REALIZACIÓN DE PRUEBAS DE FUNCIONALIDAD DEL SOFTWARE

GA10220501097–AA7–EV 01

Sebastian Fajardo Martinez

Instructor: Andrés Parra

ANALISIS Y DESARROLLO DE SOFTWARE

SENA

Ficha 2758289

Febrero 2025

TABLA DE CONTENIDO

INTRODUCCION

Durante el proceso de desarrollo de software, resulta imprescindible verificar que el sistema se comporte tal como fue definido en los requisitos funcionales. Las pruebas de funcionalidad tienen como objetivo validar que cada componente del software responda correctamente a las acciones del usuario, asegurando que las entradas generen los resultados previstos.

Este tipo de pruebas se enfoca en simular el uso real de la aplicación, tal como lo haría un usuario final. De esta manera, es posible identificar fallos o inconsistencias que podrían afectar la experiencia general, anticiparse a errores críticos y garantizar un producto confiable.

Al aplicar pruebas funcionales desde etapas tempranas y a lo largo del ciclo de vida del software, se fortalece la calidad del sistema, se mejora su estabilidad y se asegura que cumpla con las necesidades del cliente o usuario final.

¿CÓMO SE DEBERÍA REALIZAR LAS PRUEBAS DE FUNCIONALIDAD DE UN SITIO PUBLICADO EN INTERNET?

Las pruebas funcionales en sitios web accesibles públicamente son esenciales para confirmar que la plataforma se comporta como se espera ante diversas interacciones. Estas pruebas permiten detectar errores, validar flujos críticos y asegurar una experiencia fluida para el usuario. A continuación, se describen los pasos fundamentales para llevarlas a cabo de forma eficiente:

1. Definir el alcance de las pruebas

Antes de iniciar, es clave delimitar qué funcionalidades serán evaluadas. Esto incluye formularios, navegación entre páginas, procesos de autenticación, carritos de compra, entre otros. Se deben priorizar aquellas áreas que resultan más sensibles o que representan puntos críticos en el flujo del sistema.

2. Preparar los casos de prueba

Cada caso de prueba debe describir claramente los pasos a seguir, los datos necesarios y el resultado esperado. Es recomendable cubrir tanto escenarios comunes como situaciones límite o errores, utilizando datos reales o simulados que se ajusten al comportamiento del sistema.

3. Realizar pruebas manuales

La evaluación manual permite comprobar visualmente que los elementos del sitio (enlaces, botones, formularios) funcionan adecuadamente. Además, se analiza la consistencia del contenido, la correcta presentación de imágenes y textos, y la navegación general del sitio en diversos navegadores y dispositivos.

4. Realizar pruebas automatizadas

Cuando se trata de validar procesos que se repiten constantemente, lo ideal es utilizar herramientas de automatización como Selenium, Cypress o Playwright. Estas herramientas permiten programar secuencias de interacción que simulan el comportamiento del usuario, facilitando la ejecución de pruebas regresivas.

5. Pruebas de usabilidad

La usabilidad es un aspecto fundamental. Es importante identificar si la interfaz es intuitiva, si la navegación es lógica y si los tiempos de respuesta generan una experiencia agradable. Se recomienda realizar pruebas con usuarios reales y verificar el cumplimiento de pautas de accesibilidad (como las WCAG).

6. Pruebas de rendimiento

El tiempo de carga y la capacidad de respuesta del sitio deben evaluarse con herramientas como JMeter, Lighthouse o GTmetrix. Estas pruebas permiten simular la actividad de múltiples usuarios y ayudan a identificar cuellos de botella que puedan surgir bajo alta demanda.

7. Pruebas de seguridad

La protección de los datos y la autenticación de usuarios son áreas clave. Se deben realizar pruebas para comprobar que los accesos están correctamente restringidos, que la información confidencial está cifrada y que el sistema no es vulnerable a ataques comunes como inyección SQL o cross-site scripting (XSS).

8. Pruebas en diferentes entornos

El comportamiento del sitio puede variar dependiendo del entorno en el que se despliegue (desarrollo, pruebas, producción). Por ello, es esencial comprobar su funcionamiento bajo configuraciones distintas de servidor, base de datos o servicios externos.

9. Documentar y reportar errores

Todo error detectado debe documentarse detalladamente, incluyendo cómo reproducirlo, su impacto y evidencia visual si es posible. Esto facilita su análisis, priorización y resolución, especialmente si se utilizan herramientas como Jira, Trello o GitLab Issues.

10. Validar correcciones y realizar pruebas de regresión

Después de corregir errores, se deben ejecutar nuevamente las pruebas para confirmar que el problema se ha solucionado y que no se han generado nuevas fallas en otras funcionalidades.

11. Pruebas continuas

En un entorno de desarrollo activo, donde se realizan cambios constantes, es recomendable incorporar las pruebas funcionales en pipelines de integración continua. Así, cada nueva versión del sistema es evaluada automáticamente para asegurar su estabilidad.

Herramientas recomendadas

- **Pruebas manuales**: Navegadores web, herramientas de desarrollo (DevTools).
- **Automatización**: Selenium, Cypress, Jest.
- **Rendimiento**: JMeter, LoadRunner, Google PageSpeed Insights.
- **Seguridad**: OWASP ZAP, Burp Suite.
- **Monitoreo**: New Relic, Datadog, Google Analytics.

CÓMO VERIFICAR SI UN DOMINIO ESTÁ BIEN CONFIGURADO

Antes de que un sitio web esté disponible de forma estable y segura en internet, es fundamental validar que el dominio asociado esté configurado correctamente. Un dominio mal configurado puede generar errores de acceso, problemas de seguridad o fallos en el envío de correos. A continuación, se presentan los aspectos clave a revisar:

Verificar la configuración de DNS

El primer paso es asegurarse de que los registros DNS del dominio están configurados adecuadamente. Esto incluye verificar:

- Registro A: apunte correcto a la dirección IP del servidor.
- CNAME: si existen alias configurados.
- MX: configuración del correo electrónico.
- TXT: validaciones como SPF o Google Site Verification.
- NS: servidores de nombres autorizados.

Comprobar la propagación del dominio

Cuando se hacen cambios en los registros, estos pueden tardar en reflejarse globalmente. Plataformas como WhatsMyDNS permiten verificar si la propagación de DNS se ha completado correctamente en distintos puntos del mundo.

Verificar el certificado SSL

Para garantizar una conexión segura (HTTPS), es indispensable que el certificado SSL esté instalado y configurado correctamente. Puedes comprobarlo en navegadores o mediante servicios como SSL Labs, que analizan el estado del certificado y muestran posibles vulnerabilidades.

Comprobar la configuración del servidor web

Es importante acceder al sitio y confirmar que responde adecuadamente. Usar herramientas como curl -I permite inspeccionar los encabezados HTTP para detectar redirecciones incorrectas o códigos de error (como 404 o 500). Esto ayuda a identificar configuraciones erróneas en el servidor.

Revisar la configuración de correo electrónico (si aplica)

Si el dominio está vinculado a servicios de email, se deben revisar:

- Que los registros MX apunten correctamente.
- Que existan y estén bien configurados los registros SPF, DKIM y DMARC, lo cual es clave para evitar que los correos lleguen como spam o sean rechazados.

Validar el tiempo de respuesta y rendimiento

Finalmente, se recomienda analizar la velocidad de respuesta del sitio y su rendimiento general. Herramientas como GTmetrix o PageSpeed Insights brindan información útil sobre tiempos de carga, peso de los recursos y oportunidades de optimización.

¿UNA APLICACIÓN WEB SE VE IGUAL EN TODOS LOS EXPLORADORES WEB?

No necesariamente. Aunque una aplicación web esté diseñada siguiendo estándares, su apariencia y funcionamiento pueden variar dependiendo del navegador utilizado. Esto se debe a diferencias en los motores de renderizado, la compatibilidad con tecnologías web y los estilos predeterminados de cada navegador.

Motores de renderizado diferentes:
Cada navegador utiliza un motor de renderizado distinto para interpretar el código:

1. Chrome y Edge: Blink.
2. Firefox: Gecko.
3. Safari: WebKit.

Estos motores interpretan HTML, CSS y JavaScript de forma ligeramente distinta, lo cual puede provocar cambios en la disposición de los elementos, animaciones, o incluso errores visuales.

Compatibilidad con estándares web:

A pesar de que HTML5, CSS3 y JavaScript tienen especificaciones bien definidas, no todos los navegadores adoptan las nuevas características al mismo tiempo. Algunas propiedades CSS modernas, APIs del navegador o funciones de JavaScript podrían no estar disponibles o comportarse diferente en versiones antiguas o menos populares.

Estilos predeterminados del navegador:

Los navegadores aplican estilos básicos propios a los elementos HTML, como márgenes, fuentes y tamaños. Esto puede generar variaciones si no se definen estilos personalizados de manera explícita. Por ello, es común el uso de *CSS resets* o *normalize.css* para unificar la base visual en todos los entornos.

Diferencias en la interpretación de JavaScript:
Aunque el lenguaje es estándar, su interpretación puede diferir en detalles como el manejo de errores o el rendimiento. Esto afecta directamente la experiencia de usuario si no se prueba adecuadamente en los navegadores más comunes.

Dispositivos y sistemas operativos:
Además del navegador, el sistema operativo (Windows, macOS, Linux, Android, iOS) también influye en la presentación. Tipografías, antialiasing, colores, y comportamientos nativos pueden cambiar dependiendo del entorno en el que se visualice la aplicación.

¿CÓMO SE VE UNA APLICACIÓN WEB EN UN EXPLORADOR DEL CELULAR?

La visualización de una aplicación web en un navegador móvil difiere significativamente de su presentación en un entorno de escritorio. Estas diferencias están principalmente relacionadas con el tamaño de la pantalla, la forma de interactuar del usuario y las características del dispositivo.

Diseño adaptable a pantallas pequeñas

Los dispositivos móviles cuentan con pantallas más reducidas, lo que requiere que las interfaces se adapten a ese espacio. Para lograrlo, se aplican principios de diseño responsivo, permitiendo que los elementos se reorganicen automáticamente en función del ancho del dispositivo.

El uso de media queries en CSS es clave para definir estilos específicos según el tamaño del viewport. Así, menús, imágenes y bloques de texto se ajustan sin necesidad de hacer zoom o desplazamientos excesivos.

Orientación del dispositivo
Una aplicación debe funcionar correctamente tanto en modo vertical (portrait) como en horizontal (landscape). Esto implica que los elementos deben redistribuirse de forma fluida y sin afectar la experiencia del usuario.

Interacción táctil

A diferencia de la navegación con mouse, en móviles se utiliza la pantalla táctil. Por eso, los botones, enlaces y otros elementos interactivos deben tener un tamaño mínimo (idealmente 48x48 píxeles) para facilitar su manipulación sin errores.

Navegación simplificada

Los menús de navegación suelen ser colapsables (como el ícono "hamburguesa") para ahorrar espacio y evitar saturar la interfaz. Además, la estructura debe ser intuitiva, permitiendo que el usuario acceda fácilmente a la información con la menor cantidad de toques posible.

Formularios optimizados

Los formularios deben ajustarse al uso de teclados virtuales. Por ejemplo, al ingresar un correo electrónico, el campo debe activar un teclado que incluya el símbolo "@". Además, se recomienda implementar **autocompletado** y validaciones claras para mejorar la usabilidad.

Imágenes y contenido multimedia

Las imágenes deben estar optimizadas para cargar rápido y ocupar poco espacio. Se recomienda el uso de formatos como **WebP** y técnicas de compresión para garantizar un buen rendimiento sin sacrificar calidad.

Problemas comunes en móviles

- Elementos que se salen de la pantalla por falta de diseño responsivo.
- Textos demasiado pequeños o apretados.
- Enlaces o botones mal posicionados, difíciles de pulsar.
- Incompatibilidad con navegadores móviles desactualizados.

QUÉ ELEMENTOS SE DEBEN PROBAR EN UNA APLICACIÓN WEB

Al validar el funcionamiento de una aplicación web, es esencial identificar qué componentes deben ser objeto de prueba para asegurar una experiencia funcional, segura y coherente para los usuarios. A continuación, se detallan los principales elementos que deben ser verificados durante un proceso de pruebas:

Los Formularios

- Verificación de campos obligatorios, límites de caracteres y formatos válidos (como correos electrónicos o contraseñas).
- Comprobación del envío correcto de datos y respuestas del sistema ante errores o entradas inválidas.
- Evaluación de los mensajes de error para que sean claros y orienten al usuario a corregir su acción.

Enlaces y navegación

- Asegurarse de que todos los enlaces redirijan correctamente a su destino, tanto internos como externos.
- Validar que los menús, rutas y botones de navegación funcionen adecuadamente, sin generar confusión o errores.

Botones y acciones

- Confirmar que cada botón cumpla su función específica: guardar, eliminar, enviar, etc.
- Verificar su estado según el contexto (activo, deshabilitado, cargando) y que respondan visualmente al hacer clic o tocar.

Funcionalidades específicas

- Flujos de autenticación: inicio de sesión, registro, cierre de sesión y recuperación de contraseña.
- Procesos como carritos de compra, pagos, filtros de búsqueda, generación de reportes, entre otros.

Diseño responsivo

- Validar que la interfaz se adapte a múltiples resoluciones y dispositivos (smartphones, tablets, monitores).

- Probar la disposición de los elementos al cambiar entre orientación vertical y horizontal.

Accesibilidad

- Comprobar que el sitio pueda ser usado por personas con discapacidades, usando lectores de pantalla, navegación con teclado o herramientas de alto contraste.

- Seguir las recomendaciones de las Pautas de Accesibilidad al Contenido en la Web (WCAG).

Navegación intuitiva

- Asegurarse de que los usuarios puedan acceder a todas las funcionalidades de forma lógica e intuitiva.

- Evaluar si el flujo de interacción permite alcanzar objetivos sin necesidad de pasos innecesarios.

Legibilidad

- Comprobar que los textos estén correctamente escritos, con buena ortografía, tamaño y contraste.

- Validar que el contenido visual (imágenes, videos, íconos) se vea correctamente en todos los dispositivos.

Pruebas de compatibilidad

- Navegadores: probar en Chrome, Firefox, Safari, Edge y otros navegadores relevantes.

- Dispositivos: comprobar funcionamiento en iOS, Android, Windows y macOS.

- Versiones: revisar cómo se comporta en versiones anteriores de navegadores, si se requiere soporte.

Pruebas de rendimiento

- Medir el tiempo de carga en distintas condiciones de red.
- Evaluar el comportamiento de la aplicación bajo múltiples usuarios simultáneos.
- Verificar el consumo de recursos del navegador (CPU, memoria).

Pruebas de seguridad

- Revisar el control de acceso y expiración de sesiones.
- Asegurar el cifrado de datos sensibles (contraseñas, pagos).
- Detectar vulnerabilidades comunes como XSS, CSRF e inyección SQL.
- Confirmar que el sitio utilice HTTPS y tenga un certificado SSL válido.

Pruebas de contenido

- Verificar que imágenes, audios y videos se carguen correctamente y se ajusten al diseño responsivo.
- Detectar y corregir enlaces rotos o recursos faltantes.

Pruebas de integración

- Validar que las API conectadas respondan correctamente y manejen errores de forma controlada.
- Asegurar la integración con servicios como pagos, mapas, autenticación externa o notificaciones.

Pruebas de regresión

- Evaluar que nuevas actualizaciones no afecten funcionalidades ya existentes.
- Automatizar pruebas repetitivas para aumentar la cobertura y eficiencia.

Pruebas de recuperación

- Validar que exista un sistema de respaldo y restauración funcional.
- Simular errores como caídas del servidor o interrupciones de red para comprobar la respuesta del sistema.

CONCLUSIONES

- Las pruebas funcionales representan un pilar esencial en el desarrollo de software, ya que permiten validar que cada parte del sistema cumple su propósito y responde correctamente ante las acciones del usuario.

- Mediante la ejecución de pruebas manuales y automatizadas, es posible verificar aspectos clave como la navegación, la lógica de negocio, la integridad de los datos y la experiencia de usuario.

- Incluir las pruebas como una etapa continua dentro del ciclo de desarrollo ayuda a detectar errores desde fases tempranas, reduciendo costos y evitando fallos en producción.

- La evaluación funcional también implica adaptarse a diferentes entornos: navegadores, dispositivos, sistemas operativos y condiciones de red, lo que exige un enfoque integral de validación.

- Aunque los estándares web buscan uniformidad, cada navegador puede interpretar el código de forma distinta. Por ello, es indispensable realizar pruebas cruzadas para garantizar una experiencia coherente en cualquier plataforma.

- En el entorno móvil, la importancia del diseño responsivo y la eficiencia de carga se vuelve crítica, ya que gran parte del tráfico web proviene actualmente de smartphones y tablets.

- En definitiva, aplicar pruebas de funcionalidad no solo garantiza un software más robusto, sino también una mejor satisfacción del usuario y una mayor confiabilidad del sistema en su conjunto.

web, sin embargo, se puede minimizar las diferencias siguiendo buenas prácticas de desarrollo, realizando pruebas exhaustivas y utilizando herramientas de normalización y detección de características.

- Una aplicación web en un navegador de celular debe ser responsiva, rápida y fácil de usar. La experiencia móvil es crucial, ya que una gran parte del tráfico web proviene de dispositivos móviles.

BIBLIOGRAFIA

- https://www.hostinger.com/co/tutoriales/test-de-usabilidad-web
- https://comparium.app/es/blog/web-test-tools/how-to-test-web-application/
- https://gc.scalahed.com/recursos/files/r161r/w21843w/presentation_content/external_files/U7.pdf
- https://www.browserstack.com/guide/web-application-testing