Universidad de las Ciencias Informáticas

Facultad 1

Título: Módulo de Generación y Almacenamiento de LOGS para el Subsistema de Auditoría del Sistema de Administración de Identidades

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

Autora: Jessica Bordón Ricardo

Tutores: Ing. José Carlos Correa Bautista

Ing. Yayneris Zambrana Hernández

La Habana, Junio de 2013

"Año 55 de la Revolución"

"Estoy convencido de que la mitad de lo que separa a los emprendedores exitosos de los que no triunfan es la perseverancia. Es tan difícil, pones tanto de tu vida en esto, hay momentos tan duros en que la mayoría se da por vencida, no los culpo, es muy difícil y consume gran parte de tu vida (...) A menos que tengas mucha pasión en lo que haces no vas a sobrevivir, vas a darte por vencido"

Steve Jobs

Declaración de Autoría

Declaro ser la única autora de la presente tesis y reconozco a la Facultad 1 y la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Jessica Bordón Ricardo

_____

(Autora)

Ing. José Carlos Correa Bautista

_____

(Tutor)

Ing. Yayneris Zambrana Hernández

_____

(Tutora)

A mi familia por apoyarme en todo momento y ser la razón de mi existencia.

A mis tíos Ana y Fernando, a mi padre Juan Carlos, a mi hermano Yoni, a mis primas Mary e Isa, y en especial a mi madre por ser la persona más importante en vida, por demostrarme su cariño y apoyo incondicional, por enseñarme a través de sus consejos a no desfallecer ni rendirme ante nada y siempre perseverar, hoy quiero darte las gracias por todo lo que me has ofrecido y agradecer a dios por tener una madre como tú.

A mi abuelito, donde quiera que se encuentre, por ser la luz de mis ojos, por enseñarme el valor de la familia, la grandeza de perdonar a quienes amamos y por hacerme la mujer que soy te llevo siempre en mi corazón.

A mi abuela por ayudarme a enfrentar la vida con sabiduría y nunca tener miedo de decir lo que pienso ante nadie.

A mis compañeros de estudio que han estado conmigo a lo largo de 5 años, que más que amigos somos una familia porque hemos compartido incontables experiencias que nos han hecho fuertes y nos han enseñado a estar siempre unidos.

Especialmente quiero expresar mis agradecimientos a alguien que ha sido una amiga incondicional desde que entré a la universidad, alguien que siempre está de mi lado y que me ha hecho reír en los momentos más tristes, alguien que no le importa los errores que cometa o las cosas que diga porque al final del día siempre me perdona, esa persona es Greysi y quiero agradecerle haber sido para mí, la hermana que nunca he tenido.

A mis amigas Islen, Yanet, Lily, Daylis, Leydis y Dayana por darme la oportunidad de compartir a su lado momentos que nunca olvidaré.

A mi tutor José Carlos y en especial a Pablo David por la ayuda que me han brindado con mi tesis.

A la memoria de mi abuelo por todo el amor que me profesó y por todo lo que soy.

A mi madre por ser la persona que más admiro y amo en el mundo.

A mi padre y mi hermano.

A mi abuelita del alma por ser la mujer más maravillosa del mundo.

A todos los que me quieren y velan por mí.

Para gestionar el acceso a la información de manera transparente, las empresas están incorporando Sistemas de Administración de Identidades (IDM por sus siglas en inglés), que ayudan a la automatización de sus procesos, pero debido a los riesgos de seguridad a los que se exponen a diario estos sistemas es de vital importancia la auditoría de la información generada por cada uno de sus componentes, asumiendo como misión principal, la detección de fraudes o errores informáticos a través de los registros de eventos, los cuales constituyen evidencias de los acontecimientos que tienen lugar dentro de los sistemas de información y las redes de la organización. La presente investigación tiene como principal objetivo desarrollar un módulo, para el subsistema de reportes y auditoría del sistema de administración de identidades perteneciente al centro CISED[1] ubicado en la Universidad de las Ciencias Informáticas, este módulo concentrará las políticas de generación y almacenamiento de logs de cada uno de los subsistemas que integran el IDM. La aplicación ha sido desarrollada bajo la tecnología .NET y empleando como metodología guía MSF, para el desarrollo del *software* ágil. La implantación del módulo permitirá registrar los eventos generados por cada uno de los subsistemas del IDM, como resultado de la ejecución automática de procesos internos o la interacción con otros sistemas o usuarios, permitiendo a través de la revisión de las salidas de los archivos logs, emprender las acciones necesarias ante la ocurrencia de un error del sistema o iniciar una investigación en el caso de un incidente de seguridad.

Palabras clave: administración de identidades, auditoría informática, logs.

[1] Centro de Identificación y Seguridad Digital

Índice General

Índice de Tablas

Índice de Figuras

Introducción

A fin de afrontar con éxito los nuevos retos que impone esta nueva era de los negocios que se proyecta altamente dinámica, cambiante y considerablemente competitiva, las empresas persiguen encaminar sus esfuerzos en mejorar el servicio a los clientes y maximizar el rendimiento de sus recursos mediante un adecuado sistema de información que integre y controle la gestión de todos sus departamentos con la rapidez y validez que el mercado mundial demanda (Martín, Justicia XXI.O, 2011), es esta precisamente la meta que se proponen alcanzar los Sistemas de Administración de Identidades (IDM por sus siglas en inglés).

La gestión de identidad global se está convirtiendo en una de las bases más importantes de la gestión óptima de los sistemas informáticos, la misma comprende programas capaces de cubrir las necesidades más significativas en cuanto a seguridad de una organización, permitiéndole además, la libertad de crecer tan rápido como le permita su negocio y reducir el coste de forma drástica. Para gestionar el acceso a la información de manera transparente, las empresas están incorporando soluciones IDM que ayudan a automatizar sus procesos (Borrmart.SA, 2005), fijando las responsabilidades en elementos tan imprescindibles como la calidad de los datos originales y la exactitud de los procesos, pero para determinar estas responsabilidades es necesario establecer controles que faciliten comprobar el cumplimiento de las directrices aprobadas, la validez de los datos y el grado de confianza que merecen.

Bajo estas circunstancias subyace la idea de establecer un proceso de auditoría informática, para el IDM, que lleve a cabo el análisis y la verificación de asuntos relativos a la planificación, control exhaustivo, eficiencia, seguridad y adaptación del servicio informático. A partir de las investigaciones emprendidas por el doctor Gonzalo Alonso Rivas[2] referentes a las auditorías informáticas, está comprobado estadísticamente que el 50% de los fraudes, en las empresas, son llevados a cabo por los propios trabajadores o por aquellas personas que mejor conocen la organización del centro.

Dados los riesgos de seguridad a los que se expone el sistema es de vital importancia la auditoría de la información generada por cada uno de los componentes que integran el IDM como resultado de la

[2] Original de España, Doctor Ingeniero Industrial, Master en economía y dirección de empresas. Es actualmente Profesor Titular de la Universidad Politécnica de Madrid.

ejecución automática de procesos internos o la interacción con otros sistemas o usuarios. La misión principal de la auditoría de la información es la detección de fraudes o errores informáticos mediante los controles oportunos sobre la información que se genera con el fin de evaluar la eficiencia de un proceso, un sistema, una persona, una organización u otro concepto que forme parte del entorno.

Los logs o registros de eventos constituyen una fuente importante de información dentro de cualquier organización para realizar la auditoría de la información. Un log es una evidencia de los acontecimientos que ocurren dentro de los sistemas de información y las redes de una organización, los cuales están compuestos por eventos de entrada y cada entrada contiene información relacionada con una acción u operación específica que se ha ejecutado en el sistema (Baryolo, 2012).

A partir del año 1996 Cuba comienza a dar sus primeros pasos en el ordenamiento de un trabajo continuo destinado a impulsar el uso y desarrollo de las tecnologías informáticas en el país (Oficina Nacional de Estadíticas (O.N.E), 2010). En la actualidad se ha evidenciado un notable crecimiento en la producción de *software*, lo cual implica un incremento en el interés por garantizar la calidad, seguridad y eficiencia del *software* que se produce.

La Universidad de las Ciencias Informáticas (UCI), es uno de los grandes pasos que ha dado la Revolución cubana en aras de consolidar y fomentar el desarrollo de la informática. Dentro de la universidad se encuentra el centro de Identificación y Seguridad Digital (CISED). Este centro ha desarrollado numerosos proyectos enfocados a la creación de soluciones relacionadas con la identidad de los usuarios.

Ejemplo de estas soluciones es el Sistema de Administración de Identidades que comprende procesos como la Autenticación, Autorización, Auditoría y Aprovisionamiento de usuarios. Este IDM no cuenta actualmente con un registro de los eventos generados por cada uno de sus subsistemas de forma sistemática y en tiempo real, como resultado de la ejecución automática de procesos internos o la interacción con otros sistemas o usuarios, lo cual provoca la pérdida de información relevante para la detección de incidencias negativas, anulando la posibilidad de realizar las acciones necesarias ante la ocurrencia de un error del sistema o iniciar una investigación en el caso de un incidente de seguridad.

Para el departamento de Seguridad Digital del centro CISED es de vital importancia la centralización de los logs generados por estos subsistemas, ya que constituyen elementos básicos de seguridad que ayudan a mantener el correcto funcionamiento del Sistema de Administración de Identidades.

Derivado de la anterior situación problémica surge el siguiente **Problema científico:**
¿Cómo proveer un mecanismo que garantice la generación y almacenamiento de logs de cada uno de los subsistemas del Sistema Administración de Identidades?
Partiendo de este problema se tiene como **Objeto de estudio** el proceso de auditoría en los sistemas informáticos.
Para dar solución a lo anteriormente planteado se propone como **Objetivo general:** Desarrollar un mecanismo para la generación y almacenamiento de logs mediante una aplicación que se integre al Sistema de Administración de Identidades y el **Campo de acción** está constituido por los procesos de generación y almacenamiento de logs en el Sistema de Administración de Identidades.

Del citado objetivo general se desglosan los siguientes **Objetivos específicos**:

- Realizar un análisis del estado del arte para la elaboración del marco teórico alrededor del objeto de estudio.
- Desarrollar el módulo de generación y almacenamiento del logs del subsistema de auditoría para registrar las acciones del IDM.
- Realizar pruebas al módulo implementado para corregir los errores y garantizar la calidad del mismo.

Para dar cumplimiento a los objetivos planteados se organizan las **tareas de investigación** siguientes:

1. Realización de búsquedas bibliográficas sobre lo que se conoce en la actualidad del proceso de centralización de logs.
2. Análisis de la metodología, las herramientas y tecnologías que apoyen el proceso de generación y almacenamiento de logs.
3. Diseño del módulo de generación y almacenamiento de logs.
4. Implementación del módulo de generación y almacenamiento de logs.
5. Realización de pruebas unitarias al módulo desarrollado.

Métodos científicos

Para el desarrollo de esta investigación y el logro de su objetivo, así como sus tareas específicas, se utilizan métodos teóricos y empíricos, a través de los cuales se obtiene una idea más clara y concisa de lo que se pretende realizar:

Métodos teóricos:

Histórico - Lógico: Se lleva a cabo un estudio del estado del arte de la problemática, así como un análisis de las ventajas y desventajas de cada una de las herramientas necesarias para la confección del módulo. Se investiga el proceso de generación y almacenamiento de logs en diferentes sistemas desde sus inicios, a partir del cual se obtienen los aspectos negativos y positivos del proceso en cuestión, determinando la necesidad de realizar un sistema que facilite este proceso.

Analítico – Sintético: Luego de una investigación acerca del tema, del estado del arte a nivel mundial y específicamente en la UCI, se sintetiza toda la información que se considere importante para el desarrollo del trabajo.

Justificación de la investigación:

A partir de la necesidad de resolver la problemática que ocupa a esta investigación se ha decidido desarrollar un Módulo de Generación y Almacenamiento de Logs para el Subsistema de Reportes y Auditoría que se integre al Sistema Administración de Identidades y concentre las políticas de generación y almacenamiento de archivos de registros de todos los eventos que ocurran en los subsistemas de Autenticación, Autorización y Aprovisionamiento de usuarios. La solución proveerá al Subsistema de Reportes y Auditoría la habilidad de monitorear las actividades del IDM a partir de la revisión de las salidas de los archivos logs donde se determinarán los eventos del sistema, estos elementos aportarán al sistema la información necesaria para emprender las acciones pertinentes a fin de corregir un problema o iniciar una investigación en caso de un incidente de seguridad.

Estructura de los Capítulos

Capítulo 1: Fundamentación Teórica: se realizará el estudio del estado del arte teniendo en cuenta sistemas similares que han incorporado el registro de eventos a sus procesos de auditoría. A partir de

dicha búsqueda se mostrarán los aspectos y características fundamentales que puedan ser útiles o sirvan de base a la propuesta de solución que se plantea. También se llevará a cabo un análisis de las herramientas, metodologías, lenguajes de programación y tecnologías que serán usadas durante el proceso de desarrollo de la aplicación.

Capítulo 2: Visión y Planificación: durante este capítulo se especificarán las funcionalidades del módulo, así como sus escenarios y requisitos de calidad de servicio. Se presentará además, el diseño de la propuesta de solución en el cual se encontrarán los elementos y aspectos necesarios para la posterior implementación del módulo.

Capítulo 3: Construcción y Prueba: se especificará la arquitectura del módulo así como el diagrama de clases y el diagrama lógico del centro de datos. También se presentará el modelo de base de datos, así como detalles de la implementación. Se realizarán las pruebas unitarias al módulo para validar y verificar la calidad del mismo, a través del método de caja negra.

Capítulo 1: Fundamentación teórica

1.1. Introducción

El desarrollo de los sistemas de información ha impulsado la integración de los IDMs a las grandes empresas, garantizando el buen flujo de información, la disponibilidad de los datos en el momento adecuado y la posibilidad de acceso desde cualquier punto.

Para avalar la eficiencia de los IDMs es de vital importancia mantener el control a través de un mecanismo que pueda recolectar las incidencias que ocurran en sus distintos componentes. Este mecanismo permitirá velar por el cumplimiento de las normas establecidas, la validez de los datos, así como el grado de confianza que merecen los mismos, con el fin de detectar los fraudes o errores que se puedan producir en el sistema y así lograr el éxito de la empresa.

El presente capítulo pretende realizar un estudio conceptual de los principales elementos que componen la base de la propuesta de solución a la problemática que enfrenta esta investigación.

1.2. Conceptos asociados al dominio del problema

Se muestra a continuación una serie de conceptos que aportarán información preliminar de todos los aspectos a tratar posteriormente.

Seguridad informática

Conjunto de herramientas destinadas a proteger los bienes o activos de una institución. La información así como la identidad digital de los usuarios, son los elementos más preciados de un sistema informático, por lo que se debe garantizar su completa seguridad. Para conservar de manera eficiente estos elementos, el sistema informático debe cumplir con las siguientes características (Galdámez, Pablo, 2010):

Confidencialidad: la información o los activos informáticos son accedidos solo por las personas autorizadas para hacerlo.

Integridad: los activos o la información solo pueden ser modificados por las personas autorizadas y de la forma autorizada.

Disponibilidad: los activos informáticos son accedidos por las personas autorizadas en el momento requerido.

No repudio: Capacidad del emisor de comprobar que cada mensaje fue enviado y de forma recíproca por el receptor.

La seguridad informática comprende el conjunto de medidas administrativas, organizativas, físicas, técnicas, legales y educativas dirigidas a prevenir, detectar y responder a las acciones que ponen en riesgo la confidencialidad, integridad y disponibilidad de la información que se procesa, intercambia, reproduce y conserva por medio de las tecnologías de información (Pfleeger, 2006). De acuerdo con el concepto introducido por el autor Antonio Villalón Huerta en su libro de Seguridad en Unix y Redes en el año 2002, la seguridad informática se define como: "*conjunto de características que indican que un sistema está libre de todo peligro, daño o riesgo*" (Huerta, 2002).

Existen diferentes definiciones del término seguridad informática. De estas definiciones la autora de la presente investigación se acoge a la ofrecida por el estándar para la seguridad de la información ISO/IEC 27001, que fue aprobado y publicado en octubre del 2005 por la *International Organization for Standardization* (ISO por sus siglas en inglés) y por la comisión *International Electrotechnical Commission* (IEC por sus siglas en inglés).

"*La seguridad informática consiste en la implantación de un conjunto de medidas técnicas destinadas a preservar la confidencialidad, la integridad y la disponibilidad de la información, pudiendo, además, abarcar otras propiedades, como la autenticidad, la responsabilidad, la fiabilidad y el no repudio*" (Security Advisor, 2005)

Sistemas de Administración de Identidades

Los sistemas de administración de identidades comprenden un determinado grupo de políticas y procesos de organización que permiten mantener el control del acceso a los sistemas de información, su principal objetivo es el control de acceso de los usuarios a recursos y aplicaciones en estricta conformidad con políticas y regulaciones estipuladas. En la actualidad los IDMs se han hecho cada vez más útiles a causa de la necesidad de las empresas de mantener la seguridad de la información, las bases de datos, las aplicaciones, pero sobre todo de los usuarios malintencionados y los sabotajes (Hitachi ID Systems, 2013).

La gestión de identidad y su infraestructura de apoyo fortalecen cada vez más su importancia, producto del esfuerzo de las empresas por reducir gastos administrativos, simplificar la experiencia del usuario y responder más rápidamente a las peticiones de recursos. De manera general aporta beneficios inmediatos

en el aumento de la productividad y eficiencia de servicios como la gestión y administración de usuarios, además de una notable reducción de errores y costes asociados a las tareas de aprovisionamiento (Novell, 2008).

A partir de los conceptos expuestos anteriormente la autora de la presente investigación define los sistemas de administración de identidades como una infraestructura técnica y organizativa que permite detallar, gestionar y administrar los atributos de identidad, comprende un grupo de procesos que permiten la implantación de sistemas de identificación y autenticación única, así como la gestión centralizada de las atribuciones de los usuarios. Entre estos procesos se encuentran la autenticación, la autorización, el aprovisionamiento y la auditoría.

Auditoría informática

La auditoría informática es el proceso mediante el cual se recoge, agrupa y evalúa todo tipo de evidencias para determinar si un sistema de información, trabaja adecuadamente, con los parámetros establecidos, mantiene la integridad y seguridad de los datos, llevando eficazmente los fines de la organización, es además, un examen que se lleva a cabo en base a diferentes, técnicas, procedimientos y herramientas, que son de gran apoyo para, analizar, evaluar, verificar y recomendar posibles mejoras, trayendo consigo seguridad y un trabajo informático de calidad en la empresa. Sus principales objetivos persiguen (Fazani, 2009):

- El control total de todo lo relacionado con la informática empresarial.
- El estudio de la eficiencia de los Sistemas Informáticos.
- La verificación del cumplimiento de los parámetros que se establecieron.
- La revisión eficaz de la gestión de los recursos informáticos.

A partir de las investigaciones emprendidas por el doctor Gonzalo Alonso Rivas en su libro “Auditoría Informática” se define la auditoría informática como: “*la revisión y evaluación de los controles, sistemas, procedimientos de informática y de los equipos de cómputo, su utilización, eficiencia y seguridad, a fin de que por medio del señalamiento de recursos alternativos se logre una utilización más eficiente y segura de la información que servirá para una adecuada toma de decisiones*” (Rivas, 2001).

Teniendo en cuenta los conceptos enunciados referentes a la auditoría informática, la presente investigación se acoge a los términos definidos por el doctor Gonzalo Alonso Rivas, ya que enmarcan de manera general el proceso de auditoría como un elemento básico para mantener la seguridad y el adecuado funcionamiento de los sistemas informáticos dentro de las organizaciones.

Archivos de registro (logs)

Un log es un archivo en el cual se almacena un registro de todos los eventos que se llevan a cabo en un determinado sistema durante un período de tiempo en particular. Los archivos de registro son comúnmente utilizados por el sistema operativo, las aplicaciones o los procesos, para el registro de datos o información sobre un evento. El almacenamiento de un log se realiza en formato estándar de manera que los archivos de registro generados un sistema puedan ser interpretados y desplegados por otro completamente diferente. La palabra log está estrechamente vinculada con el término evidencia digital ya que los logs pueden ser recolectados y analizados con herramientas y técnicas especiales (Oscar, 2010).

Microsoft Corporation define un log como: "*mensaje generado por el programador de un sistema operativo, una aplicación o un proceso en el cual se muestra un evento del sistema"* (Microsoft, 2007).

Nivel	Fecha y hora	Origen	Id. del evento	Categoría de la tarea
Información	04/04/2013 2:17:30	Microsoft Office ...	300	Ninguno
Información	03/04/2013 10:29:56	Microsoft Office ...	300	Ninguno
Información	02/04/2013 10:41:32	Microsoft Office ...	300	Ninguno
Información	02/04/2013 6:24:56	Microsoft Office ...	300	Ninguno
Información	02/04/2013 6:24:47	Microsoft Office ...	300	Ninguno

Figura 1: Ejemplo de Log generado por Microsoft Office en el sistema operativo Windows Seven.

A partir de los criterios anteriormente expuestos se definen los logs como archivos especiales del sistema que son capaces de almacenar información referente a cualquier anomalía ocurrida en el sistema o a la ejecución de programas y servicios internos, permitiendo la solución de problemas y evitando que vuelvan a ocurrir.

❖ Formato de los logs

Los ficheros logs contienen información de eventos internos ocurridos diariamente en los sistemas y

aplicaciones, la información registrada es solo legible por programas capaces de interpretarla y mostrarla en un entorno amigable, por lo que ha sido necesario estandarizar el formato de los logs para que puedan ser interpretados por una gran variedad de programas de análisis de registros.

El *Common Log Format National Center for Supercomputing Applications* (NCSA por sus siglas en inglés), también conocido como Formato de Registro Común NCSA, es un formato de archivo de texto estándar utilizado por los servidores web cuando se generan archivos de registros. Este patrón de registro tiene como objetivo establecer una línea base para el formato de los logs, a fin de garantizar su interpretación por la mayor parte de los sistemas de análisis de eventos.

Los campos de cada línea de los logs generados en este formato están separados por espacios y poseen la siguiente sintaxis (International Business Machines Corporation(IBM), 2013):

cliente	ident	usuario	fecha	petición	status	bytes	referer	user-agent

- ✓ **cliente**: dirección ip del ciente.
- ✓ **ident**: en este campo siempre aparece un guión (-), se conserva por razones de compatibilidad.
- ✓ **usuario**: el identificador de usuario.
- ✓ **fecha**: la fecha y hora de la petición, en el formato: [día/mes/año:hh:mm:ss zona].
- ✓ **petición**: la primera línea de la petición recibida del cliente.
- ✓ **status**: el código de estado devuelto por el servidor en respuesta a esta petición.
- ✓ **bytes**: número de bytes enviados por el servidor, incluidas las cabeceras HTTP.
- ✓ **referer**: la URL de la página que hacía referencia a la solicitada por el cliente.
- ✓ **user-agent**: cadena de texto que identifica el navegador empleado por el cliente.

1.3. Análisis de soluciones similares existentes

La administración de identidades es, sin duda, un elemento crítico que permite a las empresas mejorar su eficiencia y disminuir los costos de operación. Uno de los elementos principales que tienen los IDMs, en términos de seguridad, es un registro de las actividades de sus componentes internos y redes. Estos registros permiten detectar ataques, intrusiones o errores que ocurran en el sistema, ya que el análisis de

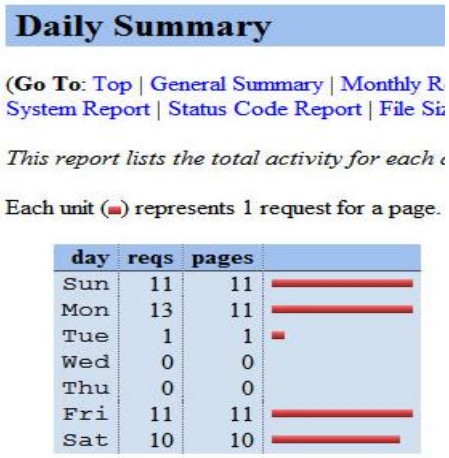
los registros facilita la ubicación exacta del problema así como una serie de datos que permiten emprender las acciones necesarias para corregirlo.

A continuación se muestran un grupo de aplicaciones que utilizan la generación y el almacenamiento de logs como un elemento primario para la seguridad de los datos.

Sistema Analog: Es un sistema para la generación de logs específicamente para servidores, a pesar de que permite llevar un registro de eventos, centra sus funcionalidades en los reportes y las estadísticas de los registros almacenados, estas estadísticas permiten obtener:

- Las páginas más populares.
- La procedencia de cada visita.
- Códigos de respuesta de errores del servidor.
- Información de archivos.
- Uso del tráfico.

Una de las características principales es que funciona en cualquier plataforma o sistema operativo, es escalable y altamente configurable, se pueden realizar reportes con los datos en más de 32 lenguajes incluido el español, es muy rápido, ofrece estadísticas detalladas sobre el uso de un servidor web, entre otras opciones (Analog, 2010).



Daily Summary

(**Go To**: Top | General Summary | Monthly R
System Report | Status Code Report | File Si

This report lists the total activity for each

Each unit (▬) represents 1 request for a page.

day	reqs	pages	
Sun	11	11	▬
Mon	13	11	▬
Tue	1	1	▬
Wed	0	0	
Thu	0	0	
Fri	11	11	▬
Sat	10	10	▬

Figura 2: Representación de las estadísticas del sistema Analog.

Sistema Catrian: El servicio gestionado de análisis y correlación de logs de Catrian permite explotar la información que se almacena en los logs de manera sistemática generando informes o alarmas de fácil

comprensión y uso. Este sistema recoge en tiempo real los logs de los diferentes elementos, los analiza, establece las correlaciones entre ellos que se consideren apropiadas, y genera los informes y alarmas oportunas (Catrian, 2010).

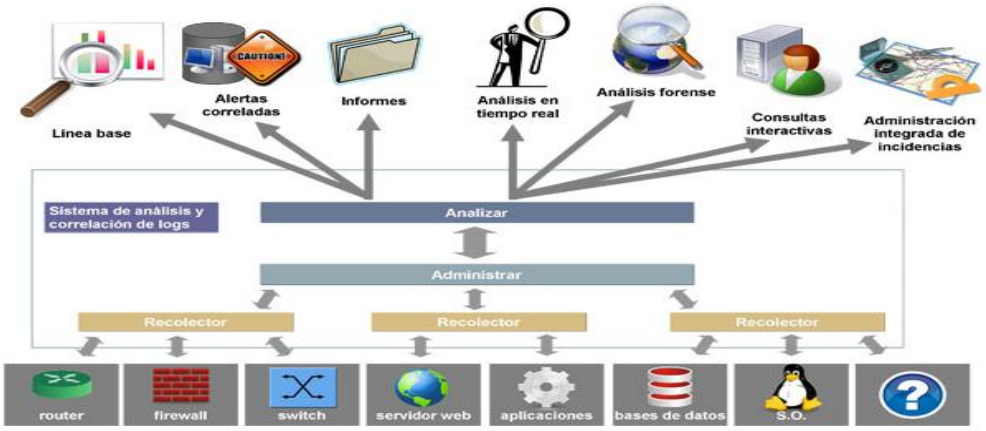


Figura 3: Servicio de análisis y correlación de logs del sistema Catrian.

Sistema Kiwi Syslog: Es uno de los servidores de logs más confiables basados en windows. Ofrece su solución basada en una amplia gama de características para la recepción, registro, visualización, alertas y reenvío de mensajes syslog, además ofrece mensajes de captura SNMP de dispositivos de red como enrutadores y *switches*. Entre sus principales características se encuentran:

- Permite ver los datos de syslog desde cualquier punto de la red a través de acceso web.
- Filtra los mensajes y crea alertas avanzadas con el procesamiento de secuencias de comandos.
- Incluye un calendario de archivo y registro de mantenimiento mediante el uso de archivos de registro automatizado.
- Permite ver los mensajes de syslog en varias ventanas al mismo tiempo.
- Automáticamente realiza acciones basadas en alertas, incluyendo el envío de correo electrónico, el reenvío de mensajes, alarmas audibles, envío de mensajes de captura SNMP.
- Conserva la dirección IP original de los mensajes reenviados como una de las múltiples opciones de reenvío avanzado.
- Reenvía mensajes de registro de sucesos de los servidores de windows al servidor Syslog Kiwi utilizando el Log Forwarder incluido para windows.

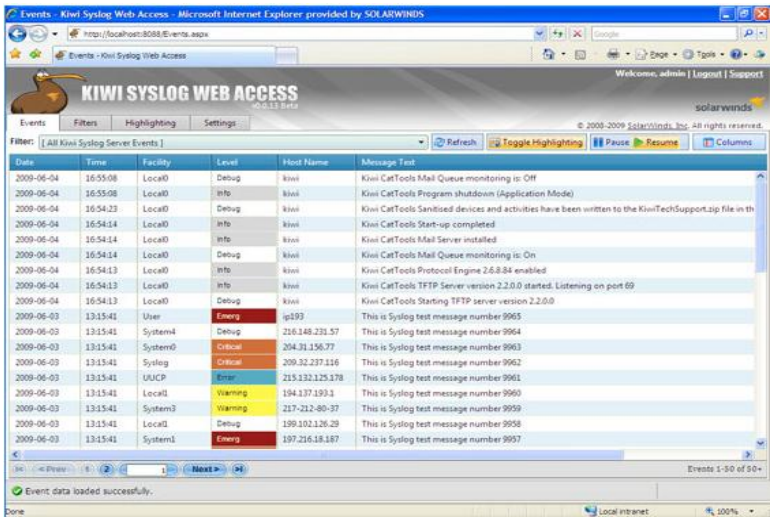


Figura 4: Interfaz de configuración de Kiwi Syslog

Sistemas similares en la UCI

Actualmente se encuentra en desarrollo una propuesta de solución para la gestión de controles de seguridad en la UCI, esta propuesta fue publicada por la revista de Ingeniería Electrónica, Automática y Comunicaciones (RIELAC) en enero del año 2013. Sus autores son Raydel Montesino Perurena[3], Walter Baluja García[4], Joelsy Porvén Rubier[5]. Dicha propuesta tiene como tema: "*Gestión automatizada e integrada de controles de seguridad informática*" y propone un modelo para la gestión automatizada e integrada de controles de seguridad informática, basado en sistemas de gestión de información y eventos de seguridad (SIEM), que posibilita aumentar la efectividad de los controles implementados y disminuir la complejidad de la gestión de la seguridad de la información.

[3] Graduado de Ingeniero en Telecomunicaciones y Electrónica en el año 2003 en el Instituto Superior Politécnico José Antonio Echeverría (ISPJAE). Actualmente es profesor e investigador de la Universidad de las Ciencias Informáticas (UCI), donde ha ocupado el cargo de Director de Seguridad Informática en los últimos siete años.

[4] Graduado de Ingeniero en Telecomunicaciones y Electrónica en el IPSJAE en el año 1997. Máster en Telemática (2000) y Doctor en Ciencias Técnicas (2006) por la misma universidad. Profesor auxiliar del departamento de Telecomunicaciones y Telemática del ISPJAE. Vicerrector del ISPJAE.

[5] Graduado de Ingeniero en Automática en el año 2003 en el Instituto Superior Politécnico José Antonio Echeverría (ISPJAE). Actualmente es profesor e investigador de la Universidad de las Ciencias Informáticas (UCI), donde se ha desempeñado como Especialista General en el área de los servicios telemáticos

Se define el concepto de automatización en el contexto de la seguridad informática y se determinan los controles que pueden ser automatizados. Como parte de la investigación se seleccionan un grupo de indicadores que permiten medir de forma automatizada la efectividad de los controles, se propone además una guía para la aplicación del modelo propuesto y se describe una posible implementación del mismo utilizando el sistema SIEM de *software* libre OSSIM.

A continuación se presenta la estructura para la Gestión automatizada e integrada de controles de seguridad informática:

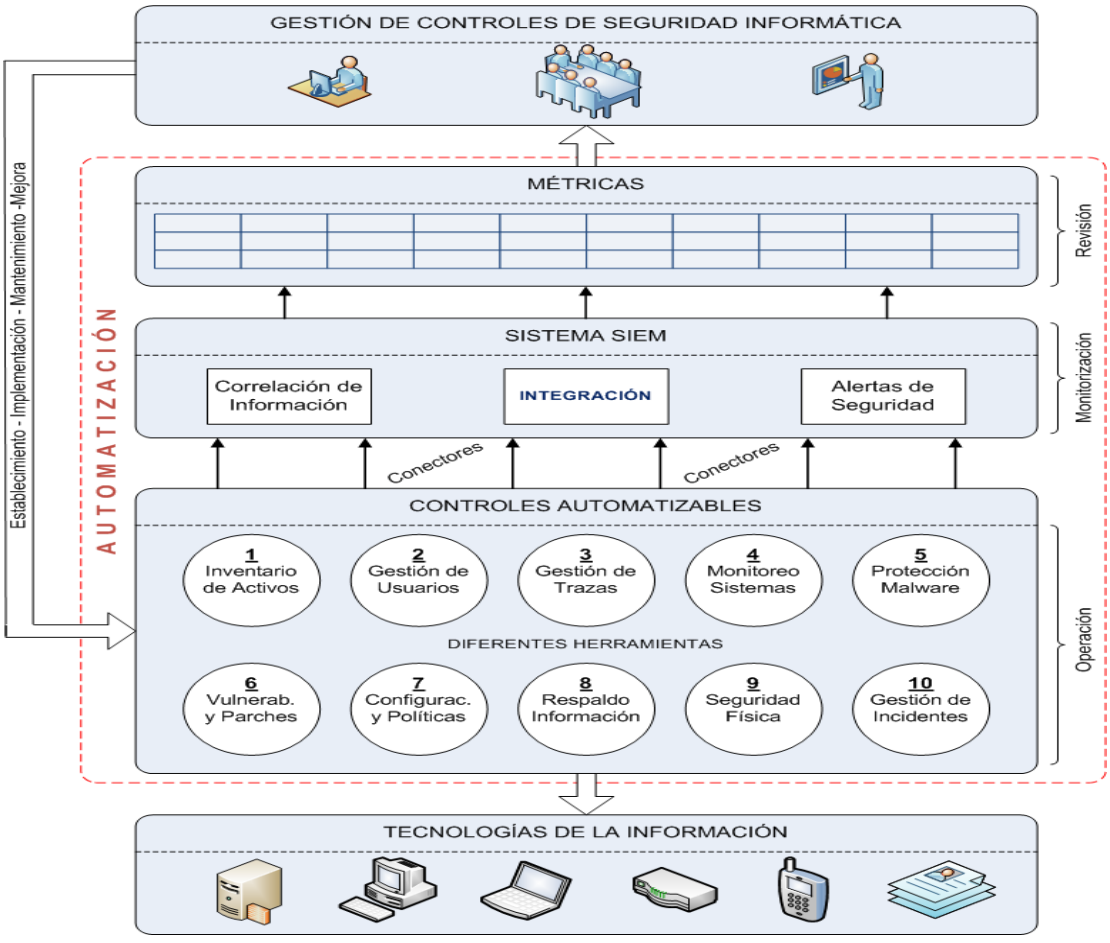


Figura 5: Modelo para la gestión automatizada e integrada de controles de seguridad informática

El modelo de gestión automatizada e integrada de controles de seguridad informática propone entre sus componentes la gestión de trazas o archivos de registro, con el objetivo de almacenar y conservar por el tiempo establecido, en una localización centralizada, las trazas de las aplicaciones, sistemas operativos y

diferentes dispositivos; donde se registre la actividad de los usuarios, errores, conexiones de red y otros eventos de seguridad en general (Joelsy P, 2013).

1.4. Inconvenientes de los sistemas analizados

Todos los sistemas analizados anteriormente realizan el proceso de generación y almacenamiento de logs como elemento esencial para la preservación de la información con la que se trabaja dentro de una empresa. A partir del estudio de estas soluciones se concluye que:

- Presentan similitudes en cuanto al proceso de generación de eventos.
- No contienen un mecanismo que permita modificar en menor medida los subsistemas del IDM.
- Centran la mayor parte de sus funcionalidades en la generación de reportes y estadísticas una vez almacenados los logs, elementos que no forman parte de esta investigación.

1.5. Herramientas para la recolección de logs

A fin de garantizar la seguridad de los sistemas informáticos se han desarrollado numerosas herramientas para recolectar los eventos que se generan dentro de las aplicaciones. Muchas de estas herramientas se especializan en la recolección de tipos específicos de eventos, mientras otros son capaces de trabajar con varios formatos de logs. Algunos de dichos sistemas han sido utilizados en Cuba y en la UCI específicamente. Actualmente los grupos de desarrolladores de sistemas de recolección de eventos han implementado *software* en las tres formas posibles en que se puede desarrollar un sistema de este tipo: Agente, Servidor y Cliente-Servidor. A continuación se brinda una descripción de los más importantes:

Agente o Cliente: Estos tipos de sistemas de recolección de logs son instalados en un computador y realizan la función de recoger los eventos ocurridos en el mismo para su almacenamiento y posterior análisis. En este grupo existen sistemas que brindan la posibilidad de enviar la información hacia un servidor que sea compatible con este, aunque existen otros que no cuentan con esta funcionalidad, por lo que solamente son usados para monitorear los logs ocurridos en un solo computador. Ejemplo de estos sistemas son:

❖ **Logwatch**

Existen un gran número de registros que son almacenados en los servidores, estos pueden contener información vital para el servidor web, el servidor de correo, las bases de datos, y otros sistemas y servicios importantes. Aunque los registros pueden ser consultados periódicamente, la tarea consume gran cantidad de tiempo y no siempre facilita la identificación de los problemas que ocurren (Source Forge, 2011). Logwatch es un programa de *software* libre y totalmente gratuito que permite personalizar el sistema de análisis de los registros. Este sistema se encarga de recoger la información de cada registro durante un período de tiempo previamente indicado y muestra toda la información referente a los registros seleccionados.

❖ **Lasso**

El proyecto Lasso es un *software* de código abierto basado en Windows diseñado para recopilar los registros de eventos de Windows, incluyendo registros de aplicaciones personalizadas. Esta herramienta fue desarrollada por el grupo LogLogic[6] en el año 2002 y actualmente cuenta con un importante grupo de clientes en todo el mundo. Lasso cuenta con muchas ventajas entre ellas que es de código libre como todas las herramientas desarrolladas por el grupo LogLogic, además fue implementado para enviar los eventos recopilados hacia un servidor Syslog compatible como Syslog-ng.

Servidor: Los servidores constituyen los elementos más importantes dentro del proceso de generación y centralización de los archivos logs ya que son los encargados de recibir la información enviada por los agentes que sean compatibles con ellos acerca de la ocurrencia de eventos en las computadoras donde estos estén instalados. A continuación se muestran un grupo de sistemas desarrollados para la recolección de logs como servidores.

❖ **Servidor SPLUNK**

Splunk es una aplicación web muy popular para Linux que brinda a los administradores de redes una amplia visión de los datos contenidos en los ficheros logs. Esta herramienta fue implementada para hacer

6 *LogLogic* es una compañía de tecnología que se especializa en la gestión de la seguridad, informes de cumplimiento, y los productos de Operaciones de TI.

la función de servidor de diferentes clientes en la recopilación de eventos. Ha sido desarrollada para recibir y asimilar la información recogida por un gran número de agentes como los de SNARE y Syslog. Permite además, indexar datos en cualquier formato enviado desde cualquier fuente en tiempo real, incluyendo logs, configuraciones, scripts, código, mensajes, alertas, reportes de actividades, desde todas las aplicaciones, servidores y dispositivos (Tech Republic, 2013). Admite buscar, navegar, alertar y hacer reportes de toda la información en tiempo real a través de una interfaz web desarrollada en AJAX. El equipo desarrollador de Splunk cuenta con una comunidad donde se comparten conocimientos y soluciones a través de SplunkBase.com, el sitio donde se realiza intercambio de criterios sobre la herramienta.

Cliente – Servidor: El sistema más completo es el que cuenta con clientes y servidor. Actualmente existen grupos desarrolladores que cuentan con su propio cliente y servidor, los ejemplos de este tipo de sistemas se muestran a continuación.

- **GFI Events Manager**

GFI Events Manager es un sistema muy completo que cuenta con facilidades para los usuarios como se detallan a continuación:

- ✓ Recolecta los logs generados por todas las computadoras y servidores que existan conectados a la red, ya sean de aplicación, sistema, seguridad y sucesos de servidores DNS a través de su propio agente, además de que en tiempo real detecta posibles violaciones o ataques, alertando a los administradores.
- ✓ Es capaz de realizar una copia de respaldo de la información y a la vez eliminarla de todas las computadoras automáticamente.
- ✓ Muchos sistemas de este tipo realizan la recopilación de la información máquina a máquina, pero el GFI Events Manager tiene una ventaja y es que lo realiza en toda la red, haciendo informes y filtrando los eventos.
- ✓ Está integrado a un Servidor Syslog que se encarga de recoger automáticamente los logs enviados por los agentes Syslog.
- ✓ Cuenta con una base de datos centralizada en la cual se pueden hacer filtrados y análisis de toda la red de forma personalizada, a intervalos deseados y en tiempo real.
- ✓ Es capaz de enviar alertas en tiempo real vía *e-mail* a uno o más destinatarios.

- ✓ Es de tipo propietario (GFI , 2013).

❖ **Nxlog**

Nxlog es una solución modular de alto rendimiento de gestión de registro con soporte multi-plataforma. En teoría, es similar a syslog-ng o rsyslog, pero no se limita a sólo Unix / syslog. Puede recopilar los registros de archivos en diversos formatos, recibirá los registros de la red de forma remota a través de UDP, TCP o TLS / SSL en todas las plataformas soportadas. Los registros obtenidos se pueden almacenar en archivos, bases de datos, o reenviados a un servidor de registro remoto utilizando varios protocolos. Un concepto clave en nxlog es ser capaz de manejar y preservar los registros estructurados. Posee además un potente filtrado de mensajes, registro de reescritura, y capacidades de conversión (NXLOG Community Edition,2012).

❖ **Rsyslog**

Rsyslog es una herramienta para la recolección centralizada de logs nativa de Unix que posee una arquitectura cliente-servidor. A continuación se detallan sus características más avanzadas:

- ✓ **Fuentes de entrada**: rsyslog tiene soporte para fuentes de entrada Unix, UDP, TCP, RELP, RFC 3195/BEEP y registro de eventos de Windows, a través de un *software* de registro de eventos de Windows como EventReportes o MonitorWare.
- ✓ **Filtrado de mensajes**: permite el filtrado de mensajes Syslog y la prioridad, filtrado de host, aplicación, de contenido de mensaje para envío de direcciones IP, posee además la capacidad de filtrar en cualquier otro campo del mensaje incluyendo las sub-cadenas. Comprende soporte para filtros reutilizables y la capacidad de utilizar expresiones regulares en filtros.
- ✓ **Salida de base de datos compatibles**: rsyslog permite el envío de mensajes a una base de datos a través del módulo de salida omlibdbi que utiliza la librería libdbi con soporte para MySQL, PostgreSQL, Oracle, SQLite, Microsoft SQL, Sybase, Firebird, Ingres, mSQL (Gerhards, 2010).

1.5.1. Herramientas para la recolección de logs en Cuba

A partir de las investigaciones realizadas se ha comprobado que no existe en Cuba un amplio desarrollo de herramientas para el tratamiento de los archivos de registros, sin embargo es importante destacar los

grandes avances que ha tenido la Revolución cubana en el campo de la informática y principalmente en el área de seguridad donde se hace necesario mantenerse a la par del resto del mundo a fin de garantizar la seguridad y calidad del *software* que se produce. La UCI es una institución que ha desarrollado trabajos sobre el tema de la gestión de la seguridad de redes y específicamente en la recolección de logs. A continuación se detallan un grupo de sistemas que evidencian el uso de herramientas de recolección de logs en Cuba y en la UCI.

SALDI

SALDI o Sistema Analizador de Logs para Detección de Intrusos fue creado en la CUJAE (Ciudad Universitaria José Antonio Echeverría) en el año 2004, con el objetivo de aliviar las tareas de los administradores por medio de la administración centralizada de un sistema IDS distribuido y ayudar al trabajo reactivo a corto plazo del administrador. Este sistema es capaz de realizar análisis de eventos con diferentes intervalos de tiempo, crear y enviar reportes por correo, almacenar los reportes en una base de datos, y otras funciones.

Para su desarrollo se utilizaron como lenguaje de programación Perl, PgAdmin como aplicación gráfica para el manejo de la base de datos en PostgreSQL y Macromedia Dreamweaver para desarrollar la página web (García, 2004). La puesta en práctica de este sistema arrojó los siguientes resultados que ayudaron a incrementar la seguridad en la red de la universidad:

- Análisis de logs con diferentes intervalos de chequeo.
- Análisis con múltiples patrones.
- Crear y enviar reportes por correo electrónico.
- Guardar organizada y localmente los reportes.

Uso de Webalizer en el portal www.cuba.cu.

El portal cubano www.cuba.cu desarrolló un estudio para profundizar en cómo los usuarios utilizan sus contenidos y servicios y una vez conocido estos realizar un plan de medidas a corto, mediano y largo plazo para perfeccionarlo, que no sólo abarcara el diseño de interfaz e información sino también orientado a los servicios que se brindan en el sitio. Para ello se utilizó la herramienta Webalizer. La herramienta Webalizer analiza la bitácora, como también se le llama a los logs, del servidor web y realiza reportes de

los datos que contiene en forma de gráficas.

Aquí se pueden ver los sitios y archivos más accedidos desde la página, así como los buscadores más utilizados para acceder a dicho sitio. La información obtenida puede ser vista por el administrador del sitio en el navegador. Para dicho estudio se analizaron los logs generados por el portal entre septiembre del 2002 hasta agosto del 2003 y fue posible determinar la forma en que los usuarios utilizan el portal así como los temas que buscan y prefieren (Lic. Adrián Coutin, 2003).

1.5.2. Herramientas para la recolección de logs en la UCI

Actualmente existen en la UCI muy pocos trabajos elaborados en relación a la recolección de logs. La universidad utilizó, durante un período de tiempo relativamente corto, una herramienta conocida como Event Log Security Manager, su funcionamiento estaba basado en la instalación de servidores en cada uno de los docentes, los cuales se encargaban de recolectar los eventos que se generaban en las computadoras del área, pero el uso de esta herramienta fue suspendido ya que es de tipo propietaria. Durante el año 2007 el compañero Robmay García Fuentes[7] desarrolló una investigación acerca de las distintas herramientas de Syslog, logrando realizar algunas configuraciones poco profundas.

La recolección de logs de manera centralizada en la UCI se lleva a cabo a través de la herramienta Rsyslog para las configuraciones en los servidores de recolección de logs y la herramienta Nxlog para las configuraciones de envío de logs hacia los servidores en los clientes con sistema operativo Windows o Rsyslog como alternativa de configuración del cliente con sistema operativo Linux.

A partir del estudio de las diferentes herramientas de recolección de logs utilizadas en el mundo, Cuba y la UCI se concluye lo siguiente:

- Las herramientas de recolección de logs estudiadas de tipo cliente y servidor, de manera independiente, fueron descartadas ya que para el desarrollo de la presente investigación se hace necesario la utilización de una herramienta con arquitectura cliente-servidor.
- Teniendo en cuenta el análisis de las diferentes características de las herramientas de recolección de

[7] Especialista general de la Dirección de Laboratorios de la Universidad de las Ciencias Informáticas.

logs de tipo cliente-servidor se descarta el uso de:

- ✓ GFI Events Manager ya que es de tipo propietario.
- ✓ Rsyslog posee características similares a Nxlog solo que esta última no se limita únicamente a Unix sino que es multiplataforma y brindaría un modelo de recolección y centralización de logs más homogéneo al poder hospedar la herramienta tanto en cliente como en el servidor.

❖ Se elige para el desarrollo de la presente investigación la herramienta Nxlog ya que:

- ✓ Es multiplataforma con soporte para Linux BSD, HPUX, Solaris Microsoft Windows.
- ✓ Es ligero, eficiente y presenta una configuración más sencilla y natural que Rsyslog.
- ✓ Permite definir numerosos destinos de entrada así como salida para el envío o recepción de los logs.

1.6. Análisis de las herramientas, metodología y tecnologías a utilizar

Luego del análisis de los principales conceptos que engloban la investigación y el estudio de sistemas similares para la generación y el almacenamiento de logs existentes en el mundo, es de suma importancia determinar las herramientas, tecnologías y la metodología de desarrollo que serán empleadas en la elaboración del módulo.

1.7. Metodología Microsoft Solutions Framework (MSF)

Las metodologías para el desarrollo de *software* proveen un conjunto de procedimientos, técnicas y soporte documental que van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, mostrando además qué personas deben participar en el desarrollo de las actividades y qué papel deben tener, detallando toda la información que se debe producir como resultado de una actividad y las pautas necesarias para darle inicio, guiando a los desarrolladores hasta la realización de un nuevo *software*. Las metodologías pueden ser clasificadas en dos grandes grupos, las ágiles y las tradicionales.

Las metodologías ágiles tienen como objetivo diseñar los valores y principios que deberían permitir a los equipos de desarrollo producir *software* rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto, pretenden ofrecer una alternativa a los procesos de desarrollo de *software* tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una

de las actividades desarrolladas. Entre las metodologías ágiles, las más utilizadas en nuestro días son MSF, XP, SCRUMP y SXP (Canós, 2009).

Por otro lado se encuentran las metodologías tradicionales las cuales imponen una disciplina de trabajo sobre el proceso de desarrollo del *software*, con el fin de conseguir un *software* más eficiente. Para ello se hace énfasis en la planificación total de todo el trabajo a realizar y el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada.

Las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos apropiados cuando se trabaja en un entorno donde los requisitos no pueden predecirse o bien pueden variar.

Entre las metodologías tradicionales o pesadas las más manipuladas son: OPEN, METRICA 3 y RUP (Roberth G, 2007).

- **Una metodología de manera general está compuesta por:**
  - ✓ Cómo dividir un proyecto en etapas.
  - ✓ Qué tareas se llevan a cabo en cada etapa.
  - ✓ Qué restricciones deben aplicarse.
  - ✓ Qué técnicas y herramientas se emplean.
  - ✓ Cómo se controla y gestiona un proyecto.

- **Principales características de las metodologías**
  - ✓ Existencia de reglas predefinidas.
  - ✓ Cobertura total del ciclo de desarrollo.
  - ✓ Verificaciones intermedias.
  - ✓ Planificación y control.
  - ✓ Comunicación efectiva.
  - ✓ Utilización sobre un abanico amplio de proyectos.
  - ✓ Fácil formación.
  - ✓ Herramientas CASE.
  - ✓ Actividades que mejoren el proceso de desarrollo.

- ✓ Soporte al mantenimiento.
- ✓ Soporte de la reutilización de *software*.

❖ Microsoft Solutions Framework (MSF)

La presente investigación está enfocada en las metodologías ágiles de desarrollo de *software* específicamente en la metodología *Microsoft Solutions Framework for Agile Software Development* (MSF por sus siglas en inglés). Esta metodología es un escenario orientado al proceso de desarrollo ágil de *software* para la plataforma. NET y otras aplicaciones orientadas a objetos. MSF incorpora directamente prácticas para el manejo de calidad de servicio, como el rendimiento y la seguridad. También un enfoque de contexto-conducida para determinar cómo operar el proyecto. Este enfoque ayuda a crear un proceso de adaptación que supere las condiciones de entorno de la mayoría de los procesos ágiles de desarrollo de *software*, mientras se logran los objetivos establecidos en la visión del proyecto (Microsoft C. , 2007).

❖ Principios de MSF-Ágil

- ✓ Potenciar todos los miembros de un equipo.
- ✓ Potencias las Comunicaciones entre el equipo y con el cliente.
- ✓ Establecer una Visión compartida de los valores de negocio del proyecto.
- ✓ Asegurar una contabilización clara de las responsabilidades compartidas.
- ✓ Mantenerse ‘Agiles’, esperar cambios.
- ✓ Aprender de las experiencias.

❖ Fases de Desarrollo de MSF- Ágil

- ✓ **Visión y alcance:** El equipo y el cliente definen el alcance del proyecto. La fase culmina cuando el alcance y la visión del proyecto sean aprobados.
- ✓ **Planificación**: El equipo prepara las especificaciones funcionales, se realiza el proceso de diseño de la solución y se preparan los planes de trabajo, estimaciones de costo y cronogramas de los diferentes entregables del proyecto. La fase culmina con la aprobación del Plan de Proyecto.
- ✓ **Desarrollo**: El equipo realiza la mayor parte de la construcción de los componentes y se crea una solución de la arquitectura a establecer. Como resultado se obtiene una versión de la infraestructura del producto después de aplicarle las revisiones al código que se va generando.

- ✓ **Estabilización:** Se llevan a cabo las pruebas sobre la solución, que enfatizan el uso y operación bajo condiciones reales. El equipo se enfoca en detectar los errores, solucionarlos y preparar la solución para el lanzamiento.
- ✓ **Implantación o Despliegue:** El equipo implanta la tecnología y los componentes utilizados por la solución, estabiliza la implantación, apoya el funcionamiento y la transición del proyecto y obtiene la aprobación final del cliente. La fase culmina con el hito de Implantación completo.

❖ Roles de MSF-Ágil

- ✓ **Líder de Proyecto:** Responsable de entregar una aplicación que satisfaga los requerimientos en tiempo y dentro del presupuesto. Se encarga de planificar, monitorear y reportar el status de las iteraciones, así como identificar y mitigar los riesgos. Además desarrolla, mantiene y ejecuta el plan de comunicaciones.
- ✓ **Analista del Negocio:** El principal objetivo del analista del negocio es entender y comunicar la oportunidad de negocio para el sistema. Ellos trabajan con los clientes y otras partes interesadas para conocer sus necesidades y objetivos para traducirlos en definiciones de personas, escenarios y requisitos de calidad de servicio que el equipo de desarrollo va a utilizar para crear la aplicación, es además el responsable de proporcionar y gestionar los vínculos entre las funciones de desarrollo de otros usuarios.
- ✓ **Arquitecto:** El principal objetivo del arquitecto es asegurar el éxito del proyecto mediante el diseño de las bases de la aplicación. Esto incluye definir tanto la estructura organizativa de la aplicación como la estructura física de su despliegue. En estos esfuerzos, el objetivo del arquitecto es reducir la complejidad mediante la división del sistema en particiones limpias y sencillas.
- ✓ **Desarrollador:** El principal objetivo del promotor es implementar la aplicación como se especifica en el tiempo previsto. También se espera que ayude a especificar las características del diseño físico, el tiempo estimado y esfuerzo para completar cada característica, construir o supervisar la ejecución de funciones, preparar el producto para su implementación y proporcionar tecnologías necesarias.
- ✓ **Probador**: El objetivo principal del probador es descubrir y comunicar problemas con el producto que podrían impactar negativamente su valor. El probador debe entender el contexto del proyecto y ayudar a otros a tomar decisiones basadas en este contexto. Un objetivo clave para el probador

es encontrar y reportar los errores significativos en el producto. Una vez que se encuentra un error, también es trabajo del probador comunicar con precisión su impacto y describir las soluciones alternativas que podrían disminuir las consecuencias. El probador hace descripciones de errores y participa con todo el equipo en el establecimiento de normas de calidad para el producto.

- ✓ **Administrador de Versión:** El objetivo del administrador de versión es gestionar el lanzamiento del producto. El mismo coordina la liberación de las operaciones o de control de los medios de comunicación. Crea un plan de implementación y certificación de candidatos de liberación para el envío o distribución.
- ✓ **Administrador de Base de Datos:** El objetivo principal del administrador de base de datos es apoyar la creación de proyectos de bases de datos, así como el despliegue de la producción de cambios en el proyecto de base de datos. Esto se suma a la función tradicional de los administradores de base de datos de realizar día a día la administración y mantenimiento de los servidores de bases de datos.
- ✓ **Desarrollador de Base de Datos:** El objetivo principal del desarrollador de bases de datos es ejecutar todas las tareas de desarrollo de base de datos dentro de los plazos previstos. Es también responsable de la estimación de costos, control de la ejecución, características de la base de datos y proporcionar conocimientos a los otros desarrolladores en el equipo.

Flujos de trabajo

Tabla 1: Listados de flujos de trabajo y los roles que lo desempeñan.

No	Flujo de Trabajos	Roles
1	Definir la Visión del Producto.	Analista del Negocio
2	Crear Escenarios.	Analista del Negocio
3	Crear requerimientos de Calidad de Servicio.	Analista del Negocio
4	Planificar Iteraciones.	Líder de Proyecto
5	Crear Arquitectura de la Solución.	Arquitecto
6	Implementar Tareas de Desarrollo.	Desarrollador
7	Construir un Producto.	Desarrollador
8	Probar un Escenario.	Probador

9	Probar Requerimientos de Calidad de Servicio.	Probador
10	Corregir Bugs.	Probador
11	*Release* del producto.	Administrador de Versión
12	Guiar el Proyecto.	Líder de Proyecto

1.8. Nxlog

❖ Características de Nxlog

✓ **Multiplataforma**

Compatible con diferentes sistemas operativos tales como: Linux (Debian, Red Hat, Ubuntu), BSD, HPUX, Solaris y Microsoft Windows.

✓ **Arquitectura modular**

Contiene módulos cargables que se encuentran disponibles para proporcionar diferentes características y funcionalidades. Posibilita cargar los módulos que son necesarios en la configuración del procesamiento de registros. Un módulo se carga sólo si es necesario dando como resultado una pequeña huella de memoria. Los módulos tienen una API común, que permite a los desarrolladores escribir nuevos módulos y ampliar las funcionalidades de Nxlog.

✓ **Modo cliente-servidor**

Nxlog puede actuar como un cliente y / o un servidor. Puede recopilar los registros de los archivos locales y del sistema operativo para enviarlos posteriormente a un servidor remoto. Puede aceptar conexiones y recibir los registros desde la red y a continuación escribir estos en una base de datos o archivo dependiendo de la configuración.

✓ **Mensajes de log orígenes y destinos**

Además de leer y escribir en archivos de registro, Nxlog soporta diferentes protocolos en la capa de red y transporte, como TCP, UDP, TLS / SSL y Socket Unix Domain, puede además leer y escribir desde cualquiera de estas fuentes.

✓ **Arquitectura escalable multi-hilo**

La lectura de entrada, salida y procesamiento de logs se manejan de forma paralela. La arquitectura multi-hilo de Nxlog permite aprovechar al máximo los sistemas multi-core y multi-procesador, de hoy en día, al

máximo rendimiento, además permite procesar los mensajes de registro de miles de conexiones de red simultáneamente.

- ✓ **Evita pérdida de mensajes**

El uso de las características o facilidades que brinda Nxlog como son: el procesamiento en paralelo, el *buffering* y la priorización de entrada y salida pueden reducir la pérdida de mensajes al utilizar el protocolo de transporte UDP. Si el protocolo usado es TCP no habrá pérdidas de mensajes.

- ✓ **Rotación de logs**

Nxlog tiene incorporado un planificador que puede llevar a cabo la automatización de tareas como la rotación de registros o estadísticas del sistema, sin tener que utilizar un programador externo. Los archivos de registro se puede rotar de acuerdo a su tamaño, el tiempo o las condiciones, no hay necesidad de herramientas externas de rotación de registros. El módulo de lectura de entrada soporta scripts externos de rotación de logs que pueden detectar cuando un archivo de entrada es movido o renombrado.

❖ Arquitectura de Nxlog

Mediante la utilización de módulos de carga dinámica, la arquitectura de *plugin* de nxlog permite leer los datos de cualquier tipo de entrada, analizar y convertir el formato de los mensajes y luego enviarlo a cualquier tipo de salida, además hace posible utilizar diferentes módulos de entrada, procesador y de salida al mismo tiempo para cubrir todos los requerimientos del entorno de registro. El flujo de mensajes de registro generado a partir del uso de dicha arquitectura se muestra a continuación:

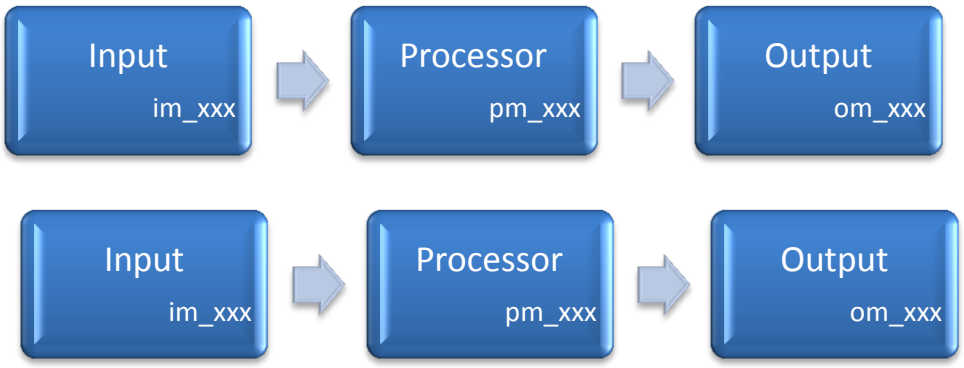


Figura 6: Arquitectura de Nxlog

1.9. Log4Net

Log4Net es un proyecto de código abierto con licencia Apache 2.0, es una librería para la generación de logs a partir del funcionamiento de una aplicación que permite identificar fallas, depurar y monitorear el funcionamiento de una aplicación. Entre los procesos que se llevan a cabo dentro del módulo de generación y almacenamiento de logs se encuentra la generación de logs, o archivos de registro, por las aplicaciones que componen el IDM (Apache Software, 2006).

La generación de los registros de archivos se lleva a cabo a través de Log4Net, herramienta que ayuda a los programadores a generar ficheros de salida con diferente información y trazas que permite depurar el desarrollo. La generación de registros, de manera general, puede traer múltiples inconvenientes asociados a la ofuscación del desplazamiento, pero Log4Net está diseñado para ser fiable, rápido, extensible y además dado que el registro no suele ser la actividad principal de una aplicación, Log4Net se esfuerza por ser fácil de entender y de usar (Apache Software, 2006).

- **Log4Net está disponible para varios Frameworks como son:**

  - ✓ Microsoft. Net Framework 1.0
  - ✓ Microsoft. Net Framework 1.1
  - ✓ Microsoft. Net Framework 2.0
  - ✓ Microsoft. Net Framework 3.5

- ✓ Microsoft. Net Framework 4.0
- ✓ Microsoft. Net Framework 3.5 Client Profile
- ✓ Microsoft. Net Framework 4.0 Client Profile
- ✓ Microsoft. Net Compact Framework 1.0
- ✓ Microsoft. Net Compact Framework 2.0
- ✓ Mono 1.0
- ✓ Microsoft Shared Fuente CLI 1.0
- ✓ CLI 1.0 Compatible

❖ **Log4net implementa las siguientes características:**

- ✓ Permite emitir logs a múltiples destinos como: archivos de texto, base de datos, Registro de Eventos de Windows, email, entre otros.
- ✓ Es posible organizar las entradas del log en categorías jerárquicas que se asemejan a espacios de nombres y clases del código fuente.
- ✓ Log4Net puede monitorear el archivo de configuración y volver a procesarlo al detectar una modificación, de esta forma no es necesario reiniciar la aplicación para cambiar el comportamiento del componente.
- ✓ Entrega facilidades para recabar información de contexto que queda disponible al momento de escribir en el log, por ejemplo, información sobre el hilo de ejecución, el proceso, el usuario.
- ✓ Genera un mínimo impacto en la performance de la aplicación, Log4Net fue diseñada para ser rápida.
- ✓ Es una arquitectura probada que ha sido portada a varias plataformas y lenguajes, muy flexible y extensible (Apache Software, 2006).

❖ **Log4Net define cinco conceptos básicos que constituyen la base de su comportamiento:**

- ✓ **Logger:** es el objeto utilizado en el código fuente para escribir las entradas en el log, el mismo implementa la interface ILog que tiene diversos métodos para escribir entradas.

- ✓ **Level (Nivel):** Log4net define cinco niveles para las entradas de logs, en orden de menor a mayor: Debug, Info, Warn, Error y Fatal. Esta clasificación es útil para determinar que entradas se escriben y donde se escriben.
- ✓ **Appender:** Representa el destino en el que se escribirán las entradas del log. Log4net permite configurar múltiples destinos y definir qué entradas se registran en uno o más destinos según su categoría, nivel y otros filtros más avanzados.
- ✓ **Filtro (Filter):** Los filtros están relacionados con los Appenders, permiten filtrar que entradas se registraran en el appender. Log4net permite filtros más avanzados tales como comparar una propiedad de la entrada con un valor específico y verificar la entrada contra un patrón de expresión regular.
- ✓ **Layout:** Define el texto que debe tener la entrada para ser registrada en un destino (Appender). Se define una plantilla indicando que información se escribe y en qué orden (Apache Software, 2006).

1.10. IDE de desarrollo Visual Studio 2010

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) específicamente para sistemas operativos Windows. Este IDE es capaz de soportar múltiples lenguajes de programación como son Visual C++, Visual C#, Visual J#, y Visual Basic .NET, al mismo tiempo que soporta entornos de desarrollo web como ASP.NET (Microsoft, 2013).

Microsoft Visual Studio 2010 Ultimate permite desarrollar sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET. Las versiones que se han desarrollado de Visual Estudio se extienden desde el año 2002 hasta nuestros días, cada una de ellas ha incorporado nuevos elementos y características que hacen al IDE más eficiente. Las principales características que presenta Visual Estudio 2010 y que lo diferencia de las versiones anteriores son las siguientes:

- ❖ **Depuración y Diagnóstico:** El IDE propone una valiosa herramienta de depuración que permite a los evaluadores archivar los errores enriquecidos y modificables para que los desarrolladores puedan reproducir el error del que se informe y el estado en que se encontró (Microsoft, 2013).
- ❖ **Herramienta de Prueba:** Visual Estudio 2010 incorpora todas las herramientas avanzadas de Microsoft para la realización de pruebas (Microsoft, 2013).

- **Arquitectura y Modelado:** Los diagramas por capas ayudan a garantizar el cumplimiento de la arquitectura y permiten validar artefactos de código con respecto a los diagramas (Microsoft, 2013).

Características de modelado y visualización

- **Generar código a partir de modelos:** Los modelos de UML pueden ayudarle a crear código y pruebas y describen la arquitectura y requisitos de un sistema (Microsoft, 2013).
- **Explorar código existente:** Los desarrolladores suelen emplear más tiempo en comprender el código existente que en escribirlo. Las herramientas de visualización de código de Visual Studio 2010 Ultimate pueden ayudarle a visualizar partes importantes del código, evaluar su flexibilidad e identificar las áreas de problema. Puede evaluar el costo potencial de los cambios propuestos más fácilmente si se sigue la traza de las dependencias entre las partes del código (Microsoft, 2013).
- **Usar y administrar los elementos del modelo:** Permite importar elementos de modelo de otras herramientas de modelado así como vincular a elementos de modelo a partir de elementos de trabajo ya que pueden ayudar a seguir y supervisar el progreso del trabajo en esos elementos (Microsoft, 2013).
- **Crear extender y validar los diagramas de capas:** Los diagramas de capas ayudan a visualizar la estructura de dependencias lógicas de la aplicación. Para asegurarse de que no se introduzcan accidentalmente esos cambios estructurales, el IDE permite validar el código con el modelo en cada protección (Microsoft, 2013).

1.11. Sistema Gestor de Base de Datos Oracle 11g

En la medida que en el mundo de las tecnologías las operaciones de negocio se vuelven más complejas, las demandas de implementación de cambios aumentan proporcionalmente, junto con los riesgos relacionados que deben eliminarse. Los profesionales de las tecnologías en nuestros días deben administrar y brindar una gran cantidad de información para sus usuarios con una mayor calidad de servicio y de manera oportuna (Oracle Corporation, 2013).

Al mismo tiempo las empresas están buscando reducir sus presupuestos en este sector y ofrecer mucho más valor de sus actuales inversiones. Oracle Database 11g es un Sistema Gestor de Base de Datos Relacional (o RDBMS por sus siglas en inglés de *Relational Data Base Management System*) que ofrece

una solución optimizada para aquellas implementaciones que requieren escalabilidad, confiabilidad y alto desempeño empresarial, ha sido desarrollado por Oracle Corporation y está considerado, en nuestros días, unos de los gestores de bases de datos más completos atendiendo a elementos como el soporte de transacciones, la estabilidad, la escalabilidad y el soporte multiplataforma.

Constituye además una base de datos que brindará exitosamente más información con una mayor calidad de servicio, reduciendo el riesgo de cambio dentro de las tecnologías y logrando utilizar los presupuestos de las empresas destinados a las tecnologías con mayor eficiencia. Las entidades pueden utilizar Oracle Database 11g para (Oracle Corporation, 2013):

- Reducir los costos de servidor.
- Reducir los requisitos de almacenamiento
- Mejorar el desempeño de los sistemas de misión crítica.
- Eliminar las redundancias del centro de datos.
- Simplificar los costos de tecnologías para las Empresas.

Ventajas de Oracle 11g

- **Máximo rendimiento de aplicación a disco y disponibilidad:** Supervisa automáticamente todo el entorno de base de datos y soluciona de forma proactiva los problemas en los distintos niveles antes de que se conviertan en emergencias (Oracle Corporation, 2013).

- **Mayor productividad del administrador y utilización de recursos:** Sistema más estratégico, más productivo, gestiona más bases de datos mientras aumenta el valor de la organización (Oracle Corporation, 2013).

- **Reduzca los fallos por errores humanos:** Asume el control del entorno informático solucionando la causa principal de los tiempos de inactividad imprevistos mediante prestaciones de automatización, configuración y gestión de cambios listas para usar (Oracle Corporation, 2013).

1.12. Protocolo Syslog

El protocolo Syslog es utilizado para el almacenamiento de manera local o remota de archivos de registro, el mismo define un único tipo de mensaje con tres campos que se transmite mediante el puerto UDP 514, esto permite concentrar los registros de múltiples máquinas en un único punto simplificando la labor de gestión al administrador siendo habitual que múltiples dispositivos lo soporten (Request for Comments 3164, 2001).

La presente investigación se ha acogido, para la utilización del protocolo Syslog, al estándar para el registro de archivos logs *Request for Comments* (RFC por sus siglas en inglés) 3164, el mismo ha documentado el funcionamiento del protocolo, su estandarización y de las diferentes capas de abstracción. A continuación se enuncian las principales características de los mensajes Syslog:

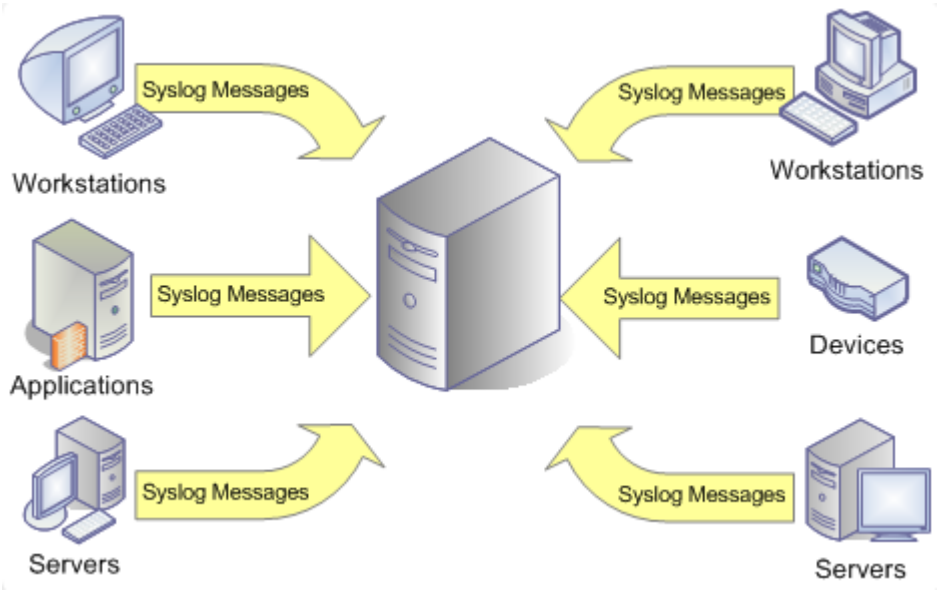


Figura 7: Esquema ideal de registro remoto de eventos usando el protocolo Syslog.

Severidad de los mensajes Syslog

Syslog se asienta sobre tres conceptos básicos: el dispositivo o aplicación generadora de eventos, el protocolo de comunicación y la aplicación o demonio que filtra dichos eventos y los escribe en un soporte según los deseos del administrador. Gracias a este diseño es posible recopilar información de múltiples

fuentes, ya sean aplicaciones locales o de múltiples máquinas, e incluso, dispositivos conectados a la red (Oscar, 2010).

Prioridad de mensajes Syslog

Los mensajes enviados a través de la red se componen por la prioridad, la cabecera y el texto. La prioridad es un número de 8 bits que indica tanto el recurso (tipo de aparato que ha generado el mensaje) como la severidad (importancia del mensaje), números de 5 y 3 bits respectivamente. Los códigos de recurso y severidad los decide libremente la aplicación, pero se suele seguir una convención para que clientes y servidores se entiendan (Oscar, 2010).

Campos de los mensajes Syslog

- **PRI:** Este campo está compuesto por 3,4 o 5 caracteres y debe estar rodeado de corchetes angulares. Comienza con el carácter "<" seguido de un número, el cual precede del carácter ">". El código utilizado en esta parte debe ser un ASCII de 7 bits en un campo de 8 bits. El número encerrado por los corchetes es conocido como la prioridad, la cual está compuesta por 1,2 o 3 enteros decimales. La prioridad representa a la vez la facilidad y severidad las cuales se encuentran codificadas numéricamente con valores decimales (Request for Comments 3164, 2001).

Aquellas facilidades que han sido asignadas son mostradas en la siguiente tabla, así como sus códigos numéricos:

Tabla 2: Código numérico de la facilidad de los mensajes Syslog (Request for Comments 3164, 2001).

Código numérico	Facilidad
0	Mensajes de kernel
1	Mensajes de nivel de usuario
2	Sistema de correo
3	Demonios del sistema
4	Mensaje de seguridad/autorización

5	Mensaje generado internamente por syslogd
6	Subsistema de impresora en línea
7	Subsistema de noticias de red
8	Subsistema UUCP
9	Demonio de reloj
10	Mensaje de seguridad/autorización
11	Demonio FTP
12	Subsistema NTP
13	Auditoria de eventos
14	Alerta de eventos
15	Demonio de reloj
16	Uso local 0
17	Uso local 1
18	Uso local 2
19	Uso local 3
20	Uso local 4
21	Uso local 5
22	Uso local 6
23	Uso local 7

- **HEADER (Cabecera):** La cabecera contiene dos campos: *TIEMSTAMP* y *HOSTNAME*. El *TIMESTAMP* va inmediatamente después del último “>” del PRI. Este corresponde a la fecha y hora local del dispositivo que transmite y su formato está definido como: “Mmm dd hh:mm:ss”. El *HOSTNAME* corresponde al nombre del dispositivo, la dirección ipv4 o la dirección ipv6.

- **MSG.** El mensaje o MSG tiene 2 campos *TAG* y *CONTENT*. El primero representa el nombre del programa o proceso que generó el evento el cual no debe exceder los 32 caracteres. El otro campo es libre de utilización y contiene los detalles del mensaje.

Cada prioridad de los mensajes tiene un indicador de severidad decimal. Estas severidades son descritas en la siguiente tabla:

Tabla 3: Severidad del mensaje según una escala de ocho valores (Request for Comments 3164, 2001).

Valor	Denominación		Descripción
0	Emergencia	EMERG	Sistema inutilizable
1	Alerta	ALERT	Requiere intervención inmediata
2	Crítico	CRIT	Condición crítica
3	Error	ERR	Condición de error
4	Peligro	WARN	Condición de peligro
5	Aviso	NOTICE	Funcionamiento normal pero con condiciones reseñables
6	Información	INFO	Mensajes informativos
7	Depuración	DEBUG	Mensajes de depuración de bajo nivel

1.13. Microsoft.Net

Microsoft.NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando durante los últimos años a fin de obtener una plataforma sencilla y potente para distribuir el *software* en forma de servicios que puedan ser suministrados remotamente y que puedan comunicarse y combinarse unos con otros de manera totalmente independiente de la plataforma, lenguaje de programación y modelo de componentes con los que hayan sido desarrollados. Ésta es la llamada plataforma.NET y a los servicios antes mencionados se les denomina servicios Web (Microsoft, 2013).

Para crear aplicaciones para la plataforma .NET, tanto servicios Web como aplicaciones tradicionales, Microsoft ha publicado el denominado kit de desarrollo de *software* conocido como .NET Framework SDK, que incluye las herramientas necesarias tanto para su desarrollo como para su distribución y ejecución.
Visual Studio.NET permite hacer todo lo anterior desde una interfaz visual basada en ventanas. Microsoft ha creado un conjunto de nuevas aplicaciones desarrolladas para ser utilizadas en la plataforma.Net, entre ellas destacan Windows.NET, Hailstorm, Visual Studio.NET, MSN.NET, Office.NET, y los nuevos servidores para empresas de Microsoft como son: SQL Server.NET, Exchange.NET, entre otros (José Antonio, 2008).

1.14. Lenguaje de programación C Sharp (C#)

C# es el lenguaje de propósito general diseñado por Microsoft donde sus principales creadores son Scott Wiltamuth y Anders Hejlsberg. Este lenguaje fue desarrollado para la plataforma .Net y aunque en la misma se pueden escribir códigos en múltiples lenguajes C# es el único que ha sido diseñado específicamente para ser utilizado en ella, de manera que programar en la plataforma usando dicho lenguaje es mucho más sencillo e intuitivo que hacerlo en otros programas. Por esta razón, se suele decir que C# es el lenguaje nativo de .NET que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo (José Antonio, 2008).

Características principales de C Sharp

- **Sencillez**

C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo (Microsoft, 2007):

- ✓ El código escrito en C# es auto-contenido, lo que significa que no necesita de ficheros adicionales a la propia fuente tales como ficheros de cabecera o ficheros IDL.
- ✓ El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina que se desea compilar, lo que facilita la portabilidad del código.
- ✓ No se incluyen elementos poco útiles de lenguajes tales como macros, herencia múltiple o la necesidad de un operador diferente del punto (.) acceder a miembros de espacios de nombres (::).

❖ **Modernidad**

C# incorpora, en el propio lenguaje, elementos que a lo largo de los años han ido demostrando ser muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++ hay que simular, como son (Microsoft, 2007):

- ✓ Un tipo básico decimal que permita realizar operaciones de alta precisión con reales de 128 bits.
- ✓ La inclusión de una instrucción *foreach* que permita recorrer colecciones con facilidad y es ampliable a tipos definidos por el usuario.
- ✓ La inclusión de un tipo básico *string* para representar cadenas o la distinción de un tipo *bool* específico para representar valores lógicos.

❖ **Orientación a componentes**

La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades, eventos o atributos (Microsoft, 2007).

❖ **Eficiencia**

En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de Java, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras y podrán usarse en ellas punteros de forma similar a cómo se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad de procesamiento muy grandes (Microsoft, 2007).

❖ **Compatibilidad**

Para facilitar la migración de programadores, C# no sólo mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes, sino que el CLR también ofrece, a través de los llamados *Platform Invocation Services*, la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos tales como las DLLs de la API Win32. La capacidad de usar punteros en código inseguro permite que se pueda acceder con facilidad a este tipo de funciones, ya que éstas muchas veces esperan recibir o devuelven punteros (Microsoft, 2007).

1.15. Programación Orientada a Aspectos (POA)

Gregor Kickzales[8], creador del paradigma de Programación Orientada a Aspectos (POA), define a un aspecto como:

"una unidad modular que se disemina por la estructura de otras unidades funcionales. Los Aspectos existen tanto en la etapa de diseño como en la de implementación. Un Aspecto de diseño es una unidad modular del diseño que se entremezcla en la estructura de otras partes del diseño. Un Aspecto de implementación es una unidad modular del programa que aparece en otras unidades modulares del programa" (Mariano Lorente López, 2008).

La Programación Orientada a Aspectos (POA) es un paradigma de programación relativamente reciente cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de incumbencias. Gracias a la POA se pueden encapsular los diferentes conceptos que componen una aplicación en entidades bien definidas, eliminando las dependencias entre cada uno de los módulos, además provee un mecanismo para separar cada uno de los componentes de una aplicación permitiendo hacer llamadas entre ellos de una manera más sencilla (DotNet, 2013).

- Evita que se duplique código, por ejemplo para hacer una auditoría cada vez que se modifique una de las propiedades de las clases, sería necesario duplicar la llamada a la clase de auditoría en cada una de ellas.
- Mejora el testeo de las aplicaciones permitiendo testear aspectos del sistemas de manera más independiente y diferenciar más rápidamente cuál de los aspectos implicados en un componente es el que falla.

La POA fue seleccionada para el desarrollo particular de este módulo, debido a la necesitada de encapsular las funcionalidades de un log dentro de un aspecto, ya que al estar en una sola unidad funcional resulta más sencilla su manipulación, aumenta la eficiencia en cuanto a la implementación y reduce ampliamente la necesidad de modificar el código en los subsistemas del IDM.

[8] Profesor del Departamento de Prácticas de Software de la Universidad de Columbia Británica en Canadá, es considerado en la literatura como el creador del paradigma de la Programación Orientada a Aspectos (POA).

1.16. Conclusiones parciales

- El análisis realizado sobre los principales conceptos asociados al dominio del problema permitieron una mejor comprensión del proceso de centralización de logs.
- El análisis detallado de sistemas similares de generación y almacenamiento de logs existentes, así como las herramientas para la recolección de logs, hicieron posible la selección de los elementos positivos de estas herramientas para ser desarrollados como parte de la propuesta de solución que se desea implementar.
- La metodología seleccionada permitirá guiar el proceso de desarrollo usando las herramientas y tecnologías definidas para la implementación de la propuesta de solución.

Capítulo 2: Visión y Planificación

2.1 Introducción

El proceso de desarrollo de un *software* informático requiere un conjunto de conceptos, modelos y mejoras prácticas de uso que controlan la planificación, el desarrollo y la gestión del proyecto en progreso, de acuerdo a la metodología del marco de trabajo de solución de Microsoft, *Microsoft Solution Framework* este conjunto de elementos, mencionados anteriormente, deben ser puntualizados de forma clara y sencilla para todos los involucrados en el proceso. Esta metodología plantea obtener una visión clara de lo que se desea desarrollar y propone realizar una planificación que guíe al grupo de trabajo hacia la construcción exitosa del sistema. Teniendo en cuenta la metodología propuesta durante este capítulo se generan los artefactos más representativos del sistema así como la descripción de los escenarios y requisitos de calidad de servicio para el módulo de generación y almacenamiento de logs.

2.2 Capturar Visión del proyecto

Iniciar un proyecto requiere el establecimiento claro de la visión del mismo. Capturar y comunicar esa visión central es el elemento más importante para mantener un proyecto enfocado. Esta visión puede cambiar como resultado del ambiente de negocios o del proyecto. Si el cambio se produce, es necesario volver a alinear el proyecto en relación a la nueva visión. Desde la visión se comienza a entender quiénes serán los usuarios. También se especifica si el proyecto es impulsado por fecha o por contenido. La visión y sus actividades relacionadas crean una base sólida sobre la cual puede ser construido un proyecto (Microsoft C. , 2007).

2.3 Descripción del módulo

El módulo de generación y almacenamiento de logs es el encargado de generar y almacenar todos los eventos que ocurren en los subsistemas de Autorización, Autenticación y Aprovisionamiento de usuarios. El proceso de generación se lleva a cabo a través de un componente proveedor de trazas que es incluido en los demás subsistemas. Una vez generados los eventos serán almacenados de manera local o remota haciendo uso del protocolo Syslog que permite concentrar los registros de múltiples máquinas en un único punto simplificando la labor de gestión del administrador, además hace posible que múltiples dispositivos lo soporten. Los registros almacenados serán posteriormente enviados a través de la red hacia un servidor

de logs para su almacenamiento central utilizando la herramienta NXLOG. Por último los logs serán almacenados en una base de datos centralizada.

El módulo de generación y almacenamiento de logs está diseñado para integrarse al Subsistema de Reportes y Auditoría permitiendo llevar a cabo un control exhaustivo a través del análisis de elementos como la planificación, la eficiencia, la seguridad y la adaptación del servicio informático, de manera general será posible, desde el subsistema, tener la supervisión y el control del IDM.

2.4 Propuesta de solución

La presente investigación propone como solución un módulo que ponga en funcionamiento las políticas de generación y almacenamiento de registros de los eventos que ocurren en cada uno de los subsistemas que componen el Sistema de Administración de Identidades. El módulo está compuesto por dos actividades fundamentales: la generación y el almacenamiento de logs. Para el proceso de generación de archivos se emplea la Programación Orientada a Aspectos (POA), mecanismo utilizado para la intersección de código como se muestra en la siguiente figura:

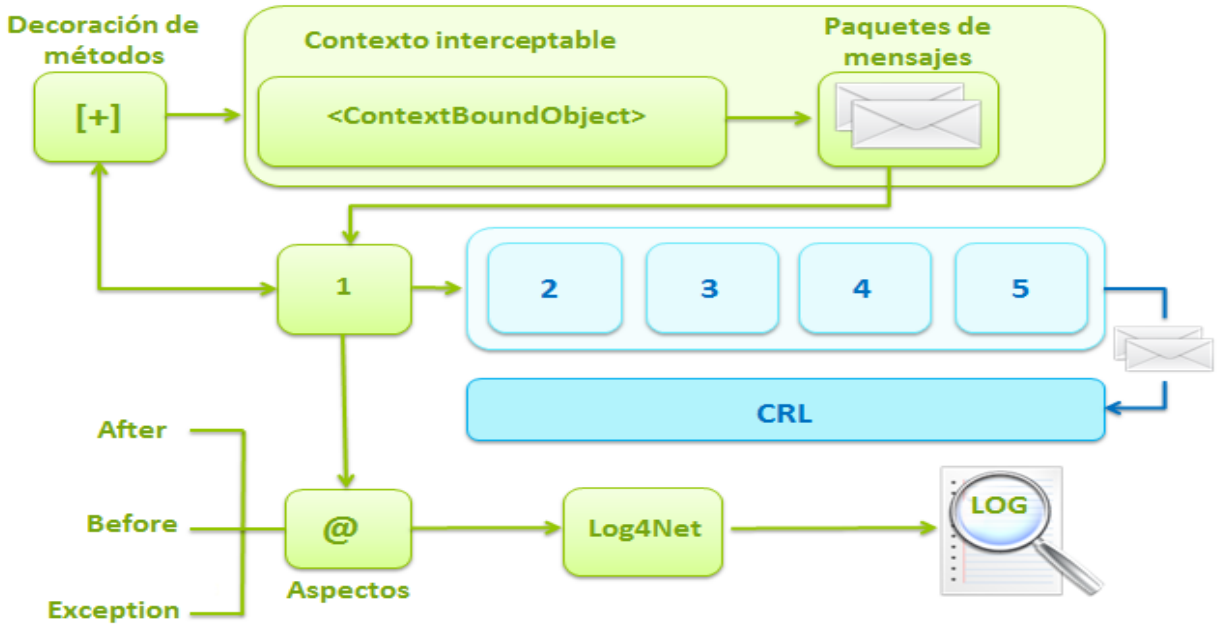


Figura 8: Mecanismo de Intersección de Código

Este mecanismo define las siguientes acciones para la intersección de código y la generación de logs:

- Todas las clases heredan de la clase *ContextBoundObject* y sus métodos están decorados definiendo cuando se deben ejecutar y que es lo que deben ejecutar.
- A través de la clase *ContextBoundObject* es posible definir un nuevo procesador.
- Los eventos de las clases interceptadas son almacenados en paquetes de mensajes y enviados al primer procesador definido.
- Todos los mensajes pasan al procesador donde se conoce si es necesario ejecutar alguna acción en el momento.
- Se hace la llamada al código que se debe ejecutar o aspecto.
- Cada aspecto, usando Log4Net, genera logs en formato Syslog.

Estas acciones se concentran en un componente proveedor de trazas que es integrado en cada uno de los subsistemas del IDM y registra los eventos ocurridos en tiempo real y de manera sistemática.

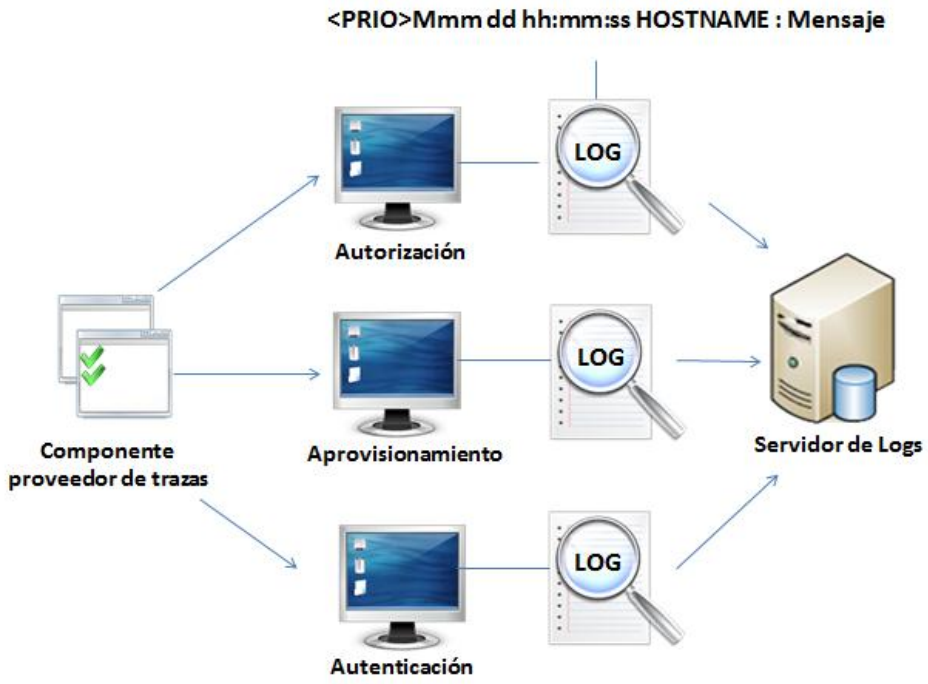


Figura 9: Proceso de generación de Logs

El almacenamiento de logs es realizado hacia el servidor con el uso del protocolo Syslog y posteriormente hacia la base de datos, para ambas acciones se utiliza la herramienta Nxlog. Esta herramienta posee una arquitectura cliente-servidor por lo que se configura como cliente en el sistema operativo Windows para el envío de los logs recolectados hacia el servidor. La instalación de Nxlog en el servidor se realizada en el sistema operativo Linux, ya que para el envío de logs hacia la base de datos es necesario utilizar la librería libdbi la cual es nativa de este sistema operativo.

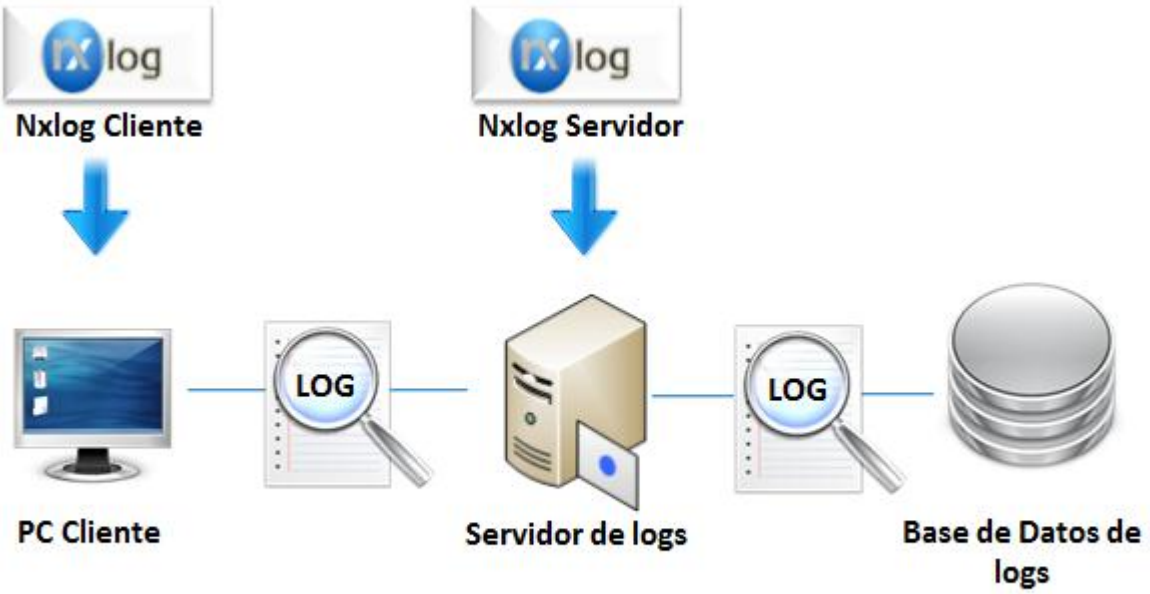


Figura 10: Proceso de Almacenamiento de Logs

Los eventos generados en cada uno de los subsistemas serán almacenados en el directorio *C:\Archivos de programas\IDM\logs\log.log.* El archivo log.log se define con una capacidad de almacenaje de 10MB[9], al alcanzar este límite se crea una nuevo archivo con el nombre log1.log. Los archivos logs comienzan a ser eliminados cuando existen en el sistema 10 archivos logs y la eliminación se realiza desde el archivo más antiguo.

[9] MB: Megabyte unidad de medida de cantidad de datos informáticos

2.5 Planificación

La siguiente fase en el proceso de desarrollo del *software* que propone la metodología *Microsoft Solutions Framework*, una vez definida la visión del proyecto, es la planificación. Durante esta fase se realiza la mayor parte de la planificación del proyecto, donde el equipo prepara los escenarios y requerimientos de calidad de servicio, se lleva a cabo el proceso de diseño de la propuesta de solución, así como el cronograma de los diferentes entregables del proyecto. A continuación se muestran los diferentes artefactos generados durante la fase de planificación.

2.5.1 Escenarios del módulo

Los escenarios consisten en describir parcialmente el comportamiento del módulo de manera que facilite la comprensión de la aplicación y su funcionalidad, los mismos se componen en un grupo de actividades con determinada información que complementan la descripción del sistema, entre estas actividades se encuentra listar los escenarios del módulo como se muestra a continuación:

❖ **Lista de escenarios**

- ✓ Capturar eventos en el módulo
- ✓ Registrar logs en formato Syslog
- ✓ Copiar archivos logs hacia el servidor
- ✓ Almacenar archivos logs en la Base de Datos

2.5.2 Priorización de escenarios

La prioridad es un atributo de cada escenario, se lleva a cabo a través de la identificación de los escenarios del sistema y la clasificación de su prioridad en “Alta” equivalente a cinco puntos, “Media” equivalente a cuatro puntos y “Baja” equivalente a tres puntos.

Tabla 4: Listado de Escenarios y sus Prioridades

No	Escenarios	Prioridad
1	Capturar eventos en el módulo	5

2	Registrar logs en formato Syslog	4
3	Copiar archivos logs hacia el servidor	3
4	Almacenar archivos logs en la Base de Datos	3

2.5.3 Plan de iteraciones

Desarrollar el *software* en iteraciones consiste en dividir el trabajo en etapas incrementales de modo que al final de cada iteración se disponga de un *software* al que se integrarán nuevas características al final de cada una de las etapas. Las iteraciones permiten obtener comentarios en una fase temprana para poder ajustar el curso del proyecto. La planeación de iteraciones comprende tareas como la toma de decisiones sobre la duración de proyecto, la cantidad del trabajo que el equipo puede realizar en ese tiempo estimado y definir qué trabajo debe ser incluido en cada iteración.

El proyecto ha sido dividido en dos iteraciones durante las cuales serán implementados todos los escenarios del módulo de generación y almacenamiento de logs. El proyecto tendrá una duración total de aproximadamente 12 semanas.

- **Iteración #1:** Se implementarán los escenarios de mayor prioridad del módulo, ocupando un tiempo total de 6 semanas.

- **Iteración #2:** Se implementarán los escenarios de mediana y baja prioridad del módulo, ocupando un tiempo total de 7 semanas.

Tabla 5: Planificación de los Escenarios

No	Escenarios	Prioridad	Riesgo	Esfuerzo(Días)	Iteración
1	Capturar eventos en el módulo	5	Alto	46	1
2	Registrar logs en formato Syslog	4	Medio	14	2
4	Copiar archivos logs hacia el servidor.	3	Bajo	10	2

5	Almacenar archivos logs en la Base de Datos.	3	Bajo	18	2

2.5.4 Cronometrar Escenarios

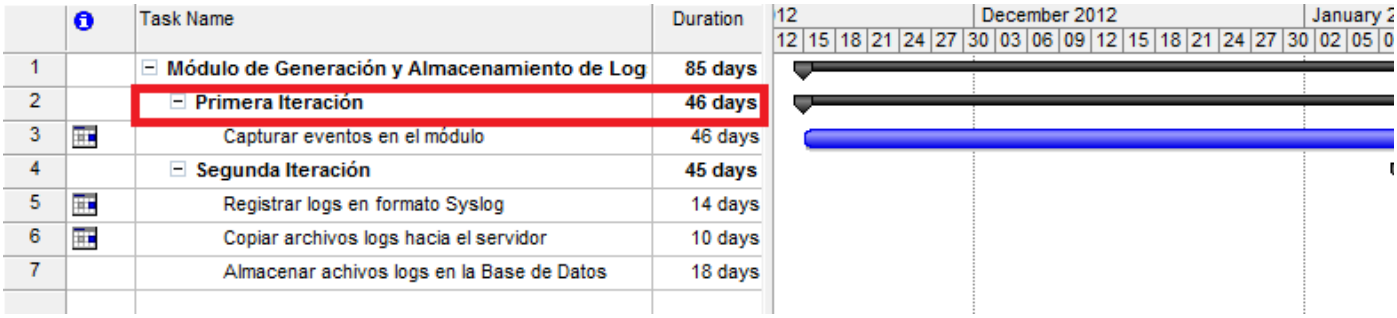


Figura 11: Primera iteración

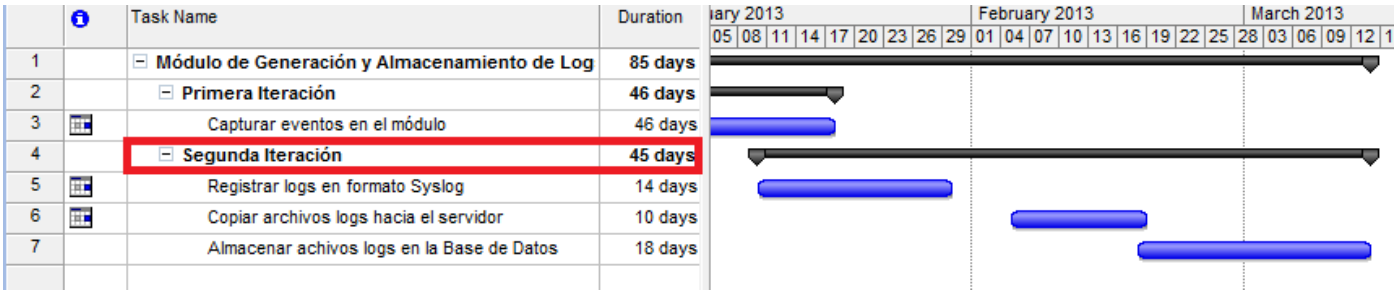


Figura 12: Segunda iteración

2.5.5 Requisitos de calidad de servicio

Usabilidad

1. El Módulo será distribuido en idioma español.
2. Los términos utilizados se establecerán acorde al negocio correspondiente para facilitar la comprensión de la herramienta de trabajo.

Fiabilidad

- El sistema estará disponible las 24 horas durante los 7 días de la semana.
- No se realizarán mantenimientos preventivos en horario laboral, deberán ejecutarse en un horario estipulado, para no afectar la disponibilidad del sistema.
- Las fallas del *software* se dividirán en dos categorías:
  - ✓ **Simples**: la solución y la actualización se realizarán en línea en un período inferior a 10 minutos.
  - ✓ **Complejas**: la solución y actualización se realizarán en un tiempo que se definirá posterior a una evaluación detallada.
- Solo se accederá a la base de datos desde la aplicación, nunca directamente desde el gestor de base de datos.
- Se garantizará la consistencia de los datos, se realizarán comprobaciones y validaciones automáticas en todos los casos posibles.

Eficiencia

- El tiempo mínimo de transferencia de archivos log al servidor debe ser inferior a 10 segundos.

Soporte

- El estilo de codificación que se adoptará por convención es el Camel.
  - ✓ Este define que los nombres de los diferentes elementos de código están formados por palabras que comienzan con mayúsculas seguidas por minúsculas. Además todos los nombres de las diferentes estructuras de código deberán ser descriptivos.
- Los nombres de las clases, propiedades y métodos comenzarán con letra inicial mayúscula. De igual forma se hará con cada palabra que compone el nombre y no se utilizará el guión bajo "_" como delimitador entre palabras.
- Los nombres de las interfaces comenzarán con la letra i.
- Los nombres de clases que implementan interfaces deberán tener el sufijo Impl.
- Los nombres de los atributos de las clases comenzarán con guión bajo "_", seguidos de la primera palabra del nombre con minúscula y el resto de las palabras que componen el nombre comenzarán con mayúsculas.

Restricciones de diseño

- El sistema debe implementarse usando el lenguaje C#.
- El sistema gestor de bases de datos será Oracle 11g.

Requisitos de licencia

- Para el desarrollo del módulo se necesitan un conjunto de aplicaciones, plataformas, sistemas operativos, gestores de bases de datos, herramientas, que son sistemas propietarios y necesitan de licencias para su buen desempeño y soporte, las cuales son:
  - ✓ Visual Studio 2010
  - ✓ SO Linux para servidor central de logs
  - ✓ Net Framework 4
  - ✓ Entity Framework 4.1
  - ✓ Sistema Gestor de Base de Datos Oracle 11g
  - ✓ PLSQL Developer 7.1.4.1390

2.5.6 Descripción de escenarios

Entre los principales escenarios a desarrollar en el módulo se encuentra “Capturar eventos en el módulo”, cuya descripción se detalla a continuación. El resto de los escenarios se encuentran descritos en el **Anexo 2**.

Tabla 6: Descripción del Escenario Capturar evento en el módulo.

Nombre del Escenario: Capturar evento en el módulo.		**Identificador:** ES_1
Objetivo del Escenario: Registrar todos los eventos generados en cada uno de los subsistemas IDM.		
Persona: Administrador del módulo		
Iteración: 1ra	**Prioridad:** 5	**Complejidad:** Alta

Descripción: El administrador del módulo realiza la configuración del mismo en un fichero definido para dichas configuraciones.
Validaciones: La ruta establecida para almacenar los logs debe ser una ruta válida en la PC local. El tamaño definido para los logs debe ser un valor numérico desde 0.1.

2.6 Conclusiones parciales

- La captura de visión del proyecto así como su planificación hicieron posible delimitar el alcance de la propuesta, así como determinar el tiempo de duración de la misma.
- La descripción del módulo de generación y almacenamiento de logs, así como la especificación de los escenarios y requisitos de calidad de servicio, permitieron detallar las características del sistema y diseñar la propuesta de solución.

Capítulo 3: Construcción y Prueba

3.1 Introducción

En el presente capítulo se especifica la arquitectura del sistema, así como la propuesta de desarrollo del módulo utilizando el mecanismo de intersección de código con el objetivo de modificar en el menor grado posible el código de los diferentes subsistemas del IDM. También se muestran los diagramas que propone la metodología de desarrollo en la fase de construcción, para facilitar el desarrollo de la aplicación así como la descripción del modelo de datos.

La realización de pruebas al módulo es el siguiente paso que propone la metodología MSF, dando inicio a la etapa de estabilización de la misma. Durante el desarrollo de un *software* los errores pueden presentarse en cualquiera de las etapas de su ciclo de vida. Las pruebas de *software* constituyen uno de los elementos de mayor importancia dentro del proyecto, ya que favorecen el aseguramiento de la calidad a través del análisis exhaustivo de los errores introducidos en las fases previas del proyecto.

Una vez concluida la fase de construcción se detallarán las pruebas que se le realizarán al módulo y un resumen de evaluación teniendo en cuenta el resultado de las pruebas.

3.2 Especificación de la arquitectura

La arquitectura del *software* es el diseño de más alto nivel dentro de la estructura de un sistema, tiene la tarea de definir los módulos principales, sus responsabilidades dentro del proyecto y las interacciones que existirán entre los mismos. Durante la fase de implementación se desarrolla la arquitectura del módulo, la cual aporta una visión abstracta que proporciona un esquema de referencia útil para guiar el desarrollo del *software*.

En el diseño del módulo de generación y almacenamiento de logs, teniendo en cuenta la sencillez del mismo, se ha adoptado como arquitectura el mecanismo de intersección de código o la utilización de la programación orientada a aspectos, elemento sobre el cual el módulo asienta su funcionamiento más básico.

3.2.1 Diagrama de Arquitectura del Módulo

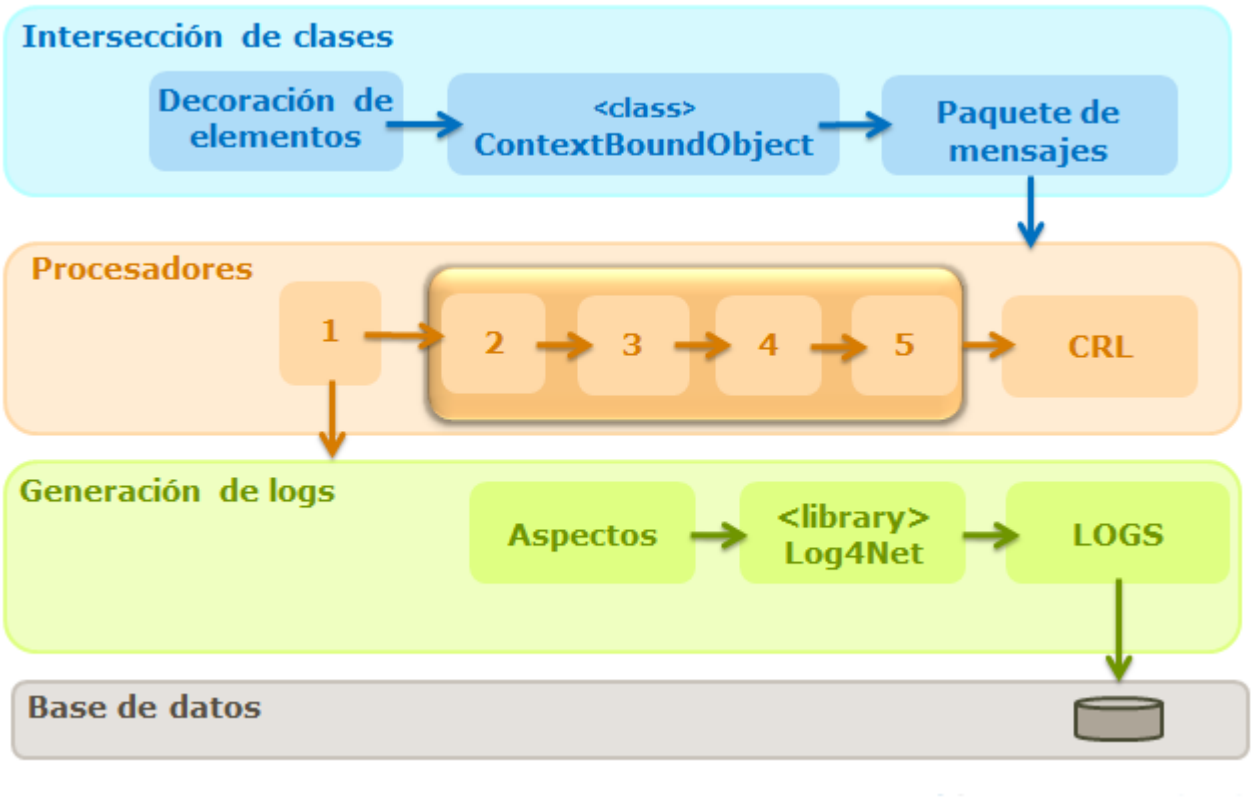


Figura 13: Diagrama de Arquitectura del Módulo

3.3 Diagrama de clases

Un diagrama de clases está determinado por las clases, atributos y las relaciones entre ellos y se utilizan durante la etapa de análisis y diseño del *software* para modelar la visión estática del sistema. El diagrama de clases correspondiente al módulo se encuentra en el **Anexo 2** del presente documento.

3.4 Modelo de datos

El modelo relacional de la base de datos que se muestra a continuación es un modelo genérico de tablas que se conoce como Entity Attribute Value (EVA por sus siglas en inglés) el cual aumenta la flexibilidad y robustez del mismo. Este patrón de diseño permite tener un dominio detallado sobre todos los atributos que puedan ser asignados a cualquier elemento almacenado, por lo que cada registro habla por sí solo y no tendrá libre interpretación, las aplicaciones que hacen uso de estos valores serán dominados

directamente por el manejador de base de datos. La principal ventaja que posee el modelo EVA es la posibilidad de ampliar el conjunto de atributos sin cambiar la estructura de la tabla (Strappazzon, 2010).

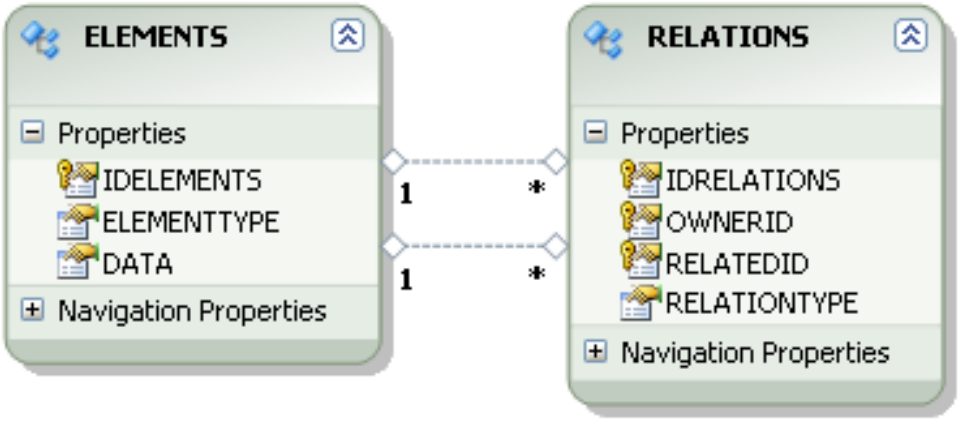


Figura 14: Modelo de Datos

3.5 Diagrama lógico de centro de datos

El diagrama lógico de centro de datos es otro de los artefactos que define la metodología MSF para el desarrollo de *software*. Este artefacto proporciona una representación lógica del centro de datos, permitiendo comunicar información importante sobre el entorno de destino en el que se implementarán los sistemas de la aplicación, además documenta las configuraciones concretas del *software* del servidor de aplicaciones, como *Internet Information Server*, SQL Server o BizTalk Server, y muestra cómo se interconectan estos servidores lógicos configurados (Microsoft, 2013).

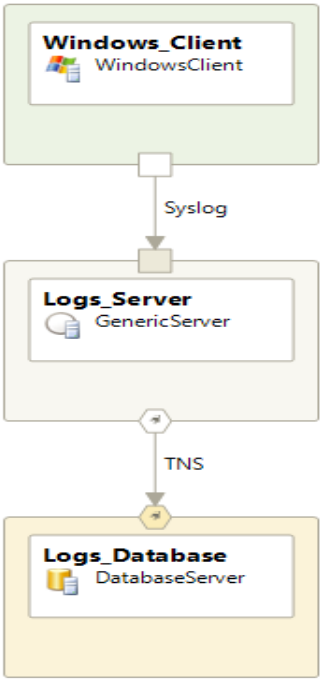


Figura 15: Diagrama Lógico de Centro de Datos del Módulo de Generación y Almacenamiento de Logs.

El diagrama muestra la relación entre los servidores del módulo así como los protocolos empleados para su comunicación. Representa un grupo de clientes que se comunican con el servidor destinado para el almacenamiento de los logs a través del protocolo Syslog. En dicho servidor se localizan todos los eventos generados por los subsistemas y se encuentra hospedada la herramienta Nxlog que se encargará del envío de los logs a la base de datos, se muestra, además, la comunicación del servidor de logs con el servidor de base de datos a través del protocolo TNS.

3.6 Pruebas

A continuación se describen el conjunto de validaciones realizadas a los resultados obtenidos durante la etapa de construcción de la solución propuesta, a fin de garantizar la calidad y el correcto funcionamiento del módulo de generación y almacenamiento de logs.

Teniendo en cuenta el concepto introducido por Pressman, las pruebas de *software* son: "*un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación*" (Pressman, 2007). El proceso de las pruebas de *software* involucra un

conjunto de herramientas, técnicas y métodos que evalúan el desempeño de un programa, comprenden además las operaciones del sistema, evaluando los resultados bajo condiciones controladas, lo que convierte a las pruebas de *software* en unos de los elementos de mayor importancia dentro del ciclo de vida del proyecto.

3.6.1. Pruebas unitarias

Para verificar el correcto funcionamiento del módulo se prosigue a definir la estrategia de prueba a seguir. A partir de las características que presenta el módulo se decide ejecutar métodos del nivel de pruebas unitarias, estas pruebas se realizan con el principal objetivo de comprobar que el *software* está correctamente codificado a través del aislamiento de cada una de las partes del programa, verificando que estas partes individuales funcionen adecuadamente (MORENO, 2005).

3.6.2. Método de prueba

Para la realización de las pruebas unitarias se define como método de prueba las pruebas de caja blanca. Las pruebas de caja blanca, denominadas también pruebas de caja de cristal, constituyen un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante este método, el ingeniero del *software* puede obtener casos de prueba que:

- Garanticen que se ejecuten por lo menos una vez todos los caminos independientes de cada módulo.
- Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez (Pressman, 2007).

3.6.3. Diseño de los casos de prueba

Una vez determinado los tipos de pruebas que se van a realizar y los métodos que se empelarán en la elaboración de las mismas, se procede a la ejecución de las pruebas unitarias, para ello se ha utilizado el IDE de desarrollo Visual Estudio 2010. A continuación se muestran los resultados arrojados por las pruebas realizadas a la secciones más importantes del módulo, el resto de las pruebas se encuentran disponibles en el **Anexo 3.**

Tabla 7: Descripción de la prueba de unidad a la sección Generar logs de módulo

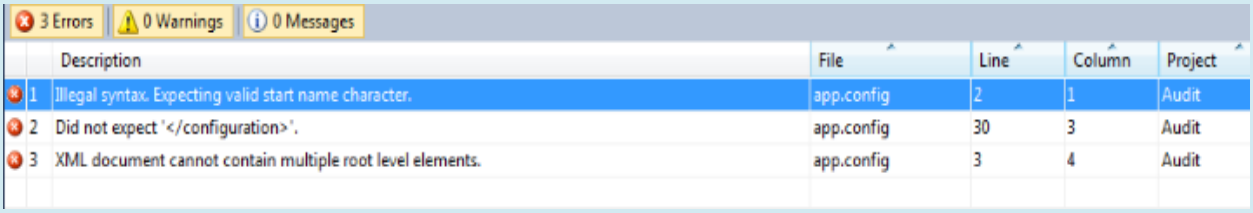
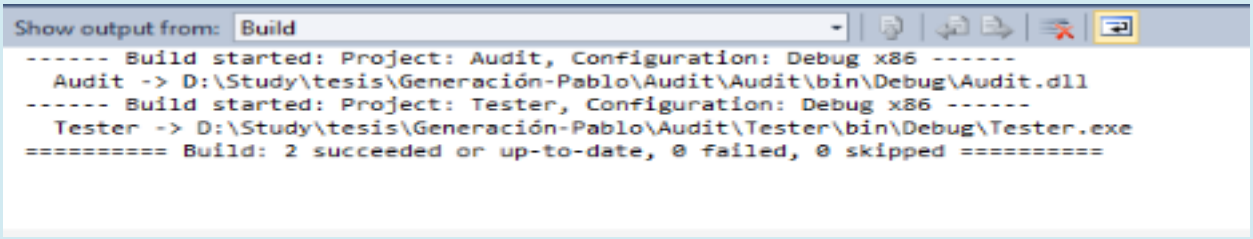
Prueba de Unidad		
Estado: Satisfactoria	**Método de prueba:** Caja Blanca	**Última ejecución:** 28/4/2013
Sección a probar: Generar logs	**Escenario de la Sección:** Configuración incorrecta	
Flujo de actividades: ✓ Importar componente. ✓ Configurar el componente usando archivo de configuración.		
Variable: Fichero de configuración	**Respuesta esperada:** Se genera un listado con los errores de configuración ocurridos.	
Resultado: 		

Tabla 8: Descripción de la prueba de unidad a la sección Interceptar código del módulo

Prueba de Unidad		
Estado: Satisfactoria	**Método de prueba:** Caja Blanca	**Última ejecución:** 28/4/2013
Sección a probar: Interceptar código.	**Escenario de la Sección:** Decoración de métodos en clase abstracta.	
Flujo de actividades: ✓ Importar componente. ✓ Decorar la declaración de la clase con *InterceptableObject.* ✓ Decorar la declaración del método con el atributo y el aspecto deseado.		

✓ Configurar el componente usando el archivo de configuración. ✓ Ejecutar la aplicación.	
Variable: Clase Abstracta	**Respuesta esperada:** Se generan logs según la configuración especificada.
Resultado:  	

3.7 Conclusiones parciales

- El modelado de los diagramas empleados durante la etapa de diseño para la elaboración del módulo, aportaron una visión estática del sistema y su estructura de datos.
- La definición de la arquitectura del sistema permitió establecer la estructura del módulo desarrollado y posibilitó tener una visión abstracta que proporciona un esquema de referencia útil para guiar el desarrollo del *software*.
- La elaboración de la estrategia de prueba y la posterior ejecución de las mismas, permitió al sistema validar el correcto funcionamiento de los procesos internos del *software* aumentando la calidad y robustez requerida.

Conclusiones Generales

El registro de eventos es un parámetro de seguridad esencial para cualquier empresa, ya que la seguridad constituye una de las bases fundamentales para mantener la integridad de la información que se maneja dentro de las organizaciones. El subsistema de auditoría del IDM, que radica en el centro CISED de la Universidad de Ciencias Informáticas, no disponía de un registro de los eventos generados por sus componentes internos, por lo que se propuso llevar a cabo un mecanismo que permitiera tener el control de los archivos de registro del IDM.

A continuación se muestran las conclusiones generales que evidencian el cumplimiento de los objetivos trazados al inicio de esta investigación.

- El estudio de diferentes sistemas para el manejo de archivos de registros facilitó la comprensión del proceso de recolección y almacenamiento logs.
- La especificación de las tecnologías, herramientas y lenguajes brindaron los elementos básicos para la elaboración de la propuesta de solución.
- El plan de iteraciones permitió la organización del trabajo, a fin de culminar el desarrollo del módulo en el tiempo requerido.
- La descripción de los escenarios durante la fase de planificación, definida por la metodología MSF para el Desarrollo de *Software* Ágil, facilitó la comprensión de las funcionalidades del módulo.
- El diseño de la arquitectura del sistema, a partir del mecanismo de intersección de código, permitió realizar la generación de logs modificando en un menor grado el código de los subsistemas del IDM.
- Las pruebas realizadas permitieron comprobar las funcionalidades del módulo, las mismas brindaron resultados satisfactorios otorgando validez a la investigación.

Se recomienda a los interesados en mejorar o continuar esta investigación:

- Realizar la integración con el subsistema de reportes y auditoría del IDM.

- Implementar otras versiones del módulo que incorporen nuevas funcionalidades que permitan la recolección de logs no solo del IDM, sino que pueda ser integrado a cualquier sistema o aplicación, a fin de mejorar la seguridad del centro.

- Presentar la investigación en eventos de cortes científicos.

Referencias Bibliográficas

Analog. 2010. Analog. [En línea] 2010. www.analog.cx.

Apache *Software*, Foundation. 2006. Logging Services. [En línea] 2006. http://logging.apache.org/log4net.

Baryolo, Gómez. 2012. Información de Ciencias de las Salud. *Modelo de gestión de log para la auditoría de información de apoyo a la toma de decisiones en las organizaciones.* [En línea] 2012. http://acimed.sld.cu/index.php/acimed/article/view/270/232.

Bautista, Jose Carlos Correa. 2012. *SUBSISTEMA DE APROVISIONAMIENTO DE USUARIOS PARA EL SISTEMA DE ADMINISTRACIÓN DE IDENTIDADES.* 2012.

2005. Borrmart.SA. [En línea] 2005. [Citado el: 15 de noviembre de 2012.] http://www.borrmart.es/articulo_redseguridad.php?id=818.

Canós, José H. 2009. *Metodologías Agiles en el desarrollo de Software.* 2009.

Catrian. 2010. Catrian. [En línea] 2010. http://www.catrian.com.

Chiavenato, Idalberto. *Introducción a la teoría general de la administración.*

Consulting, BI. 2010. BI Consulting. [En línea] 2010. http://www.biconsulting.com.mx/index.php?option=com_content&view=article&id=112&Itemid=150.

David Lang - Intuit. 2012. *Building a 100K log/sec logging infrastructure.* 2012.

DotNet. 2013. DomiDotNet. [En línea] 2013. http://dotnet.domitienda.com/index.php/tag/spring-net/.

Fazani, Amanda. 2009. Cultura Empresarial. [En línea] 2009. http://culturaempresarialparatodos.blogspot.com/2009/02/62-auditoria-informatica.html.

Galdámez, Pablo. 2010. Seguridad Informática. [En línea] 2010. http://web.iti.upv.es/actualidadtic/2003/07/2003-07-seguridad.pdf.

García, Walter Baluja. 2004. SALDI. [En línea] 2004. http://www.criptored.upm.es/guiateoria/gt_m189b.htm.

Gerhards, Rainer. 2010. Rsyslog. [En línea] 2010. www.rsyslog.com.

GFI . 2013. GFI *Software*. [En línea] 2013. http://www.gfi.com/eventsmanager.

Historia de la Informática en Cuba - EcuRed. [En línea] [Citado el: 11 de noviembre de 2012.] http://www.ecured.cu/index.php/Historia_de_la_Inform%C3%A1tica_en_Cuba.

Hitachi ID Systems. 2013. Identity Management Terminology. [En línea] 2013. http://hitachi-id.com/identity-manager/docs/identity-management-terminology.html.

Huerta, Antonio Villalón. 2002. Red Iris. *Seguridad en Unix y Redes Versión 2.1.* [En línea] 2002. http://www.rediris.es/cert/doc/unixsec/node5.html.

International Business Machines Corporation(IBM). 2013. IBM Tivoli Composite. [En línea] 2013. http://publib.boulder.ibm.com/tividd/td/ITWSA/ITWSA_info45/en_US/HTML/guide/c-logs.html.

Joelsy P. **Joelsy Porvén Rubier, Otros. 2013.** s.l. : RIELAC, 2013, Vol. Gestión Auntomatizada e Integrada de Controles de Seguridad.

Joelsy Porvén Rubier, otros. 2013. *Gestión automatizada e integrada de controles de seguridad informática.* Ciudad de la Habana : s.n., 2013.

José Antonio, González Seco. *El lenguaje de programación C#.*

La importancia de la Gestión de Identidades (II). [En línea] http://www.ibermaticajusticia.com/la-importancia-de-la-gestion-de-identidades-ii/.

Lic. Adrián Coutin, otros. 2003. [En línea] 2003. www.cuba.cu.

López, Osmany. 2008. *Propuesta para la Recolección Centralizada de Logs.* 2008.

Martín, Beatriz. 2011. Justicia XXI.O. [En línea] 4 de marzo de 2011. [Citado el: 11 de noviembre de 2012.] http://www.ibermaticajusticia.com/la-importancia-de-la-gestion-de-identidades/.

—. 2011. Justicia XXI.O. [En línea] 4 de marzo de 2011. [Citado el: 11 de noviembre de 2012.] http://www.ibermaticajusticia.com/la-importancia-de-la-gestion-de-identidades/.

Media, O'Reilly. 2010. Learning Python Fourth Edition. [En línea] 2010. www.oreillynet.com/pub/au/446.

Microsoft. 2013. Visual Estudio. [En línea] 2013. http://msdn.microsoft.com/es-es/library/vstudio/dd460723%28v=vs.100%29.aspx.

Microsoft, Corporation. 2007. [En línea] 2007.

MORENO, JURISTO. 2005. Técnicas de evaluación de *software*. [En línea] 2005. http://is.ls.fi.upm.es/udis/docencia/erdsi/Documentacion-Evaluacion-6.pdf. .

Novell. 2008. *Gestión de Identidad y acceso.* 2008.

NXLOG Community Edition,2012 . NXLOG Community Edition. [En línea] http://nxlog-ce.sourceforge.net.

Oficina Nacional de Estadíticas (O.N.E). 2010. *Tecnologías de la Información y las Telecomunicaciones en Cifras.* Cuba : s.n., 2010.

Oscar. 2010. Syslog: la piedra angular de los registros del sistema. [En línea] 2010. http://ocubom.wordpress.com/2010/10/13/syslog-la-piedra-angular-de-los-registros-del-sistema/.

pepito. 2013. [En línea] 2013.

Pfleeger, Charles P. 2006. *Security in computing.* 2006.

Pressman. 2007. *Ingeniería de Software 7 edición.* 2007.

Request for Comments 3164. 2001. Request for Comments. [En línea] 2001. http://www.normes-internet.com/normes.php?rfc=rfc3164&lang=es.

Rivas, Gonzalo Alonso Versión 1.2. 2001. *Auditoría Informática.* 2001.

Roberth G, Figueroa. 2007. *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.* 2007.

Sánchez. 2009. Identidad. [En línea] 2009. http://www.bdigital.unal.edu.co.

Security Advisor. 2005. Estándar de Seguridad Infromática ISO 27001:2005. [En línea] 2005. www.gcpglobal.com/docs/Intro_ISO27001.pdf.

Source Forge. 2011. Source Forge.Net. [En línea] 2011. www.logwatch.org.

Srinivasan, Madhan Kumar. 2010. *ANALYSIS ON IDENTITY MANAGEMENT SYSTEMS.* 2010.

Strappazzon, N. 2010. Uso del modelo Atributos & Tipos (EAV) en el diseño de Base de Datos. [En línea] 2010.

Tech Republic. 2013. Tech Republic. [En línea] 2013. http://www.techrepublic.com/.

Valls, Aina Giones. *La gestión de la identidad digital: una nueva habilidad.*

WSO2 . 2013. WSO2. [En línea] 2013. http://wso2.com.

Bibliografía Consultada

Bautista, Jose Carlos Correa. 2012. *SUBSISTEMA DE APROVISIONAMIENTO DE USUARIOS PARA EL SISTEMA DE ADMINISTRACIÓN DE IDENTIDADES.* 2012.

Borrmart.SA. 2005. [Disponible en: http://www.borrmart.es/articulo_redseguridad.php?id=818.]

Catrian. 2010. Catrian. [Disponible en: http://www.catrian.com.]

David Lang - Intuit. 2012. *Building a 100K log/sec logging infrastructure.* 2012.

Joelsy Porvén Rubier, otros. 2013. *Gestión automatizada e integrada de controles de seguridad informática.* Ciudad de la Habana : s.n., 2013.

López, Osmany. 2008. *Propuesta para la Recolección Centralizada de Logs.*

Novell. 2008. *Gestión de Identidad y acceso.* 2008.

Roberth G, Figueroa. 2007. *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.* 2007.

Sánchez. 2009. Identidad. [Disponible en: http://www.bdigital.unal.edu.co.]

Srinivasan, Madhan Kumar. 2010. *ANALYSIS ON IDENTITY MANAGEMENT SYSTEMS.* 2010.

DatabaseSpy, 2011b. [Disponible en: http://www.altova.com/es/databasespy.html.

DiffDog, 2011c. [Disponible en: http://www.altova.com/es/diffdog/diff-merge-tool.html.

***MapForce*, 2011d.** [Disponible en: http://www.altova.com/es/mapforce.html.

SchemaAgent, 2011e. [Disponible en: http://www.altova.com/es/schemaagent.html.

SemanticWorks, 2011f. [Disponible en: http://www.altova.com/es/semanticworks.html.

StyleVision, 2011g. [Disponible en: http://www.altova.com/es/stylevision.html.

UModel, 2011h. [Disponible en: http://www.altova.com/es/umodel.html.

XMLSpy, 2011i. [Disponible en: http://www.altova.com/es/xmlspy.html.

CARLOS B. QUnit, 2011. *Testeando nuestras aplicaciones*

CARLOS, O. R. J. *Metodología de Desarrollo MSF*. Universidad Interamericana para el Desarrollo, 2008.

CÉSAR, T. L.; J. C. NELSON*, et al. Guía de Arquitectura N-Capas orientada al Dominio con .Net 4.0*. MICROSOFT, 2011.

CISED. *Proyecto Técnico Sistema de Administración de Identidades.*, 2012.

CRAIG, L. *UML y Patrones Introducción al análisis y diseño orientado a objetos.*, 2004. p. 8420534382.

Glosario de Términos

A

API (Application Programming Interface): interfaz de programación de aplicaciones, conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación.

Arquitectura: constituye una guía general que indica la estructura, funcionamiento e interacción entre las partes de un *software*. Es el diseño de más alto nivel de la estructura de un sistema y está conformada por un conjunto de patrones y abstracciones coherentes que proporcionan el marco de trabajo.

Aplicación: cualquier programa que corra en un sistema operativo y que haga una función específica para un usuario. Por ejemplo, procesadores de palabras, bases de datos, agendas electrónicas, entre otros.

Autenticación: la autenticación es el acto de proveer una identidad a una red, aplicación o recurso. Las técnicas de autenticación van desde una simple entrada de identificador de usuario y contraseña hasta potentes mecanismos como los ficheros, certificados de clave pública y biométrica.

Autorización: es el proceso de decidir si una identidad digital es permitida para ejecutar una acción de respuesta. La autorización sucede después de la autenticación y usa atributos o derechos, asociados con la identidad digital para determinar a qué recursos puede acceder dicha identidad digital.

B

Buffering: almacenando de información en una memoria intermedia con el objetivo de que el flujo de información pueda realizarse de manera continua.

C

C#: (pronunciado si sharp en inglés) es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Cronometrar: cronometrar el desarrollo de un *software* es tener el control del tiempo de vida de este.

D

DLL (Dynamic-Link Library): es el término usado para referirse a los archivos con código ejecutable que se cargan bajo demanda de un programa por parte del sistema operativo. Esta denominación es exclusiva a los sistemas operativos Windows siendo ".dll" la extensión con la que se identifican estos ficheros, aunque el concepto existe en prácticamente todos los sistemas operativos modernos.

E

Escenarios: término empleado para hacer referencia a los requisitos del *software* los cuales consisten en describir parcialmente el comportamiento de un sistema facilitando la comprensión de la aplicación y su funcionalidad.

Eventos: son sucesos que ocurren en los sistemas operativos o aplicaciones como resultado de la ejecución de procesos internos o la interacción con otros sistemas.

F

Framework: estructura de soporte definida en 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*s para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

I

IDE: *software* compuesto por un conjunto de herramientas de programación. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Iteración: es el acto de repetir un proceso con el objetivo de alcanzar una meta deseada, objetivo o resultado. Cada repetición del proceso también se le denomina una "iteración", y los resultados de una iteración se utilizan como punto de partida para la siguiente iteración.

M

Módulo: constituye una parte autónoma de un sistema.

P

Plugin: complemento que se relaciona con otra aplicación para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal. También se conoce como *plug-in* (del inglés enchufable o inserción), *add-on* (conector o extensión).

R

Rol: representa el conjunto de conductas esperadas de quien ocupa una determinada posición en el grupo del que forma parte.

S

Servidor: un servidor es una computadora que maneja peticiones de datos, correo, servicios de redes y transferencia de archivos de otras computadoras (clientes). También puede referirse a un *software* específico. Una computadora puede tener distintos *software*s de servidor, proporcionando muchos servidores a clientes en la red.

Servidor Remoto: es una combinación de hardware y *software* que permite el acceso remoto a herramientas o información que generalmente residen en una red de dispositivos.

T

TCP (Transmission Control Protocol): Protocolo de Control de Transmisión, es un protocolo utilizado dentro de las redes de datos, para establecer conexiones a través de las cuales puede enviarse un flujo de datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto.

TNS (Transparent Network Substrate): es el nombre por el que se conocen las instancias de una base de datos Oracle en una red. El nombre de servicio TNS se asigna al configurar la conectividad a la base de datos de Oracle. La replicación utiliza el nombre de servicio TNS para identificar al suscriptor y establecer las conexiones.

U

UDP (User Datagram Protocol): es un protocolo del nivel de transporte basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera.

Anexos

4.1 Anexo 1: Descripción de escenarios del módulo

Tabla 9: Descripción de escenario Registrar logs en formato Syslog.

Nombre del Escenario: **Registrar logs en formato Syslog**		**Identificador:** ES_2
Objetivo del Escenario: Permitir el registros de los eventos generados en formato Syslog.		
Persona: Sistema		
Iteración: 2da	**Prioridad:** 4	**Complejidad:** Media
Descripción: Se generan los ficheros en formato Syslog teniendo en cuenta las definiciones de mensajes Syslog documentadas en el RFC 3164.		

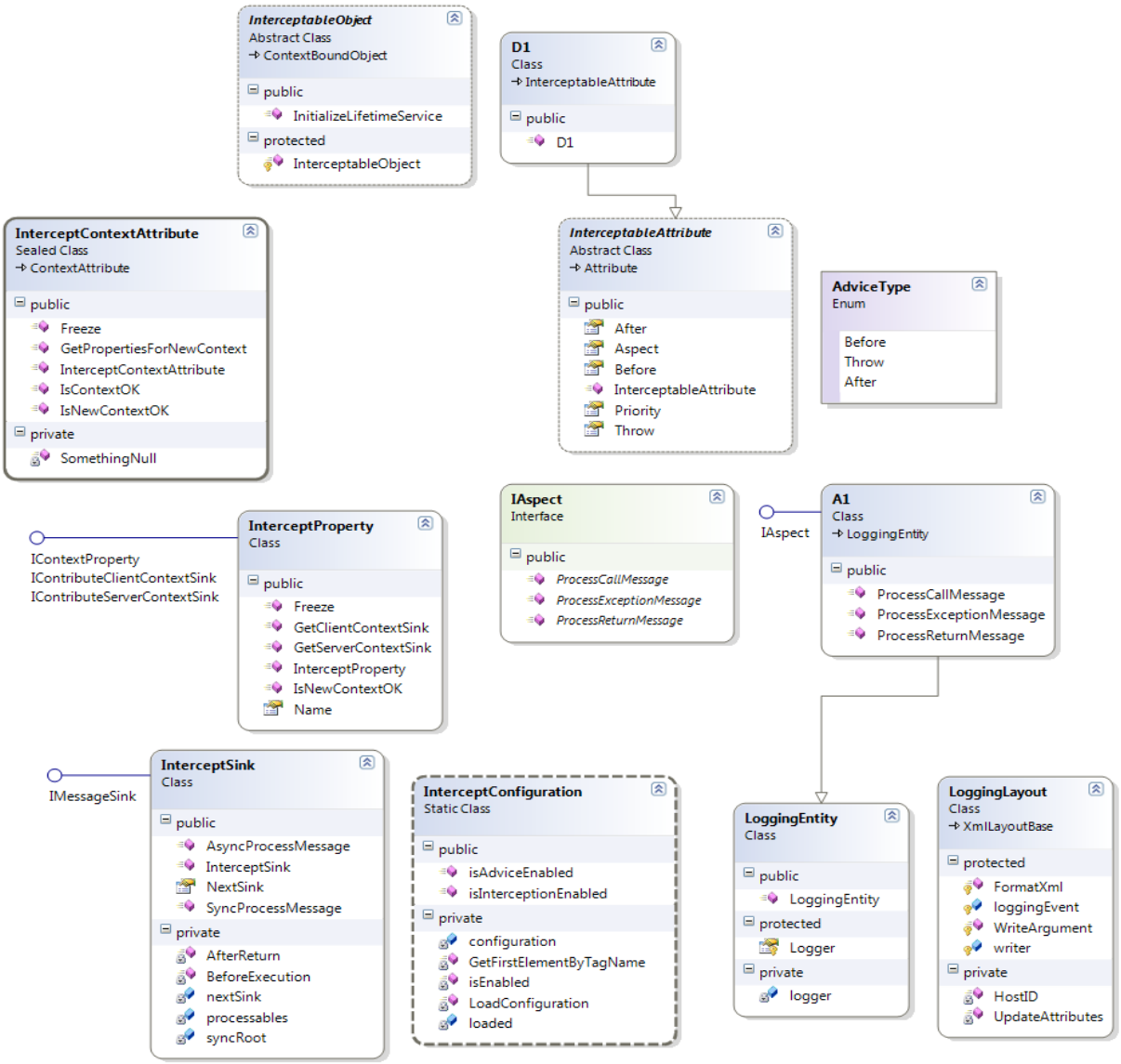
Tabla 10: Descripción de Escenario Copiar archivos logs hacia el servidor.

Nombre del Escenario: Copiar archivos logs hacia el servidor		**Identificador:** ES_4
Objetivo del Escenario: Permitir la copia de todos los eventos generados hacia el servidor de logs.		
Persona: Sistema		
Iteración: 3era	**Prioridad:** 3	**Complejidad:** Baja
Descripción: A través de la herramienta Nxlog ubicado en el servidor central de logs, se recolectan todos los logs cliente a cliente.		

Tabla 11: Descripción de Escenario Almacenar archivos logs en la Base de Datos.

Nombre del Escenario: Almacenar archivos logs en la Base de Datos		**Identificador:** ES_5
Objetivo del Escenario: Permitir el almacenamiento de todos los archivos de registros en la Base de Datos.		
Persona: Sistema		
Iteración: 3era	**Prioridad:** 3	**Complejidad:** Baja
Descripción: A través de la herramienta Nxlog se extrae la información generada en los logs y se almacena como un registro más, dentro de la base de datos.		
Validaciones: Cada tipo de dato extraído de los logs debe corresponder con el esperado en la base de datos.		

4.2 Anexo 2: Diagrama de clases del módulo



4.3 Anexo 3: Descripción de casos de prueba

Tabla 12: Descripción de la prueba de unidad a la sección Interceptar código de módulo.

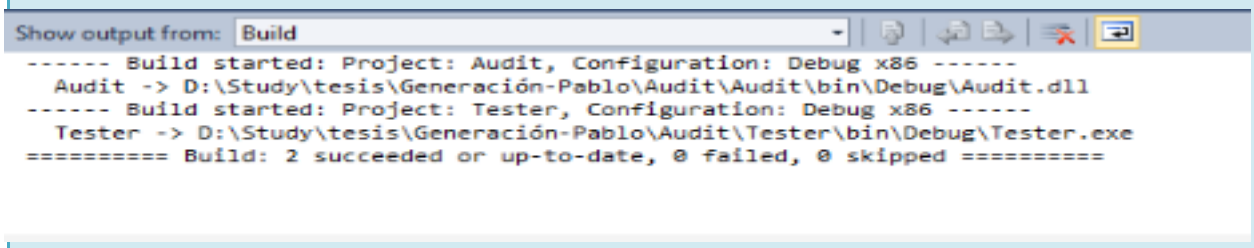
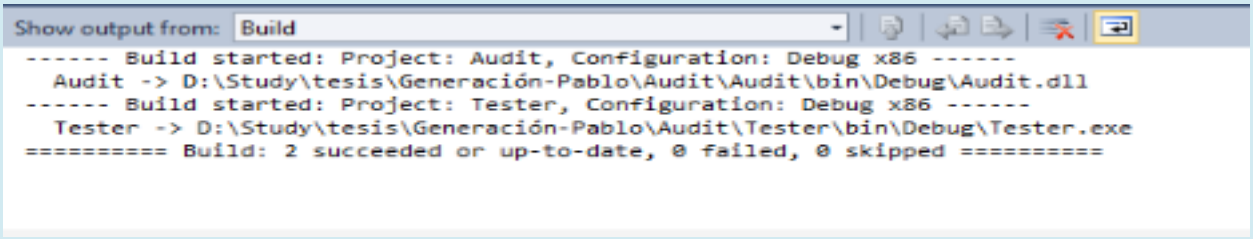
Prueba de Unidad		
Estado: Satisfactoria	**Método de prueba:** Caja Blanca	**Última ejecución:** 28/4/2013
Sección a probar: Interceptar código.	**Escenario de la Sección:** Decoración de métodos en la clase interfaz.	
Flujo de actividades: ✓ Importar componente. ✓ Decorar la declaración de la clase con *InterceptableObjec.t* ✓ Decorar la declaración del método con el atributo y el aspecto deseados. ✓ Configurar el componente usando el de configuración. ✓ Ejecutar la aplicación.		
Variable: Clase interfaz.	**Respuesta esperada:** No se generan logs.	
Resultado:  		

Tabla 13: Descripción de la prueba de unidad a la sección Interceptar código de módulo.

Prueba de Unidad

Estado: Satisfactoria	**Método de prueba:** Caja Blanca	**Última ejecución:** 28/4/2013
Sección a probar: Generar logs	**Escenario de la Sección:** Configuración correcta	
Flujo de actividades: ✓ Importar componente. ✓ Configurar el componente usando el fichero de configuración.		
Variable: Fichero de configuración	**Respuesta esperada:** Se generan logs según la configuración especificada.	

Resultado:



4.4 Anexo 4: Configuración de la herramienta Nxlog

❖ Configuración del archivo nxlog.conf en el cliente

```
Moduledir %ROOT%\modules
CacheDir %ROOT%\data
Pidfile %ROOT%\data\nxlog.pid
SpoolDir %ROOT%\data
LogFile %ROOT%\data\nxlog.log
<Extension syslog>
        Module xm_syslog
</Extension>
<Extension xml>
        Module xm_xml
</Extension>
<Input in>
        Module im_file
        SavePos TRUE
        File "C: / Archivos de programas/IDM/logs/ log.log"
</Input>
<Output out>
        Module om_tcp
        Port 514
        Host 10.8.150.14
</Output>
<Route tcproute>
        Path in => out
</Route>
```

❖ Configuración del archivo nxlog.conf en el servidor

```
<Extension syslog>
        Module xm_syslog
</Extension>
<Input in>
        Module im_tcp
        Port 514
        Host 0.0.0.0
        Exec parse_syslog_bsd();
```

```
</Input>
<Output dbi>
        Module om_dbi
        SQL INSERT INTO log (facility, severity, hostname, timestamp, application, message) \
        VALUES ($SyslogFacility, $SyslogSeverity, $Hostname, '$EventTime', $SourceName, $Message)
        Driver pgsql
        Option host 127.0.0.1
        Option username dbuser
        Option password secret
        Option dbname logdb
</Output>
<Route 1>
        Path in => dbi
</Route>
```