TECHNOLOGY

INVESTIGACIÓN APLICADA 2

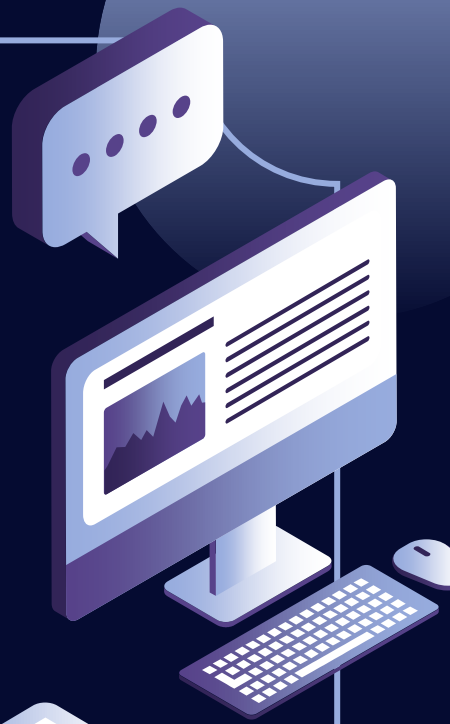
Herramientas de Testing

Integrantes:

Magaña Martínez, Samuel Eduardo	MM220035
Padilla Ramírez, Alexandra Guadalupe	NR221019
Pineda Fuentes, Geovanny Arturo	PF211251
Trujillo Osorio, Erick Geovanni	TO220989
Rodriguez Sanchez, Enrique Ernesto	RS132134
Villalobos Eguizábal, Oscar Alejandro	VE220589

Grupo: 01T

Repositorio GitHub:
https://github.com/enriqueRodriguez/InvestigacionAplicada2

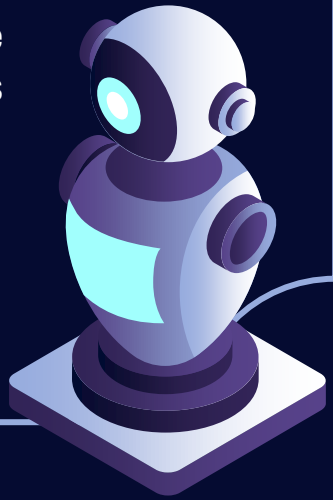


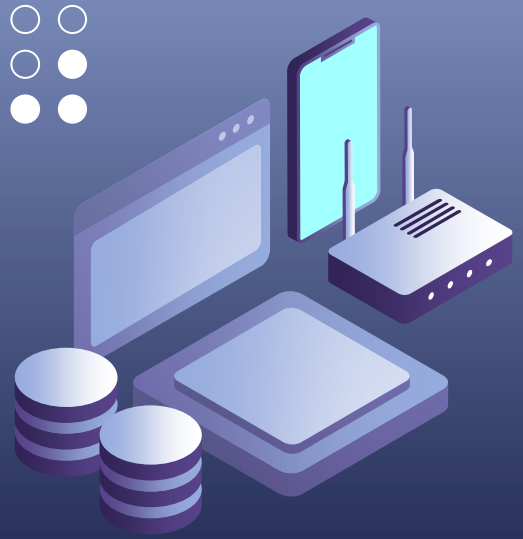
INVESTMENTS

QUOTE

Herramientas de Testing

Las herramientas de testing son aplicaciones o frameworks diseñados para automatizar y facilitar el proceso de prueba de software. Su objetivo principal es verificar que una aplicación funcione correctamente, cumpla con los requisitos especificados y esté libre de errores.





JEST

Jest es un testing framework de JavaScript de código abierto diseñado para simplificar y agilizar el proceso de escritura y ejecución de pruebas. A su vez, Jest es el framework de pruebas más popular en la actualidad, utilizado en grandes empresas como Facebook, Airbnb, Twitter o Spotify.

Características Principales

1. No requiere configuración:

Jest está bien diseñado y es fácil de usar. Los evaluadores no necesitan realizar ninguna configuración para comenzar a trabajar con Jest.

2. Interfaz de línea de comandos de la aplicación:

Jest ofrece una herramienta CLI que brinda a los usuarios más control sobre sus pruebas.

3. Jest hace que los casos de prueba sean aislados:

Jest tiene una función de aislamiento para aumentar el rendimiento de las pruebas. Las pruebas individuales se ejecutan en paralelo, lo que permite realizar múltiples tareas.

4. Pruebas de código asincrónico:

Dado que JS es un lenguaje asincrónico, el marco de pruebas también debe poder probar fragmentos de código asincrónicos. Jest se puede utilizar para probar fragmentos de código asincrónicos como promesas, devoluciones de llamadas, etc.

Características Principales

5. Instantáneas:

Jest nos permite probar fácilmente las propiedades de estos objetos grandes mediante instantáneas. Jest garantiza que los datos en estos objetos grandes tengan un valor correcto y que el tipo de datos también sea correcto.

6. Funciones simuladas:

Jest nos permite probar el flujo de control de la aplicación mediante funciones simuladas que no implementan la funcionalidad de una función, sino que actúan como un marcador de posición.

7. API bien mantenida:

La API de Jest no solo es rápida, sino que también ofrece una experiencia sin complicaciones. Cada punto final de la API es intuitivo, comprensible y está bien documentado.

Ventajas de Jest

- Sintaxis intuitiva y legible
- Ejecución rápida y paralela
- Prueba de instantáneas
- Mocking integrado
- Versatilidad
- Cobertura de código integrada
- Facilidad de uso y configuración
- Amplia comunidad y soporte activo
- Integración con herramientas populares

Desventaja de Jest

- Curva de aprendizaje inicial
- Uso intensivo de recursos
- Tamaño de la instalación
- Configuración predeterminada
- Posibles actualizaciones disruptivas
- Falta de soporte para navegadores

Funcionalidades

- **Pruebas de Código Asíncrono**: Permite probar funciones que manejan promesas, callbacks y otras operaciones asincrónicas, lo cual es clave en JavaScript.
- **Instantáneas (Snapshots)**: Jest permite capturar el estado de los objetos en un punto dado del tiempo y compararlo en futuras ejecuciones para verificar que no haya cambiado inesperadamente. Estas instantáneas pueden ser almacenadas en archivos separados o incrustadas en el código.
- **Funciones de Mock**: Permite simular el comportamiento de funciones o módulos, lo que es útil para probar el flujo de control sin depender de las implementaciones reales.
- **Modo Interactivo**: Al trabajar junto con sistemas de control de versiones como Git, Jest puede ejecutar solo los tests relacionados con los cambios recientes, mejorando la eficiencia durante el desarrollo.
- **Cobertura de Código**: Ofrece informes detallados sobre qué partes del código están cubiertas por las pruebas, lo cual es importante para asegurar que todas las funcionalidades clave estén probadas.

Tipo de pruebas que soporta

1. Pruebas Unitarias:
Estas prueban unidades individuales del código, como funciones, clases o componentes, de manera aislada para asegurarse de que se comporten como se espera.

2. Pruebas de Integración:
Verifican cómo interactúan diferentes módulos o componentes del sistema entre sí. Son útiles para probar interacciones entre diferentes partes del código, asegurando que se integren correctamente.

3. Pruebas de Instantáneas (Snapshot Testing):
Utilizadas principalmente en pruebas de interfaces de usuario, capturan la salida renderizada de un componente y la comparan con una instantánea almacenada para detectar cambios no intencionados.

4. Pruebas Asíncronas:
Permiten verificar el comportamiento de código que se ejecuta de manera asíncrona, como funciones que utilizan promesas o async/await. Jest proporciona soporte nativo para probar este tipo de código, asegurando que las operaciones asíncronas, como solicitudes a APIs o temporizadores, se ejecuten correctamente

Integraciones disponibles

Jest se integra de manera sencilla con diversas herramientas y frameworks para mejorar la experiencia de pruebas en proyectos JavaScript. Algunas de las integraciones más importantes incluyen:

- **React:** Jest se integra de manera muy eficiente con React, permitiendo realizar pruebas unitarias y de componentes. Junto con herramientas como react-testing-library o Enzyme, puedes crear pruebas detalladas para asegurar el correcto comportamiento de los componentes en diferentes escenarios.

- **Vue.js:** Para desarrolladores que utilizan Vue.js, Jest se combina con Vue Test Utils para permitir pruebas unitarias y de interacción de los componentes, proporcionando un entorno robusto y fácil de configurar.

- **Node.js:** Al ser parte del ecosistema de JavaScript, Jest funciona bien con aplicaciones construidas con Node.js, lo que facilita la creación de pruebas para backend, incluyendo la validación de controladores y rutas.

Comparativa con otras herramientas similares

Característica	Jest	Mocha	Cypress
Facilidad de Uso	Muy fácil de configurar y usar. Viene con un entorno completo.	Requiere configuración adicional con bibliotecas como Chai y Sinon.	Interfaz gráfica intuitiva, configuración simple.
Tipo de Pruebas	Unitarias, de integración, de snapshots, asíncronas.	Unitarias, de integración, asíncronas.	End-to-end, integración, UI.
Integración	Excelente con React, soporte para mocks y spies integrados.	Flexible, se puede integrar con Chai para aserciones y Sinon para mocks.	Diseñado para pruebas en el navegador con soporte para pruebas visuales y de rendimiento.
Casos de Uso Típicos	Pruebas unitarias en aplicaciones JavaScript y React, pruebas de componentes y lógica de negocios.	Pruebas unitarias y de integración con gran flexibilidad y personalización.	Pruebas end-to-end para validar el comportamiento del usuario en aplicaciones web.

REFERENCIAS

- C, D. E. (2022, 12 septiembre). Test unitarios en Node.js con Jest. *Medium*. https://medium.com/@diego.coder/test-unitarios-en-javascript-con-jest-4f120f5e7124
- Villalta Ortiz, D. A. (s. f.). *Ventajas de JEST*. Issuu. https://issuu.com/denisvillalta/docs/presentaci_n_jest_grupo_az_l.pptx/s/29246697
- Villalta Ortiz, D. A. (s. f.-a). *Desventajas de JEST*. Issuu. https://issuu.com/denisvillalta/docs/presentaci_n_jest_grupo_az_l.pptx/s/29246698
- *Jest*. (s. f.). Jest.js. https://jestjs.io/
- Singhal, A. (2023, 4 mayo). *Jest testing Framework Introduction*. Scaler Topics. https://www.scaler.com/topics/nodejs/jest-testing-framework/
- Testsigma, G. (2023, 27 diciembre). Jest vs Mocha vs Jasmine: Which Framework to choose. *testsigma*. https://testsigma.com/blog/jest-vs-mocha-vs-jasmine/
- Gimeno, A. (2022, 30 mayo). *Jest testing: Top features and how to use them - LogRocket Blog*. LogRocket Blog. https://blog.logrocket.com/jest-testing-top-features/
- Li, T. (2023, 5 diciembre). *Playwright vs. Jest - A Comprehensive Guide to Choosing the Right Testing Framework*. Headspin. https://www.headspin.io/blog/playwright-vs-jest-which-framework-to-consider