Métodos HTTP - Códigos de Resposta e Mensagens de Erro

José Glauber - UFCG
2025.1

Requisições e respostas

Entender como os sistemas web se comunicam e respondem às solicitações dos clientes.

Quais tipos de métodos HTTP existem?

GET, POST, PUT, PATCH e DELETE

HEAD, OPTIONS…

Quando devemos utilizar cada método?

Quais mensagens de erro é esperado de cada método?

Códigos de erro

1xx: informativo

2xx: Sucesso

3xx: Redirecionamento

4xx: Erro do cliente (por exemplo, erro de solicitação)

5xx: Erro do servidor (por exemplo, falha interna do servidor)

200 OK: A solicitação foi bem-sucedida. O conteúdo da resposta depende do método HTTP (por exemplo, GET retorna dados, POST retorna o resultado da operação).

- **Exemplo**: Uma solicitação GET para /users/1 retorna os dados do usuário com ID 1.

201 Created: A solicitação foi bem-sucedida e um novo recurso foi criado. Normalmente usado com métodos POST ou PUT.

- **Exemplo**: Uma solicitação POST para /users cria um novo usuário e retorna os dados do usuário criado.

204 No Content: A solicitação foi bem-sucedida, mas não há conteúdo para enviar na resposta. Usado frequentemente com DELETE.

- **Exemplo**: Uma solicitação DELETE para /users/1 remove o usuário com ID 1 e retorna uma resposta vazia.

400 Bad Request: A solicitação do cliente é inválida ou malformada. O servidor não pode processá-la.

- **Exemplo**: Uma solicitação POST para /users com dados de formulário inválidos retorna um erro 400.

401 Unauthorized: A solicitação requer autenticação. O cliente deve fornecer credenciais válidas.

- **Exemplo**: Uma solicitação GET para /profile sem um token de autenticação retorna um erro 401.

403 Forbidden: O servidor entendeu a solicitação, mas o cliente não tem permissão para acessá-la.

- **Exemplo**: Uma solicitação DELETE para /admin/users/1 por um usuário não administrador retorna um erro 403.

404 Not Found: O servidor não encontrou o recurso solicitado.

- **Exemplo**: Uma solicitação GET para /non-existent-page retorna um erro 404.

409 Conflict: A solicitação não pôde ser concluída devido a um conflito com o estado atual do recurso.

- **Exemplo**: Uma solicitação POST para /users com um nome de usuário já existente retorna um erro 409.

500 Internal Server Error: O servidor encontrou uma condição inesperada que impediu a solicitação de ser atendida.

- **Exemplo**: Uma exceção não tratada no servidor ao processar uma solicitação GET para /users retorna um erro 500.

502 Bad Gateway: O servidor, atuando como um gateway ou proxy, recebeu uma resposta inválida do servidor upstream.

- **Exemplo**: Uma solicitação GET para /api/data retorna um erro 502 se o servidor upstream está inoperante.

Métodos GET

O **método HTTP GET** solicita uma representação do recurso especificado. Solicitações usando GET só devem recuperar dados.

Requisição tem corpo	Não
Resposta bem-sucedida tem corpo	Sim
Seguro	Sim
Idempotente	Sim
Cacheável	Sim
Permitido em formulários HTML	SIm

→ não altera o estado do servidor

→ uma requisição idêntica pode ser feita uma ou mais vezes em sequência com o mesmo efeito enquanto deixa o servidor no mesmo estado

Métodos POST

O **método HTTP POST** envia dados ao servidor. O tipo do corpo da solicitação é indicado pelo cabeçalho Content-Type.

Requisição tem corpo	Sim
Resposta bem-sucedida tem corpo	Sim
Seguro	Não
Idempotente	Não
Cacheável	Somente se as informações de atualização estiverem incluídas
Permitido em formulários HTML	Sim

Métodos PUT/PATCH

O **método de requisição HTTP PUT** cria um novo recurso ou substitui uma representação do recurso de destino com os novos dados.

A requisição tem corpo	Sim
Resposta bem sucedida tem corpo	Não
Safe	Não
Idempotent	Sim
Cacheable(inglês)	Não
Permitido em formulários HTML	Não

Métodos Delete

O **método de requisição HTTP DELETE** remove o recurso especificado.

Requisição tem corpo	Talvez
A resposta bem sucedida tem corpo	Talvez
Seguro	Não
Idempotente	Sim
Cacheável	Não
Aceito nos formulários HTML	Não

Hot reload ou Hot Module Replacement - HMR

Permite que o servidor se atualize automaticamente durante o desenvolvimento sem a necessidade de reinicialização manual.

Como funciona?

monitora os arquivos do projeto para mudanças e, quando uma mudança é detectada, ele recompila e recarrega apenas os módulos afetados, sem reiniciar completamente a aplicação.