

Taller testing

Presentado por:

Cristian Medina - crmedinab@unal.edu.co

Cristian Montañez - cmontanez@unal.edu.co

Justin Rodriguez - jusrodriguez@unal.edu.co

Sergio Ruiz – seruizh@unal.edu.co

Profesor:

Oscar Eduardo Alvarez Rodriguez

27/02/2025



Universidad Nacional de Colombia

Facultad de Ingeniería

Ingeniería de Software

Departamento de Ingeniería de Sistemas

Descripción general de la aplicación.

La aplicación es una plataforma centralizada para gestionar objetos perdidos y encontrados en la Universidad Nacional de Colombia. Los usuarios se registran y pueden reportar, buscar y gestionar objetos mediante filtros, facilitando la comunicación entre quienes encuentran y pierden objetos. En esencia, la "Plataforma de Objetos Perdidos y Encontrados" busca transformar la experiencia de la comunidad universitaria al proporcionar una herramienta eficiente y accesible para la gestión de objetos extraviados. Al centralizar el proceso contribuye a reducir la pérdida de tiempo y la frustración asociadas a la búsqueda de objetos perdidos, al mismo tiempo que fomenta una cultura colaborativa dentro de la universidad.

Tipo de prueba realizada y Herramienta

Se implementaron pruebas unitarias enfocadas en el frontend utilizando Jest, el framework de testing de JavaScript desarrollado por Meta. Esta elección se basó en su compatibilidad nativa con ecosistemas frontend como React. Estas pruebas se diseñaron para validar la funcionalidad de cada componente de la interfaz, asegurando que la renderización, el manejo de eventos y la interacción del usuario se ejecuten de manera correcta. Este enfoque no solo incrementó la confiabilidad y robustez del código, sino que también facilitó su mantenibilidad y escalabilidad a lo largo del proyecto.

Home.test.jsx - Sergio Alejandro Ruiz Hurtado

Descripción del componente:

El componente Home es la página principal de la aplicación y se encarga de integrar componentes clave como Header, LostObjectsSection y Footer.

Pruebas realizadas:

- Verificación del renderizado correcto de todas las secciones principales (header, sección de objetos perdidos y footer).
- Uso de mocks para simular componentes hijos y confirmar su presencia en el documento.

Código del test

```
1 import React from 'react';
2 import { render, screen } from '@testing-library/react';
3 import { BrowserRouter } from 'react-router-dom';
4 import Home from '../templates/home/Home';
5 import Header from '../organisms/header/Header';
6
7 // Mock de los componentes hijos
8 jest.mock('../organisms/header/Header', () => {
9   const Header = () => <div data-testid='mock-header'>Header</div>;
10   Header.displayName = 'Header';
11   return Header;
12 });
13
14 jest.mock('../organisms/lostObjectsSection/LostObjectsSection', () => {
15   const LostObjectsSection = () => (
16     <div data-testid='mock-lost-objects'>Lost Objects Section</div>
17   );
18   LostObjectsSection.displayName = 'LostObjectsSection';
19   return LostObjectsSection;
20 });
21
22 jest.mock('../organisms/footer/Footer', () => {
23   const Footer = () => <div data-testid='mock-footer'>Footer</div>;
24   Footer.displayName = 'Footer';
25   return { Footer };
26 });
27
28 describe('Home Component', () => {
29   test('renders all main sections', () => {
30     render(
31       <BrowserRouter>
32         <Home />
33       </BrowserRouter>
34     );
35
36     expect(screen.getByTestId('mock-header')).toBeInTheDocument();
37     expect(screen.getByTestId('mock-lost-objects')).toBeInTheDocument();
38     expect(screen.getByTestId('mock-footer')).toBeInTheDocument();
39   });
40 });
```

Resultado de la ejecución

```
PS C:\Users\Usuario\Desktop\ADESK\Software_Engineering\Proyecto\frontend> npm test Home.test.jsx
> frontend@0.0.0 test
> jest Home.test.jsx

jest-haste-map: duplicate manual mock found: fileMock
  The following files share their name; please delete one of them:
    * <rootDir>\__mocks__\fileMock.cjs
    * <rootDir>\__mocks__\fileMock.js

(node:20488) [DEP0040] DeprecationWarning: The 'punycode' module is deprecated. Please use a userland alternative instead.
(Use 'node --trace-deprecation ...' to show where the warning was created)
PASS src/__test__/Home.test.jsx
  Home Component
    ✓ renders all main sections (16 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.903 s, estimated 1 s
Ran all test suites matching /Home.test.jsx/i.
```

| Login.test.jsx - Justin Brad Rodriguez Sanchez | |
|--|--|
| Descripción del componente: | El componente Login gestiona la autenticación de usuarios mediante un formulario de inicio de sesión. |
| Pruebas realizadas: | <ul style="list-style-type: none">• Renderizado correcto del formulario de login.• Comprobación de la presencia de los campos de email y contraseña.• Verificación de los botones "Iniciar Sesión" y "Registrate".• Simulación del formulario mediante mocks para aislamiento de pruebas. |
| Código del test | |
| <pre>1 import React from 'react'; 2 import { render, screen, fireEvent } from '@testing-library/react'; 3 import { BrowserRouter } from 'react-router-dom'; 4 import Login from '../templates/login/Login'; 5 6 // Mock de los componentes hijos 7 jest.mock(8 '../organisms/loginRegistrationForget/LoginRegistrationForget', 9 () => { 10 const LoginRegistrationForget = () => (11 <div data-testid='mock-login-form'> 12 <input data-testid='email-input' type='email' /> 13 <input data-testid='password-input' type='password' /> 14 <button data-testid='login-button'>Iniciar Sesión</button> 15 <button data-testid='register-button'>Registrate</button> 16 </div> 17); 18 LoginRegistrationForget.displayName = 'LoginRegistrationForget'; 19 return LoginRegistrationForget; 20 }, 21); 22 23 describe('Login Component', () => { 24 const renderLogin = () => { 25 render(26 <BrowserRouter> 27 <Login /> 28 </BrowserRouter>, 29); 30 }; 31 32 test('renders login form', () => { 33 renderLogin(); 34 expect(screen.getByTestId('mock-login-form')).toBeInTheDocument(); 35 }); 36 }</pre> | |

```

37     test('renders email and password inputs', () => {
38         renderLogin();
39         expect(screen.getByTestId('email-input')).toBeInTheDocument();
40         expect(screen.getByTestId('password-input')).toBeInTheDocument();
41     });
42
43     test('renders login and register buttons', () => {
44         renderLogin();
45         expect(screen.getByTestId('login-button')).toBeInTheDocument();
46         expect(screen.getByTestId('register-button')).toBeInTheDocument();
47     });
48 });
49

```

Resultado de la ejecución

```

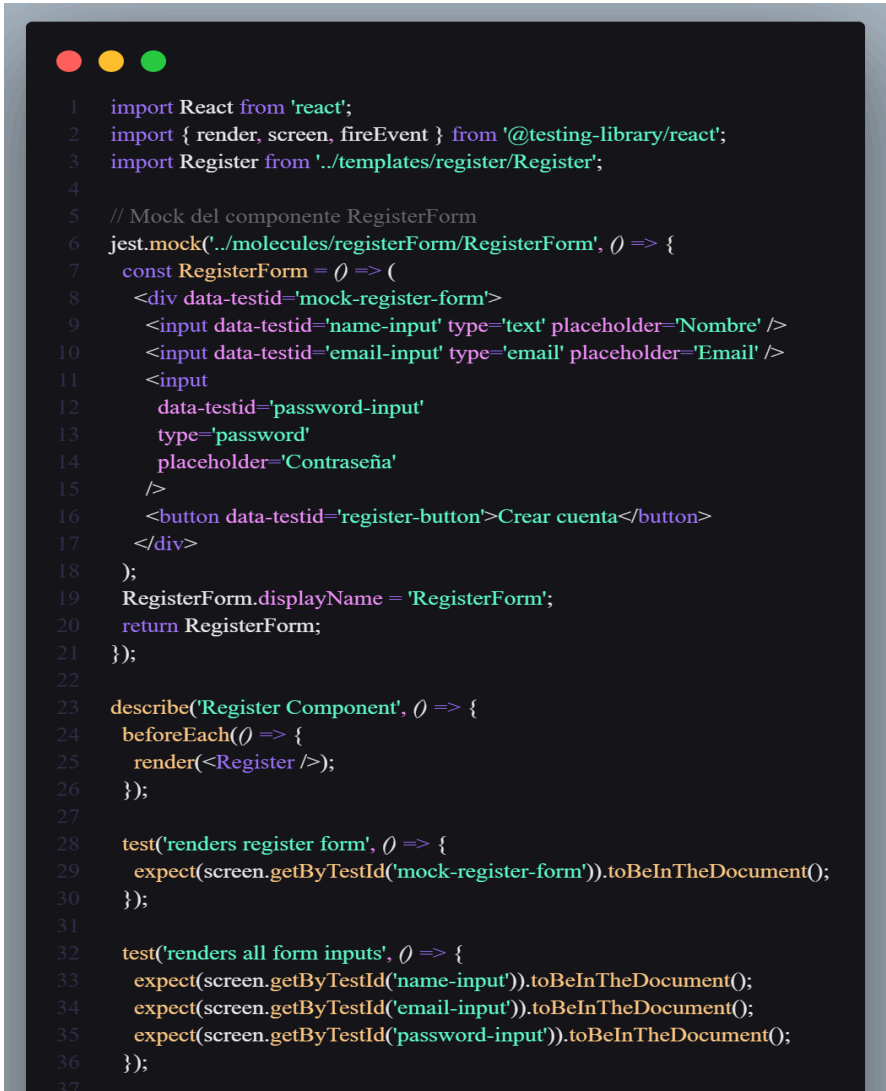
PS C:\Users\Usuario\Desktop\ADESR\Software_Engineering\Proyecto\Frontend> npm test Login.test.jsx
> frontend@0.0.0 test
> jest Login.test.jsx

jest-haste-map: duplicate manual mock found: fileMock
  The following files share their name; please delete one of them:
  * <rootDir>\_mocks_\fileMock.cjs
  * <rootDir>\_mocks_\fileMock.js

(node:9516) [DEP0040] DeprecationWarning: The 'punycode' module is deprecated. Please use a userland alternative instead.
(Use 'node --trace-deprecation ...' to show where the warning was created)
PASS src/__test__/Login.test.jsx
  Login Component
    ✓ renders login form (18 ms)
    ✓ renders email and password inputs (3 ms)
    ✓ renders login and register buttons (2 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        0.91 s, estimated 1 s
Ran all test suites matching /Login.test.jsx/i.

```

| Register.test.jsx - Cristian Daniel Montanez Pineda | |
|--|--|
| Descripción del componente: | El componente Register permite la creación de nuevas cuentas de usuario a través de un formulario de registro. |
| Pruebas realizadas: | <ul style="list-style-type: none"> • Renderizado adecuado del formulario de registro. • Confirmación de los campos obligatorios (nombre, email, contraseña). • Verificación del botón "Crear cuenta" y su texto correcto. • Uso de mocks para simular el formulario y garantizar pruebas independientes. |
| Código del test | |
|  <pre> 1 import React from 'react'; 2 import { render, screen, fireEvent } from '@testing-library/react'; 3 import Register from '../templates/register/Register'; 4 5 // Mock del componente RegisterForm 6 jest.mock('../molecules/registerForm/RegisterForm', () => { 7 const RegisterForm = () => (8 <div data-testid='mock-register-form'> 9 <input data-testid='name-input' type='text' placeholder='Nombre' /> 10 <input data-testid='email-input' type='email' placeholder='Email' /> 11 <input 12 data-testid='password-input' 13 type='password' 14 placeholder='Contraseña' 15 /> 16 <button data-testid='register-button'>Crear cuenta</button> 17 </div> 18); 19 RegisterForm.displayName = 'RegisterForm'; 20 return RegisterForm; 21 }); 22 23 describe('Register Component', () => { 24 beforeEach(() => { 25 render(<Register />); 26 }); 27 28 test('renders register form', () => { 29 expect(screen.getByTestId('mock-register-form')).toBeInTheDocument(); 30 }); 31 32 test('renders all form inputs', () => { 33 expect(screen.getByTestId('name-input')).toBeInTheDocument(); 34 expect(screen.getByTestId('email-input')).toBeInTheDocument(); 35 expect(screen.getByTestId('password-input')).toBeInTheDocument(); 36 }); 37 </pre> | |

```

38   test('renders register button', () => {
39     expect(screen.getByTestId('register-button')).toBeInTheDocument();
40     expect(screen.getByTestId('register-button')).toHaveTextContent(
41       'Crear cuenta',
42     );
43   });
44 });
45

```

Resultado de la ejecución

```

PS C:\Users\Usuario\Desktop\ADESK\Software_Engineering\Proyecto\frontend> npm test Register.test.jsx

> frontend@0.0.0 test
> jest Register.test.jsx

jest-haste-map: duplicate manual mock found: fileMock
  The following files share their name; please delete one of them:
    * <rootDir>\__mocks__\fileMock.cjs
    * <rootDir>\__mocks__\fileMock.js

(node:17816) [DEP0040] DeprecationWarning: The 'punycode' module is deprecated. Please use a userland alternative instead.
(Use 'node --trace-deprecation ...' to show where the warning was created)
PASS src/_test_/Register.test.jsx
  Register Component
    ✓ renders register form (17 ms)
    ✓ renders all form inputs (3 ms)
    ✓ renders register button (2 ms)
Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        0.881 s, estimated 1 s
Ran all test suites matching /Register.test.jsx/i.

```


| ObjectDetails.test.jsx - Cristian Javier Medina Barrios | |
|---|---|
| Descripción del componente: | El componente ObjectDetails muestra información detallada de un objeto perdido, incluyendo su estado y ubicación. |
| Pruebas realizadas: | <ul style="list-style-type: none">• Visualización de la tarjeta del objeto con todos sus elementos (imagen, título, detalles).• Validación de la presentación de detalles específicos (categoría, estado, ubicación).• Presencia del botón "Reclamar Objeto". |
| Código del test | |
| <pre>1 import React from 'react'; 2 import { render, screen } from '@testing-library/react'; 3 import { BrowserRouter } from 'react-router-dom'; 4 import ObjectDetails from '../templates/objectDetails/ObjectDetails'; 5 6 // Mock de los componentes 7 jest.mock('../organisms/header/Header', () => { 8 const Header = () => <div data-testid='mock-header'>Header</div>; 9 Header.displayName = 'Header'; 10 return Header; 11 }); 12 13 jest.mock('../organisms/objectCard/ObjectCard', () => { 14 const ObjectCard = (props) => (15 <div data-testid='mock-object-card'> 16 21 <h1 data-testid='object-title'>{props.title}</h1> 22 <div data-testid='object-details'> 23 <p>Categoría: {props.category}</p> 24 <p>Estado: {props.state}</p> 25 <p>Ubicación: {props.location}</p> 26 <p>Fecha: {props.date}</p> 27 <p>Descripción: {props.description}</p> 28 <p>Contacto: {props.contactInfo}</p> 29 </div> 30 <button data-testid='claim-button'>Reclamar Objeto</button> 31 </div> 32); 33 ObjectCard.displayName = 'ObjectCard'; 34 return ObjectCard; 35 }); 36</pre> | |

```

37   jest.mock('../organisms/footer/Footer', () => {
38     const Footer = () => <div data-testid='mock-footer'>Footer</div>;
39     Footer.displayName = 'Footer';
40     return { Footer };
41   });
42
43   describe('ObjectDetails Component', () => {
44     beforeEach(() => {
45       render(
46         <BrowserRouter>
47           <ObjectDetails />
48         </BrowserRouter>,
49       );
50     });
51

```

Resultado de la ejecución

```

PS C:\Users\Usuario\Desktop\ADESK\Software_Engineering\Proyecto\frontend> npm test ObjectDetails.test.jsx
> frontend@0.0.0 test
> jest ObjectDetails.test.jsx

jest-haste-map: duplicate manual mock found: fileMock
  The following files share their name; please delete one of them:
    * <rootDir>\__mocks__\fileMock.cjs
    * <rootDir>\__mocks__\fileMock.js

(node:6092) [DEP0040] DeprecationWarning: The 'punycode' module is deprecated. Please use a userland alternative instead.
(Use 'node --trace-deprecation ...' to show where the warning was created)
PASS src/test/ObjectDetails.test.jsx
  ObjectDetails Component
    ✓ renders header and footer (20 ms)
    ✓ renders object card with details (3 ms)
    ✓ displays correct object information (5 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        0.911 s, estimated 1 s
Ran all test suites matching /ObjectDetails.test.jsx/i.

```

Lecciones aprendidas y dificultades

Nuestros principales desafíos fueron entender la configuración inicial de Jest y la correcta implementación de los mock, tomando en cuenta el uso de la librería de babel para transpilar el código. Además, la integración de las pruebas necesito de varios ajustes, ya que tuvimos problemas como errores en la transformación de módulos, conflictos entre import/export.

Otra dificultad fue mantener la consistencia en los criterios de prueba, ya que trabajamos con múltiples componentes que interactúan entre sí. Establecer estándares claros para la estructuración de las pruebas nos permitió garantizar resultados confiables y facilitar la escalabilidad del proyecto.

Al testear la plataforma pudimos identificar aspectos positivos y áreas de mejora en nuestro proceso de desarrollo. Pudimos comprobar que implementar pruebas unitarias nos permite asegurar una correcta funcionalidad de cada componente del frontend. Detectando errores de forma temprana y también nos permitió reforzar la calidad, mantenibilidad y escalabilidad del código. En general la realización de los testing fortalecieron nuestra confianza en el código.

