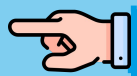
Pruebas Unitarias y end-to-end en un entorno Vue

Pruebas end-to-end

{desafío} latam_

Implementar pruebas unitarias utilizando las herramientas provistas por Vue para verificar el correcto funcionamiento del aplicativo.

- Unidad 1:
  Vue Router
- Unidad 2:
  Vuex
- Unidad 3:
  Firebase
- Unidad 4:
  Pruebas Unitarias y end-to-end en un entorno Vue



Te encuentras aquí

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Reconoce los conceptos y herramientas utilizados para la realización de pruebas end-to-end en un entorno Vue.*

¿Por qué es importante realizar pruebas en una aplicación web?

{desafío} latam_

Contexto antes de iniciar

En esta sesión estaremos revisando las pruebas end to end, estas al igual que las pruebas unitarias nos ayudarán a evaluar y verificar que nuestras aplicaciones cumplan con ciertos estándares de seguridad. Sin embargo, aunque el fin es el mismo que las pruebas unitarias, en las end to end tenemos algunas diferencias.

/* Características de las pruebas end to end */

Ventajas y limitaciones

Ventajas

- Simular interacciones de un usuario, desde la apertura del navegador hasta la ejecución de acciones como hacer clic en botones.
- Detección de problemas de integración, por ejemplo, en los componentes
- Verificar que toda la aplicación, incluyendo la interfaz de usuario y la lógica del servidor, funcionen de forma correcta.
- Ejecutar validaciones reales simulando el comportamiento de los usuarios.

Ventajas y limitaciones

Limitantes

- No son muy adecuadas para evaluar lógica.
- El costo de mantenimiento de este tipo de pruebas es mayor en cuanto a tiempo.
- Dentro del costo de mantenimiento se incluye la constante actualización y verificación de funcionalidades.
- Tienden a ser más complejas en configuración, dado que se deben considerar limitantes entre los navegadores web.
- Tienden a ser frágiles dado que están sujetas a los cambios que se generen en la interfaz.

Diferencia con pruebas unitarias

- A diferencia de las pruebas unitarias, las end to end nos permiten evaluar todo el flujo de una aplicación. Recordemos que el enfoque de las unitarias es evaluar idealmente cada pieza de código.
- Las pruebas end to end utilizan otras herramientas, en este caso utilizaremos Cypress. Acá no podemos utilizar Jest.
- Tienden a ser más lentas dado que simulan el comportamiento del usuario. A diferencia de las unitarias que podemos probar componentes aislados.

/* Setup de herramientas con create-vue */

Demostración: "Integrando pruebas end to end en Vue JS"

Integrando pruebas end to end en Vue JS

Contexto

A continuación, realizaremos la configuración inicial para ejecutar pruebas end to end en una aplicación Vue JS. Para ello, utilizaremos create-vue.

- **Paso 1:** Crear la aplicación con el nombre endtoend-test

  Usaremos el comando `npm create vue@latest`

Sigue los pasos

- **Paso 2:** En las opciones de configuración, en la sección de End-to-End Testing, seleccionaremos Cypress.

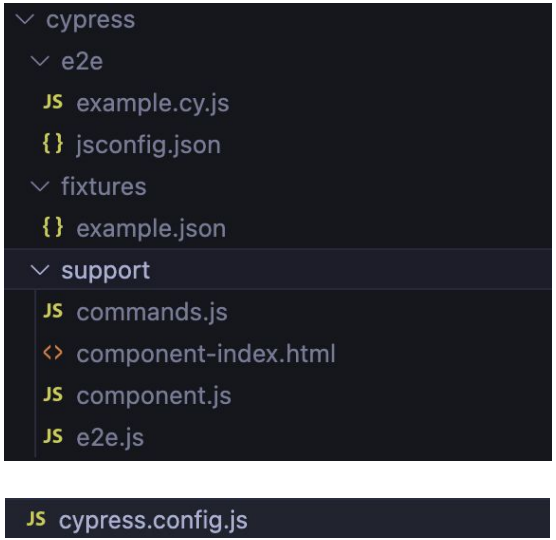
```
Vue.js - The Progressive JavaScript Framework

✔ Project name: … endtoend-test
✔ Add TypeScript? … No / Yes
✔ Add JSX Support? … No / Yes
✔ Add Vue Router for Single Page Application development? … No / Yes
✔ Add Pinia for state management? … No / Yes
✔ Add Vitest for Unit Testing? … No / Yes
? Add an End-to-End Testing Solution? › - Use arrow-keys. Return to submit.
    No
❯   Cypress - also supports unit testing with Cypress Component Testing
    Nightwatch
    Playwright
```

Sigue los pasos

- **Paso 3:** La configuración final debe quedar así.

```
Vue.js - The Progressive JavaScript Framework

✔ Project name: … endtoend-test
✔ Add TypeScript? … No / Yes
✔ Add JSX Support? … No / Yes
✔ Add Vue Router for Single Page Application development? … No / Yes
✔ Add Pinia for state management? … No / Yes
✔ Add Vitest for Unit Testing? … No / Yes
✔ Add an End-to-End Testing Solution? › Cypress
✔ Add ESLint for code quality? … No / Yes
✔ Add Prettier for code formatting? … No / Yes
✔ Add Vue DevTools 7 extension for debugging? (experimental) … No / Yes
```

Revisemos la estructura de carpetas y dependencias




```json
{
  "name": "endtoend-test",
  "version": "0.0.0",
  "private": true,
  "type": "module",
  Debug
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview",
    "test:e2e": "start-server-and-test preview http://localhost:4173 'cypress run --e2e'",
    "test:e2e:dev": "start-server-and-test 'vite dev --port 4173' http://localhost:4173 'cypress open
    "test:unit": "cypress run --component",
    "test:unit:dev": "cypress open --component",
    "lint": "eslint . --ext .vue,.js,.jsx,.cjs,.mjs --fix --ignore-path .gitignore",
    "format": "prettier --write src/"
  },
  "dependencies": {
    "vue": "^3.4.29"
  },
  "devDependencies": {
    "@rushstack/eslint-patch": "^1.8.0",
    "@vitejs/plugin-vue": "^5.0.5",
    "@vue/eslint-config-prettier": "^9.0.0",
    "cypress": "^13.12.0",
    "eslint": "^8.57.0",
    "eslint-plugin-cypress": "^3.3.0",
    "eslint-plugin-vue": "^9.23.0",
    "prettier": "^3.2.5",
    "start-server-and-test": "^2.0.4",
    "vite": "^5.3.1"
  }
}
```

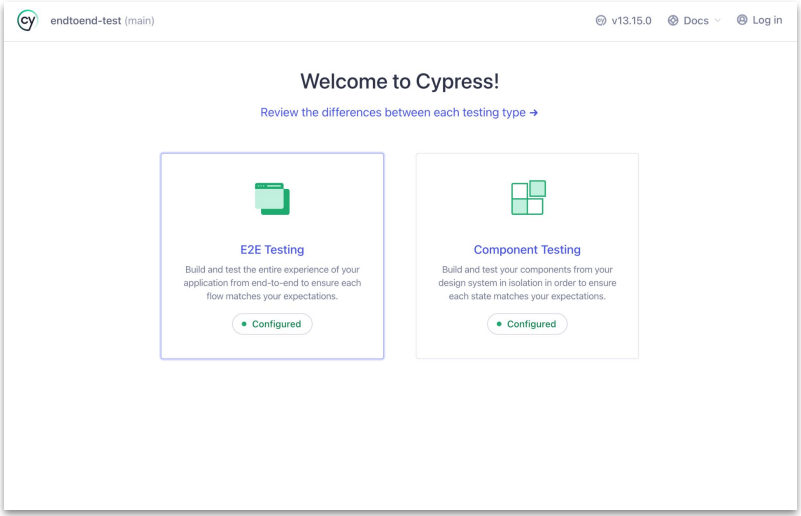
Levantando el servidor de pruebas

Al correr el comando npm run `test:e2e:dev` veremos el siguiente mensaje en la consola

```
> endtoend-test@0.0.0 test:e2e:dev
> start-server-and-test 'vite dev --port 4173' http://localhost:4173 'cypress open --e2e'

1: starting server using command "vite dev --port 4173"
and when url "[ 'http://localhost:4173' ]" is responding with HTTP status code 200
running tests using command "cypress open --e2e"


  VITE v5.4.8  ready in 262 ms

  ➜  Local:   http://localhost:4173/
  ➜  Network: use --host to expose

DevTools listening on ws://127.0.0.1:58781/devtools/browser/52104cd3-aa41-4da1-9f98-a8a5c9c59c2c
```

Configuración de Cypress

Luego de que Cypress se ejecute, veremos en su interfaz la siguiente pantalla y seleccionaremos E2E Testing.

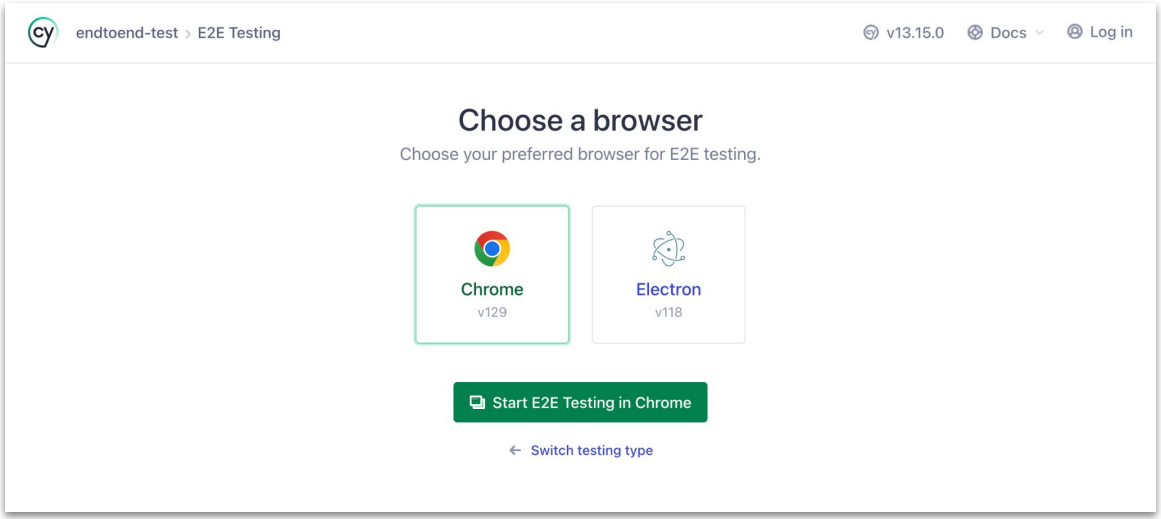


O podría venir ya seleccionado por defecto

{desafío} latam_

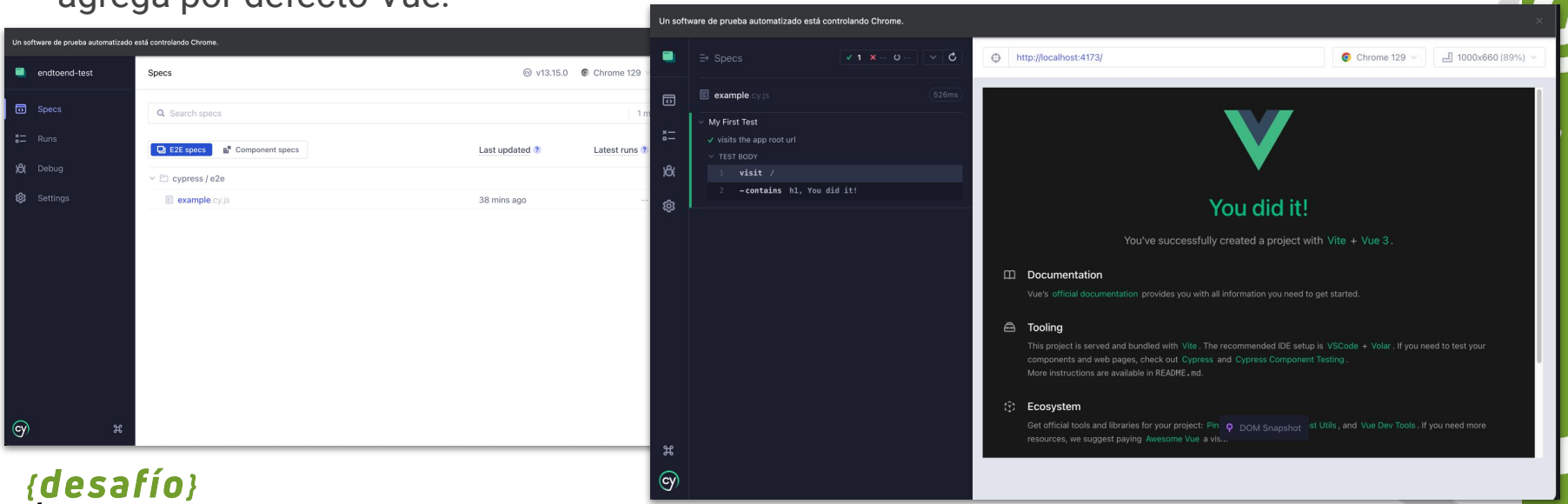
Configuración de Cypress

Ahora, seleccionamos el navegador en el cual se correrán las pruebas de componentes, en este caso utilizaremos chrome.



Configuración de Cypress

Ahora realizamos una prueba sobre uno de los componentes, en este caso, los que agrega por defecto Vue.



{desafío} latam_

Resumen de la sesión

- Con las pruebas E2E podemos simular interacciones de un usuario, desde la apertura del navegador hasta la ejecución de acciones como hacer clic en botones.
- Podemos verificar que toda la aplicación, incluyendo la interfaz de usuario y la lógica del servidor, funcionen de forma correcta
- La integración de Cypress tiende a ser lenta y compleja en una aplicación.

¿Existe alguna duda con los conceptos y temas vistos en la clase?

Próxima sesión...

- *Reconoce los conceptos y herramientas utilizados para la realización de pruebas end-to-end en un entorno Vue. (continuación)*

{desafío} latam_

{desafío}
latam_

Academia de
talentos digitales