

Introducción a la unidad 5

En la unidad 5 de nuestro curso de Tester de Software se exploran diferentes tipos de pruebas para evaluar el software. Esto es crucial para mejorar la calidad, garantizar la funcionalidad, optimizar el rendimiento, aumentar la seguridad, ser eficiente en el desarrollo, cumplir normativas y ofrecer confianza al cliente.

- Mejorar la Cobertura de Pruebas: Cada tipo de prueba se enfoca en aspectos específicos del software, lo que permite una cobertura más completa.
- Asegurar la Funcionalidad Correcta: Las pruebas de funcionalidad, por ejemplo, garantizan que el software cumpla con sus requisitos y funcione según lo previsto. Comprender cómo llevar a cabo estas pruebas es esencial para confirmar que el software satisface las necesidades de los usuarios y las especificaciones del cliente.
- Optimizar el rendimiento: Las pruebas de rendimiento y carga son cruciales para evaluar cómo se comporta el software en condiciones de uso intensivo. Estudiar estos tipos de pruebas ayuda a identificar y resolver problemas de rendimiento, como tiempos de respuesta lentos o fallos bajo carga, lo que mejora la experiencia del usuario.
- Aumentar la Seguridad: Las pruebas de seguridad y vulnerabilidad son esenciales en un entorno de software cada vez más amenazado por ciberataques. Conocer cómo realizar pruebas de seguridad permite detectar y mitigar posibles brechas de seguridad antes de que puedan ser explotadas.
- Eficiencia en el Proceso de Desarrollo: Comprender cuándo y cómo aplicar cada tipo de prueba en el ciclo de desarrollo de software mejora la eficiencia del proceso. Esto significa que los problemas se identifican y corrigen más temprano en el ciclo, lo que reduce los costos y el tiempo de desarrollo.
- Cumplir con Estándares y Normativas: En muchas industrias, como la financiera o la médica, existen regulaciones estrictas que requieren pruebas específicas. Conocer los diferentes tipos de pruebas y sus requisitos es esencial para cumplir con estas normativas y evitar sanciones legales.
- Ofrecer Confianza al Cliente: Las pruebas exhaustivas y bien ejecutadas proporcionan una base sólida para la confianza del cliente. Los clientes quieren software confiable y seguro.

Conocer estos tipos de pruebas ayuda a los profesionales a enfrentar los desafíos de un entorno exigente. Se analizan pruebas de funcionalidad, rendimiento, seguridad, entre otras, para diseñar estrategias efectivas.

VIDEO 01

Pruebas funcionales:

- Se puede realizar en los diferentes niveles de prueba (Test de aceptación, Test de sistema, Test de integración, Test Unitario).
- Objetivo principal: El objeto de prueba debe funcionar correctamente.
- Para que sirve: Para verificar los requisitos funcionales, estos son los que están establecidos en las especificaciones (historias de usuarios, casos de uso de la regla de negocio, documento relacionado)
- Se prueban 5 características principales:
 - La interoperabilidad: Nos preguntamos.. ¿Las interacciones con el entorno del sistema presentan algún problema?
 - La exactitud: Nos preguntamos.. ¿Las funciones presentan los resultados correctos que hemos acordado con el cliente?
 - La adecuación: Nos preguntamos.. ¿Las funciones implementadas son adecuadas para su uso?
 - El cumplimiento de funcionalidad: Nos preguntamos.. ¿El sistema cumple con las normas y reglamentos que se deben aplicar en este caso?
 - La seguridad: Nos preguntamos.. ¿Tenemos los datos o programas protegidos contra accesos no deseados o contra pérdidas?

Pruebas no funcionales:

- Se puede realizar en los diferentes niveles de prueba (Test de aceptación, Test de sistema, Test de integración, Test Unitario).
- Objetivo principal: Cómo funciona un sistema, describiendo las pruebas necesarias para medir las características que se pueden cuantificar en una escala. EJ: el tiempo de respuesta en las pruebas de rendimiento.
- El problema: Es que la mayoría de los proyectos están especificados de manera muy escasa o no están especificados de manera adecuada.
- Que se prueba: Las pruebas de volumen, estabilidad, robustez, usabilidad, rendimiento, carga.
 - Volumen: Procesamiento de grandes cantidades de datos. ¿Qué pasa cuando mi sistema maneja gran cantidad de datos?
 - Estabilidad: ¿Qué sucede cuando el sistema trabaja en un modo de operación continua?
 - Robustez: Es la reacción de entrada erróneas. ¿Qué pasa cuando y cómo se recupera de ese desastre?
 - Usabilidad: Si es estructurado, comprensible y fácil de aprender para el usuario final.
 - Rendimiento: Rapidez del sistema cuando ejecuta determinada función.
 - Cargas: ¿Qué pasa cuando el sistema tiene que trabajar con una cierta carga? EJ: Si tiene muchos usuarios , transacciones, como reacciona ante esto.

Pruebas estructurales y asociadas a cambios:

Finalidad: Es medir el grado de la estructura del objeto de prueba, es decir la estructura del objeto que estamos probando si ha sido cubierto por los casos de prueba. Se las denomina pruebas de CAJA BLANCA, realizándose en los diferentes niveles de prueba pero sobre todo cuando estamos realizando el test unitario o componentes o pruebas de integración.

Pruebas relacionadas a cambios:

- Objetivo: es repetir una prueba de funcionalidad que haya sido verificada previamente, puede ser de confirmación o regresión.
 - Confirmación: Si detectamos un defecto y los desarrolladores ya lo arreglaron, el software tiene que ser probado de vuelta para corroborar que el defecto original haya sido solucionado con éxito. La confirmación se realiza cuando tenemos un defecto solucionado, y hay que corroborar que haya sido solucionado
 - Regresión: Conjunto de pruebas repetidas de un programa que ya ha sido probado pero que ha sufrido alguna modificación y sirve para determinar si se ha introducido algún nuevo error. La regresión es cuando ha habido un cambio del entorno o porque se instaló alguna corrección de algún defecto haciendo pruebas repetidas para determinar que no se haya introducido algún error adicional.

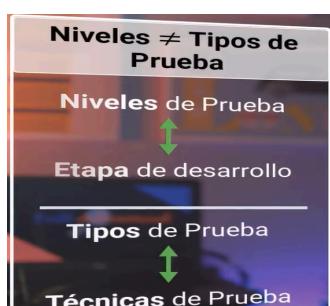
VIDEO 03

Tipos de pruebas de software:

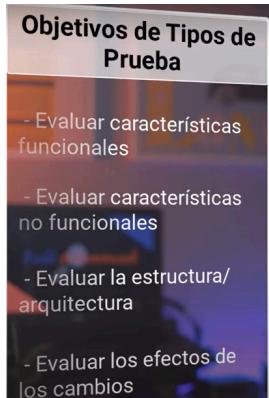
Una vez que hemos cubierto los niveles de pruebas podemos abordar los tipos de pruebas.

Todos los tipos de pruebas se pueden usar en todos los niveles de prueba, debemos identificar qué pruebas son más óptimas para que nos den buenos resultados.

las pruebas/técnicas de caja blanca: Revisamos documentos o el código fuente revisando defectos y pruebas de caja negra utilizadas para validar entradas y salidas durante las pruebas de aceptación



Los niveles son distintos a las tipas de prueba. Los niveles de prueba se definen por la etapa de desarrollo del software. Los tipos de prueba se relacionan con las técnicas para ejecutar las pruebas de cada nivel.



Tipos de prueba: Es un grupo de actividades que se centra en un objetivo en particular

- Funcionales de calidad, como integridad, corrección y oportunidad.
- No funcionales: confiabilidad, eficiencia de rendimiento, seguridad, compatibilidad y usabilidad.
- Estructura/arquitectura: Si es correcta, completa y como se especifica
- Efectos de los cambios: como confirmar que se han reparado los defectos, lo que sería la prueba de confirmación. También la búsqueda de cambios en el comportamiento resultantes de cambios en el software o entorno (prueba de regresión).

Pruebas funcionales: Las funciones son “QUE” debe hacer el sistema. Se basan en funciones descritas en documentos o entendidas por los probadores. Las pruebas funcionales deben realizarse en todos los niveles de prueba, aunque el enfoque es diferente en cada nivel. EJ:

1. Las pruebas de componentes pueden basarse en especificaciones del componente, mientras que las pruebas de integración se pueden basar en la verificación que se estén enviando los datos correctamente de un módulo a otro a través del uso de alguna interfaz.

Las pruebas funcionales pueden utilizar técnicas de caja negra para derivar condiciones de prueba y casos de prueba para la funcionalidad del componente o sistema.

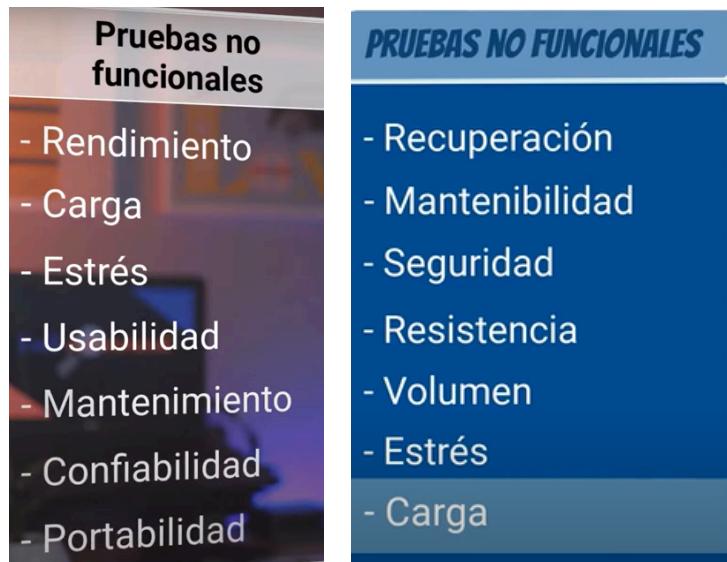


La minuciosidad de las pruebas funcionales se pueden medir a través de la cobertura funcional. Es una medida sobre cuánto se ha probado algún tipo de elemento funcional, y es expresado como un % del tipo de elemento cubierto.

El conocimiento de negocio sera de gran ayuda al ejecutar las pruebas funcionales.

Pruebas NO funcionales: Conocida también como pruebas de las características del producto de software.

- Evalúan que tan bien o que tan rápido se hace algo. Se prueba algo que necesite medirse en una escala, ej= Tiempo de respuesta del sistema.
- Se realizan en todos los niveles de prueba y deben ejecutarse en todo momento posible, ya que el descubrimiento tardío de este tipo de defectos no funcionales puede ser extremadamente peligroso para el éxito de un proyecto.



-Recuperación: Verifican que tan rápido y que tan bien se recupera una app luego de experimentar un fallo en el hardware o software.

-Mantenibilidad: Evalúan qué tan fácil es realizar el mto de un sistema o app, es decir que tan fácil es analizar, cambiar y probar estos cambios.

-Seguridad: Consisten en probar los atributos o características de seguridad del sistema, si un sistema es seguro o no, si puede ser vulnerado, si existe control de acceso por medio de cuentas de

usuario, si pueden ser vulnerados estos accesos.

-Resistencia: Implican someter a un sistema o aplicación a una carga determinada durante un período de tiempo, para determinar cómo se comporta luego de un uso determinado.

-Volumen: Validan el funcionamiento de la app con ciertos volúmenes de datos.

-Estrés: son pruebas de carga que se realizan con demandas mayores a la capacidad operativa, con frecuencia hasta llegar al punto de ruptura.

-Carga: Simulan una demanda sobre una app de software y medir el resultado.

Las técnicas de caja negra se pueden usar para crear condiciones de pruebas y casos de pruebas no funcionales.

Pruebas de Caja Blanca:

A partir de estas pruebas podemos derivar pruebas basadas en la estructura o en la implementación interna del sistema y medir la exhaustividad de las pruebas a través de la cobertura estructural. La estructura interna puede incluir código, arquitectura o flujos de datos dentro del sistema. La técnica esta se puede aplicar en cualquier nivel de prueba, lo más común es que se apliquen principalmente en los niveles de Componentes e integración.

Pruebas relacionadas a cambios:

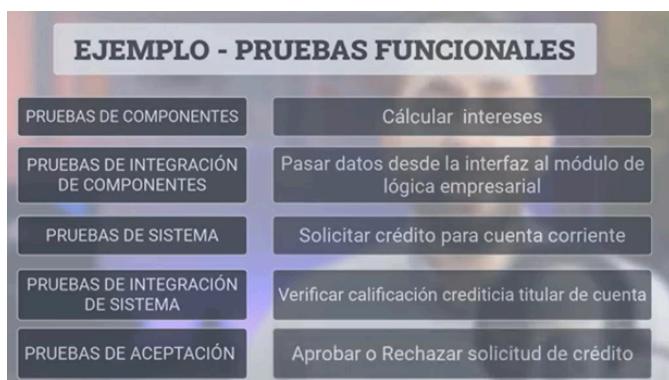
Prueba de confirmación: confirman si un defecto ha sido corregido correctamente. La nueva versión del software se puede probar mediante todos los casos de prueba que fallaron debido al defecto inicial, también se pueden probar con nuevos casos de pruebas. Al realizar esta prueba de confirmación, es importante asegurarse de que la prueba se ejecute exactamente de la misma manera que la primera vez, utilizando las mismas entradas datos y entornos.

"La forma de detectar 'efectos secundarios inesperados' de las soluciones es realizar pruebas de regresión"

Pruebas de Regresión: verifican que los cambios no hayan causado efectos secundarios no deseados en el software y que el sistema cumpla con los requisitos. Estas pruebas se ejecutan cada vez que el software cambia, ya sea como resultado de correcciones o funcionalidades nuevas o modificadas. También es buena idea ejecutarlas cuando cambia algún aspecto del entorno. Una funcionalidad en el sistema se deben agregar nuevas pruebas de regresión y al cambiar o eliminar la funcionalidad se debe eliminar o cambiar las pruebas de regresión, EJ= cuando se introduce una nueva versión de un sistema de adm de bases de datos o se usa una nueva versión de un compilador de código fuente.

Tipos de pruebas y Niveles de pruebas:

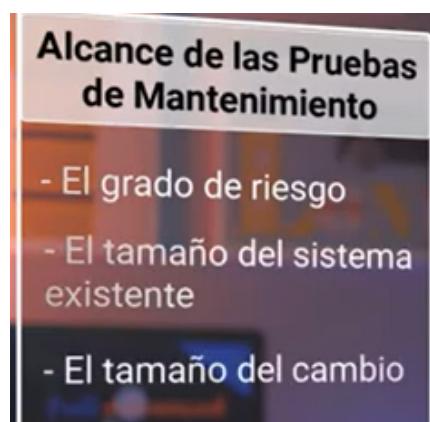
Ejemplos de usos en una app bancaria:



EJEMPLO - PRUEBAS DE CAJA BLANCA	
PRUEBAS DE COMPONENTES	Lograr cobertura total de sentencias y decisiones
PRUEBAS DE INTEGRACIÓN DE COMPONENTES	Pasar datos entre pantallas y módulos lógicos
PRUEBAS DE SISTEMA	Cubrir flujo de trabajo para solicitud de crédito
PRUEBAS DE INTEGRACIÓN DE SISTEMA	Consultas al servicio de puntaje de crédito
PRUEBAS DE ACEPTACIÓN	Validar estructura de archivos y rango de valores para transferencias entre bancos

EJEMPLO - PRUEBAS RELACIONADAS CON EL CAMBIO	
PRUEBAS DE COMPONENTES	Realizar pruebas de regresión automatizadas
PRUEBAS DE INTEGRACIÓN DE COMPONENTES	Realizar pruebas de confirmación relacionadas con la interfaz entre componentes
PRUEBAS DE SISTEMA	Realizar pruebas de regresión si alguna pantalla en el flujo de trabajo cambia
PRUEBAS DE INTEGRACIÓN DE SISTEMA	Realizar pruebas de regresión como parte del despliegue continuo
PRUEBAS DE ACEPTACIÓN	Realizar pruebas de confirmación

Pruebas de mantenimiento: El mantenimiento de sistemas es inevitable para reparar defectos, agregar funcionalidades, preservar la calidad y realizar pruebas de regresión. Se pueden identificar tres situaciones que desencadenan el mantenimiento: modificaciones, migración y retiro del sistema. El mantenimiento también es crucial para preservar o mejorar las características de calidad no funcionales del sistema, como eficiencia y rendimiento.

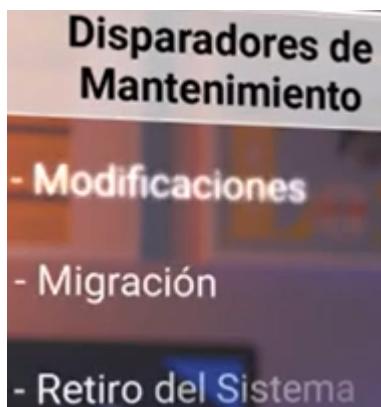


-El grado de riesgo: ej= el grado en que el área modificada del software se comunica con otro componentes o sistemas.

Las pruebas de mantenimiento son necesarias para evaluar el éxito de los cambios, verificar posibles efectos secundarios y realizar pruebas de regresión.

Una actividad importante dentro de estas pruebas es un análisis de impacto, se toma una decisión sobre qué partes del sistema pueden verse afectadas involuntariamente y por lo tanto necesitan pruebas de regresión cuidadosas. El análisis de riesgo ayudarán a decidir dónde enfocar las pruebas de regresión. Si se mantiene las especificaciones de prueba del desarrollo original del sistema, uno puede reutilizarlas para pruebas de regresión y adaptarlas para cambios en el sistema

Disparadores de Mantenimiento:



- **Modificadores:** incluimos cambios en el entorno operativo, ej= actualización de versión de un sistema operativo, de la base de datos, parches por defectos y vulnerabilidades.

- **Migración:** De una plataforma a otra, EJ? de un sistema operativo Windows a Linux, que puede requerir pruebas operativas del nuevo entorno así como del software modificado o pruebas de conversión de datos.

- **Retiro del sistema:** Cuando una app llega a su vida útil, puede requerir pruebas de migración o archivo de datos si se requieren largos periodos de retención de datos.



- Las modificaciones planificadas incluyen adaptaciones perfectas, adaptativas y correctivas, cada una con enfoques específicos para mantener la funcionalidad del sistema.

- **Perfectas:** adaptamos el software según lo indicado por el usuario, ej= proporcionando nuevas funciones o mejorando el rendimiento.

- **Adaptativas:** Adaptamos el software a los cambios ambientales, ej= hardware nuevo, software de sistemas nuevos o legislación nueva.

- **Correctivas planificadas:** El enfoque de prueba estructurada estándar es casi totalmente aplicable a las modificaciones planificadas. La mayoría de las modificaciones que enfrentaremos serán planificadas.



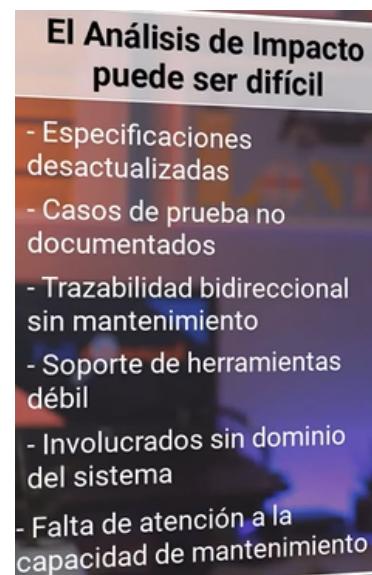
- Se refieren a defectos que requieren una solución inmediata, ej= una falla en proceso de producción. En una etapa posterior se sigue el proceso de prueba estándar para establecer una solución robusta, probarla y establecer el nivel adecuado de documentación.

Análisis de impacto para el mantenimiento:

Evalúa los cambios que se realizaron para una versión de mantenimiento para identificar las consecuencias tras su implementación, así como los efectos secundarios esperados y posibles de un cambio, e identificar las áreas del sistema que se verán afectadas directamente.

El análisis de impacto puede ayudar con la identificación del impacto de un cambio en las pruebas existentes

El análisis de impacto se puede realizar antes de un cambio, para apoyar la decisión sobre si se debe realizar el cambio o no.



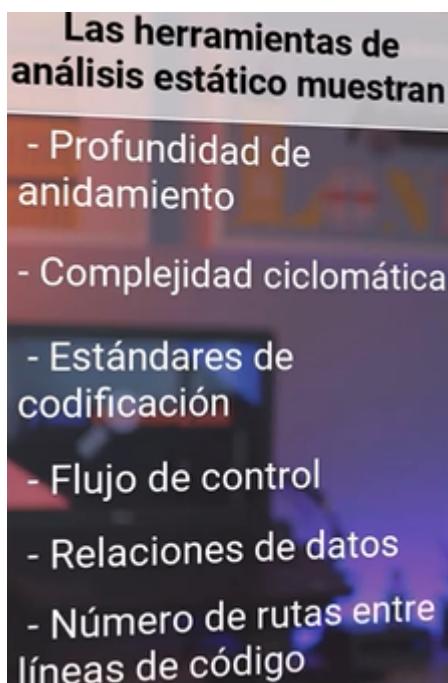
VIDEO 04

Pruebas estáticas

La importancia de las pruebas estáticas en el desarrollo de software se destaca, ya que ayudan a identificar errores en las etapas tempranas del ciclo de vida del sistema.

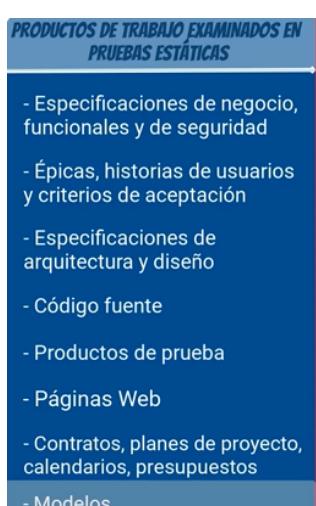
Conceptos básicos:

Se centran en la revisión de documentos y código fuente sin ejecutar. El análisis estático se utiliza para probar lenguajes formales y es beneficioso para los desarrolladores antes y durante de las pruebas de componentes e integración, mejorando la calidad del software.



El compilador también puede decirse que es una herramienta de análisis estático, ya que crea una tabla de símbolos, señala el uso incorrecto y comprueba el incumplimiento de la sintaxis del lenguaje de codificación que están utilizando.

El análisis estático es fundamental para garantizar la calidad y seguridad en el desarrollo de software. Se utiliza en sistemas críticos como los aeronáuticos, médicos y bancarios. Las herramientas de análisis estático permiten detectar errores de sintaxis y uso incorrecto del código, mejorando así la calidad del software. Estas herramientas son esenciales en el desarrollo ágil.



- Resumiendo las pruebas estáticas: Consisten en ejecutar revisiones y realizar analizar estáticas. Estas se pueden aplicar a cualquier producto de trabajo que los participantes sepan cómo leer y comprender.
 - El análisis estático puede ser aplicado a cualquier producto de trabajo con una estructura formal para el cual exista una herramienta de análisis estático apropiada, ej = código o modelos.
 - Análisis estático es aplicable mediante herramientas que evalúan los productos de trabajo escritos en lenguaje natural, como los requisitos, EJ= corrector ortográfico o gramática.
 - Herramientas más comunes en el análisis estático revisión de estándares de código , revisión de métricas y estructura del código.

Métricas

- Líneas de código
- Comentarios
- Anidamiento
 - Condiciones
 - Estructuras repetitivas

Las métricas del código son cruciales para entender su complejidad. Estas métricas ayudan a identificar áreas de riesgo y a enfocar esfuerzos de prueba en partes críticas.

Estructuras de código: Existen diferentes mediciones relacionadas al código, que nos indican qué tan complejo fue escribirlo, comprenderlo para modificarlo en el futuro o probarlo.

Estructura del código

- Flujo de control
- Flujo de datos
- Estrutura de datos

El flujo de control y de datos son elementos clave en el análisis de la estructura del código. Permiten detectar defectos y optimizar la ejecución de las instrucciones. La estructura de datos nos muestra cómo se organizan los datos en sí mismos, no tienen relación con la estructura del código o con el flujo de control.

Ventajas de las pruebas estáticas:

Las pruebas estáticas son fundamentales para detectar defectos en el software antes de las pruebas dinámicas. Esto permite ahorrar tiempo y dinero, mejorando la calidad del producto final. La identificación temprana de defectos es crucial, ya que corregir errores en fases iniciales es más económico que hacerlo después del despliegue. Esto resalta la importancia de las pruebas estáticas.

Son más baratas las pruebas estáticas que las pruebas dinámicas.

VENTAJAS ADICIONALES DE LAS PRUEBAS ESTÁTICAS

- Identifica defectos propios de la técnica
- Identifica defectos en diseño y codificación
- Incrementa la productividad del desarrollo
- Reduce costo y tiempo de desarrollo
- Reduce costo y tiempo de pruebas
- Reducir el coste total de la calidad
- Mejorar la comunicación del equipo
- Identificar defectos de mantenibilidad

Las pruebas estáticas previenen problemas en el diseño y codificación, descubriendo inconsistencias y redundancias que pueden afectar el rendimiento del software. Esto mejora la calidad general del producto.

Incrementar la productividad y reducir costos son beneficios adicionales de las pruebas estáticas. Al detectar problemas antes, el desarrollo se vuelve más eficiente y menos propenso a errores.

Diferencias entre pruebas estáticas y dinámicas:

Los objetivos de ambas pruebas pueden ser los mismos, estas se complementan entre sí para encontrar diferentes tipos de defectos. Ambas técnicas ayudan a evaluar la calidad de los productos de trabajo e identificar los problemas tan temprano como sea posible.

- La prueba estática: Las pruebas estáticas son esenciales para identificar defectos en productos de trabajo sin ejecutarlos, lo que ahorra tiempo y costos en el desarrollo de software. Su enfoque en la calidad interna permite detectar errores que podrían ser difíciles de alcanzar con pruebas dinámicas. Las pruebas estáticas se centran en la calidad interna y pueden detectar defectos como variables no definidas, lo que facilita su corrección. Los defectos encontrados son más fáciles de conseguir y reparar.
- Las pruebas dinámica identifica fallas, se enfocan en el comportamiento visibles desde el exterior, utilizando un conjunto de valores de entrada y comparando la salida obtenida con el valor que esperamos.

DEFECTOS TÍPICOS EN PRUEBAS ESTÁTICAS
<ul style="list-style-type: none">- Defectos en requisitos- Defectos en diseño- Defecto de codificación- Desviaciones de estándares- Especificaciones de interfaz incorrectas- Vulnerabilidades de seguridad- Trazabilidad o cobertura de la base de pruebas

Unidad 6 - Proceso de revisión:

Planificación:

Introducción a la unidad 6

¡Bienvenidos a la sexta unidad del emocionante curso de Tester de Software! En la unidad anterior, exploramos en detalle los niveles de pruebas y su aplicación en la evaluación de software. En esta nueva unidad nos adentraremos en los diferentes tipos de pruebas disponibles para testear software, con el objetivo de comprender a fondo cada tipo de prueba, sus objetivos y cómo implementarlos de manera efectiva.

Estudiar los diferentes tipos de pruebas en el curso de Tester de Software es crucial para garantizar la calidad del software y el éxito de los proyectos de desarrollo.

Destacamos la importancia de los siguientes puntos:- Mejora la calidad del software: Cada tipo de prueba se centra en aspectos específicos del software para identificar y corregir defectos, resultando en un software más fiable y de mayor calidad.

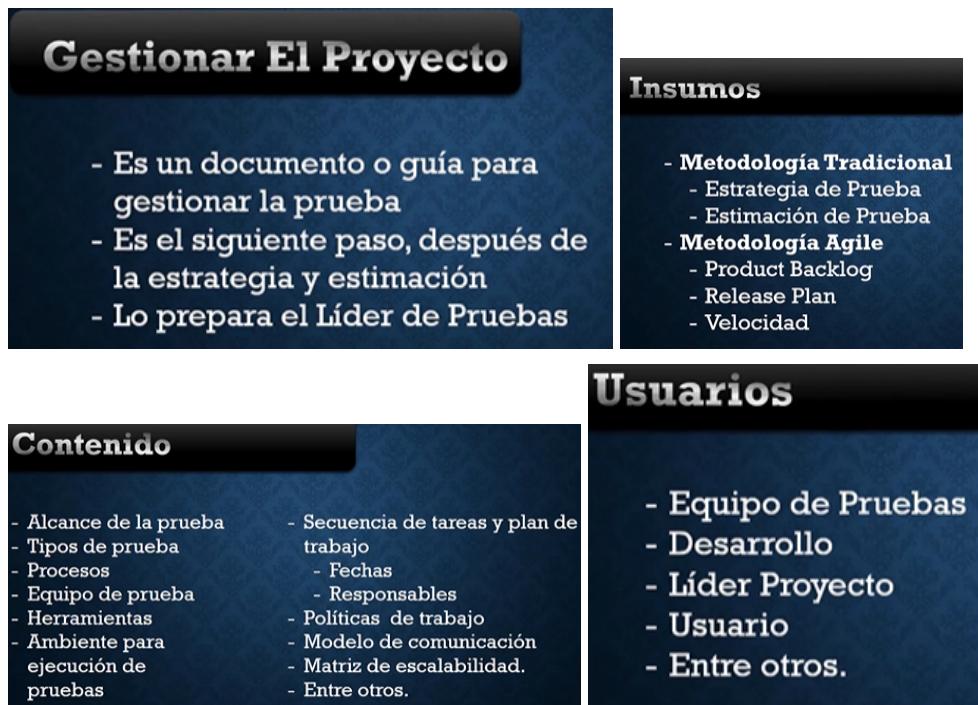
- Asegura la funcionalidad correcta: Las pruebas funcionales garantizan que el software cumpla con los requisitos y funcione según las expectativas del cliente para satisfacer sus necesidades y mantener su satisfacción.
- Detecta problemas de rendimiento: Las pruebas de rendimiento y carga ayudan a identificar problemas como lentitud o fallos bajo carga, crucial para garantizar la eficiencia del software en aplicaciones críticas.
- Incrementa la seguridad: Las pruebas de seguridad y vulnerabilidad son esenciales para identificar vulnerabilidades y fortalecer la seguridad del software, especialmente en entornos donde la seguridad de los datos es crítica.
- Reduce costos y tiempos de desarrollo: Identificar y solucionar problemas en etapas tempranas del ciclo de desarrollo es más económico y rápido, lo que se logra con un conocimiento profundo de los tipos de pruebas.
- Cumple con normativas y regulaciones: En industrias reguladas, las pruebas son fundamentales para cumplir con normativas estrictas, por lo que comprender y aplicar los tipos de pruebas necesarios es crucial.

En resumen, estudiar los tipos de pruebas en el curso de Tester de Software proporciona las herramientas y habilidades necesarias para evaluar y mejorar la calidad del software, esencial en un entorno competitivo y exigente. Esta unidad explorará en detalle los diferentes tipos de pruebas para testear software, permitiendo seleccionar estrategias adecuadas, identificar escenarios relevantes y mejorar la calidad del software. ¡Les deseamos mucho éxito en esta emocionante unidad!

Planificación:

VIDEO 01

¿Qué es un plan de pruebas de software? Nos sirve para gestionar el proyecto.



El proceso de las pruebas de software:

El proceso de pruebas de software es una parte crucial en el desarrollo del software moderno, ya que permite identificar errores, mejorar la calidad, optimizar la experiencia del usuario y garantizar la seguridad del producto. Durante las revisiones técnicas formales (RTF), se busca identificar errores, mejorar la calidad y asegurarse de que el software cumpla con los requisitos establecidos. Estas revisiones implican una planificación, control y mantenimiento adecuados, una definición clara de los roles de los participantes y la documentación de hallazgos y conclusiones. La detección temprana de errores es fundamental en cada etapa del proceso de desarrollo de software, ya que errores no detectados pueden convertirse en problemas mayores más adelante. Las pruebas de usabilidad, rendimiento y carga son esenciales para garantizar la eficiencia y funcionalidad del software, así como para mejorar la experiencia del usuario y la competitividad del producto. En conclusión, comprender y aplicar efectivamente el proceso de pruebas de software es esencial para los profesionales de la industria de la tecnología de la información, ya que contribuye al éxito del software y al logro de los objetivos del proyecto. Es importante llevar a cabo las revisiones adecuadas para detectar y corregir errores, mejorar la calidad general del software y garantizar que cumpla con los requisitos establecidos. Mediante una planificación cuidadosa, un control adecuado y una documentación detallada, es posible optimizar el proceso de pruebas y maximizar la eficacia de estas actividades durante todo el ciclo de desarrollo del software.

Creación del plan de pruebas:

Una descripción detallada de los objetivos de pruebas a alcanzar, los medios y el cronograma para lograrlos, todo esto organizado para coordinar las actividades de pruebas para un elemento o un conjunto de elementos de las pruebas.

Un proyecto puede tener uno o varios planes de prueba.

Puede hacer un plan "Master" donde se abarcan todas las actividades de pruebas del proyecto, pero también se puede dividir en planes específicos como por ejemplo el plan de pruebas de rendimientos, plan de sistemas, plan de pruebas automatizadas, donde se describirán en más detalles las actividades específicas.

Hay planes de pruebas para actividades ajenas al proyecto ej= plan para actividades de mantenimiento, esto ocurre cada vez que un proyecto finalice

El plan de pruebas que haremos sera para un proyecto donde se incluyen funcionalidades nuevas para calcular bonos salariales de un proyecto de RRHH a un sistema que ya existe .

Cálculo de Bono Salarial - Sistema de Recursos Humanos

En el sistema de Recursos Humanos de la empresa Quality Stream se necesita añadir una nueva funcionalidad que permita calcular bonos salariales que comenzará a pagarse a los trabajadores.

Existirán bonos trimestrales (3 meses) que se pagarán de acuerdo al cumplimiento de los objetivos individuales.

Existirá un bono anual que se pagará al finalizar el año de acuerdo al crecimiento financiero de la empresa

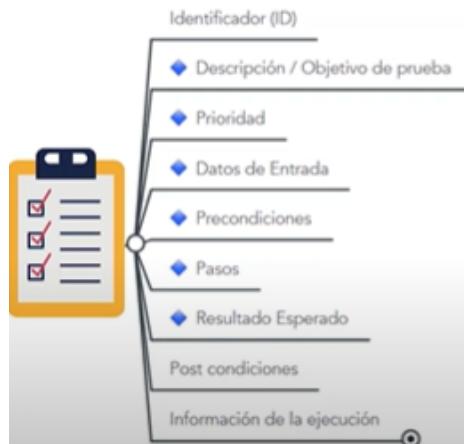
- Comienza con el nombre del proyecto, un logo, la versión.
- Historia de revisiones con autores descripción y fecha
- 1 Índice del documento
 - 1.1 Alcance del documento.
 - 1.2 Referencias
 - Especificaciones
 - ISO 29119
 - 1.3 Glosario
 - incluimos abreviaciones, términos para aclarar el entendimiento del plan
- 2 Contexto de las pruebas
 - 2.1 Proyecto/Subproceso o tipos de prueba (descripción) mostramos módulos
 - 2.2 Elementos de prueba. Listamos los elementos a los cuales le vamos a realizar las pruebas.

- 2.3 Alcance de las pruebas. Lo que vamos a probar que son los módulos mencionados en el 2.2 y lo que no probemos en el proyecto.
 - 2.4 Suposiciones y restricciones.
 - 2.5 Partes interesadas. Personas que se involucran en el proyecto y tienen responsabilidades del proyecto/ pruebas.
- 3. Comunicación de las pruebas. Detallamos las responsabilidades del clientes y los miembros del equipo, como va a hacer el protocolo de comunicación externa/interna y como va a hacer las resoluciones de equipos.
- 4. Registro de riesgos. con la probabilidad, impacto, severidad y plan de mitigación.
- 5. Estrategia de prueba. Describe el enfoque de las pruebas para el proyecto de prueba o subprocesso de prueba específico. diferentes estrategias para las diferentes pruebas(aceptación,sistema, componente).
 - Subproceso de prueba
 - Entregables de Prueba. (documentación)
 - Técnica de diseño de prueba (explicación de las técnicas de diseño a utilizar)
 - Criterio de finalización y prueba. (descripción cuando terminamos de probar
 - Métrica (métricas de recolección del proyecto).
 - Requisitos del entorno de pruebas (hablamos del hardware, del ambiente)
 - Ambiente de pruebas
 - Herramientas de pruebas.
 - Re-Testing y regresión de las pruebas. Debemos definir cuantas se realizarán
 - Criterios de suspensión y reanudación
 - Criterios de suspensión
 - Criterio de reanudación
 - Desviaciones de la estrategia de prueba organizacional. (muchos proyectos no lo tendrán, este dependerá de estrategias de pruebas a nivel organizacional con elementos genéricos). - Se puede obviar o NO.
- 6. Actividades y estimados de prueba. Se puede poner en un documento o sistema aparte (se puede obviar, depende). Aca se listan un conjunto de actividades genéricas.
- 7. Personal. Lo proporciona RRHH
 - Roles, actividades y responsabilidades (matriz RACI)
Responsible-Accountable-Consulted-Informed.
 - Necesidades de contratacion
 - Necesidades de entrenamiento
- 8. Cronograma. Se lleva en un sistema aparte colocando el link sino se incluye el cronograma

Casos de prueba: Es un conjunto de entradas y condiciones de ejecución que busca comprobar que el software funcione de la manera esperada hay partes esenciales que los casos de prueba deberán tener y hay otras que dependen del proyecto y herramientas que utilicemos

PARTES DE UN CASO DE PRUEBA

Estructura principal de un caso de prueba.

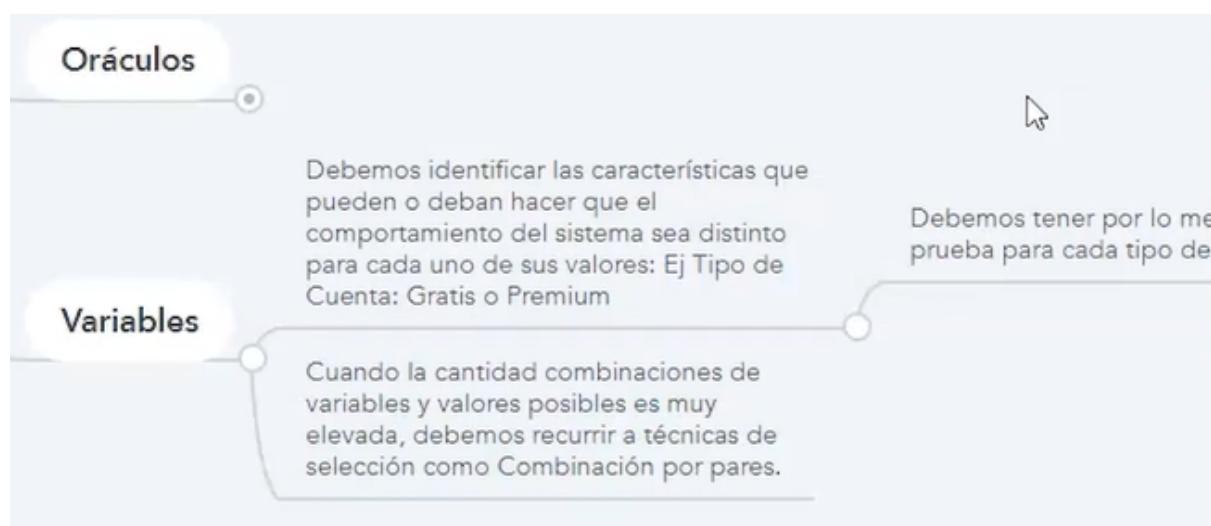
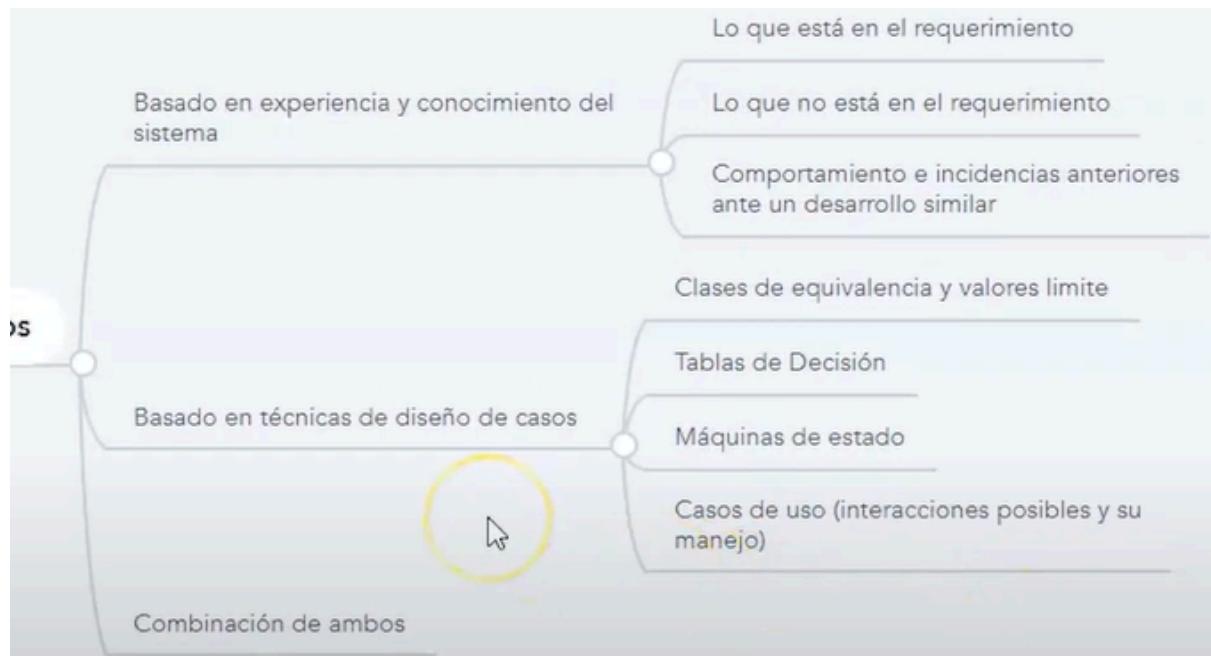


¿Cómo sabemos cuál es nuestro resultado esperado? Acá entra en juego el concepto de "Oráculo", para cada conjunto de pasos y de datos de entrada cual va a ser nuestro resultado esperado.

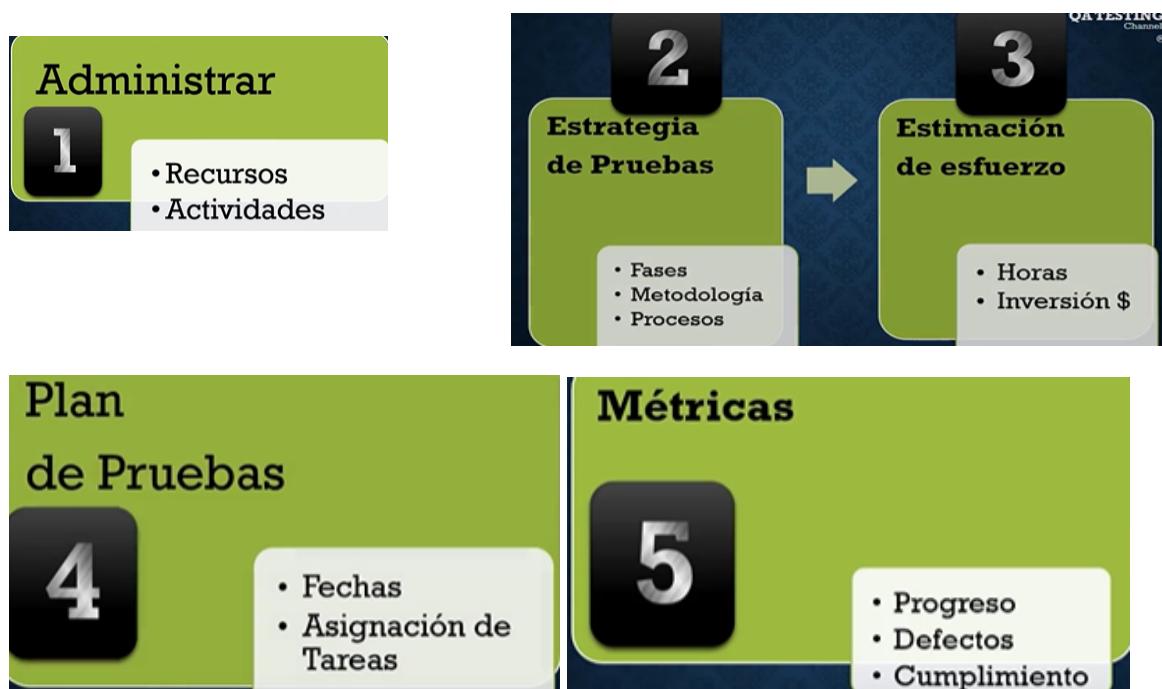
Un oráculo puede ser

- personas que conozcan el sistema o negocio, que nos indican cual es el resultado esperado
- Documentación
- Especificación de requerimientos en el cual está claramente escrito y definido secuencias de paso.
- Modelos, diagramas que especifican todo el flujo.
- Normas, estándares que especifican determinadas reglas para el software.
- Sistemas o app similares si no hay ninguno de los anteriores.

¿Cómo identificamos los casos de prueba? Tratar de encontrar la mayoría de los defectos en la menor cantidad de tiempo (uno de los principios era que la mayoría de los casos no se puede hacer pruebas exhaustivas ya que la combinación de las variables puede ser muy grande y normalmente no hay tiempo para probar todo.)



Puntos a considerar en una gestión de prueba:



Paso a paso como realizar una gestión de pruebas exitosa:

Paso 1. Comprender los Objetivos del Proyecto

Antes de comenzar, es fundamental comprender los objetivos del proyecto. Esto incluye entender las expectativas del cliente, los requisitos del software y las metas de calidad. Cuanto mejor se comprendan estos aspectos, más efectivas serán las pruebas.

Paso 2. Planificar las Pruebas

La planificación es esencial. Debes definir claramente qué se probará, cómo se probará, cuándo se realizarán las pruebas y quién las llevará a cabo. Elabora un plan de pruebas detallado que sirva como guía durante todo el proceso.

Paso 3. Diseñar Casos de Prueba

Crea casos de prueba que describan escenarios específicos de prueba. Define las entradas, las acciones y los resultados esperados. Estos casos de prueba deben ser claros y detallados para garantizar una ejecución efectiva.

Paso 4. Seleccionar Herramientas y Recursos

Elige las herramientas de pruebas adecuadas y asigna recursos humanos según las necesidades del proyecto. Puedes utilizar herramientas de automatización, gestión de defectos y seguimiento de pruebas para mejorar la eficiencia.

Paso 5. Ejecutar las Pruebas

Lleva a cabo las pruebas según el plan establecido. Ejecuta los casos de prueba, registra los resultados y documenta cualquier defecto encontrado. Asegúrate de seguir un enfoque sistemático y completo.

Paso 6. Gestionar Defectos

Si se identifican defectos durante las pruebas, regístralos y priorízalos. Establece un proceso para gestionar la resolución de defectos, incluyendo la revisión, la corrección y la verificación.

Paso 7. Realizar Pruebas de Regresión

Después de corregir los defectos, realiza pruebas de regresión para garantizar que las correcciones no hayan introducido nuevos problemas en el software. Esto es especialmente importante cuando se realizan actualizaciones.

Paso 8. Comunicación y Reportes

Mantén una comunicación constante con el equipo de desarrollo. Informa sobre el progreso de las pruebas, los resultados y los problemas encontrados. Genera informes detallados que documenten el estado de las pruebas.

Paso 9. Evaluar la Cobertura de Pruebas

Evalúa la cobertura de pruebas para asegurarte de que se han probado todas las funcionalidades críticas del software. Ajusta los casos de prueba según sea necesario para lograr una cobertura completa.

Paso 10. Mejora Continua

Después de cada ciclo de pruebas, realiza una revisión retrospectiva. Identifica áreas de mejora en los procesos de pruebas y en los casos de prueba. Implementa cambios para aumentar la eficiencia y la efectividad en futuros proyectos.

Paso 11. Cierre de Pruebas

Una vez que las pruebas se hayan completado con éxito y el software cumpla con los requisitos y estándares de calidad, realiza una revisión final y aprueba el producto para su implementación.

Paso 12. Entrega y Seguimiento Post-implementación

Después de implementar el software, realiza un seguimiento post-implementación para verificar que funcione de manera adecuada en el entorno de producción. Aborda cualquier problema que surja de manera oportuna.

Siguiendo estos pasos, puedes llevar a cabo una gestión de pruebas efectiva que garantice la calidad y el éxito de tu proyecto de desarrollo de software. La gestión de pruebas es un proceso continuo que requiere atención constante para asegurar que el software cumpla con los estándares de calidad y satisfaga las necesidades del cliente.

- Ejemplo práctico para una Gestión de Pruebas

Bienvenidos a la Clase sobre Gestión de Pruebas de Software! Hoy, vamos a sumergirnos en un emocionante ejemplo paso a paso de cómo llevar a cabo una gestión de pruebas exitosa en un proyecto de desarrollo de software. Este ejemplo práctico nos ayudará a comprender cómo aplicar los principios y las mejores prácticas de la gestión de pruebas en el mundo real.

Paso 1: Comprender los Objetivos del Proyecto

Imaginemos que estamos trabajando en el desarrollo de una aplicación móvil de banca en línea para un banco. El objetivo del proyecto es ofrecer a los clientes una forma segura y conveniente de acceder a sus cuentas y realizar transacciones desde sus dispositivos móviles.

Paso 2: Planificar las Pruebas

Nuestro primer paso es crear un plan de pruebas detallado. Definimos los objetivos de las pruebas, como verificar la seguridad de las transacciones y la usabilidad de la aplicación. También establecemos un calendario de pruebas que coincide con los hitos del desarrollo.

Paso 3: Diseñar Casos de Prueba

Luego, diseñamos casos de prueba específicos para la aplicación móvil. Por ejemplo, creamos casos como "Iniciar sesión en la aplicación", "Realizar una

"transferencia de fondos" y "Ver el historial de transacciones". Especificamos las acciones que realizarán los usuarios y los resultados esperados.

Paso 4: Seleccionar Herramientas y Recursos

Elegimos herramientas de pruebas adecuadas, como Appium para pruebas móviles y JIRA para la gestión de defectos. Además, formamos un equipo de pruebas que incluye probadores manuales y automatizados, así como expertos en seguridad de aplicaciones.

Paso 5: Ejecutar las Pruebas

Llevamos a cabo las pruebas según el plan. Los probadores manuales realizan las acciones definidas en los casos de prueba y documentan cualquier problema, como problemas de navegación o errores de funcionamiento. Los probadores automatizados ejecutan pruebas de seguridad y estabilidad.

Paso 6: Gestionar Defectos

Si se encuentran defectos, los registramos en JIRA y los asignamos a los desarrolladores para su corrección. Seguimos de cerca el proceso de resolución y verificamos las correcciones antes de volver a probar.

Paso 7: Realizar Pruebas de Regresión

Después de las correcciones, realizamos pruebas de regresión para asegurarnos de que las actualizaciones no hayan introducido nuevos problemas. Ejecutamos los casos de prueba relevantes para garantizar la estabilidad de la aplicación.

Paso 8: Comunicación y Reportes

Mantenemos una comunicación constante con el equipo de desarrollo. Proporcionamos informes regulares sobre el progreso de las pruebas y compartimos los resultados de las pruebas de seguridad. Esto permite abordar los problemas de manera oportuna.

Paso 9: Evaluar la Cobertura de Pruebas

Evaluamos la cobertura de pruebas para asegurarnos de que se hayan probado todas las funciones críticas de la aplicación. Ajustamos los casos de prueba según sea necesario para lograr una cobertura completa.

Paso 10: Mejora Continua

Al finalizar el proyecto, realizamos una revisión retrospectiva. Identificamos áreas de mejora en los procesos de pruebas y en los casos de prueba. Esto nos permite aprender de la experiencia y aplicar mejoras en futuros proyectos.

Con este ejemplo, hemos recorrido un viaje completo de gestión de pruebas que garantiza que nuestra aplicación móvil de banca en línea sea segura, eficiente y cumpla con las expectativas de nuestros clientes. ¡Prepárense para aprender más sobre la gestión de pruebas y cómo aplicar estos principios en sus propios proyectos!