Verbos HTTP e Status Codes

Os verbos HTTP são métodos utilizados pelo protocolo HTTP para indicar a ação desejada para um recurso específico. Cada verbo tem um propósito e significado distintos:

1. **GET**: Solicita a representação de um recurso específico. As requisições utilizando GET devem apenas recuperar dados (**leitura**).
2. **POST**: Envia dados ao servidor para **criar** ou atualizar um recurso. Os dados são incluídos no corpo da requisição.
3. **PUT**: Usado para **atualizar** um recurso existente com os dados fornecidos. Diferente do POST, o PUT geralmente é utilizado para substituir o recurso completo.
4. **DELETE**: Usado para **remover** um recurso específico do servidor.
5. **PATCH**: Aplica modificações parciais a um recurso.
6. **OPTIONS**: Descreve as opções de comunicação para o recurso de destino.
7. **HEAD**: Solicita uma resposta idêntica à de uma requisição GET, mas sem o corpo da resposta.

Status Codes

Os códigos de status HTTP são respostas padronizadas fornecidas pelos servidores para indicar o resultado da requisição. Eles são agrupados em cinco classes:

1. **1xx (Informacional)**: A requisição foi recebida e o processo continua.
2. **2xx (Sucesso)**: A requisição foi recebida, compreendida e aceita com sucesso.
    - **200 OK**: A requisição foi bem-sucedida.
    - **201 Created**: A requisição foi bem-sucedida e um novo recurso foi criado.
3. **3xx (Redirecionamento)**: É necessário tomar medidas adicionais para completar a requisição.
4. **4xx (Erro do Cliente)**: A requisição contém erros que impediram o servidor de processá-la.
    - **400 Bad Request**: A requisição não pôde ser entendida pelo servidor devido à sintaxe malformada.
    - **401 Unauthorized**: A requisição requer autenticação do usuário.
    - **404 Not Found**: O servidor não encontrou o recurso solicitado.
5. **5xx (Erro do Servidor)**: O servidor falhou ao tentar processar a requisição.
    - **500 Internal Server Error**: O servidor encontrou uma condição inesperada que o impediu de completar a requisição.
    - **503 Service Unavailable**: O servidor está temporariamente indisponível (normalmente devido à sobrecarga ou manutenção).

Configurando diferentes Rotas e Respostas

Criação do **Servidor**

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

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

server.listen(3000, '127.0.0.1', () => {
  console.log('Servidor rodando em http://127.0.0.1:3000/');
});
```

Rotas **GET**

```
const server = http.createServer((req, res) => {
    if (req.method === 'GET' && req.url === '/') {
        res.statusCode = 200;
        res.setHeader('Content-Type', 'text/plain');
        res.end('Home\n');

    } else if (req.method === 'GET' && req.url === '/about') {
        res.statusCode = 200;
        res.setHeader('Content-Type', 'text/plain');
        res.end('Sobre');

    } else {
        res.statusCode = 404;
        res.end('Not Found');
    }
});

server.listen(3000, '127.0.0.1', () => {
    console.log('Servidor rodando em http://127.0.0.1:3000/');
});
```

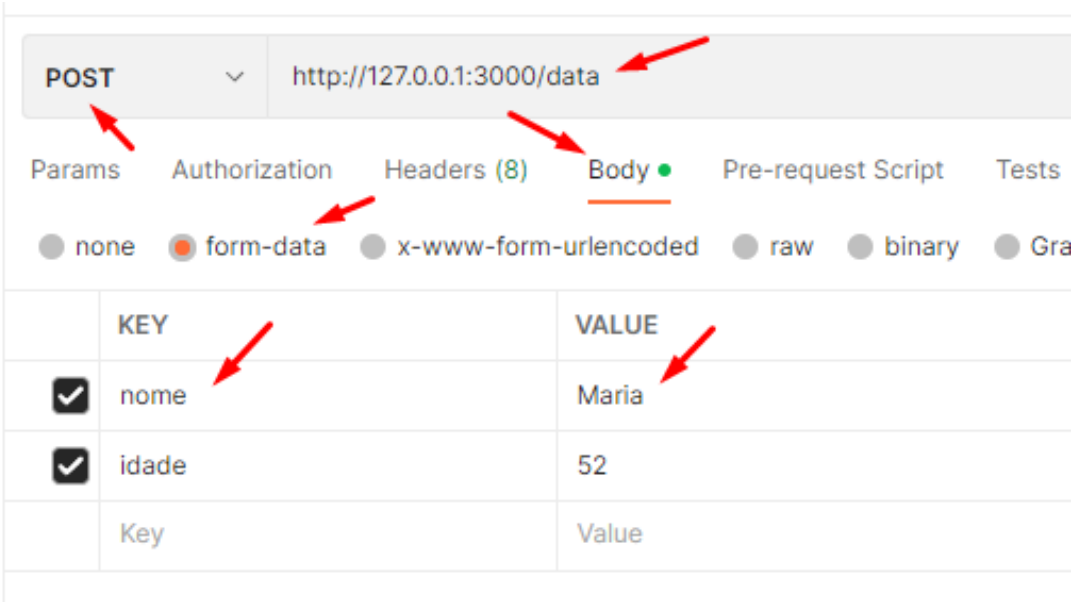
Rotas **POST**

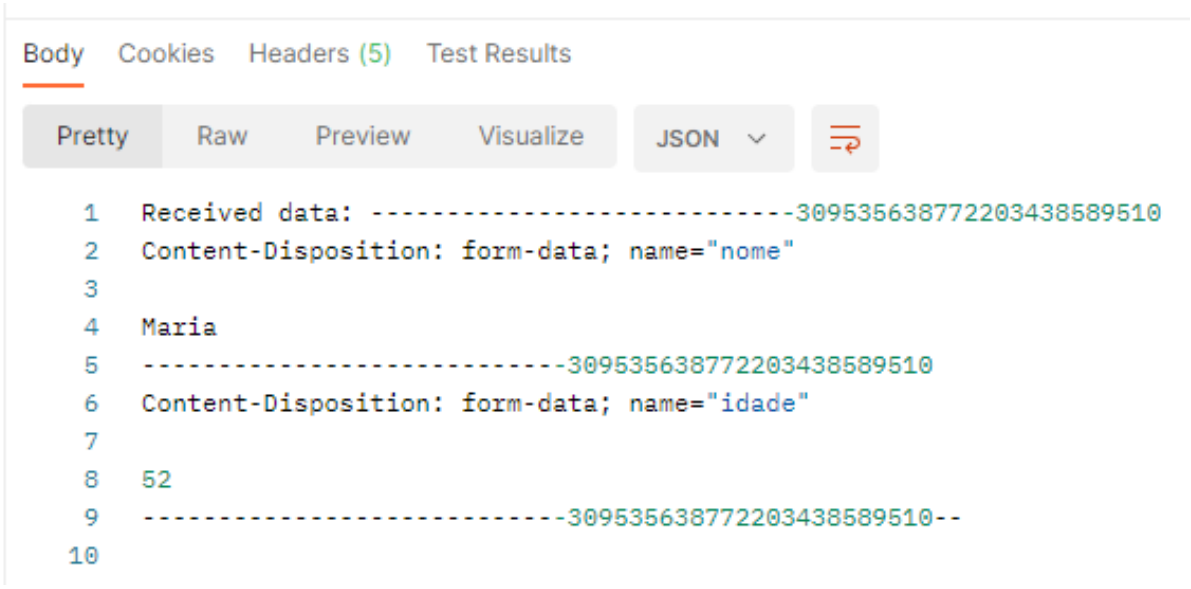
```
const server = http.createServer((req, res) => {
    if (req.method === 'POST' && req.url === '/data') {
        let body = '';
        req.on('data', chunk => {
            body += chunk.toString();
        });

        req.on('end', () => {
            res.statusCode = 200;
            res.setHeader('Content-Type', 'application/json');
            res.end(`Received data: ${body}`);
        });

    } else {
        res.statusCode = 404;
        res.end('Not Found');
    }
});

server.listen(3000, '127.0.0.1', () => {
    console.log('Servidor rodando em http://127.0.0.1:3000/');
});
```

Para testar, podemos usar **cURL**:

- No cmd
- `curl -X POST http://127.0.0.1:3000/data -d "name=Marnei&age=47"`

Podemos testar com **Postman** (exemplo) ou **Insomnia**





Rotas **PUT**

- `curl -X PUT http://127.0.0.1:3000/update -d "name=Marnei&age=23"`

```
const server = http.createServer((req, res) => {
    if (req.method === 'PUT' && req.url === '/update') {
        let body = '';
        req.on('data', chunk => {
            body += chunk.toString();
        });
        req.on('end', () => {
            res.statusCode = 200;
            res.setHeader('Content-Type', 'application/json');
            res.end(`Updated data: ${body}`);
        });
    } else {
        res.statusCode = 404;
        res.end('Not Found');
    }
});

server.listen(3000, '127.0.0.1', () => {
    console.log('Servidor rodando em http://127.0.0.1:3000/');
});
```

Rotas **DELETE**

- `curl -X DELETE http://127.0.0.1:3000/delete`

```
const server = http.createServer((req, res) => {
    if (req.method === 'DELETE' && req.url === '/delete') {
        res.statusCode = 200;
        res.setHeader('Content-Type', 'text/plain');
        res.end('Deleted resource');

    } else {
        res.statusCode = 404;
        res.end('Not Found');
    }
});

server.listen(3000, '127.0.0.1', () => {
    console.log('Servidor rodando em http://127.0.0.1:3000/');
});
```