



escola
britânica de
artes criativas
& tecnologia

Engenheiro de Qualidade de Software

Testes de API e Contrato

O que é API?

API

API - Application Programming Interface

Interface de Programação de Aplicações



Tipos de API

APIs públicas ou abertas

APIs disponíveis para desenvolvedores e outros usuários de forma pública para o consumo e com restrições mínimas. Elas podem exigir registro, uso de uma chave de API ou OAuth, ou mesmo podem ser completamente abertas.

Ex. Google Maps, Correios

APIs privadas ou internas

APIs ocultadas de usuários externos e expostas apenas para sistemas internos de uma empresa. Elas não são usadas para o consumo fora da empresa, mas sim ao uso em diferentes equipes de desenvolvimento interno para melhor produtividade e reutilização de serviços.

APIs de parceiros

As APIs de parceiros são APIs expostas por ou para os parceiros de negócios estratégicos. Eles não estão disponíveis publicamente e precisam de direitos específicos para acessá-los.

APIs compostas

As APIs compostas combinam vários dados ou APIs de serviço. Eles são desenvolvidos usando os recursos de orquestração de API de uma ferramenta de criação de API. Permitem que os desenvolvedores acessem vários terminais em uma chamada.

Protocolos de API

Um protocolo fornece regras definidas para chamadas de API. Ele especifica os tipos de dados e comandos aceitos.

Os principais tipos de protocolos para APIs são **SOAP**, **RPC** e **REST**.

SOAP

- SOAP significa protocolo de acesso a objeto simples, e é um protocolo bem estabelecido semelhante ao REST no sentido de que é um tipo de API da Web;
- O SOAP foi aproveitado desde o final dos anos 1990 e foi o primeiro a padronizar a maneira como os aplicativos devem usar conexões de rede para gerenciar serviços.

RPC

- RPC é um protocolo de chamada procedural remota. Eles são os tipos mais antigos e simples de APIs. O objetivo de um RPC era que o cliente executasse o código em um servidor;
- APIs RPC são fortemente acopladas, então isso torna difícil mantê-las ou atualizá-las.

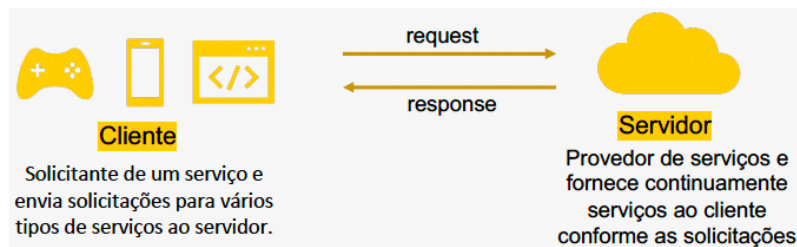
Protocolos de API

REST

Representational State Transfer e é uma API de serviços da web;

- A arquitetura REST é simples e fornece acesso aos recursos para que o cliente REST acesse e reenderize os recursos no lado do cliente. No estilo REST, URI ou IDs globais ajudam a identificar cada recurso;

- Esta arquitetura usa várias representações de recursos para representar seu tipo, como XML, JSON, Texto, Imagens e assim por diante.



Restfull

Uma API que segue todos os princípios de arquitetura é chamada de RESTful.

- **Stateless** - A API REST é stateless por natureza, onde o servidor não precisa saber em qual estado o cliente está e vice-versa
- **Interface uniforme** - um cliente e um servidor devem se comunicar um com o outro via HTTP (protocolo de transferência de hipertexto) usando URIs (identificadores de recursos exclusivos), CRUD (criar, ler, atualizar, excluir) e convenções JSON (JavaScriptObject Notation).
- **Cliente-servidor** - o cliente e o servidor devem ser independentes um do outro. As alterações feitas no servidor não devem afetar o cliente e vice-versa.
- **Cache** - o cliente deve armazenar em cache as respostas, pois isso melhora a experiência do usuário, tornando-as mais rápidas e eficientes.
- **Em camadas** - a API deve oferecer suporte a uma arquitetura em camadas, com cada camada contribuindo para uma

HTTP Status Code

Cada resposta que a aplicação REST retorna, é enviado um código definindo o status da requisição:



HTTP Status Code

Exemplos de 2xx: Sucesso.

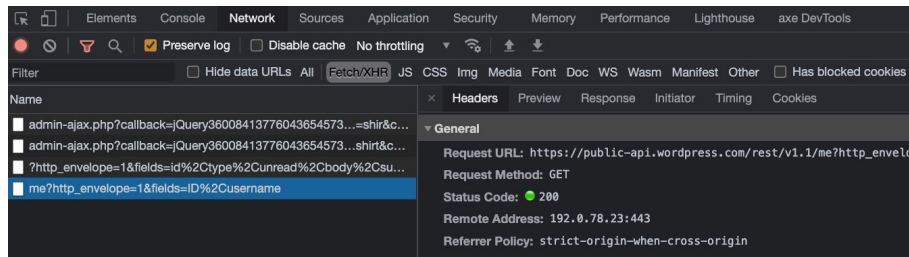
A requisição foi recebida com sucesso, entendida e aceita.

200 OK – Resposta padrão de sucesso

201 Created – Indica que um recurso foi criado

202 Accepted – Indica que a requisição foi aceita

204 No content – Indica que a requisição foi processada com sucesso, mas que não há conteúdo para retornar



HTTP Status Code

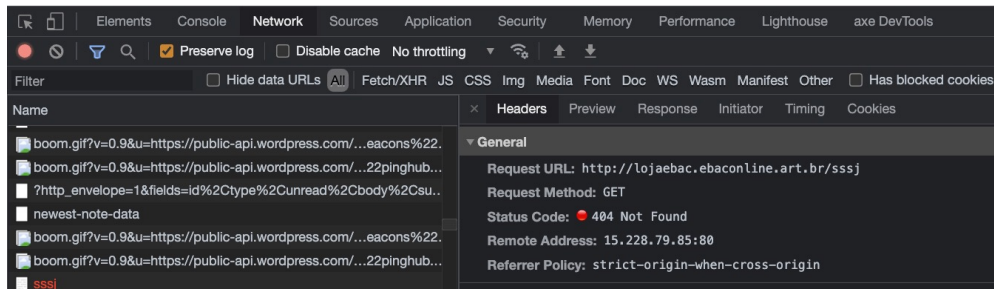
Exemplos de **4xx: Erro do cliente**. Ação que ocorre um erro do cliente/servidor.

400 Bad Request – A requisição com problemas
(Ex: mudanças de contrato)

401 Unauthorized - Cliente não autenticado ou sem autorização

403 Forbidden – Cliente autenticado, mas sem permissão

404 Not found – Recurso não localizado

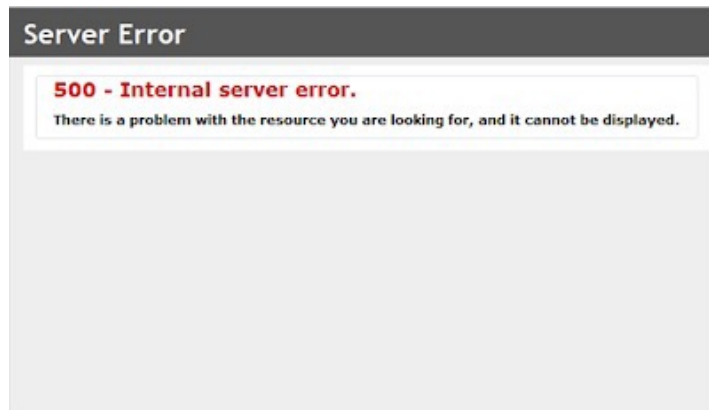


HTTP Status Code

Exemplos de **5xx: Erro do servidor: Ação que ocorre um erro do servidor**

500 Internal Server Error – Um erro ocorreu durante o processamento da requisição (erro no processamento)

503 Service Unavailable – Serviço indisponível (por manutenção ou sobrecarga)



Métodos HTTP

O protocolo HTTP (**Hypertext Transfer Protocol**) define um conjunto de métodos de requisição responsáveis por indicar a ação a ser executada para um dado recurso;

Também são chamados de **Verbos HTTP**;

Os principais são **GET, POST, PUT, DELETE**;

Outros métodos **HEAD, CONNECT, OPTIONS, TRACE, PATCH**.

Método GET

O método GET solicita a representação de um recurso específico. Requisições utilizando o método GET devem retornar apenas dados.



Ingrid Running Jacket

☆☆☆☆☆ (0 customer reviews) | 26 sold

R\$84,00

This is a variable product called a Ingrid Running Jacket

Size

XS S M L XL

Color

Orange Red White

- 1 +

COMPRAR

♥ WISHLIST

🔄 COMPARAR

SKU: WJ04

Category: Performance Fabrics|Clothing

Método **POST**

O método POST é utilizado para submeter uma entidade a um recurso específico, frequentemente causando uma mudança no estado do recurso ou efeitos colaterais no servidor.

Cadastro

Email address *

ebac.qualidade@gmail.com

Password *

.....

Forte

Seus dados pessoais serão usados para apoiar sua experiência em todo este site, para gerenciar o acesso à sua conta e para outros fins descritos em nossa política de privacidade.






ENVIAR

Método PUT

O método PUT substitui todas as atuais representações do recurso de destino pela carga de dados da requisição.






Dados atuais

DETALHES DA CONTA

 PAINEL	First name *	Last name *
 PEDIDOS	<input type="text" value="Fábio"/>	<input type="text" value="Araújo"/>
 DOWNLOADS	Display name *	
 ENDEREÇOS	<input type="text" value="Fábio Araújo"/>	
 DETALHES DA CONTA	<small>This will be how your name will be displayed in the account section and in reviews</small>	
	Email address *	
	<input type="text" value="email@email.com"/>	







Dados novos

DETALHES DA CONTA

 PAINEL	First name *	Last name *
 PEDIDOS	<input type="text" value="Fábio"/>	<input type="text" value="Araújo"/>
 DOWNLOADS	Display name *	
 ENDEREÇOS	<input type="text" value="faraújo"/>	
 DETALHES DA CONTA	<small>This will be how your name will be displayed in the account section and in reviews</small>	
	Email address *	
	<input type="text" value="email_novo@email.com"/>	

Método DELETE

O método DELETE remove um recurso específico.

	Product Name	Unit Price	Stock Status	Add to cart	Remove
	Stellar Solar Jacket	R\$75,00	In Stock	 Ver Opções	 Remove this product
	Augusta Pullover Jacket	R\$57,00	In Stock	 Ver Opções	

Contratos e documentação

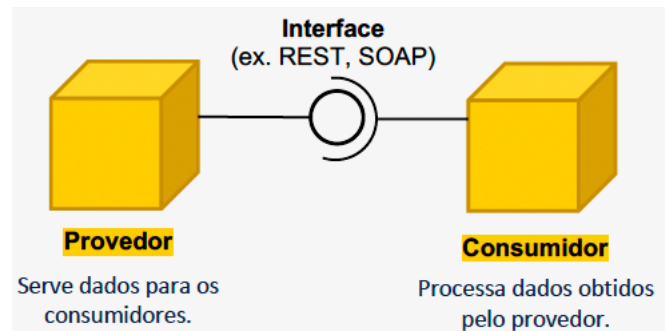
Interfaces

Dividir seu sistema em muitos serviços pequenos, geralmente significa que esses serviços precisam se comunicar uns com os outros;

Eles se comunicam através de certas interfaces de diferentes aplicativos, formas e tecnologias.

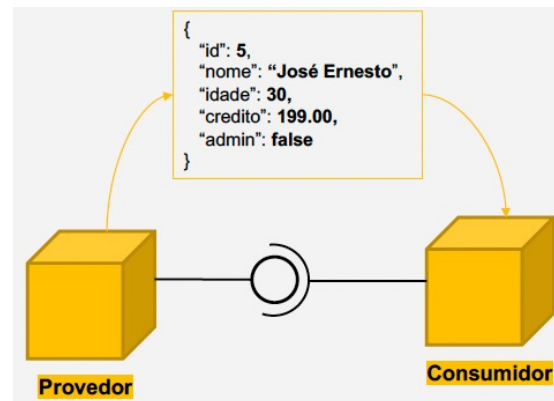
Os mais comuns são:

- REST e JSON via HTTPS
- RPC usando algo como gRPC
- Comunicações via SOAP/XML



Contratos

Como pode ser enviado informações para diferentes Consumers, há a necessidade de especificar claramente a interface entre esses serviços (o chamado **contrato**) afim de evitar problemas de integração.



Contratos

```
{  
  "id": 5,  
  "nome": "José Ernesto",  
  "idade": 30,  
  "credito": 199.00,  
  "admin": false  
}
```

Atributo	Tipo	Exemplo
id	int	5
nome	string	José Ernesto
idade	int	30
credito	decimal / float	199.00
admin	boolean	false

**Onde encontrar
essas informações?**

Documentação

ENVIRONMENT **No Environment** LAYOUT **Double Column** LANGUAGE **cURL** ⚙️

POSTMAN API
 Introduction
 Overview
 Authentication
 Rate Limits
 Support
 Terms of Use
 API Reference

Collections
 GET All Collections
 GET Single Collection
 POST Create Collection
 PUT Update Collection
 DEL Delete Collection
 POST Create a Fork
 POST Merge a Fork
 Environments
 Mocks
 Monitors
 Workspaces
 User
 Import
 API

GET All Collections

`https://api.getpostman.com/collections`

The `/collections` endpoint returns a list of all **collections** that are accessible by you. The list includes your own collections and the collections that you have subscribed to.

The response contains an array of collection information containing the `name`, `id`, `owner` and `uid` of each collection.

Requires **API Key** as `X-API-Key` request header or `apikey` URL query parameter.

HEADERS

X-API-Key

GET Single Collection

`https://api.getpostman.com/collections/{collection_uid}`

Access the contents of a collection that is accessible to you using its unique ID (`uid`).

Requires **API Key** as `X-API-Key` request header or `apikey` URL query parameter.

Example Request

Valid Response

```
curl --location --request GET 'https://api.getpostman.com/collections' \
  --header 'X-API-Key:'
```

Example Response

200 OK

Body

Header (16)

```
{
  "collections": [
    {
      "id": "dac5eac9-148d-a32e-b76b-3edee9da28f7",
      "name": "Cloud API",
      "owner": "631643",
      "uid": "631643-dac5eac9-148d-a32e-b76b-3edee9da28f7"
    },
    {
      "id": "f2e66c2e-5297-e405-155c-200090900e3",

```

View More

Example Request

Valid Response

```
curl --location -g --request GET 'https://api.getpostman.com/collection' \
  --header 'X-API-Key:'
```

Example Response

200 OK


Body

Header (16)

```
{
  "collection": {
    "variables": [],
    "info": {
      "name": "Sample Collection"
    }
  }
}
```

<https://documenter.postman.com/view/631643/JsLs/?version=latest#3190c896-4216-a0a3-aa38-a041d0c2eb72>

Documentação


Swagger
Supported by SMARTBEAR

Explore

EBAC - Shop API

[Base URL: lojaebac.ebaonline.art.br/wp-json]
<http://lojaebac.ebaonline.art.br/rest-api/schema>

Página de teste

[Contact the developer](#)

Schemes
 HTTP

Authorize

coupons

GET /wc/v3/coupons

POST /wc/v3/coupons

GET /wc/v3/coupons/{id}

POST /wc/v3/coupons/{id}

PUT /wc/v3/coupons/{id}

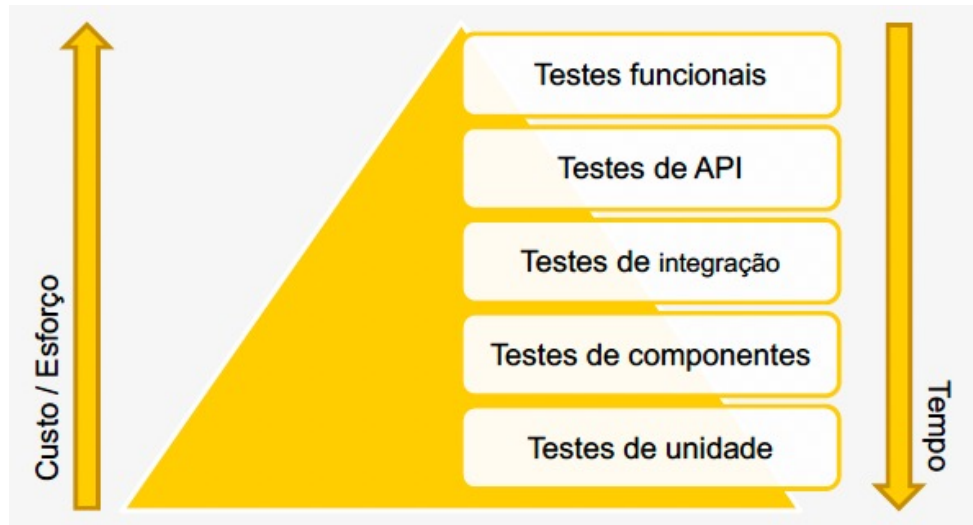
PATCH /wc/v3/coupons/{id}

DELETE /wc/v3/coupons/{id}

http://lojaebac.ebaonline.art.br/rest-api/docs/#/customers/delete_wc_v3_customers_id

Estratégia de testes de API

Estratégia de testes



Fluxo de testes



O que testar?

Segue uma lista de possíveis testes na API:

- 1. Status:** o código de resposta está adequado (2xx, 3xx, 4xx e 5xx);
- 2. Performance:** a resposta retornou dentro do tempo adequado;
- 3. Syntaxe:** o tipo de conteúdo retornado está adequado (Content-Type) / o servidor aceita requisições no formato adequado;
- 4. Tratamento de erro:** o servidor rejeita requisições no formato inadequado / excluir campos obrigatórios deve resultar em erro / requisições com tipos de dados inadequados deve resultar em erro;
- 5. Detecção de erros:** testes negativos para identificar exceções;
- 6. Schema:** o conteúdo da resposta está de acordo com a estrutura ou formato esperado (contrato);
- 7. Funcional:** o servidor retorna o valor previsto de acordo com a requisição / a requisição insere, atualiza ou exclui um recurso esperado;
- 8. Segurança:** Injeções de SQL não impactam na integridade dos dados.

JSON

JSON (JavaScript Object Notation - Notação de Objetos JavaScript) é uma formatação leve de troca de dados.

Para seres humanos, é fácil de ler e escrever. Para máquinas, é fácil de interpretar e gerar. JSON é em formato texto e completamente independente de linguagem ideal para troca de dados.

Vantagens:

- Leitura mais simples
- Analisador(parsing) mais fácil
- JSON suporta objetos
- Performance na execução e transporte de dados
- Arquivo com tamanho reduzido

Objeto

Um objeto é um conjunto desordenado de pares nome/valor.

Um objeto começa com { chave de abertura e termina com chave de fechamento }.

Cada nome é seguido por : dois pontos e os pares nome/valor são seguidos por , vírgula.

```
{  
  "id": 123,  
  "nome": "Fábio",  
  "email": "fabio@ebac.com.br"  
}
```

Array

Uma array é uma coleção de valores ordenados.

O array começa com `[` colchete de abertura e termina com colchete `]` de fechamento.

Os valores são separados por `,` vírgula.

```
[1,2,3,5,8,13,21]
```

```
["banana", "maçã", "melancia", "uva"]
```

```
[  
  {  
    "usuario": "admin",  
    "senha": "psw!123"  
  },  
  {  
    "usuario": "user",  
    "senha": "psw!321"  
  }  
]
```

Valor

Um valor pode ser uma:

- cadeia de caracteres (string)
- um número
- true ou false
- null
- objeto
- array

```
{  
  "id": 123,  
  "nome": "Fábio",  
  "email": "fabio@ebac.com.br",  
  "admin": "true"  
}
```

Testes de API com Postman

Postman

O **Postman** é uma ferramenta que proporciona criar, compartilhar, testar e documentar APIs. Permite aos usuários criar e salvar solicitações HTTP e HTTPS simples e complexas, bem como ler suas respostas.

<https://www.postman.com/downloads/>



ServeRest

- Pré-requisito: Node JS
- Inserir o seguinte comando no console:
`npx serverest`
- Para acessar basta abrir seu navegador com a seguinte url:
<http://localhost:3000/> ou <http://127.0.0.1:3000/>

Obs.: Enquanto estiver usando, deixe o console aberto.

*Caso tenha problema com a instalação, use o endereço de produção:
<https://serverest.dev/>, mas corre o risco de alguém mexer nos seus
dados, pois é um ambiente compartilhado.*

Referências:

- <https://pt.slideshare.net/elias.nogueira/de-a-mxima-cobertura-nos-seus-testes-de-api>
- <https://martinfowler.com/articles/microservice-testing/#anatomy-connections>
- <https://pokeapi.co/>
- <https://swapi.dev/>
- <https://petstore.swagger.io/#/>
- <https://www.json.org/json-pt.html>
- <https://serverest.dev/>
- http://lojaebac.ebaonline.art.br/restapi/docs/#/customers/delete_wc_v3_customers_id
- <https://documenter.postman.com/view/631643/JsLs/?version=latest#99810ef3-3cc0-a6cc-06f5-d8e2ae9d84e4>
- <https://learning.postman.com/docs/getting-started/introduction/>
- <https://github.com/public-apis/public-apis>
- <https://www.linkapi.solutions/blog/quais-sao-os-tipos-de-apis>
- <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>