

Tutorial 08 to do in class – there is no grade for this tutorial.

Antes de iniciar:

- Terminar los tutoriales anteriores.
- Este tutorial muestra cómo utilizar GitHub Actions, Laravel Pint y Cypress para llevar a cabo un mini proceso de integración continua en nuestros proyectos Laravel.

A. Laravel Pint y GitHub Actions

Sugerencia

Trabaje sobre el proyecto Laravel en el que ha venido realizando todos los tutoriales anteriores.

Creación de workflow `laravel-lint.yml`

- En la raíz de su proyecto cree el directorio `.github/` y dentro del directorio `.github/` cree el directorio `workflows/`
- Luego en `.github/workflows/` cree el archivo `laravel-lint.yml` con el siguiente contenido.

Add Code

```
name: Laravel-lint
```

```
on:
```

```
  push:
```

```
    branches: [ "main" ]
```

```
  pull_request:
```

```
    branches: [ "main" ]
```

```
jobs:
```

```
  laravel-lint:
```

```
    runs-on: ubuntu-latest
```

```
    steps:
```

```
      - name: Copy Project Code
```

```
        uses: actions/checkout@v3
```

```
      - name: Setup PHP
```

```
        uses: shivammathur/setup-php@v2
```

with:

php-version: '8.2'

- name: Copy .env

run: php -r "file_exists('.env') || copy('.env.example', '.env');"

- name: Install Dependencies

run: composer install -q --no-ansi --no-interaction --no-scripts --no-progress --prefer-dist

- name: Generate key

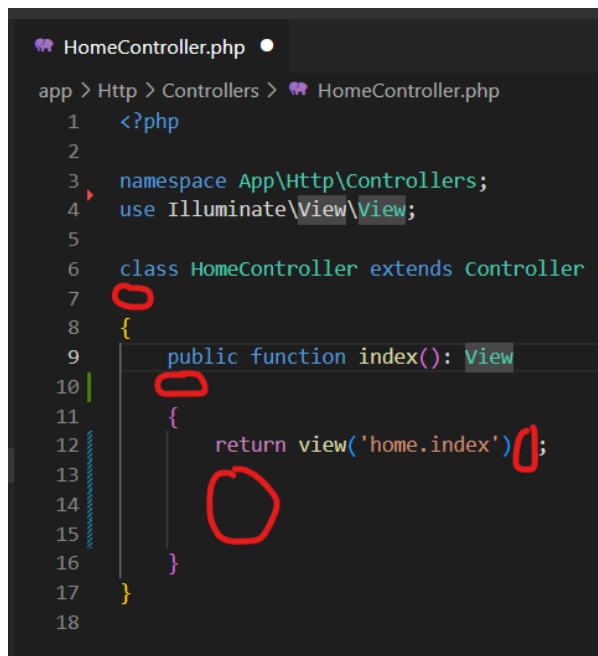
run: php artisan key:generate

- name: Execute Laravel Pint

run: vendor/bin/pint --test

Agregando errores

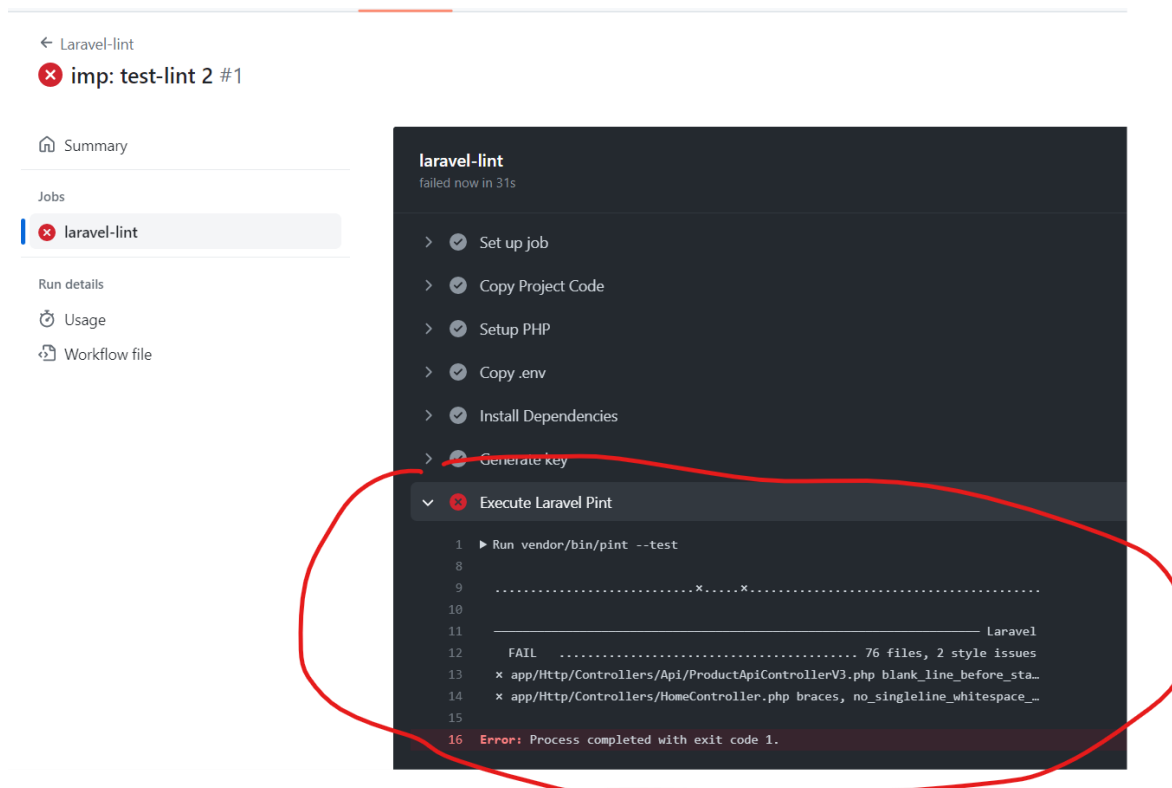
- Vaya a el archivo `app/Http/Controllers/HomeController.php` y agregue errores de estilo. Deje espacios en blancos, saltos de línea, entre otros.



```
app > Http > Controllers > HomeController.php
1  <?php
2
3  namespace App\Http\Controllers;
4  use Illuminate\View\View;
5
6  class HomeController extends Controller
7  {
8
9      public function index(): View
10     {
11         return view('home.index');
12     }
13 }
14
15
16
17
18
```

Enviando los nuevos cambios

- Suba los nuevos cambios a “GitHub”, espere un par de minutos, vaya a su repositorio y luego vaya a “Actions”, y deberá observar un Action que falló, similar a la siguiente imagen.



- El Action anterior indica errores de estilos, pero no los corrige ni nada.

Automatizando la ejecución de PINT

- En su repositorio de GitHub vaya a “Settings” -> “Actions” -> “General” -> “Workflow permissions” y marque la opción “Read and write permissions” y de click en “Save”

Workflow permissions

Choose the default permissions granted to the GITHUB_TOKEN when running workflows in this repository. You can specify more granular permissions in the workflow using YAML. [Learn more.](#)

☒ Read and write permissions

Workflows have read and write permissions in the repository for all scopes.

☐ Read repository contents and packages permissions

Workflows have read permissions in the repository for the contents and packages scopes only.

Choose whether GitHub Actions can create pull requests or submit approving pull request reviews.

☐ Allow GitHub Actions to create and approve pull requests

Save

- Modifique el archivo *laravel-lint.yml* con el siguiente contenido resaltado en negrilla (borre el `--test` y agregue las cinco líneas de código finales).

Modify Code

name: Laravel-lint

on:

push:

branches: ["main"]

pull_request:

branches: ["main"]

jobs:

laravel-lint:

runs-on: ubuntu-latest

steps:

- name: Copy Project Code

uses: actions/checkout@v3

- name: Setup PHP

uses: shivammathur/setup-php@v2

with:

php-version: '8.2'

- name: Copy .env

run: php -r "file_exists('.env') || copy('.env.example', '.env');"

- name: Install Dependencies

run: composer install -q --no-ansi --no-interaction --no-scripts --no-progress --prefer-dist

- name: Generate key

run: php artisan key:generate

- name: Execute Laravel Pint

run: vendor/bin/pint ~~--test~~

- name: Commit and push changes if Pint modified files

uses: stefanzweifel/git-auto-commit-action@v4


with:

commit_message: PHP Linting (Pint)

skip_fetch: true

- Haga un commit y push del nuevo código, y ahora podrá observar que un bot de GitHubActions le corrige automáticamente el código con los cambios realizados por Pint.

PHP Linting (Pint)

 main

 danielgara authored and github-actions[bot] committed now

```
▼ 9  app/Http/Controllers/HomeController.php 
```

...	...	@@ -1,18 +1,13 @@
1	1	<?php
2	2	
3	3	namespace App\Http\Controllers;
	4	+
4	5	use Illuminate\View\View;
5	6	
6	7	class HomeController extends Controller
7	-	-
8	8	{
9	9	public function index(): View
10	-	-
11	10	{
12	-	-
13	-	return view('home.index') ;
14	-	-
15	-	-
16	-	-
	11	+
		return view('home.index');
17	12	}
18	13	}

B.Cypress y GitHub Actions

Nota: debe tener instalado Node.js para poder hacer la sección de este tutorial.

Instalación de Cypress y creación de spec por defecto

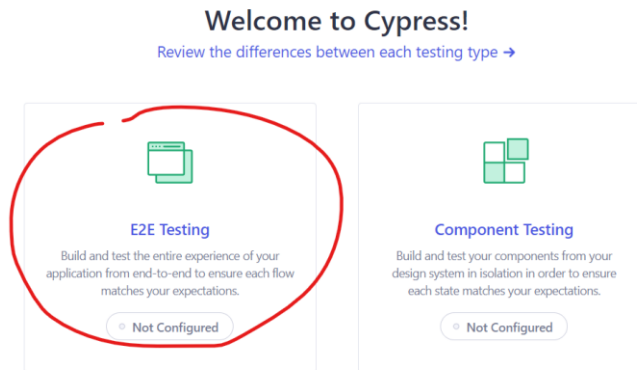
- En la raíz de su proyecto Laravel ejecute el siguiente comando:

```
npm install cypress --save-dev
```

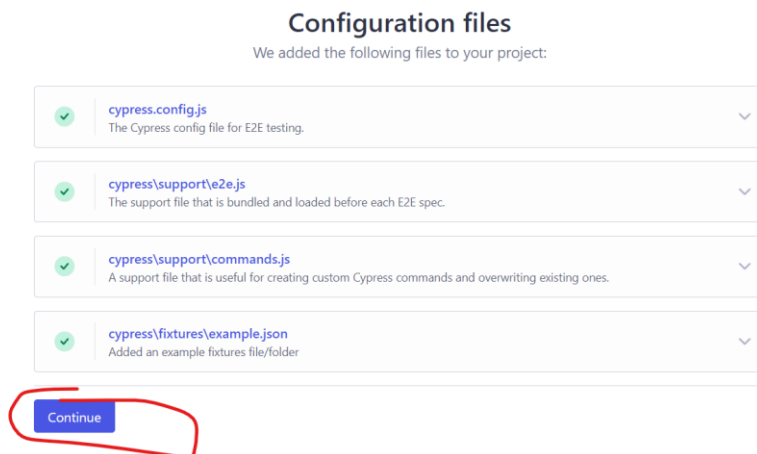
- Luego ejecute el siguiente comando para realizar la configuración inicial de cypress.

```
./node_modules/.bin/cypress open
```

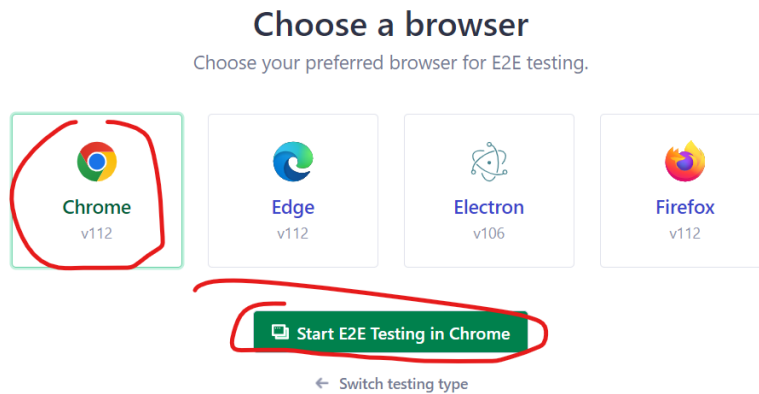
- Seleccione “E2E Testing”.



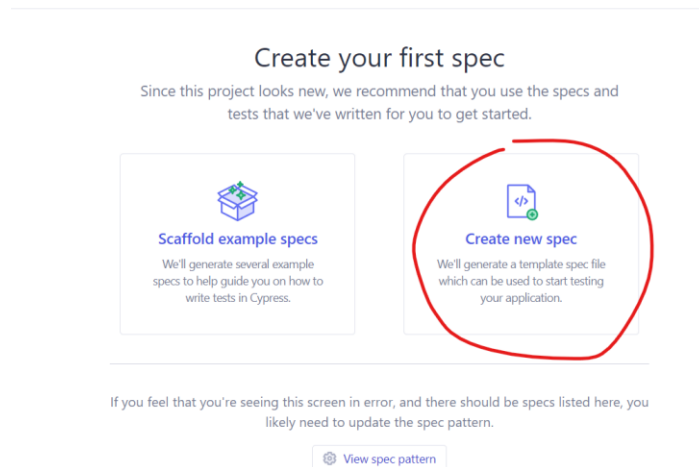
- Dele “Continue”



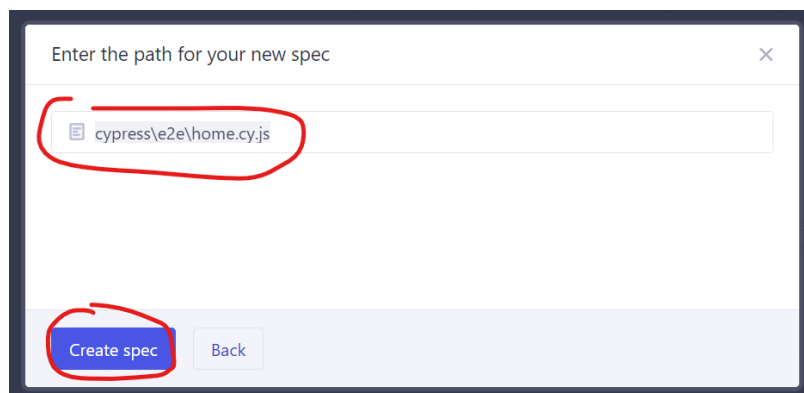
- Seleccione “Chrome” y seleccione “Start E2E Testing in Chrome”.



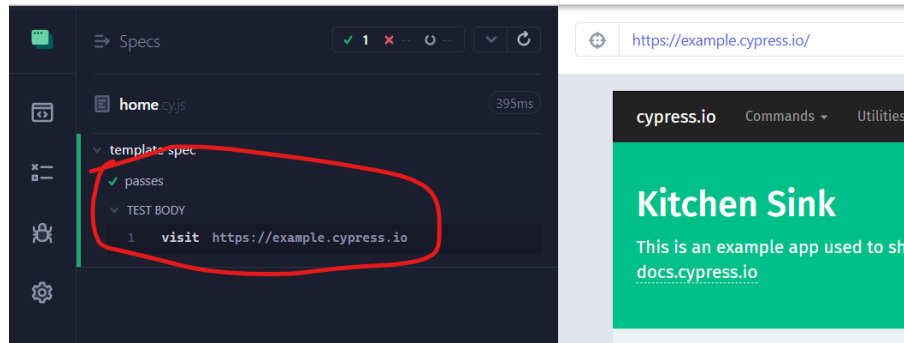
- Seleccione “Create new spec”



- Llamelo `cypress\e2e\home.cy.js` y de click en “Create spec”



- Corra el spec anterior y deberá salir un “chulito” en verde, que indica que la prueba end-to-end fue satisfactoria. En este caso, el código por defecto verificaba que se pudiera acceder a un sitio web de ejemplo de Cypress.



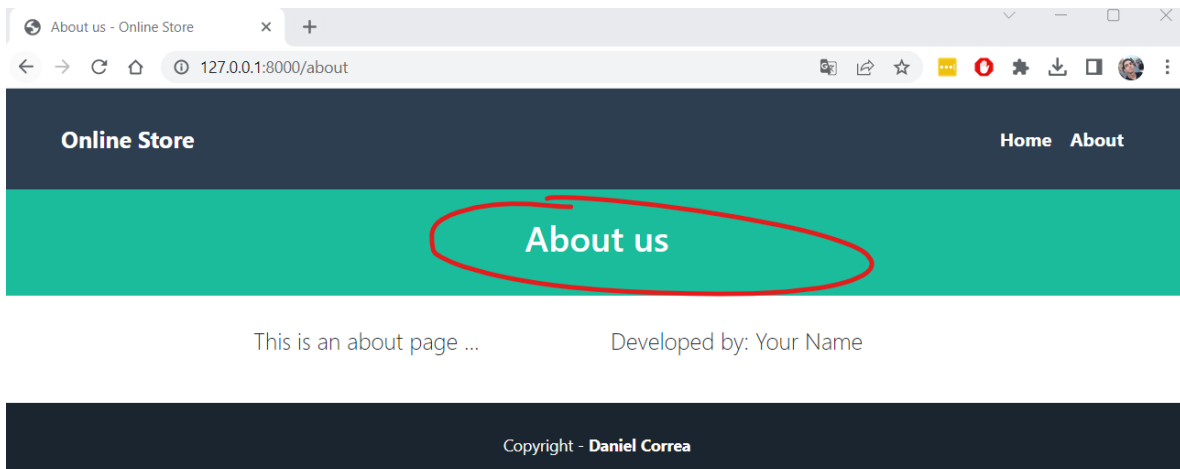
Modificando el spec home

- Reemplace el contenido del archivo `cypress\e2e\home.cy.js` con lo siguiente

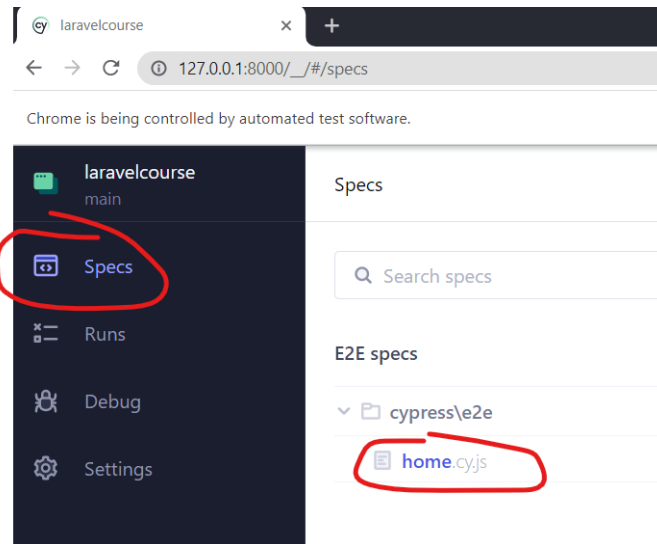
Replace Code

```
describe('Home Tests', () => {  
  it('About test message', () => {  
    cy.visit('http://127.0.0.1:8000/');  
    cy.get('a').contains('About').click();  
    cy.url().should('include', '/about');  
    cy.get('h2').should('contain', 'About us');  
  });  
});
```

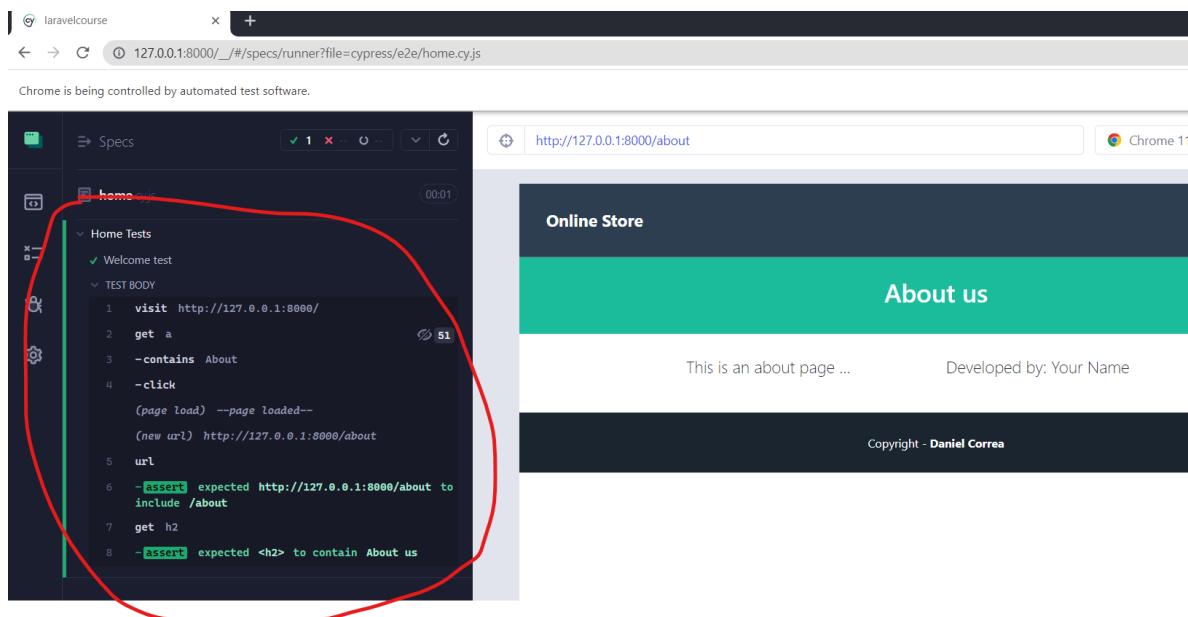
- **Corra en otra terminal el proyecto Laravel**, y verifique que la ruta `/about` tiene el siguiente texto resaltado en rojo.



- Si no lo tiene, agréguelo, que la prueba no funcionará.
- Vuelva a la ventana de Cypress -> Specs -> y corra nuevamente el "home".



- Esta vez la prueba se realizará sobre nuestro proyecto Laravel, y verificará una serie de cosas descritas en el spec (trate de entender el spec, y si no lo entiende, pregúntele al docente).



- Felicidades, ha ejecutado su primera prueba end-to-end sobre un proyecto Laravel. Ahora veamos como automatizar este proceso.

Creación de workflow laravel-e2e.yml

- En `.github/workflows/` cree el archivo `laravel-e2e.yml` con el siguiente contenido.

Replace Code

```
name: Laravel-e2e
```

on:

push:

branches: ["main"]

pull_request:

branches: ["main"]

jobs:

laravel-e2e:

runs-on: ubuntu-latest

steps:

- name: Copy Project Code

uses: actions/checkout@v3

- name: Setup PHP

uses: shivammathur/setup-php@v2

with:

php-version: '8.2'

- name: Copy .env

run: php -r "file_exists('.env') || copy('.env.example', '.env');"

- name: Install Dependencies

run: composer install -q --no-ansi --no-interaction --no-scripts --no-progress --prefer-dist

- name: Generate key

run: php artisan key:generate

- name: Use Node.js

uses: actions/setup-node@v1

with:

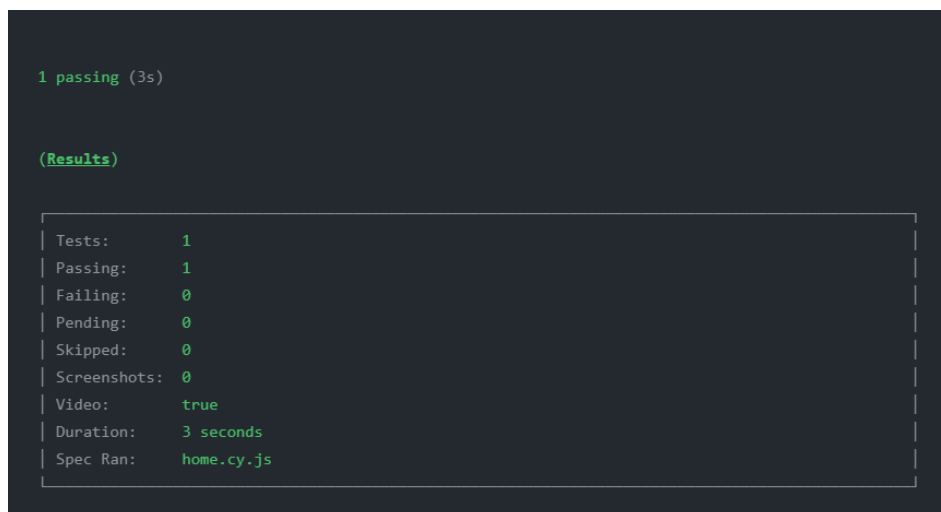
node-version: '16.x'

- name: Build site

run: npm install

```
- name: Cypress run
uses: cypress-io/github-action@v2
with:
  browser: chrome
  install: false
  start: php artisan serve
  wait-on: http://localhost:8000
  config-file: cypress.config.js
```

- El workflow anterior, instala y corre el proyecto laravel en una maquina Ubuntu, y luego ejecuta Cypress para correr los “specs”, en este caso solo la prueba sobre el home.
- Haga commit y push, y deberá observar que paralelo al proceso de laravel-lint, ahora se ejecuta un laravel-e2e que con cada cambio al main, ejecuta pruebas end-to-end.
- Si todo sale bien, deberá observar el Action en verde, y si ingresa a la ejecución de ese action, podrá ver la sección donde sale que la prueba del home fue satisfactoria.



```
1 passing (3s)

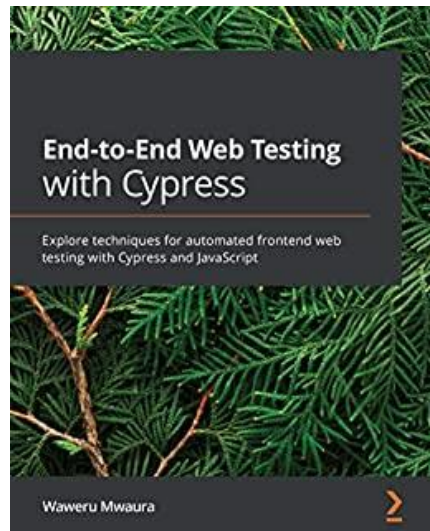
(Results)

| Tests:      1
| Passing:    1
| Failing:    0
| Pending:    0
| Skipped:    0
| Screenshots: 0
| Video:      true
| Duration:   3 seconds
| Spec Ran:   home.cy.js
```

Reto interesante

- 1) Debido a que en el entregable 2 usted debe implementar pruebas unitarias. trate de integrar las pruebas unitarias al proceso de integración continua mostrado en este tutorial y preséntelo en la entrega del entregable 2.
- 2) Si lo desea, trate de averiguar cómo ejecutar workflows que dependan de resultados de otros workflows. Por ejemplo, que primero se ejecute laravel-pint, si es satisfactorio, entonces que luego se ejecute laravel-unit-test y si ese es satisfactorio que luego se ejecute laravel-e2e.

¿Quiere aprender más sobre Cypress? Muy corto, muy bueno, de mis favoritos. Libro recomendado:



¿Quiere aprender más sobre GitHub Actions? Muy corto, muy bueno. Libro recomendado:

