

Cypress para tests E2E en
RoR

Sobre mí

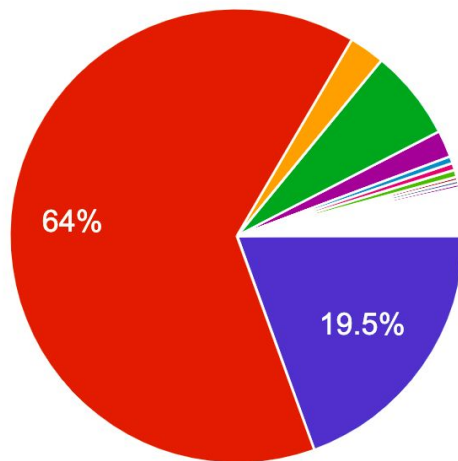
- Ingeniería UC, generación 2016
- Titulado el 2022 vía trabajo de título con temática QA
- MEng. de la École Centrale de Lyon (doble título, 2018 - 2020)
- Casi 2 años como ingeniero de software en Total Abogados + 10 meses como practicante full-time en ingeniería de software en Francia

¿Qué es Cypress?

- Herramienta para automatizar tests E2E y de integración
- 100% JavaScript
- Autocontenido, más que un framework apunta a ser un ecosistema completo

¿Cómo conocí a Cypress?

What e2e testing tools do you use?



- Protractor
- Cypress
- Puppeteer with a testing library
- Selenium WebDriver
- Webdriver.io
- TestCafe
- Testcafe
- Playwright








Aplicación a utilizar

Ejercicio práctico: integrar Cypress con RoR en 5 pasos

1. Ejecutar `yarn add --dev cypress` o `npm i cypress --save-dev` para instalar Cypress
2. Ejecutar su aplicación en RoR en ambiente de test
3. En otra consola, ejecutar `yarn run cypress open` o `npm run cypress open` para inicializar Cypress y configurarlo
4. Después de la configuración global, configurar la sección E2E
5. En el archivo `cypress.config.js` que se creó automáticamente, definir la variable `e2e.baseUrl`

1.

```
)$ yarn add --dev cypress  
yarn add v1.22.19  
info No lockfile found.  
[1/4] Resolving packages...  
[2/4] Fetching packages...  
[3/4] Linking dependencies...  
[4/4] Building fresh packages...  
success Saved lockfile.  
success Saved 145 new dependencies.  
info Direct dependencies  
└─ cypress@10.11.0  
info All dependencies
```

-  Gemfile
-  Gemfile.lock
-  Makefile
-  package.json
-  Rakefile
-  README.md
-  yarn.lock

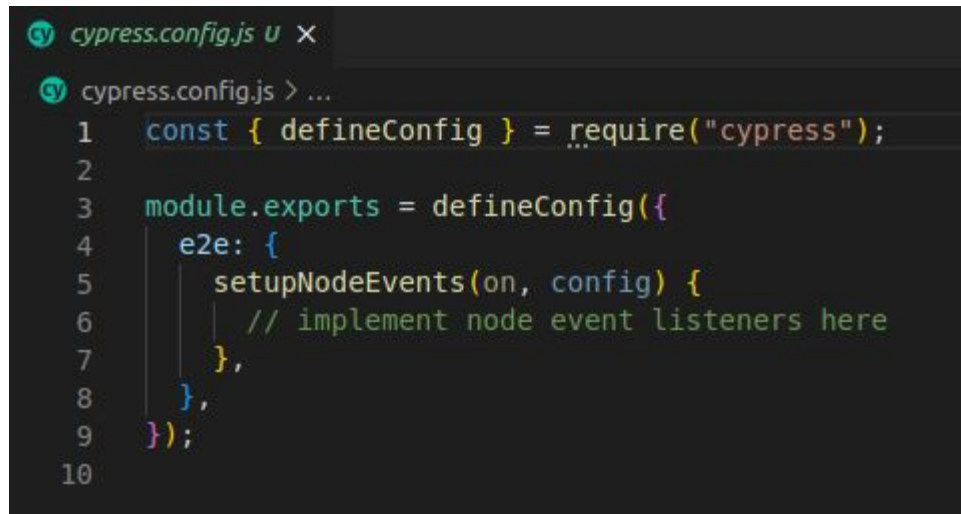
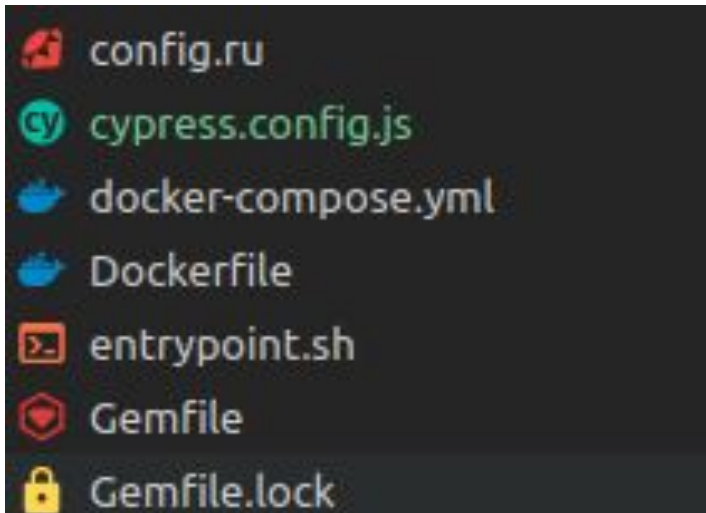
```
package.json U x
package.json > ...
1  {
2    "devDependencies": {
3      "cypress": "^10.11.0"
4    }
5  }
6
```


2.

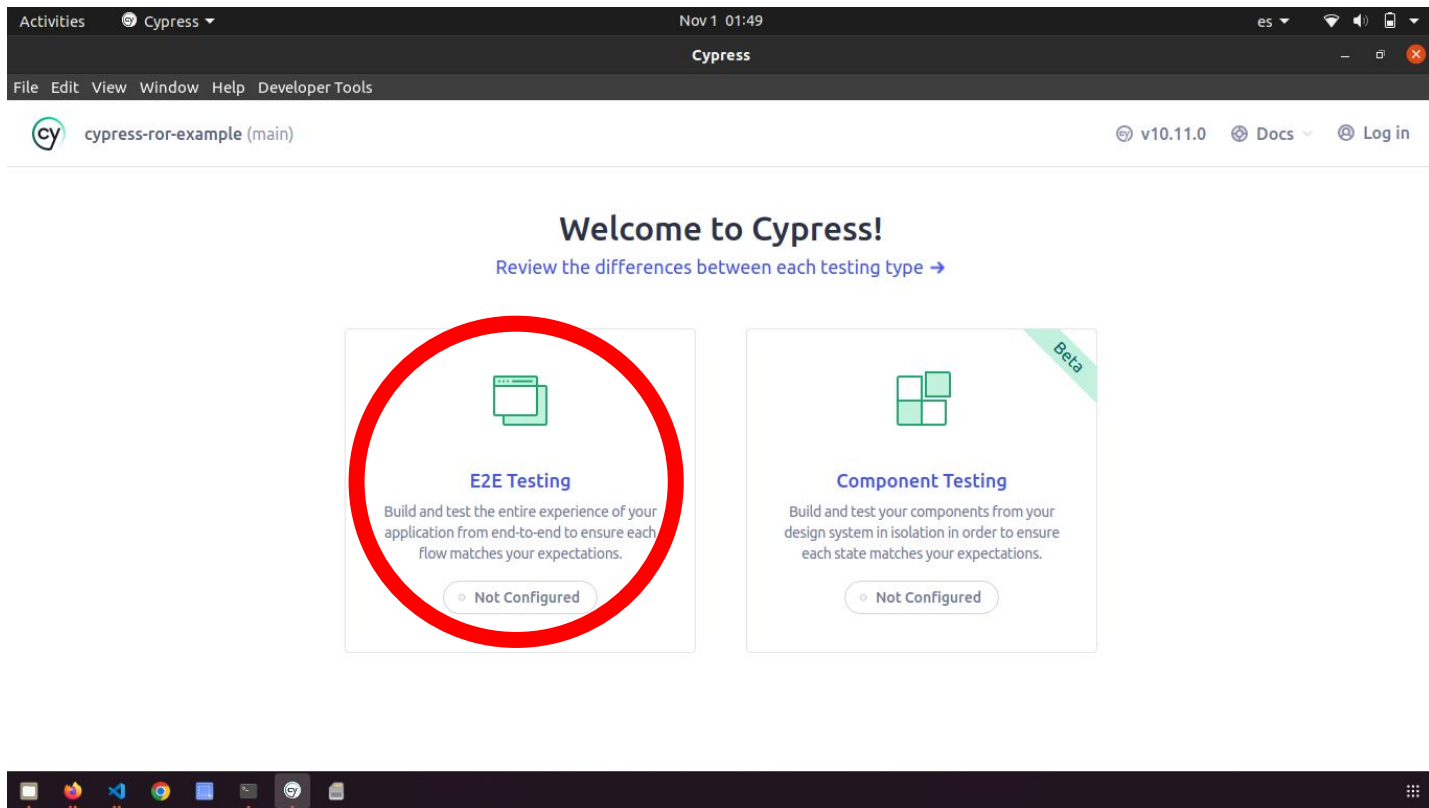
```
to accept connections
web_1 | => Booting Puma
web_1 | => Rails 7.0.4 application starting in test
web_1 | => Run `bin/rails server --help` for more startup options
web_1 | Puma starting in single mode...
web_1 | * Puma version: 5.6.5 (ruby 3.0.4-p208) ("Birdie's Version")
web_1 | *   Min threads: 5
web_1 | *   Max threads: 5
web_1 | *   Environment: test
web_1 | *             PID: 1
web_1 | * Listening on http://0.0.0.0:3002
web_1 | Use Ctrl-C to stop
```

3.

```
)$ yarn run cypress open  
yarn run v1.22.19  
warning package.json: No license field
```



4.





Configuration files

We added the following files to your project:



[cypress.config.js](#)

The Cypress config file for E2E testing.



[cypress/support/e2e.js](#)

The support file that is bundled and loaded before each E2E spec.



[cypress/support/commands.js](#)

A support file that is useful for creating custom Cypress commands and overwriting existing ones.



[cypress/fixtures/example.json](#)

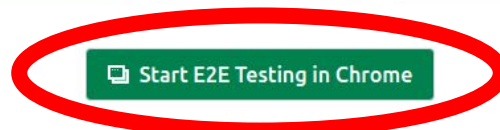
Added an example fixtures file/folder



Continue


Choose a browser


Choose your preferred browser for E2E testing.





← Switch testing type



Chrome is being controlled by automated test software. 

 cyress-ror-example
main

 Specs

 Runs

 Settings


 

Specs

v10.11.0 Chrome 107 Docs Log in


Create your first spec

Since this project looks new, we recommend that you use the specs and tests that we've written for you to get started.



Scaffold example specs

We'll generate several example specs to help guide you on how to write tests in Cypress.



Create new empty spec

We'll generate an empty spec file which can be used to start testing your application.

If you feel that you're seeing this screen in error, and there should be specs listed here, you likely need to update the spec pattern.

Chrome is being controlled by automated test software.

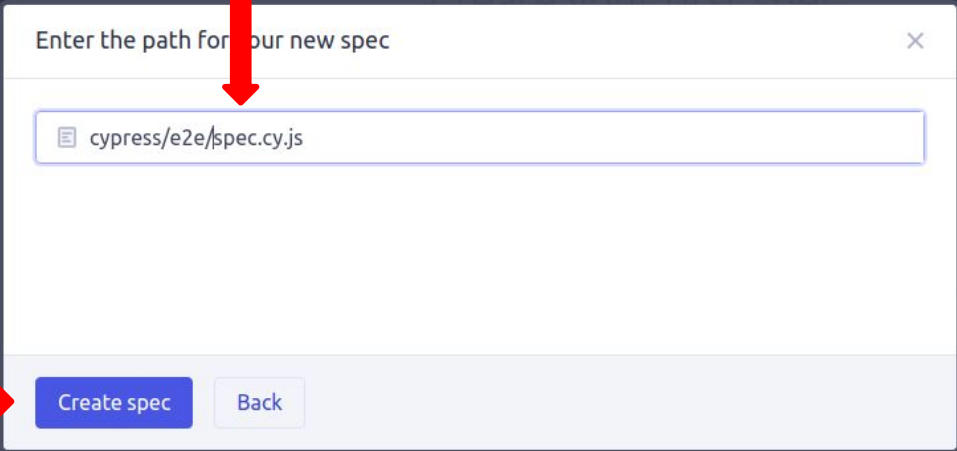
1

Enter the path for your new spec


cypress/e2e/spec.cy.js

2

Create spec Back

A screenshot of the Cypress application interface. A modal dialog box titled "Enter the path for your new spec" is centered on the screen. The dialog has a text input field containing the path "cypress/e2e/spec.cy.js". A red arrow labeled "1" points to this input field. At the bottom of the dialog, there are two buttons: "Create spec" (highlighted in blue) and "Back". A red arrow labeled "2" points to the "Create spec" button. The background shows the Cypress "Specs" view with a sidebar on the left and a main area displaying a spec tree.

X



A screenshot of a terminal window. A blue button with a white cursor icon and the text "Okay, run the spec" is circled in red.

Chrome is being controlled by automated test software.

Specs

spec cy.js 00:04

empty spec

passes

TEST BODY

visit https://example.cypress.io

https://example.cypress.io/

Chrome 107

1000x660 (74%)

cypress.io

Commands Utilities Cypress API

GitHub

Kitchen Sink

This is an example app used to showcase Cypress.io testing. For a full reference of our documentation, go to docs.cypress.io

Commands

Commands drive your tests in the browser like a real user would. They let you perform actions like typing, clicking, xhr requests, and can also assert things like "my button should be disabled".

Querying

get

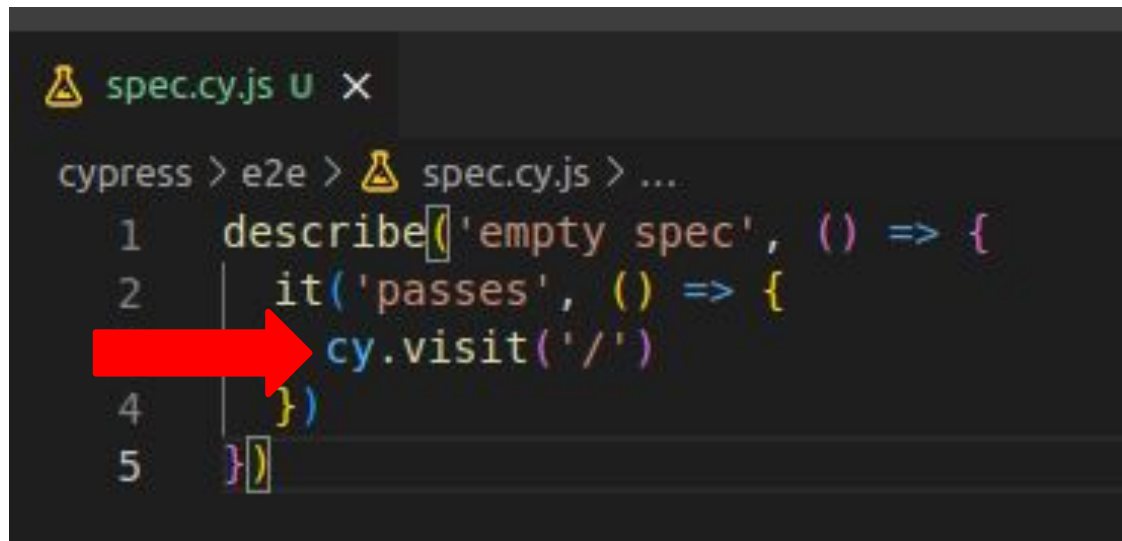
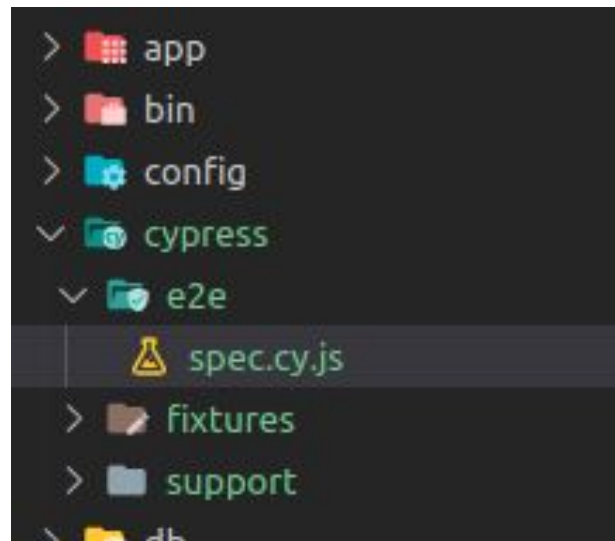
contains

within

root

5.

```
cy cypress.config.js u x
cy cypress.config.js > [e] <unknown> > 🔑 e2e
1  const { defineConfig } = require("cypress");
2
3  module.exports = defineConfig({
4    e2e: {
5      baseUrl: 'http://127.0.0.1:3001',
6      setupNodeEvents(on, config) {
7        // implement node event listeners here
8      },
9    },
10  });
11
```



×

The image shows a Cypress test runner interface on the left and a browser window on the right. The Cypress interface has a dark theme. At the top, it says 'Specs' with a green checkmark, a red 'x', and a refresh icon. Below that, the file 'spec.cy.js' is listed with a timer '00:04'. Under 'empty spec', there is a 'passes' section with a green checkmark and a 'TEST BODY' section containing a single command: 'visit /'. The browser window on the right shows the URL 'http://127.0.0.1:3001/'. The page content includes a 'Home' link, 'Log In' and 'Sign Up' links, a 'List of articles' section, a question '¿Será posible?' with a cartoon image of a man gaining weight, and a 'Festival de cine' section at the bottom.

Ejercicio práctico 2: escribir tests

1. Agregar seeds para usar en los tests
2. Comenzar a jugar con el archivo `cypress/e2e/${NAME}.cy.js`

1.

```
Article.destroy_all
User.destroy_all

User.create!({
  email: "nelson.haha@gmail.com",
  name: "Nelson Muntz",
  password: "colección-de-estampas-haha",
  username: "nmuntz"
})

Article.create!([
  {
    title: "Festival de cine",
    description: "Frame de mi cortometraje en el festival de Sundance",
    image_url: "https://i.ytimg.com/vi/qt65_GTEVBI/hqdefault.jpg"
  }, {
    title: "¿Será posible?",
    description: "",
    image_url: "https://preview.redd.it/yt1sqo8qls911.jpg?width=640&crop="
  }
])
```



spec.cy.js U X



Makefile M



seeds.rb M

cypress > e2e > spec.cy.js > ...

```
1 describe('Anonymous user visiting the site', () => {
2   beforeEach(() => {
3     cy.exec('make db-restart')
4       .its('code')
5       .should('eq', 0);
6   });
7
8   it('should be able to access the homepage', () => {
9     cy.visit('/');
10
11     // Navbar display right links
12     cy.contains('a', 'Log In');
13     cy.contains('a', 'Sign Up');
14     cy.contains('a', 'Home');
15
16     // Displays the list of articles
17     cy.contains('h1', 'List of articles');
18     cy.contains('h1', '¿Será posible?');
19     cy.contains('h1', 'Festival de cine');
20     cy.contains('p', 'Frame de mi cortometraje en el festival de Sundance');
21   });
22 });
23 |
```