Universidad de las Ciencias Informáticas
Facultad 3

Componente para el registro y monitoreo de accesos a la base de datos en el marco de trabajo Sauxe.

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autor: Yaniel Yero Cardoso
Tutores: MsC. Oiner Gómez Baryolo
Ing. René Bauta Camejo

La Habana, 24 de junio de 2011

"...Sobre todo, sean siempre capaces de sentir en lo más hondo cualquier injusticia cometida contra cualquiera en cualquier parte del mundo. Es la cualidad más linda de un revolucionario..."

Despedida del Che a sus hijos.

DECLARACIÓN DE AUTORÍA

Yo, Yaniel Yero Cardoso, con CI: 88013121625, declaro que soy el único autor de este trabajo y autorizo al Centro de Informatización de la Gestión de Entidades (CEIGE) de la Universidad de las Ciencias Informáticas (UCI) a hacer uso del mismo en su beneficio.

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

_____	_____	_____
Yaniel Yero Cardoso	MsC. Oiner Gómez Baryolo	Ing. René Bauta Camejo
Firma del Autor	Firma del Tutor	Firma del Tutor

AGRADECIMIENTOS

Son muchas las personas que me han ayudado, tanto en la realización de este trabajo como en mi tránsito por esta maravillosa universidad. A todos ellos: GRACIAS.

A mis profesores Esther Cristina, Dariela, Leidily, Yaima, Yoan, Navarrete, Marbis, Merlis, Leonardo, Yisel, Chacón, Osmín, Zulueta, Yudel, Daniel, Lieen, Yaniselis, Yoandris, Pascual, Ruben y Daimi. También a las instructoras Lidia Odalis y Mayra.

A mis compañer@s de proyecto, tanto profesionales como estudiantes Javier, Oiner, los René, Roberto, Omar, Dalinda, Ariel, Rogsany, Oscar y Yaneisy; y a los integrantes del tribunal de tesis Pedro, Larisa, Pascual y Ernesto.

A todos mis compañer@s de clase durante estos 5 años y AMIG@S, que son parte importante en mi vida. Daina, Yendri, Yandy, Randy, Alexander García, Mauricio, Leonardo, Arianne, Inna, Ana Ivette, Sandra, Lisday, Yubisel, Jorge Humberto, Jorge César, Eusebio, Yami, Yesneli, Arislay, Daymara, Keily, Yarianna, Alexander Milán, Heyler, Deyanira, Daylén, Anay, Mayara, Grettel, Greter, Isabel, Rubi, Aibett, Raiko, Lázaro, Karel, Leodanis, MalberS, Pedro Benitez, Yordanka, Anais, Thais, Yuneikys, Carlos, Ever, Aliandro, Aurelio, Manuel, Enmanuel, Leydiana, Marisabet, Mayelin, Mavis, Mayté, Alianis, Lexy, Yadelis, Yoanna, Yunier, Yerenia, Lisbet, Liliana, Mairobis, Aldo, Yoshany, Laritza, Arisday, Annia, Diamicelys, Janna, Katia, Leisy, Lisanne, Rafael, Yuniesky, Jorge Camero, Dayanna, Yadirka, Isis, Alianna, Danarys, Yisumis, Susana Montiel y Arlena.

A mis Amig@sX100PRE. Melba, Manuel, Yudmila, Loriet, Servando, Lilian, Liliana, Wendy, Anet, Adiari, Annia, Chele, Yiyi y Chiqui.

DEDICATORIA

La realización de este trabajo va especialmente dedicada a la memoria de mi abuelita Elvira, que siempre estará presente en mi pensamiento y en todos los momentos importantes de mi vida.

A Zoraida: pues no podría decir en unas pocas palabras todo el amor que siento por ella. Por confiar en mí, darme su apoyo, ser mi guía y mi mayor ejemplo a seguir. Si a partir de hoy soy Ingeniero, es porque he tenido la dicha de tener la mejor madre del mundo.

A Alain: porque me enseñó que nada en la vida es fácil, que se requiere de mucho empeño, sacrificio y que la única forma de tener éxito es proponiéndoselo. Gracias por ser mi otro gran ejemplo, y por ser, además de mi hermano, mi mejor amigo.

A mi papá Ariel; a mis hermanos Gladilys, Onel, Yilbert, Yurisbel, Arisney; y a mi padrastro Arnaldo: que con su amor, confianza y cariño me han rodeado siempre de felicidad, y junto a los cuáles he aprendido que lo principal en la vida es la familia.

A mis sobrinos, en especial a Jose y Disney: para que nunca renuncien a sus sueños, que luchen hasta el final por hacerlos realidad y que mantengan a la familia siempre unida.

A mis amigos, en especial a Melba, Daina y Yadira: por demostrarme que la amistad sobrepasa cualquier límite, cualquier cirscuntancia. Porque los amigos verdaderos siempre estarán ahí para ti, sin importar las adversidades.

RESUMEN

La auditoría de bases de datos es una de las tareas más importantes que se llevan a cabo en cualquier sistema web de gestión de empresas. Estas permiten detectar posibles anomalías en la base de datos, relacionadas con violaciones de seguridad, alteración de la información, entre otras causas. Para ello se debe llevar un registro de todas las acciones que tienen lugar sobre la base de datos, almacenando toda la información relacionada con cada acceso a la misma. El marco de trabajo Sauxe fue creado en un principio para el desarrollo del sistema ERP Cedrux que será utilizado en la gestión de entidades en Cuba, pero por sus características ha sido utilizado además como marco para el desarrollo de otros sistemas web de gestión con resultados satisfactorios. Actualmente dicho marco no cuenta con un mecanismo para el registro y monitoreo de las acciones realizadas sobre la base de datos, por lo que se dificulta el análisis para detectar violaciones de seguridad sobre esta. El presente trabajo propone como solución a esta problemática el desarrollo del componente Trazas de Datos, el cuál se pretende integrar al módulo Trazas que ya posee Sauxe para la gestión de las acciones de los usuarios en el sistema. Trazas de Datos será el encargado de registrar todos los movimientos que se realicen desde el marco de trabajo sobre la base de datos, facilitando su análisis posterior durante la realización de auditorías.

Palabras Clave

Aplicación Web de Gestión, Auditoría de Base de Datos, Acceso a Base de Datos, Procesos, Trazas de Datos.

ABSTRACT

The audits of databases are one of the most important tasks are carried out in any system of business management website. These allow you detect any anomalies in the database related to security breaches, alteration of the information, among other causes. This requires keeping a record of all actions that take place on the database, storing all information associated with each access to it. The framework Sauxe was created initially to develop Cedrux ERP system to be used in the management of entities in Cuba, but these characteristics are also used as a framework for the development of other web management systems successfully. Currently the framework has no mechanism for recording and monitoring of actions performed on the database, so the analysis is difficult to detect safety violations on this. This paper proposes as a solution to this problem, the development of the Data Trace component, which aims to integrate the module has already Sauxe Traces for managing the activities of users on the system. Data Trace will be responsible for any movements that are made from the framework on the database, facilitating further analysis during the audits.

Keywords

Web Application Management, Database Audit, Database Access, Process, Data Trace.

Contenido

INTRODUCCIÓN

La reducción del espacio físico de almacenamiento, la facilidad de búsqueda en grandes volúmenes de información y la disminución de errores provocados en la entrada de datos son algunos de los avances que ha tenido la informática y las comunicaciones en los últimos años.

A raíz de este desarrollo y a la necesidad de las grandes empresas de brindar mejor calidad en sus servicios en el menor tiempo posible, existe una tendencia en estas entidades de adoptar sistemas informáticos para la administración y control de sus operaciones de forma automatizada, algunas de estas operaciones son la contabilidad, el control interno, la gestión del capital humano, la calidad, por mencionar algunas. Estos sistemas deben ser capaces de gestionar, almacenar y proteger toda esa información, pues de ello dependerá el correcto funcionamiento de las entidades.

Actualmente existen gran variedad de sistemas que procesan dicha información, algunos más complejos que otros, pero todos desarrollados con el mismo fin. Ejemplo de ello son los sistemas integrales de Planificación de Recursos Empresariales (ERP[1] por sus siglas en inglés). Estos son sistemas de gestión que cuentan con un conjunto de funcionalidades que automatizan e integran todos los procesos que se llevan a cabo en una empresa. Este tiene gran importancia ya que es una vía o alternativa para la mejora e integración de los procesos de negocio de la empresa trayendo consigo la obtención de ganancias. Las razones principales por las cuales las instituciones implantan sistemas de este tipo, son: aumentar su competitividad, controlar mejor sus operaciones e integrar su información. Debido a las facilidades que brindan, los ERP son ampliamente utilizados en todo el mundo. (1)

En Cuba la mayoría de las instituciones utilizan para estas operaciones sistemas informáticos desarrollados sobre plataformas antiguas que no están acorde a las exigencias actuales de seguridad. Es por ello que, debido a la necesidad de crear un mecanismo para el control de los recursos económicos a nivel nacional, el gobierno cubano comenzó el desarrollo del ERP Cedrux, que constituye una alternativa a los sistemas ya existentes a nivel mundial y favorece la independencia tecnológica con el uso de tecnologías libres.

[1] ERP: Enterprise Resource Planning

Para llevar a cabo dicho proyecto, se necesitó diseñar una arquitectura que soportara la gestión y transferencia de información, y que se adaptara a las condiciones y exigencias de las instituciones cubanas. Con esa idea se creó el marco de trabajo[2] Sauxe, que ya se encuentra en su versión 1.5 y es sobre el cuál se ha desarrollado Cedrux en su totalidad.

Todos los sistemas de este tipo necesitan optar un grupo de medidas con el fin de proteger su información y evitar la pérdida, mal uso, alteración, revelación, destrucción, acceso no autorizado y robo de datos. Las medidas de seguridad de los sistemas es uno de los aspectos más importantes a tratar en la actualidad; existen muchos sistemas que están siendo amenazados pues su información no está asegurada. Estas medidas deben tender a garantizar la confidencialidad, disponibilidad, integridad y seguridad de los datos almacenados en programas y sistemas informáticos, que se encuentran situados físicamente en un determinado local o centro. La realización de auditorías es una de las medidas más importantes en cualquier sistema de gestión para las empresas.

Las auditorías que se llevan a cabo con el objetivo de detectar irregularidades en la base de datos del marco de trabajo Sauxe, se realizan analizando cada uno de los extensos logs que se generan desde el propio gestor de base de datos, lo cuál hace verdaderamente tediosa dicha tarea y se requiere en ocasiones de bastante tiempo para realizarla. Si a ello se agrega que la información que se registra en los logs del gestor no brinda datos indispensables como el usuario de la aplicación web que realizó la transacción, así como el subsistema desde el cuál se llevó a cabo, se puede concluir que existe una brecha de seguridad que atenta contra la calidad del sistema, por lo cuál se impone la creación de un mecanismo para registrar los accesos a la base de datos desde el marco de trabajo que facilite la realización de auditorías.

La problemática existente genera el siguiente **problema a resolver**:

El marco de trabajo Sauxe no provee un mecanismo de registro y monitoreo del acceso a la base de datos, dificultando el análisis para detectar violaciones de seguridad sobre los datos.

Para el desarrollo de la investigación se define como **objeto de estudio** el registro y monitoreo de las acciones que se realizan sobre las aplicaciones de gestión;

[2] Marco de Trabajo: Es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

delimitándose como **campo de acción** las herramientas de registro y monitoreo del acceso a la base de datos en aplicaciones web de gestión.

Para dar solución al problema planteado, se define como **objetivo general** desarrollar un componente para el marco de trabajo Sauxe, que permita el registro y monitoreo del acceso a la base de datos en aplicaciones de Gestión.

Para dar cumplimiento al objetivo general se han trazado los siguientes **objetivos específicos**:

- Realizar el Marco Teórico sobre el registro y monitoreo del acceso a la base de datos en aplicaciones de gestión.
- Realizar el Análisis y Diseño de la solución.
- Implementar la solución.
- Validar la solución aplicando pruebas y métricas de software.

En la presente investigación se establece como **idea a defender** que si se desarrolla un componente para el marco de trabajo Sauxe, que permita el registro y monitoreo del acceso a la base de datos en sistemas de gestión, se logrará fortalecer el análisis para detectar violaciones de seguridad sobre los datos.

Para llevar a cabo esta investigación y dar cumplimiento a los objetivos propuestos, se planificaron las siguientes **tareas:**

1. Estudio de las soluciones existentes respecto al registro del acceso a bases de datos en aplicaciones de gestión.
2. Estudio de los modelos, estándares y normas utilizados para la realización de auditorías informáticas.
3. Estudio de las tecnologías, lenguajes y herramientas propuestas para el desarrollo de la aplicación.
4. Análisis de los componentes del marco de trabajo Sauxe para una familiarización con la estructura y funcionamiento de los mismos.
5. Definición de requisitos y escenarios arquitectónicos.
6. Modelado del diseño del sistema.

7. Implementación de las interfaces.
8. Implementación de la capa de negocio que dé respuesta a los requisitos.
9. Implementación de las validaciones, excepciones, e integración al marco de trabajo.
10. Validación de la implementación de los componentes aplicando pruebas y métricas de software.
11. Validación de la implementación desplegando la solución en entornos reales.
12. Documentación de la solución obtenida.

Los **posibles resultados** de la investigación radican en:

La obtención de un componente para el marco de trabajo Sauxe que lleve a cabo el registro y monitoreo del acceso a bases de datos en aplicaciones web de gestión desarrolladas con el mismo, lo cuál implica:

- Modelo de procesos de negocio.
- Descripción de procesos de negocio.
- Modelo Conceptual.
- Prototipo de Interfaz de Usuario.
- Especificación de requisitos.
- Casos de Prueba.
- Implementación de componentes.
- Descripción de los componentes.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

El presente capítulo constituye el marco teórico de la investigación a realizar. En él se definen una serie de conceptos necesarios para entender el objetivo fundamental del trabajo, profundizando en temas como la auditoría de bases de datos y los diversos métodos para el registro y monitoreo del acceso a las bases de datos, haciendo énfasis en las funcionalidades que brindan algunas herramientas de auditoría de bases de datos a nivel mundial. Además se describen las tecnologías que serán utilizadas durante la implementación.

A continuación se definirán algunos conceptos relacionados con el dominio del problema. Primero que todo se debe conocer qué es una aplicación Web, para luego definir el tipo de aplicación Web para el cuál está diseñado dicho marco de trabajo, que es el de gestión.

1.2 Aplicación Web

Las aplicaciones Web se desarrollan como una extensión de los Sistemas Web para agregar funcionalidades del negocio al proceso. En otros términos, una aplicación Web es un sistema Web que permite a los usuarios ejecutar lógica de negocio a través de un Navegador, o lo que es lo mismo: modificar el estado del negocio. Su arquitectura general es la de un sistema Cliente - Servidor. El protocolo principal de comunicación en una aplicación Web es HTTP, el cual funciona normalmente desconectado, es decir, el cliente hace una petición al servidor, este la procesa y le devuelve el resultado, terminando la comunicación entre estos. (2)

Como se ha dicho anteriormente el marco de trabajo Sauxe está desarrollado para aplicaciones web, específicamente las de gestión.

1.3 Aplicación Web de Gestión

Estas son aplicaciones Web que permiten gestionar la totalidad o la mayor parte de los procesos de negocio, es una combinación de CMS, CRM, y ERP. (3)

- ***CMS (Content Management System):*** Sistema gestor de contenidos web, o

aplicación automatizada que permite gestionar los contenidos de la web.

- ***CRM (Customer Relationship Management):*** Gerencia de relaciones con los clientes. Mediante el software CRM se logra conocer más a fondo las necesidades y preferencias de los clientes, y de esta forma poder ofrecerles un producto con mayor valor agregado dependiendo de cada usuario.
- ***ERP (Enterprise Resource Planning):*** Sistema que permite gestionar los recursos empresariales (recursos humanos, proveedores, recursos de capital, materias primas, procesos internos, entre otros). (3)

Es importante dejar claro el concepto de proceso de negocio. Para ello se escogieron algunas definiciones de diversos autores, las cuales son mencionadas a continuación.

1.4 Proceso de negocio

La norma internacional ISO-9001 define un proceso de negocio como *"una actividad que utiliza recursos y que se gestiona con el fin de permitir que los elementos de entrada se transformen en resultados"*. (4)

Oscar Barros señala que *"un proceso es un conjunto de tareas lógicamente relacionadas que existen para conseguir un resultado bien definido dentro de un negocio; por lo tanto, toman una entrada y le agregan valor para producir una salida. Los procesos tienen entonces clientes que pueden ser internos o externos, los cuales reciben a la salida, lo que puede ser un producto físico o un servicio. Éstos establecen las condiciones de satisfacción o declaran que el producto o servicio es aceptable o no"*. (5)

Thomas Davenport define un proceso como *"un conjunto estructurado, medible de actividades diseñadas para producir un producto especificado, para un cliente o mercado específico. Implica un fuerte énfasis en CÓMO se ejecuta el trabajo dentro de la organización, en contraste con el énfasis en el QUÉ, característico de la focalización en el producto"*. (6)

William Loyola lo define como *"flujo o progresión de actividades que se siguen para alcanzar algún objetivo del negocio. También se le define como el conjunto de actividades que sirven para crear valor para el cliente, ya sea este un cliente interno (otras áreas del negocio) o externo"*. (7)

Este concepto se utilizará durante toda la investigación, ya que todas las operaciones que se realizan en los sistemas web de gestión, como por ejemplo: registrar cliente, brindar servicio, actualizar información de recursos humanos, entre otras; corresponden a procesos de negocio.

Un elemento a tener en cuenta durante todas estas operaciones es que en casi todas se maneja información, que por lo general está almacenada en bases de datos, y para la cuál debe estar garantizada su seguridad.

1.5 Seguridad en los datos

La seguridad puede entenderse como aquellas reglas, técnicas y actividades destinadas a prevenir, proteger y resguardar lo que es considerado como susceptible de robo, pérdida o daño, ya sea de manera personal, grupal o empresarial. En este sentido, es la información el elemento principal a proteger, resguardar y recuperar dentro de las redes empresariales. (8)

La seguridad de la información se ha mantenido sobre los diferentes pilares:

- Confidencialidad: permitir acceso únicamente a los datos a los cuales el usuario está permitido.
- Integridad: asegurar que los datos no se falsifican o alteran por usuarios no autorizados.
- Disponibilidad: asegurar que los sistemas y datos están disponibles para los usuarios autorizados cuando lo necesiten.

Los datos se alojan en servidores con sistemas de almacenamiento altamente fiables y se ven libres de problemas que usualmente sufren los ordenadores de usuarios comunes como virus y roturas de disco. La utilización de esta tecnología unida al uso del patrón arquitectónico Modelo-Vista-Controlador (MVC) y la arquitectura en capas conlleva a reducir costos y complicaciones y proporciona mayor libertad a la hora de realizar cualquier tipo de cambios, siendo así una excelente opción para que las pequeñas empresas automaticen sus procesos sin invertir demasiado en equipo, desarrollo y capacitación.

Para el acceso a los datos es necesario poseer una arquitectura que permita a los

usuarios acceder a estos con la debida autorización, y que además permita llevar un control de dichos accesos.

1.6 Arquitectura AAA

AAA (*Authentication, Authorization and Accountig, por sus siglas en ingles*) es una arquitectura de sistema, la cual sirve para configurar un conjunto de tres funciones de seguridad (Autenticación-Autorización-Contabilización) de una forma coherente. Dicha arquitectura ofrece una forma modular de proveer los siguientes servicios:

- ***Autenticación:*** Proporciona el método de identificación de usuarios, incluyendo nombre de usuario y contraseña, desafío y respuesta, soporte de mensajería, y, según el protocolo de seguridad que seleccione, puede ofrecer cifrado. La autenticación es la forma en que un usuario se identifica antes de poder acceder a la red y los servicios que esta ofrece.
- ***Autorización:*** Provee el método de control de acceso remoto, incluyendo autorización total o autorización para cada servicio, lista de cuentas y perfil por usuario, soporte para grupos de usuarios, y soporte para IP, IPX, ARA y Telnet. Autorización trabaja agrupando un grupo de atributos que describen lo que el usuario está habilitado a usar o acceder. Estos atributos son comparados con la información contenida en una base de datos de un usuario determinado y el resultado se devuelve a AAA para determinar las capacidades reales de los usuarios y las restricciones.
- ***Contabilización:*** Posee un método de recolección y envío de información al servidor de seguridad, el cual es usado para facturar, auditar y reportar nombres de usuario, tiempo de inicio y final, comandos ejecutados (como PPP), cantidad de paquetes enviados, y número de bytes. Contabilidad permite realizar el seguimiento de los usuarios que tienen acceso a los servicios, así como la cantidad de recursos de red que están consumiendo. Estos datos pueden ser analizados para la gestión de la red, la facturación del cliente, y / o auditoría.

AAA provee los siguientes beneficios:

- Incrementación de flexibilidad y control de configuración de acceso.

- Escalabilidad.
- Métodos de autorización estandarizados, como RADIUS, TACACS+ o Kerberos.
- Múltiples sistemas de backup.

La arquitectura AAA está diseñada para que el administrador de la red pueda configurar dinámicamente el tipo de autenticación y autorización que se quiera. (9)

Una de las tareas más importantes en cuanto a la seguridad de los datos es la realización de auditorías.

1.7 Auditoría

De una forma muy general, la auditoría es una actividad o acción (o un grupo de estos) realizadas por uno o varios elementos (humanos, máquinas...) con el objetivo de prevenir, detectar o corregir errores, omisiones o irregularidades que afecten al funcionamiento de algo. También podemos definirla como la actividad o conjunto de actividades realizadas para determinar, por medio de la investigación, la adecuación de los procedimientos establecidos, instrucciones, especificaciones, codificaciones y estándares u otros requisitos, la adhesión a los mismos y la eficiencia de su implantación. (10)

Por otro lado la Auditoría Informática es el proceso de recoger, agrupar y evaluar evidencias para determinar si un Sistema de Información salvaguarda el activo empresarial, mantiene la integridad de los datos, lleva a cabo eficazmente los fines de la organización, utiliza eficientemente los recursos, y cumple con las leyes y regulaciones establecidas. También permite detectar de forma sistemática el uso de los recursos y los flujos de información dentro de una organización y determinar qué información es crítica para el cumplimiento de su misión y objetivos, identificando necesidades, duplicidades, costes, valor y barreras, que obstaculizan flujos de información eficientes. (11)

Para la presente investigación se hará énfasis en la auditoría de bases de datos, pues es la que se relaciona directamente con la problemática expuesta.

1.7.1 Auditoría de Bases de Datos

La Auditoría de bases de datos es el monitoreo y registro de las acciones realizadas por los usuarios sobre la Base de Datos, esto puede basarse en acciones individuales, tal como ejecutar una sentencia o en una combinación de factores como el nombre,

aplicación, el tiempo, etc.

La Auditoría es normalmente usada para:

- Investigar actividades sospechosas. Por ejemplo si un usuario no autorizado borra datos de las tablas, el administrador puede auditar todas las operaciones realizadas sobre los objetos de la Base de Datos, y puede determinar información como la hora de conexión del usuario, los objetos modificados, etc.
- Monitorear y almacenar información de las actividades de la Base de Datos. Por ejemplo el administrador puede acumular información estadística, sobre las tablas actualizadas últimamente, determinar el número de usuarios que se conectan, conocer los objetos más utilizados, conocer las horas pico de transacciones, etc. Es decir le permitirá obtener información útil para la optimización de la Base de Datos.

Para implementar una Auditoría de la Base de Datos, primero se debe determinar las razones por las cuales se va a realizar una Auditoría y posteriormente solo auditar el menor número de sentencias, usuarios, objetos requeridos para prevenir información innecesaria de Auditoría y optimizar el uso de recursos, por ejemplo al auditar una determinada acción de la Base de Datos, se determina exactamente que tipo de actividad se va a auditar que se requiere rastrear, auditar solo actividades que le interesen, auditar solo el tiempo necesario para generar la información. (12)

Un aspecto importante en este punto es la auditoría de bases de datos basándose en los procesos de negocio, ya que esta permite definir cuáles son las operaciones que se desean auditar.

A continuación se describen algunos métodos utilizados para realizar auditorías de bases de datos.

Auditoría de bases de datos a nivel de aplicaciones

Muchos gestores web de información incluyen pequeñas aplicaciones que ayudan a gestionar los datos (datos personales, pedidos, pagos online, control de acceso, etc.). Existen otros que se utilizan para realizar una gran variedad de operaciones complejas y que requieren una alta seguridad (portales corporativos, banca por Internet, comercio

electrónico, redes privadas virtuales o extranet, etc.) y esto implica la utilización de una compleja aplicación que gestione todas estas operaciones.

Por ello es necesario un servicio para poder analizar todas estas aplicaciones, de forma independiente y exhaustiva. En este caso, el alcance de la auditoría se delimitará al funcionamiento del día a día de la aplicación, así como en los procesos en los cuales participe. Es realizada por la aplicación y es totalmente externa a la base de datos. (10)

Auditoría de bases de datos mediante disparadores

Los disparadores son procedimientos que se ejecutan cuando se produce un evento de base de datos entre los que están las operaciones de manipulación de datos, de conexión y desconexión de usuarios, de arranque de la base de datos o de fallo. El acto de ejecutar un disparador se conoce como disparo. La idea está en que cuando un usuario inserta, actualiza o borra datos de una tabla, se ejecute el disparador correspondiente para copiar la información a una tabla diseñada para ello: una tabla de auditoría. Así, el disparador guarda en la tabla de auditoría información que el administrador de la base de datos o el auditor considere necesario: fecha en la que se produjo el borrado, modificación o inserción, usuario que realizó la acción, tabla sobre la que se realizó la acción, dato anterior a la modificación, dato posterior a la modificación, dato que se borró, etc.

Los sistemas gestores de bases de datos hoy en día poseen herramientas que utilizan triggers para el registro de estos eventos. Ejemplo de estos sistemas son Oracle, Microsft SQL Server y la alternativa de código abierto MySQL. Este método para la auditoría de bases de datos aprovecha las funcionalidades de los diferentes sistemas gestores de base de datos, pero son muy difíciles de crear, gestionar y mantener, y además conllevan a una sobrecarga del servidor, una vez que se ejecuta un trigger por cada acción sobre la misma. (10)

1.7.2 Logs de auditoría

Durante la auditoría se analiza la información contenida en los logs que se generan durante la ejecución de determinadas acciones en un sistema. Los logs son los ficheros informáticos conteniendo detalles de cambios o alteraciones a registros de datos, que pueden usarse en el caso de que se precise la recuperación del sistema. Activar esta capacidad normalmente incurre en cierta carga añadida al sistema, pero permite la

revisión de toda o parte de la actividad producida por o en el sistema. (13)

Para la presente investigación se utilizará el término 'traza de datos', el cuál guarda relación con los logs de auditoría y el acceso a las bases de datos.

1.8 Traza de Datos

Una traza es la marca dejada por un elemento al haberse desplazado a una nueva posición, por lo que llevándolo al contexto informático se puede decir que las trazas son una colección de eventos y datos devueltos por una aplicación, que representan el historial o rastro dejado por cierto proceso cuando se ejecuta en un sistema. (14)

Al llevar este concepto al tema relacionado con la auditoría de bases de datos se puede llegar a un concepto que sea aún más específico a los intereses de esta investigación, las **trazas de datos**. Para el presente trabajo se definen las trazas de datos como *"el registro de las acciones, ya sea de inserción, modificación o supresión, que se llevan a cabo sobre una base de datos; las cuales incluyen información acerca de quién llevó a cabo la transacción, desde dónde, cuándo y cuáles fueron los datos accedidos, así como la operación que se realizó."*

1.9 Herramientas para la gestión de trazas de datos.

Las herramientas tecnológicas son ampliamente utilizadas en el proceso de auditoría y la captación de evidencias. Un sistema de gestión de logs de auditoría es aquel que controla los eventos que ocurren dentro de un sistema determinado, lo cual es muy importante para su correcto funcionamiento. Un sistema informático que no posea un componente con esta funcionalidad implica grandes riesgos, entre ellos: recuperarse tras una falla del sistema o en caso de ser quebrantada la seguridad del sistema, quede impune, dado que sin las trazas puede ser muy difícil o, en ocasiones, incluso imposible averiguar qué ha pasado realmente para que la aplicación haya fallado.

En las trazas se almacena toda la actividad y eventos que ocurren en un sistema: quién entra, qué comandos ejecuta, qué errores muestran las aplicaciones, etc. Para el registro de los logs relacionados con los accesos a las bases de datos existen actualmente diferentes herramientas. Para la presente investigación se estudiaron las siguientes:

1.9.1 Apex SQL Log

Herramienta de auditoría y recuperación de bases de datos SQL que analiza el registro de transacciones del SQL Server para mostrar información sobre cambios estructurales y de datos. Ya que Apex SQL Log lee el Registro de Transacciones, no se requiere sobrecargar la base de datos y se puede ejecutar la auditoría sobre cambios hechos aún antes de instalar la herramienta que examinará el registro. Apex SQL Log muestra la computadora, usuario (incluyendo el usuario NT) y la aplicación que hizo el cambio y muestra la historia completa del artículo auditado. Esta herramienta incluye una funcionalidad de recuperación que puede recuperar datos perdidos, eliminados y truncados y datos de tablas eliminadas.

Una de las limitaciones que posee esta herramienta es que si el archivo log de transacciones contiene detalles de las modificaciones realizadas a una tabla que se ha eliminado, no se podrá mostrar el nombre de la tabla ni detalles de la modificación. (15)

Además, su uso está restringido a bases de datos gestionadas por SQL Server y su costo por concepto de licencias va desde los $999.00 dólares a los $1450.00 dólares por licencia. (16)

1.9.2 Oracle 9i

La auditoría en Oracle 9i se realiza de tres formas distintas:

1. Mediante las herramientas propias de Oracle.

El SGBD[3] Oracle tiene la capacidad de auditar todas las acciones que tienen lugar en la BD. Se puede auditar tres tipos de acciones:

- Intentos de entrada en cuentas de la BD.
- Accesos a los objetos de la BD.
- Acciones sobre la BD.

El gestor registra todos los intentos de acción, tanto los exitosos como los infructuosos, aunque es un parámetro configurable.

Para habilitar la capacidad de auditoría, se debe fijar el parámetro AUDIT_TRAIL en el fichero init.ora. Los registros de auditoría se almacenan en la tabla SYS.AUD$ o bien su

[3] SGBD: Sistema Gestor de Bases de Datos

gestión se deja al SO. Cuando se decide utilizar la tabla SYS.AUD$ esta debe revisarse periódicamente, por si hiciera falta truncarla debido a que su aumento de tamaño puede causar problemas de espacio en el tablespace SYSTEM. (17)

2. Utilizando herramientas de terceros como Audit DB Entegra de Lumigent.

Esta herramienta carga a una base de datos los logs de la base de datos, en estos logs quedan registradas todas las operaciones que se realizaron (create, alter, delete, insert, select, etc.). (18)

3. A través de triggers en las tablas con el evento insert, update, delete que alimente varias tablas en donde queda guardada la información necesaria para saber cuando se insertó, modificó o eliminó. (18)

1.9.3 Audit DB Entegra de Lumigent

Dentro de las prestaciones que presta esta herramienta se puede mencionar el registro de la actividad en múltiples servidores, como cambios a schemas y permisos (DLL), logins, modificaciones a datos (DML), lecturas a tablas. También posee un repositorio compartido para archivo histórico que constituye un colector de datos centralizado, independiente de los servidores auditados y un gestor de archivos históricos. Además posee un sistema de alerta y aviso en tiempo real de cambios a estructura y permisos, accesos y otras acciones. Está diseñado y desarrollado para auditorías de empresas, no utiliza triggers en la base de datos, y no necesita modificar los sistemas de gestión o aplicaciones.

Ilustración 1.9.1 Historial de las acciones realizadas a una tabla en específico

Esta herramienta carga a una base de datos los logs de la base de datos, en estos logs

queda registrada todas las operaciones que se realizaron (create, alter, delete, insert, select, etc.). (18)

El principal inconveniente de esta herramienta es que está desarrollada para bases de datos que utilicen Oracle o SQL Server como gestor de bases de datos. Además el precio de la licencia es muy elevado, alcanzando los $11.500 dólares. (19)

1.9.4 Oracle Audit Vault

Oracle Audit Vault es una herramienta que permite automatizar la recolección de datos de auditoría, monitorear y generar informes, volcando dicha información en un esquema propio. Básicamente, recolecta datos de auditoría de una base de datos, ya sea Oracle o MS SQLServer, y, además de guardar los datos en un esquema propio en forma de DataWarehouse, estudia dichos datos para detectar cualquier anomalía.

Oracle Audit Vault consta de un servidor y un agente. El servidor, que en la versión más reciente de Audit Vault no está desarrollado para usarlo en entorno Windows, debe ser configurado por el administrador, y se encargará de recolectar todos los datos de auditoría que genera la base de datos, organizarla y generar los informes. El agente se encarga de ofrecer sus servicios al auditor de la base de datos. Desde el agente, el auditor podrá crear políticas de auditoría (sólo para bases de datos Oracle), crear alertas y obtener informes. El agente puede estar instalado en la misma máquina donde está la base de datos Oracle o en otra máquina. (20)

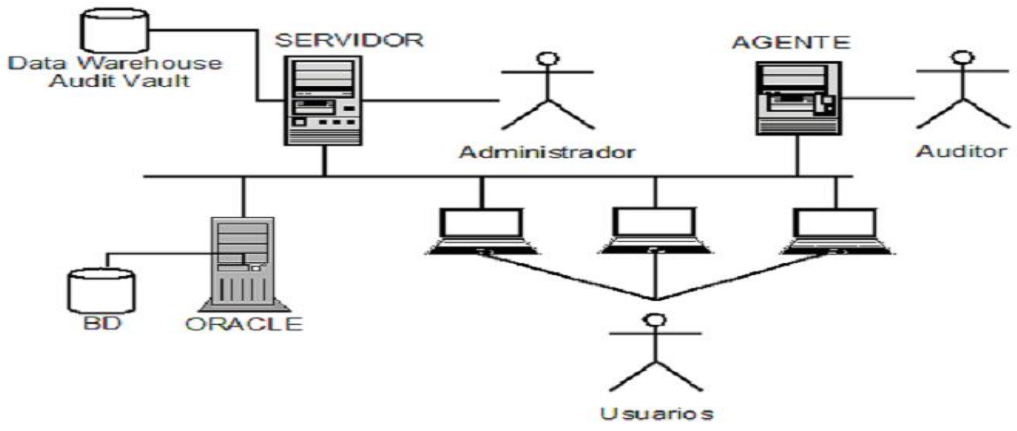


Ilustración 1.9.2 Arquitectura de alto nivel del sistema Audit Vault.

El principal inconveniente es el hecho de que esta herramienta no tiene compatibilidad

con el gestor de bases de datos PostgreSQL.

Otro inconveniente a priori es el coste de licencia. La licencia para el Servidor de Oracle Audit Vault cuesta 50.000 dólares por procesador, y la licencia para el Agente de Oracle Audit Vault, 3.000 dólares por procesador. (21)

1.9.5 Componente Trazas del marco de trabajo Sauxe

Para la presente investigación se hizo un profundo estudio de las funcionalidades que brinda el marco de trabajo Sauxe, prestando especial atención al componente Trazas, que se encuentra en su versión 1.0.

El componente Trazas es un módulo creado sobre el mismo marco de trabajo, y registra un conjunto de acciones que se realizan a través de este, como son:

- Registro de trazas URL: Registrar una traza si se accede a alguna acción introduciendo una dirección en el navegador y sin pasar por el portal y seguir el flujo normal.
- Registro de trazas de autenticación: Registrar una traza cada vez que se autentique un usuario.
- Registro de trazas de cierre de sesión: Registrar una traza cuando se cierre una sesión.
- Registro de trazas de acción (enlace acción): Registrar una traza con las acciones que se ejecutan a partir de una acción.
- Registro de trazas de integración (IoC): Registrar una traza con quien se producen las integraciones.
- Registro de trazas de excepción.
- Registro de trazas de excepción de integración (IoCexcepcion).
- Registro de trazas de rendimiento.

Dicho componente centraliza todos esos registros en una sola base de datos, en tablas diferentes para cada categoría. Al estar integrado al marco de trabajo Sauxe utiliza para el registro de las acciones el gestor de base de datos PostgreSQL. Además presenta una interfaz visual para la gestión de dicha información, que permite la búsqueda por distintos

criterios, e incluye la generación de reportes.

La principal limitante de esta herramienta radica en que no cuenta dentro de sus funcionalidades con el registro de acciones sobre la base de datos.

1.10 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. (22)

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.

1.11 Tecnologías utilizadas para el desarrollo

CEIGE ha tomado decisiones tecnológicas que involucran el uso de tecnologías libres para el desarrollo de sus aplicaciones. El presente trabajo forma parte de este proceso de desarrollo, y se ajusta a dichas medidas. A continuación se describen estas tecnologías.

1.11.1 Librerías y Marcos de Trabajo

El desarrollo de la solución se realizará utilizando el marco de trabajo Sauxe desarrollado por el CEIGE, el cual 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. (23)

El marco de trabajo Sauxe da solución a un sin número de escenarios o aspectos arquitectónicos como: gestión y configuración dinámica de cache, integración de componentes, acceso a bases de datos a través de una capa de abstracción, gestión de concurrencia de recursos, administración centralizada de transacciones, gestión dinámica de las trazas generadas por los sistemas, implementación de mecanismos de autenticación y autorización, visualización de las funcionalidades de un sistema, entre otros escenarios de alta complejidad. Cuenta con una arquitectura en capas que a su vez presenta en sus capas superiores un MVC[4]. (24)

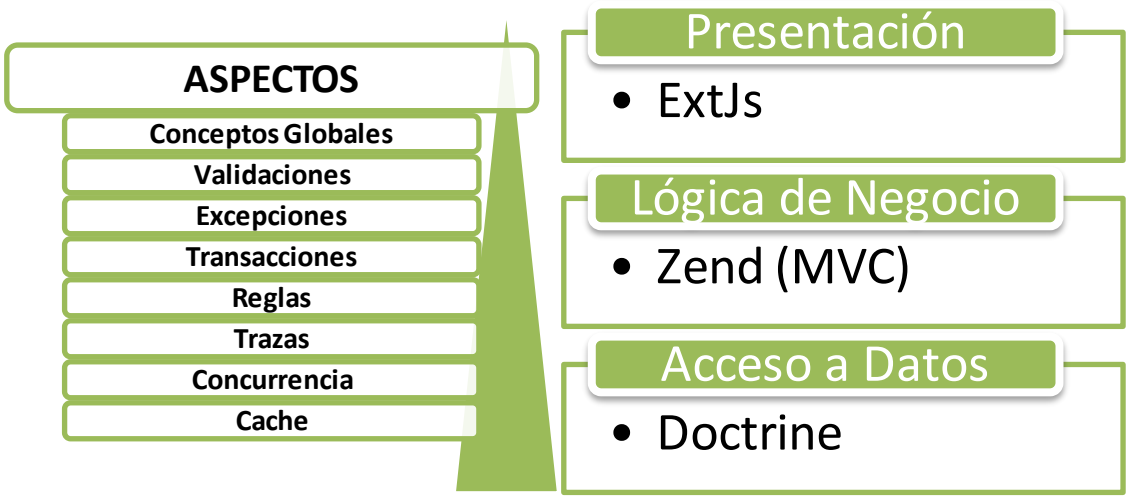


Ilustración 1.11.1 Estructura del marco de trabajo Sauxe

[4] MVC: Patrón arquitectónico Modelo-Vista-Controlador.

Dicho marco de trabajo posee particularidades que lo hacen único en su tipo, por ejemplo, el núcleo está desarrollado utilizando PHP, Zend Framework, Doctrine y ExtJS. Además se desarrolló la extensión del núcleo ZendExt, creada por el equipo de tecnologías del proyecto Cedrux y que está basado en las particularidades del sistema como tal. ZendExt es una extensión de código abierto del marco de trabajo ZendFramework, diseñado para PHP 5, que reutiliza el MVC. Está constituido por un controlador vertical para el control de las acciones que son realizadas por las vistas hacia el controlador, un motor de reglas utilizadas en las validaciones en el servidor, así como la inclusión del IoC para la comunicación entre módulos y componentes. También tiene incorporado el marco de trabajo Doctrine para el trabajo con la capa de abstracción de la base de datos y con marco de trabajo ExtJS para el desarrollo de las vistas. Brinda también abstracción de las operaciones transaccionales (TransactionManager) y extensión de los dominios de tipo (Validador). Está basado en componentes, con principios de integración distribuidos, ofrece conectores y configuraciones mediante interfaces bien definidas.

Sauxe utiliza programación orientada a aspectos, que permite una adecuada modularización de la aplicación y posibilita una mejor separación de los diferentes conceptos.

Zend Framework 1.8

Zend Framework no es más que un framework MVC (Modelo-Vista-Controlador) open-source para el desarrollo de aplicaciones Web con PHP. Brinda soluciones para construir sitios web modernos, robustos y seguros. Posee un bajo acoplamiento entre sus componentes, lo que posibilita la utilización de los mismos a conveniencia, aunque todos estos en conjunto conforman un potente y extensible framework para aplicaciones web. Brinda una alta abstracción de bases de datos, haciendo extremadamente simple la interacción con estas, sin necesidad de escribir ninguna consulta SQL. Cuenta con módulos para el manejo de ficheros PDF, canales RSS, entre otros. Cuenta con clientes para el acceso a WS y robustas clases para la autenticación y el filtrado de entrada; completa documentación y test de alta calidad. (25)

Doctrine 0.11

Doctrine es un potente y completo sistema ORM (Object Relational Mapping) para PHP

con un DBAL (Database Abstraction Layer) incorporado, el mismo cuenta con disímiles funcionalidades, una de ellas es que da la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. Su principal ventaja radica en poder acceder a la base de datos utilizando la programación orientada a objetos (POO[5]), debido a que Doctrine utiliza el patrón Active Record para manejar la base de datos, tiene su propio lenguaje de consultas y trabaja de manera rápida y eficiente.

El patrón Active Record es una extensión del patrón Domain Model (Modelo de Dominio), que se entiende como una clase o un grupo de clases que representan a objetos o responsabilidades particulares en la aplicación. (26)

Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria. (27)

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. (28)

PHP

Preprocesador de Hipertexto o PHP (por sus siglas en inglés) es un lenguaje de programación de alto nivel orientado al desarrollo de aplicaciones web, que es interpretado del lado del servidor. Puede ser utilizado en casi todos los sistemas

[5] La terminología Programación Orientada a Objetos (POO u OOP según siglas en inglés) representa un paradigma de programación donde se definen los programas en términos de "clases de objetos", tales objetos son entidades que combinan estado (datos), comportamiento (procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto).

operativos existentes, permitiendo migrar las aplicaciones de un sistema a otro sin necesidad de realizar cambios en el código. Su rapidez en la ejecución y los bajos requerimientos de consumo en los sistemas donde es desplegado lo hacen uno de los preferidos por los desarrolladores. Se integra perfectamente a la mayoría de los Sistemas Gestores de Bases de Datos. Su mayor ventaja radica en ser un lenguaje libre, por lo que se convierte en una alternativa de muy fácil acceso, además de poseer una comunidad de desarrolladores que intercambian experiencias. (29)

JavaScript

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. 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. (28)

1.12 Herramientas de Desarrollo

1.12.1 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 las sus capacidades. Actualmente existen muchos módulos para Apache que son adaptables a este, y están disponibles para su instalación cuando son 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. (30)

1.12.2 PostgreSQL 8.3

Es un sistema de gestión de bases de datos libre basado en el proyecto Postgres, perteneciente a la Universidad de Berkeley. Es un sistema objeto-relacional, que incluye

características como la herencia, valores no atómicos (atributos basados en vectores y conjuntos), funciones, disparadores, entre otras. Es altamente extensible, permitiendo el uso de operadores, funciones y tipos de datos definidos por el usuario. Soporta la integridad referencial garantizando la integridad de los datos en la base de datos.

PostgreSQL permite realizar múltiples conexiones desde procesos clientes, existiendo un proceso maestro en el servidor que siempre se ejecuta y está a la espera de nuevas conexiones clientes, de forma tal que cuando alguien se conecta, se inicia un nuevo proceso, asegurando que la nueva conexión no necesite del proceso postgres original, por lo que una de sus principales características es la alta concurrencia. (27)

1.12.3 NetBeans 6.8

NetBeans es 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 extenderlo. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso. (31)

1.12.4 Herramientas CASE

CASE son las siglas correspondientes a Computer Aided Software Engineering, que en su traducción al español significa Ingeniería de Software Asistida por Computadoras. Las herramientas CASE son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y dinero. Para el modelado de la solución se utilizará Visual Paradigm 4.3.

Visual Paradigm 4.3

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado 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. Esta herramienta también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. (32)

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de

un sistema que involucra una gran cantidad de software. Probablemente, una de las innovaciones conceptuales en el mundo tecnológico del desarrollo de software que más expectativas y entusiasmo han generado en muchos años. Es un estándar en la industria del software, creado por Grady Booch, James Rumbaugh e Ivar Jacobson.

UML también intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos se crea una documentación que cualquier desarrollador con conocimientos de UML será capaz de entender. Su utilización es independiente del lenguaje de programación y de las características de los proyectos ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otra rama. (33)

1.12.5 Herramientas para el desarrollo colaborativo

Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas. Para el desarrollo de la solución se utilizará para el control de versiones, el sistema Subversion 1.6.12.

Subversion 1.6.12

También conocido como SVN[6], es un sistema de control de versiones que se ha popularizado bastante, en especial dentro de la comunidad de desarrolladores de software libre. Está preparado para funcionar en red y se distribuye bajo licencia libre.

- Mantiene versiones no sólo de archivos, sino también de directorios.
- Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.
- Atomicidad de las actualizaciones, una lista de cambios constituye una única transacción o actualización del repositorio, esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.
- Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos.

[6] Es el nombre que se utiliza en la línea de comandos Ej.: $ svn export nombre-de-repositorio

1.12.6 Patrones de Diseño

Los patrones de diseño (*del inglés design patterns*) son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una solución a un problema del diseño.

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas de software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente. (34)

1.12.6.1 Patrones GoF

Los patrones GoF son patrones de diseño publicados en el libro *Design Patterns: Elements of Reusable Object-Oriented Software* por Gamma, Helm, Jonson y Vlissides conocidos mundialmente por Gang of Four o Pandilla de los cuatro. En este libro se encuentran recopilados un total de 23 patrones clasificados en patrones creacionales, estructurales y de comportamiento. (34)

- Los patrones creacionales se encargan de la creación de los objetos ayudando a que el sistema sea independiente de la creación, composición y representación de los objetos.
- Los patrones estructurales son los encargados de cómo las clases y objetos están compuestos para formar estructuras más grandes. Los patrones estructurales usan la herencia para componer interfaces u objetos en tiempo de ejecución.
- Los patrones de comportamiento plantean algoritmos y la asignación de responsabilidades entre objetos. Estos patrones no sólo describen clases y objetos

sino también describen la comunicación entre ellos.

Para el diseño de la solución propuesta en este trabajo se utilizaron dos patrones GoF:

Singleton

Clasificado dentro de los creacionales, el propósito del patrón Singleton es garantizar que una clase sólo tiene una única instancia, proporcionando un punto de acceso global a la misma.

Su estructura es de la forma:

SOLITARIO
static Instancia()
OperacionDelSolitario()
ObtenerDatosDelSolitario()
static InstanciaUnica
DatosDelSolitario

Tabla 1 Estructura del patrón Singleton

El método “Instancia” es el punto de acceso a la “InstanciaÚnica” del “Solitario” para los clientes. Lo que ocurre en realidad con este método es que, si “InstanciaÚnica” no ha sido aún creada, la crea llamando al constructor “Solitario”, que no es público. (35)

Observer

El patrón Observer está clasificado dentro de los de comportamiento. Un efecto lateral muy frecuente en aquellos sistemas que se fragmentan en un conjunto de clases que cooperan es la necesidad de mantener la consistencia entre los distintos objetos interrelacionados. Para no recurrir a soluciones fuertemente acopladas (que reducen la posibilidad de reutilización), este patrón define una dependencia “uno-a-muchos” entre objetos, para que, cuando uno de ellos cambie su estado, todos los que dependan de él sean avisados y puedan actualizarse convenientemente. (35)

1.12.7 Patrones de Arquitectura

MVC

Modelo-Vista-Controlador (MVC) es una de las arquitecturas de aplicaciones web más comunes. Está conformada como su nombre lo indica, por tres capas, las cuales se describen a continuación:

- **Vista:** La renderización de código front-end, frecuentemente llamada nivel de presentación, debería aspirar a producir la salida HTML para el usuario con poco o nada de lógica de aplicación. Como muchas aplicaciones serán internacionalizadas (por ejemplo no conteniendo cadenas localizadas o información cultural en la capa de presentación), deben usar llamadas al modelo (lógica de aplicación) para obtener la información requerida para suministrar información útil al usuario en su lenguaje y cultura preferido, dirección del script, y unidades. Todas las entradas de los usuarios se encuentran redireccionadas hacia los controladores en la lógica de la aplicación.
- **Controlador:** El controlador (o lógica de la aplicación) toma entradas de los usuarios y las dirige a través de varios flujos de trabajo que llaman a los objetos del modelo de la aplicación para extraer, procesar, o almacenar información. Los controladores bien codificados, validan información centralmente en el servidor contra problemas de seguridad comunes antes de pasar la información al modelo de procesamiento y se aseguran que la salida de datos sea segura o en un formato preparado para una salida segura por parte del código de visualización.
- **Modelo:** La idea es encapsular el trabajo sucio en el modelo de código, en lugar de exponer primitivas. Si el controlador y el modelo se encuentran en diferentes máquinas, la diferencia de rendimiento será asombrosa, por lo que es importante para el modelo ser útil a un nivel alto. El modelo es responsable de la comprobación de datos en contra de las reglas de negocio, y cualquier riesgo residual para el único almacén de datos en uso. Si el modelo almacena los datos en un lenguaje interpretado, como SQL, entonces el modelo se encarga de la prevención de inyección de SQL. (36)

1.13 Conclusiones Parciales

A lo largo del capítulo se definieron una serie de conceptos necesarios para el entendimiento del problema tratado en el presente trabajo y se realizó un estudio de las herramientas para el registro y monitoreo del acceso a bases de datos existentes a nivel mundial.

La investigación realizada demuestra que no existe una herramienta de registro de accesos a datos que se adapte completamente a las características del marco de trabajo Sauxe. Por un lado la gran mayoría de las herramientas existentes están desarrolladas para ser utilizadas en bases de datos Oracle y SQL Server, y el marco de trabajo Sauxe requiere PostgreSQL como gestor. Otra de las limitaciones encontradas es el elevado coste por concepto de licencias, que atenta contra la política del centro de utilizar tecnologías libres. Por último, ninguna de las herramientas estudiadas posee dentro de sus funcionalidades el registro de trazas de datos orientado a procesos.

Por lo antes mencionado se determinó que es necesario el desarrollo de una solución que se pueda integrar al marco de trabajo Sauxe, que permita el registro de los accesos a la base de datos basándose en procesos determinados. Para ello se definió una metodología para el proceso de desarrollo del software, se definieron los patrones arquitectónicos y de diseño, y se propusieron las herramientas y tecnologías que se utilizarán para llevar a cabo el desarrollo de la solución.

CAPÍTULO 2. ANÁLISIS Y DISEÑO DEL SISTEMA

2.1 Introducción

El siguiente capítulo recoge los resultados obtenidos durante el proceso de desarrollo de la solución, así como algunos de los artefactos generados por el mismo. Primeramente se elabora un modelo conceptual para lograr un entendimiento de lo que se quiere desarrollar, y luego se especifican y describen los requisitos funcionales de la solución. Posteriormente se realiza una descripción del diseño elaborado por los autores para alcanzar las metas trazadas.

2.2 Requisitos de Software

La captura de requisitos de software es una de las actividades fundamentales que se realiza en el proceso de desarrollo de software, ya que estos son la condición o capacidad que tiene que ser alcanzada por un sistema o componente software para satisfacer un contrato, estándar, u otro documento impuesto formalmente. Los mismos pueden dividirse en requisitos funcionales y requisitos no funcionales. Los primeros son capacidades o condiciones que el sistema debe cumplir. Los segundos son propiedades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. A continuación se muestra un modelo conceptual utilizado para relacionar los principales conceptos asociados al problema.

2.2.1 Modelo conceptual

En el modelo conceptual se muestra el marco de trabajo Sauxe, el cuál contiene varios subsistemas (Finanzas, Contabilidad, Logística, etc.). Estos a su vez tienen asociados procesos que se realizan desde el mismo a través de los cuales se accede a diferentes tablas contenidas en esquemas de la base de datos. Dichos procesos son gestionados desde una interfaz visual de Configuración, y pueden estar activados o desactivados (on/off). Al estar activo un proceso, todas las tablas asociadas a este serán auditadas.

Cada vez que se lleva a cabo una transacción sobre la base de datos, se comprueba si la tabla accedida está incluída en la configuración de alguno de los procesos que se encuentran activos. De ser así se creará un registro con la información de la transacción, el cuál será almacenado, de lo contrario la información no será registrada.

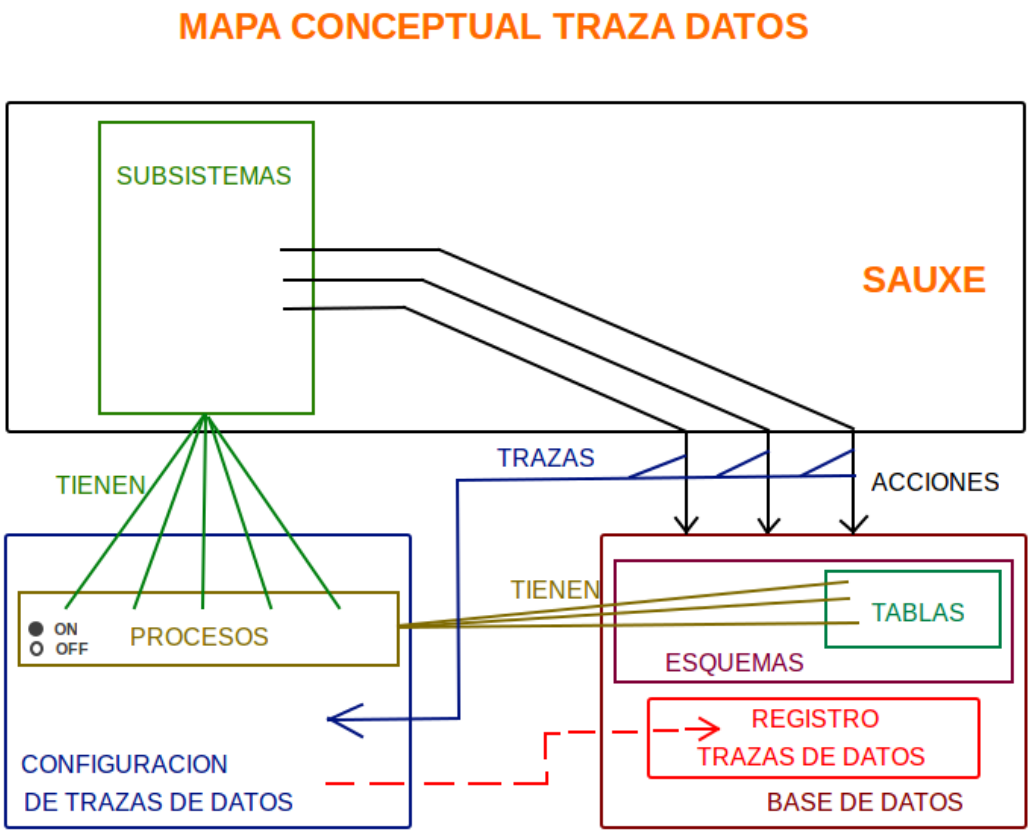


Ilustración 2.2.1 Modelo Conceptual Componente Trazas de Datos

Glosario de Términos

Sauxe: Marco de trabajo Sauxe.

Subsistemas: Cada uno de los módulos (Finanzas, Contabilidad, Logística).

Procesos: Operaciones que realiza el usuario dentro de un subsistema, que contienen diferentes acciones relacionadas para obtener un resultado.

Tablas: Tablas de la base de datos contenidas dentro de esquemas.

Esquemas: Esquemas de la base de datos.

Configuración de Trazas de Datos: Interfaz visual para la gestión de procesos, así como la activación de su respectivo registro de trazas de datos.

Registro de Trazas de Datos: Conjunto de información sobre las acciones realizadas

sobre las tablas de la base de datos, que están relacionadas con algún proceso activo.

Acciones: Las que se realizan sobre la base de datos desde cualquier subsistema.

Trazas: Información de cada acción (usuario, fecha, sentencia SQL, operación, etc.).

2.2.2 Identificación de los requisitos

2.2.2.1 Funcionales

RF1. Adicionar Proceso.

RF2. Modificar Proceso.

RF3. Eliminar Proceso.

RF4. Mostrar Proceso.

RF5. Buscar Proceso.

RF6. Adicionar Tabla.

RF7. Eliminar Tabla.

RF8. Mostrar Tabla.

RF9. Registrar Trazas de Datos.

2.2.2.2 No funcionales (rendimiento, seguridad, software, hardware)

Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación.

Como se ha mencionado con anterioridad, el presente trabajo forma parte de un proceso productivo iniciado por el CEIGE y los resultados que se obtengan formarán parte del marco de trabajo desarrollado en el mismo. Por tanto los requisitos no funcionales con los que debe cumplir la aplicación a desarrollar fueron establecidos por el centro al inicio del proceso de desarrollo, a continuación se describen los más importantes.

Rendimiento

Para un funcionamiento óptimo de la aplicación se seguirán las diferentes técnicas de elaboración de la Web. La eficiencia del producto estará determinada en gran medida por el aprovechamiento de los recursos que se disponen en el modelo cliente/servidor. La aplicación propuesta debe ser rápida y el tiempo de respuesta debe ser rápido; no mayor de 5 segundos para las actualizaciones y 20 para las recuperaciones.

Seguridad

Autenticación y Autorización (Contraseña de acceso). Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos. La atención al sistema así como la salva de la información, se realizará de forma centralizada por el administrador, además el sistema debe mostrar opción de advertencia antes de borrar cualquier elemento o información que pueda existir.

Software

Servidores	Especificaciones
Servidor Aplicaciones Web	Sistema Operativo: Ubuntu Server
	Servidor Web: Apache 2.0
	Librerías Adicionales: PHP 5
Servidor de Base de Datos	Sistema Operativo: Ubuntu Server
	Sistema Gestor de Base de Datos: PostgreSQL 8.3.8
Servidor de Clientes Ligeros	Sistema Operativo: Nova Server
	Navegador Web: Mozilla Firefox 2.2 ó superior.
	Herramientas Ofimáticas
	Visualizador de ficheros pdf.
	Herramienta de administración de clientes ligeros.

Tabla 2 Requisitos de Software para servidores.

Clientes	Especificaciones
PC con Disco	Sistema Operativo: Linux o Windows
	Navegador Web: Mozilla Firefox v2.2 ó superior.
	Herramientas Ofimáticas
	Visualizador de ficheros pdf.
Cliente Ligero	Todo se instala en el servidor de clientes ligeros.

Tabla 3 Requisitos de Software para clientes.

Hardware

	Especificaciones
Servidores	Procesador: 3.00 GHZ
	RAM: 1GB
	Disco duro: 160 GB
	UPS: 1
	Lector de CD: 1
PC con Disco	Procesador: 1.40 GHZ
	RAM: 256 MB (recomendado 512 Mb)
	Tarjeta de Red: 1
Cliente Ligero	Procesador: 2.0 GHZ
	RAM: 256 MB (recomendado 512 Mb)
	Tarjeta de Red: 1

Tabla 4 Requisitos de Hardware.

Político-culturales

Se deberá hacer un uso correcto del idioma español en la Interfaz de la aplicación, con logotipos e imágenes que se encuentren en correspondencia con el carácter de la misma.

Confiabilidad

El subsistema debe ser confiable y preciso en la información que le suministra al usuario para evitar cualquier tipo de error. Estará disponible todo el tiempo, permitiendo el trabajo a los usuarios y las acciones de mantenimiento. Este debe ser estable, fiable y la velocidad de respuesta debe ser rápida durante la utilización del mismo. La información almacenada debe ser confiable en cuanto a su veracidad e integridad desde su recopilación y durante.

Portabilidad

El subsistema será multiplataforma (Linux-Windows) lo que permitirá ejecutarse sobre diferentes sistemas operativos sin importar sus versiones, y sin necesidad de modificar su código fuente.

2.2.3 Descripción de requisitos

Descripción del Requisito Funcional Adicionar Proceso

Precondiciones	**El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción adicionar proceso.** **Debe estar seleccionado un subsistema para poder adicionar el proceso.**
Flujo de eventos	
Flujo básico <<Nombre del flujo básico>>	
1	Seleccionar del árbol de subsistemas el que se le adicionará un proceso.
2	Dar clic en el botón Adicionar del Panel Procesos.
3	Introducir los datos del proceso que se adicionará.
4	Dar clic en el botón Aceptar.
5	El sistema confirma que se adicionó el proceso satisfactoriamente.
Pos-condiciones	
1	Se ha adicionado un nuevo proceso.
Flujos alternativos	
Flujo alternativo 4.a Información errónea	

1	El sistema señala los datos erróneos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 3 del flujo básico.	
Flujo alternativo 4.b Información incompleta		
1	El sistema señala los datos vacíos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 3 del flujo básico.	
Flujo alternativo *.a El administrador cancela la acción		
1	Concluye el requisito	
Pos-condiciones		
1	No se adiciona el Proceso	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual <<Referencia al modelo conceptual en cuestión>>.	
Conceptos	**<<Nombre del concepto>>**	Atributos del concepto que se utilizan en el requisito, tanto internamente como para mostrarlos al usuario. Deben separarse en dos grupos: Visibles en la interfaz: <<Nombre del atributo 1>> Utilizados internamente: <<Nombre del atributo 2>>
Requisitos especiales	Son los requisitos no funcionales específicos para el requisito. Por ejemplo, estándares de intercambio de información.	
Asuntos pendientes	Posibles mejoras al requisito.	

Tabla 5 Descripción del requisito funcional Adicionar Proceso.

Para ver la descripción de los demás requisitos funcionales ver Anexo C.

2.3 Diseño

El modelo de diseño describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Este modelo se puede utilizar para visualizar la implementación y para soportar las técnicas de programación gráfica de la aplicación. Para una mejor calidad del

diseño fueron aplicados patrones de diseño durante la realización de los diagramas de clases permitiendo asignar las responsabilidades a los objetos y diseñar la colaboración entre ellos. Dentro de este epígrafe se verán los elementos que se tuvieron en cuenta durante el diseño de la solución, dando paso así a la exposición de los resultados de este flujo de trabajo, que refleja cómo será implementado el sistema en términos de clases del diseño.

2.3.1 Diseño de clases

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases y de las interfaces en una aplicación, contiene información como asociaciones, atributos, métodos y dependencias. Los mecanismos de diseño se utilizan con el objetivo de simplificar los diagramas de clases. Cada diseñador acorde a sus necesidades establece sus propios mecanismos de diseño, teniendo siempre en cuenta los patrones y estilos seleccionados. Los beneficios que ofrecen los mecanismos de diseño son:

- Mantienen la homogeneidad en el diseño.
- Permiten reutilizar soluciones ya probadas.
- Permiten reutilizar documentación.

Para el presente trabajo se definieron los siguientes mecanismos de diseño:

- Mecanismo de diseño para las páginas clientes.
- Mecanismo de diseño para las clases controladoras.
- Mecanismo de diseño para las clases modelos.

Las clases creadas para el desarrollo de la solución son las de color verde, y las del marco de trabajo que han sido rediseñadas son de color naranja.

Diagrama de clases Configuración de Trazas de Datos

A continuación se muestra el diagrama de clases del diseño basado en estereotipos web que se realizó durante el proceso de desarrollo. El diagrama de clases para la Configuración de las Trazas de datos se realizó teniendo en cuenta las características del marco de trabajo Sauxe, el cuál presenta una arquitectura de 5 capas, de las cuales se utilizarán para el desarrollo de la solución las 3 superiores. El diagrama muestra en la

capa superior o de Presentación la clase Vista, que utiliza la librería ExtJS para la interfaz visual y que contiene los formularios para el envío de mensajes hacia la clase controladora contenida en la capa de negocio, que es la encargada de interactuar con la capa de Acceso a Datos, donde se utilizan las clases de Doctrine para el acceso a estos.

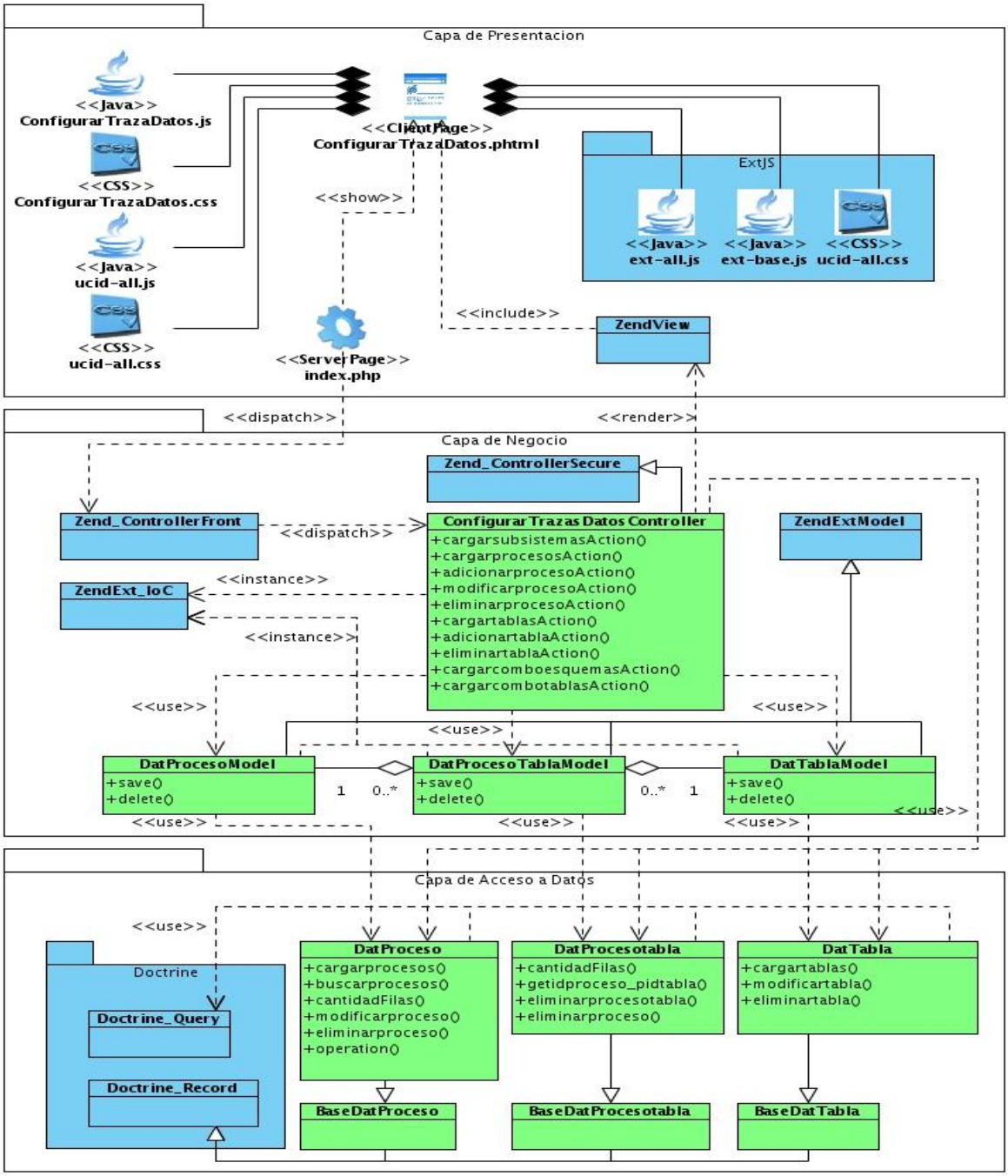


Ilustración 2.3.1 Diagrama de clases del diseño Configuración de Trazas de Datos

Diagrama de clases Registro de Trazas de Datos

Para el diseño de clases del registro se reutilizaron algunas clases del marco de trabajo y se crearon otras nuevas utilizando los patrones Singleton para permitir una única instancia de las clases y Observer para cambiar el comportamiento en dependencia del tipo de evento. Desde la clase Doctrine_Record se dispara un evento luego de realizar una operación sobre la base de datos e instancia la clase ZendExt_Event, la cual se encarga de crear una instancia de la clase observadora que comprueba si la tabla accedida corresponde a un proceso activo utilizando un servicio creado para ese fin. En caso de que así sea, se creará un contenedor con la traza de datos, y se almacenará en la base de datos, en otro caso se descartará esa operación.

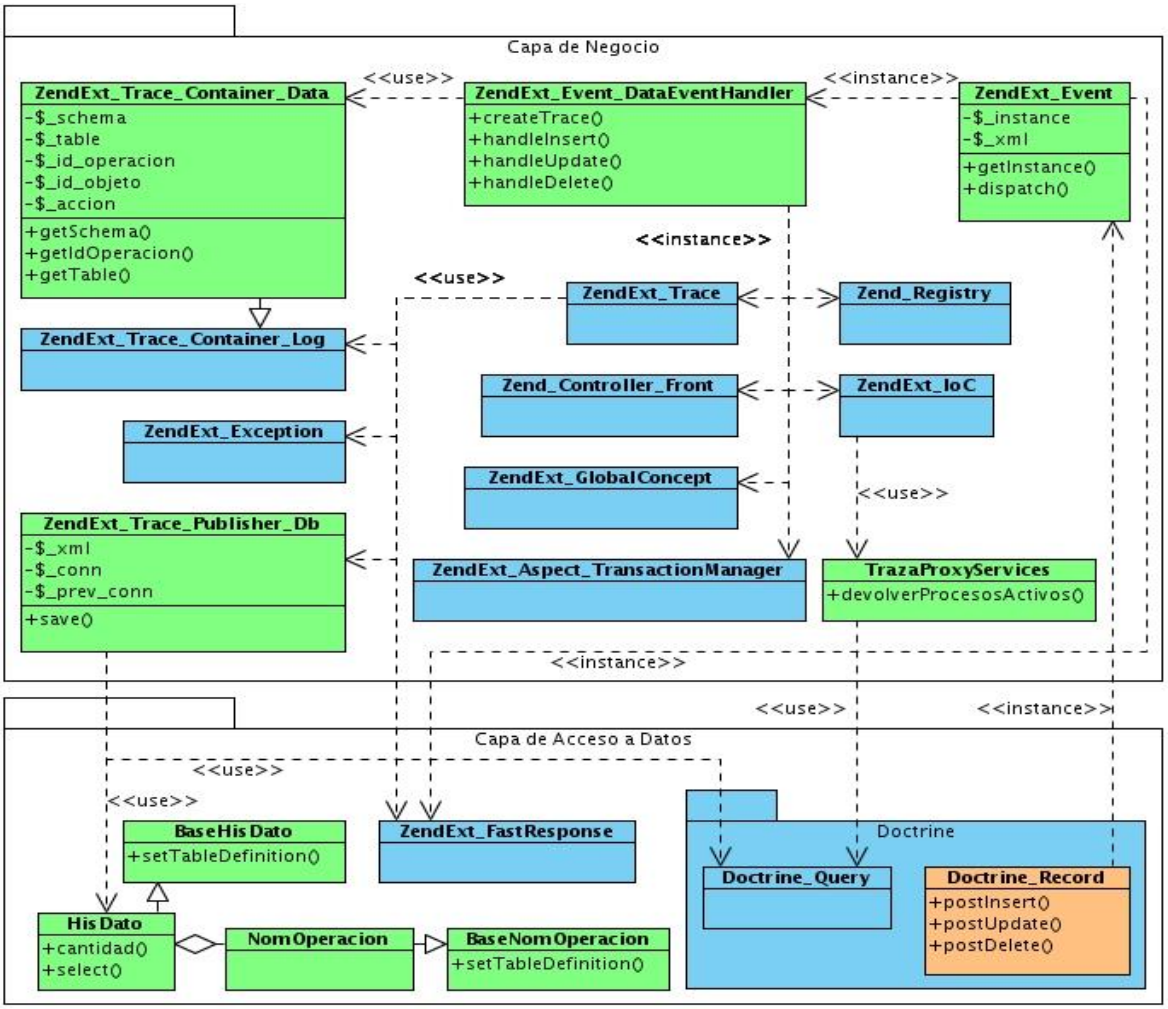


Ilustración 2.3.2 Diagrama de clases del diseño Registro de Trazas de Datos

2.3.2 Modelo de Datos

Para la persistencia de los datos de configuración se utilizaron tres tablas, DatProceso y DatTabla, relacionadas en la tabla DatProcesotabla.

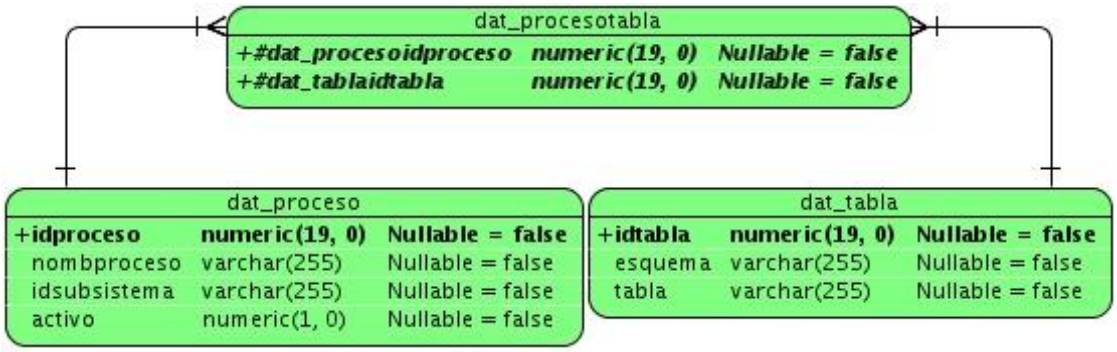


Ilustración 2.3.3 Modelo de datos Configuración de Trazas de Datos

Para la persistencia de las trazas de datos se diseñó un modelo de datos utilizando varias tablas del componente Trazas 1.0, y se creó una tabla para la información relacionada con las trazas de datos y una tabla con las denominaciones de las operaciones.

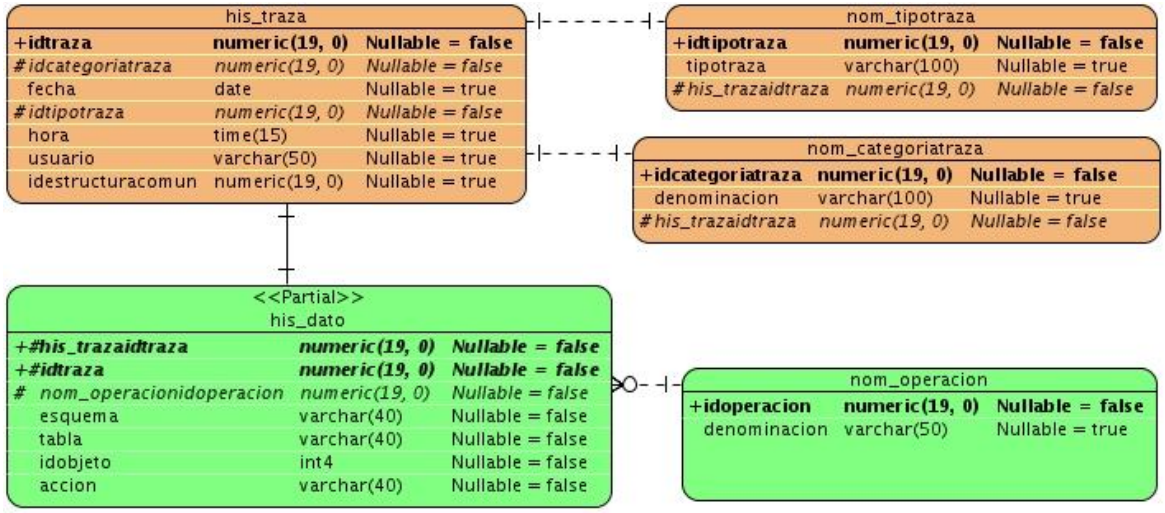


Ilustración 2.3.4 Modelo de datos Registro de Trazas de Datos

2.4 Patrones

Los patrones brindan soluciones ya probadas y documentadas a problemas que están sujetos a contextos análogos en el desarrollo de software. Cada patrón permite que algunos aspectos de la estructura del sistema puedan cambiar independientemente de

otros aspectos. Además posibilitan la reusabilidad, flexibilidad y el mantenimiento. Dichos patrones se clasifican según el propósito para el que han sido definidos en: Creacionales, que solucionan problemas de creación de objetos, ayudan a encapsular y abstraer dicha creación; Estructurales, que solucionan problemas de composición y/o agregación de clases y objetos y de Comportamiento, que brindan soluciones respecto a la interacción y responsabilidades entre clases y objetos, así como los algoritmos que encapsulan. (37)

2.5 Conclusiones Parciales

A lo largo del capítulo se abordaron temas de interés correspondientes a las etapas de análisis y diseño del componente para la configuración visual del registro de trazas de datos en el marco de trabajo Sauxe. Ilustrando el proceso con los principales artefactos que propone el modelo orientado a componentes para estos flujos de trabajo. En este capítulo se han generado los diagramas de procesos, el modelo conceptual relacionando los conceptos más importantes de la herramienta que se quiere desarrollar. Se describió la solución propuesta mediante los requisitos funcionales para así tener una guía de lo que se quiere implementar. Se elaboró el diagrama de clases compuesto por las clases de diseño y sus relaciones, aplicando los patrones de diseño definidos para la solución. También se elaboró el modelo de datos que se utilizará para la persistencia de estos. Los objetivos planteados para la realización del presente capítulo han sido cumplidos, ya que resulta en el análisis y diseño de un componente para la configuración visual del registro de trazas de datos. Una vez concluido, puede darse paso a los flujos de Implementación y Pruebas del mismo.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS

En este capítulo se muestra el modelo de implementación que pone en práctica el diseño de la solución realizado en el capítulo anterior y se especifica el conjunto de validaciones y pruebas que evalúan la calidad del sistema.

3.1 Modelo de Implementación

El modelo de implementación 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 o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros, además de los recursos necesarios para poder ejecutar el sistema desarrollado.

3.1.1 Diagrama de Componentes

El diagrama de componentes representa la estructura física del sistema, su agrupación por paquetes y muestra las dependencias entre estos.

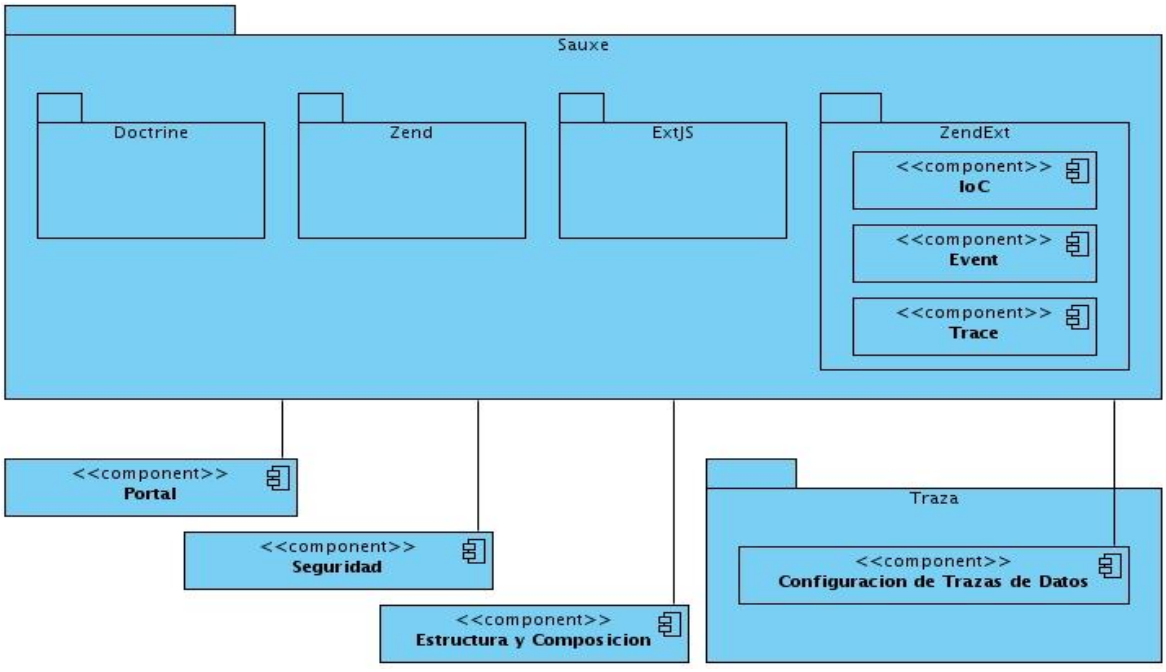


Ilustración 3.1.1 Diagrama de Componentes.

3.1.2 Diagrama de Despliegue

En el diagrama de despliegue se describió la configuración del sistema para su ejecución en un ambiente del mundo real. En una primera variante el usuario desde una PC con disco duro podrá acceder al sistema que estará desplegado en el servidor web. Este a su vez se conectará al servidor de bases de datos que es necesario para el funcionamiento interno del marco de trabajo Sauxe. Las PC con disco tienen ventajas en aplicaciones ricas en multimedia que serían intensivas en ancho de banda si estuvieran completamente residentes en los servidores. (38)

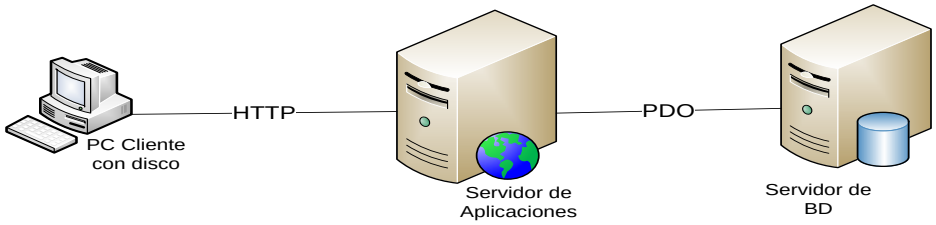


Ilustración 3.1.2 Escenario de despliegue para PC cliente con disco.

En una segunda variante el usuario contará con un cliente ligero conectado a un servidor de clientes ligeros, el cuál será el encargado de acceder al sistema alojado en el servidor web. Y este a su vez se conectará al servidor de bases de datos. Las ventajas de los clientes ligeros pueden incluir un costo de producción más bajo. En un posible ambiente peligroso donde los equipos puedan ser dañados o destruidos, se convierte en ventaja la necesidad de equipos baratos, y de mínimo hardware. Estos además consumen poca energía y hacen poco ruido, lo que implica potenciales ventajas ambientales. (39)

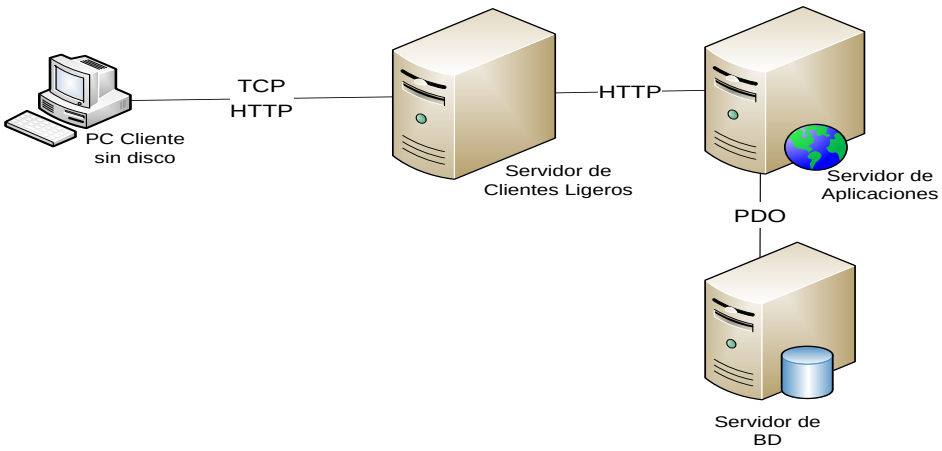


Ilustración 3.1.3 Escenario de despliegue para PC cliente sin disco.

3.2 Estándares de Codificación

Un estándar de codificación comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, que dé la impresión de que fue escrito por un solo programador, esta importante determinación se debe tomar al iniciar un proyecto haciendo que los implementadores trabajen de forma coordinada. La legibilidad del código fuente repercute directamente en el entendimiento que pueda tener otro programador del mismo, aspecto crucial ya que todo software tiene que someterse constantemente a mantenimiento y mejora de sus funcionalidades. El mejor método para lograr que un grupo de desarrolladores mantenga un código de calidad es establecer un estándar de codificación sobre el cual se realizarán revisiones rutinarias.

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.

3.2.1 Nomenclatura de las clases

Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto se pondrá con minúscula, cuando sea un nombre compuesto se utilizará la notación PascalCasing.

Ejemplo: Trace.

3.2.2 Nomenclatura según el tipo de clases

Clases controladoras

Las clases controladoras deben llevar la palabra “Controller” después del nombre.

Ejemplo: ConfigurarTrazaDatosController.

Clases de los modelos

- Bussines

Estas clases que se encuentran dentro de Bussines llevan la palabra “Model” después del nombre.

Ejemplo: DatProcesoModel.

- Domain

Las clases que se encuentran dentro de Domain el nombre que reciben es el de la tabla en la Base de Datos.

Ejemplo: DatProceso.

3.2.3 Nomenclatura de las funciones

Para estas funciones se debe escribir la primera palabra en minúscula, si es un nombre compuesto se empleará la notación CamelCasing y en caso de ser una acción de la clase controladora se debe especificar el nombre de dicha acción en minúscula y seguido del sufijo “Action”.

Ejemplo: cargarProcesosAction.

3.2.4 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	Bool

Tabla 6 Prefijos a utilizar en la creación de variables.

3.2.5 Normas para los comentarios

Se debe comentar todo lo que se haga dentro del desarrollo, establecer las pautas que conlleven a lograr un código más legible y reutilizable y así se pueda aumentar su mantenimiento a lo largo del tiempo.

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:

```
/*
 * ConfigurartrazadatosController
 * Clase Controladora para la configuración de las Trazas de Datos
 *
 * @package Traza
 * @copyright UCID-ERP Cuba
 * @author Yaniel Yero Cardoso
 * @version 1.0-0
 */
```

3.3 Métricas de Software

Cuando los modelos de diseño aumentan en tamaño y complejidad resulta beneficioso

aplicar métricas para verificar la calidad y objetividad del mismo. La visión objetiva de un modelo de diseño Orientado a Objetos (OO) se obtiene a través de la adecuada aplicación de métricas, cuyo resultado aporta el componente cualitativo en la evaluación que se hace. Dentro del diseño OO, se definen nueve características medibles: tamaño, complejidad, acoplamiento, suficiencia, integridad, cohesión, originalidad, similitud y volatilidad. (40) Entre estas características se encuentran:

- **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 de 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, esta 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.

La valoración de las actividades realizadas dentro del modelo de diseño se ha realizado con la aplicación de las métricas ***TOC*** y ***RC***.

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	Aumento del TOC provoca aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Aumento del TOC provoca aumento de la complejidad de implementación de la clase.
Reutilización	Aumento del TOC provoca disminución del grado de reutilización de la clase.

Tabla 7 Atributos de calidad evaluados por la métrica TOC.

Se definieron los siguientes criterios y categorías de evaluación para los atributos de calidad anteriores:

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad de 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 8 Criterios de evaluación para la métrica TOC.

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	Aumento del RC provoca aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Aumento del RC provoca aumento de la complejidad del mantenimiento de la clase.

Reutilización	Aumento del RC provoca disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Aumento del RC provoca aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 9 Atributos de calidad evaluados por la métrica RC.

Se definieron los siguientes criterios y categorías de evaluación para los atributos de calidad anteriores:

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 10 Criterios de evaluación para la métrica RC.

3.3.1 Resultados obtenidos utilizando la métrica TOC

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
DatProcesoModel	3	Baja	Baja	Alta
DatProcesotablaModel	3	Baja	Baja	Alta

DatTablaModel	3	Baja	Baja	Alta
DatProceso	7	Media	Media	Media
DatProcesotabla	5	Baja	Baja	Alta
DatTabla	6	Baja	Baja	Alta
BaseDatProceso	2	Baja	Baja	Alta
BaseDatProcesotabla	2	Baja	Baja	Alta
BaseDatTabla	2	Baja	Baja	Alta
ConfigurartrazadatosController	14	Alta	Alta	Baja
TrazaProxyServices	4	Baja	Baja	Alta
ZendExt_Event_DataEventHandler	4	Baja	Baja	Alta
ZendExt_Trace_Publisher_Db	1	Baja	Baja	Alta
ZendExt_Trace_Container_Data	3	Baja	Baja	Alta
ZendExt_Event	2	Baja	Baja	Alta
Doctrine_Record	3	Baja	Baja	Alta
HisDato	2	Baja	Baja	Alta
BaseHisDato	3	Baja	Baja	Alta
NomOperacion	2	Baja	Baja	Alta
BaseNomOperacion	3	Baja	Baja	Alta

Tabla 11 Instrumento de evaluación de la métrica TOC.

Tabla 12 Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad.

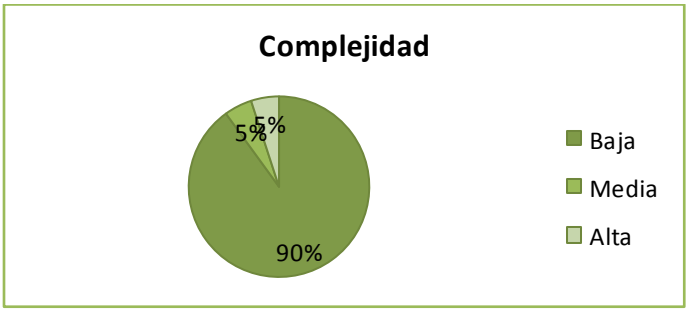


Tabla 13 Resultados de la evaluación de la métrica TOC para el atributo Complejidad.

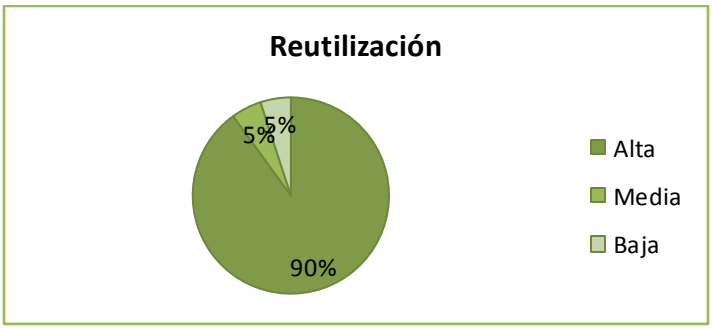


Tabla 14 Resultados de la evaluación de la métrica TOC para el atributo Reutilización.

Luego de la interpretación de los resultados arrojados por la evaluación de la métrica TOC se puede decir que los mismos son satisfactorios y se incluyen dentro del rango calidad aceptable. El 70% de las clases se encuentra en el rango de las evaluaciones positivas en los atributos sobre los que incide esta métrica (responsabilidad, complejidad y reutilización).

3.3.2 Resultados obtenidos utilizando la métrica RC

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
DatProcesoModel	1	Bajo	Baja	Alta	Baja
DatProcesotablaModel	1	Bajo	Baja	Alta	Baja
DatTablaModel	1	Bajo	Baja	Alta	Baja
DatProceso	1	Bajo	Baja	Alta	Baja
DatProcesotabla	1	Bajo	Baja	Alta	Baja
DatTabla	1	Bajo	Baja	Alta	Baja
BaseDatProceso	0	Ninguno	Baja	Alta	Baja
BaseDatProcesotabla	0	Ninguno	Baja	Alta	Baja
BaseDatTabla	0	Ninguno	Baja	Alta	Baja
TrazaProxyServices	1	Bajo	Baja	Alta	Baja
ConfigurartrazadatosController	8	Alto	Alta	Baja	Alta
ZendExt_Event_DataEventHandler	7	Alto	Alta	Baja	Alta
ZendExt_Trace_Publisher_Db	5	Alto	Alta	Baja	Alta
ZendExt_Trace_Container_Data	0	Ninguno	Baja	Alta	Baja
ZendExt_Event	2	Medio	Baja	Alta	Baja
Doctrine_Record	1	Bajo	Baja	Alta	Baja
HisDato	0	Ninguno	Baja	Alta	Baja

BaseHisDato	0	Ninguno	Baja	Alta	Baja
NomOperacion	0	Ninguno	Baja	Alta	Baja
BaseNomOperacion	0	Ninguno	Baja	Alta	Baja

Tabla 15 Instrumento de evaluación de la métrica RC.

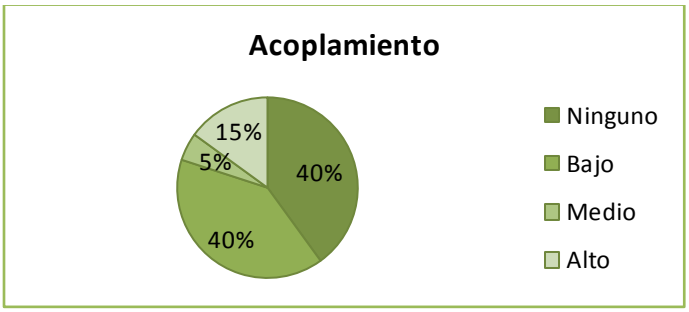


Tabla 16 Resultados de la evaluación de la métrica RC para el atributo Acoplamiento.

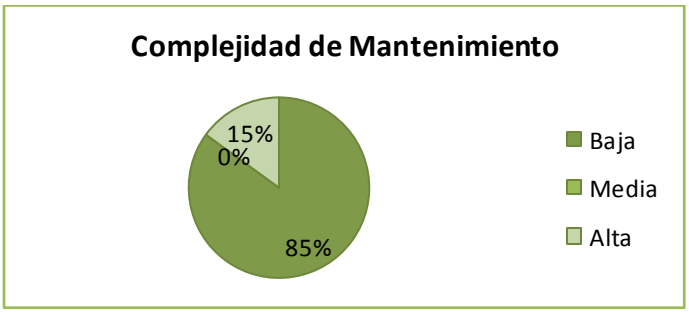


Tabla 17 Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento.

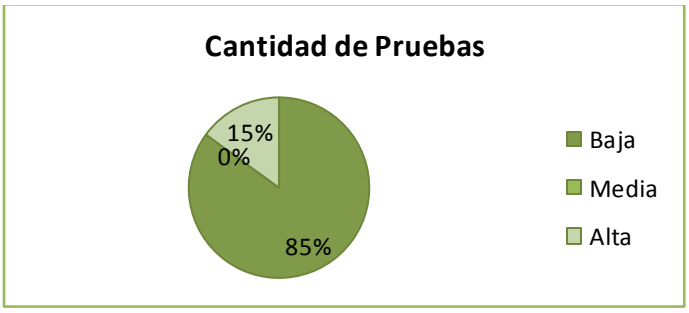


Tabla 18 Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas.

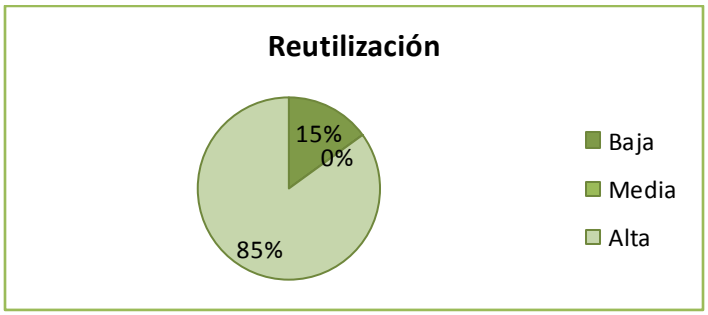


Tabla 19 Resultados de la evaluación de la métrica RC para el atributo Reutilización.

El análisis de los resultados obtenidos en la evaluación con la métrica RC evidencia que es posible concluir que el diseño del componente Trazas de Datos tiene una calidad aceptable. Para fundamentar lo anterior, se puede mencionar que el 80 % de las clases posee menos de 3 dependencias de otras clases. Igualmente los atributos de calidad Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización se comportan satisfactoriamente en un 85 % de las clases.

3.3.3 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 20 Rango de valores para la evaluación de la relación Atributo/Métrica.

Atributo/Métrica	TOC	RC	Promedio
Responsabilidad	1	-	1

Complejidad de Implementación	1	-	1
Reutilización	1	1	1
Acoplamiento	-	1	1
Complejidad de Mantenimiento	-	1	1
Cantidad de pruebas	-	1	1

Tabla 21 Resultados de la evaluación de la relación Atributo/Métrica.

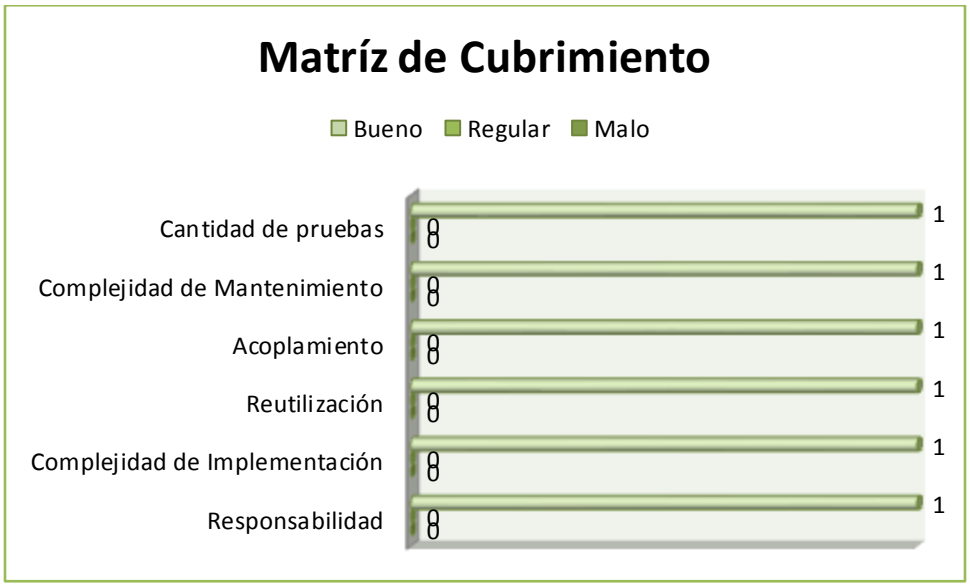


Tabla 22 Matriz de Cubrimiento.

3.4 Pruebas de Software

Las pruebas del software son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. Dichas pruebas son realizadas con el objetivo de detectar errores en el sistema, por lo que se llevan a cabo durante todo el ciclo de vida del producto.

Los casos de prueba especifican una forma de probar el sistema, incluyendo las entradas con las que se ha de probar, las condiciones bajo las que ha de probarse, así como los resultados esperados.

3.4.1 Pruebas estructurales o de “caja blanca”

Consiste en realizar pruebas para verificar que líneas específicas de código funcionan tal como está definido. También se le conoce como prueba de caja-transparente. Esta se basa en el minucioso examen de los detalles procedimentales. 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 gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar. (41)

Estas tienen como objetivos:

- Garantizar que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- Ejercitar todas las decisiones lógicas en las vertientes verdadera y falsa.
- Ejecutar todos los bucles en sus límites operacionales.
- Ejercitar las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de Caja Blanca como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

Entre las técnicas de prueba de Caja Blanca podemos ver:

1. Prueba de Condición: Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.
2. Prueba de Flujo de Datos: Se selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
3. Prueba de Bucles: Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.
4. Prueba del Camino Básico: Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición

de un conjunto básico.

Para el presente trabajo se utilizarán las pruebas de camino básico, la esencia de las mismas es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática.

Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

```
function cargarsubsistemasAction() {
    $idnodo = $this->_request->getPost('node'); //1
    $accion = $this->_request->getPost('accion'); //1
    $idsistema = $this->_request->getPost('idsistema'); //1
    $idservidor = $this->_request->getPost('idservidor'); //1
    $tieneservidor = $this->_request->getPost('existeconexion'); //1
    $idgestor = $this->_request->getPost('idgestor'); //1
    $idbd = $this->_request->getPost('idbd'); //1
    $certif = $this->global->Perfil->certificado; //1
    $entidad = $this->global->Estructura->idestructura; //1
    $sistemaArr = array(); //1

    $subsistemas =  $this->integrator->seguridad->getSistems($certif, $entidad, $idnodo); //1
    $cant = 0; //1
    if (count($subsistemas)) //2
    {
        foreach($subsistemas as $valor) //3
        {
            $sistemaArr[$cant]['idsistema'] = $valor['idsistema']; //4
            $sistemaArr[$cant]['text'] = $valor['text']; //4
            $sistemaArr[$cant]['idpadre'] = $valor['idpadre']; //4
            $sistemaArr[$cant]['icono'] = $valor['icono']; //4
            $sistemaArr[$cant]['id'] = $valor['id'].$valor['idpadre']; //4
            $subsistemashijos =  $this->integrator->seguridad->getSistems($certif, $entidad, $valor['id'].
            $valor['idpadre']); //4
            $sistemaArr[$cant]['leaf'] = $subsistemashijos == null ?true:false; //5 //6 //7
            $cant++; //8
        } //9
    }
    echo json_encode($sistemaArr);return; //10
} //11
```

Ilustración 3.4.1 Código fuente de la funcionalidad cargarsubsistemasAction

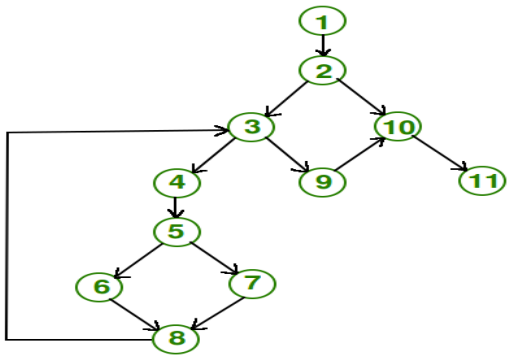


Ilustración 3.4.2 Grafo de flujo asociado a la funcionalidad cargarsubsistemasAction.

Luego de haber construido el grafo se realiza el cálculo de la complejidad ciclomática mediante las tres fórmulas descritas a continuación, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

1. V (G) = (A – N) + 2

Siendo "**A**" la cantidad total de aristas y "**N**" la cantidad total de nodos.

V (G) = (13 - 11) + 2

V (G) = ***4***

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) = 3 + 1

V (G) = ***4***

3. V (G) = R

Siendo "**R**" la cantidad total de regiones, para cada formula "V (G)" representa el valor del cálculo.

V (G) = ***4***

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, dando como resultado 4, lo que indica que existen 4 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: 1-2-3-4-5-6-8-3-9-10-11
- Camino básico #2: 1-2-3-4-5-7-8-3-9-10-11
- Camino básico #3: 1-2-3-9-10-11
- Camino básico #4: 1-2-9-10-11

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento cumpliendo con las siguientes exigencias:

Descripción: Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.

Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.

Entrada: Se muestran los parámetros que entran al procedimiento.

Resultados Esperados: Se expone resultado que se espera que devuelva el procedimiento.

Para cada camino se realizó un caso de prueba:

1. Caso de prueba para el Camino básico #1:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: No hay datos de entrada.

Condición de ejecución: -

Entrada: -

Resultados esperados: [{"idsistema":"2", "text":"Seguridad", "idpadre":null, "icono":null, "id":"2", "leaf":**true**}]

2. Caso de prueba para el Camino básico #2:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: No hay datos de entrada.

Condición de ejecución: -

Entrada: -

Resultados esperados: [{"idsistema":"2", "text":"Seguridad", "idpadre":null, "icono":null, "id":"2", "leaf":**false**}]

3. Caso de prueba para el Camino básico #3:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: No hay datos de entrada.

Condición de ejecución: -

Entrada: -

Resultados esperados: []

4. Caso de prueba para el Camino básico #4:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: No hay datos de entrada.

Condición de ejecución: -

Entrada: -

Resultados esperados: []

3.4.2 Pruebas funcionales o de "caja negra"

La prueba de Caja Negra se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. (41)

La misma no es una alternativa a las técnicas de prueba anteriormente vistas, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca.

Estas pruebas permiten encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.

- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de los estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien.

Para esta prueba, existen varias técnicas, entre ellas están:

1. Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
2. Técnica del Análisis de Valores Límites: esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Dentro del método de Caja Negra la técnica de la Partición de Equivalencia es una de las más efectivas, pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar. En este apartado se hará uso de esta técnica para aplicarle a una funcionalidad del sistema desarrollado el método de caja negra.

Requisito a probar: Adicionar Proceso

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario

Adicionar proceso.	Se realizará la adición de los atributos del proceso.	EP 1.1: Adicionar proceso introduciendo datos válidos.	– En este escenario se debe seleccionar un subsistema. – Presionar el botón Adicionar en el panel Procesos. – En el formulario deberá introducir los datos correspondientes a: nombre del proceso y la descripción, y marcar o desmarcar el checkbox 'Activo', se mostrará error en los campos con datos incorrectos. – Debe presionar el botón Aceptar. – El sistema muestra un mensaje, 'Se ha insertado satisfactoriamente.'. – Presiona el botón Aceptar.
		EP 1.2: Adicionar proceso introduciendo datos inválidos.	– En este escenario se debe seleccionar un subsistema. – Presionar el botón Adicionar del panel Procesos. – Luego se introducen datos inválidos y se presiona el botón aceptar. – El sistema no permite adicionar datos inválidos y muestra un mensaje de error, 'Datos incorrectos.'. – Se presiona el botón Aceptar. – Se procede a introducir los datos válidos.
		EP 1.3: Adicionar proceso dejando campos vacíos.	– En este escenario se debe seleccionar un subsistema. – Presionar el botón Adicionar del panel Procesos. – Luego se presiona el botón aceptar dejando cualquiera los campos vacíos. – El sistema mostrará un mensaje para que llene los campos vacíos 'Datos incorrectos.'. – Presionar el botón Aceptar. – Se procede a introducir los datos válidos.

	EP 1.6: Cancelar.	– En este escenario se debe seleccionar un subsistema. – Presionar el botón Adicionar del panel Procesos. – Se introducen los datos del proceso. – Se presiona en el botón Cancelar. – Se cierra la ventana sin realizar ninguna operación.

Tabla 23 Escenario de prueba para el requisito Adicionar Proceso.

No	Nombre de campo	Tipo	Válido	Inválido	Inválido	Inválido	Inválido
1	Proceso	campo de texto	Alfabeto inglés	Vacío	Textos con espacio	Textos con números y/o caracteres especiales	Textos de longitud mayor que 40
2	Descripción	campo de texto	Alfanumérico, espacio, signo de punto (.)	Vacío	Textos de longitud mayor que 255	N/A	N/A
3	ID Subsistema	campo de texto	Numérico (Predeterminado)	N/A	N/A	N/A	N/A
4	Activo	Checkbox	Seleccionado Deseleccionado	N/A	N/A	N/A	N/A

Tabla 24 Descripción de Variables.

Id del escenario	Escenario	Variable1	Variable2	Variable3	Variable4	Respuesta del sistema	Resultado de la prueba
EP 1.1	Adicionar proceso introduciendo datos válidos.	V(NomUNO)	V(Proceso número 1.)	V(2)	V(Checked) V(Unchecked)	El sistema almacena el proceso. Se muestra un mensaje de información, 'Se ha insertado satisfactoriamente'.	

EP 1.2	Adicionar proceso introduciendo datos inválidos.	I($%í _6&ĝ)	I("$%G:_* &ĝ)	N/A	N/A	El sistema muestra un mensaje de error, 'Datos incorrectos.'
EP 1.3	Adicionar proceso dejando campos vacíos.	Vacío	Vacío	N/A	N/A	El sistema muestra un mensaje para que llene los campos correctamente, 'Datos incorrectos'.
EP 1.6	Cancelar	N/A	N/A	N/A	N/A	Se cierra la ventana sin realizar ninguna operación.

Tabla 25 Juego de datos a probar.

La solución fue probada por el Departamento de Calidad del CEIGE, donde se comprobó el correcto funcionamiento de la misma, lo cual queda avalado por el acta de liberación de la herramienta expuesta en el Anexo A.

En el Anexo B se puede ver la interfaz visual para la Configuración de las Trazas de Datos.

3.5 Conclusiones Parciales

En este capítulo fueron expuestos los artefactos generados como parte del modelo de implementación de la solución propuesta, tales como el diagrama de componentes y los estándares de codificación. Se realizó una validación del diseño expuesto en el capítulo anterior mediante la aplicación de métricas para la evaluación de atributos de calidad, lo cual permitió valorar el diseño propuesto de muy bueno dado los excelentes resultados obtenidos. Además se describieron las pruebas estructurales y funcionales realizadas que permitieron comprobar el correcto funcionamiento del sistema.

CONCLUSIONES

Durante el desarrollo del presente trabajo se llevaron a cabo una serie de fases que permitieron dar cumplimiento al objetivo planteado y que abarcaron todas las tareas investigativas propuestas, llegando a las siguientes conclusiones:

- Se realizó un estudio del arte que permitió sentar las bases para el desarrollo de la investigación identificando conceptos fundamentales, herramientas y tecnologías relacionadas con la misma.
- Se desarrolló la solución para el registro y configuración de las trazas de datos empleando tecnologías y herramientas libres.
- La aplicación de las métricas Tamaño Operacional de Clase y Relaciones entre Clases permitió validar el diseño propuesto.
- La realización de pruebas de caja negra y caja blanca permitió comprobar el buen funcionamiento de la aplicación.
- La solución desarrollada apoya las tareas de auditoría relacionadas con el acceso a las bases de datos facilitando el análisis para la detección de violaciones de seguridad.

La solución obtenida permitirá que los sistemas desarrollados sobre el marco de trabajo Sauxe puedan contar con un mecanismo para el registro y monitoreo de las acciones realizadas sobre la base de datos a través este, lo que permitirá detectar posibles violaciones sobre la base de datos.

RECOMENDACIONES

Con vistas a mejorar la solución en futuras versiones se recomienda:

- Aumentar la eficiencia en la gestión de la Configuración.
- Brindar la posibilidad de que los procesos permitan contener subprocesos.
- Desarrollar un módulo de análisis de las trazas de datos para identificar cuellos de botella y otros aspectos que puedan atentar contra el rendimiento y la seguridad del sistema.

BIBLIOGRAFÍA

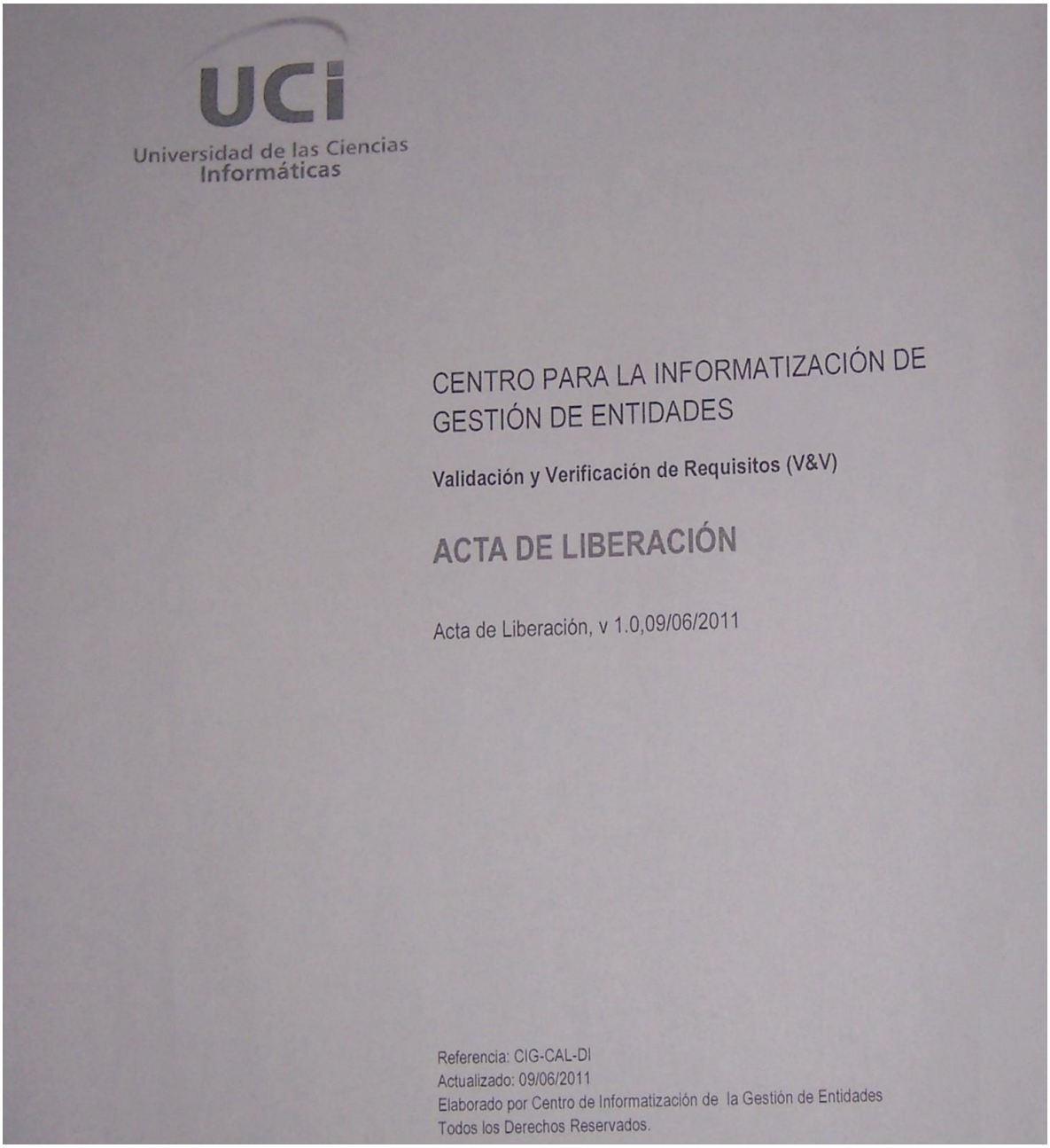
1. **Iberia, Sescoi.** Las empresas que gestionan proyectos necesitan soluciones ERP especializadas. [En línea] 2010. http://www.workplan-enterprise.com/fileadmin/pdf/myworkplan/WhitePaper_ERP-JobManag-2_SP.pdf.
2. **UVA.** Introducción a las aplicaciones web. [En línea] http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node11.html.
3. **DDW.** Desarrollo de Web. *DDW.* [En línea] http://www.desarrollodeweb.com.ar/archivo/25-como-encargar-un-proyecto-web.
4. **ISO.** *Norma Internacional ISO 9001 - Sistemas de gestión de la calidad – Requisitos.* Ginebra : Secretaría Central de ISO, 2000.
5. **Barros, Oscar.** *Reingeniería de Procesos de negocio*. Chile : Editorial Dolmen, 1994.
6. **Davenport, Thomas.** *Process Innovation*. s.l. : Harvard Business School Press, 1993.
7. **Loyola, William.** *Business Process Modelling*. 2006.
8. **EXXA-UNNE.** ¿Qué es seguridad? [En línea] http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/SEGUNIX012.htm.
9. **Proyecto Teleco.** Wikispaces.com. *AAA*. [En línea] Septiembre de 2010. http://proyecto-teleco-2010.wikispaces.com/file/view/Marco+teorico+AAA.pdf.
10. **Delgado Picazo, Mario.** *Análisis de impacto y desarrollo de buenas prácticas de auditoría en bases de datos Oracle 11G*. Madrid : s.n., 2009.
11. Auditoría de una base de datos Oracle. *ajpdsoft.com*. [En línea] http://www.ajpdsoft.com/.
12. **CONCHAMBAY MUZO, MAYRA DEL PILAR y VARELA BOLAÑOS, GABRIELA FERNANDA.** *Herramienta para la extracción de información de auditoría para aplicaciones implementadas sobre bases de datos Oracle 9i y 10G*. Quito : s.n., 2007.
13. **Application LifeCycle Solutions, S.L.** Application LifeCycle Solutions, S.L. *als-es.com*. [En línea] Septiembre de 2006. http://www.als-es.com/home.php?location=recursos/articulos/auditar-sistemas-informacion.
14. **Corrales Martínez, Yormis.** *Componentes para la configuración de la gestión del multilenguaje y la gestión de trazas del sistema Cedrux*. Ciudad de la Habana : s.n., 2009.
15. [En línea] http://apexsqllogreview.com/apexsql-log-limitations/#more-40.
16. [En línea] http://apexsqllogreview.com/buying-apexsql-log/.
17. **Vegas, Jesús.** Oracle: Seguridad. [En línea] http://www.infor.uva.es/~jvegas/cursos/bd/oraseg/oraseg.html#4.
18. **Reyes, Rodolfo.** Como_hago_auditoria_en_Oracle_9i. *Como_hago_auditoria_en_Oracle_9i*. [En línea] http://www.lawebdelprogramador.com/foros/Oracle/932372-Como_hago_auditoria_en_Oracle_9i__.html.
19. [En línea] http://www.scmagazineus.com/lumigent-audit-db/review/975/.
20. **Oracle.** Oracle® Audit Vault Administrator's Guide. *Oracle.com*. [En línea] 2008. http://download.oracle.com/docs/cd/E13850_01/doc.102/e13841/toc.htm.
21. —. Oracle Technology Global Price List. Software Investment Guide. *Oracle.com*. [En línea] Enero de 2008. http://www.oracle.com/corporate/pricing/ePLext.PDF.
22. **Pérez Sarduy, 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.
23. **Gómez Baryolo, Oiner, Tenrero Cabrera, Marianela y Silega Martínez, Nemuris.** *Plantilla Registro de la Propiedad intelectual (Sauxe)*. La Habana : s.n., 2008.
24. **Gómez Baryolo, Oiner.** *Solución Informática de Autorización en entornos multientidad y multisistema*. Ciudad de la Habana : s.n., 2010.
25. *Secure Programming with the Zend-Framework*. **Esser, Stefan.** Amsterdam : s.n., 2009.
26. **Celis, Ismael.** Active Record. *ESTADOBETA desarrollo web con estándares*. [En línea] 2 de Mayo de 2006. http://www.estadobeta.com/2006/05/02/active-record/.
27. Doctrine. *Doctrine*. [En línea] 2008. http://www.doctrine-project.org.
28. **Frederick, Shea, Ramsay, Colin y Blades, Steve 'Cutter'.** *Learning Ext JS*. Birmingham - Mumbai : PACKT, 2008. 978-1-847195-14-2.
29. **Grupo de documentación de PHP.** *Manual de PHP*. 2001.

30. Área temática de Linux. *Portal de Linux - Ciberaula*. [En línea] http://linux.ciberaula.com.
31. NetBeans Docs & Support. [En línea] http://netbeans.org/kb/index.html.
32. **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.
33. **Systems Popkin, Software.** Modelado de Sistemas con UML. [En línea] 2008. http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf.
34. **Gamma, Erich, y otros, y otros.** *Design Patterns: Elements of Reusable Object-Oriented Software*. s.l. : Addison-Wesley, 1994. 0-201-63361-2.
35. **Universidad Politécnica de Madrid.** UPM.es. *UPM.es*. [En línea] http://is.ls.fi.upm.es/docencia/proyecto/docs/patrones_gof.pdf.
36. **OWASP.** *Una Guía para Construir Aplicaciones y Servicios Web Seguros*. 2005.
37. **Lago, Ramiro.** Patrones de Diseño Software. *Proactiva-Calidad*. [En línea] 2007. http://www.proactiva-calidad.com/java/patrones/index.html.
38. LanCore. *Sourceforge.net*. [En línea] http://lancore.sourceforge.net/es/.
39. Thin clients. *jeuazarru.com*. [En línea] www.jeuazarru.com/docs/Thin_clients.pdf.
40. **Milestone, C.** Milestone. *Milestone*. [En línea] http://www.milestone.com.mx/CursoModeladoNegociosBPMN.htm.
41. **RIZZI, I. F. M.** Complejidad Ciclomática. *ITBA.edu.ar*. [En línea] 2006. http://www.itba.edu.ar/capis/rtis/articulosdeloscuadernosetapaprevia/RIZZICOMPLEJIDAD.Pdf.
42. Requerimientos de seguridad. *Cantabria*. [En línea] http://amap.cantabria.es/confluence/display/DOCS/Requerimientos+de+Seguridad.
43. **Pinto, Fred.** Retos en Seguridad de Base de Datos. [En línea] http://www.acis.org.co/fileadmin/Base_de_Conocimiento/X_JornadaSeguridad/ConferenciaFredPinto.pdf.
44. **Restrepo Gómez, Jorge Alberto.** Guía para la auditoría de sistemas. [En línea] http://jorgearestrepog.comunidadcoomeva.com/blog/index.php?/archives/4-GUIA-PARA-LA-CLASE-DE-AUDITORIA-DE-SISTEMAS.html.
45. **Pressman, Roger S.** *Ingeniería de Software, un enfoque práctico*. s.l. : McGraw-Hill Companies, 2002. 8448132149.
46. PostgresSQL - es Portal en español sobre PostgresSQL. [En línea] [Citado el: 2 de 12 de 2011.] http://www.postgresql-es.org/sobre_postgresql.
47. **Daylin Real Madrigal, Reinaldo Alain Hernández Valdés.** *Sistema de Log de Auditoría*. La Habana : s.n., 2008.
48. **Gómez Baryolo, Oiner, Tenrero Cabrera, Marianela y Nemuris, Silega Martínez.** *Plantilla Registro de la Propiedad intelectual (Sauxe)*. La Habana : s.n., 2008.
49. **Gonzales, Benjamín.** Zend Frameworks DES.com. *Zend Frameworks DES.com*. [En línea] Zend Technologies Inc, 28 de 11 de 2010. [Citado el: 14 de 1 de 2011.] http://manual.zfdes.com/es/introduction.overview.html.
50. **Group, Object Management.** Unified Modeling Language. *Unified Modeling Language*. [En línea] OMG, 2010 de 10 de 21. [Citado el: 10 de 1 de 2011.] http://www.uml.org/.
51. *Un método para el diseño de la base de datos a partir del modelo orientado a objetos*. **Hernández González, Dra. Anaisa.** 4, México DF : s.n., 2004, Computación y Sistemas, Vol. 7. 1405-5546.
52. **Skarmeta, D.A.F.G.** *Definición de una infraestructura de control de acceso basada en la arquitectura AAA y el uso de credenciales de autorización*. 2006.
53. **ISO/IEC.** WWW.ISO27000.ES. *WWW.ISO27000.ES*. [En línea] http://WWW.ISO27000.ES.
54. **Collins-Sussman, Ben, W. Fitzpatrick, Brian y Pilato, C. Michael.** Version Control with Subversion. [En línea] 2002. [Citado el: 17 de 01 de 2011.] http://svnbook.red-bean.com/nightly/es/svn-book.pdf.
55. **Boda, Pedro, y otros, y otros.** *Zend Framework Manual en español*. 2009.
56. Visual Paradigm. [En línea] [Citado el: 24 de 11 de 2010.] http://www.visual-paradigm.com/product/vpuml/.
57. **Producción, Equipo de.** Modelo de Desarrollo orientado a componentes del proyecto ERP-CUBA.
58. **Bauta Camejo, René R. y Torres Galvez, Ariel.** Herramienta para el registro y monitoreo de trazas en el sistema Cedrux. [En línea] http://semanatecnologica.fordes.co.cu/public/site/315.pdf.
59. **OWASP.** *Una Guía para Construir Aplicaciones y Servicios Web Seguros*. 2005.

60. **Salva, S., Laurenco, P. y Rabhi, I.** *An Approach Dedicated for Web Service Security Testing*. s.l. : IEEE, 2010.
61. **Myers, J., Grimaila, M.R. y F., M.R.** *Log-Based Distributed Security Event Detection Using Simple Event Correlator*. Hawaii : s.n., 2011.
62. **OWASP.** *OWASP Top Ten Release Candidate for 2010*. 2010.
63. **Steven, J. y Peterson, G.** *A Metrics Framework to Drive Application Security Improvement*. s.l. : IEEE, 2007.
64. **Suárez, D.** *ISO 27000: Garantizar la Seguridad de la Información*. s.l. : Dintel, 2008.
65. **Dolphin, M.** *Introduction to ISO 27002 and friends*. s.l. : ISACA, 2005.

ANEXOS

Anexo A. **Acta de liberación de la solución.**



Universidad de las Ciencias Informáticas

CENTRO PARA LA INFORMATIZACIÓN DE GESTIÓN DE ENTIDADES

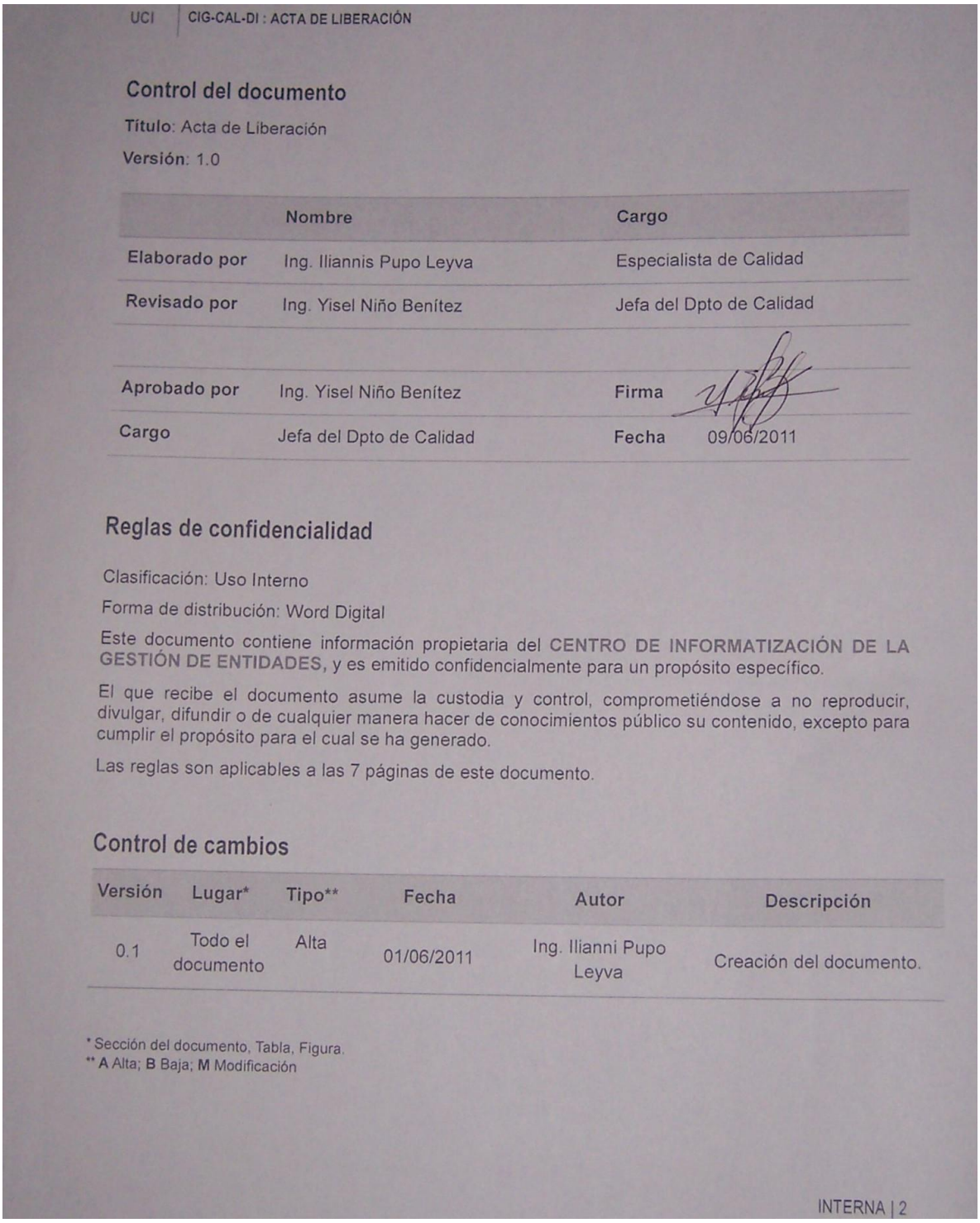
Validación y Verificación de Requisitos (V&V)

ACTA DE LIBERACIÓN

Acta de Liberación, v 1.0,09/06/2011

Referencia: CIG-CAL-DI
Actualizado: 09/06/2011
Elaborado por Centro de Informatización de la Gestión de Entidades

Ilustración 3.5.1 Acta de liberación página 1.



UCI | CIG-CAL-DI : ACTA DE LIBERACIÓN

Control del documento

Título: Acta de Liberación

Versión: 1.0

	Nombre	Cargo
Elaborado por	Ing. Iliannis Pupo Leyva	Especialista de Calidad
Revisado por	Ing. Yisel Niño Benítez	Jefa del Dpto de Calidad
Aprobado por	Ing. Yisel Niño Benítez	**Firma**
Cargo	Jefa del Dpto de Calidad	**Fecha** 09/06/2011

Reglas de confidencialidad

Clasificación: Uso Interno

Forma de distribución: Word Digital

Este documento contiene información propietaria del **CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DE ENTIDADES**, y es emitido confidencialmente para un propósito específico.

El que recibe el documento asume la custodia y control, comprometiéndose a no reproducir, divulgar, difundir o de cualquier manera hacer de conocimientos público su contenido, excepto para cumplir el propósito para el cual se ha generado.

Las reglas son aplicables a las 7 páginas de este documento.

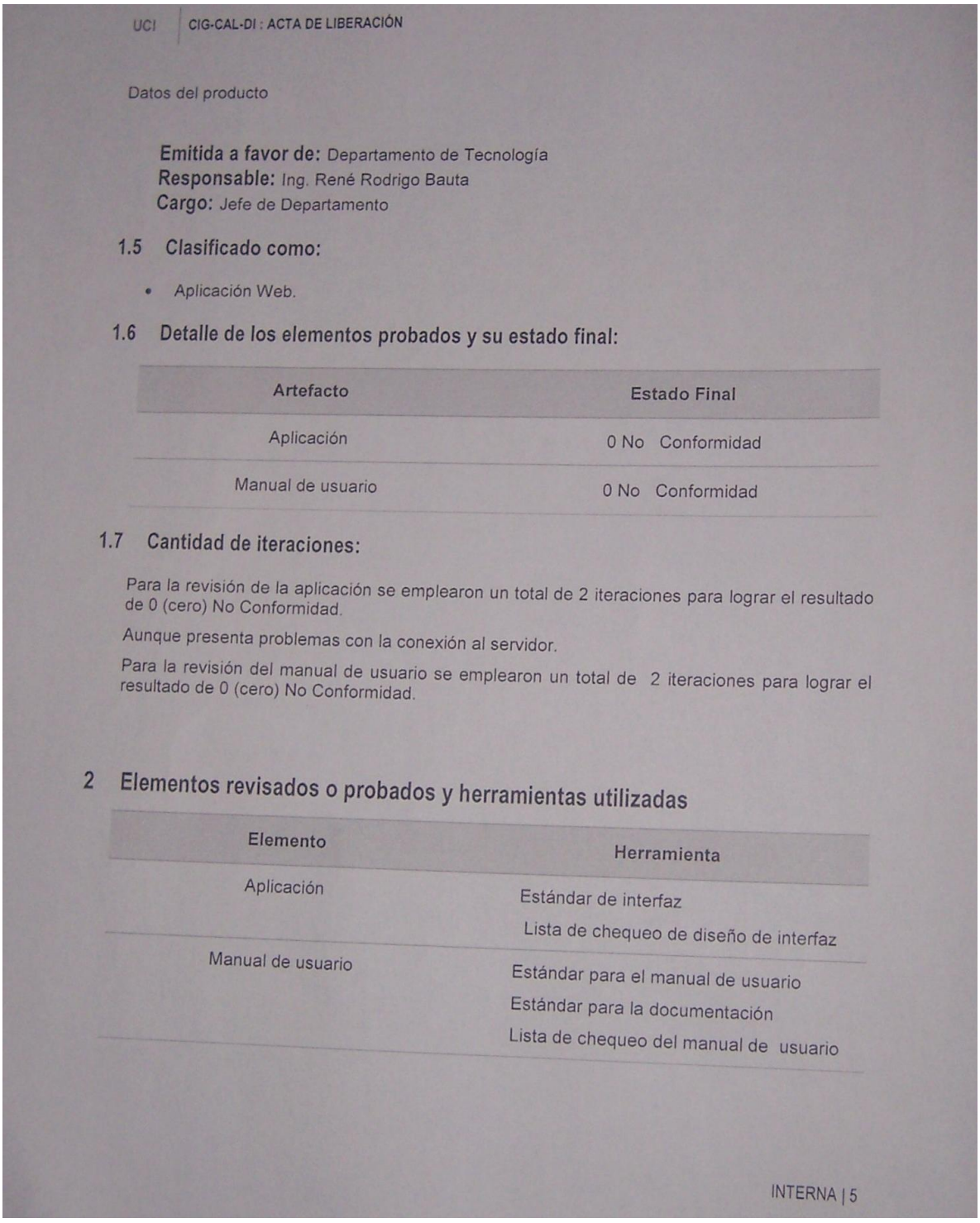
Control de cambios

Versión	Lugar*	Tipo**	Fecha	Autor	Descripción
0.1	Todo el documento	Alta	01/06/2011	Ing. Ilianni Pupo Leyva	Creación del documento.

* Sección del documento, Tabla, Figura.
** **A** Alta; **B** Baja; **M** Modificación

INTERNA | 2

Ilustración 3.5.2 Acta de liberación página 2.



UCI | CIG-CAL-DI : ACTA DE LIBERACIÓN

Datos del producto

Emitida a favor de: Departamento de Tecnología
Responsable: Ing. René Rodrigo Bauta
Cargo: Jefe de Departamento

1.5 Clasificado como:

- Aplicación Web.

1.6 Detalle de los elementos probados y su estado final:

Artefacto	Estado Final
Aplicación	0 No Conformidad
Manual de usuario	0 No Conformidad

1.7 Cantidad de iteraciones:

Para la revisión de la aplicación se emplearon un total de 2 iteraciones para lograr el resultado de 0 (cero) No Conformidad.

Aunque presenta problemas con la conexión al servidor.

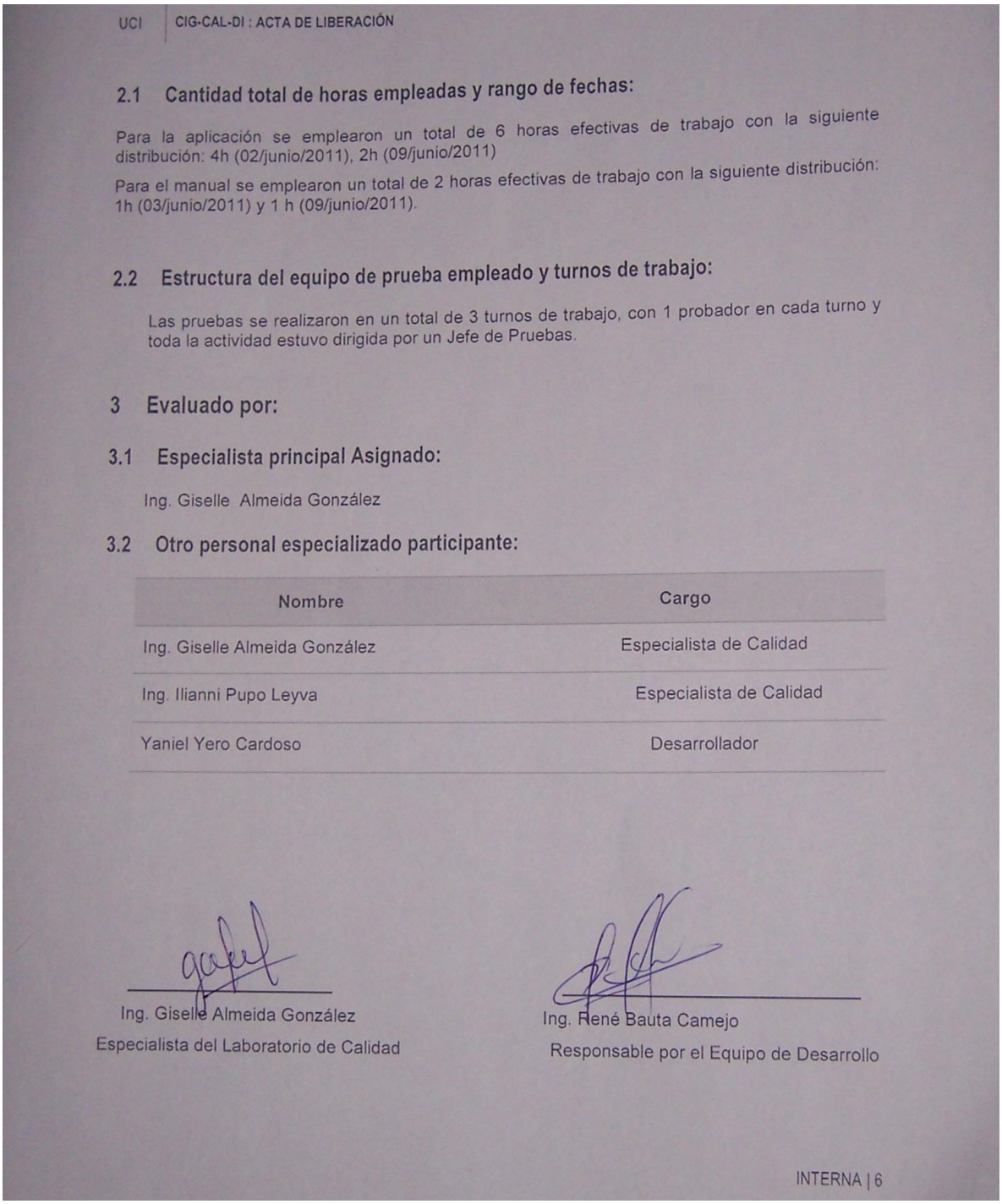
Para la revisión del manual de usuario se emplearon un total de 2 iteraciones para lograr el resultado de 0 (cero) No Conformidad.

2 Elementos revisados o probados y herramientas utilizadas

Elemento	Herramienta
Aplicación	Estándar de interfaz
	Lista de chequeo de diseño de interfaz
Manual de usuario	Estándar para el manual de usuario
	Estándar para la documentación
	Lista de chequeo del manual de usuario

INTERNA | 5

Ilustración 3.5.3 Acta de liberación página 5.



UCI | CIG-CAL-DI : ACTA DE LIBERACIÓN

2.1 Cantidad total de horas empleadas y rango de fechas:

Para la aplicación se emplearon un total de 6 horas efectivas de trabajo con la siguiente distribución: 4h (02/junio/2011), 2h (09/junio/2011)

Para el manual se emplearon un total de 2 horas efectivas de trabajo con la siguiente distribución: 1h (03/junio/2011) y 1 h (09/junio/2011).

2.2 Estructura del equipo de prueba empleado y turnos de trabajo:

Las pruebas se realizaron en un total de 3 turnos de trabajo, con 1 probador en cada turno y toda la actividad estuvo dirigida por un Jefe de Pruebas.

3 Evaluado por:

3.1 Especialista principal Asignado:

Ing. Giselle Almeida González

3.2 Otro personal especializado participante:

Nombre	Cargo
Ing. Giselle Almeida González	Especialista de Calidad
Ing. Ilianni Pupo Leyva	Especialista de Calidad
Yaniel Yero Cardoso	Desarrollador

Ing. Giselle Almeida González
Especialista del Laboratorio de Calidad

Ing. René Bauta Camejo
Responsable por el Equipo de Desarrollo

INTERNA | 6

Ilustración 3.5.4 Acta de liberación página 6.

Anexo B. **Interfaz de Usuario “Configuración de Traza de Datos”**

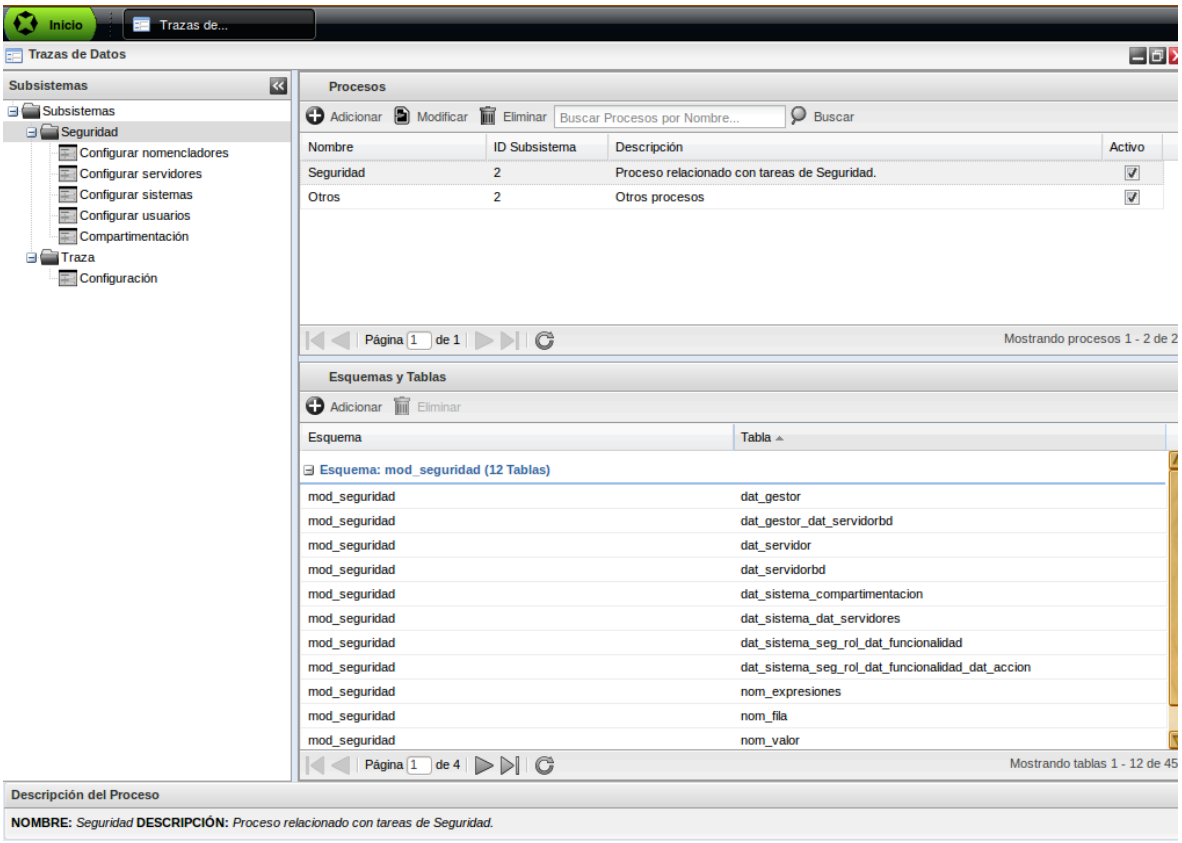


Ilustración 3.5.5 Interfaz de Usuario "Configuración de Traza de Datos".

Anexo C. **Descripción de requisitos**

Descripción del requisito funcional “Buscar Proceso”

Precondiciones	**El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción modificar proceso.** **Debe estar seleccionado un subsistema para poder buscar los procesos.**
Flujo de eventos	
Flujo básico <<Nombre del flujo básico>>	
1	Introducir el criterio de búsqueda en el campo de texto “Buscar procesos”.
Pos-condiciones	
1	Se muestran los procesos que cumplen con el criterio de búsqueda.
Flujos alternativos	

Pos-condiciones		
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual <<Referencia al modelo conceptual en cuestión>>.	
Conceptos	**<<Nombre del concepto>>**	Atributos del concepto que se utilizan en el requisito, tanto internamente como para mostrarlos al usuario. Deben separarse en dos grupos: Visibles en la interfaz: <<Nombre del atributo 1>> Utilizados internamente: <<Nombre del atributo 2>>
Requisitos especiales	Son los requisitos no funcionales específicos para el requisito. Por ejemplo, estándares de intercambio de información.	
Asuntos pendientes	Posibles mejoras al requisito.	

Tabla 26 Descripción del requisito funcional Buscar Proceso.

Descripción del requisito funcional "Adicionar Tabla"

Precondiciones	**El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción adicionar tabla.** **Debe estar seleccionado un proceso para poder adicionar la tabla.**
Flujo de eventos	
Flujo básico <<Nombre del flujo básico>>	
1	Seleccionar del panel de procesos el que se le adicionará una tabla.
2	Dar clic en el botón Adicionar del Panel Tablas.
3	Introducir los datos de la tabla que se adicionará.
4	Dar clic en el botón Aceptar.
5	El sistema confirma que se adicionó la tabla satisfactoriamente.
Pos-condiciones	
1	Se ha adicionado una nueva tabla.
Flujos alternativos	
Flujo alternativo 4.a Información errónea	
1	El sistema señala los datos erróneos y permite corregirlos.
2	El usuario corrige los datos.

3	Volver al paso 3 del flujo básico.	
Flujo alternativo 4.b Información incompleta		
1	El sistema señala los datos vacíos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 3 del flujo básico.	
Flujo alternativo *.a El administrador cancela la acción		
1	Concluye el requisito	
Pos-condiciones		
1	No se adiciona la Tabla	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual <<Referencia al modelo conceptual en cuestión>>.	
Conceptos	**<<Nombre del concepto>>**	Atributos del concepto que se utilizan en el requisito, tanto internamente como para mostrarlos al usuario. Deben separarse en dos grupos: Visibles en la interfaz: <<Nombre del atributo 1>> Utilizados internamente: <<Nombre del atributo 2>>
Requisitos especiales	Son los requisitos no funcionales específicos para el requisito. Por ejemplo, estándares de intercambio de información.	
Asuntos pendientes	Posibles mejoras al requisito.	

Tabla 27 Descripción del requisito funcional Adicionar Tabla.

Descripción del requisito funcional "Registrar Traza de Datos"

Precondiciones	**El usuario se ha autenticado en el sistema y tiene permisos para ejecutar las acciones sobre la base de datos.**
Flujo de eventos	
Flujo básico <<Nombre del flujo básico>>	
1	El usuario realiza una acción sobre la base de datos.
2	El sistema comprueba si la tabla y esquema sobre las cuales se realizó la acción están relacionados con un proceso en estado Activo.

3	El sistema registra la acción realizada sobre la base de datos con sus atributos.	
Pos-condiciones		
1	Se ha registrado una nueva traza de datos.	
Flujos alternativos		
Flujo alternativo 2.a Tabla y Esquema no relacionado con ningún proceso activo.		
1	El sistema no registra la traza de datos.	
Pos-condiciones		
1	No se registra una nueva traza de datos.	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual <<Referencia al modelo conceptual en cuestión>>.	
Conceptos	**<<Nombre del concepto>>**	Atributos del concepto que se utilizan en el requisito, tanto internamente como para mostrarlos al usuario. Deben separarse en dos grupos: Visibles en la interfaz: <<Nombre del atributo 1>> Utilizados internamente: <<Nombre del atributo 2>>
Requisitos especiales	Son los requisitos no funcionales específicos para el requisito. Por ejemplo, estándares de intercambio de información.	
Asuntos pendientes	Posibles mejoras al requisito.	

Tabla 28 Descripción del requisito funcional Registrar Traza de Datos.

GLOSARIO DE TÉRMINOS

CASE (Computer-Aided Software Engineering): Conjunto de herramientas y métodos asociados que proporcionan asistencia automatizada en el proceso de desarrollo del software a lo largo de su ciclo de vida.

CEIGE: Centro para la Informatización de Gestión de Entidades.

Cliente Ligero: Se refiere a las computadoras personales utilizadas por los usuarios, para interactuar con aplicaciones informáticas alojadas en un servidor. Los componentes indispensables que debe tener el cliente son CPU, Monitor, UPS, Teclado y Mouse. Tiene como característica especial que el CPU no cuenta con disco duro y que el sistema operativo del cliente es cargado por la red de datos, por tanto para utilizar el cliente es necesario estar conectado a la red.

Marco de Trabajo: Es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Métrica: Medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

Módulo: Es un componente autocontrolado de un sistema, el cual posee una interfaz bien definida hacia otros componentes. De las varias tareas que debe realizar un programa para cumplir con su función u objetivos, un módulo realizará una de dichas tareas (o quizás varias en algún caso).

MVC: Es un patrón de Arquitectura de Software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

ODBC: Open Database Connectivity (ODBC). Estándar de acceso a Bases de datos desarrollado por Microsoft Corporation, el objetivo de ODBC es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué Sistema Gestor de Bases de

Datos almacene los datos.

ORM: Componente de software que permite trabajar con los datos persistidos como si ellos fueran parte de una base de datos orientada a objetos.

Proceso: Conjunto de actividades relacionadas o que interactúan las cuales transforman elementos de entradas en resultados.

Proceso de negocio: Conjunto de tareas relacionadas lógicamente para lograr un resultado de negocio definido, definen entradas, funciones y salidas.

Subsistema: Cada uno de los componentes principales de un sistema que este dividido en componentes. Cada subsistema abarca aspectos del sistema que comparten alguna propiedad común.

SQL: Lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas.

UCI: Universidad de las Ciencias Informáticas.

UML (Unified Modeling Language): Lenguaje unificado de modelado de sistemas de software.

Validación: Confirmación mediante el suministro de evidencia objetiva de que se han cumplido los requisitos para una utilización o aplicación especifica prevista.