Universidad de las Ciencias Informáticas

FACULTAD 6

Título: Componente generador de logs y reportes para el sistema de video vigilancia Suria Vision.

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

Autor(es): Javier Roberto Bertrán Pompa.

Tutor(es): Ing. Nilo Tomas Díaz Ales.

Co-tutor(es): Ing. Rafael Leodan Cardero Álvarez.

La Habana Junio de 2013 “Año 55 de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a 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 _____.

Javier Roberto Bertrán Pompa

_____
Firma del Autor

Ing. Nilo Tomas Díaz Ales

_____
Firma del Tutor

DECLARACIÓN DE AUTORÍA

DATOS DEL CONTACTO

Datos del Tutor

Nombre y apellidos: Ing. Nilo Tomas Díaz Ales

Correo electrónico: ntdiaz@uci.cu

Categoría docente: Instructor

Año de graduación: 2012

Profesión: Ingeniero en Ciencias Informáticas.

Breve descripción: Graduado en la Universidad de las Ciencias Informáticas. Actualmente pertenece al proyecto Video Vigilancia del departamento Señales Digitales del centro de desarrollo Geoinformática y Señales Digitales de la facultad 6.

Datos del Co-Tutor

Nombre y apellidos: Ing. Rafael Leodan Cardero Álvarez

Correo electrónico: rlcordero@uci.cu

Categoría docente: Instructor

Año de graduación: 2008

Profesión: Ingeniero en Ciencias Informáticas.

Breve descripción: Graduado en la Universidad de las Ciencias Informáticas. Actualmente pertenece al proyecto Video Vigilancia del departamento Señales Digitales del centro de desarrollo Geoinformática y Señales Digitales de la facultad 6.

AGRADECIMIENTOS

Quiero agradecerme por todo el esfuerzo y el sacrificio que he realizado por estar en esta universidad, por el tiempo que he pasado lejos de mi familia y de mis amigos que se encuentran en mi provincia, por ser fuerte y confiar en que podría graduarme, por todos los días tratar de ser mejor y luchar por conseguirlo.

A mi familia por apoyarme incondicionalmente y por confiar en mí todo el tiempo, por criarme de la forma en que soy, fuerte y sin miedo a tratar de conseguir lo que quiero. A mi madre por darme la fuerza y la razón de vivir, siempre demostrándome que los sentimientos y las acciones son imprescindibles para triunfar en este mundo, sin importar quién eres, ni de dónde vienes. A mi padre por estar siempre presente, dándome todo lo que pido y lo que sin saber yo necesito, motivándome a cumplir el sueño de crecer cada día y sentirme la persona más afortunada en este mundo. A mi hermano por ser la persona más importante en mi vida, por estar siempre peleándose conmigo y en desacuerdo en todo, pero así mismo con su amor y su lealtad hacia su hermano mayor, me ha demostrador la felicidad y la verdadera identidad de un gran amigo. A mis tías, mis madres del alma, por acogerme como su hijo durante tanto tiempo, por criarme y enseñarme todo lo que en este momento me define como ser humano. A mi tío Emilio por ser el hombre que utilizó como patrón a seguir, por mostrarme la capacidad y la habilidad necesaria para ser un trabajador honrado, sencillo, honesto y sobre toda la cosas, ser dedicado a mi familia y a los deberes hacia ella. A mis hermanas por hacerme sentir orgulloso todos los días, pues ellas se sienten orgullosas de lo que he hecho y me toman como un ejemplo a seguir, demostrándome que soy el mejor hermano mayor en todo el mundo. A mis primas que durante todo este tiempo se han preocupado por mi situación y siempre están atentas a todo lo hago, mostrándome siempre su apoyo. A todos mis amigos y personas que me quieren y me admiran en mi provincia natal, que los quiero mucho y quiero que sepan que le estoy eternamente agradecido.

A mis amigos y compañeros aquí en la universidad, que entre bromas y risas, se han comportados como verdaderos baluartes, en los que he encontrado un apoyo incondicional y sincero. A Yunet que se ha convertido en una persona especial para mí, brindándome su mano cuando más lo necesitaba, ayudándome con sus consejos precisos y honestos, siempre respetando mis virtudes y mis defectos, a ella le agradezco tener esta confianza absoluta de lograr lo que me proponga y por ser mi ángel de la guarda cuando la soledad y la desconfianza me abatían. Quiero agradecer a todas las personas que me ayudaron a realizar esta tesis, mencionar algunas sería faltar el respeto a otros, por eso les dedico este triunfo y quiero que sepan que nunca olvidaré lo que desinteresadamente han hecho por mí.

DEDICATORIA

Quiero dedicarme este triunfo, como premio a todo el esfuerzo y el sacrificio realizado, por ser el protagonista de esta batalla que he librado desde que comencé en esta universidad, me siento muy orgulloso de ser capaz de graduarme y cumplir con el sueño de mis abuelos América Bahades Beirit y Juan Jorge Bertrán Tamame, quiero dedicarle esta victoria a ellos, que no están conmigo físicamente pero sé que me acompañan en estos momentos de alegría y gozo, al ser ellos los promotores de esta aventura, los quiero muchos mis abuelitos y quiero que sepan que donde estén siempre estaré orgulloso de sentirme su nieto querido, de todas las vivencias que tuve el placer de compartir con ustedes y que siento mucho no haber podido estar cuando partieron a descansar, los amo mucho y los extraño con tanto dolor por no estar a sus lados cuando ocurrió. A mi hermano para que me tome como ejemplo y el curso que viene cuando empiece la universidad sienta orgullo de su hermano mayor y con la competencia sana que nos caracteriza, se sienta tentado a cumplir este maravilloso sueño que significa ser universitario. A mi madre, por estar siempre pendiente y ser la primera persona que me dijo, tú puedes. A mi padre por todo lo que me ha dado y le digo ahora, todo eso y mucho más te lo retribuiré el doble. A mi tía Betty, viste gracias tu formación y tu entrega soy ahora un hombre de bien y he conseguido cumplir con el sueño de toda la familia. A mi tía Yaquelín y a esa preciosura que tiene como hija, les dedico este triunfo y espero que se sientan orgullosos de los logros que he obtenido durante este tiempo. A mi tío Emilio por ser mi segundo padre y acogerme como el hijo varó que no tiene. A mi tía Angela por tener la confianza necesaria y suficiente de que podía triunfar, manteniéndome siempre enfocado y centrado en la meta que me trajo a esta universidad.

Quiero dedicarles este triunfo a todos mis amigos, compañeros y conocidos que de una forma u otra han sido participantes en el cumplimiento de este sueño.

RESUMEN

El proyecto Video Vigilancia perteneciente al centro de producción Geoinformática y Señales digitales contiene actualmente un sistema para cumplir con la actividad de la video vigilancia, llamado Suria Vision. Actualmente dicha aplicación está implementada en plataforma privativa, por lo que cuando se comercializa el producto se debe pagar por licencias y otras normativas que engloba utilizar herramientas de este tipo. Con la migración a software libre que experimenta Cuba y la universidad de las ciencias informáticas, se decidió migrar dicho software a plataformas libres, obteniendo la ventaja que no se debe pagar ninguna licencia cuando se comercialice el producto. Suria cuenta actualmente con una biblioteca que le permite administrar los sucesos (error, advertencia e información) que se genera durante su implementación y ejecución, permitiendo realizar acciones como reconocimiento de fallos, de errores y otros inconvenientes en un tiempo deseado y sin la necesidad en la mayor parte de detener el sistema y buscar entonces por el código, brindando la ventaja de resolver los problemas sin tener detenido al producto por mucho tiempo. Esta investigación consiste en la realización del diseño, implementación y prueba de un componente para la gestión de logs y la generación de reportes con los sucesos que ocurran en Suria Vision, contribuyendo a obtener todas las facilidades que incluyen administrar los datos de los diversos sucesos, para lograr en la actual versión que se obtengan las ventajas antes mencionadas. La implementación del componente para la gestión de logs y la generación de reportes está basada en un ambiente de escritorio, utilizando tecnologías libres.

Palabras claves

Cifrar, descifrar, reportes, sistemas de video vigilancia, sucesos.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

INTRODUCCIÓN

En el mundo de hoy, dada la evolución continua de las TIC[1], se han desarrollado sistemas que permiten la manejabilidad de actividades cotidianas y necesarias para las civilizaciones actuales; automatizando procesos que mejoran la calidad, la seguridad y el progreso de vida de cada individuo. Este desarrollo incluye esferas que convergen en el aspecto económico, político y socio-cultural de las personas. Una de las ramas fundamentales en la que se inserta este avance de la tecnología es la actividad de vigilancia y protección física de las personas y de los activos (tangibles e intangibles). Desde la evolución de la tecnología se han empleado diversas técnicas para garantizar esta actividad y para a su vez poder contener pruebas legales e irrefutables para ajusticiar a los criminales que cometan actos ilícitos.

Los sistemas de video vigilancia permiten el uso de diversos equipos de alta tecnología como cámaras, computadoras, servidores y dispositivos de interconexión para garantizar el flujo de datos entre estos equipos electrónicos. Desde la aplicación de estos sistemas se ha logrado proveer de una potente herramienta a sectores específicos de la sociedad, como: las fuerzas policiales, el ejército, las instituciones de comercio, los bancos y empresas que necesitan de servicios para la protección de sus cuantiosos recursos. Por la utilización global de la informática se han podido difundir cientos de productos que abarcan los procesos antes mencionados, brindando un gestor de servicios a estas instituciones, que permiten automatizar mediante actividades programadas las diversas funciones que cumplen los sistemas de video vigilancia.

Con el desarrollo de la informática ha surgido una problemática esencial para la ciencia computacional referida a los sucesos generados por los diversos software implementados, como son: los software de sistema, los de programación y los de aplicación, que durante su interacción entre ellos mismos o con el hardware de las computadoras generan volúmenes inmensos de datos, los cuales son imprescindibles para reconocer estados, configuraciones, notificaciones, cambios y modificaciones que ocurren durante la ejecución de estos sistemas. Para darle solución a la problemática planteada se emplea el término registro informático que se refiere a la técnica de almacenar los datos de forma relacional y colocarlos en un índice o sistema de orden que permita su acceso y uso en cualquier momento. Los registros son el método que tanto el usuario como el sistema informático utilizan para acceder y administrar toda la información. En los sistemas de video vigilancia ocurre la misma situación, por lo que el uso de registros es un apéndice

[1]Tecnología de la Información y la Comunicación.

fundamental que se debe incluir como funcionalidad en las definidas originalmente (Definición ABC, 2007-2013).

En Cuba se ha incrementado gradualmente la producción de aplicaciones o soluciones informáticas y la necesidad de la informatización de las diversas ramas de la sociedad para lograr un desarrollo sostenible y estable. Los sistemas de video vigilancia son utilizados correctamente por las instituciones cubanas como los bancos para la vigilancia en sus centros por todo el país, la Policía Nacional Revolucionaria que instala el sistema por algunas calles de diversas ciudades cubanas, fundamental es el sistema de monitoreo que existe en la ciudad de La Habana y la aduana que usa la video vigilancia para la protección de la frontera. La necesidad de informatizar ha conllevado a la revolución informática que hoy experimenta el país, específicamente en la UCI[2].

En el centro de producción Geysed[3] particularmente en el proyecto Video Vigilancia existe un sistema llamado Suria Vision, creado para cumplir actividades relacionadas con los sistemas de video vigilancia. En este proyecto se realiza actualmente una nueva versión a dicho producto, cumpliendo la política de migración a software libre establecida por la universidad y el país. En el transcurso de esta migración ha surgido una adversidad referida a que el proyecto no cuenta con una aplicación o herramienta que sea capaz de crear registros para la administración de los sucesos de errores, informaciones y advertencias generados durante el despliegue de Suria Vision. La situación antes mencionada imposibilita reconocer errores, fallas en el sistema, administrar configuraciones, actualizar eventos, reconocer patrones en el funcionamiento de los módulos, sin la necesidad de detener la aplicación y buscar entonces en el código, lo cual dificulta la trazabilidad y demora la entrega completa del producto.

Otro problema importante se refiere a que durante la etapa de implementación los IDE[4]utilizados para dicho fin, no generan estos registros correctamente o no los generan adecuados a las necesidades que solicitan los integrantes del equipo de desarrollo; propiciando los atrasos en dicha etapa y evitando así que no exista una estructuración adecuada al cronograma de trabajo.

A partir de la problemática existente se define como **problema a resolver** ¿Cómo administrar los sucesos de advertencia, error e información generados por el sistema Suria Vision?

---

[2]Universidad de las Ciencias Informáticas.
[3]Geoinformática y Señales digitales.
[4]Entornos Integrados de Desarrollo.

Por lo que se define como **objeto de estudio** los procesos relacionados con la generación y administración de logs.

Teniéndose como **objetivo general**: proponer un componente para la generación de logs, su administración y visualización de reportes para Suria Vision.

Determinándose el **campo de acción** como: la informatización de los procesos referentes a la generación de logs y su administración con los sucesos generados por Suria Vision.

Proponiéndose como **idea a defender** el desarrollo de un componente que permita la generación de logs con los sucesos generados por Suria Vision durante las etapas de implementación y ejecución y que garantice la administración de estos.

Para el cumplimiento del objetivo general se realizarán las siguientes **tareas**:

- Caracterización del tema a nivel internacional y nacional.
- Caracterización de los procesos relacionados con la generación de logs y reportes en el sistema de video vigilancia Suria Vision.
- Caracterización de las herramientas y tecnologías seleccionadas para la implementación del componente generador de logs y reportes.
- Definición de los reportes necesarios a emplear en el sistema de video vigilancia Suria Vision.
- Realización del diseño del componente que permitirá la generación de logs y reportes para el sistema de video vigilancia Suria Vision.
- Implementación del componente que permitirá la generación de logs y reportes para el sistema de video vigilancia Suria Vision.
- Realización de pruebas al componente que permitirá la generación de logs y reportes para el sistema de video vigilancia Suria Vision.

Al culminar estas tareas de investigación se esperan obtener los siguientes **resultados**:

- Documentación que refleja el análisis de metodologías, métodos de funcionamiento y servicios brindados por aplicaciones similares.
- Documentación de los artefactos resultantes del análisis y diseño del componente que permitirá la generación de logs y reportes para el sistema de video vigilancia Suria Vision.

- Artefactos:
  - ✓ Prototipos de Interfaz de Usuario.
  - ✓ Clases del Diseño con las descripciones de los atributos y métodos.
  - ✓ Diagrama de colaboración del diseño.
  - ✓ Modelo de Implementación: Diagrama de Componentes.
  - ✓ Ejemplo de código fuente de las principales clases.
  - ✓ Componente generador de logs y reportes para el sistema de video vigilancia Suria Vision.
  - ✓ Informe de pruebas realizadas al componente generador de logs y reportes para el sistema de video vigilancia Suria Vision.

Durante el desarrollo de esta investigación se utilizan un conjunto de **métodos científicos** como son:

- Los **métodos teóricos:**

- ✓ Analítico-Sintético: Se emplea para la ardua investigación en bibliografías seguras y confiables para una fundamentación teórica del tema a investigar, aprovechándose del estudio de conceptos, aristas, tecnologías y herramientas adecuadas para la realización de dicha disertación, centrándose en el área fundamental para describir los procesos que engloba un sistema generador de logs y a su vez administrador de estos registros.
- ✓ Histórico-Lógico: Mediante un análisis profundo de aplicaciones o soluciones referentes al tema a investigar existentes en el mundo; para un estudio detallado de las características que especifican a estas soluciones, aprovechando conocimientos y mejorar deficiencias encontradas. Abarcando las diversas funcionalidades de dichos productos, para así valorar si puede ser utilizada para la solución de la problemática planteada o algunos de sus eventos pueden ser reutilizados y agregados a la solución a implementar.
- ✓ Modelación: Se emplea para generar abstracciones entendibles en vista de explicar la realidad en la investigación, es el patrón a seguir mediante un modelo que se crea con UML[5] para representar los

[5]Lenguaje Modelado Unificado.

procesos que requiere un componente generador de logs y a la vez modelar la funcionalidad del generador de reportes.

-Métodos empíricos:

- ✓ Observación: Utilizada para el mejor entendimiento del sistema de video vigilancia Suria Vision, observando las funcionalidades que se despliegan en este producto, tanto en ejecución como en su etapa de implementación; facilitando definir las funcionalidades que estructuraran al componente generador de logs. Utilizando de este método técnicas tales como: observación participante y la observación sistemática.
- ✓ Entrevista: Se emplea en la fase inicial para el entendimiento de la problemática existente en el proyecto Video Vigilancia, de las características de la relación del negocio y para el levantamiento de los requisitos funcionales y los no funcionales. Se realizan unas series de preguntas al jefe de proyecto y a una población determinada de integrantes de dicho proyecto ver Anexo 1.

El presente trabajo está dividido en 3 capítulos:

Capítulo 1: En este capítulo se aborda la fundamentación teórica del proceso para generar logs y reportes en sistemas de video vigilancias. Especificar conceptos asociados al tema; describir de forma detallada el objeto de estudio que determina el tema a investigar y describir aplicaciones existentes en el contexto actual.

Capítulo 2: Se describe la arquitectura propuesta para la solución a desarrollar, especificando modelos arquitectónicos y estilos arquitectónicos a utilizar. Se caracterizan las tecnologías, las herramientas y la metodología a utilizar durante la investigación. Así como la realización del modelo del dominio, definir los requisitos tanto funcionales como los no funcionales y culminando este capítulo el modelo del sistema.

Capítulo 3: En este capítulo se trabaja en la implementación y en las pruebas para la solución de la propuesta, obteniéndose artefactos del diseño y la implementación de estos, finalizando con pruebas realizadas en el componente obtenido.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En este capítulo se abordan los conceptos fundamentales acerca de los términos relacionados con el proceso de video vigilancia y de los sistemas generadores de logs y reportes asociados a ese proceso. Se describe el objeto de estudio para un mejor entendimiento de la problemática existente, valorando la importancia y la necesidad de la solución para Suria Vision, fundamentando el tema con la investigación de sistemas existentes en el mundo que cumplen con el proceso ya mencionado, describiendo datos fundamentales de estas aplicaciones, tales como: sus funcionalidades, características fundamentales, el objetivo de su creación, entre otros. Se valora si dichos productos pueden ser utilizados para darle solución a la problemática existente en el proyecto Video Vigilancia.

1.2. Conceptos asociados al dominio del problema

Para un mejor entendimiento del tema a investigar se explican ciertas definiciones técnicas de los términos mencionados durante la etapa de investigación, describiéndose mediante conceptos. Los **sistemas de video vigilancia** no son más que soluciones para la vigilancia global en el cual se asume como objetivo principal la protección de personas y activos en todos los ambientes (aeropuertos, ferrocarriles, carreteras, puentes, puertos, áreas residenciales, comerciales) y ante todas las causas posibles: sabotajes, terrorismo, robo o vandalismo en general. Utilizándose la tecnología digital como patrón descriptivo de sucesos o hechos que ocurran dentro del local o institución a proteger; mediante el uso de la captura y gestión de videos monitoreados en tiempo real.

Dichos sistemas incluyen en su interconexión equipos electrónicos, como cámaras, computadoras, cables de red, dispositivos de interconexión que permiten el flujo constante de datos que provienen de los diferentes eventos (Pallavicini, 2012). El control automatizado que despliegan los sistemas de video vigilancia propicia también en los entornos monitorizados la inteligencia artificial, que se define en analizar e interpretar por sí solo la incidencia de elementos que requieran alertar al personal encargado de la vigilancia. Mediante los productos diseñados para la actividad de video vigilancia se modifica el patrón de utilizar personal para la tarea de vigilar recursos, que pueden comportarse como humanos y materiales. Las aplicaciones usadas con el fin de cumplir con la actividad de vigilancia para administrar sucesos que surgen durante su despliegue hacen uso de **logs**.

FUNDAMENTACIÓN TEÓRICA

Los **logs** son registros informáticos definidos como herramientas para el monitoreo y solución de problemas, enfatizado en el comportamiento del sistema al cual se le aplica dicho software. Son mediciones concretas realizadas a diversos entornos de software y se comportan como una herramienta omnipresente para el diagnóstico de fallas y anomalías en cualquier producto (Ariel Rabkin, y otros, 2012). Los logs forman un conjunto de sucesos almacenados en cualquier aplicación, con el fin de organizar notificaciones en relación y colocarlos al alcance, bajo un índice o sistema de orden que permita su acceso y uso en cualquier momento, se definen como el método que tanto el usuario como el software utiliza para administrar toda la información, a medida que se realizan acciones sobre el producto, se agregan datos o líneas a dichos registros.

Los logs son almacenados en los sistemas para su administración, mediante un recurso de cómputo y memoria llamado **ficheros** que es una unidad lógica de almacenamiento que crea el sistema operativo y cuyo contenido especifica el usuario que trata con esta unidad. Los ficheros están constituidos por un conjunto estructurado y jerárquico de registros lógicos; que almacenan un conjunto de datos virtualmente y pueden ser accedidos y modificados por medio de una computadora. Estos ficheros se identifican por la extensión y el nombre que le asigne el usuario; la extensión es en dependencia de la información y la clasificación del programa que requiera guardar dichos datos (texto, audio o video). Se administra de forma automática y se estructura mediante carpetas y subcarpetas que son localizadas en el disco duro (Rivera, 2012).

Los **reportes** son eventos que agrupan información de acuerdo a un interés específico, donde se muestran resultados de una búsqueda determinada, los cuales pueden ser presentados en forma de textos o a través de gráficos; con los datos reportados se pueden realizar funcionalidades, como cálculos, agrupamientos, patrones estadísticos y mostrarlos al usuario que desea conocerlos. Los reportes son generados dinámicamente, porque cada vez que se llaman o se invocan se actualizan los datos que contienen; la función de estos es aplicar un formato determinado a los datos para mostrarlos por medio de un diseño atractivo y que sea fácil de interpretar por los usuarios (SIPEC WEB, 2008). Son eventos continuos que convergen en resultados deseados; utilizando gráficas, figuras, letras o símbolos que en el mundo se definen como medio de comunicación.

Los **componentes** son unidades de composición de aplicaciones software, que poseen un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio

(Terreros,2012). Clemens Szyperski plantea que un componente de software es una unidad de composición con una interfaz especificada contractualmente y con dependencias explícitas de contexto, un componente de software puede ser desplegado independientemente y es tema de composición por terceras partes. Por lo que en término de programación son el conjunto de código pre elaborado que encapsula la funcionalidad de algún evento, que se inserta en el código de cualquier aplicación informática.

La **administración** es *"el proceso de planear, organizar, dirigir y controlar el uso de los recursos para lograr los objetivos organizacionales"*(Chiavenato, 2004). En el plano informático la administración de logs permite el monitoreo, la recopilación, la consolidación y el análisis de archivos de log (Ipswitch, 2012). Las características del proceso descrito permite dirigir con funcionalidades establecidas la actividad de planificar como almacenar, después organizar dentro de los registros los datos importantes, por criterios de comparación, dirigir y controlar las acciones a realizar sobre los logs como pueden ser: copiar, eliminar y principalmente acceder a la información que se encuentra almacenada dentro de estos. El monitoreo consiste en acceder constantemente a los registros y actualizar los datos almacenados según las acciones que sucedan durante el acceso a los logs.

Los **sucesos informáticos** son datos que permiten una representación simbólica (numérica, alfabética, algorítmica, entre otros) de un atributo o característica de una entidad, donde se describen valores que se generan en cualquier aplicación informática durante la fase de ejecución de la misma. Dichos datos marcan el transcurso que puede tomar un producto y permite comprobar la trazabilidad de los diversos sistemas, a los administradores les facilita conocer las funcionalidades y cómo realizarle diversas acciones para lograr un correcto funcionamiento. Los sucesos representan diferentes facetas en las aplicaciones, se pueden comportar como: errores, información, advertencia, configuraciones, estados, entre otros. Engloban las distintas características que permiten reconocer cualquier acción realizada a cualquier sistema y sobre todo reconocer el origen y la gravedad que pueda tener como consecuencia dicha actividad (Microsoft, 2005).

Todos estos términos forman parte de las definiciones necesarias, que requiere el lector para el conocimiento pleno y el mejor entendimiento de la problemática existente. Suria Vision basado en los fundamentos que engloba un sistema de este tipo, contiene procesos que son necesarios explicar para que se logre comprender la situación, teniendo ya el conocimiento del significado de cada uno de estos términos, se puede explicar detalladamente el objeto de estudio referente al tema de tesis, señalando la

importancia que requiere la implementación de un componente generador de logs y reportes para el sistema. Al conocer las diversas aristas que describen cada uno de los conceptos asociados al dominio del problema, se entiende entonces todo el proceso relacionado con el funcionamiento de los sistemas para la generación de logs y las ventajas que desencadenan.

1.3. Los procesos relacionados con la generación y administración de logs

Para un mejor entendimiento del objeto de estudio, se explican los procesos, actividades y su relación. Los sistemas generadores de logs y reportes son necesarios para todos los programas informáticos que engloben gran cantidad de información, que sus funcionalidades requieran para su correcto funcionamiento los datos generados durante su ciclo completo y que durante los diversos eventos y actividades que cumplan surja la necesidad de estructurar y organizar notificaciones, cambios, informes de sistema y un punto importante los mensajes de errores. Tienen diversas características y cumplen con objetivos definidos como pueden ser: trazabilidad de aplicaciones, configuraciones de seguridad y protección, administración en los hilos de ejecución de diversos programas, entre otros (Mañas, 2006).

Para una mejor descripción del proceso de generación y administración de logs, es necesario abordar como este se adiciona a los originales que tiene cualquier sistema. Empieza haciendo uso de bibliotecas, componentes o aplicaciones que en sus funcionalidades esté definido el trabajo con los datos que generan los productos al que se aplican. Todo se relaciona con peticiones que son realizadas por los administradores de los programas o son programadas automáticamente, las peticiones son tratadas por interfaces en el código que de manera estándar se adiciona al ya existente, complementándose con las funcionalidades y acciones que describe la correcta ejecución del producto. En el proceso se organiza la información obtenida de acuerdo a las peticiones de los administradores, clientes o como se implemente automáticamente y se requiere que se permita agregar nuevos datos organizadamente sin afectar la correlación y manejabilidad de las líneas que fueron adicionadas anteriormente.

Cada registro tiene un formato diferente, según el producto al que se le aplica, pues el objetivo cambia de acuerdo a las características que se encuentran en la aplicación. Existen diversos tipos como son: de errores, de configuración, de seguridad, de protección, de accesibilidad y otros de acuerdo a las funciones que realizan; los cuales se pueden aplicar a sistemas operativos, webs, bases de datos y otras aplicaciones informáticas. Los tipos son especificados por la función que cumpla el registro en ese momento, facilitando que en el tiempo que se accede a estos, pueda cumplir con el objetivo con que se almacenó y permita administrar de manera ágil y coherente toda la información. Los registros validan

diversos campos que son importantes para la verificación del correcto funcionamiento y que sean necesarios para realizar actividades de carácter administrativo (Microsoft, 2005).

Después que se realiza la etapa que se explica en el párrafo anterior, viene definir donde se almacenarán los registros informáticos, con esa utilidad son usados los ficheros, los cuales son administrados por los diferentes dispositivos de almacenamientos como: disco duro, memorias externas, disco compacto, entre otros. La extensión que puedan tener los ficheros son nombrados por los administradores para un mejor reconocimiento o facilidad en el acceso a estos, es implementado junto al código para generar y administrar los sucesos. El administrador inserta la extensión según la característica del sistema y la cantidad de información a administrar que exista; pueden usarse las bases de datos, el paquete office, otros editores de textos y algunas aplicaciones que garanticen el trabajo con datos. Cuando los registros se almacenan, el sistema reconoce los ficheros que son creados para cumplir dicho fin.

La administración de los logs, permite acciones como copiar, eliminar y ordenar todos los datos que contienen, brindando opciones a los administradores de los sistemas al que se le aplique, de manejar a su conveniencia la información y realizar tareas para el acceso a esta, consolidando y probando si las funcionalidades de su producto son correctas. Los registros pueden ser creados y eliminados para facilitar su almacenamiento. Existen acciones o procedimientos que permiten evitar que se aglomeren los datos en dichos registros, algunas de estas actividades se refieren a eliminar los ficheros que contengan logs que no se usen o la información implícita en estos esté caduca; también permite ordenar de diferentes maneras los sucesos como puede ser: ascendentemente, por fecha de almacenamiento, entre otras.

Una parte importante de los procesos para la administración de logs, es la visualización del contenido implícito dentro de estos, pues su objetivo principal es acceder a los datos especificados para medir la trazabilidad del sistema al que se aplica. El acceso a estos registros puede realizarse mediante reportes automáticos, que revisan toda la información contenida y por criterios de búsqueda, muestra lo que el administrador desea encontrar; son eventos continuos que convergen en resultados deseados haciendo uso de: gráficas, figuras, letras o símbolos que en el mundo se definen como medio de comunicación. Los reportes estructuran los resultados según la conveniencia de los clientes y sirven como medio de comunicación e intercambio de informaciones. Mayormente los reportes se definen en herramientas independientes a la que se utiliza para generar los logs (CLM, 2011).

1.4. Descripción actual del dominio del problema

Lo mencionado en el epígrafe 1.3 facilita un mejor entendimiento de los procesos que engloba un sistema para la generación de logs y reportes, en su ámbito general, en este epígrafe se profundiza sobre la problemática que existe en el proyecto Video Vigilancia con respecto a la administración de los sucesos que ocurren en Suria Vision. El sistema contiene principalmente 4 módulos de trabajo que son: Suria Visor que tiene la funcionalidad de mostrar los flujos de videos correspondientes a las cámaras registradas en el sistema, el Suria Grabador es el encargado de la gestión del proceso de grabación en el sistema, Suria Recuperador es el encargado de gestionar las búsquedas y visualización de las grabaciones realizadas y por último Suria Gestor es el encargado de gestionar la interacción entre los demás módulos.

Suria Vision debido a las cantidad de funcionalidades que posee y los eventos que describe durante su despliegue, se requiere el uso de dispositivos de red, tales como: cables, switch, y otros, los que sirven como herramienta de comunicación entre diversos equipos electrónicos, como: computadoras y cámaras. El sistema funciona con la interacción de los diversos equipos de cómputo, en los que funcionan los módulos o subsistemas antes mencionados, las peticiones que realizan para su despliegue generan sucesos importantes como: configuraciones pertinentes, estados de conectividad, notificaciones de acceso, direcciones de computadoras, protocolos y puertos de red para la transmisión de datos, las rutas de almacenamientos, fechas de grabaciones y modificaciones, que son necesarios para el correcto funcionamiento entre ellos mismos.

La plataforma en que se encuentra realizado el sistema es privativa, unos de los IDE utilizado para crear el producto es Visual Studio, que brinda funcionalidades para la administración en el código de sucesos como: errores, datos de configuración, contenido de clases e interfaces, y estructura de objetos, que agiliza el tiempo de implementación y mejora el entendimiento que tienen los programadores y diseñadores con la estructura lógica de Suria Vision. El IDE contiene funcionalidades y eventos agregados, que permiten realizar acciones, que facilitan el trabajo y el proceso de entendimiento de los integrantes del proyecto con respecto al código y la trazabilidad del sistema, como: el encapsulamiento de clases e interfaces, la facilidad de recorrer los datos generados durante la fase de implementación, el reconocimiento de patrones y la verificación de estructuras (Microsoft, 2005).

Existe en el proyecto actualmente un componente que se encarga del proceso de generar logs con los sucesos mencionados. El componente contiene funcionalidades que complementan las que existen en el producto desarrollado, agregándose como biblioteca para que en tiempo de ejecución, los sucesos

generados por los eventos sean almacenados en un fichero txt[6]. Esta herramienta trae como ventaja que se pueda reconocer en tiempo de ejecución la existencia de algún fallo, de errores en el sistema, de datos incorrectos y otros inconvenientes que puedan surgir durante el despliegue, sin la necesidad de detener la aplicación y buscar entonces en el código.

El componente crea un fichero demasiado extenso y con mucha información, debido al tamaño que tiene Suria Vision y a la cantidad de sucesos que generan las funcionalidades del sistema. La situación mencionada provoca que a la hora de buscar ciertos datos se haga engorroso y lento, desencadenando atrasos en el reconocimiento de los percances y en el tiempo de respuesta, pues los ficheros contienen cientos de líneas de datos y características y no existe una herramienta que permita por criterios de búsqueda encontrar los deseados, actualmente se realiza manual, donde un integrante del proyecto revisa los archivos, analizando y accediendo a las notificaciones, en un tiempo demasiado largo. La existencia de una aplicación para visualizar los logs almacenados puede acortar las diversas etapas de recuperación ante cualquier error o fallo.

Actualmente se está realizando con Suria Vision una migración a software libre, específicamente al IDE QT Creator, plataforma que no brinda explícitamente las funcionalidades del Visual Studio, lo cual provoca que en la fase definida para la implementación, existan ciertos atrasos, que se dificulte el trabajo y el proceso de entendimiento de los integrantes del proyecto con respecto al código. Influye en que no se cumpla el cronograma establecido por el proyecto, que la plataforma no sea muy entendible para algunos trabajadores y estudiantes, atrasando el tiempo de implementación de cada funcionalidad con respecto al que tiene en el IDE privativo. Suria Vision contiene requisitos que son cumplidos en la versión actual y su implementación en la nueva plataforma requiere del uso de las facilidades que brinda un generador de logs.

La ventaja que brinda el componente que existe en el sistema actualmente, impone la necesidad de realizar un equivalente para la plataforma libre, para brindar las opciones mencionadas al producto y pueda seguir realizando los eventos correctamente con los datos generados; que permita administrar los sucesos para medir la trazabilidad en su tiempo de ejecución. Se muestra también la necesidad de implementar una aplicación o herramienta que sea capaz de acceder a la información contenida en los registros y realizarle acciones como: visualizarlos, mediante criterios de búsqueda que definan los

[6]texto sin formato.

integrantes del proyecto Video Vigilancia para un mejor entendimiento, utilizando reportes automáticos como medio entendible para acceder a la información y realizar algunas acciones para facilitar la manejabilidad de las características que contengan los diversos sucesos almacenados.

Se evidencia que para realizar exitosamente la nueva versión de Suria Vision en software libre, es necesario contener una aplicación que permita generar y administrar logs, con los datos que produce dicho sistema. Observando las ventajas que brinda tener una herramienta para dicho proceso en software privativo y valorando la importancia que pueda obtener durante la migración. Por lo planteado con anterioridad se puede deducir la necesidad de desarrollar un componente que se encargue de generar diversos registros informáticos tanto en el despliegue, como en la etapa de implementación de Suria Vision; garantizando que se cumplan correctamente los procesos y las funcionalidades que se definen y brindando facilidades para la búsqueda de datos referente con la trazabilidad. Aportando en la administración de los logs la acción de visualizarlos en reportes.

1.5. Análisis de otras soluciones existentes

Para la argumentación de este epígrafe se realizó una investigación acerca de los sistemas informáticos existentes a nivel internacional y nacional que se encargan de la generación de logs y reportes, para cualquier aplicación informática. Explicando las características y los objetivos fundamentales que engloba, mediante una valoración de las funcionalidades principales que realizan y de los eventos que contienen, se especifica si pueden ser utilizados dichos sistemas para darle solución a la problemática existente en Suria Vision. Mostrando cómo se comportan a nivel internacional y a nivel de instituciones los sistemas relacionados con la gestión de logs y la visualización de los datos contenidos en dichos registros en reportes automáticos. Se muestra también los diversos productos por la plataforma en que se implementó, buscando describir principalmente las aplicaciones que se han realizados en software libre.

Estos son ejemplos de las soluciones informáticas:

- KD Reports

Ilustración 1: KD REPORTS

Un sistema informático que permite generar informes o reportes imprimibles de códigos y descripciones XML. Los informes pueden contener párrafos de texto, tablas, encabezados, pies de página y mucho más; permite desde la visualización de los contenidos de bases de datos a la creación de facturas o documentos de identidad. Se trata de una herramienta de desarrollo utilizada en el código fuente, pero permite el uso de plantillas que se crean por el personal de diseño. De esta manera, la actualización de informes no requiere necesariamente cambios de código. KD Reports es capaz de recuperar los datos de los informes de bases de datos. Además, interactúa con Qt Modelo-Vista tecnología para acceder a los datos existentes sobre aplicaciones del modelo. Los informes creados se pueden mostrar en un cuadro de diálogo de vista previa que forma parte de los informes de KD. Ellos pueden ser guardados en archivos PDF o ser enviados directamente a una impresora (Head Office, Hagfors, Sweden: Klarälvdalens Datakonsult AB , 2012), dicha aplicación está implementada en software libre.

- SOLARWINDS

Ilustración 2: SolarWinds

Un sistema informático basado en la colección de registros, análisis y correlación en tiempo real. SolarWinds y Event Manager (LEM) proporciona registro fácil y potente, mediante la automatización inteligente y la gestión de eventos desde cualquier lugar donde se generan los datos para las operaciones de seguridad y protección. El producto permite solucionar de manera eficaz los problemas de rendimiento y seguridad donde se aplique, generando un volumen considerable de registros mediante la correlación de eventos múltiples. Sus funcionalidades permiten aplicar dicho producto en entornos con autonomía en sus centros de datos y tenga necesidad de contener un alto nivel de visibilidad, recurriendo a los almacenamientos de largo plazo y los requisitos de archivo. SolarWinds y Event Manager (LEM) le proporciona la visibilidad y la protección que necesita para la seguridad y el cumplimiento y protección de sus datos y de la intimidad de los clientes en cuestión de horas (SolarWinds Inc., 2012).

- WHATSUP LOG MANAGEMENT

Un sistema informático que permite recoger, almacenar, archivar, analizar e informar sobre Syslog, registros de eventos de Windows, o W3C registros generados por servidores de aplicaciones Web,

balanceadores de carga, firewalls, servidores proxy o aparatos de seguridad de contenido. Herramienta que permite el almacenamiento de sucesos, filtrar, analizar e informar sobre los datos de registros para verificar el éxito de las políticas de seguridad interna y demostrar el cumplimiento regulatorio, generando informes centrados en el cumplimiento de reglas oficiales de personal, de seguridad y cumplimiento. Los archivos de registro (syslog, sucesos Windows, registros W3C) contienen información crítica que podría reducir la exposición de la organización a los intrusos, malware, daños, pérdidas y responsabilidades legales (Ipswitch, 2012).

- TRIPWIRE LOG CENTER

Ilustración 3: Tripwire

Tripwire log center es un registro inteligente dedicado a la solución de la seguridad y la gestión de eventos con las capacidades como: la vigilancia de las amenazas en tiempo real y control de configuración integrada del sistema a que se aplica, creando registros y analizándolos según un patrón definido por la aplicación al que se le integra, para así detectar según datos almacenados posibles violaciones en la seguridad y en las configuraciones pertinentes. El producto brinda a los administradores la posibilidad de investigar los problemas y supervisar el sistema y disponibilidad de las aplicaciones con capacidades de gestión de registros que te ayudarán a descubrir la existencia de algún fallo. Tripwire log center alerta en tiempo real cuando el registro de los sistemas críticos se apaga, ya sea accidentalmente o con intención maliciosa. También se puede utilizar para determinar la causa raíz de las investigaciones forenses de seguridad (Tripwire, Inc, 2013).

- **Biblioteca para la generación de logs en la plataforma privativa en el proyecto Video Vigilancia**

Por las ventajas que brinda la biblioteca para el sistema de video vigilancia que se exporta actualmente en dicho proyecto, es necesario realizar uno que cumpla semejantes funciones, pero con otras adicionadas, para obtener los resultados positivos y que el proyecto contenga de una herramienta activa y capaz de administrar los diversos sucesos. El sistema descrito contiene características específicas que se adaptan a las propias de Suria Vision, cumple con los parámetros de entrada que exige para el correcto

funcionamiento de todo el sistema tanto en la etapa de ejecución como en la etapa de implementación, por lo que la solución a crear obtendrá de esta solución gran parte de sus ejemplos y definiciones. La existencia de dicha aplicación sirve de base para moldear y estructurar todas las funcionalidades a englobar por el componente generador de logs y reportes en la plataforma libre.

Los sistemas informáticos mencionados fueron creados con las características propias de cada compañía o institución que los implementó, además de responder a ciertas disposiciones que justifican el propósito de su creación, tales como: el objetivo especificado por un mercado, las herramientas que pueden usar, la estructura definida y en algunos casos el lenguaje en que se programa, que son privativos. Dichos sistemas utilizan mayormente sus funcionalidades para los registros de datos con referencia a la seguridad y protección de los sistemas al que se integra. Suria Vision posee diversas características propias, que lo diferencia de los restantes productos de video vigilancia o de cualquier otro tipo, la fundamental es que acepta diversos pluggins para cámaras, definiendo una arquitectura variada y con el propósito de que en el sistema se pueda hacer uso de diversos tipos de cámaras.

La estructura definida en la implementación de Suria Vision exige que el componente generador de logs y reportes se implemente de acuerdo a las funcionalidades que englobe y a la necesidad definida por los administradores de dicho sistema. Las aplicaciones antes mencionadas están limitadas por especificaciones propias de cada compañía que las creó, por lo que se puede decir que es poco factible adquirir una de estas aplicaciones que están desarrolladas en otro esquema que habría que adaptar a las características del sistema Suria Vision.

1.6. Conclusiones

- Mediante los métodos empíricos utilizados como: la entrevista y la observación, se logró reconocer la situación existente en el proyecto Video Vigilancia, con respecto a la no existencia de una herramienta o aplicación capaz administrar los sucesos generados durante la fase de implementación y despliegue del sistema Suria Vision.
- Se logró mediante la investigación realizada en bibliografías segura del tema, obtener el conocimiento necesario del funcionamiento que demuestran actualmente los sistemas generadores de logs y reportes, logrando una mejor comprensión del objeto de estudio de dicha solución.
- Al cumplir los métodos teóricos planteados como: el analítico-sintético y el histórico-lógico, se logró profundizar en los conceptos fundamentales que engloban los sistemas de video vigilancia y los

procesos para generar logs y reportes. Al realizar un estudio detallado de soluciones informáticas que se encargan de dicha actividades, se obtuvo características y funcionalidades importantes a incluir en el componente generador de logs y reportes para Suria Vision y se comprobó que dichas aplicaciones no puedes ser utilizadas para darle solución a la problemática existente en el proyecto Video Vigilancia.

CAPÍTULO 2: TECNOLOGÍAS, METODOLOGÍAS, HERRAMIENTAS Y CARACTERÍSTICAS DEL SISTEMA

2.1. Introducción

En este capítulo se caracterizan las herramientas, la metodología y las tecnologías a utilizar para el proceso de desarrollo del software. Se explica las ventajas que brindan y se valora el objetivo general que cumplen, reflejando la importancia de utilizarlas para el desarrollo de la solución propuesta; mediante una descripción detallada se gestiona conceptos y términos necesarios para entender la razón por la cual se eligen y qué consecuencia tiene realizar el software sobre esta base. Se realiza la propuesta del modelo del dominio, describiendo el flujo de relación entre los componentes más importantes y una detallada descripción de las clases presentes. Se especifican los requisitos funcionales y los no funcionales que la aplicación deberá contener y en el modelo del sistema se especifican los actores junto con los diagramas de casos de uso y las descripciones.

2.2. Metodología de desarrollo

Las metodologías de desarrollo del software constituye la combinación de modelos convencionales que representan guías que planifican y organizan el trabajo que se realiza para obtener la calidad requerida por el proyecto final. Estas permiten que se realice los procesos y las actividades definidas en el tiempo planificado para la construcción del producto y que se administre de forma correcta los recursos disponibles para el desarrollo de cualquier aplicación informática (Pressman, 2002).

Metodología de software pesada RUP[7]

RUP se define como una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas y una arquitectura configurable, es el conjunto de procesos que dan guía para conducir las actividades de desarrollo del equipo, permite seleccionar el conjunto de componentes de proceso que se ajustan a las necesidades específicas del proyecto. El **Rational Unified Process** unifica todo el equipo de desarrollo de software y optimiza su comunicación proveyendo a cada miembro de una aproximación al desarrollo de software con unas bases de conocimiento. La base de conocimiento unifica aún más al equipo identificando y asignando responsabilidades, artefactos y tareas de forma que cada miembro del equipo comprenda su contribución y papel en el proyecto. **RUP** se mantiene al equipo enfocado en

[7]Proceso Unificado de Rational (*Rational Unified Process*)

producir incrementalmente software operativo a tiempo, con las características y la calidad requerida (Rational, 2007).

La metodología **RUP** en ciertas ocasiones se considera un proceso demasiado costoso y difícil de realizar, por la cantidad de responsabilidades, artefactos y tareas a entregar o asignar que se definen en ella, implica que se genere un flujo constante de informaciones necesarias para la complementación y construcción de las diversas funcionalidades que contiene el producto que se quiere realizar utilizando como guía dicha metodología. **RUP** es un modelo que identifica cuatro fases diferentes en el proceso del software (inicio, elaboración, construcción y transición), dentro de estas fases se llevan a cabo un conjunto de actividades que se denominan flujos de trabajo. Existen seis principales flujos de trabajo identificados en el proceso y tres flujos de trabajo de soporte. **Rational Unified Process** presenta tres características importantes que definen su estructura básica:

1. Guiado por los casos de uso, porque especifica la importancia que tiene la comunicación con el cliente y los métodos encaminados en el punto de vista de este, con respecto a un sistema, permitiendo que se inserten las opiniones e ideas de estos en el diseño e implementación de las funcionalidades.
2. Centrado en la Arquitectura, pues enfoca las actividades y las acciones en las metas correctas, ayudando a organizar y estructurar los esfuerzos de los integrantes y ayuda al ajuste en las configuraciones y modificaciones en los cambios futuros.
3. Iterativo e incremental pues sugiere un flujo de procesos iterativo e incremental y proporciona el sentido evolutivo en el desarrollo de los softwares (Pressman, 2007).

El sistema Suria Vision es extenso y contiene una gran cantidad de funcionalidades por lo que se definió para el desarrollo del componente generador de logs y reportes el uso de la metodología **RUP**, para lograr almacenar artefactos y documentos que permitan describir y explicar de forma detallada el funcionamiento de los diversos eventos que engloba un producto de estas características; además de que se realizó una investigación con respecto a los resultados que obtienen diversos software durante su implementación, al hacer uso de la metodología mencionada anteriormente, comprobándose la efectividad que obtiene este proceso guía en aplicaciones de esta envergadura. Se seleccionó esta metodología porque reúne todos los modelos de procesos genéricos, durante su uso se especifica las mejores prácticas en el modelo de desarrollo de software y su objetivo es producir productos de alta calidad. El componente generador de logs y reportes necesita administrar ciertos artefactos y documentos que permita guardar las

características y la descripción de sus funcionalidades para que otras personas que trabajen en el proyecto Video Vigilancia o en tiempo posteriores quieran realizar ciertas modificaciones al producto, entiendan sus objetivos y tenga resguardos para poder realizar dichas acciones.

2.3. Lenguaje modelado

UML 2.0

Es importante señalar aparte de la metodología de software a utilizar para el desarrollo de la solución, el tema de crear abstracciones entendibles para describir las características y la conducta del producto a implementar, para definir y modelar de forma correcta las funcionalidades y eventos que necesita contener el componente generador de logs y reportes. El proceso mencionado se refiere al lenguaje de modelado, el cual crea modelos entendibles para los integrantes del proyecto, de los diversos eventos necesarios para implementar un producto de este tipo, permitiendo que todo el personal entienda los procesos involucrados y conozca previamente las características y distinciones necesarias para crear un programa correctamente y en el tiempo especificado. Al hacer uso del método teórico ¨modelación¨ permite tener un conocimiento amplio de las diversas funcionalidades que necesita un componente generador de logs y reportes para Suria Vision.

UML[8] es un lenguaje estándar para la escritura de los planos de un software, puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema intensivo. Genera abstracciones de las diversas funcionalidades y acciones a realizar durante la implementación de cualquier producto informático (Larman, 1999). Este lenguaje estándar se utiliza para describir un sistema, incluyendo aspectos conceptuales tales como: procesos de negocios, funciones del sistema y especificaciones concretas como: expresiones en los lenguajes de programación, esquemas de bases de datos y compuestos reciclados. Se puede aplicar en el desarrollo de software gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como RUP), pero no especifica en sí mismo qué metodología o proceso usar.

Para la implementación de la solución a realizar, es necesario el uso de UML, pues de manera entendible y concreta se explica y define los procesos imprescindibles para el correcto funcionamiento de un componente generador y administrador de logs y reportes. El lenguaje apoya la metodología que se define

[8] Lenguaje Unificado de Modelado (*Unified Model Language*)

como guía para diseñar todo el flujo de trabajo y actividades durante la fase de implementación de dicha solución, aportando fluidez durante las diversas etapas y mejorando el entendimiento que alcanzan los integrantes del proyecto Video Vigilancia con respecto a los continuos sucesos que ocurren durante el cronograma de trabajo. Es utilizado UML en la realización del módulo pues la metodología RUP se basa en ese lenguaje de modelado para la creación de sus artefactos y los modelos creados sirven como documentación técnica del componente para obtener un mejor entendimiento del mismo.

2.4. Lenguajes de Programación

Es el lenguaje que utilizan las computadoras para interpretar las instrucciones que reciben y para que las personas puedan controlar el comportamiento de las máquinas en general, está conformado por una serie de reglas sintácticas y semánticas que serán utilizadas por el programador y a través de las cuales creará un programa o sub programa. Los lenguajes de programación pueden clasificarse de diversas manera, como por ejemplo según su nivel de abstracción: lenguaje de bajo nivel (es el código fuente de la máquina, es decir el que la máquina puede interpretar); lenguaje de nivel medio (un término entre el lenguaje de la máquina y el lenguaje natural) y lenguaje de alto nivel (los que están compuestos por elementos del lenguaje natural, es decir el humano, especialmente el inglés). La particularidad es que ese lenguaje que utiliza le permite hacer las especificaciones en forma precisa, esto significa que todo se interpreta de la misma manera, sea quien fuere el programador que lo realice (Lanzillotta, 2012).

C++

Es un lenguaje imperativo orientado a objetos derivado del **C,** que se mantiene ligado al hardware subyacente, manteniendo una considerable potencia para la programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción (Ecured, 2013). Al lenguaje se le han añadido nuevos tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, funciones online, sobrecarga de operadores, referencias, operadores para manejo de memoria persistente, y algunas utilidades adicionales de biblioteca. Las características que definen a este súper conjunto, le permite crear diferentes aplicaciones o productos informáticos con funcionalidades potentes y adaptables a diversos sistemas operativos y aplicativos, permitiendo la versatilidad y la potencia necesaria para el correcto funcionamiento de los diversos programas que desarrollen con dicho lenguaje (Mastermagazine, 2012).

En el desarrollo de cualquier aplicación informática el lenguaje c++, permite realizar operaciones centradas en el proceso orientado a objetos y el trabajo con los punteros, permitiendo asignar las zonas de memorias a guardar por el implementador y cuando deseen eliminarlas, manteniendo el control del flujo de programación enfatizado en las acciones que realizan los diversos usuarios. En la implementación de esta solución brinda ventajas tales como: permite que el tiempo de ejecución sea menor, sea más potente y pueda realizar los eventos de **señales** y **slots**; dando opciones de conectar cada clase y estructura de manera dinámica en el transcurso de la ejecución de dicho componente. Se utiliza dicho lenguaje por ser el nativo en entorno de desarrollo a utilizar y por ser utilizado globalmente por software de programación que trabajan en plataformas libres.

2.5. Plataformas, Herramientas de desarrollo

Visual Paradigm 8.0

Es una herramienta CASE[9] que propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Visual Paradigm utiliza UML como lenguaje de modelado para soportar el ciclo de vida del proceso de desarrollo del software a través de la representación de todo tipo de diagramas y componentes. Dicha herramienta fue diseñada para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos. Se caracteriza por la disponibilidad en múltiples plataformas (Windows, Linux) y que su diseño es centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad. Constituye un herramienta en software libre probada para todo el proceso del diseño y análisis en el desarrollo de cualquier software.

Se utiliza Visual Paradigm 8.0 pues está basada en software libre, es multiplataforma y brinda unas series de características que propician un conjunto de ayudas para todos los integrantes del equipo de desarrollo de cualquier aplicación informática. Esta herramienta permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento de este. Aumenta el conocimiento informático de una empresa ayudando así a la búsqueda de soluciones para los requisitos. También permite la reutilización del software, portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería del Software. Lo más importante lo constituye que

[9] Ingeniería de Software Asistida por Computador (*Computer Aided Software Engineering*)

es fácil de instalar, actualizar y la disponibilidad de múltiples versiones que responden a diversas necesidades (Pressman, 2002).

QT CREATOR como IDE

QT CREATOR 2.4

Qt es un marco de desarrollo integral con herramientas diseñadas para simplificar la creación de aplicaciones e interfaces de usuario para el escritorio, embebidos y plataformas móviles. Qt Creator IDE es un entorno multi-plataforma de desarrollo integrado que dispone de lo siguiente: apoyo a las metas de escritorio y móviles, editores de código para C + + y Java Script, integra diseñador de interfaz de usuario, proyecto y construcción de herramientas de gestión y depuradores, simulador para interfaces de usuario móviles y soporte para el control de versiones. Con Qt, se puede reutilizar el código de manera eficiente para trabajar en múltiples plataformas con una base de código. El sistema modular de C++, las bibliotecas de clases y las herramientas que posee dicho entorno para desarrolladores, fácilmente permite a los implementadores crear aplicaciones para una plataforma y generar y ejecutar el despliegue en otra plataforma (Digia Oyj, 2013).

Para la propuesta de esta solución se utiliza el QT CREATOR como entorno de desarrollo, por las características mencionadas anteriormente y por la política de migración establecida por la universidad y el país a software libre; además de las características descritas haciendo referencia al lenguaje C++ y aprovechando ventajas que brinda su uso. El framework QT muestra unas series de funcionalidades que lo especifican como un marco de desarrollo capaz y estable para la implementación dentro de su ámbito de aplicaciones con diversos objetivos. Importante también señalar que una de las ventajas que tiene es la capacidad de ser utilizada en multiplataforma, facilitando el trabajo con otros framework y otros marcos de desarrollo que puedan ser necesitados para incluir y modificar ciertas funcionalidades y eventos que el componente generador de logs necesite durante la implementación o cuando se quiera realizar otra versión.

2.6 Descripción de la Arquitectura del Sistema

La solución que se propone será integrada al sistema de video vigilancia Suria Vision, y, por tanto debe describir una arquitectura adecuada para profundizar en los diversos elementos que engloba Suria y por la gran cantidad de datos e información que genera. La necesidad del manejo de la arquitectura de un sistema de software nace con los sistemas de mediana o gran envergadura, que se proponen como

solución para un problema determinado. En la medida que los sistemas de software crecen en complejidad, bien sea por número de requisitos o por el impacto de los mismos, se hace necesario establecer medios para el manejo de esta complejidad, por lo que una arquitectura de software define la estructura del sistema (ERIKA CAMACHO, ABRIL – 2004). Por la importancia que especifica utilizar una arquitectura como guía de desarrollo se define a continuación el estilo y el patrón de arquitectura a utilizar para el despliegue del componente para la generación y visualización de los logs para Suria Vision.

Estilos Arquitectónicos

Shaw y Garlan (1996) definen ***estilo arquitectónico*** como una familia de sistemas de software en términos de un patrón de organización estructural, que define un vocabulario de componentes y tipos de conectores y un conjunto de restricciones de cómo pueden ser combinadas. Buschmann et al. (1996) definen ***estilo arquitectónico*** como una familia de sistemas de software en términos de su organización estructural. Expresa componentes y las relaciones entre estos, con las restricciones de su aplicación y la composición asociada, así como también las reglas para su construcción. Los estilos arquitectónicos fundamentan la relación de cada funcionalidad de manera estructurada y jerárquica, son artefactos de ingeniería importantes porque definen clases de diseño junto con las propiedades conocidas asociadas a ellos. Ofrecen evidencia basada en la experiencia sobre la forma en que se ha utilizado históricamente cada clase, junto con razonamiento cualitativo para explicar por qué cada clase tiene esas propiedades específicas (Carlos Reynoso, Marzo de 2004).

Para el desarrollo del componente para la generación y administración de los logs y los reportes asociados a estos se emplea un patrón de la familia de patrones encapsulados en los ***Estilos de Llamada y Retorno***. Es utilizado porque enfatizan en la disponibilidad y la escalabilidad, pues permite organizar las estructuras y componentes fundamentales de manera dinámica y porque son los estilos más generalizados en sistemas en gran escala. Son las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, los sistemas orientados a objeto y los sistemas jerárquicos en capas.

Patrón Arquitectónico

Arquitecturas en Capas

Garlan y Shaw definen el estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la

inmediatamente inferior. Las capas suelen ser entidades complejas, compuesta por varios paquetes o subsistemas. Se encarga de dividir el peso de las funcionalidades en diversas capas, fundamentando así las responsabilidades y permitiendo separar los diversos eventos manifestando un trabajo adecuado con la lógica del negocio y las interfaces para los usuarios. Cada sistema exige un número de capas de acuerdo con la cantidad de funcionalidad y de representación pueda incurrir, definiendo el trabajo y los protocolos de comunicación que se necesita para que cada capa pueda comunicarse con la superior, los cambios en las capas de bajo nivel tienden a filtrarse hacia las de alto nivel, en especial si se utiliza una modalidad relajada; también se admite que la arquitectura en capas ayuda a controlar y encapsular aplicaciones complejas.

Existen diversas ventajas que definen a dicho estilo:

- El estilo soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.
- El estilo admite muy naturalmente optimizaciones y refinamientos.
- Proporciona amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces.

Para la implementación del componente para la generación y visualización de logs para Suria Vision se emplea una arquitectura de 2 capas, donde se divide la parte lógica del interfaz que se utiliza para representarle al usuario el producto. La parte lógica representa la biblioteca para gestionar todos los datos que contienen los sucesos (dígase salvarlo en logs, borrar dichos logs, visualizarlos pues la información estará cifrada dentro del fichero que se almacena). Mientras que la parte visual es la herramienta que tiene incluida la misma biblioteca e invoca las funcionalidades de visualizar y borrar los logs, manteniendo la programación alejada de los usuarios y haciendo uso del encapsulamiento para una mejor cohesión y organización en el diseño de dicho sistema. Posibilitando que durante el funcionamiento no existan interferencias en el código de las diversas funcionalidades y privando a los clientes del permiso de modificar y cambiar argumentos, métodos y otras acciones.

2.7 Modelo del Dominio

El Modelo del Dominio representa la capacidad de identificar y reconocer la lógica del funcionamiento de un determinado sistema, comienza como paso previo al diseño. Es un medio para comprender el sector de negocios al cual el sistema va a servir y se obtiene desde la base de conocimientos de unos pocos

expertos del dominio o posiblemente del conocimiento asociado con sistemas similares al que estamos desarrollando. Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes software, se denomina como la etapa orientada a objetos esencial del análisis o investigación, es la descomposición de un dominio de interés en clases conceptuales individuales u objetos, referentes a las cosas de las que somos conscientes. El modelo del dominio, podría considerarse como un *diccionario visual* de las abstracciones relevantes, se comporta como el vocabulario o conjunto de información del negocio (Larman, 1999).

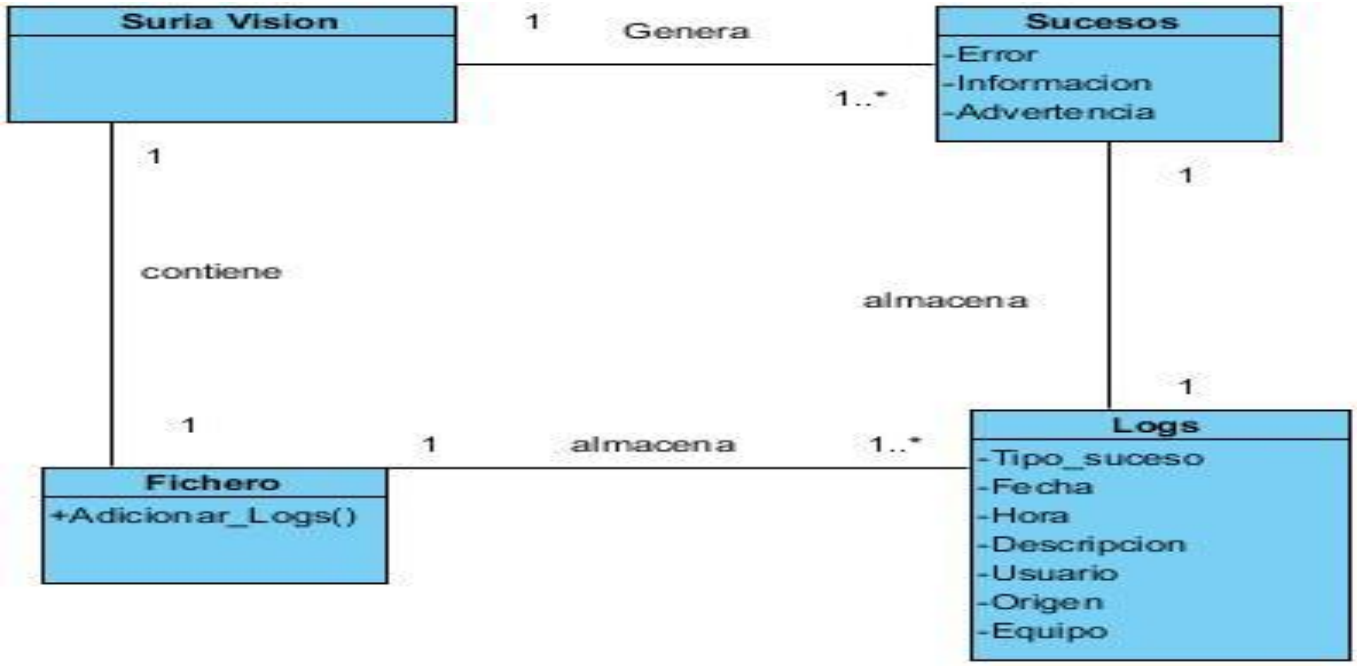


Ilustración 4 Modelo del Dominio

Descripción del flujo

Para el conocimiento acertado acerca del funcionamiento que abarca algunos sistemas informáticos que sus objetivos se concentran en la generación y administración de logs, se muestra el modelo del dominio, donde se realiza una breve explicación acerca del funcionamiento que engloba y como se manifiesta las relaciones necesarias y suficientes para el mejor entendimiento de los requisitos tantos funcionales como no funcionales a desarrollar. Las relaciones implícitas en dicho modelo mejora el análisis de las diferentes funcionalidades a implementar y describe de manera gráfica los diversos eventos que garantizan una solución factible y necesaria para darle respuesta a la problemática existente en el proyecto Video Vigilancia, con respecto a la nueva migración que se experimenta a software libre del producto

desarrollado llamado Suria Vision, basándose en las experiencias obtenidas por la versión actual de dicho producto que contiene un sistema que realiza las actividades referidas.

Suria genera sucesos, los cuales pueden comportarse como: errores, informaciones y advertencias, los cuales son almacenados en logs, que son los registros informáticos encargados de administrar los sucesos que ocurren durante sus diversas etapas (tanto en ejecución, como en la implementación). En el caso de la solución a implementar se necesita un solo fichero el cual almacena todos los logs generados por Suria Vision, en dichos archivos se almacenan los diversos datos de forma estructurada y relacional. Se dispone de reportes para acceder a la información almacenada en los ficheros, por lo que de un archivo pueden generarse uno o varios reportes, en los cuales pueden estar contenidos de uno a varios logs. Suria Vision al igual que los diferentes productos informáticos puede acceder automáticamente a los diversos sucesos almacenados en los registros, haciendo uso de los diversos componentes mencionados anteriormente y realizando correctamente los eventos que engloba generar y administrar los logs.

Descripción de las clases

- Se denominará ***Suria Vision*** al sistema que su propósito radica en las funciones de video vigilancia, donde una de sus funcionalidades resulta ser generar los sucesos que ocurren durante su etapa de implementación y su etapa de ejecución.
- Se denominará ***Sucesos*** al conjunto de datos o informaciones que se genera durante diversas etapas de vida de cualquier sistema informático y que guardan configuraciones, estados, descripciones y unas series de características importante para medir el correcto funcionamiento.
- Se denominará ***Logs*** al conjunto relacional y jerárquico de sucesos almacenados y administrados.
- Se denominará ***Fichero*** al recurso informático que permitirá almacenar de forma estándar toda la información contenida por los registros o logs.
- Se denominará ***Reporte*** a los eventos automatizados que permitan visualizar todos los datos que engloban los diversos sucesos.

2.8 Requisitos Funcionales

Después de analizar los conceptos fundamentales que describen al objeto de estudio, se analiza las acciones que el sistema debe ser capaz de realizar para darle cumplimiento a los objetivos planteados. Los requisitos funcionales son declaraciones de los servicios que el sistema debe proporcionar,

aplicándose a entradas particulares y situaciones en que se debe mostrar correctamente los resultados deseados, describen el objetivo fundamental que caracteriza la creación del sistema como tal (Somerville, 2005). A continuación se especificará los requisitos fundamentales para el correcto funcionamiento y que propicie el cumplimiento del objetivo planteado, estos representan las funcionalidades que deberán regir el proceso que describirá al componente generador y administrador de logs durante su tiempo de implementación y ejecución.

- **RF1 Cargar archivo de configuración del sistema:** Esta funcionalidad deberá permitir al componente cargar un archivo donde se encuentre la configuración que describe que tipo de suceso se desea administrar en los logs.
- **RF2 Cifrar la información generada por los diversos sucesos:** Esta funcionalidad deberá permitir cifrar los datos que contienen los diversos sucesos para escribirlos en el fichero de forma segura.
- **RF3 Descifrar la información almacenada en los ficheros:** Esta funcionalidad deberá permitir descifrar los datos de los sucesos que se encuentran almacenados en los ficheros.
- **RF4 Escribir los logs en un fichero:** Esta funcionalidad deberá permitir escribir los logs que contienen la información cifrada de los distintos sucesos en un fichero con formato txt, para administrar los datos de dichos sucesos en una unidad de almacenamiento.
- **RF5 Visualizar los datos que contienen los logs que están almacenados en los ficheros:** Esta funcionalidad deberá permitir acceder al contenido dentro de los ficheros y poder visualizar todos los datos que describen a los sucesos, que se encuentran estructurados dentro de los logs.
- **RF6 Visualizar los datos de los sucesos por criterios de búsqueda:** Esta funcionalidad deberá permitir buscar en los logs la información de los sucesos y visualizarla por criterios de búsquedas.
- **RF7 Generar reportes:** Esta funcionalidad deberá permitir generar reportes con los datos de los sucesos, específicamente ficheros de formato pdf.
- **RF8 Eliminar Logs:** Esta funcionalidad deberá permitir eliminar los datos de los logs que ya no sean útiles para el sistema, ya sea por el tiempo que llevan almacenado o por que se refiera a una funcionalidad que no sea necesaria**.**
- **RF9 Autenticar usuario:** Esta funcionalidad deberá permitir autenticar los usuarios que deseen invocar a las diversas funcionalidades.

2.9 Requisitos no funcionales

Los requisitos no funcionales se denominan las restricciones de los servicios ofrecidos por el sistema, donde se incluyen las restricciones de tiempo, de estándares, de patrones y sobre el proceso de desarrollo del producto. Son las propiedades emergentes como: la fiabilidad, tiempo de respuesta, la usabilidad, eficiencia, soporte y la capacidad de almacenamiento. Dichas propiedades constituyen accesorios fundamentales para la interfaz, la rapidez y los elementos que representan un sistema factible y seguro, además de especificar los requisitos básicos que deberá cumplir el sistema para un correcto funcionamiento. Los requisitos no funcionales representan los datos que hacen posible un mejor entendimiento de las funcionalidades específicas de cualquier sistema y hacen más entendible los resultados obtenidos (Somerville, 2005). En este sub-epígrafe se expondrán los requisitos no funcionales que debe cumplir el componente propuesto.

Requisitos de usabilidad

- El componente deberá ser usado solamente por los integrantes del proyecto Video Vigilancia o por los clientes que incluyan el producto a su sistema informático.
- La información deberá estar disponible en el período de trabajo definido por los dirigentes del proyecto Video Vigilancia, limitada solamente por las restricciones de acuerdo a las políticas de seguridad definidas.

Requisitos de fiabilidad

- El componente deberá estar disponible para los integrantes antes mencionados en el período de trabajo definido por los dirigentes del proyecto Video Vigilancia, solo no lo estará cuando se le esté realizando mantenimiento o suceda alguna inconveniencia externa.
- El tiempo de reparación de fallo ante alguna incidencia al componente será determinado por el equipo de desarrollo del proyecto Video Vigilancia, se recomienda preferentemente un día, debido a la importancia que tendrá dicho componente para el sistema Suria Vision.

Requisitos de confiabilidad

- A la aplicación que se realizará con el fin de visualizar los logs se accederá a través de la autenticación convencional: usuario y contraseña.

- Cada usuario deberá tener solo los permisos necesarios para realizar las operaciones que le sean permitidas en la aplicación.

Requisitos de eficiencia

- El componente generador de logs para Suria Vision estará conformado por una biblioteca y una aplicación que lea del fichero a guardar, por lo que el tiempo de eficiencia varía según la actividad que se esté realizando en ese momento. La biblioteca deberá escribir en el fichero según se le haga peticiones tanto en ejecución como en la implementación de Suria, por lo que debe oscilar entre 2 a 5 segundos. Mientras que en la aplicación se estima que el tiempo sea de 4 a 8 segundos, dependiendo de la cantidad de datos que estarán almacenados en los ficheros.

Requisitos de soporte

- Las restricciones y el tiempo de soporte será establecido por el equipo de desarrollo y los clientes que utilicen el sistema, se recomienda preferentemente 1 día, debido a la importancia que tendrá dicho componente para el sistema Suria Vision.

Requisitos de interfaz

- La herramienta para visualizar los logs deberá tener indicadores que permitan conocer al usuario las acciones que debe realizar, por ejemplo botones con íconos sugerentes y alternativa textual.

- La herramienta para visualizar los logs deberá permitir a los usuarios primero realizar los eventos que él desee, sin la necesidad de realizar actividades no deseadas, debe mantener una estética y permitir al cliente acceder a las funcionalidades sin dar más de 2 clics.

Requisitos de hardware

- El componente deberá estar instalado en una computadora que tenga memoria **RAM** 512 **MB** o más, velocidad de procesamiento del microprocesador 1**GHz** o superior y un disco duro de 80 o más **GB** de capacidad para poder almacenar el fichero que genera el componente.

Requisitos de software

- Sistemas operativos: Microsoft Windows 2000/NT o superior, distribución de GNU/Linux.

- Instalación del framework Qt Creator en las versiones 4.8 y las superiores para realizar nuevas modificaciones o nuevas funcionalidades al componente.

Requisitos Legales, de Derecho de Autor y otros

- Como producto, el componente generador de logs y reportes se distribuirá amparado bajo las normativas legales establecidas en el registro comercial emitido por las entidades jurídicas de la Universidad de las Ciencias Informáticas.
- El componente deberá regirse por la ley y los decretos establecidos, que norman los diversos procesos que son automatizados.

2.10 Modelo del sistema

El Modelo del Sistema es la representación gráfica que describe los procesos de negocio, la tónica del problema a resolver y las características del software a desarrollar. Mediante una abstracción de las diversas funcionalidades y eventos que engloba el componente a desarrollar se resalta de forma relevante las características más importantes que describe al sistema. Constituye una explicación detallada que representa un puente entre el proceso de análisis y diseño. Es el momento donde se especifica las diversas acciones que se realiza en cualquier sistema por parte de los actores reconocidos y la respuesta por parte del sistema a dichas situaciones. Se consolida el entendimiento del mismo a través de la definición de los procesos del negocio y de las personas involucradas en dichos procesos, mostrando los flujos tantos normales y alternos que ocurren durante la ejecución del producto. A continuación se describe el Modelo del Sistema.

Diagrama de casos de uso del sistema

Un Caso de Uso del Sistema (CUS) es un fragmento de una funcionalidad que el software ofrece para aportar un resultado de valor para sus actores. En ellos se agrupan algunos requisitos funcionales, se especifica las acciones o actividades que caracterizan la interacción actor-sistema, mostrando un flujo de eventos que describen gráficamente los procesos y condiciones indispensables para entender la correlación existente entre los diversos requisitos funcionales. Un caso de uso es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso y representa una secuencia de iteraciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de una aplicación mediante su interacción con los usuarios y/u otras aplicaciones (Addison Wesley Longman, 1992).

Un Diagrama de Casos de Uso del Sistema (DCUS) representa gráficamente a los procesos y sus interacciones con los actores. Los diagramas de casos de usos sirven para mostrar la interacción que ocurre entre los actores principales con las diversas funcionalidades que engloba un sistema determinado, en sí con los requisitos funcionales y muestra cómo reacciona estos a los diversos eventos que puedan ocurrir en el mismo ámbito. Mostrando cómo puede o cómo debe actuar el sistema antes diversas situaciones y cómo debe responder ante las acciones que realiza un actor determinado, ya sea cumpliendo con las funcionalidades principales o las respuesta antes incidentes que necesiten de una acción rápida y correcta por parte del software, mostrando a los implementadores que características y estructura debe implementar para darle una acertada solución a los diversos inconvenientes.

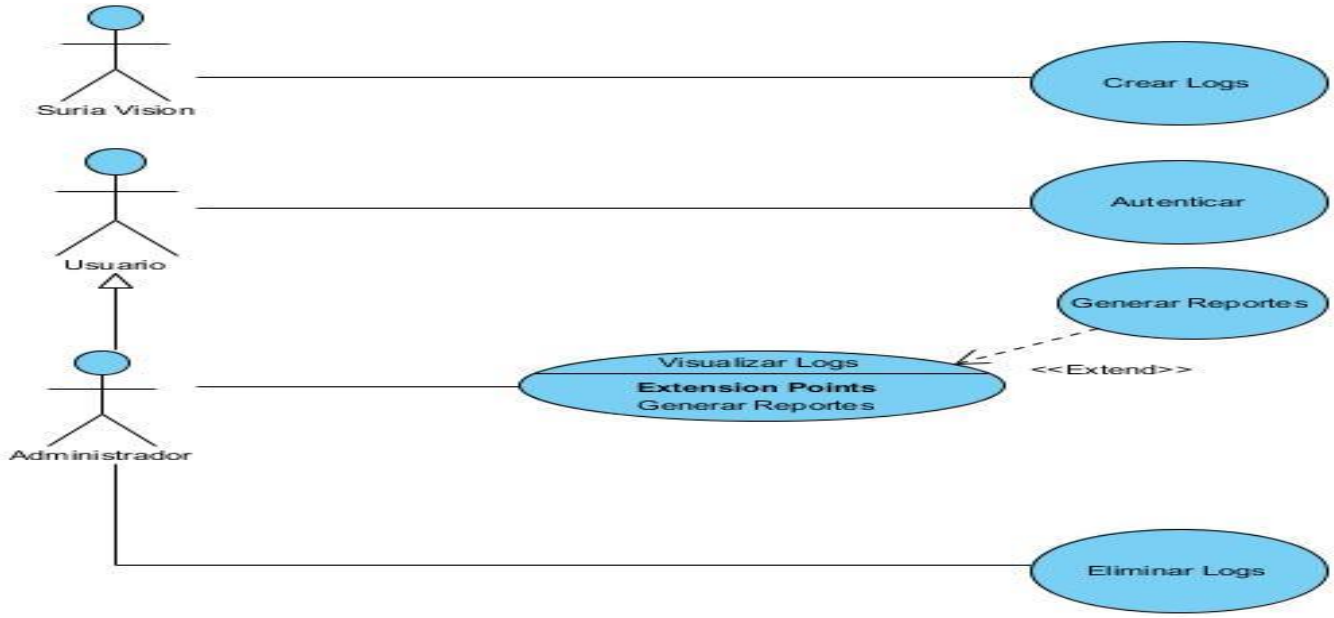


Ilustración 5: Diagrama de Casos de Uso del Sistema

Descripción de los actores

Actores del sistema: Son los trabajadores del negocio que poseen actividades a automatizar y puede comportarse como actores del negocio siempre y cuando interactúe con el software, influenciando en la inicialización de las funcionalidades o requisitos que engloban los eventos que existen en un determinado caso de uso, especificando la relación existente entre estos y el producto como tal (Jacobson, 2000). Un actor es cualquier dispositivo o aplicación que intercambia datos con el sistema y puede comportarse como un usuario, la diferencia entre ellos, es que el primero representa una clase concreta de cliente en

lugar de uno real. Varios usuarios pueden tener el mismo rol, lo que significa que pueden realizar las mismas acciones o actividades. Es vital distinguir las características propias de un actor y su papel, porque siempre tendrá interacción con el sistema ya sea inicializando el caso de uso o intercambiando información.

Luego de haberse definido los requisitos funcionales y los no funcionales que modelarán el componente a construir, corresponde definir que actores intervienen en el sistema y describir su responsabilidad en la creación de las actividades o acciones que representan o engloban los diversos casos de usos.

Actores	**Descripción**
Suria Vision	El actor Suria Vision es el encargado de inicializar el caso de uso Crear Logs, necesario para guardar los datos de los sucesos que ocurren durante las etapas de implementación y ejecución de él mismo.
Usuario	Es el actor que da inicio al caso de uso Autenticar, permitiendo que se autentique los permisos que debe tener un usuario en especifico.
Administrador	Es el actor que da inicio a los casos de uso Eliminar Logs, Visualizar Logs y Generar Reportes y, es el que posee el permiso adecuado para realizar las diversas funcionalidades.

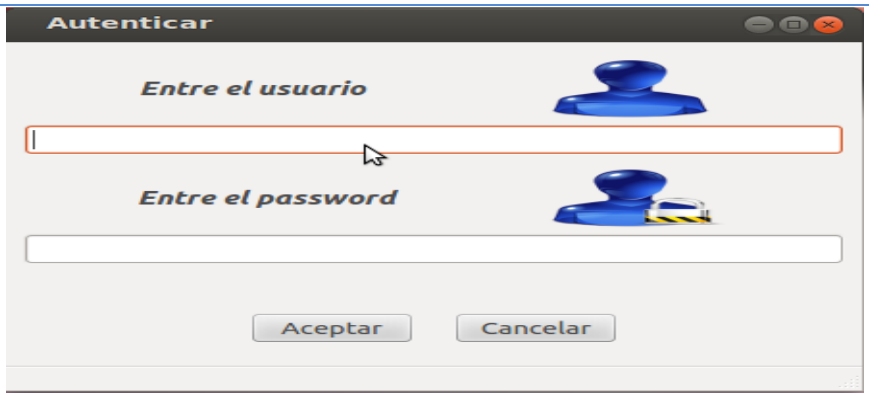
Tabla 1: Descripción de los actores

Descripción de los casos de usos

CUS Autenticar

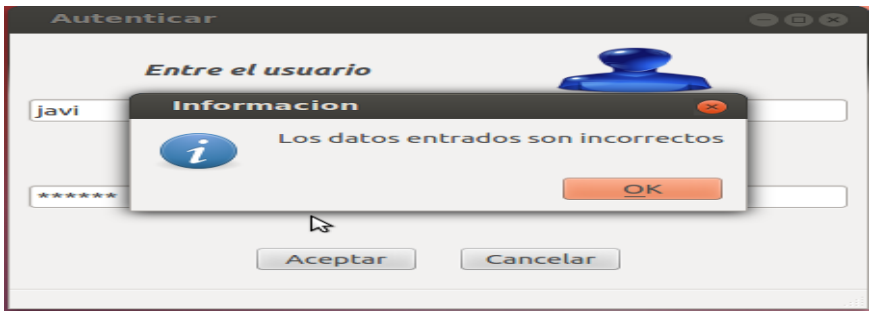
Objetivo	Autenticar el usuario que desea utilizar la herramienta para visualizar y eliminar la información que está cifrada en los ficheros.

Actores	Usuario	
Resumen	El caso de uso se inicia cuando el usuario ejecuta la herramienta para visualizar y eliminar la información que contiene los ficheros la cual está cifrada y la herramienta pide autenticarse.	
Complejidad	Alta	
Prioridad	Crítico	
Precondiciones	Debe existir almacenado el usuario y su respectiva clave para cuando el cliente ingrese la suya se pueda comparar.	
Postcondiciones	Después de autenticarse el usuario debe comportarse como administrador y la herramienta debe habilitar todas sus funcionalidades.	
Flujo de eventos		
Flujo básico: Iniciar Aplicación.		
	Actor	**Sistema**
1.	Ejecuta la herramienta para visualizar los datos de los sucesos.	
2.		Se ejecuta y levanta una ventana, donde se le pide al usuario que entre el nombre que tiene definido y la contraseña (Ver Interfaz 1).
3.	Llena los campos del usuario y la contraseña y presiona el botón de aceptar.	
4.		Compara los datos entrados con los que tiene definido y si son correctos se habilita las funcionalidades que contiene.
Interfaz 1		



Flujos alternos		
Contraseña y usuario incorrecto.		
	Actor	**Sistema**
1.	Introduce el usuario y la contraseña y presiona el botón aceptar.	
2.		Compara los datos entrados con los que tiene definido y si son incorrectos se muestra un mensaje de información (Ver Interfaz 2).

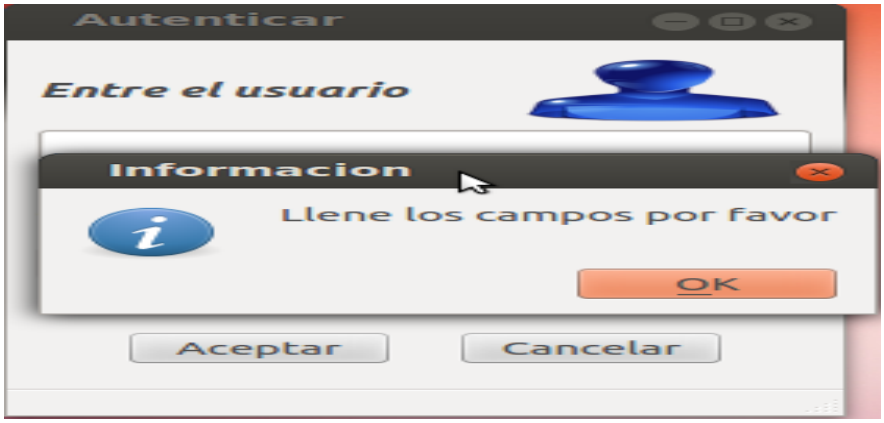
Interfaz 2



Flujos alternos		
Campos Vacíos		
	Actor	**Sistema**

1.	Presiona el botón aceptar sin haber llenado los campos correspondiente.	
2.		Muestra un mensaje de información automáticamente (Ver Interfaz 3).

Interfaz 3



Flujos alternos		
Cancela autenticación		
	Actor	**Sistema**
1.	Presiona el botón cancelar.	
2.		Cierra automáticamente.

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos funcionales	**RF9**	
Asuntos pendientes		

Tabla 2: CUS Autenticar

CUS Crear Logs

Objetivo	Cargar un archivo de configuración, para definir qué tipo de suceso desea el cliente administrar, donde se cifra los datos de los sucesos y con ellos se crea los logs y se guardan en un fichero.	
Actores	Suria Vision	
Resumen	Este caso de uso se inicia cuando Suria Vision es ejecutado, al adicionarse la biblioteca encargada para cumplir el objetivo de administrar los sucesos que ocurran durante la ejecución e implementación de dicho sistema. Al ocurrir un suceso el actor invoca el método para salvar los logs que está en la biblioteca.	
Complejidad	Alta	
Prioridad	Crítico	
Precondiciones	En Suria Vision debe estar adicionada la biblioteca con que se realiza dichas actividades y a su vez debe contar con un fichero que contenga la configuración que define que tipo de suceso se debe guardar.	
Postcondiciones	El componente queda actualizado luego de realizada cualquier acción descrita anteriormente y debe estar creado el fichero con los datos de los sucesos cifrados.	
Flujo de eventos		
Flujo básico: Crear Logs		
Sección 1: Cargar el archivo de configuración.		
	Actor	**Sistema**
1.	Invoca la biblioteca.	
2.		Se ejecuta y carga el archivo de configuración para definir los tipos de sucesos que se desea administrar.
Flujos alternos		

Archivo de configuración corrupto o que no exista.		
	Actor	**Sistema**
1.	Invoca la biblioteca.	
2.		Se ejecuta, carga el archivo de configuración para definir los tipos de sucesos que se desea administrar, si dicho fichero está corrupto, se encuentra en un formato que no lea la biblioteca o no exista, no debe guardar en el fichero la información que se generó.
Sección 2: Salvar los datos de los sucesos de forma cifrada en ficheros.		
	Actor	**Sistema**
1.	Ejecuta en el sistema la funcionalidad de la biblioteca que salve los datos de los sucesos que se generen y entra por parámetro el nombre que él desea para el archivo y los datos que se requieren para los sucesos.	
2.		Ejecuta dicha funcionalidad, mediante un algoritmo para encriptar, cifra la información y almacena en el fichero con el nombre especificado los datos cifrados, si ya existe el fichero nombrado, se le adiciona el contenido al final, si no existe se crea un nuevo archivo con dichos datos.
Flujos alternos		
Problema con el algoritmo para cifrar.		

	Actor	Sistema
1.	Ejecuta en el sistema la funcionalidad de la biblioteca que salve los datos de los sucesos que se generen y entra por parámetro el nombre que él desea para el archivo y los datos que se requieren para los sucesos.	
2.		Ejecuta dicha funcionalidad y ocurre cualquier error para cifrar, ya sea por la compatibilidad de la llave, por la compatibilidad del algoritmo con las diversas funcionalidades u otro percance que exista. El sistema no guarda la información en ningún fichero.
Flujos alternos		
Problema con el archivo o fichero para almacenar los datos de los sucesos.		
	Actor	**Sistema**
1.	Ejecuta en el sistema la funcionalidad de la biblioteca que salve los datos de los sucesos que se generen y entra por parámetro el nombre que él desea para el archivo y los datos que se requieren para los sucesos.	
2.		Ejecuta dicha funcionalidad, puede ocurrir que exista el fichero y cuando el sistema acceda a él, para almacenar los sucesos, halle que está corrupto por lo que el sistema no salva dicha información.
Relaciones	**CU Incluidos**	

	CU Extendidos	
Requisitos funcionales	**RF1, RF2, RF4**	
Asuntos pendientes		

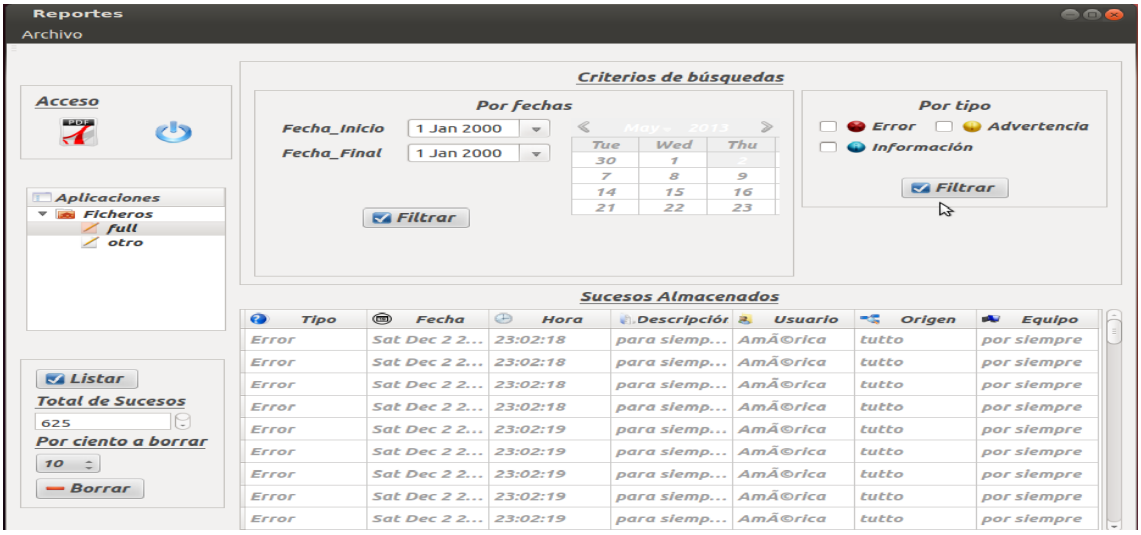
Tabla 3: CUS Crear Logs

CUS Visualizar Logs

Objetivo	Visualizar los datos que están cifrados dentro de los ficheros.	
Actores	Administrador	
Resumen	Este caso de uso se inicia cuando el administrador accede a las funcionalidades que dan solución a los requisitos funcionales de visualizar los datos cifrados de los sucesos.	
Complejidad	Alta	
Prioridad	Crítico	
Precondiciones	El administrador debe haberse autenticado correctamente y debe tener los permisos adecuados.	
Postcondiciones	Deben visualizarse todos los datos registrados en los logs.	
Flujo de eventos		
Flujo básico: Visualizar los sucesos administrados en los logs.		
Sección 1: Visualizar todos los sucesos.		
	Actor	**Sistema**
1.	Selecciona de la interfaz el nombre del fichero y presiona el botón de listar.	

2.		Se ejecuta, invoca de la biblioteca el método para visualizar los datos cifrados de los sucesos dentro del archivo con el nombre especificado, descifra el contenido y muestra todos los datos (Ver Interfaz 1).

Interfaz 1

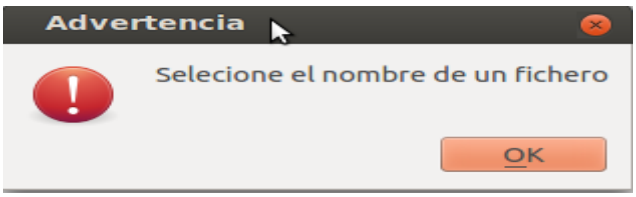


Flujos alternos

No se especifica por parámetro el nombre del fichero.

	Actor	Sistema
1.	Presiona el botón de listar.	
2.		Se ejecuta y muestra un mensaje explicando que tiene que seleccionar un nombre del fichero deseado (Ver Interfaz 2).

Interfaz 2



Sección 2: Visualizar los sucesos por fechas:

	Actor	Sistema
1.	Selecciona de la interfaz el nombre del fichero, la fecha inicial y la fecha final y presiona el botón de filtrar.	
2.		Se ejecuta e invoca de la biblioteca el método que permite visualizar los datos almacenados por fechas en un fichero, seleccionando el nombre del fichero, una fecha de inicio y una fecha final, carga el archivo con el nombre especificado, descifra el contenido y muestra todos los datos por el criterio de búsqueda específico. (Ver Interfaz 3).

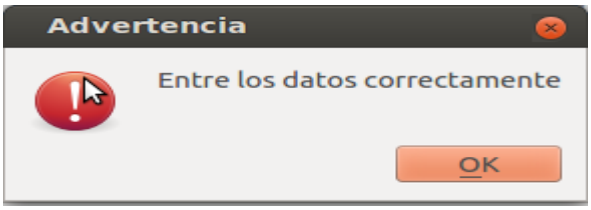
Interfaz 3

Flujos alternos

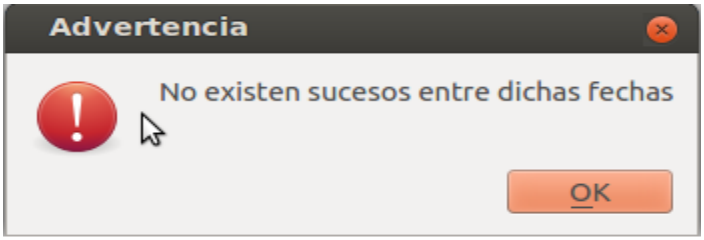
Fechas Incorrectas.

	Actor	Sistema
1.	Selecciona de la interfaz el nombre del fichero, la fecha inicial y la fecha final y presiona el botón de filtrar.	
2.		Se ejecuta y verifica que los datos entrados o seleccionados son incorrectos, por lo que muestra un mensaje de advertencia (Ver interfaz 4).

Interfaz 4

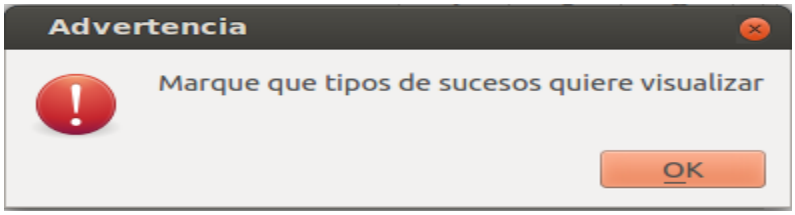


Flujos alternos		
No existen sucesos que tengan dichas fechas.		
	Actor	**Sistema**
1.	Selecciona de la interfaz el nombre del fichero, la fecha inicial y la fecha final y presiona el botón de filtrar.	
2.		Se ejecuta e invoca de la biblioteca el método que permite visualizar los datos almacenados por fechas en un fichero, seleccionando el nombre del fichero, una fecha de inicio y una fecha final, carga el archivo con el nombre especificado, descifra el contenido, al buscar por el criterio de búsqueda se percata que no

		existen sucesos con esas condiciones y muestra un mensaje de advertencia explicando el problema (Ver interfaz 5).
Interfaz 5		
		
Sección 3: Visualizar los sucesos por tipos de sucesos.		
	Actor	**Sistema**
1.	Selecciona de la interfaz el nombre del fichero y los tipos de sucesos y presiona el botón de filtrar.	
2.		Se ejecuta e invoca de la biblioteca el método que permite visualizar los datos almacenados por tipos de sucesos en un fichero, seleccionando el nombre del fichero y carga el archivo con el nombre especificado, busca por los tipos de sucesos seleccionados y muestra los datos pertinentes (Ver Interfaz 6).
Interfaz 6		

Flujos alternos		
No selección de los tipos de sucesos a mostrar.		
	Actor	**Sistema**
1.	Selecciona de la interfaz el nombre del fichero y presiona el botón de filtrar.	
2.		Se ejecuta y muestra un mensaje de advertencia indicando que debe seleccionar los tipos de sucesos que desea (Ver Interfaz 7).

Interfaz 7



Flujos alternos		
No existen sucesos dentro del fichero seleccionado que cumplan con los tipos deseados.		
	Actor	**Sistema**

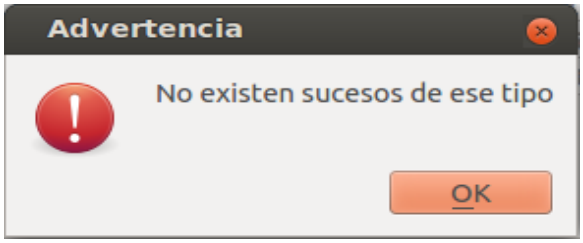
1.	Selecciona de la interfaz el nombre del fichero y los tipos de sucesos y presiona el botón de filtrar.	
2.		Se ejecuta e invoca de la biblioteca el método que permite visualizar los datos almacenados por tipos de sucesos en un fichero, carga el archivo con el nombre especificado, descifra el contenido, al buscar por los tipos de sucesos seleccionados se percata que no existen sucesos que cumplan dichos criterios, muestra un mensaje de advertencia explicando el problema (Ver interfaz 8).
Interfaz 8 		
Relaciones	**CU Incluidos**	
	CU Extendidos	**CUS Generar Reportes**
Requisitos funcionales	**RF3,RF5, RF6**	
Asuntos pendientes		

Tabla 4: CUS Visualizar Logs

CUS Eliminar Logs

Objetivo	Eliminar los datos que están cifrados dentro de los ficheros.	
Actores	Administrador	
Resumen	Este caso de uso se inicia cuando administrador accede a la funcionalidad que da solución al requisito funcional de eliminar los datos cifrados de los sucesos.	
Complejidad	Alta	
Prioridad	Crítico	
Precondiciones	El administrador debe haberse autenticado correctamente y debe tener los permisos adecuados.	
Postcondiciones	Debe visualizarse todos los datos registrados en los logs después de haberse eliminado ciertas informaciones.	
Flujo de eventos		
Flujo básico: Eliminar sucesos administrados en los logs.		
	Actor	**Sistema**
1.	Selecciona de la interfaz el nombre del fichero, el por ciento a eliminar y presiona el botón de borrar.	
2.		Se ejecuta e invoca de la biblioteca el método que permite eliminar los datos almacenados en un fichero, carga el archivo con el nombre especificado, descifra el contenido, entonces busca dentro de todos los datos los que mayor tiempo lleva almacenado en dicho fichero y los elimina, al terminar muestra un mensaje de éxito (Ver Interfaz 1).
Interfaz 1		

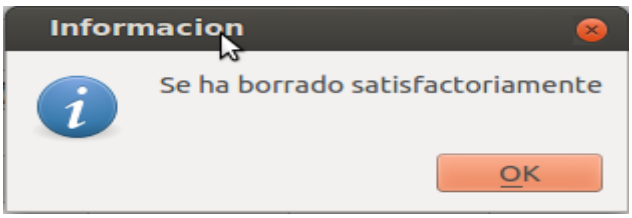
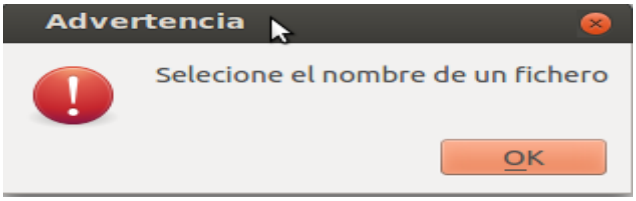
<table>
<tr><td colspan="4"> 
</td></tr>
<tr><td colspan="4">Flujos alternos</td></tr>
<tr><td colspan="4">No se especifica por parámetro el nombre del fichero.</td></tr>
<tr><td></td><td colspan="2">Actor</td><td>Sistema</td></tr>
<tr><td>1.</td><td colspan="2">Selecciona de la interfaz el por ciento a eliminar y presiona el botón de borrar.</td><td></td></tr>
<tr><td>2.</td><td colspan="2"></td><td>Se ejecuta y muestra un mensaje de advertencia explicando que tiene que seleccionar un nombre del fichero deseado (Ver Interfaz 2).</td></tr>
<tr><td colspan="4">Interfaz 2
 
</td></tr>
<tr><td rowspan="2">Relaciones</td><td>CU Incluidos</td><td colspan="2"></td></tr>
<tr><td>CU Extendidos</td><td colspan="2"></td></tr>
<tr><td>Requisitos funcionales</td><td colspan="3">RF8</td></tr>
<tr><td>Asuntos pendientes</td><td colspan="3"></td></tr>
</table>

Tabla 5: CUS Eliminar Logs

La descripción del caso de uso Generar Reportes, se encuentra en el expediente de proyecto.

2.11 Conclusiones

- Con el estudio de las características y ventajas que brindan algunas metodologías, herramientas y lenguajes de programación que trabajan en plataformas libres, se logró obtener las adecuadas para apoyar el ciclo de vida del proceso de desarrollo del componente generador de logs y reportes.
- Se logró establecer la arquitectura para estructurar la aplicación informática referente a la solución, después de haberse realizado la investigación a los estilos arquitectónicos y sus ventajas, verificando que las condiciones deseadas puedan ser estructuradas por el estilo o patrón elegido.
- Al estudiar las soluciones informáticas que se encargan del proceso de generar logs y reportes, tanto a nivel nacional como internacional, principalmente el componente que existe actualmente en el proyecto Video Vigilancia pero en plataforma privativa, se obtuvo un dominio de la lógica funcional que debe tener la solución y cómo funciona todo ciclo actualmente, llevando lo real a la lógica de clases conceptuales.
- Al realizarse varias entrevistas al jefe de proyecto y una población de los integrantes del equipo de desarrollo se especificaron los requisitos funcionales y los no funcionales, de acuerdo a las necesidades expresadas por estos, centrándose en las ventajas que les brinda actualmente el producto que se encuentra en plataforma privativa.
- Con la utilización de los métodos científicos antes mencionados se logró estructurar todas las funcionalidades y métodos que debe contener el componente y las relaciones que puede existir en los actores y el sistema como tal, definiendo las posibles respuestas que debe mostrar el componente ante cualquier tipo de acción que se le realice.

CAPÍTULO 3: CONSTRUCCIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.

3.1 Introducción

Para seguir los pasos en la construcción de la aplicación es necesario continuar por las demás fases del proceso de desarrollo de un software. En el capítulo presente se abordan los flujos restantes definidos por la metodología RUP: Análisis y Diseño, Implementación y Prueba. Con la descripción de las etapas antes mencionados se definen los Diagramas de Clases del Diseño, el Diagrama del Despliegue y el de Componentes, donde se definirá la estructura y los elementos básicos para la construcción de la solución y por último se especifica y realizan las pruebas pertinentes para corroborar que el producto satisface las necesidades planteadas durante el estado del arte. Los flujos que restan describen en sí la etapa de implementación, el diseño permite modelar los componentes fundamentales y guiar el proceso de desarrollo, mientras que en la fase de implementación y prueba se prioriza los elementos fundamentales que construye la aplicación en sí.

3.2 Modelo del Diseño

El Modelo del Diseño es físico y concreto, es un plano de la implementación, donde se especifica las características, componentes, las clases, los métodos y las interfaces, que describe la solución para lograr satisfacer las necesidades. Define la arquitectura del sistema y tiene como objetivos trasladar requisitos en especificaciones de programación, se refiere a refinar el análisis para poder implementar los diagramas de clases de análisis, de colaboración, el de clases y de secuencia del diseño de cada CU, el de estados de las clases y el Modelo de Despliegue de la arquitectura. Su esencia es la elaboración de diagramas de interacción**,** que muestran gráficamente como los objetos se comunicaran entre ellos a fin de cumplir con la realización de los requisitos. Dicho modelo es una representación de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de construcción, tienen impacto en el sistema a considerar (Jacobson, 2000).

La etapa de análisis permite el uso de una combinación de formatos entre textos y diagramas para representar los requisitos de los datos, las funciones y el comportamiento de una manera que es fácil de entender y aún más importante, conduce a una revisión para lograr la corrección, la integridad y la consistencia. El modelado del análisis representa los requisitos en múltiples dimensiones, con lo que se incrementa la probabilidad de encontrar errores, de que se descubra inconsistencias y de que se

descubran omisiones (Roger Pressman, 2013). El objetivo de la fase de análisis debe alcanzarse en todos los proyectos de una forma o de otra, pero la forma de hacerlo varía según las características de la aplicación. Al ser la metodología RUP un proceso configurable, por lo que no se está obligado a realizar todas las actividades, sino que se puede configurar el proceso para que realice las partes que se consideren necesarias, por lo que se decidió prescindir de la realización del **Modelo de Análisis** en el desarrollo de esta investigación debido a que:

- Los requisitos funcionales y no funcionales son sencillos y globalmente se contiene gran conocimiento de la forma en que se realizan o se implementan y que existen muchos ejemplos y aplicaciones donde se define dichas funcionalidades.
- Los requisitos funcionales se entienden correctamente y existe el conocimiento de la necesidad por lo que aparecieron.
- El negocio es entendible y los requisitos no funcionales se adaptan a las necesidades expresadas.
- Es posible obtener un mayor formalismo en el modelo de casos de uso pues el cliente es capaz de comprender los resultados que estos pueden proporcionar.
- No se requiere este artefacto en el proyecto Video Vigilancia.

Patrones de diseño

Los patrones de diseño constituyen la base y las características fundamentales para la búsqueda de soluciones a problemas comunes que existen durante el desarrollo o construcción de cualquier software. Dichos patrones guían mediante expresiones y elementos el proceso de desarrollar cualquier aplicación informática, haciendo uso de métodos y estructuras que hayan sido probadas en diversos sistemas y se pueda utilizar como un conjunto estructural de relaciones e interrelaciones de clases, componentes y módulos para abstraer la posible implementación de la solución a desarrollar y así poseer un mejor entendimiento del problema al que se le quiere dar solución. Al ser utilizados permite organizar los distintos componentes del sistema mediante un criterio de diseño (Larman, 1999). Para la implementación del componente generador de logs y reportes para Suria Vision, se hace uso de algunos patrones, entre ellos se encuentran.

Patrones generales de software para asignar responsabilidades (GRASP): Los patrones GRASP describen los diez principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Carlos Reynoso, Marzo de 2004).

- **Alta Cohesión:** Dicho patrón tiene como objetivo asignarle a cada clase la responsabilidad solo necesaria para su correcto funcionamiento, en sí no asignar más eventos a los que ya contiene y a la vez permitir que cada clase pueda colaborar con otras para realizar diferentes operaciones tales como: instanciar objetos y acceder a las propiedades, es decir, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas proporcionando que el software sea flexible frente a grandes cambios (Larman, 1999). Cada clase de la biblioteca aporta un evento para los diversos métodos que representan el cumplimiento de los requisitos funcionales. La clase controladora acumula todos los datos que brinda cada clase y realiza las acciones pertinentes, cumpliendo con la asignación que se realiza y colaborando con las operaciones de los diversos componentes.
- **Creador:** Es el patrón que ayuda a identificar quien debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases, basándose en reconocer si dicho creador tiene la información suficiente y necesaria para realizar dicha actividad, que use alguna instancia del objeto que cumple el rol de creado y que almacene o maneje varias instancias (Larman, 1999). En la clase *log_managment* de la biblioteca se encuentra en todos los métodos, la acción crear un objeto de *log_entidad*, por lo que cumple el rol de creador permitiendo crear un objeto donde se acceda a todas las propiedades y eventos. Al igual que en la herramienta para leer todos los sucesos, existe una clase creadora que se llama reportes la cual instancia todas las demás clases. Dichos creadores contienen toda la información necesaria para darle cumplimiento a dicho patrón.
- **Controlador**: Permite asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas, facilitando que se centre todos los métodos en una clase global permitiendo manejar todas las peticiones con un objeto perteneciente a esta. Este es utilizado en el componente ya que se cuenta con una controladora en la biblioteca y una en la herramienta encargada de interactuar con las restantes clases de la aplicación. Además dicho patrón exhorta a dividir la lógica de negocios de la capa de presentación, para aumentar la reutilización de código y tener un mayor control de las diversas funcionalidades (Larman, 1999). Al contenerse una gran cantidad de controladores en el componente generador de logs y reportes para Suria Vision permite aumentar la cohesión y disminuir el acoplamiento en dicha aplicación y permite delegar responsabilidad sobre otros objetos cuando se

encuentre sobrecargado.

- **Experto**: Está diseñado para que la responsabilidad de realizar una labor sea de la clase que tiene o puede tener los atributos involucrados, mantiene la responsabilidad dentro de las mismas clases que contienen los diversos atributos y las diversas propiedades, logrando que exista una alta cohesión y un mejor desempeño (Larman, 1999). En la herramienta del componente para la generación de logs y reportes existen clases donde se le aplica dicho patrón, las cuales contienen toda la información necesaria para realizar cualquier funcionalidad que requiera el uso de sus atributos o propiedades, en ambos lugares cada clase que contiene los medios necesarios tiene las funcionalidades o puede tenerlas en cualquier momento, por lo que se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento).
- **Bajo Acoplamiento:** Permite establecer las conexiones entre las clases solo cuando son necesarios, manteniendo un bajo acoplamiento en las diversas responsabilidades y que cuando se le realice alguna modificación a un objeto especifico, se tenga la mínima repercusión en el resto de los objetos que puedan contener al modificado y permite disminuir la dependencia entre estos. La clase log_*managment* hereda solamente de QObject para lograr un bajo acoplamiento de clases y existe tanto en la biblioteca como en la herramienta poca dependencia entre las diversas clases, manteniendo las relaciones con las respectivas clases controladoras y en la herramienta para la visualización mantiene las dependencias en la herencia de los componentes fundamentales, asegurando que se pueda extraer el software de forma independiente y reutilizable.

Diagrama de clases del diseño (DCD)

Es una descripción o abstracción lógica de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones. Permite identificar la especificación de las clases de software (y las interfaces) que participan en la solución de software y complementarlas con detalles de diseño, por ejemplo los métodos. Podemos bosquejar muchos componentes, nombres de métodos y relaciones al inicio de la fase de diseño, aplicando los patrones de asignación de responsabilidades. El diagrama de clases del diseño describe gráficamente las especificaciones de los contenidos de software y de las interfaces en una aplicación cualquiera y expresan para el sistema computarizado la definición de las estructuras de diseño como componentes del software (Ivar Jacobson, 2000).

A continuación se representarán los diagramas de clases del diseño para el componente que engloba todos los métodos representativos de los requisitos funcionales. Al aplicarse el patrón arquitectónico por capas, en su variante dos capas, se divide la lógica del negocio de la capa de presentación, por lo que la biblioteca contiene los métodos que cumplen los objetivos especificados y los requisitos funcionales y en la herramienta solo se incluye la biblioteca y después se realiza el llamado a las funcionalidades respectivas para la visualización de los sucesos. Los diagramas de clases de diseño se realizan respectivamente con los casos de usos del sistema, pero al utilizar la arquitectura antes mencionada la biblioteca da solución a los dos casos de usos existentes por lo que se decide realizar el diagrama tanto a la biblioteca como a la herramienta, para así especificar el diseño y que los administradores o personas que puedan trabajar sobre dicho sistema comprendan la estructura.

En la siguiente ilustración se representa los diversos métodos fundamentales para el cumplimiento pleno de los requisitos fundamentales para la visualización de dichos datos, como se utiliza la arquitectura dos capas, en la capa de la interfaz se incluye la biblioteca y todos las funcionalidades encapsuladas en esta, en la interfaz solo se limita a invocar los eventos necesarios y a capturar lo que devuelve para mostrarse simple y sencillo al usuario, sin la necesidad de que este se relacione directamente al código principal. Dentro de las clases que representan los diversos componentes se invoca la clase controladora de la biblioteca y en la controladora de la herramienta se instancia objetos de cada una de esas clases permitiendo utilizar el patrón creador y experto.

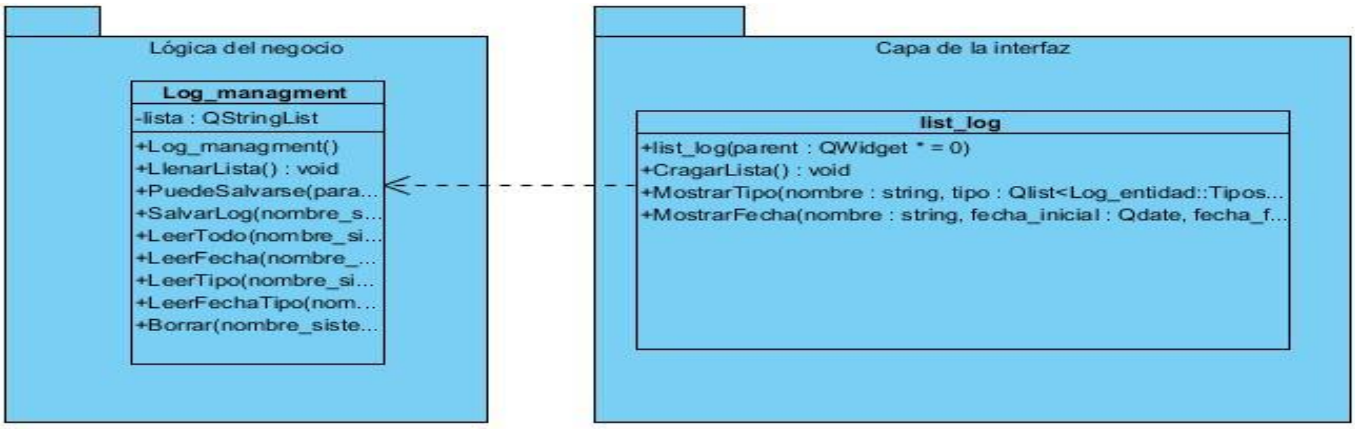


Ilustración 5: Diagrama de clases del diseño

El Diagrama de Clases del Diseño para la implementación de la herramienta tiene como objetivo la visualización de los sucesos cifrados en los diversos ficheros almacenados. La clase controladora de la

biblioteca es invocada dentro de la clase list_log, para acceder a todos los métodos que contiene la lógica del negocio, list_log se comporta como una clase controladora, experta y creadora por las diversas responsabilidades que ejecuta. Clase general llamada Reportes es la controladora de toda la lógica programada dentro de la herramienta, controlando el flujo de eventos entre todos los demás componente y participando de intermediaria entre el usuario y la biblioteca.

En la siguiente ilustración se muestra el diseño que representa la implementación en la biblioteca, definiéndose como clase controladora y creadora *log_managment*, en la cual se describe la realización de todos los métodos fundamentales para dar cumplimiento a los requisitos funcionales. La clase *log_entidad* es definida por el patrón experto, pues es la que contiene los atributos imprescindibles para almacenar los sucesos, con sus respectivas descripciones. Existe la clase *simplecrypt* es la encargada de construir el algoritmo para descifrar y cifrar todos los datos, convocándose como la experta para realizar dicha actividad, dentro de *log_managment* se instancia tanto *log_entidad*, como *simplecrypt* permitiendo que en cualquier aplicación externa necesite al incluir dicha biblioteca contener solamente un objeto de dicha clase para acceder y obtener los resultados que despliegan los métodos fundamentales.

Diagrama de interacción

El diagrama de interacción explica gráficamente las interacciones entra las diversas instancias de clases que existen en las diversas operaciones que definen las funcionalidades. UML define dos tipos de estos diagramas, los cuales sirven para expresar interacciones semejantes o idénticas de mensaje: de colaboración y de secuencia. Para describir la solución deseada se utiliza el de colaboración ya que permite expresar de forma contextual el tipo de visibilidad entre los objetos, resultando más simple para representar la lógica condicional y la concurrencia. Un diagrama de comunicación puede comportarse también como uno de clases, que contiene roles de clasificador y de asociación en lugar de sólo clasificadores y asociaciones, que sirven para describir la configuración de los objetos y los enlaces que puede ocurrir cuando se ejecuta alguna instancia, mostrando la implementación que describe su ejecución (Pressman, 2002).

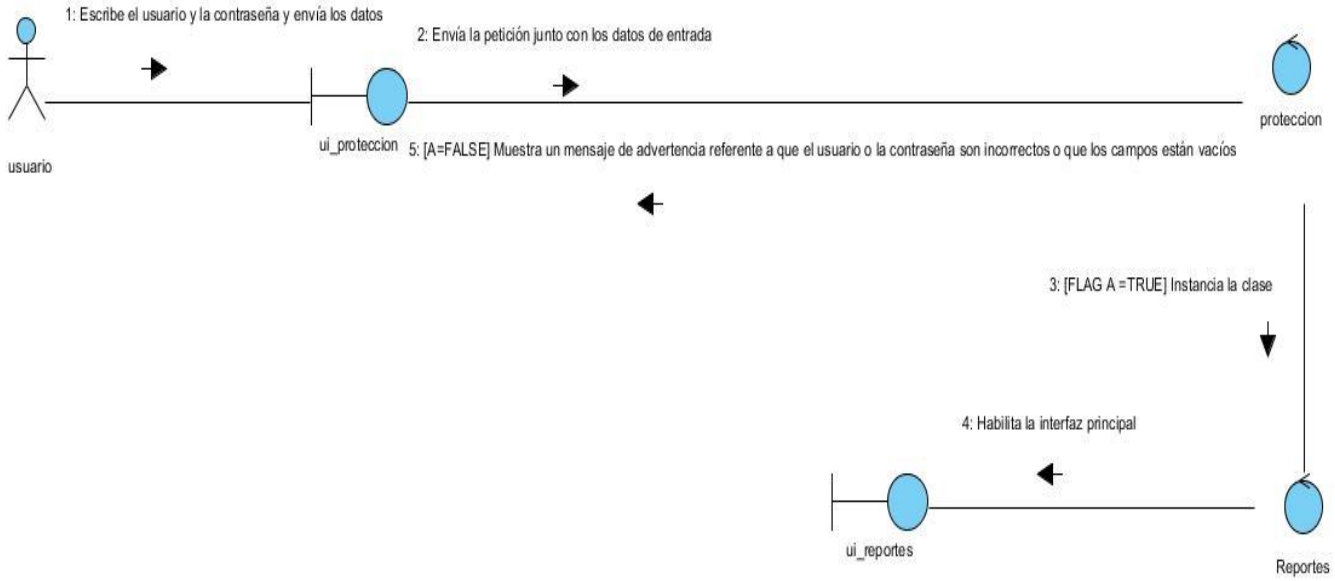


Ilustración 6: Diagrama de colaboración CUS Autenticar

Las restantes ilustraciones de los diagramas de colaboraciones se encuentran en el expediente de proyecto.

3.3 Modelo del Despliegue

Es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. En las actividades referentes al diseño e implementación representa la capa estructural que define el comportamiento de los diversos dispositivos de hardware, lenguaje de programación, sistemas operativos y diversos clúster de terminales, lo cual implica modelar la topología del hardware sobre el que se ejecuta el sistema. Las funcionalidades se distribuyen de acuerdo a los papeles que jueguen los diversos dispositivos y la interrelación que se establezca de acuerdo a las condiciones establecidas anteriormente tanto en los requisitos funcionales como los no funcionales (Jacobson, 2000).

En la ilustración siguiente se muestra el Modelo de Despliegue donde se especifica la estructura esencial que define el funcionamiento del componente generador de logs y reportes para Suria Vision. El nodo

representa el dispositivo donde se instalará el componente, que está integrado por una biblioteca, la cual contiene toda la responsabilidad de implementar todas las funcionalidades que le dan solución a los requisitos funcionales y una interfaz que permite interactuar con el usuario para cumplir con los requisitos de generar los reportes con los datos cifrados que se encuentran almacenados dentro de ficheros. La computadora es la encargada de albergar el producto, que al ser instalado no necesita de otros dispositivos para su correcto funcionamiento, solo necesita que esté instalado Suria Gestor y el paso continuo a ese es invocar la biblioteca para que puedan ser instanciadas las clases que contiene.

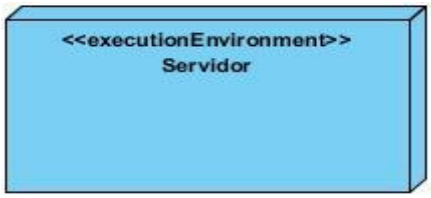


Ilustración 7: Diagrama del Despliegue

3.4 Modelo de Implementación

El Modelo de Implementación permite describir como todos los elementos del diseño se implementan en términos de componentes, por lo que describe la organización de estos de acuerdo a mecanismos, estrategias de modularización que se encuentran disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen unos de otros. En este flujo de trabajo se implementan las clases y objetos en ficheros fuente, binarios y ejecutables. Se prioriza realizar pruebas de unidad, definiendo que cada implementador es responsable de testear las unidades que produzca. Definir la organización del código, en términos de los subsistemas de implementación organizados en capas y modelos entendibles para cualquier usuario que revise los artefactos generados durante dicha fase. El resultado final de este flujo de trabajo es un sistema ejecutable (Ivar Jacobson, 2000).

A continuación se explican los paquetes representados, los componentes que contiene y las relaciones entre ellos:

- **Marco de trabajo QT:** En este paquete se encuentran los siguientes componentes **QTCore** es el núcleo de todas las funcionalidades y métodos que incluye el QT y **QTGui** es el que se encarga de todos los eventos relacionados con formularios e interfaces para representar y visualizar datos.
- **Biblioteca:** En este paquete se encuentran los principales componentes y las dependencias entre estos, los cuales conforman la biblioteca para el componente generador de logs y reportes.

- **Reportes:** En este paquete se encuentran los principales componentes y las dependencias que existen entre estos, los cuales conforman la herramienta que pertenece al componente generador de logs y reportes.

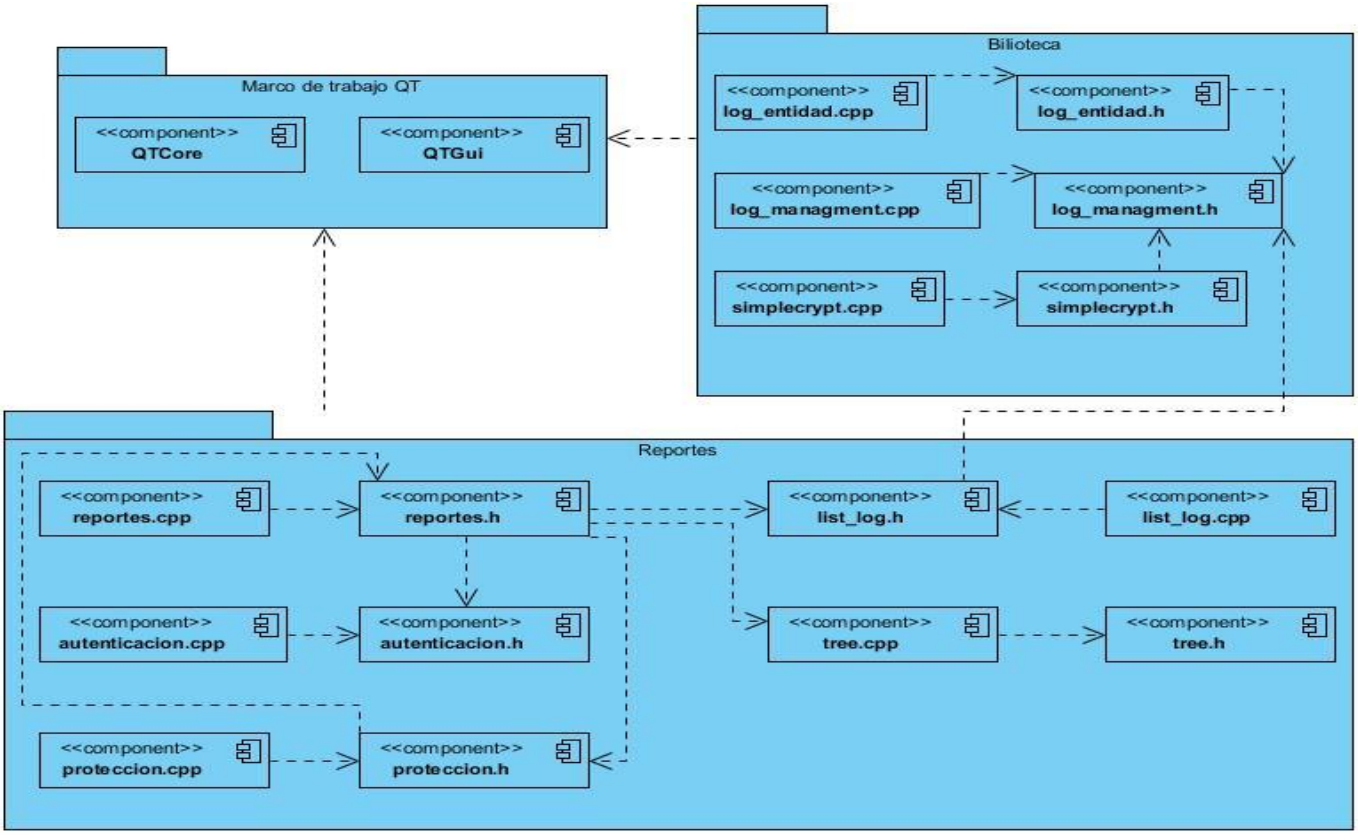


Ilustración 8 Diagrama de Componentes

3.5 Pruebas de la solución

Al terminar y haberse generado el código fuente es necesario comenzar con las pruebas al componente, donde se cumple los objetivos de revisar todo lo diseñado e implementado, para descubrir y corregir la mayor cantidad de errores antes de entregarse al proyecto. Se utilizan para eso diversas técnicas de prueba del software que se encarga de comprobar las interfaces, la lógica del negocio, las colaboraciones entre los componentes y si se dio cumplimiento a los requisitos funcionales. Las pruebas caracterizan el control que se enmarca en comprobar el correcto funcionamiento y las posibles respuestas que el producto despliega antes posibles situaciones. Son básicamente un conjunto de actividades dentro del desarrollo de software, dependiendo del tipo, estas actividades podrán ser implementadas en cualquier

momento de dicho proceso de desarrollo. Son una actividad más en el proceso de control de calidad en todas las aplicaciones (Pressman, 2007).

En el flujo de trabajo de la prueba se verifica el resultado de la implementación, al probar la calidad y el resultado obtenido en cada construcción, incluyendo tanto las internas, las intermediarias y las versiones finales, que se destinan para ser entregados a terceras personas. Se realizan con dichos objetivos:

- Planificar las pruebas o los casos de pruebas necesarias para cada iteración, incluyendo las pruebas de integración y las pruebas de sistemas.
- Diseñar e implementar las pruebas creando los casos de pruebas para especificar lo que se va a probar, a partir de los procedimientos que describe la forma a realizar y creando si es posible componentes de pruebas ejecutable para automatizar dicho proceso.
- Realizar dichos procedimientos antes mencionados y manejar los resultados que traiga como consecuencia dichas acciones. Realizando reparaciones y otros casos de pruebas a las construcciones que se le detecten errores o fallos (Ivar Jacobson, 2000).

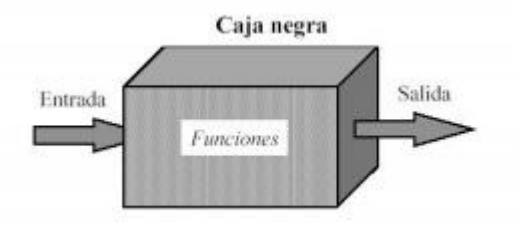


Ilustración 9: Método Caja Negra

El método escogido para realizarle las pruebas al componente generador de logs y reportes es el de la **caja negra** conocidas también como pruebas de comportamiento, su objetivo fundamental radica en comprobar el cumplimiento de los requisitos funcionales del sistema. Mediante un caso de prueba realizado al interfaz del software, permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. Dicho procedimiento no es una alternativa a las técnicas de prueba de la Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados. Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de los estos casos y que permitan que el sistema se ejecute en todas sus variantes.

Específicamente dentro de este método se utilizará la técnica de **Partición Equivalente**, la cual divide el dominio de entrada de un programa en clases de datos, a los que se le pueden aplicar los diversos casos de pruebas y se basa en una evaluación de las clases de equivalencia para una determinada condición de entrada. En esta técnica se plantea que una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada, los estados representan los parámetros de entrada que representan las informaciones necesarias para el cumplimiento de los métodos que se refiere al cumplimiento de los requisitos funcionales, a los cuales se les aplican distintas directrices basado en los casos de pruebas definidos para cada elemento de datos del campo de entrada a desarrollar. Los casos se seleccionan de forma que ejerciten el mayor número de atributos de cada clase de equivalencia a la vez.

A continuación se muestran las pruebas realizadas a los casos de uso del sistema.

Diseño de las pruebas al sistema

CUS Autenticar

Descripción general:

El caso de uso inicia cuando el actor introduce el usuario y la contraseña, la información insertada es verificada en un fichero, de ser correctos los datos insertados el sistema habilita sus funcionalidades, según los permisos del usuario autenticado que pueden ser de administración, de lo contrario niega el acceso a la aplicación y brinda la posibilidad al usuario de volver a autenticarse.

Condiciones de ejecución:

Inicializar el software.

SC1: Autenticar usuario

Escenario	Descripción	Usuario	Contraseña	Respuesta del sistema	Flujo central
EC 1.1 Autenticar usuario correctamente	Se inserta el usuario y la contraseña en los campos correspondientes, el	**V** Administrador	**V** Suria_Vision12@	Si el usuario es correcto habilita la interfaz y se le	Accede a la aplicación o herramienta para visualizar

	sistema verifica que los datos sean correctos, de ser así permite el acceso al sistema con los permisos establecidos según los datos proporcionados.			da inicio a la aplicación.	los logs, ejecutando dicha aplicación/ Llena los campos de textos usuario y contraseña/ Botón "Aceptar"
EC 1.2 Autenticar usuario incorrectamente.	Si el usuario insertado no coincide con la contraseña proporcionada se muestra un mensaje de advertencia: "Usuario o contraseña no válidos". Si se deja algún campo vacío el sistema muestra el siguiente mensaje " No se permiten campos vacios" y se niega la entrada al sistema. Se ofrece la oportunidad de volver a introducir los datos.	**V** Administrador	**I** prueba	No se permite el acceso al sistema, brinda la opción de insertar los datos nuevamente.	Accede a la aplicación o herramienta para visualizar los logs, ejecutando dicha aplicación/ Llena los campos de textos usuario y contraseña/ Botón "Aceptar"
		I vacio	**V** normal		
		I vacio	**I** vacio		

Tabla 6: Escenario CUS Autenticar

Descripción de las variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Usuario	Campo de texto.	No	Solo permitirá letras, números y guión bajo.
2	Contraseña	Campo de texto.	No	Acepta cadena de caracteres de no menor de 8 caracteres que contengan al menos una mayúscula, una minúscula, un número y un carácter especial.

Tabla 7: Variables CUS Autenticar

Las pruebas realizadas a los casos de uso del sistema Visualizar Logs y Crear Logs se encuentran en el expediente de proyecto.

Se realizaron dos iteraciones de pruebas en las cuales se encontraron una serie de inconformidades que fueron corregidas. En la primera iteración surgieron circunstancias adversas en las repuestas que el sistema debía arrojar ante situaciones de fallo, corrigiéndose con las diversas implementaciones de mensajes para informar de lo sucedido a cualquier administrador o personal que utilice en el futuro el componente para la generación del logs y reportes. En la segunda iteración se comprobó que todo los problemas antes descrito se habían solucionados y se mostró un índice satisfactorio de resultados ante diversas pruebas, por lo que se decidió no realizar otra iteración y solo mejorar la respuesta del sistema ante posibles combinaciones de entrada erróneas. La acción realizada demuestra la viabilidad y el correcto funcionamiento del componente, probándose con el método de la **caja negra** que se cumplió el objetivo de los requisitos funcionales.

3.6 Conclusiones

- Utilizando como ejemplo la biblioteca que existe en plataforma privativa se logró definir la distribución física que representará el hardware utilizado para la implementación y ejecución del componente generador de logs y reportes, obteniéndose el diagrama de despliegue donde se visualiza el ambiente de ejecución.

CONSTRUCCIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

- Con el objetivo de describir los componentes fundamentales y sus relaciones, se obtuvo el Diagrama de Componentes, permitiendo describir los elementos fundamentales que participan en el entorno de desarrollo.

- Se comprobó mediante la realización de pruebas al producto, la viabilidad y el funcionamiento, enfocándose en aspectos fundamentales como: las respuestas que brinda el sistema ante entradas correctas e incorrectas, los flujos centrales que sirven de camino para lograr el escenario deseado y las descripciones de las variables fundamentales que intervienen en flujo de ejecución de cada operación.

CONCLUSIONES

Durante el desarrollo de la presente investigación se ha arribado a las siguientes conclusiones:

- Se comprobó mediante la investigación realizada, la importancia que contienen los sistemas generadores de logs y reportes para la mayoría de las aplicaciones informáticas, obteniéndose un conjunto de documentos y bibliografías que respaldan dicha afirmación.
- Se cumplieron los objetivos planteados durante la fase de investigación, principalmente la idea a defender, dando solución a la problemática existente en el proyecto Video Vigilancia y obteniéndose un componente generador de logs y reportes que permite administrar los sucesos generados por Suria Vision.
- Se obtuvo un conjunto de documentos y artefactos que permiten describir todo el ciclo de desarrollo experimentado durante la construcción del componente, almacenando las operaciones principales y las descripciones de todas las funcionalidades.
- Al realizarse dos iteraciones de pruebas, utilizando el método de la **caja negra** se comprobó la viabilidad del componente y las posibles acciones a realizar antes situaciones de fallos, logrando medir el correcto funcionamiento de las funcionalidades que responden ante los requisitos planteados y por la técnica de partición equivalente el flujo que el producto despliega antes entradas incorrectas y operaciones de fallos.
- El componente fue implementado utilizando herramientas multiplataforma cumpliendo las políticas de migración a software libre establecidas por el país y la UCI. La plataforma utilizada no requiere que se pague las licencias por su uso, por lo que es más económico para el proyecto y para la universidad en sí.

RECOMENDACIONES

Al concluir el componente y haber cumplido los objetivos trazados se plantean las siguientes recomendaciones:

1. Se le podrá incluir nuevas estructuras y datos a guardar por el sistema, ampliando su campo de acción.
2. Se le podrá implementar un nuevo algoritmo para encriptar la información.
3. Se le podrá definir otros criterios de búsqueda de datos, para que sean visualizados organizadamente por los usuarios o los administradores.
4. Podrá ser utilizado por otros proyectos pertenecientes al centro de producción GEYSED, que necesiten administrar los sucesos que generan.
5. La biblioteca que contiene el componente, podrá ser incluida en cualquier otra aplicación que se cree para visualizar los datos cifrados de los sucesos.

TRABAJOS CITADOS

Addison Wesley Longman, Upper Saddle River. 1992.*Un acercamiento a través de los casos de uso.* Nueva Jersy : s.n., 1992.

Ariel Rabkin, Wei Xu and Avani Wildani, Armando Fox, David Patterson, Randy Katz. 2012. USENIX. *A graphical representation for identifier structure in logs.* [Online] febrero 13, 2012. [Cited: noviembre 10, 2012.] http://static.usenix.org/events/slaml10/tech/full_papers/Rabkin.pdf.

Carlos Reynoso, Nicolás Kiccillof. Marzo de 2004.*Estilos y Patrones en la Estrategia de Microsoft.* BUENOS AIRES : s.n., Marzo de 2004.

Chiavenato, Idalberto. 2004.*Introducción a la Teoría General de la Administración.* s.l. : McGraw-Hill Interamericana, 2004. Séptima Edición.

CLM. 2011. CLM. *Administración de Logs.* [Online] 2011. [Cited: noviembre 8, 2012.] http://www.clm.com.co/soluciones/administracion-registros.htm.

Corner Bowl Software. 2012. Cornerbowl. *Centralized Event Log Management and Server Monitoring.* [Online] 2012. [Cited: noviembre 29, 2012.] http://www.cornerbowl.com/.

Dambrosio. 2010.*El concepto de datos.* [Online] 2010. [Cited: noviembre 11, 2012.]

Definición ABC. 2007-2013. Definición ABC. *Definición ABC.* [Online] 2007-2013. [Cited: noviembre 12, 2012.] http://www.definicionabc.com/.

Digia Oyj. 2013. QT. *QT.* [Online] 2013. [Cited: febrero 1, 2013.] http://qt.digia.com/.

Ecured. 2013. Ecured. [Online] 2013. [Cited: enero 30, 2013.] http://www.ecured.cu/index.php/C%2B%2B.

EcuRed. 2013. EcuRed. *Visual Paradigm.* [Online] 2013. [Cited: febrero 1, 2013.] http://www.ecured.cu/index.php/Visual_Paradigm.

eIQnetworks, Inc. 2012. Eiqnetworks. *Eiqnetworks.* [Online] 2012. [Cited: diciembre 1, 2012.] http://www.eiqnetworks.com/.

2013. Entorno Virtual de Aprendizaje. *04_Parte_IV_Fase_del_Diseno_1.* [Online] 2013. [Cited: abril 1, 2013.] http://eva.uci.cu/file.php/161/Documentos/Materiales_basicos/Materiales_basicos_de_la_Unidad_3/UML_y_Patrones/04_Parte_IV_Fase_del_Diseno_1_.pdf.

TRABAJOS CITADOS

ERIKA CAMACHO, FABIO CARDESO, GABRIEL NUÑEZ. ABRIL – 2004.*ARQUITECTURAS DE SOFTWARE.* ABRIL – 2004.

2013. eva.uci.cu. *UML_y_Patrones.* [Online] 2013. [Cited: febrero 28, 2013.] http://eva.uci.cu/file.php/161/Documentos/Materiales_basicos/Materiales_basicos_de_la_Unidad_3/UML_y_Patrones/04_Parte_IV_Fase_del_Diseno_1_.pdf.

Head Office, Hagfors, Sweden: Klarälvdalens Datakonsult AB . 2012. Kdab. *Kdab Reports.* [Online] 2012. [Cited: diciembre 1, 2012.] kdab.com.

Ipswitch. 2012. WhatsUpGold. *WhatsUp Log Management.* [Online] 2012. [Cited: noviembre 28, 2012.] http://www.whatsupgold.com/log-management/tools.aspx.

Ivar Jacobson, Grady Boochy, James Rumbaugh. 2000.*El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educación., 2000.

Jacobson, Ivar, Boochy, Grady, & Rumbaugh, James. 2000.*El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educación, 2000.

Lanzillotta, Analía. 2012. Mastermagazine. *Definición de Lenguaje de programación.* [Online] 2012. [Cited: enero 30, 2013.] http://www.mastermagazine.info/termino/5560.php.

Larman, Craig. 1999.*UML y Patrones.* México : s.n., 1999.

Mañas, José A. 2006. Log: trazas de ejecución. [Online] febrero 5, 2006. [Cited: noviembre 10, 2012.] http://www.lab.dit.upm.es/~lprg/material/apuntes/log/log.htm.

Martinto, Pedro Carlos Perez. eva.uci.cu. *La observación.* [Online] [Cited: noviembre 14, 2012.] http://eva.uci.cu/file.php/104/Tema_3/Bibliografia/La_Observacion_-_Tema_3.pdf.

2012. Mastermagazine. *Mastermagazine.* [Online] 2012. [Cited: noviembre 1, 2012.] http://www.mastermagazine.info/termino/6510.php#ixzz2BJkA9hbX.

Mastermagazine. 2012. Mastermagazine. *Definición de C++.* [Online] 2012. [Cited: enero 30, 2013.] http://www.mastermagazine.info/termino/4124.php.

Microsoft. 2005. MSDN. *Cómo ver registros de sucesos para el servicio Integration Services con el Visor de sucesos.* [Online] 2005. [Cited: noviembre 12, 2012.] http://msdn.microsoft.com/es-es/library/ms137978%28v=sql.90%29.aspx.

TRABAJOS CITADOS

Pallavicini, Julio. 2012. Business Transformation The GBM Journal(Edición 53). *Business Transformation The GBM Journal.* [Online] Marzo/Mayo 2012. [Cited: noviembre 15, 2012.] http://www.gbm.net/bt/bt53/tendencias/video-vigilancia-analitica.php.

Pressman, Roger. 2002.*Ingeniería de Software, un enfoque práctico.* s.l. : McGraw-Hill Companies, 2002.

—. 2007.*Ingeniería de Software: un enfoque práctico.* Nueva York : 6ta Edición. Editorial McGraw-Hill. , 2007.

Rational. 2007. Rational. [Online] 2007. [Cited: enero 30, 2013.] http://www.rational.com.ar/herramientas/rup.html.

Rivera, Javier Fernández. 2012.*Ficheros.* [Online] 2012. [Cited: noviembre 8, 2012.] http://aurea.es/wp-content/uploads/ficheros-archivos.pdf..

Roger Pressman. 2013. eva.uci.cu. *Modelado del Análisis parte 1.* [Online] 2013. [Cited: febrero 28, 2013.] http://eva.uci.cu/file.php/161/Documentos/Materiales_basicos/Materiales_basicos_de_la_Unidad_3/Pressman_6ta_Edicion/Pressman_Cap_08_Modelado_del_Analisis_Parte_1.pdf.

Sanabria, Cesar Santos.*Geysed-VV-0120_6_ArqSWVtaEntornov1.0.* Ciudad Habana : s.n.

Silva, Reinaldo de Oliveira Da. 2002.*Teorías de la Administración.* s.l. : International Thomson Editores, S.A. de C.V., 2002.

SIPEC WEB. 2008. Reportes. *Reporte.* [Online] marzo 11, 2008. [Cited: noviembre 11, 2012.] http://sipec.sep.gob.mx/WebHelp/reportes/reporte.htm.

SolarWinds Inc. 2012. Solarwinds. *Solarwinds.* [Online] 2012. [Cited: diciembre 1, 2012.] http://www.solarwinds.com/.

Somerville, Ian. 2005.*Ingeniería de Software.* Madrid : Séptima Edición, 2005.

Stephen Robbins, Mary Coulter. 2005.*Administración.* s.l. : Pearson Educación, 2005. Octava Edición.

Terreros, Julio Casal. 2012. Msdn. *Desarrollo de Software basado en Componentes.* [Online] 2012. [Cited: noviembre 10, 2012.] http://msdn.microsoft.com/es-es/library/bb972268.aspx#mainSection.

Tripwire, Inc. 2013. Tripwire. *Tripwire.* [Online] 2013. [Cited: enero 4, 2013.] http://www.tripwire.com/it-security-software/log-event-management/log-management/.

Video Vigilancia. 2012.*Suria Vision.* 2012.

ANEXO

Anexo 1 Entrevista realizada al jefe de proyecto y a una población de integrantes del proyecto Video Vigilancia.

1. ¿En qué plataforma está implementada la aplicación que permite generar logs con los sucesos que despliega actualmente Suria Vision?
2. ¿Cuáles son los tipos de sucesos que se generan en Suria Vision?
3. ¿Que facilidades o ventajas brindará tener en la versión actual un producto que cumpla con los objetivos de administrador los sucesos antes mencionados?
4. ¿En qué tipo de fichero se guardan actualmente los datos de dichos sucesos?
5. ¿El proceso para cifrar y descifrar los datos de los sucesos con qué objetivos se realiza?
6. ¿En qué forma se visualizará los datos de los sucesos cifrados y en qué tipo de archivo deberá guardarse?
7. ¿Cuántas personas trabajarán con el componente para la generación de logs y reportes?
8. ¿Qué criterios de búsquedas se necesita para visualizar los datos de los sucesos de forma simple y rápida?
9. ¿Cuales son los datos que debe administrador el componente generador de logs y reportes?
10. ¿Qué reportes son los que necesitan visualizar?
11. ¿En qué formato desean generar los informes?
12. ¿Por cuáles criterios desea que se realice la búsqueda?

GLOSARIO

Software: Conjunto de instrucciones y datos codificados para ser leídas e interpretadas por una computadora. Estas instrucciones y datos fueron concebidos para el procesamiento electrónico de datos. Series de instrucciones codificadas que sirven para que la computadora realice una tarea. Son los programas de la computadora.

Programa: Un programa es un conjunto de instrucciones u órdenes basadas en un lenguaje de programación que una computadora interpreta para resolver un problema o una función específica.

Event Manager: Evento administrador de registros lógicos utilizados en un gran por ciento de los sistemas generadores y administradores de logs.

W3C: Registros generados por servidores de aplicaciones webs, donde se guarda datos de configuraciones y conectores que permiten almacenar los datos de los estados y las notificaciones de las diversas conexiones.

Lenguaje de bajo nivel: Lenguaje predeterminado para la comunicación directa con las computadoras, es el código fuente de la máquina, es decir el que la máquina puede interpretar.