Universidad de las Ciencias Informáticas
Facultad 3

Título: Herramienta para realizar pruebas de volumen a Bases de Datos en PostgreSQL.

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

Autor: Leiser Arias Fernández

Tutoras: Ing. Mairelis Quintero Rios

Ing. Yadira Machado Peña

Co-tutora: Ing. Yailin Fundora Carrasco

La Habana. Junio, 2013

"Casi todo lo que realice será insignificante, pero es muy importante que lo haga."

Mahatma Gandhi

Dedicatoria

DECLARACIÓN DE AUTORÍA

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Firma del Autor

Leiser Arias Fernández

_____

Firma de la Tutora

Ing. Mairelis Quintero Rios

_____

Firma de la Tutora

Ing. Yadira Machado Peña

_____

Firma de la Co-tutora

Ing. Yailin Fundora Carrasco

RESUMEN

El presente trabajo de diploma se basa en la necesidad de desarrollar una aplicación que permita mejorar la realización de las Pruebas de Volumen a las Bases de Datos en el Sistema Gestor de Bases de Datos PostgreSQL. Para el cumplimiento del mismo se realizó un estudio de las tecnologías, herramientas, lenguajes y metodologías. Para guiar el proceso de desarrollo se tomó la metodología SXP (Scrum y Extreme Programming) indicada para proyectos de equipos pequeños y cambiantes, utilizando como Entorno de Desarrollo Integrado la herramienta Netbeans y Java como lenguaje de programación. Con el estudio realizado se logró una aplicación que satisface los requisitos descritos por el cliente, puesto que se realizan las Pruebas de Volumen de forma sencilla: orientadas al usuario y en menor tiempo.

Palabras Claves: Bases de Datos, PostgreSQL, Prueba de Volumen.

Índice

Índice:

Índice

Índice de Ilustraciones

Índice de Tablas

Índice de Tablas

Introducción

INTRODUCCIÓN

Con el avance de las tecnologías y la creación del software en el mundo, la industria de software ha tenido la necesidad de verificar cada vez más la calidad de sus productos. Las pruebas de software son los procesos que permiten verificar y revelar la calidad de un producto de software. Son utilizadas para identificar posibles fallos de implementación, funcionalidad, seguridad, usabilidad, rendimiento a cualquier tipo de software [1].

Hoy en día las industrias del software están matizadas con una fuerte competencia y estas desarrollan a su vez un concepto de calidad del software, donde su objetivo fundamental es satisfacer la expectativa del cliente. En Cuba, la industria del software se ve enmarcada en la necesidad de dominar e introducir en la práctica una eficiente y alta gestión de la calidad para lograr un prestigio y reconocimiento a nivel mundial. Para ello se cuenta con el Centro Nacional de Calidad de Software (CALISOFT), el cual se encarga de gestionar la calidad de los productos de software.

CALISOFT es un centro adscrito a la Agencia de Control y Supervisión del Ministerio de la Informática y las Comunicaciones (MIC) de Cuba. Organización enfocada a contribuir al desarrollo de la Industria Cubana de Software (ICSW), que facilita la implementación de las mejores prácticas en el proceso de desarrollo y/o mantenimiento del software. Responsable de la verificación y validación de productos, procesos y organizaciones según normas nacionales e internacionales y de la asesoría, adiestramiento y formación continua de los especialistas en el país en los temas de Calidad e Ingeniería de Software.

Dentro del Centro existe el Departamento de Pruebas de Software (DPSW), donde se realizan pruebas de liberación, aceptación y pilotos a proyectos nacionales e internacionales. Como parte de los procedimientos establecidos en el departamento se ha definido que los tipos de pruebas serán realizados en función de las características de calidad definidas en la norma ISO/IEC 25000:2005 y teniendo en cuenta las subcaracterísticas que se definen por cada una de estas la que concierne a la presente investigación es la siguiente [2]:

Tabla 1: La característica de calidad: Funcionalidad

Características de calidad	**Tipos de Prueba**

Funcionalidad	**Funcionalidad:** Consisten en la revisión de los requisitos aceptados por el cliente contra las funcionalidades presentes en la aplicación. **Seguridad:** Asegurar que los datos o el sistema solamente es accedido por los actores definidos según niveles de acceso. **Volumen:** Enfocada en verificar las habilidades de los programas para manejar grandes cantidades de datos, tanto como entrada, salida o residente en la BD.

Las pruebas de volumen son las que se encargan de determinar si la Base de Datos de una aplicación soporta un alto número de usuarios realizando peticiones simultáneas y muestran el comportamiento de la aplicación con un determinado volumen de datos. Influyen además en varias características de calidad de software, una de ellas es la fiabilidad, específicamente en la subcaracterística de madurez, la cual se refiere a la capacidad del software para evitar una avería como resultado de haberse producido un fallo del mismo y la tolerancia ante fallos, que es la capacidad del software de mantener un nivel de ejecución específico en caso de fallos de este o de infracción de sus interfaces especificadas.

Otra de las características de la calidad de software que se ve influenciada por este tipo de pruebas es la eficiencia, específicamente en la utilización de recursos, que se traduce como la capacidad del software para usar los recursos apropiados en un plazo de tiempo adecuado cuando el software realiza su función bajo las condiciones declaradas. De modo general las pruebas de volumen también intervienen en la subcaracterística conformidad, siendo esta la capacidad del software para adherirse a las normas que se le apliquen, convenciones, regulaciones, leyes y las prescripciones similares.

Actualmente en el DPSW de CALISOFT, los especialistas utilizan la herramienta JMeter para la ejecución de pruebas de software, específicamente pruebas de carga y stress. En el caso específico de las pruebas de volumen se estuvieron realizando con el JMeter pero se interrumpieron debido a que el JMeter presenta deficiencias. Muestra de ello es que no permite poblar las Bases de Datos que se desean probar lo cual dificulta el trabajo, pues se hace muy engorroso hacerlo de forma manual.
Además son pruebas costosas tanto en personal como en tiempo de máquina porque se necesita de una cantidad de máquinas trabajando al mismo tiempo para hacer las peticiones simultáneas y en el peor de

los casos esas máquinas deben ser operadas por trabajadores, en consecuencia se incrementa el tiempo de máquina que estos deben dedicar a la realización de estas pruebas. Otra de sus debilidades es en cuanto al manejo del JMeter, pues para la ejecución de las pruebas de volumen específicamente se hace difícil, debido a que no provee la documentación necesaria para apoyarse durante el curso de las mismas.

Se ha realizado un estudio de los SGBD más usados en la universidad según la situación de las solicitudes de liberación realizadas hoy en CALISOFT, lo cual arrojó: PostgreSQL con un 70 por ciento, le siguen MySQL y Oracle con un 12 y 10 por ciento respectivamente, mientras que SQLServer, SQLite y MongoDB son usados en bajo por ciento. Existen solicitudes que no requieren de pruebas de volumen, en dependencia del tipo de artefacto, como son: documentación, manuales de usuario, sistemas operativos, multimedias y aplicaciones sin bases de datos.

Todo programa que requiera un buen funcionamiento, referente a indicadores de la base de datos, precisa de la ejecución de pruebas de volumen, para evitar consecuencias tales como: tiempos de respuestas deficientes en la búsqueda de datos y generación de reportes, incremento de transacciones erróneas. Es por ello que se precisa que CALISOFT, como centro rector en la calidad de los productos de software, cuente con los recursos necesarios para brindarle este servicio a los proyectos que utilicen como gestor de base de datos PostgreSQL.

Partiendo de la problemática anteriormente planteada se tiene el siguiente **problema a resolver:**

¿Cómo contribuir a potenciar las Pruebas de Volumen a las Bases de Datos a partir de la necesidad de lograr la calidad de los datos?

A partir del problema a resolver, la investigación tiene como **objeto de estudio:** las pruebas de volumen para la gestión de la calidad del software, tomando como **campo de acción:** el proceso de pruebas de volumen a las Bases de Datos.

El **objetivo general** de este trabajo es: Implementar una herramienta para realizar pruebas de volumen a Bases de Datos en PostgreSQL.

Por esta razón, la **idea a defender** que se plantea es la siguiente: Con la implementación de una herramienta para la mejora de las Pruebas de Volumen a las Bases de Datos para el gestor PostgreSQL, se garantizará potenciar estas pruebas partiendo de la necesidad de lograr la calidad de los datos.

A modo de desglose del objetivo general y en base a la idea a defender, se trazan los siguientes **objetivos específicos:**

- ✓ Analizar los conceptos relacionados con la calidad de software, pruebas de software, Bases de Datos y Pruebas de Volumen.
- ✓ Realizar un estudio de las diferentes herramientas que se utilizan para la ejecución de las pruebas de software.
- ✓ Realizar un estudio de PostgreSQL como Sistema Gestor de Bases de Datos.
- ✓ Analizar y diseñar una herramienta para la realización de las Pruebas de Volumen.
- ✓ Desarrollar una herramienta para la realización de las Pruebas de Volumen.
- ✓ Validar los resultados de la solución propuesta.

Las **tareas de la investigación** que serán realizadas para darle cumplimiento a los objetivos trazados son las siguientes:

- ✓ Investigación sobre los conceptos de calidad de software, pruebas de software, Bases de Datos y Pruebas de Volumen.
- ✓ Estudio de las herramientas de pruebas automatizadas existentes.
- ✓ Determinación de los aspectos negativos y positivos de las herramientas automatizadas determinando las necesidades de las mismas.
- ✓ Definición de la herramienta de modelado para el proceso de software a desarrollar.
- ✓ Definición del lenguaje de programación para la implementación de la herramienta a desarrollar.
- ✓ Generación de los artefactos para el desarrollo de la herramienta.
- ✓ Validación de la solución propuesta.

Métodos Teóricos

Análisis – Síntesis: Para el estudio y análisis de la bibliografía, la formulación del problema y los objetivos, también para la elaboración de las conclusiones y recomendaciones a fin de alcanzar los objetivos de la investigación.

Histórico – Lógico: Permitiendo hacer un estudio del comportamiento actual de la herramienta JMeter proyectando el análisis de la evolución histórica de los fenómenos, con la proyección lógica de su comportamiento futuro.

El presente trabajo presenta la siguiente **estructura por capítulos:**

Capítulo 1: Fundamentación teórica.

En este capítulo se realizará una fundamentación teórica y/o estado del arte de las pruebas de software, teniendo en cuenta su definición e importancia. Se investigará acerca de las herramientas automatizadas para las pruebas de software y por último un análisis de las tecnologías y metodologías seleccionando las más adecuadas para el desarrollo del trabajo.

Capítulo 2: Propuesta de solución.

En este capítulo se realizará la propuesta del diseño de la herramienta a desarrollar, además de otros elementos como los requisitos funcionales para el diseño.

Capítulo 3: Implementación y validación de la solución.

En este capítulo se detallará el proceso de implementación, además de la validación de la herramienta durante la ejecución de las pruebas de volumen como parte del proceso de pruebas de liberación a tres productos de software.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se realiza un estudio de los conceptos de pruebas de software, base de datos, Prueba de Volumen. Seguidamente se desarrolla una comparación de las herramientas que se utilizan para las pruebas de software, donde se selecciona la óptima para darle solución a la problemática descrita.

1.2 Calidad de software

La calidad de un software puede definirse de muchas maneras. Una de las más limitadas, conocida como “calidad pequeña”, define la calidad como la ausencia de defectos [3]. Para evaluarla de esta forma se emplean procedimientos estadísticos a partir de las tendencias de aparición de fallos durante la prueba de software.

Según Pressman, la calidad de software es la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente [4].

La calidad de software es una actividad de protección que se aplica durante todo el proceso de Ingeniería del Software.

La calidad debe ser especificada, planificada, administrada, medida y certificada. Esto implica una visión integral que arroja la comprobación del software, con el fin de lograr un mayor grado de satisfacción y confianza del cliente hacia la organización productora de software. Constituye entonces las pruebas del software, tarea de alta prioridad para las empresas productoras [5].

Al obtener la calidad requerida del software, se logra reducir su número de errores, por lo que se alcanza una mayor fiabilidad en cuanto a las funciones que debe realizar el mismo, se logra a la vez una mayor eficiencia e integridad de los datos, así como flexibilidad y reusabilidad. En la siguiente figura se muestran las características que debe cumplir un software en su etapa final.

Ilustración 1: Calidad de un producto de software (modelo ISO/IEC 25000)

La calidad de software constituye un problema actual que afecta a los productores de software, así como a sus clientes. La demanda de este tipo de producto crece exponencialmente a escala mundial, debido al aumento de la informatización; lamentablemente, los desarrolladores le han brindado poco interés a la calidad de sus productos, pues en varias ocasiones los clientes obtienen el software sin que este haya pasado por la etapa de pruebas.

1.3 Pruebas de software

El único instrumento adecuado para determinar el status de la calidad de un producto de software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requisitos.

Un concepto importante a tomar en consideración es el emitido por Pressman en su edición de 1998, que plantea lo siguiente: La prueba del software es un elemento crítico para la garantía de calidad del software y representa una revisión de las especificaciones, del diseño y de la codificación [6].

Se puede concluir que la prueba de software es un proceso en el cual el sistema se ejecuta bajo condiciones específicas para demostrar si este software cumple o no, con la madurez necesaria para ser implantado.

La norma ISO/IEC 25000 dentro de sus características define como tipos de pruebas de seguridad, de funcionalidad, de rendimiento, de volumen, entre otras; que garantizan la calidad en las Bases de Datos como elemento fundamental de los proyectos de gestión.

1.4 Base de Datos

Base de datos es un conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. Es decir, que una base de datos puede considerarse una colección de datos variables en el tiempo.

El software que permite la actualización y/o utilización de los datos almacenados de una o varias bases de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina Sistema de Gestión de Bases de Datos (SGBD).

Los Sistemas de Bases de Datos se diseñan e implementan para gestionar grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información. Una de las pruebas que se le hacen a la base de datos es la Prueba de Volumen [7].

1.5 Pruebas de Volumen

La Prueba de Volumen sujeta al elemento de prueba a grandes cantidades de datos, para determinar si se alcanzan los límites que hacen fallar al software. También identifica la carga máxima o volumen que el elemento de prueba puede manejar por un período dado. Por ejemplo, si el elemento de prueba está procesando un sistema de expedientes de la base de datos para generar un informe, una Prueba de Volumen utilizaría una base de datos grande para la prueba y comprobaría que el software se comportó normalmente y que produjo el informe de forma correcta. Además, estas pruebas determinan si el sistema puede trabajar con grandes cantidades de datos, indicando cuándo los límites son alcanzados, lo que causaría que el software falle [8].

Son diseñadas específicamente para aplicaciones que acceden a repositorios de datos, es importante saber cómo la aplicación opera no solamente cuando es nueva, las bases de datos se convierten en

pobladas cuando se está muy cerca de exceder el máximo de capacidad, los problemas incluyen tiempos de respuesta deficientes en la búsqueda de datos y generación de reportes, incremento de transacciones erróneas [9].

Las pruebas de volumen también identifican las cargas continuas o el volumen que el sistema puede manejar por un tiempo dado.

1.5.1 Objetivos

El objetivo de las pruebas de volumen consiste en verificar que la aplicación funcione exitosamente bajo los siguientes escenarios de gran volumen:

- ✓ El máximo número de clientes conectados, todos ejecutando la misma funcionalidad de negocio con el peor caso (de desempeño) por un período largo de tiempo.
- ✓ El tamaño máximo de la BD ha sido alcanzado y múltiples transacciones de consultas y reportes son ejecutados simultáneamente.

1.5.2 Técnicas

- ✓ Se utilizan los *scripts* diseñados para las pruebas de desempeño.
- ✓ Deben usarse múltiples clientes, ya sea corriendo las mismas pruebas o pruebas complementarias para producir el peor caso de volumen por un período extendido.
- ✓ Se utiliza un tamaño máximo de base de datos. (Actual, escalados o con datos representativos) y múltiples clientes para correr consultas simultáneamente para periodos extendidos [10].

1.6 Herramientas para la realización de pruebas de software

En el mundo de la informática existen diversas herramientas para realizarle pruebas al software para verificar su correcto funcionamiento y asegurar la aceptación del cliente, algunas de estas son:

1.6.1. Webserver Stress Tool

Es una potente herramienta de pruebas de rendimiento que permite encontrar problemas críticos en aplicaciones y servidores Web (HTTP-cliente/servidor).

Permite simular un gran número de usuarios accediendo a una aplicación Web vía HTTP/HTTPS. La versión de pago permite simular hasta 10.000 usuarios, mientras que la gratuita sólo permite simular un

único usuario. El precio de la licencia no es muy elevado (200€), barata en comparación con otras herramientas de este tipo que están en el mercado.

Esta herramienta permite realizar pruebas de rendimiento, stress, carga y disponibilidad. Pero no realiza las pruebas de volumen [11].

1.6.2. AdventNet QEngine

Es una herramienta automática para hacer pruebas de funcionalidad, rendimiento, carga, tráfico de datos, servicios Web, control del rendimiento de servidores, pruebas de regresión, para ejecutar pruebas desde la línea de comandos para todo tipo de pruebas. QEngine funciona tanto en plataformas Windows como en Linux. Permite grabar y reproducir los exploradores Microsoft Internet Explorer, Firefox. Permite a múltiples miembros de equipos, el acceso, creación, modificación y ejecución de conjuntos de pruebas desde cualquier lugar del mundo. Pero no posee la funcionalidad de realizar pruebas de volumen [12].

1.6.3. Jameleon Test Suite

Es un marco de pruebas automatizadas que pueden ser fácilmente utilizados tanto por los usuarios técnicos como los usuarios no técnicos. Uno de los conceptos principales detrás de Jameleon es crear un grupo de palabras claves o etiquetas que representan las diferentes pantallas de una aplicación. Toda la lógica necesaria para automatizar cada pantalla en particular se puede definir en Java y se asigna a estas palabras claves. Las palabras claves pueden ser organizadas con diferentes conjuntos de datos para formar scripts de prueba sin necesidad de un conocimiento profundo de cómo funciona la aplicación. Los scripts de prueba se utilizan para automatizar las pruebas y para generar la documentación del manual de caso de prueba. Esta herramienta no realiza las pruebas de volumen [13].

1.6.4. Nessus

Nessus es un escáner de vulnerabilidades que funciona mediante un proceso de alta velocidad por el que encuentra los datos sensibles y trabaja con la auditoría de configuraciones y el perfil activo. Es muy importante destacar que este programa es de gran utilidad para los dispositivos de uso personal pero también para las grandes empresas que se manejan con equipos conectados en red. Es un sistema de respuesta rápida que permite realizar exploraciones y análisis ad-hoc. Además, es fundamental tener en cuenta que utilizándolo en conjunto con el SecurityCenter nos permite que las recomendaciones y los resultados de vulnerabilidad sean enviados a los responsables para que ellos puedan iniciar la

recuperación mediante un minucioso rastreo o la audición de los parches de seguridad. La herramienta Nessus no realiza pruebas de volumen [14].

1.6.5. LoadRunner

LoadRunner es una herramienta de pruebas de rendimiento proveída por HP que permite probar y analizar el comportamiento de una aplicación cuando esta es usada en condiciones normales, de stress o de forma prolongada.

Otra forma de describirlo es que LoadRunner es una herramienta de rendimiento (performance), que permite simular diferentes comportamientos “cotidianos” en un sistema.

LoadRunner crea archivos de registro mientras trabaja. En estos archivos de registro, el usuario puede encontrar información sobre cualquier problema que se haya producido. Puede definir rastreos para:

- El motor
- El gestor de nivel medio
- La GUI de Web
- El módulo RIM (RDBMS Interface Module)

Los archivos de registro para grabación y reproducción sólo se crean en las máquinas donde se han iniciado recopilaciones de datos de la simulación de grabación y reproducción. Los archivos de registro que se crean son tapm_playback_session.log y tapm_playback_driver.log. Estos pueden volverse a utilizar para monitorizar la aplicación una vez terminada su implantación. Pero el LoadRunner no realiza pruebas de volumen [15].

1.6.6. Centro de aplicaciones de prueba (Application Center Test)

Centro de aplicaciones de prueba (ACT según sus siglas en inglés) es una herramienta de Microsoft incluida en Visual Studio.NET.

Diseñada especialmente para desarrollar pruebas de carga y estrés, permitiendo obtener toda la información necesaria para detectar problemas de rendimiento y escalabilidad en las aplicaciones. Además permite realizar pruebas funcionales gracias a las pruebas dinámicas, su funcionamiento reside en simular un gran número de usuarios abriendo múltiples conexiones al servidor y enviando peticiones HTTP.

Permite crear pruebas estáticas al importar archivos de registro de *Internet Information Server* (IIS) y soporta la creación de pruebas al registrar una sesión del explorador de Internet Explorer, archivos que pueden ser modificados utilizando diferentes comandos.

El objetivo principal de Application Center test son las pruebas de nivel de carga de larga duración y carga alta, las pruebas dinámicas programables también pueden ser útiles en pruebas funcionales [16].
Es un software privativo y de igual manera no realizar las pruebas de volumen.

1.6.7. Selenium IDE

Selenium es una herramienta para realización de pruebas de aplicaciones Web, está orientado a la ejecución de pruebas funcionales a nivel de usuario directamente desde el navegador.
Esta herramienta es muy útil para el desarrollo Web donde se tienen que realizar cúmulos de pruebas cada vez que se saca una versión nueva o se realizan modificaciones en un portal. Selenium automatiza el proceso de pruebas y permite ejecutar un conjunto de pruebas completo si es necesario o pruebas particulares [17].
Selenium es una herramienta de Software Libre. Las pruebas de Selenium se ejecutan directamente en un navegador como se mencionó antes y facilitan las pruebas de compatibilidad en navegadores, también como pruebas funcionales de aceptación de aplicaciones Web.

- ✓ Selenium IDE es un *plugin* para Firefox que permite grabar y ejecutar scripts directamente desde tu navegador.
- ✓ Selenium RC es una biblioteca (*library*) y servidor escrito en lenguaje Java que permite ejecutar scripts en forma local o remota a través de comandos.
- ✓ Selenium Grids permite coordinar múltiples servidores Selenium para así poder ejecutar scripts en múltiples plataformas y equipos al mismo tiempo. A pesar que permite un conjunto de pruebas las pruebas de volumen no están dentro de dicho conjunto [18].

1.6.8. JMeter

Originalmente el Apache JMeter fue diseñado para realizar pruebas de estrés sobre aplicaciones Web (pruebas Web clásicas). Sin embargo, hoy en día su arquitectura ha evolucionado, ahora no sólo puede llevar a cabo pruebas en componentes típicos de Internet (HTTP), sino también puede realizar pruebas

sobre Bases de Datos, *scripts Perl*, *servlets*, objetos java, servidores FTP y prácticamente cualquier medio de los que se pueden encontrar en la red.

El Apache JMeter está diseñado para desarrollar diferentes tipos de test; permitiendo diseñar tanto sencillas pruebas que soliciten simples páginas Web, como complejas secuencias de requisiciones que permitan evaluar el comportamiento de una aplicación o la capacidad de carga máxima que pueda tener una aplicación en un servidor (pudiendo llegar a saturar el servidor).

JMeter también permite la ejecución de pruebas distribuidas entre distintos ordenadores, para realizar pruebas de rendimiento.

El Apache JMeter incluye una interfaz gráfica de usuario que facilita el diseño de las pruebas. Esta interfaz gráfica permite guardar y alterar tanto las pruebas desarrolladas como los componentes que lo integran. Gracias a esto se pueden reutilizar las pruebas o módulos de las mismas en el desarrollo de nuevas pruebas.

Además de las funcionalidades de prueba antes mencionadas, el Apache JMeter también ofrece la posibilidad de activar un Proxy Web. Por lo tanto, se puede grabar la navegación de un usuario para posteriormente usarla en la generación de una prueba.

JMeter es una herramienta que sirve para realizar pruebas de rendimiento y de funcionalidad sobre aplicaciones tipo cliente/servidor. Por tanto será útil durante la fase de desarrollo de las aplicaciones para la realización de pruebas funcionales y de regresión. Además será de gran utilidad durante la fase de pruebas para la realización de pruebas de carga y de volumen de las aplicaciones [19].

1.7 Comparación de las herramientas

Tabla 2: Comparación de las herramientas

Herramientas	Propietaria	Prueba de Volumen	Código abierto
Webserver Stress Tool	Si	No	No
AdventNet QEngine	Si	No	No
Jameleon Test Suite	Si	No	Si
Nessus	Si	No	No
LoadRunner	Si	No	No
Application Center Test	Si	No	No
Selenium	No	No	Si
JMeter	No	Si	Si

La herramienta JMeter, se destaca por su versatilidad y estabilidad dentro de las herramientas de libre distribución. Presenta una estructura en árbol que le da potencia, permitiendo que sea la imaginación de quien la use la que ponga los límites a la hora de diseñar el plan de prueba, brindando mayor cantidad de variantes para recoger los resultados obtenidos que el resto de las herramientas de libre distribución, lo que permite hacer un análisis exhaustivo de las pruebas realizadas.

Otro aspecto a favor de esta herramienta es su desarrollo en el marco del software libre, política que apoya el país. Además, por las características que presenta la herramienta, existe un por ciento superior de confianza en la utilización de la misma ya que se puede contar con todo el código que genera.

A pesar de todas las ventajas que ofrece el JMeter, es necesario aclarar que presenta deficiencias con todos los indicadores necesarios a la hora de realizar las pruebas de volumen, debido a que no puebla las Bases de Datos, elemento esencial para la realización de una mejor prueba. Puesto que obviamente, la Prueba de Volumen es muy costosa, tanto en tiempo de máquina como de personal, porque debe contar con varias máquinas trabajando distribuidas y en el peor de los casos deben tener una gran cantidad de usuarios realizando peticiones a la base de datos concurrentemente lo que incrementa el tiempo de máquina, además el manejo y trabajo con esta herramienta resulta ser muy engorroso, ya que no cuenta con una documentación que ayude a realizar las Pruebas de Volumen.

Sin embargo, el software que desee tener un buen funcionamiento de su base de datos, debe ser expuesto al menos a algunas Pruebas de Volumen, debido a que la no realización de estas pruebas podrían derivar en diversas consecuencias tales como: tiempos de respuestas deficientes en la búsqueda de datos y generación de reportes e incremento de transacciones erróneas. De este modo surge la necesidad de realizar una herramienta que ayude agilizar y mejorar el proceso de las Pruebas de Volumen en el Departamento de Pruebas de Software de Calisoft.

1.8 Metodología de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. En el que se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla [20].

1.8.1 SXP (SCRUM y XP)

El grupo de Ingeniería y de Análisis de Sistemas, adscrito al Grupo Unicornios en la facultad 10 de la UCI, ha investigado y desarrollado una metodología a la medida que permita soportar desarrollos ágiles de aplicaciones como SISCLON, esta metodología es un hibrido entre las metodologías Scrum y XP el cual fue avalada por el Departamento de Calidad de la UCI.

SXP es una metodología ágil conformada por sus referentes internacionales SCRUM y XP (Extreme Programming), diseñada pensando en proyectos de software libre. Brinda una estrategia tecnológica a partir

de procedimientos ágiles, lo que permite el aumento de la creatividad y responsabilidad del personal implicado. Se encuentra conformada por SCRUM que permite gestionar un equipo de manera que trabaje de forma eficiente y por XP para el desarrollo el cual consiste en una programación rápida con el usuario final como parte del equipo.
Consta de 4 fases principales:

Planificación-Definición: Establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.

Desarrollo: Realiza la implementación del sistema hasta que esté listo para ser entregado.

Entrega: Realiza la entrega del software y su documentación, generándose aquellos documentos que son imprescindibles para el entrenamiento y entendimiento del producto.

Mantenimiento: Realizan las actividades relacionadas con el soporte del software y se generan los documentos relacionados con los cambios que puedan ocurrir en el mismo.

De cada una de estas fases se realizan numerosas actividades tales como el levantamiento de requisitos, la priorización de la Lista de Reserva del Producto, definición de las historias de usuario (HU), Diseño, Implementación, Pruebas, de donde se generan artefactos para documentar todo el proceso. Las entregas son frecuentes, y existe una refactorización continua, que permite mejorar el diseño cada vez que se añade una nueva funcionalidad.

Se selecciona SXP ya que se cuenta con un solo desarrollador, los clientes son trabajadores de Calisoft por lo que se puede realizar continuos cambios en los requisitos y se orienta a una entrega rápida de resultados y una alta flexibilidad. Ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro, permitiendo además seguir de forma clara el avance de las tareas a realizar, de forma que los jefes pueden ver día a día cómo progresa el trabajo. Además es un proyecto donde consta de pocos procesos por lo que será un producto pequeño y se cuanta con poco tiempo para su realización.

1.9 Lenguajes de desarrollo y modelado

1.9.1 Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML) prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos y describe la semántica esencial de lo que estos diagramas y símbolos significan. Existen muchas notaciones y métodos usados para el diseño orientado a objetos, ahora el personal sólo tienen que aprender una única notación. UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.

Los objetivos de UML son muchos pero se centra en expresar de forma gráfica el sistema que se desee implementar de una forma entendible y logra especificar cada una de sus características. A partir de los modelos especificados se logran construir los sistemas diseñados y el diseño se reutilizaría como parte de la documentación del producto. Aunque UML es bastante independiente del proceso de desarrollo que se siga, los mismos creadores de UML han propuesto su propia metodología de desarrollo, denominada el Proceso Unificado de Desarrollo [21].

Los aspectos que definen este Proceso Unificado son tres: es iterativo e incremental, dirigido por casos de uso y centrado en la arquitectura [22].

Dirigido por casos de uso: Basándose en los casos de uso, los desarrolladores crean una serie de modelos de diseño e implementación que los llevan a cabo. Además, estos modelos se validan para que sean conformes a los casos de uso. Finalmente, los casos de uso también sirven para realizar las pruebas sobre los componentes desarrollados.

Centrados en la arquitectura: En la arquitectura de la construcción, antes de construir un edificio éste se contempla desde varios puntos de vista: estructura, condiciones eléctricas y fontanería. Cada uno de estos aspectos está representado por un gráfico con su notación correspondiente. Siguiendo este ejemplo, el concepto de arquitectura de software incluye los aspectos estáticos y dinámicos más significativos del sistema.

Iterativo e incremental: Dividir un proceso en varias fases y ciclos de vida en los que se realizan varios recorridos por todas las fases. Cada recorrido por las fases se denomina iteración en el proyecto en la que se realizan varios tipos de trabajos (denominados flujos). Además, cada iteración parte de la anterior incrementando o revisando la funcionalidad implementada. Se suele denominar proceso.

Para cumplir con los objetivos trazados y darle solución a la problemática anteriormente descrita, se define como lenguaje de modelado UML, centrándose las características relevantes anteriormente descritas. Permite desarrollar el trabajo fácil y organizado, brindando muchísimas comodidades en el uso de la programación orientada a objetos y permite un fácil entendimiento entre el equipo de desarrollo y el cliente.

1.9.2 Lenguaje de programación

Un lenguaje de programación es aquel elemento dentro de la informática que nos permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes [23].

1.9.2.1 Java

Java es un lenguaje de programación creado por Sun Microsystems en 1995. Tecnología que permite el uso de programas tan importantes como herramientas, juegos y aplicaciones de negocio. Java es rápido, seguro y fiable, se ejecuta en más de 850 millones de ordenadores personales en todo el mundo y en miles de millones de dispositivos, como dispositivos móviles y equipos de televisión. Existe un gran número de aplicaciones y sitios Web que no funcionan al menos que java este instalado [24].
Java ha sido probado, ajustado, ampliado y probado por toda una comunidad. Más de nueve millones de los desarrolladores de Java la convierte en la comunidad de desarrollo más grande y más activa del planeta. Con su versatilidad, eficacia y portabilidad, Java se ha convertido en incomparable para los desarrolladores, ya que les permite:

- ✓ Escribir software en una plataforma y ejecutarla virtualmente en otra.
- ✓ Crear programas que se puedan ejecutar en un explorador y acceder a servicios Web disponibles.
- ✓ Desarrollar aplicaciones de servidor para foros en línea, almacenes, encuestas, procesamiento de formularios HTML y mucho más.

- ✓ Combinar aplicaciones o servicios que utilizan el lenguaje Java para crear aplicaciones o servicios con un gran nivel de personalización.
- ✓ Escribir aplicaciones potentes y eficaces para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier otro dispositivo con un latido digital [25].

JMeter es una herramienta desarrollada en java, por lo que se decide utilizar esta tecnología para evitar posibles errores o daños en la integración del desarrollo con esta. Además Java es una conocida tecnología multiplataforma por lo que brinda la oportunidad de que el desarrollo pueda ser utilizado en Windows o Linux.

1.10 Herramientas de Desarrollo

1.10.1 Visual Paradigm 5.0

Visual Paradigm para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación [26].

Para el modelado de cada uno de los procesos y artefactos que guiarán el trabajo del equipo de construcción del software se empleará Visual Paradigm para UML en su versión 8.0. Herramienta que utiliza UML como lenguaje de modelado, permite agilizar el trabajo ya que entre sus funcionalidades permite generar todos los diagramas de clases, código fuente desde diagramas y documentación en diferentes formatos; previendo que cuenta además con gran cantidad de documentación y demostraciones interactivas. En la UCI se cuenta con la licencia para su uso y tiene la característica de ser multiplataforma, elemento de vital importancia debido a las políticas de migración definidas en la Universidad.

1.10.2 Entorno de desarrollo integrado

Un Entorno de Desarrollo Integrado, traducido del inglés Integrated Development Environment (IDE) es un programa informático compuesto por un conjunto de herramientas de programación. Este puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

Un IDE puede denominarse como un entorno de programación que ha sido tratado como un programa o aplicación. Esto significa que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc. Es posible que un mismo desarrollo integrado funcione con varios lenguajes de programación, como por ejemplo el NetBeans que tiene soporte de varios lenguajes [27].

1.10.2.1 NetBeans 6.9.1

El IDE NetBeans es un reconocido entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. El proyecto NetBeans está formado por un IDE de código abierto y una plataforma de aplicación que permite a los desarrolladores crear con rapidez aplicaciones Web, empresariales, de escritorio y móviles utilizando la plataforma Java, así como JavaFX, PHP, JavaScript y Ajax, Ruby y Ruby on Rails, Groovy and Grails y C/C++.

El proyecto de NetBeans está apoyado por una comunidad de desarrolladores dinámica y ofrece documentación y recursos de formación exhaustivos, así como una amplia selección de complementos de terceros.

El IDE NetBeans 6.9.1 introduce JavaFX Composer, una herramienta de diseño para la creación de aplicaciones gráficas JavaFX, parecido al constructor de aplicaciones gráficas Swing para aplicaciones Java SE. Otras notoriedades incluyen la interoperabilidad OSGi para aplicaciones de plataforma NetBeans y la compatibilidad para desarrollar paquetes OSGi con Maven; compatibilidad para el SDK de JavaFX 1.3.1, Famework Zend PHP y RoR (Ruby on Rails) 3.0; así como mejoras en el editor Java, Depurador Java, seguimiento de incidencias y muchas más [28].

Vistas las ventajas que brinda el IDE NetBeans 6.9.1, es el elegido para la implementación de la herramienta a desarrollar.

1.10.3 Sistema Gestor de Bases de Datos

Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas, procedimientos y lenguajes que nos proporcionan las herramientas necesarias para trabajar con una base de datos. Incorpora una serie de funciones que nos permita definir los registros, sus campos, sus relaciones, insertar, suprimir, modificar y consultar los datos [29].

1.10.3.1 PostgresSQL 9.1

PostgreSQL es un sistema de gestión de Bases de Datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el Sistema de Gestión de Bases de Datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarles a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

Sus características técnicas la hacen una de las Bases de Datos más potentes y robustas del mercado. Su desarrollo comenzó hace más de 16 años y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. Además contiene:

- ✓ Numerosos tipos de datos y posibilidad de definir nuevos tipos. Además de los tipos estándares en cualquier base de datos, tenemos disponibles, entre otros, tipos geométricos, de direcciones de red, de cadenas binarias, UUID, XML, matrices, etc.
- ✓ Soporta el almacenamiento de objetos binarios grandes (gráficos, videos, sonido, ...)
- ✓ APIs para programar en C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, PHP, Lisp, Scheme, Qt [30].

1.11 Conclusiones

En este capítulo se realizó un análisis sobre todos los conceptos estrechamente relacionados a la calidad de software.

- ✓ La calidad de software misma como herramienta rectora para la implementación de una aplicación adoptando el concepto dado por el modelo ISO/IEC 25000 para el desarrollo del trabajo de diploma.
- ✓ Se trató el tema de las pruebas de software, entre las que se encontraban las pruebas de volumen; estratégicamente importantes para el correcto funcionamiento de la Base de datos de un sistema y

la necesidad existente de una herramienta capaz de desarrollar estas pruebas con la calidad requerida.

- ✓ Se estudiaron herramientas para la realización de pruebas de software tales como: NESSUS, LoadRunner y JMeter que no satisfacen la necesidad existente en el departamento de pruebas de Calisoft, respecto a ese tema.
- ✓ Se hizo un estudio sobre los sistemas gestores de Base de Datos para saber cuál era el más usado y las estadísticas arrojaron un aplastante 70 % de uso del gestor PostgreSQL.
- ✓ Otro tema que se abordó en el capítulo fue el estudio de una metodología óptima para el desarrollo de la herramienta. Por sus características se decidió utilizar la metodología SXP, pues es ágil, está pensada para proyectos de software libre y pequeños como el caso de esta aplicación.

De esta manera se cumple con los objetivos esperados por el capítulo abriendo paso al diseño de la herramienta con las pautas y las necesidades más esclarecidas.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

2.1 Introducción

El presente capítulo se enfocará en los detalles de la solución que se desea implementar, teniendo en cuenta las necesidades del centro CALISOFT. Incluye un análisis de la situación en el centro CALISOFT donde nace la necesidad de desarrollar una herramienta para la ejecución de pruebas de volumen a los software. Se describirán artefactos tales como: la Lista de Reserva del Producto y las Historias de Usuarios.

2.2 Descripción de la situación actual

Actualmente en el centro CALISOFT no se están realizando las pruebas de volumen porque no cuenta con todos los indicadores necesarios a la hora de realizar estas pruebas, debido a que el JMeter no puebla las Bases de Datos, elemento esencial para la realización de una mejor prueba. Puesto que obviamente, la Prueba de Volumen es una prueba costosa, tanto en tiempo de máquina como de personal, se debe tratar de no exceder los límites; además el manejo y trabajo con esta herramienta resulta ser muy engorroso. Para ello se quiere realizar una herramienta que ayude a poblar la base de datos para agilizar y mejorar el proceso de prueba. Como una necesidad real de vital importancia se tiene la realización de una herramienta, la misma poblará la base de datos y realizará las Pruebas de Volumen.

2.3 Modelo del dominio

El modelo de dominio permite capturar y expresar el entendimiento del diseño de un sistema en un área de bajo análisis. Debido a la poca estructuración de los procesos de negocio que provee el desarrollo de la herramienta se utilizará dicho modelo, el que permitirá mostrar de manera visual los principales conceptos que se manejan, ayudando a los usuarios a utilizar un vocabulario común, lo más real posible a las características físicas que posee el mismo, para poder entender el contexto en que se desarrolla el sistema.

En la figura 2 se presenta dicho modelo, cuyo objetivo principal está centrado en lograr un entendimiento del contexto en que se sitúa la aplicación, describir el funcionamiento de la herramienta mediante conceptos y relaciones.

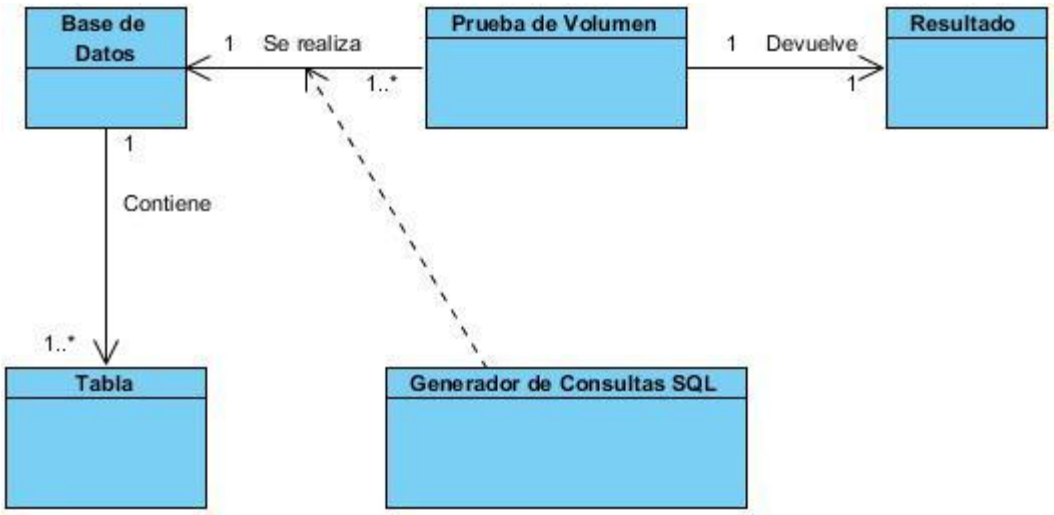


Ilustración 2: Diagrama de Modelo del Dominio

Descripción de Clases:

- ✓ **Base_de_Datos:** Se encarga de emular una base de datos.
- ✓ **Tabla:** Se encarga de emular una tabla.
- ✓ **Generador de Consultas SQL:** Se encarga de crear y eliminar consultas SQL.
- ✓ **Prueba de Volumen:** Se encarga de realizarle peticiones a la base de datos mediante las consultas SQL.
- ✓ **Resultado:** Se encarga de mostrar el resultado de la prueba.

2.4 Requisitos del software

Los requisitos del software forman parte de la documentación asociada al software que se está desarrollando, por tanto se deben definir correctamente todos los requisitos, pero no más de los necesarios. Esta documentación no debería describir ningún detalle de diseño, modo de implementación o gestión del proyecto, ya que los requisitos se deben describir de forma que el usuario pueda entenderlos. Al mismo tiempo, se da una mayor flexibilidad a los desarrolladores para la implementación [31].

2.4.1. Lista de Reserva del Producto

Tabla 3: Lista de Reserva del Producto (LRP)

Código	Descripción del requisito funcional	Prioridad
RF1	Realizar la conexión a la base de datos.	Muy alta
RF2	Determinar cuáles tablas poblar de la base de datos.	Alta
RF3	Gestionar las consultas SQL que se le pedirán a la base de datos.	Alta
RF3.1	Adicionar consulta SQL.	Alta
RF3.2	Eliminar consulta SQL.	Alta
RF3.3	Listar consultas SQL.	Alta
RF4	Configurar el modo de ejecución de la prueba.	Alta
RF5	Poblar la base de datos para el Sistema Gestor de Bases de Datos PostgreSQL	Muy Alta
RF6	Generar reportes de los resultados de la prueba.	Alta
RF6.1	Generar reporte en formato PDF con los resultados de las pruebas.	Alta
RF7	Cargar una prueba.	Media
RF8	Guardar la prueba.	Media
RNF (Requisitos no funcionales)		
	Usabilidad	
RNF1	Facilidad de uso por parte de los usuarios o probadores: el sistema debe presentar una interfaz amigable que permita una fácil interacción con él y llegar de manera rápida y efectiva a configurar. Además la interfaz debe permitir un manejo cómodo que posibilite a los usuarios sin experiencia una rápida adaptación al mismo.	

	Especificación de la terminología utilizada
RNF2	El sistema debe adaptarse al lenguaje y términos utilizados por los probadores con vista a una mayor comprensión por parte del cliente de la herramienta de trabajo.
	Software
RNF3	Para ejecutar y correr el programa solo se necesita contar con una computadora con la Máquina Virtual de Java instalada.
	Requisitos del diseño e implementación
RNF4	Como característica específica se utilizarán estas herramientas en las fases de construcción del software: ✓ Se debe implementar en el lenguaje Java. ✓ Para el análisis y el diseño se usará el lenguaje de modelación UML y como herramienta para llevar a cabo el modelado Visual Paradigm. ✓ Para la implementación de la herramienta se utilizará el IDE NetBeans, específicamente Swing, una herramienta para el desarrollo de aplicaciones de escritorio.

2.5 Descripción de las historias de los usuarios

Se realiza la descripción de las historias de los usuarios con sus tareas ingenieriles y su responsable asignado para el cumplimiento de la misma.

Se muestran a continuación las historias de los usuarios más importantes del proceso:

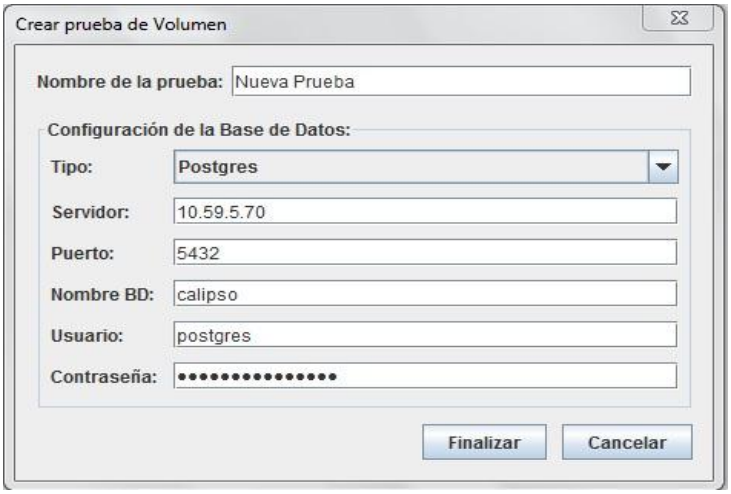
Historias de los Usuarios (HU)

HU1- Realizar la conexión a la Base de Datos

Tabla 4: HU1- Realizar la conexión a la Base de Datos

Historia de Usuario	
Código: HU1	**Nombre Historia de Usuario:** Realizar la conexión a la Base de Datos
Modificación de Historia de Usuario Número: 1	
Referencia: RF1	
Programador: Leiser Arias Fernández	**Iteración Asignada:** Primera
Prioridad : Alta	**Riesgo en Desarrollo:** Alta
Descripción: Una vez que se ejecuta la aplicación el sistema muestra una ventana (Ver Anexo 1). Para gestionar la conexión a la base de datos se va a la opción “Nuevo”. El sistema muestra una ventana que contiene los campos siguientes: Nombre de la prueba, Tipo de gestor de base de datos, Servidor, Puerto, Usuario y Contraseña. Luego de entrar todos los datos correctos el sistema muestra las tablas de la base de datos solicitada. **Rol:** Probador	
Observaciones: 1. En caso que la conexión a la base de datos falle, el sistema debe mostrar un mensaje “Existen errores con sus parámetros de conexión. Por favor verifíquelos”. Los fallos parten de datos entrados incorrectamente. 2. Si hay campos vacíos la opción que permite iniciar la conexión no se habilita.	

Prototipo de interface:



HU2- Determinar cuáles tablas poblar de la Base de Datos

Tabla 5: HU2- Determinar cuáles tablas poblar de la Base de Datos

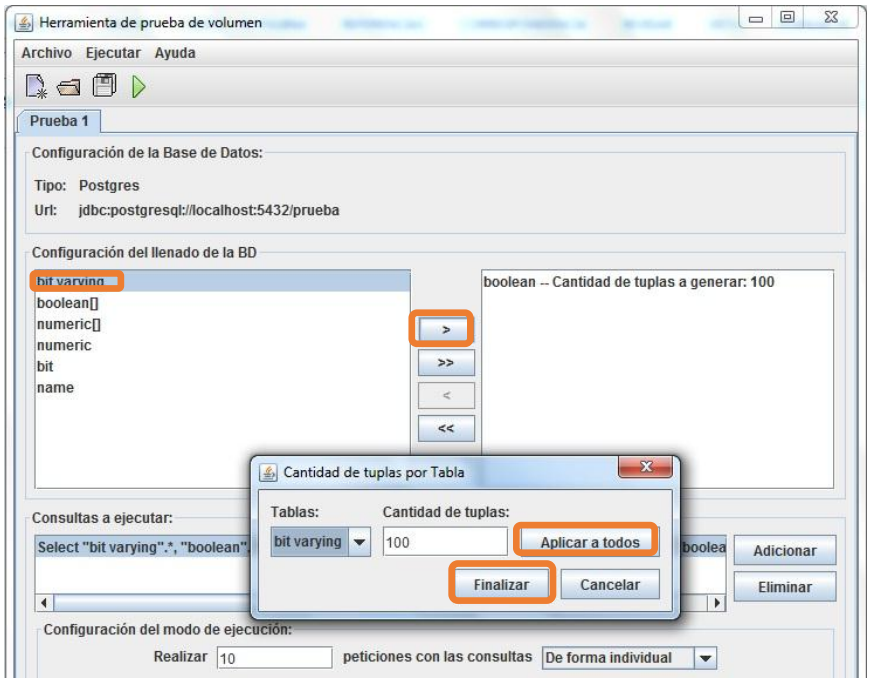
Historia de Usuario	
Código: HU2	**Nombre Historia de Usuario:** Determinar cuáles tablas poblar de la Base de Datos
Modificación de Historia de Usuario Número: 1	
Referencia: RF2	
Programador: Leiser Arias Fernández	**Iteración Asignada:** Primera
Prioridad: Alta	**Riesgo en Desarrollo:** Alta
Descripción: Para el poblado de la base de datos se selecciona la tabla de la base de datos y se presiona el botón “>”. El sistema muestra una ventana donde permite especificar la cantidad de tuplas a poblar. Se selecciona el botón “Finalizar” y se muestra en la pantalla inicial el panel derecho con la tabla seleccionada y la cantidad de tuplas a generar. **Rol:** Probador	

Observaciones:

1. Si no se selecciona ninguna tabla todos los botones están deshabilitados menos el botón “>>” que permite pasar todas las tablas para el panel derecho con las cantidades de tuplas deseadas o aplicarle la misma cantidad a todas con la opción “Aplicar a todos”.
2. Una vez que estén las tablas listas para poblar en el panel derecho, se activa el botón “<<” y el botón “<” para dar la posibilidad de regresar una tabla específica o todas.

El máximo de tuplas a poblar por cada tabla es 10000 y el mínimo es 1.

Prototipo de interface:



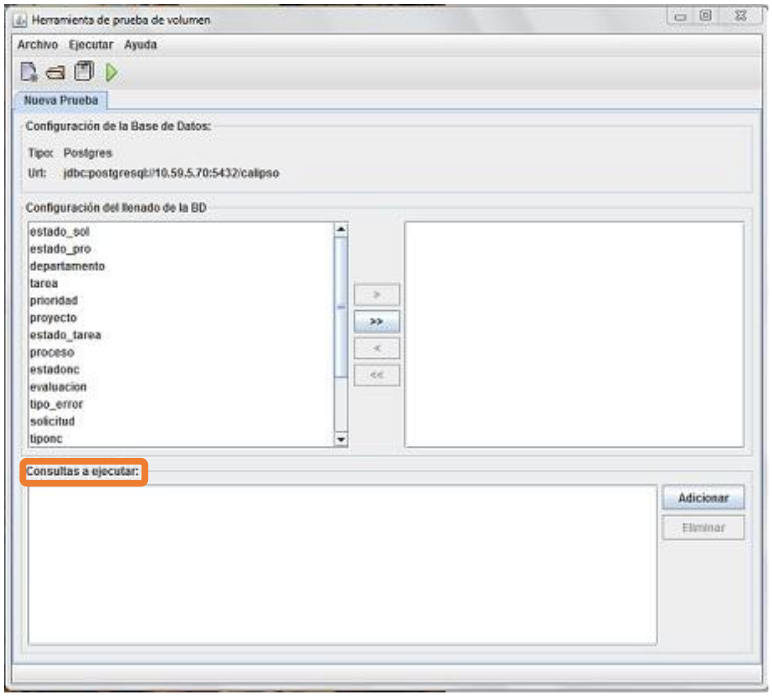
HU3- Gestionar las consultas SQL que se le pedirán a la Base de Datos

Tabla 6: HU3- Gestionar las consultas SQL que se le pedirán a la Base de Datos

Historia de Usuario

Código: HU3	**Nombre Historia de Usuario:** Gestionar las consultas SQL que se le pedirán a la Base de Datos
Modificación de Historia de Usuario Número: 1	
Referencia: RF3	
Programador: Leiser Arias Fernández	**Iteración Asignada:** Primera
Prioridad: Alta	**Riesgo en Desarrollo:** Alta
Descripción: El sistema muestra la pantalla que permite gestionar las consultas SQL. Brinda la posibilidad de Adicionar una nueva consulta (Ver HU4), Eliminar las consultas (Ver HU5) y Listar consultas (Ver HU6). **Rol:** Probador	
Observaciones: N/A.	

Prototipo de interface:



HU4- Adicionar consulta SQL

Tabla 7: HU4- Adicionar consulta SQL

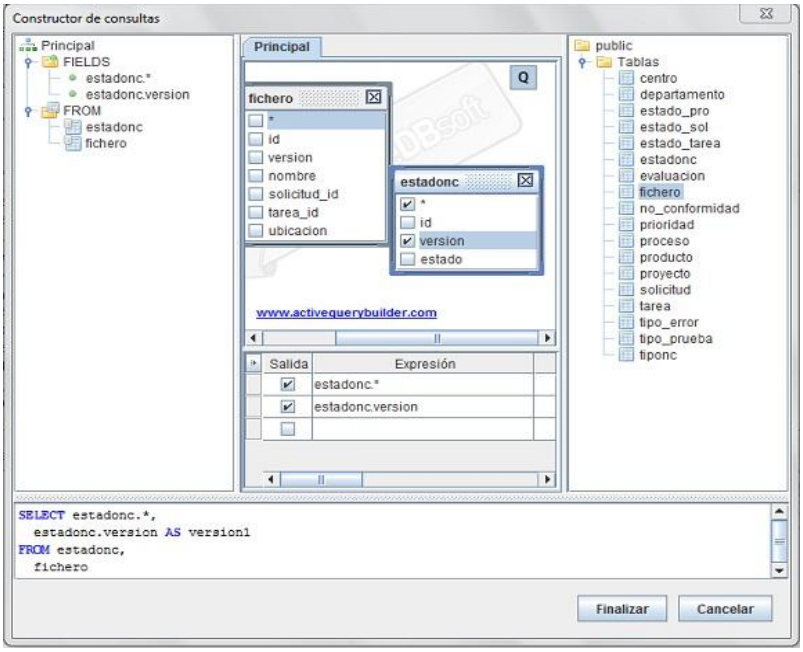
Historia de Usuario	
Código: HU4	**Nombre Historia de Usuario:** Adicionar consulta SQL
Modificación de Historia de Usuario Número: 1	
Referencia: RF3.1	
Programador: Leiser Arias Fernández	**Iteración Asignada:** Primera
Prioridad: Alta	**Riesgo en Desarrollo:** Baja
Descripción: Una vez presionado el botón “Adicionar” el sistema muestra en la parte	

derecha de una ventana, las tablas de la base de datos para realizar las consultas. Una vez seleccionada la tabla esta muestra sus atributos para decidir cuales pedir en la consulta. El sistema muestra en el panel inferior como se va realizando la consulta SQL cada vez que se selecciona una tabla o un atributo de la tabla seleccionada.

Rol: Probador

Observaciones: N/A.

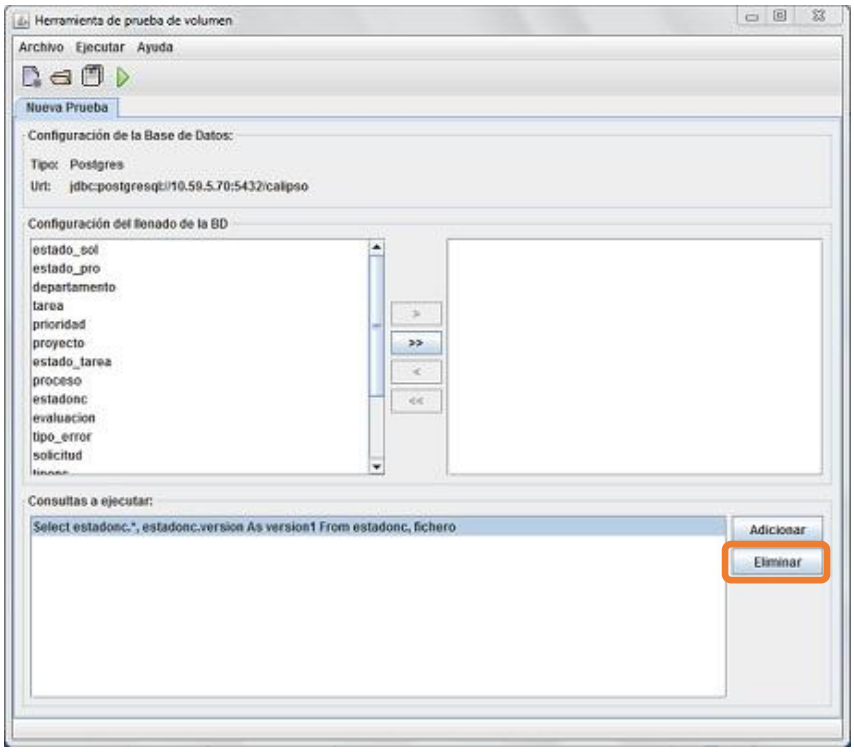
Prototipo de interface:



HU5- Eliminar consulta SQL

Tabla 8: HU5- Eliminar consulta SQL

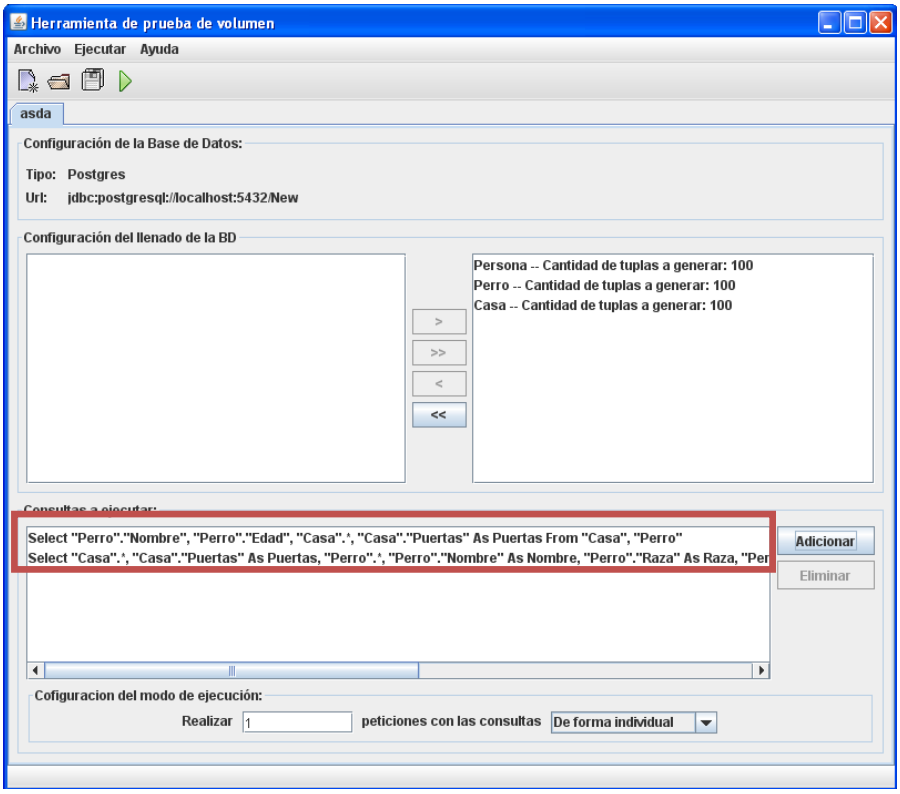
	Historia de Usuario
Código: HU5	**Nombre Historia de Usuario:** Eliminar consulta SQL
Modificación de Historia de Usuario Número: 1	

Referencia: RF3.2	
Programador: Leiser Arias Fernández	**Iteración Asignada:** Primera
Prioridad: Media	**Riesgo en Desarrollo:** Baja
Descripción: Se selecciona una consulta y se activa el botón “Eliminar” el cual permite eliminar la consulta seleccionada. **Rol:** Probador	
Observaciones: **1.** Si no se selecciona una consulta no se activa el botón “Eliminar”.	
Prototipo de interface: 	

HU6- Listar consultas SQL

Tabla 9: HU6- Listar consultas SQL

Historia de Usuario	
Código: HU6	**Nombre Historia de Usuario:** Listar consultas SQL
Modificación de Historia de Usuario Número: 1	
Referencia: RF3.3	
Programador: Leiser Arias Fernández	**Iteración Asignada:** Primera
Prioridad: Alta	**Riesgo en Desarrollo:** Baja
Descripción: El sistema muestra en el panel “Consultas a ejecutar” las consultas adicionadas actualizando el campo cada vez que se adiciona o se elimina una consulta. **Rol:** Probador	
Observaciones: **1.** Si no hay consultas el campo debe estar vacío.	
Prototipo de interface:	



HU7- Configurar el modo de ejecución de la prueba

Tabla 10: HU7- Configurar el modo de ejecución de la prueba

	Historia de Usuario
Código: HU7	**Nombre Historia de Usuario:** Configurar el modo de ejecución de la prueba.
Modificación de Historia de Usuario Número: 1	
Referencia: RF4	
Programador: Leiser Arias Fernández	**Iteración Asignada:** Primera
Prioridad : Alta	**Riesgo en Desarrollo:** Alta
Descripción: En la parte inferior del sistema se muestra el panel “Configuración del modo de ejecución”. Se definen la cantidad de peticiones y se realizaran las mismas en “Bloques” o de forma “Individual”. Se selecciona el botón “Ejecutar” y se comienza	

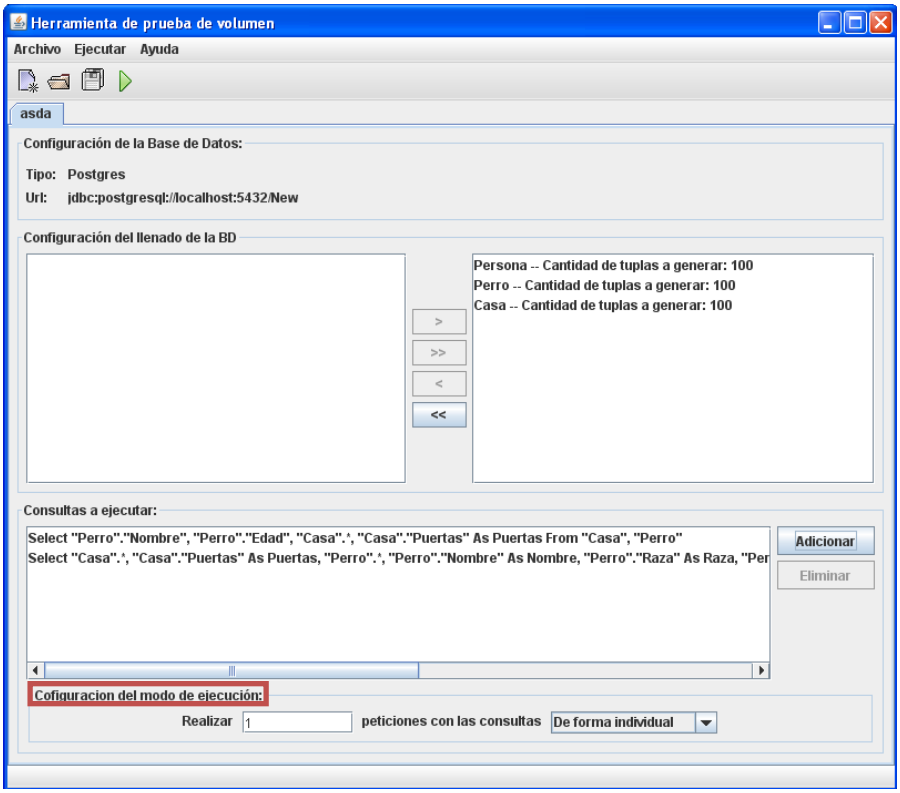
a poblar la base de datos (Ver HU8).

Rol: Probador

Observaciones:

1. La cantidad de peticiones deben ser mayor que cero.

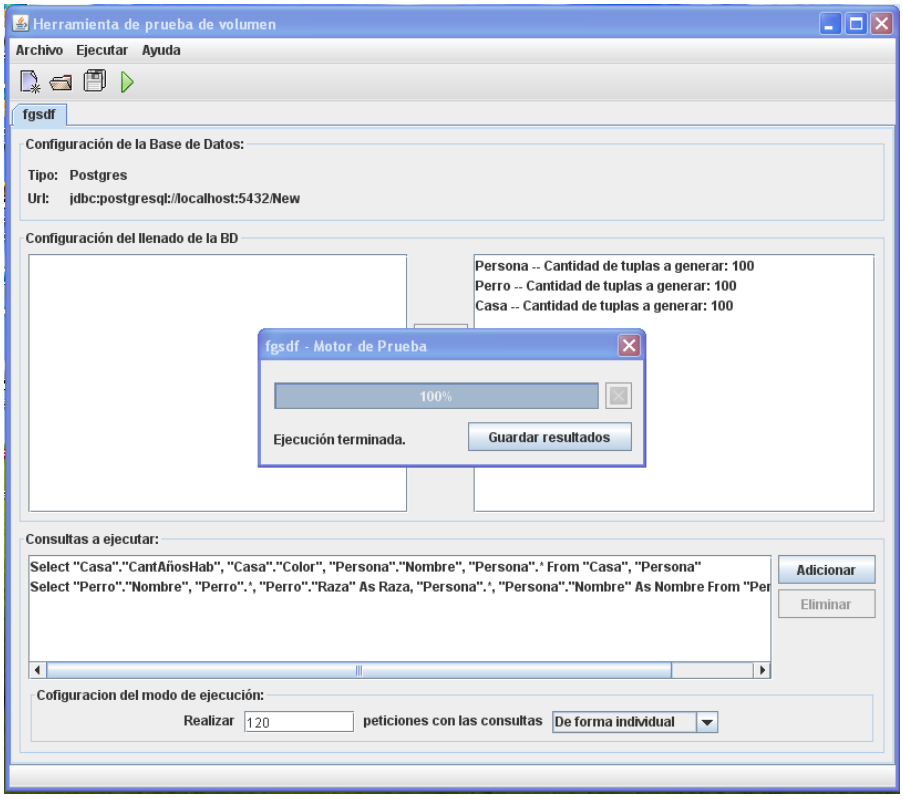
Prototipo de interface:



HU8- Poblar la Base de Datos para el Sistema Gestor de Bases de Datos PostgreSQL

Tabla 11: HU8- Poblar la Base de Datos para el Sistema Gestor de Bases de Datos PostgreSQL

	Historia de Usuario
Código: HU8	**Nombre Historia de Usuario:** Poblar la Base de Datos para el Sistema Gestor de Bases de Datos PostgreSQL
Modificación de Historia de Usuario Número: 1	
Referencia: RF5	

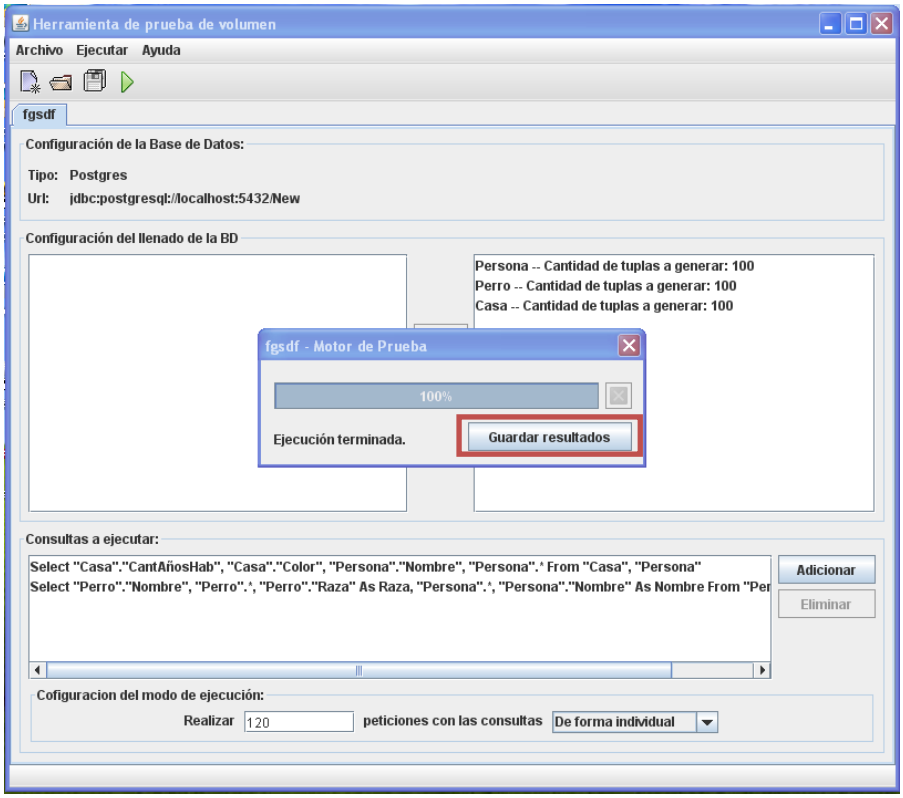
Programador: Leiser Arias Fernández	**Iteración Asignada:** Primera
Prioridad : Alta	**Riesgo en Desarrollo:** Alta
Descripción: Después de presionado el botón “Ejecutar” se muestra una ventana con el avance del poblado de la base de datos. Acto seguido se muestra el avance de la ejecución de las consultas y permite guardar el resultado (Ver HU9). **Rol:** Probador	
Observaciones: N/A.	
Prototipo de interface: 	

HU9- Generar reporte en formato PDF con los resultados de las pruebas

Tabla 12: HU9- Generar reporte en formato PDF con los resultados de las pruebas

Historia de Usuario

Código: HU9	**Nombre Historia de Usuario:** Generar reporte en formato PDF con los resultados de las pruebas
Modificación de Historia de Usuario Número: 1	
Referencia: RF6.1	
Programador: Leiser Arias Fernández	**Iteración Asignada:** Primera
Prioridad: Alta	**Riesgo en Desarrollo:** Alta
Descripción: Una vez terminada la ejecución de las consultas se muestra un botón "Guardar resultado" que permite guardar el resultado de la prueba en formato PDF. **Rol:** Probador	
Observaciones: N/A.	
Prototipo de interface:	

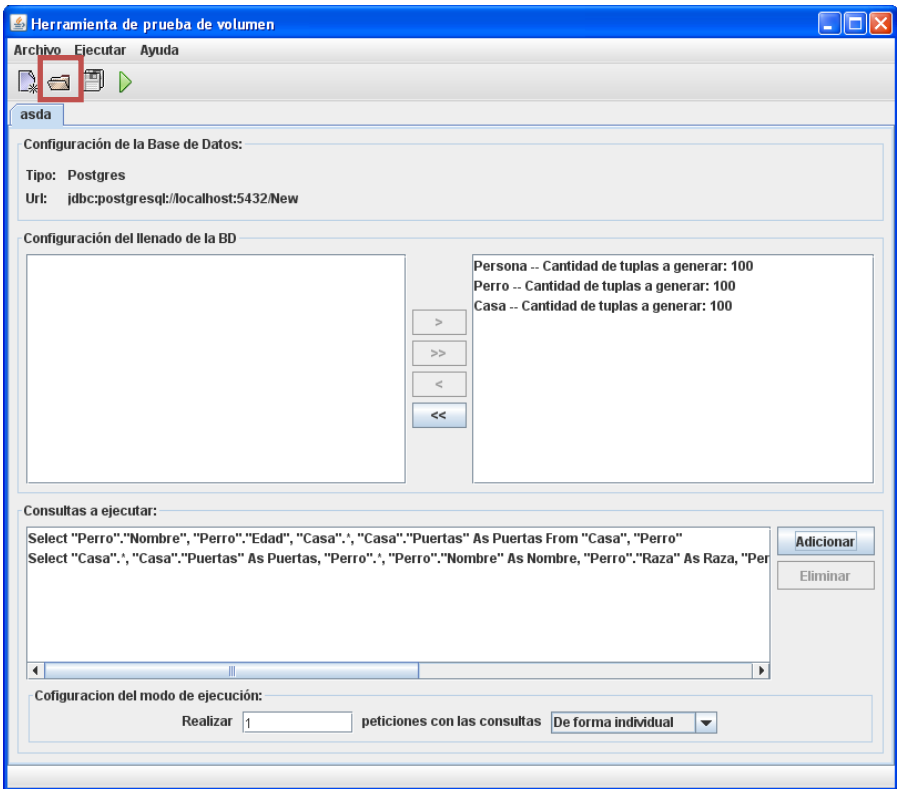


HU10- Cargar prueba

Tabla 13: HU10- Cargar prueba

Historia de Usuario	
Código: HU10	**Nombre Historia de Usuario:** Cargar prueba
Modificación de Historia de Usuario Número: 1	
Referencia: RF7	
Programador: Leiser Arias Fernández	**Iteración Asignada:** Primera
Prioridad: Alta	**Riesgo en Desarrollo:** Alta
Descripción: Al seleccionar la opción “Abrir” se muestra una ventana que permite buscar una prueba guardada. Se marca la deseada y se presiona el botón “Abrir”. El sistema carga la prueba con los últimos datos registrados. **Rol:** Probador	
Observaciones: 1. Para cargar una prueba se debe guardar una anteriormente (Ver HU11).	

Prototipo de interface:

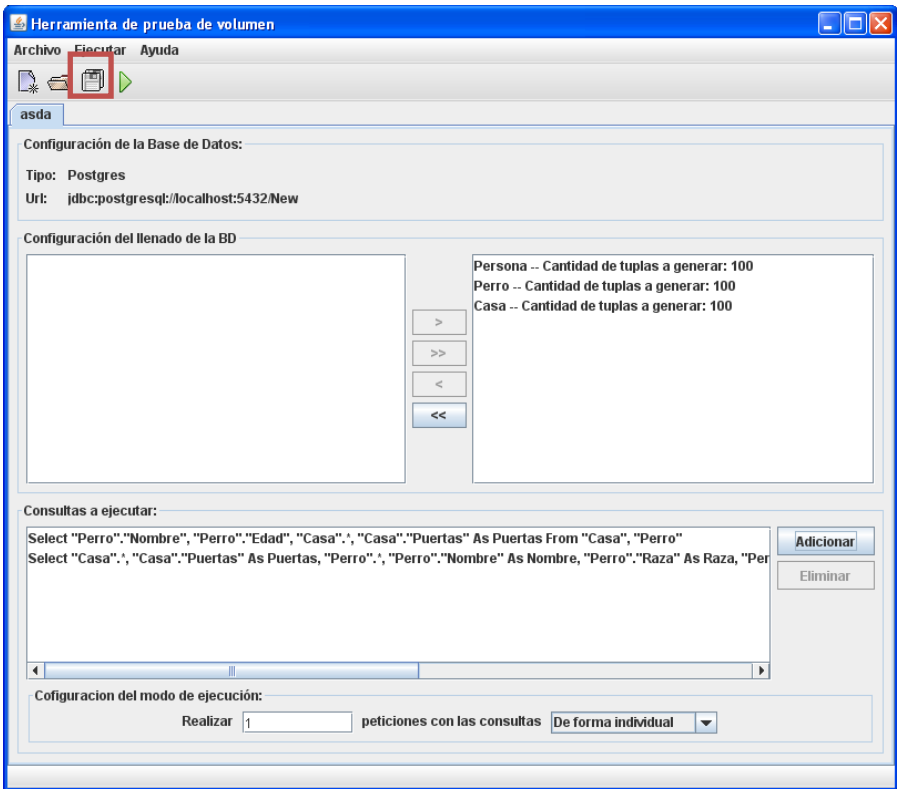


HU11- Guardar la prueba

Tabla 14: HU11- Guardar la prueba

Historia de Usuario	
Código: HU11	**Nombre Historia de Usuario:** Guardar la prueba
Modificación de Historia de Usuario Número: 1	
Referencia: RF8	
Programador: Leiser Arias Fernández	**Iteración Asignada:** Primera
Prioridad: Alta	**Riesgo en Desarrollo:** Alta
Descripción: Se selecciona la opción “Guardar” se muestra la ventana para elegir donde guardar la prueba y cambiarle el nombre.	

Rol: Probador
Observaciones: 1. Debe existir una configuración de una prueba para poderla guardar. 2. Si se está trabajando sobre una prueba guardada se actualiza la misma.
Prototipo de interface:



2.6 Tareas de ingeniería

Tabla 15: Implementar el código para realizar la conexión a la base de datos

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** HU1
Nombre Tarea: Implementar el código para realizar la conexión a la base de datos.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Leiser Arias Fernández	

Descripción: Al seleccionar la opción “Nuevo” el sistema muestra una ventana con campos de textos a llenar como “Nombre de la prueba”, “Servidor”, “Puerto”, “Nombre BD”, “Usuario”, “Contraseña” y un combobox para escoger el gestor de base de datos. Una vez llenados los campos se presiona el botón “Finalizar” y el sistema gestiona la conexión a la base de datos.

Tabla 16: Proceso de configuración de la prueba

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** HU2, HU3, HU5, HU6, HU7.
Nombre Tarea: Proceso de configuración de la prueba	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Leiser Arias Fernández	
Descripción: Al realizar correctamente la conexión a la base de datos, el sistema muestra 4 paneles uno con la configuración a la base de datos, el cual está compuesto por 2 labels, el tipo de base de datos y la dirección URL de la misma; otro panel con dos campos de textos uno para listar las tablas de la base de datos que se escogerán y el otro para mostrar las tablas escogidas para poblar, en medio de los campos deben aparecer 4 botones para pasar las tablas de un lado a otro. Cuando se pasa una o varias tablas para poblarlas el sistema muestra una ventana con un combobox con la(s) tabla(s), un campo de texto para definir la cantidad de tuplas a poblar, un botón para aplicarle a todas las tablas seleccionadas la misma cantidad de tuplas definidas, un botón para finalizar el proceso y un botón para cancelar el proceso. El tercer panel muestra el gestor de consultas que está compuesto por un campo de texto para listar las consultas agregadas y dos botones uno para adicionar y otro para eliminar la consulta seleccionada y por último el otro panel compuesto por un campo de texto para definir la cantidad de peticiones que se le van hacer a la base de datos y un combobox para escoger de qué forma se realizarán las peticiones en bloques o individuales.	

Tabla 17: Adicionar las consultas SQL que se le pedirán a la Base de Datos

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** HU4
Nombre Tarea: Adicionar las consultas SQL que se le pedirán a la Base de Datos.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Leiser Arias Fernández	
Descripción: Cuando se presiona el botón adicionar el sistema muestra una ventana con divida en 4 partes en el extremo derecho se muestran las tablas de la base de datos y se pasan para el campo del centro con doble clic o arrastrando la tabla permitiendo seleccionar los atributos deseados de la(s) tabla(s), en el campo del extremo izquierdo se va creando un árbol con las relaciones entre los atributos seleccionados y en el campo inferior se va formando de manera automática la consulta SQL. Esta ventana también tiene dos botones uno para finalizar el proceso y el otro para cancelarlo.	

Tabla 18: Poblar la Base de Datos para el Sistema Gestor de Bases de Datos PostgreSQL

Tarea de Ingeniería	
Número Tarea: 5	**Número Historia de Usuario:** HU8
Nombre Tarea: Poblar la Base de Datos para el Sistema Gestor de Bases de Datos PostgreSQL.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Leiser Arias Fernández	
Descripción: Una vez configurado todo el proceso de prueba se presiona el botón Ejecutar y el sistema muestra una ventana con una barra de progreso el cual indica que se llenó la base de datos y se realizaron las peticiones de las consultas a la misma, después de terminado este proceso se muestra en la ventana un botón para guardar el resultado de la prueba en formato pdf.	

Tabla 19: Cargar y guardar una prueba

	Tarea de Ingeniería
Número Tarea: 6	**Número Historia de Usuario:** HU10,HU11
Nombre Tarea: Cargar y guardar una prueba	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Leiser Arias Fernández	
Descripción: Cuando se ejecuta la aplicación, se muestran varias opciones una de ellas es cargar una prueba y la otra guardar una prueba. Cuando se ha realizado una prueba el sistema permite guardarla para poder utilizarla en otra ocasión, una vez presionado el botón de guardar prueba el sistema muestra una ventana de navegación para definir en qué lugar guardarla. Cuando existe una prueba guardada el sistema permite cargarla para realizar esa misma prueba o hacerle modificaciones.	

2.7 Diseño del sistema

El diseño del sistema define todo el proceso que permite representar todos los aspectos a construir en el sistema, materializando en detalles los requisitos y necesidades del cliente.

2.7.1 Arquitectura del sistema

Según la IEEE, la arquitectura del software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el entorno, así como los principios que dirigen su diseño y evolución [32].

La arquitectura es la encargada en organizar el proyecto que se desea desarrollar, logrando así una visión general y detallada de las características y relaciones de cada uno de los elementos que conforman al software.

2.7.2 Diagrama de paquete

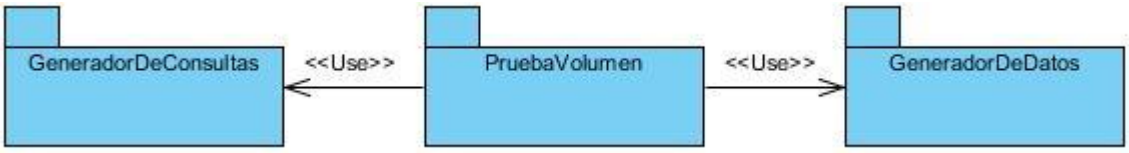


Ilustración 3: Diagrama de paquetes

Descripción de paquetes:

GeneradorDeConsultas: Se encarga de Gestionar las consultas SQL que se le pedirán a la base de datos.

GeneradorDeDatos: Se encarga de generar los tipos de datos para poblar la base de datos.

Prueba de Volumen: Es la encargada de realizar las pruebas a la base de datos.

2.7.3 Patrón de arquitectura

Un patrón es la descripción de un problema con su resultado, el cual se puede emplear en diferentes contextos y recibe un nombre. Los patrones indican al desarrollador cómo utilizarlo en diversas etapas. Ante determinada categoría de problemas muchos patrones ofrecen orientación de cómo asignar responsabilidades a los objetos.

Modelo-Vista-Controlador:

Se escoge el patrón de arquitectura MVC (Modelo Vista Controlador) porque es un patrón que define la organización independiente del Modelo (Objetos de Negocio), la Vista (interfaz con el usuario u otro sistema) y el Controlador (controlador del grupo de trabajo de la aplicación).

El patrón de arquitectura "modelo vista controlador", es una filosofía de diseño de aplicaciones, compuesta por:

- ✓ **Modelo**

  Contiene el núcleo de la funcionalidad (dominio) de la aplicación.

  Encapsula el estado de la aplicación.

- ✓ **Vista**

  Es la presentación del Modelo.

Puede acceder al Modelo pero nunca cambiar su estado.

Puede ser notificada cuando hay un cambio de estado en el Modelo.

- ✓ **Controlador**

  Reacciona a la petición del Cliente, ejecutando la acción adecuada y creando el modelo pertinente [33].

2.7.4 Patrones de diseño

Los Patrones GRASP son patrones de diseños que describen cada uno de los principios principales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Experto: La asignación de una responsabilidad a la clase que cuenta con la información necesaria para cumplir una funcionalidad en específico permite tener un código correcto y funcional a la hora de utilizarlo. No pretende otorgar una idea extraña ni oscura, simplemente expresa la intuición de que los objetos realizan funciones mediante la información que posee.

Es uno de los patrones más usados, se puede ver en casi todas las clases de un sistema, en la herramienta se refleja en la clase Tabla, la cual se encarga de emular una tabla de la base de datos.

Creador: Este patrón permite guiar la asignación de las responsabilidades que tengan relación con la creación de objetos. Su objetivo fundamental es encontrar un creador para conectarlo con el objeto producido en cualquier evento.
Se utilizó dicho patrón para las clases que tienen como responsabilidad crear instancias de otras para poder realizar completamente el funcionamiento del proceso. Esto se ve en la clase AdministradorDeConexiones la cual crea instancias para conectarse a la base de datos utilizando la clase ConfiguracionDB.

Bajo Acoplamiento: Este patrón soporta el diseño de clases más independientes, el cual disminuye el impacto de cambios y también más reutilizables, ya que amplían la oportunidad de alcanzar una mayor productividad. No debería considerarse independiente de otros patrones como “El Experto” o “El Alta cohesión”, sino que debería tomarse como uno de los principios del diseño que influye en la decisión de asignar responsabilidades.
Se utilizó este patrón en la clase ConfiguracionDB la cual no depende de otras clases para realizar la conexión a la base de datos.

Alta cohesión: Al igual que el patrón Bajo Acoplamiento, el patrón de Alta cohesión es unos de los principios que se debe tener en cuenta en todas las decisiones del diseño. Generalmente una clase que esté altamente cohesionada tiene bajo acoplamiento desde el punto de vista de clases, ya que posee la menor cantidad de dependencias posibles con otras clases y a su vez posee responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas.

Este se ve evidenciado en el diseño de clases, realizando agrupaciones por funcionalidades y así facilitar la reutilización de las clases. Cada elemento del diseño realiza una única funcionalidad dentro del sistema, este se puede ver en la clase GeneradorDatosPostgres utiliza la clase AdministradorDeConexiones para conectarse a la base de datos auxiliándose en la clase ConfiguracionDB [34].

2.7.5 Diagrama de Clases del Sistema

El diagrama de clases es el diagrama principal de diseño y análisis para un sistema. En él, la estructura de clases del sistema se especifica, con relaciones entre clases y estructura de herencia. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama y se modifica para satisfacer los detalles de las implementaciones [35].

En el epígrafe anterior se definió la arquitectura, por lo que en este se abordará sobre las clases persistentes.

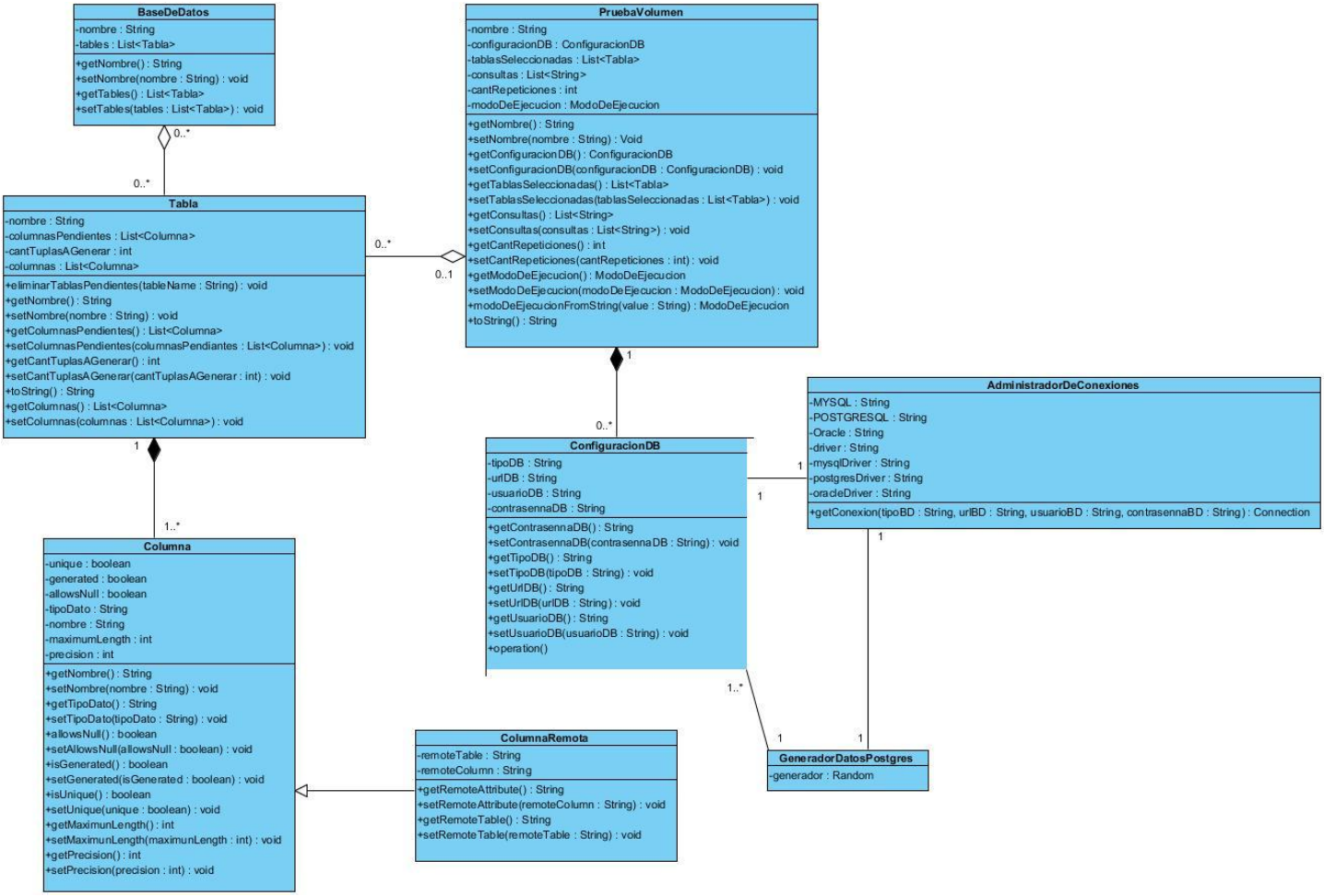


Ilustración 4: Diagrama de Clases del Sistema

Descripción de clases:

PruebaVolumen: Esta clase se encarga de hacer la configuración de la Prueba de Volumen.

BaseDeDatos: Esta clase se encarga de emular una base de datos.

Tabla: Esta clase se encarga de emular una tabla en la base de datos.

Columna: Esta clase se encarga de emular una columna en la Tabla.

ColumnaRemota: Esta clase se encarga de emular una columna remota (Llave foránea).

ConfiguracionDB: Esta clase se encarga de hacer la configuración para la conexión a la base de datos.

AdministradorDeConexiones: Esta clase se encarga de administrar las conexiones a la base de datos.

GeneradorDatosPostgres: Esta clase se encarga de generar los tipos de datos para el poblado de la base de datos.

2.8 Conclusiones

En este capítulo se realizó una descripción de particularidades de la herramienta a través de los distintos artefactos de análisis y diseño que se describieron, historias de usuarios, tareas ingenieriles etc. Posterior al análisis de la situación problémica se dieron a conocer los elementos necesarios para el proceso de desarrollo.

El análisis y diseño ha sido la línea a seguir durante este capítulo arrojando excelentes resultados y permitiendo el siguiente paso que será la implementación de todo lo estudiado.

Capítulo 3: Implementación y Validación de la Solución

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN

3.1 Introducción

Para implementar y validar la herramienta que se desea realizar, es necesario fundamentar los objetivos propuestos. Para ello se realizará el Diagrama de Despliegue, el Diagrama de Componentes y se muestra los Diseños de Casos de Pruebas para cada una de las Historias de Usuario implementadas.

3.2 Diagrama de Despliegue

El diagrama de despliegue es el encargado de mostrar cómo están relacionados físicamente la aplicación con el sistema de base de datos que se quiera probar. El cual está formado por nodos Aplicación y Servidor de Base de Datos los que están conectador por el protocolo TCP/IP para el funcionamiento de las pruebas de volumen.

Ilustración 5: Diagrama de Despliegue

3.3 Diagrama de Componentes

Los diagramas de componentes muestran cómo el sistema está dividido en componentes y las dependencias entre ellos, proveen una vista arquitectónica de alto nivel del sistema. Ayudan a los desarrolladores a visualizar el camino de la implementación [36].
Permiten una sencilla representación de la organización y dependencia de los componentes, además que pueden ser usados para modelar y documentar la arquitectura de cualquier sistema.

Capítulo 3: Implementación y Validación de la Solución

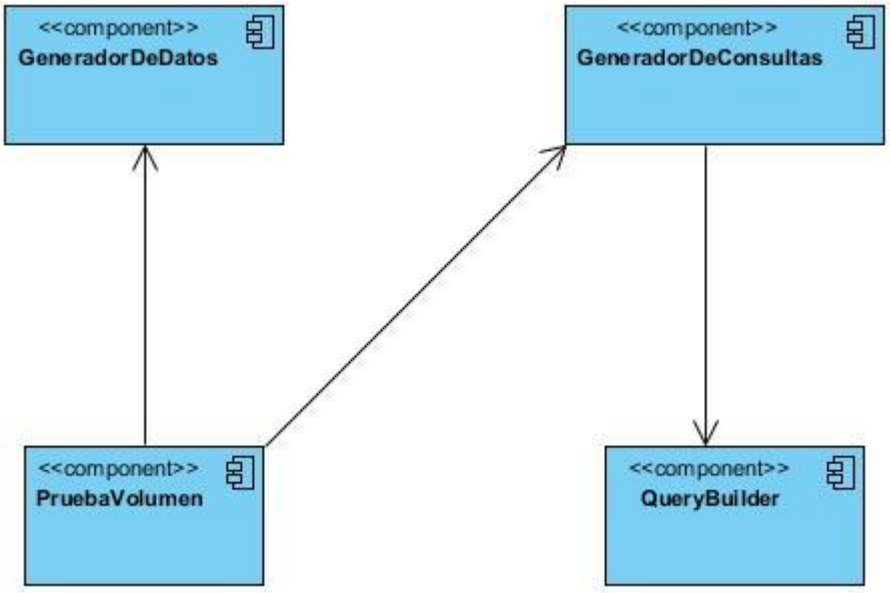


Ilustración 6: Diagrama de Componentes

3.4 Estándares de codificación

Convenciones de nomenclatura

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto con minúscula, en caso de ser una palabra compuesta se empleará la notación Pascal Casing. Ejemplo: ConfiguracionManager.

Los nombres de los métodos y los atributos de las clases, comienzan con la primera letra en minúscula, en caso de que sea un nombre compuesto se empleará notación Camel Casing. Ejemplo: idMensaje.

Nomenclatura según el tipo de clase

Las clases que se encuentran dentro del paquete Controller se nombran adicionándoles el nombre del controlador. Ejemplo: MainFormController.

3.5 Validación del diseño propuesto

Los atributos de calidad que se tienen en cuenta para la evaluación del diseño propuesto son:

- ✓ **Responsabilidad**: Responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta.
- ✓ **Complejidad del mantenimiento**: Nivel de esfuerzo necesario para sustentar, mejorar o corregir el diseño de software propuesto. Puede influir significativamente en los costes y la planificación del proyecto.
- ✓ **Complejidad de implementación**: Grado de dificultad que tiene implementar un diseño de clases determinado.
- ✓ **Reutilización**: Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.
- ✓ **Acoplamiento**: Dependencia o interconexión de una clase o estructura de clase respecto a otras.
- ✓ **Cantidad de pruebas**: Número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto (componente) diseñado.

Para la evaluación de la calidad del diseño se utilizaron las siguientes métricas:

Tamaño operacional de clase (TOC): Se refiere al número de métodos pertenecientes a una clase. Está determinada por los atributos: Responsabilidad, Complejidad de implementación y la Reutilización, existiendo una relación directa con los dos primeros e inversa con el último antes mencionado.

Relaciones entre clases (RC): Dado por el número de relaciones de uso de una clase. Está determinada por los atributos: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado.

Las métricas escogidas son bastante eficientes ya que dan una medida de la calidad del diseño del componente y su utilización es sencilla y fácil.

Tabla 20: Tamaño operacional de clase (TOC)

Atributo que afecta	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.

Reutilización	Un aumento del TOC implica una disminución en el grado de reutilización de la clase.

Tabla 21: 7 Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC

Atributo	Categoría	Criterio
Responsabilidad	Baja	< =Prom (7,62)
	Media	Entre Prom. Y 2* Prom
	Alta	> 2* Prom
Complejidad implementación	Baja	< =Prom
	Media	Entre Prom. y 2* Prom
	Alta	> 2* Prom
Reutilización	Baja	> 2*Prom
	Media	Entre Prom. y 2* Prom
	Alta	<= Prom

Tabla 22: Atributo de calidad Responsabilidad

Responsabilidad	Cantidad de clases	Promedio
Baja	28	75,67567568
Media	6	16,21621622
Alta	3	8,108108108

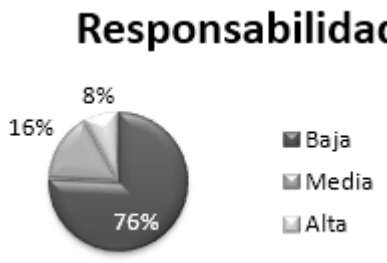


Ilustración 7: Atributo de calidad Responsabilidad

Tabla 23: Atributo de calidad Complejidad

Complejidad	Cantidad de clases	Promedio
Baja	28	75,67567568

Media	6	16,21621622
Alta	3	8,108108108

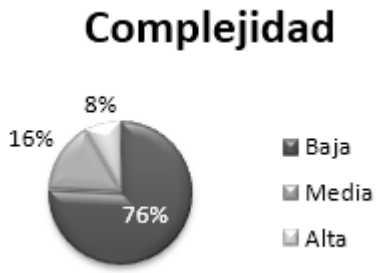


Ilustración 8: Atributo de calidad Complejidad

Tabla 24: Atributo de calidad Reutilización

Reutilización	Cantidad de clases	Promedio
Alta	28	75,67567568
Media	4	10,81081081
Baja	2	5,405405405

Ilustración 9: Atributo de calidad Reutilización

Analizando los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, teniendo en cuenta que el 76% de las clases son bajas en los atributos de calidad Responsabilidad y Complejidad ya que estas clases poseen menor cantidad de operaciones que el promedio registrado en las mediciones, y que el 82% de las clases tienen una alta Reutilización. Se puede concluir que el diseño es aceptable ya que posee evaluaciones positivas en los atributos de calidad.

Tabla 25: Relaciones entre clases (RC)

Atributo que afecta	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad del mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 26: Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica RC

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de Mantenimiento	Baja	< =Prom (2,68)
	Media	Entre Prom. y 2* Prom
	Alta	> 2* Prom
Reutilización	Baja	> 2*Prom
	Media	Entre Prom. y 2* Prom
	Alta	<= Prom
Cantidad de Pruebas	Baja	< =Prom
	Media	Entre Prom. y 2* Prom
	Alta	> 2* Prom

Tabla 27: Atributo de calidad Acoplamiento

Acoplamiento	Cantidad de clases	Promedio
Ninguno	14	37,83783784
Bajo	7	18,91891892
Medio	7	18,91891892
Alto	9	24,32432432

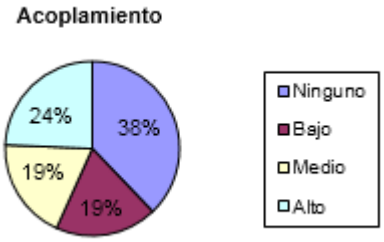


Ilustración 10: Atributo de calidad Acoplamiento

Tabla 28: Atributo de calidad Complejidad de Mantenimiento

Complejidad de Mantenimiento	Cantidad de clases	Promedio
Baja	28	75,67567568
Media	4	10,81081081
Alta	5	13,51351351

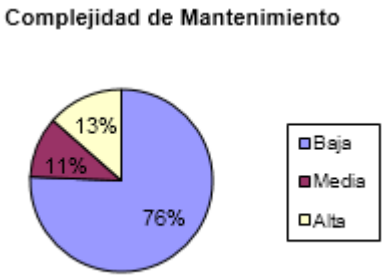


Ilustración 11: Atributo de calidad Complejidad de Mantenimiento

Tabla 29: Atributo de calidad Cantidad de Pruebas

Cantidad de Pruebas	Cantidad de clases	Promedio
Baja	28	75,67567568
Media	4	10,81081081
Alta	5	13,51351351

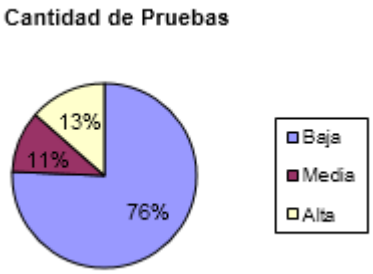


Ilustración 12: Atributo de calidad Cantidad de Pruebas

Tabla 30: Atributo de calidad Reutilización

Reutilización	Cantidad de clases	Promedio
Baja	5	13,51351351
Media	4	10,81081081
Alta	28	75,67567568

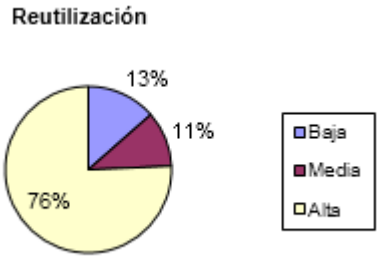


Ilustración 13: Atributo de calidad Reutilización

Al analizar los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño tiene una calidad aceptable para realizar la implementación, teniendo en cuenta que el 76% de las clases no tiene o es bajo el acoplamiento entre clases. Además los atributos de calidad Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización poseen niveles aceptables en un 76 % de las clases.

Capítulo 3: Implementación y Validación de la Solución

3.6 Pruebas

Los Diseños de Casos de Pruebas son los encargados como su nombre indica de diseñar pruebas que tengan más posibilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y de tiempo. Se escogieron los casos de pruebas que se utilizan en Calisoft ya que los que plantea la metodología son los casos de pruebas de aceptación y en el departamento de prueba de Calisoft se realizan primero las pruebas de liberación y después las pruebas de aceptación; además de ser más explícito a la hora de realizar las pruebas.

HUAP01- Realizar la conexión a la Base de Datos

Tabla 31: HUAP01- Realizar la conexión a la Base de Datos

Escenario	Descripción	Variable	Respuesta del sistema	Flujo central
Realizar la conexión a la base de datos correctamente.	Mediante este escenario se realiza la conexión a la base de datos satisfactoriamente.	Acepta todo tipo de carácter.	Una vez conectada la base de datos el sistema muestra todas las tablas de la base de datos.	1) Se selecciona el menú “**Archivo**”. 2) Se selecciona la opción “**Nuevo**”. 3) Se llenan los campos para la configuración de la base de datos. 4) Se presiona el botón “**Finalizar**”.
Realizar la conexión a la base de datos incorrectamente.	Mediante este escenario se realiza la conexión a la base de datos llenando los campos incorrectamente.	Acepta todo tipo de carácter.	Una vez presionado el botón finalizar el sistema muestra el siguiente mensaje “Existe errores con sus parámetros de conexión. Por favor verifíquelos”.	1) Se selecciona el menú “**Archivo**”. 2) Se selecciona la opción “**Nuevo**”. 3) Se llenan los campos para la configuración de la base de datos. 4) Se presiona el botón “**Finalizar**”.
Realizar la	Mediante este	Acepta todo	El sistema muestra el	1) Se selecciona el menú “**Archivo**”.

conexión a la base de datos con el mismo nombre de una abierta.	escenario se añade una conexión con el mismo nombre de una que esté abierta.	tipo de carácter.	mensaje de error “Ya existe una prueba con el nombre especificado.”.	2) Se selecciona la opción “**Nuevo**”. 3) Se llenan los campos para la configuración de la base de datos. Se presiona el botón “**Finalizar**”.
Cancelar la conexión a la base de datos.	Mediante este escenario se cancela la conexión a la base de datos.	Acepta todo tipo de carácter.	El sistema cierra la ventana.	1) Se selecciona el menú “**Archivo**”. 2) Se selecciona la opción “**Nuevo**”. 3) Se llenan los campos para la configuración de la base de datos. 4) Se presiona el botón “**Cancelar**”.

HUAP02- Determinar cuáles tablas poblar de la Base de Datos

Tabla 32: HUAP02- Determinar cuáles tablas poblar de la Base de Datos

Escenario	Descripción	Variable	Respuesta del sistema	Flujo central
Determinar cuáles tablas poblar de la base de datos.	Mediante este escenario se determina cuáles tablas poblar de la base de datos.	Números, número máximo 10000.	Una vez determinado cuáles tablas poblar de la base de datos el sistema muestra las tablas.	1) El usuario marca la tabla que desea poblar y presiona el botón “**>**”. 2) Define la tabla y la cantidad de tuplas. 3) Si presiona el botón “**Aplicar a todos**” le da a todas las tablas que aparecen en el combobox “**Tablas:**” la misma cantidad de tuplas definida por el usuario. 4) Presiona el botón “**Finalizar**”. 5) Si quiere poblar todas las tablas presiona el botón “**>>**”.

				6) Define la tabla y la cantidad de tuplas. 7) Si presiona el botón "**Aplicar a todos**" le da a todas las tablas que aparecen en el combobox "**Tablas:**" la misma cantidad de tuplas definida por el usuario. 8) Presiona el botón "**Finalizar**".
Cancelar cuáles tablas poblar de la base de datos.	Mediante este escenario se cancela la operación de determinar cuáles tablas poblar de la base de datos.	Números, número máximo 10000.	El sistema cierra la ventana.	1) El usuario marca la tabla que desea poblar y presiona el botón "**>**". 2) Presiona el botón "**Cancelar**". 3) Si quiere poblar todas las tablas presiona el botón "**>>**". 4) Presiona el botón "**Cancelar**".

HUAP03- Adicionar consulta SQL

Tabla 33: HUAP03- Adicionar consulta SQL

Escenario	Descripción	Variable	Respuesta del sistema	Flujo central
Adicionar consulta SQL.	Mediante este escenario se adiciona una consulta SQL.	N/A	Una vez adicionada la consulta SQL el sistema actualiza el listado de consultas con la nueva consulta adicionada.	1) Se presiona el botón "**Adicionar**". 2) Se escoge la(s) tabla(s) de las cuales se van a pedir atributos. 3) Se presiona el botón "**Finalizar**".
Cancelar la adición de la	Mediante este escenario se	N/A	Una vez cancelada la adición el sistema	1) Se presiona el botón "**Adicionar**". 2) Se escoge la(s) tabla(s) de las

consulta SQL.	cancela la adición de la consulta SQL.		cierra la ventana.	cuales se van a pedir atributos. Se presiona el botón "**Cancelar**".

HUAP04- Eliminar consulta SQL

Tabla 34: HUAP04- Eliminar consulta SQL

Escenario	Descripción	Variable	Respuesta del sistema	Flujo central
Eliminar consulta SQL.	Mediante este escenario se elimina la consulta SQL seleccionada.	N/A	Una vez eliminada la consulta SQL el sistema actualiza el listado de consultas, quitando la consulta seleccionada.	1) Se selecciona la consulta que se desea eliminar. 2) Se presiona el botón "**Eliminar**".

HUAP05- Listar consultas SQL

Tabla 35: HUAP05- Listar consultas SQL

Escenario	Descripción	Variable	Respuesta del sistema	Flujo central
Listar consultas SQL.	Mediante este escenario se listan las consultas SQL que se han adicionado.	N/A	El sistema muestra el listado de consultas SQL.	N/A

HUAP06- Configurar el modo de ejecución de la prueba

Tabla 36: HUAP06- Configurar el modo de ejecución de la prueba

Escenario	Descripción	Variable	Respuesta del sistema	Flujo central
Configurar el modo de	Mediante este escenario se	Solamente números.	N/A	1) Se define la cantidad de peticiones que se desean realizar con las

ejecución de la prueba de forma individual.	configura el modo de ejecución de la prueba de forma individual.			consultas. 2) Se selecciona la opción "**De forma individual**" en el combobox.
Configurar el modo de ejecución de la prueba como un bloque.	Mediante este escenario se configura el modo de ejecución de la prueba como un bloque.	Solamente números.	N/A	1) Se define la cantidad de peticiones que se desean realizar con las consultas. 2) Se selecciona la opción "**Como un bloque**" en el combobox.

HUAP07- Poblar la Base de Datos para el Sistema Gestor de Bases de Datos PostgreSQL

Tabla 37: HUAP07- Poblar la Base de Datos para el Sistema Gestor de Bases de Datos PostgreSQL

Escenario	Descripción	Variable	Respuesta del sistema	Flujo central
Poblar la base de datos para el Sistema Gestor de Bases de Datos PostgreSQL	Mediante este escenario se puebla la base de datos para el Sistema Gestor de Bases de Datos PostgreSQL.	N/A	Una vez poblada la base de datos el sistema pasa a realizar las peticiones de las consultas a la base de datos.	1) Se selecciona el menú "**Ejecutar**". 2) Se selecciona la opción "**Ejecutar prueba**".

HUAP08- Generar reporte en formato PDF con los resultados de las pruebas

Tabla 38: HUAP08- Generar reporte en formato PDF con los resultados de las pruebas

Escenario	Descripción	Variable	Respuesta del sistema	Flujo central
Generar reporte en	Mediante este escenario se	N/A	Una vez terminada la ejecución de la	1) Se presiona el botón "**Guardar resultados**".

formato PDF con los resultados de las pruebas.	genera un reporte en formato PDF con los resultados de las pruebas.		Prueba de Volumen el sistema nos permite guardar un reporte del resultado en formato PDF.	2) Se determina donde se guardarán los resultados. 3) Se presiona el botón "**Guardar**".

HUAP09- Cargar prueba

Tabla 39: HUAP09- Cargar prueba

Escenario	Descripción	Variable	Respuesta del sistema	Flujo central
Cargar prueba.	Mediante este escenario se carga una prueba que ya haya sido guardada.	N/A	El sistema permite al usuario cargar una prueba que se haya realizado.	1) Se selecciona el menú "**Archivo**". 2) Se selecciona la opción "**Abrir**". 3) Se busca la prueba que se desea cargar. 4) Se presiona el botón "**Abrir**".

HUAP10- Guardar prueba

Tabla 40: HUAP10- Guardar prueba

Escenario	Descripción	Variable	Respuesta del sistema	Flujo central
Guardar prueba.	Mediante este escenario se guarda la prueba realizada.	N/A	Una vez terminada la Prueba de Volumen el sistema nos permite guardar la prueba en un formato PB.	1) Se selecciona el menú "**Archivo**". 2) Se selecciona la opción "**Guardar**" o "**Guardar como**". 3) Se selecciona la carpeta donde guardar la prueba. 4) Se presiona el botón "**Guardar**".

Capítulo 3: Implementación y Validación de la Solución

3.7 Prueba de Liberación

El proceso de validación de este sistema fue a través de las pruebas de liberación, servicio que brinda Calisoft, consta de las siguientes etapas:

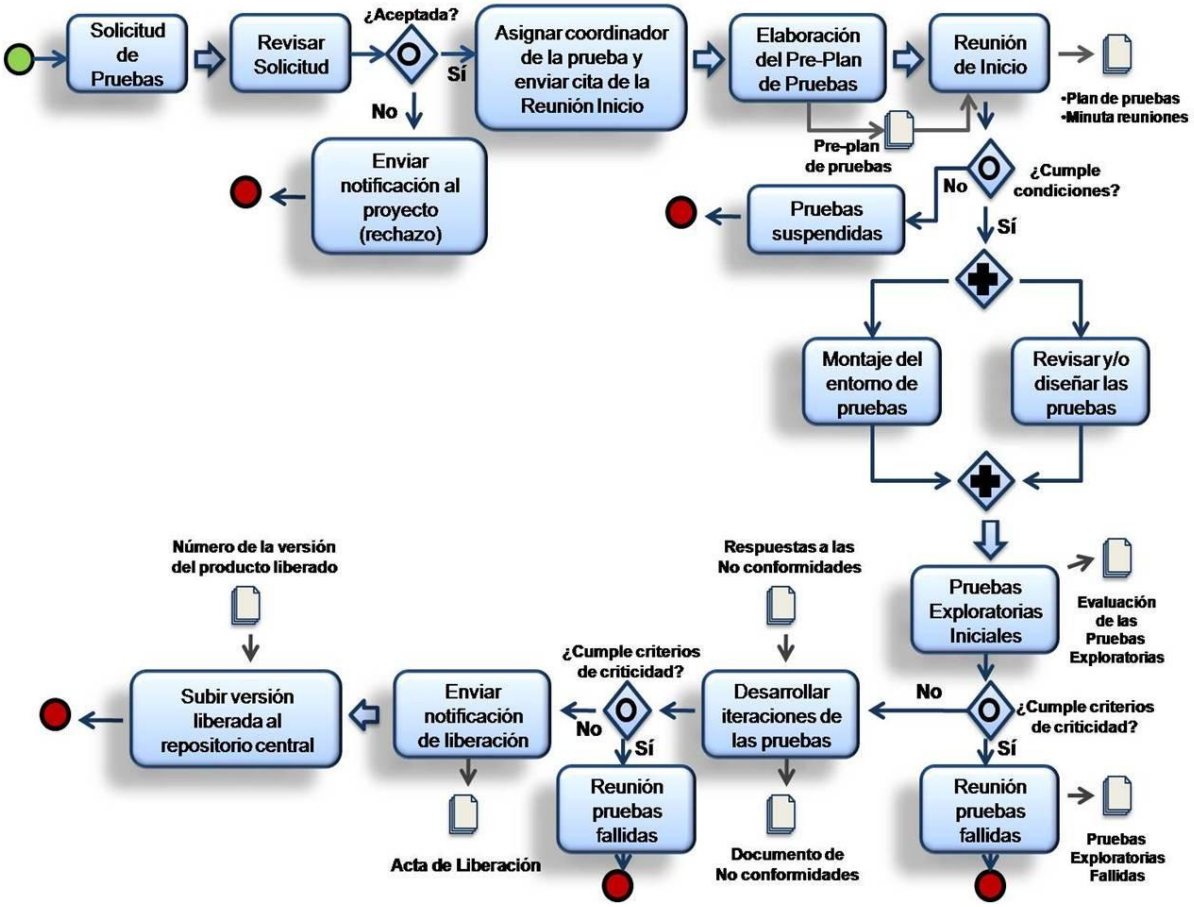


Ilustración 14: Proceso de Liberación de calisoft.

En el proceso de liberación se realizó 1 prueba exploratoria y 2 iteraciones, en las siguientes figuras se evidencian los resultados de las mismas.

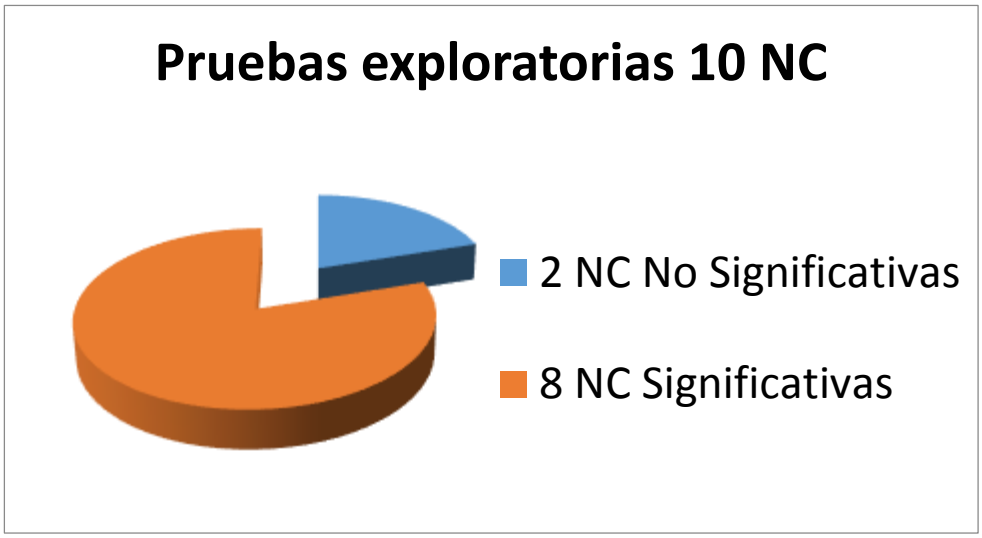


Ilustración 15: Prueba exploratoria

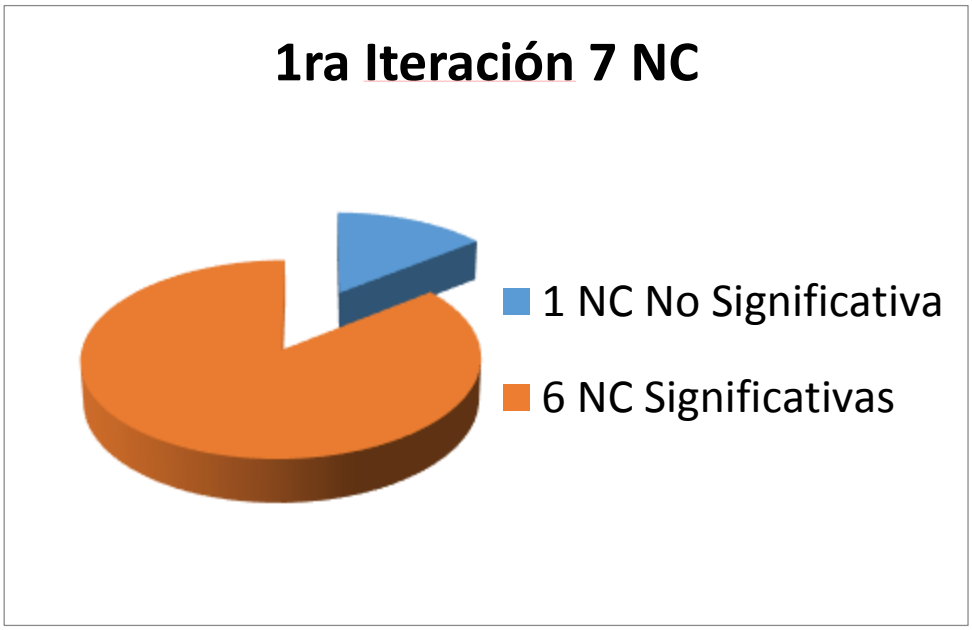


Ilustración 16: 1ra Iteración

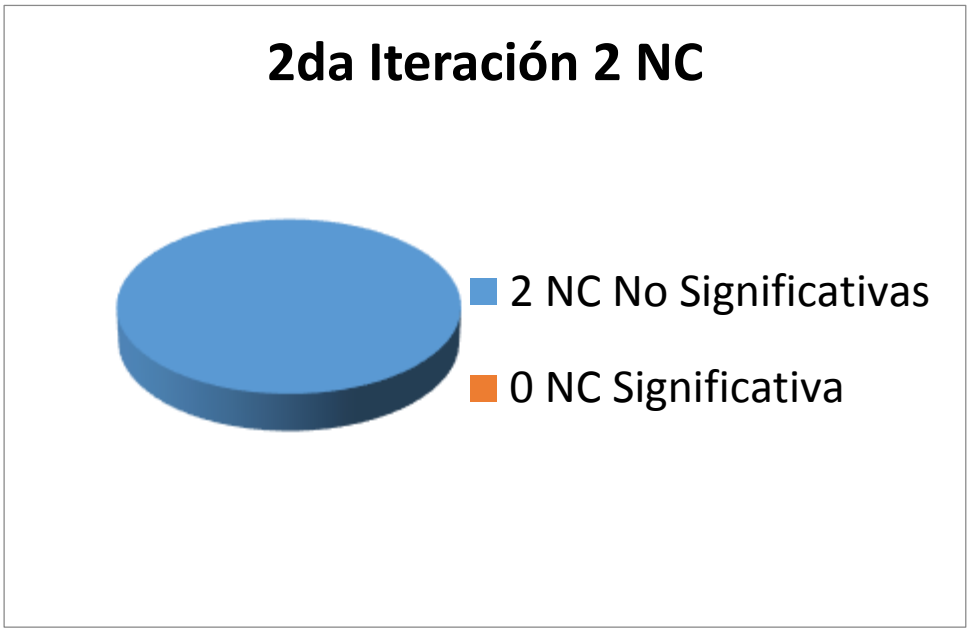


Ilustración 17: 2da Iteración

En la 2da Iteración se encontraron 2 NC No significativas las cuales se les dio resultado satisfactorio en la regresión de la iteración.

3.8 Conclusiones

En este capítulo se realizó la implementación de la herramienta capaz de realizar las pruebas de volumen, se validó la misma mediante las pruebas de liberación, realizando casos de pruebas a todas sus funcionalidades. Este proceso permitió detectar la mayor cantidad de no conformidades que presentaba la herramienta para darle soluciones. Las pruebas de forma general devolvieron resultados satisfactorios, siendo corregidos los errores encontrados.

CONCLUSIONES GENERALES

Para dar cumplimiento a los objetivos específicos y en concordancia con las necesidades planteadas por el cliente, se realizó un estudio de la situación actual del Centro Nacional de Calidad de Software en cuanto a las Pruebas de Volumen, para mejorar el proceso de pruebas de la entidad. Se desarrolló una herramienta capaz de realizar las Pruebas de Volumen con más facilidad y eficiencia al software que la requiera, siempre que tenga como gestor de base de datos PostgreSQL. Se utilizó la metodología SXP y como lenguaje de modelado UML. La realización por parte del cliente de las pruebas de liberación fueron satisfactorias.

RECOMENDACIONES

Se recomienda seguir con la implementación para los demás Sistemas Gestores de Base de Datos, como son MySQL y Oracle para el llenado de la base de datos de dichos Gestores.

Bibliografía

BIBLIOGRAFÍA

1. Arrieche, E., *Pruebas de Sistema*. 2011, Universidad Nacional Experimental Politécnica de la Fuerza Armada Núcleo Caracas Chuao Implantación de Sistema: Caracas.
2. Capote, T., *Conceptualización e implantación de un Laboratorio Industrial de Pruebas de Software*. 2011, Universidad de las Ciencias Informáticas: La Habana.
3. Kan, S.H., *Metrics and models in software quality engineering*. 1995, USA.
4. Pressman, R.S. 1998.
5. Pressman, R.S., *Software Engineering: A Practitioner's Approach*. 2000.
6. Pressman, R.S., *Can Internet based Applications Be Engineered*. 1998.
7. García, L.R.M.M., *Diseño de Bases de Datos*. 1999.
8. *Pruebas*. 2001; Available from: http://lsc.mxl.uabc.mx/~angelica/Pruebas.pdf.
9. *Pruebas de desempeño*. Available from: http://qvision.us/es/articulos-de-interes/126-pruebas-de-desempeno.html.
10. Londoño, J.H.A. *Tipos de pruebas de software*. 2005 6 de abril del 2005; Available from: http://ing-sw.blogspot.com/.
11. *Webserver Stress Tool*. 2007 noviembre de 2007; Available from: http://www.softqanetwork.com/webserver-stress-tool.
12. *AdventNet QEngine*. 2006 agosto de 2006; Available from: http://www.freedownloadmanager.org/es/downloads/AdventNet_QEngine_22367_p/.
13. *Jameleon*. 2008 febrero de 2008; Available from: http://jameleon.sourceforge.net/index.html.
14. Mutti, M. *Nessus, seguridad a alta velocidad*. 2011 febrero de 2011; Available from: http://www.aplicacionesempresariales.com/nessus-seguridad-a-alta-velocidad.html.
15. Ruvalcaba, M. 2008 septiembre de 2008; Available from: http://www.manuelruvalcaba.com/tag/loadrunner/.
16. Microsoft. *Application Center Test (ACT) prueba las aplicaciones Web mediante su carga simultánea*. 2004 febrero de 2004; Available from: http://support.microsoft.com/kb/307492/es.
17. *Selenium IDE, una herramienta para realizar pruebas de aplicaciones web*. 2008 abril de 2008; Available from: http://dacosta51.wordpress.com/2008/04/24/selenium-ide-una-herramienta-para-realizar-pruebas-de-aplicaciones-web/.

Bibliografía

18. *¿Qués es Selenium? breve introducción a Selenium*. 2012 abril de 2012; Available from: http://documentados.com/blog/oskar/ques-es-selenium-breve-introduccion-selenium.
19. *Apache JMeter*. 2012; Available from: http://jakarta.apache.org/jmeter/.
20. Asensio, R.M.-B. 2011 Octubre de 2011]; Available from: http://www.um.es/docencia/barzana/IAGP/IAGP2-Metodologias-de-desarrollo.html.
21. I. Jacobson, G.B., J. Rumbaugh, *El Proceso Unificado de Desarrollo*. 2000.
22. E. Hernández, J.H., C. Lizandra, *C++ Es-tandar*. 2001.
23. Marin, M.D.A. *Definición de lenguaje de programación.* 2008; Available from: http://catedraprogramacion.foroactivos.net/t83-definicion-de-lenguaje-de-programacion-tipos-ejemplos.
24. Oracle. *¿Qué es la tecnología Java y por qué lo necesito?* 2012; Available from: http://www.java.com/es/download/faq/whatis_java.xml.
25. Oracle. *Conozca más sobre la tecnología Java*. 2012; Available from: http://www.java.com/es/about/.
26. *Visual Paradigm for UML*. 2007; Available from: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%20%28Iglesia_Anglicana%29%20_%5BMac_OS_X_cuenta_14717_p/.
27. *Qué es un entorno de desarrollo integrado, IDE.* 2011; Available from: http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide/.
28. Oracle. 2012; Available from: https://netbeans.org/community/releases/69/index_es.html.
29. MSc. Silvia López Riquelme, M.M.C.M. *Sistema Gestor de Base de Datos Relacionales.* 2010; Available from: http://www.fec.uh.cu/CUGIO/1%20acciones/Contenidos/BDR.pdf.
30. Martinez, R. *Sobre PostgreSQL*. 2012; Available from: http://www.postgresql.org.es/sobre_postgresql.
31. 830-1998, I.S., *Especificación de Requisitos Software según el estándar de IEEE 830.* 2008.
32. IEEE, *Recommended Practice for Software Architecture Descriptions of Software Intensive Systems.* 2000.
33. *Patrón de arquitectura Modelo Vista Controlador (MVC)*. Available from: http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html.
34. Larman, C., *UML y Patrones*. 2a Edición ed.

Bibliografía

35. *Modelado de Sistemas con UML*. Available from: http://mmc.geofisica.unam.mx/LuCAS/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x219.html.

36. Alva, E.R. *Diagramas de Componentes y Despliegue*. 2008 noviembre de 2008; Available from: http://es.scribd.com/doc/7884665/Arquitectura-de-Software-II-Diagrama-de-Componentes-y-Despliegue.

ISO/IEC 25000:2005

Apache JMeter. Manual de usuario v1.2 Autor: Consultoría de áreas de conocimiento. Fecha: 20/05/2009

Plan de pruebas maestro v1.0. Autores: Paolo Carrasco, Nilton Guevara y Jorge Palacios.

Revista Cubana de Ciencias Informáticas. “Enfoque del procedimiento de calidad de datos del sistema informativo del Banco Central de Cuba” Autores: Verónica Évora Torres, Ramiro A. Pérez Vázquez.

Revista Cubana de Ciencias Informáticas. “El empleo de métodos de toma de decisión y técnicas de soft computing en la selección de personal” Autores: Lizandra Arza Pérez, Edistio Yoel Verdecia Martínez, José Lavandero García.

Anexos

Anexo 1.