UNIVERSIDAD NACIONAL
TORIBIO RODRÍGUEZ DE
MENDOZA DE AMAZONAS

FACULTAD DE INGENIERIA
DE SISTEMAS Y MECANICA
ELECTRICA

PRUEBAS DE SOFTWARE

Dr. Italo Maldonado Ramirez

Mg. Juliana del Pilar Alva Zapata
Mg. Roberto Carlos Santa Cruz Acosta

2021

PRUEBAS DE SOFTWARE

Hecho el Depósito Legal en la Biblioteca Nacional del Perú N° 2021- -07592.

EDITOR

Juliana del Pilar Alva Zapata

juliana.alva@untrm.edu.pe

Av. Unión 575 La Victoria Chiclayo

Celular : 972678717

Bagua - Perú

Primera Edición Digital

Julio 2021

AUTORES

Dr. Italo Maldonado Ramirez

Mg. Juliana del Pilar Alva Zapata

Mg. Roberto Carlos Santa Cruz Acosta

INDICE

Contenido

PRUEBAS DE SOFTWARE

I. Ciclo de vida de las pruebas de software

Una serie de actividades que realizamos para realizar pruebas de software, se refiere a un proceso de prueba con pasos específicos, se debe realizar estos pasos en un orden específico para garantizar que el software cumpla con los objetivos de calidad. Llevamos a cabo cada actividad de forma planificada y sistemática y cada fase tiene sus propios objetivos y resultados. Las fases de STLC son diferentes para cada organización.

Las fases son las siguientes:

- Análisis de requisitos
- Fase de planificación
- Integración de casos de prueba
- Configurar entorno de prueba
- Fase de implementación
- Cierre del ciclo de prueba

1.1. Análisis de requisitos de prueba

Durante esta fase, el equipo de prueba estudiará el software requisitos con el objetivo de identificar los requisitos comprobables. El equipo de control de calidad se comunicará con varias partes interesadas (cliente, analista de negocios, líderes técnicos, arquitectos de sistemas, etc.) para comprender los requisitos en detalle. Los requisitos pueden ser funcionales (definir lo que debe hacer el software) o no funcionales (definir el rendimiento y la seguridad del sistema).

TABLA 1 - LISTA DE REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

NOMBRE DEL PROYECTO						
LISTA DE REQUISITOS FUNCIONALES Y NO FUNCIONALES						
Num_Req	**Nom_Corto**	**Descripción**	**Estado**	**Prioridad**	**Nivel Riesgo**	**Tipo_Req**
1	REQ_1	Descripción del requerimiento 1	Aprobado	Alto	Significativo	Funcional
2	REQ_2	Descripción del requerimiento 2	Aprobado	Alto	Significativo	Funcional
3						

- **Num_Req** = Numero del requerimiento
- **Nom_Corto** = Nombre corto del requerimiento
- **Descripción** = contendrá la descripción del requerimiento. Se sugiere que este sea entendible y fácil de identificar.
- **Estado** = se considera la siguiente escala

TABLA 2 . ESTADOS DEL REQUERIMIENTO

ESTADOS DEL REQUERIMIENTO	
ESCALA	**SIGNIFICADO**
Propuesto	El requerimiento ha sido propuesto por el usuario y está en análisis
Aprobado	El requerimiento ya ha sido aprobado para desarrollo
Depurado	El requerimiento no fue aprobado y debe retirarse.

- **Prioridad** = indicará la prioridad que presenta el requerimiento para ser atendido

TABLA 3 - PRIORIDAD DEL REQUERIMIENTO

PRIORIDAD DEL REQUERIMIENTO	
ESCALA	**SIGNIFICADO**
Alto	El requerimiento es muy importante y necesita atención urgente.*
Medio	El requerimiento es importante pero puede esperar para su atención.
Bajo	El requerimiento no es importante y se puede atender luego.

- **Nivel_Riesgo** = se indicará el nivel de riesgo que presenta un requerimiento.

TABLA 4 - NIVEL DE RIESGO DEL REQUERIMIENTO

NIVEL DE RIESGO DEL REQUERIMIENTO	
ESCALA	**SIGNIFICADO**
Critico	El requerimiento representa un riesgo crítico para la organización.
Significativo	El requerimiento representa un riesgo que puede ser tratado con otras medidas mientras se desarrolla
Ordinario	El requerimiento no representa un riesgo para la organización.

- **Tipo_Req = Funcional** / No funcional

1.2. Planificación de pruebas dentro del ciclo de vida de pruebas de software

En esta fase, un gerente senior de control de calidad generalmente determinará los esfuerzos y las estimaciones de costos para el proyecto y completará el plan de prueba. Por el momento también determinamos la estrategia de prueba.

Actividades:

- Preparación del plan de prueba / documento de estrategia para diferentes tipos de pruebas.
- Selección de herramientas de prueba.
- Estimación de los esfuerzos de prueba.
- Planificación de recursos y determinación de roles y responsabilidades.

Entregables:

- Plan de prueba / estrategia. (Anexo 01)
- Estimación del esfuerzo.

1.3. Desarrollo de casos de prueba dentro del ciclo de vida de prueba

En esta fase, se crean, verifican y reelaboran los casos de prueba y los scripts de prueba. Identificamos, creamos y evaluamos los datos de prueba para su posterior edición.

Actividades

- Cree casos de prueba, scripts de automatización (si corresponde).
- Revisión y referencia de casos de prueba y guiones.
- Crear datos de prueba (como entorno de prueba está disponible).

Entregables

- Casos de prueba / scripts.
- Datos de prueba

TABLA 5 - CASO DE PRUEBA

CASO DE PRUEBA – CP001						
NOMBRE DEL PROYECTO:	Indicar el nombre del proyecto					
COD. CASO DE PRUEBA:	Código del caso de uso (formato CP00X; x es la numeración iniciando de 1)					
PRIORIDAD:	(ALTA / MEDIA / BAJA)					
NOMBRE DEL MODULO:	Nombre del módulo en donde se encuentra la opción a probar					
DESCRIPCION:	Nombre del caso de uso (tiene que ver con el nombre de la opción a probar)					
FECHA DE PRUEBA:	Fecha que se está realizando la prueba					
INGENIERO DE PRUEBA A CARGO:	Ingeniero encargado de ejecutar la prueba					
PRE-CONDICION:	Opción que se realiza antes de ejecutar la opción a probar					
POST-CONDICION:	Opción que se realiza después de ejecutar la opción a probar					
CONSIDERACIONES	Consideraciones a tener en cuenta antes de iniciar la ejecución de la prueba					
VERSION DEL CASO DE PRUEBA	Numero de versión de la prueba					
N° ACT.	**DESCRIPCION**	**DATOS DE PRUEBA**	**R. ESPERADO**	**R. REAL**	**ESTADO (paso / no paso)**	**OBSERVACION**
1	Descripción de la actividad a realizar (Ejemplo: Ingresar al link del sistema:)	Datos que se deben de tener en cuenta para probar (ejemplo: http://kljdfkjdsfkl	Resultado esperado (Ejemplo: ingresar al sistema)	Resultado Realizado de la prueba (ejemplo: se pudo cargar la página del sistema)	pasó	Indicar las observaciones que se encontraron al realizar la actividad (Ejemplo: tardo unos minutos considerables en acceder
2	(Ejemplo: Nos mostrará la página de logeo del sistema)		Ejemplo: muestre la página de logeo)	Ejemplo: se mostró la página de logeo.	pasó	
3	Ejemplo: revisar la presentación de la página de logeo)		Ejemplo: la presentación esté correcta	Ejemplo: el formulario de acceso no es responsive	pasó	

1.4. Configure un entorno de prueba como parte del ciclo de vida de prueba

El entorno de prueba determina las condiciones de software y hardware bajo las cuales probamos. La configuración del entorno de prueba es uno de los aspectos críticos del proceso de prueba. el equipo de prueba deberá realizar una prueba de preparación (prueba de humo) del entorno de prueba para determinar si es satisfactoria.

Actividades

- Comprenda la arquitectura, el entorno necesario y haga una lista de los requisitos de hardware y software para el entorno de prueba.
- Prueba de configuración del entorno y los datos de prueba.
- Realice una prueba de humo en la construcción.

Entregables

- Configuración del entorno y datos de prueba listos.
- Resultados de la prueba de humo.

1.5. Ejecución de prueba dentro del ciclo de vida de pruebas de software

Durante esta fase, los evaluadores realizarán las pruebas según los planes de prueba y los casos de prueba preparados. Reportamos errores al equipo de desarrollo para su corrección y volveremos a hacer la prueba después de la recuperación.

Actividades

- Realizar pruebas de acuerdo al plan.
- Documente los resultados de las pruebas y el registro de errores.
- Prueba las correcciones.
- Siga los defectos hasta el cierre.

Entregables

- Casos de prueba actualizado con resultados.
- Reporta defectos.

1.6. Cierre de ciclo de vida de pruebas de software

El equipo de prueba discute y analiza el ciclo de prueba y para mejorar las estrategias de prueba utilizadas. Debemos extraer lecciones del ciclo de prueba actual. La idea es eliminar los cuellos de botella del proceso para futuros ciclos de prueba. También recogemos Mejores Prácticas para proyectos similares en el futuro.

Actividades

- Evalúe los criterios para completar el ciclo en función del tiempo, la cobertura de la prueba, los costos, el software, los objetivos comerciales críticos y la calidad.
- Prepare estadísticas de prueba basadas en los parámetros anteriores.
- Documentar los detalles del proyecto.
- Prepare el informe para la conclusión de la prueba.

Entregables

- Informe final de prueba, Estadísticas de prueba

II. Estrategias de las pruebas de software

durante el proceso de software, debe definirse una plantilla para la prueba del software: un conjunto de pasos que incluyen métodos de prueba y técnicas de diseño de casos de prueba específicos.

Una plantilla para la prueba debe tener las siguientes características genéricas:

- ✓ Para realizar una prueba efectiva, debe realizar revisiones técnicas efectivas. Al hacerlo, eliminará muchos errores antes de comenzar la prueba.
- ✓ La prueba comienza en los componentes y opera “hacia afuera”, hacia la integración de todo el sistema de cómputo.
- ✓ Diferentes técnicas de prueba son adecuadas para distintos enfoques de ingeniería de software y en diferentes momentos en el tiempo.
- ✓ Las pruebas las realiza el desarrollador del software y (para proyectos grandes) un grupo de prueba independiente.
- ✓ Prueba y depuración son actividades diferentes, pero la depuración debe incluirse en cualquier estrategia de prueba.
- ✓ Una estrategia para probar el software también puede verse en el contexto de la espiral
- ✓ La prueba de unidad comienza en el vértice de la espiral y se concentra en cada unidad (por ejemplo, componente, clase u objeto) del software como se implementó en el código fuente.
- ✓ prueba de integración, donde el enfoque se centra en el diseño y la construcción de la arquitectura del software.
- ✓ prueba de validación, donde los requerimientos establecidos como parte de su modelado se validan confrontándose con el software que se construyó.
- ✓ Finalmente, se llega a la prueba del sistema, donde el software y otros elementos del sistema se prueban como un todo

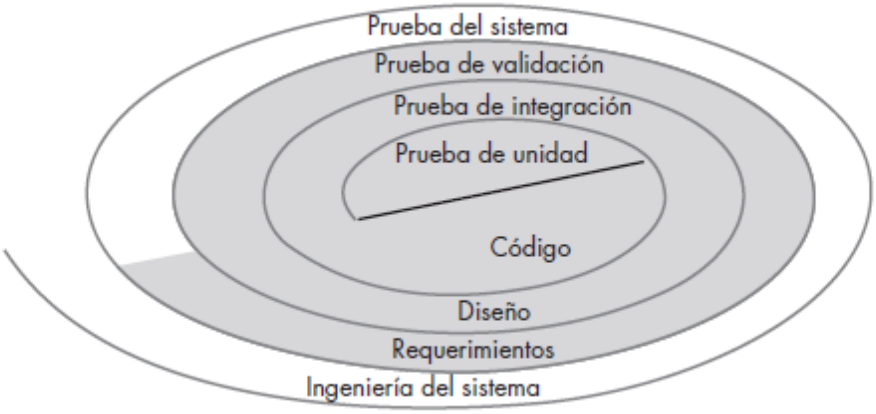

IMAGEN 1 -TIPOS DE PRUEBAS DENTRO DEL MODELO EN ESPIRAL

La imagen 1, nos muestra como los tipos de pruebas van a ir interactuando en cada iteración de acuerdo al nivel de avance del software en desarrollo

III. Plan de Pruebas

El plan de pruebas de software se elabora para atender los objetivos de calidad en un desarrollo de sistemas, encargándose de definir aspectos como por ejemplo los módulos o funcionalidades sujeto de verificación, tipos de pruebas, entornos, recursos asignados, entre otros aspectos. Incluyendo el alcance, estrategia de pruebas, tipos de pruebas de software a incluir, criterios de aceptación, requisitos de infraestructura, requisitos de personal y la planificación.

3.1. Analizar los requerimientos de desarrollo de software

Para elaborar un plan de pruebas de software lo primero que debes hacer es entender los requerimientos de usuario que componen la iteración o proyecto, que son el sujeto de la verificación de calidad que se va a realizar.

Deberás analizar toda la información de la ingeniería de requisitos, incluyendo la matriz de trazabilidad, especificaciones y diseño funcional, requisitos no funcionales, casos de uso, historias de usuario (si estás trabajando con metodologías ágiles), entre otra documentación.

3.2. Identificar las funcionalidades nuevas a probar

A partir de la documentación del análisis de requisitos y de las entrevistas con el equipo de ingeniería de requisito y desarrollo, debes identificar e incluir en el plan de pruebas de software la lista de las funcionalidades (Características) totalmente nuevas.

3.3. Identificar las funcionalidades de sistemas existentes que deben probarse

Se debe identificar las funcionalidades existentes que estén siendo impactadas por el desarrollo de alguna forma, considerando todos los componentes afectados en todas las capas de la arquitectura de software.

Existen dos situaciones que se puede encontrar al identificar estas funcionalidades:

- Funcionalidades modificadas de cara al usuario: si una funcionalidad está siendo modificada agregando más pantallas o cambios a su flujo de proceso, debe ser incluida en el plan de pruebas de software.
- Funcionalidades modificadas en sus componentes internos: se modifican componentes internos que comparten con otras funcionalidades del sistema

3.4. Definir la estrategia de pruebas

Consiste básicamente en seleccionar cuáles son los tipos de pruebas de software que se deben realizar. Es recomendable seguir un marco de referencia para determinar los tipos de prueba, como por ejemplo los tipos de pruebas de software definidos por el ISTQB.

- ✓ Pruebas funcionales
- ✓ Pruebas no funcionales
- ✓ Pruebas de caja blanca
- ✓ Pruebas de regresión

3.5. Definir los criterios de inicio, aceptación y suspensión de pruebas

A. **Criterios de aceptación o rechazo:** Para definir los criterios de aceptación o rechazo, es necesario definir el nivel de tolerancia a fallos de calidad. Si la tolerancia a fallos es muy baja puede definirse como criterio de aceptación que el 100% de los casos de prueba estén sin incidencias.

   Una vez logradas las condiciones, se darán por aceptadas las pruebas y el desarrollo de software.

B. **Criterios de inicio o reanudación:**

C. Criterios de suspensión: Las condiciones van a depender de los acuerdos de nivel de servicio (SLAs) internos de la organización y también de los acuerdos establecidos en cada proyecto individual.

3.6. Identificar los entornos (ambientes) requeridos

Posteriormente se definen y documentan las características de los entornos de Hardware y Software necesarios para realizar la ejecución de las pruebas de software. Además, en esta sección del plan de pruebas, también se definen los requisitos de sistemas operativos, software y herramientas de las estaciones de trabajo de los Testers.

También deben definirse los requisitos de hardware y software para los siguientes componentes:

- ✓ Herramienta de gestión de calidad de software.
- ✓ Herramientas para automatización de pruebas.
- ✓ Herramientas de BDD, TDD y Testing de Web Services).

3.7. Elaborar la planificación de las pruebas

La planificación de las pruebas abarca:

Matriz de responsabilidades

Puede usarse una Matriz RACI o Matriz RAM como plantilla. Esta se define con perfiles genéricos o inclusive con el equipo de trabajo si ya se conoce cuál es el que será asignado.

Cronograma:

Para elaborar un cronograma real, es importante definir actividades críticas como por ejemplo los tiempos de instalación de versiones en los entornos de pruebas, pruebas de validación de ambientes antes de comenzar a hacer las pruebas y las iteraciones por incidencias, que es el tiempo invertido en volver a probar los casos de prueba fallidos.

3.8. Identificar los riesgos y definir planes de respuesta

Para el Software Testing, los riesgos por lo general están vinculados con factores como:

- ✓ Posibles dificultades en la disponibilidad de entornos.
- ✓ Pruebas que dependen de factores externos al proyecto y la organización.
- ✓ Disponibilidad de personal con conocimientos especializados en alguna herramienta, o en la funcionalidad especifica que se está desarrollando.
- ✓ Dependencias con otros proyectos.
- ✓ Posibilidad que alguna premisa no se cumpla.

Para identificar los riesgos es necesario enumerar cada una de estas dependencias y por medio de mesas de trabajo y tormentas de ideas pensar en las posibilidades de que algo salga mal (u oportunidades para que salga bien).
Luego de la identificación, es necesario también definir planes de respuesta, los cuales deben ser específicos para cada situación particular y riesgo.

IV. Tipos de pruebas

4.1. Pruebas funcionales

Se basa en la funcionalidad de un sistema que se describen en las especificaciones de requisitos.

- Completitud funcional:
- Corrección funcional:
- Pertenecía funcional:

4.2. Pruebas no funcionales

Tiene en cuenta el comportamiento externo del software, es decir cómo funciona el sistema y se suelen usar técnicas de diseño de caja negra

- Pruebas de carga
- Pruebas de rendimiento
- Pruebas de volumen
- Pruebas de esfuerzo
- Pruebas de seguridad
- Pruebas de estabilidad
- Pruebas de compatibilidad
- Pruebas de usabilidad

4.3. Pruebas de caja blanca

Conocida también como caja de cristal o caja transparente. Es una técnica de diseño de caso de prueba que usa la estructura de control para obtener los casos de prueba. Dentro de esta estructura de control podemos encontrar la estructura de un componente de software como pueden ser sentencias de decisiones, caminos distintos del código, la estructura de una página web, etc.

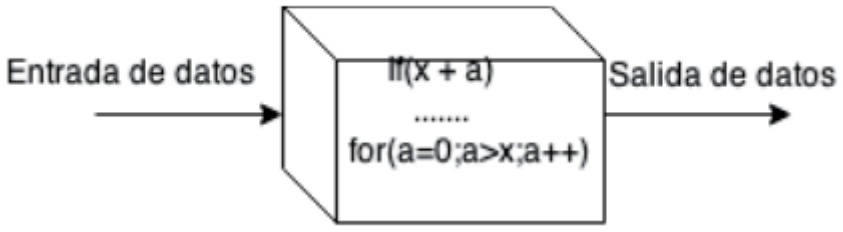

IMAGEN 2 - EJEMPLO DE CAJA BLANCA

Los métodos de pruebas de caja blanca aportan los siguientes puntos:

- ✓ Garantizan que todas las rutas del código se revisan al menos una vez
- ✓ Revisan las condiciones lógicas
- ✓ Revisan las estructuras de datos.

4.4. Pruebas de caja negra

Llamada pruebas de comportamiento. Son las que utilizan el análisis de especificación, tanto funcional como no funcional, sin tener en cuenta la estructura interna del programa para diseñar los casos de pruebas. A diferencia de las pruebas de caja blanca, estas suelen realizarse durante las ultimas etapas de las pruebas

IMAGEN 3 - EJEMPLO CAJA NEGRA

Con los métodos de caja negra se intentan encontrar los errores:

- ✓ Funciones incorrectas y faltantes
- ✓ Errores de inicialización y terminación
- ✓ Errores de interfaz
- ✓ Errores en la estructura

4.5. Pruebas unitarias

Es el primer nivel de las pruebas. Consiste en la verificación de unidades de software de forma aislada, es decir, probar el correcto funcionamiento de una unidad de código. Unidad de código es considerada como unidad de programa, como una funcional o método de una clase que es invocada desde fuera de la unidad y que puede invocar otras unidades. Es por ello que hay que probar que cada unidad funcione separada de las demás unidades de código. Estas pruebas suelen hacerla los desarrolladores.

4.6. Pruebas de integración

Es el segundo nivel de las pruebas. Estas pruebas se ocupan de probar las interfaces entre los componentes, las interacciones con distintas partes de un mismo sistema, como el sistema operativo, el sistema de archivo, el hardware y

las interfaces entre varios sistemas. Estrategia para llevar acabo las pruebas de integración:

- ✓ Integración descendente
- ✓ Integración ascendente
- ✓ Integración Ad-hot
- ✓ Integración del esqueleto

4.7. Pruebas del sistema

Es el tercer nivel de las pruebas. Se comprueba si el producto cumple con los requisitos especificados. Las pruebas de sistema pueden incluir pruebas basadas en riesgos y/o especificaciones de requisitos, proceso de negocio, casos de uso y otras descripciones de texto de alto nivel o modelos de comportamiento de sistema, interacciones con el sistema operativo y recursos del sistema. Las pruebas de sistemas deben estudiar los requisitos funcionales y no funcionales del sistema y las características de calidad. Para ello se aplicarán técnicas de caja negra.

4.8. Pruebas de validación o aceptación

Las pruebas indicadas anteriormente se encuentran bajo la responsabilidad de quien está produciendo el programa. Las pruebas de aceptación son responsabilidad del cliente y pueden ser la única parte en donde estén involucrados. Estas pruebas se llevan a cabo antes de que el programa se ponga en funcionamiento real y tienen que satisfacer las expectativas del cliente. Hay cuatro formas típicas de pruebas de aceptación:

- ✓ Pruebas de aceptación del contrato
- ✓ Pruebas de aceptación del usuario
- ✓ Pruebas operativas
- ✓ Pruebas alfa y beta

4.8.1. Pruebas de aceptación del usuario

Hay casos en que los clientes y el usuario final son diferentes y lo que le parece valido a un usuario final, puede ocurrir que no le parezca valido a otro. Es fundamental pruebas con los usuarios finales

4.8.2. Pruebas operativas

Estas pruebas son llevadas a cabo por los administradores del sistema que se va a poner en producción. En estas pruebas se incluyen las siguientes tareas:

- ✓ Copias de seguridad / restauración
- ✓ Recuperación de desastres

- ✓ Gestión de usuarios
- ✓ Comprobación de vulnerabilidades de seguridad
- ✓ Carga de datos
- ✓ Tareas de mantenimiento

4.8.3. Pruebas alfa y beta

Suele ocurrir que cuando se realizan las entregas al cliente de un programa, este puede ser usado por diferentes usuarios finales y que la cantidad de usuarios finales sea muy amplia. Cuando ocurre esto no es practico pruebas de aceptación con cada uno de estos clientes, por el contrario, se utilizan las pruebas alfa y beta que van a descubrir errores que solo el usuario final va a encontrar.

A. **Las pruebas alfa**

Son las que se ejecutan en las oficinas del desarrollador del producto por un grupo de personas que representan al usuario final. En estas pruebas, el desarrollador estará junto a estos usuarios registrando errores y problemas de uso.

B. **Las pruebas beta**

Se realizan en las oficinas de un cliente o en varias situaciones específicas, ya que el usuario final puede ser varios clientes a los que se les entrega el producto. En estas pruebas el desarrollador no está presente. Estos clientes registran todos los problemas derivados del uso de este producto que más tarde se harán llegar al desarrollador.

4.9. Herramientas de pruebas

los proyectos por más pequeños o grandes que sean, pueden llegar a tener una cantidad de casos de pruebas muy elevado, sin contar que las pruebas se repetirán varias veces debido a las pruebas de regresión. Estos proyectos necesitan de una administración, planificación y ejecución, así como las herramientas que permitirán realizar pruebas automáticas. Para llevar a cabo estas tareas existen diferentes tipos de herramientas que ayudaran en todo lo posible a que el proyecto se maneje más eficientemente y que ayudará a conseguir la calidad deseada.

Existen herramientas que se utilizan para diseñar casos de pruebas, gestionar y administrar pruebas y monitorizar sistemas de pruebas. Al inicio del proyecto, el desarrollador y el probador son los encargados de estudiar y plantear el tipo de herramientas necesarias que se van a usar durante el proyecto.

Se van a clasificar en varias categorías dependiendo del uso que se pueda hacer de ellas:

- ✓ Herramientas para pruebas estáticas
- ✓ Herramientas para planificación y gestión de pruebas
- ✓ Herramientas para pruebas de automatización
- ✓ Drivers y stubs
- ✓ Herramientas para pruebas de carga y rendimiento
- ✓ Herramientas de monitorización y seguridad

4.9.1. Herramientas para pruebas estáticas

Este tipo de herramientas ayudan a encontrar defectos en las etapas tempranas del proyecto.

- ✓ PDM (libre): es una analizadora de código fuente, encuentra defectos comunes de programación como las variables utilizadas, bloques catch vacios, creación de objetos innecesarios, etc.
- ✓ Ckekstyle (libre): es una herramienta de desarrollo para ayudar a los programadores a chequear que el código Java cumple un estándar de codificación.
- ✓ Simian (libre): herramienta que detecta software duplicado. Acepta: Java, C#, C++ C Cobol, -ruby, asp, jsp. Html, xml y visual studio.

A. Herramientas de revisión

Son útiles en los procesos de revisión, listas de comprobación y directrices de revisión, y se utilizan para almacenar y comunicar comentarios de revisión, informes sobre defectos y esfuerzos.

B. Análisis estáticos:

Estas herramientas sirven para localizar defectos en el código antes de realizar las pruebas dinámicas proporcionando soporte para aplicar las normas de codificación, análisis de estructuras y las dependencias.

C. Herramientas de modelado:

Herramientas que sirven para validar modelos de software, como por ejemplo modelos de datos físicos (PDM) de una base de datos relacional, enumerando inconsistencia y localizando defectos.

4.9.2. Herramientas para planificación y gestión

- ✓ **Testlink** (libre): permite crear y gestionar casos de prueba y organizarlo dentro de planes de pruebas, permite generación de informes, asi como priorizar y asignar tareas.

- ✓ **Mantis** (libre): herramienta para gestionar incidencias.
- ✓ **IBM Rational Quality Manager** (licencia): permite gestionar requisitos del proyecto, la gestión, diseño y la creación de pruebas y gestión de incidencias.
- ✓ **Testopia**: es una administradora de casos de pruebas. Diseñado para realizar seguimiento a los casos de pruebas.

4.9.3. Herramientas para pruebas de automatización

- ✓ **Selenium**: es un conjunto de herramientas para automatizar los navegadores web a través de muchas plataformas que nos permitirán crear conjuntos de pruebas sobre aplicaciones web.
- ✓ **SoapUI**: permite probar, simular y generar código de servicios web partiendo del contrato de los mismos en formato WSDL y con vinculo SOAP sobre http

4.9.4. Herramientas para pruebas de carga y rendimiento

- ✓ **Jmeter**: herramienta de prueba de carga para análisis y medir el desempeño de una variedad de servicios con énfasis en aplicaciones web
- ✓ **FunkLoad**: es una herramienta que permite realizar pruebas de carga de aplicaciones web, monitorizar rendimiento de servidores o realizar pruebas de estrés para comprobar la capacidad de recuperación de aplicaciones.

4.9.5. Herramientas de seguridad y monitorización

- ✓ **WiresShark:** permite al desarrollador / probador realizar análisis y solucionar problemas en las redes de comunicaciones. Es un analizador de protocolos de redes. Se utiliza en auditorias de seguridad y pruebas de intrusión.

- ✓ **NMAP:** Permite la exploración de las redes. Utiliza paquetes IP para determinar que equipos están disponibles en la red, que servicios ofrecen, que sistemas operativos están usando, o que tipo de filtros / cortafuegos están usando, etc.

Anexo 01

FORMATO DE PLAN DE PRUEBAS DE SOFTWARE

HOJA RESUMEN DE MODIFICACIONES

VERSIÓN	*FECHA*	*PUNTO*	*CAMBIOS RESPECTO DE LA VERSIÓN ANTERIOR*	*PREPARADO POR*	*APROBADO POR*
Numero de versión	fecha		Si es versión inicial, especificarlo	¿Quién lo hizo?	¿Quién lo aprobó?

INDICE

1. INTRODUCCIÓN

Proyecto(s)		**Tipo de Proyecto**	
Nombre del proyecto		Proyecto de Desarrollo de Software Académico.	
Equipo de Proyecto			
Jefe de Equipo	Nombre de Jefe de proyecto	**Arquitecto de Producto**	Nombre del arquitecto del producto

1.1. OBJETIVOS DEL PLAN DE PRUEBAS

Objetivo de este documento

2. ALCANCE DE LAS PRUEBAS

Mediante los siguientes cuadros se describen los requerimientos de pruebas del sistema.

LISTA DE REQUISITOS DE PRUEBAS					
Num_Req	**Nom_Corto**	**Descripción**	**Estado**	**Prioridad**	**Nivel_Riesgo**
1	REQ_1	DESC_1	Aprobado	Alto	Significativo
2	REQ_2	DESC_2	Aprobado	Alto	Significativo

2.1. CUADRO RESUMEN DE LAS PRUEBAS

Módulos del Sistema a ser probados:	**Módulos:** - Módulos del sistema
Objetivos de las Pruebas	**Objetivos de cada módulo.**
Detalle del orden de ejecución de los módulos	**Orden de los módulos para la ejecución de las pruebas**
Responsabilidad de la Prueba	**Quien será responsable de las pruebas**

2.2. CASOS DE PRUEBAS INCLUIDOS

# Casos Disponibles	Tipo de prueba	Modulo	Total de Casos
Cantidad	funcional	Nombrel modulo	
Cantidad	No funcionales	Nombrel modulo	
		…	
Cantidad	integrales	-	
Cantidad	integración	-	
Cantidad	Del sistemas	-	
			total

3. ENTORNO Y CONFIGURACIÓN DE LAS PRUEBAS

Para el proceso de pruebas del proyecto se requiere de la disponibilidad de los siguientes entornos, a saber:

a. Sistema Operativo del servidor.
   - Caracteristicas del equipo.

b. Equipos Cliente: Equipos de Prueba.
   - Caracteristicas de los equipos de pruebas.

c. Base de Datos a usar

3.1. CRITERIOS DE INICIO

Aceptación del plan de pruebas**.** Revisión y aceptación del documento que contiene los casos de pruebas para la certificación del proyecto.

Aceptación de paquetes. Revisión y aceptación de los paquetes de desarrollo, y que este cumpla con las condiciones de aceptación.

Aceptación de ambiente. Revisión y aceptación del ambiente de certificación, y que este cumpla con las condiciones de aceptación.

3.2. BASES DE DATOS DE PRUEBAS

Base de Datos :

Nombre Servidor

BD :

Nombre

3.3. CRITERIOS DE APROBACION / RECHAZO

- ✓ **Errores Graves:** información crítica presentada erróneamente, información mal registrada en la base de datos, caídas de programas, incumplimiento de objetivos en funciones principales, etc.
- ✓ **Errores Medios (comunes):** errores en documentos impresos que se entregan a personas ajenas a la organización, errores en presentación de datos, incumplimiento de objetivos en funciones secundarias, caídas de programas auxiliares, etc.
- ✓ **Errores Leves:** errores en presentación de datos secundarios, no adecuación a estándares, comportamientos correctos pero diferentes en situaciones similares, dificultades de operación, etc.

4. ESTRATEGIA DE PRUEBAS

Indicar y detallar el número de fases o etapas y las herramientas de pruebas que usaras asi como la distribución de los recursos y tiempos.

4.1. ESCENARIO DE LAS PRUEBAS

Especificar la cantidad de escenarios de pruebas y cuales son (ejemplo: tres escenarios de pruebas las cuales son: Pruebas de Instalación, Pruebas de GUI o Interfaz y Pruebas de Operación o Funcionales)

- Para las Pruebas de Instalación se debe comprobar que (Ejemplo):
  - Aplicación no presenta anomalías.
  - Que apunta al servidor y base de datos definidos.

- Para las pruebas de GUI se debe comprobar que:
  - Comportamiento de aplicación con casos de bordes inválidos y válidos, donde las pruebas de borde se definen como aquellas pruebas en las cuáles los datos de prueba a utilizar son valores límites.

  - Carga, despliegue, foco, modalidad, navegabilidad y usabilidad de las GUI del Sistema y sus elementos. Donde las métricas y Heurísticas de usabilidad y funcionalidad a utilizar son las siguientes:
    - Comprensión Global del Sitio.
    - Aspectos de Interfaces y Estéticos.
    - Métricas de confiabilidad.
    - Navegación y Exploración.

Para las pruebas de Operación o Funcionales se debe comprobar:

- El comportamiento de aplicación con casos inválidos y válidos, de flujo completo del proceso de las propuestas y proyectos.
- El comportamiento de aplicación con casos inválidos y válidos, de flujo completo del proceso de los documentos generados por el consejo.

- El comportamiento de aplicación con casos inválidos y válidos, de flujo completo del proceso de las diferentes acitividades relacionadas a una propuesta y proyecto de titulación.
- El comportamiento de la aplicación para el A
- El comportamiento de la aplicación para el B
- El comportamiento de la aplicación para el C

4.2. ORDEN DE EJECUCIÓN DE PRUEBAS

Las pruebas se llevarán a cabo de la siguiente forma:

- **Secuencias de pasos para la Configuración**

  Configuración de los Equipos Cliente y del Servidor de Aplicación Web y de Base de Datos.

- **Secuencias de pasos para la generación de archivos para los x módulos.**

  Si en caso hubiera procesos que generen algún archivo y que se deben ejecutar

- **Secuencias de pasos para la generación de datos para los x módulos.**

  Ejecución del proceso (manual) de generación de datos, donde las tablas y campos a utilizar serán llenados manualmente.

4.3. EQUIPO DE PRUEBAS Y RESPONSABILIDADES

Nombre	Responsabilidad
Nombre del arquitecto del producto (si en caso hubiera)	Arquitecto de Producto, responsable de evaluar las condiciones de término para el proceso de pruebas junto al Jefe de Proyectos.
Nombre del jefe del proyecto	Jefe de Proyectos, responsable de evaluar las condiciones de término para el proceso de pruebas junto al Arquitecto de Producto.
Nombre del analista funcional (si en caso hubiera)	Analista funcional, responsable de la resolución de las incidencias de certificación para los módulos de Proyectos, Revisión y Aprobación.
Nombre de los testeadores	Testing de Solución, responsable de la generación del plan de pruebas.