INSTITUTO FEDERAL
Alagoas

Testes automatizados

Márcio Robério da Costa Ferro
marcio.roberio@ifal.edu.br
junho de 2021

Conteúdo

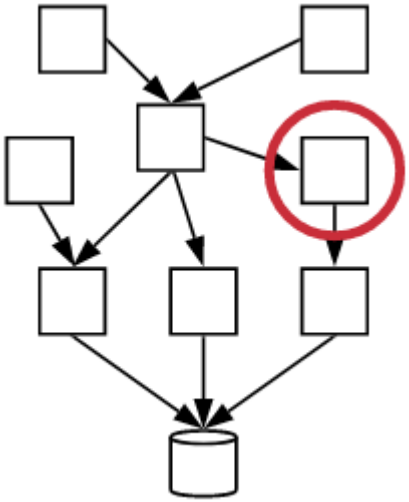
INSTITUTO FEDERAL
Alagoas

- Testes de Software
- Tipos de Testes
- Testes Automatizados com JavaScript
- Cypress

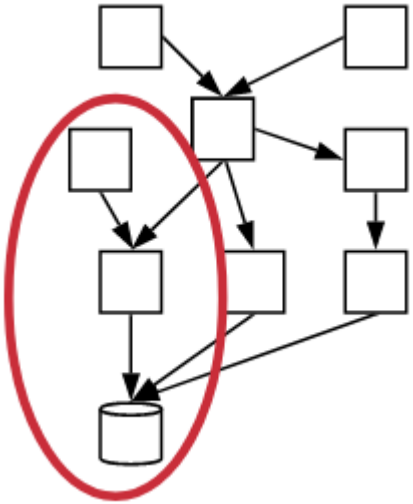
O que são testes?

- Meio de identificar se um software possui erros e inconsistências
- É recomendável que os testes sejam realizados a medida que o código seja criado
  - não deixe para testar tudo por último!
- Há ferramentas para realizar testes automatizados

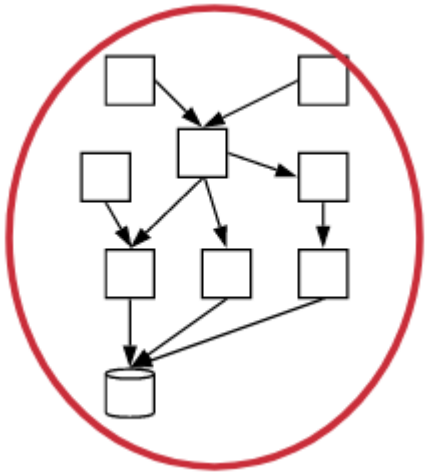
Tipos de Testes



Testes de Unidades



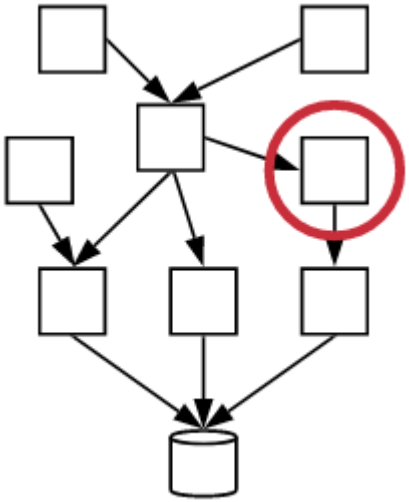
Testes de Integração



Testes de Sistema

https://engsoftmoderna.info/cap8.html

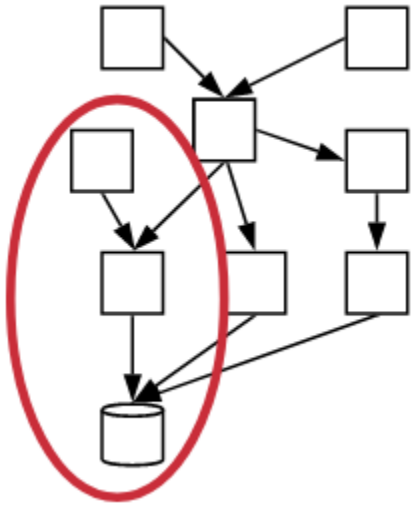
Testes de Unidade



Testes de Unidades

- Verificam pequenas partes de um código, como por exemplo:
  - Uma função
  - Uma classe ou um método da classe
- São mais simples de implementar que os demais testes

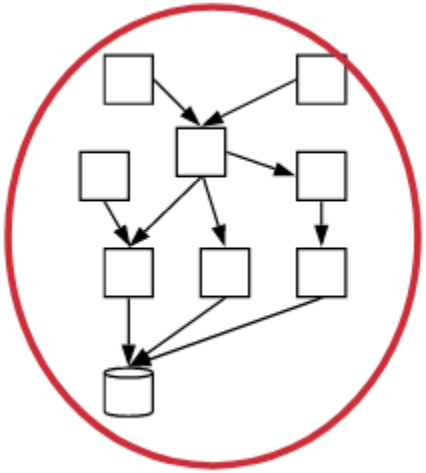
Testes de Integração



Testes de Integração

- Também chamados de testes de serviços
- Verificam uma funcionalidade ou uma transação completa de um sistema. Exemplo:
  - Cadastro de um usuário
  - Remoção de um produto
- Esses testes geralmente usam diversas classes ou funções, além de poder usar o banco de dados
- Demandam mais esforço de implementação que os testes de unidades

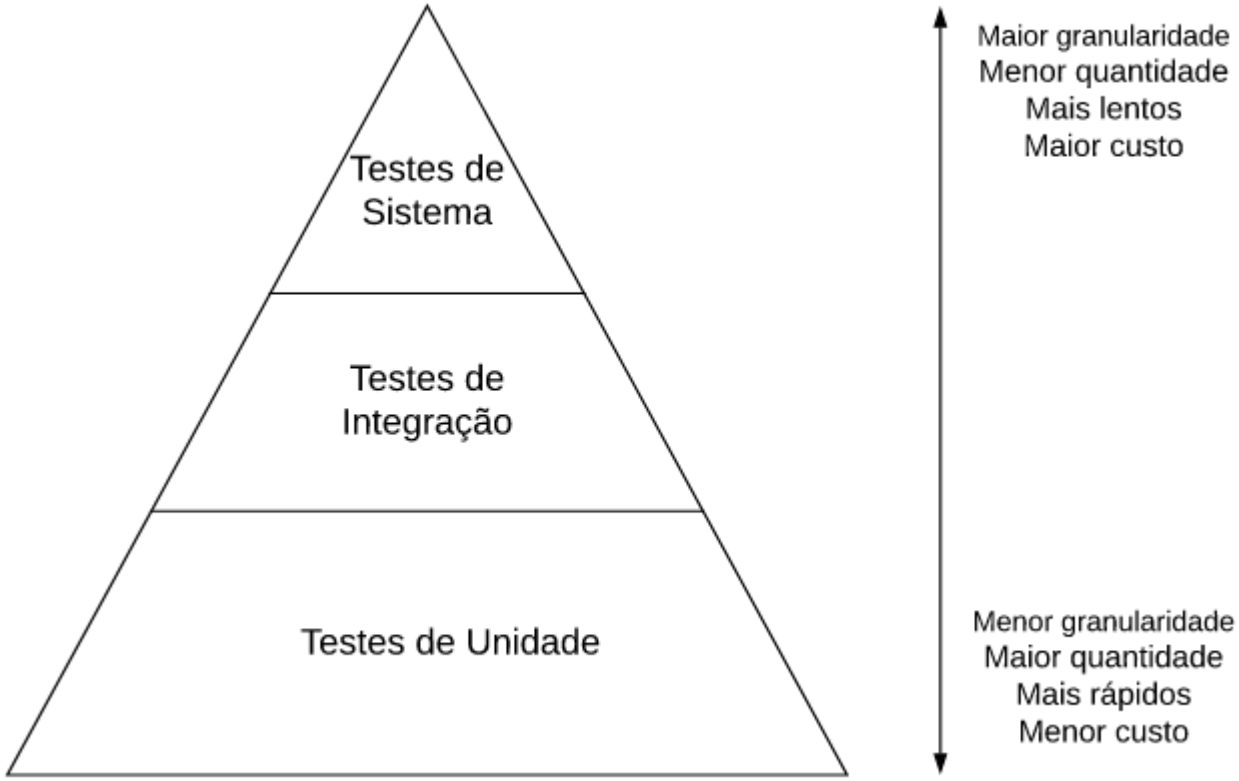
Testes de Sistema



Testes de Sistema

- Também chamados de testes de interface com o usuário ou testes ponta a ponta (*end-to-end)*
- Simulam o uso do sistema por um usuário
- Demandam mais esforço de implementação que os testes de integração
- São frágeis (alterações na interface podem demandar alterações nos testes)

Tipos de Testes



https://engsoftmoderna.info/cap8.html

Teste Unitário

Implementando uma “unidade”:
➔ uma função que compara se um número é maior, menor ou igual a outro número

funcoes.js > <unknown>

```
1  const compara = (a,b) => {
2      if (a > b)
3          return "maior"
4      else
5          return "menor"
6  }
7
8  module.exports = { compara }
```

Teste Unitário

Testando a unidade “compara”

```
JS funcoes.test.js > ...
1
2     const { compara } = require("./funcoes.js");
3     var assert = require('assert');
4
5     assert(compara(8,2)=="maior","8 é maior que 2")
6     assert(compara(8,8)=="igual","8 é igual a 8")
```

Para executar no Visual Studio: Control + ALT + N

Teste Unitário

INSTITUTO FEDERAL
Alagoas

Saída

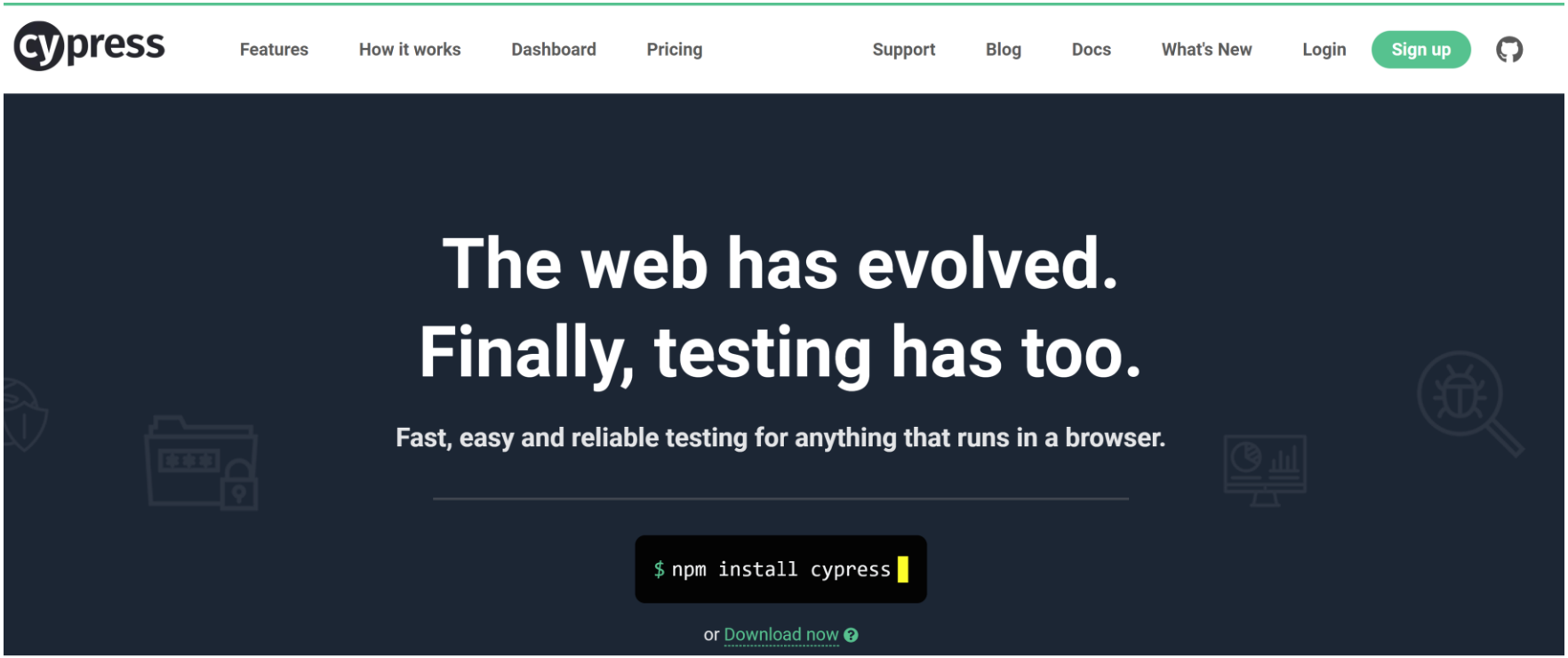
```
node:assert:399
    throw err;
    ^

AssertionError [ERR_ASSERTION]: 8 é igual a 8
    at Object.<anonymous> (c:\Users\marci\OneDrive\Área de Trabalho\test\funcoes.test.js:7:1)
    at Module._compile (node:internal/modules/cjs/loader:1103:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1157:10)
    at Module.load (node:internal/modules/cjs/loader:981:32)
    at Function.Module._load (node:internal/modules/cjs/loader:822:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:77:12)
    at node:internal/main/run_main_module:17:47 {
  generatedMessage: false,
  code: 'ERR_ASSERTION',
  actual: false,
  expected: true,
  operator: '=='
}
```

Testes Automatizados

- Há ferramentas de testes para diferentes linguagens de Programação
- Algumas bibliotecas que podem ser usados para criação de testes de sistemas

Cypress



https://www.cypress.io/

Primeiros passos

Como instalar Cypress no Windows:

mkdir <dir_name>>

cd <dir_name>>

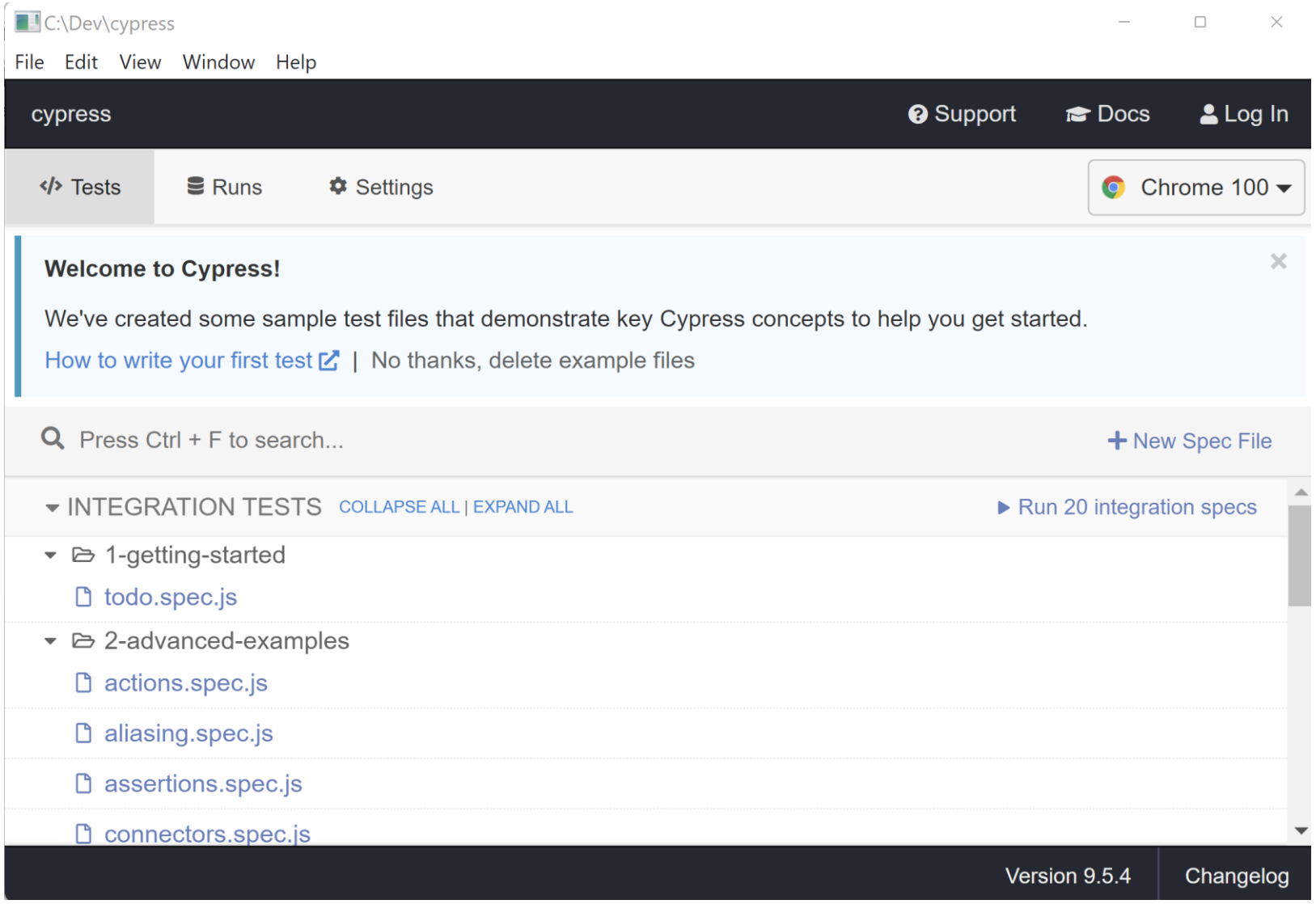
npm init –y

npm install cypress

Como abrir Cypress:

.\node_modules\.bin\cypress open

Primeiros passos



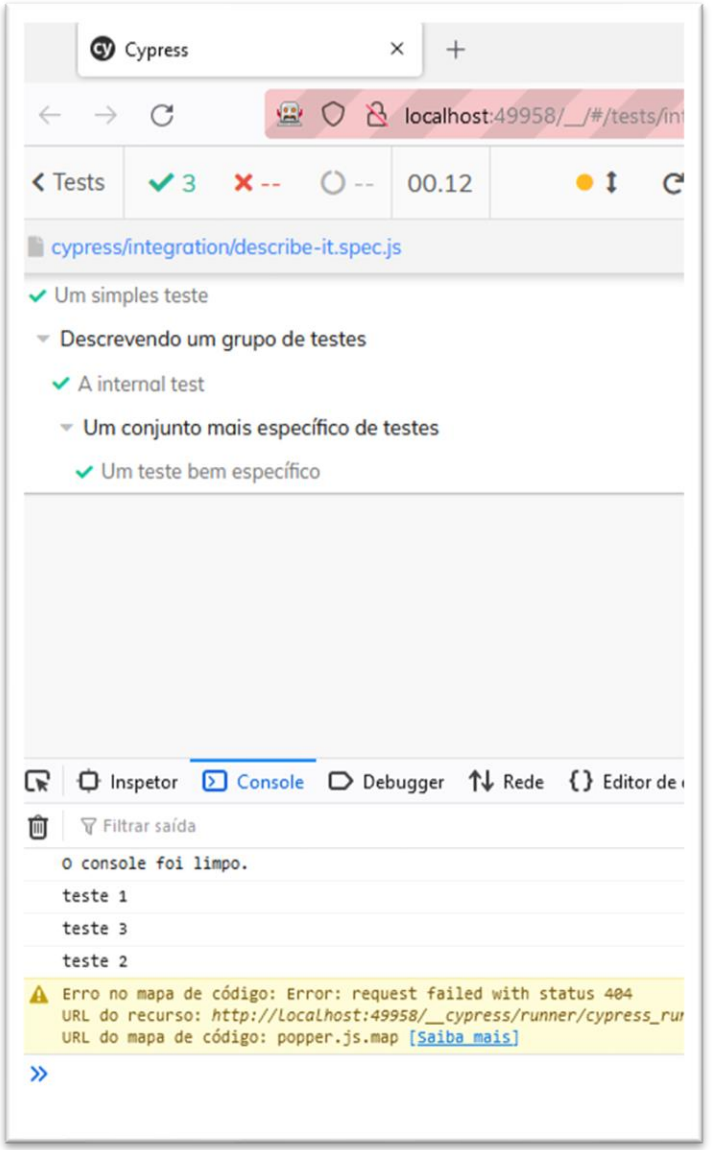
Describe e It

- Describe ➔ Grupo de testes
- It ➔ Um teste

describe-it.spec.js U

cypress > integration > describe-it.spec.js > ...

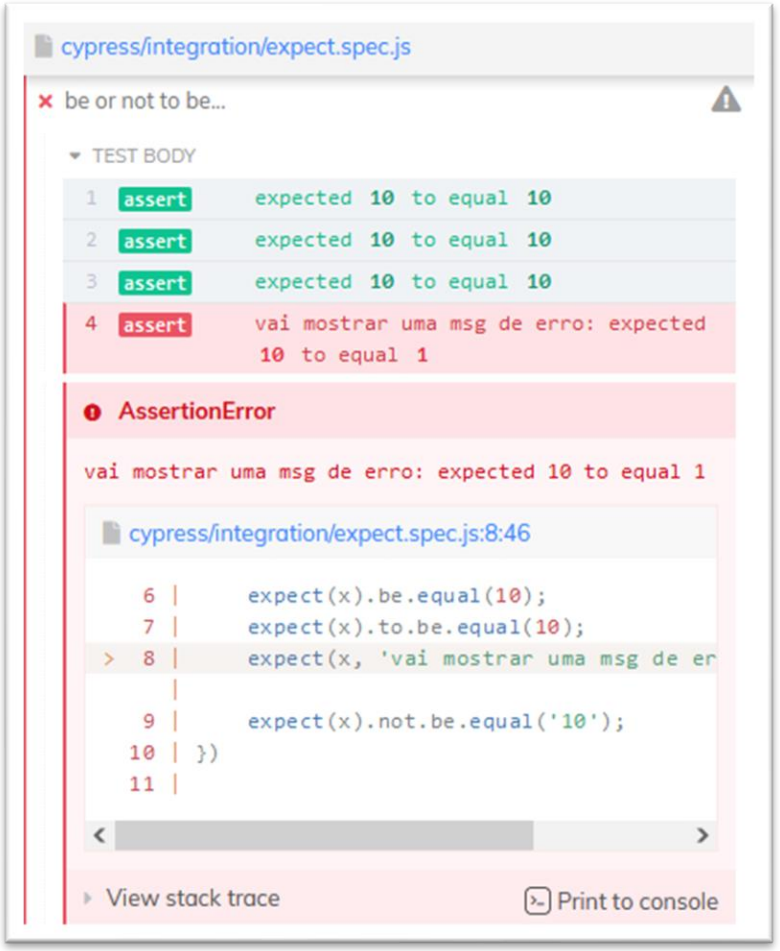
```
1   /// <reference types="cypress" />
2
3   it('Um simples teste', () => {
4       console.log('teste 1')
5   })
6
7   describe('Descrevendo um grupo de testes', () => {
8       describe ('Um conjunto mais específico de testes', () => {
9           it('Um teste bem específico', () => {
10              console.log('teste 2')
11          })
12      })
13
14      it ('A internal test', () => {
15          console.log('teste 3')
16      })
17  })
```



Assert

- Expect → assertiva
- Pode-se comparar um valor com outro, usando a seguinte sintaxe:
  `expect(variável).equals(valor esperado)`
  ou uma de suas variações...

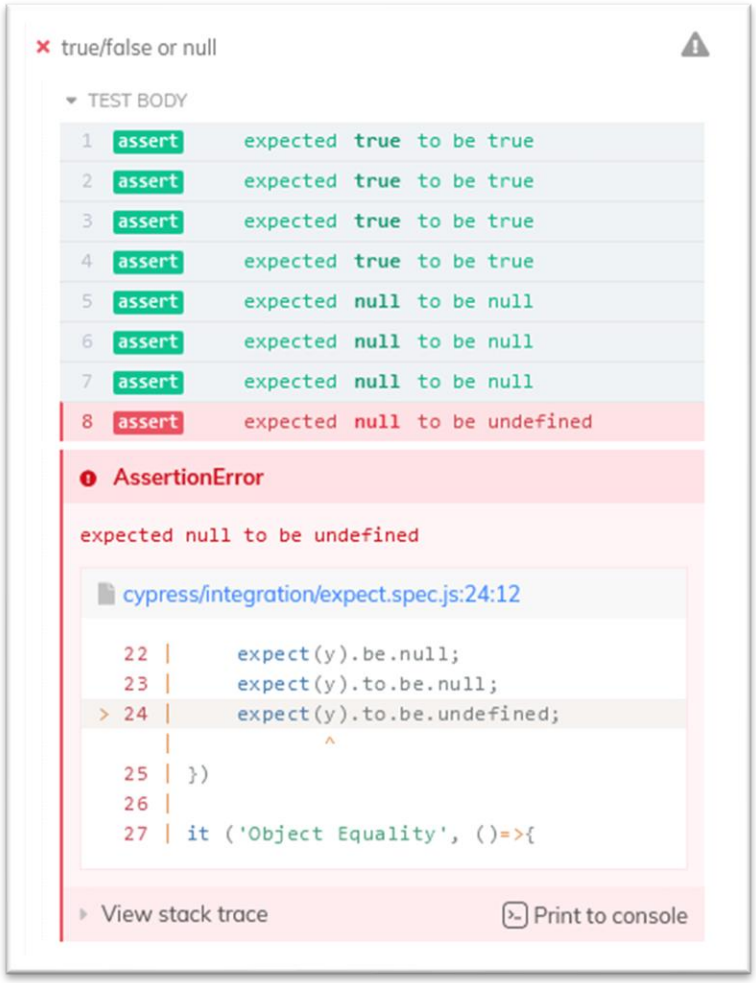
```
expect.spec.js U
cypress > integration > expect.spec.js > it('Verificando objetos') callback
1   /// <reference types="cypress" />
2
3   it('be or not to be...', () => {
4       const x = 10
5       expect(x).equal(10);
6       expect(x).be.equal(10)
7       expect(x).to.be.equal(10)
8       expect(x, 'vai mostrar uma msg de erro').equal(1)
9       expect(x).not.be.equal('10')
10  })
```



Assert

- Expect → Verificando valores true, false ou null

```
12  it('true/false or null', () => {
13      const x = true
14      let y = null
15
16      expect(x).true
17      expect(x).be.true
18      expect(x).to.be.true
19      expect(x).not.to.be.false
20
21      expect(y).null
22      expect(y).be.null
23      expect(y).to.be.null
24      expect(y).to.be.undefined
25  })
```



Assert

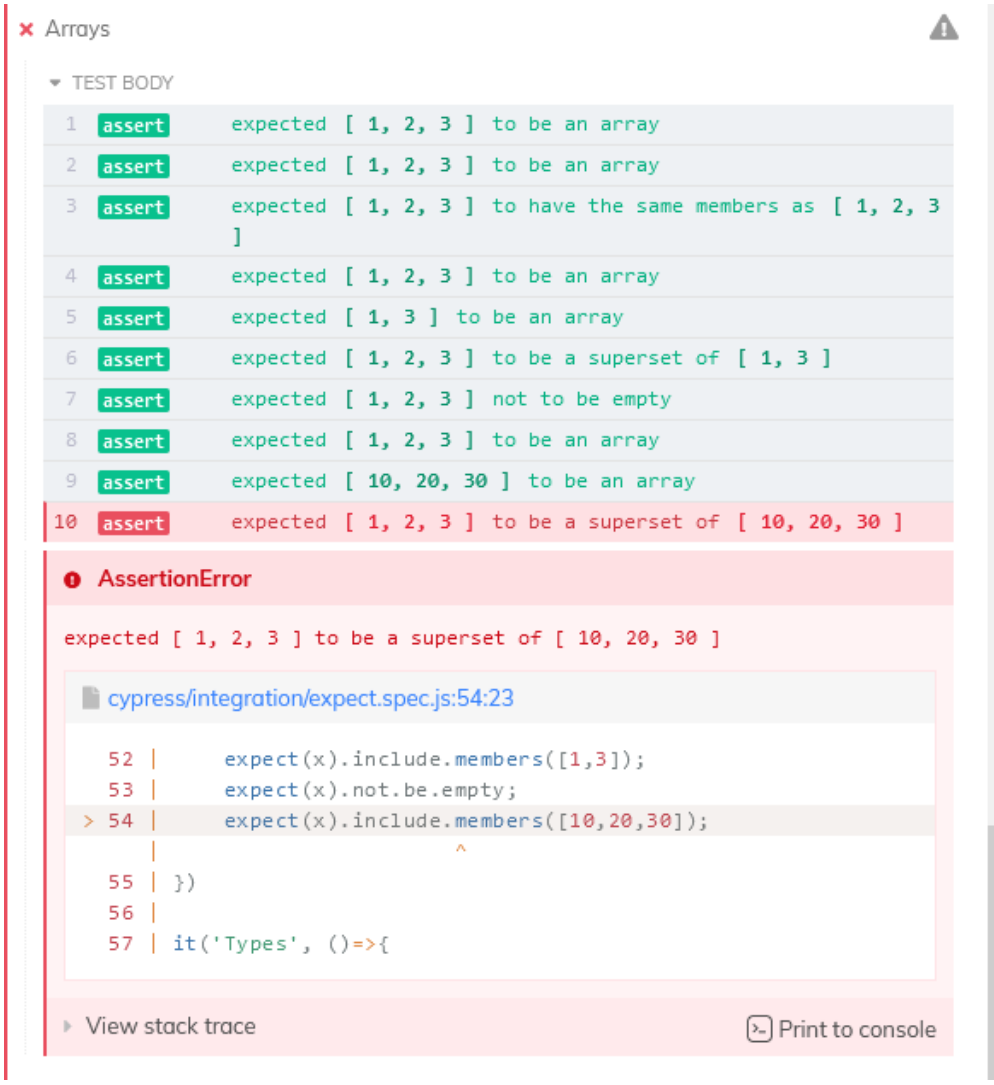
- Expect → Verificando valores de objetos

```
27  it ('Verificando objetos', ()=>{
28      const obj = {
29          x:10,
30          y:20
31      }
32      expect(obj).equal(obj)
33      expect(obj).equals(obj)
34      expect(obj).eq(obj)
35
36      expect(obj).not.be.empty
37
38      expect(obj).eql({x:10,y:20})
39      expect(obj).deep.equal({x:10,y:20})
40      expect(obj).include({x:10,y:20})
41
42      expect(obj).to.have.property('x')
43      expect(obj).to.have.property('x',10)
44
45      expect(obj).include({c:20})
46  })
```

Assert

- Expect → Verificando arrays

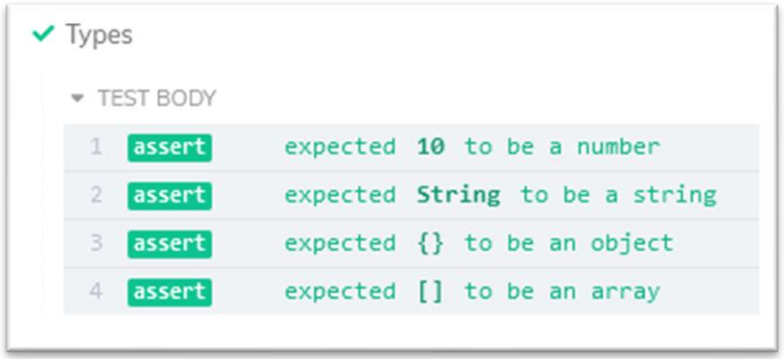
```
49  it('Arrays', ()=>{
50      const x = [1,2,3];
51      expect(x).have.members([1,2,3]);
52      expect(x).include.members([1,3]);
53      expect(x).not.be.empty;
54      expect(x).include.members([10,20,30]);
55  })
```



Assert

- Expect → Verificando tipos de dados

```
57  it('Types', ()=>{
58      const x = 10
59      const y = 'String'
60      expect(x).a('number')
61      expect(y).a('string')
62      expect({}).an('object')
63      expect([]).an('array');
64  })
```



Assert

- Expect → comparação de números

```
66  it('String', ()=>{
67      const x = 'teste@gmail.com'
68      const y = 'testegmail.com'
69
70      expect(x).equal('teste@gmail.com')
71      expect(x).to.have.length(15)
72      expect(x).to.contains('@')
73      expect(x).to.match(/^teste/)
74      expect(x).to.match(/.com$/)
75      expect(x).to.match(/\w+/)
76      expect(x).to.match(/^[a-z]\w*@\w*\.[a-z]+/)
77      expect(y).to.match(/^[a-z]\w*@\w*\.[a-z]+/)
78  })
```

O básico de expressões regulares

^	Inícia com o caractere que vem posteriormente
$	Termina com o caractere que vem anteriormente
[a-z]	Qualquer letra entre “a” a “z” (minúsculo)
[A-z]	Qualquer letra entre “a” a “z” (maiúsculo e minúsculo)
[0-9]	Qualquer número entre 0 e 9
[A-z0-9]	Qualquer letra ou número
[a-e]	Valida as letras a, b, c, d, e
[^xy]	A String não deve ter a sequência “xy” (negação)
\w	Qualquer caractere alfanumérico (letras e números)
\W	Pontos e espaços
*	Zero ou mais ocorrência de um caractere antecedente
+	Uma ou mais vezes do caractere antecedente
.	Qualquer caractere
?	Zero ou uma vez do caractere antecedente
\	Escape *(exemplo: \$ ➔ escapa o símbolo usado para definir o final da expressão)*
|	Concatenação
()	Forma um grupo lógico *(exemplo: (23)+ ➔ valida qualquer String que contenha 23)*

Exemplo:

```
[a-z]\w*@\w*\.[a-z]+
```

Testar expressões ➔ https://regex101.com/r/vS7vZ3/224#Javascript

Assert

- Expect → Strings

```
80   it('Números', () => {
81       const inteiro = 10
82       const flutuante = 20.5
83       expect(inteiro).equal(10)
84       expect(inteiro).above(9)
85       expect(inteiro).below(11)
86
87       // próximo a 21, +- 0.5
88       expect(flutuante).closeTo(21,0.5)
89   })
```

✔ Números

▾ TEST BODY

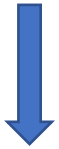
1	assert	expected 10 to equal 10
2	assert	expected 10 to be above 9
3	assert	expected 10 to be below 11
4	assert	expected 20.5 to be a number
5	assert	expected 20.5 to be close to 21 +/- 0.5

Abrindo uma página

cy.visit(“ URL DA PÁGINA ”) ➔ abre uma página

cy.title() ➔ retorna o título da página

cy.title().should(FUNÇÃO, “valor para comparar”)



“be.equal” ➔ ser igual a...

“contain” ➔ contém a string informada

Acessar uma página

visit.spec.js M

cypress > integration > visit.spec.js > describe('acessando uma página') callback

```
1   /// <reference types="cypress" />
2
3   describe('acessando uma página', () => {
4       it ('verifica o título da página', () =>{
5           cy.visit('https://www.ifal.edu.br/')
6
7           cy.title().should('be.equal', 'Instituto Federal de Alagoas')
8           cy.title().should('contain', 'Alagoas')
9
10          cy.title()
11              .should('be.equal', 'Instituto Federal de Alagoas')
12              .should('contain', 'Alagoas')
13
14          cy.title()
15              .should('be.equal', 'Instituto Federal de Alagoas')
16              .and('contain', 'Alagoas')
17
18          cy.title().should('be.empty')
19      });
20
21  })
```

Diferentes formas de verificar o conteúdo do título

Formulários

Preenchimento

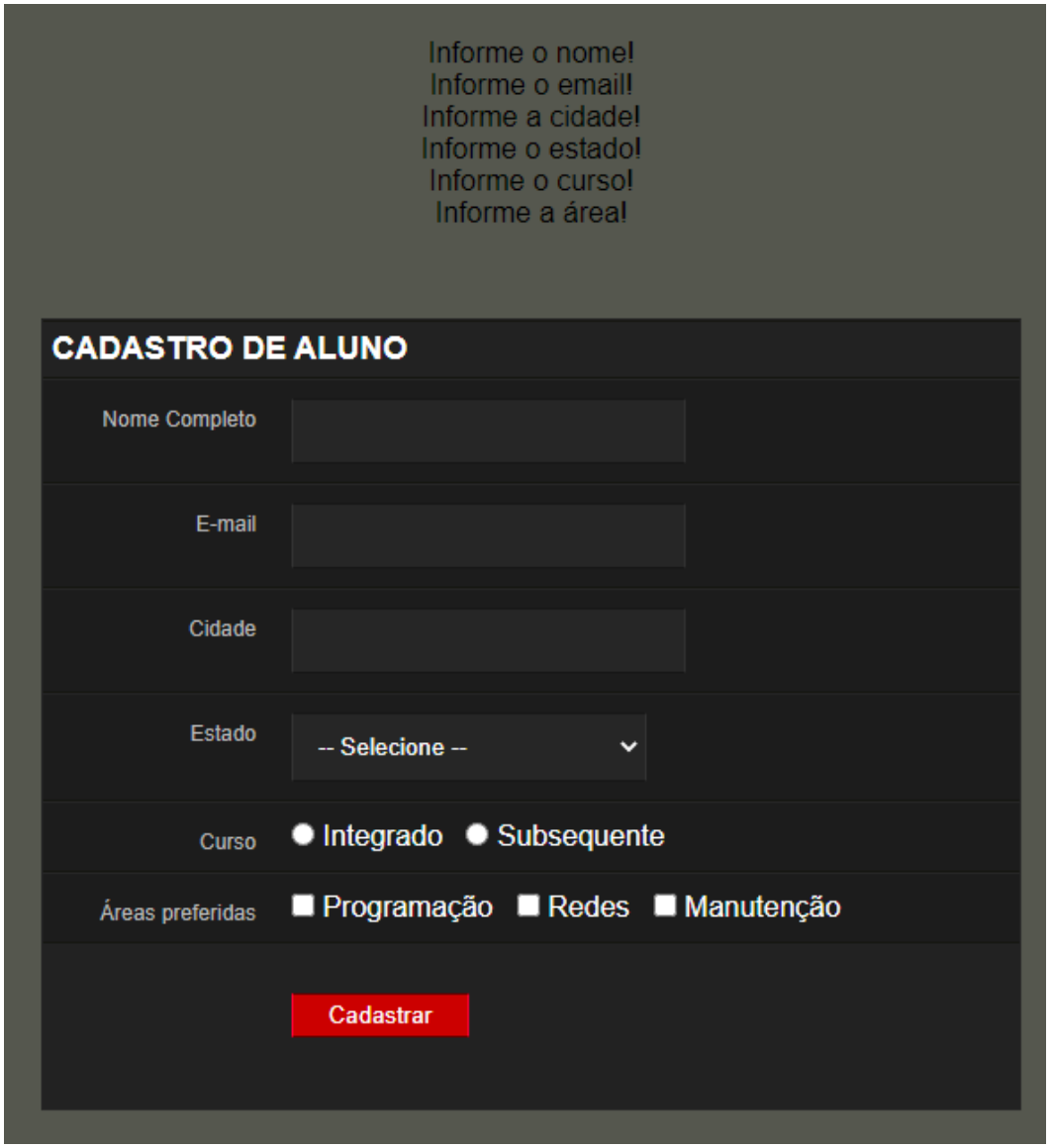
get(“id_elemento).type()

get(“id_elemento).select()

get(“id_elemento).click()

Verificação

get(“id_elemento).should()



Formulário

formulario-atividade.html M | form.spec.js U

cypress > integration > form.spec.js > describe('Preenchimento de formulário') callbac

```
1  /// <reference types="cypress" />
2
3  describe('Preenchimento de formulário', () => {
4
5      it ('preenchendo todos os campos', () =>{
6          cy.visit('../form/formulario-atividade.html')
7
8          cy.get("#nome").type("Roberio")
9          cy.get("#email").type("marcio.roberio@ifal.edu.br")
10         cy.get("#cidade").type("Maceió")
11         cy.get("#estado").select("Alagoas")
12         cy.get("#curso_subsequente").click()
13         cy.get("#area_programacao").click()
14         cy.get(".button").click()
15
16         cy.get("#msg_nome").should("have.text", "")
17         cy.get("#msg_email").should("be.empty")
18         cy.get("#msg_cidade").should("be.empty")
19         cy.get("#msg_estado").should("be.empty")
20         cy.get("#msg_curso").should("be.empty")
21         cy.get("#msg_area").should("be.empty")
22
23     })
24 })
25
```

Tests | 1 | -- | -- | 23.59

cypress/integration/form.spec.js

TEST BODY

1	visit	../form/formulario-atividade.html
2	get	#nome
3	-type	Roberio
4	get	#email
5	-type	marcio.roberio@ifal.edu.br
6	get	#cidade
7	-type	Maceió
8	get	#estado
9	-select	Alagoas
10	get	#curso_subsequente
11	-click	
12	get	#area_programacao
13	-click	
14	get	.button
15	-click	
	(form sub)	--submitting form--
	(page load)	--page loaded--
16	get	#msg_nome
17	-assert	expected <div#msg_nome> to have text ''
18	get	#msg_email
19	-assert	expected <div#msg_email> to be empty
20	get	#msg_cidade
21	-assert	expected <div#msg_cidade> to be empty
22	get	#msg_estado
23	-assert	expected <div#msg_estado> to be empty
24	get	#msg_curso
25	-assert	expected <div#msg_curso> to be empty
26	get	#msg_area
27	-assert	expected <div#msg_area> to be empty

Formulário

Lista de comandos

https://example.cypress.io/