Clase 9

“Conceptos Básicos del Testing”

¿Qué vamos a ver hoy?

Vamos a hacer un recorrido sobre los conceptos principales del universo del testing, cuál es la diferencia entre un testeo funcional y un testeo no funcional y en dónde ubicamos el testeo de accesibilidad.

Luego en la segunda parte de la clase de hoy vamos a retomar las herramientas de la clase pasada para testeo manual de contraste y zoom, para ubicarnos según las necesidades particulares de uso.

¿Qué es el Testing?

El testing de accesibilidad web o digital, al igual que las pruebas de software, son los métodos para validar que un producto cumple determinados requerimientos y niveles de conformidad.

Es una suma de metodologías, herramientas y procesos para probar y encontrar defectos en un sistema o similar. El tester es el profesional que los lleva adelante.

¿Qué es Probar?

Probar es una forma de evaluar la calidad de algo y de reducir el riesgo de fallos en un entorno de operaciones o en producción.

¿Qué es la Calidad?

Es el grado o nivel de operatividad en la que un sistema, producto, aplicación, etc… satisface las necesidad pedidas y cumple ciertas expectativas de los usuarios finales.

Ejemplos de experiencias como usuario en el cual haya funcionado mal la plataforma / página web.

¿Por qué probamos?

- Para detectar errores lo antes posible.
- Verificar que el producto funcione acorde a lo requerido
- Para generar confianza en el usuario final

Tester + Probar = Calidad

La actividad de probar, realizada por el tester, es lo que nos genera calidad en nuestro producto final… a grandes rasgos.

Diferencia Entre Error, Bug, y Falla

Es muy frecuente entre los profesionistas en tecnología utilizar los términos error, bug, y falla para describir un resultado inesperado durante la ejecución de un programa o para referirse a la causa que genera dicho resultado.

Sin embargo, ***error, bug, y falla no son lo mismo.*** Comprender las diferencias entre estos conceptos y usarlos adecuadamente simplifica y estandariza la comunicación.

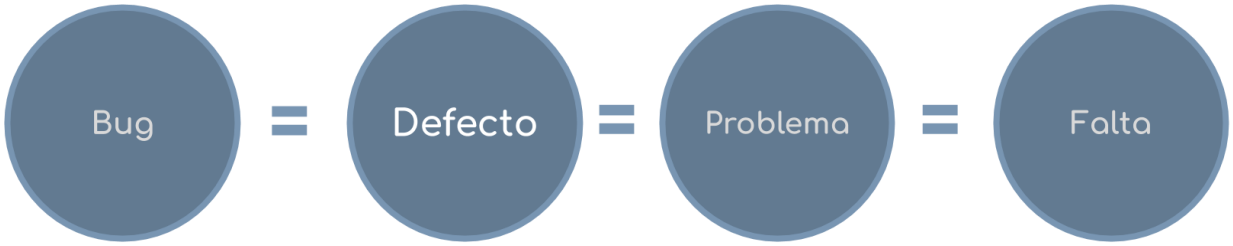
Conceptos

El glosario de la ***Junta Internacional de Aptitudes de Pruebas de Software*** (ISTQB, por sus siglas en inglés) define los términos de la siguiente manera:

- ***Error:*** Acción humana que produce un resultado incorrecto.
- ***Bug:*** Imperfección en un componente o sistema que puede causar que el componente o sistema falle en desempeñar las funciones requeridas.

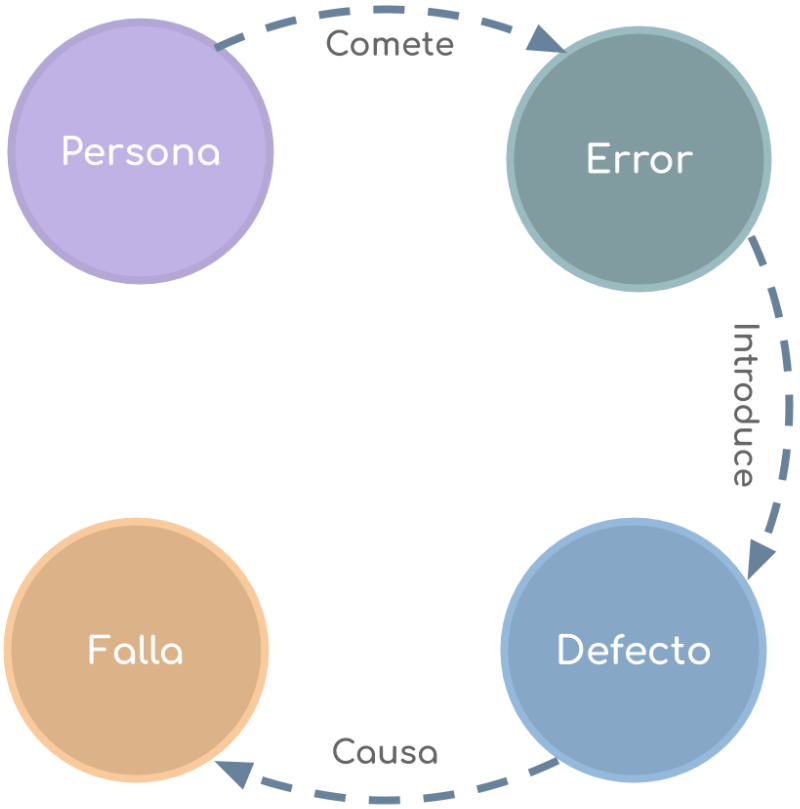
- ***Falla:*** Desviación del componente o del sistema respecto de prestación, servicio o resultado esperado.

La **ISTQB** señala que las palabras ***defecto, problema y falta son sinónimos de bug*** y que es ***preferible el uso de la palabra defecto*** sobre los otros términos a pesar de que el término bug es más utilizado.



El siguiente enunciado es de utilidad para entender, tanto la diferencia como la relación, que existe entre los términos error, defecto y falla.

Una persona comete un error e ***introduce un defecto*** en el código del programa. Si durante la ejecución del programa, se ejecuta la línea de código con el defecto, el programa funciona de manera inesperada, es decir, ***el defecto causa una falla.***



Ejemplo

Un programador tiene la tarea de implementar un programa que controle el acceso a un sitio web permitiendo el acceso sólo a usuarios de 18 o más años de edad.

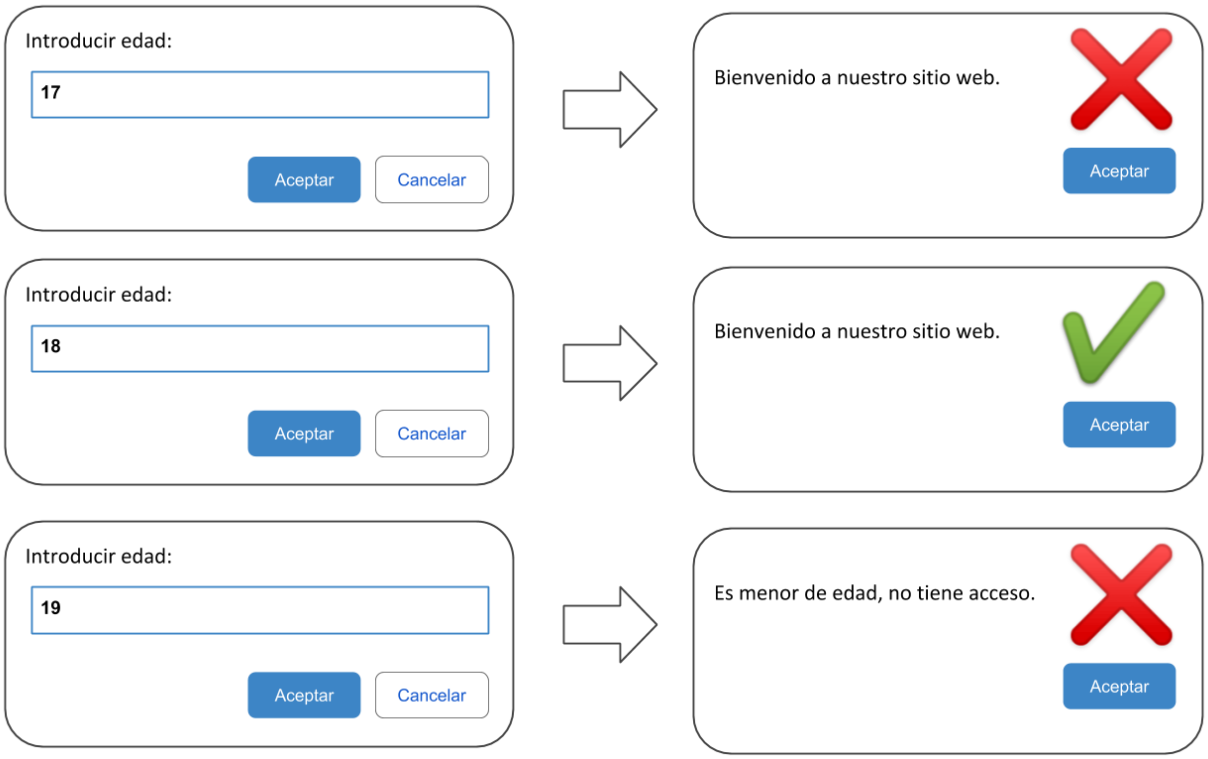
El programa solicita al usuario introducir su edad y si ésta es de 18 o más, le da la bienvenida al sitio web, si no, se le niega el acceso por ser menor de edad.

El programador implementa el software y decide probarlo utilizando los siguientes casos de prueba:

Edad y resultado esperado

- 17 = Es menor de edad, no tiene acceso
- 18 = Bienvenido a nuestro sitio web
- 19 = Bienvenido a nuestro sitio web

Los resultados de sus pruebas se muestran a continuación:

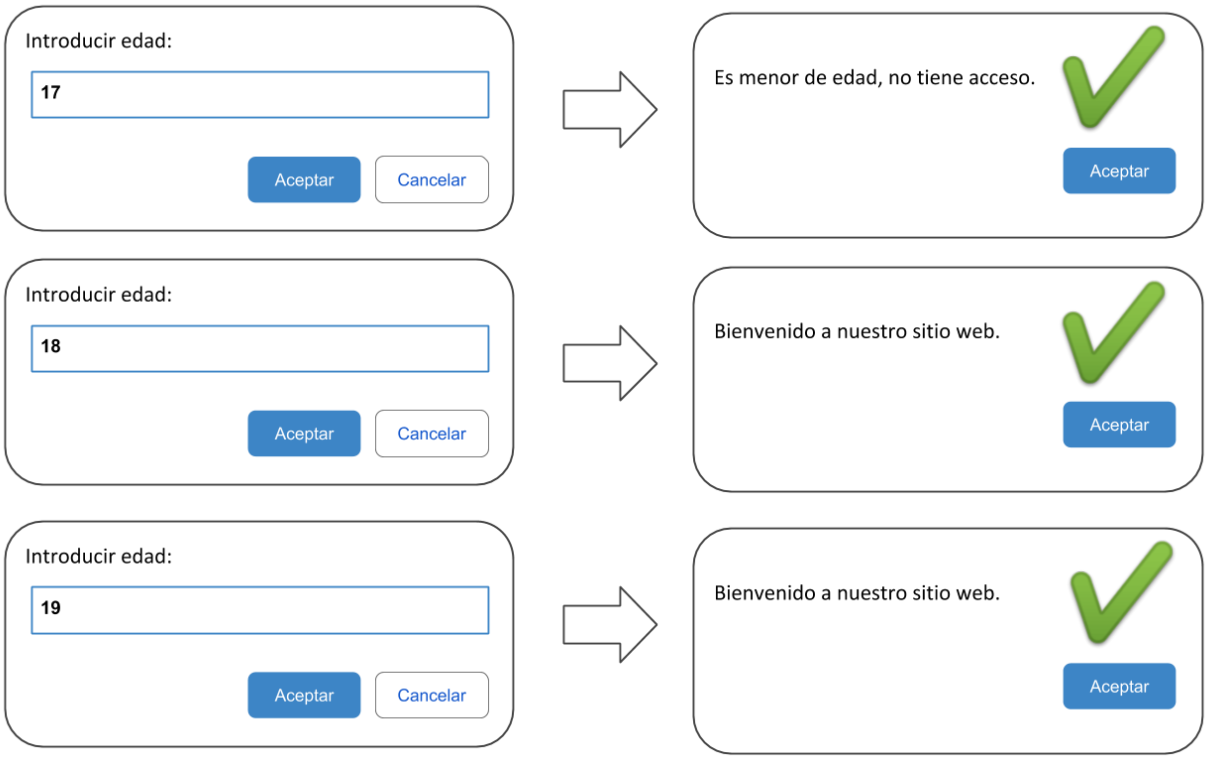


Como se observa en los resultados, el programa autoriza correctamente el acceso al sitio web, cuando se introduce la edad de 18 años. Sin embargo ***falla*** en los otros dos casos.

En el caso en el que la edad es de 17 años el programa debió de negar el acceso al sitio pero el acceso fue autorizado y en el caso en el que la edad es de 19 años el programa debió permitir el acceso pero lo negó.

El programador inspecciona el código y detecta un ***defecto*** en la línea 3. El programador introdujo por ***error*** el defecto en la condición.

Una vez que el programador identificó y corrigió el ***defecto*** que causó la ***falla***, el programador prueba nuevamente el programa utilizando los mismos casos de prueba y verifica que funciona adecuadamente.



Cabe destacar que los defectos no son la única fuente de fallas en el software, otros factores como cambios inesperados en librerías o sistemas de los que se depende también resultan en fallas.

Entendimos que…

Los términos ***error, defecto, y falla no son lo mismo*** y utilizarlos adecuadamente nos ayuda a tener una ***comunicación eficaz.***

Cuales son las tareas del tester

- Confección de planes de prueba.
- Entender los requerimientos o requisitos según normas.
- Diseñar casos de prueba.
- Establecer los ambientes de pruebas necesarios.
- Realizar los casos de pruebas.
- Informes de pruebas.
- Reportes de defectos.

Cualidades de un Tester

- ***Curiosidad:*** Más allá del conocimiento a la hora de probar, un tester debe sentir curiosidad por el producto para poder imaginar mejores planes de prueba.
- ***Observación:*** El tester está siempre en los detalles que el resto del equipo no observa.

- ***Pensamiento lateral o razonamiento creativo:*** Viene junto con la creatividad para intentar resolver problemas buscando vías alternativas, siempre siendo imaginativo en los resultados.
- ***Comunicación:*** Dado que el tester es el informador de reportes o defectos, es importante tener un buen nivel de comunicación tanto oral como escrito para poder transmitir la información.
- ***Empatía:*** Detrás de cada desarrollo, hay , al menos, una persona que trabajó en eso. Es importante tener empatía al momento de entregar información para cuidar las buenas relaciones con los pares, teniendo en cuenta, que todos tenemos el mismo objetivo: la calidad.

Quality Assurance y Quality Control

DEVs vs QAs

Es bien conocido la riña que existe entre ambas profesiones. Esto sucede debido al tipo de análisis que realiza cada rol.

¿Cuál es la diferencia?

Los desarrolladores se interesan porque el producto funcione, mientras que los testers buscan que funcione bien y acorde a lo que se documenta.

¿Qué tan técnico es un tester?

Tanto como el profesional desee… Existen diversas áreas dentro del QA, y cada una requiere (o no) ciertos conocimientos técnicos.
No es necesario saber programar para ser tester QA.

Seniority en QA

Al QA se lo denomina Analista de Calidad, y acorde al desarrollo profesional y las nuevas habilidades que vamos consiguiendo con el tiempo, se irá estableciendo nuestro seniority.

- ***QA Junior:*** podrá hacer documentacion y realizar pruebas como algo cotidiano
- ***QA SemiSenior:*** sabrá manejar muchas herramientas para realizar distintos tipos de prueba.
- ***QA Senior:*** podrá abrir el código del desarrollador para corregirlo.

Diferencias entre Quality Assurance y Quality Control

A diario confundido por cualquier entidad fuera del área de calidad, pero ambos puestos son tan distintos como similares.

Quality Assurance QA

Llamado Aseguramiento de Calidad.

Es el encargado de llevar adelante todo tipo de pruebas que encuentren errores en etapas tempranas del desarrollo, asegurando la calidad antes de la salida del producto y velando porque se cumplan todos los requerimientos del sistema a entregar.

Quality Control QC

Llamado Control de Calidad.
Busca asegurar la calidad del servicio o producto a base de las especificaciones pautadas por la organización donde este se desenvuelve. Se incluye recolección y análisis de datos del servicio para intentar resolver, con distintas áreas del negocio, los problemas en el producto.

Áreas de Calidad

Aunque explicamos algunos tipos de profesiones de calidad, es importante comprender que muchas empresas poseen áreas gigantescas de calidad, donde hay todavía más puestos.
Un área de calidad puede llegar a contar con testers dedicados a tareas como la creación de casos de prueba, de generar pruebas de retroceso, métricas, automatización, seguridad, disponibilidad de herramientas, etc…

Origen del Rol

El tester QA posee, según expertos, el rol más importante dentro del ciclo de desarrollo, ya que es la persona que va a aprobar que un desarrollo se deploye y sea utilizado por los clientes.

Si el QA hace mal su trabajo, podría generar pérdidas muy importantes para el negocio, desde monetaria, hasta clientes.

Principios del testing

Las máximas del Testing

Hace muchos años se han establecido 7 verdades absolutas dentro del universo del testing, que siempre se regirán, sin importar el proyecto en el que estés.

Estas máximas son:

1. Las pruebas muestran la presencia de defectos, no su ausencia.
2. Las pruebas exhaustivas son imposibles.
3. Las pruebas tempranas ahorran tiempo y dinero.
4. Los defectos se agrupan.
5. Cuidado con la paradoja del pesticida.
6. Las pruebas dependen del contexto.
7. La ausencia de errores es una falacia.

1. La prueba muestra la presencia de defectos, no su ausencia

El testing nos permite demostrar la existencia de incidentes en un producto, no su ausencia.

Tras el reporte de un incidente y su consecuente restauración, es posible reducir la probabilidad de que incidentes aún no descubiertos persistan en el sistema, pero la inexistencia absoluta de incidentes no sólo resulta imposible de comprobarse sino que además es una posibilidad sumamente improbable.

Ejemplo Primer Principio:

El Samsung Galaxy Note 7 tenía una falla y hacía que se prendiera fuego. Tuvo que ser retirado del mercado.
Siempre puede haber una falla.

2. La prueba exhaustiva es imposible

No es posible probar todo, excepto en casos triviales. En lugar de intentar realizar pruebas exhaustivas se deberían utilizar el análisis de riesgos, las técnicas de prueba y las prioridades para centrar los esfuerzos de prueba.

Ejemplo Segundo Principio

Aplicación de temperatura global. Deberían probar los 195 países, por cada provincia y por cada ciudad… ¿Es un poco alta esa cantidad de casuísticas no?

3. La prueba temprana ahorra tiempo y dinero

Para detectar defectos de forma temprana, las actividades de prueba, tanto estáticas como dinámicas, deben iniciarse lo antes posible en el ciclo de vida de desarrollo de software.

La prueba temprana en el ciclo de vida de desarrollo de software ayuda a reducir o eliminar cambios costosos.

Ejemplo Tercer Principio

Según un estudio realizado por IBM, el costo de reparación de un defecto en su etapa de post-producción es de hasta 30 veces más costoso, que en la etapa de diseño, y 3 veces más que en la etapa de testing.

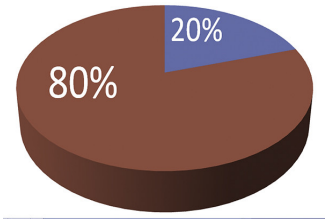
Design and architecture	Implementation	Integration testing	Customer beta test	Postproduct release
1X*	5X	10X	15X	30X

4. Los defectos se agrupan

En general, un pequeño número de módulos contiene la mayoría de los defectos descubiertos durante la prueba previa al lanzamiento, o es responsable de la mayoría de los fallos operativos.

Las agrupaciones de defectos observadas en la prueba o producción son una aportación importante a un análisis de riesgos utilizado para centrar el esfuerzo de la prueba.

Ejemplo Cuarto Principio



El 80% de los defectos se encuentra en el 20% de los módulos, ya que son el puntapié para desencadenar otros defectos en el resto del producto.

5. Cuidado con la paradoja del pesticida

Si las mismas pruebas se repiten una y otra vez, eventualmente estas pruebas ya no encontrarán ningún defecto nuevo.

Para detectar nuevos defectos, es posible que sea necesario cambiar las pruebas y los datos de prueba existentes, y es posible que sea necesario redactar nuevas pruebas.

Las pruebas ya no son efectivas para detectar defectos, de la misma manera que los pesticidas ya no son efectivos para matar insectos después de un tiempo

Ejemplo Quinto Principio

Si utilizamos los mismos métodos una y otra vez, estos dejan de surtir efecto, de la misma forma que el pesticida deja de surtir efectos en las plagas.

6. Las pruebas depende del contexto

Las pruebas se realizan de distintas maneras en diferentes contextos. Comprendiendo que el contexto va a indicar herramientas que pueden utilizarse para probar, ambiente o tipo de objeto que se prueba, metodología implementada por la empresa, o cualquier otro contexto que pueda existir.

Ejemplo Sexto Principio

Tenemos 3 tostadoras en la imagen. ¿Creen que van a tener que hacer las mismas pruebas para las tres? ¿O cada tostadora tendrá su propio set de pruebas distinta a las otras?

7. La ausencia de errores es una falacia

Supongamos que todos los incidentes detectados en un sistema dado son corregidos. A continuación, se realiza un nuevo ciclo de pruebas, tras el cual no se detecta la presencia de nuevos incidentes.

La ausencia de nuevos errores detectados no implica necesariamente que el software sea útil, ya que la utilidad del mismo no es indicada por este parámetro, sino por la satisfacción de las expectativas del cliente que el producto pueda brindar.

¿Cuáles son las responsabilidades de un tester manual?

Responsabilidad en Calidad

En la clase pasada se explicó qué significa la calidad, y quizá hacemos mucho hincapié en ello, pero es el punto más importante del QA, y por ello hacemos todas las pruebas necesarias para alcanzar el máximo nivel de calidad.

Nuestra responsabilidad como aseguradores de la calidad, es que el producto sea lo más confiable para el cliente.

Máximo nivel de Calidad

Al encontrar errores y reportarlos al área de Desarrollo, estos volverán corregidos por lo que se probará nuevamente y en cada ciclo de testing el producto va a ir mejorando y tendrá mayor calidad.

No es nada malo que existan errores en el software, siempre y cuando el tester los encuentre.

¿Cómo probamos?

- La actividad de realizar pruebas se divide en dos etapas fundamentales. Las pruebas dinámicas y las pruebas estáticas.
- Las pruebas dinámicas es la propia acción de probar un sistema.
- Las pruebas estáticas son la acción de leer documentación.

¿Qué es la Documentación?

Es toda la información que nos va a respaldar y es muy importante que exista en un Proyecto.

Existen muchos tipos de documentación, aca detallaremos los más importantes:

- Historias de usuario.
- Plan de pruebas.
- Manuales.
- Criterios de aceptación.

Importancia de la Documentación

El trabajo del tester es bastante invisible, ya que no se puede palpar en sí mismo, para eso entra en juego la documentación.

Es nuestro respaldo como profesionales para indicar que pudimos corroborar, que cosas se probaron, en base a qué estándares, quien los pidió, y cualquier otro tipo de afirmación escrita.

Herramientas para documentar

En el curso vamos a utilizar como herramienta para plasmar nuestra documentación:

Casos de prueba:

- Excel

Capturas de pantalla:

Apex Screen Recorder & Screenshot Master

¿Falta documentación?

La falta de documentación, es una realidad más común de la que creemos, y es una de nuestras responsabilidades como tal hacer pedido de ella en caso de que no exista.

Ciertos tipos de pruebas, llamadas pruebas exploratorias, hacen uso nulo de documentaciones, pero son solo una primera instancia de trabajo. El uso de la documentación es algo sumamente prioritario.

Métricas

Las métricas son otra de nuestras herramientas de trabajo, ya que es lo que usamos para medir y presentar la cantidad de trabajo realizado.
Se realizan métricas en base a casos realizados, bugs encontrados, tiempo de trabajo implementado, etc...
Es un estándar de medida que nos permite tomar decisiones.

Ejemplos de métricas en el Testing:

- Cantidad de errores encontrados con severidad alta
- Cantidad de casos de prueba que fallaron
- Cantidad de errores por tipo y criticidad
- Cantidad de ciclos de testing

Adaptabilidad

Una cualidad del tester, y responsabilidad, es comprender que tenemos que adaptarnos a las circunstancias que nos van surgiendo. Ya sea ambientes de prueba, falta de documentación, cambios de último momento, etc… La

adaptabilidad es una herramienta que se construye para poder adaptar a lo que tengamos que probar y bajo las condiciones que surjan.

Tipos de testing

Tipos de prueba

Es un grupo de actividades de prueba destinadas a probar características específicas de un sistema de software, o de una parte de un sistema, basadas en objetivos de prueba específicos.

Tipos de pruebas:

- Funcional
- No funcional
- Caja blanca
- Asociada al cambio

4 Objetivos de los tipos de pruebas

- Evaluar la completitud, corrección y pertinencia - Funcional.
- Evaluar la viabilidad, eficiencia de desempeño, seguridad, compatibilidad y usabilidad - No Funcional.
- Evaluar si la estructura del sistema es correcta, completa y según lo especificado - Caja Blanca.
- Evaluar los efectos de los cambios, tales como confirmar que los defectos han sido corregidos y buscar cambios no deseados en el comportamiento que resultan de cambios en el software o en el entorno - Asociada al Cambio.

Prueba funcional

Definición

La prueba funcional de un sistema incluye pruebas que evalúan las funciones que el sistema debe realizar. Los requisitos funcionales pueden estar descritos en productos de trabajo tales como especificaciones de requisitos de negocio, épicas, historias de usuario, casos de uso, o especificaciones funcionales.

Las funciones describen "qué" debe hacer el sistema.

¿Cuándo se hacen las pruebas funcionales?

Se deben realizar pruebas funcionales en todos los niveles de prueba, aunque el enfoque es diferente en cada nivel.
Intensidad pruebas funcionales.

Se puede medir la intensidad de la prueba funcional a través de la cobertura funcional. La cobertura funcional es la medida en que algún tipo de elemento funcional ha sido practicado por pruebas, y se expresa como un porcentaje del tipo o tipos de elementos cubiertos.

Alcance pruebas funcionales

El diseño y la ejecución de pruebas funcionales pueden implicar competencias o conocimientos especiales, como el conocimiento del

problema de negocio específico que resuelve el software o el papel particular que desempeña el software.

Prueba No Funcional

Definición

Las pruebas no-funcionales evalúan otras características de sistemas, como por ejemplo el UX/UI, conectividad o la seguridad. La prueba no-funcional prueba que tan bien se comporta un sistema básicamente. Hacen foco en los Requerimientos No Funcionales.

Requerimientos no funcionales

Son los requerimientos que especifican propiedades del sistema tales como:

- Rendimiento
- Seguridad
- Disponibilidad
- Performance
- Accesibilidad
- Estabilidad, etc.

Ejemplos

- El sistema estará restringido bajo contraseñas cifradas y usuarios definidos. -Seguridad

- El sistema deberá tener interfaces intuitivas y deberan respetar los colores azul y naranja.-Usabilidad
- El sistema no debe tardar más de 10 segundos en mostrar resultados de una búsqueda.-Performance

¿Cuándo se hacen pruebas no funcionales?

Se sugiere que las pruebas no-funcionales se realicen en cada nivel de prueba ya que pueden prever problemas en los sistemas que van por fuera de los errores encontrados en pruebas funcionales. Encontrar problemas del tipo no-funcional sobre el final de un desarrollo, puede jugar un papel crucial en la implementación del mismo.

Intensidad pruebas no funcionales

Tal como en las pruebas funcionales, acá también se puede medir la intensidad de una prueba no funcional a través de la cobertura no funcional. También se mide de acuerdo a la medida de cómo algún objeto no funcional ha sido probado.

Por ejemplo, se puede hacer una trazabilidad entre las pruebas y los sistemas operativos compatibles con una aplicación, se puede calcular la cantidad de sistemas operativos trabajados en las pruebas de compatibilidad, identificando el porcentaje de carencia de pruebas.
Alcance pruebas no funcionales

Para este tipo de pruebas, se debe contar con ciertos conocimientos o competencias especiales al momento de hacer el diseño y la ejecución de las mismas.

Prueba de caja blanca

Definición

Se obtienen pruebas leyendo la infraestructura del sistema, el código del mismo, en su implementación o arquitectura. También incluye flujos de datos, bases de datos o flujos de trabajo. Esta es la razón por la que es un tipo de prueba que necesita un seniority más avanzado.

Intensidad pruebas de caja blanca

Este tipo de pruebas también lleva una medición de intensidad a través de la cobertura estructural. En esta, la medida se toma a partir de los objetos estructurales que se hayan realizado contra lo que resta probar. Dependiendo qué objeto sea el que se esté probando es el tipo de porcentaje que se aplicará.

Conocimiento de código

Claramente las pruebas del tipo de caja blanca necesitan de un tipo de conocimiento relacionado a otras áreas que van por fuera del tester QA. Su diseño y ejecución va a estar basado de las competencias que se tenga sobre el tema.

Más allá de los conocimientos, se recomienda contar con el soporte de un experto del área durante estas pruebas.

Prueba asociada al cambio

Definición

Como su nombre lo indica, son pruebas que se utilizan para comprobar corrección de bugs o mejoras en un sistema o nuevas funcionalidades, para poder asegurar que el sistema funciona como corresponde.
Aquí es donde entra en juego el papel del QA de Automatización.

Áreas de las pruebas

Las vamos a separar en dos grandes áreas:

- Pruebas de confirmación.
- Pruebas de regresión.

Pruebas de confirmación

Cuando nos entregan el sistema corregido de algún defecto, se llevan adelante estas pruebas para asegurar que el error ya no se encuentra. Esto implica realizar los pasos que llevaron adelante ese error, y continuar con los próximos pasos, si es que este error bloqueo alguno o probar si otras

áreas del sistema, relacionadas con el bug, siguen funcionando de forma correcta.

Pruebas de regresión

Como su nombre lo indica, se realizará una regresión sobre lo que ya se probó. Esta prueba se utiliza para corroborar que ninguna otra área del sistema haya sido afectada por el cambio realizado en él. Esto es para ver que no haya ningún defecto como daño secundario a causa de estos cambios.

¿Cuándo se aplican estas pruebas?

Estas dos pruebas se realizan en todos los niveles de prueba, especialmente en metodologías del tipo Agile, debido a la gran cantidad de iteraciones incrementales en sus ciclos de desarrollo.
Aplicación de estas pruebas

Como se deben imaginar, acá es donde la labor de los QA de automatización, o como algunos los llaman, los QAA, tienen un papel importantísimo. Sin las pruebas de regresión automatizadas, el tiempo de estas pruebas pasaría por encima de los tiempos de las demás. Generando que la dependencia del QA sea entera en volver a probar funcionalidades para ver que no pierdan compatibilidad.

Material para profundizar

Si te interesó algunos de los temas de hoy, te dejo material alternativo para que profundices al respecto, te recomiendo ver los videos que son cortos e interesantes! *No es material obligatorio*

Ted talk sobre testing y su importancia.

Glosario

Testing: Conjunto de procesos que se utilizan para encontrar defectos en el software.
QA: Aseguramiento de Calidad
QC: Control de Calidad
Calidad: Nivel de operatividad en la que un software satisface las necesidades pedidas de los usuarios finales.
Quality Management: Gerencia de Calidad
Quality Planning: Calidad de Planeamiento
Quality Improvement: Calidad de Implementación.
Bug: Defecto en un software que produce un resultado indeseado.