Universidad de las Ciencias Informáticas

Facultad 6

Trabajo de diploma para optar por el título de Ingeniero

En Ciencias Informáticas.

Tema de la investigación: Diseño y aplicación de pruebas

al Sistema LiberMaps.

Autora: Liudmila Betancourt Rivero

Tutor (es):

Ing. Liuba M. Infante Infante

Ing. Roig Calzadilla Díaz

Ciudad de La Habana,

¨Año Del 53 Aniversario de la Revolución¨

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo de diploma y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de su beneficio.

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

Autora:

Liudmila Betancourt Rivero.

_____

Tutor (es):

Ing. Liuba María Infante

_____

Ing. Roig Calzadilla Díaz

_____

Dedicatoria.

A Dios porque como lo prometió en su palabra, siempre estuvo conmigo.

A mi esposo por su amor y comprensión.

A mis padres por sus grandes esfuerzos, amor y preocupación.

Agradecimientos.

Agradezco a Dios por su amor, por darme aliento con sus palabras cuando más lo necesité y por su gracia y misericordia para conmigo.

A mi esposo, por darme apoyo, amor y palabras de aliento cuando me sentía estresada o desanimada, y por compartir conmigo cada día de su vida.

A mi mamá, porque fue mujer, amiga y madre al mismo tiempo. Porque aprendió a darme aliento cuando mis notas no eran las que ella esperaba. Me inculcó valores como la responsabilidad y dedicación. Me inspiró a ser lo que hoy soy, me apoyó en cada una de las decisiones importantes que tomé en mi vida y me dio la mejor vida que pueda tener un ser humano.

A mi papá, por ser tan especial y carismático. Por sus llamadas que llenaban mi vida de alegría y me inspiraba a seguirme esforzando porque la meta estaba cerca. Nunca olvidé sus palabras: "Estudia, estudia y estudia, cuando te canses descansa, y luego sigue estudiando". Por su amor, por apoyarme en las decisiones importantes que he tomado en mi vida, por darme todo lo necesario, y porque pude sentir su amor en cada viaje que hizo para verme y ver mis profesores cuando tenía problemas de docencia. El y mami hicieron todo lo que yo esperaba de ellos.

A todos mis familiares, que siempre se han preocupado por mí.

A mis pastores Norges, Sonia, Lourdes, Ortega y al Obispo Ricardo por sus enseñanzas y oraciones de apoyo.

A mis hermanos de la UCI, por ser personitas especiales que siempre llevaré en mi corazón, especialmente a Guille, Maridalia, Maryanis, Lisandra, Jose Carlos, Ángel.

A Lisandra Hernández por su amistad tan sincera, por sus consejos y por su amor.

A Taimé por su cariño, amor, consejos y enseñanzas.

A los hermanos de mi iglesia en San Luis, por sus oraciones.

A Dayron Hernández por ser un verdadero amigo, por su amor, sus consejos, por cada detalle, por su cariño. Gracias Tito.

A la Revolución cubana, por la oportunidad que le brinda a jóvenes como yo, de hacer su sueño realidad.

Así que, hermanos míos amados, estad firmes y constantes, creciendo en la obra del Señor siempre, sabiendo que vuestro trabajo en el Señor no es en vano.

1 Corintios 15:58

Resumen.

Desarrollar un software con calidad debería ser la meta de todas las empresas desarrolladoras de soluciones informáticas en el mundo, pero no todas aplican dicho principio. El Aseguramiento de la Calidad de un Software (SQA) está basado en un conjunto de actividades bien planificadas y sistemáticas necesarias para tributar a la confianza de que el producto satisfará los requisitos dados de calidad.

El presente trabajo se enfoca en el diseño y la aplicación de pruebas de software como un proceso de vital importancia para asegurar la calidad del Sistema LiberMaps. Se diseñaron y aplicaron Pruebas de Caja Negra, Pruebas de Carga, Pruebas de Estrés y Pruebas de Seguridad a partir de la elaboración de un Plan de Pruebas que establecía la estrategia a seguir para desarrollar este proceso. Una vez cumplido el objetivo general de la investigación se pudo comprobar la calidad del producto mediante la detección de no conformidades o errores que contenía el software.

La aplicación de métodos empíricos, tales como la Observación y el Experimento, permitieron obtener como resultado, quince no conformidades en las Pruebas de Caja Negra y hacer una plantilla resumen para la Prueba de Seguridad y para las Pruebas de Carga y Estrés. La detección de No conformidades estableció las bases para realizar próximas iteraciones de las pruebas realizadas.

Palabras Claves: Pruebas de software, Calidad de software, Pruebas Caja Negra, Pruebas de Carga, Pruebas de Estrés, Pruebas de Seguridad.

Abstract

Developing quality software should be the goal of all companies developing solutions in the world, but not all apply this principle. The Quality Assurance of Software (SQA) is based on a set of well-planned and systematic activities necessary to pay taxes to the trust that the product will satisfy given requirements for quality. This paper focuses on the design and implementation of software testing as a process vital to ensure the quality of LiberMaps System.

Tests were designed and applied for Black Box Testing Load Testing Stress Testing and Safety from the development of a test plan that established the strategy for developing this process. Having completed the general objective of the research was able to verify product quality by detecting non-conformities or errors that contained the software.

The application of empirical methods such as observation and experiment, allowed obtaining as a result, fifteen non-conformities in the Black Box testing and making a summary template for Security Testing and Load Testing and Stress. The detection of non-conformities laid the groundwork for future iterations of the tests.

Keywords: Software Testing, Software Quality, Black Box Testing, Load Testing, Stress Testing, Security Testing.

Índice.

Capítulo I: Fundamentos teóricos del proceso de pruebas de software. 16

1. Introducción. 16

1.1. Definiciones Básicas. 16

1.2. Pruebas de software. 19

1.2.1. Contexto de las pruebas de software. 19

1.2.2. Objetivo de las pruebas. 20

1.2.3. Proceso de Prueba. 20

1.2.4. Plan de Prueba. 22

1.3. Diseño de Pruebas. 22

1.4. Niveles de Prueba. 23

1.5. Otros tipos de prueba. 26

1.6. Metodología de desarrollo de software utilizada en el desarrollo del Sistema LiberMaps. 29

1.6.1. Proceso Unificado de Rational o Rational Unified Process (RUP). 29

1.6.2. Pruebas según RUP. 30

1.6.3. Relación de RUP con otras disciplinas. 31

1.6.4. Actividades fundamentales del flujo de trabajo de prueba. 32

1.6.5. Artefactos generados por el flujo de trabajo de prueba. 32

1.6.6. Roles del flujo de trabajo de prueba. 34

1.7. Métodos de prueba. 35

1.8. Herramientas de pruebas. 38

1.9. Situación actual en la Universidad de las Ciencias Informáticas. 39

1.10. Comparación entre pruebas de software. 40

1.11. Conclusiones parciales. 40

Capítulo II: Diseño y ejecución de pruebas al Sistema LiberMaps. 42

2. Introducción. 42

2.1. Catálogo de Mapas LiberMaps. 42
2.2. Características a probar. 45
2.3. Plan de Pruebas para el Sistema LiberMaps. 45
2.4. Procedimientos de prueba. 48
2.5. Casos de prueba de Carga y Estrés. 56
2.6. Conclusiones parciales. 64
Capítulo III: Evaluación de las pruebas del Sistema LiberMaps. 65
3. Introducción. 65
3.1. Resultados de la prueba de Caja Negra. 65
3.2. Resultados de la prueba de Carga y Estrés. 66
3.3. Resultados de la Prueba de Seguridad. 79
3.4. Conclusiones parciales. 79
Conclusiones Generales. 81
Recomendaciones. 82
Bibliografía Referenciada. 83

Índice de Tablas.

Tabla 1 Cronograma de Pruebas. 48
Tabla 2 Caso de Prueba "Configuración de Datos Globales". 49
Tabla 3 Caso de prueba "Exportar Mapfile". 50
Tabla 4 Caso de prueba "Gestionar Mapfile". 52
Tabla 5 Caso de prueba "Gestionar Usuarios del Sistema". 54
Tabla 6 Caso de prueba "Autenticar Usuario". 56

Índice de Ilustraciones.

Ilustración 1Proceso de pruebas. 21

Ilustración 2 Flujos de trabajo de RUP..... 31

Ilustración 3 Casos de uso arquitectónicamente significativos..... 126

Índice de Anexos.

Anexo 1 Comparación de las pruebas de software..... 88

Anexo 2 Casos de Pruebas de Caja Negra. 91

Anexo 3 No conformidades detectadas en las Pruebas de Caja Negra. 119

Anexo 4 Casos de uso arquitectónicamente significativos..... 126

Anexo 5 Resultados de las Pruebas de Carga y Estrés..... 126

Anexo 6 Método Empírico "Entrevista". 143

Introducción.

El desarrollo de las Tecnologías de la Informática y las Comunicaciones (TIC) ha cambiado los paradigmas tecnológicos existentes por muchos años. Cada uno de estos paradigmas ha sido sustituido por uno nuevo, lo que produce enormes cambios en la organización social y las relaciones económicas.

La actualidad del mundo está marcada por una crisis global, siendo los países desarrollados los que llevan el liderazgo en cuanto a desarrollo de las nuevas tecnologías. Estos países desarrollados cuentan con empresas de software, la mayoría de ellas, certificadas por CMMI (Integración de Modelos de Madurez de Capacidades), lo cual garantiza la calidad de sus procesos y le ofrece gran competitividad en el mercado de software.

En Cuba también existen empresas desarrolladoras de soluciones informáticas con calidad, entendiéndose esta como el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario (IEEE, 1990).

En la Universidad de las Ciencias Informáticas existen Centros de Desarrollo de Software, distribuidos en las siete facultades. La Facultad 6 cuenta con dos centros de desarrollo, uno de ellos es el Centro de Desarrollo Geoinformática y Señales Digitales (GEySED) y el otro es el Centro de Tecnologías de Gestión de Datos (DATEC). El Departamento Señales Digitales tiene como misión desarrollar productos, servicios y soluciones informáticas en el campo del procesamiento de Señales Digitales.

El Departamento Geoinformática agrupa proyectos dedicados a la conceptualización y desarrollo de soluciones informáticas para la geología y la minería, uno de estos proyectos es GeneSIG, el cual ha desarrollado varios productos de software entre ellos el Catálogo de Mapas LiberMaps. LiberMaps es un Sistema de Catálogo para la Gestión de Mapas en la Web que ha sido creado con el fin de gestionar las propiedades de los mapas y configurarlos en función de las necesidades del usuario. Constituye una herramienta muy útil para aquellos clientes que se apoyan en SIG.

Una de las mayores potencialidades que posee, es que permite controlar el acceso de los usuarios a los datos, lo que comúnmente es una necesidad de las entidades pues no todos los usuarios deben acceder a la misma información ni con los mismos derechos. Debido a la

importancia que tiene para la economía del país desarrollar soluciones de este tipo, se hace necesario el diseño y la aplicación de un conjunto de pruebas que aseguren la eliminación de la mayor cantidad de defectos antes del despliegue.

Actualmente la aplicación se encuentra en la etapa terminal de implementación en su primera versión. Durante este proceso se ha podido constatar que finalizadas las etapas de análisis, diseño e implementación de la aplicación, no existe un diseño de pruebas que valide la calidad del producto, el cumplimiento de los requisitos y especificaciones para ser entregado al cliente.

Anteriormente se le habían diseñado pruebas a la aplicación, pero se produjeron cambios en las funcionalidades del sistema. Este cambio en las funcionalidades acarreó como consecuencia que el diseño de pruebas realizado anteriormente ya no era válido, por lo que es necesario realizar diseños de casos de pruebas nuevos.

A partir de lo explicado anteriormente se concluye que el Sistema LiberMaps corre el riesgo de contener errores, tales como, funcionalidades ausentes, errores ortográficos, documentación incompleta, no mostrar mensajes de error cuando se entran datos no válidos, permitir el acceso a personas no autorizadas. Puede ser que el tiempo de respuesta de la aplicación al procesar información no es el que está establecido en los requerimientos funcionales, o la velocidad de procesamiento no coincide con la definida.

Esas consideraciones permiten reconocer como **problema científico**: ¿Cómo asegurar que el Sistema LiberMaps sea entregado con la calidad requerida a los clientes?

El **objeto de estudio es:** Pruebas de Software y el **campo de acción**: Proceso de pruebas para el Sistema LiberMaps.

La presente tesis se estructura y desarrolla en función del siguiente **objetivo general**: Diseñar y aplicar pruebas al Sistema LiberMaps. Se plantean como **objetivos específicos** de la investigación:

- ✓ Seleccionar tipos de prueba a aplicar.
- ✓ Realizar revisiones a los documentos pertenecientes al Sistema LiberMaps empleando listas de chequeo.
- ✓ Diseñar casos de prueba para el Sistema LiberMaps.
- ✓ Realizar pruebas al Sistema LiberMaps.
- ✓ Evaluar los resultados obtenidos.

La **idea a defender** plantea que: El diseño y la aplicación de las pruebas garantizarán que el Sistema LiberMaps sea entregado con calidad a los clientes.

Para darle solución a los objetivos específicos de la investigación se han planteado las siguientes **tareas investigativas**:

- ✓ Caracterizar la calidad de software, pruebas de software, herramientas y métodos para realizarle las pruebas a una solución informática.
- ✓ Caracterizar el Proceso de Pruebas.
- ✓ Describir el rol Diseñador de Casos de Prueba.
- ✓ Elaborar el Plan de Pruebas a realizar para el Sistema LiberMaps.
- ✓ Revisar los documentos del Sistema LiberMaps a través de Listas de Chequeo.
- ✓ Diseñar los Casos de Prueba.
- ✓ Realizar las Pruebas de Caja Negra al Sistema LiberMaps.
- ✓ Realizar las Pruebas de Carga y Estrés.
- ✓ Realizar las Pruebas de Seguridad.
- ✓ Validar las pruebas realizadas.
- ✓ Evaluar el producto.

La investigación exigió la utilización de **métodos de investigación** teóricos y empíricos:

Métodos Empíricos.

Entrevista: hizo posible contactar a especialistas del centro CALISOFT, para recibir información acerca de los tipos de pruebas que se le podían aplicar a una aplicación web y a personas del equipo de desarrollo del sistema que se propone evaluar, para tener acceso a la Herramienta y al expediente del proyecto. También se consultó al especialista del Laboratorio de Seguridad Informática de la facultad 2, para conocer las herramientas que se utilizan para realizar estas pruebas.

Observación: permitió comparar los resultados obtenidos en las pruebas con los resultados esperados, hacer un análisis y una evaluación de la respuesta del sistema, para determinar si la solución informática cumplía con los requerimientos funcionales.

Experimento: se utilizó para ejecutar los casos de prueba y apreciar con juegos de datos cual sería la respuesta del sistema. Propició el sometimiento de la aplicación a un entorno de prueba, en el que cada caso de prueba fue ejecutado con el fin de evaluar el software.

Métodos Teóricos.

Analítico-Sintético: Para analizar toda la bibliografía utilizada de una forma exhaustiva tomando de esta los elementos que sean de gran interés para la realización del Proceso de pruebas.

Hipotético-deductivo: se utilizó para llegar a nuevos conocimientos y pronósticos a partir de una idea a defender y posteriormente someterlas a verificaciones empíricas. Hizo posible verificar los diseños y ejecutar los casos de prueba, y así dar validez a la idea a defender planteada en la presente investigación.

Histórico-Lógico: hizo posible realizar un estudio analítico de la trayectoria histórica de las Pruebas de Software teniendo en cuanto a su evolución hasta la actualidad. Modelación para realizar el diseño de los casos de prueba a la aplicación.

El **Capítulo I: Fundamentos teóricos del proceso de pruebas de software** hace referencia a los fundamentos teóricos del proceso de pruebas de software. En el mismo se realiza un análisis de las diferentes pruebas que se le realizan a una solución informática. Se establece una comparación entre las diferentes pruebas analizadas para luego determinar cuáles se van a diseñar y aplicar al Sistema LiberMaps.

El **Capítulo II: Diseño y ejecución de pruebas al Sistema LiberMaps** refleja el Plan de Pruebas donde se establecen las características que se probarán de la aplicación, la estrategia a seguir para aplicar las pruebas y los diseños de casos de pruebas que se utilizarán para probar el sistema. Cada uno de los diseños refleja los resultados esperados y las precondiciones de la funcionalidad.

El **Capítulo III: Evaluación de las pruebas del Sistema LiberMaps** muestra el cumplimiento y evaluación de las pruebas realizadas al Sistema LiberMaps luego de diseñar y aplicar este proceso. Para evaluar el resultado obtenido en las pruebas se tendrán en cuenta las no conformidades encontradas.

Capítulo I: Fundamentos teóricos del proceso de pruebas de software.

1. Introducción.

En este capítulo se hace referencia a los fundamentos teóricos del proceso de pruebas de software. En el mismo se realiza un análisis de las diferentes pruebas que se le realizan a una aplicación y los niveles en los que se aplican estas pruebas. Se establece una comparación entre las diferentes pruebas analizadas y se determinan cuáles serán las pruebas a diseñar y aplicar para el Sistema LiberMaps.

1.1. Definiciones Básicas.

- ✓ **Calidad.**

Grado en el que un conjunto de características inherentes cumplen con los requisitos. (ISO, 9000)

La adecuación para el uso, satisfaciendo las necesidades del cliente. (Juran, 1990)

Conjunto de características de una entidad que le confiere la aptitud para satisfacer las necesidades establecidas y las implícitas. (ISO, 1994)

Se puede concluir que calidad es la medida con que un elemento, producto o sistema, satisface las necesidades del usuario o cliente.

- ✓ **Calidad de Software**

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, 1990)

Es el grado con el cual el software cumplirá con las expectativas del cliente. Se basa en combinación de atributos o características, expectativas de clientes y percepciones del usuario. (IEEE, 1983)

Se puede entender entonces que la calidad de un software está dada por el grado con el que ese producto satisface las necesidades y expectativas del cliente y que además cumple con los

requisitos del sistema establecidos por el usuario. Depende además de lo que el usuario considere importante aunque no se encuentre en los requerimientos de la aplicación.

- ✓ **Aseguramiento de la Calidad (SQA).**

El Aseguramiento de la Calidad de un Software está presente en métodos y herramientas de análisis, diseño, programación y prueba. También se encuentra en inspecciones técnicas formales, en todos los pasos del proceso de desarrollo de soluciones informáticas, en el control de la documentación del producto informático y de los cambios realizados, además en las estrategias de pruebas multiescala. (Lovelle, 1999)

Según Lovelle, el aseguramiento de la calidad está presente en la realización de pruebas durante el ciclo de vida de un software. En la mayoría de los países desarrollados esto es pasado por alto, y por lo general los productos terminan retrasados y con un presupuesto muy elevado, y lo primero que eliminan al retardarse es la etapa de prueba, realizándose las pruebas días antes de enviarse a aceptación.

- ✓ **Prueba de Software.**

La etapa de prueba de software consiste en que el ingeniero de software crea una serie de casos de prueba que pretenden demoler el software construido. (Pressman, 1998)

Es el proceso de evaluar un sistema o componente de un sistema de forma manual o automática para verificar que satisface los requisitos esperados, o para identificar diferencias entre los resultados esperados y los reales. (IEEE, 1983)

Las pruebas de software, se definen como el proceso de ejercitar o evaluar el sistema, por medios manuales o automáticos, para verificar que satisface los requerimientos o, para identificar diferencias entre los resultados esperados y los que producen el sistema. (Myers, 1979)

Realizado el análisis de los conceptos anteriormente citados se concluye que la prueba de software es el proceso en el que un sistema o producto de software es probado, con el objetivo de verificar si el mismo cumple con las especificaciones del cliente

- ✓ **Casos de Uso**

Describen un uso del sistema y cómo este interactúa con el usuario. (Gracia, 2003)

Un caso de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. (Servicios, 2006)

Se puede entender entonces que un caso de uso describe el uso de un sistema y cómo este interactúa con el usuario.

- ✓ **Requisitos**

Necesidad o expectativa establecida, generalmente implícita u obligatoria. (ISO, 2000)

Condición o capacidad que un usuario necesita para resolver un problema o alcanzar un objetivo. (IEEE, 1997)

Capacidad o condición que debe poseer el sistema o los componentes del sistema para satisfacer un contrato, estándar, especificación, u otro documento formalmente impuesto. (IEEE, 1997)

A partir del análisis de las definiciones anteriores se llega a la conclusión de que un requisito es una condición que debe poseer un sistema o los componentes del mismo, establecidos previamente por el cliente o usuario y que responde a las necesidades del consumidor.

- ✓ **Caso de Prueba.**

Casos de prueba o test case son un conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio. (Saravia, 2008)

Este artefacto es la especificación de un conjunto de entradas de prueba, condiciones de ejecución y resultados esperados, identificados con el propósito de hacer una evaluación de algún aspecto particular de un escenario. (Beizer, 1995)

Se coincide con los criterios planteados por los autores ya mencionados, donde se define que un caso de prueba es un conjunto de condiciones bajo las cuales el analista o probador determina si el sistema funciona correctamente.

- ✓ **Software**

El software es un conjunto de programas elaborados por el hombre, que controlan la actuación del computador, haciendo que este siga en sus acciones una serie de esquemas lógicos predeterminados. (Vergara, 2007)

1.2. Pruebas de software.

Las pruebas se centran principalmente en la evaluación o la valoración de la calidad del producto y representan un elemento crítico para la garantía del mismo. Es una actividad en la cual un sistema o uno de sus componentes se ejecutan en circunstancias previamente especificadas, los resultados se observan y registran y se realiza una evaluación de algún aspecto.

Es el proceso de ejecutar un programa con el fin de encontrar errores. El nombre prueba, además de la actividad de probar, se puede utilizar para designar un conjunto de casos y procedimientos de prueba. (Myers, 1979)

Realizar pruebas a un sistema informático no significa que el proceso de desarrollo esté asegurado, pero implementar un proceso de pruebas de software y sostenerlo en tiempo, es un buen inicio para aumentar el alcance. 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.

1.2.1. Contexto de las pruebas de software.

Inicialmente, la prueba fue confundida con el debugging (Depuración de programas), y se llevaba a cabo para garantizar que el programa trabajara con la calidad requerida. Los trabajos de G. Myers en los años setenta planteaba desde su punto de vista que la labor del tester o prueba era demostrar que el sistema no satisfacía los requerimientos.

Principios de G. Myers al respecto:

- ✓ La organización desarrolladora debiera evitar probar sus propios sistemas (hay una barrera mental que dificulta que se encuentre defectos en aquello que se ha construido con esmero y dedicación).
- ✓ No debe planearse un esfuerzo de prueba bajo el supuesto de que no se encontrarán defectos.
- ✓ Probar es una tarea creativa e intelectualmente demandante.

La prueba de software debiera verse como un proceso paralelo al desarrollo del software, y que se realiza por el convencimiento de que todo sistema debe ser revisado con el objetivo de establecer si el nivel de calidad requerido es alcanzado. Esto aceptando limitantes prácticas que

implican la inconveniente de revisarlo exhaustivamente, pero aplicando técnicas ingenieriles para subsanar estas limitantes.

Se concluye, basado en los principios de Myers, que las pruebas de software deben ser realizadas por personas que no pertenecen al equipo de desarrollo de la solución informática, esto garantizará que se encuentren la mayor cantidad de errores en la misma. Es de vital importancia que cuando se vaya a probar el software no se llegue con ideas preconcebidas de que la aplicación está libre de errores.

1.2.2. Objetivo de las pruebas.

Entre los principales objetivos de las pruebas de software, Myers plantea los siguientes:

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

El objetivo es diseñar casos de prueba que, sistemáticamente, identifiquen diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y de esfuerzo. La prueba no puede asegurar la ausencia de errores; solo puede demostrar que existen defectos en el software.

Se llega a la conclusión de que el objetivo que se persigue con las pruebas de software es demostrar la presencia de errores y no la ausencia de ellos. Para poder demostrar la presencia de fallas en una aplicación es necesario diseñar casos de prueba con altas probabilidades de detectar faltas.

1.2.3. Proceso de Prueba.

Las pruebas se realizan a lo largo del desarrollo del sistema y no simplemente al final. Esto significa descubrir problemas no conocidos y no demostrar la perfección de programas manuales o equipo.

Aunque probar es un proceso que requiere tiempo, es una serie esencial de pasos que ayuda a asegurar la calidad del sistema. La prueba se realiza en subsistemas o módulos de programa a medida que el trabajo avanza y se puede hacer en niveles diferentes y a diversos intervalos.

Fundamentos teóricos del proceso de pruebas de software

Capítulo 1

Antes de que el sistema sea puesto en aceptación, todos los programas deben ser probados, revisados con datos de prueba y examinados para ver si los módulos trabajan juntos entre ellos, tal como se planeó. Verificar que los módulos trabajen juntos validará que el sistema trabaja como un todo y que estos módulos se integran y funcionan correctamente.

El proceso de prueba tiene dos entradas: la configuración del software que incluye la especificación de requisitos del software, la especificación del diseño y el código fuente y una segunda entrada la configuración de prueba, que incluye un plan y un procedimiento de prueba.

Si el funcionamiento del software parece ser correcto y los errores encontrados son fáciles de corregir, se puede concluir que la calidad y la fiabilidad del software son aceptables, o que las pruebas son inadecuadas para descubrir errores serios.

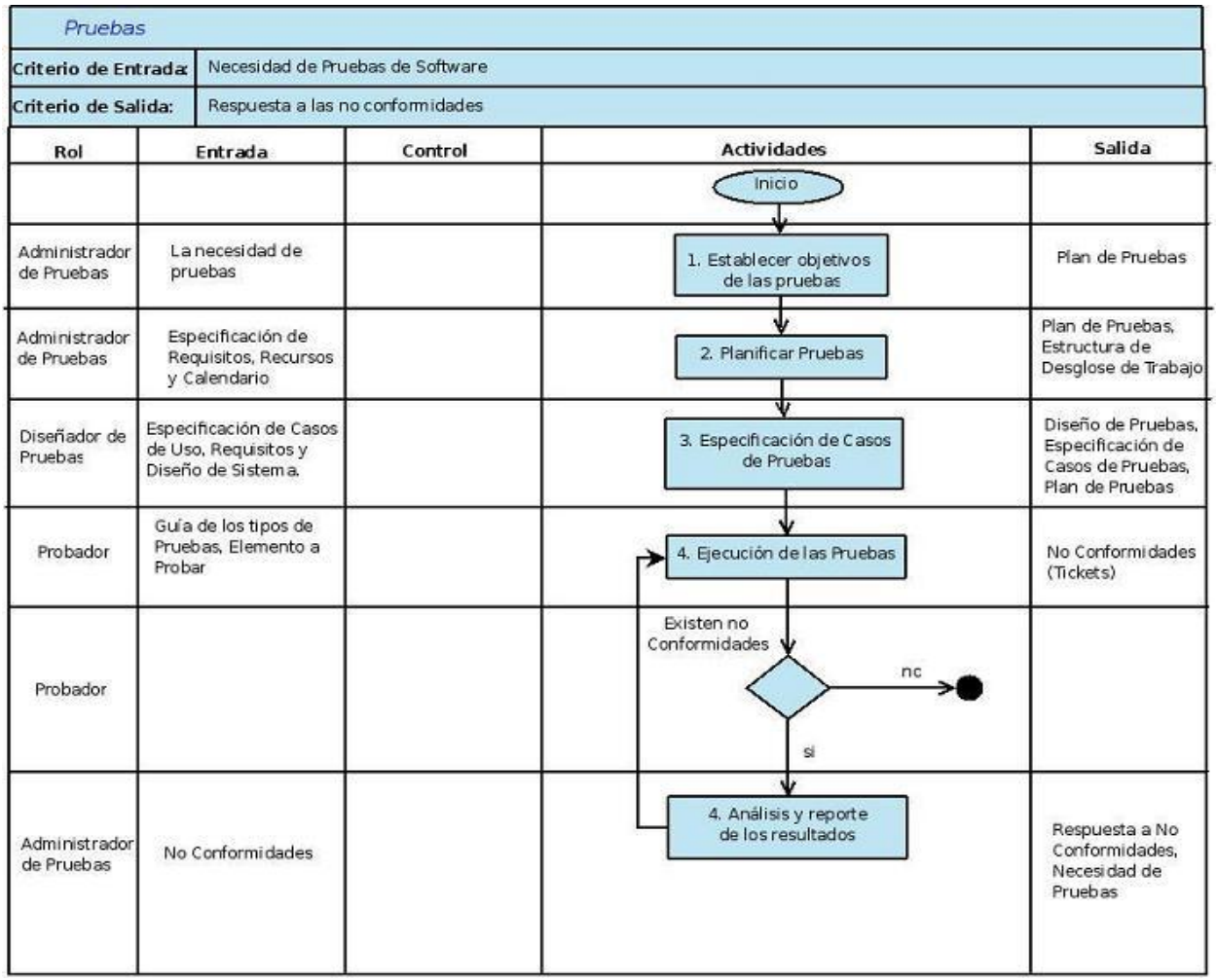


Ilustración 1Proceso de pruebas.

1.2.4. Plan de Prueba.

En el proceso de pruebas de software existen dos elementos fundamentales: Elaboración del Plan de Pruebas y diseño de pruebas específicas. El Plan de Pruebas describe la estrategia, recursos y planificación de las pruebas.

La estrategia de prueba incluye la definición de los tipos de pruebas a realizar para cada iteración y sus objetivos. Proporciona el marco dentro del cual el equipo de pruebas, desarrolla las pruebas trabajando con los recursos y la planificación dada.

El Plan de Pruebas proporciona la siguiente información:

- ✓ La definición de los objetivos de las pruebas en el ámbito de la iteración.
- ✓ La definición de los elementos que se van a probar.
- ✓ Una explicación del enfoque o estrategia que se usará.
- ✓ Los recursos necesarios y la planificación.
- ✓ Los resultados que se obtienen del proceso de prueba.

1.3. Diseño de Pruebas.

Al igual que la etapa de análisis, el diseño de las pruebas es una parte considerable de trabajo. Durante esta fase se diseñan e implementan los casos de prueba (*Test Cases*).Un caso de prueba es un conjunto de datos o situaciones de prueba que se utilizarán para ejecutar la unidad que se prueba o para revelar algo sobre el atributo de calidad que se está midiendo.

Tomando como punto de partida el resultado del análisis de pruebas, estas se diseñan en detalle, indicando cuáles son los pasos o el procedimiento a seguir y las técnicas, medidas o análisis que hay que aplicar. También se definen los conjuntos de datos que se usarán, y los resultados esperados que se producirán para conseguir demostrar que los objetivos de las pruebas se cumplen.

Pasos a seguir para realizar pruebas de software:

- ✓ Seleccionar qué es lo que debe medir la prueba, es decir, cuál es su objetivo, para qué exactamente se hace la prueba.
- ✓ Decidir cómo se va a realizar la prueba que ha de utilizarse para medir la calidad escogida y qué elementos de pruebas se deben usar.

- ✓ Diseñar los casos de prueba.
- ✓ Determinar cuáles deberían ser los resultados esperados o correctos de los casos de prueba y crear el documento con los diseños de casos de pruebas, antes de realizar la prueba.
- ✓ Ejecutar los casos de prueba.
- ✓ Comparar los resultados de la prueba con los resultados esperados. Cualquier discrepancia entre ellos significa un error.

1.4. Niveles de Prueba.

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 Independiente.

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

El objetivo de las pruebas independientes es proporcionar una perspectiva diferente y, por lo tanto, pruebas diferentes; además de dirigir estas pruebas a un entorno más rico de lo que es posible para el desarrollador. (Beizer, 1990)

Prueba de Unidad.

La prueba de unidad se centra en el módulo. Usando la descripción del diseño detallado como guía, se prueban los caminos de control importantes con el fin de descubrir errores dentro del ámbito del módulo. La prueba de unidad hace uso intensivo de las técnicas de prueba de caja blanca. (profesores., 2005-2006)

La prueba de unidad es un proceso para probar los subprogramas, las subrutinas, los procedimientos individuales o las clases en un programa. Es decir, se prueban primero los bloques desarrollados más pequeños del programa, antes de probar el software en su totalidad.

Las motivaciones para realizar este tipo de prueba son: manejar elementos de prueba combinados, puesto que centra la atención inicialmente en unidades más pequeñas del programa.

En segundo lugar, facilita la tarea de eliminar errores (el proceso de establecer claramente y de corregir un error descubierto), puesto que, cuando se encuentra un error, se sabe que existe en un módulo particular. Finalmente, introducen paralelismo en el proceso de pruebas del software presentándose la oportunidad de probar los múltiples módulos simultáneamente. (Myers, 2004)

Las pruebas de unidad son orientadas a caja blanca. Una razón es que como en pruebas de entidades más grandes tales como programas enteros (es el caso para los procesos de prueba subsecuentes), la prueba de caja blanca llega a ser menos factible. Una segunda razón es que los procesos de prueba subsecuentes están orientados a encontrar diversos tipos de errores. Por lo tanto, el procedimiento para el diseño de casos de prueba para una prueba de unidad es la siguiente: analizar la lógica del módulo usando uno o más de los métodos de caja blanca y después completar los casos de prueba aplicando métodos de caja negra a la especificación del módulo. (Myers, 2004)

Prueba de Integración.

La necesidad de realizar las pruebas de integración viene dada por el hecho de que los módulos que forman un programa suelen fallar cuando trabajan de forma conjunta, aunque previamente se haya demostrado que funcionan correctamente de manera individual. Con el uso de estas pruebas se consigue ir formando el programa global a medida que se comprueba como los distintos componentes interaccionan y se comunican libres de errores. (Vence, 2009)

Las pruebas de integración se realizan para garantizar que los componentes del modelo de implementación funcionan correctamente cuando se combinan para ejecutar un guión de uso. El destino de la prueba es un paquete o un conjunto de paquetes del modelo de implementación. A menudo, los paquetes que se combinan proceden de diferentes empresas de desarrollo. Las pruebas de integración exponen el estado incompleto o los errores de las especificaciones de la interfaz del paquete.

Prueba del Sistema.

Normalmente, la prueba del sistema se realiza cuando el software funciona en su totalidad. Un ciclo vital repetitivo permite que las pruebas del sistema se realicen mucho antes, en cuanto se

hayan implementado subconjuntos bien formados del comportamiento de guiones de uso. Normalmente, el destino son los elementos en funcionamiento de extremo a extremo del sistema.

El objetivo de las pruebas del sistema es comprobar la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica.

En la realización de estas pruebas es importante comprobar la cobertura de los requisitos y el total cumplimiento de los mismos, dado que su incumplimiento puede comprometer la aceptación del sistema por el equipo de operación responsable de realizar las pruebas de implantación del sistema, que se llevarán a cabo en el proceso de implantación y aceptación del sistema. (Noa, 2007)

Pruebas de Rendimiento.

Las pruebas de rendimiento son aquellas que son realizadas para determinar que tan rápido un sistema realiza una tarea bajo ciertas condiciones pre-planificadas de trabajo. Estas pruebas también son utilizadas para validar y verificar diferentes aspectos de la calidad de software, como por ejemplo, escalabilidad, fiabilidad y el buen uso de los recursos. Las pruebas de rendimiento constituyen un subconjunto de la Ingeniería de Pruebas, la cual se esfuerza en mejorar el rendimiento, basándose en el diseño y la arquitectura de un sistema, antes de la realización del proceso de codificación. (Corporación Sybven, 2011)

Las pruebas de rendimiento pueden servir para diferentes propósitos. Pueden demostrar que el sistema cumple los criterios de rendimiento. Pueden comparar dos sistemas para encontrar cuál de ellos funciona mejor o medir que partes del sistema o de cargas de trabajo provocan que el conjunto ofrezca bajo rendimiento. Para su diagnóstico, los ingenieros de software utilizan herramientas que permiten monitorear y medir qué partes de un dispositivo o software contribuyen a disminuir el rendimiento. (Corporación Sybven, 2011)

Es fundamental para alcanzar un buen nivel de rendimiento de un nuevo sistema, que los esfuerzos en estas pruebas comiencen en el inicio del proyecto de desarrollo y se amplíen durante su construcción. Cuanto más se tarde en detectar un defecto de rendimiento, mayor es el costo de la solución.

Pruebas de Aceptación.

El objetivo de las pruebas de aceptación es validar que un sistema cumpla con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento. Son definidas por el usuario del sistema y preparadas por el equipo de desarrollo, aunque la ejecución y aprobación final corresponden al usuario. (Barrios, 2007)

En la UCI las pruebas de aceptación se le realizan a todos los artefactos que constituyen entregables, esta actividad es desarrollada en un laboratorio en casa del cliente. Estas pruebas las realizan los propios clientes en un laboratorio de pruebas de aceptación y en un ambiente controlado. El cliente puede detectar no conformidades del negocio o del sistema, así como pedidos de cambios. Se establecen criterios de tolerancia para la aceptación de los mismos.

Prueba Alfa.

Se lleva a cabo, por un cliente, en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario. Las pruebas alfa se llevan a cabo en un entorno controlado. Para que tengan validez, se debe primero crear un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente. Una vez logrado esto, se procede a realizar las pruebas y a documentar los resultados. (Pressman, 2002)

Prueba Beta.

Se lleva a cabo por los usuarios finales del software en los lugares de trabajo de los clientes. A diferencia de la prueba alfa, el desarrollador no está presente, así la prueba beta es una aplicación en vivo del software en un entorno que no puede ser controlado por el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador. Como resultado de los problemas informados, el desarrollador del software lleva a cabo modificaciones y así prepara una versión del producto de software para toda la clase de clientes. (Pressman, 2002)

1.5. Otros tipos de prueba.

Pruebas de Liberación.

El proceso de pruebas de liberación no es más que un conjunto de experimentos 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 niveles para los que se emplean varios métodos y técnicas de una manera organizada y eficiente que conducirán a un mejor funcionamiento del producto. (Santanach, 2010)

En la UCI, las pruebas de liberación en los laboratorios del Departamento de Pruebas de Software (DPS) se realizan a todos los artefactos que constituyen entregables al cliente. Son realizadas por los probadores del DPS guiados por los especialistas en un laboratorio de prueba. Se libera el artefacto cuando ya no hay no conformidades en el mismo.

Pruebas Piloto.

Las pruebas piloto se utilizan principalmente para obtener las no conformidades y los cambios que necesita el sistema a medida que se está explotando en manos de usuarios reales. Ayudan a determinar los recursos que necesita la aplicación, cómo debe desarrollarse el despliegue, cómo debe ser la capacitación de los trabajadores, en fin definir los riesgos que puede traer el piloto y la mejor forma de mitigarlos. (Santanach, 2010)

Se emplean en la solución de los problemas que surgen con las tecnologías de apoyo que se comportan de forma diferente dependiendo de la configuración de la entidad y el modo en que los usuarios las utilizan, además para realizar ajustes en la obtención de datos en caso que sea necesario. En la UCI las pruebas piloto se realizan sobre aplicaciones que requieran un despliegue. Se realiza en un ambiente real del cliente. El usuario puede detectar no conformidades y realizar pedidos de cambios.

Pruebas Funcionales.

Se centran en las funciones, las entradas y las salidas. Su objetivo fundamental es asegurar el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados, estas pruebas tienen como meta:

- ✓ Verificar el procesamiento, recuperación e implementación adecuada de las reglas del negocio.
- ✓ Verificar la apropiada aceptación de datos.

Estas pruebas se realizan aplicando las pruebas de caja negra ejecutando cada caso de uso, flujo de caso de uso, o función, usando datos válidos e inválidos, para verificar lo siguiente:

- ✓ Que se aplique apropiadamente cada regla de negocio.

- ✓ Que los resultados esperados ocurran cuando se usen datos válidos.
- ✓ Que sean desplegados los mensajes apropiados de error y precaución cuando se usan datos inválidos.

De acuerdo con Pressman, las pruebas funcionales pueden realizarse en base a dos enfoques principales (Pressman, 1998):

- ✓ Prueba de Caja Blanca.
- ✓ Prueba de Caja Negra.

Pruebas de Seguridad.

Las pruebas de seguridad garantizan que los usuarios estén restringidos a funciones específicas o que su acceso esté limitado únicamente a los datos que están autorizados a acceder. Sólo aquellos usuarios autorizados a acceder al sistema son capaces de ejecutar las funciones del sistema. Su objetivo fundamental es comprobar los niveles de seguridad lógica del sistema.

Para aplicativos Web buscan garantizar:

- ✓ Que la información sea accesible únicamente por las personas autorizadas (Confidencialidad).
- ✓ La fiabilidad de la información (Integridad).
- ✓ Que la información esté disponible cuando se requiera y durante el tiempo que sea necesario (Disponibilidad).

Beneficios de las pruebas de seguridad:

- ✓ Aumenta la credibilidad e imagen corporativa.
- ✓ Control de la información sensible.
- ✓ Evita gastos asociados a la repetición de procesos.
- ✓ Impacto positivo a la operativa del negocio.
- ✓ Evalúan la existencia y control de los riesgos identificados.

Las Pruebas de Seguridad al producto LiberMaps serán basadas en la guía de OWASP (Proyecto Abierto de Seguridad para Aplicaciones Web). Se utilizará esta guía pues permite mejorar las habilidades para probar y además algunas de las pruebas que aquí se explican no requieren herramientas. Con esta guía se pueden detectar vulnerabilidades ahorrando tiempo y recursos. Se aplicará la prueba:

- ✓ **Comprobación del Sistema de Autenticación.**

Comprobará el sistema de autenticación, esto significa comprender como funciona el proceso de autenticación y usar esa información para eludir el mecanismo de autenticación. Para esto se utilizará la herramienta Brutus, es una herramienta craqueadora de contraseñas. Se incluyen dentro de estas pruebas otros tipos de pruebas tales como:

- ✓ **Pruebas de gestión de caché de navegación y de salida de sesión.**

Comprobar que la función de cierre de sesión esté correctamente implementada, y que no es posible reutilizar una sesión después del cierre. También comprobamos que la aplicación automáticamente cierra la sesión de un usuario cuando ha estado inactivo durante un cierto lapso de tiempo, y que ningún dato sensible permanece en el caché del navegador.

- ✓ **Pruebas de gestión de sesiones.**

La gestión de sesiones cubre ampliamente todos los controles que se realizan sobre el usuario, desde la autenticación hasta la salida de la aplicación.

1.6. Metodología de desarrollo de software utilizada en el desarrollo del Sistema LiberMaps.

Una metodología de desarrollo de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Utilizar una metodología para el desarrollo de un software es de vital importancia, pues ayuda a asegurar la calidad del producto y obtener un mejoramiento continuo de los procesos relacionados con la construcción del mismo. Sin embargo cada proyecto debe tener una metodología definida para su desarrollo. (Murcia, 2006)

1.6.1. Proceso Unificado de Rational o Rational Unified Process (RUP).

Es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Posee tres características esenciales:

- ✓ Centrado en la arquitectura.
- ✓ Dirigido por casos de uso.

- ✓ Iterativo e incremental.

Es la metodología utilizada en el desarrollo de LiberMaps, por estas razones las pruebas necesarias para validar la calidad del sistema se realizan siguiendo las actividades de los roles Diseñador de Pruebas y Probador. Esta metodología divide en cuatro fases el desarrollo de un software y agrupa las actividades en grupos lógicos definiéndose nueve flujos de trabajo principales, los primeros seis flujos son conocidos como flujos de ingeniería y los tres últimos de apoyo. Para esta investigación el flujo principal es el flujo de pruebas.

1.6.2. Pruebas según RUP.

Las pruebas según RUP se centran en tres objetivos fundamentales, estos son:

- ✓ Planificar las pruebas de integración; tanto de integración para cada construcción, como del sistema, dentro de cada iteración.
- ✓ Diseñar e implementar las pruebas con la elaboración de casos de prueba; así como evaluar la utilización de algún software que permita automatizar las pruebas.
- ✓ Realizar las pruebas según lo planificado anteriormente y arreglar los defectos detectados como resultado de la ejecución de las pruebas.

Fases que define RUP:

Inicio: Se desarrolla el prototipo exploratorio de demostración. No se requiere la elaboración de las pruebas.

Elaboración: Se prueban los componentes ejecutables que se han implementado y que deben corresponderse con la arquitectura básica de la aplicación.
Construcción: Se desarrollan los casos de prueba y procedimientos de prueba ya planteados.
Transición: En esta fase el producto se encuentra en su entorno de operación por lo que es probado por usuarios reales.

Flujos de trabajo.

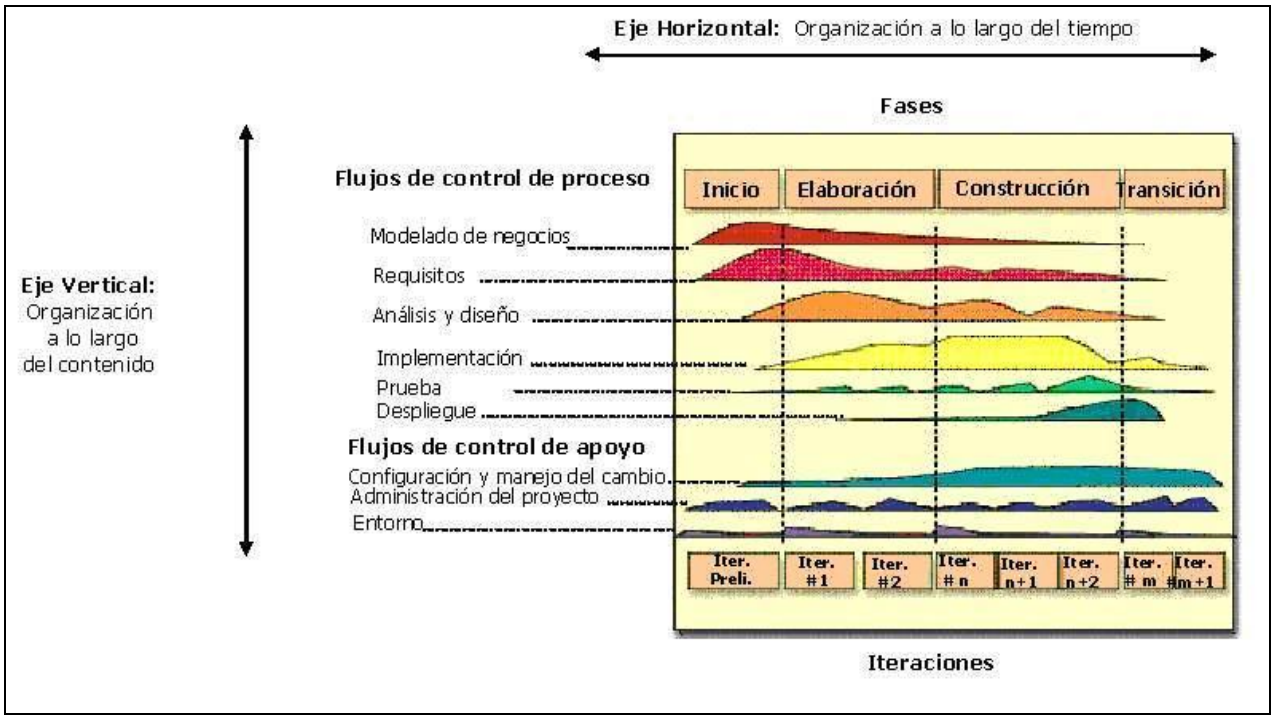


Ilustración 2 Flujos de trabajo de RUP.

1.6.3. Relación de RUP con otras disciplinas.

- ✓ La Disciplina de **Requerimientos**, captura los requerimientos para el producto software, los cuales son una de las entradas principales para identificar que pruebas ejecutar.
- ✓ La Disciplina de **Análisis y Diseño**, determina el diseño apropiado para el software, el cual es otra de las entradas principales para identificar que pruebas ejecutar.
- ✓ La Disciplina de **Implementación** produce versiones operacionales del sistema (builds) que son validados por pruebas, dentro de una iteración, múltiples builds serán probados (1 por ciclo de prueba).
- ✓ La Disciplina de **Despliegue** entrega el producto de software completo al usuario final. Antes el software es validado por Prueba, aunque pruebas de aceptación y pruebas betas son ejecutadas como parte del despliegue.
- ✓ La **Gestión de Proyecto** planifica el proyecto y las iteraciones, el Plan de Iteración descrito, artefacto que es una importante entrada usada cuando se va a definir la misión de evaluación para la prueba.
- ✓ La Disciplina de **Configuración y Control de cambio**: controla los cambios dentro del proyecto. La Prueba verifica que cada cambio ha sido completado apropiadamente.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

El ciclo de vida que se desarrolla por cada iteración, es llevado a cabo bajo dos disciplinas:

Disciplina de Desarrollo.

Ingeniería de Negocios: Entendiendo las necesidades del negocio.
Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.
Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.
Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo lo solicitado está presente.

Disciplina de Soporte.

Configuración y administración del cambio: Guardando todas las versiones del proyecto.
Administrando el proyecto: Administrando horarios y recursos.
Ambiente: Administrando el ambiente de desarrollo.
Distribución: Hacer todo lo necesario para la salida del proyecto.

1.6.4. Actividades fundamentales del flujo de trabajo de prueba.

Las actividades fundamentales que se desarrollan en este flujo de trabajo son:

- ✓ Planificar las pruebas.
- ✓ Diseñar las pruebas.
- ✓ Implementar las pruebas.
- ✓ Realizar pruebas de integración.
- ✓ Realizar pruebas de sistema.
- ✓ Evaluar pruebas.

1.6.5. Artefactos generados por el flujo de trabajo de prueba.

Un artefacto es todo producto o subproducto resultante del proceso de desarrollo del software, es un trozo de información que es producido, modificado o usado. No siempre en todo proyecto se crean los mismos artefactos, esto dependerá de las características del proyecto y los requisitos del cliente.

Para la fase de pruebas se tienen los siguientes artefactos:

Modelo de Pruebas: Describe principalmente cómo se prueban los componentes ejecutables en el modelo de implementación con pruebas de integración y de sistema. Puede describir también cómo se han probado aspectos específicos del sistema, por ejemplo, si la interfaz de usuario del sistema cumple con su objetivo y describe el cumplimiento de los requisitos funcionales y no funcionales del sistema. Es una colección de casos de pruebas, procedimientos de prueba y componentes de prueba.

Caso de Prueba: Un caso de prueba especifica una forma de probar el sistema, incluyendo la entrada o resultado con el cual será probado y las condiciones bajo las que ha de probarse. Un caso puede derivarse de las descripciones de los casos de uso del diseño.

Procedimiento de Prueba: Un procedimiento de prueba especifica cómo realizar uno o varios casos de pruebas o partes de estos. Un procedimiento de prueba puede ser una instrucción para un individuo sobre cómo realizar un caso de prueba manualmente, o una especificación de cómo interactuar manualmente con una herramienta de automatización de pruebas, para crear componentes ejecutables de prueba.

Componente de Prueba: Un componente de prueba automatiza uno o varios procedimientos de prueba o partes de ellos. Estos pueden ser desarrollados utilizando un lenguaje de guiones o un lenguaje de programación, o pueden ser gravados con una herramienta de automatización de pruebas.

Plan de Prueba: El plan de prueba describe las estrategias, recursos y planificación de la prueba, el nivel de cobertura de prueba y de código necesario, además del porcentaje de prueba que deberían ejecutarse con un resultado específico. El propósito del plan de pruebas es dejar de forma explícita el alcance, el enfoque, los recursos requeridos, el calendario y los responsables del proceso de pruebas.

Defecto: Un defecto es un síntoma de un fallo del software o un problema descubierto en una revisión, el cual puede ser utilizado para localizar cualquier cosa que los desarrolladores necesiten registrar cómo síntoma de un problema en el sistema.

Evaluación de Prueba: Es una evaluación de los resultados de los esfuerzos de prueba, tales como la cobertura del caso de prueba, de código y el estado de los defectos. Los diseñadores evalúan los resultados de la prueba comparando los resultados obtenidos con los objetivos trazados en el plan de prueba. Se realizan métricas que les permiten determinar el nivel de calidad del software y qué cantidad de pruebas es necesario realizar.

Estrategias de Pruebas: Una estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que llevan a la construcción correcta del software. Es el proceso de analizar cómo plantear las pruebas en el Ciclo de Vida. La metodología RUP a diferencia de otras metodologías, posee una gran cantidad de artefactos que brinda la posibilidad de que se tenga un mayor entendimiento del proceso de desarrollo a realizar y una mejor forma de organizar el trabajo.

RUP incluye el artefacto procedimiento de pruebas que es el objetivo principal de este trabajo. El flujo de pruebas que propone RUP recorre las 4 etapas de la metodología, lo que posibilita la aplicación de este procedimiento en cualquiera de ellas. Esto garantiza una vía factible para corregir los errores, partiendo de que los productos que son desarrollados con RUP, garantizan desde sus comienzos una arquitectura lo suficientemente fuerte, para no verse afectada por la aplicación de cambios que puedan surgir en el camino.

1.6.6. Roles del flujo de trabajo de prueba.

Aunque las pruebas pueden ser realizadas por usuarios finales y otros trabajadores del equipo, RUP define que los roles que intervienen en el flujo de prueba son los siguientes:

Diseñador de pruebas: Planifica, diseña y evalúa los resultados de la prueba. Es el responsable de definir el método de prueba y asegurar su implementación exitosa. El rol incluye identificar técnicas apropiadas, herramientas e instrucciones para implementar las pruebas necesarias y encauzar los recursos correspondientes para las pruebas.

Ingeniero de componentes: Responsable de la elaboración de los componentes de prueba para los procedimientos que pueden ser automatizados.

Ingeniero de pruebas de integración: Realizan las pruebas de integración.

Ingeniero de pruebas de sistema: Realiza las pruebas de sistema.

Administrador de prueba: Es el responsable del éxito de la prueba. Dirige el comportamiento de las pruebas y define los tipos de pruebas necesarias, además de manejar los resultados de las mismas.

Analista de prueba: Es el responsable de identificar y definir las pruebas requeridas, monitorear el progreso de la prueba y el resultado en cada ciclo de prueba, evaluando la calidad total experimentada como un resultado de las actividades de prueba. Este rol lleva la responsabilidad para representar apropiadamente las necesidades de los stakeholder que no tienen representación regular y directa en el proyecto.

Probador: Es el responsable de la ejecución de las pruebas necesarias y el registro del resultado de las mismas.

Para el proceso de pruebas que se llevará a cabo en el proyecto LiberMaps se toman en cuenta solo dos roles de los que propone RUP: diseñador de pruebas y probador. Se considera que con la definición de estos roles se le da cumplimiento al objetivo general de esta investigación, diseñar y aplicar pruebas al Sistema LiberMaps.

1.7. Métodos de prueba.

Los métodos de prueba de software tienen el objetivo de diseñar pruebas que descubran diferentes tipos de errores con menor tiempo y esfuerzo. (zonajava, 2005)

Método de la Caja Negra.

A la mayor parte de los usuarios de programas extensos no les interesa los detalles de su funcionamiento, lo único que desean es conseguir respuestas. Es decir, desean tratar el programa como una caja negra a la cual le introducen datos de entrada y obtienen de ella los datos de salida que esperan. De ahí el nombre de este método. (Beizer, 1990)

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 éste corra bien. A continuación se citan los criterios mínimos que deben guiarnos al escoger los datos de prueba de los programas (Beizer, 1995):

Valores fáciles: el programa se depurará con datos de fácil comprensión.

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 referencian a las dimensiones de un arreglo se equivoquen por uno.

Valores ilegales: Es preferible que el programa ofrezca a esta alguna indicación de probables errores detectados en los datos de entrada 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.

El Método de la Caja Negra se centra en los requisitos fundamentales del software y permite obtener entradas que prueben todos los requisitos funcionales del programa. Con este tipo de pruebas se intenta encontrar:

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

Algunos de los métodos empleados en las pruebas de Caja Negra son los siguientes:

Métodos de prueba basados en grafos: en este método se debe entender los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas. (Beizer, 1990)

Partición Equivalente: método de prueba de Caja Negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. (Pressman, 1998)

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. La partición equivalente es subjetiva. Dos probadores quienes prueban un programa complejo pueden llegar a diferentes conjuntos de particiones.

Para los casos de uso incluidos no se diseñaran casos de pruebas sino que estos se agregaran en el caso de uso que los contenga y se probaran cuantas veces aparezcan en el software. A medida que se diseñan los casos de pruebas se revisará el documento de especificación de caso de uso, se comprobará que lo que se encuentra en la aplicación corresponde a lo descrito en el documento así como la ortografía y la redacción.

Si es un software complejo el cual está compuesto por varios módulos, se probarán primero los módulos por separados y luego se diseñará un caso de prueba de integración para probar el software como un todo.

Análisis de valores límite: es una técnica de diseño de casos de prueba que completa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL (árbol binario de búsqueda) 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. (Pressman, 2002)

Prueba de la tabla ortogonal: hay aplicaciones donde el número de parámetros de entrada es pequeño y los valores de cada uno de los parámetros está claramente delimitado. Cuando estos números son muy pequeños (por ejemplo, tres parámetros de entrada tomando tres valores diferentes), es posible considerar cada permutación de entrada y comprobar exhaustivamente el proceso del dominio de entrada. (Pressman, 2002)

Adivinando el error: dado un programa particular, se supone, por la intuición y la experiencia, ciertos tipos probables de errores y entonces se escriben casos de prueba para exponer esos errores. Es difícil dar un procedimiento para esta técnica puesto que es en gran parte un proceso intuitivo. 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. Con la aplicación de esta técnica se obtiene un conjunto de pruebas que reduce el número de casos de pruebas y nos dicen algo sobre la presencia o ausencia de errores. (Myers, 2004)

Método de la Caja de Cristal (o Caja Blanca).

El segundo enfoque en la selección de los datos de prueba inicia con la observación de que un programa difícilmente puede considerarse como probado por completo si su código contiene partes que nunca han sido ejecutadas. En el método de la caja de blanca, se analiza la estructura lógica del programa y, para cada alternativa que pueda presentarse, los datos de prueba ideados conducirán a ella.

Así pues, se procura escoger los que verifiquen cada posibilidad en las estructuras de selección múltiple, las cláusulas de cada proposición if, cada alternativa de cada instrucción case (switch) y la condición de terminación de cada ciclo. En el caso de un programa extenso, este método es sin duda poco común, pero en un módulo pequeño constituye un excelente medio de pruebas y depuración. En un programa bien diseñado, cada módulo incluirá unos pocos ciclos y opciones. De ahí que basten algunos casos de prueba bien seleccionados para probar cada módulo por separado.

Antes de concluir que la prueba con el método de la caja de blanca siempre es el más adecuado, conviene señalar que en la práctica suele ser más eficaz para descubrir errores. Esto se debe, entre otras cosas, a que los errores más sutiles a menudo ocurren no dentro de un subprograma, sino en la interfaz de los subprogramas, al no entenderse bien las condiciones y normas precisas del intercambio de información entre los subprogramas. Por tanto, un buen criterio de prueba para proyectos extensos consiste en aplicar el método de la caja de blanca a cada módulo pequeño conforme se escriba; luego se usan esos datos empleados en esas pruebas en la verificación de las secciones más amplias del programa una vez terminadas.

1.8. Herramientas de pruebas.

JCrawler: es una herramienta para realizar pruebas no funcionales (de estrés principalmente) a sistemas de información o aplicaciones desarrolladas en ambiente web. JCrawler es una aplicación de software libre, desarrollada bajo la licencia GPL. Su principal característica es la de implementar la premisa de rastreo y exploración dentro de las aplicaciones en las que esta se aplica.

WebInject: es una herramienta libre y gratuita (GPL) para realizar test automáticos de aplicaciones y servicios web (SOAP/XML). Puede utilizarse para examinar componentes

individuales que tengan interfaz HTTP (JSP, ASP, CGI, PHP, Servlets, HTML Forms, XML/SOAP Web Services). Puede utilizarse como framework para realizar diferentes test y recolectar los resultados obtenidos.

Apache JMeter: aplicación java que nos permite definir comportamientos para casos de pruebas y medir su rendimiento. Válido para contenido estático y dinámico (ficheros, Servlets, scripts de Perl, objetos Java, bases de datos y consultas, FTP). Puede simular una gran carga en el servidor, HTTP y FTP testing y bases de datos mediante JDBC, multithreading y con grandes facilidades para extender su funcionalidad mediante plugins.

Brutus: Un cracker de autenticación de fuerza bruta para redes. Este cracker sólo para Windows se lanza sobre servicios de red de sistemas remotos tratando de averiguar contraseñas utilizando un diccionario y permutaciones de éste. Soporta HTTP, POP3, FTP, SMB, TELNET, IMAP, NTP, y más. El código fuente no está disponible. Los usuarios de UNIX deberían darle una mirada a THC-Hydra.

En este procedimiento se utilizarán las herramientas Apache JMeter, pues esta es la herramienta utilizada en la universidad para realizar Pruebas de Carga y Estrés y además porque es compatible con el sistema operativo libre Linux. Se utilizará además la herramienta Brutus, pues brinda la posibilidad de hacer pruebas de autenticación básica y es una de las herramientas que utiliza la universidad para realizarle Pruebas de Seguridad a los productos que se desarrollan en la UCI.

1.9. Situación actual en la Universidad de las Ciencias Informáticas.

El Centro de Desarrollo de Software GEYSED cuenta con un grupo de calidad conformado por estudiantes y profesores que vinculan el trabajo, el estudio y la investigación. Este grupo es encargado de velar por la calidad de todos los productos que se desarrollan en el centro. Cada cierto tiempo este realiza diferentes iteraciones de revisiones a los proyectos y aplican pruebas a los productos que serán liberados. Estos productos primero son probados por el personal encargado de realizar esta actividad en el centro GEySED, luego por CALISOFT.

Este último emplea un laboratorio llamado Departamento Industrial de Software (LISP), con el fin de realizar pruebas de liberación, de usabilidad, pruebas funcionales, de carga, estrés y de aceptación. Las pruebas mencionadas se realizan con el objetivo de obtener un producto con la

calidad requerida y que cumpla con las especificaciones del cliente, de no realizarse estas pruebas, traería como resultado que las aplicaciones fueran entregadas al cliente con un número elevado de errores, que se atrase el cronograma del producto y que aumente el costo del mismo por corregirlo nuevamente. Cada una de estas pruebas son aplicadas en dependencia del tipo de aplicación.

1.10. Comparación entre pruebas de software.

A partir de un análisis de las pruebas que se le pueden aplicar a un software según las características del software, se llega a la conclusión de que las pruebas que se le aplicarán al Sistema LiberMaps serán: Pruebas Funcionales, Pruebas de Carga, Pruebas de Estrés y Pruebas de Seguridad. (Ver Anexo 1)

Las Pruebas Funcionales se realizarán utilizando el método de Caja Negra partiendo de la técnica Partición Equivalente, diseñando casos de prueba por cada caso de uso; siempre que este no sea incluido, probando los módulos de la aplicación por separados, y por último se realizará un caso de prueba que pruebe a los dos módulos de la aplicación trabajando como un todo.

Las Pruebas de Carga y Estrés se realizarán utilizando la herramienta automatizada existente en la universidad (JMeter). Esta herramienta permitirá probar el sistema trabajando bajo condiciones de carga y bajo fallo. Las pruebas de seguridad se le realizarán para determinar si el sistema cumple con los niveles de accesos establecidos, utilizando la herramienta Brutus.

1.11. Conclusiones parciales.

Las características que se tuvieron en cuenta para elegir el tipo de prueba a aplicar fueron: el tipo de aplicación y la composición de la misma. Es una aplicación web con dos módulos y estos son relativamente grandes. No se escogió el método de Caja Blanca, basándonos en los planteamientos de Myers que define las pruebas a nivel de unidad son orientadas a Caja Blanca, por lo que no es factible utilizar este método, debido a que LiberMaps es una aplicación ya terminada, la cual necesita pruebas a nivel de sistema, por tal motivo se escogió el método funcional de Caja Negra.

Se aplicarán pruebas de Carga y Estrés porque estas pruebas se le realizan principalmente a aplicaciones web con el fin de verificar cómo trabaja la aplicación en extrema sobrecarga, cuánto

tiempo demora el software en cargar las páginas y cuántos usuarios concurrentes se pueden conectar a ella. Se escogieron además las pruebas de Seguridad porque uno de los módulos de LiberMaps se encarga de gestionar los perfiles, gestionar los roles con sus niveles de acceso a la información y restringir el acceso a la aplicación a solo los que estén autorizados.

Capítulo II: Diseño y ejecución de pruebas al Sistema LiberMaps.

2. Introducción.

En este capítulo se presenta el diseño y la aplicación de pruebas al Sistema LiberMaps. Se precisará además el Plan de Pruebas como el fundamento para todo el proceso que se ejecutará, así como la estrategia a seguir para realizar las pruebas y la configuración del entorno de prueba.

2.1. Catálogo de Mapas LiberMaps.

El proyecto LiberMaps surge a partir de la necesidad de contar con una aplicación capaz de publicar descripciones de datos geográficos mediante métodos estándares que posibilitan la realización de búsquedas a lo largo de múltiples servidores, además de controlar cada objeto mapa añadiendo seguridad a los mismos. LiberMaps es una aplicación que brinda servicios de catálogo de mapas en la web, constituyendo una herramienta útil para integrar con los SIG que utilizan como servidor de mapas a UMN MapServer.

Es un sistema configurable en correspondencia a un usuario y un rol, permitiendo controlar la seguridad sobre los datos y las funcionalidades, elemento importante para muchas entidades que necesitan establecer niveles de acceso sobre la información de acuerdo a su sensibilidad. La aplicación se encuentra actualmente dividida en dos módulos Gestión de Mapas y Gestión de Perfiles.

El Módulo Gestión de Mapas es el que se encarga de crear y modificar los mapas o mapfiles con todos sus objetos y brinda funcionalidades básicas como la importación, exportación y duplicado de un Mapfile, configuración global de la aplicación, gestión de grupos de capas, entre otros.

Por su parte, el Módulo Gestión de Perfiles se distingue por la creación de los perfiles de usuarios tanto sobre los datos como a las funcionalidades del catálogo, permitiendo otorgar a cada usuario permisos sobre un determinado mapa y a tantas capas como desee. Posibilita además de acuerdo a los roles definidos, asignar una configuración previa de las funcionalidades, de modo que cuando un usuario se autentique solo pueda acceder a los datos definidos en su perfil y a las funcionalidades específicas de su rol.

Entre los dos módulos agrupan 14 casos de uso arquitectónicamente significativos, estas funcionalidades son: CU Gestionar Mapfile, CU Exportar Mapfile, CU Configurar Datos Globales,

CU Modificar Objeto, CU Modificar Objeto Web, CU Gestionar Objetos OUTPUTFORMAT, CU Gestionar Objetos SYMBOL, CU Gestionar Objetos LAYER, CU Crear Perfiles, CU Configurar Perfil de Usuarios, CU Crear Perfiles de Funcionalidades, CU Gestionar Usuarios, CU Autenticar Usuario, CU Gestionar Roles.

CU Gestionar Mapfile: Permite al usuario crear un mapfile a partir de los atributos básicos del mismo, almacenándolos de manera persistente en la base de datos. Además posibilita al usuario la opción de eliminar un mapfile con todos los objetos que lo conforman, eliminándose también de la base de datos.

CU Exportar Mapfile: Permite exportar a un fichero con extensión .map la información del mapfile almacenada en la base de datos.

CU Configurar Datos Globales: Permite configurar datos generales de funcionamiento de la aplicación como la dirección donde se encuentra el servidor de mapas, la dirección donde se debe exportar el mapfile, etc.

CU Modificar Objeto: Permite modificar todos los datos de los objetos reference, scalebar, querymap, legend del mapfile.

CU Modificar Objeto WEB: Permite modificar todos los datos del objeto web del mapfile.

CU Gestionar Objetos OUTPUTFORMAT: Permite realizar la edición, adición y eliminación de los outputformat del mapfile.

CU Gestionar Objetos SYMBOL: Permite realizar la edición, adición y eliminación de los objetos symbols del mapfile.

CU Gestionar Objetos LAYER: Permite adicionar y eliminar las capas del mapfile, así como su agrupación de acuerdo a los grupos ya definidos.

CU Crear Perfiles: Permite crear perfiles de usuario a los administradores, para dar acceso a los mapas y a las capas de dichos mapas a los usuarios de la aplicación. Para ello se parte de usuarios existentes, también posibilita eliminar dichos perfiles.

CU Configurar Perfil de Usuarios: Permite que el administrador configure el perfil de cada usuario eligiendo de los mapas a los que tienen acceso, con cuales deben trabajar, al igual que

con las capas. En función de esta configuración la aplicación se personaliza al usuario autenticado.

CU Crear Perfiles de Funcionalidades: Permite crear perfiles de usuario a los administradores, para dar acceso a los usuarios a las funcionalidades que posee la aplicación. Para ellos se parte de usuarios existentes. También se posibilita eliminar dichos perfiles.

CU Gestionar Usuarios: Permite la adición de nuevos usuarios al sistema, así como la modificación y eliminación de los ya existentes.

CU Autenticar Usuario: Permite identificar al usuario que se registre en la aplicación, y en función de su rol y nivel de acceso la aplicación se configura.

CU Gestionar Roles: Permite la adición de nuevos roles de usuario al sistema, así como la modificación y eliminación de los ya existentes.

A los módulos mencionados anteriormente se le aplican pruebas de Caja Negra con el objetivo de verificar si las funcionalidades funcionan correctamente, pruebas de Carga y Estrés para validar y valorar la aceptabilidad de los límites operacionales de un sistema bajo carga de trabajo variable y de estrés para evaluar cómo el sistema responde bajo condiciones anormales. Se le aplican también pruebas de Seguridad para certificar que los datos o sistemas que son objetos de prueba, son accedidos sólo por los actores que tienen permiso para hacerlo.

Los atributos que se miden varían, y su relevancia y el nivel de detalle dependen del propósito de cada aplicación particular y los objetivos de la prueba. Hay algunos atributos que son comunes a la hora de desarrollar las pruebas de software, por ejemplo si la aplicación cumple sus requerimientos y especificaciones, si se ejecuta sin errores, si es fácil de usar, si es aceptable para el usuario, si la documentación está completa y correcta. También aquellos que determinan si la aplicación es confiable, precisa, completa, rápida, utilizable, flexible y bien documentada. (Lamancha, 2007)

En LiberMaps se miden principalmente los atributos de funcionamiento, para validar con la comprobación de estos la calidad del sistema. Se miden además atributos que determinan que la aplicación es confiable, precisa y completa.

2.2. Características a probar.

En la aplicación que se prueba en el presente trabajo (Catálogo de mapas LiberMaps) los principales atributos a medir en las Pruebas de Caja Negra son los de funcionalidad, asegurando que la funcionalidad es la requerida, incluyendo la entrada de datos, su procesamiento y recuperación.

La aplicación de Pruebas de Carga medirá atributos tales como velocidad de respuesta ante una petición del usuario dependiendo de la cantidad de trabajo del sistema.

Aplicando Pruebas de Estrés se evaluarán los atributos: extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible y recursos compartidos no disponibles.

En las Pruebas de Seguridad se tendrán en cuenta las siguientes características: seguridad a nivel de aplicación (verificar que el usuario no pueda acceder ni modificar los datos a los que no tiene permiso) y seguridad a nivel de sistema (verificar que los usuarios que no están registrados en la aplicación no pueden acceder a la herramienta).

2.3. Plan de Pruebas para el Sistema LiberMaps.

Propósito.

El plan de pruebas para el Sistema LiberMaps trata de cumplir los siguientes objetivos:

- ✓ Identificar la información existente en el proyecto y los componentes que deben ser testados.
- ✓ Listar los principales requisitos a probar.
- ✓ Definir las estrategias de pruebas que deben emplearse.
- ✓ Identificar los recursos necesarios que puedan requerirse.
- ✓ Listar los artefactos entregables del proceso de prueba.

Roles y responsabilidades.

El equipo encargado del diseño y la ejecución de las pruebas estará compuesto por:

- ✓ 1 Diseñador de pruebas.
- ✓ 1 Probador.

Estrategia de prueba.

Prueba Funcional: Este tipo de prueba se realiza en base a dos enfoques principales: prueba de Caja Blanca y pruebas de Caja Negra. Para probar la aplicación en cuestión se utilizará el enfoque prueba de Caja Negra con el objetivo de verificar que se aplique correctamente cada regla del negocio, que los resultados esperados ocurran cuando se usen datos válidos y por último que sean desplegados los mensajes apropiados de error y precaución cuando se usan datos inválidos.

El método de prueba que se utilizará será Partición equivalente, el mismo consiste en diseñar un caso de prueba por cada caso de uso, para los casos de uso incluidos no se diseñarán casos de pruebas, sino que estos se agregarán en el caso de uso que los contenga y se probarán cuántas veces aparezcan en el software. Se realizará esta prueba basada en los casos de uso, se probarán cada uno de los casos de prueba y se terminará cuando se hayan probado todos.

Prueba de Carga: Para esta prueba se utilizará la herramienta JMeter que permitirá diagnosticar si la respuesta que ofrece el sistema es la adecuada. Con esta herramienta podremos realizar un análisis de la capacidad de carga y el rendimiento general de la solución informática.

- ✓ Se seleccionarán de los casos de uso arquitectónicamente significativos los escenarios que requieran mayor cantidad de tiempo de procesamiento para crear los planes de pruebas.
- ✓ Crear planes de pruebas con la herramienta JMeter.
- ✓ Ejecutar los planes de pruebas.
- ✓ Analizar los resultados del Informe Agregado generado por la herramienta JMeter.

Prueba de Estrés: Para realizar esta prueba se utilizará la herramienta JMeter, la misma nos permitirá parametrizar el número de usuarios simultáneos para probar la secuencia de peticiones http que guardó.

- ✓ Se utilizarán los mismos casos de prueba que se diseñaron para las Pruebas de Carga y se aumentará en número de usuarios concurrentes para ejecutarlos con la herramienta.
- ✓ Se realizará una plantilla resumen con los resultados obtenidos.

Prueba de Seguridad: Se verificará que la aplicación no puede ser accedida por los usuarios que no han sido establecidos y que no se puede tener acceso a la información si el rol no tiene permisos previamente establecidos. Para esta prueba se utilizará la herramienta Brutus.

- ✓ Se creará un fichero con posibles usuarios y otro con posibles contraseñas.

- ✓ Se introducirán estos ficheros en la herramienta y ella se encargará de crear todas las combinaciones posibles de usuarios y contraseñas e intentará entrar al Sistema LiberMaps.
- ✓ Los resultados se observarán y se realizará una plantilla resumen con estos resultados.

Tipos de pruebas y técnicas empleadas.

- ✓ **Prueba Funcional.**

Objetivo: Su objetivo fundamental es asegurar el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados.

Técnica: A partir de la técnica partición equivalente que consiste en diseñar un caso de prueba por cada caso de uso, se probarán cada una de las funcionalidades basándose en los diseños de casos de prueba, probando con datos válidos e inválidos, verificando que los datos de salida son los esperados.

Herramienta: No se utilizarán herramientas para hacer esta prueba.

Criterio de terminación: Se han probado todos los casos de prueba.

- ✓ **Prueba de Carga.**

Objetivo: Determinar y asegurarse de que el sistema funciona correctamente más allá de la carga de trabajo máxima prevista. Evaluar las características de funcionamiento, tales como tiempos de respuesta e índices de transacción.

Técnica: Concurrencia usuario-rendimiento.

Herramienta: JMeter.

- ✓ **Prueba de Estrés**

Objetivo: Determinar como el sistema responde bajo condiciones anormales, tales como, extrema sobrecarga, insuficiente memoria.

Técnica: Concurrencia usuario-rendimiento.

Herramienta: JMeter.

- ✓ **Prueba de Seguridad.**

Objetivo: verificar que la información contenida en la aplicación se encuentra protegida por permisos previamente establecidos.

Técnica: se verificará que la aplicación puede ser accedida por los usuarios establecidos y que se puede tener acceso a la información según los roles previamente establecido.

Herramienta: se utilizará la herramienta Brutus, la misma es una craqueadora de contraseñas que viola la seguridad del sistema e intenta penetrar a la aplicación.

Cronograma de Pruebas.

Tabla 1 Cronograma de Pruebas.

Actividad	Fecha inicial	Fecha final
Definición del plan y estrategia de pruebas.	6/Enero/2011	17/Enero/2011
Diseño de casos de prueba	17/Enero/2011	20/Marzo/2011
Ejecución de las pruebas funcionales	21/Marzo/2011	01/Abril/2011
Ejecución de las pruebas de carga y estrés.	02/Abril/2011	16/Abril/2011
Ejecución de las pruebas de seguridad	17/Abril/2011	24/Abril/2011
Validación de las pruebas.	25/Abril/2011	05/Mayo/2011

2.4. Procedimientos de prueba.

Para el Sistema LiberMaps se ha definido un procedimiento de prueba en dependencia de los casos de uso que conforman la aplicación, estos casos de uso serán la base para el diseño de los casos de prueba y guiarán la confección de los mismos, pues especifican como se debe ejecutar un caso de prueba de manera general.

Se diseñaron veintiuno casos de pruebas de Caja Negra, de ellos 14 son arquitectónicamente significativos y esos son los que se recogen en la presente tesis. (Ver Anexo 2) En cada uno de estos procedimientos se realiza una descripción de la funcionalidad, las condiciones de

ejecución y los escenarios a probar. También se encuentra una descripción de las variables del caso de prueba, el flujo central y la matriz de datos.

CU Configurar Datos Globales.

Descripción de la funcionalidad.

El caso de uso inicia cuando el usuario desea realizar la configuración global, que es donde se exportan los mapas y termina cuando el usuario luego de haber introducido los datos para la configuración guarda los datos.

Condición de ejecución.

El usuario debe estar autenticado y poseer los privilegios necesarios para acceder a esta opción.

Flujo central.

Catálogo de Mapas.
Módulo Configuración.
Clic en la opción “Configuración global”.
Llenar todos los campos del formulario.
Clic en el botón “Guardar”.

Tabla 2 Caso de Prueba "Configuración de Datos Globales".

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: “Configuración global”	EC 1.1: “Configuración global con éxito”	El usuario selecciona la opción “Configuración global”, el sistema muestra una ventana “configuración global” para que el usuario introduzca los datos de la URL base del servidor, WSDL de seguridad, CGI de MapServer, Mapfile para vista previa y Ruta de exportación, el usuario introduce los datos donde se

		exportarán los mapas y da clic en el botón guardar, el sistema guarda los datos introducidos por el sistema.
	EC 1.2: "Configuración global incorrecta"	El usuario no introduce todos los datos para realizar la configuración y oprime el botón "Guardar", el sistema muestra una ventana informando que el formulario no es válido y marca en rojo los campos que ha dejado en blanco. El usuario da clic en el botón "OK" de la ventana de aviso. El sistema retorna al usuario a la ventana "Configuración global".

CU Exportar Mapfile.

Descripción de la funcionalidad.

El caso de uso inicia cuando el usuario selecciona la opción "Exportar un Mapfile" y termina cuando el usuario selecciona la opción de "Exportar".

Condiciones de ejecución.

El usuario debe estar autenticado y poseer los privilegios necesarios para acceder a esta opción.

Flujo central.

Catálogo de Mapas.
Módulo Configuración.
Clic en la opción del menú "Exportar".
Seleccionar un mapfile.
Clic en el botón "Exportar".

Tabla 3 Caso de prueba "Exportar Mapfile".

Nombre de la sección.	Escenarios de la sección.	Descripción de la funcionalidad.
SC 1: "Exportar Mapfile".	EC 1: "Exportar mapfile correctamente"	El usuario selecciona la opción "Exportar un mapfile", el sistema muestra una ventana "Exportar" con un ComboBox con todos los mapas, el usuario selecciona un mapfile de los que están contenidos en el ComboBox y da clic en el botón "Exportar".
	EC 2: "Exportar mapfile incorrectamente"	El usuario no selecciona un mapfile de los contenidos en la lista y oprime el botón "Exportar", el sistema muestra un cartel informando que se debe elegir un mapfile para exportar y marca en rojo el campo "Mapfiles". El usuario da clic en el botón "OK" de la ventana de aviso y el sistema retorna a la ventana "Exportar un mapfile".

CU Gestionar Mapfile.

Descripción general.

El caso de uso inicia cuando el usuario desea adicionar o eliminar un Mapfile, este CU le permite al usuario crear un Mapfile con todos sus componentes y almacenar su información de forma persistente en la base de datos o eliminarlo de la misma, y termina cuando se ejecuta cualquiera de las dos operaciones.

Condiciones de ejecución

El usuario debe estar autenticado y poseer los privilegios necesarios para acceder a esta opción.

Flujo central.

Catálogo de mapas.
Módulo Configuración.

Clic en la opción “Crear”.

Funcionalidad Crear nuevo Mapfile.

Llenar todos los datos requeridos.

Clic en el botón Guardar.

Tabla 4 Caso de prueba "Gestionar Mapfile".

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: “Adicionar Mapfile”	EC 1.1: “Adicionar Mapfile correctamente”	El usuario selecciona la opción Crear, el sistema le muestra una ventana “Crear nuevo Mapfile” en la que deberá llenar todos los campos de la interfaz. El usuario entra los datos al sistema y da clic en el botón guardar, el sistema guarda los datos introducidos.
	EC 1.2: “Importar Mapfile correctamente”	El usuario selecciona la opción “Importar Mapfile”, el sistema muestra una ventana “Importar un Mapfile” mostrando un TextBox para entrar la localización del Mapfile, el usuario introduce la localización, el sistema carga el Mapfile desde la dirección y lo almacena en la base de datos.
	EC 1.3: “Crear Mapfile incorrectamente”	El usuario no introduce todos los datos requeridos para crear un Mapfile, el sistema muestra un cartel informando que el formulario no es válido porque todos sus campos no fueron llenados marcándolos en rojo, si el campo que no se llenó fue la

		proyección el sistema muestra un cartel donde especifica que se debe seleccionar un tipo de proyección.
SC 2: "Eliminar Mapfile"	EC 2.1: "Eliminar Mapfile correctamente"	El usuario selecciona la opción "Eliminar", el sistema muestra una ventana "Eliminar" con un Grid que muestra todos los mapas del sistema, el usuario selecciona el Mapfile que desea eliminar, el sistema elimina el Mapfile y actualiza los cambios en la base de datos.
	EC 2.2: "Eliminar Mapfile incorrectamente"	El usuario no selecciona el Mapfile o los Mapfiles que desea eliminar, el sistema muestra un cartel informando que debe seleccionar al menos un mapa.

CU Gestionar Usuarios del Sistema.

Descripción general.

El caso de uso se inicia cuando el usuario selecciona la opción Gestionar Usuarios, y termina cuando el usuario finaliza las operaciones guardando los cambios.

Condiciones de ejecución.

El usuario debe estar autenticado y debe ser administrador.

Flujo central.

Catálogo de Mapas.
Módulo Perfil.

Clic en la opción “Gestionar Usuarios”.
Clic en la opción “Nuevo Usuario”.
Llenar los campos del formulario.
Clic en el botón “Guardar”.

Tabla 5 Caso de prueba "Gestionar Usuarios del Sistema".

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: “Adicionar nuevo usuario”	EC 1.1: “Adicionar nuevo usuario con éxito”	El administrador elige la opción adicionar usuario. El sistema muestra un Findlabel con todos los roles existentes, un ComboBox Nombre de usuario y otro contraseña. El administrador introduce los datos y guarda los cambios. El sistema adiciona el nuevo usuario a la base de datos.
	EC 1.2: “Adicionar nuevo usuario incorrectamente ”	El administrador no introduce los datos correspondientes y da clic en el botón “Guardar”. El sistema muestra un mensaje informando que el formulario no es válido, mostrando en rojo el campo que falta.
	EC 1.3: “Adicionar nuevo usuario existente”	El administrador introduce el nombre de un usuario existente y da clic en el botón “guardar”. El sistema muestra un mensaje donde detalla que ese usuario ya existe.
SC 2: “Editar roles de usuario”	EC 2.1: “Editar roles de usuario con éxito ”	El administrador elige la opción editar usuario. El sistema muestra un Findlabel “Editar roles del usuario” con un ComboBox usuario y otros ComboBox role, con todos los usuarios y roles

		existentes. El administrador escoge el usuario y el rol y da clic en el botón “Guardar”. El sistema actualiza los datos en la base de datos.
	EC 2.2: “Editar roles de usuario incorrectamente”	El administrador introduce los datos dejando campos en blanco y da clic en el botón guardar. El sistema marca en rojo los campos requeridos y muestra un mensaje de aviso donde especifica que el formulario no es válido.
SC 3: “Eliminar usuario”	EC 3.1: “Eliminar usuario con éxito”	El administrador elige la opción eliminar un usuario. El sistema muestra todos los usuarios registrados en el sistema. El administrador elige el usuario que desea eliminar y da clic en el botón eliminar. El sistema elimina el usuario de la base de datos.
	EC 3.2: “Eliminar usuario incorrectamente”	El administrador no elige el usuario que desea eliminar y selecciona la opción eliminar. El sistema muestra un mensaje diciendo que se debe seleccionar al menos un usuario.

CU Autenticar Usuario.

Descripción General.

El caso de uso inicia cuando el usuario desea entrar a la aplicación y termina cuando el usuario finaliza las operaciones deseadas en la aplicación.

Condiciones de ejecución.

No tiene condiciones de ejecución.

Flujo central.

Catálogo de Mapas.

Página principal de LiberMaps.

Tabla 6 Caso de prueba "Autenticar Usuario".

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: "Autenticar usuario"	EC 1.1: "Autenticar usuario con éxito"	El usuario solicita la entrada a la aplicación. El sistema muestra una ventana para que el usuario pueda autenticarse poniendo usuario y contraseña. El usuario introduce los datos requeridos para autenticarse y obtener los privilegios conferidos. El sistema verifica la existencia del usuario y carga la aplicación según los privilegios que le han sido otorgados y es capaz de configurarse en función de sus perfiles de datos y funcionalidades.
	EC 1.2: "Autenticar usuario dejando campos en blanco"	El usuario no introduce los datos necesarios para autenticarse. El sistema marca en rojo los campos requeridos.
	EC 1.3: "Autenticar usuario con datos incorrectos"	El usuario introduce datos incorrectos. El sistema muestra una ventana de error indicando que el usuario no existe o que la contraseña no es válida.

2.5. Casos de prueba de Carga y Estrés.

Para las Pruebas de Carga se diseñaron quince casos de pruebas, donde se especifican los escenarios que se prueban por cada caso de uso y las variables necesarias para probar el caso de prueba. (Ver Anexo 3)

Caso de prueba "Autenticar usuario".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)
Autenticar usuario

Variables	**Valores**
Número de usuarios concurrentes: (número de hilos)	50
Tiempo entre conexión y conexión: (período de subida(en segundos))	1,3
Número de iteraciones: [Contador del bucle]	1

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	100

Tiempo entre conexión y conexión: (período de subida(en segundos))	1,3
Número de iteraciones: [Contador del bucle]	1

Caso de prueba "Configuración de Datos Globales".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)
Configuración de Datos Global

Variables	**Valores**
Número de usuarios concurrentes: (número de hilos)	50
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	100
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Caso de prueba "Configurar Perfil de Usuarios".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)
Configurar capas activas del Mapfile
Editar datos

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	80

Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Caso de prueba "Crear Perfil de Funcionalidades".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)
Modificar perfil de funcionalidades
Eliminar acceso a funcionalidades

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	70
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Caso de prueba "Crear Perfiles".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)
Crear Perfil
Eliminar acceso a datos

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	70
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Caso de prueba "Exportar Mapfile".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)
Exportar Mapfile

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	70
Tiempo entre conexión y conexión: (período de subida(en segundos))	2
Número de iteraciones: [Contador del bucle]	1

Caso de prueba "Gestionar Mapfile".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)
Adicionar Mapfile
Eliminar Mapfile

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	50
Tiempo entre conexión y conexión:	2

(período de subida(en segundos))	
Número de iteraciones: [Contador del bucle]	1

Caso de prueba "Gestionar METADATA del Objeto Web".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)
Editar METADATA
Crear nuevo metadato
Eliminar metadato

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	90
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Para ejecutar las Pruebas de Estrés se utilizarán los mismos diseños de casos de prueba de carga, solo se aumentarán la cantidad de conexiones. En la ejecución de las Pruebas de Carga se utilizó un rango de 1 a 100 conexiones, por lo que en las Pruebas de Estrés se utilizará una cantidad de conexiones mayor.

2.6. Conclusiones parciales.

En este capítulo quedó reflejado el Plan de Pruebas a aplicar y las estrategias para realizar las pruebas. Se realiza una breve descripción de la aplicación con sus principales funcionalidades. En el Plan de Prueba se establece el cronograma de pruebas a aplicar, los recursos necesarios para realizar las mismas, las herramientas que se utilizarán y las características que se probarán.

También se reflejan los diseños de las Pruebas de Caja Negra, de las Pruebas de Seguridad y de las Pruebas de Carga y Estrés, solo para las funcionalidades arquitectónicamente significativas.

Capítulo III: Evaluación de las pruebas del Sistema LiberMaps.

3. Introducción.

En este capítulo se realizará una evaluación de la aplicación a partir de los resultados obtenidos en el proceso de prueba. Para realizar la evaluación del sistema se tendrán en cuenta las no conformidades encontradas. En este capítulo se realiza un resumen de todos los fallos detectados cuando se aplicaron las pruebas de Caja negra, Carga, Estrés y Seguridad al Sistema LiberMaps.

En el proceso de pruebas es muy importante el registro de la documentación pertinente. Los resultados obtenidos en cada iteración de pruebas, en términos de cantidad de no conformidades por tipo deben ser registradas en un entregable para el equipo de desarrollo, donde adicionalmente se presentan hallazgos generales relevantes y recomendaciones. Como responsabilidad de este proceso está el análisis de los resultados obtenidos en cada una de las pruebas versus los resultados esperados, con base en lo cual se determina la presencia de no conformidades. (Blez, 2007)

El grupo de proyecto que desarrolla el sistema debe asimilar e interiorizar 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 y control de configuración. La retroalimentación realizada por el grupo de pruebas debe basarse en medidas del proceso de pruebas y del software, pero también en análisis cualitativos como la descripción de las causas más frecuentes de los defectos, tendencia del proceso de pruebas de software. (Blez, 2007)

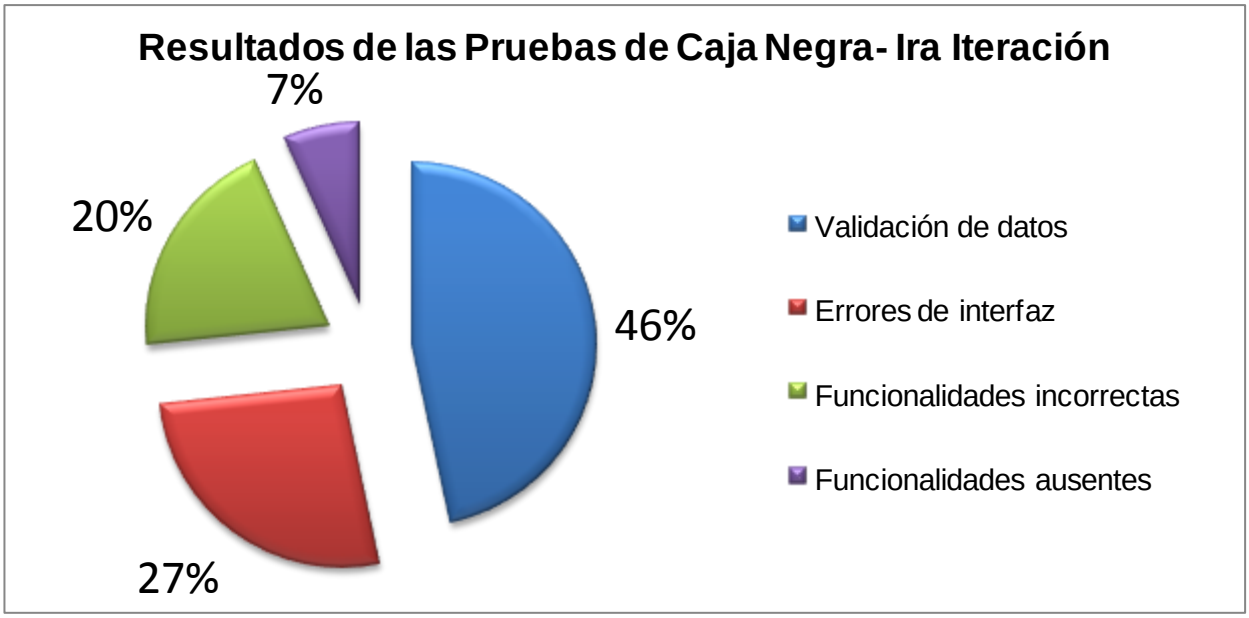
Los resultados pueden ser reflejados de formas diferentes, en este trabajo específicamente se describen por cada tipo de prueba, se hace un resumen general de los resultados obtenidos.

3.1. Resultados de la prueba de Caja Negra.

A partir del diseño de los casos de prueba, se procedió a ejecutar cada uno de ellos, introduciendo juegos de datos, tanto datos correctos como incorrectos. Al ejecutar los casos de prueba, se detectaron errores tales como: funcionalidades ausentes, errores de interfaz, no se

valida la entrada de datos incorrectos y funcionalidades incorrectas, para un total de quince No conformidades.

Resumen de No conformidades.



La gráfica que se muestra representa los resultados alcanzados en las Pruebas de Caja Negra, representando en % las No conformidades detectadas. Como se puede ver el 7% representa las funcionalidades ausentes, un 20 % simboliza a las funcionalidades incorrectas, el 27 % significa los errores de interfaz y el 46 % constituye la validación de datos. Al realizar un análisis de los resultados, se concluye que la mayor cantidad de no conformidades se refiere a la validación de datos.

Se llegó a la conclusión de que la Prueba de Caja Negra cumplió los objetivos previamente establecidos, encontrar diferentes clases de errores, para así poder eliminarlos y entregar la aplicación con la menor cantidad de defectos. Se espera que cada uno de los errores encontrados sean eliminados por el equipo de desarrollo y que una vez aplicada la segunda iteración se encuentren menos de los que se encontraron en la primera y diferentes a los ya encontrados.

3.2. Resultados de la prueba de Carga y Estrés.

Resultados de las Pruebas de Carga.

Caso de prueba "Autenticar Usuario".

Tipo de prueba: Prueba de Carga.

Durante la ejecución de la prueba el CPU se mantuvo entre el 60 y el 100 % de utilización, la memoria física estuvo por debajo del 60 % y la taza de errores se mantuvo en 0%.

Se realizó una primera prueba, para 50 usuarios con un tiempo entre conexiones de 1,3 segundos. Al terminar la prueba se lograron 1250 peticiones para 50 usuarios concurrentes que logran concluir con éxito todas las peticiones. El sistema mantuvo un rendimiento estable y se considera que son valores aceptables. (Ver Anexo 5)

Se realizó una segunda prueba, para 100 usuarios con el mismo tiempo entre conexiones. Al terminar la prueba se lograron 5000 peticiones para 100 usuarios concurrentes que lograron concluir con éxito todas las peticiones. El sistema mantuvo un rendimiento estable y se considera que son valores aceptables.

Se puede llegar a la conclusión que el sistema para esta funcionalidad soporta 100 usuarios concurrentes, presentando un buen rendimiento y un tiempo de respuesta dentro de los parámetros normales. Esto se pudo verificar aumentando el número de conexiones en el plan de prueba generado por la herramienta utilizada para esta prueba (JMeter). Esta herramienta te permite crear los planes de pruebas y luego cambiarle los datos (cantidad de conexiones, tiempo entre conexión y conexión).

Al aumentar el número de conexiones, la herramienta cargó todas las conexiones permitiendo ver los resultados, pero con una cifra mayor que esta cargaba todos los hilos pero no se podían ver los resultados, no teniendo otra alternativa que parar la ejecución de las pruebas.

Caso de prueba "Configuración de Datos Globales".

Tipo de prueba: Prueba de Carga.

Durante el período de la prueba el CPU se mantuvo entre el 95 y el 100 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores se mantuvo al 25 %. Analizando

los valores de cantidad de peticiones, y rendimiento se puede concluir que el sistema responde satisfactoriamente, pues para 100 peticiones concurrentes mantuvo un rendimiento estable en comparación con la respuesta del sistema con 50 usuarios concurrentes. Se alcanzaron 400 peticiones para 50 usuarios y 800 peticiones para 100 usuarios. (Ver Anexo 5)

Luego del análisis de los valores anteriores se llega a la conclusión de que la funcionalidad Configuración de Datos Globales cumple con los requerimientos establecidos. Se especifica que debe realizarse la revisión de los vínculos que dan error para su corrección.

Caso de Prueba "Configurar Perfil de Usuarios".

Tipo de prueba: Prueba de Carga.

Durante la ejecución de la prueba el CPU se mantuvo en el 100 % de utilización, la memoria física estuvo por debajo del 85 % y la taza de errores se mantuvo al 25 %. Analizando los valores de cantidad de peticiones, y rendimiento se puede concluir que el sistema responde satisfactoriamente pues para 80 peticiones aunque mantuvo un rendimiento inestable. Se alcanzaron 2240 peticiones para 80 usuarios concurrentes. (Ver Anexo 5)

Luego del análisis de los valores realizado anteriormente, se concluye que la funcionalidad Configurar Perfil de Usuarios cumple con los requerimientos establecidos. Aún así se especifica que debe realizarse una revisión de los vínculos que dan error para su corrección. Debe realizarse la verificación del código que se muestra a continuación pues presenta un 100 % Error.

Caso de prueba "Crear Perfiles de Funcionalidades".

Tipo de prueba: Prueba de Carga.

Durante la ejecución de las pruebas el CPU se mantuvo en el 100 % de utilidad, la memoria física estuvo por debajo de 75 % y la taza de errores se mantuvo al 16,00 %. Analizando los valores de cantidad de peticiones y el rendimiento, se puede concluir que el sistema responde satisfactoriamente, pues para 70 usuarios concurrentes mantuvo un rendimiento estable para un total de 1750 peticiones. (Ver Anexo 5)

Luego del análisis de los valores, realizado anteriormente se llega a la conclusión de que la funcionalidad Crear Perfiles de Funcionalidades cumple con los requerimientos establecidos. Aún así se especifica que debe realizarse la revisión de los vínculos que dan error para su corrección.

Caso de prueba "Crear Perfil de Usuarios".

Tipo de prueba: Prueba de Carga.

Durante la ejecución de la prueba el CPU se mantuvo en un 100 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores se mantuvo en el 27,27 %. Analizando los valores de cantidad de peticiones y rendimiento, se puede concluir que el sistema responde satisfactoriamente, pues para 70 usuarios concurrentes mantuvo un rendimiento estable en comparación a la respuesta del sistema con 50 peticiones concurrentes. (Ver Anexo 5)

Luego del análisis de los valores realizado anteriormente, se llega a la conclusión de que la funcionalidad Crear Perfil de Usuarios cumple con los requerimientos establecidos. Se especifica que debe realizarse la revisión de los vínculos que dan error para su corrección. El tiempo general de respuesta de la aplicación es correcto.

Caso de prueba "Exportar Mapfile".

Tipo de prueba: Prueba de Carga.

Durante la ejecución de las pruebas el CPU se mantuvo en el 90 % de utilización, la memoria física estuvo por debajo del 50 % y la taza de error se mantuvo al 44,44 %. Analizando los valores de cantidad de peticiones y rendimiento, se puede concluir que el sistema responde satisfactoriamente, pues para 70 peticiones concurrentes mantuvo un rendimiento estable en comparación con la respuesta del sistema para 50 usuarios concurrentes. (Ver Anexo 5)

Luego del análisis de los valores realizado anteriormente, se llega a la conclusión de que la funcionalidad Exportar Mapfile funciona correctamente. Se especifica que debe realizarse la revisión de los vínculos que dan error para su corrección.

Caso de prueba "Gestionar Mapfile".

Tipo de prueba: Prueba de Carga.

Durante la ejecución de las pruebas el CPU se mantuvo en el 100 % de utilización, la memoria física estuvo por debajo del 40 % y la taza de errores se mantuvo al 22,86 %. Analizando los valores de cantidad de peticiones y rendimiento, se puede concluir que el sistema responde satisfactoriamente, pues para 50 peticiones concurrentes mantuvo un rendimiento estable en comparación a la respuesta del sistema con 25 usuarios concurrentes. El número de peticiones total alcanzado es de 6121. (Ver Anexo 5)

Luego del análisis de los valores realizado anteriormente, se puede llegar a la conclusión de que la funcionalidad Gestionar Mapfile cumple con los requerimientos establecidos. Se especifica que debe realizarse la revisión de los vínculos que dan error para su corrección.

Caso de prueba "Gestionar METADATA del Objeto Web".

Tipo de prueba: Prueba de Carga.

Durante la ejecución de la prueba el CPU se mantuvo entre el 80 y 100 % de utilización, la memoria física estuvo por debajo del 60 % y la taza de errores se mantuvo en el 38,23 %. Analizando los valores de la cantidad de peticiones y el rendimiento, se llegó a la conclusión de que el sistema responde satisfactoriamente, pues para 90 conexiones concurrentes mantuvo un rendimiento estable en comparación a la respuesta del sistema con 100 usuarios concurrentes, para un total de 9520 peticiones. (Ver Anexo 5)

Luego del análisis de los valores realizado previamente, se llega a la conclusión de que la funcionalidad Gestionar METADATA del Mapfile cumple con los requerimientos establecidos. Se recomienda que se realice una revisión de los vínculos que dan error para su corrección.

Caso de prueba "Gestionar Objetos Layer".

Tipo de prueba: Prueba de Carga.

Durante la ejecución de la prueba el CPU se mantuvo en el 100 % de utilización, la memoria física estaba por debajo de del 75 % y la taza de errores se mantuvo al 43,75 %. Analizando los valores de cantidad de peticiones y rendimiento, se puede llegar a la conclusión de que el

sistema responde satisfactoriamente, pues para 70 peticiones concurrentes mantuvo un rendimiento estable. (Ver Anexo 5)

Luego del análisis de los valores realizado anteriormente se concluye que la funcionalidad Gestionar Objetos Layer cumple con los requerimientos establecidos. Se recomienda que se realice una revisión de los vínculos que dan error para su corrección. Aún así el tiempo de respuesta de la funcionalidad es correcto.

Caso de prueba "Gestionar objetos OUTPUTFORMAT del Mapfile".

Tipo de prueba: Prueba de Carga.

Durante la ejecución de la prueba el CPU se mantuvo en el 100 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores se mantuvo en el 33,6 %. Analizando los valores de cantidad de peticiones y rendimiento, se concluye que la aplicación responde satisfactoriamente, manteniendo un rendimiento estable para 80 conexiones concurrentes. (Ver Anexo 5)

Luego de un análisis de los valores realizado anteriormente, se llega a la conclusión de que la funcionalidad Gestionar Objetos OUTPUTFORMAT del Mapfile cumple con los requerimientos especificados. Se recomienda que se le realice una revisión de los vínculos que dan error para ser corregidos.

Caso de prueba "Gestionar Objetos SYMBOL".

Tipo de prueba: Prueba de Carga.

Durante la ejecución de la prueba el CPU se mantuvo en el 100 % de utilización, la memoria física estuvo por debajo del 65 % y la taza de error se mantuvo en el 39,29 %. Analizando los valores de cantidad de peticiones y el rendimiento, se puede llegar a la conclusión de que el sistema responde satisfactoriamente, aunque el % de error es relativamente elevado. La aplicación muestra un rendimiento estable para 70 usuarios concurrentes. (Ver Anexo 5)

Luego del análisis de los valores, realizado anteriormente, se concluye que la funcionalidad Gestionar Objetos Symbol cumple con los requerimientos establecidos. Se especifica que debe

realizarse la revisión de los vínculos que dan error para su corrección. Con todo esto, se llega a la conclusión de que el tiempo general de respuesta de la funcionalidad es correcto.

Caso de prueba "Gestionar Roles".

Tipo de prueba: Prueba de Carga.

Durante la ejecución de la prueba el CPU se mantuvo en el 100 % de utilización, la memoria física estuvo por debajo del 60 % y la taza de error se mantuvo en el 26,26 %. Analizando los valores de cantidad de peticiones y el rendimiento, se puede concluir que el sistema responde satisfactoriamente a pesar de los errores que muestra. Mantuvo un rendimiento estable para 100 usuarios concurrentes. (Ver Anexo 5)

A partir del análisis de los valores, realizado anteriormente, se llega a la conclusión de que la funcionalidad Gestionar Roles cumple con los requerimientos especificados. Aún así se recomienda realizar una revisión de los vínculos que dan error para su corrección. Con todo esto, el tiempo general de respuesta de la funcionalidad es correcto.

Caso de prueba "Gestionar Usuarios del Sistema".

Tipo de prueba: Prueba de Carga.

Durante la ejecución de las pruebas el CPU se mantuvo entre el 70 y el 99 % de utilización, la memoria física estuvo por debajo del 85 % y la taza de error se mantuvo en el 30,43 %. Analizando los valores de cantidad de peticiones y rendimiento, se puede concluir que el sistema responde satisfactoriamente aunque arroja gran cantidad de vínculos con errores, con 70 peticiones concurrentes mantuvo un rendimiento de 1,2 segundos lo cual se considera un buen rendimiento. (Ver Anexo 5)

Luego del análisis de los valores anteriormente, se concluye que la funcionalidad Gestionar Usuarios del Sistema cumple con los requerimientos establecidos. Se le recomienda realizar una revisión de los vínculos que dan error para su corrección. El tiempo general de respuesta de la funcionalidad es correcto.

Caso de prueba "Modificar Objeto Web".

Tipo de prueba: Prueba de Carga.

Durante la ejecución de la prueba el CPU se mantuvo entre el 50 y el 99 %, la memoria física se mantuvo entre el 52 % y el 90 % y la taza de error se mantuvo en el 50 %. Esta funcionalidad es la primera que alcanza el 50 % de errores, por lo cual es la que más riesgo representa. Analizando los valores de cantidad de peticiones y rendimiento, se llega a la conclusión de que el sistema a pesar de arrojar errores responde satisfactoriamente con un rendimiento de 5,7 segundos. (Ver Anexo 5)

Luego de hacer un análisis de los valores, se puede concluir que la funcionalidad Modificar Objeto Web cumple con los requerimientos establecidos. Se especifica que se debe realizar una revisión de los vínculos que dan error para su corrección. En sentido general el tiempo de respuesta de la funcionalidad es correcto.

Caso de prueba "Modificar Objeto".

Tipo de prueba: Prueba de Carga.

Durante la ejecución de la prueba el CPU se mantuvo entre el 40 y el 99 % de utilización, la memoria física estuvo por debajo del 89 % y la taza de error se mantuvo en el 51,92 %. Es esta la funcionalidad más crítica pues supera el 50 % de error. Con todo esto se puede decir que el sistema responde satisfactoriamente, pues para 100 peticiones concurrentes mantuvo un rendimiento estable logrando alcanzar todas las peticiones. (Ver Anexo 5)

Luego del análisis de los valores realizado, se llega a la conclusión de que la funcionalidad cumple con las especificaciones del cliente. Se recomienda que se revisen los vínculos que dan error para su corrección pues de esta forma la funcionalidad tendrá un mejor rendimiento.

Resultados de la Prueba de Estrés.

Caso de prueba "Modificar Objeto".

Tipo de prueba: Prueba de Estrés.

Funcionalidades:

Funcionalidad(es)
Modificar objeto MAP.
Modificar objeto LEGEND.
Editar LABEL.
Modificar objeto SCALEBAR.
Modificar objeto REFERENCE.
Modificar objeto QUERYMAP.

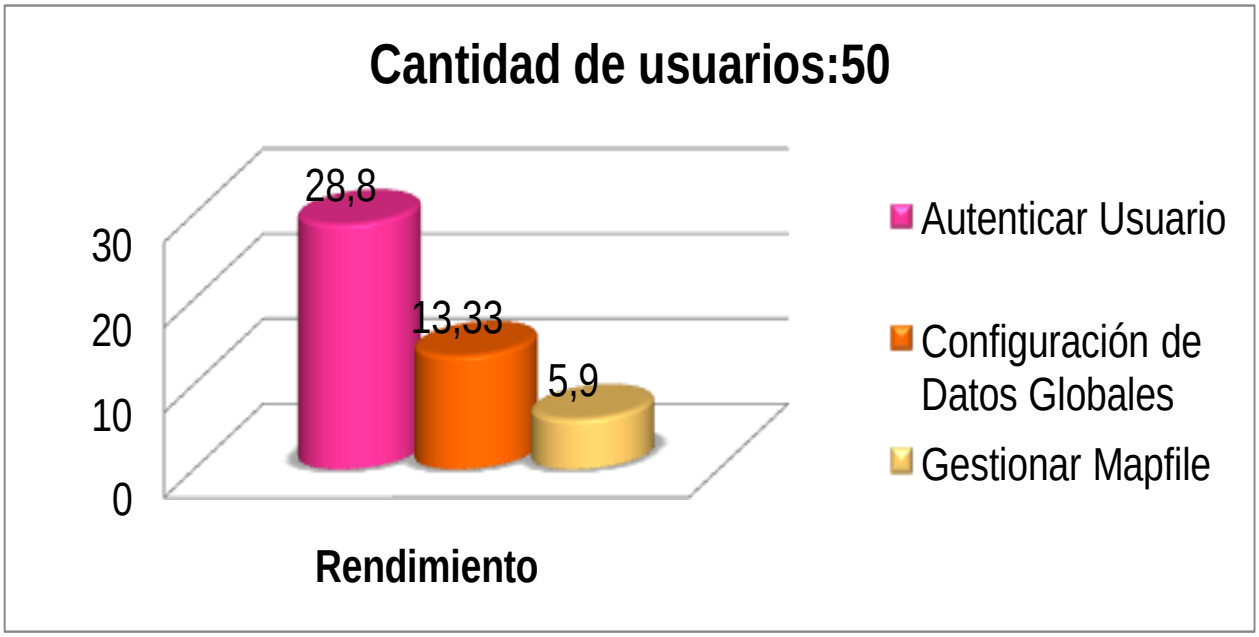
Variables	**Valores**
Número de usuarios concurrentes: (número de hilos)	120
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Al aplicar la Prueba de Estrés se pudo comprobar que la aplicación soporta entre 1 y 100 conexiones, con un número que esté por encima de este rango la aplicación entra en el estado que llamamos estrés. Esto se pudo verificar aumentando el número de conexiones en el plan de prueba de este caso de uso y ejecutándolo luego en la herramienta utilizada para esta prueba (JMeter). Esta herramienta permite crear los planes de pruebas y luego cambiarle los datos (cantidad de conexiones, tiempo entre conexión y conexión) y ejecutarlos para ver el resultado.

Al aumentar el número de conexiones, la herramienta cargó todas las conexiones pero no se puedo determinar el rendimiento pues se bloqueó y solo se puedo parar la ejecución del plan. Por estas razones se llegó a la conclusión de que el número de conexiones límites para que la aplicación haga estrés es 120.

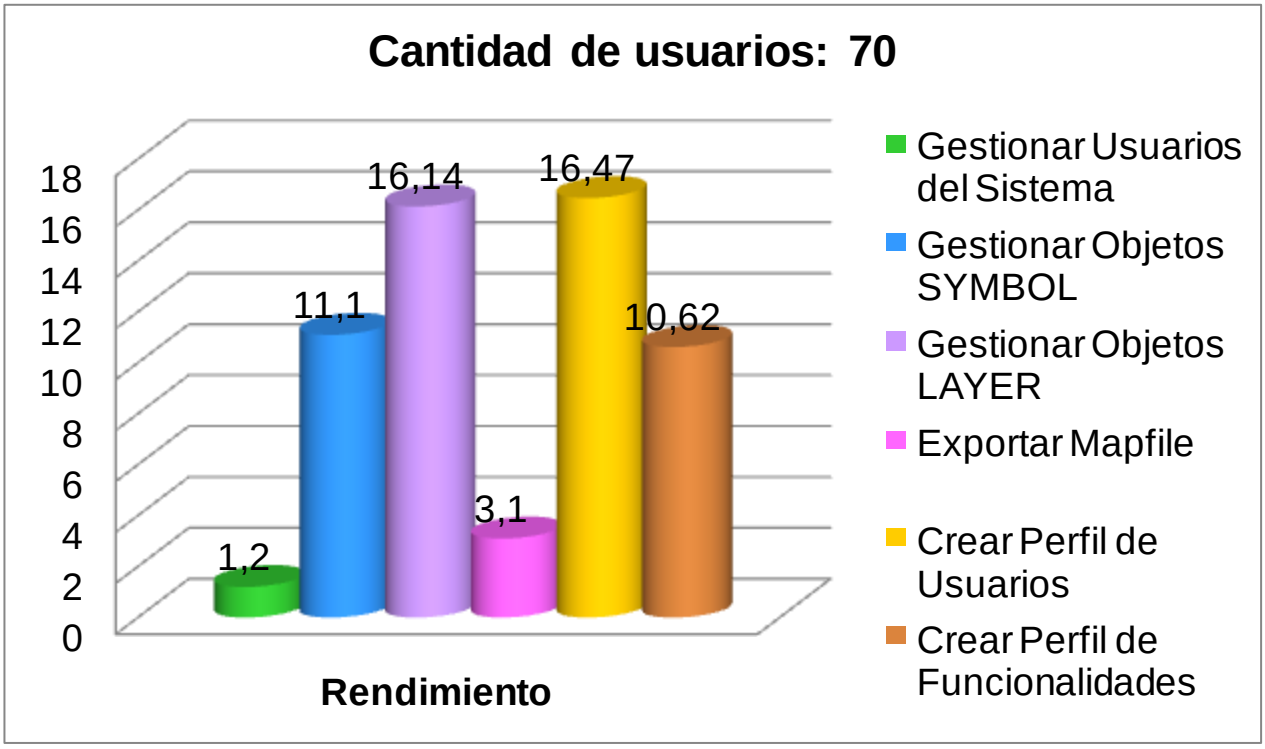
Resultados generales de rendimiento para Pruebas de Carga.

Rendimiento para 50 usuarios concurrentes.



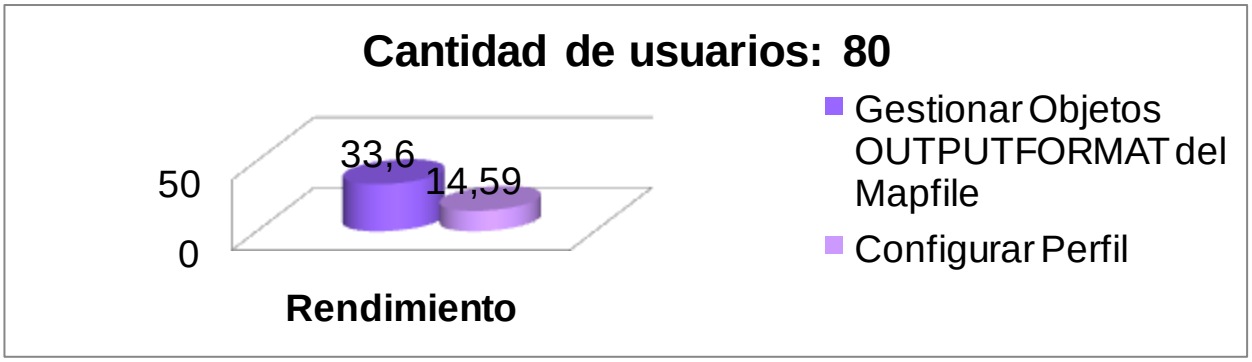
Esta gráfica representa el rendimiento alcanzado por la aplicación para 50 usuarios concurrentes. Para el caso de uso Autenticar Usuario se obtuvo un rendimiento de 28.8 segundos, para la Configuración de Datos Globales el rendimiento fue de 13.33 segundos y el caso de uso Gestionar Mapfile alcanzó un rendimiento de 5.9 segundos.

Rendimiento para 70 usuarios concurrentes.



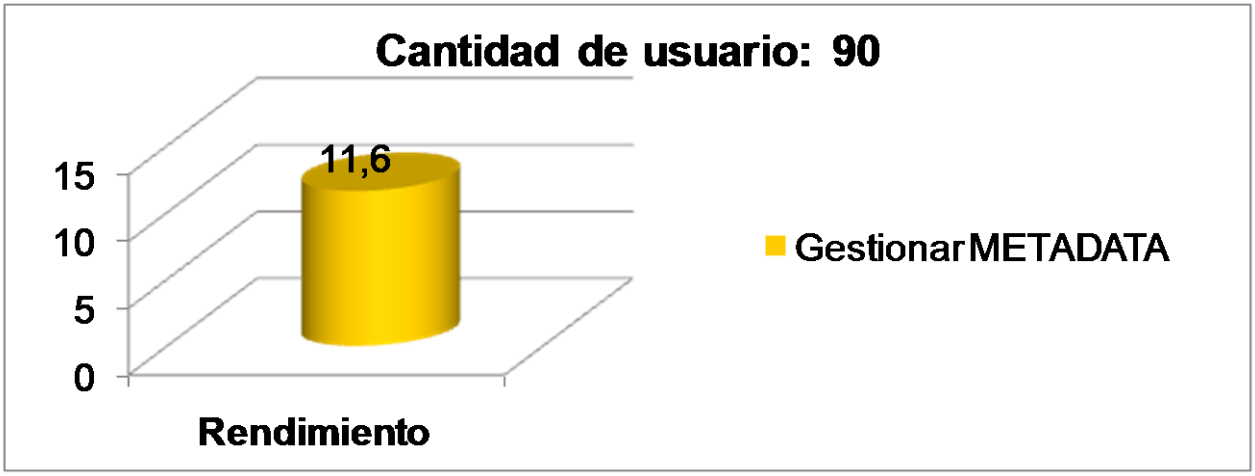
Esta gráfica muestra el rendimiento alcanzado por la aplicación para un total de 70 usuarios concurrentes. El caso de uso Gestionar Usuarios del Sistema alcanzó un rendimiento de 1.2 segundos, el caso de uso Exportar Mapfile obtuvo 3.1 segundos, el caso de uso Crear Perfil de Funcionalidades tuvo un rendimiento de 10.62 segundos, el caso de uso Gestionar Objetos SYMBOL tuvo un rendimiento de 11.1 segundos, el caso de uso Gestionar Objetos LAYER alcanzó un rendimiento 16.14 segundos y el caso de uso Crear Perfil de Usuarios con un rendimiento de 16.47 segundos.

Rendimiento para 80 usuarios concurrentes.



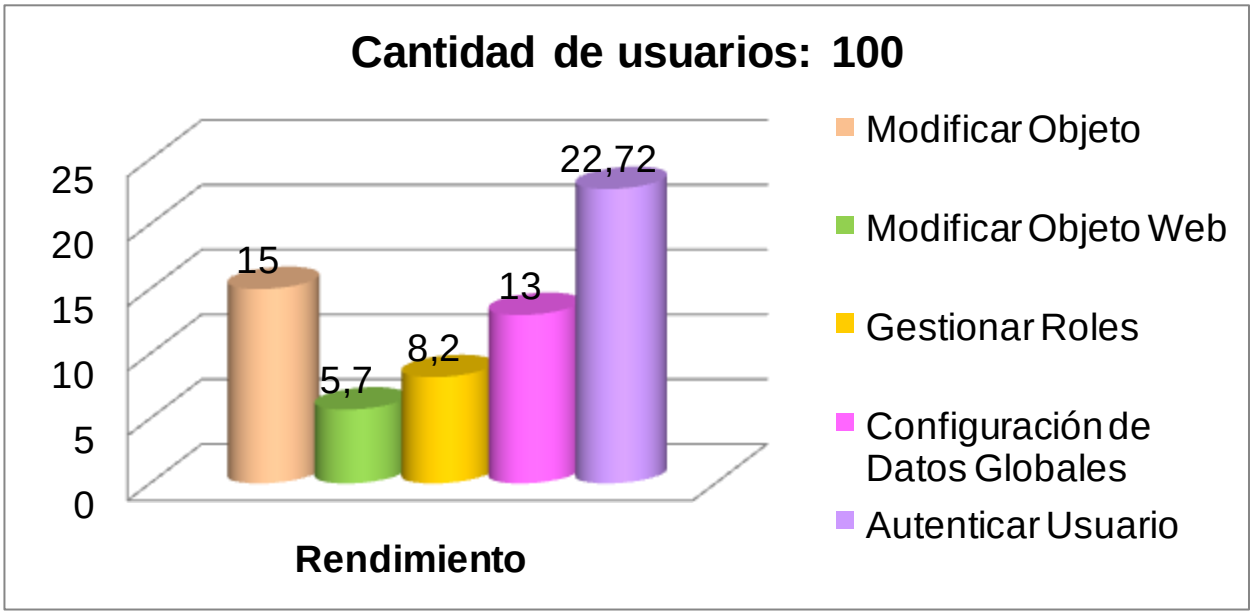
Para un total de 80 usuarios concurrentes el caso de uso Gestionar Objetos OUTPUTFORMAT del Mapfile obtuvo un rendimiento de 33.6 segundos y el caso de uso Configurar Perfil alcanzó un rendimiento de 14.59 segundos.

Rendimiento para 90 usuarios concurrentes.



Para un total de 90 usuarios concurrentes, el caso de uso Gestionar METADATA alcanzó un rendimiento de 11.6 segundos.

Rendimiento para 100 usuarios concurrentes.



Para 100 usuarios concurrentes, el caso de uso Modificar Objeto tuvo un rendimiento de 15 segundos, Modificar Objeto Web alcanzó un rendimiento de 5.7 segundos, Gestionar Roles obtuvo un rendimiento de 8.2 segundos, Configuración de Datos Globales adquirió un rendimiento de 13 segundos y el caso de uso Autenticar Usuario tuvo un rendimiento de 22.72 segundos.

Al realizar un análisis de los valores de rendimientos alcanzados por la aplicación, se puede llegar a la conclusión de que aunque existen algunos casos, por ejemplo el caso de uso Gestionar Roles, donde el rendimiento no es el más optimo, se considera que el rendimiento de la aplicación de forma general es bueno y está incluido dentro del rango 1.2 y 34 segundos. Para llegar a la conclusión de que es un buen rendimiento se tuvo en cuenta la cantidad de usuarios y el número máximo de rendimiento alcanzado. Para 90 usuarios se obtuvo un rendimiento de 11.6 segundos, siendo este rendimiento menor que el que se obtuvo para 50 usuarios. Además se puede concluir que la aplicación tiene soporte para un rango entre 100 y 110 conexiones. Con un número de conexiones mayor (120 conexiones), la herramienta se torna lenta y nunca para de ejecutarse, como si estuviera en un tiempo infinito de espera, esto es lo que llamamos estrés.

3.3. Resultados de la Prueba de Seguridad.

Luego de aplicar las Pruebas de Comprobación del Sistema de Autenticación y para acreditar el sistema de autenticación de la aplicación se pudo probar que el sistema LiberMaps es sensible a la entrada de usuarios. De los usuarios y contraseñas que se probaron al cargarlos en la herramienta Brutus todos permitieron penetrar, lo cual demuestra que existen deficiencias en el mecanismo de seguridad de la aplicación.

Las Pruebas de gestión de caché de navegación y de salida de sesión evidenciaron que la aplicación no cierra la sesión de un usuario cuando ha estado inactivo durante un lapso de tiempo, exponiendo a la aplicación a posibles penetraciones usuarios maliciosos. Además se pudo comprobar con las Pruebas de Gestión de sesiones que al no existir un botón de cierre en la aplicación las cookies no son eliminadas y ellas constituyen un vector de ataque para los usuarios maliciosos, esto a la vez constituye una vulnerabilidad del sistema y un atacante podría robar la sesión del usuario.

Luego de haber explicado el resultado de las Pruebas de Seguridad se puede concluir que el Sistema LiberMaps es vulnerable y que el sistema de seguridad implantado es insuficiente. Se le recomienda corregir cada una de las vulnerabilidades descritas y someter la aplicación a nuevas verificaciones.

3.4. Conclusiones parciales.

Con este capítulo concluye el proceso de prueba diseñado y aplicado al sistema LiberMaps, durante el mismo se pudieron obtener los resultados de las pruebas que se aplicaron mostrando la efectividad de las mismas pues se pudieron encontrar errores. De esta forma podrán ser corregidos los errores y entregar la solución informática con la menor cantidad de errores. Se prevé que los errores encontrados pueden solucionarse en un tiempo corto período de tiempo, pues están debidamente documentados, para una rápida comprensión del equipo de desarrolladores.

Luego de haber terminado de probar el sistema LiberMaps, se puede afirmar que no cumple con las normas de calidad por las que se rige la universidad para realizar la entrega del producto al cliente, al contener errores. Se da así por concluida la primera iteración del proceso de pruebas,

de forma clara y entendible, documentándose todos los defectos, siendo de gran ayuda para el equipo de desarrollo en posteriores versiones.

Conclusiones

Conclusiones Generales.

Entregar productos con calidad es la principal exigencia en la industria del software, por lo que se hace necesario el desarrollo de un proceso de pruebas que se encargue de validar la calidad de las soluciones informáticas. Por estas razones al sistema LiberMaps, se le aplicaron pruebas para descubrir aquellos errores que no se habían detectado durante el desarrollo del mismo. En el transcurso de este proceso se comprendió la importancia de las Pruebas de Caja Negra en el diseño de casos de pruebas certeros capaces de descubrir errores.

✓ Mediante la confección del Plan de Pruebas se logró definir los pasos a seguir para realizar cada una de las pruebas escogidas dentro de la estrategia; se determinó el ambiente donde se desarrolló el proceso, además este plan permitió planificar este proceso teniendo en cuenta el tiempo y los recursos disponibles.

✓ La ejecución de las Pruebas de Carga y Estrés utilizando la herramienta JMeter, permitió elaborar una plantilla resumen con los resultados de las mismas, para ser entregado al equipo de desarrolladores.

✓ La ejecución de las Pruebas de Seguridad utilizando la herramienta Brutus, permitió elaborar una plantilla resumen con los resultados obtenidos en este proceso, para ser entregada al equipo de desarrollo para fomentar la seguridad de la aplicación.

✓ Con la aplicación de este proceso de pruebas se pudo llegar a la conclusión que el sistema no cumple con la calidad requerida para ser entregada al cliente, pues aunque tiene un buen rendimiento, posee errores y vulnerabilidades en cuanto a la seguridad de la información.

✓ Con la aplicación del proceso de prueba se identificaron diferentes clases de errores, lo cual demuestra que se le dieron cumplimiento a los objetivos de las pruebas, basados en los principios de Glenford J. Myers, que planteaba que: la prueba es un proceso de ejecución de un programa con la intensión de detectar un error.

Recomendaciones.

Una vez concluido el proceso de pruebas al Sistema LiberMaps y de ver la importancia del mismo para la detección de errores a tiempo, se recomienda:

- ✓ Que los desarrolladores corrijan los errores detectados en un corto período de tiempo.
- ✓ Una vez realizada la corrección de los defectos, realizar otras iteraciones con el fin de verificar si fueron corregidos los errores anteriores.
- ✓ Realizar un estudio de las pruebas que no son aplicadas en este procedimiento para su posterior aplicación.
- ✓ Hacer uso de este trabajo de diploma como bibliografía para posteriores trabajos de diplomas, además de que puede servir como complemento para la asignatura de Ingeniería de Software, que aborda temas de calidad, en la Universidad de Ciencias Informáticas.

Bibliografía

Bibliografía Referenciada.

Ana Sanz, Javier Saldaña, Javier García, Domingo Gaitero. 2008. TestPAI: Un área de proceso de pruebas integrada con CMMI. Madrid: s.n., 2008.

Barrio, Francisco José Sáez. 2010. La calidad de las aplicaciones. 2010.

Beizer, Boris. 1995. Black-Box Testing. 1995.

Blez, Lesdey Saavedra López y Yohandris Pupo. 2007. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. Diseño y aplicación de pruebas al producto Registro. Habana: s.n., 2007.

Bolívar, Universidad Simón. 2001. El Plan de Prueba. 2001.

Brito, Irina Napal y Irina. 2003. Las pruebas de software, su aplicación al Config.CASE.Tesis presentada en opción al título de Ingeniero Informático. Ciudad de la Habana: s.n., 2003.

Cárdenas, Maria Clara Choucair. 2007. Pruebas de software ¿la salvación, un proceso sin utilidad, trivial, simplemente una moda, o...? Bogotá: s.n., 2007.

Carrillo, Luis Vinicio León. 2005. El Contexto de la Prueba de Software. Probar para incrementar la calidad. 2005.

Chapin, Ned. 1999. Types of software evolution and software maintenance. 1999.

Cortés, Oscar Hernando Guzmán. 2003-2004. Aplicación práctica del diseño de pruebas de software a nivel de programación. 2003-2004.

Cueva, Juan Manuel. 1999. Calidad del Software. 1999.

Edumilis Méndez, María Pérez, Luis E. Mendoza. 2007. Aplicación de un Método para Especificar Casos de Prueba de Software en la administración pública. Caracas : s.n., 2007.

Frida. 2009. informaticafrida.blogspot.com. informaticafrida.blogspot.com. [En línea] 2009. [Citado el: 28 de Noviembre de 2010.] http://informaticafrida.blogspot.com.

Gracia, Joaquin. 2005. CMM - CMMI. 2005.

1990. IEEE Standard Glossary of Software Engineering Terminology. 1990.

Bibliografía

1997. IEEE Standard Glossary of Software Engineering Terminology. 1997.

Isabel Blank, Larissa Herrera, Miguel Ortiz. 2005. Pruebas Funcionales. 2005.

Kaner, C. 2006. 2006.

Lamancha, Beatriz Pérez. 2007. Gestión de las Pruebas Funcionales. Montevideo : s.n., 2007.

Lovelle, Juan Manuel Cueva. 1999. Calidad del Software. 1999.

LuchOO. 2010. artehack.net. artehack.net. [En línea] Mayo de 2010. http://artehack.net.

Mateo, Pedro Reales. 2011. Calidad de casos de prueba. 2011.

Matías Fuentes Contreras, Pablo Teuber Henríquez. 2008. Modelo CMMI. Santiago de Chile : s.n., 2008.

Myers, Glenford J Myers. 2004. The art of sotware Testing. 2004.

Myers, Glenford J. 1979. The art of sotware Testing. 1979.

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

Pressman, Roger S. 2002. Ingeniería del software, un enfoque práctico. 2002.

Sáez, Francisco José. 2010. Las Pruebas y la Calidad del Software,ahora una necesidad más que nunca. 2010.

Sanz, Luis Fernández. 2008. La importancia de la calidad del software. 2008.

Sommerville, Ian. 2005. "Ingeniería del Software", Capítulo 23. 7ª Edición, Pearson-Addison Wesley. 2005.

Soto, Lauro. 2010. www.mitecnologico.com. www.mitecnologico.com. [En línea] 2010. http://www.mitecnologico.com.

Toledo, Alvaro. 2009. Uptodown.com. Uptodown.com. Uptodown.com. Uptodown.com. [En línea] 2009.

Turing, Alan. 1936. Los números computables. 1936.

Bibliografía

Tuya, Javier. 2007. Las Pruebas del software. 2007.

Vence, Jose Maria Perez. 2009. Plan de pruebas de integracion. Madrid : s.n., 2009.

Vergara, Kervin. 2007. Concepto y tipos de software: programas, definición. 2007.

Vivanco, Pablo. 2010. www.vivanco.com.mx. www.vivanco.com.mx. [En línea] 2010.

zonajava. 2005. Métodos de prueba del software. 2005.

Estratecgias. 2008. www.estratecgias.com. www.estratecgias.com. [En línea] 2008. http://www.estratecgias.com/page/pruebas_de_seguridad.

Noa, Eliane Paz. 2007. Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas, Diseño y aplicación de pruebas al producto “Árbol Genealógico". Habana : s.n., 2007.

Phadke, M.S. 1997. Planning efficient software tests. . 1997.

Santanach, Juan Emilio Vargas y Alejandro. 2010. Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas, Método de Estimación de tiempo y esfuerzo para las pruebas de liberación, aceptación y piloto de los proyectos de desarrollo de software de la Universidad de las Ciencias . Habana: s.n., 2010.

Murcia, Universidad de. 2006. www.um.es. www.um.es. [En línea] 30 de Diciembre de 2006. http://www.um.es/docencia/barzana/IAGP/Iagp2.html.

Glosario de Términos

Glosario de términos.

LiberMaps: nombre identificativo de la Plataforma Soberana que brinda servicios de catálogo, creada con el fin de gestionar propiedades de los mapas y configurarlos.

MAP (mapa): representación gráfica abstracta de la superficie terrestre que despliega las relaciones espaciales entre los elementos.

Mapfile: componente muy importante de UMN MapServer, es un archivo con extensión ".map" en formato texto, que contiene todas las definiciones y configuraciones iniciales necesarias para la ejecución de un servidor de mapas UMN MapServer.

Archivo de configuración de la aplicación. El Mapfile también incluye información sobre como dibujar el mapa, la leyenda, y los mapas resultantes desde una consulta.

Feedback: nos permite establecer un seguimiento y control de las acciones de comunicación con el fin de comprobar si se cumplen los objetivos previstos y, en su defecto, corregir las desviaciones existentes.

MapServer: servidor de mapas desarrollado con tecnologías libres, compatible con frameworks como Cartoweb y con tecnologías de representación de mapas como OpenLayers.

Metadato: describen el contenido, la calidad, el formato y otras características que lleva asociadas un recurso, constituyendo un mecanismo para caracterizar datos y servicios de forma que usuarios (y aplicaciones) puedan localizarlos y acceder a ellos.

Outputformat: formato de salida.

Symbol: símbolo.

Layer (capa): forman la base de una técnica de programación HTML dinámico y específica para Netscape 4.

Objeto SCALEBAR: herramienta para mostrar la escala de un mapa. Define como se construirá la escala gráfica.

Objeto REFERENCE: define como será creado el mapa de referencia**.**

Glosario de Términos

QUERYMAP: define como serán presentados los resultados al momento de hacer una consulta.

LEGEND: herramienta para mostrar la leyenda de un mapa.

Objeto web: define como operará la interfaz web.

Label: es usado para colocar una etiqueta en el mapa, a partir de datos alfanuméricos.

Catálogo: es la relación ordenada de elementos pertenecientes al mismo conjunto, que por su número precisan de esa catalogación para facilitar su localización.

OpenGIS: estándar internacional orientado a sistemas de información geográfica (SIG).

Estrategia de prueba: vela porque el producto de software que se está construyendo o modificando, reúna los requerimientos de lógica del negocio que el cliente ha pedido realizar mediante el debido contrato de desarrollo de software. A continuación se hablará más detalladamente acerca de la estrategia de pruebas. Son las líneas guía del equipo de pruebas de un proyecto de desarrollo de software.

Anexos.

Anexo 1 Comparación de las pruebas de software.

Tipo de prueba	Funcional
Descripción	Es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. Las pruebas funcionales se hacen mediante el diseño de modelos de prueba que buscan evaluar cada una de las opciones con las que cuenta el paquete informático.
Artefacto de entrada	Especificación de requerimientos. Especificación de CU del Sistema. Documento de Arquitectura SW. Diseño de Caso de Prueba.
Artefacto a probar	Aplicación Web. Aplicación Desktop. Multimedia. Aplicaciones en ambientes Reales.
Caso de prueba	Diseño de Caso de Prueba (CP) basado en Caso de Uso (CU).

Tipo de prueba	Seguridad
Descripción	Prueba centrada en asegurar que los datos o sistemas que son objetos de prueba, son accedidos sólo por los actores que tienen permiso para hacerlo.
Artefacto de entrada	Especificación de CU del Sistema. Especificación de roles y responsabilidades. Especificación de Requisitos No Funcionales.
Artefacto a probar	Aplicación Web.

	Aplicación Desktop. Multimedia. Aplicaciones en Ambientes Reales
Caso de prueba	Se hará una plantilla estándar para las herramientas que se propondrá en el laboratorio de Calidad.

Tipo de prueba	Volumen
Descripción	Prueba centrada en verificar las habilidades de los objetos de prueba para manejar grandes cantidades de datos, tanto en entrada como en salida, o residente en la base de datos
Artefacto de entrada	Especificación de Requisitos No Funcionales. Modelo Entidad Relación. Modelo de Datos.
Artefacto a probar	Aplicación Web. Aplicación Desktop. Multimedia. Aplicaciones en Ambientes Reales.
Caso de prueba	Se hará una plantilla estándar para las herramientas que se propondrá en el laboratorio de Calidad.

Tipo de prueba	Usabilidad
Descripción	Prueba encaminada a factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente

	documentación de usuarios y materiales de entrenamiento.
Artefacto de entrada	Manual de Usuario Manual de Instalación. P1 y PA de Cursos de Capacitación. Aplicación.
Artefacto a probar	Aplicación Web. Aplicación Desktop. Multimedia. Aplicaciones en Ambientes Reales. Especificación de Requisitos Funcionales. Especificación de Requisitos No Funcionales.
Caso de prueba	Lista de chequeo usabilidad. Lista de chequeo manual de usuario. Lista de chequeo manual de instalación. Lista de chequeo curso de capacitación.

Tipo de prueba	Estrés
Descripción	Enfocada a evaluar cómo el sistema responde bajo condiciones anormales. (Extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible).
Artefacto de entrada	Especificación de Requisitos No Funcionales. Documento de Arquitectura SW.

Artefacto a probar	Portal Web. Aplicación Web. Aplicación Desktop. Multimedia. Aplicaciones en Ambientes Reales.
Caso de prueba	Se hará una plantilla estándar para las herramientas que se propondrá en el laboratorio de Calidad.

Tipo de prueba	Carga
Descripción	Usada para validar y valorar la aceptabilidad de los límites operacionales de un sistema bajo carga de trabajo variable, mientras el sistema bajo prueba permanece constante.
Artefacto de entrada	Especificación de Requisitos No Funcionales. Documento de Arquitectura SW.
Artefacto a probar	Portal Web. Aplicación Web. Aplicación Desktop. Multimedia. Aplicaciones en Ambientes Reales.
Caso de prueba	Se hará una plantilla estándar para las herramientas que se propondrá en el laboratorio de Calidad.

Anexo 2 Casos de Pruebas de Caja Negra.

Procedimientos de Pruebas de Caja Negra.

CU Modificar Objeto.

Descripción general.

El caso de uso inicia cuando el usuario desea modificar objetos ya sea MAP, LEGEND, SCALEBAR, REFERENCE o QUERYMAP, y finaliza cuando este es modificado, guardando los cambios en el sistema.

Condiciones de ejecución.

El usuario debe estar autenticado y poseer los privilegios necesarios para acceder a esta opción.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: "Modificar objeto MAP"	EC 1.1: "Modificar objeto MAP con éxito"	El usuario selecciona la opción "MAP" del mapa que desea modificar, el sistema muestra los datos del Mapfile para su modificación, el usuario modifica los datos que desea, incluyendo la proyección y selecciona la opción guardar, el sistema guarda los cambios en la base de datos.
	EC 1.2: "Modificar objeto MAP incorrectamente"	El usuario deja en blanco los campos NAME, UNITS, MIN X, MIN Y, MAX X, MAX Y, IMAGECOLOR. El sistema muestra una ventana "Aviso" donde especifica que el formulario no es válido.
	EC 1.3: "Vista previa"	El usuario escoge la opción "Vista previa", el sistema muestra una ventana "Vista previa de exportación" brindándole al usuario como quedaría su fichero.
SC 2: "Modificar	ES 2.1: "Modificar objeto	El usuario elige la opción "LEGEND" del mapa que desea modificar para editar sus datos, el

objeto LEGEND”	LEGEND”	sistema muestra una ventana “Propiedades del LEGEND”, el usuario escoge la opción “Modificar LEGEND” donde introduce los datos en el sistema para modificar un objeto LEGEND y presiona el botón Guardar, el sistema guarda los datos introducidos por el usuario y muestra un mensaje para que este compruebe que ha sido modificado el LEGEND.
	EC 2.2: “Editar Label ”	El usuario selecciona la opción “Editar Label”, el sistema muestra una ventana con las propiedades del Label, el usuario introduce los datos para actualizar LABEL y presiona el botón Guardar, el sistema guarda los cambios realizados por el usuario.
	EC 2.3: “Vista previa”·	El usuario escoge la opción “Vista previa”, el sistema muestra una ventana “Vista previa de exportación” brindándole al usuario como quedaría su fichero.
	EC 2.4: “Cancelar Editar Label”.	El usuario da clic en el botón “Cerrar”. El sistema cierra la ventana “Propiedades de la sección LABEL” y retorna al usuario a la ventana “Propiedades de la sección LEGEND”.
SC 3: “Modificar objeto SCALEBAR”	EC 3.1: “Modificar objeto SCALEBAR ”	El usuario escoge la opción “Modificar SCALEBAR”, el sistema muestra la ventana “Propiedades de Scalebar” para que el usuario pueda modificar los datos que desea, el usuario introduce los datos necesarios para cambiar el SCALEBAR y da clic en el botón guardar, el sistema guarda los datos

		introducidos por el usuario.
	EC 3.2: "Editar Label"	El usuario da clic en la opción "Editar Label", el sistema muestra una ventana "Propiedades del Label" para que el usuario edite el Label, el usuario introduce los datos para editar un nuevo Label y presiona el botón Guardar, el sistema guarda los cambios realizados por el usuario.
	EC 3.3: "Vista previa"	El usuario selecciona la opción "Vista previa", el sistema carga la vista previa y muestra los datos de la vista.
	EC 3.4: "Cancelar Editar label"	El usuario da clic en el botón cerrar de la ventana "Propiedades de la sección LABEL". El sistema regresa al usuario a la ventana "propiedades de la sección SCALEBAR".
SC 4: "Modificar objeto REFERENCE"	EC 4.1: "Modificar objeto reference con éxito"	El usuario selecciona la opción "reference", el sistema muestra una venta "Propiedades del objeto reference", el usuario entra los datos en el sistema para modificar las propiedades del objeto REFERENCE y presiona el botón Guardar, el sistema guarda los datos introducidos por el usuario y muestra un mensaje para que este compruebe que han sido modificado correctamente.
	EC 4.2: "Modificar objeto REFERENCE incorrectamente"	El usuario deja campos en blancos tales como: MIN X, MIN Y, MAX X, MAX Y. El sistema muestra una ventana de error especificando que el formulario no es válido y marca en rojo los campos requeridos. El usuario da clic en el botón "OK". El sistema retorna al usuario a la

		ventana “Propiedades de la sección REFERENCE”.
	EC 4.3: “Vista previa”	El usuario selecciona la opción “Vista previa”, el sistema crea la vista previa y le muestra al usuario la vista creada.
SC 5: “Modificar objeto QUERYMAP”	EC 5.1: “Modificar objeto QUERYMAP”	El usuario selecciona la opción “QUERYMAP”, el sistema muestra una ventana con las propiedades del querymap, el usuario entra los datos en el sistema para modificar un QUERYMAP y presiona el botón guardar, el sistema guarda los datos introducidos por el usuario.
	EC 5.2: “Vista previa”	El usuario selecciona la opción “Vista previa”, el sistema muestra una venta “Vista previa de exportación” brindándole al usuario como quedaría su fichero.

CU Modificar Objeto Web.

Descripción general.

El caso de uso inicia cuando el usuario selecciona la opción “Navegación” y dentro de la misma selecciona el “Mapfile” al que le va a modificar el objeto WEB, selecciona la opción “WEB“ y termina cuando el usuario finaliza la operación guardando los cambios.

Condiciones de ejecución.

El usuario debe estar autenticado y poseer los privilegios necesarios para acceder a esta opción.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad

SC 1: “Modificar objeto web”	EC 1.1: “Modificar objeto web ”	El usuario selecciona la opción “WEB”, el sistema muestra todas las propiedades del objeto web, el usuario modifica todos los campos del objeto web que desea modificar, el sistema guarda los cambios realizados en el objeto web y actualiza la información.
	EC 1.2 “Editar METADATA ” Ver DCP Gestionar METADATA del objeto WEB	
	EC 1.3 “Vista previa”	El usuario da clic en el botón “Vista Previa”. El sistema muestra una ventana “Vista previa de exportación” donde le muestra al usuario como quedará el objeto.

CU Gestionar METADATA del objeto WEB.

Descripción general.

El caso de uso se inicia cuando el usuario selecciona la opción “Navegación” y dentro de la misma selecciona el “Mapfile” al que le va a modificar el objeto WEB, selecciona la opción “Editar METADATA“ y termina cuando el usuario finaliza la operación guardando los cambios.

Condiciones de ejecución.

El usuario debe estar autenticado y poseer los privilegios necesarios para acceder a esta opción.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad

SC 1: "Editar METADATA"	EC 1,1: "Editar METADATA"	El usuario selecciona la opción Editar METADATA. El sistema muestra la ventana Propiedades de la sección METADATA. El usuario introduce los datos requeridos y da clic en el botón Guardar. El sistema guarda los datos en la base de datos y los actualiza.
	EC 1,2: "Cancelar Editar METADATA"	El usuario da clic en el botón cerrar de la ventana "Propiedades de la sección METADATA". El sistema cierra la ventana y retorna al usuario a la ventana "Propiedades de la sección WEB".
	EC 1,3: "Vista previa"	El usuario da clic en el botón Vista previa. El sistema muestra una ventana "Vista previa de exportación" con los datos del objeto WEB.
SC 2: " Crear nuevo metadato"	EC 2,1: "Crear nuevo metadato con éxito"	El usuario selecciona la opción "Nuevo metadato". El sistema muestra una ventana "Propiedades de la sección METADATA". El usuario introduce los datos requeridos y da clic en el botón "Guardar". El sistema guarda los datos en la base de datos.
	EC 2,2: "Crear nuevo metadato con campos vacíos"	El usuario no introduce todos los datos requeridos para crear el nuevo metadato y da clic en el botón Guardar. El sistema muestra un formulario de aviso donde especifica que el formulario no es válido y marca en rojo los campos dejados en blanco y que son obligatorios.

	EC 2,3: “Cancelar crear nuevo metadato”.	El usuario da clic en el botón cerrar de la ventana Propiedades de la sección METADATA. El sistema cierra la ventana y retorna al usuario a la ventana Propiedades de la sección METADATA.
SC 3: “Eliminar nuevo metadato”	EC 3,1: “Eliminar metadato con éxito”	El usuario selecciona un metadato y da clic en la opción Eliminar de la ventana Propiedades de la sección METADATA. El sistema muestra un cartel de confirmación para que el usuario especifique si en realidad desea eliminar el metadato. El usuario da clic en el botón Si. El sistema elimina el metadato seleccionado y muestra la misma ventana con los datos actualizados.
	EC 3,2: “Eliminar metadato incorrectamente”.	El usuario no selecciona un metadato de la lista y da clic en la opción Eliminar. El sistema muestra un mensaje donde le especifica que debe elegir al menos un metadato.

CU Gestionar Objetos OUTPUTFORMAT.

Descripción general.

El caso de uso inicia cuando el usuario selecciona la opción “Navegación” y dentro de la misma selecciona la opción “OUTPUTFORMAT del Mapfile” y luego procede a “Eliminar”, “Modificar”, “Adicionar” o “Reordenar” las OUTPUTFORMAT del Mapfile y termina cuando el usuario finaliza la operación guardando los cambios.

Condiciones de ejecución.

El usuario debe estar autenticado y poseer los privilegios necesarios para acceder a esta opción.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: “Adicionar un nuevo OUTPUTFORMAT.”	EC 1.1: “Adicionar un nuevo OUTPUTFORMAT con éxito”.	El usuario desea adicionar un nuevo OUTPUTFORMAT. El sistema muestra una ventana Propiedades de la sección OUTPUTFORMAT. El usuario introduce los datos correspondientes y guarda los cambios. El sistema crea el OUTPUTFORMAT guardando los cambios en la base de datos y los actualiza.
	EC 1.2: “Adicionar un nuevo OUTPUTFORMAT incorrectamente”.	El usuario no llena todos los campos del outputformat. El sistema muestra en rojo los campos cuyos datos no pueden faltar. El usuario da clic en cerrar. El sistema cierra la aplicación.
SC 2: “Reordenar outputformat”	EC 2.1: “Reordenar outputformat”	El usuario cambia el orden de las outputformat dando clic en uno de los campos de información que aparecen en la interfaz OUTPUTFORMAT del mapfile y selecciona si van a ser ordenados ascendente o descendentemente. El sistema muestra las OUTPUTFORMAT ordenadas en función de los cambios seleccionados por el usuario. El usuario selecciona el botón “Reordenar” para cambiar el orden de los OUTPUTFORMAT. El sistema confirma la reordenación y actualiza la base de datos.
SC 3: “Eliminar	EC 3.1: “Eliminar	El usuario desea eliminar una

outputformat”	outputformat correctamente”	outputformat. El sistema muestra todas las outputformat que se encuentran en el sistema. El usuario elige la outputformat que desea eliminar y guarda los cambios. El sistema elimina el outputformat de la base de datos y los actualiza.
	EC 3.2: “Eliminar outputformat incorrectamente”	El usuario no señala algunas de las outputformat mostradas por el sistema. El sistema muestra una ventana de error, informándole al usuario que debe escoger una de las outputformat.
SC 4: “Modificar outputformat”	EC 4.1: “Modificar outputformat con éxito”	El usuario desea modificar el outputformat. El sistema le muestra una carpeta “OUTPUTFORMAT” con todas las outputformat existentes. El usuario selecciona la outputformat que desea modificar. El sistema muestra una ventana “Propiedades del outputformat” con todos los datos posibles a modificar. El usuario realiza las modificaciones correspondientes y da clic en el botón Guardar. El sistema guarda los datos con las modificaciones realizadas por el usuario y lo guarda en la base de datos.
	EC 4.2: “Modificar outputformat incorrectamente.”	El usuario no llena correctamente todos los datos. El sistema muestra en rojo los datos que debe completar.
	EC 4.3: “Vista previa”	El usuario da clic en el botón “Vista previa”. El sistema muestra una ventana “Vista previa de exportación” la que le

		permite al usuario visualizar los datos modificados antes de ser exportado.

CU Gestionar Objetos SYMBOL.

Descripción general.

El caso de uso inicia cuando el usuario selecciona la opción “Navegación” y dentro de la misma selecciona la opción “SYMBOLS” y luego procede a “Eliminar”, “Adicionar”, “Modificar” “Editar” o “Mostrar la vista previa” de los objetos SYMBOL del Mapfile y termina cuando el usuario finaliza la operación guardando los cambios.

Condiciones de ejecución.

El usuario debe estar autenticado y poseer los privilegios necesarios para acceder a esta opción.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: “Adicionar objeto symbol”	EC 1.1: “Adicionar objeto symbol con éxito”	El usuario selecciona la opción adicionar. El sistema muestra una ventana con las propiedades del objeto symbol. El usuario introduce los datos correspondientes y guarda los cambios. El sistema crea el objeto symbol guardando los datos en la base de datos y los actualiza.
	EC 1.2: “Adicionar objeto symbol incorrectamente”	El usuario no crea el symbol con todos los campos válidos. El sistema muestra un aviso de que el formulario no es válido y selecciona en rojo los campos cuyos datos no pueden faltar.
SC 2: “Modificar symbol”	EC 2.1: “Modificar objeto symbol con	El usuario desea modificar el objeto symbol. El sistema muestra una carpeta symbol con todos los objetos symbol existentes. El usuario elige el symbol que desea modificar. El sistema muestra

	éxito”	una ventana “Propiedades de la sección symbol” con todos los datos que se puedan modificar. El usuario hace las modificaciones deseadas y da clic en el botón guardar. El sistema guarda los cambios realizados por el usuario en la base de datos.
	EC 2.2: “Modificar objeto symbol incorrectamente”	El usuario no entra todos los datos correctamente del symbol. El sistema muestra una ventana de error marcando en rojo el formulario inválido.
	EC 2.3: “Vista previa”	El usuario da clic en el botón “vista previa”. El sistema muestra una ventana “vista previa de exportación” con todos los datos modificados por el usuario.
SC 3: “Eliminar objeto symbol”	EC 3.1: “Eliminar objeto symbol con éxito.”	El usuario desea eliminar un objeto symbol. El sistema muestra todos los symbol en un ListView que se encuentran en el sistema. El usuario elige los symbol a eliminar y guarda los cambios. El sistema elimina los symbol de la base de datos y luego actualiza la base de datos.
	EC 3.2: “Eliminar objeto symbol incorrectamente. ”	El usuario no selecciona al menos un objeto symbol para eliminar. El sistema muestra un mensaje informando que se debe elegir al menos un objeto symbol.

CU Gestionar Objetos LAYER.

Descripción general.

El caso de uso inicia cuando el usuario selecciona la opción de “LAYER”, con el objetivo de adicionar, agrupar, eliminar, reordenar y mostrar la vista previa de un objeto LAYER, y termina la operación, cuando se elige la opción correspondiente.

Condiciones de ejecución.

El usuario debe estar autenticado y poseer los privilegios necesarios para acceder a esta opción.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: “Adicionar objeto Layer”	EC 1.1: “Adicionar objeto layer con éxito”	El usuario selecciona la opción “Adicionar”. El sistema muestra una ventana “Propiedades de la sección layer”. El usuario introduce los datos y selecciona la opción “Guardar”. El sistema guarda el nuevo objeto layer y actualiza los datos.
	EC 1.2: “Adicionar objeto layer incorrectamente.”	El usuario introduce los datos de manera incorrecta y selecciona la opción guardar. El sistema muestra un aviso de que el formulario no es válido y marca en rojo los campos cuyos datos no pueden faltar. El usuario da clic en el botón cerrar. El sistema cierra la ventana “Propiedades de la sección LAYER”.
	EC 1.3: “Cancelar Adicionar LAYER”	El usuario da clic en el botón cerrar. El sistema cierra la ventana “Propiedades de la sección LAYER”.
SC 2: “Agrupar objeto layer”	EC 2.1: “Agrupar objeto layer con éxito”.	El sistema muestra un ListView con todos los Layers existentes. El usuario selecciona el layer que desea agrupar y luego da clic en la opción “Agrupar”. El sistema muestra una ventana “Agrupar

		Layers". El usuario selecciona el grupo en el que desea agrupar el layer y selecciona el botón "Guardar". El sistema agrupa los datos del layer y actualiza los datos.
	EC 2.2: "Agrupar objeto layer sin haberlo seleccionado"	El usuario no selecciona el layer o los Layers que desea agrupar y da clic en la opción Agrupar. El sistema muestra un aviso de que debe seleccionar al menos una capa para poder agruparla.
	EC 2.3: "Agrupar objeto layer llenando el formulario incorrectamente"	El usuario llena los campos incorrectamente. El sistema muestra un mensaje de error especificando que ha ocurrido un error agrupando las layers indicadas.
	EC 2.4: "Crear nueva agrupación con éxito".	El usuario selecciona la opción "Nueva agrupación" de la ventana Agrupar LAYERS y llena los datos correspondientes y da clic en el botón Guardar. El sistema guarda los datos introducidos por el usuario.
	EC 2.5: "Crear nueva agrupación incorrectamente".	El usuario no llena los campos requeridos y da clic en el botón Guardar. El sistema muestra un mensaje de aviso donde especifica que debe llenar todos los campos.
SC 3: "Eliminar objeto layer"	EC 3.1: "Eliminar objeto layer con éxito"	El usuario elige la opción "Eliminar". El sistema muestra todos los Layers que se encuentran en el sistema. El usuario elige el layer que desea eliminar y da clic en la opción Eliminar. El sistema muestra un

		mensaje de conformación para que el usuario confirme que desea eliminar la capa. El usuario confirma. El sistema elimina el layer de la base de datos y actualiza los datos.
	EC 3.2: "Eliminar objeto layer sin haberlo seleccionado primero."	El usuario no selecciona una capa para eliminar. El sistema muestra un mensaje informando que debe seleccionar al menos una capa para eliminar.
SC 4: "Reordenar objeto layer"	EC 4.1: "Reordenar objeto layer "	El usuario desea reordenar Layers. El sistema muestra todos los Layers que existen. El usuario cambia el orden de los LAYER dando clic en uno de los campos de información que aparecen en la interfaz "Capas de un Mapfile" y selecciona si van a ser ordenados ascendente o descendentemente los objetos LAYER en función de ese campo. El sistema muestra los objetos layer ordenados en función de los cambios seleccionados por el usuario. El usuario da clic en el botón "Reordenar". El sistema reordena y actualiza la base de datos.
SC 5: "Vista previa"	EC 5.1: "Vista previa"	El usuario selecciona la opción "Vista previa". El sistema muestra una ventana "Catálogo de Mapas", en el cual muestra un mapa con todos los Layers existentes. El usuario da clic en el mapa escogiendo las capas que desea que se visualicen en

		el mapa. El sistema crea la vista previa.
	EC 5.2: “Cerrar”	El usuario escoge la opción cerrar. El sistema cierra la ventana “Catálogo de mapas”.

CU Crear Perfiles.

Descripción general.

El caso de uso inicia cuando el usuario selecciona la opción Crear Perfil, y termina cuando el usuario finaliza la operación guardando los cambios.

Condiciones de ejecución.

El usuario debe estar autenticado y ser administrador del sistema.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: “Crear perfil”	EC 1.1: “Crear perfil con éxito”	El administrador desea crear un perfil. El sistema muestra los datos a ser llenados por el administrador. El administrador elige el usuario al que le desea crear el perfil y elige el mapa sobre el que va a darle acceso, escogiendo las capas a las que el usuario va a tener acceso y da clic en el botón actualizar. El sistema actualiza los datos del usuario con las capas escogidas.
	EC 1.2: “Crear perfil incorrectamente”	El administrador desea crear un perfil. El sistema muestra los datos a ser llenados por el administrador. El administrador elige el usuario al que le desea crear el perfil y elige el mapa sobre el que va a darle acceso, sin escoger las capas a las que el usuario va a

		tener acceso y da clic en el botón actualizar. El sistema muestra un mensaje de error indicándole al administrador que debe elegir al menos una capa.
SC 2: "Eliminar acceso a datos"	EC 2.1: "Eliminar acceso a datos con éxito"	El administrador desea eliminar mapas del usuario, eligiendo el usuario al que le va a quitar acceso sobre los mapas. El sistema muestra un ListView con los mapas a los cuales el usuario tiene acceso. El administrado elige el o los mapas al que desea denegarle el acceso al usuario dando clic en el botón Eliminar. El sistema muestra una ventana eliminar preguntándole al administrador si desea eliminar el o los Mapfile seleccionados. El administrador da clic en el botón Si. El sistema elimina el acceso al Mapfile y actualiza la base de datos.
	EC 2.2: "Eliminar acceso a datos incorrectamente"	El administrador no selecciona el mapa que desea eliminar. El sistema muestra un mensaje de aviso donde especifica que debe seleccionar al menos un mapa para eliminar. El administrador da clic en el botón cerrar. El sistema cierra la ventana.

CU Configurar Perfil de Usuarios.

Descripción general.

El caso de uso se inicia cuando el administrador selecciona la opción Configurar Perfil, y termina cuando el administrador finaliza la operación guardando los cambios.

Condiciones de ejecución.

El usuario debe estar autenticado y debe ser administrador.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: "Configurar capas activas del mapfile"	EC 1.1: "Configurar capas activas del Mapfile con éxito"	El administrador elige las capas activas del Mapfile. El sistema muestra un ComboBox con todos los Mapfile existentes. El administrador elige el mapa al que desea configurarle las capas activas. El sistema muestra las capas del mapa seleccionado por el administrador a las que tiene acceso. El administrador elige las capas sobre las que desea trabajar y da clic en la opción actualizar. El sistema guarda los cambios en la base de datos.
	EC 1.2: "Configurar capas activas del Mapfile sin seleccionar un Mapfile"	El administrador no elige el mapa al que va a configurarle las capas activas y da clic en el botón Actualizar. El sistema muestra un mensaje informando que se debe elegir un mapa.
	EC 1.3: "Configurar capas activas del Mapfile sin seleccionar capas"	El administrador no elige las capas activas que tendrá el Mapfile. El sistema muestra un mensaje de aviso donde especifica que debe elegir al menos una capa activa.
SC 2: "Editar	EC 2.1: "Editar	El administrador desea cambiar su

datos.”	datos correctamente.”	contraseña. El sistema muestra un TextBox para entrar los datos contraseña actual, nueva contraseña y repetir contraseña. El administrador entra todos los datos y da clic en guardar. El sistema guarda los cambios de la contraseña.
	EC 2.2: “Editar datos incorrectamente”	El administrador deja de llenar algún campo. El sistema marca en rojo los campos obligatorios y muestra un mensaje de aviso especificándole que el formulario no es válido.

CU Crear Perfil de Funcionalidades.

Descripción general.

El caso de uso inicia cuando el usuario selecciona la opción Perfil Funcionalidades, y termina cuando el usuario finaliza la operación guardando los cambios.

Condiciones de Ejecución.

El usuario debe estar autenticado y debe ser administrador del sistema.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: “Modificar perfil de funcionalidades”	EC 1.1: “Modificar perfil de funcionalidades correctamente”	El administrador desea modificar el perfil de las distintas funcionalidades existentes. El sistema muestra un ComboBox Roles con los datos de los diferentes roles que existen y otro ComboBox con todos los módulos. El administrador elige el rol al que le desea crear el perfil y elige el módulo sobre el que

		va a darle acceso. El sistema muestra un Findlabel “Funcionalidades con acceso” con las funcionalidades a las que tiene acceso el rol y módulo escogido por el administrador. El administrador elige las funcionalidades sobre las que va a darle acceso al rol y da clic en el botón “Actualizar”. El sistema guarda la configuración y actualiza los datos.
	EC 1.2: “Modificar perfil de funcionalidades incorrectamente”	El administrador no elige un rol o módulos o ninguno de los dos. El sistema muestra un mensaje indicando que debe elegir un rol y un módulo.
	EC 1.3: “Modificar perfil de funcionalidades sin elegir las capas a las que tendrá acceso”	El administrador no elige las capas del módulo a las que tendrá acceso. El sistema muestra un mensaje en el que especifica que debe elegir al menos una opción.
SC 2: “Eliminar acceso a funcionalidades”	EC 2.1: “Eliminar acceso a funcionalidades con éxito”	El administrador desea eliminar permisos sobre las funcionalidades. El sistema muestra un Findlabel “módulos de acceso” con un ListView los módulos a los cuales el rol tiene permiso. El administrador elige el rol al que va a quitarle acceso sobre los módulos. El sistema muestra un ListView con los módulos a los que el rol seleccionado tiene acceso. El administrador elige el o los módulos al que desea

		denegarle el acceso al rol y da clic en la opción “Eliminar”. El sistema elimina el módulo y guarda los cambios del perfil en la base de datos y actualiza los datos.
	EC 2.2: “Eliminar acceso a funcionalidades incorrectamente”	El administrador no elige el módulo que desea eliminar. El sistema muestra una ventana con un aviso que especifica que debe elegir al menos un módulo.

CU Gestionar Roles.

Descripción general.

El caso de uso se inicia cuando el usuario selecciona la opción Gestionar Roles, y termina cuando el usuario finaliza las operaciones guardando los cambios.

Condiciones de ejecución.

El usuario debe estar autenticado y ser administrador del sistema.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: “Crear rol”	EC 1.1: “Crear rol con éxito”	El administrador desea crear un rol. El sistema muestra un Findlabel Nuevo Rol con un TextBox Descripción y un RadioButton Acceso. El administrador indica la descripción del rol que desea crear y su nivel de acceso y da clic en el botón “Guardar”. El sistema guarda el perfil en la base de datos.
	EC 1.2: “Crear rol ya	El administrador entra un rol ya existente. El sistema muestra una

	existente”	ventana de error indicándole que ya existe ese rol.
	EC 1.3: “Crear rol con campos vacíos”	El administrador no entra los datos correspondientes. El sistema marca en rojo el campo requerido y un mensaje de error que especifica que el formulario no es válido.
SC 2: “Eliminar rol”	EC 2.1: “Eliminar rol con éxito”	El administrador desea eliminar un rol existente. El sistema muestra todos los roles registrados. El administrador selecciona el rol que desea eliminar y da clic en el botón “Eliminar”. El sistema muestra un mensaje para que el administrador confirme que desea eliminar el rol y cuando el usuario confirma el sistema elimina el rol de la base de datos.
	EC 2.2: “Eliminar rol sin seleccionar rol”	El administrador no selecciona el rol que desea eliminar y da clic en el botón “eliminar.” El sistema muestra un mensaje de error donde especifica que se debe seleccionar al menos un rol.
SC 3: “Editar rol”	EC 3.1: “Editar rol con éxito”	El administrador desea editar un rol. El sistema muestra un Find Label “Editar roles” con un ComboBox “Roles”, un TextBox “Renombrar” y un RadioButton” Acceso”. El administrador elige el

		rol que desea editar y hace los cambios correspondientes. El sistema efectúa los cambios correspondientes.
	EC 3.2: "Editar rol incorrectamente"	El administrador no elige el rol que desea editar ni lo renombra. El sistema marca en rojo los campos que son requeridos y muestra una ventana de error donde indica que el formulario no es válido.

Casos de prueba de Carga y Estrés.

Caso de prueba "Gestionar Objetos Layer".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	Comentario
Adicionar objeto Layer	
Agrupar objeto Layer	
Crear nueva agrupación	

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	70

Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Caso de prueba "Gestionar Objetos OUTPUTFORMAT del Mapfile".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	**Comentario**
Adicionar nuevo OUTPUTFORMAT.	
Modificar OUTPUTFORMAT	

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	80
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones:	1

[Contador del bucle]	

Caso de prueba "Gestionar Objetos SYMBOL".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	**Comentario**
Adicionar objeto SYMBOL	
Modificar objeto SYMBOL	

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	70
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Caso de prueba "Gestionar Roles".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	**Comentario**
Crear rol	
Editar rol	
Eliminar rol	

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	100
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Caso de prueba "Gestionar Usuarios del Sistema".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	Comentario
Adicionar nuevo usuario	

Editar rol	

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	70
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Caso de prueba "Modificar Objeto Web".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	**Comentario**
Modificar objeto web	

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	100

Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Caso de prueba "Modificar Objeto".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	**Comentario**
Modificar objeto MAP.	
Modificar objeto LEGEND.	
Editar LABEL.	
Modificar objeto SCALEBAR.	
Modificar objeto REFERENCE.	
Modificar objeto QUERYMAP.	

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	100

Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Anexo 3 No conformidades detectadas en las Pruebas de Caja Negra.

No conformidades detectadas en las Pruebas de Caja Negra.

No.	Criterio de evaluación	Elemento	Recomendaciones	IMP	ED	EST
1	Adherencia a producto	Aplicación LiberMaps. CU Configuración de Datos Globales. El sistema no tiene validado la entrada de datos al formulario “Configuración Global” y permite guardar los datos introducidos.	Se le recomienda que realice la validación de los datos de entrada del formulario Configuración Global.	A	O	PD
2	Adherencia a producto	Aplicación LiberMaps. CU Gestionar Mapfile. El escenario “Importar Mapfile” no existe en la aplicación por lo que el sistema no permite importar un Mapfile.	Se le recomienda que incluya esta funcionalidad en el sistema.	A	O	PD
3	Adherencia a producto.	Aplicación LiberMaps. CU Gestionar Usuarios del	Se le recomienda corregir la interfaz.	M	O	PD

No.	Criterio de evaluación	Elemento	Recomendaciones	IMP	ED	EST
		Sistema. En el módulo perfiles, el formulario “Gestión de usuarios del sistema”, existe un error de interfaz en la opción “Editar roles del sistema” dice: “Editar role del usuario”. Debería decir “Editar roles del usuario”.				
4	Adherencia a producto.	Aplicación LiberMaps. CU Modificar Objeto Web. El sistema permite introducir datos inválidos para modificar objetos web.	Se le recomienda que valide la entrada de datos al formulario “Propiedades de la sección WEB”	A	O	PD
5	Adherencia a producto.	Aplicación LiberMaps. CU Exportar Mapfile. El sistema no tiene validado que se pueda exportar el mismo Mapfile más de una vez.	Se le recomienda que valide que no se puede exportar un mapfile que ya ha sido exportado.	M	O	PD
6	Adherencia a producto.	Aplicación LiberMaps. CU Gestionar METADATA del Objeto Web. El formulario “Propiedades	Se le recomienda que valide la entrada de datos al formulario “Propiedades de la	A	O	PD

No.	Criterio de evaluación	Elemento	Recomendaciones	IMP	ED	EST
		de la sección METADATA" no tiene validación de datos incorrectos. El formulario "Propiedades de la sección METADATA" que sale cuando se selecciona la opción nuevo metadato no tiene validación de datos incorrectos.	sección METADATA" y al formulario "Propiedades de la sección METADATA" que sale luego de que se selecciona la opción nuevo metadato.			
7	Adherencia a producto.	Aplicación LiberMaps Formulario Propiedades de la sección OUTPUFORMAT. Módulo Navegación. Opción OUTPUTFORMA Clic en la funcionalidad Adicionar. No existe validación de entrada de datos incorrectos en el formulario "Propiedades de la sección OUTPUTFORMAT"	Se le recomienda que realice validación de entrada de datos.	M	O	PD
8	Adherencia a producto.	Aplicación LiberMaps. CU Gestionar Roles. Cuando el sistema	Se le recomienda que corrija la interfaz.	M	O	PD

No.	Criterio de evaluación	Elemento	Recomendaciones	IMP	ED	EST
		muestra un mensaje porque el rol ya existe, la interfaz tiene un error de escritura, dice “El role especificado ya existe”. Debería decir: “El rol especificado ya existe”.				
9	Adherencia a producto.	Aplicación LiberMaps. CU Gestionar Roles. La interfaz de aviso al usuario cuando quiere eliminar un rol sin seleccionar uno, contiene un error de escritura. Dice “Debe seleccionar al menos un role” Debería decir “Debe seleccionar al menos un rol”.	Se le recomienda que corrija la interfaz.	M	O	PD
10	Adherencia a producto.	Aplicación LiberMaps. CU Modificar Objeto. No existe una validación de los datos de entrada al formulario. Formulario “Propiedades de la sección MAP”. Formulario “Propiedades de la sección LEGEND”.	Se le recomienda que valide la entrada de datos al formulario.	M	O	PD

No.	Criterio de evaluación	Elemento	Recomendaciones	IMP	ED	EST
		Formulario “Propiedades de la sección LABEL”. Formulario “Propiedades de la sección SCALEBAR” Formulario “Propiedades de la sección REFERENCE”. Formulario “Propiedades de la sección QUERYMAP”				
11	Adherencia a producto.	Aplicación LiberMaps. CU Gestionar Metadatos de los Objetos Layer. El formulario Nuevo METADATA permite crear el mismo METADATO una y otra vez.	Se le recomienda que valide que no se pueda crear el mismo METADATO una y otra vez.	M	O	PD
12	Adherencia a producto.	Aplicación LiberMaps. CU Gestionar Objeto Style de los Objetos Layer. En el escenario Eliminar objeto style con éxito, dice que el usuario debe seleccionar un estilo para	Se le recomienda que valide esta funcionalidad.	A	O	PD

No.	Criterio de evaluación	Elemento	Recomendaciones	IMP	ED	EST
		eliminar y que si no lo selecciona debe mostrar un error. El sistema no muestra un mensaje de error indicando que debe seleccionar al menos un estilo.				
13	Adherencia a producto.	Aplicación LiberMaps. Formulario "Nueva agrupación" Formulario "Editar agrupación". Estos formularios no tienen validación de datos de entrada, es decir que se puede entrar lo mismo números que letras.	Se le recomienda que valide la entrada de datos a los formularios.	M	O	PD
14	Adherencia a producto.	Aplicación LiberMaps. Cu Modificar Objetos Layer. Formulario "Propiedades de la sección METADATA". En el formulario "Propiedades de la	Se le recomienda que haga coincidir la interfaz con lo que realmente se está tratando de eliminar "un metadato".	M	O	PD

No.	Criterio de evaluación	Elemento	Recomendaciones	IMP	ED	EST
		sección METADATA", cuando se desea eliminar un nuevo Metadata creado, si no se selecciona un metadato el sistema muestra una ventana de aviso, esa ventana de aviso dice "debe seleccionar al menos un mapa", pero debería decir "Debe elegir al menos un metadato".				
15	Adherencia a producto.	Aplicación LiberMaps. CU Modificar Objetos Layer. En el formulario "Propiedades de la sección JOIN", "Propiedades de la sección GRID", "Propiedades de la sección METADATA" y en el formulario "Propiedades de la sección FEATURE" no existe una validación de datos incorrectos.	Se le recomienda que realice una validación de datos incorrectos.	M	O	PD

Anexo 4 Casos de uso arquitectónicamente significativos.

Casos de uso arquitectónicamente significativos.

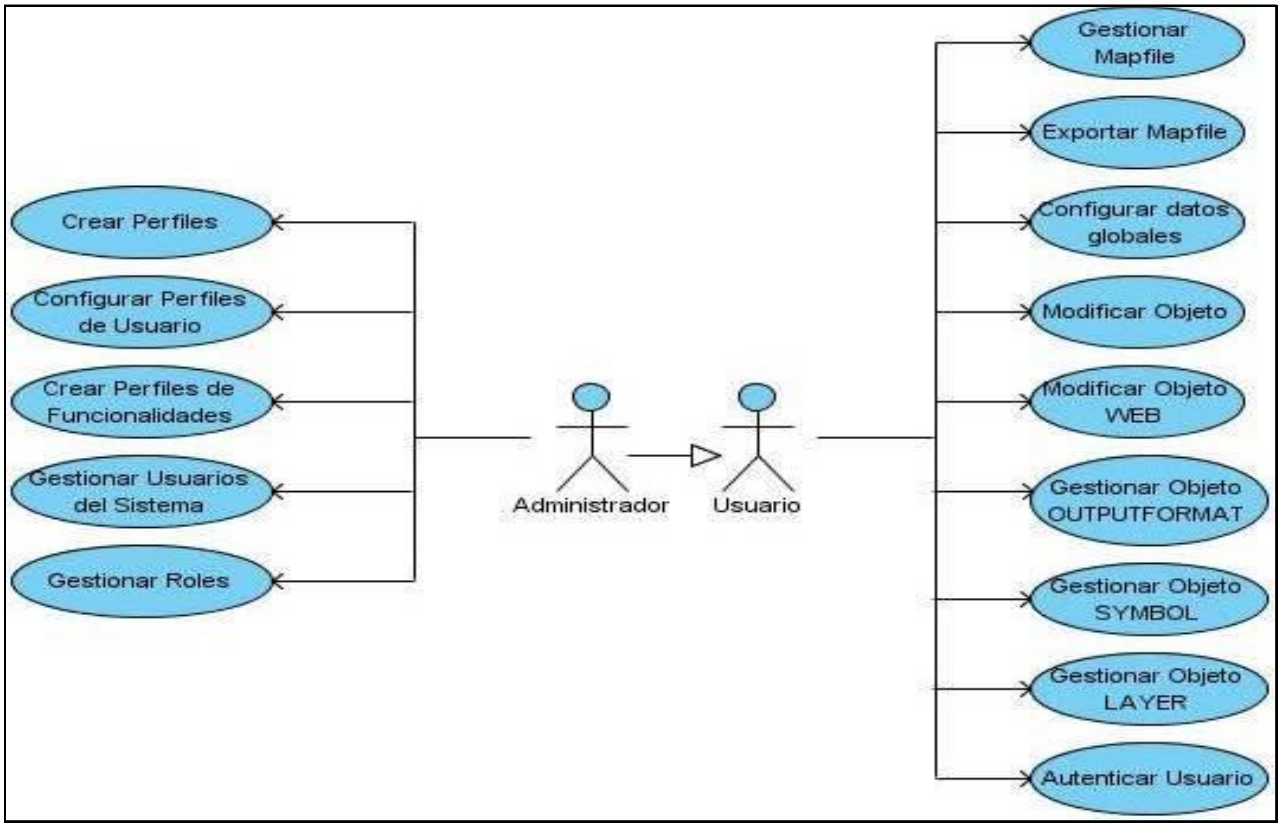


Ilustración 3 Casos de uso arquitectónicamente significativos.

Anexo 5 Resultados de las Pruebas de Carga y Estrés.

Caso de prueba "Autenticar Usuario".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	**Comentario**
Autenticar usuario	Se recogen los datos usuario y contraseña para poder autenticarse y entrar al sistema. El sistema verifica que sean correctos los datos y de ser correctos permite el acceso a la aplicación.

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	50
Tiempo entre conexión y conexión: (período de subida(en segundos))	1,3
Número de iteraciones: [Contador del bucle]	1

Label	# Muestras	Media	Mediana	Línea de 90 %	Min	Max	%Error	Rendimiento	Kb/sec
TOTAL	1250	155	3	83	0	4440	0.0	28,8 sec	879,2

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	100
Tiempo entre conexión y conexión: (período de subida(en segundos))	1,3
Número de iteraciones: [Contador del bucle]	1

Label	# Muestras	Media	Mediana	Línea de 90 %	Min	Max	%Error	Rendimiento	Kb/sec
TOTAL	5000	197	2	43	0	7884	0.0	22.72	692.75

Lista de links con errores:	Comentario
Ninguna	

Caso de prueba “Configuración de Datos Globales”.

Tipo de prueba: Prueba de Carga.

Funcionalidad(es)	Comentario
Configuración de Datos Global	El sistema permite guardar la configuración de los datos que se necesitan para guardar los mapas exportados, la url de la aplicación.

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	50
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Label	# Muestras	Media	Mediana	Línea de 90 %	Min	Max	%Error	Rendimiento	Kb/sec
TOTAL	400	3310	3031	8984	0	14075	25,00	13.33	9.03

Variables	Valores

Número de usuarios concurrentes: (número de hilos)	100
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Label	**# Muestras**	**Media**	**Mediana**	**Línea de 90 %**	**Min**	**Max**	**%Error**	**Rendimiento**	**Kb/sec**
TOTAL	800	6610	4622	19998	0	27402	25.00	13.00	8.79

Lista de links con errores:	**Comentario**
libermaps/web/index.php/main/appFunctionalities	
libermaps/web/index.php/main/appFunctionalities	
libermaps/web/index.php/main/appFunctionalities	
libermaps/web/index.php/useful/loadTreeConfig	

Caso de pruebas “Configurar Perfil de Usuarios”.

Tipo de prueba: Prueba de Carga.

Funcionalidad(es)	**Comentario**
Configurar capas activas del Mapfile	El sistema permite configurar las capas activas que tendrá el Mapfile.
Editar datos	Permite editar los datos del usuario, tales como los mapas a los que tendrá acceso.

Variables	**Valores**

Número de usuarios concurrentes: (número de hilos)	80
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Label	**# Muestras**	**Media**	**Mediana**	**Línea de 90 %**	**Min**	**Max**	**%Error**	**Rendimiento**	**Kb/sec**
TOTAL	2240	5187	2674	15547	0	26057	25.00	14.59	4.76

Lista de links con errores:	**Comentario**
/libermaps/web/index.php/main/appFunctionalities	
/libermaps/web/index.php/main/appFunctionalities	
/libermaps/web/index.php/useful/loadTreePerfil	
/libermaps/web/index.php/perfil/newPerfil	
/libermaps/web/index.php/useful/loadTree	
/libermaps/web/index.php/perfil/deletePerfil	

Caso de prueba “Crear Perfiles de Funcionalidades”.

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	**Comentario**
Modificar perfil de funcionalidades	
Eliminar acceso a funcionalidades	

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	70
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Label	# Muestras	Media	Mediana	Línea de 90 %	Min	Max	%Error	Rendimiento	Kb/sec
TOTAL	1750	6408	4737	13226	0	16362	16.00	10.62	8.07

Lista de links con errores:	Comentario
/libermaps/web/index.php/perfil/newPerfilConfig	
/libermaps/web/index.php/useful/loadTreePerfil	
/libermaps/web/index.php/useful/loadTreeConfig	
/libermaps/web/index.php/main/appFunctionalities	

Caso de prueba “Crear Perfil de Usuarios”.

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	Comentario
Crear Perfil	
Eliminar acceso a datos	

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	70

Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Label	**# Muestras**	**Media**	**Mediana**	**Línea de 90 %**	**Min**	**Max**	**%Error**	**Rendimiento**	**Kb/sec**
TOTAL	770	3713	3599	10576	0	17727	27,27	16.47	20.21

Lista de links con errores:	**Comentario**
/libermaps/web/index.php/useful/loadTreePerfil	
/libermaps/web/index.php/useful/loadTreeConfig	
/libermaps/web/index.php/main/appFunctionalities	

Caso de prueba “Exportar Mapfile”.

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	**Comentario**
Exportar Mapfile	

Variables	**Valores**
Número de usuarios concurrentes: (número de hilos)	70
Tiempo entre conexión y conexión: (período de subida(en segundos))	2
Número de iteraciones: [Contador del bucle]	1

Label	# Muestras	Media	Mediana	Línea de 90 %	Min	Max	%Error	Rendimiento	Kb/sec
TOTAL	3420	3461	2981	9353	0	17983	44,44	3,1	8.86

Lista de links con errores:	Comentario
/libermaps/web/index.php/global/export	
/libermaps/web/index.php/perfil/getMapsByUser	
/libermaps/web/index.php/useful/loadTreeConfig	
/libermaps/web/index.php/main/appFunctionalities	

Caso de prueba "Gestionar Mapfile".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	Comentario
Adicionar Mapfile	
Eliminar Mapfile	

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	50
Tiempo entre conexión y conexión: (período de subida(en segundos))	2
Número de iteraciones: [Contador del bucle]	1

Label	# Muestras	Media	Mediana	Línea de 90 %	Min	Max	%Error	Rendimiento	Kb/sec
TOTAL	6125	1866	12	4485	0	27010	22,86	5,9	4,1

Lista de links con errores:	Comentario
/libermaps/web/index.php/global/deleteMapfile	
/libermaps/web/index.php/global/loadMapsList	
/libermaps/web/index.php/useful/loadTree	
/libermaps/web/index.php/global/newMapfile	
/libermaps/web/index.php/useful/loadTreeConfig	
/libermaps/web/index.php/main/appFunctionalities	

Caso de prueba "Gestionar METADATA del Objeto Web".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	Comentario
Editar METADATA	
Crear nuevo metadato	
Eliminar metadato	

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	90
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Label	# Muestras	Media	Mediana	Línea de 90 %	Min	Max	%Error	Rendimiento	Kb/sec

TOTAL	9520	5430	1308	7976	0	122001	38,23	11.60	12.88

Lista de links con errores:	**Comentario**
/libermaps/web/index.php/webobject/deleteWebmetadata	
/libermaps/web/index.php/webobject/addWebmetadataNew	
/libermaps/web/index.php/webobject/gridWebmetadataNew	
/libermaps/web/index.php/webobject/editWebmetadata	
/libermaps/web/index.php/webobject/loadWebMetadata	
/libermaps/web/index.php/webobject/loadWebobject	
/libermaps/web/index.php/useful/updateSession	
/libermaps/web/index.php/useful/loadTree	
/libermaps/web/index.php/useful/loadTreeConfig	
/libermaps/web/index.php/main/appFunctionalities	

Caso de prueba “Gestionar Objetos Layer”.

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	**Comentario**
Adicionar objeto Layer	
Agrupar objeto Layer	
Crear nueva agrupación	

Variables	**Valores**
Número de usuarios concurrentes: (número de hilos)	70

Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Label	**# Muestras**	**Media**	**Mediana**	**Línea de 90 %**	**Min**	**Max**	**%Error**	**Rendimiento**	**Kb/sec**
TOTAL	2240	4151	3764	11956	0	18750	43,75	16.14	3.31

Lista de links con errores:	**Comentario**
/libermaps/web/index.php/layers/AgruparLayers	
/libermaps/web/index.php/layers/addNewLayer	
/libermaps/web/index.php/layers/viewLayers	
/libermaps/web/index.php/useful/updateSession	
/libermaps/web/index.php/useful/loadTree	
/libermaps/web/index.php/useful/loadTreeConfig	
/libermaps/web/index.php/main/appFunctionalities	

Caso de prueba “Gestionar objetos OUTPUTFORMAT del Mapfile”.

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	**Comentario**
Adicionar nuevo OUTPUTFORMAT.	
Modificar OUTPUTFORMAT	

Variables	**Valores**

Número de usuarios concurrentes: (número de hilos)	80
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Label	**# Muestras**	**Media**	**Mediana**	**Línea de 90 %**	**Min**	**Max**	**%Error**	**Rendimiento**	**Kb/sec**
TOTAL	7520	2117	14	6859	0	31848	13,84	33,6	291,6

Lista de links con errores:	**Comentario**
/libermaps/web/index.php/outputformats/editOutputformat	
/libermaps/web/index.php/outputformats/loadOutputformat	
/libermaps/web/index.php/outputformats/addOutputformat	
/libermaps/web/index.php/useful/loadCombo	
/libermaps/web/index.php/outputformats/viewOutputformats	
/libermaps/web/index.php/useful/updateSession	
/libermaps/web/index.php/useful/loadTree	

Caso de prueba “Gestionar Objetos SYMBOL”.

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	**Comentario**
Adicionar objeto SYMBOL	
Modificar objeto SYMBOL	

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	70
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Label	# Muestras	Media	Mediana	Línea de 90 %	Min	Max	%Error	Rendimiento	Kb/sec
TOTAL	1960	6195	4654	12400	0	20151	39,29	11,1	5,5

Lista de links con errores:	Comentario
/libermaps/web/index.php/symbols/editSymbol	
/libermaps/web/index.php/symbols/loadSymbol	
/libermaps/web/index.php/symbols/addSymbol	
/libermaps/web/index.php/symbols/viewSymbols	
/libermaps/web/index.php/useful/updateSession	
/libermaps/web/index.php/useful/loadTree	
/libermaps/web/index.php/useful/loadTreeConfig	
/libermaps/web/index.php/main/appFunctionalities	

Caso de prueba “Gestionar Roles”.

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	Comentario
Crear rol	

Editar rol	
Eliminar rol	

Variables	**Valores**
Número de usuarios concurrentes: (número de hilos)	100
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Label	**# Muestras**	**Media**	**Mediana**	**Línea de 90 %**	**Min**	**Max**	**%Error**	**Rendimiento**	**Kb/sec**
TOTAL	2300	11446	4476	30772	0	116547	26,26	8,2	2,8

Lista de links con errores:	**Comentario**
/libermaps/web/index.php/perfil/deleteRoles	
/libermaps/web/index.php/perfil/editRole	
/libermaps/web/index.php/perfil/getEditRolesList	
/libermaps/web/index.php/perfil/addRole	
/libermaps/web/index.php/useful/loadTreePerfil	
/libermaps/web/index.php/useful/loadTreeConfig	
/libermaps/web/index.php/main/appFunctionalities	

Caso de prueba “Gestionar Usuarios del Sistema”.

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	Comentario
Adicionar nuevo usuario	
Editar rol	

Variables	Valores
Número de usuarios concurrentes: (número de hilos)	70
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Label	# Muestras	Media	Mediana	Línea de 90 %	Min	Max	%Error	Rendimiento	Kb/sec
TOTAL	4830	5648	4091	12195	0	31287	30,43	1,2	0,3

Lista de links con errores:	Comentario
/libermaps/web/index.php/perfil/editUserRole	
/libermaps/web/index.php/perfil/getUserRole	
/libermaps/web/index.php/perfil/addUser	
/libermaps/web/index.php/useful/loadTreePerfil	
/libermaps/web/index.php/useful/loadTreeConfig	
/libermaps/web/index.php/main/appFunctionalities	

Caso de prueba “Modificar Objeto Web”.

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	Comentario

Modificar objeto web	

Variables	**Valores**
Número de usuarios concurrentes: (número de hilos)	100
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Label	**# Muestras**	**Media**	**Mediana**	**Línea de 90 %**	**Min**	**Max**	**%Error**	**Rendimiento**	**Kb/sec**
TOTAL	1400	16762	4945	82740	0	133759	50,00	5,7	2,0

Lista de links con errores:	**Comentario**
/libermaps/web/index.php/webobject/editWebobject	
/libermaps/web/index.php/webobject/loadWebobject	
/libermaps/web/index.php/useful/updateSession	
/libermaps/web/index.php/useful/loadTree	
/libermaps/web/index.php/useful/loadTreeConfig	
/libermaps/web/index.php/main/appFunctionalities	

Caso de prueba “Modificar Objeto”.

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)	**Comentario**

Modificar objeto MAP.	
Modificar objeto LEGEND.	
Editar LABEL.	
Modificar objeto SCALEBAR.	
Modificar objeto REFERENCE.	
Modificar objeto QUERYMAP.	

Variables	**Valores**
Número de usuarios concurrentes: (número de hilos)	100
Tiempo entre conexión y conexión: (período de subida(en segundos))	1
Número de iteraciones: [Contador del bucle]	1

Label	**# Muestras**	**Media**	**Mediana**	**Línea de 90 %**	**Min**	**Max**	**%Error**	**Rendimiento**	**Kb/sec**
TOTAL	5200	6120	4643	17145	0	64139	51,92	15,0	23,7

Lista de links con errores:	**Comentario**
/libermaps/web/index.php/querymap/editQuerymap	
/libermaps/web/index.php/querymap/loadQuerymap	
/libermaps/web/index.php/refmap/editRefmap	
/libermaps/web/index.php/refmap/loadRefmap	
/libermaps/web/index.php/scalebar/editScalebar	
/libermaps/web/index.php/scalebar/loadScalebar	
/libermaps/web/index.php/legend/editLegendLabel	

/libermaps/web/index.php/legend/loadLegendLabel	
/libermaps/web/index.php/legend/editLegend	
/libermaps/web/index.php/legend/loadLegend	
/libermaps/web/index.php/mapobject/editMapobject	
/libermaps/web/index.php/mapobject/loadMapobject	
/libermaps/web/index.php/useful/updateSession	
/safebrowsing/downloads?client=navclient-auto-ffox&appver=3.6.16&pver=2.2&wrkey=AKEgNitKMhowYN9bWT0ts29Aqx-xiWJ0OqG1YwmjrXQjNIwSY4P1xvtVIW9AE-pHq_Fz0VPCKJm5gnvv0To03avxga0AjwT8eQ==	
/libermaps/web/index.php/useful/loadTree	
/libermaps/web/index.php/useful/loadTreeConfig	
/libermaps/web/index.php/main/appFunctionalities	

Anexo 6 Método Empírico "Entrevista".

Guía para la entrevista realizada a la analista del Sistema LiberMaps.

1. ¿En qué etapa de desarrollo se encuentra la aplicación?
2. ¿Existen diseños de pruebas para la aplicación LiberMaps?
3. ¿Por qué no son válidos los diseños de pruebas realizados anteriormente?
4. ¿Cuáles son las características del sistema? ¿Cuántas funcionalidades tiene y cuántas son significativas?

Guía para la entrevista de especialistas de calidad del Centro de Calidad para Soluciones Tecnológicas (CALISOFT)

1. ¿Qué tipo de pruebas se realizan en este centro, dónde las realizan?
2. ¿Qué herramientas utilizan para realizar las pruebas?
3. Según el tipo de aplicación ¿Qué tipo de prueba se le puede aplicar?
4. ¿Cómo se realiza el análisis del rendimiento arrojado por la herramienta JMeter luego de ejecutar un plan de prueba?

5. Existe algún estándar para determinar cuándo una aplicación web tiene un buen rendimiento.