Índice del Glosario

Prueba de Software

- ✧ Verificar si cumple con los requisitos esperados.
- ✧ Manuales y Automaticas.
- ✧ Que el software esta sin bugs (Errores).

¿Por qué es importante?

- ✓ Para identificar temprano los errores.
- ✓ Confiable.
- ✓ Satisfacción del cliente.

¿Cuál es la necesidad de las pruebas?

- ✓ Perdidas economicas.
- ✓ Perdidas humanas.
- ✓ Desprestigio Empresa.

¿Cuáles son los beneficios?

- ✓ Rentable ahorramos dinero.
- ✓ Seguridad.
- ✓ Calidad del software 100% mínimo.
- ✓ Satisfacción del cliente.

Tipos de pruebas de software

Pruebas funcionales **¿El software cumple con lo esperado?**

- ➢ Pruebas unitarias.
- ➢ Pruebas de integración .
- ➢ Pruebas de sistema.
- ➢ Pruebas de aceptación.

Pruebas no funcionales

- ➢ Pruebas de rendimiento.
- ➢ Pruebas de seguridad.
- ➢ Pruebas de usabilidad.

Mantenimiento

- ➢ Regression.
- ➢ Mantemiento.

7 principios de las pruebas de software

 # 1 No es posible realizar pruebas exhaustivas
Necesitamos una cantidad optima de pruebas.

 # 2 Agrupación de defectos
Modulo reporte 20% defectos.

 # 3 Paradoja de los pesticidas
- Mejorar las pruebas.
- Las pruebas que son repetitivas automatizarlas.

 # 4 Las pruebas muestran la presencia de defectos
Sus pruebas están enfocadas a detectar defectos.

 # 5 Ausencia de Error – falacia
Libre de errores al 99% pero no se pueda usar.

 # 6 Pruebas tempranas
Nos ahorra tiempo y dinero.

 # 7 Las pruebas dependen del contexto
No es lo mismo probar una app bancaria que una app de compras.

EL Modelo V

 # Terminos a conocer

- **SDLC** (Software Development Life Cycle).
- **STLC** (Software Testing Life Cycle).

 # ¿Que es el Modelo V?

Analisis Pruebas
Diseño Pruebas
Implementación Pruebas
Verificación Pruebas
Mantenimiento

 # Problemas con el modelo cascada

- Poco margen para realizar ajustes a lo largo del proyecto debido a un cambio en las exigencias.
- El usuario final no se integra en el proceso de producción hasta que no termina la programación.
- En ocasiones, los fallos solo se detectan una vez finalizado el proceso de desarrollo.

 # Solucion al modelo Cascada

Modelo V.

 # Conclusion

- La prueba no es una actividad independiente y tiene que adaptarse al modelo de desarrollo elegido para el proyecto.
- En cualquier modelo, las pruebas deben realizarse en todos los niveles, es decir, desde los requisitos hasta el mantenimiento.

Ciclo De Vida

 # ¿Qué es el STLC?

1) Es una secuencia de actividades realizadas durante el proceso para garantizar que se cumpla la calidad.
2) Abarca tanto actividades de verificación como de validación, y no se limita a una única. prueba aislada, sino que comprende una serie de pasos sistemáticos.

 # Fases STLC

 # 1 Analisis de requisitos

En esta fase, se analizan los requisitos del software para comprender sus funcionalidades y características clave. Se identifican los escenarios de prueba y se establecen las bases para el proceso de prueba.

 # 2 Planificación de pruebas

Aquí se crea un plan detallado que define el alcance.

 # 3 Desarrollo de casos de prueba

- Se definen los pasos, los datos de entrada y se espera la salida esperada.
- HLTCs Verificar que le buscador muestre los resultados relacionados.
- Tests cases.

 # 4 Configuración del entorno de prueba:

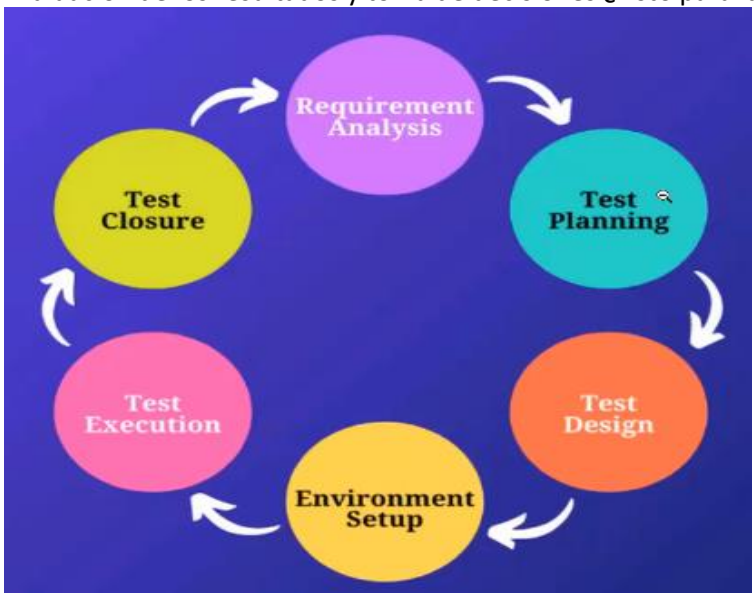
- Configuración de servidores.
- Creación de base de datos.
- Instalación de software necesario.

 # 5 Ejecución de pruebas

- Ejecutar los casos de prueba.
- Generar informes.

 # 6 Cierre del ciclo de prueba

Evaluación de los resultados y toma de decisiones ¿Listo para lanzarlo?.



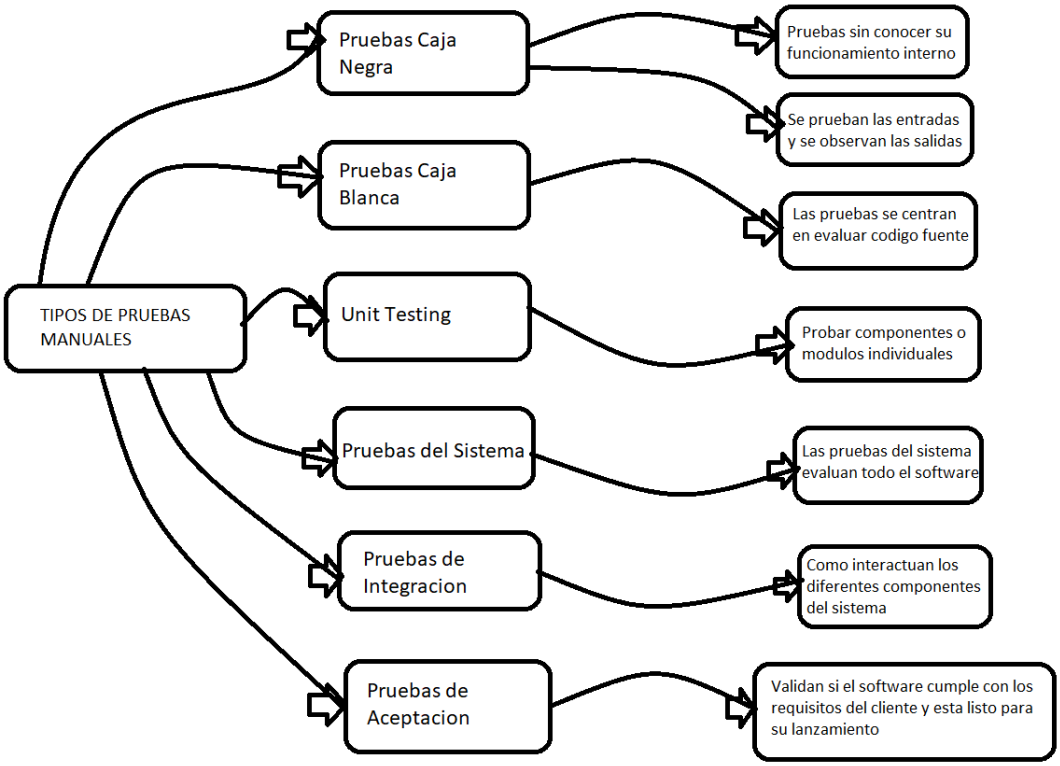
Pruebas Manuales

- ✧ El QA ejecuta pruebas de software manualmente sin herramientas de automatización.
- ✧ Cualquier app debe probarse manualmente antes de automatizarse.
- ✧ El 100% de automatización no es posible.

 # Objetivo de las pruebas manuales

- ✓ Garantizar que la aplicación esté libre de errores.
- ✓ Cobertura de prueba del 100%.
- ✓ Que los DEVs corrijan los bugs(Errores) reportados.

 # Tipos de pruebas Manuales



 # ¿Como realizar pruebas manuales?

- ✓ Leer y comprender la documentación/guías.
- ✓ casos de prueba que cubran todos los requisitos mencionados en la documentación.
- ✓ Ejecutar los casos de prueba.
- ✓ Reportar errores.
- ✓ Una vez que se solucionen los errores, vuelva a ejecutar los casos de prueba fallidos.

 # Pruebas Manuales vs Pruebas de Automatización

Manuales	Automaticas
Requiere la intervención humana	Uso de herramientas
Mano de obra calificada	Ahorran tiempo y mano de obra
Cualquier tipo de software se puede probar	Solo para sistemas estables
Repetitivas aburridas	La parte aburrida se reduce

 # Herramienta para pruebas Automaticas

Selenium	Jmeter
Loadrunner	Cypress

Pruebas Unitarias

- ✧ validan unidades más pequeñas de software; funciones y métodos de forma aislada desde dentro del sistema.

OBJETIVO:
detectar y corregir errores en una fase temprana del desarrollo, garantizando que cada unidad de software funcione correctamente.

TIPOS:
- ❖ Funciones y métodos: validan que produzcan resultados correctos para diferentes empleados.
- ❖ Lógica y negocios: validan la parte lógica de la aplicación.
- ❖ Prueba de integración: verifican la integración entre componentes.

HERRAMIENTAS:

Pruebas Integración

✧ se combinan componentes individuales de un sistema para validar su interacion.

OBJETIVO
asegurar que los diferentes módulos de software funcionen correctamente juntos como un sistema completo.

ENFOQUES PRINCIPALES
arriba hacia abajo (top - down)
las pruebas comienzan desde los módulos superiores a inferiores (los módulos principales/ funcionalidades claves se prueban primero y luego se ejecutan gradualmente las pruebas para módulos mas pequeños).

demoblaze.com -- probamos que funcione el carrito agregando un iphone -- comprarlo (ya que es lo primordial para ganar dinero).

abajo hacia arriba (button - up)
las pruebas comienzan desde los módulos mas bajo del sistema (los mas simples se prueban primero y luego se agrupan y prueban en combinación mas grande hasta que se halla probado todo el sistema).

demoblaze.com -- me fijo cada categoría de productos (ya que me permite ver de cada producto diferentes modelos) y termino en carrito haciendo combinaciones de agregar diferentes productos.

Pruebas Sistema

✧ se centra en evaluar que el sistema completo como entidad única este funcionando correctamente y cumpla con los requisitos especificados.

EJEMPLO: *demoblaze.com*

se enfocarían que el sitio web funcione correctamente y se espere lo que el cliente requiera.

Pruebas Regresión

✧ Asegurar que los cambios recientes en el código no hayan roto nada.

PASO 1	Identificar las Funcionalidades a Probar
PASO 2	Crear un Conjunto de Casos de Prueba
PASO 3	Automatizar Casos de Prueba
PASO 4	Ejecutar las Pruebas Automatizadas
PASO 5	Análisis de Resultados
PASO 6	Repetir el Proceso

 # Consejos Adicionales
✓ Mantén una Suite de Pruebas Robusta.
✓ Automatización Eficiente.
✓ Versionamiento.

Pruebas Funcionales - No Funcionales

 # Funcional Testing (lo que la aplicación tiene que hacer)

- *UAT:* se realiza por el cliente para determinar si el sistema cumple con los criterios de aceptación (beta tester).
- *Exploratory Testing:* se basa en la experiencia y habilidad de tester para descubrir los defectos.
- *Sanity Testing:* pruebas rápidas para asegurar que una versión del software funcione bien antes de seguir con pruebas mas exhaustivas.
- *Regression Testing:* asegura que una modificación de código no afecte funcionalidad existente.
- *Smoke Testing:* pruebas iniciales del software asegurando que las funciones criticas funciones correctamente (son Sanity).
- *Unit Testing:* partes individuales del software a funciones o métodos (se escribe código y casi siempre lo hacen los devs).
- *Integration Testing:* verifica si un software esta compuesto por muchos componentes o módulos (información de celulares en BD -- 2 módulos).

 # Non Functional Testing (como se comporta la aplicación)

- *Load Testing:* cargas pesadas para ver como reacciona la pagina (múltiples usuarios).
- *Performance Testing:* evalúa la velocidad, capacidad de respuesta, estabilidad con diferentes cargas de usuarios o volúmenes de datos.
- *Stress Testing:* estresan la aplicación superando los limites (alta carga de usuarios).
- *Security Testing:* verifican la vulnerabilidad y seguridad de la pagina.
- *Accessibility Testing:* evalúa que tan accesible es para personas con discapacidad.

Usabilidad	Evalúa la facilidad con la que los usuarios pueden interactuar con el sistema.
Rendimiento	Evalúa cómo se comporta el sistema en términos de velocidad, respuesta y estabilidad bajo diferentes condiciones.
Seguridad	Evalúa la resistencia del sistema contra ataques y su habilidad para proteger datos sensibles.
Fiabilidad	Evalúa la capacidad del sistema para realizar funciones requeridas sin fallos durante un período de tiempo específico.
Mantenibilidad	Evalúa la facilidad con la que se pueden realizar cambios en el sistema, como correcciones de errores o mejoras.
Compatibilidad	Evalúa la capacidad del sistema para funcionar en diferentes entornos y configuraciones.
Eficiencia	Evalúa el uso eficiente de los recursos del sistema, como memoria y capacidad de procesamiento.
Escalabilidad	Evalúa la capacidad del sistema para manejar un aumento en la carga de trabajo o el número de usuarios.

Sanity Testing

- ✧ Se realizan después de una nueva versión o cambio menor para verificar las funciones criticas puedan realizarse con un guion o no (contactos - cuantos caracteres soporta, si valida el mail). Son menos exhaustivas que las pruebas de regresión por ejemplo y se enfocan en el área especifica que se modifico.

 # Pasos para realizar Sanity Testing

1) Entender los Requisitos.
2) Selección de Escenarios de Prueba.
3) Preparar el Entorno de Prueba.
4) Ejecutar Pruebas.
5) Comparar Resultados.
6) Tomar Decisiones: La app esta lista para salir a producción.

 # Consejos para realizar Sanity Testing de manera efectiva

- ✓ Automatización.
- ✓ Enfoque en lo Crítico.
- ✓ Documentación.
- ✓ Retroalimentación Rápida.

Smoke Testing

- ✧ Verifican si las funciones básicas del programa funcionan correctamente (que se vea la pestaña carrito). Se llevan a cabo antes de pruebas detalladas o exhaustivas.

OBJETIVO

determinan si el software esta suficientemente estable para pruebas mas duras. Si fallan, significa que hay problemas graves para el funcionamiento del mismo.

Pruebas Exploratorias

✧ las pruebas exploratorias implican la exploración libre del software (son útiles al principio cuando no hay documentación y se necesita realizar pruebas rápidas).

 # Características de las pruebas exploratorias

1) No hay escenarios predefinidos.
2) Adaptabilidad.
3) Uso intensivo de la experiencia y habilidades del probador.
4) Descubrimiento temprano de defectos.

 # Cómo se realizan las pruebas exploratorias

1) Entender el contexto.
2) Explorar el software.
3) Interactuar con el software.
4) Registrar defectos.
5) Retroalimentación continua.
6) Iterar según sea necesario.

IMPORTANTE

con f12 o inspeccionar elemento --- lighthouse:

1. **Performance** es rendimiento, ancho de banda, etc.
2. **Accessibility** es para personas con discapacidad visual, etc.
3. **Best Practices** es practicas de código.
4. **Seo** es desde el motor de búsqueda que tan optimizado esta la pagina.

Regression Testing

✧ Son pruebas que se realizan para comprobar que los cambios nuevo no afectaron las cosas que ya funcionaban.

 # ¿Cómo se hacen las pruebas de regresión?

1) Selección de casos de prueba.
2) Automatización de pruebas.
3) Ejecución de pruebas.
4) Comparación de resultado.
5) Repetición regular.
6) Gestión de versiones.
7) Optimización de casos de prueba.

 # Pruebas Versión 1.1, Versión 1.19 -> Versión 2.0 (a clientes en mercado)

1. Confirmar que la página web se carga correctamente. PASS

Resultados esperados

2. Verificar que todos los enlaces de la página funcionan correctamente. PASS
3. Confirmar que los formularios en la página funcionan correctamente. PASS
4. Confirmar que la página es responsiva (se adopte a diferentes dispositivos y que no se descuadre nada) y se muestra correctamente. PASS

Verificación y Validación

 # Verificación

- ❖ Cumple con las especificaciones y los requisitos durante su construcción.
- ❖ Se trata de verificar que el producto se está construyendo correctamente, siguiendo las especificaciones y estándares.

 # Validación

- ❖ Cumple con los requisitos del cliente al final del proceso.
- ❖ Se trata de validar que el producto cumple con las necesidades reales del usuario/cliente.

Caja Blanca

 # Pruebas de caja blanca

- ✓ son un tipo de prueba de software que evalúa el código interno de un programa.

 # Cómo se realizan las pruebas de caja blanca

1) Análisis de flujo de control.
2) Diseño de casos de prueba.
3) Ejecución de pruebas.
4) Análisis de cobertura.

Caso de prueba 1:
verificar los números positivos
3 + 4 = 7
Caso de prueba 2:
-3 + 7 = 7
Caso de prueba 3:
-3 + -5 = -5

Caja Negra

 # Pruebas de caja Negra

- ✓ probar las entradas y salidas (porque no ves el código, solo lo visual de la pagina web o aplicación).

al QA no le interesa el codigo fuente.
Ejemplo: registrar un usuario (nombre, mail y contraseña) (entrada), veo mi cuenta y verifico que los datos ingresados son correctos (salida).

Documentación de Pruebas

1) Plan de Pruebas.
2) Casos de Prueba.
3) Informes de Pruebas.
4) Matriz de Rastreo de Requisitos.

Requisito	**Casos de Prueba Asociados**
R1	Inicio de Sesión Exitoso
R2	Recuperación de Contraseña
R3	Validación de Entradas

5) Registro de Defectos.
6) Estrategia de Pruebas.

Escenario de Pruebas

Título del Escenario: [Describe breve mente qué se está probando].
Objetivo: [Propósito de la prueba, qué se quiere lograr].
Pre-condiciones: [Condiciones necesarias para ejecutar la prueba, por ejemplo, datos preexistentes, configuraciones específicas].
Pasos de Ejecución: [Pasos específicos para ejecutar la prueba, acciones que el probador debe realizar].
Criterios de Éxito: [Cómo se determinará si la prueba es exitosa].
Pos-condiciones: [Estado esperado del sistema después de completar la prueba].

EJEMPLO

Título del Escenario: Verificación del inicio de sesión de usuario.
Objetivo: Asegurar que los usuarios puedan iniciar sesión en la aplicación.
Datos de prueba: Usuario: Admin / Contraseña: Admin.
Pre-condiciones:

- ✓ Un usuario registrado.
- ✓ Un navegador.
- ✓ Una conexion a Internet.

Pasos de Ejecución:

1. Abrir la aplicación de wikipedia en la url *https://www.wikipedia.org.*
2. Hacer clic en acceder.
3. Ingresar un usuario valido y contraseña.
4. Hacer clic en el boton acceder.

Criterios de Éxito: El usuario es redirigido a la pagina principal de la aplicación después de iniciar sesión correctamente.
Pos-condiciones: El usuario ha iniciado sesión y puede acceder a las funcionalidad de la aplicación.

Casos de Prueba

Caso de Prueba 1: Búsqueda Exitosa. PASS
Descripción: Verificar que el cuadro de búsqueda pueda encontrar un producto existente cuando se ingresa su nombre correctamente.
Pasos:
1) Ingresar el nombre del producto ("AN/FSQ-32") en el cuadro de búsqueda.
2) Presionar "Enter" o hacer clic en el botón de búsqueda.
Resultado Esperado: La página debe mostrar los resultados relacionados con el producto ingresado.

Caso de Prueba 2: Búsqueda sin Resultados. FAIL
Descripción: Verificar que el cuadro de búsqueda informe correctamente cuando no se encuentra ningún producto con el término ingresado.
Pasos:
1) Ingresar un término de búsqueda que no esté relacionado con ningún producto existente (por ejemplo, "ZZZ123").
2) Presionar "Enter" o hacer clic en el botón de búsqueda.
Resultado Esperado: La página debe mostrar un mensaje indicando que no se encontraron resultados para el término de búsqueda ingresado.

EJEMPLO

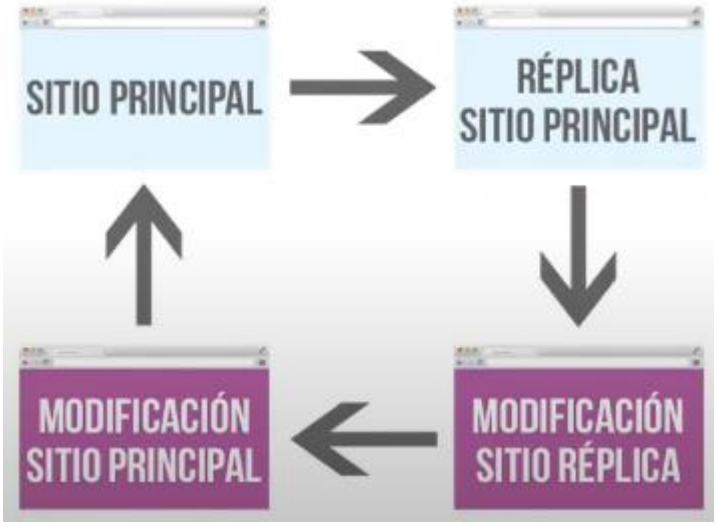
Título del Caso de Prueba: Verificación del Inicio de Sesión en la Aplicación
Objetivo: Probar el proceso de inicio de sesión en la aplicación.
Descripción:
El propósito de esta prueba es verificar que el inicio de sesión de la aplicación estén funcionando correctamente en los navegadores chrome, firefox, etc.
Pre-condiciones:
- ✓ La aplicación está instalada en el dispositivo.
- ✓ El usuario tiene credenciales válidas.

Pasos:
1. Abrir la aplicación de la pantalla de inicio.
2. Hacer clic en el botón "Iniciar Sesión".
3. Ingresar el nombre de usuario y la contraseña en los campos correspondientes.
4. Hacer clic en el botón "Iniciar Sesión".

Resultado Esperado:
El usuario debe ser redirigido a la página de inicio después de iniciar sesión correctamente.
Criterios de Éxito:
Después de hacer clic en "Iniciar Sesión", el usuario es redirigido a la página de inicio.
No hay mensajes de error mostrados durante el proceso de inicio de sesión.

Escenario: Inicio de sesión en una aplicación de redes sociales.
Descripcion: Verificar el proceso de inicio de sesión en una aplicación de RS.
Caso de prueba 1: Verificar las credenciales correctas
Caso de prueba 2: Verificar el login con credenciales incorrectas.
Caso de prueba 3: Verificar múltiples intentos de login con credenciales incorrectas.

Entorno de Prueba



Tipos de Entornos de Pruebas:

1) *Entorno de Desarrollo (Development Environment):* Es donde los desarrolladores escriben y prueban el código.
2) *Entorno de Pruebas (Testing Environment):* Es una copia del entorno de producción.
3) *Entorno de Staging (Staging Environment):* Es una copia exacta del entorno de producción.
4) *Entorno de Producción (Production Environment).*

Pasos para Configurar un Entorno de Pruebas:

1. *Planificación:*
   - Identificar los requisitos de prueba.
   - Determinar qué aspectos del software se probarán.

2. *Configuración del Hardware y Software:*
   - Configurar servidores y redes similares a las del entorno de producción.
   - Instalar el software necesario, incluidas las herramientas de prueba y las aplicaciones bajo prueba.

3. *Creación de Datos de Prueba:*
   - Se debe tener cuidado con los datos de los clientes.

Ejecución de pruebas en el entorno de pruebas:

1. Ejecución de Pruebas.
2. Registro y Seguimiento.
3. Optimización y Retesting.
4. Aprobación para Despliegue.

Tecnicas de Prueba

✧ Las técnicas de prueba de software lo ayudan a diseñar mejores casos de prueba. Dado que no es posible realizar pruebas exhaustivas.

1) Analisis de valor Limite (Boundary)

- Condición de entrada 1 - 10

Pruebas:

0,1,2 y 9,10,11

2) Partición de clases de equivalencia

- Condición de entrada: 1 a 10 y 20 a 30

Pruebas:

- a 0 (no válido)

1 a 10 (válido)

11 a 19 (no válido)

20 a 30 (válido)

31 a - (no válido)

Seleccion de valores: -2, 3, 15, 25, 45

3) Pruebas basadas en tablas de decisiones

- Condición de entrada:

➢ Tipo de usuario (Estudiante, profesor, miembro publico).
➢ Tipo de libro (ficción, no ficción, referencia).
➢ Estado del libro (disponible, prestado).

Tipo de Usuario	Tipo de Libro	Estado del Libro	Acción Esperada
Estudiante	Ficción	Disponible	Permitir Préstamo
Estudiante	Ficción	Prestado	Denegar Préstamo
Estudiante	No Ficción	Disponible	Permitir Préstamo
Estudiante	No Ficción	Prestado	Denegar Préstamo
Estudiante	Referencia	Disponible	Denegar Préstamo
Profesor	Ficción	Disponible	Permitir Préstamo
Profesor	Ficción	Prestado	Denegar Préstamo
Profesor	No Ficción	Disponible	Permitir Préstamo
Profesor	No Ficción	Prestado	Denegar Préstamo
Profesor	Referencia	Disponible	Denegar Préstamo
Miembro Público	Ficción	Disponible	Permitir Préstamo
Miembro Público	Ficción	Prestado	Denegar Préstamo
Miembro Público	No Ficción	Disponible	Permitir Préstamo
Miembro Público	No Ficción	Prestado	Denegar Préstamo
Miembro Público	Referencia	Disponible	Denegar Préstamo

4) Transición de estado

❖ Cuando los cambios en las condiciones de entrada cambian el estado de la aplicación.

Inicio se sesión:

➢ Estado Inicial (AUT Desconectada).
➢ Transición de Estado - Intento de Inicio de Sesión Exitoso.
➢ Transición de Estado - Intento de Inicio de Sesión Fallido.
➢ Transición de Estado - Cuenta bloqueada.

5) Error al adivinar

❖ Se basa en la experiencia de los analistas y se basa en adivinar donde podría estar el error.

Estimación

- ✧ La estimación de pruebas de software es el proceso de calcular o predecir la cantidad de esfuerzo, tiempo y recursos necesarios para llevar a cabo las actividades de prueba.

QUE ESTIMAR?
el tiempo, recursos humanos, herramientas y entorno de pruebas, cantidad de pruebas necesarias según la complejidad del software y requisitos y documentación.

EJEMPLO:
Aplicación de Comercio Electrónico.
necesidades basicas: buscar, agregar productos en carrito, realizar pago, ver historial de pedidos.
ESTIMACION: Tiempo determinado por equipos, ahorro de tiempo - ahorro de dinero.
PASOS DE COMO ESTIMAR:

1) Entender los requisitos.
2) Descomposición de trabajo (crear casos de prueba para: funcionalidad de búsqueda, carrito de compras, proceso de pago y historial de pedidos).
3) Uso de tecnica de estimacion (crear casos de prueba para: busqueda 2 dias, carrito 3 dias, pago 4 dias, historial 2 dias).
4) Tamaño y Complejidad de cada función.
5) Considerar los riesgos (ing de vacas o mantenimiento y no podemos hacer ambientes de prueba).
6) Revisión y Validación.
7) Documentación Detallada.
8) Comunicación Clara.
9) Flexibilidad.

Generación de Datos

- ✧ La generación de datos de prueba se refiere al proceso de crear conjuntos de datos sintéticos que imitan las características y estructuras de los datos reales

Herramientas para Generación de Datos de Prueba:

1. Mockaroo: *https://www.mockaroo.com*
2. Faker (en Python)
3. SQL Data Generator: *https://www.red-gate.com/products/sql-data-generator/*
4. DataFactory (en Azure)
5. Postman

Test Basis

Especificaciones de Requisitos del Usuario (ERU) o Requisitos del Cliente:
Estos documentos describen las necesidades y expectativas del cliente o usuario final del software.

Especificaciones de Requisitos del Sistema (ERS) o Requisitos del Sistema:
Estos documentos detallan los requisitos técnicos y funcionales del sistema.

Documentos de Diseño del Sistema o Diseño Técnico:
Estos documentos contienen información detallada sobre la arquitectura del sistema

Casos de Uso:
Los casos de uso describen cómo los usuarios interactúan con el sistema en situaciones específicas.

Diagramas de Flujo:
Estos diagramas muestran el flujo de datos y las interacciones entre diferentes componentes del sistema.

Prototipos y Maquetas:
Si se han creado prototipos o maquetas del software.

Manuales de Usuario:
Los manuales de usuario proporcionan información detallada sobre cómo utilizar el software.

Documentación de Proceso:
La documentación que describe el proceso de desarrollo utilizado para construir el software también puede ser útil.

Web Testing

- ✧ prueba de sitios web o pruebas de aplicaciones web.

Pruebas de funcionalidad

- ❖ Todas las funciones de la pagina funcionen; formularios - enlaces- cookies - sesiones de usuario etc.

Pruebas de usabilidad

- ❖ Evalúan la facilidad de uso y experiencia del usuario en la pagina; navegación - diseño - legibilidad del contenido etc.

Pruebas de rendimiento

- ❖ Evaluar la velocidad, Escalabilidad y estabilidad de la pagina; cargas pesadas, trafico simultaneo, picos de uso.

Pruebas de seguridad

- ❖ Identifica vulnerabilidades; ataques Inyeccion SQL, Crosside, Scripting.

Pruebas de compatibilidad

- ❖ Se vea y funcione correctamente en diferentes sistemas, navegadores.

Pruebas de contenido

- ❖ Verifican la precisión y la relevancia del contenido de la pagina.

Pruebas de enlace

- ❖ Verifica que los Links funcionen y lleven a los sitios correctos.

SE PUEDEN USAR MANUALMENTE O AUTOMATICO

Pruebas Funcionales: WebApps

Pruebas Funcionales en Wikipedia - Funcionalidad

- ✧ Se realizan para asegurar que un sitio web funcione según lo previsto y que los usuarios puedan interactuar con él de manera efectiva y sin problema.

1) Planificación:

1. Identifica las funciones clave

- a) Búsqueda en el motor de búsqueda de Wikipedia.
- b) Navegación por diferentes categorías y secciones.
- c) Visualización de páginas con contenido diverso (artículos, imágenes, videos).
- d) Edición de una página existente.

2. Define escenarios de prueba

- a) Escenario de búsqueda: Buscar "Inteligencia Artificial" y verificar que la página de resultados incluye un enlace relevante y que al hacer clic en él, la página se carga correctamente.

2) Ejecución de Pruebas:

1. Pruebas de navegación:

- Verificar que se pueda navegar desde la página de inicio a diferentes categorías como Historia, Ciencia, Tecnología, etc., y que las páginas se carguen correctamente. PASSED

2. Pruebas de búsqueda:

- Buscar "Albert Einstein" y verificar que la página del resultado principal muestra información relevante sobre Albert Einstein. PASSED

3. Pruebas de contenido:

- Abrir una página sobre un tema específico (por ejemplo, "El Renacimiento") y verificar que el contenido, las imágenes y los enlaces funcionan correctamente. PASSED

4. Pruebas de formularios:

- Editar una página y verificar que los campos de edición funcionan correctamente y que los mensajes de error se muestran si se intenta enviar un formulario incompleto. PASSED

5. Pruebas de interactividad:

- Verificar que los botones como "Editar", "Historial" y "Enlaces" funcionen correctamente en las páginas de artículo. PASSED

3) Verificación de Dispositivos y Navegadores:

1. Pruebas multi-navegador:

- Acceder a Wikipedia desde Chrome, Firefox, Safari y Edge, y verificar que la página se vea correctamente en todos los navegadores. PASSED

2. Pruebas en dispositivos móviles:

- Acceder a Wikipedia desde un teléfono móvil y verificar que la versión móvil del sitio se cargue correctamente y sea fácil de navegar. PASSED

4) Registro y Reporte:

1. Documentar errores:

- Error 404 al hacer clic en el enlace "Referencias" en la página de "Historia de la Informática".

2. Preparar el Reporte.

Informe: Wikipedia

Informe de Pruebas de Funcionalidad de Wikipedia - Versión 1.0

Resumen:

✧ Se llevaron a cabo pruebas exhaustivas en la página de Wikipedia para verificar su funcionalidad y usabilidad en diferentes escenarios y dispositivos. A continuación, se presenta un resumen de los resultados obtenidos durante las pruebas.

Pruebas Realizadas:

- *Navegación:* Se probó la navegación en diferentes secciones y categorías del sitio, asegurando que todas las páginas se cargaran correctamente.
- *Búsqueda:* Se realizaron búsquedas con términos específicos para evaluar la precisión y relevancia de los resultados.
- *Edición de Contenido:* Se realizaron ediciones de prueba en páginas existentes para verificar la funcionalidad del sistema de edición.
- *Enlaces Internos:* Se comprobó la validez de los enlaces internos en varias páginas.
- *Dispositivos y Navegadores:* Se realizaron pruebas en *Chrome, Firefox, Safari y Edge*, así como en dispositivos móviles para evaluar lo responsivo del sitio.

Resultados:

- *Navegación:* Todas las secciones y páginas se cargaron sin problemas.
- *Búsqueda:* Las búsquedas fueron precisas y los resultados relevantes.
- *Edición de Contenido:* Las ediciones se guardaron correctamente y se reflejaron en las páginas.
- *Enlaces Internos:* La mayoría de los enlaces funcionaron correctamente, pero se encontró un ***error 404*** al hacer clic en el enlace "Referencias" en la página "Historia de la Informática".
- *Dispositivos y Navegadores:* El sitio fue compatible con todos los navegadores probados y se mostró correctamente en dispositivos móviles.

Bugs Encontrados:

- Error 404 en Enlace "Referencias": Al hacer clic en el enlace "Referencias" en la página "Historia de la Informática", los usuarios son redirigidos a una página de error 404.

Recomendaciones:

- Se recomienda corregir el ***error 404*** en el enlace "Referencias" para mejorar la experiencia del usuario y garantizar el acceso a información relevante en el artículo.

Conclusión:

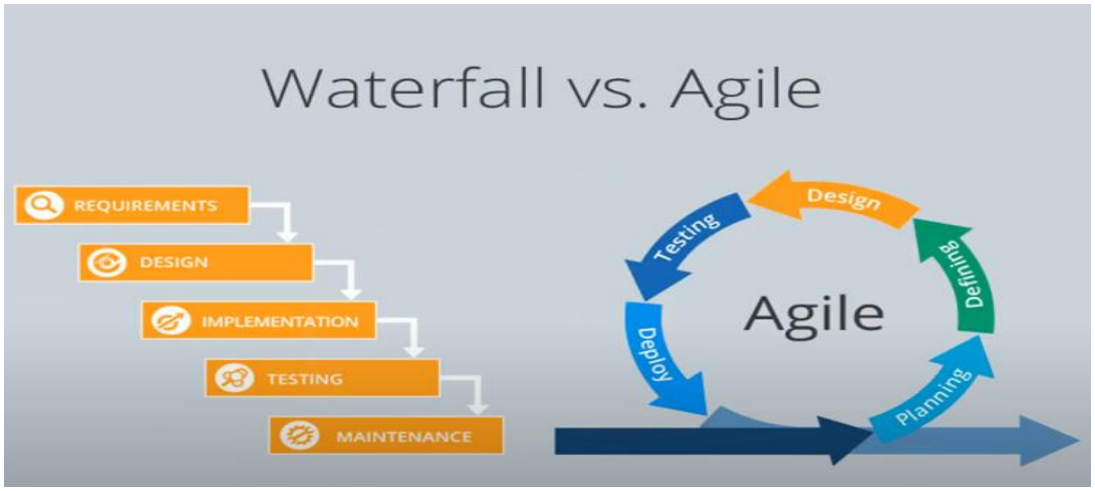
En general, Wikipedia funcionó correctamente en la mayoría de las áreas probadas. Sin embargo, se recomienda abordar el problema identificado para garantizar la calidad y la accesibilidad del contenido para los usuarios.

Waterfall vs Agile

WATERFALL
VENTAJAS: posee una estructura clara, documentación completa.
DESVENTAJAS: poca flexibilidad de ajustar los requisitos después de que el proyecto comenzó, entregas tardías, riesgos de error.
AGILE
metodología en enfoque de desarrollo de software, basado en principios colaborativos, flexibilidad y adaptabilidad.
permite a los equipos responder a los cambios y desarrollar software de manera iterativa e incremental.
EJEMPLO: scrum, camban y xp.
VENTAJAS: permite a los equipos adaptarse fácilmente, fomenta la colaboración y comunicación continua.
entregas iterativas (muestra pequeñas partes funcionales), tiende a satisfacer mejor a los clientes, etc.
DESVENTAJAS: complejidad de gestion, menos predictivo.
RESUMEN:

- ✓ Agile un proyecto puede durar semanas y en Waterfall meses. Ahora en Agile desde el primer momento el cliente puede ver los cambios, mejoras, etc.
- ✓ Waterfall hasta el fin del proceso no puede ver como esta el proyecto.

DEPENDEN DE LA NATURALEZA DEL PROYECTO (EXISTEN HIBRIDOS)



Conceptos básicos y como trabajar en Agile

✧ Agile Testing es una metodología de prueba que se integra con los principios del desarrollo ágil de software.

Prácticas clave para implementar Agile Testing:

1. **Colaboración constante:**
   - ❖ *Equipos multidisciplinarios:* Desarrolladores, probadores y otros miembros del equipo trabajan juntos.
   - ❖ *Comunicación constante:* Se promueve la comunicación abierta y regular entre los miembros del equipo.

2. **Iteraciones y pruebas continuas:**
   - ❖ *Desarrollo iterativo:* El software se desarrolla en incrementos pequeños y funcionales.
   - ❖ *Pruebas continuas:* Las pruebas se automatizan tanto como sea posible para que puedan ejecutarse con cada cambio en el código.

3. **Automatización y TDD (Desarrollo Guiado por Pruebas):**
   - ❖ *Automatización de pruebas:* Las pruebas unitarias, de integración y de aceptación se automatizan para que puedan ejecutarse rápidamente y de forma repetible.
   - ❖ *TDD:* Los casos de prueba se escriben antes del código, lo que garantiza que el código se desarrolla para cumplir con los requisitos específicos.

4. **Pruebas exploratorias:**
   - ❖ *Exploratory Testing:* Los probadores utilizan su experiencia y conocimiento para explorar el software y encontrar defectos que podrían no estar cubiertos por las pruebas automatizadas.

5. **Retroalimentación continua y mejora:**
   - ❖ *Revisiones regulares:* El equipo revisa regularmente el progreso y ajusta las estrategias de prueba según sea necesario.
   - ❖ *Mejora continua:* Se alienta a los equipos a aprender de las experiencias pasadas y a mejorar constantemente los procesos y las prácticas de prueba.

6. **Pruebas de aceptación del cliente:**
   - ❖ *Participación del cliente:* Los clientes y usuarios finales participan activamente en las demostraciones y las pruebas de aceptación para garantizar que el software cumple con sus expectativas y necesidades.

7. **Gestión de defectos eficiente:**
   - ❖ *Registro y seguimiento:* Los defectos se registran detalladamente y se les da seguimiento hasta su resolución.
   - ❖ *Priorización:* Los defectos se priorizan en función de su impacto en el usuario y se solucionan de manera oportuna.

Scrum

- ✧ Es un marco de trabajo Agile. (conjunto estandarizado de conceptos, prácticas y criterios).

Roles en Scrum:
1. Scrum máster: Es un facilitador del equipo.
2. Product Owner: Representa las necesidades del cliente.
3. Equipo de desarrollo: Profesionales que desarrollan el producto + QAs.

Sprint 1: Es un periodo de tiempo de 1 semana a 1 mes, entrega software funcional de calidad.
2 Semanas -> Modulo de suma. (30 tareas)
Sprint 2: 2 Semanas -> Modulo de resta. (20 tareas)

Eventos de SCRUM:
1) Sprint planning: Es una réunion al comienzo donde se definen que tareas se van a llevar.
2) Daily Scrum: Reunión donde se informa como va el trabajo. 15 Min.
3) Sprint Review: Reunión final del Sprint se muestran las funcionalidades y se recibe Retroalimentación.
4) Sprint Retrospective: Reunion al final del sprint.

Como trabajar con scrum en el testing:
A. Participación en el Sprint planning.
B. Participar del Daily Scrum: Solo en el equipo de QA.
C. Colaboración continua.
D. Automatizacion.
E. participar del Sprint Review.
F. Participar en el retrospectiva.

Automatización de Pruebas con Scrum

Se utiliza para:
1. Mejorar la calidad del software.
2. Liberar a los tester para tareas más estratégicas.
3. Mejorar la eficiencia del proceso de desarrollo.

Para automatizar pruebas en un equipo de Scrum:
1) Planificar la automatización
2) Desarrollar la automatización
3) Ejecutar la automatización
4) Mantener la automatización

Consejos para automatizar pruebas en un equipo de Scrum:
A. Comienze con pruebas simples.
B. Trabajar en colaboración.
C. Utilice herramientas adecuadas.

Automatización Testing

✧ Ejecutar casos de prueba automáticamente con una herramienta de automatizacion.

Ventajas del Automation Testing:

1. Eficiencia.
2. Repetibilidad.
3. Re utilización.
4. Cobertura.
5. Ahorro de Costos.
6. Detección Temprana de Defectos.

Desventajas de Automation testing:

1) Inicialmente Tiempo Consumidor.
2) Pruebas Complejas.
3) Costo de Mantenimiento.
4) Pruebas de Interfaz de Usuario.
5) No Puede Reemplazar Completamente las Pruebas Manuales.

Automatización Back-end

✧ La parte que no es visible para los usuarios.

El back-end incluye varios componentes, como:

1. Servidor.
2. Base de datos.
3. Lógica de la aplicación.
4. Seguridad.
5. Integraciones.

Que es BackEnd Automation:

1) Gestión de Base de Datos
2) Procesamiento de Datos
3) Integración de Sistemas
4) Gestión de Usuarios y Accesos
5) Procesos de Negocio
6) Gestión de Tareas y Flujos de Trabajo