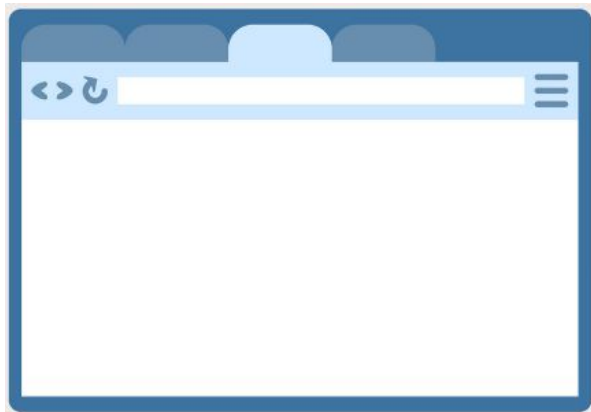
```
<html>
  ...
  <link rel="stylesheet" href="css/styles.css" />
  ...
  <body>
    
    ...
    <script src="js/main.js"></script>
  </body>
</html>
```

O que acontece quando vem na resposta um HTML deste?

```
<html>
  ...
  <link rel="stylesheet" href="css/styles.css" />
  ...
  <body>
    
    ...
    <script src="js/main.js"></script>
  </body>
</html>
```

- O navegador recebe a primeira resposta (exemplo: index.html)
- Ao interpretar a página, faz uma nova requisição (request) para cada recurso que a página inicial faz referência!
- Uma página simples pode encadear diversas novas requisições (+HTTP GET) para o servidor!

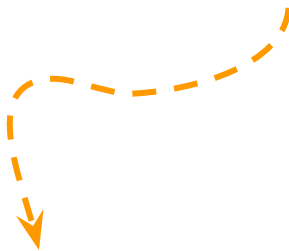
CLIENT



/index.html



REQUEST

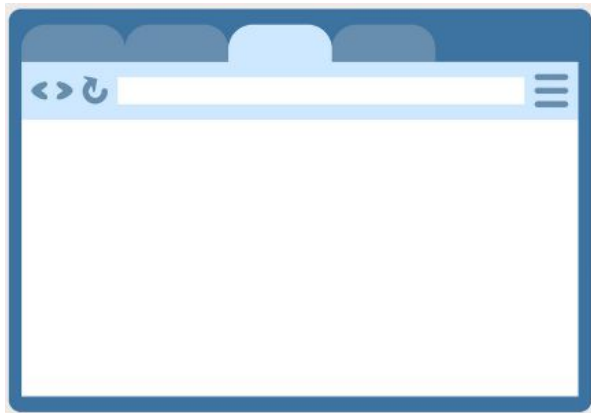


SERVER



```
GET https://mydomain.com/css/styles.css
HTTP/[versão]
User-Agent: Chrome/80 (Ubuntu 20.04)
```

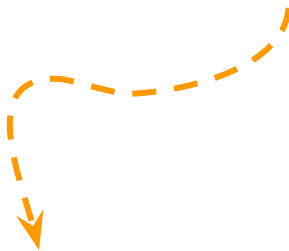
CLIENT



/index.html



REQUEST



SERVER

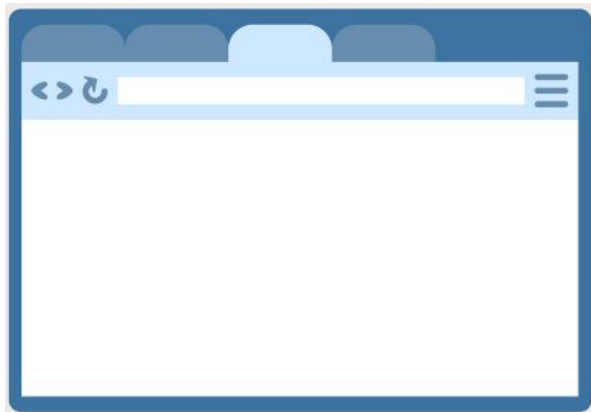


```
GET https://mydomain.com/imgs/hero.jpg
```

```
HTTP/[versão]
```

```
User-Agent: Chrome/80 (Ubuntu 20.04)
```

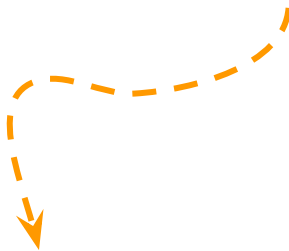
CLIENT



/index.html



REQUEST



SERVER



```
GET https://mydomain.com/js/main.js
```

```
HTTP/[versão]
```

```
User-Agent: Chrome/80 (Ubuntu 20.04)
```

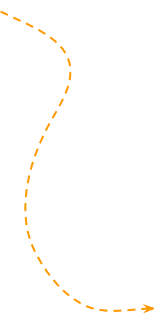
SERVIDOR que responde às requisições:

Como ele sabe responder? Achar o código correspondente da requisição recebida?

CLIENT

SERVER

REQUEST



Ubuntu 20.04 LTS

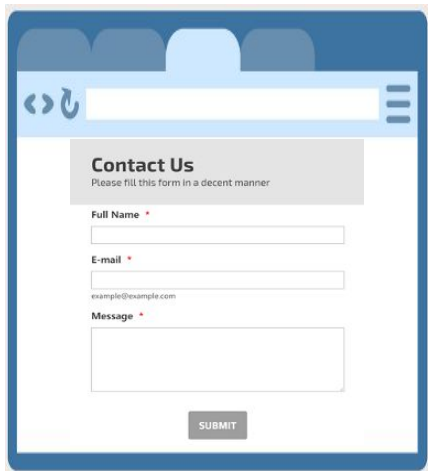
?



Front-end
(estático)

```
minha-app/  
css/  
js/  
imgs/  
index.html
```

RESPONSE



CLIENT

SERVER

REQUEST

A screenshot of a web browser window. The browser has a blue header with a back button and a search bar. The main content area displays a 'Contact Us' form with the instruction 'Please fill this form in a decent manner'. The form includes three input fields: 'Full Name', 'E-mail', and 'Message'. The 'E-mail' field has a placeholder text 'example@example.com'. A 'SUBMIT' button is located at the bottom of the form.

Ubuntu 20.04 LTS



HTTP SERVER



Front-end
(estático)

```
minha-app/  
  css/  
  js/  
  imgs/  
  index.html
```

RESPONSE



O SERVIDOR precisa de um componente que saiba entender o protocolo HTTP.

Um **HTTP SERVER**, desta forma, ele vai ter configurações para saber o que fazer com cada requisição, por

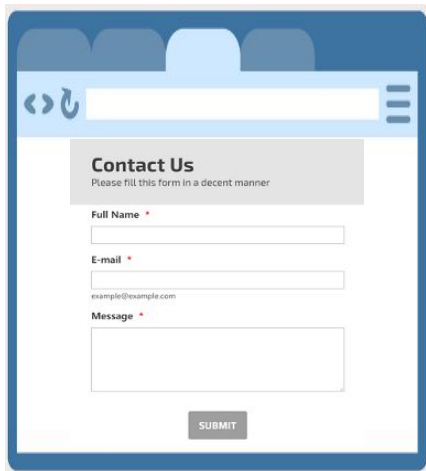
exemplo:

```
app1.dominio.com.br → Encaminha para app1  
app2.dominio.com.br → Encaminha para app2  
api.dominio.com.br  → Encaminha para api
```

CLIENT

SERVER

REQUEST →



Contact Us
Please fill this form in a decent manner

Full Name *

E-mail *

Message *

SUBMIT



Ubuntu 20.04 LTS

HTTP SERVER



Front-end
(estático)

```
minha-app/  
css/  
js/  
imgs/  
index.html
```

Front-end 2
(estático)

```
minha-app-2/  
css/  
imgs/  
index.html
```

← RESPONSE

E se temos Frontend e Backend no mesmo servidor.

Como é a comunicação entre eles?

```
// js/tasks.js

function getTasks() {
  return new Promise((resolve, reject) => {
    fetch('http://api.domain.com/v1/tasks/')
      .then((response) => response.json())
      .then((tasks) => {
        resolve(tasks)
      })
      .catch((error) => {
        reject(error)
      })
  })
}
```

Nossa aplicação Frontend tem uma ação que gera uma nova requisição que deverá chegar até o backend (uma API por exemplo)

Onde é a origem desta requisição?

CLIENT

REQUEST

SERVER

Ubuntu 20.04 LTS

HTTP SERVER



Front-end
(estático)

```
minha-app/  
css/  
js/  
imgs/  
index.html
```

Backend

```
cursos/  
settings/  
core/  
app.py
```

Será que esta requisição tem início dentro do próprio servidor?
Dado que temos o arquivo JS do frontend lá!

A resposta é NÃO!

Vamos pensar que nosso servidor armazena o código do frontend, mas ele entrega tudo para o cliente (um navegador por exemplo). Assim, o código que está lá do lado do cliente vai fazer a requisição para o HTTP SERVER, o qual saberá entender (com base nas configurações) e encaminhar para a aplicação Back-end.



Front-end
(estático)

```
minha-app/  
  css/  
  js/  
  imgs/  
  index.html
```



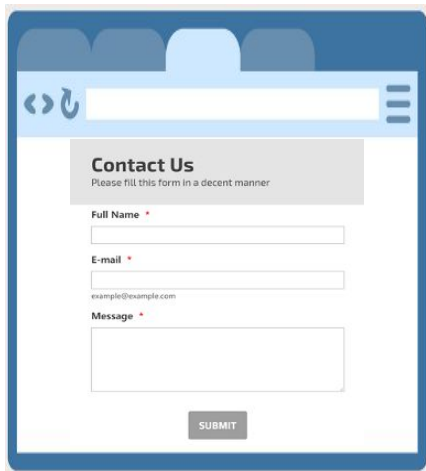
Backend

```
cursos/  
  settings/  
  core/  
  app.py
```

CLIENT

SERVER

REQUEST →



Contact Us
Please fill this form in a decent manner

Full Name *

E-mail *

example@example.com

Message *

SUBMIT

REQUEST →

← RESPONSE



Ubuntu 20.04 LTS

HTTP SERVER



Front-end
(estático)

```
minha-app/  
css/  
js/  
imgs/  
index.html
```



Backend

```
cursos/  
settings/  
core/  
app.py
```


CLIENT

SERVER

REQUEST →

Ubuntu 20.04 LTS



HTTP SERVER

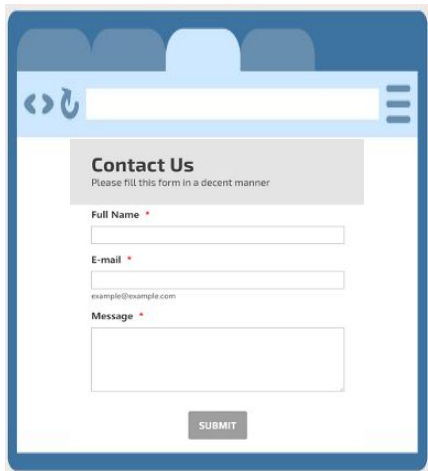


Front-end
(estático)

```
minha-app/  
css/  
js/  
imgs/  
index.html
```

```
minha-app/  
css/  
js/  
imgs/  
index.html
```

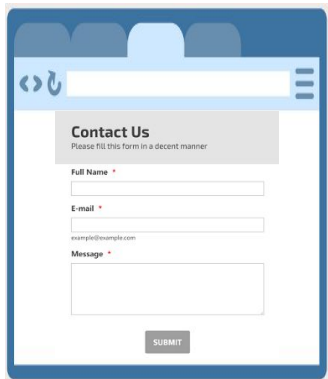
A primeira requisição, vai disparar diversas outras requisições, assim, todo código necessário será levado para o lado do cliente.



CLIENT

SERVER

REQUEST



Cópia do Front-end
no cliente

```
minha-app/  
css/  
js/  
imgs/  
index.html
```



Ubuntu 20.04 LTS

HTTP SERVER



Front-end
(estático)

```
minha-app/  
css/  
js/  
imgs/  
index.html
```

Backend

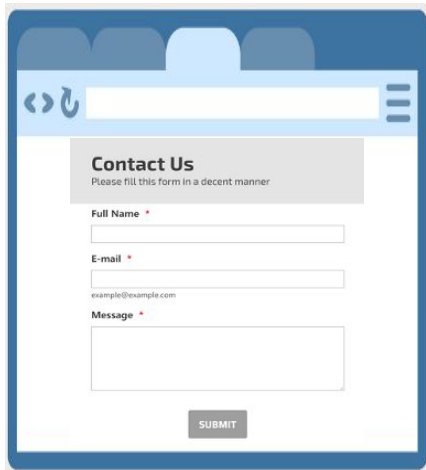
```
cursos/  
settings/  
core/  
app.py
```

O **tasks.js** que faz a requisição para o Back-end tem como origem o lado cliente. Assim irá chegar no HTTP server como as outras requisições, mas esta será encaminhada para o lugar correto (backend)

CLIENT

SERVER

REQUEST →



A web browser window with a blue header and a contact form titled "Contact Us". The form includes fields for "Full Name", "E-mail", and "Message", each with a red asterisk indicating required fields. A "SUBMIT" button is at the bottom. The browser's address bar shows a code icon and a refresh icon.

REQUEST →

← RESPONSE



Ubuntu 20.04 LTS

HTTP SERVER



Front-end
(estático)

```
minha-app/  
css/  
js/  
imgs/  
index.html
```



Backend

```
cursos/  
settings/  
core/  
app.py
```

FIM