Cómo escribir casos de prueba efectivos en 5 pasos

(y evitar errores comunes)

www.linkedin.com/in/sandravals

¿Por qué es importante escribir buenos casos de prueba?

Casos de prueba bien escritos aseguran que el software sea probado de manera eficiente, reducen errores y mejoran la calidad final del producto

Comparación rápida entre un caso de prueba mal escrito y uno bien escrito

Verificar que un usuario con credenciales correctas pueda iniciar sesión exitosamente en la plataforma

Probar login

Paso 1 – Define claramente el objetivo del test

Antes de escribir el caso de prueba, pregúntate:
¿Qué funcionalidad quiero verificar?
¿Cuál es el resultado esperado?

Ejemplo:

✅ Verificar que un usuario puede restablecer su contraseña correctamente

❌ Probar recuperación de contraseña (Demasiado genérico)

Paso 2 – Escribe pasos claros y detallados

Cada paso debe ser específico para evitar confusiones. No asumas que el tester sabe lo que tiene que hacer

Ejemplo de estructura ideal:

1. Abre la aplicación web en Chrome
2. Introduce usuario y contraseña válidos
3. Presiona el botón “Iniciar sesión”
4. Verifica que el usuario sea redirigido al dashboard

Error común:
Abrir la app e iniciar sesión
(Demasiado genérico)

Paso 3 – Especifica los datos de prueba

Si tu caso de prueba necesita datos específicos, inclúyelos en el test para evitar dudas

Ejemplo correcto:

Usuario: usuario@test.com
Contraseña: Password123!

Error común:
Introducir credenciales
(No se especifican datos)

Paso 4 – Incluye el resultado esperado

Cada caso de prueba debe tener un resultado esperado para que sea fácil saber si pasó o falló

Ejemplo correcto:

El usuario es redirigido al dashboard y se muestra su nombre en la esquina superior derecha

Error común:
Funciona bien
(No es específico)

Paso 5 – Prioriza los casos de prueba según su importancia

No todos los test son igual de importantes
Usa la técnica de **Riesgo + Impacto** para priorizar:

Alta prioridad
Funcionalidades esenciales o de alto riesgo
Ejemplo: Validar pagos y accesos de usuario

Media prioridad
Características secundarias o integraciones
Ejemplo: Editar perfil o cambiar contraseña

Baja prioridad
Detalles visuales o mejoras sin impacto funcional
Ejemplo: Verificar colores de botones

Test bien priorizados = Eficiencia en el testing