Universidad de las Ciencias Informáticas
Facultad 3

Título: Implementación de una herramienta para el Análisis de las trazas en el Marco de trabajo Sauxe.

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

Autora: Mayara López Padrón

Tutor(es): Ing. Ariel Torres Gálvez

Ing. René R. Bauta Camejo

Habana, Cuba
Junio, 2011

"Podrán morir las personas pero jamás sus ideas."

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

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

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

Mayara López Padrón

_____

Firma del Autor

Ing. René R. Bauta Camejo

_____

Firma del Tutor

Ing. Ariel Torres Gálvez

_____

Firma del Tutor

DATOS DE CONTACTO

Síntesis del Tutor: Ing. Ariel Torres Gálvez.

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2008. Es categorizado como instructor, se ha desarrollado como arquitecto de datos del sistema Cedrux (ERP-Cubano) desde el 2008 y se mantiene activo en ese rol.

Síntesis del Tutor: Ing. René R. Bauta Camejo

Graduado del 2009, se vinculó al desarrolló como programador en el proyecto ERP-Cuba desde el 4to año de su carrera en la línea de arquitectura. Allí desarrolló el mapeador de doctrine, herramienta que actualmente se usa en el proyecto para la generación de ficheros de mapeo. Luego esa línea pasa a ser el Departamento de Tecnología del centro CEIGE. En este departamento se desempeñó como desarrollador, luego pasó a ser Jefe de Equipo y actualmente se encuentra desempeñándose como Jefe del Proyecto Marco de trabajo para aplicaciones web de gestión Sauxe. Ha participado en eventos como la IX y X Semana Tecnológica organizadas por FORDES, en UCENCIA, Fórum de Ciencia y Técnica de la UCI, GESTEC 2010 en los cuales ha obtenido buenos resultados y las publicaciones de sus trabajos en cada uno de esos eventos. Ha estado vinculado además al desarrollo de otros productos del departamento como el sistema para el registro de las trazas.

AGRADECIMIENTOS

Agradezco el cumplimiento de este sueño:

A mi mamá, por ser mi musa, mi guía, mi más grande tesoro. A ti mami, por ser mi madre y mi padre, por ser la persona que más amo en el mundo.

A mi hermana porque la adoro, por ser la persona por la que quiero ser mejor cada día, para poder ser un digno ejemplo para ella.

A mi tato (Alfredo) y a mis tíos (Ramón, Porfirio, Carmen, Deciderio y Rafe) por haber sido como padres para mí, por estar siempre ahí para apoyarme en todo. A ellos más de mil gracias.

A Ruby y a Arturo porque sin su ayuda no estuviese hoy aquí, por su apoyo incondicional por su dedicación y su entrega, por siempre estar ahí cuando la necesité.

A Isabel por saber aceptarme y quererme como una hermana, por saber ser una buena amiga, por apoyarme en todo a pesar de no estar de acuerdo.

A Aurelio por su amor, dedicación y entrega.

A Grettel Marcos, a Anay y a Greter Izquierdo por regalarme su amistad y por tratar siempre de que sea una mejor persona.

A Omar, a Leo y a Yero no solo por ser mis amigos, sino por todo, por ayudarme siempre, porque sin ellos no lo hubiese logrado, por tenerme tanta paciencia.

A mis primos Doris y Rene por confiar en mí y guiarme siempre.

A mis compañeros del proyecto, de apto, a Yordanis y a todos los que de una forma u otra han estado cerca de mí.

A mis tutore y a todo aquel que de una forma u otra ayudó que mi estancia en esta universidad fuera la experiencia más linda que haya tenido jamás.

A TODOS MUCHAS GRACIAS.

DEDICATORIA

- A mi mamá y a toda mi familia por nunca haberme fallado, por su amor y cariño y por su apoyo incondicional.
- A mis amigas, amigos y a mi novio por ser especiales para mí, por siempre estar ahí.
- A la Revolución y a nuestro invaluable Comandante en Jefe.

RESUMEN

Los procesos de análisis de información para las trazas en el marco de trabajo Sauxe del Centro de Informatización de la Gestión de Entidades (CEIGE) no engloban una solución automatizada. Esto provoca demora en los servicios, pérdida de la información y que los reportes estadísticos no sean los más confiables en las entidades donde la gestión de estos procesos se lleva a cabo de forma manual o semiautomática.

Este trabajo tiene como objetivo principal el desarrollo de una herramienta para el análisis de trazas del marco de trabajo Sauxe, la cual permitirá agilizar los procesos de detección y rectificación de errores tanto de rendimiento como de excepciones. La herramienta Análisis de Trazas tendrá como entrada los datos almacenados con el sistema de Trazas de Sauxe, a partir de los cuales detectará posibles errores y soluciones a los mismos así como los responsables de brindar dichas soluciones.

PALABRAS CLAVE
Análisis de trazas, proceso de minería, sistemas basados en casos, trazas.

ABSTRACT

The processes of data analysis for traces in the framework SAUXE of Centre of Management Entities (CEIGE) do not cover an automated solution. This causes delay in services, loss of information and that statistical reports are not the most reliable in institutions where the management of these processes is carried out manually or semi-automatically. This work has as main goal the development of a tool for trace analysis of the framework SAUXE, which will streamline the process of error detection and correction of both performance and exceptions. The same will input the data stored in the Trace system SAUXE, from which detect possible errors and solutions to them as well as those responsible for providing such solutions.

KEY WORDS: case-based systems, process mining, traces, trace analysis.

ÍNDICE

INTRODUCCIÓN

En la actualidad, se ha vuelto indispensable el uso de las Tecnologías de Información y las Comunicaciones (TIC) que son el resultado de los avances en la microelectrónica, la informática y las telecomunicaciones. Las TIC se han introducido en cada área de la sociedad, y de manera especial en las empresas.

El avance de las tecnologías de la información ha generado una gran presión competitiva sobre las empresas, las cuales para sobrevivir deben focalizarse en las áreas centrales del negocio con agilidad y seguridad sin descuidar los objetivos primarios y la razón de ser, por esta razón la tendencia es cada vez mayor a la automatización de procesos.

Esos procesos se traducen a código fuente y luego se reproducen constantemente, la prueba de que estos procesos han ocurrido son las **trazas**. Estas son un mecanismo de registro de eventos y datos, o información sobre quién, qué, cuándo y dónde un evento ocurre, ya sea para un módulo del sistema en particular o para toda la aplicación. Las trazas son utilizadas para dejar un registro de todas las funciones que se realizan en el sistema, así como acreditar las validaciones de datos previstas, sin modificar el sistema en ningún momento. También son usadas con el fin de monitorear las acciones que se realicen (acceso a ficheros, dispositivos, empleo de los servicios, etc.), y para detectar indicios de hechos relevantes a los efectos de la seguridad que puedan afectar la estabilidad o el funcionamiento del sistema informático. Además rastrean los caminos que siguen los datos a través del programa. (1)
Las trazas se acumulan llegando a hacer casi imposible para un ser humano procesar toda la información que muestran. Para poder extraer información concreta surgen los sistemas de análisis de trazas.

La creación de software pasó de ser una simple producción artesanal a ser una producción masiva, en las empresas desarrolladoras de software han jugado un papel importante los marcos de trabajo. Los marcos de trabajo son: una *estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un marco de trabajo puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto*. (2)

Muchos de ellos cuentan con herramientas que permiten visualizar las trazas para así lograr un mejor complemento y facilitar el proceso de desarrollo en cuanto a reducción de tiempos y errores durante la fase de implementación de las soluciones.

En la Universidad de las Ciencias Informáticas (UCI) existe el Centro de Informatización de la Gestión de Entidades (CEIGE) el cual se dedica al desarrollo de aplicaciones web de gestión, para apoyar el desarrollo de las soluciones de dicho centro fue necesaria la implementación del marco de trabajo Sauxe que contiene un conjunto de componentes reutilizables que provee la estructura genérica para el desarrollo de aplicaciones web de gestión, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Este marco de trabajo se utiliza hoy en distintos proyectos como son: Cedrux (CEIGE), GINA (CEIGE), Generador Dinámico de Reportes GDR (DATEC), Sistema de Gestión Estadística (Datec), Minería de datos (Datec), Sistema de auditoría y control (SIGAC) del CAR (Controlaría general de la República)(Facultad 2), Fuerza de Trabajo Calificada del MEP (Ministerio de Economía y Planificación)(Facultad 1), UCID-FAR (Sistema de apoyo a la toma de decisiones SIMEM, Sistema de representación geoespacial LiberGIS, Sistema de supervisión y control de los PSI (Hoyo), Sistema para la gestión de las transportaciones en las FAR (DITRANS)), en entidades como: Aduana, TRANSOFT, DESOFT (Camagüey, Habana, Holguín, etc.), Centro de desarrollo de Holguín entre otras.

Sauxe cuenta además con un componente que se encarga del registro y seguimiento de los eventos que se ejecutan, denominado Trazas. En el componente Trazas el reconocimiento de incidencias se realiza de forma manual; lo que hace el proceso lento y con gran probabilidad de introducir errores humanos, omitiendo aspectos que son necesarios que se analicen como el rendimiento y las excepciones. Requiere además que existan personas calificadas para realizar las operaciones con el sistema, es decir que posean el conocimiento necesario para poder interpretar los resultados que se muestran.
Por otro lado es válido mencionar que las personas que operan con el componente de trazas no necesariamente son las que resuelven las anomalías, por lo que se extiende aún más el proceso de resolución. Por tanto se hace necesario el desarrollo de un componente que se encargue del análisis de las trazas para el marco de trabajo Sauxe, que permita agilizar el proceso de detección de anomalías en el sistema y la corrección de las mismas.

Según lo descrito anteriormente se define el **Problema de la investigación** de la siguiente manera:

¿Cómo agilizar el proceso de análisis de las trazas registradas por el marco de trabajo Sauxe para evitar retrasos en la resolución de las incidencias detectadas?

Se define como **Objeto Estudio** los sistemas basados en casos y como **Campo de acción:** los sistemas basados en casos para el análisis de trazas.

Para darle solución al problema se trazó como **Objetivo General:** Desarrollar una solución que permita el análisis de las trazas del marco de trabajo Sauxe y para darle solución a estos objetivo se trazaron los siguientes **Objetivos Específicos:**

- Realizar el Marco Teórico sobre el análisis de trazas en software de gestión.
- Realizar el Análisis y Diseño de la solución.
- Implementar la solución.
- Validar la solución.

La presente investigación tiene como **Idea a Defender**: Si se desarrolla una herramienta que permita el análisis de las trazas registradas por el marco de trabajo SAUXE se evitarán los retrasos en la resolución de las incidencias detectadas.

Los **Métodos investigativos** utilizados fueron:

Métodos Teóricos:

- Hipotético-deductivo.
- Análisis Histórico-lógico.

Métodos empíricos:

- Observación.

La estructura de la investigación está constituida de la siguiente manera:

- **Capítulo 1: Fundamentación teórica**

En este capítulo se define la base teórica necesaria para entender el entorno que rodea el problema a resolver. Se realiza el estudio del estado del arte y se describen los conceptos fundamentales, técnicas, tendencias, metodologías y herramientas utilizadas, fundamentando los elementos escogidos para el desarrollo del trabajo.

- **Capítulo 2: Propuesta de solución**

En este capítulo se detalla la propuesta de solución del componente siguiendo la metodología de desarrollo escogida. Figuran los artefactos: modelo conceptual, especificación de procesos y lista y descripción de los requisitos. Además se obtiene a partir del diseño, la arquitectura base, la descripción de las principales clases que fueron definidas, los diagramas de clases y se hace referencia a los principales patrones de desarrollo que fueron utilizados, así como el modelo de datos.

- **Capítulo 3: Implementación y prueba de la solución**

Contiene los elementos fundamentales referentes a la etapa de implementación según la metodología de desarrollo de software utilizada. Se hace referencia a los principales estándares de codificación para la nomenclatura de las clases, así como los artefactos generados durante esta fase. Se realiza además una validación de la solución mediante pruebas de concepto. Se explican cada una de las métricas utilizadas y se muestran los resultados de la implantación del componente.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En este capítulo se realiza un proceso investigativo de los aspectos teóricos necesarios para el conocimiento y la comprensión de la herramienta analizador de traza. Se realiza un análisis de sistemas existentes vinculados a los analizadores de trazas para evaluar sus ventajas y deficiencias. Se hace además un análisis de las herramientas, metodologías y tecnologías a utilizadas en el desarrollo del módulo que se pretende implementar.

1.2. Conceptos fundamentales

La herramienta del marco de trabajo Sauxe que se encarga del monitoreo y registro de los eventos que se ejecutan denominada Traza, necesita de un sistema que analice la información, básicamente que se encargue del análisis de las trazas por lo que es necesario usar un sistema de razonamiento basado en casos. Los sistemas basados en casos son un tipo de sistema experto desde el punto de vista de la inteligencia artificial, es un sistema que intenta imitar el comportamiento de un ser humano experto en alguna temática, es decir imitan las actividades de un ser humano para intentar resolver los problemas de esta índole. Para una mejor comprensión a continuación se enumeran los conceptos fundamentales.

1.2.1. Traza

Son rastros, serie de huellas o pista dejadas por el paso de un animal, persona u objeto. (3)

Log: Son archivos en los que se recogen las visitas que tienen las páginas de un sitio web. (4)

Luego de analizar los conceptos se puede decir que en la informática las trazas son una colección de eventos y datos que representan el rastro dejado por cierto proceso cuando se ejecuta en un sistema determinado.

1.2.2. Análisis de trazas

El análisis de trazas es el acto de separar las colecciones de eventos y datos que representan el rastro dejado por cierto proceso, para estudiar su naturaleza, su función y/o su significado y llegar a resultados más específicos.

1.2.3. Inteligencia Artificial

La Inteligencia Artificial es considerada una rama de la computación y relaciona un fenómeno natural con una analogía artificial a través de programas de computador. Puede ser tomada como ciencia si se enfoca hacia la elaboración de programas basados en comparaciones con la eficiencia del hombre, contribuyendo a un mayor entendimiento del conocimiento humano. Si por otro lado es tomada como ingeniería, basada en una relación deseable de entrada-salida para sintetizar un programa de computador."El resultado es un programa de alta eficiencia que funciona como una poderosa herramienta para quien la utiliza."A través de la inteligencia artificial se han desarrollado los sistemas expertos que pueden imitar la capacidad mental del hombre y relacionan reglas de sintaxis del lenguaje hablado y escrito sobre la base de la experiencia, para luego hacer juicios acerca de un problema, cuya solución se logra con mejores juicios y más rápidamente que el ser humano. (5)

1.2.4. Sistemas basados en casos

El RBC [1] representa un nuevo método para resolver problemas no estructurados en el cual el razonamiento se realiza a partir de una memoria asociativa que usa un algoritmo para determinar una medida de semejanza entre dos objetos. En este paradigma la base del comportamiento inteligente de un sistema radica en recordar situaciones similares existentes en el pasado. Debe destacarse que es una técnica en la cual la memoria se sitúa como fundamento de la inteligencia artificial y más concretamente de los sistemas basados en el conocimiento. (6)

1.3. Sistemas para el registro y monitoreo de trazas del marco de trabajo Sauxe

En el marco de trabajo Sauxe una vez que se produzca algún evento en el sistema se crea una traza y mediante publishers[2], se persisten sus datos (identificador de la traza, fecha y hora de generación, etc.) en la base de datos. Esta herramienta registra los eventos lo que permite corregir los problemas que se presenten. Estos eventos son:

[1] **(RBC)** Razonamiento Basado en Casos.
[2] **(publishers)** Publicador, clase que persiste las trazas.

- Inicio de una acción: Se dispara un evento al comienzo de una acción.
- Terminación de una acción: Se dispara un evento al finalizar una acción.
- Error en una acción: Se dispara un evento cuando en una acción ocurre un error.
- IoC Externo: Se dispara un evento cuando ocurre una integración entre diferentes subsistemas.
- IoC Interno: Se dispara un evento cuando ocurre una integración entre diferentes componentes de un subsistema.
- Excepciones: Se dispara un evento cuando ocurre una excepción en el sistema.
- Rendimiento: Es el tiempo de ejecución de una acción.
- Error IoC Externo: Se dispara un evento cuando ocurre un error en la integración entre diferentes subsistemas.
- Error IoC Interno: Se dispara un evento cuando ocurre un error en la integración entre diferentes componentes de un subsistema.

De forma general para el registro y monitoreo de trazas del marco de trabajo Sauxe se tiene que conocer qué eventos están ocurriendo en un determinado momento, así como cual es el usuario que lo está ejecutando. Esta información es vital para descubrir brechas de seguridad, para tener conocimiento de que acciones son las que más se ejecutan y cuales pueden provocar que el sistema colapse, permitiendo así que los equipos de desarrollo se enfoquen en optimizar la implementación de soluciones. (1)

1.4. Herramientas para el análisis de Trazas

1.4.1. ProM (Proceso de investigación de Minería)

El proceso de investigación de Minería se ocupa de la extracción de conocimiento acerca de un negocio o proceso (desde su proceso de ejecución de registros). Dicho proceso de minería se esfuerza en profundizar en diversas perspectivas, tales como el proceso (o el control de flujo), el rendimiento, los datos y la perspectiva de la organización.

ProM es un marco extensible que soporta una amplia variedad de técnicas de minería de proceso en forma de plug-ins[3]. Es independiente de cualquier plataforma, el marco ProM ha sido publicado bajo una licencia de código abierto.

Esta herramienta cuenta con plug-ins de análisis relativos a:

- La verificación de los modelos de procesos (por ejemplo, el análisis Woflan).
- La verificación de la Lógica Lineal Temporal (LTL) fórmulas en un tronco.
- La verificación de la conformidad entre un modelo de proceso dado y un registro.
- Análisis de rendimiento (análisis estadístico de base y análisis de rendimiento con un modelo de proceso dado).
- Por último, ProM aporta una gran variedad de filtros de registro, que son una valiosa herramienta para la limpieza de los registros, o sin importancia, los artefactos no deseados. (7)

1.4.2. IEEE Explore Automatizado de análisis de trazas de modelos de sistemas de eventos discretos

Se obtiene una visión modelista en el comportamiento dinámico de un complejo modelo estocástico de simulación discreta de eventos basado en el análisis de trazas. Se propone un algoritmo para distinguir progresivos del comportamiento repetitivo en un seguimiento y para extraer un fragmento mínimo progresivo de una traza. El problema de optimización combinatoria para la implícita reducción de la traza se resuelve en tiempo lineal con la programación dinámica. Se presentan y comparan varios aproximados y un método de solución exacta. La información sobre la operación de reducción, así como la reducción de trazas le permite a un modelador reconocer la presencia de ciertos errores e identificar sus causas. Hace un seguimiento por un error de modelado sutil en un modelo de fiabilidad de un sistema de servidor multi-clase, para ilustrar la eficacia de nuestro enfoque en la revelación de la causa de un efecto observado. La técnica propuesta se ha implementado e integrado en Traviando, que es un analizador de trazas para la depuración de los modelos estocásticos de simulación. (8)

1.4.3. Z-Insight-IBM MVS Sistema analizador de Trazas

[3] (**plug-in)** Es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

Z-Insight es una herramienta visual para la depuración y análisis de rendimiento en sistemas Z. Permite al usuario explorar, navegar y entender las trazas tomadas con IPCS Systrace, que puede contener los eventos del sistema y, posiblemente, de las aplicaciones (Java, C/C + +, CICS, DB2, o en otros entornos). La inspección de las trazas IPCS con las herramientas tradicionales de texto puede ser una tarea desalentadora, ya que pueden contener millones de eventos. Z-Insight proporciona una serie de nuevas técnicas para hacer esta tarea más fácil. En primer lugar, el usuario puede ver los patrones visuales en la traza, a un nivel detallado, así como a nivel general. Por ejemplo para el análisis de rendimiento, el usuario puede ver fácilmente los bucles como patrones visuales repetitivos, una intensa actividad en una región determinada, en un momento dado, es fácil de detectar. El usuario también puede rápidamente maximizar, minimizar o desplazarse a las áreas de interés.

En segundo lugar, el usuario puede aislar partes interesantes de la traza, por ejemplo, ejecución en un determinado ASID y Unidad de Trabajo, o la ejecución en un procesador específico. El usuario también puede tallar partes del espacio de ejecución mediante el uso de potentes funciones de búsqueda, lo que limita el tamaño de la traza de depuración. Una vista de las estadísticas permite al usuario ordenar eventos por diferentes criterios para que los puntos calientes sean fáciles de encontrar. Además, los eventos de excepción o de diagnóstico están claramente marcados.

Por último, este sistema tiene una secuencia de contexto gráfica novedosa que muestra el flujo de control. A modo de ejemplo, ayuda al usuario a comprender cómo una ejecución típicamente tiene una determinada clase de eventos, por ejemplo, spinlocks. Del mismo modo, nuestra técnica puede revelar el contexto de los acontecimientos frecuentes (puntos calientes), mostrando "¿cómo llegamos aquí?". Esta técnica se basa en un algoritmo que primero encuentra secuencias comunes, a continuación, construye un gráfico de secuencia, y finalmente, elimina las transiciones con menos frecuencia tomadas de este gráfico, que muestra solo el contexto de procedencia más esencial. (9)

1.4.4. Intel(R) Trace Analyzer and Collector

El procesador Intel (R) es un colector de seguimiento a la cabeza, el rastreo que realiza la colección de baja basado en eventos de seguimiento en las aplicaciones. Puede analizar los datos de seguimiento recogidos para las zonas activas de rendimiento y cuellos de botella. El producto es completamente seguro para subprocesos y se integra con C/ C + +, Fortran y los procesos y subprocesos múltiples con o

sin MPI[4]. Es compatible con la instrumentación binaria y el modo seguro. Además se puede consultar la programación del sistema y errores de MPI. El procesador Intel (R) Trace Analyzer proporciona una manera conveniente de controlar la aplicación actividades recogidas por el colector de seguimiento de Intel a través de pantallas gráficas. Usted puede ver el nivel de detalle deseado, identificar rápidamente los puntos críticos de rendimiento y cuellos de botella, y analizar sus causas. La herramienta está disponible en Linux y Microsoft Windows. **(10)**

Características principales

- Avanzada interfaz gráfica de usuario: amigable interfaz de usuario, con un nivel de alta escalabilidad, apoyo de archivo de seguimiento estructurado (STF.
- La agregación y filtrado: vistas detalladas del comportamiento de tiempo de ejecución agrupados por funciones o procesos.
- A prueba de errores de seguimiento: mejora de la funcionalidad, permitiendo terminar aplicaciones con detección de punto muerto.
- La función de comparación: comparar dos archivos de seguimiento y/o dos regiones (en uno o dos archivos de seguimiento).
- Contador de línea de tiempo: analizar los datos recogidos a través de la lucha contra el Rendimiento de las aplicaciones de interfaz de programación (PAPI) y OS módulos o mediante el uso manual de Intel Trace Collector AP.

1.4.5. Valoración del estado del arte

Con la investigación realizada se obtuvo como resultado, que cada uno de los sistemas estudiados gestionaba las trazas de forma diferente, la herramienta ProM es muy general lo que complica enormemente a la hora de adaptarlo al contexto de las aplicaciones de gestión, la aplicación de IEEE Explore no puede ser utilizada pues los sistemas ERP no poseen elementos discrecionales para la generación de sus trazas, sus eventos son un entramado continuo lo que da al traste con las metodologías convencionales para el tratamiento de las trazas como eventos discretos y esta herramienta es privativa. Otra de las herramientas estudiadas fue Zinsight-IBM MVS en el caso de este sistema la problemática se reduce a el manejo de las trazas de los sistemas IPCS los que son una limitante para los

[4] **(MPI)** Message Passing Interface es un Interfaz estandarizado para la realización de aplicaciones paralelas basadas en paso de mensajes.

demás sistemas que generan trazas y requieren de estos servicios. Además se estudio la herramienta Trace Analyzer and Collector que usa lenguajes potenciados por su desarrollo y las aplicaciones del centro no se encuentran entre la gama de los que esta aplicación soporta. Por lo antes expuesto se propone la construcción de una herramienta que potencie el análisis de las trazas para el Marco de Trabajo Sauxe.

1.5. Modelo de desarrollo

El desarrollo de la documentación se hará utilizando un modelo de desarrollo elaborado para el Centro de Informatización de la Gestión de Entidades (CEIGE), el mismo tiene su basamento en los principios y buenas prácticas de las metodologías ágiles estudiadas para garantizar rapidez y un buen funcionamiento en el desarrollo de los procesos. (11)

Este se caracteriza por ser:

Centrado en la arquitectura

La arquitectura determina la línea base, los elementos de software estructurales a partir de los elementos de la arquitectura de negocio. Interviene en la gestión de cambios y diseña la evolución e integración del producto. La arquitectura orienta las prioridades del desarrollo y resuelve las necesidades tecnológicas y de soporte para el desarrollo.

Orientado a componentes

Las iteraciones son orientadas por el nivel de significancia arquitectónicas de los componentes, los mismos son abstracciones arquitectónicas de los procesos de negocio y requisitos asociados que modelan, el componente es la unidad de medición y ordenamiento de las iteraciones.

Iterativo e incremental

Las iteraciones son planificadas y coordinadas con el equipo de arquitectura, los clientes y la alta gerencia. Cada iteración constituye el desarrollo de componentes, los cuales son integrados al término de la iteración, permitiendo de esta manera la evolución incremental del producto.

Ágil y adaptable al cambio

El desarrollo de las partes formaliza solamente las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas. Los clientes y funcionales están involucrados en el proyecto y poseen parte de la responsabilidad del éxito del mismo. Los cambios son conciliados semanalmente, discutidos y aprobados.

1.6. Tecnologías y herramientas para el desarrollo

El presente trabajo forma parte de un proceso productivo iniciado en el CEIGE en el cual se tomaron decisiones tecnológicas que involucran la utilización de tecnologías de código abierto (Open Source) para el desarrollo de sus productos. El desarrollo de las nuevas soluciones está determinado por estas decisiones.

1.6.1. Herramientas CASE

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Una innovación en la organización, un concepto avanzado en la evolución de tecnología con un potencial efecto profundo en la organización. Se puede ver al CASE como la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales. (12)

1.6.1.1. Visual Paradigm 6.4

Visual Paradigm es una herramienta UML libre y profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Soporta UML versión 2.1, permite modelado colaborativo con CVS y Subversión, generación de código, ingeniería inversa, generación de bases de datos (transformación de diagramas entidad-relación en tablas de la base de datos), importación y exportación a ficheros XML, distribución automática de diagramas, entre otras característica. (13)

1.6.2. Herramientas para el desarrollo colaborativo

1.6.2.1. Subversion 1.6.6

Subversión es un sistema de control de versiones libre y de código fuente abierto. Subversión maneja ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos. Subversión puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. Algunos sistemas de control de versiones son también sistemas de administración de configuración de software. (14)

1.6.3. Herramientas para el desarrollo

1.6.3.1. Apache 2.2.9

Apache es el servidor web por excelencia, su facilidad de configuración, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. Se ejecuta en gran cantidad de sistemas operativos, lo que lo hace prácticamente universal. Es una tecnología gratuita, open-source y altamente configurable de diseño modular por lo que resulta muy sencillo ampliar sus capacidades. Actualmente existen muchos módulos para Apache que son adaptables a este, y están disponibles para su instalación cuando sean necesarios. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor y es posible configurarlo para que ejecute un determinado script cuando esto suceda. (15)

1.6.3.2. PostgreSQL 8.3

Es un servidor de base de datos relacional, libre. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados, almacenamiento de objetos de gran tamaño. Se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y joins[5] de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Como toda herramienta de software libre PostgresSQL tiene entre otras ventajas que cuenta con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y algo muy importante es que dicha

[5] **(JOIN)**. En lenguaje SQL permite combinar registros de dos o más tablas en una base de datos relacional.

herramienta es multiplataforma. Además de sus ofertas de soporte, cuenta con una importante comunidad de profesionales y entusiastas de PostgreSQL de los que los centros de desarrollos pueden obtener beneficios y contribuir. Está disponible en casi cualquier Unix (34 plataformas en la última versión estable), y una versión nativa de Windows está actualmente en estado beta de pruebas. (16)

1.6.3.3. NetBeans 6.9

El NetBeans es un IDE, una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso. Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring. Modularidad: Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones. NetBeans contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permite al usuario comenzar a trabajar inmediatamente. (17)

1.6.4. Librerías y marcos de trabajo

1.6.4.1. Sauxe 1.5

El desarrollo de la solución se realizará utilizando el Sauxe como marco de trabajo, el cual fue desarrollado por el Departamento de Tecnología del CEIGE, el mismo contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. (18)

Sauxe presenta una arquitectura basada en componentes y una estructura en capas que contiene en sus capas superiores el patrón Modelo Vista Controlador (MVC) como se muestra en la figura 1.

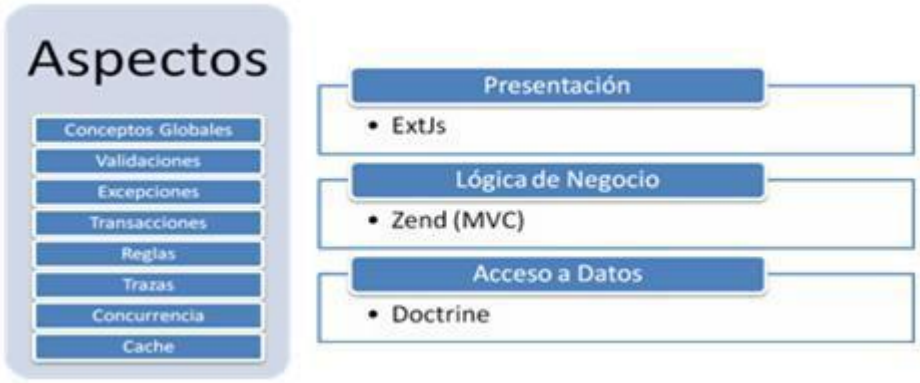

Figura 1. Arquitectura de Sauxe

Sauxe está compuesto por los siguientes marcos de trabajo:

1.6.4.1.1. ExtJS 2.2

Es una librería Java Script open-source de alto rendimiento para la creación y desarrollo de aplicaciones web dinámicas. Provee interfaces gráficas de usuario que brindan experiencias parecidas o iguales a las que se tienen con aplicaciones de escritorio. Permite la creación de aplicaciones complejas utilizando componentes predefinidos. Es extensible para la gran mayoría de los navegadores, evitando el tedioso problema de validar el código para cada uno de estos. Entre sus principales ventajas se encuentra el balance entre Cliente-Servidor, distribuyendo la carga de procesamiento en el último, y este al tener menor carga, maneja los clientes de manera más eficiente. La comunicación asíncrona permite el intercambio de información con el servidor sin necesidad de pedirle una acción al usuario, dando la libertad de cargar la información sin que este lo note. (19)

1.6.4.1.2. Zend Framework 1.7

Se trata de un framework para desarrollo de aplicaciones Web y servicios Web con PHP, te brinda soluciones para construir sitios web modernos, robustos y seguros. Además es Open Source y trabaja con PHP 5. (20)

Características:

- Trabaja con MVC (Modelo Vista Controlador).

- Cuenta con módulos para manejar archivos PDF, canales RSS, Web Services (Amazon, Flickr, Yahoo), etc.
- El Marco de Zend también incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar su base de datos, sin tener que escribir ninguna consulta SQL.
- Una solución para el acceso a base de datos que balancea el ORM[6] con eficiencia y simplicidad. Podemos ver esta función en el futuro.
- Un buscador compatible con Lucene.
- Robustas clases para autenticación y filtrado de entrada.
- Clientes para servicios web, incluidos Google Data APIs y StrikeIron. (20)

1.6.4.1.3. Doctrine 0.1

Doctrine es un sistema ORM (siglas en inglés de “mapeador relacional de objetos”) para PHP con un DBAL (siglas en inglés de “capa de abstracción de bases de datos”) incorporado que permite exportar una base de datos a sus clases correspondientes y viceversa, o sea, a partir de las clases creadas y siguiendo las especificaciones de ORM, generar las tablas de la base de datos. Se encuentra en la parte superior de una Capa de Abstracción de Base de Datos (DBAL). Una de sus principales características es la opción de escribir las consultas de base de datos en un objeto con una propiedad orientada al dialecto SQL llamada Doctrine Query Language (DQL), inspirada en Hibernate HQL. Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria. (15)

1.6.5. Lenguaje de desarrollo y modelado

1.6.5.1. Lenguaje de desarrollo

1.6.5.1.1. PHP 5.2.6

PHP (acrónimo de "PHP: Preprocesador de Hipertexto ") es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor, PHP es un soporte para una gran cantidad de bases de datos. Al nivel más básico, PHP puede hacer cualquier cosa que se pueda hacer con un script

[6] **(ORM)** Mapeador relacional de objetos

CGI, como procesar la información de formularios, generar páginas con contenidos dinámicos, o mandar y recibir cookies. Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir una interfaz vía web para una base de datos es una tarea simple con PHP, también soporta el uso de otros servicios que usen protocolos como IMAP, SNMP, NNTP, POP3, HTTP y derivados. (21). PHP es un lenguaje libre que puede ser utilizado en casi todos los sistemas operativos existentes.

1.6.5.1.2. JavaScript 1.8.2

JavaScript es un lenguaje de scripting basado en objetos, que se utiliza principalmente para crear páginas web dinámicas y permite el desarrollo de interfaces de usuario mejoradas. Una página web dinámica es aquella que permite la interacción entre el contenido de la misma y el usuario. JavaScript permite incorporar a dichas páginas efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. A pesar de su nombre, este no guarda relación directa con el lenguaje Java, sino que simplemente la compañía dueña del mismo lo adoptó por una cuestión de mercado. (15)

1.6.5.1.3. HTML 4.0

Es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

HTML también es usado para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML. (22)

1.6.5.2. Lenguaje de Modelado

1.6.5.2.1. UML 2.0

El proceso unificado de desarrollo (UML) representa un número de modelos de desarrollo basados en componentes que han sido propuestos en la industria. El proceso unificado define los componentes que se utilizarán para construir el sistema y las interfaces que conectarán los componentes. Utilizando una combinación del desarrollo incremental e iterativo, el proceso unificado define la función del sistema aplicando un enfoque basado en escenarios (desde el punto de vista del usuario) y acopla la función con un marco de trabajo arquitectónico que identifica la forma que tomará el software (23).

1.6.5.2.2. BPMN 1.0

Notación para el Modelado de Procesos de Negocio (BPMN), provee una notación estándar que es fácilmente legible y comprensible por todos los usuarios del negocio, presta apoyo a un modelo interno que permite la generación de ejecutables. BPMN crea un puente estandarizado para la diferencia entre los procesos de negocio, diseño e implementación. El diagrama de proceso de negocio (BPD) se basa en una técnica de diagrama de flujo adaptado para la creación de modelos gráficos de las operaciones de proceso de negocio. (24)

1.7. Conclusiones parciales

Luego de haber realizado una revisión de los sistemas de análisis de trazas existentes en el ámbito nacional e internacional; las ventajas y desventajas de las herramientas para el desarrollo de los subsistemas definidas para el análisis, se demuestra que las herramientas existentes no cumplen con las exigencias necesarias, por lo que se hace necesario la implementación de una aplicación que permita el análisis de las trazas para el marco de trabajo SAUXE. Dicha herramienta será implementada con las políticas de software libre a que se acoge el país y además cumplirá con las condiciones necesarias que exigen las entidades empresariales y unidades presupuestadas a escala nacional, por lo que se está en condiciones de describir y analizar la propuesta de solución para la implementación del subsistema.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

2.1. Introducción

En el siguiente capítulo se muestra la descripción de la propuesta de solución. Primeramente se describen cada uno de los procesos de negocio mediante los cuales se capturan los requisitos necesarios para la realización del sistema, se enumeran y especifican cada uno de ellos, tanto los funcionales como los no funcionales. Posteriormente se muestran los artefactos correspondientes al diseño de clases y modelo de datos. Por último se realiza la descripción del diseño de la solución que se elaboró para darle solución a los objetivos propuestos.

2.2. Modelo de dominio

El primer paso en el proceso de desarrollo de software es precisamente alcanzar cierto nivel de conocimientos sobre el problema en cuestión.

En el presente trabajo de diploma se intenta capturar los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema, por lo que decide realizar un modelo de dominio. Para conformar dicho modelo se ha elaborado un modelo conceptual que contiene los objetos y conceptos que se enmarcan en la problemática expuesta, así como un diccionario de datos

2.2.1. Modelo Conceptual

El Modelo Conceptual es una representación visual de los conceptos u objetos del mundo real, significativos para un problema o área de interés. El objetivo de la creación de este artefacto es aumentar la comprensión del problema y contribuir a esclarecer la terminología o nomenclatura del dominio. (25)

El modelo conceptual se representa usando el lenguaje de modelado (UML) como se muestra en la Figura 2, donde se observan conceptos del dominio y sus relaciones y atributos.

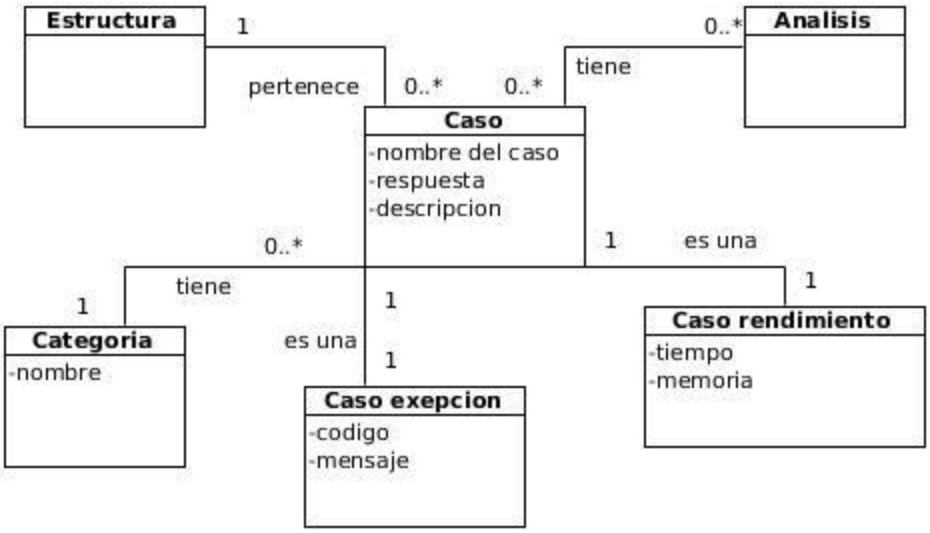

Figura 2: Modelo conceptual

2.2.2. Diccionario de Datos

El diccionario de datos contiene las características lógicas de las entidades y sus atributos que se van a utilizar en el sistema y que fueron contenidos en el modelo conceptual, incluyendo nombre de la entidad, descripción de la entidad, nombre del atributo, descripción del atributo, tipo del atributo, restricciones del atributo entre otras. A continuación se presenta el diccionario de datos con los conceptos que tienen mayor relevancia en el modelo, los demás se encuentran en el Anexo 1.

Nombre de la entidad	Caso.			
Descripción de la entidad	Entidad que representa los casos.			
Nombre del atributo	**Descripción**	**Tipo**	**¿Puede ser nulo?**	**¿Es único?**
Nombre	Atributo que identifica el nombre del caso.	Cadena de Caracteres.	No.	No.
Respuesta	Atributo que indica como se reacciona ante ese	Cadena de	No.	No

	tipo de caso.	Caracteres.		
Descripción	Atributo que representa una breve descripción del caso.	Cadena de Caracteres.	Si.	No.

Tabla 1 Diccionario de Datos para la entidad Caso

Nombre de la entidad	Caso Excepción.			
Descripción de la entidad	Entidad que representa los casos de excepciones específicamente.			
Nombre del atributo	**Descripción**	**Tipo**	**¿Puede ser nulo?**	**¿Es único?**
Código	Atributo que identifica un caso excepción.	Entero.	No.	No.
Mensaje	Atributo que representa una breve descripción de la excepción.	Cadena de Caracteres.	No.	No

Tabla 2 Diccionario de Datos para la entidad Caso Excepción

Nombre de la entidad	Caso Rendimiento.			
Descripción de la entidad	Entidad que representa los casos.			
Nombre del atributo	**Descripción**	**Tipo**	**¿Puede ser nulo?**	**¿Es único?**
Tiempo	Atributo que identifica un tiempo de ejecución de una acción.	Cadena de Caracteres.	No.	No.
Memoria	Atributo que representa el valor del consumo de memoria.	Cadena de Caracteres.	No.	No

Tabla 3 Diccionario de Datos para la entidad Caso Rendimiento

2.3. Modelación del negocio

2.3.1. Descripción de los procesos de negocio

Un proceso de negocio es un conjunto de tareas relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio definido. La descripción de los procesos de negocio hace más viable el paso a las actividades del análisis ya que posibilita una compresión más clara de los procesos en cuestión y contribuye a que los requisitos definidos satisfagan las necesidades del usuario. (23)
Teniendo en cuenta la opinión de especialistas y de los usuarios finales, se ha identificado un proceso de negocio. Este proceso es descrito a continuación.

2.3.1.1. Descripción del proceso: Análisis de traza

El proceso representado en la figura 3 muestra la detención de inconsistencias que se me muestran en la herramienta Traza.

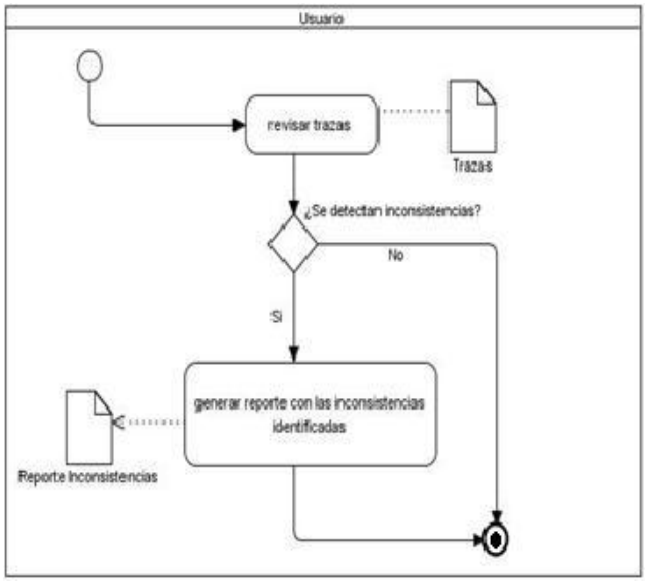

Figura 3: Proceso Analizar traza

El proceso inicia realizando un análisis profundo y detallado a cada uno de los registros mostrados por la herramienta Trazas para identificar anomalías como se muestra en la figura 3. Este proceso presenta una bifurcación que responde a la interrogante de: ¿Se detectaron inconsistencias?
En el caso de que existan inconsistencias se procede al siguiente flujo de actividades:

- Generar reporte con las inconsistencias identificadas: se estudia, valora y evalúa la inconsistencia para darle solución y se plasma un reporte donde se da un veredicto de la posible solución y su responsable.
- Luego se finaliza el proceso.

En caso de que no se identifiquen inconsistencias de precede a finalizar el proceso.

2.4. Requisitos de software

2.4.1. Requisitos funcionales

Los requisitos funcionales denotan una funcionalidad del sistema, son característica requerida que expresan una capacidad de acción del mismo o una funcionalidad; generalmente expresada en una declaración en forma verbal. (26)

Sirve de base para estimar el coste, el tiempo necesario para desarrollar el sistema y para planificar los contenidos técnicos de las iteraciones posteriores. A partir de la descripción de los procesos de negocios, se identificaron los requisitos a cumplir por el sistema, los cuales se muestran a continuación:

2.4.1.1. Lista de requisitos

R.1.1 Gestionar nomenclador categoría.

R.1.1. Adicionar categoría.

R.1.2. Modificar categoría.

R.1.3. Eliminar categoría.

R.1.4. Listar categoría.

R.2.1 Gestionar casos.

R.2.1. Listar casos rendimiento.

R.2.2. Listar casos excepción.

R.2.3. Adicionar casos rendimiento.

R.2.4. Adicionar casos excepción.

R.2.5. Modificar casos rendimiento.

R.2.6. Modificar casos excepción.

R.2.7. Eliminar casos rendimiento.

R.2.8. Eliminar casos excepción.

R.3.1 Realizar Análisis.

R.3.1. Realizar análisis para el rendimiento.

R.3.2. Realizar análisis para la excepción.

2.4.1.2. Especificación de requisitos

Para lograr una mejor comprensión de la funcionalidad asociada a cada proceso es necesario realizar una descripción textual de los requisitos. A continuación se describirán los requisitos principales del sistema.

Especificación del requisito Gestionar Casos

Conceptos tratados	**Conceptos**	**Atributos**
	Caso.	Denominación, estructura, categoría, respuesta, tiempo, memoria.
Precondiciones	**Precondiciones**	**Pre-requisito**
	No procede.	No procede.
Descripción	**Flujo Básico** 1. Se introducen los datos. 2. El sistema valida los datos introducidos. 2.1 El sistema señala los datos erróneos y permite corregirlos. 2.2 El usuario corrige los datos. 3. Si los datos son correctos el sistema los registra. 4. El sistema confirma el registro de los datos. 5. Concluye el requisito. **Pos-condiciones**	

	1. Se registró en el sistema un nuevo caso. **Flujos alternativos** No procede**.** **Pos-condiciones** No procede.
Validaciones	Se validan los datos según lo establecido en el Modelo conceptual.
Post-condiciones	Se muestra un listado actualizado de los casos de rendimiento.
Post-requisito	Listar casos de rendimiento.

Tabla 4 Especificación del requisito adicionar Casos Rendimiento

Conceptos tratados	**Conceptos**	**Atributos**
	Caso.	Id_casos, denominación, estructura, categoría, respuesta, tiempo, memoria.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir al menos un caso rendimiento registrado en el sistema.	No procede.
Descripción	**Flujo Básico** 1. Se selecciona el caso que se desea actualizar. 2. El sistema muestra y permite editar los datos del caso. 3. Se introducen los datos. 4. El sistema valida los datos introducidos. 4.1 El sistema señala los datos erróneos y permite corregirlos. 4.2 El usuario corrige los datos.	

	5. Si los datos son correctos el sistema los registra. 6. El sistema confirma el registro de los datos. 7. Concluye el requisito. **Pos-condiciones** **1.** Se registró en el sistema la modificación del caso. **Flujos alternativos** No procede **Pos-condiciones** No procede
Validaciones	Se validan los datos según lo establecido en el Modelo conceptual.
Post-condiciones	Se modifican los datos del caso. Se muestra un listado actualizado de los casos rendimiento.
Post-requisito	Listar casos rendimiento.

Tabla 5 Especificación del requisito Modificar Casos Rendimiento

Conceptos tratados	**Conceptos**	**Atributos**
		Id_casos, denominación, estructura, categoría, respuesta, tiempo, memoria.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir al menos un caso rendimiento registrado en el sistema.	
Descripción	**Flujo Básico** 1. Se selecciona el caso rendimiento a eliminar.	

	2. Se solicita confirmación para eliminar el caso rendimiento. 3. Si el usuario confirma se elimina el caso rendimiento. 4. El sistema no confirma la eliminación. 4.1. El usuario cancela la acción. 4.1.1. Concluye el requisito. 5. Concluye el requisito.
Validaciones	Se validan los datos según lo establecido en el Modelo conceptual.
Post-condiciones	Se eliminó el caso rendimiento. Se muestra un listado actualizado de los casos rendimiento.
Post-requisito	Listar casos rendimiento.

Tabla 6 Especificación del requisito Eliminar Caso Rendimiento

Conceptos tratados	**Conceptos**	**Atributos**
	Caso	Id_casos, denominación, estructura, categoría, respuesta, tiempo, memoria.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir al menos un caso rendimiento registrado en el sistema.	Adicionar casos.
Descripción	**Flujo Básico** 1. El sistema muestra un listado de los casos rendimiento. Se muestran denominación, estructura, categoría, respuesta, tiempo, memoria. 2. Concluye el requisito.	
Validaciones	No procede.	

Post-condiciones	Se muestra un listado de los casos de rendimiento.
Post-requisito	No procede.

Tabla 7 Especificación del requisito Listar Casos de Rendimiento

Conceptos tratados	**Conceptos**	**Atributos**
	Caso.	Denominación, estructura, categoría, respuesta, código, mensaje.
Precondiciones	**Precondiciones**	**Pre-requisito**
Descripción	**Flujo Básico** 1. Se introducen los datos. 2. El sistema valida los datos introducidos. 2.1 El sistema señala los datos erróneos y permite corregirlos. 2.2 El usuario corrige los datos. 3. Si los datos son correctos el sistema los registra. 4. El sistema confirma el registro de los datos. 5. Concluye el requisito. **Pos-condiciones** 1. Se registró en el sistema un nuevo caso. **Flujos alternativos** No procede**.** **Pos-condiciones** No procede.	
Validaciones	Se validan los datos según lo establecido en el Modelo conceptual.	

Post-condiciones	Se muestra un listado de los casos excepción.
Post-requisito	Listar casos Excepción.

Tabla 8 Especificación del requisito adicionar Casos Excepción

Conceptos tratados	**Conceptos**	**Atributos**
	Caso.	Id casos, denominación, estructura, categoría, respuesta, código, mensaje.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir al menos un caso Excepción registrado en el sistema.	
Descripción	**Flujo Básico** 1. Se selecciona el caso que se desea actualizar. 2. El sistema muestra y permite editar los datos del caso. 3. Se introducen los datos. 4. El sistema valida los datos introducidos. 4.3 El sistema señala los datos erróneos y permite corregirlos. 4.4 El usuario corrige los datos. 5. Si los datos son correctos el sistema los registra. 6. El sistema confirma el registro de los datos. 7. Concluye el requisito. **Pos-condiciones** **1.** Se registró en el sistema la modificación del caso. **Flujos alternativos**	

	No procede **Pos-condiciones** No procede
Validaciones	Se validan los datos según lo establecido en el Modelo conceptual.
Post-condiciones	Se muestra un listado de los casos excepción.
Post-requisito	Listar casos Excepción.

Tabla 9 Especificación del requisito Modificar Caso Excepción

Conceptos tratados	**Conceptos**	**Atributos**
	Caso.	Id casos, denominación, estructura, categoría, respuesta, código, mensaje.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir al menos un caso Excepción registrado en el sistema.	
Descripción	**Flujo Básico** 1. Se selecciona caso excepción a eliminar. 2. Se solicita confirmación para eliminar el caso excepción. 3. Si el usuario confirma se elimina el caso excepción. 4. El sistema no confirma la eliminación. 4.1. El usuario cancela la acción. 4.1.1. Concluye el requisito. 5. Concluye el requisito.	
Validaciones	Se validan los datos según lo establecido en el Modelo conceptual.	

Post-condiciones	Se muestra un listado de los casos de excepción.
Post-requisito	Listar casos Excepción.

Tabla 10 Especificación del requisito Eliminar Caso Excepción

Conceptos tratados	**Conceptos**	**Atributos**
	Caso.	Id casos, denominación, estructura, categoría, respuesta, código, mensaje.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir al menos un caso Excepción registrado en el sistema.	Adicionar casos.
Descripción	**Flujo Básico** 1. El sistema muestra un listado de los casos excepción. Se muestran denominación, estructura, categoría, respuesta, código, mensaje. 2. Concluye el requisito.	
Validaciones	No procede.	
Post-condiciones	Se muestra un listado de los casos excepción.	
Post-requisito	No procede.	

Tabla 11 Especificación del requisito Listar Casos Excepciones

Otros requisitos que no han sido contemplados como arquitectónicamente significativos son especificados en el Anexos 3.

2.4.1.3. Validación de los requisitos

Los requisitos luego de ser definidos necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de estos precisa realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de los requisitos sean correctos. Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación y muchas de ellas consisten en revisar los modelos obtenidos en la definición de requisitos con el usuario; para detectar errores o inconsistencias.

Aún así, existen algunas técnicas que pueden aplicarse para ello, como la Revisión técnica formal, las Listas de chequeo, las Auditorías, y los Prototipos.

De estas técnicas se empleó la **Revisión técnica formal** donde los analistas del subsistema luego de terminadas las especificaciones de requisitos realizaron la reunión de revisión, a la cual convocaron al cliente el Ing. Ariel Torres Gálvan, al Jefe del proyecto el Ing. René Bauta Camejo y la analista principal Ing. Lilian Alvares los cuales aprobaron y firmaron las especificaciones descritas sin posteriores modificaciones.

2.4.2. Requisitos no funcionales

Son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. (27)

Requisitos no funcionales de la solución propuesta:

- **Rendimiento**

Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de 5 segundos para las actualizaciones y 20 para las recuperaciones.

- **Seguridad**

- Autenticación (Contraseña de acceso).
- Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.
- La atención al sistema incluyendo, el mantenimiento de las bases de datos así como la salva de la información se realizarán de forma centralizada por el administrador.
- Verificación sobre las acciones irreversibles (eliminaciones).

- **Software**

La plataforma requiere que en el entorno de ejecución se encuentren instaladas y configuradas las siguientes aplicaciones:

- Servidor Web Apache versión 2.0 con soporte para PHP 5.2.
- Mozilla firefox versión superior a la 3.6.

- **Hardware**

En aras del mejor comportamiento de la plataforma se requiere que en el entorno de ejecución se encuentren disponibles las siguientes características:

- Procesador: 3.00 GHz.
- RAM: 1 Gb.
- Disco duro: 80 Gb.
- Con tarjeta de Red y Lector de CD.

2.5. Modelo de diseño

El Modelo de diseño es una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es utilizado como entrada esencial en las actividades relacionadas a la implementación. El Modelo de Diseño puede contener: diagramas, clases, paquetes, subsistemas, cápsulas, protocolos, interfaces, relaciones, colaboraciones y atributos. (28)

2.5.1. Diagrama de clases

Como parte del modelo de diseño se elaboraron los diagramas de clases.

Un diagrama de clases es un tipo de diagrama estático, que sirve para visualizar las relaciones y la descripción gráfica de las especificaciones de las clases del software y de las interfaces en una aplicación. Un diagrama de clases está compuesto por los siguientes elementos: clases, atributos, métodos y las relaciones entre ellos. (29)

A continuación se muestran el diagrama de clase basado en estereotipos web que fue confeccionado durante el proceso de desarrollo. Primeramente se muestra el diagrama con estereotipos web para el requisito Gestionar Casos.

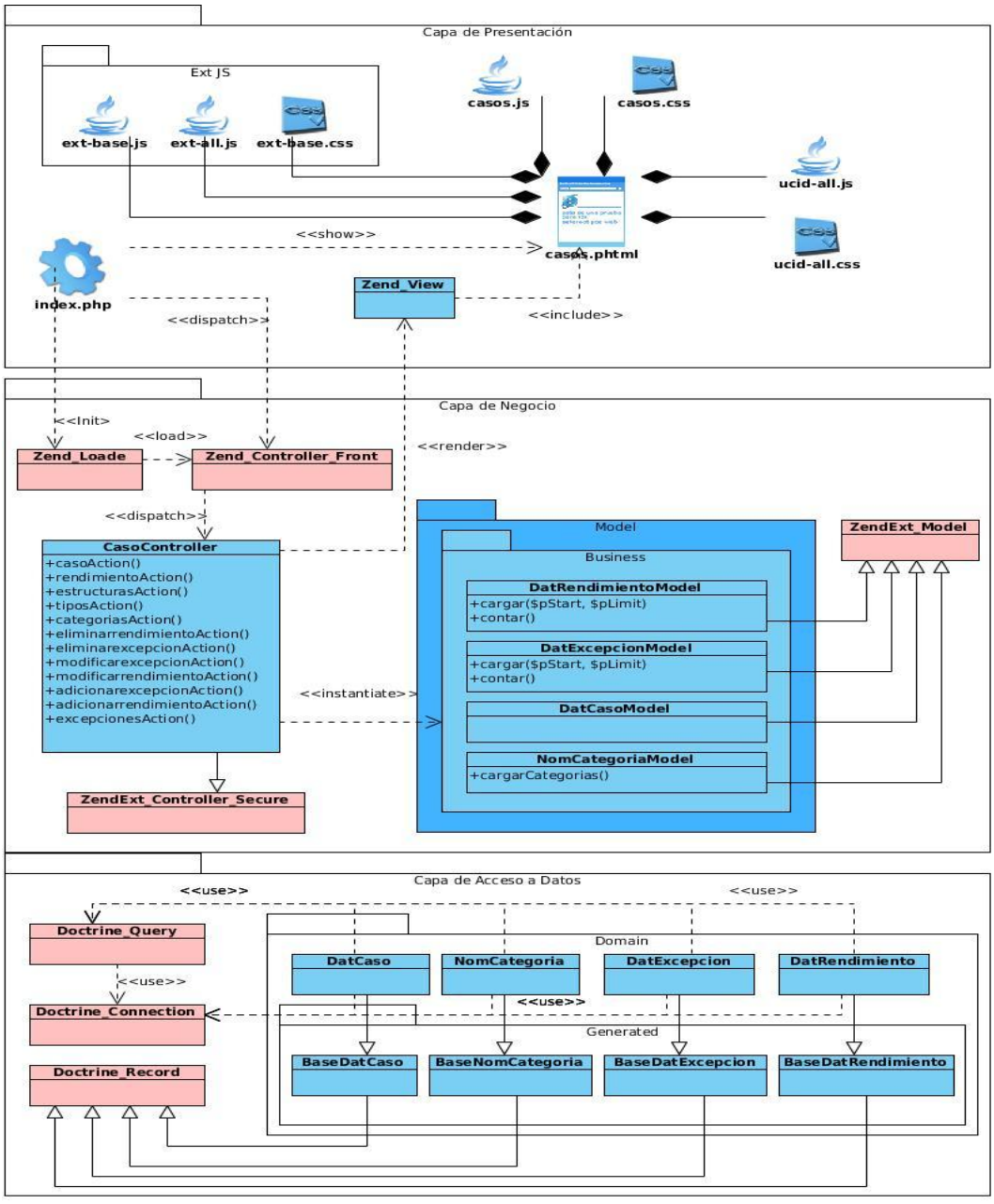

Figura 4 Diagrama de Clases del Diseño

2.5.2. Descripción de las clases

A continuación se describen las clases principales.

Nombre: CasoController.	
Tipo de Clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
estructurasAction()	Carga las estructuras.
tiposAction()	Responsabilidad que se encarga de cargar el grip correspondiente a cada tipo.
categoriasAction()	Responsabilidad que se encarga de llenar el combo con las categorías que están en la Base de Datos.
rendimientoAction() adicionarrendimientoAction() modificarrendimientoAction() eliminarrendimientoAction()	Responsabilidades que se encargan de realizar el CRUD para los casos de Rendimiento.
excepcionesAction() adicionarexcepcionAction()	Responsabilidades que se encargan de realizar el CRUD para los casos de Excepciones.

modificarexcepcionAction() eliminarexcepcionAction()	

Tabla 12 Descripción de la clase CasoController

Nombre: NomCategoriaModel.	
Tipo de Clase: Model.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
cargarCategorias() Insertar($Nomcategoria) Actualizar($Nomcategoria) Eliminar($Nomcategoria)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 13 Descripción de la clase NomCategoriaModel

Nombre: DatRendimientoModel.	
Tipo de Clase: Model	
Atributo	**Tipo**
Para cada responsabilidad	

Nombre.	**Descripción.**
cargar($pStart, $pLimit) Insertar($DatAgrupacionModel) Actualizar($DatAgrupacionModel) Eliminar($DatAgrupacionModel)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 14 Descripción de la clase DatRendimientoModel

Nombre: DatExcepcionModel.	
Tipo de Clase: Model.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
cargar($pStart, $pLimit) Insertar($DatAgrupacionModel) Actualizar($DatAgrupacionModel) Eliminar($DatAgrupacionModel)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 15 Descripción de la clase DatExcepcionModel

2.5.3. Patrones utilizados

2.5.3.1. Patrón arquitectónico Modelo Vista Controlador

El patrón de arquitectura conocido como Modelo-Vista-Controlador (MVC), separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario; es decir separa en tres capas diferentes los datos de una aplicación, la interfaz de usuario, y la lógica de control:

Modelo: Esta capa administra el comportamiento y los datos del dominio de la aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: Esta capa maneja la visualización de la información, es decir que presenta el modelo en un formato adecuado para interactuar, que usualmente es la interfaz de usuario.

Controlador: Esta capa controla el flujo de datos entre la vista y el modelo; es decir que responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista, tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. (30)

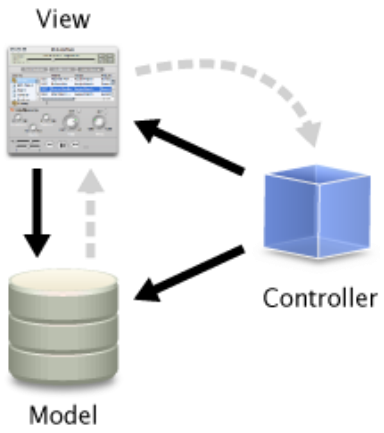

Figura 5: Patrón Modelo-Vista-Controlador

2.5.3.2. Singleton

Es un patrón creacional que tiene como propósito garantizar una única instancia de una clase, proporcionando un punto de acceso global a la misma. (31)
En la solución se hacen varios usos de este patrón, un ejemplo de estos lo constituye el IoC que es utilizado por los componentes para integrarse entre sí.

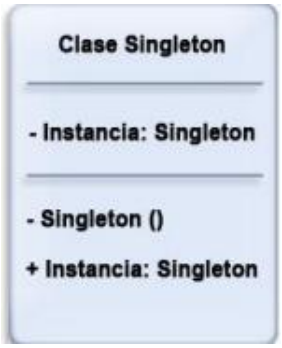

Figura 6 Patrón Singleton

2.6. Modelo de datos

Un modelo de datos es “n conjunto de conceptos, reglas y convenciones que nos permiten describir y en ocasiones manipular los datos de un cierto mundo real que deseamos almacenar en la base de datos”. (32)

El modelo de la solución cuenta con 3 clases persistentes dat_casos con los atributos (idcaso, denominación, respuesta, idestructuracomun y idcategoría que no es mas que la representación de la relación de uno a mucho con la tabla nom_categoria), dat_rendimiento con los atributos (idcaso siendo este una representación de la relación de uno a uno que posee esta tabla con la tabla dat_casos, tiempo y memoria) y dat_expeciones con los atributos (idcaso siendo este una representación de la relación de uno a uno que posee esta tabla con la tabla dat_casos, mensaje y código), exciten un total de 4 tablas. Para su confección se tuvieron en cuenta los procesos de normalización y la utilización de 1 nomencladores para facilitar su diseño.

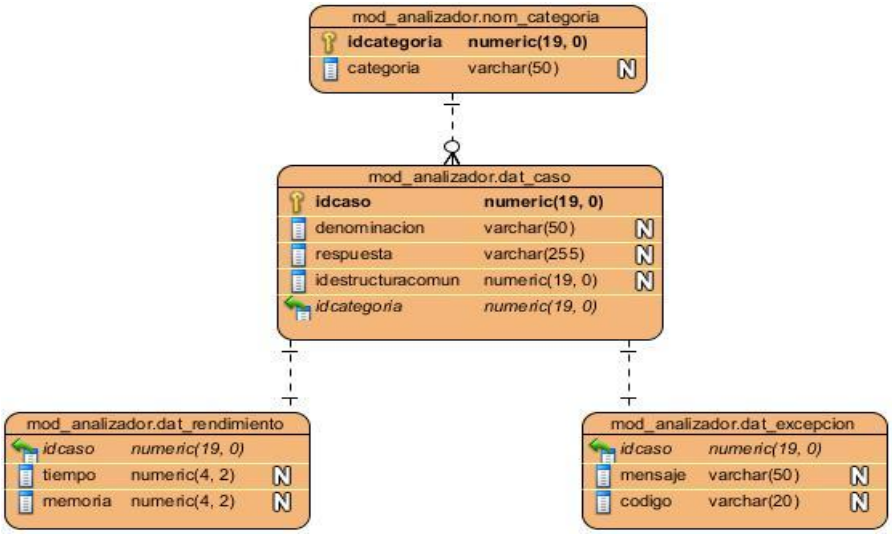

Figura 7 Diagrama del Modelo de Datos

2.7. Conclusiones Parciales

Con el desarrollo de este capítulo se conocieron los procesos objeto de automatización, lo que proporcionó una temprana idea de la complejidad de la solución. Se explicaron los componentes que fueron reutilizados para el desarrollo de la aplicación, teniendo en cuenta la arquitectura del sistema se conformó la arquitectura base que rige el diseño, el modelo de datos y el diagrama de clases del diseño. La descripción de clases importantes permitió tener una visión para la puesta en práctica de patrones. Se establecieron las bases para poder pasar a la implementación de la solución.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1. Introducción

Teniendo en cuenta el diseño de la solución realizado en el capítulo anterior, es necesario definir el modelo de implementación de la solución. Se da paso así al presente capítulo en el que se muestra dicho modelo de implementación y se especifica el conjunto de validaciones y pruebas que evalúan la calidad del sistema.

3.2. Modelo de Implementación

El modelo de implementación se inicia a partir de los resultados obtenidos en el diseño y describe cómo los elementos del modelo de diseño, se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje de programación utilizados, y cómo dependen los componentes unos de otros además de los recursos necesarios para poder ejecutar el sistema desarrollado. Estos conforman lo que se conoce como un modelo de implementación al describir los que se van a construir, su organización y dependencia entre nodos físicos en los que funcionará la aplicación.

3.2.1. Diagrama de Componentes

En el diagrama de componentes se muestran los elementos de diseño de un sistema de software. Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento de los servicios que estos componentes proporcionan y utilizan a través de las interfaces. (33)

La solución propuesta consta de 3 componentes llamados Casos, Nomencladores y Análisis que interactúan entre sí. A continuación se muestra el diagrama de componentes.

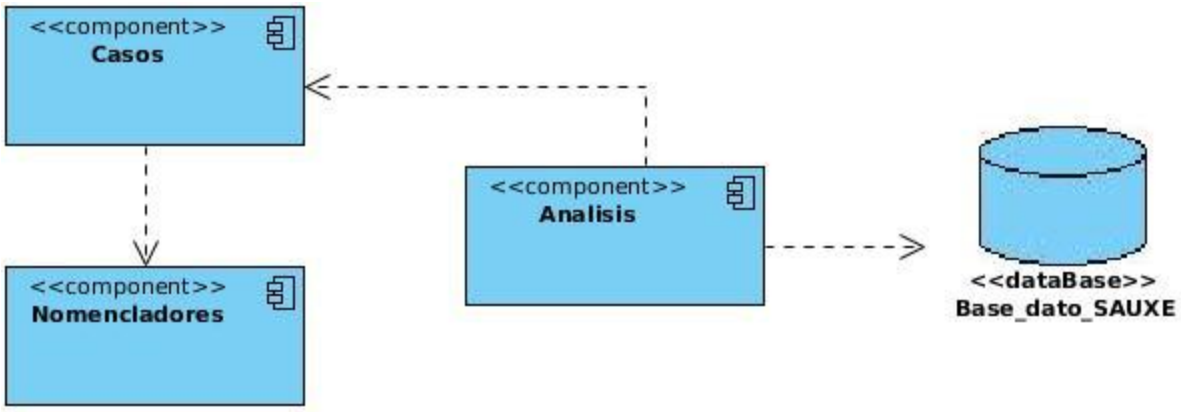

Figura 8 Diagrama de Componentes

Cada uno de los paquetes representados agrupa un conjunto de requisitos, a partir de los cuales se realiza el diseño de clases.

- **El paquete Nomencladores engloba los requisitos:**
  - Gestionar nomenclador categoría.
- **El paquete Casos engloba los requisitos:**
  - Gestionar agrupaciones.
  - Gestionar casos.
- **El paquete Análisis engloba los requisitos:**
  - Realizar Análisis.

3.2.2. Diagrama de Despliegue

El usuario, desde una estación de trabajo, podrá acceder al sistema el cual estará desplegado en el mismo servidor web donde se encuentre ubicada la aplicación a la cual se le desee generar servicios web. Dicho servidor estará conectado a un servidor de bases de datos en el cual se almacenará la información de interés para la solución. A continuación se muestra el diagrama de despliegue.

Figura 9 Diagrama de Despliegue

3.3. Normas y estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Constituyen una guía para el desarrollo en las líneas de producción, desde el punto de vista arquitectónico, con el propósito de lograr una estandarización del código. Permiten una mejor integración entre las líneas de producción y se establecerán las pautas que conlleven a lograr un código más legible y reutilizable, de tal forma que se pueda aumenta su mantenibilidad a lo largo del tiempo.

Los Estándares utilizados en la codificación fueron los siguientes:

- Notación Húngara**:** Esta convención se basa en definir prefijos para cada tipo de datos y según el ámbito de las variables. También es conocida como notación REDDICK (por el nombre de su creador). La idea de esta notación es la de dar mayor información al nombre de la variable, método o función definiendo en ella un prefijo que identifique su tipo de dato y ámbito.
- Notación PascalCasing: Es como la notación húngara pero sin prefijos. En este caso, los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula.

  Notación CamelCasing: Es parecido al Pascal-Casing con la excepción que la letra inicial del identificador no debe estar en mayúscula. (34)

3.3.1. Nomenclatura de las clases

1. Clases controladoras

Las clases controladoras después del nombre llevan la palabra:"Controller".

Ejemplo: CasoController

2. Clases de los modelos

- Business (Negocio)
  Las clases que se encuentran dentro de Business después del nombre llevan la palabra: "Model".
  Ejemplo: NomCategoriaModel
- Domain (Dominio)
  Las clases que se encuentran dentro de Domain reciben el nombre de la tabla en la Base de Datos.
  Ejemplo: NomCategoria
- Generated (Dominio bases)
  Las clases que se encuentran dentro de Generated el nombre comienza con la palabra: "Base" y seguido el nombre de la tabla en la Base de Datos.
  Ejemplo: BaseNomCategoria

3.3.2. Nomenclatura de las funciones

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing, y con solo leerlo se reconoce el propósito de la misma.

Ejemplo: insertarRendimiento

- En caso de ser una acción de la clase controladora se le pone el nombre y seguida la palabra:"Action".

  Ejemplo: insertarMonedaAction

3.3.3. Nomenclatura de las variables

El nombre de estas variables se escribe la primera palabra con minúscula, si es un nombre compuesto se utilizará notación CamelCasing donde se comienza con un prefijo.

Ejemplo: $objProceso.

Prefijos para los tipos de datos

Tipos de Datos	Prefijos
arreglos	arr
Objetos	obj
Enteros	int
Cadena	str
float	flt
Boolean	Boo

Tabla 16 Prefijos a utilizar en la creación de variables

3.3.4. Nomenclatura de los comentarios

Deben ser claros y precisos de forma tal que se entienda el propósito de los que se está desarrollando. Antes de declarar una clase se brinda una descripción de esta donde se explica el propósito de la misma y se escribe de la siguiente manera:

Ejemplo:**/***

*** Configurar casos Controller**

*** Clase Controladora para la configuración de los Casos.**

**** @package Traza**

*** @copyright UCID-ERP Cuba**

*** @autor: Mayara López Padrón**

*** @versión 1.0-0**

***/** (34)

3.4. Métricas de software

Las métricas de software son una medida cuantitativa que permite a los desarrolladores tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas, de manera que se puedan desarrollar los remedios y mejorar el proceso del software. (23)

Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:

Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.

Complejidad de implementación: Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.

Reutilización: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.

Acoplamiento: Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.

Complejidad del mantenimiento: Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.

Cantidad de pruebas: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, modulo, clase, conjunto de clases, etc.) diseñado.

Las métricas escogidas como instrumento para evaluar la calidad del diseño descrito en el capítulo anterior y su relación con los atributos de calidad son las siguientes:

3.4.1. Tamaño operacional de clase (TOC):

Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

Atributo de calidad	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 del grado de reutilización de la clase.

Tabla 17 Atributos de calidad evaluados por la métrica TOC

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Tabla 18 Criterios de evaluación para la métrica TOC

3.4.1.1. Resultados obtenidos de la aplicación de la métrica TOC

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 80% de las clases empleadas en el sistema posee 5 operaciones o menos, lo que conlleva a evaluaciones positivas

de los atributos de calidad involucrados (responsabilidad, complejidad de implementación y reutilización). A continuación se muestran los resultados obtenidos.

Tabla 19 Instrumento de evaluación de la métrica TOC

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
CasoController	13	Alta	Alta	Alta
DatCaso	1	Baja	Baja	Baja
DatExcepcion	1	Alta	Baja	Media
DatRendimiento	1	Alta	Baja	Media
NomCategoria	1	Media	Baja	Baja
DatCasoModel	0	Baja	Baja	Baja
DatExcepcionModel	2	Media	Media	Media
DatRendimientoModel	2	Media	Media	Media
NomCategoriaModel	1	Baja	Media	Media
AnalisisController	15	Alta	Alta	Alta

Tabla 19 Instrumento de evaluación de la métrica TOC

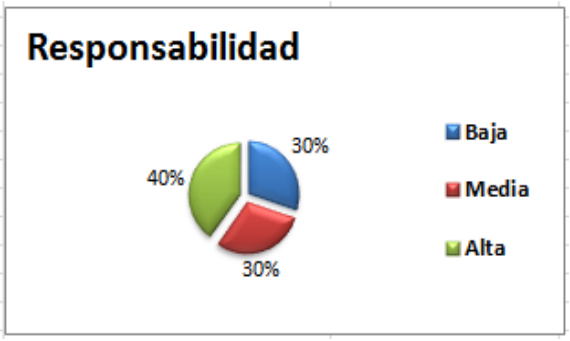

Figura 10 Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad

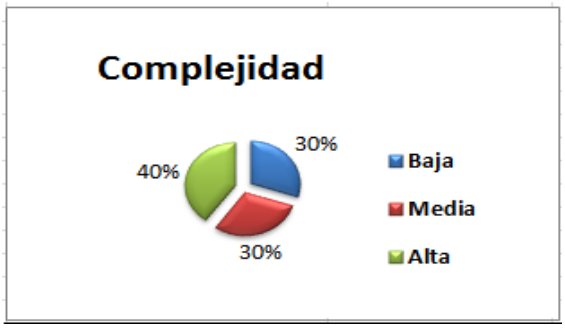

Figura 11 Resultados de la evaluación de la métrica TOC para el atributo Complejidad

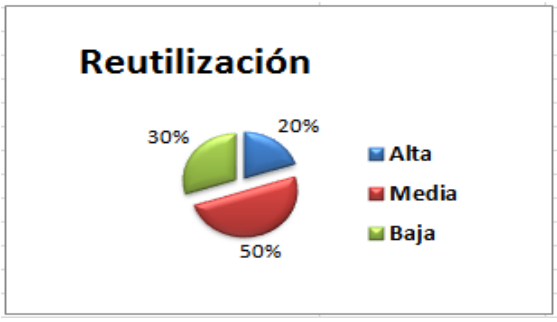

Figura 12 Resultados de la evaluación de la métrica TOC para el atributo Reutilización

3.4.2. Relaciones entre clases (RC):

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad de 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 20 Atributos de calidad evaluados por la métrica RC

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

Tabla 21 Criterios de evaluación para la métrica RC

3.4.2.1. Resultados obtenidos de la aplicación de la métrica RC

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 60% de las clases empleadas posee menos de 3 dependencias de otras clases, lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización). A continuación se muestran los resultados obtenidos.

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
CasoController	8	Alto	Alta	Baja	Alta
DatCaso	0	Ninguno	Baja	Alta	Baja
DatExcepcion	2	Medio	Baja	Alta	Baja
DatRendimiento	2	Medio	Baja	Alta	Baja
NomCategoria	0	Ninguno	Baja	Alta	Baja
DatCasoModel	0	Ninguno	Baja	Alta	Baja
DatExcepcionModel	1	Bajo	Baja	Alta	Baja
DatRendimientoModel	1	Bajo	Baja	Alta	Baja
NomCategoriaModel	1	Bajo	Baja	Alta	Baja
AnalisisController	3	Alto	Media	Media	Media

Tabla 22 Instrumento de evaluación de la métrica RC

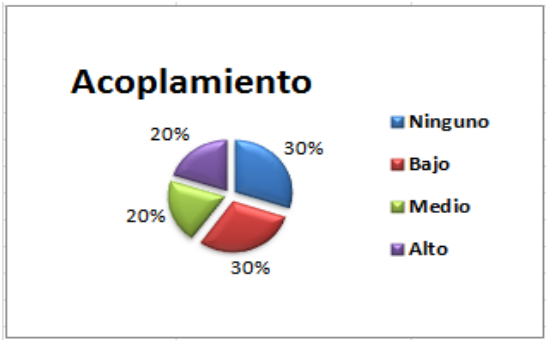

Figura 13 Resultados de la evaluación de la métrica RC para el atributo Acoplamiento

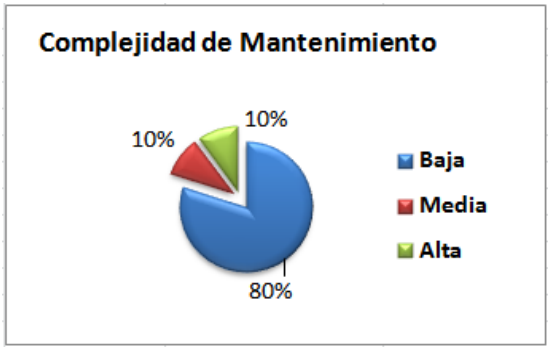

Figura 14 Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento

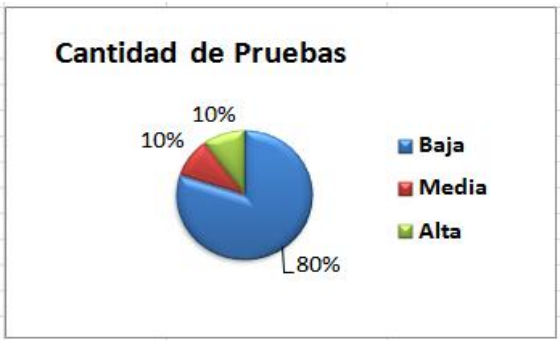

Figura 15 Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas

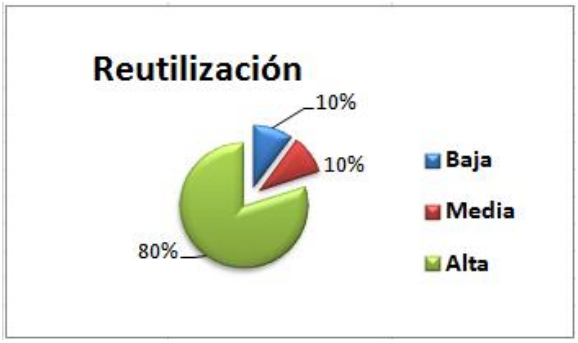

Figura 16 Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento

3.4.3. Matriz de inferencia de indicadores de calidad

Con la matriz inferencia de indicadores de calidad se representan los atributos de calidad y las métricas que se utilizaron para medir la calidad del diseño del componente. La matriz demuestra si los resultados de las relaciones entre atributos y métricas son positivos o negativos a partir de una escala numérica. En la escala si los resultados son positivos se le asigna el valor de 1, si son negativos toma valor 0 y si no existe relación es considerada como nula y es representada con un guión simple "-". Luego se puede obtener un resultado general para cada atributo promediando todas sus relaciones no nulas.

Categoría	Rango de valor
Malo	<=0.4
Regular	>0.4 y <0.7
Bueno	>=0.7

Tabla 23 Rango de valores para la evaluación de la relación Atributo/Métrica

		TOC	RC	Promedio
1	Complejidad Implementación	1	(-)	1
2	Reutilización	1	1	1
3	Acoplamiento	(-)	1	1
4	Complejidad Mantenimiento	(-)	1	1
5	Cantidad de Pruebas	(-)	1	1
7	Responsabilidad	1	(-)	1

Tabla 24 Resultados de la evaluación de la relación Atributo/Métrica

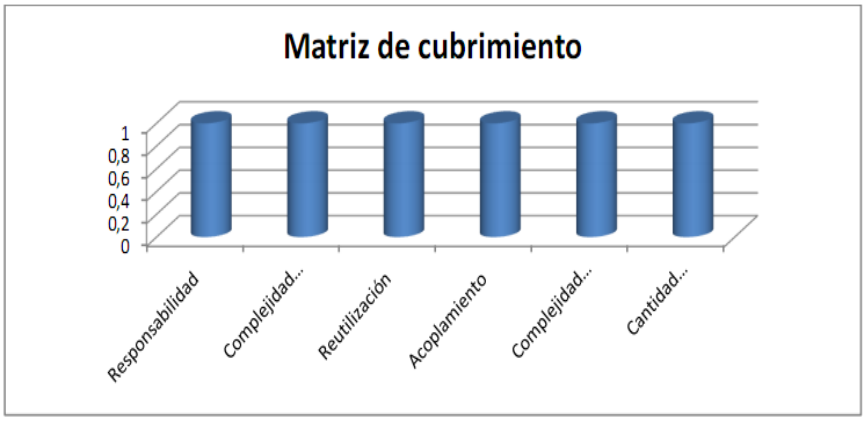

Figura 17 Resultados obtenidos de la evaluación de los atributos de calidad

3.5. Pruebas de Software

Existen dos enfoques principales para el diseño de casos de prueba:

1. El enfoque estructural o de caja blanca: que se basa en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.
2. El enfoque funcional o de caja negra: que realiza pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. (35)

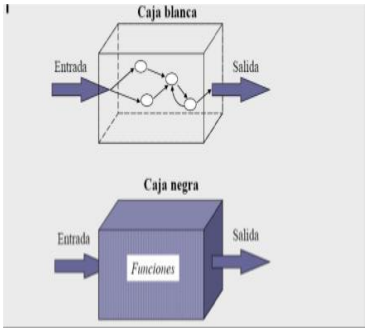

Figura 18 Representación de pruebas de Caja Blanca y Caja Negra

3.5.1. Pruebas de Caja Blanca

Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado o afirmado. Esto genera caminos posibles, por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar.

A continuación se muestra un ejemplo de las pruebas de caja blanca realizadas a nuestra aplicación, en el que para ejemplificar se escogió el método cargar de la clase DatRendimientoModel, lo primero es enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo correspondiente.

```
public function cargar($pStart, $pLimit) {

    $q = Doctrine_Query::create();       //1
    $q->from('DatRendimiento e')         //1
    ->offset($pStart)->limit($pLimit);   //1
     $exceptions = $q->execute ()->toArray();   //1
    $result = array ();                         //1

    foreach ($exceptions as $excepcion) {    //2
        $excepcion['categoria'] = Doctrine::getTable ('NomCategoria')->find ($excepcion ['idcategoria'])->categoria; //3
        $tmp = $this->integrator->metadatos->DameEstructura ($excepcion ['idestructuracomun']);  //3

        $excepcion['estructura'] = $tmp[0]->denominacion; //3
        $result [] = $excepcion;   //3
    } //4

    return $result;   //5
}
```

Figura 19: Código fuentes de la funcionalidad Insertar un cargar

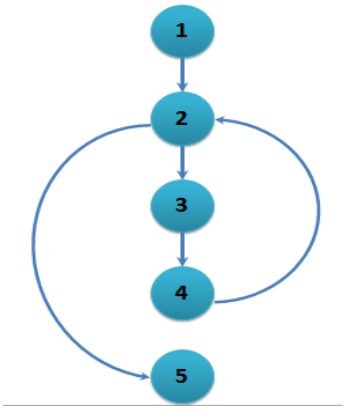

Figura 20 Grafo de flujo correspondiente a la funcionalidad cargar

Cálculo de la complejidad ciclomática

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.

Para efectuar el cálculo de la complejidad ciclomática del código es necesario tener varios parámetros como son la cantidad total de aristas del grafo 5, cantidad total de nodos 5, cantidad total de nodos predicados 1 y la cantidad total de regiones 2, para las siguientes fórmulas:

1. **V (G) = (A – N) + 2**

Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.

V (G) = (5 - 5) + 2

V (G) = 2

2. **V (G) = P + 1**

Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = 1 + 1 = 2

3. V (G) = R

Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del calculo.

V (G) = 2

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, dando como resultado 2, lo que indica que existen 2 posibles caminos por donde el flujo puede circular, y determina el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo:

- ✓ **Camino básico #1:**1-2-3-4-2-5
- ✓ **Camino básico #2:**1-2-5

Para cada camino se realiza un caso de prueba.

- ✓ **Caso de prueba para el Camino básico #1:** 1-2-3-4-2-5

  Descripción: Los datos de entrada se obtienen a través del método rendimientoAction () el que le da un valor a las variables limit y start.

  Condición de ejecución: Ningún dato debe ser nulo.

  Obtenido: $estructura = $estructura, $categoría= $categoría, $tiempo = $tiempo, $memoria = $memoria.

  Resultados esperados: Un arreglo de sistemas.

- ✓ **Caso de prueba para el Camino básico #2:** 1-2-5

Descripción: Los datos de entrada se obtienen a través del método rendimientoAction () el que le da un valor a las variables limit y start.

Condición de ejecución: Ningún dato debe ser nulo.

Obtenido: $estructura = $estructura, $categoría= $categoría, $tiempo = $tiempo, $memoria = $memoria.

Resultados esperados: No muestra nada.

3.5.2. Pruebas de Caja Negra

Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La misma no es una alternativa a las técnicas de prueba de caja blanca. Es un enfoque complementario.

Estas pruebas permiten encontrar errores tales como:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación. (35)

Nombre del requisito.	Descripción general.	Escenarios de pruebas.	Flujo del escenario.

1- Adicionar Caso Excepción.	El sistema debe permitir adicionar casos excepción.	EP 1.1: Adicionar caso excepción introduciendo datos válidos.	Se introducen los datos del caso excepción correctamente. Se presiona el botón Aceptar. Se muestra un mensaje de confirmación. Se presiona Aceptar. Se cierra la interfaz.
		EP 1.2: Adicionar caso excepción introduciendo datos válidos presionando el botón Aplicar.	Se introducen los datos de la correctamente. Se presiona el botón Aplicar. Se muestra un mensaje de información. La interfaz permanence abierta.
		EP 1.3: Adicionar caso excepción introduciendo datos inválidos.	Se introducen los datos inválidos de la distribución. Se presiona el botón Aceptar. Se muestra un mensaje informando del error.
		EP 1.4: Adicionar caso excepción dejando campos vacíos.	Se introducen los datos dejando algún campo en blanco. Se presiona el botón Aceptar. Se muestra un el campo

			encerrado en rojo y te señala que no debe dejar campos vacíos.
		EP 1.5: Cancelar.	Se introducen o no los datos del caso excepción. Se presiona el botón Cancelar. Se cierra la interfaz.

Tabla 25: Escenarios de prueba

No	Nombre de campo	Tipo	Válido	Inválido	Inválido	Descripción
1	Denominación.	Campo de texto.	Letras y números.	Caracteres especiales.	Vacío.	Se escoge el nombre del caso.
2	Categoría.	Combo.	Valor predeterminado.		Vacío.	Se escoge la categoría.
3	Código.	Campo de texto.	Letras y números.	Caracteres especiales.	Vacío.	Se escoge el código.
4	Mensaje.	Campo de texto.	Letras y números.	Caracteres especiales.	Vacío.	Se escoge el mensaje.
5	Estructura.	Árbol.	Valor predeterminado.		Vacío.	Se escoge la estructura.

6	Respuesta.	Area de texto.	Todo.		Vacío.	Se introducen las posibles respuestas.

Tabla 26: Descripción de Variables para el caso de prueba

Id EP	V1	V2	V3	V4	V5	V6	Respuesta del sistema	Resultado de la prueba
EP 1.1	V(Nom UNO)	V(Cat 1)	V(5)	V(Ocurrió un error)	V(Estructura de intalación)	Mandar mensaje.	El sistema almacena el proceso en base de datos. El sistema muestra un mensaje de información. 'El proceso fue insertado satisfactoriamente' y se cierra la interfaz.	
EP 1.2	V(Nom UNO)	V(Cat 1)	V(5)	V(Ocurrió un error)	V(Estructura de intalación)	Mandar mensaje.	El sistema almacena el proceso en base de datos. El sistema muestra un mensaje de información, El proceso fue insertado satisfactoriamente.	
EP 1.3	V(,)	V(Cat 1)	V(5)	V(Ocurrió un error)	V(Estructura de intalación)	Mandar mensaje.	El sistema muestra un mensaje de error, 'Datos incorrectos.'	
EP 1.4	Vacío	V(Cat 1)	Vacío	V(Ocurrió un error)	V(Estructura de intalación)	Mandar mensaje.	El sistema muestra un mensaje para que llene los campos.	

EP 1.5	Vacío.	Vacío.	Vacío.	Vacío.	Vacío.	Vacío.	Se cierra la ventana sin realizar ninguna operación.	

Tabla 27 Juego de datos a probar del requisito Gestionar

Resultados de las pruebas de caja negra

Para la realización de las pruebas de caja negra al sistema fueron analizados los casos de pruebas, de los requisitos del sistema. Para esto se realizó una descripción de diferentes escenarios de prueba, los cuales fueron entre 2 y 5 escenarios a probar, los mismos se probaron entre 1 y 4 juegos de datos. Atendiendo el comportamiento del sistema ante diferentes situaciones (entradas válidas y no válidas). Los errores que arrojaron el resultado de esta prueba fueron las faltas de ortografía en las interfaces y los mensajes de información o de errores, los cuales fueron corregidos para corroborar que ya cumplían con los resultados esperados. Se tomaron inicialmente 2 probadores, los cuales encontraron en una primera iteración 11 no conformidades, 3 en la aplicación y 8 en la documentación, las mismas fueron arregladas y en la segunda quedo liberada la aplicación.

3.6. Conclusiones Parciales

En el desarrollo de este capítulo se pudo apreciar la importancia que tiene el proceso de prueba en el desarrollo de software, sus objetivos y alcance. Se abordaron las métricas de software que fueron aplicadas en la solución y se determinó que el diseño de la implementación estuvo correcto, tal como fue mostrado por las métricas. Esta fase añade valor al producto, todos los programas poseen errores y la fase de pruebas los descubre, para lograrlo se analizaron diferentes aspectos como el significado de las pruebas de software se realizó una descripción de las pruebas de unidad, dentro de los cuales se analizaron los tipos de prueba: Caja Blanca y Caja Negra, y la ejecución de una de ellas, analizando los resultados obtenidos.

CONCLUSIONES

La investigación realizada en el presente trabajo de diploma permite concluir que la gestión manual de los archivos de configuración de la gestión de trazas trae consigo consecuencias desfavorables para la mejora de la resolución de anomalías tanto para el rendimiento y las excepciones en el marco de trabajo Sauxe del proyecto Sistema Integral de Gestión Cedrux.

A raíz de lo antes planteado con la culminación de la presente investigación se arriba las siguientes conclusiones:

- La herramienta obtenida permitirá a los equipos de desarrollo del centro, agilizar los procesos de rectificación de incidencias, lo cual acorta el tiempo de desarrollo de las mismas, logrando de esta manera que los desarrolladores se enfoquen en optimizar las soluciones.
- La recolección y posterior análisis de las trazas a largo plazo puede ser una fuente de información así como un aval en si mismo del cumplimiento de los procesos, además de un mecanismo de validación de los requisitos tanto funcionales como no funcionales de los sistemas construidos por Sauxe.
- Los resultados arrojados por las métricas orientadas a clases permitieron concluir que el diseño presentaba valores positivos en indicadores de calidad tales como Reutilización, Facilidad de Mantenimiento, Complejidad del Diseño, Complejidad de Implementación.
- Las pruebas estructurales y funcionales realizadas a la aplicación, validaron el correcto funcionamiento de la misma.

RECOMENDACIONES

Se considera importante que una vez concluido el proceso de desarrollo se lleven a cabo las siguientes recomendaciones.

La construcción de una segunda versión donde se incorporen las siguientes funcionalidades.

- Incorporar a la gestión del análisis de las trazas los demás tipos de trazas, como Integración, Excepción de Integración y Datos.

La implementación de algoritmos más eficientes para el análisis de trazas**.**

BIBLIOGRAFÍA

1. **Galvez, René R. Bauta Camejo y Ariel Torres.** *HERRAMIENTA PARA EL REGISTRO Y MONITOREO DE TRAZAS EN EL SISTEMA DE GESTIÓN INTEGRAR CEDRUX.* Habana : s.n., 2010.

2. Universidad de Belgrano. [En línea] [Citado el: 12 de 3 de 2011.] http://www.ub.edu.ar/.

3. Dictionary.com . [En línea] [Citado el: 10 de 3 de 2011.] http://dictionary.reference.com/.

4. Definición.org. [En línea] [Citado el: 10 de 3 de 2011.] http://www.definicion.org/diccionario/.

5. **Huerta, Hugo Vega.** Inteligencia Artificial 2011. [En línea] [Citado el: 21 de 4 de 2011.] 2011.

6. Monografias.com. [En línea] [Citado el: 10 de 3 de 2011.] www.monografias.com/.

7. **Bose, Jagadeesh Chandra.** *ProM 6 Tutorial.*

8. IEEE Explore. [En línea] [Citado el: 1 de 5 de 2011.] ieeexplore.ieee.org.

9. IBM. [En línea] [Citado el: 5 de 5 de 2011.] https://researcher.ibm.com/.

10. Intel. [En línea] [Citado el: 19 de 5 de 2011.] software.intel.com.

11. **Sarduy Pérez, Mileidy Magalys y Hernández Cisneros, Sergio.** *Propuesta de modelo de desarrollo de software tecnológico del Centro de Soluciones de Gestión.* Ciudad de la Habana : s.n., 2009.

12. Scribd. [En línea] [Citado el: 9 de 3 de 2011.] http://es.scribd.com.

13. **Prada Nicot, Héctor y Sánchez González, Kenner.** *Desarrollo de los componentes Puesto de Trabajo y Pagos Adicionales del subsistema Capital Humano integrado al sistema integral de gestión CEDRUX.* La Habana : s.n., 2009.

14. **Fogel, Karl.** Subversion. [En línea] 14 de 3 de 2004. [Citado el: 9 de 3 de 2011.] http://svnbook.spears.at/nightly/es/svn-book.html#svn-ch-1-sect-1.

15. **Javier Ruiz Durán, Eyonys González Marcaida.** *Diseño e implementación de una herramienta para la gestión de servicios web sobre aplicaciones PHP.* Ciudad de la Habana : s.n., 2010.

16. *Manual de usuario de Postgres.* 2008.

17. **Peña, Omar A. Díaz.** *Diseño arquitectónico de una plataforma para la arquitectura distribuida en PHP basada en Servicios Web.* Ciudad de la Habana : s.n., 2010. pág. 18 y 19.

18. **Gómez Baryolo, Oiner, Tenrero Cabrera, Marianela y Nemuris, Silega Martínez.** *Plantilla Registro de la Propiedad intelectual (Sauxe).* La Habana : s.n., 2008.

19. **Blades, Steve,Nigel, Ramsay, Colin y Frederick, Shea.** *Learning Ext JS.* 2008.

20. **Leopoldo, Carlos.** Zend Framework, una introducción. [En línea] 27 de noviembre de 2007. http://techtastico.com/post/zend-framework-una-introduccion/.

21. **otros, Stig Sæther Bakken y.** *Manual de PHP.* 2001.

22. HTML Castellano. [En línea] 2009. http://www.programacion.com/html/..

23. **Pressman, Roger S.** *Ingeniería de software.Un enfoque practico.* 2005. pág. 28.

24. **Corporation, Stephen A. White IBM.** *Introduction to BPMN.* 2044.

25. **Cruz, Betancourt.** *Procedimiento para desarrollar la Ingeniería de Requisitos en el proyecto Sistema de Gestión Penitenciaria.* Habana : s.n. págs. 4-18.

26. Tecnología y Synergix. [En línea] [Citado el: 25 de 5 de 20011.] http://synergix.wordpress.com.

27. **Glez, Anaisa Hernandez.** *REQUISITOS A PARTIR DEL MODELO DEL NEGOCIO.* 2005.

28. MeRinde. [En línea] [Citado el: 6 de 3 de 2011.] http://merinde.net/index.php?option=com_content&task=view&id=94&Itemid=296.

29. **Rosa, Luis Antonio.** *Fundamentos POO, Diagramas de Caso de Uso y Diagramas de Clase.* 2011.

30. **.NET, Colectivo de Desarrollo.** *Arquitectura y Patrones de diseño.* 2008-2009.

31. PATRON DE DISEÑO SINGLETON . [En línea] [Citado el: 1 de 6 de 2011.] wwwsingleton.blogspot.com/.

32. **Duque, Raúl González.** Mundo geek. [En línea] [Citado el: 6 de 3 de 2011.] http://mundogeek.net/archivos/2004/08/26/modelo-de-datos/.

33. MSDN. [En línea] [Citado el: 1 de 6 de 2011.] http://msdn.microsoft.com/.

34. **ERP, Proyecto.** *Normas y Estandares de codificación.* Habana : s.n., 2008.

35. **Huanca, Daniel.** Herramientas de Prueba de Software. [En línea] [Citado el: 29 de 5 de 2011.] http://herrorsoft.zxq.net/.

ANEXOS

Anexo 1: Diccionario de Datos

Nombre de la entidad	Análisis.			
Descripción de la entidad	Entidad que representa el análisis a cada una de los registros de Traza.			
Nombre del atributo	**Descripción**	**Tipo**	**¿Puede ser nulo?**	**¿Es único?**

Tabla 28 Diccionario de Datos para la entidad Análisis

Nombre de la entidad	Categoría.			
Descripción de la entidad	Entidad que representa a un nomenclador con las categorías que van a existir.			
Nombre del atributo	**Descripción**	**Tipo**	**¿Puede ser nulo?**	**¿Es único?**
Nombre	Atributo que identifica el nombre de la categoría.	Cadena de Caracteres.	No.	No.

Tabla 29 Diccionario de Datos para la entidad CasoCategoría

Nombre de la entidad	Estructura.			
Descripción de la entidad	Entidad que muestra la estructura que tiene el proyecto.			
Nombre del atributo	**Descripción**	**Tipo**	**¿Puede ser nulo?**	**¿Es único?**

Tabla 30 Diccionario de Datos para la entidad CasoEstructura

Anexo 2: Prototipos de Interfaz visual

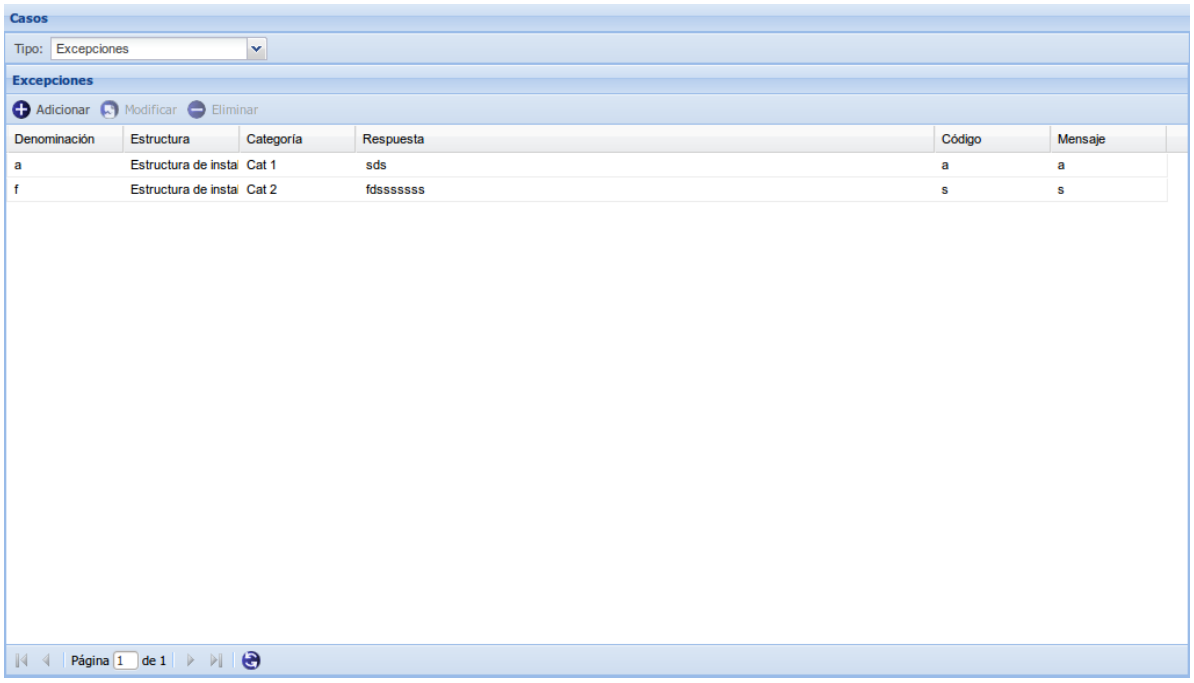

Figura 21 Prototipo de interfaz principal para los casos de Excepción

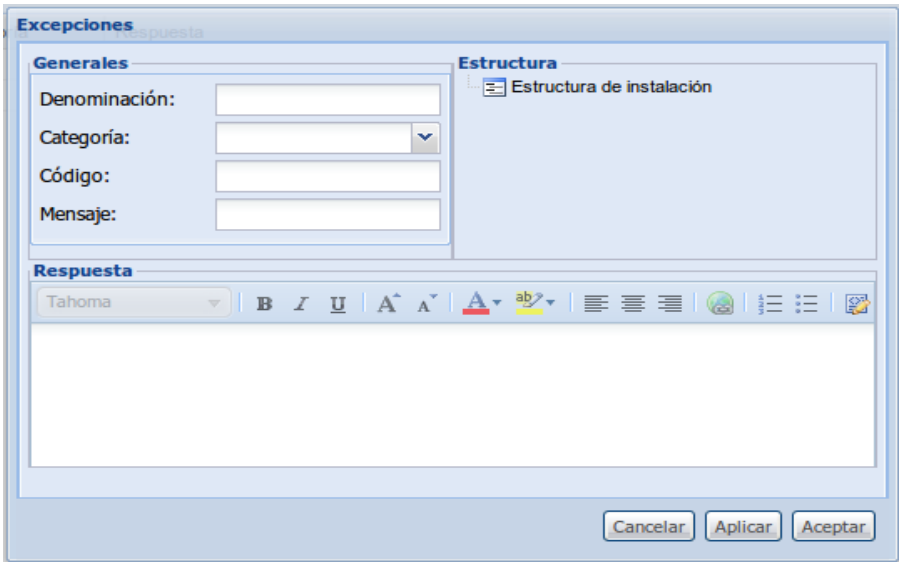

Figura 22 Prototipo de interfaz para adicionar casos de Excepción

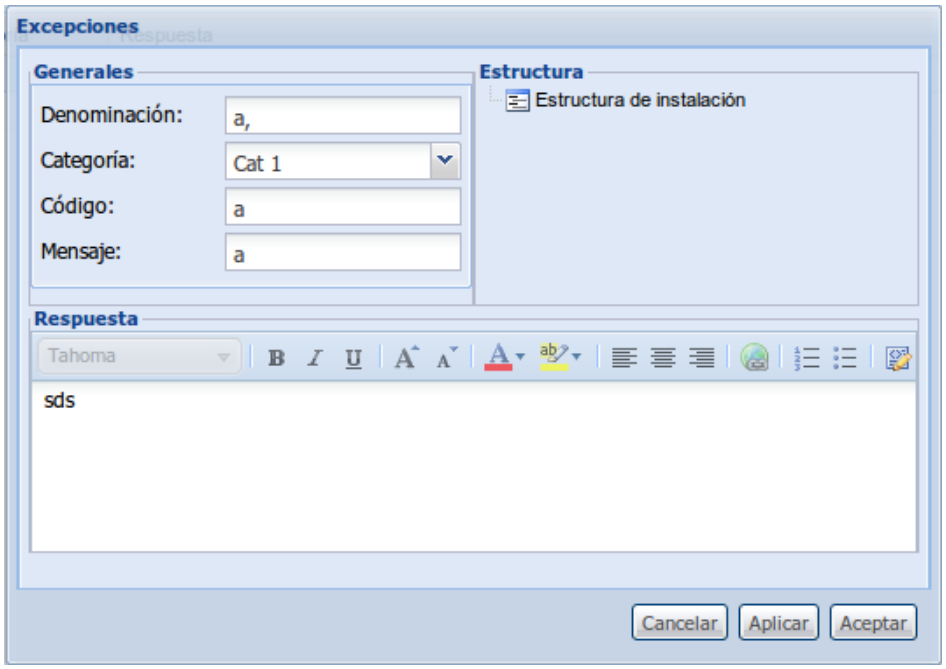

Figura 23 Prototipo de interfaz para modificar casos de Excepción

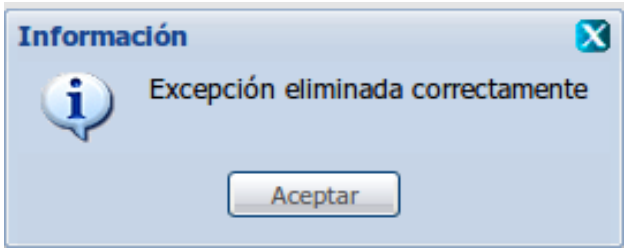

Figura 24 Prototipo de interfaz para eliminar casos de Excepción

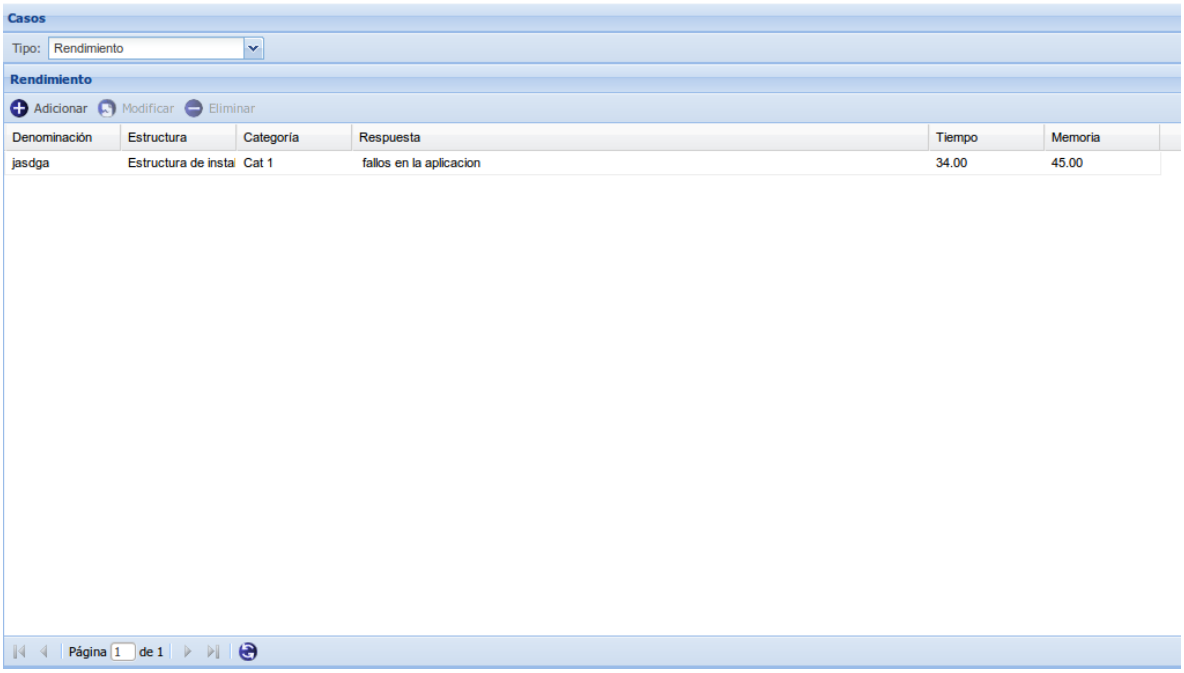

Figura 25 Prototipo de interfaz principal para los casos de Rendimiento

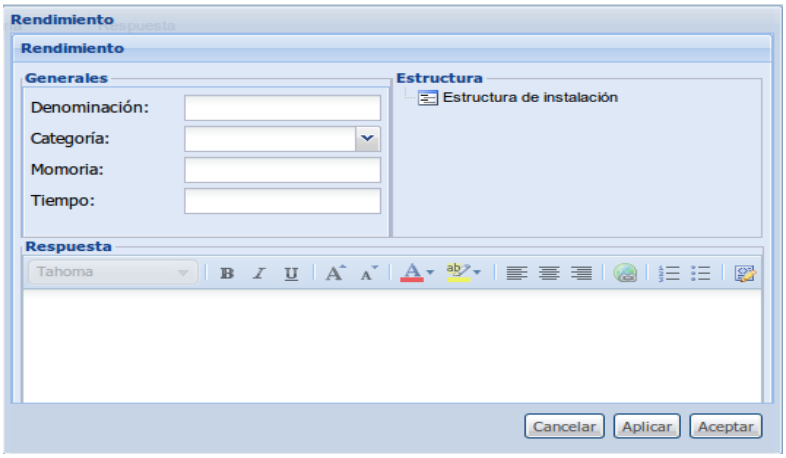

Figura 26 Prototipo de interfaz para adicionar casos de Rendimiento

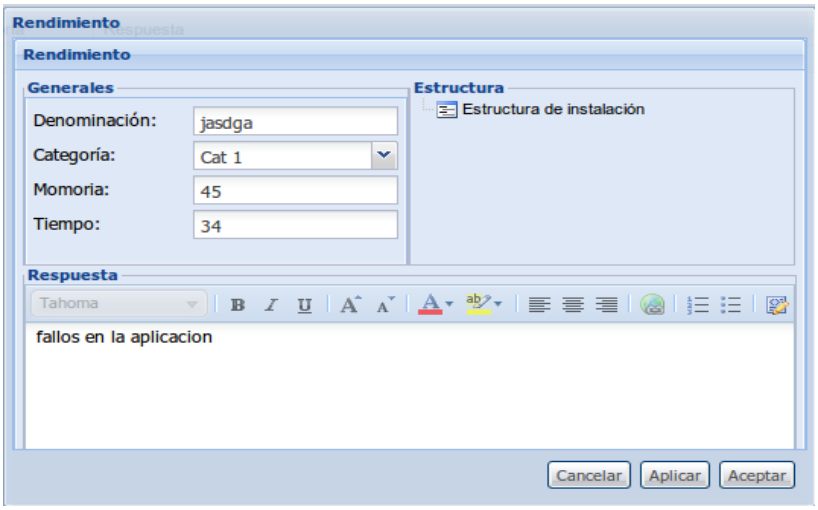

Figura 27 Prototipo de interfaz para modificar casos de Rendimiento

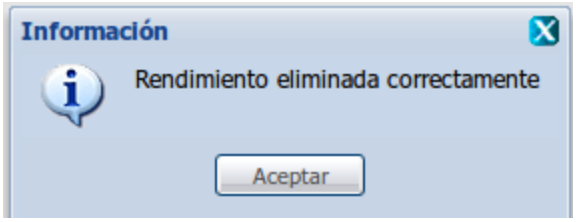

Figura 28 Prototipo de interfaz para adicionar casos de Excepción