CENTRO PAULA SOUZA
ETEC PROF. MARIA CRISTINA MEDEIROS
Técnico em Informática para Internet Integrado ao Ensino Médio

Pedro Henrique Assunção Medeiros

Conceitos e funcionamento de APIs RESTful em PHP sem uso de frameworks

Ribeirão Pires
2025

O que é uma API e para que serve

Uma API (Application Programming Interface) é um conjunto de regras que estabelece a comunicação entre diferentes sistemas, ela define como eles irão interagir, permite configurar solicitações à um sistema por códigos e devolver de maneira simplificada ao usuário, sendo por JSON, XML ou integrada a um banco SQL. No caso de uma API Web isso é feito pela internet, porém existem mais de um tipo de API's como de bibliotecas ou locais.

Diferença entre API REST e SOAP

A SOAP (Protocolo Simples de Acesso a Objetos) e REST (Transferência Representacional de Estado) são dois modelos utilizados para comunicação entre aplicações. O SOAP é um protocolo, enquanto o REST é considerado um estilo de arquitetura para projetar interfaces de comunicação entre sistemas.
14 A API SOAP expõe as operações, ou seja, os métodos que podem ser executados. Já a API REST expõe os dados, permitindo a manipulação de recursos via métodos HTTP. O REST opera exclusivamente sobre o protocolo HTTP.
Ao formato dos dados, o SOAP suporta apenas XML, o que pode aumentar a complexidade de integração. O REST é mais flexível, oferecendo suporte a diversos formatos como o XML, JSON e HTML.
O SOAP também tende a ser mais lento, pois as mensagens são maiores. O REST apresenta melhor desempenho por utilizar mensagens menores.
Em escalabilidade, o SOAP é menos eficiente, pois mantém as interações entre cliente e servidor, o que exige maior capacidade de armazenamento e controle. Já o REST é stateless (sem estado), tratando cada requisição de forma independente, o que facilita a escalabilidade da aplicação.
Em relação à segurança, o SOAP oferece mecanismo, mas com sobrecarga adicional no processamento. O REST utiliza HTTP, sem comprometer a performance do sistema.
Por fim, o SOAP é mais indicado para sistemas legados, integrações corporativas complexas e APIs privadas, enquanto o REST é utilizado em aplicações modernas, web services e APIs públicas devido à sua leveza e simplicidade.

Como funcionam as rotas em uma API RESTful e Os principais métodos HTTP (GET, POST, PUT, DELETE)

Em vez de criar rotas manualmente a API RESTful faz isso por meio da URL e requisições HTTP, sendo programada geralmente com ID's para identificação.
O tipo de operação lembra as operações CRUD (Create, Read, Update e Delete.)
Isso pois as requisições podem ser as seguintes:
GET: utilizado para listar dados (inclusive é o método padrão do navegador);
POST: usado para criar novos dados no sistema;
PUT: utilizado para atualizar os dados de algo já existente;
DELETE: utilizado para remover dados.
Todas essas requisições podem ser feitas de maneira específica, buscando um ID por exemplo e trocando apenas o nome ao invés do conteúdo.

Estrutura básica de uma API em PHP puro

```php
<?php
// Define que a resposta será em JSON e usa UTF-8 para acentuação correta
header("Content-Type: application/json; charset=UTF-8");

// Captura o método HTTP da requisição (GET, POST, PUT, DELETE, etc.)
$metodo = $_SERVER['REQUEST_METHOD'];

// Nome do arquivo onde os dados dos usuários serão armazenados
$arquivo = 'usuarios.json';

// Se o arquivo não existir, cria um novo com um array vazio
if (!file_exists($arquivo)) {
    file_put_contents($arquivo, json_encode([], JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE));
}

// Lê os dados do arquivo JSON e converte em array associativo
$usuarios = json_decode(file_get_contents($arquivo), true);

// Início da estrutura condicional com base no método HTTP
switch ($metodo) {

    // ============================================
    // MÉTODO GET: Retorna todos os usuários ou um por ID
    // ============================================
    case 'GET':
        if (isset($_GET['id'])) {
            $id = intval($_GET['id']); // Converte o ID para número inteiro
            $usuario_encontrado = null;

            // Procura o usuário com o ID correspondente
            foreach ($usuarios as $usuario) {
                if ($usuario['id'] == $id) {
                    $usuario_encontrado = $usuario;
                    break;
                }
            }

            // Se encontrar, retorna o usuário
            if ($usuario_encontrado) {
                echo json_encode($usuario_encontrado, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
            } else {
                // Caso contrário, retorna erro 404
                http_response_code(404);
                echo json_encode(["erro" => "Usuário não encontrado."], JSON_UNESCAPED_UNICODE);
            }
        } else {
            // Sem ID: retorna todos os usuários
            echo json_encode($usuarios, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
        }
```

Nesse caso foi feita uma API simples usando JSON para leitura dos códigos, no exemplo é feito um cabeçalho para definir a resposta, captura do método (http) para uma APIRESTful. Posteriormente o caminho para onde os dados serão enviados e armazenados.

O Método feito é o padrão para navegadores, foi usado de primeiro caso o GET, para fazer uma listagem dos dados (por enquanto não retornada nada, por conta de que não foi feito o caso POST, onde é para inserir e criar os dados). Tudo isso como citado antes, usa de resposta o formato JSON para ser compreendido e listado os dados de forma facilitada.

Vantagens e desvantagens de usar arquivos JSON em vez de banco de dados

Usar arquivos JSON como armazenamento traz simplicidade, portabilidade e facilidade para protótipos e projetos pequenos, já que são fáceis de criar, ler e editar manualmente, sem depender de servidores externos. No entanto, apresentam limitações em escalabilidade, controle transacional, concorrência, desempenho em consultas complexas e segurança. Quando integrados a um banco SQL, os arquivos JSON geralmente são usados como um formato intermediário para importar ou exportar dados, facilitar a troca entre sistemas ou armazenar dados semi-estruturados dentro de colunas específicas do banco. Nesse cenário, o banco SQL oferece robustez, controle de acesso, integridade e otimização de consultas, enquanto o JSON agrega flexibilidade para lidar com dados menos estruturados, combinando o melhor dos dois mundos para aplicações que precisam de escalabilidade e flexibilidade simultaneamente.