
Cypress vs TestCafe

— Taís Mafioleti —
Automation QA



Quem sou eu



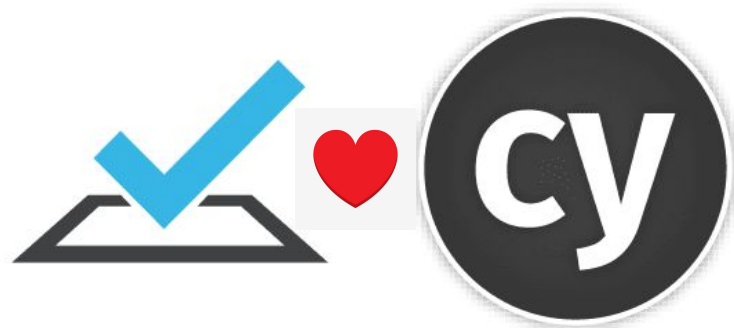
- > 6 anos de experiência como QA
- > Quase 1 ano como desenvolvedora front-end
- > Atualmente na FCamara em Santos
- > Foco em automação e análise



O que elas têm em comum?



1. Ambas são frameworks de testes e2e de código aberto e são executados parcialmente no navegador e parcialmente no Node.JS
2. Ambas têm tentativas de asserções transparentes (que elimina qualquer tipo de semelhança com o Selenium <3)
3. Comunidade forte e robusta
4. Ambas utilizam a biblioteca Chai.JS



As diferenças



Além da utilização do Chai para asserções, o Cypress também possibilita o uso do Sinon para mocks.

Com isso o Dev já se sente familiarizado com a tecnologia

* Sinon é um framework de testes js utilizado para mocks, spies e stubs



As diferenças



TestCafe tem seu próprio executor com sua própria sintaxe "fixture"

```
fixture`Tests`.page(constants.URL).beforeEach(async (t) => {  
  await t.click(Pagemodel.buttonSingin);  
});
```

As diferenças



Cypress trabalha com filas para fazer o código assíncrono parecer mais síncrono

4	get	#password
5	- type	Bevi@Teste052
6	get	#onLoginButton
7	- click	
	(xhr)	● POST 200 /authentication
8	wait	5000
	(xhr)	● POST 200 /rb_bf25498xj?type=js38sn=v_4_srv_1_s...
	(new url)	https://bevicred-homologacao.tooseg...
	(xhr)	● GET 200 /v1/vendas/search?identificacaoVendedor...
	(xhr)	● POST 200 /rb_bf25498xj?type=js38sn=v_4_srv_1_s...

▼ TEST BODY

1	get	a.btn
2	- click	{force: true}
	(new url)	https://bevicred-homologacao.tooseg...
3	get	slide:animated:nth-child(1) > div:nth-child(1) > div:nth-child(1) > button:nth-child(4)
4	- click	
5	get	#nome
6	- type	Manley Herzog
7	get	#cpf
8	- type	03505620670
9	get	#sexo

As diferenças



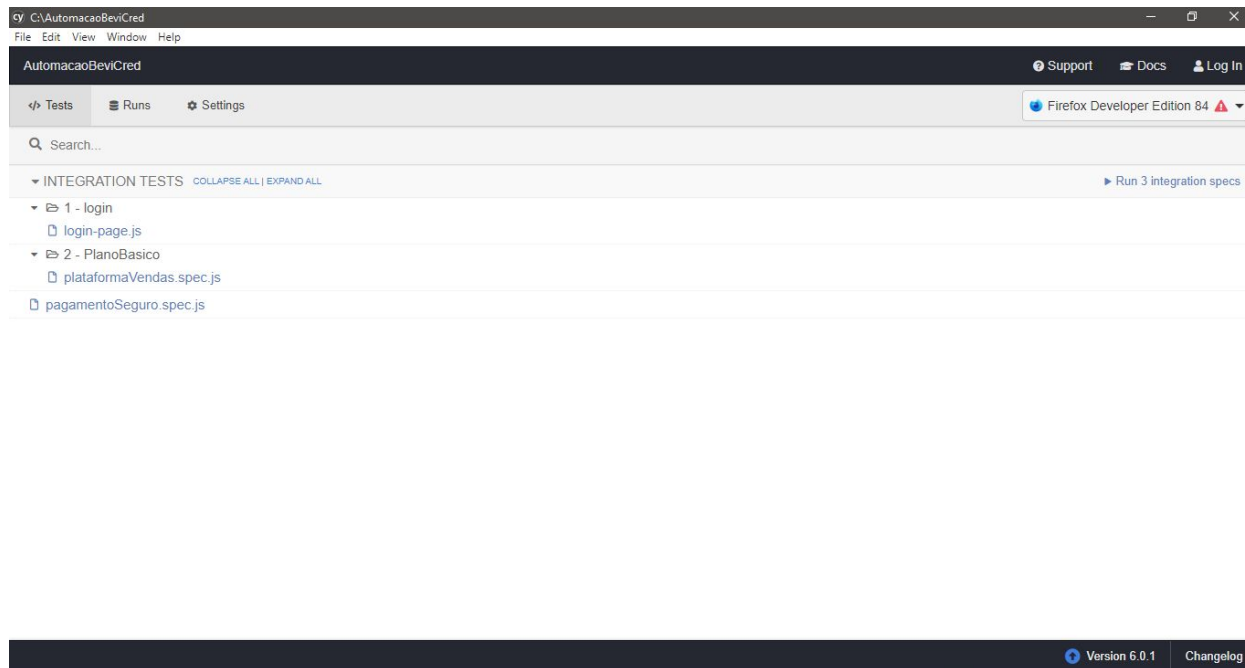
No TestCafe existe a necessidade de explicitar o `async/await` no gerenciamento das execuções

```
✓ test("Try to create an account by sending an existent email", async (t) => {  
  await t  
    .typeText(Pagemodel.inputEmailCreate, t.fixtureCtx.validEmail)  
    .click(Pagemodel.buttonCreateAccount)  
    .expect(Pagemodel.messageCreationAccount)  
    .eq(constants.MESSAGE_DUPLICITY);  
});
```

As diferenças



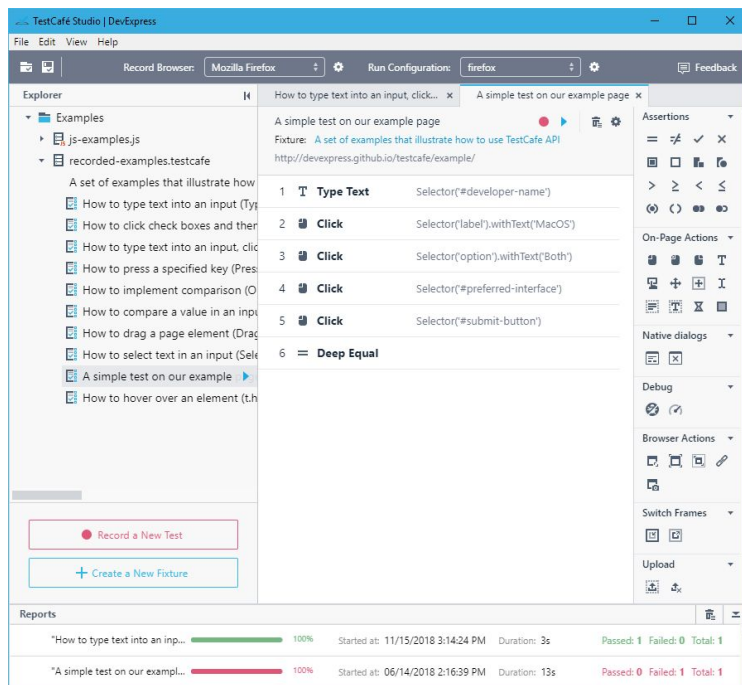
Cypress possui sua própria interface e realiza os testes utilizando suas próprias API's



As diferenças



Já o TestCafe tem sua própria IDE - O TestCafe Studio



As diferenças



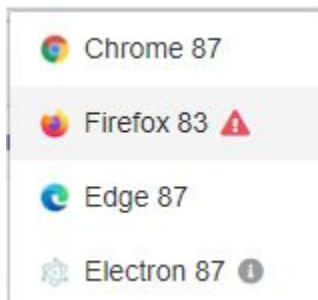
TestCafe funciona servindo o site que está sendo testado por meio de um servidor proxy. O servidor injeta scripts na página que podem inspecionar e controlar os elementos da página.

Isso significa que funciona em qualquer navegador da web.

As diferenças



Já o Cypress só tem suporte à Chrome, Edge e Firefox (que saiu recentemente)



As diferenças



Cypress executa seu código no teste real no processo do navegador, pois os testes têm acesso aos elementos DOM reais

Já a comunicação do TestCafe com os elementos DOM deve ser serializada pois ele realiza a execução com o Node

As diferenças



Em casos de erro, o cypress mostra os testes lado a lado com a tela e o local do erro, possibilitando passar o mouse pelos elementos para entender

```
20 -click {force: true}
(xhr) GET (aborted) /v2/cep/06401160
21 get #numero
```

❌ AssertionError

Timed out retrying: Expected to find element: `#numero`, but never found it.

▶ View stack trace Print to console

As diferenças



E apresenta o stacktrace no próprio executor

▼ View stack trace

Print to console

```
at novaVenda (https://bevicred-homologacao.tooseguros.com.br/__cypress/tests?p=cypress\integration\2:503:11)
at ./cypress/integration/2 - PlanoBasico/plataformaVendas.spec.js/ (https://bevicred-homologacao.tooseguros.com.br/__cypress/tests?p=cypress\integration\2:379:39)
at setRunnable/runnable.fn (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_runner.js:170263:46)
at callFn (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_runner.js:104396:22)
at ../driver/node_modules/mocha/lib/runnable.js/Runnable.prototype.run (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_runner.js:104383:14)
at onRunnableRun (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_runner.js:175798:29)
at finallyHandler (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_runner.js:7138:24)
at tryCatcher (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_runner.js:10584:24)
at .././node_modules/bluebird/js/release/promise.js/module.exports/Promise.prototype._settlePromiseFromHandler (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_runner.js:10584:24)
at .././node_modules/bluebird/js/release/promise.js/module.exports/Promise.prototype._settlePromise (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_runner.js:10584:24)
at .././node_modules/bluebird/js/release/promise.js/module.exports/Promise.prototype._settlePromise0 (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_runner.js:10584:24)
at .././node_modules/bluebird/js/release/promise.js/module.exports/Promise.prototype._settlePromises (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_runner.js:10584:24)
at _drainQueueStep (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_runner.js:5291:13)
at _drainQueue (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_runner.js:5284:25)
at .././node_modules/bluebird/js/release/async.js/Async.prototype._drainQueues (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_runner.js:5300:17)
at Async/this.drainQueues (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_runner.js:5170:15)
```

As diferenças



É possível debugar, e ao utilizar o debugger, no momento que o elemento do `.then` não for encontrado, o inspetor de código do navegador é aberto automaticamente

```
it('let me debug when the after the command executes', () => {  
  cy.visit('/my/page/path')  
  
  cy.get('.selector-in-question')  
    .then(($selectedElement) => {  
      // Debugger is hit after the cy.visit  
      // and cy.get command have completed  
      debugger  
    })  
})
```

As diferenças



E utilizando o `.debug`, é possível interagir com os elementos pelo próprio console

```

Elements Console Sources Network Performance Memory Application Security Audits
top Filter Default levels ▾ [x] Group similar

Console was cleared

[HMR] Waiting for update signal from WDS...

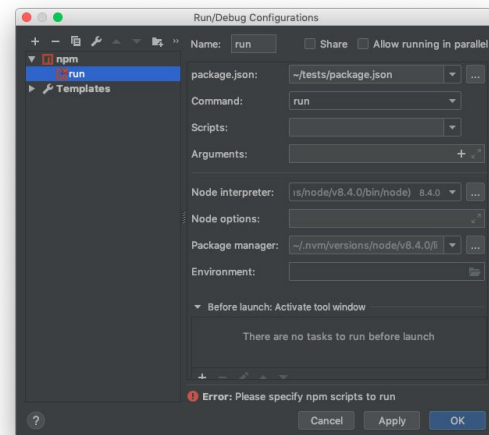
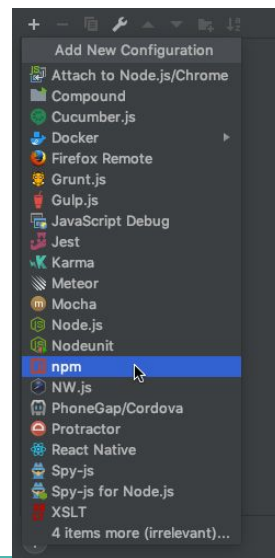
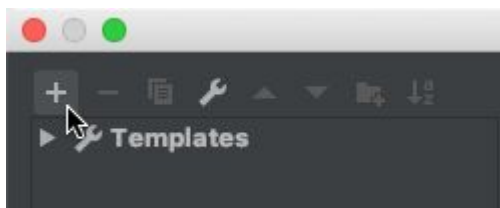
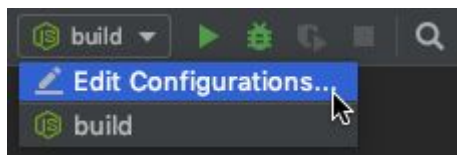
Download the Vue Devtools extension for a better development experience:
https://github.com/vuejs/vue-devtools

----- Debug Info -----
Command Name:      get
Command Args:      ▶ [".examples button"]
Current Subject:   ▶ jQuery.fn.init [button, prevObject: jQuery.fn.init(1), context: document, selector: ".examples button"]
> subject
< ▶ jQuery.fn.init [button, prevObject: jQuery.fn.init(1), context: document, selector: ".examples button"]
> subject.text()
< "Way Cool"
>
```


As diferenças



No TestCafe também é possível debugar, porém, para isso, é necessário uma série de configurações do NPM, e também do próprio package.json para que seja possível adicionar o debugger no código.



E por aí vai...

As diferenças



Mas no fim é possível adicionar breakpoints no próprio código do teste

The screenshot shows the Visual Studio Code editor with a file named `example.js` open. The code is a TestCafe test script. A breakpoint is set on line 13, which is highlighted in yellow. The code includes imports, a fixture, a page model, and a test function. The debug console at the bottom shows the command to run the test with debugging options and the message "Debugger attached."

```
1 import Page from './page-model';
2
3 fixture 'A set of examples that illustrate how to use TestCafe API'
4   .page 'https://devexpress.github.io/testcafe/example/';
5
6 // Page model
7 const page = new Page();
8
9 // Tests
10 test('Text typing basics', async t => {
11   await t
12     .typeText(page.nameInput, 'Peter')
13     .typeText(page.nameInput, 'Parker', { replace: true })
14     .typeText(page.nameInput, 'r', { caretPos: 2 })
15     .expect(page.nameInput.value).eql('Parker');
16 });
17
18
```

node --inspect=20018 --debug-brk node_modules/testcafe/bin/testcafe.js firefox /Users/testuser/tests/example.js
Debugger listening on ws://127.0.0.1:20018/Sb84c6c8-b518-45d4-940b-2878d20abe39
Debugger attached.

Ln 13, Col 10 Spaces: 4 UTF-8 LF JavaScript

As diferenças



Naturalmente realizar automações E2E requer muitas ferramentas para execução e criação, o Cypress consegue unificar diversas bibliotecas, libs e frameworks num lugar só.

Ele consegue também realizar testes back e front simultaneamente e sem perder a velocidade utilizando os recursos do próprio sistema operacional

As diferenças



Cypress também permite personalizar e sobrescrever comandos que já existem, só alterar no arquivo `cypress/support/commands.js`

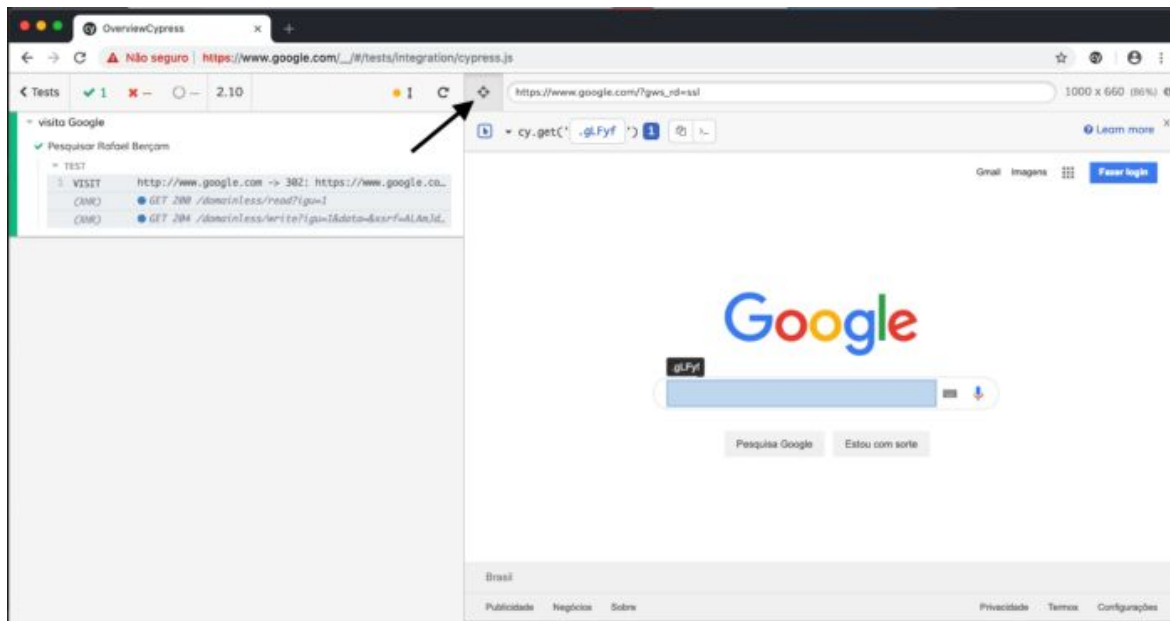
Ex:

```
Cypress.Commands.add('login', (email, pw) => {})  
Cypress.Commands.overwrite('visit', (orig, url, options) => {})
```

As diferenças



Uma função interessante do Cypress é o seletor de elementos no playground



As diferenças



O seletor de elementos segue uma hierarquia começando por atributos com o sufixo “data-*”, e quando isso é utilizado, ao ter alterações do JS ou CSS, a automação não é impactada mantendo a integridade do código dos testes

- data-cy
- data-test
- data-testid
- id
- class
- tag
- attributes
- nth-child

As diferenças



O TestCafe por sua vez possui um plugin que fornece extensões de seletores que tornam mais fácil testar componentes React com TestCafe

testcafe-react-selectors

This plugin provides selector extensions that make it easier to test ReactJS components with [TestCafe](#). These extensions allow you to select page elements in a way that is native to React.

Install

```
$ npm install testcafe-react-selectors
```

Usage

- [Wait for application to be ready to run tests](#)
- [Creating selectors for ReactJS components](#)
 - [Selecting elements by the component name](#)
 - [Selecting nested components](#)
 - [Selecting components by the component key](#)
 - [Selecting components by display name](#)
 - [Selecting components by property values](#)
 - [Properties whose values are objects](#)
 - [Searching for nested components](#)
 - [Combining with regular TestCafe selectors](#)
- [Obtaining component's props and state](#)
- [TypeScript Generic Selector](#)
 - [Composite Types in Props and State](#)
- [Limitations](#)

As diferenças



E o Cypress possui uma gama maior de plugins com extensões de mais linguagens

cypress-angular-unit-test

experimental

Test Angular component using Cypress Test Runner

#component #angular

cypress-angularjs-unit-test

experimental

Unit test Angularjs code using Cypress Test Runner

#component #angular.js

cypress-cycle-unit-test

experimental

Test Cycle.js components using Cypress Test Runner

#component #cycle.js

cypress-hyperapp-unit-test

experimental

Test Hyperapp components and applications using Cypress Test Runner

#component #hyperapp

cypress-react-unit-test

experimental

Test React components using Cypress Test Runner

#component #react

cypress-svelte-unit-test

experimental

Test Svelte components using Cypress Test Runner

#component #svelte

cypress-vue-unit-test

experimental

Test Vue.js components using Cypress Test Runner

#component #vue #vue.js

As diferenças



Com o Cypress é possível fazer testes unitários e de integração, diferentemente do TestCafe



@MafioletiTais



Taís Mafioleti

Muito obrigada!
