Universidad de las Ciencias Informáticas

Facultad 6

Título: Diseño y Aplicación de Pruebas a la Plataforma VideoWeb 1.0

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor: Ivania Cortina Herrera

Tutor: Ing. Eridniel Suárez Contreras

Ciudad de La Habana, Junio del 2011.

"Año 53 de la Revolución"

Agradecimientos

A mi Mamá, porque es lo que más quiero en la vida y por comprenderme siempre.

A mi Papá, por quererme tanto y seguir a mi lado confiando en mí.

A mi novio José Angel, por haberme ayudado infinitamente, por estar a mi lado compartiendo buenos y malos momentos por cuatro años.

A mis hermanos Tito y Joan, que son mi razón de ser.

A mi Tía Tita, por guiarme en todo momento y ser mi segunda madre.

A mi Tío Luis, por ser un ejemplo de superación para mí y por el apoyo que me brindó cuando más lo necesité.

A mi suegra Daris, porque la distancia no ha sido suficiente obstáculo para impedirle estar cerca de nosotros en todo momento, le agradezco por su preocupación, apoyo y por quererme como a una hija.

A mis primas Neni y Yoa, por ser como hermanas para mí.

A mis abuelitos Mami y papi, por creer en mí siempre.

A mis abuelos Luis y Virginia por su sabiduría y por darme un lugar en su corazón.

A Yara, Aliuska, Elizabetha por su amistad incondicional, por apoyarme en todo momento.

A todos mis amigos del 9108, siempre estarán presente en mi vida, especialmente Yasiel, Lisette, Eduardo, Siomelis, José Carlos, Oslanier, Yudalis, Alain, Eduanis, Pedro, Cralos.

A todos los profesores que han contribuido a mi formación profesional.

A todos los que confiaron en mí.

Dedicatoria

A mis padres por ser mis mejores amigos y apoyarme en los momentos más difíciles, siempre demostrándome su amor, apoyo y confianza.

Declaración de autoría

Declaro que soy la única autora del trabajo titulado: “Diseño y Aplicación de Pruebas a la Plataforma VideoWeb 1.0” y autorizo a la Facultad 6 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de Junio del año 2011.

Ivania Cortina Herrera

_____

Ing. Eridniel Suárez Contreras

_____

Resumen

La calidad del software es un aspecto fundamental que se tiene en cuenta en la producción de los mismos. Se pretende que los productos lleguen a mano de los usuarios con la mayor calidad posible. La plataforma VideoWeb, permite la gestión y transmisión de contenidos multimedia a través de la red utilizando tecnología streaming. VideoWeb es una solución tecnológica integral que le permite al cliente la transmisión, administración y publicación de contenido multimedia en la web utilizando tecnologías libres. El presente trabajo tiene como principal objetivo diseñar una estrategia de pruebas, debido a su vital importancia para asegurar y garantizar en gran medida el buen desempeño de este software. Se confeccionaron todos los diseños de casos de pruebas tanto de caja negra como de caja blanca; también se describió el procedimiento a seguir para la realización de las pruebas de usabilidad y se configuró un Plan de Pruebas en la herramienta seleccionada para realizar pruebas de carga y estrés, simulando la conexión de usuarios concurrentes. Se estableció un proceso de prueba lo más completo posible, tratando de detectar la mayor cantidad de errores. Se documentaron todos los resultados obtenidos durante la aplicación de las pruebas planificadas.

Palabras Claves

Calidad, Casos de pruebas, Pruebas de software, VideoWeb (VW).

Abstract

A fundamental aspect taken into account in their production is the Software quality. It is intended that the products reach the hands of users with the highest possible quality. The VideoWeb platform allows the management and delivery of multimedia content over a network using streaming technology. VideoWeb is a comprehensive technology solution that allows the customer the delivery, management and publishing of multimedia content on the web using free technologies. The paper's main objective is to design a test strategy due to its vital importance to ensure and guarantee a large extent the good performance of this software. Were made all test case designs both black box as well as white box, also described the procedure for conducting usability testing and set up a test plan in the selected tool for load testing and stress, simulating concurrent users connection. It was established a testing process as complete as possible, trying to detect as many errors. Have been documented all results obtained during the implementation of the planned testing.

Keywords

Quality, Test cases, software testing, VideoWeb (VW)

Índice de contenido

Índice de Tablas

Índice de Figuras

Introducción

Desde los tiempos más remotos, el hombre se ha visto obligado mediante el trabajo, a hacerse la vida más fácil y sencilla, buscando poder hacer más cosas con vista a lograr un mayor desarrollo que le traiga beneficios y bienestar. Según fue creciendo el desarrollo industrial fue imposible controlar de forma manual el flujo de información que este generaba. Con todos estos cambios significativos el hombre se dio cuenta de lo importante que podía ser el proceso de automatizar la información. Este empezó a apoyarse en máquinas desde el ábaco hasta los primeros sistemas informáticos. El auge de la informática a nivel mundial va en ascenso, logrando cada vez más éxito en la informatización de los servicios.

Gracias a la creación de los primeros sistemas informáticos se ha logrado en pocos años un avance sin igual en las Tecnologías de la Información y las Comunicaciones (TIC), lo que ha permitido hacer menos complejo el trabajo en bancos, centros comerciales, científicos y en general han mejorado la calidad de vida a todas la personas que se benefician con los avances de las TIC.

A medida que los grandes consorcios de software fueron desarrollándose, fue inminente el desarrollo de técnicas que garantizaran el correcto funcionamiento de sus aplicaciones. Con la aparición de estos grandes sistemas informáticos se incluyó un nuevo proceso en la confección de los mismos, el proceso de prueba. Este proporciona orientación sobre cómo evaluar y valorar la calidad del producto. Muchas empresas de software han avanzado en la actualidad en cuanto a tecnologías y desarrollo de software. La calidad de estos ha aumentado en sentido general, por tal razón es que se hace imprescindible desarrollar productos con calidad para que estos estén a la altura y puedan adentrarse en el mercado nacional e internacional.

En Cuba se redoblan esfuerzos con vista a lograr un desarrollo importante en el área de la informática con el fin de disminuir a cero la importación de software propietario, el que representa grandes gastos para el país. Se han llevado a cabo muchos proyectos para lograr este desarrollo como lo son la creación de empresas de desarrollo de soluciones informáticas y la Universidad de las Ciencias Informáticas (UCI).

En la Universidad de las Ciencias Informáticas se forman profesionales informáticos altamente capacitados para el desarrollo de software en Cuba. La universidad cuenta con innumerables proyectos productivos comprometidos con la calidad de los productos tanto para la producción nacional como de exportación. Debido al poco tiempo que tiene la universidad insertada en algunos mercados

es fundamental ganar prestigio brindando productos de máxima calidad pues es la única manera de desarrollarse y ser reconocidos a niveles mucho más altos y así lograr insertarse en los mercados de software más prestigiosos del mundo con vista a exportar más y lograr tener un fuerte impacto económico-social y dar un aporte constante y sólido al país que tanto lo necesita.

Uno de los productos con que cuenta la UCI, en la facultad 6 dentro del Centro de Desarrollo "GEYSED", en el departamento de Señales Digitales es: VideoWeb, plataforma[1] para la transmisión de contenido audiovisual por la red de datos. VideoWeb le ofrece a clientes interesados en tener acceso a información mediática con un menor costo de espacio y que además puedan disponer de esta información sin que sea necesaria la copia física de la misma en los ordenadores clientes, una solución integral utilizando la tecnología streaming[2] y enfocada a entornos de software libre. Se brindan servicios de video y audio sobre Internet e incorpora gestores tanto de usuarios, de contenido, documentales, como de información audiovisual.

Se puede encontrar aplicaciones similares a esta como son youtube.com, tv.com, VideoWeb Perú e Inter-nos. Es necesario que la plataforma VideoWeb supere la funcionalidad de las aplicaciones antes mencionadas con el fin de introducirse en el mercado internacional. Constituye un reto para los desarrolladores lograr una aplicación que compita con otros sistemas creados por grandes empresas de software a nivel mundial.

Un software de alta calidad es esencial para el éxito de los sistemas que funcionan en la red de datos, donde una anomalía puede provocar pérdida de datos e insatisfacción con el cliente. Es posible que el impacto de un defecto provoque daños considerables en la empresa que utiliza el software, en beneficios perdidos y posiblemente, costes legales. Con una creciente demanda de servicios de medias a través de Internet, muchos sistemas requieren un exhaustivo proceso de prueba, pues la aparición de anomalías en estos no le permitiría desempeñar sus funciones, conllevando a que las empresas pudieran tener desperfectos en su proceso productivo.

---

[1] *"Conjunto de subsistemas o tecnologías que proporcionan un conjunto coherente de funciones a través de interfaces y patrones de uso especificado que cualquier subsistema que depende de plataforma puede utilizar sin preocuparse de los detalles de cómo se implementan las funciones que proporciona la plataforma."* (19)

[2] *Es una tecnología para transmitir video o audio por la web. Permite la reproducción de archivos multimedia sin la necesidad de copiarlo completamente.*

Se identifica como **problema a resolver**: ¿Cómo reconocer la mayor cantidad de errores posibles en el producto VideoWeb que posibilite la verificación y validación de los requisitos definidos para el sistema?

Para validar la calidad del producto es necesario realizar un conveniente diseño de pruebas, por tanto el **objeto de estudio** de la investigación es: el proceso de pruebas en el desarrollo de software; enmarcando el **campo de acción** en la plataforma VideoWeb.

El **objetivo general** de este trabajo es diseñar y aplicar pruebas de software al producto VideoWeb que permitan revelar hasta qué punto las funciones del software operan de acuerdo con las especificaciones y requisitos del cliente.

Para darle cumplimiento a este objetivo general se plantean las siguientes **tareas de la investigación**:

1. Analizar los distintos tipos de pruebas más comunes que se emplean en el desarrollo de aplicaciones Web.
2. Caracterizar los tipos de pruebas más comunes empleadas en el desarrollo de aplicaciones Web para realizar la fundamentación teórica de la investigación.
3. Analizar los diferentes métodos, niveles y tipos de pruebas que se deben aplicar a la plataforma VideoWeb.
4. Desarrollar una estrategia de pruebas para la evaluación plataforma VideoWeb.
5. Diseñar los casos de prueba.
6. Aplicar las pruebas seleccionadas a la plataforma VideoWeb.
7. Documentar los resultados arrojados por las pruebas.

Se plantea la siguiente afirmación como **idea a defender**: La aplicación de las pruebas de software a la plataforma VideoWeb permitirá la verificación y validación de los requisitos funcionales definidos.
Para darle cumplimiento a las tareas propuestas se utilizaron los siguientes métodos de investigación.

Teóricos:

- **Histórico lógico:** este método se usó con el fin de recopilar la información que se posee hasta el momento sobre las pruebas y resumir los aspectos fundamentales necesarios para la investigación en curso.
- **Analítico sintético:** para un mejor estudio de la situación y poder analizar las características del proceso de pruebas dentro del flujo de desarrollo del software.

Empíricos:

- **Observación Científica:** este se utilizó en toda la investigación, pues se recoge información de los aspectos tratados en la tesis desde el punto de vista de otros autores así como sus definiciones y resultados.
- **Entrevista:** este método se utilizó con el objetivo fundamental de seleccionar las pruebas de software usadas en la universidad para la liberación de aplicaciones web.

Capítulo 1

Fundamentación Teórica

1.1. Introducción

Probar bien un sistema no es una actividad trivial, requiere de mucha experiencia y práctica, son muchas las pruebas que se encuentran para la evaluación de los sistemas. Existen distintos niveles de pruebas, que se realizan en diferentes momentos del ciclo de vida del software. Algunos de estos niveles contienen tipos de pruebas específicas. Aunque no hay una clasificación oficial o formal acerca de los diversos tipos de pruebas de software, en este capítulo se caracterizan los más comunes y usados para mostrar hasta qué punto las funciones del software operan de acuerdo con las especificaciones y requisitos del cliente. También se exponen los principales conceptos y definiciones de diferentes autores relacionados con las pruebas de software y el proceso de pruebas en general.

1.2. Calidad de software

La calidad es una propiedad y cualidad inherente de las cosas, que permite la comparación entre éstas y otras de su misma especie. Se trata de una apreciación subjetiva que, respecto a un usuario, implica satisfacer las necesidades y deseos, si lo logra, es de buena calidad.

La calidad del software varía de un sistema a otro pero de forma general se puede decir que la calidad del software la conforman un conjunto de cualidades que los caracterizan y determinan su utilidad. La calidad se puede expresar como eficiencia, flexibilidad, corrección, confiabilidad, portabilidad, usabilidad, seguridad e integridad.

La ISO 8402 (International Standard Organización), define la calidad como la ausencia de deficiencias: "Es la totalidad de aspectos y características de un producto o servicio que se refieren a su capacidad para satisfacer necesidades dadas en la adecuación de sus objetivos".

Otra definición que propone Pressman plantea que "la calidad es la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo

explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente" (1)

También se puede definir la calidad de software como "el conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas" (2)

Todos los conceptos expuestos hacen referencia a lo que es la calidad de software por lo que se puede definir de forma general que la calidad de software es el grado con el que un software cumple con las necesidades del cliente, que cuenta con una serie de parámetros definidos y lo dotan de cualidades únicas, cumpliendo la totalidad de características de un producto, posibilitando satisfacer las necesidades explícitas o implícitas del usuario. Para garantizar la calidad en los sistemas es necesario la aplicación de pruebas de software, para tal razón es necesario analizar en qué consisten las pruebas de software.

1.2.1. ¿Qué es una prueba de software?

Las pruebas deben realizarse durante el desarrollo del software, con el objetivo de certificar la calidad del producto. Las mismas son diseñadas y aplicadas para evaluar el comportamiento del sistema y comprobar que el software cumple con lo establecido y como está descrito. A un sistema se le realizan pruebas para identificar a tiempo las posibles fallas que pueda presentar y corregirlas en el menor tiempo posible para que esto no suponga un costo mayor más adelante. En muchas ocasiones las pruebas tienen el objetivo de forzar intencionalmente al programa a producir errores.

Las pruebas de software pueden ser definidas como:

"Un elemento crítico para la garantía de calidad del software y representa una revisión de las especificaciones, del diseño y de la codificación".(3)

"Las pruebas consisten en la verificación dinámica del comportamiento de un programa en una serie finita de casos de pruebas". (4)

"Una actividad en la cual un sistema o uno de sus componentes se ejecuta en dos o más circunstancias previamente especificadas, los resultados se observan y registran y se realiza una evaluación de algún aspecto". (5)
Se puede concluir que las pruebas de software son las acciones que se llevan a cabo con el fin de encontrar errores en un sistema y comprobar el funcionamiento del mismo. Para llevar a cabo estas

pruebas es necesario diseñar cada caso de prueba que será usado en la ejecución de las pruebas seleccionadas, por esto se hace inevitable analizar en qué consisten los casos de pruebas.

1.2.2. Casos de Pruebas

Los casos de pruebas son un conjunto de condiciones o variables que permiten determinar si los requisitos del sistema han sido parcial o completamente satisfactorios. Se pueden realizar muchos casos de prueba para determinar que un requisito es completamente satisfactorio. Con el propósito de comprobar que todos los requisitos de una aplicación son revisados, debe haber al menos un caso de prueba para cada requisito. RUP recomiendan crear por lo menos dos casos de prueba para cada requisito. Uno de ellos debe realizar la prueba positiva de los requisitos y el otro debe realizar la prueba negativa.

Si la aplicación es creada sin requisitos formales, entonces los casos de prueba se escriben basados en la operación normal de programas de una clase similar. Un caso de prueba tiene éxito si logra encontrar un error no descubierto hasta el momento de la ejecución del mismo.

1.2.3. Términos defecto, falla y error

Los términos defecto, falla y error son usados indistintamente por muchas personas, pero en realidad cada uno tiene su significado específico en el marco del proceso de desarrollo de software. A continuación se definen cada uno de ellos enmarcados en el proceso de pruebas.

El **error** puede ser definido como:

“La diferencia entre un valor calculado, observado o medido y el valor verdadero, especificado o teóricamente correcto.” (6)

Un error es una equivocación cometida por uno de los desarrolladores, algunos errores pueden ser: una malinterpretación de un requisito o de la funcionalidad de un método. El estándar 829 de la IEEE[3] (Institute of Electrical and Electronics Engineers) y el diccionario de la Real Academia de la Lengua Española (RAE) definen un error como “una idea falsa o equivocada”. Los errores son propios de las personas, pues los programas no pueden estar en un error, sino que son sus desarrolladores los que pueden cometer errores.

[3] Asociación de profesionales norteamericanos que aporta criterios de estandarización de dispositivos eléctricos y electrónicos.

Un **defecto** puede ser definido como:

“Un paso, proceso o definición de dato incorrecto en un programa de computadora. El resultado de una equivocación.” (5)

Un defecto puede encontrase en los artefactos, como resultado de una equivocación. Según la Real Academia de la Lengua Española (RAE) se define un defecto como "imperfección".

Un **fallo** puede ser definido como:

“La incapacidad de un sistema o de alguno de sus componentes para realizar las funciones requeridas dentro de los requisitos de rendimiento especificados.” (6)

Un fallo según la IEEE es la disconformidad evidente que se origina al ejecutar un programa con un defecto.

La siguiente figura muestra la relación entre error, defecto y fallo:

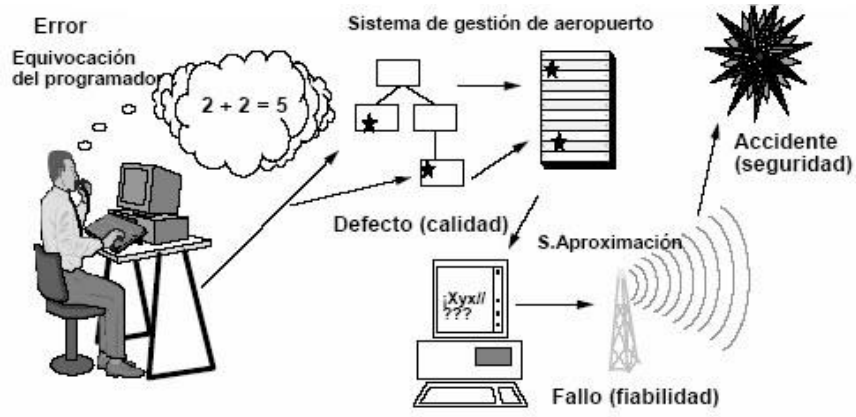


Figura 1: Relación entre error, defecto y fallo.

Luego de haber analizado estos conceptos es importante precisar que cuando se va a comprobar la eficacia de un producto, es necesario tener en cuenta una serie de factores para poder probar la calidad del mismo. Por tal razón es necesario que estos sean examinados detalladamente con el fin de saber cuáles son y por qué precisamente con ellos se evalúa la calidad de un producto.

1.3. Factores que miden la calidad

Para establecer la calidad de un software es importante tener en cuenta los factores que estipulan la misma. Estos se dividen en tres grupos teniendo en cuenta tres aspectos importantes de un producto de software. En la siguiente tabla se exponen los factores de calidad definido por McCall[4] en 1977, aun cuando fueron definidos en 1977 siguen teniendo vigencia.

Tabla 1: Factores que miden la calidad del software (7)

Factores que miden la calidad del software		
Aspecto	**Factor**	**Descripción**
Características operativas	Corrección	Hasta dónde satisface un programa su especificación y logra los objetivos del cliente.
	Fiabilidad	Hasta dónde se puede esperar que un programa lleve a cabo la función con la que fue concebido, con la exactitud requerida.
	Eficiencia	La cantidad de recursos informáticos y de código necesarios para que un programa funcione.
	Seguridad	Hasta dónde se puede controlar el acceso al software o a los datos por personas no autorizadas.
	Facilidad de uso	Aprender a operar los datos de entrada e interpretar las salidas de un programa.
Capacidad de soportar los cambios	Facilidad de mantenimiento	La capacidad para localizar y arreglar un error en un programa.

[4] Autor de Factors in Software Quality: General Electric.

	Flexibilidad	El esfuerzo necesario para modificar un programa operativo.
	Facilidad de prueba	El esfuerzo necesario para probar un programa para asegurarse de que realiza su función pretendida.
Adaptabilidad a nuevos entornos	Portabilidad	La facilidad para transferir el programa de un entorno de sistema hardware y/o software a otro entorno diferente.
	Reusabilidad	Hasta dónde se puede volver a emplear un programa (o partes de un programa) en otras aplicaciones.
	Interoperabilidad	El esfuerzo para acoplar un sistema con otro.

En la tabla 2 se muestran las métricas planteadas por McCall para comprobar los factores de calidad, ya que es difícil establecer medidas directas de los factores anteriormente señalados, según el factor, se utilizarán unas determinadas métricas ponderadas para comprobar ese factor.

Tabla 2: Métricas planteados por McCall (7)

Métricas planteadas por McCall	
Métrica	**Descripción**
Facilidad de auditoría	La facilidad con que se puede comprobar la conformidad con los estándares.
Exactitud	La precisión de los cálculos y del control.
Normalización de las comunicaciones	El grado en que se usa el ancho de banda, los protocolos y las interfaces estándar.

Completitud	El grado en que se ha conseguido la total implantación de las funciones requeridas.
Concisión	Lo compacto que es el programa en términos de líneas de código.
Consistencia	El uso de un diseño uniforme y de técnicas de documentación a lo largo de todo el programa.
Estandarización en los datos	El uso de estructuras de datos y de tipos estándar a lo largo de todo el programa.
Tolerancia de errores	El daño que se produce cuando el programa detecta una situación errónea.
Eficiencia en la ejecución	El rendimiento en tiempo de ejecución de un programa.
Facilidad de expansión	El grado en que se puede ampliar el diseño arquitectónico, de datos o procedimental.
Generalidad	La amplitud de aplicación potencial de los componentes del programa.
Independencia del hardware	El grado en que el software es independiente del hardware sobre el que opera.
Instrumentación	El grado en que el programa muestra su propio funcionamiento e identifica errores que aparecen
Modularidad	La independencia funcional de los componentes del programa.
Facilidad de operación	La facilidad de utilización de un programa.
Seguridad	La disponibilidad de mecanismos que controlen o protejan los programas o los datos.
Auto-Documentación	El grado en que el código fuente proporciona documentación

	significativa.
Simplicidad	El grado en que un programa puede ser entendido sin dificultad.
Independencia del sistema de software	El grado en que el programa es independiente de características no estándar del lenguaje de programación, de las características del sistema operativo y de otras restricciones del entorno.
Facilidad de traza	La posibilidad de seguir la pista a la representación del diseño o de los componentes reales del programa hacia atrás, hacia los requisitos.
Formación	El grado en que el software ayuda a permitir que nuevos usuarios empleen el sistema.

1.4. Objetivos de las pruebas

Todos los sistemas presentan errores debido a que estos son desarrollados por humanos quienes pueden cometer equivocaciones, pero es en la fase de pruebas donde todos estos posibles errores cometidos se descubren y se corrigen.

El diseño y ejecución de pruebas tienen como objetivo:

- Encontrar el mayor número de errores con la menor cantidad de tiempo y esfuerzo posibles.
- Mostrar hasta qué punto las funciones del software operan de acuerdo con las especificaciones y requisitos del cliente.
- Mejorar la calidad del producto del software.
- Validar y controlar formalmente la calidad del trabajo realizado.
- Encontrar y documentar los defectos que puedan afectar la calidad del software.
- Validar que el software trabaje como fue diseñado.
- Validar y probar los requisitos que debe cumplir el software.

- Validar que los requisitos fueron implementados correctamente.

1.5. Principios de pruebas

A continuación se exponen diez de los principios de realización de pruebas más relevantes que se tienen en cuenta a la hora de diseñar y aplicar pruebas de software a un sistema. Estos principios son necesarios tenerlos en cuenta en el momento de diseñar los casos de pruebas ya que orientan y guían a los responsables de las pruebas.

1. La prueba puede ser usada para mostrar la presencia de errores, pero nunca de su ausencia.
2. La principal dificultad del proceso de prueba es decidir cuándo parar.
3. Evitar casos de pruebas no planificados, no reusables y triviales a menos que el programa sea verdaderamente sencillo.
4. Una parte necesaria de un caso de prueba es la definición del resultado esperado.
5. Los casos de pruebas tienen que ser escritos no solo para condiciones de entrada válidas y esperadas sino también para condiciones no válidas e inesperadas.
6. Los casos de pruebas tienen que ser escritos para generar las condiciones de salida deseadas.
7. El número de errores sin descubrir es directamente proporcional al número de errores descubiertos.
8. Las pruebas deberían empezar por “lo pequeño” y progresar hacia “lo grande”.
9. Con la excepción de las pruebas de unidad e integración, un programa no deberá ser probado por la persona u organización que lo desarrolló.
10. Las pruebas deberían ser realizadas por un equipo independiente.

1.6. Características de las pruebas

Kaner, Falk y Nguyen (8) coinciden en que algunas de las características que no deben faltar en una buena prueba son:

- Una buena prueba tiene una alta probabilidad de encontrar un error**.** El ingeniero de software debe tener un alto nivel de entendimiento del software para poder diseñar buenos casos de prueba que encuentren el mayor número de defectos.

- Una buena prueba no debe ser redundante. Uno de los objetivos de las pruebas es “encontrar el mayor número de errores con la menor cantidad de tiempo y esfuerzo posibles”, por lo cual no se deben diseñar casos de prueba que tengan el mismo propósito que otros sino que se debe buscar diseñar el menor número de casos de prueba que permitan probar adecuadamente el software y que permitan optimizar los recursos.

- La limitación en tiempo y recursos puede impedir que se ejecuten todos los casos de prueba de un grupo de pruebas similares por lo cual en estos casos se debería seleccionar la prueba que tenga la mayor probabilidad de descubrir errores.

- Una buena prueba no debería ser ni demasiado sencilla ni demasiado compleja.

Las pruebas de software necesitan de una guía para que el proceso se desarrolle con la máxima eficiencia posible, es entonces donde juega un papel muy importante la metodología de desarrollo que se utilizará en el proyecto. Esta es la encargada de dirigir de cierta manera las pruebas de software, definiendo el cuándo y el tiempo que se le dedicará a cada una de las pruebas.

1.7. Metodología de Desarrollo

Una metodología de desarrollo representa el camino a seguir para la construcción de software de forma sistemática. Una metodología está conformada por un conjunto de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto. El objetivo de estas metodologías es guiar a los desarrolladores al crear un nuevo software y lograr mejores aplicaciones mediante un proceso de desarrollo más organizado que permita planificar y controlar el proyecto. Debido a que los requisitos de un software a otro son tan variados que ha dado lugar a que exista una gran variedad de metodologías para la creación de los mismos, se podrían clasificar en dos grandes grupos:

1. Las metodologías orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Estas metodologías son llamadas metodologías pesadas.

Ejemplos de metodologías pesadas: Rational Unified Process (RUP), Microsoft Solution Framework (MSF), Métrica.

2. Las metodologías orientadas a la interacción con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de

tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando. Estas son llamadas metodologías ligeras/ágiles.

Ejemplos de metodologías ágiles: Extreme Programming (XP), Scrum, Cristal Methods, Feature Driven Development.

1.7.1. Rational Unified Process (RUP)

La metodología más usada en la universidad para el desarrollo de software es la metodología RUP, y es la misma que guía el proceso de desarrollo de la plataforma VideoWeb.

RUP es una recopilación de prácticas de ingeniería de software que se están mejorando continuamente de forma regular para reflejar los cambios en las prácticas de la industria. Esta metodología se puede utilizar desde el principio de un nuevo proyecto de software y puede seguir utilizándose en los ciclos de desarrollo subsiguientes. La clave para lograr un equilibrio entre desarrollar un software con calidad y desarrollarlo con la mayor rapidez posible es entender los elementos esenciales del proceso y seguir ciertas directrices para adaptar el proceso a las necesidades específicas del usuario. RUP utiliza UML[5] como lenguaje de modelado.
Los principales elementos que define RUP en su modelado son:

Tabla 3: Principales elementos que define RUP

Principales Elementos	
Elementos	**Descripción**
Trabajadores	Define el comportamiento y responsabilidades de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades.
Actividades	Tareas que tienen un orden y un propósito determinado, es realizada por un trabajador y manipula elementos.

[5] Es un lenguaje para visualizar, especificar, construir y documentar los artefactos que se crean durante el proceso de desarrollo.

Artefactos	Resultados tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
Flujos de trabajo	Secuencia de actividades desarrolladas para producir determinados artefactos, que aumentarán el desarrollo del proyecto.

La metodología RUP divide en 4 fases el desarrollo del software:
Inicio: objetivos del ciclo vital.

Elaboración: objetivo de la arquitectura del ciclo de vida.

Construcción: capacidad operacional inicial.

Transición: Liberación del producto.

Es importante mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevada bajo flujos de trabajo, estos forman dos grupos:

Flujos de trabajo de ingeniería

- Modelación de Negocios: entender las necesidades del negocio.
- Requisitos: transformar las necesidades del negocio a un sistema automatizado.
- Análisis y Diseño: transformar los requisitos en un diseño de cómo se va a implementar el sistema.
- Implementación: definir la organización del código, en términos de los subsistemas de implementación organizados en capas.
- Pruebas: proporciona orientación sobre cómo evaluar y valorar la calidad del producto.
- Despliegue: describe las actividades asociadas al garantizar que el producto de software esté disponible para los usuarios.

Flujos de trabajo de apoyo

- Configuración y Gestión de cambios: explica cómo controlar y sincronizar la evolución del conjunto de productos de trabajo que componen un sistema de software.
- Gestión de proyectos: planificación del proyecto, la gestión del riesgo, la supervisión del progreso y la métrica.

- Entorno: proporciona el entorno de soporte para un proyecto.

A continuación se muestra una imagen con los flujos de trabajo que define RUP para cada una de las fases.

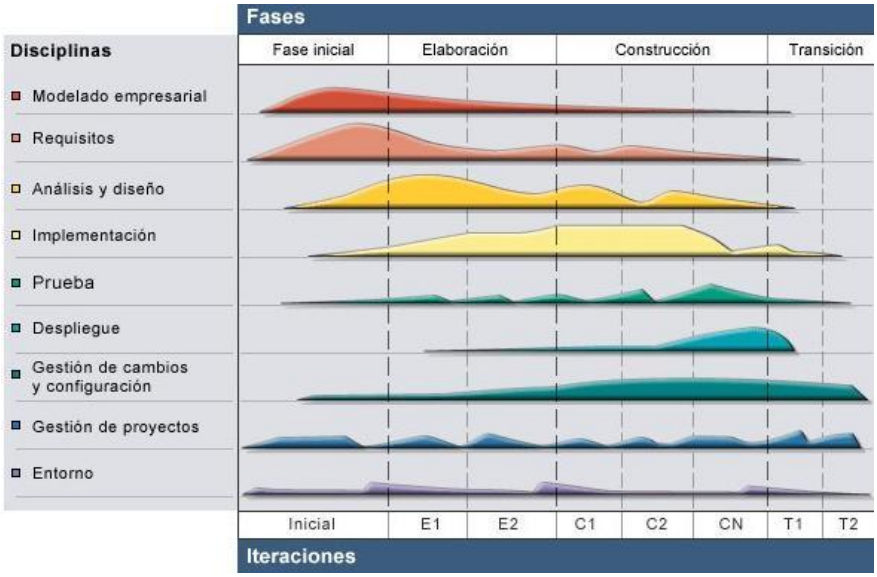


Figura 2: Diagrama flujos de trabajos RUP.

1.7.2. El proceso de pruebas en RUP

El objetivo de la disciplina de pruebas es asegurar la calidad del producto desarrollado, además es en esta etapa donde se verifica que todos los requisitos se hayan implementado adecuadamente según lo descrito en su especificación. Igualmente en esta etapa es preciso diseñar las pruebas que identificarán los posibles defectos y fallas en el sistema.

1.7.2.1. Niveles de Pruebas

Las pruebas son aplicadas para diferentes objetivos, en diferentes escenarios o niveles de trabajo. RUP define los siguientes niveles de pruebas:

Prueba de Desarrollador

Indica los aspectos de diseño e implementación más adecuados que debe llevar a cabo el equipo de desarrolladores. La ejecución de la prueba se produce inicialmente con el grupo de pruebas del desarrollador que la diseñó e implementó.

Prueba Independiente

Las pruebas independientes indican el diseño y la implementación de las pruebas realizadas más

adecuadamente por alguien ajeno al equipo de desarrolladores. Puede considerar esta distinción un súper conjunto, que incluye validación y verificación independientes. En la mayoría de los casos, la ejecución de la prueba se produce inicialmente con el grupo de pruebas independientes que la diseñó e implementó.

Prueba de Unidad

Se centra en la verificación de los elementos más pequeños del software que se puedan probar. Normalmente, las pruebas de unidad se aplican a componentes representados en el modelo de implementación para verificar que se cubren los flujos de control y los flujos de datos y que funcionan como se esperaba. La prueba de unidad siempre está orientada a caja blanca.

Prueba de Integración

Es ejecutada para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados para ejecutar un caso de uso. Se prueba un paquete o un conjunto de paquetes del modelo de implementación. Estas pruebas descubren errores en las especificaciones de las interfaces de los paquetes. Esta prueba debe ser responsabilidad de desarrolladores y de independientes, sin solaparse las pruebas.

Prueba de Sistema

Son las pruebas que se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar la aplicación, después que todos los componentes de software y hardware han sido integrados. En un ciclo iterativo, estas pruebas ocurren más temprano, tan pronto como subconjuntos bien formados de comportamiento de caso de uso son implementados.

Prueba de Aceptación del Usuario

Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido. Esta prueba también es conocida como “Prueba Piloto”.

1.7.2.2. Tipos de pruebas

RUP define qué pruebas usar para la validación de los sistemas y cómo realizar su ejecución, teniendo en cuenta un conjunto de artefactos y trabajadores.

Tabla 4: Tipos de pruebas basados en las dimensiones de calidad

Tipos de pruebas basados en las dimensiones de calidad

Funcionalidad	**Prueba de función:** Se centran en la validación de funciones, los métodos y los servicios necesarios.
	Prueba de seguridad: Se centran en garantizar que los datos del destino de la prueba (o sistemas) sólo son accesibles para los actores a los que se dirigen.
	Prueba de volumen: Se centran en la verificación de la capacidad para manejar grandes cantidades de datos, ya sean de entrada y salida o residentes, en la base de datos.
Usabilidad	**Prueba de usabilidad:** Pruebas que se basan en factores humanos, estética, coherencia de la interfaz de usuario, ayuda en línea y según contexto, asistentes y agentes, documentación de usuario, materiales de formación.
Fiabilidad	**Prueba de integridad:** Enfocada a la valoración de la robustez (resistencia a fallos).
	Prueba de estructura: Pruebas que se centran en la evaluación de la adherencia a su diseño y formación. Normalmente, esta prueba se realiza en aplicaciones web y garantiza que todos los enlaces están conectados, se muestra el contenido adecuado y no hay ningún contenido huérfano.
	Prueba de tensión: Se centra en la evaluación de cómo responde el sistema en circunstancias anormales. Las tensiones del sistema pueden ser cargas de trabajo extremas, memoria insuficiente, servicios y hardware no disponibles o recursos compartidos limitados.
Rendimiento	**Prueba de puntos de referencia:** Se trata de un tipo de prueba de rendimiento que compara el rendimiento de un destino de la prueba nuevo o desconocido con una referencia conocida, carga de trabajo y sistema.

	Prueba de contienda: Pruebas que se centran en la validación de la capacidad para manejar de forma aceptable varias demandas del actor en el mismo recurso.
	Prueba de carga: Se trata de un tipo de prueba de rendimiento que se utiliza para validar y evaluar la aceptabilidad de los límites operativos de un sistema bajo cargas de trabajo variables, mientras el sistema que se está probando permanece igual.
	Perfil de rendimiento: Se trata de una prueba en la que se controla el perfil de tiempo del destino de la prueba, incluidos el flujo de la ejecución, el acceso de datos, las llamadas del sistema y de funciones para identificar y tratar los cuellos de botella de rendimiento y los procesos ineficaces.
Capacidad de soporte	**Prueba de configuración:** Se centra en garantizar que las funciones son las adecuadas en diferentes configuraciones de hardware y software. Esta prueba también se puede implementar como una prueba de rendimiento del sistema.
	Prueba de instalación: Se centra en garantizar que se instala correctamente en diferentes configuraciones de hardware y software, y en condiciones diferentes (ejemplo, espacio de disco insuficiente o interrupciones de la alimentación).

El flujo de trabajo de prueba de RUP define pruebas para todo tipo de aplicaciones. De estas pruebas las que se deben aplicar durante el desarrollo de aplicaciones web son:

- Prueba de funcionalidad.
- Prueba de seguridad.
- Prueba de usabilidad.
- Prueba de carga.
- Prueba de tensión.
- Prueba de configuración.

- Prueba de instalación.

1.7.2.3. Técnicas de diseño de pruebas

Los niveles de prueba y los tipos de prueba que se han mencionado, necesitan de procedimientos o técnicas que lo guíen en su ejecución. A continuación se analizan cada una de estas técnicas.

Se identifican tres enfoques para el diseño de casos de prueba:

El enfoque estructural o de caja blanca consiste en centrarse en la estructura interna (implementación) del programa para elegir los casos de prueba. (6)

El enfoque funcional o de caja negra consiste en estudiar la especificación de las funciones, la entrada y la salida para derivar los casos. (6)

I. Enfoque estructural o de caja blanca

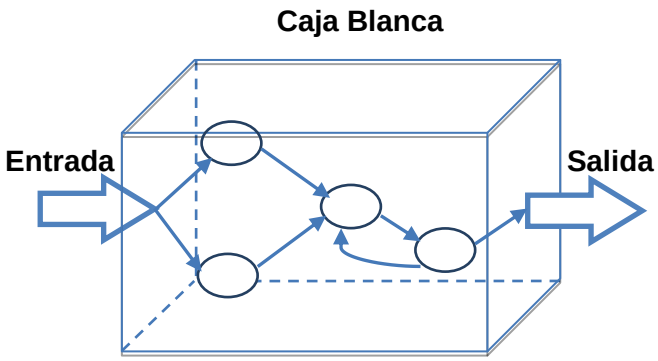


Figura 3: Enfoque de diseño de pruebas de caja blanca.

De acuerdo con Pressman, la prueba de caja blanca denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. (8)

La prueba de caja blanca del software se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o ciclos. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado. (9)

Mediante este método es posible obtener casos de prueba que:

- Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo.
- Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

Métodos de pruebas de caja blanca

Algunos de los métodos empleados en las pruebas de caja blanca son los siguientes:

1. **Prueba del camino básico:** Es una técnica que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuten por lo menos una vez cada sentencia del programa. (1)

Algunos conceptos relacionados con este método son los siguientes:

- Grafo de flujo o grafo del programa: representa el flujo de control lógico de un programa y se utiliza para trazar más fácilmente los caminos de éste. Cada nodo representa una o más sentencias procedimentales y cada arista representa el flujo de control. (10)
- Complejidad ciclomática: es una métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto de las pruebas, el cálculo de la complejidad ciclomática representa el número de caminos independientes del conjunto básico de un programa. Esta medida ofrece al probador de software un límite superior para el número de pruebas que se debe realizar para garantizar que se ejecute por lo menos una vez cada sentencia. (10)
- Camino independiente: cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una nueva condición. (10)

De forma general, los pasos que se deben seguir para la obtención de los casos de prueba en este método, son los siguientes:

1. Emplear el diseño o el código para elaborar el grafo de flujo.
2. Determinar la complejidad ciclomática del grafo de flujo.

3. Determinar un conjunto básico de caminos linealmente independientes.
4. Preparar los casos de prueba que forzarán la ejecución de cada camino del conjunto básico.

Para construir el grafo es necesario tener en cuenta las notaciones pertinentes para cada una de las instrucciones como se muestra en la figura a continuación:

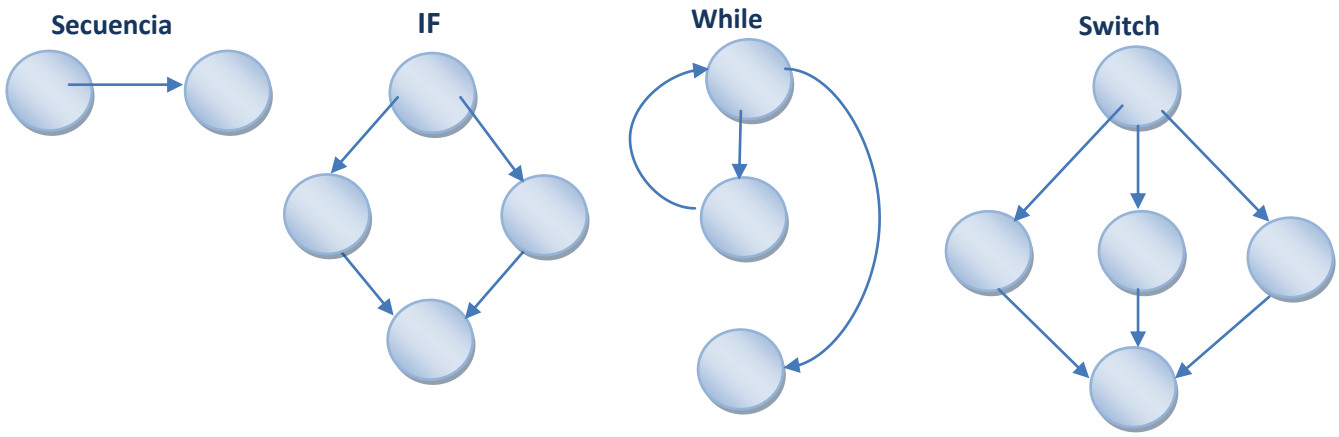


Figura 4: Notación de grafos de flujo para las instrucciones: Secuenciales, If, While, Switch.

El resultado obtenido en el cálculo de la complejidad ciclomática define el número de caminos independientes dentro de un fragmento de código y determina la cuota superior del número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. (1)

La complejidad ciclomática se calcula de una de las siguientes formas:

a. V (G)= Número de Regiones.
b. V (G)= Número de Aristas – Número de Nodos + 2.
c. V (G)= Número de Nodos Predicados +1.

2. **Prueba de la estructura de control:** Dentro de este tipo de prueba se contempla el método del camino básico mencionado anteriormente pero además existen otras pruebas asociadas que permiten ampliar la cobertura de la prueba y mejorar su calidad, las cuales son:

2.1. **Prueba de condición:** es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa. (10)

Algunos de los conceptos relacionados son condición simple, que es una variable lógica o una expresión relacional (E1 < operador - relacional > E2) y condición compuesta, la misma está formada por dos o más condiciones simples, operadores lógicos y paréntesis.

Los tipos de errores más comunes que se buscan en una prueba de condición son errores en operador lógico, errores en variable lógica, errores en paréntesis lógico, errores en operador relacional y errores en expresión aritmética.

2.2. **Prueba del flujo de datos:** selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. (10)

2.3. **Prueba de bucles:** es una técnica que se centra exclusivamente en la validez de las construcciones de bucles (bucles simples, anidados, concatenados y no estructurados). (11)

2.3.1. Bucles simples.

Se les aplica el siguiente conjunto de pruebas:

- Pasar por alto totalmente el bucle.
- Pasar una sola vez por el bucle.
- Pasar dos veces por el bucle.
- Hacer m pasos por el bucle con m < n (donde n es el número máximo de pasos permitidos por el bucle).
- Hacer n - 1, n y n + 1 pasos por el bucle.

2.3.2. Bucles anidados.

Si se empleara el mismo enfoque de prueba de bucles simples a los bucles anidados, el número de pruebas aumentaría considerablemente por lo cual Beizer sugiere emplear el siguiente enfoque:

- Comenzar por el bucle más interior. Establecer o configurar los demás bucles con sus valores mínimos.
- Llevar a cabo las pruebas de bucles simples para el bucle más interior, mientras se mantienen los parámetros de iteración de los bucles externos en sus valores mínimos. Añadir otras pruebas para valores fuera de rango o excluidos.
- Progresar hacia afuera, llevando a cabo pruebas para el siguiente bucle, pero manteniendo todos los bucles externos en sus valores mínimos y los demás bucles anidados en sus valores típicos.

- Continuar hasta que se hayan probado todos los bucles.

2.3.3. Bucles concatenados.

Estos bucles se pueden probar utilizando el enfoque de bucles simples, siempre y cuando cada uno de los bucles sea independiente del resto de lo contrario se debe emplear el enfoque de bucles anidados.

2.3.4. Bucles no estructurados. Siempre que sea posible estos deben rediseñarse.

II. Enfoque funcional o de caja negra

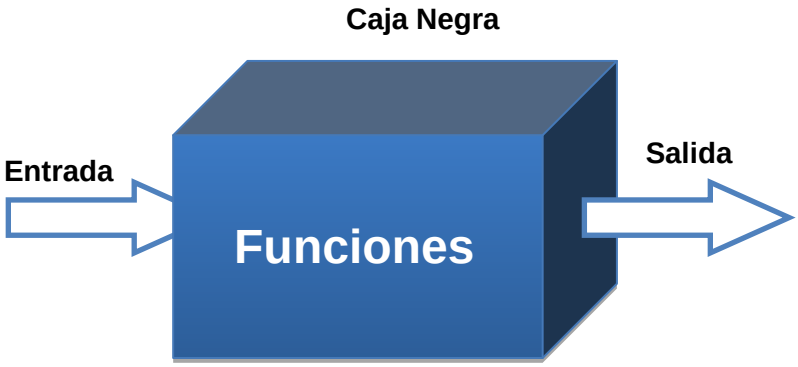


Figura 5: Enfoque de diseño de pruebas de caja negra.

Según Pressman, las pruebas de caja negra se centran en los requisitos funcionales del software. La prueba de caja negra permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Se trata de un enfoque que intenta descubrir diferentes tipos de errores que no se encuentran con los métodos de caja blanca. (12)

Estas pruebas pretenden encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

El objetivo de las pruebas de caja negra es estar totalmente despreocupado del comportamiento interno y de la estructura del programa. En lugar de esto, concentrarse en encontrar las circunstancias en las cuales el programa no se comporte según su especificación. En este acercamiento, los datos de prueba se derivan solamente de la especificación.(13)

Métodos de pruebas de caja negra

1. **Métodos de prueba basados en grafos:** en este método se debe entender los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas. (12)
   En este método se crea un grafo de objetos importantes con sus relaciones, posteriormente se diseñan una serie de pruebas que cubran el grafo de manera que se ejecuten todos los objetos y sus relaciones para descubrir errores.

2. **Partición equivalente:** es una técnica de simplificación de pruebas. Hay que dividir el dominio de entrada de un programa en clases de datos que tengan algo en común, de ahí derivan clases de prueba y por lo tanto también derivan clases de errores, de esta forma no hay necesidad de ejecutar muchas pruebas para encontrar errores genéricos. (12)

3. **Análisis de valores límite:** los casos de prueba que exploran las condiciones límite de un programa producen un mejor resultado para la detección de defectos, es decir, es más probable que los defectos del software se acumulen en estas condiciones. Se pueden definir las condiciones límites como las situaciones que se hallan directamente arriba, abajo y en los márgenes de las clases de equivalencia. (12)

   Cuando se habla de Valores Límites lo que se dice es que se eligen casos de prueba en los límites o bordes de la clase que se está probando. (Por ejemplo: Si una condición de entrada o salida exige un rango entre B y R, se realizan los casos de prueba para los valores B y R, y además a los valores que están por encima de ellos, es decir A y S).

4. **Prueba de la tabla ortogonal:** hay aplicaciones donde el número de parámetros de entrada es pequeño y los valores de cada uno de los parámetros está claramente delimitado. Cuando estos

números son muy pequeños (por ejemplo, 3 parámetros de entrada tomando 3 valores diferentes), es posible considerar cada permutación de entrada y comprobar exhaustivamente el proceso del dominio de entrada. En cualquier caso, cuando el número de valores de entrada crece y el número de valores diferentes para cada elemento de dato se incrementa, la prueba exhaustiva se hace impracticable. (12)

5. **Adivinando el error:** la idea básica de esta técnica consiste en enumerar una lista de equivocaciones y de las situaciones propensas a ciertos errores. Después se generan casos de prueba en base a dicha lista (que generalmente corresponden con defectos comunes y no con aspectos funcionales). Esta técnica también se ha denominado generación de casos especiales, ya que no se obtienen en base a otros métodos sino mediante la intuición o la experiencia. Es difícil dar un procedimiento para esta técnica puesto que es en gran parte un proceso intuitivo. (13)

1.8. Plan de Pruebas

La realización de un plan de pruebas es imprescindible, ya que en este se describe detalladamente la estrategia a seguir durante el proceso de pruebas, la planificación de las pruebas y los recursos asociados. Se debe especificar las pruebas a realizar, así como sus objetivos. Del mismo modo, cuando se analizan las pruebas para evaluar la calidad, no hay una sola perspectiva de qué es calidad o cómo se mide. En RUP, se categoriza la calidad mediante el modelo **FURPS+** (Funcionalidad, Utilización, Fiabilidad, Rendimiento, Capacidad de soporte y otros) cada prueba debe especificar a cuales de estos elementos está enfocada. En el plan de pruebas un elemento fundamental es la descripción de los criterios de evaluación para las pruebas, como clasificación de las no conformidades, pedidos de cambios y listas de chequeo.

1.9. Herramientas para pruebas de carga en aplicaciones web

Una prueba de carga se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperadas. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. Esta prueba puede mostrar los tiempos de respuesta de todas las transacciones importantes de la aplicación. Se analizarán algunas de las herramientas usadas para automatizar este tipo de pruebas. Con el uso de estas herramientas se ahorran recursos, se obtienen mejores resultados y una mayor eficiencia en las pruebas.

JMeter

JMeter es una herramienta de Apache que facilita la ejecución de pruebas de estrés en aplicaciones web. Con esta herramienta es posible atacar aquellos puntos que se quieran probar respecto a concurrencia de usuarios y velocidad de carga en un sitio. (15)

Para ello generalmente se elabora un plan de trabajo y se definen los parámetros de las peticiones que se adicionan al plan de prueba. Para ello deben configurarse las peticiones que se harán y también deberán agregarse los *listeners* (formas de recuperar la información) para obtener los resultados de las pruebas. (15)

LoadRunner

LoadRunner es una herramienta comercial de carga y performance de Hewlett-Packard. LoadRunner permite emular cientos o miles de usuarios concurrentes para simular carga sobre aplicaciones, y a la vez recolectar información clave de la infraestructura de componentes (servidores web, bases de datos, etc.).

The Grinder

The Grinder es un framework que facilita la ejecución de pruebas de carga distribuyendo los inyectores de carga en distintos equipos. Los scripts de test se escriben en Python, y los scripts HTTP pueden ser grabados directamente desde una sesión en un navegador.

Luego de analizar algunas de las características de estas herramientas en busca de cuál podría ser más factible en la aplicación de las pruebas de carga en la plataforma VideoWeb se decide usar por su amplia documentación, estabilidad, facilidad, eficiencia y ser de uso gratuito la aplicación JMeter.

1.10. Modelos y Estándares

Un modelo de calidad es un conjunto de prácticas vinculadas a los procesos de gestión y el desarrollo de proyectos. Este modelo supone una planificación para alcanzar un impacto estratégico, cumpliendo con los objetivos fijados en lo referente a la calidad del producto. Al implementar un modelo de calidad, una empresa busca desarrollar sistemáticamente productos y servicios que cumplan con los requisitos y las exigencias de los clientes. Es importante que los elementos que forman el conjunto del modelo de calidad se encuentren estructurados en forma tal que sea posible realizar un control y seguimiento de los procesos. El modelo debe reunir las actividades y funciones relacionadas con la calidad para que puedan ejecutarse de un modo sistemático y formal.

Para todas las industrias hay marcos de trabajo compuestos por normativas y estándares más o menos estables y consensuados, que en forma de modelos ayudan a mejorar o evaluar la calidad de sus sistemas de producción.

La ISO 9000 proporciona un conjunto de estándares para la gestión de la calidad en cualquier actividad relacionada con el proceso de producción. Cada vez más las empresas están a favor de crear sistemas de calidad para supervisar todas las fases de sus procesos de producción. La ISO 9000 se ha especializado en todo lo referente a la solución del software en la ISO 9000-3, puesto que esta disciplina tiene características propias diferentes como para distinguirse del proceso de producción en general. Proporciona una guía útil que sirve para detectar y corregir una serie de problemas de los productos software, consiguiendo tras su aplicación una mejora en la calidad de los mismos. (16)

CMMI[6] es un modelo de referencia para el crecimiento de capacidades y madurez, que se enfoca tanto en procesos de Administración como de Ingeniería de Sistemas y Software. Con su instauración se espera alcanzar beneficios como:

- Calendarios y presupuestos predecibles en los proyectos.
- Mejora del ciclo de vida dentro del desarrollo de software.
- Mayor productividad.
- Mayor calidad de los productos y servicios que ofrece la universidad a sus clientes y por ende la satisfacción de los mismos.

El modelo CMMI constituye un marco de referencia de la capacidad de las organizaciones de desarrollo de software en el desempeño de sus diferentes procesos, proporcionando una base para la evaluación de la madurez de las mismas y una guía para implementar una estrategia para la mejora continua de los mismos. Actualmente se usa en la mayoría de las empresas y organizaciones productoras de software en el mundo.

CMMI tienen dos utilidades. Pueden servir tanto como guía para la mejora en una organización, o como criterio para evaluar su nivel; además introduce una segunda dimensión, también válida para guiar las actividades de mejora y para evaluar a las organizaciones la capacidad de los procesos.

[6] Capability Maturity Model Integration.

Los modelos CMMI presentan 2 versiones:

Versión escalonada: Guía a la organización sobre las áreas de procesos que debe ir abordando, las prácticas que debe implantar y los objetivos que debe alcanzar para conseguir los sucesivos niveles de madurez.

Versión continua: Permite cierta libertad en la organización sobre las áreas de proceso que desea mejorar, le orienta para ir elevando su nivel de capacidad.

1.11. Conclusiones

El proceso de prueba es uno de los más difíciles, y por lo tanto requiere del mismo esfuerzo que cualquier proceso de implementación o diseño. Es necesario el uso de métricas en los proyectos productivos que evalúen los factores de la calidad y faciliten dicho proceso. El plan de pruebas es el artefacto rector para el desarrollo de este proceso en el cual se especifican detalladamente los escenarios de pruebas y la planeación general de las mismas. Las pruebas son aplicadas a diferentes niveles del software empezando por el nivel de desarrollador hasta finalmente llegar a la aceptación de los usuarios.

Para llevar a cabo este proceso eficientemente se necesita de una estandarización que guíe en todo momento el proceso, y donde los implicados puedan orientarse sobre qué pruebas aplicarle a su producto según el tipo de aplicación y condiciones de los mismos. En los proyectos de la universidad no se cuenta con documentación oficial sobre cómo desarrollar este proceso, donde cada miembro del equipo de desarrollo tenga conocimientos sobre el tema y sepa sus responsabilidades e importancia del trabajo que desarrolla.

El principal objetivo de las pruebas consiste en encontrar errores, pero si ya no se encuentran (no quiere esto decir que no los haya) debe existir algún criterio de terminación de la pruebas. Algunos de los criterios pueden ser cuando el tiempo de la prueba ha expirado, o cuando todos los casos de pruebas se hayan ejecutado sin errores.

De forma generalizada existen deficiencias en los procesos de pruebas que se deben aplicar durante el desarrollo de aplicaciones web, algunas de estas son el incoherente diseño de las pruebas y la no utilización de herramientas en la realización de estas.

Capítulo 2

Diseño y aplicación de pruebas de software

2.1. Introducción

En el desarrollo de este capítulo se analizan todas las funcionalidades del sistema con la intención de seleccionar los requisitos más importantes para el funcionamiento de la aplicación, que serán probados. También quedan diseñados los casos de prueba correspondientes, para ejecutar eficientemente las pruebas planificadas. Cada proceso de prueba se inicia con un plan de pruebas por lo que es necesario precisar un plan de pruebas para la plataforma VideoWeb en el que se describa la estrategia a seguir para la aplicación de estas pruebas.

2.2. Plan de prueba general

Un plan de prueba establece los objetivos y la estrategia para la evaluación del proyecto que será sometido a pruebas. En este se describe al equipo de pruebas, se identifican los participantes en el proceso, sus responsabilidades y el escenario en el que se ejecutarán las pruebas. En el plan de pruebas definido para la plataforma VideoWeb, se identifican los elementos que serán probados y los recursos necesarios para hacer las pruebas.

El objetivo fundamental que tiene este plan de pruebas para la plataforma VideoWeb es el establecimiento del flujo de trabajo que se va a realizar para la puesta en marcha de las pruebas. Todo el proceso de prueba debe ser documentado. Para alcanzar los objetivos trazados es necesario que se lleve a cabo una estrategia para dar cumplimiento a los mismos. En esta estrategia de prueba deben quedar definidos los recursos necesarios para que las mismas sean llevadas a cabo con toda la eficacia posible. Para llevar a cabo las pruebas planificadas es necesario disponer de algunos recursos indispensables, en el siguiente epígrafe se exponen de forma clara.

2.3. Recursos para las pruebas

Especificaciones de Software:

- Sistema Operativo: GNU/Linux Ubuntu.
- Servidor Web: Apache2.

- Servidor de Base de Datos: PostgreSQL.
- Servidor Streaming: Flumotion Streaming Server.

Especificaciones de Hardware

Tabla 5: Servidores

Recurso	Tipo
Servidor de archivos multimedia	160 GB
Servidor de aplicaciones	Disco duro 160 GB, 1 GB RAM, procesador Core 2 Duo a 2.2 GHz y MB Intel DG 965.

Tabla 6: PC Clientes

PC Clientes	
Cantidad	1
Descripción	Disco duro SATA 80 Gb, 3 Gb RAM, procesador Intel Dual-Core Centrino a 3.2 GHz y tarjeta de red Intel Ethernet.
Software base	GNU/Linux Ubuntu.

2.4. Pruebas a aplicar

A la plataforma VideoWeb se le aplican pruebas de caja negra usando la técnica de partición equivalente, también se aplican pruebas de caja blanca para validar su funcionamiento interno usando la técnica del camino básico. Para comprobar la solidez de la plataforma en los momentos de carga extrema se aplican pruebas de carga y pruebas de estrés para doblar dicha carga hasta romper la aplicación. También se aplican pruebas de estructura para evaluar el estado de los enlaces y pruebas de usabilidad. Durante el desarrollo del presente capítulo se aborda en detalle cada una de estas pruebas con sus correspondientes diseños.

La plataforma VideoWeb cuenta con un total de 27 casos de usos (CU), distribuidos en dos módulos, uno de administración compuesto por 7 CU y otro de gestión y presentación de contenidos, con 20 CU.

Para realizar las pruebas de caja negra y caja blanca se han seleccionado las funcionalidades críticas para la plataforma, en la siguiente tabla se muestra dicha selección.

Tabla 7: Casos de usos seleccionados

Casos de uso seleccionados.				
Subsistemas	**CU**	**Prioridad**	**CN**	**CB**
Administración	Gestionar usuario	Secundario	X	
	Autenticar	Secundario	X	
	Gestionar Sección	Secundario	X	
	Gestionar Categoría	Secundario	X	
	Gestionar tipología de archivo multimedia	Crítico	X	
	Buscar usuario registrado	Secundario	X	
	Crear cuenta de usuario	Crítico	X	X
Gestión y presentación de contenidos	Adicionar archivo multimedia	Crítico	X	X
	Modificar datos asociados a los archivos multimedia	Crítico	X	
	Eliminar archivo multimedia	Crítico	X	X
	Crear publicación de archivo multimedia.	Crítico	X	X
	Modificar publicación de archivo multimedia	Crítico	X	X
	Eliminar publicación de archivo multimedia	Crítico	X	X
	Publicar archivo multimedia	Crítico	X	X

	Dejar de publicar archivo multimedia	Crítico	X	
	Reproducir archivos multimedia	Crítico	X	
	Adicionar artículo de contenido	Secundario	X	
	Cambiar Interfaz	Opcional	X	
	Modificar artículo de contenido	Secundario	X	
	Eliminar artículo de contenido	Secundario	X	
	Publicar artículo de contenido	Secundario	X	
	Dejar de publicar artículo de contenido	Secundario	X	
	Visualizar artículo de contenido	Secundario	X	
	Gestionar lista de reproducción	Opcional	X	
	Buscar contenido	Secundario	X	
	Buscar archivos multimedia	Crítico	X	X
	Gestionar señal en vivo	Crítico	X	

2.5. Diseño de casos de prueba de caja negra

Las pruebas de caja negra permiten identificar las posibles fallas en el funcionamiento del sistema. Estas pruebas se centran principalmente en las características del producto independiente del código. Con la ejecución de estas pruebas es posible encontrar errores de interfaz, errores de inicialización y terminación así como funciones incorrectas o ausentes.

Para llevar a cabo estas pruebas es necesario tener conocimiento sobre los escenarios de prueba y secciones que serán probadas, con estos escenarios es muy fácil comprender el funcionamiento del requisito en cuestión. En los casos de pruebas se combinan los posibles juegos de datos válidos e inválidos que son necesarios para verificar el correcto funcionamiento del caso de uso. Estas pruebas se realizan a nivel de sistema, el tipo de prueba es funcional, empleando el enfoque estructural o de caja negra, específicamente usando la técnica de particiones equivalentes. Esta técnica es muy efectiva al realizar pruebas sobre la interfaz de la plataforma, estas prueban la validez de cada entrada de datos o información al sistema. Pretenden demostrar que las funciones de la plataforma son operativas.

2.5.1.Subsistema: Gestión y presentación de contenidos

Nombre del caso de uso: Adicionar archivo multimedia.

Descripción general: El caso de uso se inicia cuando un autor desea adicionar archivos multimedia en el servidor de medias y crear las referencias pertinentes a estos y toda la información asociada en la base de datos.

Condiciones de ejecución: Que el autor se haya autenticado. Que el archivo a adicionar esté en el formato de archivo requerido. Que exista un punto de almacenamiento con espacio libre disponible para contener el nuevo archivo.

Secciones a probar en el Caso de Uso:

Tabla 8: Escenarios de prueba: CU_ Adicionar archivo multimedia

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1 Adicionar archivo multimedia	EC 1.1: Adicionar archivo multimedia con éxito	El sistema muestra un formulario con un botón para adicionar un nuevo archivo multimedia, un filtro para encontrar un archivo previamente adicionado y un listado de los archivos multimedia existentes en los puntos de almacenamiento. Al presionar el botón adicionar el sistema muestra un formulario con los metadatos definidos para

		la tipología seleccionada. El autor introduce los datos y el sistema verifica el formato del archivo y los datos asociados a este. Si los datos introducidos son correctos el sistema muestra un mensaje informando al usuario que la operación se ha realizado correctamente.
	EC 1.2: Adicionar archivo multimedia con falla	Si los datos introducidos son incorrectos o se dejó en blanco algún campo obligatorio el sistema muestra un mensaje informando que los datos entrados son incorrectos.
	EC 1.3: Adicionar archivo multimedia. Cancelar	El autor presiona el botón “Cancelar”, el sistema cancela la operación y vuelve a la interfaz anterior.

Nombre del caso de uso: Eliminar archivo multimedia.

Descripción general: Este caso de uso se inicia cuando el revisor desde la opción Gestionar archivo multimedia, en el Menú de trabajo, desea eliminar un archivo multimedia. El caso de uso termina cuando el sistema muestra un mensaje de notificación al revisor de operación realizada correctamente o por la cancelación de la operación por parte del revisor.

Condiciones de ejecución: Que el revisor se haya autenticado, que el archivo no se encuentre en uso.

Secciones a probar en el Caso de Uso:

Tabla 9: Escenarios de prueba: CU_ Eliminar archivo multimedia

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1 Eliminar archivo	EC 1.1: Eliminar archivo	El sistema muestra un formulario con un

multimedia	multimedia	botón para adicionar un nuevo archivo multimedia, una serie de opciones de filtrado así como un listado de los archivos multimedia existentes en los puntos de almacenamiento con las opciones de editar y eliminar (en caso que no esté en uso) para cada uno. El revisor presiona el vínculo “Eliminar”, el sistema muestra un formulario con un mensaje de confirmación y las opciones “Eliminar” y “Cancelar”, el revisor selecciona la opción “Eliminar” y el sistema elimina el archivo del servidor de medias.
	EC 1.2 Cancelar la operación	El revisor selecciona la opción “Cancelar” y el sistema muestra el mensaje “Operación cancelada por el usuario” y vuelve al flujo normal de los eventos.

Nombre del caso de uso: Crear publicación de archivo multimedia.

Descripción general: Este caso de uso tiene lugar cuando el autor crea una publicación de archivo multimedia a partir de un archivo multimedia previamente almacenado en el servidor de medias o de un archivo local con el formato requerido y le define la sección y la categoría a la que pertenece. El caso de uso termina cuando el sistema valida los datos de la publicación de archivo multimedia, muestra un mensaje de notificación al autor de operación realizada correctamente, o termina cuando el autor cancela la creación de la publicación del archivo multimedia.

Condiciones de ejecución: Que el autor se haya autenticado, que existan archivos multimedia previamente adicionados, que existan puntos de publicación de streaming disponibles.

Secciones a probar en el Caso de Uso:

Tabla 10: Escenarios de prueba: CU_ Crear publicación de archivo multimedia

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1 Crear publicación de archivo multimedia	EC 1.1: Crear publicación a partir de un archivo multimedia almacenado en el servidor. Satisfactoriamente.	El sistema muestra un formulario que permite nombrar la publicación, escoger las secciones y categorías a las cuales va a pertenecer, escoger el punto de publicación donde se va a guardar el archivo multimedia en el servidor de streaming, seleccionar o buscar el archivo a publicar así como introducirle una descripción a la publicación. El sistema valida los datos entrados, y muestra un mensaje indicando que la publicación se ha creado correctamente.
	EC 1.2: Crear publicación a partir de un archivo multimedia almacenado en el servidor. Falla.	Si se dejó algún campo en blanco el sistema muestra un mensaje informando que los datos entrados son incorrectos.

Nombre del caso de uso: Eliminar publicación de archivo multimedia.

Descripción general: Este caso de uso se inicia cuando el revisor desde la opción Gestionar publicación de archivo multimedia, en el menú de trabajo, desea eliminar una publicación de archivo multimedia. El caso de uso termina cuando se elimina una publicación de archivo multimedia y el sistema muestra un mensaje de notificación al revisor de operación realizada correctamente o cuando la operación es cancelada por el revisor.

Condiciones de ejecución: Que el revisor se haya autenticado que existan publicaciones de archivo multimedia.

Secciones a probar en el Caso de Uso:

Tabla 11: Escenarios de prueba: CU_ Eliminar publicación de archivo multimedia

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1 Eliminar publicación de archivo multimedia	EC 1.1: Aceptar	El revisor presiona el vínculo “Eliminar” de una de las publicaciones de archivo multimedia del listado. El sistema muestra un formulario con un mensaje de confirmación y las opciones “Eliminar” y “Cancelar”. El usuario selecciona “Eliminar”.
	EC 1.2: Cancelar	El usuario selecciona la opción “Cancelar”.

A continuación se muestra una tabla con una descripción de las variables usadas en la matriz de datos para evaluar las funcionalidades. Es necesario hacer combinaciones de estos datos para forzar al sistema a mostrar las posibles fallas.

Tabla 12: Descripción de variables

Descripción de variables.			
Nombre de campo	**Datos Válidos**	**Datos Inválidos**	**Clasificación**
Tipo de servidor	Es necesario seleccionar un tipo de servidor	No seleccionar ningún tipo de servidor	Lista de selección
Punto de almacenamiento	Selecciona el punto de almacenamiento en caso de tener más de uno	No seleccionar ningún punto de almacenamiento	Lista de selección.
Explorar	Explorar en busca del	No cagar ningún archivo	

	archivo multimedia	multimedia	
Publicado	Marcar el *CheckBox*, para publicar la media	No se marca	*CheckBox*
Usuario	Cualquier combinación de Letras y números (Ivania, José, Ale, Ivan88)	Dejar el campo vacío	Campo de texto.
Dirección de correo electrónico	Las direcciones de correo deben de contener en su estructura @ y puntos. icortina@uci.cu	Dejar el campo vacío. @@ icortina@ @uci.cu	
Contraseña	Cantidad de caracteres > =8 (Para que sea segura debe contener letras mayúsculas, minúsculas y signos de puntuación	Dejar el campo vacío Que no coincidan la contraseña y la confirmación	Campo de texto.

2.6. Diseño de casos de prueba de caja blanca

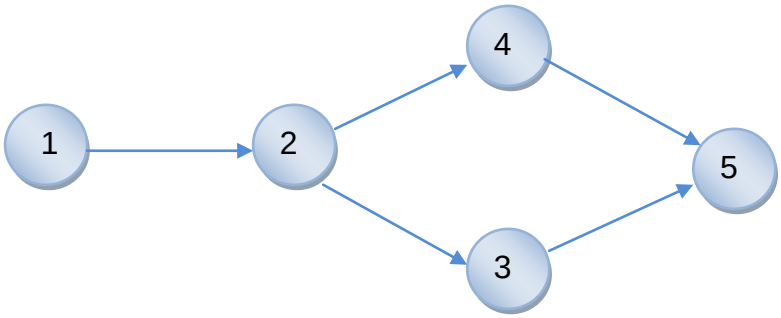
Las pruebas de caja blanca permiten al probador comprobar que se verifiquen las funciones internas de un módulo. La técnica que se emplea para la construcción de los casos de prueba es la técnica del camino básico. Para realizar los diseños de pruebas se eligió un camino principal que representa una función válida, es decir que no sea un tratamiento de error y además que atraviese el máximo número de decisiones en el grafo. A continuación se presentan los diferentes diseños realizados para cada módulo de la plataforma VideoWeb.

Nombre del caso de uso: Eliminar archivo multimedia.

Paso1: Dibujar el grafo de flujo asociado a partir del código fuente.

```
function Eliminar_Datos_AM($id)
{
   1 $uso = archivo_multimedia_controlador::En_Uso_AM($id);
   2 if ($uso)
        {
           3 drupal_set_message("No se pudo eliminar. Está en uso", 'error');
           3 return false;
        }
     else
        {
           4 $am = archivo_multimedia_modelo::Seleccionar_AM($id);
           4 tipologia_am_controlador::Eliminar_Datos_Tipologia_AM($am->id_tipologia_am, $am->id_dato_tipologia_am);
           4 almacen_eliminar_fichero($am->ftp_punto_publicacion_id, $am->nombre_fichero);
           4 archivo_multimedia_modelo::Eliminar_AM($id);
           4 drupal_set_message("Se pudo eliminar la información asociada al archivo multimedia. ");
           4 return true;
        }
5 }
```

Figura 6: Código del CU Eliminar archivo multimedia.



Paso 2: Cálculo de complejidad ciclomática.

a. V (G)= Número de Aristas – Número de Nodos + 2. V (G) = 5 – 5 + 2 = 2.

Paso 3: Caminos básicos.

Camino 1: 1-2-3-5.

Camino 2: 1-2-4-5.

Paso 4: Caso de prueba para el camino 2: **1-2-4-5**

Entrada: id = colgando en tus manos.

Resultado Esperado: El sistema elimina el archivo del servidor de medias, elimina todas las referencias a este en la base de datos y muestra un mensaje indicando que el archivo fue eliminado.

Condiciones: Que un usuario este autenticado como revisor y que el archivo no se encuentre en uso.

Nombre del caso de uso: Adicionar archivo multimedia.

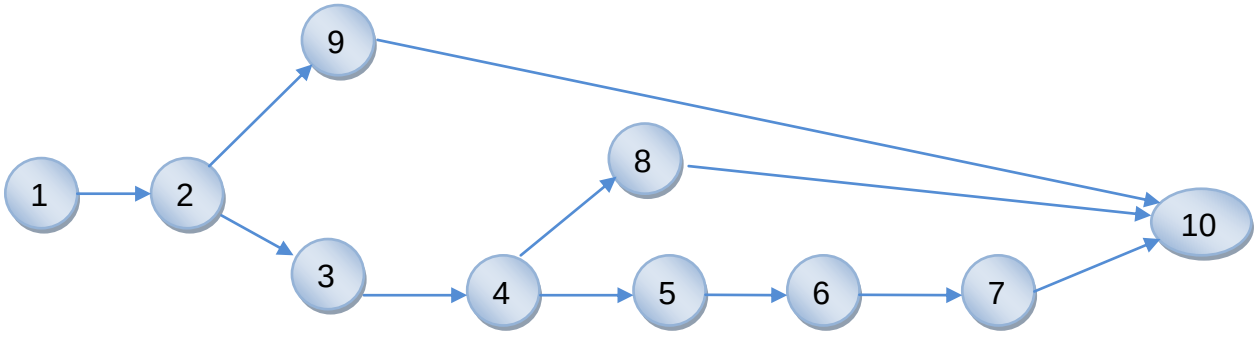
Paso1: Dibujar el grafo de flujo asociado a partir del código fuente.

```
function Insertar_AM($datos, $fichero, $dir)
    {
       1 $ftp_punto_publicacion_id = $datos["select_almacen"];
       1 $ftp_punto_publicacion = Puntos_Publicacion_FTP_Modelo::Listar_Puntos_Publicacion(array("id" => $ftp_punto_publicacion_id));
       1 $ftp_punto_publicacion = $ftp_punto_publicacion[0];
       1 $ruta = $ftp_punto_publicacion['ruta'];
       1 $ftphtml = ftphtml_objeto($ftp_punto_publicacion['nid']);
       1 $ruta = $ftphtml->Cambiar_Carpeta($ruta);
       1 $nombre_real_fichero = $ftphtml->Subir_Fichero($fichero, true, $dir);

       2 if ($nombre_real_fichero)
            {
               3 $id_tipologia = $datos['tipologia_id'];
               3 $id_datos_tipologia = tipologia_am_controlador::Insertar_datos_tipologia_creada($id_tipologia, $datos);

               4 if ($id_datos_tipologia)
                    {
                       5 $r = archivo_multimedia_modelo::Insertar_AM($nombre_real_fichero, $datos['ftp_punto_publicacion_id'], $id_tipologia,
$id_datos_tipologia);
                       6 if ($r)
                            {
                              7 return $nombre_real_fichero;
                            }
                    }
               8 return false;
            }
       9 return false;
   10 }
```

Figura 7: Código del CU Adicionar archivo multimedia.



Paso 2: Cálculo de complejidad ciclomática.

a. V (G)= Número de Aristas – Número de Nodos + 2. V (G) = 11 – 10 + 2 = 3.

Paso 3: Caminos básicos.

Camino 1: 1-2-3-4-5-6-7-10.

Camino 2: 1-2-3-4-8-10.

Camino 3: 1-2-9-10.

Paso 4: Caso de prueba para el camino 1: **1-2-3-4-5-6-7-10.**

Entrada: Tipo de servidor = seleccionado, punto de almacenamiento = seleccionado, archivo multimedia = cargado, datos asociados a la tipología = (se escoge la tipología a la que pertenece el archivo y se llenan todos los campos del formulario según la tipología).

Resultado Esperado: El sistema verifica el formato del archivo y los datos asociados a este, sube el archivo multimedia para el servidor de medias y crea la referencia al mismo en la base de datos. Muestra un mensaje informando al usuario que la operación se ha realizado correctamente

Condiciones: Que el autor se haya autenticado. Que el archivo a adicionar esté en el formato de archivo requerido. Que exista un punto de almacenamiento con espacio libre disponible para contener el nuevo archivo.

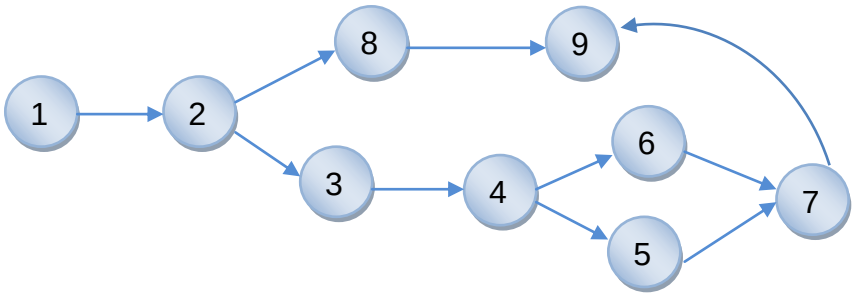
Nombre del caso de uso: Publicar archivo multimedia.

Paso1: Dibujar el grafo de flujo asociado a partir del código fuente.

```
function Publicar_AM($id_ftp_punto_publicacion_streaming, $id_archivo_multimedia, $publicar = true, $cron = false)
       {
          1 $am = archivo_multimedia_modelo::Seleccionar_AM($id_archivo_multimedia);
          1 $id_almacen = $am->ftp_punto_publicacion_id;
          1 $id_streaming = $id_ftp_punto_publicacion_streaming;
          1 $nombre_fichero = $am->nombre_fichero;
          2     if ($publicar)
                   {
                      3 $r = almacen_publicar_fichero_streaming($id_almacen, $id_streaming, $nombre_fichero);
                      3 $almacen_s = almacen_controlador::Obtener_Datos_Id($id_almacen);
                      3 $almacen_c = almacen_controlador::Obtener_Datos_Id_Streaming($id_streaming);
                      3 $sender = $cron ? t('Maintenance task performed by cron') : '';
                    4 if ($r)
                        {
                           5 $mensaje = t('File %nombre copied from server %id_s to %id_c. !sender0',
                           5 array('%nombre' => $nombre_fichero, '%id_s' => $almacen_s->nombre, '%id_c' => $almacen_c->nombre, '!sender' => $sender));
                        }
                     else
                        {
                           6 $mensaje = t('Failed to copy file !nombre from server %id_s to %id_c.!sender', array('!nombre' => $nombre_fichero, '%id_s'
=> $almacen_s->nombre, '%id_c' => $almacen_c->nombre, '!sender' => $sender));
                           6 form_set_error('ftp', $mensaje);
                        }
                    7 watchdog('videoweb', $mensaje, array());
                    7 return $r;
                     }
                  else
                     {
                      8 $almacen_c = almacen_controlador::Obtener_Datos_Id_Streaming($id_streaming);
                      8 $r = almacen_eliminar_fichero($almacen_c->id, $nombre_fichero);
                      8 return $r;
                     }

       } 9
```

Figura 8: Código del CU Publicar archivo multimedia.



Paso 2: Cálculo de complejidad ciclomática.

a. V (G)= Número de Aristas – Número de Nodos + 2. V (G) = 10 – 9 + 2 = 3.

Paso 3: Caminos básicos.

Camino 1: 1-2-3-4-5-7-9.

Camino 2: 1-2-3-4-6-7-9.

Camino 3: 1-2-8-9.

Paso 4: Caso de prueba para el camino 1: **1-2-3-4-5-7-9.**

Entrada: Publicado = activar el *CheckBox* de una publicación de archivo multimedia no publicada.

Resultado Esperado: El sistema cambia el estado de la publicación de archivo multimedia ha publicado, se habilita el archivo multimedia en el servidor de streaming y muestra un mensaje indicando que el archivo fue publicado satisfactoriamente.

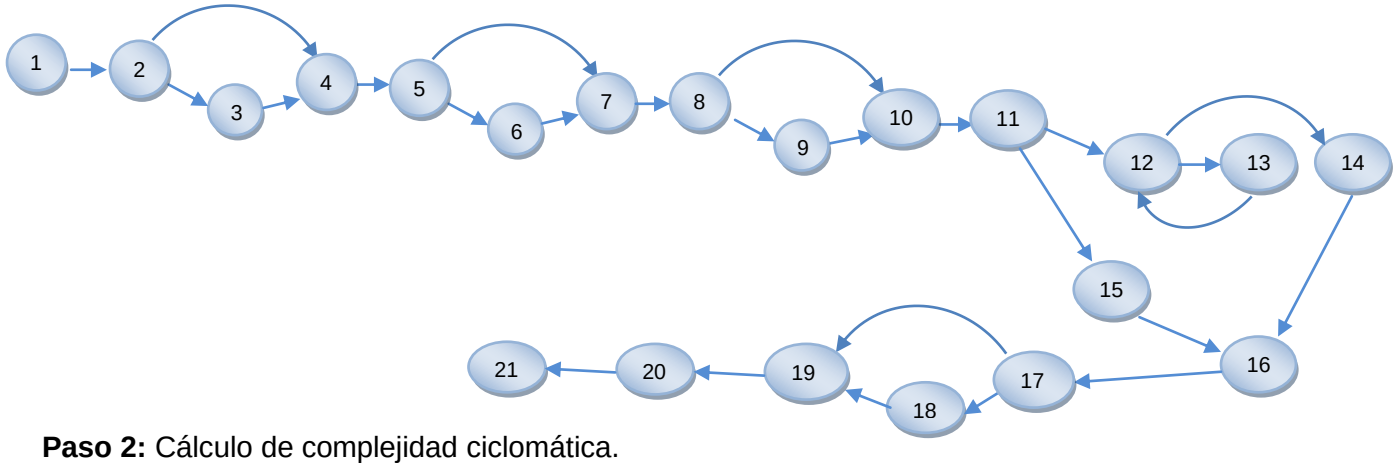
Condiciones: Que el revisor se haya autenticado, que existan publicaciones de archivos multimedia cuyo estado sea no publicada y que exista espacio disponible en el servidor de streaming.

Nombre del caso de uso: Crear Cuenta de usuario.

Paso1: Dibujar el grafo de flujo asociado a partir del código fuente.

```
function user_register()
{
    1 global $user;
    1 $admin = user_access('administer users');
    2 if (!$admin && $user->uid)
    {
        3 drupal_goto('user/' . $user->uid);
    }
    4 $form = array();
    5 if (!$admin)
    {
        6 $form['user_registration_help'] = array(
            '#value' => filter_xss_admin(variable_get('user_registration_help', '')),
            '#weight' => -20,
        );
    }
    7 $form = array_merge($form, user_edit_form($form_state, NULL, NULL, TRUE));
    8 if ($admin)
    {
        9 $form['account']['notify'] = array(
            '#type' => 'checkbox',
            '#title' => t('Notify user of new account')
        );
        9 $form['destination']=array('#type'=>'hidden','#value'=>isset($_REQUEST['destination'])?$_REQUEST['destination']:$_GET['q']);
    }
    10 $null = NULL;
    10 $extra = _user_forms($null, NULL, NULL, 'register');
    11 if (!$extra)
    {
        12 foreach (array('name', 'mail', 'pass', 'status', 'roles', 'notify') as $key)
        {
            13 if (isset($form['account'][$key]))
            {
                13 $form[$key] = $form['account'][$key];
            }
        }
        14 unset($form['account']);
    }
    15 else
    {
        15 $form = array_merge($form, $extra);
    16}
    17 if (variable_get('configurable_timezones', 1))
    {
    18 $form['timezone'] = array(
            '#type' => 'hidden',
            '#default_value' => variable_get('date_default_timezone', NULL),
            '#id' => 'edit-user-register-timezone',
        );
    }
    19 $form['submit'] = array('#type' => 'submit', '#value' => t('Create new account'), '#weight' => 30);
    20 $form['#validate'][] = 'user_register_validate';
    21 return $form;
}
```

Figura 9: Código del CU Crear cuenta de usuario.



Paso 2: Cálculo de complejidad ciclomática.

a. V (G)= Número de Aristas – Número de Nodos + 2. V (G) = 26 – 21 + 2 = 7

Paso 3: Caminos básicos.

Camino 1: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-16-17-18-19-20-21.

Camino 2: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-16-17-19-20-21.

Camino 3: 1-2-3-4-5-6-7-8-9-10-11-15-16-17-18-19-20-21.

Camino 4: 1-2-3-4-5-6-7-8-10-11-12-13-14-16-17-18-19-20-21.

Camino 5: 1-2-3-4-5-7-8-9-10-11-12-13-14-16-17-18-19-20-21.

Camino 6: 1-2-4-5-6-7-8-9-10-11-12-13-14-16-17-18-19-20-21.

Camino 7: 1-2-4-5-7-8-10-11-15-16-17-19-20-21.

Paso 4: Caso de prueba para el camino 1: **1-2-3-4-5-6-7-8-9-10-11-12-13-14-16-17-18-19-20-21.**

Entrada: Usuario = icortina, Correo electrónico = icortina@estudiantes.uci.cu, Contraseña = *******, confirmar contraseña = *******.

Resultado Esperado: El sistema almacena los datos del usuario.

Nombre del caso de uso: Crear publicación de archivo multimedia.

Paso1: Dibujar el grafo de flujo asociado a partir del código fuente.

```
function archivo_multimedia_insert($node)
{

    1 return archivo_multimedia_modelo::Insertar($node);
}
```

Figura 10: Código del CU Crear publicación de archivo multimedia.

1

Paso 2: Cálculo de complejidad ciclomática.

a. V (G)= Número de Aristas – Número de Nodos + 2. V (G) = 0 - 1 + 2 = 1

Paso 3: Caminos básicos.

Camino 1: 1

Paso 4: Caso de prueba para el camino 1: **1**

Entrada: Nombre = Harry Potter, Punto de publicación = seleccionado, Archivo multimedia = seleccionar el archivo a publicar, Secciones = escoger las secciones y categorías a las cuales va a pertenecer.

Resultado Esperado: El sistema almacena los datos de la publicación en la base de datos y muestra un mensaje indicando que la publicación se ha creado correctamente.

Condiciones: Que el autor se haya autenticado y que existan archivos multimedia previamente adicionados.

2.7. Diseño de pruebas de Carga y Stress

Las pruebas de carga se realizan para observar el comportamiento de la plataforma VideoWeb bajo una cantidad de peticiones esperada. Esta carga será el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. Con las pruebas de estrés se irá doblando el número de usuarios concurrentes hasta romper la aplicación. Con esta prueba es posible comprobar la solidez de la plataforma en los momentos de carga extrema, además permite determinar si la aplicación rendirá lo suficiente en caso

de que la carga real supere a la carga esperada. Esta prueba se realizará usando la herramienta JMeter.

2.7.1. Plan de Prueba en JMeter

Un plan de pruebas desarrollado en JMeter se compone de elementos. Cada uno de estos elementos realiza una función específica. A continuación se describen los elementos que se usan en el Plan de Pruebas diseñado para realizar esta prueba a la plataforma VideoWeb.

1. Grupo de Hilos: Utilizando este elemento, se podrá definir: **Número de Hilos**, donde se definen la cantidad de usuarios que se van a simular en la prueba. **Período de subida**, indica cuantos segundos se tardará en alcanzar el número máximo de usuarios. El **contador del bucle** representa el número de veces que cada hilo ejecutará la prueba.
2. Gestor de *Cookies* HTTP: Es un elemento de configuración que permite que cada hilo (usuario) tenga acceso a las cookies enviadas por el servidor. Es la manera de que el Plan de Pruebas pueda mantener el estado de la aplicación web.
3. Peticiones generadas por el navegador mediante el Proxy HTTP de JMeter. Dentro de cada controlador se encuentran las peticiones HTTP a enviar al servidor

Para interpretar los resultados es necesario adicionar varios *Listeners* al plan de prueba. Los *Listeners* son los elementos que permiten tener acceso a los datos recopilados por las pruebas; estos pueden ser desde datos estadísticos hasta gráficos de evolución. Los *Listeners* empleados fueron:

4. Informe Agregado: En este se muestran una serie de datos, los cuales exponen el estado en el que se encuentra el software con respecto a las funcionalidades probadas.
5. Ver Árbol de Resultado: Con este *Listeners* se puede observar el instante en el que se van cargando las peticiones http de manera factible o no. Para los casos en los cuales ha fallado la petición http se mostraran en rojo dicha peticiones y para el caso en que sean cargadas satisfactoriamente se mostraran en verde.
6. Gráfico de Resultados: Una breve descripción de los elementos del gráfico puede ser la siguiente:

   - Datos: muestra los valores actuales de los datos.

- ❖ Media: representa la Media.
- ❖ Mediana: dibuja la Mediana.
- ❖ Desviación: muestra la Desviación Estándar (una medida de la variación)
- ❖ Rendimiento: representa el número de muestras por unidad de tiempo.

7. Escritor de Datos Simples: Este *listeners* permite guardar en un informe los datos de tiempo de respuesta, número de errores y otros datos estadísticos de la ejecución de un Plan de Pruebas y luego cargar dicho informe con cualquiera de los restantes *listeners* para visualizar los resultados de forma diferente.

8. Servidor Proxy HTTP: Los controladores de Petición HTTP pueden añadirse uno a uno, pero es muy engorroso cuando se quiere capturar, por ejemplo, diez minutos de uso de la aplicación web que generen 200 peticiones HTTP. Para ello JMeter cuenta con un elemento que mediante un patrón Proxy puede capturar el tráfico entre un navegador y el servidor web. De la configuración del proxy es importante destacar dos cosas: el puerto y la agrupación de los datos de la muestra: Controlador Objetivo y Agrupación. Ambos valores se refieren a cómo se van a agrupar las muestras (las peticiones HTTP) en el árbol del plan de pruebas. Para que queden más legibles y para poder identificar los distintos clics del usuario se establece en el combo de Agrupación, Poner cada grupo en un nuevo controlador.

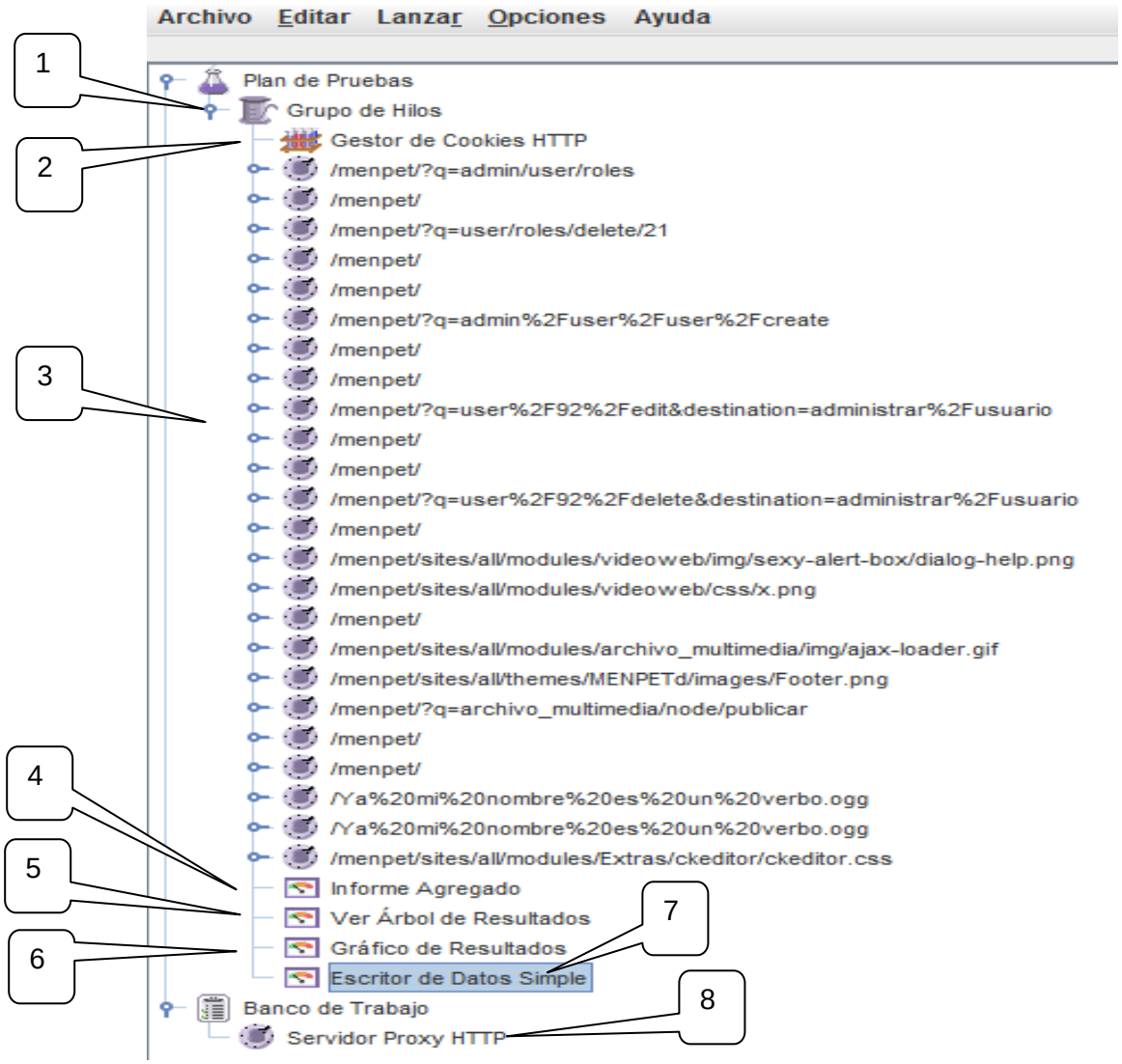


Figura 11: Plan de Prueba en JMeter.

2.8. Diseño de pruebas de usabilidad

A la hora de calificar la calidad de una aplicación web, uno de los factores más importantes es la usabilidad. Es el atributo más visible ya que determina el grado de satisfacción del usuario respecto de la aplicación web; de ello depende que sea utilizada o no. Un sistema es usable si su funcionamiento es correcto, si es eficiente, fácil de aprender, recordar y subjetivamente agradable. La usabilidad para sitios web es muy importante por varias razones, primeramente la web se ha convertido en un elemento clave en el desarrollo de las empresas, debido a que estas ofrecen información y servicios a través de la web. La usabilidad es un factor estratégico fundamental para conseguir un máximo aprovechamiento de estos recursos. Otras de la razones por la que es tan importante tener bien

controlado este factor de la calidad son los usuarios, estos pueden ir fácilmente de un sitio a otro, en la mayoría de los casos solo demoran 1 o 2 minutos en conocer el funcionamiento.

Para la realización de esta prueba a la plataforma VideWeb se emplea como herramienta un *test* de usabilidad. Ahora se hace factible analizar cómo se lleva a cabo este procedimiento.

2.8.1. Procedimiento

El procedimiento está organizado en tres fases. En la primera fase o fase de Planificación, se lleva a cabo la planificación del trabajo, el aseguramiento de los medios tecnológicos y documentos necesarios para llevar a cabo la evaluación.

La fase de Evaluación estará encaminada a aplicar el test de usabilidad con el fin de detectar todas las dificultades referentes a la usabilidad que presente la plataforma VideoWeb. Es decir se recogen las no conformidades encontradas y se llena el registro de no conformidades para que posteriormente, el equipo de desarrollo proceda a corregirlas.

La tercera fase es la de valoración de los resultados, es en la que se recopilan los resultados principales de la evaluación y se elabora el informe de resultados, el cual recogerá las conclusiones de la evaluación.

2.8.2. Test de Usabilidad

El *test* de usabilidad es el artefacto más importante del procedimiento, ya que a través de su aplicación se podrán obtener las dificultades con las que se podría encontrar el usuario final, y por tanto es necesario eliminar o mitigar en caso de no ser posible su eliminación por completo. Es la herramienta que se define para apoyar la ejecución de dicho procedimiento, cuya estructura se basa en las sub-características abordadas en el modelo de Calidad definido por la ISO 9126-1, donde se definen además un conjunto de criterios o especificaciones de usabilidad para cada una de estas sub-características. Cada uno de estos criterios podrá ser verificado a partir de un cuestionario, de forma tal que facilite la práctica de la evaluación. Cada pregunta se responderá con Si o No y cada una tienen definida la respuesta ideal, además de una argumentación en algunos casos que permite fundamentar mejor el aspecto a evaluar. En caso de no responderse de acuerdo al ideal se identifica pues un problema de usabilidad y el mismo deberá registrado con las recomendaciones asociadas.
A continuación se expone el *test* de prueba confeccionado para llevar a cabo las pruebas de usabilidad.

Comprensibilidad

Ayuda

1. ¿Posee una sección de Ayuda?

Respuesta: Sí___ No___ Ideal: En correspondencia con la respuesta que se de en la pregunta 1.1

1.1 ¿Es verdaderamente necesaria?

Respuesta: Sí___ No___

Ideal: Existen sitios que tienen establecidas aplicaciones que, por su grado de complejidad, requieren de asistir al usuario durante el proceso de ejecución. Debe integrarse la ayuda al contenido de las páginas que alojan estas aplicaciones. Siempre que se pueda prescindir de ella simplificando los elementos de navegación e interacción, debe omitirse esta sección.

2. ¿El enlace a la sección de Ayuda está colocado en una zona visible y "estándar"?

Respuesta: Sí___ No___ Ideal: Sí

La zona de la página más normal para incluir el enlace a la sección de Ayuda, es la superior derecha.

Estructura y Navegación

3. ¿Es la página web obvia y auto-explicativa?

Respuesta: Sí___ No___

Ideal: El visitante debe entender qué hace la página web, cómo funciona y cómo puede ir a otro lado

4. Contiene:

Una estructura clara. Respuesta: Sí___ No___ Ideal: Sí

Ayudas visuales moderadas. Respuesta: Sí___ No___ Ideal: Sí

Vínculos fácilmente identificables. Respuesta: Sí___ No___ Ideal: Sí

5. ¿Se utilizan convenciones?

Respuesta: Sí___ No___ Ideal: Sí

Las convenciones son muy útiles para reducir la curva de aprendizaje y el tener que pensar cómo funcionan las cosas. Con ellas ganas la confianza y credibilidad de tus usuarios. Entiende qué es lo que esperan los usuarios de la navegación de un sitio, de la estructura del texto y del lugar de la búsqueda. Si las convenciones están bien aplicadas, aunque la página web esté en otro idioma, estos usuarios podrán ubicarse en ella.

6. Los elementos visuales deben mostrar claridad, no presentar ambigüedad. ¿Esto se cumple?

Respuesta: Sí___ No___ Ideal: Sí

7. Las imágenes con enlace, ¿se reconocen como seleccionables?

Respuesta: Sí___ No___ Ideal: Sí

7.1 ¿Incluyen un atributo 'titulo' describiendo la página de destino?

Respuesta: Sí___ No___ Ideal: Sí

En este sentido, también hay que cuidar que no haya imágenes que parezcan enlaces y en realidad no lo sean.

8. Cuando los usuarios retornan al sistema, recuerdan como llevar a cabo las tareas fundamentales.

Respuesta: Sí___ No___ Ideal: Sí

9. Si existe una imagen en un ícono o botón, debe ser referente a la tarea a la que se refiere.

Respuesta: Sí___ No___ Ideal: Sí

10. ¿Se ha controlado que no haya enlaces que no llevan a ningún sitio?

Respuesta: Sí___ No___ Ideal: Sí

11. ¿Los enlaces son fácilmente reconocibles como tales?

Sí___ No___

11.1 ¿su caracterización indica su estado (visitados, activos)?

Respuesta: Sí___ No___ Ideal: Sí

Los enlaces no sólo deben reconocerse como tales, sino que su caracterización debe indicar su estado (para orientar al usuario).

12. ¿Se utilizan “breadcrumbs” o “migas de pan”?.

Respuesta: Sí___ No___ Ideal: Sí

Cuando se implementen procesos que impliquen una determinada consecución de pasos, es aconsejable la creación de mecanismos que orienten al usuario e informen sobre la etapa del proceso en la que se encuentran. Esto se conoce como "breadcrumbs" o "migas de pan".

13. ¿Es posible que el usuario pueda controlar las actividades que ejecuta en un sitio?

Respuesta: Sí___ No___ Ideal: Sí

Es importante que el usuario experimente la sensación y pueda controlar las actividades que ejecuta en un sitio. En ocasiones, el usuario se equivoca y debe existir un mecanismo de cancelación o vuelta al inicio que lo ayude a recomenzar la tarea.

14. Las opciones en la navegación están ordenadas en el orden más lógico o de alguna forma orientadas a las tareas más importantes.

Respuesta: Sí___ No___ Ideal: Sí

15. El sistema requiere muy poco desplazamiento y emplea métodos para evitar el abuso del *scroll.*

Respuesta: Sí___ No___ Ideal: Sí

16. Con solo un vistazo a la página de inicio el usuario que ingresa por primera vez puede entender por dónde comenzar.

Respuesta: Sí___ No___ Ideal: Sí

17. La página de inicio tiene una dirección URL fácil de recordar.

Respuesta: Sí___ No___ Ideal: Sí

18. El diseño de la página de inicio va a animar a los usuarios a utilizar el sistema.

Respuesta: Sí___ No___ Ideal: Sí

19. La página de inicio es diferente a las demás páginas de la aplicación.

Respuesta: Sí___ No___ Ideal: Sí

20. Cada página contiene el logo de la marca de la compañía para que el usuario sepa que se mantiene en el mismo sitio.

Respuesta: Sí___ No___ Ideal: Sí

Lenguaje y Redacción

21. ¿Se utilizan frases cortas y concisas?

Respuesta: Sí___ No___ Ideal: Sí

22. Cada página está claramente etiquetada con un título útil y descriptivo.

Respuesta: Sí___ No___ Ideal: Sí

23. El sitio está libre de errores tipográficos y de errores ortográficos.

Respuesta: Sí___ No___ Ideal: Sí

24. Los textos de los *links* son lo suficientemente largos para ser entendidos, pero lo suficientemente cortos para evitar el corte de palabras en diferentes renglones (especialmente si son usados en una lista de navegación).

Respuesta: Sí___ No___ Ideal: Sí

25. La página de inicio tiene una dirección URL fácil de recordar.

Respuesta: Sí___ No___ Ideal: Sí

26. El diseño de la página de inicio va a animar a los usuarios a utilizar el sistema.

Respuesta: Sí___ No___ Ideal: Sí

27. La página de inicio es diferente a las demás páginas de la aplicación.

Respuesta: Sí___ No___ Ideal: Sí

28. Cada página contiene el logo de la marca de la compañía para que el usuario sepa que se mantiene en el mismo sitio.

Respuesta: Sí___ No___ Ideal: Sí

Formularios

29. Los campos de entrada contienen valores predeterminados cuando así se requiera.

Respuesta: Sí___ No___ Ideal: Sí

30. Las etiquetas para los campos explican claramente cuáles campos son requeridos (obligatorios).

Respuesta: Sí___ No___ Ideal: Sí

31. Las cajas de texto en los formularios tienen el tamaño adecuado para el dato que se debe introducir.

Respuesta: Sí___ No___ Ideal: Sí

32. Los formularios son validados cuando la información es enviada.

Respuesta: Sí___ No___ Ideal: Sí

33. El sistema hace fácil corregir los errores (ej. cuando un formulario está incompleto, resaltar el elemento donde la corrección debe ser hecha).

Respuesta: Sí___ No___ Ideal: Sí

34. Las etiquetas están cerca de los campos del formulario (ej. las etiquetas están justificadas a la derecha).

Respuesta: Sí___ No___ Ideal: Sí

Control y Retroalimentación

35. ¿Se informa al usuario de lo que ha pasado?

Respuesta: Sí___ No___ Ideal: Sí

Por ejemplo, cuando un usuario valora un artículo o responde a una encuesta, se le debe informar de que su voto ha sido procesado correctamente.

36. Cuando se produce un error, ¿se informa de forma clara y no alarmista al usuario de lo ocurrido y de cómo solucionar el problema?

Respuesta: Sí___ No___ Ideal: Sí

Siempre es mejor intentar evitar que se produzcan errores a tener que informar al usuario del error.

Atracción

Esquema de colores

Se abordan 6 cuestiones que deberían tenerse en cuenta para la correcta percepción y utilización del color en diseño web:

1. Impacto emocional

Teniendo en cuanta la sensación emocional que provoca la navegación por el sitio o el trabajo con la aplicación web y las combinaciones de colores que se utilizan en el diseño. ¿Se ajusta dicha sensación emocional al contenido u objetivo del sitio o aplicación web?

Respuesta: Sí___ No___ Ideal: Sí

Sensaciones por colores:

Figura 12: Sensaciones por colores.

- ✓ Colores análogos: Se utilizan de manera adjunta y producen una sensación de armonía.
- ✓ Colores complementarios: Cuando son usados producen un efecto de agresividad, provocado por el máximo contraste al utilizarlos juntos.
- ✓ Colores monocromáticos: Al utilizarlos producen una sensación de unidad y estabilidad se pueden usar con diferente intensidad (más claro o más oscuro) esto va a depender de la luz.
- ✓ El color rojo se utiliza en medicina como sensación de peligro para las ALARMAS

2. Enfoque del usuario

¿Son las áreas donde el usuario enfoca su atención las de mayor interés por el equipo de desarrollo?

Respuesta: Sí___ No___ Ideal: Sí

Figura 13: Áreas de mayor interés

Lo más frecuente es la aplicación de colores neutros o pasteles en general y colores vivos en aquellas zonas en las que se desea que los visitantes se enfoquen.

3. Efectividad en comunicar información

¿Existe contraste entre los colores de fondo con los textos incluidos?

Respuesta: Sí___ No___ Ideal: Sí

Figura 14: Contraste entre colores de fondo y textos

Es recomendable a los diseñadores web considerar incluir un color suave en lugar del blanco, algo que podría hacer que los usuarios se queden por un tiempo. Así se aseguraría que lo que se quiere decir a través de los sitios (en especial si contienen mucho texto) llegue como se desea y no muera en el intento.

4. Efecto de las combinaciones

¿Existe armonía entre lo colores empleados en el diseño?

Respuesta: Sí___ No___ Ideal: Sí

Figura 15: Armonía entre colores.

5. Impacto y límites de la variedad

¿Es monocromático el diseño del sitio?

Respuesta: Sí___ No___ Ideal: Sí

¿Existe mucha distracción en el usuario durante la su navegación por el sitio o aplicación web?

Respuesta: Sí___ No___ Ideal: Sí

En el uso del color hay un rango de colores para aplicar en diseño, siempre utilizar entre 3 y 5 colores. Menos de 3 hará que el sitio se vea monocromático, más de 5 logrará un nivel interesante de distracción (que se intenta evitar).

6. Sugestión subliminal

¿Provoca reacciones negativas en el usuario la navegación o utilización de la aplicación?

Respuesta: Sí___ No___ Ideal: No

¿Contiene mensajes subliminales que provoquen estas reacciones?

Respuesta: Sí___ No___ Ideal: No

Figura 16: Reacciones adversas

El perfecto ejemplo de reacciones adversas sería ese párrafo con palabras en diferentes colores dónde encontraremos la palabra verde en color rojo, para quienes lo han intentado saben cómo se puede confundir el cerebro a partir de esta práctica.

Es necesario dar un mensaje claro aún desde los colores. Lo ideal siempre es reducir al mínimo posible los efectos subliminales del color, si se tiene un sitio de ecología por ejemplo lo ideal será utilizar tonos verdes.

Tipografía

1. ¿Se utilizan máximo 3 tipos de letras en cómo máximo 3 tamaños?

Respuesta: Sí___ No___ Ideal: Sí

3. ¿El texto que se muestra en la aplicación o sitio web es legible?
Respuesta: Sí___ No___ Ideal: Sí
Se recomienda escribir el cuerpo de los textos más o menos largos en letra preferiblemente verdana o arial, y en un tamaño de 12 puntos, resulta importante la rapidez y comprensión de los textos en función del tipo de letra.

2.10 Prueba de estructura

Para evaluar el estado de los enlaces RUP propone las pruebas de estructura. Los enlaces son los elementos fundamentales que constituyen un sistema web, estos permiten la fácil navegación entre una página y otra. El insuficiente acceso a los recursos debido al estado de los enlaces, desmotiva al cliente en cuanto a la exploración y navegación de los mismos.

Los enlaces que están dirigidos a destinos inexistentes se clasifican como enlaces rotos, cualquier recurso de un sitio que no sea destino de algún hipervínculo se clasificará como archivo o contenido huérfano. Para este tipo de prueba no es necesario un diseño definido para su ejecución. Estas

pruebas se centran en la evaluación de la adherencia del destino de la prueba a su diseño y formación. La existencia de vínculos rotos o contenido huérfano en las aplicaciones web implica falta de calidad por lo que no es recomendable, debido a que no permite a los usuarios navegar correctamente el sitio.

La prueba de estructura se implementa y ejecuta para verificar que todos los enlaces tanto estáticos como activos estén conectados correctamente. Estas pruebas incluyen:

- **Verificación de que se muestra el contenido correcto (texto, gráficos, etc.) de cada enlace.** Deben comprobarse todos los enlaces para garantizar que se muestra el contenido de destino correcto a los usuarios.
- **Comprobación de que no hay enlaces rotos.** Los enlaces rotos son los enlaces cuyo contenido de destino no se puede encontrar. Los enlaces pueden estar rotos por muchos motivos, incluidos el desplazamiento, la eliminación o el cambio de nombre de los archivos de contenido de destino. Los enlaces también pueden estar rotos debido a un uso incorrecto de la sintaxis, incluidos los dos puntos, las barras inclinadas o las letras que falten.
- **Verificación de que no hay contenido huérfano.** El contenido huérfano son los archivos que no tienen enlaces "de entrada" en la aplicación es decir, no se puede acceder a su contenido ni se puede presentar. Debe investigarse con cuidado el contenido huérfano para determinar la causa.

2.11 Conclusiones

Es una buena práctica utilizar herramientas que automaticen los procesos de pruebas, gracias a las ventajas que traen en conjunto con su utilización. Para lograr una mayor calidad y eficacia en el proceso de automatización es necesaria una amplia visión de software que se necesita probar; distinguir las áreas que se necesitan probar y las prioridades de las mismas; llevar a cabo la concepción de las pruebas con el régimen que dictaminan las pautas para su desarrollo y además un buen acondicionamiento del ambiente de las pruebas.

Capítulo 3

Resultados de las pruebas

3.1. Introducción

Luego de haber llevado a cabo todo el proceso de pruebas en este capítulo se hace indispensable documentar todos los resultados obtenidos, con el objetivo de realizar un análisis de los errores encontrados durante todo el proceso de prueba. Ello posibilita tener un registro de estos y corregirlos en el menor tiempo posible, garantizando que la plataforma VideoWeb sea entregada al usuario final con la menor cantidad de defectos posibles. Las no conformidades[7] encontradas pueden clasificarse por su nivel de importancia en significativas, no significativas o recomendación. Las significativas se identifican si el sistema se detiene y no puede avanzar, o cuando se aprecian errores de validación y las no significativas son efectos no significativos en la funcionalidad y usabilidad del sistema. A continuación se registran todos los resultados obtenidos durante la ejecución de las pruebas a la plataforma VideoWeb.

3.2. Resultados de las pruebas de caja negra

Para la ejecución de las pruebas de caja negra se usó como herramienta el Selenium IDE que es un plugin para Firefox, este permite realizar juegos de pruebas sobre aplicaciones web. Para ello permite grabar las acciones realizadas sobre el navegador en un *script*, el cual se puede editar cambiando los valores de los parámetros deseados, para adaptarse a los diferentes escenarios. Los resultados se muestran en el panel de comandos donde se irán marcando en verde los que se ejecuten correctamente y en rojo los que no. Las pruebas que realiza son como las que haría cualquier usuario desde un navegador, con la ventaja de que las hace mucho más rápido y evita el trabajo repetitivo de probar una y otra vez lo mismo de forma manual. Con la aplicación de los casos de pruebas se obtuvieron los siguientes resultados.

Tabla 13: Resultados de las pruebas de caja negra

[7] una no conformidad es el incumplimiento de un requisito.

Secciones	Resultados esperados	Resultados obtenidos
Eliminar archivo multimedia	El sistema elimina el archivo y se actualiza la BD.	La prueba es satisfactoria. *Eliminar archivo multimedia* click link=Gestionar archivos click link=Eliminar click edit-eliminar
Crear publicación de archivo multimedia	Se crea una nueva publicación de archivo multimedia.	La prueba es satisfactoria. *Crear publicación* click link=Gestionar publicaciones click link=Crear publicación de type edit-title prueba select id_archivo_multimedia label=I Am click LOCATOR_DETECTION_FAILED click edit-submit
Eliminar publicación de archivo multimedia	Se elimina la publicación de archivo multimedia.	La prueba es satisfactoria. *Eliminar publicación* click link=Gestionar publicaciones de click //div[@id='block-- click edit-eliminar
Publicar archivo multimedia	Cambia el estado de la publicación de archivo multimedia a publicado y se habilita el archivo multimedia en el servidor de streaming.	La prueba es satisfactoria. *Publicar archivo multimedia* open /menpet/?q=taxonomy/term/51 type edit-name administrador type edit-pass adminvideoweb click edit-submit click link=Gestionar publicaciones de click status_265 click BoxConfirmBtnOk
Dejar de publicar archivo multimedia	Se marca la publicación de archivo multimedia como no publicada y se deshabilita el archivo multimedia asociado del servidor de streaming.	La prueba es satisfactoria.

		Dejar de publicar archivo multimedia open /menpet/?q=taxonomy/term/51 type edit-name administrador type edit-pass adminvideoweb click edit-submit click link=Gestionar publicaciones de click status_265 click BoxConfirmBtnOk
Reproducir archivo multimedia	Se reproduce el archivo multimedia.	La prueba es satisfactoria. *Reproducir archivo* open /menpet/ click link=Estrenos (2) click //div[@id='art- waitForPopUp ventana1 30000 selectWindow name=ventana1 click //img[contains(@src,'http:
Buscar archivos multimedia	Se muestra un listado con los archivos multimedia que coincidan con el criterio de búsqueda.	La prueba es satisfactoria. *Buscar archivo* open /menpet/ type edit-search-theme-form-1 casa click edit-submit-1
Crear cuenta de usuario.	Queda registrado el usuario en el sistema.	La prueba es satisfactoria. *Crear cuenta* type edit-name ivania11 type edit-mail ivania11@uci.cu type edit-pass-pass1 ivania type edit-pass-pass2 ivania click edit-submit
Autenticar	El sistema brinda acceso al usuario en dependencia del rol y muestra un mensaje de bienvenida.	La prueba es satisfactoria. *Autenticar* open /menpet/ type edit-name administrador type edit-pass adminvideoweb clickAndWait edit-submit clickAndWait link=Terminar

3.3. Resultados de las pruebas de caja blanca

Las pruebas de caja blanca fueron realizadas de forma manual probando directamente en el Netbeans para PHP. Al aplicar los casos de pruebas de caja blanca a cada uno de los procedimientos o métodos por separado se obtuvieron los siguientes resultados.

Tabla 14: Resultados de las pruebas de caja blanca

Caso de prueba	Resultados esperados	Resultados obtenidos
Adicionar archivo multimedia	El sistema verifica el formato del archivo y los datos asociados a este, sube el archivo multimedia para el servidor de medias y crea la referencia al mismo en la base de datos. Muestra un mensaje informando al usuario que la operación se ha realizado correctamente.	Satisfactorio
Eliminar archivo multimedia	El sistema elimina el archivo del servidor de medias, elimina todas las referencias a este en la base de datos y muestra un mensaje indicando que el archivo fue eliminado.	Satisfactorio
Crear publicación de archivo multimedia	El sistema almacena los datos de la publicación en la base de datos y muestra un mensaje indicando que la publicación se ha creado correctamente.	Satisfactorio
Modificar publicación de archivo multimedia	El sistema actualiza los datos de la publicación de archivo multimedia y muestra un mensaje indicando que los datos de la publicación han sido actualizados de forma satisfactoria.	Satisfactorio
Publicar archivo multimedia	El sistema cambia el estado de la publicación de archivo multimedia ha publicado, se habilita el archivo multimedia en el servidor de streaming y muestra un mensaje indicando que el archivo fue publicado satisfactoriamente.	Satisfactorio

Buscar archivos multimedia	El sistema busca en los metadatos de los archivos multimedia almacenados en la base datos y muestra el resultado en una lista de archivos multimedia.	Satisfactorio
Crear cuenta de usuario.	El sistema almacena los datos del usuario.	Satisfactorio

3.4. Resultados de las pruebas de carga y estrés

Para expresar el resultado de las pruebas es necesario tener en cuenta las condiciones del escenario donde se encuentra la aplicación, tanto del hardware como software. Se realiza una primera prueba en un ambiente con hardware: 1 GB RAM, procesador Core 2 Duo a 2.2 GHz y tarjeta de red Intel(R) 82562V y con software: Apache2 como servidor web, con una memoria máxima de 128mb y un número máximo de hilos concurrentes ilimitado, instalado en Linux, usando como servidor de base de datos PostgresSQL y lenguaje PHP, en este ambiente no fue posible atender solicitudes de 55 usuarios concurrentes, debido al consumo excesivo de CPU y memoria RAM. Se decide realizar la prueba en un ambiente con características de hardware superior, 3 GB RAM, procesador Intel Centrino Duo a 3.2 GHz y tarjeta de red Intel Ethernet.

Al realizar la simulación de 100 usuarios, con 12 peticiones por usuarios con un período de subida de 500 segundos (cada 5 segundos se conecta un nuevo usuario), el servidor es capaz de atender 1200 peticiones, realizando 1.3 solicitudes por segundos, se observa que las pruebas se han realizado sin errores. Se dobla el número de usuarios y se procede a simular con 200, realizando en esta ocasión 5 peticiones por usuario con un período de subida de 1000 segundos, se generan 1000 peticiones, siendo atendidas 59.8 peticiones por minuto, concurrentemente accedieron 100 usuarios de los 200 que realizaron la prueba, la aplicación se mantuvo estable sin presentar errores. Para la simulación de 300 usuarios se establece un período de subida para el cual cada nuevo hilo se inicializa cada 4 segundos, generando 1500 peticiones, concurrentemente accedieron a la aplicación 200 usuarios, donde se pudo apreciar que el consumo de CPU era demasiado, utilizando el 100% en ocasiones, también la memoria RAM es usada en gran medida, lo que provoca que la máquina usada como servidor deje de prestar servicios.

La siguiente tabla muestra el estado en el que se encontraba el servidor en el momento de la ejecución de las pruebas.

Tabla 15: Datos de las pruebas

	Prueba 1	Prueba 2	Prueba 3
Total conexiones	100	200	300
Usuarios Concurrentes	83	100	200
Peticiones atendidas	1200	1000	1500
CPU	99.0%	98.0%	99%
Memoria RAM	89.4%	28.9%	90%
%error	0.0%	0.0%	0.0%

Al observar los resultados obtenidos en la tabla 15, se puede decir que en este ambiente de hardware no es posible atender solicitudes de más de 300 usuarios concurrentes, ya que aumentaría considerablemente el tiempo de respuestas, quedando estas en una fila de espera hasta llegar a un punto en el que el servidor no pueda atenderlas, provocando un colapso total en este.

3.5. Resultados de las pruebas de usabilidad

Para la realización de las pruebas de usabilidad se aplicó un cuestionario, al procesar los resultados del mismo se registraron las principales deficiencias y errores encontrados en la plataforma VideoWeb en cuanto a usabilidad.

Tabla 16: Resultados de las pruebas de usabilidad

Principales dificultades y errores encontrados	Clasificación
La aplicación no posee una sección de ayuda.	Recomendación
No se utilizan migas de pan para indicar una determinada secuencia de pasos y orientar a los usuarios sobre la etapa del proceso en la que se encuentran.	Recomendación
La página de inicio no tiene una dirección URL fácil de recordar	Significativa

Los elementos visuales deben mostrar claridad, no presentar ambigüedad. Las imágenes presentes en la página de inicio deberían ser un enlace a la sección donde se encuentran dichas medias.	Recomendación

3.6. Resultados de las pruebas de estructura

Para la ejecución de las pruebas de estructura se tuvo en cuenta los menús principales de la aplicación, verificando que cada enlace se correspondía con la funcionalidad adecuada. Para realizar esta prueba se utilizó la herramienta Xenu, este es un programa que analiza todos los enlaces que ha encontrado en la aplicación y marca en rojo los enlaces deficientes, al acabar la prueba genera un reporte de los resultados. Finalizada la prueba se detectó que de las 203 URLs analizadas la plataforma VideoWeb cuenta con un enlace roto y no presenta contenidos huérfanos, los enlaces analizados satisfactoriamente representan el 99,01%, lo que garantiza que el producto cuenta con una buena calidad en cuanto a la estructura de su sistema.

En el gráfico que se muestra a continuación se representa de forma clara el nivel de error con el que cuanta el sistema después de finalizar la prueba de estructura.

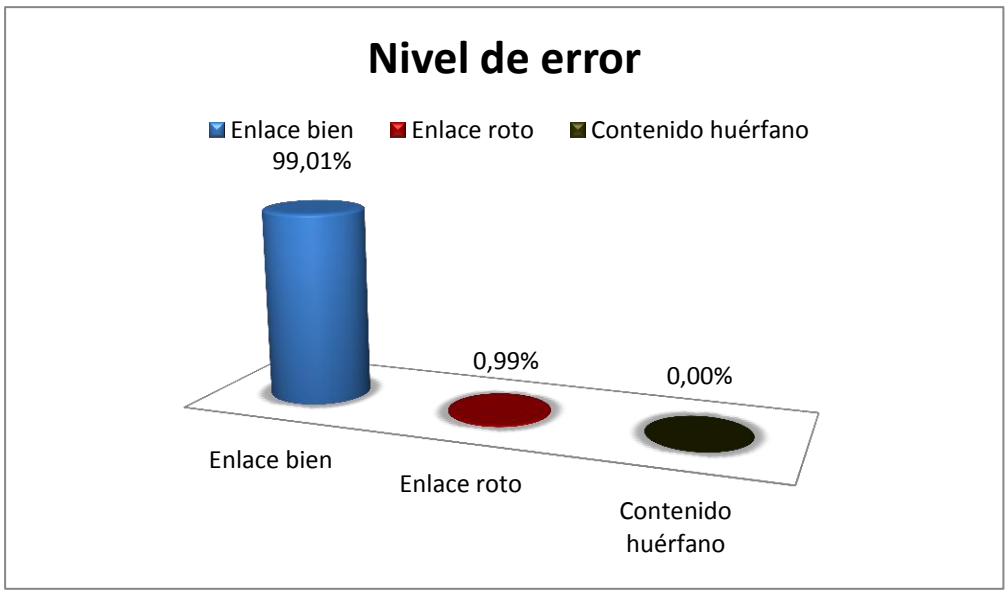


Figura 17: Nivel de errores

3.7. Resultados generales

La mayoría de los errores que se han detectado en el sistema no significan amenaza alguna para el funcionamiento estable de la aplicación. Se detectan varios aspectos en cuanto a la usabilidad que se recomienda mejorar para un mayor aprovechamiento de los recursos y para que los usuarios finales se sientan a gusto navegando por la plataforma, para lograr esto es recomendable la implementación de una ayuda del sistema que pueda guiar a los usuarios en su utilización. En cuanto a la estructura del sitio se detectó un enlace roto, pero este no es significativo en el funcionamiento de la aplicación, la misma no cuenta con contenidos huérfanos y el 99,01% de sus enlaces se encuentran en perfecto estado. Durante la aplicación de las pruebas de caja blanca no se detectaron errores en las funcionalidades implementadas, todos los caminos básicos seleccionados para ser probados cumplen con sus objetivos. Con la aplicación de las pruebas de caja negra fue posible detectar que el sistema no traduce cuando se selecciona el idioma inglés, al igual que se encuentran palabras en inglés y en español en el pie de página, fueron las únicas deficiencias detectadas en la aplicación, por otra parte todas las funcionalidades probadas fueron satisfactorias. Finalmente con la prueba de carga y estrés queda demostrado que la aplicación es capaz de atender solicitudes de 300 usuarios.

Figura 18: Resultados generales

3.8. Conclusiones

El proceso de pruebas realizado se llevó a cabo de forma satisfactoria, con el apoyo de los casos de pruebas diseñados se hizo posible detectar errores con los que contaba este sistema. Quedan documentadas todas las deficiencias encontradas durante el proceso de pruebas, también se hicieron algunas recomendaciones para mejorar la usabilidad de la plataforma haciéndola más amigable. La plataforma VideoWeb cuenta con una buena estructura lo que permite a los usuarios navegar

correctamente por el sitio. Luego de analizar todos los resultados obtenidos se puede decir que la plataforma VideoWeb cuenta con una buena calidad, ya que las deficiencias encontradas son mínimas, de fácil corrección y las más significativas son debido al ambiente de hardware con el que se cuenta.

Conclusiones generales

Luego de aplicar todo el proceso de pruebas a la plataforma VideoWeb se obtuvieron los siguientes resultados:

- El manejo adecuado de las herramientas seleccionadas para el desarrollo de las pruebas automatizadas han facilitado el trabajo y mejorado los resultados. Para la interpretación de dichos resultados es necesario contar con gran preparación.
- Las pruebas aplicadas a la plataforma VideoWeb permitieron detectar la mayor cantidad de errores presentes y evaluar el funcionamiento de la misma.
- Las deficiencias más significativas detectadas se deben a que no se cuenta con el ambiente de hardware adecuado para el despliegue de la plataforma.
- La confección del plan de pruebas para la plataforma VideoWeb permitió describir la estrategia, recursos, planificación de las pruebas y el cronograma de pruebas, garantizando con esto que todo el proceso de pruebas fuera de una forma más clara y tangible.
- Se diseñaron todos los casos de prueba posibles, elaborándose juegos de datos para su validación apoyándose en los procedimientos de prueba construidos.
- Se documentaron los resultados correspondientes de las pruebas de estructura, usabilidad, carga, estrés, caja blanca y caja negra aplicadas a la plataforma VideoWeb.

Recomendaciones

Por la importancia que tiene el proceso de pruebas para cualquier sistema informático, se recomienda:

- Profundizar en la fase de prueba, llevando la documentación necesaria cada vez que se apliquen las pruebas.
- Que se le apliquen las pruebas a los caso de usos que no fueron probados completando en su totalidad los CU del proyecto y a los que fueron sometidos a prueba que se le corrijan todos los errores encontrados.
- Que se apliquen pruebas al servidor de streaming.
- Que se apliquen pruebas de seguridad a la plataforma VideoWeb.
- Que se le realicen nuevamente pruebas de carga y estrés a la plataforma VideoWeb, pero en un ambiente donde el hardware se corresponda con los requisitos no funcionales especificados.
- Perfeccionar el plan de pruebas a partir de la retroalimentación obtenida con la aplicación de las pruebas.

Bibliografía

1. McGraw-Hill. *R.S. Pressman.Ingeniería de Software. Un enfoque Práctico.* s.l. : 4ta Edición, 1998.

2. *Norma ISO 8402 UNE (30 páginas).*

3. Pressman, Roger S. *Ingenieria de Software. Un enfoque practico.* España : s.n., 1998. Segunda ed..

4. Hetzel, B. *The complete guide to software testing.* 1998: QED Information Science. Segunda ed. .

5. *IEEE, Standard Glossary of Software Engineering Terminology. Software Engineering Standars.* 1990.

6. Mario G. Piattini, Jose A. Calvo-Manzano, Joaquin Cervera Bravo, and Luis Fernandez Sanz. *Análisis y diseño de aplicaciones informáticas de gestión, una perspectiva de ingeniería del software.* s.l. : Alfaomega, 2004.

7. McCall.J. *Factors in software Quality:General Electric.* 1977.

8. C. Kaner, J. Falk, and H.Q Nguyen. *Testing Computer Software.* s.l. : Van Nostrand Reinhold. , 1993. second edition..

9. Pressman, Roger S. *Ingeniería del Software, Un enfoque práctico.* s.l. : McGrawHill, 2002. Quinta edición .

10. Wiley, John. *Software Engineering Standards.* s.l. : IEEE Press, 2001. ANSI/IEEE Standard 829-1983 .

11. Tai, K.C. *What to do beyond branch testing. ACM Software Engineering Notes.* 1989. 14:58-61.

12. Beizer, Boris. *Software Testing Techniques.* s.l. : Van Nostrand Reinhold, 1990. second edition.

13. Mc-Graw-Hill. *Pressman, R.Ingenieria del software: un enfoque práctico.* 2002.

14. Myers, Glenford J. *The art of software testing.* s.l. : Wiley , 2004. second edition.

15. The Apache Jakarta Project Apache Jmeter. [En línea] The Apache Software Foundation. [Citado el: 29 de Noviembre de 2010.] http://jakarta.apache.org/jmeter/ .

16. 90003, ISO/IEC. *Software engineering — Guidelines for the application of ISO 9001:2000 to computer software.* 2004. first edition.

17. Frederick P. Brooks, Jr. *The Mythical Man-Month-Essays on Software Engineering Reading.* MA : Addison Wesley Longman, 1995. 2nd ed. .

18. Phadke., M.S. *Planning efficient software tests.* s.l. : Crosstalk, 1997.

19. calisoft. Curso calidad. *Introducción a la Pruebas de software.* C. Habana : s.n., 2009.

20. Pozo, Delmys Zulueta. *PROCEDIMIENTO PARA PRUEBA DE ESTRUCTURA EN APLICACIONES WEB.* 2010.

21. [aut. libro] Consultoría de áreas de conocimiento. *Apache JMeter. Manual de usuario v1.1.*

22. Almenares, Liudmila Sánchez. *Cómo realizar Pruebas de Carga y Estrés con JMeter.* UCI : s.n., 2008.

23. Pérez, Carlos García. *JMeter, pruebas de stress sobre aplicaciones web: Grabando y reproduciendo navegaciones.*

24. Mary Belh Chrissis, Mike Konrad, Sandy Shrum. *CMMI for Development v12.* 2006.

25. Collazo, Manuel. *Técnicas de prueba del software. Estrategias de prueba del software.* s.l. : IEEE Standard Glossary of Software Engineering Terminology 1990, 2003.

26. Mitecnologia. [En línea] [Citado el: 15 de noviembre de 2009.] http://www.mitecnologico.com/Main/CalidadDelSoftware.

27. Informática Profesional. [En línea] [Citado el: 10 de 12 de 2009.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=jmeter.