REQUISIÇÕES HTTP
(AULA 21)

CURSO BÁSICO DE PROGRAMAÇÃO COM JAVASCRIPT

MAYARA MARQUES

mmrosatab@gmail.com

JS

SUMÁRIO

- O que são requisições HTTP?
- Principais partes de uma requisição
- Verbos HTTP
- Usando os verbos http na Fetch API
- Status de respostas para requisições HTTP
- Mão na massa

REQUISIÇÕES HTTP

Na aula anterior, começamos a entender como poderíamos solicitar uma informação a um outro sistema. Nesta aula, aprofundaremos o conhecimento a respeito de requisições HTTP.

REQUISIÇÕES HTTP

Uma requisição HTTP é um pedido que um cliente envia a um servidor usando o ***protocolo HTTP*** para obter ou enviar informações.

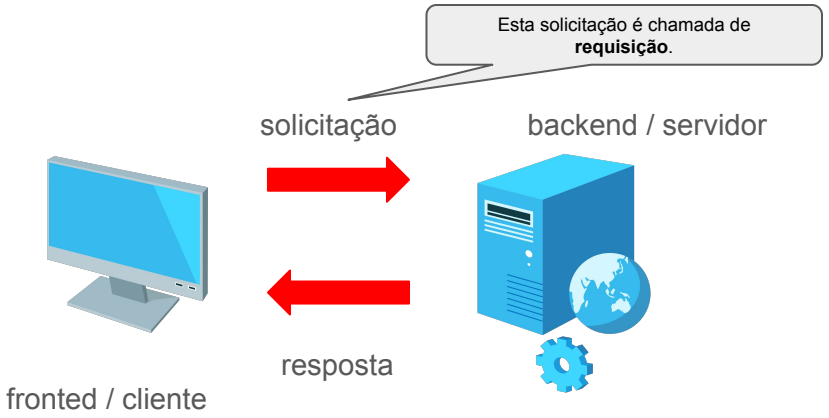
REQUISIÇÕES HTTP

O **protocolo HTTP** é o conjunto de regras que define como esse pedido é feito e como a resposta será enviada de volta.

A **requisição HTTP** é a ação de pedir algo seguindo essas regras.

Exemplo:

- Você digita https://jsonplaceholder.typicode.com/posts no navegador.
- O navegador envia uma requisição HTTP para o servidor deste site pedindo a lista de posts de um usuário.



PARTES PRINCIPAIS DE UMA REQUISIÇÃO

- **Verbo HTTP**
  - Diz que tipo de ação você quer (buscar, criar, atualizar, apagar).
- **URL**
  - O endereço do recurso no servidor.
- **Cabeçalhos (Headers)**
  - Informações extras, como formato esperado (application/json).
- **Corpo (Body)**
  - Dados que você quer enviar (opcional).

VERBOS HTTP

Verbo	Para que serve?	Exemplo
GET	***Buscar*** informações no servidor.	Buscar um post de um usuário.
POST	***Enviar*** dados para o servidor **criar** um recurso.	Criar um post de usuário.
PUT	***Atualizar*** um recurso existente.	Atualizar um post de usuário.
PATCH	***Atualizar parcialmente*** um recurso.	Alterar uma informação em um post de um usuário.
DELETE	***Remover*** um recurso.	Excluir o post.

USANDO OS VERBOS HTTP NA FETCH API

- GET
  - **Buscar** informações no servidor.

```
fetch('https://jsonplaceholder.typicode.com/posts')
  .then(resposta => resposta.json())
  .then(dados => console.log(dados))
  .catch(erro => console.error('Erro:', erro))
```

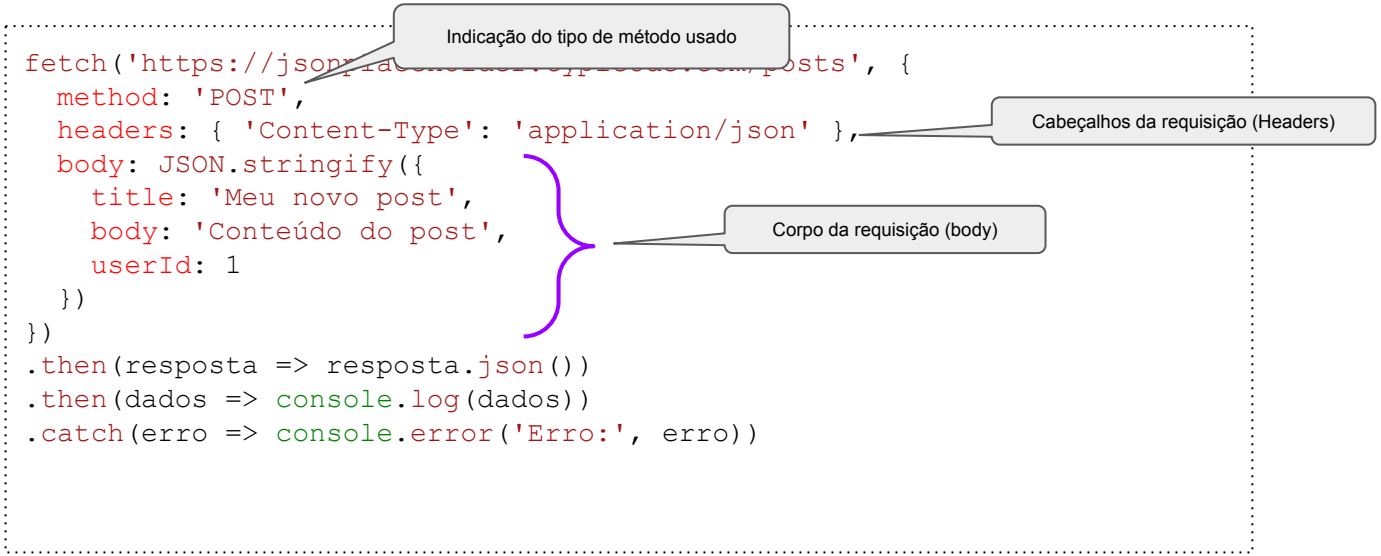
USANDO OS VERBOS HTTP NA FETCH API

- POST
  - **Enviar** dados para o servidor **criar** algo.

```
fetch('https://jsonplaceholder.typicode.com/posts', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    title: 'Meu novo post',
    body: 'Conteúdo do post',
    userId: 1
  })
})
.then(resposta => resposta.json())
.then(dados => console.log(dados))
.catch(erro => console.error('Erro:', erro))
```

USANDO OS VERBOS HTTP NA FETCH API

- POST
  - Dados anexados no corpo (body) da requisição.
  - Indicação de tipo método.




```
fetch('https://jsonplaceholder.typicode.com/posts', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    title: 'Meu novo post',
    body: 'Conteúdo do post',
    userId: 1
  })
})
.then(resposta => resposta.json())
.then(dados => console.log(dados))
.catch(erro => console.error('Erro:', erro))
```

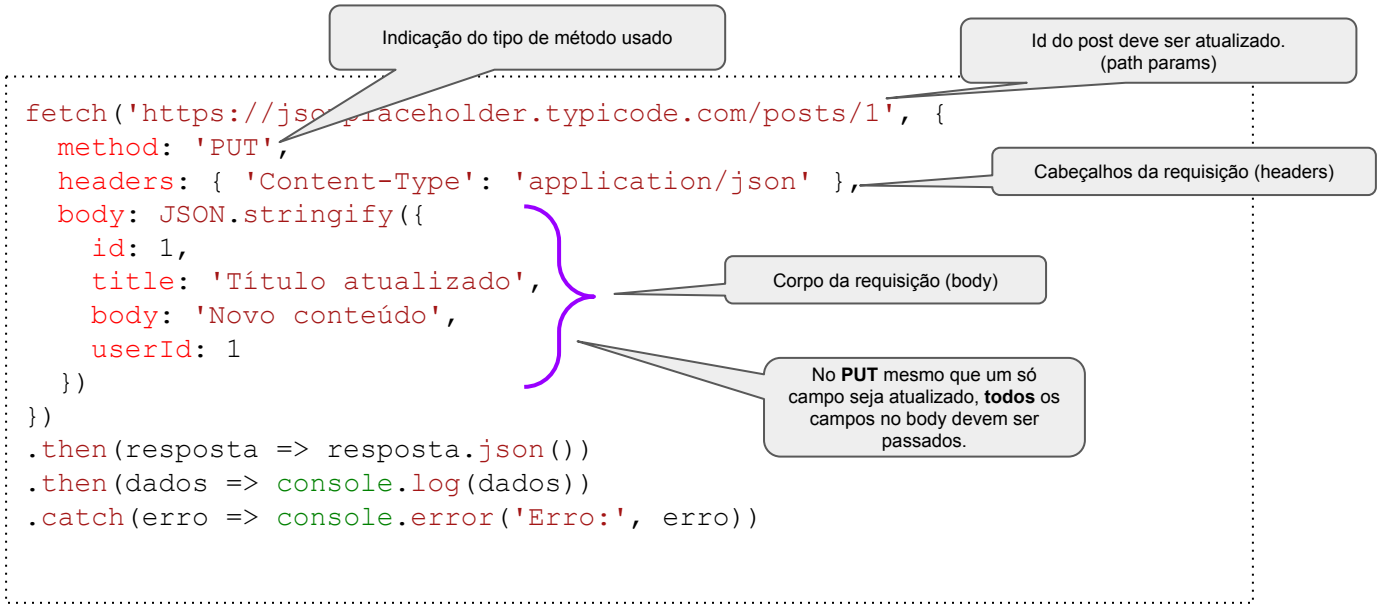
USANDO OS VERBOS HTTP NA FETCH API

- PUT
  - **Atualizar** um recurso existente.

```
fetch('https://jsonplaceholder.typicode.com/posts/1', {
  method: 'PUT',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    id: 1,
    title: 'Título atualizado',
    body: 'Novo conteúdo',
    userId: 1
  })
})
.then(resposta => resposta.json())
.then(dados => console.log(dados))
.catch(erro => console.error('Erro:', erro))
```

USANDO OS VERBOS HTTP NA FETCH API

- PUT
  - Dados anexados no corpo (body) da requisição.
  - Indicação de tipo método.




```
fetch('https://jsonplaceholder.typicode.com/posts/1', {
  method: 'PUT',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    id: 1,
    title: 'Título atualizado',
    body: 'Novo conteúdo',
    userId: 1
  })
})
.then(resposta => resposta.json())
.then(dados => console.log(dados))
.catch(erro => console.error('Erro:', erro))
```

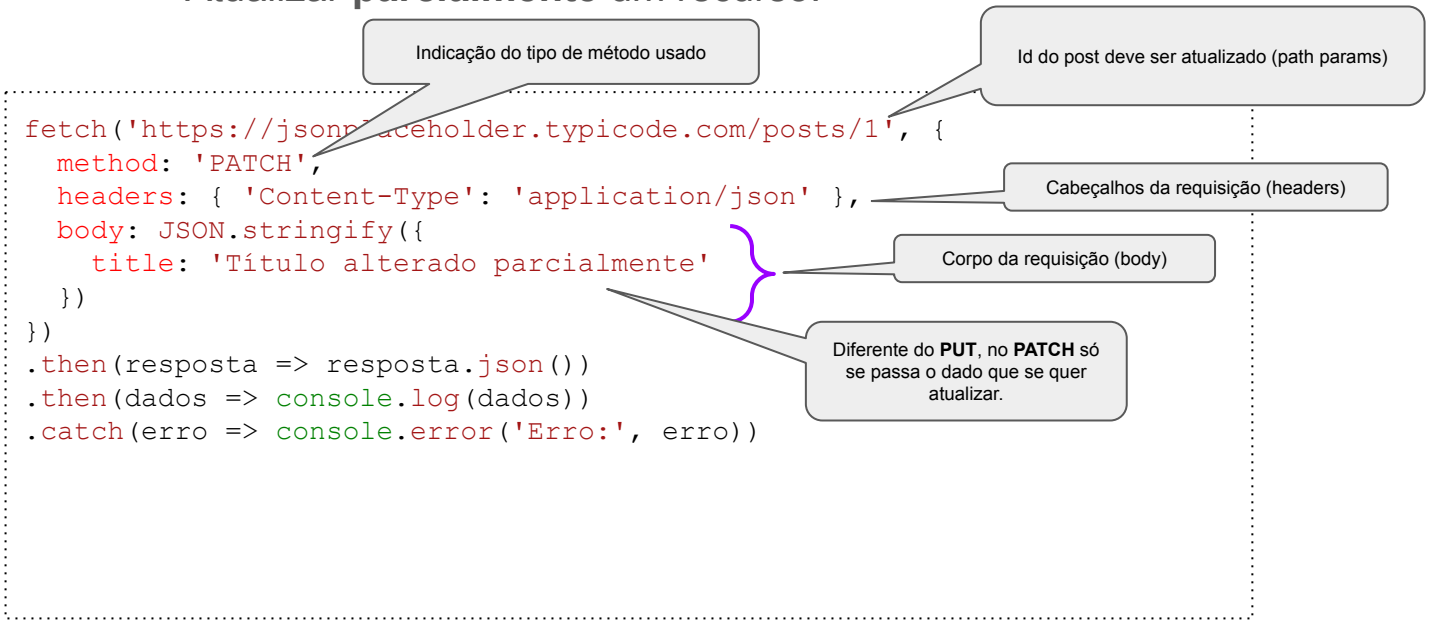
USANDO OS VERBOS HTTP NA FETCH API

- PATCH
  - Atualizar **parcialmente** um recurso.

```
fetch('https://jsonplaceholder.typicode.com/posts/1', {
  method: 'PATCH',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    title: 'Título alterado parcialmente'
  })
})
.then(resposta => resposta.json())
.then(dados => console.log(dados))
.catch(erro => console.error('Erro:', erro))
```

USANDO OS VERBOS HTTP NA FETCH API

- PATCH
  - Atualizar **parcialmente** um recurso.




```
fetch('https://jsonplaceholder.typicode.com/posts/1', {
  method: 'PATCH',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    title: 'Título alterado parcialmente'
  })
})
.then(resposta => resposta.json())
.then(dados => console.log(dados))
.catch(erro => console.error('Erro:', erro))
```

USANDO OS VERBOS HTTP NA FETCH API

- DELETE
  - **Remover** um recurso.

```
fetch('https://jsonplaceholder.typicode.com/posts/1', {
  method: 'DELETE'
})
.then(resposta => {
    if (resposta.ok) console.log('Post deletado com sucesso')
})
.catch(erro => console.error('Erro:', erro))
```

USANDO OS VERBOS HTTP NA FETCH API

- DELETE
  - **Remover** um recurso.

PATH PARAMS X QUERY PARAMS

Path params

Os parâmetros de rota (Path params) fazem parte do próprio caminho da URL e são usados para identificar um recurso específico. Normalmente aparecem entre barras */* na URL.

Ex:

```
'https://jsonplaceholder.typicode.com/posts/1'
```

Aqui, 1 é um **path param** que identifica um post específico.

```
'https://jsonplaceholder.typicode.com/posts/1/comments'
```

Aqui, 1 é um **path param** que identifica um post específico.

PATH PARAMS X QUERY PARAMS

Path params

Quando usar ?

Quando o parâmetro é essencial para a identificação do recurso.

Ex:

```
'https://jsonplaceholder.typicode.com/posts/1'
```

Aqui, 1 é um **path param** que identifica um post específico.

```
'https://jsonplaceholder.typicode.com/posts/1/comments'
```

Aqui, 1 é um **path param** que identifica um post específico.

PATH PARAMS X QUERY PARAMS

Query params

Os parâmetros de consulta (Query params) são adicionados à URL após um **?** e separados por **&**. São usados para **filtrar**, **ordenar** ou **modificar** a resposta de uma API.

Ex:

```
'https://jsonplaceholder.typicode.com/comments?postId=1'
```

Aqui, 1 é um **query param** que identifica um post específico.

```
'https://jsonplaceholder.typicode.com/comments?postId=1&postId=2'
```

PATH PARAMS X QUERY PARAMS

Query params

Quando usar?

Quando os parâmetros são opcionais ou servem para modificar a busca.

Ex:


```
'https://jsonplaceholder.typicode.com/comments?postId=1&postId=2'
```

CÓDIGO DE STATUS HTTP

Os códigos de status indicam o resultado da requisição.

1xx → Informativo (raramente usados diretamente)

2xx → Sucesso

200 OK → Deu tudo certo!

201 Created → Novo recurso criado.

3xx → Redirecionamento

301 Moved Permanently → Página mudou de endereço.

4xx → Erro do cliente

400 Bad Request → Pedido mal formulado.

401 Unauthorized → Precisa se autenticar.

404 Not Found → Recurso não encontrado.

5xx → Erro do servidor

500 Internal Server Error → Erro no servidor.

Leitura recomendada

HTTP Status code

Bónus

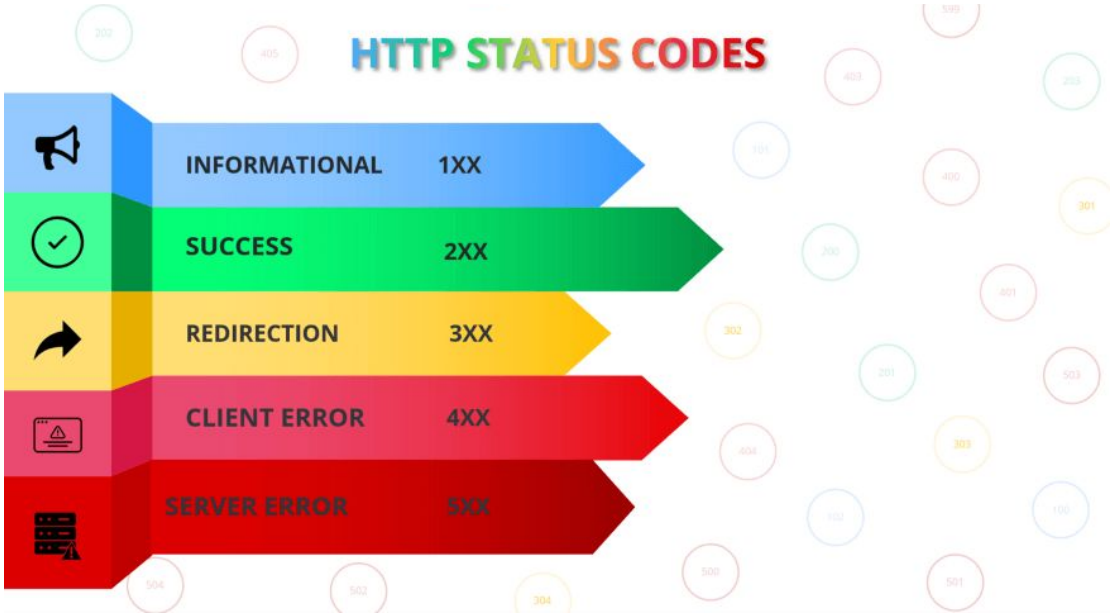


Imagem extraída de: https://www.kuraztech.com/blog/details/http-status-codes-101

API’s legais para estudar

The RESTful Pokémon API

https://pokeapi.co/

HP API

https://hp-api.onrender.com/

Bónus

Dog API

https://dog.ceo/dog-api/

MÃO NA MASSA

MÃO NA MASSA

- Clone em seu computador o repositório da aplicação backend de signos através do link abaixo.
  - HTTPS
    - https://github.com/mmrosadev/signs.git
  - SSH
    - git@github.com:mmrosadev/signs.git
  - Ou acesse o repositório e baixe o zip
    - https://github.com/mmrosadev/signs#
- Abra a pasta do repositório no vscode.
- Dentro do terminal, execute **npm i** ou **npm install**, para instalar as dependências do NodeJS.
- Execute **npm run start** para rodar a aplicação.

MÃO NA MASSA

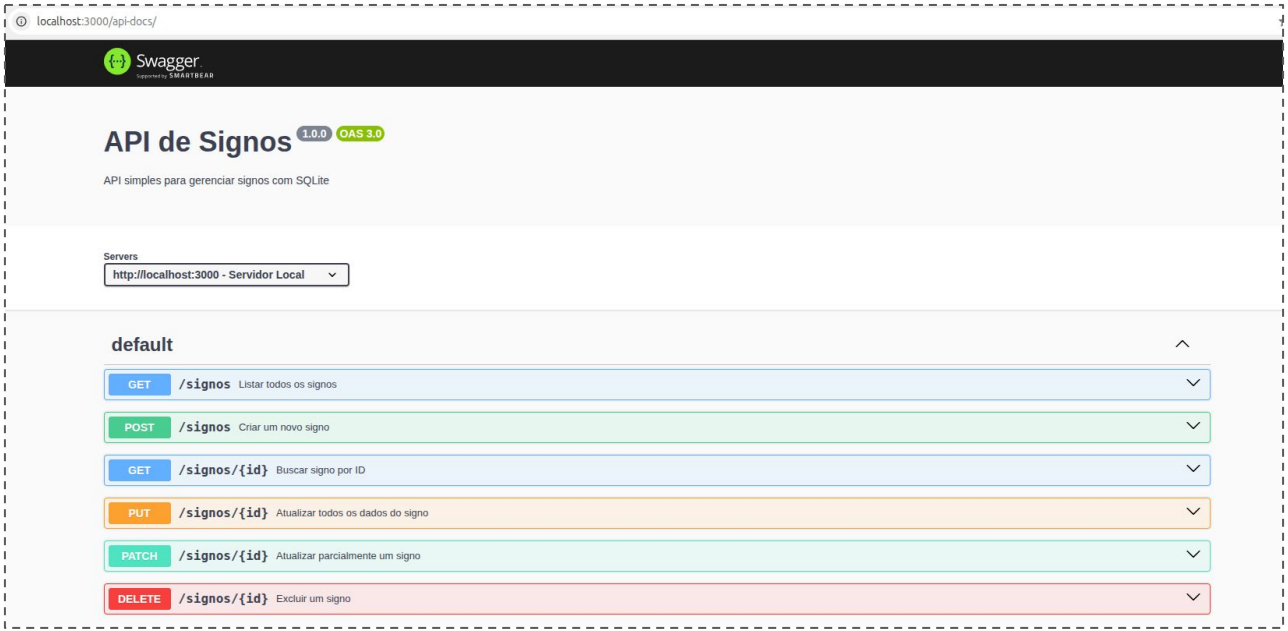
Se tudo der certo, após executar o comando **npm run start** para rodar a aplicação, você verá algo como o mostrado abaixo em seu terminal.

```
> signs@1.0.0 start
> node --watch src/index.js

Signs API running at http://localhost:3000
Connected to SQLite database.
```

MÃO NA MASSA

Acesse: http://localhost:3000/api-docs/ em seu navegador para visualizar a documentação da api de signos.



MÃO NA MASSA

Com a aplicação de signos rodando em seu computador. Abra o **postman** e realize as atividades abaixo:

1. Crie um signo.
2. Liste o signo recém criado pelo ***id*** dele.
3. Crie outro signo.
4. Liste ***todos*** os signos criados até o momento.
5. Altere algum registro de um signo usando o PUT.
6. Altere algum registro de um signo usando o PATCH.
7. Delete um registro de signo qualquer usando o **id** dele.

MÃO NA MASSA

Agora, implemente funções que usam a fetch api para realizar as operações abaixo:

1. Criar um signo.
2. Lista um signo por id.
3. Listar todos os signos cadastrados.
4. Alterar algum registro de uma signo usando o PUT.
5. Alterar algum registro de uma signo usando o PATCH.
6. Deletar um registro de signo qualquer usando o id dele.

REFERÊNCIAS

- https://developer.mozilla.org/pt-BR/docs/Glossary/Callback_function
- https://www.w3schools.com/js/js_function_definition.asp
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Working_with_objects