

Service Web

Comidas

Documentação da API

Guilherme Amorim



Conexão com API

Consumindo e Decodificando JSON em PHP

```
Pokeapi

1 <?php
2
3 /* CONSUMO DE API - Utilização de um serviço web */
4
5 $url = "http://localhost/servico_web/consumo_de_json/apiGet.php?
   comida=Parmegiana+de+Frango&info=ingredientes";
6
7 // guardar um valor de resposta da API
8 $resposta = file_get_contents($url);
9
10 echo $resposta;
11
12 // convertendo JSON para um Array associativo
13 $valores = json_decode($resposta, true);
14
15
16
17 ?>
```

A primeira linha de código é realizar a conexão com o a API que tem como o endereço exclusivo do recurso (dado ou funcionalidade) que você deseja acessar ou manipular.

A segunda linha do código armazena todo o conteúdo da resposta (em formato JSON) na variável \$resposta como uma única string.

A terceira linha ela pega o texto da resposta (\$resposta), que geralmente está em formato JSON (string), e o converte em uma estrutura de dados que o PHP pode manipular (um array associativo, graças ao true).



API - Comidas

Criação da API de Comidas

Estrutura Base e Roteamento HTTP (Headers e Switch)

```
1  /* Estruturando uma API */
2
3  // Cabeçalho da API
4  header("Content-Type: application/json; charset=UTF-8");
5  header("Access-Control-Allow-Origin: *");
6
7
8  $metodo = $_SERVER['REQUEST_METHOD'];
9
10 switch ($metodo){
11
12     case "GET":
13         metodoGET();
14         break;
15
16     case "POST":
17         break;
18
19     default:
20         echo "Método usado não foi identificado";
21 }
```

- Este é o novo ponto de partida, onde a API se prepara para receber a requisição e decide qual função chamar com base no método HTTP.
- Headers: Mantêm a definição de resposta em JSON (Content-Type) e as permissões de acesso (Access-Control-Allow-Origin: *).
- `$metodo = $_SERVER['REQUEST_METHOD'];`: Captura o método HTTP usado na requisição (ex: GET, POST, PUT, DELETE).
- `switch ($metodo):`
- Roteamento: Direciona o fluxo do código com base no método HTTP.
- `case "GET": metodoGET();`: Se for uma consulta (GET), chama a função dedicada à leitura de dados.
- `case "POST":`: O bloco POST está preparado, mas ainda vazio.
- `default:`: Retorna uma mensagem de erro se o método usado não for reconhecido (ex: um método não implementado).

API - Comidas

Criação da API de Comidas

Função metodoGET(): Início e Parâmetros

```
ComidasAPI

1 // Sistema do Serviço Web
2 function metodoGET() {
3     // Leitura do arquivo JSON (comida.json) e armazenando e transformando em Array
    na variável
4     $dados_comidas = json_decode(file_get_contents("comida.json"), true);
5
6     $comida_especifica = $_GET['comida'] ?? null;
7     $info_solicitada = $_GET['info'] ?? null;
8
9     // ... continua para a lógica de saída
```

```
ComidasAPI

1 // Saída da API
2 if ($comida_especifica && isset($dados_comidas['comidas'][$comida_especifica]))
3 {
4     $comida = $dados_comidas['comidas'][$comida_especifica];
5
6     // ... continua para o switch de informações
```

- Este é o início da função responsável por todas as consultas e leituras de dados.
- function metodoGET() { ... }: Encapsula toda a lógica de leitura de dados.
- Carregamento de Dados: Lê e decodifica o arquivo comida.json para o array \$dados_comidas (mesma lógica do código anterior).
- Captura de Parâmetros: Continua a capturar os parâmetros da URL:
- \$comida_especifica: Qual prato está sendo consultado (ex: ?comida=Lasanha).
- \$info_solicitada: Qual campo específico se deseja (ex: &info=ingredientes).

Este segundo trecho da função é responsável por todas as consultas e leituras de dados.

PHP

- if (\$comida_especifica && isset(...)): Validação dupla.
 1. Verifica se o parâmetro comida foi enviado.
 2. Verifica se a comida solicitada existe no array de dados.
- \$comida = ...: Se a validação for positiva, armazena os dados do prato específico em \$comida. Isso torna o código mais limpo para o próximo passo.

API - Comidas

Criação da API de Comidas

Função metodoGET(): Roteamento Específico de Campos

```
1 switch($info_solicitada) {
2     case "nome":
3     case "tipo":
4     case "ingredientes":
5         // ... outros casos
6
7     case "origem":
8         $resposta_origem = [
9             'Origem' => $comida['Origem'] ?? 'Não especificado',
10        ];
11        echo json_encode($resposta_origem, JSON_UNESCAPED_UNICODE);
12        break;
13
14    case "tudo":
15    default:
16        echo json_encode($comida, JSON_UNESCAPED_UNICODE);
17        break;
18 }
```

- O bloco switch detalha o que será retornado com base no parâmetro info.
- switch(\$info_solicitada): Roteamento interno, tratando os campos nome, tipo, ingredientes, origem e nutricao.
- Tratamento de Campos: Para campos como nome e tipo, há uma checagem (isset) e um tratamento de erro se o campo estiver faltando.
- case "origem": Exemplo de retorno onde o campo é encapsulado e, se não existir, retorna 'Não especificado' (usando ??).
- default: (e case "tudo"): Se nenhum campo for especificado ou o campo for "tudo", retorna o objeto JSON completo da comida.
- JSON_UNESCAPED_UNICODE: Flag importante para garantir que caracteres especiais (acentos, cedilha) sejam exibidos corretamente no JSON de saída.

API - Comidas

Criação da API de Comidas

Função metodoGET(): Tratamento de Erro e Saída Padrão

```
ComidasAPI

1 } else {
2     if ($comida_especifica) {
3         echo json_encode(['erro' => "Comida '{$comida_especifica}' não
4         encontrada."], JSON_UNESCAPED_UNICODE);
5     } else {
6         echo json_encode($dados_comidas, JSON_UNESCAPED_UNICODE);
7     }
8 }
```

- O bloco else que lida com requisições inválidas ou com a listagem geral.
- Bloco else: Executado quando a condição do if principal falha.
- Erro (if (\$comida_especifica)): Se o usuário enviou o parâmetro comida, mas o valor não corresponde a nenhum dado, retorna uma mensagem de erro clara.
- Saída Padrão (else): Se o usuário não enviou o parâmetro comida, retorna todo o array de dados (\$dados_comidas), ou seja, a lista completa das comidas.

API - Comidas

Criação da API de Comidas

Funções Auxiliares de Gerenciamento (cadastrar_comida e salvar_dados)

```
ComidasAPI

1 function cadastrar_comida($nome, $tipo, $ingredientes, $origem, $nutrientes){
2     // Carrega dados atuais, adiciona a nova comida e salva
3     $dados_comidas = json_decode(file_get_contents("comida.json"), true);
4     // ... atribuição dos novos campos ...
5
6     if(false){
7         salvar_dados($dados_comidas);
8     }
9 }
10
11 function salvar_dados($variavel){
12     file_put_contents('comida.json', json_encode($variavel, JSON_PRETTY_PRINT |
13     JSON_UNESCAPED_UNICODE));
14 }
```

- Funções destinadas a futura implementação do método POST, permitindo adicionar e salvar novos dados no arquivo JSON.
- `cadastrar_comida()`: Função que simula a lógica de adicionar um novo item. Nota-se que ela precisa carregar o JSON internamente para garantir que o novo dado seja adicionado ao conteúdo atual.
- `if(false){ salvar_dados($dados_comidas); }`: A chamada para salvar os dados está desativada. Isso é comum para garantir que a API em modo de leitura (GET) não altere o arquivo de dados.
- `salvar_dados()`: Função responsável por converter o array de volta para JSON e escrevê-lo no arquivo `comida.json`.
- `JSON_PRETTY_PRINT`: Formata o JSON no arquivo com quebras de linha e indentação para facilitar a leitura humana.

Json da API

```
ComidasAPI

1 {
2   "comidas": {
3     "Parmegiana de Frango":{
4       "Nome":"Parmegiana de Franco",
5       "Tipo":"Salgada",
6       "Ingredientes": [
7         "Filé de peito de frango",
8         "farinha de trigo",
9         // ... mais ingredientes ...
10      ],
11      "Origem":"Italiana",
12      "Nutricao": [
13        "Proteínas",
14        "Carboidratos",
15        // ... mais nutrição ...
16      ]
17    },
18    "Pizza Marguerita": {
19      // ... estrutura similar ...
20    }
21    // ... e outras 4 comidas
22  }
23 }
```

Explicação

- O que esta estrutura representa:
- Raiz da API ("comidas": {}):
 - O objeto principal que agrupa todas as informações.
 - Quando a API é consultada sem parâmetros, ela retorna este objeto completo.
- Chave de Acesso (Ex: "Parmegiana de Frango": {}):
 - Este nome é a chave de consulta utilizada pela API. É o valor que o usuário deve passar no parâmetro ?comida= na URL.
 - Campos de Dados Roteáveis:
 - Cada comida possui campos específicos que podem ser consultados individualmente:
 - "Nome" e "Origem": Campos simples de texto.
 - "Ingredientes" e "Nutricao": São arrays (listas) de dados, permitindo a consulta detalhada via &info=....
- Conexão com o Código PHP:
 - O PHP lê esta estrutura, e o bloco switch dentro da função metodoGET() é responsável por buscar e retornar exatamente o campo que o cliente solicitou (ex: apenas a lista de "Ingredientes").

Como Utilizar a API de Comida

Ação	URL de Exemplo	Resultado
1. Listagem Geral	sua_api.php	Retorna TODOS os dados de comidas ("comidas": {...})
2. Consulta Completa	sua_api.php?comida=Lasanha	Retorna o objeto completo (todos os campos) da "Lasanha".
3. Campo Específico	sua_api.php?comida=Feijoada&info=origem	Retorna apenas o campo "Origem" da Feijoada.
4. Detalhe em Lista	sua_api.php?comida=Pizza Marguerita&info=ingredientes	Retorna apenas a lista de "Ingredientes" da Pizza.
5. Erro (Não Encontrado)	sua_api.php?comida=Salada	Retorna um objeto de erro em JSON ({"erro": "Comida '...' não encontrada."}).

Como Utilizar a API de Comida

Parâmetro	Descrição	Exemplo
Comida	Nome da Comida	pizza, sushi
Info	Tipo de informação desejada	nome, tipo, ingredientes, origem, nutricao, tudp

Visualizar toda as comidas

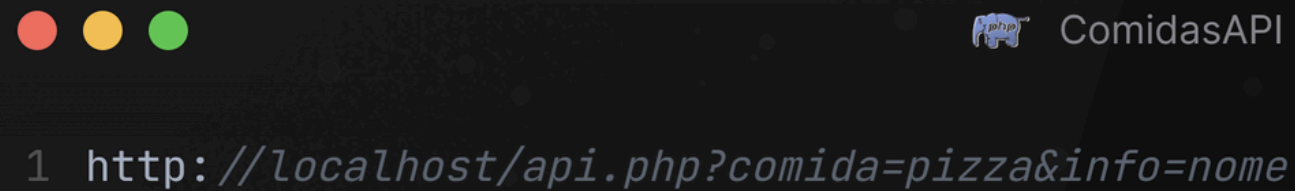
```
ComidasAPI  
1 http://localhost/api.php
```

Ver apenas uma comida específica

```
ComidasAPI  
1 http://localhost/api.php?comida=pizza
```

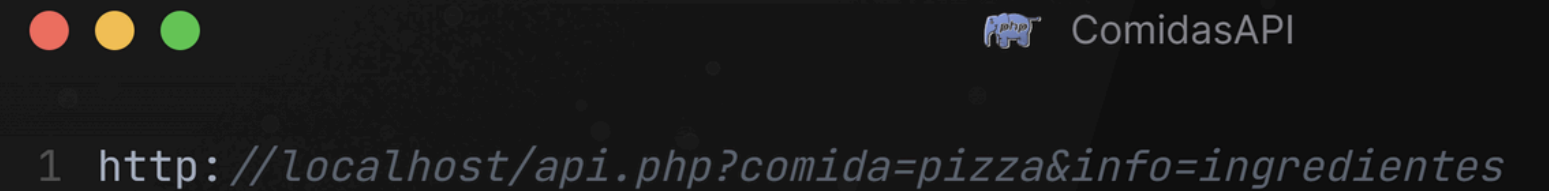
Como Utilizar a API de Comida

Ver somente o nome

A terminal window with a dark background. At the top left are three colored circles (red, yellow, green). At the top right is a small icon of a person with a speech bubble and the text "ComidasAPI".

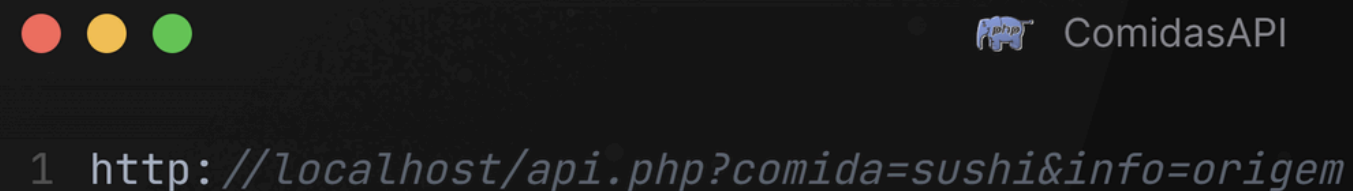
```
1 http://localhost/api.php?comida=pizza&info=nome
```

Ver ingredientes

A terminal window with a dark background. At the top left are three colored circles (red, yellow, green). At the top right is a small icon of a person with a speech bubble and the text "ComidasAPI".

```
1 http://localhost/api.php?comida=pizza&info=ingredientes
```

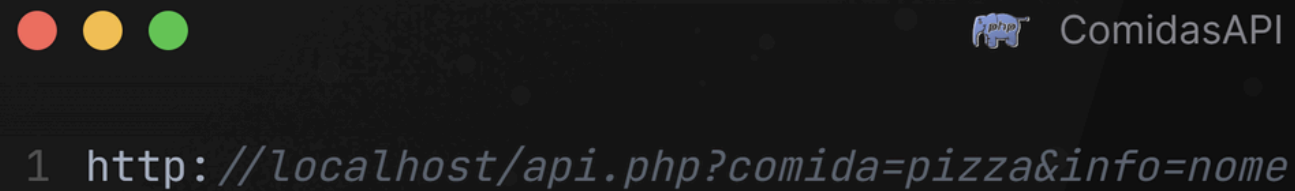
Ver origem

A terminal window with a dark background. At the top left are three colored circles (red, yellow, green). At the top right is a small icon of a person with a speech bubble and the text "ComidasAPI".

```
1 http://localhost/api.php?comida=sushi&info=origem
```

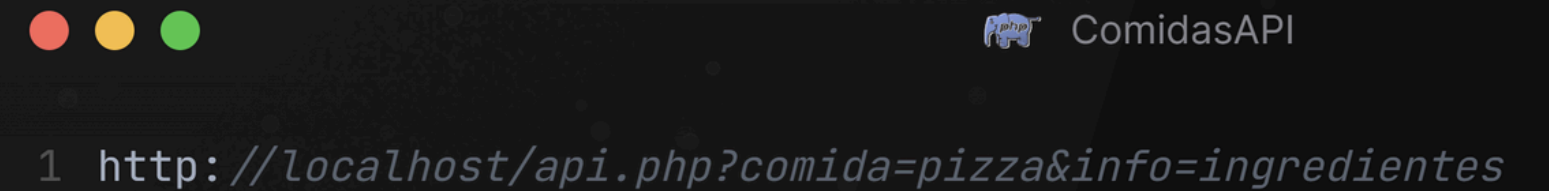
Como Utilizar a API de Comida

Ver somente o nome

A terminal window with a dark background. At the top left are three colored circles (red, yellow, green). At the top right is a small icon of a person with a speech bubble and the text "ComidasAPI".

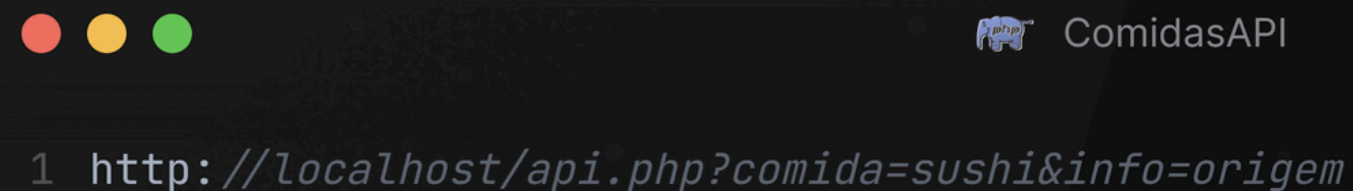
```
1 http://localhost/api.php?comida=pizza&info=nome
```

Ver ingredientes

A terminal window with a dark background. At the top left are three colored circles (red, yellow, green). At the top right is a small icon of a person with a speech bubble and the text "ComidasAPI".

```
1 http://localhost/api.php?comida=pizza&info=ingredientes
```

Ver origem

A terminal window with a dark background. At the top left are three colored circles (red, yellow, green). At the top right is a small icon of a person with a speech bubble and the text "ComidasAPI".

```
1 http://localhost/api.php?comida=sushi&info=origem
```

Como Utilizar a API de Comida

O que voce quer fazer	URL/ Ação
Ver todas as comidas	api.php
Ver uma comida específica	api.php?comida=pizza
Ver só o nome	api.php?comida=pizza&info=nome
Ver os ingredientes	api.php?comida=pizza&info=ingredientes
Ver a origem	api.php?comida=pizza&info=origem
Ver os dados nutricionais	api.php?comida=pizza&info=nutricao

URL da API

http://localhost/servico_web/consumo_de_json/api.php?comida=Parmegiana+de+Frango&info=ingredientes



Obrigado(a) por sua participação e interesse.

A estruturação dos dados (JSON) e o código PHP demonstram a base para
a nossa solução

Se houver sugestões ou perguntas sobre a implementação, este é o
momento!

[GITHUB](#)