Pruebas de Software

Tecnológico de Monterrey,

Campus Santa Fe - 2022

TC2007B.401

TC2007B.401 - G3

Presentan:

Ana Paula Katsuda - A01025303

Regina Rodríguez Sánchez - A01284329

Salvador Federico Milanés Braniff - A01029956

Introducción

Las pruebas de software se utilizan para poder evaluar y verificar que un producto o aplicación de software funcione adecuadamente. Estas pruebas se pueden hacer a través de distintos procesos y técnicas, que, dependiendo de las necesidades particulares del software se utilizan. Además, las pruebas son beneficiosas para prevenir errores, reducir el costo de desarrollo y mejorar el rendimiento (IBM, s.f.). Con lo anterior en mente, el objetivo del presente escrito, es profundizar los tipos de pruebas existentes, las herramientas utilizadas para estas y aplicar dichas pruebas.

En cuanto a los tipos de pruebas, existen múltiples y tienen propósitos distintos. En términos generales, existen las pruebas funcionales y las no funcionales. Haciendo referencia a las pruebas funcionales, estas tienen el propósito de garantizar que se cumplan los comportamientos del software y para comprobar las características críticas del mismo. Por otro lado, las pruebas no funcionales se relacionan con seguridad, confiabilidad y elementos que no tienen que ver con la funcionalidad del software en sí. A continuación, se explican con mayor detalle algunas de las pruebas existentes:

- Pruebas locales

  O unitarias; se enfocan en validar el comportamiento de un componente en particular de forma independiente. De esta manera, el código invoca individualmente un bloque en el sistema y revisa una “asumpción” sobre el comportamiento esperado (Munot, K., 2019).

- Pruebas de integración

  Las pruebas de integración verifican el funcionamiento en conjunto de los distintos módulos o servicios presentes en la app. Por ejemplo, puede probarse la interacción con la base de datos. Las pruebas de integración pueden ser costosas al depender de varias partes de la aplicación en funcionamiento. (IBM, s. f.)

- Pruebas alfa

  Pruebas iniciales para validar el funcionamiento de una aplicación; se llevan a cabo al principio del proceso de desarrollo y son seguidas con pruebas beta. (Pruebas alfa, 2022)

- Pruebas de regresión

  Pruebas para averiguar si una aplicación en forma aún funciona como se esperaba tras haber sido actualizada o alterada. Es vital hacerlas cada vez que el código se modifica. (Singh, K. 2022)

- Pruebas dinámicas de validación

  Pruebas que analizan el comportamiento dinámico del código. Para realizarlas, el software debe compilarse y ejecutarse. Se analizan parámetros como el uso de memoria, CPU; el tiempo de respuesta; y el rendimiento general del software.

  El software se prueba mediante los valores de entrada y se analizan los valores de salida. (IBM, s. f.)

- Pruebas bajo condiciones frontera

  Consisten en probar un componente cuya estructura está basada en condiciones de un tipo de dato particular, un rango determinado, una extensión en dígitos o caracteres, etcétera; Validando que el software filtre adecuadamente los datos válidos como parte del funcionamiento (es decir, sin obstruirse), y/o que tenga un comportamiento determinado en caso de recibir un dato que no cumple con las condiciones definidas. (Ramirez. C, 2022)

Herramientas de pruebas de software

Nombre de la herramienta	**Tipos de prueba y ejemplos**	**Licencia (USD)**	**Plataforma/ lenguajes**	**Particularidades**
Jest	Pruebas locales: *render()* Pruebas de integración: *screen*	Free	JavaScript: Babel, TypeScript, Node, React, Angular, Vue	Zero config- Trabaja ‘out of the box’ sin configuraciones, en la mayor parte de los proyectos de Javascript
Selenium	Pruebas bajo condiciones frontera: *selenium.common.exceptions*	Freemium open-source	La mayoría de los lenguajes de programación contemporáneos.	Portable- Portátil y disponible en Windows, Linux, macOS, Android, Firefox y Solaris

WebdriverIO	Pruebas locales: *touchAction* *waitForEnabled* *setValue* Pruebas dinámicas de validación: *abort* *respond* *restore*	Free	Webdriver protocol, Chrome Devtools protocol: Node	Large application support: Soporte a cualquier app web o móvil desarrollada con librerías modernas como Vue y React.
Cypress	Pruebas locales Pruebas de integración: *group* *api*	Free; + priced versions: Team - $75.00/m Business - $300/m Enterprise - A cotización (contacto)	Front-end: Javascript	Local testing- Permite hacer pruebas de automatización a nivel local en el sistema. Parallelization capabilities
Robot Framework	Pruebas de integración: *OperatingSystem* Pruebas bajo condiciones frontera: *String*	Free open-source	Robotics: Python	Open-source, Library support- Permite librerías escritas tanto en python o Java
EarlGrey	Pruebas locales: *UnitTests.xcodeproj* Pruebas bajo condiciones frontera: *GREYcondition*	Free	iOS native: Objective-C, Swift	Automatic Tracking- Detecta iterativamente cambios en los requests de Network y UI

Caso de prueba

Para fines del presente escrito, se decidió utilizar Jest para realizar pruebas a una aplicación que lee, convierte y muestra datos de un ‘csv’ en forma de tabla. Se hicieron pruebas unitarias de cada uno de los componentes que conforman la aplicación como se muestra a continuación (link al repositorio):

Archivos de Prueba

AskCSV.test.js

HW-CSV-Convert > csv-read-convert-final > src > components > JS AskCSV.test.js > ...

```
/* Test for the AskCSV component -- Tests render


Ana Paula Katsuda - A01025303
Regina Rodriguez - A01284329
Salvador Federico Milanes - A01029956 */


import { render, screen, cleanup } from "@testing-library/react";
// Importing the jest testing library
import '@testing-library/jest-dom'
import AskCSV from "./AskCSV";

// Reset DOM
afterEach(() => {
    cleanup();
});

// Test for renderer
describe('AskCSV component', () => {
    // Component render --> mock screen
    render(<AskCSV />);
    // save into variable
    const csv = screen.getByTestId("csv");
    // Test
    test('CSV renderer', () => {
        expect(csv).toBeInTheDocument();
    });
})
```

BuildTable.test.js

```jsx
/* Test for the BuildTable component -- Tests render and text


Ana Paula Katsuda - A01025303
Regina Rodriguez - A01284329
Salvador Federico Milanes - A01029956 */


import { render, screen, cleanup } from "@testing-library/react";
// Importing the jest testing library
import '@testing-library/jest-dom'
import BuildTable from './BuildTable';

// Reset DOM
afterEach(() => {
    cleanup();
});

// Test for renderer
describe('Build Table Component', () => {
    // Variables to fill first row and first line of values
    const firstRow = ['Name', 'ID'];
    const data = [['Ana Paula', '1'], ['Regina', '2'], ['Salvador', '3']];

    // Component render --> mock screen
    render(<BuildTable tableRows={firstRow} values={data} id={'table build'} />);
    // save into variable
    const tBuild = screen.getByTestId("table build");

     // Test that table is in document
     test('Table renderer', () => {
         expect(tBuild).toBeInTheDocument();
     });

     test('Table data', () => {
         expect(tBuild).toHaveTextContent('Name');
         expect(tBuild).toHaveTextContent('ID');
         expect(tBuild).toHaveTextContent('Ana Paula');
         expect(tBuild).toHaveTextContent('1');
         expect(tBuild).toHaveTextContent('Regina');
         expect(tBuild).toHaveTextContent('2');
         expect(tBuild).toHaveTextContent('Salvador');
         expect(tBuild).toHaveTextContent('3');
     })
 })
```

convert.test.js

HW-CSV-Convert > csv-read-convert-final > src > components > JS convert.test.js > ...

```jsx
/* Code that tests data conversions


Ana Paula Katsuda - A01025303
Regina Rodriguez - A01284329
Salvador Federico Milanes - A01029956 */


// Importing the jest testing library
import '@testing-library/jest-dom'
import {changeGrade, changeDate, noSurname, format_line} from './convert'

// Many unit tests for functions
describe('Individual functions', () => {
    // Test grade conversion
    test('grade conversion', () => {
      expect(changeGrade(98)).toBe('A+');
      expect(changeGrade(95)).toBe('A');
      expect(changeGrade(91)).toBe('A-');
      expect(changeGrade(89)).toBe('B+');
      expect(changeGrade(85)).toBe('B');
      expect(changeGrade(81)).toBe('B-');
      expect(changeGrade(79)).toBe('C+');
      expect(changeGrade(75)).toBe('C');
      expect(changeGrade(71)).toBe('C-');
      expect(changeGrade(68)).toBe('D+');
      expect(changeGrade(66)).toBe('D');

    });
```

```
  // Test date conversion
  test('date conversion', () => {
    expect(changeDate('24/06/2021')).toBe('06/24/2021');
    expect(changeDate('18/12/2021')).toBe('12/18/2021');
    expect(changeDate('23/06/2021')).toBe('06/23/2021');
  });
  // Test remove surname
  test('remove surname', () => {
    expect(noSurname('Regina Rodriguez Sanchez')).toBe('Regina Rodriguez');
    expect(noSurname('Gilberto Echeverria Furio')).toBe('Gilberto Echeverria');
    expect(noSurname('Salvador Milanes Braniff')).toBe('Salvador Milanes');
  });

});

// Test format line
describe('Complete line', () => {
  test('convert line', () => {
    let line = [];
    let result = [];
    line = ['1', 'Regina Rodriguez Sanchez', 'A01284329', '24/06/2021', '93'];
    result = ['1', 'Regina Rodriguez', 'A01284329@tec.mx', '06/24/2021', 'A'];
    expect(format_line(line)).toStrictEqual(result);
    line = ['2', 'Gilberto Echeverria Furio', 'a09876543', '15/12/2022', '85'];
    result = ['2', 'Gilberto Echeverria', 'a09876543@tec.mx', '12/15/2022', 'B'];
    expect(format_line(line)).toStrictEqual(result);
    line = ['12', 'Salvador Milanes Braniff', 'A01029956', '07/03/2021', '75'];
    result = ['12', 'Salvador Milanes', 'A01029956@tec.mx', '03/07/2021', 'C'];
    expect(format_line(line)).toStrictEqual(result);
  });
});
```

DisplayResults.test.js

HW-CSV-Convert > csv-read-convert-final > src > components > JS DisplayResults.test.js > ...

```
/* Test for the DisplayResults component -- Tests render and text



Ana Paula Katsuda - A01025303
Regina Rodriguez - A01284329
Salvador Federico Milanes - A01029956 */


import { render, screen, cleanup } from "@testing-library/react";
// Importing the jest testing library
import '@testing-library/jest-dom'
import DisplayResults from './DisplayResults';

// Reset DOM
afterEach(() => {
    cleanup();
});

// Test for renderer
describe('Display Results Component', () => {
    // Variables to fill first row and first line of values
    const firstRow = ['Name', 'ID'];
    const data = [['Ana Paula', '1'], ['Regina', '2'], ['Salvador', '3']];
    // Variable for new data (second table)
    const newData = [['Akemi', '21'], ['Gina', '22'], ['Salva', '23']];

    // Component render --> mock screen
    render(<DisplayResults tableRows={firstRow} values={data}
        newValues={newData}/>);
```

```
30
31      // save into variables
32      const firstTable = screen.getByTestId("og table");
33      const secondTable = screen.getByTestId("new table");
34
35      // Test that tables are in document
36      test('Table renderer', () => {
37          expect(firstTable).toBeInTheDocument();
38          expect(secondTable).toBeInTheDocument();
39      });
40
41      // Test text in tables
42      test('Table data', () => {
43          // For original table
44          expect(firstTable).toHaveTextContent('Name');
45          expect(firstTable).toHaveTextContent('ID');
46          expect(firstTable).toHaveTextContent('Ana Paula');
47          expect(firstTable).toHaveTextContent('1');
48          expect(firstTable).toHaveTextContent('Regina');
49          expect(firstTable).toHaveTextContent('2');
50          expect(firstTable).toHaveTextContent('Salvador');
51          expect(firstTable).toHaveTextContent('3');
52
53          // For edited table
54          expect(secondTable).toHaveTextContent('Name');
55          expect(secondTable).toHaveTextContent('ID');
56          expect(secondTable).toHaveTextContent('Akemi');
57          expect(secondTable).toHaveTextContent('21');
58          expect(secondTable).toHaveTextContent('Gina');
59          expect(secondTable).toHaveTextContent('22');
60          expect(secondTable).toHaveTextContent('Salva');
61          expect(secondTable).toHaveTextContent('23');
62      })
63  })
```

Tests passed

```
PASS  src/components/convert.test.js
PASS  src/components/DisplayResults.test.js
PASS  src/components/BuildTable.test.js
PASS  src/components/AskCSV.test.js

Test Suites: 4 passed, 4 total
Tests:       9 passed, 9 total
Snapshots:   0 total
Time:        6.711 s
Ran all test suites.
```

Fuentes

Everett, G., McLeod, R. (2007). Software testing. Recuperado el 23 de agosto, de http://worldcolleges.info/sites/default/files/software-testing-testing-across-the-entire-software-development-life-cycle.9780471793717.28214.pdf

Getting Started · Jest. (2022). Recuperado el 24 de agosto, de https://jestjs.io/docs/getting-started

GitHub - google/EarlGrey: iOS UI Automation Test Framework. (2022). Recuperado el 24 de agosto, de https://github.com/google/EarlGrey

Herramientas de Desarrollo de Software | Software de Desarrollo. (2022). Recuperado el 25 de Agosto, de https://okhosting.com/blog/herramientas-de-desarrollo-de-software/

Introduction | WebdriverIO. (2022). Recuperado el 24 de agosto, de: https://webdriver.io/docs/api/

Munot, K. (2019). Unit-testing important role in software development. Recuperado el 25 de agosto, de https://medium.com/nonstopio/unit-testing-important-role-in-software-development-1f52f7c810f8

Prueba alfa. (2022). Recuperado el 24 de Agosto, de https://isolution.pro/es/t/software-testing-dictionary/alpha-testing/prueba-alfa

¿Qué es la prueba de software y Cómo Funciona? IBM. (s.f.). Recuperado el 23 de agosto de https://www.ibm.com/mx-es/topics/software-testing

Ramirez. C (2022). ¿Qué es el Análisis del Valor Límite (BVA) de la Técnica de Pruebas de Caja Negra? | . Recuperado el 25 de Agosto, de https://ca-ra.org/es/an%C3%A1lisis-del-valor-l%C3%ADmite-una-t%C3%A9cnica-de-pruebas-de-caja-negra/

Robot Framework documentation. (2022). Recuperado el 24 de Agosto, de https://robotframework.org/robotframework/

The Selenium Browser Automation Project. (2022). Recuperado el 24 de agosto, de https://www.selenium.dev/documentation/

Singh, K. (2022). Pruebas de regresión: todo lo que necesita saber. Recuperado el 24 de agosto, de https://geekflare.com/es/regression-testing-tools/

Turrado, J. (2020). Qué son las pruebas de software - campusMVP.es. Recuperado el 23 de agosto, de: https://www.campusmvp.es/recursos/post/que-son-las-pruebas-de-software.aspx

Why Cypress? | Cypress Documentation. (2022). Recuperado del 23 de agosto, de https://docs.cypress.io/guides/overview/why-cypress

-

Trabaja en equipo lo siguiente:

2.1 Comentar cada una de las pruebas de software documentadas.

2.2 Investigar las herramientas de pruebas automatizadas existentes actualmente para cada uno de los diferentes tipos de pruebas, de la lista dada arriba (comerciales y open source), identificando para cada una: Nombre, tipo de prueba, licencia (comercial u open source), plataforma y/o lenguajes, particularidades.

2.3 Seleccionar una herramienta de pruebas y aplicarle un tipo de prueba (de la lista de arriba) a un producto/componente de software, preferentemente en alguna de las aplicaciones móviles que desarrollaste como actividad en el módulo **4: Desarrollo de aplicaciones móviles**. Diseña y documenta el caso de prueba.

2.4 Entregar un reporte gerencial de resultados de las pruebas realizadas.

2.5 Una vez terminada la actividad, el equipo realizará la presentación de su investigación y una demostración de la herramienta de prueba, mostrando sus resultados en la herramienta.

Pruebas de software			
Criterios	**Calificaciones**		**Pts**
Pruebas de software	**10 pts** **Cumple** Las pruebas de software están completas y documentadas	**0 pts** **No cumple** Las pruebas de software solicitadas están completas y no lo suficientemente documentadas	10 pts
Herramientas de prueba automatizadas	**15 pts** **Cumple** La investigación está completa en los rubros solicitados	**0 pts** **No cumple** La investigación está incompleta con base en lo especificado	15 pts
Aplicación de la herramienta de prueba	**25 pts** **Cumple** Se aplicó la herramienta de prueba en una aplicación hecha en clase. Se diseñó y documentó el caso de prueba.	**0 pts** **No cumple** Se aplicó la herramienta de prueba en una aplicación hecha en clase. Se diseñó el caso de prueba sin haberlo documentado.	25 pts
Reporte gerencial	**20 pts** **Cumple** El reporte gerencial de pruebas está completo en todos los rubros especificados	**0 pts** **No cumple** El reporte gerencial de pruebas está incompleto.	20 pts
Presentación de resultados	**30 pts** **Cumple** El equipo presentó los resultados de lo investigado, entregando oportuna y completamente lo especificado	**0 pts** **No cumple** El equipo presentó los resultados de lo investigado, entregando con retraso el documento solicitado	30 pts
			Puntos totales: 100