Documento de Pruebas Unitarias del Sistema de Reporteo "Incidentia" para Fundación por México

Índice

Preparación del Entorno

Antes de comenzar con las pruebas, asegúrate de haber configurado correctamente el entorno de pruebas. Para la ejecución de pruebas se instala la dependencia Jest (si se siguió el manual de usuario de manera adecuada, esta dependencia ya debe estar instalada) y otras dependencias necesarias como react-testing-library. Además, verifica que los componentes que se van a probar estén importados y configurados adecuadamente.

Directorio de Pruebas

Es una buena práctica organizar las pruebas unitarias en una estructura de directorios coherente. A continuación, se muestra en donde se encuentra la ubicación de los archivos de pruebas en este proyecto.

```
Incidentia/front/tests
```

Nota

Las pruebas que se encuentran en ese directorio funcionan como se verá a continuación con las capturas de pantalla. Sin embargo, para poder ejecutarlas de manera exitosa tuvimos que comentar secciones de código de los archivos en donde se encuentran definidos los componentes a probar debido a que la herramienta utilizada, `Jest` no reconocía los imports de diferentes módulos o cierta sintaxis de javascript.

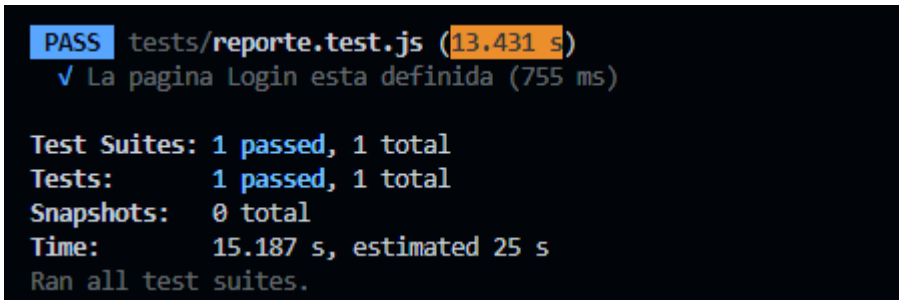
Por tanto, si se desean probar las pruebas directamente, es probable que no pasen, pero es por la razón descrita anteriormente y no otra cosa.

Pruebas Unitarias

A continuación, se describen las pruebas unitarias realizadas en los componentes más importantes de la aplicación web y lo que prueban:

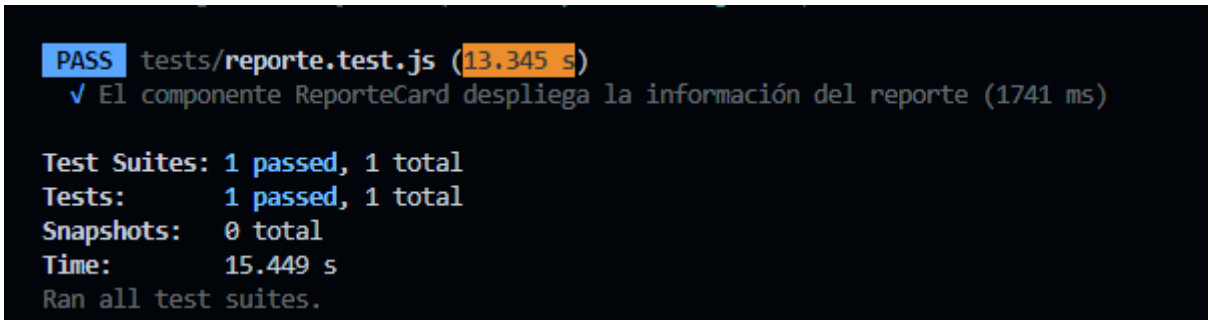
Pruebas para `LoginPage`

1. **Prueba 1: Comprobar que la página Login está definida**
    - **Descripción**: Esta prueba verifica que el componente `LoginPage` está definido en la aplicación web.
    - **Cómo funciona**: Utilizamos `render` de `@testing-library/react` para renderizar el componente y luego seleccionamos el componente LoginPage por su clase usando `querySelector`. Por último, usamos las funciones `expect` y `toHaveTextContent` de Jest para comprobar que en efecto está definida.



Pruebas para `ReporteCard`

2. **Prueba 2: Comprobar renderizado de la información que recibe el componente ReporteCard sin errores**
    - **Descripción**: Esta prueba verifica que el componente pueda renderizar la información que recibe sin errores.
    - **Cómo funciona**: Utilizamos `render` de `@testing-library/react` para renderizar el componente y luego verificamos que el contenedor contenga toda la información que recibe con la funciones `expect` y `toHaveTextContent` de Jest



3. **Prueba 3: Verificar que el botón para actualizar el estatus del reporte está definido dentro del componente**
    - **Descripción**: Esta prueba verifica que el botón para actualizar estatus está definido cuando se renderiza un componente ReporteCard.
    - **Cómo funciona**: Renderizamos el componente `ReporteCard` y luego seleccionamos el botón por su clase utilizando `querySelector`. Una vez seleccionado, utilizamos la función `expect` y `toBeDefined` de Jest.

```
PASS tests/reporte.test.js (17.269 s)
  √ el botón 'actualizar estatus' se encuentra definido en el componente ReporteCard (1552 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        19.243 s, estimated 38 s
Ran all test suites.
```

4. **Prueba 4: Verificar que todos los reportes se despliegan en la página**

    - **Descripción**: Esta prueba verifica todos los reportes se renderizan.
    - **Cómo funciona**: Renderizamos todos los componentes `ReporteCard` definidos y luego los seleccionamos por su clase `querySelectorAll`. Una vez seleccionados, utilizamos la función `expect` y `toBe` de Jest para comparar la longitud de los arreglos definidos. Si la longitud es la misma después de la prueba entónces se cumple la prueba

```
PASS ./reporte.test.js (11.931 s)
  √ Verificamos que se renderizan todos los reportes en la página (1273 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        13.692 s, estimated 17 s
Ran all test suites matching /reporte.test.js/i.
```

Pruebas para `AulaCard`

5. **Prueba 5: Comprobar renderizado de la información que recibe el componente AulaCard sin errores**

    - **Descripción**: Esta prueba verifica que el componente pueda renderizar la información que recibe sin errores.
    - **Cómo funciona**: Utilizamos `render` de `@testing-library/react` para renderizar el componente y luego verificamos que el contenedor contenga toda la información que recibe con la funciones `expect` y `toHaveTextContent` de Jest

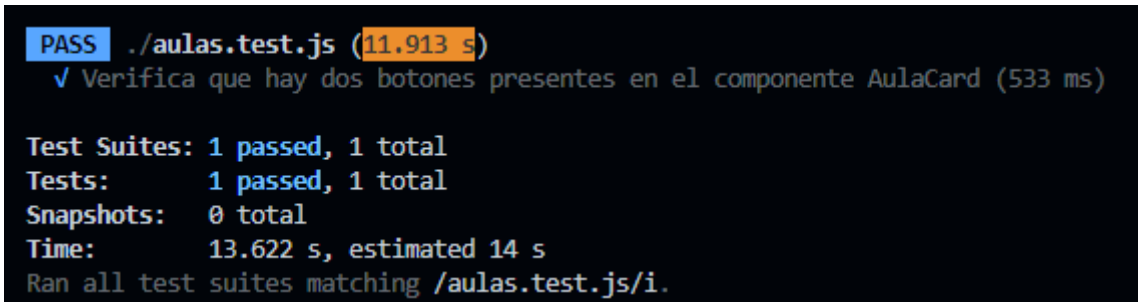
```
PASS ./aulas.test.js (12.094 s)
  √ Verifica que el componente AulaCard despliega la información que se le pasa como props (528 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        14.08 s
Ran all test suites matching /aulas.test.js/i.
```

6. **Prueba 6: Comprobar renderizado de botones para ver reportes pendientes y reportes archivados del componente AulaCard**

    - **Descripción**: Esta prueba verifica que el componente pueda renderizar los botones `Ver Reportes` y `Ver Archivados`
    - **Cómo funciona**: Utilizamos `render` de `@testing-library/react` para renderizar el componente y luego verificamos que el contenedor una longitud igual a 2 con las funciones

`expect` y `toHaveTextContent` de Jest.



Estas pruebas aseguran que los componentes más importantes funcionan según lo esperado y que se muestran correctamente en la aplicación. Tenemos más componentes que están basados en la misma lógica de renderizado que los componentes probados anteriormente.

Ejecución de Pruebas

Para ejecutar las pruebas, utiliza el siguiente comando:

```
npm run test
```