

Módulo 2 / Encuentro 10/17

Casos de

prueba 1/2

## 

## 

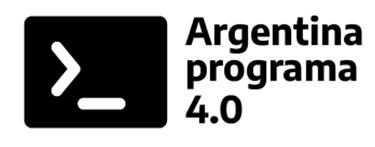
## OBJETIVOS DEL MÓDULO 2

¿Qué habilidades desarrollarás?

* Manejo y priorización de la información
* Introducción al manejo de proyectos
* Análisis y diseño de pruebas

¿Qué herramientas técnicas aprenderás?

* Procesos de Prueba
* Análisis de Requerimientos
* Matriz de Trazabilidad de Requerimientos
* Historias de usuario
* Documentación de pruebas
* Introducción a formularios HTML



|  |
| --- |
| **Aprender sobre casos de prueba es muy importante para tu desempeño en el futuro como Tester.** |

# **MATERIAL DE LECTURA**

## ¿Qué es un caso de prueba?

Los casos de prueba[[1]](#footnote-1) definen cómo probar un sistema, software o una aplicación. Un caso de prueba es un conjunto singular de acciones o instrucciones que debe realizar un tester que valida un aspecto específico de la funcionalidad de un producto o aplicación. Si la prueba falla, el resultado podría ser un defecto de software que la organización puede clasificar para arreglar.

Un tester o profesional de control de calidad generalmente escribe casos de prueba que se ejecutan después de completar una característica o el grupo de características que componen la versión. Los casos de prueba también confirman si el producto cumple con los requisitos de software.

Un grupo de casos de prueba se organiza en un conjunto de pruebas, que prueba un segmento lógico de la aplicación, como una característica específica.

|  |  |
| --- | --- |
|  | ¿Necesitas complementar la explicación anterior?  [**Mira un video**](https://youtu.be/Jegndzw3DEs) **sobre qué es un Caso de Prueba de Software.** Tiene una duración de sólo dos minutos. |

### Caso de prueba vs términos similares

Un caso de prueba es un concepto básico en las pruebas de software, pero existen términos similares que pueden causar confusión a los principiantes o a las personas menos familiarizadas con el control de calidad. Explicaremos qué es un caso de prueba, en relación con otros términos técnicos o con nombres similares.

### **Caso de prueba vs escenario de prueba**

Como su nombre lo indica, un escenario de prueba describe una situación o funcionalidad que requiere pruebas. Por ejemplo, un escenario de prueba podría ser: "Verificar la funcionalidad de inicio de sesión". Los escenarios de prueba suelen tener sus propios números de identificación para el seguimiento. Los equipos de control de calidad a menudo derivan casos de prueba (acciones de bajo nivel) a partir de escenarios de prueba (acciones de alto nivel); y los escenarios de prueba generalmente provienen de la documentación de requisitos comerciales y de software.

|  |  |
| --- | --- |
|  | ¿Quieres descubrir más sobre los escenarios de prueba? Mira el siguiente video. Tiene una duración de 6 minutos. [DESCUBRE Qué son los ESCENARIOS de PRUEBA🔍 (Son Mejores que los CASOS❓)](https://www.youtube.com/watch?v=UyaRph_jgyU) |

### **Caso de prueba vs script de prueba**

Estos términos son esencialmente intercambiables. Tanto un caso de prueba como un script de prueba describen una serie de acciones que prueban un elemento de la funcionalidad del software.

¡Advertencia! Un script de prueba se usa a menudo en el contexto de la automatización de pruebas, en el que una máquina realiza las pruebas. Por lo tanto, en un contexto de automatización, un desarrollador debe escribir un script de prueba para que sea legible por una máquina, mientras que un caso de prueba sería interpretado por un ser humano para la prueba manual.

### **Caso de prueba vs plan de prueba**

Un caso de prueba cubre una situación de prueba particular o una parte específica de la funcionalidad del producto. Un plan de prueba es un documento mucho más completo que cubre todos los aspectos de las pruebas de software inminentes. El propósito del plan de prueba es alinear las expectativas de toda la organización sobre lo que ocurrirá durante la prueba, incluido el alcance del proyecto, los objetivos, las fechas de inicio y finalización, las funciones y responsabilidades, los entregables y la mitigación de defectos.

### **Caso de prueba vs caso de uso**

Un caso de uso describe cómo un sistema realizará una tarea bajo ciertas condiciones. La documentación de requisitos comerciales o de software describe los casos de uso, que detallan cómo el usuario final interactuará con el sistema y el resultado que debe recibir. Los casos de uso describen cómo debería funcionar el producto, mientras que los casos de prueba describen cómo se debe probar el producto. Los casos de prueba se derivan de los casos de uso para garantizar que el producto se pruebe a fondo.

## ¿Por qué son importantes los casos de prueba?

Los casos de prueba definen lo que se debe hacer para probar un sistema, incluidos los pasos ejecutados en el sistema, los valores de datos de entrada que se ingresan en el sistema y los resultados que se esperan durante la ejecución del caso de prueba. El uso de casos de prueba permite a los desarrolladores y testers descubrir errores que pueden haber ocurrido durante el desarrollo o defectos que se pasaron por alto durante las pruebas ad hoc[[2]](#footnote-2).

Los beneficios de un caso de prueba efectivo incluyen:

* Buena cobertura de prueba garantizada.
* Costos reducidos de mantenimiento y soporte de software.
* Casos de prueba reutilizables.
* Confirmación de que el software satisfaga los requisitos del usuario final.
* Mejora de la calidad del software y la experiencia del usuario.
* Los productos de mayor calidad conducen a clientes más satisfechos.
* Más clientes satisfechos aumentarán las ganancias de la empresa.

En general, escribir y usar casos de prueba conducirá a la optimización del negocio. Los clientes están más satisfechos, aumenta la retención de clientes, disminuyen los costos de servicio al cliente y reparación de productos, y se producen productos más confiables, lo que mejora la reputación y la imagen de marca de la empresa.

## Tipos de casos de prueba

Para validar y verificar la funcionalidad del sistema, la organización debe adoptar un enfoque multifacético que evalúe la parte delantera y trasera del producto. Hay diferentes formas de categorizar los distintos tipos de casos de prueba.

**¡MANOS A LA OBRA! **

Pregúntate a tí mismo: ¿se me ocurre cuál o cómo podría ser la clasificación de los tipos de casos de prueba? Si de casualidad preguntas en tu equipo y alguien más se encuentra en este mismo punto de la guía, puedes aprovechar y preguntarle: ¿se te ocurre cuál o cómo podría ser la clasificación de los tipos de casos de prueba? Luego de haberte planteado la pregunta a ti mismo o habérsela hecho a un compañero de equipo, ¡puedes continuar con la lectura!

# **MATERIAL DE LECTURA**

### Casos de prueba formales

Con estos tipos de casos de prueba, el tester escribe una prueba en la que todas las entradas son conocidas y detalladas, como las condiciones previas y los datos de prueba. Las pruebas formales tienen una entrada predefinida, lo que significa que proporcionan una salida esperada, que la prueba intenta validar.

### Casos de prueba informales

Estos no tienen entradas o salidas conocidas. Los testers ejecutan este tipo de casos de prueba para descubrir y registrar los resultados, lo que puede revelar hallazgos interesantes sobre la calidad digital.

La mayoría de los tipos de casos de prueba son formales, planificados con anticipación de acuerdo con los requisitos del software. Exploremos algunos tipos y ejemplos de casos de prueba más:

Imagen 10.1: 8 tipos de casos de prueba. Fuente: elaboración propia.

### **Casos de prueba de funcionalidad**:

Estas pruebas determinan si la funcionalidad de destino tiene éxito o no al realizar su función dentro del sistema. El equipo de control de calidad escribe estos tipos de casos de prueba en función de los requisitos y los ejecuta cuando el equipo de desarrollo termina con la función. Muchos tipos diferentes de pruebas funcionales pueden validar la funcionalidad de la aplicación, incluidas las pruebas unitarias que verifican los segmentos de funcionalidad aislados más pequeños posibles. Los casos de prueba funcionales deben incluir:

* una descripción y/o nombre de la función bajo prueba
* condiciones previas
* pasos para la prueba
* un resultado esperado

**¿NECESITAS UN EJEMPLO? **

Ejemplo de caso de prueba de funcionalidad: realice un inicio de sesión exitoso y valide que el usuario haya iniciado sesión.

### **Casos de prueba de IU[[3]](#footnote-3)**:

Estas pruebas confirman que la interfaz de usuario (con lo que interactúa el usuario final) funciona como se esperaba. Por lo general, las pruebas de IU se enfocan en los elementos visuales de una aplicación o página web para confirmar que funcionan y se desempeñan de acuerdo con los requisitos. Las pruebas de IU a menudo examinan elementos de visualización como menús, submenús, botones, tablas y columnas para asegurarse de que sean legibles y consistentes.

Las interfaces de usuario siguen evolucionando. Por esta razón, las pruebas de IU también pueden significar validar una interfaz de voz o video. Las pruebas de IU también deben incluir problemas de accesibilidad, como si un lector de pantalla puede identificar un botón en una página.

**¿NECESITAS UN EJEMPLO?**

Ejemplo de caso de prueba de IU: navegue a la página de inicio, valide que el menú de hamburguesas se muestre correctamente para escritorio y web móvil.

### **Casos de prueba de rendimiento**:

Las pruebas funcionales comprueban si la aplicación funciona. Las pruebas no funcionales, como las pruebas de rendimiento, comprueban el rendimiento de la aplicación en diferentes tipos de cargas de trabajo. Una prueba de rendimiento debe ser específica con cada paso y resultado esperado documentados, así como datos de entrada claramente definidos, para que el tester pueda evaluar con precisión cómo funciona el sistema en las condiciones dadas.

Hay una variedad de tipos de pruebas de rendimiento, incluidas pruebas de carga, tensión, picos y escalabilidad. Cada tipo de prueba de rendimiento, y cada prueba individual, revela información diferente sobre cómo responde el sistema a las distintas cargas de usuarios.

**¿NECESITAS UN EJEMPLO?**

Ejemplo de caso de prueba de rendimiento: mida la mayor cantidad de usuarios que un sistema puede manejar antes de que se bloquee.

### **Casos de prueba de integración**:

Estos tipos de casos de prueba evalúan cómo funciona la funcionalidad combinada cuando se fusiona con la aplicación. Si bien es importante probar unidades individuales de software, es igualmente importante asegurarse de que los sistemas dispares puedan comunicarse entre sí de manera efectiva. El tester debe comprender bien los flujos de la aplicación para escribir pruebas de integración efectivas.

Las pruebas de API[[4]](#footnote-4) son un aspecto de las pruebas de integración. Las aplicaciones se comunican entre sí a través de las API, especialmente a medida que los productos se interconectan más en el mundo centrado en dispositivos móviles de hoy. Las pruebas de API son un ejercicio vital para cubrir con casos de prueba de integración.

**¿NECESITAS UN EJEMPLO?**

Ejemplo de caso de prueba de integración: inicie sesión a través del mercado de un vendedor, valide que el mercado reconozca al usuario como conectado; en otras palabras, los módulos de inicio de sesión y del mercado se comunican entre sí.

### **Casos de prueba de usabilidad**:

En lugar de probar la funcionalidad o el rendimiento de la aplicación, las pruebas de usabilidad examinan lo que los posibles usuarios finales, no los testers, piensan de un producto. Los investigadores de UX preparan pruebas para participantes fuera de la organización para medir qué tan fácil o difícil es usar el producto.

Las organizaciones pueden realizar pruebas de usabilidad de varias maneras, incluidas moderadas o no moderadas, remotas o en persona. El objetivo es aprovechar la perspectiva de un usuario final para identificar puntos en la aplicación que harían que dejara de usarla. Las pruebas de usabilidad pueden ser formales o informales, según el objetivo y el método UX de investigación.

**¿NECESITAS UN EJEMPLO?**

Ejemplo de caso de prueba de usabilidad: encargue al participante una transferencia de dinero entre sus cuentas corrientes y de ahorros, luego evalúe si puede completar con éxito la tarea y si experimenta alguna dificultad con el proceso.

### **Casos de prueba de bases de datos**:

El hecho de que la funcionalidad de una aplicación, la interfaz de usuario y las API funcionen no significa que los datos se almacenen correctamente. Las pruebas de la base de datos validan si los datos de la aplicación se almacenan de acuerdo con los requisitos y las reglamentaciones. Al igual que las pruebas de funcionalidad, las pruebas de bases de datos pueden variar en alcance, desde la validación de un pequeño objeto de base de datos hasta una acción compleja que involucra múltiples partes de la aplicación.

Algunos criterios que las pruebas de la base de datos pueden evaluar incluyen si los datos se almacenan de manera consistente, si personas no autorizadas pueden acceder a ellos y cómo se almacenan localmente en un dispositivo. Los datos consistentes y seguros deben ser una prioridad para todas las empresas, independientemente de los estándares de cumplimiento de la industria; las pruebas de bases de datos ayudan a lograrlo.

**¿NECESITAS UN EJEMPLO?**

Ejemplo de caso de prueba de base de datos: Valide que los nuevos datos de PII del cliente se almacenen en un formato cifrado.

### **Casos de prueba de seguridad**:

Estas pruebas identifican vulnerabilidades dentro de un sistema o producto. Otro tipo de prueba no funcional, las pruebas de seguridad tienen como objetivo encontrar formas de proteger mejor los activos de software, así como identificar cómo el sistema resiste los tipos comunes de ataques y definir el riesgo asociado con el producto.

Algunas pruebas de seguridad pueden incluir análisis de vulnerabilidades, análisis de configuración y pruebas de penetración, también llamadas pruebas intrusivas. En última instancia, el objetivo de las pruebas de seguridad es generar comentarios procesables que la organización pueda usar para remediar las vulnerabilidades.

**¿NECESITAS UN EJEMPLO?**

Ejemplo de caso de prueba de seguridad: valide que no puede acceder a los documentos de la empresa sin un inicio de sesión exitoso.

### **Casos de prueba de aceptación del usuario**:

Estos tipos de casos de prueba validan el producto desde la perspectiva del usuario final. Un usuario final o cliente realiza pruebas de aceptación del usuario en un entorno de prueba para validar el flujo de extremo a extremo del producto.

Las pruebas de aceptación del usuario pueden ser útiles cuando los requisitos comerciales cambian durante el curso del desarrollo. Las partes interesadas no siempre comunican de manera efectiva estos cambios al equipo de desarrollo. A través de los casos de prueba UAT, la organización puede documentar los criterios de entrada y salida que cubren las lagunas en las pruebas anteriores.

**¿NECESITAS UN EJEMPLO?**

Ejemplo de caso de prueba de aceptación del usuario: validar que un usuario puede registrarse para obtener una nueva cuenta y que recibe una confirmación por correo electrónico.

### **Casos de prueba exploratorios**:

Estos casos de prueba informales ocurren cuando el tester evalúa el sistema de forma ad-hoc para intentar descubrir defectos que no se detectaron en las pruebas estructuradas. Si bien las pruebas exploratorias no están definidas por un conjunto prescrito de acciones, el enfoque aún requiere cierta estructura, particularmente en lo que respecta a la documentación de resultados y el calendario, para garantizar una retroalimentación efectiva.

Las pruebas exploratorias pueden ayudar a validar los requisitos al verificar el sistema de maneras no cubiertas en las pruebas con guión. Las pruebas exploratorias permiten que la organización de control de calidad sea adaptable y aprenda de las lagunas en la cobertura de las pruebas.

**¿NECESITAS UN EJEMPLO? **

Ejemplo de caso de prueba exploratorio: verifique cómo el uso del botón Atrás del navegador afecta la funcionalidad de la aplicación y si requiere otro inicio de sesión.

Otras plataformas, como las plataformas de desarrollo de código bajo, también pueden tener sus propias pruebas específicas. Tenga en cuenta cómo se desarrollará el producto, así como cualquier detalle único que pueda requerir más pruebas.

|  |  |
| --- | --- |
|  | ¿Necesitas reforzar con un par de **ejemplos** para reforzar el concepto de **casos de prueba**? Mira el siguiente video. Tiene una duración de 4 minutos.[Caso de prueba, Hands-on QA, VD-5.](https://youtu.be/cxiGxr8ohJY) |

## Resultados del caso de prueba

Si bien los objetivos de los casos de prueba varían, la mayoría de los casos formales tienen resultados predecibles. De hecho, el formato de caso de prueba típico debe detallar el resultado esperado y el resultado real, que la prueba misma valida. La mayoría de los resultados de los casos de prueba entran en estas categorías:

* aprobado
* fallo
* sin ejecutar
* obstruido

Las pruebas de **aprobación** y **falla** indican que el sistema logra lo que se supone que debe hacer o falla en ese intento. Estos resultados no deben confundirse con las pruebas diseñadas para ser positivas o negativas, que pueden pasar o fallar. Las pruebas positivas aseguran que los usuarios puedan seguir todos los pasos y pasar el resultado esperado cuando la entrada es correcta, como una transferencia de dinero exitosa entre cuentas cuando hay un saldo superior a $0. Las pruebas negativas aseguran que el sistema maneja correctamente las entradas no válidas, como no permitir el inicio de sesión si la contraseña es incorrecta. Ambos tipos de pruebas pasan o fallan dependiendo del resultado esperado.

Los resultados de las pruebas que se marcan como **no ejecutados** son, como sugiere el nombre, pruebas que aún no se han ejecutado o que no se ejecutarán como parte de esta ronda de pruebas. Las pruebas bloqueadas resultan de una circunstancia externa o condición previa que inhibe la ejecución de la prueba. Por ejemplo, una falla del sistema que impide que la funcionalidad esté disponible provocará una prueba bloqueada, al igual que un entorno de prueba configurado incorrectamente.

### ¿Cuándo tenemos que empezar a construir casos de prueba?

**Lo antes posible**, empezando la planificación de un “testing básico” generando casos de prueba junto con la especificación de requerimientos. Esto se debe a que los casos de prueba NO deben estar influenciados por las estrategias de solución que hemos de elegir sino por la especificación del problema.

### ¿Qué casos debo testear?

Idealmente uno querría probar absolutamente todas las posibles combinaciones de datos y situaciones. Esto se hace normalmente imposible debido a la cantidad de casos que deberíamos generar. Lo que se hace comúnmente es reconocer las situaciones que creemos son más extremas y algunas de las más comunes. Este proceso debe ser planificado y documentado.

Cada caso de prueba cuesta dinero: esfuerzo para generarlo, tiempo para prepararlo y ejecutarlo, esfuerzo para evaluar los resultados y hacer seguimiento en el caso de hallar errores, teniendo que volver a probarlo y reportar. Por lo tanto, el número de casos de prueba necesarios para detectar los errores debe ser minimizado para reducir costos.

Perseguimos dos premisas al crear casos de prueba:

* Los casos de prueba deben ser finitos, cuantos menos y mejor diseñados, es mejor.
* Los casos de prueba buscan errores, queremos el menor número de casos de prueba que encuentre la cantidad máxima de errores

### Formato de caso de prueba

La documentación del caso de prueba generalmente incluye toda la información pertinente para ejecutar y recopilar datos de la prueba. Si bien el formato de caso de prueba específico puede diferir entre organizaciones, la mayoría incluye los siguientes detalles:

* Nombre del módulo. Este es el módulo o característica bajo prueba.
* Identificación y/o nombre de la prueba. Este es un identificador único que debe seguir una convención de nomenclatura estándar.
* Nombre del tester. La persona que realiza la prueba.
* Datos de prueba. Esto describe los conjuntos de datos que se utilizarán para la prueba.
* Supuestos o condiciones previas. Describa los diversos pasos que se deben realizar antes de la prueba, o lo que podemos suponer situacionalmente sobre la prueba, como "después de un inicio de sesión exitoso".
* Prioridad de prueba. Defina si la prueba es de prioridad baja, media o alta.
* Escenarios de prueba. Como se describió anteriormente, esta es la acción de alto nivel de la que se deriva el caso de prueba.
* Entorno de prueba. Identificar el nombre y/o las características del entorno de prueba.
* Pasos de prueba. Detalle los pasos que debe seguir el tester en el orden deseado.
* Resultados previstos. Esta es la salida que espera recibir del sistema.
* Resultados actuales. Esta es la salida que realmente recibe del sistema.
* Determinación de pasa/falla. Si los resultados reales coinciden con los resultados esperados, la prueba pasa. Si no, la prueba falla.

Al seguir el formato de caso de prueba anterior, la organización puede adherirse a una forma estándar de escribir pruebas, lo que resulta útil durante el mantenimiento. La organización debe revisar, mantener y aprobar regularmente los casos de prueba para garantizar que cubran adecuadamente la funcionalidad nueva y antigua. Los casos de prueba completamente detallados reducen la necesidad de pruebas exploratorias que requieren mucho tiempo para llenar los vacíos de cobertura.

### Modelo

|  |  |
| --- | --- |
| ID | El ID se le da al caso de prueba. |
| Descripción | La descripción del caso de prueba. |
| Requisito relacionado | El ID se proporciona al requisito al que se asigna este caso de prueba. |
| Requisitos previos | Cualquier condición previa o requisito que debe cumplirse antes de ejecutar la prueba. |
| Pasos de la prueba | Se dieron instrucciones paso a paso para ejecutar la prueba. |
| Datos de prueba | Datos que se utilizan mientras se realiza la prueba. |
| Resultado esperado | El resultado que se espera de la prueba, registrado antes de ejecutar la prueba. |
| Resultado real | El resultado real obtenido después de ejecutar la prueba |
| Estado | El estado obtenido después de ejecutar la prueba. Puede ser Pasa, Falla, No Ejecutado, Bloqueado. |
| Comentarios | Cualquier comentario que se haga para la prueba. |
| Información del entorno | Incluye información de red/hardware/software en la que se ejecuta la prueba. |

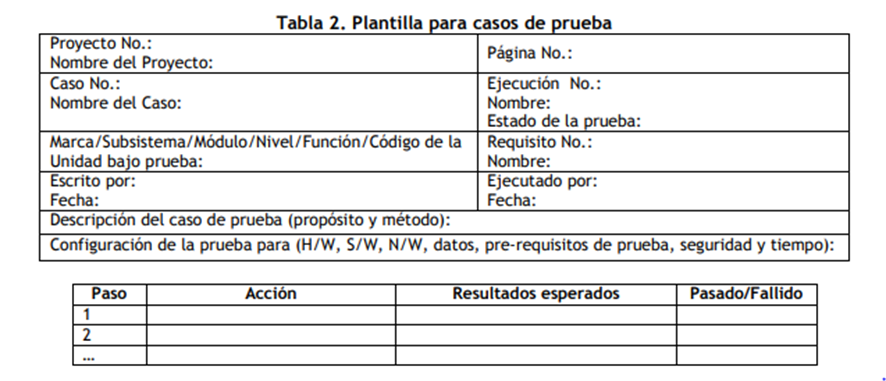
## Formato de elaboración de casos de prueba

Veremos 3 (tres) formatos para elaborar casos de prueba:

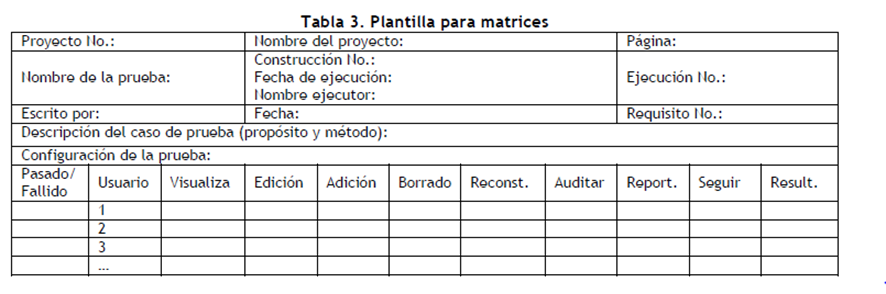
1. **Paso a paso.** Este formato se utiliza en:

* La mayoría de las reglas de procesamiento
* Casos de prueba únicos y diferentes
* Interfaces GUI
* Escenarios de movimiento en interfaces diferentes
* Entradas y salidas complicadas para representar en matrices.

**¿NECESITAS UN EJEMPLO? **

Ejemplo de una plantilla para casos de prueba por pasos:

1. **Matrices.** Sus usos más productivos son los siguientes:

* Formularios con información muy variada, mismos campos, valores y archivos de entrada diferentes
* Mismas entradas, diferentes plataformas, navegadores y configuraciones
* Pantallas basadas en caracteres
* Entradas y salidas con mejor representación matricial.

1. **Scripts automatizados.** La decisión de utilizar software para automatizar las pruebas depende de la organización y del proyecto que se esté probando. Existen algunas cuestiones técnicas que deben concretarse y que varían de una herramienta a otra. El beneficio real de automatizar las pruebas se obtiene en la fase de mantenimiento del ciclo de vida del software.

Un caso de prueba **paso a paso** tiende a ser más verbal, y el de **matrices** más numérico. A menudo, la ruta más productiva es utilizarlos todos.

## 

## Escribir casos de prueba de manera eficiente

Los casos de prueba bien escritos tienen beneficios obvios: productos de mejor calidad, clientes más felices, mayores ganancias y un mantenimiento de prueba más fácil. Pero se requiere de esfuerzo y organización para redactar casos de prueba que cooperen para lograr estos objetivos.

En general, el tester debe escribir casos de prueba al principio del SDLC, como durante la fase de recopilación de requisitos. Los testers deben consultar los requisitos y la documentación de casos de uso, así como el plan de prueba general cuando escriben casos de prueba. Un prototipo también puede informar al tester sobre cómo se verá la característica o funcionalidad cuando se complete.

Una vez que el tester tiene toda esta información, puede comenzar a escribir los diversos tipos de casos de prueba mencionados anteriormente. Al escribir casos de prueba, el tester debe considerar los flujos de la aplicación: la forma en que el usuario llega a la funcionalidad de la aplicación es un elemento importante de su viaje y debe validarse adecuadamente. Por ejemplo, los cambios en la configuración de la cuenta deben funcionar correctamente en una aplicación móvil, que puede ser el flujo principal, pero también deben funcionar en un navegador web, así como en cualquier otro lugar donde los usuarios puedan interactuar o cambiar la configuración.

|  |
| --- |
| **¡Pro tip alert!** Escribe casos de prueba de forma clara y concisa para garantizar la precisión sin importar quién lea y ejecute la prueba. Si bien algunos detalles son importantes, trata de mantener los casos de prueba económicos y fáciles de ejecutar a un alto nivel para reducir el mantenimiento cuando cambia la aplicación.  Los casos de prueba bien escritos también deben ser repetibles y reutilizables. Pocas pruebas se ejecutan sólo una vez, y las pruebas reutilizables pueden ahorrar tiempo al desarrollar funciones adicionales. Haz que cada uno sea rastreable, de modo que la documentación y los resultados puedan informar fácilmente al equipo. |

Un diseño de caso de prueba efectivo será:

* Exacto o específico sobre el propósito.
* Económico: no se utilizan pasos ni palabras innecesarias.
* Rastreable: los requisitos se pueden rastrear.
* Repetible: el documento se puede utilizar para realizar la prueba varias veces.
* Reutilizable: el documento se puede reutilizar para volver a realizar la prueba con éxito en el futuro.

Para lograr estos objetivos, los ingenieros de pruebas y control de calidad pueden utilizar las siguientes prácticas recomendadas:

1. Priorizar qué casos de prueba escribir en función de los plazos del proyecto y los factores de riesgo del sistema o la aplicación.
2. Crear casos de prueba únicos y evitar casos de prueba irrelevantes o duplicados.
3. Confirmar que el conjunto de pruebas verifica todos los requisitos especificados mencionados en el documento de especificación.
4. Escribir casos de prueba que sean transparentes y directos. El título de cada caso de prueba debe ser corto.
5. Los pasos del caso de prueba deben dividirse en los segmentos más pequeños posibles para evitar confusiones durante la ejecución.
6. Los casos de prueba deben escribirse de manera que permitan que otros los entiendan fácilmente y modifiquen el documento cuando sea necesario.
7. Ten en cuenta al usuario final siempre que se cree un caso de prueba.
8. No asumir las características y la funcionalidad del sistema.
9. Cada caso de prueba debe ser fácilmente identificable.
10. Las descripciones deben ser claras y concisas.

## Buenas prácticas para elaborar casos de prueba

|  |
| --- |
| La mejor práctica es desarrollar por lo menos dos casos de prueba para cada requerimiento a probar. A medida que se va conociendo el negocio y teniendo más experiencia en el testing, se va estimando mejor la cantidad de casos de prueba necesarios. Por ejemplo, para testear correcta y completamente un Login, requiere de 7 casos de prueba. |
| Los casos de prueba deben ser **simples**: se deben crear casos de prueba que sean lo más simple posible ya que otra persona que no sea el autor puede ejecutarlos.​ Usa un lenguaje asertivo como *“ir a la página de inicio”*, *“ingresar datos”*, *“hacer clic en esto”*, etc. Esto facilita la comprensión de los pasos de prueba y hace que la ejecución sea más rápida. |
| El **título** debe ser **fuerte**: la manera correcta de comenzar con el título de un caso de prueba es con un verbo en infinitivo. Los verbos en infinitivo denotan un mandato. |
| Tomar al **usuario final** en cuenta: el objetivo final es crear casos de prueba que cumplan con los requisitos del cliente y que sean fáciles de usar. Un tester debe crear casos de prueba tomando en cuenta la perspectiva del usuario final. |
| **Evitar la repetición** de casos de prueba​: si se necesita un caso de prueba para ejecutar algún otro, llama al caso de prueba por su id. Inclúyelo en la columna de pre-condiciones o donde corresponda dependiendo la herramienta. |
| **No asumir**: No asumir la funcionalidad y las características de la aplicación mientras preparas el caso de prueba. Apégate a los documentos de especificación y si tienes alguna duda, pregunta. |
| Asegurar la **mayor cobertura** posible: Asegúrate de escribir casos de prueba para todos los requisitos especificados. ​Algo que se puede utilizar es una matriz de trazabilidad para garantizar que se prueben todos los casos de pruebas asociados. La mayoría de las herramientas permiten vincular los casos de prueba entre sí. ​ |
| **Autonomía**: el caso de prueba debe generar los mismos resultados cada vez, sin importar quién lo pruebe. |
| Implementar **técnicas** de prueba: no es posible verificar todas las condiciones posibles de una aplicación, pero las técnicas de prueba ayudan a seleccionar los casos de prueba con la máxima posibilidad de encontrar un defecto.​ |
| Mantener un **repositorio estándar** de casos de prueba reutilizables para su aplicación garantizará que la mayoría de los errores comunes se detecten más rápidamente. La reutilización de los casos de prueba ayuda a ahorrar dinero en recursos para escribir pruebas repetitivas. Siempre se cubrirán casos de prueba importantes, por lo que será casi imposible olvidarlos. |
| Una **lista de verificación** ayuda a completar la redacción de casos de prueba rápidamente para nuevas versiones de la aplicación. Los casos de prueba se pueden consultar en la lista de verificación de pruebas para asegurarse de que los problemas más comunes se solucionen en la fase de desarrollo. |
| Definir la **prioridad** de una prueba. Generalmente es mejor utilizar cualquiera de los 3 niveles, alto, medio y bajo, o 1, 50 y 100. Por lo tanto, cuando tengas un cronograma, primero debes completar todas las pruebas de alta prioridad y luego pasar a las pruebas de prioridad media y baja. Para identificar estos niveles se debe conocer bien el negocio y la importancia de los niveles de error para el usuario. |

**¿NECESITAS UN EJEMPLO? **

**Ejemplo:** Si se debe testear un sistema de venta de publicaciones científicas, para el cliente pueden ser de alta prioridad: “verificar que los títulos y autores de las publicaciones estén bien escritos”, así como “verificar que se dispongan de una lista de valores o filtros para clasificar una publicación” pueden ser de alta prioridad=1. En otros tipos de sistemas, para el usuario estas pruebas quizás no serían tan relevantes.

**Ejemplo de prioridades:** “un sitio web de compras”:

* Verificar “la denegación de acceso por un intento no válido de iniciar sesión” en la aplicación puede ser un caso de alta prioridad=1
* Verificar “la visualización de productos relevantes en la pantalla del usuario” o “que se encuentren disponibles todos los productos en un catálogo”, pueden ser casos de prioridad media=2
* Verificar “el color del texto que se muestra en los botones de la pantalla” o que “una etiqueta o label esté bien escrita”, pueden ser pruebas de baja prioridad=3.

**MATERIAL DE LECTURA **

La forma de escribir pruebas y casos de prueba eficaces se puede optimizar con el tiempo. Algunas de las mejores prácticas incluyen el uso de títulos sólidos, descripciones sólidas y mantener el lenguaje conciso y claro. Pero también querrá incluir condiciones previas, suposiciones y los resultados esperados.

Toda esta información es relevante para el tester de software, especialmente cuando se determina si el caso de prueba es un "pasó" o un "error" en su lugar.

Como sabes, si se trata de una aplicación web, los resultados de la prueba pueden diferir según el navegador en el que se ejecute la prueba. Para la facilidad de otros evaluadores, desarrolladores o quien esté revisando el documento de prueba, debe agregar el nombre y la versión del navegador al caso para que el defecto se pueda replicar fácilmente.

Mantén las cosas simples y transparentes, pero no demasiado simples, hazlo complejo, pero no demasiado complejo: mantén todos los pasos de los casos de prueba atómicos y precisos. El caso de prueba debe ser auto explicativo y fácil de entender.

La revisión por pares es importante, revisando los más seniors las pruebas de otros testers más juniors para mejorarlos. Este no solo es un momento de verificación y validación, sino también de aprendizaje continuo. Es importante que quien revise tenga conocimientos de testing y del negocio. Es una práctica muy importante y que se aplica muy seguido.

Especifica los resultados esperados y los supuestos antes de ejecutar los casos de pruebas. Recuerda que muchas veces se supone que hay, por ejemplo, datos cargados para poder correr un caso de prueba, y muchas veces no están cargados o fueron pisados por otra versión de la base de datos o porque alguien los utilizó para probar.

## Factores de calidad de los casos de prueba

Un caso de prueba debe cumplir con los siguientes factores de calidad:

1. **Correcto**. Ser apropiado para los probadores y el entorno. Si teóricamente es razonable, pero exige algo que ninguno de los probadores tiene, se caerá por su propio peso.
2. **Exacto**. Demostrar que su descripción se puede probar.
3. **Económico**. Tener sólo los pasos o los campos necesarios para su propósito.
4. **Confiable** y **repetible**. Ser un experimento controlado con el que se obtiene el mismo resultado cada vez que se ejecuta, sin importar qué se pruebe.
5. **Rastreable**. Saber qué requisitos del caso de uso se prueban.
6. **Medible**. Este es un ejercicio muy útil para quienes escriben pruebas, para verificar constantemente dónde están, si pierden alguno de los elementos, o si no se cumple un estándar.

## 

## Gestión de casos de prueba

Una forma de asegurarse de que los casos de prueba sean fáciles de localizar y comprender es revisarlos minuciosamente. Los casos de prueba requieren coherencia en las convenciones de nomenclatura y las descripciones. Una verificación de cordura también puede revelar si la descripción "simple" del escritor de los pasos de la prueba realmente tiene sentido para otro lector y si refleja las condiciones del mundo real.

A medida que aumenta el alcance de un producto, también lo hace la huella de sus casos de prueba. En pocas palabras, cuanto más desarrolle, más necesitará probar, lo que puede generar desafíos cuando se trata de escalar conjuntos de pruebas. Los casos de prueba no solo tienen que mantenerse al día con la nueva funcionalidad, sino que la necesidad de realizar pruebas de regresión significa que los casos de prueba más antiguos también necesitan actualizaciones.

Las herramientas o productos de administración de pruebas pueden ayudar a las organizaciones a rastrear y actualizar las pruebas según sea necesario. Hay muchas opciones para las herramientas de gestión de pruebas. En última instancia, la mejor opción es la que se adapta de la mejor manera posible a sus flujos de trabajo, lo que permite al equipo ver, comentar y acceder a los registros de auditoría.

La generación de informes es otro elemento importante de la gestión de casos de prueba. Los informes de casos de prueba deben brindarle al equipo información procesable sobre cómo se están realizando las pruebas, qué cobertura tiene y dónde puede mejorar el equipo en el futuro.

Si bien puede ser desalentador administrar conjuntos de pruebas, en última instancia, es una tarea necesaria para mantener la calidad digital de sus productos. Si la tarea es difícil de mantener internamente, busque herramientas o servicios que le ayuden a mantenerse al día.

## ¿Cómo convertir historias de usuarios en casos de prueba?

La siguiente es una plantilla de muestra que incluye una historia de usuario, una prueba de aceptación y un caso de prueba:

Historia del usuario:

Como [función de usuario],

Necesito [la capacidad de hacer algo],

para que pueda [obtener algún beneficio o evitar alguna consecuencia]

Criterios de aceptación/Prueba:

Dado [entrada | condiciones previas],

cuando [acciones | disparadores],

entonces [salida | Consecuencias].

Caso de prueba:

ID de TC [id/número único]

Resumen [objetivo]

Requisitos previos [condiciones previas que deben cumplirse para llevar a cabo más pasos]

Procedimiento de prueba [procedimiento paso a paso para la ejecución]

## Resultados esperados y reales

Una historia de usuario promueve más discusiones y colaboración dentro del equipo. Los criterios de aceptación definen límites y requisitos. Los siguientes son los pasos clave involucrados en la creación de criterios de prueba en un modelo Agile:

1. La empresa proporciona requisitos y criterios de aceptación para una historia de usuario.
2. El control de calidad revisa las consultas y extrae los casos de prueba existentes. Se crean nuevos casos de prueba basados ​​en las áreas funcionales.
3. Los ingenieros controlan los escenarios que afectan el flujo de trabajo comercial en caso de que no se capturen en la historia del usuario.
4. QA proporciona una sesión a los analistas de negocios para implementar cambios en las primeras etapas.
5. Mientras redactan los casos de prueba, los equipos de control de calidad visualizan el flujo de trabajo para que toda la cobertura del camino feliz o escenario positivo, el análisis de límites y los escenarios negativos puedan cubrirse de una manera bien estipulada.

* Escenario positivo: el escenario positivo (Happy Path) prueba los requisitos básicos de una aplicación. Si el resultado de Happy Path es un fracaso, entonces el resto de las pruebas pueden bloquearse debido a defectos críticos.
* Escenarios negativos: prueba los escenarios negativos para ver cómo se comporta el sistema en caso de entradas inapropiadas.
* Escenario de regresión: verifica algunos elementos estándar, como encabezados, pies de página, opciones de menú estándar, hipervínculo y escenarios interrelacionados. Estos escenarios de prueba lo ayudarán a realizar pruebas de regresión.
* Análisis de límites: prueba las condiciones de límites para verificar que el sistema se comporte correctamente en los límites mínimo y máximo sin ningún comportamiento abrupto.
* Escenario de integración: realizar pruebas de integración es necesario cuando un proyecto tiene varios módulos pero los cambios se realizan en un solo módulo, verifica si la funcionalidad está funcionando entre todos los módulos colectivamente.

Analogía entre una historia de usuario y su caso de prueba:

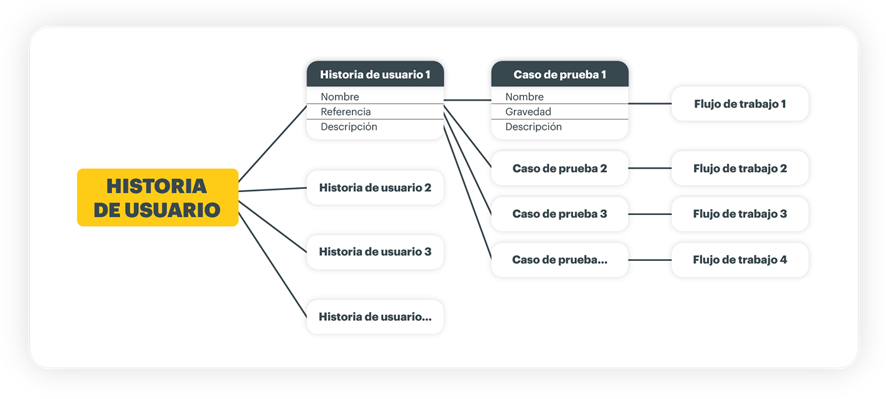


Imagen 10.2: Analogía entre una historia de usuario y su caso de prueba. Fuente: elaboración propia.

# 

# Escenarios de análisis

A continuación, te presentaremos varios escenarios de análisis en donde veremos aplicados algunos de los conceptos que aprendimos hoy. Te invitamos a que tomes algunos minutos de reflexión con tu equipo y discutan cada escenario y su respuesta. Cada uno deberá exponer las razones por las que elige cada opción. Podrán coincidir o no entre ustedes, no se preocupen, intenten llegar a una opinión unánime, pero de no hacerlo cada uno enviará sus propias respuestas.

En el [formulario de fin de guía](https://forms.gle/sYe2dFwXa5itc3eS6) encontrarás estos mismos escenarios y sus respuestas, envíala de forma individual luego de haberlo discutido.

1. John está redactando casos de prueba. Se le ocurren varias ideas para encontrar todos los errores posibles, pero duda al ver la cantidad de casos que acumula. ¿Qué consejo le darías?
   1. Que se mantenga trabajando así, puede crear cuantos casos quiera. Mientras más errores encuentre mejor.
   2. Que diseñe casos de prueba acorde al presupuesto que posea
   3. Que procure crear casos de prueba que sean más abarcativos y prueben varias funcionalidades de una sola vez.
   4. Que redacte todos los casos de prueba que se le ocurran y luego piense cuales usa
2. A Carlos le han encargado el diseño de los casos de prueba, para comenzar está esperando que le envíen los requisitos y el plan de pruebas. Esto es:
   1. Correcto, no puede saber qué debe hacer sin esa información.
   2. Correcto, pero sólo debe esperar los requisitos.
   3. Incorrecto, él mismo debe hacer el plan de pruebas luego de los casos de prueba.
   4. Incorrecto. Sólo con el prototipo ya puede empezar a realizar los casos de prueba.
3. Javier está en una encrucijada. Ha creado casos de prueba muy buenos, pero se dio cuenta que los mismos sólo servirán la primera vez. Esto es:
   1. Correcto, el caso de prueba sólo debe usarse una vez.
   2. Correcto, pero el diseño sería más efectivo si fuera repetible y reutilizable.
   3. Incorrecto, los casos de prueba deben poder ejecutarse en repetidas oportunidades.
4. Galia está ejecutando casos de prueba. Los está ejecutando en el orden que piensa encontrará más errores. Esto es:
   1. Correcto, al diseñar casos de prueba no los ordenamos
   2. Correcto, es preferible encontrar más errores primero para ponerse a trabajar en soluciones.
   3. Incorrecto, es indistinto el orden de ejecución.
   4. Incorrecto, al ejecutar los casos de prueba debe tenerse en cuenta la prioridad.
5. Valentino está redactando casos de prueba. Al redactar algunas pruebas de usabilidad, se da cuenta que su tiempo se agota por lo que, para ejecutarlas, las realiza él mismo. Esto es:
   1. Correcto, él podrá detectar todos los errores.
   2. Correcto, él está capacitado para ejecutar las pruebas, pero sería más conveniente si las ejecutara un usuario final para encontrar más errores.
   3. Incorrecto, las pruebas de usabilidad deben ser ejecutadas por usuarios finales
   4. Incorrecto, él sólo escribe los casos de prueba, no los ejecuta

Antes de ponernos manos a la obra, mira este próximo **video**

|  |  |
| --- | --- |
|  | [**Haz click aquí**](https://youtu.be/_UwAwLp7UyU). Este video tiene una duración de 7 minutos. |

# Ejercicios

### Ejercicio #1

Completa el siguiente cuadro con los conceptos a los que se relaciona. Puedes utilizar este [cuadro como plantilla](https://docs.google.com/document/d/14tFenkKHu-W2pAkHkX695d-jXbRWcAhsTAwRoLHecOE/edit?usp=sharing).

|  |  |
| --- | --- |
| **Frases** | **Concepto/s con los que se relaciona la frase** |
| Estos tipos de casos de prueba evalúan cómo funciona la funcionalidad combinada cuando se fusiona con la aplicación |  |
| Examinan a los posibles usuarios finales |  |
| Comprueba el rendimiento de la aplicación en diferentes tipos de cargas de trabajo |  |
| Evalúa el sistema de forma ad-hoc para intentar descubrir defectos que no se detectaron en las pruebas estructuradas |  |
| Identifican vulnerabilidades dentro de un sistema o producto |  |
| Validan el producto desde la perspectiva del usuario final |  |
| Estas pruebas confirman que la interfaz de usuario (con lo que interactúa el usuario final) funciona como se esperaba |  |
| Determinan si la funcionalidad de destino tiene éxito o no al realizar su función dentro del sistema |  |
| Validan si los datos de la aplicación se almacenan de acuerdo con los requisitos y las reglamentaciones |  |

### 

### Ejercicio #2

1. Realizar los casos de prueba de las siguientes historias de usuario:
   1. Como usuario de android, quiero que al establecer una alarma se muestre el tiempo que falta para que se active, para saber cuánto tiempo voy a dormir.
   2. Como gestor, quiero poder comprender el progreso de mis compañeros, para poder informar sobre nuestros éxitos y fallos.
   3. Como usuario me gustaría poder buscar transacciones y así poder detectar gastos que sean innecesarios en mi cuenta en algún período de tiempo.
2. Realizar los casos de prueba usando las historias de usuario del ejercicio de la guía anterior (Caso 1, 2, 3 y 4). ¿No recuerdas de qué casos hablamos? No te preocupes, te los recordamos en la plantilla para responder al ejercicio que te dejamos a continuación.
3. Tómate unos minutos para comparar tus respuestas con las de tu equipo.

Un ejemplo de cómo podríamos resolverlo:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id Caso de prueba | Caso de Prueba  (Objetivo) | Precondiciones (Requisitos previos) | Procedimiento (Pasos para el caso de prueba) | Datos de prueba | Resultados Esperados | Resultado obtenido |
| CP 1 | Verificar que se pueda ver la lista de pacien | El operador debe estar logueado dentro del sistema. | 1- En la vista principal, presionar el botón “Turnos”  2- Seleccionar fecha en el calendario  3- Apretar el botón “Confirmar” |  | Ver la lista de pacientes con turnos para la fecha indicada. | Ver la lista de pacientes con turnos para la fecha indicada |

|  |
| --- |
| Ingresa a [este documento](https://docs.google.com/document/d/1rNF_2qYijlKYN9-77oIqCzLP2lsCYAjX7akHlwRPhR0/edit?usp=sharing) para descargar una plantilla y poder realizar los puntos (1) y (2) de este ejercicio. |

**¡Hora de cerrar! **

Ya hemos llegado al final de esta guía. En el Aula Virtual, cuentas con el Check de conocimiento para validar cómo vienes avanzando en tu aprendizaje. De igual manera, aquí te dejamos el [link](https://forms.gle/sYe2dFwXa5itc3eS6) para que puedas completarlo.

1. Test cases en inglés :-) [↑](#footnote-ref-1)
2. Ad hoc es un término del latín que significa literalmente 'para esto'. Se emplea como locución adjetiva con el sentido de 'adecuado, apropiado, dispuesto especialmente para un fin'; y como locución adverbial, con el sentido de 'a propósito para la ocasión'. [↑](#footnote-ref-2)
3. ¿Qué significa IU en diseño? El diseño UI se refiere al término «diseño de interfaz de usuario». [↑](#footnote-ref-3)
4. API significa “interfaz de programación de aplicaciones”. En el contexto de las API, la palabra aplicación se refiere a cualquier software con una función distinta. La interfaz puede considerarse como un contrato de servicio entre dos aplicaciones. Este contrato define cómo se comunican entre sí mediante solicitudes y respuestas. La documentación de su API contiene información sobre cómo los desarrolladores deben estructurar esas solicitudes y respuestas. Fuente: <https://aws.amazon.com/es/what-is/api/> [↑](#footnote-ref-4)