Universidad de las Ciencias Informáticas

Facultad 3

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

Título: Desarrollo de la Gestión de Trazas en el Sistema de Seguridad suAdminPlugin para la Aduana General de la República de Cuba.

Autor(a): Danarys Cancio Quintana

Tutor: Ing. Maurice Cabrejas Martínez

Ciudad de la Habana, 27 de junio del 2011.

"Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte."

Ernesto Guevara de la Serna

Declaración de autoría

Declaro que soy la única autora de este trabajo y autorizo al Departamento de Soluciones para la Aduana 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 2011.

Danarys Cancio Quintana	**Ing. Maurice Cabrejas Martínez**
_____	_____
Firma del Autor(a)	Firma del Tutor

Agradecimientos

Primero que todo agradecer a la Revolución Cubana por permitir que mi sueño se haya hecho realidad: Estudiar en esta universidad.

A mis padres y familiares por confiar en mí.

A Ale por haberme aguantado tantas malcriadeces durante los 4 años que estuvo conmigo, por apoyarme y ayudarme en todo, por darme lo mejor de sí y hacerme feliz.

A mi marido Adri, a Eli, a mi papá Maurice, a Weeden y a Ricardo por tener que cargar conmigo y ayudarme en el desarrollo de este trabajo, a ustedes, muchísimas gracias.

A mi mimín y familia, por quererme tanto, yo también los quiero mucho.

A mis amigas por haberme soportado durante estos 5 años: mis nanas Eli, Yuri, Yuya, Nely, Leydi, Lismary y Ailenis y a mis demás compañeras de aula.

A mis varones por quererlos tanto, por ayudarme en todo lo que pudieron y en lo que no, por considerarme además su embajadora en el grupo: A Tito, Oscar, a Rey, Grabiel, Driggs, a Álvaro, Sandy, a Pescuezo y Robert, a todos muchos besos, siempre los tendré presente.

En fin a todos los que me han apoyado y están hoy aquí.

A todos ustedes…, Muchas Gracias.

Danarys

Dedicatoria

A Idarmy y José Ramón, mis padres, como muestra fehaciente que mi formación profesional se la debo en parte a ellos. Por ser ejemplos a seguir en todo momento y quererme tanto.

A mi hermanito del alma, Tata te quiero muchote.

A mis dos abus: Rosita y Eneida, a las dos las quiero con la vida.

A mi tía Aida, tía Ángela, tío Silvio y tío Neno por ayudarme en todo y confiar en mí.

A mis dos adorables primas, Marbe y Deyvis, por ser como hermanas para mí, por brindarme su apoyo incondicional y darme tanto cariño.

A mis tíos y demás primos, como tengo tantos no puedo mencionarlos a todos. Pero para Uds. miles de besos.

Resumen

La Aduana General de la República de Cuba (AGR) ha sido designada como el órgano facultado para dirigir en materia aduanera, recaudar los derechos de aduanas y dar respuesta dentro de su jurisdicción y competencia a los hechos que incidan en el tráfico nacional e internacional de mercancías, viajeros, postal y los medios que los transportan, previniendo, detectando y enfrentando el fraude y el contrabando. Para esto se basa en leyes y regulaciones establecidas por los distintos organismos de la Administración Central del Estado. La misma ha decidido automatizar parte de sus procesos con el propósito de perfeccionar su funcionamiento.

En la Universidad de Ciencias Informáticas (UCI), específicamente en la facultad 3, se desarrolla el proyecto Gestión Integral de Aduanas (GINA), sistema que automatiza los procesos aduaneros. Por la importancia de estos procedimientos es necesario llevar registros rigurosos de las acciones realizadas sobre los subsistemas. Debido a la inexistencia de un sistema que facilite estos controles, se propone el desarrollo de un servicio que permita el registro de las trazas que sean generadas a partir de las acciones que son ejecutadas sobre los subsistemas que componen el GINA.

El presente trabajo se centra en el desarrollo de un componente que permita gestionar las trazas que son generadas a partir de las acciones que son ejecutadas por los usuarios sobre los subsistemas que pertenezcan a GINA; para lo cual fue necesario el estudio de otras soluciones informáticas existentes a nivel nacional e internacional. Se espera como resultado una aplicación que permita registrar de forma automática estas acciones.

Palabras claves: AGR, GINA, Trazas, Auditorías, Symfony.

Tabla de Contenido

Introducción

Con el objetivo fundamental de garantizar la seguridad nacional, el 5 de febrero de 1963, se crea la Aduana General de la República de Cuba (AGR), organización que controla el tráfico nacional e internacional de medios de transporte, mercancías y viajeros, tanto a la entrada como a la salida del país. Estas actividades están acompañadas de un alto nivel profesional y eficiencia de sus trabajadores.

La AGR se ha trazado como meta automatizar parte de sus procesos y digitalizar los documentos que utilizan en los mismos. En el Centro de Automatización para la Dirección y la Información de la AGR (CADI) se han desarrollado varios sistemas que cumplen estas metas; los que han ido quedando obsoletos por el cambio en las normativas de la Aduana y por el desarrollo vertiginoso que ha tenido el mundo de la informática en la actualidad. Por tal motivo, estos sistemas han sido objeto de constantes actualizaciones, lo que ha dificultado el desarrollo de nuevos productos que se necesitan, ya que no se cuenta con el personal necesario, así como el mantenimiento de los mismos.

Actualmente, en la Aduana, se emplea un software que automatiza, controla y gestiona los procesos aduanales. El SUA[1], sistema que anteriormente se mencionaba, automatiza los cinco procesos aduaneros (despacho comercial, no comercial, de viajeros, de medios de transporte y de postal y envío), posee una base de datos única y centralizada, a la que acceden en línea todas las aduanas del país. Este software ha sido desarrollado de forma estructurada, bajo malas prácticas de programación y se encuentra obsoleto ya que no se ajusta a los nuevos requerimientos. Carece además de una funcionalidad que permita controlar todas las acciones realizadas por los usuarios sobre los procesos mencionados anteriormente.

Por la importancia de estos procedimientos aduaneros se hace necesario llevar registros rigurosos de las acciones realizadas sobre los subsistemas. A partir de aquí se requiere desarrollar un nuevo producto que permita la ejecución de nuevas funcionalidades.

A petición de la AGR, en colaboración con la Universidad de las Ciencias Informáticas (UCI), y el CADI, se desarrolla el sistema GINA (Gestión Integral de Aduanas), en el Departamento de Soluciones

[1] Sistema Único de Aduanas

para la Aduana, perteneciente al Centro para la Gestión Integral de Entidades (CEIGE) de la facultad 3. El producto que se está desarrollando, actualmente, no posee un mecanismo que permita registrar las acciones que son generadas por los usuarios. Al carecer de dicha funcionalidad la seguridad del sistema se ve comprometida a posibles amenazas y ataques que puedan ocurrir durante la ejecución de las operaciones que se realizan en la institución.

Por las características del sistema GINA, el cual está desarrollado bajo la arquitectura de Symfony[2], ninguna herramienta destinada para el seguimiento de la trazabilidad satisface íntegramente los requisitos que exige la aplicación, ya que centran su funcionalidad en el registro y control de errores que se producen internamente en los sistemas y no en el mal uso por parte de los usuarios, como por ejemplo, no registran las acciones que ellos realizan sobre las aplicaciones, ni datos importantes relacionados con los eventos como es el caso de la fecha, la hora y el subsistema del que se realizó la operación; por lo que se hace imprescindible desarrollar una herramienta que permita registrar las trazas que son realizadas por los usuarios.

Al no poseer un mecanismo que posibilite el control de las acciones ejecutadas por los usuarios en los subsistemas que integran el Sistema GINA surge el siguiente **problema:** ¿Cómo gestionar las acciones ejecutadas sobre los subsistemas que componen el Sistema GINA para la AGR?

Derivado del problema anterior el **objeto de estudio** se enmarcará en los sistemas de Auditoría Informática. Teniendo como **campo de acción** los procesos de Auditoría en los sistemas aduaneros (GINA). Siendo el **objetivo general** de la investigación: Desarrollar un servicio de Auditoría que permita registrar las acciones ejecutadas sobre los subsistemas que componen el sistema GINA.

Para alcanzar el objetivo general se hace necesario el desglose de los siguientes **objetivos específicos**:

1. Evaluar el estado del arte acerca de la gestión de trazas en los subsistemas existentes en el mundo.
2. Realizar el análisis y diseño de la solución.
3. Implementar la solución propuesta.
4. Validar el funcionamiento de la solución propuesta.

[2] Framework desarrollado con PHP5. Diseñado para optimizar y simplificar el desarrollo de las aplicaciones web.

Para darle cumplimiento a todos los objetivos trazados durante la investigación, se han definido las siguientes tareas esenciales:

1. Evaluación del estado del arte acerca de la gestión de trazas en los subsistemas existentes en el mundo.
2. Realización del análisis y el diseño del producto.
3. Implementación del sistema.
4. Validación de la solución propuesta.

El presente trabajo de diploma está estructurado por 3 capítulos, los cuales se describen a continuación:

Capítulo 1: Se enuncian los principales conceptos relacionados con los sistemas destinados a la gestión de trazas. Se realiza un estudio de algunas soluciones existentes en el mundo, concluyendo con la selección de la más indicada para desarrollar la solución propuesta.

Capítulo 2: Se brinda la propuesta de solución y se puntualizan los requisitos del futuro componente. Se describen todos los artefactos que se generan durante el análisis y el diseño del sistema, se realiza el diseño de los diagramas de clases, secuencia y diagrama de paquetes. Además de explicar los patrones que se utilizan en el desarrollo del producto.

Capítulo 3: Se hace una descripción de las clases y componentes que se generan durante la implementación del sistema y se detallan las pruebas que se le realizan al componente, especificando las pruebas unitarias realizadas.

Capítulo 1: Fundamentación Teórica

Introducción

En los sistemas informáticos es necesario tener conocimiento de que es lo que se está haciendo en todo momento por un usuario. Esto se debe a que es de vital importancia poder monitorear que es lo que este hace durante su acceso al sistema para así poder conocer en caso de errores todos los datos referentes a estos; ninguno es una excepción. Ante comportamientos inesperados o el acceso de algún usuario, cuando un error inexplicable aparece, siempre se plantea la pregunta: ¿Qué está pasando? Para responder esa interrogante se hace necesario la utilización de un mecanismo de registro oficial de eventos (trazas) y datos o información sobre quién, qué, cuándo, dónde y por qué un evento ocurre, ya sea para un módulo del sistema en particular o en toda la aplicación.

En el presente capítulo se realizará un estudio del estado del arte relacionado con el problema a resolver. Se verán conceptos, tecnologías y software que se utilizan actualmente a nivel mundial, relacionadas con el control y registro de eventos, y que podrán ser utilizados en la solución del problema.

Para el mejor entendimiento de la investigación, se hace necesario conocer los conceptos fundamentales relacionados al dominio del problema.

Conceptos fundamentales

Las Trazas o Eventos

La traza es la marca dejada por un elemento al haberse desplazado a una nueva posición, por lo que llevándolo al contenido informático se puede decir que es una colección de eventos y datos devueltos por el Database Engine (Motor de base de datos) que representan el historial o rastro dejado por cierto proceso cuando se ejecuta en un sistema determinado. (1)

En el contexto de la presente investigación, se puede definir una traza como el registro persistente de la secuencia de acciones ejecutadas sobre un sistema informático que incluye el usuario que la generó, detalles sobre el origen, desarrollo y fin de la acción.

Sistemas de Gestión

La información puede llegar a ser el elemento decisivo que, en un momento dado, determine el éxito o el fracaso de un negocio. Con el fin de lograr su máxima utilidad, debe administrarse de manera correcta, como ocurriría con cualquier otro recurso de la empresa. (2)

Un sistema de gestión es aquel que establece un control sobre un conjunto de información que, relacionada y ordenada, contribuye al cumplimiento de un determinado objetivo.

Los sistemas de gestión se aplican en el marco de todas las actividades que se ejecuten en la organización y son válidos solo si cada uno de ellos interactúa con los demás armónicamente. Su estructura tiene que ser tal que sea factible realizar una coordinación y un control ordenado y permanente sobre la totalidad de las actividades que se realizan. (3)

Sistema de gestión de trazas

Luego un Sistema de Gestión de Trazas es aquel que registra y permite auditar los eventos que ocurren dentro de un sistema determinado, lo cual es muy importante para su correcto funcionamiento. Un producto informático que no posea un componente con esta funcionalidad implica grandes riesgos.

Al no existir dicho componente, muchos aspectos se verán afectados, a continuación se muestran tres de ellos:

- El control de las funciones informáticas de la empresa y/o institución.
- Cumplimiento de las normativas generales de la empresa y/o institución.
- Análisis de los controles y procedimientos tanto organizativos como operativos.

La siguiente exploración documental tiene como objetivo informar el conocimiento que ya se produjo respecto al control y registro de eventos, y comenzar a recuperar las nociones, conceptos, teorías, metodologías y perspectivas desde las cuales se interrogará al objeto de investigación que se está construyendo.

Auditoría Informática

La Auditoría informática tiene como principal objetivo, evaluar el grado de efectividad de las TIC's[3], dado que evalúa en toda su dimensión, en qué medida se garantiza la información a la organización, su grado de eficacia, eficiencia, confiabilidad e integridad para la toma de decisiones, convirtiéndola en el método más eficaz para tales propósitos.

Su ámbito de acción se centra, en revisar y evaluar: los procesos de planificación; inversión en tecnología; organización; los controles generales y de aplicación en proyectos de automatización de procesos críticos; el soporte de las aplicaciones; aprovechamiento de las tecnologías; sus controles específicos, los riesgos inherentes a la tecnología, como la seguridad de sus recursos, redes, aplicaciones, comunicaciones, instalaciones y otras.

Quedaría definida entonces la Auditoría Informática como el conjunto de procedimientos y técnicas que evalúan, parcial o totalmente, los controles internos de los Sistemas Informáticos; la protección de sus activos y recursos; verifica si su explotación se desarrolla con eficiencia, de acuerdo con las políticas y normativas establecidas por cada entidad y valora si se alcanza el grado de organización previsto para el marco donde participa y actúa. (4)

Estado del Arte en el control y registro de trazas.

La existencia de disímiles herramientas, con funcionalidades homólogas a la que se quiere desarrollar a partir del presente trabajo, tanto a nivel nacional, internacional y en la Universidad de las Ciencias Informáticas (UCI) es real. A continuación se brinda la explicación correspondiente a una muestra de las herramientas mencionadas, sus características principales y los motivos que imposibilitan su utilización.

[3] Tecnologías de la Informática y las Comunicaciones.

1.1 Herramientas de registro de trazas a nivel internacional.

1.1.1 Logging Application Block (LAB)

¿En qué consiste?

El bloque de registro de las aplicaciones (Logging Application Block, por sus siglas en inglés) provee extensiones de la arquitectura del framework .NET[4] para ayudar a las direcciones en el uso común de escenarios para la salva de las trazas. Dichos escenarios incluyen:

1. Formato de la información de los eventos.
2. Configuración de los niveles de registro.
3. Mejora y adiciona información de los eventos publicados.
4. Forma de registro asíncrona.
5. Registro fiable.
6. Registro de la información centralizada.
7. Salva registro de peticiones a los servicios web.
8. Medición de los servicios web.
9. Uso del bloque de administración de excepciones (EMAB: Exception Manager Application Block) a través del publicador de la instrumentación del framework de Microsoft (EIF: Microsoft Enterprise Instrumentation Framework).

Ventajas

1. Métodos para extraer de un XML[5] la información de los eventos, transforma los datos con XSL[6] y serializa la información usando XML.

[4] **.NET** es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

[5] Metalenguaje extensible de etiquetas que permite definir la gramática de lenguajes específicos.

2. Usa nombres de formatos que pueden ser usados al leer el formato de un evento; el nombre se configura en el fichero de configuración de la aplicación o en otro fichero Web de configuración.

3. Usa registro de niveles de la aplicación, registra el nivel en la aplicación del evento que se disparó. Este valor es configurado dentro de los ficheros de configuración de aplicación.

4. Registro público de los niveles de eventos. El registro de los niveles cambia cuando el evento lo ejecuta la aplicación. Este valor es compilado dentro del código y por lo tanto no puede ser cambiado en tiempos de corrida de la aplicación.

5. Utiliza un tipo de evento para publicar información pedida. El tipo de evento tiene solamente una propiedad para llevar el cuerpo del mensaje, la respuesta del servicio web. Puede tener más propiedades si se determina.

6. Una forma sencilla y coherente de información de registro de eventos.

7. Distribución de información a múltiples fuentes.

8. Marca el inicio y el final de una actividad como un caso de uso.

9. Configuración simplificada del bloque de aplicación que utiliza la configuración de la consola.

10. Extensibilidad de seguimiento personalizado.

Desventajas

1. Basado totalmente en la tecnología .Net de Microsoft y por lo tanto no es soportado por ningún software de código libre.

2. Su utilización y construcción debe ser por especialistas informáticos ya que es muy compleja.

Herramientas para persistir lo eventos

- Microsoft SQL Server 2000 para el receptor de sucesos del servidor SQL.

[6] Lenguaje extensible de hojas de estilo: familia de lenguajes basados en el estándar XML que permite describir cómo la información contenida en un documento XML cualquiera debe ser transformada o formateada para su presentación en un medio.

- Windows Management Instrumentation (WMI[7])
- Kit de desarrollo de software (SDK) para WMI[8]. (5)

1.1.2 Trac

Consiste en un sistema de código abierto, simple, ligero, extensible y basado en la web para la gestión de proyectos y seguimiento de errores. Desarrollado en Python[9], con fuerte dependencia de Subversion. Consiste en un wiki[10], una visión temporal (Timeline) e hitos (Roadmap) del proyecto, un cliente para acceder al código en Subversion, un asignador de tareas y un completo sistema de búsquedas que localiza información en todos los recursos anteriores. (6)

Trac hace de mediador para el enlace entre una base de datos de errores de software, un sistema de control de versiones y el contenido de una wiki. Además puede ser usado como interfaz web para ciertos sistemas de control de versiones, tales como Subversion y Git. 0.11.6. (6)

En cuanto a control y registro de trazas Trac brinda diferentes funcionalidades que facilitan la realización de los mismos, por ejemplo, TracLogging, TracTimeLine, TracRevisionLog y TracReports.

- **TracLogging:** Permite ir registrando el Tracking clasificados como se describe a continuación, errores críticos, errores normales, advertencias, información del diagnóstico y registro sobre todo el proceso, y mensajes de rastro de errores eliminados.
- **TracTimeLine:** Proporciona una vista histórica del proyecto en un solo informe. Enumera todos los acontecimientos que han ocurrido en orden cronológico, proporcionando una breve descripción de cada acontecimiento y si fuera aplicable, la persona de Trac responsable del cambio.

[7] Instrumental de administración de Windows: implementación de WBEM (Administración de empresas basado en la Web) de Microsoft, una iniciativa que pretende establecer normas estándar para tener acceso y compartir la información de administración a través de la red de una empresa.

[8] Conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de software, frameworks, entre otros.

[9] Lenguaje de programación de alto nivel, multiparadigma ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

[10] Sitio web cuyas páginas web pueden ser editadas por múltiples voluntarios a través del navegador web.

- **TracReports:** Permite explotar la información de los Tickets, se genera a través de sentencias SQL y la información que se puede obtener es la siguiente: tiempo, tiempo de cambio, componente, prioridad, propietario, reportero, versión, estado, resolución, resumen, descripción, entre otros aspectos. (7)

Trac permite hacer un seguimiento de los errores detectados y sus correcciones, además puede ser utilizado como sistema de seguimiento de tareas a corta duración. Debe controlar cada incidencia ocurrida en el desarrollo de un producto de software. La aplicación se apoya 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. Otras ventajas del uso de esta herramienta es la posibilidad de habilitar el acceso a un cliente que esté interesado en seguir el desarrollo. Puede consultar el historial para comprobar los avances, revisar/proponer objetivos para los hitos o informar de bugs[11] de aplicaciones en producción. (8)

Además contiene plugins [12] para el trabajo con aplicaciones de integración continua, como CruiseControl, Continuum y Hudson.

1.1.3 Open Populi y OP Framework

Open Populi es una empresa de consultoría en el desarrollo e implantación de aplicaciones informáticas Open Source[13], que cuenta con una Plataforma de Desarrollo propia. (9)

Open Populi Application Framework (OP_Framework) es un marco de desarrollo de software de gestión desarrollada en lenguaje PHP 5 bajo licencia GPL[14] que funciona en entorno web y ha sido diseñado para cubrir las necesidades existentes en el campo de la Administración Electrónica. (10)

OP_Framework intenta partir de las necesidades de los clientes finales, para lograr en conjunto una plataforma robusta, segura y fácil de implementar. Está concebido para trabajar a través de la web, es

[11] Un defecto de software, es el resultado de un fallo o deficiencia en el mismo.

[12] Conjunto de componentes de software que agrega capacidades específicas a una aplicación de software mayor.

[13] Término con el que se conoce al software distribuido y desarrollado libremente.

[14] La Licencia Pública General (*inglés: General Public License o GPL*) otorga al usuario la libertad de compartir el software licenciado bajo ella, así como realizar cambios en él.

decir todas las funcionalidades del sistema están dadas mediante un navegador convencional, por lo que no necesita la instalación de ningún software para su funcionamiento.

La empresa distribuye este producto bajo una licencia de Software Libre. Los sistemas básicos en los que se apoyan las aplicaciones (servidor web, lenguajes de desarrollo, base de datos, entre otros) permiten su implantación de forma independiente del sistema operativo y plataforma del servidor utilizado. (10)

OP_Framework está pensado para el desarrollo de aplicaciones web en múltiples idiomas, con uso intensivo de base de datos, autenticación de usuarios, encriptación y requerimientos avanzados en lo referente a la auditoria del sistema. (10)

Todo Framework brinda diferentes herramientas y funcionalidades para facilitar el trabajo del programador, las de OP Framework son:

1. Seguridad
2. Trazas y Auditoría
3. Debugging
4. Routing
5. Sesiones de Usuario
6. Helpers

OP_Framework dispone de un sistema de registro de eventos o trazas (Logs). Se dividen en dos tipos, los de seguimiento de la aplicación y los de auditoría. Entre los primeros se pueden encontrar:

Logs de Plataforma

Registran eventos y errores propios de la plataforma y sus aplicaciones. El registro se realiza en tres salidas diferentes. En un archivo de texto, en formato "FireBug" para FireFox y un Syslog. (11)

Logs en archivos de texto

La salida en archivo de texto se guarda en *OP_Framework/***miplataforma***/log/***miplataforma***.log,* (11)es por eso que crea un nuevo archivo para cada una de las plataformas. Un log en archivo de texto quedaría de la siguiente manera:

*OP_Framework/***miplataforma***/log/***miplataforma***.log*

```
//...
2008 07 14 10:47:16 OP [warning] {ayuntamiento} Open Populi logea en Firebug.
2008 07 14 10:47:16 OP [info] {ayuntamiento} APPLICATION: , MODULE: , ACTION:
REDIRECTS TO:
http://intranet.openpopuli.com/ayuntamiento/backend/cliente
2008 07 14 10:47:16 OP [info] {op_view} inicialización de la vista con el
layout "ayuntamiento.tpl.php"
2008 07 14 10:47:16 OP [info] {op_view} render template "index.tpl.php"
2008 07 14 10:47:16 OP [info] {op_view} renderiza en el navegador
//... (10)
```

En el ejemplo se muestra como cada evento está dado en una nueva línea en el texto, y cada línea posee fecha, hora, tipo de evento, el objeto procesado, plantillas procesadas, llamadas realizadas entre objetos, entre otros detalles notables que dependen del tipo de evento en cuestión.

Estos archivos log también dotan a los eventos de una prioridad (el elemento que aparece entre corchetes “[]”), esta se divide en 8 niveles.

EMERG: El sistema está inutilizable.

ALERT: Requiere una acción inmediata.

CRIT: Condiciones críticas.

ERR: Condiciones de Error.

WARNING: Condiciones de Warning.

NOTICE: Normal pero significativa.

INFO: Informativo.

DEBUG: Mensajes a Nivel Debug.

Logs de FireBug

Los eventos también pueden ser registrados en FireBug de Firefox, extensión creada para analizar, editar, monitorizar y depurar el código fuente, CSS, HTML y Java Script de una página web de manera instantánea y online, es útil para ver las líneas de log en tiempo real, con la aplicación en marcha.

Logs en SysLog

Los mensajes pueden ser enviados además por SysLog. Sólo se necesita un ordenador servidor ejecutando el servidor de Syslog (demonio de syslog), y la aplicación le enviará el mensaje de texto (de menos de 1024 bytes) a dicho servidor a través del puerto 514.

Los datos de configuración del servidor externo de Syslog se establecen en el archivo OP_Framework/**miplataforma**./config/config.ini.php editando las siguientes líneas.

```
//...
[syslog]
host = 192.168.10.2;
port = 514;
//... (11)
```

El segundo grupo solamente está compuesto por los:

Logs de Auditoría

En este tipo de log se registra cada evento relacionado con la Base de Datos, todas las consultas SQL que transformen el contenido de esta. Los logs se guardan en OP_Framework/miplataforma/logs/miplataforma.audit.log. Para posteriormente ser auditados por personal calificado para la realización de la actividad.

1.2 Control y registro de trazas a nivel nacional

En el país el control de las trazas por lo general se realiza de forma manual, en la mayoría de los casos, cada empresa revisa las trazas generadas por aplicaciones como el correo electrónico y servicios de mensajería, así como las visitas a internet.

1.2.1 ACINOX Comercial

Un ejemplo de ello es la empresa ACINOX Comercial, con sede en La Ciudad de la Habana, que brinda un servicio de red WAN a sus 13 sucursales en todo el país, es por eso que hay trazas que se almacenan en la Casa Matriz, y otras en las sucursales correspondientes, debido a los principios de territorialidad de las resoluciones de Seguridad Informática. Aun así, como cada sucursal tiene que utilizar los mismos proxies[15], en un análisis forense, desde la Casa Matriz se puede encontrar casi cualquier cosa.

Dicha empresa para el correo cuenta con un servidor externo que es básicamente un Gateway[16] para todos los servidores internos de correo en toda Cuba, o sea, los que no están en direcciones reales. Hasta ahora todos los servidores internos de correo son MDaemon/Windows.

En LINUX como "gateway externo" utilizan sendmail como mailer/router y solo almacenan las trazas de comunicación de los tres últimos meses. Estas trazas son de utilidad solamente para comprobar y/o arreglar problemas de comunicación entre sus servidores y cualquier otro servidor.

En los MDaemon/Windows cuando ha sido necesario algo más "complejo" que lo que ya se ha explicado, han descargado e instalado el SABCOR de Segurmática. También han hecho bases de datos en Microsoft Access en alguna ocasión, para poder hacer "consultas" detalladas. Estas trazas se mantienen de igual forma por tres meses.

Ahora bien, en los MDaemon/Windows lo principal es que almacenan copia de todos y cada uno de los correos que entran o salen, y estos no son borrados hasta pasados los seis meses. Son decenas de gigabytes de información.

En cuanto al servicio de mensajería, "descargan" periódicamente todos los mensajes de todos y cada uno de los usuarios. Lo que se hace con ellos es analizarlos cuando se les ha solicitado "subiéndolos" a una base de datos MySQL "local" y entonces de ahí ejecutan las "consultas". Es por eso que almacenan todos los mensajes, y cada copia de seguridad es mayor que el anterior.

[15] En una red informática, es un programa o dispositivo que realiza una acción en representación de otro.

[16] Puerta de enlace: dispositivo, con frecuencia una computadora, que permite interconectar redes con protocolos y arquitecturas diferentes a todos los niveles de comunicación.

Para la navegación en Internet cuentan con dos servidores LINUX/SQUID con reglas de seguridad para que solo puedan conectarse PCs "internas" de la WAN. Un SQUID tiene habilitado dar servicio de navegación internacional, y el otro nacional. En el ACTIVE DIRECTORY tienen las políticas del mismo y los derechos de navegación que tiene cada usuario. Lo que crea dos conjuntos de trazas, por la existencia de los dos proxies en línea.

De igual manera las trazas son almacenas por tres meses. Cuando tienen la necesidad o se les solicita, utilizan el AAinternet de Segurmática. En ocasiones toda esta gestión se realiza a mano. También utilizan el propio "Squid Analysis Report Generator" para comprobaciones ad-hoc[17].

1.3 Herramientas de registro de trazas en la UCI

1.3.1 SRNI

El Sistema de Reportes de la Navegación por Internet (SRNI) proporciona reportes dinámicos de la navegación de los usuarios a través de internet. Crea reportes a partir de las trazas del servidor proxy y los usuarios pueden consultar la información de accesos por las siguientes formas:

- Acceso por URL
- Acceso por dirección IP
- Acceso por días
- Acceso por horas

Además el sistema tiene la característica de mostrar el consumo equivalente al tamaño de los recursos que fueron accedidos mediante la WEB. También ofrece la posibilidad de conocer el nuevo sistema de consumo que se calcula en correspondencia con los sitios a los que se accede y la hora en la que se realizó este acceso.

[17] Formas de acceder a la base de datos, el sistema permite al usuario personalizar una consulta en tiempo real, en vez de estar atado a las consultas prediseñadas para informes.

1.4 Gestión de Trazas en los Sistemas ERP[18]

Actualmente, en el mundo se ha puesto en práctica el uso de los sistemas ERP, los cuales han permitido el auge y el completo desarrollo de numerosas empresas. Muchos de ellos cuentan con herramientas destinadas para la gestión de trazas. A continuación se muestran algunos de los sistemas.

1.4.1 SAP ERP

El sistema SAP ERP contiene varias herramientas que permiten visualizar información detallada sobre los servidores de aplicaciones, las sesiones de usuario y procesos de trabajo. La herramienta de seguimiento (o de trazas) es una de ellas, la cual permite rastrear los errores, configurar y evaluar las trazas en distintos niveles. La herramienta permite utilizar las funciones de la traza para seguir el proceso de varias operaciones en el sistema. Lo que posibilita controlar el sistema y aislar los problemas que se producen. (12)

Características

Los siguientes componentes pueden ser controlados mediante la herramienta de seguimiento del sistema SAP:

- Autorización de los controles.
- Funciones del kernel.
- Módulos del kernel.
- Acceso a Bases de Datos (traza de SQL).
- Tabla de buffer.
- Llamadas RFC (Request for Comments por sus siglas en inglés).
- Bloqueo de las operaciones (del lado del cliente).

[18] (*Enterprise Resources Planning*, por sus siglas en inglés) es un sistema basado en la aplicación integrada que se utiliza para gestionar los recursos internos y externos de una empresa.

El sistema SAP registra todos los errores del sistema, las advertencias, los usuarios bloqueados debido a los intentos de inicio de sesión fallidos, y procesa los mensajes en el registro del sistema. Hay dos tipos diferentes de registros creados por el registro del sistema:

Registros Locales

Cada servidor de la aplicación del sistema SAP tiene un registro local que recibe todos los mensajes salientes de este servidor. System Log registra estos mensajes en un archivo circular en el servidor.

Registros Centrales

El registro central consiste en dos archivos: el archivo activo y el archivo antiguo. Cada servidor de aplicaciones individuales envía sus mensajes de registro locales a un servidor de aplicaciones designado. El servidor que se designa para mantener el registro central recoge los mensajes de los otros servidores de aplicaciones y los escribe en el registro central. El archivo activo contiene el registro actual. Cuando se alcanza el tamaño máximo, el sistema realiza un "cambio de archivo de registro". Elimina el archivo de registro antiguo y crea un nuevo archivo activo. (13)

1.4.2 Software.AG

Visor de registros de Software.AG

El sistema ERP Software.AG posee una herramienta llamada Visor de registros o “The Log Viewer”, por sus siglas en inglés. Es una herramienta independiente diseñada para ver los archivos individuales o múltiples de las trazas generadas. Las actividades y operaciones realizadas sobre un ordenador o servidor son archivadas en un registro de trazas manteniendo el historial de estas actividades por un tiempo prolongado. Estos archivos de registro se identifican por la extensión *.log*. (14)

El Visor de registros brinda la posibilidad al usuario de:

- Especificar intervalos de actualización.
- Especificar el número de líneas que se muestren de un archivo.
- Personalizar la manera de ver los archivos mediante la especificación de las condiciones del filtro.

- Guardar los archivos de registro agregado en el visor de forma que el usuario es capaz de acceder a ellos fácilmente la próxima vez.
- Proporcionar puntos de extensión a otros plugins Eclipse para contribuir con sus archivos de registro para su visualización.

1.5 Manejo de Trazas en PHP y Symfony

Para comprender lo sucedido cuando falla la ejecución de una petición es necesario echar un vistazo a la traza generada por el proceso que se ejecuta. Afortunadamente, tanto PHP como Symfony guardan mucha información de este tipo en archivos de log. A continuación se explicará cómo funcionan las trazas o Logs de PHP y Symfony respectivamente.

1.5.1 Trazas de PHP

PHP dispone de una directiva llamada *error_reporting*, que se define en el archivo de configuración *php.ini*, y que especifica los eventos de PHP que se guardan en el archivo de log. Symfony permite redefinir el valor de esta opción, tanto a nivel de aplicación como de entorno, en el archivo *settings.yml.* (15)

Para no penalizar el rendimiento de la aplicación en el entorno de producción, el servidor solamente guarda en el archivo de log los errores críticos de PHP. No obstante, en el entorno de desarrollo, se guardan en el log todos los tipos de eventos, de forma que el programador puede disponer de la máxima información para seguir la pista a los errores.

El lugar en el que se guardan los archivos de log de PHP depende de la configuración del archivo *php.ini.* Si no se ha modificado su valor, PHP utiliza las herramientas de log del servidor web (como por ejemplo los logs de error del servidor Apache). En este caso, los archivos de log de PHP se encuentran en el directorio de logs del servidor web.

1.5.2 Trazas de Symfony

Además de los archivos de log creados por PHP, Symfony también guarda mucha información de sus propios eventos en otros archivos de log. Los archivos de log creados por Symfony se encuentran en el directorio *miproyecto/log/*. Symfony crea un archivo por cada aplicación y cada entorno. El archivo

del entorno de desarrollo de una aplicación llamada *frontend* sería *frontend_dev.log* y el archivo de log del entorno de producción de la misma aplicación se llamaría *frontend_prod.log.*

Si se dispone de una aplicación Symfony ejecutándose, se puede observar que la sintaxis de los archivos de log generados es muy sencilla. Cada evento resulta en una nueva línea en el archivo de log de la aplicación. Cada línea incluye la fecha y hora a la que se ha producido, el tipo de evento, el objeto que ha sido procesado y otros detalles relevantes que dependen de cada tipo de evento y/o objeto procesado. (16) A continuación se muestra un ejemplo de un archivo de Symfony que contiene los Logs:

```
//...
Nov 15 16:30:25 symfony [info ] {sfAction} call "barActions->executemessages()"
Nov 15 16:30:25 symfony [info ] {sfPropelLogger} executeQuery: SELECT
bd_message.ID...
Nov 15 16:30:25 symfony [info ] {sfView} set slot "leftbar" (bar/index)
Nov 15 16:30:25 symfony [info ] {sfView} set slot "messageblock" (bar/mes...
Nov 15 16:30:25 symfony [info ] {sfView} execute view for template "messa...
Nov 15 16:30:25 symfony [info ] {sfView} render "/home/production/myproject/...
Nov 15 16:30:25 symfony [info ] {sfView} render to client
...//
```

Estos archivos de log contienen mucha información, como por ejemplo las consultas SQL enviadas a la base de datos, las plantillas que se han procesado, las llamadas realizadas entre objetos, entre otras.

Configuración del nivel de log de Symfony

Symfony define ocho niveles diferentes para los mensajes de log: *emerg, alert, crit, err, warning, notice, info, debug.* En el archivo de configuración *factories.yml* de cada aplicación se define el nivel de los mensajes que se guardan en el archivo de log.

Por defecto, en todos los entornos salvo en el de producción, se guardan en los archivos de log todos los mensajes (hasta el nivel menos importante, el nivel debug). En caso de que se activen los logs, asignándole el valor *on* a la opción *logging_enabled* en el archivo de configuración *settings.yml*, solamente se guardan los mensajes más importantes (de crit a emerg). (16)

Metodología, herramientas y lenguajes de modelado destinados para el desarrollo de software.

1.6 Metodología de desarrollo del software

1.6.1 RUP

RUP (Proceso Racional Unificado) es un proceso de desarrollo de software. Define flujos de trabajo, donde en cada uno de ellos se especifican las actividades a realizar por el equipo de desarrollo, así como los artefactos que se deben desarrollar. En la siguiente figura se pueden apreciar las fases y flujos de esta metodología.

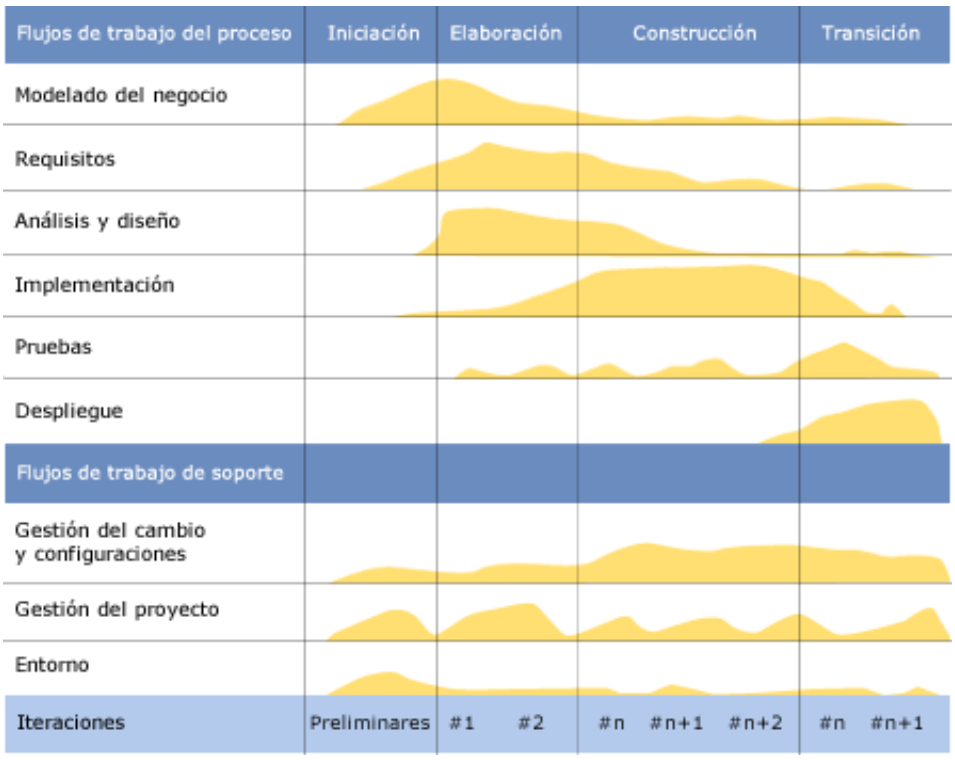

Fig. 1.1. Fases y flujos de trabajo de la metodología RUP.

Es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de

sistemas orientados a objetos. Esta metodología representa una guía que establece *quién* hace *qué*, *cuándo* y *cómo* lo hace.

El RUP está basado en 5 principios claves que son:

- Adaptar el proceso.
- Balancear prioridades.
- Demostrar valor iterativamente.
- Elevar el nivel de abstracción.
- Enfocarse en la calidad.

Sus características principales son:

- ***Dirigido por casos de uso:*** los casos de uso guían el proceso de desarrollo pues los modelos que se obtienen representan la realización de los mismos.
- ***Centrado en la arquitectura:*** la arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo.
- ***Iterativo e incremental:*** una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. En el caso de una iteración de la fase de elaboración, se centra la atención en el análisis y diseño, a la vez que se refinan los requerimientos y se obtiene un producto con un determinado nivel, que irá creciendo incrementalmente en cada iteración. (17)

1.6.2 Modelo de desarrollo de software orientado a componentes

Todo desarrollo de software es riesgoso y difícil de controlar, es por eso que se debe tener un especial cuidado a la hora de seleccionar una metodología adecuada que cumpla y satisfaga todas las expectativas del cliente y desarrolladores por igual. Es por ello que debido a las particularidades inherentes al desarrollo de software llevado a cabo en el Centro de Informatización y Gestión de Entidades (CEIGE), se asumió el Modelo de Desarrollo orientado a componentes, que facilita la producción de software, siguiendo otras directrices durante el proceso.

Dicho modelo, fue creado inicialmente para ser aplicado al ERP cubano CEDRUX, el cual por ser un proyecto que por su complejidad en los procesos de negocio, tamaño, integración entre los módulos

que lo conforman y además por el cúmulo de información que almacena y analiza, requiere de un largo período de elaboración, construcción y coordinación entre los equipos de desarrollo. Sin embargo después del proceso de fusión en la producción, el centro CEIGE lo adaptó para algunos de los proyectos que desarrollan en el centro.

El desarrollo de cualquier tipo de proyecto consiste en una serie de fases, muchas veces secuenciales conocidas como el ciclo de vida del proyecto y generalmente definen: qué trabajo técnico se hará en cada fase, cuándo se generarán los entregables, cómo se revisarán, verificarán y validarán en estas, quién está involucrado en cada fase y cómo controlar y aprobar cada una. Las principales características del ciclo de vida expuesto son:

- Las fases son secuenciales y su transferencia debe ser precedida por un proceso de revisión o liberación del Centro de Calidad y su aprobación en Consejo Técnico Formal.
- El nivel del personal es bajo al comienzo, alcanza su nivel máximo en la fase de construcción y decae rápidamente cuando el proyecto se aproxima a su conclusión.
- La participación de los interesados es alta en las etapas de Inicio y Modelación, baja en la etapa de Construcción y vuelve a subir en las etapas finales del proyecto.

Las fases que propone el modelo son cinco: Inicio, Modelación, Construcción, Explotación Experimental y Despliegue; de las que a continuación se muestra un resumen:

- ***Fase de Inicio:*** se logra una visión de la problemática a resolver, se identifica el alcance preliminar del proyecto, se especifican los involucrados y se establece la estrategia a seguir para realizar la captura de requisitos funcionales.
- ***Fase de Modelación:*** se definen los procesos del negocio, se identifican las necesidades del usuario de las que se derivan los requisitos del producto a desarrollar, se identifican las reglas del negocio, entre otros.
- ***Fase de Construcción:*** se deben aclarar los requisitos restantes y completar el desarrollo del sistema sobre una base estable de la arquitectura. En esta fase todas las características, componentes y requerimientos deben ser integrados, implementados y probados en su totalidad, obteniendo una versión estable del producto comúnmente llamada versión beta. Se realiza la planificación detallada de la siguiente fase.
- ***Fase de Explotación Experimental*****:** tiene como propósito asegurar que el software está disponible para realizar las pruebas de aceptación por un grupo de usuarios. Incluye las

pruebas del producto como parte de su preparación para ser entregado, y la realización de ajustes en respuesta a la retroalimentación recibida de los usuarios. En este punto del ciclo de vida la retroalimentación de los usuarios debe enfocarse fundamentalmente en ajustes específicos y de corto alcance al producto junto a otros temas como configuración, instalación, y usabilidad.

- ***Fase de Despliegue:*** se realiza la generalización del producto en las entidades y órganos según lo aprobado en el Cronograma de implantación. Durante el proceso de implantación por lo general no es necesaria la participación de los integrantes del equipo de desarrollo. (18)

1.7 Herramientas CASE

Las herramientas CASE[19] (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son el mejor soporte para el proceso de desarrollo de software. Son 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. (19)

La realización de un software requiere que las tareas sean organizadas y completadas de forma eficiente. Las herramientas CASE han sido desarrolladas para automatizar esos procesos, incrementando la velocidad de desarrollo de los sistemas.

1.7.1 Visual Paradigm para UML

Es una herramienta CASE multiplataforma de modelado visual UML, muy potente y fácil de utilizar. Soporta el ciclo de vida completo del desarrollo de software, 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.

- Soporte de UML versión 2.1.

[19] Aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Ingeniería inversa - Código a modelo, código a diagrama.
- Generación de código - Modelo a código, diagrama a código.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Poderosa herramienta de generación de PDF/HTML a partir de diagramas UML.
- Sincronización entre el código fuente y el modelo en tiempo real. (20)

Entre sus características principales está que permite el control de versiones, consta de un entorno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso. También permite el despliegue de EJB (Enterprise java Beans), la construcción de diagramas de flujo de datos, la generación de objetos Java desde la base de datos, la transformación de diagramas de Entidad-Relación en tablas de base de datos. Posee un generador de informes, la reorganización de las figuras y conectores de los diagramas y un editor de figuras. (21)

1.7.2 Rational Rose

Rational Rose es una herramienta software para el Modelado Visual mediante UML de sistemas software. Permite especificar, analizar y diseñar el sistema antes de codificarlo. Rational Rose es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases.

Características de Rational Rose:

- Mantiene la consistencia de los modelos del sistema software.
- Chequeo de la sintaxis UML.

- Generación de documentación automáticamente.
- Generación de código a partir de los modelos.
- Ingeniería Inversa (crear modelo a partir código). (22)

1.7.3 Enterprise Architect

Agilian
Necessity for Agile Modeling

Es una herramienta de construcción y modelado de software de alto rendimiento basado en el estándar de UML 2.1, propiedad de la empresa SPARX SYSTEMS[20]. Con una trazabilidad completa desde los requisitos iniciales hasta las decisiones de diseño de software; provee el tipo de visualización y colaboración eficiente y robusta requerida en los entornos de desarrollo de software que actualmente son altamente demandantes. Como una solución de modelado verdaderamente ágil, Enterprise Architect provee una sobrecarga de instalación baja, un rendimiento brillante y una interfaz intuitiva (incluyendo una versión de "sólo lectura"). Permite la integración con los IDE[21] Visual Studio y Eclipse, para el trabajo en equipo se integra con la herramienta de integración Teamcenter Systems Engineering[22], Posee un módulo que permite importar proyectos realizados en Vicio, además de un módulo que amplía las capacidades de la herramienta permitiendo incluso soportar la ingeniería de código directa e inversa para el lenguaje Python. Enterprise Architect es una herramienta para ayudar en su trabajo a prácticamente todo el equipo de trabajo, desde el analista hasta el equipo de despliegue, genera documentación compatible con MS Word. Soporte para ActionScript 2.0, Java, C#, C++, VB.Net, Delphi, Visual Basic, Python y PHP, permite importar y exportar XML. Aunque es un

[20] Empresa consolidada para la provisión de servicios y productos de Tecnologías de Información y Comunicaciones, tanto para equipos de desarrollo de software como para clientes en general.

[21] programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

[22] Potente herramienta para crear e integrar los requisitos. Esto permite que se adhiera perfectamente a los procesos del ciclo de vida y necesidades de información que existen dentro de una organización.

software bajo licencia comercial (propietaria), sus licencias se consideran de bajo coste, además brinda la posibilidad de trabajar con una versión de prueba por 30 días. (23)

1.8 Lenguajes de Modelado

1.8.1 UML (Unified Modeling Language)

Es un lenguaje que permite visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. (24) UML es un conjunto de herramientas, que permite modelar (analizar y diseñar) sistemas orientados a objetos. Se ha convertido en el estándar de facto de la industria. UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos. 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. UML cuenta con varios tipos de diagramas que son la representación gráfica de un conjunto de elementos y sus relaciones que visualizan el sistema desde diferentes perspectivas. (25)

1.8.2 BPMN (Business Process Management Notation)

Es una notación que modela los procesos de negocio, basada en diagramas de flujo fácil de entender. Está diseñada para cubrir varios tipos de modelado, permite la creación tanto de segmentos de procesos, como procesos de negocio de comienzo a fin en diferentes niveles de representatividad. BPMN es gráficamente más rico, con menos símbolos fundamentales, pero con más variaciones de éstos, lo que facilita su comprensión por parte de personas no expertas. (23)

Con la utilización de la mencionada notación se comprende rápidamente todo el negocio, desde el analista de negocio que hace el borrador inicial hasta los desarrolladores, responsables de implementar la tecnología que llevarán a cabo dichos procesos, para llegar finalmente a los clientes

que son los que gestionarán y monitorizarán esos procesos. El modelado en BPMN se realiza mediante diagramas muy simples con un conjunto muy pequeño de elementos gráficos. Las cuatro categorías básicas de elementos son:

- ***Objetos de flujo:*** Eventos, Actividades, Rombos de control de flujo (Gateway).
- ***Objetos de conexión:*** Flujo de Secuencia, Flujo de Mensaje, Asociación.
- ***Carriles de piscina (Swimlanes):*** Pool, Lane.
- ***Artefactos:*** Objetos de Datos, Grupo, Anotación.

Estas cuatro categorías de elementos dan la oportunidad de realizar un diagrama simple de procesos de negocio (en inglés Business Process Diagram o BPD). En un BPD se permite definir un tipo personalizado de Objeto de Flujo o un Artefacto, si con ello se hace el diagrama más comprensible. Otra ventaja de utilizar BPMN para los modeladores y las herramientas de modelado es que muestra flexibilidad a la hora de extender la notación básica. La más importante de todas es que este lenguaje de modelado toma un enfoque orientado al proceso, esto posibilita modelar tal y como suceden estos, en la empresa.

1.9 Lenguaje de programación

1.9.1 PHP

PHP (PHP Hypertext Pre-processor por sus siglas en inglés) es un lenguaje de programación interpretado, usado normalmente para la creación de páginas web dinámicas, además permite la conexión a diferentes tipos de servidores de base de datos, es multiplataforma, libre, por lo que se presenta como una alternativa de fácil acceso para todos y permite las técnicas de Programación Orientada a Objetos.

La versión utilizada de PHP es la 5.2.5 que incluye varias ventajas:

- Mejoras de rendimiento.

- Mejor soporte para MySQL con extensión completamente reescrita.
- Mejor soporte a XML (XPath, DOM, entre otros.).
- Soporte nativo para SQLite.
- Soporte integrado para SOAP[23].
- Iteradores de datos.
- Manejo de excepciones. (26)

1.10 Marco de Trabajo (Framework)

1.10.1 Symfony

Symfony es un framework desarrollado en lenguaje PHP, estable, productivo y muy bien documentado, muy difundido en la actualidad para la construcción de aplicaciones web, utilizando las mejores prácticas y los patrones de diseño más importantes. Incorpora muchas de las ideas del RAD (Desarrollo Rápido de Aplicaciones) para conseguir que la programación de las aplicaciones sea lo más productiva y correcta posible. Es multiplataforma y se integra con varios sistemas gestores de bases de datos, lo que le brinda una versatilidad muy considerable, al poder migrar la base de datos sin tener que realizar ningún otro cambio en la aplicación, más que editar un archivo de configuración. El mismo hace uso del patrón de diseño MVC (Modelo-Vista-Controlador) para un mejor control de las partes de la aplicación y las mantiene por separado.

Entre sus principales características se encuentran:

- Fácil de instalar y configurar en la mayoría de las plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix[24] estándares).

[23] Objeto de protocolo simple de acceso: protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de *"convenir en vez de configurar"*, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales, y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor [25] y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros. (27)

1.11 Gestor de Base de Datos

1.11.1 Oracle 11g

Es un potente Sistema Gestor de Base de Datos (SGBD), multiplataforma, que se caracteriza por haber sido concebido con el fin de manejar grandes cantidades de información, además de admitir conexiones concurrentes de multitud de usuarios (entornos multi-usuario) hacia los mismos datos.

Las principales funcionalidades aportadas por todo el SGBD Oracle son:

- Soporte y tratamiento de una gran cantidad de datos (Gbytes)
- Soporte de una gran cantidad de usuarios accediendo concurrentemente a los datos.

[24] Unix, Linux, Solaris, entre otros.

[25] Generador de documentación de código abierto escrito en PHP.

- Seguridad de acceso a los datos, restringiendo dicho acceso a las necesidades de cada usuario.
- Integridad referencial en su estructura de base de datos.
- Conectividad entre las aplicaciones de los clientes en sus puestos de trabajo y el servidor de base de datos Oracle (estructura cliente/servidor).
- Conectividad entre bases de datos remotas (estructura de bases de datos distribuidas).
- Portabilidad.
- Compatibilidad.

Conclusiones del capítulo

Una vez realizado el presente capítulo se obtuvo como resultado un análisis, mediante el estado del arte, de las principales soluciones relacionadas con el registro y control de las trazas, la realización lenta y manual, en ocasiones, de la gestión de estas a distintos niveles (nacional, internacional y en la universidad) y la carencia de un mecanismo que humanice su funcionamiento con el uso del sistema GINA.

Symfony, el marco de trabajo en el que está desarrollado el Sistema GINA, presenta varias herramientas que permiten administrar las aplicaciones, dentro de las que se encuentran los Logs. Dicha herramienta presenta varias funcionalidades que permiten la gestión de trazas, pero debido a la estructura de la AGR se hace necesario el desarrollo de una aplicación más completa y flexible que permita gestionar y controlar todas las acciones que se ejecuten sobre los subsistemas que componen el Sistema GINA.

Se logró conocer entonces, una vez concluido el estudio del estado del arte, que ninguna de las herramientas estudiadas anteriormente satisface cabalmente los requisitos que exige la nueva aplicación, ya que sus funcionalidades se centran en el registro de errores internos de los sistemas, además de estar desarrollados bajo la arquitectura de otros Framework, lo que obstruye el uso de las mismas. Como ninguna de las aplicaciones mencionadas anteriormente cumple con las necesidades especificadas, se propone el desarrollo del presente trabajo de diploma.

Capítulo 2: Análisis y Diseño de la propuesta de solución.

Introducción

En el presente capítulo se hace un levantamiento de los requisitos funcionales del producto a desarrollarse, se realiza además la descripción de los artefactos que son generados durante el flujo de trabajo del análisis y el diseño.

2.1 Propuesta de solución

En la AGR se ha llevado a cabo la implantación de varios sistemas informáticos y la construcción de otros por parte de los especialistas del CADI; los cuales han permitido automatizar gran parte de los procedimientos aduaneros que se realizan en dicha institución. A pesar de realizar numerosas funcionalidades, estos sistemas no cumplen con las nuevas exigencias de la Aduana, por lo que se requiere de la producción de nuevos productos.

Debido a lo anteriormente planteado se lleva a cabo el desarrollo del sistema GINA, que además de automatizar los procesos aduaneros, permitirá garantizar la seguridad e integridad del mismo mediante la utilización del componente de seguridad suAdminPlugin perteneciente al subsistema de Administración, el cual aporta un mayor control de las acciones que pueden realizar los usuarios sobre los subsistemas que componen a GINA.

El suAdminPlugin fue desarrollado en el Departamento de Soluciones para la Aduana, el mismo, actualmente, carece de una funcionalidad que permita gestionar y controlar las trazas que son generadas por los demás subsistemas de GINA, para que, posteriormente, sean auditadas por el personal indicado para ello.

Producto a que todos los procedimientos aduaneros son de gran importancia para la AGR, es necesario controlar y llevar registros rigurosos de las acciones que son realizadas sobre los subsistemas. Para darle solución al problema que ha sido planteado, se propone el análisis, diseño e implementación de un componente que se encontrará ubicado dentro del suAdminPlugin; componente que estará estructurado tal y como lo plantea la arquitectura de Symfony. El mismo estará conformado por una estructura de directorio que cumple con las especificaciones requeridas por el framework.

El componente a desarrollar posibilitará registrar todas las trazas que sean generadas por los subsistemas que integran a GINA; estas trazas estarán conformadas por los siguientes datos: (operación que se realizó, usuario que la generó, fecha y hora en la que se realizó, IP[26] desde donde accedió el usuario, objeto y tipo de operación de negocio que fue modificado, el subsistema al que pertenece, y un texto formateado). Dicho componente tendrá la forma de un servicio tal y como lo describe la arquitectura definida en el Departamento de Soluciones para la Aduana, al cual podrán acceder todos los subsistemas del GINA para generar las trazas. Una vez desarrollada la solución se garantizará que no ocurran fraudes en la institución por parte de los usuarios que tienen acceso a estas aplicaciones, contribuyendo además a mantener la integridad y la seguridad de los datos.

Para llevar a cabo el desarrollo de la solución se utilizarán las herramientas, lenguajes y metodología y/o modelo de desarrollo que están definidas en el Departamento de Soluciones para la Aduana, las cuales se detallan a continuación:

Se hizo necesario la utilización de la metodología de desarrollo RUP para la realización de las actividades del procedimiento para la Ingeniería de Requisitos, ya que es una de las más aplicadas en la actualidad, por ser un proceso iterativo e incremental, flexible, que divide el trabajo en fases teniendo bien definidas las tareas a realizar en cada una de ellas. Se le realizó modificaciones a RUP ya que es un proceso de desarrollo de software configurable que puede ser adaptado a las características propias del proyecto. Estas modificaciones se detallan en el Trabajo de Diploma titulado Procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE. (28)

Por su facilidad de uso, el Departamento de Soluciones para la Aduana seleccionó Visual Paradigm por ser una herramienta CASE que facilita el modelado de artefactos en un proceso de desarrollo de software mediante el lenguaje de modelado UML, que soporta el ciclo de vida completo del desarrollo de software. Soporta la ingeniería inversa, generación de código, importación desde Rational Rose, exportación/importación XML, generado de informes, editor de figuras, entre otras funcionalidades. En el trabajo se utilizará la versión 6.4.

[26] Protocolo de Internet: etiqueta numérica que identifica, de manera lógica y jerárquica, a un interfaz (elemento de comunicación/conexión) de un dispositivo (habitualmente una computadora) dentro de una red.

Como lenguaje de modelado se utilizará UML que es el propuesto por la metodología de desarrollo que se utiliza en el proyecto. UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software.

Como SGBD se empleará Oracle, en su versión 11g, ya que es un potente gestor que permite almacenar gran cantidad de datos, además de ser multiplataforma. El lenguaje de programación a utilizar será PHP, en su versión 5.2.7, como IDE de desarrollo se empleará el NetBeans 6.9 y como Framework se utilizará Symfony, pues es el seleccionado por el Departamento de Soluciones para la Aduana por las facilidades que brinda.

2.2 Modelo del Dominio

Un modelo del dominio captura los tipos de objetos más importantes en el contexto del sistema. Los objetos del dominio representan las "cosas" que existen o los eventos que suceden en el entorno en el que trabaja el sistema. El mismo se describe mediante diagramas de UML (especialmente mediante diagramas de clases). Estos diagramas muestran a los clientes, a los usuarios, revisores y a otros desarrolladores las clases del dominio y cómo se relacionan unas con otras mediante asociaciones. (29)

A continuación se muestra el modelo del dominio correspondiente al componente Auditoría del suAdminPlugin, cuyo objetivo principal es identificar y mostrar el comportamiento del negocio a través de conceptos relacionados entre sí que evidencian cómo interactúan los componentes en el registro y administración de trazas.

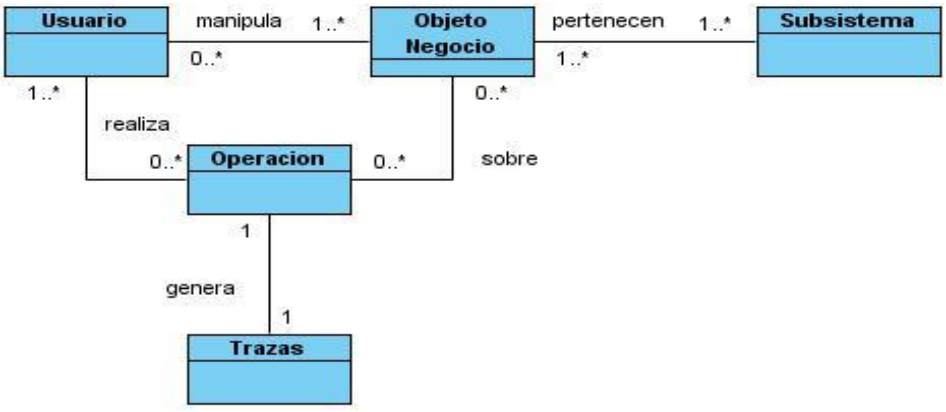

Fig. 2.1. Modelo del Dominio

2.2.1 Definición de las clases del modelo del dominio.

Usuario: Persona que interactúa con los diferentes subsistemas, es el encargado de realizar las diferentes operaciones sobre los objetos de su negocio que serán auditadas.

Objetos negocio: Los objetos con los que interactúa cada subsistema en su negocio, sobre estos se pueden llevar a cabo diferentes operaciones dependiendo del subsistema al que pertenece.

Subsistema: Sistema que trabaja en la automatización de un determinado proceso llevado a cabo en la Aduana para cumplir con un objetivo.

Operación: Acciones que se ejecutan en los subsistemas, estas pueden ser insertar, modificar, crear, eliminar, entre otras propias de cada subsistema.

Traza: Las acciones son guardadas como registros persistentes de las acciones ejecutadas sobre un sistema que incluye el usuario que la generó, el IP desde donde se generó y la fecha.

2.3 Requisitos

Se ha comprobado que los requisitos o requerimientos se han convertido en la pieza fundamental en un proyecto de desarrollo de software, pues marcan el punto de partida para actividades como la planeación, básicamente en lo que se refiere a las estimaciones de tiempos y costos. Además la especificación de los requerimientos es la base que permite verificar si se alcanzaron o no los objetivos establecidos en el proyecto ya que son un reflejo detallado de las necesidades de los clientes o usuarios del sistema.

Un requerimiento es una descripción de una condición o capacidad que debe cumplir un sistema, ya sea derivada de una necesidad de usuario identificada, o bien, estipulada en un contrato, estándar, especificación u otro documento formalmente impuesto al inicio del proceso. (30)

2.3.1 Técnicas para la captura de requisitos

La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema. (31). El proceso de captura de requisitos puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas, y depende mucho de las personas que

participen en él. Por la complejidad que todo esto puede implicar, la ingeniería de requisitos ha desarrollado técnicas que permitan hacer este proceso de una forma más eficiente y precisa.

A continuación se presentan las técnicas que se usaron para la captura de requisitos para darle solución al problema planteado en la presente investigación.

Entrevistas: Resultan una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada.

Tormentas de ideas: Una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas. El grupo de personas que participa en estas reuniones no debe ser muy numeroso (máximo 10 personas), una de ellas debe asumir el rol de moderador de la sesión, pero sin carácter de controlador.

A través de las tormentas de ideas y las entrevistas realizadas a especialistas del CADI y técnicos de la Aduana se identificaron las necesidades que posee actualmente la Aduana con respecto a la gestión y control de las trazas generadas por los subsistemas para su posterior auditoría. La aplicación de estas técnicas permitió definir el requisito funcional del sistema a desarrollar.

Seguidamente se muestran los requisitos funcionales y no funcionales del componente de Auditoría pertenecientes al subsistema de Administración del sistema GINA.

2.3.2 Requisitos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. El sistema está compuesto por un solo módulo, a continuación se hará una delineación del requisito que lo compone.

Requerimiento funcional del componente Auditoría

R1 Registrar trazas

Cuando es ejecutada una acción de interés para ser auditada en uno de los subsistemas que componen el sistema GINA, se genera una traza que es guardada por el subsistema de administración con los parámetros:

- Usuario que ejecutó la acción
- Subsistema sobre el cual se ejecutó
- Objeto y tipo de objeto sobre el cual se ejecutó la acción.
- Fecha de registro
- IP desde donde se accedió
- Operación y tipo de operación que se realizó
- Texto formateado

A continuación se describe el requisito funcional Registrar trazas del componente Auditoría.

Precondiciones	Ha sido ejecutada una operación sobre un subsistema del SUA.	
Flujo de eventos		
Flujo básico Registrar autenticación de usuario		
No	**Actor**	**Sistema**
1	Un subsistema del GINA genera una traza de las operaciones que fueron realizadas sobre él.	
2		Guarda la traza generada por el subsistema. De esta traza se guarda: • Id de la Operación y tipo de operación realizada. • Usuario que realizó la operación. • IP desde donde se realizó la operación. • Fecha y Hora en la que se realizó. • Id del tipo de objeto del negocio. • Id del objeto del negocio. • Texto formateado. • Subsistema
3		Concluye el requisito.
Pos-condiciones		
1	Queda registrada la traza generada sobre el subsistema.	
Validaciones		

1	Se validan los datos introducidos según el Modelo Conceptual – Auditoría [1].	
Relaciones	**Requisitos Incluidos**	No aplica.
	Extensiones	No aplica.
Requisitos especiales	No aplica.	
Asuntos pendientes	No aplica.	

2.3.3 Requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requerimientos funcionales, es decir una vez se conozca lo que el sistema debe hacer se determina cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser.

A continuación se muestran los requisitos no funcionales definidos en el Departamento de Soluciones para la Aduana tomados del documento de especificación de los requisitos no funcionales del proyecto GINA.

2.3.3.1 Usabilidad

Los sistemas desarrollados deben prestar facilidades de usabilidad que satisfagan las necesidades de todos los usuarios. Estos deben contar con un menú que les permita a los usuarios acceder a las principales funciones que son de su interés; así como brindar a los usuarios la posibilidad de interactuar con los diferentes productos sin tener previa preparación, solo con los conocimientos necesarios del negocio, en este aspecto no se incluye la parte administrativa de las aplicaciones que si requerirán de una preparación para su manipulación. La resolución de la página se adaptará la resolución de pantalla en cada cliente.

2.3.3.2 Fiabilidad

- ***Disponibilidad:*** Las aplicaciones deben ser capaces de estar operativas durante el mayor tiempo posible para brindar sus servicios a los usuarios ininterrumpidamente durante el tiempo que estos lo necesiten. El tiempo para la mantención de las aplicaciones no debe exceder el 10% con respecto al tiempo que la aplicación debe estar disponibles.
- ***Tiempo medio entre fallos:*** El tiempo mínimo de disponibilidad ante posibles fallos será en dependencia de las especificaciones de cada sistema.
- ***Tiempo medio de reparación:*** La recuperación ante los fallos será en dependencia de las especificaciones. de cada sistema.

2.3.3.3 Eficiencia

- **Tiempo de respuesta por transacción:** 1.09 segundos.
- **Rendimiento:** 1000 transacciones por segundos, cantidad de datos que pueden ser transferidos.
- **Capacidad:** 80 clientes que pueden estar conectados.
- **Utilización de recursos:** 18 % memoria y 20 % en disco duro.

2.3.3.4 Soporte

- Las aplicaciones clientes deben funcionar sobre cualquier plataforma. Para la plataforma Windows, se recomienda utilizar la versión XP por la experiencia acumulada por los usuarios. Para la parte servidora se recomienda que corra sobre plataforma Linux.
- Ser programado en PHP 5.x o superior y con un gestor de base de datos Oracle 8i o superior, o PostgreSQL 8.x o superior.
- El sistema debe poseer una alta seguridad.

2.3.3.5 Software

Los usuarios deben tener instalado como navegador el Mozilla Firefox 3.5, el Internet Explorer 6 o superior. Sistema Operativo Windows NT o Linux, para PC o MacOS para Macintosh. Nada de esto es necesario para los clientes ligeros.

El servidor de aplicación debe tener instalado PHP 5.2.7 o superior, Servidor Web Apache 2.2.10 o superior. En la Base de Datos es necesario que esté instalada una versión de PostgreSQL 8.2.4 o superior.

2.3.3.6 Hardware

***Cliente*:** Las aplicaciones son desarrolladas para que las PC clientes de los usuarios puedan usar las aplicaciones con el menor requerimiento posible. Estos requerimientos mínimos son:

- Procesador Pentium IV o superior, 333 MHz mínimo pero recomendado 1.8 GHz
- 40 GB o más de capacidad.
- Mínimo de memoria RAM 128 DIM
- Adaptador de Red y conectividad.

Se recomienda la utilización de clientes ligeros para aprovechar las facilidades de mantenimiento que esto brinda.

***Servidor*:** Debido a que la cantidad de datos manejados por este tipo de aplicaciones se requiere separar el servidor de aplicación y el de Base de Datos. Cada uno de ellos requiere de características de Hardware específicas.

Servidor de Aplicaciones:

- CPU = 2 Dual Core AMD Opteron 64 bits a 2.2 GHz
- RAM = 2 GB
- Almacenamiento = 1 Discos SATA de 73 GB en RAID 1

***Servidor de Base de Datos*:**

- CPU = 2 Dual Core AMD Opteron 64 bits a 2.2 GHz
- RAM = 4 GB
- Almacenamiento = 2 Discos SATA de 73 GB en RAID 1

2.3.3.7 Restricciones de diseño

Los sistemas realizados para los procesos aduanales incluyen el manejo de varios asuntos que se enlazan entre sí, por lo que constantemente se necesita acceder a información común para varios de

estos procesos, es aquí donde entra la reutilización de dicha información, de forma tal que los datos no aparezcan duplicados.

2.3.3.8 Interfaz

Para los requisitos de interfaz se debe consultar la descripción de los requisitos funcionales de cada subsistema donde se detallarán todos los requisitos de interfaz a continuación del prototipo de interfaz de usuario que se propone. (32)

2.3.4 Aplicación de las técnicas de validación de los requisitos

Una vez concluida la especificación de los requisitos es de vital importancia que los mismos sean validados, pues esto permite detectar errores antes de pasar a la siguiente fase en el ciclo de desarrollo del software, además de evitar pérdidas de tiempo y que no se produzcan costos excesivos. A continuación se aplica la técnica para la validación del requisito Registrar trazas identificado.

Revisión Técnica Formal (RTF): Terminada la especificación de los requisitos funcionales, el analista principal del módulo Auditoría en conjunto con la analista principal del Departamento de Soluciones para la Aduana realizaron la reunión de revisión donde se aprobó la especificación descrita.

2.4 Modelo conceptual

Con la realización del modelo conceptual se trata de obtener el esquema conceptual de la base de datos a partir de la lista descriptiva de objetos y asociaciones identificadas en la organización durante el análisis.

Seguidamente se presenta el diseño del modelo conceptual del componente de Auditoría.

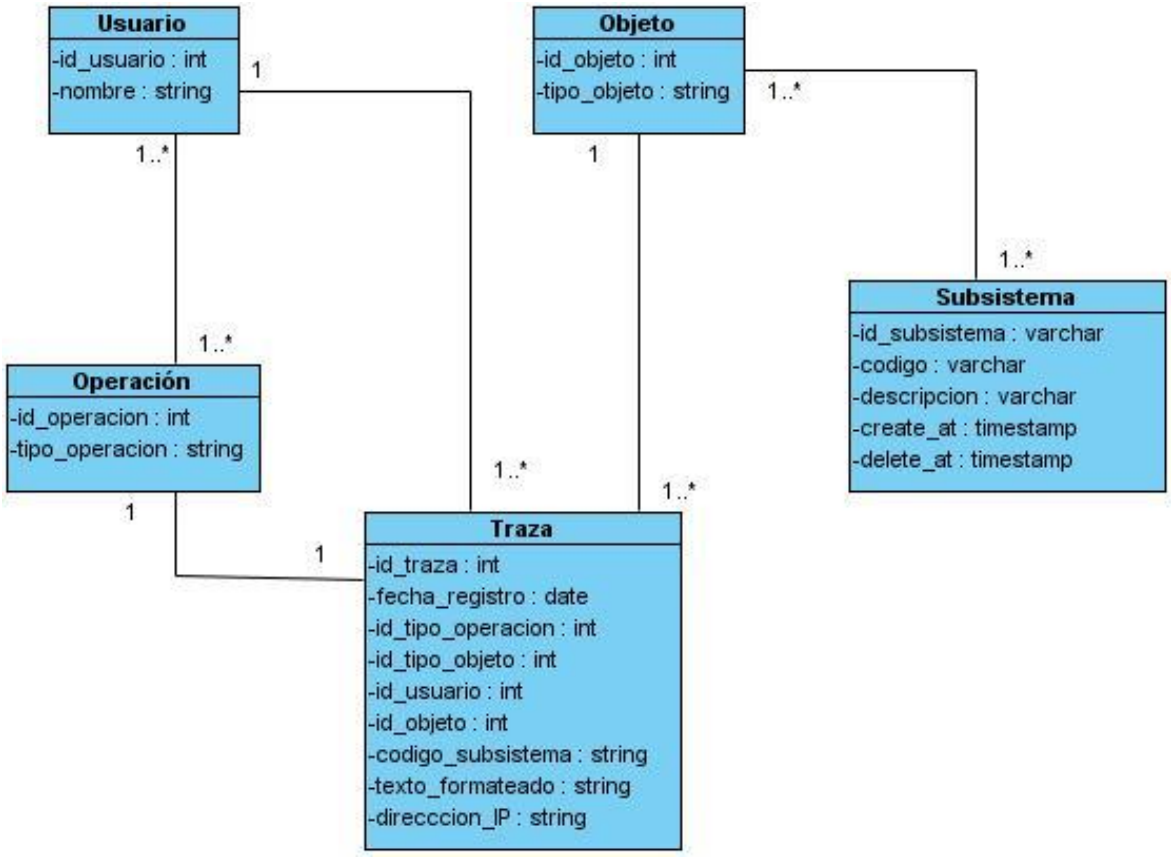

Fig. 2.2 Modelo Conceptual

Se adjuntan como anexos los diccionarios de datos correspondientes a este modelo conceptual.

2.5 Diseño de clases

En este epígrafe se presenta el diseño de las clases persistentes que pertenecen al componente de Auditoría del subsistema de Administración, sirve como una primera aproximación al diseño definitivo del modelo de datos.

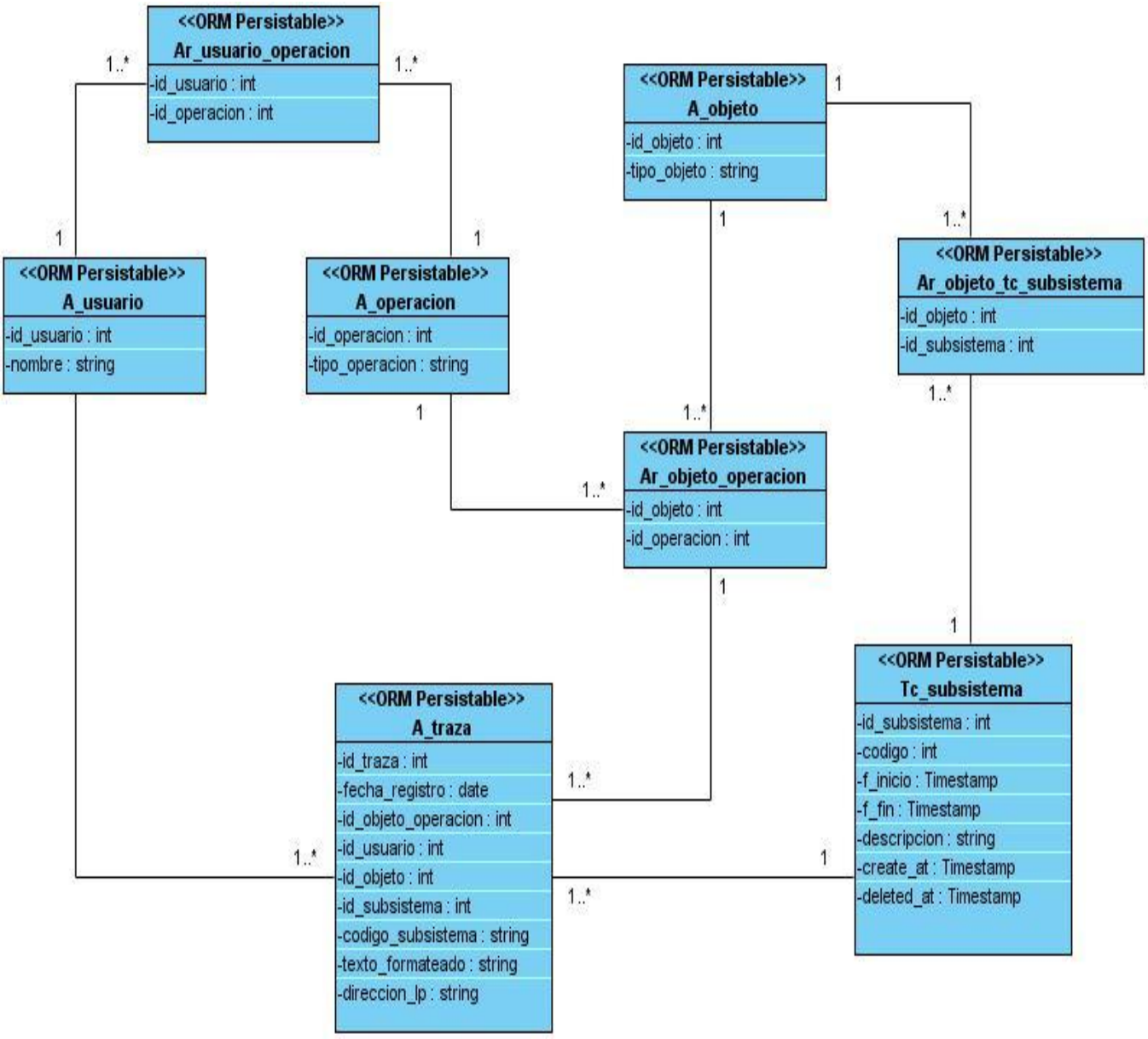

Fig. 2.3 Diagrama de clases del componente de Auditoría

2.6 Descripción de las clases

A continuación se hace una descripción de las clases que intervienen en el diagrama de clases persistentes.

2.6.1 Clase AUsuario

Nombre de la clase:	AUsuario	
Estereotipo:	<<ORM Persistable>>	
Descripción:	Esta clase es la encargada de manipular información referente al usuario.	
Tabla correspondiente:	SU_ADMIN_USUARIO	
Atributos		
Nombre	**Tipo**	**Descripción**
id_usuario	Integer	Identificador del usuario
usuario	String	Nombre de usuario que será utilizado para acceder a la aplicación.

2.6.2 Clase AObjeto

Nombre de la clase:	AObjeto	
Estereotipo:	<<ORM Persistable>>	
Descripción:	Esta clase es la encargada de manipular información referente a los objetos del negocio.	
Tabla correspondiente:	A_OBJETO	
Atributos		
Nombre	**Tipo**	**Descripción**
id_objeto	Integer	Identificador del objeto.
tipo_objeto	String	Tipo de Objeto

2.6.3 Clase AOperacion

Nombre de la clase:	AOperacion	
Estereotipo:	<<ORM Persistable>>	
Descripción:	Esta clase es la encargada de manipular información referente a las operaciones.	
Tabla correspondiente:	A_OPERACION	
Atributos		
Nombre	**Tipo**	**Descripción**
id_operacion	Integer	Identificador de la operación que se realiza.
tipo_operacion	String	Tipo de operación que se realiza sobre un objeto.

2.6.4 Clase TcSubsistema

Nombre de la clase:	TcSubsistema	
Estereotipo:	<<ORM Persistable>>	
Descripción:	Esta clase es la encargada de manipular información referente a los subsistemas.	
Tabla correspondiente:	TC_SUBSISTEMA	
Atributos		
Nombre	**Tipo**	**Descripción**
id_subsistema	Integer	Identificador del subsistema desde el que se realiza la operación.
codigo	Varchar	Nombre del subsistema desde donde se realiza la operación.
descripción	Varchar	Breve descripción del subsistema.
create_at	TIMESTAMP	Fecha de creación del subsistema.
delete_at	TIMESTAMP	Fecha en la que se deshabilita el subsistema.

2.6.5 Clase ATraza

Nombre de la clase:	ATraza
Estereotipo:	<<ORM Persistable>>
Descripción:	Esta clase es la encargada de manipular información referente a las trazas.
Tabla correspondiente:	A_TRAZA

Atributos

Nombre	**Tipo**	**Descripción**
Id_traza	Integer	Identificador de la traza que se genera.
Id_objeto_operacion	Integer	Identificador de la operación que se realizó y el objeto que fue modificado.
Id_usuario	Integer	Identificador del usuario que realizó la operación.
Id_objeto	Integer	Identificador del objeto de negocio que ha sido modificado.
direccion_IP	String	Dirección desde la cual se accede para realizar la operación.
codigo_subsistema.	String	Código del subsistema donde se realizó la operación.
Id_subsistema	Integer	Identificador del subsistema en el cual se realizó la operación.
texto_formateado	String	Descripción del objeto que se está modificando en ese momento.
fecha_registro	Date	Fecha y hora en la que se realizó la operación a registrar.

2.7 Diagrama de paquetes

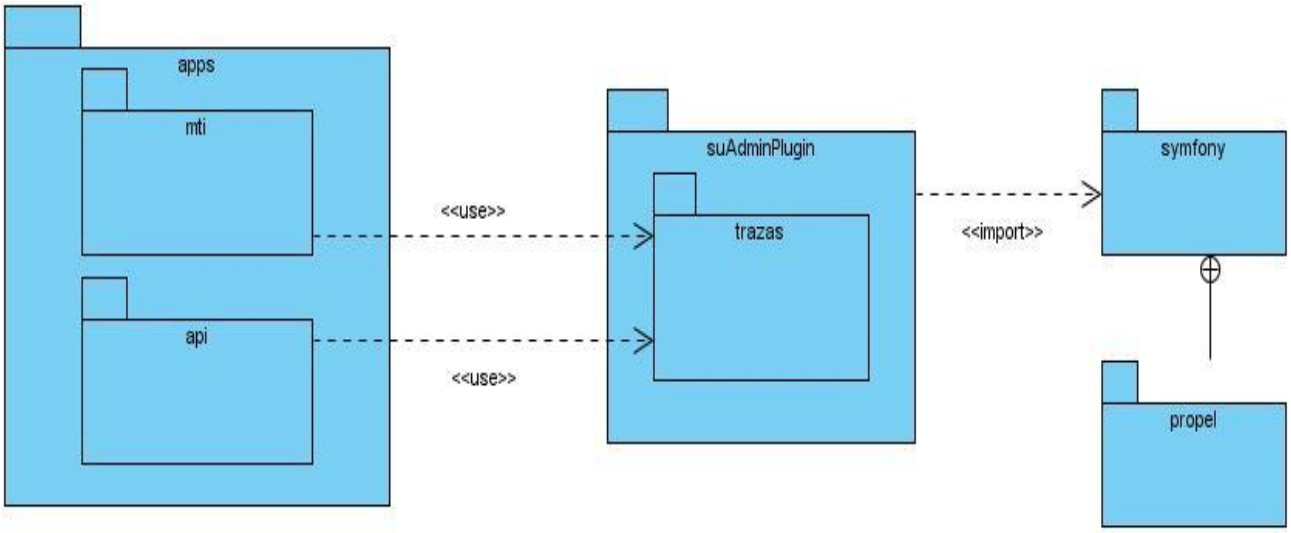

Fig. 2.4 Diagrama de paquetes

Todas las aplicaciones que se encuentran en el sistema GINA utilizarán las funcionalidades del componente Trazas del subsistema de seguridad suAdminPlugin.

2.8 Diseño del modelo de datos

A continuación se muestra el diagrama entidad – relación correspondiente al componente de Auditoria del subsistema de seguridad suAdminPlugin.

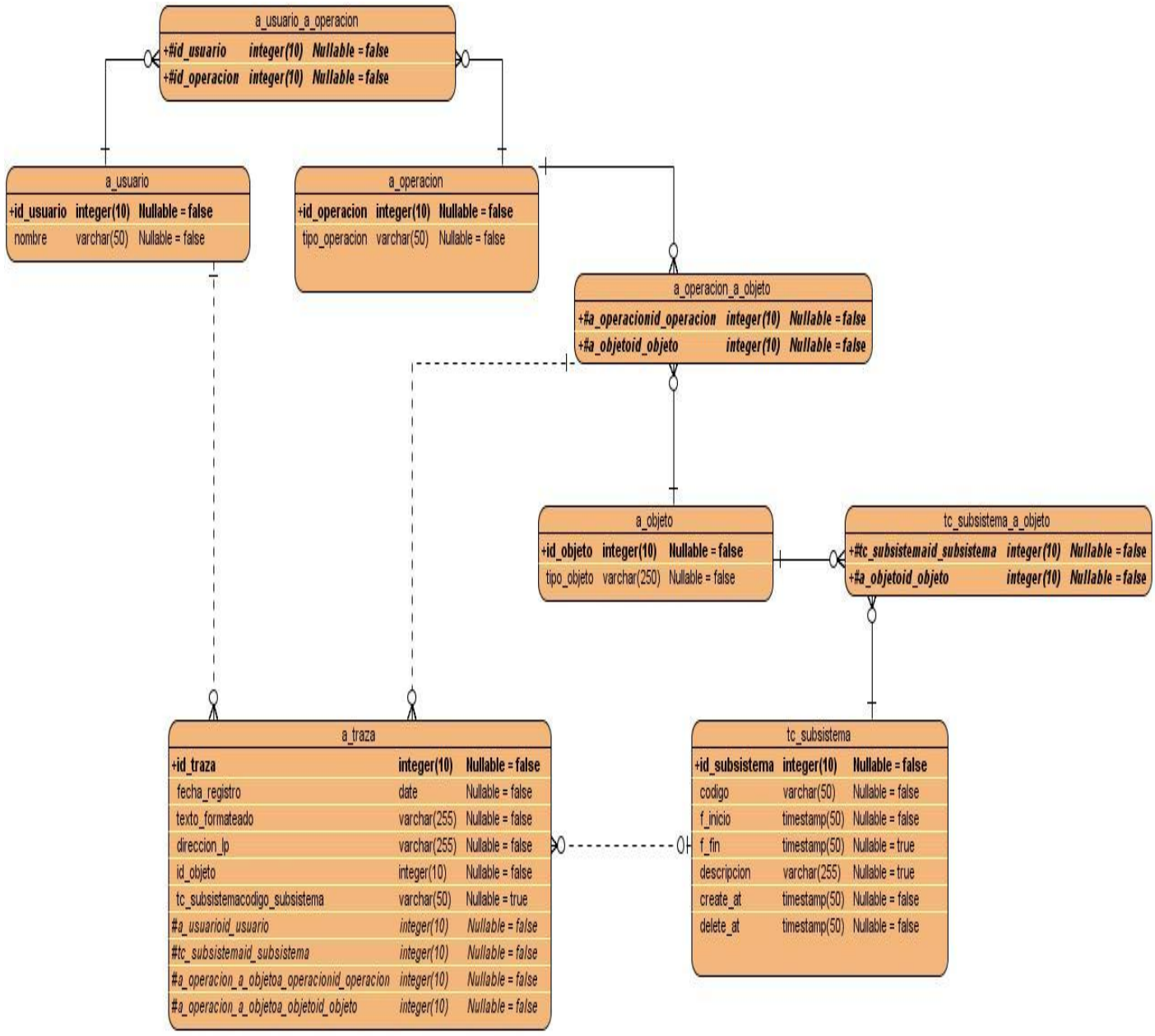

Fig. 2.5 Diagrama entidad – relación

2.9 Diagrama de clases del diseño

A continuación se muestra el diagrama de clases del diseño para una mejor comprensión del componente que se desarrolla.

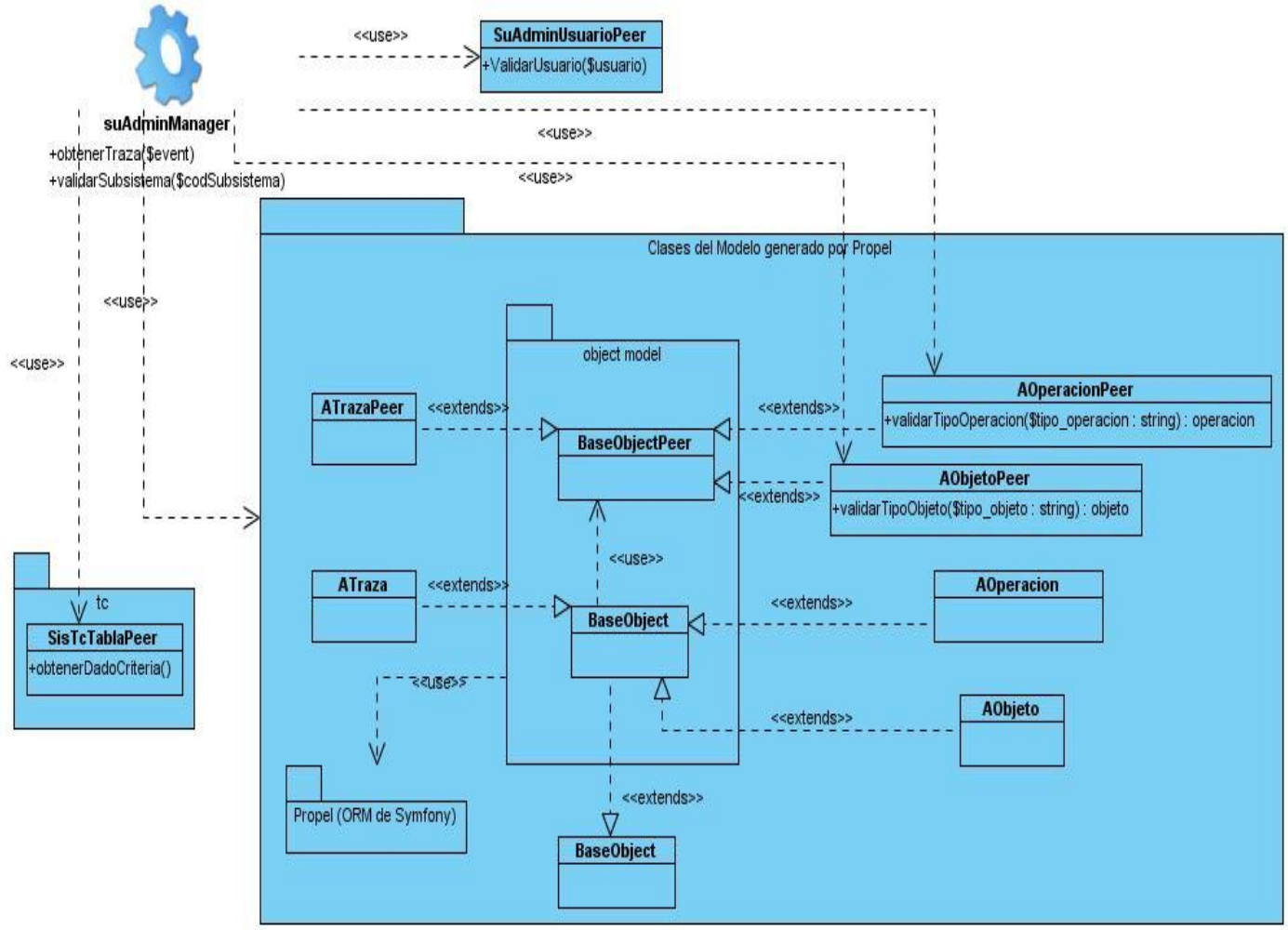

Fig. 2.6 Diagrama de clases del diseño

2.10 Diagrama de secuencia

En este epígrafe se muestra el diagrama de secuencia del requisito funcional identificado Registrar Trazas.

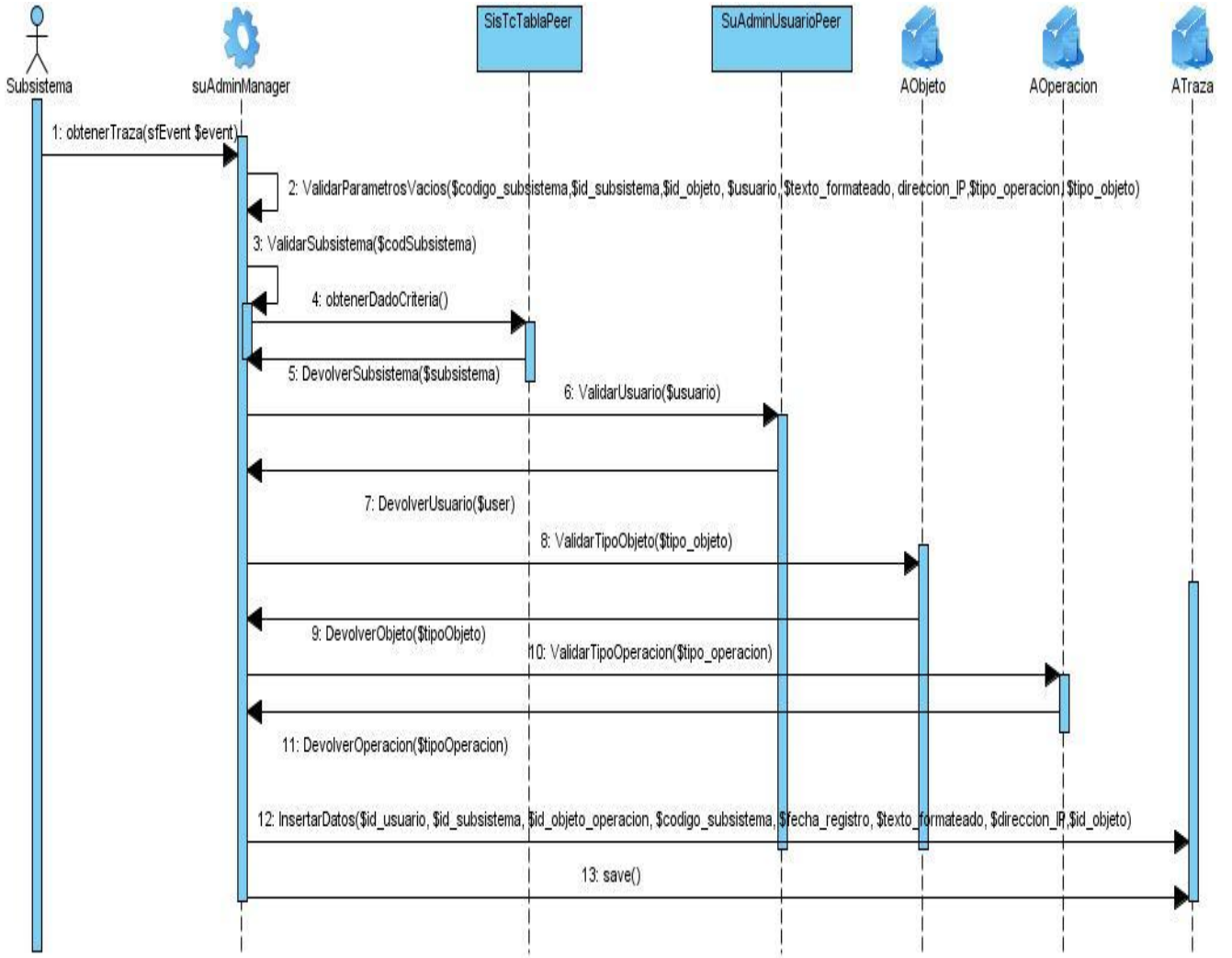

Fig. 2.7 Diagrama de secuencia

2.11 Patrones de software utilizados

Patrón Arquitectónico

La selección del patrón arquitectónico utilizado se basa en la selección del Framework a utilizar, a continuación se da una pequeña descripción de la implementación descrita por el patrón Modelo Vista Controlador (MVC) ya que es el que se pone de manifiesto en Symfony.

2.11.1 MVC

Es un patrón de diseño 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.

***Modelo*:** Esta es la representación específica del dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. Es el responsable de acceder a la capa de almacenamiento de datos.

***Vista*:** Esta presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de la interfaz de usuario. Ejemplo un Formulario.

***Controlador*:** Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Muchas aplicaciones utilizan un mecanismo de almacenamiento persistente (como puede ser una base de datos) para almacenar los datos.

MVC no menciona específicamente esta capa de acceso a datos. Es común pensar que una aplicación tiene tres capas principales: presentación (IU), dominio, y acceso a datos. En MVC, la capa de presentación está partida en controlador y vista. La principal separación es entre presentación y dominio; la separación entre la vista y el controlador es menos clara. Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente: El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace) MVC no menciona específicamente esta capa de acceso a datos. Es común pensar que una aplicación tiene tres capas principales: presentación (IU), negocio y acceso a datos. En MVC, la capa de presentación está partida en controlador y vista. La principal separación es entre presentación y negocio; la separación entre MVC es menos clara.

El principio más importante de la arquitectura MVC es la separación del código del programa en tres capas, dependiendo de su naturaleza. La lógica relacionada con los datos se incluye en el modelo, el código de la presentación en la vista y la lógica de la aplicación en el controlador.

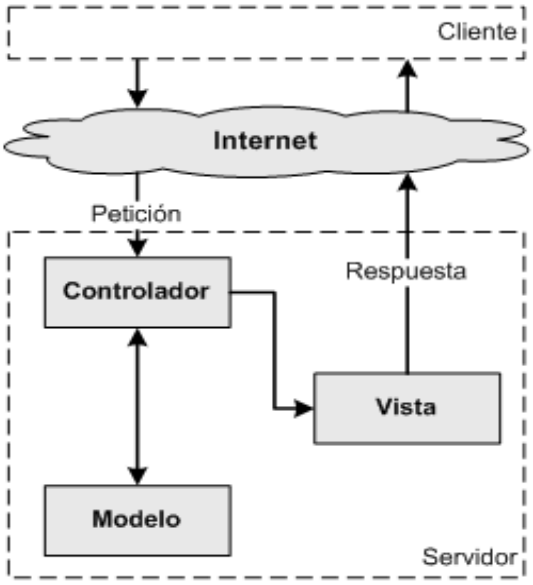

Fig. 2.8 Separación de las capas lógicas del MVC

Patrones de Diseño

Los patrones de diseño de software son soluciones reutilizables de problemas recurrentes que aparecen durante el proceso de diseño de software orientado a objetos. Estos surgen por la necesidad de transmitir la experiencia a los demás desarrolladores.

Con el conocimiento de estos patrones, los programadores expertos son capaces de identificar las situaciones en las que éstos tienen aplicación, y utilizarlos sin tener que detenerse para analizar el problema y vislumbrar diferentes estrategias de resolución.

A continuación se realiza una descripción de algunos de los patrones de diseño utilizados en la solución planteada.

2.11.2 Patrones GoF[27]

2.11.2.1 Fachada

Problema: ¿Como acceder a diferentes clases a través de una única clase?

[27] Group of Four (Grupo de los 4).

Propósito: Simplificar el acceso a un conjunto de clases o interfaces. Proporcionar una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. La "fachada" satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas.

Solución: Symfony permite relacionar dos aplicaciones o subsistemas a través de la clase *SysComponetLocator.php* la cual actúa como fachada entre ambas (os) permitiendo el acceso a una de ella mediante servicios.

2.11.2.2 Plantilla de Método (Template Method)

Problema: ¿Cómo definir la estructura de un algoritmo?

Propósito: Definir el esqueleto básico de un algoritmo. Se trata de una clase que define parte de un algoritmo mediante clases abstractas. Posteriormente, las subclases modifican los métodos abstractos para implementar las acciones concretas. Factorizar el comportamiento común de varias subclases Implementar las partes fijas de un algoritmo una sola vez y dejar que las subclases implementen las partes variables.

Solución: Symfony tiene una estructura específica a la hora de implementar un método, esto depende de la clase en que se implemente la funcionalidad.

2.11.2.3 Singleton

Problema: ¿Cómo obtener un punto de acceso global a una clase?

Propósito: Asegurar que solo exista una instancia de una clase especifica en un sistema a desarrollar.

Solución: A través de la clase *SysComponentLocator.php* se garantiza una única instancia, a través de la cual se pueden acceder a los demás servicios que sean implementados en el sistema en general.

2.11.3 Patrones GRASP[28]

2.11.3.1 Bajo acoplamiento

Problema: ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización?

Propósito: Aumentar la reutilización y eliminar las dependencias entre las clases para propiciar un fácil mantenimiento y entendimiento.

Solución: Symfony asigna a cada clase una responsabilidad para mantener pocas dependencias entre las mismas.

2.11.3.2 Alta cohesión

Problema: ¿Cómo mantener la complejidad dentro de límites manejables?

Propósito: Cada elemento dentro del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable.

Solución: Symfony agrupa las clases por funcionalidades que son fácilmente reutilizables, bien por su uso directo o por herencia.

2.11.3.3 Controlador

Problema: ¿Quién debería encargarse de un evento del sistema?

Propósito: Facilitar la centralización de actividades, delegar las actividades en otras clases con las que mantiene un modelo de alta cohesión.

Solución: En la solución planteada, la clase *suAdminManager.php* hace función de controladora ya que ahí es donde se implementa las principales funcionalidades de la misma y maneja a su vez los eventos del sistema.

[28] (General Responsibility Assignment Software Patterns): Patrones generales de software para asignación de responsabilidades.

2.11.3.4 Experto

Problema: ¿Cual es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos?

Propósito: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Solución: Symfony incluye la librería Propel para mapear la base de datos. La misma es utilizada para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades, las clases de abstracción de datos (Peer del Modelo) poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan.

Conclusiones del capítulo

Durante el desarrollo del capítulo, luego de llevar a cabo la descripción detallada de la solución que ha sido propuesta y la generación de los artefactos correspondientes a los flujos de Análisis y Diseño, además de la selección de los principales patrones de software a utilizarse, se obtuvo, como resultado, una primera aproximación a la solución del problema que ha sido planteado inicialmente en la introducción del presente trabajo de diploma.

Capítulo 3: Implementación y Prueba de la solución.

Introducción

En el presente capítulo se realiza una descripción de los artefactos que son generados durante la implementación del sistema. Se muestra la implementación de los componentes que se utilizan en la aplicación, así como los estándares de codificación utilizados para lograr el desarrollo del mismo. También se notificarán los resultados de un conjunto de pruebas unitarias realizadas durante las diferentes fases de construcción.

3.1 Diagrama de componentes

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo del diseño. Algunos de los estereotipos estándar de componentes son los siguientes:

- <<executable>>: Es un programa que puede ejecutar en un nodo.
- <<file>>: Fichero que contiene código fuente o datos.
- <<library>>: Es una librería estática o dinámica.
- <<table>>: Es una tabla de la base de datos.
- <<document>>: Es un documento.

A continuación se muestra el diagrama de componentes correspondiente al servicio de Auditoría que se desarrolla:

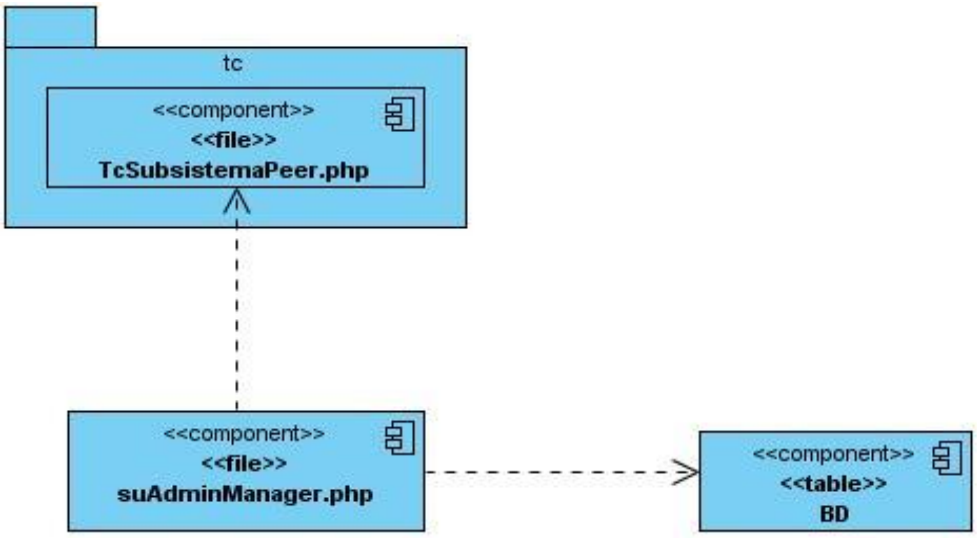

Fig. 3.1 Diagrama de componentes del servicio de Auditoría.

3.2 Estándares de codificación

Un estándar de codificación completo 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, como si un único programador hubiera escrito todo el código de una sola vez.

Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada.

La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. El mantenimiento del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento.

3.2.1 Symfony

Entre las normas las normas seguidas por el código de Symfony, se encuentra el estándar UpperCamelCase[29] para el nombre de clases y variables. Solamente existen dos excepciones: las

[29] Consiste en escribir frases o palabras compuestas eliminando los espacios intermedios y poniendo en mayúscula la primera letra de cada palabra incluyendo la primera letra de la frase.

clases del núcleo de Symfony empiezan por *sf* (por tanto en minúsculas) y las variables utilizadas en las plantillas que utilizan la sintaxis de separar las palabras con guiones bajos.

Es necesario comentar todo el código para facilitar su revisión y entendimiento, lo que aumentará su legibilidad y por consiguiente que sea más fácil de darle mantenimiento a la aplicación.

3.2.1.1 Plugins

Symfony especifica que la notación de los plugins siempre debe estar atada al sufijo *Plugin.* El nombre de un Plugin debe indicarse con la menor cantidad de palabras cuál es el objetivo del Plugin e iniciarse con el prefijo *sf.*

Ejemplo: *sfMiPropioPlugin*

3.2.1.2 Nombre de las clases

- Los nombres de las clases deben estar expresados en notación UpperCamelCase.
- No se deben utilizar guiones bajos en su nombre "_".
- Deben expresar con claridad cuál es el alcance y la responsabilidad de la clase.
- Los nombres de las clases no deben estar atados a las clases de las que se deriva, cada clase debe tener un significado por ella misma, no en dependencia de la clase de la que deriva.
- En los nombres compuestos por más de tres palabras se debe revisarse el diseño, no sea que se le estén dando a la clase más responsabilidades de las que realmente tiene.

Modelo

Las clases del modelo generadas por Propel deben cumplir con la nomenclatura que propone el Framework para las clases de este tipo, el mismo nombre de la tabla que está haciendo referencia basándose en la utilización de la notación UpperCamelCase y sustitución de los guiones bajos de los nombres de las tablas por palabras compuestas.

Se mantiene el uso de los sufijos *"Peer"* para las clases que se encargan de estas funciones en el modelo.

Negocio

Las clases del negocio deben cumplir con claridad las reglas de nomenclatura de las clases.

Formularios

Los formularios tendrán el sufijo *Form* para identificarlos.

Ejemplo: *AOperacionForm.class.php*

3.2.1.3 Pruebas unitarias

Las pruebas unitarias deben estar agrupadas por clases a no ser que una clase contenga demasiadas funcionalidades y por rendimiento sea necesario separar alguna o algunas funcionalidades en archivos diferentes.

- Los nombres de los archivos deben coincidir con el nombre de la clase o la funcionalidad que están probando terminada en el sufijo "*Test.php*".
- En el caso de que sean funcionalidades de una clase o archivo determinado deberán agruparse dentro de un directorio con el nombre de la clase.
- Los archivos de pruebas se colocarán en el directorio raíz del proyecto, dentro del directorio "*test/unit*" y dentro de esta seguirán una estructura de directorios similares al archivo que es objeto de las pruebas.

```
##Archivo de la clase que se desea probar
MyProyecto\
    apps\
      fronted\
            lib\
           MyClaseAProbar.class.php
 ##Archivos de pruebas
MyProyecto\
    test\
      unit\
          fronted\
              lib\
                 MyClaseAProbarTest.php
```

- Cada vez que se cree un grupo de pruebas a una funcionalidad se debe comenzar la llamada a la prueba con la función *diag()* pasándole como parámetros el nombre de la función que se está

probando. Este mensaje saldrá al comienzo de la llamada a las pruebas de dicha función mostrando en la CLI[30] que es lo que se está mostrando.

- En las llamadas a cada una de las pruebas se le pasará como tercer parámetro un mensaje de texto indicando qué es lo que se está probando.

```
// srttolower()

$t->diag('strtolower()');

$t->isa_ok(strtolower('Foo'), 'string', 'strtolower() devuelve una cadena de texto');

$t->is(strtolower('FOO'), 'foo', 'strtolower() transforma la entrada en minúsculas');
```

3.2.2 Base de Datos

El objetivo de crear un estándar de codificación a nivel de Base de Datos es que los diseñadores de base de datos y diseñadores del sistema se rijan por el mismo a la hora de implementar. Este estándar debe servir para el personal del proyecto a identificar de forma sencilla cual es el objetivo y las funcionalidades que brinda cada una de las tablas y campos de la Base de Datos dada su nomenclatura, es necesario que esto se pueda identificar a simple vista. Además de crear una estandarización en el desarrollo de las mismas. Esto posibilita que el código sea más fácil de mantener, sirve como punto de referencia para los programadores, mantiene un estilo de programación y ayuda a mejorar el proceso de codificación, haciéndolo más eficientes.

3.2.2.1 Esquemas

Todas las tablas pertenecerán a un esquema determinado, este esquema debe tener un identificador de no más de 8 caracteres alfanuméricos. Nunca comenzarán con un número, siempre con una letra en el primer lugar.

3.2.2.2 Tablas

Los nombres de las tablas se escribirán siempre en mayúsculas y en singular. Esto es debido a que una tabla representa un objeto del Sistema, mapeado a la Base de Datos.

Ejemplo: *COCHE, MARCA, ELECTRODOMESTICO*

[30] Líneas de comando.

Las tablas comenzarán con el identificador del esquema al que pertenecen, seguido de un guión bajo y el nombre de la tabla.

Ejemplo: *RRHH_TRABAJADOR*

En caso de que el nombre de una tabla conste de más de una palabra, los espacios en blanco se marcarán con un guión bajo. Además, las palabras seguirán estando todas en singular.

Ejemplo: *LINEA_FACTURA, MOTOR_AVION.*

Únicamente se utilizarán caracteres alfabéticos, salvo que por la naturaleza del nombre se necesiten dígitos numéricos. Se prohíbe el uso de caracteres de puntuación o símbolos.

Ejemplo: *LOCALIDAD_CENSO_2003*

En el caso de las especializaciones las tablas se nombrarán con la nueva denominación, el nombre de la especialización no tiene que estar atado a la tabla más general que la precede.

Las tablas de relación (objetos asociativos, representan relaciones de N a M) deben nombrarse utilizando los nombres de las tablas intervinientes, siguiendo un orden lógico de frase, se utilizarán sólo caracteres en mayúsculas, y se separarán los nombres de las tablas intervinientes en la relación utilizando: _

Ejemplo: *PASAJERO_VIAJE*

Toda relación entre tablas debe implementarse mediante constraints (claves foráneas) con integridad referencial, de acuerdo al motor de base de datos utilizado. A las tablas se le escribirá para que son usadas en la pestaña Notes donde se especificará de la siguiente manera:

"*La tabla NOMBRE DE LA TABLA se utiliza para conocer (…), esto se conoce a partir del campo (…), además de conocer (…), mediante el campo (…).*"

3.2.2.3 Campos

Los campos clave deben ubicarse al inicio de la definición de la tabla (deben ser los primeros).
Toda tabla debe poseer uno o más campos llave.

El nombre del campo clave debe estar compuesto por “id” + nombre de la tabla en singular (para claves no compuestas). Dependiendo de la naturaleza de la entidad, el nombre de la tabla a usar es el de la misma tabla, o el de la relacionada.

Ejemplo para la tabla *COCHE: id_coche, para la tabla LINEA_FACTURA: id_linea_factura*

Las claves compuestas sólo deben utilizarse en casos específicos, por ejemplo, tablas de relación o entidades débiles. Si una tabla X con clave compuesta necesita ser referenciada desde otra tabla Y, deberá generarse un campo clave en X al inicio de la misma como “id_X”, y generar un constraint único en los campos que la identificaban.

Los campos se escribirán en minúsculas y en singular. Solo se escribirán en plural aquellos que por su significado lo requieran.

Ejemplo: *nombre*

Las palabras de los campos se separarán por guiones bajos.

Ejemplo: *segundo_nombre*

Los campos de tipo fecha pueden comenzar con “f_” seguido del nombre del campo. Además se pueden utilizar las nomenclaturas “fecha” para los nombres compuestos.

Se puede usar la nomenclatura en inglés create_at, update_at para los campos que indiquen cuando un registro es creado o actualizado. Estos casos se utilizarán en casos muy puntuales y cuando se apruebe por los diseñadores de Base de Datos.

Ejemplo para el campo fecha de nacimiento*: f_nacimiento, fecha_registro, create_at*

Los campos booleanos deberán nombrarse de acuerdo al estado correspondiente al valor 1/Verdadero/True del contexto en el que se encuentran modelados.

Los campos de relación (foreign keys, claves foráneas) deben nombrarse de la misma manera que los campos clave (usando el nombre de la tabla a la que hacen referencia).

Ejemplos: *tabla persona => id_tipo_documento, id_estado_civil*

En ninguno de los casos estará reflejado en el nombre del campo el tipo de dato que se representa.

No se pueden poner caracteres extraños en los nombres de los campos como ñ, tilde, u otros; estos deben ser sustituidos por símbolos semejantes.

Ejemplo:

Carácter extraño	Símbolo
ñ	nn
á	a
ä	a
/	_

Validación de la solución

Una vez terminada la implementación del producto que se requiere es necesario realizarle pruebas con el objetivo de detectar errores en la aplicación y la documentación; este proceso resulta de gran importancia ya que da una medida de la calidad al mismo siempre que se lleve a cabo de la forma correcta.

Las pruebas de unidad tienen un impacto importante en el código y la calidad del producto final, por esta razón, al componente Trazas implementado se le realizarán las pruebas unitarias tal y como están definidas en el marco de trabajo utilizado en el desarrollo del sistema GINA.

3.3 Pruebas unitarias en Symfony

Las pruebas unitarias constituyen la forma de probar el correcto funcionamiento de un código perteneciente a un módulo o subsistema en general. El objetivo de las mismas es aislar cada parte del programa y mostrar que las partes individuales son correctas.

Las pruebas unitarias aseguran que un único componente de la aplicación produce una salida correcta para una determinada entrada. Este tipo de pruebas validan la forma en la que las funciones y métodos trabajan en cada caso particular.

Para que una prueba unitaria se buena se deben cumplir una serie de requisitos, estas deben cumplir con ser:

- *Automatizables:* no debería requerirse una intervención manual.
- *Completas:* deben cubrir la mayor cantidad de código.

- *Repetibles o reutilizables:* no se deben crear pruebas que sólo puedan ser ejecutadas una sola vez.
- *Independientes:* la ejecución de una prueba no debe afectar a la ejecución de otra. +
- *Profesionales:* las pruebas deben ser consideradas igual que el código, con la misma profesionalidad y documentación.

Aunque estas precisiones no tienen que ser cumplidas al pie de la letra, es recomendable seguirlas, ya que pueden perder parte de su función.

Symfony incluye su propio Framework de automatización de pruebas llamado Lime, creado con el objetivo de facilitar la lectura de los resultados de las pruebas. Ofrece además la facilidad de que cada una de estas se realice en un entorno independiente para evitar interferencias de funcionamiento creadas por otras aplicaciones.

A continuación se detallan los procedimientos llevados a cabo para ejecutar las pruebas unitarias al componente desarrollado.

3.3.1 Procedimientos desarrollados para la ejecución de las pruebas.

3.3.1.1 Definición del alcance

Se realizarán pruebas de unidad de forma automática utilizando el framework Symfony. Estas pruebas están orientadas a descubrir errores en el código del programa.

3.3.1.2 Definición de los objetivos

Verificar que el código de las funcionalidades implementadas en el componente de Auditoría del subsistema de seguridad suAdminPlugin está funcionando correctamente; comparar los resultados obtenidos respecto a los resultados que se esperan de los métodos.

3.3.1.3 Descripción de la plantilla Casos de Prueba de Unidad

En este paso se prosigue a llenar los datos solicitados en la Plantilla Casos de Prueba de Unidad para el componente Trazas.

Nombre del Proyecto: Gestión Integral de Aduanas

Nombre del Componente: Auditoría

Versión: 1.0

Tipo de prueba: Prueba de Unidad

Funcionalidad a probar	**Descripción**
obtenerTraza	Obtiene las traza que el envían por parámetro y la inserta en la base de datos.

Caso de Prueba de Unidad					
Código del caso de prueba: CPU1					
Iteración: 1ra					
Descripción de la prueba: Se realizan pruebas de unidad a las funcionalidades del componente de Auditoría.					
Nombre del encargado: Danarys Cancio Quintana					
Funcionalidad	**Método Utilizado**	**Recibe**	**Resultado esperado del método.**	**Resultado esperado de la prueba.**	**Resultado real de la prueba.**
obtenerTraza	is	´dnc´, ´18´, ´danarys´, ´3´, ´aeronave´, ´denuncia´, ´Se realiza una denuncia´, ´10.32.17.164´	true	ok	ok

3.3.1.4 Implementar pruebas automáticas

A continuación se muestra la imagen de las pruebas automáticas que se le realizaron a las funcionalidades del componente de Auditoría implementadas en la clase *ServicioTrazasTest.class.php.*

```
require_once dirname(__FILE__) . '/../bootstrap/unit.php';
$configuration = ProjectConfiguration::getApplicationConfiguration('tc', 'test', true);
$f = new sfDatabaseManager($configuration);
$conexion = Propel::getConnection('administracion');
$conexion->beginTransaction();

$t = new lime_test(1, new lime_output_color());
$syscomponent = SysComponentsLocator::getInstance();
$componente = $syscomponent->getComponent('administracion');
//---------------PRUEBA-----------------
$db = Propel::getDB('administracion');
$num = $db->getId($conexion,'A_TRAZA_SEQ');
$array['codigo_subsistema'] = 'dnc';
$array['id_subsistema'] = '18';
$array['usuario'] = 'danarys';
$array['id_objeto'] = '3';
$array['tipo_objeto'] = 'aeronave';
$array['tipo_operacion'] = 'denuncia';
$array['texto_formateado'] = 'Se realiza una denuncia';
$array['direccion_IP'] = '10.32.17.164';

try {
    $traza = $componente->executeService(new sfEvent(null, 'administracion.obtenerTraza', $array));
} catch (Exception $exc) {
    echo $exc->getMessage();
}
$t->diag('obtenerTraza()');
$t->is($traza, true, 'obtenerTraza - suAdminManager::Queda registrada correctamente la traza en la BD');
$conexion->commit();
```

Fig. 3.2 Vista de las pruebas realizadas al componente de Auditoría

3.3.1.5 Evaluación de los resultados

Se procede a la ejecución de las pruebas una vez que este todo listo y definido. En este procedimiento se realiza además la evaluación de los resultados, la cual se detalla a continuación.

Al ser ejecutada la prueba unitaria implementada al componente de Auditoría no se encuentra ningún error en la funcionalidad que en este se encuentra.

En la siguiente imagen se muestra el resultado de la prueba unitaria realizada a la funcionalidad *obtenerTraza* perteneciente al componente de Auditoría.

```
7  require_once dirname(__FILE__) . '/../bootstrap/unit.php';
8  $configuration = ProjectConfiguration::getApplicationConfiguration('tc', 'test', true);
9  $f = new sfDatabaseManager($configuration);
10 $conexion = Propel::getConnection('administracion');
11 $conexion->beginTransaction();
12
13 $t = new lime_test(1, new lime_output_color());
14 $syscomponent = SysComponentsLocator::getInstance();
15 $componente = $syscomponent->getComponent('administracion');
16 //---------------PRUEBA----------------
17 $db = Propel::getDB('administracion');
18 $num = $db->getId($conexion,'A_TRAZA_SEQ');
19 $array['codigo_subsistema'] = 'dnc';
20 $array['id_subsistema'] = '18';
21 $array['usuario'] = 'danarys';
22 $array['id_objeto'] = '3';
23 $array['tipo_objeto'] = 'aeronave';
24 $array['tipo_operacion'] = 'denuncia';
25 $array['texto_formateado'] = 'Se realiza una denuncia';
26 $array['direccion_IP'] = '10.32.17.164';
27
28 try {
```

```
Output - auditoria (test:unit ServicioTrazas)
1..1
# obtenerTraza()
ok 1 - obtenerTraza - suAdminManager::Queda registrada correctamente la traza en la BD
 Looks like everything went fine.
```

Fig. 3.3 Resultado de la prueba unitaria al componente de Auditoría

Se puede llegar a la conclusión que la implementación del componente Trazas esta implementado correctamente y con la calidad requerida, ya que no se encontraron errores en la ejecución de las pruebas unitarias desarrolladas.

Conclusiones del capítulo

Se puede concluir que, durante la elaboración del presente capítulo, una vez generados los artefactos pertenecientes al flujo de implementación y la realización de las pruebas, se obtuvo como resultado el desarrollo de un componente con la calidad requerida, ya que las pruebas ejecutadas arrojaron resultados satisfactorios; dándole cumplimiento así al objetivo general de la investigación.

Conclusiones generales

Durante el desarrollo del trabajo se hizo un estudio de algunas de las diferentes herramientas destinadas al registro de trazas en los sistemas informáticos, se analizaron sus características, ventajas y desventajas; además de investigar sobre aplicaciones que se han desarrollado en la Universidad de las Ciencias Informáticas.

Se llevó a cabo la captura de los requerimientos del sistema y se generaron todos los artefactos y descripciones correspondientes al flujo de trabajo de análisis y diseño. Se realizó el diseño de las clases persistentes, el modelo de datos y el diagrama de secuencia, paquetes y clases del diseño con estereotipos web del requisito funcional identificado. Se analizaron también los patrones de diseño utilizados en la solución planteada. Además se diseñó el diagrama de implementación que dio paso a la programación de los componentes que forman parte de la aplicación.

Se considera entonces que se le ha dado cumplimiento a los objetivos propuestos para llevar a cabo la investigación, quedando como resultado el desarrollo de un servicio que permite realizar el registro de las trazas que son generadas por los subsistemas que integran a GINA. Para esto se llevaron a cabo una serie de tareas que contribuyeron al resultado final.

El desarrollo del componente Trazas del subsistema de seguridad suAdminPlugin aportará mayor seguridad a las aplicaciones que se encuentren bajo la arquitectura de Symfony, y específicamente a las implementadas por el Departamento de Soluciones para la Aduana, lo que se traduce en un mayor control de las acciones que pueden realizar los usuarios sobre los sistemas que utilicen el componente.

Recomendaciones

Luego de llevar a cabo el cumplimiento y desarrollo de los objetivos propuestos se ofrecen las siguientes recomendaciones:

- Continuar profundizando en el estudio de los temas de seguridad en aplicaciones Web.
- Continuar el desarrollo del módulo de Auditoría en el Departamento de Soluciones para la Aduana, para obtener los reportes y hacerle las auditorías a los mismos, a partir de las trazas que son registradas por el componente desarrollado en la presente investigación y así garantizar el registro de las operación realizadas en las aplicaciones que se encuentren utilizando el subsistema de seguridad suAdminPlugin.
- Integrar el componente realizado con los demás subsistemas pertenecientes al sistema GINA para su correcto funcionamiento.

Referencias bibliográficas

1. **Díaz Pupo, Susana y Morán Isla, Alcibíades.** *Desarrollo de un componente de trazas y módulo de reportes para el Sistema Nacional de Rehabilitación.* Universidad de las Ciencias Informáticas. Cuidad de la Habana : s.n., 2009. Clase de tesis inédita.

2. **Miranda Delgado, María Elena.** *Pendiente de Título.Tesis de Especialidad de Contabilidad.* Universidad de la Habana. Ciudad de la Habana : s.n., 2009.

3. IMPLANTACIÓN DE SISTEMAS INTEGRADOS EN CALIDAD Y MEDIOAMBIENTE.IMPLANTACIÓN DE LAS NORMAS UNE-EN ISO 9001 Y UNE-EN ISO 14001 EN UN SISTEMA INTEGRADO DE GESTIÓN. [En línea] [Citado el: 5 de noviembre de 2010.] http://www.scmempresa.com.

4. **123 Innovation Group, S.L.** AUDITORÍA SISTEMAS. [En línea] [Citado el: 15 de noviembre de 2010.] http://auditoriasistemas.com/auditoria-de-sistemas-informaticos/.

5. **Sacerio Martínez, Aida.** *Análisis, diseño e implementación del Framework de registro de eventos y su herramienta de seguimiento.* Universidad de las Ciencias Informáticas. Ciudad de la Habana : s.n., 2009. Trabajo de Diploma.

6. Trac: Gestión de Proyectos de Software. *Entorno de acs.* [En línea] 2005. [Citado el: 9 de mayo de 2010.] http://acsblog.es/?p=1752.

7. **Infotec.** *Administración del Proceso de Software. Categoría B3. Soluciones para la administración del conocimiento.* 2006.

8. Trac. *Descargar Trac.* [En línea] 2009. [Citado el: 29 de abril de 2010.] http://translate.google.com.cu/translate?hl=es&sl=en&u=http://trac.edgewall.org/wiki/TracDownload&ei=AMx0S6uHJ4XU8QajuaD0CQ&sa=X&oi=translate&ct=result&resnum=2&ved=0CBgQ7gEwAQ&prev=/search%3Fq%3Dtrac%26hl%3Des%26sa%3DG.

9. Open Populi. [En línea] 2008. [Citado el: 15 de mayo de 2010.] http://pruebas.openpopuli.com/index.php.

10. Open Populi. [En línea] 2008. [Citado el: 5 de mayo de 2010.] http://pruebas.openpopuli.com/manual/que_es_open_populi_framework.php.

11. Open Populi. [En línea] 2008. [Citado el: 5 de mayo de 2010.] http://pruebas.openpopuli.com/manual/logs_y_auditoria.php.

12. **SAP.** SAP Library. *The system Trace.* [En línea] [Citado el: 9 de noviembre de 2010.] http://help.sap.com/saphelp_nw70ehp1/helpdata/en/1f/8311784bc511d189750000e8322d00/frameset.htm.

13. **SAP.** SAP Library. *The System Log.* [En línea] [Citado el: 9 de noviembre de 2010.] http://help.sap.com/saphelp_nw70ehp1/helpdata/en/c7/69bcbaf36611d3a6510000e835363f/content.htm.

14. **Software.AG.** Software.AG. [En línea] [Citado el: 13 de noviembre de 2010.] http://documentation.softwareag.com/webmethods/cit80/appx/logviewer.htm..

15. **Potencier, Fabien y Zaninotto, François.** Logs de PHP. *Symfony 1.2, la guía definitiva.* 2008.

16. —. Logs de Symfony. *Symfony 1.2, la guía definitiva.* 2008.

17. **CORPORATION, R. S.** *Ayuda extendida de RUP (Rational Unified Process) Version 2003.06.00.65.* 2003.

18. *Proceso de Desarrollo y Gestión de Proyectos de Software.* Universidad de las Ciencias Informáticas : s.n., 2009.

19. **Alfaro, F. M.** *Herramientas Case.* INSTITUTO NACIONAL DE ESTADISTICA E INFORMATICA.

20. Visual Paradigm. [En línea] 2007. [Citado el: 13 de noviembre de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

21. Sitio Oficial de Visual Paradimg . [En línea] http://www.visual-paradigm.com.

22. **JACOBSON, I., BOOCH, G., RUMBAUGH, J.** *Proceso Unificado de Desarrollo de Software. .* 2000.

23. **Ruíz, F.** *Tecnología para la Gestión de Procesos de Negocios.* Escuela Superior de Informática. noviembre 2006.

24. **García, J.** IngenieroSoftware. [En línea] 7 de mayo de 2005. [Citado el: 15 de noviembre de 2010.] http://www.ingenierosoftware.com.

25. **Ferré Grau, X., Sánchez Segura, M. I.** ElQuintero. [En línea] 2008. [Citado el: 30 de octubre de 2010.] http://www.elquintero.net/Manuales/UML/umlTotal.pdf.

26. **Group, The PHP.** PHP. [En línea] 2001. [Citado el: 15 de enero de 2011.] http://www.php.net/.

27. **Javier Eguiluz.** Symfony.es. [En línea] [Citado el: 15 de enero de 2011.] http://www.symfony.es/.

28. **Martínez, Jenni Manso.** *Procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE.* Universidad de la Ciencias Informáticas : Facultad 15, 2010.

29. **Ivar Jacobson, Magnus Christerson, Patrik Jonsson, Gunnar Overgaard.** *Object-Oriented Software Engineering: A Use-Case-Driven Approach.* 1993. 84-7829-036-2.

30. **CHAVES, M. A.** *La Ingeniería de Requerimientos y su importancia en el desarrollo de proyectos de software.* Costa Rica : s.n., 2005. vol. VI, ISBN 1409-4746.

31. **Escalona, M.J. y Koch, N.** *Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo.* España, Alemania : s.n.

32. **Barrera, Liannis Soria.** *Requisitos_Software_GINA.* Universidad de las Ciencias Informáticas : s.n., 2010.

33. **IBM.** IBM. [En línea] [Citado el: 17 de Febrero de 2011.] http://www-01.ibm.com/software/solutions/soa/.

34. **SOAction.** SOAction. [En línea] [Citado el: 17 de Febrero de 2011.] http://soaction.sisorg.com.mx/definicion.html.

35. **Champion, Michael.** *Towards reference architecture for Web services. XML Conference and Exposition 2003.* Filadelfia : s.n., 2003.

36. **Potencier, F. and F. Zaninotto.** *Symfony: la guía definitiva.* 2008.

37. **GUERRA, I., DE FRUTOS, E.** Eduforge. [En línea] 12 de diciembre de 2007. [Citado el: 15 de octubre de 2010.] https://eduforge.org/.

Glosario de términos

Aduana: Oficina pública, establecida generalmente en las costas y fronteras, para registrar, en el tráfico internacional, los géneros y mercaderías que se importa o exportan, y cobrar los derechos que adeudan.

AGR: Aduana General de la República.

CADI: Centro de Automatización para la Información y la Dirección.

GINA: Gestión Integral Aduanera.

SUA: Sistema Único de Aduanas.

CASE: Ingeniería de Software Asistida por Computadora (Computer Aided Software Engineering).

Plugin: Pequeño programa que añade alguna función a otro programa, habitualmente de mayor tamaño. Un programa puede tener uno o más conectores. Son muy utilizados en los programas navegadores para ampliar sus funcionalidades.

Framework: 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.

Metodología: Es un proceso de software detallado que define con precisión los artefactos, roles y actividades involucradas.

Clase: Descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.

Diagrama: Representación gráfica de un conjunto de elementos. Visualizan un sistema desde diferentes perspectivas.

Especificación de requisitos: Captura los requerimientos de software para el sistema completo o una porción del mismo.