Cypress vs TestCafe

Taís Mafioleti
Automation QA

Quem sou eu

> 6 anos de experiência como QA

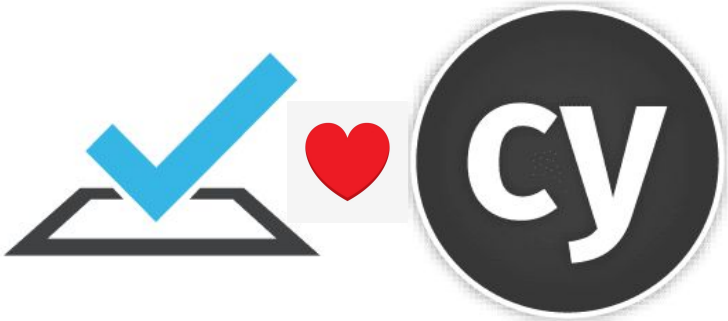
> Quase 1 ano como desenvolvedora front-end

> 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

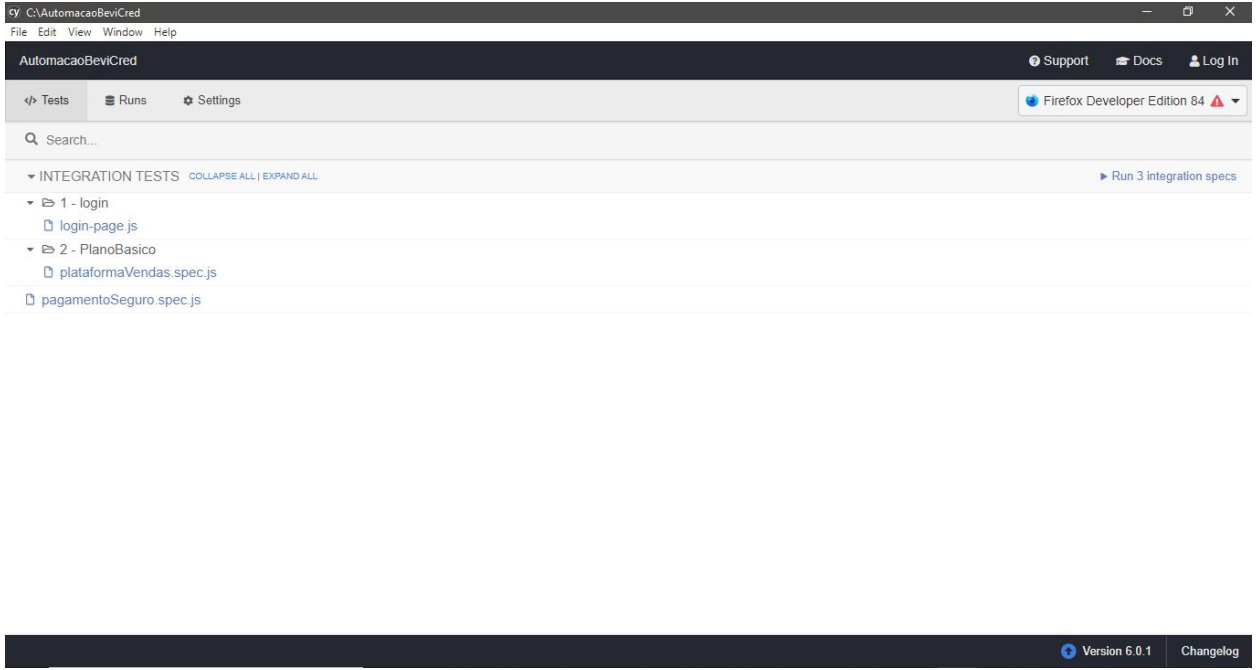
As diferenças

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)
      .eql(constants.MESSAGE_DUPLICITY);
});
```

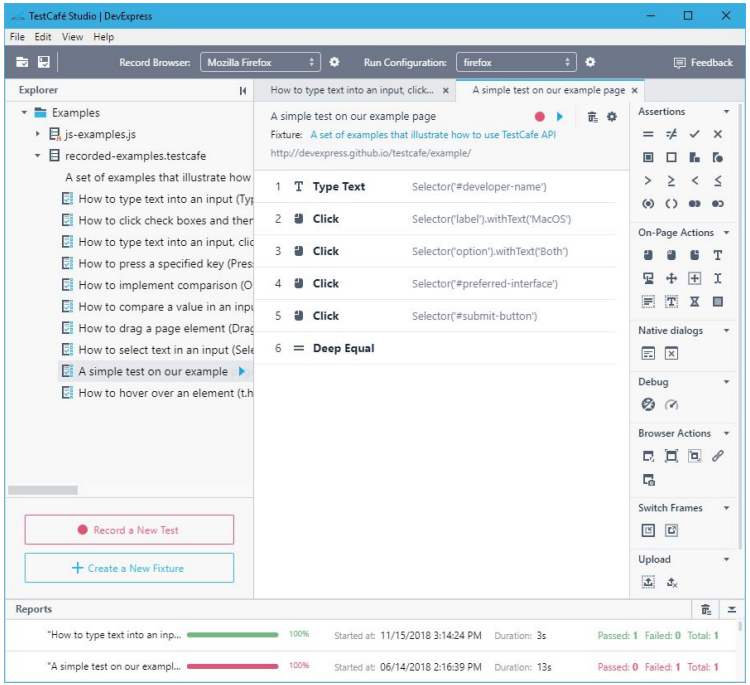
As diferenças

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)

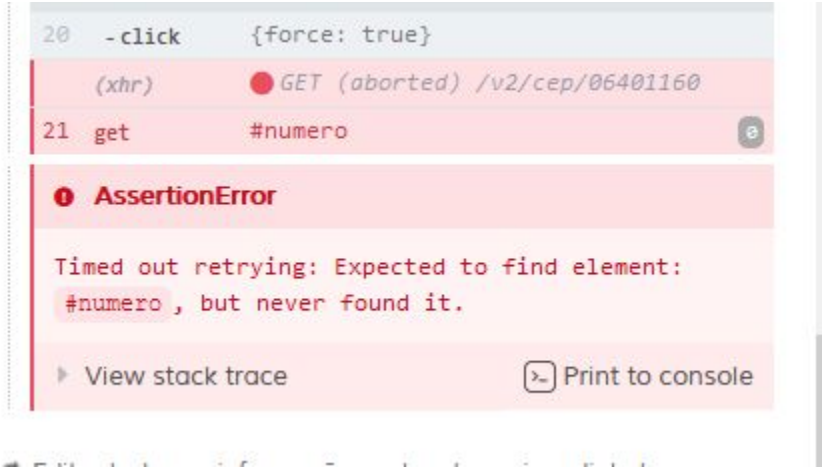
As diferenças

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



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
at ../../node_modules/bluebird/js/release/promise.js/</module.exports/Promise.prototype._settlePromise (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_ru
at ../../node_modules/bluebird/js/release/promise.js/</module.exports/Promise.prototype._settlePromise0 (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_r
at ../../node_modules/bluebird/js/release/promise.js/</module.exports/Promise.prototype._settlePromises (https://bevicred-homologacao.tooseguros.com.br/__cypress/runner/cypress_r
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

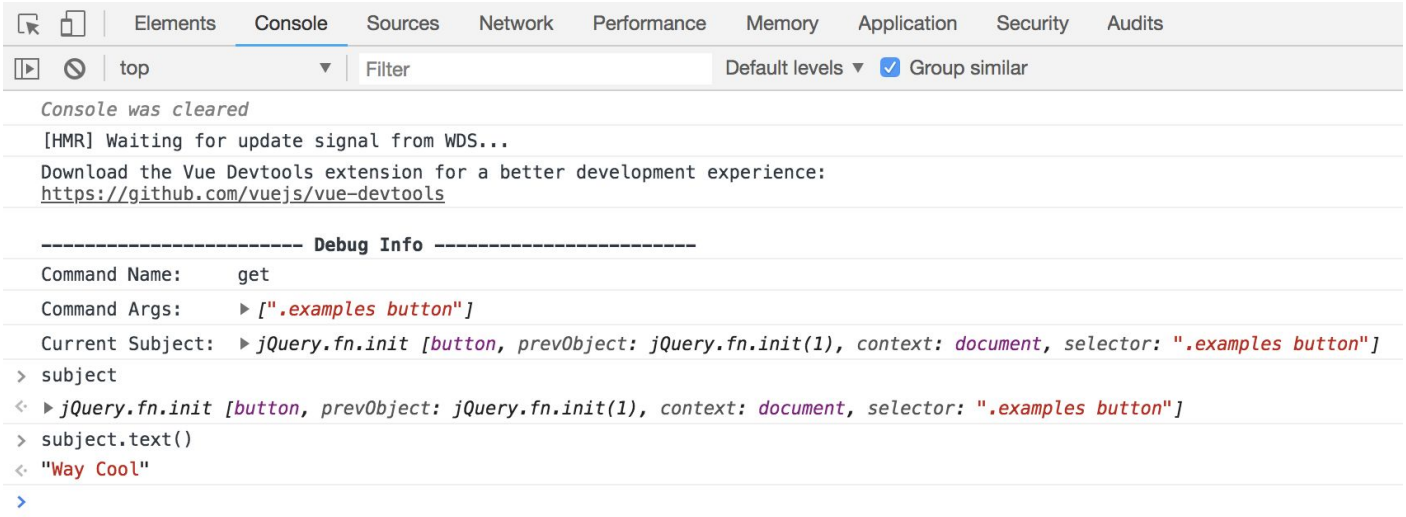
É possível debugar, e ao utilizar o debugger, no momento que o elemento do .then não for encontrado, o inspetor de código do navegador é aberto automaticamente

```
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
    })
})
```

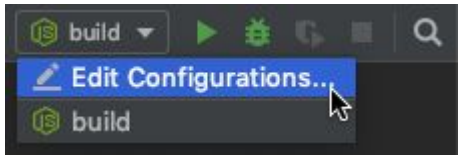
As diferenças

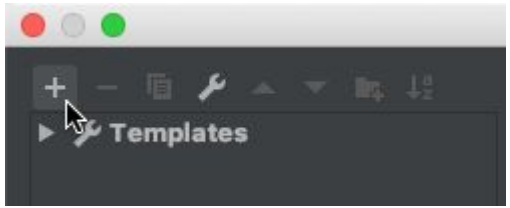
E utilizando o .debug, é possível interagir com os elementos pelo próprio console

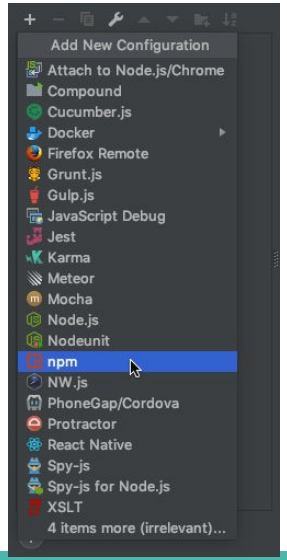


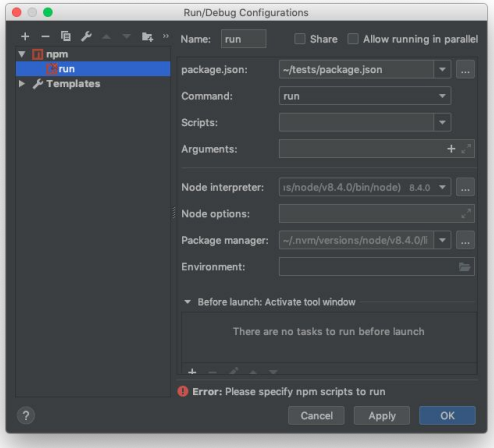
As diferenças

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.





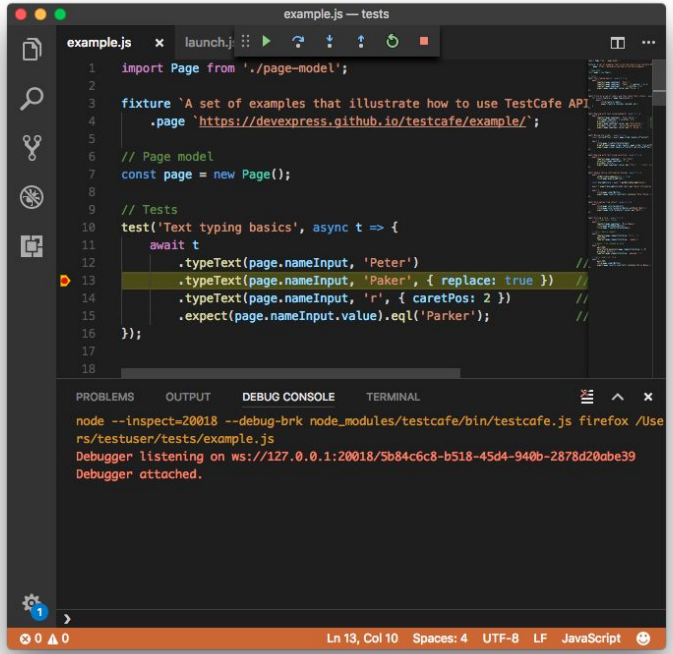




E por aí vai...

As diferenças

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



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

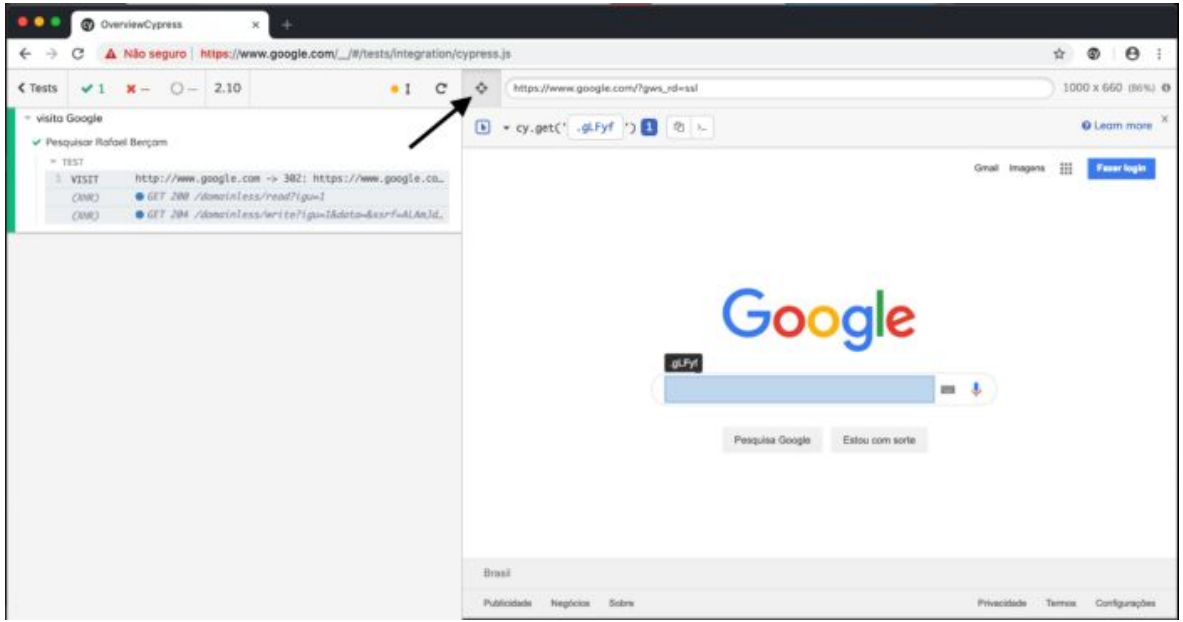
Cypress também permite personalizar e sobrescrever comandos que já existem, só alterar no arquivo cypress/support/commands.js

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!