Curso
Qualidade de software

Qualidade de Software

Módulo 13

escola britânica de artes criativas & tecnologia

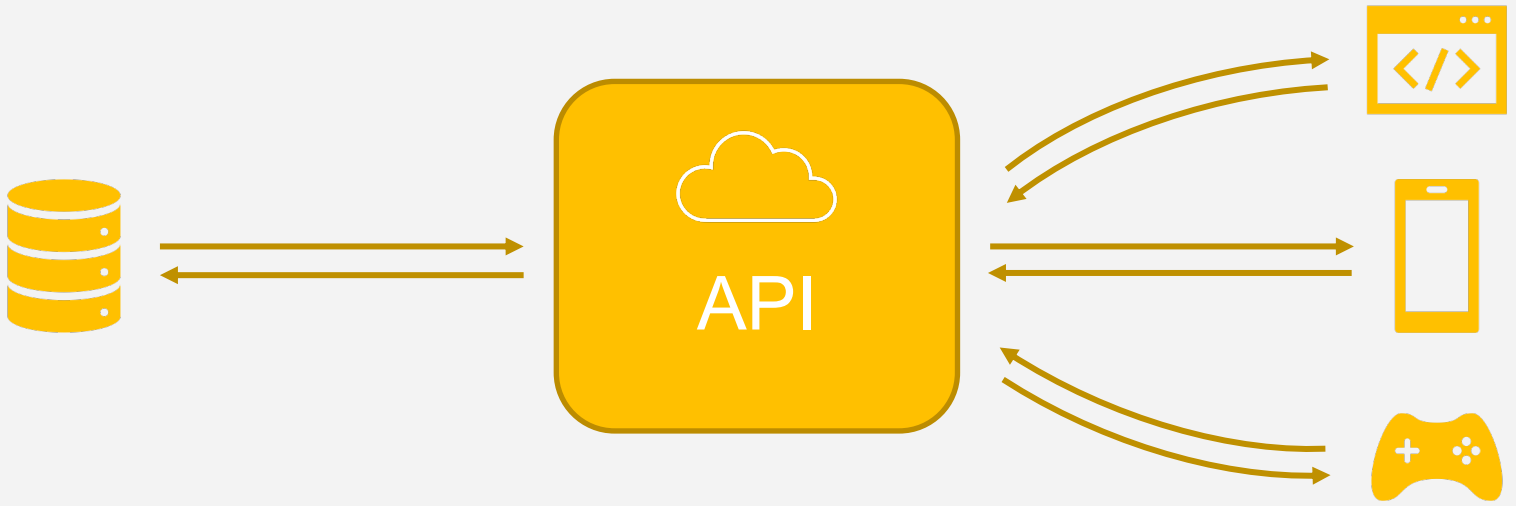
O que é API?

Aula 1

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 abertos.
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.

https://www.linkapi.solutions/blog/quais-sao-os-tipos-de-apis

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

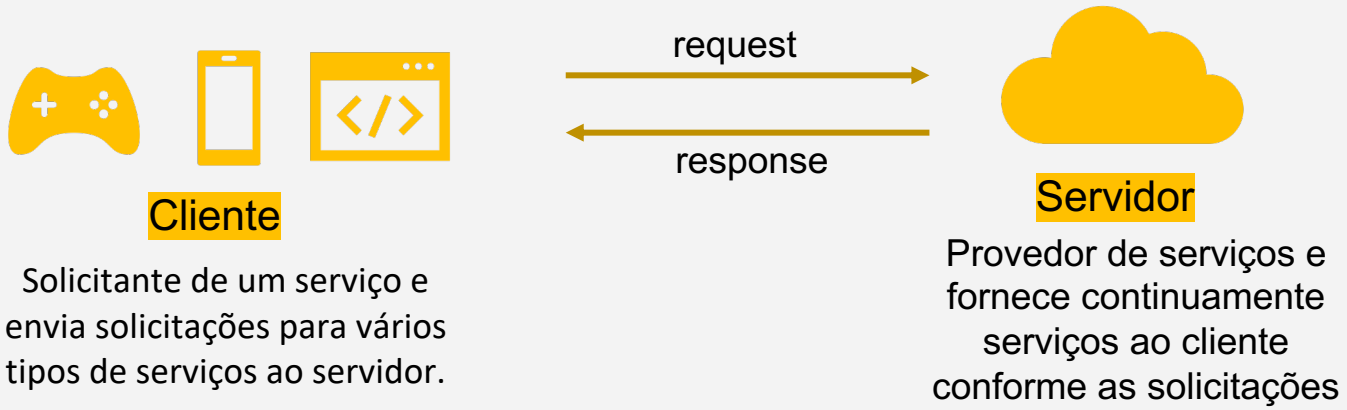
- SOAP significa protocolo de acesso a objeto simples, e é um protocolo bem estabelecido semelhante ao REST no sentido de que é um tipo de API da Web;
- O SOAP foi aproveitado desde o final dos anos 1990 e foi o primeiro a padronizar a maneira como os aplicativos devem usar conexões de rede para gerenciar 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** : **Re**presentational **S**tate **T**ransfer 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;
- Está arquitetura usa várias representações de recursos para representar seu tipo, como XML, JSON, Texto, Imagens e assim por diante.



RESTful

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** (JavaScript Object 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

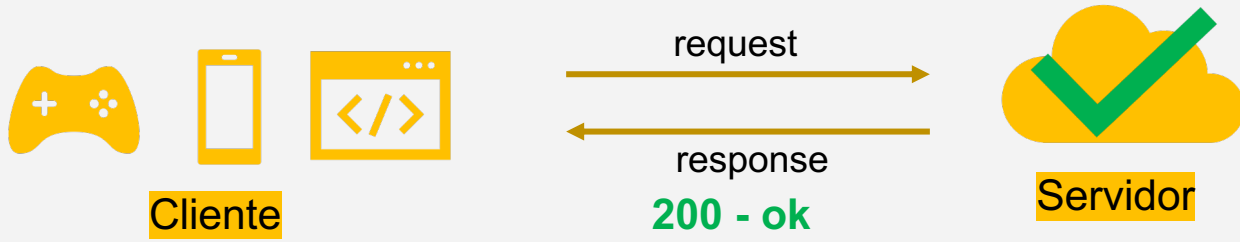
HTTP Status Code

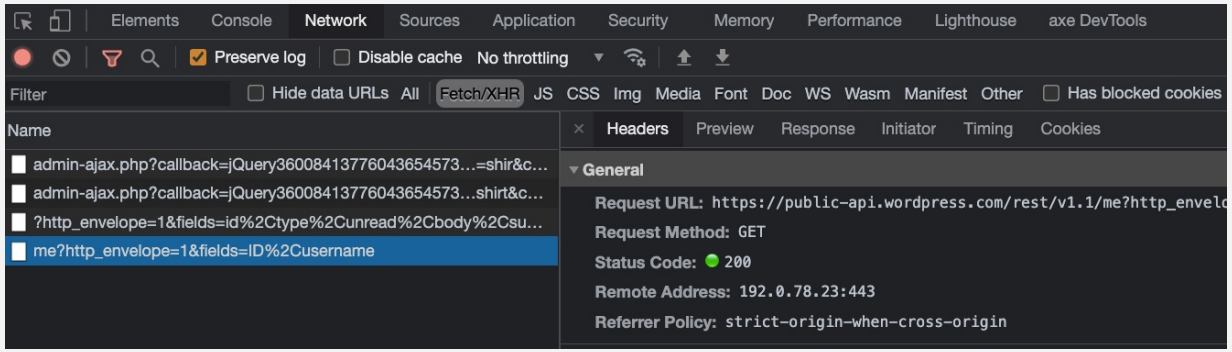
- Cada resposta que a aplicação REST retorna, é enviado um código definindo o status da requisição:

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 contente – Indica que a requisição foi processada com sucesso, mas que não há conteúdo para retornar

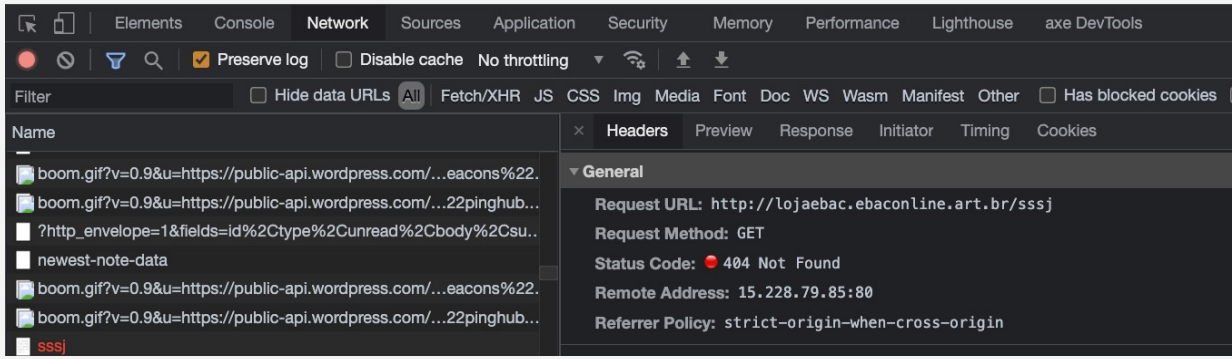




HTTP Status Code

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

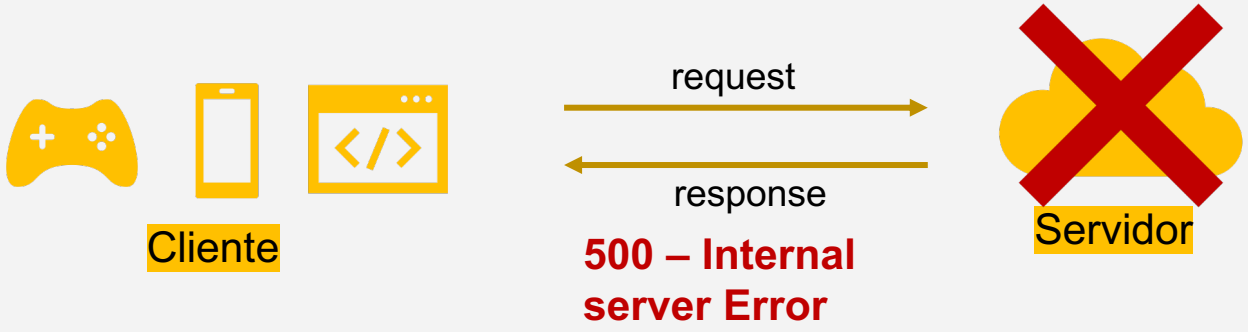
- 400 Bad Request – A requisição com problemas (Ex: mudanças de contrato)
- 401 Unauthorized - Cliente não autenticado ou sem autorização
- 403 Forbidden – Cliente autenticado, mas sem permissão
- 404 Not found – Recurso não localizado

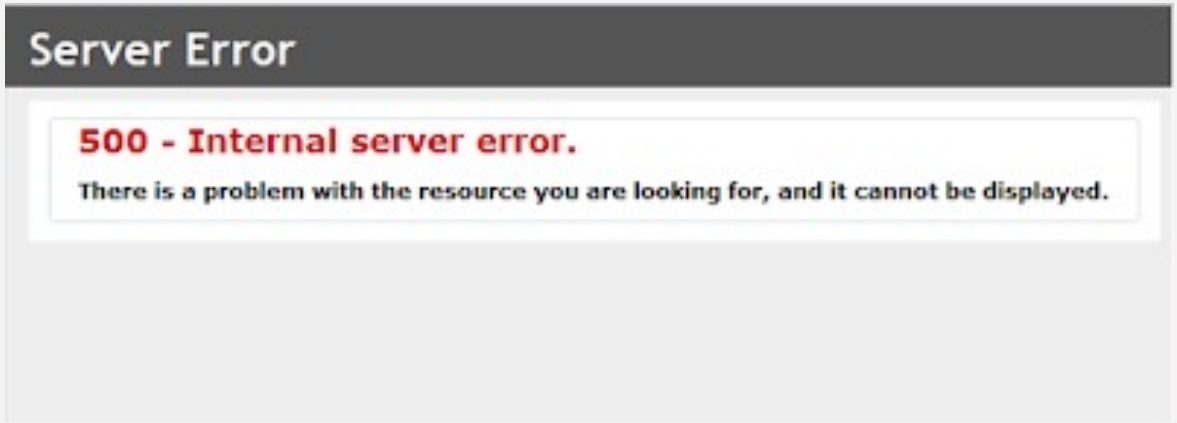


HTTP Status Code

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)



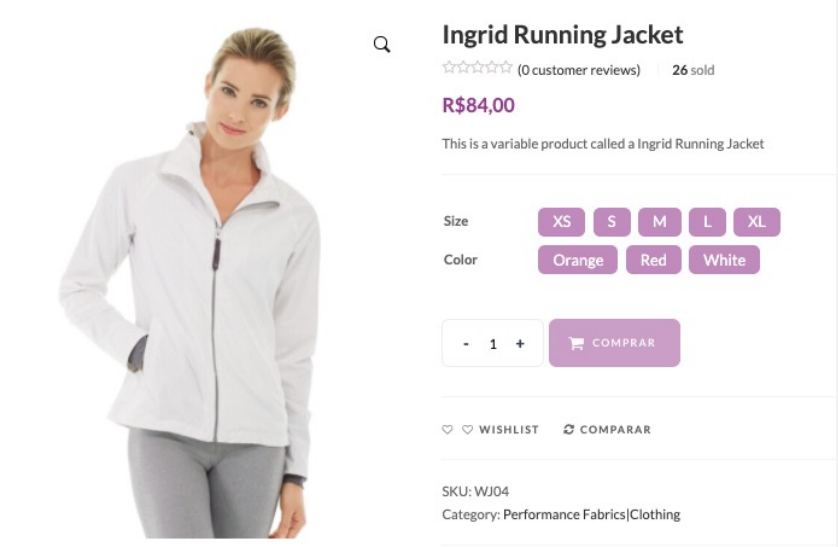


Métodos HTTP

- 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.

GET

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



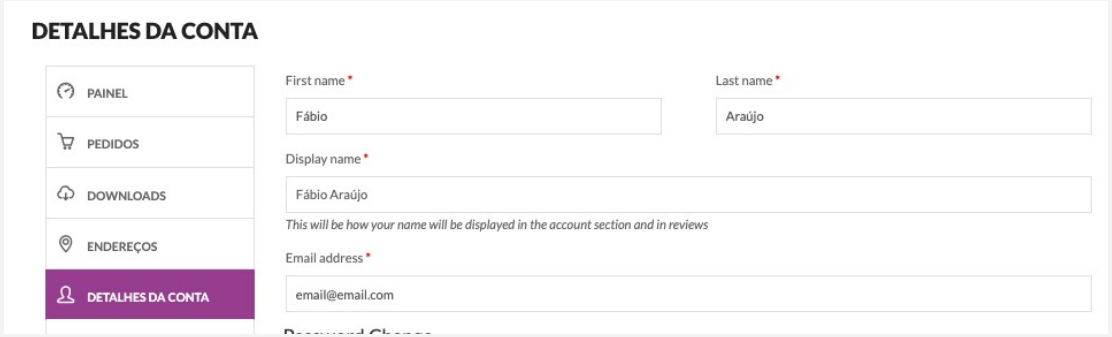
POST

- 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.

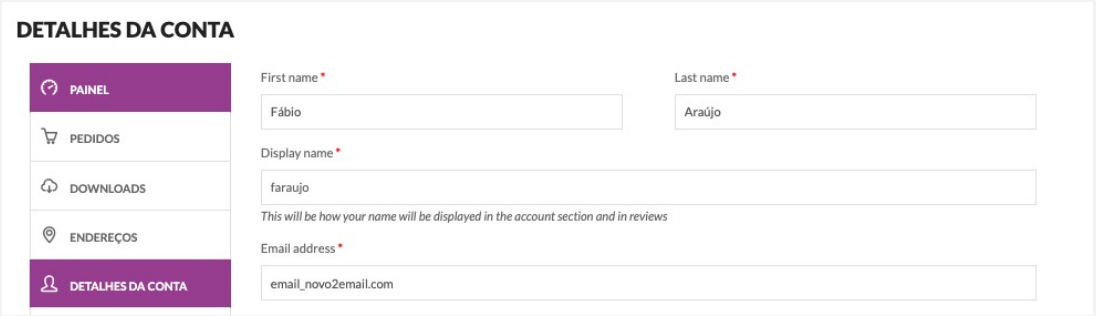
PUT

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

Dados atuais

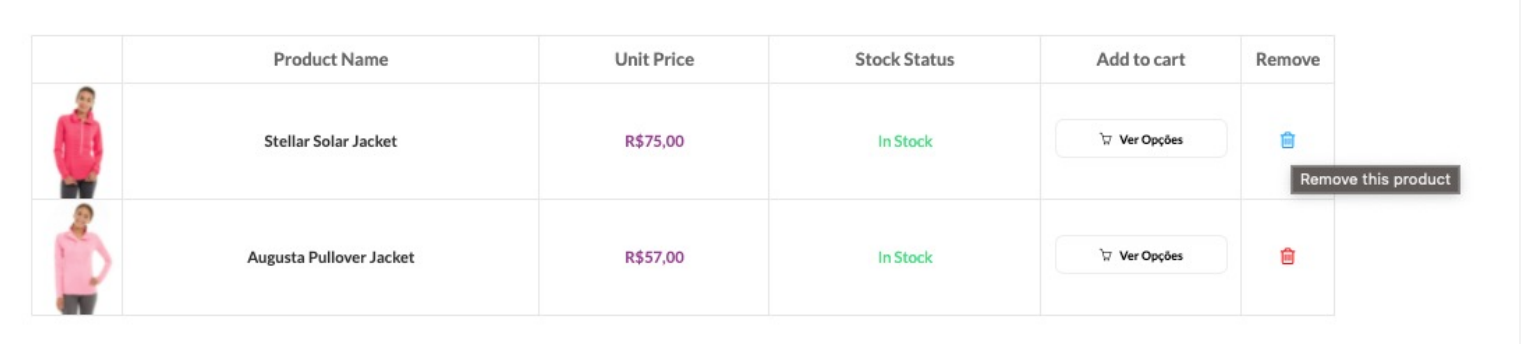


Dados novos



DELETE

- O método **DELETE** remove um recurso específico.

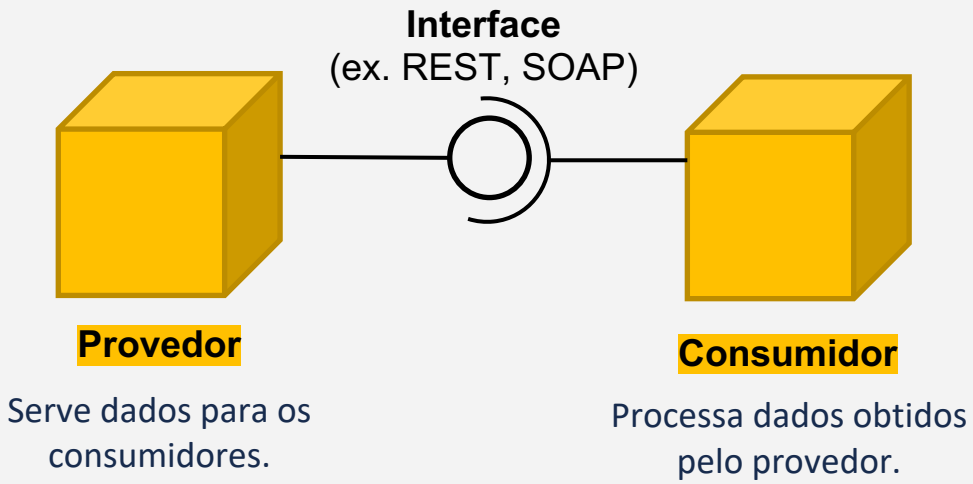


Contratos e documentação

Aula 2

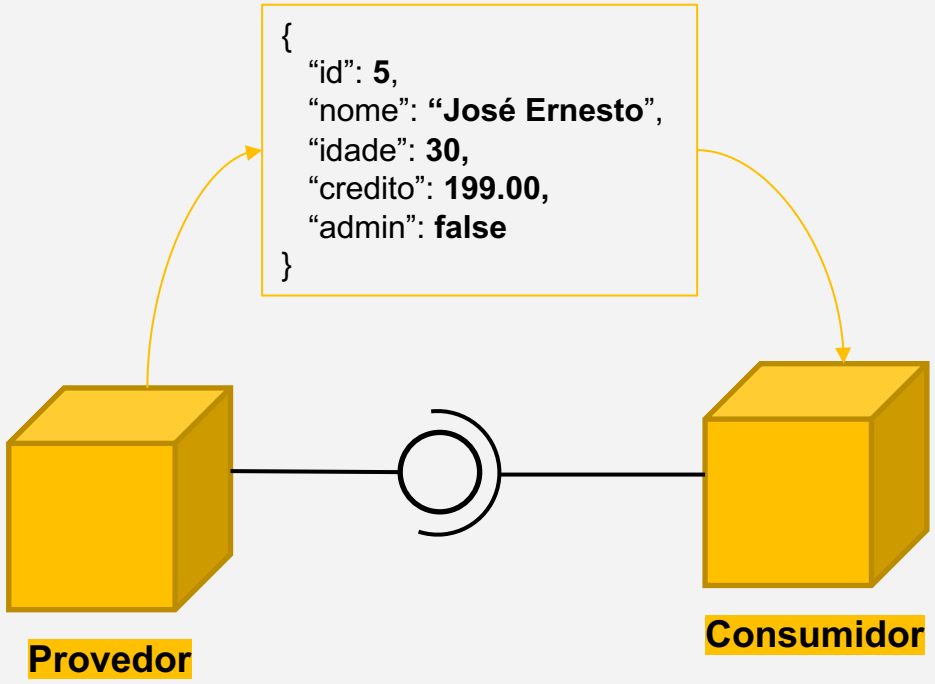
Interfaces

- 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.



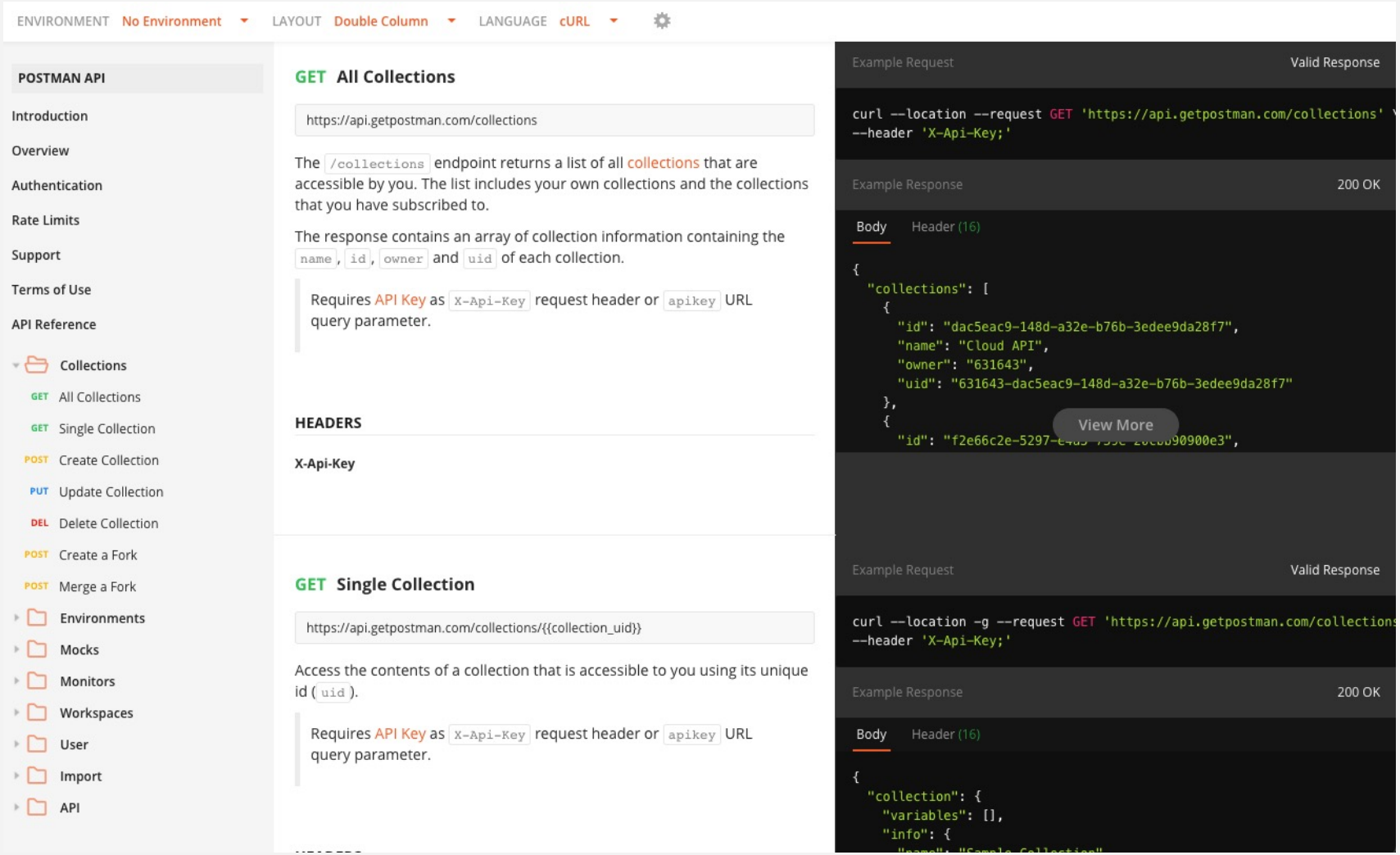
Contratos

```
{
  “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

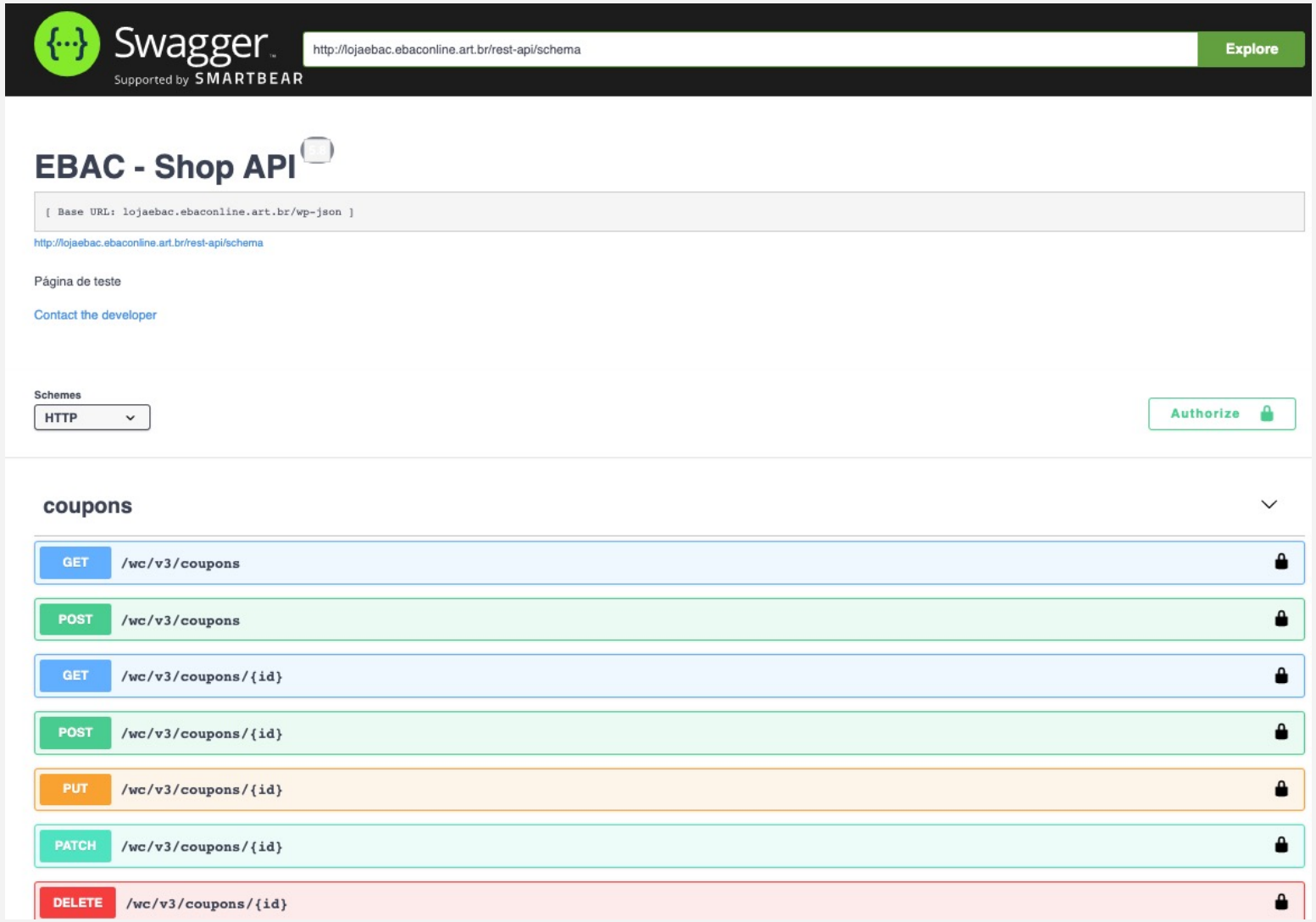
Onde encontrar essas informações?

Documentação



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

Documentação

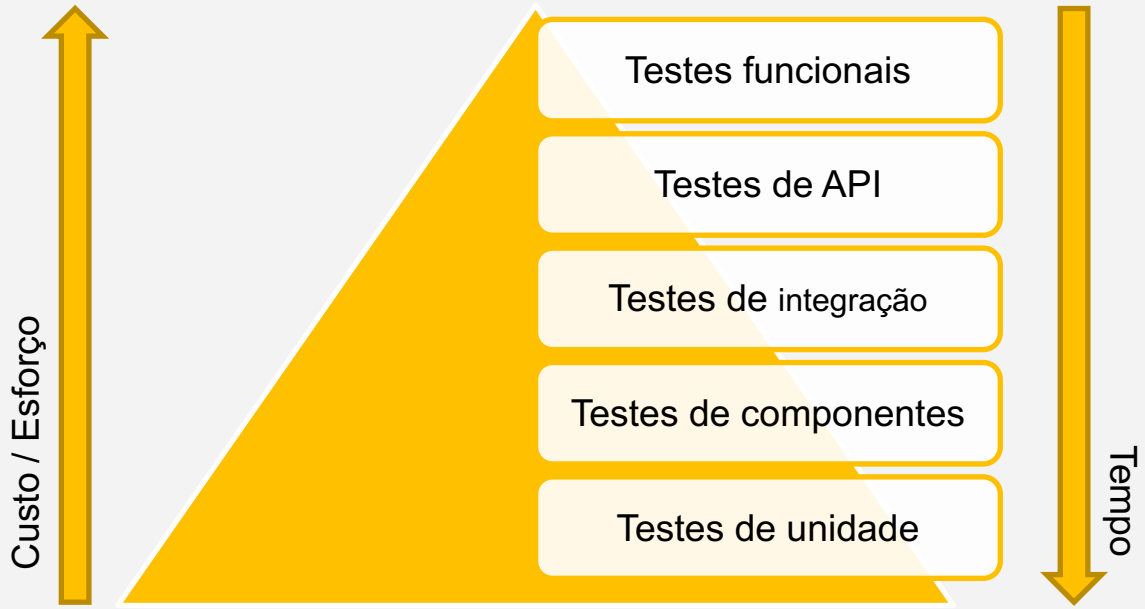


http://lojaebac.ebaconline.art.br/rest-api/docs/#/customers/delete_wc_v3_customers__id_

Estratégia de testes de API

Aula 3

Estratégia de testes



Fluxo de testes

Health Check

Garantir que o endpoint está respondendo

Contrato

Garantir que o endpoint não teve seus atributos alterados

Aceitação

Garantir que o endpoint funciona ou apresenta os resultados de falha esperados

Funcional

Garantir que um conjunto de endpoints funcionam como na UI

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

Aula 4

Postman

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

- 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.

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.ebaconline.art.br/rest-api/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