Universidad de las Ciencias Informáticas

Facultad 6

Título: "Proceso de Pruebas de Liberación al Sistema de Manejo de Datos de Ensayos Clínicos Cubano"

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autoras:

Onaysi Vasallo Artigas

Yislén Dolores Ramírez Camejo

Tutoras:

Ing. Heydi Menéndez Avalos

Ing. Martha Denia Hernández Ramírez

Ciudad de La Habana, Junio, 2009

"Año del 50 Aniversario del Triunfo de la Revolución"

"La responsabilidad nuestra es luchar porque la calidad del producto que aquí se haga sea de las mejores y la mejor posible…"

Ernesto Che Guevara

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 firmamos la presente a los ___ días del mes de junio del año 2009.

_____ _____

Firma de Onaysi Vasallo (Autora) Firma de Yislén D. Ramírez (Autora)

_____ _____

Firma de Heydi Menéndez (Tutora) Firma de Martha D. Hernández (Tutora)

DATOS DE CONTACTO

Autoras:

Onaysi Vasallo Artigas
Universidad de las Ciencias Informáticas, Habana, Cuba
Email: ovasallo@estudiantes.uci.cu

Yislén Dolores Ramírez Camejo
Universidad de las Ciencias Informáticas, Habana, Cuba
Email: ydramirez@estudiantes.uci.cu

Tutoras:

Ing. Heydi Menéndez Avalos
Universidad de las Ciencias Informáticas, Habana, Cuba
Email: hmenendez@uci.cu

Ing. Martha Denia Hernández Ramírez
Universidad de las Ciencias Informáticas, Habana, Cuba
Email: mdhernandez@uci.cu

AGRADECIMIENTOS

De Onaysi:

Quiero agradecer a todas aquellas personas que con su apoyo incondicional han hecho posible la realización exitosa de este trabajo:

A mis padres por el apoyo, la paciencia y la comprensión infinita.

A mi hermano Otniel por apoyarme siempre y saberme guiar por el buen camino.

A mis abuelos Paula Nereyda y Alejo Rolando por quererme tanto y ser tan atentos y preocupados por mis estudios.

A mi tío Pedro por estar siempre ahí cuando lo he necesitado para apoyarme.

A Carly, como cariñosamente le llamo por brindarme todo su amor, cariño, comprensión y saber guiarme, por aguantarme durante estos cuatro años en momentos malos y buenos. A toda su familia por preocuparse por mí y brindarme todo su apoyo.

A toda mi familia por confiar siempre en mí.

A mis compañeros del 6106 por compartir alegrías, tristezas, y por apoyarme en estos cinco años.

A Yislén y a todos mis amigos y amigas por saberme comprender y brindarme su cariño, su ternura, su amor, iluminando mis días con sus alegrías.

A Martha Denia, Heydi, Martha Nieves, Delvis, Richard por toda la atención y el apoyo brindado en este periodo de tiempo.

A todos los profesores que tuve en estos cinco años, gracias por brindarme cada día, con paciencia y esmero sus conocimientos.

A la Tía Maritza y a las compañeras de apartamento por brindarme su apoyo en este poco tiempo.

A todos aquellos que de alguna manera han transitado por mi vida y han dejado en ella una huella indestructible que no se borrará con el paso del tiempo.

A todos, muchísimas gracias.

De Yislén:

A mis padres por el amor brindado en todos estos años y por guiarme siempre por el mejor camino.

A mis abuelos Nelsa y René por ser tan intransigentes con mis estudios, confiar tanto en mí y brindarme siempre su amor y su apoyo incondicional.

A mi hermanito por ser el pedacito de persona que inconscientemente hace que me esfuerce y me sacrifique para vencer cualquier obstáculo.

A mi novio por el amor y la comprensión en estos últimos años y por llenar mi corazón en los momentos más difíciles de mi vida.

A mis suegros por ser tan maravillosos y por tener siempre dispuesto ese corazón lleno de nobleza, ternura y amor.

A mi tío Renesito, mi primo Yunier y toda mi familia que esperan lo mejor de mí y me han brindado el cariño, la fe y el aliento que me ha hecho falta para seguir adelante.

A Roberto por entenderme cuando más lo he necesitado.

A mi hermanita Leydis por llegar a ser más que una simple amistad, más que una compañera, más que una amiga, por enseñarme lo que es el amor de una hermana en todos estos cinco años de estudios.

Agradecimientos

A Diana, Linet, Alberto, Armando, Onaysi, Danay por ser aquellas personas que estuvieron conmigo en los buenos y en los malos momentos, siempre brindándome lo mejor de sí.

A Martha Denia, Heydi, Martha Nieves, Delvis, Richard, Carlos Luis por toda la atención y el apoyo brindado en este periodo de tiempo.

En fin a los que han sabido marcar una etapa inolvidable en mi vida.

De corazón, a todos, muchísimas gracias.

DEDICATORIA

"A mi mamá por ser mi guía, mi razón de ser, mi apoyo y por brindarme siempre su amor y cariño, gracias por existir mamita"

"En especial a mi papá por ser tan comprensivo conmigo cuando pudo estar a mi lado y aunque no se encuentra físicamente siempre estará en mi pensamiento y mi corazón"

Onaysi

"A mi mamá por ser la razón de mi ser, lo más grande que tengo y lo que más quiero en esta vida, por ser la persona por la que lucho y me entrego a diario para que se sienta orgullosa de mí"

Yislén

RESUMEN

Con el aumento de la informatización a escala mundial, la demanda de software crece exponencialmente y por consiguiente la calidad en los productos se hace más importante. Cuba ha comenzado a formar parte de este mercado debido a las perspectivas económicas que brinda.

Las pruebas de software son la actividad más comúnmente realizada para el control de calidad en los proyectos de desarrollo. La Universidad de las Ciencias Informáticas (UCI) desde la creación de los grupos de calidad, viene dando pasos de avances en el tema.

Dentro del conjunto de proyectos acogidos por la UCI se desarrolla el Sistema de Manejo de Datos de Ensayos Clínicos Cubano (SIMDECC) para el Centro de Inmunología Molecular (CIM). Este centro está dedicado al desarrollo de biomoléculas y otros fármacos para el tratamiento de enfermedades, principalmente el cáncer, de ahí que surja la necesidad de una herramienta informática con el objetivo de agilizar y automatizar los Ensayos Clínicos (EC), facilitando el manejo y almacenamiento de los mismos.

Este trabajo se centra en el diseño y aplicación de pruebas de liberación al proyecto SIMDECC, para lograr eliminar el mayor número de fallas existentes en el mismo, y de esta manera obtener una aplicación con un mínimo de errores y por consiguiente mayor calidad. Para conseguir este objetivo se trazó una estrategia de prueba como guía para todo el proceso de liberación, analizándose los resultados obtenidos a partir de cada una de las pruebas realizadas y concluyendo con una valoración del sistema.

PALABRAS CLAVES:
Calidad de software, pruebas de software, niveles de prueba, técnicas de prueba, métodos de prueba, proceso de pruebas de liberación, carga estrés.

ÍNDICE

Índice

INTRODUCCIÓN

La informática, y como parte de esta la producción de software, han alcanzado en la actualidad un elevado auge e importancia a nivel mundial. La calidad de los sistemas informáticos se ha convertido hoy día en uno de los principales objetivos estratégicos de las organizaciones debido a que cada vez más, el aseguramiento de la calidad en los sistemas informáticos influye en su buen funcionamiento.

Cuba también se ha ido desarrollando en este sentido y hoy día la vinculación de todas sus ramas: económicas, políticas y sociales con el mundo informático, son de suma importancia para el estado cubano. La producción de software no solo trae beneficios desde el punto de vista del desarrollo de sistemas para el uso interno; sino también es una manera de introducirse en el mercado a escala mundial aprovechando su perspectiva económica. Este proceso requiere de una mejora constante del producto que garantice la calidad del mismo.

La calidad de software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia, [1] y principalmente interviene en la competencia y aceptación del producto por parte de los clientes.

Es por ello que el desarrollo de la industria de software implica también que los productos realizados deban ser confiables, precisos, rápidos de utilizar, flexibles y además tengan una buena documentación. Un producto cumple con estos requisitos cuando se le han aplicado las técnicas de Ingeniería de Software adecuadas, se han utilizado los roles apropiados para el desarrollo de las tareas en la empresa de software y en su etapa de comprobación se le han aplicado las pruebas necesarias para lograr el nivel de calidad requerido. [2] Estas pruebas no son más que el proceso de probar un sistema con el propósito de descubrir errores antes de ser entregado al usuario final.

Con el vertiginoso auge de la industria de software en Cuba se han desarrollado diferentes productos informáticos para el sector de la salud. Entre sus elementos puntuales se encuentra el polo científico médico, el cual está formado por diferentes centros orientados a variadas vertientes investigativas que han llevado desde sus inicios, una incesante búsqueda científica y la elaboración de nuevos fármacos para el tratamiento de disímiles enfermedades. El Centro de Inmunología Molecular (CIM) es uno de

estos protagonistas, dedicado al desarrollo de biomoléculas y otros fármacos para el tratamiento de enfermedades, principalmente el cáncer.

Como resultado de estas investigaciones surge el Sistema de Manejo de Datos de Ensayos Clínicos Cubano (SIMDECC) en el marco de un proyecto de desarrollo colaborativo entre la Universidad de las Ciencias Informáticas (UCI) y el CIM, enfocado a mejorar la calidad de los Ensayos Clínicos (EC). El SIMDECC es una herramienta informática que tiene como objetivo automatizar los EC, facilitando el manejo y almacenamiento de los mismos. Consta de seis módulos fundamentales:

- Administración: permite gestionar los permisos y roles de los usuarios que accederán al sistema; así como la gestión de sitios, hospitales y direcciones IPs de las computadoras en las cuales estará instalada la aplicación.
- Cronograma: permite la edición del Cronograma de Ejecución del EC existente.
- Validación: permite la validación de las subvariables del Cuaderno de Recogida de Datos (CRD), a través del establecimiento de reglas de validación y de derivación.
- Publicador: permite el llenado del CRD para los pacientes del ensayo, según el Cronograma de Ejecución generado para cada uno.
- Monitoreo: permite el monitoreo de los datos de los pacientes de un sitio.
- Reporte: permite realizar reportes a partir de los datos entrados de los pacientes en el ensayo.

El SIMDECC, certificado con el nombre alas CLÍNICAS, ha llegado a su fase final de desarrollo, pero aún no ha pasado por un laboratorio de pruebas que garantice un producto con una documentación libre de errores, y una aplicación que cumpla con los requerimientos especificados y las necesidades o expectativas del cliente, por lo que se plantea como **problema científico de la investigación** ¿Cómo verificar la calidad del Sistema de Manejo de Datos de Ensayos Clínicos Cubano?

El **objeto de estudio** que se define es el proceso de realización de pruebas en la industria de software y el **campo de acción de la investigación** se enmarca en las pruebas de Liberación en el Laboratorio de Calidad de la facultad 6.

Para dar solución a este problema científico de la investigación se define como **objetivo general:** Desarrollar un Proceso de Pruebas de Liberación de Software al SIMDECC.

Del objetivo general se definen los siguientes **objetivos específicos de la investigación:**

- Realizar revisiones a los documentos empleando listas de chequeo.
- Diseñar casos de pruebas.
- Realizar pruebas de liberación al software.
- Evaluar los resultados obtenidos.

Para lograr el cumplimiento de los objetivos anteriormente señalados se proponen las siguientes **tareas**:

- Estudio bibliográfico de los temas de calidad de software, pruebas de software, herramientas y métodos para realizar las pruebas de software.
- Realización del Plan de Pruebas.
- Revisión de los documentos del SIMDECC a través de listas de chequeo.
- Realización del informe de No Conformidades.
- Diseño de Casos de Prueba.
- Realización de pruebas funcionales al software.
- Realización de las pruebas de Carga y Estrés.
- Realización del resumen de los errores más significativos detectados en el proceso de pruebas.
- Análisis de las pruebas de carga y estrés.
- Evaluación del software teniendo en cuenta atributos de calidad.

CAPÍTULO 1: Fundamentación Teórica

En este capítulo se brinda información acerca del estado del arte de las pruebas de software y se da una panorámica acerca de sus objetivos, estrategias, niveles, técnicas y métodos conocidos para la realización de estas pruebas de software. Se hace alusión a algunas herramientas que se utilizan para el desarrollo de las mismas y la importancia que tiene desarrollar un proceso de pruebas de liberación.

CAPÍTULO 2: Diseño, aplicación y propuesta de pruebas de liberación de software

En este capítulo se da a conocer una propuesta del proceso de pruebas de liberación de software donde inicialmente se describe el sistema a probar y se da una breve panorámica acerca del proceso

de liberación de un software; se elabora el plan de pruebas donde se describe la estrategia de trabajo llevada a cabo en el proceso de pruebas, especificando cuáles fueron los niveles, técnicas, métodos y herramientas que se aplicaron; así como una breve descripción de cómo se fueron empleando cada uno de ellos.

CAPÍTULO 3: Resultados de las pruebas de software aplicadas al SIMDECC

En este capítulo se muestran y se analizan los resultados obtenidos en la documentación del sistema y en la aplicación luego de ejecutar las pruebas pertinentes en el proceso de liberación, y se da una evaluación al software teniendo en cuenta atributos de calidad.

CAPÍTULO 1

Fundamentación Teórica

1.1 Introducción

Las organizaciones que desarrollan software manifiestan cada vez más que es crucial verificar y evaluar la calidad de lo construido de modo que se minimice el costo de su reparación, siendo el proceso de pruebas un punto clave a la hora de detectar errores o fallas y obtener un producto con un alto grado de fiabilidad.

En este capítulo se brinda información acerca del estado del arte de las pruebas de software y se da una panorámica acerca de sus objetivos, estrategias, niveles, técnicas y métodos conocidos para la realización de estas pruebas de software. Se hace alusión a algunas herramientas que se utilizan para el desarrollo de las mismas y la importancia que tiene desarrollar este proceso de pruebas.

1.2 Calidad de Software

La calidad de software es un problema actual que afecta tanto a los productores de software como a los clientes, y es una actividad de protección que se aplica a lo largo de todo el proceso de Ingeniería de Software. Dentro de sus definiciones se encuentran:

“Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente” R.S. Pressman (1992). [3]

“El conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas” ISO 8402 (UNE 66-001-92). [4]

"La calidad del software es el grado con el que un sistema componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario" (IEEE, Std. 610-1990). [4]

En el desarrollo de software, la calidad es un factor preocupante a la que se dedican múltiples esfuerzos y engloba los siguientes aspectos:

- Un enfoque de gestión de calidad.
- Tecnología de Ingeniería del Software efectiva (métodos y herramientas).
- Revisiones técnicas formales que se aplican durante el proceso del software.
- Una estrategia de prueba multiescala.
- El control de la documentación del software y de los cambios realizados.
- Mecanismos de medición y de generación de informes.

La calidad debe ser especificada, planificada, administrada, medida y certificada. Esto implica una visión integral que arroja la comprobación del software con el fin de lograr un mayor grado de satisfacción y confianza del cliente hacia la organización productora de software. Constituye entonces las pruebas de software, tarea de alta prioridad para las empresas productoras. [3]

1.3 Pruebas de Software

Las pruebas de software se llevan a cabo en una fase en el desarrollo del software consistente en probar las aplicaciones construidas. Es el proceso de ejercitar un programa con la intención específica de encontrar errores previos a la entrega al usuario final. Es una actividad en la cual un sistema o componente es ejecutado bajo diferentes condiciones o requerimientos específicos, donde los resultados son observados y registrados, y se realiza una evaluación de algún aspecto del sistema o componente. Forman un conjunto de herramientas, técnicas y métodos que permiten verificar y revelar la calidad de un producto de software.

Realizar pruebas a un sistema informático no significa necesariamente que el proceso de desarrollo esté asegurado y tampoco que de manera directa esté mejorando. Pero implementar un proceso de pruebas de software y más aún sostenerlo en tiempo, es un buen inicio para más adelante aumentar el

alcance y con base en las reflexiones al interior del equipo y de los hallazgos registrados en su producto, realizar un mejoramiento del proceso de desarrollo basado en los lineamientos del aseguramiento de calidad de software.

Obteniendo la calidad requerida en el software a través de las pruebas, se logra reducir el número de errores, se alcanza una mayor fiabilidad para las funciones que debe realizar el mismo, mayor eficiencia e integridad de los datos, así como mayor flexibilidad y reusabilidad.

1.3.1 Objetivos de las pruebas de software

Dentro de los objetivos fundamentales que se persiguen al aplicarle las pruebas a un software se encuentran los siguientes:

- Brindar un mayor nivel de confiabilidad en los productos que se van generando.
- Detectar fallas o errores.
- Aumentar la calidad del producto final.

Los objetivos anteriores cambian la idea que normalmente se tiene cuando se plantea que una prueba exitosa es aquella donde no se detectan errores. El objetivo principal es diseñar pruebas que sistemáticamente reflejen diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo. [3]

1.3.2 Estrategia de prueba de software

Una estrategia de prueba de 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. [5]

Las características generales son:

- La prueba comienza a partir de los componentes más pequeños del software.
- En diferentes puntos son adecuadas a la vez distintas técnicas de prueba.
- La pruebas son realizadas por un grupo de pruebas independiente al equipo de desarrollo.

- La prueba y la depuración son actividades diferentes.

Una estrategia de prueba para el software debe constar de pruebas de bajo nivel, así como de pruebas de alto nivel. Más concretamente, los objetivos de la estrategia de prueba son:

- Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de unidad, integración y las pruebas de sistema. Las pruebas de unidad y de integración son necesarias dentro de la iteración, mientras que las pruebas de sistema son necesarias sólo al final de la iteración.
- Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar, cómo realizar las pruebas y creando, si es posible, componentes de prueba ejecutables para automatizar las pruebas.
- Realizar diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Los productos de desarrollo de software en los que se detectan defectos son probados de nuevo y posiblemente devueltas a otra etapa, como diseño o implementación, de forma que los defectos puedan ser arreglados.

1.3.3 Plan de prueba

El plan de prueba describe las estrategias, recursos y planificación de las pruebas que se llevarán a cabo en un proceso de liberación.

El propósito del plan de pruebas es dejar de forma explícita el alcance, el enfoque, los recursos requeridos, el calendario, los responsables y el manejo de riesgos de un proceso de pruebas. [6]

Está constituido por un conjunto de pruebas. Cada prueba debe:

- Dejar claro qué tipo de propiedades se quieren probar: corrección, robustez, fiabilidad.
- Dejar claro cómo se mide el resultado.
- Especificar en qué consiste la prueba.
- Definir cuál es el resultado que se espera.

1.3.4 Niveles de prueba de software

Cuando se le van a aplicar pruebas a un software, se tienen en cuenta una serie de objetivos en diferentes escenarios y niveles de trabajo, debido a que las pruebas son agrupadas por niveles que se encuentran en distintas etapas del proceso de desarrollo. Los niveles de pruebas de software son los siguientes:

- Prueba de unidad
- Prueba de integración
- Prueba de sistema
- Prueba de aceptación

1.3.4.1 Prueba de unidad

La prueba de unidad es la primera fase de las pruebas dinámicas y se realizan sobre cada módulo del software de manera independiente. El objetivo es comprobar que el módulo, entendido como una unidad funcional, está correctamente codificado. [7]

Se enfocan en un programa o un componente que desempeña una función específica que puede ser probada y que se asegura que funcione tal y como lo define la especificación del programa. Los programadores siempre prueban el código durante el desarrollo, por lo que las pruebas unitarias son realizadas solamente después de que el programador considera que el componente está libre de errores.

Consiste en una prueba estructural enfocada a los elementos más pequeños del software. Esta prueba es aplicable a componentes representados en el modelo de implementación, para verificar que los flujos de control y de datos estén cubiertos y que ellos funcionen como se espera.

Durante la prueba de unidad, la comprobación selectiva de los caminos de ejecución es una tarea esencial. Se deben diseñar casos de prueba para detectar errores debidos a cálculos incorrectos, comparaciones incorrectas o flujos de control inapropiados. Las pruebas del camino básico y de bucles son técnicas muy efectivas para descubrir una gran cantidad de errores en los caminos.

1.3.4.2 Prueba de integración

Su objetivo es identificar errores introducidos por la combinación de programas o componentes probados unitariamente, para asegurar que la comunicación, enlaces y los datos compartidos ocurran apropiadamente. Se diseñan para descubrir errores o completitud en las especificaciones de las interfaces.

En este nivel se asegura que las interfaces y ligas entre las partes del sistema trabajen apropiadamente. Antes de las pruebas de integración, los componentes tuvieron que haber pasado sus pruebas individuales, por lo que, el enfoque ahora es sobre el flujo de control entre los módulos, y sobre los datos que son intercambiados entre ellos de manera independiente.

1.3.4.3 Prueba de sistema

Esta prueba tiene como objetivo verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las operaciones apropiadas funcionando como un todo. Es similar a la prueba de integración pero con un alcance mucho más amplio. Es en esta prueba donde se buscan los defectos globales dados por la mala integración de los módulos y que impiden una buena aceptación en la decisión del cliente. La responsabilidad es de todos los creadores de cada uno de los elementos del sistema.

1.3.4.4 Prueba de aceptación

Son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales.

La prueba de aceptación puede tener lugar a lo largo de semanas o meses, descubriendo así errores acumulados que pueden ir degradando el sistema. Es muy recomendable que las pruebas de aceptación se realicen en el entorno en que se va a explotar el sistema incluyendo el personal que lo

va a manejar. En caso de un producto de interés general, se realizan pruebas con varios usuarios que reportarán sus valoraciones sobre el producto.

1.3.4.5 Niveles de pruebas seleccionados

Después del estudio de los diferentes niveles de prueba realizado anteriormente, se decide utilizar las pruebas de integración y sistema.

Se decide escoger la prueba de integración porque es necesario verificar cómo funciona el intercambio de datos en cada una de las interfaces expuestas por los módulos en el sistema y su interacción a partir de estas funcionalidades una vez que se hayan integrado los programas o componentes previamente probados.

Se decide escoger la prueba de sistema porque es necesario verificar el programa final, después que todos los componentes de software y hardware han sido integrados, asegurando que la incorporación del software a los elementos del sistema funciona correctamente y realiza las debidas funciones.

1.3.5 Técnicas de prueba de software

Cuando se aplican pruebas a un software se hace necesaria la utilización de varias técnicas que facilitan comprobar el funcionamiento del mismo de una forma más detallada. Existen diferentes técnicas de prueba de software, las cuales se nombran a continuación:

- Pruebas de Funcionalidad.
- Pruebas de Usabilidad.
- Pruebas de Fiabilidad.
- Pruebas de Rendimiento.
- Pruebas de Soportabilidad.

1.3.5.1 Pruebas de Funcionalidad

La prueba de funcionalidad se enfoca en requerimientos para verificar que se corresponden directamente a casos de uso 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. Esta técnica de prueba se basa en el método de caja negra, y consiste en verificar la aplicación y sus procesos interactuando con la aplicación por medio de la interfaz de usuario y analizar los resultados obtenidos.

Prueba Funcional

La prueba funcional tiene como objetivo asegurar el trabajo apropiado de los requisitos funcionales; incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados.
Verifica el procesamiento, recuperación e implementación adecuada de las reglas del negocio y la apropiada aceptación de datos utilizando el método de caja negra.

Prueba de 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 y seguridad en el ámbito de sistema, incluyendo conexión, o acceso remoto al sistema.

La seguridad en el ámbito de aplicación consiste en que los usuarios finales están restringidos a funciones o casos de uso específicos o limitados, en los datos que están disponibles para ellos; por lo que tiene como objetivo 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 finales con derecho a acceder al sistema son capaces de acceder a las aplicaciones a través de los puntos de ingresos apropiados, por lo que tiene como objetivo verificar que solo los actores con acceso al sistema y a las aplicaciones, puedan acceder a ellos.

Prueba de Volumen

La prueba de volumen somete al software a grandes cantidades de datos para determinar si se alcanzan límites que causen la falla del software. La prueba de volumen identifica la carga máxima continua que puede manejar el software a prueba en un período dado.

Tiene como objetivo verificar que el software funciona correctamente con volúmenes de datos grandes:

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

1.3.5.2 Pruebas de Usabilidad

La prueba de usabilidad consiste en realizar pruebas efectuadas con usuarios, con el objetivo de determinar si la organización de los contenidos y las funcionalidades que se ofrecen desde el sitio web son entendidas y utilizadas por los usuarios de manera simple y directa. Las pruebas tradicionales son:

Prueba Inicial

La prueba inicial se hace para ver cómo funciona la organización de contenidos y elementos iniciales de diseño (botones, interfaces). El material con que se prueba es una imagen dibujada del sitio web.

Prueba de Boceto Web

La prueba de boceto web tiene como objetivo entender la navegación y verificar si se pueden cumplir tareas y si el usuario entiende todos los elementos que se le ofrecen. El material con que se prueba es una maqueta web semifuncional.

Prueba de Interfaz de Usuario

La prueba interfaz de usuario verifica que la interfaz del usuario proporcione al usuario el acceso y navegación a través de las funciones apropiadas. Además asegura que los objetos presentes en la

interfaz de usuario se muestren como se espera y conforme a los estándares establecidos por la empresa o de la industria. [8]

Esta prueba tiene como técnica crear o modificar pruebas para cada ventana verificando la navegación y los estados de los objetos para cada ventana de la aplicación y cada objeto dentro de la ventana.

1.3.5.3 Pruebas de Fiabilidad

La fiabilidad de un software es la probabilidad de que un programa realice su objetivo satisfactoriamente, sin fallos, en un determinado periodo de tiempo y en un entorno concreto. Las pruebas de fiabilidad están constituidas por prueba de integridad, prueba de estructura, prueba de stress y prueba de falla y recuperación.

Prueba de integridad

La prueba de integridad es la prueba que se le realiza a un software para comprobar la resistencia a los fallos. Debe estar conforme técnicamente en cuanto a lenguaje, sintaxis y uso de recursos de implementación.

Prueba de Estructura

Utilizan el método de Caja Blanca próximo a tratar, son también conocidas como "pruebas basadas en código", donde se enfocan en probar cada una de las estructuras de código, para que su comportamiento sea el esperado. Son las pruebas donde se conoce la estructura interna del componente a probar, y se efectúa una prueba sobre dicha estructura. En el caso de una aplicación web también se revisa la estructura interna de los links y otros elementos.

Prueba de Estrés

La prueba de estrés es 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 estrés también puede usarse para identificar el trabajo máximo que el software puede manejar. [8]

Tiene como técnica usar las pruebas desarrolladas para performance y prueba de carga. Para probar recursos limitados, las pruebas se deben ejecutar en una sola máquina, y se debe reducir o limitar la memoria en el servidor. Para las pruebas de estrés restantes, deben usarse múltiples clientes, cualquiera que ejecute las mismas pruebas o pruebas complementarias para producir el peor caso de volumen de operaciones.

La prueba de estrés se puede ver reflejada cuando se quiere medir la resistencia del sistema, aumentando la carga simultánea de trabajo con la misma cantidad de recursos disponibles.

Prueba de Falla y Recuperación

Las pruebas de fallas y recuperación aseguran que el software puede recuperarse de fallas de hardware, software o mal funcionamiento de la red sin pérdida de datos o de integridad de los datos. [8] La técnica es usar pruebas creadas para probar funcionalidad y ciclos de negocio para crear una serie de operaciones. En la prueba se incluyen los siguientes tipos de condiciones:

- Interrupción de energía al cliente.
- Interrupción de energía al servidor.
- Interrupción de comunicaciones mediante los servidores de la red.
- Interrupción de comunicación o pérdida de energía de los discos del servidor o con los controladores.
- Ciclos incompletos (procesos de filtro de datos interrumpidos, procesos de sincronización de datos interrumpidos).
- Punteros a la base de datos o claves inválidos.
- Elementos de datos en la base de datos inválidos o corruptos.

1.3.5.4 Pruebas de Rendimiento

Las pruebas de rendimiento son las pruebas que se realizan, desde una perspectiva, para determinar lo rápido que realiza una tarea de un sistema en condiciones particulares de trabajo. También puede servir para verificar otros atributos de calidad del sistema, tales como la escalabilidad, fiabilidad y uso de los recursos. Existen diferentes tipos de pruebas de rendimiento como son:

Prueba de Contención

Las pruebas de contención son aquellas que tienen como objetivo verificar que el elemento que se prueba maneja adecuadamente cuando muchos actores solicitan el mismo recurso, por ejemplo: registro de datos, memoria, entre otros.

Prueba de Carga

La prueba de carga somete los objetos a verificar a diferentes cargas de trabajo para medir y evaluar los comportamientos de performance y la habilidad de los objetos de continuar funcionando apropiadamente bajo diferentes cargas de trabajo. El objetivo es determinar y asegurar que el sistema funciona apropiadamente en circunstancias de máxima carga de trabajo esperada. Además evaluar las características de performance, como tiempos de respuesta, tiempos de transacciones y otros elementos sensitivos al tiempo. [8]

La técnica que utiliza es el uso de pruebas desarrolladas para funciones o ciclos de negocios y modificar archivos de datos para aumentar el número de transacciones, o las pruebas para aumentar la cantidad de ocurrencia de transacciones.

Prueba Profile

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. Esta prueba tiene como técnica usar procedimientos de prueba desarrollados para verificar funciones o ciclos de negocio, modificar archivos de datos para aumentar el número de transacciones o los procedimientos de prueba para aumentar el número de iteraciones de ocurrencia de transacciones. Las pruebas se deben ejecutar en una y se debe repetir con múltiples usuarios.

1.3.5.5 Pruebas de Soportabilidad

Las pruebas de soportabilidad son otras de las pruebas que se le pueden realizar al software para solucionar problemas de una aplicación. Existen diferentes tipos de pruebas de soportabilidad, las cuales son: prueba de configuración y prueba de instalación.

Prueba de Configuración

La prueba de configuración verifica el funcionamiento del software con diferentes configuraciones de software y hardware. [8] Su técnica es usar las pruebas de funcionalidad.

- Abrir y cerrar varias sesiones de software que no son objeto de prueba, como parte de la prueba o antes de comenzar la prueba.
- Ejecutar operaciones seleccionadas para simular la interacción del actor con el software objeto de prueba y con el software que no es objeto de prueba.
- Repetir los procedimientos anteriores minimizando la memoria convencional disponible en la máquina cliente.

Prueba de Instalación

La prueba de instalación tiene dos propósitos. Uno es asegurar que el software puede ser instalado en diferentes condiciones (como una nueva instalación, una actualización, y una instalación completa o personalizada) bajo condiciones normales y anormales. Condiciones anormales pueden ser insuficiente espacio en disco, falta de privilegios para crear directorios, etc. El otro propósito es verificar que, una vez instalado, el software opera correctamente. Esto significa normalmente ejecutar un conjunto de pruebas que fueron desarrolladas para prueba de funcionalidad. [8]

Tiene como técnica:

- Validar la condición de la máquina destino (nueva, nunca instalado, misma versión, versión anterior ya instalada).
- Realizar la instalación.

- Ejecutar un conjunto de pruebas funcionales ya implementadas para la prueba de funcionalidad.

1.3.5.6 Técnicas de prueba seleccionadas

Después del estudio realizado sobre las diferentes técnicas de prueba se llega a la conclusión de que se utilizarán para la realización de pruebas funcionales: prueba funcional, prueba de seguridad, prueba de interfaz de usuario y prueba de carga y estrés.

Se aplicará prueba funcional porque asegura el funcionamiento apropiado del objeto de prueba, incluyendo la navegación, entrada de datos, proceso de errores y recuperación; la prueba de seguridad porque mediante ella se determinan los niveles de permiso de usuarios, las operaciones de acceso al sistema y acceso a datos; la prueba de interfaz de usuario ya que verifica a través de la navegación la estandarización de las interfaces y su vinculación teniendo en cuenta el usuario autenticado, y las pruebas de carga y estrés porque es necesario comprobar el rendimiento del sistema soportando la cantidad máxima de usuarios conectados y su comportamiento al aumentar esta carga con los mismos recursos disponibles.

1.3.6 Métodos de prueba de software

Los métodos de prueba de software tienen el objetivo de diseñar pruebas que descubran diferentes tipos de errores con menor tiempo y esfuerzo. Proporcionan un mecanismo de ayuda para asegurar la calidad con la que cuenta un software y la mayor probabilidad de descubrimiento de errores que tiene el mismo.

Los métodos tradicionales son:

- Método de Caja Negra
- Método de Caja Blanca

1.3.6.1 Método de Caja Negra

El método de caja negra se centra en los requisitos funcionales y se lleva a cabo sobre la interfaz del software. Tiene como objetivo demostrar que las funciones del software son operativas, que las entradas se acepten de forma adecuada y se produzca un resultado correcto, y que la integridad de la información externa se mantenga. . De manera análoga, los datos de prueba se escogerán atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que este corra bien.

A continuación se citan los criterios mínimos por los que se deben guiar al escoger los datos de prueba de los programas:

- Valores fáciles: el programa se depurará con datos que sean fáciles de comprobar.
- Valores típicos realistas: siempre se ensayará un programa con datos seleccionados para que representen cómo se aplicará. Tales datos han de ser suficientemente sencillos, de modo que los resultados sean verificables en forma manual.
- Valores extremos: muchos programas cometen errores en los límites de sus rangos de aplicaciones. Es muy fácil que las instrucciones que incluyen a los contadores de ciclos o que hacen referencias a las dimensiones de un arreglo se equivoquen por uno.
- Valores ilegales: cuando a un programa se le insertan valores ilegales, su reacción inmediata habrá de ser por lo menos un mensaje de error adecuado para el usuario.

Es preferible que el programa ofrezca alguna indicación de errores detectados en los datos de entrada incorrectos que se han ingresado y que realice cálculos que sigan siendo factibles luego de desechar la entrada equivocada, o intente funcionar con datos predefinidos evitando así que el programa colapse. Algunos de las técnicas empleadas en las pruebas de caja negra son los siguientes:

Técnica de prueba basada en grafos

En esta técnica se debe entender los objetos que se modelan en el software y las relaciones que conectan a estos, tales como objetos de datos, objetos de programa como módulos o colecciones de sentencias del lenguaje de programación. Una vez que se ha llevado a cabo esta operación, el

siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas. [9]

En este método:

- Se crea un grafo de objetos importantes y sus relaciones.
- Se diseña una serie de pruebas que cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir errores.

La técnica de prueba basada en grafos también describe un número de modelados para pruebas de comportamiento que pueden hacer uso de los grafos:

- Modelado del flujo de transacción: los nodos representan los pasos de alguna transacción y los enlaces representan las conexiones lógicas entre los pasos.
- Modelado de estado finito: los nodos representan diferentes estados del software observables por el usuario y los enlaces representan las transiciones que ocurren para moverse de estado a estado.
- Modelado de flujo de datos: los nodos objetos de datos y los enlaces son las transformaciones que ocurren para convertir un objeto de datos en otro.
- Modelado de planificación: los nodos son objetos de programa y los enlaces son las conexiones secuenciales entre esos objetos. Los pesos de enlace se usan para especificar los tiempos de ejecución requeridos al ejecutarse el programa.
- Gráfica causa-efecto: la gráfica causa-efecto representa una ayuda gráfica en seleccionar, de una manera sistemática, un gran conjunto de casos de prueba. Tiene un efecto secundario beneficioso en precisar estados incompletos y ambigüedades en la especificación.

Partición equivalente

La técnica de prueba partición equivalente 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.

En otras palabras, este método intenta dividir el dominio de entrada de un programa en un número finito de clases de equivalencia. De tal modo que se pueda asumir razonablemente que una prueba realizada con un valor representativo de cada clase es equivalente, a una prueba realizada con cualquier otro valor de dicha clase. Esto quiere decir que si el caso de prueba correspondiente a una clase de equivalencia detecta un error, el resto de los casos de prueba de dicha clase de equivalencia deben detectar el mismo error. Y viceversa, si un caso de prueba no ha detectado ningún error, es de esperar que ninguno de los casos de prueba correspondientes a la misma clase de equivalencia encuentre ningún error.

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 equivalente es reducir el posible conjunto de casos de prueba en uno más pequeño, un conjunto manejable que evalúe bien el software. Se toma un riesgo porque se escoge no probar todo. Así que se necesita tener mucho cuidado al escoger las clases. [9]

En el diseño de casos de prueba para partición equivalente se procede en dos pasos:

- Se identifican las clases de equivalencia. Las clases de equivalencia son identificadas tomando cada condición de entrada y repartiéndola en dos o más grupos.
- Se define los casos de prueba usando las clases de equivalencia.

Análisis de Valores Límite

El análisis de valores límite (AVL) 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. [9]

Dentro del AVL se encuentran las condiciones sublímites o condiciones límites internas. Las condiciones límite normales son las más obvias de descubrir. Estas son definidas en la especificación o son evidentes al momento de utilizar el software. Algunos límites, sin embargo, son internos al

software, no son necesariamente aparentes al usuario final, pero aún así deben ser probadas por el probador. Estas son conocidas como condiciones sublímites o condiciones límite internas.

Prueba de la tabla ortogonal

La prueba de la tabla ortogonal puede aplicarse a problemas en que el dominio de entrada es relativamente pequeño pero demasiado grande para posibilitar pruebas exhaustivas ya que hay aplicaciones donde el número de parámetros de entrada es pequeño y los valores de cada uno de los parámetros están claramente delimitados. Cuando estos números son muy pequeños, es posible considerar cada permutación de entrada y comprobar exhaustivamente el proceso del dominio de entrada. En cualquier caso, cuando el número de valores de entrada crece y el número de valores diferentes para cada elemento de dato se incrementa, la prueba exhaustiva se hace impracticable.

De lo dicho anteriormente es que se puede decir que el método de prueba de la tabla ortogonal es particularmente útil al encontrar errores asociados con fallos localizados y permite proporcionar una buena cobertura de pruebas con menos casos de prueba que en la estrategia exhaustiva.

Adivinando el error

Adivinando el error es una técnica que dado un programa particular, se conjetura, por la intuición y la experiencia, ciertos tipos probables de errores y se escriben casos de prueba para exponer esos errores. La idea básica es enumerar una lista de errores posibles o de situaciones propensas a error y después escribir los casos de prueba basados en la lista. [9]

1.3.6.2 Método de Caja Blanca

El método de caja blanca denominada a veces prueba de caja de cristal, requiere del conocimiento de la estructura interna del programa. Permite comprobar los caminos lógicos del software y examinar el estado del programa en varios puntos para determinar si el estado es real y coincide con el esperado.

Estas pruebas deben garantizar como mínimo que:

- Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo.
- Se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsas.
- Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- Se ejerciten las estructuras internas de datos para asegurar su validez.

Algunas de las técnicas empleadas en las pruebas de caja blanca son los siguientes:

La prueba del camino básico

La prueba del camino básico 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. Se basa en construir un caso de prueba por camino básico que se encuentre en el grafo de programa asociado al método de la clase que se desea someter a pruebas.

La idea de esta técnica es derivar casos de prueba a partir de un conjunto dado de caminos independientes, que no son más que aquellos que introduce al menos una sentencia de procesamiento que no estaba considerada. Para obtener el conjunto de caminos independientes se construirá el grafo de flujo asociado y se calculará su complejidad ciclomática.

Prueba de ruteo

La prueba de ruteo tiene como objetivo ejecutar al menos una vez cada sentencia. Para ejecutar esta prueba se necesitan varios casos de prueba, entre estas está determinar varios caminos independientes y cada condición tomada debe cumplirse en un caso pero en el otro no.

Prueba de condición

Método que ejercita las condiciones lógicas contenidas en un módulo del programa. Una condición simple es una variable booleana o una expresión relacional. Esta prueba se concentra en la prueba de cada condición del programa para asegurar que no contiene errores. [10]

Prueba de bucle

La prueba de bucle se concentra exclusivamente en la validez de la construcción de bucles. Se pueden definir cuatro tipos de bucles: simples, anidados, concatenados, no estructurados.

1.3.6.3 Método de prueba seleccionado

Teniendo en cuenta el estudio realizado de los métodos de pruebas se decide utilizar el método de caja negra porque es una buena alternativa para comprobar las funcionalidades del sistema centrándose en los requisitos que exige el usuario final. Dentro de este método de prueba se utilizará la técnica de partición de equivalencia ya que se dirige a la definición de casos de prueba que descubren clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

1.3.7 Herramientas utilizadas en las pruebas de software

Las herramientas de automatización consolidan y mejoran la efectividad de las pruebas siempre y cuando se manejen las expectativas y se seleccione una herramienta compatible con el ambiente de programación. Algunas de ellas son:

JMeter

JMeter es una herramienta Java dentro del proyecto Jakarta, que permite la automatización de pruebas de carga y estrés para aplicaciones web.

Esta herramienta se destaca por su versatilidad y estabilidad, porque permite la simulación de usuarios, ahorra tiempo, recursos y es gratis. Su arquitectura hoy día, ha evolucionado no sólo para llevar a cabo pruebas en componentes habilitados en Internet (HTTP), sino además en Bases de Datos, programas en Perl, requisiciones FTP, entre otras.

JMeter como herramienta de prueba dispone de varios componentes que facilitan la elaboración de los escenarios de prueba con la ventaja de simular para cada uno de esos escenarios miles de usuarios.

Con esta herramienta se pueden ejecutar dos tipos de pruebas, es decir, los scripts de prueba se pueden realizar de diferentes condiciones en dependencia de la importancia del sistema que se desea probar; la elaboración de scripts simples, es recomendable para sitios estáticos, donde hay poca interacción con los usuarios, poco flujo de datos y está enfocada solo a las direcciones http del sitio web que se desea probar, mientras que las complejas son para sitios dinámicos, donde se permitirá entrar datos al sistema y enviarlos a la base de datos del mismo; esta nueva acción se realiza para lograr una mayor similitud con los escenarios reales de la navegación en el sistema.

JUnit

JUnit es un conjunto de clases que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es también un medio de controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación.

Existen diferentes extensiones de JUnit como son:

- Entorno.Net (NUnit): es un grupo de clases opensource que pertenecen al conjunto de herramientas XUnit. Constituye una herramienta que se utiliza para realizar pruebas unitarias. Es equivalente a JUnit pero con un menor nivel de integración.
- Interfaz Web (HTTPUnit): el objetivo de esta herramienta es identificar e interactuar con objetos de la página. Es responsable de mantener el contexto de sesión a través de preguntas y respuestas.
- Interfaz de Usuario (JFCUnit): es una extensión del conjunto JUnit y su objetivo es identificar e interactuar con objetos gráficos.
- Bases de Datos (DBUnit): es la herramienta que tiene como objetivo la conexión de la base de datos y diversos formatos para cargar los datos como son: XML, CSV y otros.

OpenSTA

OpenSTA es una herramienta opensource, basada en arquitectura CORBA que permite el testeo de estrés de aplicaciones web, de una forma similar a ACT (herramienta para test de estrés de

aplicaciones ASP.NET). Esta herramienta permite un mayor control del número y distribución de usuarios virtuales.

TEST

TEST es una herramienta líder en automatización de pruebas unitarias y de productos con código estándar, trabaja sobre clases escritas en la plataforma Microsoft .NET, sin requerir que los desarrolladores realicen un solo caso de prueba. Mejora la fiabilidad, funcionalidad, seguridad, desarrollo y mantenimiento de las aplicaciones .NET. Puede probar cualquier fichero o paquete que haya sido construido. Facilita la creación y ejecución de pruebas definidas por el usuario basado en el framework11 de NUnit.

JTEST 8.0

JTEST 8.0 es una herramienta que ayuda a verificar de manera automática la funcionalidad de aplicaciones cada vez más complejas, en empresas con sistemas en permanente cambio, permite reducir el tiempo de entrega del software así como una disminución del riesgo de generar software defectuoso o con problemas de vulnerabilidad.

TestNG

TestNG es un framework para pruebas de códigos creados en JAVA, que realiza pruebas de caja blanca. Es una herramienta de código abierto, fue creado para superar las principales limitaciones JUnit, y para proporcionar características adicionales necesarias para probar la última generación de aplicaciones Java. Mediante esta herramienta se pueden realizar:

- Pruebas de unidad de caja blanca
- Pruebas de lógica de código
- Pruebas funcionales de caja blanca
- Pruebas de regresión de caja blanca

Push To Test TesMaker

Push To Test TestMaker es una herramienta que pertenece a software libre y de libre distribución. Puede ser usada en conjunto de otros lenguajes muy utilizados en la actualidad como son: Java, .NET, Jython, Groovy, PHP, Ruby, para la realización de las pruebas. Esta herramienta cuenta con asistentes wizards12 y grabadores recorders13, como el TestGen4Web o el Selenium, para la construcción de pruebas, aun cuando no se cuente con ninguna experiencia como programadores. Los tipos de pruebas que realiza esta herramienta son los siguientes:

- Pruebas funcionales de caja negra
- Pruebas de carga de caja negra
- Pruebas de estrés de caja negra
- Pruebas de regresión de caja negra
- Pruebas de volumen de caja negra

Existen gran variedad de herramientas con las que se pueden realizar las pruebas de software. Después del estudio realizado de algunas de ellas se decide utilizar JMeter para la automatización de las pruebas de carga y estrés, porque logra disminuir los costos en recursos y en tiempo destinados a su realización; además utilizar JMeter en aplicaciones web para la comprobación de los recursos del sistema, supone una mayor efectividad en el proceso y en la fiabilidad de los resultados.

1.3.8 Importancia de las pruebas de software

En la producción de software, el proceso de pruebas tiene gran importancia, ya que para contribuir a la estabilidad, escalabilidad, eficiencia y seguridad del producto es necesario verificar la calidad del mismo, la cual se logra mediante las pruebas de software. La realización de estas permite:

- Reducir la posibilidad de agregar defectos al software: si hay que realizar una adición de características requeridas por el cliente y se ve que ya no funcionan bien algunas de las cosas que anteriormente servían, se puede inferir que la nueva funcionalidad es la que contiene defectos, por lo que no hay necesidad de realizar modificaciones a los componentes realizados anteriormente.
- Reducir la posibilidad de encontrar defectos en funcionalidades ya implementadas.

- Permitir una buena documentación: es preferible ver unas pequeñas líneas de "código de prueba", que son concisas y realmente son fáciles de entender, a revisar línea por línea del código para poder entender qué hace determinado componente de software.
- Reducir el costo del cambio: ya que evitan que se descubran los defectos hasta el final del desarrollo de software.
- Permitir la reimplementación: se puede llegar a necesitar reimplementar determinada funcionalidad en un sistema, debido a fallos de seguridad, rendimiento, o simplemente porque no reunía lo que el cliente esperaba de éste, entonces dada tal situación, las pruebas que se le realizan a dicha funcionalidad van a permitir que se verifique si se desarrolló de manera segura, ya que la prueba encierra el criterio de aceptación sobre la funcionalidad, permitiendo de esta forma que se vuelva a desarrollar algo acorde con la especificación.
- Restringir las características a implementar: muchas veces los programadores pierden tiempo en detalles que la especificación no pedía. Con las pruebas el programador sabe que tiene que programar dicha funcionalidad y también probarla, por lo que se restringe a lo que los diseñadores de las pruebas hayan realizado.
- Hacer que el desarrollo sea más rápido: ya que a medida que se le agregan características al software, las funcionalidades anteriores pueden fallar, pero si se han hecho las pruebas pertinentes a las funcionalidades anteriores, se puede descartar inmediatamente que existan defectos en éstas, por lo que se puede concentrar tranquilamente en la nueva funcionalidad.

1.4 Proceso de Pruebas de Liberación

El proceso de pruebas de liberación se llevará a cabo a partir de un proceso que propone la dirección de calidad de software de la universidad y por el cual se rige la facultad 6. El mismo se basa en el área de procesos de Verificación del nivel 3 de CMMI, encargada de asegurar que los productos de trabajo seleccionados cumplen sus requisitos especificados. Es decir, se realizan pruebas bajo el principio de verificar que exista correspondencia entre el producto final y las expectativas del cliente.

El proceso de pruebas de liberación no es más que un conjunto de pruebas que se realizan en la etapa final del desarrollo del software y previo a las pruebas de aceptación y despliegue. Básicamente se centra en la revisión de la documentación y la aplicación del sistema teniendo en cuenta diferentes

niveles para los que se emplean varios métodos y técnicas de una manera organizada y eficiente que conducirán a un mejor funcionamiento del producto.

1.5 Conclusiones

Después de haber realizado un análisis del estado del arte se puede determinar que para lograr una excelencia en el desempeño del producto, así como la mejor publicidad que una empresa dedicada a la producción de software pueda tener, se hace necesario realizar un proceso de liberación capaz de aumentar la calidad del mismo; por lo que se decide utilizar para el desarrollo del proceso que se llevará a cabo, pruebas de integración y sistema con la ayuda de diferentes técnicas de prueba, en este caso, prueba funcional, prueba de seguridad, prueba de interfaz de usuario y pruebas de carga y estrés basadas en el método de caja negra. Para la realización de las pruebas de carga y estrés se utilizará la herramienta JMeter.

CAPÍTULO 2

Diseño, aplicación y propuesta de pruebas de liberación de software

2.1 Introducción

En este capítulo se da a conocer una propuesta del proceso de pruebas de liberación de software donde inicialmente se describe el sistema a probar y se da una breve panorámica acerca del proceso de liberación de un software; se elabora el plan de pruebas donde se describe la estrategia de trabajo llevada a cabo en el proceso de pruebas, especificando cuáles fueron los niveles, técnicas, métodos y herramientas que se aplicaron; así como una breve descripción de cómo se fueron empleando cada uno de ellos.

2.2 Descripción del Sistema a probar

El sistema de manejo de datos que se propone para el país capaz de ayudar en los procesos de gestión y conducción de los EC, es una aplicación web que consta de seis módulos que abarcan todas las funcionalidades requeridas para estos ensayos: Administración, Cronograma, Validación, Publicador, Reporte y Monitoreo. Cada módulo tiene asociado diferentes requisitos funcionales que se agrupan por casos de uso; los cuales constituyen la secuencia de acciones que el sistema debe ejecutar, y producen un resultado observable de valor para un actor que interactúa con el sistema.

A continuación se muestra una relación de todos los casos de uso y requerimientos funcionales correspondientes a cada uno de los módulos del sistema. Cada caso de uso se encuentra clasificado como arquitectónicamente significativo, excepto uno de ellos en el módulo Publicador que está clasificado como secundario.

Módulo Administración

Casos de Uso:

- Autenticar.
- Gestionar Sitio.
- Gestionar Hospital.
- Gestionar Usuario.
- Gestionar Asignaciones.
- Gestionar Ensayo Clínico.
- Gestionar IP.

Requisitos Funcionales:

- RF1: Autenticar usuario.
- RF2: Cambiar Contrasenna.
- RF3: Registrar sitio.
- RF4: Modificar datos del sitio.
- RF5: Eliminar sitio.
- RF6: Mostrar listado de los sitios.
- RF7: Buscar sitio.
- RF8: Registrar hospital.
- RF9: Modificar hospital.
- RF10: Eliminar hospital.
- RF11: Mostrar listado de los hospitales.
- RF12: Buscar hospital.
- RF13: Registrar usuario.
- RF14: Modificar datos del usuario.
- RF15: Eliminar un usuario.
- RF16: Mostrar listado de los usuarios.
- RF17: Buscar usuario.
- RF18: Asignar rol e IPs a un usuario.
- RF19: Modificar asignación.
- RF20: Eliminar asignación.
- RF21: Mostrar listado de asignaciones.

- RF22: Buscar asignación.
- RF23: Mostrar datos del EC.
- RF24: Registrar un EC.
- RF25: Modificar EC.
- RF26: Buscar EC.
- RF27: Eliminar EC.
- RF28: Asignar número de IP y/o rango a un sitio.
- RF29: Mostrar números de IP asignados.
- RF30: Modificar datos de un IP asignado.
- RF31: Buscar IP asignado.
- RF32: Eliminar un IP asignado.

Módulo Cronograma

Casos de Uso:

- Entrar a Cronograma.
- Crear cronograma general.
- Editar cronograma general.
- Mostrar listado de etapas.
- Crear etapa.
- Gestionar etapa.
- Planificar modelo asintomático en una visita.
- Planificar modelo asintomático en un periodo.
- Definir modelo sintomático.
- Mostrar lista de planificaciones de modelos.
- Gestionar una planificación de un modelo.
- Búsqueda avanzada de planificaciones de modelos.

Requisitos Funcionales:

- RF1: Crear cronograma.
- RF2: Guardar datos del cronograma.
- RF3: Mostrar datos del cronograma.
- RF4: Modificar los datos de cronograma.

- RF5: Crear etapa.
- RF6: Eliminar etapa.
- RF7: Modificar etapa.
- RF8: Mostrar etapa.
- RF9: Mostrar listado de etapas.
- RF10: Planificar modelo asintomático en una visita.
- RF11: Planificar modelo asintomático en un periodo.
    - RF11.1: Mostrar listado de modelos.
    - RF11.2: Determinar las visitas (los días) en que se llenará el modelo de acuerdo al periodo definido.
- RF12: Definir tiempo de llenado para un modelo en una visita.
- RF13: Definir modelo sintomático.
- RF14: Planificar modelo sintomático en un plazo de tiempo.
- RF15: Mostrar la planificación de un modelo asintomático.
    - RF15.1: Mostrar la visita definida en la planificación.
    - RF15.2: Mostrar el tiempo de llenado del modelo en esta visita.
- RF16: Mostrar la planificación de un modelo sintomático.
    - RF16.1: Mostrar el plazo de llenado del modelo.
    - RF16.2: Mostrar cuándo debe llenarse el modelo.
- RF17: Modificar la planificación de un modelo en una visita.
- RF18: Mostrar lista de planificaciones de los modelos.
    - RF18.1: Mostrar lista de planificaciones de asintomáticos.
    - RF18.2: Mostrar lista de planificaciones de sintomáticos.
- RF19: Mostrar cronograma.
    - RF19.1: Mostrar modelos sintomáticos en las visitas definidas.
    - RF19.2: Mostrar modelos asintomáticos en el plazo del tiempo definido.
- RF20: Eliminar la planificación de un modelo.
- RF21: Buscar planificaciones por etapa.
- RF22: Buscar planificaciones por periodo.
- RF23: Entrar al módulo Cronograma.

Módulo Validación

Casos de Uso:

- Validar Variable.
- Modificar la validación de una variable.
- Ver validación de una variable.
- Firmar validación.
- Quitar firma a la validación.
- Iniciar Validación.

Requisitos Funcionales:

- RF1: Mostrar modelos.
- RF2: Mostrar las variables de un modelo.
- RF3: Mostrar el tipo de una variable.
- RF4: Mostrar los tipos de reglas de una variable según el tipo de la variable.
- RF5: Guardar las reglas especificadas para una variable.
  - RF5.1: Cambiar estado de una variable a “validada”.
- RF6: Modificar las reglas especificadas para una variable.
- RF7: Ver las reglas de validación de una variable.
- RF8: Firmar la validación de las variables.
- RF9: Quitar firma a la validación de las variables.
- RF10: Entrar al módulo Validación.

Módulo Publicador

Casos de Uso:

- Iniciar Publicador.
- Adicionar paciente.
- Definir estado de inclusión de un paciente.
- Mostrar listado de visitas de un paciente.
- Generar cronograma de un paciente.
- Mostrar cronograma de un paciente.
- Llenar modelo a un paciente.
- Modificar modelo de un paciente.
- Mostrar modelo de un paciente.

- Firmar modelo.
- Quitar firma a un modelo.
- Cerrar sitio.
- Cerrar ensayo.
- Mostrar queries de un sitio.
- Mostrar inconsistencias de un sitio.
- Mostrar queries libres de un sitio.
- Mostrar queries de un paciente.
- Mostrar inconsistencias de un paciente.
- Mostrar queries de un modelo.
- Mostrar inconsistencias de un modelo.
- Gestionar query libre.
- Gestionar query asociada.

Requisitos Funcionales:

- RF1: Entrar al módulo Publicador.
- RF2: Mostrar listado de sitios y hospitales.
- RF3: Mostrar datos de un sitio.
- RF4: Mostrar pacientes de un sitio.
- RF5: Mostrar datos de un paciente.
- RF6: Adicionar paciente.
- RF7: Modificar estado de inclusión de un paciente.
- RF8: Incrementar cantidad de pacientes de un ensayo.
- RF9: Mostrar visitas de un paciente.
- RF10: Mostrar los datos de una visita de un paciente.
- RF11: Mostrar modelo asintomático de una visita.
- RF12: Mostrar modelos sintomáticos en un periodo.
- RF13: Mostrar estado de inclusión de un paciente.
- RF14: Mostrar estado de tratamiento de un paciente.
- RF15: Mostrar estado de un modelo.
- RF16: Mostrar cantidad de inconsistencias de un modelo.
- RF17: Mostrar cantidad de queries de un modelo.
- RF18: Generar cronograma de un paciente.

- RF19: Definir fecha de una visita.
- RF20: Mostrar modelos sintomáticos.
- RF21: Mostrar visitas.
- RF22: Mostrar fechas de las visitas.
- RF23: Mostrar etapas.
- RF24: Mostrar modelos asintomáticos.
- RF25: Mostrar plazo de tiempo para el llenado de los modelos asintomáticos.
- RF26: Mostrar variables de un modelo.
- RF27: Adicionar datos a un modelo.
- RF28: Modificar estado de un modelo.
- RF29: Modificar estado del tratamiento de un paciente.
- RF30: Mostrar inconsistencia de una variable.
- RF31: Generar query automática.
- RF32. Quitar inconsistencia de una variable.
- RF33: Mostrar datos de las variables de un modelo.
- RF34: Modificar estado de un sitio.
- RF35: Modificar estado del ensayo.
- RF36: Mostrar listado de queries de un sitio.
- RF37: Mostrar datos de una query.
- RF38: Mostrar listado de inconsistencias de un sitio.
- RF39: Mostrar datos de una inconsistencia.
- RF40: Mostrar listado de queries libres de un sitio.
- RF41: Mostrar datos de una query libre.
- RF42: Mostrar listado de queries de un paciente.
- RF43: Mostrar listado de inconsistencias de un paciente.
- RF44: Mostrar listado de queries de un modelo.
- RF45: Mostrar listado de inconsistencias de un modelo.
- RF46: Adicionar mensaje a una query libre.
- RF47: Adicionar mensaje a una query asociada.

Módulo Reporte

Casos de Uso:

- Entrar a reporte.
- Realizar reporte por variables independientes.
- Realizar reporte por modelo.
- Filtrar reportes.

Requisitos Funcionales:

- RF1: Entrar al módulo reporte.
- RF2: Mostrar visitas.
- RF3: Mostrar modelos de una visita.
- RF4: Mostrar submodelos de un modelo.
- RF5: Mostrar variables de un submodelo.
- RF6: Mostrar variables de un modelo.
- RF7: Buscar pacientes por sitio.
- RF8: Buscar pacientes por “estado del paciente”.
- RF9: Buscar pacientes en periodo de inclusión.
- RF10: Buscar variables de un paciente en una visita.
- RF11: Mostrar reporte en formato HTML, Excel y CSV.

Módulo Monitoreo

Casos de Uso:

- Iniciar Monitoreo.
- Mostrar listado de pacientes.
- Mostrar listado de queries de un sitio.
- Modificar estado de una query.
- Mostrar listado de inconsistencias de un sitio.
- Mostrar listado de queries libres.
- Responder query libre.
- Adicionar query libre.
- Monitorear variables de un modelo.
- Mostrar listado de queries de un paciente.
- Mostrar listado de inconsistencias de un paciente.

Requisitos Funcionales:

- RF1: Entrar al módulo Monitoreo.
- RF2: Mostrar listado de Hospitales y Sitios asociados al Monitor.
- RF3: Mostrar cantidad de pacientes incluidos de un sitio.
- RF4: Mostrar cantidad de pacientes mal incluidos Waiver de un sitio.
- RF5: Mostrar cantidad de pacientes mal incluidos de un sitio.
- RF6: Mostrar cantidad de pacientes con estudio completado.
- RF7: Mostrar por ciento de datos monitoreados de un sitio.
- RF8: Mostrar por ciento de datos limpios de un sitio.
- RF9: Mostrar listado de queries de un sitio.
- RF10: Mostrar una query de un sitio.
- RF11: Mostrar listado de queries libres de un sitio.
- RF12: Mostrar una query libre.
- RF13: Adicionar una query libre.
- RF14: Adicionar un mensaje a una query libre.
- RF15: Modificar el estado de una query.
- RF16: Mostrar listado de inconsistencias de un sitio.
- RF17: Mostrar listado de pacientes de un sitio.
- RF18: Mostrar listado de pacientes con estado “Mal Incluido”.
- RF19: Mostrar listado de pacientes con estado “Mal Incluido Waiver”.
- RF20: Mostrar listado de pacientes con estado “Incluido”.
- RF21: Mostrar por ciento de datos monitoreados de un paciente.
- RF22: Mostrar por ciento de datos limpios de un paciente.
- RF23: Mostrar cantidad de inconsistencias de un paciente.
- RF24: Mostrar cantidad de queries de un paciente.
- RF25: Mostrar listado de queries de un paciente.
- RF26: Mostrar listado de inconsistencias de un paciente.
- RF27: Mostrar las visitas de un paciente.
- RF28: Mostrar los modelos por visitas de un paciente.
- RF29: Mostrar un modelo de un paciente.
- RF30: Monitorear variables de un modelo.
- RF31: Asociar query a una variable.
- RF32: Modificar query asociada a una variable.

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.

2.3 Estrategia de prueba

Para efectuar el proceso de liberación del software se trazó una estrategia de prueba capaz de reflejar el conjunto de acciones a realizar en dicho proceso de manera organizada y secuencial. La estrategia que se llevará a cabo para evaluar dinámicamente el producto de software comienza por los elementos más simples y más pequeños y va avanzando progresivamente hasta probar todo el software en su conjunto.

A continuación se muestra la guía de pasos a seguir, como procedimiento para efectuar el proceso de pruebas de liberación:

- Realización del plan de prueba.
- Definición de los niveles, técnicas y método de prueba a utilizar.
- Revisión de la documentación con la aplicación de listas de chequeo.
- Elaboración de la plantilla de no conformidades.
- Realización de pruebas exploratorias al software.
- Realización de diseños de casos de prueba.
- Realización de pruebas funcionales al software.
- Realización de las pruebas de carga y estrés.

2.3.1 Realización del plan de prueba

Teniendo en cuenta que la realización del plan de prueba [Ver expediente de proyecto SIMDECC] es el principal factor crítico para la puesta en práctica de un buen proceso de prueba, que sirva para que el producto se entregue con mejor calidad, se plasmó un conjunto de pasos y actividades que indican cómo se realizará dicho proceso. Se tuvieron en cuenta las características del software a liberar y las

pruebas que se le iban a aplicar al mismo, las cuales sirvieron para describir cada punto por los que está compuesta la plantilla del plan de prueba.

En el plan de prueba se dejaron plasmados aspectos como:

- Introducción
- Organización del equipo de prueba
- Arquitectura técnica
- Especificaciones del software y hardware
- Descripción del plan de prueba
- Estrategia de prueba
- Recursos requeridos
- Plan de proyecto
- Calendario y plazos
- Definición de los entregables
- Seguimiento y reporte de defectos
- Aprobación del plan
- Documentación de los resultados

Los problemas que surgieron en el trayecto del proceso de liberación y que produjeron varios cambios en dicho proceso, se fueron actualizando en la plantilla del plan de prueba, esto contribuyó a que en el documento quedara reflejado de manera correcta cuál fue el proceso real que se llevó a cabo para la ejecución de las pruebas con la ayuda de los recursos con los que se contaban, el plazo de tiempo que se utilizó para las mismas, y una precisión en todos los datos y aspectos tenidos en cuenta en dicho documento.

2.3.2 Revisión de la documentación

La revisión de la documentación del sistema se realizó analizando cada documento de cada módulo que compone el sistema y basándose en listas de chequeo [Anexo 1] como herramienta para detectar e identificar errores con el objetivo evaluar las especificaciones del documento a revisar. Para el proceso de medición y evaluación se tuvo en cuenta varios indicadores de la tabla que aparece en la

lista de chequeo, los cuales permitieron encontrar los puntos ineficientes que tenían los elementos chequeados. 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

Los principales aspectos que se tuvieron en cuenta para la revisión de los documentos fueron: no presencia de errores ortográficos, concordancia y claridad en los textos para un buen entendimiento de los mismos, coincidencia del número de página puesto en el índice con la página en la que se encuentra el contenido, especificación en las definiciones de los términos, entre otros más específicos para cada documento. Todas las ineficiencias fueron registradas, en una tabla de no conformidades detectadas.

2.3.3 Elaboración de la plantilla de no conformidades

La plantilla de no conformidades [Anexo 2] es un artefacto del proceso de pruebas que se realizó con el objetivo de dejar plasmados todos los errores que se encontraron de una forma clara y entendible para quienes van a darle cumplimiento. Los errores detectados durante la revisión de la documentación del sistema, fueron registrados en esta plantilla, donde se elaboró un documento por cada iteración que se hizo. Estas versiones fueron controladas en la sección de versiones hasta que se eliminaron todos los errores encontrados en el elemento que se estaba probando. Las no conformidades detectadas fueron registradas en una tabla que contiene los siguientes campos:

- Elemento: se especificó el nombre del elemento.
- No: se especificó el número de la no conformidad.
- No conformidad: se describió la no conformidad.

- Aspecto correspondiente: se describió el aspecto correspondiente.
- Etapa de detección: se describió la etapa de detección del error.
- Significativa: se puso una (X), en caso de que la no conformidad estuviera clasificada como significativa.
- No significativa: se puso una (X), en caso de que la no conformidad estuviera clasificada como no significativa.

En este documento también se reflejó una tabla con las recomendaciones realizadas, donde se especificaron los mismos aspectos de la tabla anterior, exceptuando los dos últimos campos que corresponden a la clasificación del error.

2.3.4 Realización de pruebas exploratorias

Las pruebas exploratorias son la primera etapa del proceso de pruebas, y se realizaron con el objetivo de explotar las vulnerabilidades mínimas del sistema, donde los requerimientos son menos precisos y más fluidos. En este paso de la estrategia de trabajo trazada, se exploró la aplicación prácticamente a ciegas y sin empleo de la documentación. Durante su ejecución se intentó comprender cómo funciona la aplicación y producir condiciones de error. Se pretendió además obtener información de los desarrolladores y deducir la interacción entre los componentes.

2.3.5 Diseño de casos de prueba

Los casos de prueba [Anexo 3] fueron diseñados con el propósito de especificar una forma de probar el sistema, incluyendo las entradas con las que se ha de probar, los resultados esperados y las condiciones bajo las que ha de probarse.

Conociendo que un caso de prueba se deriva de un caso de uso del modelo de casos de uso, se diseñó un caso de prueba para cada caso de uso de los diferentes módulos con que trabaja el software. Con estos casos de prueba se verificaron los requerimientos funcionales en el que se especificó cómo probar un caso de uso o un escenario específico del mismo, para verificar el resultado de la interacción entre el usuario final y el sistema y conocer si se satisfacían las precondiciones y

poscondiciones especificadas. Para detallar cada caso se uso se utilizó una tabla que consta de los siguientes campos:

- Nombre de la sección: se especificó el nombre de la sección [SC 1: Nombre de la sección].
- Escenarios de la sección: se especificaron los escenarios de la sección [EC 1.1: Nombre del Escenario].
- Descripción de la funcionalidad: se describió brevemente la funcionalidad.
- Flujo central: se describieron los pasos a desarrollar para probar la funcionalidad que se indicó.

A partir de las especificaciones del caso de uso donde se describieron los distintos caminos del flujo básico y flujo alternativo, se realizó una descripción de las variables que se encontraban en todas las interfaces que están asociadas al caso de uso, al cual se le está realizando el caso de prueba. En esta descripción se llenaron algunos campos por los cuales está conformada dicha tabla:

- No: se enumeró todos los campos o variable, descrito en el caso de uso.
- Nombre de campo: se especificó el nombre del campo de entrada.
- Clasificación: se especificó la clasificación según el componente de diseño utilizado [ejemplo: campo de texto, lista desplegable o combo box, entre otros].
- Puede ser nulo: se especificó si el campo puede ser nulo o no, para ello solo se puso Sí o No.
- Descripción: se describieron brevemente los datos que debían introducirse.

Esto dio lugar a que se realizara un juego de datos utilizando las clases válidas e inválidas identificadas con la técnica partición de equivalencia con el objetivo de probar la validez de cada uno de los datos que se introdujeron en el sistema a través de sus entradas. 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 especificó el id del escenario [EC1, EC2,…,ECn]
- Escenario: se especificó el nombre del escenario.
- Variables [1, 2,…, n]: se especificó el nombre de la variable y en su celda correspondiente se indicó el valor del dato [V (Válido), I (Inválido), N/A (No Aplica)].
- Respuesta del sistema: se escribió el resultado que se esperaba al realizar la prueba.
- Resultado de la prueba: se escribió el resultado que se obtuvo al realizar la prueba.

Luego de haber obtenido esta serie de datos se compararon los resultados que se esperaban al realizar las pruebas y los que verdaderamente se obtuvieron. Los resultados de las pruebas que no fueron satisfactorios se manifestaron como no conformidades y se emitieron en el registro de efectos y dificultades detectadas que se encuentra en la parte final de cada diseño de caso de prueba con los siguientes campos:

- Elemento: se especificó el nombre del elemento.
- No: se especificó el número de la no conformidad.
- No conformidad: se describió la no conformidad.
- Aspecto correspondiente: se especificó el aspecto correspondiente a la no conformidad.
- Etapa de detección: se especificó la etapa de detección del error.
- Significativa: se puso una (X), en caso de que la no conformidad estuviera clasificada como significativa.
- No significativa: se puso una (X), en caso de que la no conformidad estuviera clasificada como no significativa.
- Recomendación: se puso una (X), en caso de que la no conformidad solo fuera una recomendación.
- Estado NC: se colocó el estado de la no conformidad y la fecha, cada vez que se revisó se dejó el estado anterior y se colocó el nuevo con la fecha en que se revisó [RA: Resuelta, PD: Pendiente, NP: No Procede].
- Respuesta del equipo de desarrollo: esta columna se comenzó a llenar a partir de la segunda iteración, y fue responsabilidad del equipo de desarrollo, quien especificó la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explicó el por qué.

2.3.6 Realización de pruebas funcionales al software

Luego de haber analizado los valores válidos e inválidos que puede tomar cada variable asociada a los diferentes casos de uso, se comenzó a probar el software en los dos niveles definidos: nivel de integración y nivel de sistema. Para llevar a cabo el procedimiento de todo lo que se deseaba verificar en cada uno de los niveles de pruebas utilizados, se emplearon varias técnicas que ayudaron a hacer una mejor revisión y por consiguiente un mejor proceso de liberación, entre ellas, prueba funcional, prueba de interfaz de usuario y prueba de seguridad.

Prueba de integración

Para conseguir el objetivo que se deseaba alcanzar en este nivel de prueba se verificó el intercambio de datos de todos los módulos con los que cuenta el sistema por separado. De esta manera se intentó encontrar errores como funciones incorrectas o inexistentes, errores relativos a las interfaces, errores en estructuras de datos o en accesos a bases de datos externas, errores de inicialización y terminación, y cualquier inconsistencia que no estuviera acorde a lo especificado en los requisitos funcionales. En general se comprobó la funcionalidad y validez de todas las entradas de datos en cada módulo, con empleo de los diseños de casos de prueba.

Prueba de sistema

Luego de haber alcanzado que el intercambio de datos en los módulos fuese el más eficiente se decidió probar el sistema completo. Dentro de este tipo de prueba se analizó el software como un todo, comprobando que todos los módulos funcionaran correctamente de forma íntegra. Para probar estos aspectos en el software se usaron como entradas un conjunto de datos de pruebas para llevar a cabo el procesamiento de las operaciones y de esta manera examinar los resultados en los demás módulos, comprobando que existiera correspondencia entre ellos según lo ejecutado anteriormente.

Prueba funcional

La prueba funcional fue aplicada teniendo en cuenta los casos de uso, donde se ejecutó cada caso de uso, flujo de caso de uso o función, usando datos válidos y no válidos, para verificar aspectos como:

- Los resultados del sistema cuando se emplearon datos válidos fueron los esperados.
- El uso de datos no válidos o funciones desplegaron mensajes de error o advertencia apropiados.
- Se aplicaron apropiadamente cada regla del negocio.

Prueba interfaz de usuario

Para la aplicación de las pruebas de interfaz de usuario se realizó un manejo de ventanas y métodos de acceso a interfaces, para verificar de esta manera las características, tamaño, posición, estado, de

acuerdo a los estándares de cada uno de los campos. A través de un conjunto de pantallas se comprobó la facilidad de navegación de las mismas y su seguimiento según los estándares establecidos. Se verificó además mediante la navegación por todos los casos de uso que las interfaces fueran sencillas, se comprendieran fácilmente y se correspondieran con el tipo de usuario que estuviera autenticado en ese momento.

Prueba de seguridad

La prueba de seguridad se ejecutó teniendo en cuenta las probabilidades que tenía el usuario de acceder al sistema, ya sea manera correcta o ilegal.

- Para probar la seguridad en el ámbito de aplicación se identificó cada tipo de usuario y se hizo una lista con las funciones y los datos sobre las que cada uno de ellos tenía permiso.
- Se verificó le permiso de cada tipo de usuario ejecutando operaciones específicas para cada uno de ellos.
- Se modificó el tipo de usuario y se ejecutaron las operaciones del tipo anterior para el usuario modificado. En cada caso según la modificación, se verificó que las funciones o datos adicionales estuvieran o no disponibles para el nuevo tipo de usuario en el sistema.

2.3.7 Realización de pruebas de carga y estrés

Posterior a la ejecución las pruebas funcionales en el proceso de liberación, se aplicaron las pruebas de carga y estrés, como propuesta de otro tipo de prueba que se puede ejecutar dentro del proceso de liberación para ampliar la gama de probabilidades de encontrar errores de rendimiento en el software y verificar que funciona eficientemente bajo condiciones anormales. Para la realización de estas pruebas se utilizó la herramienta JMeter para la automatización de las mismas.

Las pruebas complejas fueron realizadas teniendo en cuenta los casos de uso arquitectónicamente significativos y dentro de estos los que más cargan implicaran en el sistema, influyendo de esta manera en el rendimiento del software debido a su complejidad. Se seleccionaron dos casos de uso por cada módulo, excepto en el módulo administración que se eligieron todos los que tiene asociado, para un total de 17 casos de uso seleccionados. A continuación se nombran cada uno de ellos:

Módulo Administración

- Autenticar.
- Gestionar Sitio.
- Gestionar Hospital.
- Gestionar Usuario.
- Gestionar Asignaciones.
- Gestionar Ensayo Clínico.
- Gestionar IP.

Módulo Cronograma

- Crear cronograma general.
- Gestionar planificación de un modelo.

Módulo Validación

- Validar Variable.
- Firmar validación.

Módulo Publicador

- Adicionar paciente.
- Gestionar query libre.

Módulo Reporte

- Realizar reporte por variables independientes.
- Realizar reporte por modelo.

Módulo Monitoreo

- Responder query.
- Monitorear variables de un modelo.

Para confeccionar el plan de prueba fue necesario adicionar un grupo de hilos en el que se tuvieron en cuenta los siguientes campos:

- Número de hilos: se definió la cantidad de usuarios que se van a simular en la prueba.
- Periodo de subida (en segundos): se especificó el periodo intermedio en segundos, en los cuales va a ir iniciándose cada uno de los usuarios.
- Contador del bucle: gracias al cual, se pudo realizar la simulación varias veces o especificarle sin fin.

El siguiente paso que se realizó fue la grabación de todos los pasos realizados en el software de manera que se generaran los escenarios de prueba de manera automática a través del elemento Servidor Proxy HTTP, el cual permitió arrancar la prueba para que se fueran organizando los resultados de la grabación. Para lograr esto se especificó en las opciones del navegador la dirección y el nuevo puerto que se utilizó para la grabación de los escenarios.

Una vez grabadas todas las peticiones http se utilizó una gama de listeners que sirvió para registrar los resultados de las pruebas, cada uno de ellos especializado en mostrar los resultados de las pruebas por aspectos y características diferentes, en correspondencias a las necesidades de la prueba. Se decidió agregar los listeners Informe Agregado y Árbol de Resultado porque se pueden ver el instante en el que se van cargando las peticiones HTTP de manera factible o no.

2.4 Evaluación del software basado en los atributos de calidad

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 llevar a cabo la evaluación del mismo se propone aplicar una lista de chequeo [Anexo 1] como mecanismo conducente y función básica para su valoración, la misma 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.

Los atributos de calidad son los componentes del servicio recibido que el cliente valora de forma especial y puede percibir con claridad por separado. El suministrador ha de investigar el diferente peso que el cliente asigna a la satisfacción o insatisfacción de cada uno de ellos. [11] Los principales componentes o atributos de la calidad que se le miden a un software son los siguientes:

- Fiabilidad: implica consistencia en la prestación del servicio. Ello significa que la empresa presta servicio correctamente en el momento preciso y que cumple su promesa.
- Usabilidad: grado en el que un producto puede ser utilizado por usuarios específicos para conseguir objetivos con efectividad, eficiencia y satisfacción en un determinado contexto de uso.
- Rapidez: se traduce en la capacidad de realizar el servicio dentro de los plazos aceptables para el cliente.
- Competencia: el personal debe poseer la información y la capacitación necesaria para la realización del servicio. Para ello debe estar bien formado.
- Cortesía: se expresa a través de la educación, la amabilidad y el respeto del personal hacia el cliente. La amabilidad puede adquirir carácter de estrategia comercial.
- Credibilidad: es la honestidad de la empresa de servicios tanto en sus palabras como en sus actos, como por ejemplo en plazos de entrega, tratamiento del pedido, garantía y servicio post-venta.
- Seguridad: ausencia de peligro, riesgo o dudas a la hora de utilizar el servicio.
- Accesibilidad: facilidad con que el consumidor puede utilizar el servicio en el momento que lo desee. El acondicionamiento de las secciones y unas señalizaciones más claras aumentan la comodidad para el cliente.
- Comunicación: se debe informar al consumidor con un lenguaje que éste entienda, para poder ayudarle a guiar su elección; exige escuchar y adaptarse a sus demandas
- Conocimiento del consumidor: se trata del esfuerzo realizado por la empresa para entender a los consumidores y sus necesidades.
- Responsabilidad: supone la disposición a proporcionar el servicio. Esta disposición debe hacerse patente, es decir, demostrar que la empresa se preocupa de los problemas de los clientes.
- Tangibles: son los elementos del servicio que pueden percibirse por los sentidos. Hay que incluir por tanto, evidencias físicas del servicio e indicios de su calidad, limpieza, aspecto personal, equipos utilizados, soporte físico del servicio, pequeños obsequios y otros.
- Confiabilidad: capacidad de desempeñar una función requerida, en condiciones establecidas.
- Portabilidad: 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.

La evaluación solo se centrará en cuatro de ellos:

- Usabilidad
- Seguridad
- Confiabilidad
- Portabilidad

2.5 Conclusiones

En este capítulo se expuso una breve descripción del sistema, donde se detallaron los casos de uso y los requerimientos funcionales del mismo. Se detalló la estrategia de trabajo llevada a cabo para la realización de las pruebas, donde se tuvo en cuenta algunos puntos claves para lograr el buen desarrollo de las mismas y su aplicación con la mayor calidad y se dejó plasmada la propuesta para la evaluación del software.

CAPÍTULO 3

Resultados de las pruebas de software aplicadas al SIMDECC

3.1 Introducción

En este capítulo se muestran y se analizan los resultados obtenidos en la documentación del sistema y en la aplicación luego de ejecutar las pruebas pertinentes en el proceso de liberación, y se da una evaluación al software teniendo en cuenta atributos de calidad.

En el proceso de liberación del software una de las tareas que se tuvo en cuenta por su importancia para elevar la calidad del producto, fue llevar el registro de los errores encontrados en las revisiones de la documentación del sistema y en las pruebas funcionales empleadas a la aplicación; además de controlar el seguimiento de los mismos. Como responsabilidad de este proceso está el estudio y análisis de los resultados obtenidos en cada una de las pruebas contra los resultados esperados, aspecto primordial que se tuvo en cuenta en la detección de no conformidades.

El valor que se le puede dar a un software en el mercado puede ser mayor o menor en dependencia de la respuesta de la dirección del proyecto a una no conformidad. Asegurando que el personal involucrado haya realizado satisfactoriamente la corrección, análisis de las causas y acción correctiva, aumenta la probabilidad de que el software resultante logre la satisfacción del cliente.

Para permitir que las no conformidades encontradas fueran resueltas de la manera correcta, se tuvo en cuenta que el grupo de proyecto que desarrolla el sistema asimilara e interiorizara el análisis realizado por el equipo de pruebas, debido a que de esto depende el mejoramiento del producto de software pero a la vez la formación de bases para el mejoramiento de aspectos relevantes en el proceso de desarrollo como: especificación de requerimientos de software, estandarización, control de configuración, etc.

Las pruebas fueron realizadas por dos probadores que no pertenecen al equipo de desarrollo del software, bajo diferentes condiciones ambientales y de trabajo que se caracterizaron como buenas y utilizando una máquina para cada uno con sistema operativo Windows XP Profesional, con 1 GB de RAM y microprocesador Pentium IV.

3.2 Análisis de los resultados obtenidos en la documentación del sistema

Una vez revisada la documentación del sistema se realizó una síntesis de los principales hallazgos en cada uno de los documentos revisados, los cuales de no haber sido resueltos imposibilitarían una precisa comprensión por parte de los clientes que impediría el entendimiento del sistema.

Por cada módulo se realizaron diferentes iteraciones, donde en cada plantilla con las no conformidades detectadas que se emitió por cada iteración, quedaron plasmadas de una manera organizada los errores encontrados por documentos, o sea, cada documento que se revisó consta de una tabla propia de él, con las no conformidades encontradas y su clasificación.

A continuación se mostrará de manera resumen una tabla con la cantidad de no conformidades encontradas en los documentos revisados y su clasificación; además del número de recomendaciones halladas en cada uno de ellos.

Módulo Administración			
	Cantidad de No Conformidades		
Iteración	Significativas	No Significativas	Recomendaciones
1	19	5	1
2	9	2	3
3	8	0	4
TOTAL	36	7	8

Tabla 1. Registro de las No Conformidades encontradas en la documentación del Módulo Administración

Módulo Cronograma			
	Cantidad de No Conformidades		
Iteración	Significativas	No Significativas	Recomendaciones
1	20	0	2
2	5	1	2
3	1	0	0
4	1	1	3
5	4	0	2
TOTAL	31	2	9

Tabla 2. Registro de las No Conformidades encontradas en la documentación del Módulo Cronograma

Módulo Monitoreo			
	Cantidad de No Conformidades		
Iteración	Significativas	No Significativas	Recomendaciones
1	9	0	1
2	5	0	2
3	1	0	3
4	2	0	3
TOTAL	17	0	9

Tabla 3. Registro de las No Conformidades encontradas en la documentación del Módulo Monitoreo

Módulo Publicador			
	Cantidad de No Conformidades		
Iteración	Significativas	No Significativas	Recomendaciones
1	5	0	2
2	3	0	1
3	7	0	5
4	2	0	2
TOTAL	17	0	10

Tabla 4. Registro de las No Conformidades encontradas en la documentación del Módulo Publicador

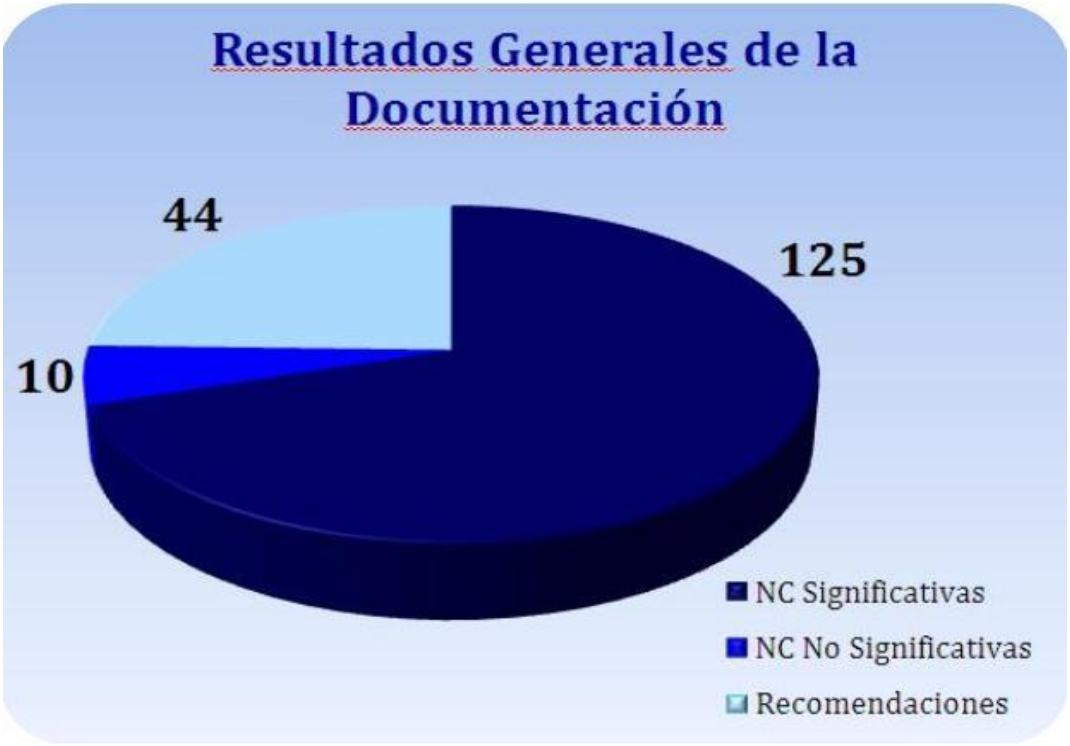
Módulo Reporte			
	Cantidad de No Conformidades		
Iteración	Significativas	No Significativas	Recomendaciones
1	4	0	0
2	8	0	4
3	2	0	4
TOTAL	14	0	8

Tabla 5. Registro de las No Conformidades encontradas en la documentación del Módulo Reporte

Módulo Validación			
	Cantidad de No Conformidades		
Iteración	Significativas	No Significativas	Recomendaciones
1	6	1	0
2	1	0	0
3	1	0	0
4	2	0	0
TOTAL	10	1	0

Tabla 6. Registro de las No Conformidades encontradas en la documentación del Módulo Validación

Después de haber visto cada una de las no conformidades por módulos e iteraciones se puede concluir diciendo que en la revisión de la documentación del sistema, se obtuvieron un total de 135 no conformidades, clasificadas en 125 significativas y 10 no significativas; y se obtuvo un total de 44 recomendaciones. A cada uno de estos errores encontrados se les dieron cumplimiento por parte del equipo de desarrollo. En la siguiente gráfica queda representado el resultado general.



Gráfica 1. Registro de los Resultados Generales de la Documentación

El principal problema en los documentos se basó en la falta de ortografía y falta de concordancia en los textos, lo que provocaba que el mensaje que se quería dejar no provocara un buen entendimiento, de aquí la mayoría de no conformidades emitidas. Existieron otros errores, por ejemplo, en el índice donde muchas veces la página asociada a un epígrafe no coincidía; y en los textos donde había en ocasiones falta de uniformidad en los mismos. Entre las recomendaciones la que más abundó fue la actualización del control de versiones y la tabla de contenido.

3.3 Análisis de los resultados obtenidos en la aplicación

La aplicación del sistema se revisó con mucha dedicación y cautela para encontrar la mayor cantidad de errores y evitar un incorrecto funcionamiento en las interfaces del sistema y en el software en general. En favorables condiciones de trabajo, con un ambiente aceptable y con los recursos materiales necesarios para aplicar las pruebas pertinentes al software, se llevó a cabo la ejecución de las mismas.

Los errores encontrados se deben a los resultados en las entradas de datos, los errores de interfaz y los resultados de las pruebas de requerimientos. La mayoría de los diseños de casos de prueba aplicados a cada caso de uso de los diferentes módulos con los que cuenta el sistema, arrojaron no conformidades, las cuales se resumen a continuación. La tabla expuesta reflejará la cantidad de iteraciones realizadas.

Módulo Administración			
	Cantidad de No Conformidades		
Iteración	Significativas	No Significativas	Recomendaciones
1	9	13	4
2	0	0	0
TOTAL	9	13	4

Tabla 7. Registro de las No Conformidades encontradas en la aplicación del Módulo Administración

Módulo Cronograma			
	Cantidad de No Conformidades		
Iteración	Significativas	No Significativas	Recomendaciones
1	2	9	0
2	0	0	0
TOTAL	2	9	0

Tabla 8. Registro de las No Conformidades encontradas en la aplicación del Módulo Cronograma

Módulo Monitoreo			
	Cantidad de No Conformidades		
Iteración	Significativas	No Significativas	Recomendaciones
1	4	1	0
2	2	0	0
3	0	0	0
TOTAL	6	1	0

Tabla 9. Registro de las No Conformidades encontradas en la aplicación del Módulo Monitoreo

Módulo Publicador			
	Cantidad de No Conformidades		
Iteración	Significativas	No Significativas	Recomendaciones
1	3	8	0
2	0	0	0
TOTAL	3	8	0

Tabla 10. Registro de las No Conformidades encontradas en la aplicación del Módulo Publicador

Módulo Reporte			
	Cantidad de No Conformidades		
Iteración	Significativas	No Significativas	Recomendaciones
1	0	17	0
2	0	9	0
3	0	0	0
TOTAL	0	26	0

Tabla 11. Registro de las No Conformidades encontradas en la aplicación del Módulo Reporte

Módulo Validación			
	Cantidad de No Conformidades		
Iteración	Significativas	No Significativas	Recomendaciones
1	1	41	0
2	2	30	0
3	0	0	0
TOTAL	2	71	0

Tabla 12. Registro de las No Conformidades encontradas en la aplicación del Módulo Validación

Una vez visto la cantidad de errores encontrados en las revisiones realizadas a la aplicación y la clasificación de las mismas, además de las recomendaciones que se le hicieron, se puede resumir diciendo que a pesar de no estar caracterizadas como catastróficas sí podían traer como consecuencia una mala aceptación por parte del cliente. Por esta razón, se hizo mucho hincapié en los procedimientos que generaran fallas y no encontraran las salidas esperadas o datos mostrados incorrectamente, los cuales podían atentar contra la confiabilidad y funcionalidad del sistema.

La aplicación en general estaba bien implementada, la mayoría de las funcionalidades que requería el sistema cumplían su objetivo. Las no conformidades más importantes que arrojaron los diseños de casos de prueba fueron:

- Falta de botones o hipervínculos que permitieran salir de la página en la que se encontraba el usuario al realizar alguna acción indebida.
- Problemas en la búsqueda avanzada.
- Usuarios insertados con la misma dirección de correo.
- Interfaces de usuarios incorrectamente validadas.
- Mensajes mostrados en situaciones que no correspondían con lo señalado en el mismo.
- Información incorrecta en la guía de usuarios con respecto a lo que realmente brindaba la interfaz.
- Ausencia de la guía de usuarios en algunas interfaces.

En cuanto a la interfaz, los principales errores clasificados como no significativos, estuvieron dados por faltas de ortografía que presenta en todos sus módulos, incluyendo los mensajes mostrados por el sistema. No son errores de una importancia alta que conlleven a una incorrecta ejecución del programa o a un mal funcionamiento del mismo, pero que sí afectan la calidad integral y la estética del software. La recomendación que más abundó estuvo dada en el mejoramiento de los enfoques de los mensajes y en las abreviaturas utilizadas para un mejor entendimiento y comprensión del usuario.

De manera general los resultados que arrojaron las pruebas a la aplicación se pueden ver en la siguiente gráfica:

Gráfica 2. Registro de los Resultados Generales de la Aplicación

3.4 Análisis de los resultados obtenidos en las pruebas de carga y estrés

Los resultados de las pruebas de carga y estrés fueron arrojados teniendo en cuenta algunas especificaciones de recursos, los cuales son imprescindibles en este tipo de pruebas.

Algunos de los requerimientos que el equipo de desarrollo exigió para el servidor de aplicación fueron los siguientes:

- ✓ Servidor Web con alta disponibilidad y rendimiento adecuado, garantizado por al menos un procesador Dual Intel Xeon 3 GHz o similar y RAM suficiente (4 GB a 8 GB).
- ✓ Servidores de almacenamiento de datos de 1 a 3 TB disponibles, pues el volumen de información es bastante grande y perdura en el tiempo hasta 15 años.
- ✓ Lenguaje de programación PHP y Gestor de Base de Datos Postgre-SQL.

- ✓ Sistema operativo Windows o GNU Linux.
- ✓ Debe tener instalado el servidor web Apache 2.0.
- ✓ Servidor de aplicación capaz de soportar un aumento de usuarios concurrentes por minuto de 1 a 400.

Sin embargo la aplicación se encontraba instalada en un servidor con las siguientes características:

- ✓ PC con Microprocesador Pentium IV a 2.00 GHz.
- ✓ 1 GB de RAM.
- ✓ 160 GB de Disco Duro.
- ✓ Conexión TCP/IP.
- ✓ Lenguaje de programación PHP versión 5.1.6 y Gestor de Base de Datos Postgre-SQL.
- ✓ Servidor web Apache 2.0.
- ✓ Sistema operativo Windows o GNU Linux.

Las pruebas de carga y estrés se ejecutaron utilizando una PC con sistema operativo Microsoft Windows XP Profesional, con 1 GB de memoria RAM y microprocesador Pentium IV, en la cual fue necesario instalar la máquina virtual de Java para la ejecución de la herramienta JMeter. Las pruebas fueron realizadas por dos probadores que no pertenecen al equipo de desarrollo del sistema.

Se realizaron pruebas complejas mezclando las grabaciones hechas al sistema de las peticiones HTTP de todos los módulos. Las pruebas se iniciaron realizando 2 iteraciones con un tiempo de subida de 1 segundo cada una: una primera iteración simulando 10 usuarios en 40 bucles y una segunda iteración simulando 100 usuarios en 4 bucles. Esto quiere decir que se simularon un total de 400 usuarios.

Los resultados comprendieron una serie de variables obtenidas en el Informe Agregado y el Árbol de Resultados como consecuencia de la ejecución de estas pruebas, las cuales se encuentran reflejadas en las siguientes figuras donde se relacionan algunos parámetros que se creen válido hacer referencia de forma general, tales como:

- ➢ Peticiones HTTP: cantidad de peticiones HTTP realizadas al servidor.
- ➢ Muestras: número de muestras para cada URL.
- ➢ Media: media de tiempo total que demoraron las peticiones en cargarse.

- Mediana: tiempo promedio que han tardado en cargarse las páginas.
- 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.
- Min: tiempo mínimo que ha demorado en cargarse una página.
- Max: tiempo Máximo que ha tardado en cargarse una página.
- %Error: por ciento de error de las páginas que no se llegaron a cargar de manera satisfactoria.
- Rendimiento: total del tiempo que demoró en cargarse la cantidad de hilos de esa prueba.
- Kb/Sec: velocidad de carga de las páginas.

Primera iteración (10 usuarios en 40 bucles)

Label	# Muestras	Media	Mediana	Linea de 90 %	Min	Máx	% Error	Rendimiento	KB/Sec
Total	20446	6599	6453	12328	0	47437	2.32	8.1/sec	55511.4

Figura 1. Comportamiento general de las Pruebas Complejas en el Informe Agregado en la primera iteración

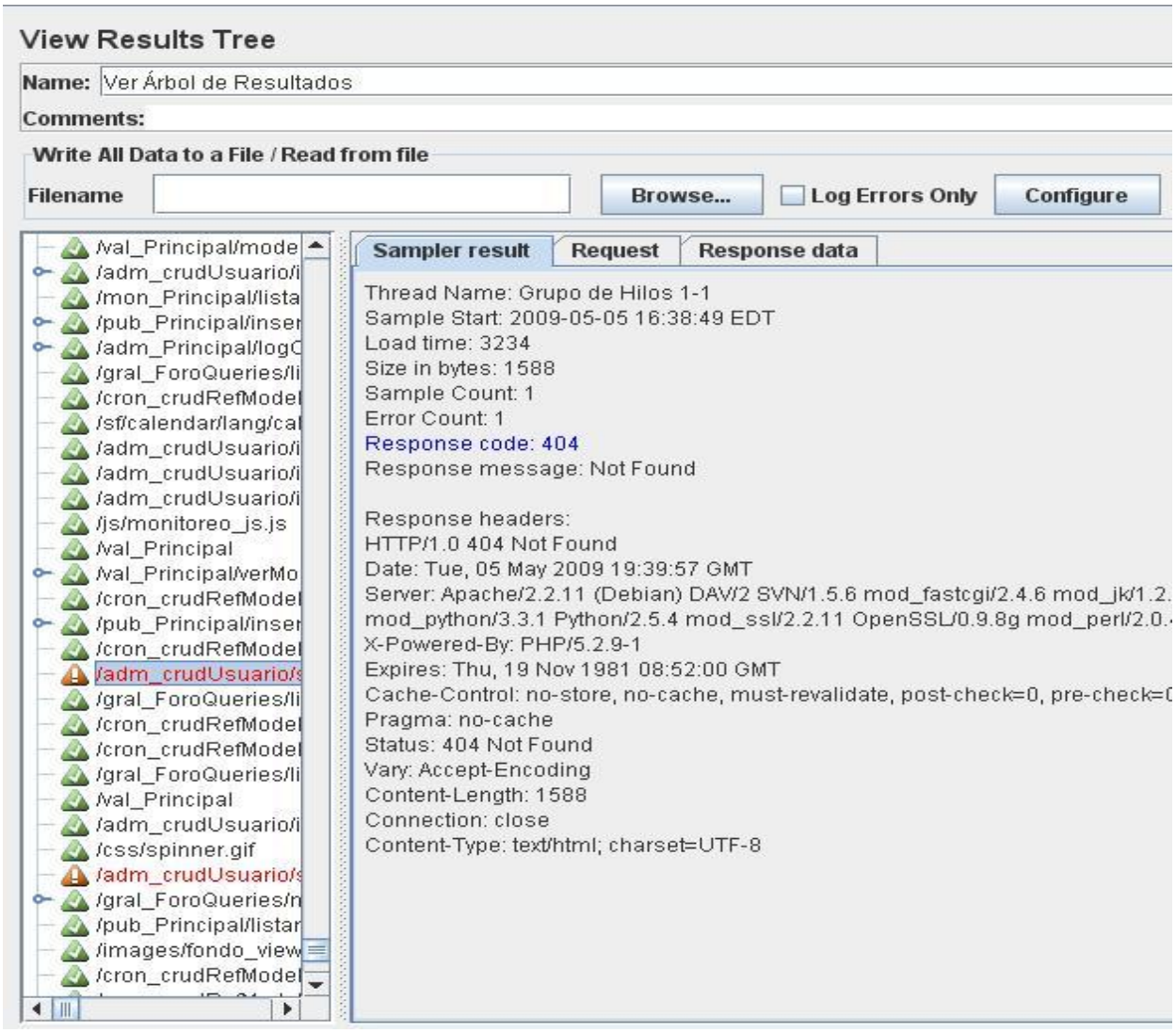


Figura 2. Resultado del Muestreador en el Árbol de Resultados en la primera iteración

Segunda iteración (100 usuarios en 4 bucles)

Label	# Muestras	Media	Mediana	Linea de 90 %	Min	Máx	% Error	Rendimiento	KB/Sec
Total	1151	30413	13875	67687	0	493844	0.78	29.5/min	3982.7

Figura 3. Comportamiento general de las Pruebas Complejas en el Informe Agregado en la segunda iteración

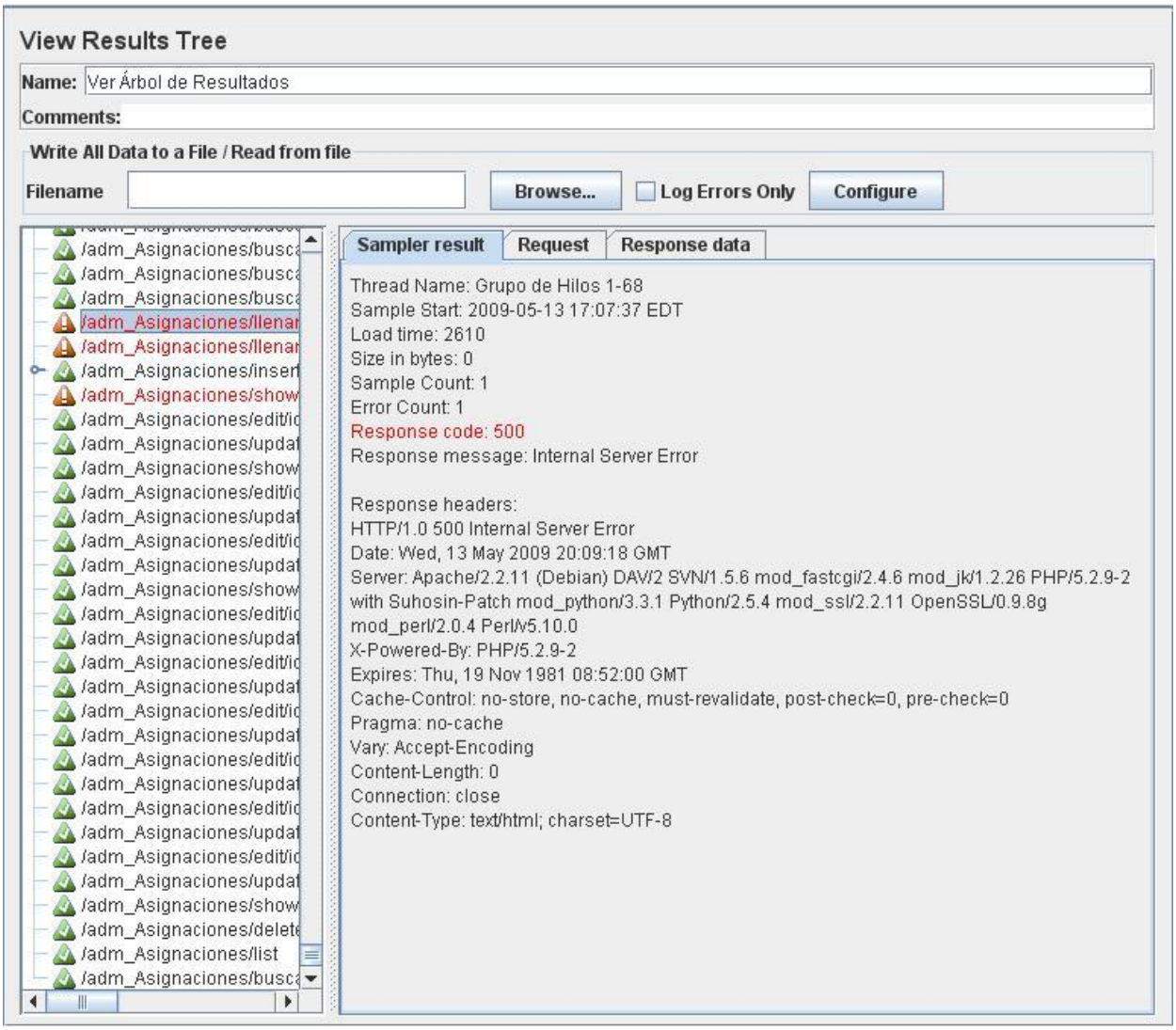


Figura 4. Resultado del Muestreador en el Árbol de Resultados en la segunda iteración

Después de haber observado los resultados generales de las dos iteraciones realizadas, se llegó a la conclusión de que hubo una buena conexión en la mayoría de las páginas previstas para cargarse. Solo algunas peticiones HTTP realizadas no tuvieron su debida conexión, como aparece reflejado en las figuras. Las causas por las que estas páginas no se cargaran satisfactoriamente estuvieron dadas por el servidor de la máquina, el cual es de pocas prestaciones y el mismo se utiliza para varios servicios, lo que puede traer como consecuencia las siguientes inconsistencias:

- Ocupación total de la memoria de la máquina

- Múltiples pedidos en 1 segundo
- Restricciones de conexión en el servidor

Los resultados obtenidos también manifestaron que mientras menor fuera el número de usuarios finales conectándose a la aplicación en subidas de segundo, mayor sería el número de muestras realizadas por cada URL y menor el tiempo promedio en cargarse las peticiones realizadas. El tiempo promedio que tardaron en cargarse las páginas fue superior en la segunda iteración debido a la mayor cantidad de usuarios conectados al mismo tiempo y mayor el tiempo en correr el 90 por ciento de las peticiones reales. El por ciento de error en las dos iteraciones estuvo dado por las páginas que no se llegaron a cargar satisfactoriamente. El rendimiento también dependió en gran medida de la cantidad de usuarios que se estuvieran conectando, pues en la segunda iteración el total del tiempo que demoró en cargarse la cantidad de hilos de la prueba, perduró más que en la primera con menor velocidad de carga de las páginas.

Luego de haber ejecutado estas pruebas se decidió realizar una tercera iteración para identificar el soporte máximo que el software puede alcanzar y de esta manera comprobar si sustenta más usuarios finales de los que debe soportar. Esta iteración se realizó simulando 150 usuarios en 4 bucles en una subida de tiempo de un segundo para una simulación de 600 usuarios finales en total. La ejecución de esta prueba ocasionó algunos problemas dejando la herramienta detenida con solamente 20 usuarios conectados y la máquina sin uso. Luego de un largo tiempo de ejecución de la prueba el resultado seguía siendo el mismo:

Tercera iteración (150 usuarios en 4 bucles)

Label	# Muestras	Media	Mediana	Linea de 90 %	Min	Máx	% Error	Rendimiento	KB/Sec
Total	306	1493	390	3656	0	28140	3.59	1.7/sec	9643.7

Figura 5. Comportamiento general de las Pruebas Complejas en el Informe Agregado en la tercera iteración

Los valores obtenidos en cada uno de los campos que brindó el Informe Agregado fueron mínimos en comparación con las otras iteraciones realizadas debido a que no llegó a la cantidad máxima de

usuarios finales que debían haberse conectado, llegando a la conclusión de que el sistema no soporta una cantidad mayor a 400 usuarios.

3.5 Evaluación del software

Una vez realizado el proceso de liberación del software se llevó a cabo un proceso de control para realizar un levantamiento y evaluación del sistema teniendo en cuenta diferentes atributos de calidad. Para efectuar dicho proceso se utilizó una lista de chequeo como mecanismo conducente al control de riesgos y por la importancia que tiene esta herramienta en la detección de peligros y fallas, la cual estuvo centrada en los cuatro atributos de calidad siguientes:

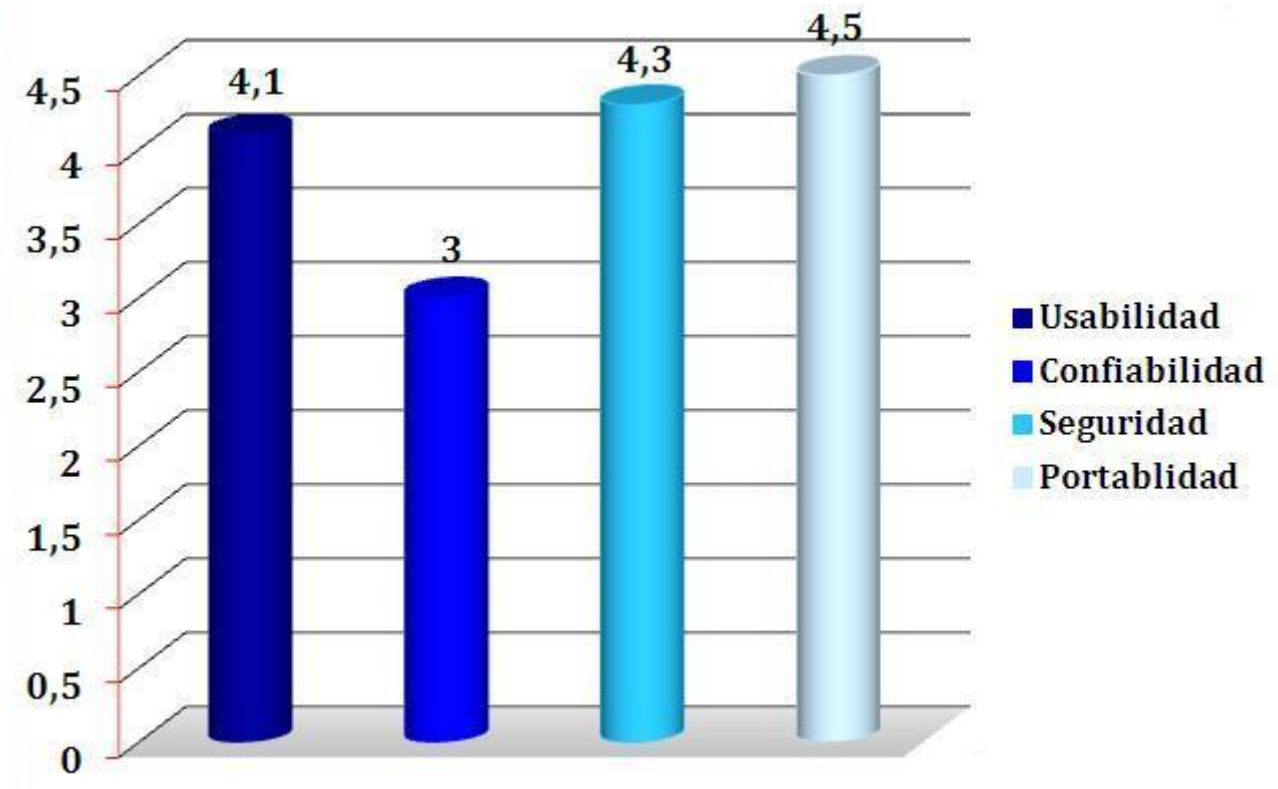
- Usabilidad
- Seguridad
- Confiabilidad
- Portabilidad

El principal objetivo de esta medición fue conocer la percepción sobre cada uno de los atributos de calidad de forma cualitativa y cuantitativa para poder tomar acciones certeras sobre las correspondientes características del software.

La lista de chequeo para la evaluación del software [Anexo 1] estaba constituida por varios campos que son los que permiten dejar plasmado de manera específica, el registro de las estimaciones asociadas a cada interrogante. Teniendo en cuenta estas interrogantes definidas para cada atributo de calidad a evaluar, se realizó la evaluación dándole a cada pregunta una puntuación de 0 a 5 puntos. Aquellas que tuvieron una calificación de 0 a 2 puntos se detalló el por qué en el campo comentario.

Una vez concluida la lista de chequeo se prosiguió a realizar una evaluación cuantitativa de estos resultados. Para conseguir un promedio de puntuación por cada atributo de calidad se sumaron todos los parámetros referentes a cada uno de estos atributos y se dividieron por la cantidad de interrogantes realizadas a cada uno de ellos, ubicándolos en una escala de 0 a 5 puntos. La usabilidad en el sistema alcanzó un total de 4,1 puntos; la confiabilidad 3 puntos; la seguridad 4,3 puntos; mientras que la

portabilidad adquirió la puntuación más alta con un valor de 4,5 puntos, como se muestra en la siguiente gráfica:



Gráfica 3. Evaluación del software

A partir de los resultados anteriores se analizó cómo se encuentra el sistema atendiendo a estos atributos de calidad de forma cualitativa. Para la evaluación se tuvo en cuenta un rango de puntuación que ayudó a conocer la eficiencia y validez del software:

- 0 – 3: No disponible - Mal
- 3,1 – 4: Parcialmente disponible - Bien
- 4,1 – 5: Disponible – Excelente

Teniendo en cuenta este rango y la cantidad de puntos alcanzados en los atributos de calidad, se llegó a la conclusión de que el sistema es usable ya que resulta fácil de manejar y con funcionalidades e interfaces entendibles para aquellos usuarios específicos que accedan al sistema, en cuanto a la confiabilidad es parcialmente confiable porque la probabilidad de que el sistema realice su objetivo

satisfactoriamente, en un determinado tiempo y en un entorno concreto no es la más alta; el sistema resultó ser seguro cumpliendo con los accesos y permisos a los diferentes tipos de usuarios; y además se caracterizó por ser portable ya que puede ser utilizado en diferentes plataformas.

3.6 Conclusiones

Los resultados obtenidos dentro del proceso de pruebas que se llevó a cabo mostraron una serie de errores en el sistema, que sirvieron para caracterizarlo y llegar a una conclusión de su puesta en práctica. Debido a que todos los errores encontrados en la documentación del sistema y en la aplicación se les dieron cumplimiento se puede afirmar que el software está en condiciones de ser utilizado, ya que cumple con todas las funcionalidades previstas por los requerimientos funcionales; sin embargo no es óptimo por los resultados arrojados por las pruebas de carga y estrés, los cuales no fueron muy satisfactorios.

CONCLUSIONES

Con el desarrollo de la industria de software, y la necesidad de que los productos realizados sean confiables y precisos, crece la exigencia por parte de clientes y usuarios finales en general, de que a los sistemas se les hayan aplicado las técnicas de Ingeniería de Software adecuadas, y en su etapa de comprobación hayan sido probados con la técnicas necesarias para lograr el nivel de calidad requerido.

Es por ello que resulta vital que se tenga en cuenta como parte importante del proceso de liberación de sistemas, velar por que la calidad de los mismos, esté a la altura del desarrollo que se exige en estos tiempos. Al planificarse y ejecutarse el proceso de liberación desarrollado se llegaron a las siguientes conclusiones:

- Se revisó toda la documentación del sistema resultando ser clara y entendible; la cual servirá de apoyo a los usuarios finales a la hora de utilizar el sistema.
- Se diseñaron todos los casos de prueba que cubrieron las funcionalidades del sistema, logrando con estos probar los requerimientos funcionales con los que cuenta el software.
- Se logró obtener un software con la menor cantidad de errores posibles y una elevada calidad, debido a que las pruebas ejecutadas durante el proceso de liberación resultaron ser eficientes.
- Se analizaron los resultados de todas las pruebas realizadas dando una valoración de cada uno de ellos que sirvió para evaluar al software a través de diferentes atributos de calidad.

Con el estudio realizado y la aplicación de las pruebas que se han llevado a cabo dentro del proceso de liberación se ha cumplido el objetivo propuesto de desarrollar un proceso de pruebas de liberación de software al SIMDECC, con vistas a la obtención del mayor número de faltas existentes en la misma, logrando con su ulterior eliminación obtener una aplicación con el mínimo de errores y por consiguiente mayor calidad.

RECOMENDACIONES

Después de concluido el proceso de prueba y conocer las facilidades que brinda a los desarrolladores de los sistemas una adecuada detección de errores, en el momento oportuno, se recomienda:

- Incluir las pruebas de carga y estrés al proceso de pruebas de liberación.
- Que la(s) persona(s) encargada(s) de desempeñar el rol de prueba dentro del equipo de desarrollo realice todas las pruebas pertinentes al rol que le ocupa.

REFERENCIAS BIBLIOGRÁFICAS

1. Nieves Borrero, Martha; Echeverría Perez, Delvis; Calzadilla Díaz, Roig; Góngora Rodríguez, Asnier E. *Definición y Automatización de los procesos de Pruebas de Software en un Laboratorio de Calidad.* Universidad de las Ciencias Informáticas, Ciudad de La Habana, 2007.
2. Reyes Brito, Irina; Napal Torrez, Irina. *Las pruebas de software, su aplicación al Config. CASE.* TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO INFORMÁTICO, INSTITUTO SUPERIOR POLITÉCNICO JOSÉ ANTONIO ECHEVERRÍA, Ciudad de La Habana, 2003.
3. Roger S. Pressman: *"Software Engineering: A Practitioner's Approach"* (European Adaptation), McGrawHill. 2000. ISBN: 0077096770. Disponible en: http://bibliodoc.uci.cu/pdf/reg02689.pdf
4. Saavedra López, Lesdey; Pupo Blez, Yohandris. *Diseño y aplicación de pruebas al producto Registro Cubano de Discapacitados.* Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, Ciudad de La Habana, 2007.
5. Pruebas. Disponible en: http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/pruebas_d.php
6. Alejandro Teruel. Universidad Simón Bolívar, 2001. Disponible en: http://www.ldc.usb.ve/~teruel/ci4713/clases2001/planPruebas.html
7. TÉCNICAS DE EVALUACIÓN DINÁMICA. Disponible en: http://www.lsi.us.es/docencia/get.php?id=361
8. Betancourt Rodríguez, Keyly; Rodríguez Martell, Frank. *Diseño de pruebas de calidad para producto LIMS control de calidad del CIGB.* Trabajo de Diploma, Facultad de Bioinformática, Universidad de las Ciencias Informáticas, Ciudad de La Habana, 2007.
9. Johanna Rojas; Emilio Barrios. Métodos de prueba de caja negra, 2007. Disponible en: http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html
10. CASOS DE PREUBA CAJA BLANCA. Disponible en: http://fcqi.tij.uabc.mx/docentes/luisgmo/data/8.2%20prb-cal-mant.pdf
11. Jose A. Pérez Fdez. de Velasco. GESTION DE LA CALIDAD EMPRESARIAL. CALIDAD EN LOS SERVICIOS Y ATENCION AL CLIENTE CALIDAD TOTAL. Disponible en: http://books.google.com.cu/books?id=2ibhVMNE_EgC&pg=PA106&lpg=PA106&dq=Los+atributos+de+calidad+son+los+componentes+del+servicio+recibido+que+el+cliente+valora+de+forma+especial+y+puede+percibir+con+claridad+por+separado.&source=bl&ots=4aWvCIlrth&sig=gcj

7t6FTC3WjMGhiSNzM_5uAGPY&hl=es&ei=n3vsSff-EpSJtgfv-O3JBQ&sa=X&oi=book_result&ct=result&resnum=1#PPP1,M1

BIBLIOGRAFÍA

1. Nieves Borrero, Martha; Echeverría Perez, Delvis; Calzadilla Díaz, Roig; Góngora Rodríguez, Asnier E. *Definición y Automatización de los procesos de Pruebas de Software en un Laboratorio de Calidad.* Universidad de las Ciencias Informáticas, 2007.
2. Reyes Brito, Irina; Napal Torrez, Irina. *Las pruebas de software, su aplicación al Config. CASE.* TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO INFORMÁTICO, INSTITUTO SUPERIOR POLITÉCNICO JOSÉ ANTONIO ECHEVERRÍA, 2003.
3. Roger S. Pressman: *"Software Engineering: A Practitioner's Approach"* (European Adaptation), McGrawHill. 2000. 5ta Edición. Disponible en: http://bibliodoc.uci.cu/pdf/reg02689.pdf
4. Saavedra López, Lesdey; Pupo Blez, Yohandris. *Diseño y aplicación de pruebas al producto Registro Cubano de Discapacitados.* Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, 2007.
5. EVA, Universidad de las Ciencias Informáticas. Disponible en: http://teleformacion.uci.cu/
6. Alejandro Teruel, 2001. Disponible en: http://www.ldc.usb.ve/~teruel/ci4713/clases2001/planPruebas.html
7. JBOM, 2001. Disponible en: http://www.angelfire.com/my/jimena/ingsoft/guia9.htm
8. Pruebas. Disponible en: http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/pruebas_d.php
9. AGAPEA libros urgentes, 2002. Disponible en: http://www.agapea.com/libros/Calidad-de-Sistemas-Informaticos-isbn-8478977341-i.htm
10. Roberto Hugo Vázquez. Taller de Calidad de Software, Introducción a la Calidad del Software, 2006. Disponible en: http://gridtics.frm.utn.edu.ar/docs/Introduccion%20a%20la%20Calidad%20de%20Software%20Vazquez.pdf
11. Programa de Formación en Ingeniería de Software. Disponible en: http://botero.pg.ing.ula.ve/moo/
12. TÉCNICAS DE EVALUACIÓN DINÁMICA. Disponible en: http://www.lsi.us.es/docencia/get.php?id=361
13. Betancourt Rodríguez, Keyly; Rodríguez Martell, Frank. *Diseño de pruebas de calidad para producto LIMS control de calidad del CIGB.* Trabajo de Diploma, Facultad de Bioinformática, Universidad de las Ciencias Informáticas, Ciudad de La Habana, 2007.

14. Guía Web, 2004. Disponible en: http://www.guiaweb.gob.cl/guia/capitulos/tres/experiencia.htm#t03usabilidad

15. Breves notas sobre la Medición de los Atributos Externos del Software. Disponible en: http://www.sc.ehu.es/jiwdocoj/mmis/externas.htm

16. net**mind.** Disponible en: http://vico.org/aRecursosPrivats/UML_TRAD/Testing/MPS_1_Introduccion_OK.pdf

17. Técnicas de prueba. Disponible en: http://indalog.ual.es/mtorres/LP/Prueba.pdf

18. CASOS DE PRUEBA CAJA BLANCA. Disponible en: http://fcqi.tij.uabc.mx/docentes/luisgmo/data/8.2%20prb-cal-mant.pdf

19. Johanna Rojas; Emilio Barrios. Métodos de prueba de caja negra, 2007. Disponible en: http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html

20. CASOS DE PREUBA CAJA BLANCA. Disponible en: http://fcqi.tij.uabc.mx/docentes/luisgmo/data/8.2%20prb-cal-mant.pdf

21. Jose María Torbio Vicente. Tutorial JMeter, 2005. Disponible en: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=jmeter

22. TestNG. Versión: 5.9. Creado: 27 de Abril de 2004, modificado 31 de Marzo de 2009. Disponible en: http://testng.org/doc/index.html

23. Javier Tuya. Las Pruebas del Software, 2007. Disponible en: http://in2test.lsi.uniovi.es/repris/actividades/TestingCadiz20070309.pdf

24. Grupo de investigación Arquitecturas de Software. Disponible en: http://www.udistrital.edu.co/comunidad/grupos/arquisoft/index.php?id=40&type=1

25. Jose A. Pérez Fdez. de Velasco. GESTION DE LA CALIDAD EMPRESARIAL. CALIDAD EN LOS SERVOCIOS Y ATENCION AL CLIENTE CALIDAD TOTAL. Disponible en: http://books.google.com.cu/books?id=2ibhVMNE_EgC&pg=PA106&lpg=PA106&dq=Los+atributos+de+calidad+son+los+componentes+del+servicio+recibido+que+el+cliente+valora+de+forma+especial+y+puede+percibir+con+claridad+por+separado.&source=bl&ots=4aWvCIlrth&sig=gcj7t6FTC3WjMGhiSNzM_5uAGPY&hl=es&ei=n3vsSff-EpSJtgfv-O3JBQ&sa=X&oi=book_result&ct=result&resnum=1#PPP1,M1

26. Ortega, M.; Pérez, M. y Rojas, T. UN ENFOQUE SISTÉMICO PARA EVALUAR EL PRODUCTO DE SOFTWARE. Disponible en: http://www.lisi.usb.ve/publicaciones/02%20calidad%20sistemica/calidad_27.pdf

27. Casos de Prueba. Disponible en: http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/prod_des/documento/casos_prueba_d.php

28. Liudmila Sánchez Almenares. Cómo Realizar Pruebas de Carga y Estrés en JMeter. Manual, 08/05/2008, UCI.

ANEXOS

Anexo 1

<Lista de chequeo>
<Especialista en Calidad de Software>

Versión <1.0>

Tabla de Contenidos

1. **Introducción**

1.1 Propósito y objetivos
1.2 Alcance
1.3 Definiciones, Acrónimos y Abreviaturas

Nivel:
Eval:
NP:
Comentario:

Las evaluaciones serán:

El peso de la pregunta será:

1.4 Referencias
1.5 Resumen

Nivel	Evaluación	Eval.	NP	Comentario

Especialista _____ Fecha: _____

Cargo: _____

Anexo 2

<Nombre del Proyecto>

Módulo <Nombre del Módulo>

Versión 1.1

No Conformidades Detectadas

Elemento de configuración: <Nombre del Elemento>

Versión 1.1

Control de versiones:

Fecha	Versión	Descripción	Autor

Tabla de Contenido

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa

3. Recomendaciones

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección

4. Anexos

Anexo 3

<Nombre del Proyecto>

Módulo <Nombre del Módulo>

Versión del proyecto

Diseño de Casos de Prueba

Nombre del caso: <Nombre del caso de uso>

Versión del CP

Control de versiones:

Fecha	Versión	Descripción	Autor

Tabla de contenido

Descripción General

Condiciones de Ejecución:

1. Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central

1.1 Descripción de variable.

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción

1.2 SC 1: <Sección #1 a revisar>

Id del escenario	Escenario	Variable 1	Variable 2	Variable N	Respuesta del Sistema	Resultado de la Prueba

2. Registro de defectos y dificultades detectados

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo

3. Anexos

GLOSARIO DE TÉRMINOS

B

- Biomoléculas: son moléculas que solamente se encuentran en la materia viva, se clasifican en glúcidos, lípidos, proteínas y ácidos nucleicos; el conjunto de todos ellos constituye lo que se llama materia orgánica.

C

- Cuaderno de Recogida de Datos (CRD): es un formulario diseñado para anotar las variables recogidas durante un ensayo clínico.
- Casos de uso (CU): Conjunto de secuencia de acciones que un sistema ejecuta y que produce un resultado observable para un actor.
- Complejidad ciclomática: es el número de caminos ejecutables en el código fuente.

E

- Ensayos clínicos (EC): es un estudio que permite a los médicos determinar si un nuevo tratamiento, medicamento o dispositivo contribuirá a prevenir, detectar o tratar una enfermedad.

F

- Fármacos: producto químico que se emplea en el tratamiento, diagnóstico o prevención de enfermedades.
- Fallas: error en el funcionamiento que emite el sistema.
- Framework: simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener.

G

- Gigabyte (GB): unidad de medida informática.

H

- HTTP: Hypertext Transfer Protocol (Protocolo de transferencia de hipertexto), es el protocolo usado en cada transacción de la Web (WWW).

I

- Iteración: Es un ciclo repetitivo de un proceso.

L

- Lista de chequeo: (cheks-list) a un listado de preguntas, en forma de cuestionario que sirve para verificar el grado de cumplimiento de determinas reglas establecidas a priori con un fin determinado.

M

- Módulo: es una parte de un programa de ordenador. De las varias tareas que debe realizar un programa para cumplir con su función u objetivos, un módulo realizará una de dichas tareas (o quizá varias en algún caso).

O

- Opensource: código abierto es el término con el que se conoce al software distribuido y desarrollado libremente.

Q

- Queries: consulta que se le hace a una tabla de una base de datos.

R

- Requisitos funcionales: son las capacidades o condiciones que el sistema tiene que cumplir para su correcto funcionamiento.
- RAM: Random Access Memory (memoria de acceso aleatorio) es la memoria desde donde el procesador recibe las instrucciones y guarda los resultados. Es el área de trabajo para la mayor parte del software de un computador.

S

- Software: es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.

- Stakholders: Personas u organizaciones que están activamente implicadas en el negocio ya sea porque participan en él o porque sus intereses se ven afectados con los resultados del proyecto.
- Servidor web: almacena documentos HTML, imágenes, archivos de texto, escrituras, y demás material Web compuesto por datos (conocidos colectivamente como contenido), y distribuye este contenido a clientes que la piden en la red.
- Servidores: es una computadora que, formando parte de una red, provee servicios a otros denominados clientes.

T

- Terabyte (TB): es una unidad de medida de almacenamiento de datos.

U

- Usuarios: Persona que utiliza normalmente el software.
- URL: Uniform Resource Locator, es decir, localizador uniforme de recurso y se refiere a la dirección única que identifica a una página web en Internet.