UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Título: Diseño y aplicación de pruebas a la Plataforma Soberana GeneSIG

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

Autora: Ivis Pages Díaz

Tutora: Ing. Liuba María Infante Infante.

La Habana

Junio de 2012.

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 firmamos la presente a los ____ días del mes de _____ del año _____.

Ivis Pages Díaz **Liuba María Infante Infante**

_____ _____

Firma de la Autora Firma de la Tutora

DATOS DEL CONTACTO

Nombre: Liuba María Infante Infante.

Correo electrónico: lminfante@uci.cu

Categoría docente: Instructor Recién Graduado.

Año de graduación: 2010.

Profesión: Ingeniero en Ciencias Informáticas.

Breve descripción: Graduado en la Universidad de las Ciencias Informáticas.

AGRADECIMIENTOS

A mis padres por toda la entrega y confianza que depositaron en mí todos estos años, por todo el sacrificio hecho hasta el día de hoy.

A mis abuelos, gracias a ellos tengo unos padres maravillosos.

A Madelaine, más que una amiga eres la hermana que nunca tuve gracias por estar siempre para mí y por apoyarme cuando más lo he necesitado.

A todos los profesores que me formaron como profesional, en especial a Zayli por brindarme su apoyo como profesora y como amiga.

A mi tutora por toda la ayuda que me ha brindado y porque sé que no ha sido fácil lidiar conmigo todo este tiempo.

A José Armando (mi novio) por su respeto, comprensión y porque sé que a su forma siempre has estado ahí para lo que me ha hecho falta.

A Lisandra por ser mi gran amiga.

A Lisset, Sucel, Gretell por toda la ayuda que me dieron en el desarrollo de mi tesis, sin ustedes no sé qué me hubiera hecho con mi documento.

A todas mis amistades del cuarto piso incluyendo a Julio y a Elier gracias por ser unos buenos amigos aunque a veces tenga deseos de que desaparezcan de delante de mí.

A mis amigos de universidad que nunca olvidaré ya que son una parte muy importante en mi vida, gracias a Rolando, Yoandy, Ernesto, Guillermo, a todos los Canecas sin excluir a ninguno, mis amigos de la facultad 2, facultad 7, mis compañeras de apartamento y cada una de las personas que han compartido conmigo en algún momento de esta universidad.

DEDICATORIA

A mis padres porque son lo más grande que tengo en este mundo, gracias por brindarme su amor y su apoyo incondicional desde el momento en que abrí los ojos hasta hoy. Todos mis logros se los debo a ustedes porque siempre me ha apoyado y han estado ahí cuando más los he necesitado, los amo muchísimo.

Resumen

La posibilidad de que se cometan errores en el ciclo de vida de un software es muy elevada, desde el momento en que se levantan requisitos hasta la implementación. Debido a esto se hace necesario que se verifique la calidad del software una vez que el equipo de desarrollo haya acabado la implementación. En la presente investigación se realizó un estudio de los principales conceptos asociados al dominio del problema, se seleccionaron las pruebas que se ejecutaron para lograr el objetivo propuesto y las herramientas utilizadas. Se confeccionó un procedimiento de pruebas que guió la puesta en práctica de cada una de estas pruebas. Se elaboró un plan de pruebas que incluyó la estrategia a seguir y que logró una mayor organización en las actividades llevadas a cabo durante el proceso de la ejecución de las pruebas. Se verificó la correcta implementación de los requisitos planteados por el cliente encontrando errores significativos que no fueron detectados por los desarrolladores y que además empañan la buena calidad de la aplicación.

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

ÍNDICE DE CONTENIDO

ÍNDICE DE CONTENIDO

Índice de Ilustraciones

Índice de Tablas

ÍNDICE DE TABLAS

Introducción

Con el transcurso de los años el mundo ha sido testigo de cómo han ido surgiendo una serie de adelantos científico técnicos, con el objetivo de desarrollar la economía y la sociedad a nivel mundial. Uno de los más importantes avances tecnológicos es el desarrollo de las Tecnologías de la Información y las Comunicaciones (TICs). Las mismas han llegado a formar parte de la cultura tecnológica que rodea a la población y con la cual hay que convivir diariamente para ampliar las capacidades físicas y mentales.

Las TICs han contribuido a que el mundo se encuentre en una crisis global, debido a que los países del primer mundo desean monopolizar estas tecnologías para sus propios fines y no para lograr un desarrollo sostenible y equitativo en el mundo. Estos países llevan el liderazgo en el avance de las tecnologías y cuentan con muchas empresas de software especializadas en Sistemas de Información Geográfica (SIG).

Los SIG, son una disciplina académica y sus aplicaciones están actualmente expandidas en muchas universidades del mundo. Los jóvenes profesionales cubanos son cada vez más conscientes de las ventajas que proporciona que un sistema almacene y manipule información geográfica, ya que cerca del 80% de la información tratada en empresas e instituciones, está relacionada con localizaciones geográficas o coordenadas espaciales. (Landestoy, 2011)

En Cuba, muy pocas empresas se dedican a la creación de software, lo que hace que la Universidad de las Ciencias Informáticas (UCI) se convierta prácticamente en la principal empresa productora del país. La UCI cuenta con Centros de Desarrollo de Software, distribuidos en sus siete facultades. En el Centro de Geoinformática y Señales Digitales (GEySED) de la facultad 6, se encuentra el Departamento de Geoinformática, donde se desarrolla el proyecto GeneSIG, encargado de la elaboración de una plataforma que constituye la base para el desarrollo de los SIG. Este surge como necesidad de contar con una línea de trabajo enfocada al desarrollo de SIG con el manejo de tecnologías libres.

Actualmente se está desarrollando la versión 1.5 de la plataforma, la cual se encuentra en la fase de pruebas. Durante este proceso se ha podido constatar que en el centro existe un equipo encargado de realizar las pruebas a los productos desarrollados, pero este equipo no tiene la experiencia necesaria para trabajar con las herramientas que se necesitan para llevar a cabo las pruebas al sistema. Teniendo en cuenta que en la versión 1.5 del producto surgieron cambios en el sistema, debido al

desarrollo de nuevas funcionalidades, se hace necesaria la creación de nuevos diseños de casos de pruebas y la ejecución de estos en la fase actual.

Por lo planteado anteriormente se puede precisar que la Plataforma Soberana GeneSIG corre el riesgo de contener errores, tales como: funcionalidades ausentes, validación incorrecta de valores de entrada, errores ortográficos, que el sistema no tenga un rendimiento óptimo relacionado a tiempos de respuesta apropiados, funcionamiento con grandes volúmenes de datos y acceso de personas no autorizadas a la aplicación.

Esas consideraciones permiten reconocer como **problema a resolver:** ¿Cómo asegurar que la Plataforma Soberana GeneSIG sea entregada con la calidad requerida a los clientes?

Se plantea como **objeto de estudio**: Pruebas de Software y el **campo de acción**: Proceso de pruebas para la Plataforma Soberana GeneSIG.
El presente trabajo de diploma se estructura y desarrolla en función del siguiente **objetivo general:** Garantizar mediante la aplicación del proceso de pruebas que la Plataforma Soberana GeneSIG sea entregada al cliente con la calidad requerida. Se plantean como **objetivos específicos** de la investigación:

- Seleccionar tipos de prueba a aplicar.
- Analizar las revisiones a los documentos pertenecientes a la Plataforma Soberana GeneSIG empleando listas de chequeo.
- Diseñar casos de prueba para la Plataforma Soberana GeneSIG.
- Analizar las pruebas de la Plataforma Soberana GeneSIG.
- Evaluar los resultados obtenidos.

La **idea a defender** plantea que: El diseño y la aplicación de las pruebas garantizarán que la Plataforma Soberana GeneSIG sea entregada con calidad a los clientes.

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

1. Caracterización de la calidad de software, pruebas de software, herramientas y métodos para realizarle las pruebas a una solución informática.
2. Caracterización del proceso de pruebas.
3. Descripción del rol diseñador de casos de prueba y el rol de probador.
4. Elaboración el plan de pruebas a realizar para la Plataforma Soberana GeneSIG.
5. Evaluación de los documentos de la Plataforma Soberana GeneSIG a través de listas de

chequeo.

6. Diseño de los casos de prueba.
7. Aplicación de al menos cuatro tipos de pruebas a la Plataforma Soberana GeneSIG.
8. Evaluación del producto teniendo en cuenta el resultado de las pruebas.

La investigación exigió la utilización de **Métodos de Investigación** teóricos y empíricos:

Métodos Empíricos.

Entrevista: Para obtener información eficazmente, a través de preguntas a profesionales que dominan la problemática en la que está enmarcada la investigación. (Ver Anexo 1)

Observación: Con el fin de obtener información individual o colectiva referente al tema que se desarrolla, en un ambiente de trabajo real dentro del proyecto GeneSIG. Se observó el funcionamiento del proyecto y de la plataforma.

Métodos Teóricos.

Analítico-Sintético: Para analizar toda la bibliografía utilizada de forma exhaustiva, tomando de ella los elementos que sean de interés para la realización del proceso de pruebas.

Hipotético-deductivo: Con el objetivo de buscar la esencia de las pruebas de software, los rasgos que las caracterizan y las distinguen, para 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: Para estudiar de forma analítica la trayectoria histórica real de las pruebas de software, su evolución y desarrollo.

La presente investigación está formada por tres capítulos, de los cuales a continuación se brinda una descripción de las funciones en particular de cada uno de ellos.

EL CAPÍTULO I: FUNDAMENTOS TEÓRICOS DEL PROCESO DE PRUEBAS DE SOFTWARE hace referencia a los fundamentos teóricos del proceso de pruebas de software. Se lleva a cabo un análisis de las distintas pruebas que existen para determinar cuáles de ellas se van aplicar en la Plataforma Soberana GeneSIG.

EL **CAPÍTULO II: DISEÑO Y EJECUCIÓN DE PRUEBAS AL SISTEMA GENESIG** evidencia el Plan de Pruebas donde se establece la estrategia a seguir para aplicar las pruebas, los diseños de casos de pruebas que se utilizarán y las características que se probarán de la aplicación.

EL **CAPÍTULO III: EVALUACIÓN DE LAS PRUEBAS DEL SISTEMA GENESIG** luego de diseñar y aplicar este proceso, se muestra el cumplimiento y evaluación de las pruebas realizadas a la

Plataforma Soberana GeneSIG y así evaluar el resultado obtenido en las pruebas teniendo en cuenta las no conformidades encontradas.

CAPÍTULO 1. : FUNDAMENTOS TEÓRICOS DEL PROCESO DE PRUEBAS DE SOFTWARE.

1.1 Introducción.

En este capítulo se fundamentan los principales conceptos asociados al dominio del problema. Se realiza el análisis de las pruebas existentes y se determinan las pruebas que se le realizarán al sistema.

1.2 Definiciones Básicas.

- **Calidad.**

  Calidad es el conjunto de propiedades y características de un producto o servicio que le confieren capacidad de satisfacer necesidades, gustos y preferencias, y de cumplir las expectativas del consumidor. (CreceNegocios, 2011)

  Grado en el que un conjunto de características inherentes cumple con los requisitos. (ISO 9000, 2008)

  Luego de haber analizado varios conceptos acerca de qué es la calidad se asume el concepto dado por la ISO 9000 en el 2008 por ser este el más preciso.

- **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)

  Después de analizar la anterior definición se puede decir que software no es más que el equipamiento o soporte lógico de un computador digital, comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica.

- **Calidad de Software**

  Conjunto de características de una entidad, que le confieren la aptitud para satisfacer las necesidades establecidas y las implícitas. (ISO 8402, 1994)

  La calidad del software es el grado con el que un sistema, componente o proceso cumple los requisitos especificados y las necesidades o expectativas del cliente o usuario. (IEEE, 1990)

  Teniendo en cuenta los conceptos analizados se puede concluir que la calidad de software es el grado con el que un producto o servicio informático satisface las expectativas y los requisitos del cliente.

- **Caso de Prueba.**

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)

Por lo tanto en un caso de prueba se especifica una vía o forma para realizarle pruebas a un sistema. El analista o el probador bajo un conjunto de condiciones determinan si el sistema funciona correctamente.

- **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)

  Proceso realizado concurrentemente a través de las diferentes etapas de desarrollo del software que utiliza y mantiene la prueba, y cuyo objetivo es apoyar la disminución del riesgo de aparición de fallas y faltas en operación. (Cárdenas, 2007)

  Una vez generado el código fuente es necesario realizar pruebas al software para encontrar los errores que pueda tener la aplicación antes de ser entregada al cliente. Esta es una actividad en la cual un sistema se ejecuta en circunstancias previamente especificadas y tiene como objetivo encontrar el número máximo de errores con la mínima cantidad de esfuerzo y de tiempo. Realizar pruebas a un sistema informático no garantiza que su desarrollo esté asegurado por completo pero es un buen inicio para obtener la calidad requerida en el software.

1.3 Objetivos de las pruebas.

El proceso de prueba según Sommerville tiene dos objetivos distintos:

- Para demostrar al desarrollador y al cliente que el software satisface sus requisitos.
- Para descubrir defectos en el software en que el comportamiento de éste es incorrecto, no deseable o no cumple su especificación. (Sommerville, 2005)

Con el primer objetivo se espera que el sistema funcione correctamente para un conjunto de casos de prueba, es decir una prueba con éxito sería aquella donde el sistema funcione exitosamente sin embargo, con el segundo objetivo, los casos de pruebas se exponen para encontrar defectos, quiere decir que para este caso una prueba con éxito es aquella que muestra un defecto que hace que el sistema funcione incorrectamente.

El objetivo de las pruebas es mediante un conjunto de casos de pruebas, encontrar la mayor cantidad de errores posibles, para esto se le hacen pruebas al software sistemáticamente, garantizando así que

los desarrolladores del sistema y los clientes queden convencidos que la aplicación funciona correctamente.

1.4 Principales roles en el Proceso de Prueba de GeneSIG.

- **Diseñador de Casos de Prueba.**

  El diseñador de casos de prueba es el encargado de diseñar y ejecutar las pruebas al software con el fin de encontrar errores en la aplicación. Una vez ejecutadas las pruebas, el resultado, se plasma en el documento de No Conformidades y posteriormente se discuten con el equipo de desarrollo quienes se encargarán de decidir si proceden estas no conformidades o no.

- **Probador.**

  El papel principal del probador es organizar y realizar una serie de pruebas al software que permitan determinar la calidad del producto.

1.5 Elementos fundamentales en el proceso de prueba.

1.5.1 Proceso de Pruebas.

Las pruebas se realizan desde el comienzo del ciclo de vida del software pues en la medida que se avanza en en el desarrollo de la solución son más costosos solucionar los fallos encontrados.

Pasos para realizar pruebas de software:

- Seleccionar el objetivo de la prueba.
- Elegir cómo se va a realizar la prueba y qué elementos se deben usar.
- Diseñar los casos de prueba.
- Acordar cuáles deberían ser los resultados esperados.
- Elaborar el documento con los diseños de caso de prueba.
- Ejecutar los casos de pruebas.
- Verificar los resultados para saber si se está en presencia de un error.

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. (Betancourt, 2011)

1.5.2 Niveles de prueba

Las pruebas de software están agrupadas por niveles, los cuales están estructurados de forma ascendente para lograr una mayor organización en el proceso de prueba. Los niveles de prueba son los siguientes:

- **Prueba de Unidad**

  Prueba de unidad es el nombre que reciben los procedimientos de pruebas locales a un módulo del sistema. Por definición dichas pruebas cubren la funcionalidad propia del módulo tanto con una perspectiva de caja blanca como de caja negra; pero prestando poca o ninguna atención a la integración con otros módulos. (Garcerant, 2008)

  Las pruebas de unidad son un proceso para probar el correcto funcionamiento de cada uno de los módulos por separados. Verifican las interfaces de cada módulo con el objetivo de asegurar una fluidez total de todos los datos de la unidad que se está probando. 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.

- **Prueba de Integración**

  Una vez concluida las pruebas de unidad, es de suma importancia probar la integración de los módulos. Estas pruebas se ven afectadas por el polimorfismo y pueden presentar algunos problemas como funciones faltantes, ya que una clase puede invocar un método de otra que aún no esté implementado o que se haya olvidado, llamadas a métodos equivocados, fallas causadas por el mal manejo de memoria, en ocasiones una clase crea o destruye un objeto de otra clase y si esto no se hace correctamente puede originar varios problemas. Estas pruebas cuentan con dos estrategias: integración ascendente e integración descendente.

  - ✓ **Integración ascendente:** Se integran primero los componentes de infraestructura que proporcionan servicios comunes, tales como el acceso a base de datos y redes, y a continuación pueden añadirse los componentes funcionales (Sommerville, 2005).
  - ✓ **Integración descendente:** Se desarrolla el esqueleto del sistema en su totalidad y se le añaden los componentes. (Sommerville, 2005)

  El objetivo de las pruebas de integración es verificar el correcto ensamblaje entre los distintos componentes una vez que han sido probados unitariamente, con el fin de comprobar que interactúan correctamente a través de sus interfaces, tanto internas como externas, cubren la funcionalidad establecida y se ajustan a los requisitos no funcionales especificados en las verificaciones correspondientes. (Rojas, y otros, 2007)

- **Prueba de Validación**

Las pruebas de validación comienzan tras la culminación de la prueba de integración, cuando se han ejercitado los componentes individuales. Se ha terminado de ensamblar el software como paquete y se han descubierto y corregido los errores de interfaz. La prueba se concentra en las acciones visibles para el usuario y en la salida del sistema que éste puede reconocer. La validación se define de una forma simple, se alcanza cuando el software funciona de tal manera que satisface las expectativas del cliente. (Mansilla, 2009)

En la realización de las pruebas de validación es de suma importancia los requisitos implantados por el cliente, pues su principal objetivo es satisfacer las expectativas de este; para ello es necesario realizar pruebas sistemáticamente.

- ✓ **Prueba de Aceptación**

El objetivo de las pruebas de aceptación es validar que un sistema cumple 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. (Rojas & Barrios, 2007)

En esta prueba se valida la calidad del software esperada por el cliente y por el equipo de desarrollo. Es muy importante que la prueba no la realicen ningún integrante del equipo, para evitar conflictos de intereses, en su realización no son necesarios conocimientos informáticos, pero si conocimientos generales sobre la gestión de la calidad, pues le facilitaría el trabajo al equipo que ejecute la prueba.

- ➢ **Prueba del Sistema**

Las pruebas de sistema se realizan durante la construcción del sistema y se encargan de probar la aplicación por completo. El principal objetivo de esta prueba es verificar que el comportamiento externo del sistema de software satisface los requisitos establecidos por los clientes y futuros usuarios del mismo.

En este nivel se realizan las siguientes pruebas:

- ✓ **Prueba de Seguridad:** Permite asegurar que el sistema modificado o nuevo, incluya controles de acceso apropiados y no contenga ningún agujero de seguridad que pudiera comprometer otros sistemas. (Castro, 2010)

  Las pruebas de seguridad aseguran que los usuarios tengan acceso solo a la información que están autorizados a acceder, garantizan que no existan brechas de seguridad en el sistema pues solo el personal autorizado puede acceder a la aplicación.

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. (Betancourt, 2011)

- ✓ **Prueba de Recuperación:** Su objetivo es verificar la capacidad que tiene el sistema de recuperarse después de una falla de software o de hardware. (Castro, 2010)

- ✓ **Prueba de rendimiento:** Su objetivo es comparar el rendimiento del sistema con otros sistemas equivalentes, usando referencias comparativas bien definidas. (Castro, 2010) Estas pruebas pueden dividirse a su vez en distintos tipos:

  - ❖ **Prueba Estrés:** Se utilizan en conjunto con las pruebas de carga. Consisten en aumentar el número de usuarios concurrentes o peticiones que se utilizarán en las pruebas de carga. Generalmente, este aumento se hace doblando en cada iteración el número de peticiones. Con estas pruebas se obtiene cuál es el máximo uso admisible por el sistema. También se podría determinar qué elemento es el que lo limita, para estudiar una futura escalabilidad si es hardware, o en un rediseño si es software. (tupakamaru, 2011)

  - ❖ **Pruebas Carga:** Su principal objetivo es comprobar el comportamiento del sistema frente a un uso intensivo del mismo, por ejemplo, un gran número de usuarios concurrentes. Durante las pruebas de carga, se pueden obtener tiempos de respuesta del sistema según el número de usuarios, carga de las máquinas, uso de memoria. (tupakamaru, 2011)

- ✓ **Pruebas de Funcionalidad:** Están enfocadas a validar las funcionalidades del sistema, así como sus métodos, servicios, casos de uso y especificaciones importantes que definen la calidad de su ejecución. (Landestoy, 2011)

- ✓ **Pruebas de Usabilidad:** Se realizan comprobando factores humanos, relacionados con la estética, verificando consistencia y utilidad en las interfaces de usuarios y en los materiales o documentación de los mismos. (Landestoy, 2011)

1.5.3 Técnicas de Diseño de Caso de Prueba

Las técnicas de diseño de caso de prueba tienen como objetivo elaborar pruebas que descubran la mayor cantidad de errores posibles, permitiendo obtener como resultado un software de alta calidad.

Los tres enfoques de pruebas que se identifican son:

Técnica de prueba de Caja Blanca: consiste en centrarse en la estructura interna, es decir, en la implementación del programa y de esta forma elegir los casos de prueba.

Técnica de prueba de Caja Negra: centrada en las funcionalidades del software para derivar los casos de prueba.

El enfoque aleatorio: consiste en la utilización de modelos que generalmente son estadísticos representando las posibles entradas al programa y creando a partir de ellos los casos de prueba. (Amarán, 2010)

1.5.3.1 Pruebas de Caja Blanca

Aseguran que la operación interna del programa se ajusta a las especificaciones, y que todos los componentes internos se han probado adecuadamente. (A. Goñi, 2007)

- Usa la estructura de control para obtener los casos de prueba.
- Intentan garantizar que todos los caminos de ejecución del programa quedan probados. (A. Goñi, 2007)

La prueba de caja blanca analiza la estructura lógica del programa y cada alternativa que pueda presentarse. En ocasiones ocurre un gasto de energía y tiempo probando la lógica del programa, pero existe una vía más factible, que sería probando el cumplimiento de las funcionalidades a partir de las pruebas de Caja Negra.

Al emplear los métodos de Caja Blanca, el ingeniero de software podrá derivar casos de prueba que:

- Garanticen que todas las rutas independientes dentro del módulo se han ejercitado al menos una vez.
- Ejerciten los lados verdaderos y falsos de todas las decisiones lógicas.
- Ejecuten todos los bucles en sus límites y dentro de sus límites operacionales.
- Ejerciten estructuras de datos internos para asegurar su validez. (Pressman, 1998)

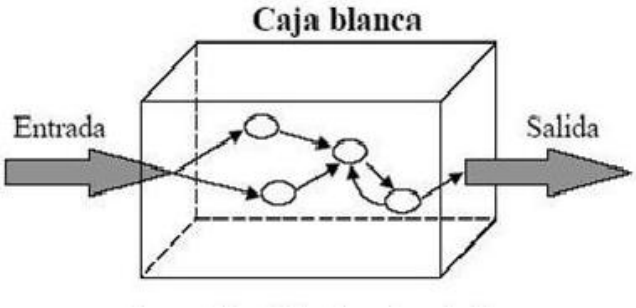


Ilustración 1: Pruebas de Caja Blanca **(**Pressman, 1998)

Dentro de las técnicas de Caja Blanca se encuentran las siguientes pruebas:

Camino básico: Consiste en diseñar un caso de prueba por cada camino independiente que exista. Este método permite medir la complejidad lógica de un diseño procedimental.

De condición: Diseñar un caso de prueba para que todas la condiciones del programa que se evalúen, sean verdaderas o falsas. Las condiciones pueden presentar errores tales como:

- ✓ Error en un operador lógico.
- ✓ Errores en variables lógicas.
- ✓ Errores en paréntesis lógicos.

De bucles: Diseñar un caso de prueba para que se intente ejecutar un bucle 0,1,…, n-1, n y n+1 veces (siendo n el número máximo). (A. Goñi, 2007)

1.5.3.2 Pruebas de Caja Negra

Las pruebas de caja negra se centran en los requisitos funcionales del software y permite obtener entradas que prueben todos los requisitos funcionales del sistema. Las pruebas de caja negra no son una opción frente a las de caja blanca, ya que encuentran errores distintos a los que se detectan en las pruebas de caja blanca y a diferencia de éstas, las pruebas de caja negra se realizan durante las últimas etapas de prueba.

Las pruebas de caja negra tratan de encontrar errores en las siguientes categorías:

- ➢ Funciones incorrectas o faltantes.
- ➢ Errores de interfaz
- ➢ Errores en estructuras de datos o en acceso a base de datos externas.
- ➢ Errores de comportamiento o desempeño.
- ➢ Errores de inicialización y término. (Pressman, 1998)

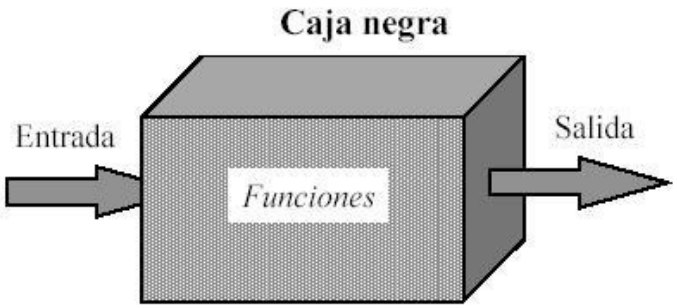


Ilustración 2: Pruebas de Caja Negra (Pressman, 1998)

El método de prueba de Caja Negra contiene técnicas que son muy utilizadas en la actualidad, entre ellas se encuentran:

Técnica de 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)

Análisis de valores límites:

Generalmente los errores tienden a producirse más en los valores límites del campo de entrada, que en cualquier otra parte, por lo que se ha diseñado la técnica de Análisis de Valores Limites (AVL). Permite seleccionar todos los casos de prueba que ejerciten los valores límites y no solo se concentra en los campos de entrada, sino también en los de salida. Las directrices de AVL son similares en muchos aspectos a las que proporciona la partición equivalente:

- Si una condición de entrada, especifica un rango delimitado por los valores a y b, se deben diseñar casos de prueba para los valores a y b, además para los valores justo por debajo y justo por encima de a y b, respectivamente.

- Si una condición de entrada especifica un número de valores, se deben desarrollar casos de pruebas que ejerciten los valores máximo y mínimo. También se deben probar los valores justo por encima y justo por debajo del máximo y del mínimo.

- Aplicar las directrices 1 y 2 a las condiciones de salida. Se deben diseñar casos de pruebas que creen un informe de salida que produzca el máximo (y el mínimo) número permitido de entradas en la tabla.

- Si las estructuras de datos internas tienen límites preestablecidos, hay que asegurarse de diseñar un caso de prueba que ejercite la estructura de datos en sus límites.

Prueba de tabla ortogonal:

Las pruebas de tabla ortogonal se les realizan a problemas donde el dominio de entrada es relativamente pequeño, pero muy grande para una prueba exhaustiva. Este método es útil cuando se deseen encontrar errores asociados a fallos localizados. (Pressman, 1998)

Prueba de comparación:

El software ha presentado situaciones, en los que su fiabilidad pasa por momentos realmente críticos, una de las causas que originan este problema es la utilización de software y hardware redundante. En estos casos varios equipos de ingeniería de software desarrollan versiones independientes de una aplicación por separado, usando las mismas especificaciones. Cuando esto sucede, es necesario probar con los mismos datos de prueba todas las versiones existentes, verificando la salida, para posteriormente ejecutar las versiones en paralelo y realizar una comparación en tiempo real de los resultados, para garantizar la consistencia. (Landestoy, 2011).

1.6 Metodología de software utilizada en el desarrollo de la Plataforma Soberana GeneSIG.

La Metodología de software utilizada en el desarrollo de la Plataforma Soberana GeneSIG es el Proceso Racional Unificado (RUP), pues esta garantiza al proyecto una forma disciplinada de asignar tareas y responsabilidades, realiza un levantamiento exhaustivo de requisitos, trata de detectar defectos en las fases iniciales y realiza el análisis y diseño, tan completo como sea posible.

1.6.1 Flujo de trabajo de pruebas según el Proceso Racional Unificado (RUP).

El flujo de trabajo de prueba según RUP plantea los siguientes objetivos:

- Encontrar y documentar los defectos que puedan afectar la calidad del software.
- Validar que el software trabaje como fue diseñado.
- Validar y probar los requisitos que debe cumplir el software.
- Validar que los requisitos fueron implementados correctamente.

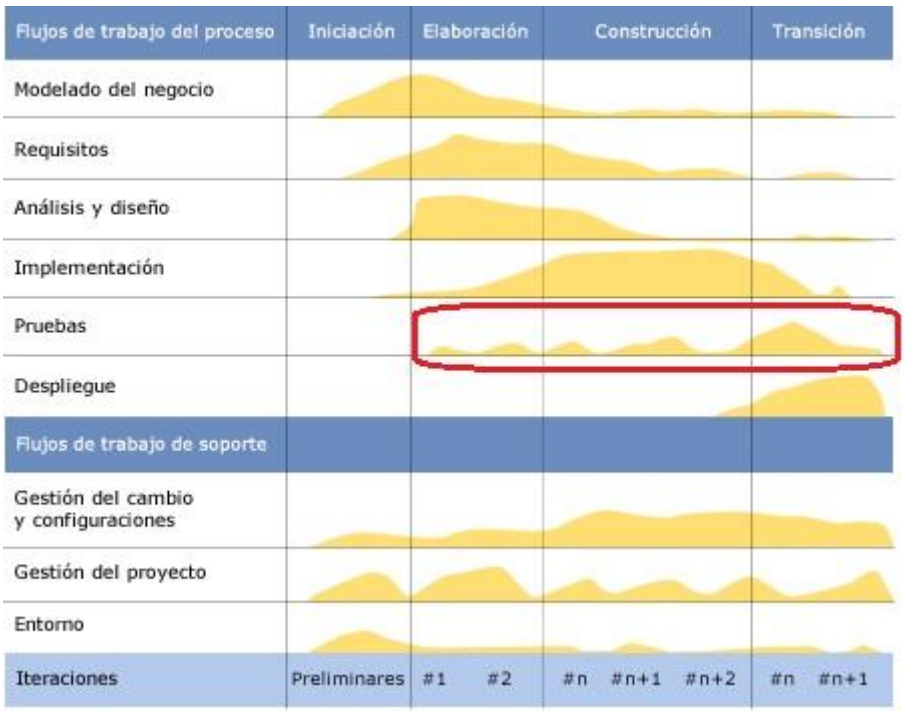


Ilustración 3: Ciclo de Vida de RUP (Zoho Writer, 2011)

El ciclo de vida de RUP se descompone en cuatro fases secuenciales: Inicio, Elaboración, Construcción y Transición. Como se muestra en la ilustración 3 el flujo de trabajo de pruebas se manifiesta durante todo el ciclo de vida del proyecto.

1.6.1.1 Artefactos del flujo de trabajo de pruebas.

Los artefactos que genera el flujo de trabajo son:

Modelo de pruebas: Está compuesto por procedimientos, componentes y los casos de pruebas. Describe cómo se prueban los componentes ejecutables en el modelo de implementación con pruebas de integración y de sistema, describe el cumplimiento de los requisitos funcionales, no funcionales del sistema.

Caso de prueba: Se especifica una vía o forma para realizarle pruebas a un sistema.

Procedimiento de prueba: Específica cómo lograr un buen manejo y dominio de las pruebas.

Componente de prueba: Automatiza uno o varios procedimientos de prueba o partes de ellos.

Plan de prueba: Es un documento en el cual se plasma la definición de las metas y objetivos del esfuerzo de pruebas y la explicación del enfoque o estrategia que se utilizará.

Defecto: Un defecto es un fallo ocurrido en el software o un problema descubierto en una revisión.
Evaluación de prueba: Es un proceso en el que se evalúan los resultados encontrados durante las pruebas.

1.6.1.2 Roles y actividades del flujo de trabajo de pruebas.

Las actividades desarrolladas durante el flujo de trabajo de pruebas son:
Planificar prueba: esta actividad ejecuta las tareas que organizan y dirigen las pruebas de software. Es la encargada de planificar los recursos y esfuerzos de las pruebas en cada iteración. Describe la estrategia de prueba que identifica los tipos de pruebas que se van a realizar, cómo hacerlo y cuándo ejecutarlas.

Diseñar prueba: esta actividad permite el diseño de los casos de pruebas que se van a realizar, además de identificar los procedimientos a seguir para elaborar dichos casos de pruebas.

Implementar prueba: el objetivo que persigue esta actividad es automatizar los posibles procedimientos de pruebas, mediante la creación de nuevos componentes.

Realizar pruebas de integración: mediante las pruebas de integración se puede construir una estructura del programa, reuniendo todos los módulos que se hayan probado, para verificar el funcionamiento de ellos juntos.

Realizar pruebas de sistema: permite la ejecución de las pruebas de sistema y la recopilación de sus resultados, ya que el software puede incorporarse a nuevos hardware.

Evaluar prueba: el propósito de esta actividad es evaluar sus esfuerzos. Establecen una comparación entre todos los resultados obtenidos y los especificados en el plan de pruebas. De esta forma se determina el número de pruebas que se deben realizar, además del nivel de calidad de software. (Landestoy, 2011)

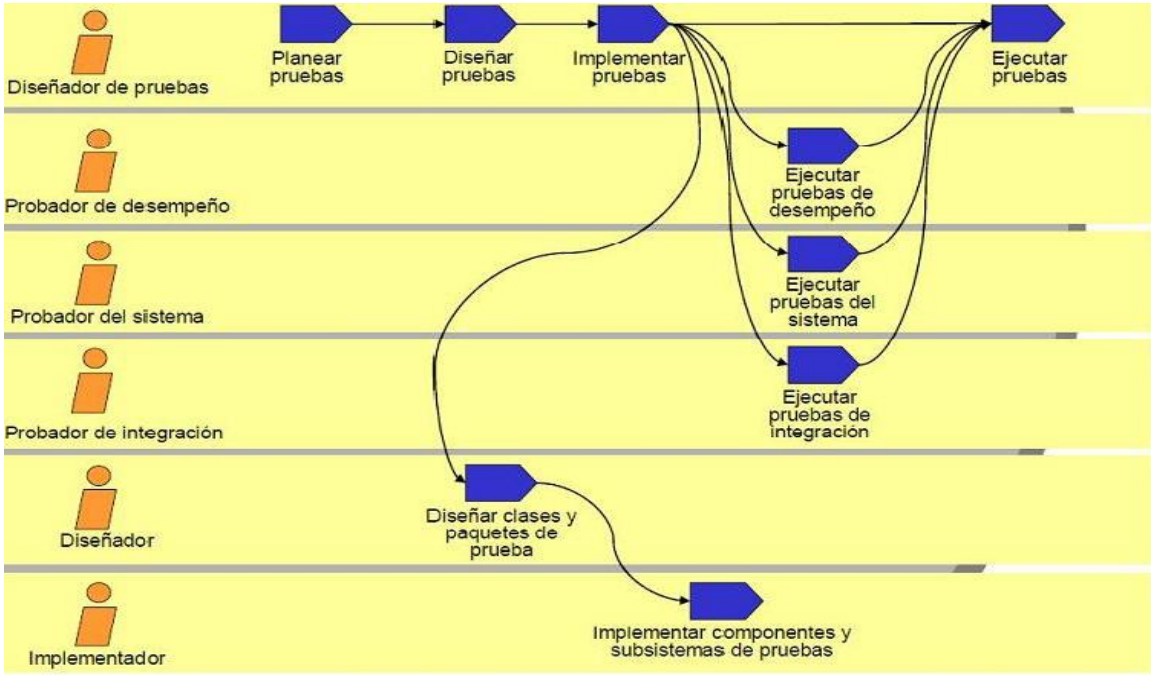


Ilustración 4: Roles y Actividades Presentes en el Proceso de Prueba

1.6.2 Relación de RUP con otras disciplinas.

- **La Disciplina de Requerimientos** tiene como propósito establecer y mantener un acuerdo con los clientes y otros interesados, acerca de qué debe hacer el sistema, además de proveer a los desarrolladores de un mejor entendimiento de los requerimientos del sistema, definir los límites (o delimitar) del sistema, proveer una base para la planeación de los contenidos técnicos de las iteraciones y para la estimación de costo y tiempo necesarios para desarrollar el sistema, definir una interfaz de usuario para el sistema, enfocada en las necesidades y objetivos del usuario. (Moreno, 2008)

- **La Disciplina de Análisis y Diseño** transforma los requerimientos a diseño del sistema, desarrolla una arquitectura robusta para éste, adapta el diseño para hacerlo corresponder con el ambiente de implementación y ajustarlo para un desempeño esperado. (Moreno, 2008)

- **La Disciplina de Implementación** define la organización del código, en términos de implementación de los subsistemas organizados en capas, implementa el diseño de elementos en términos de elementos (archivos, código fuente, binarios, ejecutables y otros), prueba los componentes desarrollados como unidades e integra los resultados individuales en un sistema ejecutable. (Moreno, 2008)

- **La Disciplina de Configuración y Control de cambio** consiste en controlar los cambios y mantener la integridad de los productos que incluye el proyecto. Los métodos, procesos y herramientas usadas para proveer la administración y configuración del cambio, pueden ser considerados como el sistema de administración de la configuración. (Moreno, 2008)

- **La Gestión de Proyecto** el propósito de la Administración de Proyectos es proveer un marco de trabajo para administrar los proyectos intensivos de software, proveer guías prácticas para la planeación, soporte, ejecución y monitoreo de proyectos, además de proveer un marco de trabajo para la administración del riesgo. (Moreno, 2008)

1.6.3 Plan de pruebas.

El plan de pruebas es un documento donde se registran todas las pruebas que se van a ejecutar en el sistema. En el que se debe dejar claro cuáles son sus objetivos, lo que se quiere probar con ellas y los resultados esperados. Un plan de pruebas tiene toda la información de los casos de prueba, sus responsables y las pruebas que están planificadas.

Un plan de pruebas Incluye:

- Propósito de la prueba.
- Roles y responsabilidades de las pruebas.
- Estrategia de prueba.
- Cronograma de Pruebas.

1.7 Herramientas para la realización de pruebas

1.7.1 Herramientas para la realización de pruebas de rendimiento

Apache JMeter: Es una aplicación de escritorio de código abierto, escrita completamente en Java por el equipo Apache Jakarta Project, para realizar pruebas de carga, para el análisis de rendimiento y verificación de funcionalidades de aplicaciones web con muchas otras funciones. Funciona de cara al servidor, como si accediera desde un navegador o muchos a la vez, sin que exista en realidad ninguno ejecutándose por detrás. JMeter captura todas las peticiones realizadas al servidor desde el “Banco de pruebas” y las introduce en el “Plan de pruebas” para ser reproducidas posteriormente. (iaranda, 2011)

Rational Performance Tester: Esta herramienta verifica el tiempo de respuesta y capacidad de ampliación aceptables para aplicaciones según cargas variables de múltiple usuarios. Permite un reconocimiento inmediato de los problemas de rendimiento con informes en tiempo real. (Johanna Rojas, 2007)

Load Runner Product Family: Herramienta para pruebas de Sistemas cliente/servidor. Permite realizar pruebas de rendimiento, pruebas de carga y afinación de aplicaciones para múltiples usuarios. (Johanna Rojas, 2007)

e – Load: Esta herramienta es un componente de la suite e-Test. Puede ser utilizada por los desarrolladores tempranamente en el proceso del desarrollo para validar la escalabilidad de la arquitectura en conjunto. Puede también ser utilizada para probar y poner a punto la aplicación completa bajo carga antes del despliegue. Utiliza las mismas Visual scripts creados para las pruebas funcionales con el e-Tester con el objetivo de emular de cientos a millares de usuarios virtuales. (Johanna Rojas, 2007)

1.7.2 Herramientas para la realización de pruebas de seguridad

Brutus: Es un crackeador de contraseñas remoto en línea, es rápido admite hasta 60 conexiones simultáneas, puede llegar hasta 2500 palabras por segundo, casi regula las conexiones según la línea y la calidad de respuesta que dé el servidor. (Ramon, 2010)

Fragroute: Entre sus características, se encuentra un lenguaje de reglas simple para retrasar, duplicar, descartar, fragmentar, superponer, imprimir, reordenar, segmentar, especificar enrutamineto de oringen y otras operaciones más en todos los paquetes salientes destinados a un host en particular, con un mínimo soporte de comportamiento aleatorio o probabilístico. Esta herramienta fue escrita para ayudar en el ensayo de sistemas de detección de intrusión, firewalls, y comportamiento básico de implementaciones de TCP/IP. (insecure, 2009)

SPIKE Proxy: Es un proxy de HTTP de código abierto que sirve para encontrar fallas de seguridad en sitios web. soporta detección de inyección de SQL automatizada, uso de fuerza bruta en formularios de entrada y detección de acceso a directorios que debieran estar fuera de los límites del sitio de web. (insecure, 2009)

Después de un estudio de las herramientas para la realización de pruebas de rendimiento y seguridad se determinó la utilización de las siguientes herramientas:

Apache JMeter:

- De las herramientas gratis, es la más completa y útil para el tipo de pruebas que se realizará.
- Brinda mayor cantidad de variantes para recoger los resultados obtenidos, que el resto de las herramientas gratis, lo que permiten hacer un análisis exhaustivo de las pruebas realizadas.

- Determina el tiempo medio de respuesta que obtendrá el usuario y el número máximo de usuarios concurrentes que pueden acceder a una página específica, o transacciones por segundo que la aplicación es capaz de soportar.

Brutus:

- Es un programa de fuerza bruta capaz de adivinar contraseñas mediante diccionarios de palabras, los diccionarios están en un .DOC.

- Soporta múltiples protocolos de autentificación de usuarios, como por ejemplo POP3, HTTP, FTP, SMB, etc.

- Permite averiguar contraseñas por el método de la fuerza bruta, es decir, probando secuencialmente cada una de las diferentes posibilidades existentes, hasta dar con la correcta.

1.8 Situación actual en la Universidad de las Ciencias Informáticas.

El Centro de Desarrollo de Software GEYSED cuenta con un grupo que es el encargado de velar por la calidad de los productos que desarrolla. Ellos son los responsables de realizar pruebas y revisiones a los productos que son liberados cada cierto tiempo. Estos productos primero son probados por el grupo de calidad del centro y luego por el centro de calidad para soluciones informáticas (CALISOFT). CALISOFT cuenta con el Laboratorio Industrial de Pruebas de Software (LIPS) en el cual se realizan pruebas de liberación, de usabilidad, pruebas funcionales, de carga, de estrés y de aceptación. Garantizando así que los productos tengan la calidad requerida y que cumplan con las especificaciones del cliente. Cada una de estas pruebas es aplicada en dependencia del tipo de aplicación informática.

1.9 Conclusiones parciales.

En el presente capítulo se logró tener un amplio conocimiento de los niveles de pruebas existentes a partir de un estudio realizado de cada uno de ellos. Se determinaron las herramientas idóneas para la realización de las pruebas de rendimiento y seguridad, teniendo en cuenta para ello las características de la Plataforma Soberana GeneSIG. Después de haber analizado la metodología de desarrollo utilizada por el proyecto se concluyó que los roles y actividades presentes en el flujo de trabajo de prueba se manifiestan durante todo el ciclo de vida del proyecto, teniendo mayor impacto en la fase final.

CAPÍTULO 2: DISEÑO Y EJECUCIÓN DE PRUEBAS AL SISTEMA GENESIG.

2.1 Introducción.

Se evidencia el Plan de Pruebas donde se establece la estrategia a seguir para aplicar las pruebas, los diseños de casos de pruebas que se utilizarán para probar el sistema y las características que se probarán de la aplicación.

2.2 Plataforma Soberana GeneSIG.

GeneSIG es un proyecto de investigación y desarrollo, que incluye varias líneas de la rama de la geomática. Su equipo de trabajo une a equipos de desarrollo como la Unidad de Compatibilización e Integración de Software para la Defensa (UCID) y GEOCUBA, que tiene como meta fortalecer la experiencia en la realización de SIG y abrir un espacio sólido en el mercado de aplicaciones de esta rama, reutilizando los componentes y funcionalidades, para personalizar los productos en cualquier negocio que lo requiera.

La plataforma GeneSIG tiene como objetivo fundamental realizar la representación geoespacial de la información asociada a cualquier tipo de negocio, además de permitir la realización de análisis de naturaleza diversa sobre dicha información. La herramienta presenta una arquitectura distribuida, empleando como base cartográfica una información certificada por especialistas funcionales que laboran en su desarrollo, y sobre ella un conjunto de objetos representados geoespacialmente que contienen información asociada, identificados a partir de las necesidades comunes de los proyectos de representación y análisis de información geográfica.

La aplicación se encuentra actualmente en la fase final de su versión 1.5. La misma está dividida en diez módulos, los cuales agrupan cuatro casos de uso arquitectónicamente significativos, estas funcionalidades son: CU Cargar Mapa, CU Configurar Mapa, CU Crear Tematización, CU Administrar Proyecto.

CU Cargar Mapa: Permite al usuario cargar una nueva capa.

CU Configurar Mapa: Permite modificar los valores de configuración correspondientes al mapa con el que se encuentra trabajando el usuario.

CU Crear Tematización: Permite al usuario realizar una tematización de tipo Corocromático y Coropleta, Gráficas Dinámicas o Símbolo Proporcional.

CU Administrar Proyecto: Permite al Administrador de Proyectos agregar, modificar o eliminar los proyectos del sistema.

2.3 Características a probar.

Los principales atributos a medir de la Plataforma Soberana GeneSIG en las Pruebas de Caja Negra son los de funcionalidad, asegurando que la aplicación funcione correctamente, que no contenga errores como funcionalidades ausentes, validación incorrecta de valores de entrada y errores ortográficos.

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

En las Pruebas de Seguridad se verificará que el usuario no pueda acceder ni modificar los datos a los que no tiene permiso y que los usuarios que no están registrados en la aplicación no puedan acceder a la herramienta.

Aplicando Pruebas de Estrés se evaluarán los atributos: extrema sobrecarga e insuficiente memoria.

2.4 Plan de Pruebas para la Plataforma Soberana GeneSIG.

- **Propósito**

El plan de pruebas para la Plataforma Soberana GeneSIG trata de cumplir los siguientes objetivos:

- ✓ Reconocer la información existente en el proyecto y los componentes que deben ser testados.
- ✓ Registrar los principales requisitos a probar.
- ✓ Determinar las estrategias de pruebas que deben emplearse.
- ✓ Reconocer los recursos necesarios que puedan requerirse.
- ✓ Registrar 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 encargado de diseñar los caso de prueba.
- ✓ 1 Probador encargado de ejecutar los diseños de caso de prueba.

- **Estrategia de prueba.**

Las pruebas que se le realizarán a la Plataforma Soberana GeneSIG están contenidas en el nivel de Prueba de Sistema, la aplicación se encuentra en la fase final de la versión 1.5 por lo que requiere probar el sistema como un todo. Dentro de este nivel se seleccionaron las siguientes pruebas:

Pruebas Funcionales: Se utilizarán las pruebas de caja negra, que pretenden demostrar que las funciones del software son operativas, que los resultados esperados ocurran cuando se usen datos válidos y el correcto funcionamiento de los interfaces del proyecto.

El método de prueba que se utiliza es Partición Equivalente, este garantiza que si un caso de prueba correspondiente a una clase de equivalencia detecta un error, el resto de los casos de prueba de dicha clase de equivalencia deben detectar el mismo error y viceversa.

Pruebas de Carga: Para verificar el funcionamiento del sistema con una cierta cantidad de usuarios concurrentes, para ello se utiliza la herramienta Apache JMeter.

- ✓ Se seleccionan 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.

Pruebas de Estrés: Con ella se comprueba el número de usuarios simultáneos que la aplicación soporta en caso de sobrecarga del sistema, para ello se utiliza la herramienta Apache JMeter.

- ✓ 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.

Pruebas de Seguridad: Serán basadas en la guía de OWASP (Proyecto Abierto de Seguridad para Aplicaciones Web). Esta guía permite mejorar las habilidades de probar y no es necesario utilizar herramientas en todas las pruebas que se realizan. Se aplica la prueba de **Comprobación del Sistema de Autenticación**, esta prueba consiste en estudiar cómo funciona el proceso de autenticación y buscar una forma de burlar este mecanismo. Para esto se utiliza la herramienta Brutus, la misma es un crackeador de contraseñas muy rápida y admite hasta 60 conexiones simultáneas.

- ✓ 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 a la Plataforma Soberana GeneSIG.
- ✓ Los resultados se observarán y se creará una plantilla resumen con éstos.

- ➢ **Tipos de pruebas y técnicas empleadas.**
- ✓ **Pruebas Funcionales.**

Objetivo: Validar si el comportamiento observado del software cumple o no con sus especificaciones.

Técnica: La técnica empleada es la de Partición Equivalente, que consiste en diseñar un caso de prueba para cada caso de uso, se prueba cada una de las funcionalidades introduciendo datos válidos e inválidos.

Herramienta: No se utilizarán herramientas para realizar esta prueba.

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

- ✓ **Pruebas de Carga.**

Objetivo: Estudiar la velocidad de respuesta ante una petición del usuario, dependiendo de la cantidad de trabajo del sistema.
Técnica: Concurrencia usuario-rendimiento.

Herramienta: JMeter.

- ✓ **Pruebas de Estrés**

Objetivo: Determinar cómo funciona el sistema en caso de ocurrir una sobrecarga.

Técnica: Concurrencia usuario-rendimiento.

Herramienta: JMeter.

- ✓ **Pruebas de Seguridad**

Objetivo: Verificar que solo el personal autorizado tenga acceso a la información.

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: Brutus.

- ➢ **Cronograma de Pruebas**

Actividad	**Fecha inicial**	**Fecha final**
Definición del plan y estrategia de pruebas.	28/Febrero/2012	29/Febrero/2012
Diseño de casos de prueba.	5/ Marzo/2012	26/Marzo/2012

Ejecución de las pruebas funcionales.	27/Marzo/2012	16/Abril/2012
Ejecución de las pruebas de carga y estrés.	17/Abril/2012	30/Abril/2011
Ejecución de las pruebas de seguridad.	1/Mayo/2012	6/Mayo/2012
Validación de las pruebas.	7/Mayo/2012	21/Mayo/2012

Tabla 1: Cronograma de pruebas

2.5 Procedimientos de prueba

Para la Plataforma Soberana GeneSIG se ha definido que los casos de uso que conforman la aplicación serán la base para los diseños de caso de prueba y guiarán la confección de los mismos. Se diseñaron doce casos de prueba de caja negra, de ellos cuatro son arquitectónicamente significativos y son los que se presentan en la presente investigación (Ver Anexo 2). En cada uno de los procedimientos se realiza una descripción de la funcionalidad, las condiciones de ejecución y los escenarios a probar.

CU Cargar Mapa.

Descripción de la funcionalidad

El caso de uso se inicia cuando el usuario selecciona la opción Cargar Mapa, y termina cuando se visualiza el mapa escogido con sus respectivas capas.

Condición de ejecución.

El usuario debe estar autenticado y para poder obtener información de una capa, esta debe ser seleccionable y visible.

Escenarios a probar

Tabla 2 Caso de Prueba **Cargar Mapa**.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1:	EC 1.1: “ Cargar	El usuario selecciona la opción Cargar Mapa

“Cargar Mapa ”	Mapa correctamente”	y el sistema muestra la ventana Cargar Mapas del Usuario. El usuario selecciona el mapa que quiere visualizar y da clic en el botón Aceptar entonces el sistema muestra una ventana de verificación ¿Desea salvar el mapa actual? , el dice que sí y el sistema muestra una ventana Nuevo nombre para el mapa actual para que el usuario introduzca el nombre con el que identificará el nuevo mapa. El usuario introduce el nuevo nombre que tendrá el mapa seleccionado y da clic en el botón Aceptar. El sistema guarda los cambios y visualiza en el árbol de capas el nuevo mapa creado con las capas que lo componen.
	EC 1.2: “Cancelar el cargar mapa ””	El usuario selecciona la opción Cargar Mapa y el sistema muestra la ventana Cargar Mapas del Usuario, da clic en el botón cancelar y el sistema cierra la ventana.
	EC 1.3: “No salvar mapa actual ”	El usuario selecciona la opción Cargar Mapa y el sistema muestra la ventana Cargar Mapas del Usuario. El usuario selecciona el mapa que quiere visualizar y da clic en el botón Aceptar entonces el sistema muestra una ventana de verificación ¿Desea salvar el mapa actual?, el usuario da clic en el botón No y el sistema cierra la ventana.
	EC 1.4: “Cancelar nuevo nombre para mapa actual ”	El usuario selecciona la opción Cargar Mapa y el sistema muestra la ventana Cargar Mapas del Usuario. El usuario selecciona el mapa que quiere visualizar y da clic en el botón Aceptar entonces el sistema muestra

		una ventana de verificación ¿Desea salvar el mapa actual?, el usuario da clic en el botón si y el sistema muestra una ventana Nuevo nombre para el mapa actual para que el usuario introduzca el nombre con el que identificará el nuevo mapa. El usuario da clic en el botón cancelar y el sistema cierra la ventana.

Tabla 2: Caso de prueba Cargar Mapa

Matriz de datos.

SC 1: "Cargar Mapa"

ID del escenario	Escenario	Nombre	Respuesta del sistema
EC 1.1	Cargar Mapa correctamente	V/"pepe"	El sistema guarda los cambios y visualiza en el árbol de capas el nuevo mapa creado con las capas que lo componen
EC 1.2	Cancelar el cargar mapa	N/A	El sistema cierra la ventana.
EC 1.3	No salvar mapa actual	N/A	El sistema cierra la ventana.
EC 1.4	Cancelar nuevo nombre para mapa actual	V/"pepe"	El sistema cierra la ventana.

Tabla 3: Matriz de dato de SC1 Cargar Mapa

2.6 Caso de prueba de Carga y Estrés.

Para las Pruebas de Carga se diseñaron cuatro 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 "Cargar Mapa".

Tipo de prueba: Prueba de Carga.

Funcionalidades a probar:

Funcionalidad(es)
Cargar Mapa

Tabla 4: Funcionalidades a probar

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

Tabla 5: Caso de prueba de Carga para Cargar Mapa

Para realizar las pruebas de Estrés se utilizarán los mismos diseños de casos prueba de Carga, pero para este caso se incrementó la cantidad de usuarios concurrentes. 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.

2.7 Conclusiones parciales.

En este capítulo se realizó una breve descripción de la aplicación donde se expresa sus características fundamentales. Se refleja el Plan de Pruebas, donde se pudo elegir las técnicas y métodos de pruebas a emplear, que garantizan con su ejecución que el sistema esté libre de errores. Se confeccionó el procedimiento de pruebas, este contiene los diseños de caso de pruebas, los cuales guían la ejecución de dichas pruebas.

CAPITULO 3: EVALUACIÓN DE LAS PRUEBAS DEL SISTEMA GENESIG.

3.1 Introducción

Este capítulo se explica la factibilidad de la aplicación de pruebas a la Plataforma GeneSIG a través de un análisis de los resultados obtenidos. Por otra parte, se verifica el cumplimiento del objetivo de la presente investigación y por tanto la resolución del problema.

Los resultados pueden ser reflejados de formas diferentes, en este capítulo específicamente se describen por cada tipo de prueba. Para esto se hace un resumen general de los resultados obtenidos mediante el empleo de gráficas que facilitan el entendimiento.

3.2 Resultados de la prueba de Caja Negra.

Después de haber ejecutado todos los diseños de Casos de Prueba se detectaron errores tales como: funcionalidades incorrectas, errores ortográficos, validación incorrecta de datos y errores de excepciones, para un total de 38 No Conformidades (Ver Anexo 4).

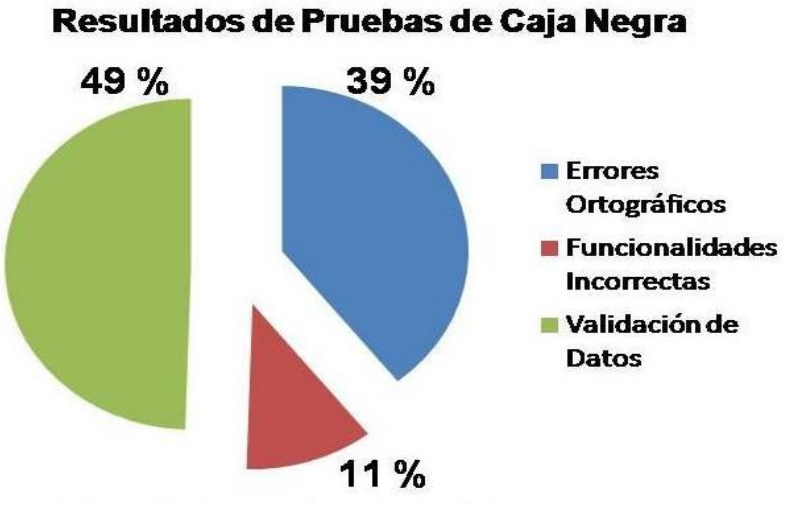


Ilustración 5: Resultados de las Pruebas de Caja Negra

La gráfica muestra los resultados alcanzados en las Pruebas de Caja Negra representando en porciento las No conformidades encontradas durante la ejecución de las pruebas. Como se puede ver el 39% representa los errores ortográficos, un 49% simboliza la validación de datos y un 11% constituyen las funcionalidades incorrectas. Al realizar un análisis de los resultados, se concluye que la mayor cantidad de no conformidades se refiere a la validación de datos en la aplicación.

Se llegó a la conclusión de que la Prueba de Caja Negra cumplió los objetivos previamente establecidos que era encontrar la mayor cantidad de errores posible para poder eliminarlos. En la siguiente gráfica quedan reflejados el cumplimiento de los mismos con una pequeña comparación entre la cantidad de No conformidades detectadas en la primera iteración [1] y las encontradas en la segunda iteración[2]. Se puede observar que en la primera iteración se encontraron una buena cantidad de errores, esto contribuyó a que el proyecto pudiera liberar la aplicación en el tiempo requerido y demuestra que el diseñador de casos de prueba realizó un buen trabajo.

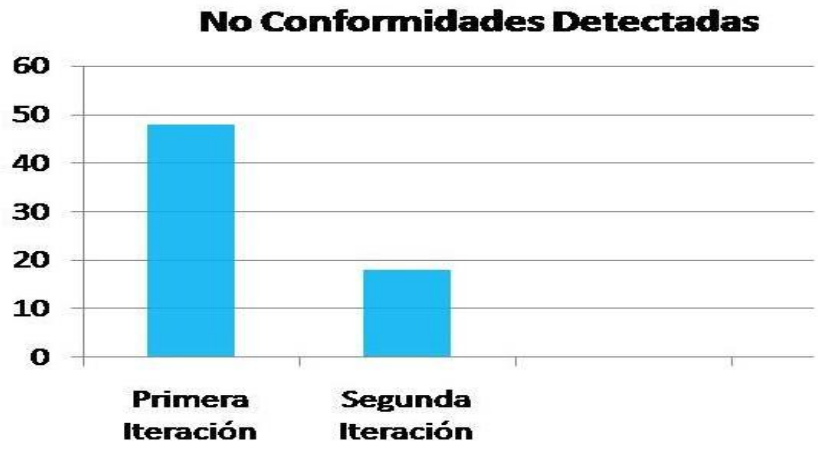


Ilustración 6: No Conformidades Detectadas

3.3 Resultados de las Pruebas de Seguridad.

Al aplicar las Pruebas de Comprobación del Sistema de Autenticación se pudo probar que la Plataforma Soberana GeneSIG es sensible a la entrada de usuarios. De los usuarios y contraseñas que se probaron al cargarlos en la herramienta Brutus, 5 usuarios pudieron penetrar en la aplicación, lo cual demuestra que existen deficiencias en el mecanismo de seguridad de la aplicación. Esto puede traer consigo que usuarios maliciosos penetren en la aplicación y se apoderen de información confidencial o introduzcan algún virus con el objetivo de destruir la aplicación.

[1] Primera Iteración: Se hace referencia a la ejecución de los diseños de casos de prueba del presente Trabajo de Diploma.

[2] Segunda Iteración: Se hace referencia a la ejecución de los diseños de casos de pruebas realizados por CALISOFT en la liberación de la versión 1.5 de la Plataforma Soberana GeneSIG.

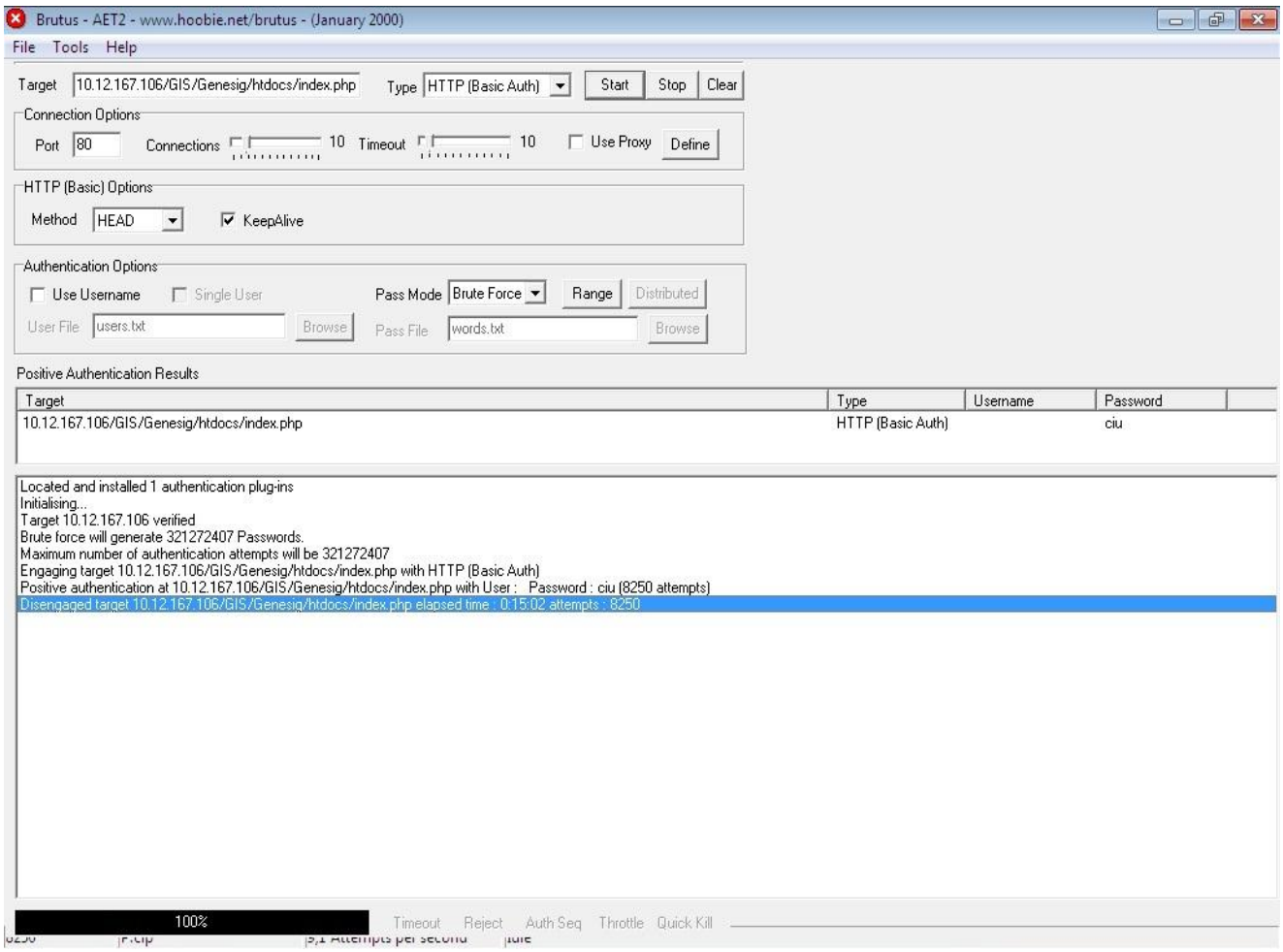


Ilustración 7: Resultados de Brutus

Las Pruebas de Gestión de Caché de Navegación y de Salida de Sesión demuestran que la aplicación después de estar inactiva durante un período de tiempo, no cierra la sesión del usuario en uso, permitiendo posibles penetraciones de usuarios maliciosos.

Luego de haber planteado el resultado de las Pruebas de Seguridad se puede concluir que la Plataforma Soberana GeneSIG 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.

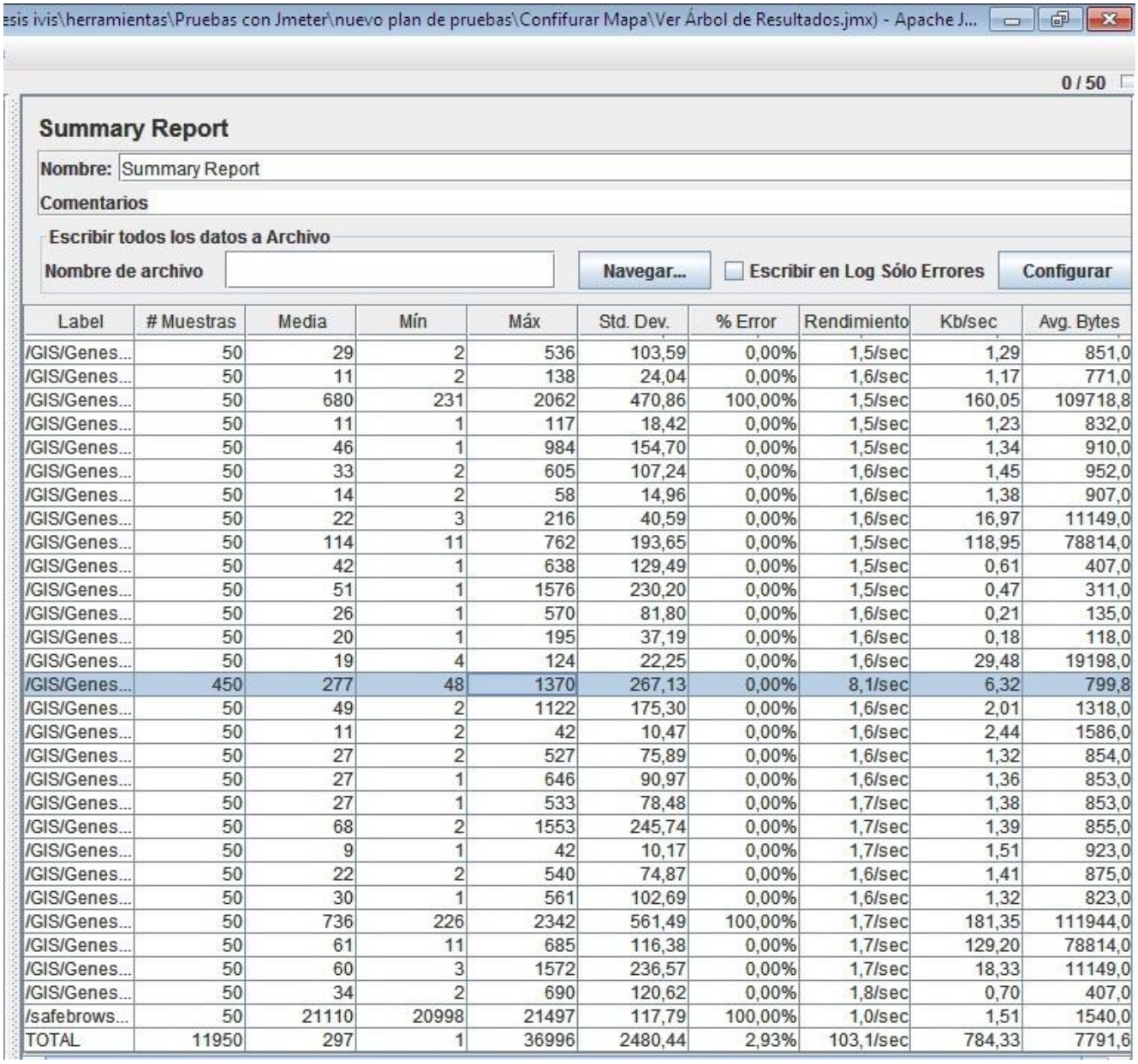
3.4 Resultados de las pruebas de Carga y Estrés.

Resultados de las Pruebas de Carga.

Caso de prueba “Configurar Mapa”.

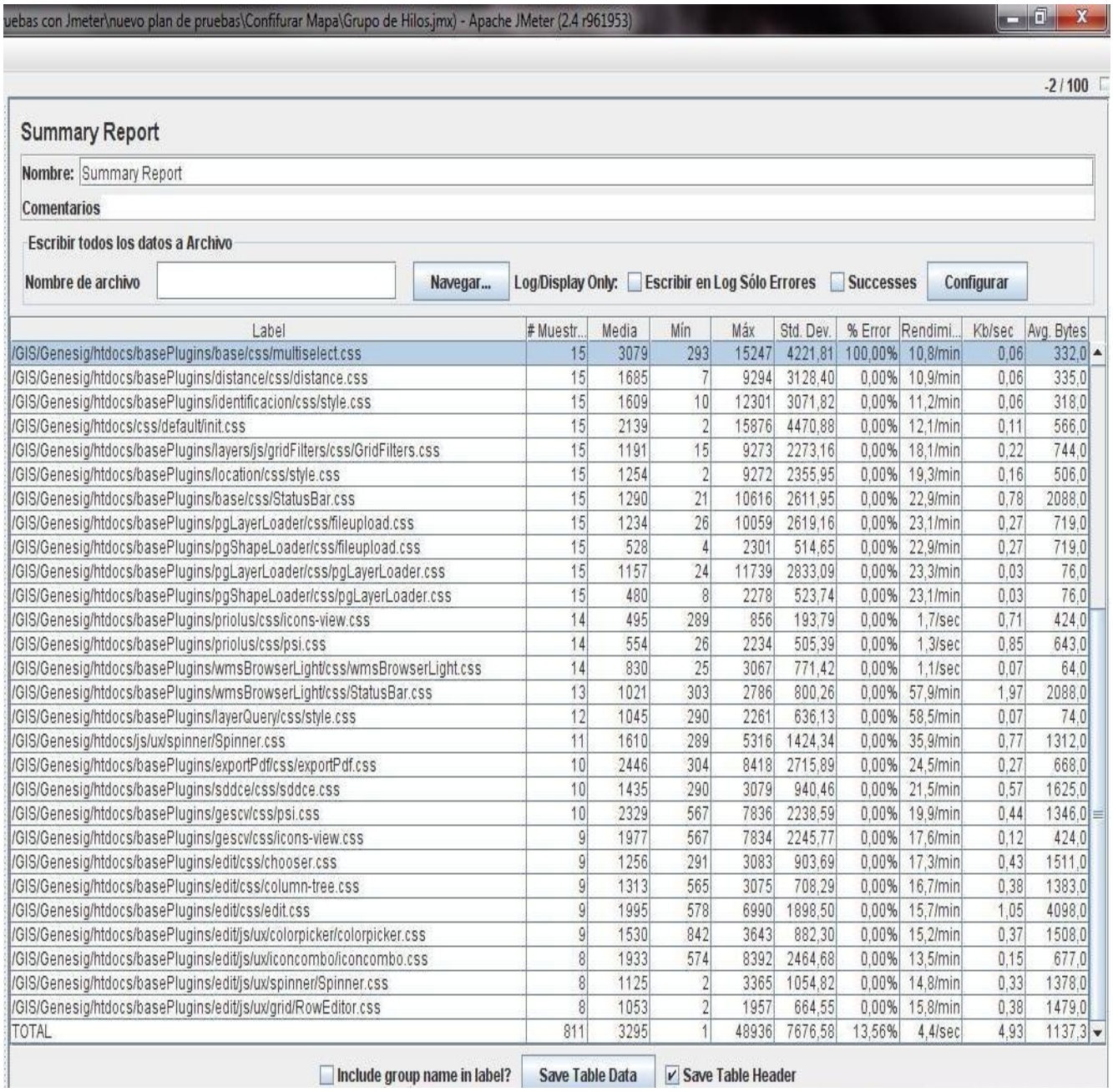
Tipo de prueba: Prueba de Carga.

Durante la ejecución de la prueba de Carga el CPU se mantuvo entre el 85 y el 100% de utilización, la memoria física estuvo por debajo del 50%. Se realizó una primera prueba con 50 usuarios donde la respuesta del sistema fue 103.1 segundos, el porciento de error fue de 3.67% y se alcanzaron 11 950 peticiones para esa cantidad de usuarios. Se llevó a cabo una segunda prueba para 100 usuarios para este caso la respuesta del sistema fue 4.4 segundos, el % de error 13.56% y la cantidad de peticiones 811.



Label	# Muestras	Media	Mín	Máx	Std. Dev.	% Error	Rendimiento	Kb/sec	Avg. Bytes
/GIS/Genes...	50	29	2	536	103,59	0,00%	1,5/sec	1,29	851,0
/GIS/Genes...	50	11	2	138	24,04	0,00%	1,6/sec	1,17	771,0
/GIS/Genes...	50	680	231	2062	470,86	100,00%	1,5/sec	160,05	109718,8
/GIS/Genes...	50	11	1	117	18,42	0,00%	1,5/sec	1,23	832,0
/GIS/Genes...	50	46	1	984	154,70	0,00%	1,5/sec	1,34	910,0
/GIS/Genes...	50	33	2	605	107,24	0,00%	1,6/sec	1,45	952,0
/GIS/Genes...	50	14	2	58	14,96	0,00%	1,6/sec	1,38	907,0
/GIS/Genes...	50	22	3	216	40,59	0,00%	1,6/sec	16,97	11149,0
/GIS/Genes...	50	114	11	762	193,65	0,00%	1,5/sec	118,95	78814,0
/GIS/Genes...	50	42	1	638	129,49	0,00%	1,5/sec	0,61	407,0
/GIS/Genes...	50	51	1	1576	230,20	0,00%	1,5/sec	0,47	311,0
/GIS/Genes...	50	26	1	570	81,80	0,00%	1,6/sec	0,21	135,0
/GIS/Genes...	50	20	1	195	37,19	0,00%	1,6/sec	0,18	118,0
/GIS/Genes...	50	19	4	124	22,25	0,00%	1,6/sec	29,48	19198,0
/GIS/Genes...	450	277	48	1370	267,13	0,00%	8,1/sec	6,32	799,8
/GIS/Genes...	50	49	2	1122	175,30	0,00%	1,6/sec	2,01	1318,0
/GIS/Genes...	50	11	2	42	10,47	0,00%	1,6/sec	2,44	1586,0
/GIS/Genes...	50	27	2	527	75,89	0,00%	1,6/sec	1,32	854,0
/GIS/Genes...	50	27	1	646	90,97	0,00%	1,6/sec	1,36	853,0
/GIS/Genes...	50	27	1	533	78,48	0,00%	1,7/sec	1,38	853,0
/GIS/Genes...	50	68	2	1553	245,74	0,00%	1,7/sec	1,39	855,0
/GIS/Genes...	50	9	1	42	10,17	0,00%	1,7/sec	1,51	923,0
/GIS/Genes...	50	22	2	540	74,87	0,00%	1,6/sec	1,41	875,0
/GIS/Genes...	50	30	1	561	102,69	0,00%	1,6/sec	1,32	823,0
/GIS/Genes...	50	736	226	2342	561,49	100,00%	1,7/sec	181,35	111944,0
/GIS/Genes...	50	61	11	685	116,38	0,00%	1,7/sec	129,20	78814,0
/GIS/Genes...	50	60	3	1572	236,57	0,00%	1,7/sec	18,33	11149,0
/GIS/Genes...	50	34	2	690	120,62	0,00%	1,8/sec	0,70	407,0
/safebrows...	50	21110	20998	21497	117,79	100,00%	1,0/sec	1,51	1540,0
TOTAL	11950	297	1	36996	2480,44	2,93%	103,1/sec	784,33	7791,6

Ilustración 8: Resultados del JMeter para 50 usuarios



Label	# Muestr...	Media	Mín	Máx	Std. Dev.	% Error	Rendimi...	Kb/sec	Avg. Bytes
/GIS/Genesig/htdocs/basePlugins/base/css/multiselect.css	15	3079	293	15247	4221,81	100,00%	10,8/min	0,06	332,0
/GIS/Genesig/htdocs/basePlugins/distance/css/distance.css	15	1685	7	9294	3128,40	0,00%	10,9/min	0,06	335,0
/GIS/Genesig/htdocs/basePlugins/identificacion/css/style.css	15	1609	10	12301	3071,82	0,00%	11,2/min	0,06	318,0
/GIS/Genesig/htdocs/css/default/init.css	15	2139	2	15876	4470,88	0,00%	12,1/min	0,11	566,0
/GIS/Genesig/htdocs/basePlugins/layers/js/gridFilters/css/GridFilters.css	15	1191	15	9273	2273,16	0,00%	18,1/min	0,22	744,0
/GIS/Genesig/htdocs/basePlugins/location/css/style.css	15	1254	2	9272	2355,95	0,00%	19,3/min	0,16	506,0
/GIS/Genesig/htdocs/basePlugins/base/css/StatusBar.css	15	1290	21	10616	2611,95	0,00%	22,9/min	0,78	2088,0
/GIS/Genesig/htdocs/basePlugins/pgLayerLoader/css/fileupload.css	15	1234	26	10059	2619,16	0,00%	23,1/min	0,27	719,0
/GIS/Genesig/htdocs/basePlugins/pgShapeLoader/css/fileupload.css	15	528	4	2301	514,65	0,00%	22,9/min	0,27	719,0
/GIS/Genesig/htdocs/basePlugins/pgLayerLoader/css/pgLayerLoader.css	15	1157	24	11739	2833,09	0,00%	23,3/min	0,03	76,0
/GIS/Genesig/htdocs/basePlugins/pgShapeLoader/css/pgLayerLoader.css	15	480	8	2278	523,74	0,00%	23,1/min	0,03	76,0
/GIS/Genesig/htdocs/basePlugins/priolus/css/icons-view.css	14	495	289	856	193,79	0,00%	1,7/sec	0,71	424,0
/GIS/Genesig/htdocs/basePlugins/priolus/css/psi.css	14	554	26	2234	505,39	0,00%	1,3/sec	0,85	643,0
/GIS/Genesig/htdocs/basePlugins/wmsBrowserLight/css/wmsBrowserLight.css	14	830	25	3067	771,42	0,00%	1,1/sec	0,07	64,0
/GIS/Genesig/htdocs/basePlugins/wmsBrowserLight/css/StatusBar.css	13	1021	303	2786	800,26	0,00%	57,9/min	1,97	2088,0
/GIS/Genesig/htdocs/basePlugins/layerQuery/css/style.css	12	1045	290	2261	636,13	0,00%	58,5/min	0,07	74,0
/GIS/Genesig/htdocs/js/ux/spinner/Spinner.css	11	1610	289	5316	1424,34	0,00%	35,9/min	0,77	1312,0
/GIS/Genesig/htdocs/basePlugins/exportPdf/css/exportPdf.css	10	2446	304	8418	2715,89	0,00%	24,5/min	0,27	668,0
/GIS/Genesig/htdocs/basePlugins/sddce/css/sddce.css	10	1435	290	3079	940,46	0,00%	21,5/min	0,57	1625,0
/GIS/Genesig/htdocs/basePlugins/gescv/css/psi.css	10	2329	567	7836	2238,59	0,00%	19,9/min	0,44	1346,0
/GIS/Genesig/htdocs/basePlugins/gescv/css/icons-view.css	9	1977	567	7834	2245,77	0,00%	17,6/min	0,12	424,0
/GIS/Genesig/htdocs/basePlugins/edit/css/chooser.css	9	1256	291	3083	903,69	0,00%	17,3/min	0,43	1511,0
/GIS/Genesig/htdocs/basePlugins/edit/css/column-tree.css	9	1313	565	3075	708,29	0,00%	16,7/min	0,38	1383,0
/GIS/Genesig/htdocs/basePlugins/edit/css/edit.css	9	1995	578	6990	1898,50	0,00%	15,7/min	1,05	4098,0
/GIS/Genesig/htdocs/basePlugins/edit/js/ux/colorpicker/colorpicker.css	9	1530	842	3643	882,30	0,00%	15,2/min	0,37	1508,0
/GIS/Genesig/htdocs/basePlugins/edit/js/ux/iconcombo/iconcombo.css	8	1933	574	8392	2464,68	0,00%	13,5/min	0,15	677,0
/GIS/Genesig/htdocs/basePlugins/edit/js/ux/spinner/Spinner.css	8	1125	2	3365	1054,82	0,00%	14,8/min	0,33	1378,0
/GIS/Genesig/htdocs/basePlugins/edit/js/ux/grid/RowEditor.css	8	1053	2	1957	664,55	0,00%	15,8/min	0,38	1479,0
TOTAL	811	3295	1	48936	7676,58	13,56%	4,4/sec	4,93	1137,3

Ilustración 9: Resultados del JMeter para 100 usuarios

Analizando los valores de cantidad de peticiones, rendimiento y porciento de error se puede concluir que el sistema no responde satisfactoriamente, pues no permite que un mismo usuario se conecte muchas veces a la aplicación. Se probaron con varias muestras, aumentando y disminuyendo la cantidad de usuarios concurrentes al sistema y siempre fue la misma respuesta variando el número de usuarios.

Resultados generales de rendimiento para Pruebas de Carga.

Rendimiento para 50 usuarios concurrentes.

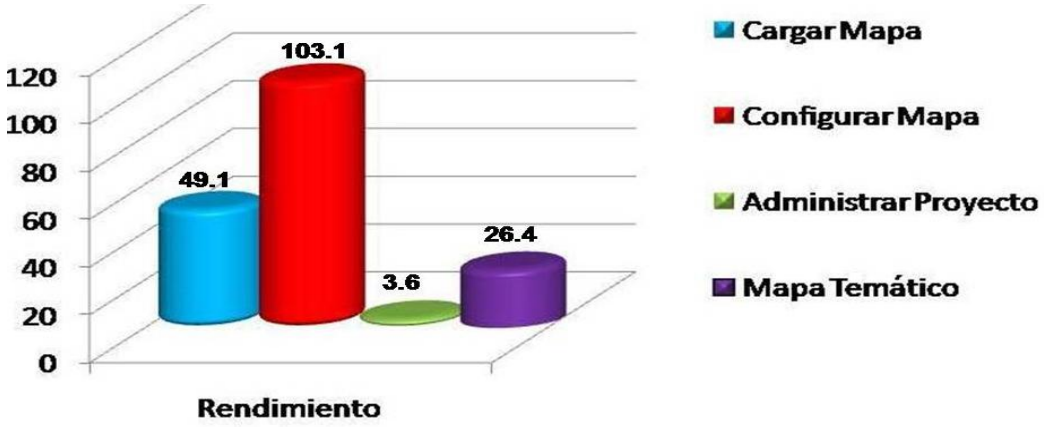


Ilustración 10: Rendimiento para 50 usuarios

La gráfica representa el comportamiento del sistema en cuanto al rendimiento para la cantidad de 50 usuarios. En el caso de Cargar Mapa se obtuvo un rendimiento de 49.1 segundos, Configurar Mapa 103.1 segundos, Administrar Proyecto 3.6 segundo y Crear Tematización 26.4 segundos.

Rendimiento para 70 usuarios concurrentes.

Ilustración 11: Rendimiento para 70 usuarios

La gráfica representa el comportamiento del sistema en cuanto al rendimiento para la cantidad de 70 usuarios. En el caso de Cargar Mapa se obtuvo un rendimiento de 17.5 segundos, Configurar Mapa 24.9 segundos, Administrar Proyecto 6.8 segundo y Crear Tematización 2.4 segundos.

Rendimiento para 100 usuarios concurrentes.

Ilustración 12: Rendimiento para 100 usuarios

La gráfica representa el comportamiento del sistema en cuanto al rendimiento para la cantidad de 100 usuarios. En el caso de Cargar Mapa se obtuvo un rendimiento de 2.7 segundos, Configurar Mapa 4.4 segundos, Administrar Proyecto 4.9 segundo y Crear Tematización 17.8 segundos.

Resultados de la Prueba de Estrés.

Tipo de prueba: Prueba de Estrés.

Funcionalidades:

Funcionalidad(es)
Configurar Mapa.
Cargar Mapa.
Administrar Proyecto.
Crear Tematización

Tabla 6: Funcionalidades a probar para pruebas de Estrés

Variables	**Valores**
Número de usuarios concurrentes: (número de hilos)	150

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

Tabla 7: Caso de Prueba de Estrés para las funcionalidades

Las pruebas de estrés se realizaron aumentando el número de usuarios de los casos de pruebas de carga, comprobando así que la aplicación entra en un estado al que se le llama estrés cuando se conectan más de 100 usuarios en el sistema. Esto se pudo evidenciar porque al incrementar la cifra, la herramienta cargó todas las conexiones pero no se pudo ver el rendimiento de la misma pues se bloqueó y solo se pudo parar la ejecución del plan. Por estas razones se puede afirmar que la cantidad máxima de usuarios concurrentes que soporta la Plataforma es 150.

3.5 Conclusiones parciales.

Durante el proceso de prueba se detectaron distintos tipos de errores que podrán ser corregidos, con el objetivo de entregar una aplicación libre de fallos. Se puede afirmar que la Plataforma Soberana GeneSIG no cumple con las normas de calidad para realizar la entrega del producto al cliente. Luego de la ejecución de las pruebas se documentaron todos los defectos encontrados que serán de gran ayuda para el equipo de desarrollo en posteriores versiones.

Conclusiones Generales.

Los fundamentos teóricos permitieron tomar una correcta decisión al elegir los tipos de pruebas que se aplicaron a la Plataforma Soberana GeneSIG. Mediante la confección del Plan de Pruebas se logró definir los pasos a seguir para realizar cada una de las pruebas escogidas y permitió planificar este proceso teniendo en cuenta el tiempo y los recursos disponibles.

- La ejecución de pruebas de carga y estrés con la herramienta JMeter, permitieron corregir errores en cuanto al rendimiento de la aplicación.
- Las pruebas de Seguridad aplicadas a la Plataforma con el empleó de la herramienta Brutus permitió detectar errores como posibles penetraciones de usuarios no autorizados a la Plataforma.
- Las pruebas de caja negra mediante la técnica de partición equivalente arrojaron resultados relacionado con los requisitos funcionales del sistema.
- Con la aplicación del proceso de pruebas se detectaron varios tipos de errores que ayudaron al proyecto entregar la aplicación libre de errores, lo cual demuestra que se dio cumplimiento a los objetivos planteados que es encontrar la mayor cantidad de errores en el menor tiempo posible.

Recomendaciones

- 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.

Bibliografía

Amarán, Tony Moisés Fernández. 2010. Sistema de Gestión de Datos Geológicos. Módulo Inventario de Petróleo y Gas. La Habana : s.n., 2010.

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

Betancourt, Liudmila. 2011. Diseño y aplicación de pruebas. La Habana : s.n., 2011.

Cárdenas, Maria Clara Choucair. 2007. Scribd. Scribd. [En línea] 19 de septiembre de 2007. [Citado el: 25 de abril de 2012.] http://es.scribd.com/doc/55972214/Pruebas-de-Software-ACIS.

Castro, Nelson. 2010. Seguridad de la Información. Seguridad de la Información. [En línea] 22 de noviembre de 2010. [Citado el: 16 de febrero de 2012.] https:// necastro.blogspot.com/2010/11/clasificaciones-de-pruebas-de-software.html..

CreceNegocios. 2011. Crece Negocios.com. Crece Negocios.com. [En línea] 20 de julio de 2011. [Citado el: 25 de abril de 2012.] http://www.crecenegocios.com/concepto-de-calidad/.

Garcerant, Iván. 2008. Tecnología y Synergix. . Tecnología y Synergix. [En línea] 15 de marzo de 2008. [Citado el: 15 de febrero de 2012.] http://synergix.wordpress.com/2008/03/15/definimos-pruebas-de-unidad-como.

iaranda. 2011. www.linuxero.es. www.linuxero.es. [En línea] 2 de febrero de 2011. [Citado el: 23 de febrero de 2012.] http://iaranda.wordpress.com/2011/02/02/test-de-rendimiento-en-alfresco/.

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

insecure.org. 2009. Las 75 Herramientas de Seguridad Más Usadas. *Las 75 Herramientas de Seguridad Más Usadas.* [Online] 2009. [Cited: abril 28, 2012.] http://sectools.org/tools/tools-es.html.

ISO 8402. 1994. Gestion Y Garantia De La Calidad. 1994.

ISO 9000. 2008. Sistemas de Gestión de la Calidad - Requisitos. 2008.

Landestoy, Liliam Rodriguez. 2011. Diseño de una estrategia de pruebas para la plataforma GeneSIG y sus aplicativos. 2011.

López, Carlos. 2001. Gestiopolis. Gestiopolis. [En línea] 2001. [Citado el: 20 de enero de 2012.] http://www.gestiopolis.com/canales/gerencial/articulos/27/asesis.htm..

BIBLIOGRAFIA

Mansilla, Ricardo. 2009. slideshere. slideshere. [En línea] 4 de nobiembre de 2009. [Citado el: 17 de febrero de 2012.] http://www.slideshare.net/cliceduca/pruebas-de-software-2420588..

Moreno, Iván Tapia. 2008. [En línea] 2008. [Citado el: 27 de febrero de 2012.] http://www.google.com.cu/url?sa=t&rct=j&q=RUP+y+sus+diciplinas&source=web&cd=1&ved=0CCgQFjAA&url=http%3A%2F%2Fwww.itson.mx%2Fdii%2Fitapia%2FConceptos%2520de%2520RUP.doc&ei=wtpLT-n4C6qB0QGR7tCnDg&usg=AFQjCNEQpd7TePSdaLSlWIwm2X_JClnGNg&cad=rja.

Murcia, Universidad de. 2006. www.um.es. www.um.es. [En línea] 30 de diciembre de 2006. [Citado el: 24 de febrero de 2012.] http://www.um.es/docencia/barzana/IAGP/Iagp2.html..

Pressman, Roger. 1998. Ingeniería del software. 1998.

Pressman, Roger S. 2005. *Ingeniería de Software un Enfoque Práctico.* 2005.

Ramon, Pablo Saiz. 2010. pablosaiz.blogspot. pablosaiz.blogspot. [En línea] 6 de octubre de 2010. [Citado el: 2 de mayo de 2012.] http://pablosaiz.blogspot.com/2010/10/tutorial-brutus.html.

Rojas, Johanna y Barrios, Emilio. 2007. INVESTIGACIÓN SOBRE ESTADO DEL ARTE EN DISEÑO Y APLICACIÓN DE PRUEBAS DE SOFTWARE. INVESTIGACIÓN SOBRE ESTADO DEL ARTE EN DISEÑO Y APLICACIÓN DE PRUEBAS DE SOFTWARE. [En línea] 2007. [Citado el: 16 de febrero de 2012.] http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiante.

Sommerville, Ian. 2005. Ingeniería de Software Séptima Edicción. 2005.

tupakamaru. 2011. Blog de WordPress.com. Blog de WordPress.com. [En línea] 27 de enero de 2011. [Citado el: 2012 de febrero de 27.] http://tupakamaru.wordpress.com/2011/01/27/pruebas-de-rendimiento/.

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

Zoho Writer. 2011. Zoho Writer. *Zoho Writer.* [En línea] 5 de Marzo de 2011. [Citado el: 30 de Mayo de 2012.] https://export.writer.zoho.com/public/cone_cahuitl/tarea---metodologia-rup/fullpage.

.....

Glosario de Término.

Cartografía

Ciencia y arte de hacer mapas y cartas.

Capa

Conjunto lógico de elementos temáticos descritos y almacenados en una biblioteca. Las *Layers* o capas organizan la biblioteca según temas.

Caso de uso arquitectonicamente significativo

Caso de uso que ayuda a mitigar los riesgos más importantes, aquel que es el más importante para los usuarios del sistema y aquel que ayuda a cubrir todas las funcionalidades significativas. *[RUP]*

Coordenadas geográficas

Un sistema de coordenadas curvas definido sobre el elipsoide de referencia. Se expresan como Longitud (lon), Latitud (lat) y Altura (h) donde la lon y la lat son medidas angulares desde el meridiano origen y el ecuador respectivamente; h es la altura sobre el elipsoide de referencia.

GeneSIG

Nombre identificativo de la Plataforma Soberana para el Desarrollo de Sistemas de Información Geográfica. Proviene de la fonética del término “génesis” que significa creación, origen. El prefijo *Gene* simboliza genérico, y *SIG*, Sistema de Información Geográfica.

Información geoespacial

(1) Tiene componentes como son los Mapas, Visualización de Mapas, Objetos Geográficos, Geometrías, Relaciones.

(2) Datos distribuidos, servicios compartidos, acoplados fuerte o débilmente a los geodatos.

No Conformidades

Problema detectado en un artefacto según:

- Error con respecto a lo definido en artefactos anteriores y/o en lo pactado con el cliente.
- No concordancia con Normas internacionales que deben ser cumplidas por el artefacto.
- Insatisfacción del cliente con el resultado final de un Elemento de Configuración según lo pactado con anterioridad en el proyecto.

- La resolución de una no conformidad siempre genera una Orden de Trabajo para el proyecto. *[DCS]*

SIG

Sistema de Información Geográfica.

Sistema de Información Geográfica

Es el “conjunto de métodos, herramientas y datos que están diseñados para actuar coordinada y lógicamente para capturar, almacenar, analizar, transformar y presentar toda la información geográfica y de sus atributos con el fin de satisfacer múltiples propósitos. Los SIG son una tecnología que permite gestionar y analizar la información espacial, y que surgió como resultado de la necesidad de disponer rápidamente de información para resolver problemas y contestar a preguntas de modo inmediato”.

Versión

Es un elemento de configuración del software que posee un conjunto definido de características funcionales.