Conceptos Básicos

¿Probar un producto?

Testing: conjunto de actividades/procesos que sirve para validar el grado de calidad….

Probar es evaluar o experimentar un producto para conocer funcionamiento, calidad, rendimiento, utilidad,etc..

Cuando pruebo un producto voy a conocer la calidad, evaluar que tan bueno esta (no si esta hecho con equis material o eso)

La calidad de las pruebas se puede aplicar en:

- Industria automotriz (probar airbag, si no se prueba bien, pueden haber incidentes y se infringen normas, se pierden clientes.)
- Industria laboratorios (probar eficacia de una formula con consecuencias etc.)
- Industria tecnológica (probar funcionalidades de apps de pago)

Para que se testea?

●Porque una falla puede afectar la credibilidad de una marca.
● Para generar confianza en la marca y en la poca probabilidad de falla de sus productos.
● Queremos estar tranquilos, aun cuando nos cueste más caro.
● Para asegurar la calidad del producto y reducir el riesgo de fallas cuando esté operando.

Salida con Fallas

No se puede probar todo, por lo que reforzar el testing y prueba disminuirá los fallos
Caso de uso es tomado como **requerimiento**.
Podes perder usuarios, perder calidad, facturar menos,etc.
Ejemplo: Explosión Challenger. Se sabía que había problemas (fallas) pero decidieron sacarlo igual.

Errores Comunes

Pruebas

Ejecución del software = pruebas dinámicas Revisión de requerimientos o código fuente=pruebas estáticas

Diferencia entre verificación y validación

1- Antes de codear, hay que **validarlo**, relacionado al requerimiento del usuario y la resolución del problema de este. ¿construimos el software adecuado?
2-Cuando ejecutamos el proceso de testing de un producto estamos **verificando.** Si está bien construido el software
Una vez ahí, se ejecuta todo

Conceptos

Software y app == programa informático (software) diseñado para resolver funciones específicas
Store: donde compramos apps
Sprint: Fijación de objetivos en un ciclo (período de tiempo fijo en el cual un equipo trabaja)
DB: Data Base (lugar donde se guardan los datos en tablas)

Query: consulta (llamada a las tablas)(relacionado a SQL)
Scrum: marco de trabajo para el desarrollo ágil del software. (proceso donde se aplican prácticas para trabajar colaborativamente)
QA: lo mismo que un tester (no tanto,l laboralmente se busca mas un QA junior)
Ciclo de desarrollo de software: Conjunto de pasos que se utilizan para crear aplicaciones de software. el resultado es un producto con la más alta calidad y el menor costo posible en el menor tiempo posible. (≠ ciclo de testing)
Arquitectura de tres capas/niveles: Arq. que separa las apps en tres niveles de informática lógica y física.

Prueba de Software

Cuando uno testea, tiene que entender el NEGOCIO que este representa.

Proceso
Abarca actividades durante todo el ciclo de vida.
Cuanto más tarde encontremos errores, más costoso.
(+ temprano = software correcto y a menor costo)
Las pruebas se deben preparar, se verifica el resultado, se prueba el código del producto.
Abarca pruebas estáticas y dinámicas. Se planifica antes-durante-después de la ejecución.

Objetivos
Revisar el diseño para ver si cumple con los requisitos.
Revisamos si el producto es el adecuado.
Detectar defectos.

Calidad (ppt)

(IEEE,ISO; W.Deming)
Calidad del software es el grado en el cual un sistema, componente o proceso se ajusta con las expectativas o necesidades del cliente o usuario
Aplica en:

- En procesos
- Debe estar orientada al cliente
- Es mejora continua siempre
- Debe ser medible
- Nos involucra a todos

QA vs QC

QA :Quality Assurance:
Orientado al **Proceso** y estándares definidos, revisa procesos, asegura que cumplan con los estándares organizacionales y previene defectos.

QC: Quality Control
Orientado al **producto**, verifica cumplimiento de requisitos y detección de defectos
Lo que no es medible no tiene calidad.

QM: Quality Management
Manejo y coordinacion.

Conceptos de Calidad y Testing

Circulo Virtuoso de Deming (PDCA)

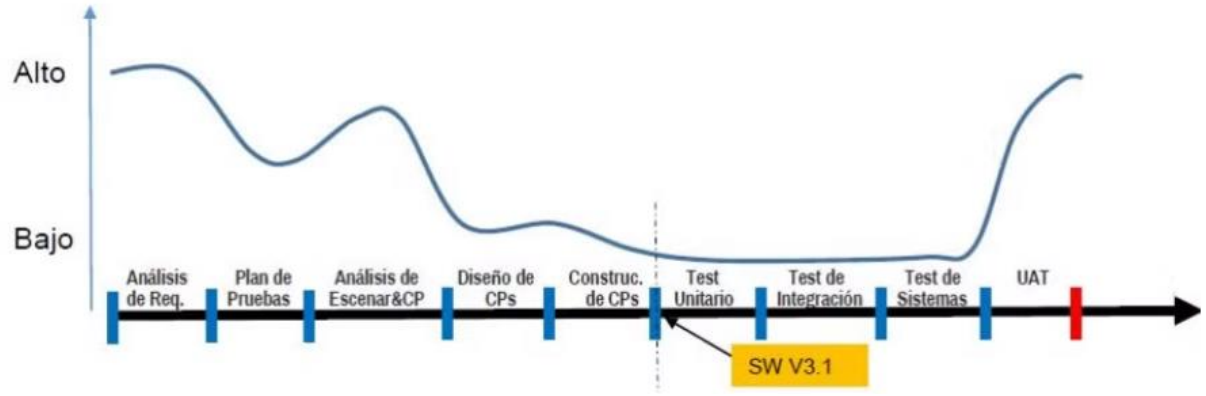
ISTQB

ISTQB= ORG líder mundial en la certificación de competencias en test de software

> Testing es el **proceso** que consiste en toda las actividades del **ciclo de vida,** tanto estáticos como dinámicas relacionadas con la **planificación**, **preparación** y **evaluación** de productos de software y productos relacionados con el trabajo para determinar que cumplan con los requisitos especificados, para demostrar que son aptos para el propósito y detectar defectos. Sirve para detectar problemas tan pronto sea posible, para verificar que las pruebas cumplan con el requerimiento, para generar compromisos con los usuarios y para trabajar orientado al cero defecto.

Primero hay que entender el requerimiento para saber que testear
Si cumple con esto, el producto tiene calidad

Nivel de compromiso/interrelación del usuario



Concepto testing

Actividades del Tester

API = ***application programming interface.***

Error, Defecto y Falla

Error = Error or Mistake Defecto
Error=equivocación
Bug=Defecto=Problema=falta

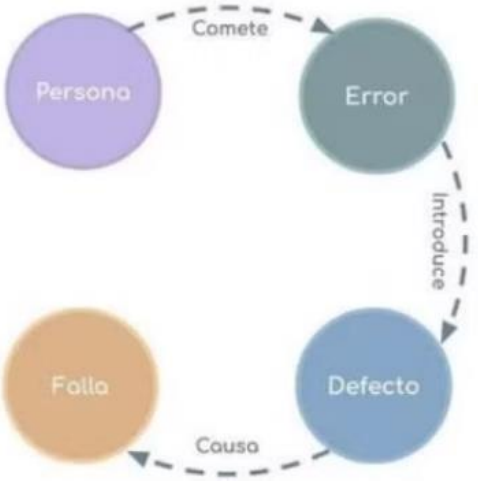
Error
Del programador

Defecto
en el software

Falla
depende del sistema

Código Defectuoso

Si se ejecuta el código defectuoso se podría generar una falla



Como resultado da un código defectuoso

Ejemplo

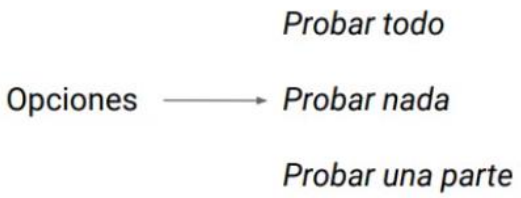
Efecto = reclamo de los clientes
Defecto = cálculo erróneo
Falla = pago incorrecto

7 Fundamentos del Testing

1 - La prueba muestra la presencia de defectos

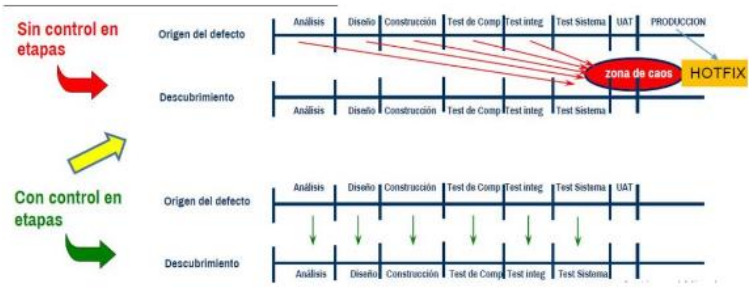
La prueba puede mostrar la presencia de defectos, pero no que no haya defectos. Reduce la probabilidad de que queden defectos

2 - La exhaustividad es imposible



Es imposible probar todo.

3 - Detección temprana de defectos



4 - Agrupamiento de defectos

Asociado a la complejidad del requerimiento.

Principio o regla de Pareto nos dice que para diversos casos, el 80% de las consecuencias proviene del 20% de las causas.

Para calcular:

- Restar las aristas menos los nodos y sumar 2:
- V(G) = Aristas – Nodos + 2
- V(G) = 7 – 6 + 2 = 3

5 - Paradoja del pesticida

Si las pruebas se repiten una y otra vez, eventualmente estas pruebas no encontrarán nuevos defectos

Necesario cambiar a nuevas pruebas y a nuevos datos de pruebas

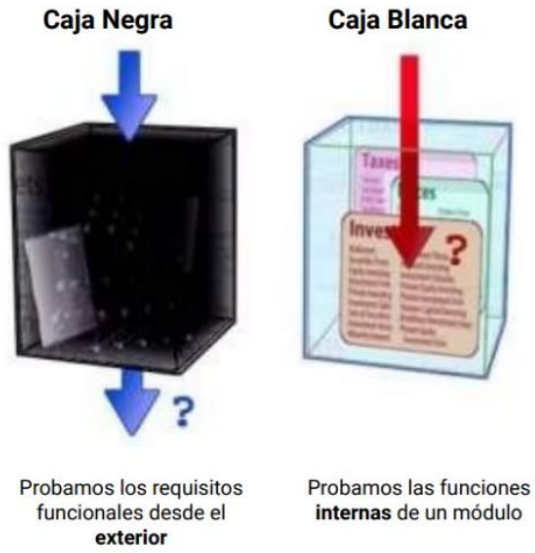
6 - Es dependiente del contexto	Las pruebas se realizan de manera diferente según el contexto. marcos predecibles (metodología tradicional) o framework agile (scrum). • Tiempo • Riesgo • Impacto
7 - La ausencia de defectos es una falacia	La ausencia de errores es una falacia (es decir, una creencia equivocada) conforme a que las pruebas contribuyen a minimizar los defectos en un gran número, pero eso no indica que se realice al 100 %

Tipo y Técnicas de Prueba

Entidades - Modelo mental

Clasificación de pruebas Tipos de Pruebas Técnicas de Pruebas

Clasificación de Pruebas



Caja Negra: FUNCIONAL

En las pruebas de caja negra se evalúa la funcionalidad de un sistema sin necesidad de conocer su estructura interna.

- Los **evaluadores** se centran en probar las entradas y salidas del sistema de acuerdo con las especificaciones. La idea es tratar el sistema como una "caja negra" donde solo se observa cómo reacciona a entradas sin considerar cómo se procesa internamente.
- Ayuda a identificar errores y problemas de usabilidad.
- **Esenciales** para asegurar que un software cumpla con los requisitos y funcione correctamente desde la perspectiva del usuario final.

Caja Blanca: AL CÓDIGO

En las pruebas de caja blanca se examina la estructura interna de un sistema.

- Involucran la inspección directa del código fuente y estructuras de datos para identificar posibles errores y garantizar una cobertura exhaustiva.
- Los **evaluadores** tienen acceso al código fuente y pueden diseñar pruebas específicas para alcanzar diferentes caminos de ejecución.
- Permite verificar la lógica interna, la coherencia del código y la implementación correcta de algoritmos.
- **Útiles** para garantizar que se cumplan estándares de codificación, detectar problemas de seguridad y optimizar el rendimiento del software.

Tipos de Pruebas

Los Tipos de Pruebas es un grupo de actividades que se centran en un objetivo en particular:

- Evaluar características funcionales
- Evaluar características no funcionales
- Evaluar estructura / arquitectura
- Evaluar código
- Evaluar efectos de los cambios

Caja Negra

Funcionales:
Verifican que el sistema haga lo que está establecido en los requerimientos funcionales.
niveles:

- Componentes
- Integración Func.
- Sistemas
- Aceptación

No funcionales

- Rendimiento
- Carga/Estrés
- Usabilidad
- Mantenimiento
- Confiabilidad
- Portabilidad
- Seguridad informática
- Implementación

Relacionadas a Cambios:
Conjunto de casos de pruebas en cada nivel (integración, sistema y componentes) para verificar funcionamiento del sistema, sin probar exhaustivamente.
Pruebas de:

- **Confirmación** (confirmar si se soluciona el defecto)
- **Regresión** (verifica que un cambio en el código no afecte por accidente otras partes.)
- **Smoke** (test de Humo)

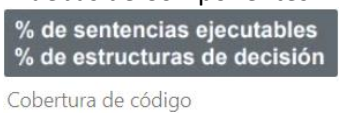
Caja Blanca

- **Unitarias**
- **Integración Técnica**

Pruebas sobre la implementación interna del sistema.

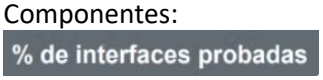
Dentro de la caja: código, arquitectura, flujos de datos,etc.

Pruebas de Componentes:



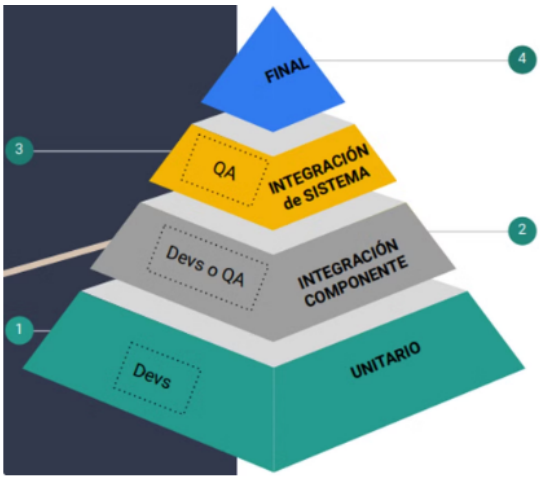
Cobertura de código

Pruebas de Integración de Componentes:



Tipos Por Nivel

Niveles de prueba ≠ Tipos de prueba



4.Pruebas de Aceptación (alfa/beta)
Normalmente realizado por el equipo de QA junto a los Usuarios en un ambiente casi idéntico al de producción, utilizando técnicas de Caja Negra.

3. Pruebas de Sistema
Realizada por el equipo de QA, se prueba el sistema como un todo. Prueba final para comprobar que lo entregado cumpla con la especificación. Se deben incluir pruebas funcionales y no funcionales.

1. Pruebas Unitarias o componentes
Probamos el código, sus módulos, objetos, estructuras y bases de datos.
Realizado por los programadores. Es común utilizar técnicas de Caja Blanca.

2.Prueba de Integración
Probamos la comunicación entre componentes dentro del software,con el sistema operativo, el sistema de archivos, etc..
Incluye la comunicación con sistemas externos (web services). Hay dos estrategias: Incremental / Big Bang

Técnicas de Prueba

- Valores Límites o Frontera
- Partición de clases de equivalencia
- Tabla de decisiones
- Árboles de decisión
- Casos de uso
- Unitarias por código
- Por cobertura
- Mutantes

Resumen

TECNICAS DE PRUEBA

Las técnicas me permiten ser más preciso, no se puede probar todo.
Es diferente a “tipo de prueba”

1-Casos de prueba: Tabla de decisión

Pasos

1-Identificar entradas/condiciones
2-salidas/acciones
3-combinaciones (V o F)
4-Escrbir casos de prueba

Cobertura

nro de decisiones probadas
—- ———————————- = %
nro total de condiciones

Tipos

Binaria: 01, no si, vf-
Decisión múltiple: pueden adquirir mas de un valor

Actividad 1

ejemplo: Sueldo de empleados según: productividad, si es encargado, si tiene infracción SE ELIMINAN TODOS LOS BONOS

> **Condiciones:**
> Empleado productivo
> Empleado encargado
> Empleado con infracción
>
> **Salidas**:
> Bono productivo
> Bono encargado
> No bono (infrac)
> No aplica

Condiciones	c1	c2	c3	c4	c5	c6	c7	c8
productivo	V	V	V	V	F	F	F	F
encargado	V	V	F	F	V	V	F	F
infracción	V	F	V	F	V	F	V	F
Salidas								
Bono productivo		X		X				
Bono encargado		X				X		
No bono (infrac)	X		X		X		X	
No aplica								X

Cn= caso de prueba numero...

Actividad 2

ejemplo: Si un alumno de vespertina tiene una nota ≥ 7 promociona, ≥4 presenta examen.

>Si un alumno de la diurna se saca ≥ 4 presenta examen, no existe promoción.

en ambos, si se saca menos de 4, no aprueba

> **Condiciones:**
> ≥ 7
> ≥4
> <4
> vespertina
>
> **Salidas**:
> Promociona
> Presenta final
> desaprueba

Condiciones	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16
≥ 7	V	V	V	V	V	V	V	V	F	F	F	F	F	F	F	F
<7 Y ≥4	V	V	V	V	F	F	F	F	V	V	V	V	F	F	F	F
<4	**V**	**V**	F	F	**V**	**V**	F	F	**V**	**V**	F	F	**V**	**V**	F	F
vespertina	V	F	V	F	V	F	V	F	V	F	V	F	V	F	V	F
Salidas																
Promociona							X									
Presenta final								X			X	X				
desaprueba													X	X	X	X

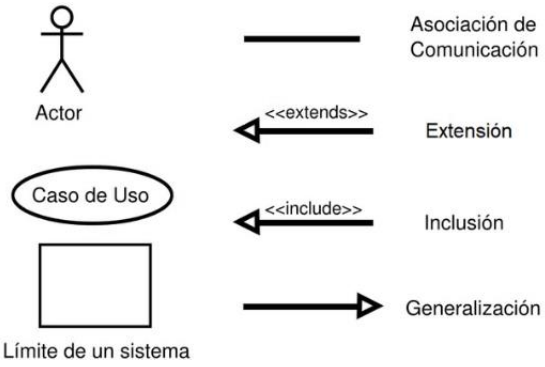
QUE LA CONDICIÓN<4 SEA V HACE QUE LAS DE ARRIBA SEAN F

DEL C1 A C6 SON ABSURDOS

2-Casos de Uso

Descripción de un uso particular del sistema por parte de un usuario del mismo (actor)
Es una plantilla estructurada con descripciones, un flujo,etc.
actor= puede ser usuario o un sistema
Vistos desde la óptica del actor.
Puede tener varios actores
Son útiles para los niveles de prueba de aceptación de usuario y sistema ya que nos sirve para encontrar fallas que un usuario detecta al usarlo por primera vez
≠ a casos de prueba

Elementos



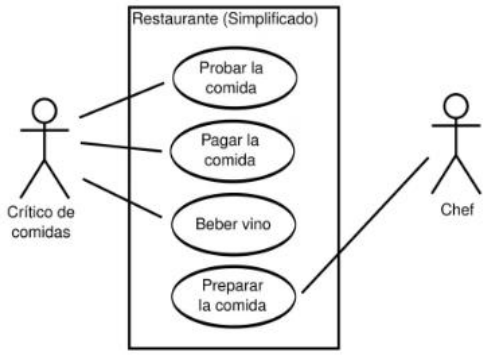
Ejemplos

ejemplo 1: restaurante

ACTOR

comensal

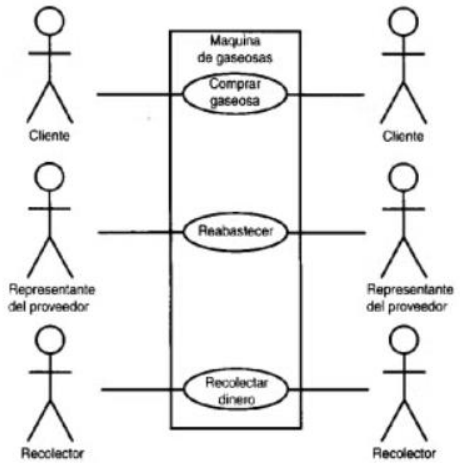
que interactúa con este sistema



ACTOR

chef

ejemplo 2:



3-Particiones

Ejemplo: Le tengo que asignar una tarjeta de crédito y hay condiciones:

- Entre 40 y 90k—→gold
- Hasta 200k—→plata
- +200k—→ black

Cobertura

$$\frac{\text{nro. de particiones probadas}}{\text{número de particiones identificadas}} = \%$$

(Hay 4 particiones y probé 3. tengo una cobertura de 75%)

4-Valores Límite

Análisis de valores frontera o límite→Para ordenar.

Ejemplo

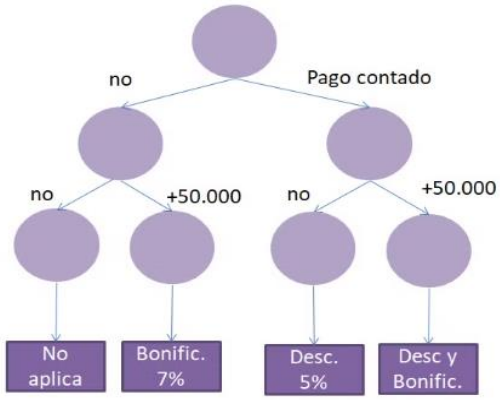
✓ Un campo de un formulario solo acepta números del 1 al 5.

Partición de equivalencia			Valores frontera	
Inválida (demasiado baja)	Válida	Inválida (demasiado alta)	Inválida (demasiado baja)-Válida	Válida- Inválida (demasiado alta)
0	1,2,3,4,5	6,7,8,9	0 y 1	5 y 6

5-Árbol de Decisión

Son diagramas que pretenden mostrar al gama de posibles resultados y las decisiones posteriores después de la decisión inicial
Un árbol lleva a cabo una evaluación a medida que este se recorre hacia las hojas para alcanzar una decisión
Todas las opciones son analizadas

Ejemplo



No siempre tiene que ser un sí o un no, se dan x condiciones.
Poner CP1, CP2…. en cada salidas

6-Basadas en Experiencia (teórico)

Aprovechan la experiencia y conocimiento de testers, usuarios y desarrolladores para diseñar e implementar casos de prueba

Se suelen combinar con pruebas de caja negra y blanca

Uso del software + su entorno + posibles defectos + distribución de los defectos

7-Pruebas de Sentencia

$$\text{Cobertura de sentencia} = \frac{\text{Número de sentencias ejecutadas}}{\text{Número de sentencias ejecutables}}$$

$$\text{Cobertura de DECISIÓN} = \frac{\text{Número de resultados de decisión ejecutadas}}{\text{Número total de resultados de decisión}}$$

Para IF THEN, ELSE WHILE,ETC

Una cobertura de decisión del 100% garantiza un 100% de cobertura de sentencia

Proceso Base de Testing

1- Entiendo requerimiento
2- Planifico: recursos, tiempo, tipo de prueba (funcional, no funcional,etc)
3- Creo los casos de prueba (c1: probas login ,c2,c3)
4- Preparo el ambiente: preparo datos de prueba y ambiente
5-Ejecuto la prueba.

Aplicados a metodologías

Cascada

Siguiente proceso empieza cuando el anterior termina.

Se pierde tiempo y dinero.

Se asegura cada etapa.

Equipo de prueba al final (desventaja)

Cascada V-Model

Adaptación.

Agile

Trabaja por ciclos (**Sprint**)

Se divide en pequeñas partes, por lo que si me equivoco, el daño es menor. Por ejemplo, sprint de 1-4 semanas. No hay jerarquía, trabajan para el producto.

1. Defino **Product Owner (PO)**: define lo que necesita el producto, alineado a lo que necesita el cliente
2. Lo que necesito para el producto se pone en el **Product Backlog**.
3. Empieza el **Sprint Planning Meeting**: para planificar que hacer
4. El Product Backlog pasa al **Sprint Backlog:** Se fija cuantos sprint se necesitan para las actividades del product backlog
5. Se crea un **Daily Scrum Meeting**: Que hice ayer, que hice hoy y qué impedimentos tuve.
6. Se crea un **Sprint Review**: Como venimos con este sprint
7. **Termina** sprint: Veo como están los entregables, si falta alguno se deja para el siguiente sprint
8. Fuera del string, se hace una **Reunión Retrospectiva** para ver oportunidades de mejora.

- **Scrum Master**: Persona responsable de que todos los procesos se ejecuten.Organiza las dailys.
- **Equipo**: DEV, QA, UX UI.
- **PO**

Casos de Prueba

Conjunto de valores de entrada, precondiciones de ejecución, resultados esperados y postcondiciones desarrollado con un objetivo en particular o condición de prueba,

- Short y long description (”login. etc.”)
- Pre-condiciones (”probar con un usuario que exista”)
- Steps (” ingresar, escribir, clickear aceptar”)
- Prioridad (orden)
- Severidad (impacto en el negocio)
- Lo Esperado (que haga x)
- Resultado **PASS** O **FAIL**

Características

- identificado unívocamente
- Tiene un estado (pendiente, aprobado fallido)
- re ejecutable

Estado

Al construir CP: iniciado, en proceso, terminado.
Al ejecutar: paso, fallo.