

**INSTITUTO**  
**DEUSTO FORMACIÓN**

**CURSO FULL STACK  
DEVELOPMENT**

**Prueba 4. Testing.**  
**Jest y Cypress**

**ALUMNO:**

Armando Yamir Blanco Castro.

Matrícula: 9535801.

DNI: 12840810W

EMAIL: microcontrolador.net@gmail.com

**27 de Marzo de 2025, Tenerife, Granadilla de Abona.**

Metodología de resolución del examen:

Para la resolución del exámen se procedió a copiar las preguntas del ejercicio y copiar inmediatamente las resolución, para facilitar la corrección. Para cada pregunta, se cambió el formato de letras grande a número 20, y se colocó el fondo en amarillo:

**ejemplo.**

Los comandos que se solicitaron para los objetivos están enmarcados en fondo azul si los hay

**comando utilizados.**

Las referencias a otros tutoriales estarán demarcados en verde letra tamaño 20, por ejemplo:

**JavaScript - ASYNC / AWAIT - ¿Cómo usarlo?**

**JavaScript - FETCH - ¿Cómo utilizarlo?**

Se colocarán las respectivas capturas de pantallas de TODOS LOS PROCESOS.

## A continuación la solución del exámen:

NOMBRE DEL ARCHIVO: Prueba04-Testing-Ejercicio-1-001.odt

Para comprender el uso del testing en proyectos JavaScript y afianzar los conceptos de la metodología TDD, deberás llevar a cabo los siguientes pasos:

Instalación de la aplicación cypress. Para la resolución del presente examen se procedió a instalar la aplicación cypress desde esta ubicación de internet:

**<https://www.cypress.io/>**

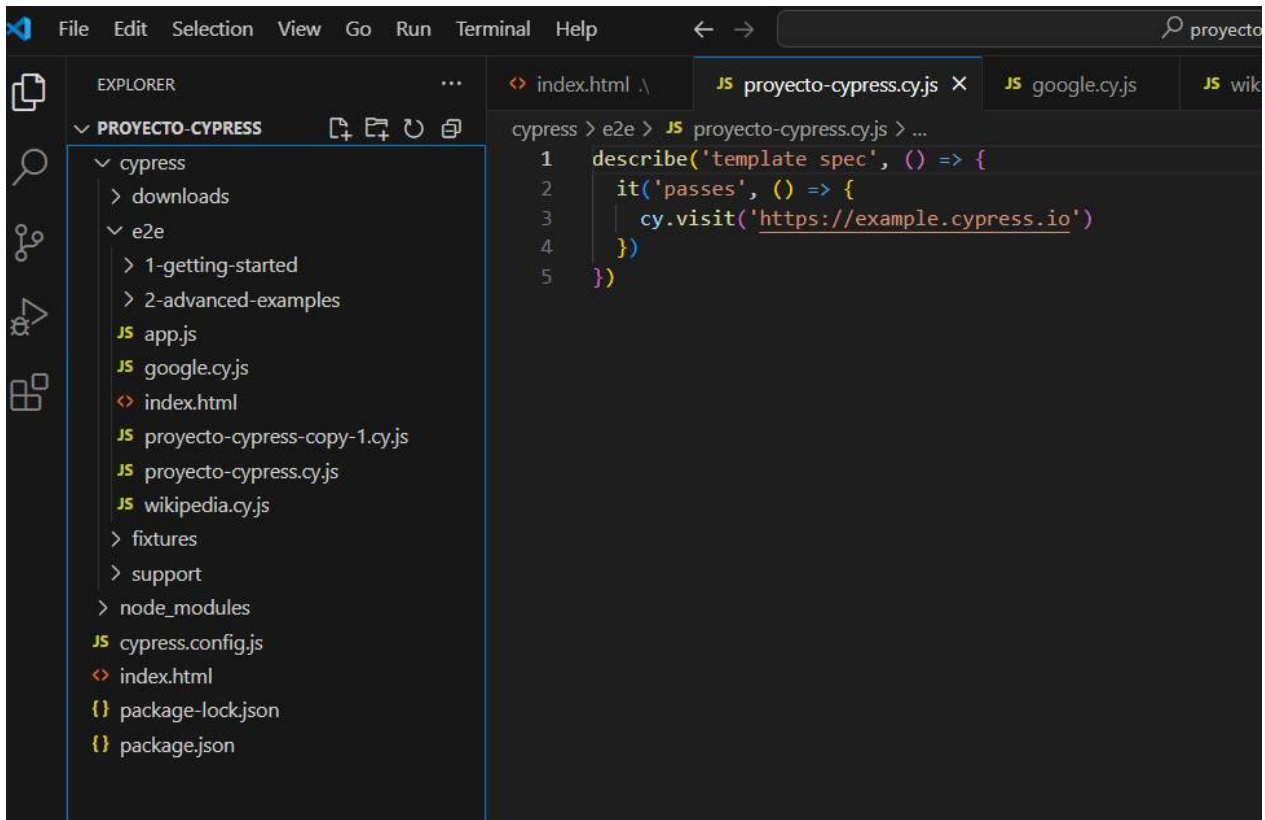
y se utilizó la documentación de cypress en esta url:

**<https://docs.cypress.io/app/get-started/why-cypress>**

al final de este documento están los detalles de la instalación y puesta en marcha con un ejemplo así como los tutoriales aportados como referencias para llevar a cabo esta asignación, en section MISCELANEOS.

- Creación de un proyecto HTML + JavaScript de registro de una cuenta de usuario.

Para ello abrimos nuestro visual code en el directorio donde está ubicado el cypress, e2e y creamos la carpeta donde colocaremos nuestros códigos, los archivos vacíos al principio:



\* Siguiendo la metodología TDD, desarrollar un test en el framework Cypress para comprobar en varios use cases los requisitos del componente.

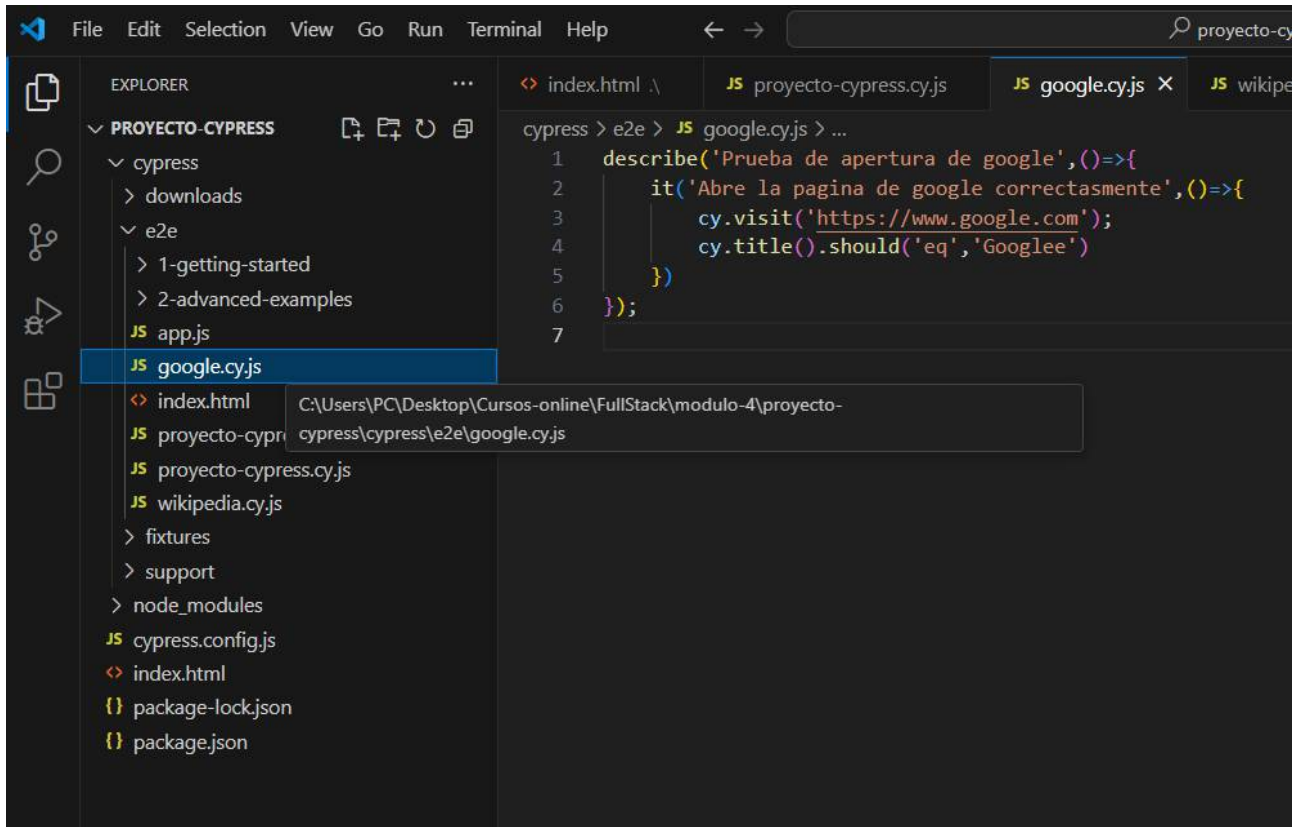
Para comprobar la metodología TDD lo haremos haciendo un test sencillo de apertura de la página web de google, veamos:

creamos un archivo google.cy.js y escribimos el siguiente código para hacer la prueba:

```
describe('Prueba de apertura de google', () => {
  it('Abre la pagina de google correctamente', () => {
    cy.visit('https://www.google.com');
  });
});
```

```
cy.title().should('eq', 'Googlee')
})
});
```

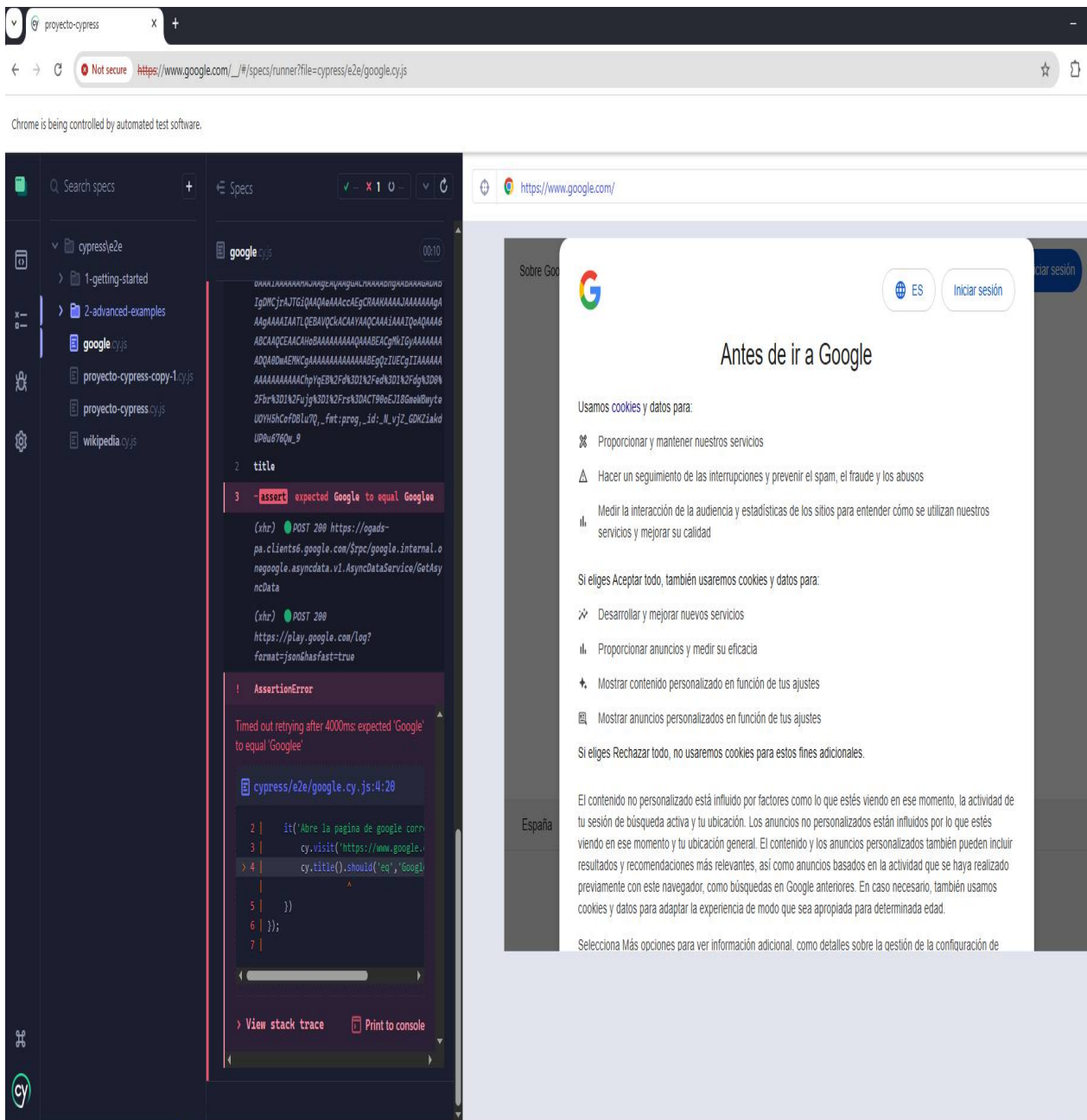
lo vemos en visual code de esta manera:



note la siguiente linea:

```
cy.title().should('eq', 'Googlee')
```

aqui colocamos un error para ver la respuesta de nuestro testing TDD, corremos el test y vemos los resultados:

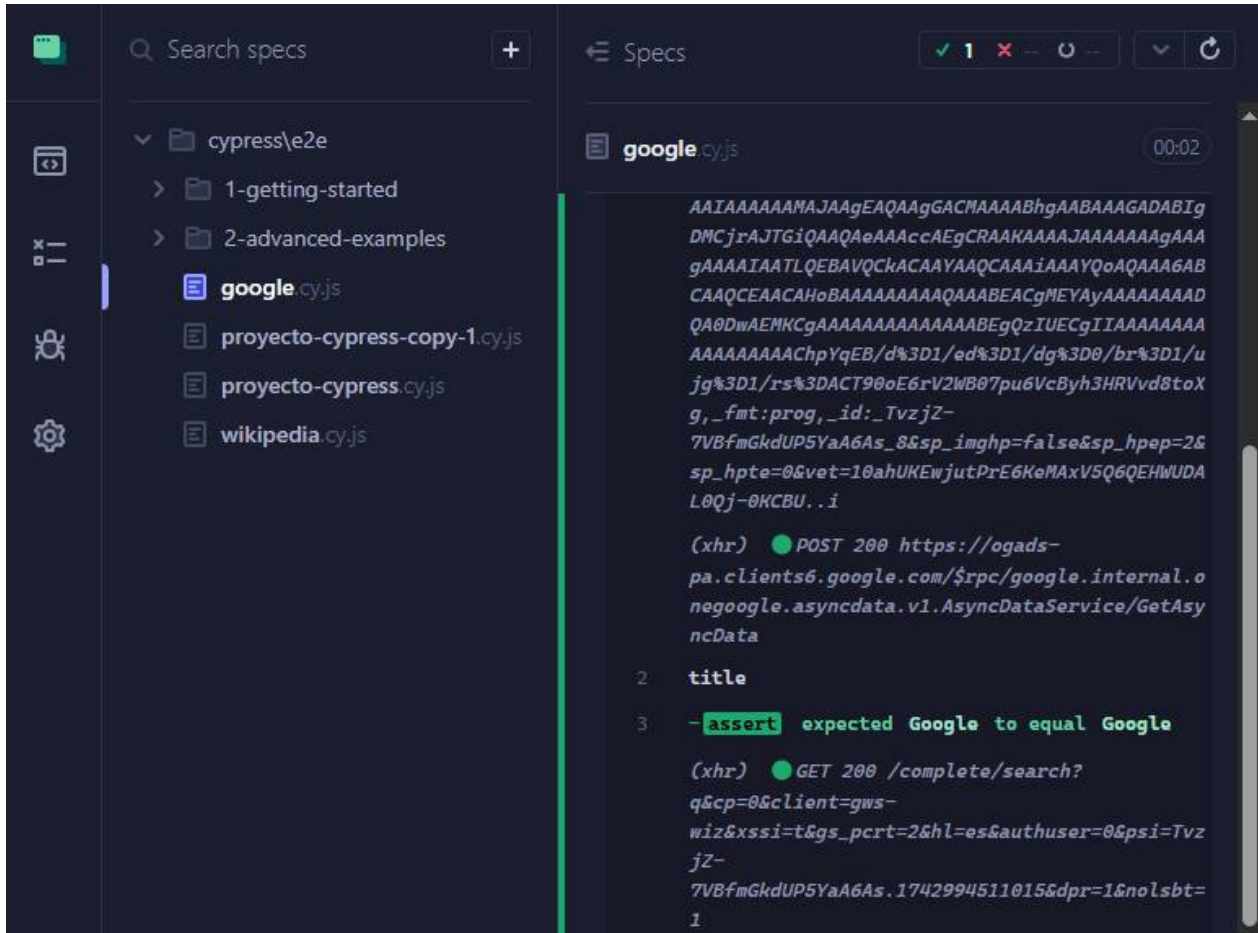


tenemos un error tipo

-assert expected Google to equal Googlee

donde se espera que la respuesta tenga la etiqueta Googlee

Si cambiamos de nuevo "Googlee" a "Google" lo escrito vemos que la corrida es satisfactoria:



-assert expected Google to equal Google

- Desarrollo de la vista en HTML con un formulario que contenga campos de correo electrónico, contraseña y repetición de la contraseña.

Para ello desarrollamos el siguiente código HTML en el IDE de visual code:

archivo: index.html

```
<!DOCTYPE html>

<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<title>Registro de Usuario</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
  <h2>Registro de Cuenta</h2>
  <form id="registroForm">
    <label for="email">Correo Electrónico:</label>
    <input type="email" id="email" required><br>

    <label for="password">Contraseña:</label>
    <input type="password" id="password" required><br>

    <label for="confirmPassword">Repetir Contraseña:</label>
    <input type="password" id="confirmPassword" required><br>

    <button type="submit">Registrar</button>
  </form>

  <p id="mensaje"></p>

  <script src="app.js"></script>
</body>
</html>

```

y el archivo javascript

app.js

```

document.getElementById("registroForm").addEventListener("submit", function (event) {
  event.preventDefault();

  const email = document.getElementById("email").value.trim();
  const password = document.getElementById("password").value.trim();
  const confirmPassword = document.getElementById("confirmPassword").value.trim();
  const mensaje = document.getElementById("mensaje");

  if (email === "" || password === "" || confirmPassword === "") {
    mensaje.innerText = "Todos los campos son obligatorios.";
    return;
  }

  if (password.length < 6) {
    mensaje.innerText = "La contraseña debe tener al menos 6 caracteres.";
    return;
  }

  if (password !== confirmPassword) {
    mensaje.innerText = "Las contraseñas no coinciden.";
    return;
  }

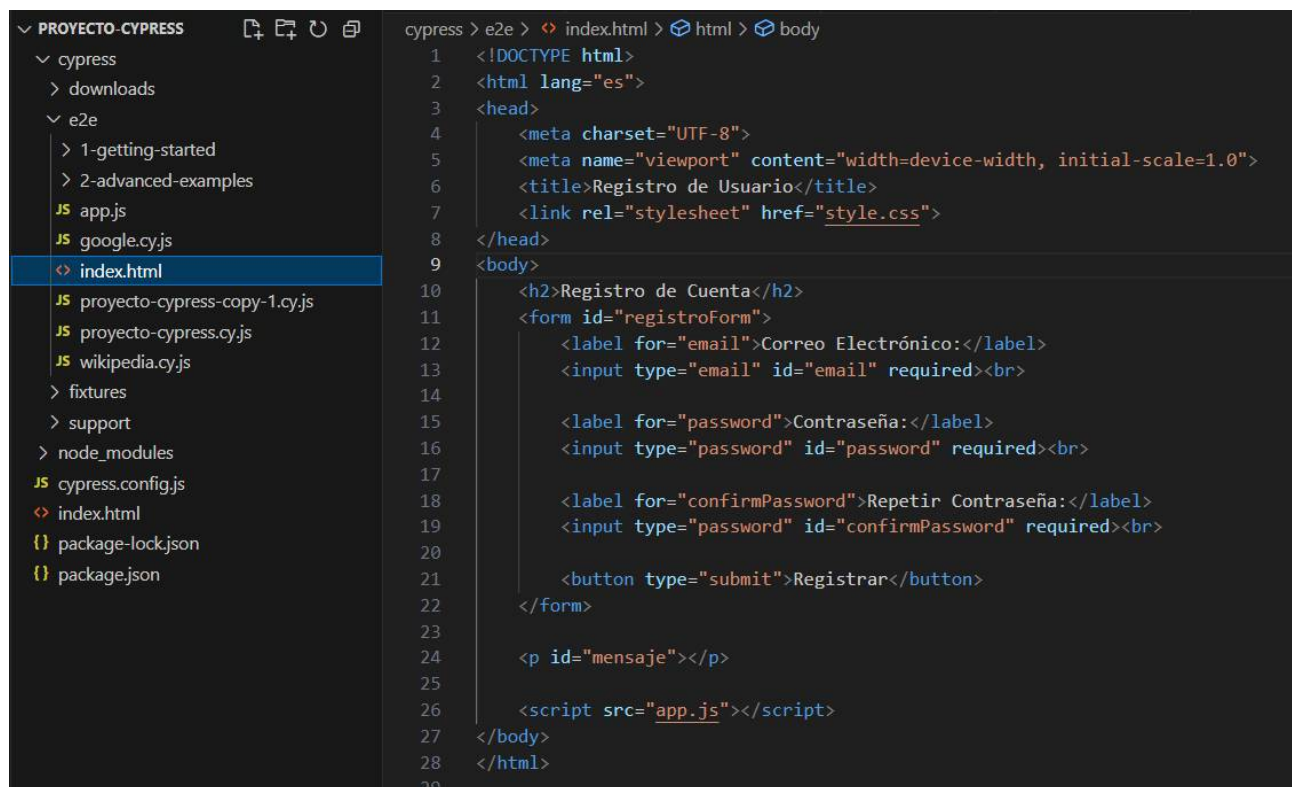
```



```
mensaje.innerText = "Registro exitoso.";
});
```

captura de pantallas archivos html y js

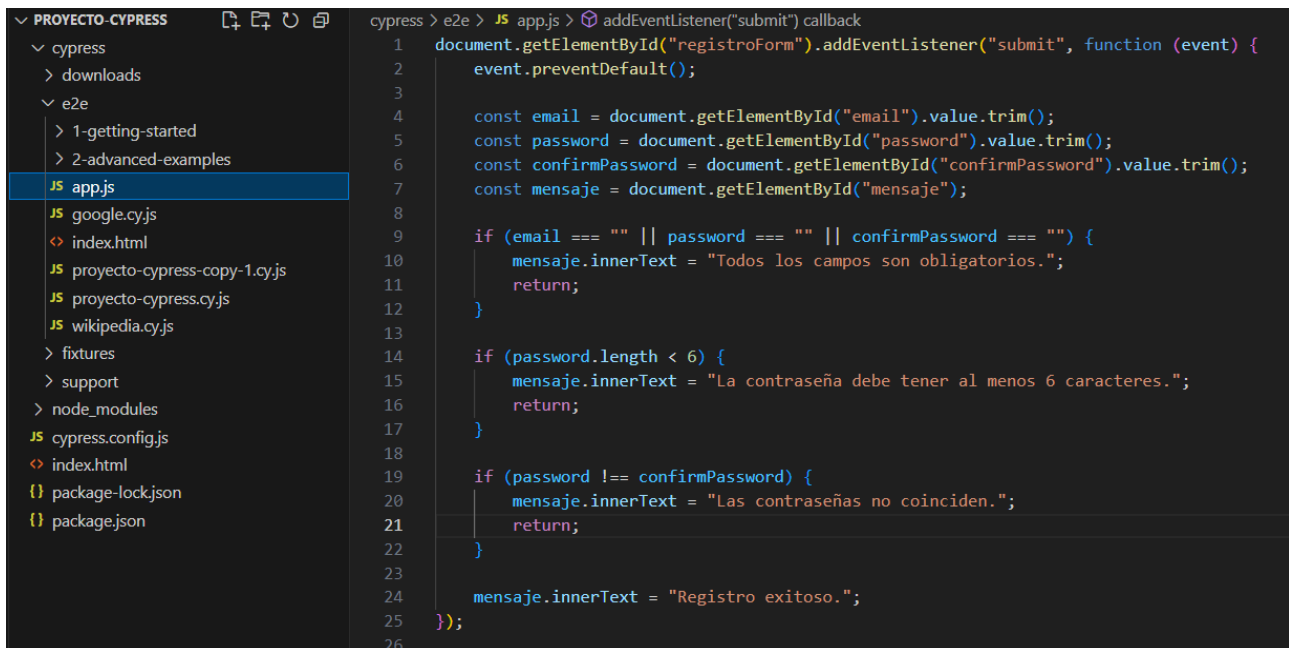
index.html



```
cyress > e2e > index.html > html > body
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Registro de Usuario</title>
7    <link rel="stylesheet" href="style.css">
8  </head>
9  <body>
10   <h2>Registro de Cuenta</h2>
11   <form id="registroForm">
12     <label for="email">Correo Electrónico:</label>
13     <input type="email" id="email" required><br>
14
15     <label for="password">Contraseña:</label>
16     <input type="password" id="password" required><br>
17
18     <label for="confirmPassword">Repetir Contraseña:</label>
19     <input type="password" id="confirmPassword" required><br>
20
21     <button type="submit">Registrar</button>
22   </form>
23
24   <p id="mensaje"></p>
25
26   <script src="app.js"></script>
27 </body>
28 </html>
29
```

app.js





```
cyress > e2e > JS app.js > addEventListener("submit") callback
1 document.getElementById("registroForm").addEventListener("submit", function (event) {
2   event.preventDefault();
3
4   const email = document.getElementById("email").value.trim();
5   const password = document.getElementById("password").value.trim();
6   const confirmPassword = document.getElementById("confirmPassword").value.trim();
7   const mensaje = document.getElementById("mensaje");
8
9   if (email === "" || password === "" || confirmPassword === "") {
10     mensaje.innerText = "Todos los campos son obligatorios.";
11     return;
12   }
13
14   if (password.length < 6) {
15     mensaje.innerText = "La contraseña debe tener al menos 6 caracteres.";
16     return;
17   }
18
19   if (password !== confirmPassword) {
20     mensaje.innerText = "Las contraseñas no coinciden.";
21     return;
22   }
23
24   mensaje.innerText = "Registro exitoso.";
25 });
26
```

- Desarrollar los pasos siguiendo el ciclo TDD para implementar cada característica.

### Resumen del ciclo TDD

**Rojo:** Escribir un test que falle porque la funcionalidad aún no existe.

**Verde:** Escribir el código mínimo para que el test pase.

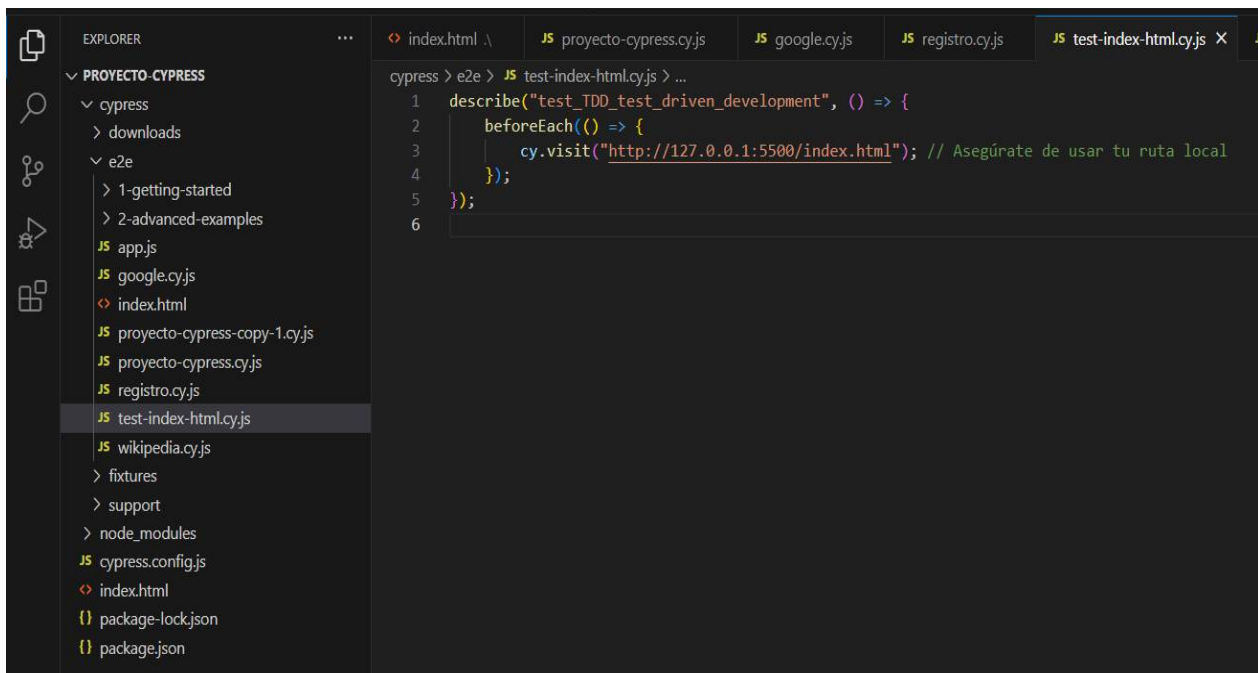
**Refactor:** Mejorar el código sin cambiar su comportamiento.

Vamos a crear al archivo de test que llamaremos test-index-html.cy.js con los elementos básicos para enlazar cypress con el navegador, aquí tenemos el código:

archivo: test-index-html.cy.js

```
describe("test_TDD_test_driven_development", () => {
  beforeEach(() => {
    cy.visit("http://127.0.0.1:5500/index.html"); // Asegúrate de usar tu ruta local
  });
});
```

Captura en visual code:

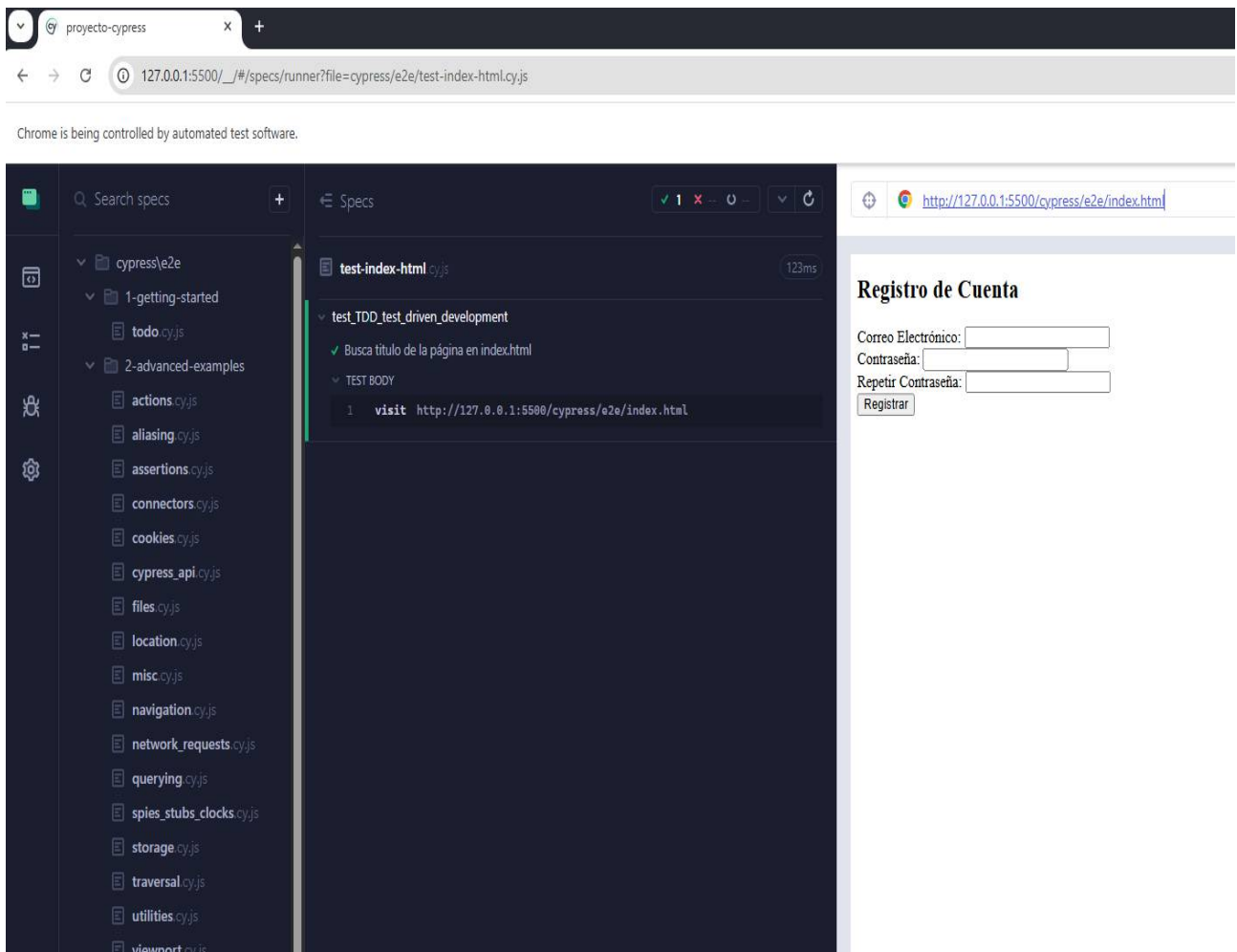


recuerda usar tu puerto para el navegador para que cypress pueda conseguir el archivo a testear.

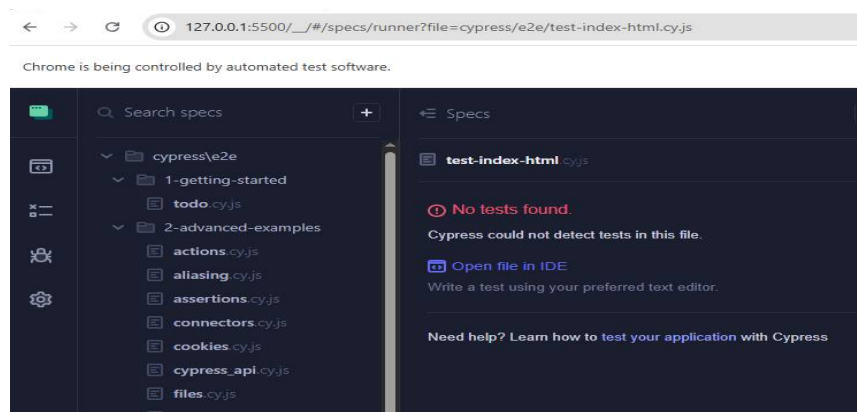
En este caso el puerto es <http://127.0.0.1:5500> (una dirección local) NOTA: AQUÍ PODEMOS UTILIZAR LA DIRECCIÓN URL PARA HACER UNA PRUEBA EN CUALQUIER SITIO DE INTERNET QUE QUERAMOS TESTEAR, CUANDO ESTEMOS DESARROLLANDO NUESTRO SITIO WEB)

si no funciona, copiamos la dirección directa que nos da open with live server de visual code, la url, en este caso <http://127.0.0.1:5500/cypress/e2e/index.html>

y en la página cypress debemos ver el programa index.html corriendo, veamos:



**Corremos el primer paso TDD:** que falle porque la funcionalidad aún no existe.



El código es el siguiente:

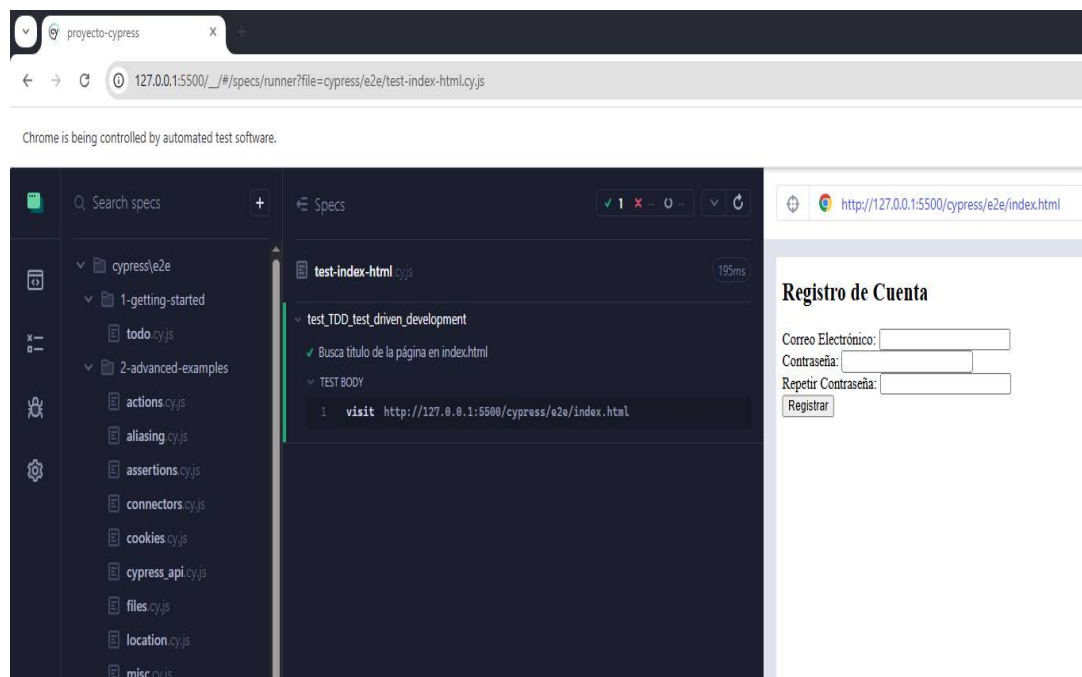
```
describe("test_TDD_test_driven_development", () => {

});
```

**Verde:** Escribir el código mínimo para que el test pase.

```
describe("test_TDD_test_driven_development", () => {
  it('Busca titulo de la página en index.html', ()=>{
    // cy.visit("http://127.0.0.1:5500/index.html"); // Asegúrate
    // de usar tu ruta local
    cy.visit('http://127.0.0.1:5500/cypress/e2e/index.html'); // sino utiliza la de
    cypress
  })
});
```

y la captura de pantalla para el correspondiente código:



- Desarrollo de la lógica en JavaScript para el cumplimiento del test en cada iteración.

Para lograr esta pregunta vamos a utilizar nuestro código en javascript para cumplir cada iteración. Tenemos nuestro archivo registro.cy.js cypress para validar los datos como sigue:

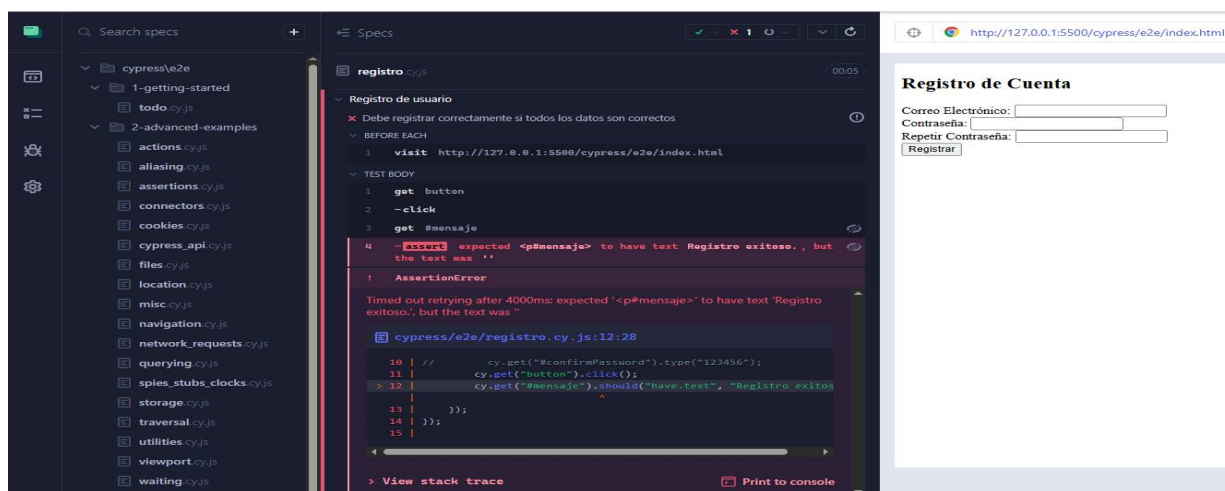
```
describe("Registro de usuario", () => {  
  beforeEach(() => {  
    cy.visit("http://127.0.0.1:5500/index.html"); // Asegúrate de usar tu ruta local  
  });  
  
  it("Debe mostrar error si los campos están vacíos", () => {  
    cy.get("button").click();  
    cy.get("#mensaje").should("have.text", "Todos los campos son obligatorios.");  
  });  
  
  it("Debe mostrar error si la contraseña tiene menos de 6 caracteres", () => {  
    cy.get("#email").type("test@example.com");  
    cy.get("#password").type("12345");  
    cy.get("#confirmPassword").type("12345");  
    cy.get("button").click();  
    cy.get("#mensaje").should("have.text", "La contraseña debe tener al menos 6 caracteres.");  
  });  
  
  it("Debe mostrar error si las contraseñas no coinciden", () => {  
    cy.get("#email").type("test@example.com");  
    cy.get("#password").type("123456");  
    cy.get("#confirmPassword").type("abcdef");  
    cy.get("button").click();  
    cy.get("#mensaje").should("have.text", "Las contraseñas no coinciden.");  
  });  
  
  it("Debe registrar correctamente si todos los datos son correctos", () => {  
    cy.get("#email").type("test@example.com");  
    cy.get("#password").type("123456");  
    cy.get("#confirmPassword").type("123456");  
    cy.get("button").click();  
    cy.get("#mensaje").should("have.text", "Registro exitoso.");  
  });  
});
```

NOTA IMPORTANTE: RECUERDE QUE PARA QUE CYPRESS PUEDA REALIZAR EL TEST DEBE ESTAR CORRIENDO EL ARCHIVO HTML EN LA VENTANA LIVE SERVER PARA QUE TOME ESTA COMO FOCO DEL TEST, O SINO UTILIZAR UN SERVIDOR APACHE EN EL MISMO TERMINAL COMO EL XAMP. TAMBIÉN TENIENDO LOS ARCHIVOS EN UN HOSTING CON UN DOMINIO Y PASARLE LA URL.

Iteración 1: Campos vacíos (Archivo: registro.cy.js).

```
describe("Registro de usuario", () => {  
  beforeEach(() => {  
    // cy.visit("http://127.0.0.1:5500/index.html"); // Asegúrate de usar tu  
    ruta local  
    cy.visit("http://127.0.0.1:5500/cypress/e2e/index.html"); // Asegúrate de usar  
    tu ruta local  
  });  
  
  it("Debe registrar correctamente si todos los datos son correctos", () => {  
    cy.get("button").click();  
    cy.get("#mensaje").should("have.text", "Registro exitoso.");  
  });  
});
```

al salvar el código en visual code, inmediatamente cypress corre el test. Dando el siguiente resultado, vemos:



Aqui vemos un error AssertionError, falta un texto y muestra ''

Iteración 2: Claves con menos de 6 caracteres (Archivo: registro.cy.js).

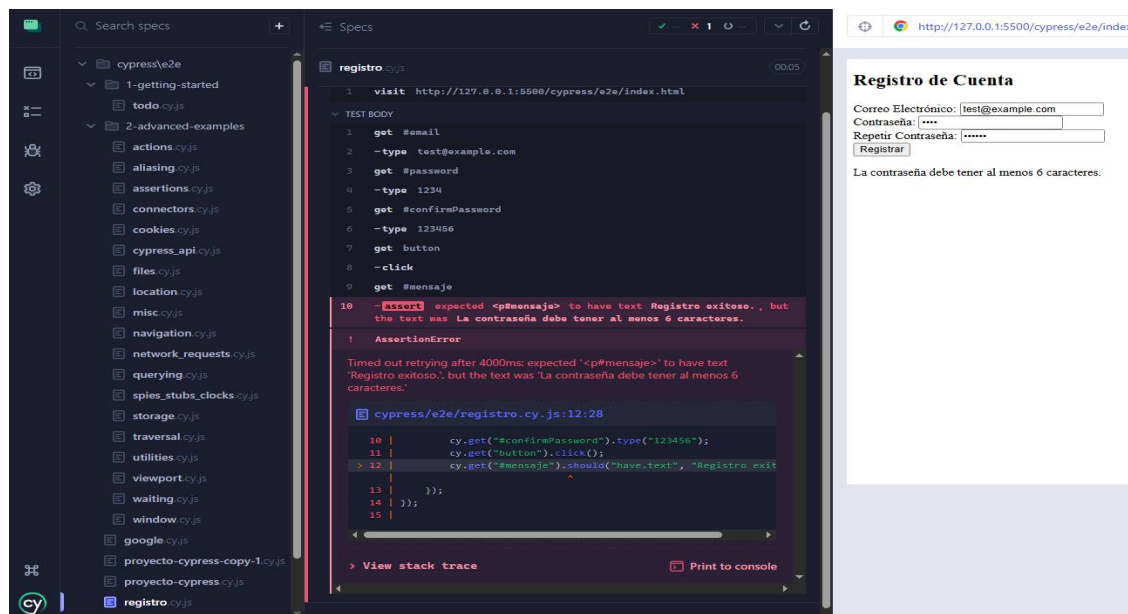
```
describe("Registro de usuario", () => {  
  beforeEach(() => {  
    // cy.visit("http://127.0.0.1:5500/index.html"); // Asegúrate de usar tu  
    ruta local  
    cy.visit("http://127.0.0.1:5500/cypress/e2e/index.html"); // Asegúrate de usar  
    tu ruta local  
  });  
  
  it("Debe registrar correctamente si todos los datos son correctos", () => {  
    cy.get("#email").type("test@example.com");  
    cy.get("#password").type("1234"); // menos de 6 caracteres
```

```

cy.get("#confirmPassword").type("123456");
cy.get("button").click();
cy.get("#mensaje").should("have.text", "Registro exitoso.");
});
});

```

al salvar se hace el test, resultando la siguiente pantalla:



El error muestra que la contraseña debe tener 6 caracteres.

Iteración 3: las contraseñas no coinciden (Archivo: registro.cy.js).

```

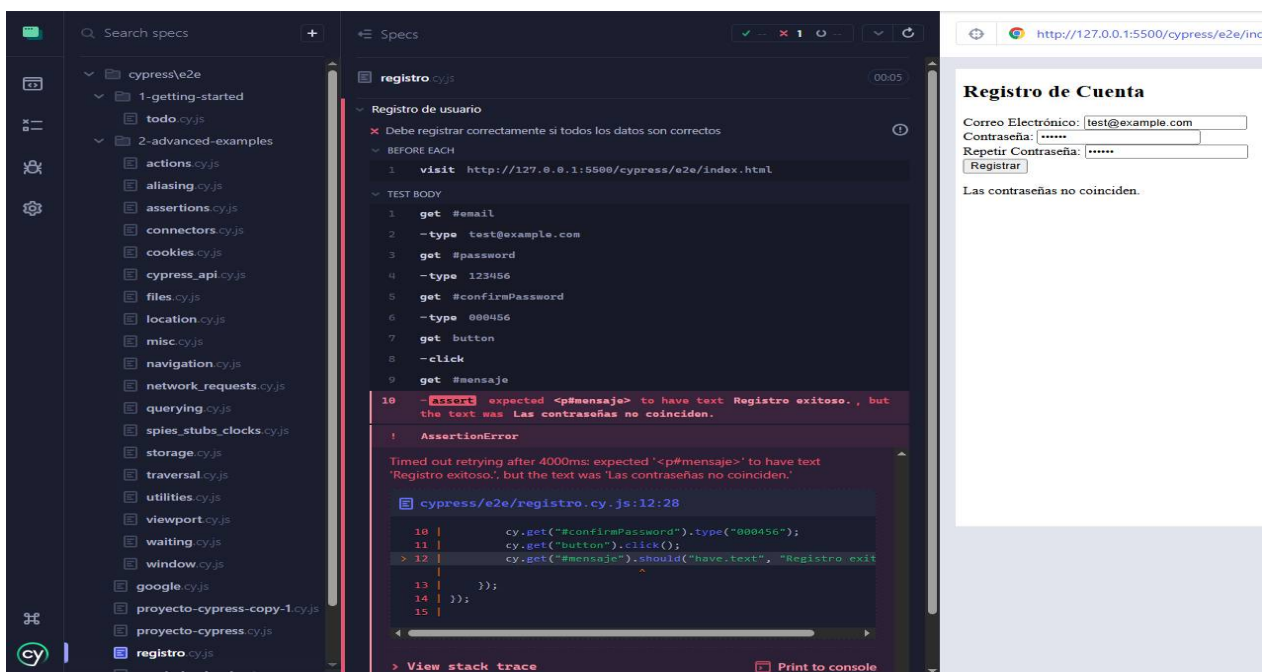
describe("Registro de usuario", () => {
  beforeEach(() => {
    // cy.visit("http://127.0.0.1:5500/index.html"); // Asegúrate de usar tu
    ruta local
    cy.visit("http://127.0.0.1:5500/cypress/e2e/index.html"); // Asegúrate de usar
    tu ruta local
  });

  it("Debe registrar correctamente si todos los datos son correctos", () => {
    cy.get("#email").type("test@example.com");
    cy.get("#password").type("123456");
    cy.get("#confirmPassword").type("000456");
    cy.get("button").click();
    cy.get("#mensaje").should("have.text", "Registro exitoso.");
  });
});

```



el resultado muestra error en las contraseñas que no son iguales.



Iteración 4: Campos sin errores (Archivo: registro.cy.js).

```
describe("Registro de usuario", () => {
```

```
  beforeEach(() => {
```

```
    // cy.visit("http://127.0.0.1:5500/index.html"); // Asegúrate de usar tu ruta local
```

```
    cy.visit("http://127.0.0.1:5500/cypress/e2e/index.html"); // Asegúrate de usar tu ruta local
```

```
  });
```

```
    it("Debe registrar correctamente si todos los datos son correctos", () => {
```

```
      cy.get("#email").type("test@example.com");
```

```
      cy.get("#password").type("123456");
```

```
      cy.get("#confirmPassword").type("123456");
```

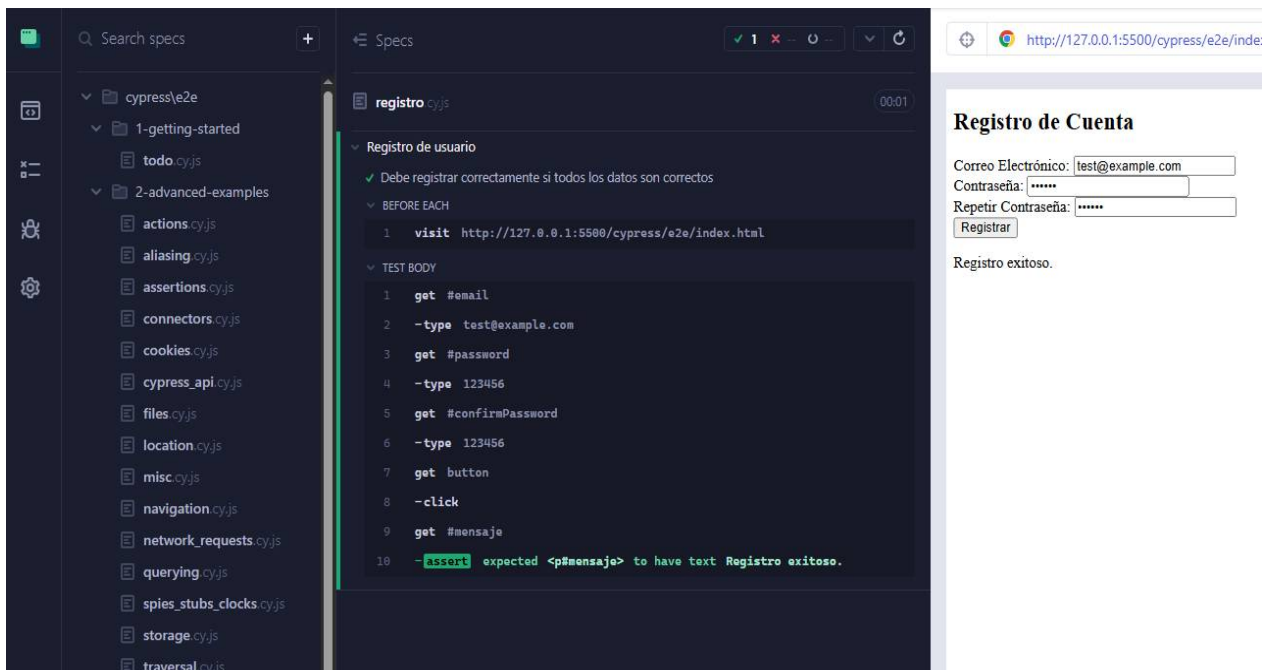
```
      cy.get("button").click();
```

```
      cy.get("#mensaje").should("have.text", "Registro exitoso.");
```

```
    });
```

```
  });
```

Finalmente corregimos el código para que no tenga ningún error y hacemos el testing.



Vemos que se generó todo lo esperado, y el código está cumpliendo todos los requerimientos y evaluaciones.

## CONCLUSIÓN

Cypress es una herramienta poderosa y moderna para realizar pruebas automatizadas en aplicaciones web. Su facilidad de configuración, integración con JavaScript y ejecución en tiempo real lo convierten en una excelente opción para desarrolladores y testers.

Técnicamente, las líneas de código y ejecución del test y el programa deben usar:

- Un método describe con la suite de test del archivo.
- Varios métodos test con cada test case que haya que comprobar.
- Formulario HTML.
- Bloques JavaScript sobre el DOM.
- Ejecución del Test en Cypress.

Desarrolla los pasos que habrá que seguir para definir todo el código de la manera que te planteamos en el PDF adjunto.

**Importante: debes adjuntar un PDF con la explicación y capturas de pantalla de cada paso del ejercicio**

Adjunta un archivo con tu respuesta.

# MISCELANEOS Y NOTAS ADICIONALES DE ESTUDIO

## JEST TEST

Para lograr aclarar algunos puntos de los temas a tratar se ha añadido este material adicional como REFERENCIA PERSONAL:

Breves notas y aclaratorias para estudio:  
comando Jest Test

Para usar jest test de forma manual donde podamos seleccionar los archivos podemos usar este comando en bash

```
npm test -- test002.test.js app.test002.js
```

el orden de colocación no afecta la ejecución.

Veamos algunos ejemplos:

Archivo: test003.app.js

```
// test003.app.js
function suma(a, b) {
  return a + b;
}

// Exportamos la función para poder usarla en el test
module.exports = suma;
```

Archivo: test003.test.js

```
// test003.test.js
const suma = require("./app"); // Importamos la función suma

test("Suma de 2 + 3 debe ser 5", () => {
  expect(suma(2, 3)).toBe(5);
});
```

Comando ejecución del test.

`npm test -- test003.app.js test003.test.js` // Nota: es indiferente la posición de los archivos  
resultado en terminal:

```
npm test -- test003.app.js test003.test.js
```

```
> test
> jest test003.app.js test003.test.js
```

```
PASS ./test003.test.js
  ✓ Suma de 2 + 3 debe ser 5 (5 ms)
```

```
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.611 s, estimated 1 s
Ran all test suites matching /test003.app.js|test003.test.js/i.
```

PC@abservo-desktop MINGW64 ~/Desktop/Cursos-online/FullStack/modulo-4/proyecto-testing

```
PC@abservo-desktop MINGW64 ~/Desktop/Cursos-online/FullStack/modulo-4/proyecto-testing
● $ npm test -- test003.app.js test003.test.js

> test
> jest test003.app.js test003.test.js

PASS ./test003.test.js
  ✓ Suma de 2 + 3 debe ser 5 (5 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.611 s, estimated 1 s
Ran all test suites matching /test003.app.js|test003.test.js/i.

PC@abservo-desktop MINGW64 ~/Desktop/Cursos-online/FullStack/modulo-4/proyecto-testing
○ $
```

ahora incluiremos un error en la función suma en el archivo test003.test.js

```
expect(suma(2, 2).toBe(5)); // aquí colocamos en esta línea el error.
```

Y ejecutamos de nuevo el comando

```
$ npm test -- test003.app.js test003.test.js
```

y obtenemos esto:

```
npm test -- test003.app.js test003.test.js
```

```
> test
> jest test003.app.js test003.test.js
```

```
FAIL ./test003.test.js
  ✕ Suma de 2 + 3 debe ser 5 (7 ms)
```

- Suma de 2 + 3 debe ser 5
- Suma de 2 + 3 debe ser 5

expect(received).toBe(expected) // Object.is equality

Expected: 5

Received: 4

```
3 |  
4 | test("Suma de 2 + 3 debe ser 5", () => {  
> 5 |   expect(suma(2, 2)).toBe(5);  
    |                   ^  
6 | });  
7 |
```

at Object.toBe (test003.test.js:5:22)

Test Suites: 1 failed, 1 total

Tests: 1 failed, 1 total

Snapshots: 0 total

Time: 0.702 s, estimated 1 s

Ran all test suites matching /test003.app.js|test003.test.js/i.

```
$ npm test -- test003.app.js test003.test.js  
  
> test  
> jest test003.app.js test003.test.js  
  
FAIL ./test003.test.js  
  ✕ Suma de 2 + 3 debe ser 5 (7 ms)  
  
    ● Suma de 2 + 3 debe ser 5  
  
    ● Suma de 2 + 3 debe ser 5  
  
      expect(received).toBe(expected) // Object.is equality  
  
      Expected: 5  
      Received: 4  
  
      3 |  
      4 | test("Suma de 2 + 3 debe ser 5", () => {  
    > 5 |   expect(suma(2, 2)).toBe(5);  
        |                   ^  
      6 | });  
      7 |  
  
      at Object.toBe (test003.test.js:5:22)  
  
Test Suites: 1 failed, 1 total  
Tests: 1 failed, 1 total  
Snapshots: 0 total  
Time: 0.702 s, estimated 1 s  
Ran all test suites matching /test003.app.js|test003.test.js/i.
```

Aquí vemos que no se recibe lo que se espera.

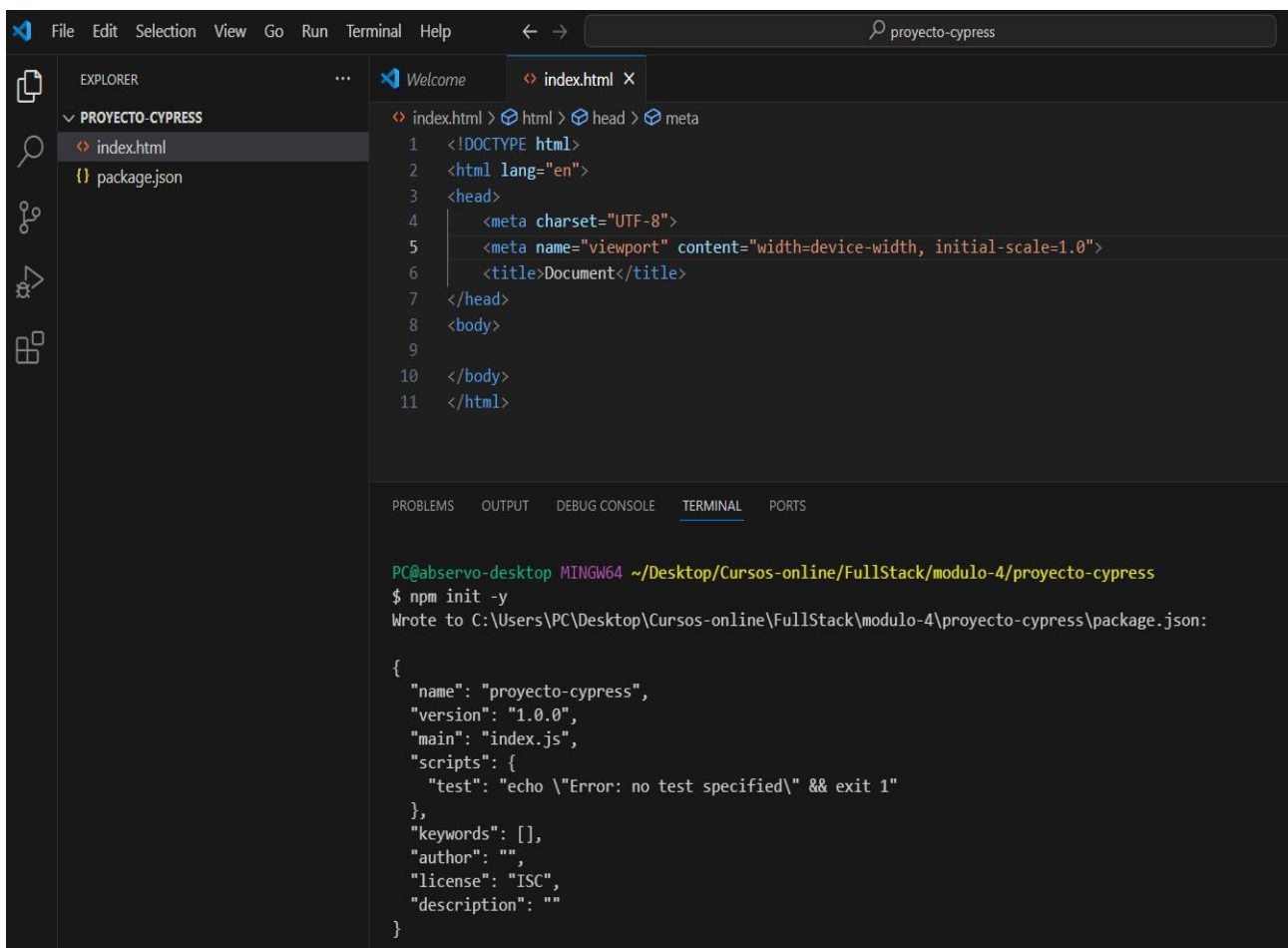
# CYPRESS TEST

Para lograr realizar la ejecución de esta herramienta, vamos a instalarla en primer lugar. Para ello crearemos una carpeta llamada proyecto-cypress con el explorador, entramos en visual code y abrimos con el menu File->Open folder... y obtenemos este resultado después de ejecutar el comando para inicializar el npm:

en el bash copiamos

```
npm init -y
```

esto creará el archivo package.json por defecto, para evitar que nos pida datos.



```
File Edit Selection View Go Run Terminal Help
projecto-cypress

EXPLORER
PROYECTO-CYPRESS
  index.html
  package.json

index.html > html > head > meta
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9
10 </body>
11 </html>

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PC@abserveo-desktop MINGW64 ~/Desktop/Cursos-online/FullStack/modulo-4/proyecto-cypress
$ npm init -y
Wrote to C:\Users\PC\Desktop\Cursos-online\FullStack\modulo-4\proyecto-cypress\package.json:

{
  "name": "proyecto-cypress",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

A continuación tecleamos el comando en el terminal bash:

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

con esto instalamos cypress

Vamos a utilizar la documentación oficial de cypress y agregaremos el script en el archivo

package.json que creamos:

Sitio oficial cypress <https://www.cypress.io/>

y documentación de instalación:

<https://docs.cypress.io/app/get-started/install-cypress>

<https://docs.cypress.io/app/get-started/open-the-app>

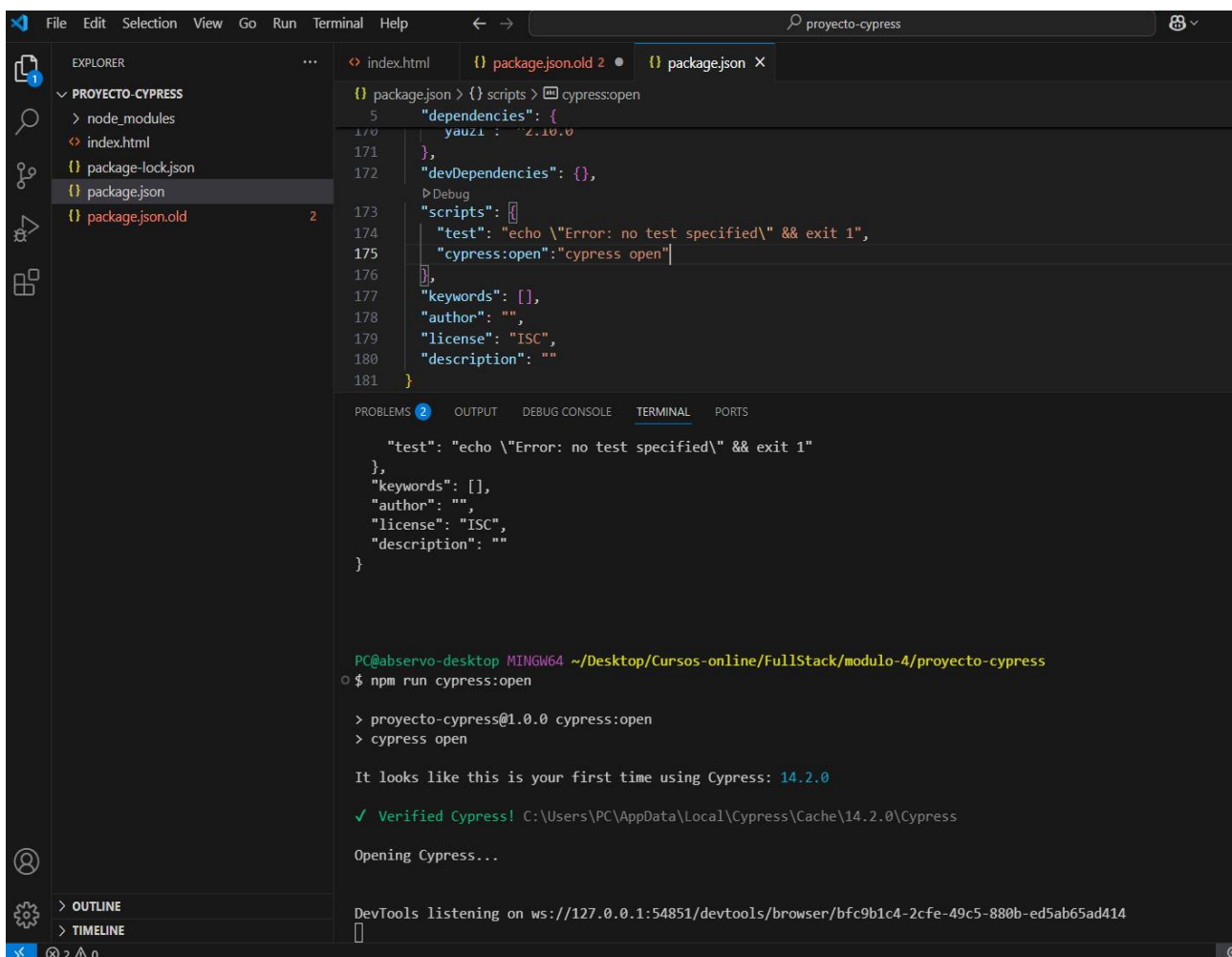
y agregaremos en el archivo package.json

```
"cypress:open": "cypress open"
```

en el script:

```
"scripts": {  
  "test": "echo \"Error: no test specified\" && exit 1",  
  "cypress:open": "cypress open"  
},
```

veamos:



The screenshot shows a Visual Studio Code editor with a project named 'proyecto-cypress'. The Explorer sidebar on the left shows the file structure with 'package.json' selected. The main editor area displays the content of 'package.json', which includes the 'cypress:open' script. The Terminal panel at the bottom shows the command 'npm run cypress:open' being executed, resulting in Cypress 14.2.0 being installed and opened. The terminal output includes the message 'Verified Cypress!' and 'Opening Cypress...'. The status bar at the bottom indicates 2 errors and 0 warnings.

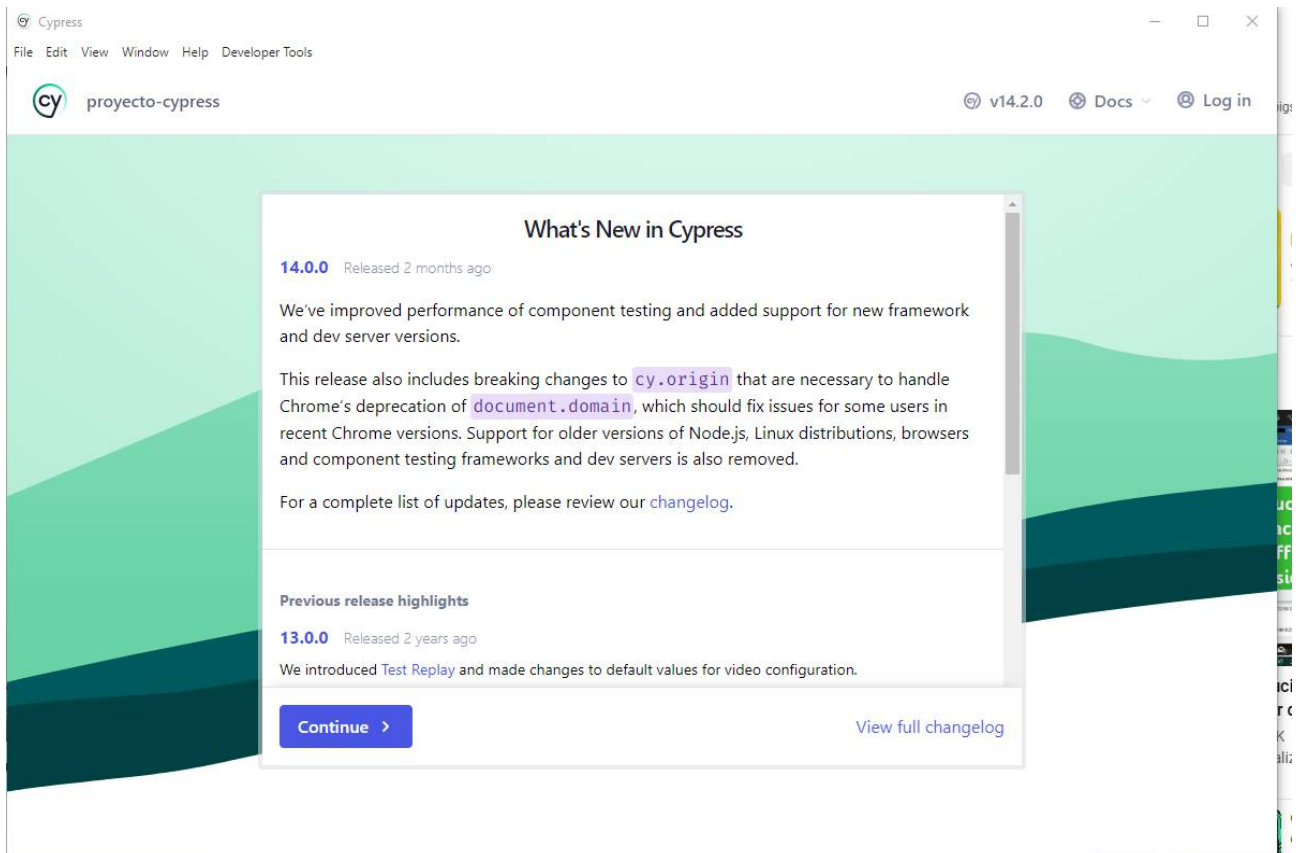
```
File Edit Selection View Go Run Terminal Help  
proyecto-cypress  
EXPLORER  
PROJECTO-CYPRESS  
  node_modules  
  index.html  
  package-lock.json  
  package.json  
  package.json.old 2  
package.json  
  {} package.json > {} scripts > cypress:open  
  5    "dependencies": {  
  170      "yauzi": "2.10.0"  
  171    },  
  172    "devDependencies": {},  
  173    "scripts": {  
  174      "test": "echo \"Error: no test specified\" && exit 1",  
  175      "cypress:open": "cypress open"  
  176    },  
  177    "keywords": [],  
  178    "author": "",  
  179    "license": "ISC",  
  180    "description": ""  
  181  }  
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS  
  "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC",  
  "description": ""  
}  
PC@abserveo-desktop MINGW64 ~/Desktop/Cursos-online/FullStack/modulo-4/proyecto-cypress  
$ npm run cypress:open  
> proyecto-cypress@1.0.0 cypress:open  
> cypress open  
It looks like this is your first time using Cypress: 14.2.0  
✓ Verified Cypress! C:\Users\PC\AppData\Local\Cypress\Cache\14.2.0\Cypress  
Opening Cypress...  
DevTools listening on ws://127.0.0.1:54851/devtools/browser/bfc9b1c4-2cfe-49c5-880b-ed5ab65ad414  
2 0
```



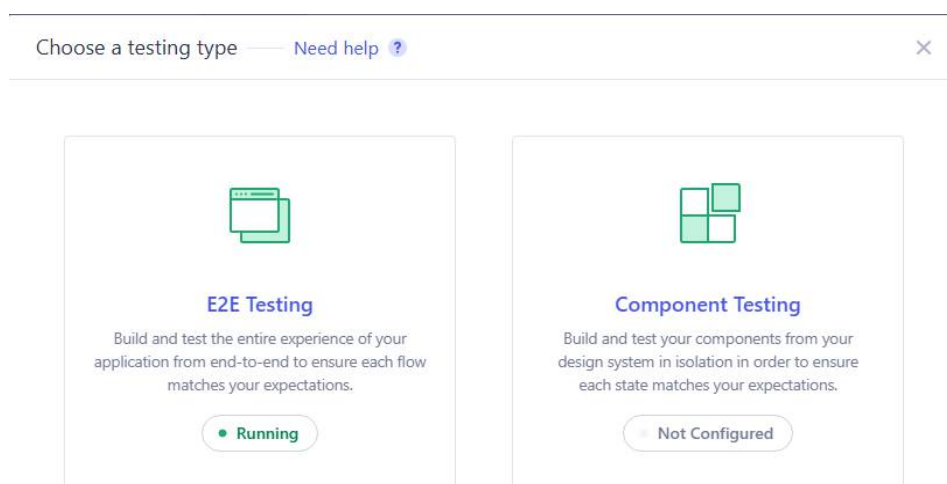
y corremos el comando que abrirá el programa cypress y su ventana en el bash

```
npm run cypress:open
```

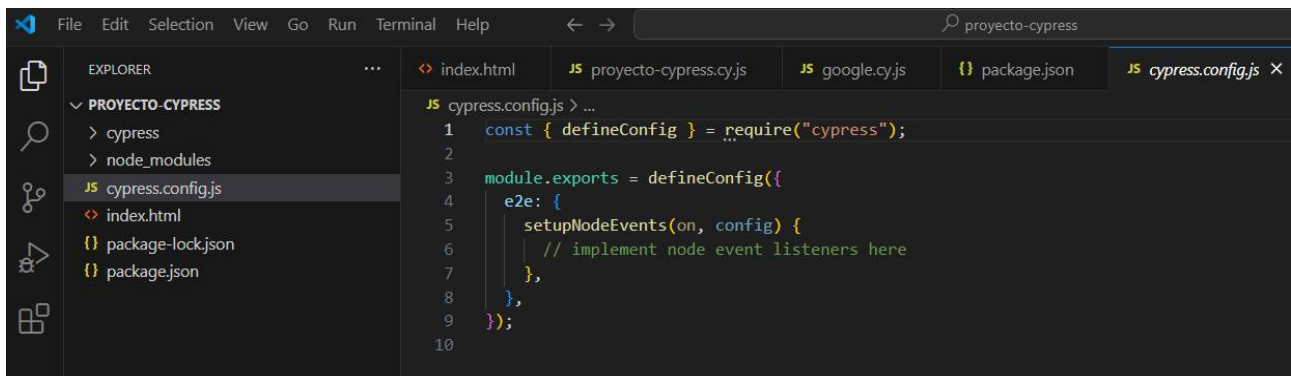
esto nos mostrara la primera ejecución tal como se ve a continuación:



luego al terminar tenemos la siguiente ventana:



luego podemos configurar el cypress en este archivo cypress.config.js de configuración :

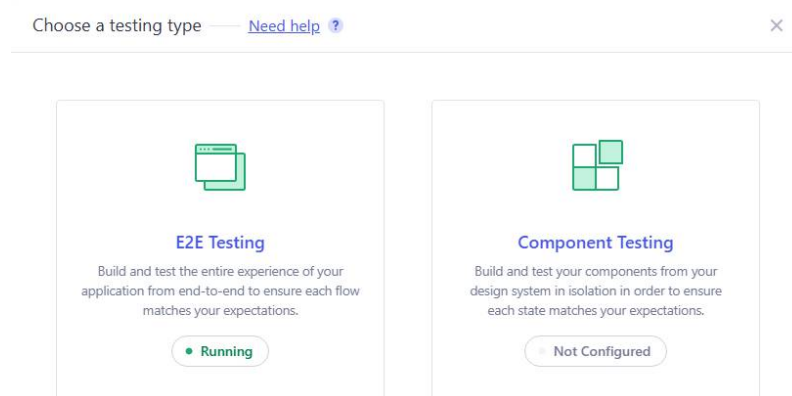


Ejemplo 01 - Prueba apertura del navegador.

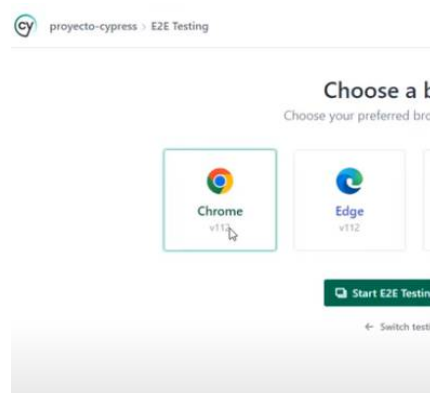
Abrimos cypress con el siguiente comando en el bash de visual code

```
npm run cypress:open
```

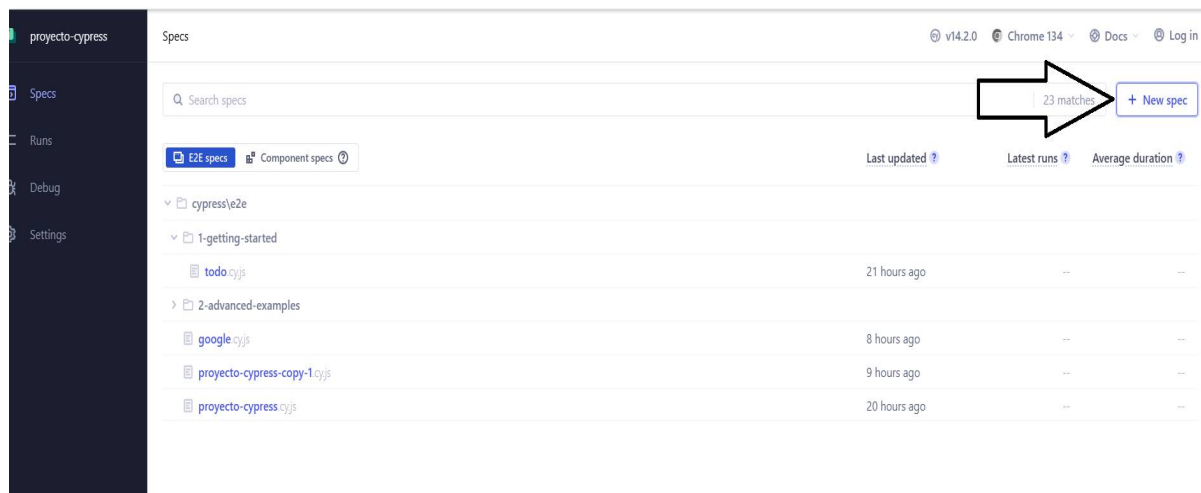
vemos la siguiente pantalla :



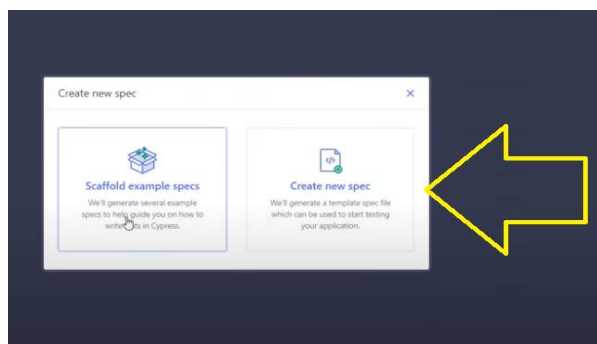
seleccionamos E2E y el navegador por ejemplo google



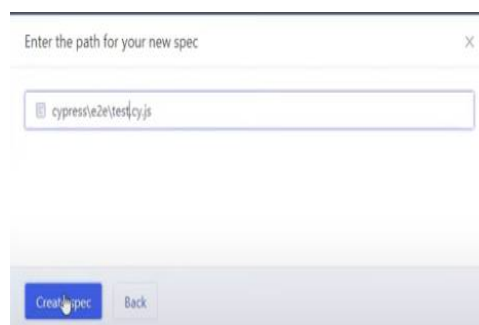
en la ventana del navegador que acaba de aparecer colocamos en nombre del archivo de test.



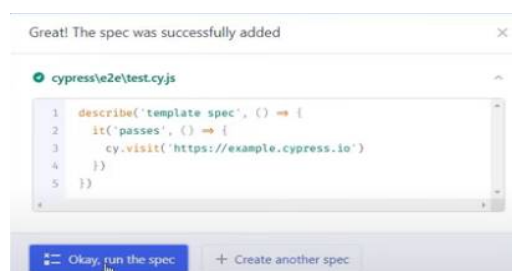
Luego seleccionamos Create new spec



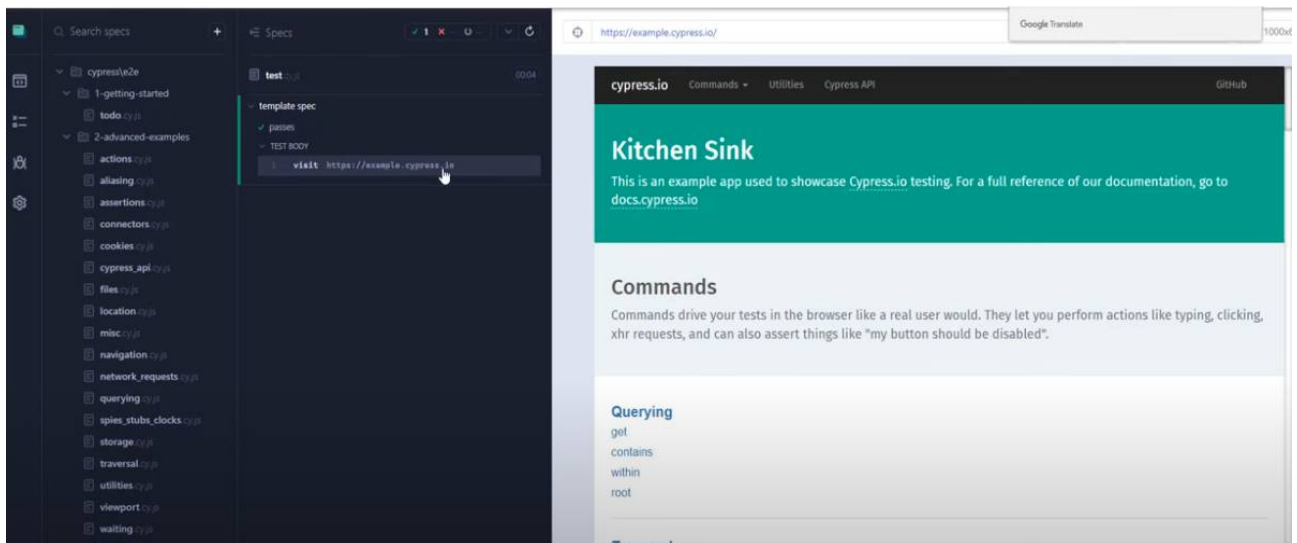
y colocamos en la ruta test.cy.js



luego aparecerá una pequeña plantilla de como se verá el test

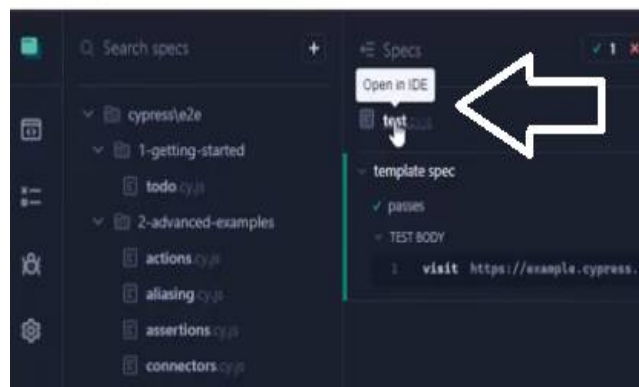


el resultado es la ejecución del `example.cypress.io`

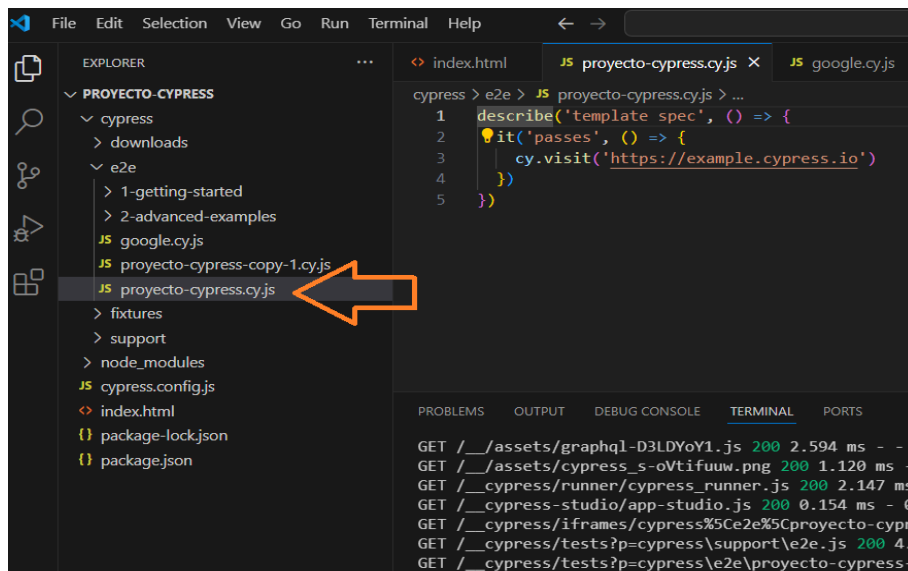


aquí ya estamos listo para ejecutar la visualización de google.

Podemos abrir el archivo que se está ejecutando el test en visual code en la siguiente opción:

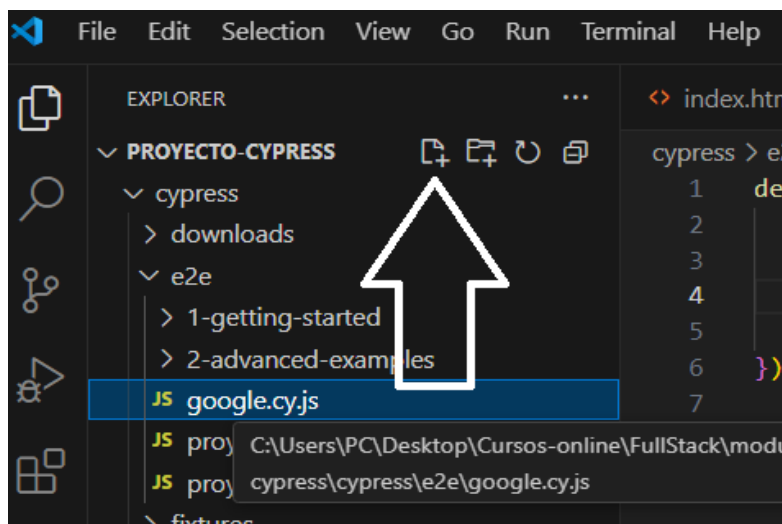


resultado el ide de visual code

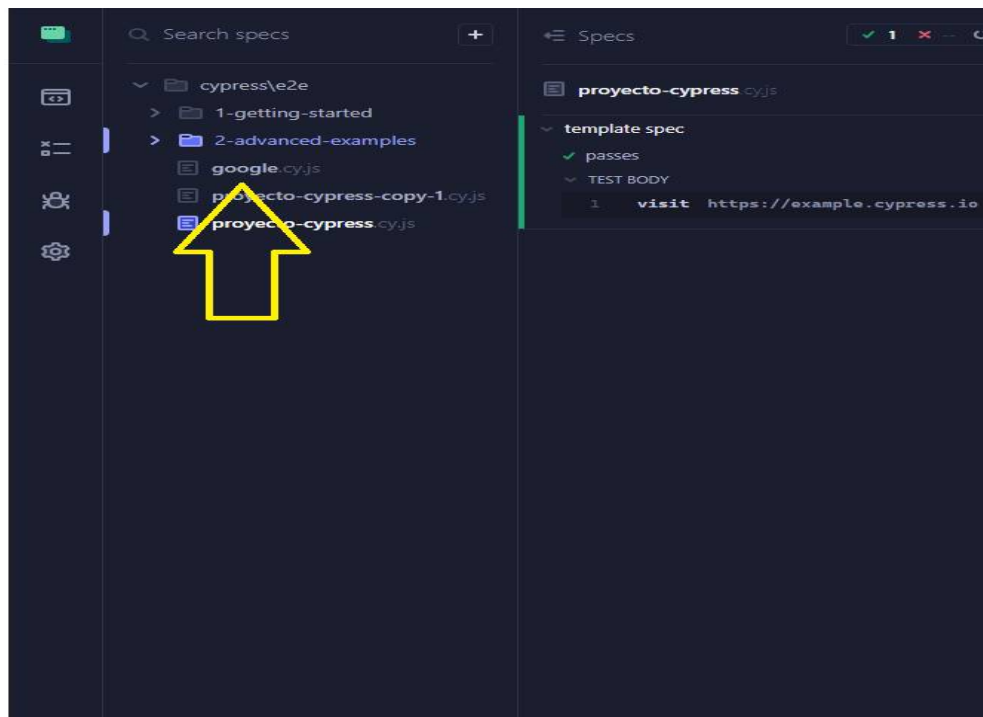


También directamente en visual code en la carpeta E2E, creamos un archivo \*.cy.js en este caso haremos el ejemplo de google.

google.cy.js



y refrescamos el navegador, allí aparecerá el archivo google.cy.js



Ya estamos listo para realizar el ejemplo. Apertura de google.

Tipeamos el siguiente código en el archivo google.cy.js

```
describe('Prueba de apertura de google',()=>{  
  it('Abre la pagina de google correctamente',()=>{  
    cy.visit('https://www.google.com');  
    cy.title().should('eq','Google')  
  })  
});
```

al salvar se ejecuta automaticamente la prueba resultando en lo siguiente:





ver tambien video tutorial en youtube:

prueba con google

<https://www.youtube.com/watch?v=ysaCj6KutDc>

<https://www.youtube.com/watch?v=u8vMu7viCm8&t=244s>

documentación consultada cypress.

<https://docs.cypress.io/app/get-started/why-cypress>

<https://www.cypress.io/>