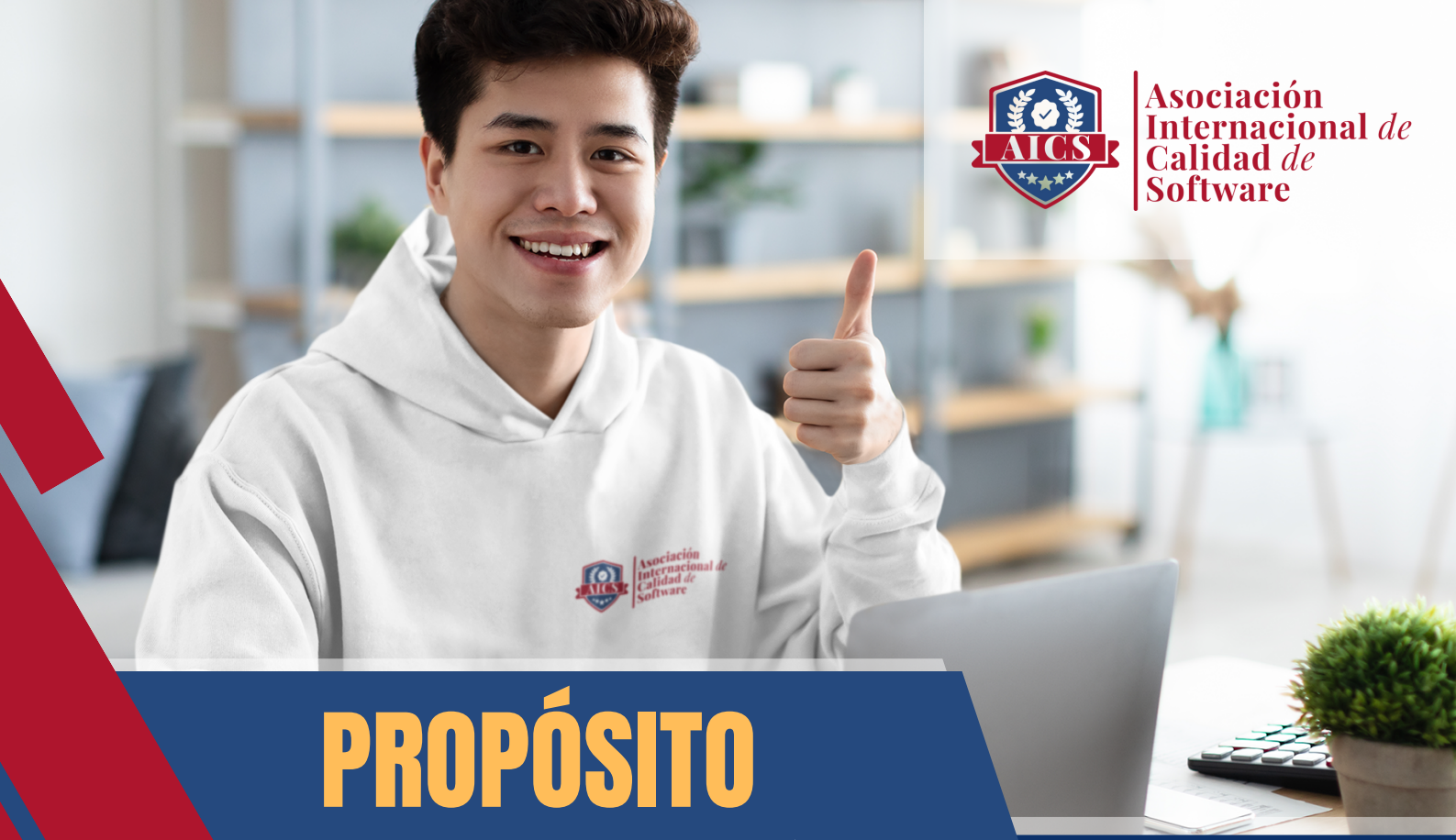
Asociación Internacional de Calidad de Software

GUÍA RÁPIDA DE FUNDAMENTOS DEL TESTING™

PRIMERA EDICIÓN

M.S. Lionel Baquero

aicsvirtual.org



Asociación Internacional de Calidad de Software

PROPÓSITO DE ESTA GUÍA

Bienvenido a nuestra guía

La Guía rápida de Fundamentos del Testing™ tiene como propósito proporcionar a los aspirantes de certificaciones de la Asociación Internacional de Calidad de Software™ un material de estudio completo y útil para comprender los principios básicos del testing. A través de esta guía, los aspirantes podrán conocer las técnicas, metodologías y herramientas necesarias para llevar a cabo una evaluación eficaz del software. Además, esta guía proporcionará una visión general de los objetivos y beneficios del testing, así como de los diferentes tipos de pruebas y sus aplicaciones. En resumen, esta guía es una herramienta esencial para aquellos que buscan mejorar sus conocimientos en el campo del testing y obtener una certificación reconocida en la industria del desarrollo de software.

TABLA DE CONTENIDOS

ESTA GUÍA SE ENCUENTRA ACTUALMENTE EN CONSTRUCCIÓN

INTRODUCCIÓN AL SOFTWARE TESTING

¿Qué es el Software Testing?

El testing es esencial en el desarrollo y mantenimiento de sistemas de software, teniendo como objetivo evaluar la calidad del software y reducir el riesgo de fallos en su funcionamiento. El testing incluye un conjunto de actividades, que permiten detectar defectos en el software, verificar si cumple con los requisitos especificados, validar que cumple con las necesidades de los usuarios y otros interesados. El testing es crucial para garantizar que el software funcione correctamente y satisfaga las necesidades de los usuarios.

Nota importante

El testing no se limita a la simple ejecución de pruebas y la verificación de resultados, sino que involucra una serie de actividades complementarias y críticas para garantizar la calidad del software y reducir el riesgo de fallos en su funcionamiento. Además, el testing puede realizarse en diferentes etapas del ciclo de vida del software y su importancia varía según el contexto y los requisitos del proyecto.

Objetivos del Software Testing

Los objetivos del testing de software son diversos, pero en general, el objetivo principal es asegurar que el software sea confiable y cumpla con los estándares de calidad necesarios para su uso en producción. El testing de software es una actividad crítica dentro del proceso de desarrollo de software, que tiene como objetivo evaluar el software para detectar posibles errores o defectos.

A continuación, se describen algunos de los objetivos más importantes del testing de software:

- **Garantizar la calidad del software:** El objetivo más importante del testing de software es garantizar la calidad del software, es decir, asegurar que el software cumpla con los requerimientos y especificaciones establecidos, y que sea confiable y seguro para su uso en producción.
- **Reducir los riesgos de fallos:** El testing de software también tiene como objetivo reducir los riesgos de fallos en el software. Los fallos en el software pueden causar pérdidas financieras y de reputación, por lo que es importante detectarlos y corregirlos antes de que ocurran en producción
- **Mejorar la fiabilidad del software:** El testing de software también tiene como objetivo mejorar la fiabilidad del software, es decir, garantizar que el software funcione correctamente y de manera consistente en diferentes situaciones y condiciones.

- **Validar los requerimientos del software:** El testing de software también tiene como objetivo validar los requerimientos del software, es decir, comprobar que el software cumpla con los requerimientos y especificaciones establecidos por el cliente o el usuario final.
- **Identificar y corregir errores tempranamente:** El testing de software también tiene como objetivo identificar y corregir errores tempranamente en el proceso de desarrollo de software. Es más fácil y menos costoso corregir errores tempranamente en el proceso de desarrollo que en producción.

Beneficios del Software Testing

El testing de software es una inversión que ofrece muchos beneficios a largo plazo. Entre sus principales beneficios se encuentran:

Reducción de costos

Las pruebas de software permiten detectar y corregir errores en las primeras etapas del ciclo de vida del software, lo que evita que los errores se propaguen y se conviertan en problemas costosos de resolver más adelante. Además, el testing de software ayuda a identificar problemas de diseño y a mejorar la eficiencia del software, lo que también contribuye a reducir los costos de desarrollo y mantenimiento.

Aumento de la satisfacción del cliente

Cuando el software funciona correctamente y se ajusta a las necesidades del usuario, la satisfacción del cliente aumenta.

Mejora de la imagen de la empresa

Cuando se entrega un software de calidad y se cumplen las expectativas del cliente, se genera una imagen positiva de la empresa y se fortalece la relación con los clientes. Por el contrario, un software defectuoso o con problemas puede dañar la reputación de la empresa y generar pérdida de confianza en los clientes.

Reducción de riesgos

Al realizar pruebas, se pueden identificar y corregir problemas que pueden afectar la seguridad del software o la integridad de los datos. Esto contribuye a proteger la empresa y a reducir los riesgos asociados con el software.

Mejora de la calidad del software

Al realizar pruebas, se pueden identificar y corregir problemas que pueden afectar la seguridad del software o la integridad de los datos. Esto contribuye a proteger la empresa y a reducir los riesgos asociados con el software.

Nota importante

El testing de software ofrece beneficios a largo plazo como reducción de costos, aumento de satisfacción del cliente, mejora de imagen de la empresa, reducción de riesgos y mejora de calidad del software.

Principios del Software Testing

El testing de software se basa en un conjunto de principios fundamentales que guían el proceso de prueba. Estos principios ayudan a los profesionales de pruebas a asegurarse de que el software se prueba de manera efectiva y se cumplan los objetivos de calidad.

Los 7 principios del testing de software son:

1

La detección temprana de defectos es más rentable

Es importante encontrar defectos en las etapas tempranas del ciclo de vida del software, ya que esto ayuda a reducir los costos y los riesgos a largo plazo.

2

La exhaustividad de las pruebas es imposible

Es imposible probar todas las combinaciones posibles de entradas y condiciones de operación del software. Por lo tanto, las pruebas deben centrarse en las áreas críticas del software y en los casos de uso más importantes.

3

La prueba muestra la presencia de defectos, no su ausencia

No es viable probar todas las combinaciones posibles de entradas y acciones del usuario en un software. Por lo tanto, se debe utilizar una combinación de técnicas de prueba para asegurar una cobertura adecuada y una alta calidad del software.

4

Los defectos tienden a agruparse

Los defectos tienden a agruparse en ciertas áreas o módulos del software. Es frecuente que aparezcan en grupos.

5

La paradoja del pesticida

Siempre que se utiliza un conjunto de pruebas fijo, eventualmente se agotará su capacidad para encontrar nuevos defectos.

6

La ausencia de errores es una falacia

Nunca se puede demostrar que un software está libre de errores, pero se puede demostrar que el software tiene un riesgo aceptable de fallos.

7

La importancia de la dependencia del contexto

El enfoque de las pruebas debe ser adaptado al contexto específico del software, teniendo en cuenta factores como la complejidad del software, los objetivos de calidad, los plazos y el presupuesto disponibles.

Nota importante

Seguir los principios del testing, puede ayudar a mejorar significativamente la calidad del software y aumentar la satisfacción del cliente.

Diferencias entre errores, defectos y fallas

En el ámbito del software testing, es esencial comprender las diferencias entre errores, defectos y fallas, ya que estos términos son fundamentales para identificar y gestionar problemas en un software. A continuación se explican detalladamente cada uno de estos conceptos:

Error

- Un error se refiere a una acción humana que produce un resultado incorrecto o inesperado en el software. Puede ser un error de diseño, un error de codificación o incluso un error en los requisitos del software.
- Los errores son cometidos por las personas que trabajan en el desarrollo del software, como diseñadores, desarrolladores o analistas. Estos errores pueden ser accidentales o simplemente el resultado de una comprensión incorrecta de los requisitos del cliente.
- Un error es la raíz de los problemas en el software, y su detección y corrección temprana son cruciales para evitar defectos y fallas posteriores.

Defecto

- Un defecto, también conocido como "bug", es una imperfección o un problema en el código o en el diseño del software que provoca que el software no funcione como se espera.
- Los defectos son el resultado de errores humanos (errores de diseño, codificación, etc.) y pueden surgir durante cualquier fase del ciclo de desarrollo de software.

- Los defectos se identifican mediante pruebas de software y otras técnicas de revisión y validación.

Falla

- Una falla es la manifestación visible y mensurable de un defecto en el software cuando el programa se ejecuta. En otras palabras, es cuando el software no se comporta como se espera en un entorno real.
- Las fallas son lo que los usuarios finales experimentan como problemas reales al utilizar el software. Pueden incluir bloqueos, mensajes de error, resultados incorrectos, comportamientos inesperados, entre otros.
- Las fallas son el resultado de defectos que no se han detectado y corregido antes de que el software se ponga en producción.

En resumen

- Los errores son acciones humanas que conducen a problemas potenciales en el software.
- Los defectos son problemas reales en el código o diseño del software que pueden detectarse durante las pruebas.
- Las fallas son problemas observados por los usuarios finales cuando utilizan el software en un entorno de producción.

Diferencias entre QA y QC

El Aseguramiento de Calidad (QA, por sus siglas en inglés) y el Control de Calidad (QC) son aspectos integrales para garantizar productos o servicios de calidad, especialmente en el desarrollo de software. Aunque están relacionados, existen diferencias fundamentales que separan estos dos procesos. Comprender estas diferencias es clave para la gestión efectiva y la entrega de software de alta calidad.

Enfoque

El QA se enfoca en establecer procesos preventivos para verificar el software en desarrollo, alineándose con los requerimientos del cliente y evitando problemas mediante procesos sólidos. Mientras tanto, el QC se concentra en medidas correctivas, examinando y verificando las características del producto para garantizar su desarrollo acorde a los requisitos definidos, donde las pruebas son esenciales.

Relación y Estructura Jerárquica

Es importante tener en cuenta que las pruebas de software son un subconjunto del control de calidad, que a su vez es un subconjunto del aseguramiento de calidad. El QA abarca un alcance más amplio, enfocándose en la implementación de procesos, mientras que el QC se adentra en los aspectos de ejecución y verificación, que incluyen las pruebas, entre otras medidas.

Quality Assurance VS Quality Control

Quality Assurance	Quality Control
Más amplio. Incluye procesos, estándares y metodologías.	Más específico. Se enfoca en la inspección y pruebas.
Garantizar que los procesos sean adecuados para producir productos de calidad.	Verificar que el producto cumpla con los estándares de calidad.
Enfoque preventivo. Se centra en prevenir problemas mediante procesos sólidos y estándares.	Enfoque correctivo. Se enfoca en identificar y corregir problemas existentes.

En resumen

En esencia, mientras que el Aseguramiento de la Calidad se centra en prevenir problemas mediante la implementación de procesos sólidos, el Control de Calidad interviene para asegurar que el producto se alinee precisamente con los requisitos establecidos. Ambos son indispensables para ofrecer software de alta calidad.

Diferencias entre QA y Testing

QA (*Quality Assurance*, Aseguramiento de la Calidad) y Testing (pruebas) son dos conceptos diferentes pero relacionados en el proceso de desarrollo de software. A continuación, se detallan las principales diferencias entre QA y testing:

1. **Objetivo:** QA se enfoca en la calidad del proceso de desarrollo de software, mientras que el testing se enfoca en la calidad del producto de software.

2. **Alcance:** QA abarca todo el ciclo de vida del desarrollo de software, desde la planificación hasta la implementación, mientras que el testing se limita principalmente a la fase de pruebas.

3. **Enfoque:** QA se enfoca en establecer estándares, procesos y procedimientos para garantizar la calidad en todo el ciclo de vida del software, mientras que el testing se enfoca en ejecutar pruebas para identificar y reportar defectos en el software.

4. **Responsabilidad:** QA es responsabilidad del equipo de desarrollo de software, mientras que el testing es responsabilidad del equipo de pruebas.

5

Resultado: QA garantiza que se sigan los estándares de calidad en todo el proceso de desarrollo de software, lo que lleva a un producto final de alta calidad, mientras que el testing identifica los defectos en el software para que puedan ser corregidos antes del lanzamiento.

Quality Assurance VS Software Testing

Quality Assurance	Software Testing
Orientado al proceso	Orientado al producto
Centrado en el control de la calidad y el cumplimiento de los requisitos	Centrado en la inspección del sistema y la detección de defectos
Enfoque preventivo	Enfoque correctivo
Tiene como objetivo garantizar la calidad	Tiene como objetivo controlar la calidad

Relación entre Testing y Debugging

En el universo del desarrollo de software, la relación entre testing y debugging (depuración) trasciende la mera distinción de actividades. Aunque ambos procesos se consideran separados, su colaboración es esencial para garantizar la robustez y la calidad del software resultante.

Testing como Desencadenante

- Las pruebas, tanto dinámicas como estáticas, juegan un papel crucial en la identificación de defectos. Las pruebas dinámicas pueden provocar fallos al descubrir defectos durante la ejecución del software, mientras que las pruebas estáticas identifican defectos directamente en el objeto de prueba.

Proceso de Depuración

- Cuando las pruebas dinámicas desencadenan un fallo, la depuración se enfoca en la identificación, análisis y eliminación de las causas subyacentes del fallo. Este proceso implica la reproducción controlada del fallo, el diagnóstico para encontrar la causa raíz y la aplicación de correcciones para resolver la cuestión. La prueba de confirmación posterior verifica la eficacia de estas correcciones.
- En el caso de las pruebas estáticas, que identifican defectos directamente, la depuración se simplifica, ya que la identificación del defecto conlleva su eliminación inmediata.

Perspectiva Comparativa

- Es vital distinguir entre testing y depuración. El testing descubre defectos, mientras que la depuración elimina esos defectos del sistema. El proceso de depuración comienza después de que el testing ha concluido, y el equipo de desarrollo inicia la búsqueda de las causas subyacentes de los defectos informados.
- La depuración es un proceso de resolución de problemas existentes. No es sabio abordar un defecto sin un conocimiento completo de todas las posibles causas. La depuración se inicia una vez que se comprende plenamente la naturaleza del defecto, evitando así la introducción inadvertida de nuevos problemas en módulos interrelacionados.
- La interconexión entre testing y depuración se manifiesta en un ciclo continuo. Las pruebas descubren, la depuración corrige, y la retroalimentación mejora las estrategias de testing. La colaboración efectiva entre estos procesos es esencial para lograr un software fiable y de alta calidad.

En resumen

En resumen, la relación entre testing y depuración es un vínculo dinámico que impulsa la evolución constante del software hacia niveles superiores de calidad y confiabilidad.

Software Testing VS Debugging

Software Testing	Debugging
Tiene como objetivo identificar defectos	Tiene como objetivo identificar y corregir defectos
Enfoque preventivo, busca evitar defectos	Enfoque correctivo, se centra en corregir
El propósito es verificar y validar el software	El propósito corregir causas subyacentes de defectos

Verificación y Validación

En el vasto universo del software testing, la verificación y la validación son dos dimensiones esenciales que, aunque interrelacionadas, abordan aspectos distintos en la búsqueda de la calidad del software. Examinemos de cerca estas dos prácticas, destacando sus diferencias y cómo se complementan en la creación de productos de software sólidos.

Verificación: La Construcción Correcta del Software

- La verificación se refiere al proceso de evaluación y confirmación de que un producto de software, sistema o componente cumple con las especificaciones y requisitos establecidos durante su desarrollo.

- Este proceso abarca diversas actividades, que incluyen pruebas unitarias, revisiones de código, pruebas de documentación y otras verificaciones para asegurarse de que cada elemento del software esté en conformidad con los estándares establecidos.

Validación: La Relevancia y Utilidad del Software

- La validación, por otro lado, se enfoca en confirmar que el software entregado satisface las necesidades y expectativas reales del usuario final.
- Este enfoque asegura que el software no solo esté construido correctamente, sino que también sea relevante y valioso para aquellos que lo utilizarán.

En resumen:

En la búsqueda de la calidad del software, la verificación y la validación son dos caras de la misma moneda. La verificación construye la base técnica, asegurando la corrección interna, mientras que la validación se asegura de que el software entregado sea relevante y valioso para el usuario final. Un enfoque equilibrado de ambas prácticas es esencial para garantizar un producto de software sólido y funcional.

> Aunque distintas, la verificación y la validación están entrelazadas a lo largo del ciclo de vida del desarrollo de software. La verificación constante asegura que cada ajuste cumpla con las normas, mientras que la validación intensifica su enfoque hacia el usuario final en las etapas finales del desarrollo.

Verificación VS Validación

Verificación	Validación
Evaluar si se construyó correctamente	Evaluar si se construyó lo correcto
Cumplimiento de requisitos y estándares	Asegurar la satisfacción del cliente
Revisar el proceso de construcción del software	Evaluar el producto final / incremento
Asegurar la calidad del proceso de desarrollo	Asegurar la satisfacción del usuario final

Actividades y Tareas de Pruebas

El proceso de pruebas en el desarrollo de software implica una serie de actividades esenciales que se ejecutan de manera interrelacionada para garantizar la efectividad y calidad del producto final. Aunque estas actividades suelen seguir una secuencia lógica, su implementación puede ser iterativa o simultánea, adaptándose a las necesidades específicas del sistema y del proyecto.

- **Planificación de Pruebas:** La base de un proceso de pruebas sólido radica en una planificación detallada. Esto implica definir objetivos de pruebas claros y seleccionar la estrategia más adecuada para alcanzarlos, considerando las limitaciones y contextos generales del proyecto.
- **Monitoreo y Control de Pruebas:** La supervisión constante de todas las actividades de pruebas y la comparación del progreso real con el plan establecido son fundamentales. Además, el control de pruebas implica tomar acciones para cumplir con los objetivos de las pruebas, asegurando así su efectividad.
- **Análisis de Pruebas:** Este paso involucra el análisis de la base de pruebas para identificar las características que pueden ser probadas, así como la definición y priorización de condiciones de prueba asociadas y sus riesgos. También se evalúa la posibilidad de defectos en los objetos de prueba y su testabilidad.
- **Diseño de Pruebas:** El diseño detalla cómo se van a ejecutar las pruebas, desde la elaboración de condiciones de prueba hasta la creación de casos de prueba y otros elementos de pruebas. Esto incluye la definición de requisitos de datos de prueba, diseño del entorno de pruebas y la identificación de la infraestructura y herramientas necesarias.

- **Implementación de Pruebas:** La implementación incluye la creación de pruebas necesarias para su ejecución, como la adquisición de datos de prueba, la organización de casos de prueba en procedimientos de prueba y la creación de scripts de prueba, tanto manuales como automatizados.
- **Ejecución de Pruebas:** Esta etapa implica la ejecución de las pruebas según un calendario establecido. Las pruebas pueden ser manuales o automatizadas, y se comparan los resultados obtenidos con los esperados. Se analizan las anomalías para identificar sus posibles causas y se registra toda la información relevante.
- **Cierre de Pruebas:** Al llegar a hitos del proyecto, se llevan a cabo actividades de cierre que incluyen la resolución de defectos pendientes, la identificación y archivado de elementos de prueba útiles para el futuro, y el análisis de lecciones aprendidas para mejorar futuros proyectos.

Roles en Equipos de Pruebas de Software

Cada empresa define su propia jerarquía y configuración de roles en sus equipos de pruebas de software. Sin embargo, en un nivel general, se encuentran algunos roles comunes:

Líder/Gerente de Pruebas (Test Lead/Manager)

El rol de gestión de pruebas tiene una misión fundamental: tomar las riendas del proceso de pruebas. Este rol asume la responsabilidad de:

- Definir y planificar las actividades de pruebas para el equipo.
- Garantizar que el equipo tenga todos los recursos necesarios.
- Supervisar la alineación entre las pruebas y el desarrollo de software.
- Generar informes de estado para las actividades de pruebas.
- Mantener una comunicación efectiva tanto con el equipo como con los clientes.

Ingeniero de Pruebas (Test Engineer)

Los profesionales encargados de la ejecución directa de las pruebas tienen la responsabilidad de:

- Comprender y analizar los documentos para identificar los elementos a probar.
- Colaborar con el líder de pruebas para determinar los recursos necesarios.
- Desarrollar casos de prueba y ejecutar las pruebas.
- Reportar y priorizar defectos, además de realizar pruebas de regresión.

Nota importante

Es importante mencionar que las personas que ocupan estos roles pueden variar. Por ejemplo, el gestor de pruebas podría ser un líder de equipo, un gerente de pruebas o incluso un gerente de desarrollo. Sorprendentemente, algunas veces una persona puede llevar a cabo ambos roles simultáneamente, demostrando una gran versatilidad y adaptabilidad.

INTRODUCCIÓN AL SOFTWARE TESTING

Test Manager	Test Engineer
Dirigir y coordinar todo el proceso de pruebas	Realizar pruebas técnicas, diseño y ejecución de pruebas
Enfoque en liderazgo y gestión de todo el equipo	Enfoque técnico en análisis y ejecución de pruebas
Toma decisiones estratégicas y de alto nivel	Implementa estrategias de pruebas según los requisitos
Diseña estrategias globales y planes de pruebas	Implementa planes de pruebas diseñados por la gestión
Requiere experiencia en gestión y liderazgo técnico	Enfocado en habilidades técnicas de prueba y ejecución.

Comprensión profunda del proceso de prueba y la calidad del software

Nota importante

En los equipos de prueba, estos roles pueden variar en sus nombres y responsabilidades según la estructura organizativa. En algunos equipos, estos roles pueden tener diferentes designaciones, como Gerente de Pruebas, Ingeniero de Pruebas, Analista de Calidad (QA Analyst), entre otros. Además, en ciertos entornos, estas funciones pueden estar fusionadas, donde una persona puede asumir tareas de gestión y técnicas de pruebas simultáneamente, adaptándose a las necesidades específicas del equipo y la organización.