# **Conceitos e Funcionamento de APIs RESTful em PHP**

**Aluno:** Guilherme Silva  
 **Curso:** Informática para Internet  
 **Disciplina:** Sistemas Web  
 **Professor:** Anderson Vanin  
 **Data:** 23/06/2025

## **1. Introdução**

No desenvolvimento de sistemas modernos, a integração entre aplicações tornou-se uma necessidade cada vez mais comum. Seja para conectar um aplicativo mobile a um servidor, consumir serviços de terceiros como Google Maps, ou até mesmo integrar sistemas internos de uma empresa, a tecnologia de APIs (Interfaces de Programação de Aplicações) tem se tornado o padrão. Esta pesquisa aprofunda-se nos conceitos, funcionamento e estrutura de APIs RESTful, com ênfase na implementação em PHP puro, linguagem amplamente utilizada no desenvolvimento web.

## **2. O que é uma API e para que serve**

A sigla **API** vem do inglês *Application Programming Interface*, traduzido como *Interface de Programação de Aplicações*. Trata-se de um conjunto de regras, protocolos e ferramentas que permite a comunicação entre dois sistemas distintos. As APIs expõem funcionalidades internas de um sistema para que outros softwares possam interagir com ele sem acesso direto ao seu código-fonte.

### **2.1 Exemplo prático:**

Imagine um sistema de e-commerce que deseja calcular o frete via Correios. Em vez de implementar toda a lógica do zero, o sistema utiliza a **API dos Correios**, enviando informações como peso e CEP de destino. A API retorna o valor e prazo de entrega. Esse é um exemplo clássico de como APIs facilitam o desenvolvimento modular e ágil.

## **3. Diferença entre API REST e SOAP**

Existem diversos padrões para implementação de APIs. Os dois principais são **REST** e **SOAP**, cada um com suas características, vantagens e limitações.

### **3.1 SOAP (Simple Object Access Protocol)**

* Baseado no protocolo XML.
* Estrutura rígida e formal.
* Suporta múltiplos protocolos (HTTP, SMTP, TCP).
* Requer contratos (WSDL) bem definidos.

### **3.2 REST (Representational State Transfer)**

* Baseado em HTTP e no uso de verbos HTTP.
* Simples, leve e altamente performático.
* Utiliza formatos flexíveis como JSON e XML (predominantemente JSON).
* Segue o conceito de recursos identificados por URLs.

### **3.3 Comparativo:**

|  |  |  |
| --- | --- | --- |
| **Característica** | **REST** | **SOAP** |
| Complexidade | Baixa | Alta |
| Formato dos dados | JSON, XML | XML |
| Suporte a verbos HTTP | Sim (GET, POST, PUT, DELETE...) | Limitado |
| Padrão de mercado | Moderno e amplamente utilizado | Antigo, usado em sistemas legados |
| Facilidade de uso | Alta | Média/baixa |
| Suporte a cache | Sim | Não |

## **4. Como funcionam as rotas em uma API RESTful**

Uma API RESTful trabalha com o conceito de **recursos**, que são representados por **URLs** (ou "rotas"). Cada rota representa um recurso (como usuário, produto, pedido), e o tipo de operação a ser feita é determinado pelo **verbo HTTP** utilizado na requisição.

### **4.1 Conceito de recurso**

plaintext

CopiarEditar

/api/usuarios  
/api/produtos  
/api/pedidos

### **4.2 Mapeamento das rotas com verbos HTTP**

|  |  |  |
| --- | --- | --- |
| **Método HTTP** | **URL** | **Ação** |
| GET | /api/usuarios | Listar todos os usuários |
| GET | /api/usuarios/1 | Obter usuário com ID 1 |
| POST | /api/usuarios | Criar um novo usuário |
| PUT | /api/usuarios/1 | Atualizar usuário com ID 1 |
| DELETE | /api/usuarios/1 | Deletar usuário com ID 1 |

Essa padronização torna a API intuitiva e de fácil consumo por qualquer cliente que conheça as convenções REST.

## **5. Principais métodos HTTP (GET, POST, PUT, DELETE)**

### **5.1 GET**

Utilizado para **consultar informações**. Não modifica os dados do servidor.

bash

CopiarEditar

GET /api/produtos

Resposta:

json

CopiarEditar

[  
 {  
 "id": 1,  
 "nome": "Notebook",  
 "preco": 3500.00  
 }  
]

### **5.2 POST**

Usado para **criar um novo recurso**. Os dados são enviados no corpo da requisição.

bash

CopiarEditar

POST /api/produtos

Corpo da requisição:

json

CopiarEditar

{  
 "nome": "Mouse",  
 "preco": 89.90  
}

### **5.3 PUT**

Utilizado para **atualizar** recursos existentes.

bash

CopiarEditar

PUT /api/produtos/1

Corpo da requisição:

json

CopiarEditar

{  
 "nome": "Notebook Gamer",  
 "preco": 4700.00  
}

### **5.4 DELETE**

Usado para **excluir** um recurso.

bash

CopiarEditar

DELETE /api/produtos/1

## **6. Estrutura básica de uma API RESTful em PHP puro**

Embora existam frameworks como Laravel, Slim e Lumen, é possível criar uma API funcional apenas com PHP nativo, o que é ideal para aprendizado.

### **6.1 Estrutura de arquivos**

bash

CopiarEditar

/api  
 ┣ index.php  
 ┣ conexao.php  
 ┣ usuarios/  
 ┣ listar.php  
 ┣ criar.php  
 ┣ atualizar.php  
 ┣ deletar.php

### **6.2 Exemplo prático: index.php**

php

CopiarEditar

<?php  
header("Content-Type: application/json");  
$request = $\_SERVER['REQUEST\_URI'];  
$method = $\_SERVER['REQUEST\_METHOD'];  
  
switch ($request) {  
 case '/api/usuarios':  
 if ($method == 'GET') {  
 include 'usuarios/listar.php';  
 } elseif ($method == 'POST') {  
 include 'usuarios/criar.php';  
 }  
 break;  
 case (preg\_match('/\/api\/usuarios\/\d+/', $request) ? true : false):  
 $id = basename($request);  
 if ($method == 'PUT') {  
 include 'usuarios/atualizar.php';  
 } elseif ($method == 'DELETE') {  
 include 'usuarios/deletar.php';  
 }  
 break;  
 default:  
 http\_response\_code(404);  
 echo json\_encode(["erro" => "Rota não encontrada"]);  
}  
?>

## **7. Vantagens e desvantagens de usar arquivos JSON em vez de banco de dados**

### **7.1 Vantagens:**

* **Simplicidade:** Fácil leitura e escrita, sem necessidade de servidor de banco de dados.
* **Portabilidade:** Arquivos podem ser movidos, copiados e versionados.
* **Velocidade de desenvolvimento:** Ideal para prototipação e testes locais.

### **7.2 Desvantagens:**

* **Baixo desempenho em grande escala:** Leitura e escrita em arquivos pode ser lenta.
* **Falta de integridade:** Não há controle transacional, como em bancos relacionais.
* **Concorrência:** Difícil gerenciar múltiplas requisições simultâneas.

### **7.3 Exemplo de leitura de arquivo JSON em PHP:**

php

CopiarEditar

<?php  
$json = file\_get\_contents('usuarios.json');  
$usuarios = json\_decode($json, true);  
echo json\_encode($usuarios);  
?>

## **8. Conclusão**

As APIs RESTful revolucionaram a forma como aplicações web e mobile se comunicam, proporcionando **desempenho, organização e escalabilidade**. Neste trabalho, foram abordados de forma aprofundada os conceitos fundamentais de API, as diferenças entre REST e SOAP, as rotas, métodos HTTP, implementação em PHP puro e considerações sobre uso de arquivos JSON como alternativa aos bancos de dados.

Dominar essas tecnologias é essencial para desenvolvedores modernos que desejam construir sistemas modulares, eficientes e preparados para integração com diversos serviços e plataformas.