

# VEM SER

## **Módulo 4 - Testes de API REST**

### **Aula 1**



DBC

# Aula 1: Introdução ao Teste de API REST



# Aula 1 - Conteúdo Programático

## 1.1 Conceitos Básicos

1.1.1 O que é API?

1.1.2 Protocolos de API

1.1.3 Métodos HTTP

1.1.4 Status Code

1.1.5 Recursos e Parâmetros

1.1.6 Autenticação e Autorização

## 1.2 Contratos e Documentação

1.2.1 Contratos

1.2.2 Documentação

## 1.3 Estratégia de Testes

1.3.1 Fluxo de Testes- Pipeline

1.3.2 O que testar?

1.3.3 Entendendo o JSON

# 1.1 Conceitos Básicos



# O que vem a sua mente quando ouve a palavra API?



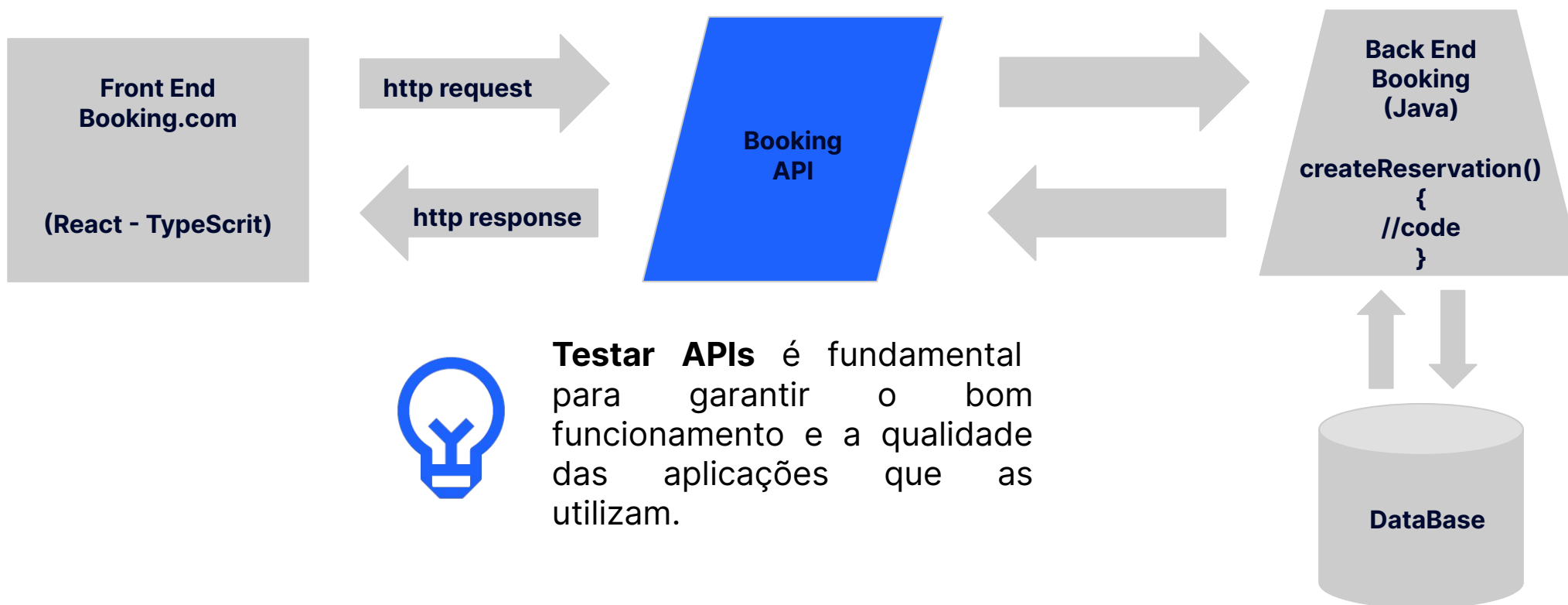
[https://www.menti.com/  
albbv26tdok6](https://www.menti.com/albbv26tdok6)





## 1.1.1 O que é API?

Acrônimo para *Application Programming Interface*, ou, em português, **Interface de Programação de Aplicação**.



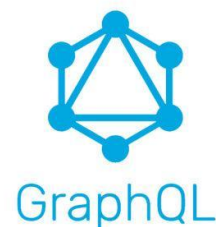


## 1.1.2 Protocolos de API

As APIs operam **protocolos (regras) ou convenções** na hora de oferecer serviços. Os principais são: **REST, SOAP e GraphQL**.



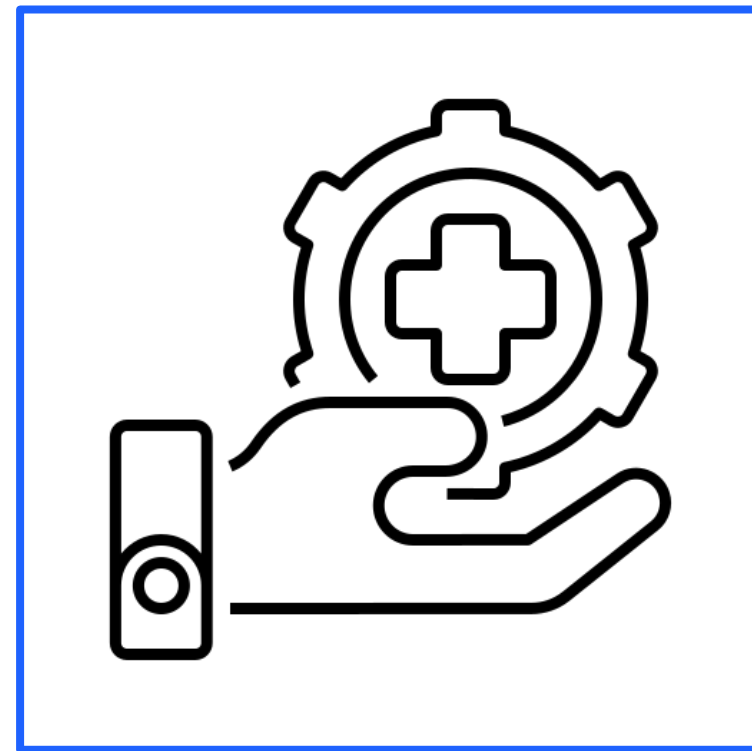
**SOAP** (*Simple Object Access Protocol*): Significa **Protocolo de Acesso a Objetos Simples** e é um protocolo baseado em **XML** para a comunicação. É menos flexível e mais difícil de implementar.



**GraphQL**: Tecnologia mais recente. Foi desenvolvida pelo Facebook e permite que os clientes solicitem exatamente os dados que precisam, evitando receber diversos dados e efetuar várias requisições ao servidor.

## 1.1.2 Protocolos de API

**REST** (**RE**presentational **S**tate **T**ransfer): Significa transferência de estado representacional. Não é um protocolo, mas sim um estilo arquitetural, onde são definidos um conjunto de **princípios e restrições** para o design de APIs.



## 1.1.2 Protocolos de API

### Princípios ou Restrições REST



- **Interface uniforme:** Estrutura consistente e previsível, com métodos HTTP (GET, POST, PUT, DELETE), convenções de uso de URIs (identificadores de recursos exclusivos), representações de recursos (JSON ou XML) e hiperlinks.
- **Stateless:** O servidor não mantém informações de estado do cliente, ou seja, cada solicitação do cliente deve conter todas as informações para que o servidor entenda e processe a solicitação.

## 1.1.2 Protocolos de API

### Princípios ou Restrições REST



- **Cliente-servidor:** Separação entre cliente e servidor, que permite que eles evoluam independentemente um do outro.
- **Sistema em camadas:** Arquitetura em camadas distintas, com cada uma delas desempenhando um papel específico (separação de responsabilidades).
- **Cache:** Uso de cache para melhorar o desempenho e a eficiência da comunicação entre cliente e servidor.

## 1.1.3 Métodos HTTP

Os **métodos HTTP** são responsáveis por especificar a **ação** que deve ser executada para um recurso. Também conhecidos como **Verbos HTTP**;

Os **mais utilizados** são: **GET, POST, PUT e DELETE**;

Operações de **CRUD**.

Create	→	POST
Read	→	GET
Update	→	PUT
Delete	→	DELETE

## 1.1.3 Métodos HTTP

### GET

O método **GET** é utilizado para solicitar a representação de um recurso específico do servidor.



Samsung galaxy s6

\$360 \*includes tax

**Product description**

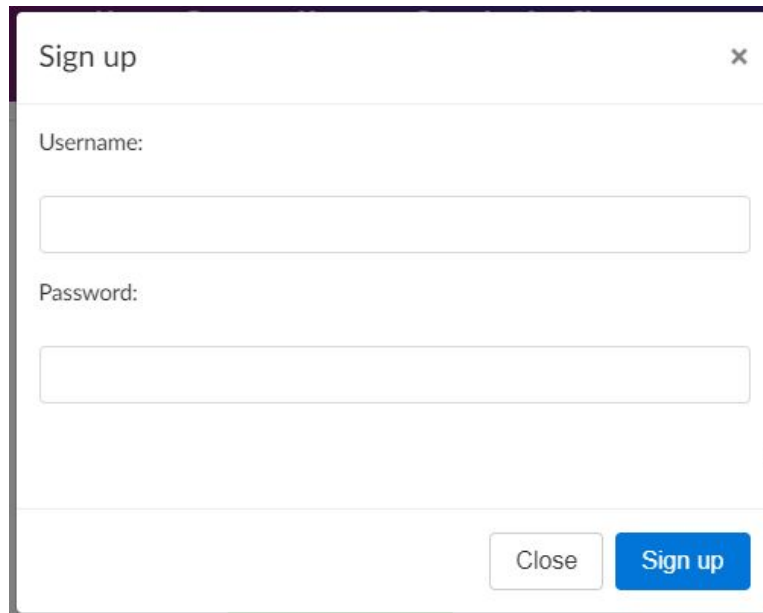
The Samsung Galaxy S6 is powered by 1.5GHz octa-core Samsung Exynos 7420 processor and it comes with 3GB of RAM. The phone packs 32GB of internal storage cannot be expanded.

Add to cart

## 1.1.3 Métodos HTTP

### POST

O método **POST** é utilizado para enviar dados ao servidor, geralmente para criar um novo recurso.



A screenshot of a web form titled "Sign up" with a close button (x) in the top right corner. The form contains two input fields: "Username:" and "Password:". Below the "Password:" field, there is a large empty space. At the bottom right of the form, there are two buttons: "Close" and "Sign up".

Sign up

Username:


Password:

Close Sign up


## 1.1.3 Métodos HTTP

### PUT

O método **PUT** é utilizado para atualizar um recurso existente no servidor.

 Your Info

Ályson QA  
 Rua Teste  
 Recife  
 Brazil


 Update



## 1.1.3 Métodos HTTP

### DELETE

O método **DELETE** é utilizado para remover um recurso específico do servidor.

Pic	Title	Price	x
	Samsung galaxy s6	360	<a href="#">Delete</a>

## 1.1.4 Status Code

### O que é?

O código de status HTTP indica para o cliente qual **a condição** atual sobre o processamento de sua requisição, ou seja, **se foi bem-sucedida ou não**.



## 1.1.4 Status Code

### 2xx: Sucesso

**200 OK:** A solicitação foi bem sucedida.

**201 Created:** A solicitação foi bem sucedida e um novo recurso foi criado. Esta é uma típica resposta enviada após uma requisição **POST**.

**204 No Content:** Indica que a solicitação foi bem sucedida e que não há conteúdo para retornar. É comumente usado em respostas a requisições **DELETE**.

## 1.1.4 Status Code

### 4xx: Erro cliente

**400 Bad Request:** Indica que a solicitação do cliente é inválida ou mal formatada.

**401 Unauthorized:** Indica que a solicitação requer autenticação e o cliente não forneceu as credenciais corretas para acessar o recurso.

**403 Forbidden:** Indica que o servidor recebeu a requisição, mas se negou a autorizá-la, mesmo com autenticação.

**404 Not Found:** O servidor não encontrou uma representação atual do recurso solicitado. URL incorreto ou recurso removido ou inexistente.

## 1.1.4 Status Code

### 5xx: Erro servidor

**500 Internal Server Error:** Indica que ocorreu um erro interno no servidor. Usado quando o servidor encontra uma condição inesperada que o impede de atender completamente a requisição.

**503 Service Unavailable:** Indica que o servidor está temporariamente indisponível. Servidor sobrecarregado, em manutenção, etc.

# 1.1.5 Recursos, Parâmetros e Cabeçalhos

## Recursos

**Recursos** representam APIs/Collections que podem ser acessadas a partir do Servidor.

- <https://www.google.com/Images>
- <https://www.google.com/maps>
- <https://www.google.com/docs>

# 1.1.5 Recursos, Parâmetros e Cabeçalhos

## Path Parameters

**Path Parameter** é uma parte variável de um URL path. Usado normalmente para apontar para um recurso específico dentro de uma coleção, como um usuário identificado por ID.

- <https://www.google.com/Images/1123343>
- <https://www.ecos.com.br/usuarios/123>

## 1.1.5 Recursos, Parâmetros e Cabeçalhos

### Query Parameters

**Query Parameter** é usado para classificar/filtrar os recursos. São anexados ao final da URL, seguindo “?” e listados em pares **chave=valor**.

- [https://www.ecos.com.br/usuarios?data\\_nascimento=2/10/2000](https://www.ecos.com.br/usuarios?data_nascimento=2/10/2000)



## 1.1.5 Recursos, Parâmetros e Cabeçalhos

### Headers/Cookies

**Headers** representam os metadados associados à solicitação e resposta da API, enviando detalhes adicionais à API para processar nossa solicitação.

**Por exemplo:** Authorization

### URL

Base URL/resource/(Query/Path)Parameters

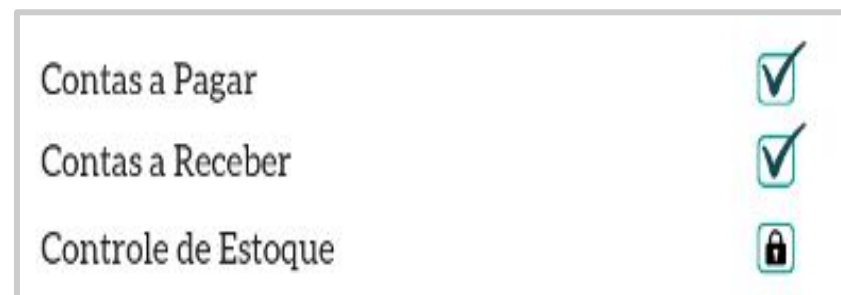
## 1.1.6 Autenticação e Autorização

**Autenticação:** Processo para provar que você é a pessoa que pretende ser. Apresentar credenciais.



A login form with two input fields. The top field is labeled 'USERNAME' and has a person icon to its left. The bottom field is labeled 'PASSWORD' and has a padlock icon to its left.

**Autorização:** Processo de dar acesso a alguém. Privilégios que são concedidos a determinado usuário ao utilizar uma aplicação.



A list of privileges with checkboxes to the right of each item. The first two items, 'Contas a Pagar' and 'Contas a Receber', have checked checkboxes. The third item, 'Controle de Estoque', has an unchecked checkbox.

Contas a Pagar	<input checked="" type="checkbox"/>
Contas a Receber	<input checked="" type="checkbox"/>
Controle de Estoque	<input type="checkbox"/>

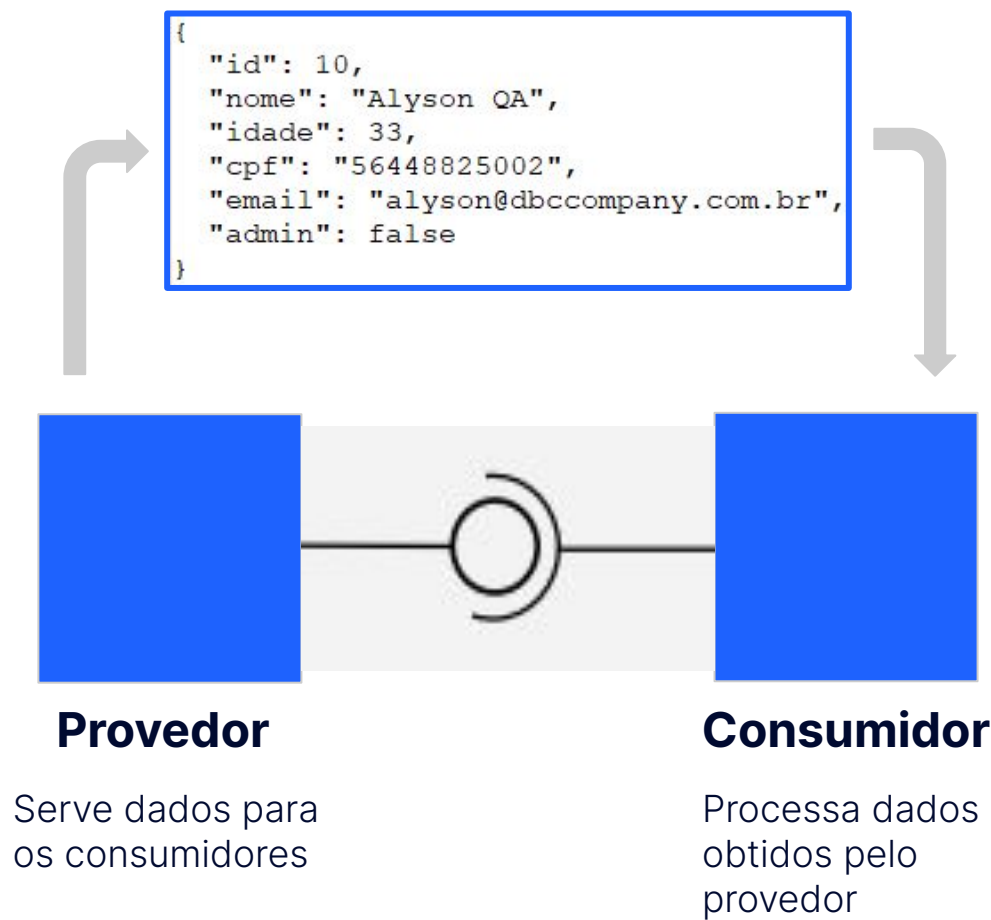
## 1.2 Contratos e Documentação



## 1.2.1 Contratos

### O que são Contratos?

Um **Contrato** é um documento que define como uma API deve ser usada e como os diferentes componentes devem interagir entre si, a fim de evitar problemas de integração.



## 1.2.1 Contratos

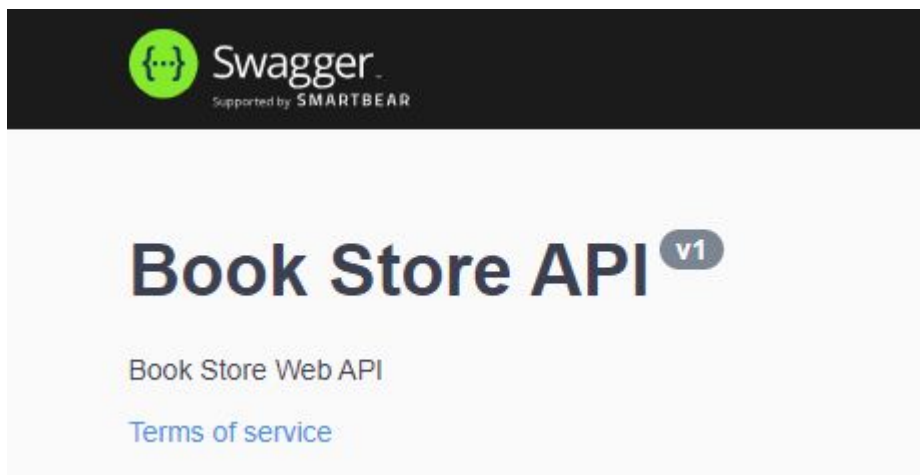
```
{  
  "id": 10,  
  "nome": "Alyson QA",  
  "idade": 33,  
  "cpf": "56448825002",  
  "email": "alyson@dbccompany.com.br",  
  "admin": false  
}
```

Atributo	Tipo	Exemplo
id	int	10
nome	string	Alyson QA
idade	int	33
cpf	string	56448825002
email	string	alyson@dbccompany.com.br
admin	boolean	false

## 1.2.2 Documentação

**Mas onde encontrar essas informações?**

Na documentação. E o formato mais conhecido para se documentar uma API é a **OpenAPI/Swagger**.



<https://bookstore.toolsqa.com/swagger/#/>

## 1.2.2 Documentação

### Exemplo - Documentação Incorreta

Produtos

GET /api/Produtos

POST /api/Produtos

Parameters Try it out

Name	Description
produto (body)	<div>Example Value : Model</div> <pre>{   "produtoId": 0,   "descricao": "string",   "unidadeMedida": 0,   "valorDeCusto": 0,   "margemDeLucro": 0,   "valorDeVenda": 0,   "dataCadastro": "2019-08-25T22:11:24.044Z",   "dataAlteracao": "2019-08-25T22:11:24.044Z",   "ativo": true }</pre> <div>Parameter content type</div> <div>application/json-patch+json</div>

Fonte: <https://marcionizzola.medium.com/implementando-o-uso-de-contratos-na-api-65658c529709>

## 1.3 Estratégia de Testes





## 1.3.1 Fluxo de testes - Pipeline

### 1º Health Check

Garantir que o endpoint está respondendo

### 2º Contrato

Garantir que o endpoint não teve seus atributos alterados

### 3º Funcional

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

### 4º Aceitação

Garantir que um conjunto de endpoints funcionam como na UI

## 1.3.2 O que testar?

Existem vários aspectos que podem ser testados em uma API para garantir seu bom funcionamento, **por exemplo:** status, performance, tratamento de erro, segurança, etc.

Para termos **um guia**, um roadmap para identificar o que testar vamos nos utilizar da **heurística VADER**.



## 1.3.2 O que testar?



**Verbs** (*Verbos*)

**Authorization** (*Autorização*)

**Data** (*Dados*)

**Errors** (*Erros*)

**Responsiveness** (*Cap. de resposta*)

## 1.3.2 O que testar?

### Verbos



Consiste em testar os métodos aptos e não aptos para o endpoint.

- GET
- POST
- PUT
- DELETE

## 1.3.2 O que testar?

### Autorização



- Validar Token (Tipo de criptografia e testes de segurança)
- Quais recursos a API Key deve acessar.
- Testes de Token, API Key ou Usuário e senha inválido ou inexistentes.
- Restrições de acesso assim que for autorizado.

## 1.3.2 O que testar?

**Dados**



Observamos os dados de uma requisição ou de uma resposta de um endpoint.

- Tipagem
- Paginação
- Formato (JSON/XML)
- Schema
- Syntaxe
- Tamanho

## 1.3.2 O que testar?

### Erros



Avaliamos o status code para cada erro e suas respectivas mensagens.

- Tratamento de erros
- Detecção de erros
- Mensagens

## 1.3.2 O que testar?

Importante estabelecer junto ao time padrões uniformes para os erros, incluindo mensagens, exceções e quando acontece cada erro. **Exemplo de tabela DE/PARA:**

Exceção/Condição	Código de Resposta HTTP	Mensagem de Erro/Retorno
Credenciais Corretas	200 OK	Login bem-sucedido.
Credenciais Incorretas	401 Unauthorized	Usuário ou senha incorretos.
Campo de Usuário Vazio	400 Bad Request	O campo de usuário é obrigatório.
Campo de Senha Vazio	400 Bad Request	O campo de senha é obrigatório.



## 1.3.2 O que testar?

### Capacidade de Resposta



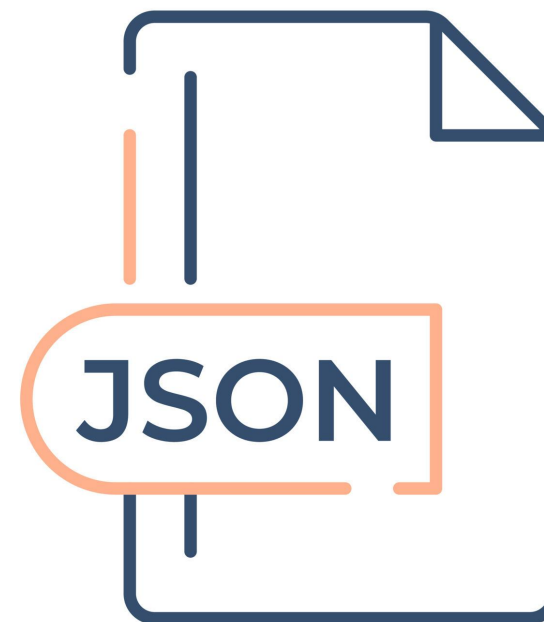
- Tempo de resposta e padrão para o projeto.
- Falha rápida
- Concorrência

## 1.3.4 Entendendo o JSON

### **JSON (Notação de Objetos JavaScript)**

é o formato de dados mais utilizado para o compartilhamento de informações entre sistemas.

Para testar uma API é importante saber ler um JSON.



## 1.3.4 Entendendo o JSON - Objeto



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

Objeto: { **nome : valor, nome : valor**}

```
{  
  "nome": "Ályson Campos",  
  "idade": 33,  
  "email": "alyson@dbcccompany.com.br"  
}
```

## 1.3.4 Entendendo o JSON - Array

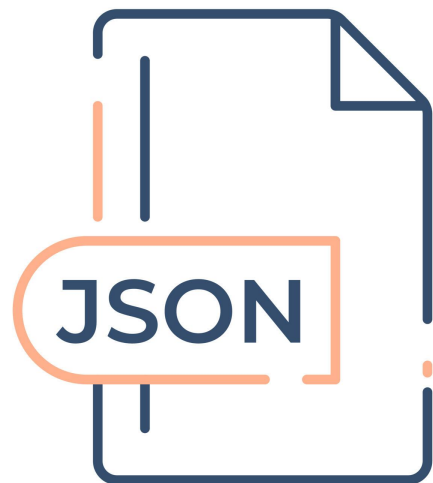
Uma **array** é uma coleção valores ordenados.

Array: **[valor1, valor2] - [objeto1, objeto2]**



```
[
  {
    "nome": "Ályson Campos",
    "email": "alyson@dbccompany.com.br"
  },
  {
    "nome": "Mailton Nascimento",
    "email": "mailton@dbccompany.com.br"
  }
]
```

## 1.3.4 Entendendo o JSON - Valor



Um **valor** pode ser: string, número, boolean, null, objeto, array.

```
{
  "idColaborador": 135,
  "nome": "Ályson Campos",
  "email": "alyson@dbccompany.com.br",
  "ativo": true
}
```

## 1.3.4 Entendendo o JSON - Prática



<https://jsoneditoronline.org/>

# Referências

# Referências

<https://www.toolsqa.com/rest-assured/what-is-rest/>

<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

<https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Objects/JSON>

<https://maximilianoalves.medium.com/vader-heuristica-para-teste-de-api-na-pratica-fcf78c6acec>

<https://qa-matters.com/2016/07/30/vader-a-rest-api-test-heuristic/>





Let's *Tech Up Together*