Documento de Pruebas

TC2007B: Integración de seguridad informática en redes y sistemas de software

Grupo 402

Andrea Alexandra Barrón Córdova A01783126

Mario Ignacio Frías Pina A01782559

Alan Anthony Hernández Pérez A01783347

Oswaldo Ilhuicatzi Mendizábal A01781988

Bajo la instrucción de

Lizbeth Peralta Malváez

20 de octubre de 2023

Índice

Introducción

En este módulo se desea realizar un sistema de tickets que permita levantar reportes sobre los distintos incidentes que ocurran en las aulas de la fundación *Por México*. Sin embargo, a lo largo de cada proyecto de desarrollo se cometen errores. En este ámbito, es necesario verificar que los distintos códigos no tengan anomalías que afecten su eficiencia, por lo que, en este objetivo, se busca realizar pruebas que ayuden a verificar el correcto funcionamiento de las diferentes funciones del código y que cumplan con sus tareas especificadas. Es por esto que se emplearán dos herramientas en específico para esto: *Jest* y *React Testing Library.*

Herramientas utilizadas

Jest

Jest es un marco de pruebas para la corrección de cualquier código JavaScript. Por otro lado, permite escribirlas a través de un (API) accesible. Así, al instalar Jest, se configura una serie de ensayos para comprobar si las diferentes partes del código funcionan de la manera esperada.

Debido a que Jest trabaja con pruebas sobre el marco del mismo código, esto hace que no sea de gran apoyo para temas de accesibilidad. Sin embargo, es útil para realizar ensayos aún más amplios.

Por ejemplo, Jest no permite acceder a las pruebas desde un punto de vista de usuario, limitando así la accesibilidad al criterio del desarrollador, donde las únicas pruebas posibles son que otros usuarios dejen comentarios sobre las dificultades que encontraron al usarlo. Asimismo, la velocidad es uno de los fuertes de Jest: es capaz de operar con pruebas en paralelo para mejorar el tiempo de ejecución.

React Testing Library

React Testing Library es una librería que cuenta con funciones basadas en lo que es DOM(*Document Object Model*) Testing Library añadiendo APIs para trabajar con componentes React utilizando eventos DOM. Esto permite hacer pruebas de accesibilidad que complementan a las hechas con Jest, como por ejemplo, a través de lectores de pantalla, lo que permite entender cómo es que los usuarios podrían interactuar con la aplicación, así como medir el índice de accesibilidad de las diversas funcionalidades de esta. Así, se puede obtener una corrección comprobada desde el punto de vista del usuario.

Pruebas realizadas

Las pruebas se realizaron sobre la comunicación con el backend, como por ejemplo, que tuviera funcionalidades correctas a la hora de presionar ciertos botones del frontend . Por otro lado, se comprobó que se puede comunicar con la base de datos de MongoDB, que no te permite recibir información si no tienes autorización, que te permite hacer login con credenciales válidas y que te impide si son inválidas, y finalmente que puedes recibir la información si tienes un ticket válido.

El primer conjunto de pruebas trata sobre una de las funcionalidades del frontend donde existe un botón que cambia el tema a claro o oscuro dependiendo del tema actual, y para verificar el comportamiento de este componente realizamos un evento de click que cambia el texto dentro del botón y se comprueba que el botón cambió su estado. Esta misma prueba se realiza 2 veces, tanto con la librería de *Jest* como con la librería de *React Testing Library*. A continuación se presenta el código implementado y los resultados de las pruebas mencionadas:

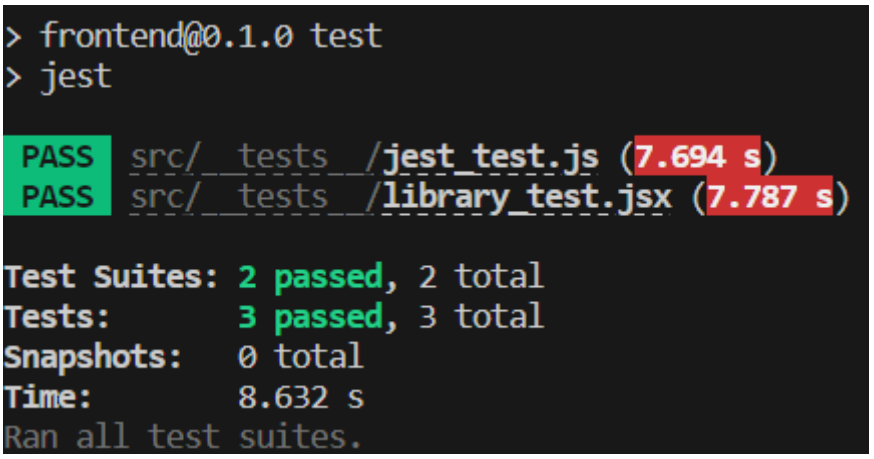

Figura 1. Imagen de las pruebas del frontend.

```
/**
 * @jest-environment jsdom
 */

import {render, fireEvent, screen} from '@testing-library/react'
import '@testing-library/jest-dom'
import { ThemeToggler } from "../CustomLayout";


test('loads and displays theme', async () => {
  render(<ThemeToggler />)

  fireEvent.click(screen.getByText('Cambiar a tema oscuro'))

  await screen.findByRole('button')

  expect(screen.getByRole('button')).toHaveTextContent('Cambiar a tema claro')
})
```

Figura 2. Imagen de la implementación de la prueba del frontend con react testing library

```
/**
 * @jest-environment jsdom
 */

import {cleanup, fireEvent, render, screen} from '@testing-library/react';
import { ThemeToggler } from "../CustomLayout";
import { AdminContext } from 'react-admin';

afterEach(cleanup);

describe('Pruebas de tema', () =>{
  it('Tema es oscuro', async() => {
    render(
      <AdminContext>
        <ThemeToggler />,
      </AdminContext>
    );
    const items = await screen.getByText('Cambiar a tema oscuro', { selector: 'button' })
    expect(items).toBeTruthy();
  });
  it('Tema es claro', async() => {
    render(
      <AdminContext>
        <ThemeToggler />,
      </AdminContext>
    );
    fireEvent.click(screen.getByText('Cambiar a tema oscuro'))
    const items = await screen.getByText('Cambiar a tema claro', { selector: 'button' })
    expect(items).toBeTruthy();
  });
});
```

Figura 3. Imagen de la implementación de la prueba del frontend con Jest

El segundo conjunto de pruebas trata sobre la funcionalidad del backend donde se requiere comprobar que esta parte se comunique con la base de datos. Para la primera prueba se inicia la conexión con la base de datos y se espera que no ocurra ningún error al momento de conectarse, logrando así que la prueba se complete de manera exitosa. A continuación se presenta el código implementado y los resultados de las pruebas mencionadas:

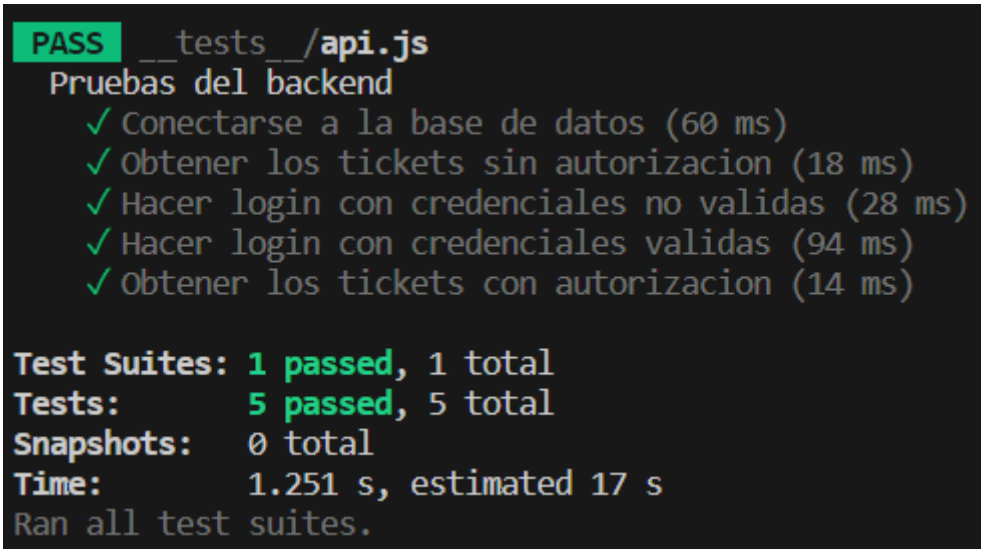

Figura 4. Imagen de las pruebas del backend.

```
describe("Pruebas del backend", () => {
    let token;
    test("Conectarse a la base de datos", () => {
        return expect(connectDB()).resolves.toBe(); //llama la conexion del server.js y verifica que se conecte
    });
```

Figura 5. Imagen de la prueba de comunicación con la base de datos

Para el último conjunto de pruebas se crean solicitudes para la base de datos donde se esperan diferentes respuestas dependiendo de la información que le es proporcionada a través de cada solicitud. Inicialmente se comunicó con la base de datos sin ninguna herramienta de autenticación, donde se espera que la respuesta sea un estado de *error 401: no autorizado.*

```
test("Obtener los tickets sin autorizacion", () => {
  return request(app)
    .get("/tickets") //hace un llamada al app al endpoint de tickets
    .then(response => {
      expect(response.statusCode).toBe(401); //espera que el estado de un error con el codigo 401 == no autorizacion
    });
});
```

Figura 6. Imagen de la prueba de obtener información sin autorización

Durante las siguientes pruebas se proporciona información de inicio de sesión a la solicitud correspondiente. Primero se intenta iniciar sesión con información no válida, a lo cual se responderá con un estado de *no autorizado*. Después se inicia sesión con información válida que responde con un vale de autorización que puede ser utilizado posteriormente para obtener información de la base de datos.

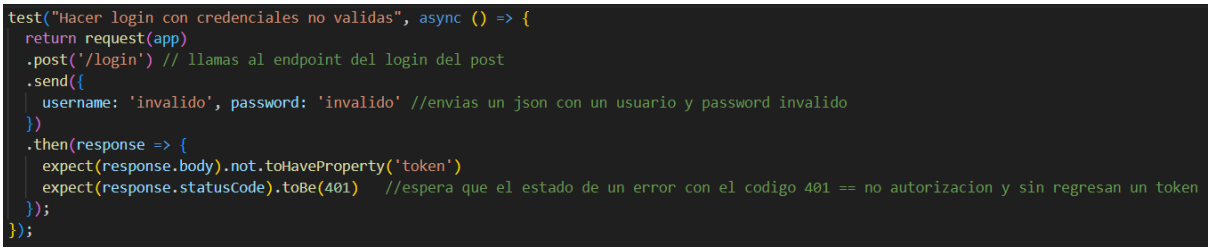

Figura 7. Imagen de la prueba de hacer login sin credenciales válidas.

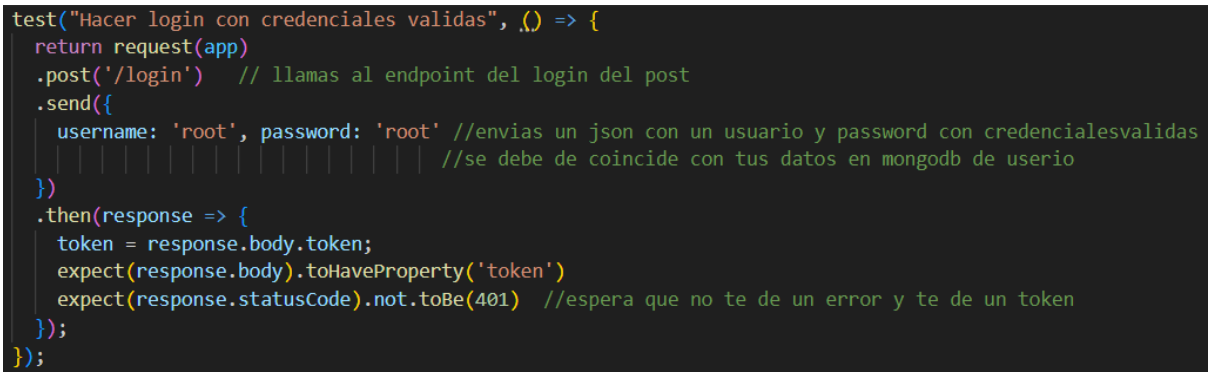

Figura 8. Imagen de la prueba de hacer login con credenciales válidas.

Finalmente, se vuelve a intentar obtener información de la base de datos proporcionando a la solicitud las credenciales contenidas dentro del vale obtenido anteriormente, y se espera que la base de datos nos proporcione la información pedida y un estado *válido*.

Figura 9. Imagen de la prueba de obtener información con autorización.

Conclusión

Al final de este recorrido, se puede constatar que cada una de las pruebas se logró realizar de manera exitosa. Así, se concretó que cada una de las funcionalidades implementadas trabajaban de la manera esperada y que cumplían incluso con sus restricciones establecidas. Asimismo, se logró complementar cada una de las pruebas con los distintos enfoques, *Jest* por un lado con su eficiencia para hacer pruebas mucho más amplias, y *React Testing Library* al dar a entender cómo es que los usuarios interactúen con la aplicación.

Referencias

Jest (2023) *Jest* Jest.
https://jestjs.io/

Testing library (n.d.) *React testing library - Api accessibility.* Testing library
https://testing-library.com/docs/dom-testing-library/api-accessibility/

Testing library (n.d.) *React testing library - Introduction.* Testing library
https://testing-library.com/docs/react-testing-library/intro/

Hernández A., Frias M., Barrón A., Ilhuicatzi O., (2023) *Repositorio del equipo.* Github
https://github.com/Alank99/Proyecto-fundacion-por-mexico-por-arbolitos