**Pruebas para un código**

En el mundo del desarrollo de software, garantizar la calidad y funcionalidad del producto es de vital importancia.

Así lo pone de manifiesto un reciente informe en el que se estima que los defectos de software cuestan a las empresas alrededor de $1.7 billones de dólares en pérdidas cada año.

Contar con un enfoque sólido de pruebas para evitar problemas costosos y entregar software de alta calidad es, pues, fundamental para cualquier proyecto.

A lo largo de este artículo examinaremos las diferencias clave, ventajas y casos de uso de cada tipo de prueba, para que puedas tomar decisiones informadas y eficientes al implementar tus estrategias de pruebas.

**Significado y alcance de las pruebas unitarias**

Las pruebas unitarias se posicionan como la primera línea de defensa en la detección de errores dentro del desarrollo de software.

Diseñadas para implementarse a nivel local, estas pruebas se encargan de validar los componentes más elementales de la lógica de tu código.

Ya sea una función, una clase o cualquier otra entidad de código, las pruebas unitarias están destinadas a evaluar cada característica y posible ruta lógica.

En el ideal de una cobertura de código al 100% se escribiría una prueba unitaria para cada flujo posible en el código. En la práctica, una buena cobertura de código se acerca más a un 80%.

La amplitud de la cobertura te proporciona un nivel adicional de confianza en tu código, ya que una vez que tus pruebas unitarias son superadas, puedes confiar en la solidez de tu software.

**Ventajas y beneficios de los test unitarios en el desarrollo de software:**

* Primera línea de defensa contra errores: Los tests unitarios son la primera línea de defensa contra los bugs en el código. Están diseñados para correr localmente y verificar los aspectos más fundamentales de la lógica de tu código.
* Cobertura de código: Los tests unitarios están diseñados para testear cada característica y camino lógico potencial. Al escribir un test unitario para cada flujo posible de tu código se le llama «cobertura de código del 100%». Aunque se considera un buen porcentaje de cobertura de código que ronda el 80%, cuanta más cobertura tengas, más confianza podrás tener en tu código.​
* Detección temprana de bugs: Los tests unitarios son una forma de bajo riesgo de detectar rápidamente los bugs. Se deben usar siempre, sin importar si estás escribiendo código para un pequeño proyecto o desplegándolo para miles de millones de usuarios​.

**Que son los test end to end y como se diferencian de otros tipos de pruebas**

Los test end-to-end o E2E son una forma de prueba de software en la que se busca asegurar que los flujos de trabajo funcionen de principio a fin, tal como lo experimentaría un usuario final.

A diferencia de las pruebas unitarias e integradas, las pruebas end-to-end buscan probar el producto de la misma forma en que un usuario real lo experimentaría, validando diferentes subsistemas y capas de la aplicación​​.

Las pruebas unitarias, por otro lado, son tu primera línea de defensa contra los errores.

Estas pruebas están destinadas a verificar las partes más fundamentales de la lógica de tu código, como una función o una clase. La idea es probar cada característica o camino lógico posible en el código​.

Por su parte, las pruebas de integración suelen implicar la comprobación de una funcionalidad específica que tiene dependencias de otra funcionalidad, como una función que llama a otra función.

Estas pruebas buscan verificar la conectividad y la comunicación entre los diferentes componentes de la aplicación​.

En que se diferencia los test end-to-end manuales de los test end-to-end automatizados.

Las pruebas End-to-End, tanto manuales como automatizadas, desempeñan un papel crucial en el aseguramiento de la calidad del software, permitiendo validar la funcionalidad y el rendimiento del sistema en su totalidad.

Aunque ambas pruebas persiguen el mismo objetivo, existen diferencias notables entre ellas en términos de implementación, eficacia, costos y recursos necesarios.

Las pruebas End-to-End manuales son realizadas por un equipo de calidad o un probador, quien simula el comportamiento del usuario final y verifica que el sistema funcione según lo esperado.

Estas permiten identificar problemas de usabilidad y de experiencia del usuario que podrían pasar desapercibidos en una prueba automatizada. Sin embargo, las pruebas manuales pueden ser más lentas, más costosas y más propensas a errores humanos.

Por su parte, las pruebas End-to-End automatizadas utilizan scripts y herramientas de software para ejecutar pruebas de manera programada.

Estas suelen ser más rápidas, más eficientes y más consistentes que las pruebas manuales, ya que pueden ejecutarse repetidamente sin intervención humana.

Las pruebas automatizadas son especialmente útiles para tareas repetitivas y para validar funcionalidades que ya han sido probadas manualmente.

Dicho esto, la creación de scripts de prueba automatizados puede requerir una inversión inicial significativa en términos de tiempo y recursos.Ventajas y desafíos de los test end-to-end en proyectos de desarrollo de software algunas de las ventajas y desafíos más importantes que presentan las pruebas E2E son:

**Ventajas de los test End to End**

* Cobertura de pruebas completa: Los test E2E se enfocan en simular la experiencia del usuario final, lo que permite verificar que todos los componentes del sistema funcionan de manera integrada.
* Detección de errores complejos: Al simular flujos de trabajo reales, las pruebas E2E pueden ayudar a identificar y solucionar problemas complejos que pueden no ser detectados en otros niveles de pruebas.
* Mejora de la confianza en el software: Los E2E test pueden incrementar la confianza en el software, dado que permiten verificar que el sistema completo funciona correctamente antes de su lanzamiento.

**Desafíos de los test End to End**

Complejidad de implementación: Dado que las pruebas E2E implican evaluar el sistema en su totalidad, pueden ser más complejas y costosas de implementar en comparación con pruebas unitarias o de integración.

* Tiempo de ejecución: Las pruebas E2E suelen requerir más tiempo para ejecutarse que otros tipos de pruebas, lo que puede incrementar el tiempo total de desarrollo.
* Falsos positivos: En ocasiones, las pruebas E2E pueden generar falsos positivos debido a problemas temporales o específicos del entorno de prueba. Esto puede traducirse en tiempo de investigación de problemas que no son errores reales en el software.

**Definición y propósito de los test de integración**

Los test de integración son una parte crucial en el proceso de desarrollo de software.

A diferencia de las pruebas unitarias, que se centran en verificar la funcionalidad de componentes individuales de un sistema, las pruebas de integración se ocupan de probar cómo interactúan entre sí diferentes partes de un sistema.

Estas se llevan a cabo en funcionalidades o módulos que dependen de otras funcionalidades, como por ejemplo una función que llama a otra función. El objetivo principal de estas pruebas es comprobar la conectividad y la comunicación entre diferentes componentes de la aplicación.

Existen tres motivos principales para realizar pruebas de integración:

Asegurar la sincronización entre módulos cuando trabajan juntos para llevar a cabo una tarea específica, dado que estos módulos podrían haber sido desarrollados por diferentes equipos o desarrolladores.

Ayudar a validar las interfaces de la aplicación para garantizar que los datos que fluyen de un módulo a otro sean apropiados.

Verificar que la conectividad entre módulos funcione como se concibió originalmente​.

Importancia de los test de integración en el proceso de desarrollo

Las pruebas de integración permiten detectar errores que pueden surgir cuando diferentes componentes o módulos del sistema interactúan entre sí. Hablamos, pues, de errores que no podrían ser detectados por las pruebas unitarias.

Además, dado que diferentes módulos pueden ser desarrollados por diferentes equipos, las pruebas de integración son esenciales para asegurar que todos los módulos trabajen de manera armónica una vez integrados.

Esto ayuda a mantener la calidad del software y a reducir el tiempo y los recursos necesarios para solucionar problemas que surgen debido a una integración defectuosa.

Por último, los test de integración también pueden ayudar a validar las interfaces de la aplicación y a garantizar que los datos fluyan correctamente de un módulo a otro, lo que resulta crucial para el funcionamiento correcto del sistema.

Comparativa y selección de los tipos de pruebas de software: Test unitarios, Test de integración y Test end-to-end (e2e)

Los tres tipos principales de pruebas de software son las pruebas unitarias, las pruebas de integración y las pruebas end-to-end. Veamos sus principales características y diferencias:

**Pruebas End to End (E2E)**

Las pruebas E2E son útiles cuando necesitas validar la funcionalidad y rendimiento de la aplicación en un escenario que simula la experiencia del usuario final.

Estas pruebas implican verificar que todo el sistema funcione correctamente en conjunto, desde las interacciones de la interfaz de usuario hasta la base de datos.

Si estás interesado en probar el producto de la misma manera que un usuario real lo experimentaría, las pruebas E2E serían la elección correcta​​.

**Pruebas de integración de software: qué son, niveles y tipos**

Conoce qué son las pruebas de integración de software, los tipos como Big Bang, Ad Hoc, entre otras y asegura la calidad de tus proyectos de software.

Al igual que una receta de cocina, cada ingrediente es un complemento del platillo final. Por si solos los alimentos aportan nutrientes, olores y sabores que al combinarse con otros pueden cambiar su composición original.

Esta analogía es aplicable en los sistemas de software, en los que deben ejecutarse distintos tipos de pruebas para garantizar su correcto funcionamiento. Cada línea de código es equiparable a un ingrediente, que al mezclarse con otros fragmentos puede llegar a sufrir alteraciones en sus funciones originales.

Las **pruebas de integración de software** son cruciales para validar la operación óptima de todos los elementos de un sistema y a continuación te explicamos en qué consisten y cuáles son los tipos de prueba aplicables de acuerdo a cada proyecto.

**¿Qué objetivo cumplen las pruebas de integración de software?**

Las **pruebas de integración de software**permiten comprobar que estas funciones se desempeñan satisfactoriamente y su relación con otras interfaces es confiable, segura y poco susceptible de riesgos en etapas avanzadas.

De acuerdo con un estudio de la empresa de gestión de pruebas PractiTest, el 42 por ciento de las organizaciones encuestadas incorporan **prácticas de integración** en sus proyectos y el 38 por ciento de ellos participa de manera activa en el mantenimiento de los mismos.

Esta cifra resulta alarmante, ya que más de la mitad de los participantes no tienen cercanía con los procesos de integración continua de sus organizaciones, e incluso desconocen si estos se llevan a cabo, lo que podría desencadenar grandes consecuencias que pongan en riesgo los avances y la reputación de sus desarrollos.

**¿Qué errores pueden detectarse con este tipo de pruebas?**

Las pruebas unitarias son un primer filtro para la detección de fallos en los sistemas, sin embargo, éstas no identifican su relación con otras interfaces. De ahí la importancia de implementar**pruebas de integración de software**.

Entre los problemas más comunes que identifican este tipo de pruebas se encuentran la pérdida de conectividad, el formateo de datos y las respuestas inesperadas.

La detección oportuna de errores puede minimizar el impacto económico y temporal, de manera que los desarrolladores puedan enfocar esfuerzos en otros proyectos y tareas prioritarias.

**Niveles de pruebas de integración**

Existen dos niveles de pruebas de integración: de componentes y de sistemas. En el primero se evalúan los elementos integrados en un mismo sistema, mientras que en el segundo se verifica la relación entre interfaces de sistemas externos.

Las**pruebas de integración de componentes** suelen ejecutarse primero que las **pruebas de integración de sistemas**, después de verificar las unidades de código de manera individual. Una vez asegurado este paso, es viable realizar los tests que involucran interfaces ajenas al sistema original.

Tipos de pruebas de integración de software

**Big Bang**

Una prueba de integración Big Bang concentra todos los módulos de un sistema para comprobar su funcionamiento en conjunto por lo que, antes de ejecutarse, el desarrollador debe cerciorarse que cada unidad ha sido completada.

Este tipo de test es viable en proyectos pequeños, de lo contrario, se pueden pasar por alto errores significativos.

**Ad Hoc**

Este término hace referencia al planteamiento de una solución para un problema específico. Para fines de [testing software](https://www.testingit.com.mx/blog/software-testing?hsLang=es-mx" \t "_blank) este tipo de prueba de integración puede ejecutarse en cualquier momento, recomendado ampliamente en etapas tempranas, con el objetivo de hallar errores no previstos.

Entre sus ventajas se encuentra la rapidez y poca planificación requerida para su realización. A pesar de ello, las dificultades pueden hallarse en fases posteriores ya que no se requiere documentación para dar cuenta de los errores encontrados.

**Top Down**

Como su nombre lo indica, las pruebas top down (de arriba hacia abajo) inician el análisis de código en los módulos posteriores, en los que se concentra la información central y de manera descendiente conectan con otras interfaces.

En este modelo pueden detectarse errores de mayor relevancia y por la complejidad de su composición las mejoras pueden implicar más tiempo.

**Down Top**

Contrario al modelo anterior, la prueba de integración down top parte de las interfaces inferiores y continúa de manera ascendente. En este caso los problemas son más fáciles de detectar, al igual que las mejoras a realizar.

Su desventaja es que los módulos complejos se sitúan al final de la prueba y la entrega del producto final puede tomar más tiempo del estimado.

**Hybrid**

Las**pruebas de integración híbridas**—también conocidas como sándwich— incluyen las prácticas de los dos modelos anteriores —top down y down top—. El desarrollador puede elegir los módulos posteriores o inferiores simultáneamente con el objetivo de hallar errores en menor tiempo.

Pese a ello, requiere de [equipo altamente capacitado](https://www.testingit.com.mx/conocenos?hsLang=es-mx) para la detección de errores de manera precisa y oportuna. Este tipo de prueba es recomendable en sistemas operativos de mayor complejidad.

**¿Qué es Figma?**

Figma es una herramienta para el diseño de interfaces que se dirige principalmente a los diseñadores web, UX y UI que necesitan un software específicamente diseñado para realizar el diseño de sitios web y aplicaciones.

En los últimos años hemos sido testigos de la aparición de numerosos programas capaces de imitar o implementar la funcionalidad de [Sketch](https://www.sketch.com/), el popularísimo y primer software de diseño de sitios web y aplicaciones disponible en iOS.

De las muchas aplicaciones que hemos visto nacer y luego morir, la solución que ofrece Figma es la que parece más convincente desde todos los puntos de vista: es completa, 100% accesible e incluso con mejores prestaciones que el antiguo Sketch.

**Ventajas que ofrece Figma**

* Accesibilidad multiplataforma.
* Sistema de colaboración en tiempo real.
* Uso total de herramientas orientadas a la respuesta para un diseño óptimo, orientado al desarrollo para la web.
* Trabaja en gráficos vectoriales.
* Todo esto es GRATIS.

**Accesibilidad multiplataforma**

Figma es, a todos los efectos, una web app, es decir, una aplicación que ofrece al usuario la posibilidad de trabajar simplemente a través del navegador. Esta es una de las ventajas más interesantes que podemos encontrar en Figma porque permite ampliar el campo de usuarios a todo aquel que tenga acceso a una red de Internet. Prácticamente todo el universo puede utilizarlo.

Desde este punto de vista, está claro que Figma se aleja al 100% de la filosofía impuesta por los desarrolladores de Sketch que, por el contrario, han creado un programa especialmente cerrado, restringido al mundo iOS.

Y si te interesa trabajar sin conexión, tampoco hay problema, porque Figma te proporciona dos aplicaciones que se pueden instalar en iOS y Windows respectivamente. Sin embargo, el hecho de poder utilizar una herramienta de este tipo a través del navegador, ahorrando así espacio en el disco, es una posibilidad muy interesante que merece la pena probar independientemente de tus preferencias.

Figma también ofrece **otras ventajas** nada desdeñables:

* Almacenamiento y organización de todos tus diseños en la nube.
* No es necesario guardar los diseños. Así que olvídate de perder tu trabajo cuando… “Photoshop CS4 ha dejado de funcionar”.
* Cada proyecto puede rastrearse a través de una URL muy trivial que podemos compartir con cualquiera.
* En definitiva, estamos ante un producto extremadamente versátil, completo y profesional que ofrece acceso a sus mejores prestaciones ya en el paquete gratuito.

**Las mejores características de Figma**

Todo lo que harías con cinco o seis herramientas diferentes, con Figma lo puedes hacer en la misma aplicación:

**1. Interfaz**

Dispone de una barra de herramientas muy práctica que proporciona todas las herramientas útiles para los diseñadores de contenidos web

**2. Función de creación de prototipos**

Adiós al «triste PDF enviado por correo electrónico». Bienvenida «simulación de app interactiva que puedo compartir con el cliente vía URL».

**3. Colaboración eficaz entre diseñadores y desarrolladores**

Si te encuentras en una agencia en la que los roles de diseñador y desarrollador son ocupados por dos profesionales diferentes, el desarrollador podrá encontrar toda la información en tiempo real sobre el proyecto Figma. Así, tendrá indicaciones de las reglas CSS, podrá calcular fácilmente la distancia del margen izquierdo al margen derecho, obtener iconos, imágenes, comprobar el comportamiento de los componentes al redimensionar la página, etc.

**4. Sistema de comentarios integrado**

Todo el mundo tiene acceso a dejar comentarios, mejorando así el flujo de trabajo colaborativo.

**5. Versionado**

Puedes simplemente navegar por el historial de cambios para restaurar los estados anteriores.

**6. Colaboración en tiempo real**

Figma te permite trabajar con tus compañeros en tiempo real y observar los movimientos del cursor en el proyecto para entender qué están haciendo exactamente.

**7. Desarrollo de componentes reutilizables**

Puedes, por ejemplo, crear tu propia cabecera y reutilizarla tantas veces como quieras en diferentes proyectos, ahorrando tiempo.

**8. Diseño responsive**

Gracias a la presencia de restricciones, tenemos la posibilidad de redimensionar las páginas en las que estamos trabajando de forma perfectamente adaptable.

Lo sensacional es que toda esta funcionalidad no va en detrimento del rendimiento, que sigue siendo igual, sino superior al de los programas comerciales más famosos.

**Bibliografía**

Azzalin, D. (2023, August 11). *Tipos de Pruebas de Software: Test Unitarios vs. Integration Test vs. Test end-to-end (e2e)*. Yeeply. <https://www.yeeply.com/blog/digitalizacion/tipos-pruebas-software-test-unitarios-vs-integration-test-vs-test-end-to-end-e2e/>

¿Qué es Figma? Ventajas y características. (2022, August 4). *Impulsoh.com*. <https://impulsoh.com/que-es-figma-y-para-que-sirve/>