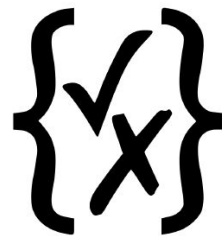


Validação de response utilizando **Json Schema**



Erick Ribeiro

<https://www.linkedin.com/in/erickgribeiro/>

O que é?

JSON





```
1 // objeto:
2 { "chave1": "valor1", "chave2": "valor2" }
3
4 // array:
5 [ "primeiro", "segundo", "terceiro" ]
6
7 // número:
8 42
9 3.1415926
10
11 // string:
12 "Esta é uma string"
13
14 // booleano:
15 true
16 false
17
18 // nulo:
19 null
```



```
1 {
2   "nome": "Erick Ribeiro",
3   "aniversario": "22 de fevereiro de 1996",
4   "endereco": "Rua Neuza Gonçalves de Moraes, 10, Divinópolis-MG, Brasil"
5 }
```



```
1 {
2   "primeiro_nome": "Erick",
3   "sobrenome": "Ribeiro",
4   "aniversario": "1996-02-22",
5   "endereco": {
6     "endereco_rua": "Rua Neuza Gonçalves de Moraes",
7     "numero": 10,
8     "cidade": "Divinópolis",
9     "estado": "MG",
10    "pais": "Brasil"
11  }
12 }
```



Testando uma API

JSON SCHEMA



```
1 {
2   "primeiro_nome": "Erick",
3   "sobrenome": "Ribeiro",
4   "aniversario": "1996-02-22",
5   "endereco": {
6     "endereco_rua": "Rua Neuza Gonçalves
7     "numero": 10,
8     "cidade": "Divinópolis",
9     "estado": "MG",
10    "pais": "Brasil"
11  }
12 }
```



```
1 {
2   "type": "object",
3   "properties": {
4     "primeiro_nome": { "type": "string" },
5     "sobrenome": { "type": "string" },
6     "aniversario": { "type": "string", "format": "date" },
7     "endereco": {
8       "type": "object",
9       "properties": {
10        "endereco_rua": { "type": "string" },
11        "numero": { "type": "number" },
12        "cidade": { "type": "string" },
13        "estado": { "type": "string" },
14        "pais": { "type": "string" }
15      }
16    }
17  }
18 }
```



Benefícios de utilizar o JSON Schema



Verificação de Integridade e Consistência dos Dados;

Em testes de API, precisamos checar se os dados que recebemos estão corretos e completos. O JSON Schema facilita isso, pois define como os dados devem ser, incluindo os tipos e formatos.



Validação Automática;

Usando JSON Schema, a gente consegue definir como os dados devem ser e validar automaticamente as respostas da API. Isso poupa tempo e trabalho, já que a gente não precisa criar testes manuais para cada parte dos dados. E aí, como a validação é automática, a gente consegue colocar isso no CI/CD e descobrir problemas mais rápido.



Documentação e Comunicação;

Por ser uma representação formal da estrutura dos dados, o JSON Schema pode servir como uma excelente documentação para a API.

Criando um JSON Schema

Temos quatro elementos principais nesse esquema, e são eles que dão o start na nossa história. Esses elementos são chamados de palavras-chave e funcionam como se fossem as chaves do nosso JSON.

```
1 {
2   "$schema": "http://json-schema.org/draft-07/schema#",
3   "$id": "https://[redacted]",
4   "title": "Retornar as informações do consumidor logado.",
5   "description": "Um esquema para validar as informações do consumidor logado.",
6   "type": "object",
7   "properties": {
```

Criando um JSON Schema

Agora vamos adicionar as propriedades específicas dos dados do usuário:

```
1 {
2   "$schema": "http://json-schema.org/draft-07/schema#",
3   "$id": "https://[redacted].org",
4   "title": "Retornar as informações do consumidor logado.",
5   "description": "Um esquema para validar as informações do consumidor logado.",
6   "type": "object",
7   "properties": {
8     "data": {
9       // ...
10    },
11    "isValid": {
12      // ...
13    },
14    "errors": {
15      // ...
16    }
17  }
```

```
8      data : {
9          "type": "object",
10         "properties": {
11             "nome": {
12                 "type": "string",
13                 "description": "Nome completo do usuário."
14             },
15             "primeiroNome": {
16                 "type": "string",
17                 "description": "Primeiro nome do usuário."
18             },
19             "dataNascimento": {
20                 "type": "string",
21                 "format": "date-time",
22                 "description": "Data de nascimento do usuário."
23             },
24             "enderecoEmail": {
25                 "type": "string",
26                 "format": "email",
27                 "description": "Endereço de e-mail do usuário."
28             },
29             "telefoneCelular": {
30                 "type": "string",
31                 "description": "Número de telefone celular do usuário."
32             },
33             "dataCadastro": {
34                 "type": "string",
35                 "format": "date-time",
36                 "description": "Data de cadastro do usuário."
37             }
38         },
39         "required": ["nome", "primeiroNome", "dataNascimento", "enderecoEmail", "telefoneCelular", "d
40         "additionalProperties": false
```



```
1 "username": {
2   "type": "string",
3   "minLength": 5,
4   "maxLength": 20
5 }
```



```
1 "dataHoraIso": {
2   "type": "string",
3   "format": "iso-date-time"
4 }
```



```
1 "uuid": {
2   "type": "string",
3   "format": "uuid"
4 }
```



```
1 "empresa": {
2   "type": "string",
3   "const": "Acerto"
4 }
```



```
1 "data": {
2   "type": "string",
3   "format": "date"
4 }
```



```
1 "horaIso": {
2   "type": "string",
3   "format": "iso-time"
4 }
```



```
1 "dataHora": {
2   "type": "string",
3   "format": "date-time"
4 }
```



```
1 "hora": {
2   "type": "string",
3   "format": "time"
4 }
```



```
1 "duração": {
2   "type": "string",
3   "format": "duration"
4 }
5
```



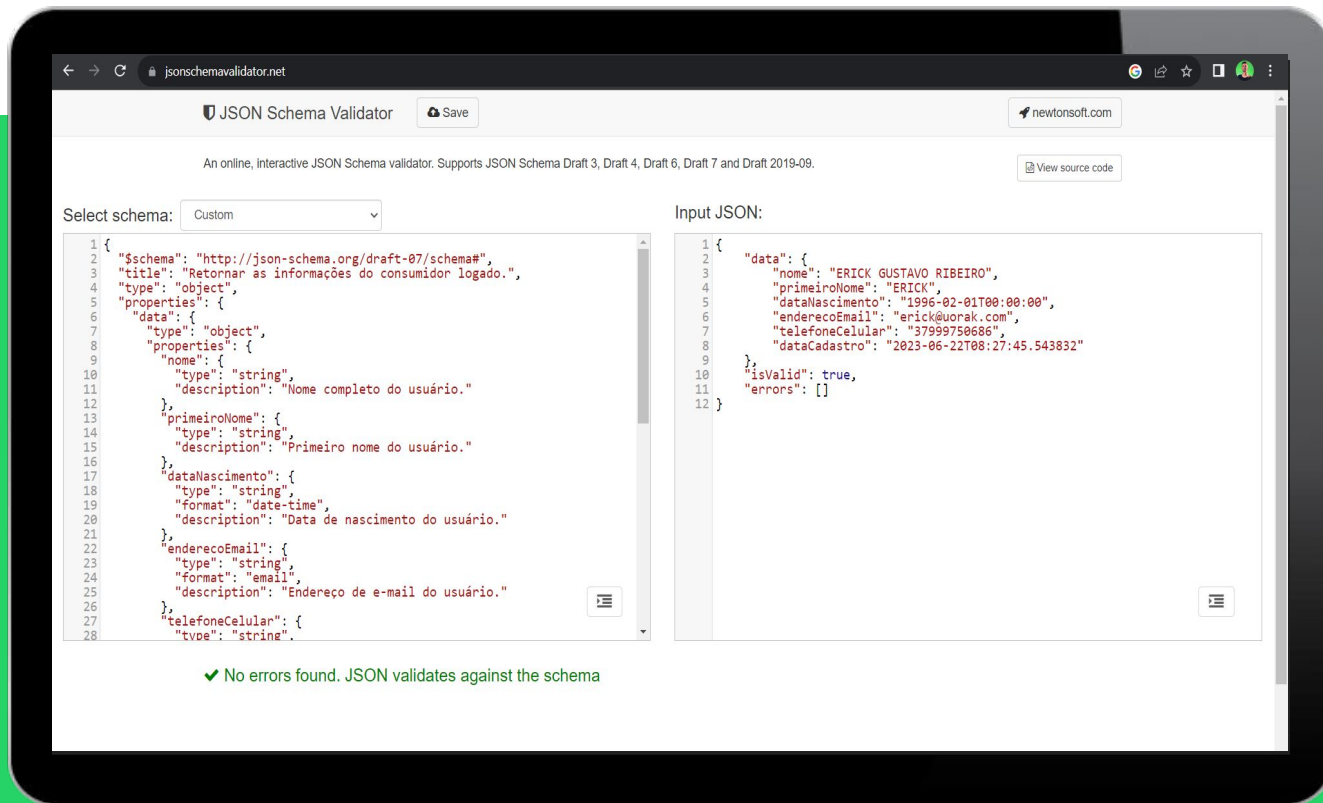
```
1 "telefoneCelular": {
2   "type": "string",
3   "pattern": "^[1-9]{2}9[1-9]{1}[0-9]{7}$"
4 }
5
```

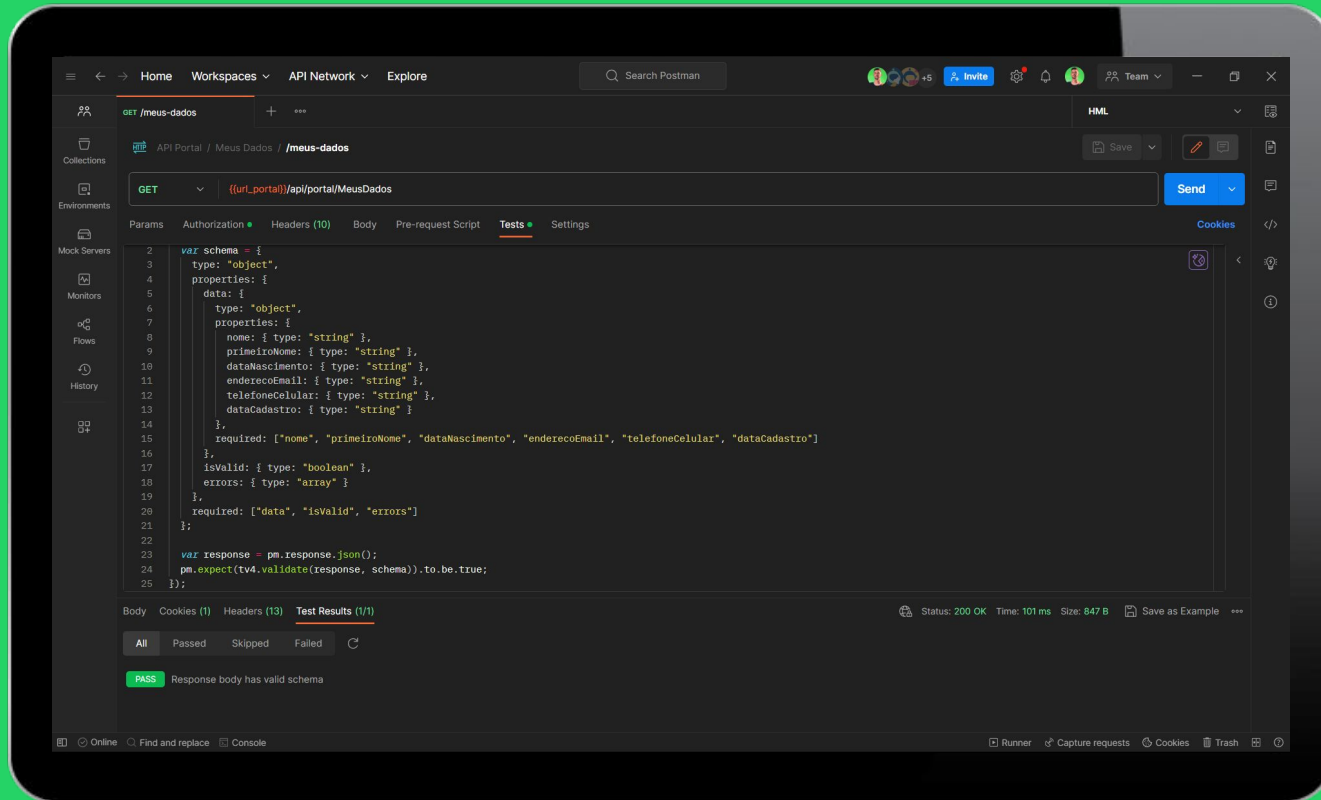


```
1 "website": {
2   "type": "string",
3   "format": "uri"
4 }
```

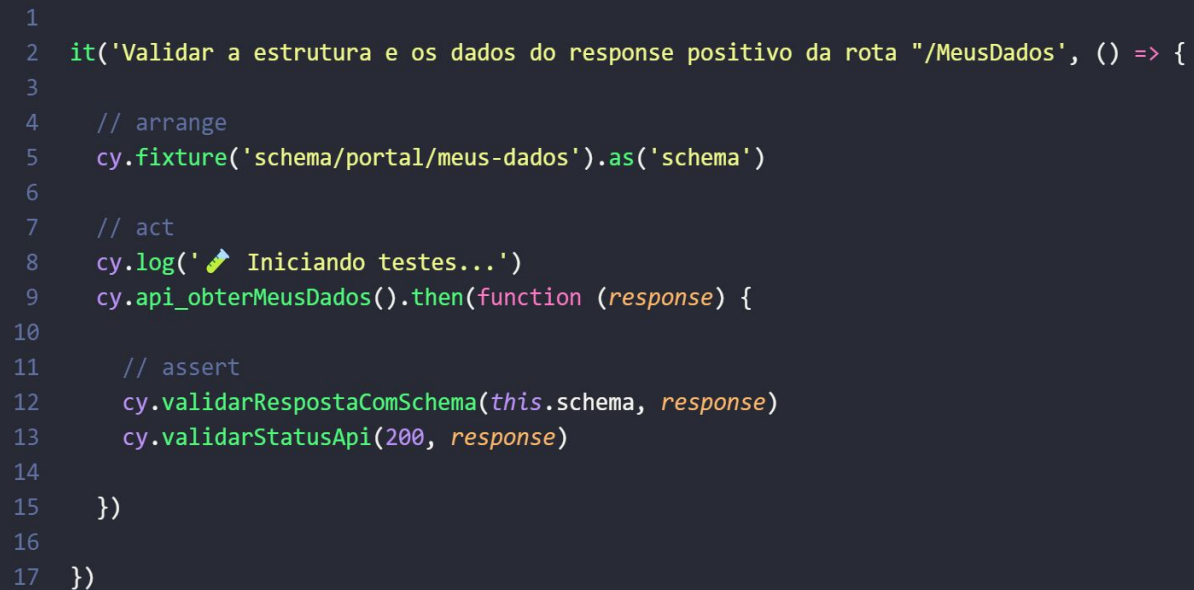


Implementando Testes de API utilizando JSON Schema





Semi Automatizado



```
1
2  it('Validar a estrutura e os dados do response positivo da rota "/MeusDados', () => {
3
4    // arrange
5    cy.fixture('schema/portal/meus-dados').as('schema')
6
7    // act
8    cy.log('🔧 Iniciando testes...')
9    cy.api_obterMeusDados().then(function (response) {
10
11      // assert
12      cy.validarRespostaComSchema(this.schema, response)
13      cy.validarStatusApi(200, response)
14
15    })
16
17  })
```

Automatizado