Universidad de las Ciencias Informáticas

Facultad 3

“Configuración de las trazas a nivel de acción e integración en el marco de trabajo Sauxe”

Trabajo de diploma para optar por el título de ingeniero en ciencias informáticas

Autor(es): Jorge Rafael Pineda Portal

Tutor(es): Msc. Oiner Gómez Baryolo

Ing. Yoel Hernández Mendoza

Ing. René Rodrigo Bauta Camejo

Ciudad de La Habana, junio del 2011

"Año 53 de la Revolución"

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo al departamento de tecnología de CEIGE de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

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

_____
Jorge Rafael Pineda Portal

Msc. Oiner Gómez Baryolo	Ing. Yoel Hernández Mendoza	Ing. René Rodrigo Bauta Camejo

Datos de Contacto

Msc. Oiner Gómez Baryolo.....Correo: oiner@uci.cu
Ing. René R. Bauta Camejo.....Correo: rrbauta@uci.cu
Ing. Yoel Hernández Mendoza.....Correo: yhmendoza@uci.cu

Agradecimientos

Quiero agradecer por el cumplimiento de este sueño:

- A mis padres por ser las personas más importante para mí y brindarme siempre su cariño.
- A toda mi familia que estuvo ahí siempre que la necesité y me brindó todo su apoyo.
- A mis amigos que me apoyaron en todo momento.
- A mi novia que supo hacerme estudiar y preocuparme más por las cosas que realmente importan.

Dedicatoria

- ✓ A mis padres que siempre me han apoyado incondicionalmente.
- ✓ A mi novia que siempre me alentó en los momentos difíciles y me ayudó mucho.
- ✓ A mis amigos por ser especiales para mí y siempre estar ahí.

Resumen

El Marco de trabajo Sauxe, creado por el Centro de Información de la Gestión de Entidades (CEIGE), posee un subsistema de traza que permite el registro de los eventos que se realizan en la aplicación. A partir de estos datos se puede monitorear el funcionamiento de la aplicación y corregir posibles problemas de seguridad. El sistema registra todas las acciones realizadas en la aplicación por lo que se guarda una gran cantidad de información que atenta contra el rendimiento de la misma. Este trabajo tiene como objetivo incorporar al marco de trabajo Sauxe la funcionalidad de configurar la activación de trazas de acción e integración con el fin de registrar solo las necesarias.

En este trabajo se realiza un estudio de los métodos y herramientas más actuales utilizadas en el campo del Registro de Trazas. En el mismo se lleva a cabo el análisis, diseño e implementación de un subsistema que facilite el trabajo de los usuarios que se encargan del mantenimiento de la aplicación. Se describe cada una de las tareas realizadas y se hace un análisis de los resultados obtenidos.

PALABRAS CLAVE: eventos, registro de eventos, trazas.

Índice

INTRODUCCIÓN: 10

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA: 13

1.1 Introducción 13
1.2 Registro y monitoreo de trazas en sistemas informáticos 13
1.3 Herramientas para el registro y monitoreo de trazas 13
1.3.1 Herramientas para el registro de trazas a nivel internacional: 13
1.3.2 Herramientas para el registro de trazas en la UCI: 14
1.4 Patrones 16
1.4.1 ¿Qué es un patrón? 16
1.4.2 Patrón de diseño 17
1.4.3 Patrones Gof (Gang of Four) 17
1.4.4 Patrones Grasp 18
1.5 Modelo de desarrollo 20
1.6 Tecnologías y herramientas para el desarrollo 20
1.6.1 Herramientas Case 20
1.6.2 Herramientas para el desarrollo colaborativo 21
1.6.3 Herramientas para el desarrollo 21
1.6.4 Librerías y Marco de trabajo 22
1.6.5 Lenguajes de modelado y desarrollo 23
1.7 Herramienta para pruebas de rendimiento 24
1.8 Normas y Estándares de Codificación 25
1.9 Conclusiones 27
CAPÍTULO 2: PROPUESTA DE SOLUCIÓN 28
2.1 Introducción 28
2.2 Modelación del negocio 28
2.2.1 Descripción del proceso: Registrar traza de acción 28
2.2.2 Descripción del proceso: Registrar traza de integración 29
2.3 Requisitos de la solución propuesta 30
2.3.1 Modelo Conceptual 30
2.3.2 Identificación de los requisitos 31
2.3.3 Especificación de requisitos 33

2.4 Diseño 36
2.4.1 Diseño de clases 36
2.4.2 Estructura de XML de configuración 38
2.4.3 Patrones empleados 39
2.5 Modelo de Datos 41
2.6 Conclusiones 42
CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA 43
3.1 Introducción 43
3.2 Modelo de Implementación 43
3.2.1 Diagrama de Componentes 43
3.2.2 Diagrama de despliegue 44
3.3 Métricas para la validación diseño 45
3.3.1 Métrica Tamaño Operacional de Clase (TOC) 45
3.3.2 Métrica Relación entre Clases (RC) 48
3.4 Pruebas de software 53
3.4.1 Pruebas de Rendimiento 53
3.4.2 Pruebas de caja blanca 54
3.4.3 Pruebas de caja negra 60
3.5 Aportes y beneficios de la investigación 61
3.6 Conclusiones 62
CONCLUSIONES 63
RECOMENDACIONES 64
BIBLIOGRAFÍA 65
BIBLIOGRAFÍA CONSULTADA 67
GLOSARIO DE TÉRMINOS 84

Índice de Figuras

Figura 1 Ejemplo de comentario de clase 26
Figura 2 Ejemplo de comentario por líneas de códigos 27
Figura 3 Descripción del proceso: Registrar traza de acción. 29
Figura 4 Descripción del proceso: Registrar traza de integración. 30
Figura 5 Modelo Conceptual 31
Figura 6 Diagrama de Clases Configurar traza de acción. 36
Figura 7 Estructura de XML de configuración. 38
Figura 8 Ejemplo de Modelo Vista Controlador 40
Figura 9 Modelo de Datos de Trazas. 41
Figura 10 Diagrama de Componentes 44
Figura 11 Diagrama de Despliegue 45
Figura 12 Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad 47
Figura 13 Resultados de la evaluación de la métrica TOC para el atributo Reutilización 48
Figura 14 Resultados de la evaluación de la métrica TOC para el atributo Complejidad 48
Figura 15 Resultados de la evaluación de la métrica RC para el atributo Acoplamiento. 51
Figura 16 Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento. 51
Figura 17 Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas. 51
Figura 18 Resultados de la evaluación de la métrica RC para el atributo Reutilización. 52
Figura 19 Resultados de la evaluación de los atributos de calidad. 53
Figura 20 Código fuente de la funcionalidad confTrazaAccion(). 55
Figura 21 Grafo de flujo correspondiente a la funcionalidad confTrazaAccion(). 56
Figura 22 Prototipo de interfaz agregar configuración de acción seleccionar acciones. 68
Figura 23 Prototipo de interfaz agregar configuración de integración seleccionar acciones 69
Figura 24 Prototipo de interfaz eliminar configuración de acción seleccionar acciones. 69
Figura 25 Prototipo de interfaz eliminar configuración de integración seleccionar acciones. 70
Figura 26 Prototipo de interfaz Configurar tiempo de limpieza de trazas. 70
Figura 27 Prototipo de interfaz Configurar limpieza de trazas automática 71
Figura 28 Prototipo de interfaz Configurar limpieza de trazas automática semanal 71
Figura 29 Prototipo de interfaz Configurar limpieza de trazas automática mensual. 71
Figura 30 Diagrama de clases Agregar configuración de acción. 77
Figura 31 Diagrama de clases Agregar configuración de integración. 78
Figura 32 Diagrama de clases Eliminar configuración de acción. 79
Figura 33 Diagrama de clases Eliminar configuración de integración. 80
Figura 34 Diagrama de clases Registrar acción. 81
Figura 35 Diagrama de clases Registrar integración. 82
Figura 36 Acta de aprobación de Calidad. 83

Índice de Tablas

Tabla 1 Requisitos Funcionales 32
Tabla 2 Especificación de requisito Adicionar configuración de trazas de acción. 34
Tabla 3 Especificación de requisito Registrar traza de acción. 35
Tabla 4 Resultados de comparación entre búsqueda secuencial y por índices. 42
Tabla 5 Métricas para el tamaño operacional de la clase. 46
Tabla 6 Criterio de evaluación de las métricas TOC 46
Tabla 7 Instrumento de evaluación de las métricas TOC. 47
Tabla 8 Métricas de relación entre clases (RC). 49
Tabla 9 Criterios de evaluación de las métricas RC. 50
Tabla 10 Instrumento de medición de las métricas RC. 50
Tabla 11 Resultados de la evaluación de la relación atributo/métrica. 52
Tabla 12 Rango de valores para la evaluación de la relación atributo/métrica. 53
Tabla 13 Pruebas de rendimiento. 54
Tabla 14 Resultados de pruebas de caja blanca. 60
Tabla 15 Casos de pruebas 61
Tabla 16 Especificación de requisito Registrar trazas de integración. 72
Tabla 17 Especificación de Requisito Agregar configuración de trazas de integración 73
Tabla 18 Especificación de Requisito eliminar configuración de trazas de acción. 74
Tabla 19 Especificación de Requisito eliminar configuración de trazas de integración. 74
Tabla 20 Especificación de Requisito Listar subsistemas. 75
Tabla 21 Descripción de Requisito Hacer Salva de Trazas. 76

INTRODUCCIÓN:

Con la evolución de las tecnologías, la sociedad está cada día más conectada electrónicamente. Los sistemas automatizados han sustituido, gracias a las mejoras tecnológicas las labores tradicionales que eran realizadas por los seres humanos. Dentro de la amplia gama de actividades que pueden automatizarse, se encuentran las relacionadas la seguridad y el seguimiento de las acciones realizadas por diferentes usuarios en los sistemas computarizados, la cual ha cobrado gran importancia. La traza es un mecanismo de registro oficial de eventos, datos o información sobre quién, qué, cuándo, dónde y por qué un evento ocurre, ya sea para parte del sistema o en toda la aplicación. Debido a su importancia este mecanismo está siendo adoptado en diversas organizaciones y sistemas informáticos para el seguimiento y registro de las acciones de los usuarios.

En la Universidad de Ciencias Informáticas (UCI) existen diferentes centros de producción, entre ellos está CEIGE que se dedica a hacer software de gestión empleados a la web. Dicho centro tiene un departamento de tecnología donde se desarrolló un marco de trabajo llamado Sauxe que entre otros subsistemas tiene el de trazas.

Las trazas son utilizadas entre otras cosas para el monitoreo de acciones para detectar hechos con relación a la seguridad de sistemas informáticos.

El sistema de trazas en el departamento de tecnología consta de una interfaz gráfica que mediante la búsqueda por tipo, fecha y categoría brinda la todas las trazas generadas en el sistema. Cuenta además con otra parte interna que registra todas la acciones que se realizan, cada acción por mínima que sea, se registra por lo que se guarda una gran cantidad de información no deseada.

La configuración de las trazas a nivel de acción e integración incorporan la funcionalidad de definir cuáles trazas serán registradas y cuáles no, lo que trae consigo que no se guarde información no deseada que pueda atentar contra el rendimiento de las aplicaciones.

Teniendo en cuenta lo planteado anteriormente se define como **Problema a Resolver**: La solución de trazas de acción e integración existente en el marco de trabajo Sauxe genera exceso de información que atenta contra el rendimiento de la aplicación.

Como **objeto de estudio** se ha definido: Trazas en los sistemas de gestión.

Para resolver el problema planteado se ha propuesto como **objetivo general**: Desarrollar una solución que permita configurar las trazas a nivel de acción e integración en el marco de trabajo Sauxe para mejorar el rendimiento de la aplicación.

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

- Realizar el marco teórico sobre la configuración y registro de trazas en software de gestión.
- Realizar el Análisis y Diseño de la solución.
- Implementar la solución.
- Realizar pruebas a la solución.

Se identificó como **campo de acción**: Configuración de trazas de acción e integración en el marco de trabajo Sauxe.
Se tiene como **idea a defender:** Si se logra configurar las trazas a nivel de acción en integración en el Marco de Trabajo Sauxe se elimina el exceso de información que atenta contra el rendimiento de la aplicación.
Se identificaron como **Tareas de Investigación:**

- Investigación de los sistemas existentes a nivel mundial relacionados con la configuración de trazas.
- Estudio de las tecnologías y arquitecturas definidas por CEIGE para el desarrollo de la solución.
- Definición de requisitos y escenarios arquitectónicos.
- Implementación de la interfaces.
- Implementación de la capa de negocio.
- Implementación de las validaciones.

Con el cumplimiento de los elementos antes planteados se tiene como **posible resultado**: Desarrollar un componente para la configuración de las trazas de acción e integración en el marco de trabajo Sauxe.
El documento está estructurado de la siguiente forma:

CAPÍTULO 1. Fundamentación Teórica: Se describen los conceptos asociados al dominio del problema a resolver, siendo estos esenciales para entender el entorno del mismo. Se presenta el estado del arte de las técnicas implicadas en el objeto de estudio y el conjunto de tecnologías involucradas en el desarrollo de la propuesta.

CAPÍTULO 2. Propuesta de Solución: Se exponen los procesos de negocios y requisitos a cumplir por la herramienta además de los artefactos generados durante el diseño de la misma.

CAPÍTULO 3. Implementación y Prueba: Se exponen los artefactos generados durante la implementación de la solución así como las métricas y pruebas utilizadas para la validación de la misma

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA:

1.1 Introducción

En el presente capítulo se realizará un estudio del estado del arte relacionado con el problema al que se le dará solución. Se revisarán conceptos, software y tecnologías usados actualmente a nivel mundial. Se hará énfasis en el modelo de desarrollo de software que se va a utilizar para dirigir el proceso creación de la solución. Se presenta un estudio de diversas herramientas a utilizar en la creación de la aplicación. Se hará un estudio de herramientas para futuras pruebas de rendimiento de la solución y se mostrará algunas normas y estándares que posteriormente serán utilizados para la implementación.

1.2 Registro y monitoreo de trazas en sistemas informáticos

A continuación se muestran los conceptos fundamentales con relación al problema para el mejor entendimiento de la investigación. Estos conceptos se verán a lo largo de todo el documento.

Trazas:

Se utilizan para comprobar la ejecución de las validaciones de datos previstas. No deben modificar el sistema, si la herramienta auditora produce incrementos apreciables de carga se convendrá de antemano la fecha y hora más adecuada para su empleo. (1)

- ✓ Importancia: las trazas permiten crear un registro de sucesos para un dispositivo o aplicación. A través de estos datos se puede monitorear el funcionamiento de la aplicación y se ofrece una oportunidad para corregir problemas de seguridad. (2)

1.3 Herramientas para el registro y monitoreo de trazas

Existen diferentes herramientas a nivel internacional, nacional y en la UCI con cierta homología a la solución propuesta en la investigación. De estas herramientas se realizó un estudio para profundizar si incluyen la configuración de las trazas y de qué forma se configuran.

1.3.1 Herramientas para el registro de trazas a nivel internacional:

A continuación se muestra una serie de herramientas con sus características principales.

Trac

Trac es un sistema de seguimiento para los proyectos de desarrollo de software. Utiliza un enfoque minimalista a la gestión de proyectos de software basado en web. Proporciona una interfaz para Subversion (u otros sistemas de control de versiones), un Wiki integrado y cómodas instalaciones de informes. Una línea de tiempo muestra todos los eventos de los proyectos actuales y pasados , por lo que la adquisición de una visión general del proyecto y seguimiento de los progresos es muy fácil. (3)
Es un sistema de código abierto desarrollado en Python, con fuerte relación con el subversion. Esta aplicación se puede apoyar en una base de datos ya sea SQLite, PostgreSQL o MySql, lo cual constituye una gran ventaja, ya que no depende solamente de un gestor de bases de datos para su funcionamiento.

Open Populi Framework:

Open Populi Framework es un marco de desarrollo de software de gestión desarrollado en lenguaje PHP5 en código abierto (Open Source) que funciona en entorno web y ha sido diseñado para cubrir las necesidades existentes en el campo de la Administración Electrónica. (4)
Open Populi ha sido creado para ser utilizado a través de la web, brindando todas sus funcionen a través de un navegador convencional, no requiere instalar ningún software en los equipos clientes. Esta herramienta es distribuida bajo y una licencia de software libre, se puede acceder a su código fuente libremente para revisarlo y modificarlo con total independencia. OP Framework está pensado para el desarrollo de aplicaciones web en distintos idiomas, autenticación de usuarios, encriptación y avanzados requerimientos con relación a la auditoría de sistemas. Una de sus ventajas es que incorpora módulos, usuarios, grupos de usuarios, acciones y auditorías. Otras de las ventajas brindadas por OP Framework es la reducción de tiempo de desarrollos en aplicaciones complejas, disminuye el tiempo necesario para realizar las pruebas, ofrece mayor facilidad para el mantenimiento de las aplicaciones y reduce tiempo para mejorar y ampliar las aplicaciones.

1.3.2 Herramientas para el registro de trazas en la UCI:

Sistema de Reportes de Navegación por Internet (SRNI):

Proporciona reportes de navegación en internet de los usuarios. Los reportes se crean a partir de las trazas del servidor proxy.
Para consultar esa información los usuarios puedes acceder de esta forma:

-Acceso por día
-Acceso por Hora
-Acceso por URL
-Acceso por dirección IP
El sistema brinda la posibilidad de conocer el nuevo sistema de consumo que se calcula con respecto a los sitios a los que se acceden y la hora en que se accede.

Marco de trabajo Sauxe:
En el marco de trabajo Sauxe la ocurrencia de eventos en las aplicaciones informáticas está ligada con su existencia en sí. Una vez se produzca algún evento en el sistema este componente creará una traza, y mediante publishers se persistirán sus datos (identificador de la traza, fecha y hora de generación, etc.) en la base de datos. Concurrentemente se utilizarán los triggers para notificar a los componentes interesados de la creación de la misma. (2)

Eventos:

- ✓ Inicio de una acción: Se dispara un evento al comienzo de una acción.
- ✓ Fin de una acción: Se dispara un evento al finalizar una acción.

- ✓ Error en una acción: Se dispara un evento cuando en una acción ocurre un error.
- ✓ Autenticar usuario: Se dispara un evento por cada URL visitada.
- ✓ IoC Externo: Se dispara un evento cuando ocurre integración entre diferentes subsistemas.
- ✓ IoC Interno: Se dispara un evento cuando ocurre integración entre diferentes componentes de un subsistema.
- ✓ Excepciones: Se dispara un evento cuando ocurre una excepción en el sistema.
- ✓ Rendimiento: Es el tiempo de ejecución de una acción.
- ✓ Error IoC Externo: Se dispara un evento cuando ocurre un error en la integración entre diferentes subsistemas.
- ✓ Error IoC Interno: Se dispara un evento cuando ocurre un error en la integración entre diferentes componentes de un subsistema.

Debido al exceso de información que actualmente se registra en el marco de trabajo Sauxe se atenta contra el rendimiento de la aplicación por lo que se decidió incorporarle un componente de configuración en el que se decida cuales trazas serán registradas y cuáles no. Las anteriores herramientas poseen ventajas para el registro de trazas pero no cumplen con las necesidades de la aplicación ya que no incluyen la configuración de las trazas.

1.4 Patrones

1.4.1 ¿Qué es un patrón?

Cada patrón es una regla compuesta por tres partes que expresa la relación entre cierto contexto, un problema y una solución. El patrón es, resumiendo, al mismo tiempo una cosa que tiene su lugar en el mundo y la regla que nos dice cómo crear esa cosa y cuándo debemos crearla. (5)

1.4.2 Patrón de diseño

“Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software.”

En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Debemos tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios). (6)

1.4.3 Patrones Gof (Gang of Four)

Clasificación:

- ✓ **Creacionales**: Inicialización y configuración de objetos.
    - Método de Fabricación (Factory Method): Parte del principio de que las subclases determinan la clase a implementar.
    - Singlenton: Restringe la instanciación de una clase o valor de un tipo a un solo objeto.
- ✓ **Estructurales**: Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
    - Adaptador (*Adapter*): Convierte una interfaz en otra.
    - Puente (*Bridge*): Desacopla una abstracción de su implementación permitiendo modificarlas independientemente.
    - Objeto Compuesto (*Composite*): Utilizado para construir objetos complejos a partir de otros más simples, utilizando para ello la composición recursiva y una estructura de árbol.
    - Envoltorio (*Decorator*): Permite añadir dinámicamente funcionalidad a una clase existente, evitando heredar sucesivas clases para incorporar la nueva funcionalidad.
    - Fachada (*Facade*): Permite simplificar la interfaz para un subsistema.
    - Peso Ligero (*Flyweight*): Elimina la redundancia o la reduce cuando tenemos gran cantidad de objetos con información idéntica.

- Apoderado (*Proxy*): Un objeto se aproxima a otro.

✓ **Comportamiento**: Más que describir objetos o clases, describen la comunicación entre ellos.

- Cadena de responsabilidad (*Chain of responsibility*): La base es permitir que más de un objeto tenga la posibilidad de atender una petición.
- Orden (*Command*): Encapsula una petición como un objeto dando la posibilidad de "deshacer" la petición.
- Intérprete (*Interpreter*): Intérprete de lenguaje para una gramática simple y sencilla.
- Iterador: Define una interfaz que declara los métodos necesarios para acceder secuencialmente a una colección de objetos sin exponer su estructura interna.
- Mediador (*Mediator*): Coordina las relaciones entre sus asociados. Permite la interacción de varios objetos, sin generar acoples fuertes en esas relaciones.
- Recuerdo (*Memento*): Almacena el estado de un objeto y lo restaura posteriormente.
- Observador (*Observer*): Notificaciones de cambios de estado de un objeto.
- Estrategia (*Strategy*): Utilizado para manejar la selección de un algoritmo. (6)

1.4.4 Patrones Grasp

Los patrones grasp describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (7)

Experto:

Se usa para asignar responsabilidades, la clase cuenta con la información necesaria para cumplir la responsabilidad. Se conserva el encapsulamiento ya que los objetos se apoyan en su propia información para realizar su tarea. (7)

Bajo Acoplamiento:

El acoplamiento es la medida de la fuerza con que una clase está acoplada a otra. Debe haber pocas dependencias entre las clases. Si todas las clases tienen una alta dependencia pueden presentar los siguientes problemas:

✓ Los cambios de las clases a fines ocasionan cambios locales.

- ✓ Son más difíciles de entender cuando están aisladas.
- ✓ Son más difíciles de reutilizar porque requieren de la presencia de otras clases de las que dependen. (7)

Alta Cohesión:

Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realizan un trabajo enorme. No conviene una clase con baja cohesión pues presenta los siguientes problemas:

- ✓ Son difíciles de comprender.
- ✓ Son difíciles de reutilizar.
- ✓ Son difíciles de conservar.
- ✓ Son delicadas y las afectan constantemente los cambios. (7)

Creador:

Se asigna a la clase B la responsabilidad de crear una instancia de la clase A solamente cuando

- ✓ B contiene a A
- ✓ B es una agregación (o composición) de A
- ✓ B almacena a A
- ✓ B tiene los datos de inicialización de A (datos que requiere su constructor)
- ✓ B usa a A.

La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia es conveniente contar con un principio general para asignar las responsabilidades concernientes a ella. El diseño, bien asignado puede soportar un bajo acoplamiento, mayor claridad, encapsulamiento y reutilización. (7)

1.5 Modelo de desarrollo

Para el desarrollo de un software de la magnitud y con las características peculiares de Sauxe se decidió seguir el modelo de desarrollo definido por CEIGE que presenta las siguientes características. (8)

- ✓ Centrado en la arquitectura:
- ✓ Iterativo incremental.
- ✓ Orientado a componentes.
- ✓ Ágil y adaptable al cambio.

1.6 Tecnologías y herramientas para el desarrollo

1.6.1 Herramientas Case

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Como es sabido, los estados en el Ciclo de Vida de desarrollo de un Software son: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación. (9)

Visual Paradigmn 6.4:

Visual Paradigm For UML es una Herramienta Case que soporta las últimas versiones del mismo, (Lenguaje de Modelado Unificado) y la Notación y Modelado de Procesos de Negocios. En adición al soporte de Modelado UML esta herramienta provee el modelado de procesos de negocios, además de un generador de mapeo de objetos-relacionales para los lenguajes de programación Java .NET y PHP. (10)

Se integra con las siguientes herramientas Java:

- ✓ Eclipse/IBM WebSphere
- ✓ JBuilder
- ✓ NetBeans IDE
- ✓ Oracle JDeveloper
- ✓ BEA Weblogic

- ✓ Está disponible en varias ediciones, cada una destinada a unas necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal.

1.6.2 Herramientas para el desarrollo colaborativo

Subversion 1.6.6:

Subversion es un sistema de control de versiones libre y de código fuente abierto. Es decir, Subversion maneja ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un *repositorio* central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos. En este aspecto, mucha gente piensa en los sistemas de versiones como en una especie de "máquina del tiempo". (11)

1.6.3 Herramientas para el desarrollo

Servidor web 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, de código abierto, altamente configurable y 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 sean necesarios. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor y es posible configurarlo para que ejecute un determinado script cuando esto suceda (12)

PostgreSQL 8.3:

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (13)

Índice

Una de las funcionalidades que brinda el Postgres es la creación de índices para mejorar el rendimiento en la base de datos.

Los índices son utilizados para mejorar el rendimiento de bases de datos. Dichos objetivos serán definidos en las columnas de la tabla (o atributos de clase) que se utilizan como las calificaciones en las consultas repetitivas. Los índices también se pueden utilizar para exigir la unicidad de la clave principal de una tabla. Hace el procesamiento más rápido para cada solicitud de consulta, ya que se elimina la necesidad de buscar en toda la tabla. Debido a la mejora en el rendimiento acelerando las consultas es muy útil cuando la tabla contiene miles de registros. (14)

NetBeans 6.9:

Es un entorno de desarrollo de Java disponible para Windows, Mac, Linux y Solaris. En esta versión se introduce el JavaFX Composer, una herramienta visual para construir aplicaciones JavaFX. Además de soportar drag drop de componentes, el composer permite realizar binding de los modelos de los componentes y las fuentes de datos que proporcionan dichos modelos.

Otra de las principales características es soporte para OSGI dentro de la plataforma de Netbeans. A nivel de actualizaciones de frameworks, ahora Netbeans incluye JavaFX SDK 1.3, el framework de PHP Zend, y Ruby on Rails 3.0. Por último, también se han realizado ciertas mejoras menores al editor de código Java y al depurador. (16)

1.6.4 Librerías y Marco de trabajo

Zend Framework 1.8:

Zend Framework es un framework de código abierto para desarrollar aplicaciones web y servicios web con PHP5. Zend Framework es una implementación que usa código 100% orientado a objetos. La estructura de los componentes de Zend Framework es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. A menudo se refiere a este tipo de diseño como "use-at-will" (uso a voluntad). Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de

Zend Framework conforman un potente y extensible framework de aplicaciones web al combinarse. Zend Framework ofrece un gran rendimiento y una robusta implementación MVC, una abstracción de base de datos fácil de usar, y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos. (17)

Doctrine 0.11:

Doctrine es un potente y completo sistema de mapeado de objetos relacionales (ORM) para PHP 5.2 o superior, que posee una formidable capa de abstracción a base de datos. Entre sus principales características se encuentra la posibilidad de escribir de manera opcional las consultas a la base de datos. Lo que proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria. Permite también exportar una base de datos existente a sus clases correspondientes y también convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. (18)

ExtJS:

Es una librería de JavaScript de alto rendimiento, ligera que es compatible con la mayoría de los navegadores. Permite crear páginas e interfaces web dinámicas. Posee una gran cantidad de componentes que se pueden utilizar en la capa de presentación para agilizar el proceso de desarrollo. Contiene modelos de componentes extensibles. Licencias de códigos abiertos y comerciales.

Ventajas:

- ✓ Es independiente o adaptable a diferentes Framework.
- ✓ Tiene una amplia comunidad de usuarios.
- ✓ Su código es reutilizable.

1.6.5 Lenguajes de modelado y desarrollo

Lenguaje de Modelado Unificado (UML):

UML es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema intensivo de software. Ofrece una forma estándar para escribir un plano del sistema, incluyendo

cosas conceptuales tales como procesos de negocios y funciones del sistema, también cosas concretas como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. (19)

Lenguaje de desarrollo:

Personal Home Page (PHP 5.2.6):

Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, es rápido, tiene mucha documentación y posee una gran librería de funciones. PHP se incrusta dentro del lenguaje HTML por lo que lo hace tan fácil de utilizar. Con el uso de PHP se pueden leer y escribir archivo, crear imágenes, comunicarse con servidores remotos y realizar consultas a base de datos. Tiene soporte para gran cantidad de Base de Datos como InterBase, Oracle, mSQL, MySQL, PostgresSQL entre otras. Debido a que tiene un código abierto, posee la ayuda de un gran grupo de programadores que reparan rápidamente los fallos que se encuentren.

JavaScript

JavaScript es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web. Puede ser usado igualmente por profesionales que por los que están iniciándose en el desarrollo de las aplicaciones web. Es un lenguaje del lado del cliente por lo que no necesita compilación. Tiene varias aplicaciones como: responder a eventos locales dentro de una página, validación de formularios dentro de una misma página, la realización de cálculos en tiempo real.

1.7 Herramienta para pruebas de rendimiento

JMeter:

JMeter es un software de código abierto, hecho en Java. Es una aplicación de escritorio diseñado para cargar el comportamiento de pruebas funcionales y medir el desempeño. Puede ser utilizado para probar el rendimiento en recursos estáticos y dinámicos (archivos, Servlets, scripts de Perl, Java Objects, bases de datos y consultas, servidores FTP y mucho más). Puede ser utilizado para simular una carga pesada en un servidor, red o un objeto para poner a prueba su resistencia o para analizar el rendimiento general en diferentes tipos de carga. (20)

1.8 Normas y Estándares de Codificación

- **Nomenclatura**

Clases controladoras:

Se definió que las clases controladoras después del nombre llevarán la palabra “Controller”

Ejemplo: ConfigurarTrazaAccionController

Clases en las vistas:

Se definió para las clases que se encuentran dentro de Views que el nombre se escribirá sin ningún sufijo o prefijo. Los nombres se escribirán en notación camello (Camel Case).

Ejemplo: ConfTrazaAccion.js

Clases de los modelos:

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

Ejemplo: BuscarModel

Las clases que se encuentran dentro de Domain reciben el nombre de la tabla de la base de datos.

Nomenclatura de las funciones:

El nombre de las funciones se escribe con la primera palabra en minúscula, en caso de ser un nombre compuesto se empleará la notación CamelCasting*.

Ejemplo: insertarAcciones.

En caso de ser un método de la clase controladora sería insertarAccionesAction

Nomenclatura de las variables:

El nombre que se emplea para las variables se escribe con la primera palabra en minúscula, en caso de ser un nombre compuesto se empleará la notación CamelCasting* y comenzando con un prefijo según el tipo de dato.

Ejemplo: arrValores.

Nomenclatura de las contantes:

El nombre de las contantes deberá ser con todas sus letras en mayúscula.

Ejemplo: VALOR.

- **Normas de Comentarios**

Con el objetivo de lograr que el código sea más legible es necesario comentar todo lo que se haga dentro del mismo.

Nomenclatura de los comentarios:

En las clases:

Ante de la declaración de una clase se escribe una descripción breve de la misma, que se escribe de la siguiente forma:

/**
* Nombre de la clase *
* Descripción *
* @author *
* @package *(módulo)
* @subpackage *(sub módulo)
* @copyright *
* @version (versión - parche)
*/

```php
<?php
/**
*Nombre de la clase Configurartrazaaccion*
*Descripción: esta es la clase que se encarga de configurar las trazas de accion*
*@author Jorge Rafael Pineda Portal*
*@package Traza*
*@subpackage configurar trazas de accion*
*@copyright CEIGE*
*verson 1.0*
*/
```

Figura 1 Ejemplo de comentario de clase

Comentario por líneas:

Para ayudar a aclarar la complejidad de los algoritmos se utilizan los comentarios para las líneas. Para ellos se utiliza //, y seguidamente el comentario que describe la línea de código, ejemplo:

```
$contr = $traz->addChild($nombcontroler); //se crea controlador un hijo del XML
$contr->addChild("nombre",$nombcontroler); //se crea un hijo con el nombre del controlador
$accion = $contr->addChild("acciones");  //se crea acciones como un hijo de controlador
```

Figura 2 Ejemplo de comentario por líneas de códigos

(21)

1.9 Conclusiones

En este capítulo se proporcionaron una serie de conceptos necesarios para el mejor entendimiento del negocio. Se realizó un análisis de algunas herramientas actuales sobre el registro de trazas, se tuvo en cuenta sus funcionalidades y la razón por lo que no se decidió utilizarlas. Se muestran las herramientas y lenguajes definidos por el departamento para lograr el desarrollo de la solución, así como las normas y estándares para la codificación en la futura implementación de la solución.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

2.1 Introducción

Durante el siguiente capítulo se recogen los resultados obtenidos durante el proceso de desarrollo de la solución así como algunos artefactos generados por el mismo. Primeramente se procede a realizar el modelo conceptual y la descripción de los requisitos funcionales de la solución para lograr un mayor entendimiento de la misma. Luego se realiza una descripción del diseño para lograr los objetivos trazados, además del camino a seguir para el proceso de desarrollo.

2.2 Modelación del negocio

Un proceso de negocio es un conjunto de tareas relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio definido. La descripción de los procesos de negocio hace más viable el paso a las actividades del análisis ya que posibilita una compresión más clara de los procesos en cuestión y contribuye a que los requisitos definidos satisfagan las necesidades del usuario. (22)
Se definieron los procesos Registrar traza de acción y Registrar traza de integración.

2.2.1 Descripción del proceso: Registrar traza de acción

El proceso definido en la figura 3 muestra cómo se desarrolla el registro de trazas de acción.

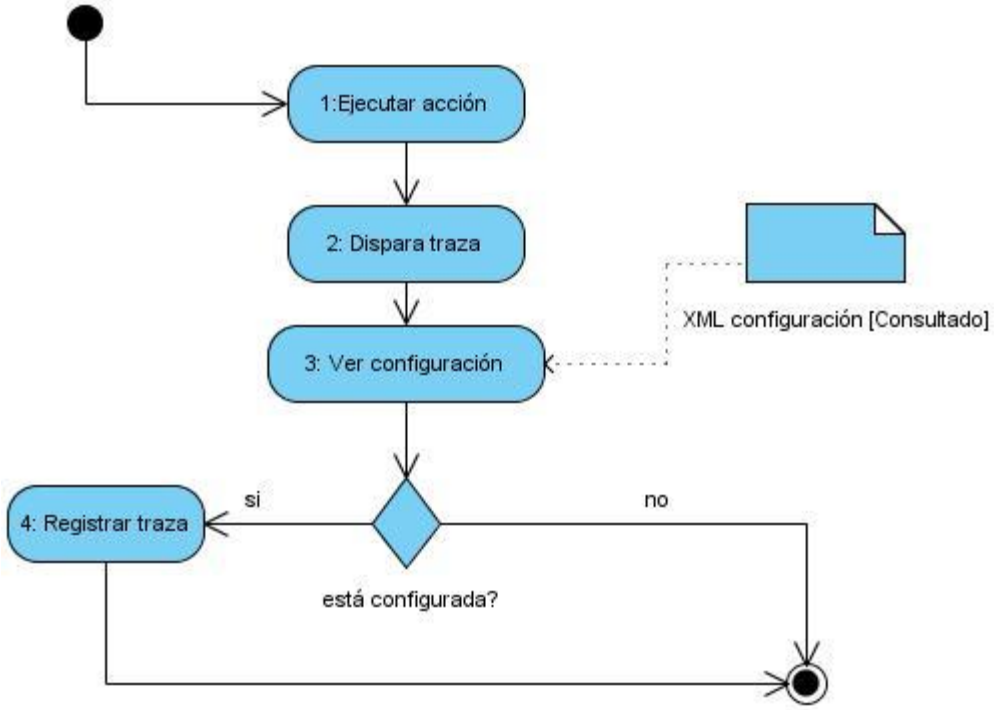

Figura 3 Descripción del proceso: Registrar traza de acción.

El proceso se inicia con la ejecución de una acción en la aplicación por parte del usuario, provocando que se dispare una traza. Luego se verifica el XML de configuración, si dicha traza está definida entonces se registra y concluye el proceso. En caso de no estar definida en el XML concluye el proceso y no se registra nada.

2.2.2 Descripción del proceso: Registrar traza de integración

El proceso definido en la figura 4 muestra cómo se desarrolla el registro de trazas de integración.

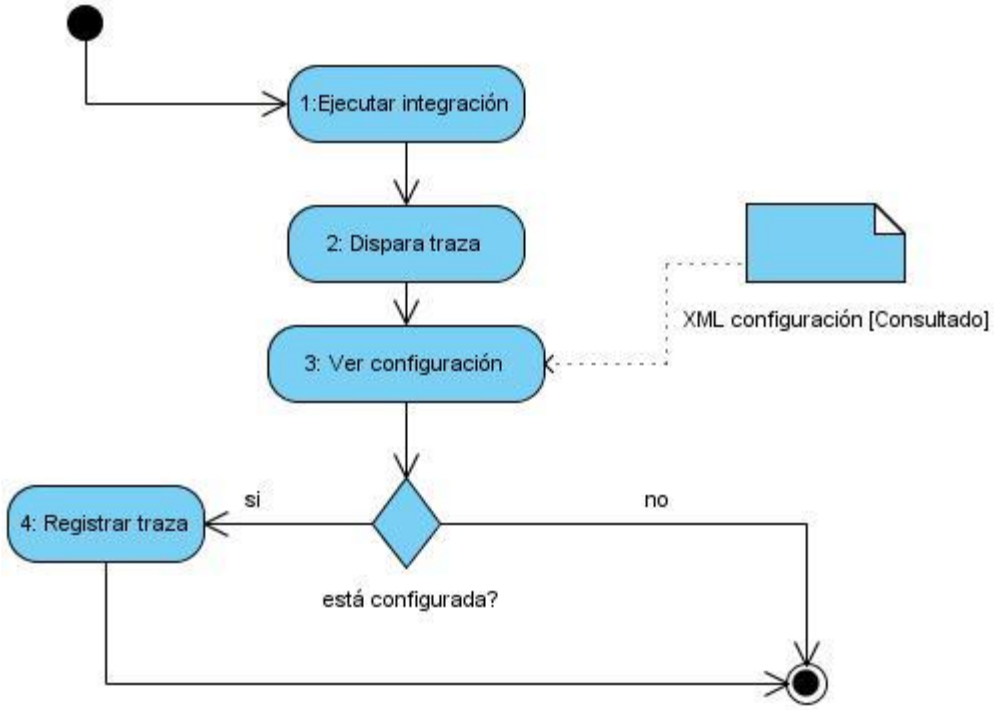

Figura 4 Descripción del proceso: Registrar traza de integración.

El proceso se inicia con la ejecución de una acción que contenga una integración con otra en la aplicación por parte del usuario, provocando que se dispare una traza. Luego se verifica el XML de configuración, si dicha traza está definida entonces se registra y concluye el proceso. En caso de no estar definida en el XML concluye el proceso y no se registra nada.

2.3 Requisitos de la solución propuesta

El requisito es la capacidad o condición que necesita un usuario para lograr un objetivo o resolver un problema. Los requerimientos deben ser especificados por escrito, ser claros y precisos. Estos son clasificados en funcionales y no funcionales. Los funcionales son condiciones o capacidades que debe tener el sistema, mientras que los no funcionales son cualidades o propiedades que debe tener el producto.

2.3.1 Modelo Conceptual

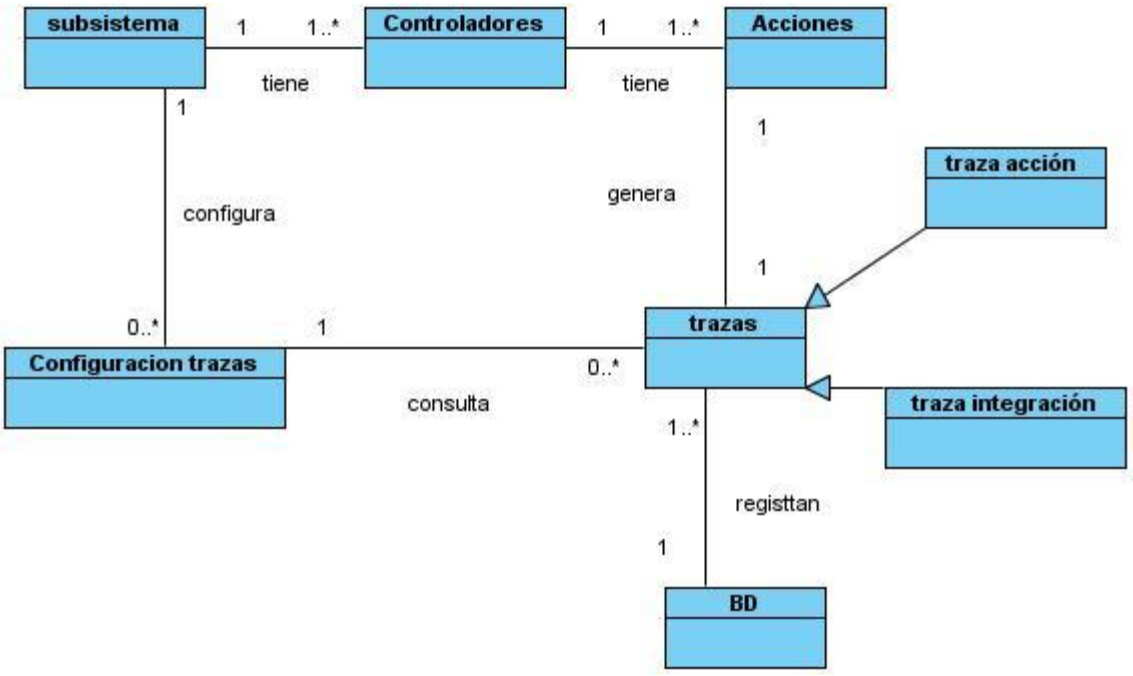

Figura 5 Modelo Conceptual

2.3.2 Identificación de los requisitos

Funcionales:

RF1	**Gestionar configuración de las Trazas de Acción**
	• Adicionar configuración de trazas de acción. • Eliminar Configuración de trazas de acción.
RF2	**Gestionar configuración de las Trazas de Integración**
	• Adicionar configuración de trazas de integración. • Eliminar Configuración de trazas de integración.
RF3	• Registrar trazas de Acción.
RF4	• Registrar trazas de Integración.

RF5	• Listar Subsistemas
RF6	• Configurar Salva de Trazas

Tabla 1 Requisitos Funcionales

No funcionales:

✓ **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 incluyendo el mantenimiento de las bases de datos, así como la salva de la información, se realizará de forma centralizada por el administrador.

✓ **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.

✓ **Hardware**

Servidor:

- Procesador: 3.00 GHZ
- RAM: 1GB
- Disco duro: 160 GB
- UPS
- Lector de CD
- Tarjeta de Red

PC Cliente:

- Procesador: 1.40 GHZ

- RAM: 256(recomendado 512MB)
- Tarjeta de Red

✓ **Software**

Servidor:

- Sistema Operativo Linux
- Servidor Web: Apache 2.0
- Librerías Adicionales: PHP 5
- Sistema Gestor de Base de Datos: PostgreSQL 8.3(si es de BD)

PC Cliente:

- Sistema Operativo: Linux o Windows
- Navegador Web: Mozilla Firefox v2.2 ó superior
- Herramientas Ofimáticas
- Visualizador de ficheros pdf

Los anteriores requisitos de software y hardware están definidos en el documento Arquitectura Vista de Infraestructura de CEIGE.

2.3.3 Especificación de requisitos

En el siguiente epígrafe se explican los requisitos funcionales mencionados anteriormente, con el propósito de lograr un correcto desarrollo del sistema. Los requisitos faltantes se muestran en el anexo 2.

➢ **Requisito funcional Gestionar configuración de las Trazas de Acción.**

- Especificación del requisito **Adicionar configuración de trazas de acción**.

Precondiciones	Se ha registrado alguna traza de acción.
Flujo de eventos	
Flujo básico	
1	Se selecciona el subsistema a configurar.
2	El sistema muestra los controladores que posee el subsistema.

3	Se selecciona el controlador a configurar.	
4	El sistema muestra las acciones que posee el controlador.	
5	Se seleccionan las acciones a configurar.	
6	El sistema guarda en un XML los datos de la configuración.	
7	Concluye el requisito.	
Pos-condiciones		
1	Se registró en el sistema una nueva configuración de traza de acción	
Flujos alternativos		
Flujo alternativo		
Pos-condiciones		
1	N/A	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo Conceptual Configuración de trazas de acción e integración.	
Relaciones	Requisitos Incluidos	Cargar Subsistemas
	Extensiones	N/A
Conceptos	N/A	Visibles en la Interfaz - Nombre Subsistema - Nombre Controlador - Nombre Acción

Tabla 2 Especificación de requisito Adicionar configuración de trazas de acción.

- **Requisito funcional Registrar Trazas de Acción.**

  - Especificación de requisito **Registrar traza de Acción**

Precondiciones	Se ha realizado una acción dentro de la aplicación.
Flujo de eventos	
Flujo básico <<Nombre del flujo básico>>	
1	Se crea una traza con los campos idtraza, fecha, hora, idcategoria, idtipotraza, usuario, idestructuracomun.
2	El sistema obtiene los datos de la acción realizada y los compara con los datos de un XML
3	Si la acción está definida en el XML el sistema guarda las trazas en la base de datos.
4	Concluye el requisito.

Pos-condiciones	
2	Se registro una traza de acción.
Validaciones	
2	Se validan los datos según lo establecido en el Modelo conceptual Configuración de trazas de acción e integración.
Conceptos	**N/S**
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 3 Especificación de requisito Registrar traza de acción.

2.4 Diseño

2.4.1 Diseño de clases

Diagrama de clases:

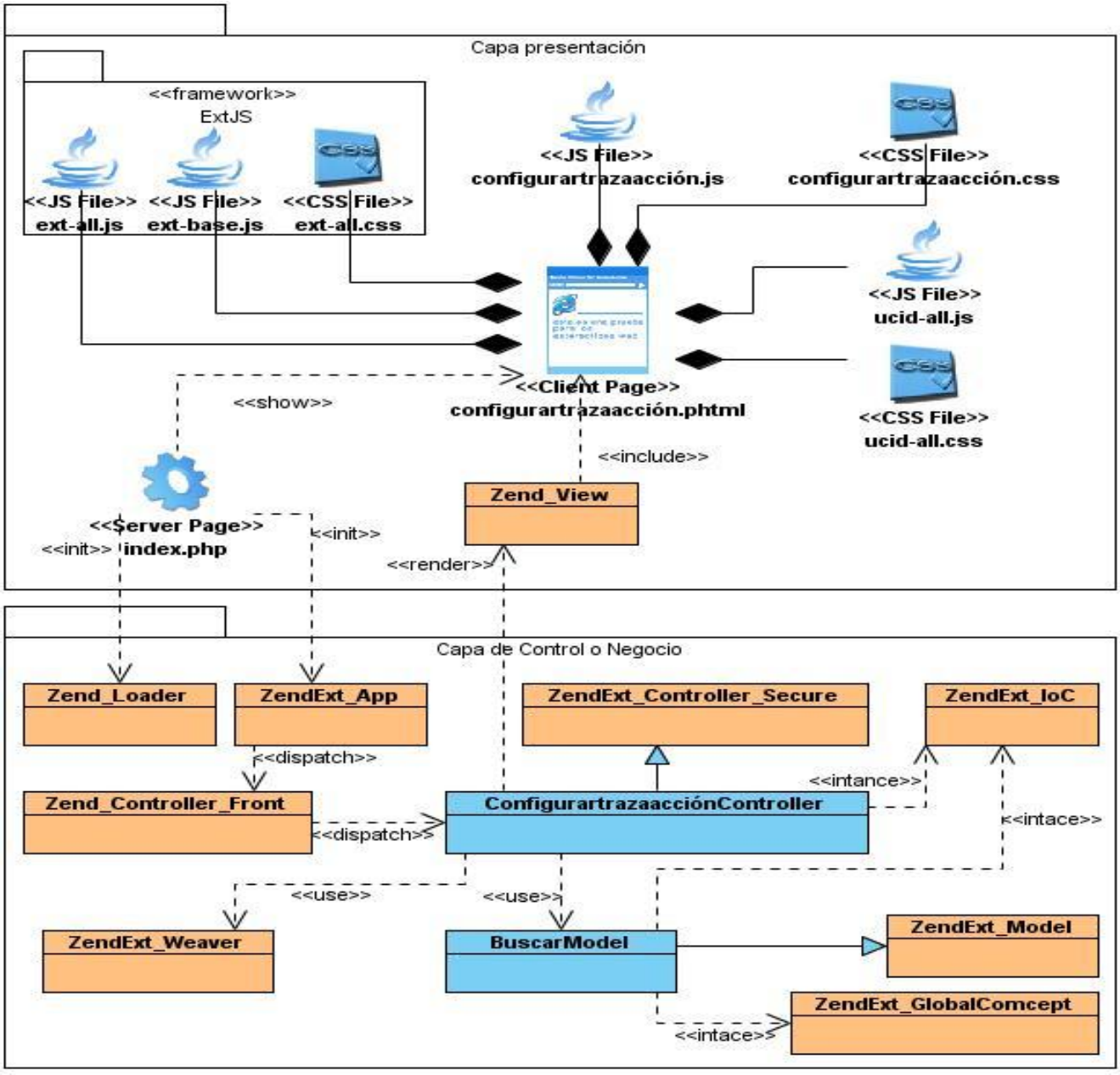

Figura 6 Diagrama de Clases Configurar traza de acción.

Descripción de las clases:

Descripción de la clase **ConfigurarTrazaAccionController**.

Nombre: ConfigurarTrazaAccionController	
Tipo de clase: Controladora	
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
cargarsubsistemasAction()	Se encarga de mostrar todos los subsistemas y mostrarlos en un árbol.
cargarcontroladoresAction()	Se encarga de mostrar todos los controladores de un subsistema seleccionado y mostrarlos en una tabla.
cargaraccionesAction()	Se encarga de mostrar todas las acciones de un controlador seleccionado y mostrarlas en una tabla.
insertaraccionesAction ()	Se encarga de guardar en un XML las configuraciones realizadas.

Descripción de la clase **BuscarModel**.

Nombre: BuscarModel	
Tipo de clase: Controladora	
Para cada responsabilidad	
Nombre.	**Descripción.**
getClassesByFile ()	Se encarga de retornar una clase dada una dirección por parámetro.
getMethodsByClass ()	Se encarga de retornar los métodos de una

	clase dada una dirección y una clase por parámetro.
directory_map()	Devuelve todos los ficheros que están en la dirección pasada por parámetro.

2.4.2 Estructura de XML de configuración

```
- <subsistema>
  - <controlador1>
      <nombre>nombrecontrolador1</nombre>
    - <acciones>
        <nombre>nombreaccion1</nombre>
        <nombre>nombreaccion2</nombre>
        <nombre>nombreaccion3</nombre>
      </acciones>
    </controlador1>
  - <controlador2>
      <nombre>nombrecontrolador2</nombre>
    - <acciones>
        <nombre>nombreaccion1</nombre>
        <nombre>nombreaccion2</nombre>
      </acciones>
    </controlador2>
  - <controlador3>
      <nombre>nombrecontrolador3</nombre>
    - <acciones>
        <nombre>nombreaccion1</nombre>
      </acciones>
    </controlador3>
</subsistema>
```

Figura 7 Estructura de XML de configuración.

En la figura 7 se muestra la estructura que tiene el XML de configuración donde en cada subsistema se crea un XML con el nombre del mismo. Dentro se muestran los controladores, con su nombre y sus acciones definidas para ser registradas. Por esta estructura se rige la parte de registro para verificar si la

traza se va a registrar o no. En el marco de trabajo Sauxe se está tratando de lograr una mayor independencia entre los módulos como parte de una política de mejora. El componente propuesto contribuye a esta política ya que la configuración crea un XML independiente para cada subsistema, otra de las ventajas que brinda es que mejora el tiempo de respuesta cuando se trata de acceder al XML para el registro de las trazas ya que su tamaño máximo es menor que si fuera un solo XML central porque los subsistemas son solo un pequeño subconjunto del marco de trabajo.

2.4.3 Patrones empleados

Modelo Vista Controlador (MVC):

Modelo Vista Controlador es un patrón de arquitectura utilizado en sistemas Web para separar los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos, permitiendo flexibilidad y facilidad a la hora de hacer futuros cambios.

- ✓ Modelo: representa la estructura de los datos. Generalmente el modelo de clases contendrá funciones para consultar, insertar y actualizar información de la base de datos.
- ✓ Vista: es la información que se presenta al usuario, puede ser una página web o parte de ella.
- ✓ Controlador: actúa como intermediario entre el modelo y la vista, realiza las solicitudes de los usuarios actuando sobre los datos del modelo.

A continuación se muestra un ejemplo de su aplicación en la solución.

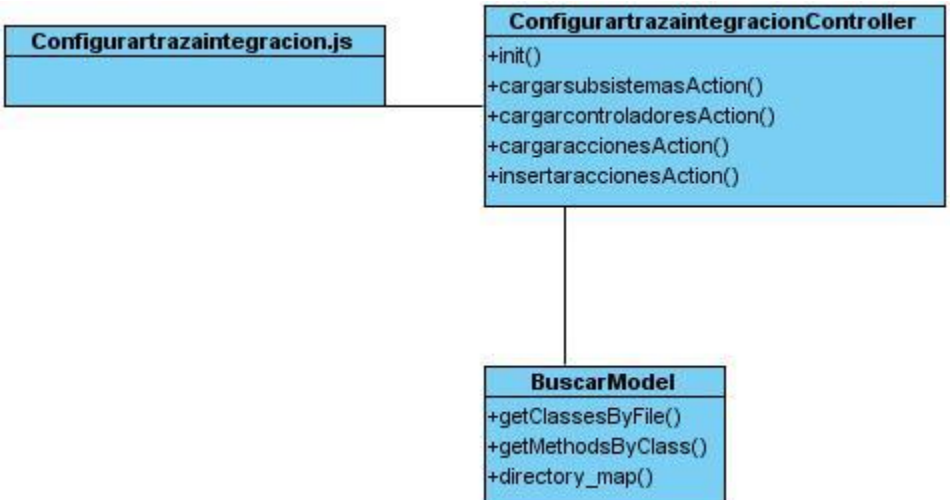

Figura 8 Ejemplo de Modelo Vista Controlador

Bajo Acoplamiento

Consiste en tener las clases lo menos ligadas entre sí que se pueda, de tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. En la aplicación se evidencia ya que se tiene un promedio de 2 relaciones por clases.

Alta Cohesión

Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. En la solución se evidencia ya que las clases tienen un promedio de 5 métodos por clases.

2.5 Modelo de Datos

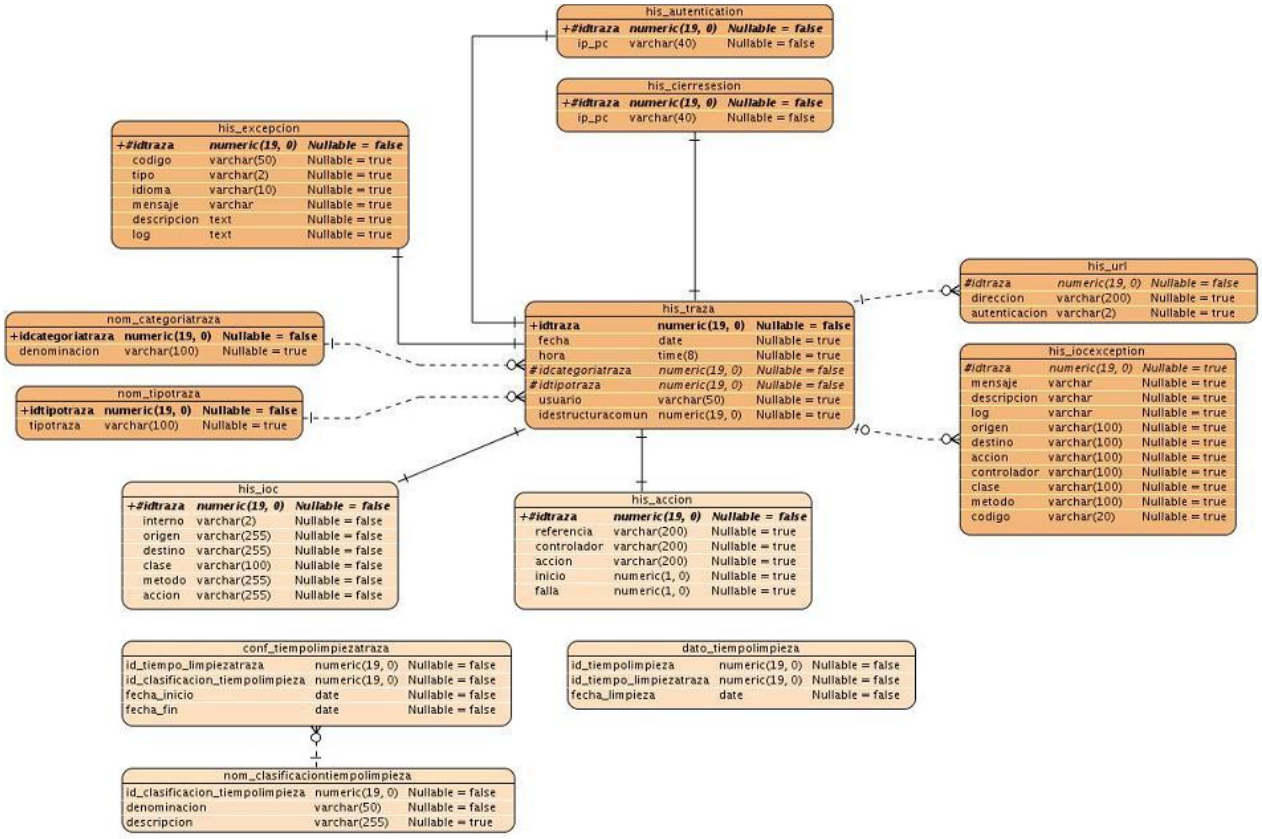

Figura 9 Modelo de Datos de Trazas.

Al modelo de dato de trazas se le agregaron varias tablas con el objetivo de configurar el tiempo de salva de las trazas de acción e integración para limpiar estas tablas de forma manual o automática y guardarlas para otras tablas, así evitar que se sobrecarguen debido al gran volumen de información. En estas tablas se aplican índices para lograr un mayor rendimiento en las consultas de la base de datos. A continuación se muestra unas pruebas realizadas.

Casos de Prueba	Búsqueda secuencial	Búsqueda por índices
Caso prueba #1 Consulta modificar usuario	Cantidad total de tuplas: 208037 Tiempo: 108,581ms	Cantidad total de tuplas: 208037 Tiempo: 2,684ms
Caso prueba #2 Consulta insertar usuario	Cantidad total de tuplas: 208037 Tiempo: 110,267ms	Cantidad total de tuplas: 208037 Tiempo: 5,358ms

Consultas realizadas
SELECT his_accion.accion FROM mod_traza.his_accion WHERE his_accion.accion = 'insertarusuario';
SELECT his_accion.accion FROM mod_traza.his_accion WHERE his_accion.accion = 'modificarusuario';

Tabla 4 Resultados de comparación entre búsqueda secuencial y por índices.

2.6 Conclusiones

En este capítulo fueron expuestos los artefactos generados durante el análisis y diseño de la solución propuesta. En el mismo se realizó el modelo conceptual, se identificaron y describieron los requisitos funcionales del componente propuesto y los requisitos no funcionales para garantizar la correcta ejecución de la aplicación. Se mostró el diseño de clases para lograr una mayor comprensión de cómo están estructuradas las clases que conforman el componente. Se explicó el uso de diferentes patrones de arquitectura y diseño empleados en la aplicación. Una vez concluido el diseño de la solución puede darse paso a los flujos de implementación y prueba de la misma.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

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.

3.2 Modelo de Implementación

En el modelo de implementación se comienza con el resultado de la etapa de diseño e implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. El objetivo principal de la etapa de implementación es desarrollar la arquitectura y el sistema como un todo. De forma más específica, define la organización del código, planifica las integraciones del sistema necesarias para cada iteración e implementa los subsistemas y las clases encontrados durante el diseño. (23)

3.2.1 Diagrama de Componentes

La solución propuesta consta de dos componentes que pertenecen al componente de Traza. El componente de Registro de Trazas que está relacionado con el componente de Configuración de Trazas. Ambos utilizan servicios del componente de Seguridad como getSistem y getRutaSistemaMod. En el componente de Configuración se realiza toda la configuración de las trazas y el de Registro comprobando si las trazas están configuradas en el componente anterior se encarga del registro de las trazas.

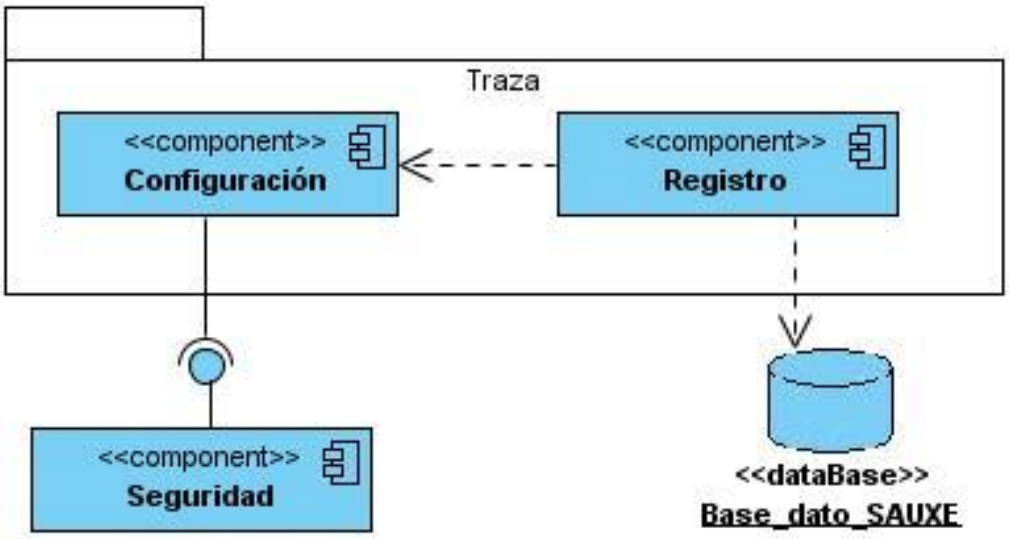

Figura 10 Diagrama de Componentes

3.2.2 Diagrama de despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software. Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación.

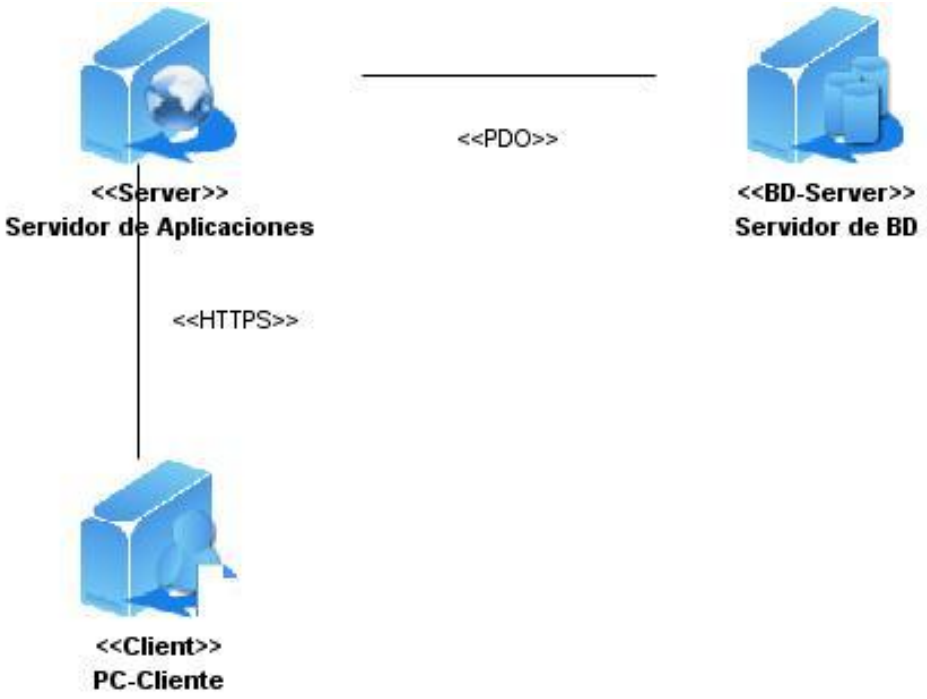

Figura 11 Diagrama de Despliegue

En el despliegue de la solución deben estar instaladas varias aplicaciones. Los servidores de aplicaciones se recomienda tener sistema operativo Ubuntu Server, servidor web Apache 2.0 y librerías adicionales de PHP 5. Los servidores de Base de Datos deben tener sistema operativo Ubuntu Server y sistema gestor de base de datos Postgres 8.3.8. Las PC cliente deben tener sistema operativo Linux o Windows, navegador web Morzilla Firefox 2.2 ó superior, herramientas ofimáticas y visualizador de ficheros pdf.

3.3 Métricas para la validación diseño

Las métricas que se utilizaron para validar el diseño propuesto son las siguientes:

- Tamaño operacional de la clase (TOC).
- Relaciones entre clases (RC).

3.3.1 Métrica Tamaño Operacional de Clase (TOC)

Tamaño operacional de clase (TOC)

Descripción:	Está dado por el número de métodos asignados una clase.
Atributos que afecta:	**Modo en que lo afecta:**
Responsabilidad	El aumento del TOC provoca un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	El aumento del TOC provoca un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC provoca una disminución en el grado de reutilización de la clase.

Tabla 5 Métricas para el tamaño operacional de la clase.

Criterios de evaluación para la métrica TOC.

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

Tabla 6 Criterio de evaluación de las métricas TOC

Instrumento de evaluación de la métrica TOC.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
ConfigurartrazaaccionController	6	Media	Media	Media
ConfigurartrazaintegracionController	6	Media	Media	Media
Trace	12	Alta	Alta	Baja
HisAccion	3	Baja	Baja	Alta
HisTraza	2	Baja	Baja	Alta
HisIoC	2	Baja	Baja	Alta
Action	6	Media	Media	Media
IoC	7	Media	Media	Media
Log	9	Media	Media	Media
Db	2	Baja	Baja	Alta
BuscarModel	3	Baja	Baja	Alta
Zend_Ext_Trace	3	Baja	Baja	Alta

Tabla 7 Instrumento de evaluación de las métricas TOC.

Resultados obtenidos de la aplicación de la métrica TOC

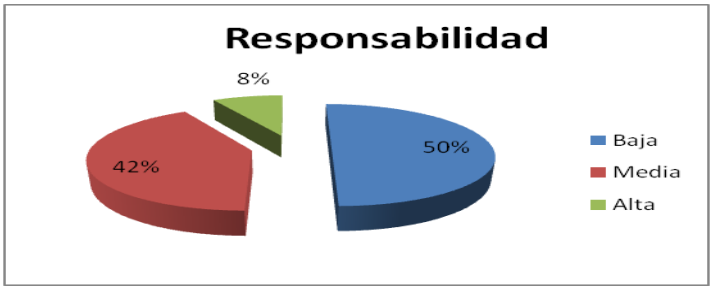

Figura 12 Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad

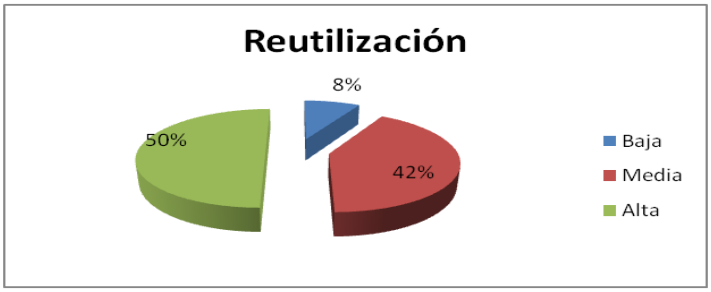

Figura 13 Resultados de la evaluación de la métrica TOC para el atributo Reutilización

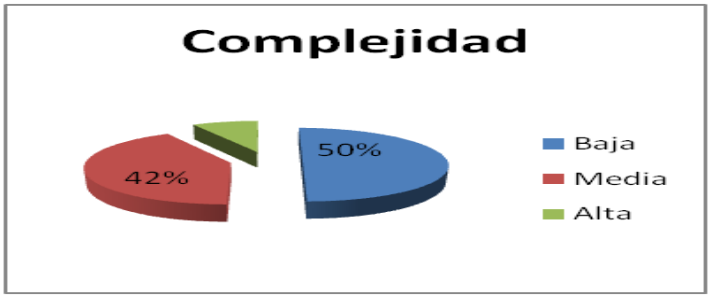

Figura 14 Resultados de la evaluación de la métrica TOC para el atributo Complejidad

3.3.2 Métrica Relación entre Clases (RC)

Relaciones entre clases (RC)	
Descripción:	Está dada por el número de relaciones de uso de una clase con otras.
Atributos que afecta:	**Modo en que lo afecta:**
Acoplamiento	El aumento del RC provoca un aumento del acoplamiento de la clase.
Complejidad del mantenimiento	El aumento del RC provoca un aumento de la complejidad del mantenimiento de la clase.

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

Tabla 8 Métricas de relación entre clases (RC).

Métrica Relaciones entre Clases (RC)

Criterios de evaluación para la métrica RC.

Atributo	Categoría	Criterio
Acoplamiento	Ninguna	0
	Baja	1
	Media	2
	Alta	>2
Complejidad y Mantenimiento	Baja	< =Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	> 2* Prom.
Cantidad de Pruebas	Baja	< =Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	> 2* Prom.
	Baja	> 2* Prom.

Reutilización	Media	Entre Prom. y 2* Prom.
	Alta	< =Prom.

Tabla 9 Criterios de evaluación de las métricas RC.

Instrumento de evaluación de la métrica RC.

Clase	Cantidad de relaciones de uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
ConfigurartrazaaccionController	5	Alta	Alta	Baja	Alta
ConfigurartrazaintegracionController	5	Alta	Alta	Baja	Alta
Trace	6	Alta	Alta	Baja	Alta
HisAccion	1	Baja	Baja	Alta	Baja
HisTraza	1	Baja	Baja	Alta	Baja
HisIoC	1	Baja	Baja	Alta	Baja
Action	0	Ninguno	Baja	Alta	Baja
IoC	0	Ninguno	Baja	Alta	Baja
Log	1	Baja	Baja	Alta	Baja
Db	2	Media	Baja	Alta	Baja
BuscarModel	0	Ninguna	Baja	Alta	Baja
Zend_Ext_Trace	2	Media	Baja	Alta	Baja

Tabla 10 Instrumento de medición de las métricas RC.

Resultados obtenidos de la aplicación de la métrica RC

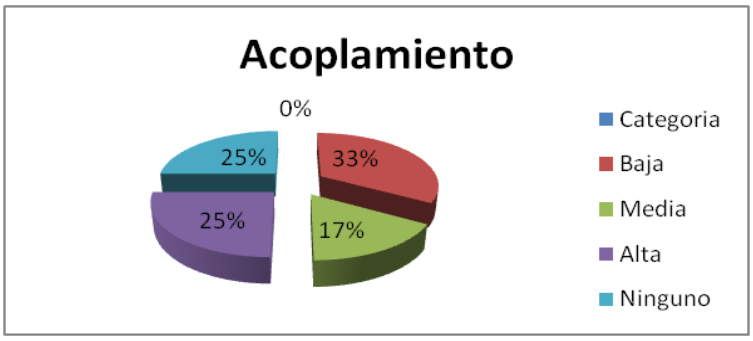

Figura 15 Resultados de la evaluación de la métrica RC para el atributo Acoplamiento.

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

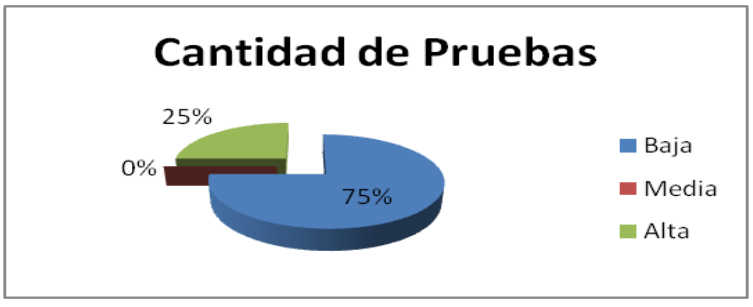

Figura 17 Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas.

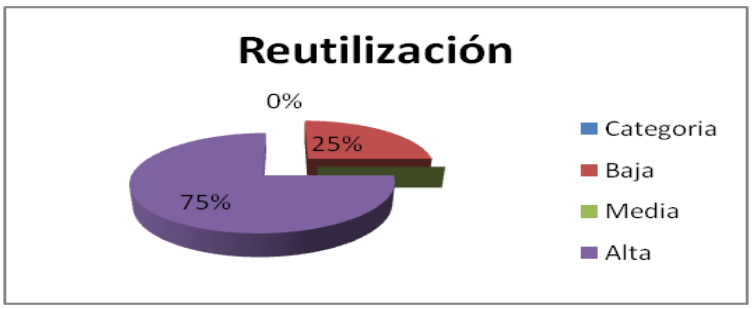

Figura 18 Resultados de la evaluación de la métrica RC para el atributo Reutilización.

La matriz inferencia de indicadores de calidad o matriz de cubrimiento, es una representación estructurada de los atributos de calidad y métricas utilizadas para evaluar la calidad del diseño propuesto. Esta matriz permite conocer si los resultados obtenidos de las relaciones atributo/métrica es positivo o no, llevando estos resultados a una escala numérica donde, 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.

A continuación se muestran los resultados obtenidos.

Resultados de la evaluación de la relación atributo/métrica.

Atributos\Métricas	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 11 Resultados de la evaluación de la relación atributo/métrica.

Rango de valores para la evaluación de la relación atributo/métrica

Categoría	Rango de Valores

Malo	<= 0.4
Regular	>0.4 y < 0.7
Bueno	< 0.7

Tabla 12 Rango de valores para la evaluación de la relación atributo/métrica.

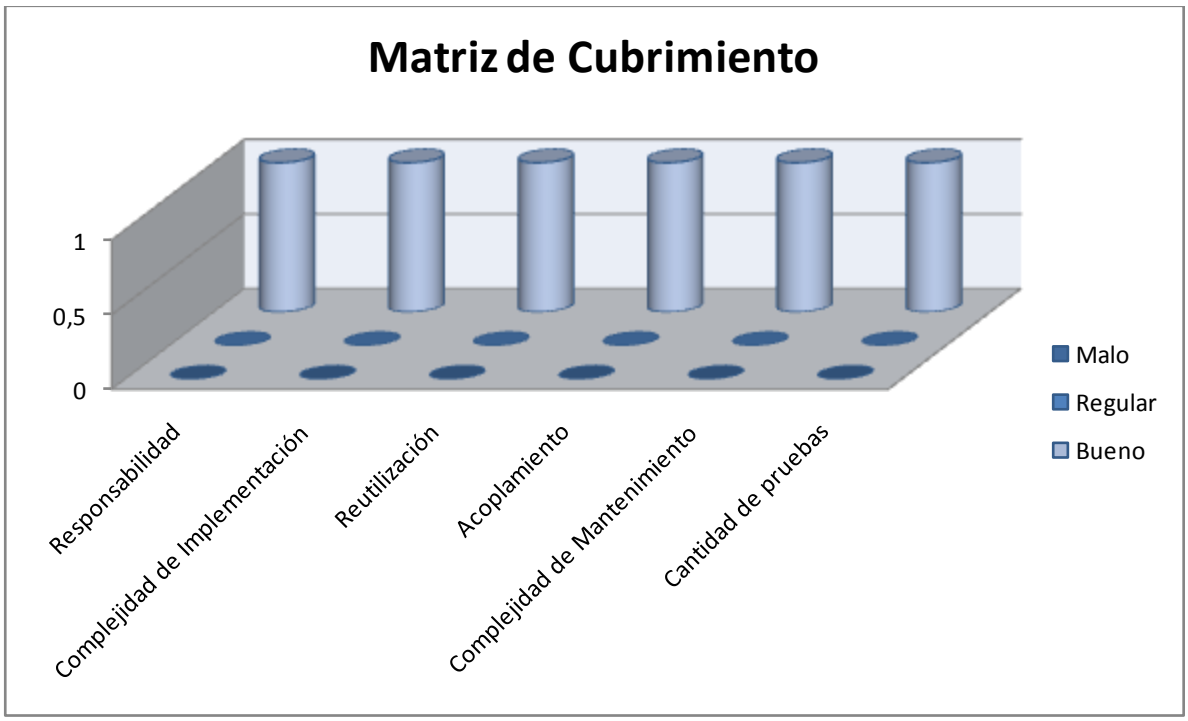

Figura 19 Resultados de la evaluación de los atributos de calidad.

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica TOC y RC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que en la matriz de cubrimiento todos los atributos de calidad dieron 1 como resultado (responsabilidad, complejidad de implementación y reutilización).

3.4 Pruebas de software

3.4.1 Pruebas de Rendimiento

Para realizar las pruebas de rendimiento a la aplicación se usó la herramienta **JMeter**, se realizaron una serie de pruebas con el fin de probar la mejora de rendimiento usando la solución desarrollada.

Caso de prueba Agregar configuración de trazas de acción y Agregar configuración de trazas de integración.						
casos	cantidad de usuarios	1ra iteración	2da iteración	3ra iteración	4ta iteración	promedio
Antes de la solución	50	33,7s	36,7s	35,4s	34,6s	35,1s
	100	37,7s	32,7s	34,2s	37,8s	35,6s
	200	36,9s	36,7s	37,0s	37,4s	37,0s
Después de la solución	50	32,7s	32,7s	31,2s	36,4s	33,25s
	100	34,4s	34,2s	33,4s	36,5s	34,625s
	200	34,0s	36,8s	37,9s	36,9s	36,4s

Tabla 13 Pruebas de rendimiento.

En la tabla anterior se muestran las pruebas de rendimiento realizadas a la aplicación, para las diferentes cantidades de usuarios realizando las acciones concurrentemente se realizaron 4 iteraciones en cada una (50,100 y 200), las mismas arrojaron un promedio de los resultados de la aplicación anterior a las solución y la posterior con la solución incluida, demostrando la mejora en rendimiento de la aplicación.

3.4.2 Pruebas de caja blanca

Para realizar la prueba de caja blanca específicamente la prueba del camino básico es necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se enumera las sentencias de código del procedimiento realizado sobre el método **confTrazaAccion ()** el cual se encarga de buscar en el XML de configuración si esta la acción realizada para registrarla.

```
public function confTrazaAccion($nombSubsistema, $controller, $action,$tipo) {
    $valor=false;
    $controller.='controller'.'.'.'php'; 1
    $direccion=(String)$_SERVER['DOCUMENT_ROOT']; 1
    $dir = substr($direccion,0,strlen($direccion)- 3); 1
    $dirXml = $dir.'apps/'.$nombSubsistema.'/comun/recursos/xml/'.$tipo.$nombSubsistema.'.xml'; 1
    $xml = simplexml_load_file($dirXml); 1
    if ($xml) { 2
        $traz = new SimpleXMLElement($xml->asXml()); 3
        if ($traz->$controller) { 4
            for ($i = 0; $i < count($traz->$controller->acciones->nombre); $i++) { 5
                if ($traz->$controller->acciones->nombre[$i] == $action) { 6
                    $valor=true; 7
                    break; 7
                } 8
            } 9
        } 10
    } 11
    return $valor; 12
} 13
```

Figura 20 Código fuente de la funcionalidad confTrazaAccion().

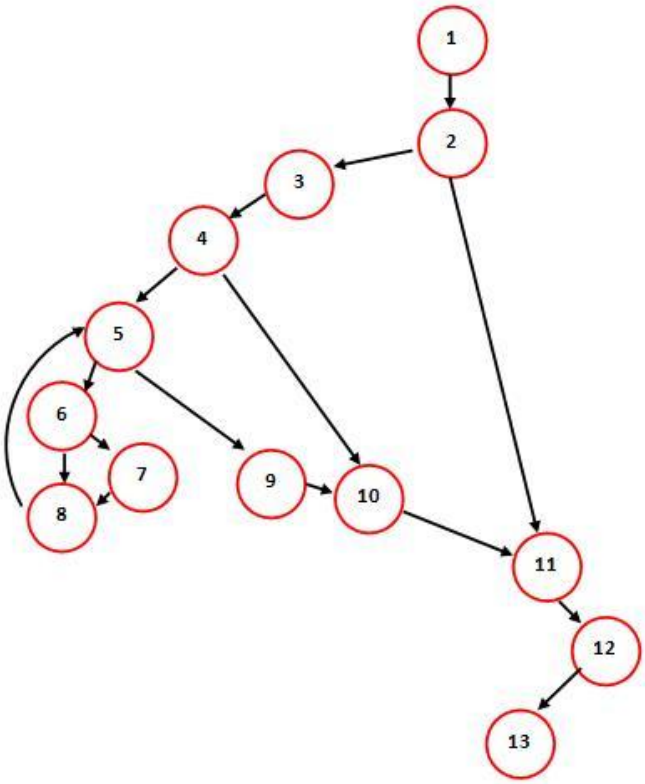

Figura 21 Grafo de flujo correspondiente a la funcionalidad confTrazaAccion().

Cálculo de la complejidad ciclomática:

El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.

Para efectuar el cálculo de la complejidad ciclomática del código es necesario tener varios parámetros como son la cantidad total de aristas del grafo, cantidad total de nodos para la siguiente fórmula:

V (G) = (A – N) + 2
V (G) = (16 – 13) + 2
V (G) = 5

Siendo: A la cantidad total de aristas y N la cantidad total de nodos. Se puede usar también:

V (G) = P + 1

V (G) = 4 + 1 V (G) = 5

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

V (G) = R

V (G) = 5

Siendo: R la cantidad total de regiones, para cada fórmula (G) representa el valor del cálculo.

Dado que el resultado de las 3 fórmulas fue el mismo, se puede plantear que la complejidad ciclomática del método es 5. Esto significa que existen cinco posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado. Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo.

Número	Camino Básico
1	1-2-11-12-13
2	1-2-3-4-10-11-12-13
3	1-2-3-4-5-9-10-11-12-13
4	1-2-3-4-5-6-8-5-9-10-11-12-13
5	1-2-3-4-5-6-7-8-5-9-10-11-12-13

Caso de prueba para el camino básico #1:

Camino: [1-2-11-12-13]

Descripción:

- ✓ Muestra si la acción realizada esta definida en el XML de configuración.

Condición de ejecución:

- ✓ Debe realizarse alguna acción.

Entrada:

- $subsistema
- $controller
- $action
- $tipo

Resultado:

- ✓ No se encuentra el XML de configuración, devuelve falso y no se registra la traza.

Caso de prueba para el camino básico #2:

Camino: [1-2-3-4-10-11-12-13]

Descripción:

- ✓ Muestra si la acción realizada esta definida en el XML de configuración.

Condición de ejecución:

- ✓ Debe realizarse alguna acción.

Entrada:

- $subsistema
- $controller
- $action
- $tipo

Resultado:

- ✓ No se encuentra el controlador en el XML de configuración, devuelve falso y no se registra la traza.

Caso de prueba para el camino básico #3:

Camino: [1-2-3-4-5-9-10-11-12-13]

Descripción:

- ✓ Muestra si la acción realizada esta definida en el XML de configuración.

Condición de ejecución:

- ✓ Debe realizarse alguna acción.

Entrada:

- $subsistema
- $controller
- $action
- $tipo

Resultado:

- ✓ No se encuentra la acción dentro del controlador en el XML de configuración, devuelve falso y no se registra la traza.

Caso de prueba para el camino básico #4:

Camino: [1-2-3-4-5-6-8-5-9-10-11-12-13]

Descripción:

- ✓ Muestra si la acción realizada esta definida en el XML de configuración.

Condición de ejecución:

- ✓ Debe realizarse alguna acción.

Entrada:

- $subsistema
- $controller
- $action
- $tipo

Resultado:

- ✓ No se encuentra la acción dentro del controlador en el XML de configuración, devuelve falso y no se registra la traza.

Caso de prueba para el camino básico #5:

Camino: [1-2-3-4-5-6-7-8-5-9-10-11-12-13]

Descripción:

- ✓ Muestra si la acción realizada esta definida en el XML de configuración.

Condición de ejecución:

- ✓ Debe realizarse alguna acción.

Entrada:

- $subsistema
- $controller
- $action
- $tipo

Resultado:

- ✓ Encuentra la acción dentro del controlador en el XML de configuración, devuelve verdadero y se registra la traza.

Estas pruebas se le realizaron a los principales métodos de la aplicación, en la tabla número 13 se muestra el resultado de las pruebas realizadas a varios métodos importantes en la aplicación.

Resultado de pruebas de caja blanca	
Método	Complejidad ciclomática
cargarControladores()	2

cargarAcciones()	3
confTrazaAccion()	5
insertarTrazaAccion()	8

Tabla 14 Resultados de pruebas de caja blanca.

3.4.3 Pruebas de caja negra

Nombre del Requisito	Descripción general	Escenario de Prueba	Flujo del Escenario
1: Gestionar Configuración de Trazas de Acción.	Permite configurar las trazas y guardar la configuración en un XML	EP 1.1: Adicionar Configuración de Trazas de Acción.	En este escenario se debe: - seleccionar un subsistema donde pertenecen las acciones a tracear. - Seleccionar el controlador donde pertenecen las acciones a tracear seleccionado. - Seleccionar las acciones. - Seleccionar la opción aceptar. - El sistema muestra un mensaje: “se agregó la configuración correctamente”. - Se presiona el botón OK.
		EP 1.2: Eliminar Configuración de Trazas de Acción.	En este escenario se debe: - Seleccionar el subsistema donde pertenecen las acciones a dejar de tracear. - Seleccionar el controlador donde pertenecen las acciones a dejar de tracear. - Seleccionar las acciones. - Seleccionar la opción aceptar. - El sistema muestra un mensaje, se eliminó la configuración correctamente. - Se presiona el botón OK
2: Gestionar Configuración de	Permite Configurar las trazas y guardar la	EP 2.1: Adicionar Configuración de	En este escenario se debe: - seleccionar un

Trazas de Integración.	configuración en un XML	Trazas de Integración.	subsistema donde pertenecen las acciones a tracear. - Seleccionar el controlador donde pertenecen las acciones a tracear seleccionado. - Seleccionar las acciones. - Seleccionar la opción aceptar. - El sistema muestra un mensaje: “se agregó la configuración correctamente”. - Se presiona el botón OK.
		EP 2.2: Eliminar Configuración de Trazas de Integración.	En este escenario se debe: - Seleccionar el subsistema donde pertenecen las acciones a dejar de tracear. - Seleccionar el controlador donde pertenecen las acciones a dejar de tracear. - Seleccionar las acciones. - Seleccionar la opción aceptar. - El sistema muestra un mensaje, “se eliminó la configuración correctamente.” - Se presiona el botón OK.

Tabla 15 Casos de pruebas.

3.5 Aportes y beneficios de la investigación

El desarrollo de la solución trae varios beneficios, primeramente ofrece la posibilidad de configurar las trazas que serán registradas ya que anteriormente solo se podía decidir si tracear todas las acciones o ninguna. Otro de los beneficio es que reduce considerablemente la cantidad de trazas que serán registradas después de haber configurado solo las necesarias, esto se muestra mediante un estudio que se hizo al subsistema Seguridad como muestra, el cual mostró que 59 acciones son de explotación y de ellas solo 28 se definieron como necesarias para un 47% de las trazas registradas anteriormente.

Tomando como muestra el subsistema Seguridad entre el 1/2/2011 y el 1/4/2011 se realizó una comparación entre la anterior aplicación y la misma con la solución incorporada. Mediante el uso de la interfaz de reporte del componente de trazas se obtuvo que en esa fecha se había realizado 3241 trazas de acción y con la solución incluida solo 501 trazas de acción reduciendo la cantidad a un 15%.
Otra ventaja que brinda es permitir configurar cada qué tiempo se salva la información de las tablas en la base de datos para mejorar el rendimiento de las mismas, ya sea de forma manual o automática. También se mejora el rendimiento en las consultas a la base de datos con la utilización de índices.

3.6 Conclusiones

En este capítulo se mostraron los artefactos generados como parte del modelo de implementación de la solución propuesta, como diagrama de componente, diagrama de despliegue. 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, dado los resultados obtenidos, valorar de satisfactorio el diseño propuesto. Además se realizó una descripción de las pruebas estructurales y funcionales realizadas que permitieron comprobar el funcionamiento del sistema.

CONCLUSIONES

Al culminar la investigación se le dio cumplimiento a los objetivos planteados, alcanzando los resultados propuestos, un componente capaz de configurar las trazas de acción e integración, evitando la sobrecarga de información y agilizando el tiempo para el registro de trazas del marco de trabajo Sauxe.

- Se analizaron los fundamentos teóricos y las principales aplicaciones vinculadas al campo de acción, tanto a nivel nacional como internacional, demostrando la necesidad del nuevo sistema.
- Se realizó un estudio acerca de la metodología, tecnologías y herramientas definidas para obtener la nueva solución, reafirmando que eran las idóneas para ser utilizadas.
- Las métricas orientadas a clases empleadas, validaron que se hizo un correcto diseño de la solución.
- Las pruebas funcionales y estructurales realizadas a la aplicación validaron el correcto funcionamiento de la misma.

RECOMENDACIONES

Una vez concluida las conclusiones de este trabajo se recomienda:

- Continuar el estudio del tema con el objetivo de incluir nuevas funcionalidades en versiones posteriores del sistema.
- Realizar el despliegue del componente propuesto como parte del subsistema de Traza del Marco de trabajo Sauxe.
- Continuar el proceso de pruebas de software al componente para aumentar la eficiencia y confiabilidad en el mismo.

BIBLIOGRAFÍA

1. Metodología de Auditoría Informática. [En línea] 8 de febrero de 2011. http://www.ub.edu.ar/catedras/ingenieria/auditoria/tpmetodo/tpmetodo2.htm.

2. *HERRAMIENTA PARA EL REGISTRO Y MONITOREO DE TRAZAS EN EL SISTEMA DE GESTIÓN INTEGRAR CEDRUX.* **Bauta Camejo, Rene y Torres Galvez, Ariel.** Ciudad de la Habana : s.n., 2010.

3. Trac integrated SCM & Project Management . [En línea] [Citado el: 14 de 3 de 2011.] http://trac.edgewall.org/.

4. Open Source platform for e-government. [En línea] 15 de febrero de 2011. http://pruebas.openpopuli.com/manual/que_es_open_populi_framework.php..

5. **Alexander, Christopher.** *The timeless way of building.* New York : Oxford University Press, 1979.

6. **Tedeschi, Nicolás.** MSDN. [En línea] [Citado el: 18 de febrero de 2011.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.

7. **Larman, Graig.** *UML y Patrones Introducción al análisis y diseno orientado a objetos.* Mexico : s.n., 1999.

8. **Pérez, Sergio Hernández Cisneros Mileidy Magalys Sarduy.** Propuesta de modelo de desarrollo de software tecnológico del Centro de Soluciones de Gestión. *Propuesta de modelo de desarrollo de software tecnológico del Centro de Soluciones de Gestión.* la Habana : s.n., Junio de 2009. 5.

9. scribd. [En línea] [Citado el: 15 de 3 de 2011.] http://es.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

10. slideshare. [En línea] [Citado el: 20 de 3 de 2011.] http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

11. control de versiones con Subversion. [En línea] [Citado el: 11 de 3 de 2011.] http://svnbook.spears.at/nightly/es/svn-book.html#svn-ch-1-sect-1..

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

13. PostgresSQL - es Portal en español sobre PostgresSQL. [En línea] [Citado el: 16 de 3 de 2011.] http://www.postgresql.org.es/sobre_postgresql.

14. Indices and Keys PostgresSQL . [En línea] [Citado el: 15 de 3 de 2011.] http://www.postgresql.org/docs/7/static/indices.htm.

15. Java Hispano. [En línea] 16 de 6 de 2010. [Citado el: 13 de 3 de 2011.] http://www.javahispano.org/contenidos/es/disponible_para_descarga_netbeans_6_9/.

16. Zend Frameworks DES.com. [En línea] [Citado el: 14 de 3 de 2011.] http://manual.zfdes.com/es/introduction.overview.html.

17. Doctrine-PHP Object Persistence Libraries and More. . [En línea] [Citado el: 4 de 3 de 2011.] http://www.doctrine-project.org/.

18. SPARX Systems. [En línea] [Citado el: 19 de 3 de 2011.] http://www.sparxsystems.com.ar/resources/tutorial/uml-tutorial.html.

19. **Project, The Apache Jakarta.** Apache JMeter. [En línea] [Citado el: 19 de 3 de 2011.] http://jakarta.apache.org/jmeter/.

20. **ERP, Proyecto.** *Normas y estándares de Codificación del ERP.* Habana : s.n., 2008.

21. **Pressman, Roger S.** *Ingenieria de Software Un enfoque practico.* 2005. pág. 28.

22. New Software Factory. [En línea] [Citado el: 23 de abril de 2011.] http://www.newsoftwarefactory.com/implementacion.htm.

23. Tablespace Postgresql. [En línea] [Citado el: 14 de 3 de 2011.] http://www.postgresql.org/docs/8.1/static/manage-ag-tablespaces.html.

BIBLIOGRAFÍA CONSULTADA

1. **Bradenbaugb, Jerry.** *Aplicaciones Java Script.* Madrid : s.n., 2000.

2. **Pressman, Roger S.** *Ingenieria de Software Un enfoque practico.* 2005.

3. **Larman, Graig.** *UML y Patrones Introducción al análisis y diseño orientado a objetos.* Mexico : PRENTICE HALL, 1999.

4. **Schmuller, Joseph.** *APRENDIENDO UML EN 24 HORAS.* Mexico : PEARSON EDUCACION, 2000.

5. **José H. Canós, Patricio Letelier y Mª Carmen Penadés.** *Metodologías Ágiles en el Desarrollo de Software.* Alicante : s.n., 2003.

ANEXOS

Anexo 1: Prototipos de interfaces.

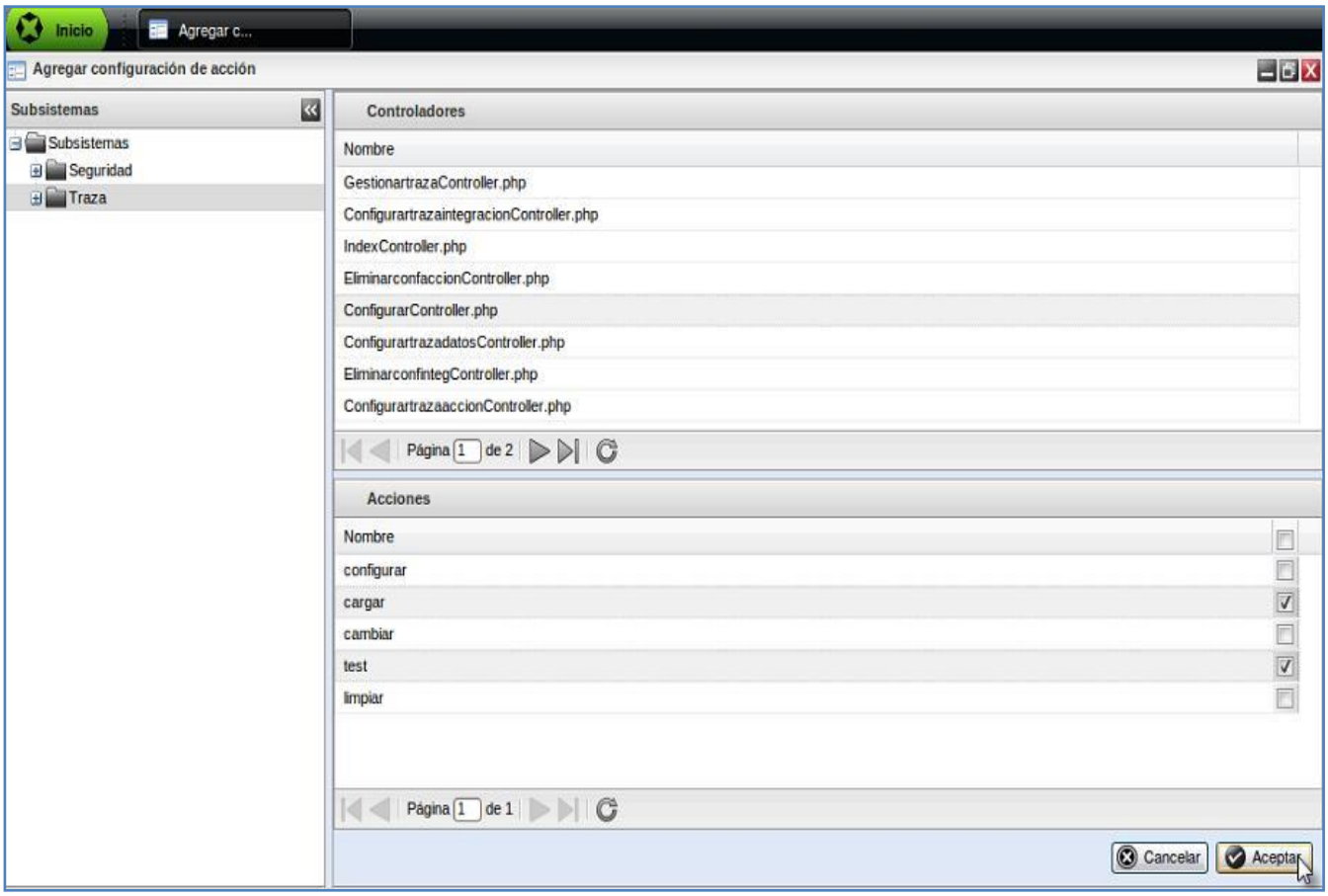

Figura 22 Prototipo de interfaz agregar configuración de acción seleccionar acciones.

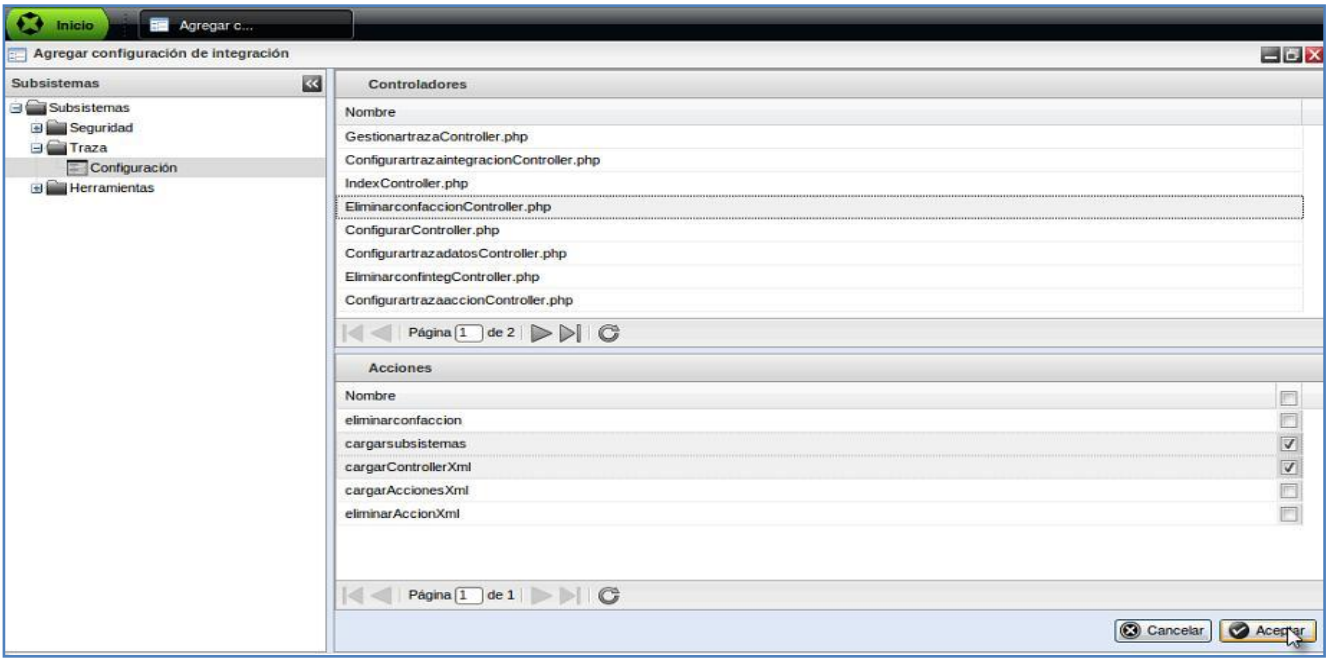

Figura 23 Prototipo de interfaz agregar configuración de integración seleccionar acciones.

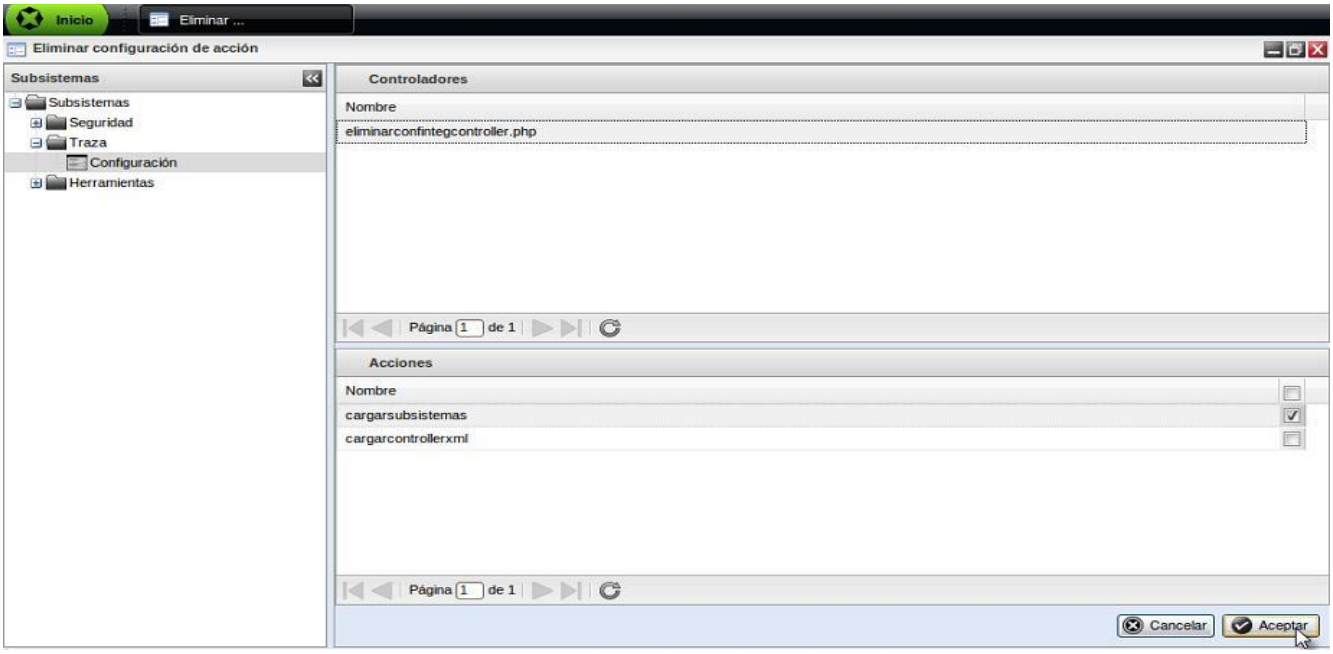

Figura 24 Prototipo de interfaz eliminar configuración de acción seleccionar acciones.

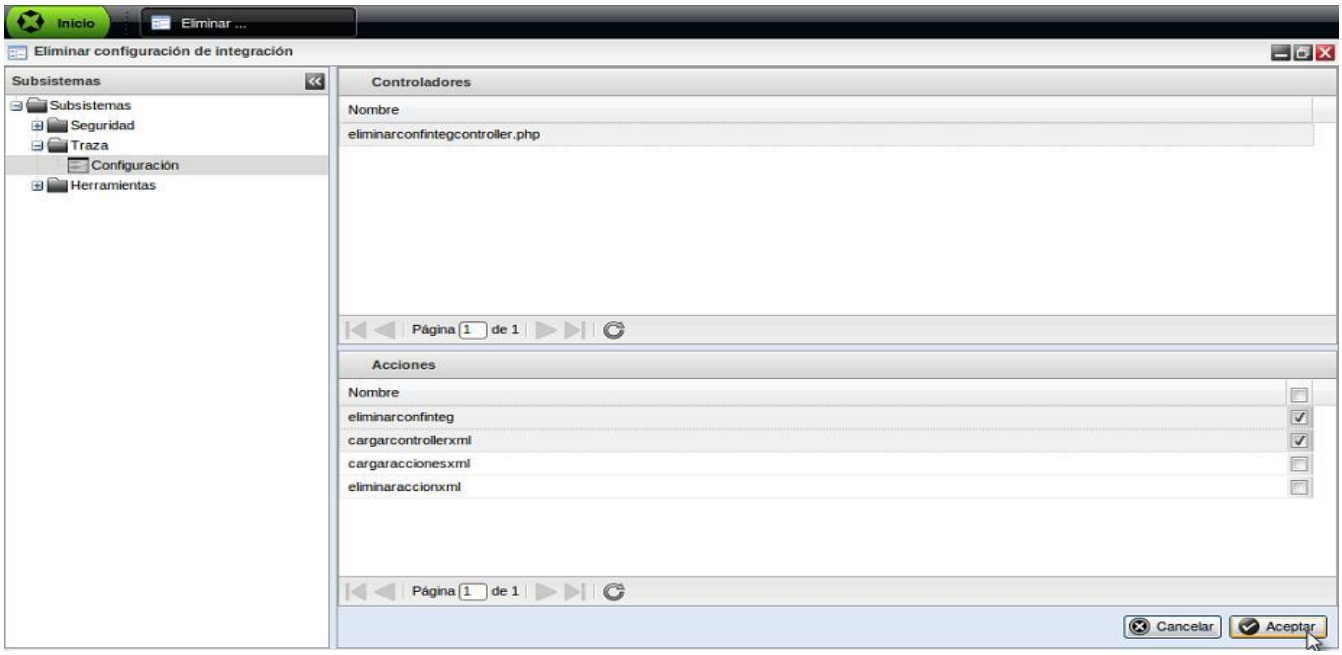

Figura 25 Prototipo de interfaz eliminar configuración de integración seleccionar acciones.

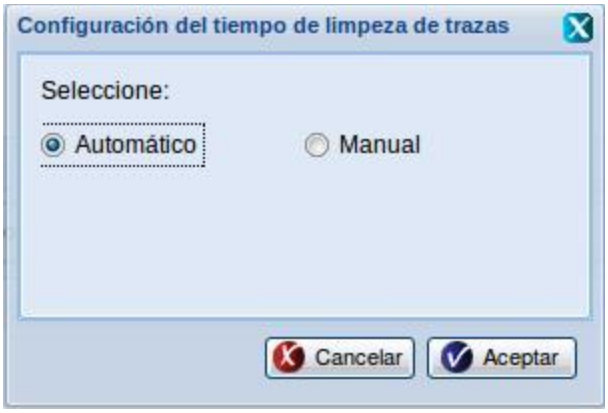

Figura 26 Prototipo de interfaz Configurar tiempo de limpieza de trazas.

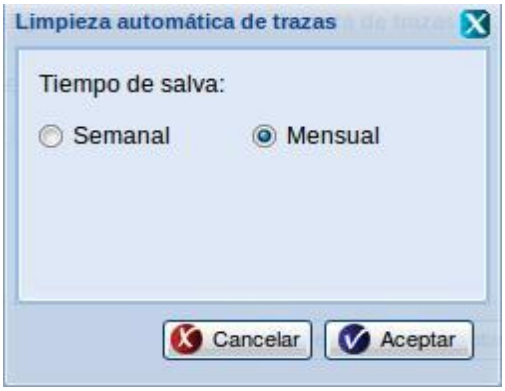

Figura 27 Prototipo de interfaz Configurar limpieza de trazas automática.

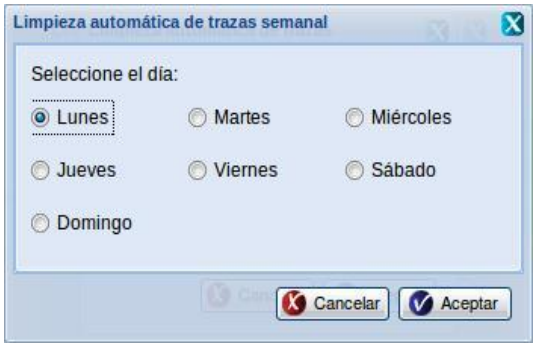

Figura 28 Prototipo de interfaz Configurar limpieza de trazas automática semanal.

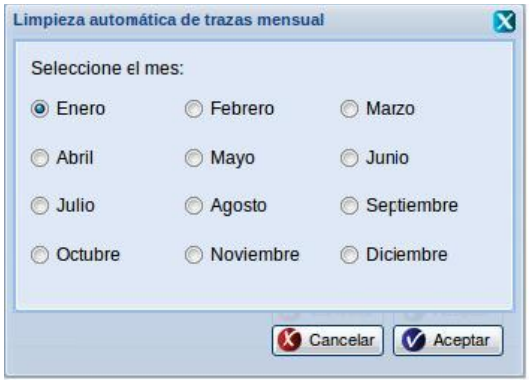

Figura 29 Prototipo de interfaz Configurar limpieza de trazas automática mensual.

Anexo 2: Requisitos Funcionales.

Precondiciones	Se ha realizado una acción dentro de la aplicación.
Flujo de eventos	
Flujo básico <<Nombre del flujo básico>>	
5	Se crea una traza con los campos idtraza, fecha, hora, idcategoria, idtipotraza, usuario, idestructuracomun.
6	El sistema obtiene los datos de la acción realizada y los compara con los datos de un XML
7	Si la acción está definida en el XML el sistema guarda las trazas en la base de datos.
8	Concluye el requisito.
Pos-condiciones	
3	Se registro una traza de integración.
Validaciones	
3	Se validan los datos según lo establecido en el Modelo conceptual Configuración de trazas de acción e integración.
Conceptos	
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 16 Especificación de requisito Registrar trazas de integración.

Precondiciones	Se ha registrado alguna traza de integración.
Flujo de eventos	
Flujo básico	
1	Se selecciona el subsistema a configurar.
2	El sistema muestra los controladores que posee el subsistema.
3	Se selecciona el controlador a configurar.
4	El sistema muestra las acciones que posee el controlador.
5	Se seleccionan las acciones a configurar.
6	El sistema guarda en un XML los datos de la configuración.
7	Concluye el requisito.
Pos-condiciones	
4	Se registró en el sistema una nueva configuración de traza de integración
Flujos alternativos	
Flujo alternativo	
Pos-condiciones	
1	N/A

Validaciones		
4	Se validan los datos según lo establecido en el Modelo conceptual Configuración de trazas de acción e integración.	
Relaciones	Requisitos Incluidos	Cargar Subsistemas
	Extensiones	N/A
Conceptos	N/A	Visibles en la Interfaz - Nombre Subsistema - Nombre Controlador - Nombre Acción

Tabla 17 Especificación de Requisito Agregar configuración de trazas de integración

Precondiciones	Debe existir al menos una configuración de trazas de acción.
Flujo de eventos	
Flujo básico	
1	Se selecciona el subsistema donde se va a eliminar la configuración.
2	El sistema muestra los controladores que posee el subsistema.
3	Se selecciona el controlador donde se va a eliminar la configuración.
4	El sistema muestra las acciones que posee el controlador.
5	Se seleccionan las acciones donde se va a eliminar la configuración.
6	El sistema actualiza el XML de la configuración.
7	Concluye el requisito.
Pos-condiciones	
5	Se elimino del sistema una configuración de traza de acción
Flujos alternativos	
Flujo alternativo	
Pos-condiciones	
1	N/A
Validaciones	
5	Se validan los datos según lo establecido en el Modelo conceptual Configuración de trazas de acción e integración.
Relaciones	Requisitos Incluidos: Cargar Subsistemas
	Extensiones: N/A
Conceptos	N/A — Visibles en la Interfaz - Nombre Subsistema - Nombre Controlador

		- Nombre Acción

Tabla 18 Especificación de Requisito eliminar configuración de trazas de acción.

Precondiciones		Debe existir al menos una configuración de trazas de integración.
Flujo de eventos		
Flujo básico		
1	Se selecciona el subsistema donde se va a eliminar la configuración.	
2	El sistema muestra los controladores que posee el subsistema.	
3	Se selecciona el controlador donde se va a eliminar la configuración.	
4	El sistema muestra las acciones que posee el controlador.	
5	Se seleccionan las acciones donde se va a eliminar la configuración.	
6	El sistema actualiza el XML de la configuración.	
7	Concluye el requisito.	
Pos-condiciones		
6	Se elimino del sistema una configuración de traza de integración	
Flujos alternativos		
Flujo alternativo		
Pos-condiciones		
1	N/A	
Validaciones		
6	Se validan los datos según lo establecido en el Modelo conceptual Configuración de trazas de acción e integración.	
Relaciones	Requisitos Incluidos	Cargar Subsistemas
	Extensiones	N/A
Conceptos	N/A	Visibles en la Interfaz - Nombre Subsistema - Nombre Controlador - Nombre Acción

Tabla 19 Especificación de Requisito eliminar configuración de trazas de integración.

Precondiciones		Debe existir al menos un subsistema.
Flujo de eventos		
Flujo básico		

1	El sistema hace una búsqueda de todos los subsistemas en la aplicación.
2	El sistema muestra en una estructura arbórea todos los subsistemas encontrados.
3	Concluye el requisito.
Pos-condiciones	
7	Se mostraron todos los subsistemas en un árbol.
Flujos alternativos	
Flujo alternativo	
Pos-condiciones	
1	N/A
Validaciones	
7	Se validan los datos según lo establecido en el Modelo conceptual Configuración de trazas de acción e integración.
Relaciones	Requisitos Incluidos N/A
	Extensiones N/A
Conceptos	N/A

Tabla 20 Especificación de Requisito Listar subsistemas.

Precondiciones	Existe alguna traza en la base de datos.
Flujo de eventos	
Flujo básico	
1	Se selecciona la forma de salvar, manual o automática
2	Si se selecciona automática el sistema muestra otra interfaz con la opción de seleccionar si se hace la limpieza semanal o mensual.
3	Si se selecciona semanal el sistema muestra una interfaz con la opción de seleccionar el día de la semana.
4	Se selecciona el día.
5	Se selecciona el botón aceptar.
6	Concluye el requisito.
Pos-condiciones	
8	Se realizó la configuración para salvar las trazas para otra tabla de forma automática
Flujos alternativos	
Flujo alternativo 3.a El usuario selecciona mensual.	
1	El sistema muestra una interfaz con la opción de seleccionar el mes.
2	Se selecciona el mes.
3	Se selecciona el botón aceptar.

4	Concluye el requisito.
Flujo alternativo 2.b El usuario selecciona manual.	
1	Se selecciona el botón aceptar.
2	Concluye el requisito.
Pos-condiciones	
1	Se realizó la configuración para salvar las trazas para otra tabla de forma manual.
Conceptos	Visibles en la Interfaz Automática Manual Día Mes

Tabla 21 Descripción de Requisito Configurar Salva de Trazas.

Anexo 3: Diagrama de Clases

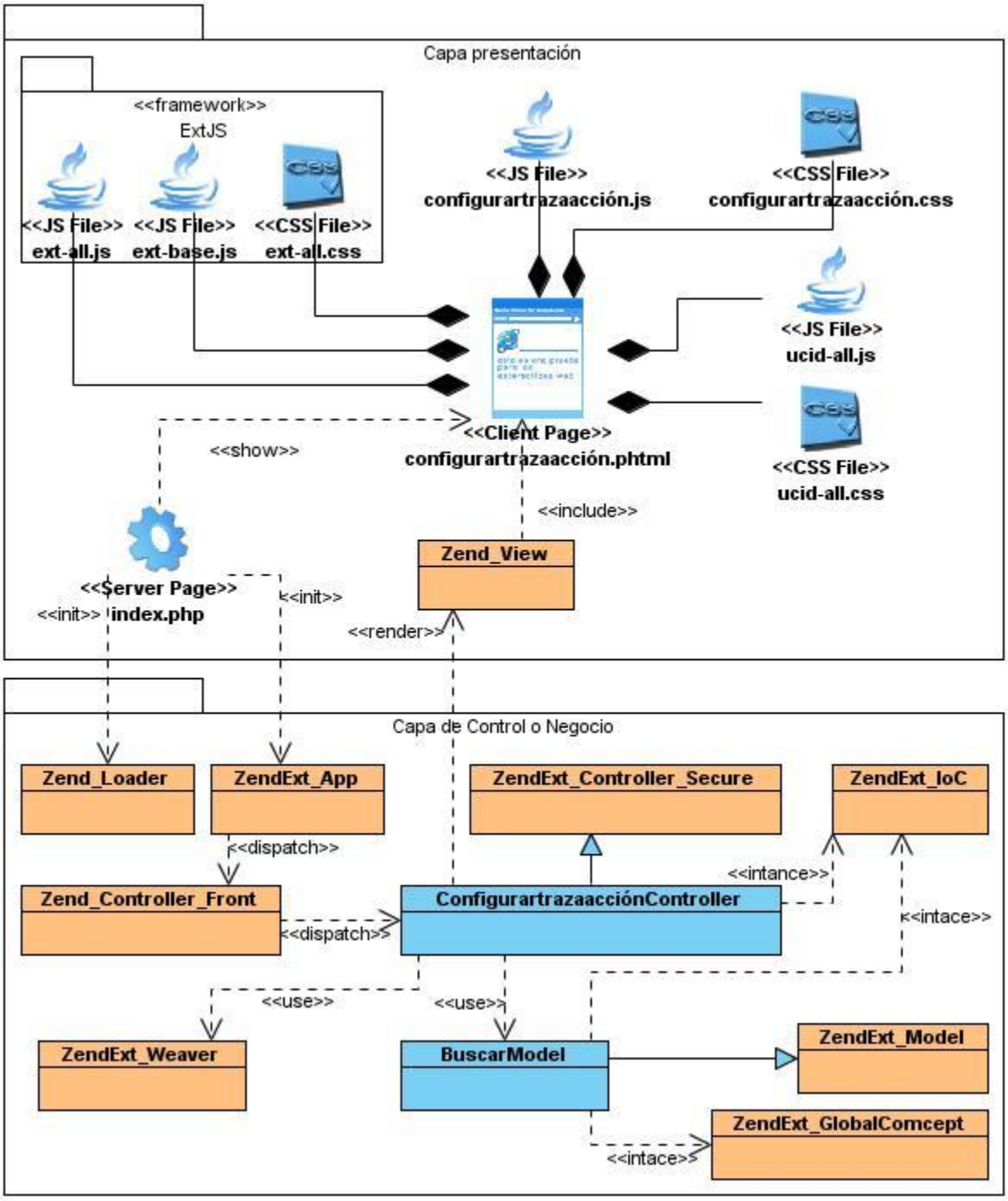

Figura 30 Diagrama de clases Agregar configuración de acción.

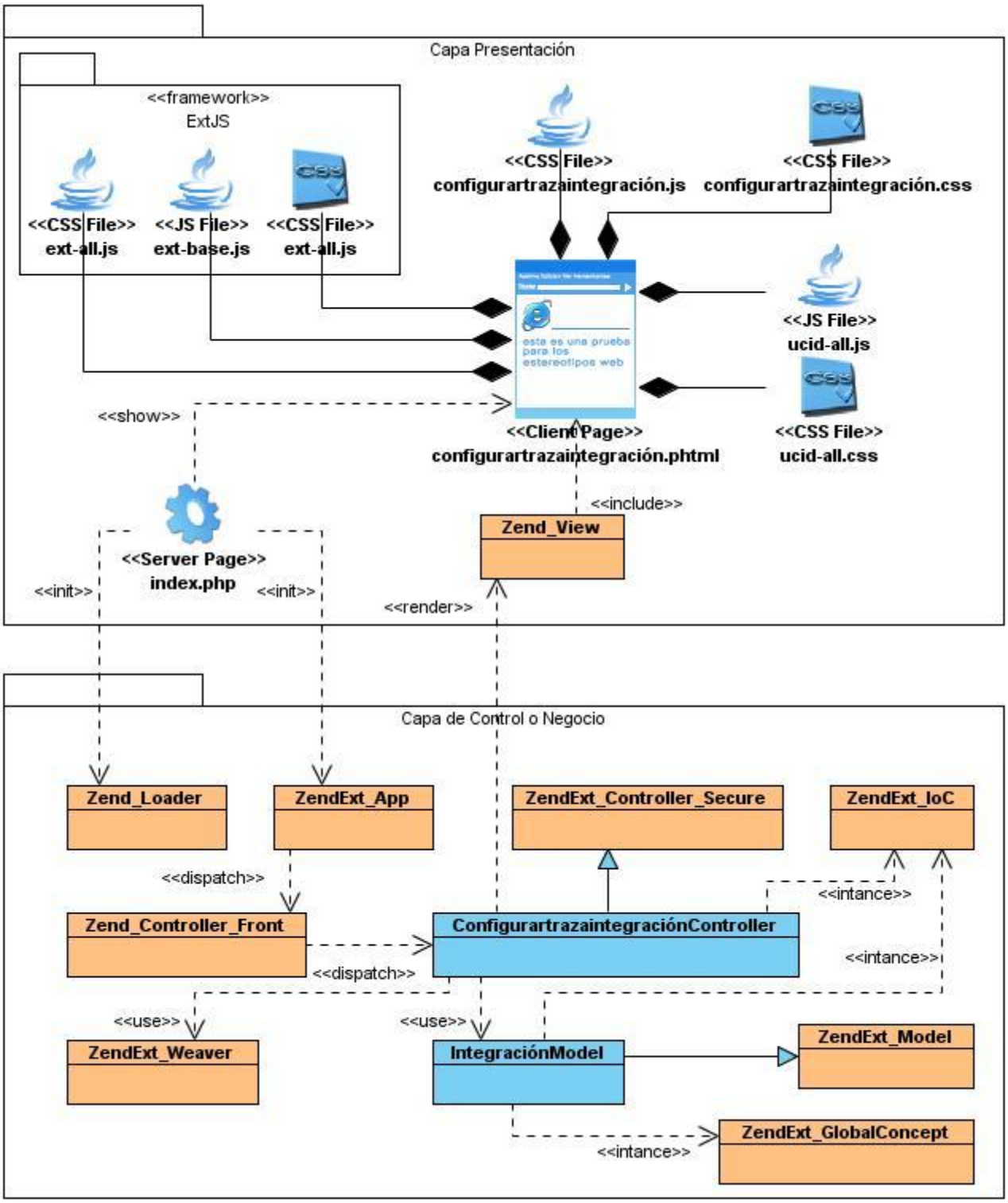

Figura 31 Diagrama de clases Agregar configuración de integración.

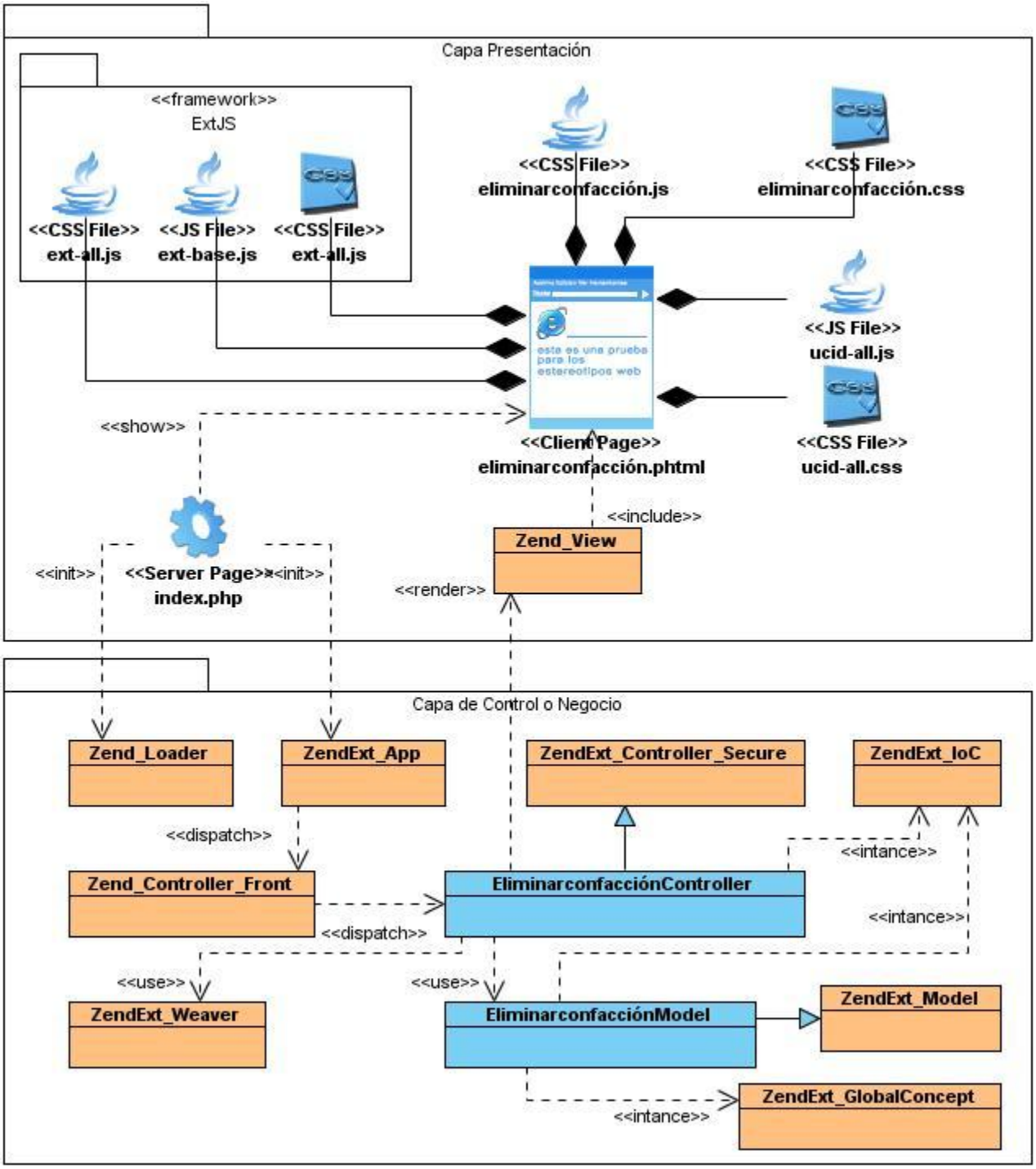

Figura 32 Diagrama de clases Eliminar configuración de acción.

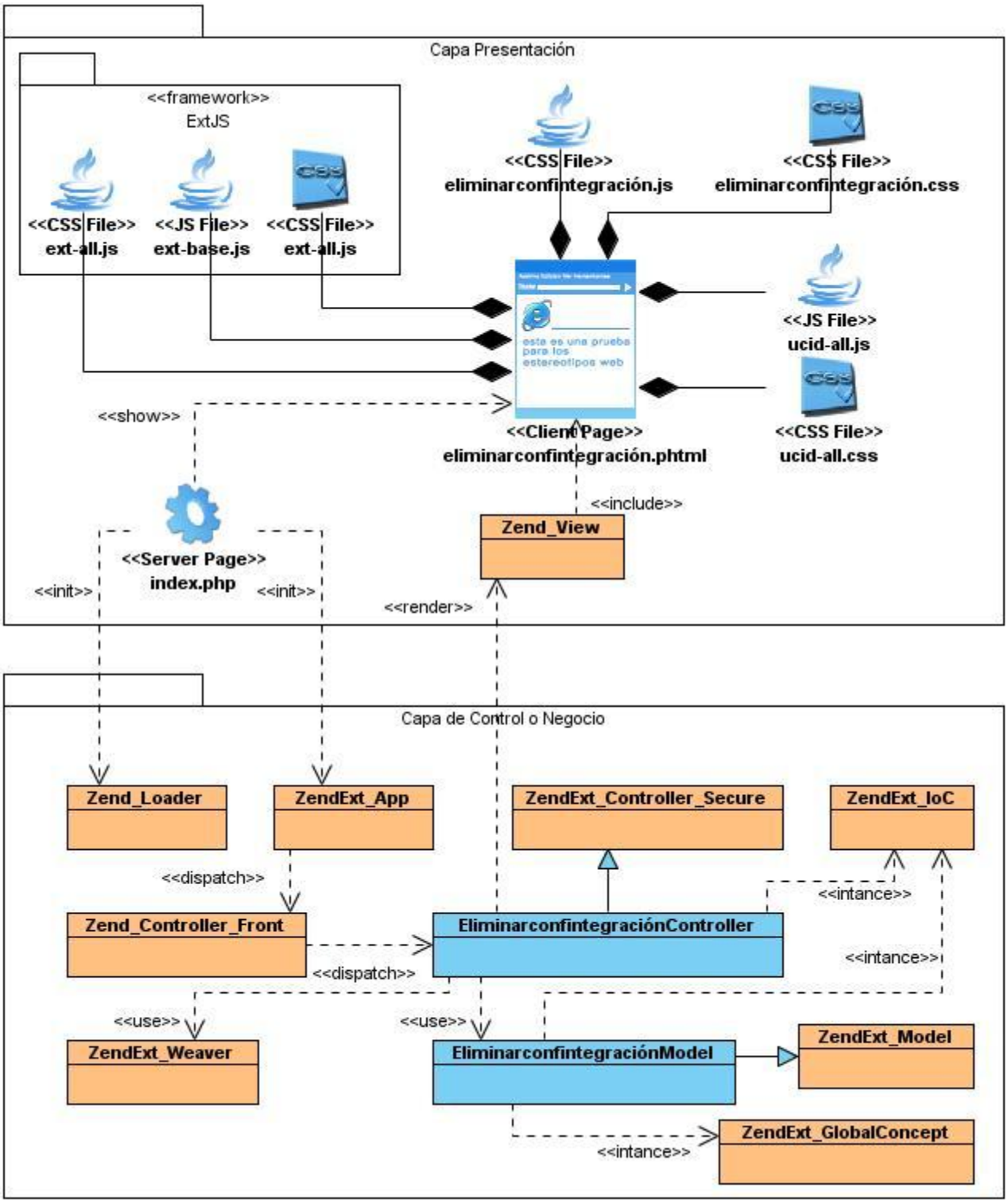

Figura 33 Diagrama de clases Eliminar configuración de integración.

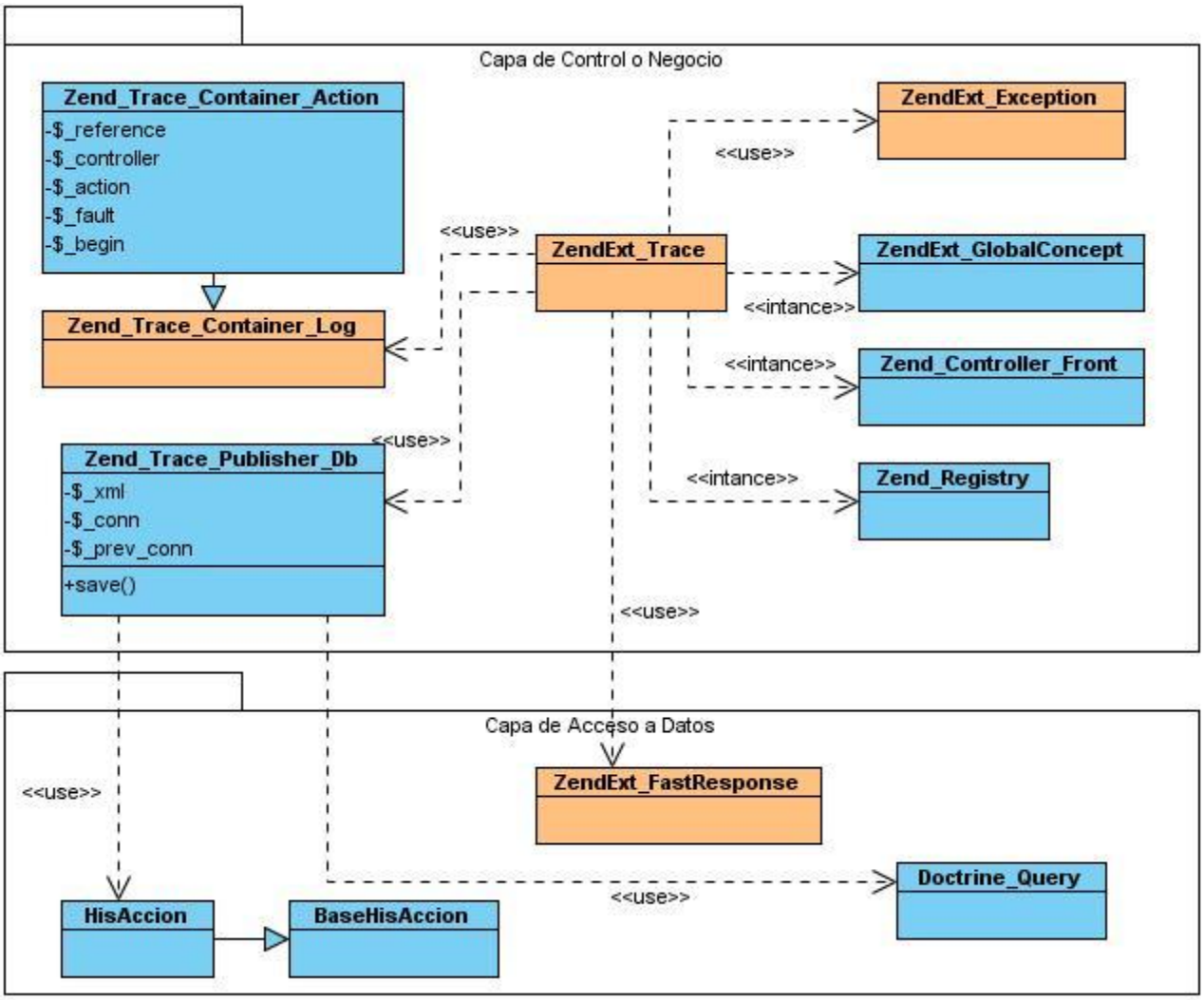

Figura 34 Diagrama de clases Registrar acción.

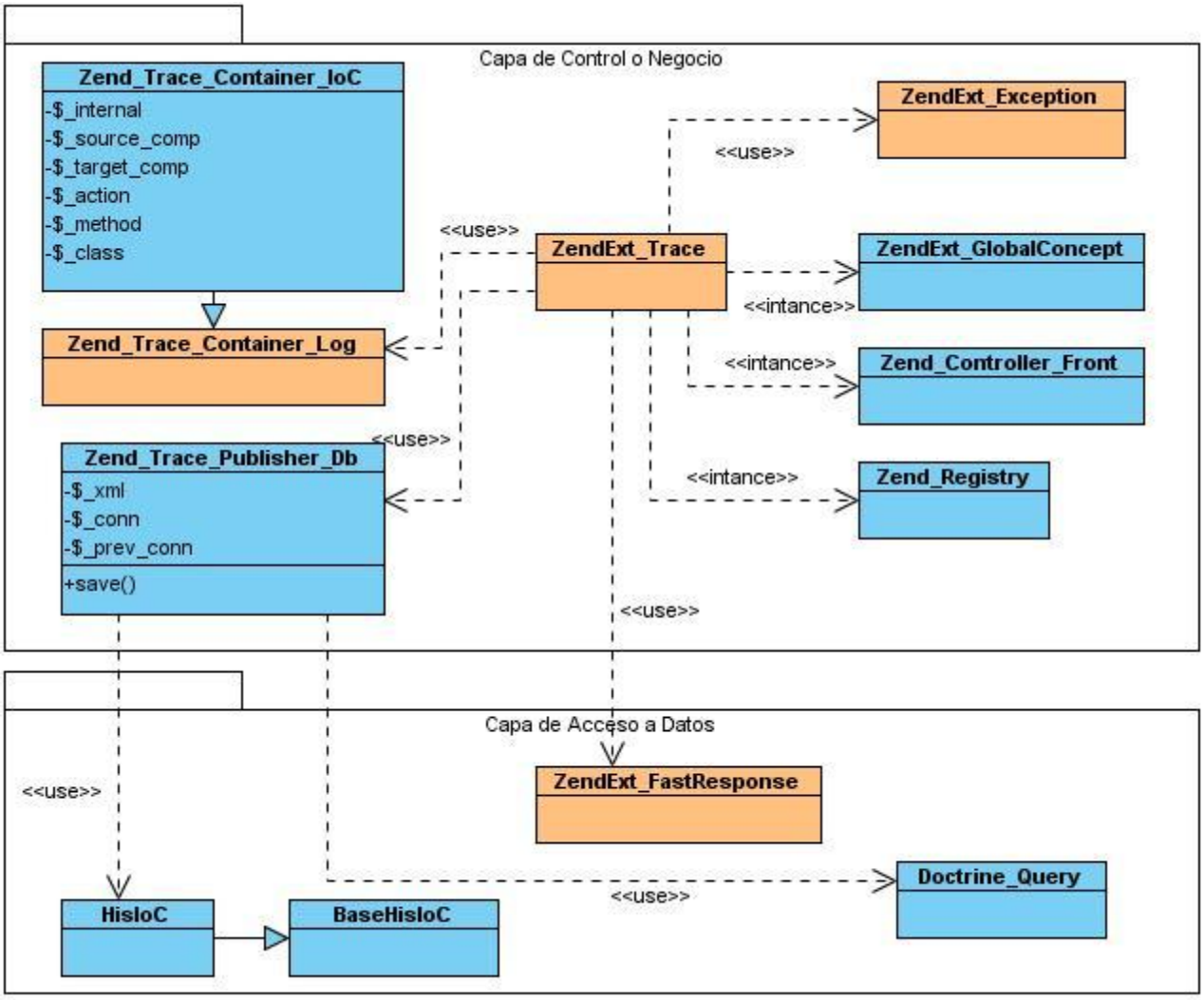

Figura 35 Diagrama de clases Registrar integración.

Anexo 4: Acta de aprobación de calidad.

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 8 horas efectivas de trabajo con la siguiente distribución: 4h (01/06/2011); 4h (03/06/2011)

Para el manual se emplearon un total de 2 horas efectivas de trabajo con la siguiente distribución: 1h (06/junio/2011) y 1 h (06/junio/2011).

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

Las pruebas se realizaron en un total de 2 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. Iliannis Pupo leyva

3.2 Otro personal especializado participante:

Nombre	Cargo
Ing. Giselle Almeida González	Especialista de Calidad
Ing.Susel Fernández Pérez	Especialista de Calidad
Jorge Rafael Pineda	Desarrollador

Ing. Giselle Almeida González
Especialista del Laboratorio de Calidad

Ing. René Bauta Camejo
Responsable por el Equipo de Desarrollo

INTERNA | 6

Figura 36 Acta de aprobación de Calidad.

GLOSARIO DE TÉRMINOS

- **Framework:** Estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.
- **UML:** Unified Modeling Language es un lenguaje de modelado de sistemas de software respaldado por el OMG.
- **Implementación:** Proceso por en el cual se escribe, se prueba, se depura y se mantiene el código fuente de un programa informático.
- **Módulo:** Parte de un sistema, que se instala y funciona por separado, integrándose con otros módulos con los que intercambian información.
- **Traza:** Son rastros, serie de huellas o pista dejadas por una persona u objeto.
- **Open Source:** Es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.