Ejercicio 1 - Conceptualización de términos clave en Testing

1. Testing:

El testing es el proceso de revisar y probar un sistema o aplicación para verificar que funcione como se espera. En el desarrollo de software, su importancia es clave porque nos permite detectar errores antes de que el producto llegue al usuario. Así evitamos problemas graves en producción y aseguramos que el software cumpla con los requerimientos del cliente.

2. Visión antigua y actual del testing:

Antes se pensaba que el testing era solo una fase final donde simplemente se ejecutaban pruebas para ver si todo funcionaba. Hoy en día, se entiende como una parte integral del desarrollo desde el principio. Incluso se aplica el enfoque de *Testing temprano* (shift-left), donde los testers colaboran desde la etapa de diseño y análisis. También se considera una responsabilidad compartida entre todo el equipo.

3. Propósitos del testing:

El objetivo del testing no es solo encontrar errores, sino también validar que el sistema hace lo que debería y verificar que no haga lo que no debe. Busca garantizar la calidad, mejorar la confianza en el producto, reducir riesgos y asegurar que se cumplan los requisitos del cliente.

4. Actividades de un analista QA/Tester:

Un analista QA o tester no solo prueba. También participa en revisar requerimientos, diseñar casos de prueba, preparar datos, ejecutar pruebas manuales o automáticas, reportar errores, hacer seguimiento de los mismos, verificar correcciones y documentar evidencias. Además, colabora con el equipo de desarrollo para asegurar la calidad desde el inicio.

5. Validación y Verificación:

- **Verificación**: Se enfoca en revisar que el software *se esté construyendo bien*. Es decir, que se sigan los procesos y estándares correctos. Por ejemplo, revisar que los requisitos estén bien definidos o que el código cumpla las buenas prácticas.
- **Validación**: Se trata de comprobar que *el producto final cumple lo que el cliente espera*. Por ejemplo, hacer pruebas funcionales para ver si el sistema resuelve correctamente el problema del usuario.

6. Error, Falla y Defecto:

- **Error**: Es una equivocación humana, como escribir mal una línea de código.
- **Defecto**: Es el resultado de ese error dentro del software (por ejemplo, una función que devuelve mal un dato).
- **Falla**: Es cuando ese defecto se manifiesta durante la ejecución y el sistema no funciona correctamente (por ejemplo, se cae la aplicación).

7. Trazabilidad:
La trazabilidad permite seguir el rastro de los requisitos a lo largo de todo el ciclo del proyecto. Gracias a ella podemos ver qué pruebas validan cada requerimiento y qué partes del código están involucradas. Esto es vital para saber qué impacta un cambio, controlar la calidad y asegurarnos de que no falte probar nada.

Ejercicio 2 - Describir las fases del ciclo de vida del testing en los siguientes enfoques:

Tradicional: Explicar las etapas principales y su secuencia.

Ágil: Detallar cómo se integra el testing en metodologías ágiles y su enfoque iterativo.

Solución:

1. Enfoque Tradicional (Cascada):

En el enfoque tradicional, como el modelo en cascada, las fases del testing están bien definidas y siguen una secuencia lineal. Cada fase depende de la anterior, y no se avanza hasta terminar la anterior. Las fases principales del ciclo de vida del testing en este enfoque son:

- **Análisis de requerimientos:** El tester revisa los requisitos del sistema para entender qué se debe probar.
- **Planificación de pruebas:** Aquí se define el alcance, estrategia, tipos de pruebas, cronograma, recursos, y herramientas necesarias.
- **Diseño de casos de prueba:** Se crean los casos de prueba que describen cómo se validarán los requisitos.
- **Preparación del entorno de pruebas:** Se configura el entorno donde se va a ejecutar el testing (datos, acceso, infraestructura).
- **Ejecución de pruebas:** Se ejecutan los casos de prueba manual o automáticamente, y se registran los resultados.
- **Reporte y seguimiento de defectos:** Si se encuentran fallas, se reportan y se hace seguimiento hasta su solución.

- **Cierre de pruebas:** Al terminar, se documenta lo realizado, se valida la cobertura y se generan informes finales.

Este enfoque es más estructurado, pero tiene como desventaja que los errores se detectan tarde y cuesta más corregirlos.

2. Enfoque Ágil:

En metodologías ágiles como Scrum o Kanban, el testing no es una fase separada, sino una actividad continua que ocurre durante todo el ciclo de desarrollo. Las pruebas se integran desde el inicio, en ciclos cortos e iterativos (sprints). Las fases, aunque similares, se adaptan al flujo ágil:

- **Análisis colaborativo de requisitos (Backlog grooming):** QA participa desde el inicio, entendiendo los requerimientos junto al equipo de desarrollo y negocio.
- **Diseño de pruebas temprano:** Los testers diseñan casos de prueba desde que se escribe la historia de usuario, incluso antes de que exista código.
- **Testing continuo:** Se prueban las historias dentro del sprint, mientras se desarrollan. Puede incluir pruebas manuales y automatizadas.
- **Automatización e integración continua:** Las pruebas automáticas (unitarias, integración, regresión) se ejecutan constantemente junto con cada cambio al código.
- **Feedback rápido:** Se reportan los errores en el momento para corregirlos lo antes posible.
- **Revisión y retrospectiva:** Al final de cada sprint se revisa el trabajo y se mejora el proceso, incluyendo el testing.

El testing en ágil es más flexible, rápido y colaborativo, lo que permite detectar errores desde el primer momento y adaptarse mejor a cambios.