UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 6

TÍTULO: "LIBERACIÓN DEL SISTEMA DE GESTIÓN DE LA INFORMACIÓN DEL PROYECTO MAPEO CEREBRAL HUMANO CUBANO: MÓDULO EXAMEN CLÍNICO."

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO INFORMÁTICO.

AUTORES: Roxana López Velázquez
Eric Benítez García

TUTOR: Ing. Haylín Corujo Numa

Ciudad de La Habana, Junio 2009
"Año del 50 Aniversario del Triunfo de la Revolución"

El precio se olvida, la calidad permanece.

(Anónimo)

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Roxana López Velázquez

_____

Firma del autor

Eric Benítez García Ing.

_____

Firma del autor

Ing. Haylín Corujo Numa

_____

Firma del tutor

Ing. Maikel Laurencio Giralt

_____

Firma del co-tutor

DATOS DE CONTACTO

Tutor

Ing. Haylín Corujo Numa

hcorujo@uci.cu

Co-Tutor

Ing. Maikel Laurencio Giralt

mlaurencio@uci.cu

Asesor

Lic. Sandy Henriquez Villafruela

sandyh@uci.cu

AGRADECIMIENTOS

A nuestra tutora Ing. Haylin Corujo Numa así como a todo el colectivo de profesores del centro.
A todos nuestros familiares que siempre nos han tenido presente.
A nuestros amigos por su preocupación constante.
A todos los que de una forma u otra hicieron posible que llegáramos hasta aquí.
Un agradecimiento muy especial a nuestra Revolución y a nuestro comandante en jefe Fidel Castro Ruz quien nos facilito, con su brillante idea de crear la Universidad de Ciencias Informáticas la gran oportunidad de formar parte de este gran proyecto.

DEDICATORIA

Roxana López:

A mi madre y mis abuelitos, que me enseñaron a dar mis primeros pasos, que estuvieron ahí en cada momento, que no dejaron de apoyarme nunca y confiaron plenamente en mí, a los que ni dedicándoles toda mi vida podré recompensar jamás.

A Yader, mi novio, que ha sido mi mejor amigo y la persona maravillosa con la que siempre he podido contar en estos largos 5 años. Gracias a su apoyo constante y su preocupación que ahora estoy cumpliendo una de mis metas.

A mi papá, por sus 50 años.

A Angelita y Luis, los padres de mi novio, que me han brindado todo su apoyo.

A mi primo Ramón por ser el ejemplo que me instaba a nunca rendirme.

A la Revolución y a nuestro comandante Fidel Castro Ruz.

Eric Benitez:

A mi familia:

A mis tíos por ser incondicionales en el apoyo brindado durante todos los años de mi vida en especial a Nicolas.

A Juana Manolo y Romelio por la preocupación y ayuda otorgada en todo momento.

A Eridanys mi hermanita la cual siempre trata de seguir mi ejemplo sin saber que ella es mi ejemplo a seguir por ser una bella persona que sabe crecerse en los momentos difíciles.

A mi novia Yanet que fue mi mayor logro en el cursar de la vida universitaria y lo seguirá siendo en mi vida como profesional, ya que es la que me apoya y ayuda en cada decisión y paso tomado.

A Zenen y Maria Del Jesus mis padres quienes me enfundaron la ética y el rigor que guían mi transitar por la vida y ser tan especiales, creo que esta es la mejor forma de hacerlos sentir que todo su esfuerzo y sacrificio no fue en vano.

RESUMEN

Todos los procesos de diseño y producción de software en la actualidad demandan de una elevada calidad como condición indispensable en su creación debido a las crecientes exigencias de los clientes y a la competitividad entre las empresas. El interés por la calidad crece de forma continua, a medida que los clientes se vuelven más selectivos y comienzan a rechazar los productos poco fiables o que realmente no dan respuesta a sus necesidades. Cuba ha comenzado a formar parte de ese mercado debido a las perspectivas económicas que brinda. Es por ello que las empresas nacionales se empeñan cada vez más en obtener productos con la mayor calidad posible para incluirse en la competencia.

Las pruebas de software constituyen un elemento crítico para definir la calidad, además de ser una de las fases más importantes en el desarrollo de un sistema. Este trabajo refleja todo el proceso de pruebas que se realizó para propiciar la entrega de un software que satisfaga las necesidades del cliente, poniendo en práctica la estrategia de pruebas definidas para llevar a cabo la actividad con el éxito requerido.

Índice

Índice de Figuras

Índice de Tablas.

INTRODUCCIÓN

El Proyecto Mapeo Cerebral Humano fue lanzado en el año 1993 con el objetivo fundamental de avanzar en el discernimiento e interpretación de la relación que existe entre la estructura y el funcionamiento del cerebro humano, además de crear una gigantesca base de datos que incluya toda la información del cerebro conocida hasta la fecha . En ese mismo año se conformó el Consorcio Internacional para el Mapeo Cerebral (ICBM, por sus siglas en inglés), con el apoyo del Instituto Nacional de Salud de los EE.UU. (NIH, por sus siglas en inglés) con el objetivo de organizar y desarrollar este proyecto a escala mundial.

Cuba a pesar de ser un país bloqueado se ha integrado al proyecto debido a la experiencia de los especialistas del Centro de Neurociencias de Cuba (CNEURO) en el procesamiento de neuroimágenes, en estos momentos coordinador del Proyecto Mapeo Cerebral Humano Cubano (PMCHC). Su objetivo es crear un atlas de la estructura y funcionamiento cerebral aportando conocimientos sobre los sustratos neurales, tanto de la función normal como sus alteraciones en las enfermedades neuropsiquiátricas cubanas.

Dentro de las entidades que apoyan este proyecto la Universidad de Ciencias Informáticas (UCI) colabora con la tarea de implementar un sistema de gestión de los datos que se generan en los exámenes clínicos que se llevan a cabo durante todo el tiempo de vida del producto. El proyecto Mapeo Cerebral Humano Cubano en la UCI consta de 4 módulos, actualmente solo se están desarrollando dos de ellos, Exámenes Clínicos, y MRI., este último para procesamiento de imágenes de resonancia magnéticas.

El módulo Exámenes Clínicos ha desarrollado una aplicación que permite gestionar los datos de los exámenes que se le aplican a sujetos para detectar si los mismos son funcionalmente sanos. Actualmente se encuentra en su etapa de liberación, sin embargo, el hecho de que un producto tenga calidad, implica que debe cumplir con ciertas exigencias, que muchas veces son pasadas por alto, y es por ello, que los productos son entregados con errores en la documentación, poco entendible o inconsistentes, o la interfaz no cuenta con todos los requisitos que se estableció con el cliente.

Para a garantizar la calidad de ejecución del Sistema de Gestión de la Información del Proyecto Mapeo Cerebral Humano Cubano (MCHC) se plantea como **problema científico de la investigación** ¿Cómo verificar la calidad del Módulo Exámenes Clínicos del producto Sistema de Gestión de la Información del Proyecto MCHC?

El problema planteado se enmarca en el **objeto de estudio**: Proceso de Pruebas de Liberación de Software en la Universidad de Ciencias Informáticas.

El objeto de estudio delimita el **campo de acción**: Las Pruebas de Liberación de Software en la facultad 6.

Para dar solución al problema se define como **objetivo general**: Desarrollar un proceso de pruebas de liberación de software al Módulo Exámenes Clínicos del Sistema de Gestión de la Información del Proyecto MCHC.

Del objetivo general se derivan los siguientes objetivos específicos:

- Realizar revisiones a los documentos.
- Diseñar Casos de Pruebas.
- Realizar pruebas de Liberación al Sistema de Gestión de MCHC.

Para dar cumplimiento a los objetivos específicos se definen las siguientes tareas:

- Estudio bibliográfico de los temas de calidad de software, pruebas de software, herramientas y metodologías de las pruebas de software.
- Realización del Plan de Pruebas.
- Revisión de los documentos del Módulo Exámenes Clínicos.
- Realización de Pruebas Exploratorias al Módulo Exámenes Clínicos.
- Realización del Documento de No Conformidades.
- Diseño de Casos de Pruebas.
- Realización de Pruebas Funcionales.
- Realización de Pruebas de Seguridad.
- Realización de Pruebas de Carga y Stress.
- Realización de Pruebas de Atributos de Calidad.

- Análisis de las pruebas aplicadas al Sistema de Gestión del Proyecto MCHC.

La estructuración del presente trabajo se distingue por tres capítulos, en los cuales se expone el diseño, la aplicación y el análisis de los resultados de las pruebas realizadas.

El primer capítulo comprende la revisión bibliográfica sobre el tema tratado con una breve explicación del problema, referenciado a conceptos importantes que urge ser comprendidos para la elaboración del trabajo, la relación entre las pruebas y la calidad de un software, los objetivos y las estrategias a seguir para aplicarlas así como también los tipos de pruebas existentes y los pasos para su diseño.

El segundo capítulo que está dedicado principalmente a describir el proyecto que se prueba, presentar el plan de pruebas general, como factor de éxito para la puesta en práctica de un proceso de pruebas que permita entregar un software de mejor nivel, y exponer cómo se llevaron a cabo las pruebas al sistema.

El tercer capítulo donde se exponen y se analizan los resultados obtenidos.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

Introducción

El desarrollo del software implica una serie de actividades de producción en las que las posibilidades de que aparezca la equivocación humana son comunes. Los errores pueden empezar a darse desde el primer momento del proceso en el que los objetivos pueden estar especificados de forma errónea. Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software ha de ir acompañado de una actividad que garantice la calidad, la prueba de software.

La prueba constituye un elemento crítico para la garantía de la calidad del software y representa una revisión de las especificaciones, del diseño y de la codificación [1]. Constituyen una etapa dentro del desarrollo de cualquier aplicación y un tema importante dentro de la ingeniería de software.

En este capítulo se hace un análisis de la revisión bibliográfica sobre este tema, teniendo en cuenta los objetivos. Se analizan algunos temas como: la relación que existe entre las pruebas y la calidad de un software, los objetivos, principios, técnicas, herramientas y estrategias a seguir a la hora de aplicar las mismas.

1.1 Calidad Del Software.

Sin lugar a dudas la década del 90 se caracterizó por la palabra calidad, normalmente sin importar a que elementos, entes y áreas hacía referencia; al final, todos dicen poseerla y dominarla, pero cómo conocer y dominar un concepto tan amplio, subjetivo y muchas veces ambiguo, que ni siquiera la mayoría de las personas podrían dar una definición exacta de qué es calidad.

La palabra calidad tiene múltiples significados. Se denomina como un conjunto de propiedades inherentes a un objeto que le confieren capacidad para satisfacer necesidades implícitas o explícitas [2].

Si se ve desde la perspectiva del usuario, la calidad implica la capacidad de satisfacer los deseos de las personas dentro de su estilo de vida, esto involucra un equilibrio entre lo objetivo/tangible y lo subjetivo/intangible, ofrecer características beneficiosas y saludables para las personas y su entorno. La calidad de un producto depende de cómo éste responda a las preferencias y a las necesidades de los clientes.

Desde una perspectiva de producto la calidad es, diferenciarse cualitativa y cuantitativamente respecto de algún atributo requerido, esto incluye la cantidad de un atributo no cuantificable en forma monetaria que contiene cada unidad de un atributo.

Desde una perspectiva de producción puede definirse como la conformidad relativa con las especificaciones, a lo que al grado en que un producto cumple las especificaciones del diseño, entre otras cosas, mayor su calidad.

Por otro lado, la calidad del software es uno de los problemas que se afrontan actualmente en la esfera de la computación.

Desde la década del 70, este tema ha sido motivo de preocupación para especialistas, ingenieros, investigadores y comercializadores de software. El software se ha convertido en un activo que determina en gran medida la operatividad de la organización por lo que la calidad del software se convierte en uno de los puntos de atención de las organizaciones actuales y se define como un *conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. Es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, portabilidad, usabilidad, seguridad e integridad. Es medible y varía de un sistema a otro o de un programa a otro* [3]. Propiciarla es una actividad que ha surgido como consecuencia de la fuerte demanda de Sistemas de Software en todos los procesos que se desarrollan en la actualidad; desde programas elementales de contabilidad hasta programas complejos como los espaciales. De allí el esfuerzo que se ha desplegado para obtener software de alta calidad.

1.2 Las Pruebas del Software.

El desarrollo de sistemas de software implica una serie de actividades de producción en las que las posibilidades de que aparezca el fallo humano son numerosas, por lo que ha de ir acompañado de una actividad que garantice la calidad, las Pruebas de Software, procesos que permiten verificar y validar la calidad de un producto.

Son elementos críticos para determinar la calidad de este. Su objetivo es garantizar la calidad del producto desarrollado. Además, esta etapa implica:

- Verificar la interacción de componentes.
- Verificar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.
- Diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

A pesar de que "*las pruebas no pueden asegurar la ausencia de defectos, ya que sólo se puede demostrar que existen defectos en el software"*[1], son parte fundamental antes de entregar el software final.

1.2.1 Objetivos de las Pruebas del Software.

Existen normas que pueden ser tomadas acertadamente como objetivos de las pruebas, estas son:

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error.

Los principales objetivos que se buscan al aplicarles pruebas a un software son:

- Verificar que el producto de software trabaja según el diseño.
- Detallar los errores que fueron encontrados.
- Ofrecer un producto altamente seguro y confiable

- Disminuir la costosa labor de soporte a usuarios insatisfechos, consecuencia de liberar un producto inmaduro. Esto puede mejorar la imagen de la organización desarrolladora (y la credibilidad en ella).

Si la prueba se lleva a cabo con éxito entonces se descubrirá errores en el software. Como ventaja secundaria, la prueba demuestra hasta qué punto las funciones del software parecen funcionar de acuerdo con las especificaciones y parecen alcanzarse los requisitos de rendimiento. Además, los datos que se van recogiendo a medida que se lleva a cabo la prueba proporcionan una buena indicación de la fiabilidad del software y, de alguna manera, indican la calidad del software como un todo.

1.2.2 Principios de las Pruebas del Software.

Antes de la aplicación de métodos para el diseño de casos de prueba efectivos, un ingeniero del software deberá entender los principios básicos que guían las pruebas del software:

- **A todas las pruebas se les debería poder hacer un seguimiento hasta los requisitos del cliente:** Se entiende que los defectos más graves (desde el punto de vista del cliente) son aquellos que impiden al programa cumplir sus requisitos.
- **Las pruebas deberían planificarse mucho antes de que empiecen:** La planificación de las pruebas pueden empezar tan pronto como esté completo el modelo de requisitos.
- **El principio de Pareto es aplicable a la prueba del software:** Dicho de manera sencilla, el principio de Pareto implica que al 80% de todos los errores descubiertos durante las pruebas se les puede hacer un seguimiento hasta un 20% de todos los módulos del programa. El problema, por supuesto, es aislar estos módulos sospechosos y probarlos concienzudamente.
- **Las pruebas deberían empezar por "lo pequeño" y progresar hacia "lo grande":** Las primeras pruebas planeadas y ejecutadas se centran generalmente en módulos individuales del programa. A medida que avanzan las pruebas, desplazan su punto de mira en un intento de encontrar errores en grupos integrados de módulos y finalmente en el sistema entero.
- **No son posibles las pruebas exhaustivas:** El número de permutaciones de caminos para incluso un programa de tamaño moderado es excepcionalmente grande. Por este motivo, es imposible ejecutar todas las combinaciones de caminos durante las pruebas. Es posible, sin embargo, cubrir adecuadamente la lógica del programa y asegurarse de que se han aplicado todas las condiciones en el diseño a nivel de componente.

- **Para ser más eficaces, las pruebas deberían ser realizadas por un equipo independiente:** El ingeniero del software que crea el sistema no es el más adecuado para llevar a cabo las pruebas para el software.

1.2.3 Estrategias de Pruebas de Software

Una estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que llevan a la construcción correcta del software. Debe ser suficientemente flexible para promover la creatividad y adaptabilidad necesaria para adecuar la prueba a todos los grandes sistemas basados en software. Al mismo tiempo la estrategia debe ser rígida para promover un seguimiento razonable de la planificación.

Sintetizando, los objetivos de nuestra estrategia de prueba son:

- Planificar las pruebas necesarias en cada iteración.
- Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar y cómo realizar las pruebas
- Ejecutar las pruebas y manejar los resultados de cada prueba sistemáticamente.

Para conseguir estos objetivos el flujo de trabajo del proceso de Pruebas que se llevará a cabo consta de las siguientes etapas:

- Planificación de las pruebas.
- Diseño de las pruebas.
- Ejecución de las pruebas.
- Evaluación de las pruebas.

1.2.4 Niveles de Pruebas de Software.

En el proceso de pruebas al software existen niveles en los cuales se ejecutan diferentes tipos de pruebas de acuerdo al desarrollo que se haya alcanzado en la aplicación. Estos niveles especifican qué tipos y métodos de pruebas se deben utilizar en cada uno de ellos y cuáles son sus objetivos.

1.2.4.1 Pruebas de Unidad.

Las pruebas de unidad es un proceso para probar los subprogramas, las subrutinas, los procedimientos individuales o las clases en un programa [4].Se trata de las pruebas formales que

permiten declarar que un módulo está listo y terminado (no las informales que se realizan mientras se desarrollan los módulos). Siempre está orientada a caja blanca.

Se habla de una unidad de prueba para referirse a uno o más módulos que cumplen las siguientes condiciones:

- Todos son del mismo programa.
- Al menos uno de ellos no ha sido probado.
- El conjunto de módulos es el objeto de un proceso de prueba.

1.2.4.2 Pruebas de Integración.

Se comprueba la compatibilidad y funcionalidad de los interfaces entre las distintas partes que componen un sistema, estas partes pueden ser módulos, aplicaciones individuales, aplicaciones cliente/servidor, etc. Este tipo de pruebas es especialmente relevante en aplicaciones distribuidas y su principal objetivo es coger los módulos probados mediante la prueba de unidad y construir una estructura de programa que está de acuerdo con lo que dicta el diseño.

Cuando el programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y corregir, se está haciendo uso de la prueba conocida como integración incremental.

Existen diferentes estrategias de integración incremental.

- **La prueba de Integración Descendente:** En esta prueba se integran los módulos moviéndose hacia abajo por la jerarquía de control, comenzando por el módulo de control principal (programa principal). Los módulos subordinados (subordinados de cualquier modo) al módulo de control principal se van incorporando en la estructura, bien de forma primero-en-profundidad, o bien de forma primero-en-anchura.
- **La prueba de Integración ascendente:** Para realizar esta prueba se empieza la construcción y la prueba con los módulos atómicos (es decir, módulos de los niveles más bajos de la estructura del programa). Dado que los módulos se integran de abajo hacia arriba, el proceso requerido de los módulos subordinados siempre están disponible y se elimina la necesidad de resguardos.

- **La prueba de regresión:** Se vuelve a ejecutar un subconjunto de pruebas que se han llevado a cabo anteriormente para asegurarse de que los cambios no han propagado efectos colaterales no deseados.
- **La prueba de Humo:** Es un método de prueba de integración que es comúnmente utilizada cuando se ha desarrollado un producto software empaquetado. Es diseñado como un mecanismo para proyectos críticos por tiempo, permitiendo que el equipo de software valore su proyecto sobre una base sólida.

1.2.4.3 Pruebas de Sistema.

La prueba de sistema tiene como propósito ejercitar profundamente el sistema para verificar que se han integrado adecuadamente todos los elementos del sistema (hardware, otro software, etc.) y que realizan las funciones adecuadas, utilizando para la generación de casos de prueba técnicas de caja negra. Concretamente se busca comprobar que:

- Se cumplen los requisitos funcionales establecidos.
- El funcionamiento y rendimiento de las interfaces hardware, software y de usuario.
- La adecuación de la documentación de usuario.
- Rendimiento y respuesta en condiciones límite y de sobrecarga.

La prueba del sistema está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Algunas de estas pruebas son:

- **Prueba de validación:** Proporciona una seguridad final de que el software satisface todos los requerimientos funcionales y de rendimiento. Además, valida los requerimientos establecidos comparándolos con el sistema que ha sido construido.
- **Prueba de recuperación:** Fuerza un fallo del software y verifica que la recuperación se lleva a cabo apropiadamente.
- **Prueba de seguridad:** Verifica los mecanismos de protección. Se determinan los niveles de permiso de usuarios, las operaciones de acceso al sistema y acceso a datos.
- **Prueba de resistencia:** Enfrenta a los programas a situaciones anormales. Determinan hasta donde puede soportar el programa determinadas condiciones extremas.
- **Prueba de rendimiento:** Prueba el rendimiento del software en tiempo de ejecución. Determina los tiempos de respuesta, el espacio que ocupa el módulo en disco o en memoria, el flujo de datos que genera a través de un canal de comunicaciones, etc.

- **Usabilidad:** Se determina la calidad de la experiencia de un usuario en la forma en la que éste interactúa con el sistema.
- **Prueba de instalación:** Se centra en asegurar que el sistema software desarrollado se puede instalar en diferentes configuraciones hardware y software y bajo condiciones particulares, por ejemplo con espacio de disco insuficiente o continuas interrupciones.

1.2.4.4 Pruebas de Aceptación.

Es la prueba planificada y organizada formalmente para determinar si se cumplen los requisitos de aceptación marcados por el cliente y se realiza una vez que el sistema se ha implantado en su entorno real de funcionamiento. El usuario comprueba en su propio entorno de explotación si acepta el software como está o precisa ser necesario aplicar nuevas optimizaciones y soluciones de fallas. Está enfocada hacia la prueba de los requisitos de usuario especificados y considerada como la fase final del proceso para crear una confianza en que el producto es el apropiado para su uso en explotación. Su principal objetivo es demostrar al usuario que el sistema satisface sus necesidades.

1.2.4.4 Alcance de las pruebas.

De acuerdo al estudio realizado las pruebas de unidad son realizadas por los desarrolladores del equipo, las de integración se llevan a cabo cuando el sistema ha sido desarrollado por módulos o componentes y es necesario determinar que estos funcionen conjuntamente e integrados, como solo se están llevando a cabo pruebas de liberación a un módulo, se le realizarán pruebas de sistema.

1.2.5 Métodos de Pruebas de Software.

Los métodos de diseño de casos de prueba proporcionan un enfoque sistemático a la prueba, así como un mecanismo de ayuda para asegurar la completitud de las pruebas y conseguir la mayor probabilidad de descubrimiento de errores.

Existen dos enfoques para el diseño de casos de prueba:

- **Enfoque estructural o de caja blanca:** consiste en centrarse en la estructura interna (implementación) del programa para elegir los casos de prueba. (Fig. 1)
- **Enfoque funcional o de caja negra:** consiste en estudiar la especificación de las funciones, la entrada y la salida para derivar los casos.

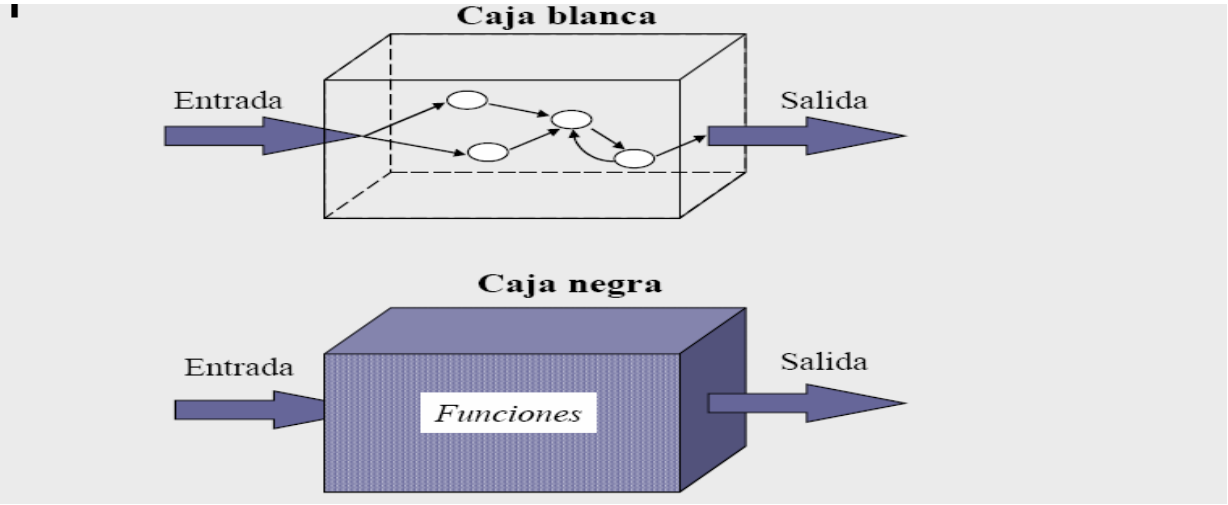

Fig.1 Enfoques de Diseño de Caso de Pruebas de Caja Blanca y Caja Negra

1.2.5.1 Pruebas Estructurales o de Caja Blanca.

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 [1]. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que:

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

La prueba de Caja Blanca usa la estructura de control del diseño procedural para derivar los casos de prueba. Tienen como idea fundamental confeccionar casos de prueba que garanticen que se verifican todos los caminos independientes.

Los errores lógicos y las suposiciones incorrectas son inversamente proporcionales a la probabilidad de que se ejecute un camino del programa. A menudo se cree que un camino lógico tiene pocas

posibilidades de ejecutarse cuando, de hecho, se puede ejecutar de forma regular. Los errores tipográficos son aleatorios. “Los errores se esconden en los rincones y se aglomeran en los límites” **[4]**

Pruebas Del Camino Básico.

La prueba del camino básico es una técnica de prueba de caja blanca 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. [1] Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Pruebas de Condición.

La prueba de condición es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el modulo de un programa. [1] Este método se centra en las pruebas de cada una de las condiciones del programa. Su propósito es detectar no solo los errores en las condiciones de un programa, sino también otros errores en dicho programa.

Las estrategias de prueba de condiciones tienen, generalmente, dos ventajas. La primera, que la cobertura de la prueba de una condición es sencilla. La segunda, que la cobertura de la prueba de las condiciones de un programa da una orientación para generar pruebas adicionales del programa.

Pruebas de Flujo de Datos.

El método de prueba de 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. [1] Las estrategias de prueba de flujo de datos son útiles para seleccionar caminos de prueba de un programa que contenga sentencias if o de bucles anidados.

Dado que las sentencias de un programa están relacionadas entre sí de acuerdo con las definiciones de las variables, el enfoque de prueba de flujo de datos es efectivo para la protección contra errores. Sin embargo, los problemas de medida de la cobertura de la prueba y de selección de caminos de prueba para la prueba de flujo de datos son más difíciles que los correspondientes problemas para la prueba de condiciones.

Pruebas de Bucles.

Los bucles son la piedra angular de la inmensa mayoría de los algoritmos implementados en software. La prueba de bucles es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles. Se pueden definir cuatro clases diferentes de bucles: bucles simples, bucles concatenados, bucles anidados y bucles no estructurados.

1.2.5.2 Pruebas Funcionales o de Caja Negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software, lo que permite al ingeniero de software obtener conjuntos de condiciones de entradas que ejerciten completamente todos los requisitos funcionales de un programa. No es una alternativa a las técnicas de prueba de caja blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores que los de caja blanca [1]. Además, a diferencia de esta, que se lleva a cabo previamente en el proceso de pruebas, la prueba de caja negra tiende a aplicarse durante fases posteriores de la prueba, dado que ignora intencionadamente la estructura de control, centra su atención en el campo de información.

La prueba de caja negra intenta 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.

Mediante las técnicas de prueba de caja negra se obtiene un conjunto de casos de prueba que satisfacen los siguientes criterios

1. Casos de prueba que reducen, en un coeficiente que es mayor que uno, el número de casos de prueba adicionales que se deben diseñar para alcanzar una prueba razonable
2. Casos de prueba que nos dicen algo sobre la presencia o ausencia de clases de errores en lugar de errores asociados solamente con la prueba que estamos realizando.

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.

Análisis de valores límites.

Los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límites (AVL) como técnica de prueba. El análisis de valores límite lleva a una elección de casos de prueba que ejerciten los valores límite.

El análisis de valores límite es una técnica de diseño de casos de prueba que completa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida.

Partición de Equivalencia.

La partición de equivalencia se presenta como un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. Este enfoque se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

El objetivo de partición de equivalencia es reducir el posible conjunto de casos de prueba en uno más pequeño, un conjunto manejable que evalúe bien el software.

El Módulo de Exámenes Clínicos se probará usando el enfoque de caja negra, dado que nuestras pruebas alcanzan el nivel de Sistema y en este las técnicas que se aplican son enfocadas a caja negra. El principal interés se centra sobre la interfaz del software, pretendiendo demostrar con los casos de pruebas que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta. Para ello se utilizará la técnica de partición de equivalencia, siendo esta muy efectiva a la hora de realizar la pruebas sobre la interfaz del software probando la validez de cada entrada de datos o información al sistema, pretendiendo demostrar que las funciones del software son operativas, que cada entrada se acepte de forma adecuada y que a la vez que se produce una salida correcta la integridad de la información externa se mantiene.

1.2.6 Técnicas de Pruebas de Software

Las técnicas de pruebas se aplican para validar y verificar las funcionalidades de un software, su buen uso, fiabilidad, confiabilidad, portabilidad, seguridad y el rendimiento. Cada técnica de prueba posee un objetivo específico y una forma de usarse.

1.2.6.1 Prueba de Funcionalidad

La prueba de funcionalidad se enfoca en requerimientos para verificar que se corresponden directamente a casos de usos o funciones y reglas del negocio. Los objetivos de estas pruebas son verificar la aceptación de los datos, el proceso, la recuperación y la implementación correcta de las reglas del negocio. Este tipo de prueba se basa en técnicas de caja negra, que consisten en verificar la aplicación y sus procesos interactuando con la aplicación por medio de la interface de usuario y analizar los resultados obtenidos.

Función

Pruebas fijando su atención en la validación de las funciones, métodos, servicios, caso de uso.

Seguridad

La Prueba de Seguridad y Control de Acceso se enfoca en dos áreas de seguridad:

Seguridad en el ámbito de aplicación, incluyendo el acceso a los datos y a las funciones de negocios.

Seguridad en el ámbito de sistema, incluyendo conexión, o acceso remoto al sistema.

La seguridad en el ámbito de aplicación asegura que, basado en la seguridad deseada los actores están restringidos a funciones o casos de uso específicos o limitados en los datos que están disponibles para ellos. Se encarga de verificar que un actor pueda acceder solo a las funciones o datos para los cuales su tipo de usuario tiene permiso.

La seguridad en el ámbito de sistema asegura que, solo los usuarios con derecho a acceder al sistema son capaces de acceder a las aplicaciones y solo a través de los puntos de ingresos apropiados. Se encarga de verificar que solo los actores con acceso al sistema y a las aplicaciones, puedan acceder a ellos.

Volumen

La Prueba de Volumen verifica que el software funcione correctamente con volúmenes de datos grandes:

- Máximo (real o físicamente posible) número de clientes conectados, o simulados, todos realizando la misma operación (peor caso de operación) por un período de tiempo extenso.
- Máximo tamaño de base de datos y múltiples consultas ejecutadas simultáneamente.

Este tipo de prueba somete el software a grandes cantidades de datos para determinar si se alcanzan límites que causen la falla del software.

1.2.6.2 Usabilidad

Prueba enfocada a factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente documentación de usuarios y materiales de entrenamiento. Estas pruebas están encaminadas a verificar el aprendizaje (que tan fácil es para los usuarios realizar tareas básicas la primera vez que tienen contacto con el sistema).

1.2.6.3 Desempeño (performance)

En esta prueba se miden y evalúan los tiempos de respuesta, los tiempos de transacción y otros requerimientos sensitivos al tiempo. El objetivo de la prueba es verificar que se logren los requerimientos de performance. La prueba de performance es implementada y ejecutada para poner a

punto los destinos de pruebas de performance como función de condiciones de trabajo o configuraciones de hardware.

Carga

La prueba de carga verifica el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado y así como la verificación de la capacidad del sistema para manejar volúmenes de datos extremos de acuerdo al tiempo de respuesta establecido para el sistema para el sistema. El objetivo es determinar y asegurar que el sistema funciona apropiadamente en circunstancias de máxima carga de trabajo esperada. Además de evaluar las características de performance, como tiempos de respuesta, tiempos de transacciones y otros elementos sensitivos al tiempo. Este tipo de prueba debe realizarse en una máquina dedicada para tener control total y exactitud de mediciones. Las bases de datos usadas para la prueba deben tener un tamaño similar a las reales.

1.2.6.4 Fiabilidad

La prueba de fiabilidad está dirigida a medir el grado en que se puede esperar que un programa lleve a cabo sus funciones esperadas con la precisión requerida.

Estructura

Enfocada a la valoración a la adherencia a su diseño y formación. Este tipo de prueba es hecho a alas aplicaciones Web asegurando que todos los enlaces están conectados, el contenido deseado es mostrado y no hay contenido huérfano.

Resistencia (stress)

Enfocada a evaluar cómo el sistema responde bajo condiciones anormales (extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible).

La prueba de esfuerzo en un tipo de prueba de performance implementada y ejecutada para encontrar errores cuando hay pocos recursos o cuando hay competencia por recursos. Poca memoria o poco espacio de disco pueden revelar fallas en el software que no aparecen bajo condiciones normales de cantidad de recursos. Otras fallas pueden resultar al competir por recursos compartidos como bloqueos de bases de datos o ancho de banda de red. La prueba de esfuerzo también puede usarse

para identificar el trabajo máximo que el software puede manejar. Persigue como objetivo verificar que el software funciona apropiadamente y sin error bajo condiciones de esfuerzo tales como:

- poca memoria o sin disponibilidad de memoria en el servidor
- cantidad máxima de clientes conectados
- múltiples usuarios realizando la misma operación sobre los mismos datos
- peor caso de volumen de operaciones.

1.2.6.5 Técnicas a Realizar.

Luego de haber realizado un estudio de las diversas técnicas que existen y teniendo en cuenta que las pruebas alcanzarán el nivel de pruebas de sistema, se decidió hacerle al Módulo Exámenes Clínicos pruebas funcionales para asegurar el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados, pruebas de seguridad y pruebas de carga y resistencia para probar el software en condiciones anormales y su rendimiento en tiempo de ejecución con grandes volúmenes de datos.

1.2.7 Roles

Los participantes responsables para realizar las actividades en este proceso de pruebas son:

- Diseñador de pruebas: Es responsable de la planificación, diseño, implementación y evaluación de las pruebas. Esto conlleva generar el plan de pruebas y modelo de pruebas, implementar los casos de prueba y evaluar los resultados de las pruebas. Los diseñadores de prueba realmente no llevan a cabo las pruebas, sino que se dedican a la preparación y evaluación de las mismas.
- Probador: Es responsable de desarrollar las pruebas de unidad, integración y sistema, lo que incluye ejecutar las pruebas, evaluar su ejecución, recuperar los errores y garantizar los resultados de las pruebas.

1.3 El Proceso de Pruebas de Liberación de un Software.

Para lograr el buen funcionamiento del producto de software, los requisitos funcionales deben ser verificados y probados en el sistema, debido a que constituyen el comportamiento del software y muestran cómo los casos de uso serán llevados a la práctica. Para llevar a cabo esta operación se hace necesario la utilización de un proceso, que ponga en marcha cómo se deben realizar las pruebas que permitan entregar el software con un mejor nivel.

El proceso de pruebas de software es una tarea técnica, precisa del dominio del lenguaje de programación en el que el artefacto a probar fue generado, también del conocimiento necesario para comprender la arquitectura del sistema implementado y de las implicaciones de tipo lógico que su diseño pueda suponer. [5] No va orientado a demostrar la ausencia de fallos en el software, sino más bien a todo lo contrario: encontrar cuantos fallos existan, por escondidos que se encuentren o difícilmente reproducibles que sean.

El proceso de liberación de un software constituye una guía fundamental para aplicar las pruebas de una manera organizada, eficiente y con los elementos necesarios que conduzcan a un mejor funcionamiento del mismo.

El proceso de pruebas también debe adaptarse a las necesidades del proyecto y consta esencialmente de las siguientes etapas:

- Análisis de pruebas
- Diseño de las pruebas
- Ejecución de las pruebas
- Evaluación de resultados

1.3.1 Verificación y Validación.

La prueba de software es el proceso de ejecutar el software de manera controlada, usualmente se usa en conjunto con los términos verificación y validación.

Verificación se refiere al conjunto de actividades que aseguran que el software implementa correctamente una función específica. No es más que la revisión o las pruebas de los elementos, incluyendo el software, para obtener conformidad y consistencia con una especificación asociada. Las pruebas de software son solo un tipo de Verificación. [6]

Validación se refiere a un conjunto de diferentes actividades que aseguran que el software construido se ajusta a los requisitos del cliente. No es más que el el proceso de evaluación de un sistema o de uno de sus componentes durante o al final del proceso de desarrollo para determinar si satisface los requisitos marcados por el usuario. [6]

Del proceso de Verificación se observa la importancia de verificar cada uno de los productos que se van construyendo, bajo la asunción de que si lo que se va construyendo es todo correcto, también lo será el producto final. Igualmente, se observa que el proceso de Validación resalta la importancia de comprobar el cumplimiento de los objetivos de los requisitos y del sistema final.

1.4 Plan de Pruebas.

Un plan de pruebas está constituido por la visión global del proceso de pruebas, y la definición de actividades y roles involucrado en cada una de ellas. Cada prueba debe dejar claro qué tipo de propiedades se quieren probar (corrección, robustez, fiabilidad, amigabilidad), cómo se mide el resultado, especificar en qué consiste la prueba y definir cuál es el resultado que se espera (identificación, tolerancia).

El plan de pruebas (ver anexo 1) describe la estrategia, recursos, planificación y cronograma donde se refleja la planificación de las pruebas. La estrategia de prueba incluye la definición del tipo de pruebas a realizar para cada iteración y sus objetivos, el nivel de cobertura y el porcentaje de prueba que deberían ejecutarse con un resultado específico; proporciona además el marco dentro del cual el equipo desarrolla las pruebas trabajando con los recursos y la planificación dada.

El propósito del plan de pruebas es explicitar el alcance, señalar el enfoque, los recursos y el esquema de actividades de prueba, así como los elementos a probar, las características y el personal responsable.

Una vez establecido el plan de pruebas, se revisa con el equipo de desarrollo y una vez aprobado se inicia el proceso de pruebas. A partir de allí el plan de pruebas se utiliza como referente para realizar control y seguimiento al proyecto a lo largo de todas las etapas a fin de garantizar el cumplimiento de los compromisos, en caso de desviaciones, el ingeniero de pruebas puede tomar las medidas respectivas.

1.5 Automatización de Prueba de carga y rendimiento.

Sería ineficiente realizar las pruebas de carga y stress de manera manual y prácticamente imposible comprobar si se cumplen los requerimientos descritos por el cliente. Para esta técnica de prueba existen diversas herramientas, que permiten realizar pruebas de carga, pruebas de stress o ambas inclusive. A continuación se muestran algunas herramientas que permiten automatizar en cierta medida el proceso de prueba

QALoad

QALoad de la empresa Compuware (Compuware (NASDAQ: CPWR) es líder mundial en proporcionar software y servicios profesionales que permiten a las empresas gestionar sus negocios y maximizar el valor de sus activos de TI.) es una herramienta de automatización de pruebas de carga para web, Java, .NET, aplicaciones ERP (Planificación de recursos empresariales) y CRM (gestión de relaciones con los clientes y ambientes distribuidos). Simula miles de usuarios desempeñando transacciones de negocio clave de una aplicación para asegurar su desempeño y escalabilidad previa a su puesta en producción. Con esta herramienta, los equipos de prueba pueden rápidamente detectar problemas, optimizar el desempeño de los sistemas y ayudar a asegurar un despliegue de aplicaciones exitoso.

LoadRunner

LoadRunner es una herramienta para realizar pruebas de carga de Mercury Interactive (compañía especializada en Business Technology Optimization, española que recientemente ha sido comprada por HP) que permite pre-ver el comportamiento y el rendimiento del sistema. Además, permite poner a prueba toda la infraestructura corporativa para identificar y aislar los posibles problemas mediante la simulación de la actividad de miles de usuarios. Realiza pruebas en toda la infraestructura corporativa, que comprende las soluciones e-business, ERP, CRM y las aplicaciones personalizadas, simulando la actividad de miles de usuarios, con lo que los equipos de desarrollo de aplicaciones y sitios Web pueden mejorar el rendimiento de las aplicaciones.

Es la herramienta de pruebas de carga más escalable que permite simular la actividad de miles de usuarios con los mínimos recursos de hardware. Se integra a la perfección con las herramientas de gestión del rendimiento de Mercury Interactive. Los mismos scripts creados durante las pruebas pueden volverse a utilizar para monitorizar la aplicación una vez terminada su implantación. Presenta

una interfaz API abierta, con la que los usuarios y otros fabricantes pueden integrar LoadRunner en sus propios entornos.

JMeter

JMeter es una herramienta Java desarrollada dentro del proyecto Jakarta la cual permite realizar pruebas de rendimiento y pruebas funcionales sobre aplicaciones web clásicas, pero también permite realizar test de FTP, JDBC, JNDI, LDAP, SOAP/XML-RPC, y WebServices (en Beta). Permite la ejecución de pruebas distribuidas entre distintos ordenadores, para realizar pruebas de rendimiento; activar o desactivar una parte del test, lo que es muy útil cuando se está desarrollando un test largo, y se desea deshabilitar ciertas partes iniciales que sean muy pesadas o largas. Además, esta herramienta tiene la forma de generar un caso de prueba a través de una navegación de usuario. Se necesita tener una Maquina Virtual Java 1.3 o superior instalada en el sistema operativo Windows XP Professional en este caso.

1.4.1 Comparación entre herramientas

Para la selección de una herramienta para carga y Stress, que no solo permita llevar a cabo la prueba satisfactoriamente, sino que sea la más eficiente y adecuada para este proceso, se hizo necesario llevar a cabo una comparación entre las herramientas que fueron investigadas y analizadas anteriormente.

Para la elección de la herramienta candidata se analizaron los siguientes aspectos:

Funcionalidades: Se refiere a las prestaciones de la herramienta, los elementos funcionales, y la respuesta a los requerimientos del sistema.

Entorno en el que fue desarrollado: Si es un software a código abierto desarrollado en el ambiente de software libre o si es un software del cual no se puede obtener información de su código o no se puede modificar y adecuar a los ambientes de trabajo del proyecto por ser un software propietario.

Modo de Obtención: Si el software se obtiene por medio de una licencia de pago o es gratuito.

Materiales de Apoyo: Si existe documentación de la herramienta que ofrezca una guía para su utilización.

Tabla 1 Comparación entre herramientas.

Herramientas	Funcionalidad	Entorno de Desarrollo	Modo de Obtención	Materiales de Apoyo
QAload	Automatiza pruebas de carga para web, Java, .NET, aplicaciones ERP y CRM. Simula miles de usuarios. Detecta problemas. Optimiza el desempeño de los sistemas. Ayuda a asegurar un despliegue de aplicaciones.	Desarrollada por empresas propietarias fuera del marco del software libre.	Compra de la licencia a la empresa propietaria	Poca existencia de materiales de apoyo y debido a los manuales para el modo de empleo son vendidos en conjunto con la licencia de la herramienta, estos son de difíciles acceso.
LoadRunner	Realiza pruebas de carga. Permite pre-ver el comportamiento y el rendimiento del sistema. Permite la simulación de la actividad de miles de usuarios. Permite la reutilización de los scripts generados para los casos de prueba. Presenta una interfaz API	Desarrollada por empresas propietarias fuera del marco del software libre.	Compra de la licencia a la empresa propietaria	Existe documentación en la Web que se puede

	abierta que permite integrar la herramienta a otros entornos de trabajo.			
JMeter	Permite realizar pruebas de rendimiento y pruebas funcionales sobre aplicaciones web clásicas. Permite la ejecución de pruebas distribuidas entre distintos ordenadores, para realizar pruebas de rendimiento. Permite realizar test de FTP, JDBC, JNDI, LDAP, SOAP/XML-RPC, y WebServices (en Beta). Permite realizar pruebas distribuidas en distintos ordenadores que actuarán como clientes simulando varios hilos que harán función de usuarios. Permite controlar todos los parámetros de una request http. Permite la reutilización de los scripts realizados para la prueba	Realizada en el marco del software libre.	No se necesita la utilización de una licencia ya que se permite bajar de manera gratuita el software desde la Web que representa al proyecto Jakarta	Existe una amplia documentación.

Por las funcionalidades que presenta, por ser un software libre y gratis, política que apoya la universidad y que por las características existe un por ciento superior de confianza en la utilización de la misma ya que se puede contar con todo el código que genera a la herramienta, además de que se pueden utilizar muchas de sus funcionalidades para probar aspectos de las pruebas de Disponibilidad y Red se ha decido utilizar para las pruebas Carga y Rendimiento la herramienta JMeter, versión 2.3.

JMeter es una herramienta dinámica que manipula datos de entrada para la ejecución de las pruebas y necesita la comparación de los resultados para la evaluación, recurre al método de caja negra. La utilización del JMeter supone un 95 % de tiempo menos para la realización de estas pruebas. La mayor inversión de tiempo que se necesita para la realización de las pruebas es la fase de estudio de los casos de uso críticos en la aplicación Web y la elaboración del plan de pruebas en la herramienta. Permite almacenar los resultados de la prueba y se generar gráficos que representan los aspectos que se han probado. El Apache *JMeter* está diseñado para desarrollar diferentes tipos de test; permitiendo diseñar tanto sencillos teses que soliciten simples páginas web, como complejas secuencias de requisiciones que permitan evaluar el comportamiento de una aplicación o como la capacidad de carga máxima que pueda tener una aplicación en un servidor (pudiendo llegar a satura el servidor).

Conclusiones del Capítulo

En este capítulo se ha realizado la descripción del proceso de pruebas, los principios y estrategias de pruebas a seguir en el desarrollo de un software, los niveles, procedimientos y técnicas que se utilizan para el diseño y aplicación de los casos de pruebas, entre otros aspectos de merecida importancia, con el objetivo de ubicar al lector en el campo que se enmarca el estudio de este trabajo.

Se determina así que para lograr la calidad y confiabilidad necesaria en un producto software y para que los clientes queden satisfechos, es necesario que los productos estén libres de errores, y esto trae consigo que los desarrolladores tomen conciencia de la influencia de este proceso en los resultados

finales de los productos y la importancia del mismo para lograr competir en el mercado ya sea a nivel nacional o internacional.

CAPÍTULO 2 DISEÑO, APLICACIÓN Y PROPUESTA DE PRUEBAS DE LIBERACIÓN DE SOFTWARE.

Introducción

Cuando se desarrolla un software, una de las actividades asociadas a este proceso es la prueba; de hecho, se ha establecido formalmente que éstas son fundamentales dentro de cada una de las etapas del proceso de desarrollo de un sistema. La principal razón es que a partir de ella se puede asegurar el cumplimiento de criterios mínimos de operatividad y garantizar la calidad de los productos implementados.

En el presente capítulo se expone el diseño, la aplicación y las propuestas de las pruebas de liberación al modulo Exámenes Clínicos del software Sistema de Gestión de la Información del Proyecto Mapeo Cerebral Humano Cubano que se está desarrollando para el Centro de Neurociencias, especificando cada uno de los elementos de las pruebas definidas. En el mismo se describirá el sistema a probar en cuanto a Casos de Usos y Requisitos Funcionales; se precisará un Plan de pruebas como base para todo el proceso de pruebas que se llevará a cabo, así como la estrategia de todas las pruebas a realizar, incluyendo técnicas, métodos y herramientas. Se especificara la estrategia de Trabajo llevado a cabo durante todo el proceso y los Diseños de Casos de Pruebas.

2.1 Descripción Del Sistema.

2.1.2 Requisitos funcionales.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. El Sistema para la Gestión de los Datos Clínicos cuenta con un total de 44 requisitos funcionales.

1. **Gestionar Cuestionario de Manualidad.**
   1.1. Insertar datos del Cuestionario de Manualidad.

1.1.1. Obtener automáticamente la manualidad.

1.2. Buscar y visualizar datos del Cuestionario de Manualidad.

1.3. Actualizar datos del Cuestionario de Manualidad.

1.4. Eliminar un Cuestionario de Manualidad.

2. Gestionar Examen Físico-Neurológico.

2.1. Insertar datos del Examen Físico-Neurológico.

2.2. Buscar y visualizar datos del Examen Físico-Neurológico.

2.3. Actualizar datos del Examen Físico-Neurológico.

2.4. Eliminar un Examen Físico-Neurológico.

3. Gestionar Examen Psicométrico.

3.1. Insertar datos del Examen Psicométrico.

3.2. Buscar y visualizar datos del Examen Psicométrico.

3.3. Actualizar datos del Examen Psicométrico.

3.4. Eliminar un Examen Psicométrico.

3.5. Visualizar los resultados del Examen Psicométrico.

3.5.1. Graficar automáticamente los resultados.

3.5.2. Imprimir los resultados del Examen Psicométrico.

4. Gestionar Cuestionario de Normalidad.

4.1. Insertar datos del Cuestionario de Normalidad.

4.1.1. Insertar datos personales del cuestionario de Normalidad.

4.1.2. Insertar datos clínicos del cuestionario de Normalidad.

4.2. Buscar y visualizar datos del Cuestionario de Normalidad.

4.2.1. Buscar y visualizar datos personales del cuestionario de Normalidad.

4.2.2. Buscar y visualizar datos clínicos del cuestionario de Normalidad.

4.3. Actualizar datos del Cuestionario de Normalidad.

4.3.1. Actualizar datos personales del cuestionario de Normalidad.

4.3.2. Actualizar datos clínicos del cuestionario de Normalidad.

4.4. Eliminar datos clínicos del Cuestionario de Normalidad.

5. Gestionar Mini Entrevista Neuropsiquiátrica.

5.1. Insertar datos de la Mini Entrevista Neuropsiquiátrica.

5.1.1. Obtener automáticamente los diagnósticos resultantes.

5.2. Buscar y visualizar datos de la Mini Entrevista Neuropsiquiátrica.

5.3. Actualizar datos de la Mini Entrevista Neuropsiquiátrica.

5.4. Eliminar una Mini Entrevista Neuropsiquiátrica.

6. Autenticar Usuario.

6.1. Comparar Usuario y contraseña con los usuarios del sistema.

6.2. Asignar privilegios un usuario.

7. Gestionar sujeto.

7.1. Insertar un sujeto.

7.2. Modificar datos del sujeto.

7.3. Buscar datos y exámenes asociados a un sujeto.

7.4. Visualizar datos del sujeto.

8. Gestionar encuestador.

8.1. Insertar datos del encuestador.

8.2. Buscar y visualizar datos del encuestador.

8.3. Modificar datos del encuestador.

8.4. Eliminar un encuestador.

9. Gestionar usuario.

9.1. Insertar datos del usuario.

9.2. Buscar y visualizar datos del usuario.

9.3. Modificar datos del usuario.

9.4. Eliminar un usuario.

10. Generar reportes automáticamente del examen clínico.

2.1.1 Casos de Uso del Sistema

Según la metodología RUP los casos de uso se derivan del análisis de los requerimientos enriqueciendo requerimientos funcionales a los que se encuentran asociados. El Sistema para la Gestión de los Datos Clínicos está compuesto por 10 casos de usos, todos ellos arquitectónicamente significativos.

Gestionar Cuestionario de Manualidad

El caso de uso Gestionar Cuestionario de Manualidad tiene como propósito llevar a cabo el llenado de un cuestionario de manualidad, insertar los datos, imprimir, eliminar, buscar y modificar.

Definido como caso de uso crítico del sistema.

El caso de uso se inicia cuando el técnico va a realizar alguna de las siguientes operaciones:

- Insertar datos en el cuestionario de manualidad.
- Buscar cuestionario de manualidad.

El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada.

Gestionar Examen Físico-Neurológico

El caso de uso Gestionar Examen Físico-Neurológico tiene como propósito llevar a cabo el llenado de un Examen Físico-Neurológico, insertar los datos, eliminarlo, buscarlo y actualizarlo.

Definido como caso de uso crítico del sistema.

El caso de uso se inicia cuando el Técnico va a realizar alguna de las siguientes operaciones:

- Insertar datos en el Examen Neurológico.
- Buscar Examen Físico-Neurológico.

El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada.

Gestionar Examen Psicométrico

El caso de uso Gestionar Examen Psicométrico tiene como propósito llevar a cabo el llenado del examen psicométrico, insertar los datos, imprimirlo, eliminarlo, buscarlo y actualizarlo.

Definido como caso de uso crítico del sistema.

El caso de uso se inicia cuando el técnico va a realizar alguna de las siguientes operaciones:

- Insertar datos del examen psicométrico.
- Buscar examen psicométrico.

El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada

Gestionar Cuestionario de Normalidad

El caso de uso Gestionar Cuestionario de Normalidad tiene como propósito llevar a cabo la inserción de los datos personales y clínico de un examen de Escala de Normalidad, buscarlos, actualizarlos y eliminar los datos clínicos**.**

Definido como caso de uso crítico del sistema.

El caso de uso se inicia cuando el técnico va a realizar alguna de las siguientes operaciones:

- Insertar datos personales en la Escala de Normalidad
- Insertar datos clínicos en la Escala de Normalidad
- Buscar datos personales de la Escala de Normalidad
- Buscar datos clínicos de la Escala de Normalidad

El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada

Gestionar Mini Entrevista Neuropsiquiátrica

El caso de uso Gestionar Mini Entrevista Neuropsiquiátrica tiene como propósito llevar a cabo el llenado de una MINI Entrevista Neuropsiquiátrica, insertar los datos, eliminarla, buscarla y actualizarla**.**

Definido como caso de uso crítico del sistema.

El caso de uso se inicia cuando el técnico va a realizar alguna de las siguientes operaciones:

- Insertar datos en la MINI Entrevista Neuropsiquiátrica.
- Buscar MINI Entrevista Neuropsiquiátrica.

El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada.

Autenticar Usuario

El caso de uso Autenticar Usuario Autenticar Usuario tiene como propósito entrar al sistema para realizar cierta operación.

Definido como caso de uso crítico del sistema.

El Caso de Uso se inicia cuando el usuario introduce los datos que se le piden para acceder a la aplicación, estos se verifican y finaliza dándole los permisos según los roles y habilitándole la entrada. El caso de uso termina cuando el usuario entra al sistema.

Gestionar Sujeto

El caso de uso Gestionar Sujeto tiene como propósito llevar a cabo una serie de operaciones con los sujetos presentes en el estudio como son: buscar un sujeto, insertarlo o modificarlo.

Definido como caso de uso crítico del sistema.

El caso de uso se inicia cuando el técnico va a realizar alguna de las siguientes operaciones:

- Insertar Sujeto.
- Buscar Sujeto.

El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada.

Gestionar Encuestador

El caso de uso Gestionar Encuestador tiene como propósito llevar a cabo una serie de operaciones con los encuestadores presentes en el estudio como son: insertar un encuestador o modificarlo.

Definido como caso de uso crítico del sistema.

El caso de uso se inicia cuando el administrador va a realizar alguna de las siguientes operaciones:

- Insertar Encuestador.
- Modificar Encuestador

El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada.

Gestionar Usuario

El caso de uso Gestionar Usuario tiene como propósito llevar a cabo una serie de operaciones con los usuarios presentes en el estudio como son: insertar un usuario, visualizar los ya existentes, modificarlo o eliminarlo. Definido como caso de uso crítico del sistema.

El caso de uso se inicia cuando el administrador va a realizar alguna de las siguientes operaciones:

- Insertar Usuario

El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada.

Generar reportes automáticamente del examen clínico

El caso de uso Generar reportes automáticamente del examen clínico tiene como propósito mostrar todos los reportes acerca de las acciones y resultados alcanzados hasta el momento en el estudio.

Definido como caso de uso crítico del sistema.

El caso de uso se inicia cuando el técnico va a realizar alguna de las siguientes operaciones:

- Generar Reporte.

El sistema le muestra la interfaz correspondiente según su solicitud con todos los reportes

2.2 Procedimiento de Pruebas

El proceso de pruebas que se está llevando a cabo no pertenece al ciclo de vida de un proyecto, tampoco lo llevan a cabo miembros propios del equipo de desarrollo, sino miembros externos cuyo interés es validar y verificar la calidad del Módulo exámenes Clínicos del producto Sistema de Gestión de la Información del Proyecto MCHC. Para llevar a cabo este objetivo la metodología usada se basó en 4 etapas principales:

- Planificación de las pruebas: que se va a hacer, cuando, como y quien lo hará, que necesidades es necesario satisfacer para llevarlo a cabo.
- Diseño de las pruebas: que pruebas se van a hacer y en qué consiste cada una (condiciones, pasos, datos, resultados esperados). Especialmente en el caso de pruebas automáticas, es necesario materializar el diseño en determinar dos artefactos (programas, scripts de herramientas de ejecución automática).
- Ejecución de las Pruebas: realización de las pruebas propiamente dichas. Todo lo anterior es la preparación para esta fase.
- Evaluación de los Resultados: análisis de los resultados obtenidos de cara a proporcionar un diagnóstico del sistema bajo prueba y del propio proceso de prueba.

El proceso de ejecución de Pruebas debe ser considerado durante todo el ciclo de vida de un proyecto, para así obtener un producto de alta calidad. Su éxito dependerá del seguimiento de una Estrategia de Prueba adecuada. La estrategia proporciona un mapa que describe los pasos que hay que llevar a cabo como parte de la prueba, cuando se deben planificar y realizar esos pasos, y cuanto esfuerzo, tiempo y recursos se van a requerir. Teniendo en cuenta las etapas definidas anteriormente se siguió la estrategia mostrada en la fig. 2.

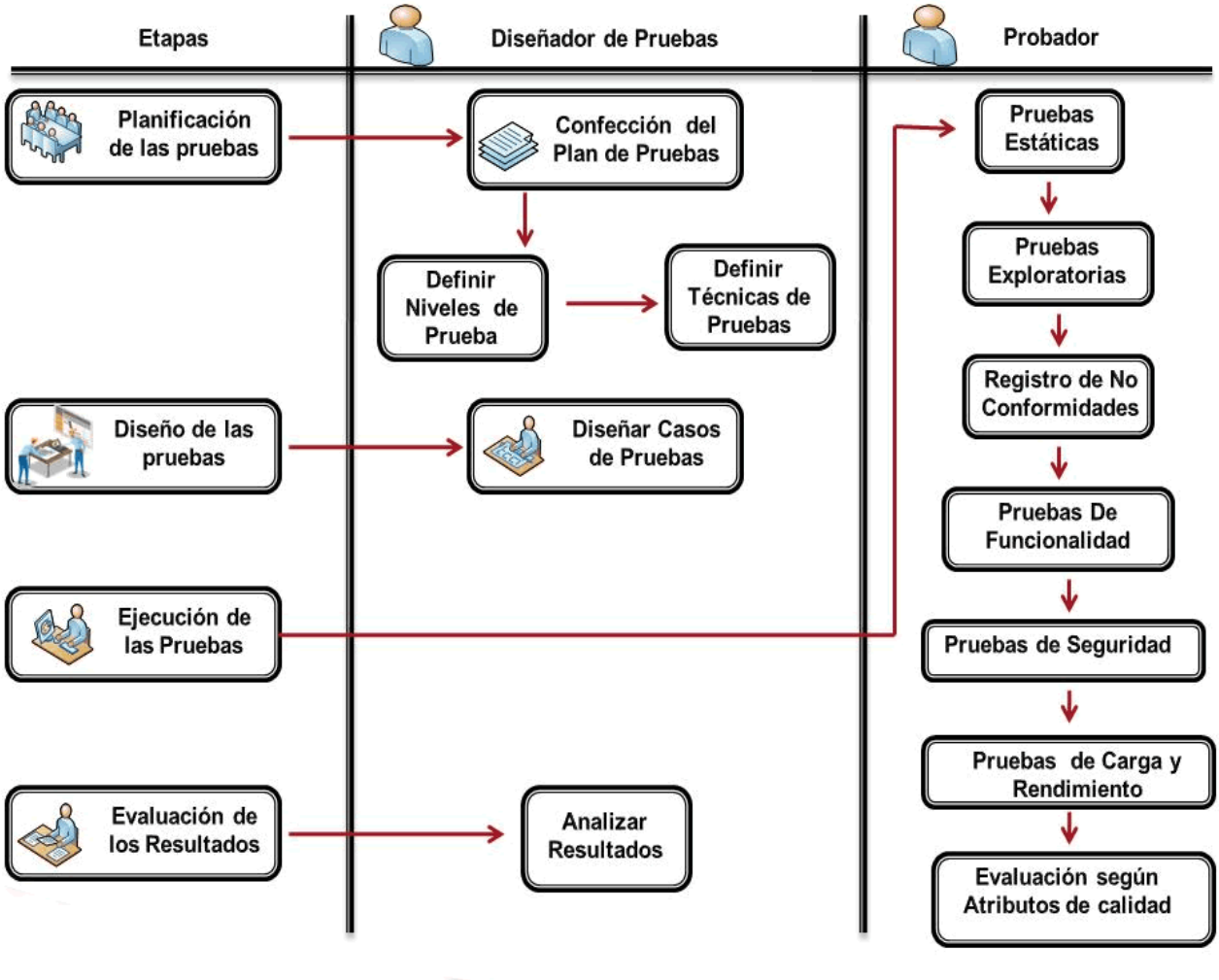

Fig.2 Estrategia de Prueba

2.2.1 Planificación de las Pruebas

Una de las actividades de mayor importancia dentro de todo proceso es la planeación ya que define el marco de trabajo sobre el cual se efectuará el proceso, sin embargo otro de los aspectos por los que la planeación resulta importante es la base histórica que genera a partir de la cual se establecen referentes para los futuros proyectos.

Durante esta etapa se llevó a cabo la confección del Plan de Pruebas, donde se identificaron los elementos que serán probados, los recursos necesarios para hacer las pruebas, así como la estrategia de pruebas que se llevaría a cabo para lograr un buen diseño de casos de pruebas que permitan encontrar la mayor cantidad posible de defectos al software en cuestión. Se describieron además, las pruebas de sistemas que se aplicarán al software desarrollado, el enfoque, y las herramientas a utilizar en el caso de automatizarlas. Se precisa también el equipo de pruebas, compuesto por dos probadores, que serán los encargados de planificar, diseñar y evaluar las pruebas. Se incluyen las especificaciones de Software y Hardware, teniendo en cuenta las herramientas y los dispositivos que se necesitan para el desarrollo de la aplicación, detallando las versiones utilizadas de cada uno de ellos y los proveedores de los mismos.

Otra actividad que se llevó a cabo durante esta etapa del proceso de pruebas fue la confección del cronograma de actividades, que recoge por fechas y esfuerzo las acciones a realizar.

Aspectos que quedaron reflejados en el Plan de Pruebas.

- Introducción
- Organización del Equipo de Pruebas
- Arquitectura Técnica
- Especificaciones del Software y Hardware
- Descripción del Plan de Pruebas
- Estrategia de Pruebas
- Recursos Requeridos
- Plan de Proyecto
- Calendario y Plazos
- Definición de los Entregables
- Seguimiento y Reporte de Defectos
- Aprobación del Plan
- Documentación de los Resultados

2.2.2 Diseño de Pruebas

Se diseñan pruebas que tengan la mayor probabilidad de encontrar el mayor número de no conformidades con la mínima cantidad de esfuerzo y tiempo, para lo cual existen fundamentalmente dos enfoques, que al combinarlos permite lograr mayor efectividad.

En el Diseño de las Pruebas se crean los pasos que deben realizarse para obtener los casos de prueba o scripts de prueba, si se utiliza una herramienta de automatización de pruebas. Se realiza la estrategia de pruebas y otros artefactos que brindan información necesaria para la ejecución de las pruebas. El diseño de las pruebas es una parte considerable de trabajo. Por lo tanto, durante esta tarea, se han de confeccionar los distintos casos de prueba (ver anexo 2) según la técnica o técnicas identificadas previamente. La generación de cada caso de prueba debe ir acompañada del resultado que ha de producir el software al ejecutar dicho caso.

El diseño de las pruebas persigue la prevención de defectos, antes de que por codificación puedan haberse llegado a producir. Un buen diseño de pruebas previene, en consecuencia, los defectos antes de que lleguen a etapas más tardías, con el consecuente ahorro que supondrá su corrección.

Durante esta fase se diseñaron e implementaron los casos de prueba. Para la elaboración de los casos de pruebas y su posterior aplicación se debe tener en cuenta que un buen caso de prueba es aquel que tiene altísima probabilidad de mostrar un error que hasta entonces no se había descubierto y que por tanto los desarrolladores no podían corregirlo.

Los casos de prueba representan los datos que se utilizarán como entrada para ejecutar el software a probar. Más concretamente los casos de prueba determinan un conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular. Cada técnica de pruebas proporciona unos criterios distintos para generar estos casos o datos de prueba.

Para la elaboración de estos se hizo uso de la Plantilla de Diseños de Casos de Prueba (ver Anexo 2) en la cual se reflejaron la descripción general de cada caso de uso a probar y las condiciones de ejecución. En una primera tabla se registraron las distintas secciones a probar en el caso de uso, los escenarios, una breve descripción de cómo debe funcionar y los pasos a desarrollar para probar la funcionalidad que se indicó.

Nombre de la sección: Se especifica el nombre de la sección [SC 1: Nombre de la sección].
Escenarios de la sección: Se especifica(n) el(los) escenario(s) de la sección EC 1.1: Nombre del Escenario.
Descripción de la funcionalidad: Breve descripción de la funcionalidad.
Flujo Central: Pasos a desarrollar para probar la Funcionalidad que se indicó.

A continuación por cada sección reflejada se enumeraron, en una segunda tabla, todos los campos descritos, denominados también variables, se registraron sus nombres, se clasificaron según el componente de diseño utilizado, se especificó si el campo podía ser nulo o no y se describieron brevemente los datos que deben introducirse.

No: Se enumera todos los campos ó variable, descrito en el Caso de Uso.
Nombre de campo: Nombre de campo de entrada.
Clasificación: Se especifica la clasificación según el componente de diseño utilizado [ejemplo: campo de texto, lista desplegable o combo box, entre otros].
Puede ser nulo: Se especifica si el campo puede ser nulo o no, para ello solo se pone Sí o No.
Descripción: Una breve descripción de los datos que deben introducirse.

Esto dio lugar a que en cada sección del caso de uso, se tuviera conocimiento de los valores de entrada que se debían insertar en cada variable asociada a los escenarios descritos, según el juego de dato ideado para probar el sistema. En esta parte de la plantilla de diseño de casos de prueba se llenaron los siguientes campos para cada sección del caso de uso:

Id del escenario: Se especifica el id del escenario [EC1, EC2,…,ECn]
Escenario: Nombre del escenario.
Variables [1, 2,…, n]: Se especifica el nombre de la variable y en su celda correspondiente se indica el valor del dato [V (Válido), I (Inválido), N/A (No Aplica)].
Respuesta del sistema: Se escribe el resultado que se espera al realizar la prueba.
Resultado de la prueba: Se escribe el resultado que se obtiene al realizar la prueba.

2.2.3 Ejecución de las pruebas.

Durante esta etapa, los responsables de la evaluación prepararon las condiciones del ambiente y los datos que se usarán para ejecutar las pruebas hasta obtener un ambiente de pruebas controlado sobre el cual se ejecutan los requerimientos de prueba.

En el momento de la ejecución, y dependiendo de la planificación del proyecto, se seleccionaron y ejecutaron las pruebas pertinentes. Distintas técnicas de prueba ejercitan diferentes criterios como guía para realizar las pruebas.

Tabla 2 Técnicas de Prueba al proyecto MCHC.

Tipos de Pruebas	Objetivo	Técnicas que aplican
Estáticas	Detectar errores e inconsistencias en la documentación.	Listas de Chequeos
Exploratorias	Detectar errores e inconsistencias en la interfaz	No se basa en ninguna técnica.
Funcionales	Asegurar la funcionalidad apropiada del objeto de prueba.	Partición de Equivalencia
Usabilidad	Determinar si la organización de los contenidos y las funcionalidades que se ofrecen desde la aplicación son entendidas y utilizadas por los usuarios de manera simple y directa.	Listas de Chequeos
Portabilidad	Verificar la ejecución del software en distintas plataformas	Listas de Chequeos
Seguridad	Verificar que los mecanismos de protección incorporados en el sistema realmente lo protegerán de accesos impropios.	Listas de Chequeos
Confiabilidad	Evaluar la capacidad que tiene el software para evitar fallas.	Listas de Chequeos
Carga	Probar diferentes cargas para evaluar	Herramienta JMeter

	la capacidad de cómputo.	
Resistencia(stress)	Verificar cómo se comporta el sistema bajo condiciones anormales.	Herramienta JMeter

2.2.3.1 Pruebas Estáticas. Revisión a la documentación.

La importancia de las técnicas estáticas de evaluación a la hora de controlar el nivel de calidad con el que se está llevando a cabo el desarrollo es crucial, ayudan a predecir la fiabilidad del software que se está entregando. Para evaluar las especificaciones de los documentos a revisar, se tuvieron en cuenta los aspectos de la tabla que aparecen en la lista de chequeo, los cuales permitieron encontrar los puntos ineficientes que tenían los elementos chequeados.

Estos son los siguientes:

- Peso: Define si el indicador a evaluar es crítico o no.
- Evaluación (Eval): Es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal y 0 en caso que elemento revisado no presente errores.
- Cantidad de elementos afectados: Especifica la cantidad de errores encontrados sobre el mismo indicador.
- Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.
- Estructura del Documento: Abarca todos los aspectos definidos por el expediente de proyecto o el formato establecido por el proyecto.
- Elementos definidos por la metodología: Abarca todos los indicadores a evaluar según la metodología.
- Semántica del documento: Contempla todos los indicadores a evaluar respecto a la ortografía, redacción y demás.
- N.P. (No Procede): Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.

Se llevó a cabo una revisión bastante estricta que cumpliera con el siguiente objetivo: la obtención de una documentación del sistema con el menor número de errores.

Los documentos que fueron revisados son los siguientes:

- Manual de Usuario
- Diccionario de Datos
- Modelo de Casos de Uso del sistema
- Documento de Roles y Permisos
- Especificación de requisitos

Con esta prueba se busca asegurar que los documentos relacionados al software que se generen en el proceso sean correctos, consistentes y entendibles:

- Correcto, esto es, que cumpla con el formato y organización para el documento establecido en el proyecto.
- Consistente, esto es, que el contenido del documento sea fiel a lo que hace referencia. Si el documento es Documentación de Usuario, que la explicación de un procedimiento sea exactamente como se realiza el procedimiento en el software, si se muestran pantallas que sean las correctas.
- Entendible, esto es, que al leer el documento se entienda correctamente lo que expresa y sin ambigüedades, además que sea fácil de leer. Se incluyen como documentos los Materiales para Soporte al Usuario, Documentación Técnica, Ayuda en Línea y todo tipo de documento que forme parte del paquete de software.

2.2.3.2 Pruebas Exploratorias.

Pruebas realizadas al software para detectar errores e inconsistencias en la interfaz, así como intentar comprender cómo funciona el programa, producir condiciones de error, obtener información de los desarrolladores acerca componentes, deducir interacción entre componentes y problemas web. [7] Es una prueba informal del software que no está basada en ningún plan o caja de prueba y a menudo los probadores aprenden del programa al explorar todas las aplicaciones posibles.

Las pruebas exploratorias muestran sus resultados y garantías en base a los resultados y garantías de prácticas previas y del mismo modo, fallan sus resultados y garantías cuando prácticas que le anteceden son ejecutadas en forma endeble o imprecisa.

Para llevar a cabo estas pruebas cada probador tomó dos papeles, de Usuarios con privilegios y de Usuarios sin privilegios, con el objetivo de verificar que cada usuario realiza las acciones a las que están destinados. Se verificó que el Usuario con privilegios pueda Gestionar Usuarios y Encuestadores, además de Gestionar Sujeto y Ver Reportes, acciones que también realizan los Usuarios sin Privilegios.

Para el éxito de esta verificación se utilizaron las definiciones que se hacen a nivel de requerimiento para verificar mediante pruebas exploratorias, que el sistema esté haciendo lo que se solicita y que no haga lo que no debe hacer.

2.2.3.3 Registro de No Conformidades

Aun cuando los procedimientos y controles buscan asegurar que los procesos permanezcan bajo control, es inevitable que a veces ocurran errores, los cuales son definidos como cualquier desajuste con respecto a los procedimientos operativos aprobados, que pueden afectar la calidad del producto o servicio. Con el propósito de manejar tales anomalías, todos los errores encontrados se registran en la Plantilla de No Conformidades (ver anexo 3)

No Conformidad: Una no conformidad es la falta de cumplimiento de especificaciones establecidas, o la producción bajo un procedimiento no aprobado o con algún error.

Tabla 3 No conformidades

No Conformidades	Descripción
Significativa	Documentación: Afectan o influyen de manera negativa en el sentido de la idea Aplicación: Afectan la función del producto
No Significativa	Documentación: No afectan o influyen de manera negativa en el sentido de la idea

	Aplicación: No afectan la función del producto

Los errores, defectos, fallos detectados durante las distintas pruebas son registrados como no conformidades en la Tabla de No Conformidades presentada en el anexo 1 donde:

Elemento: Se especifica si es documentación o aplicación

No.: Identificador del error (un número).

No conformidad: Se especifica la no conformidad que fue encontrada o sea el error detectado.

Aspecto correspondiente: Se especifica la localización.

Etapa de la detección: En que etapa de la revisión se detectó el error.

Clasificación (Significativa): Se pone una (X), en caso de que la no conformidad esté clasificada como significativa.

Clasificación (No Significativa): Se pone una (X), en caso de que la no conformidad esté clasificada como no significativa.

Clasificación (Recomendación): Se pone una (X), en caso de que la no conformidad solo sea una recomendación.

Estado NC: Se coloca el estado de la no conformidad y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó [RA: Resuelta, PD: Pendiente, NP: No Procede].

Respuesta del Equipo de Desarrollo: Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué.

2.2.3.4 Pruebas de Funcionalidad.

El objetivo que persiguen las pruebas funcionales es asegurar la funcionalidad apropiada del objeto de prueba, incluyendo la navegación, entrada de datos, proceso y recuperación. (Ver cap. 1)

Se ejecuta cada caso de uso, flujo de caso de uso, o función usando datos válidos y no válidos, para verificar lo siguiente:

- Se obtienen los resultados esperados cuando se usan datos válidos.

- Cuando se usan datos no válidos se despliegan los mensajes de error o advertencia apropiados.
- Se aplica apropiadamente cada regla del negocio.

Según este tipo de proceso de verificación y validación, no se pretende que las pruebas funcionales indiquen que el sistema funciona correctamente, sino entender que el concepto es la detección de fallos y errores.

Métodos de prueba Caja Negra: Partición de equivalencia.

Un aspecto importante a medir cuando se llevan a cabo este tipo de pruebas es que se cumplan a cabalidad los requerimientos funcionales que fueron previstos en la etapa de análisis más específicamente en el flujo de trabajo de Requerimientos, tanto los requeridos por los clientes como los necesarios para el funcionamiento correcto de la aplicación, estos requerimientos se listan a continuación:

- Autenticarse al sistema.
- Buscar sujeto.
- Insertar sujeto
- Buscar Encuestador
- Insertar Encuestador
- Buscar usuario.
- Insertar usuario
- Gestionar Cuestionario de Manualidad.
- Gestionar Examen Físico-Neurológico.
- Gestionar Examen Psicométrico.
- Gestionar Cuestionario de Normalidad.
- Gestionar Mini Entrevista Neuropsiquiátrica.
- Generar reportes automáticamente del examen clínico.

Para llevar a cabo esta prueba, se hicieron uso de los diseños de casos de pruebas por cada caso de uso del sistema ya previamente conformado y se ejecutaron los escenarios por secciones descritos en estos, dándole valores válidos y no validos a las variables ya enumeradas. Mientras se llevó a cabo la

ejecución de los casos de pruebas, se registró si la respuesta del sistema es o no la respuesta esperada. En caso de que no lo sea, en la tabla de registros de no conformidades se detalla el tipo de error y la ubicación exacta en donde fue detectado. (Anexo 2)

Elemento: Se especifica el nombre del elemento.
No: Se especifica el número de la no conformidad.
No conformidad: Descripción de la no conformidad.
Aspecto correspondiente: Descripción del Aspecto correspondiente.
Etapa de detección: Etapa de detección del error.
Significativa: Se pone una (X), en caso de que la no conformidad esté clasificada como significativa.
No Significativa: Se pone una (X), en caso de que la no conformidad esté clasificada como no significativa.
Recomendación: Se pone una (X), en caso de que la no conformidad solo sea una recomendación.
Estado NC: Se coloca el estado de la no conformidad y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó [RA: Resuelta, PD: Pendiente, NP: No Procede].
Respuesta del Equipo de Desarrollo: Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué.

2.2.3.5 Pruebas de seguridad.

Estas pruebas se hacen con los objetivo a Nivel de Usuario de verificar que un actor solo pueda acceder a las funciones y datos que su usuario tiene permitido y que solo los actores con acceso al sistema y a la aplicación están habilitados para accederla. Se busca garantizar que los usuarios están restringidos a funciones específicas o su acceso está limitado únicamente a los datos que está autorizado a acceder y que solo aquellos usuarios autorizados a acceder al sistema son capaces de ejecutar las funciones del sistema.

Para llevar a cabo esta prueba se tomaron en cuenta los siguientes pasos:

1. Identificar cada tipo de usuario y las funciones y datos a los que se debe autorizar.

2. Crear pruebas para cada tipo de usuario y verificar cada permiso, creando transacciones específicas para cada tipo de usuario.
3. Modificar tipos de usuarios y volver a ejecutar las pruebas.

Tras ejecutar las pruebas se verificó que para cada tipo de usuario conocido, las funciones y datos apropiados y todas las transacciones funcionan como se esperaba.

2.2.3.6 Pruebas de Carga y Stress usando JMeter.

El proceso de la ejecución de las pruebas depende de la forma en que vayan a ser ejecutadas las pruebas. JMeter muestra opciones para realizar pruebas automatizadas de manera planificada con un tiempo de arranque y de parada de las pruebas. Como resultado final se obtiene un listado de tiempos con los cuales se formulan gráficos, si se desea obtener una imagen que exprese el análisis de los resultados. Estas pruebas pueden ejecutarse cuantas veces se quiera y sus resultados son visibles en pocos minutos.

El proceso de automatización de las pruebas constó de 4 etapas fundamentales.

1. Grabación de los escenarios de prueba.

La grabación de los escenarios de prueba se conformó por una grabación independiente, guiándonos por el camino lógico conformado a partir de los casos de uso críticos definidos anteriormente por los desarrolladores.

Camino Lógico

- Autenticar Usuario.
- Gestionar Usuario.
- Gestionar Mini Entrevista Neuropsiquiátrica.
- Gestionar Cuestionario de Normalidad.
- Gestionar Cuestionario de Manualidad.
- Gestionar Examen Físico-Neurológico.
- Gestionar Examen Psicométrico.
- Generar reportes automáticamente del examen clínico.

La grabación fue dirigida a navegar el sitio y recoger todas las peticiones que se generan al realizar acciones entre el usuario y el servidor del sistema de gestión de Exámenes Clínicos.

2. Confección de los test.

La confección de los test de las pruebas de carga y stress para el sistema de gestión de Exámenes Clínicos se realizó utilizando los escenarios de pruebas complejas, debido a que las páginas críticas que se definieron realizan intercambios de datos entre el cliente y el servidor.

Los test que se elaboraron para probar el Sistema de gestión de Exámenes Clínicos tuvieron la siguiente estructura jerárquica en los elementos:

Plan de Pruebas

- Grupo de Hilos
  - Informe Agregado
  - Ver Árbol de Resultados
  - Gráfico de Resultados

3. Ejecución de las pruebas y recolección de los resultados.

Para la ejecución de las pruebas se comprobó el funcionamiento de la aplicación, la disponibilidad del servidor Web y las características necesarias para la ejecución de la herramienta de pruebas. Se ejecutó la prueba tomando como base la cantidad máxima de usuarios que debería soportar el sistema, en este caso 50 usuarios y se llevaron a cabo 2 iteraciones de la simulación.

- Iteración 1: Simulación de 50 usuarios con un periodo de subida de 1 segundo.
- Iteración 2: Simulación de 50 usuarios con un periodo de subida de 30 segundos realizando el test dos veces.

4. Análisis de los resultados.

Este punto será analizado en el capitulo posterior.

2.2.3.7 Evaluación del producto según los atributos de Calidad

En el caso de la calidad del software, el término es difícil de definir, en el modelo de calidad ISO 9126-1 se propone un conjunto independiente de seis altos niveles de características de calidad, que son definidas como un conjunto de atributos de un producto de software para el cual su calidad es descrita y evaluada.

Funcionalidad:

La capacidad del producto de software para proporcionar funciones que reúnan una serie de condiciones y necesidades cuando el software se utiliza bajo determinadas condiciones (lo que el software debe hacer para cumplir las necesidades).

Fiabilidad:

La capacidad del producto de software para mantener su nivel de rendimiento bajo unas condiciones indicadas después de un período determinado de tiempo.

Usabilidad:

La capacidad del producto de software para ser comprendido, aprendido, usado y atractivo para el usuario, cuando se utiliza en condiciones específicas (el esfuerzo necesario para usarlo).

Eficiencia:

La capacidad del producto de software para proveer un rendimiento apropiado, en relación con la cantidad de recursos utilizados, bajo ciertas condiciones.

Mantenibilidad:

La capacidad del producto de software para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptaciones del software a los cambios del entorno y en los requerimientos y especificaciones funcionales (el esfuerzo necesario para ser modificado)

Portabilidad:

La capacidad del producto de software para ser transferido de un entorno a otro. El entorno puede incluir la organización, hardware o software.

Confiabilidad:

Capacidad para continuar el trabajo aunque haya interrupciones.

Para llevar a cabo la evaluación del sistema de gestión de exámenes clínicos el proceso se centró en 5 atributos de calidad, los cuales se caracterizarán a continuación.

Usabilidad

"Usabilidad es la eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico" [8]. Cuando hablamos de Usabilidad nos referimos a "Facilidad de Uso", ya sea de una página web, una aplicación informática o cualquier otro sistema que interactúe con un usuario. La usabilidad permite mayor rapidez en la realización de tareas y reduce las pérdidas de tiempo.

Se persigue asegurar en la aplicación:

- Visibilidad del estado del sistema
- Control y libertad del usuario
- Consistencia y estándares
- Prevención de errores
- Flexibilidad y eficiencia de uso
- Estética y diseño minimalista
- Ayudar a reconocer, diagnosticar y solucionar errores
- Ayuda y documentación

Portabilidad

La portabilidad es uno de los conceptos clave en la programación de alto nivel. Se define como la característica que posee un software para ejecutarse en diferentes plataformas, el código fuente del

software es capaz de reutilizarse en vez de crearse un nuevo código cuando el software pasa de una plataforma a otra. A mayor portabilidad menor es la dependencia del software con respecto a la plataforma.

Se persigue asegurar en la aplicación:

- La capacidad del software para ejecutarse en diferentes plataformas.

Confiabilidad

La confiabilidad de un sistema de cómputo es una propiedad que implica el grado de confianza esperado por parte del usuario en la operación adecuada del sistema al utilizarlo.

Se persigue asegurar en la aplicación:

- Madurez
- Tolerancia a Defectos
- Facilidad de Recuperación.

Seguridad

Sub-característica del atributo Funcionalidad. Con esta prueba se busca verificar que los mecanismos de protección incorporados en el sistema realmente lo protegerán de accesos impropios.

Se persigue asegurar que en la aplicación:

- Los datos del sistema solo pueden ser accesibles por los autores debidamente autorizados.
- Las funciones del sistema solo pueden ser accesibles por los autores debidamente autorizados.
- Las funciones que atenten contra la integridad de los datos de negocios sean debidamente impedidas.

Eficiencia

Con esta característica se persigue asegurar en el sistema:

- Velocidad de procesamiento.
- Tiempo de respuesta.
- Consumo de recursos.

- Rendimiento efectivo total.
- Eficacia.

Una vez efectuadas todas las pruebas pertinentes durante el proceso de liberación del software, se hará un levantamiento y evaluación de los resultados obtenidos, considerando la eficiencia y la efectividad del producto. Para la evaluación del producto se harán uso de Listas de Chequeos (Ver Anexo 4) que respondan a los parámetros de calidad seleccionados y contribuirá a conocer cuál es el valor del producto y cuanta seguridad y confianza brinda a la hora de ejecutar sus procedimientos a través de diferentes atributos de calidad.

Las listas de chequeo enumeran una serie de ítems que deberían verificarse uno a uno para asegurarse de lograr el producto final con un nivel de calidad previamente aceptado. Las respuestas a las preguntas de las Listas de Chequeo se pueden dar de forma directa o mediante la realización de Casos de Prueba. Las respuestas dependerán de los resultados obtenidos en los mismos.

Los **Casos de Prueba** serán evaluados por medio de la siguiente escala:

Las evaluaciones serán:

-- Malo - propiedad no disponible. (0)

- Satisfactorio - propiedad parcialmente disponible. (2)

+ Bien - propiedad disponible. (4)

++ Excelente - propiedad muy bien implementada. (5)

El peso de la pregunta será:

!! Muy importante (5)

! Menos importante (3)

2.2.3 Evaluación de los resultados

Durante esta etapa se analizarán los resultados obtenidos con el propósito de proporcionar un diagnóstico del sistema bajo prueba y del propio proceso de prueba.

Conclusiones del Capítulo

Luego de planificarse y aplicarse posteriormente las pruebas al Sistema de Gestión de la Información del producto MCHC se llegaron a las siguientes conclusiones:

- La planificación de la Estrategia de Prueba puede reducir significativamente el esfuerzo necesario para el desarrollo de las pruebas adecuadas, reducir el tiempo de realización y ejecución de las mismas y disminuir los altos costos que se generan.
- Se lograron aplicar correctamente las técnicas de pruebas siguiendo una estrategia fiable para su aplicación y los pasos establecidos.
- Se desarrollaron los casos de pruebas de calidad utilizando la técnica de caja negra, para validar el diseño de prueba propuesto para el producto.
- Las pruebas se realizaron con un alto nivel de responsabilidad y por ende de calidad a todo el producto comprobando posteriormente los resultados que realmente se obtuvieron con los pronosticados.

CAPÍTULO 3 EVALUACIÓN DE LOS RESULTADOS

Introducción

Luego de la aplicación de las pruebas se procede a la realización de una evaluación de los resultados obtenidos donde se exponen cada uno de los detalles de las pruebas así como los errores que fueron detectados. En este capítulo se exponen de manera clara y entendible un resumen de todos los fallos que se detectaron a la hora de aplicar las pruebas de caja negra, de carga y stress al software Sistema de Gestión de la Información del proyecto Mapeo Cerebral humano Cubano, su evaluación atendiendo a los parámetros de calidad seleccionados, así como las recomendaciones que se hacen al respecto.

3.1 Análisis de los resultados.

A la hora de realizar las pruebas mantener en orden la documentación del software se convierte en un requisito fundamental para su satisfactoria ejecución. Se debe tener claramente los documentos de descripción de casos de usos, del análisis, el diseño, así como el manual de usuario, siendo este un documento importantísimo a ser revisado ya que será la cara del producto ante el cliente. El hecho de que la documentación se encuentre completa da la posibilidad de tener un control estricto de todos los casos de prueba que se van a aplicar.

Los resultados de las pruebas deben ser bien documentados, de manera que los desarrolladores del software tengan una guía oficial por la cual regirse a la hora de corregir los errores de cada una de las partes del sistema a la que corresponda antes de que el proyecto pase a una fase de mayor complejidad, donde la corrección de errores se hace más compleja y costosa.

Por otro lado, el valor que puede proporcionarse a una organización puede ser aumentado o disminuido por la respuesta de la organización a una no conformidad. Asegurando que la organización ha realizado satisfactoriamente la corrección y análisis de las causas y acción correctiva se incrementará la probabilidad de que la organización logre la satisfacción del cliente. Por ello, durante la ejecución de las pruebas, registrar de manera ordenada las no conformidades

detectadas y mantener un control estricto de estas en cuanto si son corregidas o no, se convierte en uno de los pasos fundamentales del proceso de liberación.

3.1.1 Análisis de las No Conformidades detectadas.

Tras llevar a cabo las pruebas estáticas, las exploratorias y las funcionales, durante tres iteraciones, los resultados arrojados se listarán a continuación:

- En la Documentación se detectaron 31 no conformidades, 17 significativas, 13 no significativas y 1 recomendación. De ellas todas fueron corregidas satisfactoriamente por el equipo de desarrollo.
- Durante las pruebas exploratorias se detectaron 8 no conformidades, 7 clasificadas como significativas y 1 no significativa. De ella solo una no procedió por la siguiente causa:

  "La automatización del proceso para determinar de la edad del sujeto sería en seudocódigo el siguiente:

  Capturar el año de la fecha de nacimiento y restársela al año de la fecha actual. La fecha actual se obtiene mediante una función del lenguaje que no hace más que devolverte la fecha que tiene el sistema. ¿Pero quién me asegura que la PC donde se encuentra la aplicación tiene la fecha correcta? De no tener el sistema la fecha correctamente configurada, el proceso de resta anteriormente mencionado devuelve un valor diferente a la edad que posee el sujeto. Por lo que dejamos a responsabilidad del usuario que inserte manualmente la fecha del sujeto encuestado. "
- Durante las pruebas funcionales se detectaron 2 no conformidades significativas las cuales fueron resueltas satisfactoriamente.

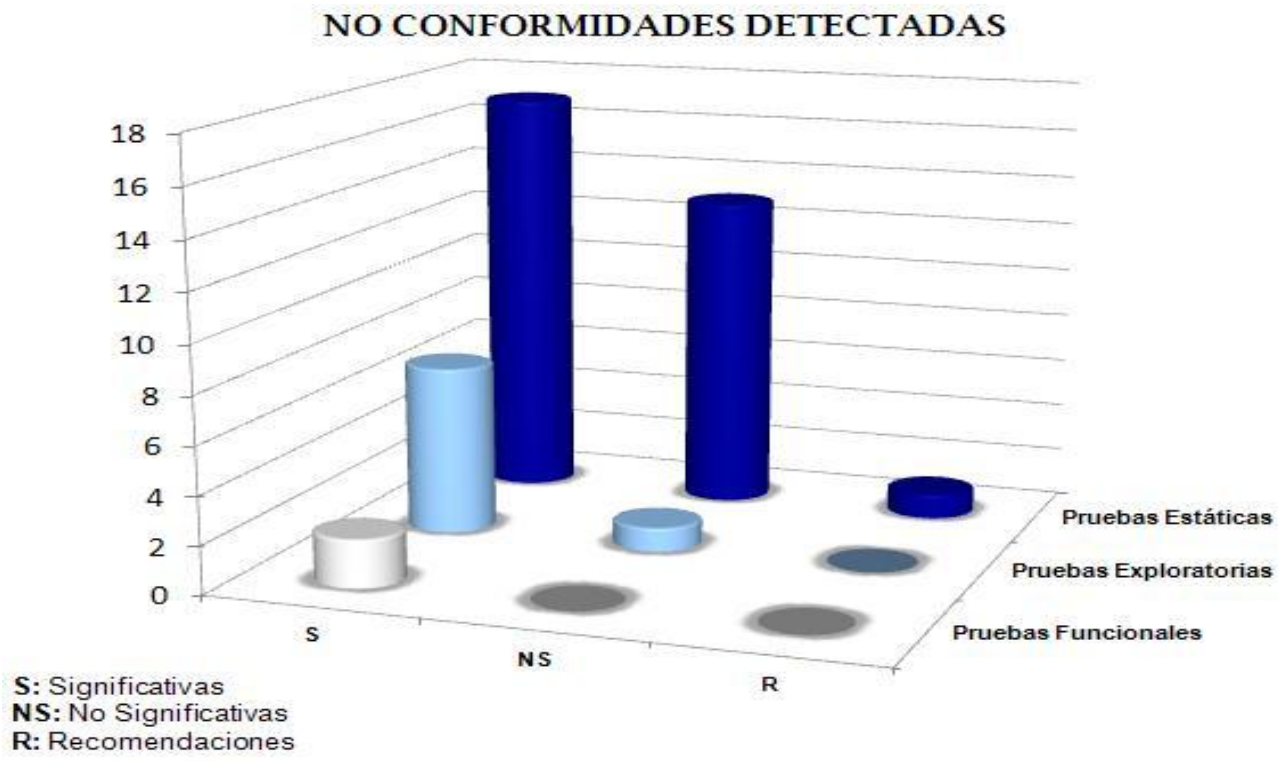

Fig.3 No Conformidades detectadas.

3.2 Evaluación del Producto

A pesar del gran número de artículos de investigación y normas existentes sobre el tema de validación de calidad del producto de software, existen hoy en día muy pocas Industrias de Software que utilicen procesos de evaluación y análisis para este efecto. Las especificaciones de la calidad de un producto de software han sido objeto de trabajo de varios grupos.

Como se reflejó en el capítulo anterior para la evaluación del Sistema de gestión de Exámenes Clínicos se centró en 5 atributos de calidad: Usabilidad, Confiabilidad, Portabilidad, Seguridad y Eficiencia apoyándonos en el uso de Listas de Chequeos. A continuación se reflejan en las siguientes tablas las evaluaciones dadas a las distintas preguntas pertenecientes a los atributos antes mencionado.

Tabla 4 Resultados Usabilidad

Preguntas	Eval.
Visibilidad del estado del sistema	
Los enlaces del menú cambian de color cuando se seleccionan o se han visitado	--
Adecuación entre el sistema y el mundo real	
El lenguaje que se emplea se adecua a los usuarios destinados a navegar el sistema.	++
La información que se presenta en la aplicación es fácil de entender y memorizar	+
La página refleja la identidad de la empresa (logos, compañía…)	++
Los mensajes de error están en texto plano, entendible.	++
El nombre de los botones de un formulario es adecuado, aplicado a la acción, no general (Ej.: Utilizar “Enviar” en vez de “OK”...)	++
Control y libertad del usuario	
Tras una acción relevante hay una opción de vuelta atrás	+
Si una acción tiene consecuencias, el sistema proporciona información y pide confirmación antes de continuar	+
Hay un acceso a la página de inicio en una zona visible y reconocible	++
Prevención de errores	
Actúa el sistema como corrector de errores	+
Actúa el sistema en la prevención de errores	++
Actúa el sistema en la información de los errores	++
El mensaje de error permite volver a la situación anterior	+
Se dan indicaciones para completar campos problemáticos	++
En situaciones donde se pueden producir errores de escritura existe la posibilidad de seleccionar la información de una lista	+
Se ofrecerán valores por defecto en los campos en caso de que tengan que estar completados y el usuario desconozca como completarlo	-
Consistencia y estándares	
No hay enlaces rotos o que no lleven a ningún sitio	++
Se mantiene una navegación consistente y coherente en todas las pantallas	++
La distribución de los elementos estructurales se mantiene constante a lo largo de la aplicación	++
El nombre de los enlaces es el mismo que el título de la página a la que dirige	++

En los formularios el texto está alineado a la derecha y los campos a la izquierda	+
Se utiliza el mismo tono en toda la web	++
Se utilizan colores estándar para los enlaces	++
Ayuda y documentación	
Se ofrece alguna especie da ayuda	--
El sitio está diseñado para necesitar el mínimo de ayuda y de instrucciones	+

Los resultados obtenidos arrojaron 4 puntos de 5 posibles lo que apunta a que el software es un 80 % usable. La calidad es crítica pero no suficiente para que un producto sea exitoso, dado que éste no posee un valor intrínseco: su valor radica en cómo se lo usa, lo que implica la existencia de usuarios. Por lo tanto, la manera en que las personas lo van a utilizar constituye una cuestión básica para los diseñadores y desarrolladores que deseen certificar la calidad del producto. Si el resultado de esta no es del todo satisfactorio, significa que tales personas puedan tener problemas para llevar a cabo sus tareas.

Tabla 5 Resultados Portabilidad

Preguntas	Eval.
Se adapta el software a diferentes ambientes sin necesidad de usar otros medios que los previstos	++
El sitio funciona sobre cualquier navegador habitual (Explorer, firefox, opera, netscape…)	-
Se accede al sistema desde la plataforma GNU/Linux o Microsoft Windows.	++

Los resultados obtenidos arrojaron 4 puntos de 5 posibles lo que apunta a que el software es un 80 % portable.

Tabla 6 Resultados Confiabilidad

Preguntas	Eval.
Se recupera el software ante fallas	++
Se previene el sistema ante fallos	-

Es tolerable el sistema ante fallos	+
Utiliza técnicas de tolerancia de fallo	-
Cuenta con una buena base de información a la hora de realizar cálculos.	++

Dándole evaluaciones que corresponden a las preguntas que indican la capacidad del software de recuperarse ante posibles fallos, la lista de chequeo arrojó 3.6 puntos obtenidos de 5 puntos posibles. Con este resultado se valora que el producto es un 72 % confiable.

Tabla 7 Resultados Seguridad

Preguntas	Eval.
Las reglas de protección de Archivos de datos, las autoridades y códigos de identificación de usuario fueron establecidas por el dueño del sistema o asignado por una autoridad más alta	+
El Acceso al control de protección está incorporado en el sistema	+
Toda la privacidad, la libertad de información, sensibilidad, y consideraciones de la clasificación fueron identificadas, resueltas, y establecidas	+
Están instrumentados autos chequeos que permitan la protección contra virus	-
Están instrumentados controles que permitan realizar la auditoría informática	++
Para interactuar con la aplicación existen diferentes sesiones con permisos respectivos.	++

Los resultados obtenidos arrojaron 4 puntos obtenidos de 5 puntos posibles, esto implica que el software sea un 80 % seguro. Uno de los principales problemas detectados es la no existencia de autos chequeos que permitan la protección contra virus, ya que para esta protección se basa solamente en el antivirus que tenga instalada la computadora en que este se encuentre el sistema. Aunque no existe el concepto de software totalmente seguro y posiblemente eso no existirá jamás, porque es imposible predecir los razonamientos que un atacante experto podrá ingeniar en el futuro para romper un sistema, está en las manos del equipo calidad asegurar que este sea lo más seguro posible.

Tabla 8: Resultados Eficiencia.

Evaluación	Eval.
Las respuestas del sistema se realizan en el tiempo requerido para la aplicación	+
Las tareas repetitivas se efectúan con facilidad	++
Se evitan los pasos inútiles	++
Las partes o secciones más importantes del sitios son accesibles desde la página de inicio	++
El cursor se desplaza adecuadamente en un formulario al presionar “tabulador”	++
Existe el completamiento de información introducida anteriormente	++

Los resultados arrojaron 4,3 puntos de 5 puntos posibles, lo que implica que el software sea un 86% eficiente.

3.3 Análisis de los resultados de las pruebas de carga y stress.

La cantidad de peticiones que generó la grabación de la navegación del Sistema de Gestión de Exámenes Clínicos fue de 50 peticiones mostrando los resultados que aparecen en la tabla 13. Analizando los resultados generales de las pruebas al Sistema de gestión de Exámenes clínicos se obtiene un total de: 303 peticiones para un total de 50 usuarios mostrando los datos que se muestran a continuación.

Iteración 1: Simulación de 50 usuarios con un periodo de subida de 1 segundo.

A continuación se muestran los artefactos generados al ejecutar las pruebas en JMeter, el resumen de resultados arrojados por el Informe Agregado (Tabla 13), el Árbol de Resultados y Gráficos de Resultados (Ver anexo 5) los cuales esbozan los resultados obtenidos al simular las pruebas para 50 usuarios con un tiempo de subida de 1 segundo.

Tabla 9 Resumen de Resultados de las pruebas de carga y estrés del Sistema

	Muestras	Media	Mediana	Línea 90%	Min	Max	%Error	Rendimiento	Kb/sec
Total	303	1253	187	2469	0	21203	0.05	61.7/min	4650.3

- Muestras: Cantidad de páginas (Hilos) que simulan la cantidad de usuarios que están interactuando con el sistema desde la misma URL.

- Media: Media de tiempo total que demoraron las petición en cargarse.
- Mediana: Tiempo promedio que han tardado en cargarse las páginas.
- Min: Tiempo mínimo que ha demorado en cargarse una página.
- Max: Tiempo Máximo que ha tardado en cargarse una página.
- Línea 90 %: Tiempo máximo en que corrieron el 90 por ciento de las peticiones reales, o sea, el tiempo más probable que se puede demorar una petición.
- %Error: Por ciento de error de las páginas que no se llegaron a cargar de manera satisfactoria.
- Kb/Seg: Velocidad de carga de las páginas.

El sistema muestra un soporte de concurrencia de 303 peticiones simultáneas, cargándose satisfactoriamente con una media de 1253 milisegundos (21 seg.), pudiéndose gestionar 62 peticiones para 50 usuarios respondidas en 187 milisegundos (3 seg.).

Iteración 2: Simulación de 100 usuarios con un periodo de subida de 10 segundos.

A continuación se muestran los artefactos generados al ejecutar las pruebas en JMeter, el resumen de resultados arrojados por el Informe Agregado (Tabla 13), el Árbol de Resultados y Gráficos de Resultados (Ver anexo 5) los cuales esbozan los resultados obtenidos al simular las pruebas para 100 usuarios con un tiempo de subida de 10 segundo.

Tabla 10 Resultados Generales

	Muestras	Media	Mediana	Línea 90%	Min	Max	%Error	Rendimiento	Kb/sec
Total	269	1952	828	3109	0	35313	0.06	52.5	5515

El sistema muestra un soporte de concurrencia de 269 peticiones simultáneas, cargándose satisfactoriamente con una media de 1952 milisegundos (33 seg.), pudiéndose gestionar 53 peticiones para 100 usuarios respondidas en 828 milisegundos (14 seg.).

3.4 Análisis de los Resultados Generales del Sistema.

La documentación correspondiente al sistema estaba conformada únicamente por errores ortográficos y gramaticales, todos ellos fueron debidamente corregidos, por lo que los distintos documentos están libres de errores de este tipo.

Los errores detectados en la interfaz previeron problemas futuros en las distintas funcionalidades que podrían traer como consecuencia una incorrecta ejecución del programa o un mal funcionamiento del mismo. Actualmente el software se encuentra libre de estas dificultades.

La figura 3 muestra los resultados de presencia de las Características de Calidad del Sistema de Gestión de Exámenes Clínicos.

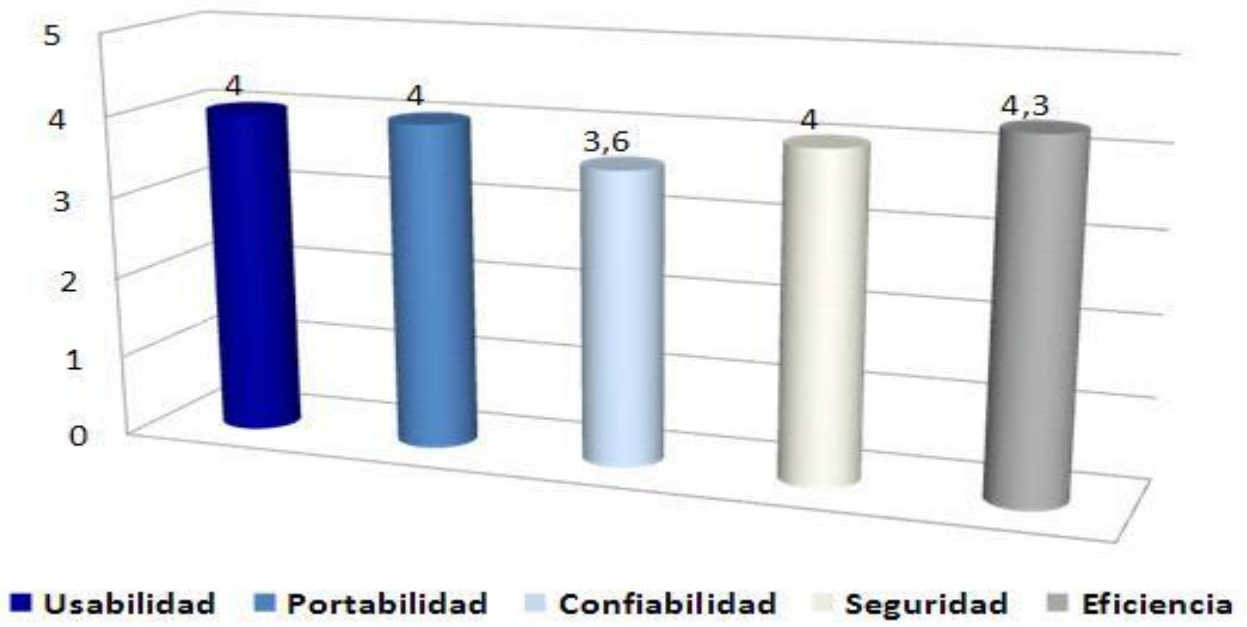

Fig.4 Presencia de las Características de Calidad en el Módulo Exámenes Clínicos.

Los resultados reflejan que el producto es portable, atendiendo a los requisitos no funcionales se probó que el sistema se adapta fácilmente a diferentes ambientes sin necesidad de usar otros medios que los previstos, accediendo al sistema a través de navegadores desde la plataforma GNU/Linux o Microsoft Windows. Por otro lado lo resultados arrojaron que el producto es casi, en su totalidad seguro, pero no es parcialmente usable.

El hecho de que la usabilidad arroje este resultado implica que en el sistema la realización de tareas se haga con cierta lentitud y que aumente las pérdidas de tiempo. Este permite a los usuarios un acceso fácil y rápido y que cualquier persona que posea conocimientos básicos en el manejo de una computadora y de un ambiente Web pueda usarlo, sin embargo, podría mejorarse en este aspecto y crear métodos que hagan mucho más usable el software, para que las personas puedan emplearlo fácil y rápidamente para llevar a cabo sus tareas. Se considera que un producto es "fácil de aprender y de usar" en términos del tiempo que les insume –al utilizarlo– hacer lo que se desea, de la cantidad de pasos que deben efectuar y del éxito que tienen en predecir la acción correcta a seguir.

Se sugiere al equipo desarrollador:

- Que los enlaces del menú cambien de color cuando se seleccionan o se visiten.
- Proveer al sistema de una Ayuda para que el usuario cuente con un apoyo a la hora de navegar por este.

Por otro lado, se comprobó que para la cantidad de usuarios que debe permitir el sistema conectarse simultáneamente y hacer uso de las funcionalidades, el software no solo acepta las peticiones, sino que responde en un tiempo razonablemente rápido, tomando en cuenta la escasez de recursos de gran potencia, no solo para llevar a cabo las pruebas, sino para que el software se ejecute sin mucho imprevistos, siendo estos inconvenientes para el desarrollo del software que no se pueden ignorar. Además, teniendo en cuenta que el sistema fue probado en condiciones inferiores a las reales en las que se debe desplegar, se concluye que el resultado fue satisfactorio dado que el software cumple con lo pactado con el cliente.

Conclusiones del Capítulo

Al finalizar este capítulo, se puede concluir con que el proceso de pruebas se llevó a cabo de forma satisfactoria, pues los casos de pruebas desarrollados mostraron muchos de los errores del sistema.

Por otro lado se evaluó el software en cuanto a Usabilidad, Seguridad, Confiabilidad y Portabilidad, haciendo uso de listas de chequeos. Se analizó el rendimiento del mismo y se obtuvieron resultados favorables. Un adecuado rendimiento y comportamiento son cruciales para que la aplicación sea aceptada por el cliente.

Se da así por concluido el proceso de prueba por parte del grupo de calidad de forma clara y entendible. Se documentaron todos los defectos lo cual sirve de gran ayuda para el equipo de desarrollo en posteriores versiones.

CONCLUSIONES

La búsqueda de altos niveles de calidad es buscar excelencia y solo se logra cuando la estrategia es óptima, en el caso de las pruebas de software es esencial aplicarlas adecuadamente porque los productos requieren comprobaciones específicas en cada caso y fase de desarrollo.

Al planificarse y aplicarse las pruebas al PMCHC, se llegó a las siguientes conclusiones:

- Se llevó a cabo la revisión de la documentación perteneciente al proyecto.
- Se diseñaron casos de pruebas para cada caso de uso del sistema.
- Se empleó el método de Caja Negra mediante la técnica de Partición de Equivalencia.
- Se registraron en la plantilla de No Conformidades los resultados obtenidos de las pruebas a la aplicación y la documentación.
- Se realizaron diferentes técnicas de pruebas con un alto nivel de seriedad teniendo en cuenta la importancia que reviste el software para la medicina cubana.
- Se realizaron pruebas de Carga y Stress al sistema para medir su rendimiento.
- Se aplicaron las pruebas de atributos de Calidad para obtener una evaluación del software.

Con el estudio realizado y la aplicación de las pruebas que se han llevado a cabo en el trabajo “Liberación del Sistema de Gestión de la Información del Proyecto Mapeo Cerebral Humano Cubano: Módulo Examen Clínico”, se ha cumplido el objetivo propuesto al inicio del trabajo, el cual consistió en desarrollar un proceso de pruebas de liberación de software al PMCHC, con vistas a descubrir el mayor número de defectos existentes en el mismo, logrando con su ulterior corrección obtener una aplicación con un mínimo de errores y por consiguiente mayor calidad.

RECOMENDACIONES

Luego de haber concluido con todo el proceso de prueba y de comprender de cuán importante es la detección de errores a tiempo, se recomienda:

- Automatizar pruebas funcionales para otros procesos de pruebas, con el propósito de obtener resultados más fiables y que reflejen un mejor comportamiento del producto a probar.
- Incluir dentro del proceso de pruebas de liberación, las pruebas de Carga y Estrés automatizadas con la herramienta JMeter, dado que con las pruebas funcionales no se asegura el rendimiento y desempeño del sistema.
- Garantizar por parte del grupo de los aseguradores de calidad del grupo de desarrollo los Diseños de Casos de Pruebas.

REFERENCIAS BIBLIOGRÁFICAS

1. **Pressman, Roger S.** *Ingeniería del software, un enfoque práctico.* s.l. : McGrawHill, quinta edición, 2002.

2. ISO 9000:2000.

3. **9126, ISO/IEC Pressman, Roger S.** *Ingeniería del software, un enfoque práctico.* s.l. : McGrawHill, quinta edición, 2002.

4. **Barrios, Johanna Rojas - Emilio.** www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas - Grupo ARQUISOFT. [En línea] 2007.

5. **Prado, Elena Raja.** *Actas de Talleres de Ingeniería del Software y Bases de Datos.* 2007. 2006.

6. **LLopis, Miguel.** http://geeks.ms/blogs/mllopis/archive/2008/02/09/191-en-qu-233-consiste-el-proceso-de-pruebas-de-software.aspx . *Serie ECDB: ¿En qué consiste el proceso de pruebas de software?* [En línea] 2008.

7. www.kinetia.es/calidad. [En línea]

8. www.ainda.info/index.html. [En línea] 2008.

BIBLIOGRAFÍA

1. **Vliet., H. Van.** *Software Engineering, Principles and Practice,.* 2001.

2. **Sánchez Almenares, Liudmila.** *Prueba Automática de Carga y Estrés en el Proyecto CICPC.* 2008.

3. **Pressman, Roger S.** *Ingenieria de Software. Un enfoque práctico.*

4. **conocimiento, Consultoría de áreas de.** *Apache JMeter. Manual de usuario v1.1.doc.*

5. **Bolívar, Universidad Simón.** www.ldc.usb.ve/~teruel/ci4713/clases2001/planPruebas.html. [En línea] 2003.

6. **Barrios, Grupo ARQUISOFT - Johanna Rojas - Emilio.** www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas . [En línea] 2007 .

7. www.kinetia.es/calidad. [En línea]

8. ISO 9000:2000.

9. cbasqa.wordpress.com. [En línea]

10. lsi.ugr.es/~arroyo/inndoc/cicloVidaSoft.html. [En línea]

11. **Noa, Eliane Paz.** *Diseño y aplicación de pruebas al producto "Árbol Genealógico" .* 2007.

12. **Natalia Juristo, Ana M. Moreno, Sira Vegas.** *TÉCNICAS DE EVALUACIÓN DE SOFTWARE .* 2006.

13. **Ortiz., Isabel Blank. Larissa Herrera. Miguel.** *Tipos de prueba de Caja Blanca y Negra Pruebas Funcionales – Software II – .* 2005.

14. **Marte, Keyly Betancourt Rodríguez Frank Gabriel Rodríguez.** *Propuesta de pruebas de calidad para producto LIMS Control de Calidad del CIGB ll .* 2007.

ANEXOS

Anexo 1: Plantilla Plan de Pruebas

Plan de Pruebas

1. Introducción

1.1 Alcance

[Proyectos con los que se involucra el Plan]

1.2 Definiciones, Acrónimos y Abreviaturas

1.3 Referencias

[Lista de documentos a los que se hace referencia en el Plan]

Código	*Título*
[1]	Documento 1
[2]	Documento 2
[3]	Modelo de Diseño - Módulo de Administración v0.0

2. Organización del Equipo de Pruebas

[Descripción del equipo de probadores, por quienes está compuesto, responsabilidad de cada miembro.
Diseñador de prueba: Planifica, diseña y evalúa los resultados de la prueba.
Ingeniero de componentes: Es el responsable de la elaboración de los componentes de prueba para los procedimientos que puedan ser automatizados.
Probadores: Realizan las pruebas unitarias, de integración y sistema.]

3. Arquitectura técnica.

[Descripción mediante diagramas (Despliegue, componentes) las partes que componen el sistema bajo prueba. Incluye el almacenamiento de datos y las conexiones para su transferencia y describe el objetivo de cada componente, inclusive la forma de su actualización. Se debe documentar tanto las capas, como la presentación / interfaz del usuario, la base de datos, los emisores de informes, etc. Un diagrama de alto nivel que muestre como el sistema en prueba se inserta en un contexto de automatización mayor también puede ser agregado, si el mismo está disponible.]

4. Especificaciones del Software y Hardware

[Corresponde a una lista individualizada de todo el hardware y el software que utiliza la aplicación, incluyendo proveedores y versiones.]

5. Descripción del Plan de Pruebas

5.1 Descripción de los requerimientos

[Esta sección del Plan de Pruebas contiene una lista de todos los requerimientos. Cualquier requerimiento no incluido en esta lista estará fuera del alcance de las pruebas. Esto libera de responsabilidad en caso de que existan problemas con la funcionalidad del sistema que se relaciona con un requisito no incluido en este listado.]

5.1.1 Requerimientos Funcionales

[Todas las funciones que deben ser probadas, como por ejemplo la creación, la corrección y supresión de registros, son puestas en esta lista. Puede incluirse la lista completa en esta sección o bien hacerse referencia a otro documento que contenga la información.]

5.1.2 Casos de Uso: <Caso de Uso1>

[Aquí se debe indicar brevemente en qué consiste el CU, y en las secciones posteriores incluir la información necesaria para verificar su funcionamiento]

Escenarios

[Aquí se plantea la matriz de escenarios correspondiente al Caso de Uso 1, esto es todas las posibles combinaciones del flujo normal de los eventos y los flujos alternativos, caminos que se derivan del flujo de eventos del Caso de Uso>]

Nombre de escenario	Flujo inicial	Flujo alternativo	Flujo alternativo	……

Plantilla de condiciones

[Aquí se plantea la planilla de Condiciones, esto es las condiciones que causan que se ejecute un escenario específico dentro de los posibles escenarios identificados para el Caso de Uso1, indicando el resultado esperado de ejercitar ese Escenario-Condición y la referencia a los casos de prueba de la planilla CasosPruebaConDatos.xls para ese CU que se corresponden con ese Escenario-Condición]

Escenario	Descripción	Condiciones o Elementos <Las condiciones o elementos requeridos para ejecutar los distintos escenarios>			Resultado Esperado
		Condición 1	*Condición 2*	*…*	
Esc 1					
…					
Esc 2					

Diagrama de Entidad Relación

Entidades que intervienen en el caso de uso en cada uno de los estados por los que transitan.

CASOS DE PRUEBA

Aquí se plantean las pruebas para ese ciclo, dado que una prueba encadena un escenario de cada caso de uso, se debe dar la referencia al Escenario del Caso de Uso, la referencia a los datos que prueban ese escenario y las entidades involucradas en cada prueba y sus estados, para el caso de Prueba del ciclo debe darse el resultado esperado

Caso del Ciclo	Datos de la Prueba				Resultado Esperado
1	**Caso de Uso**	**Escenario-Condición**	**Caso de Prueba**	**Entidades/Estados**	
2	**Caso de Uso**	**Escenario-Condición**	**Caso de Prueba**	**Entidades/Estados**	

1.1.1. Requerimientos de Diseño

Las pruebas de la interfaz de usuario, las estructuras de menú u otros elementos de diseño también deberían ser puestas en una lista o referenciados hacia otro documento.

1.1.2. Requerimientos de Integración

Los requerimientos para probar el flujo de datos desde un componente a otro deben ser incluidos si ellos harán parte del Plan de Pruebas.

1.1.3. Otros Requerimientos

Cualesquiera otras exigencias que tenga la aplicación y que necesiten ser probadas.

Estrategia de Prueba

Use esta sección para describir como los objetivos de la prueba serán alcanzados para cada uno de los tipos de pruebas que hacen parte del plan, ejemplo:

- Unitarias
- De integración
- De sistema
- Validación y Verificación
- De estrés
- De configuración y/o de instalación

Nota: Se puede agregar alguna otra prueba que se haga.
Para cada subconjunto requerido o definido como necesario, debe detallarse lo siguiente:

1.2. Objetivo

El objetivo global de esta estrategia debe alcanzarse. Por ejemplo, para una prueba de sistema, este objetivo puede ser una declaración de que todos los requerimientos funcionales deben comportarse de acuerdo a lo esperado, o como quedó documentado.

1.3. Técnica

Especifica como los casos de prueba serán desarrollados, el instrumento o herramienta usado para almacenarlos y donde pueden ser encontrados; como ellos serán ejecutados y los datos que serán usados. Declare aquí si las pruebas deben ser realizadas en ciclos, o de común acuerdo con los otros esfuerzos de pruebas.

1.4. Entorno de Prueba

Especificar las condiciones de hardware y configuración bajo las cuales se deben realizar las pruebas.

1.5. Proceso

Breve descripción del proceso que se realiza.

1.6. Casos de Prueba

Hacer una lista detallada o una referencia a los casos reales de prueba que serán utilizados para poner en práctica el plan.

1.7. Criterios de Término

Registrar los criterios que serán usados para determinar la aprobación o rechazo de pruebas y la acción que debe ser tomada con base en los resultados de la prueba.

1.8. Herramientas

Documentar los instrumentos o herramientas que serán empleados para las pruebas. Citar al proveedor, la versión y el número de la Mesa de Ayuda para pedir el apoyo, si fuera necesario.

2. Recursos Requeridos

Identificar los roles y las responsabilidades que serán requeridas para la ejecución del Plan de Pruebas.

3. Plan de proyecto

Parte del cronograma del proyecto que abarca la etapa de pruebas.

4. Calendario y Plazos.

Documentar el plazo en el cual la aplicación a probar estará disponible para pruebas y el tiempo estimado para ejecutar los casos de prueba. Especifique si se proporcionará partes construidas, sobre una base regular durante el ciclo de prueba, o cuando se espera que los componentes del sistema estén listos para pruebas.

5. Definición de los Entregables.

Ponga en una lista cualquier entregable asociado con el esfuerzo de pruebas y donde las copias de estos entregables o documentos pueden ser localizados. Esto incluye el Plan de Pruebas en sí mismo, escenarios para prueba, casos de prueba y el plan de proyecto.

6. Seguimiento y Reporte de Defectos.

Documente el instrumento y el proceso usado para registrar y rastrear los defectos. Ponga en una lista todos los informes que serán generados incluyendo repositorios, frecuencias, mecanismos de entrega y ejemplos. Identifique los recursos involucrados en el proceso de seguimiento.
Describa cualesquiera calificación, categoría o clasificación que se usará para identificar o priorizar defectos. Las siguientes son categorías, de ejemplo, para priorizar o calificar defectos:

- **Crítico**: Denota una función inutilizable que causa un término anormal o una falla general, o cuando un cambio en un área de la aplicación causa un problema en otra parte.
- **Severo**: Una función no actúa como fue requerido o diseñado, o un objeto de interfaz no trabaja como se muestra.
- **Advertencia**: La función trabaja, pero no tan rápidamente como esperado, o no se ajusta a las normas y convenciones.
- **Cosmético**: No crítico para el funcionamiento de sistema: palabras con mala

ortografía, formateo incorrecto, mensajes de error vagos o confusos o advertencias.

7. Aprobación del Plan.

El Plan de Pruebas debe ser revisado por todos las partes responsables de su ejecución y aprobado por el equipo de prueba, el jefe del proyecto y el gerente de desarrollo.
Obtenga las firmas de aprobación en todas las páginas del mismo.
Una reunión final de verificación con todas las partes involucradas, es comprobadamente el método más eficaz para obtener la aprobación del Plan de Pruebas.

8. Documentación de los Resultados.

Cuando el esfuerzo de prueba esté terminado, documente los resultados y mediciones. Identifique cualquier discrepancia entre el plan y la puesta en práctica real y describa adecuadamente como aquellas discrepancias fueron manejadas.

Anexo 2: Plantilla de Diseño de Casos de Pruebas

Descripción General

[Descripción general del CU]

Condiciones de Ejecución:

[Precondiciones del CU].

1. Secciones a probar en el Caso de Uso:

[Para cada sección los escenarios van a ser flujo básico + los alternativos].

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
[SC 1: Nombre de la sección]	EC 1.1: Nombre del Escenario.	Descripción de la Funcionalidad.	Pasos a desarrollar para probar la Funcionalidad que se indicó.
	EC 1.2: Nombre del Escenario.	Descripción de la Funcionalidad.	Pasos a desarrollar para probar la Funcionalidad que se indicó.
	EC 1.n: Nombre del Escenario.	Descripción de la Funcionalidad.	Pasos a desarrollar para probar la Funcionalidad que se indicó.
[SC 2: Nombre de la sección]	EC 2.1: Nombre del Escenario.	Descripción de la Funcionalidad.	Pasos a desarrollar para probar la Funcionalidad que se indicó.
	EC 2.2: Nombre del Escenario.	Descripción de la Funcionalidad.	Pasos a desarrollar para probar la Funcionalidad que se indicó.

1.1. Descripción de variable.

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción
[1]<Se enumera todos los campos ó variable, descrito en el Caso de Uso >	[Nombre de campo de entrada]	[la clasificación es según el componente de diseño utilizado][ejemplo: campo de texto, lista desplegable o combo box]	[Se especifica si el campo puede ser nulo o no]para ello solo se pone Sí o No	Una breve descripción de los datos que deben introducirse.
[2]				
[3]				

SC 1.2: <Sección #1 a revisar>

Id del escenario	Escenario	Variable 1 *(Nombre de la variable)*	Variable 2 *(Nombre de la variable)*	Variable N *(Nombre de la variable)*	Respuesta del Sistema	Resultado de la Prueba
EC 1	Nombre del escenario.	V	V	V	Se escribe el resultado que se espera al realizar la prueba.	Se escribe el resultado que se obtiene al realizar la prueba.
		V	V	V		
		V	V	V		
EC 2	Nombre del escenario.	NA	NA	NA	El sistema emite un mensaje para que llene los campos obligatorios	Se escribe el resultado que se obtiene al realizar la prueba.

EC n	Nombre del escenario.	I	V	V	El sistema muestra un mensaje informativo.	Se escribe el resultado que se obtiene al realizar la prueba.
		V	I	V		
		V	V	I		

[Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.]

1. Registro de defectos y dificultades detectados.

Elemento	No	No conformidad	Aspecto correspondient	Etapa de detección	S	NS.	Rcmd.	Estado NC	Respuesta del Equipo Desarrollo
<Nombre del Elemento>	< 1>	<Descripción de la No Conformidad>	<Descripción de Aspecto correspondiente>	<Etapa de detección del error>	<X>	<X>	<X>	[Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó.] RA: Resuelta PD: Pendiente NP: No Procede	[Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué.]

S: significativa
NS: No significativa
Rcmd.: Recomendaciones

Anexo 3: Plantilla No Conformidades

1. Aspectos Generales

[Descripción de Aspectos Generales a tener en cuenta a la hora de analizar el resultado de las pruebas, incidencias en el momento de su desarrollo y otros aspectos relevantes.]

Elementos Probados.

[Descripción general o lista de los Elementos Probados, y otros aspectos importantes a tener en cuenta a la hora analizar las No Conformidades Detectadas.]

Elementos no Probados y causas.

[Descripción general o lista de los Elementos no Probados, la causa de que no se hayan podido realizar las Pruebas y cualquier otro elemento importante que aporte la información necesaria para que sean analizadas estas causas y resueltas para la siguiente iteración.]

2. Tabla de No Conformidades Detectadas

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Clasificación	Estado No Conformidad	Respuestas equipo de desarrollo
<Nombre del Elemento>	<1>	<Descripción de la No Conformidad>	<Descripción del Aspecto correspondiente>	<Etapa de detección del error>	<S: Significativa NS: No Significativa R: Recomendación	< Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó.] RA: Resuelta PD:Pendiente NP:No Procede>	< Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué. >

Anexo 4: Listas de Chequeos: Atributos de Calidad

LCH: Usabilidad

Nivel	Evaluación	Eval.	NP	Comentario
	Visibilidad del estado del sistema			
	Los enlaces del menú cambian de color cuando se seleccionan o se han visitado.			
	Adecuación entre el sistema y el mundo real			
	El lenguaje que se emplea se adecua a los usuarios destinados a navegar el sistema.			
	La información que se presenta en la aplicación es fácil de entender y memorizar			
	La página refleja la identidad de la empresa (logos, compañía…)			
	Los mensajes de error están en texto plano, entendible.			
	El nombre de los botones de un formulario es adecuado, aplicado a la acción, no general (Ej.: Utilizar “Enviar” en vez de “OK”...)			
	Control y libertad del usuario			
	Tras una acción relevante hay una opción de vuelta atrás			
	Si una acción tiene consecuencias, el sistema proporciona información y pide confirmación antes de continuar			
	Hay un acceso a la página de inicio en una zona visible y reconocible			
	Prevención de errores			
	Actúa el sistema como corrector de errores			
	Actúa el sistema en la prevención de errores			
	Actúa el sistema en la información de los errores			
	El mensaje de error permite volver a la situación anterior			
	Se dan indicaciones para completar campos problemáticos			
	En situaciones donde se pueden producir errores de escritura existe la posibilidad de seleccionar la información de una lista			
	Se ofrecerán valores por defecto en los campos en caso de que tengan que estar completados y el usuario desconozca como completarlo.			
	Consistencia y estándares			

	No hay enlaces rotos o que no lleven a ningún sitio.			
	Se mantiene una navegación consistente y coherente en todas las pantallas.			
	La distribución de los elementos estructurales se mantiene constante a lo largo de la aplicación.			
	El nombre de los enlaces es el mismo que el título de la página a la que dirige.			
	En los formularios el texto está alineado a la derecha y los campos a la izquierda.			
	Se utiliza el mismo tono en toda la web.			
	Se utilizan colores estándar para los enlaces.			
	Ayuda y documentación			
	Se ofrece alguna especie da ayuda.			
	El sitio está diseñado para necesitar el mínimo de ayuda y de instrucciones.			

LCH: Portabilidad

Nivel	Evaluación	Eval.	NP	Comentario
	Se adapta el software a diferentes ambientes sin necesidad de usar otros medios que los previstos			
	El sitio funciona sobre cualquier navegador habitual (Explorer, firefox, opera, netscape…)			
	Se accede al sistema desde la plataforma GNU/Linux o Microsoft Windows.			

LCH: Confiabilidad

Nivel	Evaluación	Eval.	NP	Comentario
	Se recupera el software ante fallas			
	Se previene el sistema ante fallos			
	Es tolerable el sistema ante fallos			
	Se recupera el software ante fallas			
	Utiliza técnicas de tolerancia de fallos			
	Cuenta con una buena base de información a la hora de realizar cálculos.			

LCH Seguridad

Nivel	Evaluación	Eval.	NP	Comentario
	Las reglas de protección de Archivos de datos, las autoridades y códigos de identificación de usuario fueron establecidas por el dueño del sistema o asignado por una autoridad más alta			

	El Acceso al control de protección está incorporado en el sistema			
	Toda la privacidad, la libertad de información, sensibilidad, y consideraciones de la clasificación fueron identificadas, resueltas, y establecidas			
	Están instrumentados autos chequeos que permitan la protección contra virus			
	Están instrumentados controles que permitan realizar la auditoría informática?			
	Para interactuar con la aplicación existen diferentes sesiones con permisos respectivos.			

LCH: Confiabilidad

Nivel	Evaluación	Eval.	NP	Comentario
	Las respuestas del sistema se realizan en el tiempo requerido para la aplicación			
	Las tareas repetitivas se efectúan con facilidad			
	Se evitan los pasos inútiles			
	Las partes o secciones más importantes del sitios son accesibles desde la página de inicio			
	El cursor se desplaza adecuadamente en un formulario al presionar “tabulador”			

LEYENDA:

Nivel: Importancia del aspecto a evaluar

E: Evaluación

NP: No Procede

Comentario: Es obligatorio en las respuestas negativas

Las evaluaciones serán:

-- Malo - propiedad no disponible. (0)

- Satisfactorio - propiedad parcialmente disponible. (2)

+ Bien - propiedad disponible. (4)

++ Excelente - propiedad muy bien implementada. (5)

El peso de la pregunta será:

!! Muy importante (5)

! Menos importante (3)

Anexo 5: Resultados JMeter

Iteración 1: Simulación de 50 usuarios con un periodo de subida de 1 segundos.

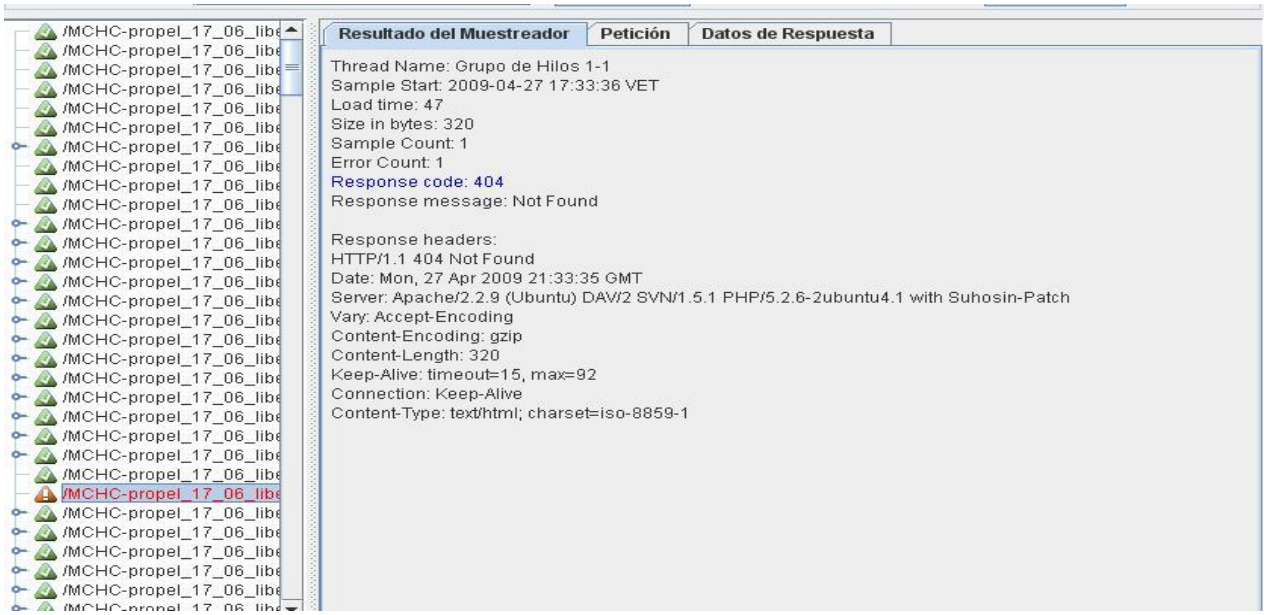

Fig.5 Árbol de Resultados para la simulación de 50 usuarios con tiempo de subida de 1 segundo

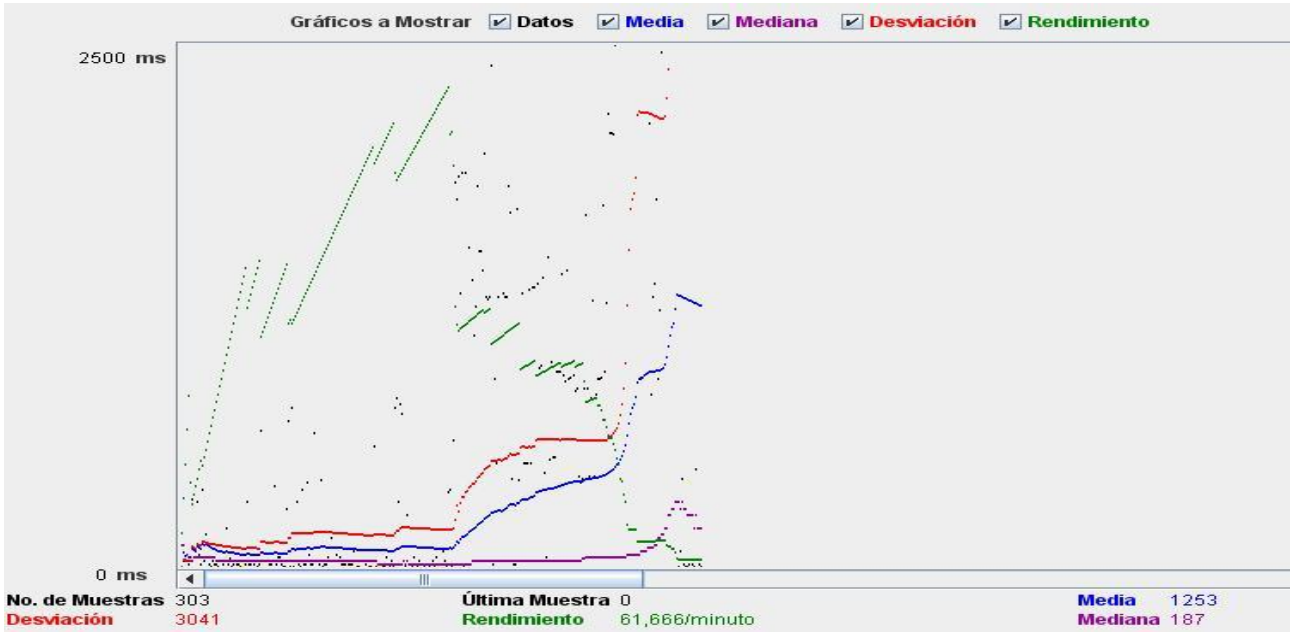

Fig.6 Gráfico de Resultados para la simulación de 50 usuarios con tiempo de subida de 1 segundo

Iteración 2: Simulación de 100 usuarios con un periodo de subida de 10 segundos.

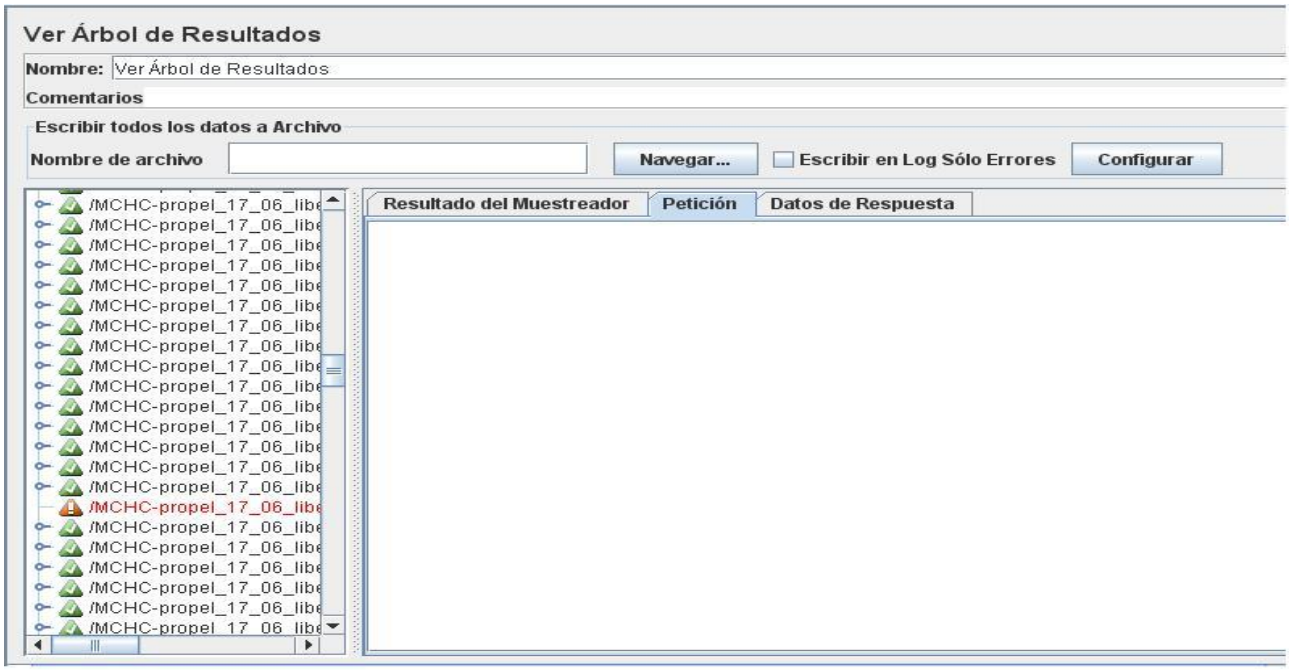

Fig.7 Árbol de resultados para la simulación de 50 usuarios con tiempo de subida de 10 segundos

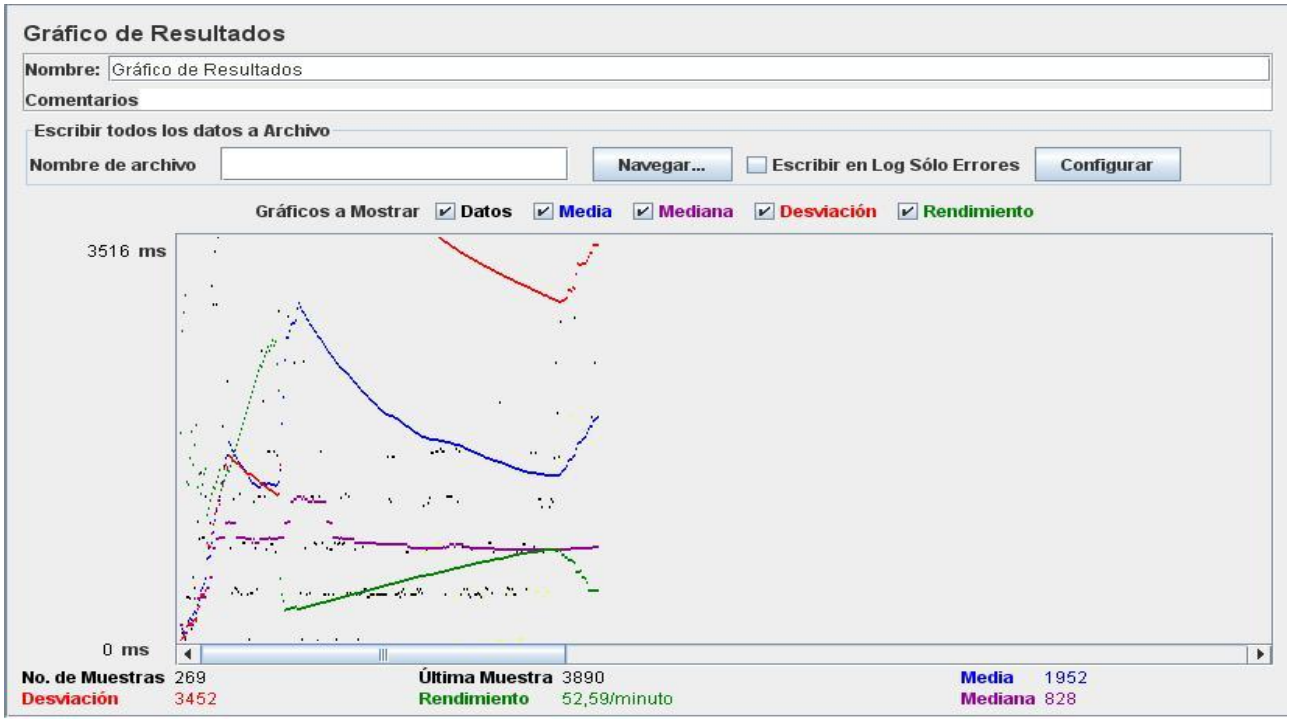

Fig.8 Gráfico de resultados para la simulación de 50 usuarios con tiempo de subida de 10 segundos

GLOSARIO DE TÉRMINOS

Caso de uso: Descripción del comportamiento del sistema en término de secuencia de acciones.

Cliente: Una persona u organización, interna o externa a la organización productora que toma responsabilidad financiera por el sistema. El cliente es el último destinatario del producto desarrollado y sus artefactos.

Defecto: Cualquier requerimiento, elemento de diseño o de implementación que si no es cambiado, causará un diseño, implementación, prueba, uso, o mantenimiento inapropiado del producto.

Fallo: "La incapacidad de un sistema o de alguno de sus componentes para realizar las funciones requeridas dentro de los requisitos de rendimiento especificados".

Funcionalidad: Funcionalidad de software. Conjunto de operaciones que realiza el software.

Interfaz: Una colección de operaciones que se usan para especificar el servicio de una clase o de un componente. Un juego nombrado de operaciones que caracterizan la conducta de un elemento. La Interfaz hombre-máquina es un canal comunicativo entre el usuario y el ordenador.

Metodología: Un sistema de principios y normas generales de organización y estructuración teórico-práctica de actividades.

Prueba: Prueba de software. Ejecución de un sistema bajo condiciones específicas, se observan y se analizan los resultados realizándose una evaluación de los mimos.

Proceso: Secuencia de actividades invocadas para producir un producto de software.

Procedural: Sinónimo para algorítmica.

Requerimiento: Cualquier necesidad de un área usuaria que debe cubrirse mediante una solución de tipo informático.

Rol: Papel, cometido o función que tiene o desempeña que interpreta un actor.

Usuario: Persona que utiliza normalmente el software.

1