📝 Cheatsheet APIs REST: Métodos HTTP

> **Nota:** Este guia é útil tanto para iniciantes quanto para quem precisa de uma referência rápida sobre APIs REST e métodos HTTP.

🔹 1. Conceitos básicos

> **REST** (Representational State Transfer) é um estilo arquitetural para comunicação entre sistemas distribuídos via HTTP.

- **Idempotência:** repetir a mesma requisição várias vezes **não muda o resultado** além da primeira execução.
  - Ex.: `PUT` ✅, `DELETE` ✅, `POST` ❌
- **Stateless:** o servidor **não mantém estado** entre requisições. Cada requisição deve trazer todas as informações necessárias.
- **Representação:** recursos podem ter múltiplas representações (JSON, XML, etc.).
- **Semântica vs Técnica:** métodos diferentes (`GET`, `POST`, `PUT`, `PATCH`, `DELETE`) comunicam **a intenção da ação**, embora tecnicamente seja possível fazer tudo apenas com `GET` e `POST`.

🔹 2. Semântica dos métodos HTTP

> 🔑 Seguir a semântica correta facilita manutenção, integração com clientes, cache, proxies e segurança.

- **GET:** ler dados
- **POST:** criar novos recursos
- **PUT:** substituir totalmente um recurso
- **PATCH:** atualizar parcialmente um recurso
- **DELETE:** remover um recurso

> Mesmo que tudo possa ser feito com GET/POST, usar cada método corretamente é **boa prática REST**.

🔹 3. Tabela comparativa de métodos

Método	Função principal	Idempotente?	Exemplo de uso	Emoji Representativo
GET	Ler dados	✅	`/tasks` → lista de tarefas	👀
POST	Criar dados	❌	`/tasks` → cria nova tarefa	➕
PUT	Substituir dados	✅	`/tasks/2` → atualiza totalmente a tarefa 2	✏️

Método	Função principal	Idempotente?	Exemplo de uso	Emoji Representativo
PATCH	Atualizar parcialmente	✅/❌	`/tasks/2` → altera apenas o status da tarefa	🔄
DELETE	Remover dados	✅	`/tasks/2` → deleta a tarefa 2	❌

🔹 4. Analogia com CRUD

```
Create → POST ➕
Read   → GET 👀
Update → PUT/PATCH ✏️/🔄
Delete → DELETE ❌
```

> Essa analogia ajuda a mapear rapidamente ações comuns em APIs REST.

🔹 5. Explicação individual e exemplos

GET 👀

> Ler dados sem alterar o estado do servidor.

```
GET /tasks
200 OK
[
  {"id":1,"title":"Estudar REST","done":false},
  {"id":2,"title":"Fazer exercício","done":true}
]
```

> ✅ **Idempotente:** ler repetidamente o mesmo recurso retorna o mesmo resultado.

POST ➕

> Criar novos recursos.

```
POST /tasks
Content-Type: application/json

{
  "title": "Fazer exercício",
  "done": false
}
201 Created
```

> ❌ **Não idempotente:** duas requisições criam dois recursos distintos.

PUT ✏️

> Substituir totalmente um recurso existente.

```
PUT /tasks/1
Content-Type: application/json

{
  "title": "Estudar REST avançado",
  "done": true
}
200 OK
```

> ✅ **Idempotente:** aplicar várias vezes o mesmo conteúdo mantém o mesmo estado.

PATCH 🔄

> Atualizar parcialmente um recurso.

```
PATCH /tasks/1
Content-Type: application/json

{
  "done": true
}
200 OK
```

- Útil quando só alguns campos precisam mudar.
- Pode ser idempotente dependendo da implementação.

DELETE ❌

> Remover um recurso.

```
DELETE /tasks/1
204 No Content
```

> ✅ **Idempotente:** deletar repetidamente não altera o estado adicionalmente.

🔹 6. Boas práticas rápidas

> ⚠️ Dicas para projetar APIs REST claras e seguras:

- Use **URLs no plural**: `/tasks` em vez de `/task`.
- Retorne **status codes HTTP corretos**:
  - `200 OK` → sucesso de leitura ou atualização
  - `201 Created` → criação bem-sucedida
  - `204 No Content` → operação concluída sem resposta
  - `400 Bad Request` → requisição inválida
  - `404 Not Found` → recurso não encontrado
  - `500 Internal Server Error` → erro no servidor
- Prefira **semântica correta** dos métodos.
- Para alterações parciais, use `PATCH`; para substituições completas, use `PUT`.
- Sempre use **HTTPS** e autenticação; valide inputs; evite expor dados sensíveis.

🧭 Leituras adicionais

- 📘 MDN Web Docs — Métodos de Requisição HTTP
  Guia completo e atualizado da Mozilla com descrições, exemplos e comportamento de cada método.
- 📙 RFC 9110 — HTTP Semantics (IETF)
  Especificação oficial dos métodos HTTP segundo o padrão da Internet Engineering Task Force.
- 📗 REST API Tutorial — HTTP Methods
  Explicações práticas sobre o uso de métodos HTTP dentro do contexto REST, com tabelas e exemplos claros.