**Testing QA**

**¿Qué es el testing?** El testing se define como el proceso para verificar y validar las funcionalidades de un software con el fin de otorgar calidad mediante el mismo.

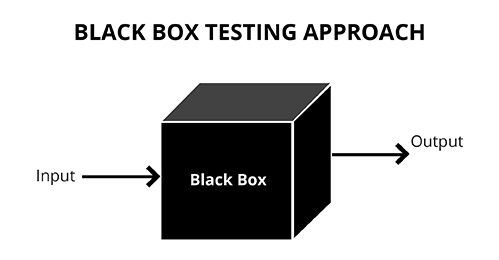
**¿Qué es QA?** Su definición al español es aseguramiento de calidad, y su objetivo es otorgar calidad mediante distintas técnicas de pruebas con el fin de asegurar que el software que se desarrolla cumple con un estándar.

**Tipos de pruebas:** Existe muchos tipos de pruebas para asegurar que el desarrollo salga con la mayor calidad posible, algunos de estos tipos son:

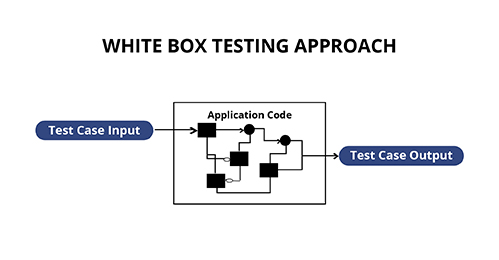
* Pruebas Unitarias: El objetivo de esta prueba es revisar cada una de las funciones o clases que componen el código del software (generalmente las ejecuta el desarrollador)
* Pruebas de integración: Se verifica que los distintos módulos o servicios de un software funcionen correctamente en conjunto.
* Pruebas Funcionales: Se valida que los requerimientos se cumplan en cuanto al funcionamiento ya sea un botón, una página, un texto, etc.
* Pruebas No funcionales: Las pruebas no funcionales nos permiten conocer que riesgos corre el producto y si tiene un mal desempeño en producción. (Pruebas de carga, pruebas de seguridad, pruebas de código fuente)
* Pruebas Integrales: Se valida el end to end de una aplicación web o Mobile con el objetivo de replicar el comportamiento de un usuario.
* Pruebas de aceptación: Se verifica que el sistema satisface los requisitos empresariales.
* Pruebas de rendimiento: Evalúan el rendimiento del sistema con una carga determinada. Ayudan a medir la fiabilidad, velocidad, escalabilidad y capacidad de respuesta de una aplicación.
* Pruebas de humo: Sirven para validar el funcionamiento básico de la aplicación.
* Pruebas UX: Son pruebas para asegurar aplicaciones user friendly e intuitivas.

**Enfoque de box testing:** Dentro de este enfoque existen dos metologias de pruebas las cuales se enfocan a distintas fases del desarrollo de software, estos enfoques son los siguientes:

* Caja negra: En este método se toman en cuenta las salidas, entradas y respuestas de un software pero sin entender el funcionamiento interno del mismo, es decir, son pruebas que se realizan en una interfaz gráfica.



* Caja blanca: Para este método de prueba se revisa el código y la estructura del producto con el fin de encontrar agujeros de seguridad internos, rutas rotas o mal estructuradas, rendimiento esperado, funcionalidad de bucles condicionales y funcionalidad de cada enunciado, objeto y función individualmente.



**Ciclo de vida de las pruebas de software:**Un ciclo de vida es la secuencia de cambios que atraviesa una entidad de una forma a otra.

**Fases del ciclo de vida del software testing:**

1. **Análisis del requerimiento**: Las actividades en esta fase incluye lluvia de ideas para el análisis de requisitos e identificación y priorización de requisitos de prueba. También incluyen la selección de requisitos para pruebas automáticas y manuales.

* Comprenda el resultado esperado del producto.
* Identifique las lagunas en las especificaciones.
* Recopile prioridades.
* Realice verificaciones de viabilidad de automatización.

1. **Planificación de pruebas**: El segundo paso es la planificación de la prueba y el equipo de control de calidad crea este plan después de analizar todos los requisitos de prueba necesarios. Describen el alcance y los objetivos después de comprender el dominio del producto. Luego, el equipo analiza los riesgos involucrados y define los horarios y los entornos de prueba para crear una estrategia.

* Prepare la documentación del plan de prueba.
* Estime tiempo y esfuerzos.
* Finalice en herramientas y plataforma.
* Asignar tareas a equipos e individuos.
* Identificar los requisitos de formación.

1. **Diseño y desarrollo de casos de prueba**: Basándose en el plan de prueba, los probadores diseñan y desarrollan casos de prueba. Los casos de prueba deben ser extensos y cubrir casi todos los casos posibles. Se deben recopilar todas las permutaciones y combinaciones aplicables. Puede priorizar estos casos de prueba investigando cuáles de ellos son más comunes o cuál de ellos afectaría más al producto.

* Investigar y recopilar posibles acciones sobre el producto.
* Crea casos de prueba.
* Priorice los casos de prueba

1. **Configuración de entorno de pruebas**: Las actividades de prueba necesitan ciertos factores ambientales, como servidores, marcos, hardware y software, para ejecutar casos de prueba desarrollados. La configuración de software y hardware, junto con la configuración de datos de prueba, son los componentes principales de esta fase. Y es obligatorio realizar pruebas de humo y equipar a los evaluadores con herramientas de notificación de errores.

* Comprender los requisitos mínimos.
* Enumere el software y el hardware necesarios para diferentes niveles de rendimiento.
* Priorice los entornos de prueba.
* Configurar entornos de prueba.
* Prueba de humo en los entornos construidos.

1. **Ejecución de pruebas**: Una aplicación está lista para probar una vez que el equipo haya terminado con todas las fases anteriores. Según el plan de prueba, los evaluadores ejecutan casos de prueba. También identifican, detectan y registran los defectos, informando así los errores. El equipo también es responsable de comparar los resultados esperados con el resultado real. Si se encuentra algún error, es necesario documentarlo para transmitirlo al equipo de desarrollo para que lo solucione.

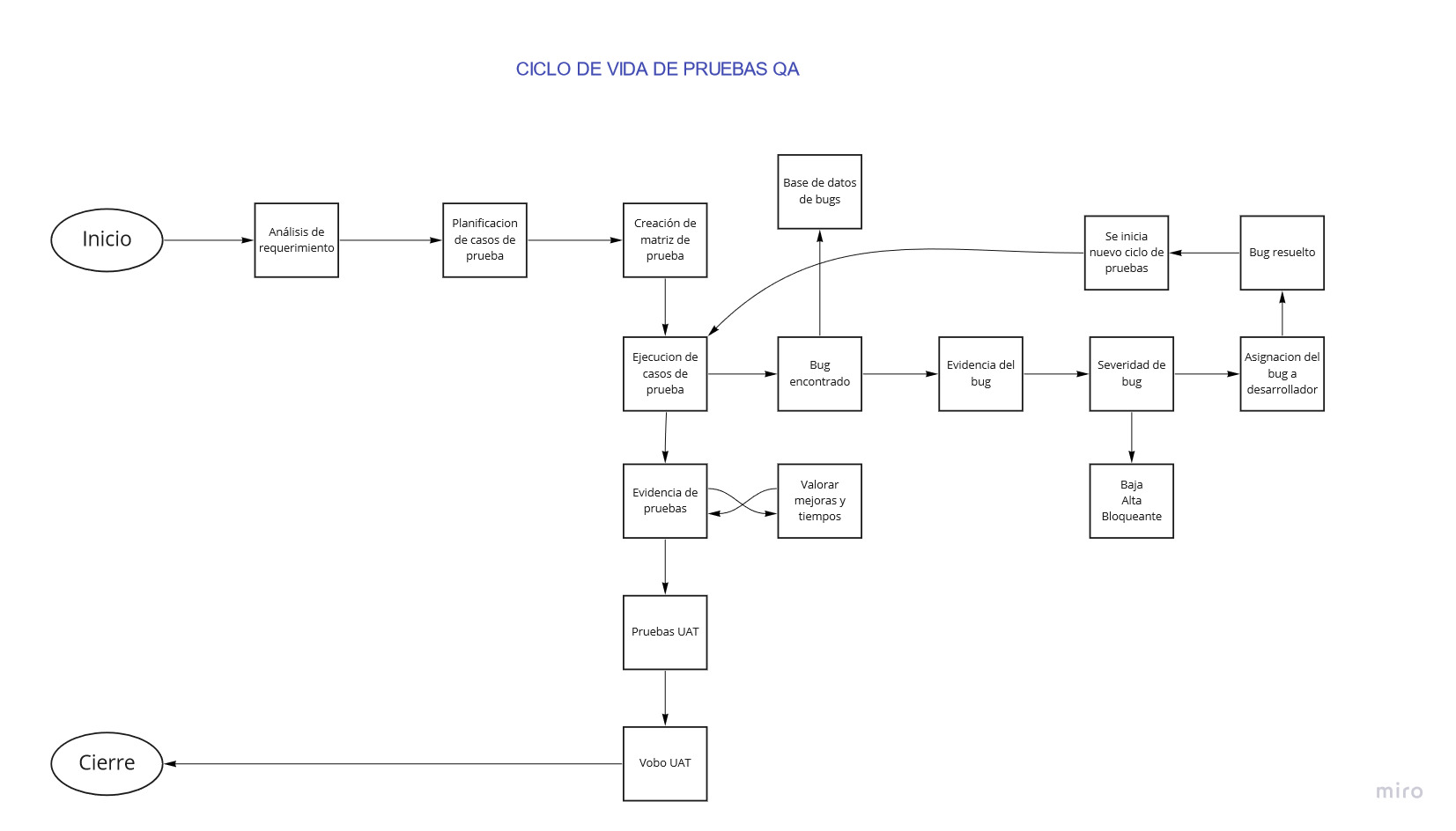
* Ejecute casos de prueba.
* Identificar la desviación del comportamiento esperado del producto.
* Registre casos fallidos con detalles.
* Pruebe de nuevo después de corregir los errores.

1. **Prueba de cierre**: El final de la ejecución de la prueba y la entrega del producto final marca el inicio de la fase de cierre de la prueba. El equipo de control de calidad verifica los resultados de la prueba y los analiza con otros miembros del equipo. Algunos otros factores que consideran son la calidad del producto, la cobertura de la prueba y el costo del proyecto. Si hay una desviación de los valores estimados, se pueden realizar más análisis para identificar lo que no salió como se esperaba.

* Verifique que se completen todas las pruebas.
* Evalúe factores como la calidad, la cobertura de la prueba, el cronograma y el costo.
* Documente la conclusión.
* Discuta el aprendizaje y averigüe si se puede mejorar el proceso de prueba.
* Prepare el informe de cierre de la prueba.

**Ambientes de pruebas:** Dentro del desarrollo de software existen distintos ambientes en los cuales se podrán realizar pruebas de acuerdo con la fase del desarrollo en el que se encuentre la aplicación. Estos son algunos de los ambientes utilizados durante del desarrollo de un software:

* Ambiente de desarrollo: En este ambiente el desarrollador genera sus cambios con el fin de revisar que funcionen de forma correcta.
* Ambiente QA/UAT: Este ambiente es considerado un espejo de producción ya que cuenta con la misma data que en producción, pero sin realizar un efecto real, generalmente en este ambiente se realizan las pruebas por parte de los testers.
* Ambiente de producción: Es el ambiente final en el cual el usuario interactúa día con día, no es recomendable realizar pruebas o cambios sobre este ambiente ya que afectarían el funcionamiento para el usuario final.



**Requerimientos de pruebas:** Se le conoce como un requerimiento de pruebas a todo aquel documento o tarea que especifica las funciones a probar, así como los resultados esperados y el alcance del requerimiento, es decir, cual es la limitante de las pruebas sobre la tarea asignada.

**Matriz de pruebas:** Se define como el concentrado de los casos o escenarios de prueba, es decir, es un documento donde se colocan todas las pruebas a realizar.

**Escenarios de pruebas:** Para escribir un escenario de pruebas solo es necesario comprender que debe realizar la aplicación, es decir, “Que se debe probar”.

**Casos de prueba:** Es un conjunto de condiciones bajo las cuales se valida que una parte determinada de la aplicación funciona de acuerdo con requerimientos, y para escribir test cases debemos tener precondiciones, pasos a reproducir, resultados esperados, datos de prueba, es decir, “Como se debe probar”.

**Defecto, problema o falla:** La ISTQB señala que la palabra defecto, problema y falla son sinónimos de bug, y que es preferible el uso de la palabra defecto sobre los otros términos a pesar de que el termino bug es más usado.

**Severidad de un defecto:** Cada error encontrado dentro de la funcionalidad de un software debe ser catalogado de acuerdo con la severidad que este presente, es decir, que tanto afecta este defecto encontrado al flujo de pruebas, esto no debe confundirse con prioridad aun que si va de la mano con este otro termino. La severidad de un defecto se cataloga de la siguiente manera:

* Baja: Se puede definir un defecto de baja severidad a todos aquellos que no afectan al flujo, pero si la calidad del software, un ejemplo de ello podrían ser las faltas de ortografía.
* Alta: Se define como todo aquel defecto que no realiza la función esperada por el requerimiento.
* Bloqueante: Se define como bloqueante a todo aquel defecto que no permite realizar un flujo completo de las pruebas, generando que no todos los componentes puedan ser testeados.

**Vo. Bo:** Este termino se refiere a un visto bueno por parte del usuario, el cual se otorga una vez aprobadas las pruebas UAT.

**UAT:** Se le conoce como pruebas de aceptación de usuario, es decir, son aquellas pruebas que realiza el usuario final con el fin de cerciorarse que el producto desarrollado sea lo que se solicito en una primera instancia.

**Prioridad de un defecto:** La prioridad de un defecto a diferencia de la severidad viene dada por el negocio y esta relacionada con la urgencia de la solución, es decir, si tenemos los siguientes ejemplos:  
  
Severidad Baja – Prioridad Alta

Severidad Alta – Prioridad baja

Esto quiere decir que no siempre un defecto de severidad alta debe ser catalogado con la misma prioridad, esto se debe a que probablemente al negocio no interese resolver ese defecto encontrado, pero si le interese resolver el de baja ya que de alguna manera afecta al usuario y por ende al negocio.

**Herramientas de testing manual**

Las herramientas para un tester son esenciales ya que permiten la correcta ejecución del trabajo, así como, el aumento de la velocidad del trabajo dependiente de la herramienta que se utiliza. Algunas de estas herramientas son:

* ScreenPresso: Esta herramienta sirve para capturar evidencia, ya sea en video o capturas de pantallas. También puede generar un documento con base en las capturas realizadas, lo que nos ahorra tiempo al entregar evidencias.
* Mysql: Las consultas a las bases de datos son primordiales por lo que esta herramienta nos ayudara a comprender de mejor manera si alguno de nuestros casos de prueba se cumple, por ejemplo, la creación de algún nuevo usuario, eliminación o actualización. Todo esto se debería reflejar en la base de datos.
* Browsers: Probar el mismo software con distintos browsers para asegurar que existe la compatibilidad con los existentes en el mercado.
* Herramientas de gestión de proyectos: Herramientas como Monday, Jira, Confluence entre otras nos ayudan a gestionar el numero de pruebas ejecutadas, así como sus estatus actuales y de esta forma tener un mejor seguimiento del proceso de pruebas.
* Postman: Esta herramienta nos permite realizar peticiones de API’S para cerciorarnos que las peticiones que se realicen no marquen algún error en un momento determinado.
* Miro: Se puede usar como un tablero para dibujar distintos procesos y tenerlos presentes siempre que sea necesario.
* OneNote: Esta herramienta se usa para generar notas o descripciones de procesos en forma de pasos, también se puede usar en conjunto con miro para complementar la información que se guarde en esta herramienta.