UNIDAD 1

CONCEPTOS BASICOS DE TESTING

1.1 INTRODUCCIÓN

El Testing o pruebas de software va relacionado con lo que es la calidad de software, dentro de este contexto vamos a repasar algunos conceptos para comprender porque es necesario realizar el Testing.

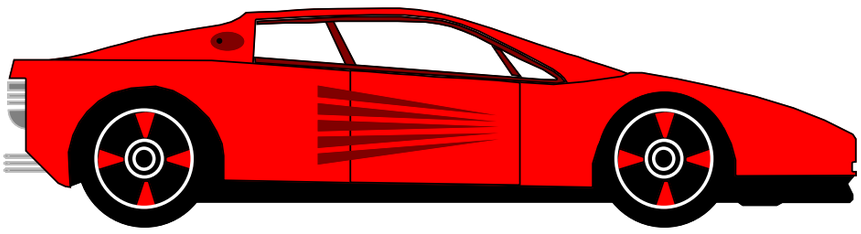
"No me preocupa si algo es barato o caro. Solo me preocupa si es bueno. Si es lo suficientemente bueno, el público te lo pagará".

Que tiene más calidad

Los dos tienen la misma calidad siempre y cuando cumplan con sus requerimientos. Para ello debemos probar sus especificaciones

La calidad es algo subjetivo.

Si nos ponemos a definir sobre lo que es calidad, puede que sea algo subjetivo porque para una persona puede ser que un producto sea de más o menos calidad que para otra, sin embargo, para poder tener un concepto global de lo que es calidad es necesario apoyarse en estandandares.



La calidad es relativa a las personas, a su edad, a las circunstancias de trabajo, el tiempo…

- Un caramelo para un niño.
- Un mapa gastronómico mundial.
- El tiempo varia las percepciones.

1.2 CONCEPTO DE CALIDAD - DEFINICIONES

- ✓ Propiedad o conjunto de propiedades inherentes a una cosa, que permiten apreciarla cono igual, mejor o peor que las restantes de su especie (DRAE).
- ✓ Totalidad de las características de un producto o servicio que le confieren su aptitud para satisfacer unas necesidades expresadas o implícitas (Norma UNE 66-001-92 traducción de ISO 8402).

1.3 CALIDAD DE SOFTWARE

La **calidad** del **software** es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La **calidad** es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad.

Calidad del Software: la suma de todos los atributos que se refieren a la capacidad del software de satisfacer los requerimientos dados.

Cuando hablamos de calidad de software tenemos que diferenciar 2 puntos:

Calidad del producto

El punto principal es ver cómo está construido el software por dentro, analizar su código, sus componentes y como interaccionan con otros. Para que el software sea de calidad si bien es cierto se debe capturar los requerimientos funcionales y no funcionales es importante determinar cuáles son los atributos de calidad, que son aquellos puntos que para el usuario o para el cliente que va a usar la aplicación son importantes ya que de nada sirve que una aplicación cumpla con todos los requerimientos funcionales y no funcionales si al final el cliente para lo que él percibe o la necesidad que él está esperando no la cumple.

Se debe aplicar diferentes estándares para ver si cumple con las características y subcaracteristicas. Por ej. La ISO 25000 que viene de la ISO 9126, pero más específicamente para lo que es calidad de software la ISO 25010.

ISO 25010, existe un conjunto de características y subcaracteristicas el cual esta norma aconseja que un software debe tener. No es necesario que el software tenga todos, sino definir un criterio de cual es más importante de acuerdo al contexto de la aplicación.

- El modelo de calidad representa la piedra angular en torno a la cual se establece el sistema para la evaluación de la calidad del producto. En este modelo se determinan las características de calidad que se van a tener en cuenta a la hora de evaluar las propiedades de un producto software determinado

Con esta guía nos podemos apoyar para ver cuáles son estas características y subcaracteristicas que para el usuario o interesado está esperando, por tanto, hay una alta probabilidad que nuestro software sea aceptado, hay muchas técnicas para evaluar el código (técnicas de caja blanca, caja negra, etc.)

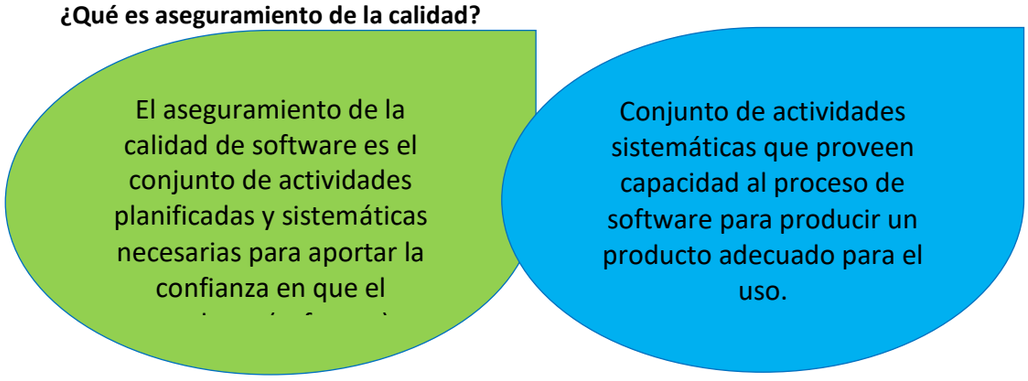
Calidad del Proceso (de software)

Se conoce de términos como CMMI/MOPROSOFT/ ISO 15504 ISO 12207. Estamos haciendo énfasis en los procesos que intervinieron para la creación de este producto, no vamos a analizar ni el código ni el producto, se va a analizar los procesos que una empresa utiliza para llegar al producto final.

- **MOPROSOFT,** origen mexicano, establece los principios a implementar en una organización a través del modelo del proceso de software.
- **CMMI,** modelo de madurez integrado, también define un conjunto de procesos que una organización debe presentar para poder crear un software de calidad. CMMI1 – CMMI5.

1.4 ASEGURAMIENTO DE LA CALIDAD vs CONTROL DE CALIDAD

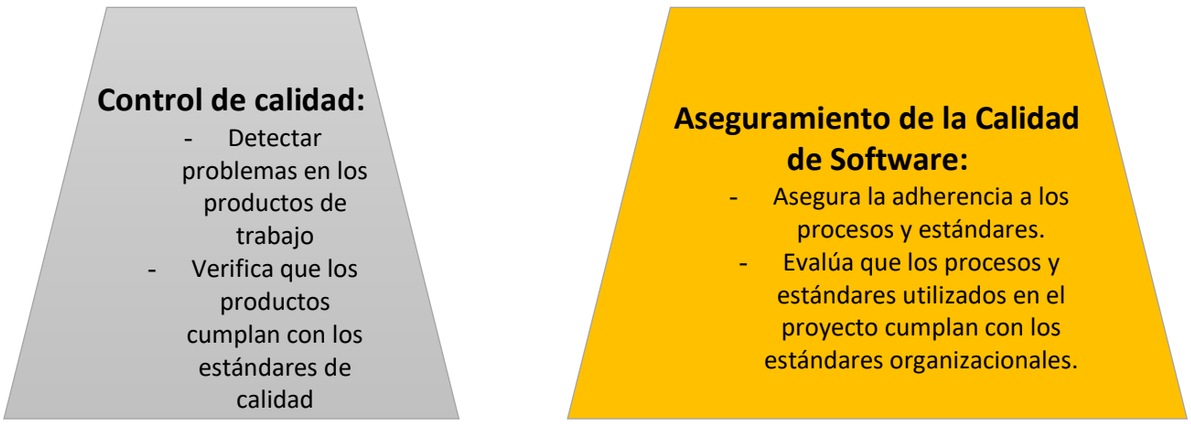
¿Qué es aseguramiento de la calidad?



¿Qué es el control de calidad de software?

Es la estructura que organiza evaluaciones, inspecciones, auditorias y revisiones que aseguren que se cumplan las responsabilidades asignadas, se utilicen eficientemente los recursos y se logre el cumplimiento de los objetivos del producto

Las definiciones de Control de Calidad y Aseguramiento de Calidad se parecen mucho, pues ambas hablan de técnicas y actividades para garantizar la calidad del producto.



En resumen: El Control de Calidad de Software verifica el cumplimiento de estándares definidos para el producto, y el Aseguramiento de la Calidad de Software asegura que el producto haya sido construido siguiendo los procesos establecidos para la organización.

Generalmente cuando le preguntamos a un ingeniero de sistemas sobre lo que Sofware Quality assurance inmediatamente piensa en testing)validación, verificación, revisiones) lo cual son solo extensiones del testing. Sin embargo en la practica vienen a ser casi lo mismo quizá con la diferencia que el SQA tiene un contexto mas global y el testing es algo mas especifico.

1.5 EL EQUIPO DE DESARROLLO

El desarrollo de software no es una actividad simple y por eso se hacen equipos, por ello lo primero que tenemos que empezar a definir es ¿Que es un equipo?

Un equipo está formado por dos o más personas que trabajan juntas porque necesitan lograr un objetivo que tienen en común, y cada una de ellas se le asignan ciertas funciones que son específicas para lograr ese objetivo.

Tenemos que si nuestro equipo es pequeño puede que una sola persona cubra múltiples roles, pero en equipos más grandes es muy común tener funciones muy diferenciadas y dedicadas.

Los roles se van a asignar de acuerdo a la experiencia y las capacidades de las personas y esta identificación de los roles nos va a ayudar a estructurar el proyecto, tenemos varios roles dentro de un equipo y se los puede llamar de igual o diferente manera de los que veremos a continuación, pero normalmente estos son los básicos que encontramos en todo equipo:

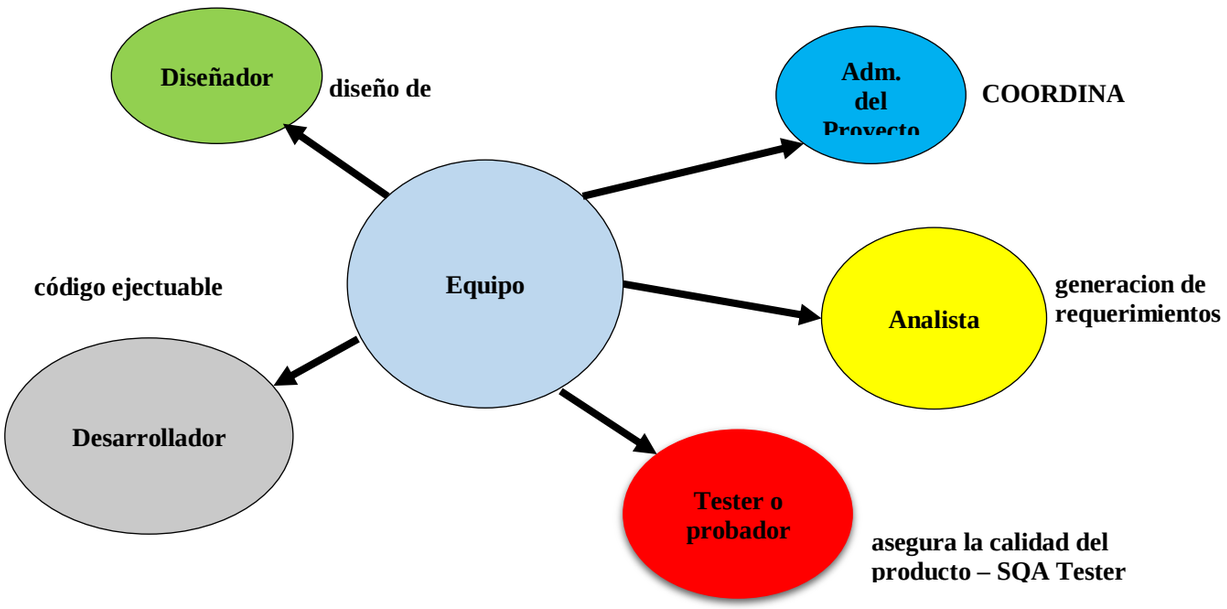
-**Project Manager** (líder de proyecto o administrador de proyecto), es quien coordina el equipo y asegura que todos los demás roles cumplan con su trabajo. Una de las preocupaciones principales que tienen estos administradores debe ser que tienen que tener una visión y misión bien clara del proyecto para que ambas se cumplan.

-**Analista Funcional**, luego de entrevistar al cliente es capaz de generar una serie de requerimientos de sistema para poder definir la estructura básica de este sistema.

-**Diseñador**, es el que toma los requerimientos que obtuvo el analista funcional y en base a ellos genera el diseño de sistemas y el prototipo.

-**Desarrollador**, toma estos requerimientos del analista y los prototipos del diseñador y los traduce en un código ejecutable utilizando algún tipo de leguaje de desarrollo por ej. Java, C#, etc.

-**Tester o probador**, es el rol que asegura la calidad del producto, es el que va a asegurar que se concuerden con los requerimientos del cliente y los prototipos del diseñador.



1.6 PORQUE ES NECESARIO TENER SQA TESTER

Como todo proyecto importante, el éxito de un software depende del trabajo en equipo. Los equipos de TI están formados por Software Quality Assurance (SQA) testers, desarrolladores, y líderes de proyecto, quienes deben trabajar juntos como reloj suizo para crear proyectos satisfactorios.

Los miembros del equipo de desarrollo de software se pueden comparar con las herramientas necesarias para completar trabajos de la máxima calidad. Por ejemplo, para construir una silla, necesitaremos madera, tornillos, un martillo, un manual de instrucciones, una sierra eléctrica, pintura, una cinta métrica y lija. Por supuesto, usted seguramente pensará que puede construir una silla con menos herramientas, y estaría en lo correcto; pero no será la mejor silla, no tendrá buena estabilidad, ni soporte, tampoco será estética y seguramente tampoco cómoda. Muchas veces para ahorrarnos materiales o pasos nuestro producto no acaba siendo el más óptimo ni adecuado.

Lo mismo pasa cuando queremos desarrollar un software, una aplicación o un sitio web: no basta con tener solo un desarrollador es necesario tener un equipo de profesionales con funciones específicas. En ese sentido, los software QA testers son esenciales para cualquier equipo de desarrollo de software, pues generan un valor agregado al producto que se va a entregar.

1.6.1 Importancia del SQA

El software testing es fundamental, ya sea implementando como un rol de soporte para desarrolladores o como una entidad independiente. Hay que hacer varias pruebas de software para poder detectar los errores, corregirlos y entregar un producto de buena calidad.

Es necesario que los software SQA testers trabajen de forma paralela con los desarrolladores, pues una comunicación más efectiva permite al equipo encontrar errores y hacer mejoras durante todo el proceso.

Un software quality assurance tester le ahorra dinero a la empresa, identificando y corrigiendo errores durante la fase de desarrollo. En comparación, si no se realizará ningún control de calidad hasta después de terminado el producto, los errores encontrados costarían más tiempo y por lo tanto, dinero para corregir.

De los múltiples beneficios que el SQA brinda al proyecto, el más importante es que **asegura la satisfacción del cliente**, lo cual a su vez mantiene la reputación de su empresa. Es por esta razón que las mejores compañías no escatiman en el salario de los software quality assurance testers.

1.6.2 Quality Control vs. Quality Assurance

Quality Control (QC) y Quality Assurance (QA) son dos términos que ocasionalmente se usan indistintamente, sin embargo, se refieren a distintos aspectos de la administración de calidad.

A través del Quality Control, el equipo verifica que el producto cumple con los requerimientos funcionales. El Consejo Internacional de Calificaciones de Pruebas de Software (ISTQB por sus siglas en inglés) lo define como la serie de actividades diseñadas para evaluar la calidad.

Por otro lado, Quality Assurance son los procesos que auditan los resultados de las mediciones de Quality Control, para garantizar que los estándares de calidad se están cumpliendo. Estos procesos son los que previenen errores y defectos en el producto.

1.6.3 ¿Cómo se lleva a cabo el QA y por quién?

El control de calidad es ejecutado durante todas las fases del ciclo de vida del desarrollo de software (SDLC por sus siglas en inglés). Realizar un control continuo durante cada etapa o iteración evita que se cometan errores significativos que causen un retraso en el proyecto y un mayor gasto de presupuesto.

Los software quality assurance testers son los responsables de garantizar este control de calidad y no necesariamente tienen que formar parte de la empresa, existe la opción de completar un equipo de desarrollo con un SQA tester mediante la subcontratación o mediante Aumento de Personal.

El Staff Augmentation es una alternativa flexible para completar temporalmente los equipos de desarrollo con talento profesional. De esta manera las organizaciones se ahorran el costo de infraestructura y entrenamiento, a comparación de contratar un QA tester a tiempo completo.

1.6.4 Actividades de un Software QA Tester

Las actividades de un software QA tester durante la ejecución se pueden resumir en los siguientes pasos:

- **Análisis de Documentación**

  El primer paso que hacen los SQA testers es el análisis de documentación del proyecto, porque tienen que entender en su totalidad los requerimientos del cliente. De este modo se aseguran que los aspectos funcionales y no funcionales del proyecto cumplan con las especificaciones.

  Posteriormente, planifican la estrategia de la prueba de software, el alcance la prueba, y establecen los plazos. Adicionalmente, definen todas las herramientas que se van a utilizar para buscar los errores, los métodos, recursos y responsabilidades a los demás SQA testers.

- **Diseño de planes**

  El siguiente paso es diseñar los planes, aquí se elaboran casos de prueba y listas de verificación donde abarcan todos los requisitos del Software. Cada caso de prueba tiene que tener condiciones, datos y los pasos necesarios para validar que funcione, además de un resultado esperado bien definido para poder comparar contra los resultados reales.

- **Ejecución de pruebas y reporte de defectos**

  Este es el paso donde se ejecutan las pruebas y se reportan los defectos, se empieza con los desarrolladores realizando unit tests y los SQA testers realizan pruebas en los niveles de API y UI. Los errores que se detectan se envían a un sistema de seguimiento de defectos. Una vez que se encontraron los errores y se corrigieron, los SQA testers vuelven a probar las funciones para asegurarse que no les haya pasado desapercibido algún otro error. También hacen pruebas de regresión para verificar que las correcciones no hayan afectado las demás funciones.

  Por último, ya que los desarrolladores hicieron un reporte con las funciones implementadas y errores corregidos, el SQA elabora un informe final en el cual se certifica que el producto cumple con todos los

requerimientos de desarrollo, cumple con los más altos estándares de calidad y está listo para ser lanzado al público.

Sin embargo, el trabajo de un Software QA tester no termina allí, puesto que aún después del despliegue a producción, se deben ejecutar pruebas post-producción para evaluar aspectos excluidos en el ambiente de testeo. Un ejemplo de esto es la diferencia de datos, como el número de usuarios que el producto puede soportar en un determinado momento.

Si bien los desarrolladores son los miembros del equipo de TI de quienes más se habla (y sin duda son vitales), los software quality assurance testers no deben de ser subestimados. Ellos trabajan mano a mano con programadores para prevenir, solucionar, y documentar errores en el producto final. Sin su buen ojo, un software plagado de errores podría llegar a desplegarse, causando grandes problemas en el futuro.

Por lo tanto, si está pensando en armar un equipo de desarrollo de software, asegúrese de que esté completo. Servicios de Aumento de Personal o Equipos Dedicados pueden ayudar a llenar cualquier vacío en sus recursos de talento actuales, asegurando que su proceso de desarrollo sea eficiente, efectivo, y de calidad superior.

1.7 QUE ES TESTING O PRUEBAS DE SOFTWARE

- Metodología para encontrar defectos de software.
- Proceso utilizado para medir la calidad de cualquier software.
- Proceso de verificación del correcto funcionamiento de una aplicación.

Existen muchas definiciones y todas son válidas, pero básicamente ***las pruebas de software o testing es el proceso que permite verificar la calidad de un producto de software***. ***Permite identificar posibles fallos en la implementación, calidad o usabilidad de un programa***. No solo es que vamos a ejecutar la prueba 1, prueba 2, etc. ***incluye muchas otras actividades desde la planificación de las pruebas, la preparación de las pruebas, la evaluación de los productos de software para determinar si cumple o no con los requerimientos especificados***.

EL TESTING PUEDE PROBAR LA PRESENCIA DE ERRORES, PERO NO LA AUSENCIA DE ELLOS.

Durante la planificación determinamos que pruebas vamos a correr, en la ejecución vamos a correr esas pruebas, pero que pasa si existe alguna funcionalidad que no estamos tomando en cuenta o al algún “test case” no cumple una determinada funcionalidad por completo.

Para realizar un buen testing de software debemos tener en cuenta varios conceptos involucrados y la diferencia entre ellos.

- **Error**, es una acción humana que produce un resultado incorrecto. Es una idea equivocada de algo, como las ideas las tienen las personas el error es una equivocación del desarrollador o del analista. Un error puede llevarnos a generar uno o más defectos. Ej. error en la programación, error en el typeo, un requerimiento que este mal especificado.

  Defecto, es un desperfecto que se encuentra ya sea en un componente o en un sistema, y que puede causar que este componente o este sistema falle en su funcionamiento. Ej. Una sentencia o una definición de datos que este incorrecta.

  La relación entre defecto y fallo es que un defecto causará un fallo. Es decir, si hemos localizado un defecto durante una ejecución entonces podremos causar un fallo en el componente o en el sistema. Ej. Que el desarrollador hubiese utilizado el operador menor en lugar de menor o igual.

Fallo, es la manifestación física o visible de un defecto. Si un defecto es encontrado durante la ejecución de una aplicación, entonces va a producir un fallo. Ej. Un crash del sistema.

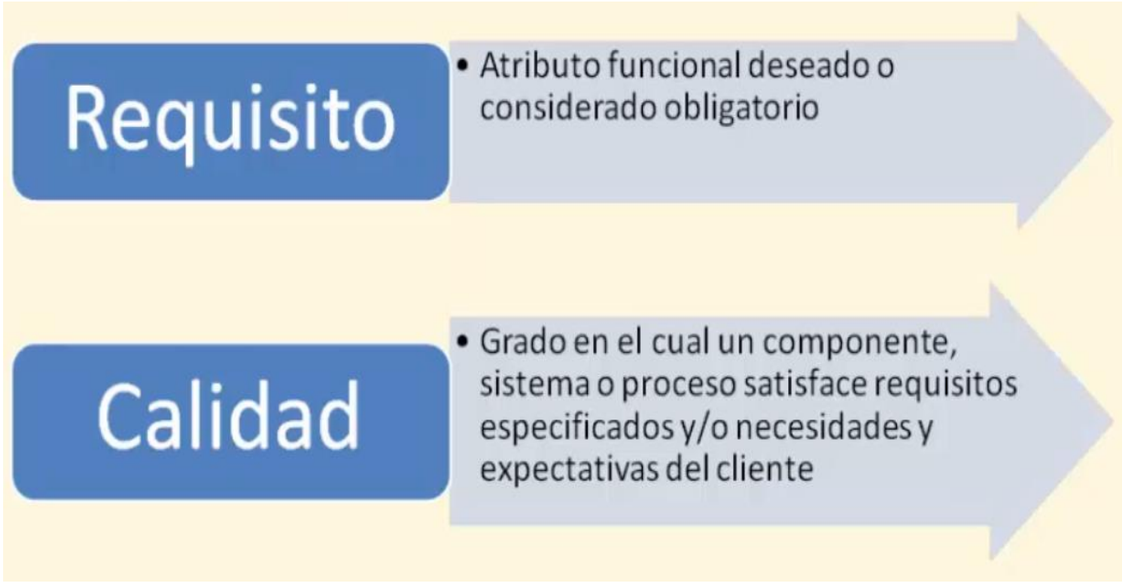
¿Cómo se relacionan estos tres conceptos?

Un error puede generar uno o más defectos, y un defecto va a causar un fallo.

Las causas de los fallos de pueden deber a dos grandes grupos:

- **Error humano**, porque el defecto se introdujo en el código de software, en los datos o en los parámetros de configuración, y ¿porque hubo un error?, porque tenemos plazos cortos para entregar las cosas o porque tenemos mucha complejidad o porque el desarrollador o analista se distrajeron.

- **Condiciones ambientales**, los cambios en las condiciones ambientales como ser: radiación, magnetismo, fallo en el disco duro, etc.



-**Requisito** (o requerimiento), describe un atributo funcional deseado que es considerado obligatorio. Significa que un sistema no va a ser aprobado por el cliente sino cumple con esos atributos que el mismo ha definido. Ej. El sistema permita editar o eliminar un determinado contacto, no se permita continuar hasta que no se haya registrado un campo obligatorio, que sea fácil de instalar (requisito no tan tangible).

-**Calidad** (calidad de software), es el grado en el cual un componente, un sistema o un proceso satisface los requerimientos o requisitos especificados y las expectativas del cliente. La calidad del software es ***la suma de todos los atributos que se refieren a la capacidad del software de satisfacer los requerimientos dados***.

Los atributos pueden estar categorizados en dos grupos:

a) **Atributos funcionales:** (tendremos 2 características básicas y sumamente importantes)
-**correctitud**, la funcionalidad satisface correctamente los atributos requeridos.
-**completitud**, la funcionalidad de software satisface todos los requisitos que el cliente ha pedido, incluirá: adecuación, exactitud, interoperabilidad, seguridad y el cumplimiento de la funcionalidad.

b) **Atributos no funcionales:** (son más difíciles de lograr porque no están exactamente definidos o bien establecidos)
-**fiabilidad**, el sistema va a mantener su capacidad y funcionalidad a lo largo de un periodo determinado de tiempo. Ej. Si tengo un bug que se presenta cada dos días mi sistema no es confiable.
-**usabilidad**, el sistema es fácil de usar, es fácil de aprender, tiene un uso intuitivo y se ha desarrollado conforme a las normas iso 9000, etc. de acuerdo al sistema.
-**portabilidad**, que sea fácil de instalar o desinstalar, que sea fácil de configurar los parámetros necesarios, que sea fácil de trasferir a otro entorno.
-**eficiencia**, que el sistema requiera de un mínimo de recursos para ejecutar una tarea determinada. Ej. Si el logeo tarda 30 seg. mi eficiencia no es buena.
-**mantenibilidad**, es una medida del esfuerzo que se requiere para realizar cambios en los componentes de un sistema.

Esto no significa que mi sistema deba tener todos estos atributos funcionales o no funcionales, el cual será dependiendo del sistema en el que estamos trabajando vamos a tener que priorizar los atributos que necesitamos, y en base a eso determinamos que tipo de testing aplicaremos a nuestro sistema. Ej. Aplicación de escritorio no hace falta priorizar la portabilidad. Aplicación web si necesitamos la eficiencia.

1.8 PORQUE SON NECESARIAS LAS PRUEBAS

El software está en todo lugar. Se podría decir que está presente en todos los aspectos y momentos de la vida de las personas o mejor dicho en casi todos. Algunas veces el software es obvio, otras veces no.

¿QUE SON LAS PRUEBAS?

Es una manera de explorar, de revisar el producto que tú quieras desarrollar, experimentarlo, verlo, entenderlo. Entre menos entiendas el producto que estas desarrollando más errores va a cometer.

¿Pasando todas las pruebas tendré un software sin errores?

¿Al hacer miles de pruebas se tiene un software sin errores? → NO

La seguridad de que el 100% de que un software no tenga errores no va a suceder, podrá llegar a un buen porcentaje de un 99%.

Se puede tener un software muy estable pero cuando los usuarios empiezan a utilizar nuevos dispositivos, salen nuevas versiones de software, etc. Entonces se debe actualizar el software para que funcione correctamente con los nuevos dispositivos.

El tiempo y los clientes nos han enseñado que pese a tener un buen plan de pruebas siempre se nos puede ir algunos detalles.

Para aclarar el motivo de esta necesidad veremos algunos ejemplos:
-**La Mariner 1**, en el año 1962, fue un cohete la 1ra. Misión de la NASA para sobrevolar Venus, el cohete no duro más de 5 min. en vuelo cuando desvió su trayectoria teniendo que ser autodestruido por la NASA. El motivo la omisión de un guion en el programa que controlaba el cohete. ¿Qué hubiera pasado si se le hubiera realizado el testing necesario?
- **Therac-25**, máquina de radioterapia usada para entornos médicos, entre 1985 y 1987 hubo seis accidentes en la que los pacientes recibieron una sobredosis de radiación donde varios murieron. Se determinó que el software tenía en su diseño un código indocumentado. Otro ejemplo de porque es necesario hacer muchas pruebas y distintos tipos de pruebas.
-**MIM 104**, misil antiaéreo que se usaba para interceptar otro tipo de misiles a modo de defensa, durante la guerra del golfo en 1991 un misil mato 28 soldados debido a una falla debido a un error en el software del reloj del sistema porque se había retrasado en un tercio de segundo al haber estado activado 100 horas seguidas.

- Apple enseña la nueva característica de reconocimiento facial, la cual fallo
- Windows, la pantalla azul WIN98 Bill Gate fue descubierto en una presentación.

No solamente hay situaciones que pueden poner en duda si lo que estamos haciendo está bien o no, sino también situaciones legales de manera implícita → caso UBER, en su cobertura de pruebas no estimaron la parte de seguridad, causado el despido del responsable de seguridad.

A pesar de toda esta situación mucha gente que desarrolla software, pequeñas o medianas empresas, free lanzers, no consideran las pruebas como parte del desarrollo ya sea porque contractualmente no lo incluyeron, o les faltó tiempo para terminar y la clásica respuesta de que no se pudo y será en la siguiente iteración, o la cobertura que tienen es muy mala y nada más están pensando en algunos procesos y no en todo el flujo de datos, ni en la parte de BackEnd, ni en la Base de Datos, o en la seguridad, etc.

Así como esto existe millones de ejemplos, el software es parte de nuestra vida, constantemente estamos interactuando con el en todos los ámbitos: Ej. cuando prendemos el microondas, cuando encendemos el televisor, por eso es necesario que el software sea de calidad.
Por lo tanto, podemos resumir que hay **muchas razones por lo que un software debe ser probado** pero las más importante son:
-**Es realizado por seres humanos**, los seres humanos cometemos errores, podemos saber mucho o poco pero no vamos a saber todo. Podemos tener muchas habilidades, pero no vamos a ser perfectos.
-**Presión en las entregas**, no hay tiempo para chequear las cosas que hacemos y asumimos que está todo bien y eso nos lleva a cometer errores.
-**Medir la estabilidad del software**, es otra de las razones por las que hacemos pruebas.
-**Hace lo que debe hacer**, para demostrar que el software no tiene fallas y que hace lo que tiene que hacer.
-**Costos**, encontrar las fallas durante el proceso de desarrollo es mucho más barato que cuando ya está en producción y lo está usando el cliente.

1.1 Objetivos de las pruebas

-**Adquirir conocimiento**, para conocer los defectos que tiene un objeto de prueba y así describirlos de tal forma que facilite su corrección.
-**Confirmación de la funcionalidad**, si se ha implementado tal cual como se había especificado.
-**Generación de información**, proporcionamos información sobre posibles riesgos relacionados al sistema antes que este sea entregado al usuario.
-**Confianza**, que el sistema cumple con la funcionalidad esperada.

El testing exhaustivo es imposible, así como probar todas las combinaciones de entrada y precondiciones, y no terminaríamos de probar nunca porque hay infinitas posibilidades, entonces como sabemos cuánto testing es necesario realizar para cada producto de software, cuando debemos parar con las pruebas, par esto debemos basarnos en riesgos y prioridades que es bastante subjetivo porque depende de cada proyecto pero básicamente por un lado tenemos **"los criterios de salida"** (no encontrar más defectos puede ser un criterio para terminar las pruebas).--> Conjunto de condiciones que se debe acordar entre todos para que el proceso concluya. Ej. Se va a terminar con el testing cuando se haya probado un 80% de la aplicación.

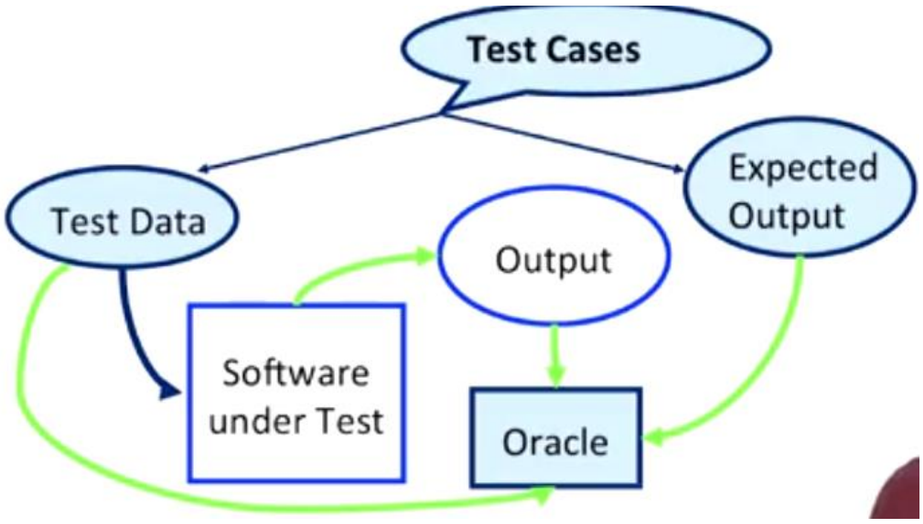
1.9 RAZONES PARA HACER PRUEBAS

Debemos considerar los siguientes aspectos:

- Si decidimos querer tener un software bien hecho, estamos viendo el problema y estamos en mejoras continuas, que nos lleve a ver que pruebas realizar para que en la siguiente liberación de software trabaje de la forma deseada.
- Los costos se están volviendo muy altos, hay equipos que están resolviendo problemas en lugar de tener buenas prácticas y monitorear todo el proceso, no conocer lo que se está queriendo hacer puede ser que genere más defectos dentro del mismo software porque al tratar de arreglar algo vamos a descomponer otras partes del software o todo lo demás.
- Estándares o implicaciones legales, que condiciones debe tener el usuario para vender el software cumplir los estándares tanto de desarrollo, entrega, etc.

1.10 PROCESO DE EJECUCIÓN DE PRUEBAS

- ✓ Proporcionar información
- ✓ ¿Observamos la salida ?¡Que hacemos con esto…! ¿Está bien?
- ✓ ¿Quién debe hacer esta comparación?
- ✓ El comportamiento coincidió con lo esperado



Que es una prueba y como se puede ejecutar algunas pruebas simples en su propio código.
Comenzamos con lo que se denomina “software bajo prueba”, no se puede probar algo que no está creado (exceptuando TDD) ya que estamos hablando de como ejecutar una prueba. Algo que solo se puede realizar una vez que se tenga el software a ejecutar.
Cuando decimos software nos referimos no necesariamente al producto de software terminado, sino todo lo contrario, aunque finalmente llegaremos a ese punto.
Software bajo prueba significa cualquier parte o subconjunto del programa que hemos completado donde podemos ejercer un comportamiento, este es algún modulo o unidad de código (pruebas unitarias).
Las unidades en este contexto generalmente significan algo así como un método, una función, subrutina o procedimiento, algún pequeño conjunto definido de pasos a tareas que tenemos una expectativa de cómo debería comportarse cuando ejecutamos este código.
Para ejecutar una prueba contra nuestro software o unidad, debemos proporcionar la información sobre la cual actúa la entrada, es decir los datos de prueba.
Hay muchas formas de seleccionar o generar estos datos de prueba.

- Basados en un perfil de cómo creemos que actuara el usuario.
- Basados en probabilidad, o estudios de usuarios
- O podemos atacar el código con algunos datos que a menudo causan errores como ser: “cero” 0, valores mayores, menores, palabras, números, null, etc.

Para cada una de estas entradas, ejecutaremos el software bajo prueba, dados los datos de prueba potencialmente después de configurar el software para que este en el estado correcto para que los datos de prueba de software tengan sentido.

Una vez que el software recibe los datos de prueba, observamos la salida, el comportamiento del programa dada la salida.

¿La pregunta entonces es?

¿Qué hacemos con esto? ¿Está bien?

¿El software ha proporcionado el resultado correcto dado los datos de prueba?

Algo o alguien tiene que hacer esa comparación, llamado “Oracle”, que viene a ser el desarrollador o probador que ejecuta las pruebas.

Se tiene el software funcionado, ingresan los datos, mira lo que pasa y decide si el comportamiento coincidió o no con lo que esperaban.

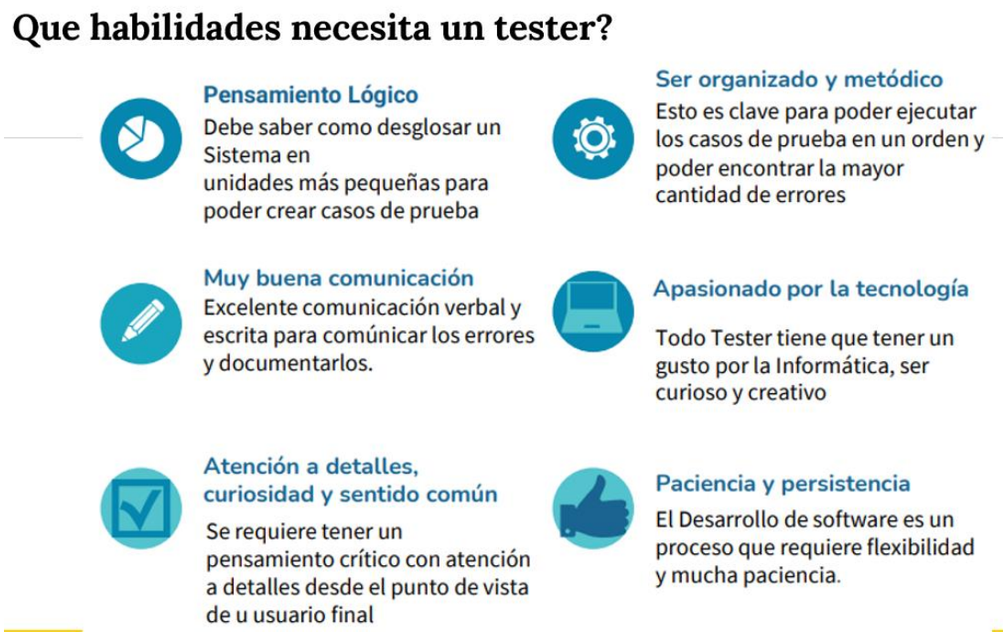
Ahora, como se puede saber o no de que puede existir un error humano en esto, ¿Los humanos somos particularmente confiables? ¿Puede decidir la diferencia entre el número “uno” 1 y la “letra l minúscula” cuando se escribe?

Es así que lo que se está empezando a ver y utilizar son los casos de “Oracle” automatizados que comparan algún resultados esperados y conocidos, determinados o recuperados a través de frameworks como ser: JUnit, PyUnit, etc.

El framework verificara automáticamente que ciertos casos sean verdaderos o falsos y proporcionara un informe visual, comparando esto con la salida que genero el software.

1.11 ¿QUE HABILIDADES NECESITA UN TESTER?

todo buen tester tiene ciertas habilidades que lo hacen ser un analista o ser una persona con un pensamiento bastante crítico.
Algunas de las más importantes son:



- ✓ **Tener un pensamiento lógico.**

Poderes de ser capaz de desglosar una aplicación, un sistema en unidades más pequeñas para poder crear casos de prueba. Entonces tú tienes que saber cuándo estás probando una aplicación, la tienes enfrente tuyo, ver la aplicación, hacer un análisis funcional de la aplicación y saber cómo se hace para sus opciones, para que sirve cada una y qué impacto tiene cada una de esas opciones para el negocio.
Se tiene que pensar como desde el punto de vista del negocio y también como desde el punto de vista de usuario y poder desglosar todas estas funcionalidades en unidades más pequeñas y poder así crear casos de prueba, entonces debes de tener un pensamiento muy lógico.

- ✓ **Tener también muy buena comunicación**

Una excelente comunicación verbal y escrita para comunicar los errores y comentarlos.
Como testers, nosotros estamos en constante comunicación con diferentes personas dentro de nuestro equipo, especialmente con los desarrolladores y con los análisis de requisitos y también con el dueño del producto. Entonces debemos de ser capaces de poder comunicar todo lo que vamos encontrando de una manera muy efectiva.
Hay personas que se les dificulta explicar cómo reproducir un error o como escribir los pasos para reproducir cierto escenario o en ciertos casos de prueba, o también se la dificultad explicárselo a una audiencia más grande, entonces nosotros tenemos que tener estas habilidades de muy buena comunicación.

- ✓ **Atención al detalle**

Esto es fundamental, los tester son atentos a los detalles. Deben de ser curiosos, y deben tener un sentido común. Cuando nosotros estamos probando una aplicación, se requiere tener un pensamiento crítico con atención a detalles desde el punto de vista de un usuario final. Si digamos que es una aplicación web y nosotros como tester tenemos que simular que somos un usuario final.

Entonces cualquier cantidad de detalles, por más mínimos que sean, nosotros tenemos que revisarlos y verificar que funcionen perfectamente. Hay que mirar detalles, hay que ser bastante críticos con todo lo que estamos mirando, porque dependiendo de lo que se esté probando, si no tenemos esta atención ante esos detalles, cualquiera de los pasos se puede convertir en un defecto y generar hasta un fallo total del sistema y la aplicación del módulo puede caerse o no funcionar cuando ya esté funcionando en producción.

- ✓ **Ser muy organizados y metódicos**

Esto es clave para poder ejecutar los casos de prueba en un orden y poder encontrar la mayor cantidad de errores ya que cuando nosotros estamos probando algo tiene que tener un cierto orden y hemos de seguir

como una metodología o los casos de prueba deben de estar organizados de una manera lógica y secuencial, porque para probar un caso de prueba hay algunos requisitos.

Muchas veces cuando vamos a hacer, por ejemplo, una prueba de una página de autenticación de usuario y tenemos que ingresar un usuario y una clave, antes de tener acceso a esa clave, deberíamos de registrarnos en otra plataforma y para registrar esa plataforma tendríamos que tener unos datos de prueba. Es así que debemos tener un orden lógico de los pasos que hay que hacer para poder ejecutar un caso de prueba.
Al ser organizados en ese sentido y de la misma manera, debemos ser muy organizados en la documentación de las evidencias de la prueba, estamos ejecutando unos casos de prueba y no tiene, y tenemos las evidencias como las fotos o los vídeos de cada uno de esos casos. Están desordenados, están por todas partes no vamos a poder encontrarlos fácilmente o reportarse a un desarrollador en caso de que haya un problema. Hay que ser muy organizados en la documentación, en la ejecución de cada uno de los pasos.

- ✓ **Ser apasionados por la tecnología**

Los testers tienen que tener un gusto por la informática, tiene que ser curioso y creativo, revisar las opciones y ser capaces también de poder configurarlas fácilmente para poder encontrar la mayor cantidad de errores a partir de toda esa curiosidad que debe tener un buen tester.

- ✓ **paciencia y la persistencia**

El desarrollo de software es un proceso que requiere flexibilidad y mucha paciencia.
Porque cuando se está probando una aplicación vamos a encontrarnos con una cantidad de problemas que no nos tienen que parar ya que seguramente no nos van a permitir continuar fácilmente. Entonces, como está en nosotros, tenemos que tener mucha paciencia y entender que el desarrollo del software es un proceso muy complejo y que requiere de repetición.
Hay que tener mucha paciencia y persistencia, porque hay que probar y probar y volver a repetir los mismos casos, hasta que todos los errores de todos los casos te pasen completamente y responden exitosamente de manera que no tengamos ningún otro problema.

1.12 RESPONSABILIDADES DEL TESTER O QA EN UN EQUIPO DE DESAROLLO DE SOFTWARE

la diferencia entre tester y QA en la práctica es lo mismo. En las empresas los tratan de igual manera, pero en la teoría un analista de QA difiere un poco de lo que es el tester, porque ***el analista de Aseguramiento de Calidad o Quality Software Assurance se encarga más del aseguramiento de la calidad y de todo el proceso de desarrollo del software viéndolo a manera macro, mientras que el tester solamente se encarga de la ejecución de las pruebas como tal, pero eso es un concepto que vamos a mirar más adelante***.
En la práctica las empresas se refieren al tester o QA como la misma persona.
Algunas de las responsabilidades más importantes son:

- ✓ **el diseño de un plan de pruebas como tester**

Son los encargados de realizar un plan de pruebas. Eso depende mucho del tipo de proyecto que estamos trabajando, si es un proyecto muy grande, vamos entonces a elaborar un plan de pruebas, que es como si

fuera un mapa o una hoja de ruta en lo que se va a hacer y la estrategia que, a usar, como se va a probar, con qué tipo de datos probar, etcétera.

- ✓ **Definir los casos de prueba con base a los requisitos del cliente o el negocio**

Son los encargados de hacer un análisis funcional de la aplicación y entender muy bien qué es, lo que hace y cuáles son los requisitos que la empresa está solicitando al equipo de desarrollo.
Interpretar esos requisitos y crear una cantidad de escenarios de prueba que va a permitir encontrar los errores después. (diseñar casos de prueba y ejecutarlos).

- ✓ **Gestionar el ambiente y los datos de prueba**

Cuando se está probando una aplicación, ya sea en una empresa o para un cliente externo, se tiene que hacer una instalación de nuestro sistema. Si creemos que es una aplicación que va a estar instalada en nuestro computador, entonces tenemos que gestionar todo un ambiente de pruebas para que esta aplicación pueda correr correctamente en nuestro computador.
Si es en una red interna de una empresa, tenemos que gestionar todos los permisos necesarios, accesos de la base de datos o todo lo que sea necesario para que esa aplicación pueda correr exitosamente y podamos efectivamente ejecutarla y probarla. Para eso debemos guiarnos con la ayuda de un desarrollador o ingeniero de infraestructura o con cualquier persona dentro de la empresa que nos ayude a configurar la aplicación para que podamos ejecutar las pruebas.
A eso se refiere toda la gestión del ambiente y de igual manera a los datos de prueba, etc.

- ✓ **Ejecutar los casos de prueba**

Obviamente, ya después de haber diseñado todos estos casos, es quien está encargado de ejecutar estos casos de prueba con todas las combinaciones posibles para poder encontrar todos los errores el tester los ejecuta y también responsable de documentar todas las evidencias de las pruebas. Ej. una aplicación en un celular, vamos a estar probando todas las opciones que nos ofrece una aplicación, estar grabando todo lo que está pasando allí para poder tener una evidencia por donde se pasó y que se probó. Cuando encontramos un error poder mirar cómo ser por medio de un video, por ejemplo, de cómo se genera el error, poderlo replicar nuevamente y poder dar la muestra a un programador para su futura corrección.

- ✓ **Documentar todo ese tipo de cosas**

y tenerla en un repositorio de evidencias y que se pueden ver en un documento de Word o un documento Excel como un administrador de casos de prueba.

- ✓ **Reportar los errores encontrados y realizar seguimiento para su corrección y revalidación**

Una vez que se ha encontrado una cantidad de bugs o errores en el sistema se va a reportar al programador y tenemos que reportarlo de una manera específica. Se puede utilizar un formato específico o formato que ya exista en un sistema de manejo o administración de efectos de errores. También es el encargado de hacerle seguimiento a este error hasta que sea corregido y se pueda probar nuevamente hasta que ya el error sea corregido y termine todo su ciclo de vida.

- ✓ **Participar en todas las reuniones de seguimiento que tenga el equipo**

En los eventos que dictamina la metodología Scrum o la metodología que se tenga definido.
Ej. metodología Scrum, en el Desarrollo de software para poder construir software de una manera ágil. El test es participar en todos estos eventos que se hacen diario y semanalmente.

- ✓ **Realizar informes de calidad del producto una vez se han ejecutado**

De todo un set de pruebas, el tester debe crear un informe de calidad del producto con todos los hallazgos que ha encontrado y lo provee al equipo de desarrollo para que posteriormente se puedan corregir todos los defectos, todos los errores que se han encontrado y también que puedan tomar decisiones acerca de la calidad del producto.
Esta información la usan mucho los Managers o los dueños del producto (product owner) para tomar decisiones sobre si un producto o un sistema completó cumple con las expectativas para poder pasar a producción o poderla liberar al público en general.

- ✓ **Ayudar a resolver las dudas a los analistas de requisitos y a los dueños del producto (PO)**

Como son los testes los que están probando y están adquiriendo el conocimiento del sistema que se está probando en el momento, es responsabilidad del tester resolverles dudas a los analistas de requisitos, inclusive a los desarrolladores o a otros tester en cuanto a la funcionalidad o a la manera como se ve, cómo funciona la aplicación que estamos probando, porque nos volvemos como unos expertos en lo que estamos haciendo de la repetición y por la experiencia que tenemos con la aplicación.
Los testers son las primeras personas que están consumiendo una funcionalidad y por tal motivo nos volvemos como los expertos en ello. Entonces son un punto de referencia para el resto del equipo.

- ✓ **Ayudar a los programadores a replicar los errores y a investigar su solución**

Cuando los testers encuentra defectos, muchas veces cuando los reportan a los programadores, el reporte debe tener un formato donde el programador pueda leer la cantidad de pasos que ejecutar para llegar al error, pero muchas veces estos reportes no son claros o no son obvios o fáciles de entender para el

programador, entonces nosotros tenemos que ir a ayudarles a los programadores a replicar los errores para que ellos puedan corregirlos.
Trabajamos en equipo con ellos y les indicamos paso a paso de manera más detallada de cómo replicar el error fácilmente para que ellos rápidamente lo puedan corregir y nosotros lo podamos probar nuevamente cuando ya esté arreglado

- ✓ **Implementar prácticas de aseguramiento de calidad para prevenir errores en el código**

Dentro del proceso de desarrollo de software podemos implementar prácticas para prevenir los errores. Por ejemplo, hacer una sugerencia de cómo implementar pruebas unitarias, pruebas a los requisitos, como se está definiendo los requisitos con los analistas de requisitos.
Como nosotros somos expertos en el producto, podríamos implementar sugerencias de al momento de escribir los criterios de aceptación, porque como ya conocemos tanto el producto, podemos ayudarles a los analistas de requisitos, como es que deberían ser las funcionalidades para que estas no sean ambiguas para los desarrolladores difíciles de entender y eventualmente ellos resulten modificando algo que no va a funcionar o que no tiene sentido desde ningún punto de vista.