Testing

La calidad es un valor en sí mismo y no un gasto que las empresas deben realizar para que su negocio prospere.

# 7 Principios de Testing

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

No se puede probar que no hay defectos. Se reduce la probabilidad de que queden defectos no descubiertos en el software pero no se asegura que no existan.

## 2. La prueba exhaustiva es imposible

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.

## 3. La prueba temprana ahorra tiempo y dinero

Las actividades de testing, tanto estáticas como dinámicas, deben iniciarse lo antes posible en el ciclo de vida de desarrollo de software para ayudar a reducir o eliminar cambios costosos.

## 4. Los defectos se agrupan

En general, un pequeño número de módulos contiene la mayoría de los defectos.

## 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 detectarlo, es posible que sea necesario cambiar las pruebas y los datos de prueba existentes.

## 6. La prueba se realiza de manera diferente según el contexto

Por ejemplo, el software de control industrial de seguridad crítica se prueba de forma diferente a una aplicación móvil de comercio electrónico.

## 7. La ausencia de errores es una falacia

# Mesa de 3 patas

Si bien cada actor tiene un rol definido, es necesario un trabajo conjunto entre los 3 actores. Es decir, es necesario que trabajen como equipo. Todos son igualmente importantes; por eso, utilizamos la analogía con una mesa de 3 patas: si falta alguna de ellas, la mesa no podría estar de pie.

## Business Analyst (BA)

Se encarga de detectar los factores clave del negocio y es el intermediario entre el departamento de sistemas y el cliente final.

## Software developer (Dev)

Su función es diseñar, producir, programar o mantener componentes o subconjuntos de software conforme a especificaciones funcionales y técnicas para ser integrados en aplicaciones.

## Quality Assurance (QA)

La principal función es probar los sistemas informáticos para que funcionen correctamente de acuerdo a los requerimientos del cliente, documentar los errores encontrados y desarrollar procedimientos de prueba para hacer un seguimiento de los problemas de los productos de forma más eficaz y eficiente

# Ciclo de vida de las pruebas de software (STLC)

Tiene como objetivo la entrega de un producto de calidad, a través de la mejora continua de sus procesos.

## Planificación

* Determinar el alcance, los objetivos y los riesgos.
* Definir el enfoque y estrategia general.
* Integrar y coordinar las actividades a realizar durante el ciclo de vida del software.
* Definir las especificación de técnicas, tareas de prueba adecuadas, las personas y otros recursos necesarios.
* Establecer un calendario de pruebas para cumplir con un plazo límite.
* Generar el plan de prueba.

La salida consiste en el plan de prueba (general y/o por nivel de prueba).

## Seguimiento y control

* Comprobar los resultados y los registros de la prueba en relación con los criterios de cobertura especificados.
* Determinar si se necesitan más pruebas dependiendo del nivel de cobertura que se debe alcanzar.

La salida consiste en el informe de avance de la prueba y las acciones correctivas en caso de ser necesario (como cambiar la prioridad de las pruebas, el calendario, reevaluar los criterios de entrada y salida, etc.).

## Análisis

Se determina qué se va a probar.

* Analizar la base de prueba correspondiente al nivel de prueba considerado.
* Identificar defectos de distintos tipos en las bases de prueba (ambigüedades, omisiones, inconsistencias, inexactitudes, etc.).
* Identificar los requisitos que se van a probar y definir las condiciones de prueba para cada requisito.
* Capturar la trazabilidad entre la base de prueba y las condiciones de prueba.

La salida consiste en los contratos de prueba que contienen las condiciones de prueba.

## Diseño

Se determina cómo se va a probar.

* Diseñar y priorizar casos de prueba y conjuntos de casos de prueba de alto nivel.
* Identificar los datos de prueba necesarios.
* Diseñar el entorno de prueba e identificar la infraestructura y las herramientas necesarias.
* Capturar la trazabilidad entre la base de prueba, las condiciones de prueba, los casos de prueba y los procedimientos de prueba

La salida consiste en los casos de prueba de alto nivel diseñados y priorizados.

## Implementación

Se completan los productos de prueba necesarios para la ejecución de la prueba, incluyendo la secuenciación de los casos de prueba en procedimientos de prueba.

* Desarrollar y priorizar procedimientos de prueba.
* Crear juegos de prueba (test suite) a partir de los procedimientos de prueba.
* Organizar los juegos de prueba dentro de un calendario de ejecución.
* Construir el entorno de prueba y verificar que se haya configurado correctamente todo lo necesario.
* Preparar los datos de prueba y asegurarse de que estén correctamente cargados.
* Verificar y actualizar la trazabilidad entre la base de prueba, las condiciones de prueba, los casos de prueba, los procedimientos de prueba y los juegos de prueba.

La salida consiste en los procedimientos y datos de prueba, el calendario de ejecución y las test suite.

## Ejecución

Se ejecutan los casos de prueba.

* Registrar los identificadores y las versiones de los elementos u objetos de prueba.
* Ejecutar y registrar el resultado de las pruebas.
* Comparar los resultados reales con los resultados esperados.
* Informar sobre los defectos en función de los fallos observados.
* Repetir las actividades de prueba, ya sea como resultado de una acción tomada para una anomalía o como parte de la prueba planificada —retest o prueba de confirmación—.
* Verificar y actualizar la trazabilidad entre la base de prueba, las condiciones de prueba, los casos de prueba, los procedimientos de prueba y los resultados de la prueba.

La salida consiste en reportes de defectos y el informe de ejecución de pruebas.

## Conclusión

Se recopilan la información de las actividades completadas y los productos de prueba. Puede ocurrir cuando un sistema software es liberado, un proyecto de prueba es completado (o cancelado), finaliza una iteración de un proyecto ágil, se completa un nivel de prueba o se completa la liberación de un mantenimiento.

* Comprobar que todos los informes de defecto están cerrados.
* Finalizar, archivar y almacenar el entorno de prueba, los datos de prueba, la infraestructura de prueba y otros productos de prueba para su posterior reutilización.
* Traspaso de los productos de prueba a otros equipos que podrían beneficiarse con su uso.
* Analizar las lecciones aprendidas de las actividades de prueba completadas.
* Utilizar la información recopilada para mejorar la madurez del proceso de prueba.

La salida consiste en el informe resumen de prueba y lecciones aprendidas.

# Niveles de Prueba

## Prueba de componente o unitaria

Se enfoca en un único componente o clase que desempeña una función específica. En general la lleva a cabo el desarrollador que escribió el código.

Base de prueba: diseño detallado, código, modelo de datos, especificaciones de los componentes.

Objetos de prueba: componentes, unidades o módulos, código y estructuras de datos, clases, módulos de base de datos.

## Prueba de integración

Se centra en las interacciones entre clases, componentes o sistemas. En general es responsabilidad del tester/QA.

Base de prueba: diseño de software y sistemas, diagramas de secuencia, especificaciones de interfaz y protocolos de comunicación, casos de uso, arquitectura a nivel de componente o de sistema, flujos de trabajo, definiciones de interfaces externas

Objetos de prueba: interacción entre clases, subsistemas, bases de datos, infraestructura, interfaces, APIs, microservicios.

## Prueba de sistema o end to end (E2E)

Debe centrarse en el comportamiento global del sistema en su conjunto, tanto funcional como no funcional. En general la llevan a cabo probadores independientes.

Base de prueba: especificaciones de requisitos del sistema y del software (funcionales y no funcionales), informes de análisis de riesgo, casos de uso, épicas e historias de usuario, modelos de comportamiento del sistema, diagramas de estado, manuales del sistema y del usuario.

Objetos de prueba: aplicaciones, sistemas hardware/software, sistemas operativos, sistema sujeto a prueba (SSP), configuración del sistema y datos de configuración.

## Prueba de aceptación

Se verifica que el comportamiento y las capacidades del sistema cumplan con los requerimientos. En general es responsabilidad de los clientes, usuarios de negocio o propietarios de producto.

Base de prueba: procesos de negocio, requisitos de usuario o de negocio, normativas, contratos legales y estándares, casos de uso, requisitos de sistema, documentación del sistema o del usuario, procedimientos de instalación, informes de análisis de riesgo.

Objetos de prueba: configuración del sistema, datos de configuración, procesos de negocio para un sistema totalmente integrado, sistemas de recuperación y sitios críticos, informes.

# Tipos de Pruebas

Un tipo de prueba es un grupo de actividades de pruebas destinadas a probar las características específicas de un sistema de software, o de una parte de un sistema, basados en objetivos de pruebas específicas.

## Prueba funcional

Pruebas que evalúan las funciones que el sistema debe realizar. La prueba funcional observa el comportamiento del software.

Se evalúan características como la completitud, corrección y pertinencia.

## Prueba no funcional

La prueba no funcional prueba “cómo de bien” se comporta el sistema.

El diseño y ejecución de la prueba funcional puede implicar competencias y conocimientos especiales, como el conocimiento de las debilidades inherentes a un diseño o tecnología.

Se evalúan características como la fiabilidad, eficiencia de desempeño, seguridad, confiabilidad y usabilidad.

# Gestión de defectos

## Errores, Defectos y Fallas

Error: es la equivocación de una persona.

Defecto: es algo incorrecto en el software debido a un error humano.

Falla: se produce en el sistema debido a la ejecución de software defectuoso.

## Ciclo de vida de un defecto

Estados: Nuevo → Asignado → En progreso → Corregido → En espera de verificación → En verificación → Verificado → Cerrado

Niveles de severidad: bloqueado, crítico, alto, medio, bajo o trivial.

Niveles de prioridad: alta, media, baja.

Cuando se detecta un defecto (como parte de las pruebas estáticas) o alguien encuentra una falla (como parte de las pruebas dinámicas), se reporta un defecto. Cuando “desarrollo” arregla un error, QA escribe un caso de prueba para verificar que el mismo fue corregido. Como resultado se puede determinar que el mismo no fue solucionado (vuelve a desarrollo), determinar que fue arreglado correctamente o encontrar otro error, lo que lleva al QA a reportar otro defecto.

## Reporte de defectos

| Id | Título | Descripción | Resultado actual | Resultado esperado | Pasos para reproducción | Estado | Prioridad | Severidad | Reportado por | Asignado a |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Inconsistencia en formato de hora AM/PM | Al momento de calcular la tarifa tipo Valet Parking, el sistema no mantiene la selección de formato AM/PM, reiniciando el valor del formato de hora ingresado. | Al presionar el botón Calculate, los valores para los datos AM/PM se reinician. | Al presionar el botón Calculate, el sistema debería mantener el valor del formato AM/PM seleccionado. | 1. Seleccionar Valet Parking  2. Ingresar fecha de entrada = 06/17/2021  3. Ingresar hora de entrada = 11  4. Seleccionar AM para hora de entrada  5. Ingresar fecha de salida = 06/17/2021  6. Ingresar hora de salida = 12  7. Seleccionar PM para hora de salida  8. Click en el botón Calculate | Asignado | Baja | Baja | Ian Vaernet | Ian Vaernet |

Campos adicionales: fecha de informe del defecto, identificación del elemento de prueba, versión, entorno, referencias, imagen.

# Casos de prueba

Un caso de prueba es un documento escrito que proporciona información sobre qué cómo probar.

## Testing positivo y negativo

* Testing positivo: son aquellos casos de prueba que validan el flujo normal de un sistema bajo prueba. Es decir, flujos que están relacionados a los requisitos funcionales del sistema bajo prueba.
* Testing negativo: son aquellos casos de prueba que validan flujos no contemplados dentro de los requisitos de un sistema bajo prueba (¿Qué pasa si hago x cosa? ¿Se obtiene un comportamiento indeseado?)
* Happy path testing: es un tipo de testing positivo que consiste en probar la aplicación a través de un único camino, que es el que debería realizar un usuario normalmente. Su propósito es el de verificar que la aplicación funcione de acuerdo a lo diseñado.

## Caso de uso

Un caso de uso cuenta la historia de cómo un usuario interactúa con un sistema de software para lograr o abandonar un objetivo. Cada caso de uso puede contener múltiples rutas que el usuario sigue, estos caminos son denominados escenario de caso de uso.

Se puede comenzar escribiendo casos de prueba para el “escenario principal” primero y luego escribirlos para “escenarios alternativos”. Es decir, se escribe uno o más casos de prueba por cada escenario de caso de uso. Los “pasos” de los casos de prueba se obtienen de la secuencia normal o alternativa detallada en los casos de uso.

Tanto el “nombre” como las “precondiciones” del caso de prueba se pueden basar directamente en los mismos campos que existen en el caso de uso. La capacidad para crear casos de prueba a partir de los casos de uso y hacer la traza de unos a otros es una habilidad vital para asegurar un producto de calidad.

Esto se conoce como pruebas de casos de uso y es una técnica de caja negra donde se verifica si la ruta utilizada por el usuario está funcionando según lo esperado o no.

## Caso de prueba

Un caso de prueba cubre el software más en profundidad y con más detalle que un caso de uso. Estos incluyen todas las funciones que el programa es capaz de realizar y deben tener en cuenta el uso de todo tipo de datos de entrada/salida, cada comportamiento esperado y todos los elementos de diseño.

| Id | Título | Descripción | Precondición | Pasos | | | Estado | Reportado por |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Acción | Resultado esperado |
| 1 | Validación de persistencia formato AM/PM | Validar que para el cálculo de una hora de estacionamiento en formato valet parking, los valores AM/PM persisten en el sistema. | La página Parking Cost Calculator debe ser cargada correctamente | 1 | Seleccionar Valet Parking | Valet Parking debe quedar seleccionado en el sistema | In progress | Ian Vaernet |
| 2 | Ingresar fecha de entrada = 06/17/2021 | La fecha de entrada debe quedar seleccionada |
| 3 | Ingresar hora de entrada = 11 AM | El horario de entrada debe quedar seleccionado |
| 4 | Ingresar fecha de salida = 06/17/2021 | La fecha de salida debe quedar seleccionada |
| 5 | Ingresar hora de salida = 12 PM | El horario de salida debe quedar seleccionado |
| 6 | Click en botón Calculate | El valor calculado debe ser correcto y los valores previos seleccionados deben persistir en el sistema |

## Test suites

Las test suite son un conjunto de casos de prueba, una agrupación de ellos.

El objetivo de todos los casos de prueba en una suite de humo es probar que se cumplen los requerimientos básicos, los más importantes.

Los casos de prueba de una suite de humo son un subconjunto de la suite de regresión, que involucra no solo a esos requerimientos más importantes sino a todos los necesarios para validar el correcto funcionamiento de la aplicación y asegurar que los cambios no produzcan errores.

Primero se lleva a cabo la ejecución de las pruebas de humo. Si alguna de ellas no pasa, ni se ejecutan las pruebas de regresión. Si todas pasan, se ejecutan las pruebas de regresión que toman un tiempo considerablemente mayor.

Una suite de pruebas consiste en un listado de los casos de prueba incluidos:

| Id Caso de prueba | Título |
| --- | --- |
|  |  |
|  |  |

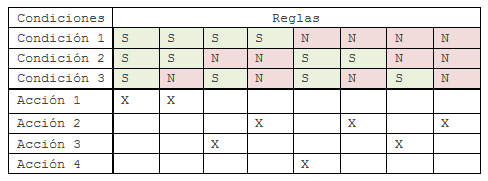
# Técnicas de Pruebas

El objetivo de una técnica de prueba es ayudar a identificar las condiciones, los casos y los datos de prueba.

## Técnicas de caja negra

Se basan en el comportamiento extraído del análisis de los documentos que son base de prueba (documentos de requisitos formales, casos de uso, historias de usuario, etc). Son aplicables tanto para pruebas funcionales como no funcionales. Se concentran en las entradas y salidas sin tener en cuenta la estructura interna.

* Prueba de casos de uso: se escribe uno o más casos de prueba por cada escenario de caso de uso. Los “pasos” de los casos de prueba se obtienen de la secuencia normal o alternativa detallada en los casos de uso
* Partición de equivalencia: se dividen los datos en particiones conocidas como clases de equivalencia donde cada miembro de estas clases o particiones es procesado de la misma manera. Características:
  + La “partición de equivalencia válida” contiene valores que son aceptados por el componente o sistema.
  + La “partición de equivalencia no válida” contiene valores que son rechazados por el componente o sistema.
  + Se pueden dividir las particiones en subparticiones.
  + Cada valor pertenece a solo una partición de equivalencia.
  + Las particiones de equivalencia no válidas deben probarse en forma individual para evitar el enmascaramiento de fallos.
* Análisis de valores límites: es una extensión de la técnica de partición de equivalencia que solo se puede usar cuando la partición está ordenada y consiste en datos numéricos o secuenciales. Se pueden utilizar 2 o 3 valores límites. Para 2 valores límites se toma el valor que marca el límite (valor que corresponde a la partición válida), y el valor anterior o posterior que corresponda a la partición de equivalencia inválida. Para 3 valores límites se toma el valor que marca el límite, un valor anterior y otro posterior a ese límite.
* Tabla de decisión: se utiliza para pruebas combinatorias, formadas por reglas de negocio complejas que un sistema debe implementar. Características:
  + Se deben identificar las condiciones (entradas) y las acciones resultantes (salidas). Estas conforman las filas de la tabla.
  + Las columnas de la tabla corresponden a reglas de decisión. Cada columna forma una combinación única de condiciones y la ejecución de acciones asociadas a esa regla.
  + Los valores de las condiciones y acciones pueden ser valores booleanos, discretos, numéricos o intervalos de números.



## Técnicas de caja blanca

Se basan en la estructura extraída de los documentos de arquitectura, diseño detallado, estructura interna o código del sistema. Se concentran en el procesamiento dentro del objeto de prueba.

## Técnicas basadas en la experiencia

Aprovechan el conocimiento de desarrolladores, testers y usuarios para diseñar, implementar y ejecutar las pruebas.

* Predicción de errores: se utiliza para anticipar la ocurrencia de equivocaciones, defectos y fallos basados en el conocimiento del probador. Se crea una lista teniendo en cuenta cómo ha funcionado la aplicación en el pasado, equivocaciones comunes en los desarrolladores y fallos en aplicaciones relacionadas. En base a esa lista se diseñan pruebas que expongan esos fallos y defectos.
* Prueba exploratoria: se diseñan, ejecutan, registran y evalúan de forma dinámica pruebas informales durante la ejecución de la prueba. Los resultados de estas pruebas se utilizan para aprender más sobre el funcionamiento del componente o sistema. Generalmente se utilizan para complementar otras técnicas formales o cuando las especificaciones son escasas o con restricciones de tiempo.
* Prueba basada en listas de comprobación: se diseñan, implementan y ejecutan casos de prueba que cubren las condiciones que se encuentran en una lista de comprobación definida. Se crean basadas en la experiencia y conocimiento de lo que el probador cree que es importante para el usuario y se utilizan debido a la falta de casos de prueba detallados. Se utiliza tanto en pruebas funcionales como no funcionales.

# Análisis de la prueba

## Pruebas estáticas

Se basa en la evaluación manual de los productos de trabajo (revisiones) o en la evaluación basada en herramientas del código u otros productos de trabajo (análisis estático). Algunos de los productos de trabajo a revisar o analizar son:

* Especificaciones, requisitos de negocio, funcionales y de seguridad.
* Épicas, historias de usuarios y criterios de aceptación.
* Especificaciones de arquitectura y diseño.
* Código.
* Productos de prueba: planes, casos, procedimientos y guiones de prueba.
* Manuales de usuario.
* Contratos, planes de proyecto, calendarios y presupuestos.

Defectos encontrados con pruebas estáticas:

* Defectos en los requisitos (inconsistencias, ambigüedades, etc.).
* Defectos de diseño (estructura de base de datos ineficiente, alto acoplamiento, etc.).
* Defectos de codificación (variables con valores no definidos, código inalcanzable o duplicado, etc.).
* Desviaciones con respecto a estándares (falta de uso de estándares de codificación).
* Especificaciones de interfaz incorrectas (unidades de medida diferente, etc.).
* Vulnerabilidades de seguridad (susceptibilidad a desbordamiento de la memoria intermedia).
* Diferencias o inexactitudes en la trazabilidad o cobertura de la base de prueba (falta de pruebas para un criterio de aceptación).
* Defectos de mantenibilidad (mala reutilización de componentes, modularización inadecuada, etc.).

## Pruebas dinámicas

Las pruebas dinámicas requieren la ejecución del software, componente o sistema. Se complementan con las pruebas estáticas debido a que encuentra diferentes tipos de defectos.

Fallas más comunes encontradas con este tipo de pruebas:

* Fallas de funcionalidad.
* Fallas de interacción entre módulos.
* Fallas de rendimiento y seguridad.

| **Pruebas estáticas** | **Pruebas dinámicas** |
| --- | --- |
| Detecta los defectos en productos de trabajo. | Detecta los defectos y fallas cuando se ejecuta el software. |
| Se centra en mejorar la consistencia y la calidad de los productos de trabajo. | Se centra en los comportamientos visibles desde el exterior. |
| El costo de solucionar un defecto es menor | El costo de solucionar un defecto es mayor |

## Proceso de revisión

Las revisiones consisten en examinar cuidadosamente un producto de trabajo con el principal objetivo de encontrar y remover errores. Pueden ser realizadas por una o más personas. Pueden ser:

* Revisiones formales: Tienen roles definidos, siguen un proceso establecido y deben ser documentadas.
* Revisiones Informales: No siguen un proceso definido y no son documentadas formalmente.

## Requisitos

Un requisito define las funciones, capacidades o atributos intrínsecos de un sistema de software, es decir, describe cómo debe comportarse un sistema. Para decir que un sistema tiene calidad deben cumplirse los requisitos funcionales y no funcionales.

* Requisitos funcionales: definen lo que un sistema permite hacer desde el punto de vista del usuario. Estos requisitos deben estar especificados de manera explícita. Ejemplo: El campo de monto acepta únicamente valores numéricos con dos decimales (pruebas funcionales y de sistema).
* Requisitos no funcionales: Definen condiciones de funcionamiento del sistema en el ambiente operacional. Ejemplos:
  + Requisito de usabilidad: es el esfuerzo que necesita hacer un usuario para aprender, usar, ingresar datos e interpretar los resultados obtenidos de un software de aplicación (pruebas de usabilidad).
  + Requisito de eficiencia: desempeño en cuanto al tiempo de respuesta, número de operaciones por segundo y consumo de recursos (pruebas de rendimiento, pruebas de carga, estrés y escalabilidad, pruebas de gestión de la memoria, compatibilidad e interoperabilidad).
  + Requisito de disponibilidad: disposición del sistema para prestar un servicio correctamente (pruebas de disponibilidad).
  + Requisito de confiabilidad: continuidad del servicio prestado por el sistema (pruebas de seguridad).
  + Requisito de integridad: ausencia de alteraciones inadecuadas al sistema (pruebas de seguridad, pruebas de integridad).
  + Requisito de mantenibilidad: posibilidad de realizar modificaciones o reparaciones a un proceso sin afectar la continuidad del servicio (pruebas de mantenimiento y de regresión).

# Ambientes y sus pruebas asociadas

## 

## DEV

* Pruebas unitarias o de componente: tiene como objetivo encontrar defectos en el componente y verificar que los comportamientos funcionales y no funcionales del componente son los diseñados y especificados.
* Pruebas de Integración: los objetivos incluyen encontrar defectos en las propias interfaces o dentro de los componentes o sistemas y verificar que los comportamientos funcionales y no funcionales de las interfaces sean los diseñados y especificados.

## QA (Quality Assurance)

* Pruebas funcionales: incluye pruebas que evalúan las funciones que el sistema debe realizar.
* Pruebas de sistema: Se centra en el comportamiento y las capacidades de todo un sistema o producto, a menudo teniendo en cuenta las tareas extremo a extremo que el sistema puede realizar y los comportamientos no funcionales que exhibe mientras realiza esas tareas.
* Pruebas de casos de uso: proporcionan pruebas transaccionales, basadas en escenarios, que deberían emular el uso del sistema.
* Pruebas de exactitud: comprenden el cumplimiento por parte de la aplicación de los requisitos especificados o implícitos y también puede abarcar la exactitud de cálculo.
* Pruebas de adecuación: implican evaluar y validar la eficiencia de un conjunto de funciones para la consecución de las tareas especificadas previstas. Estas pruebas pueden basarse en casos de uso.
* Pruebas de Humo: se lleva a cabo para verificar que las funciones más importantes de un programa están trabajando correctamente, pero sin molestarse con los detalles más finos.
* Pruebas de regresión: pruebas para detectar efectos secundarios no deseados, luego de cambios hechos en una parte del código que puedan afectar accidentalmente el comportamiento de otras partes del código.
* Pruebas de confirmación: consiste en volver a ejecutar los pasos para reproducir el fallo o los fallos causados por un defecto en la nueva versión de software, una vez corregido el defecto, para así confirmar que el defecto original se ha solucionado satisfactoriamente o detectar efectos secundarios no deseados.
* Pruebas de cordura: es una prueba de regresión acotada que se centra en una o unas pocas áreas de funcionalidad. Se utiliza para determinar si una pequeña sección de la aplicación sigue funcionando después de un cambio menor.

## UAT (User Acceptance Testing)

* Pruebas de aceptación: se centra normalmente en el comportamiento y las capacidades de todo un sistema. Además, pueden producir información para evaluar el grado de preparación del sistema para su despliegue y uso por parte del cliente (usuario final).
* Pruebas exploratorias: Se diseñan, ejecutan, registran y evalúan de forma dinámica pruebas informales (no predefinidas) durante la ejecución de la prueba. Los resultados de la prueba se utilizan con el objetivo de aprender más sobre el componente o sistema y crear pruebas para las áreas que pueden necesitar ser probadas con mayor intensidad.
* Pruebas de usabilidad: evalúan la facilidad con la que los usuarios pueden utilizar o aprender a utilizar el sistema para lograr un objetivo específico en un contexto dado.
* Pruebas de accesibilidad: incluyen y evalúan la accesibilidad que presenta un software para aquellos con necesidades particulares o restricciones para su uso. Esto incluye a aquellos usuarios con discapacidades.

## STG (Stage)

* Pruebas de mantenimiento: se centra en probar los cambios en el sistema, así como en probar las piezas no modificadas que podrían haberse visto afectadas por los cambios. El mantenimiento puede incluir lanzamientos planificados y no planificados.
* Pruebas de seguridad: conjunto de actividades que se llevan a cabo para encontrar fallas y vulnerabilidades en el sistema, buscando disminuir el impacto de ataques y pérdida de información importante.
* Pruebas de rendimiento: evalúan las características relacionadas con el rendimiento del destino de la prueba, como los perfiles de tiempo, el flujo de ejecución, los tiempos de respuesta y la fiabilidad y los límites operativos.
* Pruebas de carga, estrés y escalabilidad: una prueba de carga garantiza que un sistema pueda controlar un volumen de tráfico esperado. Una prueba de estrés es en la que se somete al sistema a condiciones de uso extremas para garantizar su robustez y confiabilidad. Las pruebas de escalabilidad garantizan la escalabilidad de un sistema, es decir, que pueda soportar el incremento de demanda en la operación. También se pueden realizar en QA encontrando el correspondiente escalar con respecto a un ambiente de PROD.
* Pruebas de infraestructura: incluyen todos los sistemas informáticos internos, los dispositivos externos asociados, las redes de Internet, la nube y las pruebas de virtualización.
* Pruebas de gestión de la memoria: evalúan el estado y la integridad de la memoria del sistema para identificar problemas potenciales.
* Pruebas de compatibilidad: incluyen las pruebas para comprobar que el sistema es compatible con todos los navegadores de Internet y todos los sistemas operativos del mercado.
* Pruebas de interoperabilidad: se realiza la evaluación de la correcta integración entre distintos aplicativos, sistemas, servicios o procesos que conforman una plataforma o solución tecnológica.
* Pruebas de migración de datos: incluyen las pruebas realizadas al transferir datos entre tipos de dispositivos de almacenamiento, formatos o sistemas de cómputo.

## PROD

No se ejecutan pruebas en el ambiente de producción.

# Métricas y reportes

El uso de métricas permite a los probadores comunicar resultados de una manera consistente y llevar a cabo un seguimiento coherente del progreso a lo largo del tiempo. Existen cinco dimensiones primarias sobre las que se monitoriza el progreso de las pruebas:

* Riesgos (de la calidad) del producto
* Defectos
* Pruebas
* Cobertura
* Confianza

## Métricas del proyecto

Miden el progreso hacia los criterios de salida previstos del proyecto, tales como el porcentaje de casos de prueba ejecutados, pasados y fallidos.

## Métricas del producto

Miden algunos atributos del producto, tales como el alcance en el que ha sido probado o la densidad del defecto.

* Tasa de detección de errores: determina la efectividad de los casos de prueba.
* Tasa de corrección de defectos: ayuda a conocer la calidad del software en términos de reparación de defectos.
* Densidad de defectos: relación entre defectos y requisitos.
* Fugas por defecto: revisa la eficiencia del proceso de prueba antes de UAT.
* Eficiencia de eliminación de defectos: permite comparar la eficiencia general de eliminación de defectos (defectos encontrados antes y después de la entrega).

## Métricas del proceso

Miden la capacidad del proceso de pruebas de desarrollo, tales como el porcentaje de defectos detectados por las pruebas.

* Productividad en la preparación de casos de prueba: se utiliza para calcular el número de casos de prueba preparados y el esfuerzo invertido en la preparación de casos de prueba.
* Cobertura de diseño de prueba: ayuda a medir el porcentaje de cobertura de casos de prueba frente al número de requisitos.
* Productividad de ejecución de prueba: determina el número de casos de prueba que se pueden ejecutar por hora.
* Cobertura de ejecución de pruebas: mide el número de casos de prueba ejecutados contra el número de casos de prueba planeados.
* Casos de prueba pasados, fallidos y bloqueados: mide el porcentaje de los casos de prueba pasados, fallidos y bloqueados.

## Métricas de las personas

Miden la capacidad de los individuos o los grupos, tales como la implementación de casos de prueba dentro de un calendario dado.

## Informes de avance de prueba

Son emitidos periódicamente por el jefe de prueba, durante la monitorización y el control de prueba. Suelen incluir:

* El estado de las actividades de prueba y el avance con respecto al plan de prueba.
* Factores que impiden el avance.
* Pruebas previstas para el próximo período objeto del informe.
* La calidad del objeto de prueba.

## Informe del resumen de la prueba

Es emitido por el jefe de prueba cuando se alcanzan los criterios de salida. Suelen incluir:

* Resumen de la prueba realizada
* Información sobre lo ocurrido durante un período de prueba.
* Desviaciones con respecto al plan, incluye los cambios en el calendario, la duración o el esfuerzo de las actividades de prueba.
* Estado de la prueba y calidad del producto con respecto a los criterios de salida o definición de hecho.
* Factores que han bloqueado o continúan bloqueando el avance.
* Métricas de defectos, casos de prueba, cobertura de la prueba, avance de la actividad y consumo de recursos.
* Riesgos residuales.
* Productos de trabajo de la prueba reutilizables desarrollados.

Kahoot

1. ¿Cuál de las siguientes no es un principio de testing?

La prueba exhaustiva es posible

2. En la mesa de 3 patas:

Todos los roles son igual de importes

3. Testing negativo contempla los escenarios que siempre fallan.

Falso

4. Happy path hace referencia a situaciones en donde todo funciona correctamente en un sistema.

Falso

5. Partición de equivalencia es una técnica de:

Caja negra

6. Prueba exploria es una técnica de:

Basada en la experiencia

7. Los ambientes de desarrollo más comunes son:

DEV, QA, UTV, STG, PROD

8. Los procesos de revisión pueden ser:

Formales o informales (se hace una charla o documento formal)

9. Dentro de un proyecto de automatización lo ideal es comenzar con:

Suite de regresión (porque cubren todas las funcionalidades del sistema, son largas y por eso son las candidatas a automatizarse)

10. Las pruebas de humo nos ayuda a:

Confirmar que las funcionalidades principales no sufrieron fallas

11. Pruebas de regresión:

Confirman que todo lo que funcionaba sigue funcionando de la misma forma

12. Los niveles de prueba son:

Componente, aceptación, sistema e integración